Agent-based Interperability in Telecommunications Applications

Vorgelegt von

Diplom-Informatiker

Tianning Zhang

Vom Fachbereich 13 - Informatik

der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor-Ingenieur

- Dr. -Ing. -

genehmigte Dissertation

Promotionsausschuß:
Vorsitzender: Prof. Dr. –Ing. Günter Hommel
Berichter: Prof. Dr. Dr.h.c. Radu Popescu Zeletin
Berichter: Prof. Dr. rer. nat. Thomas Christaller

Tag der wissenschaftlichen Aussprache: 3. April 2001

Berlin 2001
D83

Abstract

The evolution of the telecommunications technology and market also calls for the evolution of the software design paradigms for the open distributed systems. Especially new frameworks for enhanced interoperability support among telecommunication application resources are required in this context, which should be based on a higher degree of understandings among the distributed systems

Traditional approaches for interoperability, such as RPC or DOT, are based on relatively *syntactical understandings* by pre-specified/agreed, or dynamically exchanged syntaxes of the co-operation interfaces, and rely on relatively static co-operations functionalities. Such a static nature can not meet the requirements of the dynamic, heterogeneous, distributed and mobile telecommunications environments. Among others, such traditional technologies do not support the dynamic and smooth adaptation of the co-operation functionality following the evolution of the business or technological requirements, limit the possibility of offering individually tailored services to the customers and users, and do not offer sufficient support for reliable and robust co-operations within a widely distributed or even nomadic application environment.

To overcome such shortcomings of the traditional paradigms for distributed software design, one level of enhanced interoperability will be needed, which will be based on the knowledge-based *understanding* among the co-operating systems. Knowledge-based understanding in this context means the capability of the co-operating systems to *exchange and interpret the knowledge about the expected behaviors associated to the data or data schemes that are conveyed by the interaction messages between the systems*.

Agent technology can be considered here as the new software design paradigm which supports such knowledge-based interoperability among autonomous distributed applications or application environments.

The main objective of this thesis is to develop an agent-based solution for enabling and utilizing knowledge-based interoperability among telecommunications service applications. As the main features, this solutions will be based on the agent framework and architecture which

- integrate the standard conform technologies for both speech act based agent intelligence and agent mobility, based on the FIPA and OMG MASIF standardisation,

- adopt an object, rule and Web-oriented knowledge representation and management framework which support enhanced reusability, accumulative construction and dynamic adaptation of the agent co-operation intelligence,

– define an agent template that implements the elementary rational and social behaviors of an agent in the provisioning and management of telecommunications services, and use such an agent template to support the flexible, adaptive co-operations among autonomous and heterogeneous applications.

For validating the solution, the new approach is applied to a dynamic VPN service application scenario, where agent-based interoperability can help to enable the dynamic customization and adaptation of services, and also to enhance reliability, robustness of service co-operations.

Zusammenfassung

Die Evolution der Telekommunikationstechnologie und des Telekommunikationsmarkts stellt neue Anforderungen an die Paradigmen für Softwareentwicklung von offenen und verteilten Systemen, insbesondere an neue Technologien zur Interoperabilität und Kooperation zwischen Telekommunikationsressourcen. Solche Technologien müssen ein höheres Niveau der Verständigung zwischen verteilten Systemen unterstützen.

Herkömmliche Ansätze für die Interoperabilität zwischen verteilten Systemen, sowie die RPC- oder DOT-Technologien stützen sich auf syntax-orientiertes Verstehen auf Basis von vor-definierten bzw. vor-vereinbarten oder dynamisch ausgetauschter Syntax der Kooperationsschnittstelle, und daher auf relativ statische Kooperationsfunktionalitäten. Dieses Merkmal der statischen Kooperation von herkömmlichen verteilten Systemen kann den Anforderungen nicht entgegenkommen, die von der dynamischen, heterogenen, weltweit verteilten und mobilen Telekommunikationsumgebung gestellt werden. Zu einem unterstützen solche herkömmlichen Technologien nicht die dynamische und reibungslose Adaptierung der Kooperationsfunktionalität, welche wegen der sich verändernden geschäftlichen oder technologischen Bedingungen benötigt wird. Zu anderem haben solche Technologien nur die eingeschränkte Möglichkeit, den Kunden individuell angepasste Dienste anzubieten. Darüber hinaus haben die statischen und unflexiblen Kooperationsbeziehungen innerhalb einer weitverteilten und nomadischen Anwendungsumgebung einige Nachteile in Bezug auf Zuverlässigkeit und Robustheit der Kooperation von verteilten und mobilen Ressourcen.

Um solche Mängel der herkömmlichen Softwareparadigmen für verteilte Systeme zu überwinden, benötigt man Interoperabilität zwischen verteilten Ressourcen auf einer höheren Ebene. Diese Ebene stützt sich auf das wissensbasierte Verstehen unter kooperierenden Systemen. Unter wissensbasiertem Verstehen versteht man in diesem Zusammenhang die Fähigkeit das Wissen über das erwartete und assoziierte Verhalten der Daten bzw. Datenschemata, die durch Interaktionsnachrichten übermittelt werden, auszutauschen und zu interpretieren.

Agententechnologie wird in dieser Arbeit als ein neues Paradigma für Softwareentwicklung betrachtet, das wissensbasierte Interoperabilität unter autonomen verteilten Anwendungen bzw. Anwendungsumgebungen unterstützt.

Das Hauptziel dieser Arbeit ist die Entwicklung einer agenten-basierten Lösung zur Unterstützung und Nutzung wissensbasierter Interoperabilität zwischen Anwendungen und zur Bereitstellung und Management von Telekommunikationsdiensten. Diese Lösung basiert auf einen Agentenparadigma und einer Agentenarchitektur mit den folgenden Merkmalen:

– Integration der sprechakt-basierten Agentenintelligenz und Agentenmobilität auf Basis der FIPA und OMG MASIF Standardisierungen.

– Anwendung eines Objekt-, Regel- und Web-orientierten Paradigmas zur Wissensrepräsentation und -management, das die erhöhte Wiederverwendbarkeit, den schrittweisen Aufbau und die dynamische Adaptierung der Kooperationsintelligenz der Agenten unterstützt.

– Definition einer Agentenschablone, die elementares, rationales und soziales Verhalten zur Bereitstellung und Management von Telekommunikationsdiensten implementiert. Diese Agentenschablone wird eingesetzt, um die flexible und adaptive Kooperation zwischen autonomen und heterogenen Anwendungen zu unterstützen.

Um die vorgeschlagene Lösung zu validieren, betrachten wir als Szenario das Management eines dynamischen VPN Dienstes, wobei die agentenbasierte Interoperabilität die dynamischen Anpassungs- und Adaptierungsfähigkeiten von Dienstressourcen ermöglicht, und gleichzeitig die Zuverlässigkeit und Robustheit der Kooperation zwischen den beteiligten, autonomen Akteuren erhöht.

Contents

CHAPTER 1

Purpose and Motivation

1.1. Background

With the ubiquity of distributed open systems in the information society and in the telecommunications market, the focus of computer science is now moving from *computations* to *co-operations* where *Interoperability* among distributed systems has always been the dominant issue.

Interoperability in this context can be regarded as the capability of a system to co-operate with other systems in the heterogeneous environments. Within an open and distributed telecommunications environment, interoperability determines the degree of usability and reusability of the application solutions. Basically speaking, a highly interoperable solution can be

- deployed in a wider range of heterogeneous application environments, and
- be reused in a variety of contexts to build up more complicated applications for different application purposes.

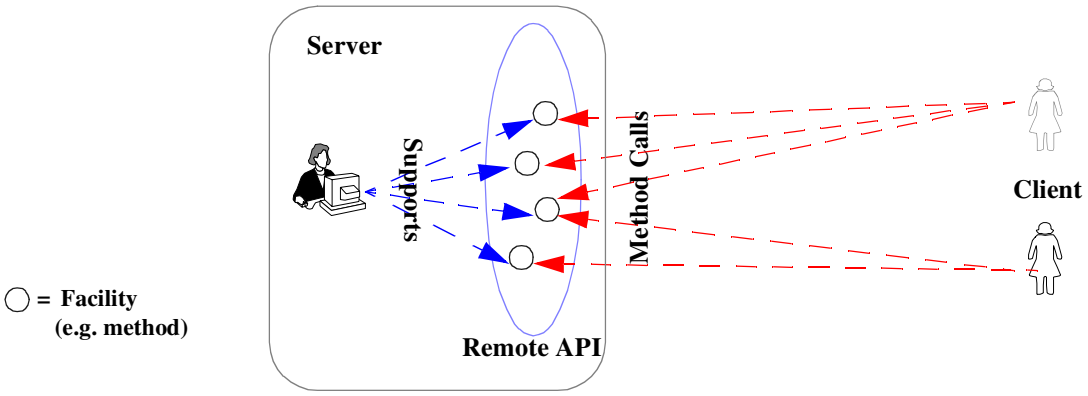
As a result, a higher degree of interoperability plays the key role in realizing cost-effective telecommunications solutions and therefore in increasing their chance of success in the open market. The core history of studies in open, distributed systems is in fact a history of developing new and enhanced support for the interoperability among distributed and heteroge-

neous solutions.

Traditional frameworks for supporting the interoperability among distributed and open systems are mainly based on Remote Procedure Call (RPC), client/server technologies or Distributed Object Technologies (DOTs), with RPC as the basic mechanism for co-operating the distributed resources.

A RPC paradigm can be abstractly depicted in figure 1.

Figure 1: RPC Paradigm



Within a RPC-based paradigm, the server entity supports a pre-specified remote API with a set of defined facilities (object classes, methods, attributes etc.). The clients can utilize such facilities, e.g. via remote method calls, to access the functionality provided by the server. The DOT extends the client/server paradigm by adopting the concept of distributed objects which typically play both the roles of client and server in different co-operation contexts.

Interoperability in this context is enabled via the shared API definition, which is agreed among the co-operating clients/server and supported by the corresponding applications. The API definition determines the *syntax* of all the interactions (methods, parameters) between the server and clients applications.

A number of standardization frameworks have been developed in this context for enabling the RPC-based interoperability among heterogeneous applications or application resources within the open environments. Most important examples of such standardization frameworks include ITU-T TMN/CMIP [53], IETF/SNMP[49], OMG CORBA[70], TINA [94]. The key components in such standards are the various formal frameworks supporting the definition of a RPC-based API, e.g. ANS.1[52], TMN/GDMO [55], SNMP/SMI[49], CORBA IDL[72], TINA ODL[92].

The API definitions in these frameworks can usually be complied to automatically generate stub/skeleton codes for the partial client/server implementation. Such stubs/skeletons can be used/extended by the client/server or DOT applications to implement the co-operations via

the selected RPC protocol.

To modify the API, we will usually have to re-compile the API definitions, rewrite the client/server or DOT programs based on the new stubs/skeletons and then re-install/update the affected software components. The associated costs, complexity and the intrusion in the normal operations of the distributed systems make it relatively difficult *to dynamically modify the APIs (and the associated server/client co-operation functionality) in a RPC-based paradigm*.

As the result of such syntax-based interoperability support, RPC-oriented software design paradigms for distributed systems rely on relatively static co-operations interfaces and functionalities.

Moreover, within a complex and versatile application environment, a RPC API design has to support all the possible services that can be requested by the versatile clients in the operational environment. At the same time, the complexity of such APIs should be kept at minimal in order to reduce the complexity of the server and client applications, and to optimize the resource usages at the client/server network nodes.

To satisfy these two requirements at the same time, and due to the static nature of RPC APIs, the only possible approach is to maximize the reusability of the RPC APIs by offering elementary, i.e. *fine grain* service functionalities to the clients. Different clients can then combine such fine grain facilities by aggregated calls, and can obtain in this way the services required by their own obligations in the environment.

Fine grain interoperation typically results in higher number of remote interactions and the dependency on the constant availability of the underlying network resources for realizing the distributed co-operations. Moreover, such fine granularity also relies on the centralized client's intelligence in composing the required functions, and typically results in centralized scheme in managing and operating the distributed resources.

All these features impose some limitations on the traditional software design paradigms within the dynamic, versatile and distributed telecommunications environment.

The world of telecommunications is now undergoing dramatic changes, which are bringing new possibilities and challenges to the traditional frameworks and technologies for the provisioning and management of telecommunications services.

Such changes are mainly driven by the following factors:

- *progress* in communication and application technologies

The telecommunications industry is experiencing a fascinating speed in the development of communication and computing technologies. New technologies, including the broadband ATM/SDH, high performance multimedia equipment, mobile and satellite telecommunications infrastructures, are changing thoroughly the face of telecommunications. This progress in the new technologies has two kinds of direct effects on the telecommunications industry.

On the one hand, rapid emergence of new communication and computation technologies means

- rapidly extending capabilities of the communication and computation resources,
- increased flexibility and complexity of the communication and computation resources, and the new possibilities (e.g. dynamic QoS management) for management operations,
- short lifetime of existing service and service resources, i.e. frequent updates, replacements and extensions of the telecommunication services or their components and functionalities.

On the other hand, the increased capability of the communication and computation technologies and the new technical possibilities enable a large class of new customer applications, especially the multimedia-based and e-commerce oriented telecommunications applications, which were not possible with traditional technologies. These new customer applications also impose new, advanced, heterogeneous and rapidly evolving requirements on the existing telecommunications infrastructures and their services.

- ***deregulation* of the telecommunications market**

Until recently, and in some cases even now, telecommunications markets are dominated by state owned telecoms, which have hosted both the roles of network providers and service providers. In this context, the customer gets a closed service with little chance to control or adapt the service features in supporting his own business activities, e.g. in Value-Added Service (VAS) provisioning. Instead, customers of the traditional telecoms services are typically end-users. Such monopoly in the market hampers the development of the market but also results in a relative simple and static market structure.

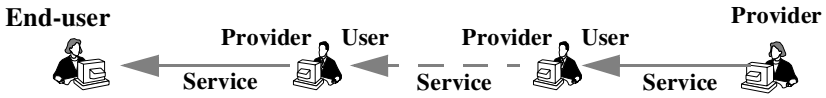
The *deregulation* of the telecom market, which is almost finished in some countries and scheduled in some other countries for the near future, is going to change thoroughly this picture. As identified in the emerging Global Information Infrastructures (NII[65], EII[18], GII[43]), a deregulated telecommunications market is creating a large number of new business sectors and new players, with heterogeneous business requirements that evolve based on the dynamic market demands. This versatility and the dynamic nature of the customers and customer requirements demand a flexible, easily/dynamically adaptable and extensible telecommunications service infrastructure. Most important in this context is the demand for rapid, dynamic and smooth creation, adaptation and integration of new telecommunications services or service features.

Within the deregulated telecommunications market, telecommunications services are typically provided to the consumers via complicated and dynamic infrastructures of value chains,

as depicted in figure 2, where a player/an actor within a value chain for a specific business case can have both the roles of

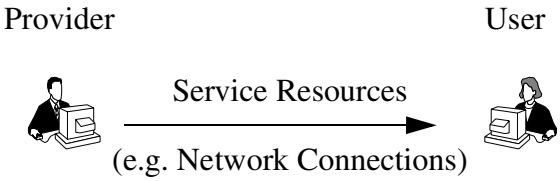
- a provider (or Value Added Service Provider - VASP)
- and a user (which can be an intermediate user, a customer that represents a group of users in a service environment, or an end-user, i.e. a consumer that terminates a value chain).

Figure 2: Value Chain in the Information Infrastructure



The elementary type of relationship in this context is therefore the user/provider relationship within a service environment, as depicted in figure 3, via which the relevant service resources are offered to the users.

Figure 3: Provider/User Relationship in the Telecommunications Services



The developments in the telecommunications market result in many new features in the user/provider relationships for telecommunications services, and correspondingly the new requirements on the design paradigms for interoperability among open, distributed systems. Among others, such relationships have the following characteristics:

– dynamic nature

The functions provided via the user/provider co-operation relationships are of dynamic nature in the sense that such functions should evolve following the changes in the business environment and in the administrative, technological or operational requirements. Moreover, the users of telecommunications services can frequently change their providers, especially via newly negotiated service contracts, in order to optimize the users' performance within their own business contexts. Therefore the service relationships have only temporal mean-

ings.

– heterogeneity and complexity

Each player in the telecommunications services can participate in multiple value chains and in different business contexts. Each user can utilize a large number of services in composing his own telecom service and can have therefore relationships to a large number of provider players or their resources. At the same time each provider will offer services to a large group of users. The heterogeneity of the players and their service requirements/features or technologies contribute to the complexity of the inter-relationships.

– distribution and mobility

With the globalization of the telecom market, the service resources and the players participating in the user/provider relationships will have a significantly wider administrative or geographical distribution compared to a traditional environment. One example in this context is the global Internet and WWW-based information infrastructure, where the users, customers, providers and the resources are distributed all over the world. Moreover, the players (e.g. mobile hosts or travelling users) can migrate dynamically across geographical or administrative borders as well, resulting in the mobile features of the players, functionalities and resources, and in *nomadic* applications. This mobility further increases the dynamic nature of a global telecommunications environment, and imposes higher demand on the ability to deal with unreliable, foreign and heterogeneous environments.

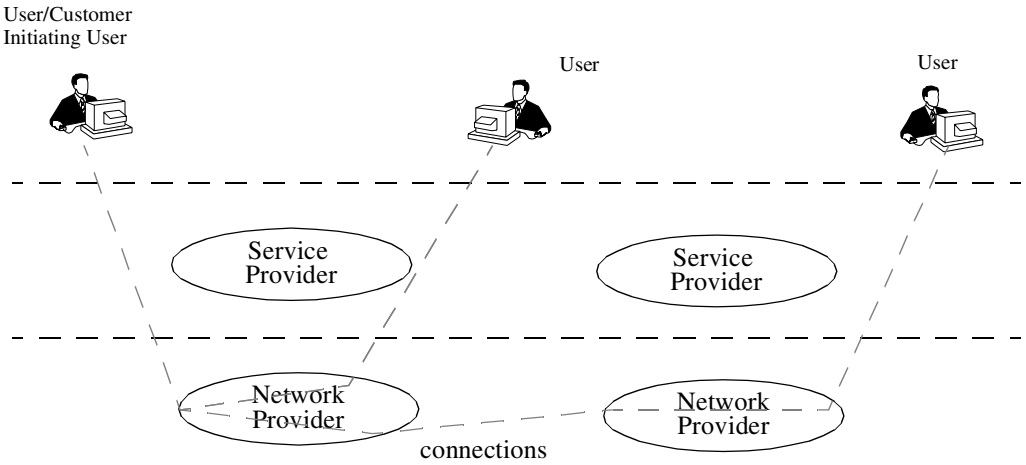
As an example, one important category of telecommunication services in this context, which are called VPN (or multimedia VPN, dynamic VPN) services in the literatures (e.g. [29], [112], [116]), focus on the provisioning and management of bearer transport connection resources for data communications, and support multimedia high level services like multimedia conferences or cooperative works. These services offer the basis for building up other high level telecommunications applications and provide some representative business cases for the study of telecommunications service provisioning and management in general. Many agent-related studies in service management (see [29], [114], [123], [124]) have therefore selected VPN as the business case for validating the added value of agent technology. This thesis, while maintaining the generality of the technical results, will focus its discussion on the application of the agent technology in the VPN-related business contexts.

Within the open and globalized telecommunications environment, a VPN service is typically provisioned in an environment of multiple autonomous and co-operating (and possibly competing) service and network providers as depicted in figure 4.

The global *connections* connecting the users for user applications like multimedia conference typically run through multiple *network providers*, which offer and manage the physical com-

munication resources. I.e. the connectivities will be established and managed via network operator co-operations.

Figure 4: The VPN Infrastructure



The VPN *service provider* in this context is the entity which

- interacts directly with the users or customers in provisioning and managing the VPN service, and
- interacts with the different network providers for the provisioning and management of the physical VPN connections.

The VPN *users* are the end users which deploy the VPN connections for their high level applications. Some users will play the role of *customers* as well. A customer in this context is responsible for negotiating, initiating and managing the VPN service and service features on behalf of all the associated users. Generally speaking any user can play the role of a customer if the user has the sufficient authority.

Users can access the VPN service via some portable equipment like laptops, and can travel around in the global telecommunications environment. As a result, a VPN service will have to deal with mobile users and hosts in its service management.

With the service environment, two kind of user/provider relationships can be identified, i.e.

- the relationship between VPN user and the VPN service provider, and
- the relationship between the VPN service provider and the network provider.

Both relationships will be based on negotiations between autonomous entities and can have a higher degree of the dynamic nature in the open environment.

A user in the environment, which can be himself a VASP in his business context, has typically evolving requirements on the service QoS and deployment strategies. E.g. due to the upgrading of the his local resources, like the support for high speed two-way video display, or after entering new business areas like video-based real-time remote control, a user can require higher bandwidth, higher reliability and even new negotiation strategies. Similarly, a service provider can dynamically change his requirements or his co-operation strategies within his relationship to the network provider.

Moreover, within the open and competing market, both the users and the service providers can dynamically select their providers based on the requirements of individual service sessions. Pre-negotiated and static service contracts are becoming insufficient and too rigid for the determine the service relationship in each session.

One the other hand, due to the rapid innovation of technologies and changing market conditions, the VPN providers (service provider and network provider) can frequently modify

- their service resource technologies (e.g. from ISDN to ATM/SDH)
- management technologies, and
- service policies or co-operation strategies in the user/service provider or service provider/network provider relationships respectively.

As a result, the user/provider relationships in the future VPN services will be of highly dynamic nature.

Besides, the autonomy of the players in a VPN service environment results in the heterogeneity of the

- resource and management technologies,
- policies and co-operation strategies, and
- goals or interests.

To guarantee the interoperabilities among these players, the software design paradigm for the VPN service has to deal with the complexity resulted from heterogeneity and from the sophistication of new technologies.

Finally, mobile hosts and users impose a new requirement on the VPN paradigm and infrastructure. Such a paradigm has to support the mobility of resources and the continuous operations of mobile users in foreign environments.

Traditional distributed system design paradigms based on the RPC mechanism, with their relatively static and inflexible interoperability support based on syntactical interfaces, are now becoming increasingly in-sufficient to meet these new challenges of the emerging open telecommunications environments. Basically, the static nature of the RPC-based interoperability

- makes it very difficult to dynamic adapt the co-operation functionality following the evolution of the telecommunications business environment,
- limits the possibility of offering individually tailored services to the users and
- limits the ability of the software-dependent mobile or nomadic players and resources to migrate to and operate in foreign and heterogeneous environments.

At the same time, the higher communication traffic, the higher dependency on the constant network availability and the centralized operational intelligence scheme associated to the RPC-based solutions further result in

- in-efficiency, un-reliability and lower robustness of the solutions within the widely distributed environments.

The VPN services are nowadays mainly implemented via TMN [53] or SNMP [49] -based service management frameworks. Co-operations among the distributed players (user, service provider and network provider) and their resources are based on pre-defined, and in many cases standardized CMIP/GDMO or SNMP interfaces. Several TMN/SNMP-based implementations of ATM- or SDH-based Pan-European VPN service testbeds are delivered by the European R&D projects ([112], [116]). Due to the static nature of the RPC technology in TMN/SNMP frameworks, current implementation of the VPN services do not provide sufficient support for the dynamic evolution of the service environment.

Interface definitions in this context for user/service provider or service provider/network provider are dedicated to the pre-selected transport technology, its abstraction level and the service co-operation strategy/algorithm. Evolution of the technology (e.g. from SNMP to TMN, or from SDH to ATM) or service algorithm typically result in a totally new R&D project and re-implementing most of the management software.

Service customization is realized via allowing the user (or service provider) to choose from a set of alternative, and pre-programmed service categories and profiles. Due to the complexity of such service profiles, only a limited (3-4 in most cases) number of profiles are supported by the network or service providers. The customization possibility is therefore limited.

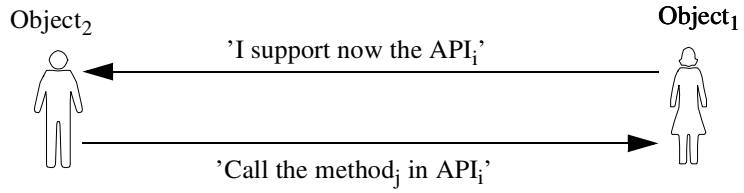
Mobile users or hosts are possible only within identical or strictly compatible (and therefore not really foreign) environments.

Beside these limitations, the TMN/SNMP-based service management framework and its fine

grain RPC paradigm result in higher dependency on the availability of the management network as well, and have the associated in-efficiency, un-reliability and lower robustness in a highly distributed environment.

Currently there are also some extensions or new technologies based on the RPC and DOT frameworks that can help to partially solve the aforementioned problems. One such technology is the Dynamic Invocation (abbreviated as DII or DSI) mechanism which is nowadays supported by almost DOT platforms, including Java and CORBA [70]. The idea behind this technology is to allow the dynamic construction of a remote method call after considering the situations or the states of the program executions. In this way, as depicted in figure 5, one co-operating system object can inform its co-operating partner object via a notification method or a message that it is supporting a specific (possibly new) interface. The co-operating partner can then invoke any methods that are allowed in the specified interface.

Figure 5: Dynamic Invocation



Due to lack of mechanisms for dynamically exchanging *knowledge* about (i.e. the behaviors or meanings associated to) the APIs, the pre-condition for the DII/DSI technology is that both partners are programmed beforehand with the set of APIs and behaviours that can be dynamically selected. Its support for the dynamic and customizable interoperability is therefore limited.

Another major technology, which is becoming mandatory in most RPC- or DOT-based application, is the Object-Oriented paradigms for software development. Within such paradigms generic services or resources are specified, implemented, and reused in different contexts to build up more specialized and complex services and systems. In this way, the Object-Oriented approach can help to ease and accelerate the construction of new services or the adaptation of existing services, and enables e.g. a service creation environment.

The TINA-C service architecture ([93], [94]) and Intelligent Network (IN) framework [62] can be regarded as the representatives in this context. Both focus on the easy/rapid service creation via generic/reusable service building blocks with standardized interfaces. E.g. the IN framework tries to standardize generic telephony-oriented service features [62] that can be used to compose customer-oriented and more complex services.

However neither TINA-C nor IN support the dynamic and run-time adaptation or customization of the service resources and interfaces due to their dependency on the traditional RPC/ DOT paradigms.

1.2. The Evolution of Interoperability among Open Systems

The evolution of the telecommunications technology and market calls for the evolution of the software design paradigms for the open distributed systems. Especially new frameworks for the enhanced interoperability support among telecommunication application resources are required in this context. One possible enhancement should be based on a higher degree of understandings among the distributed systems.

Traditional approaches for interoperability are based on *syntactical understandings* by pre-specified/agreed, or dynamically exchanged (in case of DII/DSI) syntax of the co-operation interfaces. Within the dynamic, versatile and globally distributed modern telecommunications environment, one level of enhanced interoperability will be based on the *knowledge-based understanding* among the co-operating systems. Knowledge-based understanding means in this context the capability of the co-operating systems to *exchange and interpret the knowledge about the meanings or the expected behaviors associated to data that are conveyed by the interaction messages between the systems.*

The resulted interoperability support, which can be called *knowledge-based interoperability*, can play an important role in meeting the requirements of the emerging telecommunications environments. With this capability, a distributed application is in the position

- to dynamically adapt its own co-operation interfaces and functions, and to teach its co-operating partners the behaviors of the new interfaces and how to use such new functionalities,

- to dynamically request the customization of the services and functions offered by the co-operating partners via informing the partners about the required behaviors of the new functionalities,

- to dynamically learn the capability of services and functions in a foreign application environment by searching, retrieving and processing the corresponding definitions for the syntax and knowledge.

These new possibilities can help to overcome many of the shortcomings inherited in the traditional RPC, client/service and DOT software design paradigms. Among others, knowledge-based interoperability enables the

- dynamic evolution of the co-operation relationships following the evolution of the business or technological requirements in the environment,

- dynamic and flexible compositions or customization of services, in order to offer individually tailored services to the versatile users,

- increased reliability and efficiency in a globally distributed environment, by reducing number of remote interactions and by reducing the possibility of bottlenecks or the dependency on the remote network, via the dynamic composition of customer-oriented higher grain service features and via the dynamic distribution of functionalities/intelligence,

- initial and accumulative co-operation between mobile/nomadic resources and the foreign environments via the capability of learning, searching and teaching the syntax/ knowledge of the services and resources in the new environment.

Therefore, knowledge-based interoperability, which is an enhancement of the traditional syntactical interoperability and which typically uses traditional technologies as its basis, provides a better solutions for the emerging telecommunications market.

The evolution of interoperability in telecommunications can be compared to the human communication or interoperability based on natural languages, where

- syntactical interoperability corresponds to the capability of defining the syntax of a language (e.g. English) and of documenting or exchanging such definitions,

- knowledge-based interoperability corresponds to the usage of some specific, elementary terms and expressions with well known/agreed meanings to characterize the knowledge about the meansings of terms or sentences (messages) in the human dialogue, e.g. about how such terms or sentences should be interpreted and used by the audience.

Generally speaking, knowledge-based understanding is enabled via standardized language components (e.g. terms and expressions) for knowledge representations whose meanings are well understood by the co-operation partners like the agents. By interpreting such language components, the receivers are in the position to rationally interpret the knowledge encoded in the messages.

The focus of this thesis will be on the support and utilization of knowledge-based interoperability in telecommunications applications, especially in the context VPN service provisioning and management.

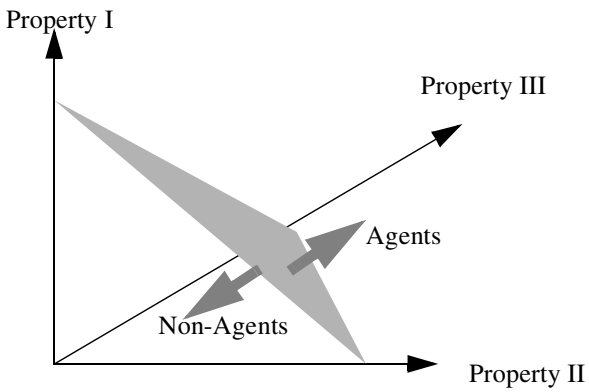
1.3. The Role of Agents Technology in Knowledge-based Interoperability

During the recent years, the concept of *autonomous software agent* (called *agent* hereafter for simplicity) has rapidly becoming an important new design paradigm for the open, distributed software systems, especially in the context of telecommunications applications. Numerous commercial enterprises are offering a wide range of new products which are claimed to be either based on agent technologies or to enable the developments of agent-based applications. Agent based solutions are frequently proclaimed as the key future technologies of the telecommunications market. At the same time, research and standardization organizations (e.g. the European Commission [11], OMG [34], FIPA [22], IETF [49]), with support from the telecommunications industry, are producing new theories, frameworks, technical specifications and new standards in a fascinating speed.

In this context, the agent technology can be considered as the continuation of the trend in shifting the emphasis from computation to co-operation within the computer science ([59]).

Although there is still no clear consensus among the agent society concerning the definition of agent and agent technology, it is generally agreed that agent can be characterized by a set of agenty properties ([10], [40]). In this context, a software system can be positioned within a multi-dimensional co-ordinate system, where the (positive) co-ordinate along one axis represents an individual (*should be*) property for agents (see figure 6). The membership of a software system to *agent* is measured by the its distance to the origin of the co-ordinate system.

Figure 6: The Open Ended Agent Definition



With slight variations, most literatures in this context share the opinion that an agent should be

– autonomous: exercises control over its own actions/algorithms

- social/communicative: communicates with other agents, perhaps including people
- reactive: responds in a timely fashion to changes in the environment
- goal-oriented/pro-active: does not simply act in response to the environment
- learning/adaptive: changes its behavior based on its previous experience
- flexible/tolerant: able to cope with unexpected situations
- character/personality: believable “personality” and mental attitudes/emotional states
- mobile: able to transport itself from one machine to another

There is at the moment no general agreements concerning the exact definitions of these properties, their inter-relationships and their relative weights in determining the agenty of specific software systems. Moreover, most agent systems nowadays do not possess all these properties, and it is very doubtful whether it is at all necessary to have simultaneously all these properties in many application environments.

However, from these properties we can still derive a high level picture of the agent technology and get some idea about its key contributions to the improvement of software design technology, especially the improvement of design paradigms for the telecommunications services.

Roughly speaking, the social capability is the key feature of the agents, while the other properties are in fact the refinement of desirable features in the agents’ social relationships with its environment.

Autonomy in this context is the direct reflection of the versatility of the telecommunications world, where systems and resources in the value chains belong to different players, have different roles and represent their autonomous interests. Such autonomous interests have to be implemented via autonomous goals, intelligence, algorithms and decisions. Autonomy also serves as the basis for implementing all the other agent properties.

Reactiveness is not the privilege of agents, any distributed object or software will react to method calls from the environment in a timely fashion that is allowed by the application context. An agent, however, has to enhance this capability by reacting in a rational way that is optimally adapted to the environment and the goals of the agent.

Different from a distributed object, which also possesses certain primitive goals represented as the autonomous algorithms coded into the object implementation, an agent should support in this context more sophisticated rational behavior for determining and enforcing its goals within the dynamic, distributed environment. E.g. an agent has to be capable of persuading

other agents, and of consistent, rationally responses to its agent environment (social ability and personality characters). Moreover, in the dynamic changing/evolving environment, each agent should have the capability to dynamically adapt its goals (or the interpretation and implementation scheme of such goals) in guiding its interactions with the environment.

Adaptation is based on learning and teaching, which have to be implemented in the interactions with the environment and its agents. An agent has to be in the position of meaningful, rational co-operations (social ability) to obtain useful information for adapting its behaviors. During the learning processes, an agent has to rationally (following the consistent personality characters) adapt its behaviors following its goals and the interpretation of such goals (goal-orientation). Similarly, personality characters in this context have to be reflected in the social behaviors and in an agent's adaptability to the changing environment.

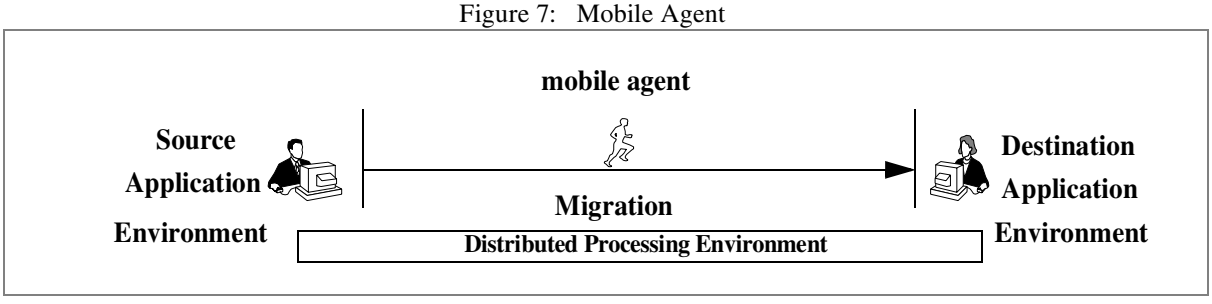
Ability to cope with exceptional/unexpected situations and especially with foreign environment is also closely related to the agent's ability to learn or teach new knowledge, i.e. the required behaviors in the context of such situations. It depends on the agent's ability to exchange (or collect) such knowledge to or from the environment.

Mental and emotional behaviors of the agents simulate the co-operation behaviors and capability of the natural human intelligence. Such behaviors and capabilities offer the contexts in which communication semantics can be rationally generated and interoperability, and further increase the degree of interoperability within the versatile and complex environment.

Mobility in the list of agent properties seems to be a bit strange at a first sight, because in most discussions about agent definitions (e.g. [10], [40]), this property seems to un-related to all the other properties. In many cases this property is just adopted to please the mobile agent community, which has in many cases little interest in most other agenty properties listed here. However, with an in depth study of the essence of the mobile agent technology, we can find a closer relationship to the core of agenty.

Different from the rest of the agent society, which focuses on the intelligence of agent technology, called intelligent agents, and which is heavily based on artificial intelligence (AI) technologies, mobile agent technology has been focused on the performance improvement in telecommunications applications and is relatively in-dependent of the AI technologies.

Figure 7: Mobile Agent



A mobile agent in this context is defined as

a piece of program ([8], [118]), which can migrate (together with the program code and execution state) to different network nodes and operate there locally to achieve its objectives/ goals.

A mobile agent, as a program, specifies the behaviors to be expected in the destination environment, and is in fact one kind of knowledge to be transmitted across the network. As a mobile agent typically moves between environments with autonomous applications in order to interact locally with such applications, the mobile agents can be in fact considered as the interaction messages carrying the knowledge about execution algorithm and states, which are transmitted between the application which generates or sends the mobile agent and the application which the mobile agents visit. In fact, as showed in some implementations (e.g. [118]), mobile agent can also be implemented as a piece of knowledge (rules and facts) in some traditional knowledge representation paradigms, which migrates to different execution environments which offer the inference mechnisms.

Therefore a mobile agent paradigm implements a specific form of knowledge-based interoperability among distributed systems.

Based on these observations, and by abstracting from the list of agenty properties, we can conclude that agent technology offers a new software design paradigm that provides enhanced support for co-operation among distributed autonomous systems.

Such enhancements to the traditional technology will be based on and enable a higher degree of interoperability among distributed systems. One key enhancement enabled by agent technology is the *knowledge-based interoperability* that extends the syntactical interoperability supported by traditional distributed systems and enables dynamically adaptive, flexible and robust co-operations among the distributed applications.

Knowledge-based interoperability can play an important role in enhancing the traditional, RPC/DOT-based frameworks for the provisioning and management of telecommunications services like the advanced VPN services, and to meet the new requirements in the emerging telecommunications market.

1.4. Objectives

Following the above discussions about the background and motivations, we identify the major objectives of work presented in this thesis as to

– develop an agent-based solution supporting the knowledge-based, dynamic, flexible and robust interoperability among autonomous telecommunications applications, and to

– validate that such an agent-based solution enables some new features and capabilities in telecommunications applications, especially in the VPN services.

To mximize the interoperability among agent-based resources and the applicability/reusability of such resources within the telecommunications application environment dominated by the Internet and the Web, an agent framework will be envidaged, which

– is based on and integrates the existing standardizations of agent technologies,

– supports the easy integration and deployment of agent technology in the Internet/Web-oriented telecommunications environment.

1.5. Summary

In this chapter we have discussed the evolution of interoperability support in the context of open, distributed telecommunications applications. By analyzing the limitations of the syntactical interoperability offered by traditional, RPC-based design paradigms for open systems, we have identified the requirement for a higher degree, knowledge-based interoperability among autonomous telecommunications systems.

After discussing the key properties of agent technology we then concluded that agents can be deployed to enhance the co-operation and interoperation capability among autonomous systems, especially the knowlege-based interoperability.

Finally we identified the focus/objective of the this thesis as the development of agent-based solution for the knowledge-based interoperability among telecommunications applications.

The rest of this thesis will present this solution and its applications in telecommunications service provisioning and management. Chapter 2 will first make a state of the art analysis of the agent technology. Chapter 3 that presents a high level view of our solution for telecommunications based on agent technology. Chapter 4 gives a more detailed account of the architecture and components of our agent-based solution. Chapter 5 then tries to prove the concept by deploying the solution in the provisioning and management of the multimedia VPN telecommunication service. Chapter 6 concludes the thesis by summarizing the results and by identifying further work items.

CHAPTER 2

State of the Art

2.1. Introduction

Agent technology has already a long history in the studies of Distributed AI (DAI) technologies ([9]), but finds its widespread applications in the telecommunications industry only during the recent years. Such applications are driven at the same time by the need of the telecommunications industry for new enhanced support for co-operating distributed systems, and by the interest of the AI community for finding real business cases for the new technologies.

Especially during the last few years, Agent technology has attracted rapidly increasingly interests from the telecommunications industry. Applications in this context cover a wide range of telecom areas, including network and service management ([29], [110], [10]), Intelligent Networks ([79]), multimedia/information services ([44], [28], [122]), business process/workflow management and electronic commerce ([114]). The active researches of some major telecom companies (e.g. BT [67], TI [44]) and other system manufacturers/providers are playing an important role in giving this technology its momentum.

The widespread applications of the agent technology nowadays determine also the versatility of the agent society in terms of the heterogeneous

- backgrounds of researchers,
- application areas, and

– business requirements of the market.

Such a heterogeneity results in heterogeneous views, emphases and mechanisms in the context of agent technology and agent-based applications.

As identified in the previous chapter, one key emphasis of the agent technology is to enable a higher degree of interoperability among distributed applications, especially the knowledge-based interoperability via the enhanced understanding among the autonomous agents.

Corresponding to the heterogeneity of ways in which the enhanced interoperability is supported, the agent society nowadays is mainly divided between two communities (sometimes called religions [121]): one for *mobile agents* (called MAs hereafter) and one for *static intelligent agents* (called IAs hereafter), which focus correspondingly on the mobility and the intelligence aspects in the agent technology.

As discussed above and will be further clarified in this chapter, Both agent paradigms in fact represent complementary open system design technologies, and have a lot of commonalities in the real applications. In fact, following the philosophy of knowledge-based interoperability for agents, MA paradigms can be considered as a special form of the IA technology for enabling adaptive, flexible co-operation among distributed systems.

In the following we will first start with an overview of the MA technology, which is now rapidly gaining its momentum in the telecommunications industry. Then we will have a more detailed analysis of the IA technology, which, with its versatility of backgrounds and applications, has played and is still playing a major role in the development and application of agent technology. Finally we will have a brief discussion of the advantages and disadvantages of the IA/MA technologies in telecommunications applications, and the state of the art of co-operating and harmonizing the two complementary agent technologies.

2.2. Mobile Agents

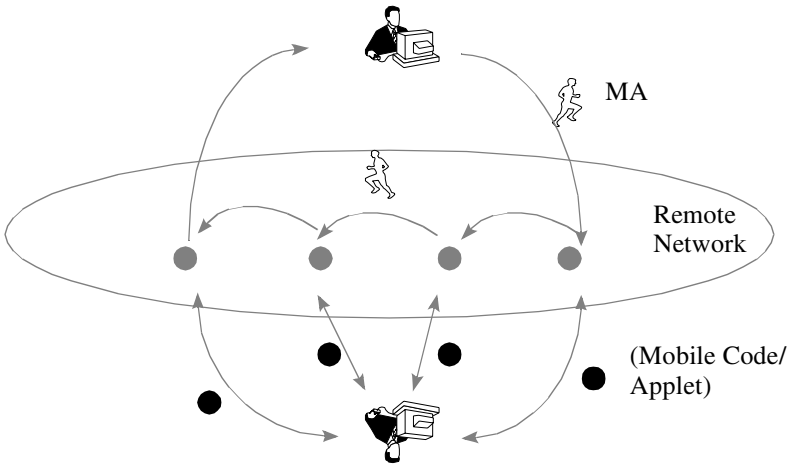
The technology of *mobile agents* has its original in the studies of AIs, but its momentum in the current information society lies almost solely in the telecommunication industry. Compared to other branches of researches in the agent technologies, MAs have most quickly find its wide acceptance in the telecommunications community. This can be partly attributed to the easy measurable and accessible advantages of the *mobility* technology within the telecommunications applications.

Basically MAs are considered as an alternative and extension to the traditional software design paradigms based on RPC and DOT, and mainly aim at performance enhancement in telecommunications applications.

2.2.1. Mobility Technology

The basic idea behind the MA technology is to enable and deploy software programs which can migrate among network node environments and operate there on the local resources.

Figure 8: Mobile Code vs. MA



In this context, some other technologies for enabling mobile *codes*, e.g. the applet technology, are sometimes regarded as primitive forms of MA technology as well. However, as a major extension to these mobile code technologies, MA paradigm puts more emphasis on the autonomy of the MA in terms of its *identity* and memory of *execution states*. This identity and state memory enables the MAs to migrate through multiple network nodes (called multi-

hop migration) and operate continuously on the distributed resources.

This possibility of multi-hop migration has significant impact on the management for the telecommunications network and services, especially for managing resources located in remote networks. As showed in figure 8, although one-hop downloading of mobile code like applet can help to reduce the number of remote interactions via aggregation of operations into a single piece of program, a number of remote interactions are still necessary if multiple remote sites are to be managed.

By utilizing the *local operations* instead of remote interactions, by *composing/aggregating elementary functions* in a large grain MA, via *dynamic downloading* and *multi-hop migrations* of the MA-based functionalities, a MA based approach can

- reduce the traffic load and the requirement of availability on the underlying networks, by reducing the number of remote interactions;
- reduce the requirement on customer intelligence during the installation, operation and maintenance of the software and solutions;
- enable “on demand” provision of dynamic/customized services (via dynamic MA migration from the provider system to the customer systems and further on back to the provider system or directly to the customer resources);
- increase the reusability, robustness and effectiveness of the software-based solutions via mobility and autonomous operation;
- allows for a more decentralized realization of software, by means of bringing the control or MAs as close as possible or even onto the resources.

The mobility of agent is typically realized by encoding/serializing the program (and its state) in a specific representation at the source network node, transporting it over the network and then interpreting it at the destination node. The basic pre-conditions in this context are

- both network nodes support the same representation and execution mechanisms for the MAs, which are mainly based on the shared MA programming language, and on the shared definition of MA structure.
- both network nodes support the same coding (serializing/marshaling and de-serializing/un-marshaling) of the MA representations for the transportation over the network, and the same transport protocol for transporting such representations.
- the operational environments of the destination network nodes (and the associated resources) support the pre-defined APIs which are known to the MAs and their program-

mers, so that, when the MAs arrive at new network nodes, they can interact with the local resources following their own goals.

Beside these basic requirements, a MA framework in the open, distributed environment also have to offer support in the following contexts:

– MA co-operations and management

beside local operations, due to the relatively higher costs (time and resources) associated to MA migration, MAs also has to communicate and co-operate with other (sometimes remote) entities including MAs, IAs or other distributed objects. The migration of MAs within the global environment also imposes a new challenge to the management of agents.

– security

As a MA typically comes from one autonomous network and application environment (e.g. generated by some user applications) and migrates to a new, foreign autonomous environment, it is vitally important to protect both the MA and its destination environments from security violations and malicious or non-malicious (but harmful) intrusions.

– efficiency

As one key objective of the MA paradigm is to improve the performance of distributed solutions, a MA-based approach has to ensure higher efficiency in the execution, transportation and operation (accessing local resources) of the migrating agents.

– reliability

MA claims to offer reliable solutions for many telecommunications applications in the global networks. This however assumes the reliable transportations of the MAs or MA communication messages.

A *MA platform* offers the necessary facilities and environment for the development and deployment of MA-based solutions. Such a MA platform can be therefore judged by its support for fulfilling all these requirements and the reusability of the support in a wide range of network and computation environments. The reusability of MA framework and platform enables the migrations and deployment of MA-based solutions in a wider range of business contexts and environments.

Currently, two key technologies - Java and OMG CORBA [70] - play an important role in the design and implementation of MA platforms and MA-based telecommunications applications.

Java, with its platform independence and Web/Internet-orientation, offers a very promising

candidate for the representation/programming and execution of MAs. Some extended Java facilities, like serialization and RMI, enable the transportation of Java objects and MAs across the network between different Java applications. Other novel Java APIs like the JNDI (Java Naming and Directory Interfaces - [91]) can be used as well to build up the naming and management services for the MAs.

Java IDL, which supports Java applications to co-operate with CORBA operations, further increases the applicability of Java-based MA solutions in an open, distributed environment.

Basically, CORBA aims at enabling interoperable, reusable, portable software within the open, distributed and heterogeneous environment, based on open and object-oriented interfaces. Via standardized interfaces defined in the standard interface definition language IDL, a CORBA ORB (Object Resource Broker) implementation in this context allows heterogeneous objects implemented in different languages to smoothly co-operate with each other.

Such interoperability, reusability and portability supported by the CORBA platform enable MAs to move among different application environments, to operate in the foreign environments and to co-operate with other MAs or IAs. Enhanced services like CORBA security services, lifecycle services and naming services can also be reused to support the implementation of a MA platform.

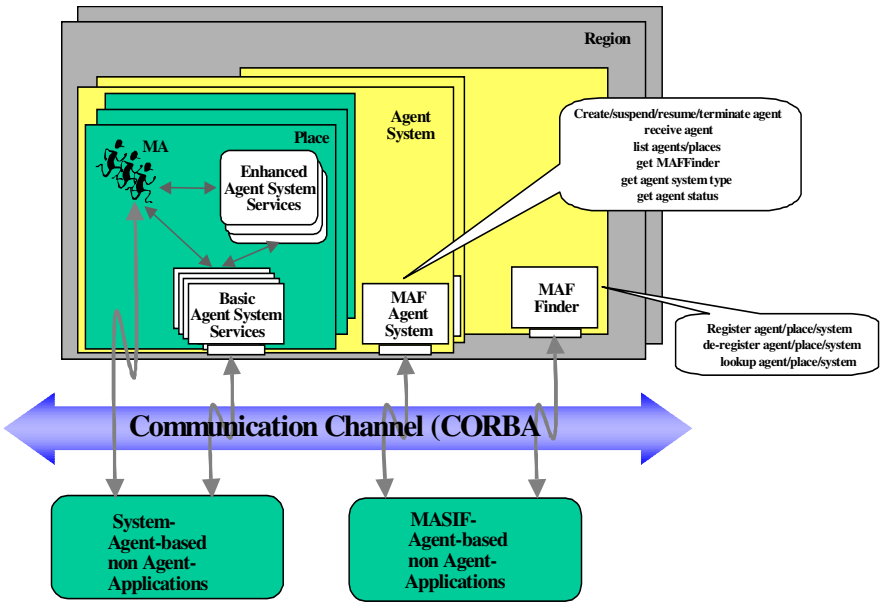
2.2.2. Standardization

Because the pre-conditions of the agent mobility are the shared MA representation, transportation and operation environments between the source and destination network nodes or the MA platforms, standardization of MA platforms and facilities plays a key role in enabling and promoting the deployment of MA technology.

The MA religion is at the moment mainly represented by the OMG MASIF ([34]) standard and the follower activities within the OMG agent working group ([74]). The idea behind the MASIF standard is to achieve a certain degree of interoperability between mobile agent platforms of different manufacturers without enforcing radical platform modifications.

The MASIF MA framework can be depicted in figure 9.

Figure 9: MASIF Architecture



Within this framework:

– An *agent system* (called MAF - *Mobile Agent Facility* Agent System) offers the platform that can create, interpret, execute, transfer and terminate MAs.

– A *place* offers the specific execution environment within an agent system. A place logically groups the functionality and services within the agent system by encapsulating certain capabilities and restrictions for the visiting MAs.

– The concept of *region* is used to group a set of agent systems that have the same authorities. Region allows more than one agent system to represent the same person or organization. Moreover, a region provides a level of abstraction to clients communicating from other region by publishing the region address (for a selected agent system) as the *access point* for the entities within the region.

Two CORBA IDL interfaces are specified in the MASIF standard:

– the *MAFAgentSystem* interface provides operations for the management and transfer of MAs, whereas,

– the *MAFFinder* interface supports the localization of MAs, agent systems, and places in the scope of a region or the whole environment.

MAs within MASIF migrate among places within the *MAFAgentSystems*. Both MAs and the external applications use the *MAFFinder* to locate the agents (both MAs and static agents), places, agent systems and regions for the purposes of migrations or other operations like agent management or agent co-operations.

With the CORBA as the basis in all OMG standards including MASIF, MA transportations and MA co-operations are all supported by the ORB and via IDL based method calls.

As the first internaltional standard for MA technology, MASIF has played a significant role in promoting the MAs in telecommunications applications. As an example, the FIPA specification of agent mobility support via IA communications ([30]) is in fact based on the MASIF standard.

However, despite its importance in the development of MA technology, MASIF itself is still an in-complete standard. Among others,

– MASIF does not sufficiently solve the security problems in MA migrations and operations,

– MASIF does not consider the problem of standardizing the mechnisms for MA co-operations/communications with other agents, especially the problem of MA communications in the dynamic and un-reliable (e.g. mobile) application environments.

This in-completeness can result in heterogeneous and non-interoperable implementations and limits to some extent the applicability of the MASIF standard.

Besides, there are currently not enough heterogeneous implementations of the MASIF standard to demonstrate and validiate its support for MA and MA platform interoperability.

The OMG Agent Working Group ([74]), which is in fact created after the MASIF standard-

ization and aims at continuing and extending the agent-oriented activities within OMG, is going to solve these problems associated to MASIF, mainly by enhancing the current specification and via collaborating the MA technology with the IA technology.

2.2.3. MA Platforms

With the strong interest and support from the research community and from the industry, numerous MA platforms have been developed and deployed in a variety of applications. Such platforms range from still proprietary and research-oriented implementations, to platforms that have become quasi industrial and commercial standards. In this sub-section we will give a brief overview of several most important MA platforms which have a significant role in the telecommunications industry.

2.2.3.1. Telescript

Telescript technology ([106], [107]) is a MA development and deployment platform, which has been developed by General Magic and an alliance of key players in the telecommunications arena including Apple, AT&T, France Telecom, Fujitsu, Matsushita, Motorola, NTT, Philips, Sony and Toshiba. It was in fact the most important pioneer in supporting MA-based telecommunications and information processing applications.

The goal of Telescript is to integrate the electronic world of computers and the networks that connect them, with a vision to provide electronic marketplaces within which providers and consumers of goods and services could find and interact with each other. The core concepts of Telescript in this context are places, agents, travel, meetings and connections. The agents *travels* among the different *places* within the marketplaces and can *meet*, i.e. co-operate with the agents residing in the same place via dynamically established *connections*. A place in the Telescript architecture offers a service within a marketplace to the MAs entering it.

The Telescript technology consists of the *Telescript language*, the *Telescript engine*, the *Telescript protocol*, and *Telescript products*.

The Telescript language is a scripting language for the implementation of Telescript agents. It is object-oriented, persistent, communication-centric programming language. Telescript language enables developers to implement major components of their distributed applications, and at the same time supports the easy integration of system components in traditional programming languages like C or C++. In fact a typical Telescript application is coded partly in Telescript language and partly in a conventional programming language.

The Telescript language is interpreted by the Telescript engine, which offers a library of APIs for the migration, operation, co-operation of the agents, and also for the integration of external (non Telescript) applications.

Agent migration in Telescript is supported by the Telescript platform interconnect protocol (PIP). PIP covers both agent encoding and agent transport between Telescript engines. Among others, it specifies how engines authenticate each other. PIP is designed as a “thin

layer" on top of a wide variety of communication. In terms of the OSI reference model, PIP would be located in the presentation and application layer.

Telescript agents can meet and interact by calling each other's method. Asynchronous communications among the agents is supported by the Telescript engine. More sophisticated cooperation protocols, which play an important role in e-commerce, are not supported.

Telescript security support is implemented by credentials and permits. Credentials, which are implemented by means of cryptographic methods, are used to verify the identity of agents when they migrate to a new destination. Permits are capabilities used for access control in the electronic marketplace.

The Telescript technology is embodied into products offered by General Magic, the Telescript Developer's Kit and the Telescript Porting Kit. The latter comprises the source code for the Telescript engine plus the tools and documentation needed to port the engine to new platforms.

Telescript has played a major role in the history of the application of MA technology in telecommunications applications. However, with the emergence and ubiquity of Java and other Web-oriented technologies, Telescript, which is based on many proprietary solutions, has already lost a major part of its share in the market.

General Magic has recently also developed a pure Java-based MA platform called Odyssey ([41]), which adopts mainly the architectural concepts of Telescript. However, this platform has not achieved the needed acceptance in the market either.

As a supporting author of the OMG MASIF standard, General Magic has also introduced many concepts from Telescript and Odyssey into the MASIF specification.

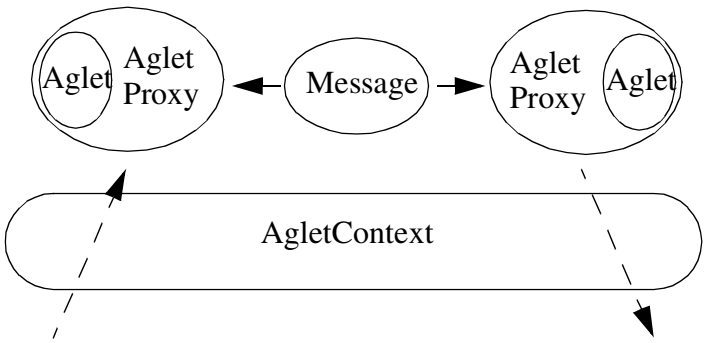
2.2.3.2. Aglets

Aglets ([75]) is a pure Java MA platform developed by the IBM lab in Japan, with the major goals to provide an easy and comprehensive model for the programming of MAs without requiring modifications to Java Virtual Machine (Java VM) or native code, and to provide a harmonious architecture with existing Web/Java technology.

Agents are called *aglets* in the platform and are written in Java. Aglets architecture consists of two layers with the corresponding APIs for accessing their functions. The *Aglets runtime layer* provides the fundamental functions for aglets to be created, managed, and dispatched to remote hosts. The *communication layer* is primarily responsible for transferring a serialized agent to a destination and receiving it. It is also responsible for supporting agent-to-agent communication and the facilities for agent management.

The Aglets platform is based on the concepts of Aglet *Proxy* and of Aglet *Context*.

Figure 10: Aglets Concepts



The *AgletProxy* interface object acts as a handle of an aglet and provides a common way of accessing the aglet behind it. Since an aglet class has several public methods that should not be accessed directly from other aglets for security reasons, any aglet that wants to communicate with other aglets has to first obtain the proxy object, and then interact through this interface. In other words, the aglet proxy acts as a shield object that protects an agent from malicious agents. When invoked, the proxy object consults the security management facility to determine whether the caller is permitted to perform the method. Another important role of the AgletProxy interface is to provide the aglet with location transparency. If the actual aglet resides at a remote host, it forwards the requests to the remote host and and returns the result to the local host.

The *AgletContext* class provides an interface to the runtime environment that occupies the aglet. Any aglet can obtain a reference to its current AgletContext object, and use it to obtain local information such as the address of the hosting context/environment and the enumeration of AgletProxies, or to create a new aglet in the context. Once the aglet has been dispatched, the context object currently occupied is no longer available, and the destination context object is attached instead when arrived.

As the Java VM does not allow stack frames to be stored, or a thread object to be resumed from them, it is impossible for a thread object to migrate from one JVM to another while preserving its execution state. Therefore all Java-based MAs are based on the *event model*, which can also be called *object mobility* model in the sense that MAs are regarded as passive objects that can be serialized and transported across the network. Different methods can be associated to the events during this whole procedure like *onDispatching*, *onArrival* etc., which will be called by the platform upon occurrence of the events to maintain the continuous thread of the MAs.

For MA migrations, a simple application level protocol is implemented to transmit an agent in an agent-system-independent manner, which is called ATP (Agent Transfer Protocol) and modeled on the HTTP protocol. Besides, ATP also supports the agent communications based

on message passing. Both synchronous and asynchronous communications are supported in this context. Similar to most other MA platforms, Aglets does not support any high level negotiation protocols.

Security support in Aglets is based on the authentication of users and *domains*, integrity checked communication between servers within a domain, and the fine-grained authorization similar to the JDK 1.2 security model.

As an author of the MASIF standard, IBM is currently enhancing the Aglets platform to make it conform to the MASIF specification. Among others, it is going to support IIOP for agent migrations and communications.

2.2.3.3. Voyager

Voyager Core Technology (abbreviated to Voyager in the following) from ObjectSpace [68] is in fact a high-performance and 100% Java ORB (object request broker) for the state-of-the-art, distributed computing that simplifies and unifies the most common industry standards. It has also dedicated interfaces for interacting or integrating CORBA, Java RMI and DCOM-based applications.

The basic concept within Voyager is *object*, which is the building block for all Voyager. Objects live, i.e. are created and operating in Voyager programs, which are the Voyager execution environments and represent the local nodes in the global network similar to the concept of place or agent system in the MASIF standard.

Objects can interact with each other via method calls which is called *messages* in the terminology of Voyager. Both synchronous and asynchronous messaging are supported.

Agent (MA) in Voyager is regarded as a special class of objects which support one method for migrating an object to a new destination, i.e. a new Voyager program or an object residing in such a remote Voyager program. This method call specifies beside the destination also a method to be called when the agent arrives at the destination, in order to maintain the continuous execution of the agent.

Voyager uses an optimized Java serialization/deserialization mechanism for the transportation of agents and agent co-operation messages (method calls and parameters). In this way, Voyager achieves a much better performance compared to most other MA platforms. This feature, together with Voyager's easy-to-use programming interfaces, make Voyager a very promising development and deployment environment for MA-based telecommunications applications.

Voyager Security provides support for secure network communication over the industry-standard SSL (Secure Socket Layer) protocol, allowing remote communication over an encrypted and authenticated channel. Beside this, Voyager also implements a *VoyagerSecu-*

rityManager that manages the access rights granted to different categories of objects.

Voyger also implements a CORBA OTS-compliant distributed transaction facility that ensures all transactions are properly committed or rolled back. Using a two-phase commit, Voyager transactions allow multiple resources to participate in a transaction across multiple Java VMs.

Voyager currently does not support any MA standard, which is one of the few shortcomings that can hamper the deployment of this platform in many application contexts.

2.2.3.4. Grasshopper

Grasshopper from GMD FOKUS and IKV++ [50] is the world wide first MA platform that fully implements the OMG MASIF standard. Its popularity within the agent community can be partially attributed to its adoption in the numerous European research projects for the development of agent-based telecommunications applications.

Grasshopper is a pure Java platform based on the architecture specified by the MASIF standard. MAs in Grasshopper is implemented as Java Thread objects that can migrate to different places in different agent systems (called agencies in the Grasshopper terminology). Such a MA will be re-activated at each new place and continue the execution based on the execution states maintained in attributes of the MA Thread object. A MA accumulates its execution state in the attributes of the MA object to enable continuous thread of execution over multiple sites.

Grasshopper supports a number of popular protocols for the transportation of MAs and the communication between agents, including IIOP, RMI and plain socket. RMI and sockets can be further combined with the SSL to enable security support in agent migration and co-operations.

Grashopper implements not only synchronous and asynchronous communication among agents, but also the multicast communication mode which enables a client to use parallelism when interacting with the server agents.

The security support of Grasshopper is divided into external security support and internal support. The external security support protects remote interactions between the distributed Grasshopper components, and is based on the SSL for ensuring confidentiality, integrity and mutual authentication of client and server. Internal security, on the other hand, protects resources of the agent systems and places from unauthorized access by agents. Internal security is based on the Java JDK 1.2 security mechanisms and uses an identity-based or group-based access control policy.

Similar to other MA popular platforms, the basic Grasshopper platform does not support any high level co-operation protocols or mechanisms except the basic RPC-based interactions.

Such a support, which will be important in a heterogeneous and dynamic environment and falls into the realm of IA, is considered as extensions to the basic Grasshopper architecture.

2.3. Intelligent Agent

The word *intelligent agent* is more frequently used in the Artificial Intelligence (AI) community. In that context, IA technologies often refer to the application of AI technologies in a distributed and co-operative environments, e.g. co-operative expert systems. Therefore IAs are sometimes regarded as intelligent autonomous entities that co-operate with other entities in order to achieve the aggregated goal of the whole distributed system. As most of the IAs in this agent religion are static, i.e. they (e.g. the programs) are installed at one network node and do not move to other node during their operations, they are also called *static agents* in the literature, simply to distinguish them from the MA paradigm.

2.3.1. IA Co-operation Technologies

One key issue in the IA technologies, which is also the focus of this thesis, is to achieve enhanced and intelligent understanding among the autonomous systems or between agent systems and humans. Such an understanding will have to be based on some encoding paradigms of the interaction messages among the distributed applications, i.e. language structures with well understood meanings. Such encoding paradigms can be deployed in characterizing and representing the meanings (i.e. the meanings or the expected behaviors) associated to information content of the co-operation messages. With these well understood meanings, both the sending and the receiving agents are in the position to generate and interpret the communication messages in the correct way in order to achieve the expected and required effects in their co-operations.

Basically we can identify two typical scenarios in the definition and deployment of encoding paradigms, which offer either *context-independent* or *context-oriented* characterization of the agent communication messages.

In the first case, a single knowledge representation language is used to compose the co-operation messages and can fully characterize the meanings of the information content.

With the assumption that the sending and receiving agents both agree on the same semantics of the representation language, i.e. how to execute the programs in that language, the sender can compose the message following its intensions and objectives, while the receiver can correctly interpret the information or action requests encoded.

The key feature in this paradigm for agent-based interoperability is that a co-operation message will be interpreted as a single semantic unit by the receiver. We can call the agent communication paradigm in this case as *context-independent* because the interpretation mechanism of the messages is based on some standardized knowledge representation language, which is independent of the context of the co-operation application, status and envi-

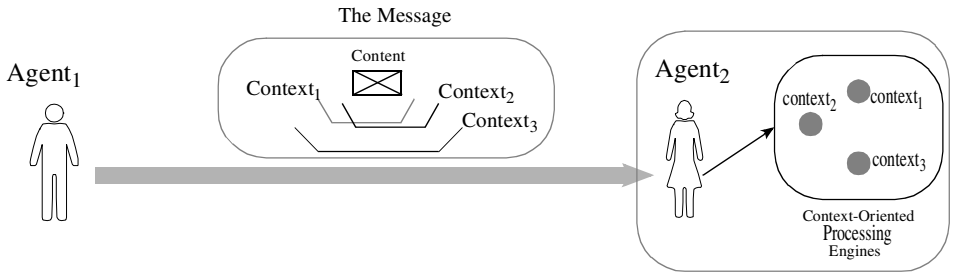
ronment. The receiver typically executes the program by following solely the instructions in the message.

The complexity of the heterogeneous and dynamic telecommunications co-operation environment, however, makes it in many situations difficult to assume that all the agents support the single, context-independent semantics of the knowledge representation. In order to get all the agents to work together for achieving their goals, the agents typically have to dynamically find out, switch to and then operate in the specific semantic contexts that are shared by the co-operating partners.

It is like human communications, when someone talks to a foreigner (or someone he does no know well), the conversation partners will typically try to use some sentences in a communication language (e.g. English, or Esperanto), and sometimes even the visual sense, to clarify the context for the conversations, e.g. where the partner comes from, his background, his interests, his intention, his mood and attitudes etc. The messages exchanged will be interpreted with that context. In fact, a human will use different semantics to interpret the messages depending on the contexts he has percepted. In most cases, the conversation will be accumulative, which means the conversation partners will constantly refine their perception about the contexts, or percept new contexts that are relevant for the conversation. In this way the partners will accumulative improve the effectiveness and profitability of their co-operations with respect to their goals.

Similarly, instead of regarding an agent co-operation message as a semantic unit, we can consider such message as a structured entity, consisting of a number of layered contexts and contents that accumulatively identify the semantics (i.e. the associated or expected rational behaviors) that is necessary for the receiver to behave accordingly to the sender's expectations. This scenario can be depicted in figure 11.

Figure 11: Context-Oriented Communication



In this paradigm for agent communications, specific elements/components in the communication language or even separate languages are used to encode or identify the specific contexts of the message *contents* exchanged between the agents. Upon receiving such a message, the receiving agent will first identify the layers of contexts by interpreting the specific language elements or components using the shared semantics. After identifying such contexts,

the agent will usually select a dedicated processing engine or processing algorithm which is appropriate for these contexts to process the message content.

Such an approach for interoperability among agents can be called *context-oriented* because it works on characterizing the semantic contexts of the co-operations and the real contents carried by the messages, especially by simulating the different contexts in human communications.

By separating the communication semantics into layers of contexts, this agent communication and co-operation paradigm can significantly reduce the complexity and increase the flexibility of agent co-operations, and it can also increase the reusability of semantic information.

More specifically, if the complexity of the semantics of a message consists of n semantic elements, whose complexity in the possible value domain (assuming the same complexity for each semantic element for simplification) is N, then the complexity of processing algorithm for the message, measured by the number of combination scenarios to be dealt with, is n^N. However, if we divide the semantics and the elements into k layers of contexts, the complexity at each layer will become $(n/k)^N$, the total complexity becomes the sum of the complexity of the k engines for dealing with the k layers, i.e. $k \times (n/k)^N = n^N/k^{N-1}$, which can be much smaller than the context-independent paradigm. Therefore in principle a context-oriented approach can simplify the processing of co-operation messages in a heterogeneous and complex environment.

Another interesting feature is that the context information can be transported and maintained separately from the main part of the message itself during the agent co-operation conversations. In this way, an agent can aggregate context semantics and reuse it in a wider range of situations for processing the co-operation messages. Such aggregation can happen also through learning the semantics from some external/third party agents or from the agent's own experience.

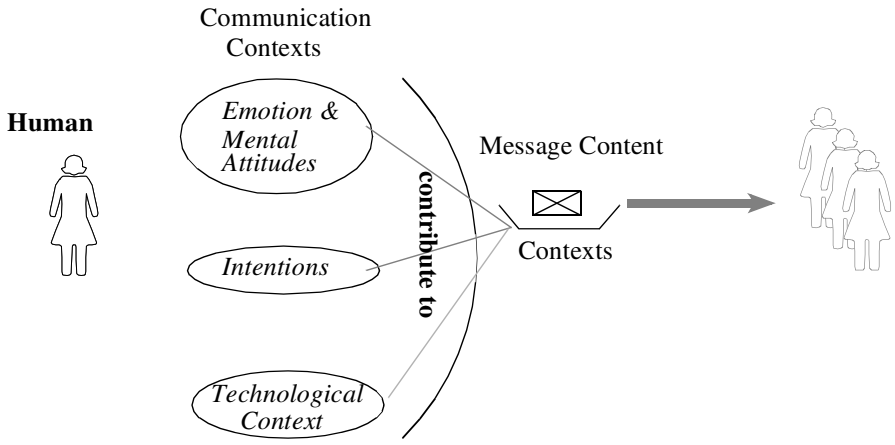
Different from the context-independent paradigm, the context-oriented interoperability still allows some degree of freedom for the agents to interpret part of the message semantics following its own interests, views and knowledge. A context in fact characterizes only part of the semantics of the co-operation messages. Co-operations among agents can be enabled by a set of contexts that sufficiently restricts the semantics for the purpose of the joint tasks in the telecommunications applications. In many cases, each agent can still interpret the rest of the semantics in its own way. Such a paradigm further reduces the complexity and increase the flexibility in co-ordinating autonomous agents.

In summary, one focus of IA co-operation technologies is to enable agent interoperability by identifying, interpreting and standardizing the contextual knowledge in agent communication messaging.

With their strong AI background, such context-oriented IA co-operation paradigms typically

are based on simulating the corresponding aspects in human communications.

Figure 12: Contexts of Human Communications



Human communications happen always within specific contexts, which affect or determine the interpretation of the communication message contents. Aggregation of such contexts can partially or fully define the semantics of the whole messages.

Among others, such contexts include the *emotions and mental attitudes* of the speakers, the *intentions* of speakers with their messages, and the *technological* contexts such as computer science, network management or agent technology, in which the conversations take place.

Basically speaking

- emotions refer to a human's view of its internal psychological or biological status, i.e. his emotional feelings like being happy, sad, bored or excited, while mental attitudes refer to a human's view (i.e. attitudes) towards the events, entities and facts in the environment, e.g. whether the human believes in something, wishes something, admits to something, dislike something etc.;

- intentions can refer to the intended acts (speech acts) of saying something, which (as will be discussed in more details in the following) are also called *illocutionary acts* in the philosophy of languages [82], e.g. to make statement, to give commands, to ask questions, to make promises and so on;

- technology specific contexts characterize, via the technology specific terminology, protocols and semantics, the meanings of the message for a specific group of experts within that technological community.

Similarly, an agent can simulate such contextual information and use such information in the

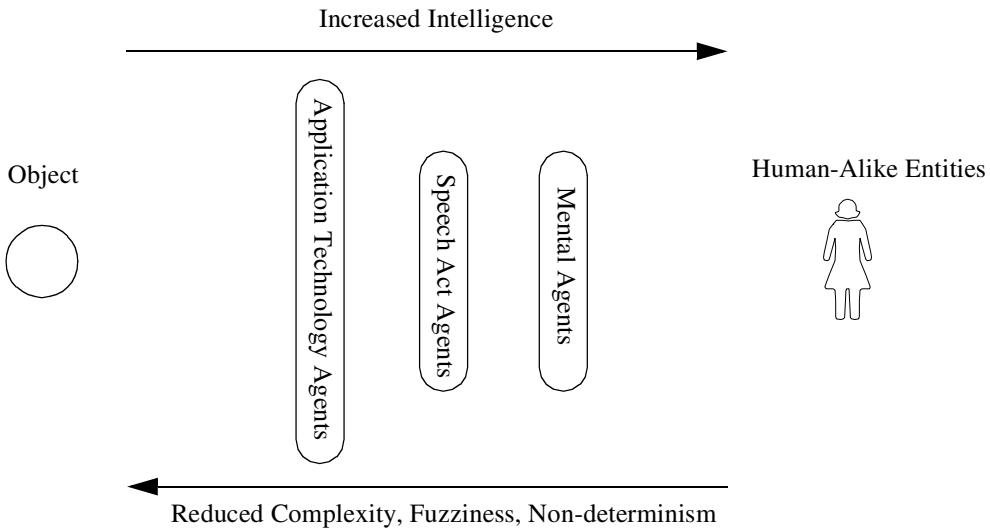
generation and interpretation of agent co-operation messages. As a result, we can approximately distinguish among the following categories of IAs by focusing on different contexts of co-operation semantics:

- *mental agents* that either present emotional characteristics or have appropriate mental attitudes, and support the associated rational behaviors,
- *speech act agent* that support agent co-operation by identifying and interpreting the illocutionary speech act contexts in the co-operation messages,and
- *application technology agents* based on some traditional application specific communication protocols that partially characterize the *application-oriented* technical contexts following some application specific standards.

The relationships among these co-operation technologies can be depicted by the spectrum in figure 13. Such a spectrum of IA technologies in fact reflects the evolution from the non-intelligent distributed *object* to the human-alike *intelligent entities*, and from traditional telecommunications applications to the highly intelligent software systems.

Agents based on the emotions or mental attitudes and the associated rational behaviors model the human intelligence at a very high level and therefore present, in a higher degree, those features that are associated to a typical AI-based system. Such features include the non-deterministic, fuzzy behaviors, and also the complexity or difficulty in the definition of a suitable formal model that can be easily implemented by a machine.

Figure 13: The Spectrum of IA Technology



Application technology-oriented agents, on the other hand, are usually regarded as straightforward extensions to the traditional DOT-based applications with intelligent capabilities, like the experience or knowledge derived from the application domains, for presenting the properties of an agent. Because of the strong application-orientation and the generally deterministic features of traditional telecommunications applications, this category of agents currently put less emphasis on offering human-alike behaviors in their dynamic co-operations with the environments. With their application oriented background and the available support from existing technology, this category of IA technology has also less complexity in modeling the problem domains for the IA co-operations.

The speech act agents in fact aim at simulating some relatively deterministic and more elementary behaviors within human communications and co-operations, and have relatively low complexity in terms of fuzziness, non-determinism etc., compared to the mental agents. Together with the integrated support for the intelligence in agent co-operations, this category of agent technology currently plays the most important role in the context of agent-based telecommunications applications, and are also the focus of international standardization efforts.

2.3.1.1. Mental Agents

The key objective of the mental agents (e.g. [14], [80]) is to present the believable character or personality of the agent-based applications by simulating the human emotions or attitudes. Such an objective can usually be achieved via

- identifying a set of emotions and mental attitudes,
- implementing the raional behaviors based on the semantics of such emotions and mental attitudes by simulating the human behaviors.

In this context, an emotional state refers to the internal emotions of IA based on the aggregation of its views on interactions with the environment. It is therefore a function of the IA and the time, i.e. we have an emotional state as

$$s = f_s\,(agent,\ \ time).$$

An attitude of an IA, on the other hand, reflects the IA's view of its environment, and is a function which also depends on the external reference entity, i.e.

$$a = f_a\,(agent,\ \ time,\ entity)$$

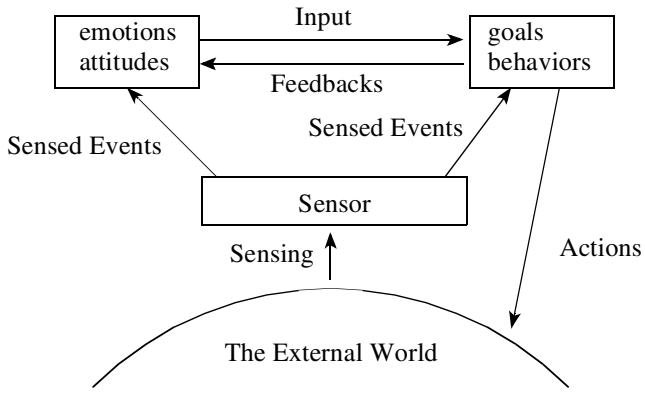
where the external entity can be a logical fact or an object in the environment. It is also in this sense that an emotional state can be regarded as the IA's attitude towards itself.

Rational behaviors in this context include

– maintaining, i.e. updating the emotional states and attitudes by calculating appropriately the f_s and f_a for the agent based on agent's current states and its interactions with its environment, and

– logical agent activities/actions that are appropriate for the agent's emotional states and attitudes.

Figure 14: Emotional Agents



Following this philosophy, the architecture of an emotional agent, as described in some literatures (see [80]) can be abstractly depicted in figure 14.

In a typical scenario, a mental agent will sense its environment via some sensor module. The sensed information events will be forwarded first to an emotions & attitudes module that governs, e.g. updates, the emotion and attitude functions (i.e. f_s and f_a) of the IA. Either reactively or proactively, the emotions and attitudes will be used by the goals & behaviors logic of the IA, together with the information about the sensed events, to make decisions on the possible actions from the IA. Such actions will be carried out upon the outside world.

Currently the applications of mental agents are still mostly restricted to universities and research labs. As some representative examples in this context:

– *Tok* ([1], [80]) from Carnegie Mellon University is an IA architecture which integrates emotions and goal-oriented reactivity in realizing the rational agent behaviors, e.g. in terms of movement within a *simulated world*. Tok integrates emotions and attitudes like hope, fear, happy, sad, pride, shame, admiration, reproach, gratification, remorse, gratitude, anger, love and hate into the internal state of the IA. Tok makes decisions upon IA actions following some rational rules and based on the current emotions and attitudes.

– The *AR* (Affective Reasoner - [14], [15], [16]) from DePaul University is a platform for researches on various aspects of computing IA emotions. In a research-oriented application environment, AR was deployed for

  • effecting a computable model of storytelling that uses a sophisticated representation of emotion interaction and personality to build a robust, dynamic model of stories, and

  • supporting theoretically rich, emotionally expressive virtual actors.

  Among others, AR bases its behaviors on emotions and attitudes like joy, distress, happy-for, gloating, resentment, jealousy, envy, sorry-for, hope, fear, satisfaction, relief, fears, confirms, disappointment, pride, admiration, shame, reproach, liking, disliking, gratitude, anger, gratification, remorse, love and hate.

– *Cathexis* from MIT Artificial Intelligence Laboratory ([99]) is based on modeling the dynamic nature of different affective phenomena, such as emotions, moods and temperaments, and their influence on the behaviors of synthetic autonomous agents. Among others, the agent framework aims at applications like entertainment (e.g. interactive video, video games), education (intelligent tutoring) and human-computer interfaces.

– The *PUMA* system from the University of Caen ([3]) integrates the mental attitudes like belief, competence and knowledge, into the logical programming language and tools for the implementation of IAs.

Beside supporting the human-alikeness in IAs, which can have an important role especially in human-oriented, social and friendly interfaces, emotions and mental attitudes have also another, currently more important application in developing the IA-based telecommunications paradigms:

Generally speaking, the emotions and especially the mental attitudes, and the associated rational behaviors build up a mental model of the agents. By regarding such a model as the basis (i.e. the meta model) for agent's co-operations, we can define the semantics of IA co-operation messages and protocols by relating them to the status and expected changes in the IA's emotions and attitudes.

E.g. Cohen and Levesque ([7]) uses an extended model of mental attitudes to characterize the semantics of the KQML agent communication language, while FIPA ([24]) uses a simplified mental model based on belief, uncertainty and choice (desire, goal) to specify the semantics

of FIPA Agent Communication Language, which is the FIPA standard for agent communication speech acts.

Using a mental model for the semantics of IA communications and the associated rational behaviors, we have also a new possibility for dynamically defining/re-defining the IA co-operation protocols or language elements via exchanging the new knowledge among foreign IAs. This possibility, which can play a key role in supporting the dynamic IA co-operation functionality and behaviors, is not yet sufficiently explored in the literatures.

2.3.1.2. Speech Act Agent Technology

The speech act based IA paradigm is derived from studies in the area of *philosophy of language* ([82], [57]), based on the linguistic analysis of human communications. Within the philosophy of language, production or issuance of a sentence token under certain condition is called a *speech act*. Speaking a language is therefore regarded as performing speech acts.

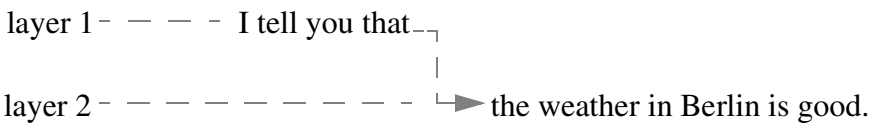
Different speech acts can be identified in the process of uttering a sentence. In this context, speech acts are decomposed into locutionary, illocutionary and perlocutionary acts ([6], [82]). Locutionary acts refers to the formulation of an utterance, illocutionary refers to a categorization of the utterance from the speakers perspective, while perlocutionary refers to the intended or indirect side-effects of the illocutionary acts on the actions, thoughts or beliefs etc. of the hearer. E.g. in the case of the agent communications, the perlocutionary effect can refer to the updating of the agent's states after processing the received speech act messages.

In the context of an agent-based application environment, the key component of agent communication messages is the identification of their *illocutionary acts*, i.e. the intended type of acts done in saying something, e.g. making statements, giving commands, asking questions, making promises and so on. Such illocutionary acts are also considered as the basic or minimal units in the agent communication linguistic, i.e. the units that are considered as the individual messages that can be transmitted over the network. In another word, each agent communication message refers to an illocutionary act. The perlocutionary acts on the other hand, will play an important role in specifying the semantics of such illocutionary acts, i.e. the rational behaviors expected from the receiving and from processing the messages of the illocutionary acts.

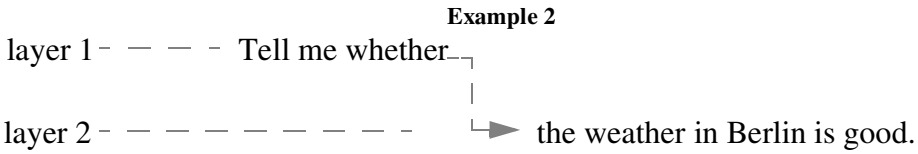
Based on this view of illocutionary acts, each message/sentence (e.g. a spoken message) in the context of human or agent communication has a layered structure as depicted by the examples in figure 15.

Figure 15: Communication Messages in Natural Language

Example 1



Example 2

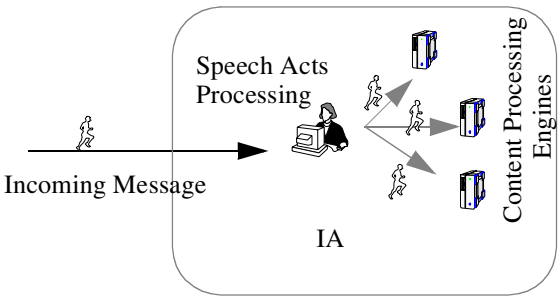


The first layer consists mainly of the *verb* plus the *subject/objects* of this verb.

The *verb* identifies the type of illocutionary act of the message. Following the tradition in the philosophy of language and within the KQML [58] community, we will call such a verb a speech act *performative*. The performative, together with the subjective/objective specifies the speech act and illocutionary context (and the intension) in which the *content of the message* (the second layer in the examples) is to be interpreted. The information contained in this outer layer of message structure is called the speech act message layer in the literatures of IA studies and is based on an Agent Communication Language (ACL), while the internal layer is called the *content* layer based on a Content Language (CL).

An ACL will be used to identify the speech act context of the IA communication messages. The content of communication message, on the other hand, is represented in the CL together with some domain-specific *ontology*, which identifies the terminology/taxonomy (i.e. the elementary terms) used in that domain and their meanings.

Figure 16: Message Processing within an IA



Different speech act performatives can determine the different ways of interpreting the con-

tents contained in the messages.

A human will typically first interpret the performatives and then decide on the response/ action that should be appropriate in that conversation. Similarly, an autonomous agent will typically first interpret the performatives in the ACL messages and then dispatch the message contents to the appropriate agent message processing engines, as depicted in figure 16.

Beside this analogy to the structures of natural language based communications, there are also some practical reasons for dividing the agent communication messages into two layers, i.e. to

- significantly reduce the complexity of syntax and semantics of the agent communication messages, and also their processing by the sending or receiving agents,
- enable the pre-processing of communication messages without processing the contents, and therefore increase the efficiency of the whole co-operation scheme,
- provide pre-defined and standardized message semantics for enabling/bootstraping the interoperability among agents.

As will be discussed later, speech act based IA communication can be further complicated (or strengthened) by the facts that

- the message content can further contain speech acts at the next recursive levels, and
- an IA ACL (i.e. the set of speech act performatives) can also be dynamically extended via generating knowledge concerning new speech act semantics, and by exchanging such information among the agents.

The DARPA Knowledge-Sharing Effort in the early 90's ([96]), lead by a group of prominent American universities (especially the Stanford University) and research institutions has played a significant role in the history of speech act-based IA technology.

The Knowledge-Sharing Effort was an initiative to develop the technical infrastructure to support the sharing of knowledge among distributed and heterogeneous systems ([64]). The key activity in this context was to develop a technology that will enable researchers to develop new knowledge-based systems by selecting components from library of reusable modules and assembling them together, which would facilitate building larger system cheaply and reliably.

The Knowledge-Sharing Effort was divided into four working groups and produced a lot of

results in supporting the sharing of knowledge among heterogeneous systems. The key results from this effort, which paved the way for most current research and development in the context of speech act IAs, are

- the Knowledge Query and Manipulation Language (KQML- [21],[58])
- the Knowledge Interchange Format (KIF- [42]), and
- the Ontolingua framework ([20], [38], [39]).

KQML is an ACL and a protocol to support the high level communications among IAs. On the one hand, it is a language for an IA application program to interact with other IAs co-operatively in problem solving. On the other hand, KQML also includes a protocol which governs the use of the language (e.g., a pragmatic component) or the performatives.

With its background in knowledge-based systems and knowledge processing, KQML views the co-operating IAs mainly as knowledge-bases and is strongly oriented towards knowledge-base management. As a result, most performatives in KQML are used to manipulate knowledge (rules and facts) associated to the IAs, e.g. to query the knowledge-base, to extend or delete knowledge items, to inform or publish knowledge etc.

The first version of KQML was defined in the early 90's ([21]). Recently, a new version of the KQML ([58]) was proposed by the original author after several years of feedbacks and experiences from the implementations and applications of KQML-based IA technology. This new version made some modification to the list of performatives to make KQML more suitable for many applications.

One key criticism on the definition of KQML is the lack of formal semantics for the language, which in many cases have resulted in ambiguity in the interpretations and non-interoperability among KQML-based applications. Although there were some efforts to give formal specifications of the KQML speech act semantics based on some mental model of the IAs and the corresponding modal logics ([7]), the results are neither well accepted by the IA community, nor widely used in KQML-based applications.

KQML as an ACL is independent of the representation of the message content in IA co-operations (i.e. the CL). However *KIF* was developed in this context as the default CL for KQML-based IA communications.

KIF is based on an extended version of first order predicate logic, and is intended to be a core language which is expandable by defining additional representational primitives. It is also designed with the goal of enabling practical means of translating declarative knowledge bases to and from typical knowledge representation languages.

KIF adopts a very simple, Lisp-alike, list-oriented syntax, and uses a number of generic structures for the different forms of definitions about concepts, objects and relationships. Such

language structures make KIF a strong language in the representation of knowledge and semantics. With the simple syntax structure, the language is also especially appropriate for the efficient coding/decoding and for the reliable transportation of messages over the global network. At the same time however, such a generic feature and expressiveness also make the language difficult and inefficient to be implemented. That's why most current KIF implementations support only a small subset of KIF. E.g. the SIF language, which is a proper subset of KIF and which is supported by most KIF implementations ([13]), corresponds to the Horn clauses of the Prolog.

Ontolingua is designed as the mechanism for writing ontologies in a canonical format, such that they can be easily translated into a variety of representation and reasoning systems. The syntax and semantics of Ontolingua are based on KIF. It extends KIF with standard primitives for defining classes and relations, and for organizing knowledge in object-centered hierarchies with inheritance. A software server framework was also implemented that supports the sharing of Ontolingua-based ontologies among autonomous IAs.

The Open Knowledge Base Connectivity (OKBC-[5]) is in this context a further development and standardization of Ontolingua and the Ontolingua server functionality. OKBC integrates a frame-based knowledge representation framework into the basic ontolingua framework of classes, objects, relations and functions.

Although Ontolingua and OKBC are very popular within the IA community, they don't have yet found sufficient acceptance in the commercial telecommunication applications. With their background in AI and formal specifications, one key problem in this context is the high *complexity* associated to the definitions and deployment of the ontologies in real application contexts. In fact, to avoid such complexity, most current IA-based applications (e.g. [26], [29], [111]) prefer to define only the syntactical aspects of the deployed ontology and leave their semantic aspects, especially the behavior information like relations and functions to the intuitions of the human programmers.

The IA framework developed by the Knowledge-sharing Effort in facts serves as the basis for all the current developments of speech act IA technology. At the moment, however, the research and development in this context is mainly represented by the FIPA (Foundation for Intelligent Physical Agents) standardization efforts ([22]), which are in fact based on tradition of KQML, KIF and OKBC, but are more focused on considering the requirements from telecommunications applications. We will discuss the FIPA standardization of IA technology in more details later in this chapter.

Speech act agent communication technology, and the associated co-operation protocols correspond to the core patterns of human negotiations and co-operations in technical or business environments. Such an analogy makes speech act agents currently the most important IA paradigm for telecommunications network, service and business process negotiation, provisioning and management.

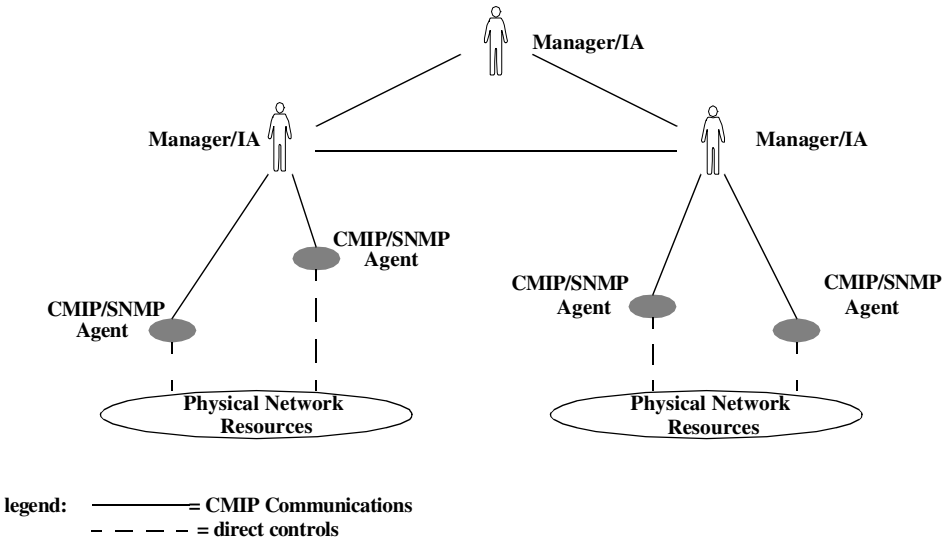
2.3.1.3. Application Technology Agents

Different from the other IA technologies, which are focusing on generic technologies that can enable the enhanced interoperability among IAs in any application domains, application technology agents work

- with some traditional, mostly RPC-based application specific protocols for clarifying the contexts in interacting distributed applications, and focus on
- enhancing the traditional applications with some agenty features like proactiveness, reactiveness, and intelligence based on reasoning and learning.

The approach in this context is to replace a application component in a traditional, RPC-based distributed system with an expert or knowledge-based system. Existing communication and co-operation technology like ITU-T TMN/CMIP and GDMO, IETF/SNMP, CORBA/IDL will be used to exchange messages for events and operation directives among such knowledge-based IA systems, or between the IAs and their non-agent environment.

Figure 17: TMN-based IA Technology



A typical example in this context is the deployment of DAI technology in a TMN or SNMP-based network management environment using the manager/agent concepts ([81]), as depicted in figure 17.

Following the TMN principle ([53]), the network resources are managed directly by some CMIP or SNMP agents, which support an abstract view of the physical resources as managed objects (MOs). The agents are responsible for updating the MOs, for implementing manage-

ment operations on the MOs and for issuing event reports on behalf of the MOs. A set of managers in this environment are responsible for receiving event reports, for issuing and coordination of management actions. Managers and agents exchange messages via the CMIP or the SNMP protocol and by operating on pre-defined interface in terms of MOs.

A manager application is usually responsible for one physical or logical domain, and can also co-operate with other managers for management operations across multiple domains. In this case either manager plays the role of agent to the other managers or the other managers become agents for this manager. Such a recursive relationship can result in a hierarchy of manager applications with CMIP/SNMP as the single protocol for co-operating the distributed applications.

The manager applications can be implemented using AI technologies and be regarded as IAs that have pro-active, reactive, reasoning and learning capabilities, e.g. for the purpose of fault analysis, automatic fault detection, prevention and recovery, resource planning and configuration, performance assurance etc. Such integrated intelligence in a TMN/SNMP environment can help to increase the automation, reliability and robustness of the management solutions.

The key property of this and other similar approaches for IA-based telecommunications applications is the static nature of IA co-operation interfaces resulted from the RPC-based co-operation technologies. IA co-operations in this context are based on pre-defined interface syntaxes. Lack of mechanisms for exchanging semantics and knowledge for dynamically defining new co-operation relationships puts some limitations on the applicability this IA paradigm in the distributed and dynamic telecommunications environments.

In fact, it is still arguable whether this category of IAs really belong to the realm of IA technology, especially because of their difficulty in distinguishing themselves from the traditional RPC/DOT-based applications or expert systems. This and also the heterogeneity of the application technologies are the reasons why most standardization efforts nowadays have not integrated the results of researches and developments in this context.

2.3.2. FIPA Standardization of Speech Act IA Technology

Within the open and distributed information and telecommunications environments, standardization aiming at enabling interoperability across heterogeneous agent-based applications plays a key role in promoting the success of IA technology.

Despite the versatility of IA researches as discussed above, the main trend of the development is still in the area of speech act agents. This is also the major category of IA technology which has achieved the maturity and the sufficient degree of consensus or interests within the agent community to justify the international standardization efforts.

The recent standardization efforts in this context are represented by the FIPA (Foundation for Intelligent Physical Agents) organization, which is a world-wide non-profit association of companies and universities that are active in the agent field. The goal of FIPA is to promote the success of agent technology via making available in a timely manner, internationally agreed specifications that maximize interoperability among agent-based systems.

Driven by the requirements on the agent technology which are imposed by specific industrial applications, FIPA has produced a series of standard specifications that specify the interfaces of different components in the environments with which an IA can interact. These specifications are divided into two categories:

- *normative* specifications (e.g. [23], [24], [25], [30], [31], [32]) mandate the external behavior of an agent and ensure the interoperability with other FIPA-compliant sub-systems,

- *informative* specifications of applications (e.g. [26], [27], [28], [29]) provide guidance to the industry on the use of FIPA technology

Among the FIPA specifications, the FIPA agent management framework, the FIPACL and content language, and the FIPA ontology service offer the key technologies that enable the knowledge-based interoperability among speech act IAs, and are also closely related to the work that is going to be presented in this thesis. We will therefore focus our discussions on these specifications.

2.3.2.1. FIPA Agent Management Framework

FIPA Specification Part 1 ([23]) defines the basic environment for the deployment of FIPA IA technology, and mainly contains the specifications of the FIPA

- agent reference model,

– agent platform,

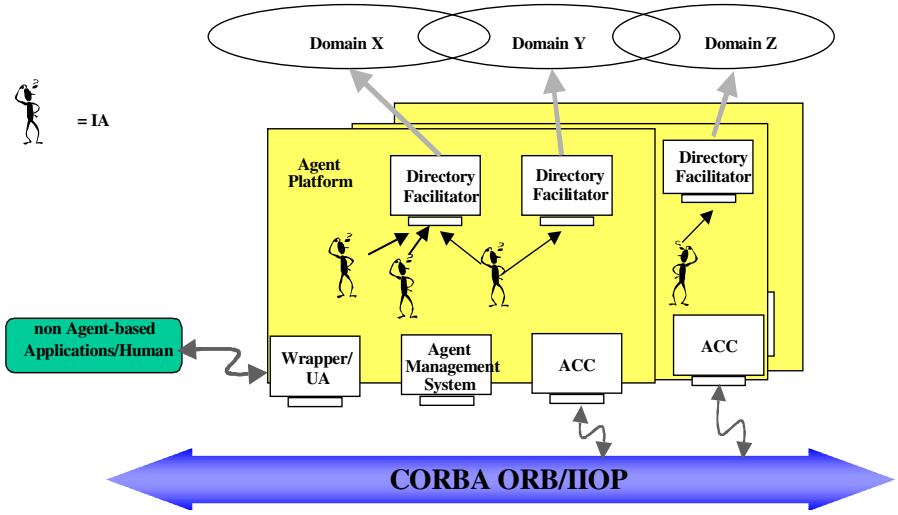
– agent management actions, and

– agent management content language and ontology.

The primary concern in this context is the interoperability between agents and the agent platforms.

The FIPA agent reference model can be depicted in figure 18.

Figure 18: FIPA Agent Reference Model



Within this reference model, an agent execution environment consists of a set of interconnected *Agent Platforms* (APs) that offer the local environments in which individual agents will reside and operate. An AP offers the generic facilities for the application independent transmission/transportation of agent interaction messages, and a generic framework in which agents can be managed. In another word, APs achieve the application independent (within a specific telecommunications context) *generic interoperability* among the agents.

An *agent* is the fundamental actor in a FIPA application environment that combines one or more service capabilities into a unified and integrated execution model which may include access to external software, human users and communications facilities. Such service capabilities include both application specific services, or application independent, platform services like the agent co-operation and management support.

Agents are also responsible for integrating the external, non-agent world, e.g. human operators, legacy applications etc., for realizing the functionality of the whole distributed system. Such non-agent entities will be integrated into the AP typically via some gateway applica-

tions like

- Wrappers [25], which adapt the APIs of the legacy software into agent communication interfaces, or
- User Agents (UAs - [32]), which interact with the human on behalf of agent-based applications via dedicated human agent interaction interfaces.

Each agent has a unique identity, called agent name in the global agent environment, via which the agent can be identified and located. Several notions of identities can be supported in the implementation of APs. The FIPA GUID (see [23]) can be used as the default agent name over all APs, which labels the agent so that it may be unambiguously distinguished in the agent universe.

The *Directory Facilitator* (DF) is an agent which provides *Yellow Page* services to other agents. In this sense, a DF specifies a logical *domain* in which the services of agents can be registered and retrieved. An agent domain is a logical grouping of agent and their services, defined by their membership (registration) in a DF. Each domain has one and only one DF, which provides a unified, complete and coherent description of the domain. The DF lists all agents in the domain and advertises the agents' existence, services, capabilities, protocols etc. An agent may present in one or more domains via registration in one or more DFs. A domain in this context can have organizational, geo-political, contractual, ontological, affiliation or physical significance.

Moreover, the DF is a trusted, benign custodian of an agent directory. It is trusted in the sense that it must strive to maintain an accurate, complete and timely list of agents. It is benign in the sense that it must provide the most current information about agents in its directory on a non-discriminatory basis to all authorized agents.

Agents may register their services with the DF or query the DF to find out what services are offered by which agents. In this context, the DF registers but does not control the internal life-cycle of any agent.

The DF can restrict access to information in its directory, and will verify all access permissions for agents which attempt to inform it of agent state changes.

An AP can host several DFs (i.e. logical domains), or several APs can share one DF in offering a shared Yellow Page over multiple APs. The DFs in an agent environment should build up either a hierarchy or a federation in order to enable the agents to look for needed services in the global agent execution environment. To support such hierarchy or federation, each DF should be configured with the capability of local information retrieval and of delegating the search operations to other appropriate DFs in case of local information unavailability.

An *Agent Management System* (AMS) is a mandatory component of the AP which mainly manages the life cycle of all the agents residing on that AP. Only one AMS will exist in a

single AP and each AP must have an AMS.

The responsibility of an AMS includes the creation of agents, suspending/resuming and deletion of agents, and the decision on whether an agent can dynamically register on the platform (for example, this could be based upon agent's or AP's ownership).

Another key function of AMS is the *White Page* service for all the agents in the local AP. The AMS maintains an index of all the agents which are currently resident on a platform. The index includes an agent's name and their associated transport address for the AP. Both the agent and the Agent Communication Channel (ACC) can query the AMS to find out the technology dependent transport addresses of the destination agents in agent communications. To send an ACL message to the destination agent, the sender can

- issue the ACL message with the agent name, rely on the ACC to look up the real address and to deliver the message, or
- look up the destination's transport address and issue the message with this address.

In the second case the ACC can directly deliver the message without the help of the AMS. This kind of ACL message is generally more efficient especially within a large, distributed AP, where AMS itself can be remotely located to the communicating agents.

The ACC corresponds to the DPE middleware component within a TINA architecture ([94]). It offers the transport environment in which agents can communicate and co-operate with each other in achieving their objectives. The key function of such an ACC is to deliver agent communication (ACL) messages transparently from sending agents to the receiving agents. For this purpose the following functional supports are necessary:

- message transport,
- message routing,
- security,
- asynchronous messaging and buffering.

OMG IIOP is considered in this context as the default protocol for communicating messages between the ACCs.

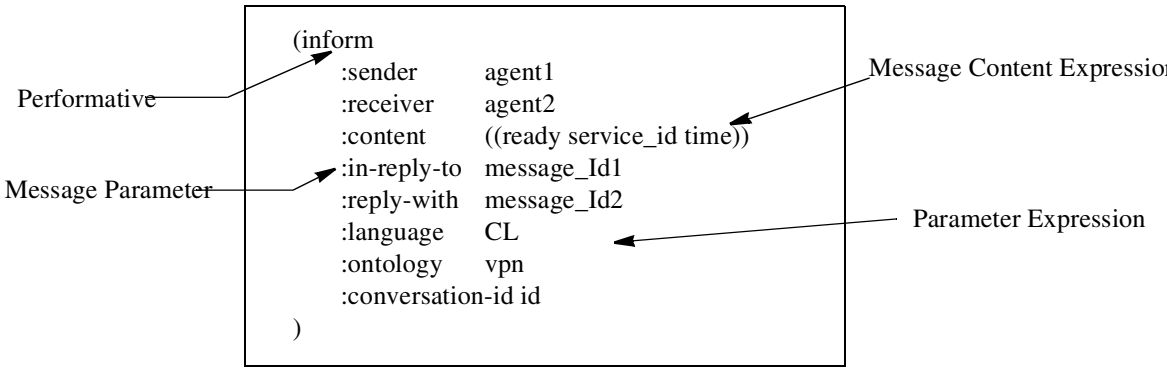
2.3.2.2. FIPA Agent Communication and Content Languages

FIPA Specification Part 2 ([24]) defines the ACL and CL to be used for communicating FIPA agents. Following the tradition of KQML, an FIPA ACL message has the basic structure in

figure 19.

The first element of the message is the name of the *peformative* (called *communicative act* in FIPA) for that message, which defines the basic illocutionary context of the content carried by the message. There then follows a sequence of message parameters, introduced by parameter keywords beginning with a colon character. One of the parameters contains the content of the message, encoded in the content language selected. Other parameters help the message transport service to deliver the message correctly (*sender* and *receiver*), help the receiver to interpret the meaning of the message content (e.g. *language* and *ontology*), or help the receiver to respond co-operatively (*reply-with*, *reply-by* and *conversation-id*).

Figure 19: Components of an ACL Message



The meanings of each parameter is described by the following table:

Message Parameter	Meaning
:sender	Denotes the identity of the sender of the message, i.e. the name or address of the agent which sends the message.
:receiver	Denotes the identity of the intended recipient of the message. Note that the recipient may be a single agent name, or a tuple of agent names. This corresponds to the action of multicasting the message. Pragmatically, the semantics of this multicast is that the message is sent to each agent named in the tuple, and that the sender intends each of them to be recipient of the content encoded in the message. For example, if an agent performs an inform speech act with a tuple of three agents as receiver, it denotes that the sender intends each of these agent to come to believe the content of the message.

Message Parameter	Meaning
:content	Denotes the content of the message; equivalently denotes the object of the speech act. The content of a message refers to whatever the performative/communicative act applies to. If, in general terms, the message is considered as a sentence, the content is the grammatical object of the sentence. In general, the content can be encoded in any language, and that language will be denoted by the *:language* parameter.
:reply-with	Introduces an expression which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously. E.g. if agent i sends to agent j a message which contains :reply-with query1, agent j will respond with a message containing :in-reply-to query1.
:in-reply-to	Denotes an expression that references an earlier message to which this message is a reply.
:language	Denotes the encoding scheme of the content of the action.
:ontology	Denotes the ontologies which are used to give a meaning to the symbols in the content expression.
:conversation-id	Denotes the conversation context of the message.

Different from KOML, which focuses on enabling co-operative management of knowledge-bases, FIPA puts more emphasis on supporting negotiations within a telecommunications environmentbh. For this purpose, some new performatives/communicative acts, like cfp, propose, accept-proposal, reject-proposal, together with some interaction protocols for deploying such performatives/communicative acts are specified. Some most important performatives/communicative acts, which will also be used as the basis for the agent solution for telecommunications applications to be presented in this thesis, are listed (in a simplified form) in the following tables:

– **inform**

Summary	The sender informs the receiver about some events or a specific situation, i.e. some statements hold in the environment perceived by the sender.

Message Content	A set of logical statements that characterize the situation in the environment.
	In this case, the sending agent: • holds that the set of statements are true; • intends that the receiving agent also comes to believe that the statements are true; • does not already believe that the receiver has any knowledge of the truth of the statements. The first two properties defined above are straightforward: the sending agent is sincere, and has (somehow) generated the intention that the receiver should know the statements (perhaps it has been asked). The last property is concerned with the semantic soundness of the speech act. If an agent knows already that some state of the world holds (that the receiver knows the statements), it cannot rationally adopt an intention to bring about that state of the world (i.e. that the receiver comes to know the statements as a result of the inform speech act). The sender is not required to establish whether the receiver knows the statements. It is only the case that, in the case that the sender already happens to know about the state of the reciever's beliefs, it should not adopt an intention to tell the receiver something it already knows. Of course it can not be excluded the case that a malicious agents trying to overload another agents by repeating some messages with the same facts. This is then a security violation that has to addressed by the resource control facility of the AP and the receiving agent. From the receiver's viewpoint, receiving an inform message entitles it to believe that: • the sender believes the statements that are the content of the message; • the sender wishes the receiver to believe these statements also; • the sender does not believe that the receiver believes the statements. Whether or not the receiver does, indeed, adopt the belief in the statements will be a function of the receiver's trust in the sincerity and reliability of the sender.
Examples	Agent i informs agent j that (it is true that) i has an alarm situation and can not recover automatically by its self: (inform :sender i :receiver j :content ((and (alarm :agent_id j :reason overload) (recover_failure))) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **cfp (call for proposal)**

Summary	The speech act of calling for proposals is used to request offers for the provisioning of a given service.
Message Content	An expression denoting the service action being requested and some constraints denoting the preconditions on the specific service.
Description	cfp is a general-purpose action to initiate a negotiation process by making a call for proposals for a given service. The actual protocol under which the negotiation process is established is known by prior agreement. In normal usage, the agent responding to a cfp should answer with a message giving its conditions on the provisioning of the service. Note that cfp can also be used to simply check the availability of an agent to perform some service actions.
Examples	Agent i calls for a proposal from agent j for a *connection service* (cfp :sender i :receiver j :content ((connection_service :type video :bandwidth 100Mbps :proposalNr 101)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **propose**

Summary	The speech act of submitting a proposal to offer a certain service.
Message Content	An expression representing the service that the sender is going to offer, and a list of constraints that describe the conditions on or even definitions of the service.
Description	Propose is a general-purpose action to make a proposal or respond to an existing call for proposal during a negotiation process by proposing to offer a given service subjected to certain conditions being true. The actual protocol under which the negotiation process is being conducted is know by prior agreement, e.g. via exchanging protocol definitions.

Examples	Agent i proposes to agent j that it can offer a connection service to j with certain parameters: (propose :sender i :receiver j :content ((connection_service :type video :bandwidth 100Mbps :proposalNr 101)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **accept-proposal**

Summary	The speech act of accepting a previously submitted proposal to offer a specific service.
Message Content	An expression representing the service and service conditions that the receiver has prevously proposed.
Description	Accept-proposal is a general-purpose acceptance of a proposal that was previously submitted (typically through a propose act). The agent sending the acceptance informs the receiver that it intends that (at some point in the future) the receiving agent will provide the service.
Examples	Agent j accepts the proposed service action from agent i: (accept-proposal :sender j :receiver i :content ((connection_service :proposalNr 101)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **reject-proposal**

Summary	The speech act of rejecting a proposal to offer some service during a negotiation.
Message Content	An expression representing or identifying the service that the receiver has previously proposed.

Description	Reject-proposal is a general-purpose rejection to a previously submitted proposal. The agent sending the rejection informs the receiver that it does not wish that the recipient provides the given service to the sender. Additional facts in the message can further identify the reasons for the rejection.
Examples	Agent j rejects the proposed action from agent i: (reject-proposal :sender j :receiver i :content ((connection_service :proposalNr 101)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **request**

Summary	The sender wants the receiver to perform some service action.
Message Content	An expression representing the action that the receiver is asked to perform, and a list of constraints that further describe the conditions and contents of the action.
Description	The sender is requesting the receiver to perform some service action. The content of the message is a description of the action to be performed, in some language the receiver understands. The action can be any action the receiver is capable of performing. An important use of the request act is to build composite conversations between agents, where the actions that are the object of the request act can contain further ACL messages.
Examples	Agent i requests agent j to reserve a communication channel: (request :sender i :receiver j :content ((reserve_connection :type video :bandwidth 100Mbps :user *a* :user *b*)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **cancel**

Summary	The action of cancelling some previously sent action request, whose processing can have temporal extent (i.e. not instantaneous).
Message Content	An expression representing or identifying the ACL message that is to be cancelled.
Description	Cancel allows an agent to stop the receiver agent from continuing to process a previously received message, and (depending on the autonomous preference of the agent) to recover any side effects resulted from process that message (e.g. messages generated by processing the current message).
Examples	Agent i cancels a service request to agent j: (cancel :sender i :receiver j :content (request :sender i :receiver j :conent ((reserve_connection :type video :bandwidth 100Mbps :user *a* :user *b*)) :language CL :ontology ontology1) :reply-with message_id1 :language CL :ontology ontology1)

– **agree**

Summary	The action of telling another agent that it agrees to carry out the service action as requested.
Message Content	An expression that identifies the action to be agreed upon.
Description	Agree is a general purpose agreement to a previously submitted request to perform some action.

Examples	Agent i informs agent j that it agrees to carry out the action requested: (agree :sender i :receiver j :content ((action :id id1)) :in-reply-to message_id2 :reply-with message_id3 :language CL :ontology ontology1)

– **not-understood**

Summary	The action of telling another agent that it did not understand the message that it has received.
Message Content	An expression that identifies the received message and the reasons why it was not understood.
Description	The sender receives a speech act which it did not understand. There could be many reasons for this. E.g. the ontology used in the message is not supported, or the received message is not the one the agent expects according to some agreed protocol.
Examples	Agent i informs agent j that the previous message was not understood: (not-understood :sender i :receiver j :content ((not-expected)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **refuse**

Summary	The action of refusing to perform an action requested, and explaining the reason for that refusal.
Message Content	An expression that identifies the received message and the reason why it was refused.

Description	The refuse speech act is performed when the agent cannot meet all the pre-conditions for the activity to be carried out.
Examples	Agent i informs agent j that the action requested was refused: (refuse :sender i :receiver j :content ((aborted :reason unknown)) :in-reply-to message_id1 :reply-with message_id2 :language CL :ontology ontology1)

– **failure**

Summary	The action of telling another agent that an action was attempted but the attempt failed.
Message Content	An expression that contains the reasons and parameters of the failure, and some facts that describe the context of the failure in the sender agent environment.
Description	The failure act is an abbreviation for infoming that an act was considered feasible by the sender, but was not completed for some reasons. The agent receiving a failure message is entitled to believe that the action is not done.
Examples	Agent i informs agent j that the action requested failed: (failure :sender i :receiver j :content ((aborted :reason unknown)) :in-reply-to message_id2 :reply-with message_id3 :language CL :ontology ontology1)

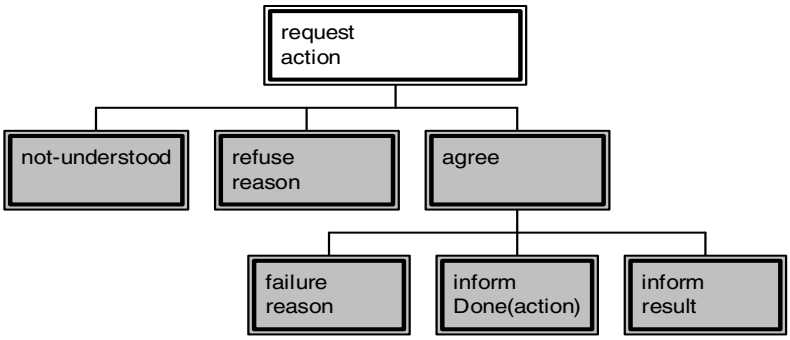
These performatives/communicative acts are accompanied by some basic protocols that define the typical patterns of conversations between agents, and the contexts in which individual communicative acts can be deployed. An agent can and should follow such pre-define protocols in order to achieve interoperability with other agents. Some examples of such FIPA

protocols are presented briefly in the following:

– FIPA-request Protocol

The FIPA-request protocol simply allows one agent to request another to perform some action, and the receiving agent to perform the action and to inform the result, or to reply, in some way, that it cannot.

Figure 20: FIPA-request Protocol



– FIPA-contract-net Protocol

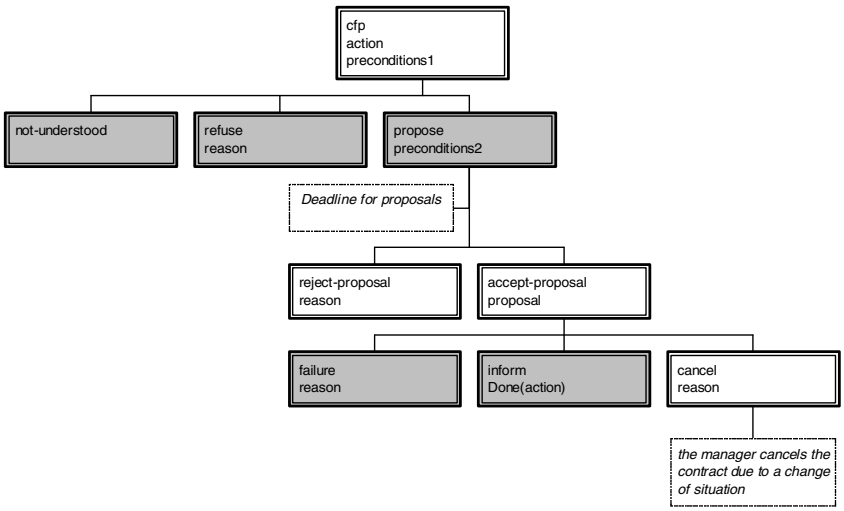
FIPA-Contract-Net is a minor modification of the traditional contract net protocol in that it adds *rejection* and *confirmation* communicative acts. In the contract net protocol, one agent takes the role of *manager*. The manager wishes to have some task performed by one or more other agents, and further wishes to optimize a function that characterizes the task. This characteristic is commonly expressed as the *price*, in some domain specific way, but could also be soonest time to completion, fair distribution of tasks, etc.

The manager solicits *proposals* from other agents by issuing a *call for proposals*, which specifies the task and any conditions the manager is placing upon the execution of the task. Agents receiving the call for proposals are viewed as potential *contractors*, and are able to generate proposals to perform the task as *propose* acts. The contractor's proposal includes the preconditions that the contractor is setting out for the task, which may be the price, time when the task will be done, etc. Alternatively, the contractor may *refuse* to propose. Once the manager receives back replies from all of the contractors, it evaluates the proposals and makes its choice of which agents will perform the task. One, several, or no agents may be chosen. The agents of the selected proposal(s) will be sent an acceptance message, the others will receive a notice of rejection. The proposals are assumed to be binding on the contractor, so that once the manager accepts the proposal the contractor acquires a commitment to perform the task. Once the contractor has completed the task, it sends a completion message to the manager.

The protocol requires the manager to know when it has received all replies. In the case that a

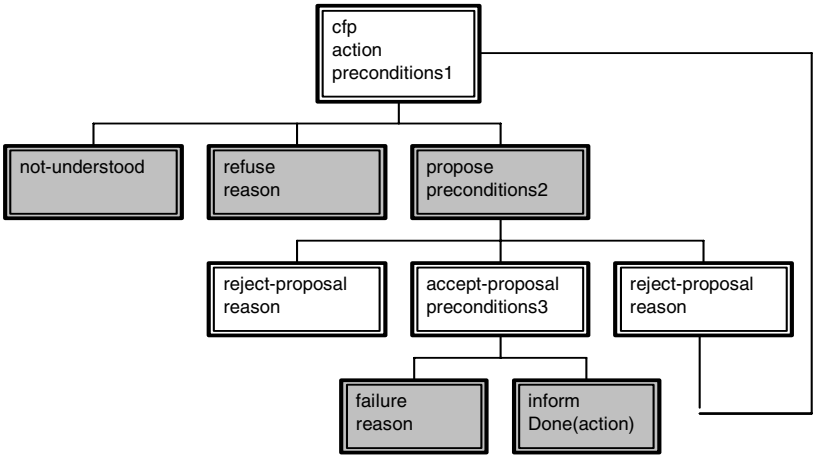
contractor fails to reply with either a *propose* or a *refuse*, the manager may potentially be left waiting indefinitely. To guard against this, the *cfp* includes a deadline by which replies should be received by the manager. Proposals received after the deadline are automatically rejected, with the given reason that the proposal was late.

Figure 21: FIPA-Contract-Net Protocol



– FIPA-iterated-contract-net protocol

Figure 22: FIPA-iterated-contract-net protocol



As depicted in figure 22, the FIPA *iterated contract net* protocol extends the basic contract net by allowing multi-round iterative bidding. As above, the manager issues the initial call

for proposals with the *cfp* act. The contractors then answer with their bids as *propose* acts. The manager may then accept one or more of the bids, rejecting the others, or may iterate the process by issuing a revised cfp. The intent is that the manager seeks to get better bids from the contractors by modifying the call and requesting new (equivalently, revised) bids. The process terminates when the manager refuses all proposals and does not issue a new call, accepts one or more of the bids, or the contractors all refuse to bid.

FIPA does not restrict the deployment of any CLs in the ACL messages, although it does specifies some vague requirements on such CLs. On the other hand, FIPA has specified a default content language - the semantic language (SL), which is in the traditions of KIF and predicate logics.

2.3.2.3. FIPA Ontology Service

FIPA 98 Specification Part 12 ([31]) specifies a framework in which ontologies can be managed within the FIPA APs. The core component of this framework is the concept of *ontology agent*, which is an IA that offers the ontology service to other agents. By contacting the ontology agent, an agent can register, retrieve, query and modify ontologies and ontology definitions.

FIPA ontology service definition is based on the OKBC ([5]) knowledge model, and all the ontology service functions correspond to the manipulation of OKBC knowledge. With the inherited heterogeneity in the ontology representations in the telecommunications, e.g. an ontology can also be represented as a group of Java classes, such an ontology service definition is still too restricted. The lack of wide acceptance and application of the OKBC framework in this application context at the moment can further hamper the adoption of FIPA ontology service in the industry.

2.3.3. IA Platforms

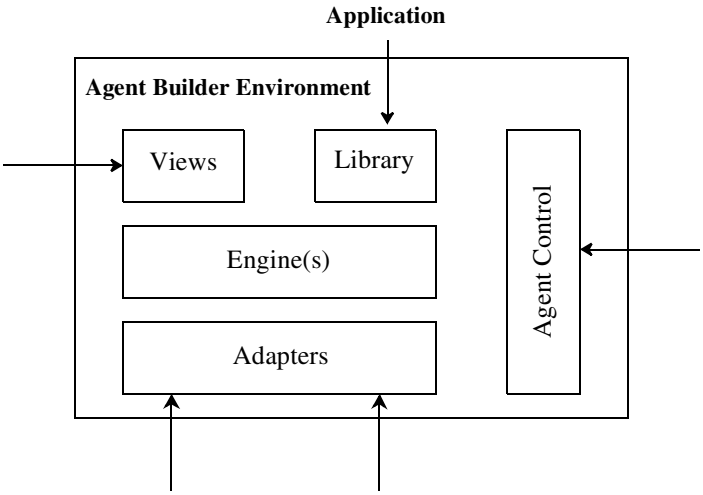
The IA community is characterized by its heterogeneity in terms of the backgrounds and application or technological focuses. Developments in this context range from proprietary systems and technologies for specific applications to generic IA platforms that aim at supporting wider application areas.

Most such IA platforms are designed for IA applications based on speech act or the similar IA communication mechanisms. The following gives an overview of the major features of some of these platforms.

2.3.3.1. IBM Agent Building Environment Developer's Toolkit

IBM Agent Building Environment Developer's Toolkit is a development and execution platform that supports the building of knowledge-based IA-based application. The basic architecture can be depicted in figure 23.

Figure 23: ABE Architecture



The ABE architecture consists of the following components:

- *Agent Control* for initialization and overall control of the running agent.
- *Engines* for interpreting the instructions to the agent. A rule-based engine is provided. It inferences on sets of rules and facts.

- *Adapters* for associating or wrapping applications to the agent, and providing the actions that are needed by the agent to carry out its work in the application world. Different protocols are supported by the adapters, e.g. NNTP for USENET news groups, HTTP for the web, a File Adapter for file interaction, and a Time Adapter for basic date/time services.

- *Views* for establishing and modifying the instructions for the agent. The IBM Agent Building Environment Developer's Toolkit includes a basic rule editor, but the anticipation is that views will grow to include a rich variety of user-oriented methods for communicating user instructions to the agent.

- *Library* for managing the caching and persistent storage of the agent instructions (currently rules and facts) that the Views produce and the Engines use. The current IBM Agent Building Environment Developer's Toolkit Library allows for storing and retrieving inferencing rules and long term facts from persistent storage, as well as functions for organizing this inferencing material into manageable groups and logging agent activities.

IBM Agent Building Environment Developer's Toolkit supports mainly the KIF based representation of rules and facts. KQML is not supported directly by the IBM Agent Building Environment Developer's Toolkit, but by regarding KIF as the basis for KQML (i.e. KQML as a subset of KIF expressions), the IBM Agent Building Environment Developer's Toolkit can also be easily used in developing KQML-based IA applications.

Agent programming can be done either in C++ or Java with the IBM Agent Building Environment Developer's Toolkit.

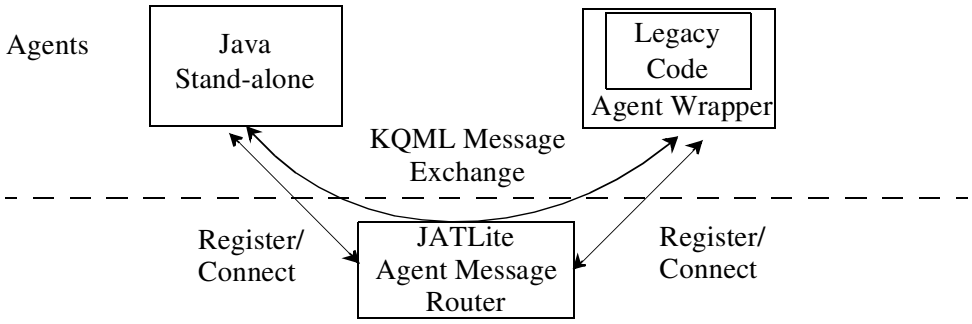
Although IBM Agent Building Environment Developer's Toolkit is no longer been supported by IBM, it has played and is still playing an important role in the development of IA technology, and has still strong influence on the many other developments in this context.

2.3.3.2. JATLite

JATLite ([84]) is developed by Stanford University based on the tradition of KQML and the Java Agent Template ([36]) from Stanford. It is basically a Java package that allows the users to quickly create new IAs that communicate robustly over the Internet. The basic infrastructure of JATLite is shown in figure 24, where the key component is the Agent Message Router

(AMR).

Figure 24: JATLite Infrastructure



An agent in this context, which can be either a stand-alone Java agent or be resulted from wrapping some legacy code/application, can register with the AMR using a name and a password, connect/disconnect from the Internet, send/receive KQML messages and transfer files with FTP. AMR is responsible for routing the messages correctly from its source to the destinations. It supports among others the asynchronous buffering and delivery of messages in case the receiver can not receive a message.

Basic communication protocols for JATLite are TCP/IP, SMTP and FTP.

JATLite does not endow agents with specific intelligence or knowledge processing capabilities. The developer is left free to use whatever theories and techniques that are best suited for the target application or research.

Mobility and FIPA conformance are not supported by JATLite.

2.3.3.3. Bee-gent

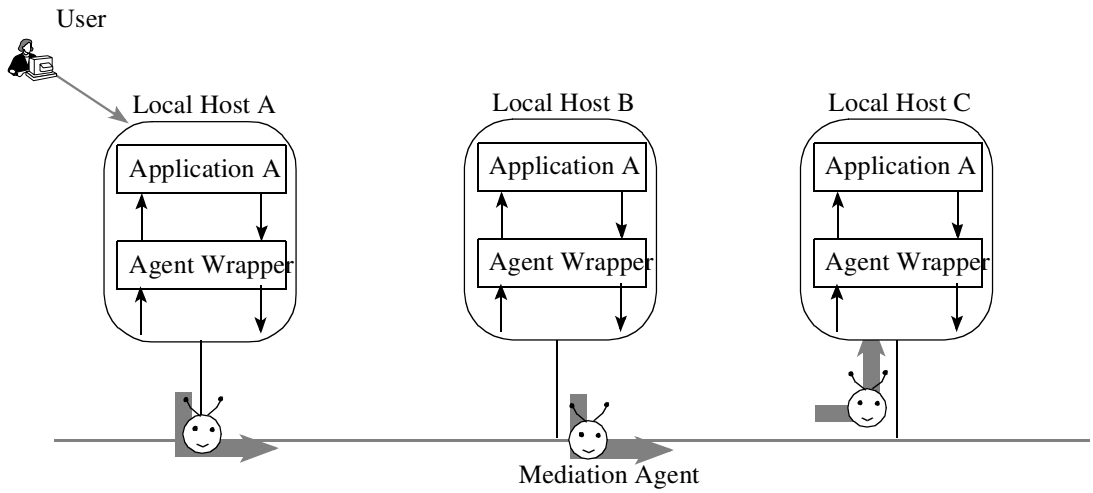
Bee-gent ([95]) is a purely Java-based IA framework developed by Toshiba, Japan. Different from other agent platforms, Bee-gent completely “agentifies” the communication that take place between software applications. The applications become agents, and all messages are carried by agents.

The Bee-gent architecture can be described by figure 25.

Two kinds of agents are envisaged in this context. *Agent wrappers* are used to agentify existing applications, while *mediation agents* support the inter-application co-ordination by hiding all communications. The mediation agents are MAs that move from the site of an application to another where they interact with the agent wrappers. The agent wrappers themselves manage the states of the applications they are wrapping around, invoking them when necessary. The inter-application co-ordination is handled through the agent wrappers gener-

ating and receiving requests, which are transported around by the mediation agents. In fact the mediation agents do more than just transporting the messages: they are able to respond to the nature of the request to determine the best course of action (e.g. to select and visit the best server).

Figure 25: Bee-gent architecture



Different mediation services are supported by the mediation agents. The basic services are workflow management, schedule co-ordination and WWW server integration. Wrapper agents are provided for WWW servers, JavaBeans, Oracle, Microsoft Access, LDAP clients and OKBC clients.

HTTP is used by Bee-gent as the baseline protocol for agent communications. XML is used for encoding FIPA ACL, in order to ease the integration of WWW and Internet-based applications.

Bee-gent is under the procedure of being transformed to a FIPA-conformant platform.

2.3.3.4. JIAC

JIAC (Java Intelligent Agent Componentware -[17]) is a Java-based agent platform developed by the DAI Lab of the Technical University Berlin, with the key application area in the context of electronic commerce and marketplace. A core idea in this context to implement MAs as specific speech act IAs that have the extra capability of migration among different marketplaces.

JIAC supports KQML for agent communication. Migration to FIPA conforment platform is also planed. Baseline transport protocols for agent communications and migrations include

TCP/IP, CORBA/IIOP and Java RMI.

JIAC does not mandate a specific CL, but Lisp/List-oriented syntax, which is similar to KIF, is frequently used in the applications.

2.3.3.5. JADE

JADE (Java Agent Development Environment - [2]) is a Java-based agent platform aiming at supporting the FIPA specifications for interoperable intelligent multi-agent systems. It implements the FIPA reference model as specified in [23]. The JESS expert system schell ([35]) is used as the default environment for implementing the intelligent applications behind JADE agents.

Mobility is not supported in JADE.

2.4. Harmonizing the IA and MA Paradigms

As discussed above, although both the MA and the IA paradigms are based on specific forms of enhanced interoperability among distributed and autonomous applications, they have different emphases in the technologies and have therefore different objectives in the application areas. As a result, the two paradigms have different advantages or disadvantages in the specific applications, and each paradigm has its most appropriate application contexts.

Generally speaking,

- the MA paradigm and technology focus on the *increasing the performance* of a distributed solution by reducing the traffic load, dependency on the network, operational delays, and requirements on the machine resources, while

- the IA paradigm and technology focus on *enhanced intelligence* (e.g. via a knowledge-based approach) in interpreting the knowledge contexts of the co-operation information, and enable, via such enhanced intelligence, solutions for more complicated problems in the dynamic and complex environments.

Correspond to this heterogeneity, MA paradigm typically adopts some low level programming language for the representation of the agents, while IA co-operation messages are typically based on high level, knowledge-based or rule-based representations.

2.4.1. Feature Divergence in Agent Paradigms

The different emphases of the agent paradigms determine the differences in features and applicability of the two technologies in the telecommunications applications. More specifically, we have the following divergence:

***Lifecycles*:** An IA communication message will usually be accepted and processed by the receiving IA. Its lifecycle terminates at its receiver.

A MA on the other hand can migrate among several agent systems. Therefore the lifecycle of a MA can span over multiple receiver application environments/network sites.

The lifecycle of a MA is therefore typically longer than the lifetime of an IA communication message.

***Complexity*:** IA communication messages are typically much smaller than MAs (especially with their execution states) and therefore easier to be transported and managed.

IA environment on the other hand requires more complicated and expensive component like inferencing engines. IAs are therefore heavier than MAs or MA agent systems in terms of processing and memory capabilities.

***Reliability & Performance* :** With the enhanced autonomy, MA helps to reduce the dependency on the constant availability of underlying network connections and on the source of the MA. In this way, MA technology enhances some performance features in the distributed solutions, including the reliability/ robustness against network failures, decentralization/balancing of processing loads etc.

Although possible, the IA technology generally does not put emphases on offering such performance enhancement.

***Adaptability* :** The knowledge contained in the IA messages can be easily absorbed by the receiving IA and integrated into the knowledge (like the rule-base) of the receiving agent, making IA technology more appropriate for accumulative adapting IA knowledge and functionality. This is not so easy with MAs. However, MAs, with the prolonged life cycles, can sometimes be used to modify/replace the remote applications or their components (autonomous software downloading/dynamic software configuration).

Richness of Interaction Protocols: By identifying the knowledge contexts in agent co-operations, IA paradigm can provide a richer set of semantically standardized interactions between software systems than the MA paradigm (where only one operation for MA migration is being standardized).

***Binding AI alike Technologies*:** IA communication, with its strong association to AI, can easily support the bindings of AI alike technologies into the individual static agents. This feature, combined with the advantages of a knowledge-based or rule-based approach, can increase the flexibility/tolerance/robustness of the co-operation/negotiation among agents.

***Security*:** MA relies on the trustfulness of its hosting agent systems for the reliable realization of its goals. This opens the door for some security violation at foreign hosts. E.g. an host can try to modify the code or to execute the instructions in a different way.

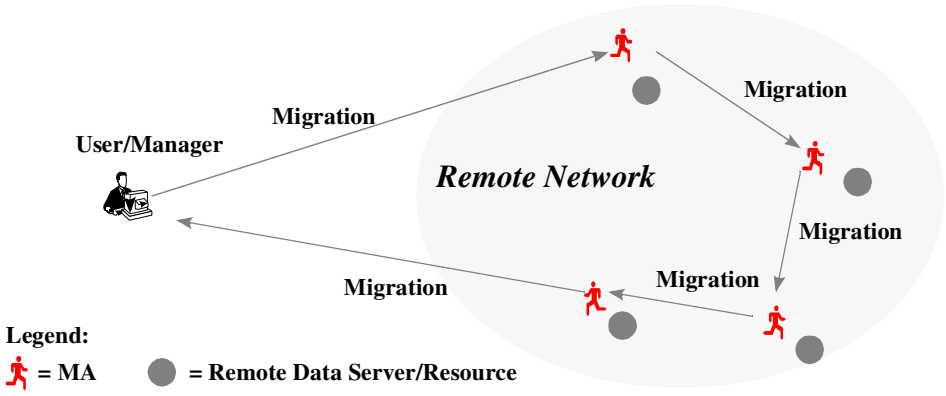
Agent interaction based on IA technology relies on the autonomous intelligence of the agent in realizing the co-operations. Most importantly, an agent will not base its actions on the assumption that the partner will do exactly what it wants, e.g. the partner can modify the co-operation request or implement it in a different way. As a result, the IA paradigm can eliminate some possibilities for security violations.

2.4.2. The Diverging Application Areas of the Agent Paradigms

The divergence in the features of the IA and MA agent technologies means each technology will have its associated advantages and disadvantages and is appropriate in certain application areas.

The key capability of the IA paradigm, with the possibility of understanding knowledge contexts, enables the definition of a rich set of interaction and negotiation protocols, and supports the dynamic negotiations for co-operation relationships. Such a capability make IA a more appropriate approach for high level, highly autonomous telecommunications applications ([114]), such as business and services management, e-commerce. The key characteritics in these application contexts are requirements for higher intelligence in dealing with the autonomous or even foreign application environments and co-operation partners to achieve the aggregated goals of the IAs.

Figure 26: Remote Data Mining and Resource Management



MAs, with their improvement in run time reliability, robustness and performance with respect to the distributed network resource availability and distributions, are frequently used as a solution for the globalization of applications. In this context, we usually have the following characteristics in the application environment [89]:

- unreliable, slow or expensive network connections between the user/manager and the servers/resources,
- accommodating mobile users or managers which are not continually connected,
- higher number of remote servers/resources.

As pointed out above, MA technology, via MA autonomy, can help to reduce the number of remote interactions and the dependency on the network and on the client applications (users or managers). Therefore MAs are more appropriate for such application scenarios, while IAs, with their emphases in service intelligence, does not offer a sufficient solution.

Typical application scenarios for MAs are remote data mining, and remote resource management. One example is remote subnetwork management (figure 26).

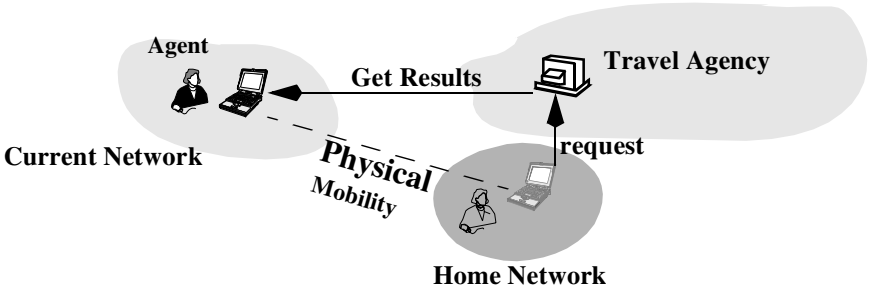
Another category of application scenario for MAs are telecommunications applications with *Physical Mobility* (figure 27), i.e. physical mobile equipments. E.g. a businessman has installed a Personal Assistant (PA) Agent in his Laptop and asked it to negotiate with some travel agency for organizing his next business trip. The PA sends the MA to the travel agency, which contains all the necessary information for finding out an optimal arrangement for the new business trip. After this, the businessman takes the laptop and flies to another country. During his travel, the PA is no longer connected to the network and no interactive co-operation is possible. The Travel Agency has to work with the MA and to produce a response. After the businessman arrives in the destination, he can plug his laptop in a new network and try to pull the result either directly from the Travel Agency, or from his home network. In fact, due to bandwidth and availability restriction in his new environment, direct interactive negotiation with the Travel Agency from the laptop can be unrealistic.

One of the most serious problems in the application of MA technology within an open market is related to the security of the MA itself. Because a MA migrates to a remote agent system and must be executed in that environment, the hosting agent system must access the codes (bytecodes or binary codes) of the MA, i.e. such codes must be open to the hosting agent environment. As MAs are typically

- short (light weight) and
- well known (frequently reused by a large group of application users) programs,

it is relatively easy to be read and analyzed by people or programs in the hosting environment, who want to make profit from such analyses.

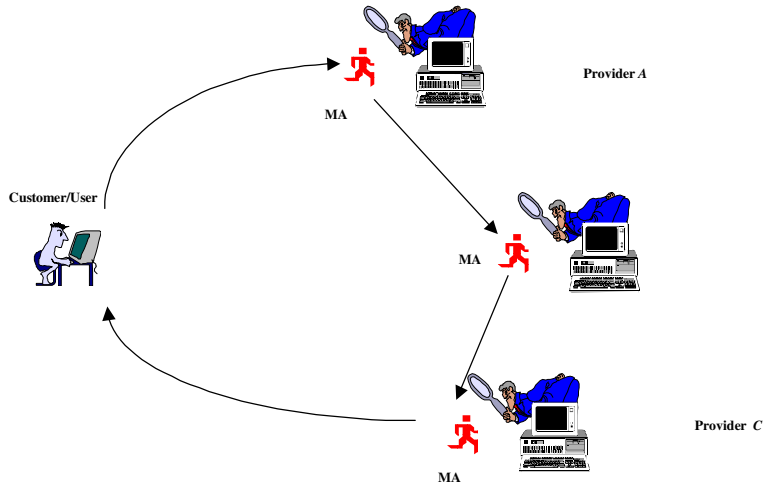
Figure 27: Physical Mobility



Suppose, as depicted in figure 28, a customer wants to buy a laptop and sends a MA which visits some potential sellers/providers to get the best price for the good. Suppose provider *A* offers the price 1000 US$, the provider *B* offers 2000US$. After visiting these providers the MA should note down that the current lowest price is 1000US$ from provider *A*.

In a normal case, when the MA visits the third provider (provider *C*), it will query the price offered by *C*, and gets the response, e.g. 3000 US$. Now the MA concludes that the best price is 1000US$ from *A* and report this to the origin customer. The customer then issues a request to *A* for buying that laptop.

Figure 28: Business Negotiation Based on MAs



However if the provider *C* is not a honest person/application, he can try to analyze the incoming MA and find out the position where it has noted/stored the current best price. It can then get the bid by doing the following:

– modifying the number to 3000.99 US$, so that *C* will have a sucessful bid with 3000 US$, or

– offering a price of 999.99 US$

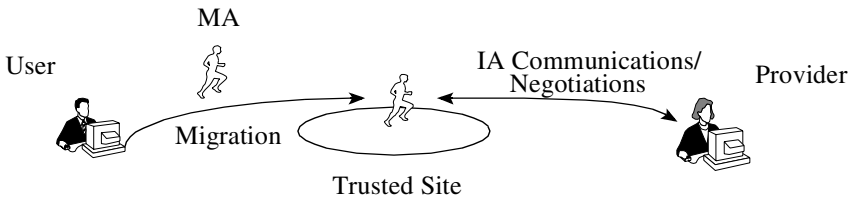
In reality, the hosting application/person can also make profits via some other, minor security violations. E.g. it can save the MA for some later analyses by the provider's application programmer/manager. The programmer/manager can then find out the negotiation strategies of the MA and use such information to adapt the algorithms of the provider applications. Or they can sell such information to a third party for money.

Currently there is no efficient and effective solution to this problem. MA should be therefore only sent to *trusted agent systems/environments*. In a global, open telecommunications market, however, such trustfulness is very difficult to be checked and guaranteed. For security/privacy critical applications like business negotiations, it is in many cases impossible to send the MA directly to the providers.

IA technology, on the other hand, does not have this difficulty. After sending an ACL message, the sender will check the usability of the response messages based on the conversation contexts. Usually it doesn't matter if the receiver analyzes (it should anyway understand the whole message) and modifies the received message.

The compromised solution in this context is to send the MA to a trusted site which is near the real provider. From this site the MA can negotiate with the provider via IA communications (figure 29).

Figure 29: Co-operating the IA and MA Paradigms



2.4.3. Integrating the IA and MA Technologies

From the above analysis, we can conclude that the IA and MA paradigms must co-operate in many telecommunications applications to offer an appropriate solution. A framework for harmonizing and integrating the two technologies and their associated standardizations can play an important role. Such a framework will be necessary at least for achieving the following goals:

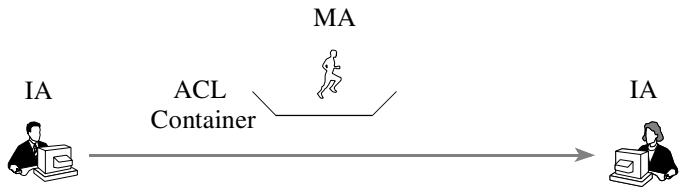
– to enable the utilization of the advantages from both technologies in the heterogeneous environments,

– to enable the interoperability between agent-based applications from the different religions.

Based on the analysis of the basic agent features, we can envisage a variety of possibilities for harmonizations/integrations.

FIPA 98 specification part 11 ([30]) specifies a framework for the integration of MA paradigm within the speech act communications among FIPA IAs. The idea in this context is to consider a MA as the content of some dedicated ACL messages, as described in figure 30.

Figure 30: FIPA Mobility Support



ACL messages of some specific performative types are used as the container for transporting MAs between the IAs. The receiving IA, which can be an agent system, is responsible for managing the lifecycle, i.e. execution, termination or migration of the MA. This approach is preferred by the IA community as it imposes no change to the current FIPA standard.

However, allowing MA to appear in the ACL messages can cause extra complexity and even confusions in the implementation of the IAs, partially because a MA is typically implemented in a different language than the normal message content, and has to be managed differently.

Another approach, which has been adopted by some IA platforms like JIAC[17], is to regard MA as IA with the additional mobility capability. This second approach can be considered as a direct extension to the MA paradigm with the IA communications. It can be easily accepted by the MA community as it poses no changes to the MA paradigms. Besides, as will be discussed later in more detailed, the paradigm can be extended to allow the direct integration of the FIPA and OMG MASIF IA/MA standards.

2.5. Summary

In this chapter we have given an overview of the state of the art agent technologies and their features or roles in the information processing and telecommunications. It can be concluded that the speech act based IA paradigm and the MA paradigm offer the promising and also sufficiently mature technologies for building the framework for telecommunications applications.

There are still some issues that are not yet sufficiently studied in the literatures. Such issues will define the key mission of our work to be presented in this thesis. Among other, solutions are still needed for

- integration of IA and MA paradigms based on standardized technologies, in order to maximize the interoperability among the different agent-based telecommunications solutions, and to maximize the applicability of such solutions,

- dynamic adaptation and configuration of agent co-operations based on knowledge, ontologies and mental models of the agents, in order to support the dynamic relationships and the intelligent understandings among the heterogeneous telecommunications applications.

The standardization of the agent technologies, especially the FIPA standard, will be used as the starting point for the developments to be presented in this thesis.

The following chapters will present our agent-based solution for telecommunications applications, which aims at meeting the new challenges of the dynamic, heterogeneous, globalized and mobile telecommunications environments.

CHAPTER 3

The Solution

3.1. Introduction

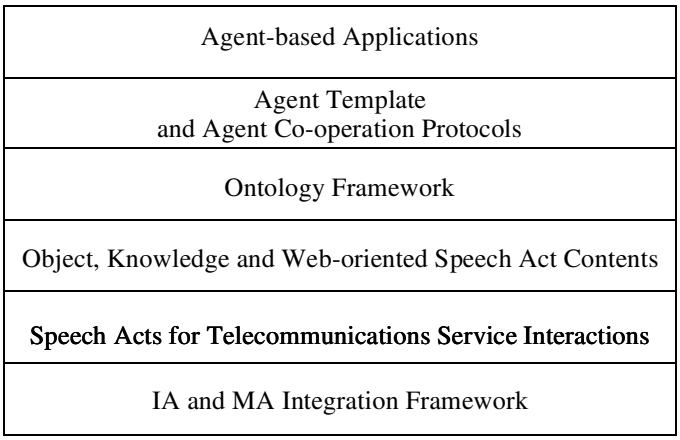
The solution for the telecommunications problems in the dynamic, heterogeneous, gloablized and mobile environment, which is to be presented in this thesis, focuses on enabling and utilizing knowledge-based and dynamic agent interoperability in co-operating autonomous telecommunications applications.

To cope with the versatility of the characteristics and requirements of telecommunications applications, and in order to offer a sufficient solution to deal with the complexity of the typical telecommunications environments like the advanced VPN service environments, such agent interoperability will have to be based on a group of basic technologies. This aggregation of basis technologies build up an integrated *agent framework*, which enables the development of new services and service features based on agent technology.

This agent framework has a layered structure as showed in figure 31, where each lower layer supports the implementation of the higher layers above it.

The following sections will first discuss briefly the individual layers of the framework and their roles or inter-relationships in supporting the overall agent-based solution. A more detailed presentation of the architecture and components that implement the framework will be made in the next chapter.

Figure 31: Agent Framework

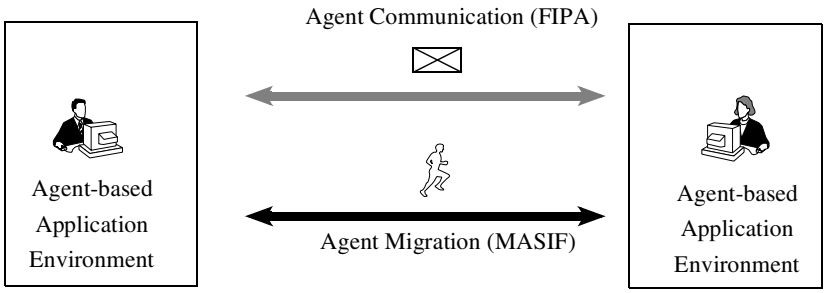


3.2. The Integrated IA and MA Framework

As discussed above, both IA and MA have their distinguished advantages and disadvantages in the different application contexts. An agent platform therefore should support both kind of agent-based interoperability in order to support sufficient flexibility in the applications.

A further requirement in this context is the adoption of standards to maximize the interoperability among heterogeneous applications and environments. Based on these considerations, our agent framework for agent interoperability can be depicted in figure 32.

Figure 32: Integrated IA and MA Framework



Basically, this interoperability among autonomous, agent-based applications is realized on two layers:

– the agent migration layer supports the migration of MAs among the different network sites, while

– the agent communication layer supports the speech act communication among agents, independent of whether they are IAs or MAs.

Similar to some existing IA platforms like JIAC, MAs are considered as a special kind of agents that have the additional capability of migrations. Any agents that do not move during their lifetime (depending on the requirements of the application) can be regarded as static agents.

To enable the interoperability among the heterogeneous application environments,

– the agent migration layer will be based on the OMG MASIF standardization, while

– the agent communication layer will be based on the FIPA standardization.

In this way, an agent can move to any trusted environment, and carry out there its co-operation or negotiation with its local or remote peer parties. Such a paradigm supports the utilization of both local operations and the intelligent/autonomous co-operations or negotiations.

3.3. Speech Acts for Telecommunications Service Interactions

As the core application domain of the agent framework lies in the telecommunications applications, our definition of ACL and the selection of the speech act performatives will be based on the needs of telecommunications interactions.

As discussed in Chapter 1, Every telecommunication oriented application within the emerging Information Infrastructures can be regarded as value chains. The items passed through such chains (between a pair of roles) can be regarded as abstract services (e.g. resources or products can be regarded as a special kind of services). The most elementary interaction type in this context can always be abstracted to the level of service provisioning and utilization between a pair of roles, i.e.

– the *user* or the *consumer* (the end user in the value chain), and

– the *provider* of the service.

The selection of speech act performatives therefore will focus on enabling the generic initial (i.e. between foreign agents) and accumulative service co-operation between the (potential) users/consumers and the providers, or their agents. The detailed, application-specific co-operation knowledge or contexts will be treated in the message content.

For this purpose, a subset of the performatives from the FIPA standardization is selected as the basis set for the speech act-base service negotiation and for the deployment in service-

oriented telecommunications applications.

The framework will further support the dynamic definition of new performatives, and the deployment of such performatives in dynamically defined protocols. Such a feature further increases the flexibility and adaptability of the agent-based solutions.

3.4. Object-, Knowledge- and Web-oriented Content Languages for Agent Communications

By supporting the representation and transmission of domain specific knowledge, CL plays an especially important role in realizing the dynamic adaptability in agents' intelligence and co-operation relationships. A CL implements in this context a knowledge representation framework for the dedicated application area. For this purpose, such a CL will have to ease the task of dynamically modifying and extending some existing functionality/intelligence of the agents via encoding/absorbing the knowledge contained in the exchanged messages.

Compared to the RPC-oriented data representations and communications, knowledge-based paradigms, combined with the Object-Oriented modeling technology, can offer better solutions in this context. Generally speaking, most knowledge representation paradigms like rule-based systems, semantic/conceptual networks, constraint networks, support extensibility and reusability by allowing adaptations or extensions of the existing knowledge-bases with other (new or existing) knowledge about the behaviors and semantics. An Object-Oriented approach further supports extensibility and reusability via hierarchical inheritance.

Another issue that significantly affects the selection of CL framework relates to the Internet-based deployment of agent-based telecommunications solutions.

Within the global telecommunications environment, Web and Internet-based technologies, and the associated WWW infrastructure are playing an increasingly important role in the management of information and services. An agent-based solution in this context must therefore have the capability of utilizing and integrating these Internet/Web technologies and applications.

At the moment, HTML is still the dominant technology for structuring and presenting information in the Web. However, XML/XSL ([101], [102]) is now emerging as the new generation of Internet/Web presentation language that promises to replace HTML in the future. While being a tag-based language similar to HTML, XML enables the user to specify arbitrary new tags in the documents and use XSL or other technologies to specify the semantics in the presentation and processing of new tagged document elements.

In fact XML is used as a meta language for defining the specific languages or models (with the dedicated tags) in specific application contexts. Such definitions are called DTDs (Document Type Declaration [101]). Using XML DTD one can practically implement any language

models in XML by emulating the associated syntactical structures.

Adapting XML for agent communication or content language will

- ease the integration of agent-based applications with other Web and Internet applications,
- support human readable presentation of agent communication messages in the Web,
- enable dynamically configurable XML/HTML-based human interfaces,
- support the management of information, including knowledge and ontologies within the WWW infrastructure, e.g. via URLs and Web servers.

Within a telecommunications service and resource management environment, the Resource Description Framework (RDF) which is newly standardized by W3C ([100]), plays an especially important role. RDF models the information in the environment as resources. As the agents in our framework talk to each other or negotiate with each about resources like services, capabilities and information objects, which are modeled as resource objects, RDF can be used as the representation language for agent communication messages in the Web. Among others, FIPA is also considering the option of using RDF as agent communication content language ([33]).

The agent framework presented in this thesis adapts a RDF-based knowledge representation for modeling the behaviors and knowledge associated to the telecommunications resources.

To achieve the simplicity in the design and deployment of the language, we adopt a very generic view in the dedicated CL. In this context, we regard every information entity (like service, resource, action, event or even relationships) as a *resource object* (called resource hereafter for simplicity) that exist in the agent's environment. The core of our CL is to model the concepts (i.e. classes) and instances associated to such resources.

Moreover, the CL extends the standard RDF knowledge model and aims at supporting both entity-oriented and the dynamic behavior-oriented knowledge. The extended RDF model allows, besides the definition of resource, resource classes and resource properties as supported by the standard RDF, the definition of constraints or rules that characterize the behaviors of resources and their logical relationships.

3.5. Knowledge-based Ontology Framework

According to FIPA ([23]):

an ontology gives the meanings to symbols and expressions within a given domain language.

In the context of the agent communications such meanings will be used by the agents which sending or receiving the messages to determine

- when and how these symbol and expression should be used in a message, and
- how to interpret the symbol and expression in the incoming messages for generating appropriate responses.

Following this definition of ontology, it is obvious that ontologies are related to the semantics and the expected behaviors associated to the words and vocabulary used in the agent communication messages. However, for its deployment an ontology does not necessarily specifies a complete semantic model of the underlying resources. Due to complexity of the telecommunications application environment and resources, and due to the general limitation in efforts for specifying and interpreting ontologies, ontologies are typically in-complete, i.e. with partial characterizations of the meanings of message symbols and expressions, which aim at the specific application domains and at supporting agents' decisions and actions.

It is in this sense that an ontology can be better regarded as a piece of domain-specific knowledge about the world and its resources, derived from the experiences of the human experts.

The set of ontologies supported by an agent characterizes the knowledge of the agent about the external environment and about itself. An ontology framework should be able to support the construction and maintenance of such knowledge, and also to support the key requirements and features of knowledge representations. These key requirements or features include reusability and extensibility of the ontologies.

In the agent framework of this thesis, an ontology is considered as a set of descriptions about the service resources, and their relationships or the constraints upon such resources. Such an ontology specifies some knowledge on the behaviors of the resources and the logical relationships among such resources, and characterizes in this way an *abstract view* on the environment. Such an abstract view specifies the context in which the resources can be deployed. Such an ontology framework is therefore considered as *context-oriented.*

As one abstract view can be based on another abstract view and extends that view, we have a hierarchical structuring of the ontologies in the knowledge universe, where children ontologies inherit all the knowledge from the parent nodes.

In this way, we have an Object-Oriented-alike development paradigm of the knowledge-based ontologies, with all the advantages like reusability, extensibility and accumulative con-

struction associated to the Object-Oriented approaches.

Agents in the telecommunications environment can get the necessary ontological knowledge either directly from the peer co-operating partner, or, which is the case in most application environments, from a trusted *ontology agent (OA)*. This ontology service agent offers an ontology service to the agents operating in the telecommunications environment.

Different from FIPA specification of the ontology service ([31]), where mainly OKBC-oriented ontologies are to be managed, our ontology service offers a generic support which does not restrict the representation language and methodology (except for the hierarchical structuring) of the ontological knowledge.

By combing the ontology framework with the RDF-based representation of the ontology knowledge in the Web environment. This ontology service also supports the storage and maintenance of the ontologies within the WWW infrastructure.

3.6. Agent Template -The Abstract View of Agents for Interoperability

An agent in a telecommunications service environment operates autonomously according to its intelligence concerning its internal goals and the possible behaviors for achieving such goals.

To enable and guarantee successful co-operation among agents, such goal-oriented behaviors have to conform to some social rules or constraints that are assumed in the specific application environment. These social rules and constraints can be regarded as the elementary ontology to be supported by any agents that what to participate in the service relationships.

Because each agent has to support the elementary ontology in its service co-operations, this ontology, which is usually coded into the basic behaviors of the agent, implements in fact an generic template, or abstract view for the agent. This template implements the minimal and mandatory behaviors of the agents in its co-operation with the environment within the service value chains.

The template of the agents and the associated elementary ontology serve as the basis for the definition and interpretation of other service co-operation ontologies. It is in this sense that the associated abstract view can be considered as an agent theory (or model) which supports the characterization of more sophisticated and specific behaviors of the agents.

Within the agent co-operations, the agent template can be used to bootstrap service negotiations between foreign agents or to support the accumulative extension and optimization of service co-operations.

Within our agent framework, the template of agent is mainly defined via a set of mental attitudes that are mandatory in the service interactions, and the behaviors associated to such

mental attitudes which are generally assumed in a service and business environment.

More specifically, the mental attitudes are represented by some expressions in the CLs, i.e. a resource class in RDF. Each agent, based on its internal intelligence, will have to support the transparent evaluation of such expression by making decisions on its own attitudes.

One key application of this agent template and the associated elementary ontology is to support the definitions of service co-operation speech acts by relating the rational interaction behaviors to the mental attitudes. E.g. if action X satisfies the goal of agent A and if B request this action, A should reply to B with an *agree* message. New speech act types can be dynamically defined and deployed in this way.

Such a freedom, which is not supported by current IA frameworks, provides another level of flexibility in the telecommunications applications.

3.7. Summary

This chapter presents briefly our agent framework, which aims at offering a more appropriate solution for the telecommunications problems in a dynamic, heterogeneous, globalized and mobile environment.

The key idea behind this framework is to integrate a number of agent-oriented technologies on different layers and to have all these technologies work together in support the dynamic and knowledge-based interoperability among the autonomous telecommunications applications.

Compared to the existing framework for either MA or IA-based solutions in telecommunications, our framework put more emphasis on

- integrating MA and IA technology based standardized technologies,
- supporting the dynamic and accumulative co-operation relationships via knowledge-based agent communications.

The next chapter will present in more details the architecture that realizes this agent framework.

CHAPTER 4

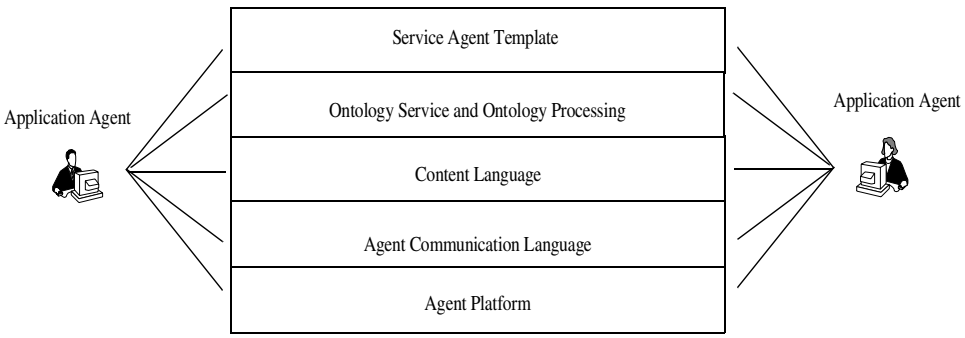
The Architecture

4.1. Introduction

The agent framework presented in the last chapter for telecommunications applications has to be realized via a functional architecture. Such an *architecture* defines a set of functional components, sub-systems or modules and how these components interact or are interconnected in order to fulfil the goals of the knowledge-based and dynamic agent interoperability.

Corresponding to the layered structuring of the conceptual agent framework, the architecture will be also divided into the functional layers as depicted in figure 33, where the corresponding agent-oriented technologies are implemented.

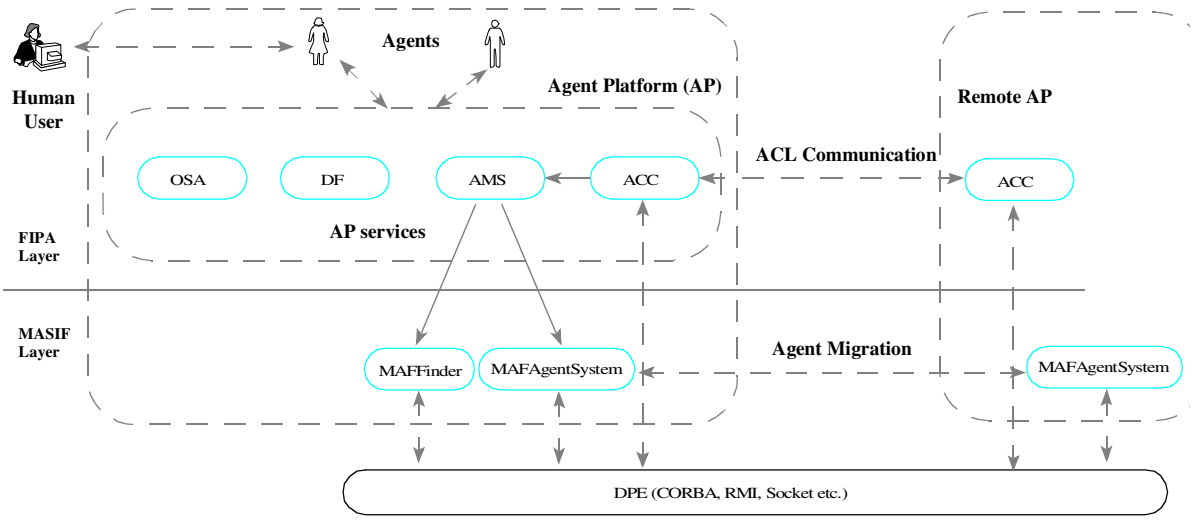
Figure 33: The Layered Architecture for Knowledge-based Agent Interoperability



4.2. The Agent Platform

The agent platform (AP) for our agent-based solution for telecommunications applications has the architecture depicted in figure 34.

Figure 34: The Agent Platform



The AP ([113]) is basically implemented by integrating two layers,

- the *FIPA Layer* which supports the AP services for agent communications and migrations, while
- the *MASIF layer* supports the MASIF conform agent migrations among different APs.

Both layers will have to utilize the facilities of a Distributed Processing Environment (DPE), which supports the communications among distributed applications. The AP should be able to use different transport protocols from the DPE. Some major protocols in this context are CORBA/IIOP, Java/RMI or plain socket communications (TCP/IP).

Usually the DPE will be integrated into a MASIF compliant MA platform (e.g. [50]). In that case the AP can be considered as being built upon a MA platform.

Based on the FIPA framework, the FIPA layer of the AP will accommodate the following platform components and their AP services.

- Agents

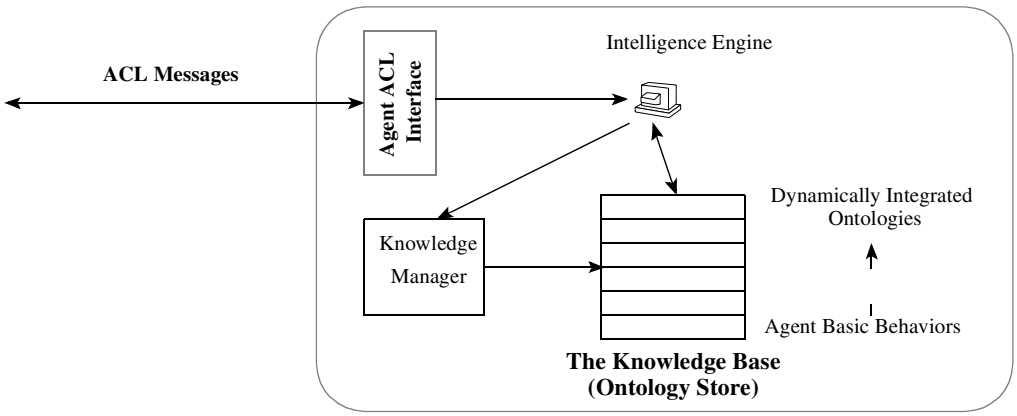
An agent is the fundamental actor in the AP which offers certain services to its environment.

In our AP, we don't distinguish between some special purpose agents, like human user interface agents or wrapper agents for legacy applications and other application agents. Only one class of agent is envisaged, which can be refined to implement agent-based services for any application contexts.

Each agent has two identifiers. One is the agent name following the FIPA GUID (*name@URL*, where URL is the URL address for the ACC, see [23]) format, which will be usually used in ACL messages and to identify the home AP (i.e. the associated ACC) of the agent. The other is the physical address of the agent (or its representative) which will be used in message transportation. Separation of agent's logical name and physical address is necessary in case of MAs, whose unique agent names can not be easily updated following the migrations of the MAs. Different from the agent name, which is assigned during creation of the agent and stays unchanged for whole life of the agent, the agent's physical address can change, e.g. after agent (MA) migrations.

The key feature of the agents in our agent-based telecommunications solution is the support for knowledge-based and dynamic interoperability based on knowledge and ontologies. An agent in this context has the structure in figure 35.

Figure 35: The Agents



The *ACL Interface* in this context is responsible for sending/receiving messages and for translation of the message between the agent internal (e.g. Hashtables or Array/Vector) and external format (e.g. XML/RDF or SL).

An agent maintains a set of ontologies that build up the knowledge for defining the dynamically extensible behaviors of the agent. Some of these ontologies can be pre-implemented/coded in the agent and define the basic behaviors of the agent, while the others will be dynamically uploaded from the incoming ACL messages.

The *Ontology Manager* is responsible for managing this dynamic store of ontologies (called *Ontology Store*) by tracking the history of modifications and by supporting the integration or

deletion of ontologies.

The *Intelligence Engine* (IE), which can be an inference engine in the AI terminology, works on the ontological knowledge of the agent to make autonomous decisions on its operations and on the interactions with the environment. Among others, the IE will also manage the different threads of dialogs by tracking the dialog contexts.

The IE will send directives to the Ontology Manager for manipulating the Ontology Store.

– Agent Communication Channel (ACC)

ACC is a platform component that serves as the contact point within the AP for sending and receiving speech act messages. Any other agent that wants to send a message to another agent should first deliver the message to the ACC by calling a method from the ACC interface. The ACC then routes the message, possibly by co-operating with intermediate ACCs in remote platforms, to the ACC of the AP in which the receiving agent resides. This ACC then delivers the message to the receiver by calling a method from this agent.

To support the routing decisions, the ACC maintains a table of physical addresses of the remote ACCs that are known by this ACC. For finding out the physical address of the message destinations within the same AP, the ACC will contact the AMS.

As specified in the FIPA standard, beside message routing and transport, the ACC is also responsible for security, and for asynchronous messaging and forwarding. As in most case such facilities are already provided by a MA platform and its DPE (e.g. [50], [68]), the ACC can simply reuse (or adapt) such facilities in realizing its own services.

– Agent Management System (AMS)

The AMS manages the physical lifecycle of all the agents residing in the AP. The responsibilities of an AMS include the creation, suspending/resuming, deletion and the migration of agents, and security management decisions like whether an agent is allowed to dynamically register at the platform (for example, this could be based upon agent's or AP's ownership), or allowed to operate on certain local resources.

For this purpose the AMS supports a dedicated agent management ontology and implements the associated functions on top of the MASIF *MAFAgentSystem* interface.

Another key function of AMS is the *White Page* service for all the agents in the local AP. The AMS maintains an index of all the agents which are currently resident on the platform. The index includes an agent's name and their associated physical transport address for the AP. Both the privileged agents and the ACC can query the AMS to find out the technology dependent transport addresses of the destination agents during agent communications.

The White Page service will be implemented on top of the MASIF *MAFAgentFinder* inter-

face.

– Directory Facilitator (DF)

Same as in FIPA, the DF is an agent which provides *Yellow Page* services to other agents.

The DF is a trusted, benign custodian of an agent directory. Agents may register their services with the DF or query the DF to find out what services are offered by which agents. The DF can restrict access to information in its directory, and will verify all access permissions for agents which attempt to inform it of agent state changes.

The hierarchy or federation of DFs will be initiated and configured by the administrator.

– Ontology Agent (OA)

The OA in our AP supports the similar functionalities as FIPA OA but is based on a different API and a different knowledge model. Unlike the FIPA Ontology Service, which supports a predicate-logic and OKBC-alike knowledge representation in order to enable the OKBC-based ontology service operations for manipulating knowledge within individual ontologies, the OA in our model supports only the management of a hierarchy of ontologies like register a new ontology, replace an ontology, search for an ontology, and not the internal definitions of these ontology. Manipulation of the internal definitions of individual ontologies, which is dependent on the specific knowledge representation paradigm, will be done within the application agents.

In this way, we can achieve a generic ontology service that can support any knowledge representation models. E.g. the ontologies can be represented as the Java classes for a MA definition, or a component Java package for building up a full solution for a specific problem.

The application agents can utilize the ontology service in dynamic adaptation and extension of their services, especially via downloading and integrating ontologies from OA.

4.3. The Agent Communication Language

Following the tradition of KQML and FIPA ACL, an ACL message within our agent architecture of this thesis will be represented as String of the format

```
(performative
    :sender:        ...
    :receiver:      ...
    :content:       ...
    :in-reply-to    ...
    :reply-with     ...
    :ontology       ...
    :language       ...
    :protocol       ...
    :conversation-id...
)
```

where the *:protocol* parameter identifies the co-operation protocol deployed for the current co-operation session.

Generally speaking, the key issue in defining an ACL is the selection of the set of performatives.

In most applications, the border between the ACL and the content can not be clearly defined. For example, we can use only one performative - *inform*, to implement the ACL layer of agent communication and put any refinements of the speech act semantics in the content layer of the messages. This is an approach which is especially popular among agent-related research projects that are not focusing on agent technology (e.g. [79]). Or we can, at the other extreme, use no content layer in agent communication and put everything in the ACL layer (i.e. the performative, subject and objects).

A simple ACL with fewer performatives will usually result in more complicated content layer and CL. An ACL with more extensive set of performatives will typically result in a complicated ACL definition and implementation, but also result in a simple CL.

The key problem with a big ACL like KQML is the complexity for the users and developers of the agent-based applications. Such complexity frequently prevent the normal users/developer to have an in-depth knowledge about the language, and also result in heterogeneous deployment of the performatives. Interoperability, which is the utmost objective of an ACL, can not be guaranteed in this context.

Based on this consideration, the strategy adopted by this thesis is to select a basic set of performatives that support the elementary service negotiation and interactions within the telecommunications environment.

As will be further clarified in the following chapters, this basic set only serves as the basis for developing agent-based co-operations in telecommunications management. It can be extended with new performatives (e.g. counter-propose, commit) to meet the requirements of specific application environments and to further optimize the co-operations in such environments. Moreover, there is also the possibility of dynamically extending this set via knowledge and ontology-based agent communications.

The basic set of performatives is selected to support the basic protocols like the request protocol or contract net (simple or iterated) as defined by FIPA [24]. This set is listed in the following table. The corresponding meanings of the speech act in the agent co-operations were presented in CHAPTER 2.

Performative	FIPA Protocols Supported
accept-proposal	Contract Net, Iterated Contract Net
agree	Request
cup	Contract Net, Iterated Contract Net
cancel	Contract Net
failure	Contract Net, Iterated Contract Net, Request
inform	Contract Net, Iterated Contract Net, Request
not-understood	Contract Net, Iterated Contract Net, Request
propose	Contract Net, Iterated Contract Net
refuse	Request
reject-proposal	Contract Net, Iterated Contract Net
request	Request

Compared to FIPA ACL and KQML, we have deliberately selected only a sub-set and have avoided some performatives that have caused controversies in the literatures, especially those that have difficulties in providing clear and simple semantics.

One example in this context is the query-re/query-if performatives, which either queries the truth of a predicate, or the object that enables the truth of the predicate. Different from the numerous performatives in KQML related to querying, these two performatives were assigned a relatively clear semantics. However, FIPA ACL based on these two performatives still have problems in the following aspects:

– The two performatives are not sufficient in many application areas (e.g. query for multiple values is not supported),

– The semantics and usage of the performatives prove to be un-intuitive in many applications, and can easily lead to ambiguity in the interpretations.

As a result, many applications, including some FIPA specifications (e.g. [23], [29]), decide to avoid the query performatives. As queries can be regarded as a special actions, they can be integrated into the content of the ACL request messages. This thesis proposes the same approach as adopted by these specifications and applications.

4.4. The Content Language for Agent Communication

An Agent Communication *Content Language* (CL) is used to represent and transmit domain-specific knowledge within the illocutionary speech act contexts of agent communication messages.

CL, and the ontologies based on the CL play the key role in enabling the knowledge-based, dynamically adaptable co-operations among the application agents. As a pre-condition for this kind of dynamic interoperations, a CL should possess the capability of representing and transmitting information about behavior knowledge associated to the data exchanged between agents. With this capability an agent can be in the position to inform or teach other agents about the evolution and the changes in its requirements, views and in its co-operation behaviors, and can be in the position to adapt the agent itself to the changed environment. This requirement suggests at least some features of knowledge representation and programming capabilities in the CL definition.

Basically, knowledge can be encoded in any popular languages or a subset of each of such languages. As an example, some people proposed to use programming languages like Java to code the knowledge in the agent communication content. In fact, with its platform independence and strong association to the Internet and WWW-based applications, Java has been considered as one of the most important language platforms for developing agent platforms and agent applications.

One scenario in this context is to transfer Java source code as message context between the agents (e.g. [33]). Another scenario is to transmit Java Class files or serialized Java objects (sequence or array of bytes) between autonomous agents or agent systems to support dynamic and adaptive co-operations among distributed software systems.

There are also efforts to use Java classes to represent knowledge and rules in agent communication contents ([63], [110], [118]) following some very simple knowledge representation principles. For example, a Java class with two methods:

- one boolean method for testing the situation in the network,
- another method for carrying out actions to deal with that situation,

can be regarded as a rule for guiding the operations of the agents. Agents can exchange such rule class files and apply such rules in reacting to their environment.

Although such programming languages can be used for representing the agent communication knowledge, they don't fulfill the requirements of a good knowledge representation language. To ease the representation and deployment of knowledge, a good knowledge representation language should be in the position to reflect succinctly and intuitively the

expressions or logics in natural languages, while maintaining at the same time the formal nature of representation in order to support machine interpretation.

Most knowledge representation models in the history of AI are based on first-order predicate logic. In another word, most knowledge representation languages are either defined in predicate logic, or can be easily mapped to a first-order logic and use first oder logic as their semantic basis.

Combined with the modal logical operators (like temporal or mental modal operators), predicate logics are also frequently used for characterizing the mental attitudes, personality and behaviors of the agents. The most important agent communication CLs in this case are based on the Knowledge Interchange Format (KIF - [42]). As mentioned in CHAPTER 2, KIF adopts a very simple, Lisp-alike syntax that eases the efficient coding/decoding and reliable transportation of messages over the global network. However, the generic feature and expressiveness of KIF also make the language difficult and inefficient to be implemented. In most cases only a subset corresponding to the Horn Clauses of the Prolog ([56]) language are supported by the implementations.

The Prolog programming language, which is based on first oder predicate logic and rules while restricting its rule-based knowledge representation to Horn Clauses, can be considered as a compromise between the expressiveness of first-order logic and the efficiency of a conventional programming language. In fact it is still the most popular language platform for implementing the variety of AI knowledge representation models.

4.4.1. Representing the Knowledge about Telecommunications Resources

As mentioned above, knowledge representation has been the core of studies in AI and knowledge-based systems. Numerous frameworks for representing knowledge were developed and tested in this context, each with its advantages or limitations for the different applications purposes or areas.

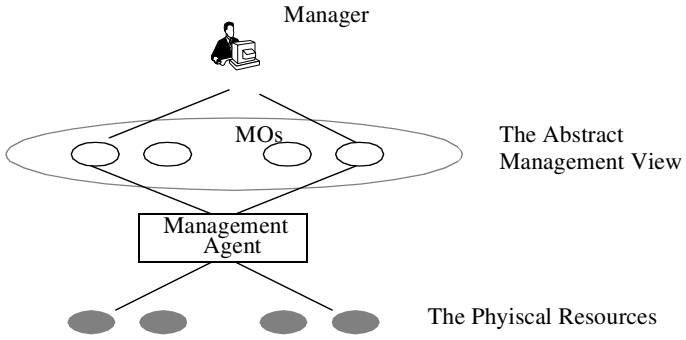
Different from these mainly AI-oriented efforts, the study of this thesis will focus on defining an agent communication content language for modeling and representing knowledge related to the management of telecommunications resources. As a result, our knowledge representation framework will:

- be based on an existing knowledge representation framework that is more intuitive and well proven in the context of modeling telecommunications resources,
- sufficient to deal with the complexity and variety of the knowledge required in managing telecommunications resources.

To meet the first requirement, we will first take a look at the tradition of telecommunications management as represented by the activities within standardization bodies. As mentioned before, telecommunication management applications are dominated either by the ITU-T TMN [53] or the IETF SNMP [49] frameworks. Especially the TMN framework, with its support for Object-Oriented development, is quickly gaining momentum in this area.

The core of these management frameworks is the abstract view of telecommunication resources as Managed Objects (MOs) as depicted in figure 36. Agents (called management agents hereafter to avoid confusions with intelligent agents) are responsible for manipulating the physical resources to offer this view of MOs. The managers, which are the intelligent applications that manage the telecommunications services, operate upon these abstract objects to achieve their management goals.

Figure 36: Modeling Telecommunications Resources as MOs



In a DOT-based telecommunications management environment, manager applications typically play both the role of management agents and manager in their co-operations with other management applications. The knowledge used for co-operations is therefore based on the abstract management views of the resources.

The management view in these frameworks is based on object classes that characterize the resource categories and the object instances that represent the physical resource entities. E.g. within the TMN/GDMO paradigm, each object instance of an object class will have some associated capabilities and attributes. The knowledge about the resources is therefore characterized via the knowledge about such capabilities and attributes and there inter-relationships, constraints or usage.

At the moment, the TMN or the SNMP framework support only the formal definition of the syntactical aspects of the MOs in the style of a RPC paradigm. The knowledge about the usage or meanings of these resources are specified in plain English texts. To support the knowledge-based dynamic interoperability among telecommunications management agents, extensions in this context will be needed.

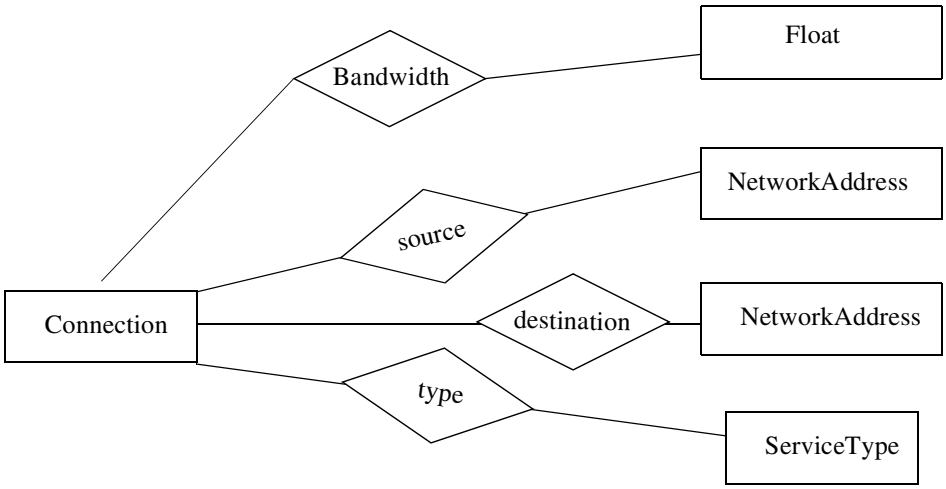
Corresponding to these traditional frameworks for representing management knowledge in

open systems, the RDF framework as mentioned in section 3.4 offers a similar solution based on XML for representing resources and the associated knowledge in an Internet and Web-oriented environment.

Originally designed for Web document management, RDF considers the Web resources as resource objects with properties and interrelationships (for a more detailed description of RDF please refer to [103] and [104]). A *RDF document* in this context consists of a list of *resource descriptions* that describe a number of resource instances and their properties. The corresponding resources classes (and the associated property types) are defined in a *RDF schema* document, which is itself a special RDF document.

Basically the RDF framework is based on the entity-relationship theory, where the RDF schema corresponds to an entity-relationship model and a RDF document defines an instantiation of the model in terms of resource instances.

Figure 37: An Example of Entity-Relationship Model in Telecommunications Resource Management



As an example to illustrate the RDF-based knowledge model, the entity-relationship model (where the entity types *Source*, *Destination* and *ServiceType* are simple strings) in figure 37 can be defined by the following RDF schema document.

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:aux="http://mimosa.fokus.gmd.de/projects/example/auxiliary#">

<rdfs:Class rdf:ID="Connection">
  <rdfs:comment>
  This class describes a generic network connection between two network addresses for
the purpose of data communication
  </rdfs:comment>
```

```
</rdfs:Class>
<rdfs:Class rdf:ID="ServiceType">
  <rdfs:comment>
   A String that describes the transport service type of the connection, e.g. SDH or
ATM.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "rdf:Literal”>
</rdfs:Class>
<rdfs:Class rdf:ID="NetworkAddress">
  <rdfs:comment>
    A String that identifies the network address that terminates the connection.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "rdf:Literal”>
</rdfs:Class>
<rdfs:Property rdf:ID="bandwidth">
  <rdfs:domain rdf:resource="#Connection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:Property rdf:ID="source">
  <rdfs:domain rdf:resource="#Connection"/>
  <rdfs:range rdf:resource="#NetworkAddress"/>
</rdfs:Property>
<rdfs:Property rdf:ID="destination">
  <rdfs:domain rdf:resource="#Connection"/>
  <rdfs:range rdf:resource=" #NetworkAddress"/>
</rdfs:Property>
<rdfs:Property rdf:ID="type">
  <rdfs:domain rdf:resource="#Connection"/>
  <rdfs:range rdf:resource="#ServiceType"/>
</rdfs:Property>
</rdf:RDF>
```

which says that a *Connection* resource object can have four properties that identify respectively the *bandwidth*, the *source/destination* and the *type* of the connection.

A possible instantiation of the model can be described by the *resource description* in the following RDF document:

```
<?xml version="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#">
<con:Connection rdf:ID="100-10-1">
  <con:type> atm-vp </con:type>
  <con:source> 193.175.136.22 </con:source>
  <con:destination> 193.175.136.36 </con:destination>
</con:Connection>
</rdf>
```

where the name space identifier *con* refers to the RDF schema defined above.

The resource and attribute/property-based simple knowledge model for RDF framework makes it easily adaptable to different application environments. In the context of telecommu-

nications management, all the capabilities associated to the resources can be modeled as properties of dedicated resource classes. E.g, an *action* supported by a managed resource object can be regarded as a property of the class *Action*, with action parameters as the properties of the Action resource.

Moreover, the Object-Oriented development style of RDF framework further enhances the extensibility and reusability of the knowledge represented, which have significant impact on the applicability of the framework in the open and dynamic telecommunications environment.

On the other hand, the simplicity of RDF based on basic entity-relationship model also means some limitation in the expressiveness of the framework. As the basic entity-relationship model corresponds to the propositional logic, the RDF framework does not allow the deployment of variables and has therefore limited expressiveness compared to first order logic.

Among others, this restriction makes it difficult to express general relationships among the object properties. As an example, suppose we want to define a new resource class, called *RemoteConnection*, which is a subclass of *Connection* but requires the source and destination addresses to be in different subnetworks (in case of IP address, the first three numbers should be different), extensions to the basic RDF language will be needed.

To obtain the expressiveness needed in managing the heterogeneous and complex telecommunications services, we will extend the basic RDF framework with the capability of first order predicate logic by allowing variables in the knowledge representation and by adopting knowledge rules based on Horn-Clauses. As an example, the characteristics of the RemoteConnection can be expressed by the following constraint rule:

– for every *RemoteConnection* ?A with *source* ?X and *destination* ?Y, ?X and ?Y should be in different subnetworks.

4.4.2. The Content Language Based on RDF

To come up with the requirements in the telecommunications management environment, the RDF-based CL framework in this thesis includes the following basic knowledge elements:

– Managed Resources

Every managed object in this application environment is considered as belonging to the RDF class *Resource* ([104]) or its subclasses. The concept of *Resource* class corresponds to the top class in many programming language, e.g. the *Object* class in Java, and can be extended to describe any entities classes in the selected problem domain. Each Resource instance will

have a group of properties and also an attributes (called ID) that uniquely identifies (together with the document identifier) the object. This ID attribute corresponds to the Relatively Distinguished Name (RDN) in the TMN/SNMP frameworks.

In this thesis we will not further restrict the format of this ID. In a specific application environment, one can, e.g. adopt the TMN naming scheme to ease the integration with TMN-based applications.

– Variables

The variables in the knowledge representation, written in the form ?*AnyString*, serve as place holders for resource instances. In this RDF-based framework, each variable is universally quantified within a resource description. E.g. the following resource description

```
<con:Connection rdf:ID="100-10-1">
  <con:type> atm-vp </con:type>
  <con:source> ?Source </con:source>
  <con:destination> ?Destination </con:destination>
</con:Connection>
```

corresponds to the expression

∀(?Source) ∀(?Destination).Connection(“100-10-1”, atm-vp, ?Source, ?Destination)

in the first order logic.

– Actions

An *Action* is regarded as a special *Resource* that describes an action to be carried out by an agent. Basically an Action possesses (beside its ID) two properties:

- the *agentID* identifies the agent that realizes the action,
- the *actionType* provides additional information (e.g. category) for the service action,
- the *service* property refers to a service description that characterizes the action to be carried out.

– Functions and Predicates

Functions and functional expressions (only pre-defined functions like +, -, equals etc. are allowed) are a special category of resources that should be evaluated to resource objects before participating in the knowledge-based reasoning.

Any other resources are considered as predicates. In fact a resource description is regarded as the logical expression about the existence of the corresponding resources.

– Rules

Similar to the constraint programming paradigms (as presented in the numerous literatures like [46], [86]), rules are envisaged as additional information that further characterizes the logical relationships among the resources and/or properties. A rule in this context can also be regarded as a specific resource with a number of other resource instances as its properties. These properties must additionally satisfy some logical constraints that are implied by the semantics of the rule. In our framework, two categories of rules are envisaged:

- *Implies*

with resource description of the form

<Implies>
  <body>
    <body_li> $body_1$ </body_li>
      ...
    <body_li> $body_n$ </body_li>
  </body>
  <head> head-expression </head>
</Implies>

where *Implies* represents logical implication, and the head is implied by the conjunction of the sequence of body expressions.

As a special case an *Implies* rule without the bodies simply means the truth of the head expression, i.e.

<Implies>
  <head> head-expression </head>
</Implies>

is equivalent to *head-expression*

– *Equivalent*

with resource description of the form

<Equivalent>

<head> head-expression </head>
<body>
<body_li> $body_1$ </body_li>
...
<body_li> $body_n$ </body_li>
</body>
</Equivilent>

where *equivalent* represents logical equivalence, i.e. if and only if.

In summary, our RDF-based CL for knowledge representation can be defined by the following RDF schema:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">

<rdfs:Class rdf:ID="Variable">
  <rdfs:comment>
    A String that starts with a ?.
  </rdfs:comment>
<rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Literal">
</rdfs:Class>
<rdfs:Class rdf:ID="Function">
  <rdfs:comment>
   This class describes an function whose implementation is outside the RDF model
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Resource">
</rdfs:Class>
<rdfs:Class rdf:ID="Action">
  <rdfs:comment>
  This class describes an action to be carried out by an agent
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Resource">
</rdfs:Class>
<rdfs:Class rdf:ID="Rule">
  <rdfs:comment>
  A definition of a logic rule that characterizes some restrictions on the resources
and properties.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Resource">
</rdfs:Class>
<rdfs:Class rdf:ID="Implies">
  <rdfs:comment>
    An Implies rule.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "#Rule">
```

```
      </rdfs:Class>
      <rdfs:Class rdf:ID="Equivalent">
        <rdfs:comment>
          An Equivalent rule.
        </rdfs:comment>
        <rdfs:subClassOf  rdf:resource = "#Rule">
      </rdfs:Class>
      <rdfs:Class rdf:ID="Body_li">
        <rdfs:comment>
          This class describes the list of bodies in a rule.
        </rdfs:comment>
        <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Seq"/>
      </rdfs:Class>
      <rdfs:Property rdf:ID="agentID">
        <rdfs:domain rdf:resource="#Action"/>
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#:Lit-
      eral"/>
      </rdfs:Property>
      <rdfs:Property rdf:ID="actionType">
        <rdfs:domain rdf:resource="#Action"/>
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#:Lit-
      eral"/>
      </rdfs:Property>
      <rdfs:Property rdf:ID="service">
        <rdfs:domain rdf:resource="#Action"/>
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
      19990303#:Resource"/>
      </rdfs:Property>
      <rdfs:Property rdf:ID="head">
        <rdfs:domain rdf:resource="#Rule"/>
        <rdfs:range rdf:resource="rdfs:Resource"/>
      </rdfs:Property>
      <rdfs:Property rdf:ID="body">
        <rdfs:domain rdf:resource="#Rule"/>
        <rdfs:range rdf:resource="#Body_li"/>
      </rdfs:Property>
      <rdfs:ContainerMembershipProperty rdf:ID="body_li">
        <rdfs:domain rdf:resource="#Body_li"/>
        <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Resource"/>
      </rdfs:ContainerMembershipProperty>
      </rdf:RDF>
```

Because the syntax and meanings of the variables must be defined and validated outside the scope of RDF standard, and also because the semantics of rules are implemented outside the basic RDF framework, the resulted CL is considered as a real extension of the current RDF framework in the context of telecommunications resource modeling.

4.4.3. An Example

Generally speaking, the management relevant resources (their existence and properties) are modeled in the CL as resources of the appropriate classes. These classes are defined in the associated RDF schema. Rules are deployed in the schema and the RDF documents to further characterize the generic logical relationships or constraints associated to these resources.

Referring to the example about RemoteConnnection in 4.4.1, we can define the following RDF schema:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#"
          xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">
<rdfs:Class rdf:ID="RemoteConnection">
  <rdfs:comment>
  This class describes a remote network connection between two subnetworks  </rdfs:com-
ment>
  <rdfs:subClassOf  rdf:resource = "con:Connection">
</rdfs:Class>
<schema:Implies rdf:ID="remote">
  <schema:head>
     <con:Connection>
        <con:source> ?X </con:source>
        <con:destination> ?Y </con:destination>
     </con:Connection>
  </schema:head>
  <schema:body>
    <schema:body_li>
      <pd:DifferentSubnetworks>
          <aAddress> ?X </aAddress>
          <zAddress> ?Y </zAddress>
      </pd:DifferentSubnetworks>
    </schema:body_li>
  </schema:body>
</schema:Implies>
</rdf:RDF>
```

This schema specifies a resource class *RemoteConnection*, which is a subclass of the *Connection* with a extra constraint specified by the rule. Basically, the rules says if a *RemoteConnection* has two terminations at *?X* and *?Y*, then the two values should satisfy the *pre-defined* boolean function *DifferentSubnetworks*. The *DifferentSubnetworks* checks if two network addresses lie in different subnetworks.

4.4.4. Implementation of the Agent Communication Content Language

It is expected that for different applications purposes and emphases, a subset of the CL presented in this section will be sufficient. In fact, to guarantee the efficiency of the applications and to deploy lightweight agents, many implementations will choose to support a subset of the presented language elements. Basically the following three subsets/profiles of CL can be envisaged in the telecommunications management applications.

- CL0 - Without Variables and Rules

CL0 can be considered as equivalent to the standard RDF framework. It is in the position to describe resources and their properties. Although it is not sufficient to support the knowledge-based and dynamically adaptive interoperability among agents, it can significantly simplify the implementation in a relatively static co-operation environment. E.g. an AI-based inference mechanism is generally not required for agents which support only CL0.

- CL1 - Without Rules

Variables can be used as place holders in the message content to indicate the necessity of providing some properties whose values can be dynamically filled in during agent co-operations. As an example, to call for proposal on some action from an agent, the initiator can send the following message

```
(cfp
  :sender  initiator
  :receiver responder
  :content
    " <?xml version="1.0"?>
      <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge-schema#"
          xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#"
          xmlns:vpn="http://mimosa.fokus.gmd.de/projects/example/vpn#">
          <vpn:reserveConnection>
            <schema:agentID> responder </schema:agentID>
            <schema:service>
                 <con:Connection>
                        <con:type> ?X </con:type>
                        <con:source> 193.175.136.22 </con:source>
                        <con:destination> 193.175.137.22 </con:destination>
                 </con:Connection>
            </schema:service>
          </vpn:reserveConnection>
      </rdf> "
...)
```

In this case the initiator agent asks the responder to make a proposal for reserving a connec-

tion between two termination points. It is open to the responder to select the type of the connection (ISDN, ATM or SDH), but it has to indicate this type in its response. E.g. the responder can reply with the following message

```
(propose
  :receiver  initiator
  :sender responder
  :content
    " <?xml version="1.0"?>
      <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#"
          xmlns:vpn="http://mimosa.fokus.gmd.de/projects/example/vpn#">
          <vpn:reserveConnection>
            <schema:agentID> responder </schema:agentID>
            <schema:service>
                 <con:Connection>
                        <con:type> atm-vp </con:type>
                        <con:source> 193.175.136.22 </con:source>
                        <con:destination> 193.175.137.22 </con:destination>
                 </con:Connection>
            </schema:service>
          </vpn:reserveConnection>
      </rdf> "
...)
```

saying that it is ready to provider an ATM-VP connection.

- CL - With Variables and Rules

CL corresponds in fact to an Object-Oriented logic programming language. Although CL does not mandate a specific implementation, we do expect that the default implementation will be based on translating the documents into Prolog-alike knowledge items (facts and rules).

Basically, when an agent receives an ACL message, the message content will be in the format of a CL documemt. The agent will interpret/process the message under the ACL performtive context by using the *combination* of the resource decriptions and agent's local knowledge (resource instances and rules) for processing the content.

The relevant knowledge will be used as the knowledge that supports the decisions of an agent on its actions. An agent will try to apply as many as possible such knowledge items in making its decisions, in order to derive detailed information and secure decisions for creating its response messages and for carrying out actions.

4.5. The Knowledge-based Ontology Framework for Agent Communications

In a co-operation relationship based on natural intelligence, human experts usually talk to each other using shared knowledge that is based on a shared *vocabulary* in a specific field of science or business. A shared interpretation of the words or concepts in this vocabulary is the prerequisite for human experts to communicate and to be understood. Depending on the complexity and sophistication of the application fields, a vocabulary can become very complicated and it can require years of educations (e.g. in the schools or universities) for someone to understand it.

Such a vocabulary together with it associated interpretation/meanings is frequently referred to as an *ontology* in the language philosophy ([82]) and AI ([76]).

Agents communicate with each other to solve problems or to achieve certain goals in an application domain. As each agent with its intelligence is specialized in a specific field and developed within a specific background, each agent can have its own application specific vocabulary, and its own interpretation of this vocabulary in generating and interpreting agent communication messages. The key issue here is how to ensure that agents can understand each other during their co-operations by correctly using and interpreting such vocabularies.

In analogy to the human communications, the prerequisite in this context is the shared knowledge, i.e. the interpretation of meanings of the vocabulary used in a specific communication/ conversation session. Such a vocabulary, together with its shared interpretation by the agents, assigns the shared menaings to the symbols and expressions in agent communication messages and becomes an *ontology* for the messages.

An ontology in this sense identifies the valid symbols, i.e. predicates, functions and constants in the content expression, and at the same time performs the functional mapping of these symbols to some well-understood meanings that guide the implementation algorithms. For a given domain and a specific implementation, the ontology may be dynamically loaded or implicitly/statically encoded in the implementation of the agent.

Within a global, open and dynamic environment, an ontology for agent communication has typically the following features:

– in-completeness

Due to the complexity of the open environments, an ontology can not fully specify the meanings of the vocabulary ontology in most application contexts. It is in-complete/partial not only vertically in the sense that it characterizes only an abstract view of the environment, but also horizontally in the sense that it only characterizes partially the behaviors within a specific abstract view.

A complete semantic characterization of an abstract view should enable the derivation of all properties that are logical consequences of that view upon the resources and objects under study. This is usually not possible due to the complexity of syntactical information and their semantics within the dynamic and heterogeneous agent environment. It is not necessary either because each agent is interested only in that part of ontological meanings that guides its generation and analysis of communication messages, and that guides its responses.

This *horizontal in-completeness* within an abstract view can significantly reduce the complexity of ontology definitions, because it is sufficient to offer the set of partial ontological information (e.g. rules) that will support the involved agents in the applications and co-operations.

– context-sensitivity

An ontology is usually developed within a specific community and with some specific applications in mind. The complexity of the global and versatile agent environments means extreme complexity in providing a universal characterization of the concepts used in agent communications. With the versatility of agent backgrounds, it is also generally impossible to provide a unique characterization of the meanings of the syntactical constants and expressions in agent communication messages that can be widely accepted by the agents community.

Each agent, with its own interests, emphases, experiences/backgrounds and intelligence, works and co-operates with other agents in a specific application context. For guiding its behaviors, what an agent needs is the meanings of agent communication messages in that application and co-operation context. It is also sufficient for an agent to shared the application specific interpretation, in order to present rational behaviors to its environment.

Moreover, context-sensitive paradigms for ontology development can significantly reduce the complexity in the ontology itself and also in the utilization of the ontology for guiding agent's behaviors in the applications.

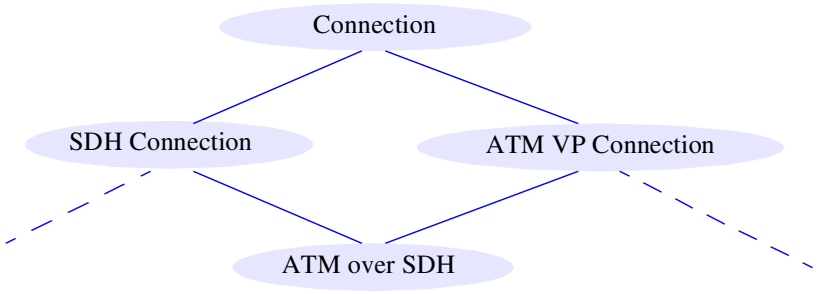
– hierarchical structuring

The dynamic nature and in-completeness of the agent-based environment also means the dynamic nature of the ontology or ontologies that are required by agent co-operations and which shall be supported by the agents. Any ontology will have its limitation and deficiencies that will emerge and become critical following the evolution of the technological innovations and the business/technical requirements in the application environment. Also due to the nature of partial characterization, an ontology has to be easily modified, extended or even replaced to support agent co-operations in the dynamic environment.

An ontology assigns meanings to new expressions in an agent communication CL by map-

ping it to some well understood concepts in the domain of meanings. This set of well understood concepts again can be built upon another ontology that is already supported by the involved agents. Similarly, for any new requirements or new application field, we can always define the new expressions and vocabulary by extending, refining or combining the existing ontologies and in this way characterize the meanings of the new concepts. As a result, the natural relationships among the ontologies are *hierarchical* as depicted in figure 38, where lower level ontologies are always built upon some higher ontologies.

Figure 38: Ontology Hierarchy for Transport Connections



Within this hierarchy, an ontology identifies in this sense a specific context in which the agent communication messages can be interpreted. Such a context can be used or refined by another ontology, which is represented by a descendant node in the hierarchy, in order to define a more detailed context and for better support in the applications.

Ontologies are used by agents for guiding the generation and interpretation of the communication messages. An agent can either statically code the ontology support into its code base, or acquire the definitions of the ontologies dynamically via run-time communications with its environment and with other agents. Although the first approach usually achieves higher performance in response time as it does not require the dynamic downloading and interpretation of ontological knowledge, it does not enable the dynamic adaptation of agent's intelligence and functions. As our agent architecture aims at supporting the dynamically adaptive and knowledge-based co-operation among distributed applications, the second way of ontology acquisition plays a more important role in our agent-based software design paradigm.

An agent can dynamically support and combine multiple ontologies in its co-operations with other agents. To acquire an ontology needed for agent communication, an agent can

- modify or extend an existing ontology via exchanging ontological knowledge with other agents, or
- download new ontology definitions directly from some ontology server.

In the second scenario, an *Ontology Agent (OA)* can be deployed to maintain a database of ontology definitions, and to support other agents in, among others,

- asserting/registering and de-registering ontologies,
- retracting/deleting ontologies,
- searching, retrieving and downloading ontologies
- modifying ontologies

These services offered by an ontology agent are called *ontology services*.

To enable the dynamic extension, downloading and integration of ontologies, the key issue in this context is to develop a suitable ontology representation and management environment. The next section will focus on the selection and definition of such a representation framework. After presenting this framework, we will discuss some guidelines for developing ontologies for telecommunications applications using the representation framework, and also present an ontology service that can support the maintenance and management of the ontologies in a distributed telecommunications environment.

4.5.1. The Knowledge-based Ontology Framework

Based on the definition, an ontology first assigns meanings to the symbols used for the message content. Together with the semantics of the CL, an ontology can be used to determine the meanings and the interpretation of any CL expressions.

Formally, an ontology Ω can be defined as

$$\Omega \in \{\wp(C) \Rightarrow \Psi\}$$

where

- C refers the universe of symbols, i.e. predicates, functions and constants that are used in the content, while $\wp(C)$ denotes the all the subsets of C,
- Ψ defines the domain of *well-understood* meanings, i.e. the shared knowledge existing among the agents,

– $\Rightarrow$ refers to a functional mapping.

As a result, an ontology involves (besides identifying the domain of well-understood meanings) mainly two steps of definitions:

– identifying the set of symbols that can be used in the message content,

– the *assignment of meanings* to a group of symbols by mapping this set of symbols to some domain of well-understood meanings.

The symbols, together with syntactical rules of the CL determine the *syntactical aspects* of the agent communication contents, while the assignment of meanings is related to the *semantics* of the contents. Assuming the pre-existence of the shared domain meanings (i.e. the range of the semantic assignment function), an agent must be provided with both categories of information in order to understand the agent communication messages.

Semantic specification is a very difficult issue in a heterogeneous, dynamic and distributed environment. The major obstacle which hampers the acceptance of semantic ontology specifications in the telecommunications applications is the complexity of the formal semantic specifications. Such a complexity makes it very difficult to produce, to validate and to deploy an extensive semantic specification of an ontology.

As a result, as mentioned in Chapter 2, in supporting the interoperability among agents there are also significant efforts, especially within activities that are not directly related to AI, in the direction of defining *syntactical ontologies* that focus only on the syntactical definition of agent communication message content. Syntactical ontologies regard ontologies as the definitions of the syntactical aspects of the content language for agent communication, i.e. the identification of the symbols or syntactical.

These ontology definitions typically assume a fixed set of syntactical rules for the communication content language and using ontologies to specify the set of symbols that can be used in the agent communication content. The ‘ontologies’ defined in some FIPA documents (e.g. FIPA - [23], [29]), or the ontologies used in some European projects (e.g. [123]) belong to this category of ontologies, where an ontology in fact lists the symbols that can be used within the message content (e.g. SL [24]).

The meanings or the interpretations of the symbols in the syntactical ontology definitions are based on informal and intuitive understandings among the developers of the agent applications and must be coded into the agents. Non-interoperable implementations are frequently resulted due to the ambiguity in the understandings.

Via exchanging syntactical ontologies, agents can succeed in dynamically generating/parsing communication messages in different syntactical forms and select (possibly with the guid-

ance of human operators) via Dynamic Invocation the appropriate, pre-existing semantic interpretations (e.g. Java classes) for determining the behaviors in response to the messages. In this way, syntactical ontologies support to some extend the dynamically adaptive co-operation behaviors between IAs. However, such a support is limited because it requires the static and local availability of the software components for processing different syntaxes.

To offer the appropriate *support for dynamic adaptability*, agent has also to be in the position to exchange information related to the meanings/behaviors beside the syntactical information. In another word, agent must be in the position to exchange ontological knowledge for defining the behaviors required for processing messages. Such semantics-oriented ontology information provides another layer of agent communication content, and is needed for

- eliminating the possibility of ambiguities in the interpretation of agent messages, and for
- guiding the agent behaviors in generating, analyzing and reacting to the messages.

The context-sensitivity and in-completeness of the agent ontologies, as identified above suggest a *knowledge-based* approach towards the ontology paradigm for agent interoperability. Knowledge in this context refers to the semantics of the agent communication content but does not directly specifies such semantics. Different from a typical framework for the direct semantic specifications in open, distributed environments, e.g. those based on Z ([77], [87]), SDL, ESTELLE, LOTOS (see [47]), a *knowledge-based ontology*

- tries to capture the human intuition, preference and experience with respect to the deployment of ontologies, and
- aims at offering only a partial and in-complete characterization of the meanings that are relevant for guiding the operations of the agents.

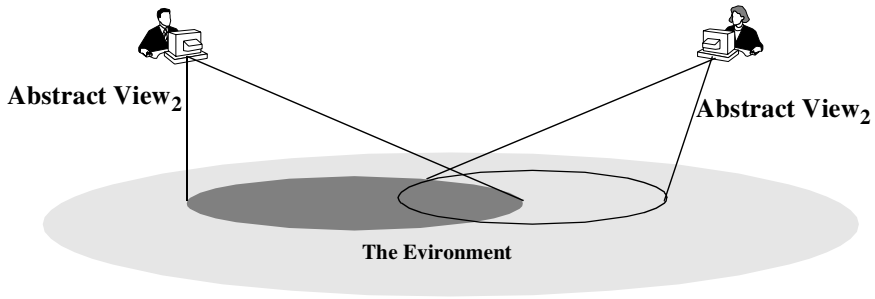
Some AI-based traditional ontology frameworks for agent interoperability, like Ontololingua ([20]) and OKBC ([5]), which aim at supporting the characterization of both syntactical and the semantic aspects of the ontologies by identifying the object classes, relations and the constraints on these classes and relations, can be considered as enabling the knowledge-based paradigms for ontology definitions and deployment. The knowledge- and RDF-based ontology framework adopted by this thesis is in fact based on this tradition, while adapting the representations for the Web-based environments and further emphasizing the support for encoding the partial and usage-oriented knowledge in the ontology definitions.

In this thesis, a *knowledge-based ontology* is defined via a *RDF document* (or a schema doc-

ument, which is considered as a special category of RDF documents). Such a RDF document specifies the classes and properties for the information entities appearing in that ontology. I.e. it specifies the symbols used in the agent communications content, and at the same time imposes some rule-based constraints on such classes and properties.

Basically, a RDF schema document defines the resource classes and properties, and implicitly the symbols, i.e. the predicates, functions and constants to be used in message contents. Rule and instance definitions in a RDF schema or RDF document then further constrains the possible abstract *views* (figure 39) of the telecommunications environment and its resources. A view in this context in fact identifies the set of resources and properties in the environment.

Figure 39: The Abstract Views



One concrete telecommunications environment can support many different views based on the different abstraction and completeness levels. Suppose Φ denotes the set of possible views that are valid in an environment, a RDF-based ontology with its resource descriptions will be satisfied (i.e. all the descriptions are valid) only by a subset of Φ, if we call this set φ then $\varphi \subseteq \Phi$. Suppose the RDF ontology, denoted as Ω_{rdf}, defines the set of symbols (i.e. class and property names) C_{rdf}, then we can regard the semantics of the ontology as the mapping from C_{rdf} to φ, i.e.

$$\Omega_{rdf} = (C_{rdf} \Rightarrow_{rdf} \varphi)$$

In another word, an ontology maps the set of constants to a set of possible views which satisfy the resource constraints specified by the RDF document. These views in fact identify the contexts in which the resources/properties can occur.

Further more, if in any environment whenever the ontology A is satisfied by the views in φ_A and the ontology B is satisfied by the views in φ_B, we always have $\varphi_A \subseteq \varphi_B$, then we say ontology A is a *specialization* of B. Within the knowledge-based ontology framework, a specialized ontology *inherits* all the knowledge from the parent ontologies, impose more constraints on its views of the environment and creates a refined picture of the resources.

Generally speaking, if for two RDF ontology definitions Ω_1 and Ω_2 we have $\Omega_1 \subseteq \Omega_2$ (i.e.

Ω_1 contains a subset of the resource descriptions in Ω_2), then Ω_2 is a specialization of Ω_1. As an example $\Omega_1 \cup \Omega_2$ (disjunction of the resource descriptions after resolving naming conflicts) is always a specialization of Ω_2.

The specialization relationship defines a hierarchical structuring of the universe of ontology definitions as previously mentioned as a requirement on the ontology framework. The root of this ontology hierarchy represents the most general ontology definition, which contains no/minimum constraints about the contexts for the resource expressions. This ontology will be called *top*. It can be in fact the empty ontology definition. The bottoms of this hierarchy is an ontology definition that contains the most strict constranits and can be satisfied by an empty set of the environments.

Between the top and the bottom, each ontology definition is the specialization of ancestor ontology definitions, and at the same time generalization of any descendant nodes. Going from the root to the bottom, we will have more and more specialized ontology definitions that give a more detailed and accurate characterization of the environments they are applied to, and which support more accurate, reliable or more efficient decisions of the agents based on such ontologies. One default behavior of any agents is to make best effort to find and apply the *most specialized* ontology definition for guiding and optimizing its operations.

Sub-hierarchies of ontology definitions from the universe about the managed resources are the ontological knowledge-bases to be deployed in an agent-based telecommunications management environment. The following subsection will further elaborate on the methodology for constructing the ontological knowledge hierarchy, and on the platform for deploying and managing such hierarchies within the open environment.

4.5.2. Developing Knowledge-based Ontologies

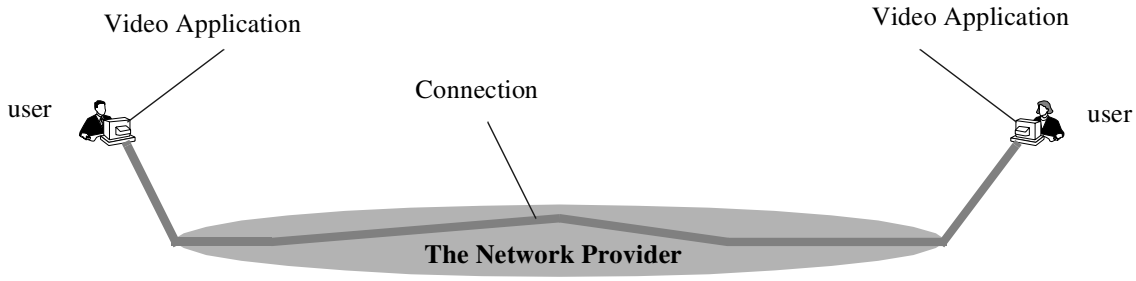
The dynamic nature of the agent environment, and the abstraction, in-completeness of ontological knowledge mean the frequent needs for extending or even modifying the existing ontologies based on new experiences and knowledge. The basic methodology in developing the ontology knowledge for agent co-operations is to accumulatively build up and extend a hierarchy of ontologies via specializations on the existing, general ontologies.

Typically, an agent will start interaction with its world based a some primitive, bootstrapping knowledge about its environment, and about the usage of its co-operation interfaces and associated messages. When time passes by, new and more detailed knowledge can be acquired. Such knowledge should be integrated into the ontologies the agent supports, resulting in the learning behavior of the agent. Alternatively, an agent can also dynamically identify new requirements on its co-operation knowledge and try to load such new ontologies across the network to extend its set of supported ontologies.

As identified in the previous subsection, extensions to an ontology is realized via adding new resource descriptions for new resources/concepts and constraints to the existing RDF definitions. The guideline in this context is to identify the application dependent contexts in which the new resources is to be used or which the new constraints apply.

To further elucidate the RDF and knowledge-based style of ontology definition we will use a simplified example scenario from the communication connectivity management (see [112] [116]) for supporting multimedia applications.

Figure 40: Communication Connectivity Management



The kernel issue within the dynamic VPN service mentioned at the beginning of this thesis is the management of global connections. In this context, a global telecommunications network based on connection-oriented technology (e.g. ATM, SDH, ISDN) is used to support some multi-media video conference sessions among some users and their video applications. For this purpose, some managing entity must first request (via some local network management facility) the required network connection or connections from the provider networks. This entity is responsible for reserving and activating the connection/connections (e.g. between certain IP addresses and port numbers) according to the requirements imposed by the user. After the activations of the connection/connections, the users can start their video applications at the pre-specified hosts and port numbers, which utilize the established connections for the video/audio stream transportations.

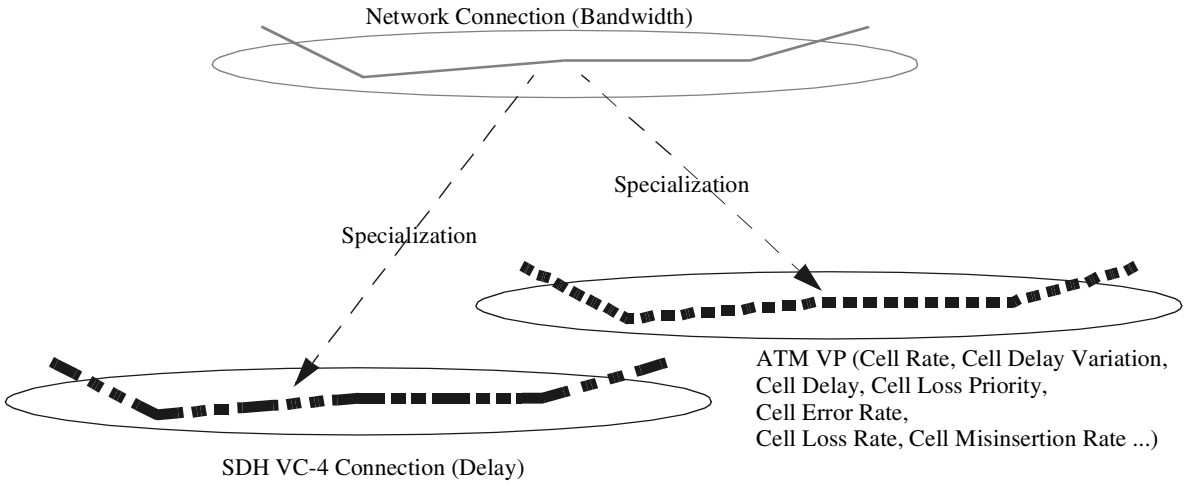
The users are generally concerned with the quality of the videos and audios they can experience via the video applications. Such qualities are characterized by some more or less subjective and high level QoS like

– *window size*, *color and picture resolutions* that are determined by the bandwidth of the connections,

– *stability* of the motion pictures that is determined by the jittering in the data transportations,

– the *audio quality* in duplex audio signals, which can be determined by the level of delays.

Figure 41: Levels of Abstractions



Users can request connections with certain QoS requirements from the network management system via a dedicated interface (e.g. *Customer Network Management* interface) or sevice management component. On the one hand such QoS requirements will support the high level QoS required by the users, while on the other hand they must be supported and understood by the network provider.

The abstract views of the connections and the associated resources can be offered by the network provider at different levels, as depicted in figure 41.

Basically, we will discuss only two level of abstractions about the resources. In the generic view we have a generic *network connection* which supports certain bandwidth. Such a connection, depending on the underlying transport technologies, can be further refined to *ATM VP connection* or *SDH-based network connection*, each supports extra capabilities in terms of extra QoS parameters.

The generic network connection can be characterized by ontology for *connection* in section 4.4.

To support the user in requesting the connection with the right bandwidth we can define a new RDF document with the following rule and call this ontology *user-view*:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
```

```
edge#"
         xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#"
         xmlns:vc="http://mimosa.fokus.gmd.de/projects/example/video-conference#"
         xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">
<schema:Implies rdf:ID="bandwidth-mapping">
  <schema:head>
     <vc:VideoConference>
       <vc:windowSize>  ?A </vc:windowSize>
       <vc:numberOfColours> ?B </vc:numberOfColours>
       <vc:resolution> ?C </vc:resolution>
       <vc:audioQuality> ?D </vc:audioQuality>
     </vc:VideoConference>
  </schema:head>
  <schema:body>
    <schema:body_li>
      <con:Connection>
        <con:bandwidth> ?X </con:bandwidth>
      </con:Connection>
    </schema:body_li>
    <schema:body_li>
      <pd:BandwidthEstimation>
        <pd:windowSize>  ?A </pd:windowSize>
        <pd:numberOfColours> ?B </pd:numberOfColours>
        <pd:resolution> ?C </pd:resolution>
        <pd:audioQuality> ?D </pd:audioQuality>
        <pd:bandwidth> ?X </pd:bandwidth>
      </pd:BandwidthEstimation>
    </schema:body_li>
  </schema:body>
</schema:Implies>
</rdf:RDF>
```

which defines the *bandwidth* parameter of a network *connection* that corresponds to a user request by relating it to the high level requirements of the video application.

Now by using more detailed knowledge about the technology deployed, we can extend the ontology by specific information that is related to either ATM VP or SDH VC-4 connections. E.g. for ATM VP we have the following ontology definition (called *atm-vp-connection*)

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
         xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
         xmlns:aux="http://mimosa.fokus.gmd.de/projects/example/auxiliary#"
         xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#">
<rdfs:Class rdf:ID="atmVPConnection">
  <rdfs:comment>
  This class describes an ATM VP network connection </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "con:Connection">
</rdfs:Class>
<rdfs:property rdf:ID="cellRate">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
```

```
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellErrorRate">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellMisinsertionRate">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellHeaderErrorRate">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellLossRate">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellDelay">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
<rdfs:property rdf:ID="cellDelayVariation">
  <rdfs:domain rdf:resource="#atmVPConnection"/>
  <rdfs:range rdf:resource="aux:Float"/>
</rdfs:Property>
 ...
<schema:Implies rdf:ID="bandwidth-mapping">
  <schema:head>
     <con:Connection>
        <con:bandwidth> ?X </con:bandwidth>
     </con:Connection>
  </schema:head>
  <schema:body>
    <schema:body_li>
      <atmVPConnection>
        <cellRate>  ?X * 53/48 </cellRate>
      </atmVPConnection>
    </schema:body_li>
  </schema:body>
</schema:Implies>
</rdf:RDF>
```

where *?X*53/48* is considered as a special abbreviated form of standard mathematical expressions which is allowed in CL. The last rule in fact determines the meanings of *cellRate* by relating it to the general *bandwidth* parameter.

This ontology extends the high level *connection* ontology by new classes properties and constraints. It offers a more detailed, low level picture of the resources upon which a more experienced and knowledge user can optimize its requests for global connectivity for the video

conference applications. The last constraint rule refines a low level technological concept, i.e. cellRate, by relating it to more high level concepts in the *connection* ontology. This kind of constraints can

- explain the meanings and usage of new concepts assuming the general agreement on the meanings of high level concepts,

- support generic adaptation and evolution of user views to more detailed resource model, e.g. in our context, by deploying the rules in the *atm-vp-connection* ontology for utilize the ATM characteristics, a *connection* request can be automatically replaced by an ATM based connection request and be applied in a context of specific technology.

Ontology definitions like *atm-vp-onnection* can be used by the sophisticated users who are accustomed to, or who want to directly access the features of ATM VP technologies. E.g. an user can operate on the following view of the resources (ontology *user-view-atm*):

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:con="http://mimosa.fokus.gmd.de/projects/example/connection#"
          xmlns:vc="http://mimosa.fokus.gmd.de/projects/example/video-conference#"
          xmlns:atm="http://mimosa.fokus.gmd.de/projects/example/atm-vp-connection#"
          xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">
<schema:Implies rdf:ID="bandwidth-mapping">
  <schema:head>
    <atm:atmVPConnection>
        <atm:delayVariation>  ?X
        <atm:delayVariation>
     </atm:atmVPConnection>
  </schema:head>
  <schema:body>
    <schema:body_li>
        <pd:stability2DelayVariation>
             <pd:stability>  ?Y </pd:stability>
             <pd:delayVariation> ?X </pd:delayVariation>
        </pd:stability2DelayVariation>
    </schema:body_li>
  </schema:body>
</schema:Implies>
</rdf:RDF>
```

where the function *stability2DelayVariation* calculates the delay variation requirement from the user video stability requirements. This rule in this ontology imposes some extra knowledge on ensuring the quality of the user service by utilizing detailed information about the

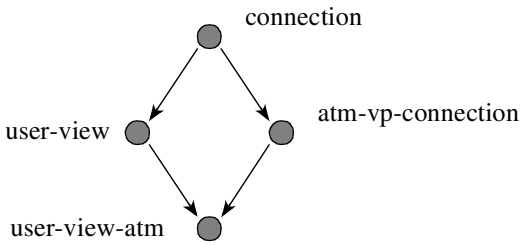
underlying technology.

user_view_atm tells/teaches the users or their assistant agents about the new service parameters by relating the contexts of such parameters to the subjective preferences of the users.

As can be seem from the definition, this ontology has to be based on the *user-view* to support the user in accessing the video conference service. Therefore *user-view-atm* must be considered as a child of *user-view* in the ontology hierarchy and inherits all the knowledge from *user-view*. With the knowledge from both ontologies, users can directly access the resources offered by ATM VPs.

The above discussion results in the ontology hierarchy depicted in figure 42.

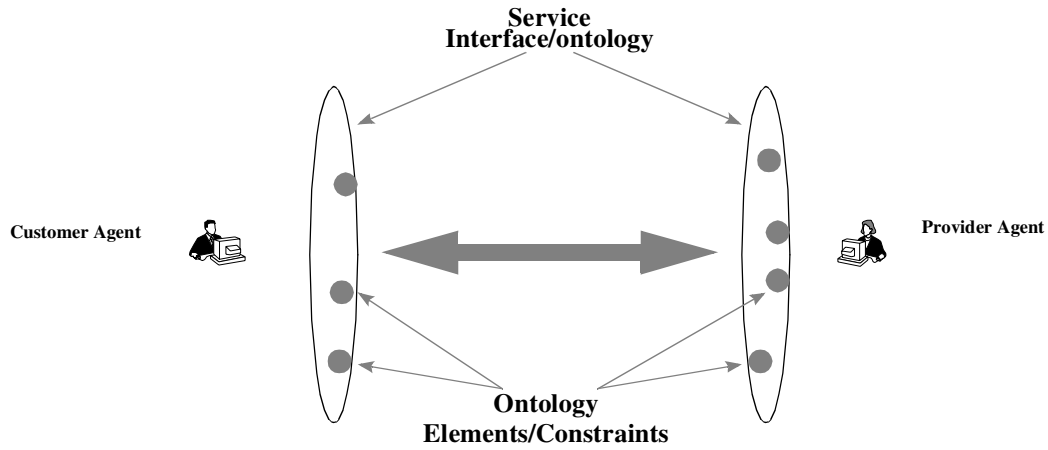
Figure 42: An Example for Developing Ontology Hierarchy



4.5.3. Dynamic Interoperability Based on Ontological Knowledge for Telecommunications Services

Knowledge-based ontologies play the most important role in supporting the flexible, robust and dynamic interoperations among the agent-based applications. Generally speaking agents in the environment can start co-operation with an initial service interface based on an initial ontology. Later, during the agent and service lifecycle, agents can exchange and absorb new ontological knowledge for improving or adapting their co-operation behaviors.

Figure 43: Service Relationship among Agents



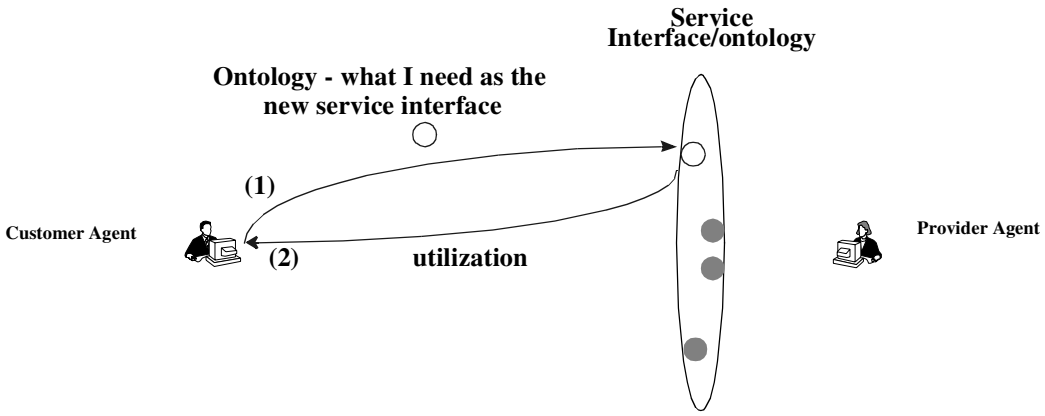
In the telecommunication application context, agent relationship can be generally regarded as service provider and customer relationship, e.g. between the customer and the network provider in the scenario discussed above, as depicted in figure 43.

The following will discuss some typical ways and scenarios (which will be refined in the following chapters in the context of agent-based telecommunications service management) in which knowledge-based ontologies can be used to support the dynamic and robust co-operations.

– dynamic service customization

Dynamic service customization can be necessary to meet the changing customer business or technological requirements, or to optimize the service provisioning and deployment procedures. Dynamic service customization can usually be realized via sending ontologies, which define the customization guides, to the service provider for configuring its service interface. With the new ontologies or ontology elements, the provider can offer new service or service features by deploying such knowledge in its customer interactions.

Figure 44: Dynamic Customization

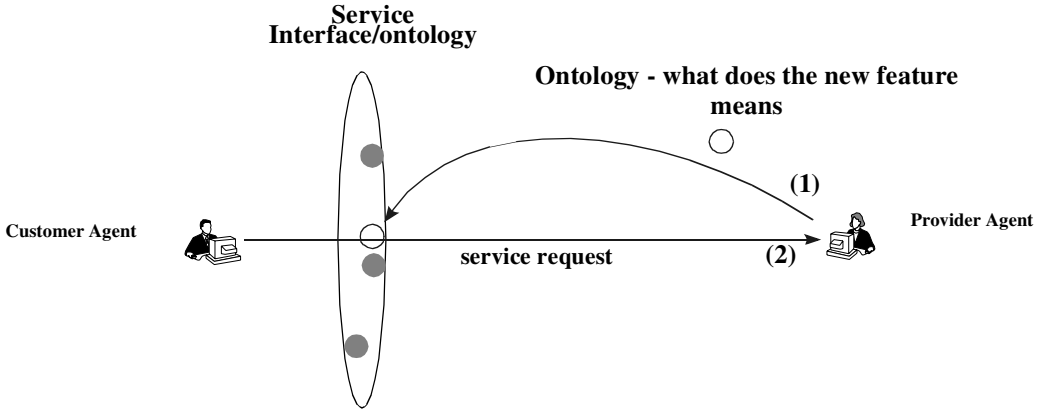


– dynamic service advertisement/publication

This kind of dynamic configuration is used to advertise the new service technologies or features of a service provider with the knowledge and behavior-oriented ontology information. Such advanced advertisement can support the dynamic deployment of such service technologies and features without re-programming the user applications.

The publication of new service technology and feature is usually done by relating the ontology for new features to existing (high level) ontologies understood by the customer agents.

Figure 45: Dynamic Publication



– robust co-operation via accumulative exception processing

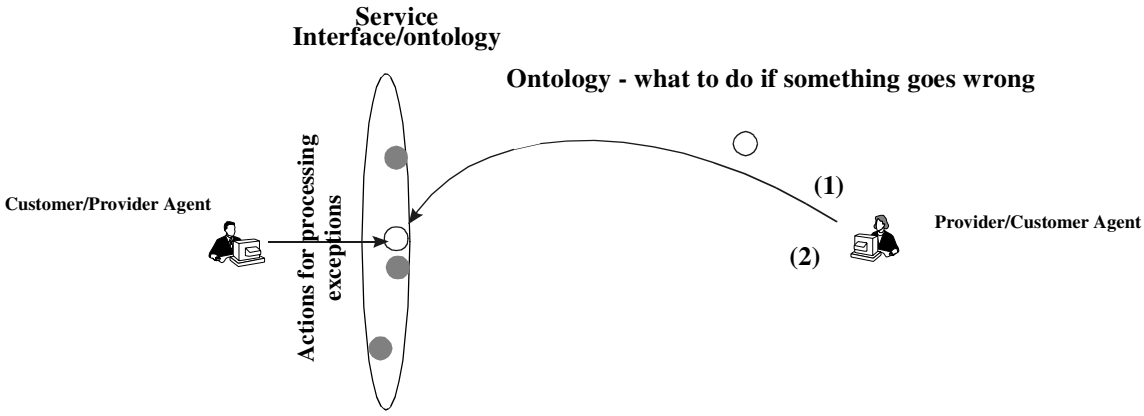
Robustness in this case means the ability to cope with unexpected situations during the agent

interactions. Theoretically, an agent can process an exceptional situation only if it knows how to process it. This is a paradox in the traditional distributed software design frameworks. There,

– a software can process an exception only if it is programmed to do so, and

– if it is programmed to do so, the exception is no longer an exception in the real sense.

Figure 46: Ontologies for Exception Processing



Within the IMA framework, agents can exchange ontologies with knowledge related to the exceptional situations to dynamically and intelligently enable the processing of new abnormal situations (figure 46).

4.5.4. The Ontology Service for Agent Interoperability

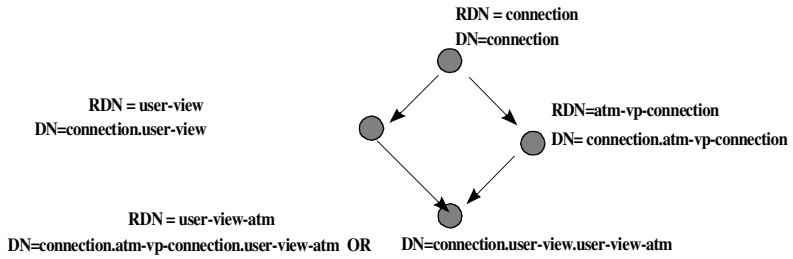
The ontologies needed in the applications environment can be managed by some specialized agents which are called Ontology Agents (OAs) and which offer the *ontology service* to the other agents. This separation of responsibility can free the application agents from the burden of maintaining a large store of ontologies that could be eventually needed for the dynamically evolving application environment.

4.5.4.1. The Ontology Hierarchy and Naming

Following the previously defined concept of *ontology hierarchy*, ontology definitions (i.e. the RDF documents) will be organized into a hierarchical structure with the general ontologies at higher levels and specialized ontologies at lower levels.

To identify the ontology definitions in the open environment, each ontology in this hierarchy will be assigned a name following the tradition of Directory Service ([54]). In this context, each ontology will be first assigned a name called the Relatively Distinguished Name (RDN), which identify the ontology with the sub-hierarchy under the parent node. The concatenation of all the RDNs of the ancestors nodes of an ontology on a path within this hierarchy, starting with the RDN of top ontology, is called a Distinguished Name (DN) of the ontology. A DN can uniquely identify an ontology in the universe of ontology definitions. However, an ontology can be assigned multiple DNs in case of multiple inheritance.

Figure 47: Ontology Hierarchy and Naming



As an example, the ontologies for the scenario discussed in the previous section will result in the ontology hierarchy depicted in figure 47.

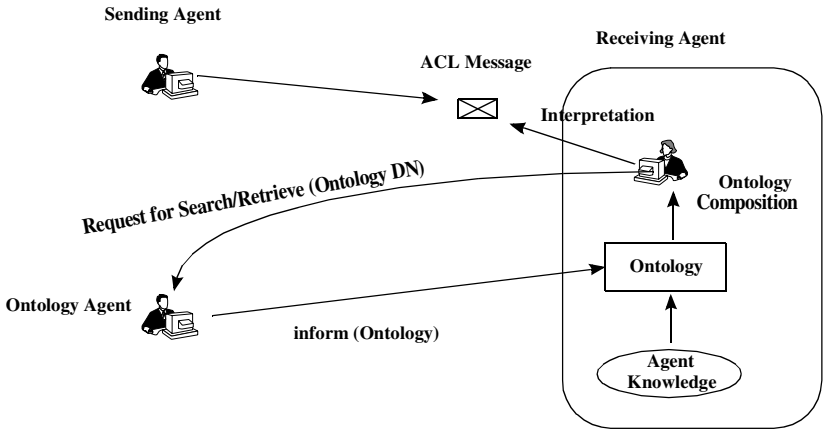
4.5.4.2. The Ontology Service Infrastructure

With the unique DNs, agents can locate or access ontology definitions at some OAs. During the interpretation of ACL messages, the ontologies needed for the interpretation do not nec-

essary exist in the local ontology knowledge of the agents. If some ontologies are missing, an agent usually retrieves them from an appropriate OA. The new ontology knowledge will be combined with the existing knowledge of the agent to support the interpretation of ACL messages and the decisions on the agent's co-operation behaviors.

It depends on the autonomy of the agent to decide on how to manage the new ontology knowledge. E.g. it can integrate the knowledge into his knowledge-based, so that the knowledge can be *reused* in future agent co-operations, or it can delete it after the current agent co-operation session if it is no longer needed.

Figure 48: Ontology Service



Ontology service is defined in the agent architecture as a service which supports the

– assert, retract, search and replacement

of ontologies in a group of ontology hierarchies. In another works, an ontology service is used to maintain the ontology hierarchies.

The hierarchical structuring of ontologies can be best reflected by hierarchical information services like the ITU-T X.500 Directory Services (DS - see [54]). The ontology service implements in fact a directory service within the agent communications framework and is typically implemented on top of a directory service.

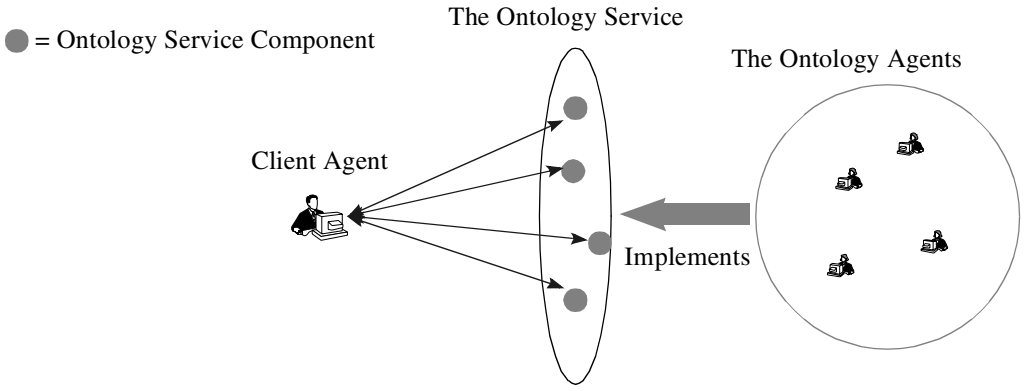
The Java Naming and Directory Interface (JNDI - [91]), which offers an Java API for accessing the different Directory Services, can be used as the basis for implementing the OAs and the ontology service.

4.5.4.3. Ontology Service Components

Like any telecommunication services, the ontology service, as depicted in figure 49, offers a group of service components via which a client agent can manipulate and retrieve ontology knowledge in supporting its generation and processing of messages.

Different from most current ontology service definitions like that in FIPA [31], which work on the individual predicates and rules, our ontology service is mainly based on manipulating the ontology as a unit identified by its DNs or RDNs. This abstraction from the internal details allows heterogeneous representations of the ontologies in the ontology service for different application contexts. Such a flexibility is very important in the typical telecommunications environments.

Figure 49: Ontology Service Components



Each service component in this context supports a specific operation on the store of ontologies maintained by the ontology service. The FIPA-request protocol is supported by all the ontology service operations. The typical interaction sequences between the client agent and the OA are depicted by the UML sequence diagrams in figure 50 and figure 51.

Figure 50: Successful OA Operation

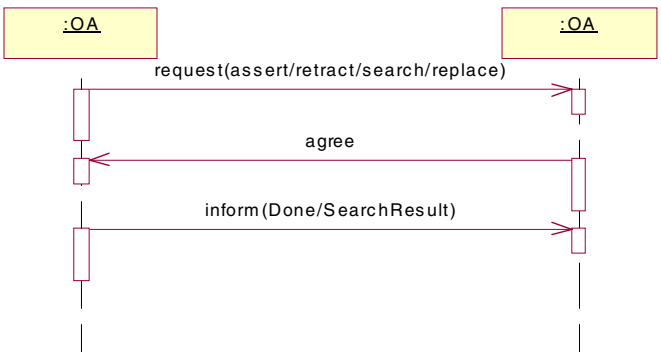


Figure 51: OA Operation Request Rejected

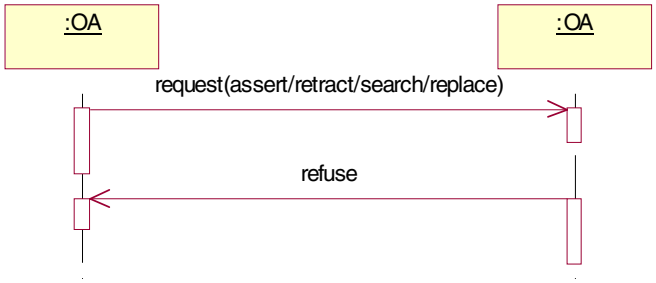
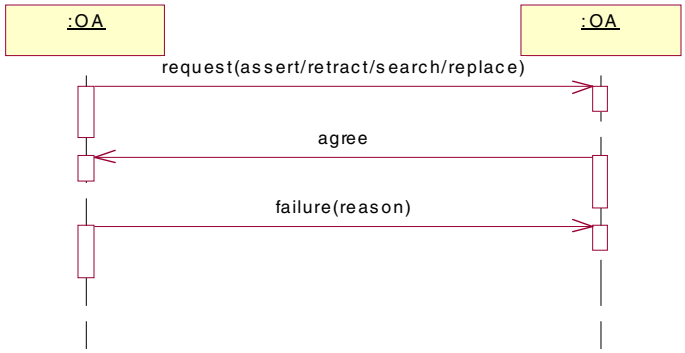


Figure 52: OA Operation Failed



The basic ontology service components are list in the following.

– assert

The *assert* service component will be used by a client agent to inform the OA a new ontology definition with a new DN. For this purpose, the client agent can issue the following ACL message

```
(request
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
          <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
           <os:Assert>
                <os:dn>example.ontology1</os:dn>
                <os:ontologyURL>
                      http://mimosa.fokus.gmd.de/projects/example/ontology1.rdf
                </os:ontologyURL>
           </os:Assert>
           </rdf> "
  :ontology ontology1 ontology2 ... ontologym
```

```
  :reply-with id
  ...)
```

Upon receiving such a message, the OA will try to register an new ontology in the hierarchy it maintains (under example.ontology1 in the above example), and will load the ontology definition from the specified URL, as depicted in figure 53. Notice in this and the following examples, words in italic like *client_agent* and *ontology_service_agent* refer to place holder/ abbreviation for some resource expressions. E.g. here *client_agent* refers to a real agent ID.

An ontology can only be inserted by some privileged agents. Inserting a node for a non-authorized agent will resulted in a refuse message from the OA.

```
(refuse
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
          <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
          <os:Failure>
                <os:reason> not authorized </os:reason>
          </os:Failure>
          </rdf> "
  :ontology ontology1 ontology2 ... ontologym
  :in-reply-to id
  ...)
```

Otherwise OA can agree to carry out the operation via the following message

```
(agree
  :sender ontology_agent
  :receiver client_agent
  :content"<?xml 1.0"?>
          <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
           <os:Assert>
                <os:dn>example.ontology1</os:dn>
                <os:ontologyURL>
                      http://mimosa.fokus.gmd.de/projects/example/ontology1.rdf
                </os:ontologyURL>
           </os:Assert>
          </rdf> "
  :ontology ontology1 ontology2 ... ontologym
  :in-reply-to id
  ...)
```

Finally, once the OA finishes its job it will send a confirmation to the client:

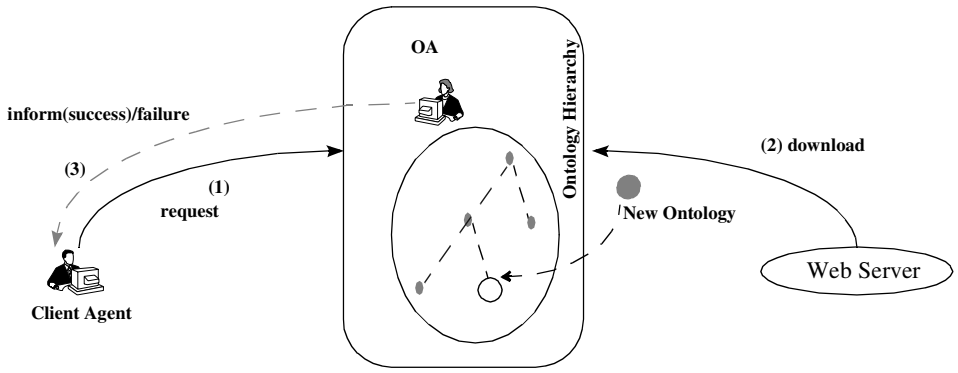
```
  (inform
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
          <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
          <os:Done/>
```

```
        </rdf> "
:ontology ontology1 ontology2 ... ontologym
:in-reply-to id
...)
```

If the operations failed, a failure message will be returned to the client.

```
(failure
:sender ontology_agent
:receiver client_agent
:content"<?xml version="1.0"?>
        <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
        <os:Failure>
              <os:reason> directory server down </os:reason>
        </os:Failure>
        </rdf> "
:ontology ontology1 ontology2 ... ontologym
:in-reply-to id
...)
```

Figure 53: Ontology Registration



Notice that ontology insertion is not only be used between an agent and an ontology server agent, it can also be used between a pair of any agents when one agent wants to teach the other a new ontology.

– retract

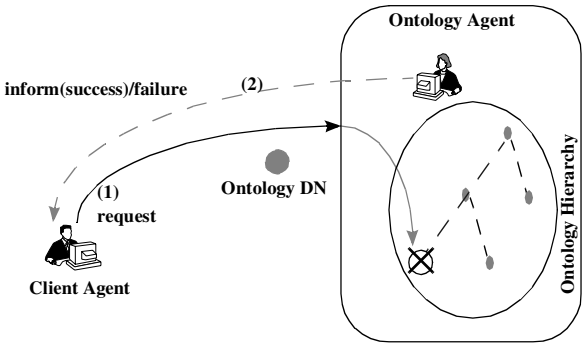
The *retract* service component is used by a client agent or the ontology service manager to delete an ontology definition from the hierarchy of ontologies maintained by an OA agent group. For this purpose, the client agent can issue the following ACL message

```
(request
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
```

```
            <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
             <os:Retract>
                  <os:dn>
                        connection.atm-vp-connection
                  </os:dn>
             <os:Retract>
           </rdf> "
:ontology ontology1 ontology2 ... ontologym
:reply-with id
...)
```

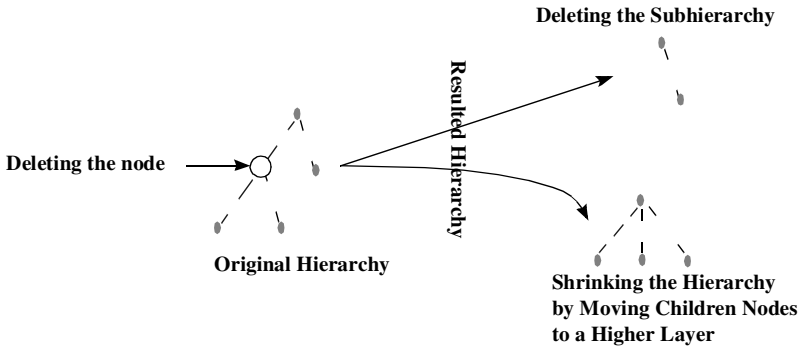
Upon receiving such a message, the contacted OA will try to delete the identified ontology from the ontology hierarchy by deleting the corresponding node and links in the knowledge base.

Figure 54: Ontology Deletion



Notice that an ontology to be deleted can be either a leaf node or a non-leaf node. In case the ontology to be deleted is non-leaf, the OSA will have two choices for realizing the deletion, as showed in figure 55.

Figure 55: Deleting a Non-Leaf Node



Notice that similar to assertion, ontology deletion is not only be used between an agent and an OA, but also be used between a pair of any agents when one agent wants the other to delete an ontology in its memory.

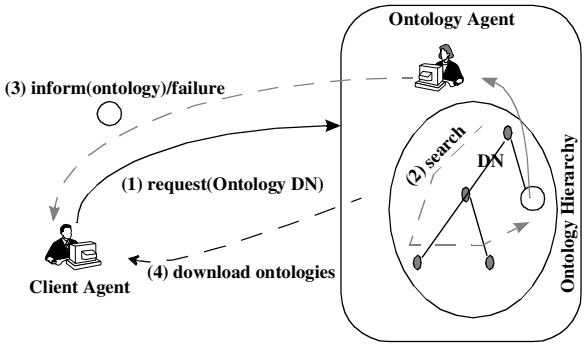
– search

The ontology *search* service component is used by a client agent to get a list of ontology definitions from the hierarchy of ontologies that satisfy the search condition. For this purpose, the client agent can issue the following ACL message

```
(request
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
          <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
           <os:search>
                <os:filter>
                       <os:context> connection </os:context>
                       <os:about> atmVPConnection </os:about>
                </os:filter>
           </os:search>
         </rdf> "
  :ontology ontology1 ontology2 ... ontologym
  :reply-with id
  ...)
```

which looks for ontology about a specific resource class *atmVPConnection* under the context *connection*.

Figure 56: Search for Ontology



Upon receiving such a message, the contacted OA will try to find the relevant ontologies from the ontology hierarchy, and send the URLs via an *inform* message to the client.

```
(inform
  :sender ontology_agent
  :receiver  client_agent
  :content"<?xml version="1.0"?>
```

```
            <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
              xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
            <os:ActionResult>
                  <os:actionResult_li> url1 </os:actionResult_li>
                  <os:actionResult_li> url2</os:actionResult_li>
            </os:ActionResult>
           </rdf> "
  :ontology ontology1 ontology2 ... ontologym
  :in-reply-to id
  ...)
```

The client agent then has to download all the ontologies from the URLs.

– replace

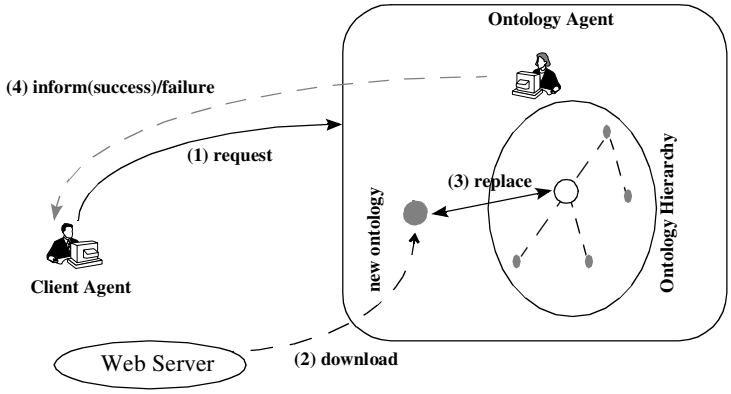
The replace service component will be used by a client agent to replace (i.e. modify) an ontology that is stored at the OA (or agent group). For this purpose, the client agent can issue the following ACL message

```
(request
  :sender client_agent
  :receiver ontology_agent
  :content"<?xml version="1.0"?>
           <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:os="http://mimosa.fokus.gmd.de/projects/example/ontology-service#">
            <os:Replace>
                  <os:dn> connection.user_view </os:dn>
                  <os:ontologyURL>
                        http://mimosa.fokus.gmd.de/projects/example/ontology1.rdf
                  </os:ontologyURL>
            </os:Replace>
          </rdf> "
  :ontology ontology1 ontology2 ... ontologym
  :reply_with id
  ...)
```

Upon receiving such a message, the ontology agent will try to find an existing ontology with the same DN and then replace it with the new ontology definition at the identified URL, as depicted in figure 57.

Notice that only an existing ontology can be replaced/modified. Replacing a non-existent node will resulted in a failure message from the ontology server.

Figure 57: Ontology Replacement



4.6. The Agent Template in Telecommunications Services

The autonomy and heterogeneity of co-operating intelligent agents, and the enhanced requirement for openness (i.e. an agent should be able to co-operate with another agent from which it has no or little knowledge) mean that an agent does not have the full knowledge about the functionality and behavior of the peer agents (or the associated interfaces) with which it is communicating. This is radically different from the traditional distributed systems where a single programmer or the joint programmer group can code the needed knowledge about the external co-operations into an autonomous system *before it is deployed*.

Each agent interacts with other agents because it wants to utilize these agents in achieving its private goals (which can be positioned within a joint, global goal of the environment). For this purpose, the agent has to act in a specific way which has the best chance to persuade the peer agents to co-operate. To decide on this specific way of acting, an agent must have at least some knowledge about the agents it is interacting with, and must behavior in a way that comply to the basic *social* rules that are enforced for the agent community.

Moreover, in order to teach another agent about changed requirements or technological knowledge, an agent must be in the position to assume some knowledge model that is supported by the partner agent. All these knowledge, social rules and knowledge model build up the *elementary ontology* that must be supported by the agents participating in the co-operations.

This elementary ontology plays an important role in at least the following contexts:

- It enables an agent to *initiate/start a co-operation relationship* and to dynamically improve/optimize the co-operation by accumulating, via agent communications, knowledge about the peer behaviors.

- It mandates some *rational behaviors* (which is a key feature of agents) on every agents in the environment, which can be used by agents in planning their interactions, and used by developers and users in guiding the design and deployment of agents.

- It offers a *knowledge framework* in which we can characterize the meanings of agent speech acts and co-operation protocols, define new speech act performatives or protocols, and even dynamically teach the agents about the meanings of these new performatives or co-operation protocols.

An agent architecture or skeleton that implement this initial ontology is called in this thesis

the *agent template*. An agent template in this context identifies only the minimal and external behaviors of the agents and does not put any constraints on how these external behaviors are implemented via the agent's internal autonomy and intelligence. In a real application environment, an agent template offers in fact the basis for implementing the specialized application agents and can be extended with application-specific intelligence to realize such application agents.

Due to the heterogeneity of application characteristics and requirements on agent's properties and behaviors for supporting openness and dynamic nature, it is very difficult to specify a universal agent template for all application contexts. However, we can still identify the general requirements of some restricted application domains and specify the basic agent templates for such domains.

The objective of this sub-section is to define an generic agent template that aims at fulfilling the requirement of dynamic, flexible and reliable co-operation in the telecommunications service provisioning and management environment. The agent template is therefore called *service agent template*.

Following the BDI-based (Belief-Desire-Intention) tradition of the IA studies ([7], [14], [24], [80], [82]) agents are usually modeled with mental models that simulate the abstract human emotions, attitudes and the rational behaviors relating to these emotions/attitudes in their co-operation with the environment. Because agent co-operations discussed in this thesis are mainly based on the external behaviors or views (i.e. how an agent sees it environment and its relationship to its environment), as discussed in Chapter 2, mental attitudes will play a more important role than emotions. Therefore we propose to base our agent template upon an appropriate mental model of attitudes for the agents.

The key issues here are to identify the set of mental attitudes appropriate for the application domain and to characterize the rational agent behaviors among such attitudes. One decision to be made in this context is the granularity of the attitudes. Generally speaking, every mental attitude can be refined and replaced by a set of more detailed (fine grained) mental attitudes following the psychological model of a human. E.g. the *wish* attitude, which can be considered as the external representation of the goals of the peer agent, can be refined to interests, preferences, fondness, objectives, priorities etc.

The FIPA model for ACL semantics ([24]), with a small set of attitudes, can be considered as a very abstract and large grain mental model for agent co-operations, while the studies for mental agents like those presented in [14] or [80] use some much bigger sets of fine grain attitudes to realize the more detailed rational agent behaviors in the agent operations.

Mental models based on fine grain mental attitudes envisage a high number of low level attitudes for the agents, simulating the detailed view of human attitudes in communicative actions. Such refinement of attitudes enables the detailed and relatively exact characterization of the semantics for agent communication speech acts.

However, such detailed views will also mandate detailed knowledge and sophisticated intelligence in realizing the agent's behaviors, and usually results in more complicated agent implementation. Therefore although a fine grain model seems to be appropriate for artificial intelligence agents, whose main responsibility is to simulate the human behavior, it can not justify its complication in many other application areas, especially in context of telecommunications applications.

Fine grain mental model also means subtle and vague difference between the different agent attitudes. With the heterogeneous backgrounds and heterogeneous requirements in the agent application environments, people tend to prefer different set of mental attitudes and different interpretation of such attitudes. This generally leads to the frequent doubts about and the debates on the exact meanings of each attitude.

The idea of large grain mental attitudes for speech act semantics is to model the agent using a small number of elementary attitudes whose meanings can be generally agreed in the agent community, and which can be easily supported by the agent implementations. Such agent models usually try to characterize only at a higher/abstract level the agent communication speech acts and leave the implementation the freedom to interpret the other part of the semantics.

One criticism (as mentioned in some comments on FIPA ACL specifications) on the large grain model for agent communication semantics is that it does not fully restrict the implementation of individual agents. This freedom is however very important for the deployment of agent technology in telecommunications applications, especially within a heterogeneous and open market environment. The mental model for the service agent template in this paper will be therefore close to the large grain end of the spectrum for knowledge-based agent communications.

Another distinct feature of this mental model lies in the focus of the objectives for deploying the agent communication technology. Different from the current frameworks for agent communication languages, which focus on statically defining the semantics for ACL performatives in agent communications, the *mental model* for service agent template focuses on enabling the dynamic and accumulative run-time co-operations among agents in the telecommunications applications. As a result, this model will

- be based on a representation which enables the utilization of the agent communication to support the automatic generation/manipulation of, and the reasoning upon partial co-operation knowledge,

- adopt a set of mental attitudes that most appropriately reflect the agent behaviors in the provisioning and management of telecommunication services.

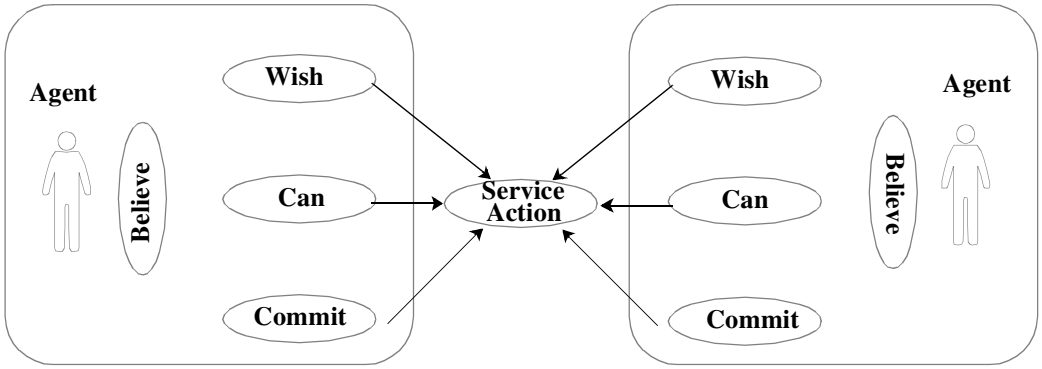
This service agent template based on the service mental model offers the basis for initial agent co-operation in the telecommunications service management environments. This co-operation can be dynamically extended in individual application environment by acquiring new ontologies for modeling extended agent mental behaviors.

4.6.1. The Mental Model for Service Agents

Within the context of telecommunications applications, the main category of agent interactions lies in the context of provisioning and managing telecommunications services that are positioned on all logical layers from network element to business processes. The main mental attitudes of agents will be therefore oriented towards the requirements in activities like the selection, negotiation, provisioning/consumption and management of different telecommunication services.

Generally speaking an agent plays either the role of user/customer or the role of a provider for a service relationship within a service value chain. It therefore possesses certain knowledge about the service and its co-operating partner, some resources, capability and commitments, and at the same time has certain needs/requirements determined by its internal goals.

Figure 58: The Agent Mental Model in Service Provisioning and Management



This abstract view of agents based on the mental attitudes can be depicted in figure 58. In this view, all the agent's attitudes are related to the service (and service resources) exchanged between the agents. Basically an agent can have the following attitudes concerning the provisioning and utilization of a service.

– wish

An agent, depending on its internal goal, can wish the deployment or provisioning of a ser-

vice. This is modeled by the mental attitude *wish*. The fact that an agent wishes an service action can be represented by the following resource description in CL:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Wish>
    <at:agentID> agent1 </at:agentID>
    <at:serviceAction>
      <schema:Action>
          <schema:agentID> agent2 </schema:agentID>
          <schema:service> service </schema:service>
      </schema:Action>
    </at:serviceAction>
</at:Wish>
...
</rdf:RDF>
```

which indicates the fact that $agent_1$ wishes that $agent_2$ provides the *service*.

Each agent within the service environment must autonomously decide, based on its goals, whether an action (e.g. a service offer) can meet its needs and is therefore wished. E.g. if an agent *A* requests an ATM connection with 20 Mbits/second, *provider1* offers 15 Mbits/second, *provider2* offers 10 Mbits/second, *A* can use its goals to decide that 15Mbits/second can be an acceptable compromise while 10 Mbits/second can not satisfy its requirement. The next activity of *A* can be to accept the service offer from *provider1*. The *wish* attitude is also used to model the agent's capability in deciding whether some resource meet its needs and goals.

– can

Each agent has certain capability that can change during its lifetime. Therefore, to participate in the service chains, the agent must have the possibility to dynamically inform other agents, either via active notification or via passive polling from peer agents, its current view of its own capability. Such a view of capability builds up the *can* attitude of the agent. The fact that an agent can carry out an service action can be represented by the following resource description in CL:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
```

```
plate#">
...
<at:Can>
    <at:agentID> agent_1 </at:agentID>
    <at:serviceAction>
      <schema:Action>
          <schema:service> service </schema:service>
      <schema:Action>
    </at:serviceAction>
</at:Can>
...
</rdf:RDF>
```

which indicates the fact that $agent_1$ can (i.e. has the capability to) provide the *service*.

– commit

Within the value chain for telecommunications services, each agent has to take up some responsibilities and has to commit to such responsibilities. Typical commitments in this context can be either the commitment to offer a specific service, or the commitment to use a service resource (and to pay for it). Such commitments are modeled by the mental attitude *commit*. The fact that an agent commit to carry out a service action can be represented by the following resource description in CL:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Commit>
    <at:agentID> agent_1 </at:agentID>
    <at:serviceAction>
      <schema:Action>
          <schema:service> service </schema:service>
      <schema:Action>
    </at:serviceAction>
</at:Commit>
...
</rdf:RDF>
```

which indicates the fact that $agent_1$ commits to provide the *service*.

Theoretically, a commitment is associated to a contract and some kind of punishment for breach of contract must be specified. This thesis however, will not try to automate these detailed legal issues via the service agent template. In fact, we expect that human interference

will be needed if a breach of contract is detected.

All these three attitudes can be considered as the refinements of the *intention* attitude used in FIPA Part II specification ([24]) for communicative act semantics.

– believe

With its autonomy and intelligence, each agent can have its own view on the resources and relations in the environment, i.e. an agent can believe in something or disbelieve something. The *believe* attitude is used to model such autonomous opinion of the agents. The following RDF description can be used to indicate that the agent believes in the truth, i.e. the existence of the *resource*.

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Believe>
    <at:agentID> agent_1 </at:agentID>
    <at:resource>
      resource_description
    </at:resource>
</at:Believe>
...
</rdf:RDF>
```

Basically an agent talks to other agents about its believes, i.e. what it maintains as valid in its knowledge. All the other attitudes in the communication can be logically regarded as being based on the agent's believes.

To simply the model, we also assume that an agent either believes in something or dis-believes in something. Uncertainty will be interpreted as dis-belief. Such a simplification is justified in telecommunications service interactions, as uncertainty has to be avoided in most cases to delete ambiguity in service relationships.

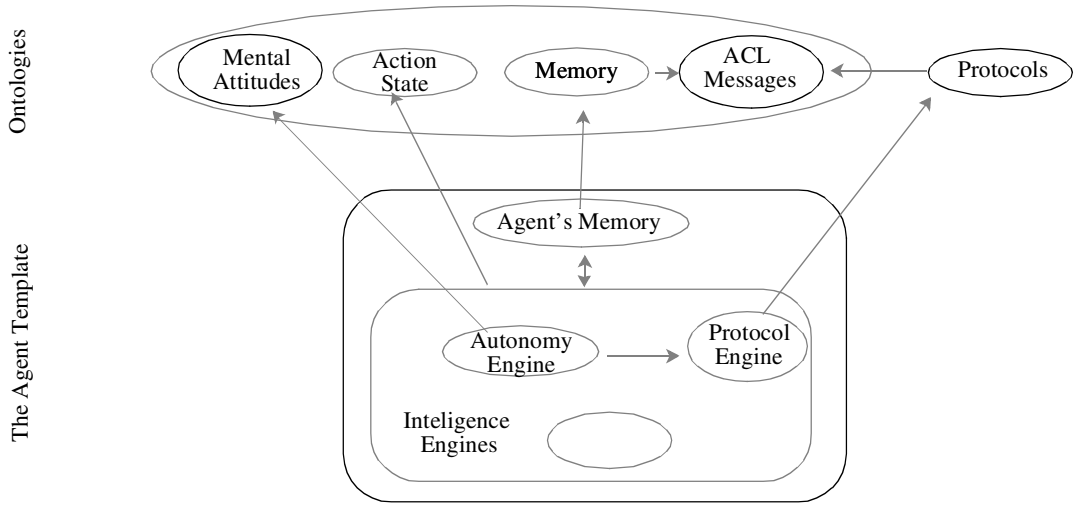
4.6.2. The Service Agent Template

With the support of the service mental model as its main objective, the service agent template implements the elementary ontology and the basic social behaviors of an agent in the tele-communication service value chain. For this purpose this agent template will have the archi-tecture as in figure 59.

The agent template is implemented via a memory of exchanged messages and a group of

agent's intelligence engines that either implement generic intelligence or are specialized in a dedicated co-operation functionality.

Figure 59: The Service Agent Template



Among the group of intelligence engines, two engines, i.e. the Autonomy Engine and the Protocol Engine are mandated in this agent template, each implements a part of the elementary ontology for the service agent template. Notice that in this thesis we focus only on the external interfaces/ontologies of the agents for supporting agent interoperability, and give the developers the freedom to choose the ways how this ontologies are implemented using the agent's internal intelligence.

The agent *Autonomy Engine* decides on the agent *attitudes* towards the resources and services in the environment. For this purpose, this engine utilizes the internal intelligence to determine the relationships between the resources/services and agent's goals and capabilities, and to derive the agent's attitudes from such relationships.

The *Protocol Engine* enforces the co-operation *protocol* in terms of the exchanged *ACL messages* that guides the current co-operation session between the agents. These protocols are regarded as special resources following the protocol ontology. This part of the elementary ontology will be discussed in detail later in Chapter 5 together with some scenarios for dynamic co-operation protocol definitions and configurations.

Besides the attitudes, the protocols and the ACL messages, the service agent template also support the following ontological elements in its co-operation interfaces:

– the *memory* of exchanged ACL *messages* and

– the view on the *states of the actions* associated to the provisioning and deployment of services.

The memory of an agent is realized via the following expressions:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          mlns:acl="http://mimosa.fokus.gmd.de/projects/example/acl-message#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Received>
    <acl:message>
      acl_message
    </acl:message>
</at:Received>
...
<at:Sent>
    <acl:message>
      acl_message
    </acl:message>
</at:Sent>
...
</rdf:RDF>
```

where the *Received* expression is valid if the agent has received the *acl_message*, the Sent expression is valid only if the agent has sent the *acl_message*.

States of a service action is modeled by the *Done* and *Failure* states. The following RDF documents can be used to represent the status of an action.

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Done>
    <at:action>
      action
    </at:action>
</at:Done>
...
</rdf:RDF>
```

or

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">
...
<at:Failure>
    <at:action>
      action
    </at:action>
    <at:reason>
      reason
    </schema:reason>
</at:Failure>
...
</rdf:RDF>
```

In summary, the elementary ontology supported by the service template agents can be defined in the following RDF schemas:

– *acl-message* defines the ontology for ACL messages

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#">

<rdfs:Class rdf:ID="ACLMessage">
  <rdfs:comment>
    Represent an ACL message in the RDF format.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#:Resource">
</rdfs:Class>

</rdfs:Property>
<rdfs:Property rdf:ID="performative">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="sender">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>
</rdfs:Property>
```

```
<rdfs:Property rdf:ID="receiver">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="content">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="language">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="ontology">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="reply-with">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="in-reply-to">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="protocol">
  <rdfs:domain rdf:resource="#ACLMessage"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

</rdfs:Property>
<rdfs:Property rdf:ID="conversation-id">
  <rdfs:domain rdf:resource="#ACLMessage"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>
</rdf:RDF>
```

– *service-agent-template* defines the ontology for agent's attitudes, memory and action view. The definition of protocol resources will be discussed in the next chapter.

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#">

</rdfs:Class>
<rdfs:Class rdf:ID="Wish">
  <rdfs:comment>
    Corresponds to the wish attitude of an agent.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Can">
  <rdfs:comment>
   Corresponds to the can attitude of an agent.
  </rdfs:commit>
  <rdfs:subClassOf  rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Commit">
  <rdfs:comment>
    Corresponds to the commit attitude of an agent.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Believe">
  <rdfs:comment>
    Corresponds to the believe attitude of an agent.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Received">
  <rdfs:comment>
```

```
    Used to check whether certain type of messages were received.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Sent">
  <rdfs:comment>
    Used to check whether certain type of messages were sent out.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Done">
  <rdfs:comment>
    To indicate that an action was successfully finished.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Class rdf:ID="Failure">
  <rdfs:comment>
    To indicate that an action was either refused or has failed.
  </rdfs:comment>
  <rdfs:subClassOf  rdf:resource = "http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource">
</rdfs:Class>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#Wish"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

<rdfs:Property rdf:ID="serviceAction">
  <rdfs:domain rdf:resource="#Wish"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/management-
knowledge#Action"/>
</rdfs:Property>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#Can"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

<rdfs:Property rdf:ID="serviceAction">
  <rdfs:domain rdf:resource="#Can"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/management-
```

```
knowledge#Action"/>
</rdfs:Property>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#commit"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#:Lit-
eral"/>
</rdfs:Property>

<rdfs:Property rdf:ID="serviceAction">
  <rdfs:domain rdf:resource="#commit"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/management-
knowledge#Action"/>
</rdfs:Property>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#believe"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>

<rdfs:Property rdf:ID="resource">
  <rdfs:domain rdf:resource="#Believe"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
19990303#Resource"/>
</rdfs:Property>

<rdfs:Property rdf:ID="message">
  <rdfs:domain rdf:resource="#Received"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/acl-mes-
sage#ACLMessage"/>
</rdfs:Property>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#Received"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#:Lit-
eral"/>
</rdfs:Property>

<rdfs:Property rdf:ID="agentID">
  <rdfs:domain rdf:resource="#Sent"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#:Lit-
eral"/>
</rdfs:Property>

<rdfs:Property rdf:ID="message">
  <rdfs:domain rdf:resource="#Sent"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/acl-mes-
sage#ACLMessage"/>
</rdfs:Property>
```

```
<rdfs:Property rdf:ID="action">
  <rdfs:domain rdf:resource="#Done"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/management-
knowledge#Action"/>
</rdfs:Property>

<rdfs:Property rdf:ID="action">
  <rdfs:domain rdf:resource="#Failure"/>
  <rdfs:range rdf:resource="http://mimosa.fokus.gmd.de/projects/example/management-
knowledge#Action"/>
</rdfs:Property>

<rdfs:Property rdf:ID="reason">
  <rdfs:domain rdf:resource="#Failure"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/
>
</rdfs:Property>
</rdf:RDF>
```

4.6.3. Defining Speech Act Performatives within the Agent Template

One application of the elementary ontology supported by the service agent template is to support the definition of new performatives for the speech acts. Before going on to discuss this possibility, we will first show, via some examples, how this ontology can be used to define the meanings of the elementary ACL performatives selected in the section 4.3.

As an example, for the *inform* performative, whose purpose is to tell the recipient agent about sender's belief in the validity of a *resource description*, we can have the following definition:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#"
          xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">

<schema:Equivalent rdf:ID="inform">
  <schema:head>
     <at:ACLMessage>
        <at:performative> inform </at:performative>
        <at:sender> ?I </at:sender>
        <at:receiver> ?J </at:receiver>
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?ResourceDescription </at:content>
     </at:ACLMessage>
  </schema:head>
  <schema:body>
```

```
    <schema:body_li>
     <at:Believe>
        <at:agentID> ?I </at:agentID>
        <at:resource> ?ResourceDescription </at:resource>
     </at:Believe>
    </schema:body_li>
    <schema:body_li>
      <pd:Not>
        <pd:expression>
         <at:Believe>
                    <at:agentID> ?I </at:agentID>
                    <at:resource>
                           <at:Believe>
                                  <at:agentID> ?J </at:agentID>
                                  <at:resource> ?ResourceDescription </at:resource>
                           </at:Believe>
                    </at:resource>
              </at:Believe>
          <pd:expression>
       </pd:Not>
    </schema:body_li>
</schema:Equivalent>
</rdf:RDF>
```

which simply means that agent *?I* believes in *?ResourceDescription* and does not think that agent *?J* already believes *?Resource*.

Another example is the *accept-proposal* speech act, which will always be used as a *response* to a preceding received *propose* message. With this message the sending agent informs the receiver that it agree with the action previously proposed by the receiver. We have in this case

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#">

<schema:Equivalent rdf:ID="accept-proposal">
  <schema:head>
     <at:ACLMessage>
        <at:performative> accept-proposal </at:performative>
        <at:sender> ?I </at:sender>
        <at:receiver> ?J </at:receiver>
        <at:in-reply-to> ?K </at:in-reply-to>
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?Action </at:content>
     </at:ACLMessage>
  </schema:head>
  <schema:body>
    <schema:body_li>
     <at:Received>
      </at:ACLMessage>
        <at:performative> propose </at:performative>
```

```
          <at:sender> ?J </at:sender>
          <at:receiver> ?I </at:receiver>
          <at:reply-with> ?K </at:reply-with>
          <at:concersation-id> ?L </at:concersation-id>
          <at:content> ?Action </at:content>
        </at:ACLMessage>
       </at:Received>
      </schema:body_li>
      <schema:body_li>
       <at:Commit>
        <at:agentID> ?I </at:agentID>
        <at:serviceAction> ?Action </at:serviceAction>
       </<t:Commit>
      </schema:body_li>
     </schema:body>
</schema:Equivalent>
</rdf:RDF>
```

which characterizes the meanings of the *accept_proposal* message as

- the sender received a preceding *propose* message,
- this *accept-proposal* message responses to that *propose* message,
- the sender informs that the proposed action meets somehow sender's current goals or requirements and the sender commits to use result of the action (e.g. service provisioning via the agent *?J*).

Similarly, we can define the semantics of all the other elementary speech acts using the elementary ontology supported by the service agent template. However, the interesting feature enabled by the service agent template based on mental attitudes is the possibility to define the semantics of new speech acts and dynamically integrate such new speech acts via loading and interpreting the semantics.

In fact, the heterogeneous telecommunications environments have also heterogeneous requirements on the set of speech acts for agent interactions, e.g. to guarantee the efficient and reliable co-operations. It is generally very difficult to specify a static set of basic speech acts for the different application contexts. One solution in this context is to allow the dynamic definition and deployment of new speech acts during agent co-operations.

E.g. to accelerate the negotiation process, an agent can directly commit to some action without first proposing the action and waiting for the proposal to be accepted as in the case of FIPA-contract-net protocol. In this case we need a new speech act, which can be called *commit*, with the following semantics:

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#"
```

```
         xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">

<schema:Equivalent rdf:ID="commit">
  <schema:head>
     <at:ACLMessage>
        <at:performative> commit </at:performative>
        <at:sender> ?I </at:sender>
        <at:receiver> ?J </at:receiver>
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?Action </at:content>
     </at:ACLMessage>
  </schema:head>
  <schema:body>
    <schema:body_li>
     <at:Can>
        <at:agentID> ?I </at:agentID>
        <at:resource> ?Action </at:resource>
     </at:Can>
    </schema:body_li>
    <schema:body_li>
     <at:Commit>
        <at:agentID> ?I </at:agentID>
        <at:resource> ?Action </at:resource>
     </at:Commit>
    </schema:body_li>
</schema:Equivalent>
</rdf:RDF>
```

I.e. by sending the commit message, an agent says that it is capable of carrying out some action, and at the same time commits to such an action. Usually if the receiver does not refuse this action in time, the sender will continue to start the action without waiting for further confirmations.

Similary, in case of negotiations, if an agent propose some service action, the client, instead of simply accepting or refusing the proposal, can make a counter proposal. For this purpose we can define a new speech act called *counter-propose*.

```
<?xmlversion="1.0"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
          xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowl-
edge#"
          xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-tem-
plate#"
          xmlns:pd="http://mimosa.fokus.gmd.de/projects/example/pre-defined-
functions#">

<schema:Equivalent rdf:ID="counter-propose">
  <schema:head>
     <at:ACLMessage>
        <at:performative> counter-propose </at:performative>
        <at:sender> ?I </at:sender>
        <at:receiver> ?J </at:receiver>
```

```
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?Action </at:content>
      </at:ACLMessage>
    </schema:head>
    <schema:body>
      <schema:body_li>
       <at:Received>
              </at:ACLMessage>
                     <at:performative> propose </at:performative>
                     <at:sender> ?J </at:sender>
                     <at:receiver> ?I </at:receiver>
                     <at:reply-with> ?K </at:reply-with>
                     <at:concersation-id> ?L </at:concersation-id>
                     <at:content> ?Action1 </at:content>
              </at:ACLMessage>
        </at:Received>
      </schema:body_li>
      <schema:body_li>
       <at:Wish>
          <at:agentID> ?I </at:agentID>
          <at:serviceAction> ?Action </at:serviceAction>
       </at:Wish>
      </schema:body_li>
      <schema:body_li>
       <pd:different>
          <pd:expression> ?Action1 </pd:expression>
          <pd:expression> ?Action </pd:expression>
       </at:different>
      </schema:body_li>
  </schema:Equivalent>
  </rdf:RDF>
```

Following this definition, a counter proposal says that the sender has received action proposal but it prefers another action which is different from (i.e. the proposed one.

4.7. Summary

This chapter presents an agent architecture with its layers of component technologies for jointly supporting the knowledge-based and dynamic interoperability among agent-based applications. This architecture can be used as the basis for developing the integrated agent platforms and agent applications.

The core issue in this architecture is to enable the dynamic adapation of agent co-operation relationships within the telecommunications service value chains, mainly via exchanging knowledge about the service resources and requirements during service co-operations.

This knowledge-based interoperability is implemented via a speech act-based agent communication platform that is compliant to the FIPA standard. This platform is realized on top of the OMG MASIF conformant mobile agent platform in order to support the utilization of both the MA and IA technologies in the telecommunications environment.

Knowledge representation within the agent communication message content is based on an extended XML/RDF framework, allowing the easy and smooth integration into Web-based environment. An ontology framework is presented which utilizes the RDF-based knowledge representation paradigm to build up hierarchical, accumulative and reusable ontological knowledge. An ontology service is also specified using this knowledge representation framework. This ontology service maintains the store of ontologies that can be downloaded by the application agents for supporting the dynamic configuration of agent co-operations.

An service agent template is specified that implements the basic agent's behaviors and views to enable the initial and extensible agent interoperability within the opne environment.

The next chapter will try to validate this architecture by applying the technologies to a concrete scenario in telecommunications service provisioning and management.

CHAPTER 5

Proof of Concept - Agents in Telecommunications Service Management

5.1. Background

Due to the development of the telecommunications technology and market, telecommunications services are nowadays typically positioned within a dynamic, distributed, heterogeneous and even mobile environment. Different from a traditional service provisioning and management environment, where the service relationships and co-operation functionalities are typically pre-programmed or configured off-line, service management in this context has to deal with the rapidly changing and versatile business, technological and administrative requirements, and correspondingly the changing co-operation relationships and functions. Moreover, the integration of mobile users or mobile hosts in the service provisioning environment further requires the ability to migrate applications across the network nodes, and requires the enhanced ability for applications to work in foreign environment.

As argued in the previous chapters, agent technology, which supports knowledge-based dynamic interoperability, offers better solutions for co-operating autonomous applications than conventional technologies in this context.

This chapter will focus on showing the applicability and advantages of the agent-based solution presented in this thesis by applying such a solution to telecommunications service management. A scenario in the context of *multi-media and dynamic VPN service* as discussed in Chapter 1 is selected for proving and validating the agent framework and architecture.

5.2. The Multi-media Dynamic VPN Service

This Multi-media Dynamic VPN service can be considered as an extension to the numerous Global Connectivity Service/VPN services developed in many TMN- or agent-based research activities (e.g. [10], [29], [116])

Within the global telecommunications environment based on connection- and QoS-oriented technologies like ATM, SDH and IPv6, one key issue is to provision and manage the transport connectivities among a group of users involved in a telecommunications application. The development of the flexible and high bandwidth communication technologies nowadays enables a variety of applications utilizing such connectivities, ranging from co-operative working, audio/video conference to real-time or multi-robot systems. Together with the evolving business requirements of the users and customers, the global connectivities provisioned in this context have to support:

- dynamic adaptation to the evolving customer/user requirements or provider technologies,
- heterogeneous QoS, service features and functionality,
- reliability and flexibility in the environment with distributed autonomous resources,
- 'any time, any place' service provisioning in a mobile host environment.

The VPN service is considered in this context as the bearer active service which provides transport connectivities to users who want to set up multimedia application sessions (multimedia video conference) with several other users, especially within the dynamic and evolving telecommunications business environment.

The dynamic nature of the service environment is reflected in the dynamically changing service conditions (QoS, prices) with respect to the individual service session. As a result, users or service providers will have dynamic relationships to their providers depending the evolution of these service conditions.

In this thesis, we will only address the resource management, especially the connectivity resource management issue within the VPN service. Some other issues, like security, which are also important in a distributed and autonomous VPN environment, are regarded as out of the scope of this thesis.

A typical scenario for VPN provisioning in this context consists of three stages, which are necessary for co-ordinating and provisioning a multi-user application across a multi-media

VPN environment.

1. User Negotiation Stage

A user proposes to schedule a video-conference meeting and starts negotiating with participating users about a mutually convenient time for the conference. Parameters that influence this negotiation include time/date/duration, cost, etc. This stage results in a scheduled video conference meeting.

2. VPN Service Negotiation Stage

Within this stage the initiating user will contact one or more (VPN) service providers, requesting the required services. Service negotiation will follow based on the replies. Negotiable parameters include time/date/duration, cost, security, quality of service, etc. Each service provider must then negotiate with one or more network providers to arrange the provision of the required VPN connections. The connectivity negotiation parameters will actually be generated from those used in the user service negotiation, but now mapped to the network level. Network providers may also co-operate with each other in order to achieve the required connectivity.

This stage results in the choice of a particular service provider and a set of service level agreements, for the actual VPN service and the required connections.

3. VPN Service Deployment Stage

The service provider requests the network providers to set up the network connections. When these are in place the involved end-users are informed that the VPN is available for use. During the VPN connection lifetime, management information released by any of the parties is passed on to relevant parties and appropriate actions can be taken. The service provider controls the decommissioning of the VPN (triggered by the initiating end-user or by a reservation system controlled by the provider). Participating users are then requested to terminate their VPN link and network providers are informed the VPN is terminated. Finally, network providers will charge the service provider for the use of their network. The service provider can add additional charges for the service and will propagate them to the end-user.

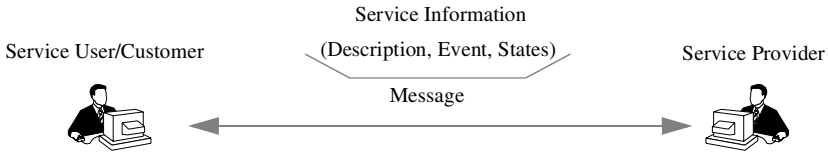
5.3. Agents for Telecommunications Service Management

As discussed in Chapter 1, the key feature of the agent technology is to support the dynamically adaptable, robust and flexible co-operations among autonomous software systems via knowledge-based agent interoperability. Such a feature can play key role in supporting the

reliability, robustness, flexibility and the dynamic adaptability in the provisioning and management of customer-oriented telecommunications services.

As depicted in figure 60, knowledge-based interoperability supports the enhanced service co-operation functionality mainly by exchanging co-operation knowledge between the provider and the user/customer of the services, which is realized by coding the *knowledge* within the message contents.

Figure 60: Knowledge-based Interoperability in Service Co-operations



By identifying and exchanging such knowledge between the service user/customer and providers, the agent-based service management framework supports

- initial interoperability and co-operation between foreign service applications, and
- the dynamic adaptation or evolution of services or service features.

Dynamic service adaptation can be either realized via MA, i.e. agent mobility, or via exchanging ontological knowledge for guiding the behaviors of the receivers. The second case can also be called *functional mobility*. Different from agent mobility, functional mobility relies on the autonomy and intelligence of the receiver agents for the message interpretation and possibly, continuous message migration (i.e. the migration of the functionality contained) to other agents.

Basically, knowledge-based agent interoperability and dynamic service adaptation based on agent technology can offer the following features in service provisioning and management.

- dynamic programmability

With MAs and the knowledge-based agent communication messages, customers, users/managers and co-operating telecommunication resources or service providers can dynamically adapt and program the funcationalities of the remote resources for their own needs in the value chains of the telecommunication services. Via intelligent negotiations, service customers and providers are in the positions to dynamically change the business requirements, conditions, relationships and objectives, in order to guide and to enable the programming of the services. It is in this sense that agent-based telecommunications service can be also called

active service ([124]), similar to the concept of *active network* ([89], [120]), which is rapidly gaining importance in the context of the network control and management.

– generic and customer-oriented services

With the concept of agent-based service provisioning and management, a service provider generally can offer some elementary and generic service facilities which are reusable in a wider context and rely on the customers to combine and adapt such elementary facilities to obtain the required service. With such a fine grain characteristic of agent-based service facilities, this new paradigm can achieve

- maximal reusability of the generic service resources
- minimal costs for the service providers in the service provisioning

With the programmability and generic character of the service resources, the new paradigm also enables the provisioning of a large class of services tailored to the individual customer's needs, by using the same service infrastructure.

– efficiency, reliability and robustness in distributed and mobile environments

Agent or functional mobility help to reduce the dependency of service programming and management activities on the underlying network via large grain MAs or via ontology information in the functional mobility. This feature helps to increase the fault tolerance and robustness of the service instances and service negotiations when the underlying network for service management is unreliable or not constantly available (e.g. in case of mobile hosts).

On the other hand, the social and knowledge communication ability of the agents enables the service players to dynamically exchange knowledge about new or exceptional situations that are not recognized during the initialization phase. Similarly, service players can also dynamically exchange solutions for dealing with new or exceptional situations.

Both possibilities help to enhance the flexibility, reliability and robustness of the active services.

The concept of dynamic service adaptation and programming has some similarity to the traditional *VAS (Value-Added Service)* paradigms. However, there are some key differences which distinguish the two approaches:

Within a VAS, a Value-added Service Provider (VASP), which is typically different from the end-users or customers, combines and adapts the services of the telecom providers in order to offer the service features required by the end-users or customers. The adapted service is in this case offered by the VASP to the users (see figure 61).

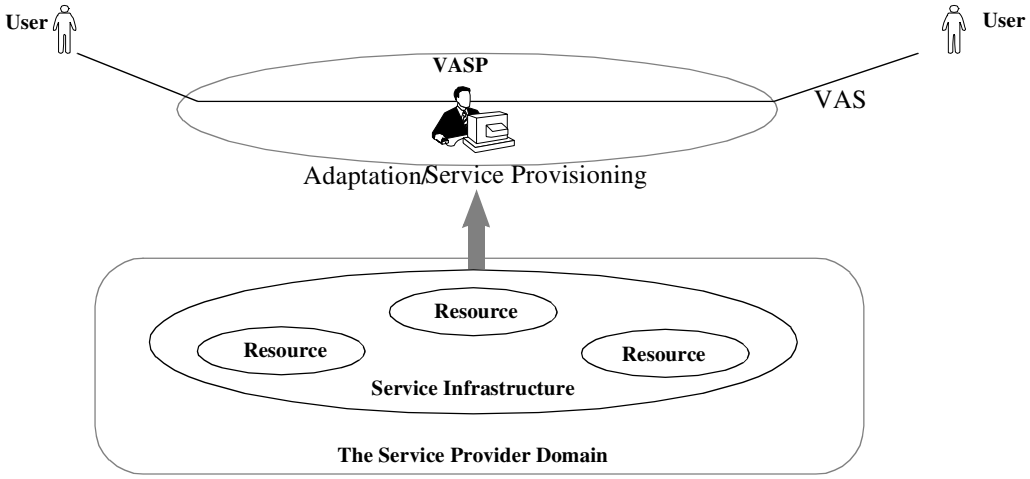
In case of agent-bases services, service programming and the programmed service are real-

ized directly within the domain of the service resource providers, and the resulted services are offered to the end-users directly from the service and resource provider.

This difference has some important influences on the provisioning of telecommunications services.

The service provider in a VAS scenario typically offers a relatively *complete* and *final* service from its viewpoint. The VASP has in this context limited freedom in programming dynamically the customer-oriented service features. Moreover, the value-added scenario requires the service programmer to offer at the same time the programmed service, and therefore limits the possibility of individual customers and users in obtaining services that are tailored to their dynamic needs. In fact, a customer, which usually also represents the users, do not necessary have the infrastructure of a *service provider*, e.g. it possibly does not have the accounting & billing, customer care and security facilities to play the role of third party service provider. The high costs associated to such a provider infrastructure is usually the reason which presents a customer or user from directly playing the role of a VASP.

Figure 61: Value-Added Services



More importantly, the services are statically adapted by the VASP, dynamic service programming means typically the implementation of a new VASP. With the versatility and complexity of the requirements, different users/customers will require different services and service features at different time, which would require a large number of heterogeneous VASPs to come up with all the requirements. Therefore, compared to the VASP paradigm, the agent-based service paradigm

– reduces the complexity of provider's service provisioning and management infrastructure and of its operations, and results the in reduced costs for both the providers and the customers,

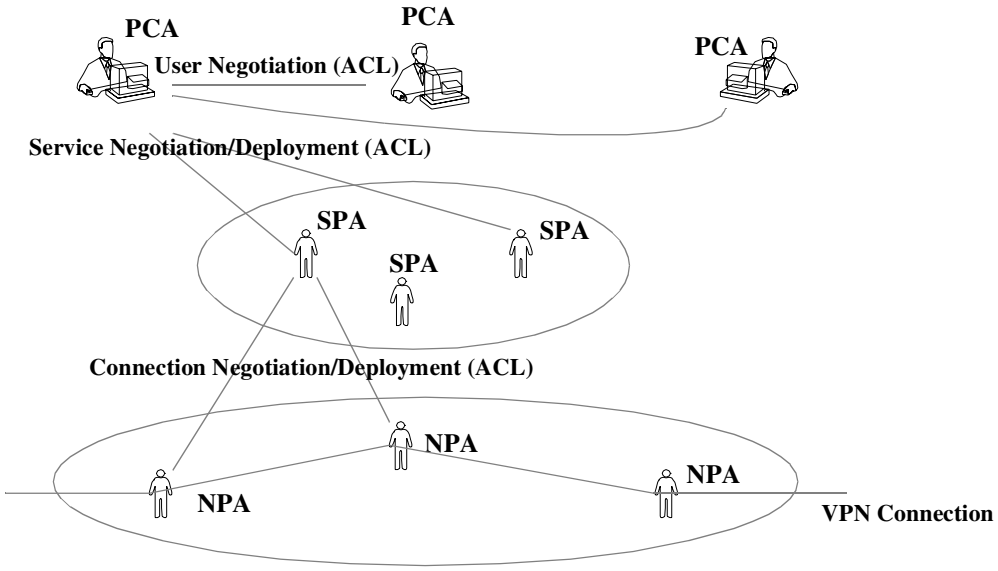
– increases the reusability of provider resource facilities,

– increases the flexibility, customer-orientation, and the optimization possibilities of the telecommunications services via the programmability of the service features.

5.4. Agent-based VPN Service

Within an agent-based VPN service framework, the actors or players in the service environment are implemented as agents ([29]), . The resulted configuration can be showed in figure 62.

Figure 62: Agent-based VPN Service



Three major categories of agents are envisaged in this service environment, each can be further implemented as a multi-agent system.

– Personal Communication Agent (PCA)

The PCAs are responsible for representing the users/customers in accessing the VPN service. Some PCAs can also play the role of customer for the VPN service by registering such a role at the providers via subscription.

PCA is also responsible for the interactions with the human users and with the local resources like video conference applications. During the VPN service deployment, the PCA will start

or terminate the video conference application following the availability of the network connections and the directives of the user.

Among others, the PCAs with the authority of a customer will also be responsible for the decision upon and the invocation (requesting) of the dynamic programming and adaptation of the services based on user's dynamic requirements.

As a representative of the users and customers, the PCA's goal is to optimize the deployment of service resources (maximal performance, minimal prices) by the user applications. For this purpose, among others, the PCA has to dynamically adapt the service features following the changes in the user's world, to keep the users informed about changes in provider's domains and to help to adapt the user applications to such changes. Besides, a PCA will usually negotiate with several SPAs to get the best offer for a service session.

– Service Provider Agent (SPA)

A SPA represents a service access point for the VPN service. Its main responsibility is the provisioning and management of the VPN service requested by the users, which are realized via managing the connectivity resources provided by the NPAs. For the purpose of service provisioning, in a default scenario, a SPA will make the routing decisions and negotiate with multiple NPAs for the individual connection segments to establish the global connection between the end-users.

Among others, the SPA is also responsible for the implementation of the service adaptation and programming by managing its internal algorithms and by adapting the connectivity service interfaces provided by NPAs.

As the representative of the providers, the goal of a SPA is to optimize the provisioning of the resources (maximal profits/prices, and minimal resource consumption) while satisfying the requirements of the users and attracting a larger group of users by new, better service features and qualities. To this end, the SPA can dynamically modify/optimize its provisioning of the network resources and also has to be informed about the changes in the user's requirements or in the technological environments.

– Network Provider Agent (NPA)

The NPA provides the subnetwork network resources for the global network connectivities. Basically each NPA implements a subnetwork and interacts with the SPA to provide and manage the local resources.

As the representative of the network providers, the goal of a NPA is to optimize the provisioning of the connectivity resources (maximal profits/prices, and minimal resource consumption) while satisfying the requirements of the SPAs and attracting a larger group of

SPAs by new service features and qualities. To this end, the NPA can dynamically modify/ optimize its provisioning of the network resources and also has to be informed about the changes in the SPA's requirements or in the technological environments.

Within the VPN service, the concept of global network is mapped to the subnetworks of the distributed, autonomous and co-operating network providers. The global network infrastructure is considered as the set of NPA subnetworks, by which and through which the connectivities are to be provisioned and managed.

Each NPA subnetwork has a dedicated Network Management System (NMS), e.g. a CMIP-based PNO NMS, which supports the management view upon the underlying network resources. To abstract from the underlying specific transport technologies like ATM, SDH or IPv6 and to avoid too much details in this thesis, we can start with a default high level management view of the network provider's subnetwork resources in the following discussions.

Only bidirectional connections are considered in our context. Each such subnetwork connection has a basic set of QoS and performance parameters like cost, bandwidth etc.

Due to the dynamic service relationships among the players, the VPN service agents will have to dynamically negotiate their user/provider relationship in each service session (or every several sessions).

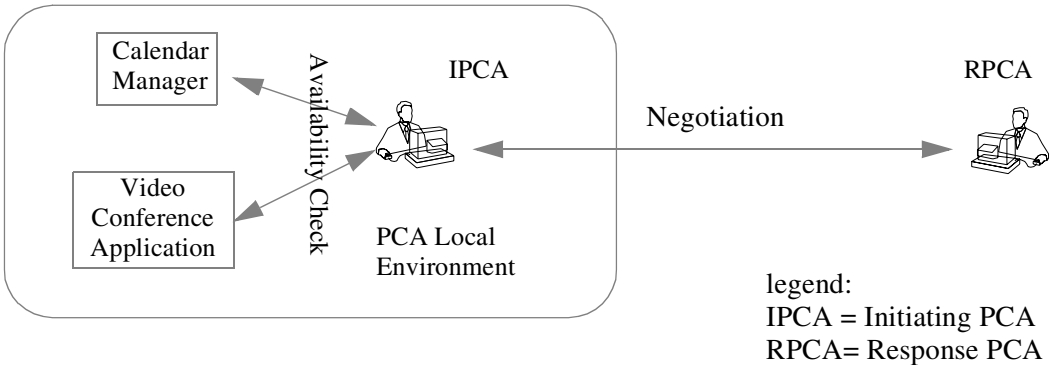
Three kinds of negotiations can be envisaged that correspond to the user negotiation and service negotiation stages in the VPN service provisioning as discussed above:

– PCA-PCA Negotiation

This negotiation will be used by the PCAs to agree upon a video-conference schedule based on mutually convenient time.

The negotiation environment for a PCA can be described by figure 63.

Figure 63: PCA-PCA Negotiation Environment



There is one PCA which initiates the negotiation, and which is responsible for managing the

application session to be established. This PCA is called Initiating PCA (IPCA). The PCAs which the IPCA negotiate with are called Response PCAs (RPCAs).

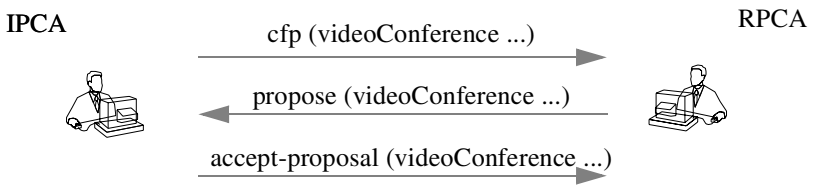
Each PCA in its negotiation with the other PCAs will check its Calendar Manager (CM) to find a free time slot for the conference, or to check the availability of the user within certain time slot. Besides, the PCA will also check, via the local facilities, the availability of the local resources (e.g. screen, audio tool) for the video conference session.

Basically, the PCAs talk to each other about video-conference and its properties, e.g. with the following description (see section 4.5).

```
<vc:VideoConference>
   <vc:windowSize>  14 </vc:windowSize>
   <vc:numberOfColours> 65536 </vc:numberOfColours>
   <vc:resolution> 1024,768 </vc:resolution>
   <vc:audioQuality> high </vc:audioQuality>
   <vc:schedule> schedule </vc:schedule>
   <vc:source> user1@fokus.gmd.de </vc:source>
   <vc:destination> user2@fokus.gmd.de </vc:destination>
</vc:VideoConference>
```

where the *user1* and *user2* are the names (e.g. e-mail addresses or URLs) of the involved users, and *schedule* is a structure that identify the time slot(s) for the meeting.

Figure 64: PCA-PCA Negotiation Protocol

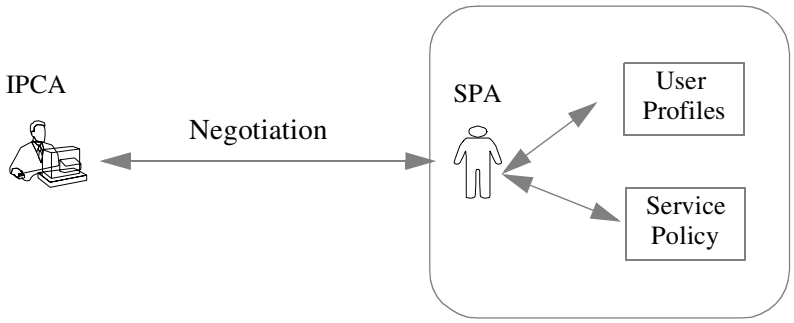


The default protocol for PCA negotiation is the FIPA-iterated-contract-net protocol as presented in Chapter 2. A typical sequence of interactions in this context can be depicted in figure 64.

– PCA-SPA Negotiation

The purpose of PCA-SPA negotiation is to get the service agreed among the co-operating PCA and SPA. The SPA in this context will have the internal architecture as depicted in figure 65.

Figure 65: PCA-SPA Negotiations



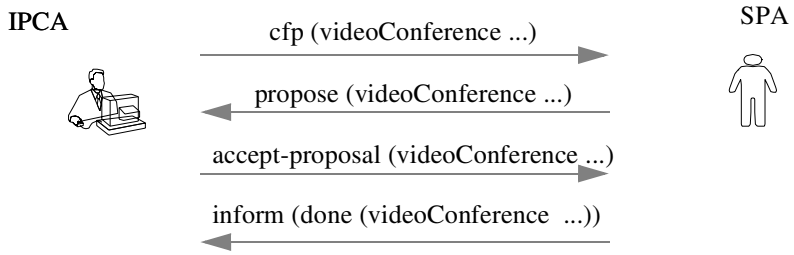
The IPCA, after reaching an agreement on the meeting schedule with the RPCAs, will start a negotiation with the SPA for reserving the service session.

The SPA maintains a profile for each user represented by the PCA and has an internal component which determines the policy of the SPA in accepting/granting service requests. The SPA utilizes these two facilities to make decisions on its responses to the IPCA.

Similar to PCA-PCA negotiations, IPCA negotiates with the SPA about the video conference and its properties, e.g. about the following service description

```
<vc:VideoConference>
   <vc:windowSize>  14 </vc:windowSize>
   <vc:numberOfColours> 65536 </vc:numberOfColours>
   <vc:resolution> 1024,768 </vc:resolution>
   <vc:startDateTime>19990324T150000000Z</vc:startDateTime>
   <vc:endDateTime>19990324T170000000Z</vc:endDateTime>
   <vc:audioQuality> high </vc:audioQuality>
   <vc:source> user1@fokus.gmd.de </vc:source>
   <vc:destination> user2@fokus.gmd.de </vc:destination>
</vc:VideoConference>
```

Figure 66: PCA-SPA Negotiation Protocol



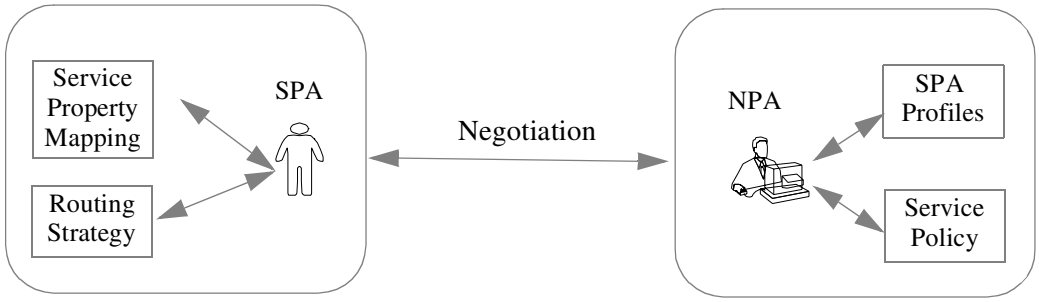
The default protocol for PCA-SPA negotiation is also the FIPA-iterated-contract-net protocol as presented in Chapter 2. A typical sequence of interactions in this context can be depicted in figure 66. Notice that different from PCA-PCA negotiation, after the service agreement was reached, the SPA has to inform the IPCA that the service is reserved. This

inform message will return the *id* of the service reserved. This *service-id* is necessary for the IPCA to initiate any management operations on the (not yet activated) service.

– SPA-NPA Negotiation

The purpose of SPA-NPA negotiation is to obtain the needed network connection (segment) for the global connectivity requested by the user. The environmental context for this negotiation can be depicted in figure 67.

Figure 67: SPA-NPA Negotiations



Either after the SPA achieves a service agreement with the IPCA, or before this happens, the SPA should negotiate with the NPAs for the real (subnetwork) network connections. For this purpose the SPA will first translate the service properties from the IPCA to network connection properties, with the help of some internal service property mapping algorithm (e.g. via the ontology definitions in section 4.5), and select the potential NPAs to be involved for connection segments based on the SPA's routing strategy.

Similar to SPA in its PCA-SPA negotiations, the NPA now plays the role of a service provider, therefore it has to maintain the profiles about the SPAs and to possess some service policy facilities for supporting the NPA's autonomous decisions in its negotiations.

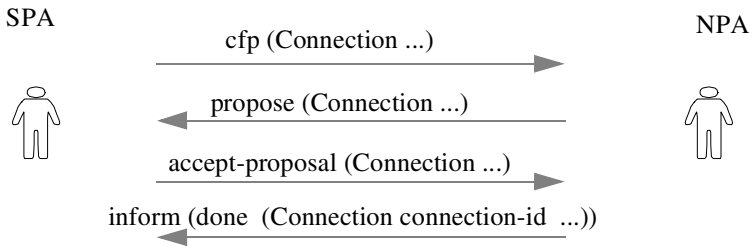
SPA-NPA talks about network connections instead of video-conferences, and they also need the IP addresses of the users to configure the connectivities required. A service description in this context can have the following properties:

```
<vc:Connection>
   <vc:bandwidth>  10 </vc:bandwidth>
   <vc:startDateTime>19990324T150000000Z</vc:startDateTime>
   <vc:endDateTime>19990324T170000000Z</vc:endDateTime>
   <vc:source> 193.175.133.135 </vc:source>
   <vc:destination> 193.175.135.72 </vc:destination>
</vc:Connection>
```

which identifies, among others the bandwidth of the connection to be reserved as 10 MBit/s.

The default protocol for NPA-SPA negotiation is also the FIPA-iterated-contract-net protocol as presented in Chapter 2. A typical sequence of interactions in this context can be depicted in figure 68. Similar to PCA-SPA negotiation, after the service agreement was reached, the NPA has to inform the SPA that the connection is reserved and also the *connection-id*. This is necessary for the SPA to initiate any management operations on the (un-activated) connection.

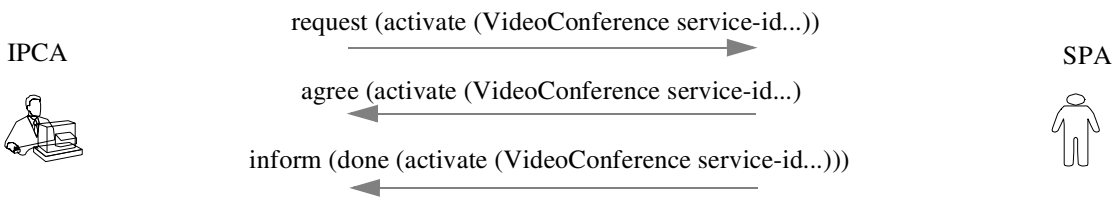
Figure 68: SPA-NPA Negotiation Protocol



Once the services and connections are reserved for a VPN session, the VPN agents go into the deployment stage of co-operations. During this period, a user (via the IPCA) can request the following management operations from the SPA:

- *activation*, which can be used to activate the service (suppose the schedule in the service agreement is suspended or does not exist),
- *modification*, which modifies the service instance within the range of the service agreement,
- *termination*, which terminates the service reserved.

Figure 69: PCA-SPA Service Deployment Protocol

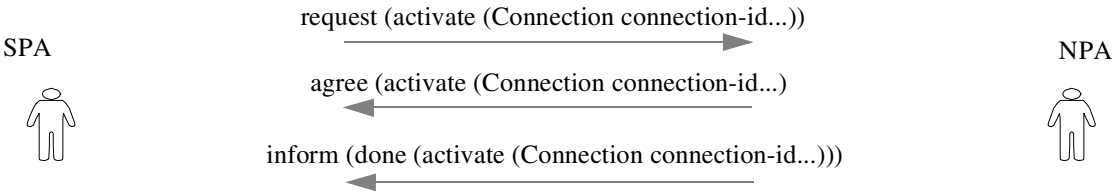


Service deployment between the PCA and SPA is based on the FIPA-request protocol as pre-

sented in Chapter 2. A typical interaction scenario can be depicted in figure 69.

To realize these requests from the IPCA, the SPA has to mapping such operations to operations on network *connections*. Correspondingly, SPA will request activation, modification or termination from the NPAs. The FIPA-request protocol is also deployed as the default interaction protocol in this context.

Figure 70: SPA-NPA Service Deployment Protocol



5.5. Knowledge-based Interoperability in VPN Service Provisioning and Management

VPN based on RDF and speech act communications enable a higher degree of interoperability in terms of platform independency and flexibility. However, as discussed above, the key feature, which distinguishes the agent-based service management from traditional framework, is the support for knowledge-based interoperability, and the resulted support for dynamic service programmability, customer-orientation and the robustness or reliability in the distributed environment.

This section will present some example scenarios in the context of VPN service provisioning and management, where the agent-based interoperability offers enhanced support for service interactions.

5.5.1. Dynamic Protocol Adaptations

Co-operation protocols play an important role in guaranteeing the flexible and reliable interoperability during the provisioning and management of telecommunications services. The most important category of co-operation protocols in this context are the negotiation protocols that aim at achieving certain agreement between the agents, which can satisfy the goals and wishes of all the agents involved.

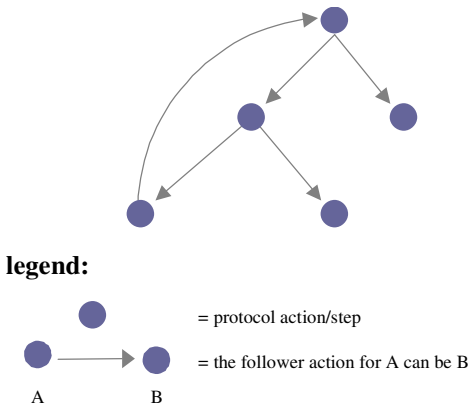
A negotiation process in this context is considered as the sequence of ACL messages exchanged between the agents. A negotiation protocol determines the possible message sequences in the negotiation process. Different negotiation protocols can have significant impact on the quality of the negotiation processes.

Within a concrete environment, some protocol will need more steps in the process, or need more time to reach an agreement between the parties (sometimes a negotiation will never reach an agreement), while some other protocols require less steps and time. At the same time, there could be also significant differences in the quality of the agreements reached. Depending on the situations in the environment, some protocols can increase the chance of achieving high quality (measured by the satisfaction of the involved agents) agreements, while the other does not guarantee the quality of the negotiation result.

As a result, an application like VPN must carefully select the negotiation protocols to ensure the performance in the service co-operations. Due to variety of protocols and the heterogeneity of the environments, it is in many cases not possible to hardcode all the possible protocol implementations into the agents. A better solution is therefore to allow the agents to dynamically decide on the deployment of new protocols and to download the necessary protocol definitions from some OAs.

The pre-condition in this context is the knowledge representation for protocol knowledge. In our agent framework protocols are defined via RDF documents and are regarded as ontologies. A protocol is define as a special resource class which specifies a directed graph of protocol actions and with the order of actions specified by the links, as depicted in figure 71.

Figure 71: The Directed Graph for a Protocol



The RDF schema in this case can be defined by the following document, which is named as *protocols*.

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
     xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#">

<rdfs:Class rdf:ID=”ServiceProtocol”>
   <rdfs:comments> This class describes a service negotiation protocol based  speechact  steps
   </rdfs:comment>
   <rds:subClassOf rdf:resource = ”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource”/>
</rdfs:Class>

<rdfs:Class rdf:ID=”ProtocolActions”>
   <rdfs:comments> This class describes the protocol actions within the service co-operation protocol
   </rdfs:comment>
   <rds:subClassOf rdf:resource = ”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource”/>
</rdfs:Class>

<rdfs:Class rdf:ID=”ProtocolAction”>
   <rdfs:comments> This class describes the protocol action within a service co-operation protocol step
   </rdfs:comment>
   <rds:subClassOf rdf:resource = ”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource”/>
</rdfs:Class>

<rdfs:Property ID=”protocolName”>
   <rdfs:domain rdf:resource=”#ServiceProtocol”/>
   <rdfs:range rdf:resource=”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal”/>
</rdfs:Property>
```

```
<rdfs:Property ID=”protocolRole”>
   <rdfs:domain rdf:resource=”#ServiceProtocol”/>
   <rdfs:range rdf:resource=”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal”/>
</rdfs:Property>

<rdfs:Property ID=”protocolType”>
   <rdfs:domain rdf:resource=”#ServiceProtocol”/>
   <rdfs:range rdf:resource=”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal”/>
</rdfs:Property>

<rdfs:Property ID=”negotiatedService”>
   <rdfs:domain rdf:resource=”#ServiceProtocol”/>
   <rdfs:range rdf:resource=”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal”/>
</rdfs:Property>

<rdfs:Property ID=”actions”>
   <rdfs:domain rdf:resource=”#ServiceProtocol”/>
   <rdfs:range rdf:resource=”#ProtocolActions”/>
</rdfs:Property>

<rdfs:Property ID=”precondition”>
   <rdfs:domain rdf:resource=”#ProtocolActions”/>
   <rdfs:range rdf:resource=”http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource”/>
</rdfs:Property>

<rdfs:Property ID=”action”>
   <rdfs:domain rdf:resource=”#ProtocolActions”/>
   <rdfs:range rdf:resource=”#ProtocolAction”/>
</rdfs:Property>

<rdfs:Property ID="messageAction">
   <rdfs:domain rdf:resource="#ProtocolAction"/>
   <rdfs:range rdf:resource=="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
</rdfs:Property>

<rdfs:Property ID="speechAct">
   <rdfs:domain rdf:resource="#ProtocolAction"/>
   <rdfs:range rdf:resource=="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
</rdfs:Property>

<rdfs:Property ID="service">
   <rdfs:domain rdf:resource="#ProtocolAction"/>
   <rdfs:range rdf:resource=="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
</rdfs:Property>

<rdfs:Property ID="reason">
   <rdfs:domain rdf:resource="#ProtocolAction"/>
   <rdfs:range rdf:resource=="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
</rdfs:Property>

<rdfs:Property ID=”actionChoice”>
   <rdfs:domain rdf:resource=”#ProtocolActions”/>
   <rdfs:range rdf:resource=”#ProtocolActions”/>
</rdfs:Property>
</rdf:RDF>
```

Within this definition, and comply to the philosophy of this thesis, the protocol definition

mainly controls the sequence of speech act messages exchanged between the agents for the purpose of agent co-operation interoperability. It does not say how the agent's internal intelligence is implemented or deployed in agent's decision for issuing (or refusing to issue) the messages. More specifically, it does not address the problem of how to represent the *legal* knowledge needed in a negotiation process and how to reason about this knowledge. More detailed discussion on these legal intelligence issue can be found in [37]. In an open environment like the VPN application environment, we expect each agent to have different mechanisms for implementing the intelligence and autonomy in legal reasonings.

Within this schema-based ontology, the resource of the class *SeviceProtocol* defines a co-operation/negotiation protocol for a specific service context. The first three properties of this class identify some general characteristics of the protocol, i.e.

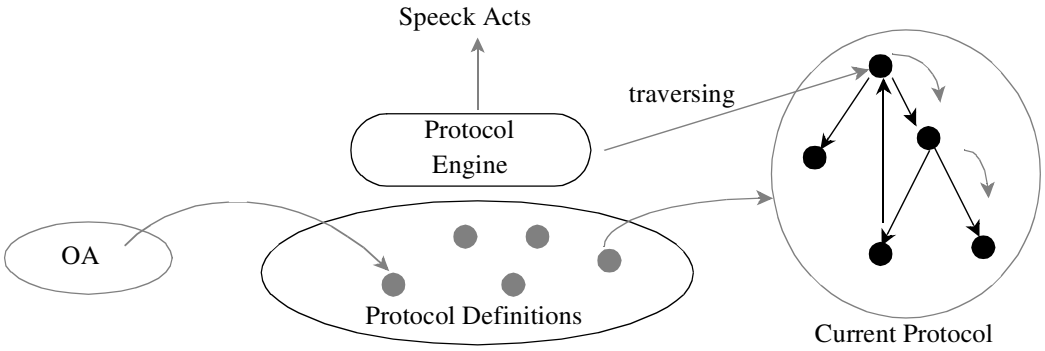
- *ProtocolName* gives the protocol to be defined a unique name, which can be used as a reference to the protocol,
- *ProtocolType* gives some general information about its deployment, e.g. whether it should be used in parallel negotiation with multiple agents, or only used in sequential negotiations,
- *NegotiatedService* identifies the telecommunication service (or service group) if the protocol is designed only for that service.
- *ProtocolRole,* which can have the value of *initiatingAgent* or *respondingAgent*, is to identify whether the protocol ontology is defined for the initiator of the protocol session or the responding agent.

For each protocol, *two* RDF ontology documents will be provided, one for the initiator of the negotiation process while the other is used by the responding agent. This separation of the protocol definition into two complementary documents is needed not only to simplify the interpretation of the protocols, but also because the protocol definitions for the two roles can have in many cases different strategies (coded in the preconditions and the speech act sequences) and are developed/provided by different parties with different interests. This role of the protocol definitions is represented by the *ProtocolRole* property.

The core component in a *ServiceProtocol* resource is the *actions* property, which recursively defines the *Directed Graph* for the protocol action sequences. This Directed Graph is implemented via the *ProtocolActions* resource class. An object of this class identifies, via the *precondition* and *action* properties the candidate protocol action to be carried out at that step.

Furthermore it contains a number of *actionChoice*, each is derived from the class *ProtocolActions* and specifies one possible follow up action for the current protocol step.

Figure 72: Protocol Execution



To enforce a selected protocol during its co-operation with other agents, the Protocol Engine, as depicted in figure 72, will typically decide on its activities by traversing through the Directed Graph for the protocol actions. In this context, the Protocol Engine maintains a store of protocol definitions that are relevant for its operations. Whenever it needs a new protocol, it can search and download the protocol definition from the OA. One protocol will be selected from this store of protocol definition as the current protocol for guiding the operations of the Protocol Engine and the agent.

To start the traversing, the root node of the current protocol definition will be used as the first candidate for the protocol action. At each candidate node, the Protocol Engine will check the first validity of the precondition. If this is true, the associated action will be invoked. If the invocation is successful, the list of children nodes will be regarded as the candidates for the next protocol step, the Protocol Engine continues with the traversing of the graph until a termination node is reached.

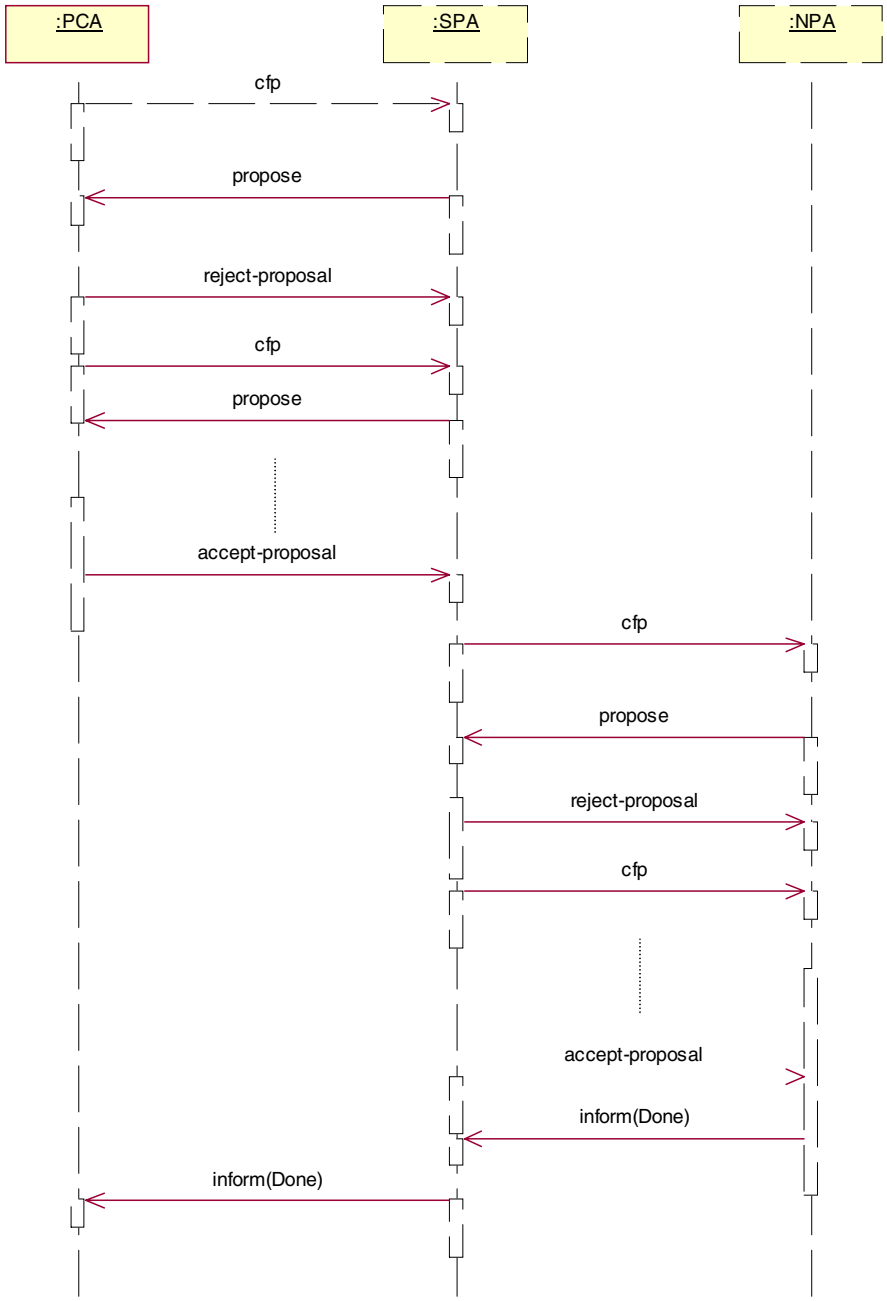
Otherwise if either the precondition is false or the invocation of the current candidate action failed, the Protocol Engine will discard the current candidate and try the next one on the list. If all the candidates failed, the protocol execution also fails.

The major category of protocol actions used in protocol definitions are speech acts, i.e. the sending or receiving of speech act messages. In most case the pre-conditions can be omitted from *ProtocolActions*, which means the semantics of the speech acts will be used as the precondition for the action.

A typical negotiations scenario for the VPN service can be depicted in figure 73. The PCA/SPA and SPA/NPA negotiations are implemented via the FIPA-iterated-contract-net protocol. After the successful reservation of the connections, the user can start their video applica-

tions to start the conference.

Figure 73: VPN Negotiations



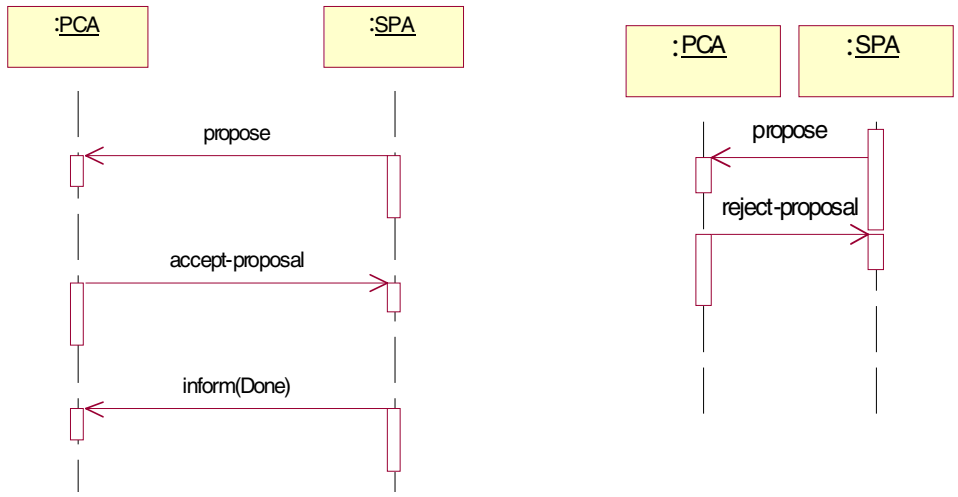
Now suppose that during the conference the SPA decide to modify the running session. E.g.

because a user has complained about the quality of the video and there is still free network capacity available.

Because the SPA has co-operated with the PCA for some time, it has some knowledge about the preference of the PCA and can make a proposal, e.g. higher bandwidth but slightly higher price, that is likely to be accepted by the PCA. Moreover, a rejection of this proposal can be considered by the SPA as a rejection to modify the service. This kind of simplification can significant reduce the number of interactions needed for achieving an agreement via the cfp/ propose iterations the iterated-contract-net protocol. The new protocol is called s*imple-contract-net* protocol, with the typical interactions described in figure 74.

To further elucidate on the idea of dynamic protocol adaptation, we first take a closer look at this negotiation within the VPN scenario.

Figure 74: Simple-Contract-Net Protocol



The protocol definition for the SPA as initiator in this context can be specified by the following RDF document:

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#"
    xmlns:pro="http://mimosa.fokus.gmd.de/projects/example/protocols#"
    xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-template#">

<pro:ServiceProtocol>
    <pro:protocolName> simple-contract-net </pro:protocolName>
     <pro:protocolType>sequential</pro:protocolType>
    <pro:actions rdf:resource= "start" />
    <pro:negotiatedService>vpn</pro:negotiatedService>
    <pro:protocolRole> initiator </pro:protocolRole>
</pro:ServiceProtocol>
```

```
<pro:ProtocolActions rdf:ID="start">
    <pro:action rdf:resource = "action-propose"/>
    <pro:actionChoice rdf:resource="actions-not-understood" />
    <pro:actionChoice rdf:resource="actions-accept-proposal" />
    <pro:actionChoice rdf:resource="actions-reject-proposal"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-propose">
    <pro:messageAction>send</pro:messageAction>
    <pro:speechAct> propose  </pro:speechAct>
    <pro:service> ?Service </pro:service>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-not-understood">
    <pro:action rdf:resource = "action-not-understood"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-not-understood">
    <pro:messageAction>receive </pro:messageAction>
    <pro:speechAct> not-understood  </pro:speechAct>
    <pro:reason> ?Reason </pro:reason>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-accept-proposal">
    <pro:action rdf:resource = "action-accept-proposal"/>
    <pro:actionChoice rdf:resource="actions-failure" />
    <pro:actionChoice rdf:resource="actions-inform" />
</pro:ProtocolActions>

<pro:ProtocolActions rdf:ID="actions-reject-proposal">
  <pro:action rdf:resource = "action-reject-proposal"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-accept-proposal">
    <pro:messageAction>receive </pro:messageAction>
    <pro:speechAct> accept-proposal  </pro:speechAct>
    <pro:service> ?Service </pro:service>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-inform">
  <pro:action rdf:resource = "action-inform"/>
</pro:ProtocolActions>

<pro:ProtocolActions rdf:ID="actions-failure">
  <pro:action rdf:resource = "action-failure"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-inform">
    <pro:messageAction>receive </pro:messageAction>
    <pro:speechAct> inform  </pro:speechAct>
    <pro:service> Done </pro:service>
</pro:protocolAction>

<pro:protocolAction rdf:ID="action-failure">
    <pro:messageAction> receive </pro:messageAction>
    <pro:speechAct> failure  </pro:speechAct>
```

```
        <pro:service> ?Service </pro:service>
        <pro:reason> ?Reason </pro:reason>
    </pro:protocolAction>

    <pro:ProtocolActions rdf:ID="action-reject-proposal">
        <pro:action rdf:resource = "action-reject-roposal"/>
    </pro:ProtocolActions>

    <pro:protocolAction rdf:ID="action-reject-proposal">
        <pro:messageAction>receive</pro:messageAction>
        <pro:speechAct> reject-proposal  </pro:speechAct>
        <pro:service>?Service </pro:service>
        <pro:reason> ?Reason </pro:reason>
    </pro:protocolAction>
    </RDF>
```

If the SPA has this definition in his protocol knowledge, it can issue the following message to the PCA:

```
    (propose
          :receiver SPA
          :sener PCA
          :content
                "<?xml version="1.0"?>
                 <RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                  xmlns:vc="http://mimosa.fokus.gmd.de/projects/example/video-conference#"
                  xmlns:net="http://mimosa.fokus.gmd.de/projects/example/simple-contract-
net#">

                  <vc:videoConference>
                      <vc:userName> user1</vc:userName>
                      <vc:userName> user2 </vc:userName>
                      <vc:serviceType> video</vc:serviceType>
                      <vc:startDateTime>19990324T150000000Z
                      </vc:startDateTime>
                      <vc:endDateTime>19990324T170000000Z
                      </vc:endDateTime>
                      <vc:bandwidth> 120 </vc:bandwidth>
                      <vc:cost> 110 </vc:cost>
                  </vc:videoConference>
                  </RDF>"
          :reply-with id1
          :language vpn
          :ontology videoConference
          :protocol simple-contract-net
    )
```

Once the PCA receives this message it will notice the protocol name in the :protocol paramter. If this protocol is not locally available, it will download the following definition for the responding agent from an appropriate OA.

```
    <?xml version="1.0"?>
    <RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#"
        xmlns:pro="http://mimosa.fokus.gmd.de/projects/example/protocols#"
        xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-template#">
```

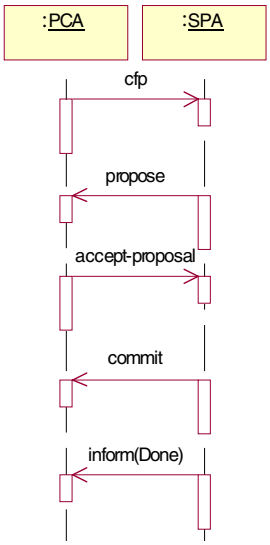
```
<pro:ServiceProtocol>
    <pro:protocolName> simple-contract-net </pro:protocolName>
     <pro:protocolType>sequential</pro:protocolType>
    <pro:actions rdf:resource= "start" />
    <pro:negotiatedService>vpn</pro:negotiatedService>
    <pro:protocolRole> respondingAgent </pro:protocolRole>
</pro:ServiceProtocol>

<pro:ProtocolActions rdf:ID="start">
    <pro:action rdf:resource = "action-propose"/>
    <pro:actionChoice rdf:resource="actions-not-understood" />
    <pro:actionChoice rdf:resource="actions-accept-proposal" />
    <pro:actionChoice rdf:resource="actions-reject-proposal"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-propose">
    <pro:messageAction>receive</pro:messageAction>
    <pro:speechAct> propose  </pro:speechAct>
    <pro:service> ?Service </pro:service>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-not-understood">
    <pro:action rdf:resource = "action-not-understood"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-not-understood">
    <pro:messageAction>send  </pro:messageAction>
    <pro:speechAct> not-understood  </pro:speechAct>
    <pro:reason> ?Reason </pro:reason>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-accept-proposal">
    <pro:action rdf:resource = "action-accept-proposal"/>
    <pro:actionChoice rdf:resource="actions-failure" />
    <pro:actionChoice rdf:resource="actions-inform" />
</pro:ProtocolActions>

<pro:ProtocolActions rdf:ID="actions-reject-proposal">
  <pro:action rdf:resource = "action-reject-proposal"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-accept-proposal">
    <pro:messageAction>send </pro:messageAction>
    <pro:speechAct> accept-proposal  </pro:speechAct>
    <pro:service> ?Service </pro:service>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="actions-inform">
  <pro:action rdf:resource = "action-inform"/>
</pro:ProtocolActions>

<pro:ProtocolActions rdf:ID="actions-failure">
  <pro:action rdf:resource = "action-failure"/>
</pro:ProtocolActions>
```

```
<pro:protocolAction rdf:ID="action-inform">
    <pro:messageAction>send </pro:messageAction>
    <pro:speechAct> inform  </pro:speechAct>
    <pro:service> Done </pro:service>
</pro:protocolAction>

<pro:protocolAction rdf:ID="action-failure">
    <pro:messageAction> send </pro:messageAction>
    <pro:speechAct> failure  </pro:speechAct>
    <pro:service> ?Service </pro:service>
    <pro:reason> ?Reason </pro:reason>
</pro:protocolAction>

<pro:ProtocolActions rdf:ID="action-reject-proposal">
    <pro:action rdf:resource = "action-reject-roposal"/>
</pro:ProtocolActions>

<pro:protocolAction rdf:ID="action-reject-proposal">
    <pro:messageAction>send </pro:messageAction>
    <pro:speechAct> reject-proposal  </pro:speechAct>
    <pro:service>?Service </pro:service>
    <pro:reason> ?Reason </pro:reason>
</pro:protocolAction>
</RDF>
```

By interpreting this definition, the PCA is now in the position to support the interactions in figure 74.

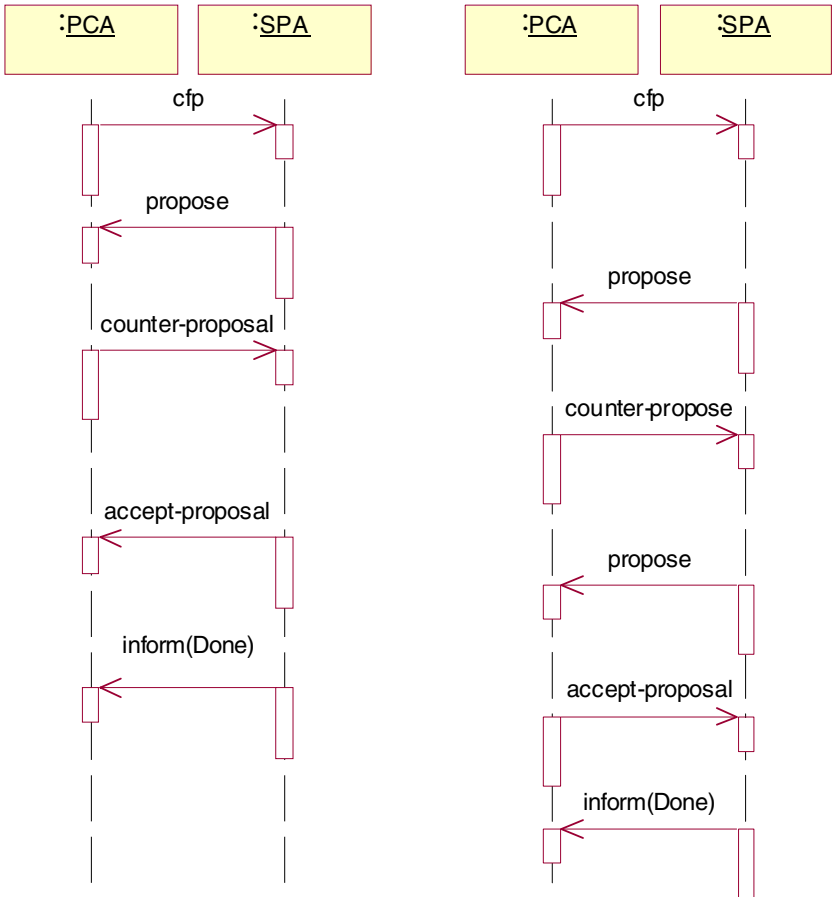
In fact depending on the requirements and properties of specific negotiation contexts, a variety of negotiation protocols can be envisaged for the contract net. Each of these protocols can help to solve some of the problems in the dedicated application contexts. There are also cases where we need some new speech act types (performatives) other than the basic ones listed in section 4.3.

Figure 75: Commit in Contract-Net Protocol



E.g. after receiving the *accept-proposal* message from the PCA, we can require the SPA to first response with a *commit* message (as defined in Section 4.6.3) to confirm that it commits to offer the service. This extra step allows further freedom in the SPA's resource reservation scheme. A SPA can make a proposal for a service action but does not necessary commit to it. I.e. it does not commit to any punishment in case of failure to provide the service even if the service is accepted. Such a freedom is important if the service provider like SPA is negotiating in parallel with different potential user groups, and can commit to some resource allocation if it is sure that the client will use the resource. This scenario can be depicted in figure 75.

Figure 76: Contract-Net with Counter Proposal



Similarly, after receiving the *propose* message, instead of refusing the proposal or to wait for all proposals/refusals and then make a new *cfp*, an initiator of the negotiation (PCA or SPA) can directly issue a *counter-propose* message which will propose a new service, which the PCA (or SPA) believes as acceptable for the responding agent (SPA or NPA). This possibil-

ity can accelerate the negotiation process in case the two negotiation partners are well acquainted and co-operative, because both side can actively try to approach an agreement via new service proposals and counter proposals. Some typical scenarios with this protocol can be depicted in figure 76.

These examples also show the necessity to support the dynamic and flexible extensions of the set of speech acts deployed in the agent negotiations.

5.5.2. Service Programming and Customization

Service programming can be considered as the basic form of service interactions based on agent technology. It allows the user/customer to request and to obtain the services or service features that tailored exactly to the needs of the user/customer.

One example is that a customer prefer to simplify the interfaces to the VPN service for its users, so that the users, in order to request a service, just have to select a service class and provide the addresses. The detailed information about the service, e.g. reliability, delay/ delay-variation and accounting policy, are all defined in the classes.

This service programming can be realized by sending the ontology definition to the associated SPA, or by asking the SPA to download this ontology from the OA.

```
<?xml version="1.0"?>
<RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:vc="http://mimosa.fokus.gmd.de/projects/example/video-conference#"
xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#">

<rdfs:Class rdf:ID="ServiceClass">
 <rdfs:comment>  This class represented dynamically specified VPN service classes
     </rdfs:comment>
<rdfs:subClassOf  rdf:resource = "vc:videoConference">
</rdfs:Class>

<rdfs:Property ID="classID">
  <rdfs:domain rdf:resource="#ServivceClass"/>
  <rdfs:range rdf:resource=="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Lit-
eral"/>
</rdfs:Property>

<schema:Equivalent rdf:ID="serviceClass1">
  <schema:head>
   <ServiceClass>
      <classID>  1 <classID>
      <vc:userList> ?Users </vc:userList>
   <ServiceClass>
  </schema:head>
<schema:body>
  <schema:body_li>
         <vc:videoConference>
```

```
                <vc:userList> ?Users </vc:userList>
                <vc:serviceType> video</vc:serviceType>
                <vc:bandwidth> 10 </vc:bandwidth>
                <vc:cost> 40 </vc:cost>
            </vc:videoConference>
  </schema:body_li>
</schema:body>
</schema:Equivalent>

<schema:Equivalent rdf:ID="serviceClass2">
  <schema:head>
   <ServiceClass>
      <classID>  2 <classID>
      <vc:userList> ?Users </vc:userList>
   </ServiceClass>
  </schema:head>
<schema:body>
  <schema:body_li>
          <vc:videoConference>
              <vc:userList> ?Users </vc:userList>
              <vc:serviceType> video</vc:serviceType>
              <vc:bandwidth> 100 </vc:bandwidth>
              <vc:cost> 110 </vc:cost>
          </vc:videoConference>
  </schema:body_li>
</schema:body>
</schema:Equivalent>
</RDF>
```

Basically, this ontology defines two VPN service classes. Class 1 has always bandwidth 10 and cost 40 (units), while Class 2 has bandwidth 100, cost 110.

After the SPA downloaded this RDF document, a user PCA can request with the following simple message content the provisioning of VPN service:

```
"<?xml version="1.0"?>
  <RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:vc="http://mimosa.fokus.gmd.de/projects/example/video-conference#"
        xmlns:sc="http://mimosa.fokus.gmd.de/projects/example/service-class#">

        <sc:ServiceClass>
                <vc:userList>
                    <vc:userName-li> user1 </vc:userName-li>
                    <vc:userName-li> user2 </vc:userName-li>
                </vc:userList>
                <sc:classID> 1 </sc:classID>
        </sc:ServiceClass>
  </RDF>"
```

Via this kind user-tailored, dynamic service adaptation we can significantly reduce the complexity of the users's access interface to the service.

5.5.3. Publication of New Service Features

The provider of the services can dynamically extend or change the service features following the changes in its local environment, e.g. the deployment of new network technology. One example is that the NPA moves to ATM technology and supports some new QoS parameter which was not supported before.

In that case the NPA has to inform the SPAs about this new parameter and its meanings (i.e. its usage) so that the SPAs can utilize this feature in their applications, i.e. the NPA has to advertise the new service features to the SPAs. Such advanced service advertisement is usually realized by defining the new concepts by relating them to the existing ontologies (e.g. initial ontology) supported by the SPAs.

To do this, the NPA can simply inform the SPA that it supports a new ontology, e.g. the ontology *connection.atm-vp-connection* as defined in section 4.5, via the following message

```
(inform :sender NPA
        :receiver SPA
        :content
          "<?xmlversion="1.0"?>
           <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
       xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#"
        xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-template#">

          <at:Done>
            <schema:Action>
              <schema:agentID> NPA </schema:agentID>
              <schema:actionType>ontologySupport</schema:actionType>
              <schema:service> connection.atm-vp-connection </schema:service>
            </schema:Action>
          </at:Done>
          </rdf:RDF>"

        . . . )
```

Upon which the SPA will retract the identified ontology from the OA and integrate it into its local knowledge base. Future service instances can then utilize the features of ATM QoS in the provisioning of the connections.

Similarly, the SPA can tell via the following message the PCA that it now supports the user view based on ATM technology, i.e. the ontology *connection.atm-vp-connection.user-view-atm*, and the user PCA can extend its serice requests by considering the new features.

```
(inform :sender SPA
        :receiver PCA
        :content
          "<?xmlversion="1.0"?>
           <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
```

```
            xmlns:schema="http://mimosa.fokus.gmd.de/projects/example/management-knowledge#"
             xmlns:at="http://mimosa.fokus.gmd.de/projects/example/service-agent-template#">

               <at:Done>
                 <schema:Action>
                   <schema:agentID> SPA </schema:agentID>
                   <schema:actionType> ontologySupport </schema:actionType>
                   <schema:service>connection.atm-vp-connection.user-view-atm</schema:ser-
vice>
                 </schema:Action>
               </at:Done>
               </rdf:RDF>"

             . . . )
```

5.5.4. Robust and Reliable Service Co-operations in the Dynamic and Distributed Environment

Robustness in this case means, among others, the ability to cope with new or unexpected situations during the interactions. Within the agent frameworks, agents can exchange behavior related information to dynamically and intelligently enable the agents to process new abnormal situations.

E.g. a SPA can tell (via an inform message) a PCA how to deal with the situation in which it receives no proposal upon Call-For-Proposal (cfp) from the PCA via the following rule:

```
<schema:Implies rdf:ID="no-proposal">
  <schema:head>
     <at:received>
      <at:ACLMessage>
        <at:performative> refuse </at:performative>
        <at:sender> ?I </at:sender>
        <at:receiver> ?J </at:receiver>
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?Service </at:content>
     </at:ACLMessage>
    </at:received>
  </schema:head>
  <schema:body>
    <schema:body_li>
     <at:sent>
      <at:ACLMessage>
        <at:performative> cfp </at:performative>
        <at:sender> ?J </at:sender>
        <at:receiver> ?I </at:receiver>
        <at:concersation-id> ?L </at:concersation-id>
        <at:content> ?Service </at:content>
     </at:ACLMessage>
    </at:sent>
  </schema:body_li>
  <schema:body_li>
     <pd:Wait>
```

```
            <pd:length> 2000 </pd:length>
          </pd:Wait>
        </schema:body_li>
      </schema:Implies>
```

which means after sending the *cfp* and waiting for 2 seconds, a reception of message for rejecting the *cfp* is assumed if no proposal received.

Another key technology for enhancing the reliability and robustness of telecommunications services is the deployment of MA technology within the IA framework in the service negotiations.

Suppose that a PCA is installed in a laptop, which can be from time to time detached from the network. Moreover, when the laptop is again plugged into the network, the user can already travel to another country with this laptop, so the laptop will find itself in a totally new and foreign environment.

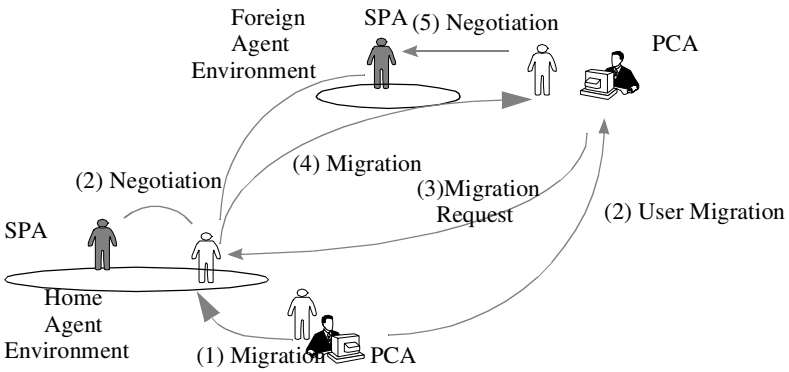
This situation can cause problems in some cases. E.g. if the PCA on the laptop plays the IPCA role, and the negotiation stage lasts for a relatively longer period of time, frequently detaching the laptop during this period can cause confusions among the negotiation partners. On the other hand, if an IPCA wants to negotiate with this RPCA and this PCA's host was detached, the IPCA will have difficulties in find out whether the meeting can be arranged.

The solution in this context is to implement the PCA as MA, and to allow this MA (before the host is detached from the network) to migrate to a static environment, e.g. to the agent environment (AP) of an trusted SPA. This mobile PCA will stay there, carry out any necessary negotiations with the SPA or other PCAs.

Once the host is plugged into the network, the user can send a request to this mobile PCA and ask it to move to the new host environment. In that environment, the PCA will register onto the local platform, it will first use the negotiation results to continue the current VPN negotiation or service session among the users.

For the current and future service sessions, the PCA will find the accessible SPAs from the local DF (the set of SPAs can be different from the PCA's original home environment), and start negotiation with these foreign SPAs using the initial ontologies and protocols. The whole scenario can be depicted in figure 77. Afterwards, via service programming and service publications, the PCA can gradually improve its performance in the new environment.

Figure 77: Mobile Host and Agent



5.6. Summary

In order prove the agent framework and architecture presented in this thesis, this chapter presents the application of the agent solution as defined in this thesis to the management of VPN service within the dynamic, heterogeneous and distributed telecommunications environment. This focus of this chapter is to show, via some example scenarios and analyses, the enhancement of services interoperability resulted from the deployment of agent technology, especially in comparison to traditional service management technologies.

As a conclusion drew from the presentation and analyses in this chapter, agent-based interoperability, and the associated possibility for both functional mobility and agent mobility can help to enhance the dynamic adaptability/programmability, customer-orientation and reliability/robustness of the telecommunications services. Such enhancement can be realized via, among others, the dynamic protocol or service customization and publication, via exchanging knowledge for exception processing and via integrated MA/IA technology for dealing with host/user mobility.

CHAPTER 6

Conclusion and Future Work

Different from the traditional RPC and DOT-based paradigms for distributed system designs, which are based on syntactical interoperability and therefore rely on relatively static co-operations interfaces among distributed applications, agent technology can be based on knowledge-based interoperability among autonomous applications or application environments.

This enhanced interoperability, the resulted capability for dynamically exchanging knowledge or behavior information among agent-based applications, and also the capability of dynamic service adaptation or programming, make agent technology a better solution for co-operating telecommunications applications within the dynamic, heterogeneous, distributed or even nomadic environment.

Based on these considerations, this thesis has presented an agent-based solution for knowledge-based co-operation among autonomous telecommunications applications. Such a solution is based on a framework and an architecture that encompass a number of technologies. The core technologies in this context include

- integration of OMG MASIF and FIPA agent paradigms and technologies to meet the requirements in the heterogeneous application contexts,

- basic speech act types (performatives) and co-operation protocols that suit the needs of telecommunications service provisioning and deployment,

- definition of an agent communication content language, which is capable carrying knowledge-based rule-based knowledge between agents for the dynamic adaptation of agent intelligence and service functionalities, and which is also based a popular XML-based resource model (RDF) in order to ease the Web-based presentation and processing, or the integration of Web-based applications.

- a knowledge-based and context-oriented ontology framework which supports the accumulative acquisition and easy deployment of reusable domain specific knowledge for agent co-operation,

- definition of a service agent template, which characterizes the basic rational behaviors of an agent in the service management environment and enables the initial agent interoperability and the dynamic definition of speech acts or co-operation protocols.

The agent-based solution and its component technologies are applied to the provisioning and management of dynamic VPN services, and have showed some advantageous over conventional technology, especially in enabling dynamic evolution of application co-operation protocols and functionality, in dynamic customization of user-oriented services and in achieving robustness and reliability via agent or functional mobility.

The analyses and solution presented in this thesis offer some bases and guidelines that help the further development of agent technology in telecommunications applications. Among other, the following issues will require more attention and efforts from the agent community:

- Standardization of Agent Technology

As mentioned in the thesis, the OMG MASIF standard does not yet offer sufficient support for guaranteeing agent mobility across heterogeneous implementations of the MASIF standard, while the FIPA standards have not yet enough results concerning the agent communication contents to ensure interoperability among heterogeneous applications. A lot of efforts are still necessary in this context in order to achieve the real goals of these standardization efforts, i.e. the interoperability among heterogeneous agents and agent platforms.

- knowledge-based ontologies for Agent-based Telecommunications Applications

Ontologies can promote the application of agent technology only if they are shared by a larger group of agent applications. In another word, ontologies need to be standardized. An extensive library of standardized ontology definitions, especially those that can be easily deploy-

ment in telecommunications applications, will play a significant role in paving the way for the wide application of agent technology. Significant efforts are still necessary to build up such a library.

– Applications of the knowledge-based Agent Technology

More efforts will be necessary for identifying the major application areas of the agent technology based on knowledge-based interoperability. Among others, with the high level of autonomy, dynamic nature and distribution, and with their requirement for enhanced support for intelligent and flexible negotiations, e-commerce and business management ([114]) are expected to be among the best candidates for the deployment of agent technology and the deployment of the solution presented in this thesis.

References

[1] Joseph Bates, A. Bryan Loyall, W.Scott Reilly, *Integrating Reactivity, Goals, and Emotion in a Broad Agent*, School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-92-142, May 1992.

[2] Fabio Bellifemine, Giovanni Rimassa and Agostino Poggi, *JADE - A FIPA-compliant Agent Framework*, in Proceedings of the Conference on Implementing Agents: Standard, Architectures, Testbeds, Programming Tools, London, March 1999.

[3] Claire Beyssade, Patrice Enjalbert, Claire Lefevre, *Co-operating Logical Agents*, in Proceedings of the IJCAI'95 Workshop, Montreal, Lecture Notes in Artificial Intelligence 1037, August 1995.

[4] J. Bradshaw ed., *Software Agent*, AAAI/The MIT Press, 1997.

[5] Vinay K. Chaudhri et al., *Open Knowledge Base Connectivity 2.0.3*, April 1998.

[6] Phillip R. Cohen, *The Role of Speech Acts in Natural Language Understanding*, August 1986.

[7] Philip R.Cohen, Hector J. Levesque, *Commnucative Actions for Artificial Agents*, In [4].

[8] S. Covaci, T. Zhang, I. Busse, *Mobile Intelligent Agents for the Management of Information Infrastructure*, Proceedings of the 31th Hawaii International Conference on System Sciences, January 1998.

[9] Communications of the ACM Journal, *Intelligent Agent*, Vol.37, No. 7, July 1994.

[10] Steve Corley, *Agents Technology and P712 Goals*, Proceedings of the Eurescom Workshop pn Distributed Object Technologies and Middleware, Heidelberg, November 1997.

[11] European Commission, *Third Call General Information Document*, 1997.

[12] L. Daigle, P. Deutsch, *Uniform Resource Agents (URAs)*, IETF RFC 2016, October 1996.

[13] Epistemics Inc., *An Overview of EPILOG 1.0 for Lisp*, 1994

[14] Clark Elliott, *Hunting for the Holy Grail With "Emontional Intelligent" Virtual Actors*, SIGART Bulletin, Summer 1998.

[15] Clark Elliott, *Research Problems in the Use of a Shallow Artificial Intelligence Model of Personality and Emotion*, In Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, 1994.

[16] Clark Elliott, I Picked up Catapia and Other Stories: *A Multimodal Approache to Expressivity for "Emotional Intelligent" Agents*, In Proceedings of the First International Conference on Autonomous Agents, 1997.

[17] DAI-Laboratory, TU Berlin, *JIAC Tutorial*, October 1998.

[18] ETSI, *SRC6 Final Report on European Information Infrastructure*, June 1995.

[19] N.J.Davies, R. Weeks and MC Revett, *Information Agents for World Wide Web*, in [67].

[20] A. Farquhar, R. Fikes, J. Rice, *The Ontolingua Server: A Tool for Collaborative Ontology Construction*, Technical Report KSL 96/26, Stanford University, Knowledge Systems Laboratory, 1996.

[21] Tim Finin, Jay Weber, *Specification of the KQML Agent-Communication Language - The DARPA Knowledge Sharing Initiative External Interfaces Working Group*, February 9, 1994

[22] FIPA: *Foundation for Intelligent Physical Agents*, http://drogo.cselt.stet.it/fipa/.

[23] FIPA, *Agent Management*, FIPA 97 Specification Part 1, November 1997.

[24] FIPA, *Agent Communication Language*, FIPA 97 Specification Part 2, November 1997.

[25] FIPA, *Agent/Software Integration*, FIPA 97 Specification Part 3, November 1997.

[26] FIPA, *Personal Travel Assistance*, FIPA 97 Specification Part 4, November 1997.

[27] FIPA, *Personal Assistant*, FIPA 97 Specification Part 5, November 1997.

[28] FIPA, *Audio/Video Entertainment & Broadcasting*, FIPA 97 Specification Part 6, November 1997.

[29] FIPA, *Network Management and Provisioning*, FIPA 97 Specification Part 7, November 1997.

[30] FIPA, *Agent Management Support for Mobility*, FIPA 98 Specification Part 11, June

1998.

[31] FIPA, *Ontology Service*, FIPA 98 Specification Part 12, October 1998.

[32] FIPA, *Human Agent Interaction*, FIPA 98 Specification Part 13, October 1998.

[33] FIPA, FIPA Content Language Library, FIPA 99 Specification Part 18, Januaray 2000.

[34] GMD FOKUS, IBM, Crystaliz, General Magic, The Open Group, *Mobile Agent System Interoperability Facilities Specification*, Joint OMG Submission, November 1997.

[35] E. J. Friedman-Hill, *Jess, The Java Expert System Schell*, Sandia National Laboratories

[36] H. Robert Frost, *Documentation for Java(tm) Agent Template*, Version 0.3, 1996.

[37] Anne von der Lieth Gardner, *An Artificial Intelligence Approach to Legal Reasoning*, The MIT Press, 1987.

[38] Thomas R. Gruber, *Ontolingua: A Mechanism to Support Portable Ontologies*, June 1992.

[39] Thomas R. Gruber, *A Translation Approach to Portable Ontology Specifications*, Knowledge System Laboratory, Stanford University, Technical Report KSL 92-71, April 1993.

[40] Stan Franklin, Art Graesser, *Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*, Intelligent Agent III, LNAI 1193, 1997.

[41] General Magic, *Odyssey Information*, http://www.generalmagic.com/technology/odyssey.html.

[42] Michael R. Genesereth, Richard E. Fikes, *Knowledge Intercahnage Format - Version 3.0 - Reference Manual*, Logic Group, Stnaford University, Report Logic-92-1, June 1992.

[43] *The Global Information Infrastructure: Agenda for Cooperation*, http://www.iitf.nist.gov/documents/docs/gii/giiagend.html.

[44] Shaw Green, Fergal Somers, et al., *Software Agents: A Review*, May 27, 1997

[45] R.V. Guha, D.B. Lenat, *Enabling Agents to Work Together*, in [9].

[46] H.W. Güsgen, *CONSAT:A System for Constraint Satisfaction*, Pitman Publiching London, 1989.

[47] D. Hogrefe, *Estelle, Lotos and SDL, Springer-Verlag*, 1989.

[48] IBM, *IBM Agent Building Environment Developer's Toolkit*, http://cs.chungnam.ac.kr/~tgkang/abe/docs.

[49] IETF, *Structure and Identification of Management Information for TCP/IP-based Internet*s, RFC 1155, May 1990

[50] IKV++, *Grasshopper*, http://www.ikv.de/products/grasshopper.html, September 1998.

[51] ISO/IEC, *ISO Committee Draft on Basic Reference Model of Open Distributed Processing - Part 1~ Part 3*, JTC1/SC21 N7524, December 1992

[52] ISO, *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*,

May 1987.

[53] ITU-T, Recommendation M.3010, *Principles for a Telecommunications Management Network*, March 1995.

[54] ITU-T, Recommendation X.500, *Information technology - Open System Interconnection - The directory: Overview of concepts, models, and services*, November 1995

[55] ITU-T, Recommendation X.722, *Information technology - Open System Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects,* January 1992.

[56] H. Kleine Büning, S. Schmitgen, *Prolog*, B.G.Teubner Stuttgart, 1988.

[57] Paul Kußmaul, *Sprechakththeorie*, 1980

[58] Yannis Labrou, Tim Finin, *A Proposal for a New KQML Specification*, CSEE/University of Maryland Baltimore, TR-CS-97-03, February 1997.

[59] Steve Laufmann, *Agents as a Practical Matter: Tools and Techniques*, Proceedings of the Conference on Implementing Agents: Standards, Architecture, Testbeds, Programming Tools, London, April 1999.

[60] M.Lenz, B. Bartsch-Spörl, H-D. Burkhard, S. Wess (Eds.), Case-Based Reasoning Technology- From Foundations to Applications, LNAI 1400, 1999.

[61] B. Liberman, F. Griffel, M. Merz, W. Lamersdorf, *Java-Based Mobile Agents - How to Migrate, Persit, and Interact on Electronic Service Market*, in Proceedings of the First International Workshop, MA'97, Berlin, April 1997, LNCS 1219.

[62] Thomas Magedanz, Radu Popescu-Zeletin, *Intelligent Networks - Basic Technology, Standards and Evolution*, Thomson Computer Press, 1996.

[63] JP. Morgenthal, *XML Agents*, July 1998.

[64] Robert Neches et. al., *Enabling Technology for Knowledge Sharing*, AI Magazine, 12(3), 1991.

[65] NIST, *United States National Information Infrastructure Virtual Library,* http://nii.nist.gov/nii.html, 14 October 1996.

[66] Network Management Forum, *A Service Management Business Process Model*, Issue 1.0, 1995.

[67] Hyacinth S. Nwana, Nader Azarmi (Eds.), *Software Agents and Soft Computing*, Lecture Notes in Artifiial Intelligence 1198, 1997

[68] ObjectSpace, *Voyager - Core Technology User Guide*, Version 2.0 Beta 1, 1997.

[69] P.D. O'Brien, M.E. Wiegand, *Agents of Change in Business Process Management*, in [67].

[70] OMG, *The Common Object Request Broker: Architecture and Specification,* Draft 29, December 1993.

[71] OMG, *Internet Inter-ORB Protocol (IIOP) : Common Object Request Broker Architec-*

ture, Version 2.

[72] OMG, *OMG IDL Syntax and Semantics*, May 1997

[73] OMG BODTF-REF 2 Submission, *Workflow Management Facility*, OMG Document Number bom/98-03-04.

[74] OMG Agent Working Group, *Agent Technology Green Paper*, OMG Document ec/99-03-11, version 0.72, March 1999.

[75] Mitsuru Oshima, Guenter Karjoth, and Kouichi Ono, *Aglets Specification 1.1 Draft*, http//www.trl.ibm.co.jp/aglets/spec/spec11.htm.

[76] S. J. Russell, P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice-Hall International, 1995.

[77] B. Potter, J. Sinclair, and D. Till, *An Introduction to Formal Specification and Z.* International Series in Computer Science, Prentice Hall, 1991.

[78] Davaid Przewozny, *Declarative Agent Programming*, http://user.cs.tu-berlin.de/~davidi/Uni/hdepp/Dokumentation/jdepp.html.

[79] J. Joachim Quantz, *XML as MARINER Content Language*, ACTS MARINER Internal Deliverable, Spetember 1998.

[80] W.Scott Reilly, Joseph Bates, *Building Emontional Agents*, School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-92-143, May 1992.

[81] Martin Saunders, *End to End Management using a Manager/Agent Architecture*, In Proceedings of the Conference on Implementing Agents: Standard, Architectures, Testbeds, Programming Tools, London, March 1999.

[82] John R. Searle, *Speech Acts: An Essaz in the Philosophy of Languages*. Cambridge University Press, 1969.

[83] John F. Sowa, *Conceptual Structures:Information Processing in Mind and Machine,* Addison-Wesley, Reading, MA, 1984.

[84] Stanford University, *JATLite Description*, http://cdr.stanford.edu/ProcessLink/papers/JATL.htm.

[85] Heather Stark/WfMC, *Understanding Worklow*, Workflow Handbook 1997, 1997.

[86] M. Stefik, Planning with Constraints, Artificial Intelligence, vol. 16, page 111-140, 1981

[87] J. M. Spivey, *The Z Notation: A Reference Manual*, second edition, 1992.

[88] Y. Su, *Ontologiedienst für die Dienstadaptierung*, Diplomarbeit, OKS/TU Berlin, December 1999.

[89] Kiminori Sugauchi, Satoshi Miyazaki, Stefan Covaci, Tianning Zhang, *Efficiency Evaluation of a Mobile Agent Based Network Management System,* In Proceedings of IS&N '99, Lecture Notes in Computer Science 1597, April 1999.

[90] Kiminori Sugauchi, Satoshi Miyazaki, Stefan Covaci, Tianning Zhang, *Flexible Network Management Using Active Network Framework,* In Proceedings of the First In-

ternational Working Conference on Active Networks, IWAN'99, Lecture Notes in Computer Science 1653, July 1999.

[91] Sun Microsystems/JavaSoft, *JNDI: JAva Naming and Directory Interface*, January 29, 1988.

[92] TINA-C, *TINA Object Definition Language Manual*, August 1997

[93] TINA-C, *Service Architecture*, Version 2.0, March 1995.

[94] TINA-C, *Overall Concepts and Principles of TINA*, TINA-C Doc. No. TP_TR_HW.001_1.0_97, 1997.

[95] Toschiba, *Bee-gent Multi Agent Framework*, http://www2.toshiba.co.jp/beegent/index.htm.

[96] UMBC, *Knowledge Sharing Effort*, http://www.cs.umbc.edu/kse.

[97] UMBC, *Representation of KIF & FIPA-ACL in XML*, http://www.csee.umbc.edu/~mjin/xml/, 1999.

[98] *Understanding Open Distributed Processing*, The Open Systems Newsletter, Vol 5, September 1993.

[99] Juan D. Velasquez, *Modeling Emotions and Other Motivations in Synthetic Agents*, 1997.

[100] W3C, *World Wide Web*, http://www.w3c.org/.

[101] W3C, *Extensible Markup Language (XML) 1.0*, W3C Recommendation, February 1998.

[102] W3C, *Extensible Stylesheet Language (XSL) Specification*, W3C Working Draft 21, April 1999.

[103] W3C, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation, February 1999.

[104] W3C, *Resource Description Framework (RDF) Schema Specification,* W3C Propose Recommendation, March 1999.

[105] W3C, *Conceptual Knowledge Markup Langauge*, http://asimov.eecs.wsu.edu/WAVE/Ontologis/CKML.

[106] J.E.White, *Telescript Technology: The _Foundation for the Electronic Marketplace*, Geneal Magic White Paper, 1994.

[107] J.E.White, *Mobile Agents*, in [4].

[108] Workflow Management Coalition (WfMC), *The Workflow Reference Model*, Version 1.1, November 1994, WfMC-TC-1003.

[109] T. Zhang, *A Knowledge-Based Model for Network Service Management,* Proceedings of the First IEEE Symposium on Global Data Networking, Cairo, December 1993.

[110] T. Zhang, S. Covaci, *Intelligent Agents for Network and Service Management*, in Proceedings of IEEE Globecom'96, London, November 1996.

[111] T. Zhang et. al., *Configuration Experiment Description*, EURESCOM P712 - Intelligent and Mobile Agents and their Applicability to Service and Network Management, PIR2.3, March 1998.

[112] T. Zhang et. al., *High Level Specification*, ACTS MISA Project Deliverable 3, September 1996.

[113] Tianning Zhang, Irina Parvan, *GrasshopperACL - FIPA Enhancement of a OMG MASIF Compliant Platform*, Internal Report, GMD FOKUS, April 1999.

[114] T. Zhang et. al., *Case Study Selection*, EURESCOM P815 - Communications Management Process Integration Using Software Agents, PIR3.1, September 1998.

[115] T. Zhang, S. Covaci, *The Semantics of Network Management Information*, in Proceedings of the IEFF INFOCOM'96, San Francisco, March 1996.

[116] T. Zhang, S. Covaci, *Pan-European ATM VP Service* , SPIE Conference Proceedings, Vol:2953, Berlin 1996.

[117] T. zhang, S.Covaci, *Java-based Mobile Intelligent Agents as Network Management Solutions*, in Proceedings of the 8th Joint European Networking Conference, May 1997.

[118] T. Zhang, S. Covaci, *Moving Knowledge Agents for Network and Service Management*, in Proceedings of IEEE/IFIP MMNS'97, Montreal, Canada, November 1997.

[119] T. Zhang, S. Covaci, *OMG and FIPA Standardisation for Agent Technology: Competition or Convergence?*, in Proceedings of the European ACTS Agent Workshop, Brussels, February 1998.

[120] T. Zhang et.al., *Programming the Active Broadband ATM Networks Using Mobile Intelligent Agents*, Project Proposal, GMD FOKUS, June 1998.

[121] T. Zhang, *Mobile Agents vs. Intelligent Agents - Interoperability and Integration Issues*, Journal of Interoperable Comminucation Networks, ISSN 1385 9501, Baltzer Science Publishers, July 1998.

[122] T. Zhang, *DOT vs. Agent Technology in Telecommunication Applications*, Proceedings of the EURESCOM DOT'98 Workshop, Heidelberg, Sptember 1998.

[123] T. Zhang, *FACTS- Validating the FIPA Standards*, Proceedings of the EURESCOM DOT'98 Workshop, Heidelberg, Sptember 1998.

[124] T. Zhang, *Active VPN Service Based on Agent Technology*, in Proceedings of the Telecommunications Information Networking Architecture Conference 1999, Oahu, Hawaii, April 1999.

[125] T. Zhang, *RDF Schema for VPN Services*, http://mimosa.fokus.gmd.de/facts/vpn_schema, 1999.

[126] T. Zhang, *Dynamic Service Adaptation*, in FACTS A2 Deliverable 1, June 1999.

ACRONYMS

ACL	Agent Communication Language
AMS	Agent Management System
AP	Agent Platform
API	Application Programming Interface
CL	Content Language
CNM	Customer Network Management
DOT	Distributed Object Technology
DTD	Document Type Declaration (W3C)
FIPA	Foundation for Intelligent Physical Agents
GCS	Global Connectivity Service
IA	Intelligent Agent
MA	Mobile Agent

MASIF	Mobile Agent System Interoperability Facilities Specification (OMG)
NMS	Network Management System
OMG	Object Management Goup
OSA	Ontology Service Agent
QoS	Quality of Service
PNO	Public Network Operator
RDF	Resource Description Framework (W3C)
RDL	Resource Description Language
RPC	Pemote Procedure Call
TMN	Telecommunications Management Network (ITU-T)
VAS	Value Added Service
VASP	Value Added Service Provider
XML	Extensible Markup Language (W3C)
XSL	XML Stylesheet Language (W3C)