Chapter 7

DESIGN AND IMPLEMENTATION MODEL FOR THE BEHAVIOR MANAGEMENT ARCHITECTURE OF LANGUAGE FACULTY OF AN AGENT

The design model helps to translate the architectural model into suitable logical abstractions that could be used to semantically represent the architectural components and explicate their intricacies in the design domain. The implementation model is the next step towards physically realizing the architecture and design by helping to map the logical abstractions into high-level language entities that can be used in the software development process.

The architectural model described in the previous chapter explained how the management and the behavior functions could best be logically abstracted as roles thus leading to the identification of the manager and behavior roles This chapter proceeds to provide the design model by assigning the roles to appropriate agents and providing the overall organization diagram of the language faculty. It further explores into the manager and behavior roles to explicate their relationships and role dynamism, all of which contribute for the design model.

The implementation model depends on the semantic representation that is used to conceptually represent roles. Since multiple alternatives for the semantic representation of roles is available, a suitable choice of the same is made. This helps to identify the suitable language level abstractions that should be used to realize the roles. The implementation model is given in terms of these language level abstractions.

Thus, the objectives of this chapter are to provide the following:

- Role-based design model and
- Implementation model using the appropriate choice of the semantic representation of roles.

7.1 Role-Based Design Model of Language Faculty of Agents

The previous chapter described how the role-based approach was appropriate to logically realize the behavior management functions and the corresponding behaviors. That is, the language faculty is conceived as a repertoire of roles corresponding to the management functions and the behaviors that they manage. Therefore, the language faculty is an organization comprising of behavior manager roles and behavior roles. When the language faculty is configured to a particular language, the behavior manager roles corresponding to the interface behavior management, competence behavior management and knowledge behavior management assume the behavior roles in the corresponding language. They then help to provide natural language interaction in the corresponding language. The subsequent sections provide the design of the language faculty in terms of the organization, static and dynamic role diagrams. The former two help to model the static aspects and the latter helps to describe the dynamic aspects of the language faculty. These diagrams and their abstractions correspond to that of MAS-ML (Multi-agent System Modeling Language) (Silva, Choren and Lucena, 2004 a, b). Though there are other modeling languages like that of AUML (Odell, Parunak, and Bauer, 2000) (Bauer, 2001) and AORML (Wagner, 2000) available, MAS-ML has been chosen because only MAS-ML helps to capture both the static and dynamic aspects of roles and also represent them effectively in the modeling diagrams.

7.1.1 Organization Diagram

An organization diagram models an environment, an organization, its sub-organizations, roles defined in the organization and the elements that play those roles. The language faculty of the agent could be considered as an organization that is composed of two sub-organizations namely, the language faculty and the functional ability. The language faculty helps to provide for the natural language interaction pertaining to the functional ability. The language faculty sub-organization is delved into further to explicate the roles in it. The language faculty is composed of roles pertaining to interface behavior management, function behavior management, knowledge behavior management and new behavior acquisition management as explained in the architecture. These roles are played by the Interface Agent, Function

Agent, Knowledge Agent and New Behavior Acquisition Agent respectively. These agencies are not individual agents and exist within the language faculty itself Figure 7.1 shows the organization diagram. The sub-organizations are shown as an oval shape, agents are represented as rounded rectangle and roles are represented as solid rectangle with a curve on its bottom. These notations correspond to that of MAS-ML.

7.1.2 Role Diagram

The role diagram complements the organization diagram by modeling the relationships between the agent roles, between agent roles and object roles, between object roles, and between the roles and the classes that they use / define. The relationships that are used in this diagram are control, dependency, association, aggregation and specialization. These roles and their relationships constitutes for the static aspects of the role diagram. The dynamic aspects elicit the interaction between the roles. Thus, the role diagram is represented in terms of:

- Static Role Diagram
- Dynamic Role Diagram.

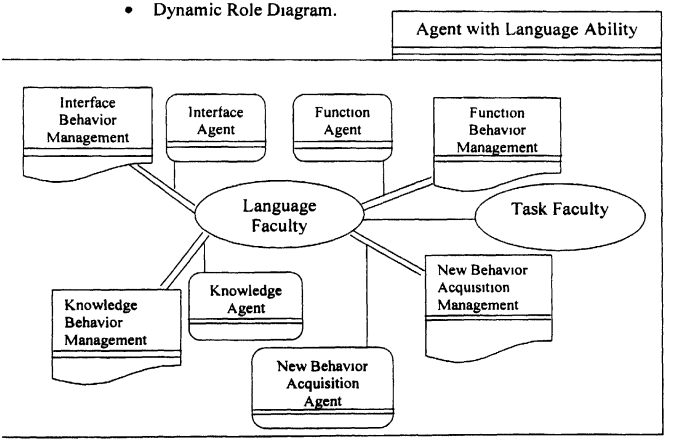

Figure 7.1: Organization diagram of an agent with language ability

To define the static role diagram, the following two things have to be described.

- Internal role details of the management functions of language faculty
- Relationships between the roles of the management functions

Internal Role Details of the Behavior Management Functions

Every management role of the language faculty is identified to be a composite role which is composed of two sub roles as follows:

- Manager role and
- Behavior roles.

The manager role has responsibilities to manage the behavior roles. This includes the following as elaborated in the BTB model described in chapter 4.

- Listen to percept and decide upon the behaviors to produce
- Maintain the task contexts
- Maintain the behavior references
- Assume the required behavior role depending on the requirement
- Perform task level inference of behaviors
- Implement the newly acquired functional behaviors.

The behavior roles correspond to the various behaviors that are managed by the manager role. The manager role assumes a behavior role when it decides that it has to deliver an appropriate behavior. If any exception is raised in the behavior role, or the current percept requires another behavior to be delivered, then the manager role is retracted back, which handles the exception or again configures itself to the required behavior role.

Relationships between the roles of the management functions

Relationships between roles is of two types viz.,

- Structural
- Logical.

Structural Relationship:- Structurally, the manager role and the behavior roles of the language faculty are related by means of a hierarchy with a superior sub-ordinate relationship. That is, the manager role is the highest role of the hierarchy that is

followed by the behavior roles which are sub-ordinate to the manager role. The manager role is the default role of the management functions so that the behavior roles retract to this role after producing the required behavior. Hence, for every percept that is received from the environment through the listener, the manager role determines whether the currently active behavior role is valid in the current context or it has to relinquish that behavior role and assume the manager role in order to decide which of the other behavior roles it has to assume. In other words, only if the agent is performing the manager role it can assume the behavior role. This is depicted in Figure 7.2

Logical Relationship:- When identifying the logical relationship between the manager and the behavior roles of the language faculty, two types of relationships could be identified viz.,

- static relationship
- dynamic relationship.

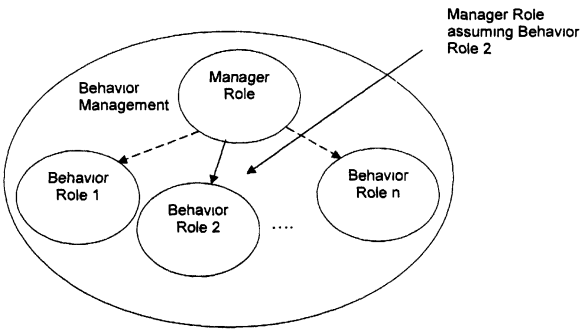

Figure 7.2: Hierarchical Structural Relationship of the Manager and Behavior Roles of the Language Faculty

The static relationship exists during the compile time and the dynamic relationship is picked up during execution time. The various forms of logical relationships that MAS-ML defines between the roles are inhabit, ownership, play, specialization,

control, dependency, association and aggregation. Analyzing these relationships, it is found that the control relationship exists at compile time which is transformed into aggregation relationship at execution time. The control relationship specifies that the controlled element must do anything that the controller element requests, which implies that the behavior roles are instructed regarding the behavior to perform by the manager role The aggregation relationship specifies that the aggregator may use functionality available in its aggregation, whereby the manger role acquires the required behavior role when required.

The above described structural and logical relationship applies to all the behavior and manager roles of the various management functions of the language faculty. As an illustration, the discussion below describes the relationships with respect to the behavior interface management function.

The interface behavior function management is composed of the interface manager role and the interface behavior roles in all the supported languages The interface manager role controls the interface behavior roles in all the supported languages. At run time, when the language faculty is configured to a particular language, the manager role aggregates the interface behavior role in the corresponding language in order to deliver the corresponding behavior This is represented in Figure 7.3.

The control relationship is represented using a small circle shape. The aggregation is represented by using a diamond shape. The dashed diamonds indicate that the corresponding behavior roles are prospective roles that could be aggregated by the manager role. The diamond with full lines indicate the behavior role that is currently aggregated by the manager.

The interface manager consists of the task context which is an object. Similarly, the various behavior roles in the different languages possess a behavior context object that helps to keep track of the behavior in the corresponding language.

When the language faculty is configured to a particular language, it ultimately results in the interaction between the various behavior roles as depicted in Figure 7.4.

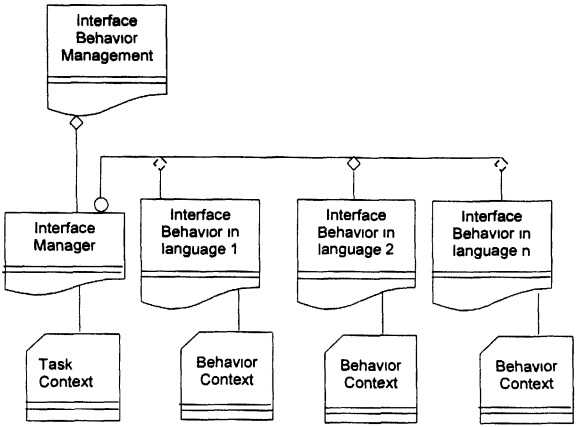

Figure 7.3: Logical (Static and Dynamic) Relationships between Interface Manager and the Interface Behavior Roles

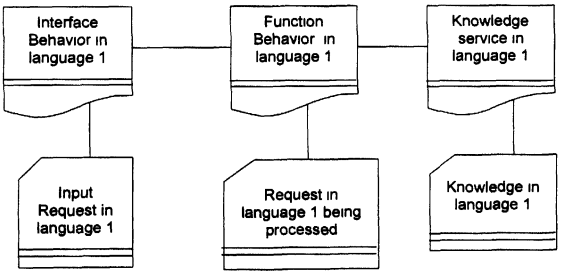

Figure 7.4: Relationship between the various roles of the language faculty

These roles are related to one another by means of the association relationship. Assuming that the language faculty is configured to language 1, the input request

received by the interface behavior role in language 1 is processed by the function behavior role in language 1. The required knowledge in language 1 is provided by the knowledge service behavior in language1. Each of roles comprise of the objects corresponding to the input request, processed or processing request and knowledge required for processing respectively.

7.1.2.2 Dynamic Role Diagram

The dynamic role diagram describes how the various manager roles of the management functions change roles between the manager and behavior roles. This diagram should help to depict the following.

- Dynamic assumption of behavior role by the manager role.
- Bringing the behavior role active only at run time when the manager role decides that the corresponding behavior role is appropriate. At all other times, the behavior roles are passive.
- Helping the manager role to retain minimum functionality when assuming the behavior role – listening to the incoming percepts and deciding whether the assumed behavior role is appropriate in the current context.

To specify the dynamic aspects of roles, MAS-ML defines the following stereotypes viz create, destroy, role commitment, role cancel, role activate, role deactivate and role change. Create and destroy are used by the language faculty to create and destroy the various management agents. The management agents are created with their corresponding manager roles activated.

Analyzing the various stereotypes, the 'Role Activate' stereotype is found to be appropriate to represent the dynamic role binding of the manager role to the behavior role. 'Role Activate' activates an inactive role. This requires that the various behavior roles should already be in an inactivate state, which implies that these roles should have been already created, i.e., statically bound. The required behavior role is activated when the corresponding behavior role is required to be performed. Also, the manager role is not cancelled when activating a behavior role. Hence, this stereotype is chosen to be appropriate to represent the role transformation from the manager role

to the behavior role. The role dynamics is described with respect to the language faculty as given below.

When a particular user logs in, the interface manager role of the interface agent that updates and maintains the user's preferred language determines the preferred language of this user and activates the interface behavior role in the corresponding language. If the user changes the working environment to language 'y', then it deactivates the current interface behavior role and changes the interface to this language 'y' by activating the corresponding behavior role This is depicted in Figure 7.5.

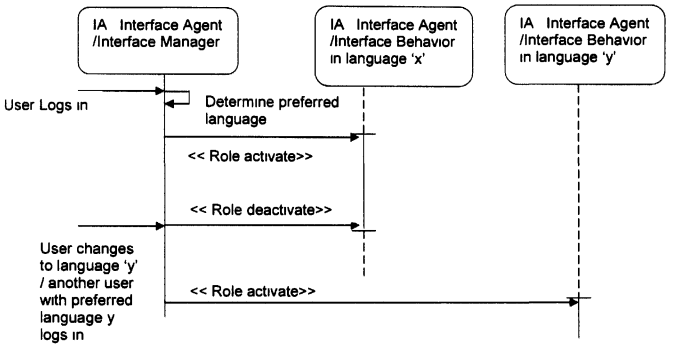

Figure 7.5: Dynamic Role assumption of interface manager role with interface behavior roles

In addition to activating the required interface behavior role, the interface manager role also intimates the preferred language of the currently logged in user to the function manager role of the function behavior management and to the knowledge manager role of the knowledge behavior management function, each of which activate the required language behavior role. This is depicted in Figures 7.6 and 7.7.

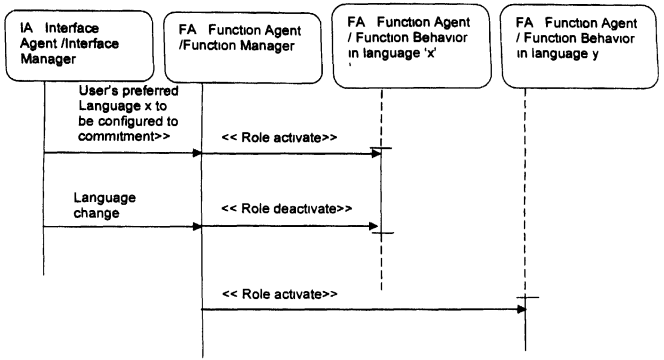

Figure 7.6: Dynamic Role assumption of function manager role with function behavior roles

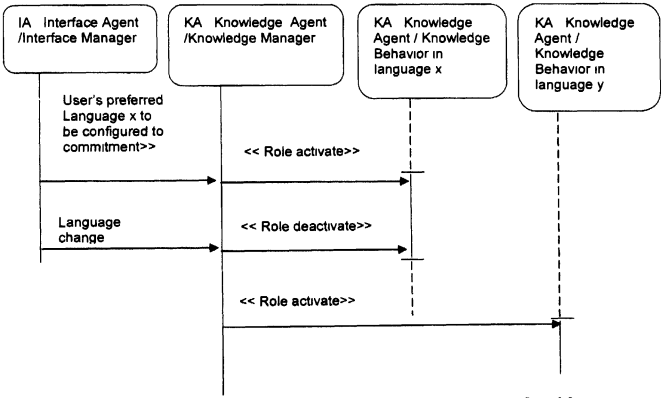

Figure 7.7: Dynamic Role assumption of knowledge manager role with knowledge behavior roles

Thus, the role-based design models describe the various roles, their relationships, and their dynamics. The implementation model described subsequently is based on the

role-based design model. The implementation model should fulfill the following design requirements:

- Manager and Behavior are the two simultaneous roles of Behavior Management.
- Manager and Behavior roles operate in different contexts.
- Manager role takes on the required behavior role dynamically.
- Manager role is the default role of the behavior management.
- Manager role manages or controls the behavior role.
- Manager role should be able to include new behaviors.

7.2 Implementation Model

The concept of roles is an inevitable abstraction in the design and implementation of agents. The role-based design model described above depicts the same. The implementation model should help to physically realize the logical abstractions namely the roles, their relationships, and their dynamism as described above. In proposing the implementation model for agents, the following two approaches are possible:

- Agent-oriented implementation
- Role-specific implementation.

In the former approach, though the role abstraction and its semantics are well captured in the design phase, it is implemented using the available object-oriented abstractions and their semantics in the implementation phase. Thus, the various roles of an agent are realized as classes and they are related to the agent or among themselves by means of association, aggregation or specialization relationship which could be directly implemented in any object oriented language. In the case of the latter, the emphasis is on the conceptual and physical implementation abstractions required to realize roles with their conceived notion. The main difference between the two approaches is highly felt only in the way in which they help to achieve the role dynamism. Whereas role dynamism is soft spoken in the former, it is of major concern in the latter.

In realizing the implementation model of the above described role-based design model, both the above approaches have been explored with.

7.2.1 Agent-Oriented Implementation

The Implementation model is provided in terms of the classes required to implement the role-based design defined above. Though the role-based design is the same for the function behavior management, knowledge behavior management and new behavior acquisition management, they differ in the class level based on the function they perform. Hence, the class diagram of each of these is described below individually.

7.2.1.1 Class Diagram of Interface Behavior Management

The class diagram of the Interface Behavior Management is given in Figure 7.8. Since the Interface Behavior Manager role is the default role of the behavior management, it becomes an integral part of the Interface Behavior Management. Hence, it is depicted by means of the aggregation relationship represented as a filled diamond. The methods 'send' and 'receive' are used for interaction with the other management components. It interacts with the function behavior management to convey the perceived input and receive the response to be conveyed to the environment in the corresponding language. It interacts with the knowledge behavior management to obtain the knowledge resources required to perform the 'perceive' and 'respond' functions in the languages supported. It interacts with the new behavior acquisition management to obtain and deploy the interface behaviors for a new language. The newly acquired behaviors are deployed in the interface behavior management using dynamic class loading. In dynamic class loading, the class file corresponding to the new behavior is obtained and included dynamically as a subclass of the function behavior class.

The behavior role class is used by the manager to deliver the required interface behavior appropriate in the current context. This is represented by the uses 'relationship'. The behavior roles are implemented as a hierarchy where, the 'perceive' and 'respond' function behaviors for the various languages are realized by means of the required subclasses and implementing the virtual perceive() and respond() functions. This is because, each of the behavior classes have to provide for the same perceive and respond functionality for which only the procedure is different

for different languages. So depending upon the environment, the required function behavior is dynamically linked to the behavior manager.

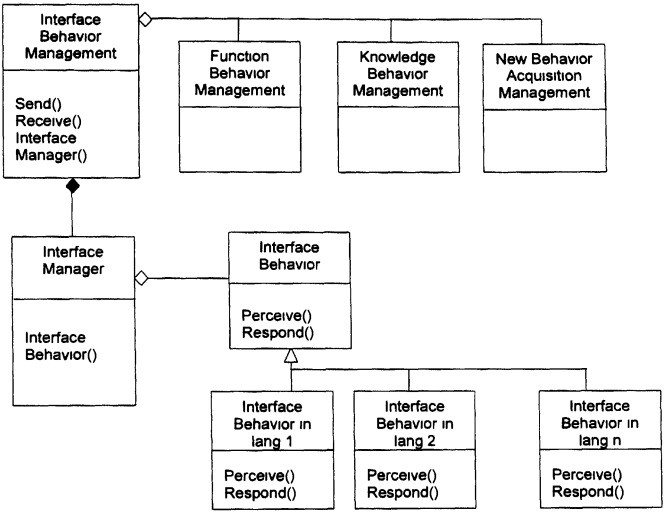

Figure 7.8: Class Diagram of Interface Behavior Management

7.2.1.2 Class Diagram of Function Behavior Management

The class diagram of the Function Behavior Management component is depicted in Figure 7.9. The manager role is the default role and it is depicted as an aggregate component of the function behavior management. It interacts with the Interface Behavior Management component to get the user's preferred language of interaction and input request. It conveys the response through the Interface Behavior Management component only. Similarly, it interacts with the Knowledge Behavior Management component for obtaining the details of the knowledge required for performing required CNLI behavior.

The Manager role makes use of the language behavior roles in the languages supported by the agent, to invoke the corresponding CNLI behavior. When a new language behavior is to be added, the class file corresponding to the new behavior is included by means of dynamic class loading.

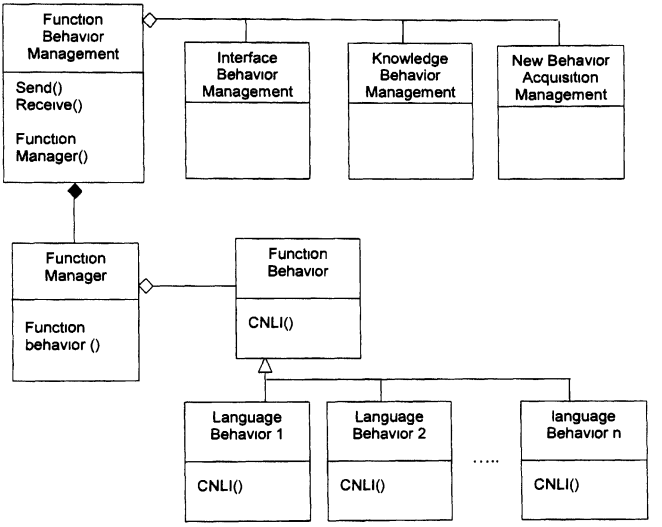

Figure 7.9: Class Diagram of Function Behavior Management

7.2.1.3 Class Diagram of Knowledge Behavior Management

The class diagram of behavior knowledge management is given in Figure 7 10. Similar to the function manager role, the knowledge manager role is an integral part of the knowledge behavior management, which is depicted by means of the aggregate relationship. The knowledge behavior management is made use of by the function behavior management for its knowledge services. The new behavior acquisition

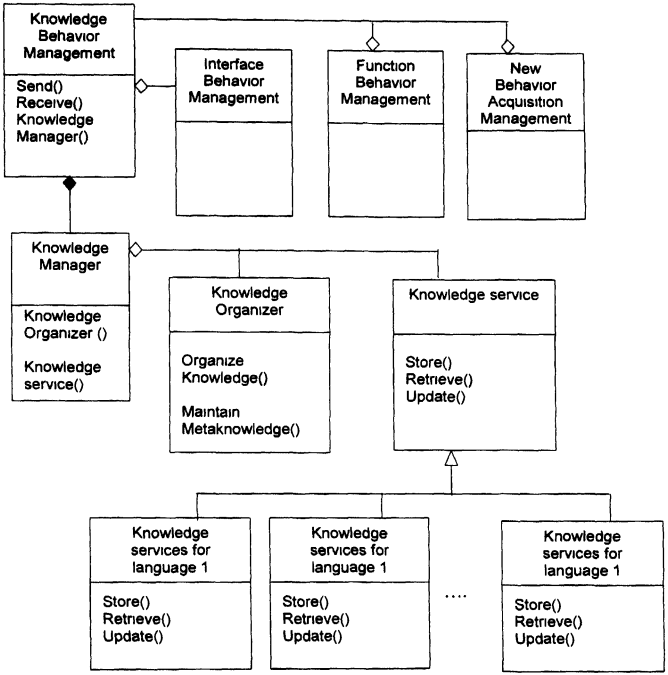

Figure 7.10: Class Diagram of Knowledge Behavior Management

management makes use of the knowledge behavior management for organizing and storing the acquired knowledge. When the knowledge of a particular language has to be updated, the interface management is made use of to obtain the update particulars.

The knowledge manager makes use of a knowledge organizer to organize the knowledge in the required folders and to keep track of the metadata of the language knowledge. This metadata is used whenever any language knowledge is to be referenced. The knowledge manipulation operations like store, retrieve and update are available for each of the languages. These are knowledge representation specific. The

knowledge manager is configured to the required manipulation operations depending upon the language to which the language faculty is configured to. Similarly, to bring in a new language behavior, its knowledge manipulation operations are acquired and implemented using dynamic class loading.

7.2.1.4 Class Diagram of New Behavior Acquisition Management

This class diagram as given in Figure 7.11 is also similar to that of the previous ones, where it indicates that the acquisition management has the acquisition manager as the inherent role by the aggregate relationship and the acquiring behavior being implemented as a generalization and specialization relationship. The acquiring behaviors are classified into different types according to the ones in which they can be grouped into. For example, in the case of acquisition of languages, the acquisition of behaviors may be classified according to the types of language, which is depicted in Figure 7.11.

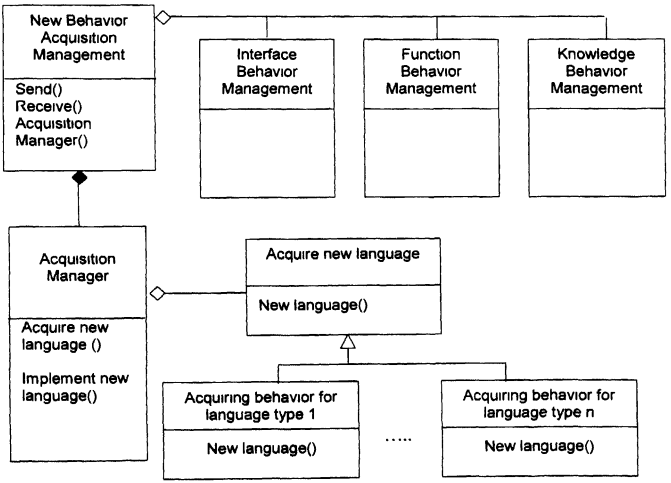

Figure 7.11: Class Diagram of New Behavior Acquisition Management

Thus, the described class diagrams are instrumental in providing the agent-oriented implementation model for the role-based design of the various management functions.

7.2.2 Role-Specific Implementation Model

Here, the implementation model is given as a role-specific implementation model and focuses on realizing the semantics of roles at the implementation level. Hence, high emphasis is given on the ways in which the semantics of roles could be transferred into semantics of the implementation model. Different alternatives of semantic representation have resulted in the different ways of realizing roles during implementation The various alternatives are discussed and the one chosen for the implementation model of the language faculty is described finally.

In providing for the role-specific implementation model, the following are performed.

- Identifying the requirements (functional and characteristic) to be fulfilled by the implementation model
- Determining the semantic representation of the roles
- Providing the role-specific implementation.

7.2.2.1 Requirements to be fulfilled by the Implementation Model

- The Functional Requirements are specified at the end of the role-based design model
- For the characteristic requirements, the set of characteristic features a role should fulfill (as described in the literature review) are considered. From among these features, the ones that are required to be fulfilled for the role-specific implementation are given in Table 7.1.

Table 7.1: Characteristic Feature requirements of the Role-specific Implementation Model

Characteristic Feature of Roles	Features required to be fulfilled by the Role-specific Implementation Model	Reason
Ownership	√	The Manager role and Behavior roles have their own properties and behavior.
Dependency	√	The Manager role and behavior roles are dependent on the corresponding Agents that play that role.
Diversity	√	The Agent may play the Manager role and the Behavior role simultaneously as the Manager role always has to listen to the incoming percept and decide whether its commitment to the current Behavior role is appropriate for the received percept.
Multiplicity	*	May be required if the language faculty supports multiple users, where multiple instances of the Behavior roles have to be created, one for processing each user's request.
Dynamicity	√	The Manager role has to acquire and relinquish the required Behavior roles dynamically.
Control	√	The Behavior role could be acquired only after the Manager role is acquired.
Roles can play role	√	The Manager role plays the various behavior roles.
Role Identity	√	Each instance of a role has its own instance identifier.
Adoption	√	Behavior roles do not inherit properties from the Manger roles.
Relationship Independency	√	The Behavior roles help to provide for functional behavior. Hence, they are meaningful even when they are out of the context of the Manager role. For example, the natural language interaction behavior roles help to provide for natural language interaction in the required languages. Hence, they are meaningful even when they are out of the context of the Language Faculty.
Common Role for unrelated Types	X	Not relevant in the context of the Manager and Behavior Roles.

Sharing Structure and behavior	√*	In a generic sense the various behaviors managed by a manager may be versatile in nature and need not share the same structure and behavior. But, in the case of language faculty all the behavior roles share the same structure and behavior as they have to provide for the same behavior in the various languages.

7.2.2.2 Semantic Representation of Roles

The Role concept has been widely used in the literature and different patterns for the representation of Roles are available. Table 2.1 in the literature review gives a summary of the same. Analyzing the various semantic representations of roles, it could be found that 'roles as entity types' (Cabot and Raventos, 2004) implementation approach helps to achieve almost all of the characteristic properties of roles which were specified above Very importantly, the representation of 'roles as entity types' would be most appropriate to represent the roles performed by agents. This is because, only this representation allows the roles to be defined as entity types with their own attributes, relationships and generalization / specialization hierarchies.

Thus, from the above descriptions, it could be inferred that the 'roles as entity types' pattern is a suitable semantic representation for implementing roles

In translating this pattern to the implementation domain, the following two possible alternatives are possible:

- Object-Oriented Implementation
- Aspect-Oriented Implementation.

In the former approach, the roles as entity types pattern is implemented by two ways - using the simple association relationship between the core entity type and the role entity, or with bytecode manipulation of the natural entity type in order to play the role entity type. The implementation described in the section 7.2.1 corresponds to the agent-oriented implementation with object-oriented abstractions. Hence, the aspect-oriented implementation is tried out now. This is because, aspect-oriented implementation exactly fits in to meet the requirements of roles and their dynamism

required by the language faculty. The aspect-oriented implementation model of the behavior management architecture is described below.

7.2.2.3 Role-Specific Implementation - Aspect-Oriented Implementation Model

The aspect-oriented implementation is given in terms of the core function, the point-cut that help to introduce the role behavior and the aspect that contains the role behavior Figure 7.12 gives a generic aspect-oriented implementation model that is applicable for any of the behavior management functions

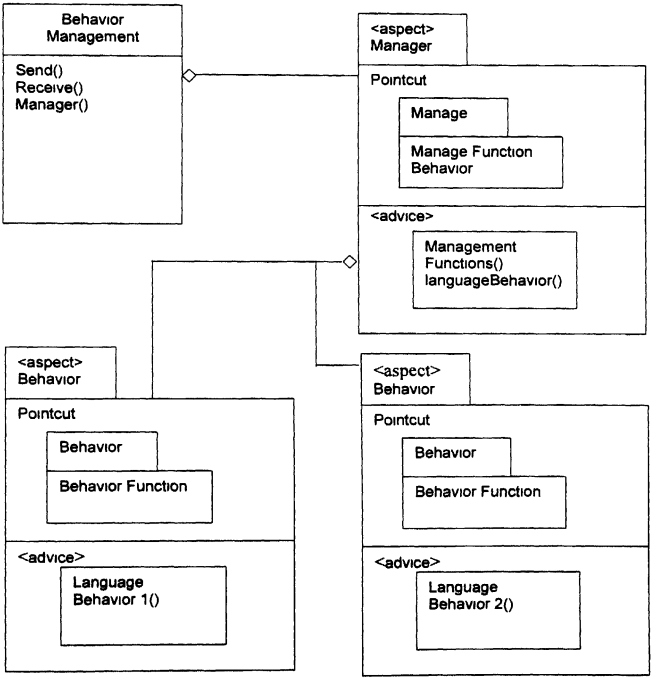

Figure 7.12: Aspect-Oriented Implementation of Manager and Behavior Roles

This implementation corresponds to the second option out of the six options described by Kendall in implementing roles, which is described in the literature review. In this option, the aspect instance adds role behavior by advising role members that already exist in a core instance. This option has been chosen because, the manager role is the default role of the behavior management and hence its interface should already exist in the core behavior management instance. The manager aspect gives the definition of the manager role that has to be woven at the manage() point-cut of the behavior management.

Similarly, the behavior role has also been implemented using the same option whereby the behavior aspect is woven into the point-cut defined in the manager aspect. However, the behavior aspect weaving is conditional depending upon which of the behavior roles the manager role wants to invoke. The behavior role is defined as a nested aspect because of the functional requirement that the behavior role is controlled by the manager role.

A comparison of the agent-oriented implementation and role-specific implementation is given in Table 7.2.

Table 7.2: Comparison of Agent-Oriented and Role-Specific Implementations of Roles

S. No.	Agent-Oriented Implementation Approach	Role-Specific Implementation Approach
1.	Emphasis is on roles from an organizational perspective focusing on role responsibilities, relationships and their interaction.	Emphasis is on the concept of roles only focusing on their relationships, and their role dynamism.
2.	No different semantic representations are available.	Various patterns for semantic representations are available.
3.	Role is only a design abstraction and no specific implementation methods are available.	Other approaches like Aspect-Oriented Programming are also available for realizing roles.
4.	Implemented using object-oriented abstractions.	Implemented using object-oriented abstractions with extensions for role dynamism.
5.	Agent roles should possess agent specific properties.	Role-specific properties are important and agent-specific properties are not applicable.

7.2.3 Validation of the Implementation Model

The requirements that should be fulfilled by the implementation model are considered and how they are fulfilled in the implementation model are given in Table 7.3.

Table: 7.3 Functional Requirements of the Role-based design model of the language faculty of agents and its fulfillment in the Implementation Model

S. No.	Requirements	Fulfillment in the Implementation Model
1.	Manager and behavior are the two simultaneous roles of Behavior Management.	The behavior role is initiated through the manager role.
2.	Manager and Behavior roles operate in different task contexts.	Both are implemented as two different classes.
3.	Manager role takes on the required behavior role dynamically.	Achieved by using run-time polymorphism or aspect weaving.
4.	Manager role is the default role of the behavior management.	Manager role aggregated with behavior management.
5	Manager role manages or controls the behavior role.	The manager role decides which of the behavior roles to initiate and activates it accordingly.
6.	Manager role should be able to include new behaviors.	New behaviors are added by dynamic class loading by the manager role in the agent-oriented implementation

7.3 Summary

This chapter delves into the role-based design of the behavior management architecture to explicit the organizational structure, relationship and the dynamism of the manager and behavior roles. This helps to provide for a role-based design model.

In implementing roles, several conceptual representations of roles are available. These could be categorized into the agent-oriented and role-specific approaches. In the agent-oriented approach, the emphasis is on the role responsibilities and relationships. The object-oriented concepts of abstraction, aggregation, association and specialization relationships are used to implement the semantics of roles.

In the role-specific approach, the emphasis is on the concept of roles and their dynamism. Various possible semantic representations suggested by various authors are available in the form of patterns. Among these, the best pattern representing roles

is chosen. In implementing this pattern, the Aspect-Oriented Programming approach has been used because the aspects are similar to roles in various ways and also better help to encapsulate agent role behavior and its dynamism.

The main contributions of this chapter towards realizing the language faculty of agents are :

- Role-based design model for the Language Faculty and
- Implementation model of the Role-based design model using
  - Agent-Oriented approach
  - Aspect-Oriented approach.