İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DISTRIBUTED DECISION MAKING: A MULTIAGENT DECISION SUPPORT SYSTEM FOR STREET MANAGEMENT

M.Sc. Thesis by
Figen ÖZTOPRAK
507031111

Date of submission : 9 May 2005
Date of defence examination: 30 May 2005

Supervisor (Chairman): Ass. Prof. Dr. Şule Önsel ŞAHİN
Members of the Examining Committee Prof.Dr. Ramazan EVREN (AÜ.)
Prof.Dr. Burç ÜLENGİN (BÜ.)

MAY 2005

FOREWORD

Nobel Prize winner Albert Szent-Gyorgi says “discovery consists in seeing what everyone else has seen and thinking what no one else has thougt”. This was the sentence I first remembered when I began to read about the subject of distributed decision making and especially about some multiagent system implementations, e.g. organizational models with inspiration from honey-bee colonies, football playing robots, communicating traffic flow sensors, etc.

After completing this thesis, I see everything around me as agents – alive or not. I think, this is an alternative style of understanding, and should be insisted on because every kind of systems are expected to become wider and more complex in the future.

Finally, I want to express my gratitudes to my supervisor Ass.Prof.Dr. Şule Önsel ŞAHİN for her help and patience and Prof.Dr. Füsun ÜLENGİN especially for directing me to this subject. I also thank to Assoc.Prof.Dr. Cengiz GÜNGÖR for providing me the chance of using street management as an example.

Figen ÖZTOPRAK

May, 2005

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DAĞITIK KARAR VERME: CADDE YÖNETİMİ İÇİN ÇOK AJANLI BİR KARAR DESTEK SİSTEMİ

ÖZET

Çevremiz, kavramsal ya da coğrafi anlamda dağınık olan ve insanlar da dahil olmak üzere çok sayıda bileşenden oluşan karmaşık sistemlerle doludur. Dağıtık karar verme (DKV) sistemi çok sayıda dağınık karar verme biriminden oluşur, DKV problemi ise basitçe "bağlantılı kararların tasarımı ve koordinasyonu" olarak tarif edilebilir.

Bu çalışmada, işbirlikçi bir insan organizasyonunda kararlar etkinliğinin operasyonel seviyede eğitim ve dağıtık öğrenme ile, daha yüksek seviyelerde ise dağıtık karar destek araçlarıyla arttırılabileceğini öngörüyoruz. Daha sonra, çok ajanlı sistem mimarisine dayalı ve insan kullanıcıların da aktif olarak karar sürecine katıldığı bir dağıtık karar destek sistemi önerisi sunuyoruz. Dağıtık karar destek araçlarının, dağıtık insan organizasyonlarının performansını arttırmadaki uygulanabilirliğini cadde yönetimi örneği üzerinde göstermeyi planlıyoruz.

Cadde yönetiminin uygulama alanı olarak seçilmesinin nedeni dağıtık karar verme için iyi bir örnek teşkil etmesidir: Kararlar, bir takım sorumlu personel tarafından farklı zamanlar ve farklı mekanlarda, ancak cadde performansını yükseltme ortak amacıyla alınmaktadır. Uzmanlık ve sorumluluk çok sayıda farklı yönetici birim arasında dağılmıştır ve yüksek performans ancak bu farklı birimlerin koordine çaba göstermesiyle mümkündür. Tecrübeler göstermektedir ki merkezi bir kontrol / koordinasyon biriminin bulunması ne etkili ne de etkin değildir.

Dağıtık organizasyonel modelimizi oluşturabilmek için, eldeki problem yapısına daha uygun olduğu için fonksiyonel ayrım tercih edilmiştir. Gerçek hayatta, herhangi bir yerel yönetim kuruluşundaki ilgili bir birim bu fonksiyonel ajanların bir ya da birkaçının faaliyetlerini gerçekleştiriyor olabilir. Böylece, "birimler" ve "ajan tipleri" kümelerini kesiştirerek 26 farklı "cadde yönetim ajanı" elde edilmiştir. Cadde yönetim ajanları cadde performansını enbüyükleme ortak amacı ile işbirlikçi bir çalışma gerçekleştirmektedir.

Cadde yönetim sisteminin performans modelini kurmak için "bulanık zihinsel harita" yaklaşımı kullanılmıştır, çünkü ihtiyacımız olan ilişkiler açık kurallardan çok uzman değerlendirmelerine dayanmaktadır. Performans modeli ile ilgili hesaplar üç uzmanın görüşlerine başvurularak yapılmıştır. Çok ajanlı karar destek sistemimizde, performans modeli bir "performans simülasyoncu" ajan tarafından yönetilmektedir, diğer ajanların aktardığı yeni durumlara ilişkin performans değerleri hakkında onları bilgilendirmektedir.

Önerilen çok ajanlı karar destek sisteminin çerçevesi *act*SMART modeline uygun olarak ve literatürde mevcut dağıtık karar destek sistemi mimarileri ve iş akışlarından faydalanarak oluşturulmuştur. Sistem iki bölümden oluşmaktadır: İnsan

kullanıcılarla iletişimi sağlayan ara yüzler ve karar vericiler için öneriler geliştiren bir çok ajanlı sistem. 11 farklı tip kullanıcı için 11 farklı ara yüz olmalıdır.

Buna karşılık, uygulama noktasında sistemin tamamı bu çalışmanın amaçlarını aşacak biçimde geniş ve kapsamlı bulunmuştur. Bu nedenle, önerilen modelin uygulanabilirliği sistemin örnek bir bölümü üzerinde gösterilmiştir; örnek iş akışı 4 cadde yönetimi ajanı ve iki kullanıcı arasında geçmektedir. Ajanları tanımlamak için daha önce de belirtildiği gibi *act*SMART yapısı kullanılmıştır.

Örnek uygulamada, çok sayıda varsayımda bulunduk. İş akışı, aşağıda adım adım verilmiştir:

- "dinleyici" ajan R1S1 caddesinde yeni bir kaplama problemi olduğu bilgisini ara yüzü aracılığıyla şikayet sahibi kullanıcıdan alır ve durumu "performans simülasyoncu" ajana iletir.
- "performans simülasyoncu", bulanık zihinsel harita yaklaşımına dayanan performans modelini işleterek onarım işinin daha uygun olacağına karar verir ve ilgili "yol onarımcısı" ajana bir mesaj yollar.
- Daha sonra, kaplama problemi ile ilgili bilgi "yol onarımcısı" ajana ait ara yüzde görüntülenir ve ilgili insan kullanıcı teklif edilen onarım işini onaylamaya karar verir.
- Onay alındığında, "yol onarımcısı" ajan, "sistem zamanlayıcısı" ajana bir mesaj gönderir.
- "sistem zamanlayıcısı" ajan, ajandasındaki işleri kontrol ederek kaplama işi için uygun bir zaman belirler ve önerisini "yol onarımcısı"na bildirir.
- Ardından, tüm gerekli destek bilgileri "yol onarımcısı" ajana ait ara yüz ekranında görüntülenir ve "dinleyici" ajana işe dair alınan kararları bildiren bir mesaj, şikayet eden kullanıcıyı bilgilendirmek üzere gönderilir.

Önerilen dağıtık karar destek sisteminin örnek uygulamasının gerçekleştirilmesi için bir yazılım geliştirilmesi gerekmiştir. Ajan oluşturmak için Java yazılım dili kullanılmıştır, çünkü Java ortamı ajan tasarımı ve oluşturulması için gerekli tüm temel fonksiyonlara sahiptir.

Java ortamında üretilen örnek uygulama yazılımı 2 soyut sınıf, 7 genel sınıf ve 3 veritabanı tablosundan oluşmaktadır. İlk soyut sınıf StreetAgent sınıfıdır; diğer ajan sınfları bu temel sınıfın açılımlarıdır. Diğer soyut sınıf ise Message sınıfıdır; bu sınıf ise üç mesaj sınıfı için bir üst sınıf teşkil etmektedir: CaseMessage, PerfMessage and TimeMessage. Son olarak, StreetManagement veritabanı STREETINFO, AGENDA ve STANDARDS adında üç tablo içermektedir.

Örnek uygulama başarıyla gerçekleştirilmiştir. Böylece, önerdiğimiz dağıtık karar destek sisteminin uygulanabilirliğini, bilgisayara dayalı desteğin gerçek zamanlı koordinasyonu nasıl sağladığını ve bir karar destek sisteminin karar vericiye daha iyi kararlar vermesinde nasıl yardımcı olduğunu –örneğin, sistem performansı üzerindeki etkileri hesaplayarak ve sunarak- göstermiş olduk. Bunun yanında, daha sonraki çalışmalar için geniş bir alternatifler kümesi bulunmaktadır. Bunlardan birisi, burada önerilen sistemin geliştirilmesidir. Gerçek hayatta kullanılabilecek biçimde daha kullanışlı bir sistem ortaya koymak için, örnek uygulama genişletilmeli, veritabanı zenginleştirilmeli, daha az sayıda varsayım ve ihmal yapılmalıdır. Diğer bir alternatif ise benzer bir yapıyı, yine dağıtık bir insan

organizasyonu içeren farklı bir uygulama alanında kullanmaktır. Gelecek çalışmalar cesaret vericidir, çünkü dağıtık karar sistemlerine olan ihtiyaç karar problemleri karmaşıklaştıkça daha da artacaktır.

DISTRIBUTED DECISION MAKING: A MULTIAGENT DECISION SUPPORT SYSTEM FOR STREET MANAGEMENT

SUMMARY

The world around us is full of complex systems that are distributed either in conceptual or geographical sense, which consist of several elements including human beings. Distributed decision making (DDM) system consists of several distributed decision making units and DDM problem can be simply defined as "the design and coordination of connected decisions".

In this study, we suggest that, decisional effectiveness in a cooperative distributed human system can be increased by training / distributed learning efforts at the operational level and by a distributed decision support tools at higher levels. Then, we propose a new distributed decision support system based on multiagent architecture, in which human agents are also actively involved in the decision process. We plan to show the applicability of distributed decision support tools to achieve high performing distributed human systems on the street management example.

Street management is selected as an application domain, since it is a good example of distributed decision making: Decisions are made by some responsible personnel, at separate points in time and in location, with a common goal of achieving high street performance. Expertise and responsibility are divided between several different directorates and well performing streets require a coherent effort of those different units. Experiences show that existence of a central control / coordination unit is nor effective, neither efficient.

In order to establish the distributed organizational model, a functional modularity is preferred, since it was suitable for the problem in hand. Any related unit in a local authority may act as one or more of those functional agents in real life. So, intersecting "units" and "agent types" sets, 26 different "street management agent"s are produced. Street management agents realize a cooperative work with the goal of maximizing the street performance.

To set up the performance model for the street management system, fuzzy cognitive map (FCM) approach is used, because representations we need are based on expert judgments, not clear rules. Performance model calculations are done with respect to the evaluations of three experts. In our multiagent decision support system, the performance model is managed by a "performance simulator" agent; it informs "others" by simulating their special problem case.

Framework of the proposed multiagent decision support system is established according to *act*SMART model and by inspiration of the distributed decision support architectures and workflows existing in the literature. The system has two parts: a user interface which provide communication with users and a MAS which develop

proposals for decision makers. There are 11 different interfaces for 11 different kinds of users.

However, for implementation, the whole system was a too comprehensive example for the limits of this study. Thus, the applicability of the proposed model is shown on a sample part of the system; implementation is done for a sample workflow among four street management agents and two users. So, the sample program includes four agents and two interfaces. The *act*SMART structure is used to define agents as stated before.

In the sample implementation, we made several assumptions. The workflow is as follows step by step:

- The PublicListener agent gets a complaint about a paving problem on Street1ofRegion1 (R1S1) and transfers it to the PerformanceSimulator.
- Using the performance model based on FCM approach, the PerformanceSimulator agent decides that a repair work is enough and sends a message to the related RoadRepairer agent.
- After that, the information about the pavement problem appears at the user interface of the RoadRepairer agent and the related human controller decides to approve the offered repair job.
- The RoadRepairer agent sends a message to the SystemSchedular agent when the approval is submitted.
- Examining existing jobs in its agenda, the SystemSchedular agent appoints an appropriate date and replies to the RoadRepairer.
- All necessary support information then presented through the RoadRepairer interface and a message is sent back to the PublicListener agent in order to inform the former complainant user.

In order to realize the sample implementation of the proposed distributed decision support system, a software development was necessary. Java program is used for agent construction since as a general-purpose and object-oriented language, Java provides all of the base functions needed to design and implement agents.

The sample implementation software developed in Java Environment has 2 abstract classes, 7 public classes and 3 database tables. The first abstract class is the StreetAgent class; other agent classes, i.e. PublicListener, RoadRepairer, PerformanceSimulator and SystemSchedular, extends this base class. The other abstract class is the Message class, it is the super class for three message classes: CaseMessage, PerfMessage and TimeMessage. Finally, the StreetManagement database include three database tables, namely STREETINFO, AGENDA and STANDARDS.

The sample implementation was successfully realized. Thus, we could show the implementability of the decision support system we proposed: how computer-based support can provide real-time coordination and how can a decision support system help a decision maker make better decisions, e.g. calculating and representing effects on system performance. Besides, there is a wide range of alternatives for future work. One of those is a further work on the street management DSS proposed here: Sample implementation should be widened, database should be enriched, less assumptions and less neglects should be made in order to produce more useful

support systems for real life applications. Another alternative is to apply a similar structure for a different implementation field including a distributed human decision system. Future work is encouraging, since the need for distributed decision systems will increase as decision problems become more complex.

1. INTRODUCTION

In the era of globalization and communication, we have to manage more complicated systems than before, systems that consist of several computational units like people, computers, robots, etc. Traditional approaches such as central information gathering, central planning and hierarchical command-control method are not sustainable any more. Complex systems require more autonomy for local units: information and decision making initiative should be distributed through the system since a central unit can neither understand nor produce solutions for the entire system.

In this study, we suggest that, decisional effectiveness in a cooperative distributed human system can be increased by training / distributed learning efforts at the operational level and by a distributed decision support tool at higher levels. Having this idea in mind as a motivation, we propose a new distributed decision support system based on multiagent architecture. We plan to show the applicability of distributed decision support tools to achieve high performing distributed human systems on the street management example.

The study consists of six basic sections. In the first section, the concept of distributed decision making is introduced and the related implementation focused research field of multiagent systems is explained. A literature review about the subject is also represented. The second section introduces concepts of decision support, group decision support and distributed decision support. The third section is dedicated to explanation of the Fuzzy Cognitive Map approach, which is going to be used in the model given in Section 4. In the forth section, first, the rationale and the aim of this thesis study are stated, then the proposed distributed decision support model is explained. The fifth section includes implementation of a simplified sample part of the distributed decision support system in Java environment, which has been proposed in the forth section. Finally, in the sixth and last section, conclusions derived from the study are represented.

2. DISTRIBUTED DECISION MAKING AND MULTIAGENT SYSTEMS

In this section, the concept of distributed decision making is introduced and the related implementation focused research field of multiagent systems is explained. A literature review about the subject is also represented.

2.1. Organizations and Organizational Decision Making

Although there is no single definition of organizations that is uniformly agreed to, in general, organizations are characterized as structures that are:

- comprised of multiple agents (human, artificial or both),
- engaged in one or more tasks,
- goal directed,
- able to affect and be affected by their environment,
- having knowledge, culture, memories, history, and capabilities distinct from any single agent,
- having legal standing distinct from that of individual agents **(Carley and Gasser, 1990)**.

Management can be defined as the union of decision making activities and the implementation efforts for the decisions made. Studies on decision making have their origins in the seventeenth century; however, they reach a point of maturity in the mid and late 1940s **(Doyle & Thomason, 1999)**. Two frontiers, who made big contributions to organizational decision making, should be mentioned here. Chester I. Barnard, one of the first researchers on decision making, argued that understanding management functions could be possible only by studying decision making activities at individual, group and organization levels **(Wolf, 1995)**. In fact, if we accept that the history of managerial decision making has three points of views: "intuitionalism", "rationalism" and "bounded rationalism", Barnard could be an intuitionalist. From "bounded rationalism" point of view on the other hand, Herbert A. Simon agreed on studying organizational decision making and advocated a systems approach to management, which was based on the decision-making process

(Pindur, 1995). Finally, "Rationalism" point of view stands for quantitative work on decision making and most of the time, it lacks the organizational dimension.

Koopman and Pool (1990) determine eight organizational decision making models from the existing literature:

2.1.1. The Classical / Rational Model

This model describes the ideal way of acting in decision making situations and offers a basis for quantitative disciplines such as econometrics and statistics. Decision making is regarded as a logical process where decision makers try to maximize their objectives in an orderly series of steps. It is assumed that the decision maker is aware of all alternatives. Several models have been developed as alternatives to the normative rational model which are based more on the actual behavior of decision makers in organizations.

2.1.2. The Information Model

One of the first adaptations of the rational model to the practice of decision making is about the information processing capacity of decision makers. Human cognitive capacity is restricted, so it uses only a limited part of the existing information and its information search is also restricted with time and money constraints. A "satisfying solution" is enough. In fact, many highly unrealistic assessments of reality can exist in a small group of relatively isolated decision makers. Preferences and desired solutions are often present before the search for information starts. Information is then used to support, sell or defend these preferences.

2.1.3. The Organization Model

Limited personal information processing and analysis of the environment can be reduced by creating a proper sort of organization. However, there are also a number of limitations within the organization. There are often several changing goals and decision makers have insufficient control. In this case, decision makers develop a simplified model of the reality with goals and solutions.

On the other hand, organizations are composed of several units and they are not "unitary" structures. The task division in organizations leads to a limited view of

problems and solutions. Decision making in organizational context has four characteristics:

- Conflicts among units are not really solved but reduced to an acceptable level by means of several procedures. The organization looks for satisfying rather than maximal results and focuses on the achievement of contrasting goals.
- The organization avoids uncertainty by bringing the environment under control as much as possible through agreements and contracts.
- Organizations only seek solutions to specific problems. When the problem is solved, the search stops.
- Organizations learn from their experiences. Search procedures and goals are adapted in the course of time.

2.1.4. The Bureaucratic Model

Decision making processes in organizations are subject to the structural division of authority and tasks. Most decisions take place within the network of accepted rules and agreements. The delegation of authority may be more of less hierarchical and it is not fixed. Agreements can be even made about the procedure to be followed for each individual decision making process.

2.1.5. The Garbage Can Model

In this model, organizations are seen as “organized anarchies”, characterized by unclear or inconsistent goals, a technology that is little understood by members and a highly variable member participation. In order for decision making to progress, it is essential that the organization manages to attract sufficient attention from the members to solve the problems at hand.

2.1.6. The Political / Arena Model

Conflict and the way of handling it are than focus points of the model. The primary criterion is not the right decision but a decision acceptable to all. The division of power within the organization is critical. In order to play the game successfully, a person must not only have sufficient power, but also the desire and the capacity to use it.

2.1.7. The Participation Model

The focus in this model is the attention of lower level organization members in decision making. A number of extensive studies have yielded little consistent support for the expected positive effects of participation. Solutions to the lack of consistent findings were sought in more complicated contingency models and related studies showed that participation is generally useful only if it makes a functional contribution.

2.1.8. The Dynamic Phases Model

In this model, the emphasis is on the process rationality. Assuming that, in strategic decisions, logical and political problems should be brought together, the question of how this should take place arises. In the literature, several phase models exist. Several supporting processes run parallel to the well-known central decision making phases: decision making control processes, communication processes and political processes. The picture can be further complicated by the effect of several dynamic factors like delays and interruptions.

2.1.9. The Model of Meaning

This model emphasizes the ambiguity of the information relevant to the decision and directs attention to the manner in which problems are defined and to the amount of consensus.

2.2. Distributed Decision Making

The earliest underlying models and algorithms developed for solving several real world problems, such as scheduling, inventory management and highway management, were centralized in nature. In this paradigm, data from one or more sources are collected at a central site and a single processor utilizes it to compute the system-wide decisions through sequential execution. However, the world around us is full of complex systems that are distributed either in conceptual or geographical sense, which consist of several elements including human beings. For many systems, like real-time internal payments processing, a centralized algorithm is not even realizable **(Ghosh, 2001)**. The interest in the distributed problem solving system

stems from the fact that many real-world problems can be better solved using a set of cooperative agents rather than a single agent **(Rogova&Menon, 1998)**.

Distributed decision making (DDM) problem can be defined as the coordination set up among agents, each of which has a restricted model of the overall problem **(Brehmer, 1990)**. That means,

- decisions are made at different points of the general system;
- those decisions are based on the partial and insufficient capabilities of the single decision makers, but effect the overall system performance;
- thus, a coordination mechanism among agents is needed.

Brehmer (1990) also states that, it is more suitable to characterize a system with DDM when a central decision or control unit does not exist. A system allows distributed decision making to the extent that it lacks a centralized decision maker and that it permits different units in the system to function autonomously. Conversely, a system may be said to require distributed decision making to the extent that its overall purpose can be realized only if some (or all) units in the system are required to act on their own. **(Brehmer,1990)** In summary, it is "the design and coordination of connected decisions" **(Schneeweiss, 2003)**.

Distributed decision making include a process of decision making by and usually for multiple agents, which often have different preferences and incomplete information. Distributed decision making is useful because many situations are not zero-sum games and the social welfare can be increased by joint decision making that leads to more desirable coordinated actions **(Weiss, 1990)**.

Brehmer (1990) emphasizes the need of distributed decision making for human organizations and indicates that there are at least three reasons for the current interest in distributed decision making:

- Complex character of the modern world: In the traditional approach, in order to handle with a complex problem, the original task is partitioned into more tractable subtasks. However, in modern work systems, character of the work people do is cognitive rather than physical, and one person cannot form the complexity of the entire system. So, decision making should be distributed among workers.

- Development of modern information systems: Distribution of tasks increases the need for coordination. In the traditional approach, a supervisor exists, who has virtual monopoly on information – information was centralized. Modern information technology has largely abolished the need for such a hierarchical organization. It makes possible distributed decision making at lower levels and provide more autonomy.
- Changed conditions on the modern battlefield: Military systems have always been characterized by distributed decision making. Current command systems are too slow with respect to modern communication and information tools. On the other hand, hierarchical command system requires complete communication and that is now less certain. Those leads to an interest in distributed decision making under limited communication or no communication.

2.3. The Extent of Distributed Decision Making

DDM is a relatively new but rapidly developing area in general decision theory. The concept fits many diverse situations, in a variety of study fields and application domains. Such diverse that it defines cases from market mechanisms to industrial robots, from hierarchical optimization problems to railway networks.

After a careful examination of the existing literature, **Schneeweiss (2003)** concludes that DDM is such a broad area that several diverse disciplines concern with different aspects of the subject such that:

- hierarchical optimization
- multistage deterministic programming
- multi-level stochastic programming
- distributed artificial intelligence
- principal agent theory
- contract theory
- auction theory
- negotiations
- group decision making
- hierarchical production planning

- supply chain management
- managerial cost accounting.

He also states that, these areas are part of different disciplines like applied mathematics, operations research, computer sciences, economics, production and operations management, management accounting, organization theory, etc., in many cases.

2.4. Classification of Distributed Decision Making Systems

As asserted in the previous section, distributed decision making is related with a wide range of disciplines and includes a variety of cases. A classification study, based on cooperation and information symmetry aspects, is belong to **Schneeweiss (2003)**.

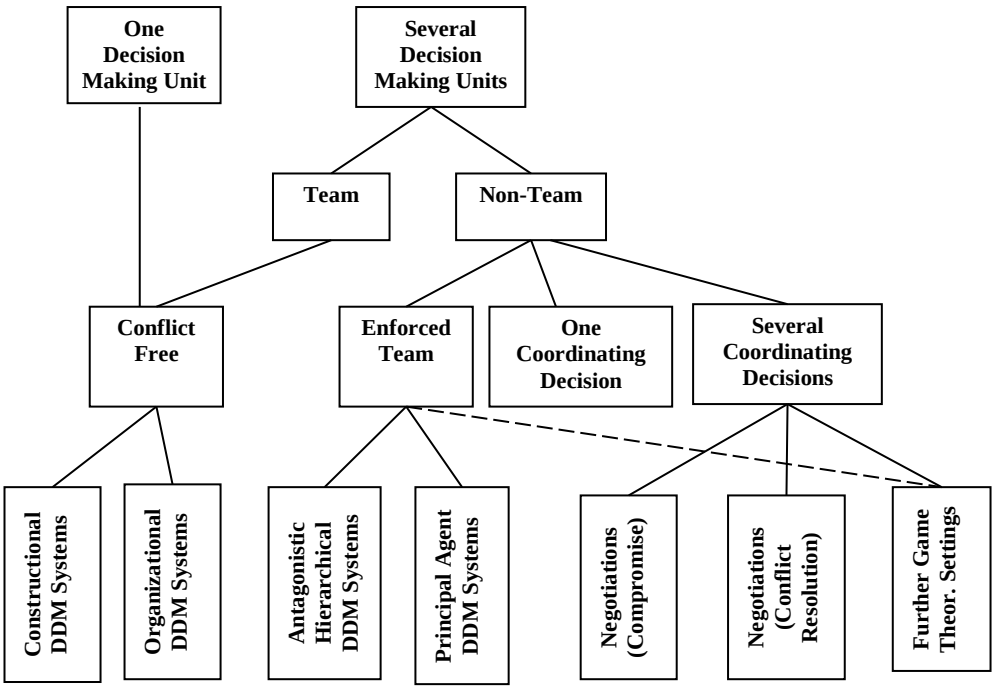

Figure 2.1. A Classification of Distributed Decision Making Cases **(Schneweiss, 2003)**

The classification of **Brehmer (1990)** is a more dimensional one, but it is also more abstract. His taxonomy has four dimensions:

- the level of decomposition: it can be very fine (statement level) to coarse (problem level) ;
- the distribution of expertise;

- the methods for achieving distributed control: as for control, distributed systems vary along three dimensions, namely coordination, organization and dynamics;
- the process of communication.

2.5. Modeling Approaches to Distributed Decision Making

Rasmussen (1990) implies that, distributed decision making establish a unified framework, which covers separate research areas such that management-organization, cognitive phenomena and systems control. Thus, distributed decision making concept can be studied by three different models based on different approaches: Cognitive Engineering Approach, Control Theory Approach and Organization and The Management Science Approach **(Rasmussen, 1990)**.

2.5.1. Cognitive Engineering Approach

The aim of cognitive engineering approach is modeling human cognitive functioning in a complex cooperative work environment to serve as a basis for design of information and decision support systems.

The modern work environment, which is restructured by the advanced information technology, require distributed decision making since simultaneous, integrated activities replace slow sequences of separate processes. Such an organization needs decision support systems, centralization of databases and advanced communication systems, which can only be designed using a reliable model of the needs of the individual agent in the new organization. To sum up, for design of decision support systems, the structure of complex socio-technical systems should be modeled from at least four points of view:

- The content and structure of the basic work domain;
- The structure of the decision making task, i.e. the control function, implemented to obtain concerted functions in this work domain;
- The level of cognitive control of the decision agents and their cognitive resource profile;
- The structure of the work organization, i.e. of the allocation of control functions to decision making agents and the resulting social organization.

2.5.2. Control Theory Approach

It is interesting to compare the concepts involved in the cognitive approach to those of control theory. The essential characteristics of hierarchies are taken to be: vertical decomposition, priority of action or right of intervention and (vertical) performance dependence. Different notions of hierarchies are important for process control are in parallel to the distinctions made between:

- Levels of description or abstraction (strata);
- Levels of decision complexity (layers);
- Organizational levels (echelons).

2.5.3. Organization and The Management Science Approach

For the two approaches above, an organization of "decision makers" and "controllers" discussed. It is suitable to relate them to the theories developed within management and organization science. Two fundamental models exist in the literature namely "rational" and "natural system" models, which are later related to "closed-system" and "open-system" strategies of analysis. However, most of the work about organizations in the literature are based on closed-system assumptions.

2.6. Literature on Distributed Decision Making

Distributed decision making is a diverse subject as implied in section 2.3., so the related literature work is huge. Thus, it seems to be meaningful to make a simple classification and indicate some samples in order to give an idea.

The literature on distributed decision making can be examined following the classification given below. A summary is represented in Table 2.1., including just some sample studies since it is almost impossible to mention here all of the related work.

- General Theory
- Computational Organization Theory (COT) : It has a "social" point of view, uses mathematical and computational methods to study both human and automated organizations as computational entities.

- Cooperative Problem Solving : Stands for the distributed solving methods developed for problems that also have central solutions, e.g. optimization problems.
- Distributed Artificial Intelligence and Multi Agent Systems (DAI and MAS) This area is dedicated to distributed artificial computing systems, it will be further explained in the next section.

Table 2.1. Literature on Distributed Decision Making

GENERAL THEORY	
Schneeweiss (2003)	A general framework for DDM
Rasmussen (1990)	Modeling DDM
Ghosh (2001)	Asynchronous DDM
Examining DDM on Sample Systems	
Schneeweiss (2003)	DDM in Supply Chain Management
Rogalski (1990)	DDM in Emergency Management
Campbell et al. (2000)	Distributed human decision making in Traffic Flow Management
COMPUTATIONAL ORGANIZATION THEORY	
Sumpter and Broomhead (1998)	Honey bee colonies, formalization of relations between worker and society
Hannoun et al. (1998)	Dependence relations between roles in multi agent systems
Leplat (1990)	Organization for realizing common tasks
Schmidt (1990)	Structural framework for cooperative work
Kok and Vlassis (2003)	Coordination graphics in distributed decision making of robotic agents
Lemaître (1998)	A new approach to multi agent organizational design (group focus instead of agent focus)
Verhagen (1998)	The effects of organizational structure and communication on organizational problem solving
Kraus (2001)	Concensus techniques in multi agent systems
Hashimoto (1998)	Category structure in communication, words clustering
Zhang (1994)	Implementation and comparison of several cognitive models on multi agent systems
Servat et al. (1998)	Artificial cognitive emergence / social learning

Castelfranchi (1998)	The importance of cognitive emergence for modeling and simulation of social phenomena
Ramos et al. (2005)	Swarm Intelligence model formation for collective intelligence
COOPERATIVE PROBLEM SOLVING	
Yakoo and Ishida (1990)	"search" algorithms for decentralized decision making agents
Rusmevichientong and Van Roy (2000)	the relationship between the number of agents and the amount of information the agents should get in decentralized decision making
DISTRIBUTED ARTIFICAL INTELLIGENCE and MAS	
Parunak (1990)	Distributed artificial intelligence in industrial systems
Singh (1994)	A formal framework for multiagent system design
Schleiffer (2005)	Fundamental issues about intelligent agents
Multiagent System Implementations	
Cohen (2003)	Development of intelligent software components which run at distributed locations on the U.S. electrical transmission and distribution network
Barber et al. (2001)	Applying a Sensible Agent system to provide Chemical-Biological detectors and responders
Multiagent Simulation	
Swaminathan et al. (1998)	Multiagent simulation of supply chain dynamics
Parunak et al. (1998)	Agent-Based modeling vs. Equation-Based modeling for supply chain simulation
Antona et al. (1998)	Multiagent simulation of economic theory of renewable resource management
Rouchier et al. (1998)	Test of hypothesis on non-merchant economy using multiagent simulation
Doran (1995)	Prospects for using Distributed AI techniques to support the computer simulation of societies

2.7. Multiagent Systems

Schneeweiss (2003) implies that multiagent systems (MAS) of distributed artificial intelligence (DAI) contributes to applied DDM by focusing on the implementation aspect. MAS is a distributed computing system and consists of a number of connected intelligent computational units **(Singh, 1994)**. It uses distributed and parallel computation; in a multi-agent system, autonomous agents live and interact with other agents and the environment. In the multiagent case, what an agent experiences as a result of its actions depends not only on how it acts but also on how its neighbors act.

Several computer engineering research activities developed software products for the implementation of information systems based on agent architectures. Agent-based architectures have been used in many software applications, applied to a variety of topics (control, information management, communication, etc.) and scenarios (aerospace, medicine, military, etc.). It seems to be a useful tool to turn DDM theories into practice.

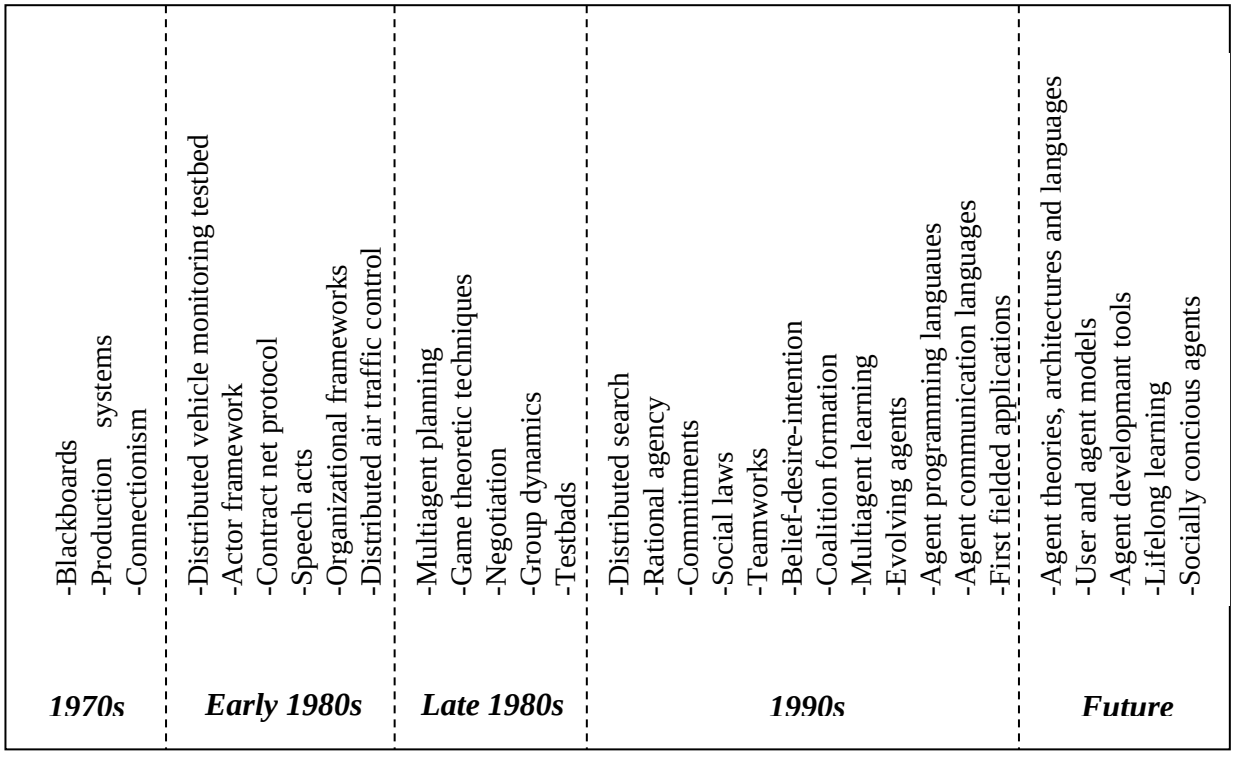

Figure 2.2. A chronological view of multiagent system research **(Sen, 1997)**

Research on Multiagent Systems dates back to the late 1970s. After a comprehensive literature review about multiagent systems, **Sen (1997)** states the chronology of multiagent systems research given in Figure 2.2.

Many frameworks have been developed for developing multiagent systems, formalizing what is an agent and how it interacts. (e.g. **Singh, 1994; Liu, 2001; d'inverno and Luck, 2004**) The SMART framework of **d'inverno and Luck (2004)** is taken as basis in this thesis; thus, concept related to multiagent system structure are explained below within this framework.

2.8. The SMART Framework and *act*SMART Implementation Environment

The "Structured and Modular Agents and Relationship Types" (SMART) framework, proposed by **d'inverno and Luck (2004)**, is essentially a four-tiered hierarchy comprising entities, objects, agents and autonomous agents, where agents viewed as objects with goals and autonomous agents as agents with motivations. *act*SMART is an implementation environment based on the SMART framework, the relationship is given in Figure 2.3.

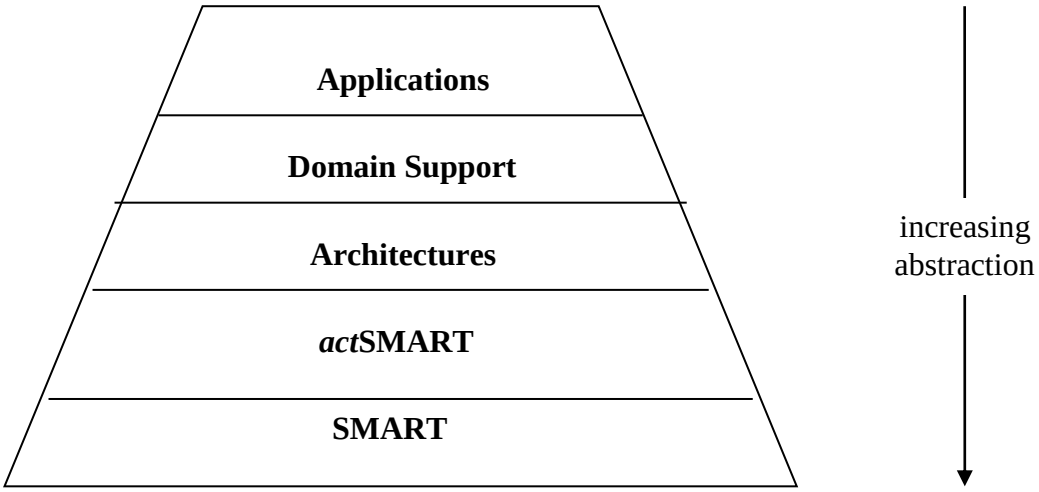

Figure 2.3. From SMART to Applications **(d'inverno and Luck, 2004, p.204)**

2.8.1. SMART Agent Framework

SMART framework consists of a four-tiered hierarchy described in Figure 2.4. The basic idea underlying this hierarchy is that an environment consists of entities, some of which are objects. Of this set of objects, some are agents and of these agents some

are autonomous agents. Formal definitions of SMART are given below **(d'inverno and Luck, 2004)**.

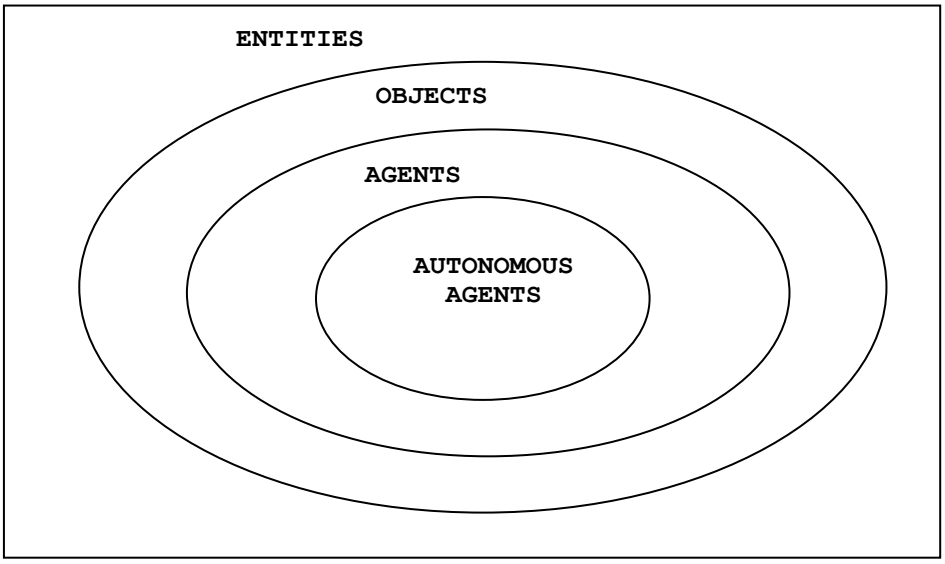

Figure 2.4. Entity hierarchy overview **(d'inverno and Luck, 2004, p.17)**

Definition 2.8.1.1. An *attribute* is a perceivable feature.

Definition 2.8.1.2. An *action* is a discrete event that can change the state of the environment when performed.

Definition 2.8.1.3. A *goal* is a state of affairs to be achieved in the environment.

Definition 2.8.1.4. A *motivation* is any desire or preference that can lead to the generation and adoption of goals and that affects the outcome of the reasoning or behavioral task intended to satisfy those goals.

Definition 2.8.1.5. An *entity* is something that comprises a non empty set of attributes, a set of actions, a set of goals and a set of motivations.

Definition 2.8.1.6. An *object* is an entity with a non-empty set of actions.

Definition 2.8.1.7. An *agent* is an object with goals.

Definition 2.8.1.8. An autonomous agent is an agent with motivations.

The above definitions are explained with an autonomous robot example in Table 2.2.

Table 2.2. Example: Descriptions of a Robot in the Agent Framework **(d'inverno and Luck, 2004, p.31)**

Schema	**Variable**	**Robot**
Entity	Attributes	{red, large, heavy,...}
Object	Capabilities	{lift, carry, hold,...}
Agent	Goals	{fix_tyres}
Autonomous Agent	Motivations	{achievement, hunger,...}

2.8.2. Agent Relations

The rationale for interconnecting computational agents and expert systems is to enable them to cooperate in solving problems, to share expertise, to work in parallel on common problems, to be developed and implemented modularly, to be fault tolerant through redundancy, to represent multiple viewpoints and the knowledge of multiple experts, and to be reusable **(Huhns and Stephens, 1990)**.

The SMART framework defines a multiagent system as follows **(d'inverno and Luck, 2004)**:

Definition 2.8.2.1. A *multiagent system* is any system that contains:

- two or more agents;
- at least one autonomous agent; and
- at least one relationship between two agents where one satisfies the goal of the other.

SMART defines eight types of agent relation: "dengages", "engages", "indengages", "owns", "downs", "uowns", "sowns" and "cooperates". Definitions of only two relations will be given here:

Definition 2.8.2.2. A *server agent* is an agent that is not autonomous.

Definition 2.8.2.3. A direct *engagement* between an agent and a server-agent exists when, through the direct information of the first agent, the server agent has adopted a goal of the agent.

Definition 2.8.2.4. An agent A is said to *cooperate* with another agent B, if they are both autonomous and A has autonomously adopted the goal of B.

In fact, agent relations are based on "communication" and "interaction" protocols. Conversation protocols enable agents to exchange and understand messages.

Interaction protocols enable agents to have conversations, where goals structure exchanges of messages. There are both binary and n-ary protocols. A binary protocol involves a single sender and a single receiver, whereas an n-ary protocol involves a single sender and multiple receivers. A protocol is specified by a data structure with the following fields **(Huhns and Stephens, 1990)**:

- sender
- receiver(s)
- language in the protocol
- encoding and decoding functions
- actions to be taken by the receivers.

Such a well-known protocol is the knowledge query and manipulation language (KQML), it is still a work in progress and its semantics have not been completely defined. The specialty of KQML is that all information for understanding the content of the message is included in the communication itself.

```
(KQML-performative
                    :sender      <word>
                    :receiver    <word>
                    :language    <word>
                    :ontology    <word>        // the vocabulary
                    :content     <expression>  // the message itself    ...)
```

Figure 2.5. KQML Communication Protocol

In practice, agent interaction occurs by sending and receiving messages through a communication network. Table 2.3. include a list of message types that are derived from the speech-act theory **(Huhns and Stephens, 1990)**.

Table 2.3. Interagent Message Types **(Huhns and Stephens, 1990, p.86)**

Communicative Action	Illocutionary Force	Expected Result
Assertion	Inform	Acceptance
Query	Question	Reply
Reply	Inform	Acceptance
Request	Request	
Explanation	Inform	Agreement
Command	Request	
Permission	Inform	Acceptance
Refusal	Inform	Acceptance
Offer / Bid	Inform	Acceptance
Acceptance		
Agreement		

Communicative Action	Illocutionary Force	Expected Result
Proposal Confirmation Retraction Denial	Inform	Offer / Bid

2.8.3. *act*SMART Agent Implementation Environment

As stated before, *actSMART* is an implementation environment based on the SMART framework. Figure 2.6. shows the basic principles. A shell acts as a container in which components are placed. It manages the sequence in which components execute and the flow of information between components. Control policies related to the permissions of an agent in a specific environment are defined within the shell to make them independent of the agent architecture. Finally, attributes describing the agent as a whole are defined as a part of the shell.

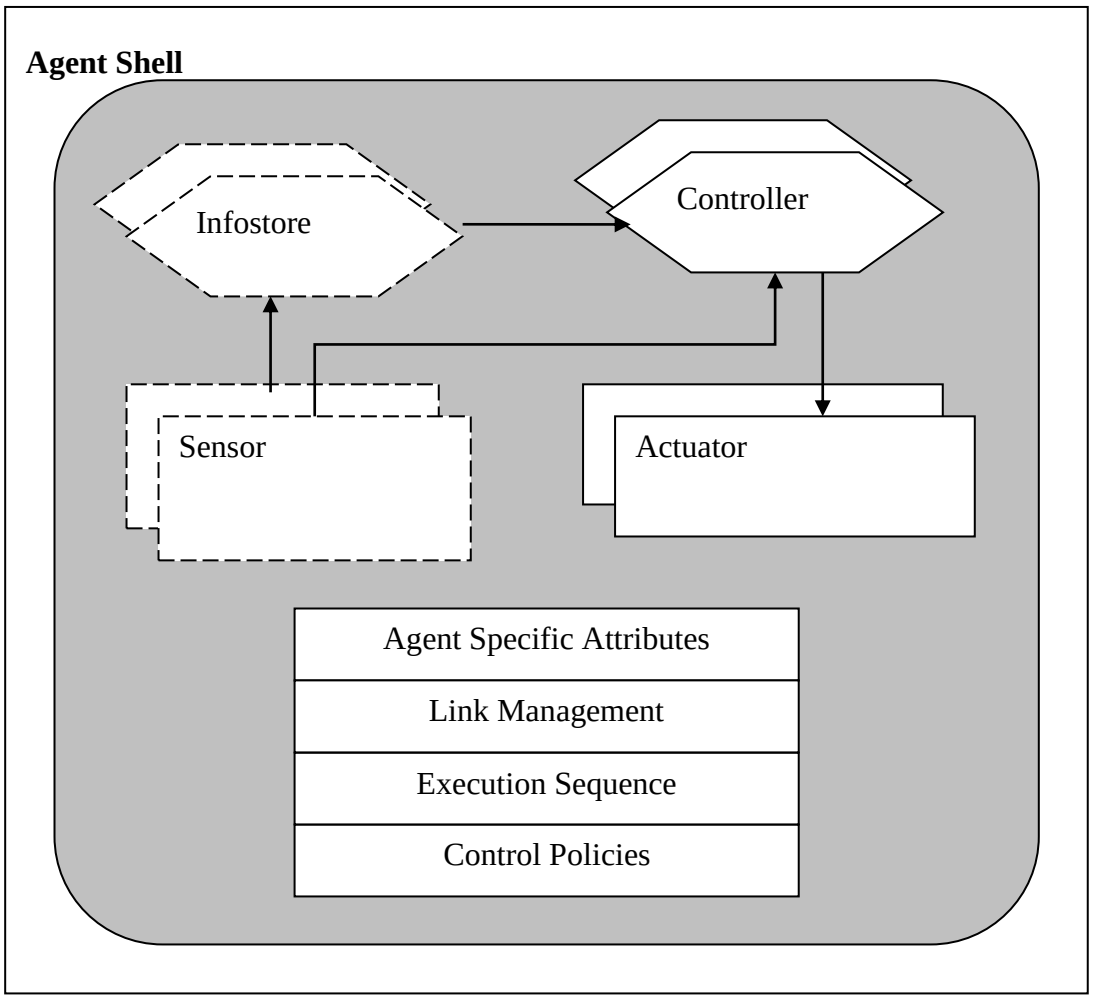

Figure 2.6. The *act*SMART agent shell **(d'inverno and Luck, 2004, p.206)**

Components of the agent shell are grouped into four categories and each component has two types of attributes: “stateless attributes” refer to persistent characteristics whereas “situational attributes” refer to attributes describing the current state of the component **(d’inverno and Luck, 2004)**:

- “Sensors” receive information from the environment representing the perception capabilities.
- “Infostores” store the models the agent uses to reason.
- “Actuators” perform actions that affect the environment.
- “Controllers” are the main decision making components.

The original *act*SMART implementation was done in Java environment, using Jini and XML technologies.

3. DISTRIBUTED DECISION SUPPORT

In this section, concepts of decision support, group decision support and distributed decision support are introduced.

3.1. Decision Support Systems and Group Decision Support

Although there is no single, agreed-upon definition of decision support systems, **Silver (1991)** makes a broad definition that includes all:

"A Decision Support System (DSS) is a computer based information system that affects or is intended to affect how people make decisions."

In fact, the concept includes its definition in its name. The word "Decision" leads to a problem oriented concept, "Support" implies a decision aide / help mechanism, and finally "System" refers to something that is a combination of computer, data, algorithms and models.

The need for computerized mechanisms for decision support comes from the well known limits of human knowledge processing **(Chen, 2000)**: cognitive limits, economic limits, time limits and competitive demands. A DSS does not replace human decision makers but just helps them in decision making. Various kinds of support can be provided, including **(Chen, 2000)**:

- User Alert (alerting the user to a decision-making opportunity or challenge);
- Problem Recognition (recognizing problems that need to be solved as part of the decision making process);
- Problem Solving;
- Facilitating / Extending the User's Ability to Process Knowledge (e.g. acquire, transform, explore the knowledge);
- Stimulation (stimulating the user's perception, imagination, or creative insight);
- Coordinating / Facilitating Interactions (among participants in multiparticipant decision makers);

- Others.

In fact, two characteristics of human decision making process are central to understanding DSS **(Silver,1991)**:

- Decision making is not a point event. It does not occur only at the moment of choice, it is a complex sequence of differentiated activities occurring over time.
- Decision making is not monolithic. Several different ways can be followed to arrive at a decision.

The core of the decision support problem can be stated by the following simple semi-formal model **(Cuena and Ossowski, 1990)**.

- *A set of World States* **S** *:* The relevant state of the world with respect to a decision support problem is given by the values of the state and control variables of the managed system.
- *A set of Ideal States* $\mathbf{S}^+$ *and A set of Undesired States* $\mathbf{S}^-$, *where* S^+, $S^- \subset S$: Ideal and undesired states determine configurations of values for state and control variables that shall be achieved or are to be avoided respectively.
- *A notion of Preference* $<$ *on states* : The notion of preference expresses the closeness of a state to another. It can be expressed either qualitatively or quantitatively.
- *A set of Control Actions* **Π** : Control actions can be performed on the system to be controlled which change the values of certain control variables directly.

The objective of the DSS is, then, to generate sets or sequences of control plans π, to transform the current world state *s* into a state *s'* that is "as close as possible" with respect to $<$ to some ideal state s^+ and "as far away as possible" with respect to $<$ from any undesired state s^-.

Sage (1991) further explains that a DSS has three principal components and an appropriate DSS design framework would consider each of these three component systems and their interrelations and interactions:

- *Data-base Management System (DBMS):* A DBMS is necessary in order to record data within a systematic data model. In almost every instance in which there are multiple decision makers, a need will exist for personal, local and system-wide data bases.

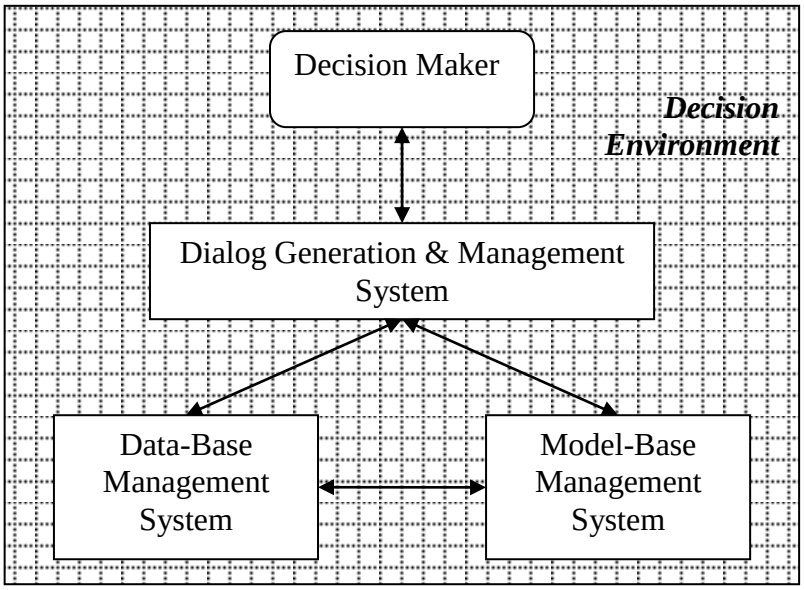

Figure 3.1. Generic Components in Decision Support Systems **(Sage, 1991, p.7)**

- *Model-base Management System (MBMS):* Through the use of MBMS, we are able to provide for sophisticated analysis and interpretation capability in a decision support system.
- *Dialog Generation and Management System (DGMS):* The DGMS portion of a decision support system is designed to satisfy knowledge representation, control and interface requirements of the DSS.

Research in decision support area was also directed to systems that support a group of people aiming to accomplish a common goal. Those systems have a variety of names, including **(Silver, 1999)**: Group Decision Support Systems, Computer Supported Cooperative Work, Electronic Meeting Systems, Groupware and Coordination Theory. Different methods used can be summarized on a time-location matrix. (Figure 3.2.)

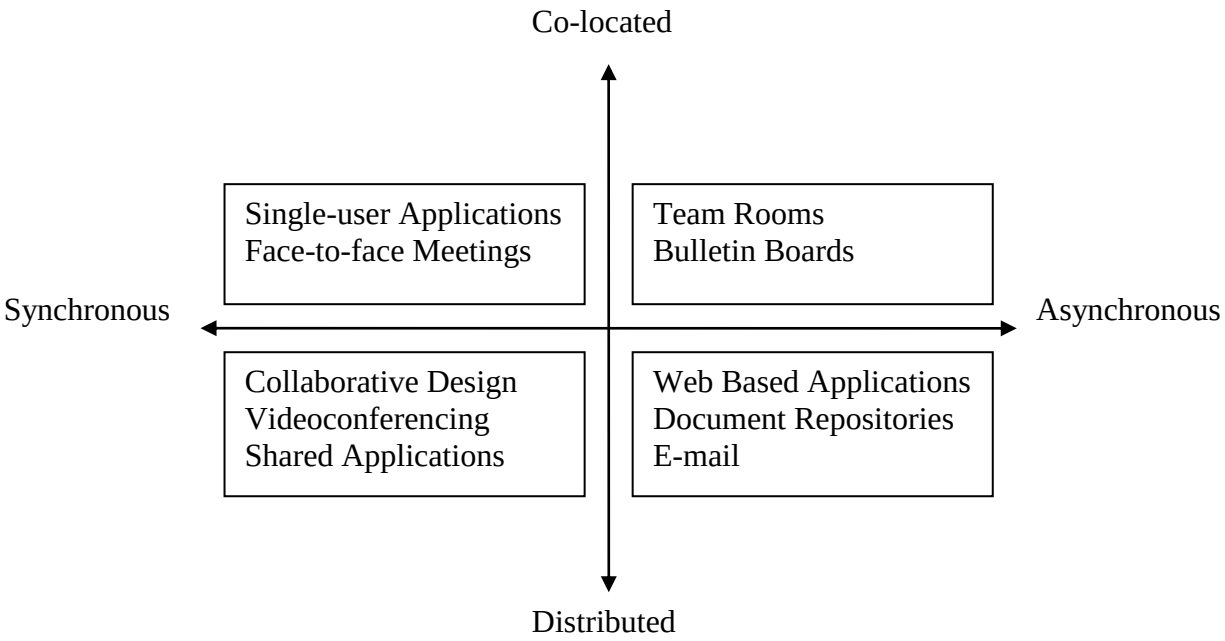

Figure 3.2. Time – Location matrix categorizing different methods used for collaboration **(Palmgran, 2004)**

3.2. Distributed Decision Support Systems

As explained previously, a decision support system (DSS) can simply be defined as a computer based activity that supports the process of decision making. As **Chen (2000)** suggests, "the task of decision support can be carried out by constructing intelligent agents". In practical use, a DSS may be one of the most appropriate tools to provide coordination among decision agents of a distributed human system, as well as providing the classical assistantship at knowledge processing and problem solving.

Modern Decision Support Systems (DSS) not only store large amounts of decision-relevant data, but also aim at assisting decision-makers in exploring the meaning of that data, so as to take decisions based on understanding. To this end, a distributed approach to the construction of DSS has become popular: decision-support agents are responsible for parts of the decision-making process in a (semi-) autonomous (individually) rational fashion **(Ossowski et al. 2004)**. Interest in the possibility of building complex problem solving systems as groups of cooperating experts has led us to develop multiagent DSSs capable to run on servers that can support a large group of users (clients) who communicate with the system over the network. **(Shaalan et al., 2004)**

Many mission-critical, decision-making situations happen in dynamic, rapidly changing, and often unpredictable distributed environments. Military, governmental, and medical contexts are examples of such situations, which can be characterized by highly decentralized, up-to-date data sets coming from various sources. Unlike other decision making tools, DSSs designed for such situations are challenged by the need to access this decentralized data at any time, from anywhere, under tight time constraints **(Gachet, 2002)**.

There are numerous research and application study on computer supported cooperative work and distributed / multiagent decision support systems. For example, Ossowski et al., 2004; Gachet, 2002; Qureshi, 2000; Goddard et al., 2002; Cuena and Ossowski,1990; Aguirre et al, 2001; Nourani, 1999; Shaalan et al, 2004; Biró, 1994; Gao et al.,2003; Laichour et al.,2002; Ydstie, 2004.

However, **Gachet (2003)** argues that although the DSS community generally shows interest in distributed computing when building decision support systems, the use of distributed technologies often remains limited to extension services added to traditional, local DSS functionalities. Many systems labeled today as "distributed DSSs" mostly remain centralized applications propagating their results to the edges of the network in a client/server manner. A distributed DSS would be closer to multiparticipant, human centered processes. Our literature review does not completely support this argument; but, we can further argue that, asynchronous and human involved decisions at the edges of the system is lacked many times probably because of the special problem structures studied. Computation units of distributed decision support system examples such as electrical power stations and traffic network sensors do local decision making activity but do not include human involvement. On the other hand, group decision support systems are focused on producing a common decision rather than supporting decisions made at the edges of the system.

Consequently, as a distributed decision support system, we imagine a system neither producing decisions and dictates them to people nor just providing a tool to let people communicate. We support a system that includes distributed but cooperating computer and human agents both of whom are actively involved in the decision process. Our model is explained in detail in section 5.

3.3. An Agent Architecture for Distributed Decision Support System Design

It was mentioned before that distributed decision support can be achieved by constructing a society of decision support agents. **Cuena and Ossowski (1990)** propose an agent model, which comprises features that are necessary to describe different case studies from a unifying view.

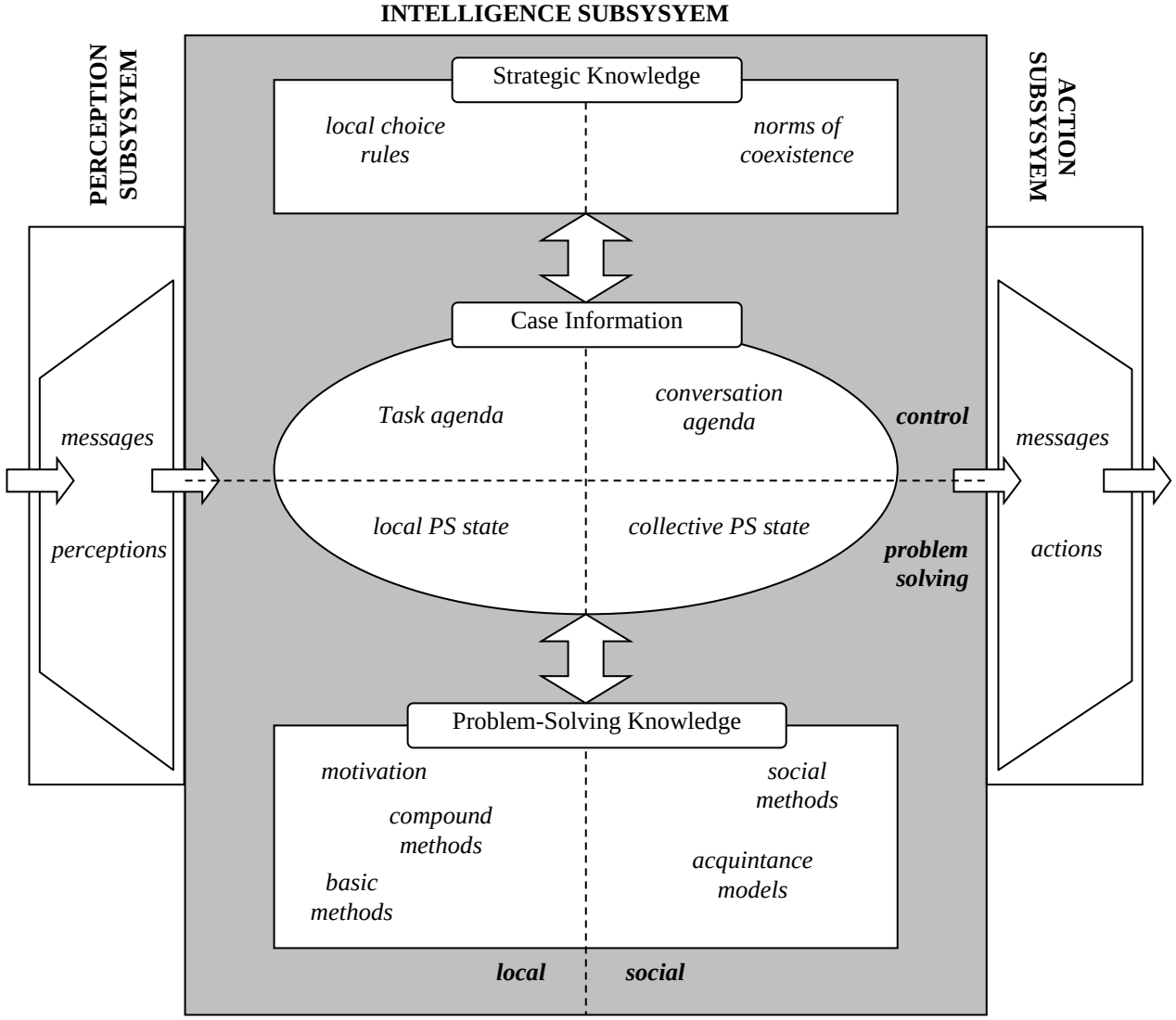

Figure 3.3. An agent architecture for a decision support agent **(Cuena and Ossowski, 1990, p.68)**

The proposed model is illustrated in Figure 3.3. It is built around three major components:

- A *perception subsystem* allows the agent to be situated in the environment by data acquisition and in the society by perceiving agent messages.

- An *intelligence subsystem* manages the different aspects of information processing as well as individual and social problem solving. So, it includes an information model and a knowledge model.

  The information model is used in order to store the dynamic beliefs of the agent about the world itself and about others. The perception subsystem writes data on it according to perceptions and received messages, when the knowledge of the intelligence subsystem is enacted, the information model is modified, while the action subsystem reads from it.

  Agent knowledge can be classified from two perspectives. On the one hand, problem-solving knowledge, which is used to determine which actions to take and on the other hand, strategic knowledge that helps to choose among different options (the tasks or conversations) that the intelligence subsystem is to process next.

- An *action subsystem* enacts the plans produced by the intelligence subsystem, displaying messages to the control personnel, sending messages to other agents or activating robotic effectors.

Cuena and Ossowski (1990) explain the mode of operation of the agent model by a simple reasoning cycle which has the following steps:

- the perception subsystem captures percepts and messages from other agents, and updates the information model accordingly;
- the conversation agenda is updated and reordered in accordance with the social strategic knowledge. As a result of the selection of some conversation, new tasks are added to the task agenda;
- the motivation is matched against the information model and eventually more new tasks are created on the task agenda;
- using the local strategic knowledge, the task agenda is reordered and some tasks are chosen for execution;
- for every task two approaches are to be followed:
  - the local problem solving approach where, using the knowledge about relation between tasks and methods, a method is chosen for execution. Usually, basic methods are preferred to compound methods, and the latter are given priority over social methods;

- the delegation approach, if in the previous process no method is available in the internal problem solving knowledge to cope with a task. In this case, the agent consults its acquaintance models and identifies a collection of agents that may perform the required tasks. The agent then assigns the task to the most adequate agent;

- the action subsystem performs actions and sends messages as indicated by the intelligence subsystem in the information model.

4. FUZZY COGNITIVE MAPS

This section explains the Fuzzy Cognitive Map approach, which is going to be used in the model given in Section 5.

4.1. Some Concepts in Fuzzy Logic

Fuzzy logic has developed rapidly since its first presentation by Lotfi Zadeh and today it affects many disciplines, being applied to numerous cases. Such an implementation for the paradigm is in the area of cognitive modeling: Fuzzy Cognitive Map (FCM). However, it is first necessary to define some concepts on fuzzy logic before explaining the approach. The definitions in this section are adopted from **Ross (1998)**.

4.1.1.Membership

Definition 4.1.1.1. A *fuzzy set* is a set containing elements that have varying degrees of membership in the set.

Elements of a fuzzy set are mapped to a universe of membership values using a function-theoretic form. This function maps elements of a fuzzy set **A** to a real numbered value on the interval [0,1]. Then, if an element *x* in the universe is a member of the fuzzy set **A**, then this mapping is given by, $\mu_A(x) \in [0,1]$

4.1.2.Fuzzy Set Operations

Define three fuzzy sets **A**, **B** and **C** on the universe **X**. For a given element *x* of the universe, the following operations are defined on **X**:

Union $\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x)$ (4.1)

Intersection $\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x)$ (4.2)

Complement $\mu_{A'}(x) = 1 - \mu_A(x)$ (4.3)

where the symbol $\vee$ is the maximum operator and $\wedge$ is the minimum operator.

4.1.3.Fuzzy Relation

Definition 4.1.3.1. A fuzzy relation **R** is a mapping from the Cartesian space $\mathbf{X} \times \mathbf{Y}$ to the interval [0,1], where the strength of the mapping is expressed by the membership function of the relation for ordered pairs from the two universes, or $\mu_R(x, y)$.

4.1.4.Defuzzification

Definition 4.1.4.1. A lambda-cut set A_λ of the fuzzy set **A** is a crisp set, where $A_\lambda = \{x | \mu_A(x) \geq \lambda\}$ and $\lambda \in [0,1]$.

Consider a fuzzy relation **R**, where each row of the relational matrix is considered a fuzzy set. Hence, a fuzzy relation can be converted to a crisp relation defining $R_\lambda = \{(x, y) | \mu_A(x, y) \geq \lambda\}$ as a lambda-cut relation of the fuzzy relation.

Defuzzification is the conversion of a fuzzy quantity to a precise quantity. There are several defuzzification methods in the literature including:

- max-membership principle,
- centroid method,
- weighted average method,
- mean-max membership,
- center of sums,
- center of largest area,
- first (or last) of maxima.

The Centroid Method procedure is the most prevalent and physically appealing of all the defuzzification methods. It is given by the algebraic expression,

$$z^* = \frac{\int \mu_C(z) \cdot z dz}{\int \mu_C(z) dz} \qquad (4.4)$$

where C is the union of fuzzy membership functions.

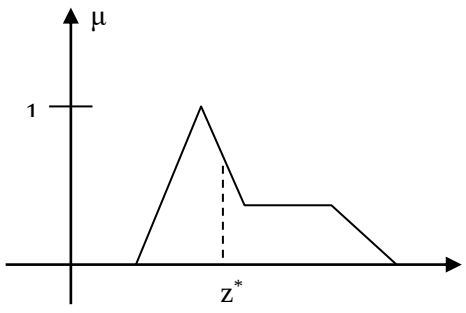

Figure 4.1. Centroid defuzzification method

4.2. Fuzzy Cognitive Map Approach

Fuzzy cognitive maps (FCM) are "a modeling methodology for complex systems which originated from the combination of fuzzy logic and neural networks" **(Xirogiannis et al., 2004)**. FCMs are used for storing uncertain causal knowledge. They consist of nodes and connections, where each node represents a fuzzy set and connections exhibit the direction and the strength of the relations between nodes. (Figure 4.2.)

FCMs are preferable quantitative tools in domains involving complex networks of casual relationships, particularly feedback, and where hard quantitative measures of influences are not available. They are easy to construct, allow users to rapidly compare their mental model of the system with the real world and because of their fuzzy elements, extremely forgiving of uncertain information **(Mohr, 1997)**.

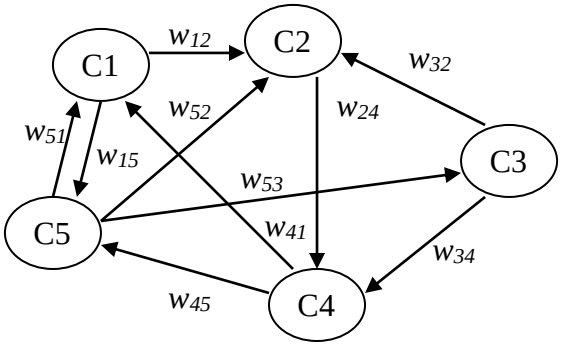

Figure 4.2. A typical fuzzy cognitive map **(Xirogiannis et al., 2004)**

FCM nodes nonlinearly transform weighted summed inputs into a numerical output. Existing knowledge on the behavior of the system is stored in the structure of nodes and interconnections of the map. A typical formula for calculating the values of concepts in FCM is:

$$A_i^{t+1} = f\left(\sum_{j=1, j\neq i}^{n} w_{ij} A_j^t\right) \tag{4.5}$$

Here, A_i^{t+1} is the value of concept C_i at the step t+1, A_j^t is the value of the interconnected concept C_j at step t, w_{ij} is the weighted arc from C_j to C_i and f is a nonlinear transformation function **(Xirogiannis et al., 2004)**.

Weights of a FCM are elements of the interval [0,1] or [-1,1]. The weights may be either crisp (e.g. {-1,0,1}) or fuzzy ({strong, medium, weak}). The negative sign (-) implies a negative effect.

The last thing to emphasize is that any number of FCMs can be combined into a single knowledge network; gathering the evaluations of several experts, a group cognitive map can be achieved.

5. A MULTIAGENT DECISION SUPPORT MODEL FOR STREET MANAGEMENT

In this section, first the rationale and the aim of this thesis study are stated, then the proposed distributed decision support model is explained.

5.1. The Rationale for the Study

As **Lanir (1990)** states, complexity can be controlled much more better by a system which,

- base its decisions to well defined distributed decision making responsibilities and several predefined scenarios, diagnosis and solutions (expert),
- evaluate information coming from environment and about the current system state simultaneously (intelligent), and
- do all those in real-time (information technologies integrated).

For any distributed human organization having a common goal, there are several decision makers at strategic, tactical and operational levels. Every single decision maker in the system should make its best possible decision and so, should make its most possible contribution to the total system performance. This work suggests that, decisional effectiveness in a cooperative distributed human system can be increased by training / distributed learning efforts at the operational level and by a distributed decision support tools at higher levels (Figure 5.1). The appropriateness of such a distributed decision support tool stems from two considerations:

- A cooperative distributed system requires coherent effort, which clearly desires stronger coordination. A classical hierarchical coordination mechanism such as a separate central coordination unit is practically ineffective. A computer-based support system can be a better alternative to achieve real-time coordination enabling a decentralized coordination structure.

- Decisions made at the tactical and strategic levels are intrinsically complex. Moreover, a geographically or functionally distributed structure dramatically increases that complexity since none of the single units in a distributed system can capture the system as a whole. A decision support system can provide necessary methodological and computational aid for decision makers in knowledge processing and problem solving.

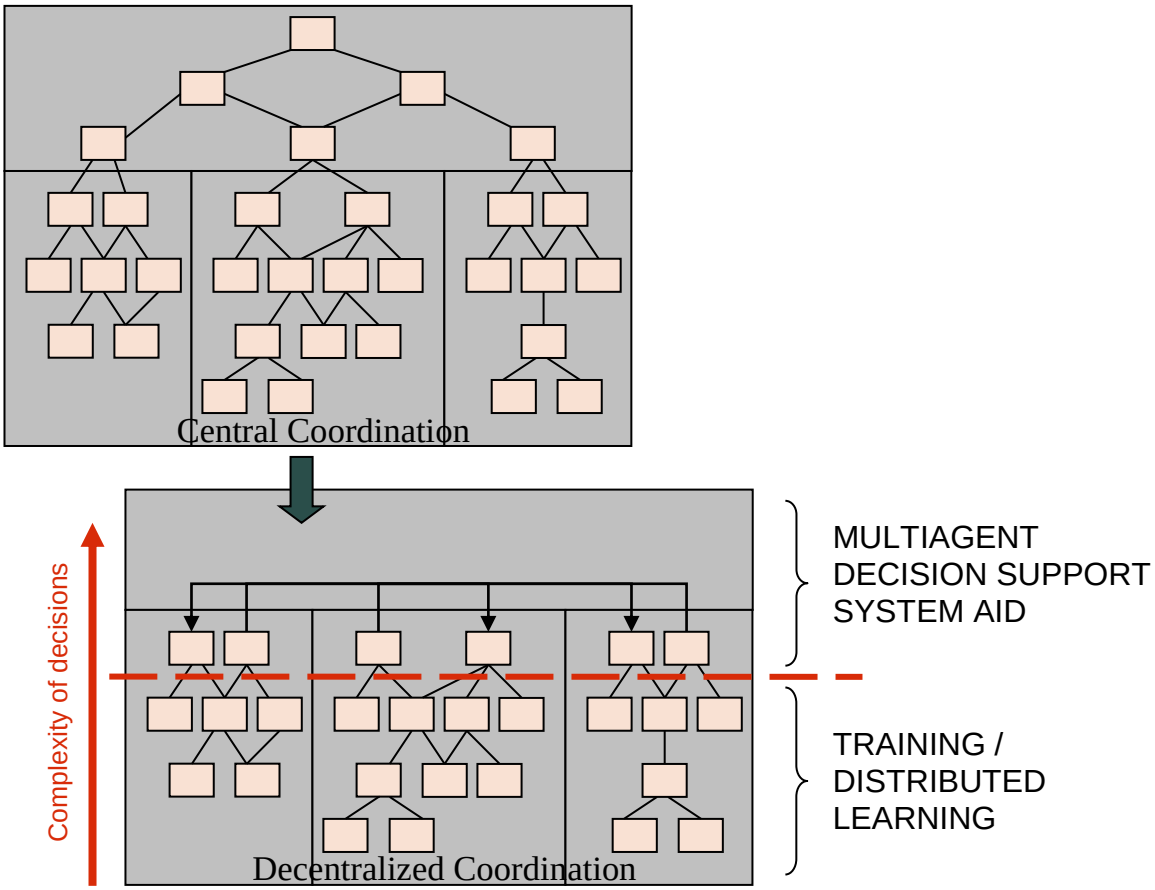

Figure 5.1. Suggestions for increasing decisional effectiveness in a distributed human system

5.2. The Aim and the Content of the Model

In this study, we propose a new distributed decision support system based on multiagent architecture. We plan to show the applicability of distributed decision support tools to achieve high performing distributed human systems on the street management example.

Street management is selected as an application domain, since it is a good example of distributed decision making: Decisions are made by some responsible personnel, at separate points in time and in location, with a common goal of achieving high street performance. Expertise and responsibility are divided between several different directorates and well performing streets require a coherent effort of those different units. (Figure 5.2., Table 5.1.) Experiences show that existence of a central control / coordination unit is nor effective, neither efficient.

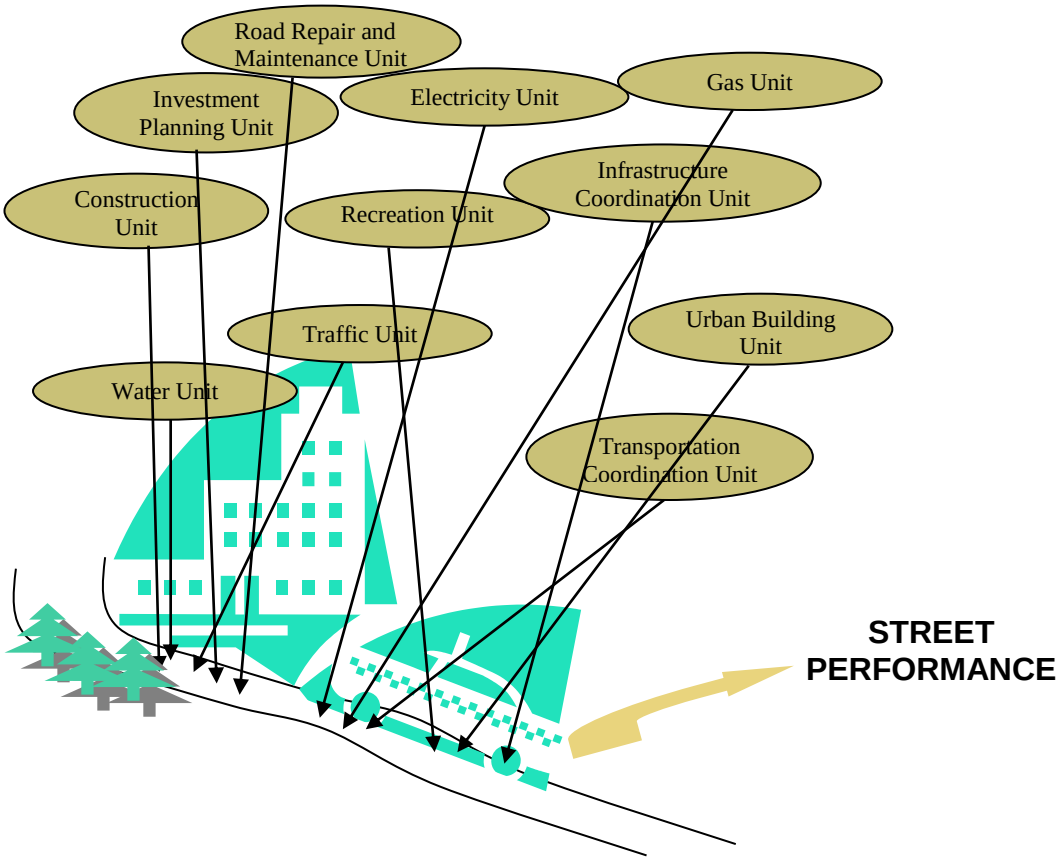

Figure 5.2. Some of the separate decision units affecting the street performance

From our point of view, in the street management example, planning and design units form the tactical level, whereas the workers physically doing the construction job on the streets stand at the operational level.

Table 5.1. Units related to street management

	Code	Unit
BASIC UNITS	U1	Top Management
	U2	Investment Planning
	U3	Road Maintenance & Repair
	U4	Infrastructure Coordination

	U5	Construction Affairs
	U6	Urban Building
	U7	Parks & Gardens
	U8	Traffic
	U9	Water Infrastructure
	U10	Gas Infrastructure
SUPPORT UNITS	*U11*	*Tender*
	U12	*Budget Finance & Inspection*
	*	*Contractor Firms*

5.3. The Organizational Model

Examining the functions of the related decision units given in Table 5.1., 12 different types of agents determined. In order to establish the distributed organizational model, a functional modularity is preferred, since it was suitable for the problem in hand. Agent types and their tasks are given in Table 5.2.

Table 5.2. Street management agent types

Agent Type	**Tasks**
Road Builder	Building road elements
Outsourcer	Outsourcing building and repair work
Designer	Designing several road elements and providing technical coordination
Road Renewal Decision Initiative	Determination of road renewal necessity
Road Repairer	Repairing several road elements
Infrastructure Repairer	Infrastructure repair determination and doing the repair work
Cost Detector	Detection of suitability of expenditures with respect to the amount determined in budget and payment
Public Listener	Receiving public complaints
Performance Simulator	Simulating the performance model
Schedular	Scheduling street operations
Law Detector	Detection of legacy
Road Repair Necessity Detector	Determination of road repair necessity

Any related unit in a local authority may act as one or more of those functional agents in real life. For example a Repair and Maintenance unit is a road repairer, an outsourcer and a repair necessity detector at the same time. So, intersecting “units”

and “agent types” sets, 26 different “street management agent”s are produced. They are listed and introduced with their names and responsibilities in Table 5.3.

Table 5.3. Street management agents

#	Code	Name	Responsibilities
1	A_RRDI_1	Top Management Road Renewal Decision Initiative	Decides which streets to renewal
2	A_RB_1	Parks & Gardens Road Builder	Builds green area along streets and plants
3	A_RB_2	Traffic Road Builder	Builds traffic signs
4	A_RB_3	Contractor Road Builder	Builds pavements for sidewalks and motorways; Decorates with street furniture
5	A_OS_1	Maintenance & Repair Outsourcer	Outsources maintenance and repair work
6	A_OS_2	Urban Building Outsourcer	Outsources street furniture placement work
7	A_OS_3	Construction Affairs Outsourcer	Outsources paving work
8	A_DSN_1	Investment Planning Designer	Designs architectural aspects of sidewalks and motorways
9	A_DSN_2	Parks & Gardens Designer	Designs green area
10	A_DSN_3	Urban Building Designer	Designs and arranges the placement of street furniture and advertisement panos
11	A_DSN_4	Traffic Designer	Designs and arranges the placement of traffic signs
12	A_SS_1	System Scheduler	Schedules infrastructure and superstructure work and states the most appropriate time for any operation.
13	A_SPS_1	System Performance Simulator	Guesses performance outputs of activities and directs decisions
14	A_PL_1	Public Relations Public Listener	Receives, classifies and distributes complaints of citizens
15	A_RR_0	Maintenance & Repair Road Repairer	Repairs paving of sidewalks and motorways
16	A_RR_1	Water Infrastructure Road Repairer	Repairs paving of sidewalks and motorways after water infrastructure work
17	A_RR_2	Gas Infrastructure Road Repairer	Repairs paving of sidewalks and motorways after gas infrastructure work
18	A_RR_3	Contractor Road Repairer	Repairs paving of sidewalks and motorways when outsourced
19	A_RR_4	Parks & Gardens Road Repairer	Repairs green area and planting

#	Code	Name	Responsibilities
20	A_RR_5	Traffic Road Repairer	Repairs traffic signs
21	A_RR_6	Urban Building Road Repairer	Repairs advertisement panos and street furniture
22	A_RRND_1	Road Repair Necessity Detector – Sub region 1	Detects whether any road repair is necessary in Sub region 1
23	A_RRND_2	Road Repair Necessity Detector – Sub region 2	Detects whether any road repair is necessary in Sub region 2
24	A_RRND_3	Road Repair Necessity Detector – Sub region 3	Detects whether any road repair is necessary in Sub region 3
25	A_LD_1	Tender Law Detector	Detects if the outsourcing job is done according to legal procedure
26	A_CD_1	Budget Finance & Inspector Cost Detector	Detects if the expenditures are done within budget limits

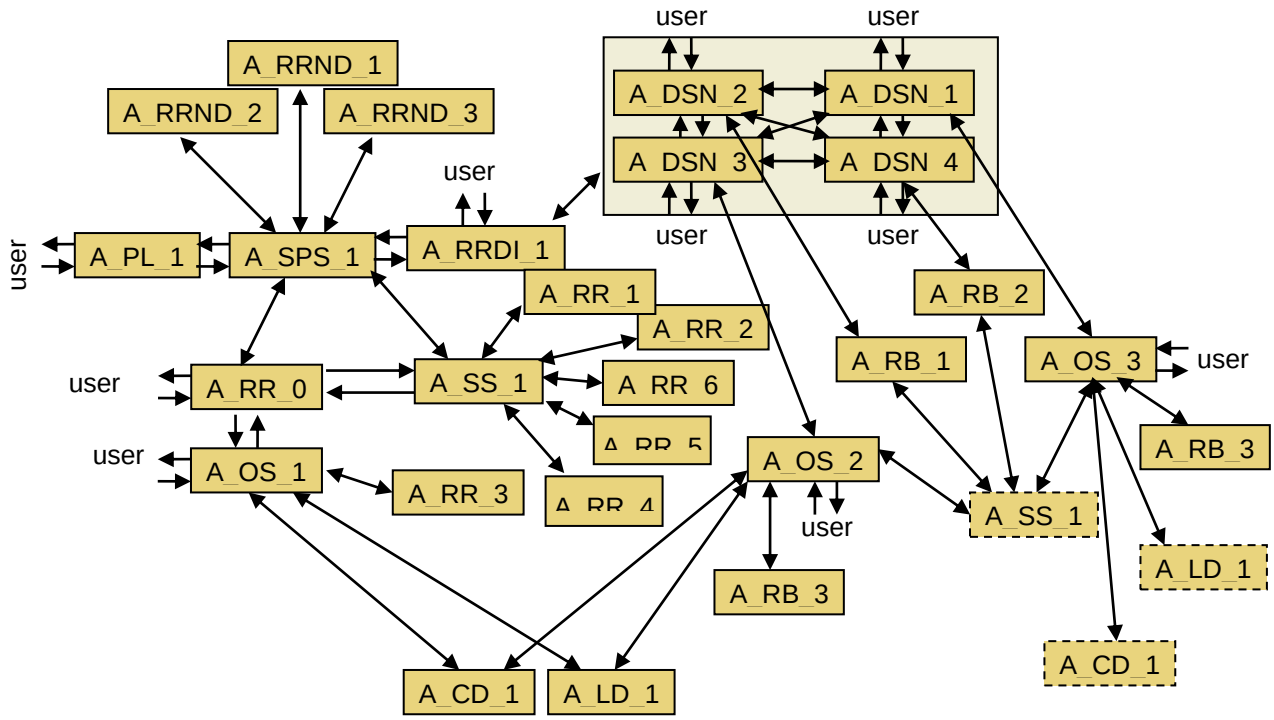

Figure 5.3. Relations between street management agents

Street management agents realize a cooperative work with the goal of maximizing the street performance. Their relations are roughly given in Figure 5.4.

5.4. Multiagent Decision Support System Architecture

The proposed decision support system has the 26 server agents introduced in the previous section. Human users of the system are the autonomous agents of the multiagent structure. They direct the process with their decisions; that means

computer and human agents cooperate and both are active elements of the system. We have argued before that as a distributed decision support system, we imagine a system neither producing decisions and dictates them to people, nor just providing a tool to let people communicate.

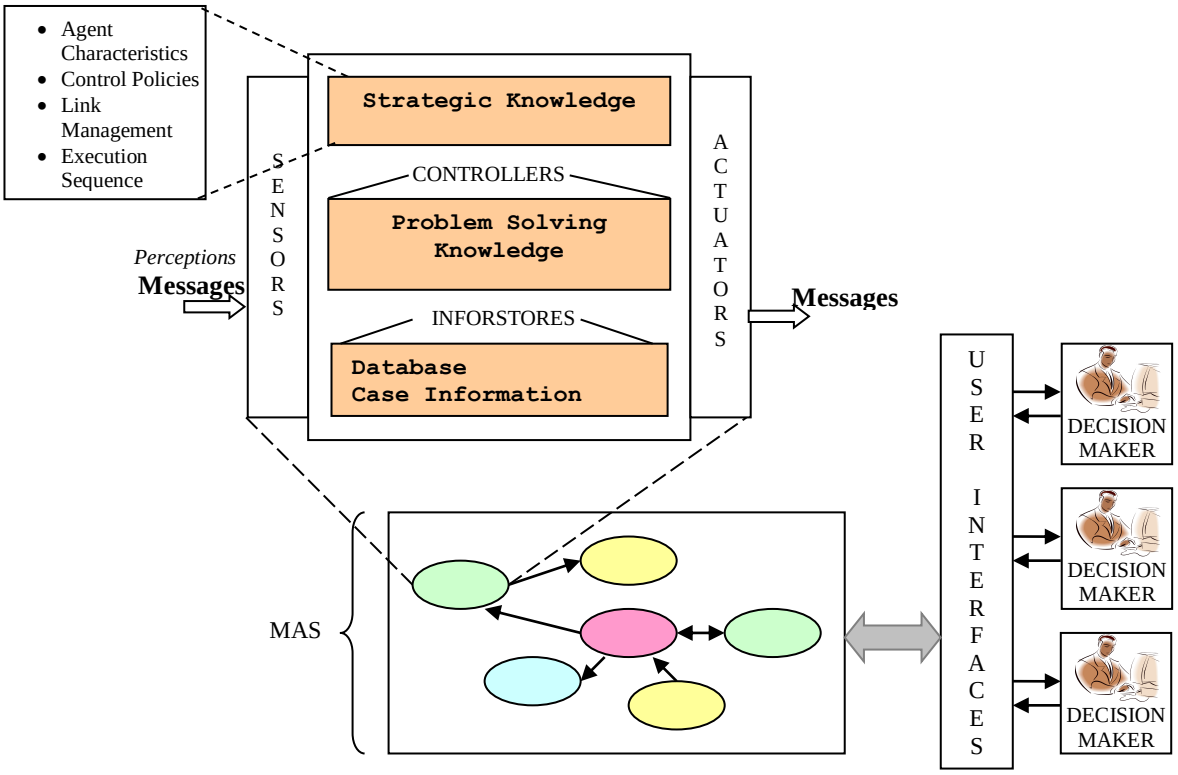

Figure 5.4. Proposed multiagent decision support system architecture

Framework of multiagent decision support system is established according to *act*SMART model of **d'Inverno and Luck (2004)** and by inspiration of the architecture and workflow explained in **Cuena and Ossowski (1990)**. The system has two parts: (Figure 5.4.) a user interface which provide communication with users and a MAS which develop proposals for decision makers. There are 11 different interfaces for 11 different kinds of users. The MAS part was explained in the previous section and was illustrated in Figure 5.3.

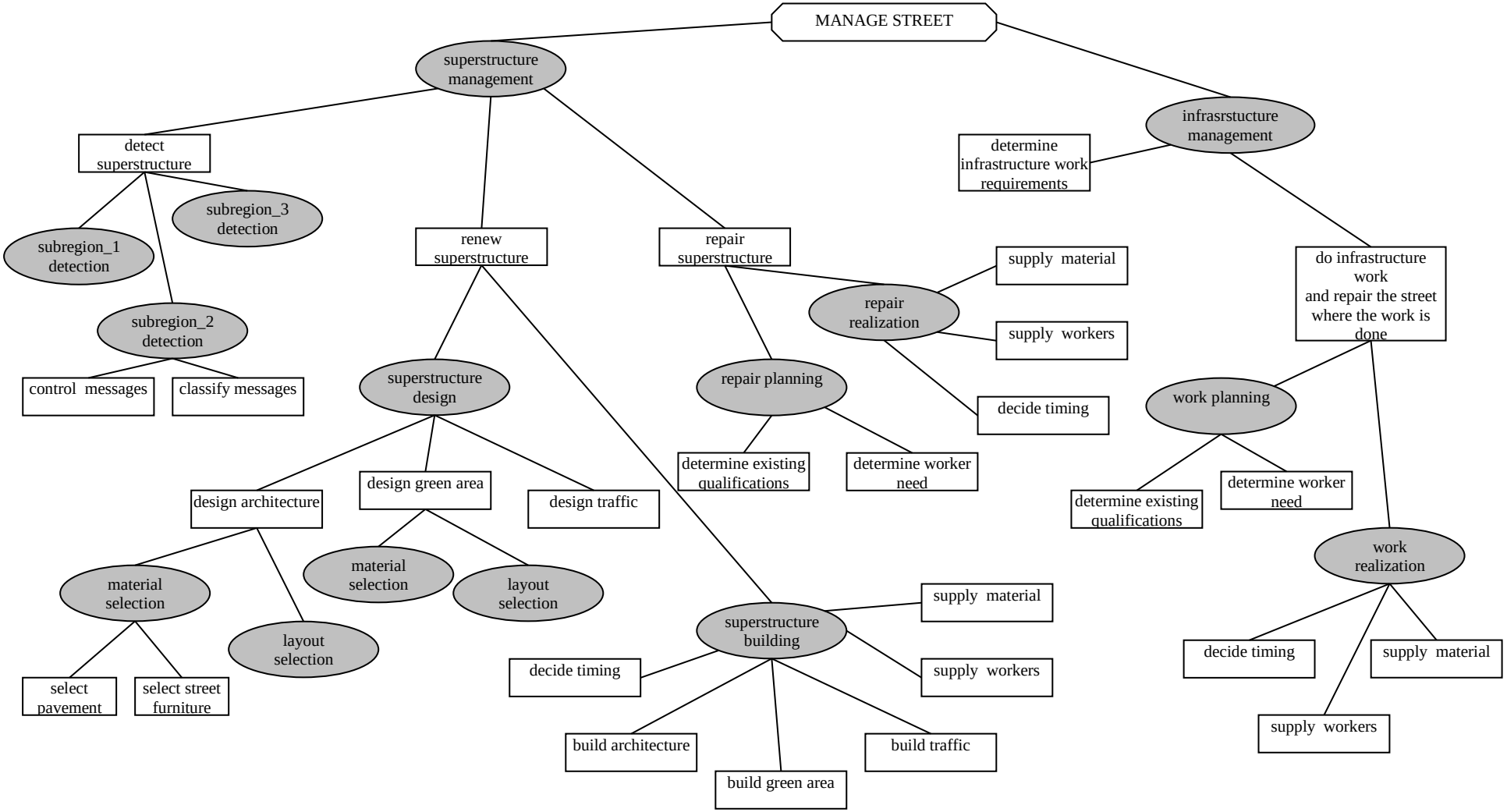

Figure 5.5. Tasks-methods-subtasks tree (TMST) of the system

The general task-methods-subtasks tree (TMST) of the decision support system is given in Figure 5.5. For implementation, the whole system seems to be a too comprehensive example for the limits of this study. Thus, the applicability of the proposed model will be shown on a sample part of the system.

5.5. The Performance Model

To set up the performance model for the street management system, we use fuzzy cognitive map (FCM) approach, because representations we need are based on expert judgments, not clear rules.

For our model, several different links are determined between the decisions made and the performance achieved; a sample is given in Figure 5.6. Nodes of FCM are determined by examining related architectural publications. The criteria for street performance are adopted from the report of **CABE (2002)**.

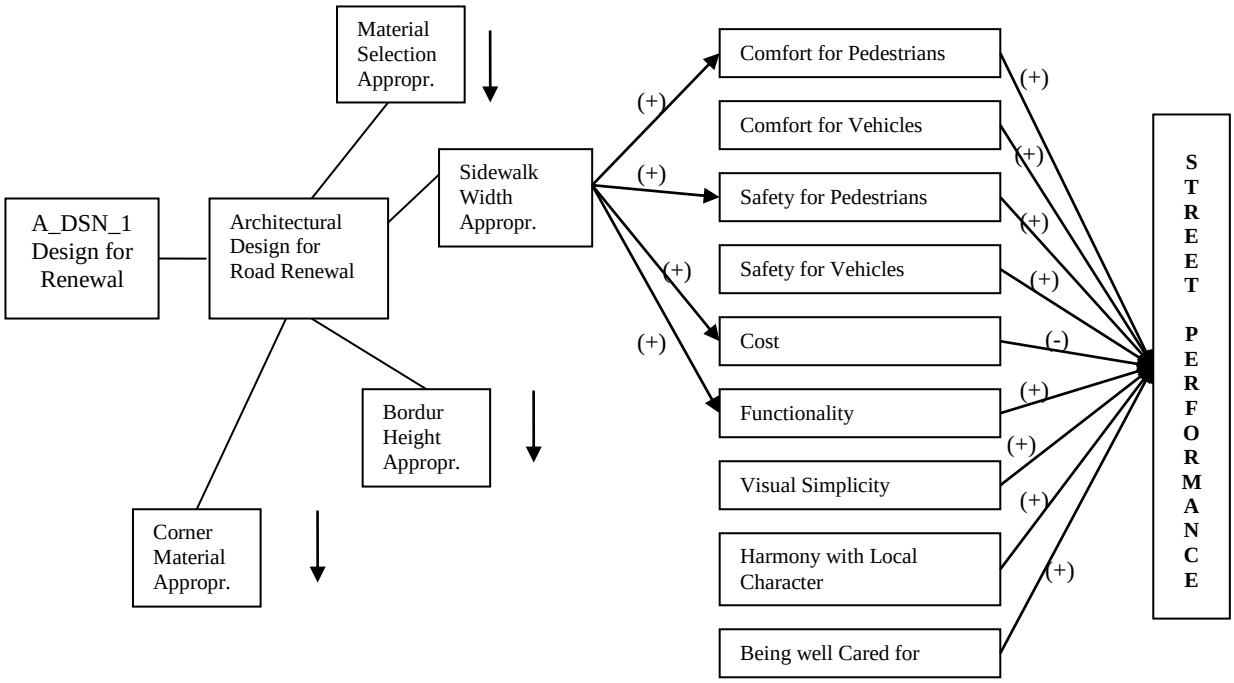

Figure 5.6. A sample part of FCM / Performance Model

In our multiagent decision support system, the performance model will be managed by a "performance simulator" agent; it will inform "others" by simulating their special problem case.

Calculations are done with respect to the evaluations of three experts. Word weights and linguistic measurements of concepts are used, i.e. negatively very very high, negatively very high, negatively high, negatively medium, negatively low, negatively very low, negatively very very low, zero, positively very very low, positively very low, positively low, positively medium, positively high, positively very high, positively very very high, following the example of **Xirogiannis et al. (2004)**. Every element of this scale has a triangular membership function; all membership functions are given in Figure 5.7. Membership functions are narrower at the edges because we assume that evaluations at those points are naturally more strict.

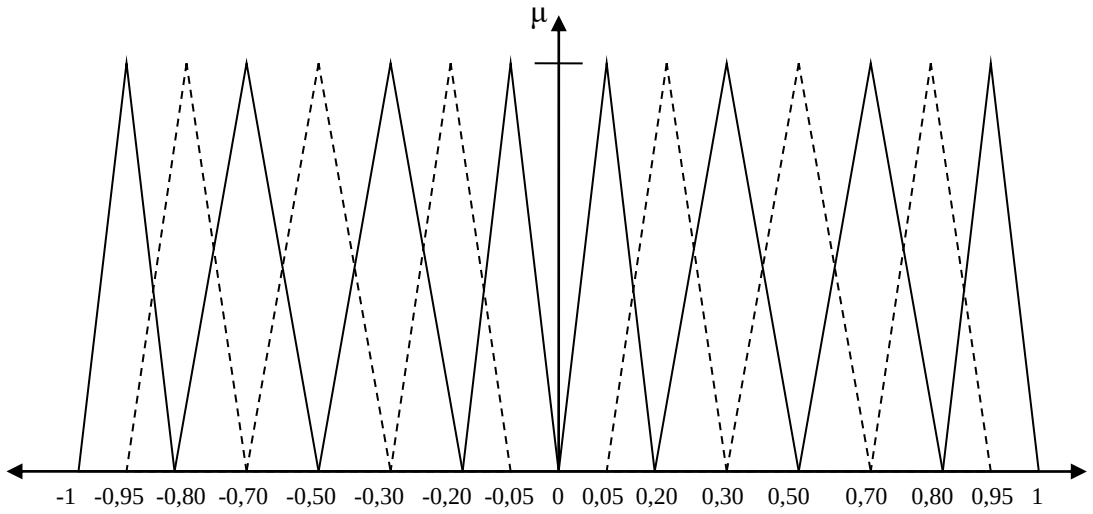

Figure 5.7. Membership functions for linguistic variables

The nonlinear transformation function used at nodes is the hyperbolic tangent function, $\tanh(x)$.

A sample calculation of FCM weights is given below. A full list of the calculated weights is given in Table 5.4. and Table 5.5. For the weight between "visual simplicity" and "street performance", three evaluations are stated as: positively low, positively medium and positively high. The combination of the related membership functions is given in Figure 5.8.

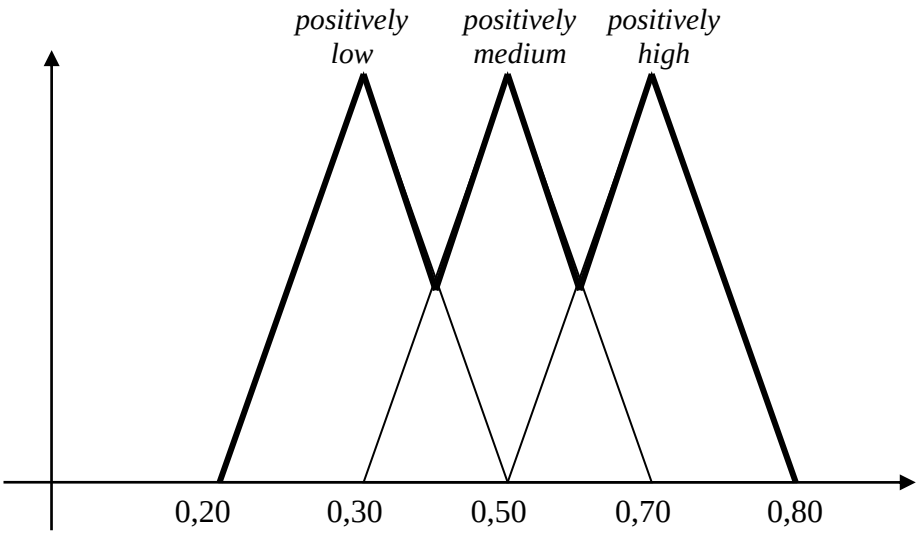

Figure 5.8. Combination of membership functions for sample weight calculation

Then, using the centroid method of defuzzification,

$$z^* = [\int_{0,2}^{0,35}\left(-\frac{4}{3}+\frac{20}{3}z\right)zdz+\int_{0,35}^{0,42}\left(\frac{10}{3}-\frac{20}{3}z\right)zdz+\int_{0,42}^{0,5}\left(-\frac{3}{2}+5z\right)zdz+\int_{0,5}^{0,59}\left(\frac{7}{2}-5z\right)zdz$$
$$+\int_{0,59}^{0,65}\left(-\frac{10}{3}+\frac{20}{3}z\right)zdz+\int_{0,65}^{0,8}\left(\frac{16}{3}-\frac{20}{3}z\right)zdz\ ]\Big/[\int_{0,2}^{0,35}\left(-\frac{4}{3}+\frac{20}{3}z\right)dz+\int_{0,35}^{0,42}\left(\frac{10}{3}-\frac{20}{3}z\right)dz$$
$$+\int_{0,42}^{0,5}\left(-\frac{3}{2}+5z\right)dz+\int_{0,5}^{0,59}\left(\frac{7}{2}-5z\right)dz+\int_{0,59}^{0,65}\left(-\frac{10}{3}+\frac{20}{3}z\right)dz+\int_{0,65}^{0,8}\left(\frac{16}{3}-\frac{20}{3}z\right)dz\ ]=w_{71}\cong 0{,}5$$

(5.1)

Table 5.4. Relations between street information and street performance / weights of the FCM - Hierarchy 1

		STREET PERFORMANCE
PERFORMANCE INDICATORS	**Comfort for Pedestrians**	0,83
	Comfort for Vehicles	0,65
	Safety for Pedestrians	0,9
	Safety for Vehicles	0,9
	Cost	-0,65
	Functionality	0,65
	Visual Simplicity	0,5
	Harmony with Local Character	0,35
	Being well Cared for	0,83

Table 5.5. Relations between street information and street performance / weights of the FCM - Hierarchy 2

	Comfort for Pedestrians	Comfort for Vehicles	Safety for Pedestrians	Safety for Vehicles	Cost	Functionality	Visual Simplicity	Harmony with Local Character	Being well Cared for
Sidewalk Width Appropriateness	0,9	0	0,9	0	0,83	0,9	0,5	0	0
Sidewalk Slope Appropriateness	0,35	0	0	0	0,1	0,18	0,35	0	0,35
Sidewalk Height Appropriateness	0,83	0	0,18	0	0,1	0,65	0,65	0	0
Paving Material Appropriateness	0,65	0,83	0,65	0,83	0,9	0,35	0,9	0,9	0
Corner Material Usage	0,35	0	0,35	0	0,65	0,5	0,83	0,18	0
Paving Quality	0,9	0,9	0,65	0,9	0,83	0,65	0,83	0	0,9
Barriers Height Appropriateness	0,18	0	0,83	0	0,35	0,65	0,83	0	0
Barriers Frequency Appropriateness	0,35	0	0,83	0	0,65	0,65	0,83	0	0,35
Lighting Sufficiency	0,83	0,9	0,9	0,9	0,65	0,83	0,5	0	0
Lighting Method (Material) Appropriateness	0,83	0,83	0,83	0,83	0,83	0,5	0,83	0,5	0
Parking Area Sufficiency	0,18	0,83	0,65	0,9	0,83	0,9	0,83	0,18	0
Drainage Channels Sufficiency	0,83	0,5	0,35	0,65	0,5	0,35	0,18	0	0,35
Plant Quantity Sufficiency	0,65	0	0,18	0	0,65	0,65	0,5	0,5	0
Plant Type Selection	0,1	0	0	0	0,83	0,18	0,83	0,65	0
Plant Layout Appropriateness	0,83	0,5	0,65	0,35	0	0,83	0,83	0	0
Green Area Design Appropriateness	0	0	0	0	0,18	0,18	0,83	0,18	0
Traffic Signs Sufficiency	0,5	0,9	0,9	0,9	0,83	0,1	0,65	0	0
Traffic Signs Quality	0,5	0,83	0,9	0,9	0,65	0	0,35	0,1	0,9
Bicycle Road Appropriateness	0,65	0,35	0,9	0,1	0,83	0,9	0,35	0,5	0
Street Furniture Sufficiency	0,65	0	0,18	0	0,83	0,83	0,65	0,5	0
Street Furniture Material Appropriateness	0,65	0	0,35	0	0,83	0,18	0,65	0,83	0
Advertisement Panos Appropriateness	0,5	0,18	0,35	0,65	0,5	0,65	0,83	0,65	0

6. SAMPLE IMPLEMENTATION

This section includes implementation of a simplified sample part of the distributed decision support system proposed in the previous section, in Java environment.

6.1. The Sample Part Subject to Implementation

Implementation will be done for a sample workflow among four street management agents and two users. Thus, the sample program includes four agents and two interfaces. The architectures of the agents subject to the implementation are given in Figures 6.2., 6.3., 6.4., and 6.5. The *act*SMART structure is used to define agents as stated before.

In the sample implementation, we made several assumptions. The workflow is as follows step by step (illustrated in Figure 6.1.):

- The PublicListener agent gets a complaint about a paving problem on Street1ofRegion1 (R1S1) and transfers it to the PerformanceSimulator.
- Using the performance model explained in the previous section, the PerformanceSimulator agent decides that a repair work is enough and sends a message to the related RoadRepairer agent.
- After that, the information about the pavement problem appears at the user interface of the RoadRepairer agent and the related human controller decides to approve the offered repair job.
- The RoadRepairer agent sends a message to the SystemSchedular agent when the approval is submitted.
- Examining existing jobs in its agenda, the SystemSchedular agent appoints an appropriate date and replies to the RoadRepairer.
- All necessary support information then presented through the RoadRepairer interface and a message is sent back to the PublicListener agent in order to inform the former complainant user.

In order to realize the sample implementation of the proposed distributed decision support system, a software development is necessary. Java program is used for agent construction as suggested by **d'Inverno and Luck (2004)**. As a general-purpose and object-oriented language, Java provides all of the base functions needed to design and implement intelligent agents **(Bigus and Bigus, 1998)**.

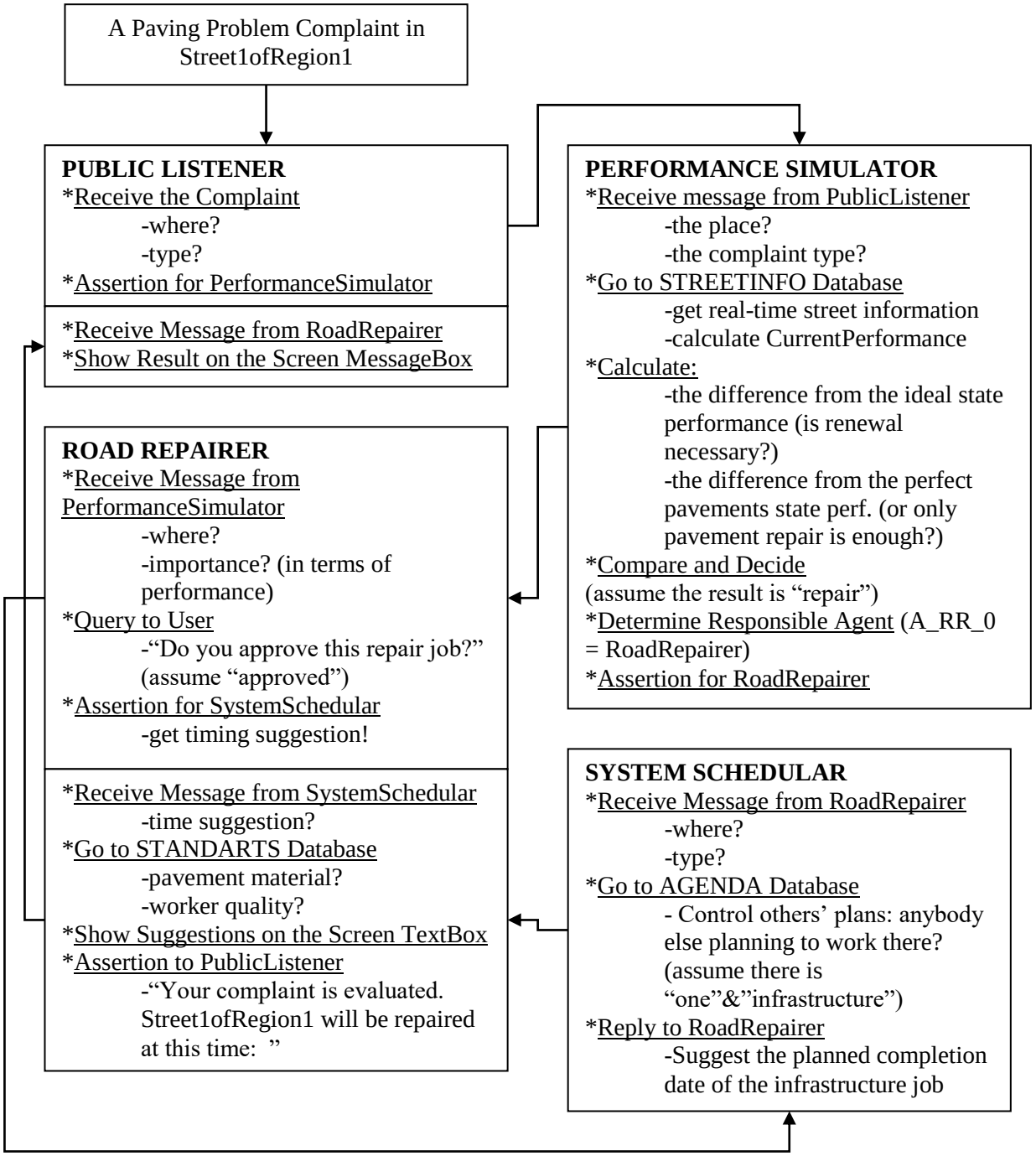

Figure 6.1. Sample workflow subject to implementation

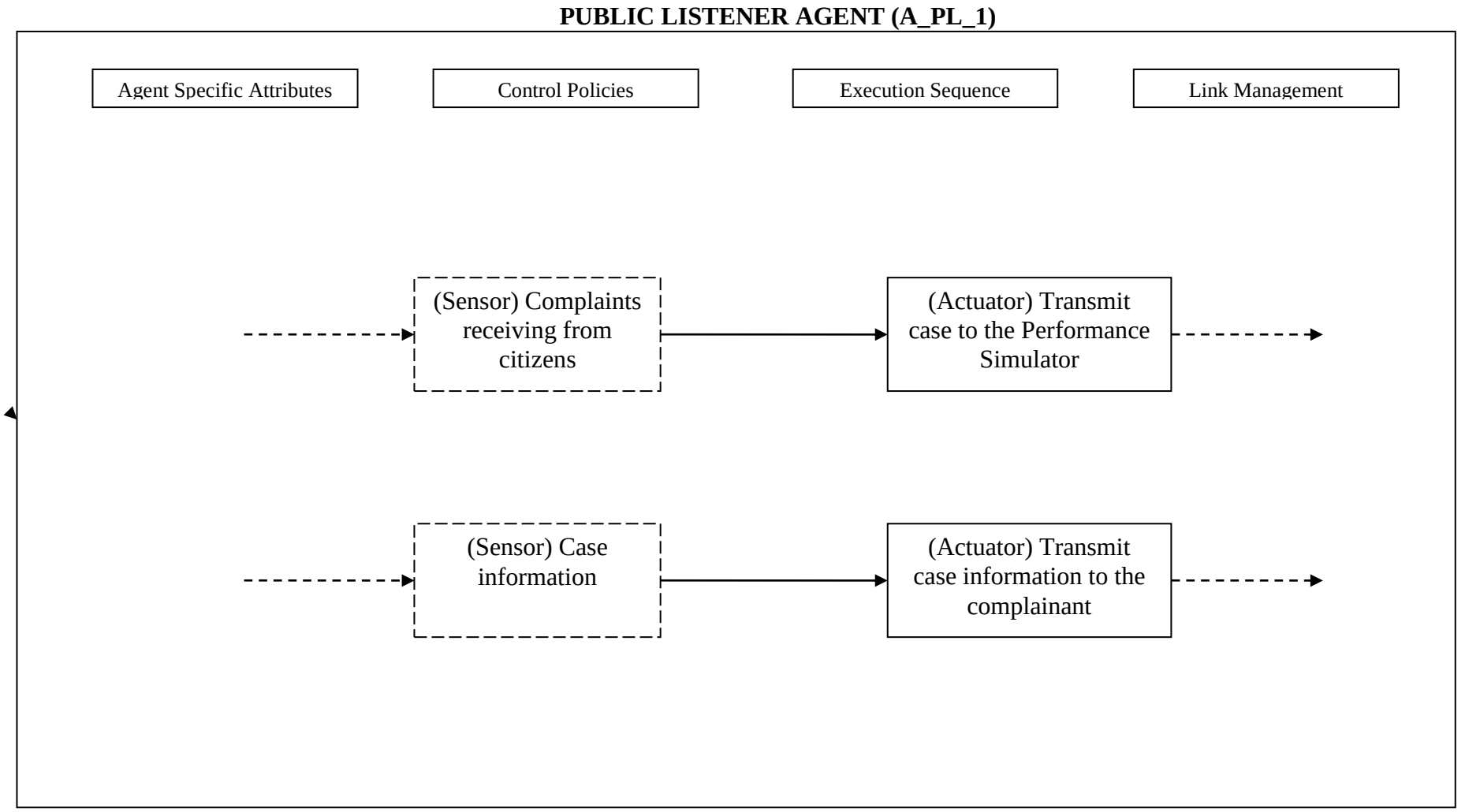

Figure 6.2. Public Listener Agent Architecture

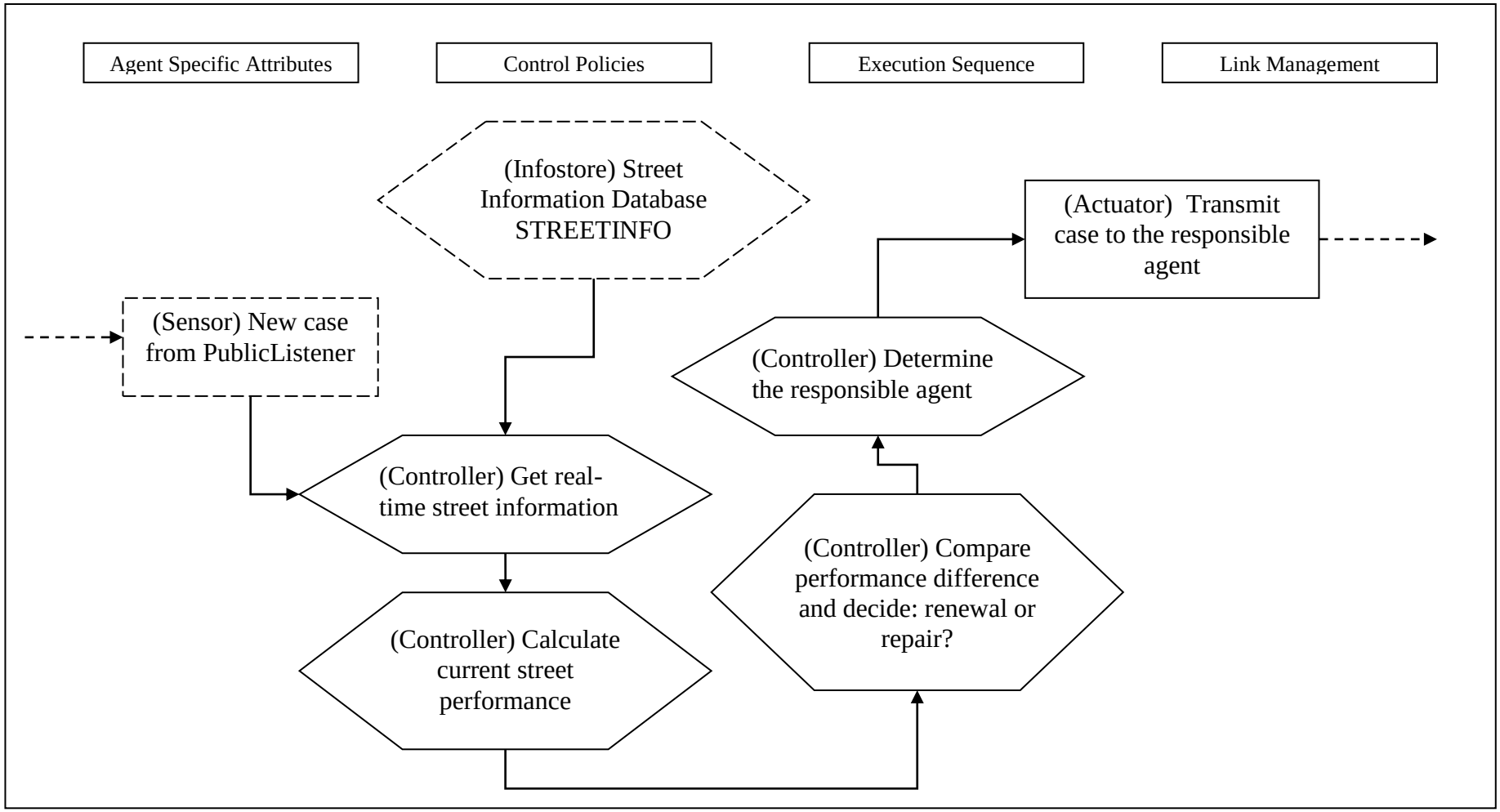

Figure 6.3. Performance Simulator Agent Architecture

ROAD REPAIRER AGENT (A_RR_0)

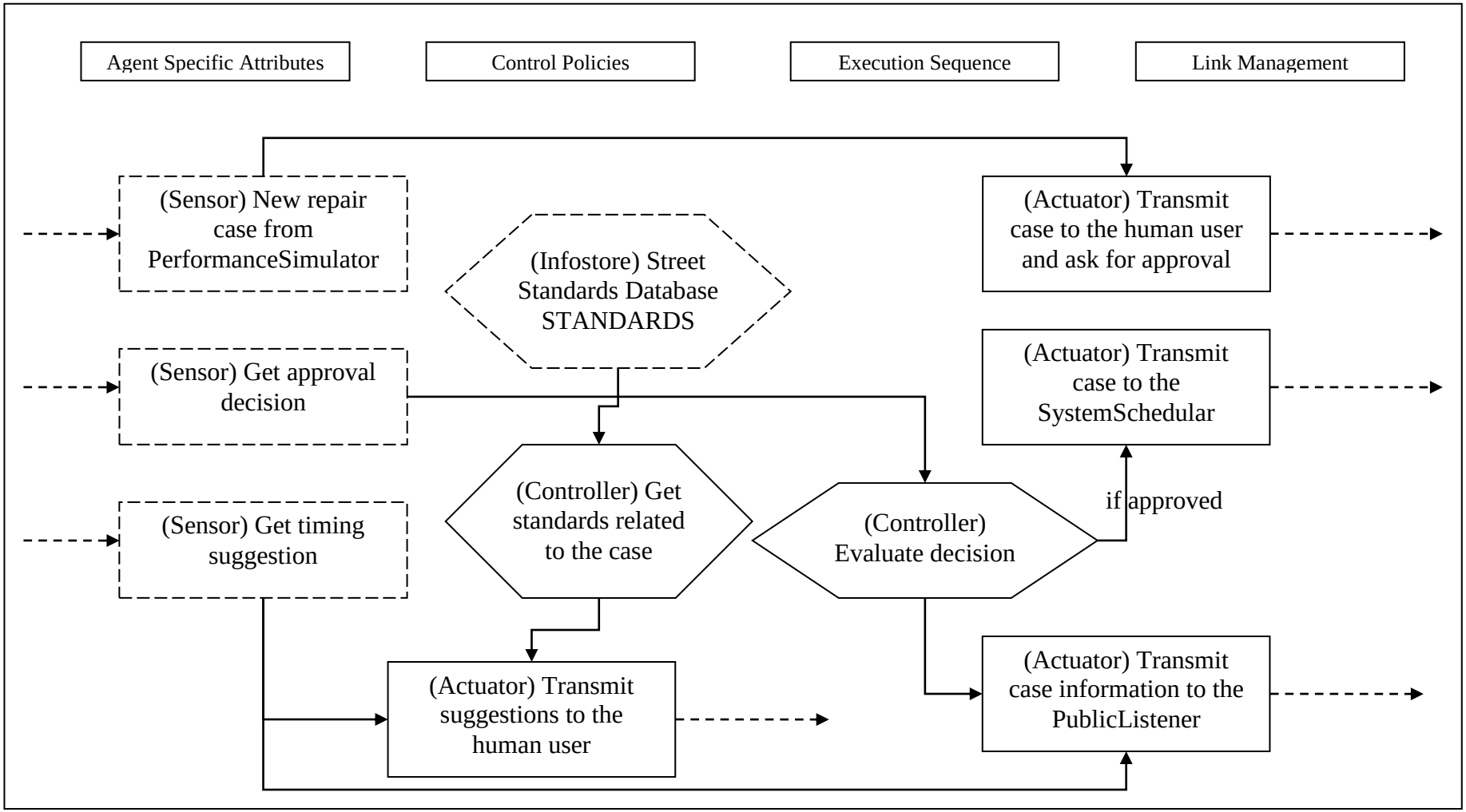

Figure 6.4. Road Repairer Agent Architecture

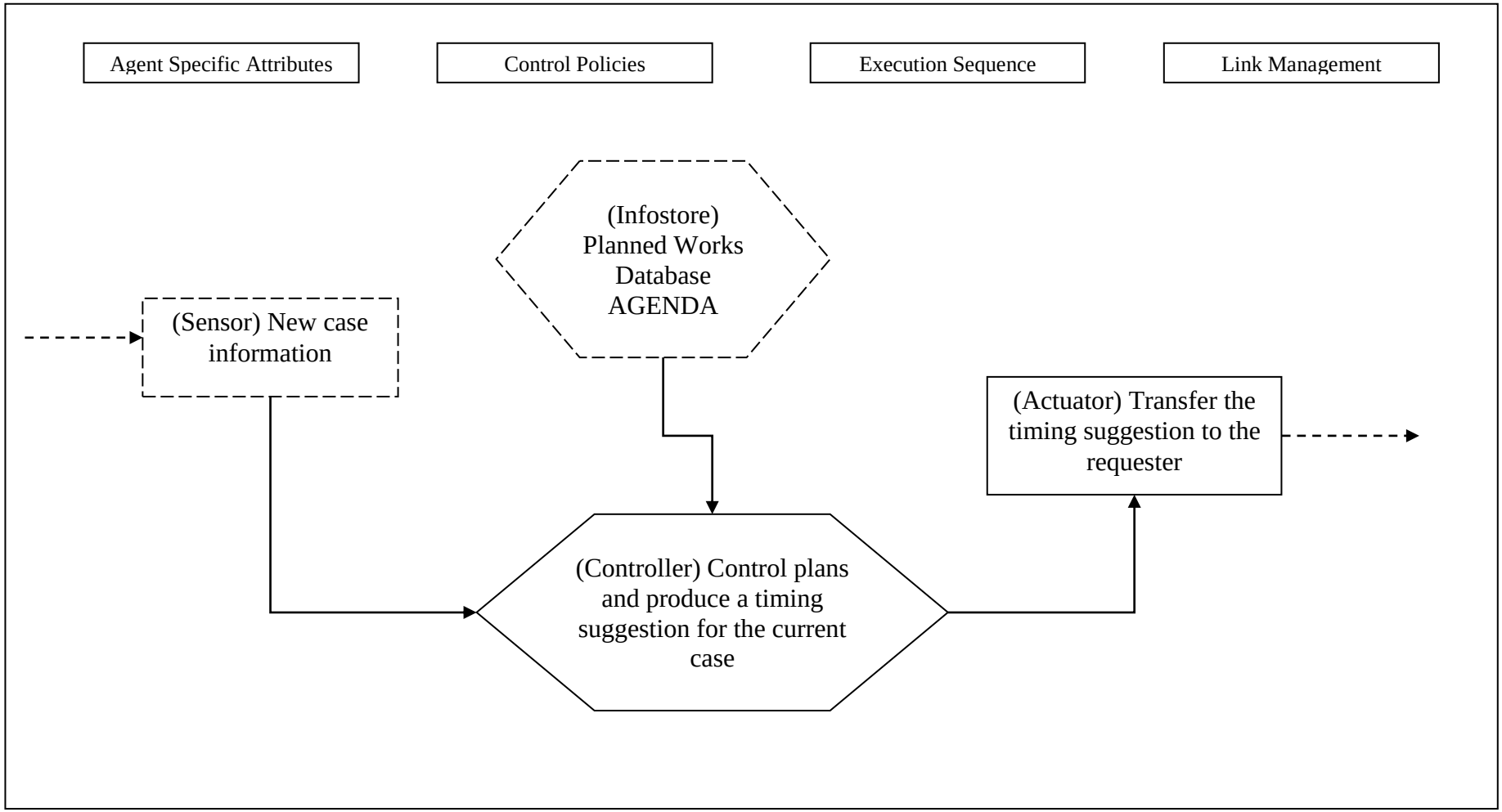

Figure 6.5. SystemSchedular Agent Architecture

6.2. Java Environment for Multiagent Systems Design

Java is an object-oriented language developed by Sun Mycrosystems. It was originally designed for programming real time embedded software for consumer electronics, particularly set-top boxes to interface between cable providers, broadcasters, and televisions or television like appliances.

Java language uses C++ syntax. A Java class contains data members and methods that define the state and behavior of the instances or objects of that class. In addition to objects, Java supports elementary data types for performance.

Java provides all of the functionality desired to design and implement intelligent software agents. Its general purpose, object-oriented language capabilities allow knowledge to be represented and reasoning and learning algorithms to be implemented easily. Java's portable byte codes allow agents to be packaged as applets or as mobile Java programs **(Bigus and Bigus, 1998)**.

6.3. Sample Program Structure

Java program classes developed for the sample implementation program are given in Figure 6.6.

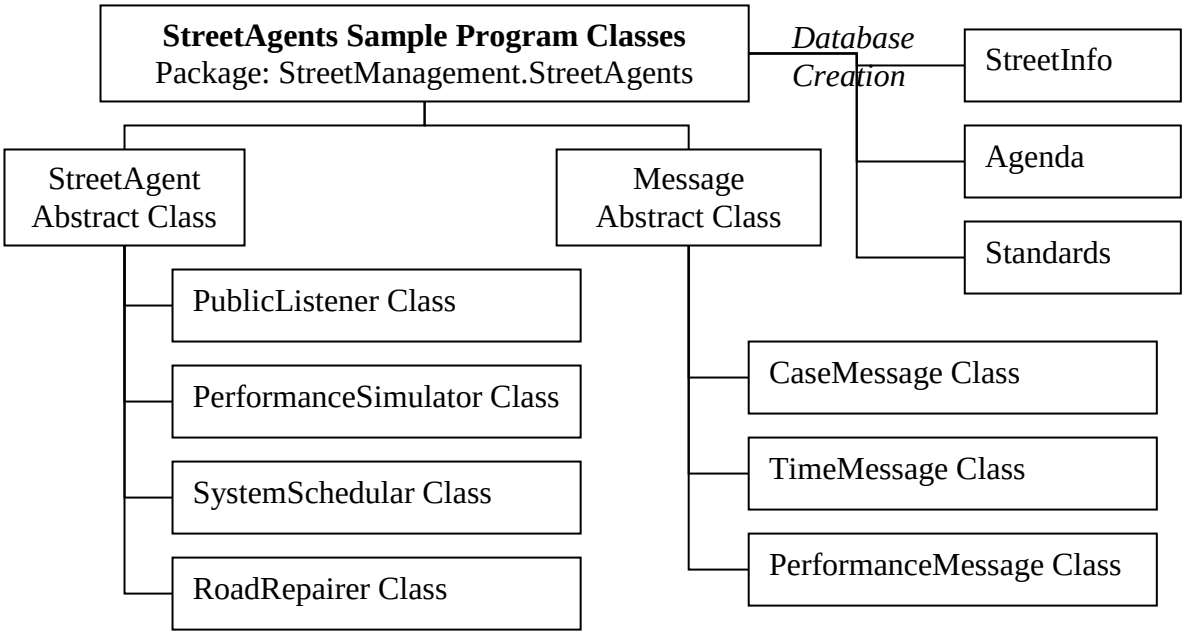

Figure 6.6. Program classes for sample implementation

The main source used during software development is “The Java Tutorial”. However, similar studies such as agentMom are also examined and used as help sources.

The StreetAgent abstract class contains common features for all agents.

```
public abstract class StreetAgent implements Runnable{
      public String AgentCode;
      public String AgentName;

      public StreetAgent(String agentcode, String agentname){
             this.AgentCode=agentcode;
             this.AgentName=agentname;
      }

      public abstract void run();
  }
```

Figure 6.7. StreetAgent abstract class

Every agent, in fact, is an extension of this base class and thus, has at least “code” and “name” features as well as a run() method. An example is given in Figure 6.8.

```
      public class SystemSchedular extends StreetAgent {
            ...
            public SystemSchedular(String name, String code){
                     super(code, name);
            ...
            }

            public void run(){
            ...
            }

            public static void main(String[] args) throws IOException {
            SystemSchedular ss=new SystemSchedular("a_sps_1","SystemSchedular");
            ss.run();
            }
      }
```

Figure 6.8. Extention example for StreetAgent class

The sample program uses three databases. Names and fields of the databases are given in Table 6.1. Data set used during the sample implementation does not belong to a real case, but sufficient to show how the application works. It can be seen in Figure 6.10.

Databases are created using simple SQL statements, such as the one given in Figure 6.9.

```
      createString = "CREATE TABLE STANDARDS (StreetCode VARCHAR(32),” +
      “MaterialType VARCHAR(32), WorkerType VARCHAR(32))";
```

Figure 6.9. Database creation example

Table 6.1. Databases for the sample program

Table	Fields
STREETINFO	StreetCode SidewalkWidthApp SidewalkSlopeApp SidewalkHeightApp PavingMaterialApp CornerMaterialUse PavingQuality BarriersHeightApp BarriersFrequencyApp LightingSuff LightingMethodApp ParkingAreaSuff DrenaigeChannellsSuff PlantQuantitySuff PlantTypeSelection PlantLayoutAppropriteness GreenAreaDesignApp TrafficSignsSuff TrafficSignsQuality BcycleRoadApp StreetFurnitureSuff StreetFurnMaterialApp AdPanosApp
AGENDA	WorkCode Type OnStreetCode BelongsTo PlannedStart PlannedFinish
STANDARDS (Designed for Repair)	StreetCode MaterialType WorkerType

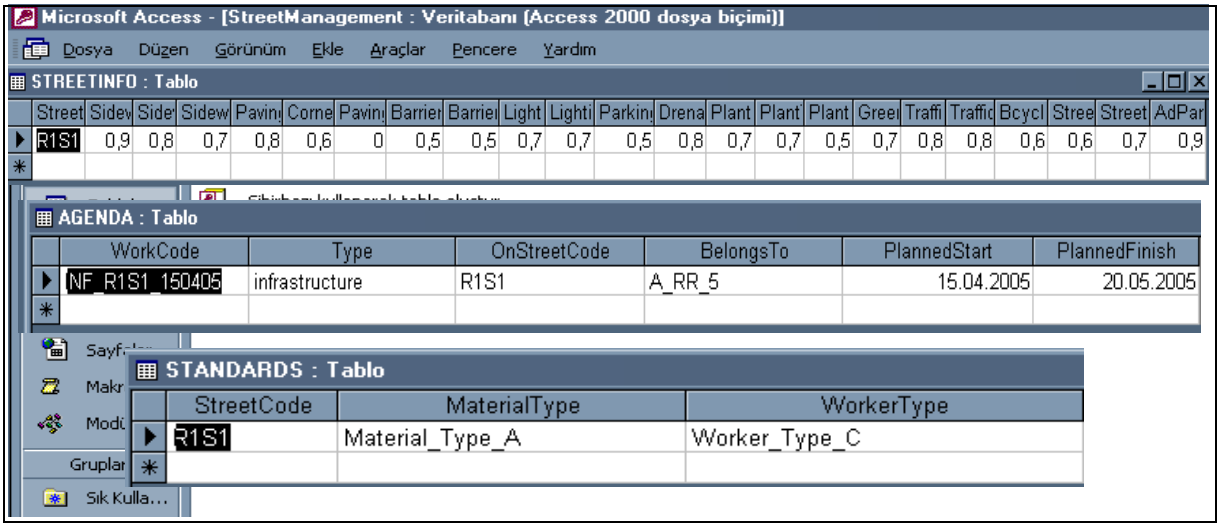

Figure 6.10. Data set used during the sample implementation in StreetManagement database

The Message class is another abstract class. It contains the minimum requirements a message should satisfy in StreetManagement program.

```
public abstract class Message implements java.io.Serializable {
        public String Case;
        public String Street;

              public Message(String c,String s) {
              super();
              this.Case=c;
              this.Street=s;

       }
}
```

Figure 6.11. Message abstract class

There are three extensions of the abstract class Message: CaseMessage, PerfMessage and TimeMessage. CaseMessage extends Message class by "complaint type" variable, PerfMessage by "performance before" and "performance after" variables and finally TimeMessage by "time suggestion" variable. Table 6.2. summarizes message exchanges between StreetManagement agents.

Table 6.2. Message exchanges between agents

		To			
		A_PL_1	**A_RR_0**	**A_SPS_1**	**A_SS_1**
From	**A_PL_1**			CaseMessage	
	A_RR_0	TimeMessage			CaseMessage
	A_SPS_1		PerfMessage		
	A_SS_1		TimeMessage		

6.4. Implementation Outputs

The sample implementation process has been explained previously. Below, program outputs of the implementation are given. Agents work as separate programs simultaneously. Each agent starts a socket on a different port to be able to listen messages coming from other agents.

The process starts with a complaint registration from the PublicListener agent interface.

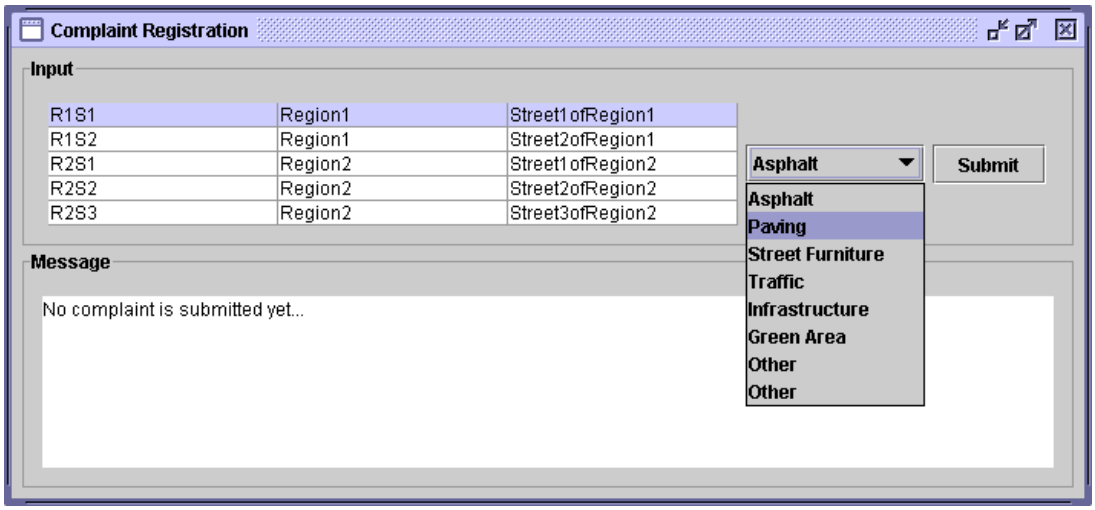

Figure 6.12 Complaint registration from the PublicListener agent user interface

When the submit button is clicked, the submitted case is transferred to the PerformanceSimulator. At this time, PerformanceSimulator has been listening on port 4000.

```
// Receive message from PublicListener

ServerSocket serverSocket = null;
try {
        serverSocket = new ServerSocket(4000);
} catch (IOException e) {
        System.err.println("Could not listen on port:"+ 4000);
        System.exit(1);
}

Socket clientSocket = null;
try {
          clientSocket = serverSocket.accept();
          ObjectInputStream input;
          ObjectOutputStream output;

          output = new ObjectOutputStream(clientSocket.getOutputStream());
          output.flush();
          input = new ObjectInputStream(clientSocket.getInputStream());

              CaseMessage m;
              m= (CaseMessage)input.readObject();
              currentcase = m.Case;
              street = m.Street;
              complaint = m.Type;

          output.close();
          input.close();
          clientSocket.close();
          serverSocket.close();

} catch (IOException e) {
          System.err.println("Accept failed.");
          System.exit(1);
} catch (ClassNotFoundException e) {
          System.err.println("ClassNotFoundException");
}

System.out.println("New case received on "+street+" !");
```

Figure 6.13. Case transfer to the PerformanceSimulator agent

PerformanceSimulator then activates its “setPerformanceValues” method in order to calculate the current performance value and the performance value after repair, using the performance model explained before. Recent street values are attained from the STREETINFO database.

```
public double calculatePerformance(String aorb){
        double[][] weights1 = {
                {0.9,0.0,0.9,0.0,0.83,0.9,0.5,0.0,0.0 },
                {0.35,0.0,0.0,0.0,0.1,0.18,0.35,0.0,0.35},
                {0.83,0.0,0.18,0.0,0.1,0.65,0.65,0.0,0.0},
                {0.65,0.83,0.65,0.83,0.9,0.35,0.9,0.9,0.0},
                {0.35,0.0,0.35,0.0,0.65,0.5,0.83,0.18,0.0},
                {0.9,0.9,0.65,0.9,0.83,0.65,0.83,0.0,0.9},
                {0.18,0.0,0.83,0.0,0.35,0.65,0.83,0.0,0.0},
                {0.35,0.0,0.83,0.0,0.65,0.65,0.83,0.0,0.35},
                {0.83,0.9,0.9,0.9,0.65,0.83,0.5,0.0,0.0},
                {0.83,0.83,0.83,0.83,0.83,0.5,0.83,0.5,0.0},
                {0.18,0.83,0.65,0.9,0.83,0.9,0.83,0.18,0.0},
                {0.83,0.5,0.35,0.65,0.5,0.35,0.18,0.0,0.35},
                {0.65,0.0,0.18,0.0,0.65,0.65,0.5,0.5,0.0},
                {0.1,0.0,0.0,0,0.83,0.18,0.83,0.65,0.0},
                {0.83,0.5,0.65,0.35,0.0,0.83,0.83,0.0,0.0},
                {0.0,0.0,0.0,0.0,0.18,0.18,0.83,0.18,0.0},
                {0.5,0.9,0.9,0.9,0.83,0.1,0.65,0.0,0.0},
                {0.5,0.83,0.9,0.9,0.65,0.0,0.35,0.1,0.9},
                {0.65,0.35,0.9,0.1,0.83,0.9,0.35,0.5,0.0},
                {0.65,0.0,0.18,0,0.83,0.83,0.65,0.5,0.0},
                {0.65,0.0,0.35,0.0,0.83,0.18,0.65,0.83,0.0},
                {0.5,0.18,0.35,0.65,0.5,0.65,0.83,0.65,0.0}
                };
    double weights2[] = {0.83,0.65,0.9,0.9,-0.65,0.65,0.5,0.35,0.83};

    double[] pvalues;
    pvalues=new double[22];
    String url = "jdbc:odbc:StreetManagement";
        Connection con;
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "manager", "open");
            PreparedStatement stQuery = con.prepareStatement(
                      "SELECT * FROM STREETINFO WHERE "+
                      "StreetCode LIKE ?");
                      stQuery.setString(1, street);

            ResultSet rs = stQuery.executeQuery();
       while (rs.next()) {
            for (int i=0;i<22;i++){
              pvalues[i]=rs.getDouble(i+2);
            }
        }
             con.close();

        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }

       if (aorb.equals("after")){
              pvalues[5]=1.0;
       }

    double[] h;
    double[] hNodeValues;
    h = new double[9];
    hNodeValues = new double [9];
       for (int j=0;j<9;j++){
              h[j]=0;
```

```
                for(int i=0;i<22;i++){
                        h[j] = h[j] + weights1[i][j] * pvalues[i];
                }
        }

    for (int i=0;i<9;i++)hNodeValues[i] = tanh(h[i]);

    double p=0;
    for (int i=0;i<9;i++){
        p = p + weights2[i] * hNodeValues[i];
        }
    return p;
}

public void setPerformanceValues(){
        pb=calculatePerformance("before");
        pa=calculatePerformance("after");
        double perfMembershipbef = tanh(pb);
        double perfMembershipaft = tanh(pa);

        perfbefore="~ "+setLinguistic(pb)+" performance with performance"+
        " membership "+perfMembershipbef;
        perfafter="~ "+setLinguistic(pa)+" performance with performance"+
        " membership "+perfMembershipaft;
        System.out.println("Performance values are calculated...");
}
```

Figure 5.14. Performance values calculation

The responsible for any complaint case is simply found with the rule given in Figure 6.15.

```
        public void setWorkType(){
                if (setLinguistic(pa).equals("High")) worktype = "Repair";
                else worktype = "Renewal";
        }
```

Figure 6.15. Responsible agent determination rule

PerformanceSimulator then sends the performance message (PerfMessage) it creates to the responsible agent RoadRepairer. Final view of its output screen is given in Figure 6.16.

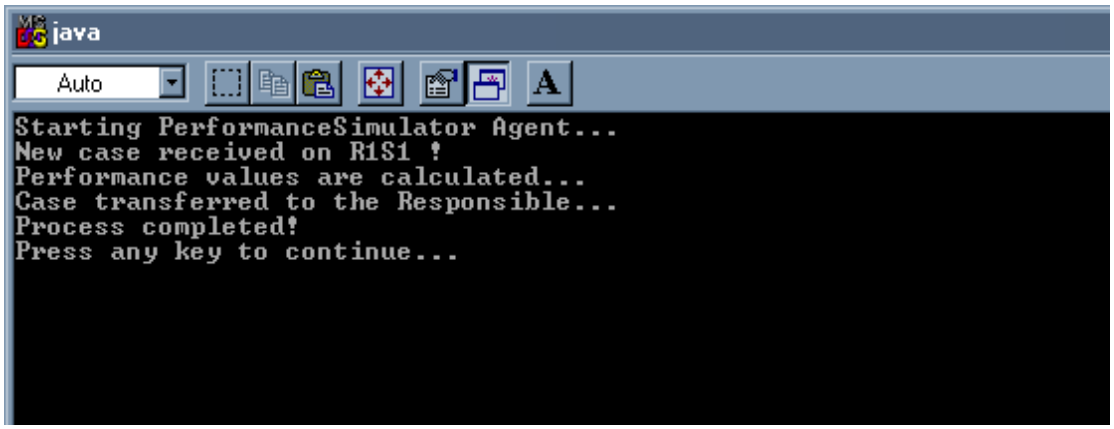

Figure 6.16. Output screen of the PerformanceSimulator agent

The RoadRepairer agent listens on port 5000. After receiving the PerfMessage coming from PerformanceSimulator, it displays the "new case" information on the user interface screen. However, since for this sample implementation we assume that the user approves the job, we do not ask but display a statement about our assumption. Then, the RoadRepairer agent sends a message to the SystemSchedular agent in order to learn the optimum time for the repair job.

```
  //Send message to SystemSchedular and get its reply

         Socket cSocket = null;
               ObjectOutputStream output;
               ObjectInputStream input;
       CaseMessage m = new CaseMessage("default", street, ctype);

       try {
             cSocket = new Socket("Figen", 3000);
             output = new ObjectOutputStream(cSocket.getOutputStream());
             output.writeObject((CaseMessage) m);
             output.flush();
             System.out.println("Case transferred to SystemSchedular...");

             input = new ObjectInputStream(cSocket.getInputStream());
             TimeMessage tm;
              tm = (TimeMessage)input.readObject();
                     optimumtime = tm.OpTime;
                     System.out.println("Reply received from SystemSchedular...");

             output.close();
             input.close();
             cSocket.close();

          } catch (UnknownHostException e) {
              System.err.println("Don't know about host computer!");
              System.exit(1);
          } catch (IOException e) {
              System.err.println("Couldn't get I/O for the connection!");
              System.exit(1);
          } catch (ClassNotFoundException e) {
              System.err.println("ClassNotFoundException");
          }

          result.append("---REPAIR INFORMATION---"+ newline);
          result.append("TIMING:"+newline);
          result.append("Ideal Time for this job: "+ optimumtime + newline);
```

Figure 6.17. Timing request and reply receive

At this moment, SystemSchedular listens on 3000. After receiving the case message from the RoadRepairer agent, it simply assigns a time for the paving work at hand by examining the AGENDA database following the rule given below and replies back.

```
public void assignTime(){

       String url = "jdbc:odbc:StreetManagement";
        Connection con;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
```

```
            con = DriverManager.getConnection(url, "manager", "open");
            PreparedStatement stQuery = con.prepareStatement(
                      "SELECT Type,PlannedFinish FROM AGENDA WHERE "+
                      "OnStreetCode LIKE ? ORDER BY PlannedFinish");
                      stQuery.setString(1, street);

            ResultSet rs = stQuery.executeQuery();
            while (rs.next()) {
                      String t = rs.getString("Type");
                      Date pf = rs.getDate("PlannedFinish");
                      Calendar c = Calendar.getInstance();
                      c.add(Calendar.DAY_OF_MONTH, +1);
                      java.util.Date tomorrow = c.getTime();

       /*Since we know in this sample the case is a paving case, we ignore
       other time assignment altervatives...*/
                      if (t.equals("infrastructure"))optime = "[ "+pf+" ]";
                      else optime = "[ "+tomorrow+" ]";
                      }

       } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
       }
       System.out.println("Time assigned for the current case..." );
}
```

Figure 6.18. Time assignment

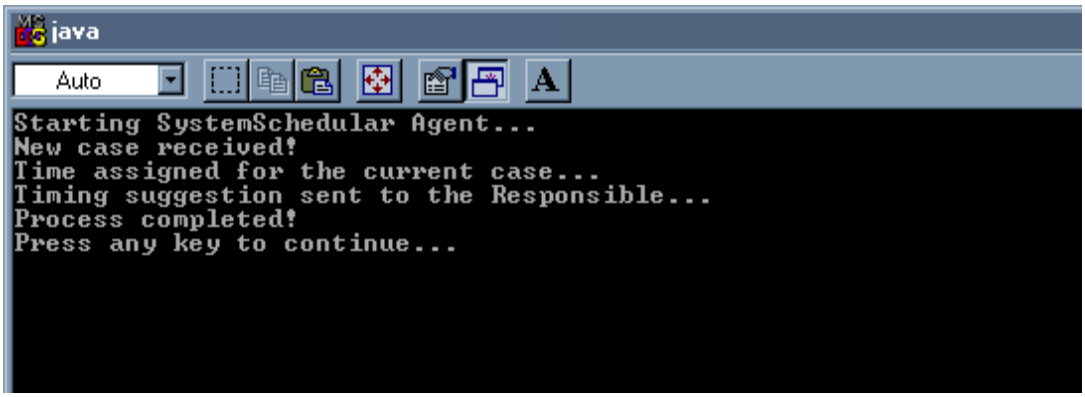

Figure 6.19. Output screen of the SystemSchedular agent

After receiving the time suggestion from the SystemSchedular agent, RoadRepairer also gets standards related to the current paving work from STANDARDS database and displays all those information on the user interface.

```
public void getStandards(){
       result.append("STANDARDS INFORMATION:"+newline);
       String url = "jdbc:odbc:StreetManagement";
        Connection con;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "manager", "open");
            PreparedStatement stQuery = con.prepareStatement(
                      "SELECT MaterialType, WorkerType FROM STANDARDS WHERE "+
                      "StreetCode LIKE ?");
                      stQuery.setString(1, street);
```

```
            ResultSet rs = stQuery.executeQuery();
            while (rs.next()) {
                      String m = rs.getString("MaterialType");
                      String w = rs.getString("WorkerType");
                      result.append("Material Standarts for this job: "
                      +m+newline);
                      result.append("Worker Standarts for this job: "
                      +w+newline);}

               con.close();

        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
    }
```

Figure 6.20. Work standards attainment

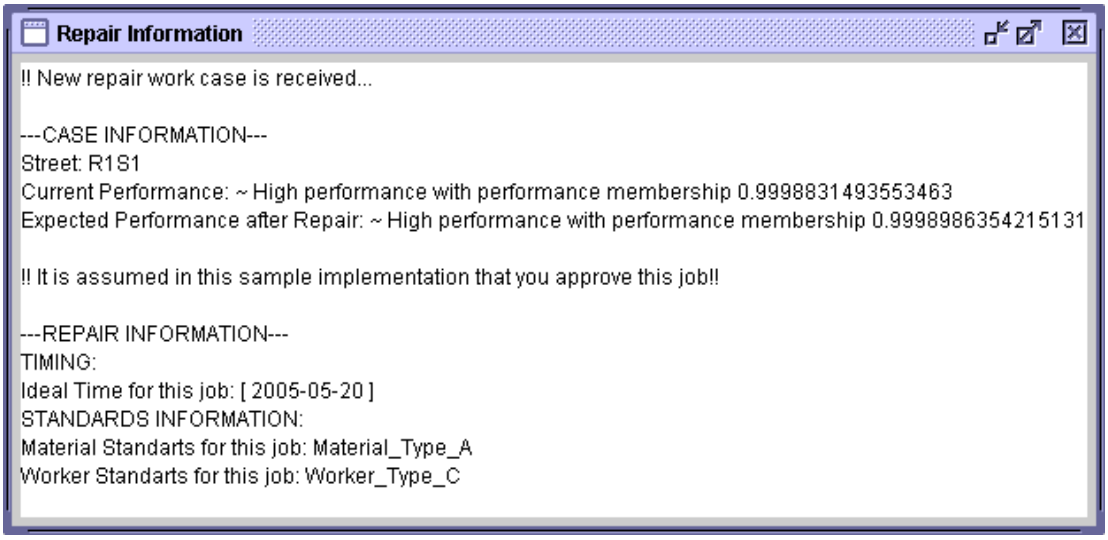

Figure 6.21. Case information display from the RoadRepairer agent user interface

Finally, RoadRepairer sends timing information to the PublicListener agent and completes its duty.

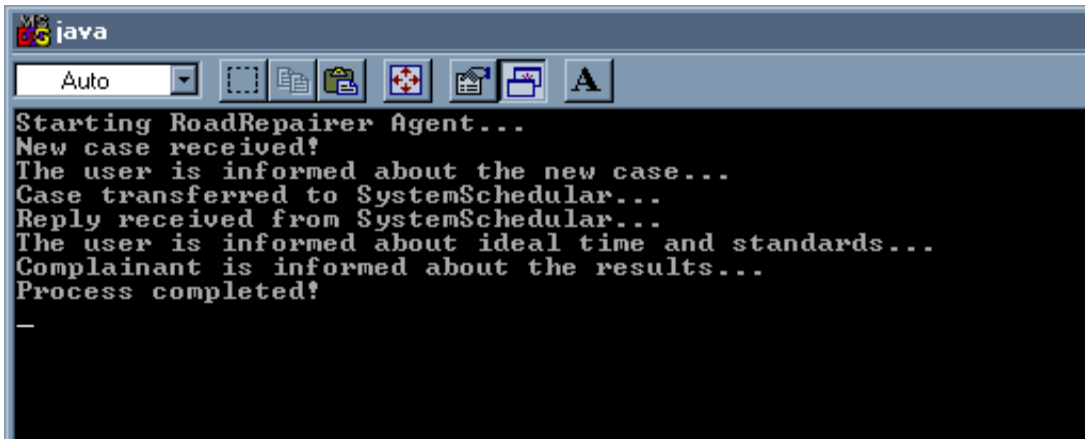

Figure 6.22. Output screen of the RoadRepairer agent

After it has sent the fist message about the new complaint submitted, the PublicListener agent has been waiting on port 6000. Now, it gets the final news and

informs the complainant, displaying the information it received on the user interface screen.

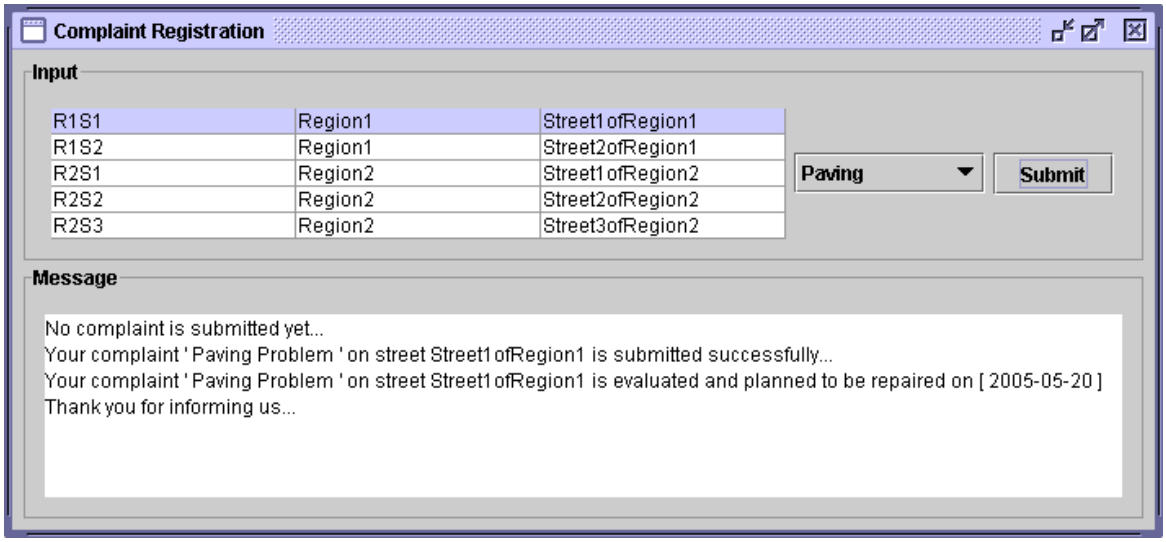

Figure 6.23. Case information display from the PublicListener agent user interface

```
    //Receive message from Responsible and inform the complainant user

             ServerSocket serverSocket = null;
             try {
                 serverSocket = new ServerSocket(6000);
             ...
             }

             Socket clientSocket = null;
             try {
                 clientSocket = serverSocket.accept();

                 output = new ObjectOutputStream(clientSocket.getOutputStream());
                 output.flush();
                 input = new ObjectInputStream(clientSocket.getInputStream());

                 TimeMessage tm;
                        tm= (TimeMessage)input.readObject();
                        PlannedTime= tm.OpTime;

                 output.close();
                 input.close();
                 clientSocket.close();
                 serverSocket.close();

             ...
             }

             System.out.println ("Case information is received from Responsible and
             displayed...");

             result.append("Your complaint ' " +
                  Complaint + " ' on street " + StreetName +
                  " is evaluated and planned to be repaired on "+PlannedTime+newline);
        ...
    }
```

Figure 6.24. Final case information transfer to the PublicListener agent

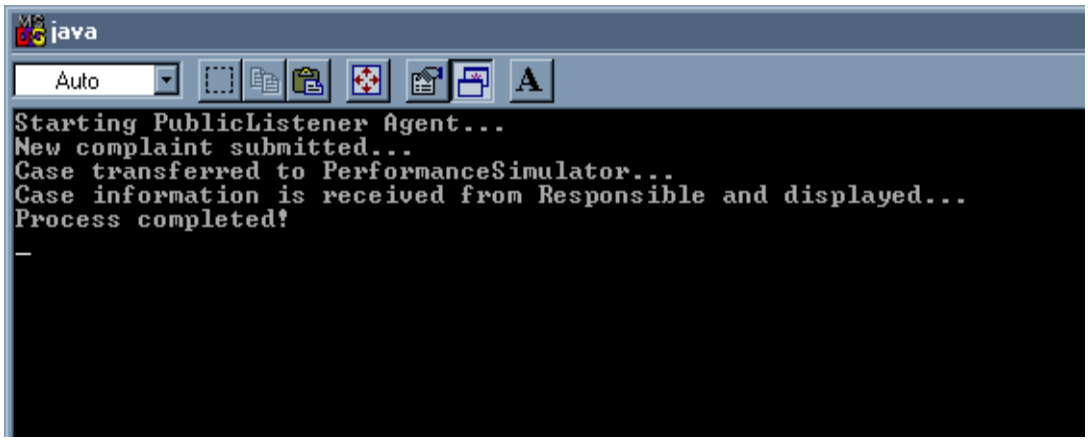

Figure 6.25. Output screen of the PublicListener agent

The overall process subject to the sample implementation is finalized when the thank message is displayed on the PublicListener agent user interface.

```
        result.append("Thank you for informing us..." );
        System.out.println("Process completed!");
```

Figure 6.26. Thank message display

7. CONCLUSIONS

Street management case is a characteristic example given for the lack of coordination among several responsible units. A classical bureaucratic coordination mechanism such as a separate central coordination unit seems to be ineffective. A computer-based support is certainly more appropriate to provide real-time coordination. In practice, a multiagent decision support tool seems to be one of the best ways to improve decisions made within such a distributed human organization.

In this study, we suggested that, decisional effectiveness in a cooperative distributed human system can be increased by distributed decision support tools at tactical and strategic levels. A new distributed decision support system based on multiagent architecture is proposed, in which human agents are also actively involved in the decision process. The performance model of the system was established using fuzzy cognitive map approach. The applicability of the proposed system was tested on a sample implementation program developed in Java environment and street management is selected as the application domain since it is a good example of distributed decision making.

The sample implementation was successfully realized. Thus, we could show the implementability of the decision support system we proposed: how computer-based support can provide real-time coordination and how can a decision support system help a decision maker make better decisions, e.g. calculating and representing effects on system performance. Besides, there is a wide range of alternatives for future work. One of those is a further work on the street management DSS proposed here: Sample implementation should be widened, database should be enriched, less assumptions and less neglects should be made in order to produce more useful support systems for real life applications. Another alternative is to apply a similar structure for a different implementation field including a distributed human decision system. Future work is encouraging, since the need for distributed decision systems will increase as decision problems become more complex.

REFERENCES

Aguirre, J.L., Brena, R., Cantu, 2001. F.J., Multiagent-Based Knowledge Networks, Expert Systems with Applications 20, pp.65-75.

Anderson, J., 2000. Agent breadth in a tool for distributed multi-agent system development, in Proceedings of the OOPSLA Workshop on Experiences with Autonomous Mobile Objects and Agent-Based Systems, Minneapolis.

Antona, M., Bousquet, F., LePage, C., Weber, J., Karsenty, A., Guizol, P., 1998. Economic Theory of Renewable Resource Management: A Multi-Agent System Approach, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Barber, K.S., MacMahon, M.T., and Martin, C.E., 2001. Distributed Software Decision Support Systems for Heterogeneous Coordination in Chemical and Biological Response, Proceedings from: Scientific Conference on Chemical and Biological Defense Research.

Bigus, J.P., Bigus, J., 1998. Constructing Intelligent Agents with Java, Wiley Computer Publishing.

Biró, M., Kovács, L., 1994. Standardization of Communication between Organizational Decision Units CON'94, 9th Austrian-Hungarian Conference in Informatics, Linz, Austria.

Brehmer, B., 1990. Distributed Decision Making: Some Notes on the Literature, in eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Campbell, K.C., Cooper, W.W., Greenbaum, D.P., Wojcik, L.A.., 2000. Modelling Distributed Human Decision Making in Traffic Flow Management Operations, 3rd USA/Europe Air traffic Management R&D Seminar paper.

Carley K.M., Gasser, L., 1990. Computational Organization Theory, in ed:Weiss, G., Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Castelfranchi, C., 1998. Simulating with Cognitive Agents: The Importance of Cognitive Emergence, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Chen, Zhengxin, 2000. Computational Intelligence for Decision Support, CRC Press.

Cohen, D., 2003. Distributed Intelligent Agents for Decision Making at Local Distributed Energy Resource (DER) Levels, U.S. Department Of Energy Electric Distribution Transformation Program Peer Review.

Cuena, J., Ossowski, S., 1990. Distributed Models for Decision Support, in ed: Weiss, G., Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Doran, J., 1995. Simulating Societies using Distributed AI, Social Science Microsimulation: A Challenge to Computer Science Seminar Notes.

Doyle & Thomason, 1999. Background to Qualitative Decision Making, AI Magazine.

Gachet, A., 2002. A New Vision in Distributed Decision Support Systems, in eds: Adam F., Brézillon P., Humphreys P., Pomerol, J., Decision Making and Decision Support in the Internet Age, proceedings of the DSIage 2002 Conference, Oak Tree Press.

Gachet, A., Haettenschwiler, P., 2003. Disributed Decision Support Systems – A federalist Model of Cooperation, in ed: Bisdorff, R., Human Centered Processes – Distributed Decision Making and Man-Machine Cooperation, proceedings of the 14th MINI EURO Conference.

Gao, Hui-Min, Jian-Chao, Z., Yu-Bin, Xu, Guo-Ji, S., 2003. Multiagent decision support system, Machine Learning and Cybernetics, 2003 International Conference, Vol. 3 , 2-5, pp.1945 – 1950.

Ghosh, S., 2001. Understanding complex, real world systems through asyncronous, distributed decision-making algorithms, The Journal of Systems and Software 58, s.153-167.

Goddard, S., Zhang, S., Waltman, W.J., Lytle, D., Anthony, S., 2002. A Software Architecture for Distributed Geospatial Decision Support Systems, Proceedings of the Second National Conference on Digital Government Research, Los Angeles, California, pp. 45-52.

Hannoun, M., Sichman, J.S., Boissier, O., Sayettat, C., 1998. Dependence Relations Between Roles in a Multiagent System, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Hashimoto, T., 1998. Dynamics of Internal and Global Structure through Linguistic Interactions, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Huhns M.N., Stephens, L.M., 1990. Multiagent Systems and Societities of Agents, in ed: Weiss, G., Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Kok, J. R., **Spaan, M. T. J., Vlassis, N**., 2003. Multi-robot decision making using coordination graphs, in Proc. 11th Int. Conf. on Advanced Robotics, Coimbra, Portugal.

Koopman, P. and Pool, J., 1990. Organizational Decision Making: Models, Contingencies and Strategies, in: Eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Kraus, S., 2001. Strategic Negotiation in Multi-Agent Environments,MIT Press.

Laichour, H., **Maouche, S., Mandiau, R**., 2002. Distributed decision-making: application to an urban transport network, Systems, Man and Cybernetics, 2002 IEEE International Conference, Vol. 6.

Lanir, Z., 1990. The Reasonable Choice of Disaster, in eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Lemaître, C., **Excelente C.,** 1998. Multi-Agent Network for Cooperative Work. In Expert Systems With Applications: An International Journal. Elsevier Science Publishers. Vol. 14, pp. 117-127.

Leplat, J., 1990. Organization of Activity in Collective Tasks, in eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Liu, Jiming, 2001. Autonomous Agents and Multiagent Systems, World Scientific Publishing.

Mohr, S.T., 1997. Software Design for a Fuzzy Cognitive Map Modelling Tool, Master's Project, Rensselaer Polytechnic Institute.

Nourani, C.F., 1999. Multiagent AI implementations an emerging software engineering trend, Engineering Applications of Artificial Intelligence 12, pp.37-42.

Ossowski, S., Fernandez, A., Serrano, J.M. et al., 2004. Designing Multiagent Decision Support System: The Case of Transportation Management, AAMAS 2004.

Palmgren, M., 2004. Supporting Decision Making in Distributed Design, Master's Thesis, 2004:202 CIV, Lulea University of Technology.

Parunak, H.V.D., 1990. Industrial and Practical Applications of DAI, in ed: Weiss, Gerhard, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Parunak,V.D., Savit, R., Riolo, R., 1998. Agent-Based Modelling vs. Equation-Based Modelling: A Case Study and Users' Guide, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Commission for Architecture and the Built Environment (CABE), 2002. Paving the way: How to Achieve Clean, Safe and Attractive Streets.

Pindur, W., **Rogers, S.E**., 1995. The history of management: a global perspective, Journal of Management History, Vol. 1 No. 1, pp. 59-77.

Qureshi, S., **Hlupic, V.**, 2000. Managing Knowledge in a Distributed Decision Making Context: The Way Forward for Decision Support Systems, ERIM Report Series Research In Management, No: ERS-2000-16-LIS.

Ramos V., Fernandes, C., Rosa, A.C., 2005. Social Cognitive Maps, Swarm Collective Perception and Distributed Search on Dynamic Landscapes, paper submitted to Brains, Minds & Media – Journal of New Media in Neural and Cognitive Science.

Rasmussen, J., 1990. Modelling Distributed Decision Malking, in Eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Rogalski, J., 1990. Distributed Decision Making in Emergency Management: Using a Method as a Framework for Analysing Cooperative Work and as a Decision Aid, in eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Rogova, G.L., **Menon, R.**, 1998. Learning in Distributed Systems for Decision Making, Final Technical Report – Task 3, Center for Multisource Information Fusion.

Ross, T. , 1998. Fuzzy Logic with engineering Applications, McGraw-Hill.

Rusmevichientong, P., Van Roy, B., 2000. Decentralized Decision Making in a Large Team with Local Information, Games and Economic Behavior.

Sage, A.P., 1991. Decision Support Systems Engineering, John Wiley&Sons.

Schleiffer, R, 2005. An Intelligent Agent Model, European Journal of Operational Research 166, pp. 666–693.

Schmidt, K., 1990. Cooperative Work: A Conceptual Framework, in eds: Rasmussen, J., Brehmer, B., Leplat, J., Distributed Decision Making, John Wiley&Sons.

Schneeweiss, C., 1999. Hierarchies in Distributed Decision Making, Springer-Verlag.

Schneeweiss, C., 2003. Distributed Decision Making – a unified approach, European Journal of Operational Research 150, s.237-252.

Shaalan, K., El-Badry, M., Rafea, A., 2004. A multiagent approach for diagnostic expert systems via the internet, Expert Systems with Applications 27, pp. 1–10.

Silver, M.S., 1991. Systems That Support Decision Makers, John Wiley&Sons.

Sen, S., 1997. Multiagent Systems: Milestones and New Horizons, Trends in Cognitive Sciences, Vol.1, No:9.

Servat, D., Perrier, E., Treuil, J.P., Drogoul, A., 1998. When Agents Emerge From Agents: Introducing Multi-scale Viewpoints in Multiagent Simulations, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Singh, S.P., 1994. Multiagent Systems: A Theoretical Framework for Intentions, Know-How and Communications, Springer-Verlag.

Sumpter, D.J.T., Broomhead, D.S., 1998. Formalizing the Link Between Worker and Society in Honey Bee Colonies, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation.

Swaminathan, J.M., Smith, S.F., Sadeh, N.M., 1998. Modelling Supply Chain Dynamics: A Multiagent Approach, Decision Sciences, Vol 29, N: 3, pp.607-632.

Verhagen, H., ACTS in Action: Sim-ACTS – A Simulation Model Based on ACTS Theory, in eds: Sichman,J.S., Conte, R., Gilbert, N., Springer, Multiagent Systems and Agent Based Simulation, 1998.

Wolf, William B., 1995. Decision processes as analysed by Chester I. Barnard, Journal of Management History, Vol. 1 No. 4, pp. 4-112.

Xirogiannis, G., Stefanou, J., Glykas, M., 2004. A fuzzy cognitive map approach to suport urban design, Expert Systems with Applications, 26, pp.257-268.

Ydstie, B.E., 2005. Distributed decision making in complex organizations: The adaptive enterprise, Computers and Chemical Engineering 29, pp. 11–27.

Yokoo, M., Ishida, T., 1990. Search Algoritms for Agents, in ed: Weiss, Gerhard, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Zhang,W.-R., 1994. Bipolar qualitative/quantitative models for cognitive modeling and multiagent decision analysis, Systems, Man, and Cybernetics, 'Humans, Information and Technology', IEEE International Conference, Vol. 3, 2-5, pp. 2755 – 2760.

The Java Tutorial: A Practical Guide for Programmers, http://java.sun.com/docs/books/tutorial/

agentMom Project Source Code, http://www.cis.ksu.edu/~sdeloach/ai/software/agentMom_2.0/home.html

CURRICULUM VITAE

Figen ÖZTOPRAK was born in İstanbul in 1982. She graduated from Pertevniyal High School in 1999 and from İstanbul Technical University Industrial Engineering department in 2003. She continues her master study also in İstanbul Technical University Industrial Engineering programme.