

**PERFORMANCE ANALYSIS OF STATIC AGENT ARCHITECTURE
WITH CENTRALIZED MATCHMAKING**

ZAFER ERENEL

IŞIK UNIVERSITY

2005

**PERFORMANCE ANALYSIS OF STATIC AGENT ARCHITECTURE WITH
CENTRALIZED MATCHMAKING**

by

Zafer Erenel

B.S., Istanbul University, 1999

Submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Işık University

2005

Approved for the University Committee on Graduate Studies.

.....

Prof. Dr. Hüsnü A. Erbay
Graduate School of Science and Engineering Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

.....

Prof. Dr. Ahmet Aksen
Computer Engineering Department Head

This is to certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and qualify as a thesis for the degree of Master of Science.

.....

Prof. Dr. Selahattin Kuru
Thesis Supervisor

Examining Committee:

Prof. Dr. Selahattin Kuru

Doç. Dr. Seyhun Altunbay

Yrd. Doç. Dr. Tamer Dağ

ABSTRACT

Large- scale network environments such as the internet possess the characteristics of distributed data, distributed access and distributed control. This gives users a powerful mechanism for building and integrating large amounts of distributed information from diverse resources. However few support tools have been developed for users to sift through this vast amount of information.

In this thesis, we advocate the integration of two entities; Static agents to create a user profile, and information integration architecture to provide the desired information. Text-Based information is the main concern due to its high significance in our daily lives. Thus, information integration architecture will gather and intelligently combine information from multiple agents and present the user with combined information using a task agent.

ÖZET

İnternet ve ona benzer büyük boyuttaki ağ yapıları dağıtık veri, dağıtık erişim ve dağıtık kontrol imkanları sağlar. Bu sayede kullanıcılar, büyük miktarlardaki veriyi farklı kaynaklardan alıp birleştirme ve yapılandırma şansına sahip olurlar. Ancak bu büyük boyutlardaki veriyi belli bir amaca göre eleyebilecek destek araçları sınırlı sayıdadır.

Bu tezde iki varlığın bir araya getirilmesi savunulmaktadır. Durağan ajanlar ile kullanıcı profili yaratmak ve bilgi entegrasyon mimarisi ile gerekli bilgiyi kullanıcılara sağlamak hedeflenmiştir. Karakter tabanlı bilgi günlük hayatımızdaki öneminden dolayı tezin ürettiği çözüm içinde yer almaktadır. Böylece bilgi entegrasyon mimarisi farklı durağan ajanlardan bilgi toplayıp, derleyip talep eden diğer ajanlara sunacaktır. Bunu yaparken görev ajanı kullanılacaktır.

ACKNOWLEDGEMENTS

I would like to thank my senior supervisor, Prof. Selahattin Kuru, for all the supports he has given to me since I entered into this graduate program. I am very grateful for the inspiring discussions that led to this thesis. I am also grateful to all the people in the Institute of Science and Engineering at ISIK University, for their constructive discussions and friendships. Lastly, I would like to thank my parents, for their love, encouragement, and support.

TABLE OF CONTENTS

| | | |
|---|--|------|
| Abstract | | iii |
| Özet | | iv |
| Acknowledgements | | v |
| Table of Contents | | vi |
| List of Figures | | viii |
| List of Tables | | ix |
| 1 Introduction | | 1 |
| 2 Software Agents and Software Agent Systems | | 3 |
| 2.1 Software Agents..... | | 3 |
| 2.2 Attributes of Software Agents..... | | 4 |
| 2.3 Types of Software Agents..... | | 8 |
| 2.4 Examples of Software Agent Systems..... | | 9 |
| 2.5 Performance Modeling of Software Agent Systems..... | | 12 |
| 2.6 Agent Based Simulations..... | | 12 |
| 2.7 Real System Using Agent Mickey..... | | 16 |
| 3 Matchmaking Mechanisms | | 19 |
| 3.1 Centralized Matchmaking..... | | 19 |
| 3.2 Localized Matchmaking..... | | 19 |
| 3.3 Market Mechanism..... | | 20 |
| 3.4 Comparison of Matchmaking Mechanisms..... | | 21 |
| 4 Agent Mickey | | 23 |
| 4.1 C# Programming Language..... | | 23 |
| 4.2 Visual Studio IDE (Integrated Development Environment) and RAD..... | | 23 |
| 4.3 SQL (Structured Query Language)..... | | 24 |
| 4.4 SQL Server 2000..... | | 25 |
| 4.5 User Agent..... | | 26 |
| 4.6 Task Agent..... | | 30 |

| | | |
|----------|---|-----------|
| 5 | Performance Analysis..... | 36 |
| | 5.1 Performance Factors..... | 36 |
| | 5.2 Task Agent Real-Time Performance..... | 36 |
| 6 | Conclusions and Future Work..... | 44 |
| | 6.1 Conclusions..... | 44 |
| | 6.2 Future Work..... | 45 |
| | References..... | 46 |
| | Appendix A: CD Containing Thesis Text and Source Code of Agent Mickey Platform | |

LIST OF FIGURES

| | |
|--|----|
| Figure 2.1 Scope of Intelligent Agents..... | 5 |
| Figure 2.2 Franklin and Graesser’s Agent Taxanomy..... | 6 |
| Figure 2.3 Cooperating Systems with a Single Agent as a Global Planner..... | 7 |
| Figure 2.4 Static Agents..... | 8 |
| Figure 2.5 Mobile Agents..... | 9 |
| Figure 2.6 Information Integration Architecture..... | 16 |
| Figure 3.1 Centralized Agents..... | 19 |
| Figure 3.2 Localized Agents..... | 20 |
| Figure 4.1 Agent Mickey Personal Information..... | 26 |
| Figure 4.2 Agent Mickey Item Information..... | 27 |
| Figure 4.3 Agent Mickey on Task Bar..... | 27 |
| Figure 4.4 Agent Mickey Search..... | 28 |
| Figure 4.5 Agent Mickey Periodical Search Results (No results found)..... | 29 |
| Figure 4.6 Agent Mickey Periodical Search Results (Results found)..... | 29 |
| Figure 4.7 Agent Mickey Immediate Search Results..... | 30 |
| Figure 4.8 Task Agent Flow Chart..... | 31 |
| Figure 4.9 Agent Server Database Tables..... | 32 |
| Figure 4.10 Database Work Flow..... | 33 |
| Figure 4.11 Static Agent Architecture Map..... | 35 |
| Figure 5.1 Performance Chart of a Task Agent for 10 Agents..... | 37 |
| Figure 5.2 Performance Limits of a Task Agent for 10 Agents..... | 38 |
| Figure 5.3 Performance Chart of a Task Agent for 1000 Agents..... | 39 |
| Figure 5.4 Performance Limits of a Task Agent for 1000 Agents..... | 40 |
| Figure 5.5 Performance Chart of a Task Agent for a range of agents from 1 to 1,000,000,000..... | 41 |

LIST OF TABLES

| | |
|--|----|
| Table 4.1 SQL Queries for the Work Done..... | 34 |
| Table 5.1 Task Agent Performance for 10 Agents..... | 36 |
| Table 5.2 Task Agent Performance Limits for Daily Updates (10 Agents)..... | 38 |
| Table 5.3 Task Agent Performance for 1000 Agents..... | 39 |
| Table 5.4 Task Agent Performance Limits for Daily Updates (1000 Agents)..... | 40 |
| Table 5.5 Performance of Task Agent (50 products versus 5000 products)..... | 40 |
| Table 5.6 Performance of Task Agent for a range of agents from 1 to 1,000,000,000..... | 41 |
| Table 5.7 Task Agent Performance for 5000 Agents..... | 42 |

CHAPTER 1

INTRODUCTION

In this thesis, performance analysis of static agent architecture has been made using a central matchmaking mechanism. Real system has been developed by using static agent architecture. Static agents reflect the behaviors [1] of internet users on buying and selling items. C# Programming Language has been used for development of agents and SQL Server 2000 database management system has been used for harboring data. The goal of the thesis is to find performance limitations of a task agent located on the Agent Server. Sample database has been designed in SQL Server 2000 for storing data of multiple agents. Random data has been sent to Agent Server. Execution time of the task agent has been studied by creating different numbers of static agents. Matchmaking has been the only purpose of the task agent. An approximate ratio has been found between the number of static agents and the execution time of the task agent.

Agent technique is one of the important technologies developed to support the Internet applications. Even if the users are off-line, the agents are still active in the world of computer network and play the roles that their users assigned. Hence the term 'software agent' might best be viewed as an umbrella term that covers a range of other more specific and limited agent types [2]. Agent can be viewed as software that assists or represents the behaviors of users in the world of computer network [3].

Multiple Agent approach would be sensible for information that is physically distributed. The notion of ownership of information and strategies in the application are important here. When information is distributed over different organizational entities, no single entity can have access to all the information. Particular examples include traders in a marketplace.

Multiple Agent Systems use object-oriented, expert systems and distributed computing technologies to implement applications. Multiple Agent Systems will not replace these technologies, but provide a different way of using them to tackle new kinds of problems. Current inter-organization applications do not require Multiple Agent Systems because they support human communication to carry out tasks. There are barriers to be crossed before a broader use of

multi-agent systems technology. For example, we need to have a stock of development methods (such as languages, etc.) for Multiple Agent systems for them to be a widely-accepted technology.

This thesis is organized as follows. In Chapter 2 the term ‘software agent’ and its attributes are discussed with an overview of developed applications. In chapter 3, matchmaking mechanisms are introduced. Key points in creating matchmaking mechanisms are examined. In chapter 4, tools used and real system Agent Mickey are introduced. Functional features of Agent Mickey are explained in detail. In chapter 5, performance issues regarding task agent are covered. In chapter 6, future work areas are defined regarding static agents.

CHAPTER 2

SOFTWARE AGENTS AND SOFTWARE AGENT SYSTEMS

2.1 Software Agents

Since the beginning of recorded history, people have been fascinated with the idea of non-human agents. Nonetheless, the public image of artificially intelligent creatures often has been a nightmare. Everyday experiences of computer users with annoying bugs, incomprehensible features, and dangerous viruses boost the fear that the software powering autonomous creatures will pose even more problems in the future. Such concerns cannot be ignored in the design of software agents.

Norman [4] observes that perhaps the most relevant predecessors to today's intelligent agents are control devices, including factory control and the automated takeoff, landing, and flight control of aircraft. However, the agents are now being considered in different ways.

Significantly, for the moment, the momentum seems to have shifted from hardware to software, from the atoms that comprise a mechanical robot to the bits that make up a digital agent [5].

The idea of an agent originated with John McCarthy in the mid 1950's, and the term was invented by Oliver G. Selfridge a few years later, when they were both at the Massachusetts Institute of Technology. They had in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a software robot living and doing its business within the computer's world [6].

As agents of many varieties have grown, there has been an explosion in the use of the term without a corresponding consensus on what it means.

Some can be scheduled in advance to perform tasks on a remote machine, some accomplish low-level computing tasks while being instructed in a higher-level of programming language or script [7]. Some abstract out or encapsulate the details of differences between information sources or computing services [8]. Some implement a primitive or aggregate cognitive function [9,10].

Some show characteristics of distributed intelligence [11]. Some serve a mediating role among people and programs [12,13,14].

Some perform the role of an 'intelligent assistant' [15,16]. Some can migrate in a self-directed way from computer to computer [17]. Some speak an agent communication language [18] and some are viewed by users as displaying intentionality and other aspects of 'mental state' [19].

This insight helps us understand why coming up with one definition of agent is so difficult. One person's intelligent agent is another person's smart object. Today's smart object is tomorrow's dumb program. The key distinction is in our expectations and our point of view.

The American Heritage Dictionary defines an agent as 'one that acts or has the power or authority to act or represent another' or 'the means by which something is done or caused'. As in the everyday sense, we expect a software agent to act on behalf of someone to carry out a particular task which has been delegated to it. In fact, the agents helping humans need knowledge their owners with regard to their particular tasks. A good travel agent blends knowledge about hotels and restaurants with knowledge about us. A real estate agent builds a model of us from a succession of houses that fit our taste with varying degrees of success. Telephone-answering agents, news agents, or electronic-mail-managing agents are familiar examples. What they all have in common is the ability to model us [20].

It is perfectly logical to treat a light switch as a very cooperative agent with the capability of transmitting current when it believes that we want it transmitted. We communicate our desires by clicking on the button. It has a simple mechanism but can be treated as an agent.

2.2 Attributes of Software Agents

All this being said, most software agents today are fairly fragile and special purpose programs. Consistent with the requirements of a particular problem, each agent might possess to a greater or lesser degree attributes like the ones enumerated in Etzioni and Weld and Franklin and Graesser [21,22].

- Reactivity: the ability to selectively sense and act.
- Autonomy: goal-directedness, proactive and self-starting behavior.

- Collaborative behavior: can work in concert with other agents to achieve a common goal.
- ‘Knowledge-level’ [23] communication ability: the ability to communicate with persons and other agents with language more resembling humanlike ‘speech acts’ than typical symbol-level program-to-program protocols.
- Inferential capability: can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility; goes beyond the information given, and may have explicit models of self, user, situation, and/or other agents.
- Temporal continuity: persistence of identity and state over long periods of time
- Personality: the capability of showing the attributes of a ‘believable’ character such as emotion
- Adaptivity: being able to learn and improve with experience
- Mobility: being able to migrate in a self-directed way from one host platform to another.

An influential white paper from IBM [24] described intelligent agents in terms of a space defined by the three dimensions of agency, intelligence, and mobility in Figure 2.1.

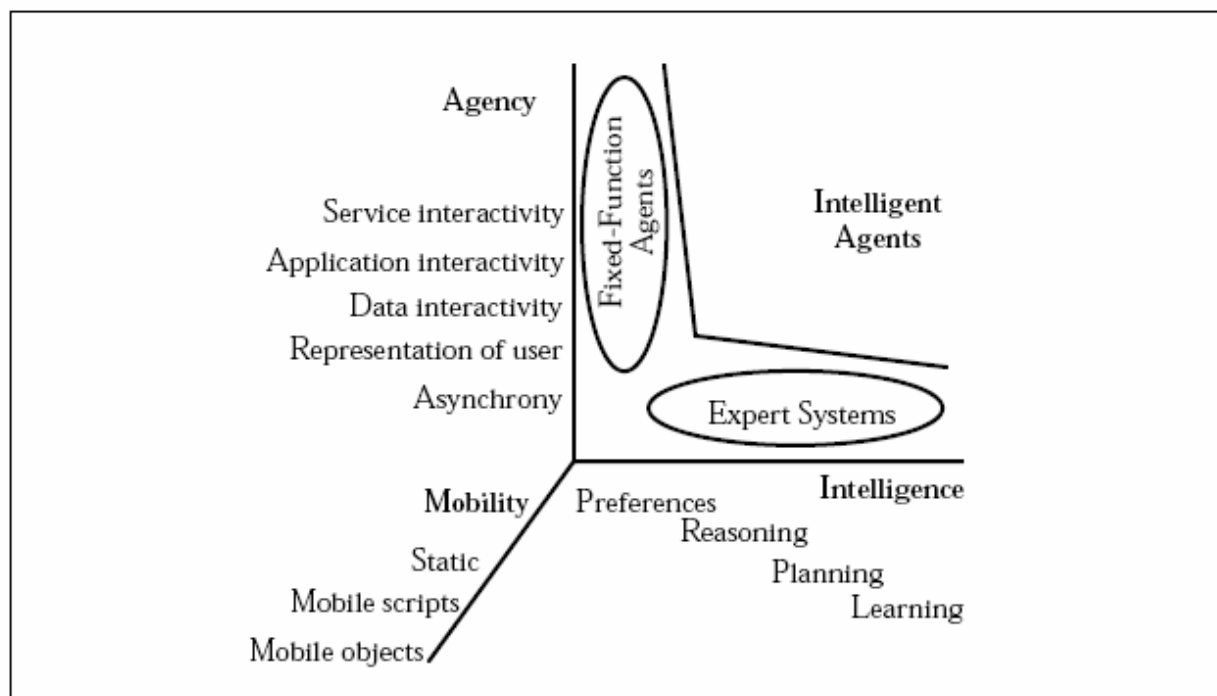


Figure 2.1 Scope of Intelligent Agents

Agency is the degree of autonomy and authority vested in the agent, and can be measured at least qualitatively by the nature of the interaction between the agent and other entities in the system.

The degree of agency is enhanced if an agent represents a user in some way. Intelligence is the degree of reasoning and learned behavior: the agent's ability to accept the user's statement of goals and carry out the task delegated to it. At a minimum, there can be some statement of preferences.

Mobility is the degree to which agents themselves travel through the network. Mobile scripts may be composed on one machine and shipped to another for execution. Mobile objects are transported from machine to machine and carrying accumulated data with them.

Franklin and Graesser [22] give their own: 'an autonomous agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.' Even a thermostat could qualify as an agent by this definition. They discuss various properties of agents and offer the taxonomy in Figure 2.2 as one that covers most of the examples found in the literature. Below this initial classification, they suggest that agents can be categorized by control structures, environments (e.g., database, file system, network, Internet), language in which they are written, and applications.

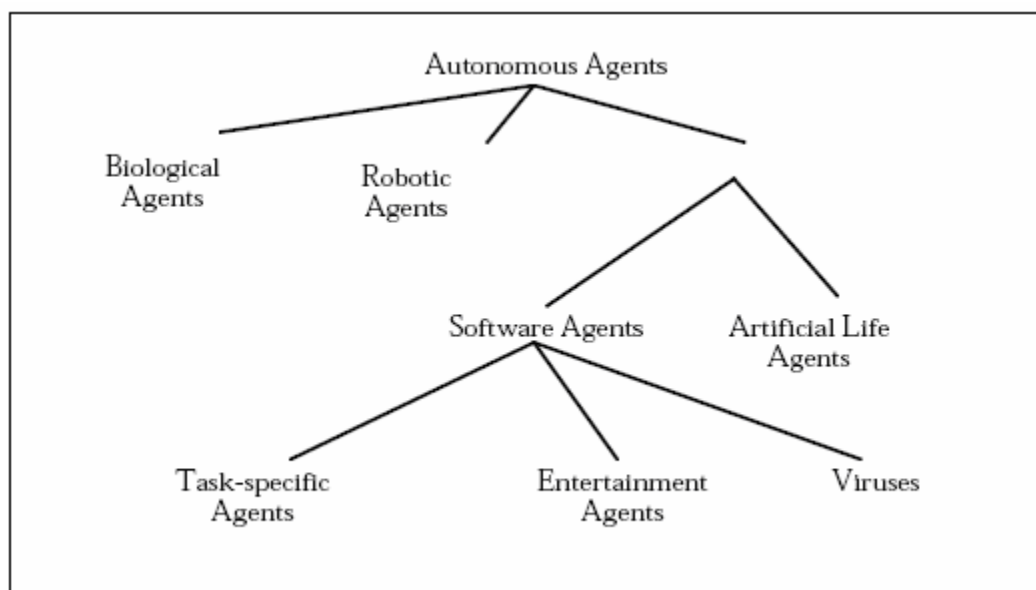


Figure 2.2 Franklin and Graesser's [22] Agent Taxonomy.

Current web-based searching and filtering agents are essentially one-time query answering mechanisms that are described by the computer science term server. Servers can be regarded as

agents. Similarly, Java applets agent-alike functions allow processes to run securely on foreign machines.

Petrie [25] argues the case for one specific class: typed-message agents. Typed-message agents are distinguished from other types of software by virtue of their ability to communicate as a community using a shared message protocol such as KQML. In the shared message protocol, some of the message semantics are typed and are independent of the applications.

Semantics of the message protocol necessitate that the transport protocol not be only client/server but rather a peer-to-peer protocol.

Time and experience will ultimately determine both the meaning and the life of the term ‘agent’. As public exposure to useful and technically doable implementations of agent software increases, the term will either come to mean something that everyone understands because they have seen many examples of it. What are unlikely to disappear are the motivations that have incited the development of agent-based software. To accomplish a high level of interoperability, an agent could function as a global resource manager [26] (Figure 2.3) as I did in my thesis. How far a single task agent might be workable as the number of cooperating systems grows is the main concern in my work.

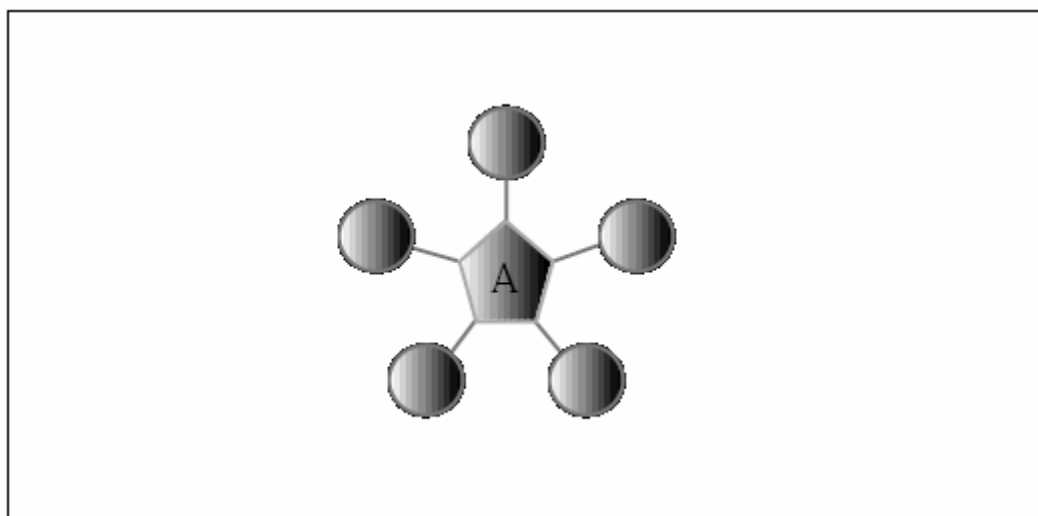


Figure 2.3 Cooperating Systems with single agent as a global planner. Connections Represent agent to application communication

2.3 Types of Software Agents

Over the past few years, multi agent systems have grown in popularity as a viable solution to complex, distributed information systems. The largest ‘open system’ is the Internet, with new content being generated every day. Multi agent systems are reported to have advantages such as faster execution time, less communication bandwidth, and greater reliability. The type of agents employed, either static or mobile, also have their own unique set of advantages and disadvantages. Today, a large body of quantitative data does not exist upon which system designers can base their ‘agent versus non-agent’ system design decisions [27]. If we take a closer look at the static agents in Figure 2.4, there is a steady environment in which movements of static agents are forbidden. They live and operate in the same computer. They can exchange messages with other agents. This can be done using an agent manager. A high level of interoperability requires a single agent as a global planner [26].

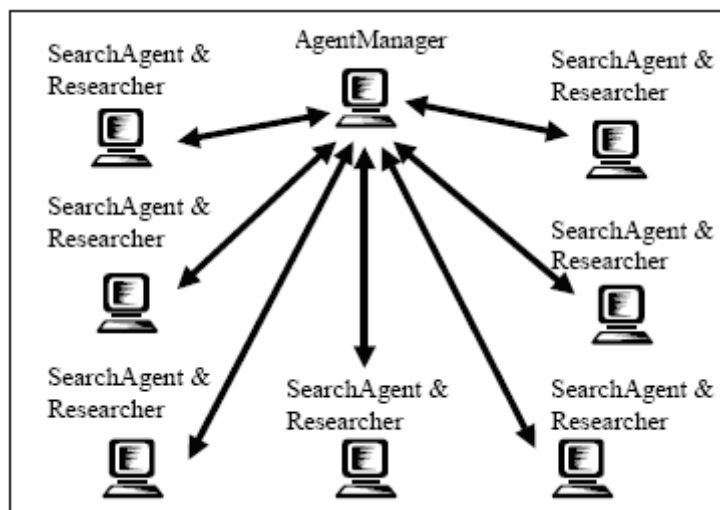


Figure 2.4 Static Agents

Mobile Agents, as seen in Figure 2.5, are on the move and can operate in different computers.

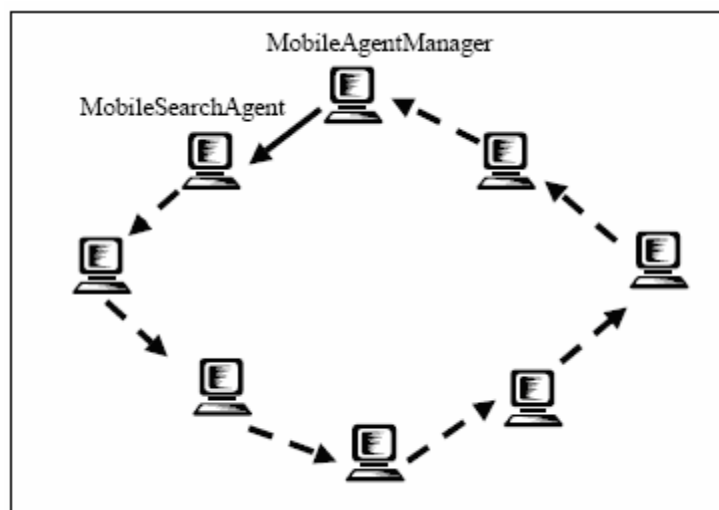


Figure 2.5 Mobile Agents

Almost thirty years after their initiation, the only widespread form of mobile software agents is malware. Each individual advantage of mobile agents can be realized equally or better by traditional approaches. Their adoption would require a very large number of network services and applications [28]. Yet, reality shows that the incentives offered by mobile agents have not been sufficient to stimulate their widespread deployment. Malicious hosts may tamper with agents, and malicious agents may attack their hosts. Java as the implementation platform for mobile agent systems falls short of providing the level of security that is required [29].

2.4 Examples of Software Agent Systems

Sakaguchi [30] proposed a shopping assistant agent for Web-shops. The shopping assistant agent works on a web server, a PCs-sale site. The agent has been applied to help potential buyers of built-to-order (BTO) PCs. There are three features of the interaction with this agent. (1) Two interaction channels: selection and conversation. (2) Flexible topic change: the user can trigger a new conversation flow even in the middle of a conversion. (3) Personalized Interaction: the interaction is personalized according to user behavior. There are three methods for the user to get advice from this agent. (1) Answer questions from the agent. (2) Ask the agent questions. (3) Refer to an additional message from the agent.

Lesser [31] developed an information gathering agent that processes Web documents to create product models and recommend purchases based on user selection criteria. The architecture of this information-gathering agent includes the following components:

- Resun (Resolving Sources of Uncertainty) planner: a blackboard-based interpretation planner
- Information extractors: text-extraction tools
- Document classifiers: text-processing filters
- Server information database: a local database of information sources stored
- Object database: a local database stores product information
- Design-to-Criteria (DTC) scheduler: an agent-control problem solver
- TAEMS modeling language: a Task, Analysis, Environment Modeling, and Simulation language
- Task assessor: a software module manages the interface between the Resun opportunistic planner and DTC scheduler.

Esmahi and Dini [32] proposed an intelligent agent market place that involves multiple mobile agents interact with other. The market place providers an infrastructure for exchanging offers and requests. Potential business partners can exhibit their services and available resources, search for offers from service provider, or send their own offers. There are three types of agent acts in the market place: the market place manager, the buyer agents and the seller agents.

Vermeulen and Bauwens [33] established standards of describing information and services that are offering to telecom network users on-line. They provided a generic service model and proposed a set of solutions using standard languages, such as KQML and SGML, within a distributed architecture hosting intelligent agents. The roles of agents in this proposed model are described as follows:

- The Personal Assistant agent: it will represent the consumers or end-users of the service.
- The On-line Provider agent: it will represent the sellers or content providers.
- The Service Broker agent: it will represent the brokers or business intermediates.
- The Resource Manager: it will represent the infrastructure providers.

- KQML (Knowledge Query and Manipulation Language) is chosen as the communication language between the different types of agents.
- KIF (Knowledge Interchange Format) and SQML (Standard Generalized Markup Language)/XML (extensible Markup Language) are chosen as content languages for exchanging information.

Lee [34] introduced an open infrastructure for agent-based E-commerce on the Internet. It is a multi-agent system in which six semiautonomous agents interact or work together to perform a user's goal. It is mainly composed of following agents:

- User Agent (UA) is a personalized learning interface agent that provides active assistance to the user and becomes smarter and more adaptive to its user,
- Customer Agent (CA) acquires user's request that which product he/she wants,
- Supplier Agent (SA) acquires user's request that which products he wants to sell and then advertises/unadvertised himself to the matchmaker agent (KBMA),
- Navigation Agent (NA) plays a role of commerce E-specific search engine. NA finds corresponding counterpart who can satisfy user's demand, and retrieves product of user's interest,
- Knowledge Base Management Agent manages the knowledge base – including Product Ontology, Agent Yellow Page, Label/Supplier Description, and so on, and Payment Agent (PA) provides a payment service for market transaction between customer and supplier.

Sohn and Yoo [35] also proposed architecture of E-marketplace based on mobile agent technologies. The major components of this architecture are described as follows:

- Management Agent for Conductor (MACD) is an agent that manages conductor. MACD supervises people joining or disjoining as members of this marketplace.
- Setup Agent for Member (SAMM) is an agent created by MACD when a user be to join as a member in the market. And the SAMM will migrate to user's context.
- Guide Agent for Member (GAMM) assists member to divide products.
- Management Agent for Provider (MAPR) handles product data for a provider member.
- Consignment Agent for Provider (CAPR) helps providers to consigning his products to be sold to shop.

- Management Agent for Consumer (MACR) handles information for a consumer member.
- Brokering Agent for Consumer (BACR) collects product data by visiting shop.

2.5 Performance Modeling of Software Agent Systems

Throughput (in terms of transactions per minute), response time, and the number of concurrent users and tasks are important performance indicators in distributed systems [36]. They are equally vital in multi-agent systems. However, there are other important factors introduced by the agent level which have costs, and therefore affect performance.

The manner in which agents are organized in a multiple agent systems has a strong impact on both the computational and communication overheads. For example, there can be a flat organization where an agent must always negotiate the terms and conditions of every service. This requires joint activity with other agents. In a hierarchical society, behavior of each agent engaged in a joint activity with other agents follows predefined roles. The latter is less expensive. The agent's knowledge model requires sophisticated data structures (i.e. storage costs) and their manipulation (i.e. computational costs).

The basic performance indicators in MAS s are computational costs and throughput.

For a multi-agent model, the variables that affect performance include the number of agents, the number of concurrent tasks/goals the agents are carrying out and the organization of the agents. I have worked on the number of agents (The number of tasks increases linearly with the number of agents in this thesis.) with a centralized static agent organization in this thesis.

2.6 Agent-Based Simulations

The rising of computational power have made computer simulations more accurate and cheaper, and at the same time they have increased the speed of the simulation process. This is particularly meaningful for those situations where too many factors and parameters have to be taken into account and there are so many interactions between agents.

The Braess's Paradox [37] is a good example to understand complexity and emergent phenomena. It is a problem that can arise when a new link in a network is added. For instance, for

each point of a road network is given the number of cars starting from it, and their destination. A driver decides to drive into a street based not only upon the quality of the road, but also upon the density of the flow. If every driver takes the path that looks most favorable to him, it comes out that, over a certain time period, a traffic jam will emerge. If another lane is added to the road, the traffic jam will emerge again. In other words, the traffic jam is only an emergence of the entire system (road and cars) due to individuals' actions and interactions.

When analyzing a specific problem, all the system features must be defined and all the variables and parameters must be set. For every system, there are different boundaries, inputs, entities, attributes, events, activities, interactions, processes, and outputs, due to the continuous and unexpected interactions of single agents. Simulation starts with a set of explicit assumptions and generates data that can be analyzed by reasoning. Unlike typical reasoning, the simulated data comes from a specified set of rules rather than direct measurement of the real world [38].

The first step in the simulation design is to choose a programming environment that grants portability to different operating systems (OS).

A good simulation model must be programmed to achieve

- specific goals,
- usability which helps to facilitate the comprehension of the interpretation of its output,
- extendibility which aims at facilitating the future use and application of the software to different problems and systems.

The second step is to analyze the results the simulation yields. A simulation can generate many different solutions every time it starts, due to the continuous interactions between agents and to the changes of the initial conditions (stochastic events). To understand if the outcome of a simulation is reliable, the simulation must be replicated many times, trying to use the same parameters and the same constraints as in the previous trials. Generally, some classical mathematical approaches, such as analysis of variance or regression, are used to establish if the effects of these changes are relevant for the simulation. The final results must be shared with an external audience, for instance by the publication of the results on scientific journals, as well as

presenting the results in some conference. This is an important stage: an external verification is particularly appropriate since simulation results are not analytically verified.

Swarm [39] is a software package for multi-agent simulation of complex systems, originally developed by a team of researchers in 1994 at the Santa Fe Institute. Swarm is intended to be a useful tool for researchers in a variety of disciplines. The basic architecture of Swarm is the simulation of collections of interacting agents that allow implementing different kind of systems and problems. Swarm is a set of libraries that facilitate implementation of agent-based models. One important feature of Swarm is the virtual machine. It allows to describe agent behaviors one by one, agent by agent, context by context, all while keeping an exact notion of time and concurrency in the world. Swarm also makes it possible to compose or decompose hierarchies of agents. This notion of composability is useful when modeling a large organization, indeed, instead of seeking denotation on how the organization should work and looking for deviations, one can build independent model components from many perspectives and then combine them (mirroring abstractions of people for real people). This bottom-up approach has the advantage of documenting the both unexpected bad and good things in the organization, as well as contextual sensitivities.

Brahms (Business Redesign Agent-based Holistic Modeling System) [40] is a set of software tools to develop and simulate multi-agent models of human and machine behavior. It was originally developed to analyze or design human organizations and work processes. Brahms is a multi-agent, rule-based, activity programming language. The Brahms language allows the representation of activities that happen in the context of a specific situation so that their execution is constrained not only by the capabilities of an agent, but also by the agent's beliefs of the external world, such as interactions and communication with other agents, state of the world, and so on. Its objective is to represent people's collaboration, interrupted and resumed activities, informal interactions and knowledge sharing that happen inside an organizational environment. Brahms agents are a representation of certain observed human behaviors that are codified by the modeler, and thus they can never be seen as possessing human intelligence or knowledge.

Orgmem [41] is a multi-agent simulation program that simulates the interpersonal

communication, information processing, and decision-making processes in organizations. Group representation is made by using the six relational matrixes and this allows to take into account not only important elements, such as resources and tasks, but also their interrelations. Agents in the model are intelligent, adaptive, and heterogeneous and have access to some knowledge and are able to conduct specific number of tasks and to learn from each other. Obviously, every agent has different characteristics, like people working in an organization, and it carries out different activities. Moreover, each of them also has a transactive memory about who talks to whom, who knows what, and who does what in the group. The simulation program assigns to every single agent a subtask and, due to single agent characteristics and interactions, they make decisions by combining personal knowledge and information and external environment stimulus. This Orgmem simulation program is designed to study how communication influences transactive memory and how the existence of transactive memory impacts group performance.

Agent-based simulations bear many problems. Outcomes produced by the simulations might be relevant at a theoretical level. Robustness of the simulated model is related to the situations where the values of the inputs are continuously changed. Another problem is related to the many software released and to the lack of a common standard and programming language. Moreover, due to the enormous amount of data that a simulation model produces, it is difficult to analyze and represent them, to share and compare them with others. Another problem is related to the computational power and to its cost. This is an important factor to be taken into account when a huge amount of interacting agents are present in a complex environment and when each agent is designed with its features and characteristics. In this sense, the duration and the speed of simulation are important factors that affect the simulation cost.

Agent-based simulation is an expanding research area because of its theoretical basis and the variety of issues that can be faced. Agent-based simulation can be used in many different applications such as theory building, virtual reality, complex and dynamics systems analyses, what-if analyses, training, entertainment and education. Moreover, supply chain management, macro-economics models, organization theory, distance learning, manufacturing systems are also growing research applications of this field. The goal of agent-based modeling is to enrich our understanding of fundamental processes that may appear in a variety of applications. When the

system involves many interactions between agents, simulation can help organizations gain a better insight on how they work.

2.7 Real System Using Agent Mickey

Text-Based information is the main concern due to its high significance in our daily lives. Therefore in my thesis, Agent Server gathers and intelligently combines information from multiple agents and presents the user with the combined information. Some form of user intervention is required to give direction to static agents. The active task of notifying the user periodically is the role of the static agent. A static agent wishing to receive information about a particular topic creates a registration query and sends it to the Agent Server. Task agent interrogates its local database and returns a list of static agents that match the query criteria instantly or over a period of time.

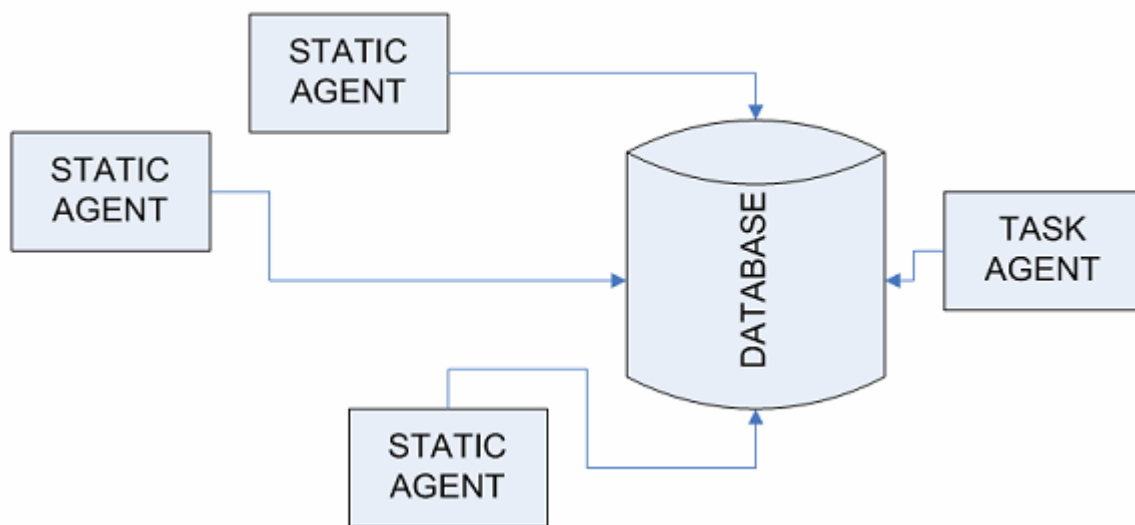


Figure 2.6 Information Integration Architecture

Since static agents do not have control over external information resources, they must store relevant pieces of information in a local database. We refer to this database as infobase [1]. The static agent's local infobase contains records that have been retrieved from external information sources in relation to one or more queries.

Upon startup every agent creates an internal goal to advertise itself by sending its infobase to the Agent Server. Static agent retrieves information from external static agents in response to one shot queries. Task agent fulfills the request of static agents for periodic information. Using C# Programming Language, characteristics of agents are built in a buyer-seller environment using Information Integration Architecture as seen in Figure 2.6.

Internet gives us new possibilities for reaching out to information. Today, it is possible to develop a highly functional web site for visitors. We browse around to find the right information. But it comes with a high cost. The amount of time we need to spend in front of a computer for specific information is very high. This raises a number of questions that are of interest to visitors and landlords on the internet. We cannot deny the fact that browsers are best ways of doing a research. Nonetheless, we can do away with internet browsers for specific purposes on the internet. The information gathering before making a purchase is crucial in choosing the right product at the right time at the right place at the right price. Static agents will help us out in deciding over such parameters.

Static (User) Agent Characteristics

- 1- Information Gathering
- 2- Reactive: Sensing and acting on the changes in the database.
- 3- Goal Oriented: Looks for specific data in the database.
- 4- Continuous: Works as long as user's machine is on.
- 5- Communicative: Tells seller what it does.
- 6- Approved: It is going to be in user's computer and will be on with user's approval.

Agent Server Characteristics

- 1- Database: holds the data received from static agents.
- 2- Task Agent reconstructs Agent Server database on a daily basis.

Advantages

- 1- Buyer will not spend time browsing internet for certain information. (price, product, location etc.)
- 2- There will be an alternative way for distributing product information.

Static agent will be demanding price, demanding feature, demanding location. At the same time, it can supply price, location, items and features. Static agents' interest will build a huge data collection on the agent server side which can be useful along the road for data mining.

CHAPTER 3

MATCHMAKING MECHANISMS

The study of multi-agent systems (MAS) focuses on systems in which many intelligent agents interact with each other [42]. In many situations, agents have to locate resources. This involves some kind of matchmaking. There are different approaches of matchmaking processes. Existing approaches are centralized matchmaking and localized matchmaking. These approaches can be compared in terms of matching time, task allocation efficiency, and communication cost.

3.1 Centralized Matchmaking

A centralized solution relies on one node being designated as the central node and it processes the entire application locally. All input data are sent to this node, and after local processing the computer sends the relevant output data to each of the other nodes as seen in Figure 3.1.

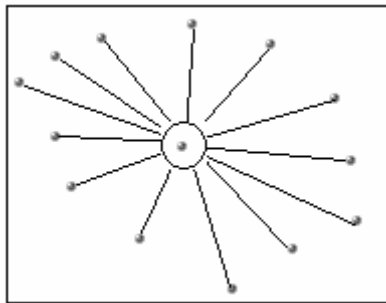


Figure 3.1 Centralized Agents

3.2 Localized Matchmaking

In localized solution, the processing steps of the application are divided among the participating nodes. The goal is to minimize communication and computation cost. The distributed model is characterized by a collection of autonomous processing elements, called nodes. In addition to some computing and storage resources, each node has the possibility to exchange information

with some of the other nodes. These are referred to as its neighbors and the communication place through a link. Whenever an agent needs resources to perform its assigned task, the agent needs to locate the available resources and make use of it as seen in Figure 3.2.

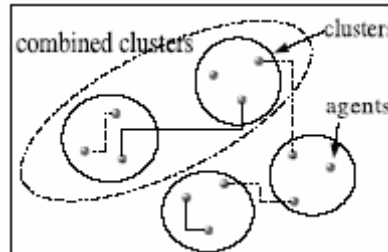


Figure 3.2 Localized Agents

3.3 Market Mechanism

In the field of matchmaking, many questions have been brought up. Most of these questions can be resolved by providing the agents with a monetary system, modeling them as buyers and sellers of tasks and resources. Each buyer and seller will set a price and quantity of how much it is willing to buy or sell. The market mechanism then computes the equilibrium price at which supply and demand converge and the market clears. In large-scale networks, certain nodes called buyers have to locate other nodes called sellers. This assumes some kind of matchmaking as a prerequisite to collaboration [43].

There are many different forms of such agents, differing in whether consumer or provider information is stored and the facilitator directly connects consumers and providers and acts as an intermediary for transactions.

Agents can be categorized into three groups: service providers, service requesters, and middle agents. Matchmaking is the process of finding an appropriate provider for a requester through a middle agent. Provider agents advertise their capabilities via middle agents who store these advertisements.

The requester asks the middle agent whether it knows of a provider with the desired capabilities. The middle agent matches the request against the advertisements and returns the result.

3.4 Comparison of Matchmaking Mechanisms

It is important to be able to determine the conditions under which particular mechanisms perform better than others.

Scalability requires that an increase in the number of new agents and resources noticeable effect on neither performance nor administrative tasks. Throughput is measured as the amount of useful work carried out by the system in a unit time. In MAS throughput is defined as the number of tasks that can be allocated to available resources in a period of time.

All facilitator architectures involve services or requests being stored in a central location. Agents wishing to find matches then submit a request to this location. Facilitator architectures are good for finding optimal matches since the facilitator is in a position to compare all available possibilities.

A centralized controller limits the scaling capability of the system. As the system grows, a central controller that stores data on all agents will need more and more memory. It will also need an increasing amount of processing time to search for matches and it will need more and more bandwidth to handle requests from the agents. Centralized matchmakings are very efficient in certain population sizes. Going beyond the boundary will increase the matching time but not the allocation efficiency.

One central middle-agent represents a single point of failure and communication bottleneck in the system. If the system loses the middle-agent, it is not possible to search for matches. Robustness can be designed into the system by allowing other nodes to assume the matchmaking function whenever required. [43]

In local matchmaking, agents can only interact within a local neighborhood. That is a small subset of other agents whose addresses are known to that agent. Any particular consumer can find the provider he is looking for within a local area. Even though such an approach guarantees scalability, it is not certain that it will result in a more efficient matchmaking. So if there is no task migration between the clusters, increasing the number of agents will increase the number of clusters but will not improve matching efficiency.

Localized matchmaking reduces communication overhead but the time required to find a match in the local neighborhood increases substantially. Nonetheless, localized mechanisms don't use one central middle-agent, so they avoid single point of failures.

To summarize, comparing different approaches shows that each mechanism has advantages and disadvantages. In terms of flexibility and robustness, localized matchmaking seems to have some advantages over a more centralized approach. However, as far as the matching efficiency is concerned localized matchmaking is less performing especially when there is little or no interaction between clusters. In addition, peer-to-peer mechanisms reduce communication overhead but they increase matching time. In terms of the number of messages that must be passed in the system in each trail, centralized matchmaking acts better but we should take into account that for large scale systems this cost increases linearly in centralized case.

CHAPTER 4

AGENT MICKEY

Agent Mickey helps us buy, sell items on the internet. User Agents reside on personal computers and task agent sits on the Agent Server. Entire data from individual users is stored on SQL Server 2000 database on the Agent Server.

4.1 C# Programming Language

The C# programming language is used to develop agent software in this thesis. Every 10 years or so, a new approach to programming hits like a tsunami [44]. In the early 1980s, the new technologies were Unix, which could be run on a desktop, and a powerful new language called C, developed by AT&T. The early 90's brought Windows and C++. Each of these developments represented a change in the way they approached programming. .NET and C# are the next wave. Object-oriented and Internet-centric programming language C# is built on C, C++, Java and Visual Basic to create a new language. The goal of C# is to provide a simple, modern, object-oriented, Internet-centric, high performance language for .NET development. C# is a new language, but it draws on the lessons learned over the past three decades. In much the way that you can see in young children the features and personalities of their parents and grandparents, you can easily see in C# the influence of Java, C++, Visual Basic (VB), and other languages.

4.2 Visual Studio IDE (Integrated Development Environment) and RAD

C# provides a Rapid Application Development (RAD) model similar to that previously available in Visual Basic. We can use this RAD model to create professional Windows programs using the Windows Forms development environment [44].

A Windows Form is a tool for building a Windows application. The .NET Framework offers

support for Windows application development, the centerpiece of which is the Windows Forms framework. This idea is borrowed from the Visual Basic (VB) environment and supports Rapid Application Development (RAD). Arguably, C# is the first development environment to marry the RAD tools of Visual Basic with the object-oriented and high-performance characteristics of a C family language. Visual Studio.NET provides a rich set of drag-and-drop tools for working with Windows Forms.

4.3 SQL (Structured Query Language)

The history of SQL begins in an IBM laboratory in San Jose, California, where SQL was developed in the late 1970s. The initials stand for Structured Query Language, and the language itself is often referred to as 'sequel'. It was originally developed for IBM's DB2 product (A relational database management system that can still be bought today for various platforms and environments.)[45]. In fact, SQL makes a relational database management system possible. SQL is a nonprocedural language, in contrast to the procedural or third generation languages (3GLs) such as COBOL and C that had been created up to that time. Nonprocedural means what rather than how. SQL describes what data to retrieve, delete, or insert, rather than how to perform the operation.

Two standards organizations, the American National Standards Institute (ANSI) and the International Standards Organization (ISO), currently promote SQL standards to industry. Although these standard-making bodies prepare standards for database system designers to follow, all database products differ from the ANSI standard to some degree. SQL is relatively easy to learn, standardized language which can be used to define and manipulate data in a database. T-SQL (Transact-SQL) is used in this thesis.

4.4 SQL Server 2000

The demand for database driven sites on the Internet is exploding (some major sites are running SQL Server 2000 as their back end). Microsoft SQL Server is a client/server database management system. A client / server database management system consists of two components: a front-end component (the client), which is used to present and manipulate data; and a backend component (database server), which is used to store, retrieve, and protect the databases. In a client/server system, the majority of the data processing is done on the server instead of the clients. This means that a client/server system can reduce our network traffic. In addition, client/server systems are easier to scale because we can upgrade their performance simply by upgrading the server's hardware [46].

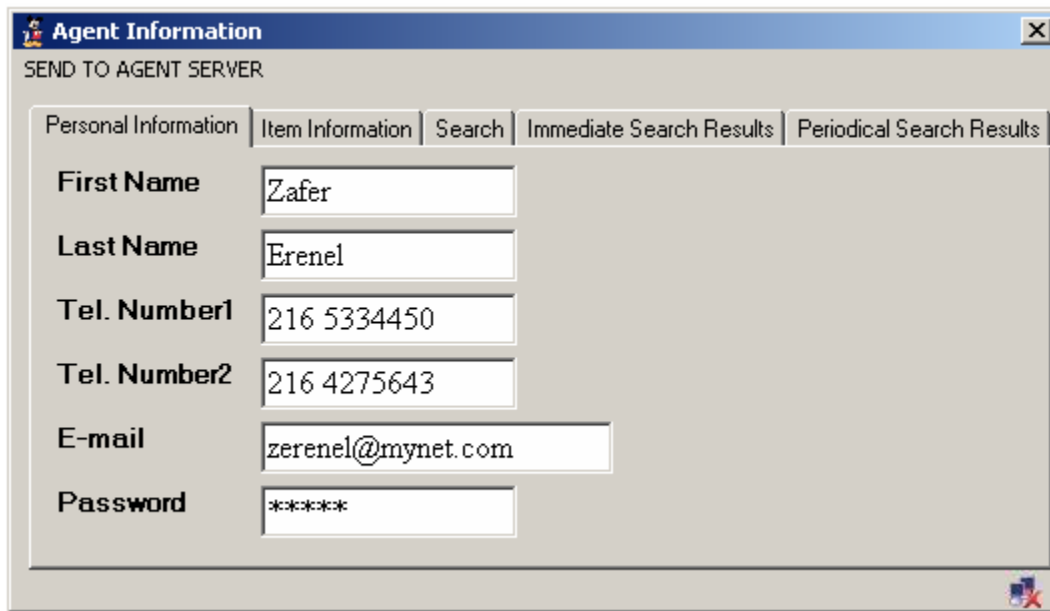
SQL Server easily supports terabyte-size databases. SQL Server databases typically take one of two forms:

- An Online Transaction Processing (OLTP) system, in which users continually make changes to the data in the database. For example, the database system for recording customers' orders at Amazon.com is an OLTP system.
- An Online Analytical Processing (OLAP) system, in which we primarily focus on analyzing the data in the database. We typically don't make many changes to such databases.

Microsoft SQL Server 2000 is a relational database management system (RDBMS). An RDBMS uses established relationships between the data in a database to ensure the integrity of the data. For example, if we're setting up an order-entry database system, we'll probably define a relationship between the customer and invoice tables so that a sales clerk can't enter a customer account number in the invoice table if that customer doesn't exist in the customer table. These relationships enable us to prevent users from entering incorrect data.

4.5 User Agent

Task agent makes periodic research and report results to the user agents. User agents store individuals' contact (first name, last name, telephone numbers, electronic mail addresses) and item information and send these to Agent Server when connected. Personal Information of the user is seen in Figure 4.1.



The image shows a Windows-style dialog box titled "Agent Information". Below the title bar, it says "SEND TO AGENT SERVER". There are five tabs: "Personal Information", "Item Information", "Search", "Immediate Search Results", and "Periodical Search Results". The "Personal Information" tab is selected. It contains the following fields:

| | |
|---------------------|-------------------|
| First Name | Zafer |
| Last Name | Erenel |
| Tel. Number1 | 216 5334450 |
| Tel. Number2 | 216 4275643 |
| E-mail | zerenel@mynet.com |
| Password | ***** |

Figure 4.1 Agent Mickey Personal Information

There is another thread in the User Agent Program that checks internet connection and updates users' info on the Agent Server.

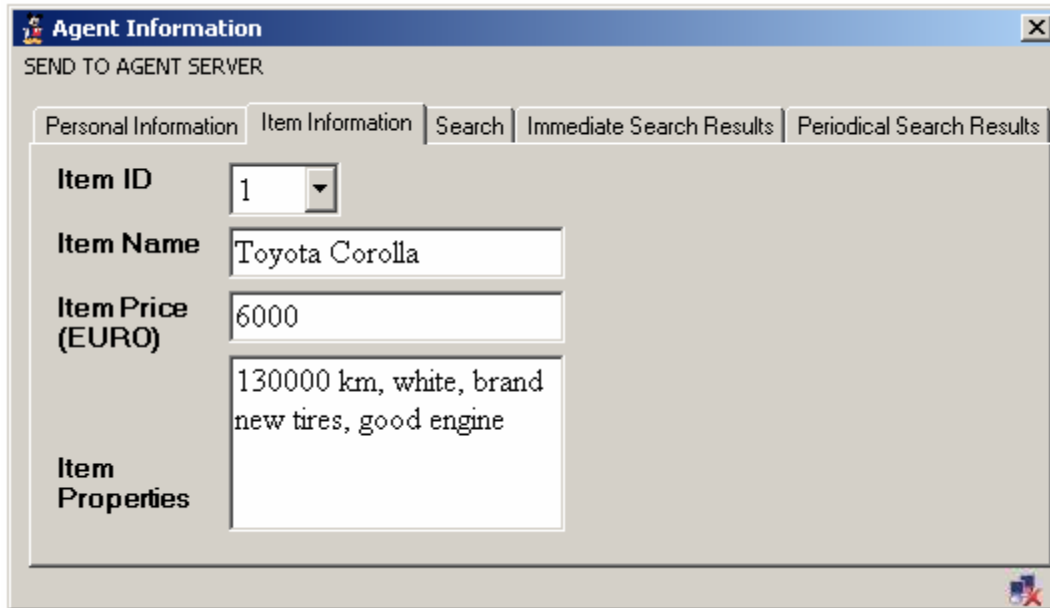


Figure 4.2 Agent Mickey Item Information

We can store up to 5 items in our user agent. In Figure 4.2, we see that user has put his car on sale. This is the first item user is offering. The price of the item one is selling is important therefore it is right under the name of the item. The properties of the item are valued information when one is trying to sell it.



Figure 4.3 Agent Mickey on Task Bar

Agent Mickey runs on start-up and appears on the task bar on far left as seen in Figure 4.3. We can turn it off anytime.

The screenshot shows a window titled "Agent Information" with a subtitle "SEND TO AGENT SERVER". It has five tabs: "Personal Information", "Item Information", "Search", "Immediate Search Results", and "Periodical Search Results". The "Search" tab is active. On the left, there are three input fields: "Item Name" containing "toyota", "Item Properties" (empty), and "Price Range (EURO)" (two empty dropdown menus). On the right, there are two sections: "Immediate Searches" with a checked radio button for "Right Away", and "Periodical Searches" with radio buttons for "1 week", "2 weeks", and "1 month". A "Search" button is at the bottom right.

Figure 4.4 Agent Mickey Search

We can do immediate and periodical searches in the Search section as seen in Figure 4.4. Periodical Search is the one where the Task Agent kicks in. Task Agent stored on Agent Server does a periodical search on behalf of the user agent. Task Agent sends queries to the SQL Server on behalf of the user agent for a period of time. Periodical results can be watched by getting on line as seen in Figure 4.5 and Figure 4.6. In Figure 4.5, no results have been found so search is unsuccessful for task 1. Task 1 is basically a query sent to agent server for periodic searches. Task Agent takes that query and runs it against the database for a period of time on behalf of the user agent. Searches in Figure 4.5 and 4.6 are minutely searches. Users who are interested in buying a Toyota at stated price can note down the e-mail address or the phone number of the owner by looking at the search results.

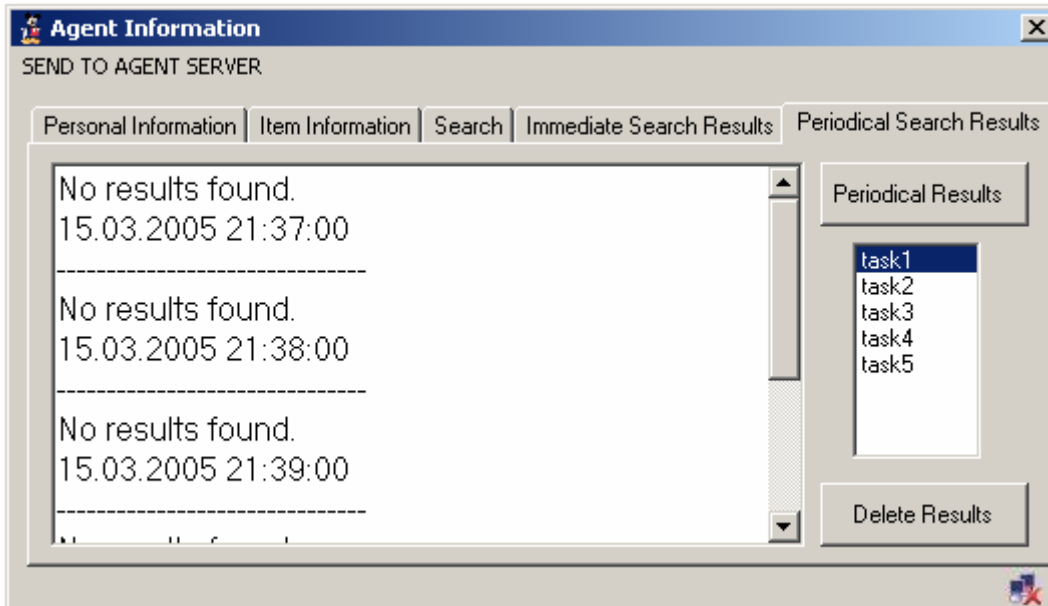


Figure 4.5 Agent Mickey Periodical Search Results (No results found)

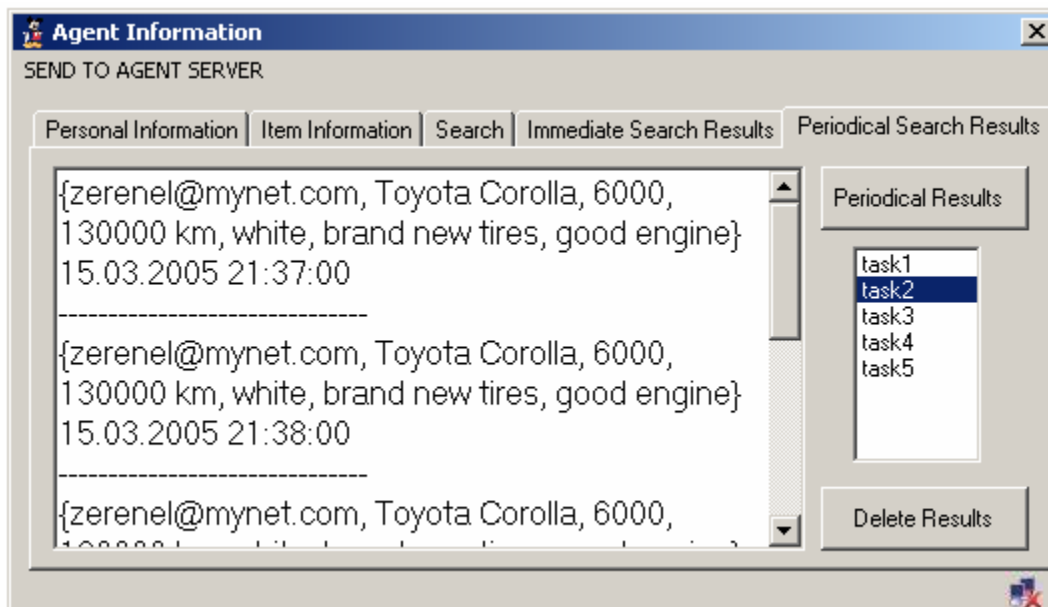


Figure 4.6 Agent Mickey Periodical Search Results (Results found)

The screenshot shows a window titled 'Agent Information' with a sub-header 'SEND TO AGENT SERVER'. It has five tabs: 'Personal Information', 'Item Information', 'Search', 'Immediate Search Results', and 'Periodical Search Results'. The 'Immediate Search Results' tab is active, displaying a table with the following data:

| | E-Mail | Item | Price (EURO) | Properties |
|---|-------------------|----------------|--------------|----------------------|
| ▶ | zerehel@mynet.com | Toyota Corolla | 6000 | 130000 km, white, br |
| * | | | | |

Below the table is a large yellow rectangular area, likely a placeholder for more results or a scrollable list.

Figure 4.7 Agent Mickey Immediate Search Results

User Agent sends SQL Query directly to the SQL Server 2000 database in immediate search if connected to the Internet. Task Agent is not involved. User Agent sends the query to the Agent Server and results are immediately available as seen in Figure 4.7.

4.6 Task Agent

We can think of user agent as a

- 1- Shop window where we display our items.
- 2- Item finder.(Both periodic and immediate)

We can think of task agent as a Query (SQL) Runner and Database constructor.

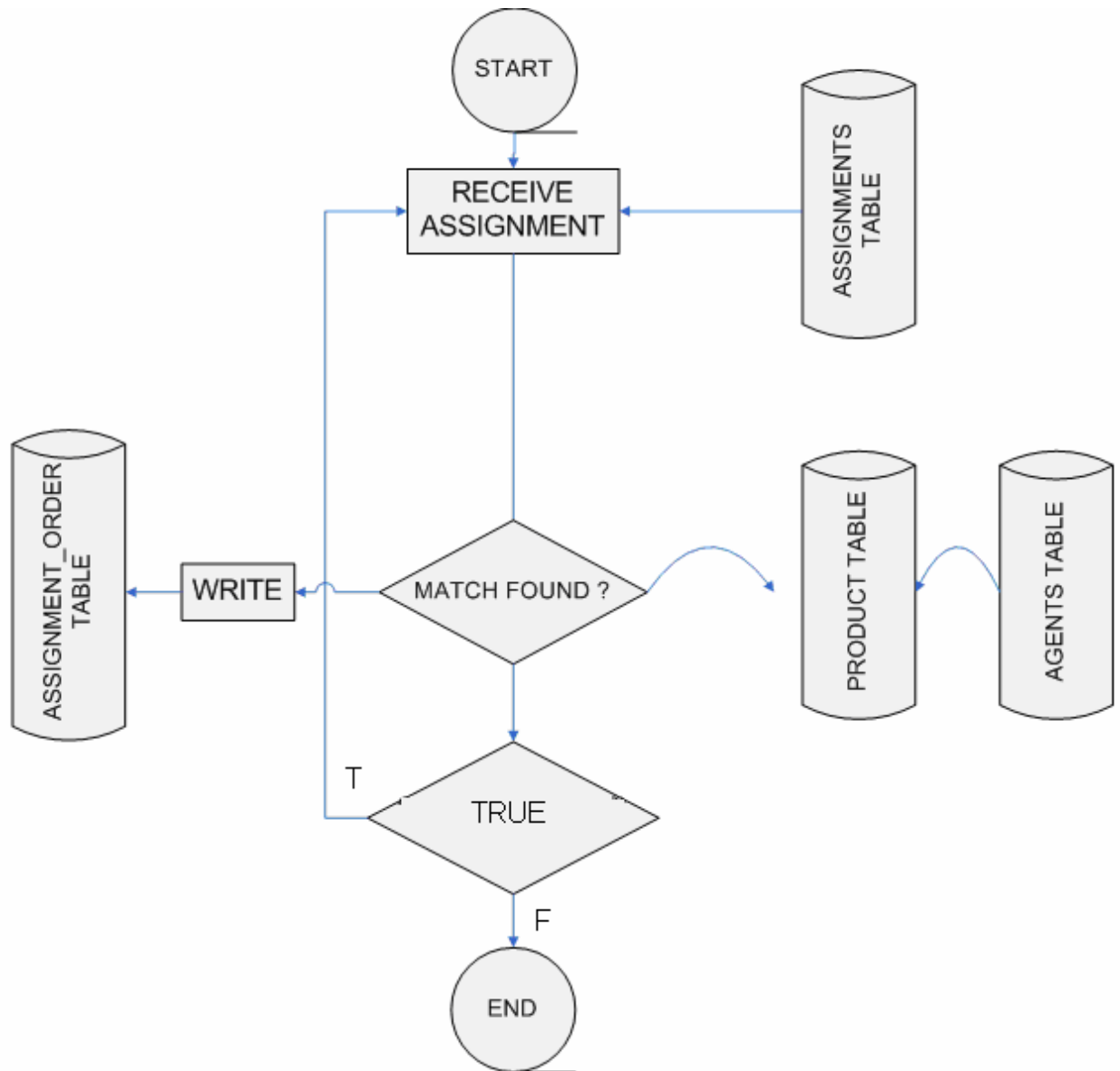


Figure 4.8 Task Agent Flow Chart

In Figure 4.8, task agent's daily tasks are defined. There are four tables that are being used on the Agent Server.

- 1- Assignments (TSQL1 : Item Name, TSQL2: Item Properties, TSQL3: Price Lower Limit, TSQL4: Price Upper Limit, DAYS_LEFT: The Number of Days Left for Periodical Search, DAYS: The Total Number of Days)
- 2- Agents (Stores personal information of the users)
- 3- Product (Stores items and their specifications.)

- 4- Assignment_Order (Stores results that are returned when queries are sent by Task Agent on users' behalf.)

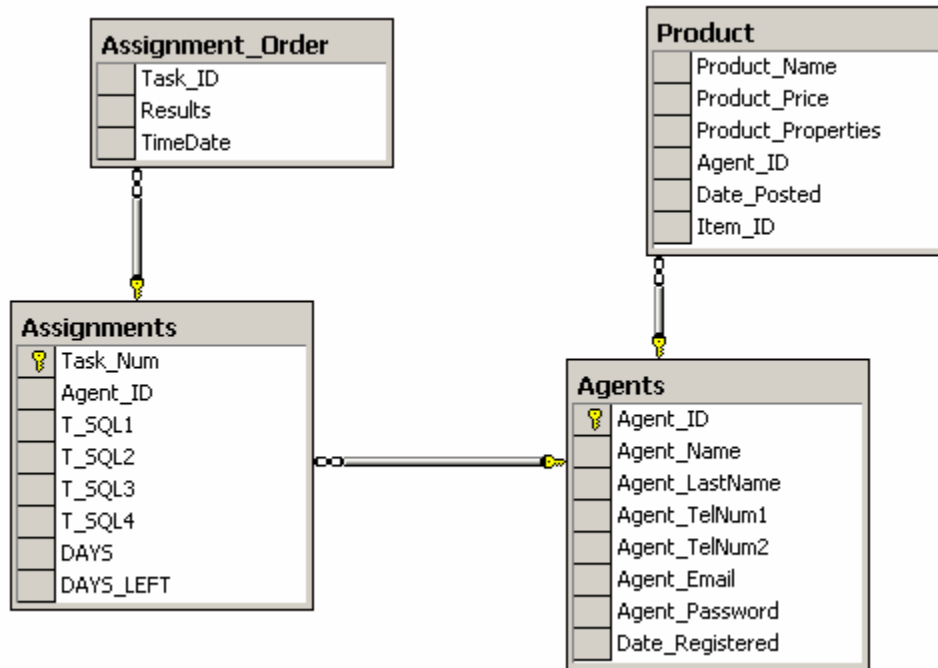


Figure 4.9 Agent Server Database Tables

In Figure 4.9, relational tables are shown. Agents table holds data of users. Products table holds data of items on sale. When Agent Server receives a new task, new task is held in Assignments table. Assignments are processed by Task Agent and results are held on Assignment_Order table.

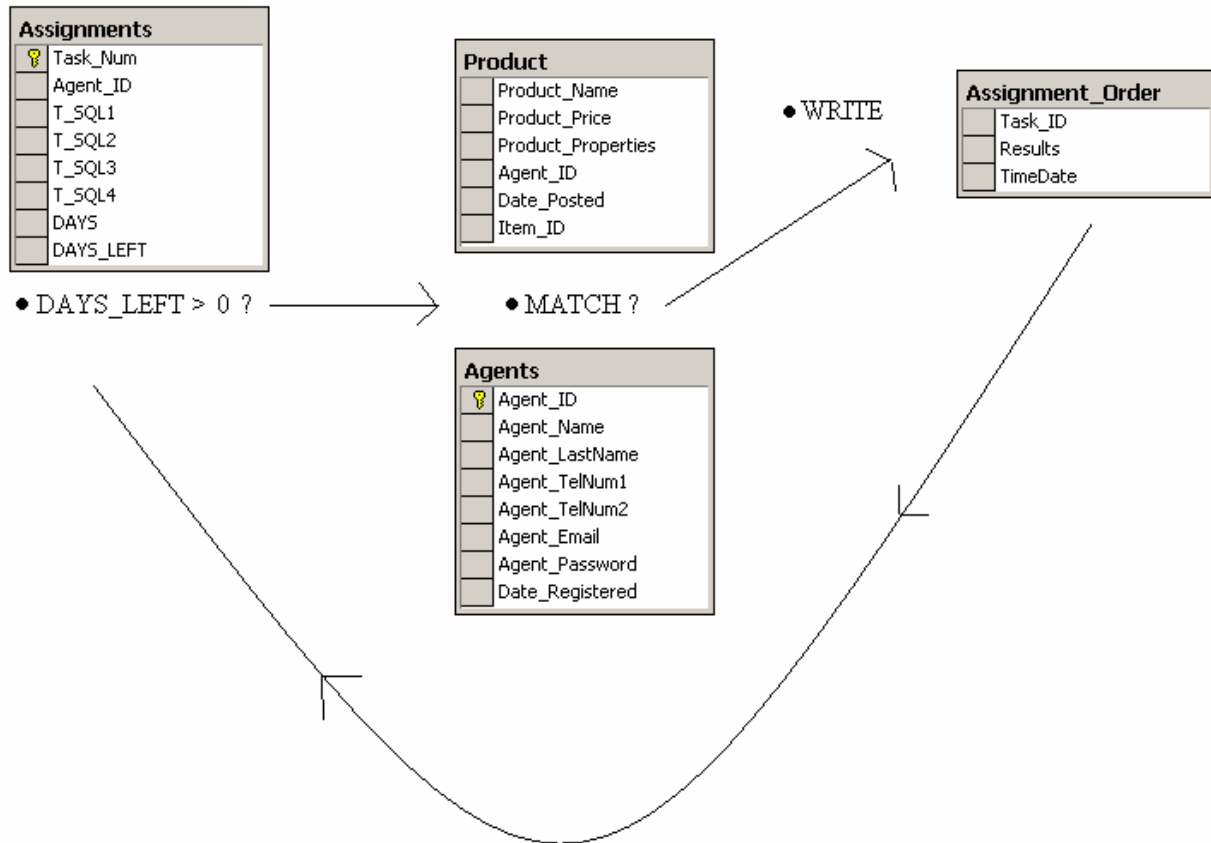


Figure 4.10 Database Work Flow

As seen in Figure 4.10, Task Agent goes over the Assignments table to find out what agents are inquiring periodic results. If there is a match in the Product table for the records whose DAYS_LEFT columns are greater than 1 in Assignments table, match results (agent information) are written to Assignment_Order table. Finally, the value in the DAYS_LEFT column is reduced by one for the given task in the Assignments table. The user agent gets on line to retrieve the results from Assignment_Order table.

Table 4.1 SQL Queries for the Work Done

| SQL QUERY | WORK DONE |
|--|---|
| SELECT * FROM Assignments WHERE DAYS_LEFT > 0 | GET AN ASSIGNMENT FROM Assignments TABLE. |
| SELECT Agent_Email, Product_Name, Product_Price, Product_Properties FROM Product, Agents WHERE Agents.Agent_ID = Product.Agent_ID AND Product_Name LIKE '%Toyota%' AND Product_Properties LIKE '%130000 km%' AND Product_Price BETWEEN 5000 and 7000 | FIND A MATCH FROM Product AND Agents TABLES. |
| INSERT INTO Assignment_Order(Task_ID,Results,TimeDate) VALUES(@ID,@RES,GETDATE()) | WRITE THE MATCHED AGENTS TO Assignment_Order TABLE. |
| UPDATE Assignments SET DAYS_LEFT = DAYS_LEFT-1 WHERE Task_Num = @A | UPDATE THE Days_Left COLUMN FOR THE GIVEN ASSIGNMENT. |

In Table 3.1, we can see the SQL queries for the work done on task agent's part.

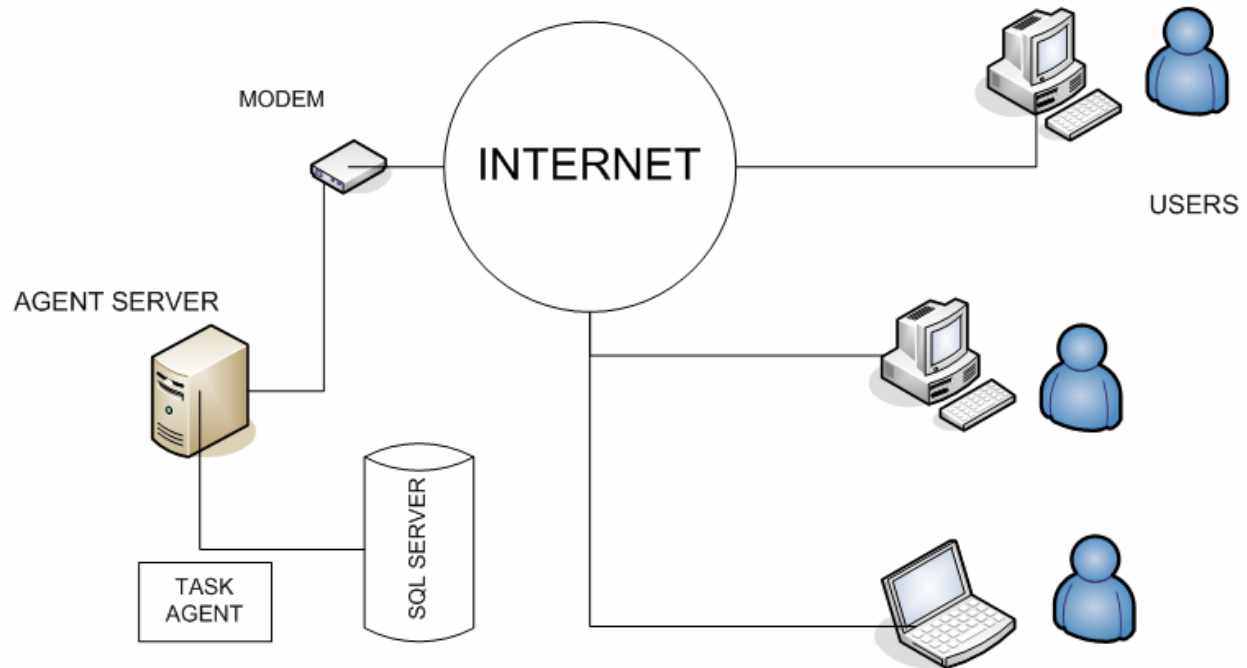


Figure 4.11 Static Agent Architecture Map

Figure 4.11 show us that user agents may be increased in number while there is only one task agent located on Agent Server. Performance Analysis of the task agent is made by increasing user agents in the following chapter.

CHAPTER 5

PERFORMANCE ANALYSIS

In this chapter, we are going to find the performance limits of the task agent. Task Agent is running queries against the database for periodical searches.

5.1 Performance Factors

There are 6 parameters to consider when we are talking about Task Agent's performance limits.

- 1- The number of tasks.
- 2- The number of advertised products.
- 3- The number of agents.
- 3- Database Management System. (SQL Server 2000)
- 4- The programming language and algorithm used for creating Task Agent. (C#)
- 5- The computer hardware and software used for harboring Agent Server Environment.

The software and hardware used for the following tests are given below.

Operating System Used: Microsoft Windows XP Professional Version 2002 Service Pack 1.

Hardware Used: Intel Celeron 2.10 GHz, 480 MB of RAM.

5.2 Task Agent Real-Time Performance

Table 5.1 Task Agent Performance for 10 Agents

| 10 AGENTS 50 PRODUCTS TASKS | seconds |
|-----------------------------|----------|
| 10 | 0.515625 |
| 20 | 0.640625 |
| 30 | 0.71875 |
| 40 | 0.859375 |
| 50 | 0.84375 |
| 60 | 0.953125 |
| 70 | 1.015625 |
| 80 | 1.15625 |
| 90 | 1.171875 |
| 100 | 1.21875 |

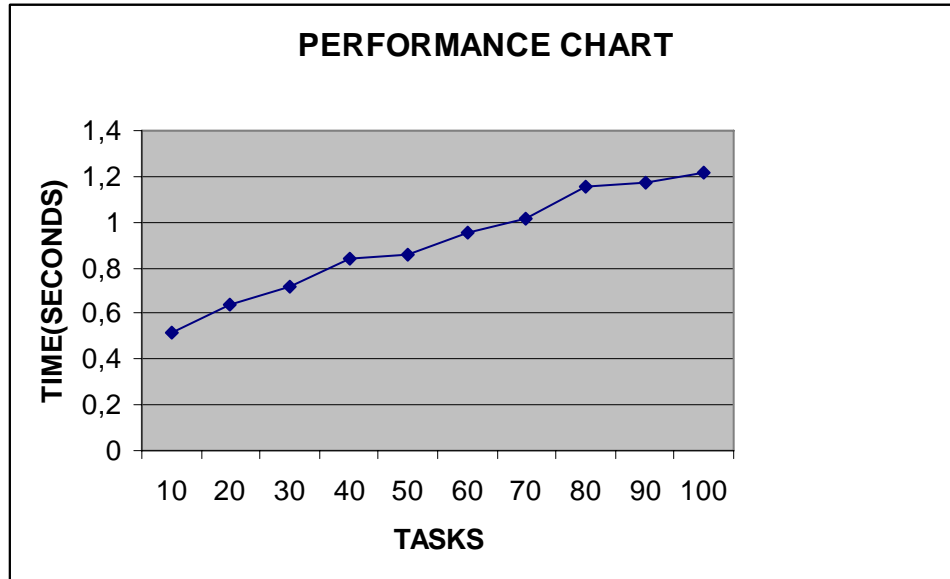


Figure 5.1 Performance Chart of a Task Agent for 10 Agents

I have created 10 agents and added 50 products. (It is assumed that each agent has put up 5 products on sale.) As I increased the number of tasks on the shoulders of the task agent, I have seen a considerable change in the amount of time it requires to finish given tasks. Table 5.1 and Figure 5.1 shows the amount of time task agent spends between 10 and 100 tasks.

Average Speed = $550 / 9.09375 = 60.48109$ tasks/second.

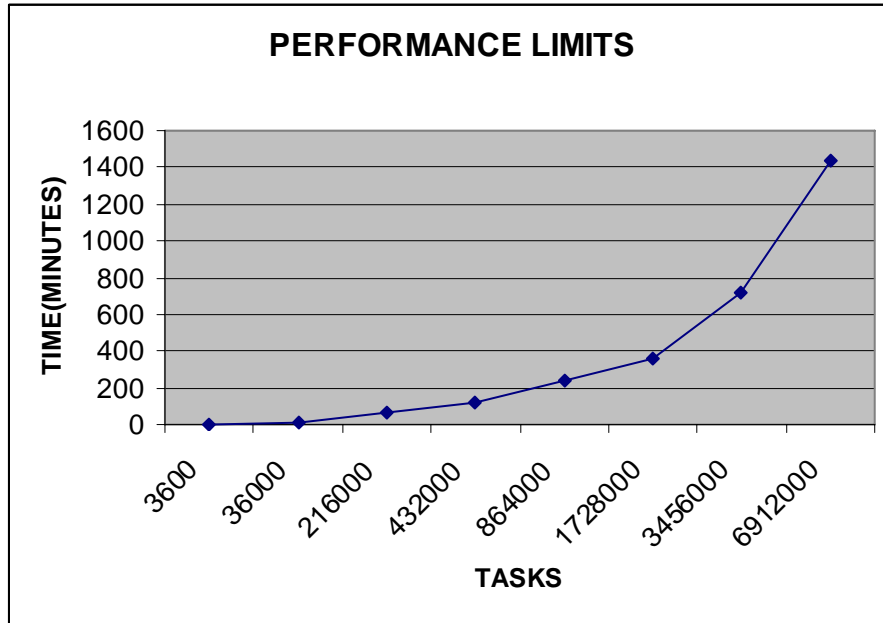


Figure 5.2 Performance Limits of a Task Agent for 10 Agents

Figure 5.2 tell us that for daily updates, our agent can handle up to 6912000 tasks. Needless to say, periodical task does not end in one day. New tasks added to old tasks will dramatically increase the workload of the task agent.

Table 5.2 Task Agent Performance Limits for Daily Updates (10 Agents)

| TASK AGENT | 1 week | 2 weeks | 4 weeks |
|----------------------------|---------------------|---------------------|---------------------|
| Daily Task Capacity | 987428 tasks | 493714 tasks | 246857 tasks |

Frequency is the number of complete scans of task agent in a given table per a given amount of time. $\text{Frequency} = 1 / (24 * 60 * 60) \text{ seconds} = 1 / 86400 = 0.00001157407 \text{ scans per second}$ if the task agent is working once a day.

Task agent can complete 987428 daily tasks for weekly updates, 493714 daily tasks for 2-week updates and 246857 daily tasks for 4-week updates if there are 10 agents as seen in Table 5.2.

I have increased the number of items to 5000 by adding another 990 agents.

Table 5.3 Task Agent Performance for 1000 Agents

| 1000 AGENTS 5000 PRODUCTS TASKS | seconds |
|---------------------------------|----------|
| 10 | 2.046875 |
| 20 | 2.4375 |
| 30 | 2.828125 |
| 40 | 3.171875 |
| 50 | 4.515625 |
| 60 | 4.671875 |
| 70 | 4.8125 |
| 80 | 5.21875 |
| 90 | 5.421875 |
| 100 | 6.9375 |

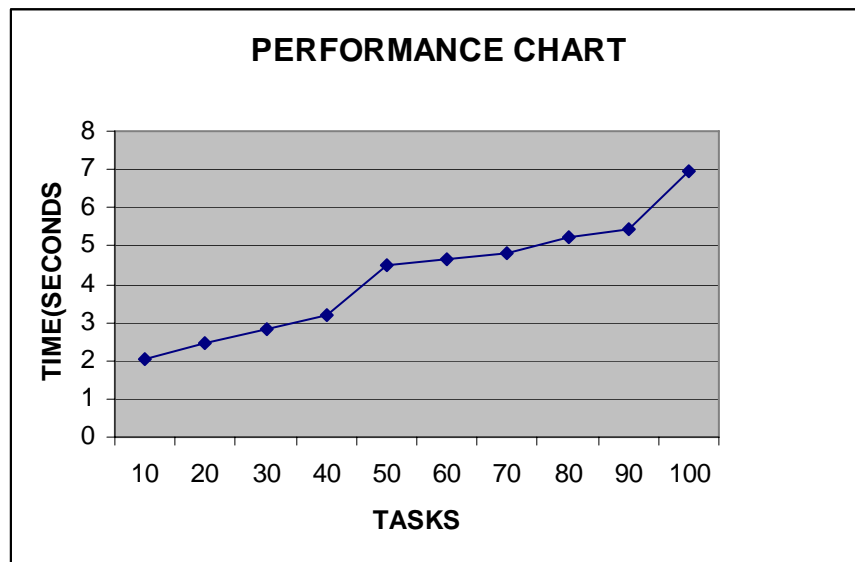


Figure 5.3 Performance Chart of a Task Agent for 1000 Agents

Average Speed = 13,0757 tasks/second. We found the average speed by dividing sum of tasks to sum of seconds in Table 5.3. This is graphed in Figure 5.3.

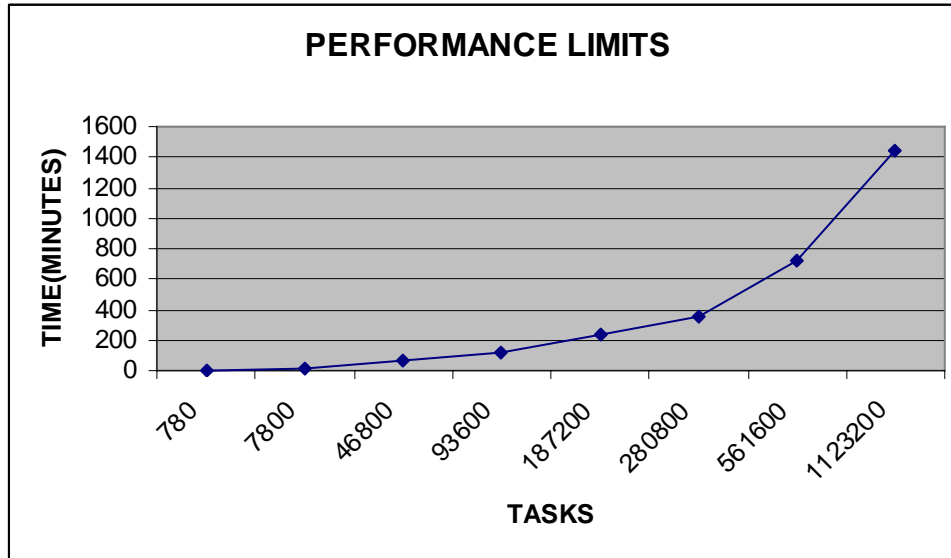


Figure 5.4 Performance Limits of a Task Agent for 1000 Agents

Task Agent Performance Limits for 1000 Agents are given in Figure 5.4.

Table 5.4 Task Agent Performance Limits for Daily Updates (1000 Agents)

| TASK AGENT | 1 week | 2 weeks | 4 weeks |
|---------------------|--------------|-------------|-------------|
| Daily Task Capacity | 161391 tasks | 80695 tasks | 40347 tasks |

Task agent can complete 161391 daily tasks for weekly updates, 80695 daily tasks for 2-week updates and 40347 daily tasks for 4-week updates if there are 1000 agents and 5000 items as seen in Table 5.4.

Table 5.5 Performance of Task Agent (10 products versus 5000 products)

| Products | 50 Items | 5000 Items |
|------------------------|---------------------|---------------------|
| Task Agent Performance | 60.481 tasks/second | 13.075 tasks/second |

We can see that if we multiply the number of items by **100**, (which is multiplying the number of static agents by 100) the performance of the task agent is divided to **4.625** in Table 5.12. There is a reverse ratio between the two. Therefore, we can measure the approximate performance limits of a task agent in any server with different hardware by measuring the task performance for any number of products using our findings. This leads us to Table 5.6 and Figure 5.5 where we can see the limits of a task agent for our computer system.

Table 5.6 Performance of Task Agent for a range of agents from 1 to 1,000,000,000

| STATIC AGENTS | ITEMS | Task Agent Performance(tasks/second) |
|---------------|------------|--------------------------------------|
| 1 | 5 | 130.043 |
| 10 | 50 | 60 |
| 100 | 500 | 28.116 |
| 1000 | 5000 | 13.075 |
| 10000 | 50000 | 6.079 |
| 100000 | 500000 | 2.827 |
| 1000000 | 5000000 | 1.314 |
| 10000000 | 50000000 | 0.612 |
| 100000000 | 500000000 | 0.284 |
| 1000000000 | 5000000000 | 0.132 |

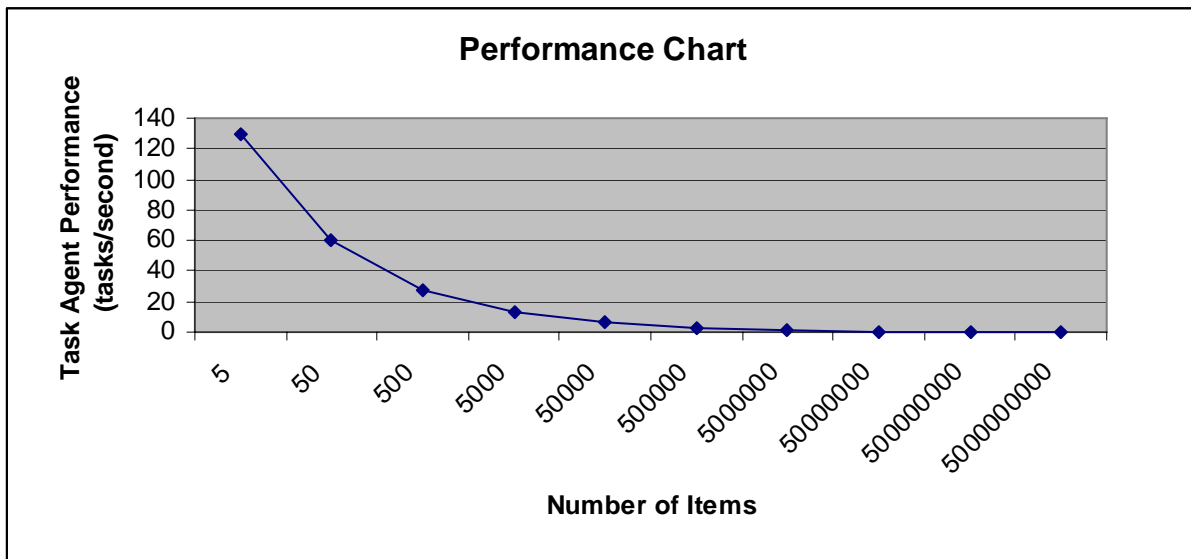


Figure 5.5 Performance Chart of a Task Agent for a range of agents from 1 to 1,000,000,000

Table 5.6 gives us a good idea about the task agent's performance. If we are more interested in the intermediate points, we can define a formula that might be applied to any number of agents. A logarithm may be defined with any base. The logs most often encountered are the logarithm to the base 10 which is called the common logarithm. We

$$y = \log_{10} x ; x = 10^y$$

What if we have the following information and we want to find the performance limits for 5000 agents.

| | |
|--------------------------------------|--------|
| Static Agents: | 1000 |
| Items: | 5000 |
| Task Agent Performance(tasks/second) | 13.075 |

If we multiply the agents by 100, the performance is divided by 4.625. Therefore if we multiply the number of agents by 5, we need to divide the performance by 1.706.

$\log_{100} 5 = 0.349$; $4.625^{0.349} = 1.706$. Thus for 5000 agents, task agent performance is 7.664 tasks/second.

$$\frac{\log_{100} X}{4.625} = Y$$

X: agent number ratio

Y: performance ratio

If we multiply the static agent number with X, we need to divide the performance by Y and vice versa. Nevertheless we need to find the error rate in our approximation. Therefore I have tested the real architecture for 5000 Agents and 25000 products. The results are below.

Table 5.7 Task Agent Performance for 5000 Agents

| 5000 AGENTS 25000 PRODUCTS TASKS | seconds |
|----------------------------------|---------|
| 10 | 3.203 |
| 20 | 5.031 |
| 30 | 5.890 |
| 40 | 6.921 |
| 50 | 8.093 |
| 60 | 8.890 |
| 70 | 9.984 |
| 80 | 11.031 |
| 90 | 12.250 |
| 100 | 13.078 |

Average Speed = 6.518 tasks/second. We found the average speed by dividing sum of tasks to sum of seconds in Table 5.7.

Approximation Error Rate = $1 - 6.518/7.664 = 0.15$.

If numbers are analyzed in Table 5.6 for the information integration architecture in this thesis, task agent can keep up with assigned tasks if there are assignment limits on each static agent. Hence, users will be prevented from assigning tasks over predetermined numbers. Otherwise, task agent will delay certain number of tasks each day. These delays will give rise to other delays which will lead to the eventual collapse of the system.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Software agents are broken up into 3 categories; 1-Task Specific Agents, 2- Entertainment Agents, 3- Viruses [22].

Software agents are helping buyers and sellers combat information overload and speed up specific stages of the online buying process. Today's first-generation agent-mediated e-commerce systems are already creating new markets and beginning to reduce transaction costs in a variety of business processes. However, we still have a way to go before software agents transform the way companies conduct business. This change will occur as agent technologies grow to better manage personalized preferences, complex goals, changing environments, and disconnected parties. The greatest changes may occur once standards are adopted to define goods and services [42], consumer and merchant profiles, and secure payment mechanisms.

In this thesis, several agent architectures are introduced and an agent environment is created using task specific static agents. Static agents reflect the behaviors of internet users on buying and selling items. Their items and queries are stored on Agent Server. Agent Server hosts task agent and a database of items' data. Task agent's real-time performance results are obtained by creating a sample database on the Agent Server from several static agents located at other nodes. We have run tests on task agent by adding 50 and 5000 items on the database. We have created an approximate ratio that can be applied to SQL Server 2000 database and C# programming language environments and found the error rate of the ratio.

Static agents are just one part of larger multi-agent systems that may include independent task agents and interface agents responsive to the goals of a human user. We presented a detailed, currently implemented architecture for these agents.

Task agent will accept properly stated goals from other agents and will work toward their achievement. Such architecture can support a range of sophisticated agent behaviors.

Using this architecture as a base, we can describe a set of behaviors for user agents. Our initial basic set of user agent behaviors includes agent item advertising, answering one-shot queries, reporting the results of a repeated query, and monitoring an information source for changes.

6.2 Future Work

This thesis has created a three dimensional model in which tasks, items and agents play an important role on the entire system and particularly the task agent.

There are 6 parameters to consider when we are talking about task agent's performance limits as I mentioned in chapter 5. I have measured performance by increasing the number of items and agents (maximum 5 items belong to one agent.) that task agent needs to scour. The following can be done regarding static agent architecture;

- The number of advertised items might be increased for further tests.
- The number of static agents might be increased for further tests.
- The number of tasks might be increased for further tests.
- Another Database Management System can replace SQL Server 2000 for harboring agents, items and assignments data.
- Another programming Language might replace 'C#' for creating task agent.
- Another algorithm might be developed for creating task agent.
- The affects of the match rate on the task agent's performance can be studied.
- The number of task agents on Agent Server might be increased for further tests.

REFERENCES

- [1] Decker K., Pannu A., Sycara K. and Williamson M., “Designing Behaviors for Information Agents”,1997.
- [2] Nwana, H. S. 1996. Software Agents: An Overview. Knowledge Engineering Review, 11(3): 205-244.
- [3] Ying-Hong W., Ching-Ling W., Hui-Hua H. and Wen-Hung L. , “Applying Mobile Agents to E-business,” Tamkang Journal of Science and Engineering, Vol. 6, No. 3, pp. 159-172 (2003)
- [4] Norman, D. A. 1997. How Might People Interact with Agents? In Software Agents, ed J. M. Bradshaw. Menlo Park, Calif.: AAI Press.
- [5] Negroponte, N. 1997. Agents: From Direct Manipulation to Delegation. In Software Agents, ed. J. M. Bradshaw. Menlo Park, Calif.: AAI Press.
- [6] Kay, A. 1984. Computer Software. Scientific American 251(3): 53–59.
- [7] Sharp, M. 1993. Reactive Agents, Technical Report, Apple Computer, Cupertino, Calif.
- [8] Knoblock, C. A., & Ambite, J.-L. 1996. Agents for Information Gathering. In Software Agents, ed. J. M. Bradshaw. Menlo Park, Calif.: AAI Press.
- [9] Minsky, M. 1986. The Society of Mind. New York: Simon & Schuster.
- [10] Minsky, M., and Riecken, D. 1994. A Conversation with Marvin Minsky about Agents. Communications of the ACM 37(7): 23–29.
- [11] Moulin, B., and Chaib-draa, B. 1996. An Overview of Distributed Artificial Intelligence. In Foundations of Distributed Artificial Intelligence, eds. G. M. P. O’Hare and N. R. Jennings,

3–55. New York: Wiley.

[12] Coutaz, J. 1990. *Interfaces Homme Ordinateur: Conception et Réalisation*. Paris: Editions Bordas.

[13] Wiederhold, G. 1992. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, March, 38–49.

[14] Wiederhold, G. 1989. *The Architecture of Future Information Systems*, Technical Report, Computer Science Department, Stanford University.

[15] Boy, G. A. 1991. *Intelligent Assistant Systems*. San Diego, Calif.: Academic Press.

[16] Maes, P. 1997. Agents that Reduce Work and Information Overload. In *Software Agents*, ed. J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.

[17] General Magic. 1996. *Tabriz White Paper: Transforming Passive Networks into an Active, Persistent, and Secure Business Advantage*, White Paper (<http://www.genmagic.com/Tabriz/Whitepapers/tabrizwp.html>), General Magic, Mountain View, California.

[18] Genesereth, M. R. 1997. An Agent-based Framework for Interoperability. In *Software Agents*, ed. J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.

[19] Shoham, Y. 1997. An Overview of Agent-oriented Programming. In *Software Agents*, ed. J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.

[20] Negroponte, N. 1997. Agents: From Direct Manipulation to Delegation. In *Software Agents*, ed. J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.

[21] Etzioni, O., & Weld, D. S. 1995. Intelligent agents on the Internet: Fact, fiction, and forecast. *IEEE Expert*, 10(4), 44-49.

[22] Franklin, S., and Graesser, A. 1996. Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. New York: Springer-Verlag.

[23] Newell, A. 1982. The Knowledge Level. *Artificial Intelligence* 18:87–127.

[24] Gilbert, D.; Aparicio, M.; Atkinson, B.; Brady, S.; Ciccarino, J.; Grosz, B.; O'Connor, P.; Osisek, D.; Pritko, S.; Spagna, R.; and Wilson, L. 1995. *IBM Intelligent Agent Strategy*, IBM Corporation.

[25] Petrie, C. J. 1996. Agent-Based Engineering, the Web, and Intelligence. *IEEE Expert*, 11(6): 24-29.

[26] Brodie, M. L. 1989. Future Intelligent Information Systems: AI and Database Technologies Working Together. In *Readings in Artificial Intelligence and Databases*, eds. J. Mylopoulos and M. L. Brodie, 623–642. San Francisco, Calif.: Morgan Kaufmann.

[27] S. and P. Chalasani Jha, O. Shehory, and K. Sycara. A formal treatment of distributed matchmaking. In *Proc. of International Conference on Autonomous Agents*, pages 457–458, May 1998.

[28] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical Report, IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, March 1995.

[29] Walter Binder and Volker Roth. Secure mobile agent systems using Java – where are we heading? In *Proc. 17th ACM Symposium on Applied Computing, Special Track on Agents, Interactions, Mobility, and Systems SAC/AIMS*, Madrid, Spain, March 2002. ACM.

- [30] Sakaguchi, M., Sugiura, A. and Kamba, T., "A Shopping Assistant Agent for Webshops," C&C Media Research Laboratories, NEC Corporation.
- [31] Lesser, V., Horling, B., Raja, A., Zhang, X. and Wagner, T., "Resource-Bounded Searches in an Information Marketplace," IEEE Internet Computing, March/April, pp. 49-58 (2000).
- [32] Esmahi, L., Dini, P. and Bernard, J. C., "Toward an Open Virtual Market Place for Mobile Agents, Enabling Technologies: Infrastructure for Collaborative Enterprises," (WET ICE '99) Proceedings IEEE 8th International Workshops, pp. 279-286 (1999).
- [33] Vermeulen, C. and Bauwens, B. "Intelligent Agents for On-line Commerce: a Crying Need for Service Standardization," IEEE (1997).
- [34] Lee, J. G., Kang, J. Y. and Lee, E. S., "ICOMA: An Open Infrastructure for Agent based Intelligent Electronic Commerce on the Internet," Parallel and Distributed Systems, Proceedings of 1997 International Conference, pp. 648-655 (1997).
- [35] Sohn, S. and Yoo, K. J., "An Architecture of Electronic Market Applying Mobile Agent Technology," International Symposiums on Computers and Communications, (1998).
- [36] Coulouris G, Dollimore J and Kindberg T: 'Distributed systems: concepts and design', Addison-Wesley (1994).
- [37] D. Braess. A paradox of traffic assignment problems, <http://homepage.ruhr-uni-bochum.de/Dietrich.Braess/paradox.pdf>, 1968
- [38] R. Axelrod. Advancing the Art of Simulation in the Social Sciences, University of Michigan, August 2003

- [39] N. Minar, R. Burkhart, C. Langton, M. Askenazi. The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations, Santa Fe Institute Working Paper, <http://www.swarm.org/> 1996
- [40] M. Sierhuis, W.J. Clancey, R. van Hoof. Brahms: A multiagent modeling environment for simulating work practice in organizations. Journal for Simulation Modelling Practice and Theory, A special issue on Simulating Organisational Processes, Elsevier.
- [41] K. Carley, Y. Ren. ORGMEM: A Computer Simulation Program of Transactive Memory Systems, 1999
- [42] Matchmaking within Multi-Agent Systems, B. Pour Ebrahimi K. Bertels S. Vassiliadis K. Sigdel, Computer Engineering Laboratory, ITS, TU Delft, The Netherlands.
- [43] Gerhald Weiss. Multiagent Systems. The MIT Press, 1999.
- [44] Liberty, J., Programming C#,3rd ed. O'REILLY, 2003.
- [45] Teach Yourself SQL in 21 Days, 2 nd ed. SAMS, 2000.
- [46] Microsoft SQL Server 2000 Database Design, Element K Press, 2000.
- [47] Guttman, R., and Maes, P. Agent-mediated integrative negotiation for retail electronic commerce. In Proceedings of the Workshop on Agent-Mediated Electronic Trading AMET'98 (Minneapolis, May 1998).