<table>
<thead>
<tr>
<th>Titlul referatului</th>
<th>ASISTAREA ACTIVA A DECIZIEI IN COMERTUL ELECTRONIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor</td>
<td>Ciprian Cândea</td>
</tr>
<tr>
<td>Coordonator de doctorat</td>
<td>Acad. Florin Gh. Filip</td>
</tr>
</tbody>
</table>

ACADEMIA ROMÂNĂ
SECŢIA DE ŞTIINŢĂ ŞI TEHNOLOGIA INFORMAŢIEI
INSTITUTUL DE CERCETARI PENTRU INTELIGENŢĂ ARTIFICIALĂ

BUCUREŞTI, IUNIE 2013
Capitolul 1  Agents and Multi-Agents .................................................................................. 4
  1.1  Introduction ................................................................................................................. 4
  1.2  Agents ......................................................................................................................... 5
      1.2.1 Objects, Components and Agents ........................................................................ 6
      1.2.2 Multi agents ........................................................................................................... 7
      1.2.3 Application of multi agent systems ....................................................................... 8
      1.2.4 Communication and Coordination ......................................................................... 9
      1.2.5 Negotiation ............................................................................................................. 14
      1.2.6 Classification of negotiation in electronic commerce ............................................ 15
  1.3  Agent and Multi-Agent architectures .......................................................................... 16
      1.3.1 Deliberative, Reactive and Hybrid ......................................................................... 16
      1.3.2 Belief Desire Intention ........................................................................................... 19
      1.3.3 ROBOT – Holon like approach .............................................................................. 21
  1.4  Agent’s types ................................................................................................................. 27
      1.4.1 Interface agents ....................................................................................................... 27
      1.4.2 Information agents .................................................................................................. 29
      1.4.3 Middle agents – Intermediaries .............................................................................. 31
      1.4.4 Seller - Buyer and negotiation ................................................................................. 33
Capitolul 2  Active support techniques .............................................................................. 38
  2.1  Intelligent Agents and Decision .................................................................................. 38
      2.1.1 Human decision process .......................................................................................... 38
      2.1.2 Agents and Decision Support Systems .................................................................. 39
      2.1.3 Agent approach on e-Commerce ............................................................................ 41
  2.2  Information retrieval elements .................................................................................... 44
      2.2.1 Traditional vector space model .............................................................................. 45
      2.2.2 Weights .................................................................................................................. 47
      2.2.3 Representation of documents in the vector space .................................................. 48
      2.2.4 Query refining / envelop ....................................................................................... 49
  2.3  User profiling ................................................................................................................ 51
      2.3.1 Structure of the user profile, data filtering and profile quality ................................. 52
      2.3.2 Learning user profile – explicit and implicit feedback .............................................. 54
      2.3.3 Learn new interest .................................................................................................. 57
Capitolul 3  Search engine agents – active decision support system .............................. 59
  3.1  Introduction ................................................................................................................... 59
  3.2  Agent based IR system ................................................................................................. 60
  3.3  Architecture of the system .......................................................................................... 61
      3.3.1 Multi agent system for active support ................................................................. 61
      3.3.2 Profile agent .......................................................................................................... 63
      3.3.3 Adaptive recommendation ..................................................................................... 65
Capitolul 4  Conclusion .......................................................................................................... 67
Capitolul 5  References ......................................................................................................... 69
Capitolul 6  Annex .................................................................................................................. 76
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Agent and Environment</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>MAS architecture (a sample)</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Languages and levels of communication</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Phases of the contract net protocol</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Layered Agent Architectures (Nicholas Jennings et al., 1998)</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>Beliefs-Desires-Intentions architectural components</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Holonic like architecture (Ciprian Candea, Hu, Iocchi, Nardi, &amp; Piaggio, 2001)</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Generic holon architecture</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>Interaction of interface agent</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>Basic skills of an information agent (Klusch, 2001)</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>Matchmaker agent (Wong &amp; Sycara, 2000)</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>Facilitator agent (Wong &amp; Sycara, 2000)</td>
<td>32</td>
</tr>
<tr>
<td>13</td>
<td>Buyer agent – Seller agent interaction (from the viewpoint of seller agent) (Oprea, 2002)</td>
<td>34</td>
</tr>
<tr>
<td>14</td>
<td>The decision process (Pomerol &amp; Adam, 2008)</td>
<td>39</td>
</tr>
<tr>
<td>15</td>
<td>Intelligent decision support system based on intelligent agents (G. E. Phillips-Wren, 2008)</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>e-Commerce system based on intelligent agents (Badica et al., 2007)</td>
<td>43</td>
</tr>
<tr>
<td>17</td>
<td>Placing a document in the space defined by the &quot;Active&quot; and &quot;Independent&quot;</td>
<td>45</td>
</tr>
<tr>
<td>18</td>
<td>Query before and after refining</td>
<td>50</td>
</tr>
<tr>
<td>19</td>
<td>User profile representation using three descriptors categories of interest</td>
<td>52</td>
</tr>
<tr>
<td>20</td>
<td>The nature of changing long term interests (Widyantoro, 1999)</td>
<td>55</td>
</tr>
<tr>
<td>21</td>
<td>Overall architecture</td>
<td>62</td>
</tr>
<tr>
<td>22</td>
<td>User profile agent structure</td>
<td>63</td>
</tr>
</tbody>
</table>
Capitolul 1  Agents and Multi-Agents

KEY TERMS

Agent: An agent in computing is an entity such as a software program or a robot that can be viewed as perceiving and acting upon its environment and that is autonomous, that is, its behavior at least partially depends on its own experience.

Agent (-Based) System: An agent (-based) system is a computer system built around a software agent as its central component.

Distributed Artificial Intelligence: Distributed artificial intelligence is a sub-field of artificial intelligence concerned with systems that consist of multiple independent entities that interact in a domain.

Distributed Decision Making: Distributed decision making is a decision-making process where several people are involved to reach a single decision, for example, a problem solving activity among a few persons when the problem is too complex for anyone alone to solve it.

Interface Agent: Interface agent is a software agent that supports the user in accomplishing certain tasks or performs tasks on the user’s behalf.

Multi-Agent System: A multi-agent system is a system incorporating multiple, interacting agents.

Personal Assistant: See “interface agent.” Software Agent: Software agent is a software program that inhabits some complex dynamic environment, senses and acts autonomously in this environment and by doing so realizes a set of goals or tasks for which it is designed (P. 1994).

Software Assistant: See “interface agent.”

1.1 Introduction

The Agent and Multi Agent Systems (MAS) paradigm represents one of the most promising approaches to build complex and flexible new architectures required for next generation of intelligent tools offering a new dimension for large-scale integration.

MAS are software systems composed of several autonomous software agents running in a distributed environment. Beside the local goals of each agent, global objectives are established committing all or some group of agents to their completion. Some advantage of this approaches are: it is a natural way of controlling the complexity of large, highly distributed systems; it allows the construction of scalable systems since the addition of
more agents is an easy task; MAS are potentially more robust and fault-tolerant than centralized systems.

An important role for agents may be the delegation of tasks. Agents interact and negotiate with each other to determine a suitable contracting agent. The contract net model (Maturana and Norrie 1997) provides a suitable general protocol to design and implement this negotiation process.

The MAS provides a platform for co-ordination and co-operation, within which its agents can work collectively to solve specific problems. Clusters or teams of agents are identified (Carley and Lin 1995) to perform specific reasoning for a given task and decision-making responsibilities are delegated to co-ordination groups made up of these agents. After all, if we don't expect people to be omnipotent, why should we expect this from agents?

1.2 Agents

Classical computers software are best when are explicitly created – each action that must be performed is clear anticipated, planned for and coded by a programmer. Many times we perform some actions into computer software that where not anticipated and software crash at best, data lost at worst.

Nowadays daily usage of Internet for personal or business scope redefine the role of computers and the way that are used and the need of computation is shifted from standard calculation and processing to delegation and interaction (Fasli 2007).

Many times is convenient that computer software is executing a clear path – in case of bookkeeping software or payment process that is a must but more and more we are looking for applications where computer software that can decide what must be done in order to achieve the designed objective. This computer software is known as agents. Agents must run under unpredictable, changing or open environments as well they must present an architecture that is robust to respond at probability that their actions can fail.

Russell defines an agent as “anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors” (Russell and Norvig 2003).

![Figure 1 Agent and Environment](image-url)
One definition that is accepted and provide a starting point for actual paper is (M Wooldridge and NR Jennings 1995):

An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.

Definition identify some of relevant properties of an agent

- **Distinct problem solving entities** – well defined design objectives and capabilities
- **Embedded** in a particular environment
- **Autonomous** they have control over their internal state and behavior
- Capable of **flexibility** - reactivity, proactivity and sociability (context dependent behavior)
- **Mobility** - an autonomous entity that has the ability to migrate in a heterogeneous network and resume its executions (Fasli 2007)

Various other attributes of agent are formally defined on (Goodwin 1995).

### 1.2.1 Objects, Components and Agents

Agent technology is different from standard software first of all because the autonomy it undertakes to achieve its user’s goals. When we are looking at object oriented approach we find that are entities that encapsulate some state, are performing actions or methods on this state and communicate by messages.

So, first major difference is the autonomy of agents, an object implements principle of encapsulation – the object have control over his own internal state but the decision lies with the object that invokes the methods. On the other side, agent, decision lies with the agent that receives the request. (Michael Wooldridge 2002)

Second major difference is addressing other property of the agent: flexibility (reactive, proactive, and social). An object model does not have such characteristics; it is possible to build into objects these behaviors but is the standard object oriented approach has nothing to do with these characteristics.

Third difference refers the way that agents and object runs into computer hardware. Agents are running on their own thread of control – and are supposed that can have also a multi-threading structure inside of agent. On the object we don’t have a dedicated thread for each object; all of them are instantiated in the context of application thread. (Michael Wooldridge 2002)
1.2.2 Multi agents

A multi-agent system represents collection of agents that communicate and interact with one another to solve problems that have been delegated by users.

The Multi Agent Systems (MAS) term is now days is used as a generic designation for all types of systems that show autonomous or semi-autonomous behavior.

Distributed Problem Solving considers how a problem can be solved by a number of independent modules that cooperate to solve it by dividing and sharing knowledge about the issue. In a distributed problem solving strategies interaction are all incorporated as an integral part of the system.

In contrast, MAS focuses on behavior that is developing a collection of pre-existing autonomous agents that help to solve this problem. MAS can be defined as an unbundled network processing units that work together to solve a problem solving exceeding or knowledge of any of the agents. These agents are autonomous and may be heterogeneous in structure.

I can outline principal characterizes of a MAS:

- Agents has incomplete information and / or limited capabilities for solving the problem; agent has a limited perspective on the global problem
- No centralized control on the whole system
- Data are not centralized – decentralized data
- Asynchronous execution

I can outline principal reasons that are taken in account when MAS is chooses to be used for specific problem:

- Efficiency and robustness
- Interoperability – even if are different standards that must be used as core
- Can act / solve problems in situations when data, expertise and control are distributed

When MAS is realized have to be taken in account also: communication language between agents, decomposition and task allocation, etc.

In next figure (classical MAS systems that show categories of agents that cooperate in a team – as team members we have also humans. Such agents, interface agents (individual assistant), mediators (facilitate communication between individual assistant and resource agent), resource agent (implement native operations of the service supplier) (Zamfirescu, Bárbat, and Filip 1998).
Within their team concept is more general, considered as a group of entities (human or artificial) working together to solve a problem or make a decision that could not be taken individually by any of them individually (Yin et al. 2000).

### 1.2.3 Application of multi agent systems

Usage of MAS appear as a solution when you have a diverse, heterogeneous and distributed information sources (on line or off line) but also in the situation that projected system must provide a large, complex and distributed information processing. In all these cases using a multi agent system you can benefit also from an organized behavior.

A system that implements MAS will present at least next advantages:

- Concurrent processing – speed
- Less communication bandwidth – processing can take place near to the information source
- Modular system development is easier – due to the fact of decomposition in agents

Main generic categories of domains where MAS find real applications are (Lesser 1999):

- Distributed situation assessment – network diagnosis, information gathering on the Internet, distributed sensor networks, etc. – agents with different level of understanding and control, share local interpretation to achieve a common and consistent level of understanding and response.
- Distributed resource scheduling and planning – factory scheduling, network management, etc. – agents coordinate their schedules to resolve conflicts over resources and to maximize system output

Figure 2 MAS architecture (a sample)
Distributed expert systems – concurrent engineering, network service restoral, etc.
- agents are sharing information and work together (negotiation) to a common solution; each agent participate with different expertise and solution criteria

Over this generic domains where MAS can be used also area like e-commerce where agents care represents interest of different entities (organizations or personal). As example are systems that describes a “electronic marketplace” (Chavez and Maes 1996) where system are using agents for “buy” and for “sell” for each trading good and commercial transaction is made by this agents. Other examples are found from agents that discover cheapest good – example CDs (Krulwich 1996), to personal shopping assistant that is capable to search on line stores for product availability and price confirmation (Doorenbos, Etzioni, and Weld 1997).

In (Nicholas Jennings, Katia Sycara, and Wooldrige 1998) we can find a list of domains where agents are used in practical applications like manufacturing, process control, telecommunications, air traffic control, transportation systems up to entertainment industry – games and interactive theatre and cinema and medical sector – patient monitoring and health care.

Other use of Multi-Agents is for simulating different behaviors and allow to model of decision processes and actions of individual agents – final customers, sellers, traders and even regulators (that come with rules on the market). Unlike traditional tools an Agent-Based simulator does not come with a single decision maker with a single objective for the system and uses agents, which represent the different independent entities that are allowed to establish their own objectives and decision rules. Also as the simulator run agents can adapt their strategies, based on the success or failure of previous steps.

In (Praça et al. 2003) we have a model of a Multi Agent simulator that is focused on electricity market that involves agents that represents entities that are found on this market: generators, consumers, market operators and network operators plus traders – as in reality.

### 1.2.4 Communication and Coordination

#### 1.2.4.1 Communication

Communication is a fundamental topic in real life as well in computer science to enable exchange of messages and to coordinate actions.

For agents communication is necessary to enable coordination’s of their actions and behaviors in a shared environment; environment where agents must exchange messages: engage in a dialog.
Based on these needs, different researchers as well as commercial organizations investigate and propose communication languages for intra-agent.

![Figure 3 Languages and levels of communication](image)

Agents can play different roles within the communication process: master, slave or peer and are two basic message types: queries and assertions, and each agent must be able to accept information through an assertion and to formulate a response.

Structure of messages is defined by the communication protocol that can be binary (that involves one sender and one receiver) and n-ary (multiple receivers). A minimal protocol will address two layers of a message: transport information and content where transport information represents at least the sender and receiver addresses. The content of the message can be represented by the Agent Communication Language (ACL) that defines two layers: syntax (actual format of the message) and semantics (meaning of the message).

In different ACL implementation we find syntax (i.e., KIF knowledge interchange formant), semantics (e.g., Ontolingua – define and maintenance of sharable ontologies), pragmatic (e.g., KQML - Knowledge Query and Manipulation Language).

As described by the designers themselves, KIF is an "interlingua" for encoding declarative knowledge by means of predicate logic, a language dedicated to knowledge reuse, implementation independent semantics, and having a strong expression.

Ontolingua is a Web service for building, publishing, and sharing ontologies (a common vocabulary with generally accepted meanings to describe a particular subject area) that has an editor based on a Web interface that allows automatic translation of ontologies in a certain language content (including KIF, Prolog etc). And also includes primitives for combining different ontologies.

KQML is a language-level communication, message-oriented and at the same time a protocol for exchange of information regardless of content ontology and syntax. KQML is independent of:

- Transport mechanism (e.g., TCP/IP, email, CORBA, IIOP, etc.),
- Language (e.g., KIF, SQL, STEP, Prolog, etc.)
- Ontology

It also includes primitive message types that are of particular interest for the construction of different architectures of agents (e.g., media, sharing intentions etc.).

FIPA (Foundation for Intelligent Physical Agents) consortium defines a similar with KQML inter-agent communication language.

FIPA is a non-profit primarily aimed at promoting the development of technology specifications for generic agents that facilitate interoperability within and between applications, services and equipment.

Communication language proposed by this organization called FIPA ACL is based on the principles of natural language in which messages are actions (communicative actions) and where acts of communication are described both in narrative form and in formal semantics based on predicate logic. The syntax is somewhat similar to that used in KQML specifications providing a normative description of a protocol-level interaction (Labrou and Tim Finin 2000).

JADE (Java Agent Development framework), now days is probably the most widespread agent-oriented middleware in use that is based on FIPA ACL.

JADE is one of the middleware platforms that allow development of complete agent-based applications and is compliant with the FIPA specifications. In some ways JADE extended the FIPA model in several areas, but in all aspects related to interoperability, the core purpose of FIPA.

The framework facilitates the development of complete agent-based applications by means of a run-time environment implementing the life-cycle support features required by agents, the core logic of agents themselves, and a rich suite of graphical tools. It also presents possibility to allow easy extension with add-on modules.

As JADE is written completely in Java, it benefits from the huge set of language features and third-party libraries on offer, and thus offers a rich set of programming abstractions allowing developers to construct JADE multi-agent systems with relatively minimal expertise in agent theory. (Bellifemine, Caire, and Greenwood 2007)

One major contribution was definition of the Xtensible Markup Language (XML, 2001) and its close relative, the DARPA Agent Markup Language (DAML, 2001) and of the semantic web.

One of the lacks of Web is possibility to have semantics within Web pages, and in response to that XML was developed. Using XML and latter web semantics it becomes possible to add semantics within web pages and for computers easy to process the information in a meaningful way.

Encoding ACL messages in XML will offer advantages like (Labrou and Tim Finin 2000):
The XML-encoding is easier to develop parsers and provides parsing information more directly; the process of developing and maintaining a parser is much simplified.

WWW-friendly – using XML makes ACL more accessible for software development and facilitates software engineering of agents; easy integration with a variety of Web technologies.

1.2.4.2 Coordination

Coordination is defined as a process where agents communicate and work together in order to ensure that community of agents acts in a coherent manner (Nwana, Lee, and N.R. Jennings 1996). We can see different reasons why multi agent systems need to coordinate:

- Agents’ goals may cause conflicts among agents’ actions,
- Agents’ goals may be interdependent,
- Agents may have different capabilities and different knowledge, and
- Agents’ goals may be more rapidly achieved if different agents work on each of them.

Coordination of agents can be achieved with different approaches like: organizational structuring, contracting, multi-agent planning and negotiation.

Organizational structuring provides a framework for activity and interaction through the definition of roles, communication paths and authority relationships (Durfee 1999).

Using an agent who has a broader view of the system, the group of agents will be able to get consistent behavior and conflict resolution, exploring the organizational structure or hierarchy. This is a classical technic a master / slave or client / server approach for task and resource allocation. The master agent play controller role and gather information, create plans and assign tasks to individual agents in order to ensure global coherence. This technic is hard to be used in practical situation because of difficulty to create such a controller and is contrary to the decentralized nature or multi-agent systems.

Contract net protocol (Smith and R. Davis 1980) was introduced as a coordination technique for task and resource allocation and agents can take two roles: manager or contractor. The basic idea is that an agent that cannot solve an assigned problem using local resource/expertise will decompose the problem intro sub-problems and try to find other agents willing and with right expertise to solve it. The assigning the sub-problems are solved by a contracting mechanism with next steps (fig. 4):

1) Contract announcement by the manager agent
2) Submission of bids by contracting agent in response to 1)
3) Evaluation of the bids and decision on the winner (contractor(s))
Coordination of agents can be addressed as a planning problem (centralized plan or distributed plan) where agents build a multi-agent plan that details all the future actions and interactions required to achieve their goals and interleave execution with additional planning and re-planning.

In centralized planning approach, we have an agent that coordinate all activity based on information of partial or local plans from individual agents, analyses them to identify inconsistencies and possible conflicts (e.g. over limited resources). Then the agent attempts to modify these plans and combine them into a multi-agent plan where conflicting situations are resolved (Georgeff 1983).

On opposite in the distributed multi-agent planning, the idea is to provide each agent with a model of other agents’ plans. Agents communicate in order to build and update their individual plans and their models of other agents until all conflicts are removed (Georgeff 1983).

A unify approach is to integrates the strengths of the organizational, planning, and contracting approaches into a single approach (Durfée and Victor 1987). With this approach we gain the multi-agent planning benefits of detailed, situation-specific coordination while avoiding excessive computation and communication costs.

Using partial global planning, coordination involves both sharing tasks and sharing results; both adhering to long-term organizational roles and reactively planning to achieve short-term objectives.

Negotiation is probably one of the most used techniques for coordinating agents. As described in chapter 1.2.5 negotiation is about the process that will come at an end and is about the path to arrive there.

Negotiation can be competitive or cooperative depending on the behavior of the agents involved. Competitive negotiation is used in situations where agents have independent goals that interact with each other; they are not a priori cooperative, share information or willing to back down for the greater good. Cooperative negotiation is used in situations where agents have a common goal to achieve or a single task to execute. In
this case, the multi-agent system has been centrally designed to pursue a single global goal.

1.2.5 Negotiation

Negotiation is defined by many authors as *Negotiation is a basic means of getting what you want from others* (Ury 1992), or *Negotiation is a field of knowledge and endeavor that focuses on gaining the favor of people from whom we want things* (Cohen 1982) or *negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter* (Bussmann and Müller 1992)

I can say that *negotiation is a process that will come at an end and is about the path to arrive there.*

On this process we need to know the objective and also context of negotiation as well the background of the other side. A series of aspects will affect the negotiation process:

- Time – specific time constraints (deadlines, a sense of urgency, etc.)
- Environment – physical environment, professional or cultural expectations
- Personalities – temperament and behavior plus number of people that attending to process
- Information – knowledge or lack of; personal knowledge base as well perception of the one another’s
- Personal issues – distractions external to the meeting, personal identity/ability
- Hierarchy – ability to command / demand / deference

All these conduct to a three dimensional feeling, making every negotiation fluid; constantly changing and create a sense of something always happening. “Web of tension” is the term that (Cohen 1982) used to capture this context.

When we must reach an agreement on variety of issues we as humans make use of negotiation. On similar way automated negotiation can be used as a dominant mode of operation for shopping agents. Using automated negotiation we can observe that time can be reduced but more interesting is that can be avoided some of the reticence of humans to engage in negotiation (e.g. because of personality).

When we came to automatic negotiation involves the design of high level protocols for agent interaction as well formalization of negotiation (Michael Wooldridge 2002) (N. R. Jennings et al. 2000).

We can find negotiation in to many different fields and as results different authors define automated negotiation (Rosenschein and Gilad 1994) and I can simple say that “process that involves communication between two or more agents with the scope to reach an agreement is negotiation”.
1.2.6 Classification of negotiation in electronic commerce

In (Michael Wooldridge and Nicholas R Jennings 2001) authors introduced a classification scheme for negotiation in electronic commerce and they identified the main parameters on which any automated negotiation depends.

Main parameters on which the negotiation can take place introduced by (Michael Wooldridge and Nicholas R Jennings 2001) are:

- **Cardinality of the negotiation**
  - Negotiation domain: single-issue or multiple issues (e.g. price, quality, warranties, delivery time, etc.)
  - Interactions: one-to-one, many-to-one, many-to-many (it not refer the total number of agents on the field but the field type – we can have 100 agents but they can interact only one-to-one)

- **Agent characteristics**
  - Role (e.g. – buyer or seller or trader)
  - Rationality (type of algorithm and time to run)
  - Knowledge (e.g. knowledge about opponent’s utility functions)
  - Commitment
  - Social behavior
  - Bidding strategy

- **Environment and goods characteristics** (environment can be static or dynamic; a static one is when variables – e.g. price – are constant over time; this affect utility function of the agent – static / dynamic)
  - Private/public value of the goods (e.g. cake or bonds – when buy a car you have both – one personal preference and how car will preserve its value over time)
  - Nature of the goods

- **Event parameters**
  - Bid validity
  - Bid visibility
  - Clearing schedule and timeouts (e.g. during a bidding phase of a English auction each round terminates with a temporary allocation of the good being auctioned to the prospective buyer that meets the auctioneer’s call)
  - Quotes schedules

- **Information parameters** (information that can help buyers and sellers reach agreements)
  - Price quotes
  - Transaction history
  - Arguments

- **Allocation parameters** (apply only to many-to-one and many-to-many; allocation is studies in auction theory)
1.3 Agent and Multi-Agent architectures

1.3.1 Deliberative, Reactive and Hybrid

The traditional approach to building artificial intelligent systems (also known as symbolic AI) is based on the fact that intelligent behavior can be generated within a computer system through the symbolic representation of the environment in which it operates together with that of the desired behavior to manipulate the point of these structures syntactically. Such a system should explain how the agent will behave, how the agent generates its own purposes to meet the design objectives and how agent will interpose with the opportunistic reaction to behavior in order to achieve his purposes.

In this category of agents internal state is supposed to be a knowledge base composed of predicate logic. Knowledge bases are actually information the agent has about the environment in which it operates, with a role similar to the views / opinions of human.

Let $L$ be a set of logical sentences and
$D = \emptyset \ (L)$ set of knowledge base $L$
Internal state of an agent is then an element of $D$; $\Delta, \Delta_1 \ldots$ members of $D$.
Internal state of an agent becomes a member of the set $D$.

A decision making process of an agent is modeled by a set of rules of inference $\rho$. These are obvious logical inference rules; internal state of an agent is a member of the set $D$.

$D \vdash \rho \Phi$ where $\Phi$ formula can be demonstrated based on knowledge base $D$ using only inference rules $\rho$.

It can formalize this perception of an agent function see: $S \rightarrow P$, and similarly a function like next: $DXP \rightarrow D$ that maps a knowledge base and a perception a new knowledge base. A function selection action to the action agent $D \rightarrow A$ is as defined in terms of the rules of inference.

When we start to create a deliberative agent programmer will encode the agent $\rho$ inference rule and knowledge base $D$ such that if it can be derived a formula $Do \ (a)$, where $(a)$ is a term which shall designate an action, then it is the best action to perform.

In this case, the agent tries to find an action that is consistent with the rules and the knowledge base, e.g. one that is not explicitly prohibited. Therefore the agent will try to find an action $\in A$ such that $Do \ (a)$ cannot be derived from its knowledge base using its own rules of inference. However, if the agent fails to find an action that is consistent means that no action has been selected. In this way the agent's behavior is determined by rules of inference and its current knowledge base that represents information about the environment in which the agent operates.
This elegant approach presents several disadvantages. Decision-making is based on the assumption of rationality - the assumption that the world will not change in a significant way while the agent decides what to do. The problems associated with the representation and reasoning in relation to complex and dynamic environments remain substantially unresolved.

Based on these findings the researchers tried a completely different approach. This can be identified by:

- Eliminating symbolic representations and decisions based on syntactic manipulation;
- Intelligent behavior / rational is perceived as inherent in the environment in which an agent operates (intelligence is the interaction with the environment);
- Intelligent behavior arises from the simple interactions

Alternative approaches are sometimes defined as behavior (because the common theme is the development and combination of individual behaviors) located / placed (because the basic theme boils down to the fact that agents are situated in a rather disjointed from it) and finally reactive (because such systems are seen as systems that react to changes in the environment without reason about it).

The best known of this class architecture is the architecture of summation (subsumption architecture) developed by Rodney Brooks. Are distinguishing two defining characteristics of this architecture;

First the agent decision making is achieved through a set of tasks that compose agent behavior. Each behavior can be seen as a function of individual action which continuously takes data from the environment in which they operate and which maps an action to be achieved. Each of these behavioral modules is dedicated to achieving a particular task. It is assumed that is not used in any symbolic reasoning (and no search). In most implementations of these behaviors are implemented as rules of the form of action $\Rightarrow$ situation simply maps the situation perceived by the sensors input the desired action.

A second defining characteristic is that it may be activated multiple behaviors. The mechanism proposed by Brooks to select the desired action in these circumstances is limited to arranging a hierarchy of behaviors in summation, the behavior is conceptualized on the levels of this hierarchy. Lower levels can be inhibited by the top or, in other words, the layer is on a lower level to the higher its priority. The upper layers are in this sense more abstract behaviors. According to the decision of the action is carried out by means of a set of behaviors inhibition together with a relationship between them.

Even if reactive architecture is looking simple presents a series of disadvantages:
• If the agent does not use models of the environment in which they operate when it should have enough information to select the desired action.
• Given that purely reactive agents make decisions based on local information (e.g. information about the current status of agents) is difficult to see how such a decision could consider external information thus a limited perspective on the context of the problem.
• Learning from experience and improvement of their performance is difficult to be implemented
• Agents are composed of layers, if we have more than ten layers, dynamic interactions between different behaviors become too complex to be understood.

In different situations you can decide to use deliberative or reactive architecture but are many situations that none of them are appropriate to be used.

For these situations a different architecture is needed where to use deliberative as well reactive approaches, *hybrid* architecture.

Figure 5 Layered Agent Architectures (Nicholas Jennings, Katia Sycara, and Wooldrige 1998)

Usually we have at least two layers to handle the behavior of reactive and deliberative plus two modes of control of the information flow in the layered architecture (see next figure):

• Horizontal layering – software modules are each connected directly to the sensor input and to which the work - as a consequence of each module acts as a single agent
• Vertical stratification – inputs and actions undertaken to more than one layer

Horizontally layered architecture big advantage is conceptual simplicity, e.g. if an agent need not manifest different behaviors when it will be implemented in different layers. Because they compete with each other to generate desired actions danger is that the overall behavior of the agent is not very coherent. In order to ensure the consistency of this type of architecture it typically includes a mediation function which allows only a single time to have control over the coating.
Complexity of horizontally architecture is: \( m^n \), where \( m \) - possible actions; \( n \) - layers numbers.

Vertical architectures can be divided into unidirectional data flow architectures and architecture with bidirectional information flow. In unidirectional data flow architecture, the information flow passes sequentially through each layer until it reaches the final layer which generates the final action. In the second case the information flow through the component modules in two phases, somewhat similar to the organizational style.

In both architectures the complexity of interactions between layers is reduced and can be computed as \( m^2(n-1) \), \( n \)-layers numbers; \( m \)-actions numbers.

Layering is a natural decomposition of functionality is easy to analyze how the reactive and proactive behavior can be generated by dedicated layers. One problem with this architecture is the lack of conceptual and semantic clarity compared to the others' approach. Another problem is that the interactions between layers. If each layer in a separate process it is necessary to consider all possibilities that these modules can interact with each other.

### 1.3.2 Belief Desire Intention

BDI architectures derive from psychological attempts to understand practical reasoning.

This kind of reasoning involves two important processes:

- The choice of goal to be achieved
- How to act to achieve it

The initial process is known as deliberative process. In practical reasoning intentions play several roles:

- Influence the practical reasoning,
- Constrain future deliberations,
- Intentions persist,
- Influence opinions reasoning underlying the future.

A key issue in the design of practical reasoning agents is to achieve a balance between these different approaches. In particular, it is clear that an agent should be from time to time to give up some intentions (whether these intentions will have a chance to ever be made, there is no motivation to achieve those intentions). Therefore it is important for an agent to reconsider its intentions. But this requires also a cost - both in time and computing resources that, which introduces some remarks:
An agent will not reconsider intentions often will persevere in trying to achieve their own intentions even when it is obvious that they cannot be reached or there is no reason to be made.

An agent that constantly reconsiders its own intentions may allocate insufficient time actually work necessary to achieve initial intentions and therefore they threaten to fruition.

In next figure it is described structure of BDI agent and can find seven components where can observe that set $Bel$ (the set of all possible views) $Des$ (the set of all possible desires) and $Int$ (set of all possible intentions):

- Set of opinions (beliefs) that represents the current information the agent has about the environment in which they operate,
- Position to review the opinions (BRF) which input data and current perceptions of the agent determines the new set of views,
- Generating a function options (options) that determine the options available to an agent,
- Set of options available to the agent representing the possible actions that the agent has
- Filter function (filter) to represent the deliberative process of determining the intentions of the agent and agent-based views, wishes and intentions of its current
- Set of current intentions, representing the agent's current concern, and
- Selection function (execute) which determines the action to be performed on the basis of current intentions.

![Beliefs-Desires-Intentions architectural components](image)

The state of a BDI agent at any time can be represented as a triple like $(B, D, I)$ where $B \subseteq Bel$, $D \subseteq Des$ and $I \subseteq Int$. Review function can be described as BRF: $\varnothing (Bel)xP \rightarrow \varnothing (Bel)$ that based on the current opinions and input determines the new set of views.

Mapping a set of believes and a set of intentions on a set of desires is represented as: $\varnothing (Bel) x \varnothing (Int) \rightarrow \varnothing (Des)$. This function plays multiple roles. First is the process responsible for deciding how the agent will achieve the intentions. Once an agent has formed an intention $x$ then it should consider options for implementation of this intention. These options will be more concrete (less abstract) than $x$. Some of these options can become intention; they will influence the resulting generation options resulting in more concrete options. The generation of options in a BDI agent can be seen as an elaboration recursive hierarchical structure plan.
As long as the primary purpose of function options is to achieve pragmatic reasoning, it must additionally satisfy a number of other constraints.

First, it must be consistent; all options must be generally consistent with both the current views and current intentions.

Secondly, it has to be opportunistic, function should notice when environmental circumstances have changed favorably, offering new ways to achieve agent's intentions or their ability to meet some intentions that in another case would be was intangible.

Deliberative process of a BDI agent (decision to know what to do) can be represented by a filter function: \( \varphi(\text{Bel}) \times \varphi(\text{Des}) \times \varphi(\text{Int}) \rightarrow \varphi(\text{Int}) \) that modifies the agent's intentions based on the old intentions, current views and desires.

This function performs two roles.

- First, has to eliminate any intention no longer available or cost for achieving it exceeds earnings related.
- Secondly, it should retain intentions were not realized and expected in the near future to have a positive overall benefit. Finally, it should adopt new intentions or to achieve existing intentions or to exploit new opportunities.

Such filter function should satisfy the following constraints \( \forall B \in \varphi(\text{Bel}), \forall D \in \varphi(\text{Des}), \forall I \in \varphi(\text{Int}); \text{filter} (B, D, I) \subseteq I \cup D \). In other words, current intentions are anterior intentions or are new options that are adopted. The function will return simply any executable intent, intentions that correspond directly with executable actions

\( \text{execute} : \varphi(\text{Int}) \rightarrow A \). Note that the representation of an agent's intentions as a set (as an unstructured collection) is generally too simplistic daily practice. A simple alternative is to associating priorities to each intention indicating their relative importance. Another method would be to represent the intention in the form of a stack.

The BDI is attractive because is very intuitive and purpose us a functional decomposition that is a much clearer compared to other architectures. The great difficulty lies in the way these features will be implemented effectively.

### 1.3.3 ROBOT – Holon like approach

Although established as wide-ranging trends in IT, some paradigms are not yet as pervasive, diversified, and – most of all – integrated (in both senses: incorporated as well as intertwined) in real-time control systems, as they should be.

The chapter considers two of them:

- Agent-orientation – particularly, multi-agent systems (MAS) (Nwana and M. Wooldridge 1997);
- Holonic approach to manufacturing systems (Hermans K. et al. 1999);
The state of the art in these fields is elaborated upon passing it through a threefold filter regarding real-time applications: high complexity, vital robustness, and critical response time.

This architecture was for the first time introduced, implemented and tested in real case by author, first time in year 2000 (C Candea, Staicu, and B Barbat 2000). The tests were conducted in 2000 and 2001 in the RoboCup competition under the simulation league. Simulation league is played by computer agents; each team is formed by 11 players, each of them implemented as a separate agent. The simulation is run by a Soccer Server (Noda et al. 1997).

The concept of “holon” has been proved to be very successful in the industrial process planning, increasing the flexibility of decisional systems.

The term “holon” was proposed by the Hungarian philosopher A. Koestler (1967, 1978) and explains the importance of the hierarchy of a system. Each organ is an element of the organic system while the organ is itself a system composed of multiple tissues. This relationship appears at every level of the system. This means that a system element is located at a hierarchy node and has both characteristics as a whole and as a part. Koestler named the node of hierarchy “holon” (Hino R. and Moriwaki T. 1999) based on combination of the Greek word “holos” that means “whole” and the suffix “on” meaning particle or part. Accordingly a “holarchy” is a hierarchy of self-regulating control buildings blocks (holons), which function:

- As autonomous wholes in supra-ordination to their parts
- As dependent parts in subordination to controls on higher levels
- In co-ordination with their local environment

Why holons?

Starting from the fact that “Holonic Manufacturing Systems” (HMS) were set up as a new approach to the manufacturing control problem (Hermans K. et al. 1999), the paper defends, among others, that:

- Because of one of the main weaknesses of MAS – the practical impossibility to deal with more than two levels (the agent and the system) – the holonic paradigm allows better modeling of multilevel systems (including the player-team-coach ensemble).
- It increases also the flexibility of decisional systems (as both HMS and robotic teams); indeed, a “holarchy” is (Hino R. and Moriwaki T. 1999) “a hierarchy of self-regulating control building blocks (holons), which function as
  - as autonomous wholes in supra-ordination to their parts,
  - as dependent parts in subordination to controls on higher levels,
  - in co-ordination with their local environment."
One of the most important characteristics of holarchies is the capacity to modify themselves, e.g. to create temporary hierarchies (Giebels, Kals, and Zijm 1999) (like modern industry, soccer is very dynamic: not only that each team comes with its own style and game strategy, but also each game phase has a dose of novelty).

Holarchies offer a balance between the two usual approaches to the guided process: the hierarchical control (fixed, static, pre-established) and the heterarchical one (autonomous, decentralized, flexible).

Holons specialization and aggregation offer a great flexibility in testing the system at different levels of details during the fine-tuned phases. Aggregated holons are defined as a set of related holons that are clustered together and form in their turn a bigger holon with its own identity, so holons may belong to multiple aggregations at the same time. Aggregated holons can dynamically change their contents depending on

Idea is inspired by the PROSA architecture developed at PMA/KLeuven as a reference model for Holonic Manufacturing Systems (H. Van Brussel 1994). The acronym PROSA came from Product-Resource-Order-Staff Architecture, the holon types used.

The resource holon contains a physical part namely a production resource of the manufacturing system, and an information processing part that controls the resource.

The product holon holds the process and production knowledge to assure the correct making of the product with sufficient quality.

The order holon represents a task in the manufacturing system. It is responsible for performing the assigned work correctly and on time.

The staff holon is implemented in the idea to assist the other three holons in performing their work.

Based on this architecture we propose a similar approach for robotic teams (see next table). The holarchy is structured on five levels (RoboCup, Coach, Team, Player, Component), each of them containing the specific holons (Product, Resource, Order, Staff). Their role, functionality and cooperation mechanisms are described and discussed in the RoboCup context. A more detailed description is given in (C Candea, Staicu, and B Barbat 2000).
At the player level are the holons who handle the resources (i.e. player stamina), implement the skills (i.e. pass, dribble, score), model the game tactic and coordinate the body parts in future. The resource holon handles the player stamina, simulation cycle and implement skills (analytical methods).

At the product level we have more complex problem like tactics where we used a fixed decision tree obtained with an off-line algorithm, but also we have to implement the necessary knowledge to resolve the advice contains strategies request comes from other holons.

At the Order level we have more abstract and complex task like learn. At this level we implemented a neural network (C Candea, Oancea, and Volovici 2000) that was used to obtain the strategic positioning when players don’t have the ball control. This type of neural network we used also in competition with promising results.

This architecture can control better our players because when a new situation appears immediately one of the holons will react at the environment changes. Is also possible that more than one holon to react at the changes; for example one react using reactive modules and the other one have some other plans generated by deliberative level and the negotiation process begin. At the moment only a very simple negotiation model based on fixed priorities have been tested.

### Holarchy Levels

<table>
<thead>
<tr>
<th></th>
<th>RoboCup</th>
<th>Coach</th>
<th>Team</th>
<th>Player</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Championship</td>
<td>Selection</td>
<td>Game</td>
<td>Skills acquired</td>
<td>Applied</td>
</tr>
<tr>
<td></td>
<td>Workshops, etc.</td>
<td>Training</td>
<td></td>
<td>Tactics learned</td>
<td>skills</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Strategies</td>
<td></td>
<td>Implemented strategies</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tactics</td>
<td></td>
<td>Implemented tactics</td>
<td></td>
</tr>
<tr>
<td><strong>Resource</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Organizational</td>
<td>Strategies</td>
<td>Time</td>
<td>Skill</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Technical</td>
<td>Tactics</td>
<td>Information</td>
<td>Schemata</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Financial</td>
<td>Experience</td>
<td>Schemata</td>
<td>Stamina</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rules</td>
<td></td>
<td>Components</td>
<td></td>
</tr>
<tr>
<td><strong>Order</strong></td>
<td></td>
<td>Building teams</td>
<td>Win game</td>
<td>Fulfill role</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Research</td>
<td>Testing teams</td>
<td></td>
<td>Preservation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entertainment</td>
<td>Winning</td>
<td></td>
<td>Learn</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>championships</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Staff</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>All individuals</td>
<td>Players</td>
<td>Spare players</td>
<td>To be adapted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>involved</td>
<td>Teams</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

Table 1 Holon-like approach for robotic teams

At the player level are the holons who handle the resources (i.e. player stamina), implement the skills (i.e. pass, dribble, score), model the game tactic and coordinate the body parts in future. The resource holon handles the player stamina, simulation cycle and implement skills (analytical methods).

At the product level we have more complex problem like tactics where we used a fixed decision tree obtained with an off-line algorithm, but also we have to implement the necessary knowledge to resolve the advice contains strategies request comes from other holons.

At the Order level we have more abstract and complex task like learn. At this level we implemented a neural network (C Candea, Oancea, and Volovici 2000) that was used to obtain the strategic positioning when players don’t have the ball control. This type of neural network we used also in competition with promising results.

This architecture can control better our players because when a new situation appears immediately one of the holons will react at the environment changes. Is also possible that more than one holon to react at the changes; for example one react using reactive modules and the other one have some other plans generated by deliberative level and the negotiation process begin. At the moment only a very simple negotiation model based on fixed priorities have been tested.
To obtain a collaborative team was necessary to implement the team level where the biggest problem was at the resource level because here the information are shared with the rest of the team mates and the schemata selection is based on the previous off-line learned knowledge.

In case of architecture for RoboCup was used an architecture of agent composed from different holons as in next picture.

Figure 7 Holonic like architecture (Ciprian Candea et al. 2001)

At first level we have the environment (RoboCup) that defines what are the rules and objects that will interact.

The coach level try to acquire strategic and tactical information from past games (in the training sessions), they also train the players, monitoring and advising the player in the real game, and in the future, when it will be possible to change the players dynamically, it will select the players.

At player level are the holons who handle the resources (e.g. players stamina), implement the skills (e.g. pass, dribble, score), model the game tactic and coordinate the body parts (3D soccer simulator).

All these levels should be understood as a holarchy, not as a hierarchical organization.

Looking more in detail a holon is composed from different blocks and communicate with the outside of world on two channels as is descried in next picture.
The reactive layer – that is indispensable – underscore one of the main common features of holons and agents.

The most important part of holon is the planning process set up in the deliberative layer. The planning process uses data acquired through the two communication layer:

- With the environment - receiving stimuli and reacting to them
- With other holons - taking advice from the higher level, negotiating with adjacent holons, monitors subjacent holons.

The negotiation part (implemented in the communication layer) is activated in two situations.

First situation is when it receives some advice that can’t be accepted (executed); in this situation it can inform the caller what can be done (e.g. has other tasks more important for it or it can’t implement the advice because of lack of capacities)

Second situation is when it sends an advice to another holon and receive rejection. Based on refuse it can begin the negotiation process, which consist in redefining the advice (less requirements) or in task decomposition (it will try to assign more holons to solve the initial task).

Based on experience I purpose a name for this kind of architecture ROBOT that exploits the domain specific application, it results from applying PROSA to robotic domain.
The holonic approach, initially developed for applications within rather controlled environment, has been adapted to the RoboCup framework. Promising results have been reported from industrial application domains (e.g. transportation and manufacturing) where decisional flow is somehow pre-established and holons typically deal with quantitative-based decisions. The application of the holonic approach to the RoboCup environment has required non-trivial adaptations in order to cope with the dynamics of the RoboCup games.

Such an adaptation has shown that the holonic approach can be applied also in other domains with more dynamic scenarios (Ciprian Candea et al. 2001).

For dynamic like environments holonic architecture provides for more operational adequacy regarding order, planning, monitoring, and stability along entire system lifecycle. In addition, the decision making process is improved, since holons can follow competently all kind of rules in the various system components (e.g. RoboCup game rules, coach directives, and so on).

### 1.4 Agent’s types

#### 1.4.1 Interface agents

Interface agents are the most important agents because of the perception of a system that is categorized as active one.

Main interaction method that defined the human interaction with computer system is the *directly manipulating*. Implications have led to the need for the user to initialize and monitor the programs that runs for him. Increasing the efficiency and effectiveness of user that don’t have a high experience led to the replacement of this type of approach. Techniques from the field of AI, in particular intelligent agents have been proposed for implementing a complementary style of interaction, called *indirect leadership*. In this situation orders initiated by the user via direct manipulation are replaced with a cooperative process in which both the human factor and the system - through agents - initiate communication, monitoring and completing tasks; and we obtain a new paradigm – personal assistant.
Personal assistant is becoming more efficient as learn user's interest, his work style and preferences. In principle, agent should not obstruct the user, in one way or another, to act on their intentions.

Agent behaves like a personal assistant working with the user to perform the task assigned and not as an interface between the user and application. The user can thus ignore all advice or recommendations. User support can manifest very differently from hiding the complexity of tasks, to achieve a certain task instead, etc.

The idea of using agents in the interface for the delegation of certain tasks was introduced by Nicholas Negroponte (1970).

In this approach there are two fundamental issues:

- Competencies (how agents acquire the knowledge they need to decide when to support the user)
- Trust (the way it is ensured that user feels comfortable delegating the task to an agent).

Two main approaches are used to build interface agents.

The first one is based on rules collections useful in processing information related to the application in question. Once created, these rules for solving the problem are called by default. The main problem of this approach is unsatisfactory treatment criterion of competence. Approach requires more detailed knowledge and considerable effort from the user. User must recognize the opportunity to use an agent to take the initiative to develop such an agent, to endow the agent with explicit knowledge and maintain as agent rules. Confidence in the ability of the agent to solve the real problem is not taking in consideration because the user has full confidence in his programming skills.

The second approach, known as the "knowledge-based approach" is to endow the interface agent with specific application knowledge plus user preferences. This approach is adopted by most researchers working in the field of AI to build so-called intelligent interfaces. At runtime, the interface agent uses its knowledge to recognize user plans and identify opportunities to help and solve the problem. Both competence and confidence
are real problems in this approach. The competence problem involves a considerable effort by the engineer requiring impressive specific domain knowledge. In this situation all knowledge developed for one agent is rarely re-used for other interface agent. Relating trust users don’t have trust in agent’s sophisticated agents that are totally autonomous. Such an agent would leave the user feeling of loss of control over the situation.

In 1995 Maes distinguishes four different sources (as in figure) in which an agent can improve its competence. First interface agent learns by continuously monitoring user activity for a period of time (e.g. week or month) and the identification of patterns.

A second source for learning is the reaction, directly or indirectly, from the user. Indirect reaction occurs when agent's suggestions remain indifferent to the user and taking its decision, other than agent suggestions. The user can also provide explicit reactions automated actions by the agent. Third, the agent can learn from the examples provided by the user explicitly. Finally, the fourth method is to directly query other agents who support other users in similar problems (and probably gained the most experience). If an agent cannot decide what action should be carried out in a particular context, it may present situation to other agents and ask for recommendations for the problem. As an alternative the user should be able to inform the agent to accept suggestions from one or more specific agents that assist a particular user. On the other hand agent can learn from the experiences of collaboration with other agents who offered suggestions satisfactory.

### 1.4.2 Information agents

Intelligent agents for the Internet are commonly called information agents. Information agent present general agent characteristics (autonomous, computational software entity) that has access to one or more information sources (heterogeneous and geographically distributed), and which pro-actively acquires, mediates, and maintains relevant information on behalf of users or other agents, preferably just-in-time.

An information agent will try to manage one or more of the next requirements:

- **Information acquisition and management** - capable of providing transparent access to one or many different information sources; it retrieves, extracts, analyzes, and filters data, monitors sources, and updates relevant information on behalf of its users or other agents. For this phase intelligent agent can present different scenarios (e.g. advance information retrieval of information or even buying information from specialized agents)

- **Information synthesis and presentation.** The agent is able to fuse heterogeneous data and to provide unified, multi-dimensional views on relevant information to the user.

- **Intelligent user assistance.** The agent can dynamically adapt to changes in user preferences, the information, and network environment as well. It provides
intelligent, interactive assistance for common users supporting their information-based business on the Internet.

Information agents can be classified based on their features

- Cooperative / Non-cooperative – if they cooperate or not with other agents for execution of a task; as presented before different protocols and methods for communication are available as well coordination methods and architectures (this class corresponds to multi – agents)
- Adaptive – can adapt to changes of their environment – network and information level – e.g. the web personal assistant agent – must take in account user preferences as well changes on the access to web resources
- Mobile – able to travel around Internet, autonomously (e.g. dynamic load balancing in large scale networks, reduction of data transfer, migration of small business logic)
- Rational – act (may collaborate) to increase their own benefit, e.g. automated trading agents for e-commerce

Based on definition and their features and classification we can observe few main skills: communication, knowledge, collaboration and low level tasks as described on next figure where corresponding key technologies are presented under each of the different skill types.

**Figure 10 Basic skills of an information agent (Klusch 2001)**

- Communication – agent skill that allow communication with information systems and databases, human users and other agents (even API for middleware platforms)
- Knowledge – representation and processing of ontological knowledge and metadata, profiles and natural language input, translation of data formats
- Collaboration – with other agents for achieving a task (e.g. service brokering, negotiation, collaborative filtering) as well collaboration with human users
Based on these can be observed that in nowadays when we have access to huge information database over Internet, information agent can implement successful tasks as finding, organizing information for us as private persons as well in enterprises.

Personal assistant may help the user’s doing everyday business working with more and more sophisticated Intranet and Internet resources.

1.4.3 Middle agents – Intermediaries

Middle agents are defined differently, e.g. facilitator (Genesereth 1997; Martin, Cheyer, and Moran 1999), intermediaries and brokers (K. Sycara, Pannu, and Williamson 1996) based on the difference they interact with clients/requesters and providers or based on the characteristics of security, robustness, adaptively, etc.. What specific type of middle agent will be used depends on the application requirements. Linking supply and demand is based on agent capabilities. Bidders/service providers are specifying their potential that is available to customers in terms of their capabilities.

This information is provided with additional parameters to specify the conditions of service under which these services are provided, e.g. price and quality of service. Customers who require specific services are sending the parameters that are needed to be taken in account by providers. There are different approaches regarding how to make the connection request to provide:

- Exact match (when both descriptions are equivalent),
- Integration of the supply demand (when the bidder will not require further or more specific information than can provide client and therefore will not return fewer or more general parameters than the client needs)
- Relaxed fit (in this case will not return all the services that could be used immediately by the client, but a subset of records received from the service providers).

For implementation, middle agents are using filters based on different criteria for identifying service providers, e.g. Larks (K. Sycara, Pannu, and Williamson 1996) – correlation context, comparing profiles, similarity, digital signature and correspondence constraints.

(Wong and K. Sycara 2000) identify six dimensions that can be distinguished by middle agents:

1. Who sends the information to the agent middle agent? (e.g. bidders or customers - in this respect the two types of agents complement each other);
2. How much information is sent to the middle-agent? (e.g. capabilities/requests or parameters/preferences)
3. What happens to the information middle-agent received? (e.g. is broadcasted, saved in local database)
4. How is the content of the database used? (e.g. browsed, queried)
5. How much information is specified in a query to the middle-agent?
6. Does the middle-agent intermediate transaction with other agents?

Based on this classification and combining the dimensions can be obtained a large number of different types of middle-agents - (Wong and K. Sycara 2000) present a possibility of 28 agents types.

As example a matchmaker (as in next figure) agent will allow providers to advertise their capabilities (services and parameters) and requesters to send requests. In response to a request a matchmaker returns the contact information of appropriate service providers.

![Figure 11 Matchmaker agent (Wong and K. Sycara 2000)](image1)

The requester agent then chooses a service provider and interacts with it directly. Matchmakers do not intermediate transactions.

Other type of agent is facilitator one (as in next picture). This agent presents next action flow:
1. Provider advertise their capabilities with the facilitator
2. The facilitator keeps the advertised capabilities in its database
3. A requester makes a request with the facilitator for a service provider that can provide a particular service
4. In response to a request for service, the facilitator selects one of the appropriate providers and delegates the service request to it. The provider does the service and returns the result to the facilitator, who then forwards the results to the requester.

![Figure 12 Facilitator agent (Wong and K. Sycara 2000)](image2)
In contrast to the matchmaking agent, facilitators “stay in the loop” throughout the transaction (they intermediate transactions).

Other roles that middle agent play are in negotiation that was presented more detailed in previous chapter and coordination presented as well.

1.4.4 Seller - Buyer and negotiation

Different models where developed to attempt to capture consumer buying behavior (CBB), and these will be detailed latter in chapter 2. All of them share a similar list of six fundamental stages:

1. Need identification
2. Product brokering
3. Merchant Brokering
4. Negotiation
5. Purchase and Delivery
6. Product service and evaluation

In a complex multi agent system will be used different approaches to models the sell and buy actions and seller and buyer agents will work with other agent’s types to achieve their objective.

Seller agent will represent user in the sales process and will offer products and services to buyer (directly or through a market), it answers queries for information about its owner’s product and services, respond to RFPs, and enter into negotiation with buyer agent. A seller agent encapsulate the vendor’s selling strategy (e.g. function for setting the bidding price) and the agent may need the vendor’s approval to complete a trade.

A possible scenario for a seller agent (Badica, Ganzha, and Paprzycki 2007) is that he is supervising the negotiation process and is waiting for finishing of the process. If the negotiation process is finished with success then he must reserve the quantity in the warehouse for a particular period of time. The process can be summarized as:

1. If the buyer confirms the purchase then make the reservation
   a. If reservation is successfully inform the buyer and start the payment and delivery process
   b. Else inform buyer about rejection
2. If buyer agent reject the purchase then release from warehouse the product (cancel reservation)
So, the seller agent is characterized by the negotiation knowledge and not all negotiation processes must have a happy end – are cases when sales process come with no sales at the end.

Buyer agent can tailor the selection and of products to the needs of its user. The agent will locates seller agents (probably using a middle agent) that offer the requested product or service and negotiates with them about price and other terms of sales (e.g. shipping).

Figure 13 Buyer agent – Seller agent interaction (from the viewpoint of seller agent) (Oprea 2002)

In the figure is presented the buyer – seller interaction from the viewpoint of the seller. As can be see it cannot be forecast if an agreement will be made and a possible learning capability on the negotiation phase can increase the chance for closing a sales.

Negotiation process is seen as a distributed search in a space of possible agreements between the agents involved in this process (negotiators). The size and topology of this space is determined by the structure of objects that are negotiated. It may be such that each attribute of the object on which bargaining takes place is associated with a specific value. When a new item is added (or an old one deleted) during the negotiation are actually added (or deleted) new dimensions. The number of points of agreement may therefore increase (or decrease).

Can be deduced minimal negotiation capabilities that an agent need to possess (NR Jennings and Faratin 2001):

- Propose acceptable alternatives in the area of negotiation, and
- Respond to such a proposal indicating whether it is acceptable or not

So, minimum requirement of an agent that is able to negotiate its ability to respond to such proposals.

In this context, a proposal submitted is a bargaining solution (either a complete / partial solution or group of solutions - partial / complete). These different types of proposals are either a single point or a region or a set of points or a set of regions. Generalizing, proposals may be made either independently or based on proposals of other agent’s negotiation history. Minimum requirement for an agent to be able to negotiate is
its ability to express disagreement on unacceptable proposals. If agents can only accept or reject the proposals of others, then negotiation (and especially negotiations on the multi-dimensional values) can be inefficient and time consuming as the one making the proposal has no means to assess why this proposal is unacceptable, even if that agent is close to an agreement, even if the dimensions / directions in the area of negotiation should be modified or not.

To improve the efficiency of the negotiation process, the receiver of a proposal must be able to provide useful responses on the proposals received. These reactions can be as criticism (comments on the parties proposal that the agent agrees or not) or some counter-proposals (an alternative proposal generated in response to a previous proposal).

In such reactions, the tenderer should be in a position to generate a proposal that will have a higher probability of leading to a common solution (if so desired). Considering first the critical reaction we have two reaction types:

- May suggest constraints related to a particular element in the negotiation,
- May indicate acceptance / non-acceptance of certain parts of the proposal (or even the whole proposal).

The second mechanism of the reaction, the counter-proposals consist of a simple motion, in response to earlier proposals, which from the point of view of the initiator is more favorable.

Counter-proposals:

- Extend the original proposal
- Amend parts of the original proposal

Suggestions, criticisms and counter-proposals are just worthless statements of what the agent wants. So their purpose is limited to the structure of tradable items. Being perfectly possible to limit the construction of negotiating at the item level, most current negotiation models are reduced only at this level.

This decreases the technological potential to support negotiation since agents may not:

- Justify the attitude of the negotiation process (an agent may have some reason for a particular taken position, in this case the ability to provide some justification for this attitude can allow your opponent to appreciate more favorable agent's behavior and constraints)
- Persuade another agent to change their attitude (agents must sometimes change the opponent's bargaining space to enable negotiation - arguments seek to identify opportunities for such changes, create new opportunities for change in attitude or modify existing evaluation criteria).
In both cases, negotiators give arguments to support these attitudes (based negotiation argument). In addition to generating proposals, counter-proposals and criticisms negotiator seeks to make proposals more attractive (acceptable) by providing additional meta-information in the form of arguments to support a particular position. Nature and type of arguments can vary widely, the most common being:

- Threats (failure to accept a particular proposal sometimes lead to negative reactions),
- Rewards (acceptance of this proposal sometimes lead to positive side effects),
- Evocations (for some reasons a particular option should be preferred over others).

All techniques discussed so far have been centered on the negotiating proposals. Although this can be achieved using sophisticated methods (game theory and heuristic approaches) they present three important limitations:

- Proposals present singular points in bargaining space,
- The only response can be made to a proposal is a counter-proposal, which itself is another point in space or acceptance or withdrawal,
- Difficult to change the set of elements under negotiation even during its performance (which corresponds with the change of negotiation space by adding new dimensions).

The way the argument finds its place in the negotiation process has been defined in (Carles Sierra and NR Jennings 1998) which defined a simple protocol for commercial negotiation. Using arguments for real agents (as opposed to mere collection of logical statements) means handling complex mental attitude of agent (Parsons, C Sierra, and N Jennings 1998).
Capitolul 2 Active support techniques

2.1 Intelligent Agents and Decision

2.1.1 Human decision process

A decision making process is containing four processes as defined by (Simon 1997):

- Intelligence
- Design
- Choice
- Implementation (added latter)
- Review (added latter)

Intelligence: under this phases decision maker acquires information and develops an understanding of the problem.

Design: user identifies criteria, develops the decision model, and investigates alternatives.

Choice: an alternative is selected and the user acts on the decision during the implementation phase.

Implementation: actions and decisions are inseparable; execution can be understood as a progression towards increasingly small decisions that can be readily implemented.

Review: Decision implemented is evaluated – was a good choice?

On the first two phases information is fundamental and as Simon indicated: information constrains the decision.

Similar decision making process was developed for defense decisions and is called observe, orient, decide act (OODA) loop (G. Phillips-Wren and Jain 2007).

Decisions can be characterized by degree of structure (Turban and Aronson 1998) and we have:

- Structured decision – deterministic with a known solution
- Unstructured decision – continuum with decision with small chance or no agreement on the solution
- Semi-structured decisions – the place where DSS are most effective at providing support in order to assist the final user

On the human decision making process we find reaction to stimulus (as in animal’s gene) but when the decision maker has time to generate a projection of future events in the mind two key phases are distinguished: diagnosis and look-ahead (Pomerol and Adam
We can observe that expectations (E) and possible actions (A) can be influenced at any time by preferences (P) but also we can have the situation when preferences (P) can be influenced by expectations.

Figure 14 The decision process (Pomerol and Adam 2008)

Approach is simplified and can appear that the process in linear but many backtracks can occur – e.g. attainable future states are not satisfactory. Diagnosis phase allow recognizing the current state of the world e.g. past and present, and in next phase decision maker must anticipate the possible decisions and the consequence of them based on the personal perception of the future that is the projection phase.

2.1.2 Agents and Decision Support Systems

Different models and architectures of IDSS implemented with intelligent agents where proposed (Boldur Barbat 2002). In next figure a system is illustrated and we can find database (e.g. values of the current state), knowledge base (e.g. guidance for selecting alternatives) and model base (e.g. formal models of the decision problem – algorithms and methodologies) as inputs, one or all of them can use intelligent agent’s methods (G. Phillips-Wren 2006).

Figure 15 Intelligent decision support system based on intelligent agents (G. E. Phillips-Wren 2008)
On the processing different action may take place:

- Status reports (e.g. different KPI – course of actions, measure of performance),
- Forecasts (e.g. simulation and resulting projected values),
- Recommendations (e.g. best values that can be used in simulation) and
- Explanations (e.g. justify the recommendations based on user preferences)

Input feedback from the processing can be provided dynamically to update the models and inputs in real time with external interventions and provides additional data, knowledge and models that can be useful for future.

Output feedback is used to extend or revise the original analyses and evaluations, and intelligent agents could interact with the user to refine or evaluate the decision further.

At the end of process user maintains the final decision – accepting or not the recommended action plan and his implementation; also user provide to the system user specific domain knowledge and preferences.

In next table some of the characteristics of intelligent agents / multi-agents systems are related to the decisions steps and it show the actions that can be performed by agents on behalf of the user (or other agents) adapted from (Filip F. 2007).

<table>
<thead>
<tr>
<th>Decision Process</th>
<th>Decision Steps</th>
<th>Autonomous</th>
<th>Adaptive</th>
<th>Proactive</th>
<th>Reactive</th>
<th>Communicate</th>
<th>Cooperative</th>
<th>Mobile</th>
<th>Goal-Driven</th>
<th>Persistent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intelligence</td>
<td>Problem Detection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Data Gathering</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem Formulation</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>Model Classification</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model Building</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model Validation</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choice</td>
<td>Evaluation</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sensitivity Analysis</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>Results Presentation</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task Planning</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task Monitoring</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Table 3 Decision process steps vs agent’s characteristics
As indicated at beginning of this chapter on the first two steps on the decisional process (intelligence and design) information play a central role and information constrains decision.

Similarities in the decisional process together with agent’s characteristics it show that agents are able to play a central role in locating desire information for the user. Intelligent agents in the information retrieval process are autonomous, proactive, communicative, mobile, goal-driven and persistent.

<table>
<thead>
<tr>
<th>Decision Making Process</th>
<th>Description</th>
<th>Information Search Process</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intelligence</strong></td>
<td>Recognize problem; Gain problem understanding; Seek and acquire information</td>
<td>Task Initiation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Topic Selection</td>
</tr>
<tr>
<td><strong>Design</strong></td>
<td>Develop criteria; Specify relationships; Explore alternatives</td>
<td>Pre-focus Exploration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Focus Formulation</td>
</tr>
<tr>
<td><strong>Choice</strong></td>
<td>Evaluate alternatives; Develop recommendations; Make decision</td>
<td>Information Collection</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Weigh consequences; Implement decision</td>
<td>Search Closure</td>
</tr>
</tbody>
</table>

Table 4 Steps in the decision making process compared to the information search process (Wang, Y. 2006)

### 2.1.3 Agent approach on e-Commerce

On literature are defined a variety of descriptive theories and models that attempt to capture buying behavior, such as the Nicosia model, the Howard-Sheth model, the Engel-Blackwell model, the Bettman information-processing model, and the Andreasen model (Maes, Gutmann, and G. 1999).

These models refer a similar list of six fundamental stages of the buying process and based on these stages the place of agent technologies that can apply to the shopping experience is find.

- **Identification**: characterizes the buyer becoming aware of the need by through product information. Agents can play an important role for those purchases that are repetitive (supplies) or predictable (habits). E.g. “monitors”: continuously monitor a set of sensors or data streams and take action when a certain pre-specified condition applies. Nowadays we have “notification agent” for example on the “Amazon.com” which monitors the catalog of books for sale and notifies the customer when certain events occur that may be of interest to the customer.
**Brokering**
- **Product Brokering**: once buyer has identified a need to make a purchase (possibly with the assistance of a monitor agent), the buyer has to determine what to buy through a critical evaluation of retrieved product information. The result of this stage is a consideration set of goods.
- **Merchant Brokering**: this stage combines the consideration set from the previous stage with merchant-specific alternatives to help determine who to buy from. The merchants do not want to compete on price only, and want the value-added services (e.g., warranty, availability, delivery time, reputation) to be included in consumers’ buying decision.

**Negotiation**: price and other terms of the transaction are settled on. The majority of business-to-business transactions involves negotiation but in retail is mostly familiar with fixed prices.

**Payment and Delivery**: this stage can either signal the termination of the negotiation stage or occur sometimes afterwards (in either order); the available payment or delivery options can influence product and merchant brokering.

**Product Service and Evaluation**: post-purchase stage involves product service, customer service, and an evaluation of the satisfaction of the overall buying experience and decision.

P. Maes (Maes, Gutmann, and G. 1999) main concern was design of an infrastructure that handle information overload, recent research show that using intelligent agents facilitate higher quality information, personalized recommendation, decision support, knowledge discovery, etc.

Now days modern agent environment (as JADE) permit implementation of realistic e-commerce scenarios and research in auction theory allow to implement automatic negotiation phase (Kowalczyk, Ulieru, and Unland 2002).

Different agent or multi agent based systems where proposed and tested in literature and each ones are designed to respond to one of the topics like: negotiation, recommendation, classification, etc.

An example of a multi-agent system (Badica, Ganzha, and Paprzycki 2007) that is composed by 1) *information center* - product catalogs (white pages, yellow pages, etc.) are handled 2) *purchasing* – user-client activities agents are present and 3) *seller* – agents that support the user-client for seller.
In presented system are defined different agents for each section and all of them are communicating to achieve the user scope. Each agent can implement a different algorithms and process flow depending on implementation. For a detailed description of each agent (Badica, Ganzha, and Paprzycki 2007).

**Client Agent** – it responding to orders that comes from user-client and he is using: queries to the CIC agent; dispatches or request creation of buyer agent for each store identified by gatekeeper agent.

**Buyer Agent** – is the agent in the presented architecture that may involve mobility. This agent may be activated by client agent or gatekeeper agent and contain the knowledge about negotiation with the seller. Negotiation protocol that must be used for current instance is obtained from gatekeeper agent.

**Shop agent** – it is the agent representative of the user-seller with main focus on supervise the negotiation process and product flow; after creation agent will persist in the system until user-seller will decide that. This agent work tide with agents: gatekeeper, warehouse and seller.

**Gatekeeper agent** – play the role of middle agent that keep records for buy intentions and sell intentions (agents that register to the gatekeeper with buy / sell intention) and based on his one algorithm decide when negotiation can start based on selected template and protocol.

**Warehouse agent** – keep the knowledge about quantities its focus is on: notification from shop agent and act on them (product reservation, purchase confirmation, purchase termination)

Seller agent – main focus of this agent is to negotiate and to sell the product – not all negotiation must be with success and his success rate depend on the negotiation strategy (moreover algorithms and parameters)
Different attempts exist on the research literature for developing agent-based system adapted to e-commerce but almost all are small-scale demonstrators. In other case applications are utilizing the agent metaphor and are not using a specific agent tool and environment.

### 2.2 Information retrieval elements

When, at the beginning of 1950’s, Calvin Moers, one of the information science pioneers, coined the term information retrieval (IR) he also defined the problems addressed by the activity:

1. How to represent and organize information intellectually?
2. How to specify a search intellectually?
3. What systems and techniques to use for those processes?

The problem underlying all of theoretical, experimental, and empirical activities in user modeling revolves around the classic and most difficult question: *What it is important to know about the user in order to support the user in interaction with the IR system?* Accordingly to Maglio and Barret (Maglio and Barret 1997), developing an explicit model of a user’s information need addresses the following issues:

1. What kind of support should this model give?
   a. Improving precision (the system can add other terms in the query from the user to cover the context of its meaning). This can improve the percentage of relevant retrieved documents.
   b. Improving information need coverage. The concepts conveyed by a user query express a vague information need. Expanding this concept will make it more likely that every aspects of this information need are captured.
   c. Pointing the user to relevant information. The system may expand and search for these expansions autonomously.

2. What aspects of an information need should be represented? A distinction can be made between topics of interest and situational factors. The first term refers to the concepts which are part of the information need. The latter provides a context for specific information need, for instance, the type of knowledge requested or the background knowledge of the user.

3. How to infer aspects of an information need? If not provided directly by the user, these aspects should be estimated from other sources. Some considered sources of information are:
   a. Document read by the user. The system can monitor the user’s browsing through the World-Wide Web. Acceptance of a document by the user can
be measured explicitly, by an option in the interface, or implicitly by measuring reading time.

b. Clicking behavior. This can be used, for instance, to estimate the user’s browsing strategy or reading capacity.

4. How to deal with the ambiguity of the actions of the user in regard to the estimation of the information need? Because of this ambiguity, the system should have a way to deal with conflicting hypotheses. As possible formalism, fuzzy logic and Bayesian networks can be considered.

### 2.2.1 Traditional vector space model

The basic idea of the model is that everything - documents, terms, queries - can be represented as a collection of vectors in a multi-dimensional space. In this model a document is a collection of terms that are associated with vector. In turn, a query (which she is also a collection of terms) is associated with another vector. By using linear algebra to discover which documents are most suitable to be retrieved with a given query (G. Salton, A. Wong, and C.S. Yang 1975; Raghavan, V.V. and Wong 1986)

A vector space is characterized by a set of basis vectors linearly independent. Each of them corresponds to a direction vector or a dimension of the space in question. For example, the two-dimensional space having two base vectors usually record the $x$ and $y$ as in next figure.

![Figure 17 placing a document in the space defined by the "Active" and "Independent"](image)

One possibility is to represent each term as a linear combination of base vectors. For example, for a two-dimensional space (active and independent), the words cat and dog might be as follows:

<table>
<thead>
<tr>
<th>Term</th>
<th>Active</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td>Cat</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td>Gravity</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 5 Values of the terms in vector space
A document will be composed of vector sum of its terms. In the previous example document would be composed of "dog, cat" (Fig. 17). Some terms, however, cannot be represented in such an area, for example, gravity term.

Under these conditions where opted for solution: the terms of the documents are used as basis vectors and solution space defined by them is increasing the vocabulary.

Let assume D1 and D2 are two documents that the words cat, dog and lion. The D1 they appear to 3, 1 and 4 times, and in document D2 appear to 8, 2 to 6 times. We query the word dog appears with weight 2. Try to determine what would be the query result on the two documents.

In the first case considered that weightings relationship words (animal species) and in the second case we consider the words taken by it, orthogonal.

<table>
<thead>
<tr>
<th>Term</th>
<th>Cat</th>
<th>Doc</th>
<th>Lion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>1.0</td>
<td>-0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Dog</td>
<td>-0.2</td>
<td>1.0</td>
<td>-0.4</td>
</tr>
<tr>
<td>Lion</td>
<td>0.5</td>
<td>-0.4</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Cat</th>
<th>Doc</th>
<th>Lion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>1.0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>Dog</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Lion</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 6 Weight vs Orthogonal values of the terms in vector space

Computing the similarity between document and query we get for each case:

a) \[ \text{Sim}(D1, Q) = (3T1 + 1T2 + 4T3)(2T2) = 6T1T2 + 2T2T2 + 8T3T2 = -2.4 \]
   \[ \text{Sim}(D2, Q) = (8T1+2T2+6T3)(2T2) = 16T1T2 + 4T2T2+12T3T2 = -4 \]

b) \[ \text{Sim}(D1,Q) = 0 + 2 + 0 = 2 \]
   \[ \text{Sim}(D2, Q) = 0 + 4 + 0 = 4 \]

Coefficients (elements vector, terms weights) encapsulates the presence or importance of the term. This model provides a way to set these weights. In theory there are several methods of information retrieval have set. The most common are:

- Binary weights, specify if the term is present (the value 1) or not (through 0) in the vector
- Use tf - Terms frequency
- Use tf * idf ; where idf term indicates the power of discrimination.

Depending on the calculation of the weighting vectors are a few common similarity measurements:
The most commonly used measure is the scalar product. A small example illustrates implementation. If you have a document consisting of terms D1 T1, T2 and T3 with weights (0.5, 0.8, 0.3) and a query Q with the same terms and weights (1.5, 1, 0) the similarity between D and Q is:

$$\text{Sim}(D, Q) = \frac{(0.5 \times 1.5 + 0.8 \times 1) + (0.8 \times 1.0)}{(0.5^2 + 0.8^2 + 0.3^2) \cdot (1.5^2 + 1.0^2)} = 0.868$$

### Table 7 Similarity functions

<table>
<thead>
<tr>
<th>Sim(X,Y)</th>
<th>Binary</th>
<th>Weighted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar product</td>
<td>$</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Dice coefficient</td>
<td>$\frac{2 \cdot</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Scalar product</td>
<td>$\frac{</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Jaccard coefficient</td>
<td>$\frac{</td>
<td>X \cap Y</td>
</tr>
</tbody>
</table>

### 2.2.2 Weights

Within each term in appears in the vector space has an associated weight that reflects the relevance in the document (define the semantics of the document). The most common model is $tf \times idf$. The weight is calculated as the product of two factors. The first factor $tf$ is the term frequency in the document being a local measure and $idf$ term reflects the strength of discrimination between two documents in which it appears and is a global measure.

For the calculation of the frequency term is commonly used in two configurations:

$$K + (1 - K) \cdot \frac{Freq_{i,j}}{MaxFreq_j} \quad \frac{Log_2(Freq_{i,j} + 1)}{Log_2(Length_j + 1)}$$

Where,

- K –value between 0 and 1 (example 0.5)
- $Freq_{i,j}$ – frequency of term i in document j
- $MaxFreq_j$ – frequency of the most relevant term in document
- $Length_j$ – length of document j
The *idf* factor helps to differentiate the documents; the more a term occurs in fewer documents the power of distinction between them is greater.

\[
idf = \log_2 \left( \frac{N}{N_t} \right) + 1,
\]

where \( N \) total number of documents from the collection and \( N_t \) number of documents where terms \( t \) appear.

It is immediately clear that the equation for the *idf* requires that the entire set of documents in the collection to be known in advance. Even if it is possible in some cases, it certainly is impossible to fulfill in documents available on the web. Even though researchers have tried to estimate *idf* values through statistical means it is difficult to have a documented high enough to provide a good estimate of the *idf*. And the harder it is to collect pages from all areas (subjects) available on the Web.

Thus *idf* cannot or at least not suitable to be used in the context of the web. It can however be substituted with another measure: taking into account the fact that the *idf* is used to vary the weights terms according to their power of discrimination between documents it can be replaced with information extracted directly from the document, analyzing underlining text.

It is assumed that the person who created the page will highlight the terms that are most important. These terminals could appear in the title, the subtitle, be underlined or marked by thickening (Scime 2005).

### 2.2.3 Representation of documents in the vector space

Documents are represented in vector space as a term-document matrix \( t \times d \) the columns represent documents and each document a specific location within the corresponding weight. Before being used documents are processed. Common words, so-called stop words are removed and the remaining is reduced to their root by removing suffixes. Many algorithms are available, the best known being those defined by Porter (Porter 1980).

The results obtained are good but it is limited to English language – for best results. Alternatively can be used the lookup dictionary. Resulting terms are used further in the analysis. Since the total number of terms in the document collection can easily exceed several hundred thousand, and the number of documents hundreds of millions of this representation is not economic or efficient in terms of computational power and memory space requirements. For a better approach of this problem is using latent semantic indexing (LSI) (Berry, Drmac, and Jessup 1999).

LSI is based on principle that words are used in same contexts can have similar meanings. The reduction is done by converting the matrix into a diagonal matrix by
Singular Value Decomposition algorithm (or later performance, Singular Value Decomposition Riemannian R-SVD). Detailed description of the mathematical background involved and the method is found in (Jiang and Berry 1998; Jiang 1998). What is essential is that R-SVD usually retain a small number of terms (around 100) both relevant documents and the query. Similarities will be calculated into the new space and will be used for the rest.

An alternative to this representation is the so-called Inverted Files used when the document collection is relatively small. Such a file is a dictionary of terms in which each term is stored characteristics of (total frequency, \( idf \) value for syntactic category) and a list of documents in which it appears. In a simple example (random values):

<table>
<thead>
<tr>
<th>Term</th>
<th>Total Freq</th>
<th>IDF</th>
<th>Pointers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>7</td>
<td>1.3</td>
<td>Doc1,doc2,doc5,doc9</td>
</tr>
<tr>
<td>Sleep</td>
<td>3</td>
<td>2.4</td>
<td>Doc2,doc7</td>
</tr>
</tbody>
</table>

Table 8 Inverted files - example

Pointers do not indicate the original source documents but another dictionary of terms associated with each document. In the previous example document doc2 dictionary could be.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Freq</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>2</td>
<td>0.02</td>
</tr>
<tr>
<td>Sleep</td>
<td>1</td>
<td>0.014</td>
</tr>
</tbody>
</table>

Table 9 Dictionary for inverted files

Frequency can be stored as raw value or value weighted (or both). The first way is more flexible because it allows us to dynamically decide what type of weight we want to calculate and use. The downside to running it needs more computing power (same calculations are carried out several times).

2.2.4 Query refining / envelop

Query just approximates required information. It was found that users usually start with short queries that provide a poor approximation of the required information. Once people notice results refine their query by adding new terms, by deleting existing in their reweighting and adding logical operators (AND, OR, NOT). The question that immediately arises is whether this process can be automated by analyzing relevant documents and irrelevant. Using a vector space, refining can be seen as in next picture:
In the first case the query was located close to irrelevant documents (with the direct consequence that they will be returned to the first position). After refining the query to its nearest to the relevant documents leading to a correct retrieval. (The relevant documents are returned to the first position in response). A wrong answer is affects the final user that should consult numerous documents irrelevant to it and leading to a loss of time and, in some cases, even loses interest in the information. The main aim of refining the query is moving closer to the relevant documents. In the vector space query can be refined by adding terms in the relevant documents and the removal of terms that appear in the irrelevant.

$$Q' = Q + \alpha \cdot \frac{1}{|R|} \sum_{D_j \in R} D_j - \beta \cdot \frac{1}{|NR|} \sum_{D_j \in NR} D_j$$

The new query will consist of the old plus the terms of the relevant documents and remove terms from the irrelevant. Factors $\alpha$ and $\beta$ are constants representing the importance given to positive and negative feedback. This method knows different variations depending on how the weighting factors are chosen, vector length (number of terms add query) but the basic idea remains the same. A small example illustrates the concept:

Query Q is identified by the vector $(5,0,3,0,1)$. The relevant document D1 is coded as vector $(2,1,2,0,0)$ and the document D2, irrelevant as $(1,0,0,0,2)$. If we choose $\alpha = 0.50$ and $\beta = 0.25$ new query will be

$$Q' = Q + 0.50*D1 - 0.25*D2 = (5,0,3,0,1) + 0.5(2,1,2,0,0) - 0.25(1,0,0,0,2) = (5.75, 0.5, 4, 0, 0.5)$$

As a result of refining the new query term appears in the second and increases the importance of the term 3 (due to the relevant document) and weight loss for the term 5 through negative feedback of irrelevant documents.

There are various ways to choose terms that will be kept. Some authors opines to keep all the terms in the new query, others opt only for the community or for those that
have the highest weights. The experiments found that the best method is the choice of all the terms, but it gives good results and keeping only the relevant terms. Choosing the best-weighted terms do not provide great results but there are many who dispute the result so that the method is still widely used.

An immediate interpretation is that refining query by adding new terms specified there and better context in which the results are to be obtained (usually obtained context of a user profile).

2.3 User profiling

With the progress in technology and Internet (Intranet) networks multiple sources of information became available but appeared more delicate problems. Using existing search engines (for Internet or Intranet) can relatively easily get relevant information. However, reaching the correct document is still a time consuming task. Given a list of documents that are relevant surface a large amount of time is lost to check and browse before the truly relevant information to be found. Moreover the search is diverted occasionally find other interesting information for users who were not part of the initial target. It increases search time before returning to the old target.

Users can use software agents to explore the sources of information and provide only the most relevant documents. A system of agents must maintain profiles of users who submit their current interests. Based on these profile documents found on user demand are filtered, sorted and recommended.

In addition, agents can retrieve information and automatically by setting alarms. An alarm will be a search request that the agent will perform at regular intervals. Retrieved documents are analyzed and if it finds that changes have occurred to the previous results the user will receive an email with them. Agents must adapt dynamically to user interests.

They can learn through direct or indirect feedback changes of interest and needs. Through this direct feedback can specify whether the documents found are suitable or not. The agent learns indirectly by observing user reactions and actions while using the system.

User modeling is one of the essential components of any intelligent information retrieval and filtering. The model should represent the interests of preserving the current user and the day will be the main task. It must be dynamic and adapt to changing information needs and interests of users.
2.3.1 Structure of the user profile, data filtering and profile quality

The basic element of a category of interest is a vector of features. Vector consists of a list of words weighted by their level of importance within the category. Simple approach is: interest category \( C \) consists of three descriptors: a positive descriptor \( DPC \); a negative \( DNC \) and one for long-term behavior \( DTLC \). The figure illustrates the portion of the profile for information retrieval and filtering.

![Profile diagram]

Figure 19 User profile representation using three descriptors categories of interest

Positive and negative descriptors maintain a vector of features learned from the documents by the positive reaction or negative ones. The long term descriptor encodes a vector of features obtained through both types of reactions. Each descriptor has an associated weight to indicate its degree of interest in the category \((w_p^c, w_n^c, w_l^c)\).

Values of \( w_p^c \) and \( w_n^c \) are between 0 and 1 and represents the degree of interest or uninteresting. The value of \( w_l^c \) is between -1 and 1. A negative value indicates that the descriptor term is not interesting area of interest while a positive value indicates that it is interesting. In addition to a long descriptor \( dCount \) counter stores the total number of documents observed for a given area of interest. An additional status indicator is used in reactions involved. \( Flagc \) indicator indicates whether at least one item of interest from category \( C \) was read during a session. The flag is set if at least one submitted document has been read by the user. Formally an interest category \( C \) can be represented as

\[
Categoria_c = \left\{ flag^c, (w_p^c, d_p^c), (w_n^c, d_n^c), (dCount^c, w_l^c, d_l^c) \right\}
\]

Users can have multiple interest categories. As a result, a user profile \( P \) having \( m \) groups of interest will be represented as:

\[
profil_P = \{ Categoria_1^p, Categoria_2^p, ..., Categoria_m^p \}
\]

Profile thus defined will be used to filter the information and operations for refining queries.
Based on the previously defined profile structure the filtering process is used to select the relevant documents in a set of documents. In the specific case of web search will be the set of documents returned by multiple search engines. For each document in the set level of interest of its content is calculated by matching the profile a category of interest and the interest of the class. Document relevance is computed as a numerical value which is then assigned and the score of the document will be considered relative to their profile. Documents will be reordered after these scores the most significant “n” documents will be recommended to the user.

Document score is a value between -1 and 1. A positive value indicates that it provides a degree of interest, and a negative one means uninteresting to document. Given a document with characteristic vector $f_{vd}$ document score after profile P is calculated according to the following algorithm. The characteristic vector $f_{vd}$ of the document is calculated according to one of the methods described in the previous sections.

1. Category C is relevant to the document D. The method of finding will be presented a little later

2. Calculate the score for each descriptor category C with the highest relevance:

   \[
   w_{pos} = w_p^c \cdot Sim(d_p^c, f_{v_d}) \\
   w_{neg} = w_n^c \cdot Sim(d_n^c, f_{v_d}) \\
   w_{long} = w_h^c \cdot Sim(d_h^c, f_{v_d})
   \]

3. Final score is calculated by the formula

   \[
   Score(P, f_{v_d}) = \max(w_{long}, w_{pos}) + \min(w_{long}, -w_{neg})
   \]

Each category has both positive and negative interests and the descriptors of the two interest vectors have opposite meanings, the final score of the document will be composed of positive interest score $w_{pos}$ and the negative score $w_{neg}$. The long-term interest $w_{long}$ contribute to both interest depending on the sign of its value.

The quality of a user profile is a key factor in the system and the user's point of view is the precision. If a large proportion of recommended documents are irrelevant to the user then the system it will confuse more than help. Also, if the system fails to provide the required amount of user relevant information then we have a so-called recall problem; then the user will have to actively look for the information that he requested. Both problems are reflected by unhappy users that will ultimately lead to an unusable system.
The recall and precision can be defined as:

\[
\text{precision} = \frac{n_{\text{Rel}}}{\text{total \_retrieved}} \]
\[
\text{recall} = \frac{n_{\text{Rel}}}{\text{total \_rel}}
\]

Where \( n_{\text{Rel}} \) is the total number of relevant documents retrieved, \( \text{total \_rel} \) is the total number of relevant documents available in the collection and \( \text{total \_retrieved} \) is the total number of documents retrieved.

Due to the nature of the web cannot know the total number of relevant documents in the set because we do not know the set in its entirety. The only way to measure the two quantities is through simulation. Based on a defined set of documents are calculated the two values (precision and recall) – using a standard algorithm e.g. used by search engine. Based on same pool of documents search is again realized but now also the profile is taking in account, and the two values are computed.

By comparing the values obtained can draw conclusions relative to the quality profile.

### 2.3.2  Learning user profile - explicit and implicit feedback

A profile is composed by a set of interest categories and represents different domain interest. Characteristic vector of the three descriptors in a category of interest (positive, negative and long-term) represent the area of interest. The interest level of a category is encoded as weights for each descriptor. Because descriptors represent the same interest group category is a balance between the positive and negative level of interest. Increasing the level of interest for positive descriptor can lead to reduced interest descriptor negative and vice versa. In addition, long-term change descriptor may be different from the descriptors in the short term due to the different nature of the long-term interests and short-term. Changing term interests tend to be reactive to accommodate the changing interest for a short period, while long-term interest changes gradually over a range of time.

The algorithm that modifies the interest category \( C \) based on the explicit feedback is:

1. Update of the \( w_i^c \) for long term descriptor. To allow unlimited gradual changes and capture a reluctance of the long-term interest’s to change after learning in the run; bipolar sigmoid function is considered to govern the change of long-term descriptor interests. The function has values in the range -1 to 1 and the input \(- \infty \ldots + \infty\). The figure illustrates the use of the tool to change the descriptor weight depending on the rate of learning.
The nature of changing long term interests (Widyantoro 1999)

The current value $w_{lt(old)c}$ (the ordinate) is projected on the abscissa. The learning rate $\alpha$ is added to the value on the abscissa for a positive response or is reduced for a negative one. The new value is then projected on the ordinate as the new value of the $w_{lt(new)c}$. Descriptor term weight change is implemented with the following equation:

$$w_{lt(new)c}^c = f(f^{-1}(w_{lt(old)c}^c) + \alpha) \quad \text{for a positive reaction}$$

$$w_{lt(new)c}^c = f(f^{-1}(w_{lt(old)c}^c) - \alpha) \quad \text{for a negative reaction}$$

Where $f(x)$ is the bi polar sigmoid function:

$$f(x) = \frac{2}{1 + e^{-\beta x}} - 1$$

2. Update the long-term descriptor $dltc$ with the learned document feature vector using the equations above. Set the value of the learning rate $\alpha$ on above equation is $\gamma$:

$$\gamma = \frac{1}{d\text{Count}} + \kappa$$

As the learning process progress the importance of the most recent documents will decrease in order to protect the interest learned previously. The constant $k$ is set to a small value (e.g. 0.5) to prevent a stoppage of learning, since $\alpha$ would converge to 0 in the limit (after learning many documents) and it allow the long term descriptor to keep learning regardless the number of previously learned examples.

Increase the number of documents $d\text{Count}$ in the long-term descriptor by 1

3. Update the positive and negative descriptors using positive and negative feedback using above equations.

4. Update descriptor level of interest for positive (or negative). Depending on the learning rate of interest associated descriptor can grow from the current value up to 1; it supports a reactive learning of short-term interests.

$$w_{p(new)c}^c = w_{p(old)c}^c + (1 - w_{p(old)c}^c) \cdot \alpha \quad \text{for a positive reaction}$$
$w_n^{(new)} = w_n^{(old)} + (1 - w_n^{(old)}) \cdot \alpha$ for a negative reaction

5. Compute the similarity of the learned document feature vector $fvd$ with the negative descriptor for positive feedback or the positive descriptor for negative feedback, and reduce proportionally the interest weight of the descriptor according to the similarity between the two vectors. The scheme guarantees that no two opposite descriptors (positive and negative) containing similar feature vector has close interest weight values. For this update using formulas:

$$w_n^{(new)} = w_n^{(old)} \cdot (1 - \alpha \cdot \text{sim}(d_n^c, f_v))$$ for a positive reaction

$$w_p^{(new)} = w_p^{(old)} \cdot (1 - \alpha \cdot \text{sim}(d_p^c, f_v))$$ for a negative reaction.

Depending on the type of reaction (positive or negative) all the steps (except step 2) will be modified accordingly. In step 1, the level of long term interest descriptor for a positive reaction will increase and decrease for a negative reaction. In step 2 the contribution of the document vector will depend on the number of documents that have already been observed in the category. The value of the learning rate in steps 4 and 5 will depend on the user feedback. A strong feedback will have a high $\alpha$ value and vice versa. In step 6, a positive feedback will reduce the interest level of negative descriptor. Conversely, the interest level of positive descriptor will be reduced due to a negative feedback.

Unlike explicit feedback that can result in significant changes of the descriptors and their interest level, implicit feedback affects them very little. A positive feedback is obtained when the user opens and reads one of the documents presented for a longer period of time; the impact is connected to the period that user is reading the document (longer time, bigger impact). Decreasing the level of interest associated with negative descriptor caused by the implicit positive feedback will be proportional to the similarity of the opened document with the negative descriptor. However, because the confidence in this feedback is low, the reduction of interest is limited to a narrow range. Steps to modify the user profile on implicit positive feedback:

1. Apply learning explicit feedback described using a low learning rate $\alpha$

2. If flag = 0 set the value of flag to one

   If for a time no article for a particular interest category has been read, a penalty will be applied to the corresponding interest category by calling an implicit negative feedback. Next steps are needed to modify the user profile on implicit negative feedback:

   $$w_p^{(new)} = \lambda \cdot w_p^{(old)} \text{ if } flag = 0$$

   $$w_p^{(new)} = w_p^{(old)} \text{ else}$$
\[
\begin{align*}
    w_{n(new)}^c &= \lambda \cdot w_{n(old)}^c & \text{if flag} = 0 \\
    w_{n(new)}^t &= w_{n(old)}^t & \text{else}
\end{align*}
\]

\[
\begin{align*}
    w_{l(new)}^c &= \lambda \cdot w_{l(old)}^c & \text{if flag} = 0 \\
    w_{l(new)}^t &= w_{l(old)}^t & \text{else}
\end{align*}
\]

Where the value of \( \lambda \) is close to 1 (e.g. 0.97). If the flag =1 then set flag = 0

The penalty results in the reduction of the tree descriptors will fall near zero and at some point the corresponding interest category will be removed from the profile.

### 2.3.3 Learn new interest

If a feature vector of the learned document cannot be classified into a category in the profile a new category will be added to a user profile. To determine that a document’s content belongs to a particular category, a threshold value \( \theta \) is defined. If the similarity between a learned document and the descriptor of the greatest category’s relevance in a profile is less that \( \theta \), a new category representing the new interest will be created.

Let \( fvd \) be the feature vector of the document to be learned and \( \alpha \) be the learning rate. The category is created by:

1. Create a new category – empty
2. Set the long-term descriptor to the feature vector of the learned document and set the interest level of the descriptor using equation described above where the interest level of the long-term descriptor \( w_{l(old)} \) is set initially to zero, then results:
   \[
   \begin{align*}
   d_h^c &= f_{v_j} \cdot w_h^c & \text{for positive feedback} \\
   d_h^t &= f_{v_j} \cdot w_h^t & \text{for negative feedback}
   \end{align*}
   \]
   where \( f(x) \) is the bipolar sigmoid function
3. Set the negative descriptor for positive feedback or the negative descriptor for negative feedback to the feature vector of the learned document and set the interest level of the corresponding descriptor. The initial value of \( w_{p(old)}^c \) and \( w_{n(old)}^t \) are set to zero.
   \[
   \begin{align*}
   d_p^c &= f_{v_j} \cdot w_p^c = \alpha & \text{for positive feedback} \\
   d_n^c &= f_{v_j} \cdot w_n^c = \alpha & \text{for negative feedback}
   \end{align*}
   \]
   Set the negative /positive descriptor for positive / negative feedback to an empty vector and set the interest level of the corresponding descriptor to zero
\[ d^+_n = \{ \} \quad \text{si} \quad w^+_n = 0 \quad \text{for positive feedback} \]
\[ d^-_n = \{ \} \quad \text{si} \quad w^-_n = 0 \quad \text{for negative feedback} \]
So, the long-term descriptor is assigned with the learned documents feature vector of both positive and negative feedback. The assignment for the positive and negative descriptor is distinguished according the type of feedback. On a positive feedback, the positive descriptor is set to the feature vector of the learned document and the negative descriptor is left empty. The positive descriptor, similarly, is left empty and the negative descriptor is assigned with the feature vector of the learned document on a negative feedback.
Capitolul 3  Search engine agents - active decision support system

3.1 Introduction

Based on the assumption from beginning of chapter 2 to undertake all kinds of such questions an implementation should support the following issues:

1. Assist the user in the diagnosis process and question reformulation;
2. Select appropriate search engine for efficient searching accordingly to their profiles;
3. Translate the question into one or more queries and search strategies acceptable to the given search engine;
4. Manage searching strategy;
5. Support the user in the results assessment;
6. Support the user in resource description
7. Provide the user with the appropriate outputs in a suitable structure; and, way not,
8. Advice he or she in the follow-up activity.

Over the last years, there has been increasing interest in intelligent agents, distributed artificial intelligence and distributed systems as presented in previous chapters. Links this with the increasing focus on IR systems and co-operative work patterns, raises the issues of how these "distributed cognition" capabilities can be integrated to create intelligent tools.

The MAS (Multi Agent Systems) paradigm represents one of the most promising approaches to build complex and flexible new architectures required for next generation of intelligent tools offering a new dimension for large-scale integration. MAS are software systems composed of several autonomous software agents running in a distributed environment. Beside the local goals of each agent, global objectives are established committing all or some group of agents to their completion. Some advantage of this approaches are: it is a natural way for controlling the complexity of large, highly distributed systems; it allows the construction of scalable systems since the addition of more agents is an easy task; MAS are potentially more robust and fault-tolerant than centralized systems.

An important role for agents may be the delegation of tasks. Agents interact and negotiate with each other to determine a suitable contracting agent. The contract net model (Maturana and Norrie 1997) provides a suitable general protocol to design and implement this negotiation process.
The MAS provides a platform for co-ordination and co-operation, within which its agents can work collectively to solve specific problems. Clusters or teams of agents are identified (Carley and Lin 1995) to perform specific reasoning for a given task and decision-making responsibilities are delegated to co-ordination groups made up of these agents. After all, if we don't expect people to be omnipotent, why should we expect this from agents?

3.2 Agent based IR system

Following Croft’s Top Ten List (Croft 1995), of the most significant questions facing current IR systems, Finin, Nicholas and Mayfield (T. Finin, Nicholas, and Mayfield 1998) identified the agents features that are able to accomplish each particular issue as in next table.

| Relevance Feedback | Information Extraction | Multimedia Retrieval | Effective Retrieval Routing & Filtering Interfaces & Browsing Term Expansion Efficiency & Flexibility Distributed IR Integrated solutions |
|--------------------|------------------------|----------------------|-------------------------------------------------|-------------------------------------------------|-------------------------------------------------|
| ✓                  | ✓                      | ✓                    | ✓                                               | ✓                                               | ✓                                               |
| ✓                  | ✓                      | ✓                    | ✓                                               | ✓                                               | ✓                                               |
| ✓                  | ✓                      | ✓                    | ✓                                               | ✓                                               | ✓                                               |

Table 10 IR and agent characteristics - adapted from (T. Finin, Nicholas, and Mayfield 1998)

Consequently, the multi-agent system paradigm represents one of the most promising approaches to build complex and flexible architectures, offering a new dimension for large-scale integration. Below will be briefly outlined some of the well-known types of agent-based IR systems.

- **Knowbot agent-based IR Systems.** A ‘knowbot’ is an agent-based IR system that provides a single query language to access a variety of information sources. It serves as a representative for the user (which demands program autonomy). Some prototypical examples from this category are MetaCrawler, SavvySearch and NetbotJango.
• **Adaptive IR systems.** The most cited and well-known example from this category is the Fab (Balabanovic and Shoham 1997) multi-agent system developed at Stanford University. Fab recommends web pages using adaptive information retrieval techniques to learn an individual’s profile. For that, it takes into account users’ feedback on how much they liked recommended pages used to adapt the user’s profile and assign credit or blame to the recommending collection agents. A “genetic algorithm” is used to evolve the population of collection agents. Collection of agents will specialize over time to different topics, serving distinct groups of users and useless collection agents die, successful ones live and reproduce.

• **Collaborative IR systems.** A collaborative filtration agent-based IR system makes recommendations to a person based on the preferences of similar users. Content-based recommendation retrieves other documents similar to those liked earlier, while collaborative recommendation retrieves documents liked by other people similar to a relevant one. Typically, they operate with a large vector space (in which each element represents one dimension) and a sparse vector of element ratings. Some classical examples for collaborative IR systems include Yenta, (recommend people), Firefly (recommend products), and Phoaks (recommend readings) (Foner. 1997).

• **Proactive IR systems.** These are systems that only provide information "on request". The user has to know that there is knowledge to be had in a particular situation. The system therefore needs at least some ability to be proactive in its suggestions. However, unlike the calendar program that warns of upcoming meetings, it is impossible to create a back end able to reliably know when a document is useful for a user. In this context Remembrance Agent (Rhodes and Starner 1996) indexes personal files and e-mails when you perform a task, automatically suggesting relevant documents providing continuous associative recall. In the same category, Letizia (Lieberman 1997) is a user interface agent that assists a user in browsing the World Wide Web.

### 3.3 Architecture of the system

#### 3.3.1 Multi agent system for active support

An implementation should support the following issues:

1. Assist the user in the diagnosis process and question reformulation;
2. Select appropriate search engine for efficient searching accordingly to their profiles;
3. Translate the question into one or more queries and search strategies acceptable to the given search engine;
4. Manage searching strategy;
5. Support the user in the results assessment;
6. Support the user in resource description
7. Provide the user with the appropriate outputs in a suitable structure; and, way not, 8. Advice he or she in the follow-up activity.

The entire system is composed by several modules as in next figure.

![Figure 21 Overall architecture](image)

The Web Browser Control has to present to the user the found results and the web pages.

The User Interface Agent deals with user input and shows the results. It consist in several sub-modules: New User Wizard assists the user in creating a new user for the system (it takes some basic information from the user which will be used later for deciding how many search process details will be hidden from him and which default parameters are to be used), New Profile Wizard that assists the user in the definition of a new interest profile (the collected information consist in the interest domains and relevant keywords - it is not necessary to provide the keywords but doing so will greatly improve the search process because the training period will be much shorter), the Search Wizard takes the request from the user and forwards it to the Profile Agent (some search parameters may be set in this wizard, also), and the Result Processing sub-module deals with various conversions needed in the presentation process.

The Profile Agent has to maintain the user profiles and use it in the search process. Its main tasks are to generate the query from the user request and to refine the user profile once the search process is completed. Thus, the Query Generator takes user requests and
build generic queries used by the search engine agents (for this it use information from the user profile), the Results Refinement perform a classification of the founded links (this classification is done by taking the page ranking given by the search engines and the keywords from the query and user profile), the XML Generator, as its name suggest, generates a XML document with the sorted results and send it to the User Interface Agent, the User Activity Monitor collects information from the web browser (these consist in links followed by the user and where the search process had finish), the Profile Refinement updates the user profile (this is done by analyzing the visited pages and extracting relevant keywords for the user).

*Search Engine Activator* will decide where to search the needed information. It selects between several search engines that are adapted for a specific domain. Here, the Agent Factory manage the available search agents (there is an agent who knows how to do an Internet search and it is cloned for each search engine to deal with it), the Agent Activator starts the agents, monitor their search progress and gather the results, and the Response Generator will filter the founded links (it deletes duplicate links and may check for death ones).

*The Database Storage* module handles the user profiles and the search engine profiles.

### 3.3.2 Profile agent

User profile is a complex structure and must capture the user profile from different views. Part of user profile is managing information of a general nature that are obtained by filling out forms and which can then be used to infer some of the user’s interests. This information covers the full name, age, geographic location, what he likes and dislikes and more. Types of information stored in this part of the profile should be determined by sociological studies.

Another component of the agent will manage the profile associated with the retrieval and filtering of information and will be described in detail below. Other components will manage specific activities oriented profiles, one of which is e-commerce.

![User profile agent structure](image)
The user may belong to one or more interest groups that may share parts of their profile. For communication with other agents, agent profile should recognize at least one communication language (e.g. KQML).

If the user uses the system for more than one language for each additional language will create a new corresponding component. Ideally, you should be able to provide online translation in one language but such an approach is not feasible at present.

For example if the user browsing pages in Romanian and one in English language system will automatically detect and use for computation documents from the appropriate profile - English portion of the document in English and Romanian portion of the document in Romanian.

Accordingly to (Saracevic, Spink, and Wu 1997), approaches to user modeling in IR can be divided in two main categories: system-centered and human-centered.

While the first put emphasis on relevance feedback (users are modeled through texts or clusters of texts) and query expansion (the initial or modified query is used as a basis for user modeling).

The second one takes into account question shape (user modeling is accomplished through various interview and analysis techniques). Another method is to build into the system ways and means by which users can on their own model articulate their problem with the system’s assistance.

On the user side we can model cognitive, affective and situational levels. Saracevic et al. (Saracevic, Spink, and Wu 1997) suggest that user modeling is an interactive process that proceeds in a dynamic way at different levels trying to capture user’s cognitive, situational, affective and possible other elements that bear upon effectiveness of retrieval. In such a framework the request must be scrutinized through all its related steps: from request formulation to answer acceptance. So, in a searching process can be delineated three key stages:

- Request formulation,
- Selection of retrieved pages and
- Locating the intended information.

Keywords occurring in a particular searching process (e.g. source description, contextual links) will be clustered using a similarity matrix for the keywords stored in the user profile, very similar with the approach followed by (N. J. Davis, Weeks, and Revett 1997) in their Jasper implementation. Contrasting with them, we look at the search process as a whole, instead of the pages stored in the ultimate part of the user’s request. System captures the user’s choice, the rationale behind each of them, the open questions
related to the request, the assumption behind it and any related supporting information. The matrix used will give a measure of the ‘similarity’ of keywords in the users profile.

For two keywords $K_i$ and $K_j$, the Dice coefficient is given by the equation below.

$$2 \times \frac{|K_i \cap K_j|}{|K_i| + |K_j|}$$

Once the similarity matrix is calculated it is exploited in two ways: profile enhancement (adding those keywords most similar to the keywords explicitly represented in the user’s profile in similar way of query reformulation techniques) and proactive searching (search proactively for new WWW pages relevant to user’s interest).

Using complete-link clustering technique (N. J. Davis, Weeks, and Revett 1997) the similarity between the least similar pair of items from two clusters is taken as the similarity between the clusters obtaining the cluster dendogram. A similarity threshold can be set to provide the similarity degree between the clusters.

### 3.3.3 Adaptive recommendation

Accordingly to (Balabanovic and Shoham 1997) for the content-based approach, there are four essential requirements:

- $w$ – a representation of a Web page.
- $m$ – a representation of the user’s interests.
- $p(w, m)$ – a function to determine the pertinence of a Web page given a user’s interest
- $u(w, m, s)$ – a function returning an updated user profile given the user’s feedback $s$ on a page $w$.

The assumption underlying content-based systems is that the content of a page is what establishes the user’s interest. Going on, the content of a page can be represented purely by considering the words contained in the text and also by its description. Considering the vector-space model of IR (Salton and McGill 1983) as a suitable mechanism for documents based representation, documents and queries are represented as vectors. This model has been used and studied extensively, representing a competitive representation form with alternative IR methods (Harman 1994). This model assumes a dictionary vector $d$, where each element $d_i$ is a word. Each document then has a vector $w$, where element $w_i$ is the weight of a word $d_i$ for that document. If the document does not contain $d_i$, then $w_i=0$. As in Fab implementation (Balabanovic and Shoham 1997), in system formulation was reduce words to their steams using the Porter algorithm (Porter 1980). That will ignore words from standard stop-list-words, and calculate a TFIDF weight: the weight $w_i$ of a word $d_i$ in a document $W$ is given by equation below.
\[ \text{Sim}(d_i, d_j) = 2 \times \frac{|d_i \cap d_j|}{|d_i| + |d_j|} \]

\[ w_i = (0.5 + 0.5 \frac{\text{tf}(i)}{\text{tf}_{\text{max}}}) \times (\log \frac{n}{\text{df}(i)}) \]

\[ p(w, m) = q(w) \times m \]

\[ u(w, m, s) = m + z(t) \times w \]

- \( w \) - a Web page
- \( m \) - user’s interests
- \( \text{tf}(i) \) - the term frequency
- \( \text{df}(i) \) - the document frequency
- \( n \) - the number of documents
- \( \text{tf}_{\text{max}} \) - the maximum term frequency
- \( p(w, m) \) - the pertinence of a Web page to user’s interest
- \( q(w) \) - return the similarity measure
- \( u(w, m, s) \) - updated user profile given the user’s feedback \( s \)
- \( z(t) \) - user’s score for a page \( w \)

\( \text{tf}(i) \) is the number of times word \( d_i \) appears in document \( W \) (the term frequency), \( \text{df}(i) \) is the number of documents in the collection which contain \( d_i \) (the document frequency), \( n \) is the number of documents in the collection and \( \text{tf}_{\text{max}} \) is the maximum term frequency over all words \( W \). To avoid over-frittering, accordingly to experiments described in (Pazzani, Muramatsu, and Billus. 1996), the optimum of used words is between 30 and 100. In implementation, because the algorithm is used especially for recommendation based on the page’s description and not for the content of the page itself, a range between 20 and 50 is sufficient. Once the top approximately 30 words have been picked we normalize \( w \) to be of a unit length to allow comparisons between documents of different lengths.


**Capitolul 4  Conclusion**

Customized systems for retrieving and filtering information modeling user interests and their constant updating are the major problem. There have been proposed several methods of solving the problem, each problem on a different viewpoint. Few of models address the problem of representing the interests of short-term and long-term. The system will quickly learn from a blank profile and will gradually adapt to changes of interest arising. Scheme and algorithms working in vector space model. By varying the weights and algorithms used in calculating vectors of document features can get various answers.

The user has a major task in the system. It must be aware of what they want and to provide feedback to the profile agent. Even if it is not desirable for the user having to explicitly provide a reaction now is the only viable approach. The system tries to cope without explicit side but then his learning ability is diminished.

Collaborative filtering can be implemented for the user to benefit from the knowledge (profiles) with other users. Expanding learning ability by trying to deduce in advance what type of information the user would need.

Although the separated features presented in the architecture of the system have been treated separately by the current approaches, the trends of economic environment impose the need of an osmotic approach able to deal with heterogeneous resources. Current system architecture promotes a more anthropocentric orientation; improve data access capabilities and communication ability. Compared with other approaches, it greatly enhances the IR effectiveness on the Web, reaches more extensive problem domains, more component problem-solving capability. To carry out these functions, three kinds of knowledge are identified that system have to deal with: user’s cognitive, situational, affective and possible other elements that bear upon effectiveness of retrieval.

The approach based on three descriptors can be extended with new ones that can improve learning of the long term interest (with different influence).

The presented methods can be used in the first decision phase (as described in chapter ...) Intelligence, where personalization can be used to produce better and faster results.

Personalization is about selecting and filtering information objects or products for an individual by using information from his profile.

A challenge is to build libraries of reusable software patterns for different types of such agents (that can support the Intelligence phase), and provide corresponding easy-to-use plug-in information agent components to the common user.

With largely unstructured pages – even that semantic web try to put order on them – authored by a massive range of people on a diverse range of topics, simple browsing has given way to filtering as the practical way to manage Web-based information. Today’s online resources are therefore mainly accessible via popular information services such as search engines.
Search engines are very effective at filtering pages that match explicit queries. Most people find articulating what they want extremely difficult, especially if forced to use a limited vocabulary such as keywords. The result is large lists of search results that contain a handful of useful pages, defeating the purpose of filtering in the first place.

Nowadays is possible to build a system that acts on the user’s behalf and that can rely on existing information services that do the resource-intensive part of the work is a system that users can accept and use. These systems can be built to run under current PC as personal assistants – as a widespread of mobile technologies (smart – phones and tables) such system must run also on mobile systems. The assistant resides on the user’s machine and there is no need to turn down intelligence. The system can have substantial local intelligence and information customization becomes possible.
Capitolul 5  References


Bellifemine, FL, G Caire, and D Greenwood. 2007. Developing multi-agent systems with JADE.


Fasli, Maria. 2007. Wiley & Sons *Agent technology for e-commerce*. http://csee.essex.ac.uk/staff/mfasli/textbook/TableOfContents.pdf (May 9, 2013).


http://books.google.com/books?hl=en&lr=&id=TDhPMs3adw0C&oi=fnd&pg=PR6&q=Web+Mining+Applications+and+Techniques&ots=r1_Ctyh7ed&sig=RzyuEXWL4BNnx90XAQaHKxHfxIc (May 25, 2013).


Wang, Y., D. 2006. “A decision theoretic approach to the evaluation of information retrieval systems.” University of Maryland Baltimore County, Baltimore, MD.


http://ebooks.cambridge.org/ref/id/CBO9780511811654.


Capitolul 6  Annex

A Stigmergic Guiding System to Facilitate the Group Decision Process - Constantin-Bala ZAMFIRESCU, Ciprian CÂNDEA, In proceedings of 28th IEEE International Conference on Data Engineering Workshop on Data-Driven Decision Guidance and Support Systems (DGSS), Washington DC, USA, 2012


To appear:

Towards a Cloud-Based Group Decision Support System - Ciprian Candea, Ciprian Radu, On 5th International Conference on Computational Collective Intelligence Technologies and Applications, 11-13 September 2013, Craiova, Romania

On Interacting with Collective Knowledge of Group Facilitation - Constantin-Bala ZAMFIRESCU, Ciprian Candea, On 5th International Conference on Computational Collective Intelligence Technologies and Applications, 11-13 September 2013, Craiova, Romania