Intelligent Software Agents for Products Penetration Strategy Selection

Nikos Matsatsinis, Pavlos Mora?tis, Vangelis Psomatakis, Nikos Spanoudakis

Technical University of Crete, Decision Support Systems Laboratory,
University Campus, 73100, Chania, Greece
{nikos, moraitis, psomat, spanoud} @dias.ergasya.tuc.gr

Abstract. This paper describes an intelligent software agents based system implementing an original consumer-based methodology for product penetration strategy selection in real world situations. Agents are simultaneously considered according to two different levels: a functional and a structural level. In the functional level, we have three types of agents: task agents, information agents and interface agents assuming task's fulfillment through cooperation, information gathering tasks, and mediation between users and artificial agents respectively. In the structural level we have elementary agents based on a generic reusable architecture and complex agents considered as an agent organization created dynamically in a recursive way.

1. Introduction

Products penetration strategy simulation is a complex distributed decision-making task, involving several actors belonging to different levels of responsibility and having complementary functionalities within an organization. Several works have been proposed in *Marketing Decision Making* literature [11], [13], [19], [20], presenting different approaches in order to support product development process. In this framework, an original consumer-based methodology is proposed by [14]. All these approaches used different information technology techniques (Intelligent Decision Support Systems, Management Expert Systems, etc.) to implement their methodologies. However, none of these works has tried the real world modeling distributed dimension of the process. Actually, many decision-makers are involved, using heterogeneous knowledge and needing vast amounts of data and information about consumers and market features available in different distributed information sources. The goal of overall process within an organization is to find the most appropriate penetration strategy for a new product design, taking into account all decision makers points of view and being as much suitable as possible to customers preferences. To achieve this, the tasks of locating information sources, accessing, filtering and integrating information in support of consistent distributed decision-making are critical.

In this paper we use reusable intelligent software agents to implement the methodology proposed [14] in a real world dimension context. Agents are considered simultaneously according to two different levels: a *functional* level and a *structural*

level. In the *functional level*, we have a natural distinction between three different agent's functionalities: the information gathering task, the task's fulfillment by different types of cooperating specialists and the mediation between users and artificial agents, in order to allow users to control the actions of their agents. Therefore, we consider three types of agents [18]: *interface agents*, *information agents* and *task agents*. In the *structural level* agents are considered as *elementary agents* and *complex agents*. We consider that complex tasks can be decomposed in a recursive way in several subtasks. In a similar way an agent structure can be considered according to different nested layers created in a recursive way. We can imagine a complex agent as a "Russian doll". The agent's layers are related to the subtasks which are carried out. This conception is inspired by the representation of a complex system through multiple layers proposed in control theory [15]. Therefore, an agent is considered as a complex one when he realizes a task (e.g. a scenario generation) involving several agents of at least one lower layer. An agent is considered as an elementary one, if he realizes a primitive task. From a methodological point of view, we believe that this consideration facilitates the conception and the design of complex systems of multiple intelligent software agents, in order to achieve the modeling of complex real applications such as the one we present in this paper.

The application of all the characteristics considered in the literature [9], [18] as necessary for agency technology use, motivated us towards an agent-oriented approach. These characteristics are: a) the inherent distribution of problem solving abilities (the agents perform different data analysis, brand choice and multicriteria analysis methods), data, information, b) the necessity of flexibility, modularity (agents can appear and disappear in the system without disturbing its functionality) and reusability (customization of agents for new decision makers), c) problem solving complexity involving coordination between actors expressing different points of view.

In the following, section 2 describes the implemented methodology in a distributed context. Section 3 presents agents functionalities, possible structure, knowledge held and interactions within an organization. Section 4 details an elementary and complex agent architecture, while section 5 presents an example of the system's operation for a penetration strategy simulation. Finally, in section 6 we conclude by comparing our approach to related work.

2. Consumer-Based Methodology for Products Penetration Strategy Selection

To support the product development process [14] proposed an original consumer-based methodology (Fig. 1). It is based on the use of different models for data analysis, multicriteria analysis and brand personal choice.

During the market survey, every consumer expresses his evaluations of a set of reference products involved in the research, on the base of a group of criteria. Finally, he is requested to rank the products according to the order of preference. The collection of this kind of data requires a specific questionnaire [14].

The initial phase of this methodology aims to acquire an overall frame of the

particular survey. This is followed by the use of data analysis models in order to determine consumer and market features. This task is called "Market Segmentation". Market trends are identified through this approach. Concurrently, the multicriteria method UTASTAR [17] is applied to the multicriteria consumer preferences, in order to determine the criteria explaining each of the consumer's choices. This method assesses a utility function $u(\underline{g})$, which is as consistent as possible with the consumer ranking, where $\underline{g} = (g_1, g_2, ..., g_n)$ is the vector of the criteria on which the products are evaluated. The consumer's utility function is assumed to be additive: $u(\underline{g}) = p_1u_1(g_1)+p_2u_2(g_2)+...+p_nu_n(g_n)$, where $u_i(g_i)$ is the estimated marginal utility of the criterion g_i, normalized between 0 and 1, and p_i is a weighting factor of the i-th criterion, the sum of weights being equal to one: $\sum_{i=1}^{n} p_i = 1$.

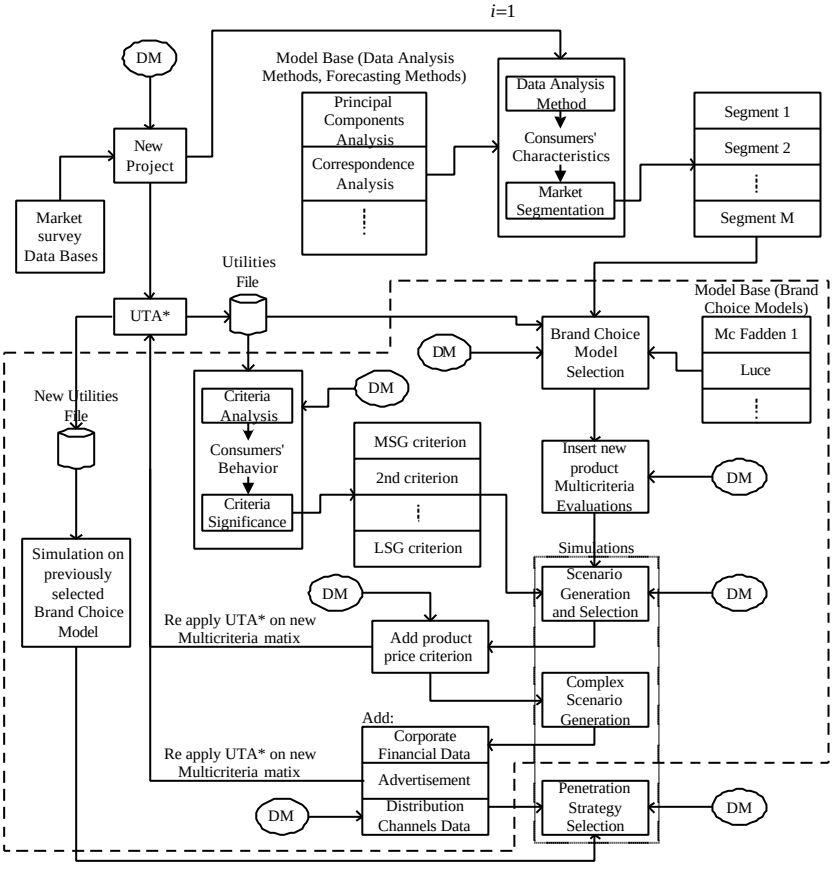


Fig. 1. Methodological flowchart (source: Matsatsinis and Siskos, 1999)

The UTASTAR method estimates for each consumer separately his utility function, which is as consistent as possible with his rank order of the products used;

the relative importance of the criteria is then derived from this utility model. This preference disaggregation analysis is called "Criteria Analysis". The use of models of consumer personal choice allows the market simulation and the calculation of the market shares of the competitive products taking part in the research. This aims at the selection of the most suitable model approach, as close as possible to the real market shares ("Brand Choice Task"). The next step concerns the design of a new product under development by simulating its introduction into the market using the multicriteria estimations. It is followed by the application of alternative scenarios. With the help of the selected brand choice model, the market simulation and the calculation of the new market shares to be expected, after the introduction of the new project, are performed. This process involves "Scenario Generation and Complex Scenario Generation". Based on the results of the scenarios application, the choice of the most appropriate penetration strategy for the new product is made. This is the main task and is called "Penetration strategy selection".

3. The System's Architecture

In Fig. 2 we present an intelligent software agent based architecture used by decision-makers, who can be corporation board members, each simulating his own scenarios and finally selecting a penetration strategy for a new or an existing product in a board meeting.

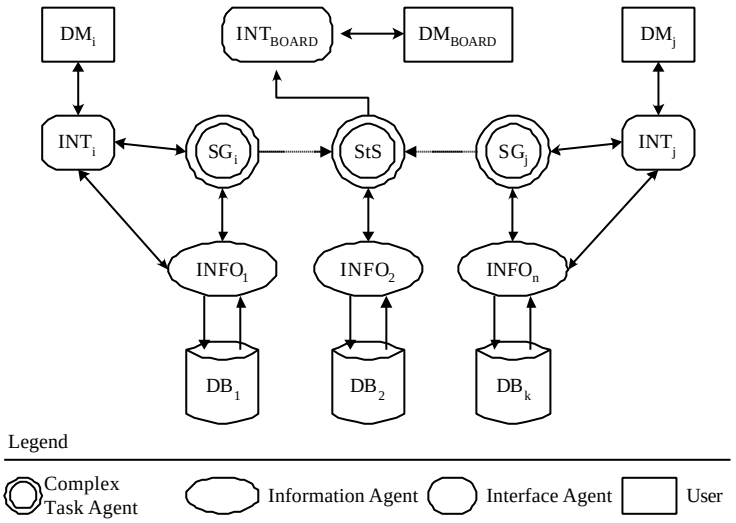


Fig. 2. Agent based architecture

3.1. Agent's Types, Functionalities, Structure and Knowledge held

An agent's knowledge is acquired during a knowledge acquisition stage, using

different domain experts, and through interactions with the other agents of the system as well as the human users (§3.2). The knowledge used by the different types of agents is discussed in section 2. In our approach, agents are considered according to two different levels: a *functional* level and a *structural* level. In the *functional level*, we consider three types of agents like in [18]: *interface agents*, *information agents* and *task agents*.

The functionalities of *interface agents* are those we can find in the literature [12], [18]: initiation of a task, responsibility of system interactions with the user, results presentation to user queries, in a way appropriate to the user's profile (e.g. according to the level of responsibility in an organization), determination of what categories of task agents should be involved, so that a user query is correctly taken into account.

The functionalities of *information agents* are also those we can find in the literature [10], [18]. Their goal is to provide information and expertise on various topics, by drawing on relevant information from the system's general database, remote heterogeneous databases in the Internet, other information agents or interface agents.

Finally, *task agents* specialize in performing specific tasks. They can interact with all types of agents in order to carry out their jobs. These are the most sophisticated agents of our system and they can have an elementary or complex structure. For the application presented in this paper, we conceived different types of task agents (elementary and complex) each corresponding to different *generic tasks* (e.g. perform a data analysis method, generate a scenario), involved in the methodology presented in section 2. Several occurrences of the same type are used to perform specific tasks, instances of the above generic ones (e.g. performing a specific data analysis method as correspondence analysis, simple regression, etc.). Finally, we can also have several occurrences performing the same specific task (for example several agents performing the same brand choice model).

We therefore have the following types of elementary task agents:

- *Data Analysis Method (DAM) agent*: such an agent performs a specific data analysis method (see Fig 1, e.g. correspondence analysis, principal components analysis, etc.)
- *Data Analysis Selection (DAS) agent*: such an agent has the knowledge that enables him to chose appropriate DAM agents to perform specific data analysis methods, which are effective on a particular input data set
- *Data Analysis Results Combination (DARC) agent*: such an agent can combine and evaluate the DAM agent's outputs
- *Brand Choice Method (BCM) agent*: such an agent performs a specific brand choice method (Fig. 1, e.g. LUCE, Mc Fadden 1, etc.)
- *Brand Choice Selection (BCS) agent*: such an agent uses results of a UTS agent (Fig. 4) enabling him to evaluate different brand choice models, in order to chose appropriate BCM agents to apply such a brand choice method effectively on a particular input data set.
- *UTASTAR agent (UTS)*: such an agent performs the UTASTAR multicriteria method

By using these elementary agents, we build complex agents (§4.2), taking into account the methodology's complex tasks achievement (§2). We consider that by using the complex agent concept to gather together agents involved in some complex task (if the task's nature allows it) achievement, the system's scale and coordination

complexity can be decreased, making the application's modeling easier. Actually, coordination, even within a large-scale application, is carried out, either between agents within relatively small-scale groups or between a reduced number of complex agents that are components of an upper layer. In the latter case, coordination is carried out by intra-agent control primitives (§4.2) assuming interaction between lower layer agents of a complex agent:

- *Data Analysis Agent (DAA)*: such an agent is composed of a DAS agent, several DAM agents (corresponding to the different available data analysis methods) and a DARC agent. He takes into account the market segmentation task.
- *Brand Choice Agent (BCA):* such an agent is composed by a BCS agent and several BCM agents (corresponding to the different available brand choice models). He takes into account the brand choice task.
- *Scenario Generation (SG) agent*: such an agent is composed by a DAA, a BCA and an UTS agent. He takes into account the scenario and complex scenario generation task (Figures 1, 4).
- *Strategy Selection (StS) agent*: such an agent is composed by a BCA and an UTS agent. He takes into account the penetration strategy selection task. He selects a market strategy depending on scenarios and on knowledge that includes corporate information, distribution channels information, etc.

3.2. Agent Organization

Agents can be geographically distributed allowing the interaction with users of different levels of responsibility and involvement in the main problem solving. Agents (elementary and/or complex) interact with each other by means of passing messages. A message is structured in such way that allows the transfer of the necessary information and semantics between agents for cooperative work accomplishment. Agent interaction within a complex agent is discussed in §4.2. Our agent organization allows the following interactions types (for an extended illustrative scenario see section 5):

- New agents are introduced while others "died". An agent will be informed when a new member enters his community or when another leaves. A new agent must present himself to the other agents community by broadcasting a message with his identity (e.g. his address, his abilities, his preferences, etc.). This is important from a software-engineering point of view, because it allows modularity, reusability and flexibility of the system.
- During the problem solving process, appropriate agents activation dynamically forms an organizational structure that fits with the current goal (a specific task accomplishment, an information retrieval, etc.). In our system, interface agents activate task agents. They can perform or assign a specific task or tasks to a set of cooperating task agents, by using appropriate criteria (e.g., their abilities, their availability, and their performance for a similar task in the past, etc.).
- Activities of information agents are initiated, either top down by a user or a task agent through queries, or bottom up through monitoring information sources for a particular information. Once the monitored for condition has been observed, the information agent notifies interested agents, by means of messages, of the updated

information.

- The interface agents can receive messages from users. After one such agent infers a user's needs from a user request, he starts a new task giving it an *id*. He then uses information gathered or previously held, in order to decide which task agent should be the first to work for this task. While the task agents are carrying out a task, the interface agent that started it monitors its progress.

4. Agent Architectures

In this section we present the elementary and complex agent architectures (Fig. 3).

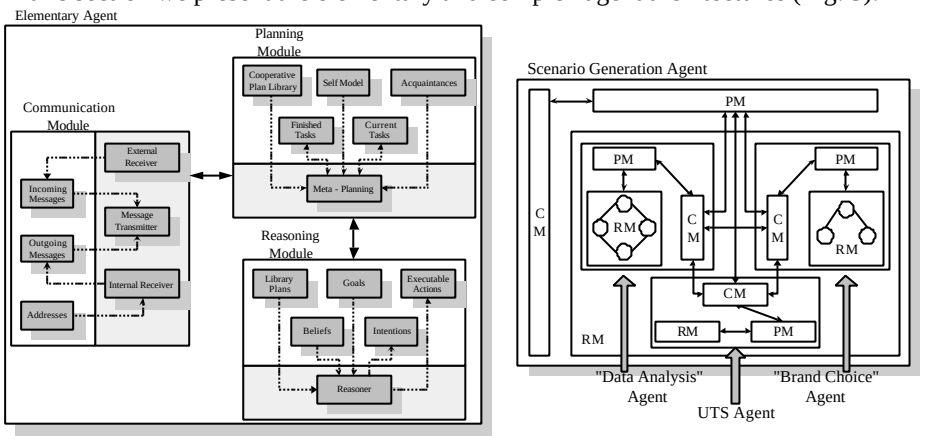


Fig. 3. a) Elementary (e.g. UTS agent) and **b)** complex (e.g. SG agent) agent architecture view in top layer

4.1 Elementary Agent Architecture

The agents, which are used for the presented application modeling, are based on a generic reusable architecture that we conceived, following the general BDI type philosophy [5], [16] and we were inspired by the different agent architectures presented in the literature [1], [18], [21]. Different functional agent types have the same basic architecture principles, regardless of the category they belong to. However, the different models are more or less sophisticated according to their specific type (e.g. the planning model of an information agent is simpler than that of a task agent). Our agent architecture (Fig. 3a) is composed of three modules (*Communication, Planning and Reasoning module*) that intercommunicate through internal message exchanging (called intra-agent messages). These modules run concurrently. An agent remains idle while no messages arrive to his communication module. As soon as a message arrives, the communication module determines its importance and, after transforming it to an intra-agent message, sends it to the planning module by means of a message queuing mechanism. All modules adopt this

behavior and remain idle while no messages are available for procession. The same intra-agent queuing mechanism facilitates all modules.

The *communication module* is responsible for the agent's interaction with his environment. It sends and receives messages, while internally it interacts with the planning module. Its functionality is quite straightforward: an *internal receiver* process transforms each internally queued message to an inter-agent message, adding the receiving agent's address and then stores it to the *outgoing messages* queue. An *external receiver* process realizes the opposite by transforming each received external message to an internal format and writing it to the *incoming messages* queue. A *message transmitter* process monitors the incoming and outgoing queues sending all queued messages either to the planning module or to another agent accordingly. It has also the necessary "intelligence" to assume the interaction of an elementary agent with his parent agent (case of complex agent, §4.2).

The *planning module* is responsible for proper cooperation between agents and also undertakes the task of planning the agent's future interactions. It interacts with the communication module as well as with the reasoning module. The planning module needs to keep track of the agent's *finished tasks* (keeping track of his history). Its *self model* (composed of agent characteristics) and *cooperation plans library*, are two parameters that influence the planning procedure, the first by evaluating his capability to undertake a certain task, the second by providing alternative cooperative scenarios with other agents (his parent complex agent is also included). The *current tasks* structure contains information regarding pending jobs, which are either delayed or postponed (for example, the agent might expect more information to arrive so that a working task is resumed). By use of the *acquaintance* structure the agent can choose his collaborators.

The *reasoning module* takes care of the correctness of the tasks or subtasks under execution, while it finds the ways that lead to task completion. It communicates with the planning module only. When a request arrives, it firstly becomes a goal. Then the *reasoner (or inference mechanism)* selects appropriate plans from the *plan library* based on *goals* and *beliefs and facts* data structures. When a plan is selected it is placed in the *intention* structure. Finally the reasoner, by using criteria (for example efficiency, speed, etc.) triggers a sequence of *executable actions* placed in the corresponding structure.

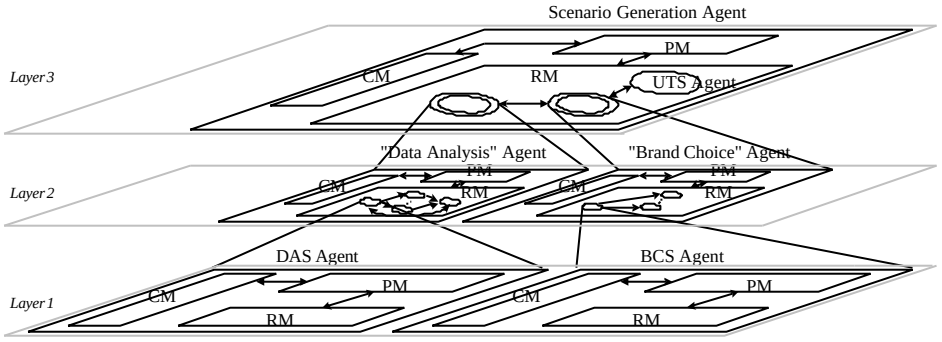
4.2 Complex Agent Architecture

Complex agents can belong to the three functional types defined in §3.1. The architecture of a complex agent (a detailed description is out of the scope of this paper) is similar to the one of an elementary agent. Therefore he is composed of the same three modules (C*ommunication, Planning and Reasoning module)* which intercommunicate through internal message exchanging. The intra-agent control (interaction between the three components) is the one of the elementary level

The difference is situated in the structure of the reasoning module. The group of agents (elementary and/or complex) which compose it assumes its role. The task achievement of an agent (parent) developed in *n-layer* is therefore the result of the set of agent's (his descendants) cooperation belonging to the previous (*n-1) layers*. The

reasoning module could be therefore considered as an *agent organization*.

The interaction between a complex agent's reasoning and planning modules (like in the elementary level) of an (i+1) layer agent is established through the i layer agents that are components of the reasoning module of the (i+1) layer agent (for example the "Brand Choice" Agent sends the results of his work to the planning module of the "Scenario Generation" Agent, Fig. 3b). In this context, an inter-agent message sent by an i layer agent to an (i+1) layer agent is transformed (by the Message Transmitter of the communication module of the i layer agent) to an intra-agent message of the (i+1) layer agent (Fig. 3b). The organization of reasoning module agents is dynamically generated as presented in §3.2. Its role is to achieve any task(s) allocated by the planning module. The agent's organization generation process is initiated selecting appropriate agent(s) (according to the task's nature) chosen (as in §4.1) by the planning module. Agents can be of a different nature (e.g. static, mobile), not necessarily implemented in the locality of the parent agent (they can communicate by using e-mail, ftp, etc.), but they are, however, aware that they have the same parent. In our application (Fig 4), a DAS agent, several DAM agents and a DARC agent compose a complex "Data Analysis" Agent, as well as a BCS agent and several BCM agents compose a complex "Brand Choice" Agent. These two complex agents perform two subtasks of the methodology and together with an elementary UTS agent they compose an upper layer SG (alias "scenario generation") complex agent.



.

Fig. 4. Layers of a SG complex agent structure

Communication and Planning modules have exactly the same structures and functionalities as in the elementary level. The acquaintances structure contains additionally the agent's "children". The communication module operates similar to that of an elementary agent. Messages exchanged between two agents belonging to different parents, transit (as intra-agent messages, §4.1) through the planning modules of their parents. The planning module interacts with the communication module as in the elementary level (§4.1) the latter enabling the message exchange. This process is transparent to original senders in a conceptual level (it is performed by the Message Transmitter of the communication modules) and can be repeated in a recursive way inside *n-layer* complex agents. Each layer adds information to the data by prepending headers to the data it receives. Let's, therefore, consider an α_i (i layer agent) belonging to an A_n (n-layer) complex agent and an α_j (j layer agent) belonging to A_m

(m-layer) complex agent. Messages exchanged between α_i and α_j transit through each layer of the two complex agents. More precisely, if α_i sends a message to α_j this message will go upwards through each layer of A_n agent until it is sent by the communication module of the (n-th) layer agent. This message will be received by the communication module of (m-th) layer agent of A_m agent. It will then go down through each layer of A_m agent until finally it is received by the communication module of α_j.

We can say, that two processes facilitate a complex task achievement. A top-down process assumes that the task's decomposition in several subtasks is achieved across the different layers, while a bottom-up process performs the synthesis of different solutions proposed at different layers. We can have complex information agents when an information retrieval must be accomplished through the achievement of several specific information gathering goals. Different specialized information agents representing layers of a complex information agent can take these goals into account. We can also have a complex interface agent able to take into account (through his elementary interface agents) the different points of view of board members during a distributed decision making process.

5. An Example of the System's Operation

The system is a prototype running under Windows NT, where the fundamental agent's components are built on a Java platform and the graphics user interface (GUI) on a Visual Basic environment.

We proceed with an abstract description of an example, which shows how the proposed methodology (see Sect. 2) is implemented by our system.

A decision-maker attempts to decide on a market penetration strategy for a new product, which is currently under development. The first available INT agent (INT) establishes immediately a new project and begins collaboration with the human agent, who initiates the project by defining his aim and constrains, details and demands. At this point the decision-maker has to define which questionnaire's database should data analyzers query, any query details, special necessary conditions, complexity matters etc. Several databases are reconstructed with respect to the human user's specification. The analysis procedure is performed to each one separately. Depending on the information gained, the INT agent begins searching for an available SG agent. Concurrently, he sends the information request to an INFO agent for database querying. When a SG agent is found, the INT agent informs the INFO agent that his data gathering results have as destination the committed SG agent. The SG agent delegates to his "Data Analysis" agent (DAA) component the market segmentation subtask and forwards to him the INFO agent's results. Within this complex agent, a DAS agent must select one or several DAM agents according to the principles discussed in §3.2. After processing the received questionnaire's data the DAS agent concludes, for instance, that the candidate data analysis methods, *Principal Components Analysis*, and *Correspondence Analysis*, should be used. Next, the DAS agent, firstly locates a DARC agent and then the proper DAM agents, who will perform the chosen data analysis methods, sending their results to the DARC agent

that will combine them. As soon as all DAM agents have sent their results, the DARC agent evaluates them and depending on whether they are satisfactory or not, he might ask the DAS agent to perform extra data analysis or repeat the process. Meanwhile, the UTS agent of the SG agent performs the UTASTAR method and creates the utility file. The DARC agent sends to the human agent (through the INT agent and according to the process presented in §4.2) his results (market segmentation). Should the decision-maker find it meaningful to add extra criteria to the multicriteria matrix, he informs the INT agent. The INT agent transfers the new data to the INFO agent, which will construct a new multicriteria matrix, after querying the resources. This result is sent to the SG agent, which will forward it to the "Brand Choice" agent component. Within this agent, a BCS agent uses the UTS agent's results to test different brand choice models in order to choose the appropriate one, the *Luce* model for instance. The BCS agent will locate a proper BCM agent. After the method's execution, the BCM agent sends the results to the INT agent, which asks for the decision-maker's confirmation on the market simulation model, so that in case it is not accepted, the BCS agent is called for another model selection. At this point the decision-maker introduces via the INT agent his multicriteria evaluations on the new product so that a market that contains it is simulated. When the SG agent receives results from a BCM agent component, he introduces the scenario generation task. He allocates to the previously selected BCM agent (Luce in this case) the product's market shares calculations task, based on the most significant criteria obtained from the criteria analysis (UTS agent). Now the SG agent is able to select the best scenario and send it for confirmation to the decision-maker (through the INT agent). The human user has the possibility to insert product price as an independent criterion (see Fig. 1) and ask through the INT agent for another simulation taking into account that criterion. Since the complex scenario generation task has been completed, the SG agent sends his results to the StS agent of the system. Each SG agent committed by a decision-maker of the overall system carries out this procedure. During this last phase the Strategy selection Agent (StS) collects the accepted scenarios from the different SG agents and firstly chooses the dissimilar ones by interacting with board members (see Fig. 2). Then for each of them he produces a new multicriteria matrix with the specialist's evaluation on corporate financial data, product distribution channels, and advertisements (see Fig. 1). Having that new multicriteria tables data, the StS agent asks his UTS agent to perform the UTASTAR method so as to gain estimation on the new product's expected utilities for this scenario. Then he sends the results to his "Brand Choice" agent specifying to the BCS agent which specific type of BCM agent should be committed (Luce in this case since this type of BCM agent was used during the complex scenario generation phase). This information is sent to the StS because he must calculate market shares. The StS agent presents these new data to the decision-maker (through the INT agent), who then selects the best developing strategy for the new product.

6. Relative Work and Conclusion

In this paper we presented an intelligent software agents based system to support new

products penetration strategy selection process for the first time, compared to works of the domain [13], [14], [19], in a real world distributed context. The proposed system is built by using a generic reusable agent architecture. Agents are considered simultaneously in two levels, a functional and a structural level. In the functional level we have three types of agents, task, INT and INFO agents, while in the structural level we have elementary and complex agents. The difference with other works [2], [7], [8] is, like in [16], the differentiation between three types of agents in the functional level, allowing efficient operation for real complex tasks achievement involving coordination, information gathering and user interaction. Compared to [16] the difference is that we introduce the structural level consideration for the three types of agents (even if we have only implemented complex task agents in this application). We consider that by using the complex agent concept, thus gathering together agents involved in some complex task (if the task's nature allows it) achievement, the system's scale and coordination complexity can be decreased, therefore simplifying application modeling. Actually, coordination, even within a large-scale application, is carried out either between agents (elementary and/or complex) within relatively small-scale groups, or between a reduced number of complex agents, components of a distributed system (e.g. Fig. 2 the SG_i, SG_j and StS agents) or components of an upper layer. In the latter case, coordination is carried out by intra-agent control primitives assuming interaction between different layers of agents of a complex agent. In the literature, important works [1], [3] have introduced, in a similar way, the concept of complex (composed) agent, considering a generic agent model composed by six components. Each of them can be refined in many ways, resulting in models of agents with different characteristics [4], [5]. The difference with these works is that in our approach communication and planning modules have the same structure in elementary and complex level (this structure is also independent of the application), while the reasoning module can be considered in the complex level as an agent organization. This organization is created dynamically in a recursive way, it can have different forms and is task dependent. Our future work will be to apply our platform to new applications and to implement complex INFO and INT agents.

References

1. Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J.: DESIRE: Modeling Multi-Agent Systems in a Compositional Formal Framework. In: Huhns, M., Singh, M. (eds.): International Journal of Cooperative Information Systems. Special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems (1997a)
2. Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J.: Formal specification of multi-agent systems. In: First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, CA (1995) 25-32
3. Brazier, F.M.T., Dunin-Keplicz, B.M., Treur, J., Verbrugge, R.: Modeling Internal Dynamic Behavior of BDI Agents. In: Schobbens, P.Y., Cesta A. (eds.): Proceedings of Third International Workshop on Formal Models of Agents, MODELAGE'97. Lecture Notes in AI. Springer Verlag (1997b)
4. Brazier, F.M.T., Jonker, C. M., Treur, J.: Formalization of a cooperation model based on joint intentions. In: Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures and Languages, ATAL'96. In: Muller, J.P., Wooldridge, M.J., Jennings,

N.R.: Intelligent Agents III. Lecture Notes in AI, Vol. 1193. Springer Verlag, (1997c) 141-156
5. Brazier, F.M.T., Treur, J.: Compositional modeling of reflective agents. In: Gaines, B.R., Musen, M.A. (Eds.): Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop, KAW'96. SRDG Publications (1996) 23/1-13/12
6. Georgeff, M.P., Ingrand, F.F.: Decision-Making in an Embedded Reasoning System. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89). Detroit, MI (1989) 972-978
7. Ishida, T., Yokoo, M., Gasser, L.: An organizational approach to adaptive production systems. In: Proceedings of AAAI-90. Boston, Mass. (1990)
8. Jennings, N.R., Corera, J.M., Laresgoiti, I.: Developing industrial multi-agent systems. In: First International Conference on Multi-Agent Systems (ICMAS'95). San Francisco, CA (1995) 423-430
9. Jennings, N.R., Faratin, P., Johnson, M. J., O'Brien, P., Wiegand, M.E.: Using Intelligent Agents to Manage Business Processes. In: Proceedings of PAAM'96, (1996) 345-360
10. Knoblock, G.A., Ambite, J.L.: Agents for Information Gathering. In: Bradshaw, J.M. (ed.): Software Agents, (1997) 347-373
11. Kotler, P.: Marketing management: Analysis, planning, implementation and control. 8th edn. Prentice-Hall, London (1994)
12. Laurel, B.: Interface Agents: Metaphors with Character. In: Bradshaw, J.M. (ed.): Software Agents, (1997) 67-77
13. Liberatore, M.J., Stylianou, A.C.: Expert support systems for new product development decision making. In: A modeling framework and applications. Management Science, Vol. 41. No. 8, (1995) 1296-1316
14. Matsatsinis, N.F., Siskos, Y.: MARKEX: An intelligent decision support system for product development decisions. In: European Journal of Operational Research, Vol. 113. No. 2, (1999) 336-354
15. Mesarovic, M.D., Marko, D., Takahara, Y.: Theory of Hierarchical, Multilevel, Systems. Academic Press, New York (1970)
16. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: Proceedings of Knowledge Representation and Reasoning (KR'92). Cambridge, Massachusetts (1992) 439-449
17. Siskos, J., Yannacopoulos, D.: UTASTAR: An ordinal regression method for building additive value functions. In: Investiçao Operational, Vol. 5. No. 1, (1985) 39-53
18. Sycara, K., Zeng, D.: Coordination of Multiple Intelligent Software Agents. In: International Journal of Cooperative Information Systems. World Scientific Publishing Company (1996)
19. Van Bruggen, G.H.: Performance effects of a marketing decision support system: A laboratory experiment. In: Proceedings of the 21st Annual Conference of the European Marketing Academy, AARHUS (1992)
20. Wierenga, B.: Knowledge-based systems in marketing: Purpose, Performance, Perceptions and Perspectives. Working Paper 112, Rotterdam School of Management, Erasmus University (1992)
21. Witting, T. (Ed.): ARCHON: An Architecture for Multi-Agent Systems. Ellis Horwood Series in AI (1992)