Another step toward reusability in agent-based simulation: Multi-behaviors & aMVC

Yassine Gangat, Denis Payet, Rémy Courdier
University of La Réunion, LIM
Saint-Denis, Island of La Réunion
Email: yassine.gangat,denis.payet,remy.courdier@univ-reunion.fr

Abstract—The multi-agent systems are successfully used in modeling of dynamic complex systems, and simulations of these models reinforce the knowledge of experts and even allow them to explore new horizons or to cross boundaries. This is the reason why the models being tackled are increasingly varied, and as one goes along with experimentation, these models are completed, intercrossed. Consequently they become increasingly complex.

In our previous work [1], we proposed a first modeling approach to support this complexity increase: the Dynamic-Oriented Modeling (DOM). The application of this approach can effectively support the increase of the model. This increase applies to both agents and environments. This DOM approach tackles the problem of the latter by splitting in multiple parts. But if DOM led to organize properly the multiple environments that come into play, little support is provided to organize and manage the increasing complexity of the agents themselves. Inevitably, when we reach a quite advanced stage of evolution of the model, the agents eventually reach a critical mass (either in formalization or code) that makes them more and more hard to comprehend.

In this paper, we illustrate this phenomenon and show that it quickly takes the upper hand against the benefits of DOM, as it can eventually block the potential development, or even reuse, of the model. Then we explain that a solution to this "side effect" could structure the architecture of agents, a structure capable of maintaining readability and flexibility of the formalization of the agent throughout the growth process of the global model. We study a well known pattern in software engineering: the MVC pattern, which can be reused here to meet this objective. We will present in details how this pattern could be instantiated in the field of MAS architecture, and how, ultimately, it can be an effective new way to formalize agents in a method called Multi-Behaviors Modelization.

I. INTRODUCTION

Multi-agent System (MAS) has been more and more used in various scientific fields, especially in the simulation field (SOA). Even though some applications bring pedagogic purposes, others are made as decision-support tool, e-commerce application, etc. which implies a wide range of complexity's level. The more accurate we want it to be, the more complex the modeling phases will be, because of the high number of agents implied, their behaviors and their diversity (heterogeneous agents). Detailed models use a large number of parameters which usually characterize the complexity of the system. These difficulties mainly consist in the fact that the multiagent system are becoming more and more complex to modelize due to the complexity of the system studied.

Moreover, whenever a new application came into being and has been validated by its evaluation panel, experts typically wish to reuse it, completely or partially, in order to reduce time and expenses due to the creation of comparable application.

In order to improve the issue of model's creation and its reuse, we have presented in a previous article [1] the Dynamic-Oriented Modeling (DOM), a modeling method DOM (Dynamic-Oriented Modeling) based on dynamics, to tackle modeling difficulties which arise in a complex system.

A. Introducing DOM

As mention in [1], elaborated models bring into interaction a great number of dynamics producing the phenomena which characterize the evolution of the modeled system. Furthermore, it is frequent that various applications designed with multi-agent systems use identical dynamics.

Due to this consideration, we based this Modelization on dynamics. A dynamic is an association of a set of activities that participate in a major characteristic of a complex problem. For example, production, distribution and consumption of energy are activities that participate to the dynamic of energy evolution.

An agent (c.f. Figure 1a) is characterized by its state and behavior: the agent's state influences its behavior; and, the agent's behavior modifies its state. In addition, agents evolve in an environment which they modify through their actions and which send them information.

The aim of this modeling method is to break down a complex problem into some less complex parts (i.e. dynamics), using the environment (which is the location where agents evolve) as the coupling element for these dynamics in the Figure 1b.

In order to that, for each dynamic we will create a layer called a Mono-Dynamic Model (MDM). A such layer will be related to a specific dynamic (such as population evolution, or flow of energy). Then the whole system will be an aggregation of all MDM into a multi-MDM model which we call DOM.

As we can see in Figure 1b, due to the splitting of the environment into MDM, it has an effect on agents themselves. We will categorize states and behaviors around this definition of dynamics.

II. THE EDMMAS APPLICATION

The EDMMAS\textsuperscript{1} [2] project is one of the application of DOM to a real complex system.

\textsuperscript{1}Energy Demand Management by Multi-Agent Simulation

\textsuperscript{2}Information Processing Society of the Institute of Electrical and Electronics Engineers

DOI 10.1109/ICTAI.2012.158
1) **Context:** One of the main projects in the Reunion Island in 2009 was the GERRI\(^2\) project. It materializes the Grenelle Environment’s goals, of which the energy management of a territory is a facet. The aim is to forecast future energy production and consumption, bearing in mind economic and ecological indicators, and using a large number of interactions schemes between the involved parties.

The originality of this EDMMAS was to present a new approach using the multi-agents systems, which could offer an appropriate alternative to other models. This work is based on one of the tools (DOMino-SMA T) developed on the Geamas-NG platform. The DS Application [3] is a MAS simulation model that was made to explore different land-use and conservation planning scenarios for our region: the island of La Réunion (in the South-West Indian Ocean).

At the end of each simulation, DS gives us some indicators. For example, the main view of the application is a map showing the effective land-use map of each parcel of the island at every simulation step. It provides us also with graphics showing the global evolution of the land-use type (*i.e.* natural, agricultural or urban).

Because DS application has been approved by experts from La Réunion, we choose to reuse models in DS to build EDMMAS, which allows the simulation and geo-localization of energy flow while reckoning with modeled interactions.

Eventually, the aim is to provide a decision-support tool for the energy planning within a territory, thanks to the different possible scenarios, *e.g.* in anticipation of new power plant and its sizing, or in case of a malfunction or maintenance work.

### A. The model of EDMMAS

This model (c.f. Figure 2) is based on two coupled dynamics using the DOM method:

1. A population dynamic, that takes care of the evolution and the distribution of the population on the territory.
2. A land-use dynamic, which involves change of the land-use type (*i.e.* natural, agricultural or urban).
3. An energy dynamic, that deals with the type and amount of energy produced by power station as well as individual (through photovoltaic systems), the price and the flexibility of the production.

The first two dynamics has been implemented (and validated by experts) in DS while the last one was new. Each dynamic have its own ways of interacting with the environment, which is made of elementary spatial units of land (called parcels) arising from a territorial grid cutting of the whole territory. DS can be initialized with different sets of scenarios for the land-use politic and the population evolution.

In our model, due to the fact that our agents are space-localized, we are able to take into account the localization of the production of energy, and thus its consumption based on the population of each parcel.

The application EDMMAS is still in progress [4]. However, we already have identified some relevant issues from it. Apart from the simulation results, we realize that the more we extend

\(^2\)http://www.gerri.fr/images/stories/livret_gerri Bd.pdf
the models with other dynamic, the more the agents become dramatically complex.

B. Feedback

The feedback we had on this application showed us that using DOM was a good choice. Indeed it enabled us to easily reuse an "old" simulation and build a new one from it. But now, we are facing some underlying problems that are general to every case of reusability. Whenever someone wants to reuse a MAS model, especially its agents, he will breast the problem of agents’ behaviors. Despite the fact that dynamics had been separated, having the agents’ states and its behaviors at the same level doesn’t facilitate the reusability.

In order to better identify and solve this problem, we needed to explore this problem outside our platform GEAMAS-NG, the complex system’s simulation platform developed within our laboratory, implementing the DOM methodology, which also is the platform used for DS and EDMMAS. This is the reason why we propose a complementary study in a project called “Turtle” in the next part where we tried a collaborative method to build the model.

III. THE TURTLE APPLICATION

A. Introducing collaborative method for the model construction

This method [5] is based on modeling and prototyping simultaneously. This allows us to identify problems that we hoped to answer, thus narrowing down the amount of data that are really exploitable, to identify missing data that should be collected.

In order to get back to the original problem, we have presented in [5] a collaborative method and a NetLogo prototype focused on green turtles in the South-West Indian Ocean, leaving for a while the DOM method. Thus, we were able to test various hypotheses and focus more on the thematic conception of the model than the technical side. It allowed us to close-in on the thematic patterns we wanted to modelize, working hand-in-hand with thematicians, experts in another domain of Science.

1) The Turtle Application: Behind the apparent simplicity of green turtle life history, there are very complex interacting mechanisms driving population dynamics. The construction of the model itself helps in the comprehension of the biological system. For this work, our approach was conducted by modeling and prototyping at the same time.

Beyond the point of view "environment" targeted by DOM, we focalized ourselves this time on the agent and its intrinsic behaviors. We structured our approach on the duality : conceptual model and computational model. Indeed, we ended on two documents that describe the conceptual model (how the experts think turtles should be modeled) and the computational model (the model as it is or will be implemented). The more we will reduce the gap between these two documents (implementation gap), the more relevant our prototype will be.

2) Feedback: This was the first step to the elaboration of a more mature conceptual model, which can be afterwards implemented on a more powerful and efficient simulation platform. Moreover, we also realize that during the modeling, one should be aware of the reusability of its model.

The experience earned after this methodology (modeling and prototyping) showed us that the DOM methodology, based on the environment splitting, was not enough to fully apprehend Modelization of complex system. Indeed, DOM does not take into account the agents’ behaviors, which is a critical point if we want to improve the reusability of our model. Thus, we also need to consider a methodology for the agents’ modeling, especially its behaviors. Indeed, each expert has different ways of modeling the turtles according to its interactions with the wind, the surface temperature, the stream, etc.

The increase of the model’s complexity applies to both agents and environments. This DOM approach tackles the problem of the latter with a method of organization, but not the former. The more advanced is the evolution of the model, the more complex will become the agents. Eventually, they will be more and more hard to comprehend in the whole model.

We then have to ponder over an new Modelization, more oriented toward agents and more specifically towards behaviors, because its the essence of any agents.

IV. MULTI-BEHAVIORS MODELIZATION

A. Problematic

Whether we used DOM or not, in a general way, the behaviors of an agent is still hard to apprehend in a complex system, because each expert is able to have a precise angle of view (we will call it "a Realm") on a specific character of the agent’s behaviors and could be distracted by other unrelated elements.

To make understanding easier, let’s define a small case study3: A Multi-Agent Simulation which takes care of the simulation and geo-localization of energy flow (production and consumption) and its cost (both its price and the social cost of carbon related to it) while reckoning with modeled interactions, in order to provide a decision-support tool for

3Another study case related to one of our project will be explained further (Section IV-C).
the energy planning within a territory, thanks to the different possible scenarios, e.g. in anticipation of new power plant and its sizing.

In this study case, an expert in energy won’t be able to grasp the agents’ behaviors if it implements Energy, Finance and Carbon Emission Control at the same time. He will be scared of collapsing the whole system if it has been misimplemented.

This issue has been almost bypassed by using the previous DOM method [1]. Each agent in the “world” may have a set of states and a set of behaviors. We had partitioned the agent itself into dynamics where each dynamic is a subset of states and a subset of behaviors related to a layer of environment. If we are going back to our study case, there will be a dynamic related to the Energy, one to Finance and the last one to Carbon Emission Control. In the context of a spatial [6], [7] and highly communicating [8] MAS, we also can define two more dynamics for its position and its communication.

We call each submodel a Mono-Dynamic Model (MDM) and the DOM itself will be a multi-MDM Model, a layer-structured agent model where layers use the environment as a place to share information.

But this Dynamic Oriented Modeling would not suffice if we want to improve the reusability of our agent modeling. For example, if two experts in Energy hold different ways of express the agent’s behaviors regarding the energy dynamic [9]: the first one started using a bottom-up statistical techniques for residential houses. After a while, the second expert comes and wants to use a top-down technological techniques for factories. This latter is not meant to replace the former, but it has been build to improve the first one with further consideration. We need to go further in the partitioning. This lead us to a Multi-Behaviors Modelization on top of the Dynamic Oriented Modeling, in order to help these kinds of refinement in behaviors.

B. Introducing MVC

1) Design Pattern: Christopher Alexander has introduce design patterns in 1977-1979 in the architectural field, but its use in software development has started in 1987 [10] and picked up fame in 1994 when the book [11] of Erich Gamma, Richard Steerage, Ralph Johnson, and John Vlissides, known as the “Gang of Four” or simply “GoF”. Design patterns empower reusability and might be utilized as “building blocks” for complex application. Numerous researches towards the reuse of model have been made in diverse fields such as Software Engineering but also in Artificial Intelligence [12], etc.

2) MVC in Software Engineering and its versions: For example, in Software Engineering field, the Model-View-Controller paradigm [13]. [14] has been wide-spreaded and furnished interesting feedbacks to the Software Engineering research community. MVC Pattern was designed for and implemented in Smalltalk when User Interface libraries did not exist. In order to understand MVC, we could say:

• A “view” is a class that is responsible for representing the part of the User Interface. It should draw it, e.g a box, a text, a selected areas, etc.
• A “controller” is another class that interacts between user and view. It took mouse events occurring within this very same view (e.g. the box or the text) like mouse events, keyboard events, etc.
• A ”model” is yet another class that represents the basic data and state of the component. For example, a text box model would have the text of course, the box’s design, the selection, etc.

In a situation like that, the original three components are very entangled, which was the reason why it has been called ”triad”. The full ”box” or ”text” would be composed by three components.

As explained in [13], there are two variations of MVC: a passive model and an active model. In the passive model, the controller exclusively manipulates the model and manages the synchronization between view and model. On the other hand, in the active model, the model is independent from the controller who is here to warn the view(s) that changes has been occurred.

After years of using, testing and refining, the application of the ”MVC” pattern has spread beyond its initial purpose. Nowadays, a ”view” could actually be a complete dialog (not to say the full UI), and a ”controller” for the (whole) application that’s respond to most events. The ”model” is usually linked to the view by providing an observer interface.

From the original MVC pattern, other design pattern also emerges like MVP (Model View Presenter) or MVVM (Model View View Model). Sometimes, the MVC pattern has been adapted to a specific language such as the Cocoa version of MVC as a compound pattern

In this section, we just wanted to show that whenever a specific pattern, with lots of potential, such as MVC, has been wide-spreaded in the community, it gave us back informations that help in improving its use.

3) MVC in MAS: Since [12], numerous works have been done as resumed by [15], but the vast majority of researches has been centered on patterns for agent-oriented software or for the agent’s interaction. Also, whenever the suggestion is a pattern-based design methodology, the proposed pattern is frequently homemade or specific to one particular domain.

In order to boost the benefits of design patterns, one specific pattern should be used uniformly throughout the MAS research community. It would result in both spreading MAS solutions and giving valuable feedback to the MAS research community, and that will help in refining this very pattern.

It’s important to note that we are not talking about the MAS software’s architecture (as we can see in [16], [17], in [18] or in [19]) but about the agent’s architecture itself. Applying MVC software’s architecture would ”simply” consist in using MVC in the context of software engineering. Here, we want to import the MVC methodology into the world of MAS and use it at as a design methodology for the agent.
Moverovern, the Multi-agent System (and the Agent-Based Simulation) have specificities that are different from the OOP’s world.

In OOP, most of interactions are predetermined. Whenever an action occurs, there will be a reaction from the application. It is a deterministic world. On the other hand, in MAS’s world, we tend to a nondeterminism: agents would try to do an influence on the environment or another agent directly (or through the environment again, according to some points of view). This influence could be successful or not. In the same way, agents would try to do a perception, but it also could be successful or not. This is one major specificity.

This is why we can tell that we are not using MVC, but an adaptation of MVC in the agent world: agentMVC (or aMVC). From this point of the article, the word aMVC will be used only if there is any confusion possible.

C. The MVC agent

After this small digression, we are coming back to the initial subject: the Multi-Behaviors Modelization, to help of refinement in behaviors in SOA.

We will propose here a new agent architecture, based on this previous stated MVC, in order to be able to benefit most of the potential behind the widely-used MVC pattern.

We have then to define identity of each concept in our current model: The model, the view and the controller.

In order to ease the identification, we would not only split the environment such as in Section I-A by means of dynamics but the agent themselves via “realm”. The word “realm” means area of behavioral expertise. Each set (states and behaviors) can be split according to the “realm” to which they are referring (i.e. in the Figure 3a the A realm, the B realm, the C realm where C is a bit related to B, and the D realm with two different behaviors).

The “inside” of the agent in the Figure 3a show us a certain amount of subset (that could be really huge according to the complexity of the system) that are linked together. Our goal is precisely to organize everything to ease its conception.

The difference between realm and dynamic is that a dynamic can be composed by one or more realms. For example, the ”social”s realm and the ”position”s realm is part of the ”communication”s dynamic.

As we can see in the Figure 3b, we will define one MVC architecture for each realm. In this figure, the relation between C & B and between D and its two behaviors has not been represented.

In this new approach, we are taking an extra step in our initial DOM partition, and we will characterize an agent by three components (as we can see in Figure 3a):

- the agent’s state, which contains its attributes (in boxes).
- the agent’s behaviors, which organizes all the actions it can undertake, also called decisional process (in circles).
- the agent’s interactor, which allow interaction (influence and perception) with the environment (blue arrows).

Study case:

Let’s take a study case related to our project EDMMAS: if we want to implement a simulation that includes at the same time the evolution of the population, the production-consumption of energy and the land-use (urban, agricultural or natural) changes on a territory such as in EDMMAS [2], we would face hundreds of thousands of Parcels agents (land units of approximately one hectare). Complexity of the system will come from both the interaction’s complexity and their numbers. Here is a brief description of the system through the
Multi-Behaviors Modelization:

- **Example of dynamics:** In order to model this system, we would divide the problem in three coupled dynamics: a population dynamic, a land-use dynamic and an energy dynamic. For example, production of energy (by power plants or individual solar panel), its distribution and its consumption (by residential or factory) are activities that participate to the dynamic of energy evolution. Each dynamic is a MDM, and the three dynamics together will constitute the multi-MDM model, also called the DOM model.

- **Example of realms in one dynamic:** Now, if we consider only the dynamic of "energy evolution", it is composed by three realms: production of energy by plants, consumption of energy by residential houses and consumption by factories.

- **Example of components in one realm:** In the realm "production of energy by plants", we would consider:
  - **States:** which kind of energy production it is (Fossil Fuels, Hydroelectric, Nuclear, Solar, etc.), its scalability, its pollution’s impact, etc.
  - **Behaviors:** produce energy, stock energy, send energy, negotiate with another plant, etc.
  - **Interactors:** its connexion with the electrical power grid or with a grid energy storage.

Now we will be able to define the **model**, the **view** and the **controller** but we will consider only one realm, bearing in mind that an agent will be split according to several dynamics and each dynamic according to several realms. The Figure 4 is the summary of our definition.

1) **Defining the model for one realm:** We usually tag **model** only as a database in Software Engineering; but the model in MVC is both the data and the domain logic needed to manipulate the data. In this agent’s decomposition, the states’ collection mixed with internal evolution laws related to the realm (such as ageing of an agent) should be the **model**. This subset of dynamical data (states of agents) evolves with time due to behaviors and internals laws of the realms (that make the consistency of the data).

As mentioned before, in [13], there are two variations of MVC: a passive model and an active model.

In our context, we should use a passive model where the controller exclusively manipulates the model and manages the synchronization between view and model. The reason is that the "Agent World" is different from the "Object World": while in Object-Oriented Programming (OOP) most of interaction are predetermined and are similar to action-reaction (such as "a click on this will do that thing"), in the MAS most of the interaction would be influence-perception (such as "this agent will try to influence that agent/environment").

2) **Defining the controller for one realm:** The **controller** is composed by the behaviors. Indeed, in an agent, the behaviors "controls and modify" the states. Moreover, the behaviors manage the agent’s interaction with the environment and, for this, use its states; either in order to consult the states to take a decision or to influence its modification in order to memorize any experience learnings.

If necessary, we may have multiple **controllers** for the very same **model** or several **models** used by one **controller**.

3) **Defining the view for one realm:** If we get back to definitions in section IV-B2, we will understand that a **view** allows the program to perceive from the outside (external informations, e.g. like a button or a checkbox) and to influence the outside (external users, e.g. like a colored gauge that will mean something to the user). In the same way, an agent has to do influence and perception with other agents and/or environment.

Thus, we can define the **view** of an agent is then the agent’s interactors (which allow influence and perception) with other agents and/or the environment in a particular realm.

D. From Basic agent to MVC agent

After this proposition of new architecture, we now have to split our basic agent in order to be able to fit this proposition. The way to do it will be as follows (some of the steps has been mentioned previously):
• Splitting of the agent according to dynamics: such as explained in section I-A, when we say "splitting of the agent according to dynamics", we mean environment splitting reflected on the agents.

• Splitting of the agent according to realms: in section IV-C we have defined what is a "realm" and why we should split according to it.

• Splitting of the agent according to its three components: also in section IV-C we have explained that an agent can be identified by its states, its behaviors and its interactors.

1) Effects of this splitting: The first two slicings will allow us to separate different fields in a SOA. As we have seen in ED-MMAS (Section II), separating those field (energy, population evolution, land-use evolution) make easier understanding of the application by all kind of experts. The "realm"’s splitting will make the "dynamic"’s one more precise, and allow us to categorize each kind of expertise needed in this SOA.

   a) "Outside" the SOA: Indeed, after this splitting according to the three components (or we could also say according to MVC), when would be able to share the difficulties related to the Modelization of one specific realm amongst a panel up to three experts.

   For example, the first one would focus on the model of the power-plants’ energy production while another will have to take care of the plants’ interaction with the energy flow in the electric power distribution and the last one to its characteristic (capacity, output, etc.).

   Moreover, as mentioned in IV-A, if two experts in Energy hold different ways of express the agent’s behaviors regarding the energy dynamic [9], we would be able to implement two models for the same realm and switch between both whenever needs arise.

   b) "Inside" the SOA: As we saw “inside” the agent in the Figure 3a, and the definition of the agent MVC, we are splitting according to three components. Our goal being to organize everything and to ease its conception, we will share the SOA by regrouping components together for each realm:

   • One or more layers\(^3\) of behaviors (LB), which consists in definition of agents behaviors related to one realm. This is the controller. The advantage of this technique is that usually in a complex system, there is a lots of agents that can be categorized by "kind".

   Each "kind" will be defined by the same set of behaviors. By taking behaviors away from the agent’s state, we are then able to factorize behaviors and reduce the complexity of the model.

   • One or more layers of physiognomies (LP), which express the "character" or "personality" of the agents (its states and some internal laws related to this realm). We used the word "physiognomy" instead of the "body" to set us free from the capacity of interaction here (which is usually associated to the body). This is the model.

• One or more couple of interactors (Influence and Perception) for each agent in relation with each dynamic. This is the view.

From this example of the Figure 3a, we then can also split as follows (in the Figure 5a):

• 4 Layers of physiognomies LP\(_A\), LP\(_B\), LP\(_C\), and LP\(_D\).

• 5 Layers of behaviors LB\(_A\), LB\(_B\), LB\(_C\), LB\(_D\(_1\) and LB\(_D\(_2\).

• 10 Interactors for each agent: 5 for Influence and 5 for Perception.

In this example, if we supposed that the splitting will result into three dynamics (Dyn\(_{AB}\), Dyn\(_{C}\) and Dyn\(_{D}\)), their relation will be like in Figure 5a. In this Figure, we choose to simplify by showing the splitting of only one agent and we can see the different layers of the system, illustrated with the splitting of one agent. If we have hundreds of agents of the same kind, the same layers will be shared amongst them. In fact, every agent will be split by the same realm and sent to the adequate layers. It could be represented as in Figure 5b.

V. CONCLUSION

In this article, we first studied an example of creation and reuse of model through the DOM Modelization. We then showed another case in order to shed light on the fact that during the modeling, one should be aware of the reusability of its model. We also put forward some problem due to model’s reusing and extending, especially in term of agent’s Modelization, and show how it quickly takes the upper hand against the benefits of DOM, as it can eventually block the potential development, or even reuse, of the model.

Lastly, we proposed a new Modelization Multi-Behavior inspired from the MVC pattern, in order to improve this issue. This new agent architecture is a structure capable of maintaining readability and flexibility of the formalization of the agent throughout the growth process of the global model. This Modelization starts from the bottom (the agent) and not from the top (the system). After cutting the environment in dynamics with DOM, we sliced the agent into a "mille-feuille" (where a layer is a realm) and then again according to its three components: physiognomy, behaviors and interactors.

Due to this splitting we would be able to give the layer to any experts, and if necessary divide the work among different experts thanks to the MVC slicing by giving any part of the layer (M, V or C). This Modelization will ease the creation of agents but also its reuse.

Additionally using the MVC pattern’s property, among other advantages, we would be able to facilitate the reuse as well as the customization of any part of an agent.

This approach allows us to perceive a new field of investigation, particularly in a global level of layers’ organization and the potential dynamic evolution of its interconnections. The most interesting part will be related to this MVC adaptation to the agent world: Will MVC and aMVC have the same similarities? Would we be able to plug one new V (view) of an agent to an already present M (model) ?

The study of this field will be the subject of further researches.

\(^3\)Each layer will contains "splits" of several agents.
REFERENCES


