AN AGENT INTEGRATION INFRASTRUCTURE FOR THE DEVELOPMENT OF ENVIRONMENTAL DECISION SUPPORT SYSTEMS BASED ON SIMULATION

Bernard ESPINASSE, Julien SERMENT, Erwan TRANVOUEZ

Laboratoire des Sciences de l'Information et des Systèmes, LSIS UMR CNRS 6168,
Universités Aix-Marseille, Marseille, France.

{firstname.lastname}@lsis.org

KEYWORD
Environmental Decision Support Systems, integration infrastructure, environmental decision support, multi-agent systems.

ABSTRACT

The environmental dynamics complexity makes more and more difficult the decision-making process in the environment domain. Therefore, this process needs to be supported by powerful computerized tools, called Environmental Decision Support Systems (EDSS). This research tries to propose software solutions to facilitate development of such systems. Searching for a generic architecture of EDSS based on simulation, first of all, this paper identifies the main functional requirements of decision makers from such systems, and then defines the main functionalities of functional software modules needed. Du to the variety of implementation of these functional modules, such architecture is difficult to define. Consequently, inspired by HLA (High Level Architecture) and FIPA specifications, an agent based generic infrastructure is proposed in order to integrate such functional modules to constitute a specific EDSS. The various components of this generic infrastructure, currently in development, are presented in detail.

1. INTRODUCTION

The environmental dynamics complexity makes more and more difficult the decision-making process in the environment domain. This complexity overloads decision-makers in quantity of data, information and knowledge of different form and quality. The environmental decision-making process, in sustainable development, in water management, or ecosystems management more or less human-influenced is increasingly complex.

Therefore, this process needs to be supported by powerful computerized tools, denoted Environmental Decision Support Systems (EDSS) (Jansen 1992). These systems have to assist the decision-makers and do not substitute them. They aim to reduce the duration of the decision-making process as well as the consistency and the quality of the decisions (Cortés and al. 2000). Such systems can be used to understand and manage an ecosystem, to accumulate qualitative and quantitative information, to adapt conceptual models to a local management, or to select appropriate management options to optimize the (often conflicting) decision criteria (Mowrer 1997).

This research tries to propose software solutions to make easier the development of such systems, mainly in proposing a generic software architecture, permitting to reduce the development time of such systems and permit reuse of existing software modules.

This paper is organized in 5 sections including this one. Section 2 briefly presents a conceptual framework based on different decision levels that is then used to study some existing relevant EDSS. Based on this study, the major functional requirements that EDSS must abide by, are identified. Section 3 presents the functionalities of software modules supporting these functional requirement, and interactions between these modules. Due to the variety of implementation of these functional modules, such architecture is difficult to define. Therefore, in the section 4, is presented an agent based generic infrastructure, inspired of HLA (High Level Architecture), and permitting to integrate such functional modules to constitute a specific EDSS. The various components of this infrastructure are presented software agents as the integration principle used (main integration services supplied by the infrastructure. Finally, in section 5, conclusions are drawn and future directions are proposed for the work.

2. FUNCTIONAL REQUIREMENTS IN ENVIRONMENTAL DECISION

Environmental decision-making can take different forms, according to the nature of the decision, the decisional context, or the characteristics of the problem the decision-makers are encountering.

In this section, first is briefly presented a conceptual framework based on different decision levels. Then this framework is used to study and compare some relevant existing EDSS projects. Afterwards, based on this comparative study, are identified the major functional requirement that have to be supplied by EDSS.

A comparative study of these EDSS projects on these different aspects is quite difficult. For each of these projects of EDSS, its objectives, its functionalities and its software architecture are specific.

B. Espinasse, J. Serment, E. Tranvouez (2007), « An Agent Integration Infrastructure for the Development of Environmental Decision Support Systems based on Simulation », IMSM07 (International Modeling and Simulation Multiconference) - CMS (Conceptual Modelling and Simulation) Conference, Buenos Aires, Argentine, February 8-10, 2007.

Conceptual framework

An EDSS is developed in a specific environmental decisional context, which can be specified according to:

- The nature of the decision (ecological, political, socio-economical …).
- The decisional objectives (Denzer 1997) (prediction, medium/long term planning, supervision and control, emergency crisis' management …).
- The characteristics of the problem decision-makers face (natural resources, natural park, forest fires …).
- The temporal constraints of the decision: short, medium or long term.
- The impact of the decisions: local or global.
- The modelling paradigm adopted: analytical models, stochastic models, individual centred models, object models…

In general, the environmental decision comes within the scope of a management activity, for instance of an ecosystem. In the management field, a management activities typology has been proposed by Anthony (Anthony 1966). Three types of activity are distinguished: the strategic planning, the management planning and control, and finally the operational control. To each type of activity corresponds a specific type of problems to solve, and consequently a specific kind of decision. An EDSS can be used on one or several of these levels:

- *Strategic planning*. It is a managerial activity that leads to major decisions with global impact and long term consequences (eg decisions involving the future of a whole ecosystem).
- *Management planning and control*. In general, this managerial activity leads to decisions with medium term consequences and intermediary impact between local and global.
- *Operational control*. This activity concerns decision with short term consequences (less than the month) and a local and limited impact (eg a small area of an ecosystem).

Faced with this diversity of approaches, the definition of different usage levels or decisional levels appears relevant to present an unified view of environmental projects.

Study of some existing EDSS

In order to better define the different forms of the decision support that can be supply by EDSS, and to define the major generic functionalities of such systems, several relevant EDSS have been studied among them, the following representative cases are underlined in this paper:

- the **BIOMAS** project (Courdier 2002) deals with practices in collective management of animal wastes,
- the **CATCHSCAPE** project (Becu and al. 2003) relates shared natural resources management,
- the **PROBIO** project (Botequilha and al. 2001) concerns by the biodiversit forecast and management in protected area,
- the **NED** project (Nute 2003) studies a forest ecosystem management situation,
- the project **ALI** (Jaber 1999) for the prevention and the struggle against forest fire,
- the **WWTP** project (Borrell and al. 2002) concerns control and management of used water treatment plants.

In general, a particular decision level is favoured. For example, the BIOMAS and CATCHSCAPE projects mainly concern strategic planning. In the two projects, the aim is not to find an immediate solution to a specific problem, but to provoke (with the stakeholders) a reflection on a process involving a whole ecosystem (stakeholders included). These projects are dedicated to these actors in order to make them realize, for example, they are part of this ecosystem, and that each of their acts has consequences. This can leads to collective decisions or progressive changes of collective behaviours with consequences in the long term.

The PROBIO and NED projects concern mainly management planning and control decisions. These projects support planning in medium or long terms and concern management means to maintain ecosystems in a stable state.

The ALI and WWTP project mainly support operational control decisions, in supervision and control. These two projects concern problems (fires, used water) where the decision-makers have to quickly react and with immediate consequences (short or very short term). Solutions to these problems affect very limited areas.

Functional requirements to support by EDSS

The study of these EDSS projects has permitted to define the following major functional requirements concerning the three decisional levels previously introduced:

- **Modelling requirement**. The decision-makers have to set the problem with models. This requirement concerns the description of entities implicated in the various models used (ecological, biological, economic …). Some of these models are simulation models and specify the behaviours of such entities. Several supplementary models can be used (multi-modelling), and theirs interactions (or couplings) have to be specified.
- **Simulation requirement**. The simulation is often used in EDSS, according to different perspectives

(Reynols and al. 1999) prediction, understanding, extrapolation or interpolation. The simulation requirement implies the design of scenarios and experiment plans related to the simulation models, the control and the supervision of the simulation execution, and sometimes the synchronisation of different simulation model executions on different software environments.

- **Visualization requirement**. This requirement allows the visualization of models and results obtained from these models especially by simulation. It improves results' interpretation and communication between different decision-makers. In EDSS, the models are often spatiallized, and such visualization is based on a defined spatial referential.
- **Analysis requirement**. Different alternatives having been simulated, synthetised evaluation is needed. For example, this requirement of the user can concern the analysis of simulation results, or to compare results of different simulations.
- **Data management requirement**. This requirement concerns the management of exogenous or endogenous data used by the previous functionalities. This implies storing data, especially spatial and temporal data, relative to the different models and managing/controlling their use, in particular for simulation. This requirement manages also data generated by simulation (intermediary and final results). The data management requirement is closely tied to the other requirements.
- **User integration requirement**. This requirement supports the user-system interaction by providing the user transparent access to the others functionalities. The user integration can be extended to his participation to the modelling process (roles game, participative approaches …), or in the simulation steering. This requirement is also closely tied with other requirement.

The two last functional requirements can be considered as auxiliary requirement. In the studied EDSS projects, the relative importance of the four first requirements depends on the decisional level concerned. The figure 1 illustrates this importance according the three decision levels for the Modelling (M), Simulation (S), Visualization (V) and Analysis (A) functionalities.

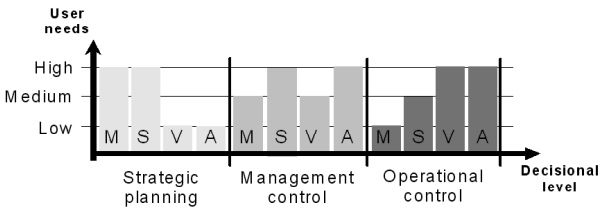

Fig. 1. Functional requirements versus decision level

The BIOMAS and CATCHSCAPE projects, which mainly involve the strategic planning level, focus on the understanding and the interaction of models. These tools should permit the user (decision-maker) to modify himself the models, and to intervene during the simulation process.

These requirements are true at the management planning and control decision level, as in the PROBIO and NED projects. In these systems, the user can elaborate and evaluate numerous of alternatives. The simulation and analyse requirement are the more used by the decision-makers.

Finally, in the ALI and WWP projects, which are mainly related to the operational control decision level, the models needs not to be modified, but the simulation results have to be easily and quickly analysed. Analysis and Visualization requirement are very important for the user to follow and exploit the simulation execution.

3. FUNCTIONAL ARCHITECTURE OF EDSS

Each of these six main requirements can be supported by a specific software module (functional module). A generic functional architecture of EDSS could be composed of six software modules in interaction, namely: (i) a modelling module, (ii) a simulation module, (iii) a visualisation module, (iv) an analysis module, (v) an user integration module, and finally a data management module.

In the following section, each of the functional module is defined by its basic functions, and the specific data it manipulates. Then the main interactions between these fuctional modules are specified.

Modelling Module

The aim of this module is to elaborate models. Models mainly concerned in this paper are simulation models. These models can be multiagents models, object models, cells automatas or others simulation models. This module permits the description of the entities composing the model, their behaviours, and their interactions. It allows describing the interactions between modeles (coupling). This module can be based on a metamodel as in the MIMOSA project [Muller 2004].

In this module, different approaches are used for modelling: cells automats, object, reactive agents, cognitive agents, hybrid agents … Cognitives agents are mainly used for the representation of social process (negociation, cooperation …). They can also enable a participative approach (Bousquet and Le Page 2004). The reactive agents are mainly used in the representation of emergent phenomenoms et are particularly adapted to the representation of behaviours of individus in interaction. The hybrid agents can be used (Andriamasinoro 2003) to take into account the reactive behaviour of cognitive agents. The object approach is in general used for the representation of entities with simpler behaviours or for analytic models. This module has also to permit the user to use some popular notations of specification as UML,

AUML or DEVS.

Simulation Module

This module plays a central role in the decision support process; it permits to test different possible scenarios, solutions or alternatives. It has to perform the execution of simulation models and their coordination.

Various simulation environments can be concerned by this module. For example, agent-based models can either be translated into objects models which in turn are executed, or conversely directly simulated through agent-oriented simulator such as MatKit (Ferber et al., 2000) or Cormas (Becu et al., 2003).

Visualisation module

This module focuses on the visualisation of models, their behaviours and simulation results. It permits to supervise the simulation, to display intermediary and final simulation results with adapted graphical representation In order to ease the communication with users, and make easier results interpretations.

This module can be composed of various visualisation tools adapted to specifica data, such as Geotools for example for geographical data (Geotools 2006). The Object Oriented programming is largely used, but agent programming begin to be used (interface agents). One advantage of this agent programming is that the visualisation can be directly linked to the simulated entities (Campos 2000), permitting a better follow-up of the simulation. The models agents can also provide themselves an interpretation of the state of the system (Servat and al. 1998).

Analyse Module

This module allows the user to analyse results (simulation results) and to compare different alternatives computed sometimes according to various criterions. This module can be implemented according to differents technologies indifferently based on objects or agents oriented programming.

User integration Module

This module gives the user access to the different functions of the EDSS, functions which are integrated in the modules previously described. It is a cross module. This module is not dependent on a particular programming technology. The agent oriented approach, with assistant agents, makes possible to integrate some intelligent functions assisting the user/EDSS interaction, resuting in more adaptative interactions.

Data Management Module

This module takes in charge the data management and as the previous module is a cross module. Its role is to store data of different type, in particular spatial and temporal data, and make it available to other modules. These data can be related to the models, scenarios, or the simulations from which they can be generated (intermediate and final simulation results). This module can be closely linked with the visalisation module. This module can be implemented according to differents technologies, as objects or agents oriented programming. Agents can be a relevant technology to assume, for example, a coupling with GIS (Geographic Information Systems) as proposed in (Maillé and Espinasse, 2005).

The table 1 summarizes the main functions of each functional module of the EDSS generic architecture. For each module specific input and output data are also mentioned.

Modules	Main function assumed	Data
Modelling	*Model elaboration* *Define interactions between models (coupling)* *...*	*Model description* *Coupling description* *...*
Simulation	*Define simulation parametrer* *Manage the simulation* *Take place during the simulation* *Define intermediary and final results to store* *Define a scenario of simulation* *Define an experience plan* *...*	*Model description* *Coupling description* *Initial values* *Simulation scenarios* *Simulation results* *...*
Visualisation	*Display models* *Define visualisation parametrer* *Display the follow up of the simulation* *Display the simulation results* *Display the analyse results* *...*	*Geographic data* *Model description* *Simulation results* *Analyse results* *Visualisation parametrers* *...*
Analyse	*Analyse simulation results* *Analyse experience plan* *...*	*Simulation results* *Analyse results* *...*
User	*Acces to modelling functions* *Acces to simulation functions* *Acces to visualisation functions* *Acces to analyse functions* *Acces to data management functions* *...*	
Data management	*Store a data* *Consult a data* *Modify a data* *Delete a data* *Take out a subscription to a data* *...*	*Model description* *Coupling description* *Initial values* *Simulation scenarios* *Simulation results* *Geographic data* *Analyse results* *Visualisation parametrer,* *...*

Table 1: Main functions and data of functional modules

Interactions between modules

The modules interaction matrix in Table 2 highlights the importante interactions between functional modules of the EDSS architecture. They somehow echo the following sequence of action.

The modelling module (M) produces and provides to others modules the models and coupling decriptions. The simulation module (S) uses these informations to execute the simulation models and produces the simulation results. The visualisation module (V) then receives information from all the others modules to visualize. The analyse module (A) produces analyse results from the simulation results, which are also used by the User integration module (U) and the data management module (D). The module U permits to the user to invoque others modules. Finally, the data management module receives information, and provides information to all the other modules of the system.

⇗	M	S	V	A	U	D
M		X	X		X	X
S			X	X	X	X
V					X	X
A			X		X	X
U	X	X	X	X		X
D	X	X	X	X	X	

Table 2. Modules interactions matrix

From a generic architecture to an integration infrastructure

The proposition of a generic functional architecture for EDSS design appears difficult to define. The main reason is that various software solutions can be choosen for each fuctional module. Moreover, the importance of interactions between the software modules composing the EDSS needs interoperability facilities.

Consequently, a generic software architecture dedicate to EDSS has to propose a specific software module supporting these interoperability facilities. This specific module, named an integration infrastructure, has a central place in this generic architecture as illustrated in the Figure 3.

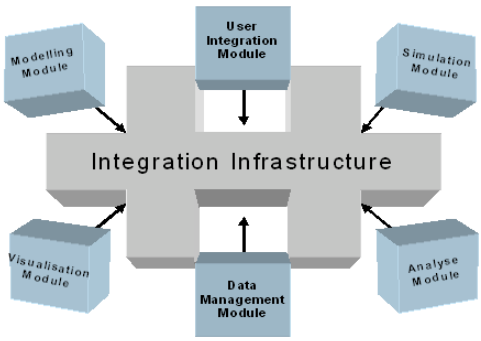

Figure 3: Functional modules integration

Such an infrastructure should integrate existing software solutions for each of the different functional modules, and to support theirs interactions. Next section presents a generic architecture based on an integration infrastructure complying with these requirements.

4. AN AGENTS BASED INTEGRATION INFRASTRUCTURE

The "functional" modules introduced in the preceding section are largely interacting with each other. Each of these modules provides and uses information and services to/from other modules. Interactions may occur in the module itself as for example in the case of coupled simulation. The user integration and data management module are transverse to all the other modules.

The integration infrastructure has to permit integration and interoperability of software module (functional modules) to fulfill the functions required for decision-making support in the specific context of EDSS design. Two levels of integration services can be distinguished: services for decision-making integration related to the user requirement in the decision-making process (functional integration), and services for software tools integration and interoperability (software integration). The first integration level is assumed by the functional modules themselves. The second integration level is performed by such an integration infrastructure.

The agent-based integration infrastructure proposed in this paper is mainly inspired on one hand of HLA (High Level Architecture) (DMSO-HLA 1996) (IEEE-HLA 2000). and on the other hand by the FIPA recommendations about agent software integration (FIPA 2000).

First, this section presents the general architecture of the integration infrastructure proposed. Then the agent oriented approach to implement this infrastructure is argued. Then the agent architecture of this integration infrastructure is presented. Each of the agents composing this architecture is presented in details, and the general integration principle is illustrated. Finally some implementation aspects are detailed.

General architecture

The general architecture of the integration infrastrcture proposed is inspired of HLA (High Level Architecture) (DMSO-HLA 1996)(IEEE-HLA 2000).

HLA, standard for distributed simulation, is a software architecture and also an interface specification. The interoperability between several simulators is mainly provided in HLA by a RTI (Run Time Infrastructure) component. This RTI component is in relation with a federed component composed of the simulator and a specific ambassador interface (federed ambassador).

Inspired of HLA, the general architecture of this integration infrastrcture is composed of three main components: Wrappers, interfaces and a mediator, as illustred in the Figure 4 (Serment and al. 2006).

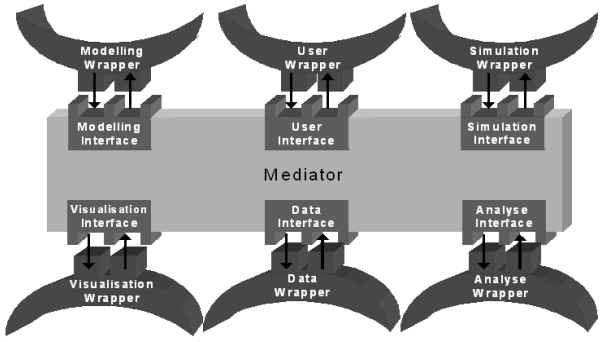

Figure 4: Integration infrastructure architecture

- **Wrappers component**: each wrapper encapsulates a specific functional software module and integrates it in the EDSS.
- **Interfaces component**: allow to connect dedicated wrappers to the mediator.
- **A Mediator component**: responsible of the interoperability of the software tools.

An agent oriented approach

For the implementation of this integration infrastructure, an agent-oriented approach has been adopted to develop the different components of this infrastructure. To provide software integration solutions, the agent-oriented approach has already been used.

First in the distributed simulation domain, the agent oriented approach has been already successfully used to solve integration problems. For example, the GRIDS infrastructure (Generic Runtime Infrastructure for Distributed Simulation) is an agent based implementation of HLA without physical RTI (Taylor and Sudra 2002). ARTI (Tan and al. 2000) is another HLA agent based implementation, characterized by a distribution of a RTI Agent on the ambassador of the federated component. Other agents based solutions for integration and interoperability can be mentioned as (Prasithsangaree and al. 2004) (Giampapa & al. 2004) or (Wilson and al. 2000) (Wang and al. 2005).

Then in the software agent domain, the FIPA (Foundation for Intelligent Physical Agents) dedicated to promoting to the industry the agent technology has defined specifications to support interoperability among agents and agent based (or not) software applications. A specific specification concerns how software resources can be described, shared and dynamiquely controlled in an agent community (FIFA 2000). This recommendation provides important normative statements suggesting ways by which agents may connect to software via specialized agents called "Wrapper" and "Broker".

In conclusion, the agent approach appears a really relevant approach to solve integration and interoperability problems. Agent approach is adapted to the design of middleware to integrate heterogenous softwares. Solutions obtained are flexible and adaptative, due to the autonomy of the agents, and also theirs ability to exploit domain knowledge. Consequently, to implement the integration infrastructure proposed, this agents oriented approach has been adopted.

Agent based architecture proposed

In the agent-based infrastructure proposed, the Wrapper component of the previous architecture is an agent that encapsulates the functional module of the EDSS. The Interface component is also an agent that assumes the communication between the Wrapper component and the mediator. The mediator, in charge of the interoperability, is an agent itself a multi-agent system composed of specialized agents. The following figure illustrates the agents based architecture of the proposed intregration infrastructure.

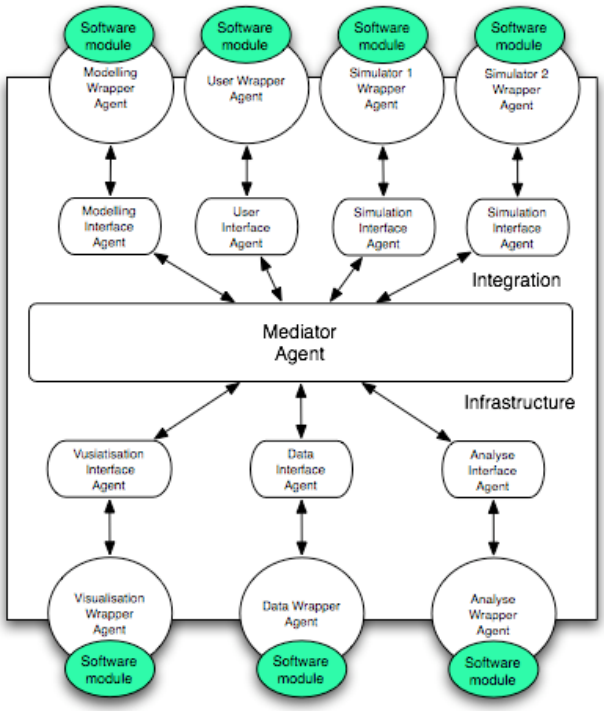

Figure 5. Infrastructure agent-based architecture

A FIPA specification on Agent Software Integration, has defined [FIPA 2000] how software resources can be described, shared and dynamiquely controlled in an agent community. The purpose of this specification is twofold: it allows agents to describe, broker and negotiate over software systems, and it allows new software services to be dynamically introduced into an agent community.

This FIPA specification is based on dedicated agents to the integration problem. Wrapper Agent assumes the connection between the software to integrate with the agent integration agent community thanks to a wrapper ontology and a software dynamic registration mechanism. For this purpose, another agent, the Agent Resource Broker (ARB) service, is defined. It allows advertisement of non-agent services in the agent domain and management of their use by other agents, such as negotiation of parameters (e.g. cost and priority), authentication and permission.

The agents of the infrastructure

Combining and adapting HLA and FIPA approach to interoperability, the agents based infrastructure constitutes an agents community composed of three main kinds of agents:

- **Wrapper agent**. is an agent able to connect to a software system uniquely identified by a software description. The role of the Wrapper agent is mainly to provide a single generic way for **other** agents to interact with software systems.
- The **Interface Agent** is associated to a particular type Wrapper agent, represents a particular functional module encapsulated to the mediator agent, and assumes the interactions between the Wrapper and Mediator agents. Therefore, there are as many as interface agent as the functional module in a EDSS.
- The **Mediator agent** insures interoperability by coordinating the dialogues between Interfaces agents, and providing other services such as data and clock synchronisation. This agent is composed of specialized agents.

These three integration agents are described in the following sections.

The **Wrapper agent** is an agent that can dynamically interface with a software system uniquely described by a software description. The Wrapper agent will allow, directly or indirectly, others infrastructure agents to invoke commands on the underlying software system, translating the commands contained in ACL messages into operations on the underlying software system. Wrapper agents may be able to support multiple connections to software systems simultaneously.

A Wrapper agent that supports the full FIPA-Wrapper ontology is considered to provide more than a simple bridging function to an external software system. Such an agent implicitly provides a management functionality. A Wrapper agent supports the FIPA-Wrapper ontology with commands and predicates for initialising and issuing requests to software systems.

The integration services provided by the Wrapper Agent are:

- Invoke a data or function of the software module encapsulated.
- Assumes communications with the corresponding Interface Agent.
- Translation of the Interface Agent requests into data or service calls of the encapsulated module.
- Translation of the replies to the others functional module to the interface component.

A Wrapper agent is already implemented and what interface exists between the Wrapper agent and the underlying software system that provides the software service is a matter not considered in this paper.
Wrapper agents can range from agent simply accessing SQL databases using for instance CORBA ORBs or it could be a more general Wrapper agent, which supports dynamic connection to any system, which has been registered with the ORB's Implementation Repository.

The **Interface Agent** has to represent the functional module to the mediator agent and sometimes to others Interface agents. It can be associated to one or more Wrapper agents encapsulating similar software. This Interface Agent transmits the data and service requests of a Wrapper agent to the Mediator agent which in turn can redirect these requests to the appropriate Interface agent. These requests and information exchange are formulated in the FIPA Agent Communication Language (ACL). The integration services provided by the Interface Agent are:

- Communication with the corresponding Wrapper Agent(s).
- Communication with the Mediator Agents
- Transfer (and possibly translation) of a demand from the Wrapper Agent to the Mediator Agent.
- Transfer of a demand from the Mediator Agent to the wrapper component.
- Calls to the Mediator Agent integration services.

The **Mediator agent** is the main agent of the proposed infrastructure. Indeed, it insures integration and interoperability between Interfaces agents and provides other services such as data and clock synchronisation. The integration services provided by the Mediator Agent are numerous, the mains services are:

- Communication with the Interfaces Agents
- Transfer a demand from an Interfaces Agent to an other one Interfaces Agent
- Acces to common/shared data
- Data flows control
- Control of the liability of the demands
- Synchronisation of the clocks for simulation
- …

To perform its numerous integration services, the Mediator Agent is composed of specialized agents. The main specialized agents composing the Mediator agents are:

- **Directory Facilitator (DF) Agent** - this is a specialized agent which provides a "yellow pages" directory service. Agents advertise their services to an agent domain by registering service- descriptions with the DF. The main actions supported by the DF Agent are: *deregister, modify, register et search.*
- **Agent Resource Broker (ARB) Agent** brokers a set of software descriptions to interested agents of the infrastructure. These agents may query on what software services are available An ARB advertises this service by registering with the DF. Software services are described by textual software descriptions, which list the properties of the software service. Part of the software description will describe where the software is located and how to interface with it (for example, networking protocols, encoding

types supported). An agent providing the ARB interface supports the FIPA-ARB ontology with commands and predicates for registering and searching for software services.

- **The Agent Management System (AMS) Agent**, controls the acces and the use of the Agent Communication Channel (ACC) of the agents. This agent creates and deletes agents, it decides if an agent can be registed to the agent community, and supervises the eventual plateform migrations of agents. The AMS agent maintains an index of all community agents with their universal address (GUID) thanks to the DF Agents. The main actions supported by the AMS Agent are: *authenticate, register-agent, deregister-agent, modify-agent.*
- **Data Integration (DI) Agent** is a specialized agent providing data integration services. This agent permits any agent part of the infrastructure to publish specific information, and insures that it can be shared among the other agents. This publishing permits to safely update information, and diffuse it, sometimes according to different formats, to the interested agents. This however requires such agents to subscribe to the DI Agent for theses informations (possibly organized in classes). Various subscriptions contract exist: “at each information updating”, “if information I exceeds a value V”, “at each time step”, etc. The (DI) Agents manages the set of subscription contracts for the agent community.
- **Synchoniser Agent** is a specialized agent providing services of time management, maily to synchronize the clocks of several simulation softwares that are integrated by the integration infrastructure. To manage time, the synchroniser Agents proposes similar services as the HLA’s RTI, in particular algorithms of synchronization (pessimistic and/or optimistic). This agent manages in particular different dates as the Lookahead date and the LBTS (Lower Bound on Time Stamp) date.

Note that the infrastructure is open to new services. For example, if the infrastructure needs to be replicated (for security or scability reason for example), an infrastructure-level DF agent could be added without modifying the previous agent organisation.

Integration principle

The integration principle can be illustrated by the treatment of a request of data or service from a functional software module A. This request, treated by the Wrappers, Interface and Mediator agents of the infrastructure, leads for instance to invoke the functional module B. The integration and interoperability process between these A and B software modules are performed by the infrastructure as follows:

- The wrapper Agent that encapsulates the functional module A receives or interpret the data or service request from this module, translates this request in ACL-FIPA message according a specific wrapping ontology and transmit it to its Interface agent of the infrastructure.
- The Interface agent receives this message specifying this request and transmits it to the Mediator agent.
- The mediator agent insures interoperability by coordinating the dialogues between agents of the infrastructure, with the help of the DF and ARB agents. The mediator provides also other services such as data and clock synchronisation. For instance, if the treatment of the request leads to the invocation of the functional software module B, the mediator will send a message to the Interface Agent associated to the module B through information provided by the ARB agent.
- This Interface Agent transmits the request to the Wrapper Agent relative to the software module B.
- The Wrapper Agent of the module B receives the message and translates it according to its ontology in specific function calls acceptable by the software module B.

A similar process will ensure the result of the request is returned to the wrapper agent A.

Thie figure 6 illustrates this integration principle.

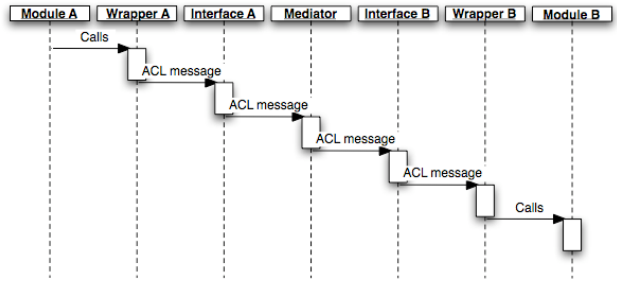

Figure 6. Integration principle

Infrastructure implementation details

The integration infrastructure for EDSS development proposed in this paper is currenty under development in the LSIS laboratory. More precisely we are specifying in details the models of each agents composing the infrastructure and the different domain ontology needed, according the FIPA specifications.

The agents implied in this infrastructure are deliberative agents and FIPA compliant. The FIPA compliant agents platform chosen to implement this infrastructure is JADE (Jade) JADE provides an Agent Communication Channel (ACC) permitting a message-routing function for inter-agent communication. These messages exchanged between agents are defined according to the ACL-FIPA standard. The ACC of JADE can be accessed by non-agent entities in order to route messages to agents but non-agent entities cannot be the recipients of messages routed via the ACC.

The deliberative agents are developed in JESS (Jess).

Consequently the agents communicate in ACL-FIPA language, with the clips contents language and use specific ontology adapted to the communication context.

The ACC of JADE provides a message-routing function for inter-agent communications. Messages are defined according to the ACL-FIPA standard. It can be accessed by non-agent entities in order to route messages to agents but non-agent entities cannot be the recipients of messages routed via the ACC.

5. CONCLUSIONS AND PERSPECTIVES

In order to facilitate the development of Environmental Decision Support Systems (EDSS) based on simulation, this paper has attempted to define a generic software architecture. In this aim, based on this study of some relevant EDSS project, we have defined the major functional requirements by such systems should meet. Then the functionalities of software modules supporting these functional requirement, and interactions between these modules have been defined.

Due to the various software solutions that can be adopted for each functional software module composing the EDSS, and the importance of interactions between these modules, integration and interoperability of such modules appear as the main problem to solve. Consequently, a generic agent based integration infrastructure has been proposed, mainly inspired on one hand of HLA (High Level Architecture), and on the other hand of the FIPA recommendations about agent software integration. Each of the agents composing this agent based integration infrastructure has been presented in details, the general integration principle illustrated, and some implementation details has been introduced

Currently we are finalizing the specifications of the integration infrastructure. The integration infrastructure is currenty in development under the JADE multiagents plateform. We are defining the behaviours of each agent composing the infrastructure and the different domain ontology needed, according the FIPA specifications. To develop and validate the infrastructure, a realistic EDSS is considerated (Serment and al 2006). This EDSS concerns the hydraulic management of the Camargue, a strongly human influences ecosystem on the Rhône river delta in the south of the France. This EDSS would integrate the various simulation softwares developed in the SimFonHyc project by the LSIS laboratory (Espinasse and Franchesquin 2005).

6. REFERENCES

Andriamasinoro F., 2003. Proposition d'un modèle d'agents eprese epr sur la motivation naturelle. Thèse de Doctorat, IREMIA, Université de La Réunion, France.

Anthony R.N. 1966. "Planning and Control Systems : a Framework for Analysis", Cambridge, Mass, Harvard University Press.

Becu N., Perez P., Walker A. Barreteau O., and Le Page C. 2003. "Agent based simulation of a small catchment water management in northern Thailand, Description of the CATCHSCAPE model", Ecological Modelling 170, Elsevier, pp.319-331.

Borrell F. Riaño D., Sànchez-Marrè M. and Rodriguez-Roda I. 2002. "Implementation of a Multiagent Prototype for WWTP Management", IEMSS, Integrated Assessment and Decision Support System, Lugano, Suisse.

Botequilha Leitão A., Grueau C. and al. 2001. "Decision Support System for Planning and Management of Biodivesity in Protected Areas, The research project PROBIO", Proceedings of the international Workshop on Geo-Spatial Knowledge Processing for Natural Ressource Management, Varese, Italy, 2001, pp. 145-151. http://alfa.ist.utl.pt/~cvrm/projects/probio

Bousquet F. and Le Page C., 2004. Multi-agent simulations and ecosystem management : a review. Ecological Modelling 176, p. 313-332,

Campos A.M.C., 2000. Une architecture logicielle pour le développement de simulations visuelles et interactives individus-centrées : application à la simulation d'écosystèmes et à la simulation sur le Web. Thèse de Doctorat, Université Blaise Pascal – Clermont II, France.

Cortès U., Sanchez-Marrè M. ad Ceccaroni L. 2000. "Artificial intelligence and environmental decision support systems", Applied Intelligence 13(1), pp.5-6.

Courdier R., Guerrin F., Andriamasinoro F.H., and Paillat J.M. 2002. "Agent-based simulation of complex systems : application to collective management of animal wastes", Journal of Artificial Societies and Social Simulation, vol. 5, no.3.

Denzer D., Swayne A. and Schimak G., Environmental Software Systems. Chapman & Hall, 1997.

IEEE-HLA. 2000. "High Level Architecture", IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)--- Federate Interface Specification --- 1516.1-2000

DMSO-HLA. 1996. HLA (High Level Architecture). https://www.dmso.mil/public/transition/hla/

Espinasse B. and Franchesquin N., 2005. Multiagent Modelling and Simulation of Hydraulic Management of the Camargue. Simulation, Vol. 81, Issue 3, p.201-221.

Ferber J., Gutknecht O. and Michel F., 2000. MadKit : une plate-forme multi-agent générique. Rapport de epresent n°R.R.LIRM 00061, LIRM.

FIPA, 2000. IEEE Foundation for Intelligent Physical Agents, http://www.fipa.org/

GeoTools. 2006. http://www.geotools.org/

Giampapa, J., K. Sycara, S. Owens, R. Glinton, Y. Seo, and B. Yu. 2004. "Extending the OneSAF testbed into a C4ISR testbed". SIMULATION 80 (12).

Jaber A. 1999. "Un système d'agents logiciels intelligents pour favoriser la epresentat entre des systèmes d'aide à la epresen dédiés à la epresenta des risques naturels". PhD thesis, Eco1e des Mines de Paris, France.

JADE, JADE tutorial http://jade.tilab.com

Janssen R. 1992. "Multiobjective Decision Support for Environmental Management", Kluwer Publishers.

JESS, http://hezberg.casandia.gov/jess.

Maillé E., B. Espinasse. 2006. "Decision Support for Forest Fire Risk Evaluation: Dynamic Modelling and Spatio-Temporal Integration", Proceedings of the IEEE 1st International Symposium on Environment, Identities and Mediterannean Area (ISEIM 2006), Corte-Ajaccio, France, July 2006.

Mowrer, H.T. 1997. "Decision support systems for ecosystem management: an evaluation of existing systems", General Technical Report RM-GTR-296.

Müller, J. P. 2004. "MIMOSA : epresentation des connaissances et simulation". http://lil.univ-littoral.fr/Mimosa

Nute D., Potter, W.D., Maier F., Wang J., Twery M., Rauscher H.M., Knopp P., Thomasma S., Dass M., Uchiyama H. and Glende A. 2003. "NED-2: an agent-based decision support system for forest ecosystem management" Environmental Modelling & Software, Elservier, 2003. http://www.fs.fed.us/ne/burlington/ned/

Prasithsangaree P., J. Manojlovich, S. Hughes and M. Lewis, 2004. "UTSAF: A Multi-Agent-Based Software Bridge for Interoperability between Distributed Military and Commercial Gaming Simulation", SIMULATION, Vol. 80, Issue 12, December 2004, pp. 647-657.

Reynolds, K., Bjork, J., Rienmann, H.R., Schmoldt, D., Payne, J., King, S., Moeur, M., DeCola, L. Twery, M. Cunningham, P., Lessard, G. 1999. "Decision Support For Ecosystem Management", Ecological Stewardship: A Common Reference for Ecosystem Management, Elsevier Science Ltd., pp. 687-722.

Serment J., B. Espinasse, E. Tranvouez. 2006. "Environmental Decision Support System for Hydraulic Management of the Camargue: Functionalities and Software Architecture», Proceedings of the IEEE 1st International Symposium on Environment, Identities and Mediterannean Area (ISEIM 2006), Corte-Ajaccio, France, juillet 2006.

Servat, D., Perrier E., Treuil J-P. and Drogoul A., 1998. When Agents Emerge from Agents: Introducing Multi-Scale Viewpoints in Multi-Agent Simulations. Proceedings of MABS'98, 183—198, LNAI n° 1534, Springer-Verlag, Berlin, Germany.

Tan G., Xu L., Moradi F., Zhang , 2000. "An Agent-Based DDM Filtering Mechanism", in proceedings of MASCOTS 2000, 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, San Francisco, USA, pp. 374-381.

Taylor S. J. E., R. Sudra, 2002. "Modular HLA RTI Services: The GRIDS Approach", Sixth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'02).

Wang F., S. J. Turner, L. Wang, 2005. "Agent Communication in Distributed Simulations", Lecture Notes in Computer Science, Volume 3415, Feb. 2005, pp. 11-24.

Wilson L. F., D. Burroughs, J. Sucharitaves, A. Kumar 2000. "An Agent-Based Framework for Linking Distributed Simulations". Proceedings of the 2000 Winter Simulation Conference J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.