

RagBus: A Universal Decision Support Platform Based on Intelligent Reflective Agents

Guangjun Jing

*State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology,
Nanjing University, Nanjing 210093, P. R. China*

Abstract: This paper describes an agent-based infrastructure named RagBus for open, integrated, and flexible system, i.e., adaptive intelligent decision support system. We may characterize the architecture of RagBus as the pattern of interrelationships (causal connections etc.) between entities representing specialist functions (e.g. they may be layers, individual components or sets of functionally similar components). Each entity may be regarded as a "slot" into which a specialist AI technique can be "plugged in". The RagBus is a reflective agent middleware (RAM) platform to construct flexible, dynamic, scalable, and robust distributed systems as multi-agent systems. We wish to discuss the approaches used and experiences gained in the development of a universal decision support platform which is being used to answer management questions. With its prototype implementation, it constitutes a useful framework for the research and testing of environmental decision problems. We only focus on the basic technologies which the RagBus needs, including agent description, notional architecture, reflective agent middleware, communication mechanism, and collaboration mechanism. According to the RagBus principles, some decision-making cases of waste tailing treatment plant have been tested, which can make good consultation and supervision for production process.

Keywords: Environmental decision support system; Intelligent agents; Reflective agent middleware; Waste tailing treatment plant.

1. INTRODUCTION

An Environmental Decision Support System (EDSS) is specific version of an environmental information system that is designed to help decision makers [Booty et al., 2001]. EDSSs can be divided into two clearly separate categories: problem specific EDSS and situation and problem specific EDSS. *Problem specific EDSS* are tailored to relatively narrow environmental problems (or domains), but they are applicable to a wide range of different locations (or situations). *Situation and problem specific EDSS* are tailored both to a specific environmental problem and to a specific location. These EDSS cannot easily be applied in a new location. Environmental systems have several distinctive features which make their formal representation different from that of other systems. These attributes, which involve dynamics, spatial coverage, complexity, randomness, periodicity, heterogeneity and scale, and paucity of information, are present simultaneously in environmental modelling [Rizzoli and Young, 1997]. So some

new and efficient techniques must be addressed to satisfy these issues.

An Artificial Intelligence (AI) assisted EDSS can be described as a multi-layered system connecting the user. The first layer is formed of the knowledge acquisition and learning module from spatial and temporal data base. Several AI, statistical and numerical models constitute the next layer. The next levels are the reasoning and integration modules that use several kind of models and knowledge to implement a predictive, planning or supervisory task over the environmental system. Finally, the upper level illustrates the interaction of the user with the EDSS [Cortés et al, 2000]. The Wastewater Treatment Plant (WWTP) control techniques applied, such as expert systems, neural networks, and hybrid AI networks (expert systems combined with neural networks), have been proved to be powerful tools, especially when applied to the control of treatment processes which are poorly understood or difficult to model with traditional control methods [Chien-Hsien and Vassiliadis, 1998].

The multi-agent architectures could be applied to develop systems able to find solutions to problems when more than one competence is needed [Balducelli and Gadomski, 1993]. In these systems, every agent is devoted to solve a certain sub-problem of the general problem; the different agents have different knowledge, problem-solving strategies, response times, and degrees of dependability and accuracy. They are autonomous, but they have a common way to communicate and co-operate.

DAI-DEPUR is a distributed and integrated supervisory multi-level agent-based architecture for a WWTP operation. It joins in a single framework several cognitive tasks and techniques such as learning, reasoning, knowledge acquisition and distributed problem solving. Also, different AI techniques are combined such as rule-based reasoning, case-based reasoning and model-based reasoning. Four levels are distinguished from the domain models: data, knowledge, situations and plans. On the other hand, taking into account the supervision tasks, seven levels are considered: evaluation, diagnosis, supervision, prediction, validation, actuation and learning [Sánchez-Marrè et al, 1996].

Because of complexity, large environmental decision problems must be distributed on a logical and physical network and interoperate with other systems. Such systems cannot be easily realized with traditional software technologies because of the limits of these technologies in coping with distribution and interoperability. The agent-based technologies seem to be a promising answer to facilitate the realization of such systems because they were invented to cope with distribution and interoperability. RagBus developed by the authors is a universal architecture to build decision support systems by use of agent based AI techniques.

2. THE TECHNOLOGIES BEHIND RAGBUS

Intelligent agents (autonomous components) that have their own goals and beliefs and can reason about their present and future behavior offer ample opportunity for rapid, incremental development of decision support systems.

2.1 The Notional Architecture

The RagBus is a universal decision support platform to make easier the development of agent application for interoperable intelligent multi-agent systems, as shown in Figure 1. The goal of RagBus is to simplify development through a

comprehensive set of system services and agents. To achieve such a goal, RagBus offers following list of feature to the agent programmer.

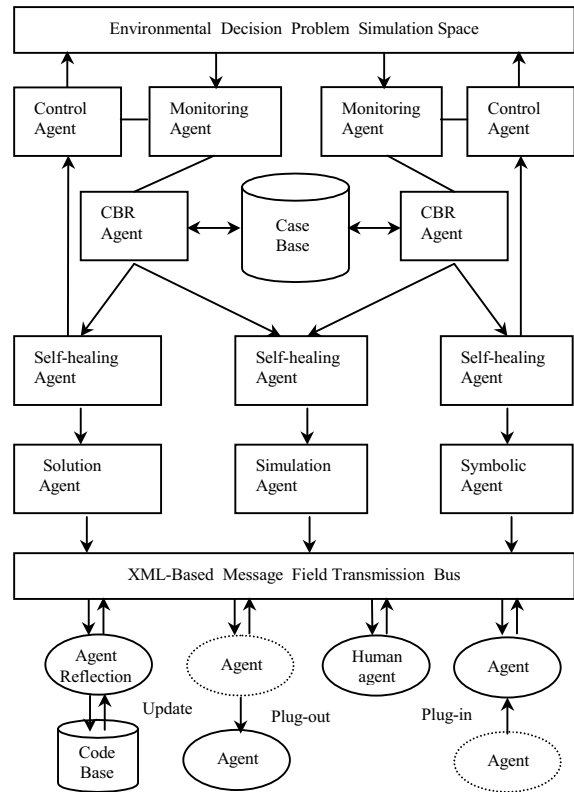


Figure 1. The Notional Architecture of RagBus Platform

The RagBus Intelligent Agents (RIA) implements a mechanism that contribute to give more flexibility to layered components: it will not be sufficient when a simple component malfunction produces degradation of the whole system. The agents must be able to substitute, repair or isolate a fault component during a certain time. RIA agents are implemented as a society of agents distributed inside four different levels of competences/roles:

Interface Level: The data-input-and-knowledge-output pattern is adopted. Monitoring agents are specialized to monitor (through the network) the behaviour of certain classes of components in any specific environmental decision problem space. Monitoring results (abnormal targets) are passed to other correlative agents in the next levels. Control agents get the components recovery methods and operation instruction from the decision-making levels, and bring them into effect.

Self-healing Level: Trying to actuate, at local level, some self-healing procedures when a fault is discovered in the condition. RIA agents (case-

based agents) specialised in the fault discovering. They confront the specific component behaviour with a list of behavioural cases: a case is a collection of indicators of a certain component working condition. They have a memory to store the normal functioning condition of the component as cases inside a case base and are able to retrieve a set of faults cases indicating characteristic statuses in presence of certain faults.

The case base is initially constructed off-line and periodically updated on-line by some learning activity of the agent itself. Initially the cases will be learned through the simulation of the possible faults. During normal operations the agents could learn from experience other characteristic fault behaviours and implements them as additional cases.

Hybrid AI Level: Establish a set of recovery actions at a more strategic level, and suggest them to the operators working, when the self-healing mechanism fails or can not be actuated. The hybrid AI system uses knowledge management mechanism, combined with rule-based reasoning, model-based reasoning, genetic neural network based learning and mining, and other AI techniques. Simulation agents select and construct a series of models based on production data. Solution agents divide the actual problem into small parts until each model can satisfy every sub-problem. Symbolic agents transform and integrate all the results from every sub-problem solution, and form unified symbol-based instruction plans.

At different reasoning levels, Self-healing agents and symbolic agents can actuate the recovery or reconstruction actions inside the operation layer. They have the capacity to move some new software and procedures inside the damaged layer and substitute/repair the function performed by the fault component. They actuate their actions with the support of the control agents belonging to the interface level.

BusCore Level: Providing an eXtensible Markup Language (XML) based message field transmission bus to support agent plug-in and plug-out, and support agent reflection [Soltysiak et al, 2000].

This message field transmission medium should be flexible, dynamic, and scalable enough to enable open communication for collaboration. The field enables such communication among the organization of agents. As a communication medium, agents communicate with others in the peer-to-peer manner or in the multicast manner on the field. As an information-sharing medium, agents can share information among themselves on the field as a message blackboard. Thus the field

works as a logical network and at the same time as a shared memory.

Communication via the field is an event-driven, multicast-based communication. All agents on the field listen to all messages in the field, and each of them reacts to messages according to its own criteria named patterns. An agent becomes aware of an event anyway and reaction to the event is up to the agent. If a set of agents always reacts to certain types of messages, these agents would constitute a weak bounded group within the organization. This kind of pre-communication and pre-grouping is required for collaboration over open environment to be flexible and dynamic.

Moreover, agents on the field can be added to and deleted from the field anytime independently of other agents. The patterns of agents can also be dynamically changed. Thus, collaboration of the field is flexible and dynamic.

2.2 Reflective Agent Middleware

The Reflective Agent Middleware (RAM) is a middle layer platform to construct flexible, dynamic, scalable, and robust distributed multi-agent systems. RAM supports collaborations of agents in a seamless and transparent way, and it enables agents to be connected in a plug-and-play manner. When a new agent is plugged into the network, RAM arranges the organization of agents, adapts the agents of the organization to each other, and manages their collaboration [Soltysiak et al, 2000].

Each agent has its own data format and protocol. Therefore, an organization of agents must agree on data formats and protocols before starting to collaborate. Within the RAM, agents can update interfaces and protocols for new interactions and collaborations in runtime.

The reflective adaptation is based on the agent/action programming model. Agents are realized as router agent components. In a Router agent component, interfaces and protocols are realized as actions that are replaceable and extensible modules of the router components. Once interfaces and protocols are agreed among the organization of agents, actions of each agent are replaced or added dynamically. These actions are supplied from repositories with or without collaboration with mediators. The module can be obtained also from the agent who talks with a new protocol.

Reflective adaptation involves changing the code dynamically. This can be used to add new features to the interface, or even update existing code

(adding an extra parameter to a method call, for example).

2.3 Communication Mechanism

All agent communication is performed through message passing and blackboard mechanism. Message passing includes synchronous communication method and asynchronous communication method.

Synchronous communication method is adopted between agents, and each agent's name and address table are sent to other agents. When the address table changes because of insertion or cancellation of agents, one agent will send new address table to the other agents. Asynchronous communication method is adopted when agents send the results to the scheduled place beforehand. Blackboard mechanism is adopted between agents to cooperate with each other. Agents can write their names and addresses on blackboard to publicize and capture information, if needed, synchronous communication and asynchronous communication methods can be built to improve communication efficiency.

The communication is based on the coexistence of several Java Virtual Machines (VM) and Java RMI (Remote Method Invocation) between different VMs and event signaling within a single VM. Each VM is a basic container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. In principle, the mechanism allows also several VMs to be executed on the same host; however, this is discouraged because of the increase in overhead and the lack of whatever benefit. Each agent container is a multithreaded execution environment composed of one thread for every agent plus system threads spawned by RMI runtime system for message dispatching.

2.4 Collaboration Mechanism

Based on RagBus, Agents work concurrently to solve a special environmental domain task as a distributed intelligent system. Cooperative solution includes two negotiatory mechanism and dividual mechanism. Negotiatory cooperative solution means that according to their own abilities and working experiences, Agents solve a domain problem by negotiation and argumentation, at last, Cooperating agent summarizes negotiatory results and confirm the resolvent. Dividual cooperative solution means that after dividing a whole task into some small granularity of sub-tasks, according to their own abilities, sub-tasks are scheduled to

agents, after sub-tasks being completed, Cooperating agent synthesize the sub-tasks to get the final result. The system consists of some agents ($A_1, A_2, A_3, \dots, A_n$), A_i figures i th agent, $G_i[A_i, T]$ shows solution result of A_i to task T . The general procedure of Dividual cooperative solution is shown as follow:

- (1) Task decomposition: cooperating agent divides a large task T into some small granularity of sub-tasks space $\{T_i | i=1 \sim r\}$, at the same time, gives the corresponding resource requiremental table $T: R_i = \{r_j | j=1 \sim k\}$ to solve sub-task;
- (2) Sub-task scheduling: According to agent's ability $A: CAP_i$, resource table $A: R_i$, and resource requiremental table to solve sub-task, Cooperative agent assigns sub-tasks to every intelligent agent.
- (3) Sub-task solution: Each solves the accepted sub-task problem, and submits the results to Cooperative agent.
- (4) Sub-task composition: After all agents have completed their sub-tasks, cooperative agent synthesizes results of the solution task.

On the assumption that task T 's condition restriction domain, $domain(X): F(x) = PBX, P = (P_{ij})_{m \times n}$ shows restriction modulus, $B = (b_1, b_2, \dots, b_m)^T$ figures restriction limit, $X = (x_1, x_2, \dots, x_n)^T$ restriction parameter. Solution procedure of sub-task is actually to find a ideal point X in $domain(X)$, in order to get the optimized result of sub-task. i.e., making $T_i(X) = c_{i1} * x_1 + c_{i2} * x_2 + \dots + c_{in} * x_n$ reach the optimization. Then the solution space of sub-task can be got:

$$Best \begin{cases} T_1(X) = C_{11}X_1 + C_{12}X_2 + \dots + C_{1n}X_n \\ T_2(X) = C_{21}X_1 + C_{22}X_2 + \dots + C_{2n}X_n \\ \dots \\ T_r(X) = C_{r1}X_1 + C_{r2}X_2 + \dots + C_{rn}X_n \end{cases}$$

Obviously, it is impossible to make all the sub-tasks reach optimization at the same point. To solve this conflict, the sub-tasks to be solved can be fuzzilized by using fuzzy mathematical method, the procedure is shown as follow:

Firstly each agent get the optimized result $T_i = Best(T_i)$ under the restriction condition, i.e., local optimized solution result. Generally these satisfactory restriction points distribute in $domain(X)$, the requirement of sub-task can be relaxed in the interest of making these restriction points consistent or contiguous as possible. Cooperative agent endue every sub-task a flexible gene $d_i (\geq 0)$,

the value of d_i increases when the importance of sub-task decreases. The optimized solution of sub-tasks have been fuzzilized by this way. Supposing fuzzilized solution of T_i as $G_i(x)$, then its subjection degree function can be shown as:

$$G_i(X) = \begin{cases} 0 & T_i(X) < T_i - d_i \\ 1 - [T_i - T_i(X)]/d_i & T_i - d_i \leq T_i(X) < T_i \\ 1 & T_i(X) \geq T_i \end{cases}$$

After that, using fuzzy mathematical methods to synthesize fuzzilized solution set $\{G_i(X)|i=1\sim r\}$ in condition domain (X), some very closer points set can be found to meet the demand of each sub-task.

3. APPLICATION CASES

To solve more complex problems, intelligent agents are activated: they include more sophisticated reasoning mechanisms based on various artificial intelligence methods. Several intelligent methods will be used by the different agents. With this perspective the global multi-agent system is a hybrid system because different reasoning methods are adopted to reach a common objective. Integration of different methods inside the same architecture, aimed to solve a co-operative problem, is a feature of RagBus system.

According to those working principles, some decision-making cases of waste tailing processing unit of Huang Shaping lead-zinc sulphide plant have been tested, which can make a precise prediction of some parameters and make good consultation and supervision for production process. Some key prediction results are shown in Table 1. This process is very complicated, which needs multi-agent and hybrid AI based cooperation between prediction agents (solutions generators), case-based agents (solutions recorders), symbolic agents (solutions inspectors), and the human agent that is the final examiner of the generated solutions.

The training data needed for the prediction agents was created from the simulation system, which was controlled by the symbolic agents. Therefore, the prediction agent was learning the control pattern from the symbolic agent. The symbolic agent generates some mineral characteristics and separate status values in the froth flotation tank, and then sends those values to the prediction agent to generate the concentrate grade and recovery (CGR). If the CGR cannot release the critical condition such as when the CGR is low in the froth tank, then the case-based agent will make surface level reasoning to confirm if the design parameters are

appropriate, including the grinding flowsheet, separation method, and separation flowsheet. If no faults, the symbolic agent will start deep level reasoning to modify the operational parameters, including flotation reagents (medicaments), water-feeding, and air-feeding. and will perform the control again, until the critical condition in the treatment process is released. In symbolic agents, the object-oriented knowledge representation method is utilized. In symbolic agents, the object-oriented knowledge representation method is utilized. The following is an example:

Object identifier	Types
Objectname: flotation	Character
Superclass name: separation method	Character
Variables: stage number of grinding-flotation	Number
Cycle	Number
Cleaner and scavenger	Number
Sequence of flotation	Character
Knowledge	
Plant name:	
Plant address:	
Ore deposit type:	
Plant scale:	
Mineral type:	
Knowledge processing methods	
Prediction rule 1: IF [Cu, Pb, Zn waste tailing] THEN [flotation]	
Prediction rule 2: IF [single metal, coarse disseminated waste tailing] THEN [one stage, one circuit flotation]	
Prediction rule 3: IF [single metal, non-uniform disseminated waste tailing] THEN [tail regrinding, two or three stage flotation]	
Prediction rule 4: IF [high content of valuable component, low requirement of concentrate quality] THEN [rougher concentrate]	
Knowledge acquisition 1: asking about substance composition, and grades of valuable components in run-of-mine waste tailing and oxidation rate	
Knowledge acquisition 2: asking about the dissemination size, dissemination character	
Knowledge acquisition 3: asking about waste tailing texture, product kings, etc.	

Table 1. Comparison of some parameters between mill design and system prediction

Parameters	Mill Design	System Prediction
Grinding flowsheet	1 stage, closed circuit	1 stage, closed circuit
Separation method	Lead equal flotation	Lead equal flotation
Separation flowsheet	Pb (I): 1 rougher, 2 scavengers	Pb (I): 1 rougher, 2 scavengers
	Pb (II): 1 rougher, 4 cleaners, 2 scavengers	Pb (II): 1 rougher, 4 cleaners, 2 scavengers
	Zn • S: 2 roughers, 2 scavengers	Zn • S: 2 roughers, 2 scavengers
Flotation reagents (g/T)	Zn / S: 1 rougher, 2 cleaners, 2 scavengers	Zn / S: 1 rougher, 2 cleaners, 2 scavengers
	Xanthates: 448; CuSO ₄ : 537; ZnSO ₄ : 181; SN-9: 18.3; 2 [#] Oleic: 180; CaO: 15550	Xanthates: 460; CuSO ₄ : 551; ZnSO ₄ : 175; SN-9: 20; 2 [#] Oleic: 182; CaO: 16000
Grade recovery (%)	$\beta_{Pb} = 71.36$; $\epsilon_{Pb} = 90.23$; $\beta_{Zn} = 44.93$;	$\beta_{Pb} = 72.21$; $\epsilon_{Pb} = 89.55$; $\beta_{Zn} = 46.74$;
	$\epsilon_{Zn} = 92.75$; $\beta_S = 35.78$; $\epsilon_S = 56.67$	$\epsilon_{Zn} = 90.01$; $\beta_S = 34.52$; $\epsilon_S = 57.89$

Notes: Zn • S --- Lead and zinc flotation; Zn / S --- Lead and zinc separation; β --- Grade; ϵ --- Recovery

4. CONCLUSIONS

People from different backgrounds and fields of interest view quite different what environmental information systems and environmental decision support systems are and what their frameworks and components should be. This paper presents a notional prototype agent-based decision support platform, RagBus, to give some key technologies, not including some specific environmental domain issues. The framework is designed to make use of the capabilities of software agents for a comprehensive range of analytical functions, and to resolve the productivity issue by providing a universal infrastructure for environmental decision problems.

In the future, we will make RagBus support collaborations of agents in a seamless and transparent way, and it enables agents to be connected in a real plug-and-play manner. Each agent has its own data format and protocol. Within the RagBus, agents can update interfaces and protocols for new interactions and collaborations in runtime.

5. ACKNOWLEDGEMENTS

This research has been supported by State Natural Science Fund of China under Grants No. 59574034, State High Technology Propulsive Project of China (2001-JS008).

6. REFERENCES

Balducelli C., Gadomski A. M., Intelligent agents in computer supported training for emergency

management. In: Proceeding of the First International Round-Table on Abstract Intelligent Agent, Rome, 23-25, 1993.

Booty W. G., Lam D. C., Wong I. W., Siconolfi P., Design and implementation of an environmental decision support system. *Journal of Environmental Modelling & Software*, 16, 453-458, 2001.

Chien-Hsien W., Vassiliadis C. A., Applying hybrid artificial intelligence techniques in wastewater treatment. *Engineering Applications of Artificial Intelligence*, 11, 685-705, 1998.

Cortés U., Sánchez-Marrè M., and Ceccaroni L., Artificial intelligence and environmental decision support systems. *Applied Intelligence*, 13, 77-91, 2000

Guariso G., Werthner (Eds.) H., Environmental decision support systems, Ellis Horwood-Wiley, 1994, ISBN 0-77458-0255-9.

Rizzoli, A. E., Young W. J., Delivering environmental decision support systems: software tools and techniques. *Journal of Environmental Modelling & Software*, 12(2/3), 237-249, 1997.

Sánchez-Marrè M., Cortés U., Lafuente J., R.-Roda J., and Poch M., "DAI-DEPUR: An integrated and distributed architecture for wastewater treatment plants supervision." *Artificial Intelligence in Engineering*, 10 (3): 275-285, 1996.

Soltysiak S., Ohtani T., Thint M., and Takada Y., An agent-based intelligent distributed information management system for internet resources. In: proceeding of INET 2000, Japan, 2000.