A CIM Application of a Multi-Agent System

Z. Kouba, L. Lhotská, P. Mikšovský
Czech Technical University in Prague, The Gerstner Lab
Technicka 2, 166 27 Praha 6, Czech Republic,
tel: ++(420)(2)24357379, fax: ++(420)(2)24357224,
e-mail: {kouba,lhotska,miksovsp}@labe.felk.cvut.cz

Abstract

The paper consists of three parts. In the first one it introduces some general remarks on multi-agent technology, in the second one the modular kit of multi-agent systems developed in the Gerstner Lab is introduced.
Finally, the third part describes a case study of a capacity planning task and its particular solution. The application consists of four basic agents developed at the Gerstner Lab of the Czech Technical University in Prague. They are playing four different roles as transaction management, on-line planning, off-line planning and off-line messaging. The development has been supported by the EUROSAT project No. 9645 of the European Comission's PECO programme.
The designed solution was tested during the 2 days long simulation game which took place in October 1996. During the game the system was running at the Technical University of Madrid and the partners from other five universities from five European countries entered their orders into the system through the Internet. The correctness, robustness of the chosen solution and the efficiency comparison of the Internet versus private VSAT interconnection was evaluated within this experiment.

Keywords
CIM, multi-agent systems, genetic algorithms

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35390-6_58

L. M. Camarinha-Matos et al. (eds.), *Intelligent Systems for Manufacturing*

1 INTRODUCTION

The idea of CIM stresses its global strategic goal: to integrate all company activities into a unified management structure exploring a large scale hierarchy of computers. It is expected to embrace corporate product design, manufacturing, marketing, sales management, planning, scheduling, real-time machine control, material handling, assembling, quality control and product dispatching (Marik, 1993).

Different software tools for CAD, CAP, CAM, CAQ etc. have been developed and are successfully applied in solving partial tasks of the entire goal. The ambitious target of CIM emphasises the complete integration of computer-aided activities in the factory as well as the use of the knowledge-based technology within the entire information processing.

The problems which are expected to be solved by computers are growing rapidly in their extend, complexity as well as in diversity of the subtasks. The growing complexity of problems to be solved requires new overall system architectures integrating the already existing "islands" of well-working algorithms into more robust global systems.

In the area of decision making systems, this integration trend has resulted in various blackboard architectures designed for distributed knowledge processing or for integration of distributed knowledge bases. Such approach is based on the ideas of Distributed Artificial Intelligence. The multi-agent technology (Wooldridge, 1995) connected with the OO-style of programming is currently considered to be much more suitable for creating open, flexible environment able to integrate software pieces of diverse nature written in different languages and running on different types of computers.

Design and implementation of multi-agent system require to solve the questions of function and task allocation to individual agents, global strategy of behaviour of the society of agents and a very important problem of inter-agent communication.

Multi-agent approach can be used advantageously where

a) present solution to a given problem based on use of a single "monolithic" program does not satisfy requirements of a customer any more, e.g. regarding possibilities of propagation, modification, documentation and software maintenance. In such a case we speak about the problem of decomposition.
b) it is necessary to integrate already tested and very reliably functioning partial software or hardware solutions into larger, global systems. That is the problem of so-called system integration. The problems of system integration are closely connected with the topic of openness of global systems.
c) complex systems are developed from modular "bricks", i.e. where there are strong natural demands for repeated use of partial blocks at composing the global system. The required result is the design of architecture or topology of a system composed of many elements, then we speak about construction or compositional problems. The dominant idea of solution of such problems is a high degree of reusability of software elements.

2 DISCIM ENVIRONMENT

Our goal was to design, develop and implement a comparatively open multi-agent environment suitable for efficient creating of complex knowledge-based or decision support systems. This environment was expected to be able to integrate geographically distributed knowledge sources or problem solving units. The task under consideration is located just on the borderline between Software Engineering and Artificial Intelligence.

The name of the developed environment DISCIM reflects its first intended application - building a geographically DIStributed decision making system for CIM purposes. The following principles have been used to design the DISCIM system:

- The agents are independent, autonomous entities communicating in a peer-to-peer way among themselves. The asynchronous message passing/broadcasting in UNIX/INTERNET environment is used to perform this communication.
- Each agent consists of a functional body (usually a stand-alone program with a well-defined functionality) and a wrapper (which is responsible for involvement of the agent into the community of agents). The wrapper contains a model of the agent's behaviour in the form of the list of classes in the Eiffel notation. Some classes represent messages which can be received and sent out by the agent. As a matter of fact, the wrapper translates the inter-agent communication into the instructions for the activity of the functional body and mediates the results of the body activity back to the agents' community.
- The simplest typical model of an agent's behaviour can be described in the following way: At the very beginning, the agent is initialised (included into the agents' community). Then it observes the messages broadcasted within the community. In the case, the knowledge contained in the agent is required by some member of the agents' community, the agent sends out a confirmation message (informs the requesting agent that it is ready to solve the given subproblem) and initialises its own body. If the body requires some piece of information from the others in the community, the wrapper broadcasts this bid. The agent then waits (for a pre-set time period) for any confirmation message from some of the agents; the agent which confirms the ability to provide the required piece of information as the first one is considered (the required information may be possessed by more than one agent). If no confirmation message has occurred within the pre-set time interval, the original agent replaces the required piece of data by a default value. The model of behaviour contained in the wrapper describes only the re-active part of behaviour (in the sense of Woold's and Jenning's classification (Wooldridge, 1995). The deliberative behaviour which relies on explicit, internally held symbolic models (if such behaviour is required) is expected to be implemented as a natural part of the agent's body. Of course, the re-active behavioural model can be enriched by extending the list of classes or messages to be received and sent out.

- There is no central memory or control in the agents' community. Similarly to ARCHON (O'Hare, 1993), the corresponding pieces of the control strategy are "owned" by the individual agents or meta-agents, respectively, they are embodied into the individual behavioural models as rules contained in the classes (input/output messages description).
- A library of standard agents contains some complete agents (body and wrapper), prepared for immediate incorporation into a system, as for example
  a) a rule-based diagnostic expert system of the FEL-EXPERT type, exploring the pseudo-Bayesian model of uncertainty;
  b) a qualitative simulation system QUASIMODO (Stepankova, 1994);
  c) a neural network emulator (the body programmed in C++).
- A meta-agent is aimed at permanent observing of the multi-agent community structure and its reconfiguration if necessary. The main goal of the meta-agent is to discover a failure of a particular agent (caused e.g. by loosing the Internet connection or by fatal failure of the corresponding computer; to re-distribute the load. The activity of this agent is based on the "cluster and clone" technique (Maturana, 1996). The reconfiguration or load relocation problem should be solved within the given cluster. This problem is usually easily solved by creating a new process (agent) which is a copy ("clone") of a cluster member. Typically, to one cluster of agents one corresponding meta-agent is assigned to manage the reconfiguration/relocation problem at hand.

3. THE CASE STUDY

The background story of the presented application is the following. Let us imagine a company producing some general products made to order rather then to stock. The orders are held in a queue and the realisation of particular order is scheduled in such a manner that it starts some time before the required due-date to be finished in time. It is not desirable to start it too early because of the capital blocking, stocking costs, etc. The planning agents are to maintain the production plans telling when to start realisation of particular orders.
The problem belongs to the capacity planning task class. It means the production capacity of the plant per a time unite (let us say one hour) is known as well as the time necessary for production of an unit (let us say unit of volume) of each particular product.

The overall architecture

The client making the orders is using a regular HTML browser to communicate with the Remote transaction agent of the server side.
After storing the order in the queue held in the database, the On-line planning agent tries to introduce the new order into the schedule. If it succeeds to find free production capacity to realise the order satisfying the delivery date required by the user, it informs the user on accepting his order. If not, the user is informed, that his order has not been accepted in this first lap. Later on, he will be informed by e-mail

on the definitive acceptance of his order in dependence on the result of the off-line planning stage.
The operator's console makes possible to check various aspects of the production planning, mainly the queue of orders. It makes possible to invoke the Off-line planning agent to carry out re-scheduling of the orders as well.
During the re-scheduling process the production capacity - orders assignment may be refined. The necessary production capacity may be found to realise some of those orders having not been accepted in the on-line planning stage.
The clients who will be affected by the result of the off-line planning stage will be informed by the Remote messaging agent via e-mail.

On-line Planning Agent

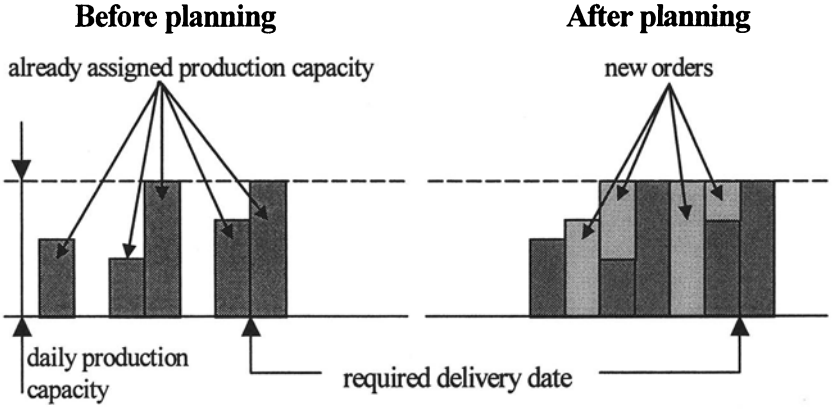

Figure 1

After a new order has been received, the on-line planning agent is triggered to check if the delivery date required by the user can be satisfied.
The planning system tries to find free position in the plant's time manager which will maximise the profit. It means that if there is enough free production capacity, it takes only working days into account as the production in the working days is cheaper than that one in holidays.
There are two basic planning strategies at disposal:

- *Looking for the latest starting date.* In this case the system tries to find the latest possible date when the realisation of the given order should start to be finished before the delivery date required by the user (see Fig. 1).
- *Looking for the closest delivery date.* If the above method fails (there is not enough production capacity to satisfy the delivery date required by the user), the closest delivery date will be found and suggested to the client. If he agrees, the order is accepted with the new due-date.

Off-Line Planning Agent

The planning methods introduced above do not find the optimal solution, of course. The aim of the planning algorithm is to maximise the profit. Usage of the on-line planning methods may cause blocking the production means by less profitable order. Therefore "shaking" the schedule time to time may improve the production efficiency. The user entering the order may assign some priority to his order. The price increases with the priority. At the other side when the producer does not catch to deliver the products in time, he will pay penalty for each day of delay. Again the penalty increases with the priority of the order.

Such a business policy makes possible to use the financial profit as the formal optimisation criterion (objective function). The off-line planning agent may find free production capacity for realisation of some profitable orders which were rejected by the on-line planning agent. As the result some of the earlier confirmed orders may be shifted in the schedule. The Off-line messaging agent informs the affected customers about the change of the delivery date. The penalty may be assigned dynamically to the order. It prevents permanent postponing of the same order (perhaps not important from the producer's profit point of view).

The core of the off-line planning agent is the modification of Simple Genetic Algorithm (SGA). First genetic algorithms were considered as computer programs imitating evolutionary processes in nature (Goldberg, 1989). Genetic algorithms manipulate a population (a set) of potential solutions to an optimisation (or search) problem. More precisely, they operate on strings (the solutions encoded as strings of bits from a binary alphabet) equivalent to the genetic material of individuals in nature, and not directly on solutions themselves. This approach is useful for using such an algorithm for many different areas. As in the nature, selections are realised with random functions. They provide the necessary driving mechanism for better solutions to survive. For comparing with other solutions in the population each solution is associated with a fitness value that reflects how good the solution is. The higher value of the fitness function means that there are the higher chances of survival and reproduction and larger representation in the next generation. Recombination of genetic material in genetic algorithms is simulated through a crossover mechanism that exchanges bits between strings. Another basic operation called mutation ensures random changes of bits of strings. Mutation has also a direct analogy in the nature and plays the role in regenerating lost genetic material.

Simple Genetic Algorithm (SGA)

Holland's genetic algorithm (Holand, 1975) is commonly called the Simple Genetic Algorithm (SGA). For SGA is typical to work with the problem encoded to binary strings. The next generation is created from strings of current population using genetic operators crossover and mutation. This generation cycle is repeated until a predefined number of generations are processed. For the SGA structure see:

```
procedure Simple Genetic Algorithm
begin
        generate first population
        evaluate population
        while (number of generations not processed) do
        begin
                select individuals for evolution
                perform crossover and mutation
                evaluate population
        end;
end;
```

Encoding the problem

First of all it is necessary to find the encoding mechanism for representing the optimisation problem's variables. The encoding problem is depended on the type of problem. In case of optimising flows in a transportation problem, variables (flows in channels) represent continuous values, whereas the variables in a travelling salesman problem are binary quantities, representing the inclusion or exclusion of an edge in the Hamiltonian circuit. Each problem's solution is encoded to a unique binary string.

Many optimisation problems have float-valued continuous variables. The simplest encoding method uses their integer representation. Each variable is linearly mapped into integer in a specified range and integer is encoded using a fixed number of binary bits. For example, consider a continuous variable defined in the range from -32.767 to 32.767. We can simply encode such a variable with an accuracy of three decimal places by multiplying its real value by 1000 and cutting decimal places. Therefore the value is linearly mapped to integers in the range from -32767 to 32767.

Fitness function

The most important function in SGA is the fitness function. This function provides the mechanism for evaluating potential solutions and through this mechanism applies our optimising strategy to the population. Although the principle of evaluation and range of values varies from one problem to another, in order to maintain uniformity over various problems we use normalised range of values from 0 to 1.

In the beginning we said that genetic algorithms work only with encoded strings without real data, but this function is an exception. It is the only connection between encoded strings and the real problem.

Selection

Selection enforces the nature's law survival of the fittest. It filters better solutions to survive while weaker ones will die. In SGA, the higher value of the fitness function means the higher chance of surviving in the next generation. In the

proportionate selection scheme, a string with fitness value f is copied into f_i / f_{avg} offspring, where f_{avg} is the average fitness value of the population. It means that an individual with a fitness value better than the average is copied into one or more offspring.
For implementation of the proportionate selection, the SGA uses the *roulette wheel selection scheme* (Michalewicz, 1994). Each string has allocated a sector of roulette wheel with the centre angle of $2\pi f_i / f_{avg}$. An individual is selected when a randomly generated number in the range 0 to 2π corresponds the sector of the individual. Roulette wheel selection can make large sampling errors in the sense that the final number of offspring allocated to an individual might vary significantly from the expected number. The allocated number of offspring is near to the expected number only for very large population sizes.

Crossover

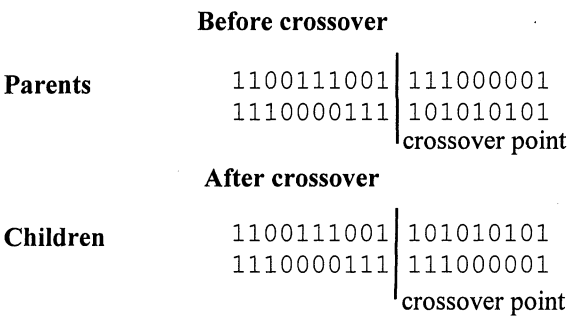

Figure 2

Crossover can be considered as the SGA's crucial operation. Parents of the next generation are randomly selected from the population. The SGA uses the simplest approach - single point crossover. Let two chromosomes p_1 and p_2 selected in the parent generation. Let i is a randomly selected number (crossover point), where $1 \leq i < length(p_1)$. Let $head(p_1)$ is the string of first i bits of p_1 and $tail(p_1)$ is the string of remaining bits of p_1. The crossover creates two children $head(p_1) \mid tail(p_2)$ and $head(p_2) \mid tail(p_1)$ where | means concatenation. The example of crossover is given in the Fig.2.

The crossover is not always effected. After choosing a pair of strings the algorithm executes crossover with the predefined probability. This probability is called *probability of crossover* and crossover is fired only if a randomly generated number in the range from 0 to 1 is lower than probability. Otherwise the strings remain without change.

Mutation

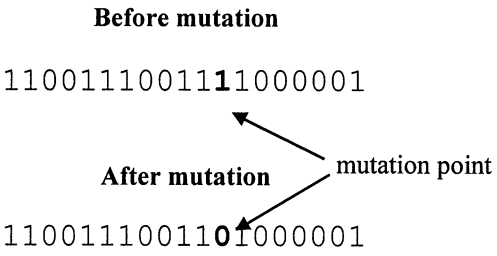

Figure 3

After crossover the strings are ready to mutate. Mutation (see Fig. 3) makes the random flipping of bits of the string. Just as probability of crossover controls crossover, another parameter called *probability of mutation* controls mutation. The bits of string are flipped independently - that means, the mutation of a bit does not affect the value of another ones.
Mutation is the means how to overcome blocking in the local extreme in optimisation tasks.

4 CONCLUSION

The aim of the prototype implementation is to demonstrate a possible way of using world-wide network as Internet for CIM purposes. The prototype made possible to analyse possible problems caused by the nature of http protocol and to show their elementary solution. At the moment when the implementation of the prototype started there was no comparable product on the market. Nowadays, there are such products like WebSpeed (product of Progress Software Inc.) solving the task of database transactions via Internet.
The core of the prototype are the planning agents. The most interesting one is the off-line planning agent based on the simple genetic algorithm.
The functionality of the presented system has been verified during the experiment on October 17-18 1996. During the experiment the server described above was running at the Technical University of Madrid, Spain. The aim of the experiment was to verify the design and implementation of the system and to compare the communication efficiency of the Internet and satellite under intensive communication traffic.

6 REFERENCES

Holand J.H. (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor. Mich.

Goldberg D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Reading, Mass.

Marik V., Lazansky J. (1993) The AI Impacts on CIM-Concepts. Research Report TR-PRG-CIM-8/93, FAW Linz and CTU Prague.

Maturana, F.P., Norrie, D.H. (1996) Multi-Agent Coordination Using Virtual Clustering in a Distributed Manufacturing System. In: Proceedings of 5th Industrial Engineering Research Conference, Minneapolis, MN, U.S.A.

Michalewicz Z. (1994) Genetic Algorithms + Data Structures = Evolution Programs, second edition, Springer-Verlag Berlin.

O'Hare, G.M.P., Woold, M.J. (1993) A Software Engineering Perspective on Multi-Agent System Design. In: Distributed AI: Theory and Praxis (Avouris, M.N., Gasser, L., eds.), Kluver Academic Publishers, Dordrecht, pp. 109-128.

Stepankova, O., Marvan, I.(1994) Time Independent Global Constraints and QSIM. Proceedings of ESM Conference, Barcelona, pp. 470-474.

Wooldridge, M.J., Jennings, N.R. (1995) Agent Theories, Architectures, and Languages: A Survey. In: Intelligent Agents (Woold, M.J., Jennings, N.R., eds.), LNAI No. 890, Springer Verlag, Heidelberg, pp. 1-39.

7 BIOGRAPHY

Zdenek Kouba received his PhD. degree in 1991. He is currently an assistant professor at the Czech Technical University in Prague.

Lenka Lhotská received her PhD. degree in 1990. She is currently an associated professor at the Czech Technical University in Prague.

Petr Miksovsky received his MSc. degree in 1996. He is currently a PhD. student at the Czech Technical University in Prague.