A Hotel Information System implementation using MAS technology

Armando Robles P.
Pablo Noriega B.V.
Institut d'Investigació en Intel·ligència Artificial
IIIA-CSIC.
Bellaterra, Barcelona, Spain
{arobles,pablo}@iiia.csic.es

Marco Julio Robles P.
Héctor Hernández T.
Victor Soto Ramírez.
Edgar Gutiérrez S.
TCA Research Group
Monterrey, N.L. México
{mjrobles,hhernandez,vsoto,egutierrez}
@grupotca.com

ABSTRACT

This paper reports on our progress towards a framework for enabling Intelligent Organizations with agent technologies. We have built and deployed the part of the framework consisting of organizational middleware and domain agents. The organizational middleware reads workflow scripts at run time and interprets them delegating to specialized server agents access to business rules and data bases. Those server agents, in turn, communicate with specialized user agents that facilitate human interactions through traditional plain and grid forms. We have used these ideas to transform a conventional Hotel Information System into a multi layered, agent supported information system. Our MAS-ified hotel information system is already in operation in 10 hotels with a total of 2040 rooms.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence—*Programming languages and software*; H.1 [**Information Systems Applications**]: Models and principles—*Systems and information theory, User/machine systems*; D.1 [**Software**]: Programming Techniques—*Miscellaneous*

General Terms

Applications of multi agent systems, Electronic Institutions

Keywords

Agent-oriented software engineering and agent-oriented methodologies, applications of autonomous agents and multi agent systems, Electronic Institutions

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

1. MOTIVATION

Grupo TCA is a medium-size privately owned information systems company. It has been active in the design and development of integral information systems for the Latin American market, since 1982. Its main business comes from integral systems for vertical industries such as hotels, hospitals and retailers.

Ten years ago, TCA became aware that the vertical industry market was being sensitive to the *Business Intelligence* buzzword and TCA realized that, in the years to come, its use would provide competitive advantages in spite of the exaggerated claims being made at that time. TCA decided then to anticipate this threat and started *re-engineering* its integral information systems in order to facilitate reusability of the components and making artificial intelligence resources available in the systems. The strategy was to build a business rules repository that would become readily available to user applications that were subject to a predefined workflow and could also profit from different artificial intelligence techniques. Along the way it became evident that agent technologies were a core element in that effort. Over the last 5 years, TCA implemented its Hotel Information System (HIS) as a consolidated set of business rules available to a middleware workflow engine that reads workflow definitions and delegates concrete tasks and procedures to participating user and server agents. This modestly *MAS-ified* HIS whose architecture is reported in this paper is already operational in 10 hotels with a total of 2040 rooms. It is also the boot-strapping version of an agent-pervasive HIS that we are in the process of designing.

The paper gives a quick sketch of the conceptual background of the hotel information system and a brief description of its architecture. In section 3 we outline the implemented architecture, in section 4 we discuss some relevant results and in section 5 we mention the framework that generalizes the approach we report in this paper.

2. BACKGROUND

2.1 Organizations and Corporate Information Systems

An organization, a firm, is in essence a group of individuals that pursue their collective or shared goals by interacting in accordance with some shared conventions and using their

available resources as best they can [4, 5, 2]. Hotels, or other types of organizations, have conventions that "organize" their activity in consistent ways so that employees and clients have some certainty about what is expected of them or what to expect from interacting with members of the organization. These organizing conventions usually take the form of canonical documents, standard procedures, rules of conduct or records that participants adhere to in a more or less strict way. Corporate information systems (*CIS*s) provide organizations with powerful means to capture the way the organization works by implementing, somehow, those procedures, rules, documents and records. A *CIS* controls the processes that the organization follows, stores its experience and keeps track of the interactions that the members of the organization have with other staff members and also with suppliers, clients and other individuals who are not part of the organization.

Traditionally, *CIS*s are developed around the main business processes of the organization which are usually organized in a hierarchical system whose components —as is notoriously the case among vertical industry organizations— are tightly interrelated. Consequently, building a *CIS* usually starts by making explicit the system components, that is, the business processes of the organization and their main functionalities. Once there is complete business process definition for an organization, the current practice is to address the design and development of a *CIS* by performing three types of activity: *forms designing*, *work–flow modelling* and *business rules programming*. Although there are different ways of organizing these *CIS* development activities, the following two alternative approaches are prevalent:

- *Form centered programming.* The flow of activities is governed by the design of forms. That is, the intervening business rules are invoked by fields of a form where they read or write data, in a sequence that is determined by the form design. This approach has the advantage of easy programming, but provides no facilities to implement work–flow control on the intervening processes and there is no room for capture declarative conventions that could govern the interactions between the intervening components.

- *Work-flow centered programming.* The sequence of business rule applications is specified from a work–flow perspective: the intervening business rules are invoked by states of the the specified work–flow. This approach has the advantage of having all processes in the right sequence and with the proper follow-up. However, all links between the states and also the links with both business rules and forms have to be specified and programmed at design time, resulting into inflexible implementation of processes.

We propose a third approach that is guided by a high–level specification of how the organization is supposed to function. Such *prescriptive approach* makes explicit the *institutional* aspects of the organization and makes them operational through agents that mediate the organizational interactions that constitute the *CIS* (see Sec. 5 below). By relying on agent technologies we are able to address, separately, interaction or procedural conventions, declarative or decisional conventions, and the actual operation of the IT components of the *CIS*. The intended outcome is that , in addition to the obvious modularity and reusability advantages, such separation allows the fine-tuning of the *CIS* at different levels and at different times and therefore brings larger flexibility in the design and update of the *CIS*.

As the initial motivation could suggest, our goal is to develop the type of information systems that support the so-called Intelligent Organizations with their inherent focus on knowledge management and their need to adapt to a dynamic business environment. [1] Crudely put, with this approach we mean to enable Intelligent Organizations with agent technologies by building *CIS*s that capture corporate knowledge in an effective manner in order to support the work of people (and agents) that make use of IT resources in a distributed and dynamic environment. We intend to capture the established procedures and practices that organize *CIS* user interactions through the *performative* representation of the organization as an electronic institution. We will further encapsulate organizational guidelines, policies and decision criteria into the deliberative component of staff agents that support —or take over— tasks that are currently performed by employees of the organization. Finally, keeping in mind that we want to develop *CIS*s for vertical industries and that companies in those industries have many similarities, we will take advantage of agent technologies to deal with standard process components (like forms and business rules) and conventional IT resources (e.g. databases or display devices).

2.2 The Hotel Information System

Grupo TCA deployed its first hotel information system —*INNSIST*— in 1986. Its successors have evolved over the years and are now supporting the integral operation of more than 200 hotels in Latin America. The *INNSIST* Hotel Information System (HIS) supports the management and operation for the Front and Back-office of a hotel or hotel chain. Those management and operational functionalities are implemented as twelve modules, whose specific procedures were programmed to become the content of a *business rules repository*. The following is a list of these modules and the "business context that organize their operations":

Reservations. Individual reservation, group reservations, airline crew management, rooming list, availability control, availability forecasting.

Front Desk. Guest check-in, group check-in, hotel status, credit management, guest folios, client's account statements, guest checkout.

Housekeeping. Room status, room blockade, guest requirements service, maid work orders.

Telephone control. Tariffs catalog, schedule discounts, service costs, receipt printing, telephone switch control, maid status, line management, call accounting.

Night audit. Daily room update, automatic posting, cashiers shift review, audit sheet.

[1] An *Intelligent Organization*, as Liebowitz characterizes them [3], is a "knowledge–based organization whose business operations and internal processes are founded on knowledge competencies and the value of its products and services is given by the know–how, the intellectual capital and the technological advantage of the organization".

Sales. Sale statistics, packages and plans, travel agencies, allotments control, groups, sale forecasting.

Guest History. Guest stays history, promotional plans, guest preferences.

Club Management. Resort club services: Spa, boutique, snack bar, therapies control, special plans, tennis, golf.

Accounting. Forecasting, departmental accounting, transactions, uniform system of accounts, financial models, executive financial information.

Accounts Receivables. Statements, AR projection, balance due.

Accounts Payable. Buy conditions, payments suggestion, discount advantages, partial payments, multiple payments, payments projection.

Inventory Control. Physical inventory, item location, inventory value, inventory cost, inventory level.

Purchase Orders. Order quote, authorization levels, automatic purchase order generation, automatic orders, merchandise reception.

3. ARCHITECTURE OUTLINE

The hotel information system is implemented in a two-layer framework as shown in Figure 1. The bottom layer contains the actual domain components for the hotel *CIS*: data and business rule repositories, forms and display devices, and the corresponding server agents (for domain components) and user agents (for staff and clients that interact with the *CIS*). The top layer is formed by a work-flow engine that guides the actual execution of the *CIS*, and an organizational middleware that, directed by the specified workflow, articulates users and domain components.

3.1 The middleware and workflow engine layer

The organizational middleware runs the *CIS* by putting *business domain* elements and users in contact subject to the workflow specifications. Hence, the basic functions of this middleware are:

- to log users into the organization, controlling user roles, agent resources and security issues.
- to monitor user interaction,
- to load and interpret the *workflow specification.*

The workflow engine has two components: the workflow specification language and the workflow interpreter.

3.1.1 Workflow specification language

The commands available for workflow specification are the following:

- define TAG
- define business-rule-< id >
- define data-base-access-definition-< id >
- define report-definition-< id >
- conditional command execute on TAG
- conditional command execute on EVENT
- form handling.
  - activate function keys
  - define plain form
  - define grid
  - plain form
    - deactivate fields
    - input fields
    - display fields
    - goto field
  - grid form
    - deactivate fields
    - input fields
    - display fields
    - goto field
- call business rule (business-rule-< id >)
- call data base access (data-base-access-definition-< id >)
- call report (report-definition-< id >)

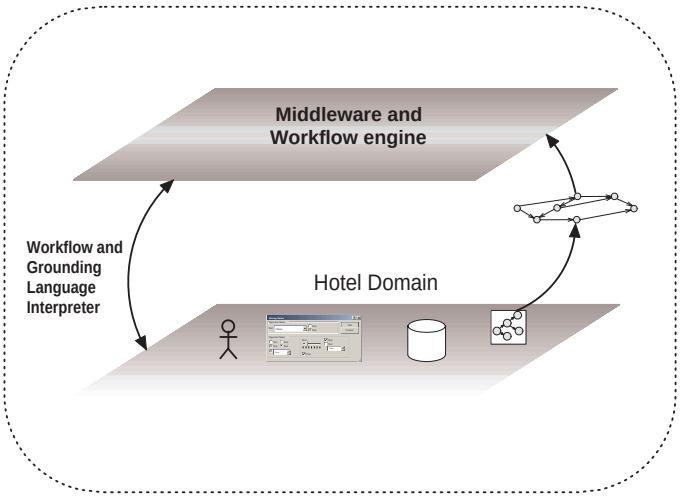

Figure 1: The workflow engine of the organizational middleware loads and interprets workflow specifications and monitors agent interactions

3.1.2 Workflow interpreter

The commands available for workflow execution are the following:

- read work flow specification
- load defaults
- initialize variables
- execute command
  - on TAG
  - on EVENT

3.2 The hotel domain layer

This layer contains the following components:

Hotel Domain Agents. We consider two types of agents:

- *User agents* (human or software) that are external users (clients, service providers and staff members of the organization) of the *CIS*, and
- *Server agents* that act as front ends for all the repositories and devices of the hotel domain and thus handle the interactions with other domain agents.

Interaction forms. These forms implement interfacing capabilities between user agents and other domain elements, e.g. form handling, data base calls, business rules triggering.

Workflow specification. Interpretable *workflow* specification that defines procedural behaviour for the organization.

Repositories. Business rules and data base repositories accessible to agents.

4. SOME COMMENTS ABOUT AGENT INTERACTIONS AND WORKFLOW EXECUTION

As indicated above, the hotel *CIS* we have implemented is run by the middleware and worfk-low engine that interacts with the hotel domain components and *CIS* users. Figure 2 illustrates how the middleware supervises the agents that handle the specialized domain components, such as data bases or business rule applications —a specialized *business rule* server agent (Bag) fetches business rules from a central repository that use data provided by another specialized *data base* server agent (Dag), to provide input to a *user agent* (Uag) that displays it in a user form.

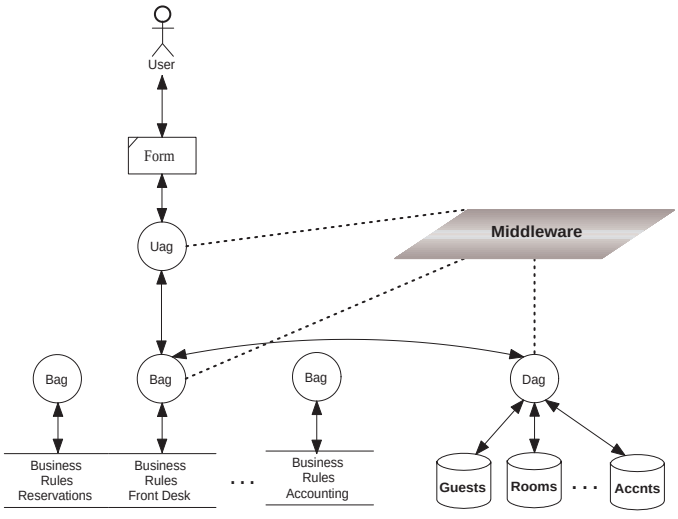

Figure 2: The middleware supervises business rules (Bag) and data base (Dag) *Server Agents* that handle all specialized tasks to serve the requirements of *User Agents* (Uag)

The middleware acts as a link between the user interface and the data base and business rule repositories, but all interactions are mediated by ad-hoc agents. The user interface is mediated by a user agent that is regarded as a client for the business rule and data base server agents. The main function of the specialized server agents is to act as a business domain components (including business rule repositories and data bases) facilitators for all user agents that may be logged-in at several client sessions. User agents can be located anywhere and in any platform; they may be implemented as independent applications, even as applications that require browsers for operation. All the communication with the middleware is made using a Tag-Value encoding via a XML protocol.

Once the interaction between the *user agent* and the *server agent* is established, the infrastructure does nothing but assuring that the communication between both agents is persistent until one of the agents decides to terminate. This is a very important point, because it is the responsibility of the agent programmer to consider all the communication contexts and to include a piece of code to deal with all the information interchange requirements. For example, in the workflow definition shown in Appendix A, once the F4 event is true and the workflow interpreter issues the command
`Interact(DataBaseSvrAg(DB_SUPPLIERS,NEW))`
it is the responsibility of the programmer to send all relevant information from the user form to the data base server agent in order to have the data base properly updated.
In a similar way, once the `SELECT_SUPPLIER` event becomes true, the workflow interpreted issues the command
`Interact(BusinessRuleSvrAg(BR_SELECT_SUPPLIER))`
and it is also the programmer's responsibility to trap the result of the business rule execution setting the value for some data or status variables. The programmer is responsible for maintaining the context of all agent interactions because as agent interaction evolves, they modify the context of the world, updating data and status variables as required. All the global variables are available in the scope of the workflow definition, that is, in the workflow specification the programmer can test for the value of variables defined as global by any *Server Agent*.

The workflow interpreter and the middleware environment impose no control over variables, neither global or local. That is, it is the programmer's responsibility to define and maintain the proper scope for the required variables. The middleware assumes that all agent interactions are between the User Agent and either the business rules server agent or the data base server agent.

Regarding workflow execution, tag and/or event value verification takes precedence over sequential process execution; that is, in the middle of a conditional execution of a TAG value, it is possible to break the sequential flow and skip directly to the first command of another conditional clause.

Regarding scope of workflow execution, once a flat form or grid is addressed, all subsequent workflow commands will be made in the scope of that specific flat form or grid, until another flat form or grid is addressed.

This system developed so far, however, has one major limitation: the workflow engine has no control over what is said between agents, as it deals only with specific conditional commands that test for contextual changes represented by changes in data and status variables. If we want to deal with complex interactions, this is an important limitation

because (up to now) we are forced to "hardwire" the control code for the execution of alternative procedures depending on what is said between agents.

In the implementations described in this paper, we only use *reactive* agents. Such a primitive implementation is enough for our current needs but we may readily change their specification to involve more sophisticated behavior. The benefits provided by the fact that they may potentially negotiate their information interchanges in a flexible way are evident. We should also note that our implementation of agents already assumes that they interact with other agents exclusively through speech acts. Such design decision will enable us to make them functional in the electronic institution environments that are under construction for the extended framework.

The middleware language outlined in this paper is being refined and extended to deal with new functionalities. We also want to go one step further and implement the concept of commitments between agents in order to provide the framework with the flexibility needed to deal with temporal constraints, that is, the effect of some interaction between agent A and agent B at time i, might commit agent B to do something at time $i + 3$, say.

The system described in this paper is currently operational in the following hotels:
– Avalon Grand de Cancún (154 rooms)
– Avalon Reef –Isla Mujeres (137 rooms)
– Blue Bay Cancún Club (310 rooms)
– Blue Bay Cancún Getaway (290 rooms)
– Blue Bay Desire Resort (110 rooms)
– Coco Beach Cancún (202 rooms)
– Karmina Palace Manzanillo (322 rooms)
– Mayan Palace Mazatlán (280 rooms)
– Radisson Casa Grande Chihuahua (115 rooms)
– Reef Club Isla Cozumel (120 rooms)

5. CLOSING REMARKS

In this paper we describe an agent-enabled information system that is being used to manage and operate medium sized hotels. In this system, agents are used as mediators between *CIS* components and system users under the control of a workflow engine.

We claim that the example indicates how current *CIS* development practices may be improved with:

- An Intelligent Organization environment that enables available information technologies resources (such as data bases, business rule repositories, data mining tools and automated decision making devices) with *MAS* technology.
- An agent oriented methodology for building complex information systems.
- Facilitating the separation of form design, workflow specification, and the programming of business rules and agent behaviour.

We are developing a framework that generalizes this example. More specifically, in order to deal with complex interactions, we want to use the theory and notions of organizations and institutions to implement prescriptive specifications that may be properly enacted. We are working with the notion of Electronic Institutions and extending the current IIIA EIDE environment to allow teleological and normative specification of agent interactions [1]. The purpose of the framework is to facilitate the building of *CIS*s that implement a high-level prescriptive specification of an organization by encapsulating the institutional elements of the organization through standard procedures and corporate decision-making criteria encapsulated in (corporate) user agents. We intend our framework to allow for the construction of *CIS*s that are flexible enough to adapt to changing requirements and business conditions. The system discussed in this paper will allow the boot-strapping of the proposed framework because, on one side, the middleware layer developed so far is being extended to deal with richer electronic institutions and, on the other side, the workflow language and interpreter, as well as the user and server agents developed so far are being extended to provide more general *grounding language* and domain agents.

Figure 3 shows our proposed extended framework. The diagram shows how our present middleware layer is to become an organizational engine that implements an electronic institution —not just a workflow engine— that controls the business domain objects. That electronic institution engine, in turn, is the implementation of a declarative description of how agents should interact or, more precisely, of the institutional conventions that will govern the business domain interactions. These developments have been partially reported already. We have reported the implementation of a *Bayesian reasoning engine* [6] and the use of this architecture in our work on *Problem Solving Plans* [8]. Finally, in [7] we have discussed a conceptual framework and outlined an architecture and its integrative elements to enact an Intelligent Organizations using the Electronic Institutions approach.

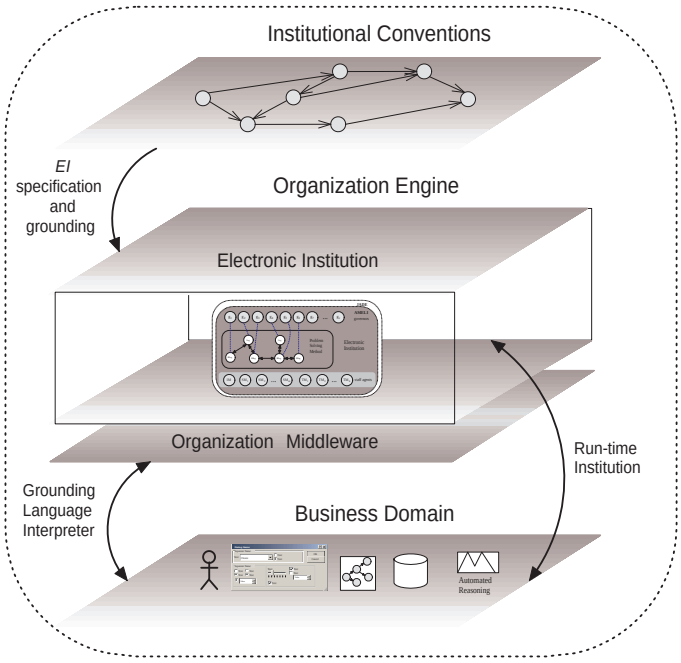

Figure 3: General Architecture using the concept of Electronic Institutions as part of the organization engine

Acknowledgments

This research is partially funded by the Spanish Ministry of Education and Science (MEC) through the Web-i-2 project (TIC-2003-08763-C02-00) and by private funds of the TCA Research Group.

6. REFERENCES

[1] J. Arcos, M. Esteva, P. Noriega, J. Rodriguez-Aguilar, and C. Sierra. Environment engineering for multiagent systems. *Engineering Applications of Artificial Intelligence*, (18):191–204, Elsevier Ltd. January 2005.

[2] R. M. Cyert and J. G. March. *A behavioral theory of the firm*. Prentice-Hall, Englewood Cliffs, N.J. USA, 1963.

[3] J. Liebowitz and T. Beckman. *Knowledge Organizations*. Saint Lucie Press, Washington, DC, 1998.

[4] J. G. March and H. A. Simon. *Organizations*. John Wiley and sons, New York, USA., 1958.

[5] D. C. North. *Institutions, Institutional change and economic performance*. Cambridge Universisy press, 40 west 20th Street, New York, NY 10011-4211, USA, 1990.

[6] A. Robles, F. Cantú, and R. Morales. A bayesian reasoning framework for on-line business information systems. In *Proceedings of the Eleventh Americas Conference on Information Systems*, Omaha, NE, USA, August 2005.

[7] A. Robles and P. Noriega. A framework for building eienabled intelligent organizations using mas technology. In M. Gleizes, G. Kaminka, A. Nowé, S. Ossowski, K. Tuyls, and K. Verbeeck, editors, *Proceedings of the Third European Conference in Multi Agent Systems (EUMAS05)*, pages pp.344–354., Brussel, Belgium, December 2005. Koninklijke Vlaamse Academie Van Belgie Voor Wetenschappen en Kunsten.

[8] A. Robles, P. Noriega, F. Cantú, and R. Morales. Enabling intelligent organizations: An electronic institutions approach for controlling and executing problem solving methods. In A. Gelbukh, Álvaro Albornoz, and H. Terashima-Marín, editors, *Advances in Artificial Intelligence: 4th Mexican International Conference on Artificial Intelligence, Proceedings ISBN: 3-540-29896-7*, pages 275 – 286, Monterrey, NL, MEX, November 2005. Springer-Verlag GmbH. ISSN: 0302-9743.

APPENDIX

A. WORKFLOW SPECIFICATION

```
// Workflow definition for the input and
// maintenance of suppliers price lists
#define_tag SUPPLIER                    1
#define_tag SUPPLIER_NAME               2
#define_tag DUE_DATE                    3
#define_tag DISCOUNT                    4
#define_tag PROMOTION                   5
#define_tag CONTACT                     6
#define_tag GRID_FIELDS                 7
#define_tag END_FIELDS                  7
#define_tag TOTAL_COLUMNS               7
#business_rule BR_CONSISTENCY_CHK      3000
#business_rule BR_SELECT_SUPPLIER      3001
#business_rule BR_SELECT_ARTICLE       3100
#data_base_access DB_ARTICLE_LIST      3200
#data_base_access DB_SUPPLIERS         3300

//
// Actions on TAG value
//

process on TAG = 0 {
   ActivateFunctions(F1);
   DeactivateFields(SUPPLIER,CONTACT);
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on TAG = SUPPLIER {
   ActivateFunctions(F2);
   DefineGrid("grid01");
   InputFields(DUE_DATE,CONTACT);
}

process on TAG = SUPPLIER and SupplierExists {
   ActivateFunctions(F1);
   DeactivateFields(SUPPLIER,CONTACT);
   GridDeactivateFields("grid01");
   GridStartInput(SUPPLIER,TOTAL_COLUMNS);
   InputFields(DUE_DATE,CONTACT);
}

process on TAG = PROMOTION {
   ActivateFunctions(F1|F2);
   // Read and show Supplier price list
   Interact(DataBaseServerAgent(DB_SUPPLIERS,SHOW));
   DefineGrid(Parameter);
   GridDisplayFields("grid01");
   GridStartInput(SUPPLIER,TOTAL_COLUMNS);
}

process on TAG = END_FIELDS {
   GotoField(DUE_DATE);
}

//
// Actions on EVENT
//
```

```
process on EVENT = F99 {
   Exit;
}

process on EVENT = F2 {
   ActivateFunctions(F3|F4|F5)
   DeactivateFields(SUPPLIER,CONTACT);
   DeactivateGrid(601);
}

process on EVENT = F3 {
   ActivateFunctions(F1);
   InitializeVariables();
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F4 {
   // Validate if all data is consistent, then
   // add the supplier's price list
   Interact(BusinessRuleServerAgent(BR_CONSISTENCY_CHK));
   Interact(DataBaseServerAgent(DB_SUPPLIERS,NEW));
   InitializeVariables();
   ActivateFunctions(F1);
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F5 {
   ActivateFunctions(F2);
   InputFields(DUE_DATE,CONTACT);
}

process on EVENT = F6 {
   // Delete supplier's price list
   Interact(DataBaseServerAgent(DB_SUPPLIERS,DELETE));
   InitializeVariables();
   ActivateFunctions(F1);
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F8 {
   GridStartInput(GRID_FIELDS,GRID_FIELDS);
   ActivateFunctions(F1|F2);
   InputFields(DUE_DATE,CONTACT);
}

process on EVENT = SELECT_SUPPLIER {
   // Show Supplier list and select one
   Interact(BusinessRuleServerAgent(BR_SELECT_SUPPLIER));
}

process on EVENT = SELECT_ARTICLE {
   // Show Article list and select one
   Interact(BusinessRuleServerAgent(BR_SELECT_ARTICLE));
}
```