Better Models for Agile Virtual Enterprises - the Enterprise and its Constituents as Hybrid Agents

Peter Bernus, Rob Baltrusch,
Griffith University
Martin Tølle, Johan Vesterager
Technical University of Denmark

Abstract

The article proposes a way to improve reference models for the management of virtual enterprises. The pattern of management roles is based on enwrapping each decision centre into an agent wrapper, using the same tools and algorithms. Using this approach enterprises (organization, networks, virtual enterprises) will have the emergent agent property – the ability to follow objectives, plan, and take timely corrective action in case the plan breaks down.

The resulting system has a dynamic, agile structure, where each decisional level (strategic, tactical, operational) may organize and re-organize the lower level according to the changes in objectives, or according to improvements in the capabilities or availability of resources.

1 Introduction

In the artificial intelligence (AI) literature, and particularly in Distributed Artificial Intelligence (DAI), agents are usually described as software programs with certain capabilities [1]. These capabilities include goal seeking behaviour [2] [3], the ability to plan [4], autonomous action [5], and reflective reasoning [6] [7] (and often some additional properties as well). In distributed AI these capabilities are extended with co-operative planning and negotiation, which includes trust between the involved agents [9] [10]. The problem is, then, how to specify, design and implement such software agents (or simply agents since this statement precludes the specification of human agents (in the form of a job role for instance). The scope of the improved Globemen Reference Model discussed in this paper is primarily aimed at supporting the management and control functions of the Partner, Network and Virtual Enterprise, since these abstract entities can each be viewed as an agent having agent properties. The notion of the Network being an agent may first seem surprising – the question being whether the Network is referred to as the Network Manager’s Office or as the collection of partners in the network, and possibly all suppliers to the Network as well. However, it must be considered that the ‘agent’ to which we refer in this article is not a physical entity. The ‘agent’ referred to here is a set of processes linked together in such a way that a competency-based and negotiated allocation of physical entities to these processes allows these physical entities jointly display agent behaviour. In the case of the Network any objective that is in the competency of the Network will be achieved through co-operation by a number of physical entities – incidentally, the same is true of a Virtual Enterprise.
The properties that collectively define an agent can be extended to any system, including a system entirely consisting of humans, or to hybrid systems, consisting of a mixture of human and automated constituents [10].

Agent properties are very attractive for any system to have. For example, if an enterprise is an agent, then it is able to determine its own objectives (perhaps through negotiation), design and follow plans of action according to certain constraints and parameters, as well as it has the ability to negotiate and co-operate with other enterprises to determine and to achieve common objectives. In case the execution of the plan breaks down, the enterprise with agent properties has the ability to change its objectives, change its plans, so as to continue a viable course of action. These properties make the agent ‘intelligent’. Thus the question is not how to implement fully automated intelligent agents, but rather, how is it possible to specify, design and organise an entire enterprise, a network of enterprises, or a virtual enterprise, in such a way that each can be considered to be an agent.

Traditional management structures have difficulty achieving agenthood on the enterprise level, because the complexity of co-ordination and planning functions precludes purely human agents to develop solutions to many management problems [10].

It has been established [10] that such emergent management structures will need to satisfy the following fundamental requirements:

*Enterprise Integration*: In order to support global competitiveness and rapid market responsiveness, an individual or collective manufacturing enterprise will have to be integrated with its related management systems (e.g., purchasing, orders, design, production, planning & scheduling, control, transport, resources, personnel, materials, quality, etc.) and its partners via networks.

*Distributed Organization*: For effective enterprise integration across distributed organizations, distributed knowledge-based systems will be needed so as to link demand management directly to resource and capacity planning and scheduling.

*Heterogeneous Environments*: supporting IT infrastructures will need to accommodate heterogeneous software and hardware in both their manufacturing and information environments.

*Interoperability*: Heterogeneous information environments may use different programming languages, represent data with different representation languages and models, and operate in different computing platforms. The sub-systems and components in such heterogeneous environments should interoperate in an efficient manner. Translation and other capabilities will be needed to enable such interoperation or interaction.

*Open and Dynamic Structure*: It must be possible to dynamically integrate new subsystems (software, hardware, or manufacturing devices) into or remove existing subsystems from the system without stopping and reinitializing the working environment. This will require an open and dynamic system architecture.

*Cooperation*: Enterprises may need to cooperate in a timely manner with their suppliers, partners, and customers in order to meet a set of negotiated objectives.
Agility: Considerable attention must be given to reducing product cycle time to be able to respond to customer desires more quickly. Agile manufacturing is the ability to adapt quickly in a manufacturing environment of continuous and unanticipated change and thus is a key component in manufacturing strategies for global competition. To achieve agility, manufacturing facilities must be able to rapidly reconfigure and interact with heterogeneous systems and partners. Ideally, partners are contracted with "on the fly" only for the time required to complete specific tasks.

Scalability: Scalability means that additional resources can be incorporated into the organisation as required. This capability should be available at any working node in the system and at any level within the nodes. Expansion of resources should be possible without disrupting organisational links previously established.

Fault Tolerance: The system should be fault tolerant both at the system level and at the subsystem level so as to detect and recover from system failures at any level and minimize their impacts on the working environment.

Even if all of the above is theoretically possible, the time and resources available to harmonise objectives and actions is often so limited that human agents alone are not capable of making correct and timely decisions. As a result the management role, or the entire enterprise, starts to display inferior (‘not intelligent’) behaviour. This harmonisation of objectives between management functions becomes increasingly difficult across organisational and functional boundaries – as is the case in the Network and Virtual Enterprise since there may be greater disparity between management functions.

2 Designing a Modular Reference Model for Virtual Enterprise Management

The improved Globemen Reference Model (based upon [11]), presented here takes this human limitation into account, and defines each management role as a role potentially filled by a hybrid agent - i.e. an individual (or a group of individuals) and their supporting automated (software and hardware) tools. This approach reflects the nature of dynamic, collaborative environments since each abstract role has insufficient resources and control to achieve it’s objectives alone, and therefore must negotiate and coordinate its activities in order to meet its objectives. For example, if a management role is to be filled by an agent (required to negotiate and co-operate with other management roles) then allocating the task to a group of individuals is likely to deliver inferior solutions. This is because a group (such as a committee) is unlikely to be able to function as an agent alone under the pressures of time and resource limitations. In addition to automated tools, each of the management roles must have certain knowledge at its disposal, and needs to have policies and procedures that collectively ensure the functioning of the group as an agent.

As a consequence, the improved Globemen Reference Model endavours to

a) Encapsulate each management role into an 'agent hull' - defining standard agent negotiation protocols [12] on the interface
b) Determine the requirements, which can be used by enterprise management to design company-specific policies and procedures that are consistent with the requirement that each management role individually (as well as the entire enterprise as a whole) should be capable of displaying agent properties.

c) Identify automated tools and reference models for management roles that improve the ability of the management role to engage in co-operative planning, control and decision making – functionally, in a timely manner, and without functional degradation due to resource limitations.

The model presented here is discussed according to these three objectives, and is proposed as a set of patterns, rather than an integrated solution. The reason for this is that each Business Model (i.e. a set of defined strategic relationships between enterprise entities, as well as each decisional system) is unique, therefore – unless we intend to limit the development of the reference model to a very narrow domain – it is not possible to define a reference model that can be applied in a wide variety of situations by simply refining it through the addition of detail.

In general there are three types (ways to define) of reference models [13,14].

1. The Reference Model (RM) may be developed as a generic model. The use of such form of RM is that particular models can be developed from it through refinement and specialisation. The advantage of such RM is that all particular models based on it share a set of properties, namely those properties that hold true for the RM. This form of RM is in fact the state of the art in industry today: all ERP systems are based on this principle. Thus ERP systems define how management should be like in any enterprise, and the implementation in any particular situation is carried out through parametrisation these reference models [14]. ERP RMs are developed to a fairly fine granularity (to allow the implementation of modules for fast deployment through parametrisation of the modules).

However, the downside of such RMs is that the variety of allowed particular models is limited. Since there is today no better alternative, enterprises are forced to accept this underlying RM of ERP systems, and make modifications to their own management system. As a result, a) the ERP system does not always fit the intended Business Model [15] [16], and may force a change that is due to the limitations of the RM rather than by some business consideration [17]; b) the ERP system implementation may meet cultural resistance, which makes successful implementation slow, expensive, and may achieve only limited success [18]. Note that we do not intend these comments as criticisms of ERP systems, rather we assess the applicability of this RM type to the problem at hand: the development of a RM for virtual enterprise management.

2. The RM model may be developed as a paradigmatic solution. What is meant by this term is that the RM is presented as an individual typical solution. Particular models would in this case be developed on its basis through changing details. This type of RM is even more limited than the generic RMs (so it would be an inappropriate choice for ERP systems, for example), i.e. a paradigmatic RM’s applicability is limited to a relatively small design space.

In spite of the above limitations, paradigmatic RMs are widely used by consultancies and engineering companies (in all fields of engineering): a set of previous cases of complex systems are documented and kept for future reuse, and subsequent particular cases reuse these either in
their entirety (through local changes to the paradigmatic solution) or in part (through cutting out part of the paradigmatic model and replacing it with a brand new part).

The advantage of paradigmatic RMs is that – provided the company using it is faced with a sufficiently similar particular case – both the design and the implementation of a new system has a substantial part that is identical to a previous solution. Thus the need for parametrisation is limited (mainly to operating parameters, rather than design parameters, i.e. the difference between particular systems is only in parameter settings of software systems, where the design of these systems is the same), the time and resource needs of a new system implementation is relatively small, and the risk is substantially reduced.

A reference model for virtual enterprise management developed earlier in the Globemen Consortium [11] is a paradigmatic model. It is developed as a model of a hypothetical case (where the requirements based on which it was developed reflect the views of a set of Globemen partners at the time).

The problem with developing an implementation of this RM is that it is not clear to what extent such implementation can be reused in the future, in other words is it modular enough? Modularity should be preserved both in the models and their implementations (called modules) to allow (re)configuration – either for future particular cases, or to change an existing system of management.

3. The RM model may be developed as a set of modular components and rules of how to combine such modules into a system. The modular components themselves have several important characteristics, including a) modules are generic, simple, discrete units enabling highly cohesive, loosely coupled configurations; b) modules have well defined interfaces, inputs and outputs; c) modules transparently produce outputs; d) modules are recursive entities, meaning they can be decomposed into a set of sub modules with the same defining properties above.

Notice that given a set of models and component modules, the design space (the variety of systems that can be configured from them) is much larger than covered by generic models through their parametrisation. At the same time, given the granularity of modular model components, there is no obstacle for designing implementations (modules) so that any new system based on this RM type could be designed and implemented through configuration. Special design principles exist to help design modular systems that allow the greatest freedom for combining modules, such as the principle of orthogonalisation [22].

Electronic engineering and construction engineering companies use this type of RM very frequently. Other domains such as software engineering (through the component based software engineering movement) attempt to use it also, but with limited success at this moment – with some exceptions (such as the UNIX operating system, which is a collection of a rich set of modules that can be combined into systems with ease). The reason for limited success in certain areas seems to be a) designing a modular system requires a larger effort than designing a particular system (even if future reconfgure needs are considered). Thus the cost is not easily absorbed by any particular system design and implementation project b) the skills required to do this (and the underlying principles) are in short supply. This is because configuration of modules can be either optimising or satisficing – that is, the design may be
functional but not robust (mainly addressing user requirements) or it may be robust, flexible, maintainable etc (addressing system requirements).

While this type of RM is very attractive, to develop such a modular RM for virtual enterprise management it is necessary to identify a principle (or a set of principles) promising enough to ensure that the resulting RM is of the right granularity. All important design decisions can be made through configuration, and modules have a detailed enough design to be able to implement them as modules. Also, learning from the limited success of modular system design efforts, we must carefully consider the extent of the domain in question, and avoid incorporating modules that are better placed on a different system layer, into an infrastructure that should be separated from the problem domain. This would allow us to limit the RM to modules pertinent to the problem domain and rely on existing transparent services through standard (or industry standard) interfaces.

3 The Agent Principle and its Applicability for Management Reference Models

As explained in the Introduction and in Section 2, our aim is to develop a modular RM that can be the basis for a variety of Business Models so that each enterprise entity should behave as an agent, as well as the complete business process implemented in by Virtual Enterprises in this Business Model should display this property.

Since in a system of management (decisional model) only management roles are defined each management role (represented by a Decision Centre) would in an organization be allocated to a manager (decision maker). It is also possible that several decision roles are allocated to one person, while it is possible for a decision role being taken up by a committee. Thus there is no on-e-to-one mapping between decision roles and individuals.

How is it possible to use the principle that enterprises should be designed so as to become agents? Is it the management role that must be an agent (thus an abstract entity), and the system of management must also be an agent, or it is that organizational entities, such as individuals, committees, departments and the complete organization must be agents? Thus we need to make the requirement of agenthood more precise.

Given an organization (such as a company) it is clearly desirable that the organization as a whole should behave as an agent. Thus the organization as a physical should be an agent (and a physical agent at that, since it is real life entity). On the other hand the decision system of the organization as a system of decision functions should also be an agent in the sense that this system of decisions should ensure that the agent property will emerge, once it is implemented in an organization. Of course the decision system is an abstract agent, since the decision system is only a system of decision functions, rather than a real life entity.

What happens when an organizational entity, such as an individual (or a group of individuals who we somehow ensured acts as an agent) is allocated to a decision task? Is it automatically ensured that the decision centre will display agent properties? The answer is simply: no. This is because the decision centre will be an agent only if the individual has the right competencies, support tools, communication channels, and information at disposal that allow the decision
function to be performed so as other agents can recognize the decision centre as an agent. For this reason, the application of the *agent pattern* to enterprise management should be done on the functional level first – where the ‘agent pattern’ is the set of functions that make up an agent’s functionality. It is only when this is accomplished that one can consider the allocation of individuals to decision roles.

Naturally for the individual (together with supporting tools or other resources) to be able to fulfill this role, and for the organization to emerge as a physical agent, certain *constraints* must be observed when role allocation is performed. Some constraints on the organizational view (allocation of individuals to decision tasks in this case) are related to competencies. Many decision tasks can only be defined as policy driven activities, rather than procedures, and these policies can only be correctly used if the individual has a certain set of competencies. Some other constraints relate to conflicts in the mapping between decision tasks and individuals: a careless allocation of individuals may create *role conflicts*, whereupon an individual has preferences that prevent it from giving equal consideration to both roles, and consequently the roles are not performed as expected. As a result, in addition to functionally designing the decision system based on the agent pattern, we must also have a set of explicit constraints that govern the construction of the organization.

Functionally (on a high level of abstraction) an agent pattern can be represented as in Fig. 1. This agent pattern represents a *negotiating* agent, since the essence of the decision system is that the objective of the enterprise can only be achieved if agents co-ordinate their activities. In a decision framework the ‘Objectives & Authorities Proposed to the Agent’ represent the objectives derived from the enterprise’s objectives, as pertinent to the given agent taking the role of this decision centre. However, as far as consistent with these objectives, other agents may propose other objectives, provided that the authorities derived from enterprise objectives allow this. The acceptance of proposed objectives is not automatic: agent co-ordination is unlike control, because the allocation of objectives is the result of a negotiation.

The Agent (the decision centre under consideration) uses its experience with previous plans to determine if – given its allocated authorities (decision variables and constraints) – the proposed objectives are feasible. Depending on the assessment of the feasibility of proposed objectives the Agent may readily accept the proposed objectives (based on previous experience the objective is feasible), or needs to develop a plan to investigate feasibility.

During this planning activity the agent may discover that the objective can not be achieved (it is not possible to generate a suitable plan) in which case the Agent may generate and alternative proposal, which either changes the objective, or requests a modification to its presently granted authorities.

Once there exist a set of Objectives and Authorities accepted by the Agent, there is a need to develop a course of action (or plan). The Agent must, using its own knowledge of other agents (potentially in a database of other agents’ capabilities and their availability) find suitable other agents for carrying out the plan. If not successful, the plan is not feasible and re-planning is necessary. If successful, the Agent must negotiate with other agents to co-operate in the performance of the plan. In the same way in which this Agent negotiated its objectives and authorities it will negotiate with potential co-operating partners. Once agreement has been reached with all necessary co-operating partners the plan is set into action and the Agent monitors (using performance indicators and status reports) the progression of the plan.
Figure 1. Agent pattern (functional view of negotiating hybrid agent)
Once again, if the plan is not progressing in a satisfactory manner, or status reports suggest a potential future breakdown, the plan needs to be revised and (some) co-operation agreements must be re-negotiated.

This pattern of agent behaviour is common to all decision centres, and is implemented by a hybrid agent – thus the Agent consists of a human and tools needed to be able to implement all the agent functionality.

Also, since this Agent is in fact unable to achieve alone the objective proposed to it, it must aggregate other agents (that possess the necessary resources and capabilities) so that this aggregate community of agents can a) perform actions necessary to achieve the common objective, and b) can modify its co-operative plan if the need arises (e.g. because of the breakdown or resources, or unforeseen events).

As a result, the aggregate community of agents together also displays agent properties, which was the objective of this design. It is important to note that there may be several partly overlapping aggregated communities, since the Network may produce several different Virtual Enterprises at any given time. However, since each Virtual Enterprise has a unique set of goals and objectives and consists of a distinct set of aggregated agent communities, the notion of agent properties still holds.

Notice that each decision centre must follow this pattern, and consequently have the same type of interface for negotiation. (The content of this negotiation of course differs from agent to agent.)

The Database of Agent Capabilities and availability may be looked at as the model that the Agent maintains about other agents. At this level of generality it is impossible to provide details of this database, but for any particular decision centre the database can of course be described.

## 4 Building a management system from modular decision centre specifications

The agent pattern can be used to describe typical decision centres, such as strategy making, strategic human resource management, strategic technology management, strategic capital management, strategic product development, strategic supply management, tactical production planning, tactical product management (marketing and supply), tactical resource management (human, technology, capital), as well as on the operational control level – customer relationship management, order processing, scheduling, factory control and supervisory control.

We envisage a set of separately specified decision centres, that we are free to combine in a specific system of management.
Figure 2. Three clusters of management function in any decision centre: (1) Negotiate aggregate objectives (2) Perform decision making task (3) Negotiate to aggregate capabilities and resources of other agents

Figure 2. shows the same pattern as Fig.1, but those tasks which are common to all decision centres are clustered into two functions (negotiation with higher authority agents and negotiation with all other agents). All the other tasks are typical decision activities, which are specific to the type of task for the decision centre.

Thus to use this model, every decision centre needs to have these two clustered functions, while the activities in-between need to be designed according to the needs of the management task.

Given that the negotiation tasks are similar for any decision centre, the collaboration and co-operation tools, as well as the reporting channels (status and performance indicators) can be implemented using the same technology, and thus common tools can be developed for this purpose.
As one can see from Fig. 2, we wrapped the decision functions into an agent envelope [10], and identified tools to support this wrapping. The two loops (I and II) are needed for the two major negotiating functions: the involvement of the Agent’s own decision making in the negotiation with higher authority, and the involvement of the negotiation with co-operating agents in the Agent’s decision making, including the need to modify plans if co-operation displays undesired characteristics. In this functional decomposition, we clustered the Agent’s negotiating functions (1) and (3) in Fig. 2, and the Agent’s own decision making function (2). The figure also shows the mechanisms which support the execution of these negotiation and decision functions (such as (ICT) tools, policies, protocols, human actors). In practice this has important consequences.

(a) The higher level authority does not need to see the details of the organization (i.e., which agent is allocated to which task – whether decision making or service to the customer), because the decision system has the systemic property of behaving as an agent. Thus either the high level objective is achieved through the lower level agents organising themselves, or the higher level agent (operating on a longer time horizon) gets timely feedback about feasible alternatives.

(b) The agent property propagates beyond company walls because the companies themselves behave as agents and their negotiation follows the same pattern. E.g. a Virtual Enterprise, created for the objective defined by a lead partner (or by a Network Organization – whichever may be the case in the given Business Model), emerges from the negotiation of the higher level agent (that defined the objective) and the suppliers that take various roles in the business process. The ‘plan’ or ‘course of action’ of Fig. 1 is in this case is the business process to be performed – such as providing after sales service for a given product or product type, or implementing a supply chain for manufacturing a given product or product type.

(c) The Virtual Enterprise is not a static organization [19] [20] [21], since the agent that negotiates the co-operative commitments of suppliers might change the overall objective (thus modify the ‘plan’, or business process), or through monitoring the performance indicators and status reports of suppliers pertinent to the business process it may conclude that role allocations must be changed (re-negotiated).

Fig. 3 shows the overall picture of this type of organization.

5 Conclusion

We propose that organizations must strive to achieve agenthood, and even virtual enterprises must have the agent property. To achieve this the management system (or decision system) of involved entities should adopt the agent pattern for implementing management roles.

For this to be achievable, existing management roles need to be enveloped by agent wrappers, which ensure a consistent way of decision making within organizations, networks, and virtual enterprises created by them.

The resulting system has a dynamic, agile structure, where each level (strategic, tactical, operational) may organize and re-organize the lower level according to the changes in objectives, or according to improvements in the capabilities or availability of resources. The
strict control channels of management systems that are designed from the top down are replaced by negotiated co-operative agreements, supported by a systematic application of performance indicators and status reports and monitoring.

Future methodological development must be done to improve and facilitate the timely implementation of Virtual Enterprises modelled after the agent paradigm. This methodological development should encompass both reference models and tool support.

Figure 3. The overall picture of a co-ordinated supply chain—each partner, and each decision role within, is an agent—these agents aggregate into higher level agents

References


