Achieving Business Process Agility in Engineering Change Management with Agent Technology

Giovanni Rimassa
Whitestein Technologies AG
Pestalozzistrasse, 24
8032 Zürich, Switzerland
gri@whitestein.com

Birgit Burmeister
DaimlerChrysler AG, Group Research
Hans-Klemm-Straße 45
71034 Böblingen, Germany
birgit.burmeister@daimlerchrysler.com

Abstract – *The importance of business processes for a successful enterprise cannot be overestimated. They are core assets through which a business turns its potential into actual competitiveness on the market. To face the challenges posed by today's changing and uncertain business environment, traditional BPM approaches are not sufficient anymore. This paper presents an approach to business process management, which leverages Agent Technology features to obtain agile business process behavior. Beyond the problem and solution description, this work presents a concrete case study in the domain of Engineering Change Management.*

Keywords: agents, business process management, engineering change management.

1 Introduction

Business processes are a fundamental component of any enterprise across all kinds of industries. Their effective setup, execution and evolution are of paramount importance to successful business operations. By definition business processes consist of a set of activities, connected in a structured whole. They describe the modes of operation of a business organization in given situations and their importance is manifold:

They constitute the *organizational knowledge* of the enterprise. The ways of operating that are captured by business processes belong to the organization, and they are made public and explicit in the face of personnel turnover and growth.

They gather and structure the *identity of the enterprise*, express its specific way to conduct business and are often the key to realizing an organization's competitive advantage. Moreover, they explicitly represent current organizational setup and are amenable to assessment and *continuous improvement*.

The whole set of activities that an organization performs in order to create, maintain, control and evolve its business processes is named *Business Process Management* (*BPM* for short). BPM is an approach to administering business processes that involves people, organizations and technologies. In addition, BPM can be carried out with varying levels of automation.

The trend toward more flexible ways of working, shorter organizational reaction times and fully embracing market and business unpredictability, along with the increase in distribution and the need to preserve understandability despite more and more complexity, characterizes the past years and shows no signs of abating.

We believe that in the face of the challenges present in today's dynamic business environments, BPM falls short of what it is commonly intended to achieve. This paper presents *agile business process management* as an effective approach to the challenges mentioned above. Moreover, the role and contribution of Agent Technology is analyzed, and the application to a concrete case is presented.

This paper is organized as follows. Section 2 presents the problem of achieving agility in BPM, with particular reference to the domain of *Engineering Change Management* (*ECM* for short). Section 3 presents the role played by Agent Technology in conceiving and realizing a solution for the problem of agile BPM. Sections 4 and 5 illustrate the major concept of the solution, namely *goal-oriented and autonomic BPM*. Lastly, Section 6 introduces the concrete application of the approach.

2 Problem Definition

Today any development project in the automotive industry has to be supported by a powerful engineering change management. The increasing complexity of the product, the shortening of time-to-market and the growing dependencies on the suppliers increase the number and the complexity of change requests in all phases of the product development. Therefore an efficient management of product changes is an important success factor.

Compared to typical business processes, e.g. in call centers or financial services, managing engineering processes is even more challenging: Engineering processes are long running tasks. Constructing a car lasts for many years. During this time period many things change – what has been an up-to-date approach in the beginning may be outdated at the end. Also, engineering processes have to cope with uncertainty because of their mixture of creative tasks, collaborative work and repeating activities. This results in very complex processes with many alternative paths and sections that cannot be planned in advance.

Traditionally, BPM systems have been developed based on a mind model of business processes as process chains or task chains. Changes, uncertainty, and hidden processes are seen (and sometimes handled) as exceptions instead as

regular events. Hence, support for the special demand of engineering processes is limited [2]. Adequate support for engineering processes in terms of modeling and execution obviously requires a completely new approach for process management that is able to deal with the requirements for flexibility, transparency, and efficiency, both in design and execution of the process.

2.1 Achieving Business Agility

A new modelling approach to enable agile processes has to

- Support the design of huge, complex processes, by using a modular process model but also allowing for an overall picture of the process.
- Decrease the effort for changing and maintaining the process model.
- Allow flexibility and agility not only in process modelling but also in process execution through software systems.

We think that agent technology can offer approaches and methods to meet these requirements. Agent-oriented software technology was first introduced to deal with large-scale, distributed software systems, which are embedded in dynamic environments, and allow for the interaction of different partners. The term "agent" is used as a name for an autonomous software component, which is able to deal with the dynamic environment and may interact with other agents [3].

Inspired by agent technology and especially by the concepts of *goal orientation* and *decomposition* the research department of Daimler has developed the idea of a *goal-* and *context-oriented* business process modelling. The main ideas of this approach are (i) to have a modular process model that describes the single steps of a process (sub-processes, activities) separate from the goals of the process and the different contexts in which the process can be executed; (ii) to have different modelling levels, for the different parts of the process model; and (iii) to have a seamless "translation" of the process model into process execution. This modular, goal- and context-based process model can then be directly executed as an agile process, by considering current goal and context when determining the next step in the process, just as realized in the BDI agent architecture (see section 3). For details see [4].

2.2 The ACM project

The feasibility of the sketched goal- and context-oriented modelling approach was first shown in a software demonstrator implemented by Daimler Group Research and applied to the area of engineering change management. This demonstrator used the JadeX agent tool as the process execution engine. JadeX is implemented by the University of Hamburg enhancing the Jade platform with a BDI agent architecture [5], see also next section.
After a successful feasibility study, which was conducted in 2005 a suitable commercial software tool had to be found to implement a system for "agile change management" (ACM) with the goal- and context-oriented approach.
The Whitestein LS/TS platform for multi-agent systems was chosen as a candidate base infrastructure, and the domain expertise and innovation-fueled vision of Daimler met with Whitestein agent technology leading offer to make this match into a joint effort. Therefore in the ACM project Whitestein and Daimler are collaborating aiming at the deployment of a novel, agile BPM system in the domain of ECM.

3 The Role of Agent Technology

As stated earlier agent technology is a specific approach to software engineering. A system is composed of a number of agents being autonomous in their behavior and interacting with each other to achieve the desired overall functionality. A specific architecture of an agent is the so-called BDI-agent. A BDI- agent is described by its Beliefs, i.e. the information an agent has about itself, its environment and possibly other agents; its Desires, i.e. motivations of the agents that drive its course of action; and finally its Intentions; i.e. the short-term goals that the agent wants to achieve, derived from its desires and external events, to which the agents wants to react. Additionally an agent has certain plans how the intentions/goals can be achieved. A plan consists of certain actions/steps that have to be executed to achieve the corresponding goal.
The BDI architecture was first implemented by [6]. The execution of the formal framework sketched above is as follows: The activities of an agent can be described as a permanent jump between two different types of actions: on the one hand the execution of basic tasks, which the agent uses to fulfill currently active goals ("execution activity"), and on the other hand the reasoning about the next basic action, which he will execute ("control activity"). Execution activities can be interacting with the environment, e.g. with the user of the system, performing some kind of computation, manipulating the agent's own data base (belief base), and sending and receiving messages to and from other agents.
A control activity results in the choice of an execution activity, which will be performed next. To find out which activity to execute next, the agent introspects its goal base, the set of possible execution activities and the belief base. From the goal base it extracts the goals, which are not yet fulfilled. Then it collects all plans, which could be used to fulfill these goals. Next, it checks which of the plans could be performed, by checking the current context (i.e. the current belief base) whether it fits to the context the plan was designed for. Different plans are designed for different contexts, which is described in the so-called context condition of the plan. Thus the agent has to drop all plans that would fulfill a goal, but only in another context. Among the remaining plans he chooses now the one he

will execute next (see Figure 3-1). The single steps of the plan are then executed as defined in the plan.

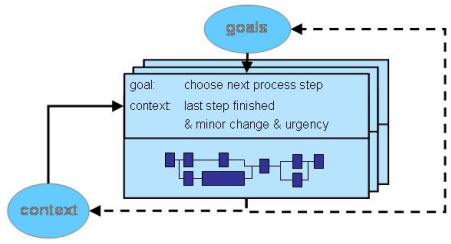

Figure 3-1: Choosing and Executing of Plans

The BDI architecture is well-established agent architecture with several agent tools and applications supporting the architecture. Georgeff also used the ideas of the BDI architecture for business process modeling and management in the Agentis platform [7].
Based on the ideas of goal-oriented and context-aware execution of agent plans, and of using it for business process modeling and execution, we have enhanced the ideas for a new form of business process modeling.

4 Goal-oriented BPM

In day-to-day management operations, it is natural to set goals, decompose a goal into sub-goals, define or reuse plans, and routinely track and check the execution of chosen plans in order to detect problems as they occur (or even better before they do), and to take appropriate actions.
On the other hand, today's dominant IT approaches focus almost exclusively on procedures. The concept of what the procedure is meant to achieve, and why, typically remains implicit in the mind of the humans who designed it. Because of this, the increase in process management automation that occurred with BPM systems has also shifted the focus away from goals and plans and toward procedures.
The limiting consequence is that processes have become more efficient in execution but less flexible in adaptation. To maintain effectiveness without sacrificing agility, the concept of plan and goal must be brought back to center stage in BPM solutions.

4.1 Plans and Goals to Express Processes

Using a goal-oriented approach separates the statement of what the desired system behavior is, from the possible ways to perform such behavior. More precisely:
The desired result is described by *achievement conditions* to make true and as *maintenance invariants* whose violation must be avoided.
The possible ways to obtain a result are represented by plans: process graphs decorated with the conditions where they are applicable and the results they obtain when successful.
In business organizations there is an upper management level, which coarsely drives the more detailed project planning and tracking. Such a level gives clear direction without unnecessarily limiting the decisional power and the adaptation leeway of the finer-grained management operations.
It is thus natural for upper managers to be more concerned with (and express their views in terms of) what is to be achieved than how to achieve it. Operating at the goal level is a natural approach for such people with the core of the business process captured through goals and sub-goals independently of the actual activities.
When moving to detailed planning in business or project management, there is usually more to the plan than just its tasks and structure. At the very least, the expected objectives of the plan need be stated, and also, in many cases, the initial requirements. Moreover, additional information such as resource and time consumption is also often attached to a plan.
To effectively tackle challenges at the organizational level, management agility has become a strategy of choice; perhaps one of the most decisive weapons in the day-to-day business world.

4.2 Keeping the Goal Level Alive

The procedural nature of computers and software must not cripple the management processes just described. In particular, a detailed, explicitly directive process specification that identifies precisely what to do in each and every envisaged variation, e.g., a BPEL execution engine, allows agility only up to a certain level.
In fact the more complex and unpredictable the situation, the more convoluted an automated directive process may become. This most often results in brittle behavior specifications that become progressively harder to extend, change and test.
To move forward and enable a BPM system to support the management of complex processes, or execute in a dynamic and unpredictable environment, both an explicit representation and a clear separation of goal and plan levels are essential.
In principle, the steps to perform goal-oriented business process modeling are:

- Expressing the intentions and requirements of the process through goals and sub-goals, connected as necessary ("*Think the end first!*").
- Organizing processes into plans by attaching them the statement of what they require and what they achieve, and grouping them when they achieve the same goals in different ways.
- Decomposing processes into tasks, specifying their aggregation structure.

Once these steps are taken, business processes in an enterprise can be modeled as a set of related goals to be achieved or maintained. One or many plans are attached to these goals, and each plan has its own feasibility requirements. Attributes such as expected completion time or resource cost can also be associated to plans.

Adopting goal-oriented BPM results in several benefits, such as:

- *Business user empowerment*. Users can work at the goal level, expressing what is to be achieved as the defining core of the business process. The details can be left out of this essential picture.
- *Increased process understandability*. The goal level alone already shows what the business process is supposed to achieve (main goal) and which are the fundamental milestones (intermediate goals). This increased understandability is also leveraged to acquire visibility of the whole process even across organizational boundaries.
- *Improved process tracking and monitoring*. The goal level allows tracking of business process evolution independently of operational details. If needed, the structure at the plan level, together with plan attributes such as cost and time, allows the continuous fine assessment of the current state of the work.
- *Encapsulation of tactics*. The set of plans attached to a given goal represents a collection of different tactics. The details of these tactics do not spark dependency chains across involved systems.
- *Lowered maintenance costs*. The widespread use of declarative specification reduces the dependence on details and makes the business process models, and their implementation, both more stable and easier to change. Moreover, plans can be reused and combined to more efficiently deal with process goals.

4.3 The GO-BPMN Language

The ideas of goal-oriented BPM are supported at the modeling and execution level by the *Goal-Oriented Business Process Modeling Notation* (GO-BPMN) for modeling processes. GO-BPMN is a visual modeling language for the specification of business processes, enriching BPMN by the explicit modeling of goals, plans and their relationships. Moreover, GO-BPMN precisely specifies the operational semantics of all its elements, including the used standard BPMN ones, so that compliant and unambiguous model execution can be obtained.

A GO-BPMN model explicitly contains elements such as:

- *Achieve goals*. They represent overall or intermediate goals that the system will try to bring about. These goals become active when some context condition is true. Achieve goals are arranged into hierarchies with a *decomposition relation*.
- *Maintain goals*. These goals are used to describe safety conditions that have to be verified at all times. Whenever one of such conditions is negated, a compensation plan is automatically scheduled.
- *Plans*. They are attached to goals and contain as body a BPMN-compliant activity. Moreover, a plan has a context condition that tells in which situations it can be executed

The Figure 4-1 shows a small sample of a GO-BPMN diagram.

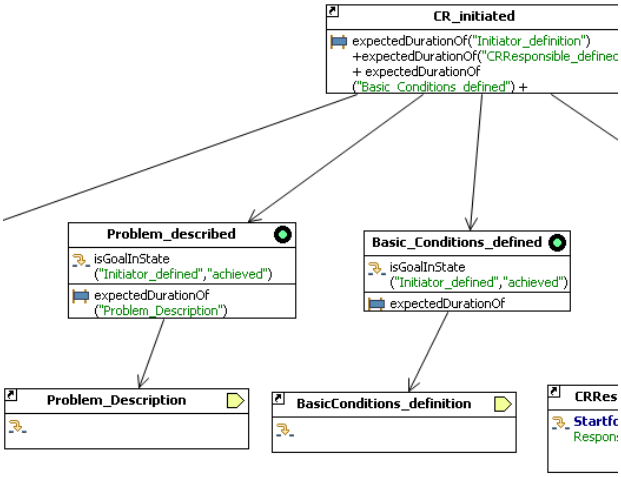

Figure 4-1 - Goal-oriented modeling with GO-BPMN

5 Autonomic BPM

Effectively managing complex business processes in the face of a dynamic and unpredictable environment requires striking a careful balance between flexibility and safety.

During operation, the definition and execution of business processes has to be easily adapted to unforeseen changes. It must also be possible to ensure that these changes are correct and do not incur any unfavorable consequences.

While some human inspection tools can and should be provided, it is unreasonable to expect that all safety controls can be handled manually. This would simply void most of the improvements in timeliness and adaptivity gained with an increased level of automation.

The only way out of this is to provide the system with means of *self-management*. This implies that the system itself (i.e., the business process management engine) is able to monitor its own operation and, to a certain extent, recognize and counteract undesirable situations. Following Autonomic Computing terminology, one can mention the major facets of autonomic, self-managing systems:

- *Self-healing*. The system is able to recover from unfavorable conditions that may result in malfunctions, by autonomously attempting to determine compensation actions and then performing them.
- *Self-optimization*. The system continuously assesses its own performance, explores possible courses of actions that would result in performance improvements, and adopts the ones that are sufficiently promising.

- *Self-protection*. The system detects threats and puts in place preventive and corrective measures to ensure correct operation even in the face of these threats.
- *Self-configuration*. The system is able to change its operating parameters to adapt to mutable external conditions, some of which may even be unpredictable at system design time.

5.1 From Autonomic IT to Autonomic BPM

The original focus of Autonomic Computing was on IT infrastructure with the targeted problem being the administration and management of complex computing environments. Nevertheless, the basic idea and the primary concepts of Autonomic Computing apply to most systems and even to organizational entities. Introducing self-management properties into applications can yield significant benefits.

Both for infrastructure and applications, a key to the Autonomic Computing vision is the presence of feedback control loops in the system. In principle, a system exhibiting autonomic self-management can be divided into:

- *A base system*, providing concrete functionality that is required to meet the system design goals.
- *An autonomic controller*, monitoring the base system and the external environment, and deciding and enacting self-management policies.

When the base system is not simply a software application, but a whole business process management system, the addition of an autonomic controller results in *Autonomic Business Process Management*.

In autonomic BPM, the "system" is the overall ensemble of software, hardware, human and physical resources, together with the norms and policies defining it. This system is the one that exhibits self-management and in particular the self-management qualities.

The benefits of the autonomic BPM approach result from the effect of self-management at various levels, such as self-healing of process activities through alternative backup tactics, or self-optimization by automatically detecting feasible remedies and proposing reasonable options to a human for selection.

In general, the above benefits can be summed up in two broad cases:

- *Self-management at the process level*. This means that the definition and enactment of the business process itself have some or all the self-management properties. There can be, e.g., special control processes that are added to the base processes. The way people are involved within the business process also has some autonomic traits.
- *Self-management at the engine level*. This means that the BPM runtime environment has self-management built into it. The BPM engine exhibits self-healing, self-optimization and other similar features.

6 Applying Goal-oriented Autonomic BPM

The two technological traits of goal-oriented and autonomic BPM, previously described in Section 4 and Section 5, are leveraged by the ACM system in a practical way.

6.1 The ACM System Architecture

The Figure 6-1 depicts the architecture of the ACM system. There the division into Presentation, Logic and Data tiers is visible. This approach represents a standard solution more and more adopted along the past ten years, and found in many installations today.

Each tier is well separated from the others and the presentation tier does not communicate at all with the data tier. The logic tier connects the presentation and the data tiers, and it is also where the complex application behavior is defined. Therefore, a lot of infrastructural issues such as communication protocols, security and resource control belong to the middle, logic tier.

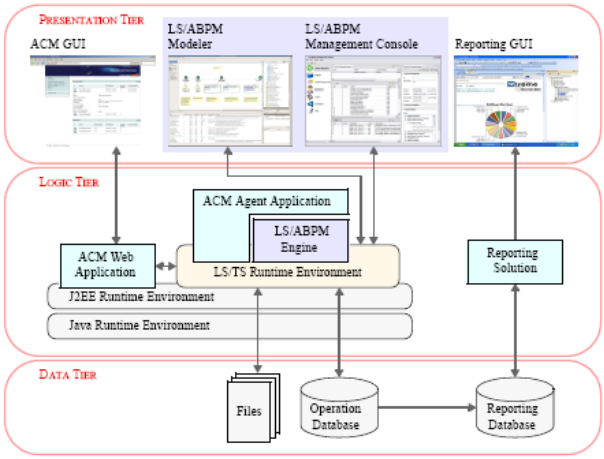

Figure 6-1 - Overall architecture of the ACM system

Instead of taking care of all these complex issues within the application, a popular and more effective approach is to rely on a support layer (called middleware, broker, or application server in different situations) that provides them. This is also shown, e.g., by blocks such as J2EE Runtime Environment and is another major best practice in modern business application development.

All the above configures a state-of-the-art architecture, but it is still not enough to cope with the agility requirement and its consequences that sit at the heart of the improvement expected from the ACM system.

While the confinement of business logic to the middle tier and the reliance on a middleware are kept, a major step forward is taken when choosing the component model for the ACM application.

The chosen component model is a major way that Agent Technology contributes to the system concept and features.

In particular, two kinds of components will execute within the ACM system:

- *Agents*. Autonomous and situated software components, capable of proactive and reactive behavior.
- *Services*. Non-autonomous software components, which are only capable of reactive behavior.

The interaction between two or more agents is based on asynchronous message passing, whereas the interaction between an agent and a service relies on the abstraction of operation invocation (which can itself be synchronous or asynchronous).

Both the agent-to-agent messaging and the agent-to-service invocation adopt a structured data model to express the topic of the interaction. Such topics are gathered in one or more ontologies, which are then made available at run-time providing advanced introspection on all the aspects of system behavior and give first-class status to entities such as resources, protocols, and organizations.

Multi-agent systems provide the right set of concepts to express and realize the loosely coupled, dynamically assembled and highly reflective architecture that can grant the desired agility and prompt adaptation to changes.

Moreover, in order to still achieve a satisfactory quality level, with particular respect to the non-functional qualities of software, such as modifiability, reliability and dependability, the approach of having multiple layered execution environments is followed.

The major point of the approach is to define a series of layers of abstraction, from the lower, more basic ones to the upper, more abstract ones. Each of these layers has two parts:

- *Execution environment*, which is fixed and sets the boundaries that define the abstraction layer.
- *Execution specification*, which can be changed, and defines the system behavior within the boundaries of the execution environment.

The execution environment acts as a containment envelope for the execution specification, preventing it from affecting other layers of the system.

This layering allows striking a good balance between flexibility and safety in modifying the system. Different users, with different skills and focuses, can operate at one or more of these abstraction levels. Concretely, in the ACM system these are:

- Final business process models described with the GO-BPMN graphical executable modeling language, for highest abstraction and safety. Both business and IT modelers can use the language effectively.
- GO-BPMN reusable modules, containing parametric goals and plan sets to be configured for application in several process models.
- High level scripting language to quickly express more complex behavior, within a well isolated programming environment, providing an easy to use API for most of the system functionalities.
- Java API to implement the core parts such as atomic tasks or intermediate service components (e.g., for specialized system integration needs)

6.2 Process Modeling with the ACM System

As described in [4] first experiences with the goal-oriented modeling approach where made during modeling the ECM process for the research demonstrator. It proved useful to concentrate on the “what should the process achieve”, i.e. the goals with process analysts. When talking to IT-people about “how it should be done”, the concrete sub-processes and all the detailed context specifications could be modeled. As a result a goal-hierarchy of the ECM process was built up. The first challenge in the ACM project was to translate this goal- and context-oriented model of the ECM process into the Whitestein platform. Since the common underlying agent technology, with the BDI agent execution engine included in the LS/TS middleware, this was rather straightforward to achieve. The Whitestein platform and idea of goal-oriented BPM turned out to be the “perfect match” for the idea of goal- and context-oriented BPM. The model could easily be built up with the modeler. Moreover, the seamless transfer from modeling into process execution was demonstrated as expected.

Also the ability to change or enhance the process model quickly and easily was used. Today, once a business process is modeled and realized as a system, it is rather hard to change the model and system. Normally bi-annual release dates exits for new system releases. This may be too long, if the process and system have to change quickly, because, e.g., a new accelerated process has to be used soon due to business requirements. Due to the seamless translation of process model to execution a changed process model can be transferred to the execution within short time, and any new processes can take the new model.

Beyond the operational goals (i.e. to realize a change in a car) several other goals where identified during process analysis. These goals are “goals to be monitored during execution” like e.g. the time of the process is below a certain limit, or cost should not increase a value. An agent, according to the ideas of autonomic BPM discussed in Section 4, can autonomously monitor these goals.

7 Conclusion

Business processes are of paramount importance to the successful operation of a modern enterprise. While the field of BPM has introduced noteworthy progress in the computer support for handling business processes, more advanced approaches are necessary in order to meet the challenges of business agility.

The ACM system has to effectively deal with a complex and challenging set of requirements, bringing an agile but dependable software infrastructure to the Change Management process at Daimler. Change Management is a critical activity to keep a constant product quality and customer satisfaction level while operating in a competitive and dynamic market.

The technological leverage of Agent Technology, together with the combined concepts of goal- and context-orientation as well as Autonomic Computing, allow the conception and realization of an advanced BPM product, and of an innovative application in the Engineering Change Management domain at Daimler, such as the ACM system is going to be.

References

[1] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, S. R. White: "A Multi-Agent Systems Approach to Autonomic Computing". AAMAS 2004: 464-471

[2] T. Beuter: Workflow-Management für Produkt-entwicklungsprozesse. Dissertation Universität Ulm. (2002) (in german)

[3] N.R. Jennings, M.J. Wooldridge (Eds.): Agent Technology – Foundations, Applications, and Markets. Springer. (1998)

[4] B.Burmeister, H.-P. Steiert, T. Bauer, H. Baumgärtel: „Agile Processes through Goal- and Context-oriented Business Process Modeling". in: J. Eder, S. Dustdar et al. (Eds.): BPM 2006 Workshops, LNCS 4103, Sprnger, 215 – 226, 2006.

[5] L. Braubach, A. Pokahr, W. Lamersdorf: "Jadex: A BDI-Agent System Combining Middleware and Reasoning". In: 18.R. Umland, M. Klusch, M. Calisti (Eds.): Software Agent-Based Applications, Platforms, and Development Kits. Whitestein Series in Software Agent Technology. Birkhäuser. (2005)

[6] A.S. Rao, M.P. Georgeff: "BDI Agents: From Theory to Practice." In V. Lesser (ed.) Proc. 1st International Conf. on Multi-Agent Systems. MIT-Press. (1995)

[7] Agentis Software: Adaptive Enterprise™ Solution Suite. http://www.agentissoftware.com