3

Goal-Oriented Autonomic Business Process Modelling and Execution

Dominic Greenwood and Roberto Ghizzioli
Whitestein Technologies AG
Switzerland

1. Introduction

Business processes are essential components of all enterprises and the use of models, languages and execution engines as components of a Business Process Management (BPM) deployment are now commonplace. By definition, these processes describe an enterprise in terms of its organizational knowledge, structure and activities, and are often essential to realizing an organization's competitive advantage. It thus follows that the design, execution and, critically, responsiveness to change in the system or environment they are affecting, is of prime significance toward establishing and maintaining efficient business operation.

Deploying a high quality and effective Business Process Management System (BPMS) is thus utterly essential to many modern enterprises. Yet current trends toward flexible methods of working, just-in-time organizational reaction times, distributed intra-organization and inter-organization collaboration and constantly changing markets are creating new and complex business landscapes. This brings about increased complexity, further motivating the need for real-time dynamic change throughout an enterprise's business processes; ever more dynamic environments require key business processes to be more flexible and automated in both their design and behaviour.

Yet many companies are now discovering that investments in conventional BPMS often suffers from poor return on investment due to a common inability to create business process models that are both meaningful to business people and capable of offering the real-time process flexibility and rapid process adaptation required to cope effectively with the fluid business conditions typifying many modern enterprises. As evidenced by our work with customers in the manufacturing domain, there is very often a need to alter executing process structures, sometimes in real-time, without perturbing the integrity of running process instances. If a BPMS is not built to innately support change in this manner the result can be reductions in both dependability and visibility, especially from a management perspective. Our observation is that many of the current procedural approaches to BPM are too inflexible and unresponsive to change, especially in any automated fashion.

In fact many BPMS solutions provide only design-time modelling, with neither support for run-time determination of process structure, nor direct execution of industry standard Business Process Modeling Notation (BPMN) process models without the need for first translating BPMN into intermediary formats, such as the Business Process Execution Language (BPEL), in preparation for execution. In practice, these issues imply that process models can tend to become overly complex and brittle through the necessity of coding-in all

Source: Multiagent Systems, Book edited by: Salman Ahmed and Mohd Noh Karsiti,
ISBN 978-3-902613-51-6, pp. 426, February 2009, I-Tech, Vienna, Austria

possible options at design-time due to the inability to change dynamically once in execution (Cordoso, 2006).

The alternative approach outlined in this chapter is to employ the notion of goals, which as recognised by other BPM vendors (Tibco, 2006) and practitioners (Benfield, 2006), are an intrinsically powerful and intuitive means to model business processes. We therefore propose an extension to standard BPMN to support the concepts of Goals and Plans, and moreover introduce an industry-validated process execution engine based on autonomic technologies, capable of directly executing Goal-Oriented BPMN (GO-BPMN) models and most importantly allowing safe, real-time alteration of both models and executing process instances. The GO-BPMN language is detailed in section 2.

The starting point for the goal-oriented approach was an observation of business management at the executive level which is typified by the assignment of achievement goals and decision points. This is also true at operational levels, but the degree of abstraction diminishes as the concrete knowledge of how to achieve goals and decision points is introduced through pre-established, or ad-hoc, processes. For humans it is natural to set goals, decompose goals into sub-goals, and to define or reuse plans to achieve those goals. This also extends to routine tracking of plan execution to detect problems as they occur, or even better before they do, in order to take timely and appropriate actions. On the other hand, computers are more easily instructed by providing them with fixed procedures. This is why many BPM solutions tend toward procedural automation where explicitly directed process specifications describe precisely which actions to take in all envisaged situations (such as with BPEL). This results in processes that are efficient in execution yet with limited expressivity and responsiveness to change.

To maintain effectiveness without sacrificing agility, we posit that the concepts of plan and goal be brought to center stage in BPM solutions. Our approach therefore uses a goal-oriented business process specification that offers a clean separation between the goals to be achieved (or maintained) and the set of task plans used to achieve or maintain them. This results in the creation of BPM deployments that are intrinsically capable of handling higher levels of complexity and change using directly executable goal-oriented process models whose structure can encode multiple degrees of freedom supporting real-time decision-making.

Goal-orientation offers a powerful, visually intuitive method of modelling and executing processes accessible to business managers and process analysts alike. Processes are described as goal hierarchies, with every leaf goal linked to one or more plans describing that part of the overall process to be executed in order to achieve the goal. Because plans can be selected at run-time, flexibility is built-in to the process structures allowing workflows to be altered safely in real-time without any need for halting or re-starting the overall process.

Autonomic Process Execution offers process responsiveness to change by creating feedback loops not only between the process engineer and the process model, but also between the underlying systems (human or computational) affected by the process tasks.

The purpose of this chapter is thus to present details of our approach and the Living Systems Autonomic Business Process Management (LS/ABPM) suite (Greenwood & Rimassa, 2007). To illustrate deployment, a current large-scale business case is described wherein Daimler AG is using the technology to manage their entire Engineering Change Management (ECM) and Procurement process catalogues. The ECM (Habhouba et al., 2006) aspect spans the documentation and execution of processes for the description, analysis,

decision and implementation of changes to all products - from individual parts to assembled vehicles. ECM is a mission-critical business operation for Daimler that can reap substantial cost reduction and time-to-market benefits if handled effectively, but it is also subject to dynamic and unpredictable environmental effects, many of which do not fall under the direct and complete control of the enterprise.
Subsequent to this Introduction, section 2 of the chapter outlines our innovative approach to business process modelling, known as Goal-Oriented Process Modeling. Section 3 then describes how these models are directly executed using the Autonomic Process Navigation Engine with section 4 introducing some of the support tooling. Section 5 describes the current business case with Daimler AG. The final section offers a discussion and conclusions.

2. Goal-oriented process modeling with GO-BPMN

The visual modelling language Whitestein Technologies has created for specifying goal-oriented process models is called the Goal-Oriented Business Process Modelling Notation (GO-BPMN). This language represents an enhancement of standard BPMN with support for the explicit modelling of goals, plans and their relationships inspired by mental modelling as defined in the Agent Modeling Language, AML (Cervenka & Trencansky, 2007). This unique combination of declarative modelling of business goals with procedural specification of business processes offers a flexible way of modelling processes that is directly applicable to the design of agile business processes with support for dynamic variation in their path of execution.
In GO-BPMN a process model consists of goals and plans structured into one or more hierarchies as illustrated in Figure 1. Goals represent objectives to be achieved, and plans represent the activities to be performed in order to satisfy a goal. For example, in Figure 1 the three plans represent three different ways to achieve goal A. Plans contain BPMN workflows of activities performed by the process engine. For human-executable activities, a mapping with an organizational model is provided. The determination of which goals will be activated and which plans will be selected and executed at runtime for a specific instance of a process model depends on the values of context conditions associated with process goals and plans.

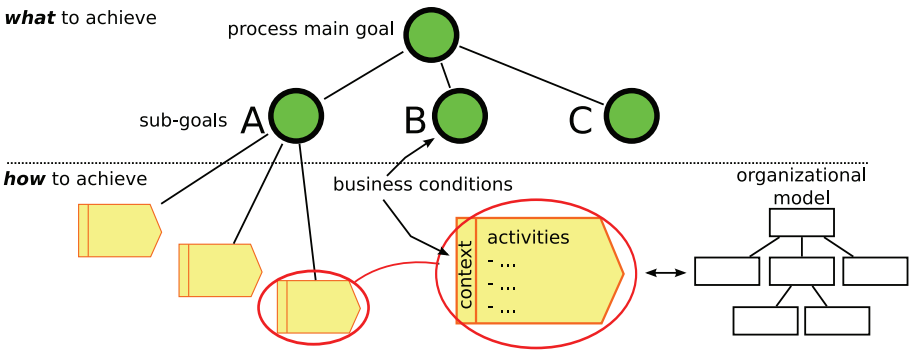

Fig. 1. The Goal-Oriented Modelling Elements

In GO-BPMN a whole process model can be divided into several independent *modules*, which represent re-usable, encapsulated, parameterizable and independently executable parts of the business process.

2.1 Goal concept

The common understanding of the term 'goal' is the result or achievement toward which effort is directed. The goal concept in GO-BPMN is essentially identical, in that goals define the states that must be reached or maintained by the process during its execution. GO-BPMN defines two types of goals, *achieve* and *maintain* goals. An *achieve goal* represents an explicit milestone, objective, desire, etc. that must be reached by the process during its execution, whereas a *maintain goal* represents the need to keep a particular condition in a persistent state, i.e., true. Goals can be composed into hierarchies with any goal considered complete only when all the associated sub-goals succeed.

A business process modelled in GO-BPMN can be seen as a set of goal hierarchies that have to be achieved or maintained. Only leaf goals in a goal-hierarchy have connected one or more *plans* which contains the activities to be performed toward achieving or maintaining the goal objective. Goals have business conditions, or rules, that control their execution, and thus the execution of the process. For example, achieve goals are defined in terms of pre-conditions, expressions that must be evaluated to true before the achieve goal can become active. For more details about business conditions please refer to Section 2.4.

When a GO-BPMN model is executed the modelled goals become stateful and the LS/ABPM Process Navigation Engine strives to achieve them. The possible states for achieve goals are: *inactive*, *ready*, *deactivated*, *running* or *failed*. The possible states for maintain goals are: *inactive*, *ready*, *running* or *deactivated*.

2.2 Plan concept

A GO-BPMN *plan* contains the BPMN-encoded specification of the functional activities to be taken toward achieving or maintaining a goal. A functional activity, called *task* in GO-BPMN, can be either human- or machine-executable. The LS/ABPM Process Navigation Engine automatically performs machine-executable tasks and issues ToDo actions to process participants for human-executable tasks. End-users, through a front-end, can browse their ToDo list and perform the required tasks.

The BPMN encoding used for plan bodies employs all standard elements of the language including flows, control gateways, sub-processes, tasks, BPMN start events, end events, intermediate events, transactions etc. Figure 2 shows a very simple example of GO-BPMN plan.

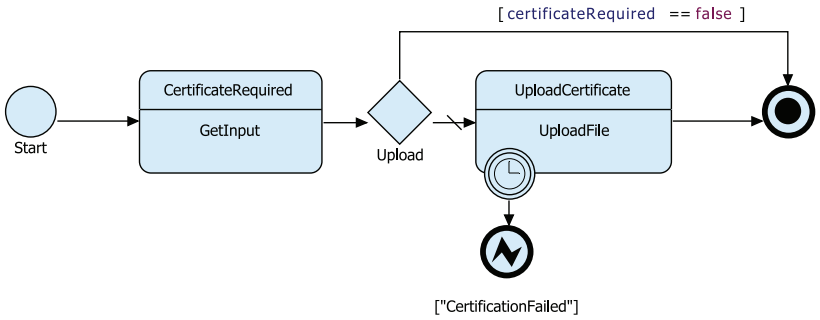

Fig. 2. Example of GO-BPMN Plan

GO-BPMN elements can be aggregated into *libraries*, with a *Standard Library* shipped with the LS/ABPM Suite providing GO-BPMN tasks, functions and data types elements for

general-purpose activities. According to their domain and requirements, application-specific libraries can be created using the LS/ABPM Standard Development Kit.
During process execution, if a leaf goal has more than one plan, a selection of the most suitable plan based on the business conditions attached to the plans is performed by the LS/ABPM engine. The possible states for plans are *not triggered*, *running* or *finished*.

2.3 The GO-BPMN expression language

LS/ABPM employs an expression language to write business conditions, triggers, task parameters, conditional flows, etc., that control the execution flow of a GO-BPMN process.
The GO-BPMN expression language is a strongly typed language. It offers two kinds of data types: **built-in types** and **user-defined types**. The built in types are *Object*, *Null*, *Boolean*, *Decimal*, *Integer*, *Date*, *String*, *List*, *Set*, *Map*, *Reference* and *Closure*. The user-defined types are *record* types, that is, types combined through a Cartesian product operation, resulting in new types. For examples, the business objects, which hold the application data, are represented as record types. Record types also support sub-typing as a partial order on types used to express if a type (subtype) is substitutable for another (supertype).
The language also supports the definition of **variables**, that is, typed storage slots declared at compile-time. In GO-BPMN variables can have different scopes, that is, different visibility. The possible scopes are *GO-BPMN modules*, *plans* or *BPMN sub-processes*.
Application-specific **functions** can also be defined. Functions are composed by a declaration and an implementation. Function bodies can be written in Java, Groovy or using the expression language itself.
Within LS/ABPM process models other **named elements** can be referred to using *identifiers*. For example, process model names, goals, plans, organizational structure elements (see later), are automatically reflected into identifiers of appropriate names. Other named elements are modules names, module imports, module parameters, localization entities, etc.
Typed variables, functions and named elements can be used in conjunction with language operators (e.g., mathematical, relational, logical, etc.) to construct expressions. In GO-BPMN, all expressions have an expected type. For example, the expressions used to define goal and plan conditions expect a *Boolean* type, *Catch Signal Intermediate Event* filters expect a *Closure* type *{Object: Boolean}*, etc.

2.4 Business conditions

The conditions associated with goals and plans are used to control the execution flow of process models. They are evaluated at runtime ensuring that executing process instances remain flexible and responsive to changes in their operating environment. The conditions supported in GO-BPMN are:

- *Pre-conditions* for achieve goals: if *true*, it runs the associated goal. Sub-goals or plans are then triggered.
- *Deactivation conditions* for achieve goals: if *true*, it deactivates the goal. Such goals can be re-activated using appropriate Standard Library tasks.
- *Maintain conditions* for maintain goals: if *false*, it runs the associated goal.
- *Context conditions* for plans: if *true*, the plan is considered as selectable by the Plan Selection Algorithm.

For example, a pre-condition for an achieve goal could be something like *MyGoal.state == achieved ()*, i.e., the engine tries to achieve the goal only when the referenced goal is finished.

MyGoal.state retrieves the status of the goal whereas *finished()* is a Standard Library function that represents the achieved, failed or deactivated goal states. The obtained result is that two objectives are achieved in sequence.

2.5 Organizational structures

An essential feature of LS/ABPM is the ability to route work to the correct process performers (e.g., workers, business roles, organizational units, business experts, etc.). This is achieved by:

- *Modelling organizational structures*: a feature of GO-BPMN for visually modelling selected organizational structures - organization units, roles, and their relationships (see Figure 3).
- *Managing Users*: managing process participants (i.e., the human workers who perform human tasks), the specification of their properties, and the connection of users to the defined organizational structure model(s).
- *Mapping of organizational models to process models*: specifying the task performers responsible for the execution of human tasks.

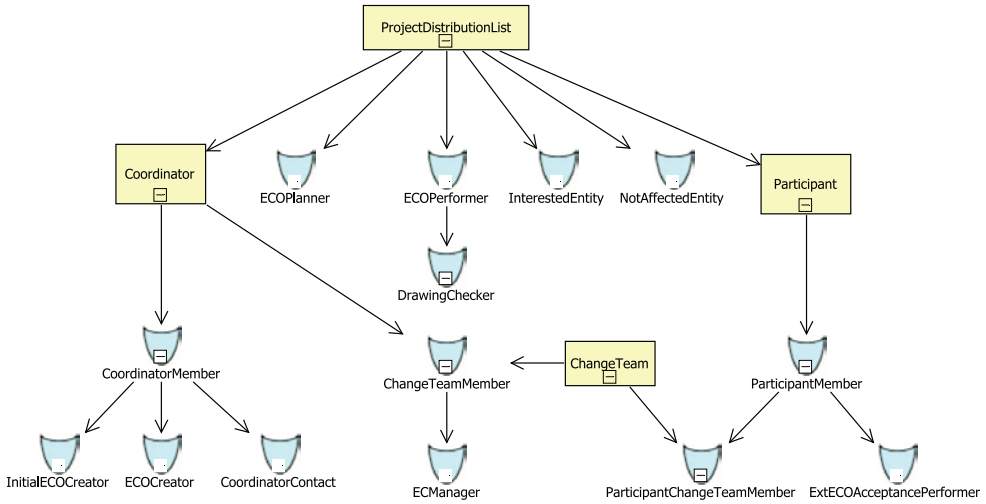

Fig. 3. Example of a GO-BPMN Organizational Model

LS/ABPM also supports escalation. It represents the set of activities that should be performed when a human-activity cannot be accomplished for several unpredictable reasons

3. Autonomic process execution

Once a process model has been created it is loaded into the autonomic process navigation engine for execution. Note that in this respect the model itself is directly executable with no requirement to translate it via an intermediary representation such as BPEL. The engine is composed of two primary computational layers, as illustrated in Figure 4: the LS/ABPM process navigation engine and the Living Systems technology Suite (LS/TS) (Rimassa et al.,

2006) autonomic middleware platform. The middleware layer executes directly over any J2EE compliant application server and can be seamlessly scaled across multiple machine clusters as demanded by deployment criteria.

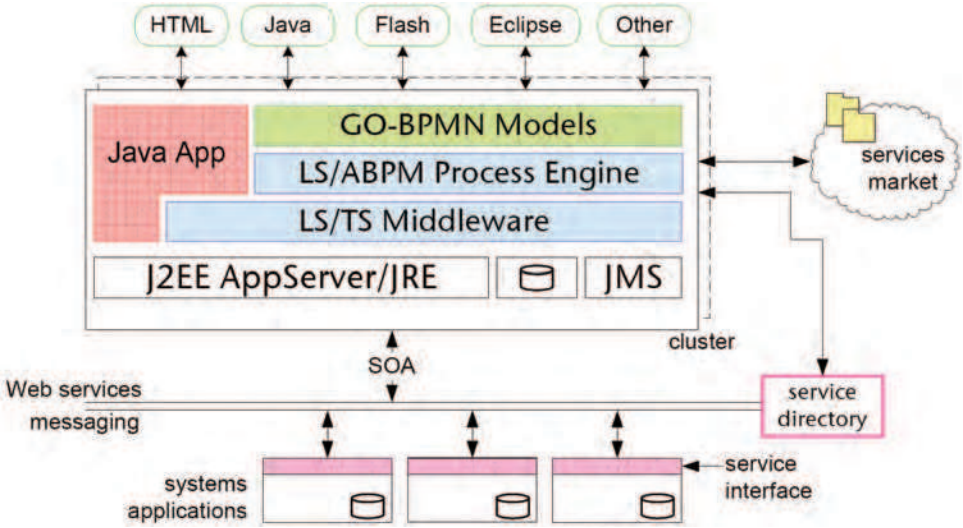

Fig. 4. Autonomic Business Process Navigation Engine system architecture

3.1 LS/ABPM process navigation engine

The LS/ABPM process navigation engine is an application developed for the LS/TS middleware runtime, consisting of a collection of goal-oriented agents acting as process instance controllers. An agent controller is assigned to each process instance, responsible for coordinating the process algebra and task structuring within goal-plan combinations, taking into account goal and plan preconditions.

When a process model is created using the Process Modeler it is directly loaded into a new process controller agent, wherein process goals are mapped onto logical goals within the goal-oriented execution engine (see section 3.2). The controller then executes the process instance by initiating the entire goal hierarchy and waiting for appropriate triggers to be sensed within the system environment to activate goals and move forward with process execution.

Each process controller is at the heart of its own autonomic feedback control loop (Pautasso et al., 2007; Tesauro et al., 2004) which uses observations made of the *system*[1] being affected by the process instance to effect decisions within the corresponding process instance relating to, for example, which goals should be activated and which plans selected to meet goal requirements. Such autonomic control allows process instances to be self-configured and self-optimized bringing about both process flexibility and resilience.

[1] The system may generally include software, hardware, human and physical resources including the constraints and policies defining their use.

Run-time execution agility is achieved as illustrated in Figure 5 wherein one of several alternative paths of execution may be taken by navigating through the goal-plan hierarchy in real time. For instance, in this example sub-goal B is satisfiable by any one of three available plans, the third plan being selected in this case according the state of a particular context parameter. In this manner the execution path of the process is determined as each goal becomes active, with context variables asserting decision criteria when multiple plans are available to satisfy any given goal.

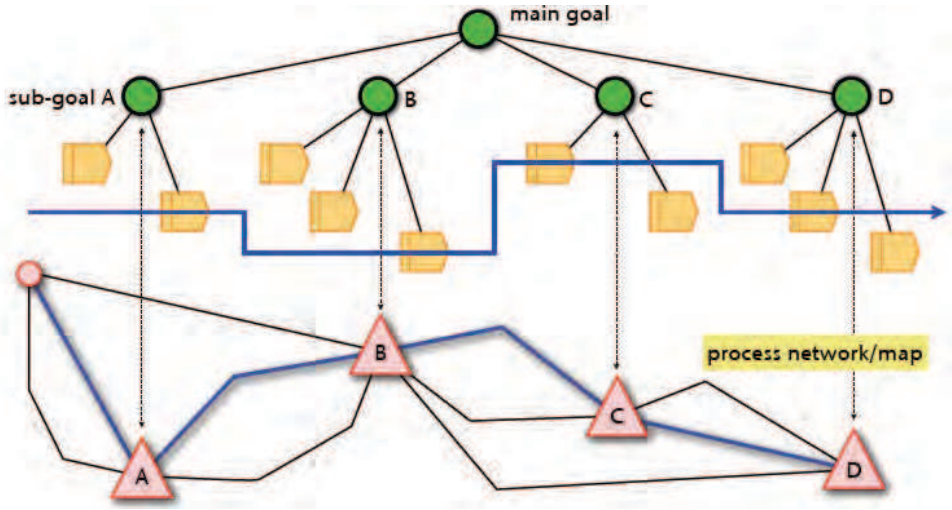

Fig. 5. Run-time agile navigation of an executing process instance

Interactions between executing process instances are managed via signal-based communication between the process controllers responsible for those instances. This is local if the instance is managed by the same controller and remote if not. Interactions can be simple bindings between the goals and plans of different processes or more complex (potentially semantic) relationships coordinating the activities of more that one controller. When multiple process instances are interacting, the influence from autonomic feedback loops is carefully monitored and controlled to ensure all effects are traceably causal and without unexpected side-effects.

3.2 LS/TS middleware

The LS/TS middleware (Rimassa, et al., 2006) is a J2EE/SE compliant runtime and associated tool suite for the development and deployment of autonomic applications driven by multi-agent technology. The runtime hosts the agents and services that define an application, providing them with life cycle support, messaging, persistence, resource management, monitoring, and more.

The tasks an agent can perform are represented as first-class objects that can be assembled into structured compounds and can be reasoned about prior to executing them. A process-algebraic model is employed to drive this task reification and assembly: tasks become Java objects and their composition follows the grammar and semantics of the operators of suitable process algebra. Building on this basic model one of the key execution engines

available to application designers is the goal-oriented execution engine, inspired by the Belief-Desire-Intention (BDI) concept (Rao & Georgeff, 1995). Using this engine, software agents maintain their own belief base of logical formulae describing their observations of the world[2], and a set of desired goals, also expressed as logical formulae, which identify the states that the agent should attempt to reach. A goal is committed to according to belief revision from world observation, at which point it becomes an intention of the agent to satisfy the goal by dynamically selecting suitable plans from plan libraries located either internal or external to the agent. It is this observe-decide-act loop intrinsic to the execution engine that allows applications to be developed that exhibit autonomic features such as self-management.

Some of the benefits of goal-oriented programming are (i) implicit programming whereby application logic is resolved only a run-time when goals are activated and plans selected according to world observation (e.g., detection of available resources), (ii) encapsulation of multiple potential strategies denoted as plan, and (iii) intuitive means of comprehending what an application should do, especially when a logical link exists to the expression of goals at the user level as is the case with the goal-oriented process modeler of LS/ABPM.

4. The LS/ABPM suite

The LS/ABPM Suite is currently composed of five main components: the *Process Modeler* for goal-oriented process modelling, the *Process Navigation Engine* for autonomic process execution, the *Management Console* for process deployment and administration, the *GO-BPMN Standard Library* to model quickly and efficiently and the *Standard Development Kit* to easily build solutions for specific needs. All components are based on standard technologies such as J2EE application servers, JMS, DBMS, and the Eclipse environment.

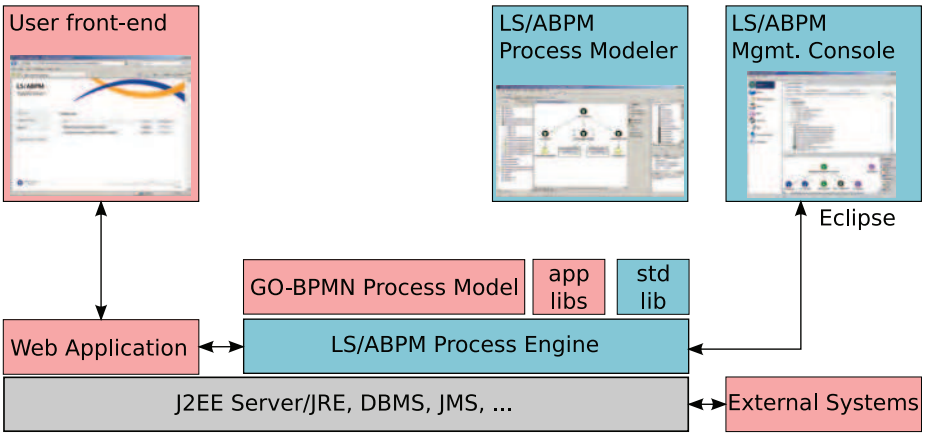

Fig. 6. The LS/ABPM Suite's Components

[2] The 'world' in this respect implies the system environment, e.g., process execution environment

For LS/ABPM-based solutions, further components can be implemented using the provided SDK. These components include: interfaces with external systems to be integrated with the process, the interface with the process participants (e.g., a Web application), a set of domain-specific GO-BPMN libraries, and the GO-BPMN process models to be executed (see Figure 6).

4.1 The process modeler

The *LS/ABPM Process Modeler*, built using the Eclipse framework, offers facilities for model creation, validation, and deployment. Using this component, process designer are allowed to model in GO-BPMN their business processes, applying all the concepts presented in section 2.

Figure 7 shows a screenshot of the GO-BPMN Process Modeler. This offers several panels including a process explorer, visual editors with tool palette, outline list of all created GO-BPMN elements, model problems list, model element properties, and more. As models can often be large, the model editor panel supports model folding whereby hierarchies can be collapsed or expanded across multiple views to ease navigation.

The process modeler is equipped with several tools to simplify model design, including model refactoring. In particular it allows the renaming and the moving of GO-BPMN elements across multiple GO-BPMN modules. Furthermore, the Modeler enables team working, that is, it is possible to share the resources of GO-BPMN process models via CVS or SVN. To facilitate task development the LS/ABPM Process Modeler allows the auto generation of Java source code from task declarations.

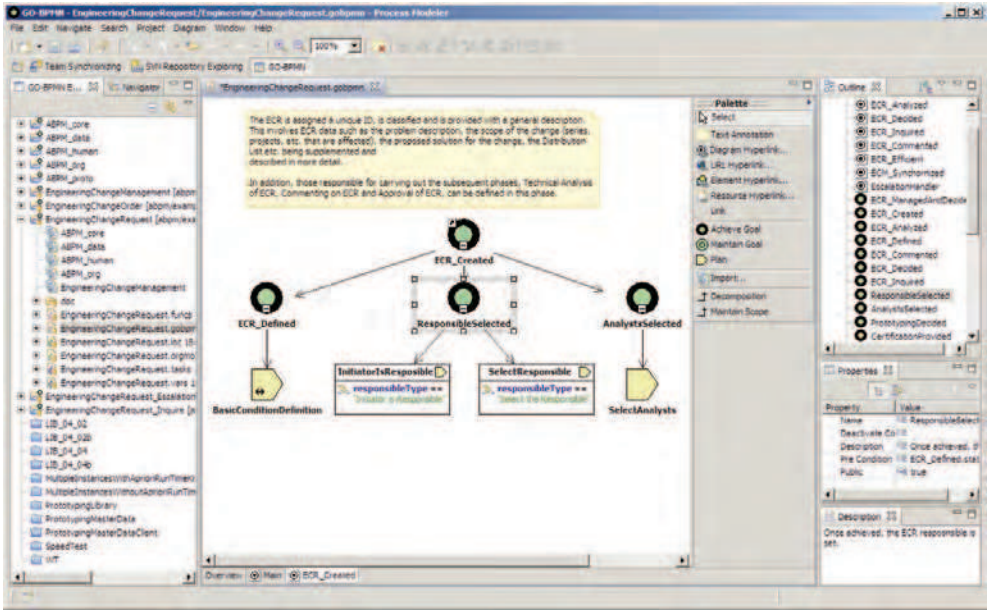

Fig. 7. The LS/ABPM Process Modeler

Every model created with the Process Modeler is automatically validated whenever the model is saved to disk with any detected problems reported in the Problems panel.

4.2 The management console

This component of the LS/ABPM suite provides powerful tools for the deployment, management and control of processes and other system administration tasks (see Figure 8).

One of the main features of the Management Console is to enable process instance monitoring, controlling and debugging. At any point, a process administrator can explore the values of context variables, the states of goals and plans and evaluate GO-BPMN expressions. It can also alter the execution of a running process by changing the values of context data, activate or deactivate goals or updating the process model a running process instance is using. Additionally, process instances can be debugged, that is, the administrator can insert breakpoints related to goal states, changing of context variables and flow selections.

The Management Console contains other tools for browsing archived processes, deploy process models into the execution engine, manage the process participants and their roles, managing security rights, managing the ToDo actions assigned to the end-users and handling logs and exceptions.

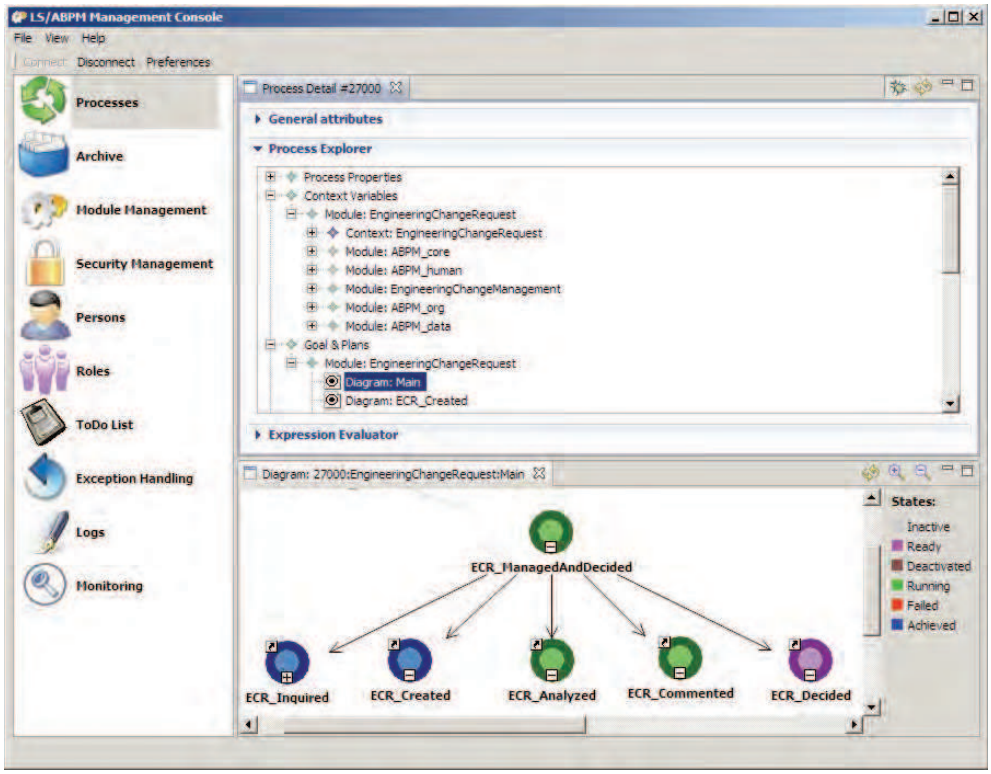

Fig. 8. The LS/ABPM Management Console

4.3 Process navigation engine

The *LS/ABPM Process Navigation Engine* is an application developed for the LS/TS middleware runtime, consisting of a collection of goal-oriented agents acting as process instance controllers. An agent controller is assigned to each process instance, responsible for coordinating the process algebra and task structuring within goal-plan combinations.

LS/ABPM runs within a J2EE environment and it is available in two distributions: a self-contained modeling suite and an enterprise suite deployable in 3rd-party J2EE servers (e.g., IBM WebSphere). The engine is capable to interact with Web applications that are used as front-ends with the process participants. Additionally, the engine can also be interfaced with external systems that have to be integrated into the modelled business processes.

4.4 Standard library

The *LS/ABPM Standard Library* contains a set of modules provided by the LS/ABPM Suite. The standard library defines the data types, functions, operation overloading and task types applicable in the following areas: reflection of the process and organization structure models, process status changing and control, process management, signal processing, data manipulation, support for large binary data, support for human processes, internationalization, prototyping, and various utilities.

4.5 SDK

The *LS/ABPM Standard Development Kit (SDK)* allows the implementation of LS/ABPM-based business process applications. In particular, the SDK enables the creation of:

- Application-specific GO-BPMN tasks and functions.
- Application-specific front-ends (either using a Web or other GUI technologies).
- Interface 3rd-party software components with the LS/ABPM engine.

Once the development environment is set up (see Figure 9), using the LS/ABPM Java API a developer is enabled to implement application-specific components. The created artifacts can be directly tested and executed within the development environment. The LS/ABPM SDK also provides Maven and ANT scripts able to build and deploy the LS/ABPM-based application into industry-grade J2EE runtime environments.

The LS/ABPM SDK includes a *Default Web Application,* that is, a simplified and domain-independent Web front-end usable when prototyping an LS/ABPM-based application. It allows a process participant to authenticate into the system, to create new process instances and to perform human-executable tasks. All the requested tasks are visible in the user's ToDo list. In general, two types of activities can be requested:

- *Input request*: the user is requested to provide input to a process instance.
- *Output notification*: the user is notified that some event occurred.

The Default Web Application, built using the Java Server Faces technology, supports a fully automatic, type-driven layout for input requests. This means that if the process model uses the human tasks provided in the Standard Library, the Web input forms are dynamically generated based on the types of the context variables bound with the task parameters.

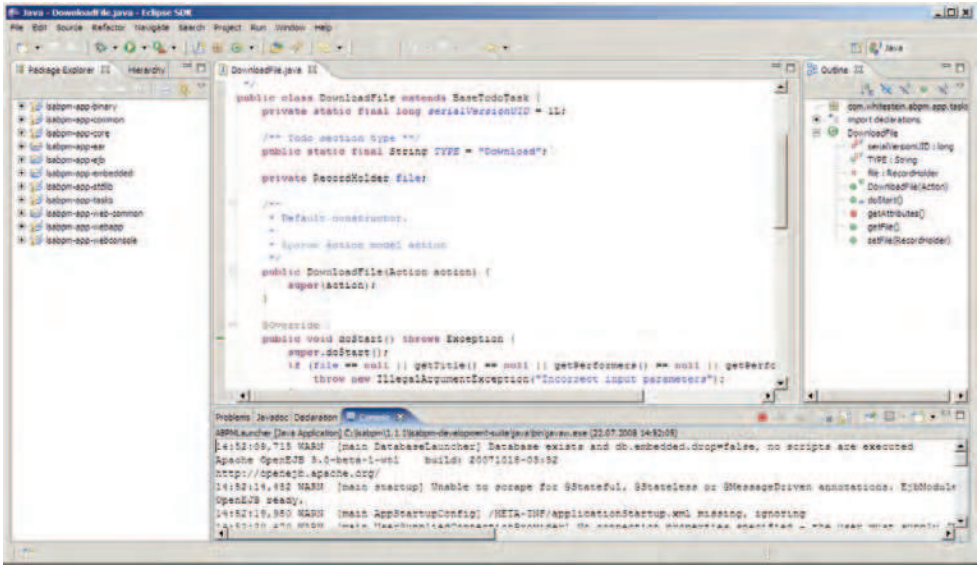

Fig. 9. The LS/ABPM SDK within the Eclipse environment

5. Engineering change management for daimler AG

In 2006 DaimlerChrysler, and now solely Daimler, took the strategic decision to radically update its approach to Engineering Change Management (ECM); the collection of integrated processes for handling the lifecycles of its entire products and parts range. The Strategic Automotive Product Data Standards Industry Group (SASIG) publishes a recommended ECM reference process (SASIG, 2008) with which many manufacturing enterprises comply, including Daimler AG. The reference process for ECM published by SASIG is illustrated in Figure 10.

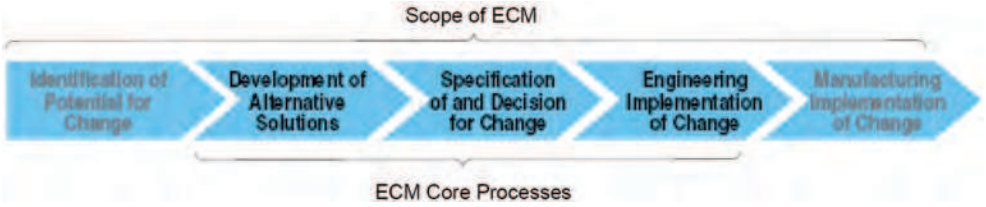

Fig. 10. SASIG ECM Reference Process

Within the context of ECM, Daimler had found that their existing BPM solution could not cope with their increasingly pressing demands for flexibility and robustness in the processes governing the lifecycles of their engineering assets. Product change and evolution is a dynamic activity by definition. Changes can be provoked by a host of reasons, most of which be neither easily be predicted nor controlled, e.g., market trends, partner integration, competition pressure, and normative regulations. Every change must be assessed and applied in consideration of critical factors such as quality, time-to-market and cost. Two of

the fundamental limitations Daimler encountered with conventional approaches to BPM were:

Rigid system design processes - the ECM process has become increasingly significant to the management of change events calling for more flexible process development. The existing approach offers only rigid and costly system design processes, not allowing managers to rapidly and optimally adapt the processes to changing priorities.

Rigid process execution - the current system's process control is not capable of governing change requests in a situation-specific and purposeful manner, that is, adapted to process content and the project's context. For example, both minor and drastic changes follow the same process steps. This places excessive strain on the organisation and slows down the overall process.

A thorough examination of options to re-engineer aspects of Daimler's current BPM approach demonstrated that these limitations could be only partially mitigated, and only in the short term. Moreover, an evaluation of conventional BPM systems available from major industry vendors demonstrated that they uniformly lacked the goal-orientation dimension in their process models, a key requirement from the perspective of capturing business-level goals at model level.

It was also apparent that few were capable of real-time adaption of process paths in response to changing influences and goals. It was this assessment that led to Daimler's commitment to seek a novel approach to BPM that met their requirements for process agility and goal-orientation. The nascent LS/ABPM suite from Whitestein Technologies was selected as the result of two global evaluation phases due to its intrinsic support for the key requirements of Daimler. In March 2008 version 1.0 of the suite was released to Daimler, who are now employing it to manage ECM processes, beginning with the critical Specification and Decision for Change (see Figure 10) otherwise known as the Engineering Change Request (ECR). A highly simplified process model typical of that governing the ECR is illustrated in Figure 11, modelled using GO-BPMN and the Process Modeler.

The top-goal relates to the overall management of an ECR with the next level of child goals dealing with the various business objectives comprising the ECR. Each of these sub-goals can be active concurrently and must all be completed (unless deactivated) in order for top-goal to be achieved. The 'plus' symbol indicted on some of these achieve goal icons indicates that further sub-goals are present, with an additional level unfolded for the `ECR_Analyzed` and `ECR_Decided` goals. The hierarchy can be as deep as required, although typically no more than three or four layers of granularity are needed.

Attached to each leaf-goal in the model are the plans containing the particular BPMN-scripted tasks to be performed when the plan is selected for execution. In the case of the `CostsAssessed` goal, two plans are available for execution yet only one will be selected at runtime according to context conditions present in the process' execution environment (set by humans or software components perhaps activated within an earlier plan). Cost evaluation is a highly relevant assessment criterion in ECM and requires experts to calculate the production cost variation resulting from the change. This calculation can either be exact or estimated, with both procedures offering benefits and tradeoffs according to the situation. Sometimes either will do, sometimes exact costs are mandated, and sometimes estimates are preferred to relieve the complexity in calculating exact figures. The structure of the process lends itself to flexibility and thus plans can be added, replaced, or removed at runtime. For

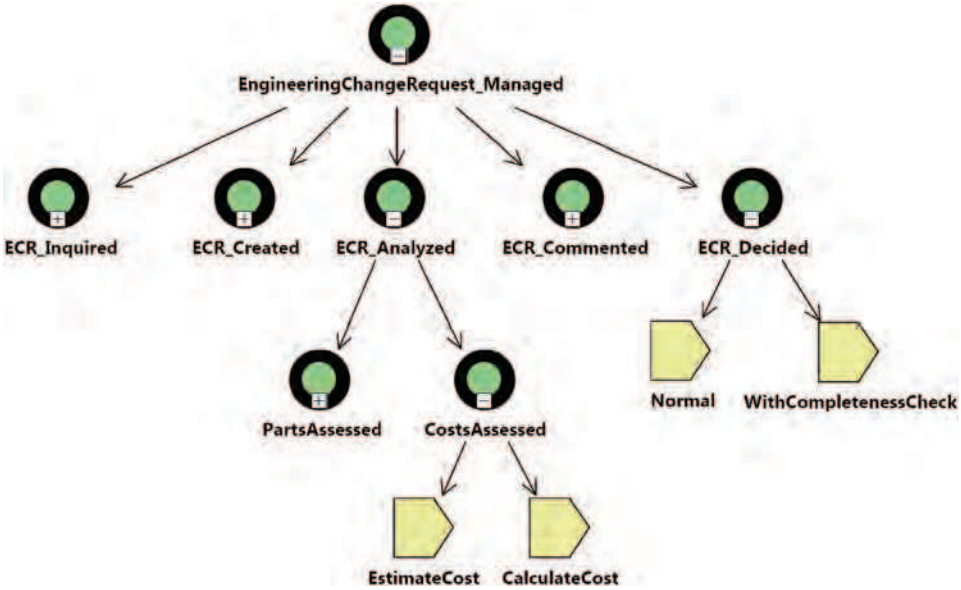

Fig. 11. A simplified model of the ECM Change Request process

example, the `CalculateCost` plan could be replaced with an alternative plan perhaps containing a new cost calculation function. Otherwise a new `CalculateCost_HighPrecision` plan could be added to accompany the two existing plans, offering a high precision calculation which may be preferred if time is available and/or extremely accurate cost assessment is required.

The goal-oriented approach thus allows the expression of a wide and diverse set of solutions while minimising the combinatorial complexity of process definition and execution.

The interface between the process structure and the environment it is affecting are actual plan tasks, typically grouped into general purpose and domain-specific task libraries. A general purpose task may be to generate a Web form, ask a human for input, or activate some domain agnostic software service. Domain-specific tasks are uniquely intended for use in processes used within a particular domain, such as ECM. Of particular interest are tasks that directly invoke SOA services, or service compositions, as is commonly the case with Service Delivery Platforms (SDPs) in the telecommunications domain. At this time we are preparing an adjunct product to LS/ABPM capable of performing goal-oriented dynamic service composition and invocation[3].

6. Discussion and conclusions

This chapter has introduced a novel, industry-proven BPMS employing a goal-oriented approach to modelling and executing agile business processes. We have demonstrated that this is an inherently flexible approach, allowing processes to be designed and executed

[3] LS/ASCO - Living Systems® Autonomic Service Composition and Orchestration.

following the logical ways in which humans naturally comprehend processes. Moreover, the approach uses autonomic self-management allowing processes to self-configure and self-optimize according to strategic requirements and changing operational constraints.
Three of the primary facets of the approach are illustrated in Figure 12: process governance, process optimization and process automation, all three with particular aspects of autonomic behaviour.
Process governance uses goal expressions to capture the purpose of a process in terms of business-level objectives and strategies. In this respect business-level knowledge and intention is expressed directly in the model providing a coherent relationship between the 'why, what, and how' of a process and a clean delineation of domain knowledge and execution logic. We have found through customer reports that goal modelling is an intuitive technique accessible to both Business Managers and Process Analysts alike.
Process optimization allows the structural and parametrical adaption of processes. Manual optimization of a model structure can induce automatic update of the corresponding process instances. Equivalently autonomic optimization of process instances can induce automatic update of process model structures in response to events/conditions arising in operational environments/controlled systems. Persistent monitoring of active models and process instances to ensure that change to in either respect is properly and safely reflected to the other. All optimizations are verifiable and reversible.

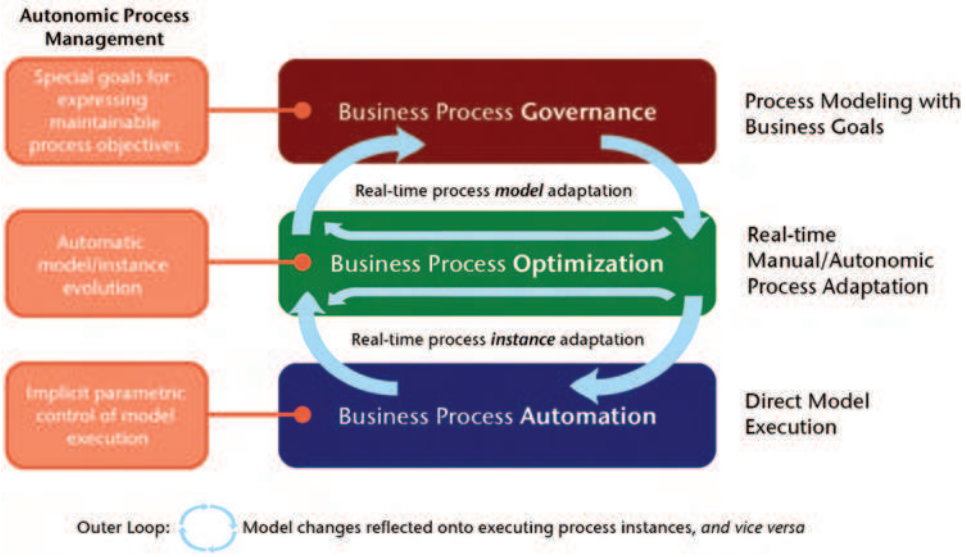

Fig. 12. Business Process Governance, Optimization and Automation

Process automation takes designed models and directly executes them using the process navigation engine. Visual process models are directly rendered into the goal-oriented execution logic of software agent process controllers; one process instance to one software agent. The activities, tasks, and resources required to attain goal objectives are selected or assembled dynamically at run-time. Executing process instances interact through inter-agent

communication. Messages are as simple as required by the process context with advanced semantic communication available for complex interactions.

Autonomic behaviour is manifested at the governance level through the use of maintain goals in process models to express iteratively sustainable process objectives. At the optimization level it is manifested as automatic model/instance evolution through feedback loops propagating change and thereby keeping the entire system in balance. At the automation level, it is manifested as implicit parametric control of model execution.

As a product suite the combination of LS/ABPM and LS/TS offer a unique and compelling approach to BPMS developed with the needs of today's evolving, globally connected enterprise in mind. The suite is domain agnostic by design with the automotive, and in particular the ECM, domain being our first large scale deployment. The case described in the latter part of the chapter is currently in the early stages of full deployment and integration, with the customer, Daimler AG, working in close collaboration to assist with ongoing product refinement and next-stage development.

We are currently in the process of approaching several other domains of application including telecommunications, data center management and financial systems. In particular, the transformation to all-IP network architecture and SOA-driven software architecture is creating a pressing need for innovative approaches to BPM in the telecommunications domain. At the time of publication we have several showcases demonstrating the application of our goal-oriented LS/ABPM technology to telecommunications-specific processes (Whitestein Technologies, 2008), especially those associated with conventional and Next-Generation Networking (NGN) Product Lifecycle Management (PLM) and Order Management.

7. References

Benfield, S. (2006). Beyond BPM: Using goal-seeking agents to tackle highly-complex SOA applications, *SOA World Conference*, New York, U.S.A.

Cardoso, J. (2006). Complexity analysis of BPEL web processes, *Software Process: Improvement and Practice Journal - Special Issue on Design for Flexibility*, 12(1):35-49

Cervenka, R. & Trencansky, I. (2007). *A Comprehensive Approach to Modelling Multi-Agent Systems,* Birkhauser, ISBN 978-3764383954, Basel, Switzerland

Greenwood, D. & Rimassa, G. (2007). Autonomic goal-oriented business process management, *Proceedings of the Third International Conference on Autonomic and Autonomous Systems (ICAS),* 43, Athens, Greece

Habhouba, D., Desrochers, A. & Cherkaoui, S. (2006). Engineering change management and decision-making assistance using software agent. *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 1694-97, Ottawa, Canada

Pautasso, C., Heinis, T. & Alonso, G. (2007). Autonomic resource provisioning for software business processes. *Information Software Technology*, 49(1):65-80

Rao, S. & Georgeff, M.P. (1995). BDI-agents: from theory to practice. *Proceedings of the First Intl. Conference on Multiagent Systems*, 312-319, San Francisco, USA

Rimassa, G., Greenwood, D. & Kernland, M. (2005). The Living Systems Technology Suite: An autonomous middleware for autonomic computing. *Proceedings of the Second*

International Conference on Autonomic and Autonomous Systems (ICAS), 33, Santa Clara, USA

SASIG: Strategic Automotive Product Data Standards Industry Group. (2008). Engineering Change Management (ECM) reference process.

Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O., & White, S.R. (2004). A multi-agent systems approach to autonomic computing. *Proceedings of the 3rd Intl. Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 464-471, New York, USA

Tibco. (2006). Goal-driven business process management: Creating agile business processes for an unpredictable environment. *Tibco Whitepaper*

Whitestein Technologies. (2008). Solution Profile Telecoms: Autonomic Business Process Management for Next Generation Networks. *Whitestein Technologies Whitepaper*

Multiagent Systems

Edited by Salman Ahmed and Mohd Noh Karsiti

ISBN 978-3-902613-51-6
Hard cover, 426 pages
Publisher I-Tech Education and Publishing
Published online 01, January, 2009
Published in print edition January, 2009

Multi agent systems involve a team of agents working together socially to accomplish a task. An agent can be social in many ways. One is when an agent helps others in solving complex problems. The field of multi agent systems investigates the process underlying distributed problem solving and designs some protocols and mechanisms involved in this process. This book presents an overview of some of the research issues in the field of multi agents. It is a presentation of a combination of different research issues which are pursued by researchers in the domain of multi agent systems as they are one of the best ways to understand and model human societies and behaviours. In fact, such systems are the systems of the future.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821