

# Autonomic Goal-Oriented Business Process Management

Dominic Greenwood  
Whitestein Technologies AG  
Pestalozzistrasse 24  
8032 Zurich, Switzerland  
Email: dgr@whitestein.com

Giovanni Rimassa  
Whitestein Technologies AG  
Pestalozzistrasse 24  
8032 Zurich, Switzerland  
Email: gri@whitestein.com

**Abstract**—This paper outlines an autonomic approach to business process management using goal-oriented principles and autonomous agent technology employing the well-known BDI method. We discuss how the structure of common business processes can be mapped onto goal-oriented models where achievement points are represented by goals and the task structures used to reach them are represented by plans. The flexibility afforded by this approach implies that autonomic feedback loops are created between the business process incarnation and its autonomous controller, which may be extended to the multiple process case where several controllers interact to form extended feedback relationships. We introduce a set of technologies designed to deliver these features in real-world applications and support the position with a case study in the domain of Engineering Change Management.

## I. INTRODUCTION

Business processes and business process management (BPM) are an essential constituent of many modern enterprises. They constitute organizational and operational knowledge and often perpetuate independently of personnel and infrastructure change. As such, their design, execution and responsiveness to change is of critical significance to establishing and maintaining efficient business operation.

In addition, 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. However, if the process management system is not built to innately manage change, the result can be further reductions in both dependability and visibility, especially from a management perspective. Our conclusion is therefore that many of the current procedural approaches to BPM are too inflexible and unresponsive to change, especially in any automated, or autonomic, fashion. The alternative approach presented in this paper targets the reduction of complexity through continuous, automatic assessment and change to maintain high levels of reliability during process execution.

Our starting point is 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. In fact for humans it is natural to set goals, decompose goals into sub-goals, and to define or reuse plans to achieve the 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<sup>1</sup>). This results in processes that are efficient in execution, but with limited expressivity and responsiveness to change; they can easily become convoluted and brittle [2].

To maintain effectiveness without sacrificing agility, we propose that the concepts of plan and goal be brought to center stage in BPM solutions. Our approach uses implicit, goal-oriented business process specification, which 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 systems that are intrinsically capable of handling higher levels of complexity and change. Business processes therefore become executable goal-oriented models [1] [10].

An autonomous agent controller is assigned to each business process, responsible for coordinating the process algebra and task structuring within goal-plan combinations, taking into account goal and plan preconditions. Such controllers form an element in a feedback control loop with their active, managed processes providing autonomic self-configuration and self-optimization at the level of individual business process [8]. Moreover, through inter-controller interaction a further level of feedback and control is possible spanning process boundaries to create collaborative process chains and graphs.

Goal-orientation offers the flexibility to dynamically form and restructure processes post-deployment, taking into ac-

<sup>1</sup>BPEL is the Business Process Execution Language

count both transience and the role of humans in processes. Autonomic self-management offers dependability in the face of changing requirements and environments: guaranteed behaviour through self-regulation, resilience to failure and the isolation of disruptions.

To illustrate our approach we describe the concrete application case of *engineering change management* [5] which is a mission-critical business operation often managed through BPM. Change management 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.

Section II of the paper introduces some of the key principles of our approach, before section III identifies our technical approach to realising these principles. Section III then discusses the engineering change management application domain, and section IV concludes the paper.

## II. KEY PRINCIPLES

In an earlier paper [9] we proposed that an autonomic system [4] [6] may be considered as a layered architectural pattern. We identified three common patterns as illustrated in Figure 1:

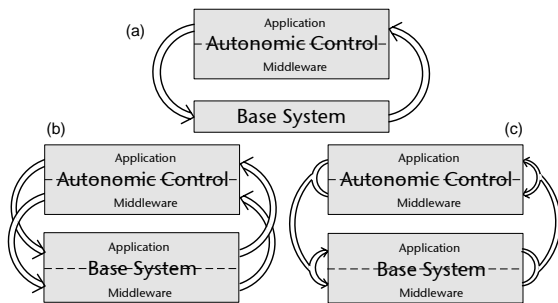


Fig. 1. Autonomic system patterns: (a) autonomic compensation, (b) autonomic partition, (c) autonomic coupling.

The *autonomic compensation pattern* is where an existing base system is coupled to a middleware which acts as a compensator. The feedback loop that takes data from the base system and operates control actions on it only involves the application level of the autonomic controller. Thus the middleware/application layering and the autonomic holism are completely independent.

The *autonomic partition pattern* is where the system can still be considered as the union of a base system and an autonomic controller, but each are partitioned into middleware and application layers. This results a separation of the middleware level into application-independent and application-specific parts. Moreover, control strategies are separated to ensure that control of the middleware-level state uses only middleware-level system state values. The resulting system has two separate feedback loops at middleware and application level.

The *autonomic coupling pattern* is where the system retains an autonomic feedback chain, and both base system and

autonomic controller have an application and a middleware layer; the two are coupled rather than partitioned. The entire base system state, regardless of the layer it belongs to, is fed back to the autonomic controller and both layers of the controller manipulate both the middleware and the application layer of the base system. This pattern is useful whenever a fixed (i.e., open-loop) relationship between the infrastructure-level and application-level parts of the system state cannot be modeled a priori.

In this paper we propose a system model for autonomic BPM that can be overlaid onto each of these patterns. Our autonomous agent system serves as the autonomic control layer segmented into middleware services, described in section III-A and agent application logic for goal-oriented BPM, described in section III-B. The agents are essentially autonomic controllers of business processes acting on base system state.

The particular pattern in use is then largely dependent on the availability of control interfaces at the base-system layer. Simple base-systems may not expose any control interfaces implying that the autonomic compensation pattern is in effect. More complex systems, such as the engineering change management system detailed in section IV typically run over established middleware such as IBM Websphere<sup>2</sup> which provides extensive control interfaces.

Thus, as feedback loops are a central tenet of autonomic computing [6], it is the presence of feedback loops between a base system with deployed business processes and the autonomous agent-based autonomic controllers that drives self-management enabling dynamic configuration of executing business process. To provide the flexibility required for such dynamic process configuration, we employ the goal-oriented approach to process construction. In fact, we assert that a goal-oriented approach to process modeling with autonomous controllers is intrinsically autonomic in nature.

In principle, goal-orientation is based on separating the declarative statements defining desired system behaviour from the various ways to achieve that behaviour. A partial view of an executable GO-BPMN process model<sup>3</sup> is shown in Figure 2.

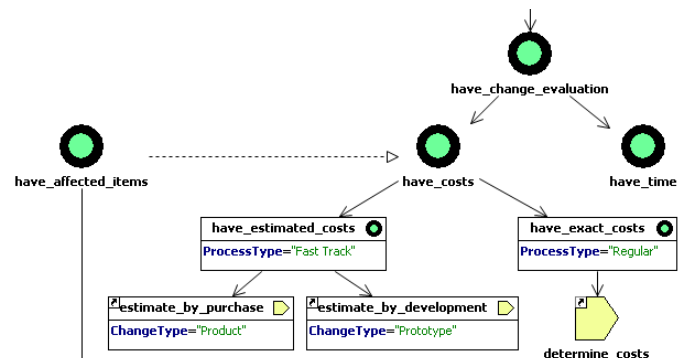


Fig. 2. Partial view of a Goal-Oriented BPMN (GO-BPMN) model

<sup>2</sup>See <http://www-306.ibm.com/software/websphere>

<sup>3</sup>GO-BPMN is our Goal-Oriented extension of traditional BPMN (Business Process Modeling Notation), and a component of our Living Systems Autonomic Business Process Management Suite

In this figure we can observe desired effects described as achievement goals and maintenance invariant goals whose violation must be avoided (the latter are not shown in this example). All goals are annotated with pre-conditions which must be true before the goal becomes active. Once a goal is active it is achieved using plans which are typically a set of tasks to be performed in accordance with metadata conditions such as resource and time consumption constraints. Plans are also annotated with pre-conditions, otherwise known as guards, which must be true before that plan is considered as a valid option to meet a goal's requirements.

Process models such as this are directly executable using our autonomic business process management application<sup>4</sup> (see section III-B). This process execution engine is capable of dynamically fitting plans to unresolved goals allowing strategy encapsulation where a set of plans attached to any given goal represents a collection of different strategies. The details of an individual strategy are enclosed within a single plan.

The execution of goal-oriented models is managed with autonomous controllers, with one responsible for each process. The controller coordinates the process algebra and task structuring within goal-plan combinations and will decide when and how to enact autonomic self-management policies according to situational change. On a per process basis, this control may be effected in the following ways:

*Alteration of Goal Preconditions:* Each goal in a process is assigned one or more preconditions that control its state, such as under what conditions it becomes active. Multiple preconditions may be composed using standard process algebra predicates. At this level the autonomous controller can *self-optimize* a process by assessing whether a goal hierarchy can, for example, be segmented by fragmenting goals into sub-goals to achieve partial results. This is particularly useful when a goal cannot be achieved due to non-satisfiable precondition and where segmenting the goal into sub-goals will allow at least some proportion of the goal condition to be met. Additionally, temporal pre-conditions can be adapted to alter the order of goal succession when possible and appropriate. Thus, the autonomic feedback loop in this instance is sensing when goal preconditions, and indeed goals (whether achievement or maintenance in nature), can be reformulated according to strategic beliefs held by the autonomous controller, and then acting to restructure the process as appropriate. Depending on the particular application domain, processes can be altered during execution with proper attention to state preservation.

*Selection of Plans:* Every goal can be achieved with one or multiple plans and every plan has a guard precondition that when satisfied indicates its suitability to achieve or maintain a goal. Multiple guards may be composed using standard process algebra predicates. At this level the autonomous controller can *self-configure* a process in

accordance with required goal pre-conditions by either (a) discovering appropriate plans from available plan repositories, (b) re-constituting an available plan in terms of the task sequence defined therein, or (c) constructing a new plan. The autonomic feedback loop in this instance is sensing when a goal may be satisfied by alternative plans, determining plans with guard conditions that meet the satisfaction criteria and positioning the plan in the process graph accordingly.

When considering interaction between multiple autonomous process controllers the feedback loop is no longer a straightforward relationship between a single agent and the process it is controlling. Instead the situation becomes more complex, with multiple feedback loops within each autonomically controlled process, potentially affecting one another through inter-process interaction. As such autonomic control is extended to include:

*Inter-process Dependency:* Inter-process relationships can occur via binding between the goals and plans of different processes. This is evident in the case of business processes that span corporate boundaries such as with supply chains and supply graphs. The preconditions of goals and guards of plans can be dependent on states in remote processes that may vary in visibility and availability. In such cases the self-management loop is extended to include inter-controller relationships such as that depicted in Figure 3. Here the three depicted controllers extend the autonomic feedback loop from the processes that they individually control with a secondary inter-controller loop used to coordinate their activities and thus influence one another's process management. For example, with reference to Figure 3, the failure of a task in a plan  $A_{p12}$  may result in controller A automatically coordinating with controller C to reconfigure the process structure of process  $C_{pr}$  to compensate for the failure with the addition of a new goal and plan to circumvent the failure with an alternative approach.

Autonomous controllers are also capable of interpreting ontological semantic annotations linked to goals and plans. These assist with more accurate and meaningful (to human users) expression of the conditions required to complete or maintain a goal, and thereby assist the engine with the programmatic selection and/or derivation of plans to achieve all, or part of the goal.

While some human inspection and control tools will always be valuable, a system that is capable of some degree of self-management can both relieve a human of mundane tasks and provide assistance with complex tasks. Indeed complexity of business processes can be mitigated with autonomic control, especially in terms of recognising and counteracting undesirable situations, resolving confictions and providing guarantees of dependability and resilience when process changes are necessary.

<sup>4</sup>The Living Systems Autonomic Business Process Management system

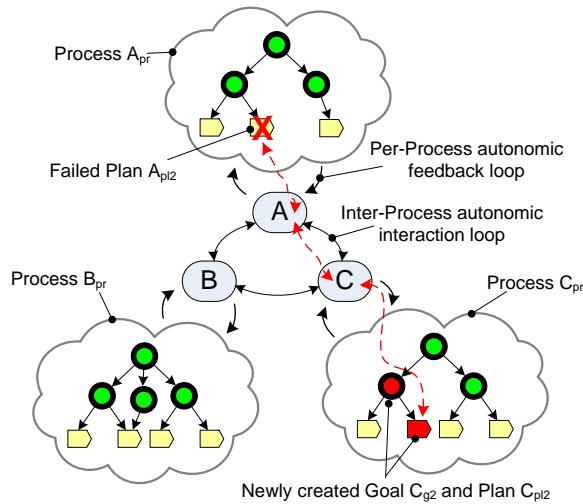


Fig. 3. Example of inter-controller interaction extending the autonomic feedback loop between processes and their controllers.

### III. TECHNICAL REALIZATION

We realize the principles discussed in section II using a suite of technologies that provide the autonomic control layers depicted in Figure 1.

#### A. The Living Systems Technology Suite

The Living Systems Technology Suite (LS/TS for short) [9] is a set of middleware and tools providing a comprehensive environment for the development of multi-agent system. It hosts the agents and services that carry out all the necessary tasks, providing them with support such as life cycle, messaging, persistence, resource management and monitoring.

LS/TS is the result of several years of professional software development and offers the means to not only develop the individual components of an autonomic system (i.e. the autonomic elements) as software agents, but more importantly, to develop entire agent-driven middlewares that embody the holistic approach to engineering autonomic computing. LS/TS provides a comprehensive set of components and tools to aid the development and deployment of professional-grade systems based on software agent technology.

The number and coverage of the features provided by a software development infrastructure such as LS/TS are much more than can be reported here, but among them it is useful to mention and present a few of the ones that turn out particularly valuable to realize autonomic systems.

Multi-agent systems can be positioned as the convergence point of the Service Oriented Architecture and Event Driven Architecture trends. A first important remark is that the multi-agent system approach to software construction includes *activity-centric programming*.

This means that the tasks an agent can perform are represented as first-class objects; they can be assembled together in structured compounds and can be reasoned about without actually executing them. The LS/TS platform adopts a process-algebraic model to drive this task reification and assembly:

tasks become Java objects and their composition follows the grammar and semantics of the operators of a suitable Process Algebra.

Activity-centric programming is a necessary enabling condition of autonomic behavior, because it makes the system immediately aware of its current and possible future execution states, thereby allowing activity control strategies to be established.

LS/TS agents are also equipped with a generic *goal-oriented* execution engine, which is based on the *Belief-Desire-Intention* (BDI) architecture. Using this engine, software agents can be given a dynamically changing set of goals and plans, which they are then able to exploit to behave promptly and effectively in unpredictable situations. Some of the benefits of goal-oriented programming are:

- *Implicit programming*. The user does not need to globally specify the application logic, and the execution engine combines plans together automatically.
- *Increased understandability*. Goals and their relationship already show what the system is supposed to do and how the various pieces of desired behavior work together.
- *Encapsulation of strategies*. The set of plans that are attached to a given goal represents different strategies, whose details are enclosed within the single plan, and do not spark dependency chains across the system.

The above benefits are intrinsic to goal-oriented programming, but it is their interrelationships that conspire to ease the design and implementation of autonomic feedback loops at the single component level. An agent that holds its own goals as internal information, and that keeps a clean separation between means and ends, is immediately ready for the addition of self-management capabilities.

Another feature that becomes extremely relevant for the realization of autonomic systems is the LS/TS *Semantic Communication*. This communication model specifies a structured approach the exchange of semantic messages within multi-agent systems. The first important issue is defining what kind of data is meaningful content for a message and what structure does it have. Moreover, a suitable semantics must be assigned to the constituents of the structure. The entities used within the communication content model can then be organized into *ontologies*, defining content categories and the relationships among them.

Beyond considering a single isolated message, the Semantic Communication model also deals with long-term conversations between several agents. In a multi-agent system using semantic communication, the conversation is the most important unit of interaction, and significant system state evolution typically takes place only when a conversation ends.

These traits become particularly relevant to autonomic systems, when system-wide feedback loops are considered. Such loops stem either from multi-agent system interaction design or emergent behavior, but in any case they are eased by a content-rich, layered messaging model.

## B. Autonomic Business Process Management

The original vision of autonomic computing mainly considers IT systems that, when equipped with sufficient self-management capabilities, can substantially reduce or even eliminate human intervention for administration. Nevertheless, the founding concepts of autonomic computing are not only applicable but also quite useful in many other cases.

In particular, the overall challenge that autonomic computing is tackling is to handle ever increasing system complexity [7]. This complexity is especially evident when the system is not merely composed of hardware and software components, but also encompasses the humans interacting with it, along with their organizational and process issues.

In a business setting, one can move away from the ideas of Business Process Management and consider how the *self-\** qualities of autonomic computing can be beneficial when applied to the whole set of mechanical, human and organizational parts that enact business processes.

The resulting approach can be named *Autonomic BPM*, and it provides for example, self-healing of process activities through backup alternative tactics, or self-optimization by automatically detecting feasible remedies and proposing reasonable options to a human for selection.

It is to be noticed that implementing autonomic BPM involves both a BPM runtime environment and suitable business processes (most likely involving also humans). The consequence is that the notion of self-management in autonomic BPM has two different facets:

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

Effectively managing complex business processes in the face of a dynamic and unpredictable environment requires striking a careful balance between flexibility and safety. The definition and execution of business processes need to be easily changed during operation to adapt to unforeseen changes. It must also be possible to ensure that these changes are correct and do not incur any unfavorable consequences.

The considerations made so far are the motivations for our development of the Living Systems Autonomic BPM platform (*LS/ABPM* for short). Illustrated in Figure 4, *LS/ABPM* provides a modeling language, tools and runtime environment to define, execute and steer business processes. It is realized as a multi-agent system, hosted by *LS/TS*, and as such has intrinsic support for the engineering of autonomic systems.

The major traits added by *LS/ABPM* are:

- A Goal-oriented, executable modeling language for business processes. The *GO-BPMN* language employs im-

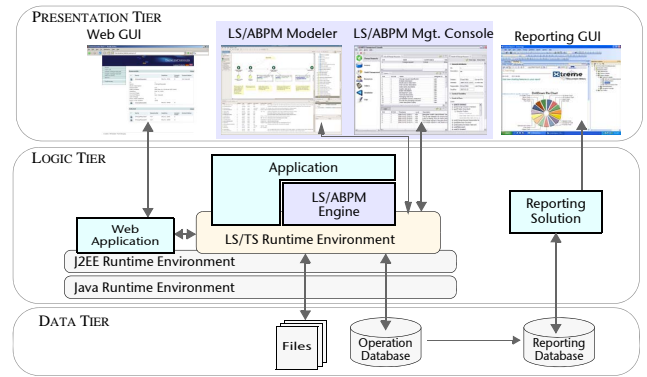


Fig. 4. LS/ABPM Architecture

plicit, goal-oriented business process specification that offers clean separation between the goals to be achieved (or maintained) and the set of plans (action graphs) used to achieve or maintain them. This results in the deployment of systems that are intrinsically capable of handling higher levels of complexity and change.

- More readily available introspection. The various elements of a *GO-BPMN* model are easily enriched with metadata, that can be readily used to control and tune the process model execution. With this feature, it becomes natural and effective to introduce suitable control processes reasoning and acting on the basis of policies of cost, time and resources.
- Layered programmability. Different parts of the system behavior can be expressed at different levels (at the business process level or at the underlying multi-agent system level). This allows to tackle each issue at the most suitable abstraction layer, using the most natural concepts; moreover, most of the changes will be dealt with at the upper, safer business process level.

## IV. CASE STUDY: ENGINEERING CHANGE MANAGEMENT

Change management relates to the evolution of a product throughout its lifetime. Changes can be provoked for many reasons including market trends, competitive pressure and normative regulations. Every change must be assessed and applied in consideration of critical factors such as quality, time-to-market and cost. As such product evolution is dynamic by definition; change management is therefore a good example of where goal-oriented autonomic BPM can enable flexible and dependable process management.

Moreover, in today's modern and global scale settings, managing the evolution of a product or a product line needs also to cope with issues such as distributed supply chain management, flexible partner integration and timely adaptation to changes in norms and standards.

The case study where we are applying the solution pair (*LS/TS* and *LS/ABPM*) previously described is a particular instance of change management, that is *Engineering Change Management (ECM)* for short). This term is used in manufacturing domains to refer to the management of changes to the

digital description of a produced item (description sometimes named *digital product* itself).

In particular, the automotive domain is to be considered, not only for its need for flexible and innovative product lifecycle management, but also due to the presence of standardization efforts, such as *VDA 4965* [3], a recommendation document published by the *Verband der Automobilindustrie* (VDA for short), and it is therefore specific to automotive industry. Its stated goal is to support ECM processes throughout the value chain of an automotive manufacturing company.

This standard augments the previous efforts, which were mostly concerned with data interchange and integration, with a strong focus on cross-company business process integration, putting the emphasis on multiple interacting business processes, interfacing with each other thanks to a common process model mapping.

More concretely, let us consider a request for an engineering change. Cost evaluation is typically a highly relevant assessment criterion 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. A goal-oriented BPM model can be populated with goals representing the achievement points in this assessment. This might imply a single achievement goal, or several sub-goals each dealing the different options. Plans can then be automatically situated to enact the alternative processes with constraint annotations defining selection policies. This selection of goals and plans can be made either manually or through self-configuration interaction between the business process and process controller, as described in section II.

The goal-plan separation of goal-oriented BPM thus supports the flexible application of several strategies where the constraints defining a strategy are expressed at the goal level. Selection of plans can be made to pursue the strategy according to situation constraints. The goal-oriented approach allows the expression of a wide and diverse set of solutions while minimising the combinatorial complexity of process definition and execution.

Another example is the use of plan metadata to support process self-optimization. A change management process can have fast-track or lightweight paths that are less thorough than the standard choice to reduce completion time or resource usage. If some unplanned event delays process execution, by combining plan metadata with plan and goal structural information, a goal-oriented BPM system can track the expected completion time and proactively determine if a deadline cannot be met. In reaction, the system can autonomically re-plan to accommodate options such as:

- *Minimise change*: Leave the process as it is and shift the deadline.
- *Minimise completion time*: Execute the process on a fast-track path.

- *Minimise cost*: Execute the process on the lightweight, low-consumption path.

## V. CONCLUSIONS

As the principles of service orientation begin to find widespread industrial use, it is now increasingly common to construct applications as composites of simple, and sometimes complex, building blocks. This growing presence in business and industrial systems is an indicator of a general shift in perspective from relatively rigid and procedural systems toward flexible composite systems that can easily adapt and be adapted over time.

It is with this in mind that we have proposed our approach to business process management that uses a goal-oriented approach to structuring processes. This is an inherently flexible approach, allowing processes to be designed and executed following the logical ways in which humans naturally comprehend processes. Moreover, this approach is readily suited to self-management as each process can be assigned an autonomous controller responsible for all aspects of the process lifecycle. This controller takes on the autonomic manager role in a feedback loop with the business process application, allowing the process to essentially self-configure and self-optimize according to strategic requirements and changing operational constraints.

The paper has introduced our particular technical solutions to both the autonomous middleware and autonomic business process management components of the identified principles. We have also used the concrete domain of Engineering Change Management to illustrate the fact that there is sufficient complexity within many business process application domains to consider a more flexible and self-sustaining approach beyond the typical procedural methods commonly employed today.

## REFERENCES

- [1] S. Benfield. Beyond bpm: Using goal-seeking agents to tackle highly-complex soa applications. In *SOA World Conference*, New York, U.S.A., 2006.
- [2] J. Cardoso. Complexity analysis of bpel web processes. *Software Process: Improvement and Practice Journal. Special Issue on Design for Flexibility.*, 12(1):35–49, 2006.
- [3] V. der Automobilindustrie. *Vda recommendation 4965: Engineering change management (ecm). VDA Recommendation*, 2005.
- [4] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Syst. J.*, 42(1):5–18, 2003.
- [5] D. Habhouba, A. Desrochers, and S. Cherkaoui. Engineering change management and decision-making assistance using software agent. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 1694–97, 2006.
- [6] B. Miller. The autonomic computing edge: The path to level 5, full autonomic maturity. *IBM Whitepaper*, 2005.
- [7] T. Nowicki, M. S. Squillante, and C. W. Wu. Fundamentals of dynamic decentralized optimization in autonomic computing systems. In *Self-star Properties in Complex Information Systems*, pages 204–218, 2005.
- [8] C. Pautasso, T. Heinis, and G. Alonso. Autonomic resource provisioning for software business processes. *Inf. Softw. Technol.*, 49(1):65–80, 2007.
- [9] G. Rimassa, D. Greenwood, and M. Kernland. The living systems technology suite: An autonomous middleware for autonomic computing. In *Proceedings of the Second International Conference on Autonomic and Autonomous Systems (ICAS05)*, 2005.
- [10] Tibco. Goal-driven business process management: Creating agile business processes for an unpredictable environment. *Tibco Whitepaper*, 2006.