BPSM: An Adaptive Platform for Managing Business Process Solutions

Jun-Jang Jeng, Steve Buckley, Henry Chang, Jen-Yao Chung, Shubir Kapoor,
John Kearney, Haifei Li and Josef Schiefer
IBM T.J. Watson Research Center

ABSTRACT

The growing number of business process solutions demands the need of platforms enabling developers to build management applications in a more efficient manner. This paper describes an adaptive platform, called BPSM (Business Process Solution Management), for managing business process solutions. A business process solution touches many enterprise entities: multiple business processes, different organizations, various execution platforms, multiple trading partners, and constantly changing business context. To build platforms for business process solutions is difficult. To develop platforms for managing business process solutions poses an even greater challenge. The BPSM platform is our response to this challenge by creating an adaptive environment so that developers can leverage it to build management applications in the domain of business process solution management. The framework of BPSM will be presented in this paper, covering three kinds of decomposition for business process solution management: horizontal decomposition, vertical decomposition and grid decomposition. A language for management policy called Management Commitment Language (MCL) will be presented. This paper will also account why BPSM is a commitment-governed management (CGM) platform. The architectural framework of BPSM will be described in the form of Q&A style and architectural patterns. Finally, a BPSM-based Supply Chain Management system is presented for illustrating the strength of this platform.

Keywords
Business Process, Solution Management, Architecture, Framework.

1 Introduction

Managing business process management applications that are executed in the Business Process Management (BPM) platforms creates special challenges to an IT organization, and when the applications are critical to business operations and used by almost every role involved the business process, the focus on management issues such as availability, performance and security for that solution grows rapidly. This requires the BPM platforms to become more proactive and manage the expectations to the provided management services as well as take the appropriate measures to actively monitor and control the behavior of business processes and critical resources. The gaining popularity of automated business process solution on BPM platforms has brought on new demands for how BPM platform administrators manage and maintain the components residing in the platform. Managing business process solutions requires knowledge of both business domains and the platforms where they are being executed. The increasing system complexity of BPM infrastructure is reaching a level beyond human ability to manage and secure. This increasing complexity with a shortage of skilled I/T technical staff points towards an inevitable need to *automate* many of the managerial functions associated with BPM platforms today. The growing number of business process solutions demands the need of platforms enabling developers to build management applications in a more efficient manner.

This paper describes an adaptive platform, called BPSM, for managing business process solutions. A business process solution touches many enterprise entities: multiple business processes, different organizations, various execution platforms, multiple partners and constantly changing business context. To build platforms for business process solutions is difficult. To develop platforms for managing business process solutions poses an even greater challenge. The BPSM platform is our response to this challenge by creating an adaptive environment so that developers can leverage it to build applications of managing business process solutions.

A conceptual framework for BPSM will be presented in this paper, covering three categories of decomposition for the domain of managing business process solutions: horizontal decomposition, vertical decomposition and grid decomposition. A language called Management Commitment Language (MCL) is used to specify the management commitments which are to be enforced by BPSM. This paper also accounts why BPSM is a commitment-governed management (CGM) platform. The architectural framework of BPSM will be described in the form of architectural patterns. Finally, a BPSM-based Supply Chain Management system is presented for illustrating the usage of this platform.

This paper is organized as follows. Section 1 introduces the targeted area of this paper. Section 2 presents the conceptual framework of BPSM. Section 3 describes the approaches of both commitment-governed management and management commitment language. Section 4 presents BPSM architecture using the description of used architectural patterns. Section 5 presents our experiences and discussion. Section 6 shows the related work of BPSM. Finally, Section 7 concludes this paper.

2 The Foundation

What is BPSM as a discipline?

A *process* is a specific ordering of activities across time and space, with commencement, a termination, and clearly defined inputs and outputs: an organization for actions. A *business process* refers to a process in which work is organized, coordinated, and focused to produce a valuable product or service. Business processes comprise both internal and external business entities and drive their collaboration to accomplish shared business goals by enabling highly fluid process networks [1].

On one hand, business processes are concrete work flows of material, knowledge and knowledge – set of activities. On the other hand business processes also refer to the unique ways in which organization coordinate work, information, and knowledge, and in ways in which management chooses to coordinate work. Therefore, business process execution is involved with many enterprise entities such as organizations, systems, and applications. The totality of all enterprise entities is called a *solution*. In other words, a solution is a combination of business processes, systems, applications, and organizations in support of overall business goals.

Formally, a *solution space* consists of four dimensions: business process, system, application and organization. The discipline of *solution management* is the set of knowledge, methodologies, and tools that can be used to manage business solutions. Solution management subsumes the traditional senses of system management, application management, business process management, and work flow management. Solution management can be elaborated in different ways depending on which dimension is the centric view of the management activities. Fixing the value of one dimension creates a subspace consisting of the other three dimensions. For example, the business process of supply-chain management (SCM) represents a value along the dimension of business process that implicates a subspace surrounding this value on the dimension of business process. This subspace may comprise the relevant organizations (suppliers, inventory, and consumers), enabling applications (order processing application, optimization algorithms, and middleware) and underlying systems (database, network, and storage). *Business process solution management* refers the domain of solution management taking the business process centric views, in which business processes are treated as the first class citizens and the values of other dimensions are derivatives of the concerned business processes. We can safely say that business process solution management is the meta-level management of business processes because not only is business process solution management is involved with business processes but also with the context in which the business processes are situated.

What is BPSM as a platform?

BPSM can be categorized as a system that is continually interacting with its environment, and that is capable of autonomous actions in this environment in order to meet its management goals. As shown in Figure 1, the BPSM platform takes inputs from the environment, and produces actions that affect it. As such, BPSM interact directly or indirectly with the situated entities in the environment. Examples of situated entities include business context, service-level agreements, and managed resources.

Business context contains information that characterizes the runtime behavior of the managed environment, and is referred to the information related to the managed business processes, for example, the status of business process execution, participating parties, the degree of customer satisfaction, corporate management strategy, and so on. *Service-level agreements* are the management contracts established between BPSM and participating parties. A trivial example is that the maximum turn-around time of any purchase order shall not be greater than 48 hours.

Managed resources comprise of manageable entities that are situated in the environment. Resource's manageability defines information that is useful for managing a resource and details the aspects of the resource including the instrumentation which allows BPSM to interact with it. There have been many standards of defining manageability at various levels, e.g., SNMP [2], CIM [3] and M12 [4]. In this paper, the instrumentation of managed resources is assumed to be in place and will not be further discussed. Through instrumentation, a resource is turned into *managed* resource because its state can be perceived, aggregated, analyzed and modified through the standard interfaces provided by the instrumentation layer that is located between BPSM and its environment.

Horizontal Decomposition

The functionality of BPSM can be decomposed either horizontally or vertically. Figure 1 shows the horizontal decomposition of BPSM that is decomposed into three pillars: *perception*, *evaluation*, and *actuation*. Perception

pillar receives the state information from the environment. Evaluation pillar processes the perceived information. Actuation pillar renders management directives to the managed resources.

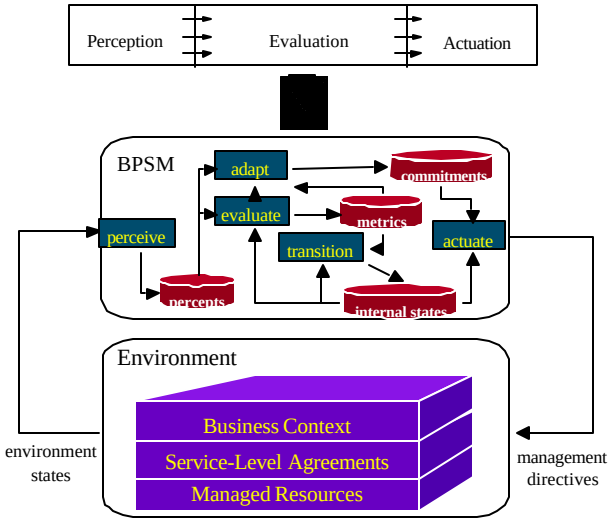

Figure 1 Horizontal Decomposition.

In Figure 1, the Evaluation pillar is further decomposed into three sub-functions: *evaluate*, *transition*, and *adapt*; and four local data stores: *percepts*, *metrics*, *internal states*, and *commitments*. The percepts store contains the perceived values from the environment. The metrics store contains the result of evaluating existing managed resources. The internal states represent the situation of BPSM. Internal states are different from the states perceived from the environment. In general, internal states capture the status of the BPSM platform as a whole, and the environment states capture the status of the environment. The commitments store contains the management commitments that the BPSM platform promises to the participating parties of managed business process solutions.

The state of an environment can be characterized as a set of environment states $S=\{s_1, s_2, \ldots\}$. At a given moment, the environment is assumed to be at one of these states. The managerial capability of BPSM is represented by a set of actions $A=\{a_1,a_2,\ldots\}$. Abstractly, the managerial capability of BPSM can be modeled as a *manage* function as follows:

$$manage: S^* \rightarrow A$$

that maps a sequence of states into actions.

Very likely, BPSM does not have a full control of its environment. The best it will have may be partial control, in that it can influence the environment.

Management commitments represent the promised managerial capabilities that BPSM will act upon its environment. Management commitments are derived from the service level agreements between participating parties and the BPSM owner. The problems of how service-level agreements can be created and how a management commitment can be derived from them are beyond the scope of this paper. One of the premises in BPSM is that the readiness of management commitments precedes the commencement of any management activities.

The behavior of an environment that BPSM interacts with can be also modeled as a function like

$$env: S \times A \rightarrow 2^S$$

that takes the current environment state s? S and an action *a* in *A*, and mapped them to a set of environment states *env(s,a)*, which results from the consequence of performing the action "a" unto the environment that is in state s. If all the sets in the range of "env" contains a single state then the environment is *deterministic*, otherwise it is *non-deterministic*. The BPSM platform is aimed for managing complex and dynamical coordination and the interaction among business processes solutions. The state of its environment can be altered by unmanageable or even unknown sources. In this case, managerial actions may cause en environment state move into different states.

Conceptually, the managerial functions of BPSM can be decomposed into five sub-functions: *perceive*, *evaluate*, *transition*, *actuate*, and *adapt*. The idea is that the *perceive* function captures the BPSM's capability to monitor its environment, the *evaluate* function signifies how the value of the environment can be computed at certain designated moment, the *actuate* function realizes BPSM's decision making capabilities for actions, and the *transition* function changes the internal states.

The *perceive* function can be optionally modeled as a software agent capturing events emitted from the situated entities in the environment. The output of a perceive function is defined as a percept, i.e., a perceptual input. Let P be a set of percepts. Then, perceive is a function as:

$$perceive: S \rightarrow P$$

that maps an environment state to a percept.

Let M represents a set of metrics that are used to assess managed resources during the execution of a business process or after its completion. A metric can be either qualitative or quantitative depending upon the characteristics of the managed resources. Let I be the set of all possible internal states of BPSM.

The *evaluate* function can be modeled as:

$$evaluate: I \times P^* \rightarrow M$$

that maps an internal state and a set of percepts to a metric defined in M.

An internal state represents the information contained in BPSM along the timeline. Examples of state-wise data include the status of business process, management commitments, monitoring policies, actuation policies, logging policies, system configuration, and so on. The *transition* function maps an internal state and a set of metrics into another internal state:

```
transition: I x M* -> I
```

Let C be the set of management commitments that BPSM uses as the management policies to select and deliver managerial actions. More about management commitments will be discussed in following sections. The actuate function can be defined as follows.

```
actuate: I x C* -> A
```

that maps an internal state and a set of commitments into an action defined in A.

Finally, the *adapt* function changes the behavior of BPSM via modifying management commitments. With the adapt function, BPSM can achieve the capability of meta-management which guides the high-order sphere of control upon managed resources. The adapt function can be modeled as a mapping as follows.

```
adapt: I x P x M* -> C
```

that maps an internal state, a percept, and a set of commitments into a commitment in C.

Vertical Decomposition

The above managerial functions lay the foundation of BPSM's managerial capability and enable modelers explore the solution space when BPSM-based platforms are being built. Suppose that we have two environments with the same state $s? S$ but they are perceived in different ways and consequently different percepts are generated. Then, different actions will be taken even the situated environments look very much the same. Horizontal decomposition decomposes BPSM into several functions. Here, we describe another way of decomposing BPSM: vertical decomposition.

We identify three kinds of management mechanisms existing in the domain of BPSM: *Reactive* Management, *Deliberate* Management and *Reflective* Management. Figure 2 illustrates the vertical decomposition, where solid lines represent the flows of management directives, and the dotted lines represent the flows of management events.

Reactive management mechanism responds to the management events quickly and directly through hard-wired event control mechanism. Examples of such mechanism are logging and tracing, which do not require much reasoning and may be default managerial tasks. Another example is the alarm system that will notify the system administrator if some managed system is suffering from severe performance problems, which demands immediate attention.

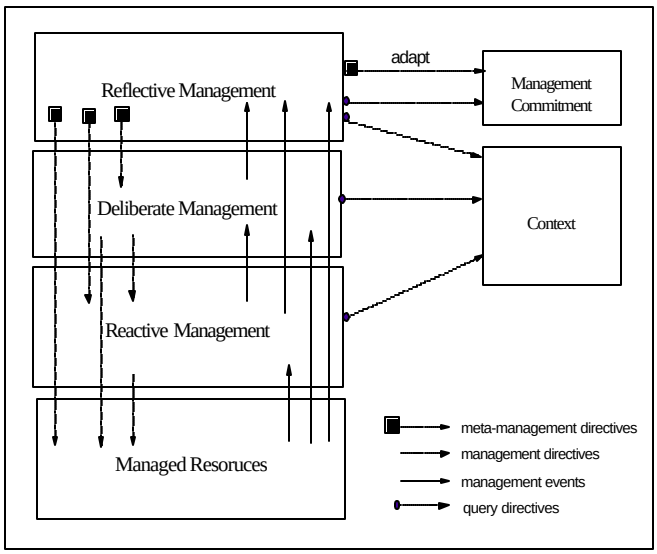

Figure 2 Vertical Decomposition.

Deliberate management mechanism performs managerial tasks that require more reasoning and computational complexity. It is not uncommon that BPSM needs to provide decision support capability so more intelligent management directives can be derived towards managed resources. An example of such managerial tasks is the measurement and analysis of business processes [5]. Another example of deliberate management is transforming events from IT-level into business process level and vice versa. An event such as "disk failure" may mean little out of business context. However, it may imply a loss of gigantic capital for an organization if it has causal relationship with critical business processes such as financial trading. The transformational rules in the layer of deliberate management should capture this relationship so some warning messages can be delivered to appropriate personnel to fix problems.

Reflective management mechanism enables BPSM to maintain information about itself and use this information to remain modifiable and extensible [6]. Reflexive management layer performs meta-management directives unto the lower management layers and managed resources. A meta-management directive is a higher sphere of control such as adapting business process management commitments, modifying measurement and analysis algorithms in the deliberative management layer, or changing the alarm rules in the reactive management layers. As such, BPSM can have detailed knowledge of the managed resources, current status of managed business processes, the ultimate capacity in the inventory, performance expectation, and all connections to other systems in order to manage itself. Therefore, through reflective management mechanism,

BPSM achieve the goals of both 2nd order management and autonomic computing [6].

Grid Decomposition

The above two decompositions can be combined to form *grid decomposition* as shown in Figure 3. Grid model has long been used as a formal modeling tool for cognitive architecture [7]. We found that Grid model is very suitable to illustrate different architectural aspects of management spaces for BPSM. Figure 3 defines nine regions, ($S_{i,j}$ 1= i,j = 3), called *management spaces* and their potential interactions. Only legitimate flows, both management directives and events, are allowed to be transmitted between management spaces. The environment and managed resources emit management events and receive management directives. Evaluation is a completely internal processing with no interaction with the environment and managed resources. The meta-management directives are rendered only through the actuation pillar. Within BPSM, management events can be generated or transformed between layers upwardly and management directives can be rendered downwardly.

Figure 3 shows two typical management scenarios based upon the decomposed management spaces:

- Flow A is a common management scenario: management events are perceived by the reactive management layer, the evaluation is performed by the deliberate management layer, and the actuation is activated through the reactive management layer. This management scenario is called O model of management flow, which is the most common one.
- Flow B is a meta-management scenario where management events are delivered all the way to the reflective layer, evaluated, eventually some meta-management directives are delivered through the actuation pillar in the reflective management layer.

We refer to members of a management space as *Management Agents*, by which we mean autonomous actors interacting with each other, and with the environment. Such a management agent can be a software component, with its own state and thread of control, or it might be a proxy for human users interacting with BPSM with some interface. Grid decomposition actually defines a couple of management agent classes, each of which embodies certain rules of engagement that its situated management agents must abide. For example, the management agents in the management space $S_{2,3}$ can only receive management events from management space $S_{2,2}$ and must comply the management commitments imposed by the management agents in the management space $S_{1,3}$. Consequently, the BPSM conceptual framework entails regulated management agents, management events, managerial functions, and management directives. The following section will address how the rules of management can be defined and processed within BPSM.

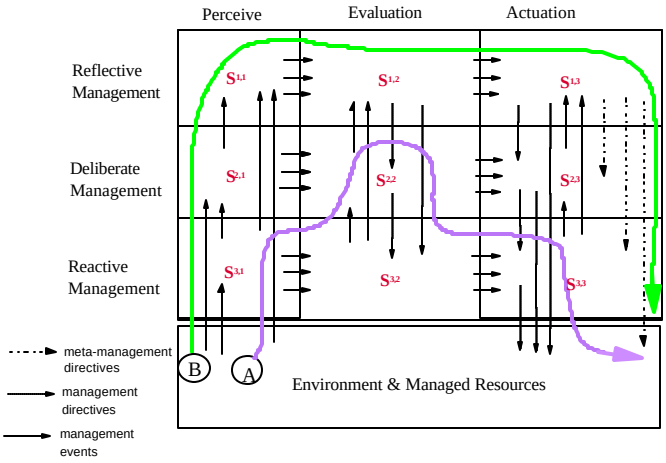

Figure 3 Grid Decomposition.

3 Commitment-Governed Management

The BPSM platform characterizes two perspectives of solution management: policy and mechanism. Management commitments specify the aspect of management policies. A management commitment commits the behavior of BPSM to both the management contracts and the service level agreements. Experiences manifest that service level agreements alone do not provide the full spectrum of the useful management policies that is required in BPSM. In general, a service level agreement merely defines the *external* management contract between BPSM and participating parties, and provides limited information of the *internal* management contract complied by management agents and can be used to characterize the internal managerial behaviors.

In general, a management commitment is a synthesized form of internal and external management contracts. Management commitments are enforced by any management agent during its lifecycle. The managerial behavior of BPSM is governed by management commitments. Hence, the management mechanism of BPSM is generally named as Commitment-Governed Management (CGM). The value of a BPSM implementation really depends on how well the managerial behavior of BPSM can be governed such that it can comply with bound management commitments.

The change of business context usually implies the change of related management commitments. Hence, in a dynamic business environment, static management commitments are insufficient. Management commitments are designed to have run time representation and can be dynamically updated, of which both features enable BPSM to create, manipulate, and propagate the information of management commitments. The need of such features can be understood in the context of Supply-Chain Management (SCM). In

general, management agents can be used to satisfy the requirements of an intelligent SCM system:

- From a *buy-and-supply* perspective, the functions for optimization analysis enable better forecasts and flexible risk management, therefore greatly reducing the risks. Improved partner management and risk sharing further simplify buy-and-supply activities, and allow companies to focus on collaboration with their partners to gain support with minimal risks.
- From a *make* perspective, the functions for flexibility and better forecasts enable dynamic risk management and partner value analysis, thus increasing flexibility and decreasing response times.
- From a *sell-and-support* perspective, the functions for planning analysis enable enterprises to optimally plan product offerings and sales, optimize channels strategies, as well as making quick, informed decisions about changing market demand. Dynamic pricing capabilities enable companies to flexibly set the price to optimize profits.
- Finally, the functions for intelligent resource planning support the allocation of resources to different enterprise activities, considering the risk and return of the activity and its strategic importance to the business.

The aforementioned functions must be adaptive to changing SCM environment in order to fulfill the above requirements. To support the above functions, the management agents at the IT level are created as follows.

- Perception management agents that sense, monitor, and organize critical business information from the environment (cp. Figure 1);
- Analysis management agents that analyze perceived information and conduct certain evaluation tasks;
- Filter management agents that filter perceived enterprise data to enable management directives to disturbances, which are events, in the environment;
- Response management agents that analyze global value chain relationships and information, and derive the optimal strategy for the best SCM performance;
- Predictive management agents that will provide predictive modeling capabilities;
- Controller management agents that coordinate management agents; and
- Learning management agents that are capable of learning by comparing previously predicted trends with recorded data and information in order to improve future response performance.

Many questions can be raised about how management agents can be coordinated and managed so the business commitments of such intelligent SCM system can be fulfilled. For example,

- What event sources should a perception management agent sense or monitor?
- What analytical rules should be used by the analysis management agents?
- What are the filtering policies?
- What are the policies of data aggregation for response management agent?
- Which predictive models should be used?
- What is the coordination mechanism for controller Management Agents? Is it policy-based?
- For learning management agents, where to get historical data and what learning algorithms should be used? What are the measurement criteria for SCM business processes?

The above questions can be much more complicated if their answers are context-sensitive and time-dependent. The BPSM platform handles the complexity via CGM mechanism.

Regarding CGM and management commitments, the following design principles are recognized:

- Management commitments should be explicitly defined by Management Commitment Language (MCL).
- Management commitments must be enforced by CGM.
- Management commitments must also specify the management policies (monitoring, evaluation, prediction, data aggregation and actuation).
- Management commitments specify the management contracts (service level agreements) and policies of BPSM – both internal and external.
- Management commitments should define management commitments at different levels of abstractions.
- Management commitments and CGM should be developed independently.
- Management commitments can be deployed and enforced incrementally.

Detailed syntax and semantics of MCL can be referred to [11]. The major abstraction mechanism of MCL is as follows:

MC 1. *dimension* defines the dimensions that can be used to characterize a particular management commitment perspective. Each dimension contains a set of metrics that can be monitored and measured during process execution. A metric has a domain of values that may be ordered. There are three kinds of domains: set domains, enumerated domains, and numeric domains. Business analysts are allowed to define commitment types representing certain perspectives of management commitments such as performance or availability.

MC 2. *context* defines the condition of the business context that will be used to enforce management commitments. In SCM, four primary management processes of Plan, Source, Make, and Deliver can be incorporated into the context to indicate which organization this management commitment is aimed for. Other examples of context include the priority of the current management commitment.

MC 3. *process* defines the type of the business process this management commitment will be used. In Supply Chain Operation Reference Model (SCOR), three basic process types are defined: *planning*, *execution* and *enabled* [12].

MC 4. *perceive* defines a set of business and IT events that are relevant to the evaluation of this management commitment.

MC 5. *evaluate* contains a set of violation predicates. The disjunction of all the predicates represents a specific management commitment that BPSM will observe, obey, and enforce.

MC 6. *actuate* defines the actions that shoule be taken when current management commitment is violated.

Considering MC3 in the SCM domain, *Planning* processes align expected resources to meet expected demand requirements; and *Execution* processes are triggered by planned or actual demand that changes the state of products. They include scheduling and sequencing, transforming materials and services, and moving product. *Enabled* processes prepare, maintain, and manage information or relationships upon which planning and execution processes rely. In MC4 and MC6, we provided another set of binding specification to map business process-level events (actions) to physical events (actions) at the IT level, and vice versa. In MC5, we devised an evaluation mechanism to evaluate the value of management commitment at the triggering point, which can be driven by either events or time. The management agents serving as the evaluators come from the management spaces $S_{2,2}$, and $S_{3,2}$. An example of management commitment is given as follows.

```
type Reliability = dimension {
  DeliveryQuantityPerformance:
type/numeric,unit/month;
  SupplierOnTimeDeliveryPerformance:
type/numeric, unit/sec;

ProductReceivedWithoutDeliveryIssuesOrErrors:
type/numeric;
};
type Flexibility = dimension {
  SourceCycleTime: type/numeric unit/sec;
};
type Cost = dimension {
  InventoryCarryingCost: type/numeric
unit/dollar;
};
type commitment SCMReceiveProductCommitment
      concerns   Reliability,   Flexibility,
Cost;

commitment            ReceiveProductCommitment
instantiates
SCMReceiveProductCommitment {
context: ReceiveProductBusinessProcess;
process: Execution;
perceive: E1 ProductReceivedEvent
perceive: E2 TimerEvent
evaluate: C1
      DeliveryQuantityPerformance < 100;
evaluate: C2
      SupplierOnTimeDeliveryPerformance < 50;
evaluate: C3
ProductReceivedWithoutDeliveryIssuesOrErrors
               < 50;
evaluate: C4 SourceCycleTime>10;
evaluate C5 InventoryCarryingCost > 2000;
actuate: C1 or C2 or C3 -> invoke
ReliabilityAnalyzer;
actuate: C4 -> invoke PerformanceOptimizer;
actuate: C5 -> invoke InventoryOptimizer;
actuate: any -> notify SolutionOwner
actuate: any ->log BusinessEvents;
};
```

The above MCL definitions include three management dimensions: Reliability, Flexibility and Cost. The Reliability dimension defines three kinds of metrics. The first metric represents the delivery performance of the interested products. It also indicates the type of this dimension and the unit of measurement. The second metric indicates the "Supplier On Time Delivery Performance." The 3rd metric indicates the counts of "Product Received Without Delivery Issues Or Errors." The 2nd management dimension Flexibility contains one metric: "Source Cycle Time" with unit of measurement as second. ." The 3rd dimension Cost contains one metric: "Inventory Carrying Cost" with unit of measurement as dollar. The commitment type SCMReceiveProductCommitment includes the dimensions Reliability, Flexibility, and Cost by using the keyword *concerns.*

We finally define a commitment ReceiveProductCommitment that *instantiates* the commitment type SCMReceiveProductCommitment. It is indicated that the *context* where this commitment will be situated is the management process organization called

ReceiveProductBusinessProcess. The process type is Execution. Two events must be monitored: ProductReceivedEvent and TimerEvent. The *commit* phrases C1, C2, C3, C4 and C5 describe the violation predicates that need to be evaluated and enforced by CGM.

The *actuate* phrases indicate what actions need to be taken when some management commitments are violated. For example, if the violation predicate of C4 is true (i.e., source cycle time > 10) then the performance optimizer will be triggered in order to analyze this problem and possibly provide remedy. The last *actuate* phrase indicates that any commitment violation should trigger both logging and sending alarms to the solution owners.

Figure 4 illustrates an instrumentation process based upon three sets of specifications: management commitments, binding specification, and process execution models.

- Sensors detect management events from the environment.
- Evaluators measure the value of managed business process solution based on events and business process context.
- Actuators render management directives to the managed resources situated in the environment according to the management commitments.

Sensors, evaluators, and actuators are attached to the management agent through the instrumentation process that is controlled by the *Configuration Agent*. Instrumentation has both static and dynamic aspects. Static instrumentation determines the organization of management agents, management services (i.e. sensors, evaluators and actuators) and their relationships at the boot-up time of BPSM. Modification made to the system and the entities situated in the environment while BPSM is running causes dynamic instrumentation process.

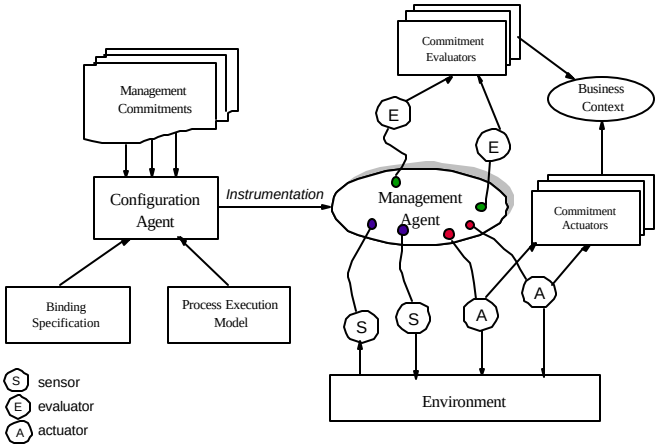

Figure 4 Commitment-Driven Instrumentation.

In general, MCL realizes the concept of the horizontal decomposition described in Section 2. Considering the Deliberate Management layer in the vertical composition, we find that the following mappings are also needed in order to execute management commitments properly:

- Semantic mapping of events between different levels. For example, the monitored event ProductReceivedEvent can be mapped to underlying database events indicating the change of the product status. As the status is changed, a database event will be sent to sensors that subsequently generate the event ProductReceivedEvent accordingly, and notify the commitment coordinator.
- Semantic mapping of actions between different levels of abstractions. For example, the commitment violation C5 will trigger the actuation "invoke InventoryOptimizer," which can be translated into an ordinary method call to an EJB named InventoryOptimizerBean for performing optimization of the inventory by consulting the events, historical data, and business context.
- Semantic mappings of evaluation mechanisms between different levels of abstractions. For example, the evaluation of violation commitment should be mapped to underlying inference engine to perform evaluation.

As mentioned, in BPSM, four perspectives of abstractions are addressed: business process, organization, application and systems. The complete mappings among different levels are beyond the scope of this paper but an architectural pattern for transforming events will be discussed in Section 4.

Since management commitments are decoupled from the execution models of the underlying platform where the managed business process solutions are running. BPSM can be integrated with third-party instrumentation subsystems to fulfill the management commitments as long as both the process execution models and binding specifications are ready for the third-party instrumentation subsystems. More detail about lower-level instrumentation subsystems can be found in [13][14][15]. Referring to the conceptual framework, the processes of dynamic instrumentation, changes of management commitments, changes of execution models and binding specifications are all attributed to the layer of reflective management. To further automate this kind of mata-management activities, a higher level control has been defined and developed in order to enable rational changes of BPSM's behaviors dynamically and adaptively.

4 Architectural Patterns for BPSM

This section describes the architectural patterns of BPSM in the form of Q&A.

Q1. How to organize BPSM management components (agents, managed resources, commitments etc) based upon the principles embedded in the conceptual framework?

A1. The major principle of BPSM conceptual framework is *decomposition*. There are many means to decompose a large system. At the architectural level, we chose to organize management entities in a stack of management layers [6]. Figure 5 shows the layered structure used in BPSM. Management applications are used to perform managerial functions such as business process analysis, system planning, analyzing inventory statistics, report generation, and visualization of system's behavior.

Management services are sharable services by management applications, and are customized managerial functions based upon user's requirements. For example, the function of visualizing business process performance can be realized by a management service dedicated to collecting business process performance statistics and rendering graphics for displaying those data. Management agents carry out managerial tasks on behalf of management applications and services, e.g., monitoring business processes, evaluating their execution, and perform control actions on them.

Meta Management Agents carry out meta-management activities on the behalf of management applications and services, such as adapting management commitments, attaching sensors (evaluator, or actuator) to management agents, detaching sensors (evaluator, or actuator) from management agents, and changing process execution models due to migration of management environment. Managed objects are an abstraction of underlying managed resources situated in the environment.

Managed resource adaptors mediate data conversion and protocol adaptation between managed objects and managed resources. As mentioned in previous section, management commitments prescribe the managerial behavior of BPSM and its associated management entities.

Q2. What is the mechanism of enforcing management commitments in BPSM?

A2. BPSM adopts Event-Condition-Action (ECA) architectural pattern to enforce management commitments. Figure 6 presents the ECA pattern used in BPSM. The architecture of enforcing management commitments is structured according to the Event-Condition-Action paradigm [16][17]. According to MCL and CGM, events are perceived by the sensors, conditions are reasoned and provided by the evaluators, and actions are actuated through actuators. As mentioned, the sensors, evaluators and actuators are attached to the management agents by the Configuration Agent through the instrumentation process.

Sensors obtain raw events from managed resources and convert them into internal event model. On the other hand, evaluators collect raw data and statistics from the business context and pick up necessary metrics from the system; and meaningful conditions are created for the concerned management agent. Actuators receive management directives (actions) from the management agent, convert them to native actions, and make an invocation accordingly.

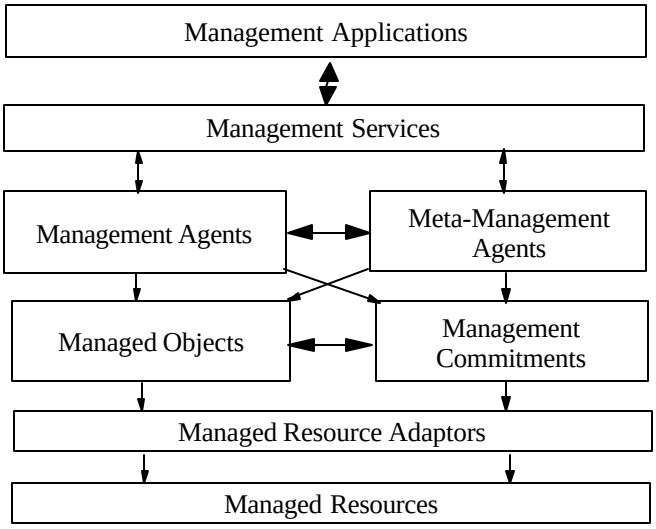

Figure 5 Layered Pattern.

Q3. How is the *transition* function implemented in BPSM?

A3. The *transition* function basically changes the internal state of BPSM according to the management commitments, business context and management events. It can be regarded as a meta-management directive. We use *state pattern* to implement this function [18].

Each state represents the overall situation of BPSM at a given moment. For example, if BPSM is in a state demanding high availability from the underlying managed system, it may create heart-beat sensors to beware of the health of the managed systems, and get ready to send high-severity alarms to concerned party if the system is really in trouble. Essentially, different internal state reflects different management commitment. Therefore, the BPSM's internal states are bound to management commitments and will be deployed to management agents when BPSM is booted up.

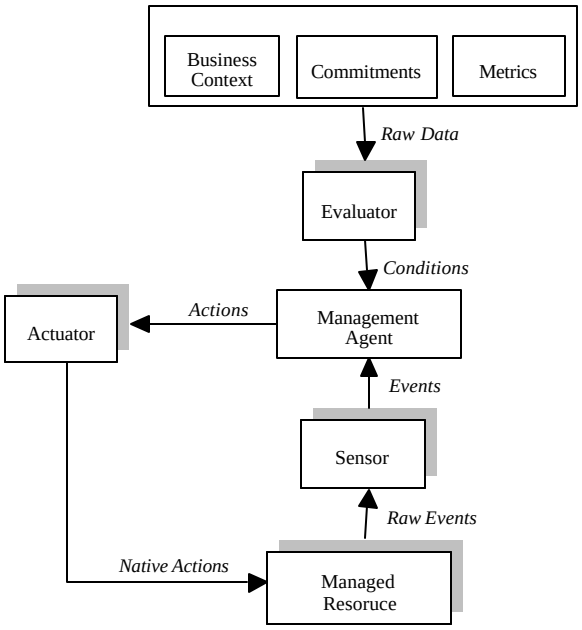

Figure 6 ECA Pattern used in BPSM.

Q4. How is the *adapt* function implemented in BPSM?

A4. The *adapt* function is a meta-management function since it changes the behavior of BPSM at the higher level. Hence, we use the *reflection* architectural pattern [6] to implement this function.

Figure 7 illustrates the reflection pattern used in the context of BPSM. The reflection architectural pattern enables the system to maintain information about itself and use this information to remain modifiable and extensible [19]. In general, this pattern embodies the notion of object/meta-object separation of concerns. A meta-object contains information about the internal structure and behaviour of one or more *base-objects*. That is, meta-objects can monitor and manage certain aspects of the base-objects.

A set of meta-objects is called *meta space* and a set of base-objects is called *base space*. There are two types of interactions between meta space and base space: (1) *Reflection* that enables a base object to reflect on its own run-time state and eventually modify it in order to change its behaviour during execution. (2) *Reification* that enables base-objects to access the meta-level knowledge. For a base-object to be supervised, it must be reified into the corresponding meta-space. The interfaces used for base-objects to access meta-objects are called meta-object protocols (MOPs).

The reflective management layer is implemented as the meta space; and the deliberate and reactive management layers are implemented as the base space. The meta space consists of meta-objects such as management commitments, business context, and meta management agent. The base space consists of base-objects such as management agents, managed resources, business processes etc.

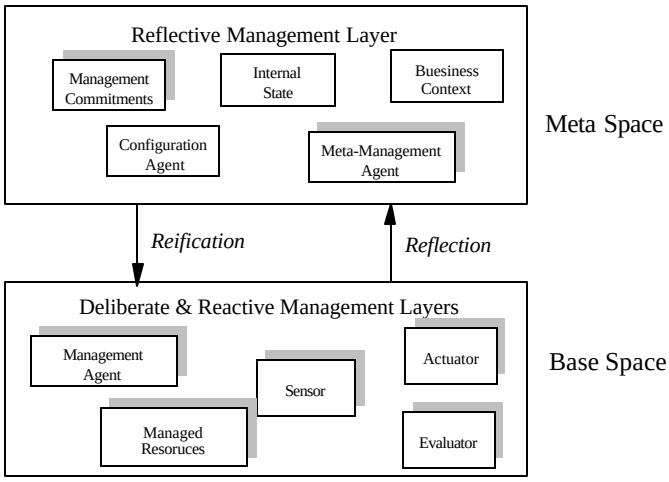

Figure 7 Reflection Pattern in BPSM.

Passive reification is done through the instrumentation process. Active reification is done by calling the APIs of the Configuration Agent from the base-objects. For example, a sensor can attach itself to a management agent by calling "attach" function of Configuration Agent.

Reflection can be achieved by opening APIs of meta management agents to the base-objects. Consequently, MOP is also defined for BPSM by defining the APIs of meta management agents.

Q5. How is the *perceive* function implemented?

A5. The perceive function receive events or fetch data from the managed resources. The main challenge for building sensors implementing perceive functions is how to adapt to the endless diversity of data formats and protocols. Our approach is using Façade pattern to develop management façades situated between sensors and managed resources, performing data transformation and/or protocol adaptation. A management façade consists of two parts: *managed object and managed resource adaptor.* The syntax and semantics of managed objects are dictated by the internal standard chosen by BPSM. On the other hand, the forms and functions of MRAs are really domain-specific and depend on what kinds of managed resources are to be instrumented and managed by BPSM.

Q6. How is the notion of *context* implemented?

A6. The architecture used for *context* is one based on the blackboard architectural pattern [20], which contains three parts: (1) *Blackboard*: Blackboard is a global data structure where the state information of the environment is stored and is organized into domain-specific ontological hierarchy. (2) *Knowledge Sources*: Knowledge Sources are separate, independent parcels of solution-dependent knowledge. (3) *Controller*: The controller drives the evaluation and

actuation processes. Evaluators and actuators are categorized as the controller in this pattern.

The blackboard contains information that characterizes the runtime behavior of the managed resources and BPSM. This characterization is accomplished by identifying entities and attributes that describe the run-time behavior of those entities. These are driven by management commitments which management policies and to-be-monitored metrics are deduced from. The blackboard data structure stores the current values of these policies and metrics. The "management agents" and "meta management agents" act as the knowledge sources. The controller (evaluator in BPSM) examines the values in the blackboard to determine if commitments are violated. If they are then an action is taken by the controller (actuator in BPSM).

Q7. How is *event mapping* implemented in BPSM?

A7. Many management events can be generated from the environment called raw events. Raw events are perceived by sensors and sent to interested management agents to process them. Before raw events become management events, some transformation needs to be done since an event at certain level of abstraction may mean nothing at different level. For example, an event of router failure implies high impact on network performance and demands immediate attention from network administrator. But, this event could mean little to a customer care business process that is concerned with the quality of services at the level of business process. However, low network performance very likely implicates degraded quality of services for the execution of business process.

To make the management agent who is in charge of customer care business process be aware of the potential impact, a mechanism of semantic event mapping needs to exist. We create an architectural pattern called *semantic event transformation* for this very purpose. We define an event mapping entity called Event Transformer (ET) at the semantic level. Each ET receives (raw) events from sensors or from other ETs, processes the events on the basis of embedded data, the management commitments, and the transformation rules, and sends the output to either registered management agents or other ETs at different semantic levels. Events are usually propagated from lower-level ETs , e.g., network domain, to higher-level ETs, e.g. customer care business process domain. Event transformation rules are derived from management commitments specified in MCL and are dynamically imported into ETs by the Configuration Agent.

Event transformation is particularly critical for the activities at the reactive management layer where substantial part of the processing is triggered by events. Event transformation resolves the problem of bridging the gap that exists between "events," which are reported by various channels, and the "reactive situations," which are the cases to which the system should react. These situations are composition of events or other situations (e.g. "when at least four events of the same type occurred") or content filtering on events (e.g. "only events that relates to inventory") or both ("when at least four alarms generated for the router failures have been processes on the platform where customer care business process is running in a week"). Hence, a situation can be regarded as a metamorphosed form of an event or a group of events, which embodies the semantic information that is meaningful to specific level of abstraction.

The event transformation used in ETs follows ECA pattern as described earlier and takes two-phased approach of filtering: composition filtering (event) and content filtering (condition part on the result of the first phase). In our solution, phase 1 combines the composition filtering with content filtering capabilities. This approach enables to construct more efficient reactive management relative to those which are being developed by current tools. The two-phase approach may be inefficient when the number of detected situations is much smaller relative to the number of the combinations that are produced in phase 1. Furthermore, the number of combinations produced in phase 1 can be exponential.

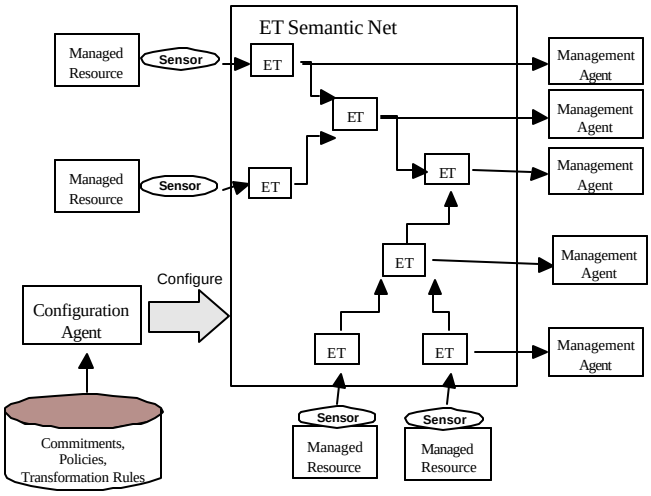

Figure 8 Pattern for Event Transformation.

The ability to combine composition and content filtering is a unique property, and it improves the performance in the general case, and enables the detection of situations that are not practically feasible in other solutions, in extreme cases. Figure 8 presents the architectural pattern for event transformation in BPSM, where managed resources act as event sources, and management agents act as event sinks. Event transformers (ETs) form an ET semantic net that contains nodes (ETs) and directional edges (event flows).

ETs generate situations based on rules, policies and management commitments that are configured by the Configuration Agent at both build time and run-time. Situations can be either consumed by management agents or transmitted to another ET for further processing. The full formalism can be described by the Colored Petri Net [21].

5 Implementation and Discussion

We have developed a BPSM -based prototype for Supply Chain Management system. Figure 9 shows the architecture of traditional supply chain management system. At the core of this system is *the supply and demand planning* systems shown in the top layer. Typically supply planning is focused on critical long lead time components (some semiconductor components may take 6 month lead times, for example).

Driven by the long horizon, demand planning becomes a lengthy process focused on long term risk-reward scenarios. The resulting control system, while updated periodically, relies on a fixed production target (build to plan) mechanism that is unresponsive to the rapid to medium term shifts in either market or supplier behavior that characterize the current environment. In a traditional "make and sell" model, there is no further response mechanism: product is built according to the plan and sold from inventory. Short term fluctuations are buffered by maintaining adequate inventory. One response is to drive the bottom layer of Figure 9 toward real-time operational control.

Two types of systems are emerging in this area, *inventory optimization* that controls manufacturing coupled with short term procurement based on inventory set points, and *event monitors* that give early warning of "real time" market events. A key issue with these techniques is how real-time decisions can be made in a consistent manner with planning decisions. We address this issue via re-engineering existing SCM system using the concepts and architectures embodied in BPSM. Figure 10 shows a BPSM-based Supply Chain Management system (called **BSCM**). SCM components are rearranged in different layers and new management agents are created.

BSCM is an autonomic integrated management system driving planning and execution in alignment with both strategy and business objectives through its *sense*, *analysis*, and *response* capabilities. BSCM takes a different approach than other management systems by viewing both supply chain planning and execution systems as part of the managed resources that are instrumented to be monitored, measured and managed by BSCM itself. Hence, supply chain planning system is not the highest sphere of control any more. On the contrary, the behavior of BSCM is driven by management commitments instead. The planning and predictive models are determined by the management commitments as well. The actual execution of managerial tasks is done by the infrastructure of BPSM with the consultation on management commitments, business context and the run-time events.

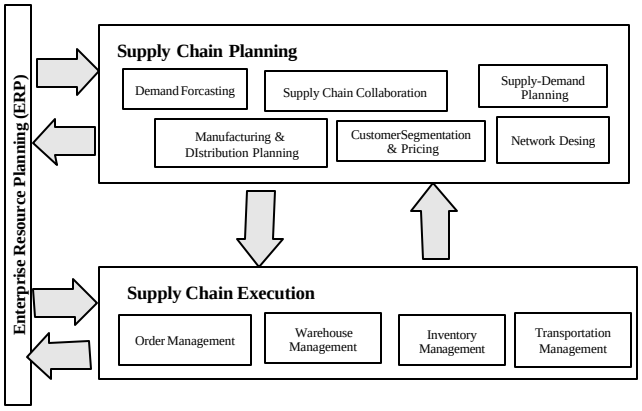

Figure 9 Conventional Supply Chain Management System.

BSCM enables automatic sensing of complex internal and external business environmental changes, and responds quickly with the best available policies in order to achieve the business objectives. The goal is to create a digital brain with sensors reaching all the way from a company's global value chain to the Internet world, identifying, monitoring, organizing and analyzing critical business information with intelligent decision-making capabilities in order to generate and activate proper business rules, policies and processes. BSCM provides visibility of supply chain events prioritization of exceptions, and issuing of exception alerts to help make timely business decisions at a tactical and operational level. The real time response agents are maintained at the layer of reactive management. Real-time data and information gathered from monitors is used as input to the response agent through the ET semantic net deployed by the configuration agent.

From our experience of developing BSCM, the advantage of using BPSM to develop Supply Chain Management systems are summarized as follows.

1. BPSM provides formalized conceptual framework for building management-oriented applications. It allows system architects and business analyst grasp the essence of the business requirements, system requirements, and their synergy in a very effective fashion. Due to the well-defined grid decomposition, management scenarios can be clearly charted on the grid in order to minimize design ambiguity.
2. The separation of management policies and management mechanisms reduces the confusion on

both sides. By following the approach of BPSM, policy makers can concentrate on devising management policies for concerned solutions, and system designers can focus on defining generic and reusable artifacts for to-be-built systems. Moreover, since BPSM embraces commitment-governed management, the management scenarios can be modified without re-implementing the system or even re-starting the system.

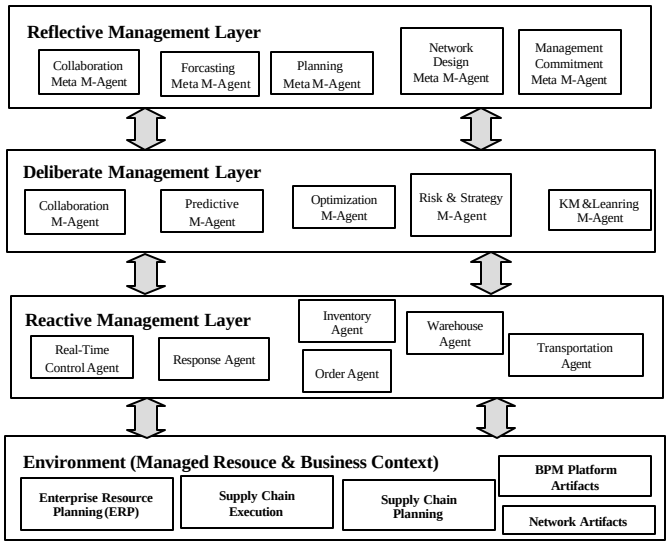

Figure 10 BPSM-based SCM System.

3. From the very beginning, the BPSM platform was designed for managing resources at various levels of abstractions. In this paper, only two levels, i.e. business process and IT, are stressed, however, the number of abstraction levels is unlimited in principle. Therefore, BPSM provides the highest flexibility for BSCM designers and developers to implement their concepts into the abstraction levels that are most appropriate.

4. The notions and components of Management Facade allow system developers easily adapt to managing various kinds of resources at different platforms. Since all managed resources have a uniform abstraction incarnated in Managed Objects, it is efficient to develop management agents without worrying about the issues of data conversion and protocol adaptation. Right now, we are using J2EE Management Specification [22] to model the Managed Objects.

5. The BPSM platform allows the semantic event mapping via the mechanism of policy-based event transformation. Transformation policies are again coded in the management commitments and can be dynamically loaded into the system to change the system behavior.

In summary, the BPSM platform provides an adaptive platform for managing business process solutions. It is especially suitable for being used as the base of building adaptive and dynamic management solutions covering various business processes, organizations, applications, and systems. It has been proven in the industry of business process integration and management that business process solutions need to be manageable and a platform to enable this manageability is in great need. The BPSM platform is our effort to address the problems.

6 Related Work

In the literature, we don't find many researches that are dedicated to developing platform for the domain of business process solution management. However, there are several technologies related to the components used to build the platform and the prototype for BPSM. Minsky and Ungureanu [23] described a mechanism called law-governed interaction (LGI), which is designed to satisfy three principles: (1) coordination policy needs to be coordinated; (2) the enforcement needs to be decentralized; and (3) coordination policies need to be formulated. BPSM satisfies all of the LGI principles. LGI uses decentralized controllers co-located with agents. LGI does not address the issues such as event transformation and meta-management.

BPSM uses ECA patterns to enforce the compliance of management commitments. ECA rules are also used in the area of active database that is strong at defining high-level business events but weak at lower-level IT events. In fact, the event composition operators used in active databases do not allow the correlation and transformation of events whose forms can only be determined at run time. With active database language, dynamic correlation can only be guessed by the designer at the build time. On the other hand, BPSM allows events can be perceived and transformed to desired forms according to policies specified in the management commitments. The performance of event transformation can be modulated through ET semantic net. In general, active databases do not address the issues of performance and scalability [24][25].

There are other efforts on system management domain, notably, Tivoli Management Environment (TME), and Microsoft Web-Based Management (WBEM) schema and protocols. Unlike BPSM, these are not targeted on business process solution management but on system and network management toolkits.

Over last years, Sun Microsystems has attempted to make Java an enabling technology for distributed systems management. The main products include JMX [26], Java

Dynamic Management Kit [27],and J2EE Management Specification [22]. Compared with BPSM, JMX and JDMK are more complementary than different. While JMX addresses the instrumentation of managed resources using MBean (Managed Bean), BPSM is more concerned about how various management agents can be coordinated and how managers can leverage the architecture to fulfill management requirements. Propagating and transforming events are not addressed in JMX.

7 Conclusions and Future Work

We described the conceptual framework and architectural patterns for BPSM. We showed how it can be used to build business process-centric solutions with managerial capabilities. An example of BPSM-based management application in the domain of Supply Chain Management has been presented to show the effectiveness of BPSM. It is a brave new world for research, development, technologies, and enterprise modeling. We will strive for shaping BPSM to make it more adaptive to different levels of abstractions and platforms. Security will be the imminent challenge for BPSM since managerial functions are usually highly critical to the success of business process solutions, hence, it is desirable that BPSM should have solid security mechanism to ensure the soundness of BPSM and its managed resources. Another challenge for BPSM is scalability. With growing number of managed resources and services, there is the risk that the federation of all entities inside BPSM will be freaky under performance stress. These issues will be discussed in subsequent papers.

ACKNOWLEDGEMENT

We would like thank Grace Lin for her support on the development of BPSM-based Supply Chain Management systems; thanks also go to Opher Etzion, David Botzer and Vladimir Shcherbina for their pioneering work on the technology of Active Middleware Management.

REFERENCES

[1] Arkin, A., Business Process Modeling Language (BPML), Working Draft 0.4, http://BPMI.org, March 8th, 2002.

[2] "SNMP Version 3 (snmpv3)," The Internet Engineering Task Force (IETF), http://www.ietf.org/html.charters/snmpv3-charter.html.

[3] Bumpus W. et al. Common Information Model: Implementing the Object Model for Enterprise Management, John Wiley & Sons, Dec. 1999.

[4] Management and Manageability: Manageability (M12) Model Core Specification, Version 1.3, Feb. 2002, IBM Internal Report.

[5] Anupindi, R. Chopra, S. Deshmukh, S.D. Mieghem, J.A.V. and Zemel, E. *Managing Business Process Flows*, Prentice-Hall, 1999.

[6] Buchmann, F., Meunier, R., Rohnerr, H., Sommerlad, P., and Stal, M., "A System of Patterns: Pattern-Oriented Software Architecture," New York: Wiley, 1996.

[7] Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM Research External Web Site, http://www.research.ibm.com/autonomic/.

[8] Sloman, A., Varieties of Affect and the CogAff Architecture Schema, in *Proceedings of AISB'01 Symposium on Adaptive Agents and Multi-agent Systems*, University of York, United Kingdom, March 21-24, 2001.

[9] Haeckel, S., *The Adaptive Enterprise – Sense & Respond*. Harvard Business School Press, 1998.

[10] Lin, G. et al., The New Frontier: Sense and Respond System for Global Value Chain Optimization, To be published in *OR/MS Today*, May 2002.

[11] Li, H., Jeng, J., Chang, H., Management Commitment Language for Business Process Solution Management, IBM Internal Report, May, 2002; also submitted to *IBM System Journal*.

[12] Supply-Chain Operations Reference Model, Supply Chain Council, http://www.supply-chain.org, Version 3.1, March, 2000.

[13] Lutfiyya, H.L., Bauer, M.A., Marshall, A.D., Stokes, D.K., Fault Management in Distributed Systems: A Policy-Driven Approach, *Journal of Network and Systems Management*, Vol.8, No.4, 2000, Pages 499-525.

[14] Wies, R., Policies in Network and Systems Management – Formal Definition and architecture, *Journal of Network and Systems Managements*, Vol. 2, No. 1, pp. 63-83, 1994.

[15] Sloman, M. and Twidle, K., Domains: A framework for structuring management policy, In Morris Sloman, ed., *Network and Distributed Systems Management*, pp. 433-453, Addison-Wesley, 1994.

[16] Tsalgatidou, A., Karakostas, V., Loucopoulos, P., Rule-Based Requirements Specification and Validation, in: Steinholtz, A. Solvberg, L. Bergman (Eds), *Proceedings of the Second Nordic Conference on Advanced Information System Engineering*, Berlin et al.: Springer 1990. pp. 251-263.

[17] Hanson, E.N., Widom, J., An Overview of Production

Rules in Database Systems, in: *Knowledge Engineering Review*, Vol. 8, No. 2, 1993, pp.121-143.

[18] Gamma,E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[19] Suzuki J. and Yamamoto Y. "Open WebServer" An Adaptive Web Server Using Software Patterns," IEEE Communications, Vol. 37, No. 4, April, 1999.

[20] Englemore, R., and Morgan, T., (Eds), *Blackboard Systems*, Addison-Wesley, 1988.

[21] Jensen, K., *Coloured Petri Net: Basic Concepts, Analysis Methods and Practical Use*. Monographs on Theoretical Computer Science, Springer-Verlag, 1992.

[22] J2EE Management Specification 1.0, JSR-077, Final Draft, http://java.sun.com/j2ee/tools/management/

[23] Minsky, N.H., and Ungureanu V., "Law-Governed Interaction: A Coordination and Control Mechanism for Heterogenous Distributed Systems," ACM Transaction on Software Engineering and Methodology, Vol. 9, No. 3, July, 2000, Pages 273-305.

[24] Casati, F., Ceri, S., Pernici, B., Pozzi, G. Specification of the Rule Language and Active Engine of Foro, Vol. 1, *WIDE Technical Report 3008-6*, 1997.

[25] Krishnamurthy, B., and Rosenblum, D., Yeast: A General Purpose Event-Action System, *IEEE Transaction on Software* Engineering, Vol. 21, No. 10, pp. 845-857, October, 1995.

[26] Sun Microsystems Inc. Java Management Extensions Instrumentation and Agent Specification, July, 2000.

[27] Sun Microsystems Inc. Java Management Tool Kit, White Paper, April, 2000.