ISeB (2004) 2:59–87
DOI 10.1007/s10257-003-0028-9

Information
Systems
and
e-Business
Management

A policy framework for Web-Service based Business Activity Management (BAM)

Jun-Jang (JJ) Jeng, Henry Chang, Jen-Yao Chung

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
(e-mail: {jjjeng,hychang,jychung}@us.ibm.com)

Abstract. Business Activity Management (BAM) is becoming one of the most critical areas to transform a business into an adaptive enterprise. To manage business activities and the related resources is a very complex task. BAM systems need to address potentially large number of business rules and unpredictable changes of business situations. Using BAM policies to drive BAM scenarios makes such challenging tasks practical and feasible. In this paper, we present a policy framework for Web-Service based BAM systems. We will cover the conceptual foundation, policy specification, policy architecture, and a case study for the framework and related components. Comparison with other similar works will be given at the end of this paper.

1 Introduction

Monitoring and controlling business activities requires dynamic instrumentation, adaptive infrastructure and sophisticated orchestration among subjects and objects. A new breed of applications called Business Activity Management (BAM) has emerged recently for enabling enterprises to manage their business solutions. BAM automates end-to-end workflows of all tasks related to monitoring and controlling business activities. BAM is set to lift the management technologies to the levels of both business processes and organizations. BAM will enhance visibility of business activities within an enterprise, enable early detection of business anomalies, and provide intelligent response to resolve business situations or exceptions. In many cases, a BAM system encompasses a set of applications, business processes, and standards that will redefine monitoring and management paradigm within process-centric industry, e.g., manufacturing and logistics.

However, to manage business activities and the related resources is a very complex task mainly because of potentially large number of operational rules and unpredictable changes of business situations. Furthermore, constantly changing business environment implies volatile requirements of monitoring and controlling business processes. Although Business Process Integration

(BPI) technologies are becoming popular in various industries, it has been a real laggard in any kind of BAM technologies. To monitor and control business processes (activities) tends to be labour-intensive and error-prone routine processes with minimal degree of automation. Frequently, a BAM system needs to interact with multiple business processes and systems. For example, there could be many points within business process where human intervention is required. Moreover, the flow and format from one business process to another happens in non-standard proprietary format. The issue of inter-operability is multiplied when a BAM client is attempting to communicate with external business processes and trading partners.

Traditional management paradigms such as Network Management and Application Management do not completely address all the above issues because most of such systems are focused on specific domains with predictable requirements from the customers. For example, the concept of "performance" is likely well-versed for network management systems, but it may have different semantics for different business processes and organizations. Most of conventional management systems operate in a homogeneous environment but BAM systems do not. On the contrary, interoperability is a major concern for developers when they are developing BAM systems. Generally speaking, the traits of interoperability and seamless integration are prerequisites of a useful BAM system. We need to eliminate much of the discrepancy among managed processes and data for the purpose of streamlining end-to-end business activity management both within an enterprise and across enterprises.

A key to the success of BAM systems is to make BAM itself as an adaptive system. In its current state of implementation within pretty much every institution, BAM systems are still limited in scope. Most of them are targeted on only a portion of BAM requirements, e.g., reporting. To provide new BAM functionality, the developers need to go through full development lifecycle. We argue that the primary reason for this state of affairs is the lack of an industry-wide policy initiative. BAM *policies* can be used to parameterize BAM operations and mandate each institution to build their BAM systems in a standard manner.

We take two approaches to develop an interoperable and adaptive BAM system: *Web Services* and *BAM Policies*. While this paper is aimed for describing the policy framework for BAM systems, Web Services are the cornerstone of the architecture of policy-driven BAM systems. Without the capabilities of interoperability and the de facto integration standards supported by Web Services, it is difficult to deploy and enforce monitoring and control policies into BAM systems, underlying components, and managed resources in a uniform fashion. Hence, in our framework, Web Services are the preconditions of making policy-driven BAM systems possible.

The rest of this paper is organized as follows. Section 2 presents the target domain on manufacturing processes in supply chain domain and shows the BAM management stack based on the target domain. Section 3 describes the Web Service based BAM framework that is used as the foundation of building BAM policy framework. In Sect. 4, we describe basic policy models for the BAM Policy Framework including the policy lifecycle model, policy meta-models and policy grammar. A reference implementation with a case

study will be illustrated in Sect. 5. We include related work in Sect. 6. And Sect. 7 presents our conclusion and further work.

2 Problem statement

Firstly, this section describes the target domain where we developed our BAM platform and solutions. Secondly, we present the conceptual foundation upon which the BAM policy framework is built.

2.1 Problem domain

The problem domain we have been focusing upon is the area of supply chain management (SCM) for microelectronic manufacturing (Fordyne 2001). An SCM process consists of four steps: *demand creation/forecasting*, *production planning*, *manufacturing execution*, and *available to promise*. Furthermore, the decisions in the semiconductor industry typically fall into one of four decision tiers: *strategic*, *tactical*, *operational*, and *response* (dispatch). The categories are based on the planning horizon, the apparent width of the opportunity window, and the level of precision required in the supporting information.

The decision tier of *strategic scheduling* is driven by the time frame required for business plan, resource acquisition, and new product introduction. The timeframes concerned here are from three months to seven years into the future. The tier of *tactical scheduling* deals with problems the enterprise encounters in the next week to six months. Issues considered are made of yields, cycle times, and binning percentages, delivery dates estimated for firm orders, available "outs" by time buckets estimated for bulk products, and daily going rates for schedule driven product are set. The tier of *operational scheduling* deals with the execution and achievement of a weekly plan such as the shipments are made, serviceability levels to be measured, recovery actions to be taken.

The tier of *real-time response* system addresses the problems of the next hour to a few weeks by responding to conditions as they emerge in real time and accommodate variances from availability assumed by systems in the plan creation and commitment phases. BAM systems are built to realize the capabilities of "sense and respond" (Lin et al. 2002) in order to satisfy the requirements addressed in the above tiers by providing awareness among various elements such as humans and business artefacts, etc. A BAM system dispatches scheduling decisions regarding either monitoring or controlling policies of actual manufacturing flows; and instructs the operators about the next steps of achieving manufacturing commitments.

2.2 An example scenario

An example of typical use case for continuous demand-driven *build plan* and *inventory optimization* in the domain of microelectronic manufacturing can be described as follows. End-of-quarter *revenue targets* (per module family) are released/updated after the meetings among business line managers and executives. A business line manager (BLM) has a pre-determined set of

module families for which he has financial responsibility and, therefore, whose actual revenue (accumulated so far) and revenue outlook (for remaining weeks in the current quarter) he is interested in tracking against the revenue target of the current quarter. Whether the progression of the accrued revenue is normal or below target is determined by the system using a wineglass model (Wu et al. 2002).

2.2.1 Main course of the scenario

- On the *i*th day of the current week, a BLM selects a set of saleable part numbers to view the future weekly actual sales and planned demand quantities through BAM monitoring portal.
- Detection of Situation and Alert:
  - BAM issues an alert showing the current sales quantities of some selected saleable part numbers in the *n*th week are out of their bands
  - BAM recommends adjusting the planned demand quantities and safety stock requirements for the *n*th week.
- Recommendation:
  - BAM invokes demand planning module and inventory planning module to provide recommended demand quantities and safety stock requirements for the *n*th week.
  - BAM disaggregates the weekly demand quantities into daily demand quantities.
  - BAM recommends altering daily build plan in order to optimally match new daily demand statements, thus high serviceability, and minimize manufacturing and inventory costs.
  - BAM invokes MRP explosion and implosion module to generate optimal daily build plan, including common wafer start quantity and manufacturing release quantity in each part number level within the BOM chart associated with the selected saleable part numbers.
- Prediction and Risk Assessment:
  - BAM predicts the to-be manufacturing cost, inventory cost, and service level associated with the selected saleable part numbers based on new demand statements and new build plan.
  - BAM predicts the as-is manufacturing cost, inventory cost, and service level associated with the selected saleable part numbers based on new demand statements and old build plan.
  - BAM concludes the financial and serviceability benefits of applying newly recommended demand statements, safe stock requirements, and build plan.
- Decision Making by BLM:
  - The BLM summarizes the financial and service benefits and reports to the strategic management team (BLE's and GM's) for their approvals.
  - Upon the approval, the BLM releases the new build plan to sites for manufacturing execution. The BLM releases the new demand statements to procurement team to alter buy plan.

Enterprise business activities usually have the following common characteristics:

- Individualism: Multiple organizations are often involved in the business process. Each organization attempts to maximize its own profit within the overall activity. The individuality makes the enterprise goals extremely diverse and volatile.
- Physical separation: Organizations are physically distributed. This distribution may be across one site, across a country, or even across continents. This situation is even more apparent for virtual organizations which form allegiances for short periods of time and then disband when it is no longer profitable to stay together. Thus, the layer of *virtualization* becomes very critical in terms of making decision making processes seamless across multiple distributed organizations.
- Distributed management: Within organizations, there is a decentralized ownership of the tasks, information and resources involved in the business process. Hence, the style of centralized control does not fit the modern management requirement.
- Autonomy: Different groups within organizations are relatively autonomous-they control how their resources are consumed, by whom, at what cost, and in what time frame. They also have their own information systems, with their own idiosyncratic representations, for managing their resources.
- Concurrency among business activities: There is a high degree of natural concurrency - many interrelated tasks are running at any given point of the business process.
- Visibility of business activities: There is a requirement to monitor and manage the overall business process. Although the control and resources of the constituent sub-parts are decentralized, there is often a need to place constraints on the entire process (e.g. total turn-around time, financial performance, etc.).
- High adaptability: Business processes are highly dynamic and unpredictable and it is difficult to give a complete pre-defined description of all the business activities that need to be performed and how they should be ordered.

2.3 Decomposition of BAM Space

BAM is aimed for providing an adaptive management platform for monitoring, controlling and managing business process focused solutions. The BAM space can be modelled into two dimensions: (1) degree of responsiveness; and (2) levels of abstraction. The former categorizes the BAM systems based on the responsiveness and the extent of deliberation for their response to business situations and specifications. The latter classifies the BAM systems into different levels of abstraction. The first dimension on the degree of responsiveness encompasses: reactive management, deliberative management and reflective management. The second dimension on the levels of abstraction encompasses three levels: execution, operation and strategy. Figure 1 illustrates the BAM space as described.

2.4 Decomposition based on degree of responsiveness

Along the axis of degree of responsiveness, we can have the following classification:

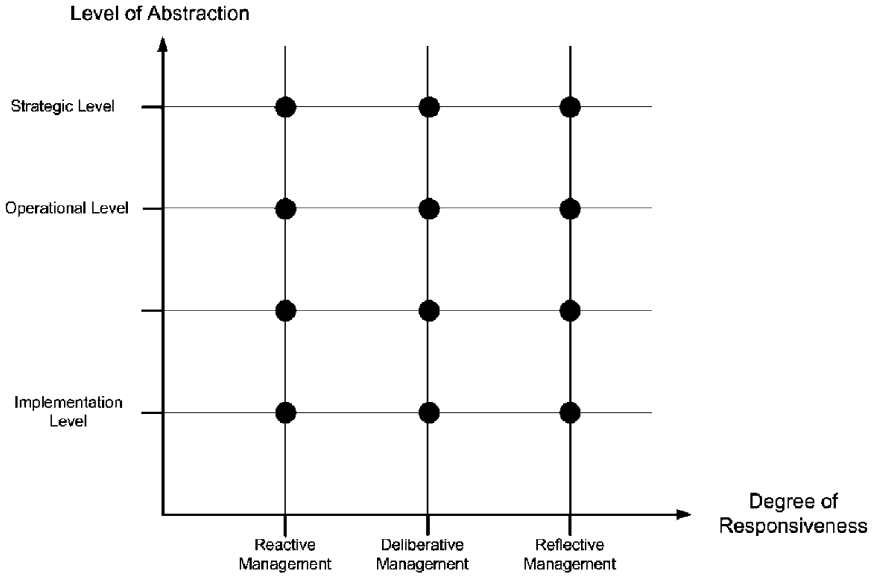

Fig. 1. BAM space

- *BAM Reactive management* layer responds to the management events quickly and directly through scripted business process models. Most of BAM scenarios can be addressed by modelling them into this layer. The implementation of this layer can be many folds. Usually, the timeliness is a critical concern for those scenarios implemented in this layer. A notable example of the reactive management is to detect the situation when a business process is shutting down by unforced errors such as electrical outage. In such case, an alarm needs to be sent to concerned parties immediately.
- *BAM Deliberate management* layer performs management tasks that require more reasoning and more complicated computation. It is not uncommon that an adaptive infrastructure needs to provide decision support capability so more intelligent management actions can be derived towards managed resources. An example of such managerial task is the business processes with the ability of sense-and-respond (Lin et al. 2002).
- *BAM Reflective management* layer enables adaptive infrastructure to maintain information about itself and use this information to remain extensible and adaptable (Buchmann et al. 1996; Gamma et al. 1995). Reflexive management layer performs meta-management directives unto the lower management layers and managed entities.

A meta-management directive is a higher sphere of control such as adapting the management commitments, modifying measurement and analysis algorithms in the deliberative management layer, or changing the alarm rules in the reactive management layers. As such, an adaptive infrastructure can have detailed knowledge of the managed resources, current status of managed business processes and business systems, the ultimate capacity in

the inventory, performance expectation, and all connections to other systems to manage itself. Therefore, through reflective management mechanism, adaptive infrastructure achieves the goals of both 2nd order management and autonomic computing.

2.5 Decomposition based on levels of abstraction

We have adopted layered approach to modelling BAM solutions. There are four levels of abstraction to classifying BAM models as follows.

- The **Strategy model** defines the business goals and objectives in the form of balanced scorecards representing a quantification of goals, and measurable objectives. A balanced scorecard expressing each perspective as a combination of objectives initiatives, measure and target will indicate how well this model is performing.
- The **Operational model,** typically developed by operation executives in collaboration with strategy executives, describes the process in business terms, commitments, and metrics which get mapped to the scorecard for comparison with their strategic targets. The metrics also known as Key Performance Indicators (KPI) The KPIs are directly linked to the measures that indicate progress on Balanced Scorecard goals in the Strategic model.
- An operational model would be transformed into an **Execution model** which defines how the business operation is executed in terms of specific applications, data sources, people and partners. It does not assume a particular implementation, allowing iterative performance improvement while assuring consistency with the business objectives.
- Finally, some development may be required to connect the platform-independent execution artifacts to a specific platform's **Implementation model** such as the WAS J2EE or MS .NET platform, and develop specific APIs using Web Services, for example.

Figure 2 shows the approach of model-driven BAM where the BAM models at different levels of abstraction are authored and deployed to corresponding M&M (monitoring & management) runtime infrastructures. Business solutions emit business and IT events to M&M runtime for further analysis based upon pre-defined BAM models. Monitored data is propagated in a bottom-up fashion and the management directives are rendered in the other way. Hence, the lower-level decisions and actions are governed by those in the upper levels. Note that the end users of BAM systems are usually different from the policy makers. Same observation can be applied to the participants at different levels of abstraction.

With these mappings, transformations, and connections in place, raw events, transactions, and environmental data can be captured and aggregated into business metrics. Business commitments can be used to compare business metrics and desired objectives. The software components running in a manufacturing execution system can be depicted as three kinds of agents: *execution* agents, *operational* agents and *strategic* agents. Execution agents are those interacting directly with the manufacturing systems, e.g., inventory agents or data agents. Operational agents include planning agents, MRP

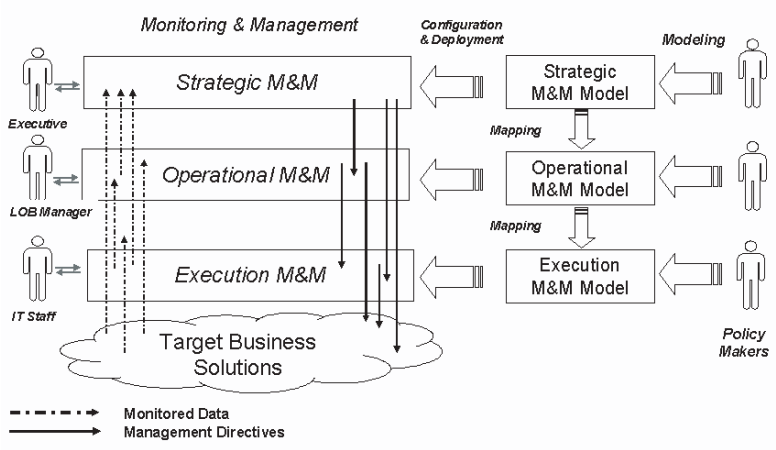

Fig. 2. Policy-driven business activity management

agents, alert and situation agents that generate business exceptions. Strategic agents are those agents that make recommendation and provide risk assessment. If two agents are performing semantically related tasks, they need to be aware of each other. In other words, they have to be ''closer'' to each other cognitively.

In this paper, the governance of interaction among agents is enabled by BAM Policy Framework that is presented in following sections. BAM systems, as a whole, can be viewed as an ecosystem composing of various artefacts such as business entities, business processes, and business participants, with distance functions connecting them to provide just-in-time information and cues for decision making. Figure 3 depicts the management food chain for BAM agents and the degree of the values added by those agents. The systems at the bottom are mostly legacy systems such as workflow management systems, inventory management systems, scheduling systems, and databases. The next level up in the food chain is the execution agents that interact directly with the E-Commerce systems and information sources. The execution agents consume the

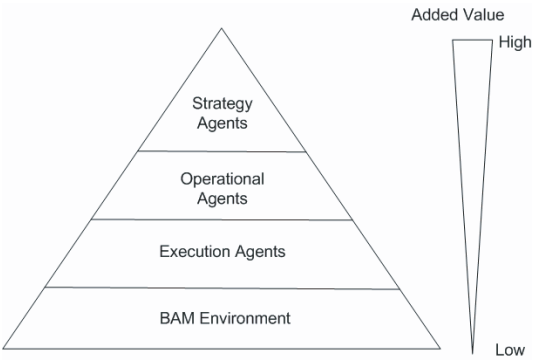

Fig. 3. BAM food chain

information generated by the lowest level E-Commerce systems, and render control on those systems on behalf the agents at the higher level of the food chain. The examples of execution agents include information agents and broker agents.

The operational agents detect the situations produced by the execution agents, conduct the analytics activities, and respond to the situations by rendering management actions back to the execution management layer. The examples of such agents include analytics and planning agents. The highest level of the management food chain is the layer of strategic agents that detect the situations from operational agents, conduct strategic analysis (automatically or cooperatively with decision makers), and respond to the situations based on existing enterprise goals and policies. The examples of such agents include decision assistant agents, risk analysis agents and so on.

3 Web-Service based BAM

A BAM system is composed of BAM Web Services. Each BAM Web Service (noted as WS-BAM) serves as a BAM agents collaborating with one another to fulfill the BAM requirements imposed by clients. A BAM-WS must make its choices among potential reasoning tasks to dynamically fulfill local and global objectives. For example, different manufacturing domains may require different optimization algorithms for performing customer demand forecasting analysis. Hence, a BAM-WS should be able to understand the business context where it is running and select the best analytic algorithm for such purpose.

A BAM-WS must adapt its control mode to dynamic goal-based constraints on its actions and uncertainty about its environments. The execution environment in an enterprise is usually diverse. Various kinds of devices, hardware, software, storages, and networks are likely to coexist in the same enterprise. When a BAM-WS is about to render control to its environment, e.g. changing build plan for certain manufacturing line, it must be able to detect the control mode based on the target objects to be updated or modified. Moreover, A BAM-WS must adapt its meta-control strategy to its dynamic configuration of demands, opportunities, and resources for behavior.

BAM reference model describes the requirement space of developing BAM-WS components in the domain of business activity management. A BAM-WS is a structure that is stable, coherent, and comprises several BAM-WS components as substructures. A BAM-WS is also used as a modeling construct describing the full range of business entities operating within all operating environments. It is similar to the concept of *virtual business units* described in IBM BizADS methodology (Gamma et al. 1995).

BAM-WS components are organizational units that contain the following characteristics:

- Autonomy: the capability of business units to create and control the execute its own decisions and policies;
- Collaboration: the process whereby a set of business units develop mutually acceptable plans through negotiation and execute them

– Optimization: the capability of business units to collect and arrange themselves in order to achieve a business goal; and
– Re-configurability: the ability of a function of a business unit to be simply changed in an efficient and cost-effective fashion.

A BAM-WS can be constructed based on the requirements of a set of business activity management. Developers can use and develop simple notations (e.g. UML) for capturing functional and non-functional requirements for WS-BAM components. A BAM-WS component potentially provides seven BAM perspectives: *Context*, *Intent*, *Value*, *Capability*, *Process*, *Constraint*, and *Resource*.

- The **context perspective** describes the environmental information for the existence and behaviour of a BAM-WS. The business context is largely about the relationships among BAM-WS components. Being aware of business contexts is essential for driving effective development of BAM-WS component. The business context is largely about the relationships with other BAM-WS components. Relationships govern value exchange among BAM-WS components.
- The **intent perspective** describes the goal of a BAM-WS and relates the goal to the other entities such as its capabilities, business values, and the goals of other WS-BAM components. The intent of a BAM-WS should have states corresponding to its context, capabilities, and values: a) Qualified intents that specified capabilities and values for meeting the intents; b) Fulfilled intents that have been satisfied; c) Violated intents that have failed meet the original conditions. A violated intent can be fulfilled if the context and internal values are changed towards satisfied states.
- The **value perspective** is the report and value-exchange view of the concerned business activities. This uses the monitoring view heavily to sample and analyze the operational performance of target business activities. This view helps a BAM-WS determines if it has achieved its goals or otherwise.
- The **capability perspective** specifies and links what a BAM-WS can do, from the strategy level to operation, to execution level, and to the resource level. These kinds of capabilities exist: a) the capabilities belonging to BAM-WS itself; b) the capabilities belong to its "child" BAM-WS components; and c) the capabilities that are not realized yet. The last group of capabilities can be concretized through Web-Service development process or collaboration with other BAM-WS components.
- The **processes perspective** describes how BAM-WS components cooperate with one another and harness passive resources in order to execute capabilities and to desired outcomes. Events trigger the commencement of processes. Interactions connect process steps. The process perspective of a BAM-WS is closely related to its capability perspective, where the latter defines "what" a process can perform upon and the former specifies "how" the capability can be realized by the BAM-WS. For some BAM-WS process perspective, five types of supporting logical processes have been identified: *sensing*, *detection*, *analysis*, *decision*, and *actuation*. They will be described shortly.
- The **constraint perspective** regulates the logical relationships among values, capabilities, intents, and contexts. Constraints impose a set of configurable

commitments and rules upon designated BAM-WS components. Constraints specify pre- and post conditions of invoking BAM-WS components in the BAM processes. Constraints also define structural relationship among BAM-WS components in terms of both *types* and *instances*.

- The **resource perspective** specifies the resources governed by certain BAM-WS. Managed resources may belong to different business units. A BAM-WS may be created to perform a complex business process simulation, or for several members of a group to come together to handle an emergency like a shutdown of target business solutions. A BAM-WS can represent a business unit to manage its associated resources. Resource sharing is possible among BAM-WS components, hence, the policies such as security and authorization need to be considered and well-defined.

Figure 4 shows the aforementioned seven perspectives of the reference model for BAM-WS components. The perspectives *value*, *capability*, *intent*, and *constraint* constitute the state of the BAM-WS. Similarly, the perspectives *context* and *resource* constitute the state of the BAM system itself that is "perceived" by the BAM-WS. The BAM-WS has five concurrent supporting logical processes for corresponding processes that are defined by its *process* perspective: sensing, detection, analysis, detection and actuation. Each logical process is realized by a Web service.

- **Sensing** Web service monitors and collects desired data (based upon the intents) from the BAM environment. The output from the sensing process is in the form of metrics or key performance indicators (Supply Chain Operations Reference Model, 2001), which provide a virtualization layer for the other logical processes: *detection*, *analysis* and *detection*.
- In general, the BAM-WS that has sensed events trigger **Detection** Web service, which, based upon perspective of intent and constraint, detects business anomalies and exceptions by evaluating the data and key performance indicators that are generated from target business solutions.
- Both business situations and intents can trigger the **Analysis** Web service in order to explore the options of resolving detected business situations. The

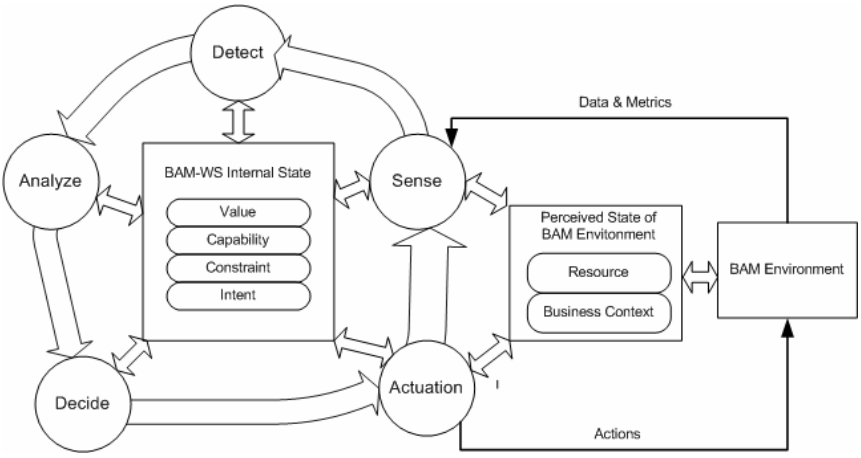

Fig. 4. The BAM-WS model

analysis Web service helps determine the root causes of the identified situations or exceptions. Key for the determination of causal factors is the ability to identify reliable relationships and interactions among variables that impact the business performance.

- **Analysis** Web service also generates alternative resolutions to resolve current business situation and help decision-making BAM-WS components to select the best solution. **Decision** Web service determines the ''optimal'' action that will be rendered to target business solutions. Such decision making Web service may be involved very complicated business intelligence modules that are carried with other BAM-WS components.
- **Actuation** Web service renders appropriate business actions based on the decision that has been made. This response may simply either change the state of the business activities or notify other BAM-WS components which are interested in the outcome of the very decision making.

Note that the Sensing and Actuation Web services should be context-aware, i.e., understanding the semantics of target business solutions. Such awareness is accomplished through the environment models described by the context and resource perspectives. Each BAM-WS represents either a management agent or a managed resource (such as business processes or organizations).

Figure 5 shows the interaction among BAM Web services. Five supporting web services are bound with BAM-WS *abc*. The data is propagated from data source to the Sensing Web service, and a message is sent to BAM-WS *abc*. This data will be sent to the Situation Web service by BAM-WS *abc* for the sake of detecting business situations and exceptions. If some situation is detected, then the controlling BAM-WS will send a message containing the situation data to the Analysis Web service to request a set of recommended resolutions in order to resolve the situation. In this example, the BAM-WS *abc* attempts to get an ultimate decision from its bound Decision Web service which, however, is unable to make such decision. Instead, it sends a request

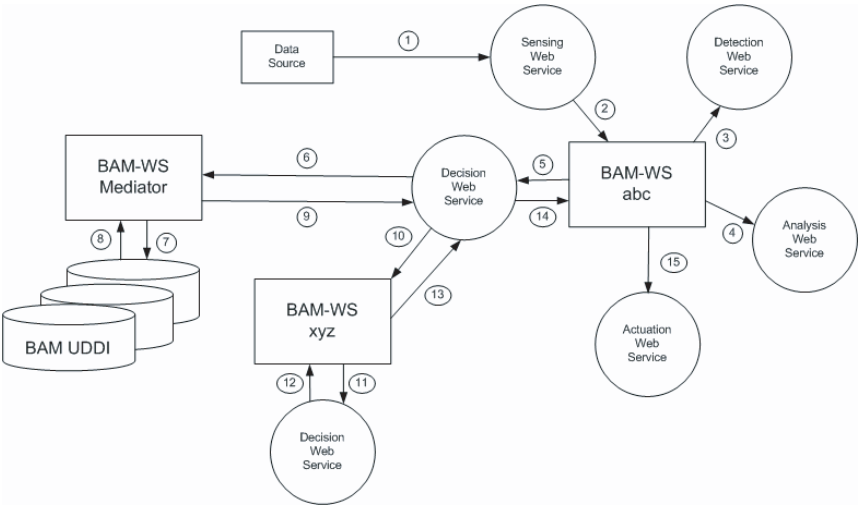

Fig. 5. Interactions between BAM Web services

of decision making to a Mediator Web service to understand who is able to make such decision. From the repository of BAM Web services, the Mediator Web service discovers the most suitable BAM-WS which has the capability of making desired decision. In the steps 10-13, the decision making process is ''outsourced'' to another BAM-WS *xyz*. Finally, a decision is received by BAM-WS *abc* and then it will send a message to the bound Actuation Web service to render actions to the target resources.

4 Policy-driven BAM

This section describes the general policy models incorporated in this framework. Three major components in the framework will be presented: policy lifecycle, policy meta-model and policy grammar.

4.1 Policy lifecycle

One way of achieving adaptive BAM infrastructure is through the development of BAM policies and using such policies to configure BAM systems. A *policy* is defined as a management directive specified by system administrators, which manages certain aspects of the desired *outcome* of interactions among users, applications and services in a distributed system (Verma 2000). A BAM policy aims to govern and constrain the behaviour of BAM systems. It usually provides policy rules for how the BAM system should behave in response to business situations. As an example, a policy of supply chain inventory may impose limits on the range of inventory levels for the manufacturing process based upon the revenue target of the enterprise. Relevant policies can be devised and applied to different aspects of BAM solutions. Examples include role-based authorization to manage target business solutions and resources, the scope of managed business solutions and resources, and service-level agreements. This paper is focused on the policy framework by presenting the policy foundation, the policy meta-models, the policy specification, and the policy architecture.

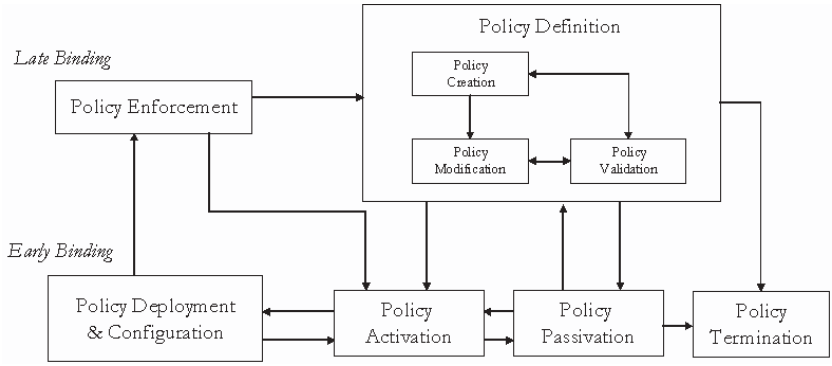

Fig. 6. Policy lifecycle

Every BAM policy has its own lifecycle. The lifecycle of a policy consists of six basic life-stages as shown in Fig. 6. They are: *policy definition, policy activation, policy passivation, policy deployment and configuration, policy enforcement* and *policy termination*.

- *Policy Definition* is the phase that a policy is created, browsed and validated. Corresponding definitional tools such as editor, browsers and policy verifiers can be used by business analysts to input the different policies that are to be effective in the BAM system.
- The stage of *Policy Deployment* & C*onfiguration* configures and deploys a policy into target system and configures the system correspondingly. A set of automated deployment & configuration utilities will usually simplify the tasks performed in this phase.
- *Policy Enforcement* is the stage when a policy is being enforced to govern and constrain the behaviour of target systems. Monitoring and reporting tools enable policy makers to understand how the status of policy enforcement and whether the policy has been defined reasonably.
- *Policy Activation* is the phase when a policy is loaded into target system and waiting for further execution. In this phase, policies are active in the memory but have not been committed to any business activities yet.
- *Policy Passivation* is the phase when a policy is put to persistent storage without any active activity. For BAM, a policy repository is usually required as the placeholder for passivated policies.
- *Policy Termination* is the phase when a policy ceases to exist in the system.

Potentially, a policy can be bound to BAM systems at two points of its lifecycle: (1) *policy deployment & configuration:* this type of binding is called *early binding* between policy and mechanism since it is realized at the build time; and (2) *policy enforcement*: this type of binding is, on the other hand, called *late binding* between policy and mechanism since this binding is realized at the run time when policy is being executed.

A deployed (configured) policy can be un-deployed (un-configured) and rolled back to the *policy activation* phase. By the same token, an *enforced* policy can be de-enforced and transits back to the *policy activation* phase. As mentioned above, a business analyst can use monitoring tools to monitor the status of policy enforcement in the policy target. If she thinks the policy does not meet her business goals, she may stop the execution and transition the policy into the *policy definition* phase in order to modify that problematic policy.

With policy lifecycle in mind, we developed a high-level logical architecture as the framework of defining policy components and services that will be presented in later parts of this paper. The policy logical architecture is built upon the policy frameworks defined by both IETF (IETF Policy Framework Working Group) and DMTF (Distributed Management Task Force Policy Working Group). The logical architecture consists of seven basic elements as shown in Fig. 7. The basic elements are: the policy management tools, the policy repository, the policy enforcement points, the policy decision point, the policy execution instances, the policy decision points and BAM model repository.

The *policy management tool* is used by business analysts to feed policies into BAM systems. The locations that can apply and execute BAM policies

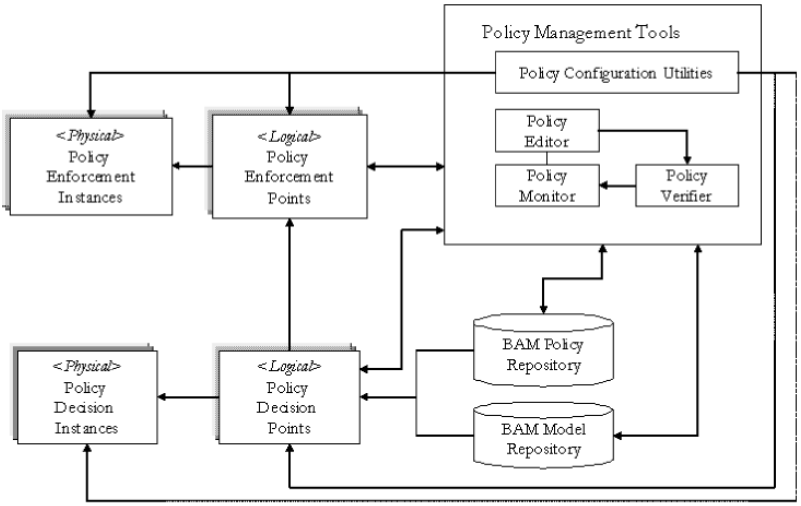

Fig. 7. Logical architecture

are named as the *policy enforcement points*. The preferred way for the management tool and policy targets to communicate is through a *policy repository*. Instead of communicating directly with the repository, a policy enforcement point can also use an intermediary known as the *policy decision point*. The *policy repository* is used to persist the policies generated by management tools. The *policy decision point* is responsible for interpreting the policies stored in the repository and communicating them to the policy enforcement point. .

The major difference between BAM Policy Framework and traditional policy framework such as IETF is the concept of *virtualization*, where decision and execution points can be decomposed into two levels of abstraction: the *logical* level and the *physical* level. Both policy enforcement and decision points belong to the logical policy level, which represents the fact that they are actually the virtualized services of underlying physical decision points and physical enforcement points. To make policies to be enforced into policy targets, the instances of policy decision and policy enforcement at the physical levels must be deployed and made to be available in advance. The BAM Model Repository is used to manage the models of target systems in the persistent storage.

Each decision and enforcement instance in policy target may be operating in one or more roles within the system, the role defining the set of policies that would need to be implemented for a particular service. As an example, in a supply chain management system, all inventory related modules have the role of ''Execution'' while all demand planning modules have the role of ''Operation'', and they retrieve their respective policies depending on their roles. The specification of policy rules within the policy repository is the key to obtaining inter-operability.

An information model for the rules to be specified has been defined. The underlying premise in the information model is that each policy should be considered as an expression of the form *event-condition-action* pattern. Here,

the event part describes the business events emitted from the target system, the conditional part describes a specific situation that can be encountered by a system, and the action part specifies the action that is to be performed by the system when situation is raised. The information model defines the structure of the policies that are to be stored in the policy repository.

The important classes and their attributes from the common information model are shown in Fig. 8. For any discipline to which policies can be applied, the information model can be refined to define a discipline-specific information model. This information model will be transformed into detailed policy specification components in next section. The discipline-specific information model would typically subclass Constraint and Action.

A very simple application of the model the aforementioned supply chain management system can be obtained by defining the following conceptual entities:

1. *Constraints* that contains the policy-specific domain attributes such as inventory levels, customer serviceability and satisfaction, and demand forecast accuracy;
2. *Actions* in that some of them are specified for re-calculating build plan of the target products, and the other for re-adjusting inventory policies;
3. *Business Events* that serve as the triggers of policy evaluation and enforcement.

A business event is domain-dependent and needs to be examined and elaborated by the experts of target domains. The *Context* defines the working memory for the execution of policy rules. Similarly, the definition of the context related model is really dependent on the models of target business solutions. *Scopes* are used to form a set of managed resources in target business solutions where the concerned BAM policy is applicable. Before evaluating or enforcing a policy rules, the Scope definitions needs to be checked because Scope can be dynamically changed. Considering the supply chain example presented in Sect. 1, an example for the information model can be modelled and described as follows:

- *Business Context*: microelectronic manufacturing processes, applications, data storages and customers.
- *Scope*: supply chain performance for the process for manufacturing certain product family.

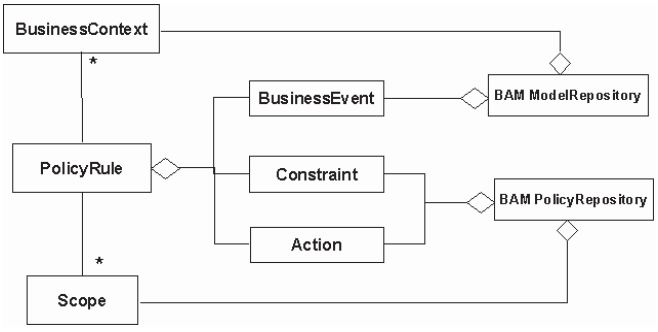

Fig. 8. Policy information model

- *Business Events*: order changes from the customers.
- *Constraint*: predict the impact on the performance.
- *Action*: (1) run prediction module for supply chain performance; (2) if the predicted performance is lower than the committed value, the re-planning process will be invoked in order to obtain new build plan.

4.2 Policy meta model

As described in the supply scenarios, an agent can be of many types: execution, operation and strategy. In general, they are called *Management Agents* (MA). The central class of the BAM policy meta-model is the virtual business unit (VBU). VBU provides a single manageable construct to describe the *capabilities* and *constraints* of underlying managed resources and business activities. An example of VBU can be supply chain management process. Its committed performance becomes its capability and its required resources are its constraints. A VBU has its own internal state and exposes a set of views to MAs and other VBUs. In the environment of business activity management, a management system is a collection of MAs to be performed on VBUs that are hosted in an environment tailored for one or more particular business organizations.

Management Enablers (MEs) constitute an abstraction layer between MAs and VBUs. A notable usage of ME is binding an MA and its managed VBUs by providing physical URLs and security credentials. The introduction of MEs increases the flexibility of interactions among MAs and VBUs, and enhances the overall productivity of the staff performing the management tasks through MAs. The MAs and MEs cross management disciplines, and operate on VBUs. Different levels of abstraction can be applied to VBUs using appropriate instrumentation unto target business solutions. For example, an inventory VBU can be instantiated into a customized inventory system and complex warehouses that are equipped with advanced monitoring and management capabilities. Hence, an example of instrumentation tasks could be configuring the inventory warehouse by specifying interested data *dimensions* for particular set of MAs. Another possibility is that the VBU may be merely an interface to an existing inventory system that has been implemented by specific technology such as SAP.

Three key *virtual* entities are identified in the BAM Policy Framework: management agents (MA), virtual business units (VBU), and management enablers (ME). The framework is comprehensive and simple to use for both business analysts and system developers. The benefits of policy virtualization are twofold: *i)* a policy can be independently defined for any target artefact and, *ii)* since a BAM system manages business activities and underlying resources through virtual entities, new physical policies and managed resources can be added or removed from the system without the need of modifying the policies or manually managing the association between policy and managed business activities/resources.

Figure 9 shows the base meta classes in BAM policy framework.

MA Policies define what activities an MA must do or refrain to do to a set of VBUs and define the duties of the MA. MA policies are triggered by

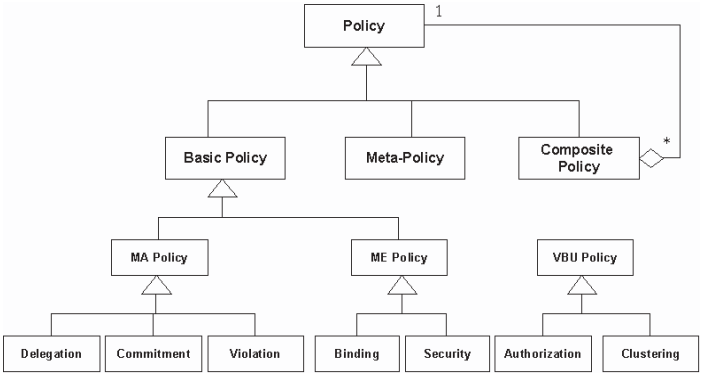

Fig. 9. Policy base classes

business events and are normally interpreted by an MA. The policies that belong to this type include commitment policies, violation policies, and delegation policies.

Commitment policies define what activities a subject MA must do to a set of VBUs and define the duties of the subject MA. *Violation* policies define what activities a subject MA must refrain to do to a set of VBUs and define the duties of the governing MAs in a negative way. Penalty and compensation activities are usually specified for violation. Delegation policies specify which actions MAs governing certain VBUs are permitted to delegate to other MAs. Hence, *delegation* policies specify authorization to delegate such management actions.

VBU policies define the constraints and obligations that a VBU needs to obey. At lease two type of policies fall into this category: authorization and security policies. Authorization policies define what activities an MA is allowed or prohibited to perform to a set of VBUs. Security policies are used to protect target VBUs sothey can be interpreted and enforced by one or more than one MAs.

ME policies define the policies enabling MAs access, monitor and control target VBUs. Examples of ME policies include binding policies, security policies, and privacy policies. *Composite policies* are used to group a set of related policies within a business scope with shared declarations in order to simplify the policy definition task for larger BAM environment. *Meta-policies* are policies about policies and their relationships. Examples of meta-policies include how policies are deployed and enforced into target VBUs for specific BAM domain.

4.3 Policy grammar

The BAM Policy specifications are defined in grammars based on regular expressions and implemented in an XML based markup language. The following shows a subset of the BAM grammars, where a BAM policy can be divided into three major parts: (1) Agent policies; (2) VBU policies; and (3) Enabler Policies. As described, agents are the *subjects* of the BAM

management scenarios and mandate the policies upon the *objects* of managed resources and business solutions that are abstracted as VBUs.

bam_specification = [import_statement | domain_statement | bam_policies]* ;
bam_policies = ['agent', [agent_definition]* | 'vbu', [vbu_definition] | 'enabler', [enabler_definition]]* ;
agent_definition = ['agent', [port_def*, intent_def+, required_capabilies_def*, required_action_def*]] ;

An agent policy consists of ports, the agent's intents, and the capabilities and management functionality that are expected from targeted virtual business units. An agent serves as the interfaces interacting with BAM managers or dashboards. A port consists of (1) a mandatory *name* attribute, which must be unique within a BAM policy instance; (2) an optional *implementation* attribute, which may reference an implementation specification for the agent, for example, the name of a BPEL process (Business Process Execution Language); (3) an optional *description* element of the agent. The intent specification describes the purpose or motivation of an agent and how this intent can be linked other intents, capabilities, and actions. Three kinds of intents have been identified: *role intent*, *target intent* and *policy intent*. Role intent is about how an agent sees itself as it is engaged with specific set of VBUs. Target intent defines the scope of the VBUs which the agent intends to interact with. Policy intent describes what the agent will govern the interactions and operations of targeted VBUs.

The types of policy intents include commitment intents, violation intents, and delegation intents. Both *required capabilities* and *required actions* describes the ''expectation'' of an agent towards its target VBUs. Required capabilities (required_capability_def) specify the expected data such as metrics, key performance indicators (KPIs) and the schedule of sampling data. Required actions (required_action_def) describe the expected management functionalities that should be provided by target VBUs. Since VBUs serve as an abstraction layer between management agents and managed resources, they are obligated to expose necessary management data and functions to governing management agents. To de-couple the development processes of management agents and VBUs, it is a rational design to allow an agent make assumption about the VBUs it expects to monitor and control. The action definition is conventional model-based specification such as VDM (Sheppard, 1994).

agent_definition = ['agent', [port_def*, intent_def+, provided_capabilies_def*, provided_action_def*]] ;
port_def = ['name', portName, 'implementation', implementation_def, 'description', expression] ;
intent_def = [policy_intent | target_intent | role_intent]*;
policy_intent = [commitment_intent | violation_intent | delegation_intent]* ;
required_capability_def = 'required capability' [metric_def | kpi_def | scheduling_def]* ;
required_action_def = ['required action', actionName, 'preCondition', pre_condition_def, 'body', required_action_body_def, 'postCondition', post_condition_def, 'invariant', invariant_def] ;
required_action_body_def = [monitoring_def | analysis_def | control_def]* ;

VBU specification defines an instrumental view of the resources and business activities that are targeted to be managed and governed. Four main parts of a VBU specification include context definition, constraint definition, provided capabilities, and provided management actions. *Context* describes the environment where the policies are to be applied for the existence and behavior of a virtual business unit. Note that the environment can be physical environment such as inventory warehouse or *virtual* environment such as planning engine for manufacturing processes.

The business context is largely about VBU relationships with other VBUs and underlying resources. In some sense, the context becomes implicit constraints of interactions among VBUs. Constraint definitions categorize the characteristics of targeted VBUs. The constraints can be pre-defined at build time or instantiated at run time. Examples of constraints include inventory thresholds, revenue targets, and maximum delivery time. Constraints set the criteria of quality of services for governed business activities. The available capabilities (provided_capability_def) and management functions (provided_action_def) are defined similarly to those in the agent policy specification.

vbu_definition = ['vbu', [context_def*, constraint_def+, provided_capabilies_def*, provided_action_def*]] ; context_def = [['atomic', boolean] | ['exception', exception_def] |
['process', process_def] | ['schedule', scheduling_def) | ['signals',
signal_def] | ['faults', fault_handlers_def]]* ;
constraint_def = [policy_constraint | authorization_constraint | organization_constraint]*;
policy_constraint =[invariant_constraint | optimization_constraint] ;
provided_capability_def = 'provided capability' [metric_def | kpi_def | scheduling_def]* ;
provided_action_def = ['provided action', actionName, 'preCondition',
pre_condition_def, 'body', provided_action_body_def, 'postCondition',
post_condition_def, 'invariant', invariant_def] ;
provided_action_body_def = [monitoring_def | analysis_def | control_def]*;

The enabler policies provide the governance rules of the interactions between VBUs and agents. The following grammar indicates two kinds of enabler policies: (1) Agent and VBU bindings; and (2) binding constraints. The bindings between agents and VBUs can be realized either at policy decision points or at policy enforcement points.
enabler_definition = ['enabler', [agent_vbu_binding, binding_constraint]*] ;
agent_vbu_binding = ['agent-vbu-binding', [agentName, vbuName, param_binding]] ;
binding_constraint = ['binding-constraint, [applicability_constraint | delegation_constraint | authorization_constrint | relationship_constraint]*] ;
Some sample questions of binding constraints are as follows.

- Applicability: Does the target VBU support all capabilities and actions that expected by the agents?
- Delegation: Which type of agents can manage VBU *X* for the agent *Y*?
- Authorization: Can agent *X* monitor and manage VBU *Y*?
- VBU Relationship: Are all of the capabilities provided by VBU *X* also provided by VBU *Y*?

- Agent Relationship: Do agent X and agent Y cooperate or compete on governing VBU type Z? If it is a competition relationship, any other agent can be the arbitrator if conflicts between agent X and agent Y occur?

Therefore, the enabler policies actually govern the interactions among agents and VBUs. The questions of applicability and relationships are difficult to answer without the assistance of advanced technologies. Some of them can be addressed by distributed artificial intelligence and multi-agent systems (Ferber, 1999).

5. Reference implementation

This section describes a reference implementation based upon the policy framework presented in the above sections. Firstly, the policy architecture will be presented, and an example implementation for SCM manufacturing process will be described.

5.1 Policy architecture

Figure 10 shows the policy physical architecture for BAM Policy Framework. Business processes refer to any business activities that require monitoring and control by BAM systems. In the example of the supply chain domain, certain kind of BAM system is needed to instrument, monitor, analyze and control the behaviour of target manufacturing management processes (business solutions).

All of the management activities are governed by BAM policies. *BAM Model Repository* stores the meta-models of managed business activities including the information such as invocation interfaces and semantics. *BAM Runtime Registry* stores the real-time values of all policy artefacts such as constraints, bindings, and commitments. Practically, the BAM Runtime

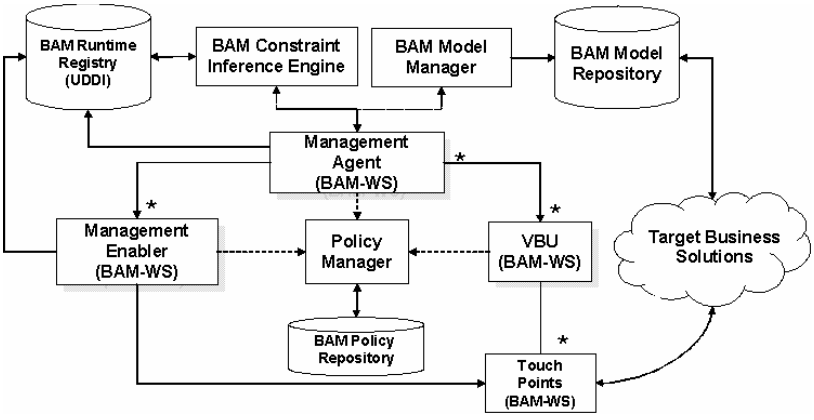

Fig. 10. Policy reference architecture

Registry is implemented as an UDDI registry that is available for posting, updating, and querying desired BAM Web Services.

The *Constraint Inference Engine* helps MAs and policy manager answer the questions that need to be elaborated and reasoned such as those related to relationships. *Touch points* serve as the interfaces between BAM systems and the governed business activities and systems. In our system, they are implemented as Web Services exposing the managed data and functions in the formats of WSDL. *Policy manager* manages the lifecycles of management policies and also provide access mechanism to the *Policy Repository*.

Considering the scenario in the supply chain domain described in Sect. 2, many business activities are required to ensure the functioning of the system. Hence, the BAM system interacts with multiple participating parties such as business line managers who make decisions, building superintendent who conducts do the day-to-day scheduling, production planning team who determine intermediate target outs, manufacturing team who make the final determination of what is built and when the products are to be built. Moreover, a BAM system needs to interact with multiple systems, for example, in our running scenario: (1) Central planning engine (CPE) that helps decision makers determine how to best meet prioritized demand without violating temporal, asset, or capacity constraints. (2) The optimal manufacturing resource planning (OMRP) tool that assists decision makers define detailed instructions about what manufacturing activities must be accomplished and when they must be completed. (3) The available to promise (ATP) tool that enables an organization to dynamically reallocate projected supply in response to incremental changes in the demand statement (new orders arriving, orders being filled, and order changes or cancellations) according to business policy guidelines, identify projected shortfalls with respect to committed orders, and provide real-time order commits and status. (4) The demand management (DM) tool that help decision makers coordinates demand estimates from different sources such as orders, sales rep forecasts, customer forecasts, internal demand, and marketing forecasts in logical step.

In general, the functioning of a manufacturing system really depends on the coordination and management of many autonomous business activities and resources. If a decision maker took arbitrary decisions without regard of any consideration of the other parts of the whole manufacturing process, the whole system will not work properly. Thus, there is a need for a set of management policies that work as the "contract" among business activities and systems by constraining their behaviour so as to guaranteeing the interests of all the stakeholders. These management policies can be viewed as exogenous commitments that a decision maker makes with respect to both the stakeholders; or as endogenous commitments that are exploited to enable and enforce the behaviours of managed business processes and systems, named as the E-Commerce systems in Sect. 2. Hence, a policy-based management is the natural choice of implementing such BAM systems. The policy types in the manufacturing systems can be of many forms, for example, the pre-defined demand boundaries, the inventory level thresholds, system performance, and so on. Abstractly, management commitments define the constraints that would follow certain courses of actions, or to hold certain agreed and trusted situations manifested by the entities in VBUs. As

mentioned, a commitment policy concerns either acting in a certain way, or it can be a commitment to hold a certain expectation.

Commitments can be about the past or the future, where the former are called retrospective commitments and the latter are called prospective commitments. BAM can be a standalone application or be composed of many MAs. In the latter case, each agent may hold its own commitment. A response will be formed in a VBU policy via bindings of required actions (in agent policy) and provided actions (in VBUs). Notably, a response can be executed when the bound constraints are either met or violated depending which way has been defined in the expectation. VBU is implemented as BAM Web Services in this framework.

Some meta-actions can be defined to perform actions on commitment policies themselves. One type of meta-actions is to *commit* a commitment to an agent. This is usually a future commitment, but it can be used for the establishment of a past commitment as well. Another type of meta-action is to put an expectation relationship between agents. For instance, an ATP agent *expects* an inventory agent to have inventory level less than certain quantified value. If the expectation is violated, some actions will be taken based upon corresponding bound *response* that is defined in the ATP agent. A commitment can be applied to several agents, i.e., those agents *share* the same commitment. We can thus define the *locus of control* for the commitment $\boldsymbol{C}$ by defining a set of agents committing to $\boldsymbol{C}$ and the resources that are governed by $\boldsymbol{C}$. It is assumed that an agent will always communicate its commitments truthfully with the other agents in response to queries and actions.

Each MA may have one or more than one commitment policies. MAs commit to one another through the agent-agent bindings. Resource models the entities in VBU's context definitions. The diagram shows some examples of resources such as sales status, balanced score card, order status, demand status, and inventory. In fact, a resource can represent even higher level concept such as manufacturing processes/activities or participating parties. Triggers initiate the process of evaluating whether expectation holds or not. A trigger consists of one or more than one situations that are nothing but logical expressions that can be evaluated by the inference engine embedded in BAM systems. Since MAs can commit to one another, the "commit-to" relationships form a network of commitments with BAM-WSs as nodes and commitment as the edges connecting nodes. For example, the Sales BAM-WS is committed to send an alert to the Demand BAM-WS when the current sales quantities of some selected saleable part numbers in the *n*th week are out of their bands. Note that both MA and ME are implemented as BAM Web Services in the BAM Policy framework.

The Demand BAM-WS is committed to the Portal BAM-WS to provide a list of recommendation on adjusting the planned demand quantities and safety stock requirements for the *n*th week. However, the Demand BAM-WS cannot work by itself. The Demand Planning BAM-WS and the Inventory Planning BAM-WS commit themselves to the Demand BAM-WS such that they can provide the Demand BAM-WS recommended demand quantities and safety stock requirements for the *n*th week. Network of Commitments (NoC) is a generalized notion of Chains of

Commitment (Ervin, 2002). NoC is used to define the triggering points, the control and data flows, the monitoring policies, the situation detection policies, and the actuation policies. Since an agent can *commit* and *uncommit* commitments dynamically, the applications built on BAM become extremely configurable and can be adaptive to the enterprise needs, e.g., Balanced Score Card, by simply modifying the definitions of NoC through the configuration tools. From the points of view of any stakeholder of the microelectronic manufacturing process, commitments themselves provide a means to define the quality of services from the system itself because the whole system is more visible, controllable and configurable. Figure 11 depicts an application of the BAM Policy Framework for the use case scenario described previously where agents are implemented as Web Services indicated as BAM-WS.

Sales BAM-WS detects the "out-of-band" situation and notifies the committed BAM-WS, i.e., Demand BAM-WS that will consequently notifies Recommendation BAM-WS and Risk Assessment BAM-WS in sequence to obtain recommended build plan(s) and necessary assessment such as inventory cost, manufacturing cost and SLA measurement. Note that commitment relationship may imply either event/situation flows or data flows between commitment-related BAM-WSs, and the actions to be taken really depend on the definition of the involved commitments, i.e., on the expectations, actions and responses that are delineated in the committed BAM-WSs. The aforementioned commitments are all endogenous commit-

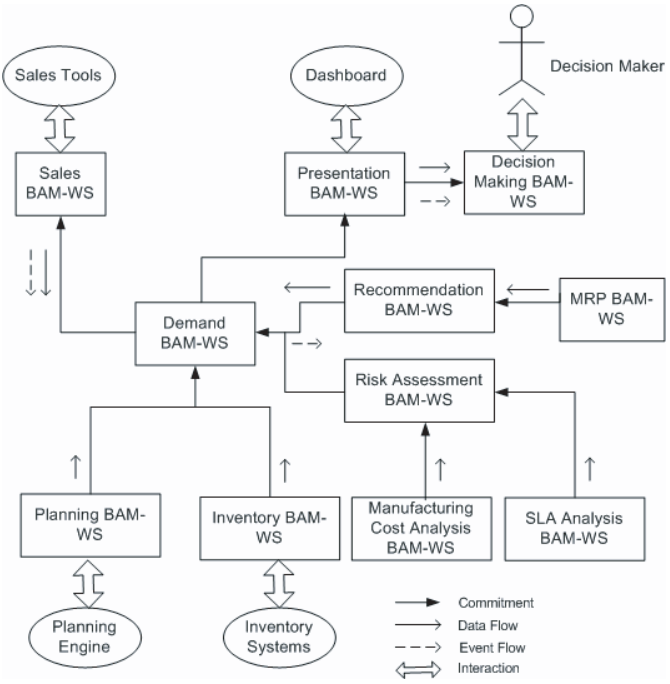

Fig. 11. Network of commitments

ments. However, Portal BAM-WS has an exogenous commitment to the decision maker that represents the ultimate user of the BAM system.

5.2 A BAM implementation for manufacturing process

We have worked with a chip maker and applied BAM Policy Framework to fulfilling the requirements raised in the electronic manufacturing domain. Specifically, we have implemented a ''sense-and-respond'' system in that domain. This system senses events generated in the manufacturing systems, detects manufacturing-related business situations, conducts analysis on the data embedded in the situations and enterprise database, and finally provided recommended actions to decision makers.

Between the dashboard and the management layer, there is a dashboard façade with the following components:

– *Widget BAM Web Services* that are customized for specific domains.
– *User BAM-WSs*: User Web Services are data containers and functional components that are also specialized for specific domain. Examples are charting controller, personal alert controller and event controllers, tag libraries. User Web Services are reusable components that aid in building quick dashboard for each domain.
– *Data BAM-WSs*: Data Web Services are connected to the management Web Services and used to serve the requests from dashboard users. This layer consists of adaptors that connect to BAM artifacts and convert all requests to XML, which is then processed on by the user Web Services and widget Web Services.

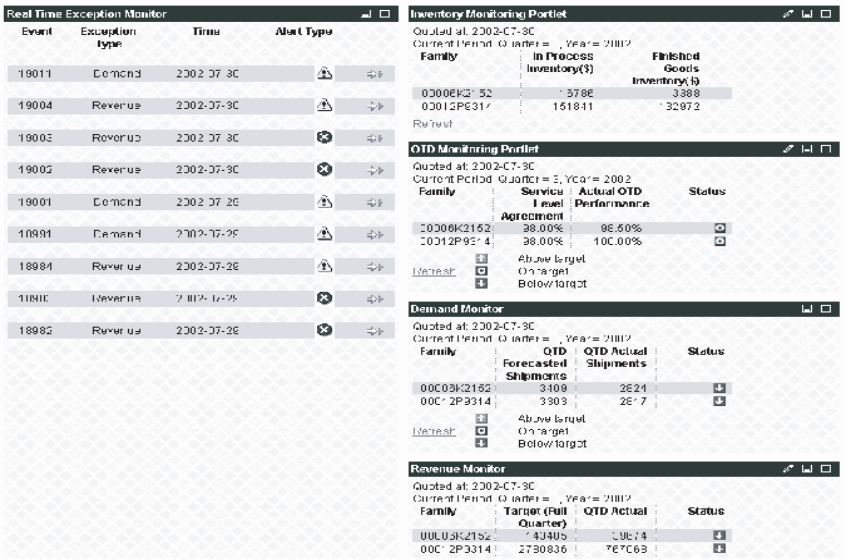

Fig. 12. BAM monitoring console

Management clients can be of in many forms: management console, manufacturing portal, planning system client, OLAP client, and business activity dashboard. The BAM management console in Fig. 12 presents the unified view for the BAM users to monitor all the manufacturing processes and activities, manufacturing exceptions, links to perform OLAP analysis, presents recommended actions to manufacturing exceptions and so on. The right-hand side of this console present four monitoring portals and the real monitoring of manufacturing events are shown in the portal on the left-hand side. Fig. 13 shows the revenue performance of a manufacturing process. The portal on the left-hand side shows the revenue statistics including both actual and predictive performance data. On the right-hand side, a graphical representation of the performance data is shown in a portal where the upper and lower bounds indicate the performance targets. A business situation will be raised whenever the revenue performance data is out of the boundaries. Thereafter, a decision making process will be triggered to resolve such situation. As mentioned, all of these interactions are handled by designated web services based upon predefined BAM policies.

6 Related work

Minsky and Ungureanu (Minsky, 2000) 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. LGI uses decentralized controllers co-located with agents. The concept of policy virtualization in the BAM Policy Framework does not exist in LGI. The policy works in the standards bodies such as IETF (IETF Policy Framework Working Group) and DMTF (Distributed Management Task Force Policy Working Group) are more focused on defining policy frameworks for

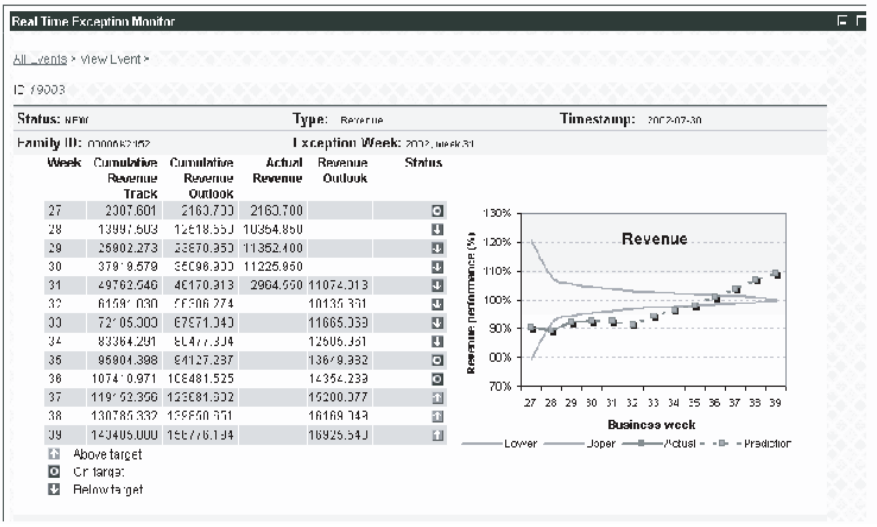

Fig. 13. Monitoring revenue performance

traditional IT systems. Hence, some important issues, e.g., virtualization and constraint inferences, are not as seriously concerned as in the BAM Policy Framework.

Verma (2002) proposes a policy service for resource allocation in the Grid environment. Due to the nature of Grid computing, virtualization has been greatly used for defining policy services in the paper. However, in contrast to their work, the BAM Policy Framework is aimed for providing policy framework for business activities instead of a service for system domain. In general, most of policy works are specific to quality of service management (Cisco Systems Inc., 2000; Hewlett-Packard Company, 2001; IP Highway Ltd, 2001; Orchestream Ltd, 2001; Orchestream Ltd, 2001). The Ponder (Damianou et al. 2001) and Policy Framework for Management of Distributed Systems (Damianou, 2002) address the implementation of managing network systems based on policies. They constitute Network Management Policy Enforcement, which is very different from the domain of BAM Policy Framework.

Policies have been made comparison with business rules. Basically, a business rule is a compact statement about an aspect of a business (Morgan, 2002). It is intended to assert business structure or to control or to influence the behaviour of a business. There are many well-known rules engines that are used to implement business rules. Our position is that the policies and business rules are complementary. While BAM policies can be implemented in business rules, there are many other possible ways of realizing policies. BAM policies can be enabled by other programming paradigms such as object-oriented methods, relational databases, or decision trees. On the other hand, the parameters of business rules and their execution strategy can be well-defined in policies and realized in non-rule software components.

In general, the BAM Policy Framework is complementary with other policy efforts in that it is aimed for both modeling and tuning up the decisional behaviours embedded in BAM related business processes. According to our experience in the area of microelectronic manufacturing, this framework shows some traits, to name a few, as follows (1) The decision strategies hidden in the manufacturing processes are externalized and formally described in policies; (2) Those externalized decisions can be changed through policies without re-implementation of related modules; and (3) Since functional components can be developed without regard of business decisions, greater reusability has been achieved through our approach.

7 Concluding remarks

In this paper, we have presented a policy framework for business activity management. The BAM Policy Framework is based on the requirements gathered from real use cases in supply chain management and other domains. We focus on deriving a framework that can be easily used by business analysts and developers via virtualization and policy utilities provided by the framework itself. We have leveraged the experience gained through earlier attempts at implementing policy architecture for manufacturing systems in the supply chain domain, to provide for the implementation of a general policy-based management platform. We are currently working on building

tools to enhance the usability and capability of the BAM Policy Framework. Using the similar concepts and framework, we are building various architectures for various domains. This actually help us grasp the whole spectrum of BAM domain. We are also working on providing further analysis by simulating the execution of policies for business activities. An integrated environment for animating the simulation and viewing the results will be part of such a task. We are developing formal BAM models based upon UML as the endeavor towards an enabling infrastructure for adaptive BAM systems. We are investigating cloning, versioning, and mobility of BAM systems with an eye towards non-functional requirements such as performance, availability, fail-over, and migration that are critical to the success of BAM system in practice. This infrastructure is being applied to real-world scenarios, including supply chain, logistics, finance, insurance, and life science. We will also present case studies of applying BAM models to developing and generating BAM solutions.

Acknowledgement. We would like thank David Cohn, Kevin McAuliffs and Kumar Bhaskaran for their support on the development of this work. Special thanks also go to the BAM team members including Steve Buckley, Grace Lin, Heng Cao, Pawan Chowdhary, Shubir Kapoor, John Kearney, Haifei Li , Josef Schiefer and Haifeng Xi.

References

Buchmann F, Meunier R, Rohnerr H, Sommerlad P, Stal M (1996) *A System of Patterns: Pattern-Oriented Software Architecture*. New York: Wiley

Business Process Execution Language, available at the URL http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Cisco Systems Inc. (2000) Cisco Assure Policy Manager, Available electronically at: http://www.cisco.com/warp/public/cc/pd/nemnsw/cap/index.shtml

Damianou N, Dulay N, Lupu E, Sloman M (2001) The Ponder Policy Specification Language. *Proceedings of the Policy Workshop 2001*, HP Labs, Bristol, UK, Springer-Verlag http://www.doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf

Damianou N (2002) A Policy Framework for Management of Distributed Systems. PhD Thesis, Faculty of Engineering of the University of London, London, England http://www-dse.doc.ic.ac.uk/Research/policies/ponder/thesis-ncd.pdf

Distributed Management Task Force Policy Working Group, Charter available at URL http://www.dmtf.org/about/working/sla.php.

Ervin R (2002) Chains of Commitment Software Architecture. *ACM SIGecom Exchanges* Special Issues Chains of Commitment, http://www.acm.org/sigs/sigecom/exchanges/-volume_3_(02)/3.1-Ervin.pdf

Ferber J (1999) *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley Pub Co

Fordyne K (2001) New Supply Chain Management Applications Provide Better Customer Service: Serious Gets Exciting. *Journal of IBM Microelectronics* pp 15–19

Gamma E, Helm R, Johnson R, Vlissides J (1995) *Design Pattersn: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series, Addison-Wesley

Haeckel SH, Slywotzky AJ (1999) *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Harvard Business School Publisher, August

Hewlett-Packard Company (2001) PolicyXpert, Available electronically at: http://www.openview.hp.com/products/policyexpert/index.asp

IETF Policy Framework Working Group: Charter available at the URL http://www.ietf.org/html.charters/policy-charter.html

IP Highway Ltd, (2001) Policy Management Console Available electronically at: http://www.iphighway.com/

Jeng JJ, Buckley S, Chang H, Chung JY, Kapoor S, Kearney J, Li H, Schiefer J (2002) *BPSM: An Adaptive Platform for Managing Business Process Solutions*. Proceedings of the Fifth International Conference on Electronic Commerce Research (ICECR-5), Montreal, Canada

Jeng JJ, Li H (2002) Business Commitments for Dynamic E-business Solution Management: Concept and Specification. *Proceedings of 6th World Multi Conference on Systemics, Cybernetics and Informatics (SCI)*, Vol VIII, Concepts and Applications of Systemics, Cybernetics and Informatics II, July 14 - 18, Orlando, Florida USA, pp 403–407

Minsky NH, Ungureanu V(2000) Law-Governed Interaction: A Coordination and Control Mechanism for Heterogenous Distributed Systems. *ACM Transaction on Software Engineering and Methodology*, Vol 9, No 3, pp 273–305

Morgan T (2002) *Business Rules and Information Systems,* Addison-Wesley

Orchestream Ltd (2001) Orchestream Service Activator, Available electronically at: http://www.orchestream.com/

Lin G et al (2002) New Frontier: Sense and Respond System for Global Value Chain Optimization. *OR/MS Today*

Sheppard D (1994) *An Introduction to Formal Specification With Z and VDM*, McGraw Hill Book Co Ltd

Strosnider J et al (2002) *IGS BizADS Handbook*. IBM Redbooks

Supply Chain Operations Reference Model (SCOR), Supply Chain Council, Version 3.1, March, 2001

Verma D (2000) *Policy Enabled Networking*. New Riders Publications

Verma D (2002) *A Policy Service for Grid Computing*. In: Parashar M (ed.) GRID 2002, LNCS 2536, pp 243–255

Wu LSY, Hosking JRM, Doll JM (2002) Business Planning Under Uncertainty: Will We Attain Our Goal? *IBM Research Report*, RC 16120 (71660)