An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises

Jun-Jang Jeng , Josef Schiefer, and Henry Chang

IBM T.J. Watson Research Center

{jjjeng,josef.schiefer,hychang}@us.ibm.com

Abstract

As the desire for business intelligence capabilities for e-business processes expands, existing workflow management systems and decision support systems are not able to provide continuous, real-time analytics for decision makers. Business intelligence requirements may appear to be different across the various industries, but the underlying requirements are similar – information that is integrated, current, detailed, and immediately accessible. In this paper we introduce an agent-based architecture that supports a complete business intelligence process to sense, interpret, predict, automate and respond to business processes and aims to decrease the time it takes to make the business decisions. In fact, there should be almost zero-latency between the cause and effect of a business decision. Our architecture enables analysis across corporate business processes notifies the business of actionable recommendations or automatically triggers business operations, effectively closing the gap between business intelligence systems and business processes.

1. Introduction

The emergence of e-business has dramatically changed the context in which decision-making takes place. While the fundamental human and organizational processes that take place remain largely unaffected, e-business places new constraints and demands on the decision maker to provide better service to the customers. Because of the increased rate of change possible in e-business, decisions must be made more quickly than in the past. Process participants must have instant access to information which is relevant for the current business context. All these factors imply that the traditional decision support solutions that are focused simply on the provision of information and analysis tools are no longer sufficient. Traditional data warehouses, report generators, OLAP and data mining tools typically do not allow a monitoring of business activities on a continuous real-time basis. To be effective, decision support must take a broader view of the whole process of decision-making that is embedded in business processes. One of the key weaknesses of the current generation of workflow management systems and decision support systems is their lack of integration.

Decision makers typically use exception-based analysis on published metrics to identify opportunities, then dig deeper into the data to understand the causes of those opportunities. From there, they model the business situation so that they have a framework against which to evaluate different decision alternatives. After selecting an alternative the user acts accordingly to the decision made.

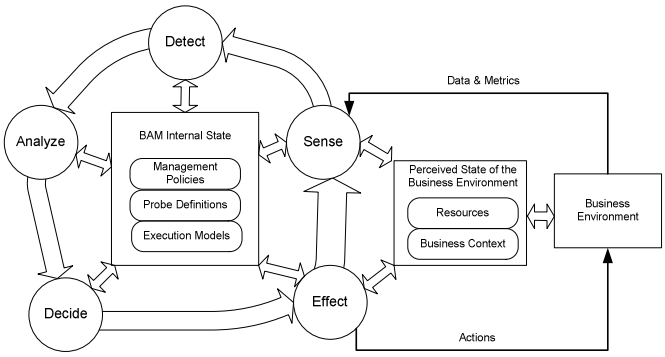

Figure 1. BAM Decision Cycle

Figure 1 shows such a decision cycle. The BAM (Business Activity Management) decision cycle involves 5 sub processes for the decision making: *sense*, *detect*, *analyze*, *decide*, and *effect*. The *sense* process monitors and collects desired data (based upon the intents) from the business environment. The output from the *sense* process is in the form of metrics or key performance indicators, which provide a virtualized environment for the other processes: detection, analysis and decisions. After integrating events from various source systems, a BAM system will start the *detect* process based on the business intents and constraints. The detection phase usually generates new business situations that are further explored by an *analyze* process. The *analyze* process helps to predict the performance and assess the risks of the available options for responding to the business environment. The *analyze* processes facilitate determining

the root causes of the identified business situations. Key for the determination of causal factors is the ability to identify inherent relationships and dependencies between variables that drive the situational or exceptional performance. The outputs of the *analyze* process are alternatives to improve the current business situation and the guidance for the decision makers to select the best alternative. The *decide* process selects the best option and also determines the most appropriate action for a response to the business environment. The *effect* process executes appropriate business actions based on the decision that has been made. This response will either change the state of the business environment or notify other agents (humans or programs) who may be interested in the outcome and result of the decision making.

The *sense* and *effect* processes need to interact with the target business environment in order to obtain data/metrics and to trigger actions, respectively. Hence, a BAM system needs to have the knowledge of the interfaces and protocols of interacting with the target business environment. A BAM system manages and enforces the policies that are configured and deployed by business experts who know the strategies and rules for managing the business activities and the underlying systems. A BAM system also needs to retain the information of the target environment, it is interacting with. Examples of such information include the status of the environment, the business context models, and resource models.

In this paper, we propose a framework for business activity management (BAM), which supports the decision cycle shown in Figure **1** and allows these steps to be accomplished in near real-time.

The BAM framework aims to:

- Provide decision makers and process analysts comprehensive information about the status and performance of business processes independent from the type of systems that are used to execute or support the business process.
- Help users to proactively identify situations and exceptions by analyzing the current process business context to focus on the opportunities offering the best business return and those deserving the most attention.
- 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 between variables that drive the situational or exceptional performance.
- Generate alternatives to improve the current business situation and help the decision-maker to select the best alternative.
- Triggering the appropriate business actions based on the decision made. This response can change the state of the business process or notify parties who may be interested in the outcome and result of the decision making.

The remainder of this paper is organized as follows. In section 2, we discuss the contribution of this paper and related work. In section 3, we present our agent-based BAM architecture that supports real-time analytics. Sections 4 – 7 describe the components of our proposed BAM architecture. These sections include a detailed description of our agent framework for the analytical processing and also the introduction of a container-based approach for the event processing that enables a near real-time event data integration. In section 8, we present a supply chain use case for our proposed architecture. Finally, in section 9 we present our conclusion and discuss our future work.

2. Contribution and related work

Although monitoring and analysis are considered as important tasks of the workflow management system (e.g.[9]), and the Workflow Management Coalition has already drafted a standard for workflow logs [17], little work has been done in integrating and analyzing the workflow audit trail information.

Some approaches emphasize the need for integrating audit trail into data warehouse systems (e.g. the process data warehouse in [15]), others are limited to a smaller set of workflow history that is managed within a workflow management system. To our knowledge there has been no work that thoroughly discusses an end-to-end solution for propagating, transforming and analyzing large amounts of workflow events in near real-time.

Sayal et al. present in [15] a set of integrated tools that support business and IT users in managing process execution quality. These tools are able to understand and process the workflow audit trail from HP Process Manager (HPPM), and can load via a loader component into the process data warehouse. Sayal et al. provide a high-level architecture and a data model for the process data warehouse, but they do not address the problem of integrating and analyzing the workflow audit trail in near real-time. An approach for history management of audit trail data from a distributed workflow system is also discussed in [13]. The paper describes the structure of the history objects determined according to the nature of the data and the processing needs, and the possible query processing strategies on these objects. These strategies show how to write queries for retrieving audit trail information. Unlike our approach, neither the

transformation and aggregation of audit trail data, nor the analytical processing of this data are considered.

Geppert and Tombros introduce in [5] an approach for the logging and post-mortem analysis of workflow executions that uses activate database technology. The post-mortem analysis is accomplished through querying the event history which is stored in an active database system which supports Event-Condition-Action (ECA) rules. Various types of events (e.g., database transitions, time events, and external signals) can trigger in the event history the evaluation of a condition and if the condition evaluates to true, the action is executed.

ADEPT (Advanced Decision Environment for Process Tasks) is an approach which uses an agent-based infrastructure for managing business processes [11]. The agents include modules for routing messages between the agent and its agency and between peer agents, for provisioning services through negotiation, and for assessing and monitoring the agent's ability to meet service levels. The key advance of the ADEPT system is that the responsibility for enacting various components of the business process is delegated to a number of autonomous problem solving agents. These agents typically interact and negotiate with other agents in order to coordinate their actions. The Agent Enhanced Workflow (AEW) system [12] uses an agent system to overcome inability of many WFMS that are monolithic in structure and cannot be used in a distributed environment. In AEW, a community of intelligent, distributed, and autonomous software agents is used to improve the management of business process under the control of a workflow management system. These improvements are achieved by allowing the software agents to negotiate with each other to establish contracts that govern the distribution of work across a number of processing centers. We also use an agent-infrastructure for performing the analytics for a business process. Distinguishing aspects of our work, not emphasized in other works, is the support of real-time processing capabilities and the introduction of management layers for the business intelligence agents.

Most of the existing WFMSs and BAMs offer only very basic monitoring and analysis capabilities, such as the retrieval status information about process instances or summary information about cycle times. Commercial systems (e.g. Staffware) usually provide rudimentary logging information in the form of an audit trail which is dumped to a file, where as others such as MQ Series Workflow store log records in a relational database. For a more comprehensive analysis, users have to use reporting tools from third-party vendors and write queries to retrieve data of interest. While this approach does provide basic reporting, it requires considerable configuration effort and assumes the existence of comprehensive knowledge of the process analysts to write correct queries.

In order to overcome these limitations, we introduce in this paper an architecture which addresses the following issues:

1. We use separate tiers for the data integration and the analytical processing in order to distribute the processing.
2. For the integration of business process events we use a container-based approach which allows to transform the incoming process events into business metrics in near real-time,
3. For the analytical processing of process audit data we use software agents,
4. We use a policy-driven approach for evaluating business process metrics,
5. The architecture allows a straight-through processing of process events that stream into the system which enables the BAM system to perform the analytical processing with minimal latency and to respond to the business environment in near real-time.

3. An architectural framework for real-time Business Activity Management

Traditional data warehouses, business report generators, OLAP and data mining solutions typically do not monitor business processes on a continuous real-time basis. These solutions are not designed for continuously integrating data from various operational sources and are not optimal for minimizing the average latency from when a fact is first captured in an electronic format somewhere within an organization until it is available for the knowledge worker who needs it.

Real-time analytics enables organizations to better monitor the health of critical business processes and operations, providing mechanisms to instantly respond to business problems or to notify the business of actionable recommendations, effectively closing the gap between business intelligence systems and business processes. Figure 2 shows an agent-based architecture for real-time analytics providing real-time access to critical business performance indicators to improve the speed and effectiveness of business operations.

The architecture includes 5 major components: 1) Business Intelligence agents for the analytical processing, 2) the event processing container (EPC) for the real-time transformation of process events, 3) a Process Information Factory for storing business process metrics, 4) a policy management system, and 5) a dashboard for the visualization of business process metrics and analytical

results. In the following sections, we will discuss these architectural components in detail.

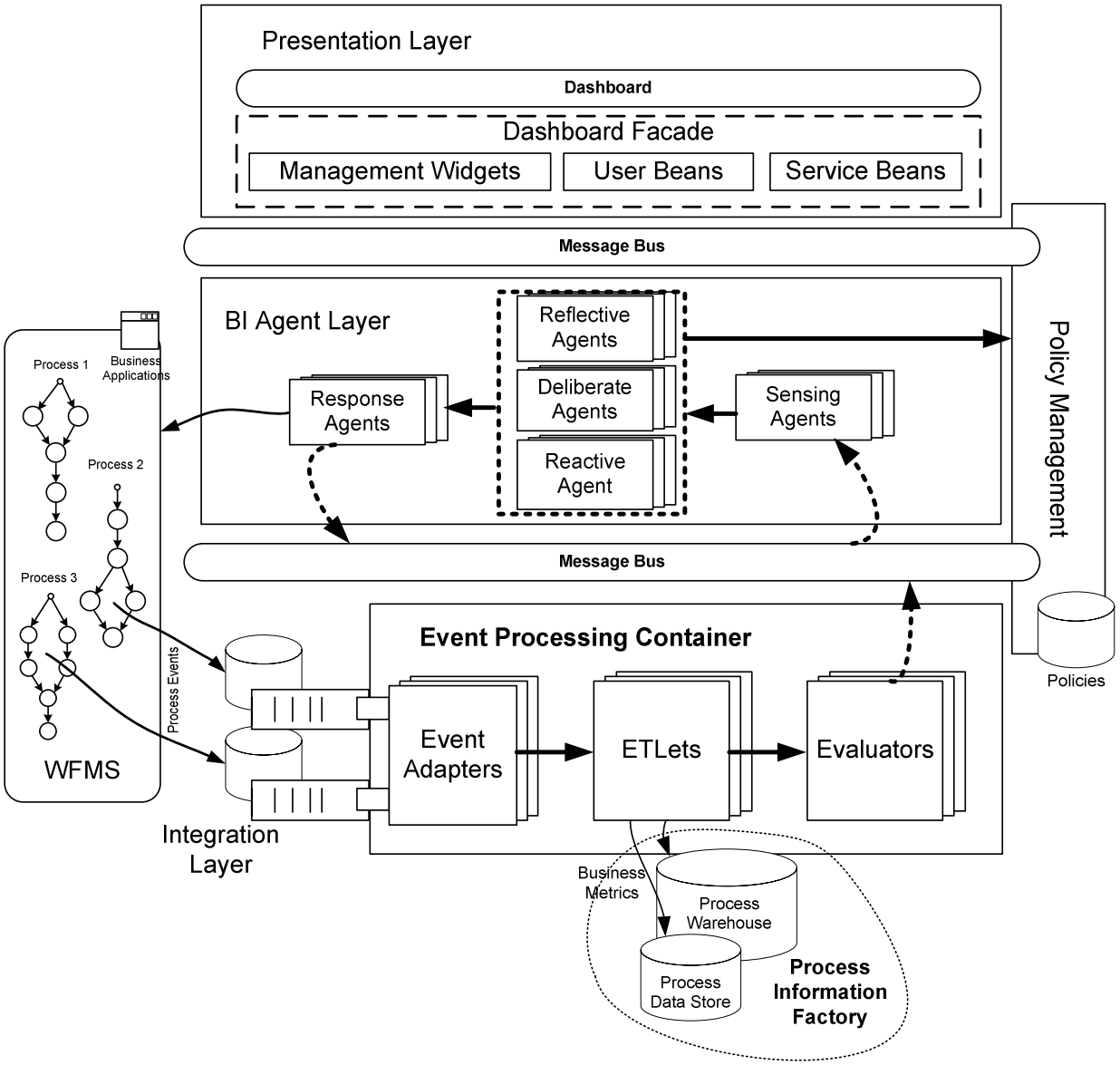

Figure 2. BAM architectural framework

4. Business intelligence agents for analytical processing

Business Intelligence (BI) agents are able to perceive the situations occurring in the business process execution environment (e.g. WFMSs) and respond in a timely fashion to changes. BI agents are able to exhibit policy-governed behavior by following pre-defined rules or taking the initiatives in order to satisfy the imposed management goals. The approach of policy-based management in this framework will be discussed in later section. BI agents are capable of interacting with other

agents and humans in order to satisfy the management goals. Hence, BI agents are self-aware objects that are able to reflect on the gap between current situations and desired management goals, and to change their own management behavior accordingly. In summary, the BI agents aim to fulfill three areas of management functionality: reactivity, deliberation and reflectivity.

Given the requirements that BI agents can be capable of reactive, deliberative, and proactive behavior, an obvious decomposition involves creating separate subsystems to deal with different types of management behaviors. (1) *Reactive management layer* responds to situations and exceptions in a business environment. The response mechanism is driven by a set of situation-action rules, like the behavior in Brook's subsumption architecture [3]. (2) *Deliberate management layer* performs managerial tasks that require more reasoning and more complicated computation. It is not uncommon that BAM needs to provide decision support capability so that more intelligent management directives can be derived towards managed resources [7][14]. (3) *Reflective management layer* enables BAM to maintain information about itself and use this information to remain extensible and adaptable [4]. Reflective management layer performs meta-management directives unto the lower management layers and managed entities. through reflective management mechanism, BAM achieves the goals of both 2nd order management and autonomic computing [8].

The agent architecture is illustrated in Figure 3. As the figure shows, the BI agent layer consists of three management layers. Each layer continually produces *business situations* for what actions the agents should perform. The *sensing subsystem* monitors and captures the situations produced in the environment, i.e., the workflow management systems, and other BAM component such as dashboard, and EPC. The *response subsystem* generates action outputs unto the business environment by following the directives delivered from the agent layers. Note that each layer is connected to the sensing inputs and response outputs. In effect, each layer acts like an agent or a group of agents, producing actions as to what behavior to manifest. The sensing and response functions are governed by management policies, which will be described shortly. The governance is enforced by the policy management subsystem. While the direct connection between agent layers and sensing/response subsystems implies simplicity, the coherence of the whole system may not be preserved since agents in different layers can compete on the performing tasks and obtaining resources. To reduce such risk, a *mediation subsystem* is introduced into the agent layer. This subsystem makes decision about which layer has the control of the whole system at any given time. In BAM, the mediation subsystem is implemented in an asynchronous event bus, where the control 'token' is carried by event bus from one layer to another based upon predefined pub/sub policies.

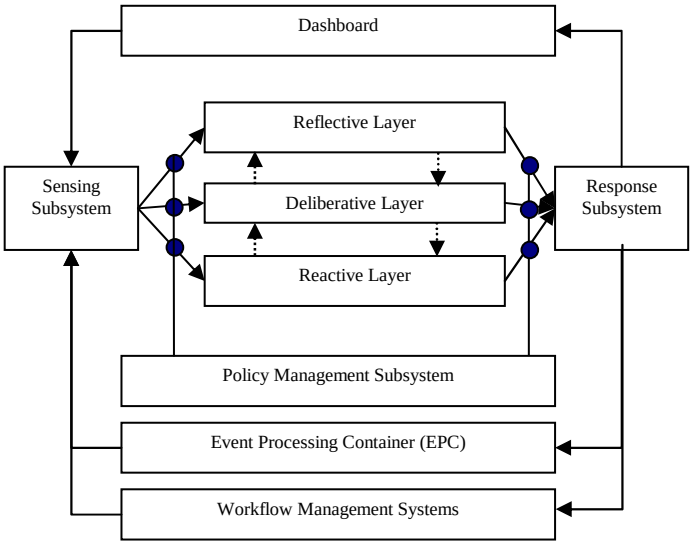

Figure 3. BAM layered agent architecture

Hence, the coordination among agents is centralized at design time and distributed at run time. One of the consequences from a design point of view is that the designer needs to consider all possible interaction patterns among agents. Fortunately, BAM is flexible in regard to the connection between agent layers and other subsystems. If desired, the reactive layer can serve the single point of sensing situations that are generated from the environment. The dotted lines in the figure depict the situation and control flows among agent layers. The advantages of vertical decomposition are its simplicity and the similarity between this idea and how organizations work, with information flowing up to the higher management levels of the organization, and directives then flowing down. The disadvantage is that each layer can be a bottleneck of the whole system. Failures in any layer are likely to have severe consequences to the agent performance.

Agent implementation

The agent layer aims to provide an adaptive platform for realizing BAM functionality. It is dynamic in the sense that management applications can be built to recognize the addition and removal of management components and resources without the system administrator's explicit intervention. The BAM infrastructure can be logically categorized into several tiers: managed resources, management probes, management beans, management commitments, BI agents, and management adaptors.

- *Managed Resources & Management Probes.* In BAM, managed resources within the environment can

be any artifacts to be monitored, measured, configured and controlled. Managed resources are often managed through control points, a set of management APIs that acquire or change the behavior of the managed resources. The states of managed resources can be captured either by polling from management beans or events emitted by managed resources.

- *Management Beans*. BAM uses the industrial standard of manageability to instrument managed resources and manufacturing processes [2][16]. Management standards provide homogeneous interfaces of touch points to managed resources, esp. legacy business systems. Management beans expose the manageable properties of the underlying managed resources to the privileged managing agents. At run time, the managed resources will be connected to some management bean so that management data and functions can be delegated between BI agents and managed resources.
- *BI Agents.* The tier of BI agents consists of agents and utilities that can be assembled, composed and committed to provide management functions. Each agent addresses specific needs and problems. BI agents obtain (by pulling or pushing) data from the tier of management beans and act on that data based upon management commitments. The composition of services and components into a purposeful set of functions contained in BI agents are enabled by the configuration agent.
- *Management Adaptors.* Management adaptors expose the management services of BI agents to external clients. In BAM, management connectors adhere to industrial standards such as J2EE/JMX [16] and Web services [6]. The design of management beans, BI agents, and managed resources does not depend in any way on the protocol an agent uses for communicating with external applications. The provided management adaptors rely on the standard APIs and do not expose any communication details. Web services provide a means for different parties to connect their BI agents with one another to conduct dynamic management services across a network, no matter what their application, design or run-time environment.
- *Management Commitments.* BAM embraces the style of commitment-based management in that BAM exploits management commitments as the vehicle to drive management scenarios on managed resources and manufacturing processes. The management commitments involved in BAM can be multifarious, 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 the BAM substrates, also called *expectations*. A commitment 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. A commitment consists of the following entities:
  - *Actions* that it will perform and resources (data) required;
  - *Resources* that are governed by the commitment;
  - *Expectations* that the commitment hold to and each expectation is composed of *situations*;
  - *Reponses* that bind actions with expectations;
  - *Triggers* that will initiate the evaluation of expectations.

  A commitment consists of triggers, resources, actions, expectations, and responses. An XML-based language, called Management Commitment Language (MCL), has been developed for describing the management policies that govern the behavior of BAM components [10].

5. Event processing container

The event processing container (EPC) provides a robust, scalable environment for integrating workflow events and covers all steps that are required to transform the these events into valuable business information. The container approach allows the handling of a large number of events that can require a complex processing logic.

Similar to Java technology for web applications, where servlets and JSPs took the place of traditional CGI scripts, our approach uses Event Adapter, ETLet, and Evaluator (see Figure 4) components that replace traditional ETL (Extraction, Transformation, Loading) solutions which very often use scripts that are hard to maintain, scale, and reuse.

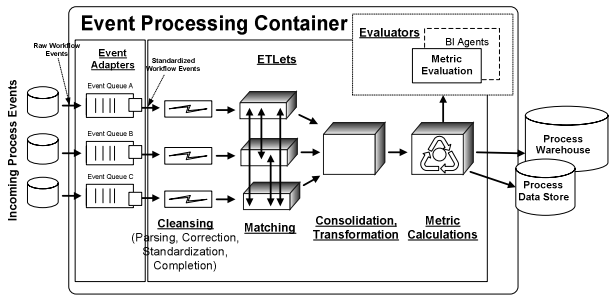

Figure 4. Event processing container (EPC) – architecture

ETL scripts are not suitable for an event-driven environment where data extracts and data transformations are very small and frequent, because the overhead for starting the processes and combining the processing steps can dominate the execution time. Another limitation of ETL scripts is that they are written for a specific task in a self-sustaining manner, and don't provide any kind of interfaces for data inputs and outputs. Because of this constraint in the traditional approach, we use a container to manage and optimize the event processing.

The EPC handles each workflow event with a lightweight Java thread, rather than a heavyweight operating system process. Figure 5 shows the internal processing of workflow events. The components shown with round boxes are components that are managed by the EPC. The components shown with square boxes are internal EPC components that are used to bind all managed EPC components together. Please note that the developers never see or have to deal with the internal components. We show these internal components for illustration purposes only.

This approach also simplifies the programming tasks for developers who have to implement the logic for the event processing, since the EPC takes responsibility for various system-level services (such as threading, resource management, transactions, caching, persistence, and so on). In our approach, we extend this concept by adding new container services, which are useful for the event processing and can be leveraged by the developers. An example of a container service is the evaluation service, which significantly reduces the effort for evaluating calculated process metrics. This arrangement leaves the developer with a simplified development task and allows the implementation details of the system and container services to be reconfigured without changing the components.

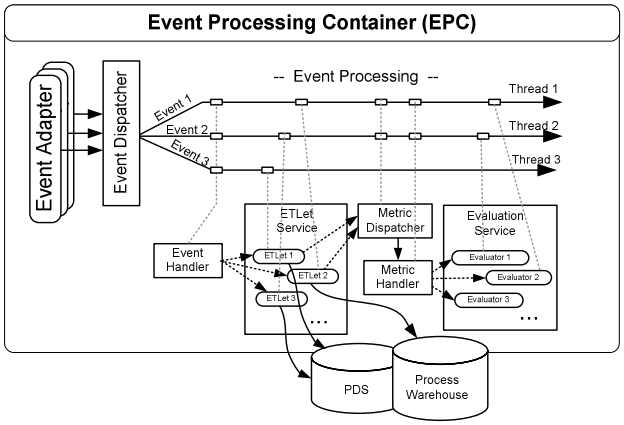

Figure 5. Multithreading within the event processing container (EPC)

Figure 5 also shows the core components (shown as round boxes) that are managed by the EPC: 1) Event Adapters, 2) ETLets, and 3) Evaluators. Each of these components must implement a certain interface that is used by the EPC in order to manage the component's lifecycle. The EPC automatically instantiates these components and calls the interface methods during the components' lifetime. Furthermore, each component has its own deployment descriptor for the configuration parameters. The EPC controls and monitors the event processing by optimizing these configuration settings.

6. Process information factory

The main purpose of the process information factory is to provide a data foundation for a process-driven decision support system to monitor and improve the business process continuously. It is a global process information repository, which enables BI agents and process analysts to access comprehensive information on business processes very quickly, at different aggregation levels, from different and multidimensional points of view, over a long period of time, using a huge historic data basis prepared for analyzing purposes to effectively support the management of business processes. Figure 6 shows the process information factory as part of an existing data warehouse environment. The arrows in Figure 6 indicate a flow of data or control among these components. Red components and arrows highlight the extensions to a conventional (passive) data warehouse environment. The event processing container (EPC) ultimately transforms on-the-fly workflow events into metrics that are stored in the process data store or process warehouse. Furthermore, the EPC also publishes information to the BI Agent Layer for the analytical processing.

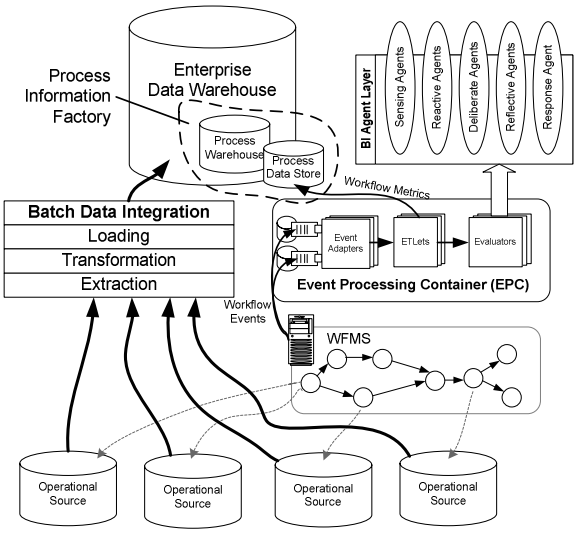

Figure 6. Process information factory as part of an existing data warehouse environment

The process information factory consist of two repositories: 1) the process warehouse which is part of the enterprise data warehouse system and which is used for storing a rich set of historical process data for the strategic decision support and 2) the process data store, which includes very detailed up-to-date process data of current running processes and also allows real-time access for the business intelligence agents. Note that process warehouse and process data store are conceptually equivalent to traditional data warehouses and operational data stores (ODSs), respectively, with the only difference being that they are used to store process- oriented and workflow data. Thereby, the process information factory adds a process perspective to an analytical environment.

7. Policy management system

The goal of the policy management system is to monitor and track all the management agents that are running within BAM. The idea is to have management agents serve as sensors that monitor and deliver events to policy management agents that are particularly interested in the events related to policies. BAM situations are captured by a policy management system and are undertaken through evaluation. In most cases, policies are modeled as situation-action pairs. As some situation occurs, corresponding actions will be triggered. The second functionality of a policy management system is to maintain the management polices deployed to management agents and other entities in BAM. For this perspective, the policy management system will be interested in such events as policy expiration and policy update. The corresponding actions can be to re-deploy management policies to management agents due to the fact that the policies are changed. The third functionality of the policy management system is to provide policy details to interested agents via policy agents. A policy agent is an agent that has access to the policy management systems, and is able to collate and manipulate the policy data obtained from policy management systems in order to answer queries posed by users or management agents.

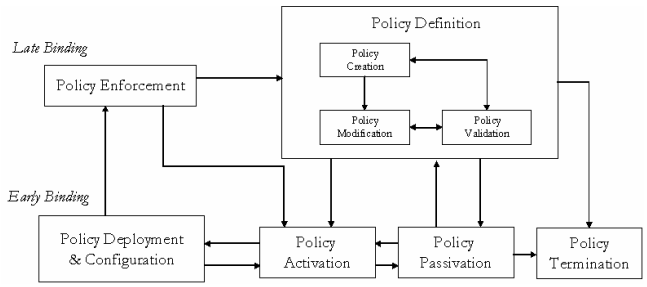

Figure 7. Policy lifecycle

The policy lifecycle for BAM consists of six basic life-stages as shown in Figure 7. The basic stages are: policy definition, policy activation, policy passivation, policy deployment and configuration, policy enforcement and policy termination.

1. *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 active in the BAM system.
2. *Policy deployment* and configuration deploys a policy into the policy target and configures the system correspondingly. A set of automated configuration utilities will simplify the tasks to be performed in this phase, e.g., deployment scripts and configuration management tools.
3. *Policy enforcement* is the stage when a policy is being used to govern and constrain the behavior of target BAM systems. Monitoring and reporting tools will make policy makers to understand how the status of policy enforcement and whether the policy has been defined reasonably.
4. *Policy activation* is the phase when a policy is loaded into BAM system waiting for further execution such as deployment and enforcement. In this phase, policies are active in the memory but have not been committed to any activities yet.
5. *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.

6. *Policy termination* is the phase when a policy ceases to exist in the system.

Potentially, a policy can be bound to BAM at two points of its lifecycle: (1) *policy deployment & configuration:* it is called early binding between policy and mechanism (BAM) since it is realized at the build time; and (2) *policy enforcement*: it is called late binding between policy and mechanism (BAM) since this binding is realized at the run time of policy targets. 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 transitions back to the *policy activation* phase. As mentioned above, a business analyst can use management 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 the high-level policy architecture that is to be used to define detailed policy components and services in later parts of this paper. The policy architecture for BAM is built upon the policy frameworks defined by both IETF [19] and DMTF [20]. BAM Policy Framework consists of seven basic elements as shown in Figure 8.

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 a business analyst to input the different BAM policies that are to be active in BAM systems. The locations that can apply and execute the different policies are known 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 store the policies generated by the management tools. The *policy decision point* is responsible for interpreting the policies stored in the repository and communicating them to the policy enforcement point.

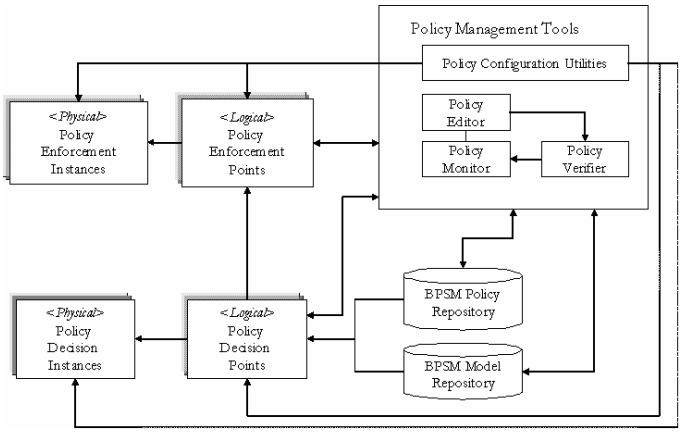

Figure 8. Policy architecture

8. Use case: supply chain management for microelectronics manufacturing

In this section we give an example of a supply chain management (SCM) system for microelectronic manufacturing that uses the proposed BAM framework for monitoring the supply chain operations. SCM 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 time window for opportunities, and the level of precision required in delivering supply chain performance information.

1. The first decision tier, strategic scheduling, is driven by the time frame or lead time required for the business plan, the resource acquisition, and new product introductions. In this tier, decision makers are concerned with a set of problems that are three months to several years in the future.

2. The second tier, tactical scheduling, deals with problems the enterprise encounters in a week to three months time period. 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 products are set.

3. The third tier, operational scheduling, deals with the execution and achievement of plans for the current week such as shipments or measured serviceability levels. Tools typically used for supporting daily activities are for material resource planning or scheduling of production runs.

4. The fourth tier, real-time response, addresses the problems of the next minute to a few hours 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. Usually, analytics modules are used to generate responses based on commitments, business policies, and business rules. A real-time response could be triggered due to a significant drop of revenue caused by the cancellation of a large order.

In the following we describe a typical use case for continuous demand-driven build plan and inventory optimization in the domain of microelectronic manufacturing. 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 (accumulated) revenue and revenue outlook (for remaining weeks in the current quarter) (s)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 [18].

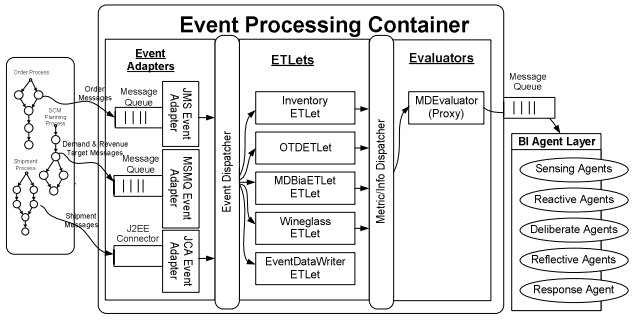

Figure 9. Event processing container for processing supply chain events

Figure 9 shows the EPC processing supply chain events that are needed for the calculation of supply chain metrics. The EPC includes various Event Adapters for receiving messages from various business process (e.g. order process, shipment process, planning process). The Event Adapters unify these incoming supply chain events into a standardized event format that will be used by the ETLets to calculate the key performance indicators.

The EPC shown in Figure 9 calculates metrics about inventory levels (Inventory ETLet), on-time delivery (OTD ETLet), demand forecast (MDBia ETLet), and revenue (Wineglass ETLet). The EventDataWriter ETLet is used to store all incoming event data in the database. All other ETLets focus on calculating the supply chain metrics which are also stored in the database. In the illustrated scenario, we use only agents for evaluating the metrics and detecting business situations. Therefore, we use an evaluator component as proxy for forwarding the calculated metrics to the BI agents.

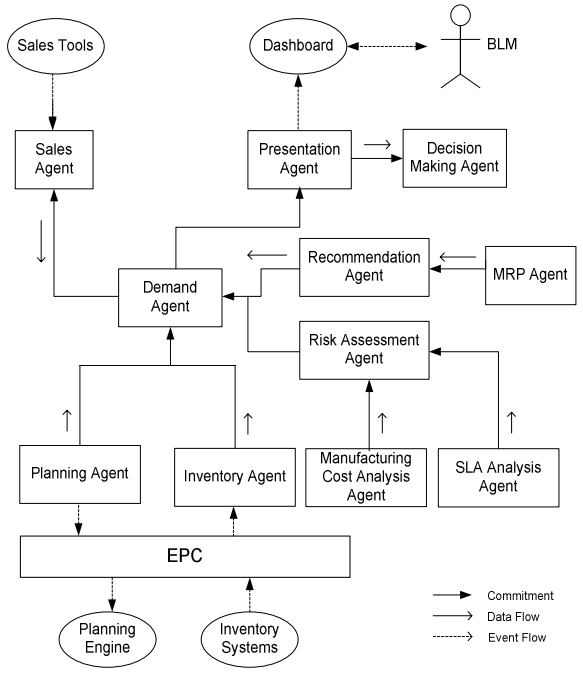

Figure 10. An example of a network of commitments

Based upon the aforementioned scenario, Figure 10 depicts a network of commitments to govern the relationships among agents in BAM. The Sales Agent detects the exceptional situations and notifies its committed agent, i.e., Demand Agent, which will consequently notify Recommendation Agent and Risk Assessment Agent 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 agents, 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 Agents.

The BAM management portal in Figure 11 presents the unified view for the BAM user for monitoring all the manufacturing processes and activities, manufacturing exceptions, links to perform OLAP analysis, presents recommended actions to manufacturing exceptions etc.

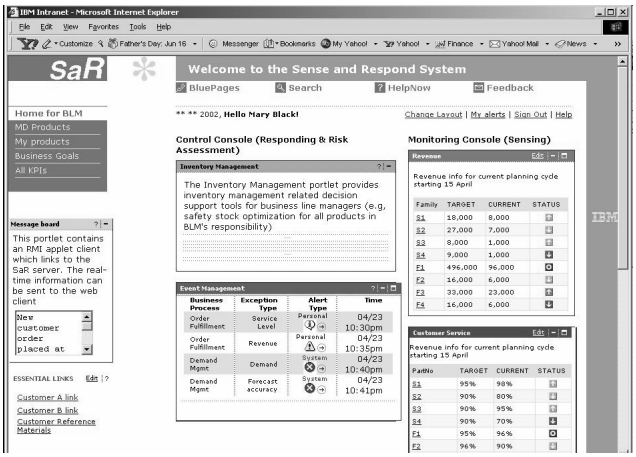

Figure 11. BAM management portal

Management clients can be in many forms - management console, manufacturing portal, planning system client**,** OLAP client, business process dashboard and so on. Between the dashboard and the agent layer (see Figure 2), there is a dashboard façade with the following components: (1) *Management Widgets* that are customized for specific domains; (2) *User Bean Layer*: User beans 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. Use beans are reusable components that aid in building quick dashboards for each domain; and (3) *Service Bean Layer*: Service beans are connected to the agent layers and used to serve the requests from dashboard users. This layer consists of adaptors that connect to BAM components and convert all requests to XML, which is then processed on by the user beans and management widgets.

9. Conclusion and Future Work

In large organizations, huge amounts of data are generated and consumed by business processes. Business managers need up-to-date information to make timely and sound business decisions. Conventional workflow management systems and decision support systems do not provide the low latencies needed for the decision making in e-business environments. This paper described an agent-based architecture with the aim of providing continuous, real-time analytics for business processes. For the analytical processing we introduced an agent framework that is able to detect situations and exceptions in a business environment, perform complex analytical tasks and reflect on the gap between current situations and desired management goals. We introduced the concept of the event processing container which provides a robust, scalable and high-performance event processing environment and which is able to handle a large number of workflow events in near real-time.

The work presented in this paper is part of a larger, long-term research effort aiming to develop a Business Process Intelligence platform for WFMSs. We are building a distributed environment for EPCs that allows them to work together in a server farm. We are also developing an evaluation framework that allows existing rule engines to be plugged into our architecture.

References

[1] Bouzeghoub, M., Fabret, E., Matulovic-Broque, M. Modeling the Data Warehouse Refreshment Process as a Workflow Application, DMDW 99, Heidelberg, Germany, 1999.

[2] Bumpus, W., Sweitzer, J.W., Thompson, P., Westerinen, A.R., Williams, R.C., Common Information Model: Implementing the Object Model for Enterprise Management, John Wiley & Sons, Dec. 1999.

[3] Brooks, R.A., A robust control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1), 14-23, 1986.

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

[5] Geppert, A. and Tombros, D., Logging and Post-Mortem Analysis of Workflow Executions based on Event Histories. Proc. 3rd Intl. Conf. on Rules in Database Systems (RIDS), LNCS 1312, Springer Verlag, Heidelberg, Germany, pages 67-82, 1997.

[6] Graham, S. et al, Building Web Services with Java, SAMS Publishing Co., 2002.

[7] Haeckel, S.H., and Slywotzky, A.J. Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations, Harvard Business School Publisher, August, 1999.

[8] Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM Research External Web Site, http://www.research.ibm.com/autonomic, 2002.

[9] Jablonski, S., Bussler, C., Workflow Management. Modeling Concepts, Architecture, and Implementation. Intl. Thomson Computer Press, London, 1996.

[10] Jeng, J.J., Buckley, S., Chang, H., Chung J.Y., Kapoor, S., Kearney, J., Li, H., and Schiefer J.,

BAM: An Adaptive Platform for Managing Business Process Solutions. Fifth International Conference on Electronic Commerce Research (ICECR-5), Montreal, Canada, 2002.

[11] Jennings, N. R., Faratin, P., Johnson, M. J., O'Brien, P. and Wiegand, M. E. Using intelligent agents to manage business processes, in Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), pp. 345-360. London, UK, 1996.

[12] Judge, D.W., Odgers B.R., Shepherdson J.W., Cui Z., Agent Enhanced Workflow, BT Technical Journal, 16:3, pp 79-85, 1998.

[13] Koksal, P., Alpinar, S. N., Dogac, A. Workflow History Management, ACM Sigmod Record 27(1): 67-75, 1998.

[14] 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.

[15] Sayal, M., Casati, F., Dayal, U., Shan M., Business Process Cockpit, VLDB 2002, Peking, 2002.

[16] Sun Microsystems Inc. Java Management Extensions Instrumentation and Agent Specification (JSR77), July, 2000.

[17] Workflow Management Coalition Audit Data Specification, Document Number WFMC-TC-1015, 1998.

[18] Wu, L. S.-Y., Hosking, J.R.M., Doll, J., Business planning under uncertainty: Will we attain our goal?, International J. of Forecasting, Vol. 8, 545-557, 1992.

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

[20] Snir, Y, Ramberg, Y., Strassner, J., Cohen, R, Moore, B., Policy QoS Information Model, IETF Internet Draft, November, 2001.