

An Approach of Modeling, Monitoring and Managing Business Operations for Just-In-Time Manufacturing

Haifei Li, Jun-Jang Jeng (JJ), Frederick Y. Wu, and Henry Chang
IBM Thomas J. Watson Research Center
1101 Kitchawan Road, Route 134, Yorktown Heights,
NY, 10598, USA
Email: {haifeili, jjjeng, fywu, hychang}@us.ibm.com

Abstract

Recently business process management has received a lot of attentions because of its potential to dramatically improve business efficiency. This paper describes an end-to-end scenario for the business process execution and monitoring for the JIT (Just In Time) schedule execution. JIT is a manufacturing method developed in Japan and later quickly spread into other countries. The purpose of JIT is to eliminate wastes and improve productivity. JIT has been widely adopted in industries like electronics, automobile and aerospace. In today's increasingly competitive business environment, it is quite common that customers may need to change their manufacturing orders. When a manufacturer receives an order change request, the manufacturer needs to conduct a series checks (such as manufacturing capacity, availability of parts, etc) in order to give an answer to the change request. The scenario described in this paper is about how to quickly schedule the manufacturing plan when the manufacturer receives customers' change request. There are many aspects of managing business processes, and the focus of this paper is on describing operational specification, performance specification, and necessary components and artifacts for JIT schedule execution. We will describe the steps involved in the process.

Keywords: Business Operation, Business Operation Modeling, Business Operation Monitoring and Just-In-Time Manufacturing.

1. Introduction

According to the American Production and Inventory Control Society (APICS), JIT [8] is "a philosophy of manufacturing based on planned elimination of all waste and continuous improvement of productivity. It encompasses the successful execution of all manufacturing activities required to produce a final product, from design engineering to delivery and including all stages of conversion from raw material onward. The primary elements include having only the required inventory when needed; to improve quality to zero defects; to reduce lead time by reducing setup times, queue lengths and lot sizes; to incrementally revise the operations themselves; and to accomplish these things at minimum cost." If the JIT method has been implemented successfully, significant competitive advantages can be realized. The JIT can be applied to all parts of a manufacturer: customer order management, purchasing, operations, distribution, sales,

accounting, design, etc. In this paper, we focus on JIT for customer order management in the electronics industry.

JIT schedule execution is triggered by the changes initiated by a manufacturer's customers, but the effects are propagated into its relationship with its suppliers. There are two aspects for JIT schedule execution: operational specification and performance specification. BODL (Business Operation Description Language) has been used to model the operational specification of the JIT schedule execution. The BODL models are transformed into various solution artifacts. A solution composer links various solution artifacts to form a solution template. After some necessary changes to the composed solution template, these artifacts are deployed into an appropriate execution platform (such as WebSphere Application Server) and the whole solution is ready to go. There is another important aspect that is not touched by BODL models. That aspect is performance specification. For example, even if the solution functions properly, business analysts may need to know the performance of the whole solution. The term "Performance" used here includes both IT level performance like throughput, network speed and availability, and business level performance like serviceability (to the manufacturer's customer) and supplier performance (percent of supply request accepted by its suppliers). Furthermore, the business analysts not only need to get the performance data, but also want to know whether the performance data is within a pre-defined bound. If the performance data is out of bound, certain actions need to be taken to remedy the performance deviation. The concept of "business commitment" is introduced to monitor the both the IT level performance and the business level performance.

In order to get the performance data, the monitoring system needs the collaboration of the underlying execution system. The concepts of "probes" and "probe points" have been introduced. In essence, a probe is a piece of code that is "injected" into the underlying execution system to report the status of the execution. Since probes have to deal with the underlying platform, probe definition is platform dependent. A probe point is a logic end point in the BOPM system that represents the basic status of the business process execution. Probe point is platform independent and relies on the probes to provide the execution status data. A probe may have multiple probe points associated with it. The data from the probes is sent to the monitoring system through events. In the context of JIT schedule execution, the monitoring system requires the solution artifacts to be "probe-aware." Specifically, it requires the generated AEs (Adaptive Entity) and BPEL flows to

“understand” the probes. This is done through the model transformation and modification to the existing transformation engine.

Since in most cases, performance data is usually an aggregated result over a period of time for a particular KPI. Therefore, it requires a special type of events (such as EndOfMonth or EndOfWeek) to trigger the evaluation of the KPI computation and its constraint checking. We have identified an independent “Timer Event Generator” as an important component for JIT monitoring.

There are some related works to business process monitoring and execution. Smith and Fingar [5] present an excellent introduction to business process management. The concept of business commitment and related language called Business Operation Performance Language (BOPL) has been described in [2] and [3]. Jeng [4] and others described Business Operation Performance Manager (BOPM), a platform to support BOPL. An execution language for business process has been described in [1]. McGregor and Kumaran [6] described a framework for business process monitoring. IBM Holosofx [7] gives an overview of an industrial product for the business process execution and monitoring.

The rest of the paper is organized as follows. Section 2 describes the scenario in texts. Sections 3 and 4 describe operational specification and performance specification respectively. Section 5 is the system architecture. Section 6 concludes the paper.

2. Scenario Description

This business scenario is typical of an electronics manufacturer that practices just-in-time scheduling. The company has a build plan that may schedule its production lines for approximately a month or two into the future, but the build plan is subject to re-optimization once every few days, or possibly once a week. In the near-term, say the next three days, the schedule is “frozen,” meaning that only minor changes might be made in response to unusual circumstances.

This scenario begins when a customer notifies the manufacturer of a last-minute order change (or a new emergency order) that cannot be accommodated by the current Build Plan. We assume that the Fulfillment process (external to this scenario) has performed an Available To Promise (ATP) check and was unable to satisfy the request without a change to the Build Plan.

The Capacity Check task examines the Build Plan to determine whether the manufacturer’s production lines have the capacity to accommodate the order change. If capacity is not available, that information is returned to Fulfillment, which then rejects the order change. If capacity is available, the Check Supply task translates the order change into a revised component demand schedule, and compares it against the Supply Plan to which suppliers have already committed. If the revised demand does not exceed the bounds of the Supply Plan, then the Build Plan is updated to fulfill the order change, and the Fulfillment process returns a positive response to the customer.

The complexity of the JIT Schedule process arises when the Build Plan can meet the new capacity demand, but the Supply Plan indicates a shortage of one or more components needed for the order change. In that case, the Check Supply task creates a set of parts requests, one for each undersupplied component, and stores the order change request in the Pending Requests repository. At this point, a sub-process procures the undersupplied components, as described below. The Completion task assembles the newly fulfilled Parts Requests and determines whether the procurement of all necessary parts was successful. If so, the Update Build Plan task changes the Build Plan, and returns the response to the Fulfillment process. If not, a negative response is generated to Fulfillment process.

A sub-process contains the tasks that procure the undersupplied components from suppliers. For any given order change, there could be multiple instances of these tasks depending on the number of undersupplied components. The Request Supply task pulls a parts request from a parts request repository. It negotiates with the Primary Vendor to try to get the required amount of the component. If the Primary Vendor can meet the requirement, the pending request is updated and the completed parts request is stored in a repository. If the Primary Vendor cannot meet the requirement (or can only partially supply the increased quantity), the pending request is updated and the Spot Purchase task runs. The Spot Purchase task selects one or more market vendors based on data in the Supplier Profile repository, and negotiates with the Market Vendors for the shortage. The result of the negotiation is used to update the pending request, and the parts request, which may or may not have been satisfied, is stored in a repository. Based on the updates to the pending requests, the Completion task can determine when the attempts to procure all the undersupplied components needed for an order change request have been completed. If all the undersupplied components have been successfully procured from Primary and/or Market Vendors, the Completion task updates the Build Plan and sends commitment messages to the vendors. If one or more components could not be obtained, the Completion task sends a negative response to Fulfillment.

The Request Supply and Spot Purchase tasks record the results of vendor negotiations in the Vendor Profile database. This ensures that the Vendor Profile database reflects the current performance of the vendors.

The JIT schedule execution brings many advantages to the manufacturer:

- Minimize setup / line changes activities.
- Ability to maintain full production capacity.
- Ability to quickly identify alternative sources of supply.
- Dynamic update of supplier evaluation scheme.

3. Operational Specification

The JIT Schedule Execution scenario depicted in Figure 1 is shown in the graphical form of the BODL model. Figure 1 is a screenshot from BVE (Business View Editor) that was developed as an Eclipse (www.eclipse.org) Plugin. In this figure, rectangles represent tasks and ovals represent artifact

repositories. It begins when a customer notifies the fulfillment system (in the manufacturer's side) of a last-minute order change (or a new emergency order) that cannot be accommodated by the current Build Plan. In Figure 1, the Capacity Check task examines the Build Plan to determine whether the manufacturer's production lines have the capacity to accommodate the order change. If capacity is not available, the information is returned to Fulfillment, which then rejects the order change. If capacity is available, the Check Supply task translates the order change into a revised component demand schedule, and compares it against the Supply Plan to which suppliers have already committed. If the revised demand does not exceed the bounds of the Supply Plan, then the Build Plan is updated to fulfill the order change, and the Fulfillment process returns a positive response to the customer.

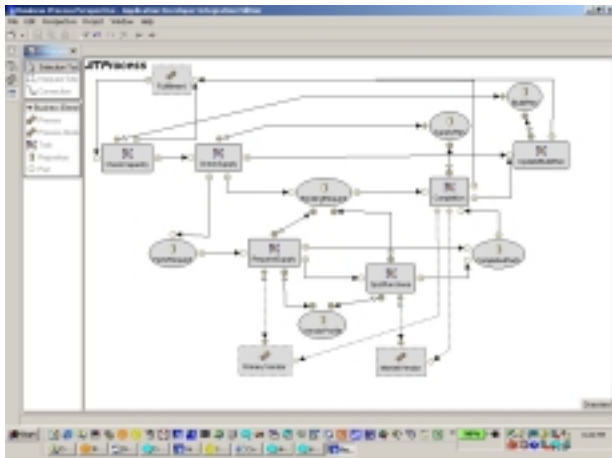


Figure 1. Operational Specification of JIT Scheduling

When the Build Plan can meet the new capacity demand, but the Supply Plan indicates a shortage of one or more components needed for the order change, the manufacturer need to procure more components for manufacturing. In that case, the Check Supply task consults with Primary Vendor at first to determine whether it can accommodate the changes. If so, a positive response is returned. Otherwise, Market Vendors are contacted to get the needed components. After the needed components are available, the Fulfillment process returns a positive response.

4. Performance Specification

Section 3 has described the operational specification for JIT. If implemented appropriately, the execution system runs as expected. However, business analysts want to know more about the execution. For example, the completion task involves a manual approval sub-task that is potentially time-consuming, so the manufacturer needs to know the level of services provided to its customers. Another example is that the manufacturer needs to know the performance of its suppliers for the future supplier selection. In this scenario, we consider the following scorecard parameters: Serviceability Guarantee and Expected Primary Supplier Performance. Serviceability is defined as the "Average Elapsed Time from Fulfillment

Request to Response", and the Supplier Performance is defined as the "Percent of Supply Request accepted by Supplier."

The following figure 2 shows the graphical BODL model together with probes inserted into the model:

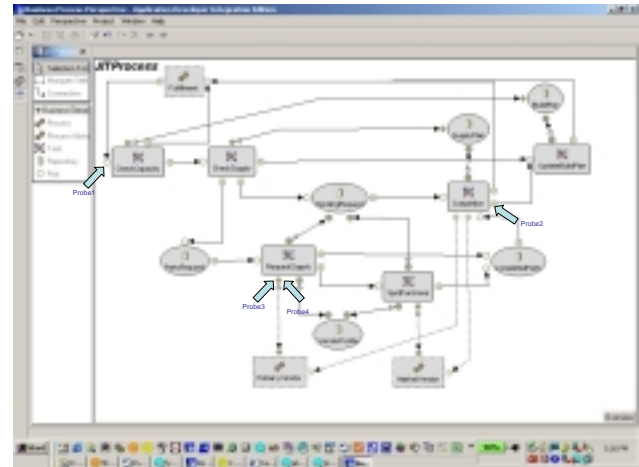


Figure 2. Operational Specification with Probes

The above figure shows the operational specification with probes. Probe1 and Probe2 are used to measure the serviceability. Accordingly there are two probe points: BaseRequestStart and BaseRequestEnd that correspond to Probe1 and Probe2 respectively. The measurement process works as follows. When an artifact (order change request in this case) arrives at the CapacityCheck task, probe1 reports the artifact arrival time, which is sent to the BaseRequestStart probe point. When the artifact leaves the Completion task, Probe2 reports the artifact departure time, which is sent to the BaseRequestEnd probe point.

Probe3 and Probe4 are used to measure the "supplier performance." Accordingly, there are two probe points: "AcceptedSupplyRequestCounter" and "SentSupplyRequestCounter" that correspond to Probe3 and Probe4 respectively. Probe3 records the total number of "Yes" responses back to the "request supply" task. Probe4 records the total number of requests sent to the primary supplier.

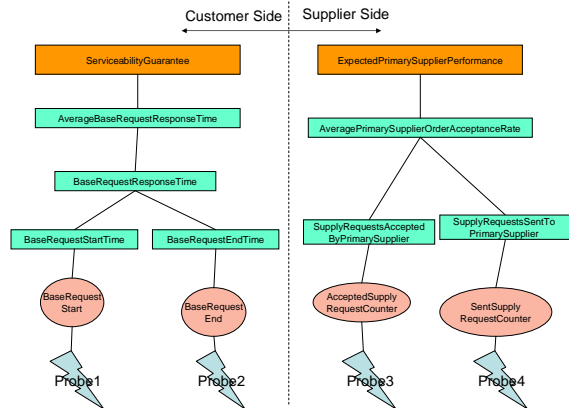


Figure 3. Relationship among Probe, Probe Point, KPI and Commitment

Figure 3 shows the relationship among probe, probe point, KPI and commitment. At the left hand side is the performance specification for the customer. At the right hand side is the performance specification for the suppliers. The figure also shows the hierarchical structures of KPI definitions. As discussed before, “BaseRequestStart” depends on Probe1 and “BaseRequestEnd” depends on Probe2. Two primitive KPIs have been defined: “BaseRequestStartTime” and “BaseRequestEndTime.” The difference between these two KPI values is the “BaseRequestElapsedTime.” The average value of the “BaseRequestElapsedTime” over a period of time (such as one month), is the value for serviceability. A business commitment (from a manufacturer to its customer) called “ServiceabilityGuarantee” would be “Serviceability must be less than 3 (days).”

At the right hand side is the performance specification for the supplier side. Over a period of time, say one month, we are able to get the values for “AcceptedSupplierRequestCounter” (a probe point defined based on Probe3) and “SentSupplyRequestsCounter” (a probe point defined based on Probe4). Two primitive KPIs are defined: “SupplyRequestsAcceptedByPrimarySupplier” and “SupplyRequestsSentToPrimary Supplier.” Once these two KPI values are available, the ratio of these two KPIs is the value for “Average Primary Supplier Order Acceptance Rate.” Based on the value for the acceptance rate, we are able to define commitments like “Acceptance rate must be greater than 0.9”, meaning that manufacturer requires that primary supplier must be able to provide the unexpected demand for components in at least 90% of situations.

5. System Architecture

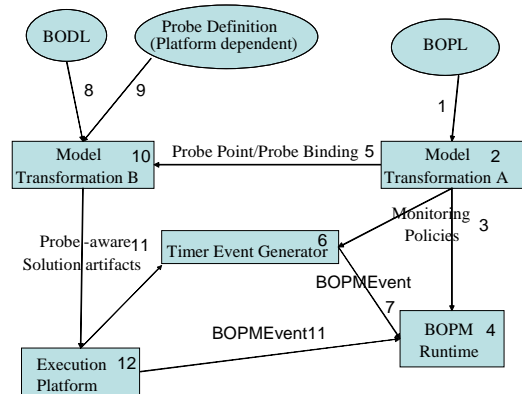


Figure 4. System Architecture for the Execution and Monitoring of the JIT Scenario

Figure 4 shows the system architecture for the execution and monitoring of the JIT schedule execution. Different styles are used for the arrows. The solid arrows are for build-time activities and the dotted arrows are for run-time activities. Numeric labels are used for information flows and software components. It is impossible to describe every flow and component in detail. The following paragraph is a concise description of each of them.

Arrow 1 represents the BOPL document and configuration specification. Arrow 2 is the model transformation. It transforms BOPL and the configuration information into the “monitoring policies” that can be deployed on the BOPM Runtime environment. Arrow 3 is a set of monitoring policies that can be understood by the BOPM runtime. Component 4 is the BOPM runtime that will be described later. Arrow 5 represents the Probe point definition and probe point to probe binding. Component 6 is the timer event generator. Arrow 7 represents the BOPM Events that are generated by the Timer Event Generator. Arrow 8 represents the BODL representation of JIT scenario (in XML document), and Arrow 9 is the platform-dependent probe definition. Component 10 is the model transformation B. It generates “probe-aware” solution artifacts (BPEL, AE and other solution artifacts) that can be deployed on the WebSphere Application Server platform. Arrow 11 is a set of probe-aware solution artifacts. Component 12 represents the execution platform. The preferred platform in our system is WebSphere Application Server.

Since the BOPM runtime (component 4) is an important part of the system, its system architecture is describes in figure 5.

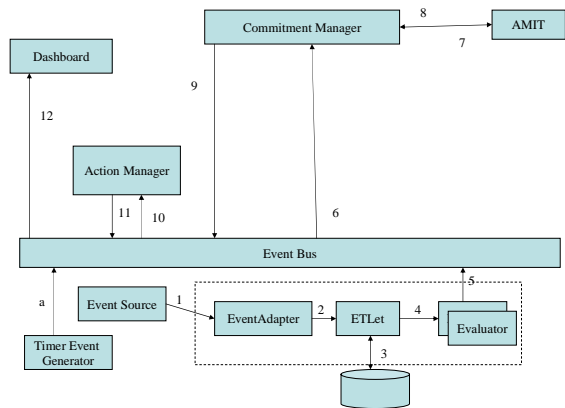


Figure 5. System Architecture for the BOPM Runtime

In the above figure, arrows 1 to 5 are for the processing logic of the ETL (Extraction Transformation and Loading) Container. Events from the event source are fed into the Event Adapter that converts the external events into the BOPM Event format (Arrow 1), then converted BOPM events are imported into ETLet (Arrow 2). ETLet may need to persistently store the data into the database (Arrow 3). When ETL finishes its processing, it needs to push the data into evaluators. We assume that the functionality of Metric Manager (for the purpose of calculating the KPI value) is done by Evaluators inside the ETL Container. Arrow 6 is the instruction sent to the commitment manager (for the purpose of evaluating the guarantees defined in commitments). Right now, the commitment manager depends on the functionality of AMIT to detect the situation (Arrow 7 and Arrow 8). The detected situations are sent back to the event bus by arrow 9. The detected situations are fed back into the Action Manager that publishes the received situation to the dashboard (arrow 12).

Figure 6 shows the relationship of AEs and BPEL flows. In the following figure, the rectangles represent AE (Adaptive Entity) and rounded rectangles represent the BPEL flows that are generated through the model transformation B in figure 4. The rounded rectangles with circle on top of it are microflows that must be executed atomically. Notice that probes (probe1 to probe4) have been inserted into the BPEL flow in the diagram.

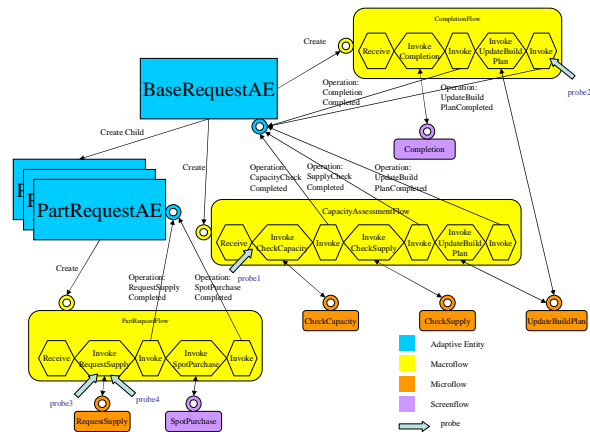


Figure 6 AE and BPEL with Probes

7. Conclusion

In this paper, we have described an end-to-end scenario for the JIT schedule execution. We mainly focus on three aspects: operational specification, performance specification and system architecture. The main contribution of the paper is to document a real-world scenario that can be easily expanded into other areas. Most work on business process management focus on operational specification. In our view, operational specification is only one aspect of process execution, and performance specification is an equally important aspect. Our system architecture takes a model driven approach that is based on industrial standards (AE, BPEL, etc) and can be easily expandable.

Authors would like to acknowledge the support from the following persons at the IBM Thomas J. Watson Research Center: Jen-Yao Chung, Santhosh Kumaran, Fred Wu, Shubir Kapoor and Pawan Chowdhary.

References

- [1] BPEL4WS, Business Process Execution Language for Web Services, available at: <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>
- [2] Haifei Li, Jun-jang Jeng, and Henry Chang, "Business Commitments for Dynamic E-business Solution Management: Concept and Specification", **6th World Multi Conference on Systemics, Cybernetics and Informatics (SCI), 2002.**
- [3] Haifei Li, Jun-jang Jeng, and Henry Chang, "Managing Business Relationships in E-Services Using Business Commitments," **3rd VLDB Workshop on Technologies for E-Services (TES'02), 2002.**
- [4] Jun-Jang Jeng, Steve Buckley, Henry Chang, Jen-Yao Chung, Shubir Kapoor, John Kearney, Haifei Li, and Josef Schiefer, "BPSM: An Adaptive Platform for Managing Business Process Solutions," **the Fifth International Conference on Electronic Commerce Research (ICECR5), Montreal, Canada, 2002.**

[5] Howard Smith and Peter Fingar, "**Business Process Management: The Third Wave**," Meghan-Kiffer Press, 2002.

[6] Carolyn McGregor and Santhosh Kumaran, "Business Process Monitoring Using Web Services in B2B e-Commerce," International Parallel and Distributed Processing Symposium: **IPDPS 2002 Workshops**, Fort Lauderdale, Florida, 2002.

[7] IBM Holosofx, "IBM Holosofx Family," <http://www-3.ibm.com/software/integration/holosofx/>

[8] Inventory Solutions, "What is JIT?" http://www.inventorysolutions.org/def_jit.htm