

TIBCO AMX vs. JBoss Executive Summary



<http://www.tibco.com>

Global Headquarters

3303 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005

© 2009, TIBCO Software Inc. All rights reserved. TIBCO, the TIBCO logo, The Power of Now, and TIBCO Software are trademarks or registered trademarks of TIBCO Software Inc. in the United States and/or other countries. All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

Table of Contents

1. EXECUTIVE SUMMARY	3
2. MODEL-DRIVEN SOA WITH SERVICE VIRTUALIZATION	6
3. USE CASE AND OPERATIONAL GOVERNANCE	6
3.1 USE CASE FOR SOA APPLICATION CONSTRUCTIONS AND DEPLOYMENT	7
3.2 HANDLING CHANGE	8
4. PROBLEMS WITH EXISTING TOOLS/PLATFORMS	9
4.1 TOP LINE OBSERVATIONS.....	9
4.2 DEVELOPMENT SKILLS AND DOMAIN EXPERTISE TO BE SUCCESSFUL	9
4.3 DEVELOPMENT REUSE SKILLS REQUIRED TO BE SUCCESSFUL	9
4.4 DEPLOYMENT SKILLS REQUIRED TO BE SUCCESSFUL	10
4.5 CONCLUSIONS	10
5. COST OF OWNERSHIP DETAILED ANALYSIS	11
5.1 COST OF LABOR	13
5.2 ANNUAL COST OF OWNERSHIP ANALYSIS.....	14
6. REFERENCES.....	16
7. APPENDIX	17
7.1 PURCHASE ORDER SCENARIO.....	17

1. EXECUTIVE SUMMARY

This section provides a summary of key findings, including high-level TCO results.

Much attention – and debate – has focused on the viability of open source middleware and its appropriateness for enterprise-scale projects. For example, a significant number of people in today's development community have found open source application servers to provide a low-cost alternative for many of their production applications. However, at the enterprise level the debate continues as to the viability of open source for mission-critical applications as an alternative to commercial products.

Beyond the performance and scalability argument, a common area of misperception is that open source is ALWAYS a cheaper way to go. Indeed, not having to pay expensive up-front license fees can appear to be an attractive option at first glance. However, there are many additional costs associated with deploying open source middleware that need to be considered, especially when investing in an ESB platform as part of a strategic SOA initiative. Of course there are the support and maintenance fees –which many open source vendors look to for their livelihood. And there is the cost of professional services, which can be extensive in more complex integration scenarios.

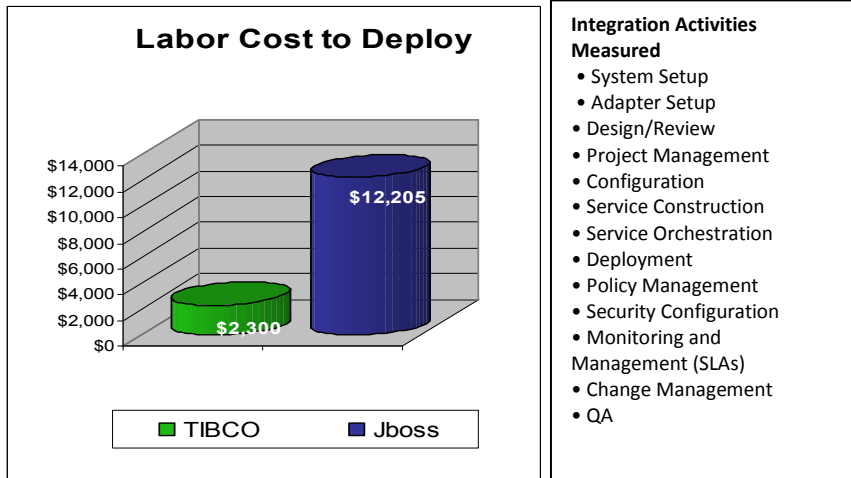
An equally important cost that is often overlooked is the cost of developer productivity. It's unlikely an enterprise architect or developer would ever confuse an open source ESB solution with a commercial product in terms of features and functions – because software vendors invest millions of dollars each year in R&D. They must continue to enhance their products to stay competitive. IT organizations that deploy open source must be willing to tradeoff rich functionality for lower cost – at least this is the perception.

In an effort to quantify the comparative cost of developer productivity, as well as understand the benefits of TIBCO's composition approach to SOA, TIBCO conducted an evaluation of SOA-related products from a leading open source vendor. This evaluation consisted of a time/motion analysis of each step of the services lifecycle to reveal the amount of time and effort required to build, integrate, deploy, and manage a range of services needed to assemble a composite application. The goal of the study was to compare the cost savings/total cost reduction (TCO) that can be achieved through greater productivity.

TIBCO evaluated SOA development and deployment tools and platforms from JBoss and TIBCO to look below surface-level marketing claims to understand the skill sets, domain expertise, and specialization it takes to be successful on each platform. The amount of developer effort to implement an SOA use case was measured and the findings are summarized in the following TCO comparison chart.

As illustrated in the figure below, the TIBCO platform and tools provide the lowest overall labor cost for the integration activities performed in the study. One of the study's greatest surprises was the developer productivity cost related to JBoss.

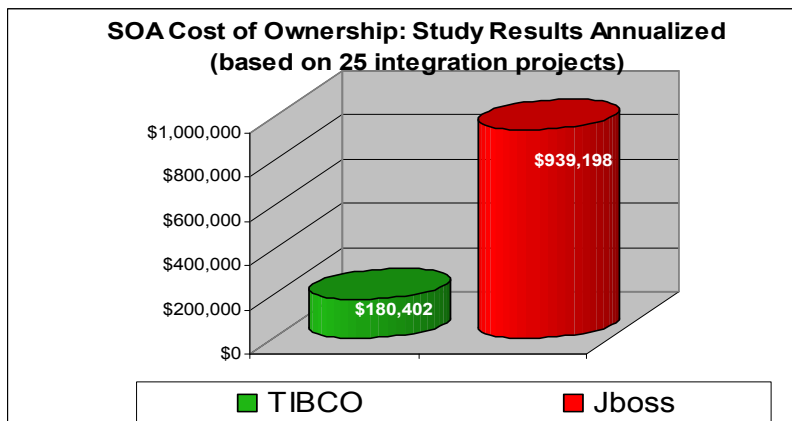
Summary Study Results for Side-by-Side Product Implementation



While JBoss offers basic functionality to build SOA services, the areas of greatest productivity gains using TIBCO were observed in service construction, orchestration, policy management, security configuration, and change management. We found that TIBCO ActiveMatrix® took 81% less time/costs than JBoss. The products used and version numbers are listed in the Resource section (Section 3) of this paper.

The difference in time and effort between vendor platforms is significant. Scaling, or extrapolating, these study results over a longer time horizon suggests that JBoss increases cost of ownership by as much as 83% compared to TIBCO. The figure below extrapolates annualized costs based on study results applied to a representative set of integration projects.

Potential Annual Cost of Ownership Impact Based on Study Results



The statistics above come from a time-and-motion analysis of each tool. To calculate these statistics for your own environment, you can use TIBCO's interactive TCO comparison tool, which is available for download from the [TIBCO SOA Resource Center](#).

After building the use case with JBoss ESB and TIBCO tools, it appears that the TIBCO ActiveMatrix suite provides a more comprehensive integration and SOA stack.

In general terms, we found the following top-line strengths and weaknesses:

- The ActiveMatrix suite provides a host of adapters to integrate with packaged and legacy applications, including mainframes, which is critical in a heterogeneous environment. The total cost of ownership is significantly lowered because the out-of-the-box adapters need zero-coding and are largely configuration driven. With the JBoss stack there are no adapters provided; building an adapter would require significant effort and has a very large overhead associated with it. As the number of applications to integrate with increases, the integration overhead increases significantly, this also increases the cycle time for development. Also, applications that provide a non-Java integration interface would be somewhat difficult to integrate because Java is the only technology supported by JBoss. JBoss does not have any support for integration with CORBA or COM technologies, which is provided out of the box by TIBCO. In JBoss the code would need to be developed for such integrations.
- TIBCO ActiveMatrix provides service governance functionality out-of-the-box. For run-time governance, TIBCO ActiveMatrix® Policy Manager provides pre-built policy templates for logging, auditing, authentication, and authorization using any standard identity management system or X.509 tokens. It also provides full support for SAML. Also, the policy agents can be embedded into third party application servers, providing better performance. With the JBoss stack there is no component providing the run-time governance, and the policies have to be implemented as a part of the service implementation, overloading the developer with development beyond building the business logic. This not only adds to the cost of development but also increases development cycle time significantly and makes it harder to change policies with the needs of the business.
- TIBCO ActiveMatrix provides lifecycle governance, which is critical to any agile enterprise. JBoss provides a UDDI registry, however the service lifecycle governance is a little weak compared to that provided with the ActiveMatrix lifecycle management registry, which provides workflow for managing various versions of the service and promoting them from development to QA, from QA to staging, and from staging to the production environment. With TIBCO ActiveMatrix, rich workflows can be built to control the approval needed, thus providing better governance.
- TIBCO ActiveMatrix® Service Performance Manager provides ease of SLA management based on rules that can change with changing business needs. This is critical functionality that is missing from the JBoss stack, except to the extent that rules can be embedded in the business logic, which makes it rigid and affects the total cost of ownership.
- TIBCO ActiveMatrix provides an easy to use IDE and wizards to help build integrations and composite services using the technology of choice, thus improving developer productivity significantly. TIBCO ActiveMatrix BusinessWorks™ container provides an IDE that helps in building interfaces. ActiveMatrix BusinessWorks is largely configuration driven and is based on XML, XPath, XSLT, and BPEL technologies. ActiveMatrix also provides .NET, C++, and Ruby on Rails containers, providing productivity benefits for service-enabling applications built with the respective technologies. JBoss provides BPEL support, however the support for a third party BPEL engine (ActiveBPEL from ActiveVOS) is much better than for the native engine.
- TIBCO ActiveMatrix is standards-based, supporting SCA, WS-Addressing, WS-Eventing, WS-Transfer, etc. JBoss is weak on support for some of these standards. Also, the service development approach is more developer-centric than service-centric. The approach followed is bottom-up, where implementation decides the service interface, and

not top-down, where the schemas and WSDL are defined first and the service is built using any technology. JBoss provides strong support for transaction management with the Arjuna transaction manager embedded within the ESB framework. JBoss ESB server is lightweight compared to the TIBCO ActiveMatrix nodes.

Taking all of these factors into account, the total cost of ownership would be significantly lower with TIBCO than with JBoss.

2. MODEL-DRIVEN SOA WITH SERVICE VIRTUALIZATION

Service virtualization within TIBCO ActiveMatrix® Service Grid allows for deployment locations, development technology choices, and binding protocols to be changed in the future without any need for code modification. And, OSGI-based deployment allows for granular deployment of components, allowing for maximum uptime of your applications.

As we extrapolate and look at the operational governance and ongoing costs associated with maintenance for this use case, TIBCO offers a significant advantage over JBoss.

Governance is built into the TIBCO ActiveMatrix unified runtime foundation, minimizing human oversight and thus reducing cost. This technical advantage has repeatedly proven to translate into significantly lower TCO.

TIBCO ActiveMatrix Service Grid is build on the SCA standard and can service-enable business logic written in C, C++, .NET, Java, RUBY, or enterprise technologies like Enterprise Java Beans. This is another significant advantage over JBoss, as all the enterprise expert groups can participate in a heterogeneous SOA environment.

3. USE CASE AND OPERATIONAL GOVERNANCE

This section introduces the use case and presents our initial thoughts on what tools and patterns to use.

Evaluating the SOA development tools for service virtualization and composition requires a unique set of use cases, proven test criteria, and an expert methodology to determine which tools best fit an enterprise's requirements. The problem most enterprises encounter when evaluating SOA tools and platforms is that they misread the use cases and underestimate the impact these tools have on developer productivity. TIBCO witnessed several large enterprises fail to gain actionable knowledge from a tools evaluation from use cases that seemed reasonable but did not provide adequate information to determine a winner.

TIBCO's methodology for an SOA, BPM, CEP, and service virtualization evaluation defines a use case with the following mandatory criteria:

- It is an orchestration of services.
- It is a long running process.
- The use case requires connectivity to SAP, JMS, and SOAP services.
- It uses medium to large payload sizes defined by complex schemata.
- It must be possible to instrument as a scalability test.
- It requires reliable messaging (RAMP).

- It uses document-oriented container architecture.

To include these criteria for testing, many different transports and technologies were incorporated. Some of the features included were:

- SOAP over HTTP transport
- SOAP over JMS transport
- XML over JMS transport
- Connectivity with Siebel
- Connectivity with SAP
- Dynamic orchestration of services

Evaluating the competitive differences between tools according to the above criteria can be a complex effort and understanding the results requires expert analysis.

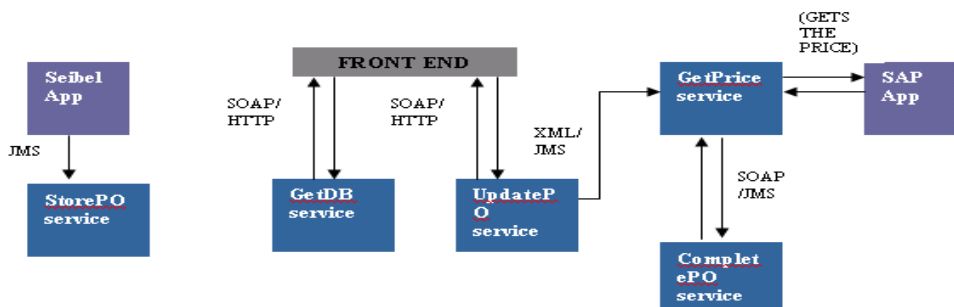
The following is a description of a use case for which we believe SOA patterns are appropriate. The scenario is a manufacturing company that adopts a standard document schema to represent the purchase and shipment of products through a supply chain. The manufacturer chooses an SOA approach using composite applications and data services to interoperate with a set of existing services: financial services group purchase order tracking, warehouse just-in-time inventory control, and price point calculation service.

3.1 USE CASE FOR SOA APPLICATION CONSTRUCTIONS AND DEPLOYMENT

The following flowchart illustrates the use case.



The following illustration shows the lifecycle of a simple purchase order.



The use case begins when a customer places an order for a product using the Siebel application. This begins an SOA workflow that consumes three services. The major logical systems in the scenario are:

- **Allocate Purchase Order** – Uses a service from the financial services group to issue and track purchase orders. When a user creates a purchase order in Siebel, the purchase order is published to this system and the system stores the purchase order in the database. The StorePO service gets the PO details and persists the information on the database.
- **Reserve Parts** – The system depends on a shipping clerk to authorize the inventory to meet the customer’s order. The clerk uses an Ajax-enabled browser interface to view a list of pending orders and quickly check-off multiple approvals. It has 2 services:
 - GetDB- Retrieves the pending purchase orders and the existing parts in the repository from the database and shows the information to the user.
 - UpdatePO - Once the front-end user approves a purchase order, the UpdatePO service is called. The service updates the parts in the repository and status of the purchase order to “approved.”
- **Assign Purchase Order Price** – Uses a service to assign a price point to the purchase order based on the current price catalog. This uses the service GetPrice. The service calls the SAP application to get the prices of all the line items in the purchase order.
- **Complete Purchase Order** – Finalizes the purchase order and indicates to the company system that the order is ready for the customer. This uses the service CompletePO, which calculates the total bill of the purchase order and updates the status of the purchase order in the database.

3.2 HANDLING CHANGE

TIBCO evaluated implementing the use case described above and then evaluated the developer effort to make changes from a product and change management perspective. This effort includes documentation that shows the learning curve, effort to develop, effort to deploy, and effort to maintain and support.

After implementing the original use case, some changes were made to the use case to test how difficult it was to manage such changes with JBoss and TIBCO. Some of the changes were:

- Ability to Use Different Schema
 - An example would be to use different schema for one of the services – recreating a service with the new schema and replacing the original one in the workflow.
- Ability to Use Different Service
 - An example would be the replacement of one service in the use case with another without changing the use case workflow and testing how tough it was to do for both JBoss and TIBCO.
- Ability to Use the Publish/Subscribe Model:
 - An example would be the integration of UpdatePO and GetPrice services and changing it to Publish/Subscribe and gauging the effort level for JBoss and TIBCO.

4. PROBLEMS WITH EXISTING TOOLS/PLATFORMS

This section describes architectural and implementation-level problems we encountered when working with tools from JBoss on the purchase order scenario.

4.1 TOP LINE OBSERVATIONS

We were able to accomplish the use case with each of the tools. However, the experience gave us valuable knowledge we would not have had if we had relied on marketing claims from the tools providers. The following summarizes our top line observations.

- The test team needed to download several tools from JBoss to accomplish the SOA use case. There were no graphic interface conventions and it required proprietary extensions and manual techniques to get the implementation to deploy. For example, JBoss requires four products to build and deploy a service.
- Features missing in JBoss:
 1. No adapters
 2. Does not support .NET, C++
 3. Does not support lifecycle monitoring
 4. No IDE
 5. No adapter SDK
- ActiveBPEL BPEL engine can be used to orchestrate business process flow through JBoss ESB

In addition to our top line observations, we identified issues regarding the development skills, reuse skills, and proprietary graphical interfaces and technology needed to be successful.

4.2 DEVELOPMENT SKILLS AND DOMAIN EXPERTISE TO BE SUCCESSFUL

We found that for each tool vendor developer skills and specific domain expertise are required to be successful.

- JBoss requires Java (JDK), JBoss ESB, JBoss ESB Server, and Apache ANT to be successful. JBoss supports Tomcat, JBoss Application Server, and JBoss ESB Server to deploy archive file.
- TIBCO typically requires Java or .NET, but in some cases neither because SOA development is more about composition of services than the actual classes or integration code one writes in an SOA environment.

4.3 DEVELOPMENT REUSE SKILLS REQUIRED TO BE SUCCESSFUL

We found that each tool vendor also required specific developer skills to accomplish service reuse in an enterprise setting. When using composition to build, deploy, and govern services, the ability to track existing services to avoid reinventing an existing service falls on registry solutions. The following is a summary of our ESB experiences with each of the tools:

- JBoss: It allows us to connect any kind of XML-based registry or repository, for example Systinet Java API.

- TIBCO ActiveMatrix provides good integration to the TIBCO Registry (an integration of Systinet Registry). The development tools featured graphical components to facilitate the interaction from composition designer tools to the registry.

Additionally, we needed to be able to capture the knowledge of one developer who is skilled at an area of development and record that person's expertise into a set of reusable policies to reduce development, configuration, and change management costs. For instance, changing a service interface from SOAP over HTTP to XML over JMS by implementing a policy is a clear win. In our evaluation of the competing tools only TIBCO provides central management and controlled policies such as these.

4.4 DEPLOYMENT SKILLS REQUIRED TO BE SUCCESSFUL

SOA development is more about composition of services than the actual classes or integration code one writes in an SOA environment. It surprised us how little focus the tools had on deployment of the resulting services. The following is a summary of our experience deploying services:

- JBoss: Coding is not required, but setting of path is required.
- TIBCO ActiveMatrix is the only platform we tested that provides a friendly web GUI to deploy and manage services.

4.5 CONCLUSIONS

Our conclusion after building the use case with JBoss and TIBCO tools is that TIBCO offers the following advantages over JBoss:

- Moves services onto the grid to benefit from operational governance
- Orchestration and repository for enhanced productivity in development and production
- Graphical integration development
- Strong interoperability: platform, message schema, service interface independence
- On-ramp for developers to understand and use BPM and CEP productively
- Support for .NET, C++ containers
- Support for lifecycle monitoring
- Adapters
- IDE to build the business scenarios
- SDK to develop adapters

Products used in the evaluation:

JBoss	TIBCO
JBoss ESB 4.5.GA	ACTIVEMATRIX BUSINESSWORKS 5.7
JBoss ESB SERVER 4.5.GA	BUSINESS STUDIO 3.0
APACHE ANT 1.7.1	ACTIVEMATRIX 2.1
JDK 1.5	BUSINESSWORKS SERVICE ENGINE 5.7

5. COST OF OWNERSHIP DETAILED ANALYSIS

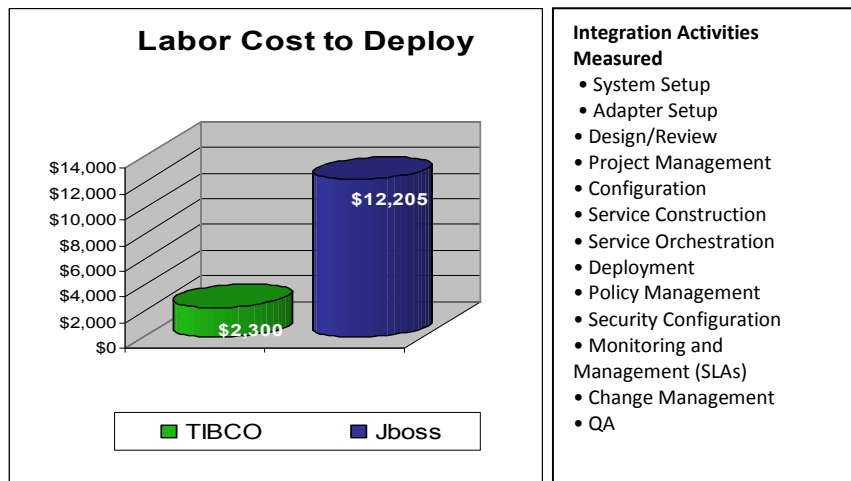
The study compares the time and effort needed to complete a well defined integration project utilizing TIBCO and JBoss platforms. The integration project included a real world deployment scenario for a manufacturing organization – a purchase order creation process and a parts reservation process, as detailed in the report.

For each vendor platform, a variety of metrics were carefully recorded along the lifecycle of common integration activities, from system setup through to post-integration change management and quality assurance activities. The primary metric, and the key to understanding cost of ownership for integration projects, is the time and effort required to complete each activity.

Upon analyzing the data in terms of developer time and cost, the TIBCO platform and integration tools provide a significant advantage in terms of both time and cost.

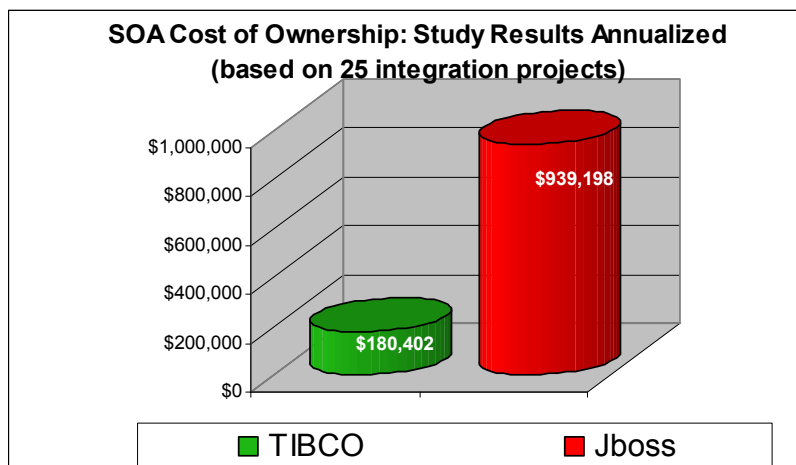
As illustrated in the figure below, the TIBCO platform and tools provide the lowest cost based on the integration activities performed against JBoss platforms and toolsets included in the study.

Summary Study Results for Side-by-Side Product Implementation



Labor Cost: TIBCO was 81% lower than JBoss

Potential Annual Cost of Ownership Impact Based on Study Results



QUANTITATIVE STUDY RESULTS

Analyzing the detailed study results provides additional insights. One important analysis point is the distribution of effort and cost across the various integration activities. The information below shows the distribution of time/effort based on an average of all the vendor projects.

The top four activities (Service Construction, Design/Review, Service Orchestration, Monitoring & Management), in terms of proportion of overall effort, represent two-thirds (65%) of the overall integration effort.

As these activities contribute the most in terms of project cost, it is most interesting to analyze how the vendor integrations fared across these particular activities. The following tables demonstrate the cost associated with these four selected activities by vendor, in order of their place in the lifecycle of the integration project.

The integration project using the TIBCO platform and associated tools resulted in the lowest labor costs for each of these activities. Looking a bit more closely at the study data yields further insights into the differences in time and effort.

SERVICE ORCHESTRATION

Category	TIBCO	JBoss
Labor Cost	\$19	\$750
Cost Basis	0.4 Total Hours 0.4 Developer Hours 0 Architect Hours 0 Project Manager Hours	11 Total Hours 16 Developer Hours 0 Architect Hours 0 Project Manager Hours
Cost Difference	Out-of-the box design-time palettes.	Coding-Centric development. No IDE. Only XML based descriptors.

MONITORING & MANAGEMENT

Category	TIBCO	JBoss
Labor Cost	\$98	\$627
Cost Basis	2.1 Total Hours 2.1 Developer Hours 0 Architect Hours 0 Project Manager Hours	12.3 Total Hours 4.3 Developer Hours 8 Architect Hours 0 Project Manager Hours
Cost Difference	AM Administrator makes it very simple and quick. SPM for SLA management.	Nothing for SLA management. JSON is equivalent to Hawk.

DESIGN/REVIEW

Category	TIBCO	JBoss
Labor Cost	\$617	\$2,848
Cost Basis	11 Total Hours 0 Developer Hours 8 Architect Hours 3 Project Manager Hours	11 Total Hours 28.5 Developer Hours 14 Architect Hours 12 Project Manager Hours
Cost Difference	Development is configuration-centric.	It takes longer, more development-centric. Missing feature/functionality.

SERVICE CONSTRUCTION

Category	TIBCO	JBoss
Labor Cost	\$638	\$4,003
Cost Basis	12.6 Total Hours 8.6 Developer Hours 2 Architect Hours 2 Project Manager Hours	84.4 Total Hours 80.4 Developer Hours 2 Architect Hours 2 Project Manager Hours
Cost Difference	Zero coding in AMBW	Require java coding to build services.

5.1 COST OF LABOR

While the analysis above articulates the bulk of the difference in labor costs based on the study data, it is also interesting to evaluate the labor costs recorded across all of the activities included in the study. These are presented in the following table. The low cost vendor for each activity is highlighted.

Study Results of Labor Costs across all Integration Lifecycle Activities

Activities	TIBCO	JBoss
System Setup	\$108	\$595
Adapter Setup	\$199	\$1,037
Design/Review	\$617	\$2,848
Project Management	\$77	\$119
Integration	\$122	\$378
Service Construction	\$638	\$4,003
Operational Management	\$33	\$131
Service Orchestration	\$19	\$750
Deployment	\$52	\$80
Policy Management	\$0	\$0
Security	\$94	\$963
Monitoring and Management (SLAs)	\$98	\$627
Change Management	\$108	\$445
Performance	\$66	\$113
QA	\$70	\$117
Total All Activities	\$2,300	\$12,205

5.2 ANNUAL COST OF OWNERSHIP ANALYSIS

The following table demonstrates the scaling of the study results presented in Section 7.1 based on a representative enterprise with significant SOA project deployments. The table identifies the cost of ownership assumptions and a description of the scaling factors that were applied to the study results for each of the integration activity line items.

SOA ANNUAL COST OF OWNERSHIP ANALYSIS		
Cost of Ownership Inputs	Assumptions	From study
Avg # of integration projects/year	25	1
Avg # of adapters required/integration project	7	1
Avg # of services per integration project	10	4
Avg # of changes per integration project/year	25	2
Annual salary assumptions		
<i>Engineer/Developer</i>	\$75,000	\$75,000
<i>Architect</i>	\$85,000	\$85,000
<i>Project Manager</i>	\$102,500	\$102,500
<i>Average Annual Burden Rate</i>	25%	25%

	Activity	SCALING FACTOR	TIBCO	JBoss
1	System Setup	one time/initial	\$108	\$595
2	Adapter Setup	based on additional applications to connect to - adapter setup required separately for each integration project	\$34,863	\$181,426
3	Design/Review	per integration project scaled based on avg # of services per integration project	\$38,574	\$178,027
4	Project Management	per integration project scaled based on avg # of services per integration project	\$4,805	\$7,422
5	Integration	per integration project scaled based on avg # of services per integration project	\$7,617	\$23,633
6	Service Construction	per integration project scaled based on avg # of services per integration project	\$39,844	\$250,195
7	Operational Management		\$33	\$131
8	Service Orchestration	per integration project scaled based on avg # of services per integration project	\$1,172	\$46,875
9	Deployment	per integration project scaled based on avg # of services per integration project	\$3,223	\$4,980
10	Policy Management	per integration project scaled based on avg # of services per integration project	\$0	\$0
11	Security	per integration project scaled based on avg # of services per integration project	\$5,859	\$60,156
12	Monitoring and Management (SLAs)	per integration project scaled based on avg # of services per integration project	\$6,152	\$39,160
13	Change Management	per integration project scaled based on # of changes per integration project per year	\$33,691	\$139,160
14	Performance		\$66	\$113
15	QA	per integration project scaled based on avg # of services per integration project	\$4,395	\$7,324
	Total All Activities		\$180,402	\$939,198

TIBCO is less by...

Baseline

81%

6. REFERENCES

- *Products used in the evaluation*

JBoss	TIBCO
JBOSS ESB 4.5.GA	ACTIVEMATRIX BUSINESSWORKS 5.7
JBOSS ESB SERVER 4.5.GA	BUSINESS STUDIO 3.0
APACHE ANT 1.7.1	ACTIVEMATRIX 2.1
JDK 1.5	BUSINESSWORKS SERVICE ENGINE 5.7

- Based on a typical “Enterprise – Order Fulfillment” use case proposed by *PushToTest*
- Implementation and developer logs provided by TIBCO India PSG
- Assessment of fifteen categories of “Integration Lifecycle”
 - System setup, adapter setup, design/design review, project management, configuration, service construction, operational management, service orchestration, deployment, policy management, security configuration, SLA monitoring and management, change management, performance and QA
- Related documents on SalesCentral:
 - Interactive Tool: <https://newsalescentral.tibco.com/docs/DOC-12271>
 - SOA Annual Cost of Ownership analysis: <https://newsalescentral.tibco.com/docs/DOC-12274>
 - Presentation: <https://newsalescentral.tibco.com/docs/DOC-12272>
 - Developer Journal Kits: <https://newsalescentral.tibco.com/docs/DOC-12273>
 - JBoss Developer Journal
 - TIBCO Developer Journal

7. APPENDIX

7.1 PURCHASE ORDER SCENARIO

System Architecture

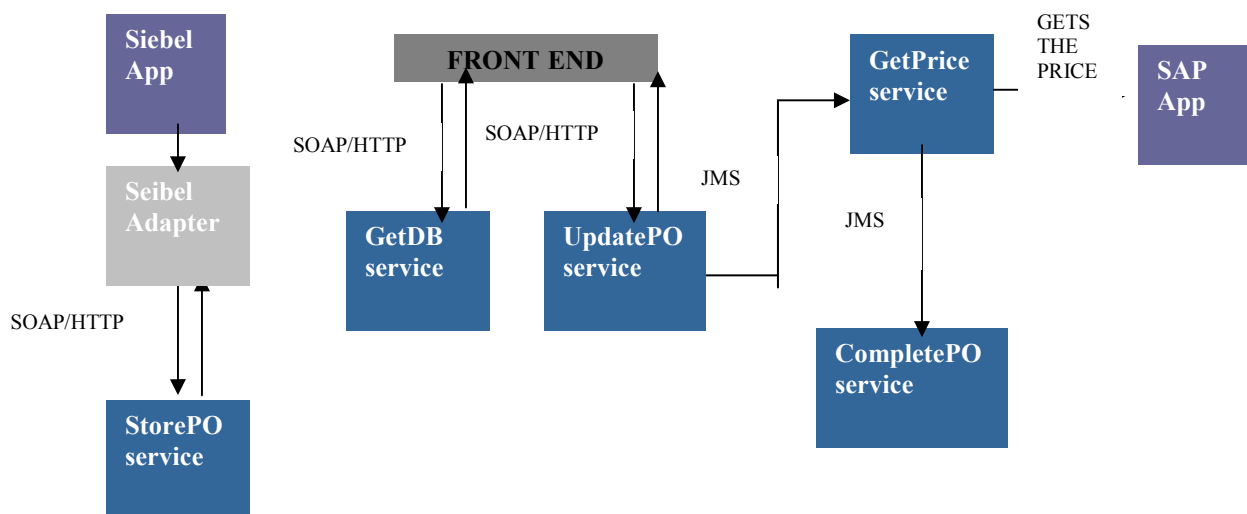


Figure 1: Architecture Diagram

A purchase order is created at the Siebel application. This purchase order is sent via the TIBCO Adapter™ for Siebel to the StorePO service. The StorePO service is called through the SOAP/HTTP interface. This service stores the purchase order (PO) information onto a database, and puts its status as “pending.”

The front-end application is opened by a clerk at the retailer end. The front end connects to the GetDB service and the UpdatePO service.

The GetDB service retrieves the pending purchase orders stored in the database and also the details of all the parts stored in the database. Now the clerk can approve any of the pending purchase orders depending on the availability of parts in the repository.

The UpdatePO service is called by the front-end application. Its purpose is to update the status of the particular order id that is sent to this service. This service updates the status of the particular purchase order from “pending” to “approved” and also updates the quantity of the parts stored in the database repository.

The GetPrice service is called by the UpdatePO service through a JMS queue. This service connects to the SAP application and gets the price information for each of the parts stored in the repository and puts the same in the database.

The CompletePO service is called by the GetPrice service through a JMS topic. This service updates the status of the purchase order in the database from “approved” to “pending.” It also calculates the total bill of the order.

Service Interfaces

StorePO service:

Operation: StorePO
 Input: PurchaseOrder
 Output: StorePOResponse

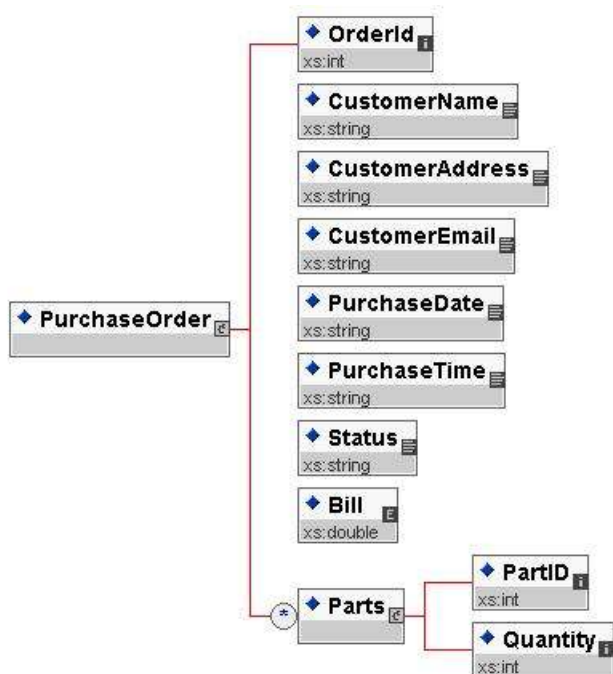


Figure 2. Schema for PurchaseOrder



Figure 3: Schema for StorePOResponse

GetDB service:

Operation: GetPO
 Input: GetDBInput
 Output: Order

Operation: GetParts
 Input: GetDBInput
 Output: Repository

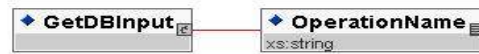


Figure 4: Schema for GetDBInput

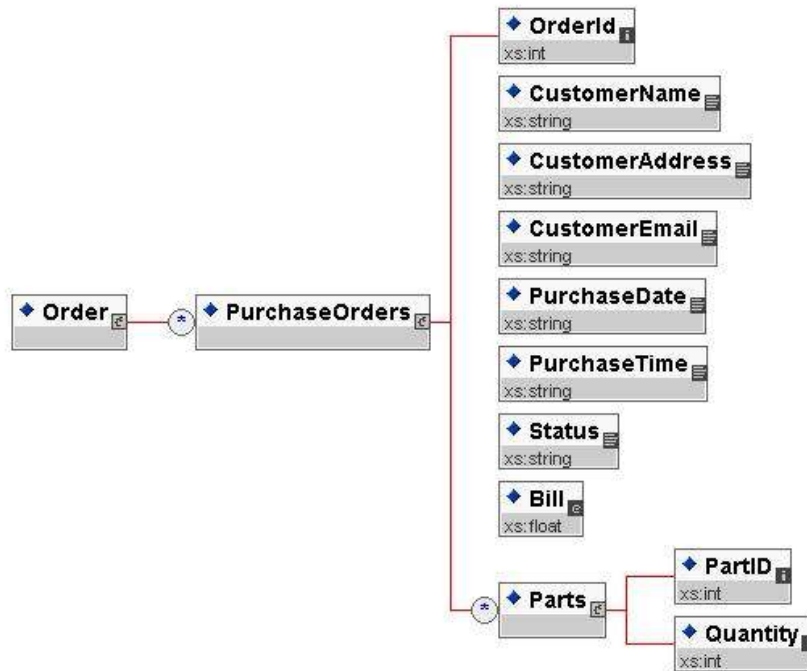


Figure 5: Schema for Order

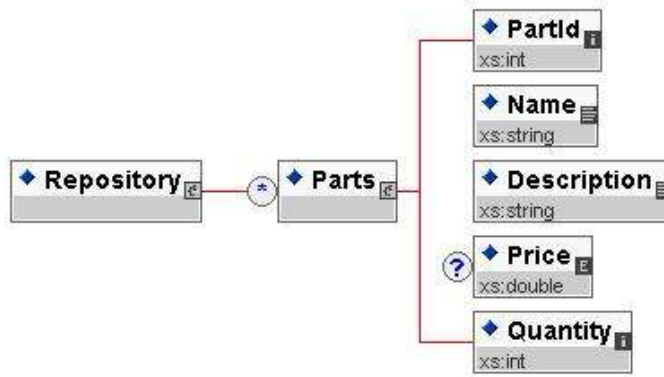


Figure 6: Schema for Repository

UpdatePO service:
 Operation: UpdatePO
 Input: UpdateRequest
 Output: UpdateResponse



Figure 7: Schema for UpdateRequest



Figure 8: Schema for UpdateResponse

GetPrice service:

Operation: GetPrice

Input: PurchaseOrder (Schema same as for StorePO PurchaseOrder)

Output: (no output)

CompletePO service:

Operation: CompletePO

Input: PurchaseOrder (Schema same as for StorePO PurchaseOrder)

Output: - (no output)

DataBase Information

We have Oracle as the database. There are 3 tables. Name and description of the tables are given below.

1. PURCHASE_ORDER

This table stores the purchase order details except for the parts that are ordered.

Name	Null?	Type
ORDER_ID		NOT NULL NUMBER
CUSTOMER_NAME		VARCHAR2(30)
CUSTOMER_ADDRESS		VARCHAR2(50)
CUSTOMER_EMAIL		VARCHAR2(20)
PURCHASE_DATE		VARCHAR2(10)
PURCHASE_TIME		VARCHAR2(10)
STATUS		VARCHAR2(10)

BILL

NUMBER

2. PARTS_ORDERED: -

This table contains the part ids of all the parts that have been ordered in a purchase order. Here order_id is the foreign key to the order_id of the purchase_order table. And part_id is the foreign key to the part_id of the parts_repository table.

Name	Null?	Type
ORDER_ID		NUMBER
PART_ID		NUMBER
QUANTITY		NUMBER

3. PARTS_REPOSITORY: -

This table contains the details of all the parts present in the warehouse. Here part_id is the primary key.

Name	Null?	Type
PART_ID	NOT NULL	NUMBER
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(40)
PRICE		NUMBER
QUANTITY		NUMBER