A new paradigm for decision-making: Business Intelligence and Digital Cockpits

A. Boukhtouta
Defence Research & Development Canada – Valcartier, Quebec, Canada

M. Debbabi
Concordia University, Montreal, QC, Canada

N. Tawbi
Laval University, Quebec, QC, Canada

ABSTRACT:. The main intent of this study is to propose a new paradigm for decision-making that is based on a synergy between business intelligence (BI) and digital cockpits (DC). The cockpits or dashboards are intended to leverage the BI technologies to display a visual, structured, real-time big picture of the entire organization. Such cockpits are endowed with the ability to drill down into details and rely on a single version of the data throughout the organization. This study explores the design, implementation and deployment of such modern BI/DC technologies to dramatically increase the competitive advantage of a given organization by intelligently using available data in decision-making. We propose in this paper a high level architecture of a BI/DC based system. The planning of such a system will be multi-generational. A significant goal of this study is to elaborate a set of recommendations in terms of cutting-edge technologies that could be used to realize such system

1 INTRODUCTION

Information technology is becoming, more and more, a vitally important underpinning of our economy and society. It is embedded in our everyday applications and animates a wide class of systems that range from small to large, and from simple to extremely complex. Actually, information systems increasingly govern nearly every aspect of our lives. This omnipresence is largely increased by the dazzling expansion of the Internet, the World Wide Web, parallel and distributed systems and mobile computation.

Nowadays, large organizations have a dizzying number of departments and services that use a large variety of networked computer systems to collect, process and produce large volumes of information. Moreover, these computer systems are very often disparate in terms of geographic locations, hardware platforms, operating systems and software applications. The ability to transform these data into deeper insights will definitely provide these organizations with a competitive advantage in mission-critical situations. Decision makers, top management and leaders need robust and efficient automatic and/or systematic tools to sense and respond to real-time changes. Accordingly, there is a desideratum that consists in providing software platforms that:

- Provide structured, regular and real-time communication of fresh information among the relevant departments and/or services.
- Keep synchronized and coherent multiple databases.
- Produce status and analysis reports on different activities/processes.
- Scrutinize to the desired level past, ongoing and future activities/processes.
- Handle the needed security services in terms of authentication, secrecy, authorization and integrity.
- Present to a given decision-maker the information in a graphical and a user-friendly way.

As a downstream result, a decision maker will have a real-time “big picture” that integrates all the needed information. He will be in a position to analyze time-variant data (i.e. data that change continuously) to either keep track of data history, or to exhibit trends. He will be in able to navigate deeper in relevant details or to abstract and synthesize data. All these capabilities will allow the decision maker to take effective, educated solid and sound decisions. In this context, our thesis is that business intelligence (BI) solutions together with digital cockpit (DC) technologies are two are two key differentiators. If brought together, in a synergic way, they may lead to fulfill the previously mentioned desideratum.

The primary objectives of this research is to propose a methodology that will allow the:

- Identification of all the sources of information and the underlying databases.
- Identification of all the dependencies between these sources and databases.
- Interconnection of all these sources of information and databases. The aim is to end up with a real-time availability of updated, conciliated, structured and unified information.
- To propose a suite of online analytical processing (OLAP) programs that extracts electronic information from the databases and synthesizes useful knowledge from the collected data using data mining, fact linking, filtering, text understanding and image analysis techniques.
- To propose a suite of digital cockpits that presents the synthesized information as a structured navigational big picture that offers the ability to drill into the details.

The remainder of the paper is organized as follows. The related work is presented in Section 2. The methodology underlying the proposed system is presented in Section 3. The system architecture is presented respectively in Sections 4. The conclusion of the paper is given in Section 5.

2 RELATED WORK

BI is a broad category of applications and technologies for gathering, storing, analyzing, and managing data to help decision makers to take better business decisions. BI systems synthesize useful knowledge from large data sets (Skriletz 2002). They integrate many data coming from different sources. They summarize and abstract data. They show ratios and trends. They compare databased information with model-based assumptions and try reconciling them when they are different. To summarize, the basic functions that a BI system have to offer are: data collection or gathering, data organization and structuring, data extraction, transformation and loading, data analysis and synthesis, information visualization, decision capabilities and scenario analysis (Hao et al. 2000). Examples of BI applications include traffic analysis, fraud detection, and customer loyalty analysis in the telecommunications industry, which require analyzing large volumes of call detail records; and target marketing, market basket analysis, customer profiling, and fraud detection in the e-commerce industry, which require analyzing large volumes of shopping transaction data from electronic storefront sites (Power 2002 , Skriletz 2002). BI applications include the activities of decision support systems, query and reporting, OLAP, statistical analysis, data warehousing, and data mining (Inmon 1996, Watson et al. 1997). Typically, business intelligence applications involve extracting data from operational databases and organizing it into data warehouses. This data is then used as input to a reporting, querying, OLAP and data mining tools. These tools role is to show trends and generate relevant information that are directly used in the decision making process.

An important function that BI systems offer is information visualization. Visualization is a key feature because it enables managers to perceive patterns in the data, and to easily choose the subsets of data on which to focus further detailed analysis. Visualization prevents managers from being overwhelmed with a great amount of data analysis and allows them to work without intermediaries helping them to make results interpretation. Most corporations have analytical tools that help them to make sense of the massive amounts of data contained within their operational systems. These tools allow users to generate reports that analyze slices of data, or to make complex queries that help measure business performance. However, a major drawback to these solutions is that they require users to take time out of their day to run reports and interpret the results. This is the reason why graphical methods for visualizing results of analysis are required. Business dashboards or digital cockpits are introduced to address this need.

3 METHODOLOGY

To reach the previously stated objectives, we recommend the use of a synergy between BI and DC. We structure such a methodology according the following paradigm: *Integrate, Display, Monitor, Analyze and Control*. Here is briefly the meaning of each step in this paradigm:

- *Integration*: What are the facts?
- *Display*: What happened or what is happening?
- *Monitoring*: So what?
- *Analysis*: What to do?
- *Control*: What is optimal?

In what follows, we explain for each component of this methodology the objectives, the benefits, as well as the technologies that might be used.

3.1 Integration

Data integration aims to connect all the sources of information within the organization. This requires the identification of:

- The sources of information to be integrated.
- The dependencies between these sources of information.

- The data formats and protocols to be used in information sharing.

As a result of this integration, the databases and data warehouses will be dynamically updated in a real-time way with freshly produced information. Consequently, all the sources will be synchronized and coherent. As a downstream result, this means that all the interested users will, in real-time, the information they need to make appropriate decisions.

Nowadays, the best enterprise technology in data integration consists in using messaging services. Messaging is a reliable communication method between software applications or middleware. Generally, a messaging system is a peer-to-peer capability. Each peer connects to a messaging client to send or receive messages. Messaging agents are entities capable of creating, sending, receiving and reading messages. Messaging is a form of loosely coupled distributed communication since the availability of the two involved entities (source and destination) is not required at the same time. This makes messaging an asynchronous communication technology. The most prominent technology in this arena is the Java Message Service (JMS). The latter is a specification of a Java API that provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an organization. It allows applications to create, send, receive, and read messages. Designed by Sun and several world-leading companies, the JMS API defines a standard set of interfaces and classes together with their associated semantics to allow Java applications, systems and middleware to communicate with other messaging implementations. The main distinctive features of the JMS API are:

- *Minimal set of concepts*: The JMS API comes with a minimal set of concepts, which makes the learning curve for a Java developer very reasonable. These few concepts are however very expressive and allow the developer to write powerful and sophisticated messaging applications.
- *Asynchronous communication*: In JMS, messages are delivered to available recipients as they arrive. The recipients do not need to require messages in order to receive them.
- *Reliable communication*: In JMS, messaging is reliable. Actually, the delivery of messages is guaranteed. Lower reliability levels are however available for applications and components that can afford to miss messages or receive duplicated messages.

The first publication of the JMS Specification came in August 1998 (Java Message Servivce API, 1998). Since the release 1.3 of the Java 2 Enterprise Edition (J2EE™), the JMS reference implementation is an integral part of the platform. The JMS addition enhances the J2EE platform by simplifying enterprise development, allowing loosely coupled, reliable, asynchronous interactions among J2EE components and legacy systems. The JMS messaging outperforms other communication solutions such as RMI or RPC in many situations such as:

- The messaging provider wants the components not to depend on information about other components' interfaces, so that components can be easily replaced.
- The messaging provider wants the application to run whether or not all components are up and running simultaneously.
- The business model is such that an application will send some information to another component and continue to operate without receiving an immediate response.

In such situations, JMS messaging allows the communication between the different components of a system without tying up network resources. The JMS API in the J2EE platform allows applications, Enterprise Java Beans (EJBs) components, and web components to send or synchronously receive JMS messages. Moreover, applications can receive messages asynchronously. Thanks to the implementation by the JMS API of a new EJB called the message-driven bean, the asynchronous consumption of messages is supported. The J2EE platform's EJB container architecture, moreover, enhances the JMS API by providing support for distributed transactions and allowing for the concurrent consumption of messages.

In what follows, we explain the basic JMS API concepts so as to make this paper self-contained. Accordingly, we will discuss the JMS API architecture, messaging domains and message consumption. A JMS application is made of the following components:

- *JMS provider*: It is a messaging system that implements the JMS interfaces and provides administrative and control capabilities. Any implementation of the J2EE platform (release 1.3 and above) must include a JMS provider.
- *JMS clients*: They are Java applications that use the JMS provider to consume and/or produce and send messages.
- *Messages*: They are the objects sent and received by the JMS clients. They vehicle the communicated information.
- *Administered objects*: They are pre-configured JMS objects created by an administrator in order to be used by JMS clients. There are two kinds

of administered objects: destinations and connection factories. A destination is the object that a client uses to specify the target of messages it

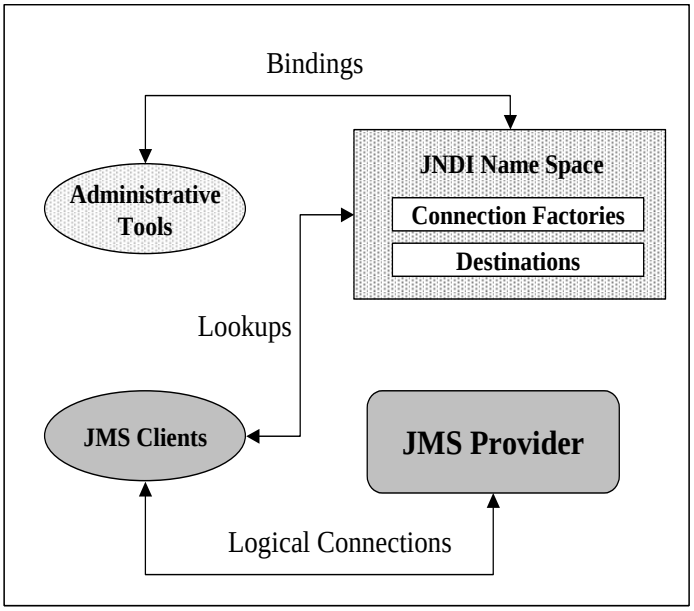

Figure 1. JMS API Architecture

produces and the source of messages it consumes. A connection factory is the object that a client uses to create a connection with a JMS provider.

- *Native clients*: They are applications that use a native messaging API instead of the JMS API.

Figure 1 illustrates the JMS architecture and how the different components interact with each other. Actually, the administrative tools allow the binding of destinations and connection factories into a Java Naming and Directory Interface (JNDI) namespace. A JMS client can then look up the administered objects in the name space and establish a logical connection to the same objects through the JMS provider.

Prior to the elaboration of the JMS API, most of the messaging systems supported either the *point-to-point* (PTP) or the *publish/subscribe* (pub/sub) approach to messaging. Now, with the JMS specification both approaches are supported. In what follows, we recall each of these 2 approaches.

- *Point-to-Point (PTP) Messaging Domain*: A PTP messaging system is built around the concept of message queues, senders, and receivers. Each message is sent to a destination queue. Receiving clients extract messages from the queue(s) established to hold their messages. Queues store all messages sent to them until they are consumed or expired. A PTP messaging system has the following traits:
  - Each message has only one consumer
  - No temporal dependencies between a sender and a receiver of a message
  - The receiver acknowledges the successful processing of a message

Figure 2 illustrates the PTP messaging approach.

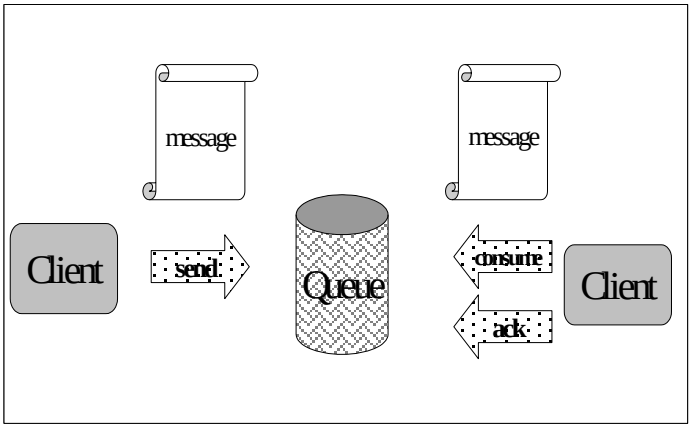

Figure 2. Point-to-Point Messaging

- *Publish/Subscribe (pub/sub) Messaging Domain*: In a pub/sub messaging system, clients send messages to recipients that subscribed to a particular topic. We generally abstract this by saying that clients send messages to topics. Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to the content hierarchy. The system takes care of the distribution of messages arriving from a topic's multiple publishers to its multiple subscribers. Topics hold messages only as long as it takes to distribute them to current subscribers. Pub/sub messaging systems have the following traits:
  - Each message may have multiple consumers.
  - There is a temporal dependency between publishers and subscribers.
  - A client can consume messages only if it has created a subscription, to the corresponding topic.

Figure 3 illustrates a pub/sub messaging system.

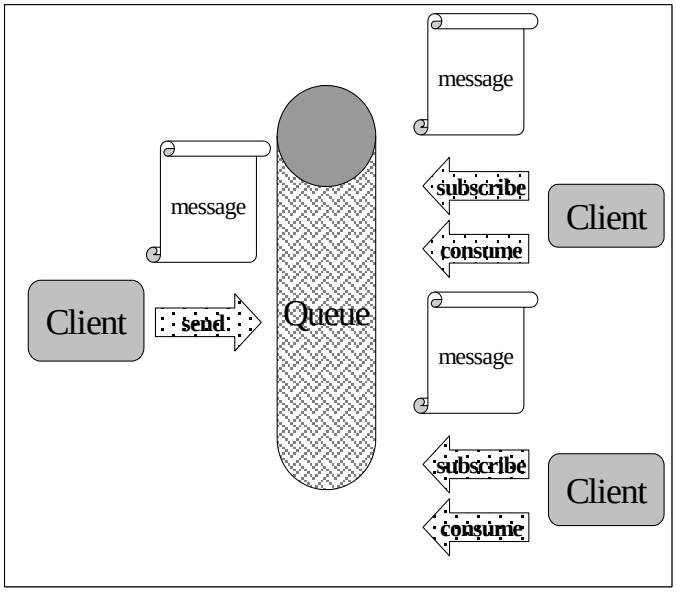

Figure 3. Publish/Subscribe Messaging

An inherent characteristic of messaging systems is asynchrony i.e. there is no temporal dependency between the production and the consumption of a message. In the JMS specification, messages can be consumed in one of two ways:

- *Synchronously*: A subscriber or a receiver explicitly consumes the message from the destination by calling the receive method. The receive method is blocking until a message arrives. More accurately, it can time out if a message does not arrive within a specified time limit.
- *Asynchronously*: A client can register a *message listener* with a consumer. A message listener is similar to an event listener. Whenever a message arrives at the destination, the JMS provider delivers the message by calling the appropriate listeners.

In what follows, we will survey the most prominent JMS messaging products as well as a comparative study and a set of recommendations. The most prominent JMS providers are:

- *IBM MQSeries*: It is the integrated JMS provider in the WebSphere Application Server. IBM MQSeries enables application integration by helping business applications exchange information across platforms by sending and receiving data as messages. MQSeries takes care of network interfaces, ensure "once only" delivery of messages, manage communications protocols, dynamically distribute workload across available resources, handle recovery after system problems, and help make programs portable. MQSeries provides a consistent, multi-platform API. MQSeries Java and an implementation of the Java Messaging Service API are supplied with the product. Additional capabilities include scalability, security, system administration, and the ability for applications to operate as publish/subscribe brokers on most platforms and as publishers or subscriber on any platform to automate the distribution of relevant information to all applications that have registered an interest in a particular topic.
- *Sun ONE Application Server 7 Integrated Message Queue*: It is the integrated JMS provider in the Sun ONE Application Server 7. Sun ONE Message Queue is a standard-based, message-oriented middleware product that fully implements the Java Message Service API, which is part of the core J2EE platform. Using the Sun ONE Message Queue integration solution, developers can establish asynchronous modes of communication between applications. Developers can also take advantage of an infrastructure that is ready to support web services today and in the future. In addition to implementing the JMS specification, the Sun ONE Message Queue provides features that include support for various messaging models, multiple broker clustering for scalability, message service administration tools, a flexible security architecture, and fail-over.
- *SonicMQ*: It is the industry's most reliable message server, delivering unsurpassed scalability, extensive connectivity, and comprehensive security for business-critical communication across the extended enterprise. SonicMQ's open, enterprise-class backbone guarantees that messages are never lost, even in the most challenging situations. It's proven performance, patent-pending Dynamic Routing Architecture (DRA) and advanced clustering technologies allow SonicMQ deployments to scale without limit. Superior authentication, authorization, and encryption support ensures that messages and systems are protected inside and outside the firewall. SonicMQ seamlessly integrates with industry-leading J2EE Application Servers such as BEA WebLogic Server and IBM Websphere, expanding the reach of global business infrastructure.

As a JMS platform we recommend the use of SonicMQ for the following reasons:

- Based on open standards
- Reasonable cost of ownership
- Most reliable JMS messaging system
- Fastest JMS messaging system
- Support of the most recent J2EE standards

Actually, an independent company called Jahming Technologies has conducted several comparative studies of various JMS providers (Jahming technol-

ogies 2001). The major outcome of these studies is that SonicMQ JMS middleware outperforms its competitors such as IBM MQSeries now called WebSphereMQ according to a series of independently performed messaging benchmark tests. SonicMQ is consistently faster in 40 different business messaging test configurations, with throughputs that are 100 - 1500 percent faster than those of WebSphereMQ. SonicMQ 4.0 achieved overwhelmingly higher message rates for both topics and queues. The comprehensive message performance comparison included point-to-point and publish/subscribe domains from light to very heavy message loads. The tests compared two implementations of the JMS API, which allows software applications to create, send, receive and read messages between applications in a reliable, secure fashion throughout the extended enterprise.

These comparative third party benchmark tests, conducted by Jahming Technologies, demonstrate that SonicMQ 4.0 outperformed WebSphereMQ 5.2 in all areas of testing:

- Point-to-point testing-in one-to-one message tests: SonicMQ 4.0 produced throughputs of up to four times that of WebSphereMQ.
- Durable publish/subscribe testing: At light loads, Sonic outpaced IBM by a 2-1 margin. That gap widened to 6 times the throughput under medium load testing. SonicMQ broker easily handled a high-volume of activity (1,000 connections and 500 topics), while WebSphereMQ broker failed to successfully complete the test.
- Non-durable publish/subscribe testing: For both one-to-one and few-to-many tests, SonicMQ yielded message rates significantly faster than WebSphereMQ. In one test, the SonicMQ broker achieved throughputs 16 times faster than those of the WebSphereMQ broker.

3.2 Dispalay

The main intent of this phase is to take the data from different sources, aggregate them and present the synthesized information into a meaningful, structured and navigational big picture that offers the ability to drill into the details (Hao et al. 2000). This is achieved by the elaboration of graphical capabilities, user interfaces, web/portal services. By doing so, we endow the system with what is called digital cockpits or dashboards. This will help the decision maker to grasp quickly the semantics of the presented data and take quickly the appropriate decisions and measures. Such dashboards will reflect a graphical view on "*what happened*" or "*what is happening*" in the organization.

The technology challenges in the elaboration of such phase are: Data gathering, legacy integration, presentation, security, visualization and data integrity. Several environments to build digital cockpits and web/portal services are available. We conducted a research study of these environments and the one that we strongly recommend is *Jitzu* from *Information Architect Inc*. The reasons that motivate the choice of Jitzu are:

- Jitzu is an ideal platform for digital cockpit elaboration because it:
  - Allows real-time views of the organization entities.
  - Allows a quick view of all data that are required to make complete business decisions even though that data is in several different systems or applications.
  - Allows the access to a large variety of systems and databases.
  - Allows the unification of data that is originating from the outside of the organization.
  - Allows the creation of reusable components that describe various business rules and logic, data and data sources thanks to the so-called meta-data concept.
  - Allows the graphical presentation of business rules, data, business content by simply dragging and dropping
  - Allows the instantaneous addition, deletion, modification of any content, security, data, programs, business intelligence, etc.
  - Allows a bi-directional flow of information.
- Jitzu is a secure platform.
- Jitzu is reasonably priced.
- Jitzu officials exhibit excellent responsiveness, openness, and dynamism.

Figure 4 illustrates the entities that a Jitzu server environment can talk to. It shows also the so-called Jitzu meta-database.

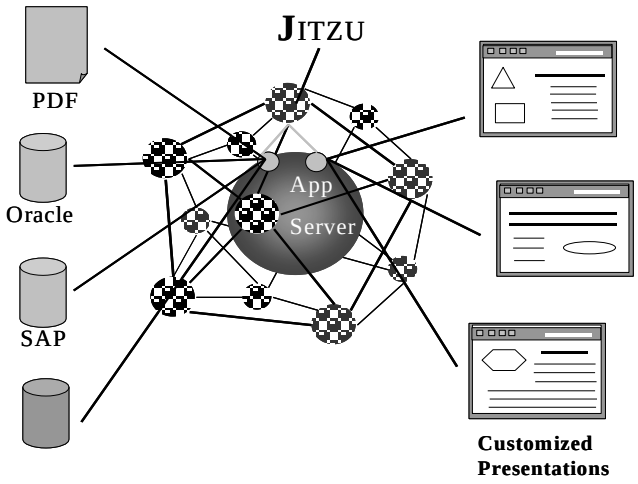

Figure 4. The Jitzu environment

3.3 Monitoring

This phase corresponds to the implementation of capabilities that allow the active monitoring of the system state. In other words, these capabilities are meant to reflect the "*so what*" i.e.:

- Test of business assumptions.
- Reactive instruments.
- Elaboration of proactive measures.
- Response to dashboard thresholds.

The expected benefits of this phase are:

- Awareness.
- Create a biz user pull.
- Create "realer time" thinking.
- Heads reduction for data roll-up & active data gathering.

The technology challenges are:

- Data Gathering
- Web scraping i.e. getting information from a web page and reformatting it.
- Mapping data sources to data needs
- Information security.
- Reliability.
- Scalability.
- Interface to create alerts

Such capabilities could be built using web and portal services tools. In the case of this study, we recommend again the use of Jitzu.

3.4 Analysis

This phase is actually what is going to bring the system to the business intelligence level. It reflects the "*do what*" i.e.:

- Business Intelligence capabilities.
- Dashboard "Knobs".
- Simulate "What-if" scenarios
- Patterns & Drilldowns
- Analytics
- Look ahead confidence intervals

The expected benefits are:

- Resource utilization productivity
- Faster, higher confidence decisions enabled

The technology challenges of this phase are:

- Model accuracy.
- Simulation scenario development.
- Data freshness and availability.

3.5 Control

This phase is meant to implement what will make all the digitized organization processes optimized. This reflects the "*what's best*" i.e.:

- Active control via cockpit
- System wide optimizations
- Suggestion of best potential scenario
- A framework to get businesses to think and react in real-time for competitive advantage.

The expected benefits are:

- Higher productivity
- Optimal processes.

The technology challenges are:

- Data integrity
- Robust transfer functions

4 ARCHITECTURE

The suggested architecture is made of two major ingredients:

- A messaging system that is JMS-based.
- A web/portal services system (Bhargava & Power 2001 , Power 2002) that is based on the Jitzu technology.

Figure 5 exhibits the messaging architecture used for data integration. As of the planning of such architecture, we propose a multi-generational approach:

- *Generation 1*: *Integrate and Display*:
  - Identification of the organization computer systems to be integrated according to the BI/DC paradigm.
  - Identification of all the sources of information and the underlying databases to be integrated.
  - Identification of all the dependencies between these sources and databases.
  - Selection of the JMS provider. We recommend the use of SonicMQ.
  - Architecture, design and implementation of the JMS messaging system.
  - Architecture, design and implementation of web services that display in a structured, graphical and navigational way the information.

- *Generation 2*: *Monitor*:
  - Architecture, design and implementation of business assumption testing.
  - Architecture, design and implementation of reactive instruments.
  - Architecture, design and implementation of proactive measures.
  - Architecture, design and implementation of responses to dashboard thresholds.
- *Generation 3*: *Analyze*:
  - Architecture, design and implementation of OLAP procedures and more generally business intelligence capabilities.
  - Architecture, design and implementation of dashboard "Knobs".
  - Architecture, design and implementation of the simulation of "What-if" scenarios.
- *Generation 4*: *Control*:
  - Architecture, design and implementation of active control via cockpit.
  - Architecture, design and implementation of system wide optimizations.

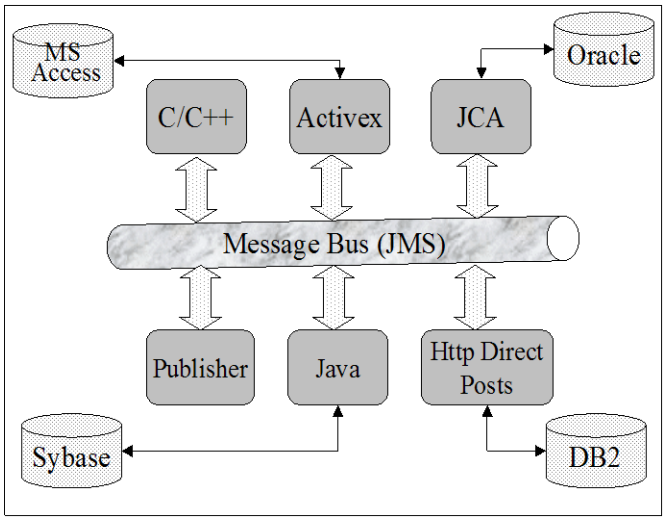

Figure 5. Messaging Architecture for Data Integration

5 CONCLUSION

We reported in this paper a new paradigm for decision-making that is based on a synergy between business intelligence (BI) and digital cockpits (DC). These cockpits or dashboards are intended to leverage the BI technologies to display a visual, structured, real-time big picture of the entire organization. Such cockpits are endowed with the ability to drill down into details and rely on a single version of the data throughout the organization. This study explores the design, implementation and deployment of such modern BI/DC technologies to dramatically increase the competitive advantage of a given organization by intelligently using available data in decision-making while keeping security in mind. We present a brief survey of the state of the art BI/DC technologies. Then we propose a high level architecture of a BI/DC-based system. The planning of such a system will be multi-generational. A significant goal of this study is to elaborate a set of recommendations in terms of cutting-edge technologies that could be used to realize such a system.

6 REFERENCES

Bhargava, H. & Power D.J. 2001. Decision Support Systems and Web Technologies: A Status Report. Proceedings of the 2001 Americas Conference on Information Systems, Boston, MA, August 3 - 5.

Dhar, V. & Stein, R. 1997. Intelligent Decision Support Methods: The Science of Knowledge. Upper Saddle River, NJ: Prentice-Hall.

Ellison, C., Frantz, B. Lampson, R. Rivest, B. Thomas & T. Ylonen. 1999. SPKI Certificate Theory, Internet Network Working Group RFC2693.

Gardner, S. R. 1998. Building the Data Warehouse, Communication of ACM, Volume 41, Number 9, 52-60.

Hao, M., Dayal, M. & Hsu M. 2000. Visual Data Mining for Business Intelligence Applications, WAIM 2000 Conference, China.

Inmon, W.H. 1996. *Building the Data Warehouse*, John Wiley and sons.

Jahming Technologies, 2001. High Performing Messaging with JMS™ A Benchmark Comparison of SonicMQ® 4.0 vs. IBM® MQSeries® 5.2, http://www.jahming.com/news.html.

Java Message Service API. Sun MicroSystems, http://java.sun.com/products/jms/

Ladley, J. 1997. Operational Data Stores: Building an Effective Strategy, Data Warehouse: Practical Advice from the Experts, Prentice Hall, Englewood Cliffs, NJ.

Nylund, A. 1990 Tracing the BI Family Tree, Knowledge Management, July.

Power, D. J. 2002. A Brief History of Decision Support Systems. In DSSResources.com, *http://DSSResources.com/history/dsshistory.html* version 2.

Skriletz, R. 2002. New Directions for Business intelligence: Critical Lessons from the Decade of Business Intelligence and Data Warehousing, In *DM review, April.*

Watson, H. H. & Haley, B. J. 1997. Data Warehousing: A framework and survey of current practices, Journal Data warehousing 2, No 1, 10-17.