What middleware for network centric operations?

Ali Benssam a, Jean Berger b, Abdeslem Boukhtouta b, Mourad Debbabi a,*, Sujoy Ray a, Abderrazak Sahi b

a Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Que., Canada
b Defence Research and Development Canada, Canadian National Defence, Valcartier, Que., Canada

Received 28 October 2005; accepted 6 May 2006
Available online 5 October 2006

Abstract

The main intent of this paper is to address the issue of middleware in network centric operations. To this end, we characterize a set of Information Technology capabilities that such a middleware should implement. Afterwards, we will discuss the design and architectural aspects of these capabilities. This will lead us to an efficient and practical decision support system that we call a digital cockpit. The latter is essentially a multi-tier IT platform that provides a plethora of services such as: data and service integration, monitoring, analysis, and process optimization. Moreover, the platform uses advanced display mechanisms to render the information in a structured and navigational representation that offers the possibility to drill down into the details. A significant subgoal of the paper is to discuss the quality attributes of such an NCO middleware. Finally, we present the results of an implementation of the aforesaid platform architecture.

© 2007 Published by Elsevier B.V.

Keywords: Network centric operations; Middleware; Information technology; Data and service integration; Display; Monitoring; Analysis; Optimization; Decision support

1. Motivation and background

The Network Centric Operations (NCO) is a term that encompasses a plethora of concepts that are popular in many countries, military organizations and institutions, e.g.:

- Network Enabled Operations, NEOps (Canada).
- Network Centric Warfare, NCW (US).
- Network Enabled Capabilities, NEC (UK).
- Network Based Defense, NBD (Sweden).
- Network Enabled Operations, NEO (Australia).

It stands for the embodiment of information age in military intelligence, planning, operations, and logistics. This entails the seamless network connectivity and interoperability of major military platforms, systems, devices, sensors, and battle-space entities leading to remarkable situation awareness. Consequently, NCO constitutes a paradigm shift where battle-space entities are no longer stove-pipes (standalone entities) but rather a collection of network-enabled and network-ready entities that communicate and interoperate in a seamless and synergistic way.

Progressively emerging from the co-evolution of organization, technology, and doctrine, net-centricity aims to provide shared information, shared intent, agility, and resource/decision synchronization. It addresses stove-pipe systems proliferation, legacy system, connectivity, reusability, and provides a suitable framework to enable organizational transformation and organization integration through net-centric enterprises services, to support analysis, coordination and to ensure decision superiority and facilitate transition from a platform centric environment for centralized/distributed resource management (complexity, multiplicity, diversity, and dynamic uncertain environment).

The critical cornerstone of any NCO platform is the underlying information system. Information superiority, situational awareness, and self synchronization, which are...
the main promises of the NCO paradigm are completely dependent on the quality of the NCO information system. Therefore, its architecture, design, and implementation are key factors in the success of the operations in the battlefield. Hence, a special care should be devoted to the engineering of NCO information systems.

In this paper, we advocate the use of a middleware-based approach to elaborate the NCO information system. Such an approach is beneficial to both developers and users of NCO applications. As of developers, it eases their design and programming tasks by abstracting from the complexity underlying the inner workings of NCO core software components. Also, it provides more functional and high-level set of application programming interfaces that increase the productivity while improving several quality attributes such as scalability and maintenance. Moreover; a middleware approach will improve the location transparency, which is a significant advantage especially with the distributed nature of NCO applications. The use of open standards within such a middleware will definitely ease more the development of applications. Finally, if such a middleware is standardized, it will allow users to have vendor-independence and the migration of their applications from one middleware solution to the other one.

The primary intent of this paper is to address the architecture, design and implementation aspects of the middleware underlying such an information system. Accordingly, our efforts will target those software components that will mediate NCO applications. We will characterize the set of capabilities that such a middleware should implement. Then, we will explore the architectural and design aspects as well as the quality attributes that should be considered in the deployment of such middleware.

Information systems have become the cornerstone that shapes the present and the future of our economy and society. They have penetrated almost every aspect of our lives. This omnipresence is stemming from the significant advancements in computer systems, software middleware and applications, telecommunication infrastructures, etc.

Enterprises and organizations use a large variety of networked computer systems to collect, process, produce, and store large volumes of information. Geographically distant computer systems are disparate in terms of hardware, platforms, operating systems, and software applications. This heterogeneity limits the scope of interchanging information inside and outside the organizational environment. On the other side, the required services to interpret the aggregated data are mostly available locally to the user’s terminal. Moreover, decision makers, top managers and leaders need a complete and efficient software infrastructure that can continuously subscribe to up-to-date local and global services and information, always synchronized to the real-time changes. In this paper, we address this high priority organizational interest to provide more competitive advantages in mission-critical situations.

While a surge of interest has lately been expressed separately on the data integration and web services technology, only a few research has been so far conducted to propose a general architecture to support integration of information systems. Moreover, there exists different heterogeneous information system architectures related to specific domains in the enterprises, with several restrictions and policy controls. We focus on a loosely coupled distributed paradigm that offers a message-based middleware, able to integrate data and services from different industrial infrastructures.

We also include in this paper an implementation of the proof of concept, referred as “Digital Cockpit”. The intent of the digital cockpit is twofold: First, it achieves a synergistic integration of the various information sources. Second, the digital cockpit displays visual, structured, navigational, and real-time big pictures, so that decision makers can drill down into the details and consequently supports enhanced decision making process using the available information. The digital cockpit uses cutting-edge visualization technologies that will highlight and analyze complex relationships, patterns, and trends.

More explicitly, the objectives of this paper may be classified as the following:

- Identify different information and service sources of the organization and the underlying databases, data models, formats, and protocols.
- Explore the dependencies and relationships between these information sources.
- Integrate all these sources of information in order to ensure a real-time availability of updated, consolidated, structured, and unified data across the networks.
- Understand the security requirements.
- Elaborate a platform that will present the synthesized information in a dynamic and real-time visual object with the ability to drilling down the information details.
- Finally, illustrate an implementation of a suite of services and tools that allow analytical processing and optimization techniques (statistical analysis, graphical techniques, simulation of what-if scenarios, trend analysis, comparative analysis, etc.) on the gathered data to disclose the enormous potential of the information systems integration.

The rest of the paper is structured as follows: in Section 2, we present the related work and discuss the scopes and limitations briefly. In Section 3, we describe the proposed architecture in details. Section 4 outlines the security requirements and the proposed solution. In Section 5, we detail the adopted methodology for the implementation of digital cockpit model. Section 6 contains the conclusion and possible future work.

2. Related work

Middleware-based information integration has been massively empowered by the large-cap corporations in the last few years to suffice their business need for information and
service sharing. This section focuses on the contributions of the previous researches on data and service integration and message oriented middleware. It also includes a brief elaboration on business intelligence and visualization techniques, used to transform data into useful patterns. These techniques are related to the decision making process, and they significantly impact our decision support system.

2.1. Data integration

Data integration approaches work on a set of decentralized data, develop a single unified schema and allow a series of transformation or source mapping to describe the relationships among each data source and mediated schemas. It can be achieved in different ways: Data Warehousing (Materialized Views), Custom Integration code (In House Development), and Federated Architecture (Virtual Views).

The warehousing solution requires the building of a physical centralized database – a subject-oriented, integrated, non-volatile collection of data in a single database – and writing programs to load the data from the data sources to the warehouse periodically via ETL (Extract, Transform, and Load) tools. This approach is relevant when data does not change frequently or when the integrated view does not need to be up-to-date. Hence, the full contents of the global schema are pre-computed and stored in a separate database. However, using ETL tools, refreshing the warehouse is typically expensive and generally done off-line.

The custom integration approach or “in-house” development – chosen at the beginning of the integration process – seemed to be an interesting and economical option, but gradually becomes a heavy-burden for many reasons such as: not well-documented or not documented at all; does not take care about the scalability of the system since it was done to fix temporary problems, etc. Thus, this solution itself turns into another legacy application.

The other approach for data integration is the federated architecture. This approach defines one or more mediated schemas, which enquires the data-sources, however keeps their local autonomy and creates a virtual repository enabling real-time on-demand data access. This approach is well suited and almost inevitable in large modern enterprises. In fact, the different parts of the organization use different systems to produce, process, store, and search their critical data. Yet, it is only by combining the information from these various systems that the enterprise can realize the full value of the data it contains [7]. Moreover, the federated architecture fits well when the global schema changes frequently, which is a real need to make educated decisions based on up-to-date information. Fig. 1 illustrates the federated architecture approach.

Initial implementations of these approaches are mostly suffering from high-level vendor lock-in, inextensibility and other dependencies. Afterwards, the standard Java APIs came as Java Database Connectivity (JDBC) [9] that allows connection to relational databases. Once the connection is established, the fetched data are sent to their requestor via a communication protocol in a tightly coupled and synchronous environment. In the next step, the asynchronous communication was addressed with the release of Java Message Service (JMS) [7], enabling loosely coupled and reliable communication. However, IT community continued the research to integrate the legacy systems. A survey shows that 50% of the software market expenditure goes to integrate legacy applications, and 80% of worldwide data is stored in mainframes [12]. Finally, J2EE Connector Architecture API (JCA) is released recently as an open standard solution to access almost all the standard Enterprise Information Systems (including legacy systems) but still in a tightly coupled synchronous/asynchronous way.

![Data integration: Federated Architecture Approach.](image)
2.2. Service integration

Retrieval of information is not enough to design a complete decision support solution in the absence of service sharing.

Initially, technologies like electronic data interchange (EDI) [11] have been used for many years to successfully perform business transactions between partners [4]. EDI relies on formatted messages (based on defined standards) and proprietary network protocols for data transport. Companies have been reluctant to invest in these technologies, because of the large underlying investments that are required in terms of software, hardware, and consultancy [4]. Afterwards, with the advancement in telecommunications and due to high organizational concern of computing industries, in the early 90s; many initiatives have been introduced to facilitate the communication between application components in a distributed computing environment. Common Object Request Broker Architecture (CORBA) [10] from Object Management Group (OMG), Distributed Component Object Model (DCOM) [2] from Microsoft and Remote Method Invocation (RMI) from IBM and SUN Microsystems are examples of these initiatives. All these technologies allowed organizations to integrate applications in a distributed infrastructure using RPC-based mechanisms to bind application clients to a server [4]. However, interoperability between these RPC-based mechanisms is still complex and limited. For instance, CORBA and DCOM can not communicate easily and may need a bridge to allow such a communication. This limitation is due to the fact that each infrastructure uses its own communication protocol (IIOP for CORBA, ORPC for DCOM, and JRMP for RMI). In addition, within these technologies, the application is statically bound to a single address and tightly coupled with request/response mode [6].

Finally, with the emergence of XML (Extensible Markup Language) as a cross-platform, extensible, and text-based standard language for representing data [11], the service integration is leaned to the Service Oriented Architecture (SOA). Hence, the concept of web services was introduced first by HP in its product e-Speak, in 1999 [8]. Shortly afterwards, many competing frameworks and proposals for web services were provided such as Microsoft .Net, IBM webspheres and SUN’s J2EE web services. The SOA vision is based on a four-layered architecture namely, (a) object-based application-level communication protocol, (b) description of web services, (c) discovery of web services and (d) execution of web services. The release of Extensible Markup Language (XML) based schema representation eased the language of web services and consequently the standard technologies were mostly defined with the similar formats in the last five years. Simple Object Access Protocol (SOAP) for web services protocol, Web Services Description language (WSDL) for service description, Universal Description, Discovery, and Integration (UDDI) and Electronic Business Extended Markup Language (ebXML) for service discovery, and finally Business Process Execution Language for Web Services (BPEL4WS) in service execution have been proved as standard technologies to implement the notion.

2.3. Real-time data processing

Analysis and visualization of the integrated data are inseparable parts of the integration middleware. Nowadays they are quite sophisticated than the so-called data analysis (as done in MsExcel) and user-interfaces. Industrial research on business intelligence (BI) based analysis is intensely motivated by the Online-Analytical Processing (OLAP) technologies and its different variations, such as: MOLAP (Multi-dimensional OLAP) and ROLAP (Relational OLAP). The role of the OLAP tools is to analyze, report and show trends and to generate relevant information that may be directly used to the decision-making processes. Using BI tools like OLAP, very complex analysis and synthesis capabilities can be incorporated that clearly outweigh traditional queries and generation of reports. Visualization, on the other hand, is favored by lot of commercially available APIs (e.g. EspressChart from Quad-base) that eases the deployment with applets, servlets, JSPs, and Java applications. However, the middleware requires a new dimension of web-based display technologies that must be activated with and without user-intervention and should view graphical representation of fresh information as it is better understandable than tedious tabular display of piles of data.

3. Architecture

The present world of information technology is running on a multi-tier architecture that fills the desideratum between the users and the sources of information and services. The choice of a proper middleware among them will significantly speed up the retrieval of data and/or services, reduce integration complexity and increase the performance as well. Architecturally, the proposed middleware represents a distributed structure on the top of existing client–server architectures as available within intra-departmental networks.

This middleware model abstracts the information sources and helps the user to get information without caring about their locations. Since the solution is assumed to address the data that changes frequently with time, the model is designed on federated architecture which is capable to retrieve present values of data substantially reduces the integration complexity. Second, the model takes care of automatic notification to the dependent representation of the same information in local or remote client machines. Actually, we leverage a Message Oriented Middleware (MOM) that allows the following capabilities:
• Autonomy of information: The model has the ability to publish information as per user’s wish within a particular domain. Two organizations may communicate through messages that are loosely coupled with their own domain of environments and share information as per their permissions previously agreed between them.

• Real-time sharing of information: MOM architecture allows real-time display of the changing information, one of the most important objectives of organizations.

• Performance: The reliability of the message communication is high and the architecture supports asynchronous communication. Therefore it does not stall the processes at the user-end.

• Scalability: The model supports easy incorporation or deduction of an organization into the infrastructure and removes the necessity of tedious mapping of all its information sources using tightly coupled technologies.

• Security: MOM allows the security modules to be imposed more freely than the RPC-based communication models. It allows the privacy of the message as well as supports the data encryption and certificates mechanisms for secured communication.

Fig. 2 shows our layer-based integration approach. It consists of four principle layers: resource, integration, acceleration, and presentation tiers. The resource tier – the lowest layer – is responsible for the persistent storage of data, and holds multiple heterogeneous sources of information such as relational databases, spreadsheets and legacy applications. The integration tier is responsible for application servers that provide adapters for information sources. For example, in case of J2EE technology, this includes JDBC and JCA. The MOM has the main role in this layer, it receives data provided by the integration tier and pushes them to the acceleration layer. This acceleration layer is associated to each client of the system and contains a receiver of information, called ‘subscriber’. The subscriber integrated information inside the client’s side as per user’s choice. It relies on a purely message-based system and conveys this information to the presentation layer after processing, if needed. This last layer consists of a set of web-based components that is responsible for the display of structured, graphical and navigational representation of the information fetched by the other tiers.

Finally, Fig. 3 depicts the architecture of the middleware for integrating the data sources. Here, we have mentioned some relevant technologies that support our architecture. The model illustrates the communication between two departments having clients and databases local to their environments. Initially the architecture allows a client to retrieve information locally or remotely according to his/her privileges. The model includes an integrator component to facilitate the fetching of data. We propose a semblance of message-based point-to-point asynchronous inter-organizational communication as supported by Java Message Service (JMS) and a tightly coupled retrieval of information inside the department using JDBC and/or JCA technologies for the purpose of retrieval of local information.

In the beginning client system requires huge amount of data to start a component. The infrastructure provides a direct peer-to-peer communication mechanism (e.g. JMS-Queue) between the requested client and related server/s. Once the client system starts displaying the retrieved information, it only requires to gather real-time notification of changes. With the update of data, once again, integrator will automatically trigger a publisher module to publish information in a common namespace named “topic”, a JMS publish-subscribe component. On the other hand, the client-end software solution invokes a listening process in the background and listens to the changes published in
the required topic. As the changes are found, they receive the information as a message and update the visualization of the information in client’s side.

The performance is expected to be high as the network deals only with the change of an information. The following formal equation clearly prove the superiority of message-based middleware over the tightly coupled RPC applications.

Suppose an application has $N$ threads and $r$ requests on average to the resource tier. If each request takes $t$ seconds to complete and the pre-request works of threads require $w$ seconds then an RPC-based application manages $N$ simultaneous threads on average $(r \times t + w)$ seconds for each of them. So, the time taken by the system to serve $R$ requests is:

$$time = (R \div N + 1) \times (r \cdot t + w);$$

where $\div$ is the integer division without remainder.

For message-based middleware, as proposed here, each thread can simultaneously send multiple requests and waits for the later tier by the amount of time [1]:

$$\delta = \text{Max}_{k=1}^{r}(t_k)$$

Since the time thread is simultaneously servicing different requests while it is waiting for other tiers, we assume that a single thread is servicing $c$ ($c \geq 1$) simultaneous requests, and the time required to serve the same $R$ requests will be [1]

$$time = (R \div c \times N) \times \delta$$

The result clearly shows that message-oriented middleware takes less time than the so-called RPC-based data integration mechanisms.

Data integration alone does not offer a complete system as the data always require a service to be processed in a meaningful way. Use of remote services via intranet/internet is vital as the user often requires lightweight services to synthesize data while the services are not available locally to his/her computer. However service integration infrastructure is typically complex in the global architectural point of view. The Fig. 4 intends to highlight the phases of web-service integration.

Here we describe the service integration with the available APIs before establishing our proposed architecture. Every web-service requires a complete description of its objects and functions. The intention here is to define common data-interchange formats for each of the needed transaction. This may be achieved by the use of Web Service

| Service Execution | - Defines the states of transactions between two services.  
| Service Discovery | - Defines a list for organization's capabilities and the business transactions.  
| Service Description | - Defines the capabilities of a service.  
| Service Protocol | - Defines a common transport mechanism for exchanging messages between organizations.  

Fig. 3. Architecture of data integration.

Fig. 4. Four phases of web services.
Description Language (WSDL). Actually, WSDL is the most promising language and the emerging standard. As of the technical logistics for WSDL, we used Axis API from the Apache foundation [5].

Concerning the discovery of services, the most popular technology is to adopt a registry/repository mechanism for allowing the dynamic discovery of the set of available services. Once a service is registered, i.e. added to the registry, the other departmental clients may access that service in a remote server provided that their requests must match with all the conditions stated in the provider’s service description. Dealing with different registries requires a common interface as provided by the Java API for XML Registries (JAXR API) for the java implementations. This API is an open-standard provided by Java Community Process (JCP), and supports both of the standard registry repository systems: UDDI and ebXML (Electronic Business Extensible Markup Language). However, WSDL is a stateless language and therefore it is not able to capture the execution of business processes. IBM and Microsoft jointly proposed BPEL4WS while Sun released WSCI, two different languages in order to model the process execution while the service is described before in WSDL syntaxes. As of the service protocols, the proven W3C open standard is Simple Object Access Protocol (SOAP) [3]. SOAP is essential to carry out the XML-request and the service description from one location to another. As we discussed before the AXIS API supports SOAP messaging over different communication protocols (e.g. HTTP/S) and messaging services (e.g. JMS). Fig. 5 depicts a general architecture of the service integration, as proposed.

4. Security

As the proposed Message-Oriented Middleware (MOM) seamlessly allows connecting applications and sharing information, on the other hand, it raises critical security issues that must be taken care to conform a secured and trustable architecture. Actually, the necessity of the distributed nature for information integration complicates the solution of the security requirements more than the so-called certificate-based secured software system. This section leverages a detailed discussion on the security requirements of such a loosely coupled middleware and an idea of possible solutions.

First of all, this architecture requires to keep the local autonomy of the information and users of the system should be bound to the organizational scope. More precisely, each department in the architecture has its own users and sources of information maintained by it. This idea is directly inherited from the generic business model of any organization. However, it suffers to identify an authentic user outside the scope of own department/organization but within the system. The available security infrastructures work well within the scope of an institution in order to identify a local user but unable to recognize a permitted user from other organization due to the lack of information structure. The research establishes a need of predefined
handshaking between two organizations in terms of certification policy and public/private key pair that allows identification of one institution to another and thereby scopes to identify the inter-organizational users.

The implementation of such multi-tier end-to-end information integration must be capable for keeping the sharing of data secured among the set of right privileged users. We have implemented this idea through a role-based access control system on the top of this architecture. Each of the client terminals should be provided a list of permitted components he/she can view. This list should be generated depending on the role of the users to the system and any change (addition of new components) of this access-control list always requires the intervention of the local server that assigns and keeps the list in its local storage for security purposes.

The secrecy of information is one of the main perspectives from the security viewpoint. This architecture supports publishing the information in the name-space (e.g. JMS Topic) local to the departmental server. Therefore, it allows the department to enforce stronger security mechanisms over the ‘topic’, the namespace for publishing information. It is, of course, a better approach than publishing remotely outside the scope of the department. Although, the details of these security mechanisms are out of the scope of this paper however the proposed architecture supports all kinds of authentication, authorization techniques on this name space in order to share the information only among the permitted users. Even the information may also be classified to be accessible to certain group/s of users, while the others cannot even read from the ‘topic’.

All the events and activities related to the architecture (for example: sending information to an external client) within the department must be logged for future traceability. Certain activities often have serious impact on the organization, however cannot be identified later if not tracked. In the implementation of the system, we have successfully developed a database dedicated to track these strategic moves that keeps the track of user events.

Finally, we claim that our architecture supports robust security infrastructure on the top of a distributed environment. It allows users to send and receive real time view of information as per his/her credentials in the integrated architecture. Moreover the distributed nature of the architecture reduces the possibility of denial of services attack, therefore increases the availability of the system. We dealt the security at the level of messaging services, therefore did not explore the sockets, and tunnelling through the firewalls. In the implementation, proper choice of messaging services and the communication protocols will take care of the communication of messages in the inter-organizational Business-to-Business Integration (B2Bi) level.

5. Implementation

This section focuses on the high level design of a project, called “Digital Cockpit”, that realizes a significant part of the aforementioned architectures. The project is fully funded by the Defence Research and Development, Canada. Actually the digital cockpit project accomplishes an efficient and useful decision support system mainly based on the information and service integration. In this discussion, we also intend to establish the huge potential lies in the fusion of the two explicit research areas: information integration and decision support system. The underlying design and implementation phases of the developed middleware are summarized below.

- **Integration**: to connect all the information and service sources within and across the organization, for an information sharing purpose.
- **Display**: to take the data from different sources, aggregate them and present the synthesized information into a meaningful, structured and big navigational “picture” that offers the ability to drill down into the details.
- **Monitor**: to design and implement the capabilities that allow the active monitoring of the information system state for the purpose of testing the organization’s assumptions, reactive and proactive measures, and response to dashboard thresholds, etc.
- **Analyze**: to bring the system to the business intelligence level, i.e. to design and implement the capabilities for pattern and trend analysis, simulation of “what-if” scenarios, etc.
- **Control**: to optimize procedures, events, and scenarios that will enhance the used processes, methods, and strategies.

Security is not an exact phase of this paradigm, however considered as a vital quality attribute that must be taken care of at each phase of this paradigm. Fig. 6 depicts the ideology more clearly.

The integrator is in charge of gathering the required information by interacting with the different data sources as required by the user. It retrieves from the local sources directly as well as uses Java messaging technology in the case of remote sources. There is a subscriber module, proposed at the client’s end that is used to express interest in the needed information and therefore open a new connection depending on user privileges. Using JMS point to point subscription mechanism, the subscriber is also able to provide the end-user with the capability of subscribing to the information of interest directly from the remote data sources. On the other hand, if the digital cockpit client updates any information in the data sources, the integrator interface associated to the sources on server side will trigger a publish mechanism that uses a common JMS topic to publish the necessary changes as mentioned in Fig. 2. When such information is produced in the topic the messaging service notifies the relevant subscribers and supplies fresh information to them. The monitor module is in charge of tracking the organization’s assumptions with this change of information and responds actively to dashboard thresholds without any direct user-intervention. The client
module also contains some local services to analyze and optimize the different scenarios comprised with the aggregated data. The analyzer module is responsible for performing the needed simulation, pattern and trend analysis, while the controller is the module that optimizes the different scenarios and processes as required by the user. Finally at the user-end, the display manager takes care of the graphical layout and the production of structured and navigational “big picture” of the system. From the experience we understood that the client module requires a consolidated security foundation in order to be deployed commercially. We implemented a security manager module that enforces the required security properties such as authentication, authorization, secrecy, integrity, non-repudiation, and availability. The architecture of a digital cockpit client is made of the following modules: the integrator, the subscriber, the display manager, the monitor, the analyzer, the controller, and the security manager.

The issue of the service integration is lately added as an extended capability in this project. We extend the integrator module that uses “hook” to connect the light-weight services situated in the remote servers. With the limited scope of the project we exclude the idea of implementing a general registry to register web services and call the service directly using JMS on top of http/https communication protocol. As mentioned before, AXIS is used through out the implementation. This API successfully binds the user request with the services defined in WSDL. However, for the global architecture of service integration, the use of a registry is inevitable as the location services are mostly unknown.

In what follows, we explain, the underlying design and implementation in different phases of the digital cockpit to explain the implementation strategies briefly. However, the detailed description of the digital cockpit is beyond the scope of this paper.

5.1. Integration

Integration of information systems is the key issue for designing the digital cockpit prototype. We defined interfaces and communication protocols of this message-based middleware in three phases: data integration, service integration, and messaging service.

Data integration is related with the retrieval of data from the structured, semi-structured, or unstructured sources of information inside and outside the organization. Moreover, they are heterogeneous in term of data storage, query procedures, operating systems, and the type of network. We used two free Java APIs: Java database connectivity (JDBC) and the J2EE Connector Architecture (JCA) Java APIs for the querying of information sources. Actually, JDBC is the de facto API, used in accessing relational databases. JCA is suggested mainly to access legacy systems (e.g. mainframes), unstructured (e.g. flat files), or semi-structured (e.g. web pages) information sources. The rationale behind this recommendation is that JDBC has shown its merits in terms of performance, convenience and ease of use. As of JCA, it is the API of choice since it allows accessing legacy systems. Besides, JCA caters for an asynchronous mode of fetching information and a strong pooling mechanism that significantly improves scalability. On the other hand, using XML documents, we are in a position to have multiple views on synthesized and/or aggregated data.

The digital cockpit platform is endowed with the capabilities that cater for web-based service integration. As per the need, we implemented the design using Axis API from the Apache foundation [5]. Axis receives SOAP request, and binds the respected service if there is a match between the request and the WSDL description of the service itself.

As of the messaging service, we implement the idea with Java Message Service (JMS) API [7]. JMS is the messaging standard in Java for the asynchronous information sharing as required for the architecture of digital cockpits. Again, the choice of JMS is also motivated by: cost-effectiveness (freely available API from JCP); Standard (Java Specification Request 914); Reliability (guaranteed delivery); Performance (outperforms other communication solutions such as RMI and RPC).

5.2. Display

As mentioned before, display of digital cockpit is more sophisticated and complex than a classical, conventional and static graphical user interface (GUI) as it collects the frequent changes of information and notifies the user in almost real-time. The display of such synthesized information is also required to be performed in a meaningful structured, navigational, and graphical way. We incorporated dynamic graphical objects and representations such as charts, curves, knobs, histograms, reports, animated maps, etc. The user can also customize the display according to his privileges: adding/removing (a) component(s), changing...
its properties, etc. Furthermore, the display screens provide the user with some utilities that operate on screen elements such as: saving, zooming, navigating, etc. We used Espress-Chart API from Quadbase Inc.

5.3. Monitoring

This phase reflects the active monitoring of the various organizational system state. It allows the testing of the organization assumptions, proactive and reactive measures, and responses to the cockpit thresholds. We introduced java listeners as well as suitable APIs according to the nature of the action or reaction. For example, Java Mail API was used for sending e-mails.

5.4. Analysis

The main intent of this phase is to design and implement a suite of procedures that will endow the digital cockpit client with analytical capabilities like pattern identification, simulation of what-if scenarios, trend analysis, probabilistic and statistical analysis, cause-effect analysis, etc. Actually, such capabilities will bring the digital cockpit to the business intelligence level. Implementation of the analytical capabilities is done using various Online Analytical Processing (OLAP) tools.

5.5. Control

This phase offers optimization of business processes, methods, strategies, and scenarios to the digital cockpit, that increase the productivity of decision makers by introducing automated decisions for the organizations. Optimization problems are known to be hard. Generally, some key techniques-like stochastic processes with techniques-like operation-research and game-theory are used for precise decision making.

5.6. Security

The digital cockpit model requires mechanisms to enforce security properties such as authentication (proving identities of the cockpit users), secrecy (not leaking sensitive information to inappropriate users), integrity (preventing data from corruption and alteration), non-repudiation (preventing users from denying their execution of cockpit operations), and authorization (giving only authorized users the right to access information and to execute services).

We wish to deploy the required technologies in the digital cockpit according to the required security level. We implemented a security infrastructure using the JSSE (Java Secure Socket Extension) API with X509 certificates. As of non-repudiation, it can be enforced by having adequate audit and logging mechanisms. The Java logging API seems to be appropriate for this need. However, the logs themselves need to be protected. Concerning integrity, we adopted encryption and hashing in order to detect any illegal modification of sensitive data. Authorization could be implemented by using JAAS (Java Authentication and Authorization Services) API.
Within the scope of that prototype, we integrated the information from a few number of remote information sources (parts of other digital cockpits) into the digital cockpit model running on the top of J2EE application server. Fig. 7 shows an user-interface of the implementation.

6. Conclusion

This paper presents a middleware architecture for a net-centric information system decision support environment. The proposed design includes an infrastructure of web services and provides capabilities for data/information/service integration, visualization, monitoring, analysis, and control. Multi-layer solution and implementation have been described in details. Accordingly, information sources dependencies, relationships and integration are explained, security issues are briefly discussed, and finally services and tools to exploit data and information and perform a variety of analysis are depicted.

Future work will address applications for targeted military problem domains. It will consist in first achieving data/information integration and then provide some visualization (asset visibility), monitoring and analysis capabilities through application (including legacy) integration and web services composition. Suitable solution will be developed and explored to meet challenges posed by service composition and system performance limitations. An assessment on the value of net-centricity on performance for planning tasks will also be carried out for selected military domains.

References