Military Technical College

Kobry El-Kobbah,

Cairo, Egypt

8th International Conference on Electrical Engineering

ICEENG 2012

A COMPARATIVE STUDY BETWEEN SOME DESIGN ISSUES IN DISTRIBUTED SYSTEMS MIDDLEWARE BASED ON WEB SERVICES AND THEIR IMPACT IN FUTURE CHALLENGES

Ahmed M. Matar [*], *Khaled M. Badran* [*], *Moatassem M. Abdallah* [**]

[*] Department of Computers, M. T. C., Cairo, Egypt
[**] Egyptian Armed Forces

ABSTRACT

In this paper, we focus on some design issues in distributed systems Middleware based on Web services,and their impact in future challenges. Middleware performance, Scalability, Management and Ubiquity are considered some of the most elite challenges facing the designers of future middleware systems. Middleware performance can degrade at some point, the scalability of the distributed application can be difficult to control, managing large heterogeneous applications arises many questions, and finally, mobility and dynamic reconfiguration of applications forms the ubiquity challenge. As a guide for middleware designers, we propose this comparative study between different well known middleware systems considering our proposed design features as points of comparison, besides analyzing their impact on the previously mentioned challenges.

Index Terms— Database systems, Middleware systems, Web services, Naming, Binding

1. INTRODUCTION

1.1. Web Services and Middleware

Many people and companies have debated the exact definition of *web services*. At a minimum, however, a web service is any piece of software (API) or more specifically a Web API that's available over the Internet, identified by a URI, executed on a remote system hosting the requested services. Web Services support direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols (HTTP famously used).

A popular interpretation of Web services is based on IBM's Web service architecture based on three elements; *Service requester:* The potential user of a service, *Service provider:* The entity that implements the service and offers to carry it out on behalf of the requester, *Service registry:* A place where available services are listed and which allows providers to advertise their services and requesters to query for services.

The Web service protocol stack is an evolving set of protocols used to define, discover, and implement Web services. Based on the previously mentioned architecture, the core protocol stack consists of four layers; service transport layer, responsible for transporting messages between applications, XML messaging layer, responsible for encoding messages in a common XML format, service description layer, responsible for describing the public interface to a specific Web service, Service Discovery layer, responsible for centralizing services into a common registry.

Now that we explained the Web service protocol stack, let's have a quick overview of the mapping of these layers in real life. Simple Object Access Protocol(SOAP) is a protocol based on the exchange of information between multiple computers in the form of XML messages, the main advantage of SOAP is its delivery via many transport protocols, the main focus of SOAP is Remote Procedural Call (RPC) transported via HTTP. SOAP is platform independent, and therefore enables diverse applications to communicate with one another. Web Service Description Language(WSDL) represents the service description layer within the Web service protocol stack containing XML grammar that enables the description of the public interface to a web service. Universal Description Discovery and Integration(UDDI) represents the discovery layer within the Web services protocol stack. UDDI is a technical specification for building a distributed directory of businesses and Web services. At its core, UDDI consists of two parts. First, API details for searching existing UDDI data and secondly, API details for publishing new data.

Web Services Design Methodology, there are two ways to design a web service; Bottom up method, implementing the working class in a programming language then finding a WSDL generating tool to expose its methods as web services. Top down method, implementing the WSDL document then using a code generating tool to build up the structure of the class, which is to be completed by the web service developer.

The term middleware first appeared in the mid 1980s to describe network connection management software. The term again gained its popularity in the late 1980s as a solution for the problem of linking newer applications to older legacy systems [1]. The appearance of the term in the late 1980s does not deny the fact that middleware systems existed long before that, a live example would be Messaging systems that were available as products in the late 1970s, along with the classical reference on Remote Procedural Call implementation in 1984 [2].

Starting in the mid-1980s, a number of research projects developed middleware support for distributed objects, and elaborated the main concepts that influenced later standards and products. Early efforts are Cronus (1986) and Eden (1985). Later projects include Amoeba (1990), ANSAware, Arjuna (1995), Argus (1988), Chorus/COOL (1993), Clouds (1989), Comandos (1994), Emerald (1988), Gothic (1991), Guide (1991), Network Objects (1995), SOS (1989), and Spring (1994).

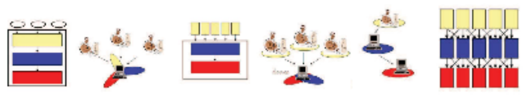
In literature, there are multiple definitions for the term middleware, ranging from a software layer between the operating system (and/or the network) from one side , and applications from the other side; to connect (glue) two applications together [3]. ObjectWeb [4] defines middleware as ”The software layer that lies between the operating system and applications on each side of a distributed computing system in a network”, while Wikipedia defines middleware as ”The computer software components or people and their applications that consists of a set of services allowing multiple processes running on one or more machines to interact”. A logical definition for middleware is ”Middleware is the intermediate software that resides on top of the operating system and communication protocols to perform a specific function; whether this function was; hiding distribution, hiding heterogeneity, providing uniform standard high-level interfaces to the application developers and integrators, or supplying a set of common services to perform general purpose functions”.

In order to understand Web Services, we need to take a look at the way middleware and enterprise application integration technology has been evolving in the last decades. Only then, we will be able to understand Web Services. We are going to give a quick explanation starting from the one tier architecture to the middleware.

First, we have the fully centralized Architecture, the one tier (Fig. 1a). Where users and/or programs access the system through display terminals (dumb terminals) but what is displayed and how it appears is controlled by the server. This was the typical architecture of mainframes offering the advantage of centralized managing and controlling of all the resources.

Second, we have the client - server architecture, the two tier (Fig. 1b). This architecture evolved from the fact that computers became more powerful, this allowed the possibility of moving the presentation layer to the client's machine. Clients are now independent of each other, one could have several presentation layers depending on what each client wants to do. This architecture offered the advantages of using the computing power of the client's machines to create more sophisticated presentation layers, besides introducing the concept of application programming interface (API) that allowed invoking a system from the outside.

Finally, we have the simple middleware, three tier architecture (Fig. 1c). Starting from this architecture, any application can be called a middleware. This architecture is just another layer of indirection between clients and other layers of the system, introducing an additional layer of business logic working with all underlying systems.



(a) One Tier (b) Two Tier (c) Three Tier - Middleware

Fig. 1: Layers and tiers

Motivation for the use of middleware comes from evolving access to information and computing resources from all across the globe into a utility, like electric power or communication grids. On the journey of achieving this goal, designers and developers of distributed software applications face some solid problems in their practical life, like reusing legacy software, managing mediation systems, developing component-based systems, and adapting client communication through proxies. One can summarize the previous problems into four challenges facing the future design and development of middleware, which are; Performance, Scalability, Ubiquity, and Management.
As a consequence to the mentioned challenges, it was an obvious move to analyze the architectural aspects of middleware design, concentrating on those aspects related to the design of distributed systems. In this paper, we focus on the aspects of naming, binding, coordination and security as they are the center in middleware design. The remainder of this paper is organized as follows. Section 2 provides a short background and related work discussing some real world applications using middleware, and some analyzed middleware systems. In Section 3, we discuss some issues (requirements) in the process of middleware design and in Section 4 we present a comparison in the form of a table between the analyzed middleware systems and the discussed design requirements as points of comparison. Finally, Section 5 concludes the paper with the proper analysis and guidelines regarding the future aspects in middleware design.

2. BACKGROUND AND RELATED WORK

In this section, we provide a short overview on two middleware sets; real world applications currently using middleware, and middleware systems found in literature. We will use the latter as our case of study.

2.1. Real World Applications using middleware

During the nineties, The TecBD laboratory at PUCRio developed HEROS: a Heterogeneous Database Management System (HDBMS) [5] [6] [7] [8] [9] [10], which allow the integration of heterogenous database systems into a federation so that queries and updates could be submitted transparently to the data location, to local access paths hiding any existing heterogeneity. Inspired by HEROS, the SINPESQ project [11] was born, adopted by the Brazilian ministry of environment, whose objective was to integrate fish disembarkation data from all the fishing colonies spread through the Brazilian shores, lakes and rivers. Another real life application is the ECOBASE project [12], whose objective was to share experiences in information integration environments. Then comes the ECOHOOD project [13], whose objective was to build a configurable DBMS adapted to specific application domains, the point here was to model systems as an object oriented framework. Last but not least, the SKYQUERY project [14], which is a functional prototype used nowadays to federate astronomical data from different heterogeneous databases using the cross-match query, its objective was to collect many details for multiple astronomical bodies from different archives all over the world.

2.2. Analyzed middleware systems

1. **The Araneus Web-base management system** [15] is a system developed at the University of Roma Tri. The main objective of the system is to handle both highly structured data, in relational or object oriented database systems, and semi structured data, in web sites. In order to accomplish this task, the system consists of three main modules, each module is dedicated for a specific job. The most important component in the system that co-ordinates between the three modules is the ADM Object Manager.
   First, the DBMS interface which uses the standard SQL-based protocol JDBC to communicate with a (remote/local) DBMS. Accessing the DBMS allows the manipulation and storage of structured data, although the DBMS is not part of the system, but the system relies on the DBMS to handle tables.
   Second, the ULIXES language which is used to query web sites by using a Navigational Algebra. Querying a web site outside the control domain of the system requires an ADM description through the analysis of the site's content. Then wrapping the system to extract semi structured data from the pages of the web site using a suitable wrapper from the wrapper library. EDITOR and MINERVA tools are used to generate the required wrappers because they allow searching and restructuring of semi structured documents.
   Finally, the PENELOPE which is used for creating and maintaining new web sites inside the control domain of the system. ADM views are defined using PDL (PENELOPE Definition Language) and maintained using PML (PENELOPE Manipulation Language). The Module supports both the pull and push techniques for web page creation.

2. **Strudel: A Web-site management system** [16] is a system developed at AT&T Labs. The Main objective of the system is also to handle structured and semi structured data like Araneus. The Main differences between the two systems are, that STRUDEL provides a tool for the integration of data from different sources because it does not require the storage of all data in one single repository, in each level in STRUDEL data is viewed as a graph. The system consists of four main modules.
   First, the Mediator accepts data from different wrappers and generates a data graph, the wrappers transforms external source's data into STRUDEL objects and collections understood by the mediator and translates STRUDEL queries into jobs understood by the source. If the sources contain semi structured data the mediator communicates with the Query Processor
   Second, the Query Processor whose main function is to generate query execution plans for both structured and semi structured data.
   Third, the STRUDEL's Site Management that receives the data graph from the Mediator along with the query execution plan from the Query Processor and forms the site graph. The STRUDEL data graph is the same as the site graph but the latter contains information on how to display each node's contents in HTML. Finally, the HTML Generator that transforms a site graph into a browsable graph of HTML pages.

3. **Le Select: A Middleware System for Publishing Autonomous and Heterogeneous**

Information Sources [17]is a system developed at INRIA, which is a framework prototype whose objective is to access data of heterogeneous nature and to invoke data processing programs over Internet/intranet environments. Le Select approach is similar to STRUDEL in the cases of fully distributed architecture(No global schema and No global repository) , and connection to multiple data sources through Java-based wrappers with additional XML definition files.
Data can be viewed in Le Select through a standard web browser, in the form of tuples in a relational database. Regardless of the native format of the data, wrappers transform the data into a relational format that can be viewed by users, then through the mediator several connections to the wrappers are made, and integration of the received data is then made in the form of tuples.

4. **Scaling access to heterogeneous data sources with DISCO** [18] is a system also developed at INRIA. The main objective of the system was to focus on the solution of three problems; fragile mediators, weak data sources, and graceless failure for unavailable data sources.
The first problem, fragile mediators, faces the Database Administrators (DBAs). For proper execution of a mediator, the DBA must define the mediator schema, in addition to some data sources and their local schemas. Finally, mapping the mediator schema to the data source's local schemas via database views. The problem here lies upon the introduction of a new data source, a new local schema should be added, and accordingly the view definition must also change to cope with the new data source. In some cases, the schema of the whole mediator have to be changed in order to accept this new data source. DISCO handles this problem by presenting the system architecture, that permits DBAs to develop and implement mediators independently, limiting the impact of addition of a new data source to mediators.
The second problem, weak data sources, faces the Database Implementors (DBIs). When an application sends a query to the mediator to be executed, the mediator transforms it into multiple sub-queries each corresponding to wrapper controlling a data source, along with the production of a certain composition query. The wrapper receives the sub-query, translating it and sending it to the data source for computation. Then the wrapper receives the answer, transforming it to the proper format and sends it back to the mediator. The mediator along with the produced composition query integrates the answers received from the different wrappers, then the answer is sent to the calling application. For proper query execution, the mediator must cope with the multiple functionalities of each wrapper, whether it supports projection or selection operations ... etc. DISCO copes with this problem by providing a wrapper language and a wrapper interface to ease the construction of wrappers with their specific functionalities, the mediator then must issue a sub-query to the wrapper within the domain of its functionalities
The third problem, graceless failure for unavailable data sources, faces Application Programmers (APs). In the absence of replication schemes, if a sub-query is issued to an underlying data source that's not available, the mediator fails to process the query. DISCO provides new semantics for query processing if a data source is unavailable during query evaluation; a partially evaluated query is used to provide a partial answer to the user issuing the query.

5. **Integrating and accessing heterogeneous information sources in TSIMMIS** is a system developed at Stanford university. TSIMMIS stands for *The Stanford - IBM Manager of Multiple Information Sources*, whose goal

is to develop tools to ease the rapid integration of heterogeneous information sources that may include structured and unstructured data. As mentioned earlier in previous systems, there is no global database or global schema to hold information about integrated objects.

The system is based on the OEM (object exchange model) and composed of four main components; mediators, translators, mediator generator, and translator generator. Translators are mainly wrappers that logically converts the underlying data objects to a common information model, in which, each data object is described (or tagged) with labels, types, values, and an optional identifier. Mediators co-ordinate the execution of queries between multiple translators. Mediator Generators, are used generate mediators automatically or semi automatic. Likewise, translator generators are used to generate translators with specific functionalities that guarantee the successful execution of the issued query.

6. **Experiences in federated databases: from IRO-DB to MIRO-Web** is a system developed at GMD. From the beginning of 1994 to the end of 1996, the IRO-DB ESPRIT has developed tools for accessing relational, object-oriented databases in an integrated way.
   The system is based on the ODMG standard as pivot model and language. The system consists of three main layers; the local layer, consisting of multiple Local Database Adapters (LDAs). Since the system is based on ODMG then the model uses OQL (Object Query Language), the function of the adapters is so much similar to the wrapper's, exporting local schemas (relational or Object Oriented) of the data sources into ODMG schemas, and OQL queries into local data source query language.
   The inter-operable layer, supporting integrated views on the imported schemas. The integrated views are derived ODMG classes with many to many relationships. The interactive tool *Integrator Workbench*; helps in designing integrated views. Eventually, the derived classes and their associated relationships are instantiated by means of OQL queries.
   The communication layer, contains Remote Object Access (ROA) modules, from which object oriented remote data access services can be implemented. The object manager integrates the ROA protocol with the interoperable layer, to give the ability of collections manipulation. Through the previously mentioned integration, it is possible retrieve collections of objects stored on local sites by invoking of primitive OQL/CLI.

7. **The Garlic project** is a system developed by IBM *members of the database group* in Computer Science. The goal of Garlic is to enable distributed large-scale multimedia information systems; large scale involves lots of data with multimedia taken as far as possible to mean data of many types. The bulk of the data in the world is not stored in database management systems. There are many specialized systems emerging to store and search for particular data types, including image management systems, etc. However, many applications can benefit from combining information from these various systems.
   The System differs in the execution procedure of sub-queries within the wrappers. Up till now, the function of the wrapper was to execute the sub-query received by the mediator and make the necessary transformation between local and global. In this system, the sub-query is not directly executed upon receival, a query execution plan along with some associated properties is returned to the mediator instead of the answer. The mediator accepts these plans from multiple wrappers and adds some operators to form the global query execution plan. Using this technique,

allows the mediator to have full knowledge of the current actual wrapper functionalities before forming the global query execution plan.

8. **MOCHA: A self extensible database middleware system for distributed data sources** is a system developed at Maryland university. MOCHA stands for Middleware Based On a Code SHipping Architecture, like other systems, the main objective is to scale large environments connecting data sources distributed over a computer network. Unlike most researches on database middleware systems focusing on the problems of translation and semantic integration for multiple data sources, the MOCHA system focuses on two main problems.
   The first problem, the deployment of application specific functionality. The definition of the word self-extensible middleware system means automatically deploying application specific functionalities needed for proper query processing. MOCHA derives its functionality from its name by shipping Java code on demand containing new capabilities to the remote sites.
   The second problem, the efficient processing of queries with user defined operators. Relying on the first solution, having automatically deployed the code, MOCHA produces efficient execution plans using the shipped code that acts as a filter on the data sources.

3. DESIGN ISSUES IN MIDDLEWARE SYSTEMS

In this section we are going to discuss some of the major requirements (issues) that must be present in any data integration middleware system.

Approach, or rather scientific approach refers to a skeleton of techniques for investigating phenomena, whether these techniques rounds up to acquiring new knowledge, or correcting and integrating previous knowledge. To be termed scientific, an approach of inquiry must be based on gathering observable changes, empirical and measurable evidence subject to specific principles of reasoning.

Integrated data model, A data model in software engineering is an abstract model that documents and organizes data for communication between team members and is used as a plan for developing applications, specifically how data is stored and accessed. In Our Case, the word ”data model” refers to the model that co-ordinates communication and processing between the different entities of the system. The word ”integrated” means the transparency regarding system functionalities.

Query language, Query languages are computer languages used to make queries into databases and information systems. Broadly, query languages can be classified according to whether they are database query languages or information retrieval query languages

Query processing, Query processing is the way user’s queries are processed by the computing system. The most famous query processing technique, currently used by a wide variety of database engines; is the traditional query processing via SQL.

Naming In a computing system, a name is an information associated with an object in the system(the name identifies the object) in order to fulfill two functions. First, identification, that provides a way to distinguish the object in question from other objects in the system. Second, providing and access path to the object through its name.

Binding is the process of interconnecting a set of objects in a computer system. For a given object in the system, provided that it has a name assigned to it, through the given name, an access path can be created to the object. The main purpose of binding, is to link, associate objects to each other inside the system.i.e., the association, or link, created between the bound objects, is also called a binding. The purpose of binding is to create an access path through which an object may be reached from another object.

Co-ordination refers to the methods and tools that allow several entities to cooperate towards achieving a common goal. A coordination model provides a framework to handle this cooperation, by defining three elements; the coordination entities, the coordination media, the coordination rules. The main objective of coordination is to integrate (glue) several separate activities into a whole. **Security** is enforced in the middleware by utilizing many security features such as; authentication (verifying identities), authorization (handling credentials), protecting messages from unauthorized modification or disclosure, and managing access policies. In addition, abstraction, portability, and automation can be maintained by careful positioning of the security functionalities, and if the needs be access control, audit policies, and cryptography; flexibility and inter-operability can be enhanced.

4. TABULATION OF DATA INTEGRATION MIDDLEWARE SYSTEMS AND REQUIREMENTS

In this section we are going to tabulate the previously mentioned data integration middleware systems against the stated requirements to provide a comparative view of the investigation. In the table, each value describes one of the previously mentioned systems in a given aspect, as shown in Table 1.

4.1. Table Terminology

As mentioned in the above table, there are some terms that might be anonymous to the reader. In this section, we provide a more detailed explanation of these terms.

- Mediator, a mediator is a neutral party who assists in negotiations and conflict resolution in a process known as mediation.
- RDBMS a database management system in which data is stored in the form of tables, and

P.O.C	Strudel	Araneus	Le Select	DISCO	TSIMMIS	Miro-Web	Garlic	MOCHA
Sponsor	AT and T Labs	Roma Tre University	INRIA	INRIA	Stanford University	GMD	IBM	Maryland University
Approach	Mediator	Mediator	Mediator	Mediator	Mediator	Mediator	RDBMS	MW
Integrated Data model	Graph of HTML pages	Relational	Relational	O.O	OEM	Relational - O.O	O.O	Relational - O.O
Query language	StruQl	HTML Interface	SQL	OQL	OEM-QL	SQL	OQL	SQL
Query processing	Based on info retrieval systems	N/C	Traditional	Traditional	Standard set of sub-queries	N/C	Traditional	Traditional
Naming	Graph Mapping	ADM object manager	Tuples in relational tables	ODMG 2.0	OEM	ODMG	ODMG	Catalog
Binding	Wrappers	Wrapper Library, JDBC,HTTP Server	Wrappers	Wrappers	Translators (Wrappers)	LDA (local layer)	Wrappers (Over Views)	DAP
Co-ordination	Mediator	Mediator	Mediator	Mediator	Mediator	Mediator (interoperable layer)	Mediator	QPC
Security	N/C	N/C	N/C	Administration Interface	N/C	N/C	N/C	Security Manager clas provided by JAVA

Table 1: Results of Analysis

relationship among the data, which is also stored in the form of tables

- Middleware (MW) is a software layer residing on top of the operating system that connects different components or applications.
- SQL (Structured Query Language) is a database computer language designed for managing data in a relational database management system (RDBMS), based upon relational algebra and calculus.
- OEM (Object Exchange Model) is a self-describing common model for heterogeneous information exchange.
- OEM-QL (Object Exchange Model - Query Language) is the logic-based language for OEM that matches object patterns, generate variable bindings, construct new OEM objects from new ones.
- OQL (Object Query Language) is a query language standard for object-oriented databases modeled after SQL. OQL was developed by the Object Data Management Group (ODMG).
- ADM (Araneus Data Model) is used to define a logical scheme for the web site by presenting common features of pages of the same type.
- ODMG (Object Data Management Group) is a group whose main objective is to put a set of specifications for both object and object-relational databases, for developers to write applications that maps object and object-relational databases to products.
  1. ODL (Object Definition Language) is used to define the object types that conform to the ODMG Object Model.
  2. OQL (Object Query Language) is a declarative (nonprocedural) language for query and updating. It used SQL as basis, where possible, though OQL supports more powerful object-oriented capabilities.
- ODMG V 2.0, is the industry standard for persistent object storage. It builds upon existing database, object, and programming language standards to simplify object storage and to ensure application portability.
- Wrapper is a program that extracts content of a particular information source and translates it into a relational form.
- JDBC (Java Database Connectivity) is an API that defines how a client may access a database. It provides methods for querying and updating data in a database.
- LDA (Local Database Adapters) a component that sits on top of database servers, constructing the *local layer* (the first layer in the IRO-DB architecture).
- QPC (Query Processing Coordinator) is the middle-tier component that controls the execution off all the queries and commands received from the client applications in MOCHA. The QPC can be reached through a well known Uniform Resource Locator (URL).
- DAP (Data Access Provider) whose role is to provide the QPC with a uniform access mechanism to a remote data source in MOCHA.
- N/C: not clear in the published paper.

5. ANALYSIS AND RECOMMENDATIONS

From the previous comparison, it is compelling to guide the computer engineers to use which system and in what conditions. In this section, we are going to assume multiple computer environments, each requiring one of the challenges as its prime dominant feature, then state which middleware system can fit best in this case.

5.1. Naming

Naming in any middleware system can be done using one of the following techniques:

- Naming context, by organizing the namespace of the middleware.
- Name resolution, finding the object associated with a cartain name.

The following systems use name contexts; Strudel, Araneus, Disco, TSIMMIS, Miro-web and the Garlic project. While LE SELECT and MOCHA use name resolution.

Considering a computer environment whose dominant feature is performance (first challenge), the use of naming contexts in this environment enhances performance. Naming contexts relaxes the latency criterion, unlike name resolution that introduces a significant delay in the resolution of the name to its physical path, regarding requests for a remote service. Using name contexts helps in speeding up the navigation through the namespace.

Considering a computer environment whose dominant feature is scalability (second challenge), the use of naming contexts in this environment increases scalability. The scalability requirement for a certain service, rises from the fact of maintaining acceptable performance with an increasing service size. At some point, a service might be managing a large number of objects spanning world wide, in addition to the creation of new objects at high rate that requires multiple merge operations, resulting in a large namespace. Naming contexts is effective in these cases providing a better performance via grouping related objects, and shortening the names of objects.

Considering a computer environment whose dominant feature is management (third challenge), the use of naming contexts in this environment elevates management. Managing large applications that are heterogeneous, widely distributed and in permanent evolution may be hard. Name contexts deals with this challenge by choosing names independently in different contexts, in other words, the same name can be used reused in different contexts.

Considering a computer environment whose dominant feature is ubiquity (fourth challenge), the use of name contexts provides better ubiquity than name resolution. As for Ubiquity, mobility and dynamic reconfiguration will become dominant features, requiring permanent adaptation of the applications. In order to maintain the same performance of the service, the service needs to be adaptable. Name contexts provide adaptability through the creation of mount points, or soft links between different contexts.

5.2. Binding

Binding can be achieved through the use of one of two basic techniques:

- Name substitution, replacing an unbound name by another name containing more information on the target of the binding.
- Indirection, replacing an unbound name by (the address of) a descriptor that contains (or points to) the target object.

The following systems use name substitution; Strudel, LE SELECT, DISCO, TSIMMIS and the Garlic project, while Araneus, Miro-web and MOCHA use indirection.

Performance can be slightly enhanced using name substitution over indirection. At most cases, name substitution replaces the name of the variable identifier by its memory location allocated to the variable, unlike indirection that calls the object descriptor, enforcing the cost of at least one extra indirection.

Scalability can be increased using name substitution. Name substitution relaxes the feature of flexibility to normal via static binding of objects, consequently decreasing the binding time of the system to objects.

Management can be alleviated through name substitution. Having a static bind to the object allows all time management and perhaps control on the given object. In the case of managing mobile objects, indirection proves to be better in locating a moving object.

Ubiquity can be improved via the use of indirection over name substitution. Indirection derives its ubiquitous power from the use of dynamic binding to a mobile object, hence permitting dynamic reconfiguration. Another powerful point for indirection mechanisms is delayed binding, deferring the decision of how to reach the mobile object to last possible moment.

5.3. Co-ordination

A coordination model consists of three main components; the coordination entities, the coordination media, and the coordination rules.

Coordination exists in one of three common patterns:

- Observer pattern, it involves the ”observed” object and an unspecified number of independent ”observing” objects.
- Publish - Subscribe pattern, it’s a more general form of the observer pattern, in which two roles are defined. A publisher (event source) from which events are generated and selectively transmitted to subscriber(s) (event sink)
- Shared data space, it identifies a form of communication between uncoupled objects, using a single information space organized as a set of tuples.

All of the previously mentioned systems use the ”Publish - Subscribe” pattern. So any of these systems can be used in any computer environment to produce the coordination efficiency. To increase scalability of the system, the event mediator (publisher) for the environment should not be implemented as a single server, but as multiple co-operating servers.

Note: In order to increase the scalability of the system, the event mediator for the environment should not be implemented as a single server, but as multiple co-operating servers, unlike the QPC, which is the single point of failure in the MOCHA system.

5.4. Security

Though security is an important aspect in the evaluation process of a given middleware, it has been mentioned in only two of the previous systems; MOCHA and DISCO, and discussed in details in MOCHA. Although it was not clear in the published papers of the other middleware systems, how the security mechanism works, it may be justified that the security feature comes with an extra overhead that each time an operation (which the administrator defines as dangerous) is attempted, a call to the security manager will be made to determine if the operation can proceed or not.

- In the DISCO middleware system, it has been mentioned that there are different operations for Database Administrators (DBAs), Database Implementers (DBIs) and Application Programmers (APs), and it has been also mentioned that there is an administration interface, from these two points we can conclude that there is a security mechanism present in the system, but not thoroughly discussed.
- In the MOCHA middleware system, it has been mentioned that the SecurityManager class provided by Java is used to implement security policies on the clients, QPC, and DAP by means of administrators.

Enforcing security on middleware systems serves in challenge of management.

6. REFERENCES

[1] "Middleware, wikipedia," `http://www-caravel.inria.fr/~xhumari/LeSelect/`.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Middleware Architecture with Patterns and Frameworks*, Sacha Krakowiak, Feb. 2009.

[3] Toni A Bishop and Ramesh K Karne, "A survey of middleware," *Technical Report Computer Information Science Dept Towson University*, vol. 10, no. 5, 2000.

[4] Sacha Krakowiak, "Objectweb - what's middleware," 2005, `http://middleware.objectweb.org/`.

[5] Elvira Uchôa, Sérgio Lifschitz, and Rubens Melo, "HEROS: A heterogeneous object oriented database system," in *Database and Expert Systems Applications*, 1998, vol. 1460, pp. 435–447.

[6] Elvira Ucha, Srgio Lifschitz, and Rubens Melo, "HEROS: A heterogeneous object oriented database system," in *DEXA Conference and Workshop Programme*, Vienna,Austria, 1998.

[7] Elvira Ucha and Rubens Melo, "HEROS: a framework for heterogeneous database systems integration," in *DEXA Conference and Workshop Programme*, Italy, 1999.

[8] Alvaro C.P Barbosa and Rubens Melo, "Using HDBMS to access and dispose web information," Technical report, PUC-Rio, Brazil, Portuguese, 1999.

[9] Alvaro C.P Barbosa and Asterio Tanaka, "Using HDBMS as an heterogeneous environmental information integrator," Technical report, PUC-Rio, Brazil, Portuguese, 1999.

[10] C.H.C. Duarte, Esther Pacitti, S.D. Silva, and Rubens Melo, "HEROS: A heterogeneous object oriented database system," in *Proceedings of the VIII Brazilian Symposium on Databases*, Paraba, Brasil, 1993, pp. 383–394.

[11] "The sinpesq project," `http://sinpesq.mpa.gov.br/pndpa/web/`, year = 2011.

[12] Luc Bouganim, Maria Claudia Cavalcanti, Franoise Fabret, Maria Luiza Campos, Franois Llirbat, Marta Mattoso, Rubens Melo, Ana Maria Moura, Esther Pacitti, Fabio Porto, Margareth Simoes, Eric Simon, Asterio Tanaka, and Patrick Valduriez, "The ECOBASE project: database and web technologies for environmental information systems," *Sigmod Record*, vol. 30, pp. 70–75, 2001.

[13] Rubens Melo, Fabio Porto, F. Lima, and Alvaro C.P Barbosa, "ECOHOOD: Constructing configured dbmss based on frameworks," in *Proceedings of the XIII Brazilian Symposium on Databases*, Paran, Brazil, 1998, pp. 39–51.

[14] Tanu Malik, Alexander S. Szalay, Tamas Budavari, and Ani Thakar, "SkyQuery: A web service approach to federate databases," in *Conference on Innovative Data Systems Research*, 2003.

[15] G. Mecca, P. Atzeni, A. Masci, Merialdo, and G. P. Sindoni, "The Araneus web-base management system," in *ACM SIGMOD International Conference on Management of Data*, Seattle, USA, may 1998.

[16] M. Fernandez, D. Florescu, and Jaewoo et al. Kang, "STRUDEL:a web-site management system," in *ACM SIGMOD International Conference on Management of Data*, Arizona, USA, may 1997.

[17] "LE SELECT: a middleware system for publishing autonomous and heterogeneous information sources," 1999,

`http://www-caravel.inria.fr/~xhumari/LeSelect/`.

[18] A. Tomasic, L. Raschid, and P. Valduriez, “Scaling access to heterogeneous data source with DISCO,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, pp. 808–823, Sept. 1998.