Enterprise Software Architecture & Financial Services Industry – Future Trends, Challenges & Frameworks

by

Ahmad Shuja

Submitted to the Sloan School of Management
on May 9, 2003 in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Management of Technology

ABSTRACT

A critical analysis was performed to determine the future of software architecture particularly in the financial services industry. Interviews were conducted with executives in numerous financial institutions to find out which business software applications (packaged out-of-the-box or custom built) really add value to services offered by the financial institutions and to their customers. It was found that 66% to 80% of applications that add real value are not packaged applications. If custom applications are expected to keep playing crucial role in these institutions, then how should these be built to reduce failures and thereby enhancing the probability of long-term strategic and business success.

Accordingly, research was conducted to critically analyze some of the most recent trends in software architectural and management frameworks that support the implementation and management of software systems. These include Zachman’s Framework, Object Management Group’s Model Driven Architecture, Convergent Architecture, and Rules-Based Systems Architecture. It was determined that there was no single framework that fully addresses the multi-dimensional challenges that arise in implementing and managing software systems. Therefore, an effort has been put to define a unified framework (Unified Systems Management Framework) that is expected to deal with and address these multi-dimensional challenges facing the organizations today.

Thesis Supervisor: Stuart Madnick
Title: John Norris Maguire Professor of Information Technology / Sloan School of Management & Professor of Engineering Systems / School of Engineering
ACKNOWLEDGEMENTS

I am grateful to Professor Stuart Madnick for all his guidance and support throughout my year at MIT. He has been a great supervisor and I would not have been able to produce such a thesis without his insights.

I am thankful to Mike Whalen (Senior Manager – Cap Gemini Ernst & Young Inc.) and Robert Holvey (Senior Vice President – Wells Fargo) for their truly valuable comments during the process of interviews and putting together this thesis.

I acknowledge and express my appreciation to my parents in recognition of their patience, devotion, and sacrifice in helping me to attain my educational goals.
TABLE OF CONTENTS

Chapter 1 – Introduction .................................................. 10
  1. Research Interests ....................................................... 14
     1.1 Software Architectural Frameworks .............................. 15
     1.2 Software & Financial Services Industry ......................... 15
  2. Thesis Organization & Overview .................................. 16

Chapter 2 – Financial Services Industry & Enterprise Software
   Architecture – An Overview ......................................... 19
  1. Custom-Built vs. Packages-Solutions ................................ 20
  2. Integration .................................................................. 23
  3. Evolving Businesses & Software Systems .......................... 26
  4. Enterprise-Wide vs. Business Specific Applications .............. 27

Chapter 3 – Software Systems Architecture & Related Challenges 30
  1. Introduction ................................................................ 32
  2. Definition of Architecture ............................................ 34
  3. Role of Architecture .................................................... 39
  4. Factors Affecting Architecture ...................................... 40
     4.1 Evolution .................................................................. 40
     4.2 Virtual Organizations ............................................... 41
Chapter 4 – Critical Technological Analysis of Current Software

Systems Architectural Frameworks 50

1. Zachman Framework 52
2. Model Driven Architecture (MDA) 56
   2.1 Background 56
   2.2 Vision & Concept of MDA 58
      2.2.1 The Core 59
      2.2.2 Middleware Layer 61
      2.2.3 Pervasive Services Layer 62
      2.2.4 Vertical Markets or Domains Layer 63
   2.3 Models in MDA 64
      2.3.1 Platform-Independent Model (PIM) 65
      2.3.2 Platform-Specific Model (PSM) 67
   2.4 Developing in the MDA 68
      2.4.1 Single Target Platform 68
      2.4.2 Multiple Target Platforms 71
      2.4.3 Code Generation 72
   2.5 Mappings of Models 74
Chapter 5 – Unified Systems Management Framework  97

1. Organizational & Management Infrastructure  101
2. Business Process Design  104
   2.1 The Enterprise Architecture Cycle  106
Chapter 6 – Conclusion & Future Research

1. Applicability of Framework
2. Knowledge Reuse & Refinement
3. Standardization

References

Glossary
**TABLE OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Model Driven Architecture</td>
<td>47</td>
</tr>
<tr>
<td>4.1</td>
<td>The Zachman Framework</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>Object Management Group’s MDA</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Using the MDA to generate a CCM Server</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Leveraging Other Middleware and Client Platforms</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>PIM-PSM Mappings</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>Roadmap &amp; Anatomy of the Convergent Architecture</td>
<td>81</td>
</tr>
<tr>
<td>4.7</td>
<td>The Modules of the IT-Architectural IDE</td>
<td>85</td>
</tr>
<tr>
<td>4.8</td>
<td>Rules Extraction</td>
<td>92</td>
</tr>
<tr>
<td>4.9</td>
<td>Overview-Ideal Rules Based Systems Architecture</td>
<td>93</td>
</tr>
<tr>
<td>4.10</td>
<td>Rules-Based Architecture</td>
<td>94</td>
</tr>
<tr>
<td>5.1</td>
<td>Unified Systems Management Framework</td>
<td>102</td>
</tr>
<tr>
<td>5.2</td>
<td>The Enterprise Architecture Cycle</td>
<td>106</td>
</tr>
<tr>
<td>5.3</td>
<td>Architecture-Centric Project</td>
<td>115</td>
</tr>
<tr>
<td>5.4</td>
<td>MDA-Zachman Unified Model</td>
<td>122</td>
</tr>
<tr>
<td>6.1</td>
<td>Request &amp; Response Messages</td>
<td>131</td>
</tr>
</tbody>
</table>
Most enterprises today have no formal architecture whatsoever (Gartner, 2003). Their IT landscapes are spaghetti-like collections of new technologies grafted onto legacy systems. Integration has been painful. Information is seldom shared efficiently, and almost never with the outside world (Gartner, 2003). The solution is not just another big-bang investment initiative. Nor is it another universe-shattering technology solution. It’s a way of thinking and an important new approach to the technology infrastructure. To present a complete set of solutions to software architectural challenges facing enterprises today is out of scope of this thesis. I am going to present the findings of some of the interviews I conducted with senior executives at several financial services institutions. Following this, I will present, critically analyze, and discuss in detail some of the architectural and management frameworks that are under research. I will show if these suggest a possible solution to the challenges being faced today.
Differences are inevitable. Most enterprises operate a vast sprawl of applications, hardware of mixed vintage, Mergers & Acquisitions (M&A) leftovers, and hot new technologies (Gartner, 2003). These enterprises constantly fight to overcome incompatibility within their own four walls by tinkering with the technology. The goal of the enterprise software architecture is to create universal compatibility, not by changing the applications, inserting a new middleware platform, or altering the hardware, but rather by replacing rigidity with adaptability and flexibility. According to Gartner (2003), part of the solution is related to effectiveness; if a technology you own can help your business processes work effectively with all your partners, keep it. If not, consider replacing it. If you do it right, business change will be able to enter the infrastructure like a welcomed guest.

Alongside, with the advent of the Internet, customers are more informed and demanding than they were ever before. They expect a superior level of service, which in turn puts the heavy burden on technologists and business professionals to find new ways of dealing with these challenges; business process re-engineering to leverage what all technology has to offer. In fact, how you deal with your customers gives you an advantage that is not easily imitable. Most recently, strategists share the view that a focus on customers in terms of anticipating, understanding and responding to their needs rapidly and efficiently, and ultimately establishing enduring relationships between service providers and customers, creates value that is sustainable and often difficult to imitate (Melnick, Nayyar, Pinedo & Seshadri, 2000). Value creation for customers is a challenging task and cannot be accomplished without specialized implementation of solutions. Packaged
solutions are a way to leverage work that has already been done. However, these systems provide most generic services and functions, leaving out the core functionalities and workflow processes that create the fundamental competitive advantage and differentiating factor. In an interview, Robert Holvey, a Senior Manager at Wells Fargo, mentioned that,

“Unfortunately, our business is reasonably complex and often it doesn’t quite fit into the rigid confines of the packaged solutions, at least not without some heavy modification.”

Such limitations posed by packaged solutions do not let huge multi-national organizations to base their competitive strategies to create value through such systems. Discussing the possible solution to this limitation, Robert Holvey said,

“Many times, we simply have to build our own systems because we need application functionality that just doesn’t exist out in the market place. For example, transaction systems that both support real-time online transaction processing for e-commerce still have to include risk algorithms and analyses that help ensure that the transactions are legitimate (not fraudulent). Additionally, since fraudsters are extremely aggressive and innovative, the integration of the risk management processes has to continuously evolve to keep pace with trends and innovations. Finally, flexibility and value added expected by the customers require us to continue
altering the primary customer interface. The end result is that we need systems that are rapidly and continuously evolving to meet the changing needs of the business and the changing expectations and needs of the customers. That level of customization can’t be found in any packaged solutions.”

Many others have similar feelings about the packaged solutions. There are a number of questions that come up including, but not limited to, the following:

✓ What options do we have? We will analyze the architectural and management frameworks that are expected to play an important role in building custom software systems.

✓ Most of these corporations find themselves with no other option but to build custom software systems. We know that software projects end up in failure more than in success. How do we build systems that are less prone to failures and more adaptable and platform independent? We will see how can the probability of success can be increased by following certain methodologies.

✓ How do we build systems that can be integrated with legacy systems that carry years of effort and knowledge to create competitive edge? Model Driven Architecture (MDA), discussed in Chapter 4, focuses on addressing the related challenges.
✓ How do we manage systems development process so that the probability of failure is reduced? *Zachman’s Framework, discussed in Chapter 4, presents a possible solution.*

✓ How can we build systems that can be migrated to a new technology that comes out in future and still make use of the accumulated knowledge? *We will see the possible approaches in Chapters 4 and 5.*

✓ How do we make sure that we don’t overkill when it comes to technology and make technology serve the business better while keeping pace with the business change?

1. **RESEARCH INTERESTS**

   My primary research interest is, “*What is the role played by Enterprise Software Architecture to address the challenges (highlighted) discussed above?*”

   According to Gartner (2002), software and IT services are the fastest growing components of financial services IT spending from 2000 to 2005, with a five-year compound annual growth rate of 13.3 percent and 11 percent, respectively. Gartner Dataquest analysts said IT services constitutes the largest single segment of IT spending by financial services. Since Financial Services is an industry that heavily invests in R&D
to find out new innovative ways to create value through the use of technology, I will investigate the application of numerous frameworks discussed in this thesis in relation to financial services industry.

Accordingly, my research interests can be categorized as follows:

1.1 SOFTWARE ARCHITECTURE FRAMEWORKS

What are the latest trends in Enterprise Software Architecture? Are there any frameworks (Technology & Management) that can increase the probability of success?

The purpose of this component is to analyze the current solutions available. I want to critically analyze the gaps within the boundaries of software architecture that result in such issues as global integration of systems, aggregation, and data standardization across numerous platforms and continents. This part will primarily be based on studying existing best practices and technological and management frameworks that are under research as well as in practice. However, there will be a detailed discussion on some new ways to manage and build software systems.

1.2 SOFTWARE & FINANCIAL SERVICES INDUSTRY

Roughly what percentage of software is custom developed in financial institutions? What issues / challenges have financial institutions faced in the past? What
issues / challenges financial institutions foresee with future e-business implementations? What is financial institution's position on custom solutions and packaged e-business solutions? What are the motivations behind pursuing custom or packaged solution? What is the level of expertise utilized in e-business implementations; Outsourcing, Consulting, In-House etc? What software development methodologies, if any, do you follow? What is financial institution's position on enterprise wide software integration? How does technology innovation take place in such big firms?

My primary focus will be on discussing the solutions to build custom built software systems that financial institutions have For this part of the research, I have interviewed as many firms as possible. I studied the approaches taken and technologies used primarily through interviews, case studies, and literature available, and recent research. I came up with a unified architectural framework (Unified Systems Management Framework) for integrating global financial systems for major platforms (.NET, J2EE etc.). An outline of content and organization of each chapter is as follows.

2. **THESIS ORGANIZATION & OVERVIEW**

Throughout this thesis, the discussion will be primarily focused on software systems architecture (frameworks, trends and challenges) in relation to financial services industry.
Chapter 2 will present a detailed analysis of literature review of financial services and Information Technology. For this part, I will present the findings of some of the interviews that I conducted with senior executives at numerous financial institutions (Wells Fargo, Merrill Lynch, Deutsche Bank, and Bank of America). Based on these interviews, I will lay out the foundation of the challenges faced by these financial institutions and the way these institutions address these challenges. This chapter is going to form the problem statement for the research.

In Chapter 3, we will discuss enterprise software architecture and how it relates to software implementation and deployment and ultimately to the quality of software produced. I will present different definitions and numerous views of software architecture. Why is it important for software engineering discipline and what is its role? What are the various factors that affect and contribute in the evolution of software architecture? A brief overview of numerous architecture and management frameworks will be presented. These frameworks will be analyzed and discussed in detail in Chapter 4.

Chapter 4 will present a detailed critical analysis of Model Driven Architecture (architectural framework by Object Management Group), Zachman’s Framework (Management Framework), Convergent Architecture (CA), and Rules-Based Systems Architecture, which have been briefly outlined in Chapter 3. This chapter will basically form the foundation of the unified framework discussed in Chapter 5. Most of the discussion will be carried out specifically in relation to financial services industry.
Chapter 5 will present a new unified framework (that I have created as a result of research conducted) that will outline the roadmap which enterprises should follow to manage and implement software systems. This framework (Unified Systems Management Framework) will provide an overview of numerous factors that affect the implementation and management of software systems. We will also discuss the frameworks discussed in earlier chapters in relation to financial services industry and where do they fall in the Unified Systems Management Framework (USMF). I interviewed executives at couple of financial institutions and will analyzed how the software products are built and managed. The aim is to understand how some of these frameworks (if not already adopted) can be applied within financial services industry.

Chapter 6 will conclude the thesis with the possibilities and directions for the future research.
Chapter 2

FINANCIAL SERVICES INDUSTRY & ENTERPRISE SOFTWARE ARCHITECTURE – AN OVERVIEW

Rapid advances in technology and financial innovation, along with an ever changing regulatory environment, are reshaping the future of the Financial Services industry (Fan, Srinivasan, Stallaert, & Whinston, 2002). Software being the most significant part of this technology will be the focus of our discussion in this chapter. This chapter is primarily going to serve as an outline of numerous challenges that are faced by the financial services industry with regards to architecting, building, and managing software systems. Numerous architectural and management frameworks that will be discussed throughout this thesis will aim to address some of the issues discussed in the following sections. In order to gain an understanding of some of the challenges facing the
financial services industry today and the way these institutions are dealing with these issues, I have interviewed couple of executives in major financial institutions across the U.S. and will summarize the findings in this chapter.

1. CUSTOM BUILT vs. PACKAGED SOLUTIONS

It is generally presumed to be best to purchase an asset if an appropriate product is available from a vendor that specializes in such products. However, there are situations in which building is better. An internally developed application may be preferable when it offers a strategic advantage, when the IT infrastructure prohibits use of a commercial application, or when such applications cannot provide sufficient control over unique processes (Expanseinc, 2003). In addition, as we will see later in this section, some business processes are not easily accommodated by packaged applications. It also may be desirable to maintain control over the process, as well as on the actions of the users who administer them.

I personally feel, from my own experience and the interviews conducted, that custom built applications will continue to play a crucial role in creating competitive advantage for enterprises; financial institutions are no exceptions. The proportion of custom applications being built may decline, but it will never totally cease to exist. Greg Levine, Vice President at Merrill Lynch Technology Infrastructure Group, commented in his interview with me that 70% to 80% of the business applications that give competitive
edge to Merrill Lynch are home grown and that most of the packaged solutions are in reality commodity solutions and commodity does not add much differentiating value for the customer. Robert Holvey, Senior Vice President at Wells Fargo, comments on this and says that about 1/3 of business applications are developed in-house, for about 1/3, they buy the license to the basic software and then heavily customize it to meet their needs and remaining 1/3, they buy and use “off the shelf” without customization. Packaged solutions are a way to leverage work that has already been done. In this case, the 1/3rd of software is custom built and for 1/3rd, the code is bought and heavy modifications are made; I consider such software to be custom built with a major portion being re-used. Note that almost 66% to 80% of software that adds value to the services provided by these financial institutions is custom built.

Most comments I received from the executives conveyed the same message that their respective businesses are reasonably complex and often do not quite fit into the rigid confines of the packaged solutions, at least not without some heavy modifications. Further, I found in my interviews that many times these financial institutions simply have to build their own systems because they need application functionality that just does not exist on the market place. For instance, Robert Holvey mentioned in his interview that transaction systems that both support real-time online transaction processing for e-commerce still have to include risk algorithms and analysis that help ensure that the transactions are legitimate (not fraudulent). Such systems have to be custom built because of inherent complexity and uniqueness of the business function.
What value do these custom developed systems and applications create and can it be measured? In an interview, one Senior Vice President at Wells Fargo said,

“Wells Fargo continues to be one of the leading e-business sites in the financial services industry. We now have millions of our customers that take advantage of our e-business implementations. The rapid double digit growth rates in both customers and transactions for these custom software implementations and the fact that we are retaining our market leadership positions both lead us to think that the customer feedback extremely positive. We track how some of our major competitors are doing, and we continue to grow our e-business faster than they do. As a result, we think that we are probably doing a lot of things right.”

We have seen couple of scenarios in which custom development is unavoidable. However, there are still challenges and issues with custom development that need to be considered and analyzed before pursuing with the development process. Even when conditions indicate the need for a custom-built application, other considerations may influence the feasibility of such a solution. Developing and maintaining the application requires substantial programming resources (Expanseinc, 2003). Even if such resources are available, it is questionable whether the application can be developed and tested in a cost-effective, timely manner. Given the rate of change in information technology today, it is entirely possible that other organizational initiatives may dictate implementation of
newer systems that render the new application obsolete before the application-development cycle is complete (Expanseinc, 2003).

There are deliberate efforts being made to stress upon the importance of defining software architecture prior to building it and how to effectively and efficiently manage software engineering and development projects. In fact, it is imperative to have the software systems architecture to be in line with the business process requirements and long-term business strategy. The software architecture and management frameworks, discussed in Chapter 4 and Chapter 5, are expected to reduce the risk inherent in custom application development and increase ROI (Return On Investment) while reducing time-to-market (implementation life time).

2. **INTEGRATION**

The basic idea is to automate handling, routing, and translation of data (Gartner, 2003). Define the sleekest pathways, then build them. Integration points need to be created wherever they are required to eliminate stops or bottlenecks, open up portals for universal access and make the business processes work in real-time (Gartner, 2003).

In order to increase organization’s agility in responding to customer, market, and strategic requirements, the information flow between the IT systems that carry out these business operations must be streamlined. This includes not only the organization’s own
IT systems but also those of its partners. It is the task of electronic business integration to automate this information flow as much as possible in order to streamline operations. Integration points need to be created wherever they are required to eliminate stops or bottlenecks. Achieving better internal integration has obvious business benefits — to instantly get the right information or functions into the right employees’ hands, in the right combinations, at the right time (Gartner, 2002 paragraph 14). Historically, organizations have generally focused on integrating own IT systems. If, however, the information flow between the organization’s own IT systems and those of its partners (particularly in the supply chain) is not also streamlined, then the overall agility of the business is still restricted (Kapil, Sadhwani, Samtani, Siddiqui, Clark, Fletcher, Hanson, Irani, Waterhouse & Zhang, 2002). Therefore, many enterprises also strive to integrate their partner’s IT systems with their own in order to more fully automate critical business processes such as sales, procurement, and research and development. The benefits of the increased agility resulting from business process automation are extensive. For example, operational costs are decreased, inventories are reduced, customer satisfaction is increased, and products are brought to market faster (Kapil et al, 2002). When we talk about integration, Enterprise Software Architecture has to support inter- and intra-enterprise integration of a number of different systems based on a different platforms and technologies. Integration will involve integrating different systems and each may be running on a different middleware platform.

Integration of e-business applications into the business that financial services industry does with its customers becomes a vital segment for creating value-added
services. However, integrating different categories of applications and systems becomes the next big challenge. Integration can be carried out through numerous channels within financial institutions (branches, phone, centers, teams of dedicated sales and services staff, and Information Systems) in a way that allows these institutions understand all aspects of the customer relationships and needs, hence creating services most valued by these customers. Other challenges associated with e-business implementations, as pointed out by Robert Holvey, include real-time processing of huge amounts of data in a way that allows automated customer interactions – cash management, transaction processing, sales offers, risk management, etc. Recently, Deutsche Bank Bauspar AG went through a major project of integrating new business applications with proven legacy systems (Interactive Objects, 2003). The new b-online system at Deutsche Bank was designed to enhance customer service and improve organizational efficiency. The focus in the development project was on the implementation of new business functionality around a proven legacy system. The core system of the Deutsche Bank Bauspar AG runs on an IBM Mainframe and is based on CICS, COBOL, and DB2. This legacy system was implemented in the eighties and has undergone continuous development ever since. The new application runs on more than 30,000 client machines in all of the approximately 1,250 Deutsche Bank branch offices and integrates the existing COBOL Mainframe system in a variety of new Web-based systems (Interactive Objects, 2003). Similar situations exist in most of the other financial institutions that started off with legacy systems in eighties and the challenge is to leverage decades of effort put into those legacy systems with new cutting edge business applications and technologies.
3. EVOLVING BUSINESSES & SOFTWARE SYSTEMS

Most enterprises remain un-evolved. While many Information Systems Organizations have moved from big iron to distributed environments, they are still unconnected with the outside world and, thus-in context of what’s ahead-prehistoric. The real leap in evolution starts with defining new enterprise software architecture. Instead of buying an application that works brilliantly, the job will be to assemble applications that work brilliantly together (Gartner, 2003). In addition, the applications that evolve as the businesses and customers’ needs evolve.

The same is true for financial services industry. I realized after carrying out a number of interviews that that these institutions need systems that are rapidly and continuously evolving to meet the ever changing needs of the businesses and the changing expectations and needs of the customers. Strong opinion was that such level of customization cannot be found in any packaged solutions and in the face of such a challenge, custom applications development becomes inevitable.

Robert Holvey (Senior Vice President at Wells Fargo) in his interview with me said:

“In case of packaged solutions, respective updates to the modules are determined and delivered by the packaged solution providers. You are not
allowed to make heavy modifications / customizations to any of the modules. If you do, the vendors don’t support these medications in case of failures. In addition, you cut yourself out of the new releases, which would over-write the medications / customizations you made. Custom business applications allow you to leverage your domain knowledge and update those systems as you know, learn and understand new dimensions of your business as a result of evolution. Additionally, since fraudsters are extremely aggressive and innovative, the integration of the risk management processes has to continuously evolve to keep pace with trends and innovations. Finally, flexibility and value added expected by the customers require them to continue altering the primary customer interface.”

4. ENTERPRISE-WIDE vs. BUSINESS SPECIFIC APPLICATIONS

With some off-the-shelf packaged applications (e.g., email or other communication functionality) enterprise-wide implementations are an obvious choice. With others that need to be more customized to specific business objectives, folks whose primary functions are to help control costs are the strongest advocates of “enterprise-wide”. On the other side of the spectrum, we have business line managers who generally understand that specific revenue generation opportunities and objectives often times don’t
lend themselves to “enterprise-wide” applications and implementation. Richard Holvey said in his interview:

“The business objectives and opportunities of one of our business areas are not the same as many of the other business areas. What they need in software for revenue generation is, as a result, often more than a little bit different. If version A of the software application helps better realize one opportunity and version B is better at producing revenues for another business, the additional revenues will often more than pay for developing and maintaining multiple customized versions of the software application. This won’t minimize costs, but it will produce the best overall bottom line business results. Also, because the business is complex and the opportunities are always evolving, the level of rigidity in the development processes and change control processes gets more extensive the closer the application gets to “enterprise-wide”. That means that you have faster and more focused development cycles when your development decisions don’t have to meet the overly-constrained change-control and business requirements of all business units. Evolving the software application to pursue specific opportunities is faster, easier and more effective at generating the specifically envisioned business revenue objectives, while the downside is extra costs.”
Presence of custom applications, packaged solutions, and integration requirements creates an environment that asks for innovative and efficient solutions. In this thesis, I am going to discuss in detail, critically analyze, and propose possible solutions to some of the challenges highlighted in this chapter.
Chapter 3

SOFTWARE SYSTEMS ARCHITECTURE

& RELATED CHALLENGES

An enterprise, as defined by Mary Johnson and Larry Whitman of the University of Texas Automation & Robotics Institute, is a complex system of culture, process and technology components engineered to accomplish organizational goals (Popkin, 2003). John A. Zachman defines Enterprise Architecture (Popkin, 2003) as:

“...the set of descriptive representations (i.e., models) that are relevant for describing an Enterprise such that it can be produced to management’s requirements (quality) and maintained over the period of its useful life (changed).”
Enterprise Software Architecture, as the name shows, ensures that software systems are built in a way that would support future evolutions of systems, businesses, and processes as well as seamlessly integrate back-end systems with front-end services to empower customers. Increasing number of online services provided by Financial Institutions includes pushing risks to the customers. Customers/end users are provided with more information than ever before and are then allowed to make well informed, independent decisions. Such tasks would involve providing partial back-end information, which was previously used by the institutions to service their clients, directly to the clients’ desktops. The next issue that comes to mind is “Integration”. To this end, we have seen couple of major Financial Institutions facing major challenges and finding innovative ways to address those. For instance, Deutsche Bank Bauspar AG designed new business online (b-online) system to enhance customer service and improve organizational efficiency. The focus in the development project was on the implementation of new business functionality around a proven legacy system (Business Experts, 2003). The project included integration of existing Cobol Mainframe Application into Modern Web-Based Systems. For this thesis, I will focus on the evolving trends of Enterprise Software Architecture and their possible applications specifically in the Financial Services Industry.

---

1 Deutsche Bank Bauspar AG is a company of Deutsche Bank AG, the leading German bank. Deutsche Bank Bauspar AG provides special savings plans within the framework of savings and loan associations, designed to help its customers build capital to acquire real estate.
1. INTRODUCTION

Over quite a few years (decades, in some cases), organizations have accumulated a large inventory of applications, services and databases hosted on all different types of platforms. Most applications were never designed to work with each other, and to the extent that they were later modified to do so, the integration was not implemented according to any specific standards. In fact, standard design patterns for the integration mechanisms were simply not available.

As a result, many organizations have IT landscapes that even defy description, let alone understanding. The enterprise architecture might be best termed as "spaghetti architecture" - not unlike the "spaghetti programs" of the 60's and 70's. And like spaghetti programs, spaghetti architectures tend to be brittle and highly resistant to change. Stressing upon the importance of Enterprise Architecture, analysis carried out by Gartner\(^2\) concludes that The Real-Time Enterprise will drive competitive edge and The New Enterprise Architecture is the necessary path to getting there. The CIOs who preside over enterprise IT landscapes with spaghetti architectures tend to be regarded as the source of the inertia that hampers the evolution of enterprises rather than the change agents that many of them undoubtedly aspire to be.

It’s difficult – in fact, next to impossible – for large enterprises to standardize on a single middleware platform. Some enterprises found themselves with more than one

\(^2\) gartner.com/ea
because their different departments have different requirements, others because mergers or acquisitions created a mix. Even the lucky enterprise with a single middleware choice still has to use other technologies to inter-operate with other enterprises and B2B markets (Soley, 2000). By providing interface standards, Web Services can be viewed as programming paradigm for extracting and integrating data from heterogeneous information systems (Hansen, Madnick & Siegel, 2002). It offers significant advantages over currently available methods and tools (Kapil, Sadhwani, Samtani, Siddiqui, Clark, Fletcher, Hanson, Irani, Waterhouse & Zhang, 2002). As we can see in (Kapil et al., p. 157, 158), a whole new set of tools has arisen to facilitate the integration and automation of business processes. These include graphical process modeling tools, middleware technologies such as CORBA (definition included in the Glossary) and JMS, integration brokers, Business Process Management System (BPMS), and B2B servers. Unfortunately, until recently the investment required by organizations to integrate the IT systems both inside their organization and across firewall has been very high. This is mainly because the different proprietary interfaces and data formats used by each application have meant that integration projects have had to invest considerable resources in expensive integration tools as well as in the time and expertise to perform the integration (Kapil et al., p. 157, 158).

The middleware environments that are most visible today are CORBA, Enterprise Java Beans (EJB), message-oriented middleware, XML/SOAP, COM+ and .NET. However, over the past decade or so, the middleware landscape has continually shifted. For years we’ve assumed that a clear winner will emerge and stabilize this state of flux,
but it’s time to admit what we’ve all suspected: The sequence has no end! And, in spite of the advantages (sometimes real, sometimes imagined) of the latest middleware platform, migration is expensive and disruptive (Soley, 2000).

Nevertheless, enterprises have applications on different middleware that simply have to be integrated even though this process is time consuming and expensive. Furthermore, the middleware they use continues to evolve. To make matters even more complicated, different technologies have been used depending on whether communication is within or beyond the firewall. A component built on the assumption that it communicates within a firewall might have to be exposed beyond the firewall for B2B e-commerce, and a component exposed via an extranet might be moved behind the firewall because of an acquisition or merger. Thus, in addition to the basic integration problem, the IT organization must find a way to preserve the development investment made in new components as enterprise boundaries shifts and the underlying technology must change accordingly (Soley, 2000).

(Important terms are defined in the Glossary attached)

2. **DEFINITION OF ARCHITECTURE**

The term “architecture” has been used for many years within the IS community to refer to various types of overviews that provide guidance to software systems and applications developer (Harmon, 2003). The term is obviously a metaphor derived from
the building trade. “Architecture” seems to have gained as much importance in software engineering community as it has been in civil engineering and building architecture. In fact, Harmon (2003) says that just as builders would not undertake the construction of a house or an office building without an architecture, documented in various blueprints, so software developers should not undertake the development of software systems without a detailed plan, documented with software “blueprints” of various kinds. According to Soni, Nord, and Hofmeister (1995):

“Software architecture is concerned with capturing the structures of a system and the relationships among the elements. The structures we found fell into several broad categories: conceptual architecture, module architecture, execution architecture, and code architecture.”

As mentioned above, there are at least four views of software architecture. The idea of separating software architecture into these views is not unique (Hofmeister, Nord & Soni, 2000). The reason behind multiple views is always the same: Separating different aspects into separate views helps people manage complexity (Hofmeister et al. 2000). These views are briefly defined below (Hofmeister et al. 2000):

**Code View:** The organization of the source code into object code, libraries, and binaries, then in turn into versions, files, and directories strongly affects the reusability of the code and build time for the system. This is the Code View.
**Module View:** Although the design of individual classes or procedures is usually too fine grained to be considered part of software architecture design, the decomposition of the system and the partitioning of modules into layers is the main purpose of the Module View.

**Execution View:** Systems have always had a dynamic aspect. As systems became distributed, programmers had to decide how to allocate functional components to runtime entities; how to handle communication, coordination, and synchronization among them; and how to map them to the hardware. Software engineering researchers responded to these challenges by designing interconnections languages that address these dynamic structural issues.

Now these issues are generally recognized as being architectural-level issues. They are better handled by the architect early on in the project, rather than being handled by the programmers as development progresses. This is the work of the Execution View.

**Conceptual View:** Conceptual View describes the system in terms of its major design elements and the relationships among them.

Some of the Architectural Structures of a System (Bass, Clements & Kazman, 1998) are shown below:
<table>
<thead>
<tr>
<th>Software Structure</th>
<th>Units</th>
<th>Relation Represented by the links</th>
<th>Useful for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Work assignments</td>
<td>Is a sub-module of; shares secret with</td>
<td>Resource allocation and project structuring and planning; information hiding, encapsulation; configuration control</td>
</tr>
<tr>
<td>Conceptual</td>
<td>Functions</td>
<td>Shares data with</td>
<td>Understanding the problem space</td>
</tr>
<tr>
<td>Process</td>
<td>Programs</td>
<td>Runs concurrently with: may run concurrently with; excludes; precedes; etc.</td>
<td>Scheduling analysis; performance analysis</td>
</tr>
<tr>
<td>Physical</td>
<td>Hardware</td>
<td>Communicates with</td>
<td>Performance, availability, security analysis</td>
</tr>
<tr>
<td>Uses</td>
<td>Programs</td>
<td>Requires the correct presence of</td>
<td>Engineering subsets; engineering extensions</td>
</tr>
<tr>
<td>Calls</td>
<td>Programs</td>
<td>Invokes with parameters</td>
<td>Performance profiling; bottleneck elimination</td>
</tr>
<tr>
<td>Data flow</td>
<td>Functional tasks</td>
<td>May send data to</td>
<td>Traceability of functional requirements</td>
</tr>
<tr>
<td>Control flow</td>
<td>System states or modes</td>
<td>Transitions to, subject to the events and conditions labeling the link</td>
<td>With timing information, can be basis for automation simulation and certification of timing and functional behavior</td>
</tr>
<tr>
<td>Class</td>
<td>Objects</td>
<td>Is an instance of; shares access methods of</td>
<td>In object-oriented design systems, producing rapid almost-alike implementations from a common template</td>
</tr>
</tbody>
</table>
Tracing back the history, the term “architecture” began to be used by business managers around mid-nineties, especially those involved in enterprise planning and in business process reengineering projects, to describe an overview of the business (Harmon, 2003). Today, there is a growing movement among both business managers and IS managers to use the term “enterprise architecture” to refer to a comprehensive description of all of the key elements and relationships that make up an organization. Increasingly, when managers talk about the alignment between business processes and goals and IS applications and middleware systems, they rely on an enterprise architecture to define how the business-IS alignment should be achieved (Harmon, 2003).

The field of software architecture is still evolving. In my personal experience, many enterprises still don’t give architecture that much importance. Professional organizations like Object Management Group (OMG) are striving to define standards and ways in which to define architecture for long-term success. There are a number of architectural frameworks and models that are in practice i.e., The New Enterprise Architecture by Gartner (2003), Model Driven Architecture (MDA) by Soley (2000), Enterprise Architecture by Harmon (2003), Applied Software Architecture by (Hofmeister, 2000). Some are proven and others are being evaluated and are evolving. In this chapter, we will very briefly discuss a new initiative by OMG i.e., Model Driven Architecture (MDA) and Zachman Framework. These, in addition to Rule-Based Architecture, which is an attempt to add flexibility in out-of-the-box solutions (ERP and CRM) as well as implementing a more manageable custom software system, will be described and analyzed in greater depth in Chapter 4.
3. **ROLE OF ARCHITECTURE**

In software as in any other systems, the architect’s basic role is the reconciliation of a physical form with the client’s needs for function, cost, certification, and technical feasibility. The architect develops the architecture. Following Brooks’ term, the architect is the user’s advocate. As envisioned in this book, the architect’s responsibility goes beyond the conceptual integrity of the systems as seen by the user, to the conceptual integrity of the system as seen by the builder and other stakeholders. The architect is responsible for both what the system does as well as how the system does it. But the responsibility extends, on both counts, only as far as is needed to develop a satisfactory and feasible system concept. After all, the sum of both is nearly the whole system, and the architect’s role must be limited if an individual or small team is to carry it out. The latter role, of defining the overall implementation structure of the system, is closer to some of the notions of software architecture in recent literature (Maier & Rechtin, 2002).

The architect’s realm is where views and models combine. Where models that integrate disparate views are lacking, the architect can supply the insight. When disparate requirements must interact if satisfaction is to be achieved, the architect’s insight can insure that the right characteristics are considered foremost; and, moreover, to develop an

---

3 Brooks, F. (1975). The Mythical Man Mouth, Addison-Wesley, Reading, MA. This classic book has recently been republished with additional commentary by Brooks on how his observations have held up over time; p. 256 in the new edition.
architecture that can reconcile the disparate requirements (Maier & Rechtin, 2002). These requirements can very well be related to future growth of the enterprise in all possible dimensions. Such growth may also include, but not limited to acquisitions as well as mergers. One of the challenges that these multi-national corporations face as a result of mergers and acquisitions is enterprise system integration. Hence the perspective required is predominantly a system perspective. It is the perspective of looking at the software and its underlying hardware platforms as an integrated whole that delivers value to the client. Its performance as a whole, behavioral and otherwise, is what gives it its value.

4. FACTORS AFFECTING ARCHITECTURE

4.1 EVOLUTION

An architecture style\(^4\) is never finished until the community of designers stops finding improvements for it or until it just goes out of style. The Ionic Temple is an example of both these situations. First, its architectural style went out with the dispersal of ancient Greek culture. The religious reasons to build such temples became a remnant of history. However, the Parthenon is still considered to be epitome of an Ionic temple. Constructions based on its architecture were built

\(^4\) IT-Architectural style is more than a pattern. It uses and consolidates specific pattern, but not all patterns. In addition, it comprises other development aspects such as component standards, modeling languages, business design concepts, and technology mappings. It even includes its own streamlined development process. Thus, just as components accompany and complement object technology, IT-architectural styles leverage and complement patterns (Hubert, 2002, p.7).
hundreds of years later in Rome, Paris, and even Thomas Jefferson’s Virginia.\textsuperscript{5} Each of these reproductions bears witness to the relevance and value of an accomplished architectural style, which is reused as is, perfectly fulfilling its purpose each time. There are also cases of architectural styles experiencing a renaissance by being reactivated into the active engineering mainstream (Hubert, 2002, p.9).

\textbf{4.2 VIRTUAL-ORGANIZATIONS}

Nowadays, many organizations form dynamic partnerships to effectively deal with market requirements. Companies focus on their core business and outsource secondary activities to other organizations (CrossFlow Project, 2003). Market opportunity and market demands are the triggers that pull the virtual organization together. A market opportunity can be defined as a latent area of need or requirement in which a company can perform profitably (Kotler, 1994, p.80). The virtual organization can fulfill this need by bringing together the competencies needed for developing, pricing and offering the appropriate products and/or services (Saabeel, Verduijn, Hagdorn, & Kumar, 2002). The virtual organization focuses on a particular market segment or target group. An example is the Smart Car in which Swatch and Mercedes-Benz took the initiative to develop and offer a new mobility concept based on a compact car.

\textsuperscript{5} The Virginia State Capitol is described as the “first adaptation of a temple for a modern public building not only in America, but in the world” (Girouard, 1963)
First, like architecture, the term “virtual organization” seems to have as many definitions and perspectives as there are people talking about it. Most definitions of the concept of virtual organizations start with stating that it is “a network between organizations or individuals…” Others (Saabeel et al., 2002) state that virtual organization is an approach to management or a strategic approach. Some other definitions include:

✓ A temporary network of independent companies that come together quickly to exploit fast-changing opportunities (Byrne, 1993, p.36-37).

✓ An opportunistic alliance of core competencies distributed among a number of distinct operating entities within a single large company or among a group of independent companies (Goldman, Nagel & Preiss, 1995, p.7).

✓ Less a discrete enterprise and more an ever-varying cluster of common activities in the midst of a vast fabric of relationships (Davidow & Malone, 1993, p.4-7).

I tend to subscribe to the definition “an alliance of core competencies distributed among a number of distinct operating entities among a group of independent companies”.

Saabeel et al. (2002) tends to stress the highly dynamic nature of the
relationships. However, in my mind, the relationships would not change so quickly – at least not initially. “You have to learn to walk before you can run.” According to Saabeel et al. (2002), sometimes described as “virtual”, “hybrid”, or “lean”, virtual organizations are built ad hoc from small, sometime globally dispersed, independent organizational entities. These organizational entities re-shape themselves dynamically as customer requirements change or as the environment evolves. Contributing entities are added to the network when they can add value and are disengaged as their competencies are no longer required. This poses a specific set of requirements on the capabilities and resources of organizations that take part or want to take part in a virtual organization.

In my experience, now, if it is going to be possible for operating entities (say, the financial management or distribution functions) of companies to participate in virtual organizations, then (at least) two things have to occur:

- First, the physical organization has to decentralize authority and responsibility to their different constituent operating entities. Business processes may have to be re-designed and/or re-aligned (with organizational boundaries) such that these entities can operate more or less independently.

- Second, the IT capabilities (including architecture) supporting the business processes of each of these entities must also be as independent as possible from the rest of the physical organization. This can result in an absolutely
profound change in the way we think about enterprise architectures. I think the enterprise architecture should be made of modules, each of which will correspond to the boundaries of a particular operating entity. The degree to which these modules are independent of each other will influence the ability of the operating entity to participate in virtual organizations with operating entities of other physical companies.

I think the ideal is, for any organization, to be able to re-design and re-cast itself in terms of a number of almost independent operating entities. Each operating entity would participate in the business of the physical company, but would also be free to offer its services externally.

The profitability of a physical company would basically be the sum of the profitabilities of its contained operating entities. And, the profitability of each entity would be directly proportional to the number of virtual organizations in which it participates.

On an ongoing basis, the different operating entities can be evaluated, (similar to the way in which GE evaluated its lines of business when Jack Welch first took command). Any operating entities which are less than world-class are either improved to that level of quality or are decommissioned and are replaced by a comparable (in functionality), but world-class (in quality) entity from somewhere else.
The optimum situation exists when each organization consists only of those functional entities in which it has developed world-class capability. It “contracts-out” for all other needed capabilities to organizations that have developed those particular capabilities to a world-class level.

I think that enabling virtual organizations to work well will be the primary motivation for both business process re-design and enterprise architecture – two areas of endeavor that have generally faltered in the past. Perhaps “virtual-organizations” is the kind of ‘killer app’ (although it is not an app) that we have been looking for, to give direction and meaning to this work.

The point that I am trying to put across is that software architecture is very crucial part of building software systems, is affected by multiple dimensions, and hence deserves dedicated research and greater attention within organizations. I will not discuss more issues related to virtual organization since this is not within the scope of this thesis.

5. **IMPORTANACE OF ARCHITECTURE**

History reveals architectural style as the most important means of efficient, high-level communication among developers (Hubert, 2002). Culture of any organization /
enterprise plays a crucial role in establishing the ways in which systems evolve and are managed / maintained, especially when we look at multi-national global corporations. Misunderstanding and cultural clashes originate from communication barriers. Architecture facilitates effective communication across virtual organizations. Accordingly, introducing an IT-architectural style is one of the best investments an organization can make toward business optimization in the Information Age (Hubert, 2002).

No global standards can be enforced within a large organization. In fact, there is rarely a single unified culture that exists in such large organizations that span over continents; different sub-organizations sometimes have different cultures, and this can cause tension whenever any attempt is made to impose any global standards in these organizations. The larger the organization, the more apparent this will be and in the case of virtual organizations, it is almost guaranteed that there will be significant differences in the values and principles which will affect architectures and development efforts. In the face of such challenges, there is a great need to have some kind of modular architecture, which will give different sub-organizations the freedom they need to evolve as they see fit. A good analogy is one of proven software engineering principles; taking the principles of high cohesion and low coupling of program modules and applying them to architecture modules with the boundaries of the architecture modules coinciding with the boundaries of the sub-organizations. I think an overall enterprise architecture that has these properties will likely be more resilient in the face of change because you will not
have changes experienced in one part of the organization inadvertently impacting other parts of the organization through a misguided attempt to standardize everything globally.

6. ARCHITECTURAL & MANAGEMENT FRAMEWORKS

We can see why architecture is the most important aspect of building enterprise scale systems. In this section, we will analyze some of the more recent approaches to architecting systems. In this section, I will briefly outline the latest trends. In Chapter # 4, I will present a detailed technical analysis of the frameworks presented in this section.

6.1 ZACHMAN FRAMEWORK

John Zachman has been promoting the concept of Enterprise Architecture for over 25 years. The Zachman Framework for Enterprise Architecture is an approach for documenting and/or developing an enterprise-wide information systems architecture. This framework provides multiple perspectives of the overall architecture and a categorization of the artifacts of the architecture.

We will analyze this framework in detail in Chapter 4.
6.2 **MODEL DRIVEN ARCHITECTURE (MDA)**

Model Driven Architecture (MDA) provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon Object Management Group’s (OMG) established standards \(^6\), MDA aims to separate business or application logic from underlying platform technology. Platform-independent applications built using MDA and associated standards can be realized on a range of open and proprietary platforms, including CORBA, J2EE, .NET, and Web Services or other Web-based platforms. Fully-specified platform-independent models (including behavior) can enable intellectual property to move away from technology-specific code, helping to insulate business applications from technology evolution, and further enable

---

\(^6\) **Key standards** that make up the MDA suite of standards include Unified Modeling Language (UML); Meta-Object Facility (MOF); XML Meta-Data Interchange (XMI); and Common Warehouse Meta-model (CWM).
interoperability. In addition, business applications, freed from technology specifics, will be more able to evolve at the different pace of business evolution\(^7\).

\(^7\) [www.omg.org/mda](http://www.omg.org/mda)
In this chapter, I will present a critical analysis of some of the key technologies / frameworks including those briefly outlined in the last chapter. A well-defined Enterprise Architecture will include techniques such as knowledge management, business (re)engineering, data warehousing, and alignment of business and IT strategy – all techniques crucial to the success of an e-commerce venture (Popkin, 2003). From this definition of an Enterprise Architecture, we can easily identify the two most important factors, discussed below, of successfully visualizing, analyzing, designing, implementing / deploying, maintaining, and managing Enterprise Systems. These include:
Firstly, managing the great deal of information that is inherent in such large scale systems; how do you make sense of all of it and decide how much of it to use and how? That is the role of Enterprise Architectural Framework.

Secondly, the architectural design itself, which has to be long-lived in terms of portability, integration with new systems, and manageability.

There is one central problem that I have seen common in numerous engagements with large organizations: How to effectively instill enterprise architectural concepts into the entire organization and make it part of the evolving culture? How to ensure and encourage everybody to understand its importance and work constructively towards the common strategic goal and business objective? My discussion in this chapter will be an effort to effectively discuss numerous frameworks (technology, organizational, as well as business process related). As a result of this discussion, I will present new enterprise system architectural and management solutions with the hope to address the challenges presented above in the next chapter. In this chapter, I will primarily focus most of the discussion around Zachman Architectural Framework, Object Management Group’s (OMG) Model Driven Architecture (MDA) and its current implementation strategies including Convergent Architecture.
1. ZACHMAN FRAMEWORK

The framework as it applies to Enterprise is simply a logical structure for classifying and organizing the descriptive representations of an Enterprise that are significant to the management of the Enterprise as well as to the development of the Enterprise’s systems (Popkin, 2003). It was derived from analogous structures that are found in the older disciplines of Architecture / Construction and Engineering / Manufacturing that classify and organize the design artifacts created over the process of designing and producing complex physical products (e.g., buildings or airplanes).

‘Zachman Framework’ (Zachman, 1987) is a matrix which describes the various ways the stakeholders of an enterprise view the business and its systems. According to ZIFA (2003), Zachman Framework is simply a classification scheme for descriptive representations of anything, subjects or objects. It characterizes architecture in terms of the perspectives of the different stakeholders (represented by rows in the matrix) and focuses on the different aspects of architecture (represented by the columns).

The framework itself is both primitive and comprehensive. The rows represent, successively, the planner, owner, designer, builder, and subcontractor perspectives. The columns reflect the aspects of data, process, location, role, timing, and motivation. Each cell in the framework (shown in Figure 4.1) represents the intersection of a particular focus and a perspective. Each focus (the question what, how, where, who, when, and why which six millennia of linguistic experience suggest are the total relevant set) is depicted
in a column and each perspective (point of view) in a row (ZIFA, 2003). Each question is unique and if one answers all six questions about any subject or object, there are no further questions to be asked, that is, it is comprehensive. Over the years, it has been shown that it is primitive in the fact that these interrogatives cannot be factored any further (ZIFA, 2003). They are not divisible. They are primitive (ZIFA, 2003). The perspectives define the point of view or the level of abstraction for the information contained in the cell. If you look across all of the cells in a single perspective, you will see all of the Enterprise’s knowledge from that perspective. According to Popkin (2003), if the Framework is properly utilized the information and models within a single row will represent a complete description of the Enterprise from that perspective. At the same time, each column captures all of the Enterprise’s knowledge for the particular question being asked, i.e., the focus. Total Enterprise knowledge is built for each focus by isolating each focus and defining the artifacts for each perspective within it.

I will now explain in detail what the matrix presented above represents. The rows at the top are the most abstract and are oriented towards very broad goals and plans. If we were building a house, this layer would describe the diagrams, pictures and plans the architect would discuss with the owner. The next level is more specific, but still abstract. These are the diagrams that the architect would discuss with the contractor. In a similar way, the top level of the Zachman framework, labeled “Scope,” is focused on the concerns of senior executives. The second level focuses on the slightly more detailed concerns of business managers. Level three focuses on concerns that business and IS
managers often work on together. Levels four through six focus on details that IS managers and software developers are concerned with (Harmon, 2003).

Figure 4.1: The Zachman Framework

If we look across the top row of the framework, we can see the types of issues Zachman expects managers to consider as they define the organization. He expects that executives will create lists of things important to the business, lists of processes the business performs, lists of locations at which the business operates, and so forth (Harmon, 2003). As we go down the rows, we can see that they are organized as levels. As we work your way down through the framework, we can see how lower level managers focus on the same general topics, but in more specific detail. The top row, labeled SCOPE, focuses on
documents that senior managers and planners would normally use. The plans and documents become increasingly specific and detailed as you drop lower. The bottom layer refers to actual data, specific applications, all the physical structures, and the people that comprise the business (Harmon, 2003).

A second glance at the matrix will reveal the various kinds of specific architectures that Zachman includes within his overall framework. The cell that represents the intersection of Function and Systems describes the organization’s application architecture (Harmon, 2003). And the cell that represents the intersection of the Network and the Technological model represents the technology architecture, which describes the hardware used and the links between the platforms. Zachman’s Framework is popular because it provides a comprehensive overview and assigns a distinct name to each of the cells in his framework (Harmon, 2003).

Zachman’s Framework presents a methodological way of looking at an opportunity (technical, functional, & strategic) at different levels. However, it does not discuss how the systems should be built (technical frameworks) that ensures the adaptability of the system in the face of constantly changing competitive landscape.

From the discussion above, we can see that Zachman’s Framework seems to be an excellent tool to get the thinking process going, especially when it comes to complex objects. I would like to call it a Problem Solving Toolkit to solve complex problems without losing sense of the bigger picture.
We will now look at a framework that deals more with the technical issues and can be adopted within Zachman’s Framework.

2. **MODEL DRIVEN ARCHITECTURE**

2.1 **BACKGROUND**

It’s all about integration. Object Management Group’s (OMG) CORBA (Common Object Request Broker Architecture) has focused on creating a truly interoperable and integrated computing environment (Siegel, 2001). CORBA has brought object technology into IT mainstream and is now one of the standards for enterprise interoperability, allowing objects to interoperate smoothly across many former boundaries such as hardware platform, operating system, and programming language. Vendors have implemented these specifications widely over the past ten years, most notably OMG’s CORBA specification. In fact, organizations polled in a recent analyst survey, conducted by Gartner, confirmed this by ranking CORBA compliance as the most important consideration in choosing middleware for application integration. By providing transparency to operating system, programming language and even network protocol, CORBA provides the adapter for the information plug, breaking down barriers.

---

8 A surprising 70 percent of respondents cited CORBA compliance as “important” or “very important” to integration, outpacing every other factor in the survey, including core functions such as integration of legacy applications with distributed systems and corporate intranets. Summary of responses from 547 organizations asked to rate middleware selection criteria in the context of application integration in Middleware: what end users are buying and why. Gartner Group, February 1999.
to interoperability at enterprises around the world. However, CORBA has its limitations. Enterprises with different platforms supporting hundreds of (if not thousands) different applications face a challenge of integrating these applications even though it is time consuming, risky and expensive (Architecture Board ORMSC, 2001). There are other middleware platforms available, each providing nearly the same set of services in its own way, with its own advantages and disadvantages and have become entrenched in enterprise computing. According to Siegel (2001), establishment of standard, vendor- and system-independent middleware technology did not provide the environment that enterprises need to reap the full benefit of their heterogeneous computing hardware and software.

To provide the kind of integration that today’s corporate computing environment requires, the OMG has added a level of standardization upward from application implementation to the level of application design (Siegel, 2001). OMG is building a model environment based on the common features of the various middleware platforms. Exploiting their fundamental commonality, without being distracted by their superficial differences, allows the OMG to define a Model-Driven Architecture (MDA) in which applications on your chosen middleware platform can be made to interoperate smoothly with those of your other departments, your customers, your suppliers, and everyone else you do business with, regardless of the middleware architecture they choose and use (Siegel, 2001).

---

9 This model is realized as a UML Profile, and the technically savvy will recognize that it is actually meta-model of the middleware environment. Additional profiles will be defined for other specialized computing environments such as Real-Time computing and embedded systems (Siegel, 2001).
2.2 VISION & CONCEPT OF MDA

MDA provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon OMG’s established standards\(^{10}\), MDA aims to separate business or application logic from underlying platform technology. Platform-independent applications built using MDA and associated standards can be realized on a range of open and proprietary platforms, including CORBA, J2EE, .NET, and Web Services or other Web-based platforms. MDA:

- Integrates what has been built, with what is being built, and all what will be built in future.
- Remains flexible in the face of constant changing infrastructure; and
- Lengthens the usable lifetime of your software, lowering maintenance costs and raising ROI.

Figure 4.2 diagrams the MDA (Siegel, 2001). We will discuss each layer of this model in detail as follows.

---

\(^{10}\) Key Standards that make up the MDA suite of standards include Unified Modeling Language (UML); Meta-Object Facility (MOF); XML Meta-Data Interchange (XMI); and Common Warehouse Meta-model (CWM).
2.2.1 The Core

At its core is a technology-independent definition of the distributed enterprise computing infrastructure. Built in OMG’s UML (Unified Modeling Language), it includes the concepts that we recognized as common to the various architectures on the market, representing component-based systems such as Enterprise Java Beans (EJB), and loosely-coupled messaging-based systems including Web Services.

Unified Modeling Language (UML) addresses the modeling of architecture, objects, interactions between objects, data modeling aspects of the application life cycle, as well as the design aspects of component based development including construction and assembly (Architecture Board ORMSC, 2001). UML allows an application model to be constructed, viewed, developed, and manipulated in a standard way at analysis and design time. UML models represent the design for an application, allowing business functionality and

(Source: Architecture Board ORMSC, 2001)

Figure 4.2: OMG’s Model-Driven Architecture
behavior to be represented clearly by business experts at the first stage of
development in an environment undistorted by computing technology (Siegel,
2001). This allows the design to be evaluated and critiqued when changes are
easiest and least expensive to make, before it is coded. Note that UML is
powerful enough to be used to represent artifacts of legacy systems. In fact,
MDA development starts with the construction of a Platform-Independent
Model (PIM; discussed in detail later in this chapter) in this technology-
independent UML environment. Unlike conventional development methods
which use this model as a basis for hand-coding by a team of developers,
however, the MDA uses the UML model to automatically generate all or most
of the running applications via the series of steps that we will detail in this
chapter, maximizing the return from modeling investment. According to
Architecture Board ORMSC (2001), artifacts captured in UML models (in
terms of Classes, Interfaces, Use Cases, Activity Graphs etc.) can be easily
exported to other tools in the life cycle chain using XMI. A number of UML
profiles (for CORBA, EJB, EDOC etc.) are at various stages of
standardization (UML profile for CORBA is adopted). These are critical links
that bridge the UML community (model based design and analysis) to the
developer community (Java, VB, C++ developers), middleware community
(CORBA, EJB, SOAP developers) etc. Additional profiles focused on systems
and application management are needed.
Three key OMG modeling technologies (presented in greater details in the Glossary), all based on UML, support the MDA (Siegel, 2001):

✔ **The Meta-Object Facility** (MOF) which not only provides a standard repository for the model, but also defines a structure that helps multiple groups work with the model and view in a standard way;

✔ **The Common Warehouse Metamodel** (CWM), the established industry standard for data repository integration, standardizes how to represent database models (schema), schema transformation models, OLAP, and data mining models.

✔ **XML Metadata Interchange** (XMI), a mapping which expresses UML models in XML and allows them to be moved around our enterprise as we progress from analysis to model to application.

### 2.2.2 Middleware Layer

The next closest ring to the core of Figure 4.2 is what is called as middleware layer. This layer presently comprises of some of the most popular middleware platforms such as Web Services environment, CORBA (in particular the CORBA
Component Model or CCM); Java (including Enterprise JavaBeans); C#/NET; XML/SOAP\textsuperscript{11}.

\subsection*{2.2.3 Pervasive Services Layer}

In Figure 4.2 above, the Pervasive Services are shown as a ring around the outside of the diagram to emphasize that they’re available to all applications, in all environments. Pervasive Services specify services that an infrastructure component offers to other assembled components (Architecture Board ORMSC, 2001). Enterprise, Internet, and embedded computing as well as true integration rely on a set of essential services. The list varies somewhat depending on the source but typically includes Directory services, Event handling, Persistence, Transactions, and Security (Soley, 2000).

When these services are defined and built on a particular platform, they necessarily take on characteristics that restrict them to that platform, or ensure that they work best there. To avoid this, OMG will define such services as \textit{Pervasive Services} at the PIM level in UML. Only after the features and architecture of a pervasive service are fixed will platform-specific definitions be generated for all of the middleware platforms supported by the MDA (Architecture Board ORMSC, 2001).

\textsuperscript{11} According to Siegel (2000), although each of these are regarded as “middleware” by at least some observers, they provide widely varying degrees of functionality and service. Clearly, those that provide minimal service levels (such as XML, which is actually a data format) will have to be combined with additional technology in order to be treated as an MDA target platform.
At the abstraction level of a platform-independent business component model, pervasive services are visible only at a very high level (similar to the view the component developer has in CCM or EJB). When the model is mapped to a particular platform, code will be generated (or dynamically invoked) that makes the calls to the native services of those platforms (Architecture Board ORMSC, 2001).

2.2.4 **Vertical Markets or Domains Layer**

The outermost and largest ring, dominating the diagram with its compass points, depicts the various vertical markets or domains whose facilities will make up the bulk of the MDA (Siegel, 2001). Since January 1996, a sizeable percentage of OMG members have been meeting in Domain Task Forces (DTF), communities focused on standardizing services and facilities in vertical markets. I personally feel that it was a great initiative by OMG i.e., standardizing components at a platform level, in terms of standards such as CORBA, is certainly a viable contribution to solving the integration and interoperability problem. The DTFs produce standard frameworks for standard functions in their application space.

For example, a Finance DTF standard for an Accounts Receivable Facility might include a platform-independent UML model, a CORBA-specific UML model, IDL interfaces, a Java-specific UML model, and Java interfaces. Platform-
independent UML models act as a knowledge-base for a particular facility. These models can be translated into other middleware platforms which are not mentioned here or which are still to come in future.

XML DTDs or schemas generated via XMI-based mapping rules could be included as well. According to Soley (2000), all of these artifacts would be normative. Such a standard would have broad impact, in that the platform-independent model would be useful even in middleware environments other than those targeted by the platform-specific parts of the specification. Since Accounts Receivable is an Enterprise Computing application, the normative, platform-specific artifacts would be derived at least partially via standard mappings of Enterprise Computing core model to the platforms (Soley, 2000).

2.3 **MODELS IN MDA**

The MDA separates certain key models of a system, and brings a consistent structure to these models i.e., Platform Independent Model (PIM) and Platform Specific Model (PSM). The PSM can be generated automatic or semi-automatic from PIM with the support of some platform specific directives. The PIM will be expressed in UML describing the concepts in the problem-domain that also includes concepts from a reference architecture which is relevant for this problem domain (Nordmoen, 2001). This is along the same ideas as expressed by Michael Jackson in his recent book "Problem Frames" (Jackson, 2001), in which he suggests to organize and classify the problems into
well defined groups and then apply a solution patterns that is relevant for that group of problems.

Whether the ultimate target is CORBA Component Model (CCM) Enterprise Java Beans (EJB), Microsoft Transaction Server (MTS) (more discussion in Glossary), or some other component or transaction-based platform, the first step when constructing an MDA-based application will be to create a platform-independent application model expressed via UML in terms of the appropriate core model. Platform specialists will convert this general application model into one targeted to a specific platform such as CCM, EJB, or MTS. Standard mappings will allow tools to automate some of the conversions (Soley, 2000). The whole process is still mostly manual and a lot of development in tools is required.

### 2.3.1 Platform-Independent Model

Fully-specified platform-independent models (including behavior) can enable intellectual property to move away from technology specific code, helping to insulate business applications from technology evolution, and further enable interoperability. In addition, business applications, freed from technology specifics, will be more able to evolve at the different pace of business evolution (OMG MDA, 2003).
The PIMs provide formal specifications of the structure and function of the system that abstracts away technical details. A Platform Independent Component View describes computational components and their interactions in a platform-independent manner (Architecture Board ORMSC, 2001). These components and interfaces, in turn, are a way of realizing some more abstract information system or application, which itself helps realize a computation-independent Business Model. OMG standards are specified in terms of a PIM and, normally, one or more PSMs, all in UML (Architecture Board ORMSC, 2001) (PIMs & PSMs will be discussed in greater details with an example in Section 2.4). Abstracting out the fundamental precise structure and behavior of a system in the PIM from implementation specific concerns in the PSMs has three important benefits:

- It is easier to validate the correctness of the model uncluttered by platform-specific semantics.
- It is easier to produce implementations on different platforms while conforming to the same essential and precise structure and behavior of the system. Even further, the business model defines business goals and policies in a computation-independent manner.
- Integration and interoperability across systems can be defined more clearly in platform-independent terms, then mapped down to platform specific mechanisms.
Where appropriate generic mappings or patterns can be shared across multiple applications, it may be possible to automatically transform a PIM, perhaps after annotating it with some platform information, to different target PSMs either fully or partially (Architecture Board ORMSC, 2001). The transformation tools are still under development. MDA based development, discussed in Section 2.4, certainly provides an edge over any attempts in the past in that it combines methodology and tools together while integrating numerous middleware environments.

### 2.3.2 Platform-Specific Model

The semantics of the platform-independent original model are carried through into the platform-specific model. A platform-independent model (PIM) is a formal specification of the structure and function of a system that abstracts away technical detail. Platforms themselves also have a specification and an implementation. A platform specific model (PSM) is expressed in terms of the specification model of the target platform (Architecture Board ORMSC, 2001).

MDA defines consistent relationships across these models. For a given system, there are cross-model refinement correspondences between business model, platform-independent components, and platform-specific components. Similarly, across two different systems to be integrated (e.g., in EAI), interactions may be

---

12 Note that it is explicitly written “A platform-independent model...”. There could be models of a system from multiple viewpoints, at multiple levels of abstraction, all of which are platform-independent e.g., RM-ODP enterprise, computational, and engineering viewpoints.
specified at platform-specific, platform-independent, and even business model levels of abstraction.

2.4 DEVELOPING IN THE MDA

In this section, I will discuss how the actual development is done in the MDA. I will take two most common scenarios, which are as follows:

- Single Target Platform
- Multiple Target Platforms

I will take the case from Siegel (2001) that will demonstrate how a server can be generated using MDA. Then, for the multiple platforms, I will show how the MDA re-uses the same mechanism. I will focus on Web Services (discussed in Glossary) for this case.

2.4.1 Single Target Platform

Platform Independent Model (PIM):

MDA development projects start with the creation of PIM (discussed in Section 2.2.1), expressed in UML and shown at the top of Figure 4.3. The base PIM expresses only business functionality and behavior. Built by business and modeling experts working together, this model expresses
business rules and functionality undistorted, as much as possible, by technology (Siegel, 2001). Because of technological independence, the base PIM retains its full value over the years, requiring change only when business conditions mandate. Some of the standard modeling infrastructure that incorporates all of this behavior into the PIM already exists: Object Constraint Language (OCL), a part of UML, lets designers specify innovation pre- and post-conditions very precisely in their model, so these will surely be included in this category. Other constraints that will carry through include whether a single-valued parameter is allowed to be null, and restrictions on combinations of attribute values (Siegel, 2001). The PIMs that have been produced in the first step of an MDA development specifies functionality and behavior for both client and server, and links to the Pervasive Services, Domain Facilities, and other MDA-modeled facilities that the application invokes (Siegel, 2001).

✓ **Platform Specific Model (PSM):**

Once the first iteration of the PIM is complete, it is stored in the MOF and input to the mapping step which will produce a Platform-Specific Model (PSM) as shown in the second row of Figure 4.3. Specializations and extensions to UML give it the power to express both PIMs and PSMs. During the mapping step, the run-time characteristics and configuration information that we designed into the application model in a general way are converted to
the specific forms required by our target middleware platform (Siegel, 2001). Mapping of these models has been discussed in detail in Section 2.5.

(Source: Siegel, 2001)

Figure 4.3: Using the MDA to generate a CCM Server
A tool restricted to a constrained environment (one that is used only for financial services and banking applications, for example, such as the one produced by Wells Fargo Bank described at www.omg.org/mda/mda_files/MDA%20briefing%20Castain.pdf) will produce complete or nearly complete PSMs. As tools mature, the process of automation will become stronger and stronger. We will have discussion specific to code generation in Section 2.4.3. I will also present one of the initiatives taken by Interactive Objects\(^\text{13}\) towards promoting MDA based development.

### 2.4.2 Multiple Target Platforms

Because the core MDA modeling environment was designed, from its inception, to support multiple platforms, OMG is defining mappings to many middleware targets. By adding various client platforms’ characteristics, the MDA can be made to generate code for these targets as well, regardless of differences in calling pattern from server to any number of client types (Siegel, 2001). These mappings are the key to MDA’s utility. Figure 4.4 below shows the process to carry out mappings to a number of middleware server platform targets.

---

\(^\text{13}\) Interactive Objects Software GmbH, founded in 1990, has built a solid reputation as a front-runner in the field of IT architecture by consistently delivering award-winning architectural platforms and services to global industry leaders. In 2000, Interactive Objects introduced the ArcStyler architectural platform. The ArcStyler is an Architectural IDE which supports Model Drive Architecture (MDA) as standardized by the Object Management Group (OMG). The ArcStyler supports UML-driven development including complete four-tier J2EE/EJB systems, .NET systems, custom infrastructure systems and Web Services for the leading application servers. The ArcStyler maintains a clear focus on adding architectural value by complementing best-of-breed tools such as Rational Rose and JBuilder, instead of competing with them. Not only does the ArcStyler boast a rapidly growing list of repeat customers, it is also the winner of numerous national and international awards (www.io-software.com).
2.4.3 Code Generation

Code generation part itself is very crucial and will be discussed in greater details in this section. For components environment, the system will have to produce many types of code and configuration files including interface files, component definition files, program code files, component configuration files including interface files, component definition files, program code files, component configuration files, and assembly configuration files (Soley, 2000). The more
complete the platform-specific UML dialect reflects the actual platform environment, the more completely the application semantics and run-time behavior can be included in the platform-specific application model and the more complete the generated code can be.

In a mature MDA environment, code generation will be substantial or, perhaps in some cases, even complete. Early versions are unlikely to provide a high degree of automatic generation, but even initial implementations will simplify development projects and represent a significant gain, on balance, for early adopters, because they will be using a consistent architecture for managing the platform-independent and platform-specific aspects of their applications (Soley, 2000).

Figure 4.4 shows that in its third row, an MDA tool for Web Services will generate all of the files that are required for the platform. Because Web Services may run on almost any application server, one particular platform needs to be specified (the one presently supported by MDA tools). In addition, if the target Application Server supports multiple programming languages, that needs to be identified as well. Immediately following code generation, the programming staff will apply any required hand-coding to the output (Siegel, 2001). One important part during this process certainly is the concept of Managing Business Rules and implementing those within the application code. I will present my work in the
area, that is expected to provide an efficient way to manage and implement the business rules in Section 4.

2.5 MAPPINGS OF MODELS

Figure 4.5 presents a metamodel description of MDA (Architecture Board ORMSC, 2001). In MDA, one of the key features of the whole approach is the notion of mapping. A mapping is a set of rules and techniques used to modify one model in order to get another model. According to Architecture Board ORMSC (2001), mappings are used for transforming:

- **PIM to PIM**: This transformation is used when models are enhanced, filtered or specialized during the development lifecycle without needing any platform dependent information. One of the most obvious mappings is the analysis to design models transformation. PIM to PIM mappings are generally related to model refinement.

- **PIM to PSM**: This transformation is used when the PIM is sufficiently refined to be projected to the execution infrastructure. The projection is based on the platform characteristics. Describing these characteristics should be done using a UML description (and eventually a profile for describing common platform concepts). Going from a logical component model to a
commercial existing component model (like EJB for J2EE platform or CCM for CORBA platform) is a kind of PIM to PSM mapping.

✓ **PSM to PSM.** This transformation is needed for component realization and deployment. For example, component packaging is done by selecting services and preparing their configuration. Once packaged, the components delivery could then be done by specifying initialization data, target machines, container generation and configuration, etc. PSM to PSM mapping are generally related to platform dependent model refinement.

✓ **PSM to PIM.** This transformation is required for abstracting models of existing implementations in a particular technology into a platform-independent model. This procedure often resembles a "mining" process that is hard to be fully automated. It may be supported by tools, though. Ideally, the result of this mapping will match the corresponding PIM to PSM mapping.

Termed as UML Profile, a standardization set of extensions (consisting of stereotypes and tagged values) defines a UML environment tailored to a particular use, such as modeling for a specific platform (Siegel, 2001). UML profiles have an important role to play in MDA, since MDA leverage usage of mappings between models. According to (Architecture Board ORMSC, 2001), to implement the mapping, one needs to know the metamodels of the input and output models.
and their mapping rules. Transformation rules execution can be done inside UML tools (for example using scripting) or by external tools (for example, by working on XMI files).

Figure 4.5: PIM-PSM Mappings

2.6 INTEGRATION WITH LEGACY & EXISTING SYSTEMS

Most part of this chapter discusses building applications from scratch. However, that is not most part of the challenge facing global enterprises today. The challenge is integrating new systems with existing ones so that they seamlessly talk to each other. Legacy applications are amongst many applications that do not
fit the MDA pattern. With many built before component environments were even conceived, applications in this category typically will not fit smoothly into any of MDA core models. According to OMG’s recommendation, legacy applications may be brought into MDA by wrapping them with a layer of code consistent with the MDA Enterprise component core model, or another core model if appropriate. MDA has been successfully used in numerous enterprises for generating new business applications and integrating those with existing legacy systems. One such case is the project conducted by Interactive Objects (www.io-software.com) in embedding existing Cobol Application into Modern Web-Based Systems at Deutsche Bank Bauspar AG (Interactive Objects, 2003).

3. **CONVERGENT ARCHITECTURE–MDA IMPLEMENTATION**

Convergent Architecture (CA) is a methodology that presents modern architectural style and its implementation using MDA. CA leverages MDA to resolve many complex IT-related problems at the source. I will present a critical analysis of CA approach, as discussed in (Hubert, 2002), in this section.
3.1 IT-ARCHITECTURAL STYLE

“An architectural style conveys the principles and the means to most effectively achieve a design vision.” According to (Hubert, 2002), an IT-architectural style comprises of the following four high-level features, or layers:

3.1.1 The Architectural Metamodel

The architectural metamodel is the top level model. It is a metamodel, meaning that when applied, it produces or influences another model as its result. In this particular case, the architectural metamodel influences practically every decision made in the entire architecture. It does this by defining the vision and principles of the architecture. It sets the fundamental judgment criteria for every design decision, analogous in many ways to a constitution, which sets the judgment criteria for legal decisions. The metamodel justifies why the things are done the way they are done throughout the architecture and provides the basis for individuals to make the proper decisions at all levels. It is the reference frame in which the architecture is refined and evolved over time. It also defines under which constraints such refinement and evolution take place.
The second level model is the development model, and it defines how we achieve the vision and fulfill the principles expressed in the architectural metamodel. It formulates and transports principles into concrete structures, such as components and development organizations, as well as procedures, such as the development process. These structures and procedures are the vehicle of the architectural style. According to (Hubert, 2002. p.19), to ensure adequate coverage (depth and breadth), the following three fundamental themes should be addressed by the development model:

- **The development structures theme.** This describes the concrete resources used to design, implement, and deliver the system. The focus here is on describing the structures to be built and the structures required along the design and development path.

- **The development process theme.** This describes the specific development tasks. These tasks focus on the creation and evolution of the development artifacts and are specifically supported by the tools and organization of the style. Its process should at least cover the entire critical development path, which must be defined by the style, including the repeat cycles necessary to address the change and evolution of the system.
properly. However, defining the process is not enough; it must be coordinated properly by an IT organization.

✓ **The IT-organization structure theme.** This defines how responsibilities, roles, and persons are best coordinated to simplify and support the specific development process. Often, methodologies neglect the intimate relationship between the process and organization, assuming that the process is complete. However, no matter how complete and well though out the process is, there is no way to cover everything that can possibly occur during a development effort.

### 3.1.3 The Full-Coverage Tool Suite

Based on specific requirements set forth by the preceding features of the IT-architectural style, the third feature of a style defines effective tools to support architecture-driven development. Due to the coverage and specificity of the development model, tools can be designed, integrated, or implemented to actively assist style-conforming development. They can be tuned specifically to intelligently support development according to the development model. With the previous definition of the feature of the style, a comparable range of coverage, intelligent support, and tuning would be impossible. Since specific requirements for tools are set in the model of the IT-architectural style, experts can be used to develop and tune the tools in one place to support all projects using the style. This
means that the time, costs, and risks associated with tool development are reduced. Moreover, the tools are more effective.

3.1.4 **The Formal Technology Projections**

As pointed out earlier, tools can be defined to support many more development tasks in the context of an IT architectural style than would be possible without a well defined style. Above all, significant new levels of automation are possible. Today the accepted level of automatic construction is the compiler. There is no reason why this process cannot evolve to the level of model driven projections to entire server platforms. Examples of such platforms would be middleware infrastructures, application server infrastructures, or even mainframe infrastructures. Model-driven technology projection simply means that high-level models are translated to entire IT infrastructures instead of just translating source code to machine code. Such higher-level generators cover much more ground than the source code based compilers while delivering comparable dependability and quality. There is no downside to this scenario if it is positioned properly as part of an overall IT architectural style. However, if not used in the context of an IT architectural style, such generators just become a faster way to produce ad hoc architecture.
3.2 **ROADMAP & ANATOMY OF CONVERGENT ARCHITECTURE**

Figure below shows the combined elements of structure, process, and tools that make up the CA (Hubert, 2002, p.34). This is both its anatomy and the top-level roadmap. Layers of abstraction run from top to bottom and then from left to right.

(Source: Hubert, 2002)

**Figure 4.6: Roadmap and anatomy of the Convergent Architecture**

---

14 RUP (Rational Unified Process) is a configurable software development process platform that delivers proven best practices and a configurable architecture that enable you to select and deploy only the process components you need for each stage of your project (Rational, 2003).
The layers of the CA and their correspondence to the four features of an IT-architectural style, as discussed in (Hubert, 2002), are as follows:

3.2.1 **The Convergent Architecture Metamodel**

This layer fulfills the role of the architectural metamodel, as discussed earlier. It comprises of the following elements:

- **The three pillars of holistic architecture**: The intimately related themes of project design, business design, and system design are addressed to provide adequate coverage of the areas critical to an IT organization and its many interrelated projects.

- **Convergence and Convergent Engineering**: Convergent Engineering demonstrates how business and IT models can be united into one, simplifying model to resolve many of today’s most complex operational and business-IT problems.

- **The machine ship metaphor**: The CA strives to achieve the level of a well-run machine shop by creating a comparable software shop.

- **Reduced Abstraction Set Computing (RASC)**: Based on years of empirical research, IT architects have been able to identify a reduced set
of design elements, or design abstractions, to significantly simplify many aspects of design and the design process. The principle of RASC says that the core abstractions of convergent engineering (that is, organization, processes, and resources) form the basic types with which we can model all business-domain aspects of a convergent system regardless of the actual business domain.

- **Conceptual isomorphism:** The ability to learn a concept once and apply it similarly in many situations is called conceptual isomorphism. The CA can be used equally across the diverse organizations. Widely usable design patterns often are good examples of conceptual isomorphism.

- **Component metamorphosis:** There are few physical or material constraints to developing software systems. They are indeed “soft” in that we can conceivably manipulate and grow our designs any way we want. The principle of component metamorphosis invokes an intentional analogy with the metamorphosis of a butterfly. Instead of developing software systems as done today, in radical heaves of translation and reformulation of information, they can be evolved through steady stages of information enhancement and growth, comparable with the metamorphosis of a butterfly. More details can be found in Hubert (2002).
3.2.2 The Development Model

The CA’s development model formulates the principles of its architectural metamodel in terms of specific design structures, a development organization, and a development process. Further details can be found in (Hubert, 2002).

3.2.3 The Full-Coverage Tool Suite (Architectural Integrated Development Environment)

The full-coverage tool suite is an integral part of the CA. The tightly integrated tools and automated assistants support immediate and effective construction of architecture-conform IT systems. This comprehensive approach to high-level architectural tools is called as IT-Architectural IDE (Integrated Development Environment).

In CA, the IT-architectural IDE is arranged into five major modules, as shown in the Figure 4.7. These modules are not just tool descriptions. More important, they are used by the CA to simplify understanding and application of its development style. An overview of each techniques, according to Hubert (2002), is presented as follows:

- **Convergent Business Object Modeler (C-BOM):** This module assists both the IT designer and the business-domain expert in their joint task of
requirements analysis and elaboration of the business structure and dynamics. It provides visual modeling assistance to help identify and document a convergent system using the RASC components.

✓ **Convergent Pattern Refinement Assistant (C-RAS):** This module picks up the results of the C-BOM and helps a designer graphically evolve the business model into a more detailed, more technically precise model representation in UML.

(Source: Hubert, 2002)

**Figure 4.7: The Modules of the IT-Architectural IDE**
✓ **Convergent UML Refinement Assistant (C-REF):** This model reads the results of the C-RAS and now presents the convergent component model at the level of an advanced UML modeler for further enrichment and tuning of the design. This is the point where system interaction and access is elaborated in detail using the standard UML.

✓ **Convergent Translative Generator (C-GEN):** This module reads the UML model in parts or in its entirety from the C-REF tool and generates the complete component infrastructure, including the environment for configuration, construction, and deployment of the convergent system. The generator translates the UML model to the particular infrastructure while preserving convergence.

✓ **Convergent Generator IDE (C-GEN-IDE):** This IDE is the visual development environment for a generator cartridge. The development of a cartridge can be regarded as meta-programming because the scripts developed here drive the translative generation of many other programs.

✓ **Convergent Implement, Deploy, and Test Environment (C-IX):** The models of the CA require that model-based development also cover the areas of user interaction and access to and from external systems.
3.2.4 The Technology Projections

A technology projection in the CA specifies how convergent components and other modeled elements are mapped (projected) to standard component frameworks and then to various implementation technologies (Hubert, 2002). There are two aspects to a technology projection:

✓ Things that can be generated automatically, effectively, and with reasonable effort

✓ Things that cannot be generated automatically

Both of these must be addressed to maintain architectural integrity because together they define the tangible results of the architecture.

4. RULES-BASED SYSTEMS ARCHITECTURE

So far, we have discussed numerous steps involved in building manageable and efficient software systems based on models primarily. The entire effort is focused on dealing with inefficiencies in current development methodologies and present framework(s) that are going to aid in building software systems more effectively. The concept of business rule, although not new, has been a challenge in organizations / enterprises where business rules change with the competitive landscape (true for most
for-profit organizations). I feel that the rules management of dynamic business rules can greatly reduce the effort put in by organizations to maintain business rules. With this in mind, I had interviewed some of the key technology architects at Interactive Objects (www.io-software.com). I discussed the Rule-Based Systems Architecture, as discussed later in this section, with them and had a great feedback. In fact, this is one thing that nobody has done enough work in.

How to capture the business rules in a fashion that would enable business managers to directly reflect the changes in business dynamics (due to competitive landscape and other factors) at the system level within a minimum amount of time?

In this section, I will present the architecture that I have implemented during my consulting engagements. I will discuss and critically analyze the Rule-based approach to designing and implementing information systems. I will discuss in detail the types of business rules and how they get captured within the information systems. We will see how a rule-based architecture can be utilized in addressing the problems arising from changes in the competitive landscape in maintaining the information systems.
4.1 DEFINITION OF A BUSINESS RULE

A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The business rules which concern the project are atomic i.e., they cannot be broken down further (Hay & Healy, 2000).

These rules may be viewed from two perspectives. From the business (‘Zachman row two’) perspective, it pertains to any of the constraints that apply to the behavior of people in the enterprise, from restrictions on smoking to procedures for filling out a purchase order. From the information system (‘Zachman row three’) perspective, it pertains to the facts which are recorded as data and constraints on changes to the values of those facts (Hay & Healy, 2000). That is, the concern is what data may or may not be recorded in the information system.

Business rule expresses specific constraints on the creation, updating, and removal of persistent data in an information system (Hay & Healy). For example,

IF the customer’s credit rating is not adequate

THEN the record of a purchase order must be declined

Note that this is captured within the application code in the format of “if … then … else” statements or other conditional statements.
4.2 TYPES OF BUSINESS RULES

I have been in IT consulting for almost seven years. From my experience so far, I have been able to categories the business rules related to information systems as follows:

- **Static Business Rules:** These rules, as the name dictates, tend to be more stable. Mostly, these rules reside in underlying data models for example each Department will have one Managing Director. Such rules do not change very often.

- **Dynamic Business Rules:** These rules vary more frequently and present challenges when it comes to updating these. These rules mostly vary depending on the ever changing competitive landscape, market, and other outside factors.

4.3 CHALLENGE

The challenge is that whenever there is change in dynamic rules, the respective implemented rules (in terms of if … then … else structure) need to be updated. It has to go through the entire lifecycle i.e., Business / Systems Analysis, Design, Implementation, Testing, Deployment, and Quality Assurance. By the time it gets
implemented, there is no more need for this change since the competition has changed again. This was the case at Softchoice Corporation (my last client) and I was able to come up with Rules-Based Architecture, specially for their Customer Relationship modules.

4.4 RULES-BASED SYSTEMS

My last engagement was with Softchoice Corporation and was working directly with the CEO & President of the firm. (Softchoice Corporation is one of Canada’s Top 50 Best Managed Companies, went public in July 2002, and the share price went up by over 200% ever since). The company had been using in-house built legacy system since 1989 because the CEO did not believe in out-of-the-box solutions primarily because of rigidity. He was extremely Customer Relationship focused and wanted to make certain that the company maintains its competitive advantage. As a result of research and successful implementation there, I put together the Rules-Based Systems Architecture, which we are going to discuss in the following sections.

Unlike most engineered systems in which *form* follows *function*, the flexible rule-based architecture presented ensures that *function* to follow the *form* i.e., we are not tied to the rules implemented by the vendor and in fact have a full control over the function as discussed below. The architecture of rule-based system encapsulates the ingredients (context, content, concept, circuitry, and character)
required for it to be considered an art as compared to an engineered system. These ingredients play the role in transforming value to an experience\textsuperscript{15}.

For the purpose of this discussion, I will be zooming into CRM system. My client (in this case Softchoice Corporation) differentiates itself in providing unique customer service that creates customer value and now wishes to transform that value to an experience for customer satisfaction. The out-of-the-box CRM systems have a collection of best practices. When you select one of these sub-systems, the vendor would normally want to re-engineer your business processes to accommodate the new CRM system. This system, though, implements some of the best practices, may not be required by our client. In addition, one of the serious drawbacks of these out-of-the-box systems is that the business rules are already implemented into the Application Layer (as shown in the Figure 4.8) of the system in the form of If…Then…Else statements. Such a structure makes it very challenging to modify the business rules (which are indeed in the form of If … Then … Else constructs or the like) with changes in

\textbf{Figure 4.8: Rules Extraction}

\textsuperscript{15} Presentation given by Steve Imrich (Member of American Institute of Architects and Associate Principal, C7A – Cambridge Seven Associates Inc.) during Systems Architecture course here at MIT.
competitive landscape. Normally, a request is generated by a business unit within the enterprise to update the business rule(s). That requirement(s) will then be analyzed by the Business Analyst, who models the requirement. Then it is analyzed by the Technical Analyst/Developer, who then makes the appropriate change(s) and so on. Such a process for implementing the requested change takes a long time and is undesirable in such a fast changing competition.

*(This is Ahmad Shuja’s model of Ideal Rule-Based Systems Architecture)*

**Figure 4.9**

To address this limitation, I architected a Rule-Based System that allows to extract the rules from the Application Code and maintain those in the Rules Repository as shown in the Figure 4.8. The overall process of this system is
shown in Figure 4.9. In a rules-based system, to modify or even completely change the rules will not take much effort. The Manager responsible for the particular business unit will be responsible for managing rules for that respective unit.

Figure 4.10: Rule Based Architecture

Figure 4.10 presents the overall architecture (I put it together) of the system that is based on Rule-Based Architecture. Note that Business Tier has been broken down into two layers i.e., Application Server and Dynamic Business Rule Management
System. Such a system is expected to support the process shown in Figure 4.9. This discussion presents initial concept that will allow efficient management of complex and ever changing business rules.

5. **CONCLUSION**

In this chapter, we have discussed couple of frameworks and architectural approaches to solving the emerging challenges in IT and software architecture. We are still far from what I feel will be the ultimate way of building systems. However, there is definitely a move in the right direction. The frameworks and solutions discussed in this chapter have some very obvious weaknesses including the following:

- Each framework / solution deals with individual challenges of building enterprise level software systems
- No single frameworks addresses the pre-requisites required for a long-term solution to the challenge of building systems that are completely aligned with the strategic objectives of the organization

In the next chapter, I will present initial level of unified approach that is anticipated to steer the future of enterprise software systems development and organizational design.
Chapter 5

UNIFIED SYSTEMS MANAGEMENT

FRAMEWORK

In the last chapter, we have seen a couple of frameworks, each addressing a particular challenge related to developing systems and managing the process. Zachman’s Framework provides a logical structure for classifying and organizing the descriptive representations of an Enterprise that are significant to the management of the Enterprise as well as to the development of the Enterprise’s systems (Popkin, 2003), whereas OMG’s MDA focuses on technical aspects of building custom software systems and provides an open, vendor-neutral approach to the challenge of business and technology change. Further, we discussed the Rules-Based Architecture, which takes us to code level details and presents an architecture that allows us to effectively manage business rules. The first intuition that comes into my mind is that these frameworks seem to be inter-related and inter-dependent. However, each focuses on different aspects of an enterprise and we will see how these are related.
There is no single framework that addresses such multi-dimensional challenges that are faced by Technology Managers and Leaders. The importance to have such frameworks increases as we realize that the need for custom built applications is ever increasing. According to (Gartner, 2003), instead of buying an application that works brilliantly, the job will be to assemble applications that work brilliantly together. In an interview, Greg Levine 16, Vice President (Technology Infrastructure) at Merrill Lynch, explicitly mentioned that most of the competitive edge is gained through custom built applications and that 70% to 80% applications are home grown and custom built. In another interview17 with Robert Holvey (Senior Vice President at Wells Fargo), he mentioned that about 1/3 of business application software is custom developed at Wells Fargo. Whereas, the license to the basic software of about 1/3 software is bought, which is then heavily customized to meet the business needs and 1/3 comes out of the packaged solutions like SAP, PeopleSoft etc. We can see that even in the case of Wells Fargo, 1/3rd is being built from scratch and for the other 1/3rd, the code is being bought and heavily customized, which brings it closer to what I call custom-built business application. In any case, from most of the interviews that I conducted, I found that between 66% - 85% of business applications are custom-built. Executives / professional that I interviewed believe that these custom-built applications actually add real value in contrast to packaged solutions.

16 I interviewed Greg Levine
17 I interviewed Robert Holvey
This gives us enough information that the importance and need for custom software will always be there. Packaged solution is treated as a commodity and competitive edge cannot be achieved / maintained by purely using commodity-like business application software. In view of this crucial need and the fact that there is no single framework available that would address the multi-dimensional challenges we face in the software industry today, I am going to present a unified framework that jointly represents numerous aspects of planning, implementing, and managing software systems. From our discussion so far, there are two major issues and important aspects that arise when talk about software systems. These can be categorized as follows:

- One issue is related to the actual architecture of the system – More technical aspect of the architecture

- The other is related to the processes that the business and technology professionals have to go through in order to make the most effective decisions, design successful technology strategies, and implement most efficient technological solutions to support the strategy in accomplishing the long-term goal of the enterprise.

From my readings of latest literature and breakthrough approaches, I have found independent approaches to the two problems but no one has actually presented a unified solution to the challenge. I believe that the first step towards meaningful and strategically successful implementation of IT solutions is the seamless integration of business strategy
with the underlying systems that would support that strategy. How to make this possible depends upon many factors, which form part of, what I want to call, Unified Systems Management Framework (USMF), shown in Figure 5.1. USMF is an attempt to present a bottom-up approach with the concrete management / organizational foundation to support the long-term successful design and implementation of IT systems. Unifying management related issues with software engineering related challenges is crucial. This fact is supported by McConnell (1996) and Burlton (1992). Management fundamentals have at least as large an influence on development schedules as technical fundamentals do (McConnell, 1996). The Software Engineering Institute (SEI) has repeatedly observed that organizations that attempt to put software-engineering discipline in place before putting project-management discipline in place are doomed to fail (Burlton, 1992). These management fundamentals consist of determining the size of the product (which includes functionality, complexity, and other product characteristics), allocating resources, and then monitoring and directing the resources to keep the project from heading into the weeds (McConnell, 1996). In many cases, upper management delegates these management tasks to technical leads explicitly, and in other cases it simply leaves a vacuum that a motivated lead or developer can fill.

An organizational-, process-, technology-roadmap is essential to any IT organization that must coordinate persons, projects, and ever-changing technology effectively over large periods of time. In fact, most IT organizations will require several levels and types of roadmaps to guide themselves properly along the precarious journey to effective systems (Hubert, 2002. p. 31). A unified framework, that is expected to support the architectural
as well as business process related requirements, will be presented here. It involves mapping of MDA core architectural models to Zachman’s Framework. Such mapping will form the initial study of the unified use of the two frameworks.

1. **ORGANIZATIONAL & MANAGEMENT INFRASTRUCTURE**

Organization and structure of an enterprise greatly influence the innovations carried out within that enterprise. According to Arnold Glasgow (Gartner), one of the basic tests of leadership is the ability to recognize a problem before it becomes an emergency. Recognizing a problem is a challenging task and management plays a crucial role in creating the kind of environment that promotes such analysis. Basic structure evaluates whether or not a company’s organizational design is conducive to achieving its strategic intent. It evaluates an organization’s structure and hierarchy in relation to its strategic plan (Business Experts, 2003). In support of this argument, Harmon (2003) says that the differences among company cultures and individual advocates assure that each company follows a more or less unique approach. The major challenge to every organization today is the transformation of its culture so that the organization can endure and grow through the current revolution (Schumann, Prestwood, Tong & Vanston, 1994). T.J. Watson, Jr. observed in Watson (1963) that:

> Technological change demands an even greater measure of adaptability and versatility of a management in the large organization. Unless management remains alert, it can be stricken with complacency – one of
the most insidious dangers we face in the business. In most cases, it’s hard to tell that you’ve even caught the disease until it’s almost too late. It is frequently most infectious among companies who have reached the top. They get believing in the infallibility of their own judgments.

The challenge is to be able to anticipate the shifts and change the organization constructively instead of waiting for the problems to get so large that only destructive change is possible. The management design greatly affects how the software / systems projects are managed. We will see later in this chapter how (Harmon, 2003) outlines the numerous steps that the organizations are involved in while creating an Enterprise Architecture.

Some of the important considerations while designing an organization are as follows (Business Experts, 2003). Note that each this is just a partial list of factors to be considered and an enterprise / organization needs to decide the kind of environment it needs over the long-term, which will impact the characteristics chosen.

✓ Structure should be conducive to strategy
✓ Structure facilitates a “value chain” approach
✓ Structure promotes collaboration between departments
✓ Structure adapts quickly to changing competitive environment
✓ Structure supports innovation and idea generation
✓ Structure can be re-organized when required
With regards to information, systems, and technology, these elements form the organizational design element that take into account the quality of an organization’s overall communications and the effective alignment of technology with strategic intent (Business Experts, 2003). Organizational design will vary from one organization to another, however, it is critical for executives and leaders to realize that the way the organization / enterprise is structured will play a crucial role in founding a creative and innovative environment and hence forms the founding layer of USMF as shown in Figure 5.1; the foundation of achieving strategic goals is organizational design.

**Unified Systems Management Framework**

Figure 5.1: Unified Systems Management Framework
2. **BUSINESS PROCESS DESIGN**

What do I mean by business process design? I like the definition given by Jacques Littre in Littre (2003) as follows:

*A business process can be a set of consecutive and/or simultaneous operations that contributes to the achievement or realization of a business objective or of a well defined business activity. The operations can be manual as well as automated or semi-automated and involve none, one or more actors/parties. A business process can be global (coarse) or specific (fine-grained). Coarse business processes may usually be decomposed into one or more sub-processes (or finer-grained processes).*

Business process design plays an important role in how effectively an organization deals with new idea generation and creativity. As mentioned by Greg Levine\(^\text{18}\) in his interview, *Managing Innovation* in a large organization is an extremely challenging process. It needs to be communicated very loud and clear that it is okay to come up with new and radically different ideas and people should be encouraged to do so. In fact, the business process should allow individual to be creative and be awarded for an innovation. At Merrill Lynch, a Interest Group (Community) is created around a new technology e.g., Handheld Transactional Devices / Applications. Such communities

---

\(^{18}\) Greg Levine is Vice President (Technology Infrastructure) at Merrill Lynch and I interviewed him on telephone.
would focus on achieving excellence in that particular domain and will come up with new ideas on how to utilize these technologies to gain competitive edge in serving customers / clients. Others approach such groups / communities to gain their insights when it comes to specific technologies. Such groups / communities also create standards for others to follow. Organizations certainly need the right kind of leadership and efficient business processes in place to actually allow such innovations to happen.

Numerous business processes that are involved in taking an idea or a concept from inception till its implementation rely heavily on the underlying structure of organization and design of management as discussed earlier. This is shown in the proposed *Unified Systems Management Framework (USMF)*. Business Process Design includes the cycle that the organization goes through before it is decided that a particular idea or concept will be implemented or just documented for future references. According to (Paulish, 2001), the process a project manager uses to do his or her job will greatly depend on the software development process of the larger organization. I will now present couple of examples that will show the variations of business process design. One of the examples is a case from (Harmon, 2003) as follows. In addition, I will present some of the finding of my interview with Robert Holvey (Senior Vice President, Wells Fargo).
2.1 THE ENTERPRISE ARCHITECTURE CYCLE

When we talk about Enterprise Architecture, one group must be responsible for maintaining the enterprise architecture. This isn’t to suggest that this group must create the architecture, but only that they must pull all the pieces together and maintain them so that anyone else can access the architecture. To remain neutral, we will assume that the group that maintains the enterprise architecture is the Enterprise Architecture Committee, a group that reports to the executive steering committee and maintains close relationships with the strategy group and those involved in business process redesign and improvement.

Figure 5.2 provides an overview of the enterprise architecture cycle that is based on the assumption that an organization has an Enterprise Architecture Committee that is ultimately responsible for the creation and maintenance of the enterprise architecture.

A process may be automated by a software application, but that application will probably have to rely on new or existing databases and it may require new infrastructure elements. The Enterprise Architecture Committee, working in conjunction with the IS organization, is responsible for initiating not only new applications to support new business processes, but also initiating data and infrastructure changes to support those applications as they come on-line in the future.
The Enterprise Architecture Committee will not create most of the documents and diagrams that are stored in the repository it manages. Other groups will create these documents in the course of doing their work. The Strategy Committee will create documents defining new strategies and goals. The IS organization will create designs for applications, databases and infrastructure. The Enterprise Architecture Committee, however, is responsible for integrating all of the information and assuring that each group is aligned with others in a satisfactory way.

(Source: Harmon, 2003)

**Figure 5.2: The Enterprise Architecture Cycle**
The process discussed above is just one example. Each organization can have a different process based on a number of factors including culture of the organization, resource pool, scale and scope etc.

### 2.2 **ITERATIVE APPROACH – WELLS FARGO**

In his interview, Robert Holvey (Senior Vice President at Wells Fargo) gives a variation of this business process discussed below. Robert said, “The particular business process adopted depends on the business problem, the skills of the group doing the development, and the style and direction provided by the manager(s) in that area.”

The common approach that is used by Robert in his part of the organization is an iterative process that goes generally as follows (with lots of variations on the general pattern depending on the business situation and time constraints and objectives):

- Starting with detailed discussions of the business objectives in business terms (to ensure that what gets built will help produce the desired business outcomes or improvements),

- Ranking of the alternative development projects to make sure that the development resources are always working on the projects that give the Company the best bottom line impact for our efforts
Assessment of the needs and skills of the eventual end users of the new application to make sure that the amount of user training and new skill development that’s required during the implementation phase is minimized (this is called as giving end-users tools they can use)

Development of a rapid prototype of the system and solicitation of user feedback on how well the application does in terms of achieving the implementation objective and how easy it is to use

Re-building to incorporate this feedback

Introduction of the application and any required user-training

Collection of ongoing feedback that is integrated into later releases in phases to keep the application evolving with our user’s needs and consistent with changing business objectives

All of the above steps are usually performed in conjunction with the completion of specific examples of business projects that use the new tool and the tracking of the P&L results of the business project. That way we can demonstrate not only how much the effort costs, but what business results the initial implementations produce. Often the first business-project based implementation provides great guidance to development decisions
and also demonstrates initial business results that more than pay for the cost of development.

When deciding on the new technology at Wells Fargo, the above process is adopted in addition to extensive piloting and testing of that technology prior to making widespread investments in Company wide deployment and implementation. Because of the very large costs of deployment and implementation on a large scale, Wells Fargo usually deploys small pilots and rigorously measures the results for the users and the impacts on the customers and on their bottom line. Continuing further, Robert said, “Technology that doesn’t meet the business criteria and objectives during the pilot is either put through additional cycles of development and new pilots until we’re ready to increase the project scope. Ultimately the business units pay for the technology, so there isn’t much in the way of free-wheeling R&D or innovation that isn’t focused on addressing specific business problems, challenges or opportunities.”

The discussion we have had so far shows variations of business processes required to incorporate new ideas into the product or encourage generation of new solutions.

3. **SYSTEMS MANAGEMENT STRATEGIES**

Management often controls all three corners of the classic trade-off triangle – schedule, cost, and product – although sometimes the marketing department controls the product specification and sometimes the development department controls the schedule
Management fundamentals consist of determining the size of the product (which includes functionality, complexity, and other product characteristics), allocating resources appropriate for a product of that size, creating a plan for applying the resources, and then monitoring and directing the resources to keep the project from heading into the weeds (McConnell, 1996). In many cases, upper management delegates these management tasks to technical leads explicitly, and in other cases it simply leaves a vacuum that a motivated lead or developer can fill (McConnell, 1996).

Systems Management Strategies play an important role in the successful implementation and management of software-based systems. This topic forms a thesis topic in its own right. I will, therefore, briefly discuss the numerous strategies in this section. The models discussed below are some of the most widely known and used. Table 5.1 shows a brief list and comparison of important attributes of some of the most famous models. Each has its own strength and weakness. However, each one will need a particular culture and team dynamics to be successful.

Some of the models discussed below include Rational Unified Process (RUP) (Kruchten, 2002), Architecture-Centric Software Project Management (Paulish, 2001), and others including Rapid Development (McConnell, 1996).
<table>
<thead>
<tr>
<th>MODEL</th>
<th>Development split into sub-cycles</th>
<th>Use of an early Prototype</th>
<th>Mechanisms for rapid on-going feedback on design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapid prototyping model (Connell &amp; Shafer, 1989)</td>
<td>No; iteration occurs prior to coding</td>
<td>Yes; early prototype up-front in process</td>
<td>N/A</td>
</tr>
<tr>
<td>Rational Unified Process (Kruchten, 2002)</td>
<td>Yes; variable length, driven by iteration sizes</td>
<td>No; each deliverable is a subsystem</td>
<td>Yes</td>
</tr>
<tr>
<td>Spiral model (Boehm, 1988)</td>
<td>No; iteration occurs prior to coding</td>
<td>Yes; prototypes used to examine risks</td>
<td>N/A</td>
</tr>
<tr>
<td>Incremental (or staged) delivery model (Wong, 1984)</td>
<td>Yes; variable length, driven by subsystems</td>
<td>No; early deliverables are working subsystems</td>
<td>N/A</td>
</tr>
<tr>
<td>Evolutionary delivery model (Gilb, 1988)</td>
<td>Yes; variable length</td>
<td>Yes; prototype at end of first subcycle</td>
<td>Customer feedback sessions</td>
</tr>
<tr>
<td>Synch-and-stabilize model (Cusumano, 1988)</td>
<td>Yes; three months</td>
<td>Yes, usually; frequent in-house (alpha) releases</td>
<td>Daily builds, regression testing, usability labs</td>
</tr>
<tr>
<td>Adaptive development model (Highsmith, 2000)</td>
<td>Yes; variable length</td>
<td>Yes; prototype at end of first subcycle</td>
<td>Peer review, joint development</td>
</tr>
<tr>
<td>Extreme programming (See <a href="http://extremeprogramming.org">http://extremeprogramming.org</a>)</td>
<td>Yes; one to three weeks</td>
<td>Yes; prototype at end of first subcycle</td>
<td>Pair programming, builds every few hours, continuous testing</td>
</tr>
<tr>
<td>SCRUM (<a href="http://extremeprogramming.org">Schwaber &amp; Beedle, 2002</a>)</td>
<td>Yes; “sprints” that last for 30 days</td>
<td>Yes; release at end of each sprint</td>
<td>Daily design meetings continuous testing</td>
</tr>
<tr>
<td>Crystal Family (Cockburn, 2001)</td>
<td>Yes; release every two-three months</td>
<td>Yes; prototype at end of first subcycle</td>
<td>Peer review, developers write test cases</td>
</tr>
</tbody>
</table>

Table 5.1
3.1 **RATIONAL UNIFIED PROCESS (RUP)**

Grady Booch, Ivar Jacobson, and Jim Rumbaugh\(^1\) got together couple of years back and developed a merged process called *Rational Unified Process* (RUP). This process is described in the process book (Jacobson, Booch & Rumbaugh, 1999). RUP contains details about what kinds of models to develop at the various stages in the process. RUP is an iterative and incremental development process, in that software is not released in one big bang at the end of the project but is, instead, developed and released in pieces. The construction phase consists of many iterations, in which each iteration builds production-quality software, tested and integrated, that satisfies a subset of the requirements of the project. The delivery may be external to early users, or purely internal (Jacobson et al., 1999). Each iteration contains all the usual lifecycle phases of analysis, design, implementation, and testing.

With regards to the project plan, according to (West, 2002), a RUP project has these two primary aspects that are important:

- RUP projects are iterative.
- Project progress is measured against clear milestones.

\(^1\)http://c2.com/cgi/wiki?ThreeAmigos
Iterative Development

The majority of RUP projects are, by definition, iterative. The RUP is an incremental process whereby the overall project is broken down into phases and iterations. The iterations are risk driven - that is, oriented toward mitigating risks - and each one should deliver executable software that’s demonstrable and testable against the project’s requirements and use cases.

The project manager uses iteration plans to manage the project. Generally, work that falls outside of an iteration plan shouldn’t be undertaken. An iteration plan:

✓ provides a detailed description of the upcoming phase of work,
✓ defines the worker roles involved, necessary activities, and artifacts to be delivered in that iteration,
✓ outlines a very clear set of measurement criteria by which progress can be assessed during the iteration and success can be measured at the end, and
✓ defines specific start/end dates and delivery dates.

Judging the size and number of iterations required for a project is described later in this paper.

Clear Milestones

The RUP identifies four phases for development projects. Each phase focuses the project team on a particular aspect of the project and has associated with it a
number of milestones. These milestones help the project manager assess project progress and ensure that the project will deliver required features and will have quality built in. The phases and what they focus on are as follows:

**Inception:** The focus of this phase is understanding the scope of the project.

**Elaboration:** The architecture as well as the requirements of the product being built must be understood by the end of this phase.

**Construction:** The software must be constructed in this phase.

**Transition:** The software must be rolled out to customers during this phase. In the context of iterative development, the milestones for a phase provide a focus for the iterations. Each iteration moves the project through certain milestones. For example, an iteration within the inception phase would be structured around the need to understand the scope of the project; the iteration(s) would provide the management framework for the team to explore the system boundary, implications of a possible solution, and the size of that solution. The number of iterations would depend on how difficult it might be to define the scope of the project. If the scope were very hard to understand or could be grouped into easily defined pieces, more than one iteration might be needed. If, as is normally the case, it would take one clear piece of work to understand the scope, one iteration would be appropriate. The milestones defined in the RUP are of necessity quite general; the project manager will need to refine the milestones so they focus the team on the needs of the project in its particular organizational context. In addition, because the aim of an iteration is to mitigate risk, during an iteration the
team will be resolving issues that apply not only to the focus of the phase but also
to other disciplines, such as architecture, testing, change management, or
construction. The manager combines the iterative, risk-oriented approach with the
refined milestones to determine the structure of the project plan.

3.2 ARCHITECTURE-CENTRIC SOFTWARE PROJECT MANAGEMENT

Figure 5.3 (Paulish, 2001) identifies some of the major steps associated with
implementing an entire project using an architecture-centric approach.

---

**Figure 5.3: Architecture-Centric Project**
The boxes represent the activities that will generally be done during the project, and the arrows represent some of the outputs that result from the activities and are inputs to other activities. Architecture-centric project management starts when an initial set of market requirements has been defined. The project manager and the architecture design team analyze the requirements. They analyze product factors and other influences as part of a global analysis. They analyze risks as part of the risk analysis. The module view of the architecture along with the feature list resulting from the requirements analysis become the basis for the release planning for the incremental development.

Once release planning is complete, a software development plan can be developed that also includes the schedule, estimated development costs, and project organization. The Software Development Plan (SDP) is generally written for the entire project, with more detailed plans and specific tasks for the first incremental release. This document will be revised for each release increment as the build plan is developed in detail for each increment. At the end of development increment, the software will be released on the date agreed to in the release plan.

### 3.3 RAPID DEVELOPMENT

To some people, rapid development consists of the application of a single pet tool or method. To the hacker, rapid development is coding for 36 hours at a stretch. To the information engineer, it’s RAP – a combination of CASE tools, intensive
user involvement, and tight timeboxes. To the manager desperate to shorten a schedule, it’s whatever practice was highlighted in the most recent issue of Business Week (McConnell, 1996). Each one of these tools and methods is fine as far as it goes, and each can contribute to increased development speed. But to provide full benefit, each must be orchestrated as part of a full-fledged strategy. No one of them applies to all cases. And no one of them can measure up to certain other practices that are not commonly thought of as rapid-development practices but that nonetheless have profound development-speed implications (McConnell, 1996). A “rapid-development project”, then, is any project that needs to emphasize development speed. McConnell (1996) presents a road-map and a strategy that is expected to allow enterprises to carry out rapid development.

We have discussed some of the software systems management strategies. The point I am trying to make is that no one strategy is best for all types of organizations. However, each organization must decide upon one strategy based on the its culture and talent pool and should try to gain expertise in that particular methodology.

4. SYSTEMS ARCHITECTURAL FRAMEWORKS & DOCUMENTATION

Most enterprises today have no formal architecture whatsoever. Their IT landscapes are spaghetti-like collections of new technologies grafted onto legacy systems
Integration has been painful and information is seldom shared efficiently, and almost never with the outside world (Business Experts, 2003). Custom development can never be replaced by packaged solutions, as discussed earlier, and therefore, this layer of USMF (Figure 5.1) will play a crucial role in the successful implementation of software systems that support the business strategy of an enterprise.

In this section, I will map numerous models / deliverables in MDA to the Zachman’s Framework. Zachman Framework forms part of this layer since it primarily deals with issues and challenges related to IT and Software Systems implementation. Such mapping will allow us to appreciate where in the range of activities carried out within an enterprise (from what to how; discussed in detail in Section 1 of Chapter 4), shown in Zachman’s Framework discussed in the previous chapter, do different models in MDA belong to. I do feel appropriate to emphasize here that Zachman’s Framework is not just meant to function simply as a way of classifying the types of documentation a company might need. Applications should be associated with specific business processes. Business processes should have goals and measures that should, in turn, be related upward to corporate goals, and downward to application and system goals. The architecture stresses both information and the relationships between information that the management team must establish and refine (Harmon, 2003).
4.1 INTRODUCTION

As we have seen in the previous section, business process design involves numerous deliverables at different levels within an organization. These deliverables might include strategic vision and long term goals, numerous processes that the business needs to perform in order to achieve those objectives and goals, and how technology can be utilized to support this strategy.

4.2 MDA & ZACHMAN

Models provide an unambiguous way of representing enterprise knowledge so that technical and non-technical people can understand and use it (Popkin, 2003). Modeling is an expression of concepts which allows each part of an organization to understand and contribute to its own evolution. Models only become useful when they cause action and provoke thought, and that only happens when all parts of an organization work together to create something rich enough for all to use (Popkin, 2003).

Although there are many types of models, those with which I am primarily concerned are graphical languages used to communicate concepts (some are part of MDA as discussed earlier). Models are the language of such frameworks as Zachman. The framework provides a means for organizing the models into useful categories and MDA provides a model driven framework to build custom
software systems. It is, therefore, entirely natural to merge these two frameworks together.

UML models are declarative models as are IDL-based object models, Java Interfaces, and Microsoft IDL interfaces. However, UML models differ from these other kinds of declarative models in some important ways (Architecture Board ORMSC, 2001).

First, UML has been defined using core UML modeling concepts and this enhances the power of MDA.

Secondly, UML models can be expressed textually as well as graphically.

Finally, UML models can be semantically much richer than models expressed in the other declarative model languages mentioned above, which can express syntax (i.e., signature) but very little about constraints on usage and behavior such as:

- Static invariants constraints on combination attributes.
- Pairs of pre and post-conditions for specifying operations.
- Whether a single-valued parameter is allowed to be null.
- Whether an operation has side effects.
- Whether subtypes of some super-type are disjoint or form a partition.
- Patterns of specifications, designs, and refinements.
Static invariants and pre/post conditions are particularly important features of an approach to rigorous software engineering called contract based design. UML did not invent the concept of contract based design, but it has very good support for it (Architecture Board ORMSC, 2001). UML defines a formal assertion language called Object Constraint Language (OCL) that facilitates specification of certain constraints. While contract based design does not eliminate the need for informal textual explanations of interface usage, it can significantly reduce dependence on them (Architecture Board ORMSC, 2001). The UML allows formalization of vocabulary otherwise left imprecise in interface specifications, as an abstract yet precise model of the state of the object providing that interface and of any parameters exchanged (Architecture Board ORMSC, 2001).

I strongly believe that UML provides a very rich language to model and analyze requirements and systems. I want to emphasize that whatever process is adopted, UML can be used within that process. UML is independent of the process adopted. And not every diagram within UML will be useful all the time. It has a wide range of diagramming notations available and the appropriate ones should be selected for each particular project. However, the following discussion and framework will briefly outline where and how different diagramming notations can be utilized. The following framework is meant to provide a guideline and does not constitute a recipe for every software project by any stretch. Indeed, I don’t believe that any single process can be utilized for every software project.
Any processes associated with a particular software development project are usually the result of multi-dimensional factors.

Figure 5.4 shows the MDA-Zachman Unified Model. The MDA separates certain key models of a system, and brings a consistent structure to these models. How the functionality specified in a Platform Independent Model (PIM) is realized is specified in a platform-specific way in the Platform Specific Model (PSM), which is derived from the PIM via some transformation. Each class of model defined below has been mapped into the Zachman’s Framework and the Figure 5.4 shows where it belongs within the unified framework.

![MDA-Zachman Unified Model](image)

**Figure 5.4: MDA-Zachman Unified Model**

<table>
<thead>
<tr>
<th>MDA-ZACHMAN MODEL</th>
<th>DATA What</th>
<th>FUNCTION How</th>
<th>NETWORK Where</th>
<th>PEOPLE Who</th>
<th>TIME When</th>
<th>MOTIVATION Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPE (Contextual) Planner</td>
<td>List of Things Important to the Business</td>
<td>List of Processes the Business Performs</td>
<td>List of Locations in Which the Business Operates</td>
<td>List of Organizations Important to the Business</td>
<td>List of Events Significant to the Business</td>
<td>List of Business Goals/Strategies</td>
</tr>
<tr>
<td>FUNCTIONING ENTERPRISE</td>
<td>Actual Business Data</td>
<td>Actual Application Code</td>
<td>Actual Physical Networks</td>
<td>Actual Business Organization</td>
<td>Actual Business Schedule</td>
<td>Actual Business Strategy</td>
</tr>
</tbody>
</table>

_BPMN = Business Process Model, ERM = Business Domain Model, PIM = Platform Independent Model, PSM = Platform Specific Model_
The PIM that is produced in the first step of an MDA development specifies functionality and behavior, and links to the Pervasive Services, Domain Facilities, and other MDA-modeled facilities that the application invokes. MDA models are UML based and depending on the objective of each model, these need to be mapped to the appropriate cell in the Zachman Model. UML class and object diagrams incorporate the structure; sequence and activity diagrams embody the behavior; class and object names, along with semantic notations, incorporate business factors; while other aspects of the model incorporate platform-independent aspects of component structure and behavior.

4.2.1 **Business Domain Models (BDM)**

In MDA, Business Domain Models (BDM) are used to understand the business domain in details. BDM describes knowledge about the business domains, independent of specific software or business processes that might be used (Architecture Board ORMSC, 2001). As such, most of BDMs belong to **Scope** row within the Zachman’s Framework. These are shown in the Figure 5.4.

4.2.2 **Business Process Models (BPM)**

Business Process Models (BPM) describe:

- business processes and include information models;
✓ detailed versions include how some software system (defined below) is used as a part of some business processes

✓ cannot violate domain knowledge that resides in BDM

Software System Specification Specifies the software independent of the various processes it may be configured for and deployed into, built using a minimal model of domain (Architecture Board ORMSC, 2001).

4.2.3 Platform Independent Models

In the MDA, the term platform is used to refer to technological and engineering details that are irrelevant to the fundamental functionality of a software component (Architecture Board ORMSC, 2001). Consider, for example, a formal definition of an operation that transfers funds from a checking account to a savings account. The fundamental functionality of this operation is that a specified amount is subtracted from a designated checking account and added to a designated savings account, with a constraint that the two accounts must belong to the same customer. This functionality remains invariant regardless of whether the operation is performed by a CORBA object, an Enterprise Java Beans, or a SOAP operation. Thus, a platform-independent model, according to (Architecture Board ORMSC, 2001), is formal specification of the structure and function of a system.
that abstracts away technical detail{superscript 20}. A Platform Independent Component View describes computational components and their interactions in a platform-independent manner. As discussed in Chapter 4, these components and interfaces, in turn, are a way of realizing some more abstract information system or application, which itself helps realize a computation-independent Business Model.

### 4.2.4 Platform Specific Models (PSM)

A Platform Specific Model (PSM) is expressed in terms of the specification model of the target platform. This is because platforms themselves also have a specification and an implementation (Architecture Board ORMSC, 2001). CORBA itself is implemented on an infrastructure, which could properly be referred to as an implementation language platform. Example of Platform-Specific Models can be a SOAP specification of the funds transfer operation (discussed in previous section under Platform Independent Models). A specification that depends on interfaces to artifacts of CORBA, like ORB, Object Services or GIOP / IIOP would be an example of a platform-specific model (Architecture Board ORMSC, 2001).

---

{superscript 20} Note that we do not say the platform-independent model. There could be models of a system from multiple viewpoints, at multiple levels of abstraction, all of which are platform independent e.g., computational and engineering viewpoints.
5. CONCLUSION

We have discussed in details the four dimensions, as shown in Figure 5.1 (Organizational & Management Infrastructure, Business Process Design, System Management Strategies, and System Architecture Frameworks, Implementation & Documentation), of, what I call, Unified Systems Management Framework. This framework is expected to highlight the numerous important factors that play crucial role in the successful implementation and management of enterprise wide software systems. This is supposed to be an initial work in this area and is a result of extensive research and interviews with experienced software development engineers, managers, and architects.
Each of the frameworks discussed in this thesis focuses on a particular aspect of software systems architecture, engineering, and management. There is certainly a disconnection between these different frameworks and a need to unify them to devise a complete solution. Accordingly, in Chapter 3, I defined and discussed software architecture in detail analyzed the importance of defining software architecture before actually beginning the construction process, just like it crucial in the fields of building architecture and civil engineering. In Chapter 4, I discussed and analyzed numerous frameworks critically, both software engineering and management related, that focus primarily on different stages and aspects of software systems development and management. Based on the analysis in Chapter 4, I proposed on a macro-level with some micro-level discussions, a unified framework, comprising of four different layers (Organizational and Management Infrastructure, Business Process Design, System Management Strategies, and Systems Architectural Frameworks, Implementations and Documentation) in Chapter
5. Each of these layers is expected to play an important role in creating the environment and providing the tools necessary for effective and efficient implementation and management of software systems.

In the following sections, I will present numerous observations that I had during the process of carrying out this research and putting together this thesis. I will also discuss the directions for future research under each section.

1. **APPLICABILITY OF FRAMEWORKS**

The thesis primarily focuses on the need for building and managing software systems in-house and how can the success probability of such projects be raised. In doing so, I interviewed couple of executives in numerous financial services institutions. The ideal type of institution that I found would benefit the most from such framework(s) is Wells Fargo. In order to successfully adopt such frameworks, there is a need for a technically savvy in-house staff. As such, Robert Holvey (Senior Vice President) at Wells Fargo said:

“We’re pretty big and we do some amount of all of these. Outsourcing, when we do it, tends to be extremely focused and confined to specific steps or processes. Consulting is reasonable at a high level, but usually has limited value for actual implementation because the consultants: very
seldom understand our business as well as the internal resources; very seldom understand the complexities of working with the rest of our systems and the data in those systems; and very seldom have many folks with real application development and programming skills. Often time Consultants get an assignment, then run around the Company to find folks who have the knowledge and skills to do the work that they need to get done, then repackage the results for the final presentation. As a result, the tendency is to use In-house resources because internal folks are already on the payroll, understand the business, our issues, and our problems pretty well, and also have a lot of experience with what it takes to effectively implement within our Company’s culture.”

In addition, as mentioned in the Unified Systems Management Framework, organizational infrastructure will certainly play an important role. I found that innovative organizations have found ways of keeping the pace with technological advances and it will vary from organization to organization. For instance, Robert Holvey described the process through which Wells Fargo utilizes the standards and guidelines put together by external organizations. He said:

“As some of the folks in the IT organization try to keep pace with ongoing standards development and guidelines from external organizations. Where they are applicable, we’ll adopt. Generally we follow an extensive"
set of internally developed processes and guidelines that, where useful, overlap with externally developed processes and guidelines.”

For future research, it will be interesting to analyze what impact does organizational structure have on building a creative environment and if it varies from industry to industry.

2. KNOWLEDGE REUSE & REFINEMENT

According to Robert J. McKain (Gartner, 2003), the reason most major goals are not achieved is that we spend our time doing second things first. Every CIO in the world has one supreme challenge: Make technology serve the business better while keeping pace with business change (Gartner, 2003). Business has to be put first if information systems were to support the business and long-term strategy that they intend to. Accordingly, the project is started with the core business process, like mass customization or sharing inventing with partners. But instead of buying a glossy new technology purported to do the job on its own, only those technologies are chosen that fit the environment of interoperability that is being created – solutions that improve flow, enable mass customization, or make the sharing of inventory easier, faster, and more partner-friendly (Gartner, 2003).
Each of the frameworks discussed in Chapter 4 provides a definite opportunity for knowledge reuse and refinement. With regards to business knowledge reuse and refinement, in case of MDA, this is done during Platform Independent Models (PIM) development stage. However, it will be interesting to analyze if there are existing standards and specifications for specific industries in general and financial services industry in particular that can be transformed into PIMs and later refined as greater understanding of the business is acquired or when it comes to differentiating business needs. For instance, in case of financial services industry, which is the main discussion in this thesis, there are certain specifications available, for example Open Financial Exchange (OFX). OFX is an effort to specify standard structures for exchanging financial information. OFX is a broad-based framework for exchanging financial data and instructions between customers and their financial institutions (OFX, 2001 p. 15). Such specification allows institutions to connect directly to their customers without requiring an intermediary. An interesting topic for research is the use of already written specifications like OFX in building Platform Independent Models (PIM) for financial services industry.

**Figure 6.1: Request & Response Messages**
OFX is an open specification that anyone can implement: any financial institution, transaction processor, software developer, or other party (OFX, 2001, p. 15). It uses widely accepted open standards for data formatting (such as XML), connectivity (such as TCP/IP and HTTP), and security (such as SSL). OFX defines the request and response messages used by each financial service as well as the common framework and infrastructure to support the communication of those messages.

Such efforts were aimed at defining specifications for standard ways of implementing information systems for financial services. It will be a useful research to find out how does OFX relate, if at all, to MDA and does it provide a reasonable platform to design and later refine the PIMs for financial services.

3. **STANDARDIZATION**

Major source / foundation / network used for exchanging information across the globe is the Web. In case of Capital Markets / Financial Services (CMFS) firms, research shows that without data integration, there are increased processing costs and ambiguity of meaning as information between subunits of the organization is processed. This results in “delays, decreases in communication, reduction in the amount of summarization, and greater distortion of meaning” (Goodhue, Kirsh, Quillard, & Wybo, 1992a, p. 300). Research (Badar, Hayward, Razo, Madnick & Siegel, 1999) shows that Senior Management in (but not limited to) CMFS firms continually makes decisions that require
aggregation of lower level information. The fact that different business units maintain
information in different data structures makes the flow and aggregation of information a
challenging and expensive task (Badar et al., 1999). Research (Goodhue et al., 1992a,
p.300) also shows that data standardization improves managerial information and
coordination across subunits of an organization. According to (Goodhue, Wybo, & Kirsh,
1992b p. 302), data standardization implies “some degree of logical centralization and
some central authority with control over the logical aspects of data.” Research (Goodhue
et al., 1992a) shows that data integration is more effective when needs of units are
homogeneous across organization. This is because subunits can avoid incurring
additional cost in compromising their heterogeneous needs, and reducing bureaucratic
delays, and more importantly, meeting both local and global needs beyond the realm of
business units.

It seems that MDA can provide a decent platform to meet a crucial and a
challenging need of achieving standardization. However, a critical analysis of
frameworks, such as MDA, and how they specifically assist in achieving standardization
is required.
REFERENCES


Services”, MIT Sloan School of Management.


GLOSSARY

BPMS (Business Process Management System)

Many business managers have a vision of enterprise systems in which business processes are fully automated--governed by business rules embedded in software programs--and human intervention is required only for the occasional exception to those business rules. But when we look at the current state of IT, with its wide range of disparate applications and huge amounts of money being poured into application integration, we soon realize that there is a wide gap between vision and reality (ebiz, 2003).

How can we bridge that gap? Among the items at the top of the priority list is a higher-level technical infrastructure that allows businesses to implement their real-world business processes and integrate applications based on standards. A business process management system (BPMS) delivers that kind of higher-level technical infrastructure (ebiz, 2003).

From a technical point of view, a BPMS allows you to seamlessly build on your existing componentization strategy. An application server framework is the ideal technical foundation for a BPMS. While application servers provided a first-generation integration platform and encouraged companies to componentize, BPMSs furnish the complementary higher-level integration platform. They enable a company to effectively integrate its business processes, something most application servers cannot offer (ebiz, 2003).
BPMS applications are complex product suites, allowing for the definition and mapping of processes, the development and deployment of processes, process execution (brokering), administration, monitoring and reporting. According to ebiz (2003), the main components of a BPMS are:

- A process modeling tool, which is used to capture, design and modify business process definitions, including the operational and interface properties of the activity implementations (i.e., applications) they interact with.

- Development tools, which define process control logic and provide the necessary "glue" for application integration. In the context of a BPMS, applications are better thought of as business services. Other tools, such as a forms editor and a business rule editor, complement the tool box.

- The process engine, which orchestrates the execution of business processes and, as such, is the core BPMS component. It also coordinates conversations among process engines based on public processes, which forms the backbone of global business collaboration.

- A sophisticated repository, which stores a wealth of data objects, most notably business process definitions, business rules, integrity constraints, security and policy definitions, and business metric definitions.

- The rule engine, which evaluates the state of objects and executes specified actions if conditions hold. At run-time, the rule engine evaluates the rules. As business rules are executed, they may change the course of process execution.

- Process management tools, which allow for the monitoring of business processes to correlate business service events to fundamental business metrics.
Subsequently, they also allow for the optimization of these business processes. By monitoring detailed business process statistics, business analysts can analyze the performance of business processes and derive approaches for optimization if necessary.

- Administration tools, which provide a means to intervene at the system, process or activity level. Users can perform actions, such as starting, stopping, suspending and resuming processes or activities.

- Reporting and analysis tools, which help present statistics and other data collected during process execution.

**COM+**

COM+ is an extension of Component Object Model (COM), Microsoft's strategic building block approach for developing application programs. COM+ is both an object-oriented programming architecture and a set of operating system services. It adds to COM a new set of system services for application components while they are running, such as notifying them of significant events or ensuring they are authorized to run. COM+ is intended to provide a model that makes it relatively easy to create business applications that work well with the Microsoft Microsoft Transaction Server (MTS) in a Windows NT or subsequent system. It is viewed as Microsoft's answer to the Sun Microsystems-IBM-Oracle approach known as Enterprise JavaBeans (EJB) (TechTarget, 2003).
Among the services provided by COM+ are:

- An event registry that allows components to publish the possibility of an event and other components to subscribe to be notified when the event takes place. For example, when a sales transaction is completed, it could trigger an event that would allow other programs to be notified for subsequent processing.
- The interception of designated system requests for the purpose of ensuring security
- The queues of asynchronously received requests for a service

**CORBA**

Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group (OMG), which currently includes over 500 member companies. Both International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components). CORBA 3 is the latest level (TechTarget, 2003).
The essential concept in CORBA is the Object Request Broker (ORB). ORB support in a network of clients and servers on different computers means that a client program (which may itself be an object) can request services from a server program or object without having to understand where the server is in a distributed network or what the interface to the server program looks like. To make requests or return replies between the ORBs, programs use the General Inter-ORB Protocol (GIOP) and, for the Internet, its Internet Inter-ORB Protocol (IIOP). IIOP maps GIOP requests and replies to the Internet's Transmission Control Protocol (TCP) layer in each computer (TechTarget, 2003).

**CORBA Component Model (CCM)**

The specification for the CORBA Component Model (CCM) is written to address these and other complexities in the CORBA object model. The CCM is part of the CORBA 3.0 specification, which is due to be released this year. The CCM is a server-side component model for building and deploying CORBA applications. It is very similar to Enterprise Java Beans (EJB) because it uses accepted design patterns and facilitates their usage, enabling a large amounts of code to be generated. This also allows system services to be implemented by the container provider rather than the application developer. The benefit and need for these types of containers can be observed through the growth of Application Server software. The CCM extends the CORBA object model by defining features and

---

services in a standard environment that enable application developers to implement, manage, configure and deploy components that integrate with commonly used CORBA Services. These server-side services include transactions, security, persistence, and events.

**CRM (Customer Relationship Management)**

Simply stated, Customer Relationship Management (CRM) is about finding, getting, and retaining customers (RealMarket, 2003).

CRM is at the core of any customer-focused business strategy and includes the people, processes, and technology questions associated with marketing, sales, and service. In today's hyper-competitive world, organizations looking to implement successful CRM strategies need to focus on a common view of the customer using integrated information systems and contact center implementations that allow the customer to communicate via any desired communication channel. Lastly, CRM is a core element in any customer-centric eBusiness strategy (RealMarket, 2003).
**EJB SERVER**

Software that provides services to an EJB container. Typically, an EJB container relies on a transaction manager that is part of the EJB server to perform a two-phase commit across all participating resource managers. The J2EE architecture assumes that an EJB container is hosted by an EJB server from the same vendor; therefore, it does not specify the contract between these two entities. An EJB server may host one or more EJB containers (About Java, 2003b).

**ERP (Enterprise Resource Planning)**

An integrated information system that serves all departments within an enterprise. Evolving out of the manufacturing industry, ERP implies the use of packaged software rather than proprietary software written by or for one customer. ERP modules may be able to interface with an organization's own software with varying degrees of effort, and, depending on the software, ERP modules may be alterable via the vendor's proprietary tools as well as proprietary or standard programming languages.

An ERP system can include software for manufacturing, order entry, accounts receivable and payable, general ledger, purchasing, warehousing, transportation and human resources. The major ERP vendors are SAP, Kewill, PeopleSoft, Oracle, Baan and J.D.
Edwards. Lawson Software specializes in back-end processing that integrates with another vendor's manufacturing system.

**Java IDL**

Java™ IDL adds CORBA (Common Object Request Broker Architecture) capability to the Java platform, providing standards-based interoperability and connectivity. Java IDL enables distributed Web-enabled Java applications to transparently invoke operations on remote network services using the industry standard OMG IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB Protocol) defined by the Object Management Group. Runtime components include an Object Request Broker (ORB) for distributed computing using IIOP communication.

**JAVA MESSAGE SERVICE (JMS)**

Enterprise messaging provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an enterprise. The Java Message Service (JMS) API adds to this a common API and provider framework that enables the development of portable, message based applications in the Java language (About Java, 2003a).

---

MIDDLEWARE

Middleware are layer(s) of software between client and server processes that deliver the extra functionality. Middleware hides the complexity of extra functionality behind a common set of APIs (Application Programming Interfaces) that client and server processes can invoke (Burden, 2003).

Middleware platforms, such as CORBA, DCOM, J2EE, and .NET, have provided abstraction and simplicity for the complex and heterogeneous computing environment. They facilitate the development of high quality distributed applications with shorter development cycle and much smaller coding effort. Middleware systems are being adopted in a very broad spectrum of application domains, ranging from traditional enterprise platforms to mobile devices, embedded systems, real time systems, and mission critical systems (Zhang & Jacobsen, 2003).

MODEL DRIVEN ARCHITECTURE (MDA) RELATED TERMS

- **eXtensible Markup Language Metadata Interchange (XMI):** XMI is a standard interchange mechanism used between various tools, repositories and middleware. XMI can also be used to automatically produce XML DTDs (and soon XML Schemas) from UML and MOF models, providing an XML

---

serialization mechanism for these artifacts. XMI has been used to render UML artifacts (using the UML XMI DTD), data warehouse and database artifacts (using the CWM XMI DTD), CORBA interface definitions (using the IDL DTD), and Java interfaces and Classes (using a Java DTD). XMI, which marries the world of modeling (UML), metadata (MOF and XML) and middleware (UML profiles for Java, EJB, IDL, EDOC etc.) plays a pivotal role in the OMG’s use of XML at the core of the MDA. In essence XMI adds *Modeling* and *Architecture* to the world of XML.

✓ **Meta Object Facility**: MOF provides the standard modeling and interchange constructs that are used in MDA. Other standard OMG models, including UML and CWM, are defined in terms of MOF constructs. This common foundation provides the basis for model/metadata interchange and interoperability, and is the mechanism through which models are analyzed in XMI. MOF also defines programmatic interfaces for manipulating models and their instances spanning the application lifecycle. These are defined in IDL (Interface Definition Language) and are being extended to Java.

✓ **Common Warehouse Metamodel**: CWM is the OMG data warehouse standard. It covers the full life cycle of designing, building and managing data warehouse applications and supports management of the life cycle. It is probably the best example to date of applying the MDA paradigm to an application area. Historically, the integration between the development tools and the deployment
into the middleware framework, has been weak. This is now beginning to change by using key elements of the MDA – specific models and XML DTDs that span the life cycle, and profiles that provide mappings between the models used in various life cycle phases.

**MTS**

The Microsoft Transaction Server (MTS) is a program that runs on an Internet or other network server with a Windows NT system and manages application and database transaction requests on behalf of a client computer user. The Transaction Server screens the user and client computer from having to formulate requests for unfamiliar databases and, if necessary, forwards the requests to database servers. It also manages security, connection to other servers, and transaction integrity.

The Transaction Server is Microsoft's bid to make distributed applications and data in a network relatively easy to create. It's one of a category of programs sometimes known as middleware or *multi-tier* programming for the enterprise market that IBM has traditionally controlled with its CICS and similar transaction management products.

Microsoft designed the Transaction Server to fit in with its overall object-oriented programming strategy. Using the Transaction Server, you can use a drag-and-drop
interface to create a transaction model for a single user, then allow the Transaction Server to manage the model for multiple users, including the creation and management of user and task threads and processes.

**SOAP (Simple Object Access Protocol)**

SOAP is a protocol specification for invoking methods on servers, services, components and objects. SOAP codifies the existing practice of using XML and HTTP as a method invocation mechanism. The SOAP specification mandates a small number of HTTP headers that facilitate firewall/proxy filtering. The SOAP specification also mandates an XML vocabulary that is used for representing method parameters, return values, and exceptions." (CoverPages, 2003)

"SOAP is the Simple Object Access Protocol, a way to create widely distributed, complex computing environments that run over the Internet using existing Internet infrastructure. SOAP is about applications communicating directly with each other over the Internet in a very rich way." (CoverPages, 2003)
Universal Discovery Description and Integration (UDDI) is a specification for distributed Web-based information registries of Web Services. UDDI is also a publicly accessible set of implementations of the specification that allow businesses to register information about the Web Services they offer so that other businesses can find them.

Web Services are the next step in the evolution of the World Wide Web (WWW) and allow programmable elements to be placed on Web sites where other can access distributed behaviors. UDDI registries are used to promote and discover these distributed Web Services.

As communications protocols and message formats are standardized in the web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for

---

25 UDDI: http://uddi.org

26 http://www.w3.org/TR/wsdl#_introduction
distributed systems and serve as a recipe for automating the details involved in applications communication.

A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message** – an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type** – an abstract set of operations supported by one or more endpoints.
- **Binding** – a concrete protocol and data format specification for a particular port type.
- **Port** – a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.
It is important to observe that WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification (XSD) as its canonical type system. However, since it is unreasonable to expect a single type system grammar to be used to describe all message formats present and future, WSDL allows using other type definition languages via extensibility.

In addition, WSDL defines a common binding mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

**WEB SERVICES**

Gartner and GartnerG2 define Web Services as software components that employ one or more of the following technologies—Simple Object Access Protocol (SOAP), Web Services Distribution Language (WSDL) or universal description, discovery integration (UDDI)—to perform distributed computing. Using these technologies constitutes a Web Service; using all of them is not required (Gartner Research, 2002).

Web Services use technology and Internet standards to expose business processes internally and to customers, suppliers, distributors and other trading partners. While Web
Services might be used to assemble information and functions to be delivered through a Web front-end, a true Web service sits between systems to enable information and processes to be shared. For example, XML (extensible markup language) used to create an interface between two systems does not necessarily establish the interface as a service available to other systems or processes. It becomes a Web service when it is constructed in such a way that it can be used or invoked by any other system with appropriate access privileges. In fact, we can say that the convergence of two major trends is creating a rapidly growing demand for a new breed of software that facilitates automation of business processes both between enterprises and within the enterprise (Kapil, 2002, p. 157, 158).

The first of these trends is Web Services technology: a collection of XML-based standards that provide a means for passing information between applications using XML documents. The second of these trends is a business driver (Kapil, 2002, p. 157, 158). In order to increase organization’s agility in responding to customer, market, and strategic requirements, the information flow between the IT systems that carry out these business operations must be streamlined. This includes not only the organization’s own IT systems but also those of its partners. It is the task of electronic business integration to automate this information flow as much as possible in order to streamline operations. Historically, organizations have generally focused on integrating own IT systems. If, however, the information flow between the organization’s own IT systems and those of its partners (particularly in the supply chain) is not also streamlined, then the overall agility of the business is still restricted (Kapil, 2002, p. 157, 158). Therefore, many enterprises also
strive to integrate their partner’s IT systems with their own in order to more fully automate critical business processes such as sales, procurement, and research and development. The benefits of the increased agility resulting from business process automation are extensive. For example, operational costs are decreased, inventories are reduced, customer satisfaction is increased, and products are brought to market faster (Kapil, 2002, p. 157, 158).

**XML (eXtensible Markup Language)**

XML is the Extensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification (Flynn, 2003).

It is called extensible because it is not a fixed format like HTML (a single, predefined markup language). Instead, XML is actually a `metalanguage' —a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML, the international standard metalanguage for text markup systems (ISO 8879) (Flynn, 2003).
XML is intended ‘to make it easy and straightforward to use SGML on the Web: easy to define document types, easy to author and manage SGML-defined documents, and easy to transmit and share them across the Web (Flynn, 2003).

It defines ‘an extremely simple dialect of SGML which is completely described in the XML Specification. The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML (Flynn, 2003).

For this reason, XML has been designed for ease of implementation, and for interoperability with both SGML and HTML (Flynn, 2003).