

A Domain Neutral Enterprise Architecture Framework for Enterprise Application Integration and Pervasive Platform Services

Master of Science Thesis

Royal Institute of Technology

Donald Baldwin

Thesis Supervisor:

Erik Perjons, PhD

Department of Computer and Systems Sciences
Royal Institute of Technology (KTH) / Stockholm University
Kista, Sweden



© 2015 Donald Baldwin. This work is licensed under a CC BY-SA 4.0 license.



This document is available on ResearchGate

ORCID: 0000-0003-3712-7454 Donald Baldwin

v40 EV – June 2015

ESAL: Updated Archive 2017 NOV

Abstract

In this thesis, the Domain Neutral Enterprise Architecture Framework (DNEAF) for Enterprise Application Integration Services (EAIS) and Platform Pervasive Services (PPS) is presented. It addresses a problem that many organizations experience that introduce an Enterprise Architecture to support the integration of different services across the enterprise. The problem is that, organizations do not know where they either have or may encounter weaknesses in their Enterprise Architecture. The framework presented is based on concepts from OMG's Model Driven Architecture and is designed to support both Transitional Gap Analysis (TGA) and Comparative Gap Analysis (CGA). TGA is supported by comparing a baseline PIM to a target PIM where both PIMs have been defined from the CIM. CGA is facilitated by mapping a PIM to two or more PSM's. The research methodology used in the thesis is design science research. The DNEAF for EAIS and PPS has been applied in many different real-world organizations. To demonstrate the framework, three applications of the framework are presented in the form of case studies. The case studies cover eHealth, Banking and Comparing Vendor Platforms. The eHealth case study demonstrates how the framework can be used to design a platform solution. The Banking case study demonstrates how the framework can be used in Transitional Gap Analysis. The Comparing Vendor Platforms case study demonstrates how the framework can be used in Comparative Gap Analysis. All three case studies are based on real-world examples. The thesis also presents a number of requirements on the framework and discusses their fulfillment.

Table of Contents

1 BACKGROUND 7

1.1 The Domain Neutral Enterprise Architecture Framework 7
1.2 Glossary 10
1.3 Problem 10
1.4 Goal 10
1.5 Requirement on the Framework 11
1.6 An overview of the Framework 11
1.6.1 Platform Service Categories of Essential Services 11
1.6.2 High-Level DNEAF Components 12
1.6.3 Security Zones 12
1.6.4 Enterprise and Solution Development Tools and Services 12
1.7 Case Studies 13

2 RESEARCH METHODOLOGY 14

2.1 Design Science Research Methodology 14
2.2 Results of the application of DSRM 17

3 RELATED RESEARCH 18

4 FRAMEWORK CONCEPTS 21

4.1 Managing Risk and Cost 21
4.2 Software Maintainability 23
4.3 Domain Neutrality 24
4.4 Patterns 25
4.5 Model Driven Architecture 26

5 DNEAF FOR EAIS AND PPS FRAMEWORK SPECIFICATION 29

5.1 Directory Services 33
5.2 Repository Services 33
5.2.1 Business Process Repository 34
5.2.2 XML Repository 34
5.3 Application Integration Services 34
5.3.1 Basic Messaging Services 36
5.3.1.1 Message Dictionary 36
5.3.2 Data Services 36
5.3.2.1 Data Dictionary 37
5.3.3 Transformation Services 37
5.3.3.1 Transformation and Routing Engine 38
5.3.3.2 XML Engine 39
5.3.4 Message Warehouse 39
5.3.5 Information Pipe (Enterprise Messaging Services and Message Queuing Services) 40
5.3.5.1 Publish and Subscribe Services 41
5.3.5.2 Connector SDK 41
5.4 Business Integration Services 42
5.4.1 Basic Integration Services 42
5.4.2 Workflow Management Services 42
5.4.3 Security Services 42
5.4.3.1 Security Interceptor 42
5.4.4 Partner Profile Services 42
5.4.4.1 ERP Services 43
5.4.4.2 EDI Services 43
5.4.5 Content Management Services 44
5.4.5.1 ECM Metadata Repository 46
5.4.6 Decision Support Services 46
5.4.7 Application Services 46
5.5 Integration Services Proxy 47
5.5.1 XML Services 47
5.5.2 B2B Services 47
5.6 Authentication Services 48
5.6.1 RBAC Repository 48

5.6.2 Security Integration Component ... *48*
5.6.3 Single Sign-On Component (SSO) ... *48*
5.7 Gateway Services ... 48
5.8 Access Point ... 49
5.8.1 Access Point Connector ... *49*
5.9 Management Services ... 50
5.9.1 Transaction Performance Manager ... *50*
5.9.2 Transaction Processor ... *50*
5.9.3 Transaction Process Monitor ... *50*
5.9.4 Logging Services ... *50*
5.10 Enterprise Development Environment ... 51
5.10.1 Developer IDE ... *51*
5.10.1.1 CASE Tools ... 51
5.10.2 Version Control Mechanism ... *51*
5.10.3 Deployment Mechanism ... *51*

6 CASE STUDIES ... 52

6.1 Case Study 1: E-Health Platform for Telemedicine ... 52
6.1.1 Executive Summary ... *52*
6.1.2 Platform ... *53*
6.1.2.1 Message-Oriented Middleware (MOM) ... 54
6.1.2.2 Loosely Coupled Systems ... 54
6.1.2.3 Workflow Transactions ... 54
6.1.2.4 Subsystems ... 54
6.1.2.5 Message Queuing Services ... 54
6.1.2.6 Message Broker ... 55
6.1.2.7 Data and Metadata Subsystem ... 56
6.1.2.8 Application Runtime Support ... 57
6.1.2.9 Portal Services ... 57
6.1.2.10 Implementation Approach Options ... 58
6.1.2.11 Platform Considerations ... 59
6.2 Case Study 2: Banking - Transitional Gap Analysis ... 59
6.3 Case Study 3: Vendor Product Evaluation - Comparative Gap Analysis ... 64
6.3.1 IBM WebSphere 5 PSM ... *67*
6.3.2 Oracle 10g PSM ... *72*
6.3.3 Microsoft .Net PSM ... *76*
6.3.4 Open Source PSM ... *82*
6.3.5 Fujitsu Interstage 7 PSM ... *84*

7 CONCLUSION ... 85

7.1 Discussion ... 85
7.2 Requirement fulfillment ... 87
7.3 Future Research ... 88

REFERENCES ... 89

Table of Figures

Figure 1: The Design Science Research Cycle based on [50] 15
Figure 2: Competition and Technology Change diagram, by [52] 18
Figure 3: The impact of change on cost during a waterfall lifecycle or iteration of a lifecycle [13] 22
Figure 4: Agile Cost of Change Curve [13] 23
Figure 5: Model Driven Architecture (MDA) 27
Figure 6: The PIM models serve as a mapping between the business and the IT implementation 27
Figure 7: The PIM as the roll of a fulcrum between the CIM and the PSM 27
Figure 8: A high-level overview of the Enterprise Portal Integration Platform in the form of a Model Driven Architecture Platform Independent Model [41] 32
Figure 9: The two viewpoints in Application Integration Services 35
Figure 10: The four most important repository artifacts 35
Figure 11: Illustration of how the repository framework can be shared by multiple applications and databases to promote information sharing across the enterprise platform 37
Figure 12: The basic functionality that Content Management Services must support 44
Figure 13: Six Enterprise Content Patterns that categorize all information managed within enterprise platforms and enterprise portal platforms [2] 45
Figure 14: A conceptual model of the Access Point Connector 49
Figure 15: DNEAF for EAIS and PPS mappings for eHealth 53
Figure 16: Domain Scoping Example 59
Figure 17: Transition to UML High-Level Model 60
Figure 18: Robustness Analysis Model (A special type of network communication model in UML) 61
Figure 19: Business Event Walkthrough Scenario Mapping 61
Figure 20: Major Areas of Change in the Architecture 62
Figure 21: Target (to be) Baseline 63
Figure 22: Comparing Current Baseline to Target Baseline 63
Figure 23: Proxy Component Example of PIM to PSM Mapping 64
Figure 24: Conceptual example of CIM to PIM to multiple PSM mapping 65
Figure 25: PENG example for cost comparison 65
Figure 26: Comparative Gap Analysis of COTS without the DNEAF for EAIS and PPS 66
Figure 27: Comparative Gap Analysis of COTS using the DNEAF for EAIS and PPS 67
Figure 28: Mapping WebSphere to the DNEAF for EAIS and PPS 67
Figure 29: Mapping Oracle to the DNEAF for EAIS and PPS 72
Figure 30: Mapping Microsoft to the DNEAF for EAIS and PPS 76

Table of Tables

Table 1: Timeline summarizing the development, application and dissemination of the DNEAF for EAIS and PPS 9
Table 2: Results of the application of the DSRM's activities 17
Table 3: How easy it is to use the DNEAF for EAIS and PPS 19
Table 4: Relative Cost of Correcting a Software Error [13, 46] 22
Table 5: Message Broker Components 55
Table 6: Databases and Object Stores 56
Table 7: Data and Metadata Subsystem Components 56
Table 8: Runtime Support Subsystem Components 57
Table 9: Portal Services Subsystem Components 57
Table 10: COTS Mappings in eHealth Architecture 58
Table 11: OpenSource Alternative to COTS 58
Table 12: Mapping Worksheet for WebSphere 69
Table 13: Comparative Product Support by Computational Device Category 70
Table 14: Mapping Worksheet for Oracle 73
Table 15: Mapping Worksheet for Microsoft 77
Table 16: Mapping Worksheet for Open Source 82
Table 17: Mapping Worksheet for Fujitsu Interstage 84

1 Background

1.1 The Domain Neutral Enterprise Architecture Framework

This thesis presents the Domain Neutral Enterprise Architecture Framework (DNEAF) for Enterprise Application Integration Services (EAIS) and Platform Pervasive Services (PPS). It is based on enterprise systems and web services technologies. It was developed by the Enterprise Systems Architecture Laboratory (ESAL), operated by Aurenav, LLC, and I have played a leading role in the development, application and dissemination of the DNEAF for EAIS and PPS including authoring the framework specification that is presented in Section 5.

Domains in which the DNEAF has been successfully applied include:

- B2B Domain - Global telecoms equipment and product provisioning platform and web portal at Ericsson in Stockholm, Sweden
- Healthcare Domain - Telemedicine platform at the Karolinska University Hospital in Huddinge, Sweden
- Telecommunications Domain - Global 3G telecommunication service provisioning platform (Orange)
- Banking Domain - Payments processing and clearing house platform for Post Office Bank in London, UK

Additional proof-of-concept work includes:

- Air Traffic Control Domain - Architectural framework approach for metadata designed for Dynamic Metadata Support in Air Traffic Environments presented before the National Airspace System Architecture Committee in the US on January 24th, 2002.

Important presentation of DNEAF:

- Christopher de Vaney and Donald Baldwin first presented the ESAL Integration Services Platform based on the DNEAF PPS at the Object Management Group (OMG) Technical Meeting in July 2001.
- Christopher de Vaney and Donald Baldwin presented an example of the Integration Services platform based on the DNEAF PPS as implemented by Ericsson at the following OMG Technical Meeting in September 2001.
- Donald Baldwin presented a Transitional and Comparative Gap Analysis framework based on the DNEAF PPS and compliant with the OMG Model Driven Architecture (MDA) at the OMG Technical Meeting in November 2001.

Background of the development of DNEAF:

- The DNEAF for EAIS and PPS is provided as a Platform Independent Model (PIM), which is a core concept of MDA. The original concept for the PIM model was developed by Chris de Vaney in 2001 who was the Chief Architect for the ESAL at the time.

- In June 2004, Donald Baldwin, Amanda Tratar and Marcial Gallardo Torres published a report as part of Tratar and Torres Master's studies at the IT University, Stockholm University and Royal Institute of Technology, titled "*A Tool For Evaluating Enterprise Portal Platform Technologies Based On Message Oriented Middleware*". The report presented a Comparative Gap Analysis of IBM WebSphere 5.x, Oracle 10g, Microsoft .Net and Open Source platforms based on DNEAF for EAIS and PPS, for the Telemedicine Platform at Karolinska University Hospital.

The current status of DNEAF:

The DNEAF for EAIS and PPS reference model has undergone extensive peer review by both solution and enterprise architects resulting in a number of important changes from the original thesis outline. The current version of the thesis also benefits from the experience in applying the concepts in a number of real world applications including both large scale enterprises and small to medium enterprise. The largest enterprise application to date spanned 165 countries with multiple integrated platforms and several thousand services and applications including extensive third-party integrations. The smallest application of the framework included an assessment of the enterprise architecture at the European Center for Disease Prevention and Control (ECDC), an example of an SME public sector enterprise. Working with smaller organizations revealed a number of weaknesses in the original framework that have been addressed in the current version. Notably, new content describing important concepts and sources that influenced the thinking process leading to the development of the reference model have been included. Additionally, more of the foundation terms have been defined and case studies added to help demonstrate how the reference model can be used.

Important input and review for this study have been provided by:

- Martin Hedberg and Isaac Allotey-Pappoe IT architects with IBM Sweden
- Hans Nystedt and Hakan Gustavsson IT architects with Oracle Sweden
- Tommy Sundling and Wolfram Meyer IT architects with Microsoft Sweden
- Ulf Hedlund and Ralf Nyren ICT architects with ESAL at Aurenav (Open Source)

The following timeline summarize the development, application and dissemination of the DNEAF for EAIS and PPS, see Table 1:

Table 1: Timeline summarizing the development, application and dissemination of the DNEAF for EAIS and PPS

2000	A formal generic conceptual model for a Portal Integration Services Platform was developed for the Ericsson global web portal and B2B services by Chris de Vaney and made part of the enterprise architecture documentation. This model was designed to support the complex platform and application service environment at Ericsson.
2001	A Platform Independent Model (PIM) archetype framework was developed by Chris de Vaney and Donald Baldwin, which transformed the Portal Integration Services Platform into a Model Driven Architecture (MDA) compliant view. The resulting reference framework was called the Domain Neutral Enterprise Architecture Framework (DNEAF). Donald Baldwin created a formal specification of the DNEAF based on the previous work and in collaboration with Chris de Vaney.
2001	Over the course of three technical meetings, the DNEAF was presented at the Object Management Group (OMG). Donald Baldwin introduced the Transitional and Comparative Gap Analysis methodology based on the DNEAF.
2001-2004	The DNEAF was used successfully in a number of large enterprise architecture programs in both the private and public sectors including IBM, Cap Gemini and Orange.
2003-2005	Donald Baldwin and Chris de Vaney worked with the Karolinska University Hospital to develop an e-Health Portal and Integration Services Platform, which was published as a short white paper and presented to the IT department. At the same time, Donald Baldwin worked with the Biomedical Engineering Department in a multiyear applied research project that evaluated and refined both the DNEAF and the Transitional and Comparative Gap Analysis (TCGA) process based on the DNEAF. This resulted in several changes to the terminology used (e.g. greater focus on services as opposed to only applications, change from horizontal to transitional gap analysis and change from vertical to comparative gap analysis).
2005	DNEAF was implemented successfully at Fujitsu Services key account Post Office bank in the UK by Donald Baldwin.
2006	DNEAF was implemented successfully at Cerdo Bank Partner in Sweden, Denmark and Norway as a reference model for evaluating different enterprise architecture alternatives by Donald Baldwin.
2006-2007	DNEAF and TCGA was successfully used on several projects in Sweden, Denmark and Finland by Donald Baldwin.
2007-2009	DNEAF and TCGA successfully used at DnB Nord Bank across an international banking platform covering six countries by Donald Baldwin.
2010	Donald Baldwin extended the DNEAF to specifically include EAI Services and Platform Pervasive Services, which form the basis of this thesis.

1.2 Glossary

This part presents the major terms used as well as their abbreviations:

BMD	Biomedical Device
CIM	Computationally Independent Model (from MDA)
DBE	Department of Biomedical Engineering (MTA in Swedish)
DNEAF	Domain Neutral Enterprise Architecture Framework
DSRM	Design Science Research Methodology
EAF	Enterprise Architecture Framework (EAFs for plural form)
EAI	Enterprise Application Integration
EAIS	Enterprise Application Integration Services (from DNEAF)
ERP	Enterprise Resource Planning
ES	Essential Services (from MDA)
ESAL	Enterprise Systems Architecture Laboratory
KUH	Karolinska University Hospital
MDA	Model Driven Architecture (from OMG)
MTA	Swedish acronym for DBE
OECD	Organization for Economic Co-operation and Development
OMG	Object Management Group
PIM	Platform Independent Model (from MDA)
PPS	Platform Pervasive Services (from MDA)
PSM	Platform Specific Model (from MDA)
RAF	Reference Archetype Framework
SME	Small to Medium Sized Enterprise (from OECD)
TCGA	Transitional and Comparative Gap Analysis

1.3 Problem

Many organizations introduce an Enterprise Architecture to support the integration of different services across the enterprise. However, they do not know where they either have or may encounter weaknesses in their Enterprise Architecture. This problem is addressed by the DNEAF for EAIS and PPS.

1.4 Goal

The goal of the thesis is to present a framework in the form of a reference model, called DNEAF, that visualizes the gaps (weaknesses) in proposed or existing enterprise architectures and to support a comparative analysis process for different alternative solution approaches.

1.5 Requirement on the Framework

Based on the problem and the goal, a set of requirements on the DNEAF can be presented:

- The DNEAF should be generic. That is, the reference model should be domain neutral and suitable for use in any type of enterprise architecture where pervasive services and integration services are important platform components.
- The framework should be usable as a reference model to describe current and proposed states of an enterprise architecture
- The framework should be usable to compare different vendor platforms
- The framework should support understanding of how systems and services are tied together
- The framework should be easy to apply

Moreover, from the OMG perspective, the DNEAF for EAIS and PPS can be considered a platform technology since it spans multiple industry domains. To be considered a platform technology by the OMG, the technology must be proven and used in three or more different industrial domains. Therefore, an important requirement for the framework is to establish it as a platform technology as defined by the OMG.

1.6 An overview of the Framework

The DNEAF for EAIS and PPS reference model is a framework that includes four important concepts: Platform Pervasive Services comprised of essential services, High-Level Components, Enterprise Security Zones and Enterprise and Solution Development Tools and Services. Each concept is presented as an outline in the following subsections and will be described in detail later sections.

1.6.1 Platform Service Categories of Essential Services

- Directory Services
- Event Handling
- Persistence
- Transactions
- Security
- Messaging
- Format Transformation
- Rules Processing and Contents-based Routing
- Security/Authentication Support (via Single Sign-On)
- Workflow Execution
- Interfaces to Applications via Access Points

1.6.2 High-Level DNEAF Components

- Platform Services
  - Integration Services
    - Application Integration Services
    - Business Integration Services
  - Directory Services
  - Repository Services
  - Authentication Services
  - Management Services
- Information Pipe
- Access Point
- Integration Services Proxy
- Gateway Services
- Web Services
- Intranet Applications
- Extranet Applications
- Partner Services
- DMZ Applications
- Public Internet Access

1.6.3 Security Zones

- Intranet Zone
- Extranet Zone
- Partner Net Zone
- DMZ Zone
- Internet Zone
- Security Enclaves
- Segment Architecture

1.6.4 Enterprise and Solution Development Tools and Services

- Program and Project Management
- Risk Management
- Analysis and Requirements
- Design
- Build and Configuration
- Test Management
- Packaging and Deployment
- Environments
  - Development
  - Test
  - Integration Test
  - Acceptance Test
  - Production Environments
  - Disaster Recovery

1.7 Case Studies

Three case studies are used to demonstrate the application of the DNAEF in the thesis:

- E-Health Platform for Telemedicine
- Banking - Transitional Gap Analysis
- Vendor Product Evaluation - Comparative Analysis

2 Research methodology

The process utilized in developing the DNEAF for EAIS and PPS corresponds to the practice of Design Science Research Methodology (DSRM). DSRM "is a discipline oriented to the creation of successful artifacts" [14, 40].

2.1 Design Science Research Methodology

From a historical perspective, DSRM can be traced back to two movements to "scientize" design: The search for scientific design products in the 1920's and later a concern for a scientific design process in the 1960's culminating with "The Conference on Design Methods" held in London in 1962 [51] that is widely seen as the birth of scientific design. In the 1970's; however, design science fell out of favor. Design science once again became a popular topic during the 1980's and 1990's as journals dedicated to design science theory and research emerged [51]. It was not until 2007 that anyone looked at formalizing DSRM [51]. The Enterprise Systems Architecture Laboratory (ESAL) adopted a number of the concepts prescribed by Design Science in 1996 and has continued working towards a disciplined methodology based around DSRM.

Alan Hevner defined three closely related cycles of activities for DSRM [50]: "The *Relevance Cycle* inputs requirements from the contextual environment into the research and introduces the research artifacts into environmental field testing. The *Rigor Cycle* provides grounding theories and methods along with domain experience and expertise from the foundations knowledge base into the research and adds the new knowledge generated by the research to the growing knowledge base. The central *Design Cycle* supports a tighter loop of research activity for the construction and evaluation of design artifacts and processes. The recognition of these three cycles in a research project clearly positions and differentiates design science from other research paradigms." ESAL incorporates these steps by using them as extension points within the applied research process:

- **Relevance:** ESAL works closely with partner organizations (including IBM, Fujitsu Services, Microsoft, Oracle, Accenture and Ericsson) and international organizations (including the OMG) to field test artifacts (processes, methodologies and frameworks). Feedback from peer reviews and field testing is used to further refine each artifact as part of a continuous and iterative development process. This principle applies to the development of the DNEAF as a concept and to the DNEAF for EAIS and PPS as a pragmatic and usable framework.

- **Rigor:** ESAL participants are primarily from industry, who in turn, work closely with both scholarly researchers and industry researchers. An important aspect of this work is to ensure that the artifacts, including the DNEAF for EAIS and PPS are easy to adopt and use by practitioners. One important aspect is to also investigate various CASE and Development tools to see how they can be used with ESAL artifacts (ESAL develops solutions suits for both Rational and Sparx Systems CASE tools). ESAL also maintains an extensive knowledge base of the various consulting, application and service development

work that the organization is involved with, which includes a "sanitized" content area for researchers.

- **Design:** The DNEAF for EAIS and PPS has been developed iteratively over time and draws from over thirty real-world applications of the concept, three of which have been chosen for the case studies included in this thesis (that represent three different domains). Feedback and peer review (both from the client organization and supplier as well as independent reviewers from vendors and universities) are used to refine the framework. The DNEAF for EAIS and PPS has been successfully used in both small and global multinational enterprises (from 1998 through 2015):
  - Small to Medium Enterprise (SME): 25% - Budget range USD $150K to USD $50M
  - Large to Global Enterprise (LGE): 75% - Budget range USD $250M to USD $4B

  Note: To date, there is an existing budget gap for application of the DNEAF for EAIS and PPS between the SME and LGE cases ranging from USD $50M to USD $250M – this is not currently seen as problematic since the scalability has worked well in both sets of organizations (SME and LGE).

The following diagram, Figure 1, illustrates the Design Science Research Cycles [50]:

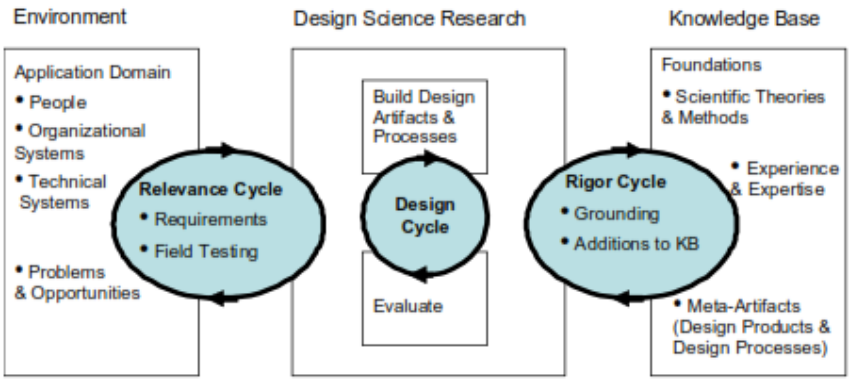


Figure 1: The Design Science Research Cycle based on [50]

Peffers et. al. outlined a methodology for DRSM that includes the following activities [49]:

- **Problem identification and motivation**: The activity includes the definition of the specific research problem and justification of the value of a solution. The problem definition will be used to drive the development of an artifact, which is the solution to the problem. Justifying the value of a solution accomplishes two things: it motivates the researcher and the audience of the research to pursue the solution and to accept the results and it helps to understand the reasoning associated with the researcher's understanding of the problem. Resources required for this activity include knowledge of the state of the problem and the importance of its solution [Definition copied from source 50].

- **Define the objectives of the solution:** The activity infers the objectives of a solution from the problem definition and knowledge of what is possible and feasible. The objectives can be quantitative, e.g., terms in which a desirable solution would be better than current ones, or qualitative, e.g., a description of how a new artifact is expected to support solutions to problems not hitherto addressed. The objectives should be inferred rationally from the problem specification [Definition copied from source 50]. In this this thesis, *the objectives* are interpreted as the requirements on the artifact.

- **Design and development:** This activity includes the creation of the artifact. Such artifacts are potentially constructs, models, methods, or instantiations (each defined broadly) or "new properties of technical, social, and/or informational resources". Conceptually, a design research artifact can be any designed object in which a research contribution is embedded in the design. This activity includes determining the artifact's desired functionality and its architecture and then creating the actual artifact. Resources required moving from objectives to design and development include knowledge of theory that can be brought to bear in a solution [Definition copied from source 50].

- **Demonstration:** This activity includes the demonstration of use of the artifact to solve one or more instances of the problem. This could involve its use in experimentation, simulation, case study, proof, or other appropriate activity. Resources required for the demonstration include effective knowledge of how to use the artifact to solve the problem. Resources required for the demonstration include effective knowledge of how to use the artifact to solve the problem [Definition copied from source 50].

- **Evaluation:** Observe and measure how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration. It requires knowledge of relevant metrics and analysis techniques [Definition copied from source 50].

- **Communication:** Communicate the problem and its importance, the artifact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences, such as practicing professionals [Definition copied from source 50].

2.2 Results of the application of DSRM

The results of the application of the DSRM's activities from [49] are described in the table, Table 2, below:

Table 2: Results of the application of the DSRM's activities

DSRM Activity	The application of the DSRM Activity
Problem identification and motivation	The problem addressed in this thesis is presented in Section 1.3: Many organizations introduce an Enterprise Architecture to support the integration of different services across the enterprise. However, they do not know where they either have or may encounter weaknesses in their Enterprise Architecture.
Define the objectives of the solution	The solution that address the problem is the DNEAF for EAIS and PPS. The objectives of the solution are interpreted as the requirements on the DNEAF for EAIS and PPS. These requirements are presented in Section 1.5.
Design and development	The solution, that is DNEAF for EAIS and PPS, is presented in Section 5.
Demonstration	The solution has been applied in several real-world solutions. Three implementations are presented as case studies included in this paper that, in turn, represent three domains (OMG Platform Technology requirement).
Evaluation	The solution has been evaluated by many different actors, such as internal, external supplier, external vendor and external independent peer reviewers. They have provided recommendations on how to refine the framework. IBM, Ericsson and the OMG were significant participants in defining the original baseline framework. In this thesis, no scientific evaluation is presented – results are based on real-world scenarios in which continuous improvement took place (the process is designed to develop a useable artifact rather than test a hypothesis). However, the demonstration in form of the three case studies can be seen as an evaluation. Moreover, in Section 7.2, the fulfillment of the stated requirements (on Section 1.5) is discussed by me, and can therefore be seen as a form of informal argument, and thereby, also be seen as a weak form of evaluation. as discussed by [50]
Communication	The DNEAF for EAIS and PPS, has been communicated to many different actors, some presented in Section 1.1.

It is important to keep in mind that the topic of this thesis is based on industry research designed specifically to create a useable solution, which necessarily requires continuous feedback to address stakeholder concerns – including business stakeholders and ICT practitioners. DSRM is useful in discussing DNEAF for EAIS and PPS from a scientific perspective.

3 Related research

In this Section, the DNEAF for EAIS and PPS is related to similar solutions, to show its role in the scientific knowledge base [50].

Two of the internationally recognized standards organizations that impact enterprise architecture are the OMG (e.g. UML and MDA) and The Open Group e.g. TOGAF) and both organizations' work has been implemented in ESAL applied research. However, the OMG and The Open Group differ in how they treat platforms. The OMG initiative with MDA explicitly addresses the high churn rate found in platform stacks as well as the tendency for many organizations to use multiple platform stacks out of necessity [63]. Alexander Osterwalder touches on this aspect with his Competition and Technology Change diagram illustrated below, see Figure 2, [52]:

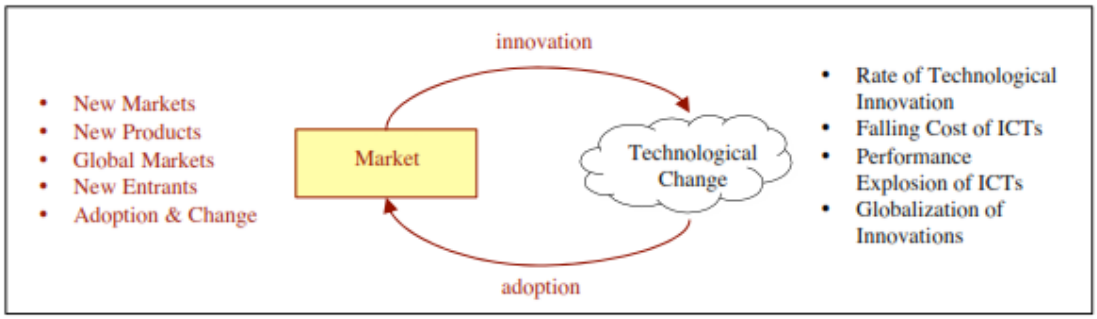


Figure 2: Competition and Technology Change diagram, by [52]

Archetype frameworks are useful in high-level abstractions by separating significant architectural concepts into more easily managed parts [53]. It is important to keep in mind that an MDA PIM can be reused across multiple domain implementations and that this reuse confers added value to an organization that adopts an archetype architecture framework such as the DNEAF for EAIS and PPS. It is also important to recognize that both MDA and the DNEAF for EAIS and PPS are usable within any Enterprise Architecture framework.

Winter and Fischer propose a set of essential enterprise architecture layers [54, 55, 56]:

- Business Architecture
- Process Architecture
- Integration Architecture
- Software Architecture
- Technology (or Infrastructure) Architecture

The DNEAF for EAIS and PPS primarily addresses architecture concerns within the Integration Architecture layer. There are many Enterprise Architecture Frameworks (EAFs) currently is use. Looking at the most common EAFs in use today [55, 56, 59] that ESAL has real-world experience with, the following table, Table 3, indicates how easy it is to use the DNEAF for EAIS and PPS:

Table 3: How easy it is to use the DNEAF for EAIS and PPS

Ease of Use	Enterprise Architecture Frameworks
Integration view directly supported	FEAF, TEAF and TOGAF
Integration view needs to be customized	Zachman and DoDAF

The DNEAF for EAIS and PPS deals with a complex subset of the overall Enterprise Architecture. Trying to include this subset in a more comprehensive view can create a complex model that is difficult to comprehend [57]. This becomes increasingly problematic when multiple parties are involved (e.g. outsourcing, suppliers, partners, vendors, mergers & acquisitions). As an archetype framework, the DNEAF for EAIS and PPS is useful when integrated with round-trip engineering tools that support reuse [60] through MDA transformations into platform specific implementations [61] through code and code stub generation [58]. ESAL has successfully demonstrated this capability in all three case studies included in this research paper.

Domain neutral framework approaches are becoming increasingly common in both solution architecture (e.g. IBM's San Francisco platform for programming business applications [61]) and enterprise architecture (e.g. policy neutral access control framework for heterogeneous environments [62]). Both the San Francisco and Policy Neutral Access Control promote functional reuse that new applications can exploit, which saves both development time and simplifies the architecture landscape. An interesting aspect of the Policy Neutral Access Control Policy is that it can in turn be incorporated into the DNEAF for EAIS and PPS as part of PPS essential services mapping [63].

Domain neutrality is also important regarding Service Oriented Architecture (SOA), which introduced a growing need for domain neutral modeling and documentation of architecture layers [64]. A key objective for SOA implementations is to decouple reusable functions so that both existing and new applications and services can be more easily integrated. A common goal behind the adoption of SOA by an organization is to move towards a model where "most software capabilities will be delivered and consumed as services" [67]. In the process, there has been a steady evolution "from modules, to objects, to components, and now to services. However, in many respects the naming of SOA is unfortunate. Whilst SOA is of course about architecture, it is impossible to constrain the discussion to architecture, because matters such as business design and the delivery process are also important considerations. A more useful nomenclature might be Service Orientation (or SO)" [67].

Many of the domain neutral concepts that ESAL worked with began with Object Oriented (OO) design principles. The limiting factor with objects is that, in isolation, an object is not very interesting. Objects working together as components become much more interesting. Component Oriented (CO) analysis and design quickly leads to a modular architecture approach from which it is relatively easy to evolve into a Service Oriented (SO) approach to analysis and design. At the SO level, middleware and essential platform services become an important enabler across the enterprise. An important approach to supporting SO, managing enterprise level integrations and managing process choreography is to develop an application domain neutral metamodel [66],

which is provided by Application Repository Services and Repository Services within the DNEAF for EAIS and PPS.

Domain Neutral Enterprise Architecture Frameworks (DNEAFs) are not limited to EAIS and PPS, ESAL is also currently working on a DNEAF for Security that focuses on a layered security model. Other organizations are also working on similar domain neutral architecture frameworks for solution and enterprise architecture as described above. As of June 2015, there are only a few well documented and proven domain neutral archetype frameworks. The reason for this is that it is difficult to identify, document and properly implement patterns and frameworks [65] – the challenges are increased when developing domain neutral patterns and frameworks since the pattern or framework needs to work with many domain instances.

4 Framework Concepts

This paper is driven from the need to develop a pragmatic approach to the analysis, design and implementation phases of the software development lifecycle process for large scale enterprise systems. The approach is architecture driven and based on the Object Management Group's Model Driven Architecture standard. This section contains a number of important foundation concepts important to understanding domain neutral architecture concepts.

The DNEAF for EAIS and PPS MDA solution provides a framework for analyzing different implementation approaches that can be used to build an enterprise platform. MDA itself prescribes a process that begins with a Computational Independent Model (CIM) based on the core business interests of the various stakeholders. The CIM is then refined into a Platform Independent Model (PIM) where the core business requirements are mapped into functional components. This functional mapping process defines proxy model elements that represent a functional component architecture. At this stage, a Gap Analysis can be performed by using the PIM as a baseline for comparing different Platform Specific Model (PSM) approaches.

Many of the current products and technologies related to web services and enterprise systems are relatively new (as new products, services and updates to existing products are rolled out). Keeping up with the technology is a challenge – even more so for SME organizations (SME – defined by the OECD as a company with up to 500 employees and annual revenues up to USD $50 Million). Traditional enterprise system technologies are typically expensive and require highly specialized skill-sets to properly implement. KUH and other healthcare providers, especially those in the developing world, often do not have staff within the organization with adequate skill-sets or adequate funding to engage external experts. Addressing analysis and design issues early on in a project can play an important role in reducing risk and costs. [8, 40] The current practice of building dedicated (vertical) applications is expensive, difficult to reuse many of the developed artifacts, and hard to implement in an environment where funding and resources are in short supply. Additionally, different projects compete for the limited funds and resources that are available.

4.1 Managing Risk and Cost

An important objective of any business activity is managing risk in terms of implementing the originally requested requirements, delivering on time based on the original schedule and staying within the original budget. As a program or project advances during each phase and through different phase iterations, the accumulation of work increases. In this sense, all ICT development activities tend to be Waterfall at a macro level even though they are often Iterative at the micro level. As time passes, more documentation, code and other deliverables accumulate. The later into the process that a change needs to be introduced, the greater the potential for delay and additional cost – especially since a change may directly and indirectly impact many of the completed deliverables. This impact is realized in more development and testing to accommodate the changes. The changes themselves may spawn even more changes. The overall impact of change is typically greater when the environment is complex – a common characteristic at the enterprise level. The table below, Table 4, illustrates how costs increase over time – it is also important to keep in mind that costs associated with earlier phases can become quite large in the later part of the lifecycle, even when an iterative process is being used, due to the accumulation of deliverables that require maintenance. One of the best ways to visualize this accumulation effect is

to look at the Spiral Model [10, 47] as each revolution of the spiral completes a mini lifecycle iteration.

Table 4: Relative Cost of Correcting a Software Error [13, 46]

Lifecycle Phase	Relative Cost of Correcting an Error (Ratio)
Requirements Phase (Analysis)	1:1
Design Phase	1:3 to 1:6
Coding Phase	1:10
Development Testing Phase	1:15 to 1:40
System Testing Phase	1:30 to 1:70
Field Operation (Product Deployment)	1:40 to 1:1000

Note: (One Resource Unit Equivalent):(Multiplier)

One Resource Unit Equivalent is the amount of labor, capital, money, and time spent to accomplish one established unit measurement of productivity. The basis is usually equated in economic man-hours of production which factors in the resources and capital required to support one man-hour of output. For instance: During the coding phase, for each hour spent on incorrect coding, ten hours of coding will be required to correct the problem.

The impact of change on cost during a waterfall lifecycle or iteration of a lifecycle is illustrated in the figure below, Figure 3. Illustration A shows a typical Cost of Change Curve. Illustration B shows the effects of mitigating risk, which results in lower costs and a flattening of the cost of change curve [13].

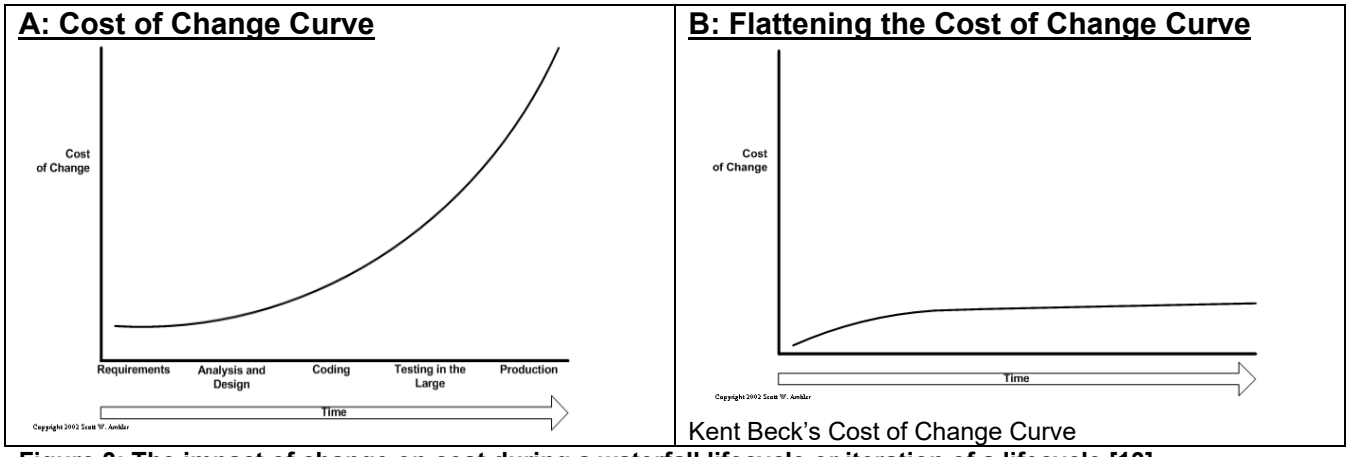


Figure 3: The impact of change on cost during a waterfall lifecycle or iteration of a lifecycle [13]

One of the perceived benefits of Agile Development [4] is that it can reduce risk; however, Agile Development methodologies can still incur costs if risks are not properly managed. The earlier a defect is discovered (or a change is required), even in an iterative agile process, the lower the cost to make a correction. Examining the Agile Cost of Change Curve [13] below, see Figure 4, illustrates this point:

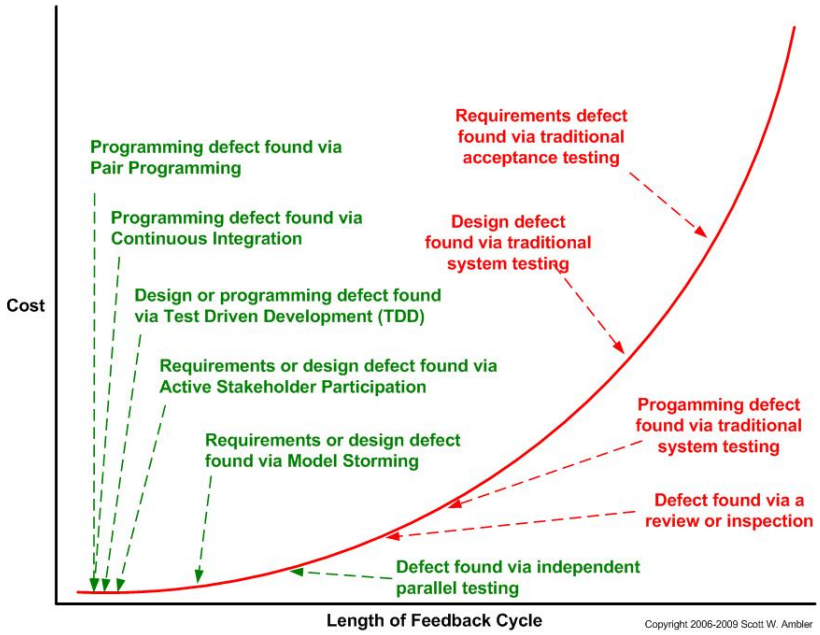


Figure 4: Agile Cost of Change Curve [13]

Ensuring that requirements are properly captured and understood is one step to reducing risk. Another important step is to spend more time on analysis and design. Elaboration of the analysis and design process creates a better understanding of the problem domain and often results in significant opportunity for refinement before any implementation begins. This is important since changes made in analysis and design phases are less costly compared to making changes to code and associated deliverables. Poor design can also require changes to the underlying platform, changes to hardware and will result in more testing.

The DNEAF for EAIS and PPS helps reduce risk by ensuring that the platform and applications are sufficiently understood, thus allowing an opportunity early in the lifecycle to find alternative approaches when there is an issue that poses a risk.

4.2 Software Maintainability

Software is developed or engineered; it is not manufactured in the physical sense. As long as a software application or service provides value to the business, it can stay in production. From this perspective, software does not wear out. Software can, however, become obsolete if it is not maintainable or maintained [41 (See Part 1)]. The ability to continue using software over a longer period of time increases the total value of the original investment made to acquire / implement the software / service [37].

Software is based on the collaboration of classes into components; collaborations of components into applications; applications in turn provide services to the enterprise. The use of composition in software design is a technique for developing loosely coupled systems, which helps reduce dependencies between classes and components [6 (pp 49-51), 7]: "Design with

composition for greater flexibility, utilize active notification to design loosely couple systems, use interface-centric design to provide flexibility in the form of extension and plug-in points." When developing an enterprise platform comprised of bespoke and off-the-shelf software, integration issues become a priority that requires a consistent way of modeling business events and data. Three important mechanisms for managing this risk include:

Canonical Data Model: "is an Enterprise Integration Pattern. A data model that represents the inherent structure of data without regard to either individual use or hardware or software implementation." [70]

Enterprise Canonical Data Model: "The data we all agree on. This is not ALL the data. This is the data that we all need to agree on in order to do our business." [69]

Business Event Ontology: "A reasonably complete list of business events, usually in a hierarchy, that represents the points in the overall business process where two "things" need to communicate or share. This does not refer to a single event, but rather to the entire list. Note that a business event is not the same as a process step. An event may trigger a process step, but the event itself is a "notification of something that has occurred," not the name of the process we follow next." [69]

4.3 Domain Neutrality

Domain Neutral Components and Frameworks are very useful building blocks that can be used across many application domains [6 (p 15)].

There are four Classification Archetypes [6 (pp 2-6)] that most Objects can be mapped to:

- **Catalog-Entry-Type Description** - Category, Subcategory, Type
  - Something that we need to describe but do not need to uniquely identify
  - Usually the most stable of our objects
- **Party/Place/Thing (PPT)** - A specific Product or Service
  - Something that we need to uniquely identify – a specific product or service
  - Usually relatively stable but can still have occasional change
- **Role** - A Product or Service Role
  - A way of participation by a party (person or organization), place, or thing that has a context that is important to the business
  - Roles are typically less stable since this area of the business tends to change more
- **Moment/Interval** (MI) - Events that are important to the Business
  - Either a moment-in-time or an interval-of-time that is important to the business
  - Usually the most unstable of our objects since these are prone to frequent change

There is also an **Extension Archetype** that is used as a deliberately designed point of extension (where all of the other Objects will map to).

In MDA, a PIM is Domain Neutral with regard to technology and platform stacks.

The DNEAF for EAIS and PPS is a domain neutral reference framework that can be used across many industry domains.

4.4 Patterns

Pattern: "The description of a general solution to a common problem or issue from which a detailed solution to a specific problem may be determined. Software development patterns come in many flavors, including but not limited to, analysis patterns, design patterns and process patterns" [1].

Archetype Pattern: "A form or template that all patterns of the same kind 'more or less' follow that specifies attributes, links, methods, plug-in points, and interactions that are typical for the category of pattern" [6].

The OMG defines a model in terms of MDA as: "The primary feature of MDA which enables us to deal with complexity and derive value from models and modeling is defining the structure, semantics, and notations of models using industry standards – models conforming to these standards are "MDA Models". MDA models can then be used for the production of documentation, acquisition specifications, system specifications, technology artifacts (e.g. "source code") and executable systems." [48, 5 (p 50)]

Modeling is essential in complexity management. Modeling benefits include:

- Viewing systems from multiple perspectives
- Discovering causes and effects using model traceability
- Improving system understanding through visual analysis
- Discovering errors earlier and reducing system defects
- Exploring alternatives earlier in the system lifecycle
- Improving impact analysis, identifying potential consequences of a change, or estimating modifications to implement a change
- Simulating system solutions without code generation

Modeling helps you reuse parts of existing information and knowledge in your new projects, saving time and money.

Abstraction, as defined by ISO 10746-2 RM-ODP, is "the suppression of irrelevant detail" [5 (p 51)]. An example of Abstraction is using the DNEAF for EAIS and PPS as a reference model to represent the essential services that an Enterprise relies upon for application and service integration. Platforms and Platform Stacks are subject to a lot of change – this "churn" can create problems for any applications and services that rely on the platform. Changes to a Platform or Platform Stack can require changes to the software.

A Platform "`is an execution environment for models`" and platforms can form platform stacks [5 (pp 51-53)]. In MDA terms, code can be considered a model. Computing systems or applications may take on specialized attributes in either their hardware or software - that is, they may be scalable, Real-Time, Fault-Tolerant, or designed to fit in a confined environment (Embedded). When these services are defined and built on a specific 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 Pervasive Services at the platform-independent model level in UML. Only after the services' features and architecture are fixed will platform-specific definitions be generated for all the middleware platforms supported by the MDA. At the abstraction level of a platform-independent business component model, services are visible only at a very high level (similar to the view the component developer has in CCM or EJB) [5].

A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented [48]. In principle a platform is a place to launch software on. Examples of platforms are operation systems, the Java Virtual Machine, runtime libraries of programming languages, etc. In computing platforms, for example, computer hardware serves as platform for an operating system which in turn is a platform for Enterprise Infrastructure Software which in turn is a platform for application software. Platforms typically reduce the amount of development required since the platform provides a common set of reusable functionality.

4.5 Model Driven Architecture

Model Driven Architecture (MDA) defines three different categories of models [5, 12]:

- **Computation Independent Model (CIM)** - Models that represent the business view
- **Platform Independent Model (PIM)** - Models that bridge the business and Information Technology views and that are independent of the technology used to implement the solution
- **Platform Specific Model (PSM)** - Models that represent the Information Technology view

All three model categories are essential in full life-cycle IT programs and projects. Each model category can exist at different levels of abstraction. Within each model (CIM, PIM, and PSM), there is a hierarchy of abstractions beginning with a high-level view and progressively drilling down into more detailed views. The three views are traditionally illustrated as a three-tier stack with the CIM on top and the PSM on the bottom, see Figure 5.

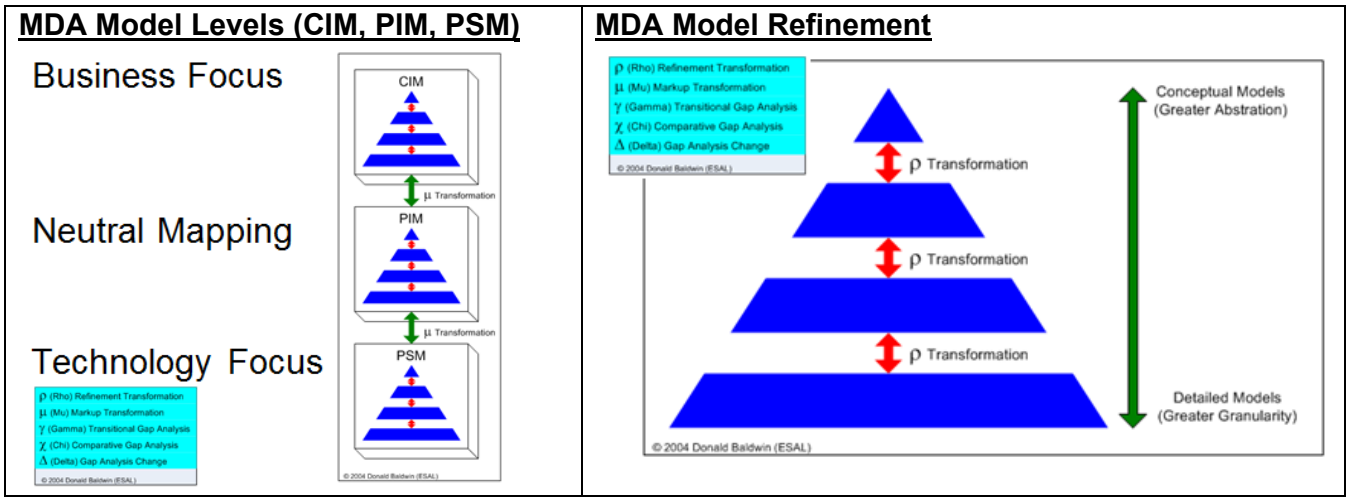


Figure 5: Model Driven Architecture (MDA)

To better understand the concepts in this document, it is more helpful to illustrate the CIM to the left and the PSM to the right of the PIM. In this view, the PIM can be seen as the fulcrum, or balancing point, for the overall model and in comparative analysis during PIM to PSM evaluations the PIM can serve as an archetype pattern framework [5 (pp 56-57)]. CIM models describe the business and its processes, stakeholders, workflows, and business entities from the perspective of the business. PSM models describe the IT solution and its processes, data structures, data transformations, workflows, and implementation from the perspective of the MIS department (an Information Technology perspective). The PIM models serve as a mapping between the business and the IT implementation [5 (pp 55-56)], see Figure 6.

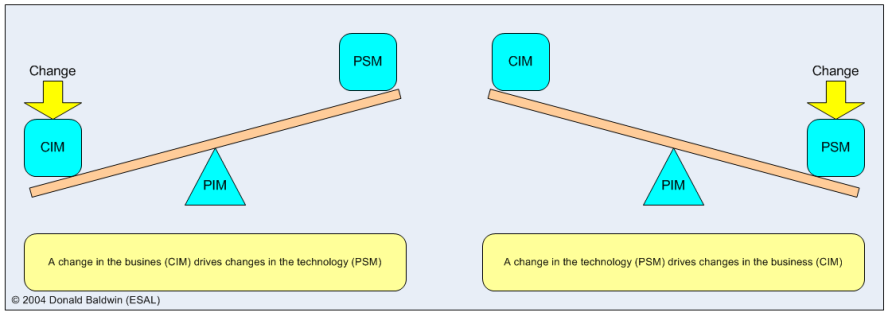


Figure 6: The PIM models serve as a mapping between the business and the IT implementation

The PIM provides the roll of a fulcrum between the CIM and the PSM in managing change by maintaining traceabilty between the CIM (business) and the PSM (technology), see Figure 7.

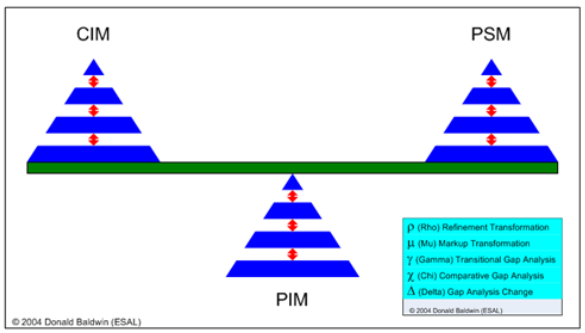


Figure 7: The PIM as the roll of a fulcrum between the CIM and the PSM

A PIM provides the formal specifications of the structure and the functionality of the system abstracting away both the business and the technology details. As its name suggests, it describes the components and their interactions in a platform-independent way. A PIM is can be mapped to both a computationally independent business model, referred to as a Computation Independent Model (CIM) and to a technology and product specific model, referred to as a Platform Specific Model (PSM). Importantly, A PIM can be mapped to multiple CIMs and PSMs.

There are four different transitions defined involving PIMs, PSMs, and CIMs (referred to as MDA model transformation [5 (p 53-55)]:

- CIM to PIM: When a CIM has come to a high level (view) of abstraction, it can be mapped to a PIM. Platform Specific Models can, such as PIMs, exist at different levels of abstraction. This transition involves moving from a highly detailed CIM to an abstract PIM. The goal is to shift from the detailed business processes and organizational structure to a functional representation of the business requirements. An example of a CIM to PIM transition can take place when planning to introduce a new software product into an organization to meet the business requirements.

- PIM to PSM: When a PIM has been developed to a high level of detail, it can be mapped to one or several PSMs. Over the past few years (up through 2004), the PIM to PSM transition has received the greatest amount of attention regarding published documents on MDA. An example of a PIM to PSM transition can take place when mapping different products to meet required functional and nonfunctional system requirements.

- PIM to CIM: When a PIM has come to a high level of detail, it can be mapped to one or several CIMs. This transition mapping is not typically emphasized in the available (as of 2004) MDA literature, but it is implied. An example of a PIM to CIM transition can take place when reengineering business processes in response to changes in the Information Technology.

- PSM to PIM: When a PSM has come to a high level (view) of abstraction, it can be mapped to a PIM. This transition involves moving from a highly detailed PSM to an abstract PIM. The goal is to shift from a detailed description of the information technology and underlying systems processes to a functional representation of the information system requirements. An example of a PSM to PIM transition can take place when planning to introduce a new Enterprise Resource Planning (ERP) product into an organization, which typically requires changing the existing business processes to fit the ERP product.

5 DNEAF for EAIS and PPS Framework Specification

In this section the artifact developed, that is DNEAF for EAIS and PPS, is presented.

Note: *The contents of this section have been previously made available as works in progress, incorporated into client projects and have been published to the web. Additionally, the worksheets derived from this work have been used in many organizations in both a commercial and research context* [5].

An Enterprise Systems Platform consists of components which offer core functionality to carry out fundamental processes in the business. These components can vary from domain to domain but there is a constant set of components which must exist in a true Enterprise Platform. The problem with assessing enterprise platform components is compounded by the practice of product vendors to describe their architectures from the perspective of their product stack and not from the perspective of functionality. To effectively compare different products, a baseline of functionality must first be established. This document presents an architectural model that is neutral regarding both the business and technology domains to identify the core components that exist within an enterprise portal platform. It is these components, which we call Proxy Components, whose functionality in the PIM is "marked up" to reflect specific technology and component implementation mappings or specific organizational processes and workflows. These core Proxy Components are described from the perspective of an Enterprise Portal / Enterprise Integration Architecture and are described in this Section [37].

Before introducing the core components, it helps to first define the categories of functionality, described as services, that the platform must provide. Any application, independent of its environment, is dependent on a set of what can be characterized as essential services. The OMG has defined five essential services as pervasive services. Pervasive services specify services that an infrastructure component offers to other assembled components. [14 (p 21)] The pervasive services identified by the OMG include:

- Directory Services
- Event Handling
- Persistence
- Transactions
- Security

An Enterprise Application Integration (EAI) platform must also provide the following six categories of functional enterprise services to all applications and in all environments; these services provide support for:

- Messaging
- Format transformation
- Rules processing and contents-based routing
- Security/authentication support (via Single Sign-On)
- Workflow Execution
- Interfaces to applications via access points

The various components that comprise an enterprise platform provide services to each other and to the applications that utilize the platform. When describing the platform components, it is common to refer to them in terms of services. For instance, the Content Management component would typically be referred to as Content Management Services (or by its shortened form, Content Services) since it is responsible for providing Content Management Services to the enterprise via the platform. Applications can also provide services to other applications and platform components when they have been integrated into the platform framework. In the component descriptions that follow, many of the components will be referred to as services. This indicates which components provide core functionality to the enterprise. Some components will be described as sub-components to indicate a special technical approach that needs to be explained. Other components will be referred to as mechanisms to indicate a technical approach for providing specified component functionality.

Each service within the enterprise platform will typically be assigned to a specific network security zone. From the enterprise perspective, there are five network security zone classifications (each zone can be further divided into "compartmentalized" zones):

- **Intranet** – This is the most secure zone in a network. The Intranet is where the enterprise maintains its most sensitive applications and data. This is also the zone where application and data development takes place.

- **Extranet** – This is the second most secure zone in a network. The Extranet is where the enterprise maintains applications and data that are accessible by employees that are located outside the company network and by partners. Some application and data development may take place in this zone depending on the requirement to have external partner participation in the development from outside the enterprise.

- **Demilitarized Zone (DMZ)** – This is the least secure zone in a network. The DMZ is where web servers, applications, and data for external customers, partners, and the public are maintained. Development should never take place in this zone – artifacts built in more secure zones are deployed to the DMZ.

- **Partner Net** – This is a network zone that is under the control of a partner organization. This zone is considered less secure than the DMZ because the enterprise typically lacks operational and audit control in a partner's network – even when connecting to an Extranet or Intranet. Security is a critical issue since it is possible that a chain of trust can exist whereby access to and through a Partner Net zone can be indirect (for instance Organization A connects to Organization B and Organization B connects to Organization C – if the trust relationships and security configurations are not set up properly, it may be possible for A to connect to C through B or even for C to connect to A through B).

- **Internet** – This is the least secure zone – a public network. Any communications sent through the Internet should be considered unsecured. Even with encryption, any information could still be compromised by a third-party willing to pay the costs. Information in this zone should be assessed for its value and timeliness. Information of high value but whose value is very short-lived is safer to transmit over the Internet compared with information that is of high value but whose value is long-lived.

In addition to the impact of security on the enterprise architecture, there is also a significant impact on scalability. In small to medium transaction volume platforms, the correlation between the

number of transactions and the decrease in performance is geometric. In large-scale enterprise platforms, the correlation will reach a point where it becomes exponential. This exponential decrease in performance on large-scale enterprise platforms is one of the most serious architectural problems that an organization will face. There are several significant contributing factors to this problem, the three most common being:

- **Application Servers** – Application Servers are used to manage transactions, sessions, and persistence through a "container." Both IBM and Oracle rely on Java Application Servers based on the J2EE standard that manage the Enterprise Java Beans (EJB's). The Open Source community has also adopted a similar approach (for instance, Red Hat Linux Enterprise running the JBoss Application Server); however, there are also alternative Open Source solutions such as Erlang, which is based on the Open Telephony Standard. Microsoft has not adopted an application server for .Net (the closest thing Microsoft has is BizTalk – but it is not an Application Server in the sense that a J2EE Application Server is. BizTalk Server has both an internal and external, via Azure, application server from 2006 to present.) [24, 25, 26, 42, 43]. The new EJB3 standard introduced a major impact on Java Application Servers. IBM for instance, was an early adopter in bringing their Application Server architecture in-line with the EJB3 standard.

- **Clustering** – Clustering functionally related servers together is a common way to share processing loads; however, it is also difficult to develop this technology. Many organizations do not have an adequate budget to invest in development, test, and production infrastructure environments. Instead, development is often done on a smaller scale using lower cost and performance infrastructure. An additional problem is maintaining synchronization between different persistence objects that reside on different servers within a cluster without creating a huge process overhead. Running Application Servers on a Cluster of servers can create numerous problems that are not often identified during development – especially if the development infrastructure is based on a single-server solution or on operating system and development environments not specifically designed for clustered deployment. Microsoft Clusters also require a high-level of management and operational synchronization that creates a performance bottleneck as the solution scales up.

- **Caching** – Caching can be done at many locations and levels within an enterprise architecture; for instance, there is persistence layer caching, web or HTTP session layer caching, and business layer caching. Alternatives to these solutions for caching can include the use of stored procedures or the use of a dedicated caching solution. In a clustered environment, there is the added challenge of making sure that every instance of the same information has the same value. Caching can create a performance bottleneck during synchronization and garbage collection.

The diagram below [41], see Figure 8, provides a high-level overview of the Enterprise Portal and Integration Platform in the form of a Model Driven Architecture Platform Independent Model. This model view is based on a real-world solution developed by the Enterprise Systems Architecture Laboratory that has been successfully deployed in many domains including financial services and banking platforms; business-to-business platforms; telecom service provisioning portal platforms; air traffic management platforms; healthcare platforms; e-business portal platforms; and e-government portal platforms.

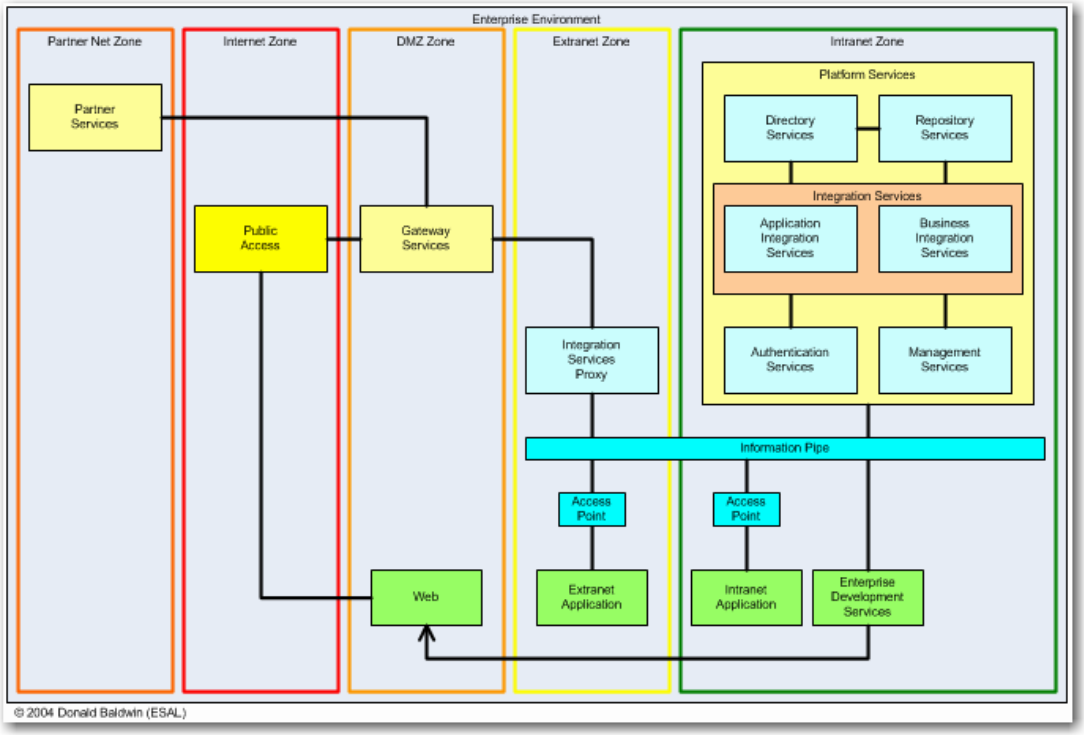


Figure 8: A high-level overview of the Enterprise Portal Integration Platform in the form of a Model Driven Architecture Platform Independent Model [41]

The model illustrated above has also been successfully deployed using IBM WebSphere, Oracle, Microsoft, Fujitsu Interstage and Open Source components including the Open Telephony Platform (OTP) with Erlang by the Enterprise Systems Architecture Laboratory in the domains listed above. Each of the services (components) is described in more detail on the following pages.

5.1 Directory Services

Directory Services are specialized catalogs intended for reading more than writing. In the context of EAI, Directory Services provide distributed access to enterprise metadata (data that is used to describe other data) such as application location, network, user, and security profile information. Since the information contained within the Directory Services is primarily read, it should ideally be stored using a technology that is optimized for reading. The most common choice of technology is the Lightweight Directory Access Protocol or LDAP for short.

LDAP is a reference, vendor-independent, network protocol used to access the information in the directory. LDAP is used to describe a rich set of attribute types and syntaxes based on X.500. It defines not only the information model and a hierarchical namespace – how information is structured and referenced – but also the operations that can be performed.

The Directory Services platform component is responsible for:

- Management of directory contents, access, replication, and configuration
- Typically stores all application location, network, user, and security profile/policy information in LDAP or LDIF formats
- Reference interfaces are typically via LDAP API using defined interface classes
- Cross-reference index to repositories for version control & dependency management

Current implementations of LDAP that have adopted the extended operation for cryptography can support SSL and TLS (while part of the standard, this functionality is not widely adopted due to it being based on an extension – based on StarTLS). Encryption support is available in Open LDAP, RedHat Linux LDAP and Microsoft AD. [44, 45]

5.2 Repository Services

Repository services are used for the management of metadata objects for application support including (but not limited to):

- Managing business model, process, and rule specifications
- Component and legacy system interface specifications
- Transactions, workflows and workflow elements
- Message, object, and data-store specifications
- Routing and platform specifications
- Manages catalogue interfaces to information systems and products

All information is represented in XML using DTD's and/or schemas. The reference interface is via a Lightweight Directory Access Protocol Application Program Interface (LDAP API) using defined interface classes. It uses a cross reference index to the Directory Services for version control and dependency management.

The Repository services can be accessed on a read only basis by legacy applications via the Access Points. The Repository Services data is maintained and infrastructure information is updated by the Application Integration Services component. Management of metadata objects for application support includes (but is not limited to):

- Business Model, Process, and Rule Specifications
- Component and Legacy System Interface Specifications
- Transactions, Workflows, and workflow elements
- Message, Object, and Data-Store specifications
- Routing and Platform Specifications
- All information represented in XML (DTD and/or Schema)
- Reference interface via LDAP API and java interface classes
- Cross-reference index to directory for version control and dependency management

5.2.1 Business Process Repository

The Business Process Repository is a specialized sub-component of Repository Services that is used to describe Business Processes in terms of business goals, enterprise architecture, and business operations. This sub-component is important for managing the collaboration between different business domains and their processes (workflows, transactions, and business objects) both within and external to the organization. This is especially important when the collaborations involve proprietary standards and ERP solutions. The Business Process Repository defines what the processes are by providing:

- Process descriptions
- Transaction descriptions
- Processing rules (these are not typically computational rules but rather a mechanism for exposing the business process rules needed for meeting legal and regulatory requirements. An example of such a business process rule is HL7, which defines over 80 data structures for standardized use in healthcare)
- Business Objects (anything that can be physically or logically represented)
- Business Object Mappings (mapping business objects typically requires significant semantics – UML is not good in representing this. Mappings are dependent on the Business Process Repository and the Message Dictionary)

5.2.2 XML Repository

The XML Repository is a specialized sub-component of Repository Services used to define data structures and behaviors in the form of XML Schemas and DTD's.

5.3 Application Integration Services

The Enterprise Model needs to provide integration for existing and future applications. Not only integration of commercially available off-the-shelf applications and databases, but it must also offer a means of collaboration with business partners and clients. The Application Integration Component Model should provide the following:

- Mechanisms for the exchange of data and messages among an increasing number of applications and components that are required to supply application transactions or to fulfill a business process
- An integration strategy that provides flexibility, scalability, robustness, and reusability

There are two views within Application Integration Services. The User View corresponds to what platform clients see as exposed services for the various enterprise applications running on the platform. The Developer View is concerned with the resources and processes behind the services.

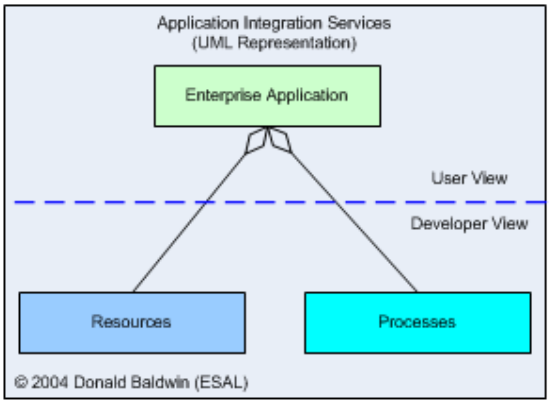


Figure 9: The two viewpoints in Application Integration Services

The diagram above, see Figure 9, illustrates the two viewpoints in Application Integration Services: The user view (or platform client view) and the developer view. Users (defined as human and system) see an enterprise application as a service that is available on the platform as either an application or workflow. Developers are concerned with the resources and processes behind the service. A platform approach to application support promotes reuse since applications can use existing artifacts. Application Integration Services relies on XML-based repositories to manage artifacts (the functional components and data that makeup the platform and that describe platform functionality). Unless there is an extraordinarily disciplined development team, many programming artifacts such as EJB's are not typically reusable. While this is contrary to popular thinking, it is significant since at least 60% of the development work on an enterprise platform is configuration – only 40% is coding. EJB development work is not finished when the coding is done. Most projects run into trouble because the project managers do not understand this issue. Much of the reuse is in the form of metadata, workflows, and repository artifacts.

The diagram below, Figure 10 shows the four most important repository artifacts (out of the core thirteen) that contribute most to promoting reuse within the enterprise platform.

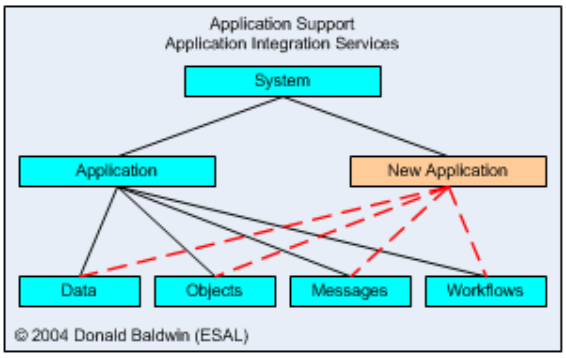


Figure 10: The four most important repository artifacts

As new applications are added to the platform, there is a greater opportunity for reuse since many business objects, especially within a vertical domain or business function space, are used throughout the enterprise. ERP and core business objects play a major role in many business processes. [3]

5.3.1 Basic Messaging Services

Basic Messaging Services support basic messaging functionality for applications. Messages from applications, application adapters, or other sources must be forwarded to their destination based on parameters and recipient message queue identities. Additionally, support is required for queue management (creation, connection, disconnection) and different messaging protocols and interfaces. Support must also be provided for file and object transfers over the transport layer with guaranteed delivery and notification. The result is a mechanism for communicating business transactions between systems in an enterprise or between enterprises. Functional requirements include:

- Creation of, connection to, and disconnection from queue managers and queues
- Support for datagrams, request/response, and publish/subscribe messaging protocols and interfaces
- Supports file and object transfer over the transport layer
- Guaranteed delivery and notification services
- Callouts to system management and security exits
- Exception Management

Specific strategies for realizing this functionality in Java environments is provided by using high-level Enterprise Java Beans and API libraries to support basic messaging functionality for the applications. IBM WebSphere provides extended support for Basic Messaging Services using the IBM Application Messaging Interface (AMI). [16, 17]

5.3.1.1 Message Dictionary

A Message Dictionary is a repository of defined message structures for messages that will be used on the platform. The structures are typically defined using XML Schemas and DTD's. Currently, the XML Schema is the dominant mechanism used to define messages. This is because Schemas provide more information about the defined objects and can be used for validation. The primary difference between the two is that DTD's define structure while Schemas define both structure and behavior. The Message Dictionary is an important service in validating messages to make sure that each message is recognized and that the contents conform to the expected structure and data types. The Message Dictionary is defined by the business processes. For every transaction, there are two messages that need to be defined by XML Schemas: One is the request and one is the response.

- Message structures are defined on the Message Dictionary
- Transformations are maintained in the XML Repository since the transformations are application specific

5.3.2 Data Services

Data Services provides a logical view to how data is stored within the enterprise platform environment by describing the structure of exposed data and exposed services (including stored procedures) in the various databases within the enterprise. Data Services provides access to databases, object servers, and datasets in file format and is defined in terms of interface managers (i.e. JDBC, ODBC, Dynamic SQL) and CORBA (Common Object Request Broker Architecture) IDL (Interface Description Language) specifications and message structures [12]. An important requirement for Data Services is the provisioning of a caching infrastructure with caching synchronization mechanisms and runtime management of transactions. Data Services Provides

access to databases, object servers, and datasets in file format (Examples include Oracle, DB2, Business Objects, ORB Interfaces).

A database can have hundreds of security views, each of which must be maintained within Data Services. Strategies for realizing this functionality include using Metadata descriptions of data store interfaces that are held in a repository and accessed via configuration tools, EJB's, API's, and workflow element interfaces. Currently only SeeBeyond and WebLogic have enterprise level support for Data Services – but the support is only partial at the time this document was written. Oracle provides internal support for Data Services [20], but there is not adequate support at the enterprise level (the only way to access Data Services externally is to use Oracle Tools). WebSphere does not currently have support for Data Services (Note that as of 2009, WebSphere has a REST API for eXtreme Scale).

5.3.2.1Data Dictionary

A data dictionary describes the data that is maintained within the enterprise platform environment using metadata. The Data Dictionary is an important requirement for enabling different platform services and applications to share data. Data Dictionaries exist at multiple layers within the platform. The best known are those associated with relational database products where the Data Dictionary is used to describe the structure of the tables, records, and fields contained within the database. In an enterprise platform, the Data Dictionary is used by the Integration Services to enable the sharing of data between the platform, applications, and partner systems.

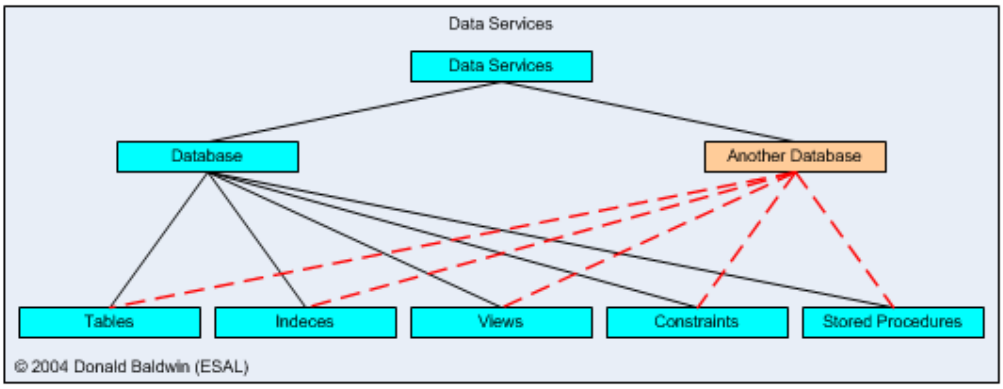


Figure 11: Illustration of how the repository framework can be shared by multiple applications and databases to promote information sharing across the enterprise platform

The diagram above, Figure 11, illustrates how the repository framework can be shared by multiple applications and databases to promote information sharing across the enterprise platform. In this way, Data Services maintains the logical view of all the databases on the platform.

5.3.3 Transformation Services

Transformation Services is responsible for the transformation among different message formats and representation languages using transformation rules based on formal specifications. Transformation rules apply to the sender's message format to obtain the receiver's required format of the same message. Transformation rules can also be used to filter the sender's message by forwarding only the parts of the message that are relevant to the receiver. The most common approach for realizing Transformation Services is to rely on XML as the standard for intermediate representation. Many vendors use XSLT as the transformation mechanism, which can create performance bottlenecks in large-scale enterprise systems. To address performance issues, there are emerging dedicated transformation solutions being introduced.

Strategies for realizing this functionality include defining high-level interfaces in the form of Enterprise Java Beans, Application Programming Interfaces (API's), and using event triggers in workflows. The Transformation Engine can also be enabled for triggering or executing business rules that can be specified in a formal Business Rules Language. Key functionality includes:

- Transformations are specified using conversion maps and rules between message/object structures
- Transformation mappings should support both fan-in and fan-out of messages

5.3.3.1Transformation and Routing Engine

A message sent across an enterprise platform needs to be able to be processed by the appropriate platform, application, and partner services. Additionally, the platform needs to know where to send messages. In Enterprise Application Integration (EAI) involving different applications and Enterprise Resource Planning (ERP) solutions, there are often different API's and messages that are proprietary to each application or ERP product. The Transformation and Routing Engine is responsible for "translating" messages between these different components and services as well as making sure that each message gets to its intended destination.

The Transformation and Routing Engine is closely coupled with the XML Engine. One of the important assumptions when implementing an enterprise platform is that everything will be based on XML. The XML Engine (covered in detail in the next section) uses XPath for header information and provides the XML parser and validation engine (validation is typically dependent on the associated Schemas). The Transformation and Routing Engine manages all the rules and calls out to the XML Engine and parser to process messages.

A common strategy for realizing Transformation and Routing functionality is to adopt a platform standard for messages and objects for use on the platform. Such a standard is referred to as a profile in Model Driven Architecture (MDA). Working with the Object Management Group (OMG), the Enterprise Systems Architecture Laboratory has refined these profiles into seven layers:

- **Application Design Profiles** – Applications that comprise enterprise (vertical) application components and (horizontal) integrated application solutions.
  An Application Design Profile is specific to each individual application.

- **Application Profiles** – Vertical domain profiles for components and applications with supporting specialized domain architectural description languages.
  Examples include transportation, telecommunication, and healthcare domains.

- **Platform Implementation Profiles** – Development platform profiles from which applications are built.
  Examples include J2EE, .Net, and CORBA.

- **Data Management Profiles** – Profiles that define identification, control, policies, and procedures for managing data, typically adopted by standards organizations that are formally defined.
  Examples include CWMI, MIS/EIS, and Database standards (i.e. RDBMS).

- **UML Standard Profiles** – Infrastructure templates, typically adopted by standards organizations that are formally defined.
  Examples include real-time, middleware, and embedded standards.

- **Data and Metamodel Profiles** – Provides the capability to extend the UML through existing standards to represent UML designs in a stored data repository and to provide a mechanism to describe all database representations and interfaces.
  Examples include MOF, TOGAF/ADL, and XMI.

- **Core UML Support Profiles** – Unified Modeling Language (UML) notation coupled with an Object Constraint Language (OCL) and supporting methodology.
  This is the formal specification for the core UML as defined by the OMG.

A given application will typically use at least one – and often more than one – profile from each of these seven layers. The use of Standard Profiles is an important first-step in the emerging development of code generation technologies based on MDA. An early example of a tool that incorporated the concept of Standard Profiles as a "plug in" for modular development is ArcStyler (a J2EE development suite *now out of production*).

5.3.3.2XML Engine

The most common XML Engine in use over the past few years is XSLT (Extensible Style sheet Language Transformations). Most XSLT engines share a common code base, which has significant performance limitations when there are large-scale transformation requirements. Oracle, Microsoft, and most Open Source technologies are impacted by this performance issue. IBM and other third parties have developed dedicated transformation engines that are more efficient with better load balancing and scalability characteristics – a critical consideration in large-scale enterprise architectures or in architectures where there is a high volume of transformations.

There are two strategies to help manage performance issues related to message transformations:

- Use a message transformation engine that is not based on the common XSLT code base. There are many solutions that can be used; however, it is critical to consider the specific enterprise architectural requirements for the target domain to address scalability, load balancing, availability, and robustness characteristics.
- Adopt a common platform standard for messages. This involves two strategies: First, in a distributed application environment, outsource message transformations to the individual applications (each application transforms messages from its internal format to a platform standard format). Second, in an application integration scenario, use Access Points (described later in this document). This strategy helps reduce the number of transformations that the XML Engine is required to process centrally by moving what transformations are required to the Access Point. Access Points in this second scenario are used to keep most messages traversing the platform in standardized form that is readily processed by the platform services.

5.3.4 Message Warehouse

Many applications have transactions that take place over a period of time measured in hours, days, weeks, and sometimes months. In addition to maintaining the state of pending transactions, the Message Warehouse allows such an application to be "put to sleep" (temporarily stopped or shutdown – hibernate) by allowing all the pending messages (transactions) to be stored and retrieved. The context of a Message Warehouse is absolutely critical – a Message Warehouse is a transaction oriented persistence mechanism and not a mechanism for just storing messages in a file. In other words, a Message Warehouse should not be used as a message persistence mechanism for long-time archival storage – the size of a Message Warehouse can quickly become a serious problem within a large-scale enterprise platform.

The ability to support transaction persistence is typically overlooked in enterprise architectures (the only enterprise platform suite to currently include a Message Warehouse is BEA WebLogic – but the current implementation is still primitive). There are several third-party Message Warehousing solutions available on the market but each one requires careful evaluation – especially in large-scale enterprise platform architectures since a number of these products have scalability issues.

Important characteristics of a Message Warehouse include:

- Provides storage and management of messages used in long-duration application transactions in a pending or suspended state.
- Provides store-and-forward communication of a message.
- Holds the message in a queue until the recipient formally accepts it.
- Assures that the message is recoverable and is received once and only once.
- Messages are managed in the context of a relational or object database framework (use of persistent queues is not a good idea since performance can be severely impacted).
- Internally organized according to transaction workspaces.
- Access via configuration tools, EJB's, API's, and workflow element interfaces.

5.3.5 Information Pipe (Enterprise Messaging Services and Message Queuing Services)

Message Services and Message Queuing Services are represented by the Information Pipe in the Platform Independent Model. There are four architectural patterns for enabling communication across the enterprise and across the Internet (not discussed here for brevity).

There are two basic architectural approaches to queuing: IBM WebSphere, Microsoft .Net, and Open Source platforms use a distributed message queue architecture that allows the queue to communicate through different network security zones. Oracle, on the other hand, relies on its database product for the queue – this means that a connection to the Oracle Advanced Queue database server is required to use the Message Queuing Services [20]. The Oracle approach can be a disadvantage in a distributed enterprise architecture since workarounds must be provided to connect to different queues within the platform.

From an implementation standpoint, IBM WebSphere and Microsoft .Net work pretty much the same. Both queue products can be easily load balanced using queue server clusters that use a "round-robin" mechanism for distributing messages. The Open Source queue available for JBoss is also very similar, but has not reached a high level of maturity when compared to the IBM and Microsoft products at the time this document was written. Two other popular queuing products are also worth mentioning: Tibco and WebMethods. Both queuing products have a large amount of domain specific components available – especially in the banking, finance, and healthcare industry sectors, which may save an organization time and money; however, both products are more complex and typically more expensive to develop and maintain when compared to the IBM, Microsoft, and Open Source products.

One important issue to keep in mind regarding developing a Message Queuing Service is that queuing clusters can only be built on an enterprise server operating system – server queuing cannot be installed on stand-alone and workstation operating systems. This means that all server-side queuing code must be developed on a server.

5.3.5.1 Publish and Subscribe Services

Publish and Subscribe is a mechanism for transmitting information to multiple consumers (a consumer is defined as a person or another system) while avoiding the pitfalls of point-to-point information transmission. The process of publishing information to many subscribers is commonly referred to as fan-out. Each "consumer" registers to receive information when it is published. The Publish and Subscribe Services manages subscription lists and facilitates information distribution.

Web Services within and between enterprises rely on many technologies to transport information over the Internet. Examples of these technologies include HTTP/HTTPS for data transport and XML for data representation. Early technologies used for information transport included the Common Object Request Broker Architecture (CORBA), Remote Method Invocation (RMI) common with Java Application Server environments, and Distributed Common Object Model (DCOM) in Microsoft .Net environments. More recently, with the emergence of Web Services, there is an increasing use of Remote Procedure Calls (RPC's) and Simple Object Access Protocol (SOAP). Publish and Subscribe Services manages the various information transport mechanisms in the platform providing greater modularity and robustness.

5.3.5.2 Connector SDK

The Connector Software Development Kit (SDK) provides support for specific Access Point and legacy system connector development. The SDK consists of access point and connector implementation frameworks often using multiple language libraries for development. The Connector Frameworks support connectivity via the transport layer or via RMI calls from application servers. Once developed, connector & access point interfaces are registered in the repository as workflow elements.

The following steps must be supported when building a Component Service Interface:

- Read the message from the Information Pipe (or bus)
- Translate the message into a format that is compatible with the internal structure of the API call (referred to as marshalling)
- Execute the function call
- Marshall the results of the function call into an XML message
- Send the resulting XML message to the designated recipient

5.4 Business Integration Services

Business Integration Services provides the integration of the business logic with the data and applications both within the enterprise and with partners. Much of the business integration relies on workflows that are used to tie together different applications, ERP systems, and platform services to meet the strategic objectives of the business. Like Application Integration, Business Integration is made up a several services: Basic Integration Services, Workflow Management Services, Security Services, Partner Profile Services, Content Management Services, Decision Support Services, and Application Services. These services provide the integration of the business logic with the data and applications.

5.4.1 Basic Integration Services

The Basic Integration Services Component uses Message Brokering to provide a framework for development and configuration of interfaces to back end applications to promote reuse. The Basic Integration Services enables an enterprise platform to provide a service-oriented architecture based on common standards including:

- Open transport protocols such as XML, UDDI, and SOAP
- Standard frameworks such as J2EE and Microsoft .Net
- Common interface specifications
- Web Services

5.4.2 Workflow Management Services

Workflow Management Services provides the framework for the definition and execution of application workflows within the EAI application space (inter-application orientation, not user oriented). It is data driven and supports business process models and/or transactional workflows with business logic and rules execution. It can be triggered from application services, from events, or from queue triggers. It uses component interfaces to legacy systems as logical workflow elements defined within the repository. It can support long duration workflows, transaction sets and chained workflows.

5.4.3 Security Services

The Security Services component supports Role-Based Access Control (RBAC), and Single Sign-On (SSO). Security Services also supports digital signatures, encryption, and decryption functions on messages and information, and key management. It is callable as part of the Workflow Management Services through API's or EJB's within applications. It uses Directory Services for PKI support, if required. Security Services is also responsible for all security logging on the enterprise platform during transactions.

5.4.3.1 Security Interceptor

This Security Interceptor Service intercepts and applies a Security Context for Role Based Access Control (RBAC). A security interceptor is used to extend the traditional RBAC model and is typically applied at the programming language level as part of a library.

5.4.4 Partner Profile Services

Partner Profile Services manages the repository information on resources, services, and partners that interact with the platform. The repository is typically XML based but can also be product based. An emerging open standard is ebXML, which is a dedicated XML specification for maintaining information trading partner profiles. Partner Profile Services defines which transport protocols are available and how information is sent and received, which protocols are used in a given instance, defines data structures and their definitions, and keeps track of business information that the enterprise needs to know about its partners.

Functional requirements that are supported by Partner Profile Services include:

- Use of XML, XPath, Schemas and DTD's to classify resources and trading partner information
- Manage common business objects and their transformation mappings with partner systems
- Manage and define business processes
- Manage and make available information about the enterprise resources available to partners (information the partners need to know about the enterprise)
- Exposes interfaces to partners expressed as business processes

5.4.4.1ERP Services

Enterprise Resource Planning (ERP) Services consists of Partner Profile information that is based on an ERP vendor standard. Examples include SAP and PeopleSoft.

5.4.4.2EDI Services

Electronic Data Interchange (EDI) Services consist of Partner Profile information that is based on EDI standards, which are typically open standards developed within the respective industry segments. EDI Services offer significant advantages over Web Services standards due to the lower overhead of transmitting the information (EDI messages are more compact and do not require as much transformation processing). An example of an EDI standard is EDIFACT.

5.4.5 Content Management Services

Content Management Services (also referred to as Content Services) supports the functionality for the integration of static and dynamic web content management and delivery systems based on applications in the EAI space. It also provides interfaces to existing Content and Knowledge Management environments where available (i.e. Web Content Management, Document Management, Knowledge Management, and Business Intelligence). Content Management Services and Integration Services are the two most critical components on any Enterprise Portal Platform. The diagram below; Figure 12, illustrates the basic functionality that Content Management Services must support.

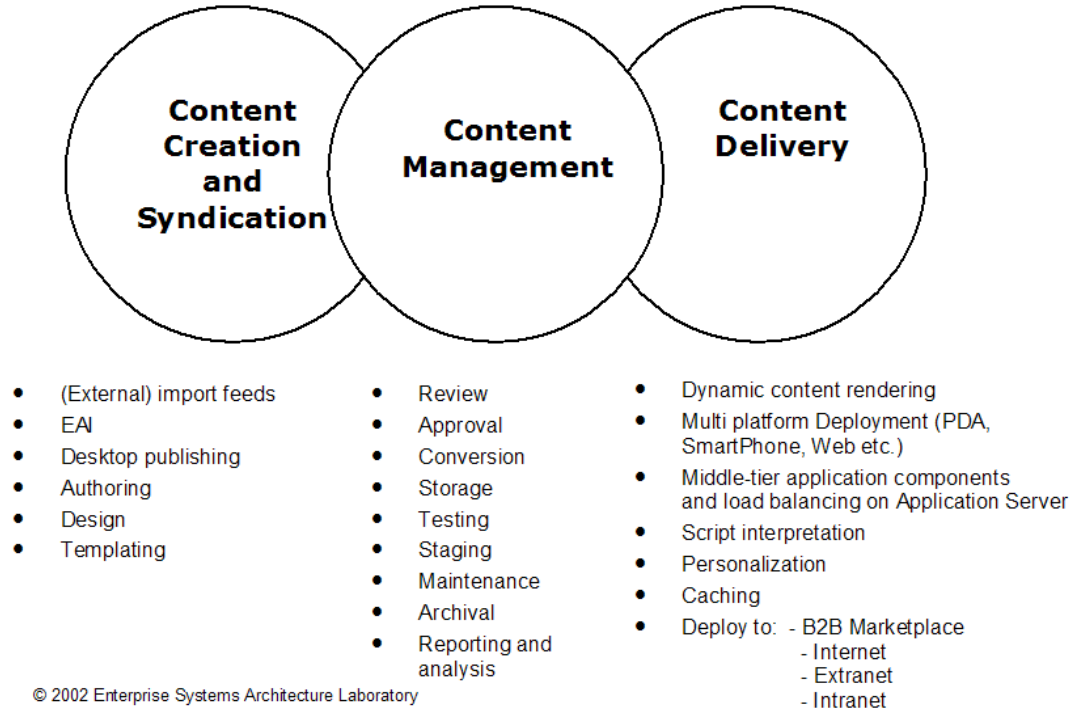


Figure 12: The basic functionality that Content Management Services must support.

Content Management Services includes support functionality for the integration of dynamic web content management and delivery systems based on applications in the EAI space. Two types of specialized components are required: First, dedicated content, knowledge, and document management applications are needed. Second, a metadata taxonomy needs to be developed to properly categorize and manage the content within the enterprise, which usually requires specialist tools and experience (most organizations do not have adequate experience in developing taxonomies and need to rely on external experts). Finally, the enterprise needs to be able to support content publishing workflows that include authoring, review, approval, and publishing steps. Additionally, versioning and archiving functionality must be provided to meet the legal and regulatory requirements of the business.

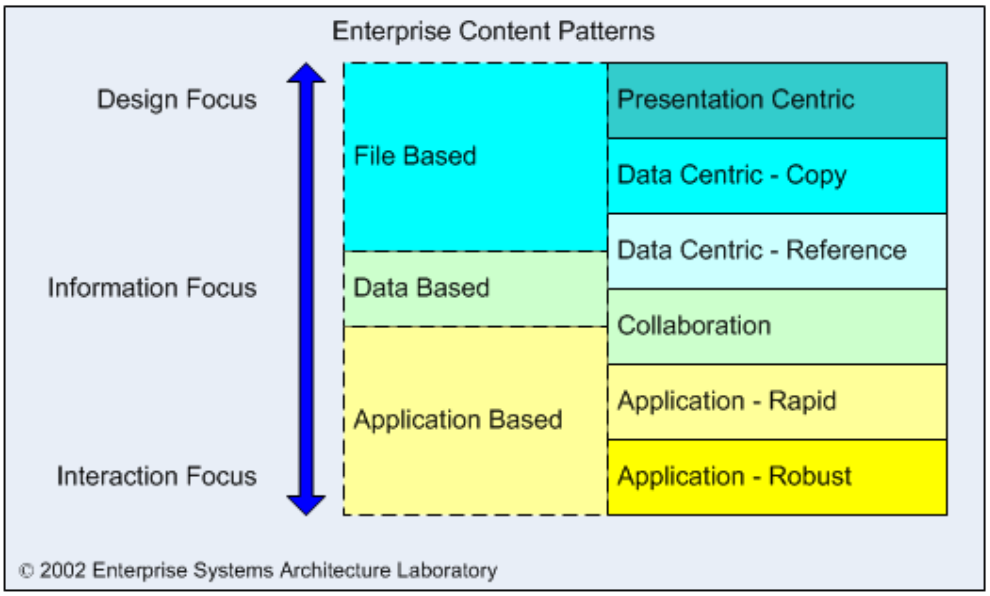


Figure 13: Six Enterprise Content Patterns that categorize all information managed within enterprise platforms and enterprise portal platforms [2]

The diagram above, Figure 13, illustrates the six Enterprise Content Patterns that categorize all information managed within enterprise platforms and enterprise portal platforms [2] Characteristics of these six design patterns are:

- **Presentation Centric:** Information is displayed in one-off pages where the presentation is as important as any content. The presentation may over-ride and standardized templates or it may form the basis for new templates.

- **Data Centric - Copy:** Content is placed against a predefined template that handles persistent navigation and brand messages. Files are copied into the web space from the client information space and no links are recorded.

- **Data Centric - Reference:** Content is placed against a predefined template that handles persistent navigation and brand messages. Application, script, and web programming languages or scripts are used to point to the content in the client information space. Updates to the content in the client information space are automatically propagated to the web without re-editing the page.

- **Collaboration:** Service where users create and share information in a collaborative manner. Pages use templates for format and brand/design elements.

- **Application - Rapid:** Service based on an application that is often needed quickly and has a short-planned life. Implementation relies on code generation tools working against a database and using platform components and services.

- **Application - Robust:** Service based on high-availability, transactional applications. The requirement for robust platform functions and components typically results in the use of high-level programming languages such as Java, C#, C++, and Visual Basic.

The six patterns can be organized into a three-tier focus: Design, Information, and Interaction. The focus indicates the overall involvement of more complex platform integration services as opposed to simple file management services.

5.4.5.1ECM Metadata Repository

The Enterprise Content Management (ECM) Metadata Repository provides a unified functional interface and set of databases for metadata management in the ECM server environment. The interface functionality is provided via an object server and a programmable shell, coupled with a GUI toolset. The metadata is managed in the following collections:

- **Data Extraction Formats** contain the data extraction specifications for back-end systems, the structures of resultant datasets, and the transformation format mappings to XML/Schema, HTML, etc. It also contains mapping specifications for transformation tools.
- **Source Descriptors** contain the metadata describing back-end databases for query and extraction specifications.
- **Metatags** store specifications of tags and tag collections for document attributes used in the formatting of HTML and XML documents.
- **Application System Profiles** describe the ECM resources used by external application systems. This acts as a partition space to allow "customized" delivery and accounting for ECM services.

5.4.6 Decision Support Services

Decision Support Services provides functionality for integration and development of data warehousing and data-mart (a database or collection of databases with a small focus) business reporting based on applications in the EAI space. It is possible to use these functions for data feeds to existing data warehouse systems using request/response or publish/subscribe messaging. It also provides interfaces to existing Decision Support environments where available (such as Oracle Express and Cognos Essbase).

5.4.7 Application Services

The Application Services component provides application servers for J2EE based systems. It also provides transaction management and recovery for applications in the EAI space. It monitors all aspects of application execution across platforms including availability and performance monitoring and provides exception handling, message routing, and logging. It is also used for the development of interfaces to application servers and EAI configuration environment. Common Application Servers for J2EE-based systems include IBM WebSphere, BEA WebLogic, Oracle 9iAS and 10gAS, and Open Source solutions such as JBoss.

5.5 Integration Services Proxy

The Integration Services Proxy provides business directory management and an information model for the enterprise. The Proxy Services are obtained from the high-level Integration Services component that represents a centralized approach to application integration with the goal of supporting reuse by establishing a commonality on interface development and management. The basic functionality is:

- Manages both information models and catalogue interfaces to information systems and services
- Uses XML for all internal representations and schemas
- Uses UML for development and documentation
- Can be accessed on a read-only basis by legacy applications via access points
- Uses Integration Services for maintenance and update infrastructure for EAI

Tactical, ad-hoc approaches to application integration will ultimately lead to an unmanageable and expensive point-to-point connectivity solution that is sensitive to changes in the underlying infrastructure and the connected applications. A coordinated Integration Services strategy is required to maintain a robust (resilient to change) platform architecture.

Reusability and manageability in the platform is achieved using a central interface repository where information on the technical details of interfaces is kept. Given the heterogeneous nature of applications within and external to the enterprise and the frequent lack of common syntactical and semantical views of the information that needs to be exchanged between applications, there is a critical need for powerful data transformation and data routing capabilities. Message and Integration Brokers provide some of these capabilities and XML will also help solve some of the problems related to standardization and content validation.

5.5.1 XML Services

XML Services provide extensions to transformation services particularly from the XML message structures supported by 3rd parties. All business objects are defined in terms of the domain using XML.

Despite the recent hype, XML is only a representation notation. XML is a key enabler for B2B integration and is an enabler for EAI as more applications (e.g. ERP solutions are "XML enabled"). Currently, XML does not resolve the semantic differences between information embedded in an XML document and the diverse set of interface formats associates with the core business systems. This area requires careful investigation and attention.

5.5.2 B2B Services

The B2B services component is where the development and maintenance of interfaces to external systems (suppliers, customers, and partners) takes place. It holds the definitions of application message protocols between enterprises. It is also a framework for development of "Virtual Enterprise" workflows. It uses UDDI (Universal Descriptor Discovery and Integration) Protocol and SOAP (Simple Object Access Protocol) technologies for interfaces. It can also be used in conjunction with Authentication Services.

5.6 Authentication Services

Authentication Services provides a role-based security access model across the enterprise. Access to services is governed by roles, which specify what resources are either accessible or denied. A deny access takes precedence over any access privileges. Each platform client (defined as a person, system, or service that has permissions on the platform) is assigned one or more roles. Each role is assigned access privilege rights and in some cases denials that are specific to services and resources managed by the platform. Additionally, trust relationships can be defined with partner systems. Authentication Services interacts with Security Services to manage client access on the platform. Managing roles is easier than managing individual users and systems.

Functional characteristics of Authentication Services include:

- Toolkit for the development and testing of partner profiles (see Access Point) and security policies for applications, transactions and queues
- Maintains a collection of channel definitions for each partner/role in the application space (partner profile)
- Provides interfaces to Single Sign-On (SSO) facilities and global index management (e.g. e*Index)
- Provides links to authentication services on external application hosts (e.g. RACF, Tivoli Director)

5.6.1 RBAC Repository

The Role-Based Access Control (RBAC) Repository is typically implemented as an LDAP repository. Microsoft .Net utilizes Microsoft Directory Services (Active Directory). [21]

5.6.2 Security Integration Component

The Security Integration Component enables web and gateway security services at the platform boundaries (e.g. IBM WebSEAL).

5.6.3 Single Sign-On Component (SSO)

The Single Sign-On Component is a dedicated application that manages RBAC and Platform Resource associations (e.g. Entrust).

5.7 Gateway Services

Gateway Services enables secure communication through firewalls for Internet or other externally enabled applications with the enterprise platform. It also provides a development framework for gateways and portals based on XML and other messaging technologies to provide links to other EAI services via the transport layer. Gateway Services is also responsible for supporting session management-based interfaces to Application Servers.

5.8 Access Point

The Access Point component is a framework (architecture pattern) used to provide a minimally-invasive connector for third-party applications by providing a consistent interface to the Information Pipe (or information bus) and the integration technology used within the platform. Access Points are packaged as a component server. An Access Point can support multiple protocols for inbound and outbound messages by exposing functionality of the underlying applications as services via the Access Point Component.

5.8.1 Access Point Connector

A conceptual model of the Access Point Connector is illustrated below, see Figure 14:

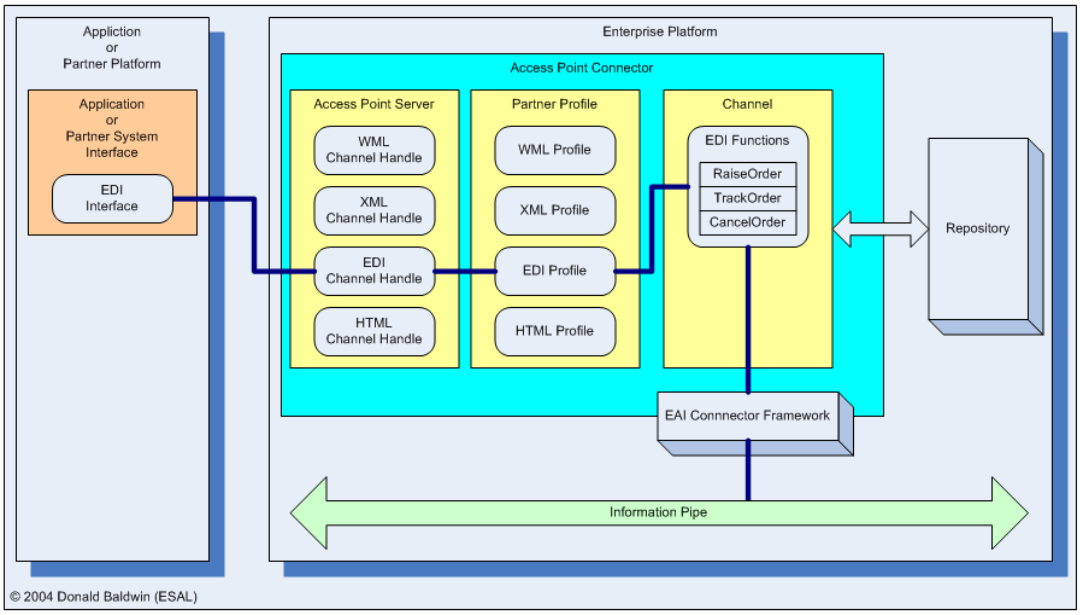


Figure 14: A conceptual model of the Access Point Connector

The Access Point Connector illustrated above contains four interface specifications: WML, XML, EDI, and HTTP. The EDI interface specification supports four function calls: RaiseOrder, TrackOrder, and CancelOrder. An Access Point Connector can be used for applications located within the enterprise platform environment or external to the enterprise such as partner applications residing on another network zone.

An Enterprise Service Bus (ESB) is one example of a realization of the Access Point Connector framework (architecture pattern).

5.9 Management Services

Management Services provides a "business view" perspective and is concerned with availability, failover, performance monitoring, and end-to-end interface management across the enterprise. Management Services support the day-to-day operation of the enterprise platform.

5.9.1 Transaction Performance Manager

The Transaction Performance Manager is required to realistically monitor and manage an Application Server or similar environment where heavy transaction processing takes place. Unfortunately, current application servers have no real Transaction Performance Manager and the support that is present for monitoring transaction performance is typically inadequate. This requires a third party or custom-built solution. Some products such as Microsoft's .Net Framework and IBM WebSphere provide API's that can be used in the Transaction Performance Manager role, but such use relies heavily on the experience of the development team – experience that may not be adequate for the needs of a complex enterprise platform. There can also be serious performance issues with many common approaches to monitoring the performance of transactions – especially in clustered environments.

5.9.2 Transaction Processor

The Transaction Processor is responsible for supporting application logic between the platform clients (humans, external partner systems, and platform services) and the resource managers (such as platform services, data-stores, message queues, and legacy systems). Examples of common Transaction Processor Application categories include request/response, conversation, queuing, and publish/subscribe. Many transactions are built around workflows.

There are many significant performance issues involved with Transaction Processors. The lack of adequate Transaction Performance Managers further compounds this problem. There are currently no solid Transaction Performance Manager solutions available for large-scale enterprise platforms. This places a heavy demand on the enterprise architects and systems engineers in designing and configuring the enterprise infrastructure environment – this is also why the Transaction Processor and Transaction Performance Manager pose some of the greatest risk issues towards the implementation of an enterprise platform (especially in larger scale implementation).

5.9.3 Transaction Process Monitor

The Transaction Process Monitor coordinates the flow of transaction requests between clients and Transaction Processors that can process them.

5.9.4 Logging Services

Logging Services provides a centralized mechanism for logging platform client access to specific resources at a granular level. In many industries, there are legal and regulatory issues that must be complied with. Additionally, Logging Services provides a central clearing house for infrastructure logging as well. Overall support for centralized logging can vary between vendors. For instance, Microsoft provides a Performance Monitor API, which requires the development team to build the performance logging functionality. IBM and Oracle provide more logging support, but both products have serious performance issues [41] that impact the entire platform when logging is enabled. In many cases, third-party or custom-built logging solutions are often required to supplement the existing logging functionality provided by the product.

5.10 Enterprise Development Environment

Development and integration work on complex enterprise platforms requires a robust set of integrated tools. Most development environments are based around several sets of tool suites:

- UML analysis and design modeling suites (Examples include IBM Rational Suite and Sparx Systems Enterprise Architect)
- Integrated Development Environments (IDE's) such as Eclipse, Microsoft Visual Studio .Net, and Oracle JDeveloper

5.10.1 Developer IDE

Major vendors (including IBM, Oracle, and Microsoft) as well as the Open Source community provide (or maintain in the case of Open Source) specialized tools that are tailored to their specific environment. Using these tools makes it easier for programmers and other developers to build code and database solutions. Many IDE tools can provide stub frameworks and in some cases automatically generate some level of code. Additionally, there are many tools designed to automatically build databases based on specifications that the developer enters into the tool.

5.10.1.1 CASE Tools

Computer Aided Software Engineering (CASE) Tools provide a more robust environment than the traditional IDE development tools. Characteristics of a CASE Tool are support for requirements, developing workflows, scheduling development tasks, and automatic code generation. Many CASE Tool suites will often have basic support for configuration management as well. A CASE Tool is valuable since it provides a project-based management system for much of the documentation required for development and maintenance.

5.10.2 Version Control Mechanism

Enterprise Platforms require very robust Version Control Systems because of the complexity and number of people involved in the development effort. Most platforms are organized within a Program Office with major components being assigned to Project Teams. Many Enterprise Platform Programs will last anywhere from two to five years from the initial concept until the platform is deployed. It usually takes an organization two to three years before real benefits are realized. During the four to eight year life-cycle, a tremendous amount of documentation and development artifacts are typically generated. Examples of commercial Version Control and Configuration Management tools include IBM Rational Clear Case and Microsoft Source Safe. A common Open Source tool is CVS (Concurrent Versions System). Specialist tool smiths are required to properly configure and maintain Version Control and Configuration Management systems.

5.10.3 Deployment Mechanism

The Deployment Mechanism interacts with the Version Control and Configuration Management development environment components to deploy builds into the test and production environments from the development environment. The Deployment Mechanism should support the ability to deploy roll-back builds (a roll-back build is an older build that is used to replace a newer build – usually when there is a problem discovered after a new build has been deployed into the production environment).

6 Case Studies

To demonstrate the application of the artifact presented in this thesis, DNEAF for EAIS and PPS, three case studies are presented in this Section (Note that parts of this section have been previously published [41]):

- **E-Health Platform for Telemedicine** – illustrates the core platform components for an e-Health platform based on the DNEAF for EAIS and PPS (Shows an example of an architecture driven approach to analysis and design)
- **Banking** – illustrates an example of Transitional Gap Analysis based on the DNEAF for EAIS and PPS (Shows an impact analysis between the initial baseline and the target baseline of a high volume transactional service based on mapping legacy systems)
- **Vendor Product Evaluation and Comparative Analysis** – illustrates an example of Comparative Gap Analysis based on the DNEAF for EAIS and PPS (Shows the functional equivalent mappings between three commercial platforms and a selection of Open Source products)

6.1 Case Study 1: E-Health Platform for Telemedicine

This case study is based on a conceptual architecture document prepared by Chris de Vaney, Chief Architect, ESAL and the Aurenav Research Institute in Stockholm, Sweden dated 2004 February 24.

6.1.1 Executive Summary

The Karolinska University Hospital (KUH) is confronted with the challenge of providing quality patient care in an environment of budgetary constraints and competing demands for limited resources. Telemedicine is playing an increasingly important role in maintaining patient quality of care while reducing costs. Telemedicine solutions have typically been implemented by building dedicated application solutions that conform to a stovepipe architecture. In 2003, KUH initiated a program within the Department of Biomedical Engineering (DBE – "MTA" in Swedish) to build a telemedicine platform (TeleMed HC) that can be used in conjunction with biomedical devices (BMD's) located both within the hospital and within patient homes. Existing experience within the DBE has demonstrated that certain categories of patient care can be provided at lower cost in a patient's home compared with providing treatment in a clinic or managed healthcare environment.

The Karolinska University Hospital Huddinge has a wide range of medical applications and networks which currently provide the operational infrastructure of the medical and administration activities. These applications are in a state of transition, with the Hospital placing significant investment in the renewal of the infrastructure and improvement in connectivity and information exchange between the applications.

In addition to these obligations, the Hospital has foreseen the need to provide for a common messaging interchange strategy for current and future medical information systems, based on internationally accepted standards. The utility of this is to provide a bridge for harmonizing the information exchange and integration of applications in a vendor-neutral format.

The actions described above have been combined into the framework of the eHealth Telemedicine initiative, which aims to provide both the EAI platform and an extensible body of application frameworks for medical, administrative and research systems.

To meet the needs of the hospital, the Telemedicine eHealth platform has been defined as an operational runtime support infrastructure for distributed medical applications. The infrastructure is geared for the development of applications based on domain-specific frameworks consisting of components, connectors and other configuration elements.

6.1.2 Platform

The objective of the platform is to provide an environment in which to execute transactions using functionality from many different systems. The transactions are defined in workflows, which execute processes within a logical framework with each workflow communicating by passing messages between applications.

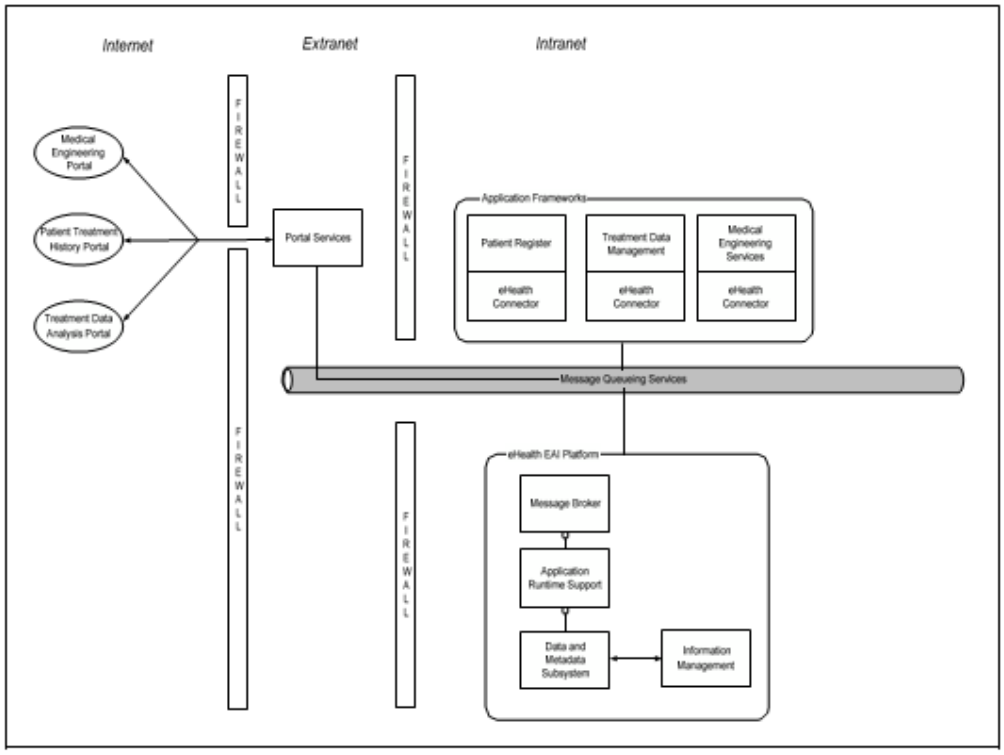


Figure 15: DNEAF for EAIS and PPS mappings for eHealth

The eHealth platform operates on the principles of:

- Message-Oriented Middleware
- Loosely coupled systems that provide services to the enterprise
- Workflow Transactions

6.1.2.1Message-Oriented Middleware (MOM)

The eHealth platform operates on the principle of a message bus which enables communication between application systems by placing messages on queues and retrieving them. This has the advantage of a single, persistent communications model which is scalable and secure. It also reduces the complexity of application development by reducing the dependence on synchronous application connectivity.

6.1.2.2Loosely Coupled Systems

The eHealth platform uses a framework for loosely-coupled systems which sponsors re-use of functionality and consistency of interface between systems. In this model, applications expose their functionality to other applications via a connector module which communicates with the eHealth application services. The connector enables the interfaces to the functions to be described, and to be available to other applications as workflow operations. The connector also manages communication for the application by initiating the necessary translations of messages from other systems – translations either of format or of content.

6.1.2.3Workflow Transactions

The eHealth platform's most salient feature is the strong support for workflow-based applications, using business logic contained within components rather than in hard-coded application programs. This has two distinct advantages:

- Applications can, in principle, be developed very quickly by adapting workflows and configurations rather than coding to a "traditional" model. In practice, the only coding needed for a workflow-based application is for the components (i.e. Java beans) implementing the distinct services, and any connector software needed for access to 3rd-party systems;
- Workflows can be re-used to create sequences of end-to-end transactions, allowing the participation of several systems in a new application service without the need to re-develop code.

6.1.2.4Subsystems

The system shown in Figure 1 is composed of a number of subsystems, each of which is a configuration of components providing well-defined services. The subsystems are:

- Message Queuing Services
- Message Broker
- Data and Metadata Subsystem
- Application Runtime Support
- Portal Services

6.1.2.5Message Queuing Services

The message queuing system manages the physical transfer of XML documents as messages between participating systems. The MQ system is configured on a hub architecture with failover provision and monitoring enabled.

6.1.2.6Message Broker

The message broker provides the services for inter- and intra-application messaging support in the context of the eHealth platform. The broker service is configured as follows:

Table 5: Message Broker Components

Component	Description
Message Queuing Services	Provides the base level support for message queue operation in the system, namely queue connection, putting messages on queues and retrieving them. This component also sets the queue persistence levels and error recovery protocols. In the eHealth platform, all messages are assumed to be in XML format. It should be noted that a Publish and Subscribe (Pub/Sub) mechanism is implicit in these services.
Message Transformation Services	Provides for the transformation of messages from one format to another. For example, an EDIFACT message can be converted to XML format on receipt from another system. Transformation services also include: - Extraction of message subsets - Merging of messaging - Application of business rules to messages
Rule Processing Services	Provides support for applying rules to messages and determining actions to be taken. Rules may be triggered based on message types, contents or destination information. Rules may directly trigger transformations, routing changes or any similar combination.
Intelligent Routing Engine	Provides for routing of messages to applications based on the application of rules or routing information in the message header. If a Pub/Sub service is present, this will be the main publication engine. (Publishing messages to multiple subscribers).

The message broker is dependent on the Message Queuing Services for operation, and on the Data and Metadata subsystem for message and metadata support.

6.1.2.7Data and Metadata Subsystem

The data and metadata subsystem manages the repository, directory, operational data interfaces, logging and transaction support in the scope of the eHealth platform. The subsystem consists of a number of base components and the information management domain of the platform.

The information domain consists of the following databases and object stores:

Table 6: Databases and Object Stores

Database / Object Store	Description
EAI Application Repository	Contains the metadata for all applications and information systems operating within, or interfacing with the eHealth platform. Each application is basically defined in terms of its workflows, messages, operations and exceptions in XML documents stored within the repository. The repository also stores interface definitions to services exposed in external systems via the eHealth connector architecture.
Message Dictionary	Contains the HL-7 message definitions and structural mapping definitions for the eHealth platform.
Message Warehouse	Contains the message queue context (content, state & routing information) for suspended or long-term transactions on the eHealth platform.
Operational Logs	Contain the transactional or other logs defined for the eHealth platform.
LDAP Directory	Contains the LDAP directory (schema and content) for the eHealth platform. All actors, roles, services, interfaces and permissions are described in this directory. This may exist as a logical view over an existing enterprise directory, if appropriate.
Master Data Repository	Contains the definitions of data elements and objects which are global to the eHealth platform. The definitions may either be the actual data, or an interface to the data at source.

The data and metadata subsystem consists of the following components:

Table 7: Data and Metadata Subsystem Components

Component	Description
Repository Services	Provides the services for management and runtime use of the EAI repository in the eHealth platform. Also provides the interfaces for the HL-7 message dictionary and the LDAP directory.
Logging Services	Provides a consistent interface and services for operational logging by applications and workflows on the eHealth platform. The logs will support all levels of transaction processing on the platform.
Data Services	Provides access to the Master Data Repository, and to external systems providing operational or other data to eHealth platform applications. Also provides the transactional save/restore interface for applications using the Message Warehouse functionality.

6.1.2.8Application Runtime Support

The application runtime support subsystem manages the operational aspects of applications running within the eHealth platform. The runtime support components, as a whole, comprise an application server environment.

The runtime support subsystem consists of the following components:

Table 8: Runtime Support Subsystem Components

Component	Description
Transaction Management Services	Provides the framework for the invocation, monitoring and exception handling for application transactions within the scope of the eHealth platform. The services will drive the workflow and J2EE application servers in the runtime environment.
Workflow Engine	Provides for the execution of workflows in the context of the eHealth platform. A workflow is a set of operations which are executed within a flow of control defined in the workflow. The operations are either local to the workflow or invoked via a connector interface in an external system.
J2EE Application Server	Provides support for running applications developed as J2EE/Java beans, with external interfaces. These are used on the eHealth platform to develop components, accessible from workflows, web services and other applications.

The application runtime support subsystem is dependent on the Message Broker for operation, and on the Data and Metadata subsystem for information management support.

6.1.2.9Portal Services

The portal services subsystem provides the access management and web-centric functionality of the eHealth platform. The subsystem provides a gateway for use by external users or systems to either run portals front-ending eHealth applications, or to connect to web services encapsulating particular functionality.

The portal services subsystem consists of the following components:

Table 9: Portal Services Subsystem Components

Component	Description
Portal Server	Provides facilities for the development and execution of portlets and servlets, supporting web-enabled access to applications or services running on the eHealth platform.
Web Services	Provides facilities for interfacing functionality available within the eHealth platform to other applications via a request/response protocol such as SOAP.
Access Management Services	Provides authentication services for applications within a single sign-on framework and using role-based access as a baseline security paradigm.

It should be noted that the portal services components communicate with the other eHealth subsystems and components only via the message queuing system. RPC and other connection paradigms are not supported within the eHealth framework due to security and quality of service issues.

6.1.2.10 Implementation Approach Options

The proposed implementation approach for the architecture is to develop the platform configuration around commercial off-the-shelf (COTS) components, with tailored developments limited to areas which need to be covered and for which COTS elements are either not available or non-viable.

The following COTS packages are known to map to functional components in the eHealth architecture. It should be noted that there are areas of overlapping functionality which would need to be addressed in the platform configuration architecture.

Table 10: COTS Mappings in eHealth Architecture

Package	Component / Subsystems Covered
IBM MQ for WebSphere	Message queuing services
Candle CASP	Rule Processing, Intelligent Routing, Workflow Engine (restricted)
Oracle 9iAS	Data and Metadata Subsystem, Workflow Engine, Message queuing services.
IBM WebSphere Application Server	J2EE Application services, Transaction management services (projected), Web Services
IBM WebSphere Portal Server	Portal Server, Access Management Services (Limited)
DataJunction	Message Transformation, Rule Processing

The main architecture components requiring development effort are the data stores and interfaces for Repository and Logging services, and the SDK for connectors to external systems. In the latter case, the existing SDK's provided for the COTS packages can probably be extended.

There are many possibilities to develop the platform using Open Source tools in place of COTS packages. The following mappings would effectively create an open- source version of the platform, assuming that both IBM's MQ for WebSphere and Oracle were available:

Table 11: Open Source Alternative to COTS

Package	Component / Subsystems Covered
IBM MQ for WebSphere	Message queuing services
Oracle 9iAS	Data and Metadata Subsystem
jBoss	J2EE Application Services, Transaction management services
jSimpleX	Message Transformation
JMS	Connector interface to IBM MQ for WebSphere
OpenLDAP	Directory services
Taverna	Workflow Engine
Zope	Portal Services, Web Services, Content Management
DataVision	Universal Database Connector

In both the above cases, it will be necessary to develop the repository databases and interfaces to support the platform.

6.1.2.11 Platform Considerations

The optimum production environment for the platform is a cluster of Linux servers, with each server running one of the platform subsystems. In addition, there should be a configuration baseline which allows hot-swapping and failovers between the servers.

The use of Linux clusters is mandated by the need to provide a distributed environment that can encompass one or more instances of the platform running on the network, thus enabling greater scalability and connectivity for future developments.

6.2 Case Study 2: Banking - Transitional Gap Analysis

A Transitional Gap Analysis begins with the DNEAF for EAIS and PPS. Each PIM component serves as a proxy and is mapped to a PSM that provides equivalent functionality within the vendor's product. If there is no equivalent functionality, the mapping is either left blank or a comment can be made. Comments can either indicate that the functionality is not supported in the product or alternatives can be suggested as a mechanism for gap closure.

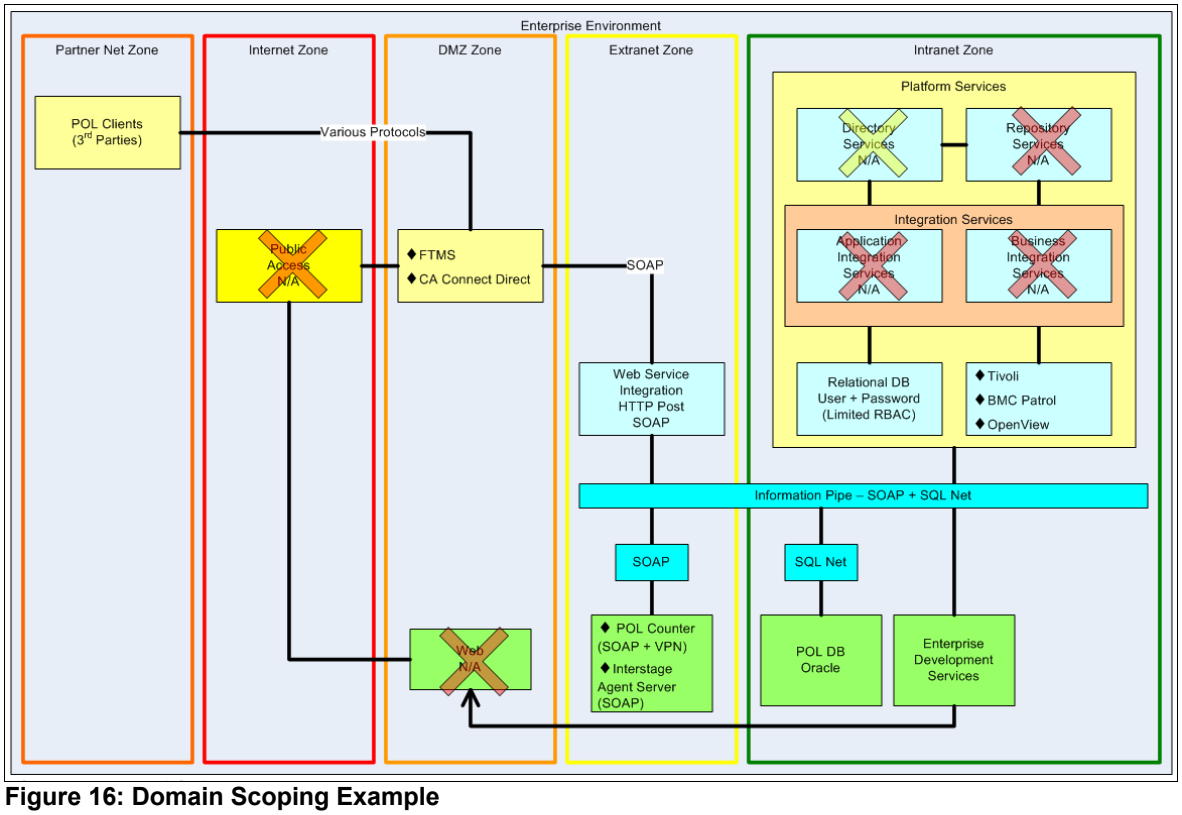


Figure 16: Domain Scoping Example

The illustration above, Figure 16, shows the Proxy mappings for a banking system – the domain neutral labels have been replaced with the specific product names that fulfill the equivalent functionality. The model above represents the largest banking system in the United Kingdom in terms of transaction volumes and as a percentage of transactions rated against the United

Kingdom's Gross Domestic Product (over 25% of GDP). In the diagram above, a red "X" indicates that functionality is not planned and a yellow "X" indicates that the functionality is planned for later.

The next step is to map the systems identified in the model into components and packages in a standard UML model as indicated below:

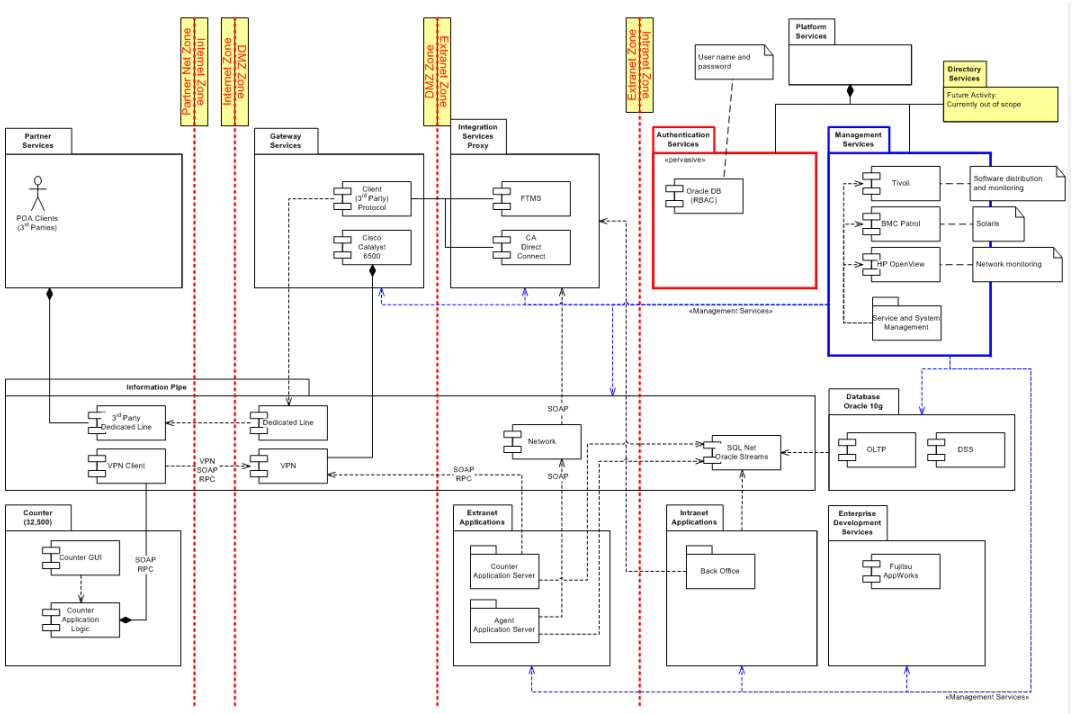


Figure 17: Transition to UML High-Level Model

What is important in this model is that each package provides a direct mapping to the previous model and this preserves traceability. This provides a mechanism for conducting a more detailed analysis using the UML Communication Model in a special role called a Robustness Analysis Model as illustrated below in Figure 18:

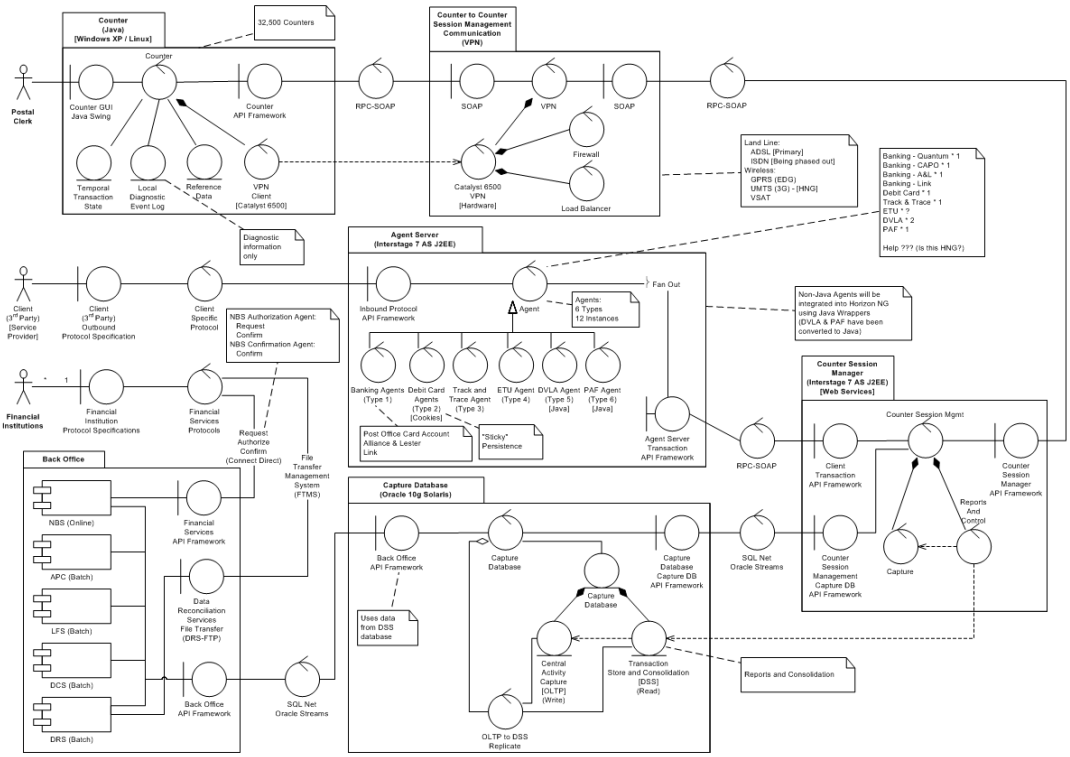


Figure 18: Robustness Analysis Model (A special type of network communication model in UML)

The resulting Robustness Analysis Model provides a high-level view of how systems and subsystems connect with each other to represent an Initial Baseline (e.g. what the system looks like today).

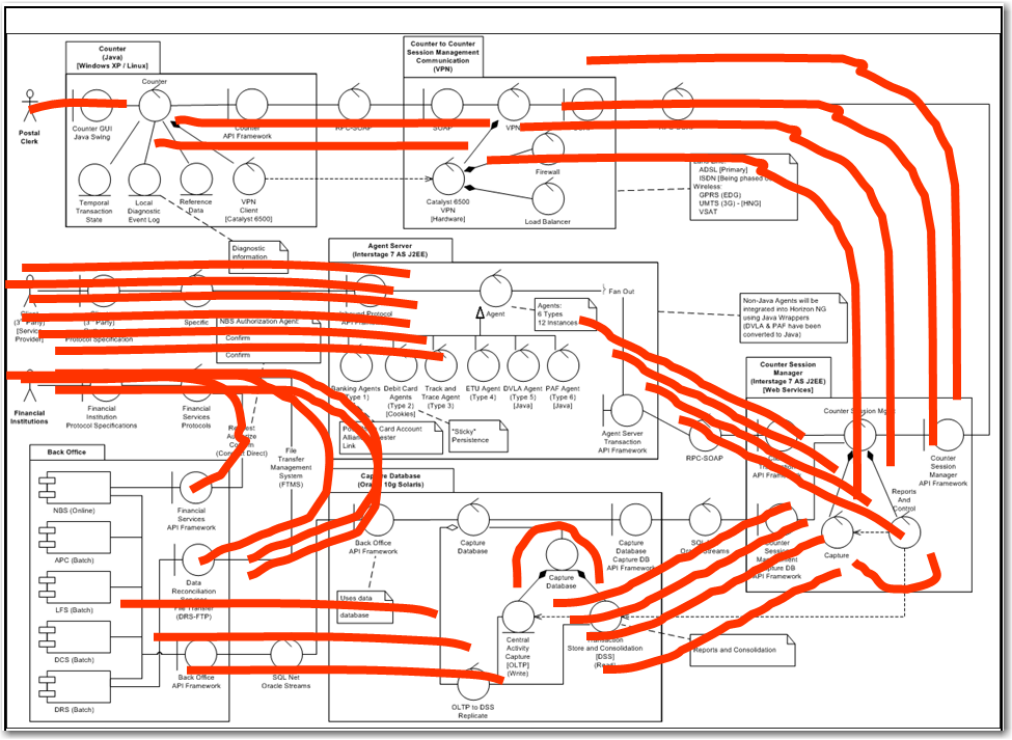


Figure 19: Business Event Walkthrough Scenario Mapping

The model can then be used to examine what happens across the enterprise in response to different business event scenarios (indicated by the red lines overlaying the model). These scenarios can be informally discussed before being modeled as part of an analysis exercise.

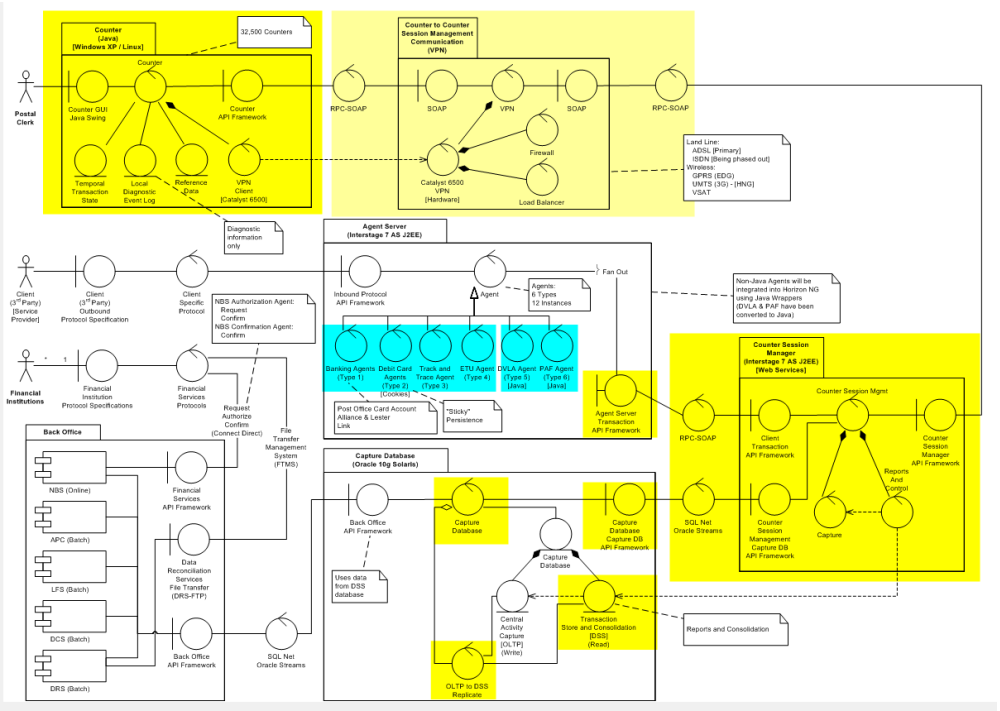


Figure 20: Major Areas of Change in the Architecture

A color coding scheme can be used to visually represent where change is required (yellow) and where changes cannot be made without significantly impacting the system (blue).

The results of the analysis are then used to create a target baseline (e.g. what the system will look like once it has been completed) as indicated in the diagram below, Figure 21:

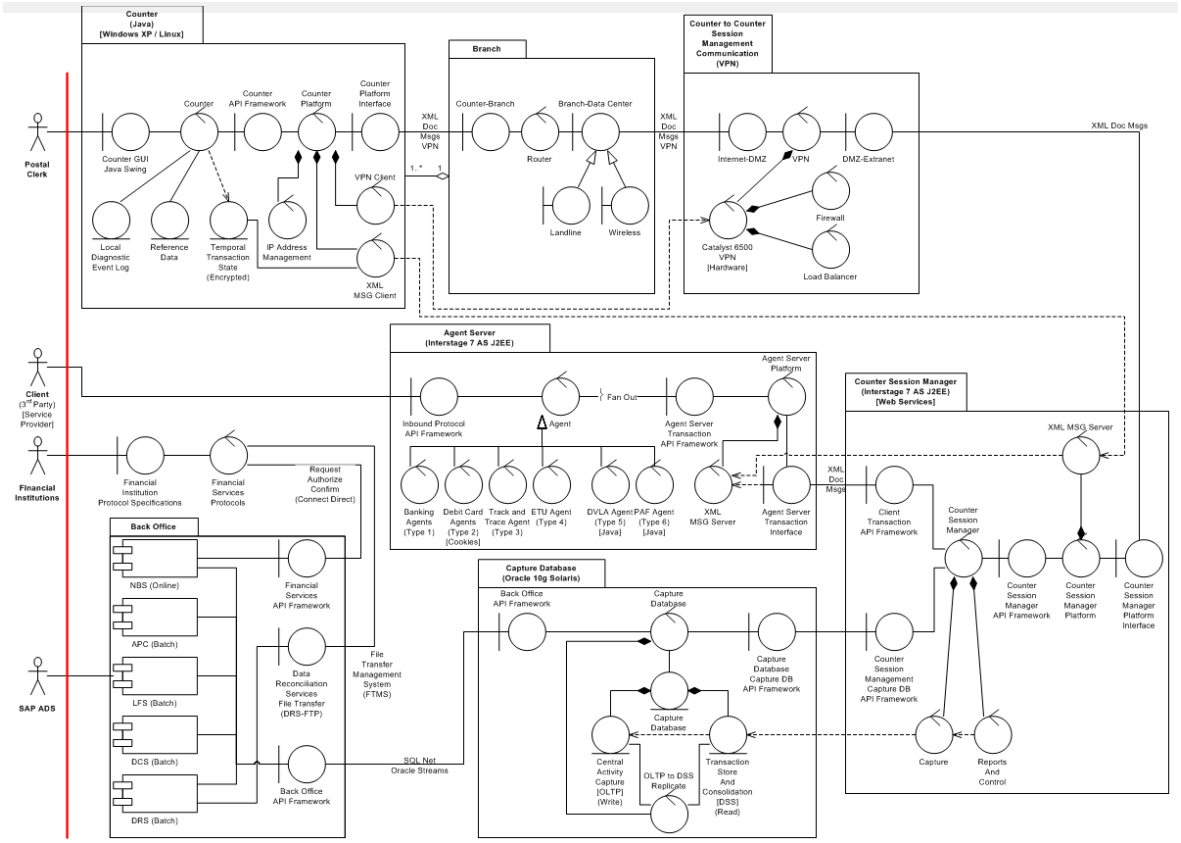


Figure 21: Target (to be) Baseline

The benefit of this approach is that a high-level view of both the current and target baselines is quickly made available from which the rest of the system can be elaborated from. This top-down approach maintains traceability. Both models can be visually compared side-by-side as shown below:

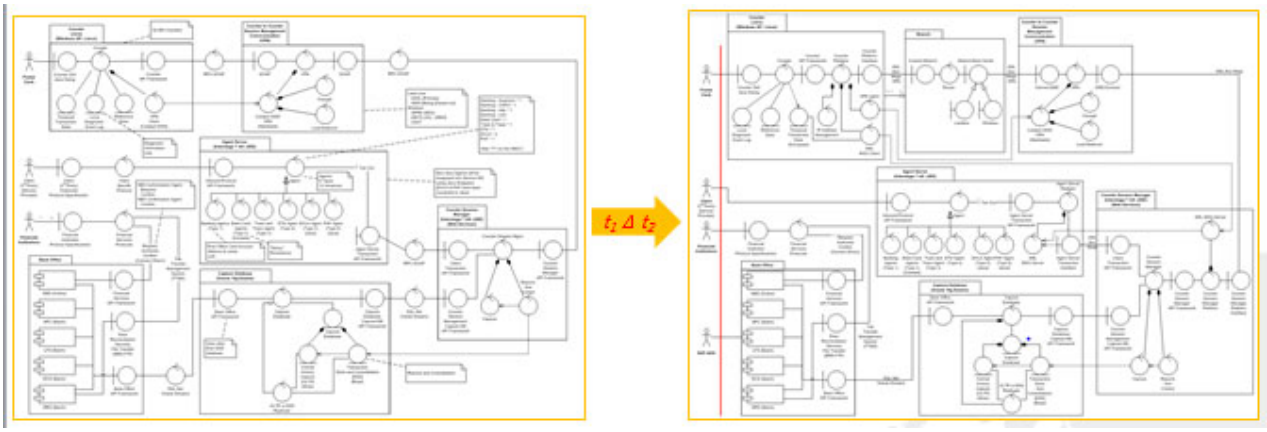


Figure 22: Comparing Current Baseline to Target Baseline

6.3 Case Study 3: Vendor Product Evaluation - Comparative Gap Analysis

PIM to PSM Mappings

This section compares the Platform Independent Model based on Proxy-Components introduced earlier in this document to each of the three commercial product stacks (WebSphere 5, Oracle 10g, and Microsoft .Net). A Proxy-Component PIM to PSM mapping was introduced by Donald Baldwin and Chris de Vaney at the OMG Technical Meeting in Toronto, Canada in September 2001. Donald Baldwin's work at the Enterprise Systems Architecture Laboratory has focused on developing a framework for using Proxy-Components in conjunction with a dual-axis Gap Analysis with both a transitional and comparative aspect. The Proxy-Component PIM is used in Comparative Gap Analysis, which compares different candidate implementation approaches based on the differences in how each approach compares to the PIM. In each case, the candidate implementation approach is functionally mapped to the Proxy-Component PIM. For example, if we look at the Proxy-Component "Application Integration Services" and map it to WebSphere 5, Oracle 10g, and Microsoft .Net we get the following mappings:

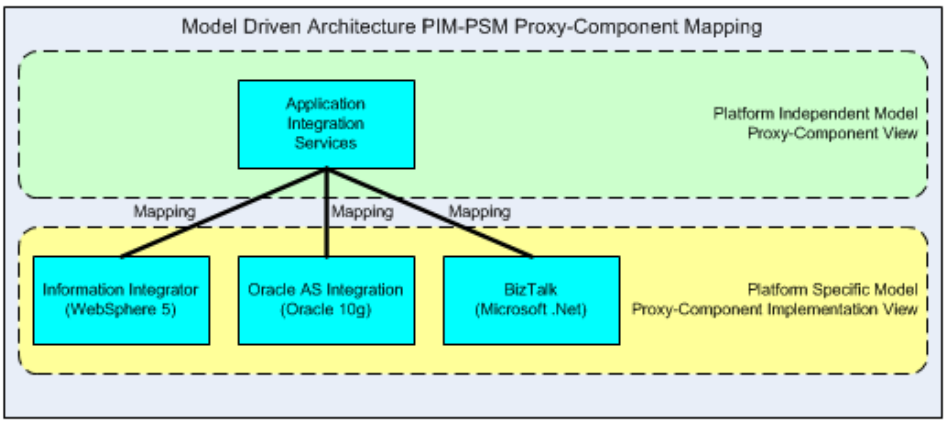


Figure 23: Proxy Component Example of PIM to PSM Mapping

The ability to make granular mappings allows a more effective evaluation of the different candidate implementation approaches and provides a framework for conducting a business and economic analysis to investigate which candidate approach should be selected. When combined with analysis tools such as PENG, the MDA Proxy-Component analysis can be a very powerful tool in planning, developing, and implementing an enterprise architecture with respect to managing technology, risk, and economic factors. The following sections provide a PIM-PSM Proxy-Component Mapping for IBM WebSphere 5, Oracle 10g, and Microsoft .Net as of 2004 June 14. Additionally, Fujitsu Interstage 7 and an example for using Open Source components are included. Each commercial product mapping worksheet was filled out by an architect employed by the vendor to ensure accuracy. The Open Source worksheet mappings were prepared by architects employed by ESAL.

Model Driven Architecture provides a pragmatic mechanism for comparing the Total Cost of Ownership between different alternate technical implantation approaches. The diagram below shows how MDA provides traceability from CIM to PIM to PSM viewpoints:

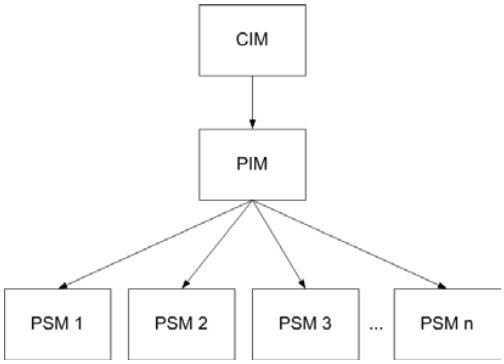


Figure 24: Conceptual example of CIM to PIM to multiple PSM mapping

The diagram below illustrates a comparative analysis of two possible messaging solutions for the Karolinska University Hospital Huddinge Telemedicine Platform based on PENG Analysis.

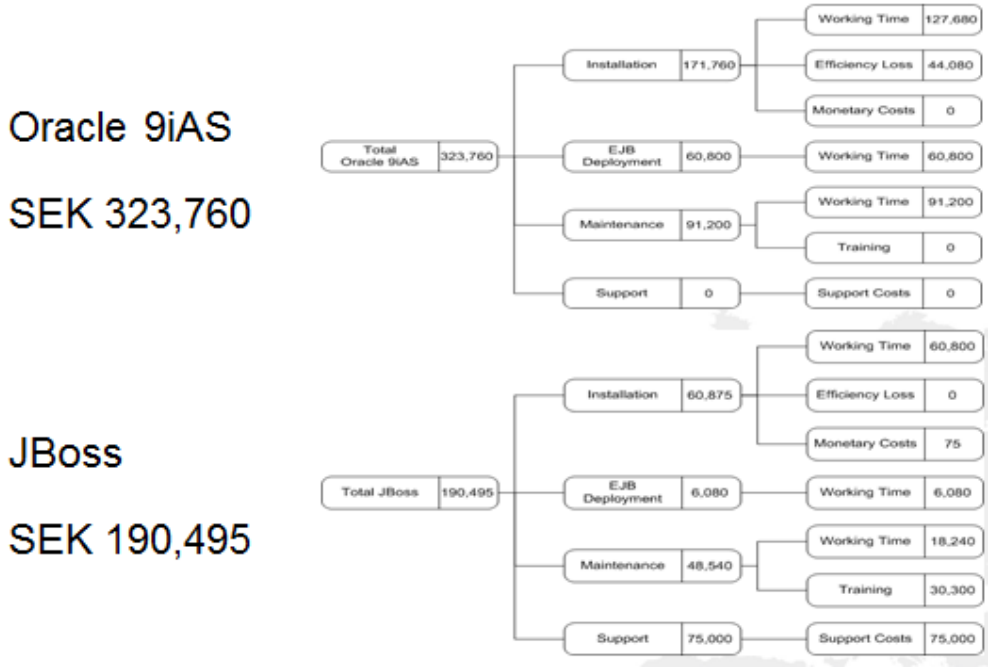


Figure 25: PENG example for cost comparison

The illustrations on the following page provide an example of how the DNEAF for EAIS and PPS can be used to conduct a comparison between IBM, Oracle and Microsoft platform products. Each vendor provides documentation on their products using quite different models; however, by mapping each product to the framework, a direct comparative mapping can be visualized. Greater detail can then be analyzed by using a worksheet to map domain neutral functions (Platform Independent Model view) to each product solution (Platform Specific Model view). In this scenario, the PIM elements of the archetype model serve as proxy components that can easily be directly compared with the product.

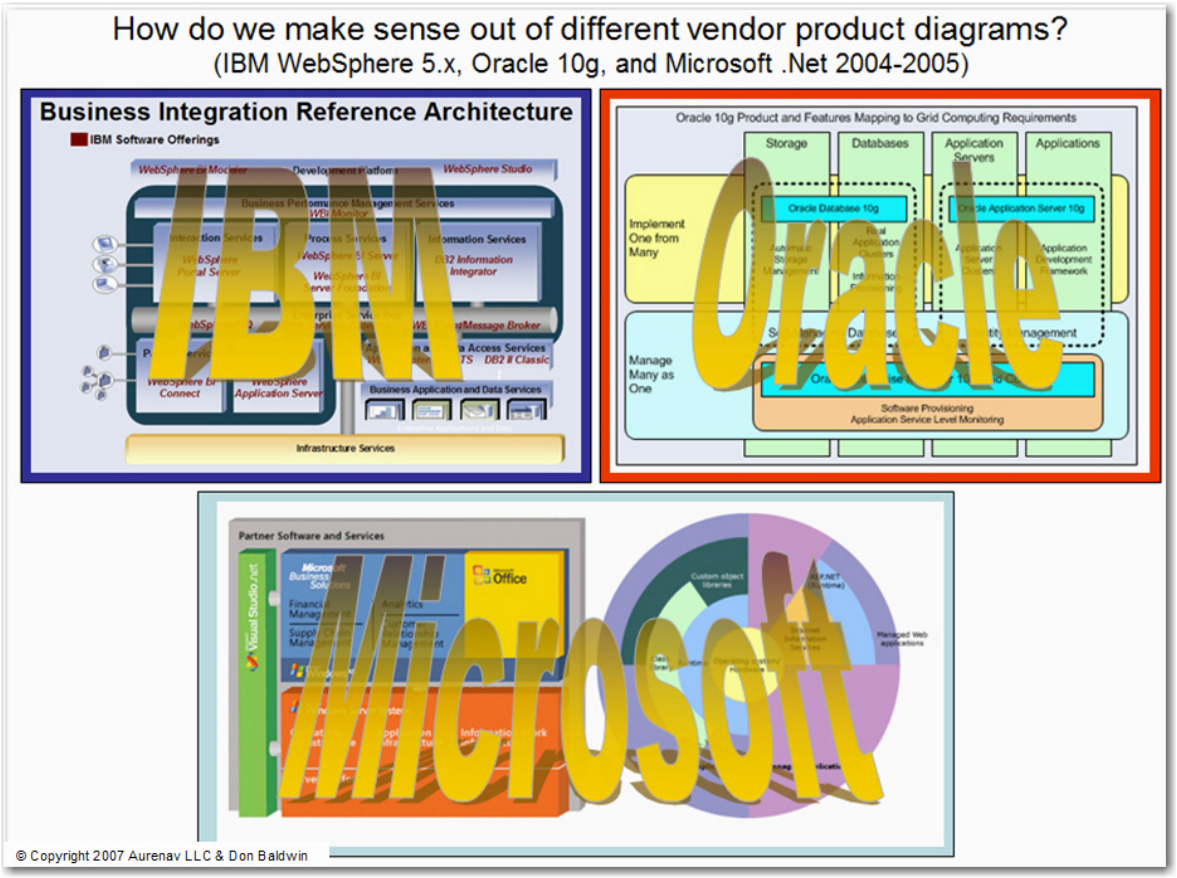


Figure 26: Comparative Gap Analysis of COTS without the DNEAF for EAIS and PPS

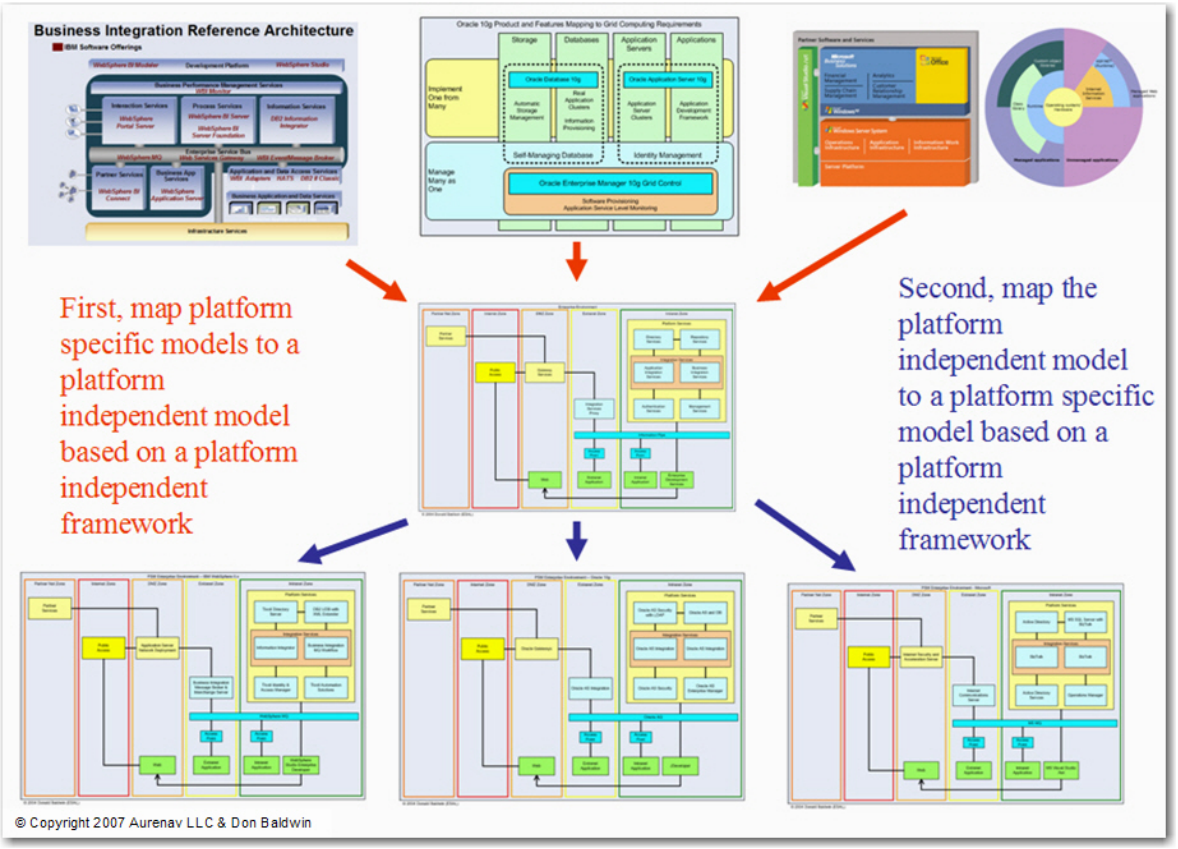


Figure 27: Comparative Gap Analysis of COTS using the DNEAF for EAIS and PPS

6.3.1 IBM WebSphere 5 PSM

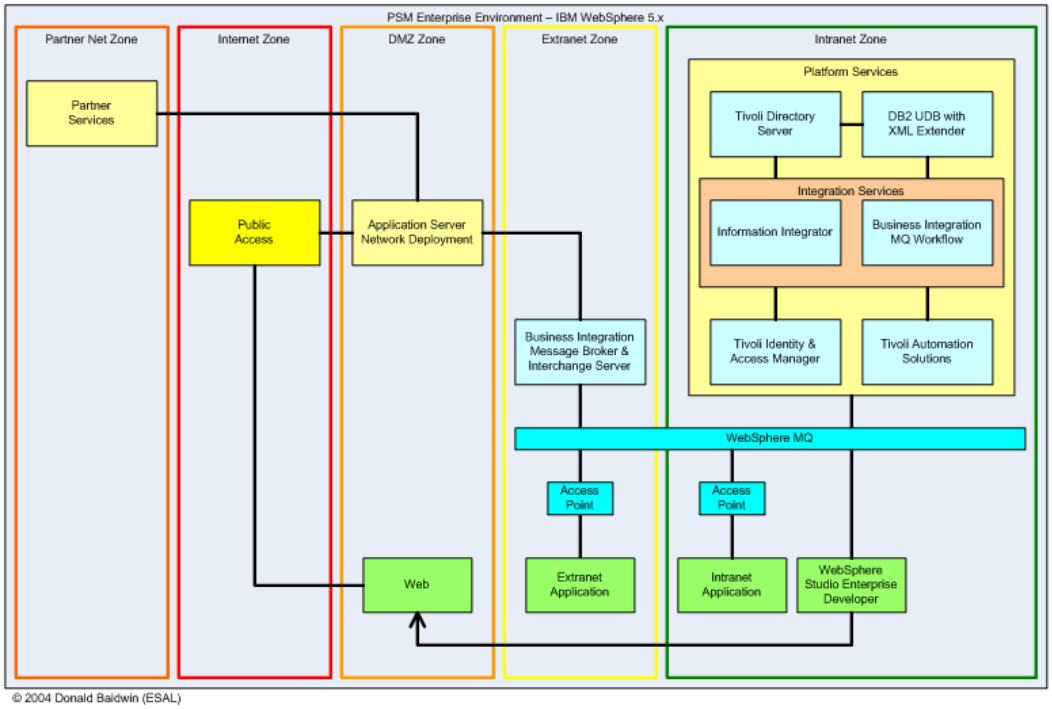


Figure 28: Mapping WebSphere to the DNEAF for EAIS and PPS

ESAL Product Notes for WebSphere version 5

WebSphere 5 is a good gateway to backend applications with its solid database, message queuing, and programming support. The current versions of WebSphere 5 have several enterprise systems components that are prone to bugs – the most critical being the Java Virtual Machine (JVM). IBM has made some important architectural changes to version 5, which is a major rewrite from version 4. Many of the current changes are impacting Java. As new Java-based changes are introduced, IBM releases new patches and updates for the product, which requires recompilations and in some cases recoding. On its own, this would not normally be a major problem; however, WebSphere 5 components are very tightly integrated so small changes anywhere in the architecture tend to spawn changes in many other components as well. Another notable change is the shift away from standard LDAP as a security repository to IBM's Commerce product. WebSphere 5 components are configured (and in many cases hard-coded) to trust the Commerce component, which creates serious problems if an enterprise has already adopted a generic LDAP solution. The tighter coupling of WebSphere components requires more knowledge and experience of the overall environment to successfully deploy a large-scale platform. Since WebSphere 5 is still relatively new, experience with the product is hard to find – even within IBM. The biggest changes (and benefits) will be realized when the EJB3 standard is adopted – many of the architectural changes in WebSphere 5 are optimized around EJB3 (IBM is a major promoter in driving the EJB3 initiative). WebSphere version 5 should be stable for production use by mid-2005 [18, 19].

Detailed WebSphere 5 Product Mapping (Comparative Worksheet)

Table 12: Mapping Worksheet for WebSphere

PIM Proxy Component	PSM Implementation Component
REPOSITORY SERVICES	Not a standard WebSphere product component
Business Process Repository	Not a standard WebSphere product component
XML Repository	DB2 Universal Database v8 with DB2 XML Extender
APPLICATION INTEGRATION SERVICES	DB2 Information Integrator v8
Basic Messaging Services	WebSphere MQ
Message Dictionary	Not a standard WebSphere product component
Data Services	DB2 Universal Database v8 Server
Data Dictionary	Not a standard WebSphere product component
Transformation Services	WebSphere Business Integration Message Broker and WebSphere Integration Interchange Server
Transformation and Routing Engine	WebSphere Business Integration Message Broker and WebSphere Integration Interchange Server
XML Engine	WebSphere Business Integration Message Broker, WebSphere Integration Interchange Server, WebSphere Business Integration Connect, and DB2 Universal Database v8 with DB2 XML Extender
Message Warehouse	Not a standard WebSphere product component
Message Queuing Services (Information Pipe)	WebSphere MQ
Publish and Subscribe Services	WebSphere Business Integration Message Broker, WebSphere Integration Interchange Server, and WebSphere MQ
Connector SDK	Not a standard WebSphere product component
BUSINESS INTEGRATION SERVICES	
Basic Integration Services	WebSphere v5 Product Suite
Workflow Management Services	WebSphere Business Integration MQ Workflow
Security Services	IBM Tivoli Access Manager
Partner Profile Services	Web Intermediaries (WBI) Connect
ERP Services	Web Intermediaries (WBI) Connect
EDI Services	Web Intermediaries (WBI) Connect
Content Management Services	DB2 Content Manager v8
ECM Metadata Repository	Not a standard WebSphere product component
Decision Support Services	DB2 Universal Database Warehouse Edition v8
Application Services	WebSphere Application Server
INTEGRATION SERVICES PROXY	
XML Services	WebSphere Business Integration Message Broker, WebSphere Integration Interchange Server, Web Intermediaries (WBI) Connect, and IBM Tivoli Access Manager (from a security perspective)
B2B Services	Web Intermediaries (WBI) Connect and IBM Tivoli Access Manager (from a security perspective)
AUTHENTICATION SERVICES	
RBAC Repository	IBM Tivoli Identity Manager and Commerce
Security Integration Component	IBM Tivoli Identity Manager
Single Sign-On Component (SSO)	IBM Tivoli Identity Manager
GATEWAY SERVICES	WebSphere Application Server Network Deployment
MANAGEMENT SERVICES	IBM Tivoli Automation Solutions (Note: This is a suite of solutions and not a specific product)
Transaction Performance Manager	IBM Tivoli Monitoring for Transaction Performance
Transaction Processor	IBM Tivoli Enterprise Console (Defined as the process taking care of all the events that report the status of the system)
Transaction Process Monitor	IBM Tivoli Monitoring
Logging Services	IBM Tivoli Enterprise Console and IBM Tivoli Data Warehouse
ENTERPRISE DEVELOPMENT ENVIRONMENT	WebSphere Studio Enterprise Developer
Developer IDE	WebSphere Studio Enterprise Developer and Eclipse
CASE Tools	WebSphere Studio Enterprise Developer (limited support)
Version Control Mechanism	IBM Rational ClearCase
Deployment Mechanism	IBM Tivoli Provisioning Manager

Note: Components with the comment "Not a standard WebSphere product component" must be provided by a third-party or built by the development team.

IBM WebSphere Product Strategy

IBM WebSphere 5.x is currently one of the most modular product stacks available. It supports many different operating systems (including Windows, Linux, and other UNIX flavors) and many different databases (including Oracle and Microsoft). WebSphere 5.x also provides wider spectrum of coverage in the enterprise than either Oracle or Microsoft as illustrated in the table below:

Table 13: Comparative Product Support by Computational Device Category

Computational Device Category	IBM WebSphere 5	Oracle 10g	Microsoft .Net
Personal Computational Devices	Client Support	N/A	Client Support
Personal Computers	Client Support	Client Support	Client Support
Small Servers	Server Support	Server Support	Server Support
Midrange Computers	Server Support	Server Support	Server Support
Mainframe Computers	Server Support	Server Support	N/A
Super Computers	Server Support	N/A	N/A

WebSphere 5 provides out-of-the-box components for many different operating systems from handheld devices such as PDA's and mobile telephones to the largest super computers.

Another characteristic of WebSphere 5 can be seen in how IBM has adopted a new modular component approach in the product stack architecture to provide closer integration between the different WebSphere 5 products. IBM has been developing this strategy over the past few years by developing its own component solutions and through mergers and acquisitions of third-party companies (examples include Lotus, Tivoli, and Candle). While this simplifies product integration in many respects, there are also some drawbacks since some of the integrations have been hard-coded, which complicates integration efforts involving third-party or custom-built components (third-party in this context defined as components and products not owned by IBM through internal development or through a merger and acquisition).

IBM WebSphere 5 offers several component advantages over its competitors including:

- A distributed Message Queue product that provides a high-level of point-to-point security encryption that can be extended through all the network zones
- A Message Queue implementation that requires minimal system overheads
- Robust load-balancing and queue management (load-balancing uses a round-robin strategy where the Queue Manager placed messages on a rotating basis to a queue cluster)
- IBM has developed an efficient XML transformation engine that is faster and easier to cluster than traditional XSLT transformation engine implementations used by competitors
- IBM has adaptors to more legacy systems than most of its competitors and offers native support for mainframe and super computer environments

IBM's enterprise portfolio is based around four main product lines: Tivoli for enterprise management services, WebSphere 5 for application and business integration services, DB2 for database services, and Lotus for workgroup services. These product lines provide a full range of services for e-Business and e-Government portal platforms.

Advantages

IBM WebSphere MQ is one of the easiest message queuing products available on the market – it has more adaptors out-of-the-box than any of its competitors and the MQ Client requires minimal overhead. The MQ Servers can be installed independently of other IBM products and have very minimal overheads. The use of individual components to provide enterprise functionality creates

an easier to manage infrastructure environment and decouples server installation and configuration dependencies between components – this promotes better robustness in the platform.

Disadvantages
WebSphere workflows cannot be triggered directly from an application in the current product offering (IBM is working on this issue). The recent revision to the WebSphere architecture in version 5 combined with the integration of components taken over by IBM through merger and acquisition has contributed to timeout and stability issues in some of the integrations between components. While WebSphere is a powerful development environment, it has also become a very complex environment. It is very difficult (if not impossible) to find people with the broad set of WebSphere skills and experience needed to successfully build a solution – instead, an organization must rely on multiple developers, each with a specialist concentration in a WebSphere component area. WebSphere development teams typically require more developers than a comparable Oracle or Microsoft project; however, WebSphere is more scalable than either Oracle or Microsoft. Finally, many of the WebSphere components are not best-of- breed – while functional, there are better solutions available (especially from the business stakeholder perspective – Content Management Services and Performance Monitors being good examples).

Summary
IBM has achieved a solid component-based architecture with WebSphere 5 that provides an organization considerable flexibility regarding integrating legacy systems into the enterprise platform framework. Over the next year (into 2005), WebSphere 5 will continue its evolution and should gain considerable stability. The modular architecture of WebSphere promotes hybrid solutions in which different WebSphere and third-party components can be integrated together, which allows an organization to leverage best-of- breed solutions.

6.3.2 Oracle 10g PSM

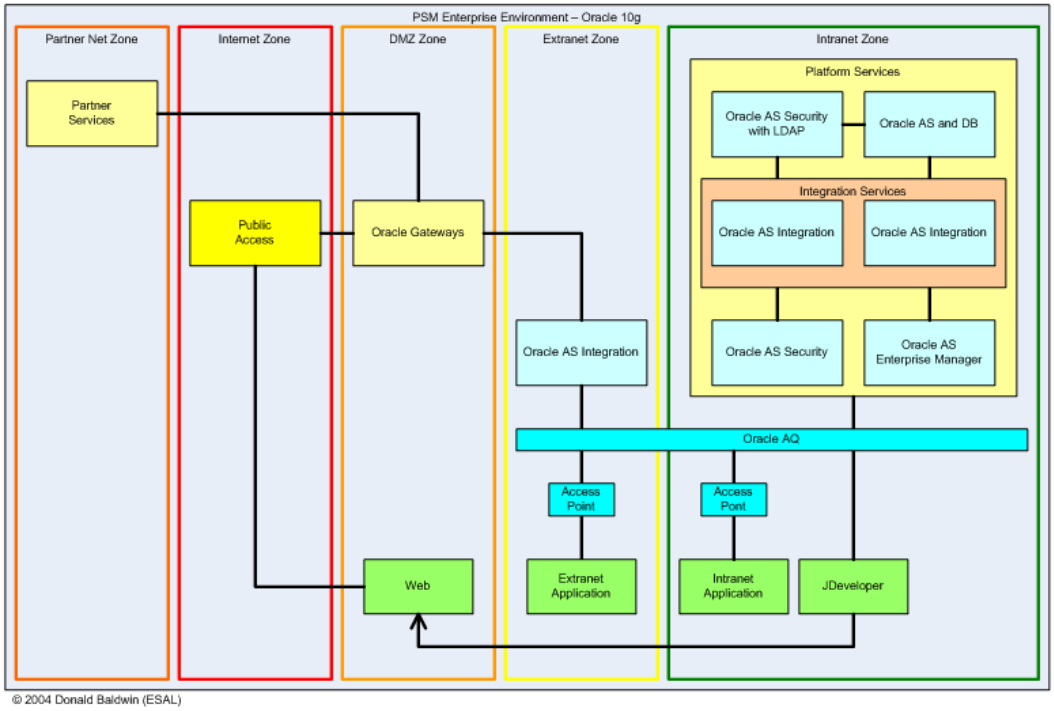


Figure 29: Mapping Oracle to the DNEAF for EAIS and PPS

ESAL Product Notes for Oracle 10g

The Oracle Application Server and Workflow Engine exhibit a number of bugs and performance problems in large-scale enterprise platforms that limit upward scalability in large enterprise platforms. ESAL product evaluation studies comparing WebSphere, Oracle, and .Net have also shown increasing problems with product support and stability over the past couple of years. The out-of-the-box installation as distributed by Oracle will not work properly until all the services patches are applied. The decision to bundle so much enterprise functionality into the Application Server product means that there is not a "standard" installation and configuration approach, which is problematic when using any of the various installation and configuration documents available through Oracle support since these documents do not tend to explain the assumptions behind the installation or configuration approach. To properly install and configure Oracle, the development team needs to be trained in the product and must also have a deep understanding of the product. For these reasons, Oracle was the most difficult product stack to implement out-of-the-box. Successful use of Oracle requires a highly trained and experienced development team. Oracle currently sits between WebSphere and .Net regarding suitability in large-scale enterprise platforms. Two architectural issues facing Oracle is the reliance on an evolved version of the XSLT code base for transformation, which creates a performance bottleneck, and the lack of a mechanism to connect to the message queue through different network zones.

Detailed Oracle 10g Product Mapping (Comparative Worksheet)

Table 14: Mapping Worksheet for Oracle

PIM Proxy Component	PSM Implementation Component
REPOSITORY SERVICES	Oracle Database and Oracle 10gAS
Business Process Repository	Not a standard Oracle product component
XML Repository	Oracle Database has specific XML functionality built in which is supported by tools within Oracle JDeveloper 10g
APPLICATION INTEGRATION SERVICES	Oracle 10gAS Integration with Interconnect and Process Connect
Basic Messaging Services	Oracle AQ (Advanced Queue) which is built into Oracle 10gAS
Message Dictionary	Oracle Integration Services (a repository where the transformation and routing is defined along with relevant rules)
Data Services	Oracle Integration Services (a repository where the transformation and routing is defined along with relevant rules)
Data Dictionary	Oracle Integration Services (a repository where the transformation and routing is defined along with relevant rules)
Transformation Services	Oracle Integration Services (a repository where the transformation and routing is defined along with relevant rules)
Transformation and Routing Engine	Oracle Integration Services (a repository where the transformation and routing is defined along with relevant rules)
XML Engine	XSLT (Oracle has developed a customized solution by extending the base XSLT code-base) and an XML architectural adaptor has been provided but third-party adaptors are available and custom adaptors can be used. Implementation note: Clustering is required to scale up and transactions are limited to on Application Server unless a pure EJB is used.
Message Warehouse	Not a standard Oracle product component Oracle only supports asynchronous messages Chorus, a third-party Partner Application, is recommended for long-term Message Warehouse requirements
Message Queuing Services (Information Pipe)	Oracle AQ (Management layer is inside Oracle 10g Integration)
Publish and Subscribe Services	Oracle AQ with rules set up within Oracle Integration
Connector SDK	Oracle 10g JDeveloper or Oracle 10g Integration Services API for developing own adaptor (typically needed with legacy Publish and Subscribe Services)
BUSINESS INTEGRATION SERVICES	Oracle 10gAS Integration
Basic Integration Services	Oracle 10gAS Integration Services (i.e. an SQL adaptor)
Workflow Management Services	Oracle 10gAS Workflow (a Workflow Engine is bundled within Oracle AS and is used by Oracle Integration)
Security Services	Oracle 10gAS Security (an LDAP catalog handles certificates and partner authority)
Partner Profile Services	Oracle Adaptors (part of Oracle10gAS Integration)
ERP Services	Oracle Adaptors (part of Oracle10gAS Integration)
EDI Services	Oracle specific EDI adaptors are part of Oracle10gAS Integration
Content Management Services	Oracle 10gAS Portal uses Workflow Engine
ECM Metadata Repository	Oracle 10gAS Portal Server includes security
Decision Support Services	Oracle 10gAS BIW (Business Intelligence Warehouse), Oracle 10gAS Decision Support, and End-user tools such as BI (Business Intelligence) Beans to develop EJB's that can handle multidimensional analysis
Application Services	Oracle 10gAS (Application Server)
INTEGRATION SERVICES PROXY	Oracle 10gAS Integration
XML Services	Oracle 10gAS Integration
B2B Services	Oracle 10gAS Integration
AUTHENTICATION SERVICES	Oracle 10gAS Security
RBAC Repository	LDAP (version 3 compatible)
Security Integration Component	Oracle Single Sign-On and Partner Applications
Single Sign-On Component (SSO)	Oracle Single Sign-On
GATEWAY SERVICES	Oracle Gateways
MANAGEMENT SERVICES	Oracle 10gAS Enterprise Manager
Transaction Performance Manager	Oracle 10gAS Enterprise Manager can handle nodal transactions but is application dependent at the Application Server, Oracle 10gAS Discover, and EJB's
Transaction Processor	Part of the EJB implementation includes Process Monitoring and Logging Services
Transaction Process Monitor	EJB container implementation
Logging Services	EJB container implementation

ENTERPRISE DEVELOPMENT ENVIRONMENT	Oracle 10g JDeveloper
Developer IDE	Oracle 10g JDeveloper
CASE Tools	Oracle 10g JDeveloper
Version Control Mechanism	Oracle 10g JDeveloper has built in support for basic Version Control Third-party tools can be used for more advanced Version Control
Deployment Mechanism	Oracle 10g JDeveloper

Note: Components with the comment "Not a standard Oracle product component" must be provided by a third-party or built by the development team.

Oracle Product Strategy

Oracle's product strategy is based on a grid computing architecture and a portfolio of software products grouped around Oracle Database 10g and Oracle Application Server 10g, which shifts the focus from the integration of applications to the integration of computing infrastructure. Oracle's grid computing architecture for the Enterprise is designed to address three factors:

- Scalability: the ability to increase the computing capacity when needed
- Capacity Growth: the ability to easily scale up computing capacity
- Management Costs: the ability to manage the cost of integrating complex systems

Oracle's Enterprise grid computing integrates business computing resources, industry standard servers, and storage. According to Oracle, Oracle Database 10g, Oracle Application Server 10g, and Oracle Enterprise Manager 10g provide the first complete grid infrastructure software [6]:

> "Oracle offers organizations a comprehensive solution to manage information and run Enterprise applications on grids. Oracle Database 10g has been designed to manage information on computing grids called database grids. Oracle Application Server 10g (OracleAS 10g) has been designed to run enterprise applications on computing grids called application server grids. Both Oracle Database 10g and Oracle Application Server 10g can be very efficiently managed in a grid computing environment using Oracle Enterprise Manager 10g Grid Control"

In addition, Oracle is embedding grid capabilities now found in the current Enterprise Manager into the whole 10g line, from the database to the application server. Oracle Application Server 10g is an integrated suite of application infrastructure software, designed to run enterprise applications on computing grids, pools of low cost servers and storage. OracleAS 10g offers functionality supporting J2EE, high-speed caching, rapid application development, enterprise portals, identity management, business intelligence, and application and business integration.

OracleAS 10g features are grouped in five categories:

- Software Provisioning: OracleAS10g provides support for the automation of installation, configuration and provisioning of software.
- User Provisioning and Security: OracleAS10g facilitates provisioning of users and their identities, including access control to resources and applications over the grid.
- Application Management and Monitoring: OracleAS10g provides facilities to monitor and tune applications to provide optimal performance.
- Work Load Management: OracleAS10g provides scalability and availability while limiting idle computing capacity.
- Systems Management and Monitoring: OracleAS10g together with Oracle Grid Control supports system management and monitoring at the grid level

OracleAS 10g provides the following capabilities:

- **Installation:** Single Node, Multi-mode, Automated, Silent installation, Pre-Install & Post-Install Checks, All OS's
- **SW Configuration Management:** Archive multiple versions, Restore configuration baseline, Apply configuration, Apply to Cluster
- **Cloning:** Cloning SW Systems, Applications Servers and Applications deployed.
- **Patching:** SW Patching and Upgrade of Application Servers
- **Upgrading:** automatic upgrade from 9iAS to 10g.
- **Operational Task Automation:** using Job Scheduler and Cluster and Group Management Systems across Application Servers.

Advantages
The Oracle enterprise platform product stack supports a variety of applications and the database product is in wide spread use. Oracle is a leader in the database server market and there are a large variety of products and tools available from third-party vendors. Oracle licenses its products on a per CPU basis and requires relatively few products to provide an enterprise platform strategy; however, there remains a large infrastructure requirement to provide a large-scale enterprise solution.

Disadvantages
The Oracle product stack is built predominantly around the database. The centralized message queue approach, which requires the database (the database must be accessible to use Oracle AQ), requires third-party solutions to traverse firewalls and the various network zones. Most of the enterprise platform services are contained within the Application Server, which requires a careful strategy to properly configure each functional cluster of computing infrastructure.

Summary
Oracle's strategy builds upon three core product offerings: Oracle 10g Database, Application Server, and Enterprise Manager. These three products fit into the Oracle Grid Computing framework, which focuses on managing infrastructure. Large-scale enterprise platforms built on the Oracle product stack require extensive use of clusters and often require a number of third-party applications in order to provide required scalability and reliability.

6.3.3 Microsoft .Net PSM

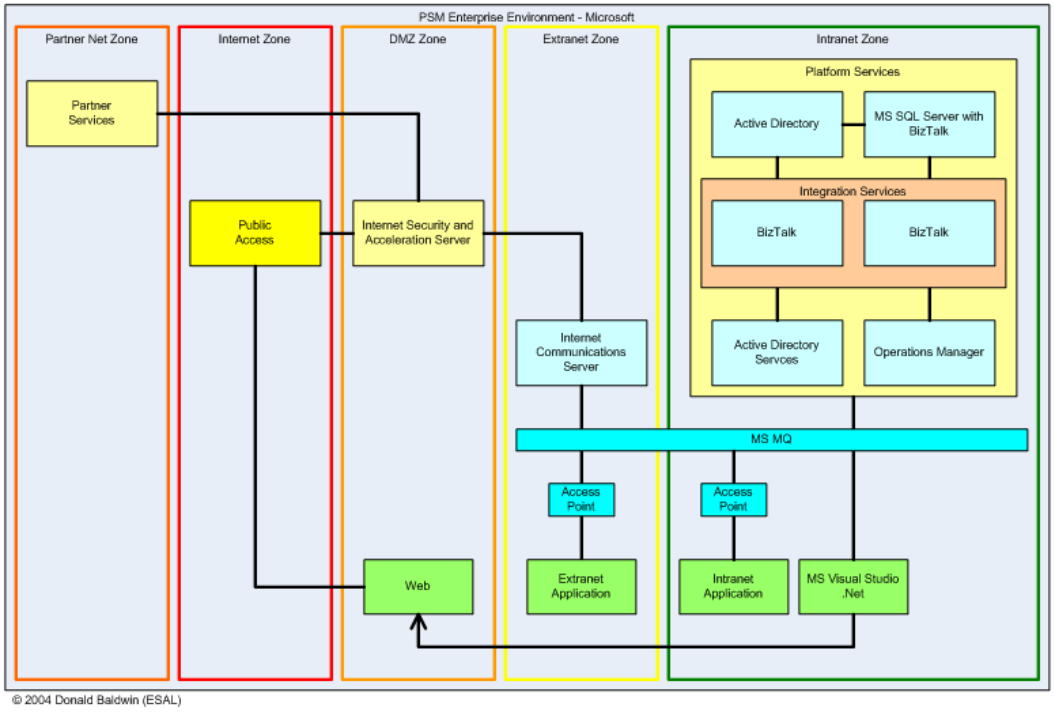


Figure 30: Mapping Microsoft to the DNEAF for EAIS and PPS

ESAL Product Notes for Microsoft .Net

Microsoft .Net is moving to a Service Oriented Architecture (SOA), which is heavily focused on SOAP and on Integration Services provided by Microsoft BizTalk. Microsoft .Net is unique from the rest of the product stacks evaluated in this report (the others being WebSphere, Oracle, and Open Source), in that Microsoft does not have the equivalent to an Application Server (the closest thing that Microsoft has is BizTalk, which provides very limited functionality normally found in an Application Server).[24, 25, 26] Microsoft .Net requires the development team to take responsibility for making sure that many of the functions provided by the traditional Application Server are taken care of, which requires a highly skilled and experienced development team. The trade-offs with this approach involve granular control over the enterprise environment compared with working within an environment controlled by the Application Server. The latest releases of .Net are removing access to some of the granular control previously available to the developer. This again has tradeoffs: The environment will become more standardized but the programmers and systems developers will not have the same degree of freedom to fine- tune their work. Currently, .Net is a very difficult environment to use in a large-scale enterprise and instead is primarily used in SME scale platforms. The heavy reliance on individual programmer and system developer experience tends to limit the effectiveness of .Net as a candidate for large-scale enterprise platforms. Where there have been successes, there has been an accompanied use of automated development and workgroup tools to assist and manage the development efforts [22, 38, 39].

Detailed Microsoft .Net Product Mapping (Comparative Worksheet)

Table 15: Mapping Worksheet for Microsoft

PIM Proxy Component	PSM Implementation Component
REPOSITORY SERVICES	MS SQL Server
Business Process Repository	MS BizTalk
XML Repository	MS SQL Server
APPLICATION INTEGRATION SERVICES	MS BizTalk
Basic Messaging Services	MS MQ
Message Dictionary	Not a standard Microsoft product component Need to build – no out-of-the-box solution
Data Services	Not a standard Microsoft product component "MS Provider Model" – this is an architecture and not a product that leverages ADO.Net, OLEDB, and ODBC
Data Dictionary	Not a standard Microsoft product component Need to build using MS SQL Server
Transformation Services	XSLT (MS adaptation of their own engine based on the XSLT code- base)
Transformation and Routing Engine	MS BizTalk
XML Engine	Microsoft has developed their own XML Engine
Message Warehouse	Not a standard Microsoft product component A number of components have this functionality built-in
Message Queuing Services (Information Pipe)	MS MQ
Publish and Subscribe Services	Not a standard Microsoft product component .Net Framework 2005 will provide a component solution, currently: Triggers can be used (but many developers choose not to), Can also use a monitor, and BizTalk is also an option by setting up a library of schemas
Connector SDK	MS Messaging SDK
BUSINESS INTEGRATION SERVICES	MS BizTalk
Basic Integration Services	MS BizTalk
Workflow Management Services	MS BizTalk – Orchestrator (Essentially using Visio to graphically ma out the workflows and using hooks to call/trigger enterprise calls)
Security Services	MS BizTalk requires Active Directory
Partner Profile Services	MS BizTalk and MS Commerce Server (provides a light-weight solution)
ERP Services	Great Plains (Acquired by MS)
EDI Services	MS BizTalk (built-in support for a number of standard protocols including EDIFACT, HL7, X.12 "EDI", etc.)
Content Management Services	MS SharePoint Portal Server (for robust services requiring file-based content with approval workflows) or MS Content Manager (for basic services that rely on forms-based workflows)
ECM Metadata Repository	Not a standard Microsoft product component
Decision Support Services	MS OLAP Tools and Crystal Decisions
Application Services	MS Application Server and IIS Web Services
INTEGRATION SERVICES PROXY	Internet Communications Server (similar to IIS but built for hosting services)
XML Services	Not a standard Microsoft product component Internet Information Server (IIS) 6 and SOAP are native with .Net but requires forwarding to MS BizTalk for processing (not a one-stop solution)
B2B Services	Not a standard Microsoft product component Internet Information Server (IIS) 6 and SOAP are native with .Net but requires forwarding to MS BizTalk for processing (not a one-stop solution)
AUTHENTICATION SERVICES	MS Active Directory Services
RBAC Repository	MS Active Directory Services
Security Integration Component	Not a standard Microsoft product component MS PassPort SDK can be used as an alternative
Single Sign-On Component (SSO)	Not a standard Microsoft product component MS PassPort (managed by Microsoft) can be used as an alternative
GATEWAY SERVICES	MS Internet Security and Acceleration Server (MS ISA Server)
ACCESS POINT	Not a standard Microsoft product component System Management Console can be used as an alternative – but this is a very primitive solution
Access Point Connector	Not a standard Microsoft product component Typically need to build – Full API support is available (and is a better solution)

MANAGEMENT SERVICES	MS Operations Manager
Transaction Performance Manager	Not a standard Microsoft product component MS Performance Monitor and .Net API
Transaction Processor	Com+ Transaction Processor (also referred to as D-COM)
Transaction Process Monitor	Not a standard Microsoft product component MS Performance Monitor and .Net API
Logging Services	Not a standard Microsoft product component MS Performance Monitor and .Net API
ENTERPRISE DEVELOPMENT ENVIRONMENT	MS Visual Studio .Net
Developer IDE	MS Visual Studio .Net
CASE Tools	MS Visio for Enterprise Architects (has limited functionality)
Version Control Mechanism	MS Visual Source Safe
Deployment Mechanism	MS Visual Studio .Net can package and deploy products

Note: Components with the comment "Not a standard Microsoft product component" must be provided by a third-party or built by the development team.

Microsoft .Net Product Strategy

According to Microsoft's website [7], ".NET is a set of Microsoft software technologies for connecting information, people, systems, and devices." built around the .NET framework. Using Web services, .NET allows software integration between applications over the Internet. Although .NET is marketed as a Web Services package, when used in conjunction with other Microsoft products, it can be used as a full EAI package. It is Microsoft's goal for .NET, utilizing other products within Microsoft's product library, to provide quickly deployable connected solutions using Web Services. The foundation of .NET relies on "Smart" Client applications accessing Web Services such as authentication, Server Infrastructure, and developer tools like Microsoft Visual Studio .NET and the Microsoft .NET Framework [36]. The use of XML-based applications and processes function as services which are integrated through information and functionality. The .NET platform uses client software, web services, servers, and developer tools to provide a business solution for the integration of people, systems, information, and devices [30].

Many have wondered if .NET is Microsoft's attempt to kill Java. Although Microsoft .NET does offer many good features, it still lags behind Java in some key areas. There are fewer options in building business logic and database components. Although Microsoft has a set of recommended best practices there are no official component models to follow as in Enterprise Java Beans (EJB). Developers must devise their own component models based on these practices. In J2EE, developers use a standardized framework of existing component models to develop EJB's [20].

Microsoft .NET offers a more robust memory and system resource management using IL (Intermediate Language). One of the biggest benefits offered is garbage collection. .NET can run for weeks or months without eroding system resources. The simplified deployment model eliminates the requirement (in previous Microsoft platform frameworks) to add entries to Windows System registry, which eliminates the problem with having to manage the registry of DLL's by allowing different versions of same software to coexist in side by side deployments [20].

Common Language Infrastructure (CLI) and Common Language Runtime (CLR) will soon run on non-windows platforms. Open source .Net is on its way. .NET does not require a particular programming language. There are currently 23 supported languages which can interoperate on the same platform such as COBAL, FORTRAN, and JUMP (Java User Migration Path, a Java conversion tool [20, 34, 35, 36].

Where Microsoft excels is the in the hardwired support for SOAP and WSDL that is integrated into .NET. It is much better than J2EE servers at hiding the complexity of Web Services. MS .NET surpasses BEA WebLogic Workshop in the simplicity of exposing code as Web methods. Even with this advantage it is still believed that .NET does not offer reliability or security needed for Enterprise Web Service Deployment. Using .NET Virtual Machine can hide complexity of making a SOAP call. Tags with attributes (called Webservice) make sure that the XML needed to make SOAP calls are handled beneath the hood [20].

For business integration, Microsoft relies on BizTalk Server 2002 builds XML-based business processes across applications and organizations. BizTalk Server provides EAI functionality to the .Net platform framework. BizTalk services are described below:

- **Internal Application Integration Services:** Provides support for the integration of off-the-shelf and custom applications into the current business process
- **Trading Partner Deployment and Management Services:** Uses wizards to create, test, and deploy the automation of trading partner relationships
- **Business Activity Monitoring:** Enables worker to monitor transactions and processes in real time
- **Comprehensive Rules Engine:** Used to define, manage, and deploy the enterprise's business rules used during the business processes
- **Document Transport and Routing Services:** Supports the evaluation of document routing and transport services
- **Data Transformation Services:** Performs data transformation from the different formats used by business applications into XML
- **Process Automation and Management Services:** Helps to more efficiently reorganized business process through the integration of applications and data sources
- **Data Analysis Services:** Utilizes graphical tools and automated data mining utilities to combine, access, and analyzing operation data to discover patterns and trends
- **Configuration and Management Services:** Supports the management, administration, and automated monitoring of production
- **Microsoft Windows Platform Integration Services:** Support the interoperability of the platform across the enterprise using directory, data, and security services as well as the operating system
- **Scalability Services:** Supports the scaling of solution to address changes in volume needs
- **Security Integration Services:** Supports the integration of security credentials across the enterprise

BizTalk also offers additional components that support industry standards such as HIPAA and HL7 and it also includes functionality which provides support for planning and deploying platform services.

BizTalk Messaging Services provide the following services:

- Receiving incoming documents
- Parsing documents
- Extracting key identifiers and specific routing rules
- Delivery and tracking of documents

Delivery of messages can be through HTTP, HTTPS, SMTP, FTP, Message Queuing, Application Integration Components (AIC), and loop-back. It has a guaranteed "once only delivery" system and a notification system. Security features are provided by integration between BizTalk Server and Windows Server 2003.

Advantages
BizTalk supports Publish and Subscribe and Computer Telephony Interface through TAPI. The user interface is browser based. BizTalk has more the 350 adapters. It does not have any proprietary remote access methods [6]. Reuse is promoted through graphical tools. Microsoft plans on a major release once every 18 months, which provides a degree of platform stability. BizTalk also supports X.509 certificates, PKI, and Kerberos [11].

Disadvantages
Some of BizTalk's major shortcomings, as reported by the EAI Toolbox [11], is that it only runs on the Microsoft/Intel Platform. They go on to talk about how BizTalk only supports XML Data Reduced (XDR) not XML Schema Definition. The Repository only works with MS SQL. It requires other software to use all features such as Microsoft Operations Manager and Application Center. BizTalk uses a "hub and spoke" architecture which increases network usage. It uses proprietary notation XLANG for process modeling. Another major issue is that it is not UML Compliant. It only provides validation of WEDI/SNIP level 1 and 2 natively through the BizTalk HIPAA Accelerator. To provide validation for levels 3 and 4, development of COM objects by Microsoft is required [23]. There are no automated backups and recovery is manual. The Fault Tolerance capabilities are supported through other Microsoft products. Features such as importing, exporting, security, and administrative capabilities are performed through SQL server and not integrated into BizTalk. The repository cannot be integrated with other repository tools. The Upgrade process requires a system reboot which affects availability. It is not tested and certified on Datacenter which is required when scaling up over eight processors. Adapter support is only given through third party vendors. BizTalk relies on Windows and SQL security [11].

It is not easy to utilize the scalability features of BizTalk and it typically requires specific versions of Windows, which adds further complexity. To achieve high availability, the backup and recovery of multiple Microsoft products is required [11].

BizTalk is intended to eliminate the need for an agreed upon transport protocol using a "non-glue" approach (using Microsoft terminology). Although there is no need for a standard messaging component, Microsoft strongly encourages developers to use either Microsoft Message Queue (MSMQ) or COM+ [10]. An Application is not supposed to need awareness of other application calling conventions for the exchange and process of BizTalk message envelopes, but several state variables that require some knowledge of platform specific application contexts have been left undefined. BizTalk is supposed to eliminate the need for adapter software layers but it strongly suggests using a general-purpose e-Commerce adapter layer [10, 27].

Microsoft Message Queuing (MSMQ) is the message queue technology used by Microsoft. It enables the applications running on the .NET platform to communicate across the network, even to applications or systems which may be offline. Applications send and read messages off the queue. MSMQ can be used for both asynchronous and synchronous messaging. Applications can be developed using either application programming interfaces (API's) or Component Object Model (COM) objects. The .NET framework comes with a set of standard managed Message Queuing objects.

The newest version of MSMQ will be delivered as a part of "Longhorn", the next version of Windows. It will interoperate with Indigo which is a set of new programming frameworks operating around the Web services architecture. BizTalk Server has its own Message Queuing technology called MSMQ-T which is an adapter for MSMQ allowing MSMQ applications to speak to BizTalk Server. Also, the next version of SQL Server, Yukon, will have a Service Broker which will facilitate building message queuing into SQL Server. Functionally and operationally, MSMQ is very similar to IBM WebSphere MQ – including a similar round-robin strategy approach to load balancing.

There are some clear advantages and disadvantages of using MSMQ. As stated in Enterprise Application Integration "MSMQ offers good out-of-the-box functionalities and integrates well into Windows-based application tools. It is also more reasonably priced since it is included in the Windows Server platforms. However, it is not possible to perform asynchronous messaging with any other platform other than Microsoft without using some type of a bridge to another MOM product native to that platform. In other words, MSMQ can be a MOM for an enterprise only if the entire enterprise is hosted on Microsoft." [23]

Microsoft's XML standard (MSXML) v4.0 and later fully support Extensible Stylesheet Language Transformations (XSLT) version 1.0. An Extensible Stylesheet Language (XSL) is a language definition for XML data presentation and data transformations. Microsoft recommends anyone needing support of newer versions of XSLT move to the new System.XML framework classes.

Microsoft Content Manager, SharePoint and Dynamics CRM are all used to provide Portal services. In this space [27, 28, 29], Microsoft's product range is typically more suitable for mid-size organizations. Microsoft's strategy towards a consistent portal platform is inconsistent [31]. Reporting features have improved with the latest versions of SQL Server but still require experienced developers [33].

Summary
Microsoft .NET offers a single vendor solution to the EAI Platform. It offers a solid selection of products with good support from Third-Party vendors for missing functionality. .NET reaps all the benefits of being strongly interwoven into the operating system. It offers a good solution for web services. On the down side, Microsoft still does not use all the industry standards.

6.3.4 Open Source PSM

The Open Source community has provided a number of solutions that can be utilized within an Enterprise Platform; however, there is not yet a full spectrum solution – commercial products are still required to build large-scale enterprise platforms. This section will introduce a stack of Open Source components that the Enterprise Systems Architecture Laboratory has used to build several enterprise portal platforms for real- world use.

Table 16: Mapping Worksheet for Open Source

PIM Proxy Component	PSM Implementation Component
DIRECTORY SERVICES	LDAP
REPOSITORY SERVICES	MySQL/SAP-DB is currently the most powerful offering (these two database development organizations have merged and SAP-DB is now available as an Open Source product) Other Open Source databases can also be used (including PostgrSQL and Informix)
APPLICATION INTEGRATION	XSLT and JBoss MQ for Message Transformation and Routing
BUSINESS INTEGRATION	Zope as a standalone is a candidate for smaller platforms but is not very scalable yet – for larger enterprise platform applications, Zope needs to be supplemented by a Java planning engine that can make call outs to EJB's The Application Server stack can be built from several components: Apache, JBoss, Jakarta, Ant, and Struts
INTEGRATION SERVICES PROXY	Cocoon and SOAP
AUTHENTICATION SERVICES	LDAP
GATEWAY SERVICES	No realistic Open Source solution currently available - Need to build
ACCESS POINT	No realistic Open Source solution currently available - Need to build
MANAGEMENT SERVICES	No realistic Open Source solution currently available - Need to build
ENTERPRISE DEVELOPMENT ENVIRONEMNT	Eclipse and Eclipse plug-ins

The Enterprise Systems Architecture Laboratory (ESAL) and Auldenfire, LLC have developed a complete framework using the components identified in the table above. The framework includes a number of components that fill the gaps in the PIM-PSM Proxy Component mappings. Additionally, ESAL has developed a real-time enterprise platform framework based on the Open Telephony Platform (OTP) standard Erlang. Erlang was originally developed by Ericsson, who have since placed it into the public domain where a large Open Source community has evolved around it.

The involvement of commercial and government interests into developing Open Source solutions has been especially important. For instance, IBM has made major contributions to Linux and Eclipse; Ericsson to Erlang; SAP to MySQL and SAP-DB. These commercial variants or contributions to Open Source provide solid components that can be used in real-world, mission critical enterprise platforms.

Advantages

Open Source components provide a low-cost, in terms of licensing, solution to an organization that wants to develop an enterprise portal platform. Additionally, the transparency of Open Source development helps an organization evaluate a component from a security perspective. Support from commercial interests is helping many Open Source products achieve high levels of performance, reliability, and scalability – the technical support available commercially also helps eliminate the risk of adopting an Open Source solution.

Disadvantages
Many Open Source solutions today have not reached a level of maturity where they can be used in large enterprise platforms – especially without the use of commercial or special-purpose components to provide critical application and business integration services. Another weakness is the lack of Access Point and management services.

Summary
Many enterprise portal platforms have taken a hybrid approach in adopting a mix of Open Source and commercial components. This trend is continuing as commercial interests such as IBM, Sun, Ericsson, and SAP commit resources to developing more mature Open Source components. The Open Source community itself is also working on large-scale enterprise capable components, which will allow greater adoption in the near future.

6.3.5 Fujitsu Interstage 7 PSM

Worksheet Mapping (Partial) provided by Anatoli Krassavine, Enterprise Architect at Fujitsu Services (London, UK) 2005.

Table 17: Mapping Worksheet for Fujitsu Interstage

PIM Proxy Component	PSM Implementation Component
DIRECTORY SERVICES	Smart Repository (LDAP Compatible Directory)
REPOSITORY SERVICES	Smart Repository (LDAP based)
Business Process Repository	-
XML Repository	XML Support can be provided by Smart Repository
APPLICATION INTEGRATION SERVICES	
Basic Messaging Services	Event Channel (Publish-Subscribe and Point-to-Point) Interstage AS uses CORBA event and notification services only with the JMS service being a wrapper around this functionality ebMS (ebXML Messaging Service builds on SOAP web services message format)
Message Dictionary	CPA Control Function (as a service) – part of ebMS by Proxy - Validates correct format of messages
Data Services	Connector
Data Dictionary	-
Transformation Services	CollaborationRing Format Editor (FEDIT) handles legacy flat files as well as XML types There are additional products and components that provide Transformation Services as well
Transformation and Routing Engine	Traffic Director – used primarily for load balancing (especially in clustered environments)
XML Engine	Uses XSLT library with added high-level support including XLink
Message Warehouse	Can be built around the Interstage Business Process Manager (BPM) or Trading Partner Manager (TPM)
Message Queuing Services (Information Pipe)	Primitive support based around JMS – Implemented by specifying channels and message store rules (maximum duration and maximum messages persisted on queue)
Publish and Subscribe Services	Primitive support based around JMS – Implementation satisfies JMS requirements (fully compliant with J2EE model) supports Normal Subscribers and Durable Subscribers
Connector SDK	CollaborationRing Adapters and Connector SDK (JCA 1.0)
BUSINESS INTEGRATION SERVICES	Interstage Integration Suite
Basic Integration Services	Interstage CollaborationRing (part of Interstage Integration Suite)
Workflow Management Services	Interstage Business Process Manager (Integrated with Interstage) i-Flow – Standalone Java Services (100% Java)
Security Services	Interstage Security Director (part of Interstage Foundation Suite)
Partner Profile Services	Interstage Trading Partner Manager with ebMS
ERP Services	-
EDI Services	CollaborationRing Format Editor (FEDIT)
Content Management Services	Interstage Content Integrator (Contentbiz)
ECM Metadata Repository	-
Decision Support Services	-
Application Services	Interstage Application Server (J2EE compliant)
INTEGRATION SERVICES PROXY	
XML Services	eBMS provides services (provides transformations, message validation, message filtering to decide where messages should be sent) – very fast Fujitsu proprietary XML search engine is included) Includes resilience function to continue trying to deliver messages that failed to be delivered • Interstage Director – Handles XML search requests from applications by passing search requests to Searcher and returning the results to the applications • Interstage Conductor – Allows applications to distribute searches over multiple Directors • Interstage Searcher – Handles and returns searches from Director: also provides high availability, scalability and fault-tolerance
B2B Services	CollaborationRing Format Editor (FEDIT)

AUTHENTICATION SERVICES	
RBAC Repository	Smart Repository (LDAP)
Security Integration Component	Single Sign-On – Limited functionality
Single Sign-On Component (SSO)	Single Sign-On – Authentication server manages RBAC via an LDAP repository service, which manages Application Servers. Flow: User requests authentication Authentication server performs RBAC look-up in repository server and returns credentials User is granted or denied access to Application Server

7 Conclusion

7.1 Discussion

This thesis presents a Domain Neutral Enterprise Architecture Framework (DNEAF) for Enterprise Application Integration (EAI) and Pervasive Platform Services (PPS) which comprise a core concept in the Object Management Group's (OMG) Model Driven Architecture (MDA) standard.

Enterprise Architecture comprises the software and hardware systems that an organization relies on for its day-to-day operations including core business systems, back office systems and, importantly, the platforms that the software applications run on. Enterprise platforms, especially those associated with integration of services, rely on what the MDA defines as a set of essential services that enterprise applications share as a set of common, or pervasive, components, which include Directory Services, Event Handling, Persistence, Transactions and Security.

The processes that support the DNEAF for EAI and PPS include an architecturally driven lifecycle that explicitly relies on elaboration during the analysis and design phases to provide a more detailed model description of the enterprise. This in turn supports both a pragmatic mechanism for developing a top down model of the enterprise landscape and a robust framework for conducting a Transitional and Comparative Gap Analysis (TCGA) process. The TCGA process can significantly aid in managing risk by ensuring a correct implementation process and an enterprise solution that is maintainable over time.

Enterprise scope architectures involve a significantly greater level of complexity than application solution architectures due to the need to integrate business processes across applications. Two critical challenges exist: First, to ensure that a business event has a consistent semantic context across all the enterprise applications – both core business and back office. Second, that the core integration services provided by a platform are both suitable for the enterprise domain and will support future maintainability. Core business applications are responsible for supporting essential business or operational activities and often deal with production, sales, customers and users (either directly or indirectly). Back office software deals with administrative functions such as inventory control, accounting, human resources, customer relationship management and management reporting. Both core business and back office applications require platforms to run on. It is important to understand what constitutes the various levels and characteristics of a given platform since this impacts both the applications and how the applications and data can be integrated.

The strategy behind the DNEAF is to create an abstract services model for Enterprise Application Integration Services (EAIS) and relevant Pervasive Platform Services in the form of an archetype framework that can be used "more or less" as a standard reference for the enterprise from a domain neutral perspective. The resulting model is intended to provide three core benefits:

- Transitional Gap Analysis (TGA) – this is the process of defining two or more baselines over time: A starting baseline and one or more target baselines.
- Comparative Gap Analysis (CGA) – this is the process of comparing different technical implementation approaches to implement a target baseline.
- Reference Archetype Framework (RAF) – this is the process of describing key functionality in the format of an archetype pattern that will "more or less" be required by any enterprise where application and service integration is a characteristic of the enterprise landscape.

The DNEAF for Enterprise Application Integration Services (EAIS) and Pervasive Platform Services (PPS) has been used in a number of large and medium sized enterprises across numerous domains in both the private and public sectors. The DNEAF for EAIS and PPS (hereinafter DNEAF) has been developed with input from the Object Management Group, Ericsson, IBM, Fujitsu Services, Cap Gemini, Microsoft and Oracle. The DNEAF has been successfully applied at Ericsson and in key accounts of the professional services organizations IBM, Fujitsu Services, Cap Gemini and Accenture. Large enterprise applications include several multiyear programs with budgets ranging from USD 250M to USD 2B. Additionally, the DNEAF has been successfully used within Small to Medium Sized (SME) organizations.

The DNEAF has been presented and peer reviewed at the OMG and in a comparative study conducted with the Karolinska University Hospital, the Royal Institute of technology (KTH in Swedish), Stockholm University, the Enterprise Systems Architecture Laboratory (originally part of Auldenfire and now part of Aurenav), IBM, Microsoft and Oracle. Several Master's Thesis have been written on various aspects of the DNEAF and how it was used at the Telemedicine Platform program which was a multiyear applied research project hosted by the Biomedical Engineering Department (MTA in Swedish) at the Karolinska University Hospital in Huddinge.

The Telemedicine Platform program was selected as a proof-of-concept for the DNEAF since all the concerned parties agreed that all the results could be made available in research papers and that the results could be made available under a Creative Commons Licensing model based on Attribution (Individuals and organizations are free to use the DNEAF and to create derivative works but must acknowledge that the DNEAF was used and is provided under license from ESAL). This is also important from an academic perspective since it is important that all the information relating to the research be available for peer review.

An important goal for the DNEAF is to provide a pragmatic framework to help individuals and organizations manage their enterprise architecture.

7.2 Requirement fulfillment

The thesis presents a number of requirements (on Section 1.5) on the DNEAF for EAIS and PPS. In this section the fulfillments of the requirements are discussed, this can be seen as a form of informal argument, and thereby, also be seen as a weak form of evaluation, as discussed by [50].

Requirement: The DNEAF should be generic. That is, the reference model should be domain neutral and suitable for use in any type of enterprise architecture where pervasive services and integration services are important platform components. The framework fulfills this requirement by using an MDA PIM viewpoint to model all the archetype components and subcomponents, thereby insuring domain, technology and vendor neutrality.

Requirement: The framework should be usable as a reference model to describe current and proposed states of an enterprise architecture. The framework supports this requirement through a UML model that can be used to scope required functionality (feedback has indicated that additional sub system UML models and component level worksheets would enhance the fulfillment of this requirement).

Requirement: The framework should be usable to compare different vendor platforms. The framework fulfills this requirement by providing a detailed worksheet that contains domain neutral proxy components for mapping vendor specific implementations of the proxy component functional requirements (this is one of the stronger aspects of the framework today).

Requirement: The framework should support understanding of how systems and services are tied together. The framework supports this requirement by providing a detailed UML model and detailed descriptions that explain each functional area. Additional commentary and illustrations are provided to explain more complex issues and emergent technologies.

Requirement: The framework should be easy to apply. This requirement has been fulfilled by providing both a conceptual model and a linked UML model that has already been populated with detailed text descriptions for each component. The model is currently available for IBM Rational and Sparx Systems Enterprise Architect.

7.3 Future Research

Currently planned future research (following the publication of this paper) will focus on the DSRM framework taking into account some of the ideas expressed by Shlaer and Mellor on application independent architecture [58] and refining Hevner's work [50, 58]. Additional research is also planned for the DNEAF for Security (a separate but related DNEAF) focusing on cloud services as part of ESAL's work on the Open Cloud Computing Integration (OCCI) working group and standard that in turn derives on earlier work by the Open Grid Forum's (OGF) Open Grid Services Architecture (OGSA) working group and published as "The Security Architecture for Open Grid Services" [68]. Both initiatives are expected to impact the DNEAF for EAIS and PPS (especially the Security Services component). There is also scope for extending the granularity of the framework and to provide more current comparative reviews of vendor and open source products and technologies.

References

[1] Ambler, S. W. (1998). Process Patterns: Building large-scale systems using object technology. Cambridge, UK: Cambridge University Press.
Book (ISBN 0-521-64568-9)

[2] Ambler, S. W. (1999). More Process Patterns: Delivering large-scale systems using object technology. Cambridge, UK: Cambridge University Press.
Book (ISBN 0-521-65262-6)

[3] Ambler, S. W. (2001). The Object Primer. Cambridge, UK: Cambridge University Press.
Book (ISBN 0-521-78519-7)

[4] Ambler, S. W. (2002). Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. New York, NY, USA: John Wiley & Sons.
Book (ISBN 0-471-20282-7)

[5] Arlow, J., & Neustadt, I. (2004). Enterprise Patterns and MDA: Building better software with archetype patterns and UML. Boston, MA, USA: Pearson Education.
Book (ISBN 0-321-11230-X)

[6] Coad, P., Lefebvre, E., & De Luca, J. (1999). Java Modeling in Color with UML. Upper Saddle River, NJ, USA: Prentice Hall PTR.
Book (ISBN 0-13-011510-X)

[7] Coad, P., & Mayfield, M. (1999). Java Design (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.
Book (ISBN 0-13-911181-6)

[8] Constantine, L., & Lockwood, L. (1999). Software for Use: A practical guide to the models and methods of usage-centered design. Reading, MA, USA: Addison-Wesley.
Book (ISBN 0-201-92478-1)

[9] Downes, L., & Mui, C. (1998). Unleashing the Killer App: Digital strategies for market dominance. Boston, MA, USA: Harvard Business School Press.
Book (ISBN 0-87584-801-X)
See reference [40] below.

[10] Simmons, D. B., Ellis, N. C., Fujihara, H., & Kuo, W. (1998). Software Measurement: A Visualization Toolkit for Project Control and Process Improvement.
Upper Saddle River, NJ, USA: Prentice Hall PTR.
Book (ISBN 0-13-840695-2)

[11] Yacoub, S. M. & Ammar, H. H. (2004). Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems. Boston, MA, USA:
Pearson Education, Inc.
Book (ISBN 0-201-77640-5)

[12] Model Driven Architecture – Object Management Group White Paper
Richard Soley and the OMG Staff Strategy Group
November 27, 2000 Draft 3.2
http://www.omg.org/mda/mda_files/model_driven_architecture.htm

[13] Examining the Agile Cost of Change Curve
Scott W. Ambler (Ambysoft)
Including excerpts from book "The Object Primer 3rd Edition" [Ref 3]
http://www.agilemodeling.com/essays/costOfChange.htm
Web Resource: Last accessed 2008-06-02

[14] A Design Science Research Methodology for Information Systems Research
Peffers, Ken; Tuunanen, Tuure; Rothenberger, Marcus A.; Chatterjee, Samir
Journal of Management Information Systems
Volume 24, Issue 3, Winter 2007-8, pp 45-78

[15] Miller, Joaquin et al. (2001): *Model Driven Architecture*. Architecture Board ORMSC, OMG. Document number ormsc/2001-07-01
http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

[16] IBM WebSphere Software Platform
http://www-306.ibm.com/software/info/websphere/r/
Web Resource: Last accessed 2004-06-10
http://www.ibm.com/software/websphere

[17] IBM WebSphere Software Portal Platform
http://www-306.ibm.com/software/info/websphere/r/portal/
Web Resource: Last accessed 2004-06-10
http://www-03.ibm.com/software/products/en/portalserver/

[18] WebSphere-World – WebSphere User Community (Not affiliated with IBM)
http://www.websphere-world.com/
Web Resource: Last accessed 2004-06-10
http://www.websphereusergroup.org/

[19] WebSphere Software Global User Group Community
http://www.websphere.org/
Web Resource: Last accessed 2004-06-10
(Link no longer functioning – 2015 June)

[20] Oracle Technology Network
http://otn.oracle.com/products/index.html
Web Resource: Last accessed 2004-06-10
http://www.oracle.com/technetwork/indexes/products/index.html

[21] Microsoft .NET framework page
www.microsoft.com/net
Web Resource: Last accessed 2004-06-10

[22] "Load Balancing COM+"
https://technet.microsoft.com/library/bb734927
http://www.microsoft.com/technet/prodtechnol/acs/evaluate/lb-cmpnt.mspx
Web Resource: Last accessed 2004-06-10

[23] http://archive.infoworld.com/articles/mt/xml/00/09/04/000904mtappctr.xml
Web Resource: Last accessed 2004-06-10
(Link no longer functioning – 2015 June)

[24] Kobielus, James. *What is the BizTalk Philosophy?*
http://www.phptr.com/articles/article.asp?p=21275
Web Resource: Last accessed 2004-06-10
(Link no longer functioning – 2015 June)

[25] BizTalk Pro/Con
http://eai.ittoolbox.com/documents/document.asp?i=2535
Web Resource: Last accessed 2004-06-10
http://eai.ittoolbox.com/documents/biztalk-benefitsweaknesses-18036

[26] Microsoft BizTalk
http://www.microsoft.com/biztalk
Web Resource: Last accessed 2004-06-10

[27] Dragan, Richard. PC Magazine. *Microsoft's Best E-Commerce Solution Yet*.
http://www.pcmag.com/article2/0,4149,425717,00.asp
Web Resource: Last accessed 2004-06-10

[28] Dragan, Richard. PC Magazine. *Publish Web Updates from within Word*.
http://www.pcmag.com/article2/0,1759,716932,00.asp
Web Resource: Last accessed 2004-06-10

[29] Clyman, John. PC Magazine. *Microsoft Exchange Server 2003*.
http://www.pcmag.com/article2/0,1759,1472370,00.asp
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20051110044414/
http://www.pcmag.com/article2/0,1759,1472370,00.asp

[30] Emigh, Jacqueline. *MOM vs. The Giants: Microsoft Struggles for Net Management Supremacy.*
http://networking.earthweb.com/netsysm/article.php/1464371
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20040810172950/
http://networking.earthweb.com/netsysm/article.php/1464371

[31] Roberts-Witt, Sarah. *Digital Doorways*. PC Magazine.
http://www.pcmag.com/print_article/0,1761,a=2318,00.asp
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20040407101844/
http://www.pcmag.com/print_article/0,1761,a=2318,00.asp

[32] Dragan, Richard. *Free Reporting for Microsoft SQL Shops*
http://www.pcmag.com/article2/0,1759,1544745,00.asp
Web Resource: Last accessed 2004-06-10

[33] Microsoft SQL Server 2002
http://www.pcmag.com/article2/0,1759,11798,00.asp
Web Resource: Last accessed 2004-06-10

[34] Dragan, Richard. *Microsoft Windows Server 2003*. PC Magazine.
http://www.pcmag.com/article2/0,1759,1215937,00.asp
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20060105072853/
http://www.pcmag.com/article2/0,1759,1215937,00.asp

[35] Windows Server 2003
http://www.microsoft.com/windowsserver2003
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20080908111919
/http://www.microsoft.com/windowsserver2003/

[36] Dragan, Richard. *Microsoft Visual Studio .NET 2003*. PC Magazine.
http://www.pcmag.com/article2/0,1759,1191817,00.asp
Web Resource: Last accessed 2004-06-10

[37] Parker Shi and Suketu Gandhi. *Enterprise Application Integration*, Volume Number 3.
http://www.diamondcluster.com/ideas/viewpoint/Viewpointv2n3.asp
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20060630122117/
http://diamondcluster.com/ideas/viewpoint/Viewpointv2n3.asp

[38] Brown, Martin. *IIS vs. Apache*. The Server Watch September 9, 2003.
http://www.serverwatch.com/tutorials/article.php/10825_3074841_2
Web Resource: Last accessed 2004-06-10

[39] Slemko, Marc. *"Microsoft Passport to Trouble"*
http://alive.znep.com/~marcs/passport/
Web Resource: Last accessed 2004-06-10
Link no longer working as of 2015 June – Archive link:
https://web.archive.org/web/20060823114349/
http://alive.znep.com/~marcs/passport/

[40] *Unleashing the Killer App – Digital Strategies for Market Dominance*
Downes, Larry and Mui Chunka
Harvard Business School Press, 1998, ISBN 0-87584-X
Updated ISBN: ISBN: 0-87584-801-X

[41] Enterprise Systems Architecture Laboratory (ESAL)
Donald Baldwin (don@aurenav.com)

Forsbackagatan 24
SE 123 43 Farsta
Stockholm, Sweden

Phone: +46 (0)8 604 07 02
Fax: +46 (0)8 604 30 68

[42] Microsoft BizTalk Comparative Matrix
https://www.microsoft.com/en-us/server-cloud/products/biztalk/Comparison.aspx
Web resource last accessed June 2015

[43] Microsoft BizTalk on Technet
https://technet.microsoft.com/en-us/library/4b40220c-ae1e-494e-902a-1b41057661fa
Web resource last accessed June 2015

[44] SSL and TLS support for RedHat LDAP Services
https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/8.2/html/Administration_Guide/Managing_SSL.html
Web resource last accessed June 2015

[45] SSL and TLS support for Microsoft AD
https://msdn.microsoft.com/en-us/library/bb332056.aspx#sdspintro_topic6_binding
Web resource last accessed June 2015

[46] Relative Cost of Correcting a Software Error
Kaplan, C., R. Clark, and V. Trang, *Secrets of Software Quality: 40 Innovations from IBM*, McGraw-Hill, 1995

[47] Spiral Development: Experience, Principles, and Refinements
Boehm, Barry; Edited by Hansen, Wilfred J.
Carnegie Mellon Software Engineering Institute, July 2000

[48] OMG MDA Guide 2.0
OMG Document ormsc/2014-06-01
http://www.omg.org/cgi-bin/doc?ormsc/14-06-01

[49] A design science research methodology for information systems research
Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007).
A design science research methodology for information systems research.
Journal of management information systems, 24(3), 45-77.

[50] A three cycle view of design science research
Hevner, A. R. (2007). A three cycle view of design science research.
Scandinavian journal of information systems, 19(2), 4.

[51] Designerly ways of knowing: Design discipline versus design science
Cross, N. (2001). Designerly ways of knowing: Design discipline versus design science. Design issues, 17(3), 49-55.

[52] The business model ontology: A proposition in a design science approach
Osterwalder, A. (2004).
The business model ontology: A proposition in a design science approach.

[53] Framework for domain-independent archetype modeling
Oglesby, D., Schloegel, K., Bhatt, D., & Engstrom, E. (2002).
U.S. Patent Application 10/242,201.

[54] Essential layers, artifacts, and dependencies of enterprise architecture
Winter, R., & Fischer, R. (2006, October). Essential layers, artifacts, and dependencies of enterprise architecture.
In Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International (pp. 30-30). IEEE.

[55] A Comparison of Enterprise Architecture Frameworks
Urbaczewski, L., & Mrdalj, S. (2006).
A comparison of enterprise architecture frameworks.
Issues in Information Systems, 7(2), 18-23.

[56] Systemic management of architectural decisions in enterprise architecture planning. four dimensions and three abstraction levels
Pulkkinen, M. (2006, January).
Systemic management of architectural decisions in enterprise architecture planning. four dimensions and three abstraction levels.
In System Sciences, 2006. HICSS'06.
Proceedings of the 39th Annual Hawaii International Conference on
(Vol. 8, pp. 179a-179a). IEEE.

[57] Enterprise architecture: The issue of the century
Zachman, J. A. (1997). Enterprise architecture: The issue of the century.
Database Programming and Design, 10(3), 44-53.

[58] Recursive design of an application-independent architecture
Shlaer, S., & Mellor, S. J. (1997).
Recursive design of an application-independent architecture.
IEEE Software, (1), 61-72.

[59] Requirements engineering for pervasive services
Kolos-Mazuryk, L., Poulisse, G. J., & van Eck, P. A. T. (2005).
Requirements engineering for pervasive services.

[60] Perspective: creating a platform-based approach for developing new services
Meyera, M. H., & DeToreb, A. (2001). Perspective: creating a platform-based approach for developing new services.
Journal of Product Innovation Management, 18(3), 188-204.

[61] Object mechanism and method that creates domain-neutral objects with domain-specific run-time extensions in an appropriate collection
Baxter, R. D., Carlson, B. A., & Lee, A. S. (2001).
U.S. Patent No. 6,289,500. Washington, DC: U.S. Patent and Trademark Office.

[62] A component-based policy-neutral architecture for kernel-level access control
Lacoste, M., Jarboui, T., & He, R. (2009).
A component-based policy-neutral architecture for kernel-level access control.
Annals of telecommunications-annales des télécommunications, 64(1-2), 121-146.

[63] Model driven architecture
Soley, R. (2000). Model driven architecture. OMG white paper, 308(308), 5.

[64] S3: A service-oriented reference architecture
Arsanjani, A., Zhang, L. J., Ellis, M., Allam, A., & Channabasavaiah, K. (2007).
S3: A service-oriented reference architecture. IT professional, 9(3), 10-17.

[65] Extracting Domain-Specific and Domain-Neutral Patterns Using Software Stability Concepts
Hamza, H., Mahdy, A., Fayad, M. E., & Cline, M. (2003).
Extracting Domain-Specific and Domain-Neutral Patterns Using Software Stability Concepts. In Object-Oriented Information Systems (pp. 191-201).
Springer Berlin Heidelberg.

[66] Increasing users' trust on personal assistance software using a domain-neutral high-level user model
Nunes, I., Barbosa, S. D., & De Lucena, C. J. (2010).
Increasing users' trust on personal assistance software using a domain-neutral high-level user model. In Leveraging Applications of Formal Methods, Verification, and Validation (pp. 473-487).
Springer Berlin Heidelberg.

[67] Understanding service-oriented architecture
Sprott, D., & Wilkes, L. (2004). Understanding service-oriented architecture.
The Architecture Journal, 1(1), 10-17.

[68] The security architecture for open grid services
Nagaratnam, N., Janson, P., Dayka, J., Nadalin, A., Siebenlist, F., Welch, V., ... & Tuecke, S. (2002).
The security architecture for open grid services.
Open Grid Service Architecture Security Working Group (OGSA-SEC-WG), 1-31.

[69] Canonical Model, Canonical Schema, and Event Driven SOA
Malik, N (2007 June 12). Inside Architecture.
Microsoft Developer – Blog
https://blogs.msdn.microsoft.com/nickmalik/2007/06/12/canonical-model-canonical-schema-and-event-driven-soa/

[70] Unknown author/source – the quoted text/phrase is widely used and found in many publications and presentations without citation.

Notes

1. **Archived web references:** To access the archive web resource, the second URL heeds to be appended to the archive URL (e.g. *both http addresses need to be combined as a single URL link to view the archive*).

2. Text that is in the font **`Courier New`** has been directly quoted (copied) from the indicated source. Definitions and other specific descriptions are copied to preserve meaning.

3. Four important sources contributed to key conceptual developments of the DNEAF for EAIS and PPS:

    a. The OMG MDA standard
    b. Chris de Vaney's early work in Enterprise Platform Integration Services
    c. Peter Coad's introduction of the concept of archetypes
    d. Scott Ambler's work on Agile and Disciplined Agile Development (DAD)