DEVELOPING ENTERPRISE ARCHITECTURES TO ADDRESS THE ENTERPRISE DILEMMA OF DECIDING WHAT SHOULD BE SUSTAINED VERSUS WHAT SHOULD BE CHANGED

by

J. Michael Harrell
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

Dr. Andrew P. Sage, Dissertation Director

Dr. Alexander H. Levis, Committee Member

Dr. Larry Kerschberg, Committee Member

Dr. Kuo-Chu Chang, Committee Member

Dr. Daniel Menascé, Senior Associate Dean

Dr. Lloyd J. Griffiths, Dean, Volgenau School of Engineering

Date: 5 December 2011
Fall Semester 2011
George Mason University
Fairfax, VA
Developing Enterprise Architectures to Address the Enterprise Dilemma of Deciding
What Should Be Sustained Versus What Should Be Changed

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at George Mason University

by

J. Michael Harrell
Master of Science, Computer Science
Old Dominion University, 1986
Bachelor of Science, Zoology
University of Oklahoma, 1974

Director: Andrew P. Sage, University Professor
Volgenau School of Engineering

Fall Semester 2011
George Mason University
Fairfax, VA
DEDICATION

From the moment I first saw this fish in the Scrapbook Desk Accessory in System version .97 on the original 128KB Macintosh, I was wowed. Gone was the status quo of the command line interface that had typified computing and was the ubiquitous interface for the all the mass produced microcomputers prior to the introduction of the Macintosh in 1984.

This work is dedicated to all who struggle to make information technology more than just ASCII* output.

* American Standard Code for Information Interchange
ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my doctoral committee from George Mason University. This dissertation improved greatly with their valuable insights and input. I wish to thank Professor Andrew P. Sage (chair), Professor Alexander H. Levis, Professor Larry Kerschberg, and Professor Kou-Chu Chang for their patience and advice.

This work could not have been completed without the support and assistance of my employer, The Aerospace Corporation. The support and encouragement of John D. Lang, Camille O. Keely and John W. Evans, Ph.D. was very much appreciated throughout my pursuit of this degree.

To all my teachers from elementary school, junior high school, C. E. Byrd Senior High School, the University of Oklahoma, Old Dominion University and lastly George Mason University. Thank you for the gift of knowledge.

Lastly, my love and deep appreciation to my wife, Margaret C. Harrell, Ph.D., my son Clayborne M. Harrell and my daughter M. Thompson (Tommie) Harrell who were my inspiration and whose support, patience and understanding truly made this endeavor possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xii</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>2. BACKGROUND</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Hindsight</td>
<td>7</td>
</tr>
<tr>
<td>2.2 The Impact of Moore’s Law</td>
<td>8</td>
</tr>
<tr>
<td>2.3 The Impact of Metcalfe’s Law</td>
<td>10</td>
</tr>
<tr>
<td>2.4 The Age of Architecture</td>
<td>12</td>
</tr>
<tr>
<td>2.5 The Age of Enterprise Architecture</td>
<td>20</td>
</tr>
<tr>
<td>2.6 Conclusion</td>
<td>23</td>
</tr>
<tr>
<td>3. TECHNICAL PROBLEM STATEMENT</td>
<td>25</td>
</tr>
<tr>
<td>3.1 What Makes an Architecture an Enterprise Architecture</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Why Enterprise Architecture Efforts Fail</td>
<td>27</td>
</tr>
<tr>
<td>3.2.1 Wicked Problems</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Complexity</td>
<td>35</td>
</tr>
<tr>
<td>3.2.3 Enterprise Learning Curve</td>
<td>38</td>
</tr>
<tr>
<td>3.3 Identifying Strategies for a Successful Enterprise Architecture</td>
<td>40</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.4 The Importance of Addressing the Enterprise Dilemma</td>
<td>41</td>
</tr>
<tr>
<td>3.5 Deconstructing the Enterprise Dilemma</td>
<td>43</td>
</tr>
<tr>
<td>3.6 Rationale: Problem Importance</td>
<td>46</td>
</tr>
<tr>
<td>3.7 Hypotheses and Research Questions</td>
<td>49</td>
</tr>
<tr>
<td>3.8 Conclusion</td>
<td>50</td>
</tr>
<tr>
<td>4. RESEARCH APPROACH</td>
<td>52</td>
</tr>
<tr>
<td>4.1 Research Perspective</td>
<td>52</td>
</tr>
<tr>
<td>4.1.1 Research Objective One: Define the Focus and Scope of Enterprise Architecture</td>
<td>55</td>
</tr>
<tr>
<td>4.1.2 Research Objective Two: Develop a Core Set of Enterprise Business Questions</td>
<td>56</td>
</tr>
<tr>
<td>4.1.3 Research Objective Three: Develop an Enterprise Architecture Metamodel</td>
<td>57</td>
</tr>
<tr>
<td>4.1.4 Research Objective Four: Develop a Focusing Methodology for Enterprise Architecture Development</td>
<td>58</td>
</tr>
<tr>
<td>5. RESULTS</td>
<td>59</td>
</tr>
<tr>
<td>5.1 Focusing the Focus</td>
<td>59</td>
</tr>
<tr>
<td>5.2 The Focus and Scope of Enterprise Architecture</td>
<td>65</td>
</tr>
<tr>
<td>5.2.1 Enterprise Architecture Maps the Enterprise Problem Space</td>
<td>70</td>
</tr>
<tr>
<td>5.3 A Core Set of Enterprise Business Questions</td>
<td>74</td>
</tr>
<tr>
<td>5.4 Enterprise Architecture Frameworks</td>
<td>88</td>
</tr>
<tr>
<td>5.4.1 Business Systems Planning</td>
<td>89</td>
</tr>
<tr>
<td>5.4.2 PRISM</td>
<td>92</td>
</tr>
<tr>
<td>5.4.3 Zachman’s Framework</td>
<td>93</td>
</tr>
<tr>
<td>5.4.4 NIST Framework</td>
<td>96</td>
</tr>
<tr>
<td>5.4.5 Enterprise Architecture Planning</td>
<td>98</td>
</tr>
<tr>
<td>5.4.6 TAFIM</td>
<td>98</td>
</tr>
<tr>
<td>5.4.7 TOGAF</td>
<td>99</td>
</tr>
<tr>
<td>5.4.8 C4ISR Architecture Framework</td>
<td>100</td>
</tr>
<tr>
<td>5.4.9 DoDAF</td>
<td>101</td>
</tr>
<tr>
<td>5.4.10 FEA</td>
<td>103</td>
</tr>
<tr>
<td>5.4.11 Current Purpose and Scope of the FEA, TOGAF and DoDAF</td>
<td>105</td>
</tr>
<tr>
<td>5.4.12 Other Frameworks</td>
<td>111</td>
</tr>
<tr>
<td>5.4.13 Conclusion</td>
<td>115</td>
</tr>
</tbody>
</table>
5.5 The MAPPMR Enterprise Architecture Metamodel ............................................. 119
  5.5.1 The MAPPMR Enterprise Architecture Metamodel for Maintain
         Accountability Core Capability ........................................................................ 124
  5.5.2 The MAPPMR Enterprise Architecture Metamodel for Produce Product
         Core Capability ............................................................................................. 127
  5.5.3 The MAPPMR Enterprise Architecture Metamodel for Manage Resources
         Core Capability ............................................................................................. 131
  5.5.4 Conclusion .................................................................................................. 135

5.6 A Focusing Methodology for Enterprise Architecture Development ............... 137
  5.6.1 Philosophy on People and Process .............................................................. 138
  5.6.2 Redefining Enterprise Architecture ........................................................... 141
  5.6.3 Enterprise Architecture Development Process .......................................... 143
  5.6.4 Applying the Focusing Methodology for Enterprise Architecture
       Development .................................................................................................. 147
       5.6.4.1 Before Determining How to Get There, First Determine the
              Destination ................................................................................................. 148
       5.6.4.1.1 Define the Purpose of the Enterprise .................................................. 150
       5.6.4.1.2 Determine the Bounds of the Enterprise ............................................ 152
       5.6.4.1.3 Define the Desired Operating Model(s) of the Enterprise .......... 153
       5.6.4.1.4 Establish Information Technology Principles for the Enterprise ... 155
       5.6.4.1.5 Develop the Core Diagram for the Enterprise .................................. 159
  5.6.5 Develop and Answer Business Questions .................................................... 160
  5.6.6 Iteration ....................................................................................................... 168
  5.6.7 Conclusion ................................................................................................... 169

5.7 Addressing the Enterprise Dilemma ................................................................. 172

6. EVALUATION OF THE RESEARCH ................................................................. 181
  6.1 Case Study Design .......................................................................................... 181

6.2 Enterprise Architecture Case Study Framework .............................................. 186

6.3 Case Study Backgrounds and Descriptions ..................................................... 192
  6.3.1 Case Study One Background and Description ............................................. 192
  6.3.2 Case Study Two Background and Description ............................................. 199
  6.3.3 Case Study Three Background and Description ......................................... 202
  6.3.4 Case Study Four Background and Description ............................................ 204

6.4 Case Study Evaluation ..................................................................................... 206
  6.4.1 Enterprise Purpose Definition .................................................................... 207
  6.4.2 Enterprise Purpose Definition Case Study Key Findings ............................. 208
  6.4.3 Enterprise Boundary Definition ................................................................. 209
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.4 Enterprise Boundary Definition Case Study Key Findings</td>
<td>210</td>
</tr>
<tr>
<td>6.4.5 Enterprise Operating Model(s) Definition, Enterprise Business Process Integration and Enterprise Business Process Standardization</td>
<td>212</td>
</tr>
<tr>
<td>6.4.6 Enterprise Operating Model(s) Definition, Enterprise Business Process Integration and Enterprise Business Process Standardization Case Study Key Findings</td>
<td>214</td>
</tr>
<tr>
<td>6.4.7 Enterprise Information Technology Principles Definition</td>
<td>216</td>
</tr>
<tr>
<td>6.4.8 Enterprise Information Technology Principles Definition Case Study Key Findings</td>
<td>217</td>
</tr>
<tr>
<td>6.4.9 Enterprise Core Diagram Development</td>
<td>218</td>
</tr>
<tr>
<td>6.4.10 Enterprise Core Diagram Development Case Study Key Findings</td>
<td>219</td>
</tr>
<tr>
<td>6.4.11 Enterprise Business Question Development</td>
<td>220</td>
</tr>
<tr>
<td>6.4.12 Enterprise Business Question Development Case Study Key Findings</td>
<td>221</td>
</tr>
<tr>
<td>6.5 Case Study Epilogue</td>
<td>222</td>
</tr>
<tr>
<td>6.5.1 Case Study One Epilogue</td>
<td>222</td>
</tr>
<tr>
<td>6.5.2 Case Study Two Epilogue</td>
<td>223</td>
</tr>
<tr>
<td>6.5.3 Case Study Three Epilogue</td>
<td>223</td>
</tr>
<tr>
<td>6.5.4 Case Study Four Epilogue</td>
<td>223</td>
</tr>
<tr>
<td>6.6 Extending the Enterprise Architecture Case Study Framework</td>
<td>223</td>
</tr>
<tr>
<td>7. CONCLUSIONS</td>
<td>225</td>
</tr>
<tr>
<td>7.1 Research Findings</td>
<td>225</td>
</tr>
<tr>
<td>7.2 Original Contributions</td>
<td>229</td>
</tr>
<tr>
<td>7.3 Recommendations for Further Research</td>
<td>233</td>
</tr>
<tr>
<td>APPENDIX: METAMODEL SPECIFICATION</td>
<td>236</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>257</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>The Martin Two-dimensional Framework for Enterprise Systems Engineering Case Study Research</td>
<td>184</td>
</tr>
<tr>
<td>Table 2</td>
<td>Aligning the Martin Case Study Framework to the Freidman-Sage Case Study Framework</td>
<td>186</td>
</tr>
<tr>
<td>Table 3</td>
<td>Aligning the Enterprise Architecture Case Study Framework to the Martin Case Study Framework and the Freidman-Sage Case Study Framework</td>
<td>188</td>
</tr>
<tr>
<td>Table 4</td>
<td>Enterprise Architecture Case Study Framework</td>
<td>189</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1:</td>
<td>Enterprise Architecture Timeline</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2:</td>
<td>U.S. Department of Defense Briefing on the Business Enterprise Architecture [BEA05]</td>
<td>61</td>
</tr>
<tr>
<td>Figure 3:</td>
<td>Summary of Findings for the Chaos Reports (1998-2009) by the Standish Group</td>
<td>63</td>
</tr>
<tr>
<td>Figure 4:</td>
<td><em>The Enterprise Dilemma</em> is Determining How to Allocate Enterprise Resources</td>
<td>73</td>
</tr>
<tr>
<td>Figure 5:</td>
<td><em>Six-step Problem-Focused Question-Based (PFQB) Process</em> [Martin06]</td>
<td>76</td>
</tr>
<tr>
<td>Figure 6:</td>
<td>Levels of Planning and Control [Kerner79]</td>
<td>90</td>
</tr>
<tr>
<td>Figure 7:</td>
<td>PRISM Framework [Hammer86]</td>
<td>93</td>
</tr>
<tr>
<td>Figure 8:</td>
<td>The Original Zachman Framework for Information Systems Architecture [Zachman87]</td>
<td>94</td>
</tr>
<tr>
<td>Figure 9:</td>
<td>The Expanded Zachman Framework for Information Systems Architecture [Sowa92]</td>
<td>95</td>
</tr>
<tr>
<td>Figure 10:</td>
<td>Levels of Architecture [Fong89: p.138]</td>
<td>96</td>
</tr>
<tr>
<td>Figure 11:</td>
<td>FEAF Levels of Planning and Control [CIO99]</td>
<td>105</td>
</tr>
<tr>
<td>Figure 12:</td>
<td>Capabilities of the Enterprise Across the Enterprise Continuum...</td>
<td>121</td>
</tr>
<tr>
<td>Figure 13:</td>
<td>Topmost Abstraction of the MAPPMR Enterprise Architecture Metamodel</td>
<td>122</td>
</tr>
<tr>
<td>Figure 14:</td>
<td>Topmost Decomposition of the MAPPMR Enterprise Architecture Metamodel</td>
<td>124</td>
</tr>
<tr>
<td>Figure 15:</td>
<td>Metamodel Decomposition of the Maintain Accountability Core Capability</td>
<td>125</td>
</tr>
<tr>
<td>Figure 16:</td>
<td>Metamodel Decomposition of the Produce Product Core Capability</td>
<td>127</td>
</tr>
<tr>
<td>Figure 17:</td>
<td>Metamodel Decomposition of the Manage Resources Core Capability</td>
<td>132</td>
</tr>
<tr>
<td>Figure 18:</td>
<td>Characteristics of Four Operating Models [Ross06: p.29]</td>
<td>154</td>
</tr>
<tr>
<td>Figure 19:</td>
<td>The NASA SMD Enterprise Core Diagram</td>
<td>160</td>
</tr>
<tr>
<td>Figure 20:</td>
<td>The Friedman-Sage Two-dimensional Framework for Systems Engineering Case Study Research [Friedman04]</td>
<td>183</td>
</tr>
<tr>
<td>Figure 21:</td>
<td>Enterprise Business Process Model Framework</td>
<td>195</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Installations &amp; Environment Domain Architecture Framework</td>
<td>197</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Enterprise Business Process Model v2.2</td>
<td>198</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Functional Decomposition of the Enterprise Business Process Model</td>
<td>199</td>
</tr>
<tr>
<td>Figure 25</td>
<td>The Road to Success [Etude13]</td>
<td>228</td>
</tr>
</tbody>
</table>
ABSTRACT

DEVELOPING ENTERPRISE ARCHITECTURES TO ADDRESS THE ENTERPRISE DILEMMA OF DECIDING WHAT SHOULD BE SUSTAINED VERSUS WHAT SHOULD BE CHANGED

J. Michael Harrell, Ph.D.
George Mason University, 2011

Dissertation Director: Dr. Andrew P. Sage

Enterprise architecture is a relatively new concept that arose in the latter half of the twentieth century as a means of managing the information technology resources within the enterprise. Borrowing from the disciplines of brick and mortar architecture, software engineering, software architecture, and systems engineering, the enterprise architecture discipline struggles to define itself and demonstrate that it is a worthwhile endeavor that is capable of producing measurable benefits for the enterprise.

Studies continue to show that most large IT projects are laborious struggles that are perpetually overdue, over-budget and deliver functionality that consistently under performs what was intended. Enterprise architecture is supposed to help lead to order out of the chaos, but the current success rate does not appear to be appreciably higher than it was before enterprise architecture was widely employed.
The key to developing a successful enterprise architecture lies in making it more beneficial as well as less costly. This requires not only knowing how enterprise architecture should be developed, but also knowing what obstacles lay in the path of success. There are many reasons why an enterprise architecture effort can fail, but there are three challenges in particular that obstruct enterprise architecture efforts greatly. These challenges are wicked problems, complexity, and the enterprise learning curve.

The goal of this research has been to discover strategies that can help attenuate the difficulties that result from wicked problems, complexity, and the enterprise learning curve and also to improve the chances of developing an enterprise architecture that delivers a positive return on investment for the enterprise. Towards this goal, this research makes the following contributions:

• It establishes the focus and scope of enterprise architecture by defining the bounds of what enterprise architecture should address.

• It develops a core set of enterprise business questions from which to begin enterprise architecture development.

• It develops an enterprise architecture metamodel named the Maintain Accountability, Produce Product, and Manage Resources enterprise architecture metamodel that supports enterprise architecture metamodel development.
• It develops a methodology that aids the enterprise architect in focusing the development effort on obtaining significant value while reducing the risk of expending resources developing architectural artifacts of little or no value.

• It develops an enterprise architecture case study framework that extends both the Freidman-Sage and Martin case study frameworks for systems engineering case study research, and provides an enterprise architecture case study framework that shares a common basis for evaluating enterprise architecture case studies in the same conceptual framework as prior system engineering case study research.
1. INTRODUCTION

All models are wrong. Some models are useful.
~ George E. P. Box

1.1 Problem Statement

Enterprise architecture is a relatively new concept that has arisen from the ashes of many failed attempts to construct increasingly complex information technology (IT) systems. Borrowing from the disciplines of brick and mortar architecture, software engineering, software architecture, and systems engineering, the enterprise architecture discipline is struggling to define itself and to demonstrate that it is a very worthwhile endeavor that is capable of producing measurable benefits.

While a few prescient individuals such as Vannevar Bush, J. C. R. Licklider, and Martin Greenberger foretold the amazing capabilities that information technology has given us, no one was so prescient as to foresee just how difficult and expensive managing, developing and exploiting information technology would be [Bush45, Licklider60, Licklider65, Licklider68, Greenberger64].

Unfortunately, the art and science of enterprise architecture has not matured to the level where the majority of enterprises that pursue enterprise architecture can routinely expect
a positive return for their efforts. Study after study such as [Standish95, Standish99, INCOSE05] have shown that most large information technology projects are laborious struggles that are perpetually overdue, over-budget and deliver functionality that consistently under-performs. Software architecture first, then systems architecture and now enterprise architecture, were supposed to help lead to order among the chaos. However, Cobb’s Paradox\(^1\) still seems to thrive as the current success rate does not appear to be appreciably higher than it was before software architecture, systems architecture or enterprise architecture were widely employed.

Optimistically, an enterprise architect would hope that there was a compelling purpose for enterprise architecture that differentiates it from the purposes of systems architecture (or any other type of architecture). An enterprise architect would also desire an enterprise architecture framework that was tailored to this unique purpose of enterprise architecture. And lastly, an enterprise architect would hope for an enterprise architecting methodology that is aligned to this enterprise architecture purpose as well as the enterprise architecture framework.

Since none of these have existed, the focus of this research has been to differentiate the purpose of enterprise architecture from systems architecture, not in any effort to supplant systems architecture, but rather to complement and enhance systems architecture. This research also defines the bounds of what enterprise architecture should address, and

\(^1\) Cobb's paradox states “We know why projects fail, we know how to prevent their failure—so why do they still fail?”
develops a framework and methodology that aids the enterprise architect in focusing architectural development efforts on obtaining value quickly, while also reducing the risk of expending resources on efforts that deliver little or no value.

Specifically, this research presents an approach to enterprise architecture that includes the following four research objectives:

• **Research Objective One: a definition for the focus and scope of enterprise architecture.** Enterprise architecture should map the enterprise problem space with the goal of assisting the enterprise address the enterprise dilemma in ways that are consistent with the enterprise’s information technology principles. The key is that enterprise architecture should constrain the set of acceptable solutions without specifying any specific solutions. Solutions are the prescriptions of systems architecture, not the enterprise architecture itself.

• **Research Objective Two: a core set of enterprise business questions.** To be of lasting value, an enterprise architecture must be able to perform some ongoing useful and valuable function. Answering business questions can be the intended use and source of value for an enterprise architecture. When so intended, business questions define the purpose, scope and functionality of the enterprise architecture and are thus the de facto requirements for an enterprise architecture. This research identifies a core set of business questions that can help the fledging enterprise architecture effort begin to develop the more
mature set of business questions the enterprise will want in order to have a more valuable enterprise architecture.

• **Research Objective Three: an enterprise architecture metamodel.** It would be beneficial to have an enterprise architecture metamodel that was aligned with the focus and scope of enterprise architecture identified in research objective one and suited to identifying and answering enterprise business questions starting with the core set of enterprise business questions identified in research objective two. Towards these goals, this research provides the flexible Maintain Accountability, Produce Product, and Manage Resources (MAPPMR) enterprise architecture metamodel that allows the enterprise to determine how big, or how small, the enterprise architecture needs to be. Most architecture frameworks that come from commercial vendors or consortia are based on a rigid underlying metamodel that is incapable of providing any appreciable useful functionality until the architecture is substantially implemented and populated with data. The focus of this effort is to provide value quickly with a modest investment of resources, while ensuring that value that will grow over time.

• **Research Objective Four: a focusing methodology for enterprise architecture development.** All of this work would be of little value if it did not include a methodology, based on enterprise architecting principles, for developing an enterprise architecture that returns value to the enterprise. For
enterprise architecture, simply knowing what must be done is inadequate and will not assure that an enterprise architecture will be successful. Knowing how to define, develop, and deploy the enterprise architecture is necessary to achieve success.

Chapter 2 discusses how the concept of enterprise architecture arose following the emergence of the discipline of systems architecture, but has yet to differentiate itself into a mature discipline that is distinct and different from systems architecture and has a widely accepted methodology governed by industry standards.

Chapter 3 discusses the technical problem statement for this research, to include discussion of what makes an architecture an enterprise architecture. Chapter 3 also discusses what makes enterprise architectures fail with a particular focus on the three biggest challenges in developing a successful enterprise architecture: wicked problems, complexity and the enterprise learning curve. There is also an explanation of the enterprise dilemma which in brief is the challenge of deciding what should be sustained versus what should be changed within the enterprise. Finally, the research hypothesis and research questions are presented.

Chapter 4 presents the research approach and describes the four research objectives:

- To define the focus and scope of enterprise architecture,
- To develop a core set of enterprise business questions,
- To develop an enterprise architecture metamodel,
• To develop a focusing methodology for enterprise architecture development.

Chapter 5 presents the results of this research and describes the outcome of each of the four research objectives. This contribution is compared and contrasted to existing enterprise architecture frameworks highlighting the advantages of this approach.

Chapter 6 presents the evaluation of this research by extending the Martin [Martin06] expansion of the Friedman-Sage two-dimensional framework for systems engineering case study research described in [Friedman04] and describing four case studies in terms of an extended case-study framework. Although the case studies presented are of U.S. federal government enterprises, the methodology is suitable for all enterprises.

Chapter 7 discusses the findings from this research and describes the original contributions of this research. Recommendations for further research are also identified.
2. BACKGROUND

*The essential problem of large systems is maintaining their conceptual integrity.*

~ Fred Brooks  
*The Mythical Man-Month*  
1975

2.1 Hindsight

Architecture concepts have been applied to information technology systems for over four decades. These have been variously referred to as business systems planning, information systems planning, software architecture, software systems architecture, distributed systems architecture, information systems architecture, information technology architecture, enterprise-level architecture, information enterprise architecture, enterprise information technology architecture, organizational architecture, enterprise architecture planning, enterprise information architecture, and enterprise architecture. In the 1950s, the term architecture was applied to describe the hardware structure and configuration of computers and their peripherals [Buchholz62]. The concept of software architecture followed and this was followed by technical architecture, then systems architecture, and finally enterprise architecture, which expanded the scope to include
human, organizational elements, and process, as well as hardware and software considerations.

Paul Clements and Linda Northrop provided excellent background on the software architecture discipline in the technical report “Software Architecture: An Executive Overview” [Clements96a]. Crediting Edsger Dijkstra with first voicing the importance of software structure [Dijkstra69], and crediting David Parnas with his later efforts on information hiding, software structures, and program families described in [Parnas72], Clements and Northrop surmise the reason for architecture as “structure is important, and getting the structure right carries benefits.”

2.2 The Impact of Moore’s Law

An early axiom of the computer industry described in [Dijkstra72] was that hardware preceded software. Often in the early days, as new and more powerful hardware became available, there was a software void whereby the existing software neither took full advantage of new hardware capabilities, nor made effective use of new hardware capabilities. As the capabilities of computing systems continued to grow, it became possible to build even more complex software, which in turn presented even bigger

\[2\] Moore’s Law is not a law, but an observation. In 1965 Gordon Moore, head of research and development at Fairchild Semiconductor and later co-founder of Intel Corporation, observed that the number of circuits per square inch on integrated circuits had doubled every year since the integrated circuit was invented. Moore predicted that this trend would continue for the foreseeable future [~10 years]. In subsequent years, both folklore and a slower pace mutated Moore’s Law into the observation that computing power doubles every eighteen months. Moore revised his own prediction in 1975 to suggest that the numbers of circuits on a chip would double every two years.
software development challenges. Concurrently, the computing systems themselves expanded beyond just a computer, card readers and printers. Many different kinds of peripheral devices, terminals, tape storage, disk storage, printers, displays, etc., became available. Along the way, the word architecture was borrowed to connote the abstract depiction of the configuration of an information technology system.

This software void problem became very obvious after the introduction of the third-generation computing hardware in 1964. This new technology was far more powerful than the second-generation hardware. Literally overnight, complex applications that could not feasibly be implemented on the older generation computers were quite feasible on the new generation computers. Suddenly, it was possible to create software applications that were so large and so complex that no single individual could comprehend the entirety of the application, nor accomplish the programming task alone in a reasonable amount of time. This situation prompted the Science Committee of the North Atlantic Treaty Organization (NATO) to sponsor a conference on this perceived “software crisis” in 1968. During the planning stages for this conference in late 1967,

... the phrase “software engineering” was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering [NATO69].

---

3 First generation mainframe computers were based on vacuum tube technology, the second generation was based on transistor technology, and the third generation was based on integrated circuit technology. The advent of Large Scale Integration (LSI) and Very-large Scale Integration (VLSI) led to fourth and fifth generation labels, but the rapid evolution of the technology described by Moore’s Law resulted in the concept of technological generations falling out of common usage.
By the early 1980s, the term software architecture began to appear [Sandewall81] and DoD had adopted the use of the term architecture to describe its software-centric lifecycle approach for modernizing the World Wide Military Command and Control System (WWMCCS).

Throughout the 1980s, there was a growing acceptance of the premise that some level of abstraction was needed beyond the software design level. Hardware that had evolved at the rate of Moore’s Law had produced personal computers that hardly seemed to labor under the daunting computational tasks that just a few years prior brought supercomputers to a crawl. Clearly, the advances of information technology had grown at a staggering pace. However, the impact of Moore’s Law has been minor compared to the upcoming impact of Metcalfe’s Law.

2.3 The Impact of Metcalfe’s Law

According to Metcalfe’s Law, the power of the network is proportional to the square of the number of nodes in the network. For information networks, bigger, in terms of the number of nodes, is not just better; bigger is way better. “The ‘power’ or ‘payoff’ … comes from information-intensive interactions between very large numbers of

---

4 Like Moore’s Law, Metcalfe’s Law is not really a law, but a conjecture. Robert Metcalfe, the co-inventor of the Ethernet protocol and founder of the networking company 3Com, first suggested the idea around 1980, but it was journalist George Gilder who called it a law. Gilder strongly maintained that Metcalfe's Law would amplify Moore's Law and, in so doing, would remake the world.
heterogeneous computational nodes in the network” according to a well quoted statement [Cebrowski98].

In 1964, Martin Greenberger wrote “The Computers of Tomorrow” for *The Atlantic Monthly*. This was an article that forecasted the evolution of an “information utility” that bears a remarkable likeness to today’s Internet [Greenberger64]. This stunningly prescient prediction came at a time when Greenberger estimated that there were probably only about 20,000\(^5\) computers in use in the United States and that none of those computers was networked. It would be five years after Greenberger’s prediction before the Internet had its humble beginnings as the ARPANET on 29 October 1969 with just two hosts. A decade later in October 1979 the ARPANET had only grown to 188 hosts. By October 1989, the ARPANET had surged to 159,000 hosts, but there were as yet no commercial Internet service providers. By October 1990, when the ARPANET became the Internet and became accessible to the public, it had ballooned to 313,000 hosts. By July 1999, when the Internet was approaching its thirtieth anniversary, the Internet had exploded to 56.2 million hosts, and by July 2005, the Internet had an astonishing 353.28 million hosts [Hobbes08]. An accurate estimate of the number of actual computers connected to the Internet is difficult to obtain since each host can provide access for a number of computers. Nonetheless, it is notable that by the third quarter of 1999, the computer industry had attained worldwide sales in excess of 12,000 computers per hour with the least capable PC handily outperforming the most capable 1964 era super-

\(^5\) Other estimates place the number at approximately 24,000.
computer. By April 2002, the billionth personal computer had been sold. As visionary as Greenberger was in 1964, his predictions vastly underestimated reality.

2.4 The Age of Architecture

Harnessing the exponential combinatoric power of Moore’s and Metcalfe’s Laws requires more than just hardware and networks. The key, as noted by George Gilder [Gilder93], lies in exploiting a set of information-intensive interactions between very large numbers of heterogeneous computational nodes. But choosing, designing, building and evolving a set of “information-intensive interactions” that can keep up with computing technology has been no easy task. The combined effect of Moore’s and Metcalfe’s Laws truly created opportunities that had never before existed.

By the late 1960s IBM perceived the need for something similar to enterprise architecture [Kerner79]. IBM’s work with Business Systems Planning [IBM’s Business Systems Planning (BSP) approach (where John Zachman cut his teeth) goes back to the late-1960s and may be considered the beginning of enterprise architecture. IBM used BSP internally at first before offering it as a commercial service in 1981.]

---

6 In 1964, the newly introduced and Seymour Cray designed Control Data Corporation CDC 6600 supercomputer ran at speeds up to nine megaflops (nine million floating-point operations per second). It went on to become the first commercially successful supercomputer selling for millions of dollars and holding the title of the world’s fastest computer from its introduction in 1964 until it was replaced in 1968 by the CDC 7600 that ran at up to forty megaflops. Today, the iPhone 4S is more than fifteen times more powerful than the CDC 6600 being clocked at 138.2 megaflops and weighing in at just 4.9 ounces.

7 IBM’s Business Systems Planning (BSP) approach (where John Zachman cut his teeth) goes back to the late-1960s and may be considered the beginning of enterprise architecture. IBM used BSP internally at first before offering it as a commercial service in 1981.
communications technologies were beginning to overlap. In a three-part series of articles entitled “The Information Archipelago” which appeared in the *Harvard Business Review*, the authors James McKenney, F. Warren McFarlan and Philip Pyburn recognized this convergence and espoused the growing need to oversee the acquisition, sustainment and evolution of information technologies within the organization [McKenney82, McFarlan83a, McFarlan83b].

By 1986, when PRISM[^8] published “Dispersion and Interconnection: Approaches to Distributed Systems Architecture,” there was already a compelling perception in many companies that there existed “problems which can only be addressed by the development of an architecture” [Hammer86]. In a confounding manner, the PRISM report also revealed that “No two I/S[^9] organizations mean the same thing by architecture when they express the above belief.” PRISM reconciled these varying perceptions into the first description of an enterprise architecture framework and methodology that has for the most part gone unnoticed with virtually every author citing John Zachman as being the first to develop an enterprise architecture framework[^10]. Granted, PRISM did not label their effort an enterprise architecture, or an enterprise architecture framework, but at the time neither did John Zachman. This came later.

[^8]: PRISM (Partnership for Research in Information Systems Management) was a joint venture of the Index Group and Hammer and Company. Index Group was originally Index Systems. Index Group was acquired by Computer Sciences Corporation (CSC) in 1989, was renamed CSC Index, and grew to over 650 employees in fourteen offices before being dissolved in 1999. Hammer and Company continues to operate.
[^9]: Information Systems.
[^10]: According to John Zachman’s website (http://www.zachmaninternational.com/) he had first developed his framework as early as 1984, but did not publish it until 1987.

> Although from the outset, it was clear that it [Zachman’s “Framework for Information Systems Architecture”] should have been referred to as a “Framework for Enterprise Architecture,” that enlarged perspective could only now begin to be generally understood as a result of the relatively recent and increased, world-wide focus on Enterprise “engineering”.

But even if Zachman waited ten years to re-name his framework to put the emphasis on the enterprise, he had used the term fifteen years earlier in [Zachman82] thereby marking the first published use of the phrase that this research has uncovered. Zachman was also a participant in the National Institute of Standards and Technology (NIST) workshop on Information Management Directions: The Integration Challenge during the fall of 1988, which included a chapter on enterprise architecture.

In 1989, NIST published the proceedings of this three-day workshop in [Fong89]. John Zachman, who was still with IBM at the time, participated in this workshop [Sessions07] and authored Chapter 5 of the proceedings [Fong89: pp.63-122] consisting of a detailed description of his “Framework for Information Systems Architecture.” Coincidentally, Chapter 7 of these proceedings [Fong89: pp.135-150], authored by W. Bradford Rigdon,
includes an enterprise architecture model published as Figure 7-1 in [Fong89: p.138]\(^{11}\)
that bares a remarkable resemblance to the graphical depiction of “Levels of Planning and Control” published as Figure 1 in [Kerner79]\(^{12}\) and adheres to the same exploit of increasing levels of abstraction from bottom to top as does Zachman’s framework.

Also in 1989, the Object Management Group was founded and began work specifying the Common Object Request Broker Architecture (CORBA). That same year, Dewayne Perry and Alexander Wolf published “Software Architecture”\(^{13}\) [Perry89]. By 1991, the Defense Advanced Research Projects Agency (DARPA) had initiated a project on Domain Specific Software Architectures [Mettala92]. In 1992, Sowa and Zachman extended the original version of the Zachman “Framework for Information Systems Architecture” and while the word enterprise only appeared twice in [Zachman87], it appeared over 40 times in Sowa’s and Zachman’s 1992 paper [Sowa92] although the expression “enterprise architecture,” which Zachman had used in 1982 in [Zachman82], was not used in either of the later papers.

In 1992, Steven Spewak published, *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology*, and advocated the data-centric approach that has formed the basis for Service-Oriented Architecture that is embraced by many today [Spewak92].

\(^{11}\) Shown in Figure 10 below.

\(^{12}\) Shown in Figure 6 below.

\(^{13}\) Published as “Foundations for the Study of Software Architecture” in 1992.
From 1992 onward, the architecture discipline began to grow, gaining frameworks, methodologies, abstractions, model representations, definition languages, description languages, tools and styles. This rapidly growing interest in architecture is reflective of the widespread proliferation of relatively cheap networks as the Internet was opened to the public in 1990 and as the World Wide Web emerged in 1991. Both developments made it possible for systems to be accessible across the enterprise and around the globe. In effect, stovepiped systems were no longer mandated by the limitations of the technology.

DoD, having some of the most daunting information management challenges, also recognized, like Dijkstra and Parnas before, that “structure is important, and getting the structure right carries benefits.” In 1994, two Defense Science Board reports were published that highlighted the need for architecture within DoD. The first report in [DSB94a] articulated the need for software architecture and stated:

Software architecture consists of:

- Software system components
- The relationships among those components
- Rules for their composition (constraints)

The second report in [DSB94b] highlighted “the need for a joint enterprise architectures framework” and described three views of information architecture that would evolve into the Operational Architecture—System Architecture—Technical Architecture framework.
later prescribed by DoD’s C4ISR\textsuperscript{14} Architecture Framework. Inspired by these two Defense Science Board reports, DoD recognized the need for a common architectural approach (at least for C4ISR systems) and:

\textit{In October 1995, the Deputy Secretary of Defense directed that a DoD-wide effort be undertaken to define and develop better means and processes for ensuring that Command, Control, Communications, Computers, and Intelligence capabilities meet the needs of warfighters. In response to that direction, a C4ISR Integration Task Force (ITF) was established under the direction of the Assistant Secretary of Defense for Command, Control, Communications, and Intelligence (ASD[C3I]). The C4ISR Architecture Framework, Version 1.0, dated 7 June 1996, was developed as a product of the Integrated Architectures Panel (IAP), one of several panels established by the ITF [DoDAF04].}

Concurrently, the Software Engineering Institute (SEI)\textsuperscript{15} began publishing the results of its research into software architecture [Garlan95, Magee95, Shaw95a, Shaw95b, Shaw95c, Clements96a, Clements96b, Shaw96, Allen97a, Allen97b, Shaw97].

On 16 July 1996, President Clinton signed Executive Order 13011 titled “Federal Information Technology.” This executive order required federal agencies to appoint a Chief Information Officer (CIO) and created the Federal CIO Council. Also in 1996, the U.S. Congress, ever mindful of all the money spent by the U.S. Government on information technology, passed the Information Technology Management Reform Act (ITMRA) and the Federal Acquisition Reform Act that together became known as the

\textsuperscript{14} C4ISR is the acronym for Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance.

\textsuperscript{15} SEI is a Federally Funded Research and Development Center (FFRDC) sponsored primarily by DoD. SEI was established in 1984 at Carnegie Mellon University in Pittsburgh.
Clinger-Cohen Act (CCA). This legislation became effective on 8 August 1996 and states in part:

... ‘information technology architecture’, with respect to an executive agency, means an integrated framework for evolving or maintaining existing information technology and acquiring new information technology to achieve the agency’s strategic goals and information resources management goals.

The CCA also assigned all U.S. Federal Government Chief Information Officers with the responsibility for “developing, maintaining, and facilitating the implementation of a sound and integrated information technology architecture.”

1997 was the birth-year for enterprise architecture within the U.S. Government. On 16 June 1997, the Office of Management and Budget (OMB) issued a memorandum for the heads of executive departments and agencies of the U.S. Government with the subject of “Information Technology Architectures” [OMB97] that stated:

This memorandum transmits guidance to Federal agencies on the development and implementation of Information Technology Architectures. The Information Technology Architecture (ITA) describes the relationships among the work the agency does, the information the agency uses, and the information technology that the agency needs. It includes standards that guide the design of new systems. An ITA makes it easier to share information internally (e.g., agency-wide e-mail) and to reduce the number of information systems that perform similar functions. The ITA provides the technology vision to guide resource decisions that reduce costs and improve mission performance.

In this memo, the OMB broadened the scope of the CCA by specifying that an ITA must also include an enterprise architecture. This is the first occurrence of enterprise architecture being required by the U.S. Government for U.S. Government departments and agencies:
The ITA is broad in scope and includes processes and products. An architecture in compliance with the Clinger-Cohen Act and OMB guidance will contain two elements:

- the Enterprise Architecture,

This memo [OMB97] goes on to define enterprise architecture and references the published proceedings of the NIST three-day workshop on Information Management Directions: The Integration Challenge published in [Fong89] as follows:

The Enterprise Architecture is the explicit description of the current and desired relationships among business and management process and information technology. It describes the “target” situation which the agency wishes to create and maintain by managing its IT portfolio.

The documentation of the Enterprise Architecture should include a discussion of principles and goals. For example, the agency’s overall management environment, including the balance between centralization and decentralization and the pace of change within the agency, should be clearly understood when developing the Enterprise Architecture. Within that environment, principles and goals set direction on such issues as the promotion of interoperability, open systems, public access, end-user satisfaction, and security.

This guidance adapts a five component model used in the National Institute of Standards and Technology (NIST) Special Publication 500-167, “Information Management Directions: The Integration Challenge.” Agencies are permitted to identify different components as appropriate and to specify the organizational level at which specific aspects of the components will be implemented. Although the substance of these components, sometimes called “architectures” or “sub-architectures,” must be addressed in every agency’s complete Enterprise Architecture, agencies have great flexibility in describing, combining, and renaming the components, which consist of:

- Business Processes
- Information Flows and Relationships
- Applications
- Data Descriptions
- Technology Infrastructure
On 18 December 1997, version 2.0 of DoD’s C4ISR Architecture Framework was approved and in February 1998, the DoD mandated the use of the C4ISR Architecture Framework version 2.0 for all DoD architecture descriptions (not just C4ISR systems). Architecture was now required throughout DoD. In this document, DoD defined architecture as follows:

The IEEE STD 610.12, as extended slightly by the IAP of the ITF, defines “architecture” as “the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.”

2.5 The Age of Enterprise Architecture

In 1997, interest in enterprise architecture was growing in the commercial sector, as well. John Zachman proclaimed, “The age of enterprise architecture is here” and went so far as to proclaim enterprise architecture the “issue of the century” [Zachman97b]. The U.S. Government was aligned with John Zachman in its thinking. The January 1998 CIO Council Strategic Plan, guided by priorities of the Clinger-Cohen Act of 1996, directed the development and maintenance of a Federal Enterprise Architecture to maximize the benefits of information technology within the government.

On 28 November 2000, OMB Circular A-130, “Management of Federal Information Resources” [OMB00] was revised and reissued incorporating language similar to the 1997 OMB memorandum [OMB97]. The revised Circular A-130 described enterprise architecture as follows:
An EA is the explicit description and documentation of the current and desired relationships among business and management processes and information technology. It describes the “current architecture” and “target architecture” to include the rules and standards and systems life cycle information to optimize and maintain the environment which the agency wishes to create and maintain by managing its IT portfolio. The EA must also provide a strategy that will enable the agency to support its current state and also act as the roadmap for transition to its target environment. These transition processes will include an agency’s capital planning and investment control processes, agency EA planning processes, and agency systems life cycle methodologies. The EA will define principles and goals and set direction on such issues as the promotion of interoperability, open systems, public access, compliance with GPEA, end user satisfaction, and IT security. The agency must support the EA with a complete inventory of agency information resources, including personnel, equipment, and funds devoted to information resources management and information technology, at an appropriate level of detail. Agencies must implement the EA consistent with following principles:

(i) Develop information systems that facilitate interoperability, application portability, and scalability of electronic applications across networks of heterogeneous hardware, software, and telecommunications platforms;

(ii) Meet information technology needs through cost effective intra-agency and interagency sharing, before acquiring new information technology resources; and

(iii) Establish a level of security for all information systems that is commensurate to the risk and magnitude of the harm resulting from the loss, misuse, unauthorized access to, or modification of the information stored or flowing through these systems.

OMB Circular A-130 [OMB00] goes on to prescribe the following approach describing how agencies should create and maintain enterprise architectures:

As part of the EA effort, agencies must use or create an Enterprise Architecture Framework. The Framework must document linkages between mission needs, information content, and information technology capabilities. The Framework must also guide both strategic and operational IRM planning.

Once a framework is established, an agency must create the EA. In the creation of an EA, agencies must identify and document:

(i) Business Processes - Agencies must identify the work performed to support its mission, vision and performance goals. Agencies must also document change agents, such as legislation or new technologies that will drive changes in the EA.
“(ii) Information Flow and Relationships - Agencies must analyze the information utilized by the agency in its business processes, identifying the information used and the movement of the information. These information flows indicate where the information is needed and how the information is shared to support mission functions.

(iii) Applications - Agencies must identify, define, and organize the activities that capture, manipulate, and manage the business information to support business processes. The EA also describes the logical dependencies and relationships among business activities.

(iv) Data Descriptions and Relationships - Agencies must identify how data is created, maintained, accessed, and used. At a high level, agencies must define the data and describe the relationships among data elements used in the agency’s information systems.

(v) Technology Infrastructure - Agencies must describe and identify the functional characteristics, capabilities, and interconnections of the hardware, software, and telecommunications.

The EGovernment Act of 2002 (Public Law (PL) 107-347) [EGov02] codified what was previously outlined in OMB Circular A-130, required the development of enterprise architectures, and described enterprise architecture as follows:

“(4) ‘enterprise architecture’—

“(A) means—

“(i) a strategic information asset base, which defines the mission;

“(ii) the information necessary to perform the mission;

“(iii) the technologies necessary to perform the mission; and

“(iv) the transitional processes for implementing new technologies in response to changing mission needs; and

“(B) includes—

“(i) a baseline architecture;

“(ii) a target architecture; and

“(iii) a sequencing plan;
Konstantin Beznosov provides an excellent discussion on the variety of definitions for enterprise architecture in his technical report on *Information Enterprise Architectures: Problems and Perspectives* [Beznosov00], as does the draft *Enterprise Architecture Body of Knowledge* (EABOK) [Hagan04]. The EABOK concludes:

"Today, there is general agreement that, at a minimum, the EA includes models of business practices or processes, data, computing systems for mission-related and business support, networks and other technology infrastructure, for both the baseline, or current, and target architectures; several source [sic] also include the organization structure. The EA includes a standards profile, security considerations, and a sequencing plan, sometimes called a transition plan; is linked to agency strategic plans; and is a major basis for investment decision. ..."

The agency achieves its target architecture by:

- Managing its investment portfolio to invest in projects that implement the agency strategy in accordance with the sequencing plan,
- Enforcing the EA through its Configuration Management (CM) boards,
- Managing the delivery of capabilities identified in the EA through its Systems/Software Development Life Cycle (SDLC) and Capital Planning and Investment Control (CPIC) processes, and
- Evaluating the result of its delivered capabilities through its CPIC processes.

The agency continually updates its EA through its EA planning processes.

## 2.6 Conclusion

Clearly, the intent of architecture has been to address “the essential problem of large systems” by focusing on “maintaining their conceptual integrity” [Brooks75]. Although it may be encouraging that there is a growing agreement on what architecture means [Hagan04], it is discouraging to note that few organizations employ the same methodologies to develop, “models of business practices or processes.” In the various
architecture definitions noted above, one cannot readily discern the difference between
the purpose of an enterprise architecture and the purpose of any other information
systems architecture. Any architecture should have “models of business practices or
processes, data, computing systems for mission-related and business support, networks
and other technology infrastructure, for both the baseline, or current, and target
architectures” [Hagan04] as well as the organizational structure. One would expect that
the purpose served by enterprise architecture would be different from systems
architecture. However, from the definitions, it is not clear what those differences in
purpose might be.
3. TECHNICAL PROBLEM STATEMENT

enterprise e.nty=rpryiz, sb...

1. A design of which the execution is attempted; a piece of work taken in hand, an undertaking; chiefly, and now exclusively, a bold, arduous, or momentous undertaking.

~ Oxford English Dictionary, 2nd Edition

3.1 What Makes an Architecture an Enterprise Architecture

The concept of enterprise architecture arose as technologies emerged which made it possible to interconnect all the information technology within an enterprise (see Figure 1). Before these internetworking technologies emerged, systems were stovepiped by necessity. Any connection to another system or component had to be specifically designed and developed. However, once internetworking technologies became widely available and economically feasible, suddenly, stovepiped systems were no longer necessitated by the constraints of technology and economy. If anything can connect to everything, then the range of enterprise solutions becomes vast and there arises an overwhelming need to manage that solution space. Unfortunately, there is little in the numerous definitions of enterprise architecture that helps clarify what differentiates enterprise architecture from any other type of architecture. One could arrive at the
conclusion that “enterprise IT architecture” is the integration of all the “system IT architectures.” However, one could also surmise that enterprise architecture may serve a different purpose than systems architecture. If the fundamental structure of architecture is the same, then the distinguishing feature that differentiates enterprise architecture from any other type of architecture is the problem that the enterprise architecture is attempting to address. This research asserts that the focus and scope of any enterprise architecture effort should be on addressing the enterprise dilemma. This establishes the basis for the conceptual integrity of an enterprise architecture. The enterprise dilemma will be explained in Section 3.4 below.

1968: Edsger Dijkstra’s “Go To Statement Considered Harmful.” NATO Science Committee conference on “software crisis.”
1969: ARPA network born on 29 October. It would be 21 years before ARPANET became the Internet.
1972: David Parnas’ “On the Criteria to Be Used in Decomposing Systems into Modules.”
1973: Ethernet invented. It would be 21 years before Ethernet came standard on PCs.
1985: Mill’s “A Pragmatic View of the System Architect.”
1986: Hammer, Champy & Davenport describe the first enterprise architecture framework in “Dispersion and Interconnection: Approaches to Distributed Systems Architecture.”
1990: Internet opened to public, 1st commercial ISP. NCSC founded.
1993: TAFIM v1.0. Mosaic, the 1st web browser introduced.
1995: TOGAF 1.0.
1998: DoD mandated C4ISR AF. TOGAF 4.0.
1999: TOGAF 5.0.
2001: TOGAF 7.0.
2004: DoDv1.0.
2009: DoDv2.0. TOGAF 9.0.

Figure 1: Enterprise Architecture Timeline
3.2 Why Enterprise Architecture Efforts Fail

Both systems architecture and software architecture had been co-evolving for some time before the concept of enterprise architecture emerged with the 1982 appearance of [Zachman82] and the first of a three article series by McKenney, McFarlan and Pyburn in the *Harvard Business Review* on the Information Archipelago [McKenney82, McFarlan83a, McFarlan83b]. Of course, the concept was not yet labeled enterprise architecture, but the perspective was much more encompassing than just systems architecture or software architecture. In 1986, the PRISM Model laid out the first enterprise architecture framework [Hammer86]. From this point, software architecture, systems architecture and enterprise architecture have co-existed and co-evolved.

By the 1980s, a major software architecture focus was on platform independence (a.k.a., software portability), principally in the form of the Portable Operating System Interface (POSIX) standard described in [IEEE1003-88, IEEE1003-95, ISO14252-96]. The goal was to standardize the UNIX operating system so that software applications could be written to a standard specification and then compiled and run on any machine that ran a standards compliant version of the UNIX operating system with little or no additional effort. This work grew to include standards and standards-based technical architectures that defined primarily the APIs (application programming interfaces), protocols and standard applications that could plug-and-play in a POSIX (and later, Microsoft Windows) compliant environment [IEEE1003-88, TAFIM93, TAFIM94, IEEE1003-95, TOGAF95, ISO9945-96, JTA1-96 TAFIM96, TOGAF96, TOGAF97, JTA2-98,
The POSIX focus was primarily on interoperability and interfacing software to other software. Additional architectural methodologies were also developed specifically for software architecture. These included RUP (Rational Unified Process), Rational 4+1 View Model, RM/ODP (Reference Model for Open Distributed Processing, later adopted as ISO/IEC 10746-3:1996 [Kruchthen95, Ellis96, ISO10746-96]). These methodologies are focused on architecting the functionality required/desired as a final outcome by an application user.

Systems architecture frameworks began to emerge in the mid 1990s and have continued to evolve [Maier96, C4ISR96, C4ISR97, Sage98, TOGAF99, TOGAF00, TOGAF01, TOGAF02, DoDAF04]. Descriptions of architecture development processes or methodologies were developed to support these frameworks [Hammer86, Bienvenu00, Levis00, Wagenhals00, Levis09]. Of note is that some of these frameworks have evolved from standards-based technical architectures, and the more recent versions of these systems architecture frameworks have been extended to include more enterprise architecture like features within these frameworks [MODAF05, DoDAF07, MODAF07, TOGAF09, DoDAF09].

Enterprise architecture research has also resulted in frameworks. In 1986, PRISM introduced the first enterprise architecture framework and methodology\textsuperscript{16} in [Hammer86]. Prior to the late 1990s, enterprise architecture was typically called systems architecture,

\textsuperscript{16} PRISM called it “distributed systems architecture” at the time.
information architecture, or information systems architecture. Only later was the phrase enterprise architecture or enterprise IT architecture applied. Since PRISM first introduced the concept in [Hammer86], there has been a growing body of published works on enterprise architecture frameworks. Some enterprise architecture frameworks and methodologies clearly evolved from software architecture and systems architecture beginnings, while others emerged from fresh perspectives [Hammer86, Zachman87, Sowa92, Spewak92, Henderson93, Luftman93, Cook96, Boar98, Armour99a, Armour99b, CIO99, TEAFO00, Armour01, Carlock01, Sage01, Morganwalp02, Sharum02, McGovern03, Perks03, TOGAF03, Bernard04, Schekkerman04, Theuerkorn05, Vasconcelos04, Whittle04, Lapkin05, Rouse05a, Rouse05b, Wagter05, Blevins06, Carlock06, Martin06, Ross06, Emery07, Sage07, Stephenson07a, Stephenson07b, TOGAF07, Sessions08].

Since first advocated by Zachman in [Zachman82] and McKenney, McFarlan and Pyburn, in [McKenney82, McFarlan83a, McFarlan83b] the need for an enterprise perspective on information technology has become almost universally accepted. Along the way, a growing body of knowledge [Hammer86, Brancheau89, Weill04, Kaisler05, Ross06, Sessions08] had shown that even though the concept is widely advocated as being beneficial, successfully building and exploiting architecture is usually an overwhelming process with a staggering failure rate. The principle determinant for the failure of an architecture effort is that architecture is very expensive. In 1995, Tom DeMarco noted [DeMarco95]:

29
The sobering fact is that a decent architecture costs as much as two full implementations. Your very first implementation will cost you about three times more with an architecture, compared to throwing the product together without paying for a single architectural thought. What we’re talking here is real money.\textsuperscript{17}

DeMarco went on to observe that most attempts at architecture are not credible:

\textit{In my consulting practice I see one organization after another engender an architecture group because they realize that architecture is the key to product family reuse, and thus a desirable goal. But then the architecture group is funded only as a part of the first implementation project, and that project’s budget is set in the usual fashion for our industry, i.e., in the mid-range between Impossible and Highly Unlikely. In other words, the architecture group is a sham. It has a charter, but no funding. After a momentary digression on reuse, it is treated as an adjunct to the project and driven by the same dynamic as all the rest of the project, getting the product out the door as quickly and cheaply as possible. It is no surprise that no useful architecture ever comes out of such a group. Organizations that pay the cost of architecture get what they pay for and organizations that don’t don’t get anything at all. My experience is that the enormous majority of architectural efforts in our industry are in this latter category [DeMarco95].}

DeMarco’s realization results in a systems architecture for a single system or product and not an enterprise architecture that attempts to describe an entire enterprise. With the assumption that enterprise architecture is the collective and integrated set of systems architectures for the enterprise, enterprise architecture then becomes unaffordable when applying DeMarco’s cost estimate across the enterprise, and there is a high likelihood that the enterprise architecture will not be able to demonstrate any positive return on investment. The empirical research of Ross \textit{et al.} [Ross06] indicates that enterprises develop their architectural competence very slowly and as a consequence the success rate

\textsuperscript{17} In practice, it is difficult to rationalize a 200% cost increase for an architecture-based development process.

30
for enterprise architecture efforts is often much lower than expected. Some enterprises greatly scale back or abandoned their enterprise architecture efforts while others seek alternative enterprise architecting strategies because the ones they have employed previously have not delivered.

The major key to increasing the probability of success for building a meaningful enterprise architecture lies in making it more beneficial while simultaneously making it less costly. This requires not only knowing how architecture must be developed, but also knowing what obstacles lay in the path of success.\textsuperscript{18} There are many reasons why an enterprise architecture effort can fail and this research can neither identify, nor prescribe a solution for all of those reasons. However, this research does address the three challenges that obstruct all enterprise architecture efforts: wicked problems, complexity, and the enterprise learning curve, and this discussion is within the context of striving to address the enterprise dilemma.

### 3.2.1 Wicked Problems

Horst Rittel and Melvin Webber described and contrasted what they called “wicked” and “tame” problems over thirty-five years ago [Rittel72, Rittel73\textsuperscript{19}]. A wicked problem is in

\textsuperscript{18} Success, for the purpose of this discussion, is determined by having an architecture that saves the enterprise more money than the architecture costs, i.e., has a positive return on investment.

\textsuperscript{19} Rittel’s and Webber’s paper, “Dilemmas in a General Theory of Planning.” [Rittel73] is a modified version of a paper presented in 1969 which pre-dates Rittel’s description of wicked problems described in [Rittel72].
a class of problems that, despite all the best systems analyses, always seems to be unsolvable by traditional linear analytical approaches. Conversely,

...a ‘tame’ problem can be exhaustively formulated so that it can be written down on a piece of paper which can be handed to a knowledgeable man who will eventually solve the problem without needing any additional information.

Rittel and Webber [Rittel72, Rittel73] posited eleven\(^{20}\) distinguishing properties of wicked problems:

1. There is no definitive formulation of a wicked problem. No one can write down an exhaustively formulated problem description that can then be solved without further interactions between the problem framers and the problem solvers. This is because the information needed to understand the problem depends upon one’s idea for solving it. Each potential solution to a sub-set of the wicked problem is dependent on the solution to one or more other subsets of the wicked problem.

2. Every formulation of the wicked problem corresponds to a statement of the solution and vice versa. “This means that understanding the problem is identical with solving the problem,” and each attempt at creating a solution changes the understanding of the problem.

3. There is no stopping rule for wicked problems. There is no obvious endpoint. Efforts to solve wicked problems end when they “run out of time,

\(^{20}\) [Rittel72] listed eleven distinguishing properties of Wicked Problems, while [Rittel73] organized the list into ten distinguishing properties.
money or patience, but that has nothing to do with the logic of the problem, and you can always try to do better.” In short, optimal is not knowable.

4. Solutions to wicked problems are not true-or-false, but good-or-bad. “There is no criterion system or rule which would tell you what is correct or false. The categories of true or false do not apply.” Thus, getting all stakeholders to agree that a resolution is “good enough” can be very challenging.

5. There is no immediate and no ultimate test of a solution to a wicked problem. “For tame problems one can determine on the spot how good a solution-attempt has been. … With wicked problems, on the other hand, any solution, after being implemented, will generate waves of consequences over an extended—virtually an unbounded period of time.”

6. Every implemented solution to a wicked problem has consequences. “Every solution to a wicked problem is a ‘one-shot operation’; because there is no opportunity to learn by trial-and-error, every attempt counts significantly. … [E]very implemented solution is consequential. It leaves ‘traces that cannot be undone’.”

7. Wicked problems do not have a well-described set of potential solutions. “There are no criteria which enable one to prove that all solutions to a wicked problem have been identified and considered.”

8. Every wicked problem is essentially unique. There are no “classes” of, or “canned” solutions that can be applied to a wicked problem. “Part of the art
of dealing with wicked problems is the art of not knowing too early what
type of solution to apply.”

9. Every wicked problem can be considered a symptom of another [wicked]
problem. One can never be sure the problem is being attacked on the right
level.

10. A wicked problem can be explained in numerous ways. There are many
viewpoints conveying the various and changing ideas about what might be a
problem, what might be causing it, and how to resolve it. “There is no rule
or procedure to determine the ‘correct’ explanation or combination of
them.”

11. The problem-solver has no right to be wrong. A scientist or mathematician
is expected to formulate an hypothesis, which may or may not be
supportable by evidence. A designer [or architect] does not have such a
luxury; they are expected to get things right the first time.

Typically, a wicked problem is any problem with numerous inter-dependent attributes.
Examples of wicked problems are locating a new highway, homeless shelter or drug
treatment center; optimizing the features on a new automobile; and defining a business
process. Each of these example wicked problems exhibit one or more of the eleven
properties identified by Rittel and Webber, and most revealing is that all of these
problems suffer from the first property.
Often, enterprise architecture efforts fall victim to wicked problems when the enterprise architecture effort strives to solve one, or more likely, multiple interdependent wicked problems. Enterprise architecture efforts need strategies for dealing with wicked problems in ways that do not jeopardize the enterprise architecture effort.

### 3.2.2 Complexity

Even if an enterprise architecture effort can avoid all the wicked problems, it is very easy for the effort’s scope to grow beyond the ability of the enterprise. Zachman’s framework clearly depicts an open-ended problem space that includes nearly everything that might conceivably be considered for inclusion in the enterprise architecture without the benefit of identifying where to begin and the relative importance of one cell in his framework over any other. Clearly, no effort can address everything at once. In addition, the combinatorics of complexity dictates that no effort can even aspire to address everything at some point in the future. There is a lack of clear insight on where to begin an enterprise architecture effort so that value can be realized early and increased over time. In effect, enterprise architecture as presented by Zachman and the other popular frameworks is an unbounded problem space that provides no clues on to where to begin and what is important and should be performed first and what is not as important and should be deferred to later or never.

The field of Computational Complexity Theory developed as a result of Alan Turing’s description of his theoretical computational model (later dubbed the “Turing Machine”),
which he developed in his treatment of the *Entscheidungsproblem*\textsuperscript{21} in [Turing36]. In 1945 John von Neumann published the architectural description for computer hardware which would go on to become the structural paradigm for virtually every one of the more than one billion computers ever manufactured [vonNeumann45]. This “von Neumann architecture,” as it would later become known, is based on the Turing Machine.

The motivation for the field of Computational Complexity Theory was to determine the resources needed to solve problems. The most common resources are time, expressed as computational steps, and storage space, as in computer memory. In the early days of computers, computer memory was outrageously expensive, so calculating the memory requirements needed to solve a certain problem, or a class of problems, was very important. Today, computer memory is trivially inexpensive except for the most memory-intensive problems. Therefore, Computational Complexity Theory typically addresses the number of processing steps, expressed as a function, a given class of problem will require to reach a solution.

As the field of Computational Complexity Theory has matured, new insights into the nature of problem solving have emerged and certain types of problems have become apparent. One of the most daunting problems is known as NP-complete. “NP” stands for Non-deterministic Polynomial and implies that a non-deterministic Turing Machine could solve an NP-complete problem in polynomial time where the time to solve the problem is no greater than a polynomial function of the problem size.

\textsuperscript{21} Literally “decision problem” in German, but known as the “decidability problem” in English.
Unfortunately, a non-deterministic Turing Machine only exists in theory. No real computer has ever been built that qualifies as a non-deterministic Turing Machine. NP-complete is a class of problems that can only be solved in polynomial time on a nondeterministic machine, but for which no deterministic polynomial time algorithm is known. Thus, in practice, NP-complete problems must be solved—if they can be solved at all—on deterministic Turing Machines, which require exponential time to solve an NP-complete problem. Exponential time is the computation time of a problem where the time required to solve the problem is an exponential function of the problem size such that as the size of the problem increases linearly, the time to solve the problem increases exponentially.

Exponential time problems become intractable for even small problems. For example, as Chapman, Rozenblit and Bahill illustrated in [Chapman01]:

Now let us look at a hard problem that takes an exponential number of steps to solve, because the size of the problem is in the exponent, such as $e^n$. For an exponential problem with $n = 10$ there would be 22,026 operations. If $n = 100$, there would be $2.7 \times 10^{44}$ operations. ... The number of operations quickly exceeds the capability of any machine to compute a solution. For example, if there were a machine that could do $10^{12}$ operations per second (none yet exist) and there was a problem that required $10^{38}$ operations, it would take $10^{20.12} = 10^8$ s[seconds] or more than 3 years to solve. If the problem required $10^{23}$ operations, then it would take 3171 years to solve. Assume the age of the universe is 9 billion years which is $3 \times 10^{17}$ s[seconds], then any problem that requires more than $10^{12.17}$ or $10^{29}$ operations could not be done in the entire age of the universe!

Or, expressed in base 10 for a relatively small problem of size $n = 100$ and assuming a computer with a processing capacity of 10 gigaflops ($10^{10}$ floating point operations per
second), the problem would require about 4 trillion years to solve, or much longer than
the estimated age of the universe.

This is significant because defining a process is the System Design Problem, and
Chapman, Rozenblit, and Bahill proved that the System Design Problem is NP-complete.
They also draw the following implication:

*The first implication of the System Design Problem being NP-complete is that humans cannot design optimal systems for complex problems. And computers will not be able to bail us out, because computers cannot design optimal solutions for complex problems either. ... Furthermore, it is unlikely that a computer can design a complex system better than a human can. Subsets of the entire design process, such as routing and checking interfaces are done better by computers now; however, no computer algorithm exists to create even a simple automobile factory or personal computer.*

In the end, optimal is not knowable—one can never determine the optimum solution and
thus must settle for less. Chapman, Rozenblit, and Bahill noted in [Chapman01] that “the
creation of a system is as much art as it is science, because the combinatorics involved
requires original solutions, rather than fixed algorithms for solving the problem.”

It is not possible to “boil the ocean,” nor is it possible to solve all problems with
architecture. Enterprise architecture efforts need strategies for dealing with complexity in
ways that do not jeopardize the enterprise architecture effort.

### 3.2.3 Enterprise Learning Curve

As if wicked problems and complexity were not daunting enough, enterprises are faced
with an enterprise learning curve that evolves more slowly than the technology the
enterprise is attempting to manage. Jeanne Ross, Peter Weill and David Robertson revealed that enterprises evolve through four architecture maturity levels: Business Silos, Standardized Technology, Optimized Core, and Business Modularity. In their empirical research, they discovered: (1) it took an average of five years for an enterprise to evolve from one architecture maturity level to the next architecture maturity level; (2) enterprises could not skip architecture maturity levels; and (3) transitioning from one architecture maturity level to the next represented a paradigm shift in information technology governance [Ross06, Ross07]. Enterprises that succeed at enterprise architecture (and thus increase their architecture maturity level) do so because they are committed to enterprise architecture and find ways to overcome the impediments that retard the enterprise learning curve. Enterprises that only pursue enterprise architecture because they are told to do so, whether by enterprise edict or federal law, generally do not succeed in increasing their architecture maturity level.

Enterprise architecture not only represents the shared understanding within the enterprise, but also can be the victim of its own failure to establish a shared understanding within the enterprise. Enterprise architecture efforts need strategies that accelerate the enterprise learning curve.

22 “Architecture maturity level” represents the maturation of an enterprise’s ability to develop architecture and govern the enterprise with architecture.
3.3 Identifying Strategies for a Successful Enterprise Architecture

Faced with a plethora of wicked problems, overwhelming complexity, and an enterprise culture that evolves slower than the technology it is trying to manage, it is a wonder that any enterprise architecture effort could ever succeed. The fact that some do succeed provides insight into how others might also succeed. Towards this goal of increasing the probability of an enterprise architecture being successful, this research set out to devise strategies that will help focus enterprise architecture development efforts. Specifically, this research:

- Defines the focus and scope of enterprise architecture by further defining the enterprise dilemma,
- Develops a core set of enterprise business questions,
- Develops an enterprise architecture metamodel,
- Develops a focusing methodology for enterprise architecture development.

These four strategies are discussed in more detail in Sections 4.1.1 through 4.1.4, but prior to discussing these four strategies, it is necessary to describe the enterprise dilemma.
3.4 The Importance of Addressing the Enterprise Dilemma

This research is based on the assertion that the focus and scope of any enterprise architecture effort should be on addressing what this research calls the enterprise dilemma.

Enterprises are hierarchical organizations with the top-most layer of the organization responsible for the strategic direction of the enterprise. Typically the enterprise “bright ideas” that give strategic direction to the enterprise come from the top of the organization while the moneymaking “products” produced by the enterprise come from at or near the bottom levels of the organization. In fact, were it not for the need for change, an enterprise would perform much better for the shareholders if it could divest itself of all the very expensive overhead that occupies the “enterprise level.” However, in reality, no enterprise is perfect and whatever the enterprise is producing now can always be improved. Thus, an enterprise is all about managing change by insuring appropriate and successful evolution.

At the topmost level of abstraction, all enterprises have the same concerns. Every enterprise wants to first survive, then thrive, and then dominate its market. This boils down to two divergent courses of action, sustain what the enterprise is doing now, or change what the enterprise is doing now so that in the future the enterprise will produce a better product and thus do better in the market. In reality, both courses of action coexist at a typical ratio of 85% sustainment and 15% change. Determining how
much of its resources an enterprise will commit to sustainment and how much it will commit to change is what this research calls the enterprise dilemma.

The goal of the enterprise architecture is not to solve the enterprise dilemma because the enterprise dilemma cannot be solved, at least not in any static sense. The enterprise dilemma must be addressed on a continuing basis because the enterprise environment is highly dynamic. It is entirely possible that a good enough decision yesterday could have been much more valuable than a perfect decision next week if an important market opportunity is missed. The enterprise dilemma must be readdressed on a recurring basis if the enterprise wants to survive in the marketplace.

Rarely, if ever, in discussions of enterprise architecture is anything mentioned that resembles what qualifies as the enterprise dilemma. Zachman did address this issue somewhat obliquely in his 1997 article titled “Enterprise Architecture: The Issue of the Century” [Zachman97b] and to some degree he always has with his hierarchical enterprise architecture framework:

*The issue of granularity is of paramount importance to today’s Enterprise because it has everything to do with the frustrations the Enterprise is feeling with regard to the legacy. [sic] and our ability to satisfy current demand. We have been operating at the lowest levels of architectural granularity for the fifty years of history of data processing. Regardless of our intention and even if we have been employing the best architectural techniques by using model driven approaches to systems development, if the implementation was architected at the program, system, application or departmental levels, we inadvertently or sometimes even deliberately introduce substantial discontinuity and redundancies into the Enterprise data, the network and the business rules (columns 1, 3 and 6 models [of the Zachman Framework]).*
What this really translates to is that the processes performed at the enterprise level are very different than the processes performed at subordinate levels within the enterprise, and while the data found within the enterprise may apply in some way to any or all levels of Zachman’s hierarchy, the information and knowledge needs are very different. This is why simply networking and putting web front-ends on all the legacy systems does not address the enterprise dilemma. An architecture that represents the collection of all systems architecture within the enterprise is not an enterprise architecture if it fails to address the enterprise dilemma. A successful enterprise architecture must address the enterprise dilemma.

3.5 Deconstructing the Enterprise Dilemma

In order to survive, all enterprises must possess three fundamental core capabilities. An enterprise must be able to:

- produce product,
- manage resources, and
- maintain accountability.

---

23 The horizontal levels in Zachman’s framework depict this relationship.
The **produce product** core capability employs the assets and consumes the resources of the enterprise. The focus of producing the product is on the present with only limited peripheral vision into the near past and near future. Producing the product is the enterprise’s *raison d'être*. However, no enterprise would exist for long if it focused solely on producing its product. Without the proper management of resources to sustain them, all production lines will grind to a halt. At the least, production lines must be sustained so there is an uninterrupted flow of raw materials in one end, finished products out the other end, and lifecycle sustainment and replacement of all assets employed to produce the product. And, since nothing is perfect, there is always a need to improve the product and the processes employed to produce the product.

The **manage resources** core capability includes the planning, acquisition, allocation and consumption of resources to the increased benefit of the enterprise towards the goals of sustaining the enterprise of today and shaping the enterprise of tomorrow. The focus of resource management is always towards the future. The role of management in any enterprise is heavily weighted toward managing the resources that will sustain and evolve the enterprise. However, it is never enough to produce today’s products; the success of any enterprise is dependent on how well and how quickly that enterprise can develop and produce new and improved products that will keep the customer satisfied and surpass the competition. Resource management is always a balancing act between sustainment and

---

24 There are enterprises that perform services but do not produce a tangible product *per se*. Nonetheless, a service is totally analogous to a product, only lacking in physical dimensions. This research uses a broad definition of the term “product” which is inclusive of service.
change. Deciding what needs to change versus what needs to be sustained is dependent on knowing what is performing below expectations. Thus the enterprise must determine how well the enterprise processes perform and how well the enterprise product satisfies the customers’ needs and desires. The information necessary to evaluate the enterprise performance can only come from maintaining accountability.

The **maintain accountability** core capability is comprised of the proper and prompt accounting of all fiscal, physical and functional assets the enterprise possesses and employs. The focus of accountability is on the past as perfect accountability can only be as current as real-time. It is important to realize that accountability is much more than clean audits and inventory. Accountability applies to:

- **Fiscal accountability:** where the money is, or where did it go, and what did it buy.

- **Physical accountability:** where is an asset, what condition is it in, and what is its expected lifespan so that the enterprise can predict when that asset should be replaced and estimate how much it will cost.

- **Functional accountability:** how well an asset performs, what is its capacity or throughput, and what is its return on investment. How well does the product meet expectations. How satisfied is the consumer with the product. In short, do things perform up to expectations, or should something be replaced, improved or supplemented prior to the end of its expected lifecycle.
Maintaining proper and complete fiscal, physical and functional accountability for the enterprise is the fundamental basis for all performance analyses, which in turn, inform all resource management decisions. Without accurate and timely accountability, resource management is just guesswork. In short, accurately foretelling the future is dubious without an accurately documented past. Therefore, maintaining accountability is a significant prerequisite to answering the business questions required to address the enterprise dilemma.

Taken together, the three enterprise core capabilities constitute the data source (produce product) the data repository (maintain accountability) and the data consumer (manage resources) for the enterprise.

3.6 Rationale: Problem Importance

There are two fundamental motivations for pursuing enterprise architecture; the first is a desire to manage resources better, the second is in response to a requirement from some higher authority to create an enterprise architecture. Experiences associated with this research assisting U.S. Government agencies in their efforts to build enterprise architecture have all begun as the former. However, the enormity of the enterprise architecture task, the absence of any formal academic degree programs specifically focused on enterprise architecture, the ambiguity of what an enterprise architecture is, the absence of any clearly accepted methodology for developing an enterprise architecture are all deterrents. These deterrents are further compounded by wicked problems,
complexity and the enterprise learning curve. As a result, many efforts resort to the latter motivation where the enterprise architecture management aims only to satisfy the mandatory requirement from higher authority to produce a stack of enterprise architecture products. Commonly, the result is reams of shelfware, but nothing of significant value to the enterprise. Any enterprise architecture effort would benefit from a clear focus and scope to provide a better understanding of what falls within the domain of enterprise architecture as opposed to systems architecture or software architecture. In short, an enterprise architecture must have conceptual integrity.

Likewise, any enterprise architecture would benefit from a set of clearly articulated business questions that the architecture will address. Not every question would be applicable to all enterprises, but most of the questions would be applicable to most enterprises. Typically, problem formulation is offered up as a blank whiteboard for the organization to fill in when, in fact, there exists a set of strategic questions that are applicable to most enterprises and should be addressed by the enterprise architecture first before addressing more complicated questions.

Additionally, enterprise architecture would benefit from a framework that was not derived from systems architecture, software architecture, or an exhaustive attempt to include anything and everything conceivable. Systems architecture and software architecture need to be focused on describing a candidate solution that satisfies a set of requirements. Enterprise architecture cannot succeed if it attempts to solve everything. Alternatively, enterprise architecture needs to focus on finding the problems—not on
finding the solutions. In effect, enterprise architecture maps the enterprise problem space while systems architecture specifies candidate solutions within the solution space.

Everything included in the architecture requires resources to build into the architecture and therefore adds to its overall complexity. Thus, the enterprise architecture framework should strive to enforce the conceptual integrity of the enterprise architecture and help to reduce complexity by excluding the minutia that will consume much of the available resources with little or no added value.

Most architecture frameworks avoid describing any detailed strategy for building the architecture. An architecture framework may present a high-level strategy or it may imply a process, or the tool that implements an architecture framework may imply or implement a process. However, most architecture frameworks do not prescribe a specific process for building architecture. Two contributing factors for this are: 1) system engineering and management process definitions are considered external to and beyond the scope of the architecture framework; and 2) the appropriateness of an architecting process may vary depending on the purpose of the architecture. Unfortunately, leaving the architecting process as an exercise to the interested architect results in frameworks with reduced value in aiding the architect in building an architecture. If the scope of enterprise architecture can be sufficiently specified and constrained then there should exist a reasonably specific enterprise architecting process that would help enterprises establish and maintain conceptual integrity while building the enterprise architecture. An
enterprise architecture framework that includes an appropriate enterprise architecting process would be of value to the enterprise architect.

### 3.7 Hypotheses and Research Questions

Ross *et al.* surmise [Ross06]:

*We have come to understand that enterprise architecture boils down to these two concepts: business process integration and business process standardization. In short, enterprise architecture is not an IT issue—it’s a business issue.*

This realization of Ross *et al.* is that integrating the information technology is trivial in contrast to the bigger issues of ascertaining the desired degree of process integration and process standardization within the enterprise. Any information technology decision made before these two issues are decided is premature. Therefore, process integration and process standardization decisions constrain the information technology solution space. Ross *et al.* found this to be absolute [Ross06]. Enterprises that failed to determine process integration and standardization first, invariably failed in their architecture efforts until they went back and addressed these two concerns. While success is not assured if an enterprise does determine process integration and standardization first, failure (at least in the current architectural iteration) is guaranteed if it does not.

In the context of this understanding, this research hypothesizes that there can exist approaches to enterprise architecture that foster a productive context and methodology for building enterprise architecture. Research questions that resonate from this hypothesis are:
• Can enterprise architecture be defined in a way that differentiates enterprise architecture from systems architecture? This is not intended to diminish the role of systems architecture, but to distinguish the role of enterprise architecture, which is to map the problem space, from the role of systems architecture, which is to describe one or more candidate solutions from within the solution space.

• Can the enterprise architecture scope be bounded in such a way that helps to focus the enterprise architecture effort and make more efficient use of enterprise architecture development resources? Since the enterprise architecture cannot be engineered all at once, what should be done first?

• Can an enterprise architecture framework be developed that reflects the enterprise focus and organizes the enterprise architecture development in a way that is resistant to distractions that would divert the focus from the primary enterprise architecture objectives? Knowing what obstacles lie ahead is good; having strategies that will avoid these obstacles is better.

3.8 Conclusion

The explosive growth of information technologies during the last half-century has resulted in a vast and diverse range of possible solutions to information technology problems that recently lacked feasible solutions. Technical problems that not long ago were unsolvable now have a complicated morass of potential solutions. It is a daunting
challenge for any enterprise first to perceive a problem and then to develop a solution to that problem in a way that is a good use of enterprise resources. This is especially daunting in an environment where the decision cycle may be longer than the lifecycles of the information technologies that comprise the solution.

The prognosis is not good; as few enterprises ever truly succeed [Ross06]. However, all is not lost. While it is impossible to know that a given solution is optimal (optimal is not knowable), it is highly probable that a given solution can be good enough [Simon96]. The key to developing a truly successful enterprise architecture lies in being able to develop an enterprise architecture that answers a set of enterprise business questions that identifies what currently does not perform up to expectations. This research provides the definitional groundwork and strategies to increase the likelihood of enterprise architecture success. The research approach and objectives of this work are described in the next chapter.
4. RESEARCH APPROACH

Once you realize that the goal of architecture is to reduce complexity, it becomes clear that the architect must focus as much on what to leave out ... as on what to put in.

~ Steve McConnell
Software Project Survival Guide
1998

4.1 Research Perspective

As a concept, enterprise architecture is very immature. What constitutes an enterprise architecture—beyond the broad definitions discussed earlier—is the subject of ongoing discussion and research. What constitutes the purpose of enterprise architecture is debatable—or worse—left up to each enterprise to define. What is known is the following:

- Architecting is a very expensive pursuit. If enterprise architecture is the combined systems (integrated solutions) architectures of the enterprise, then no enterprise could afford an enterprise architecture without spreading the cost and effort over many years. If the enterprise architecture effort takes many years, then it will be many years before the enterprise architecture can provide any significant value and it is probable that the enterprise architecture will
evolve slower than the enterprise it is trying to depict thus being doomed to being hopelessly out of date.

• By necessity, enterprise architectures are concerned with information technology. Information technology evolves at such a rapid pace that hardware has lifecycles as short as two to four years. Software lifecycles can be much shorter. Therefore, architecture development does not have the luxury of being stretched out over many years lest the effort be overcome by reality changing faster than the architecture.

• Enterprise architecture is a highly contextualized endeavor. Just as Fred Brooks noted there is “no silver bullet” for software development [Brooks75], there is no “ready-to-wear,” or “just-add-data” enterprise architecture framework that will instantly reveal insights once it has been filled with data. For an enterprise architecture to provide value to the enterprise, that enterprise architecture must be tailored specifically to that enterprise and the issues that challenge that enterprise. The evolution of an enterprise architecture framework from its beginnings as a technical architecture framework, then as systems architecture framework, and finally to an enterprise architecture framework has not been conducive to producing enterprise architecture frameworks that adequately address the enterprise architecture problem domain. Principally, there is a growing realization that enterprise architecture
is not solution architecture\textsuperscript{25}. Yet many enterprise architecture frameworks that have evolved from systems architecture frameworks remain centered on a “solutions architecture” perspective. For reasons identified in Chapter 3, enterprises are filled with wicked problems, and many (if not all) are NP-complete. Even if an enterprise could afford a massive enterprise architecture effort compressed into a short period of time, a solution-focused approach that attempts to solve all the (wicked) problems of the enterprise will be stifled by the complexity of the task as predicted by Brooks’ Law\textsuperscript{26} and Cobb’s Paradox. Implementing a process within an enterprise is not an unsolvable challenge, yet often determining what that process should do is an unsolvable challenge for the enterprise. And because of the wickedness and complexity of enterprise-level problems, the enterprise learning curve has the daunting challenge of attempting to match pace with the ever-shortening technology lifecycles.

In short, if an enterprise architecture is worth pursuing at all, then in the interest of attempting an architecture that is achievable, there is at least as much value in knowing what to leave out of the architecture as in knowing what to put into the architecture.

This research differentiates enterprise architecture from systems architecture, defines the bounds of what enterprise architecture should address, and develops a framework and

\textsuperscript{25} A solution architecture is a systems architecture that presents the design for an implementation that satisfies the requirements for that system.

\textsuperscript{26} Brooks’ Law states “adding manpower to a late software project makes it later” [Brooks75].
methodology that aids the architect in focusing the development effort on obtaining value while reducing the risk of expending resources developing architectural artifacts of little or no value. This research addresses these issues in the context of four research objectives.

4.1.1 Research Objective One: Define the Focus and Scope of Enterprise Architecture

The first objective of this research is to define the focus and scope of enterprise architecture. One of the most common failings for any architecture effort is neglecting to achieve the conceptual integrity noted by Fred Brooks as essential to the success of the effort [Brooks75]. The Information Age has made it possible to connect anything to everything. This, of course, makes any system virtually boundless. When systems become boundless, the unfortunate result is that one needs a theory of everything to have a theory of anything [Kawasaki07]. James Martin emphasized the importance of establishing conceptual integrity first, and developed knowledge modeling methods to identify and answer the business questions the architecture was being built to address [Martin06]. This first objective establishes the purpose and bounds of the enterprise so that the enterprise architecture effort can begin to establish the requisite conceptual integrity.
This objective addresses two of the research questions introduced in the prior chapter:

- Can enterprise architecture be defined in a way that differentiates enterprise architecture from systems architecture?
- Can the enterprise architecture scope be bounded in such a way that helps to focus the enterprise architecture effort and make more efficient use of enterprise architecture development resources?

Since it is not possible to do everything at once, it is necessary to limit the scope of enterprise architecture efforts. No enterprise architecture can hope to provide the scope, breadth and depth of the Zachman framework in the near term and Zachman does not provide any insights on where to begin or how to proceed. The strategy proposed here differentiates enterprise architecture from systems architecture where enterprise architecture maps the problem space such that it aids the enterprise in identifying inefficiencies that need improving. The enterprise architecture is not used to generate candidate solutions that describe the design of an implementation. Generating candidate solutions for individual systems is the role of systems architecture.

### 4.1.2 Research Objective Two: Develop a Core Set of Enterprise Business Questions

The second research objective is to develop a core set of fundamental business questions that an enterprise architecture should answer. The enterprise architecture should be more than a static stack of shelfware. To be successful, the enterprise architecture should be
worth its cost, and have a positive return on investment. The enterprise architecture is at least, in part, an information repository. Therefore, it should be able to provide answers to relevant business questions. The set of questions that an enterprise architecture can answer is related to the focus and scope of the enterprise architecture. Like the first objective, this objective addresses the second research question:

- Can the enterprise architecture scope be bounded in such a way that helps to focus the enterprise architecture effort and make more efficient use of enterprise architecture development resources?

The questions an enterprise architecture can answer are dependent on the information requirements of those questions (see Section 5.6.5). This, in turn, drives the data requirements of the enterprise architecture.

### 4.1.3 Research Objective Three: Develop an Enterprise Architecture Metamodel

The third research objective is to develop a metamodel for enterprise architecture that is consistent with the focus and scope of enterprise architecture. Architecture metamodels vary widely in what each offers the architect. As a minimum, a metamodel provides a taxonomy that defines the viewpoints and contexts for architecture. A metamodel might also provide a set of views that depict a subset of the architecture from a given viewpoint in a given context. A third feature that a metamodel may provide is the underlying data model that defines the entities, their attributes and the relationships between entities.
This objective addresses the third research question:

- Can an enterprise architecture framework be developed that reflects the focus and organizes the enterprise architecture development in a way that is resistant to distractions and diversions from the primary objective of the enterprise architecture?

4.1.4 Research Objective Four: Develop a Focusing Methodology for Enterprise Architecture Development

The fourth research objective is to develop a focusing methodology for enterprise architecture development. This effort also addresses the third research question and outlines strategies for building the enterprise architecture based on the results of the previous three research objectives above so that the enterprise architecture can provide usable results quickly while resisting diverting resources to tasks of little to no value.
5. RESULTS

Architectures, like fondue sets and sandwich makers, are rarely used. We occasionally dig them out and wonder why we ever spent the money on them.

~ Chief Architect of a large telecommunications company quoted in [Ross06: p.47]

5.1 Focusing the Focus

Enterprise architecture development, like the enterprise itself, is a momentous undertaking. However, enterprise architecture is neither necessary nor sufficient to ensure the success of an enterprise. Throughout history, far more enterprises have thrived without an enterprise architecture than those that have thrived with an enterprise architecture. Yet, since the dawn of the information age\textsuperscript{27} there has been a growing conventional wisdom that values enterprise architecture-like approaches as a worthwhile endeavor. This conventional wisdom is propagated in part by attention-grabbing headlines from Gartner, Forrester, Standish, Bain & Company, TDWI, and others with claims like “6 of 10 firms with transformation initiatives fail,” and “8 of 10 companies

\textsuperscript{27} The dawn of the information age has been linked to 1956 when the number of white collar jobs in the United States first exceeded the number of blue collar jobs thus marking the shift from a labor based economy to an information based economy.
with cost reduction initiatives fail.” These headlines imbue the message that the probability of success is small and therefore enterprises are in dire need of an approach that will improve the odds of succeeding. The accuracy, validity, implications and significance of these statistics are debatable, but the message these headlines deliver certainly suggests that something like an enterprise architecture is needed to improve the chance of success and thus would be of great benefit.

Unfortunately, the young discipline of enterprise architecture suffers from vague definitions of what enterprise architecture is supposed to be, an absence of any consensus of what an enterprise architecture is supposed to do, and an absence of any general agreement on how an enterprise architecture is supposed to function. This is not surprising in light of the fact that the concept of enterprise architecture has arisen from multiple disciplines in a relatively short period of time. Perspectives from software engineering and software architecting, systems engineering and systems architecting, business management and strategy, and management information systems (MIS) have all contributed to the concept of enterprise architecture as it has evolved over the past four decades.

In the almost three decades since John Zachman described the need for enterprise architecture in [Zachman82], there has been a growing enterprise architecture support industry in terms of frameworks, conferences, books, articles, blogs, curricula and certification programs on the topic as well as a growing and maturing selection of enterprise architecture tools. However, the appeal of enterprise architecture has lost some
of its luster in the most recent decade and now resides near the bottom of the “trough of disillusionment” in the 2010 Gartner Hype Cycle [Gartner10]. This deflation may well be explained by the large number of enterprise architecture efforts that have failed—some spectacularly—to produce the expected benefits. The quest for architectural enlightenment is littered with reams of unused or even unread architectural artifacts, often derisively referred to as “shelfware”, which have been very costly to develop and have returned little to nothing of tangible value\textsuperscript{28} as noted in [BEA05] in Figure 2.

\begin{boxedminipage}{\textwidth}
\textbf{Realities}

We have invested over $400M developing architecture within OSD since 2001.

There has been value created in this effort, but none of that value is recognized due to lack of tangible program output.

There is a strong opinion among oversight bodies, and within the business organizations, that the effort has been one of developing architecture in the absence of any plans for implementation.

Architecture development has too many “constructs”, too many models, too little recognizable links to business processes & outcomes within business missions.
\end{boxedminipage}

\textbf{Figure 2: U.S. Department of Defense Briefing on the Business Enterprise Architecture [BEA05]}

\textsuperscript{28} [Hammer86] noted this problem in 1986.
One would expect that if enterprises were realizing benefits from enterprise architecture, then these same enterprises would be better able to develop the software that plays an integral role in these enterprises’ architectures. However, the recurring *Chaos Report* from The Standish Group reveals that the success rates for software development projects have made only minor fluctuations since 1997 when John Zachman proclaimed enterprise architecture as the “issue of the century” [Zachman97b] (see Figure 3). Now, more than a decade later, enterprise architecture is widely applied, yet no real significant impact has been demonstrated in terms of percentage of successful software developments. Other similar metrics abound, but none have hinted at any significant improvements resulting from enterprise architecture for the majority of enterprise-architecture-practicing enterprises. Successes surely do exist, but something less than success remains the norm for the majority of enterprises. If enterprise architecture was having any significant impact for a significant number of enterprises, then one would expect to see a significant shift in these statistics, but this has not happened to any degree that can be attributed to enterprise architecture.
Figure 3: Summary of Findings for the Chaos Reports (1998-2009) by the Standish Group

The motivations for enterprise architecture can be traced at least as far back as Frederick Winslow Taylor and his pioneering work that began in the 1880s on time and motion studies that Taylor published in 1903 [Taylor03] and what Taylor described as “scientific management” in 1911 [Taylor11]. Taylor posited [Taylor11: p.9]:

63
The principal object of management should be to secure the maximum prosperity for the employer, coupled with the maximum prosperity for each employee.

The words “maximum prosperity” are used, in their broad sense, to mean not only large dividends for the company or owner, but the development of every branch of the business to its highest state of excellence, so that the prosperity may be permanent.

In today’s parlance, this can be paraphrased to: the principal object of management should be to improve the efficiency of the enterprise for the betterment of the enterprise and its stakeholders. In the adolescence of the industrial age, Taylor was most concerned with improving the productivity of the manpower that was at the time still the “engine” of the industrial enterprise. Today, over a century later, but now in the adolescence of the information age, the quest remains the same—to improve the efficiency of the enterprise for the betterment of the enterprise and its stakeholders. However, today’s management is increasingly concerned with improving the productivity of the information technology that has become the “engine” of the information enterprise. The goal remains the same, but the environment that defines the problem domain has radically changed from what it was in Taylor’s day.²⁹

---

²⁹ This is not meant to imply that the challenges that Taylor was trying to address have either been solved or gone away. Taylor’s challenges remain while new challenges have emerged.
5.2 The Focus and Scope of Enterprise Architecture

\[ Pluralitas\ non\ est\ ponenda\ sine\ necessitate.^{30} \]

“Occam’s Razor”
~ attributed to William of Ockham
\[ \textit{circa} \ 1285-1349 \]

It is noted in Chapter 1 that, historically, architecture efforts fail with alarming regularity. It is also noted that the definitions identified in Chapter 2 do not readily distinguish the differences in purpose or function of enterprise architecture and systems architecture. Also as noted above, as a concept, enterprise architecture is very immature. What constitutes an enterprise architecture—beyond the broad definitions discussed earlier—is the subject of ongoing discussion and research. What constitutes the purpose of enterprise architecture is highly debatable—or worse—left up to each enterprise to define for itself. How to develop enterprise architecture is even more of an enigma with most enterprise architecture frameworks focusing on architectural artifacts that the framework asserts are needed to be produced without presuming to prescribe for the practitioner how to produce those artifacts. Christopher Koch noted these shortcomings in [Koch05]:

\[ \textit{Very little having to do with enterprise architecture is available out of the box. It is a competency that must be built from within, unique to each company and that company’s culture. But developing that internal capability is critical to bridging the value gap that persists between IT and the business today.} \]

\[^{30}\] Literally, “\textit{plurality should not be posited without necessity},” applied heuristically as “\textit{one should not increase, beyond what is necessary, the number of entities required to explain anything}.”
The perception that enterprise architecture spans the enterprise from the “penthouse to the outhouse” has been prevalent almost since enterprise architecture’s inception as evidenced by the abstractions and frameworks of Kerner, Zachman, NIST and the Federal Enterprise Architecture Framework (FEAF). This has engendered a conventional wisdom that propagates the expectation that these comprehensive types of enterprise architectures can be successful. However, there exists a body of empirical research on enterprise architecture [Hammer86, Ross03, Weill04, Ross06, Ross07, Weill09] that has focused on examining enterprises that are pursuing enterprise architecture by ascertaining the commonalities among those that succeed at enterprise architecture as well as those that fail. This body of research has yielded results that run counter to this expectation that these comprehensive types of enterprise architectures can succeed.

The enterprise architecture concept is difficult enough, most enterprise architecture frameworks compound the difficulties by offering ever-increasingly-comprehensive frameworks that make little or no significant attempt to set limits on the enterprise-architecting problem domain. This is because most enterprise architecture frameworks are based on the assumption that there is value in producing all of the requisite architecture framework outputs, products, and artifacts that the framework prescribes, although the empirical evidence does not confirm this assumption either. To satisfy a framework’s underlying metamodel normally requires a huge amount of effort to first identify and then to collect all the data necessary to populate a model. But after the enterprise does all this data identification and collection, then maintenance of the enterprise architecture often cannot keep pace with the evolving enterprise. Enterprise
architecture is an ongoing endeavor to develop a shared understanding of the enterprise and to capture the knowledge that is the basis of that shared understanding. But if the amount of information that must be captured becomes too onerous, the magnitude of the task becomes a liability.

In 1969, Herbert Simon [Simon69] predicted the burdens that an information-rich environment based on a large number of artifacts like an enterprise architecture would bestow upon the enterprise:

Now when we speak of an information-rich world, we may expect, analogically, that the wealth of information means a dearth of something else – a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention, and a need to allocate that attention efficiently among the overabundance of information sources that might consume it.

Clearly, an architecture that exceeds all the available bandwidth of the intended consumers of that architecture will fail to communicate its intended message to those consumers of that architecture.

At the time DoD was questioning the efficacy of its BEA effort, the Business Management Modernization Program (BMMP) office estimated that to complete any architectural what-if excursion would require 20-60 days of effort by 20-40 full-time equivalent employees. This meant that in some cases the cost of doing architecture far exceeded the cost of the proposed change. In other words, and ironically, the “ounce of prevention” could cost much more than the “pound of cure.”
Fortunately, the wisdom of William of Ockham has not gone unnoticed. If Zachman, FEAF, TOGAF and DoDAF represent comprehensive approaches to enterprise architecture, then there have begun to emerge advocates on the opposite side of the spectrum of architecture methodologies. Gerald Khoury and Simeon Simoff concluded in [Khoury04: p66, emphasis added]:

> What we need from an enterprise architecture is a method that is effective in:

1. Developing a single and coherent model of an enterprise, and
2. Allowing us to guide the future development of an enterprise without the creation of arbitrary internal boundaries.

As noted in Figure 2 above, BMMP observed in [BEA05] the following even before DoDAF v2.0 doubled the number of its defined DoDAF artifacts (views/models) to fifty-two:

> Architecture development has too many “constructs”, too many models, too little recognizable links to business processes & outcomes within business missions.

Jeanne Ross, Peter Weill and David Robertson concluded from the results of their empirical research in [Ross06, p.50]:

> While the architecture for a new building is captured in blueprints, enterprise architecture is often represented in principles, policies, and technology choices. Thus, the concept can be difficult for managers to get their arms around. We have found that a simple picture, which we refer to as the “core diagram,” helps managers debate and eventually come to understand their company’s enterprise architecture. This simple one-page picture is a high-level view of the processes, data, and the technologies constituting the desired foundation for execution. The core diagram provides a rallying point for managers responsible for building out and exploiting the enterprise architecture.

---

31 It is highly significant that the two major empirical studies of enterprise architecture, PRISM [Hammer86] and [Ross06] both strongly correlated architectural success with enterprises that established information technology principles.
Yet while calls for reducing the number of architectural constructs and models, or advocating for a simple “single and coherent model of the enterprise” like a core diagram are gaining momentum, Jeanne Ross et al. concluded that while the benefits of a core diagram can be of great value, achieving that simplicity is usually extremely difficult [Ross06, p66]:

Enterprise architecture core diagrams are tough to draw because they force management to develop a simple vision of a complex organization. Management is trying to articulate the essence of the company in the core diagram. … Delta Air Lines required sixty iterations to complete the drawing of its core diagram. But once complete, management should anticipate that, despite rapidly changing business conditions, the essence of the company will stay the same. The goal is to digitize the core to make it predictable, reusable, and reliable. By digitizing the core processes, data, and technologies depicted in the core diagram, management expects to provide a foundation for execution, not a set of handcuffs.

The inclination of some management teams is to include too many processes or too much detail in the enterprise architecture core diagram. Agreeing on what not to include can be a challenging but fruitful exercise in management focus. Intense management debate can expose options for the foundation for execution. The enterprise architecture design exercise forces clarification of a workable vision.

This echoes the realization of Steve McConnell in [McConnell98] that:

... once you realize that the goal of architecture is to reduce complexity, it becomes clear that the architect must focus as much on what to leave out ... as on what to put in.

Therefore if the enterprise architecture is not helping to reduce the complexity of the enterprise, then it is probably not helping the enterprise. If it is a laborious, difficult and expensive task to produce an enterprise architecture of little or no value to the

32 This underlies the shortcomings of frameworks like TOGAF and DoDAF, in that each expands over time with more and more views, outputs and artifacts without any meaningful discussion on where to apply Occam’s Razor.
enterprise, it is probably no less a laborious, difficult and expensive task to produce an enterprise architecture that is of value to the enterprise. But the objective should not be to produce more architectural artifacts. The objective should be to apply Occam’s Razor and produce the least number of architectural artifacts that focus on what Ross et al. calls the core diagram [Ross06]. This core diagram should embody the operating model(s) that expresses the desired extent of process integration and process standardization within the enterprise along with the principles that the enterprise is attempting to uphold in its future state. This will then better enable the enterprise to address the enterprise dilemma.

5.2.1 Enterprise Architecture Maps the Enterprise Problem Space

This research asserts that the purpose of enterprise architecture should not be the same as the purpose of systems architecture. Some assert that enterprises can be architected and engineered as if they were systems writ large. Admittedly, from an engineer’s perspective there is an intuitive appeal to this argument. However, the lessons that have been highlighted in earlier discussions show that the complexities and combinatorics of trying to solve numerous simultaneous instances of the System Design Problem scale to the realm of the impossible very quickly. If the focus of systems architecture is the conceptual design that defines the structure and behavior of a system—i.e., solution focused—then it does not make sense for enterprise architecture to have the same focus. Therefore, if systems architecture is focused on designing solutions, then enterprise architecture would best be employed for another purpose. However, since the notion of
designing solutions lacks a context without a specific problem to solve, the role of enterprise architecture should be on finding relevant problems that systems architecture and systems engineering must then attempt to solve.

No enterprise architecture will reveal what line of business an enterprise should pursue, nor will it reveal how an enterprise should conduct its business. What an enterprise does and how it does it is bounded by a set of management decisions. Given that management has made these requisite decisions, enterprise architecture should aid in the management of enterprise resources with the aim of achieving the goals established for the enterprise. At the top-most level of abstraction, this equates to determining how much of the enterprise’s resources should be committed to sustainment—doing tomorrow what the enterprise did today—and how much of the enterprise’s resources should be committed to change. This thesis defines this as the enterprise dilemma.

For this reason, this dissertation asserts that an enterprise architecture best addresses the enterprise dilemma if it is employed to map the enterprise problem space for the purpose of identifying gaps and overlaps in the allocation, employment and consumption of the enterprise’s assets and resources. Once these gaps and overlaps are identified, then systems architecture can be applied to design candidate solutions that alleviate whatever undesirable conditions exists as a result of these gaps and overlaps.

33 In reality, this is not a given. Empirical evidence shows that these key decisions are either never made, poorly made, poorly communicated, or made too often for the enterprise to align itself before new decisions are pronounced.
Enterprise architecture can best address the enterprise dilemma if it is designed to answer questions relevant to the enterprise dilemma. James Martin developed The Six-step Problem-Focused Question-Based (PFQB) Architecture Process in [Martin06] to focus the architecture development “… on the most important issues to be addressed with the architecture.” Martin presents a general method that can be applied to all architecture development [Martin06]. However, this research applies specifically to the challenge of enterprise architecture and its focus on the enterprise dilemma.

Martin noted in [Martin06: p.3]:

Enterprise architectures are developed for the purpose of discovering more efficient operational approaches for a business venture. Enterprise architecture models are developed to give business managers a better understanding of the things the enterprise owns, operates, and produces so they can make better business decisions.

The focus of this understanding should be ascertaining what within the enterprise is performing adequately and thus needs to be sustained (kept the way it is) and what within the enterprise is not performing adequately and thus needs to be changed (improved), all within the bounds of resource constraints.
The enterprise dilemma, as depicted in Figure 4, reduces to the oft-recurring problem of ascertaining the allocation of resources between sustainment and change.
5.3 A Core Set of Enterprise Business Questions

*I try to take large, general questions that are difficult to resolve and break them down into small, very specific questions that have clear answers.*

~ Bill James
quoted in *Wired*, May 2011

Beyond the contextual awareness interrogatives or interrogatory categories of what, how, where, who, when, and why, in the Zachman framework, there do not appear to be enterprise architecture frameworks that focus *a priori* on what questions need to be answered to derive any insights (and presumably value) from an enterprise architecture towards the goal of addressing *the enterprise dilemma*. The shortcomings of the Zachman, FEAF, TOGAF and DoDAF frameworks, as well as other architecture frameworks, is that each prescribes a set of architecture products, views, and artifacts that must be produced, but there is an absence of any context that explains what specific questions these products, views, and artifacts will answer. This may be due in large part to information technology architecture’s borrowed legacy from traditional architecture where there is an understandable over-emphasis on structural perspectives that can be readily captured in illustrations rather than on the functional perspectives that are more difficult to depict. In designing a structure, the load-bearing components must be designed for the lifespan of the structure else the structure will collapse. Also, the role of a load-bearing component is usually obvious within the architectural illustrations. In an
enterprise, many “load-bearing” components may have very short lifecycles and thus need to be replaced or updated on a recurring basis as both the load increases or decreases and as the functions evolve from what was originally intended. Additionally, in an enterprise architecture the functions performed by “load-bearing” components may not be obvious within an architectural illustration.

The emphasis in the enterprise architecture must be to highlight the functional perspective of the enterprise which mandates that as the functionality changes, so must the (infra)structure of the enterprise. This does not mean that structural changes are less important than functional changes, but that structural changes (e.g., end-of-life for an asset) only have real significance in the context of how the structural changes influence and effect functionality. If an asset reaches its end-of-life and this event does not cause any perceived effect on any functionality, then it certainly raises doubts about the need to replace the asset.

For enterprise architecture there is no existing pre-defined “master set” of business questions that specify what issues the enterprise architecture must address. However, there is a logical place from which to start developing business questions. Martin describes the business question development for three architectural case studies using his Six-step Problem-Focused Question-Based (PFQB) Architecture Process [Martin06: p.154, p.162, p.174]. While each of those sets of business questions appears very different, owing to their origins from three different organizations with different cultures and different lexicons, there are similarities. These similarities reveal that there should
exist a beneficial set of core business questions that can serve as a point of departure for a new enterprise architecture effort.

![Diagram showing the Six-step Problem-Focused Question-Based (PFQB) Process](image)

Figure 5: Six-step Problem-Focused Question-Based (PFQB) Process [Martin06]

In [Martin06], business question definition is part of the Problem Framing (Step 1) phase of the PFQB process (see Figure 5), which is decomposed into the following activities:

- Business Analysis
  - Purpose Identification
  - Business Question Definition
  - Business Objectives Prioritization
- Conceptual Definition of the Problem
For *Purpose Identification*, [Martin06: p.116] provides examples of various purposes for pursuing architecture. The purpose of enterprise architecture should be consistent with the purpose expressed by [Martin06: p.3] which is to provide decision makers with “… a better understanding of the things the enterprise owns, operates, and produces so they can make better business decisions.” Establishing the purpose of the architecture is critical to ensuring that an architecture performs a useful function. Documentation that has no utility beyond a historical snapshot of the enterprise architecture at the time of its publication becomes the shelfware that many architecture efforts have produced. If the architecture has a defined purpose and becomes the repository for knowledge capture as well as the source of current information relevant to that knowledge, then the architecture has a chance of becoming more than static shelfware. But first it must have a purpose.

Business questions define the purpose, scope and functionality of the enterprise architecture and are thus the *de facto* requirements for the enterprise architecture. Therefore, the *Business Question Definition* activity is a critical component of architecture development as defined by Martin [Martin06]. Business questions are later used to identify the entities (nouns), relationships (verbs) and processes of the enterprise
that need to be accommodated by the enterprise architecture metamodel. This step is what Martin identified as Business Objects Identification [Martin06]. Identification of these business objects facilitates the Business Facts Identification, Conceptual Schema Definition, Architectural Framework Development, and Conceptual Definition Validation that will enable the enterprise architecture to function as desired (i.e., answer the business questions).

In practice, the enterprise dilemma is too big to address all at once. Therefore, the enterprise architecture effort must settle on a sub-set of the enterprise dilemma to address in each iteration of development of the architecting effort. Determining what subset of the enterprise dilemma to decompose and model is challenging for all the same reasons that confound enterprise architecture as a whole. This is the focus of Business Objectives Prioritization activity in [Martin06] and is meant to specify an achievable set of business questions that can be modeled in the architecture in a reasonable period of time.

As noted earlier, Ross et al. [Ross06: p.47] identified where an enterprise should focus its efforts in the beginning stages of enterprise architecture development by identifying where an enterprise architecture should focus at its maturity:

\[ \text{The key to effective enterprise architecture is to identify the processes, data, technologies, and customer interfaces that take the operating model from vision to reality.} \]

In Section 3.5 it was noted that an enterprise must have three core capabilities: produce product, manage resources, and maintain accountability. For an enterprise to survive in the long run, it must be sufficiently competent in each of these three core capabilities.
For enterprise architecture efforts that are just beginning, or for efforts that are starting afresh, the focus should be on the **produce product** core capability. This focus will ensure that the effort concentrates on the enterprise’s *raison d’être* and not on tangential aspects of the enterprise. Future iterations of the enterprise architecture development effort, should expand the scope to include the **maintain accountability** and **manage resources** core capabilities, as the overarching goal of the enterprise architecture is to ultimately aid in the management of the enterprise’s resources by aiding the enterprise in addressing what this research denotes as *the enterprise dilemma*.

As with anything new, the first iteration in an enterprise architecture development effort will generally be very difficult for the architecture team. It is very important to scope the effort realistically and pragmatically. Therefore, while at the beginning, it is very important to think of the end while recognizing the need for multiple iterations. Later iterations will become easier as the architecture team develops skills, experience and the shared understanding needed to be successful.

Business questions span the spectrum from the trivial to the highly complex. Obviously, the goal of any enterprise architecture is to answer non-trivial questions that will assist the enterprise in addressing *the enterprise dilemma*. However, many non-trivial questions can be resolved by answering a set of concatenated trivial questions. Therefore, it is important to identify the trivial questions that will later be combined with other trivial questions to develop answers to more complex questions. This is achieved with both top-down and bottom-up approaches.
From the top, the effort begins by identifying non-trivial questions and then decomposing these questions to determine the trivial questions that are combined to answer more complex questions. While ascertaining relevant complex business questions and decomposing these complex questions into simpler questions is a necessary knowledge management strategy, experience has shown that most architecture teams do not have the requisite expertise to formulate an acceptable set of business questions without extensive involvement of subject matter experts and senior managers.

While waiting for this top-down effort to get organized and yield results, the bottom-up effort can begin by identifying and answering some simplex questions that have a very high likelihood of contributing toward answering many of the complex questions. Toward this objective, the architecture team begins determining the critical entities within the enterprise. The first step is to identify the trivial questions, which henceforth will be called simplex 1st order34 business questions. Simplex 1st order business questions only require one entity, and therefore no relationship between entities, to derive an answer. The following are recommended simplex 1st order business questions:

- What are the products produced by the enterprise?
- Who are the customers of the enterprise?
- What are the facilities used by the enterprise?
- What are the production lines used by the enterprise?

---

34 These 1st order business questions are comprised of one entity or one noun representing something within the enterprise.
• What are the organizations within the enterprise?
• What are the enterprise architecture principles?

These simplex 1\textsuperscript{st} order business questions are trivial, but necessary to answer more complex business questions. The rationale for identifying the simplex 1\textsuperscript{st} order business questions is to identify the entities that will need to be modeled in the enterprise architecture. These entities represent the assets that the enterprise employs to produce its products or represent the actual products themselves. Once a representative set of entities has been identified from these simplex business questions, then more complex, or 2\textsuperscript{nd} order,\textsuperscript{35} business questions can be identified. Recommended 2\textsuperscript{nd} order business questions are:

• What are the products of a production line?
• What are all the production lines that produce a product?
• Who are the customers of the products of the enterprise?
• What production lines are at what facilities?
• What are the production line components of a given production line?

\textsuperscript{35} These 2\textsuperscript{nd} order business questions are comprised of two entities and a direct relationship, or two nouns and a verb.
• When is the earliest end-of-life (EOL) for a production line component on a given production line?\textsuperscript{36}

• What is the minimum mean-time-between-failure of any production line component on a given production line?

• What is the minimum actual-time-between-failure of any production line component on a given production line?

• What production line components are acquired by a given acquisition program?

• What technology a given research and development program is developing?

These 2\textsuperscript{nd} order business questions begin to establish what will become a rich set of relationships that need to exist between entities, and will enable the enterprise architecture to answer increasingly complex questions as the architecture matures. Note that each of these 2\textsuperscript{nd} order business questions contains two simplex 1\textsuperscript{st} order business questions with a direct relationship between the entities.

3\textsuperscript{rd} order business questions need to be decomposed into the two 2\textsuperscript{nd} order business questions necessary to answer the more complex 3\textsuperscript{rd} order business question. Examples of 3\textsuperscript{rd} order business question are:

\textsuperscript{36} In this question, “end-of-life” is an attribute associated with “production line component” entities. Therefore, formulation of this question depends on the attribute value for one of the entities in a 2\textsuperscript{nd} order business question. The two business questions that follow are similar.
• What are the production line components that contribute to producing a product?
  
  o What are the production line components of a given production line?

  o What are the products of a production line?

• At what facility is a product produced?
  
  o What production lines are at what facilities?

  o What are the products of a production line?

• At what facility is a production line component located?
  
  o What production lines are at what facilities?

  o What are the production line components of a given production line?

• Which acquisition program requires a technology being developed by a research and development program?
  
  o What technology is being developed by a research and development program?

  o What technologies are required by given acquisition program?

At this point, one could be compelled to attempt an effort to complete a few more iterations towards the goal of developing the comprehensive set of business questions that would completely address the enterprise dilemma. However, on closer examination, one can see that this is a problem that expands exponentially and would become intractable
very quickly. Since it is not feasible to identify every question, it is important to identify relevant questions that bear significance on how the enterprise desires to manage its resources towards the goal of addressing the enterprise dilemma. For reasons that are discussed in Section 5.6.1, it is critical that empowered stakeholders participate in the development of the enterprise business questions. Lack of empowered stakeholder involvement in this critical step will be detrimental to the enterprise architecture development effort as participation is necessary in order to achieve a shared understanding of the problem domain by the empowered stakeholders and to establish the conceptual integrity needed to ensure the efforts success. This realization is the basis for selecting the three domains of principle activity: empowered stakeholder, surrogate stakeholder, and no stakeholder, as a focus of comparison for enterprise architecture case studies in Chapter 6.

Higher order (n\textsuperscript{th} order\textsuperscript{37}) business questions need to be decomposed into the set of 2\textsuperscript{nd} order business questions necessary to answer the more complex n\textsuperscript{th} order business question. Examples of complex business question are:

- What is the effect on a capability given some specified change in the architecture?
- What are the functional dependencies among acquisitions?

---

\textsuperscript{37} This involves two entity types and an implied relationship, but no direct relationship. The order of a business question is determined by the number of actual entities needed to derive the answer. A 3\textsuperscript{rd} order business question requires three entities with at least two relationship pairings in order to derive an answer.
• Which programs are not synchronized to provide users capability in the same timeframe?

• What is the impact of closing a given facility?

• What is the impact on a production line if an acquisition program is delayed or canceled?

These business questions are ambiguous because words in these questions like “effect,” “capability,” “change,” “functional dependencies,” “synchronized,” “timeframe,” “impact,” and “delayed” are not singular entities or conditions, but represent multiple states or outcomes or even unspecified conditions. These types of business questions require disambiguation resulting in explicit definitions for these terms within the enterprise architecture before the enterprise architecture metamodel can resolve answers for these questions.

Martin discusses the difficulty of obtaining good business questions and presents a strategy for eliciting usable business questions [Martin06: p.119]. An enterprise architecture effort is an attempt to make explicit many concepts that have only been perceived in the abstract within the enterprise. Typically, many business questions that are suggested at the beginning of an enterprise architecture effort are too ambiguous and abstract to be answerable. For example:

• What strategic path should senior decision makers take?

• What guidance needs to be provided to the acquisition community?
• How well is the community following the strategic path?

• How do we measure benefit among competing concepts?

• What are the costs versus benefits of select concepts?

• How can we optimize investments?

• What are the assessed impacts and associated potential of emerging/advanced technology?

• What are the strategic vectors and architectures needed to guide our investments in technology and advanced concepts?

• What are the potentials for horizontal integration (vulnerabilities, efficiencies, synergies, opportunities)?

One could surmise that if the first two questions could be answered by the architecture, then all the senior leaders of the enterprise could be terminated. Other questions are seeking qualitative assessments without having prescribed ways to measure “wellness,” “benefit” or “optimality.” All of these questions were suggested by stakeholders as being within the scope of enterprise architecture, but none of the individual stakeholders suggesting these questions could specify any unambiguous definition for what was being asked nor could they provide any suggestions on how their questions could be modeled and answered. Therefore, it is critical to coach the participants towards the goal of selecting business questions that can be explicitly decomposed into definable entities and relationships that can be modeled in the architecture. It does no good to define questions
about “associated potential,” “advanced concepts,” and “potentials for horizontal integration” if these concepts cannot be explicitly modeled. Therefore, as noted by Maier, it is necessary to “relentlessly grind the ambiguity out of buzzwords” to ensure that business questions are clear, unambiguous, and can be answered by the architecture model [Maier09, p.72].

The requisite skills that contribute to an enterprise’s “architectural maturity level” as described in [Ross06] include: how an enterprise employs an enterprise architecture to define and depict key entities; establish levels of expectations; determine the operating model(s) that incorporates the desired degrees of process integration and process standardization; and then identify the gaps and overlaps that need to be eliminated in order for the enterprise to achieve its desired future state. This architectural maturity cannot be bestowed upon an enterprise a priori, but must be developed and nurtured from within the enterprise, achieved within the bounds of a shared understanding developed by the empowered stakeholders, and in concordance with the conceptual integrity envisioned for the enterprise.
5.4 Enterprise Architecture Frameworks

Underlying the ... statements about benefits and risks is a basic principle: architectures—at whatever levels—are important only to the extent they link to and enable success in the basic mission and performance of the organization.

Information Management Directions: The Integration Challenge, p144, [Fong89] 1989

There is no authoritative definition for either “architecture framework” or “enterprise architecture framework.” As a result, there is disparity about what is or is not included in a framework ranging from a one-page matrix like Zachman’s framework to hundreds of pages for some others. A framework may include a taxonomy, which is usually based on viewpoints and contexts, a data model or a metamodel of a data model, a set of predefined views, and possibly a methodology for creating architectures that conforms to the framework. Understanding how an architecture framework was developed and how it has evolved provides insights into what problems that framework was originally developed to address and how its problem space has evolved over time. The emergence and evolution of the major architecture frameworks are discussed in Section 5.4.1 through Section 5.4.13. This discussion includes frameworks from open source, consortia and commercial developers.
5.4.1 Business Systems Planning

Endeavors that resemble enterprise architecture began at least as early as the 1960s. The integration of very expensive information technologies into the enterprise was motivation for seeking ways of identifying not only what technologies to integrate, but the role these technologies should play and how that integration should be accomplished. Initially, this “architecture” role was typically provided by the mainframe vendor supporting the enterprise. The undisputed leader among vendors of mainframes during that period was IBM. Starting in the late 1960s and throughout the 1970s, IBM pursued its Business Systems Planning (BSP) effort.\textsuperscript{38} Initially BSP was an internal effort to help IBM model IBM’s own enterprise, but eventually BSP became one of the services IBM offered to customers. BSP was an enterprise-architecture-like process to align information technology within the enterprise. Over the course of IBM’s efforts, a conceptual enterprise planning and control model emerged as shown in Figure 6 from [Kerner79]. This depiction was not referred to as “enterprise architecture” in the beginning, but the metaphor used by the BSP process was very similar to the perspectives adopted by enterprise architecture frameworks that emerged afterwards. Initially, the “architecture” that emerged from BSP focused on the information technology architecture of the system (or systems) supplied by the vendor and typically fell short of the entire enterprise.

\textsuperscript{38} John Zachman led IBM’s Business System Planning program in IBM’s western region between 1973 and 1984.
By 1982, John Zachman had begun speaking of the need for “enterprise architectures” and “enterprise-level architectures” [Zachman82]. In time, as the proliferation of
information technologies grew towards ubiquity, the scope of enterprise architecture would likewise grow in significance within the enterprise.

As the information technology marketplace evolved away from implementations supported by single-source mainframe vendors towards plug-and-play hardware and hardware-independent software, enterprises no longer had a single vendor supporting the enterprise’s information technology infrastructure and therefore, by necessity, had to assume much more responsibility for their own enterprise architecture. As a result, what began as a vendor supplied and supported architecture methodology transformed into an in-house architecture methodology usually represented by “technical architectures” that specified what hardware, software, and standards must be used or adhered to by enterprise information technology to achieve standardization and interoperability within the enterprise.

Then, as many enterprises were implementing technical architectures, networking rapidly transformed into internetworking.\(^{39}\) Suddenly, within the foreseeable future, it would be possible to connect anything to everything. The scope of architecture became virtually boundless and the concept of an enterprise architecture became a pressing need for any enterprise that was trying to develop and integrate information technologies. Soon, enterprise architecture frameworks would begin to emerge.

\(^{39}\) 3Com built the first commercial Ethernet adapter in 1981. Initially very expensive, by 1994 Ethernet cards would come pre-installed on virtually all personal computers on the market.
5.4.2 PRISM

The 1986 PRISM report is the first published study on enterprise architecture identified by this research and it marked the publication of the first *bona fide* enterprise architecture framework and methodology. The PRISM report was based on empirical research examining what enterprises were doing at the time, and identifying what successful architecture efforts had in common [Hammer86]. The PRISM framework is represented as a four-by-four matrix as shown in Figure 7. The PRISM framework provided a high level taxonomy based on viewpoints represented by the *Inventory*, *Principles, Models*, and *Standards* columns and contexts represented by the *Infrastructure, Data, Application* and *Organization* rows. PRISM also described a methodology and architecture development process. Interestingly, the authors of PRISM identified from their research that the architecture had to be developed and maintained at a faster pace than the evolution of the enterprise. However, this philosophy of architecting was soon overshadowed by a holistic philosophy espoused by other frameworks.

---

40 Equates to what is referred to as the “As-Is,” today.
41 Equates to what is referred to as the “To-Be,” today.
The year following PRISM, the first version of Zachman’s framework was published [Zachman87]. Note in Figure 8 that Zachman’s hierarchical top-to-bottom depiction readily maps to the planning and control model from [Kerner79] in Figure 6 save that Zachman used a rectangular matrix and Kerner used a triangular matrix. One can also observe that the 1987 Zachman framework was only based on a three-by-five matrix (not counting the “actual system” or implementation level) while the PRISM framework was based on a four-by-four matrix (see Figure 8 and Figure 7). However, Zachman’s approach was much broader in perspective and scope than PRISM even before Sowa and Zachman expanded the Zachman framework to a six-by-five (again not counting the

![PRISM Framework](image)

**Figure 7: PRISM Framework [Hammer86]**

### 5.4.3 Zachman’s Framework

The year following PRISM, the first version of Zachman’s framework was published [Zachman87]. Note in Figure 8 that Zachman’s hierarchical top-to-bottom depiction readily maps to the planning and control model from [Kerner79] in Figure 6 save that Zachman used a rectangular matrix and Kerner used a triangular matrix. One can also observe that the 1987 Zachman framework was only based on a three-by-five matrix (not counting the “actual system” or implementation level) while the PRISM framework was based on a four-by-four matrix (see Figure 8 and Figure 7). However, Zachman’s approach was much broader in perspective and scope than PRISM even before Sowa and Zachman expanded the Zachman framework to a six-by-five (again not counting the
“actual system” or implementation level) matrix in 1992 in [Sowa92] as shown in Figure 9.

![The Original Zachman Framework for Information Systems Architecture](image)

**Figure 8:** The Original Zachman Framework for Information Systems Architecture [Zachman87]
Zachman’s original framework provided a high level taxonomy based on viewpoints represented by the *Ballpark View*, *Owner’s View*, *Designer’s View*, *Builder’s View*, *Out-of-context View*, and *Actual System* rows of the matrix and contexts represented by the *Data Description*, *Process Description* and *Network Description* columns. Zachman now asserts that his framework is a schema, ontology, and metamodel [Zachman08], but this current description is a bit of an overstatement for Zachman’s original framework as it was presented in 1987. Following the initial publication of his framework in 1987, Zachman discussed a methodology for using his framework in [Fong89: Chapter 5].

Figure 9: The Expanded Zachman Framework for Information Systems Architecture [Sowa92]
5.4.4 NIST Framework

In 1989, when NIST published its workshop proceedings titled *Information Management Directions: The Integration Challenge* [Fong89], the authors of Chapter 7 of this document chose the triangular depiction of a hierarchical decomposition shown in Figure 10 which is very similar to the depiction used in [Kerner79] shown in Figure 6.

![Diagram of Levels of Architecture](image)

Figure 10: Levels of Architecture [Fong89: p.138]
The NIST framework identified different contexts; *Business Unit Architecture*, *Information Architecture*, *Information System Architecture*, *Data Architecture*, and *Deliver System Architecture*, but did not identify any viewpoints. The NIST framework did not provide a metamodel, data model, views, nor a methodology for developing an enterprise architecture. However, the authors did identify the notion of hierarchical dependency within the framework [Fong89: p.137]:

*The idea of an enterprise architecture reflects an awareness that the levels are logically connected and that a depiction at one level assumes or dictates that architectures at the higher levels have been completed.*

A difficulty with this concept is that the authors of the NIST enterprise architecture framework did not consider an enterprise architecture completed until everything in the hierarchy was completed. This carries great risk if the enterprise architecture is never completed, or is not completed before any layer in the architectural hierarchy changes.  

The serious consequences of this holistic approach to enterprise architecture cannot be overcome in a rapidly evolving enterprise. First, this leads to the perception that an enterprise architecture effort is of limited value until the architecture is completed. Second, due to the enormity of the task, there is a high probability that the enterprise architecture will never be completed, or when “completed,” the enterprise will have evolved significantly away from what is depicted in the enterprise architecture.

---

42 Interestingly, John Zachman was chairman of the panel that authored Chapter 5 of the NIST workshop proceedings. This chapter, titled “The Integration of Systems Planning, Development, and Maintenance Tools and Methods,” contains a methodology that describes how one could build an architecture based on Zachman’s 1987 framework. However, there is no apparent link between [Fong89] Chapter 5, which Zachman chaired, and the enterprise architecture framework discussed in [Fong89] Chapter 7.
5.4.5 **Enterprise Architecture Planning**

In 1992 Steven Spewak introduced Enterprise Architecture Planning (EAP), which he aligned to the top two rows of Zachman’s 1987 framework. Spewak’s EAP provided a taxonomy composed of viewpoints from Zachman’s *Ballpark View* and *Owner’s View* and specified contexts as *Data Architecture*, *Application Architecture*, and *Technology Architecture* which aligned with Zachman’s *Data Description*, *Process Description* and *Network Description* columns. Spewak limited EAP to the top two layers (rows) of Zachman’s Framework because [Spewak92: p.13]:

> The design of systems begins in the third layer, which is beyond the purview of EAP. Restated, EAP focuses on defining data, applications, and technology architectures for the overall enterprise as opposed to designing these for specific purposes.

Spewak was aware of the shortcomings of a holistic approach to enterprise architecture and recognized that while the levels needed to be connected in some way, they also needed to be mostly independent so they could evolve independently.

5.4.6 **TAFIM**

In 1993, the initial version of DoD’s Technical Architecture Framework for Information Management (TAFIM) was published. TAFIM was never intended to be an enterprise architecture framework, but has been used as the conceptual basis for other frameworks (TOGAF, C4ISR, DoDAF, and other DoDAF based frameworks) that have expanded

---

43 In later versions of Zachman’s framework, these become the “what,” “where” and “how” columns.
over time to include enterprise architecture within their scope. Work on TAFIM had been in progress since at least 1986 and by the time version 3.0 was published in 1996 [TAFIM96], TAFIM had grown to eight volumes totaling 1378 pages and prescribed an approach that included the “DoD Standards-Based Architecture (SBA) Planning Process.” A TAFIM technical architecture was comprised of Data Architecture, Application Software Architecture (mission specific), and Technical Infrastructure Architecture contexts as well as Enterprise Level, Mission Level, Function Level, Application Level, and Personal Level viewpoints. TAFIM did not have a specified metamodel or a defined data model, but did provide a technical reference model. TAFIM suggested a number of views and provided a text description of what those views should depict. TAFIM’s contexts bear a remarkable resemblance to Spewak’s Data Architecture, Application Architecture, and Technology Architecture contexts described in EAP and TAFIM’s viewpoints readily map to Zachman’s Planner, Owner, Designer, Builder, Sub-contractor layers (rows) in the 1992 version of the Zachman framework.

5.4.7 TOGAF

With the permission of DoD, The Open Group used TAFIM as the basis for The Open Group Architecture Framework (TOGAF). The first version of TOGAF appeared in 1995 and for contexts it included four architectures or architectural domains: Data Architecture, Application Architecture, Technology Architecture, and Business

---

44 Considering the resemblance between EAP and TAFIM, it is odd that [Spewak92] is absent from the references cited in [TAFIM96].
Architecture [TOGAF95]. Data Architecture, Application Architecture, and Technology Architecture were aligned with TAFIM and Spewak. TOGAF’s Business Architecture aligns with the Business Unit Architecture depicted in the NIST framework. The Open Group recently released TOGAF version 9 [TOGAF09]. Over time, TOGAF has grown to include a methodology, a metamodel and many views.\textsuperscript{45}

5.4.8 C4ISR Architecture Framework

Within the DoD, the C4ISR Architecture Framework marked the transition to a systems architecture framework from the technical architecture framework of TAFIM. When initially published, the C4ISR Architecture Framework was only applicable for C4ISR systems while TAFIM remained applicable for all other DoD information systems. Once DoD expanded the scope of the C4ISR Architecture Framework to all DoD systems in February 1998, TAFIM and other DoD architecture practices and policies were re-evaluated. In January 2000, DoD officially canceled TAFIM.

Initially, the C4ISR Architecture Framework was intended “to define and develop better means and processes for ensuring that Command, Control, Communications, Computers, and Intelligence capabilities meet the needs of warfighters” [DoDAF04] principally by improving interoperability among C4ISR systems. The focus of the C4ISR Architecture

\textsuperscript{45} In addition to sharing three of TOGAF’s four architecture domains, the resemblance between TAFIM v3.0 and TOGAF v9.0 is still evident when comparing TAFIM v3.0 Vol. 4 Figure 1, “The DoD Standards-Based Architecture (SBA) Planning Process.” and TOGAF v9.0 Figure 5-1, “Architecture Development Cycle.”
Framework was on systems architecture. Toward this objective, the C4ISR Architecture Framework specified an All Views, an Operational Architecture, a Systems Architecture, and a Technical Architecture as contexts and specified twenty-six views that revealed subsets of the underlying Core Architecture Data Model (CADM) that, when populated with data, became the de facto architecture. The C4ISR Architecture Framework did not specify viewpoints but recommended that viewpoints be defined and stated for the architecture. The C4ISR Architecture Framework implied a methodology yet intentionally stayed methodologically neutral.

5.4.9 DoDAF

The DoD Architecture Framework (DoDAF) version 1.0 was a name change and a minor evolutionary update to the C4ISR Architecture Framework [DoDAF04]. DoDAF version 1.0 remained essentially unchanged from the C4ISR Architecture Framework version 2.0. DoDAF version 1.0 specified the contexts of All View, Operational View, Systems View, and Technical Standards View. As the original intent of the C4ISR Architecture Framework was very focused on systems architecture, neither of the two versions of the C4ISR Architecture Framework, nor version 1.0 of DoDAF had been intended as an

---

46 These became the Operational Architecture View, Systems Architecture View and Technical Architecture View in C4ISR Architecture Framework version 2.0 and were shortened to Operational View, Systems View and Technical Standards View in DoDAF version 1.0.

47 Paradoxically, one might think that if the authors of these frameworks possessed great wisdom on what architectural artifacts needed to be produced then they would also possess wisdom on how to produce these architectural artifacts.
enterprise architecture framework. This did not preclude the C4ISR Architecture Framework or DoDAF from being employed by enterprise architecture efforts. For example, DoD’s Business Enterprise Architecture effort employed DoDAF with poor results [BEA05]. Constituencies within DoD had recognized DoDAF’s limited scope and strove to expand DoDAF’s capabilities. There was a failed attempt to produce a DoDAF version 2.0 in the 2005-2006 timeframe. Instead, DoD published the incrementally updated DoDAF version 1.5 [DoDAF07] which provided a set of service views as corollaries to the systems views as a way to facilitate the description of service-oriented architectures. This increased the number of views to twenty-nine in DoDAF version 1.5.

In 2009, DoDAF version 2.0 was published [DoDAF09]. DoDAF version 2.0 represents a paradigm shift for the framework as it has grown from the original twenty-six views to fifty-two defined views which DoDAF version 2.0 now calls models, and grants “process owners” carte blanche to include, omit, or create any and all views deemed “fit-for-purpose.”

No longer is DoDAF version 2.0 intended only for systems architecture for the primary purpose of achieving interoperability. Instead, DoDAF version 2.0 [DoDAF09: vol. 1, p. ES-1] is the:

\[
\text{overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate the ability of Department of Defense (DoD) managers at all levels to make key decisions more effectively through organized information sharing across the Department, Joint Capability Areas (JCAs), Mission, Component, and Program boundaries.}
\]

---

48 One can argue that within DoDAF version 2.0 there is no longer any requirement to adhere to the framework as “process owners” are free to include, omit or add any models as long these are deemed “fit-for-purpose” thus, empowering every process owner with the authority to decide what is, or is not, and architecture.
DoDAF version 2.0 includes contexts for All Viewpoint, Operational Viewpoint, Systems Viewpoint, Services Viewpoint, and Standards Viewpoint which are very similar to the All View, Operational View, Systems/Services View, and Technical Standards View contexts specified in DoDAF version 1.5. DoDAF version 2.0 also added contexts for Data and Information Viewpoint, Capability Viewpoint, and Project Viewpoint. Most of what is included in the Capability Viewpoint and Project Viewpoint is new to DoDAF. The Data and Information Viewpoint adds the Conceptual Data Model, which did not exist previously in DoDAF and moved the Logical Data Model (formerly the OV-7) from the Operational View and the Physical Schema (formerly the SV-11) from the Systems/Services View.

In both the DoD and The Open Group cases, the architectural paradigms evolved from a technical architecture framework to a systems architecture framework to an enterprise architecture framework. And while both DoDAF and TOGAF share the same lineage and there is a great deal of similarity between these two architecture frameworks, there is no commonality.

5.4.10 FEA

While both DoDAF and TOGAF evolved from TAFIM, the Federal Enterprise Architecture Framework (FEAF) descended from the NIST enterprise architecture
framework as described in [Fong89] and shown in Figure 10. The Federal Chief Information Officers’ (CIO) Council began developing the FEAF in April 1998. In September 1999, the CIO Council issued the FEAF version 1.1 [CIO99]. John Zachman and Steven Spewak were identified as the only two named members of the Panel of Experts for the FEAF effort. The FEAF used the triangle metaphor from the NIST framework, but substituted Spewak’s *Data Architecture*, *Application[s] Architecture*, *Technology Architecture* contexts and changed *Business Unit Architecture* to *Business Architecture*. The FEAF adopted the rows from Zachman’s framework as the perspectives (or viewpoints) of FEAF. In essence, the FEAF merged NIST, Zachman, and Spewak, and added the concept of segments to turn the two-dimensional metaphor into the three-dimensional metaphor shown in Figure 11. The FEAF did not provide a data model, metamodel, views nor a methodology in [CIO99], but some of these came later with the publication of the FEA reference models.

---

49 It is curious that in 2001 the CIO Council, in *A Practical Guide to Federal Enterprise Architecture*, version 1.0, [CIO01] failed to mention that the FEAF was based on NIST framework, and on page 24 of that document only recognized the NIST framework as “other frameworks exist and have been used in Government programs ….”

50 Subsequent versions of the NIST framework graphic also omit “unit” from the “business unit architecture” label. “Business unit architecture” is how it originally appeared in [Fong89].
5.4.11 Current Purpose and Scope of the FEA, TOGAF and DoDAF

FEA, TOGAF and DoDAF each can trace its origins to the mid-1990s and have continued to evolve towards the goal of specifying an enterprise architecture framework. What follows is how each of these frameworks defines its own purpose and scope.

The FEA states its *raison d'être* as [OMB07: p.1-2]:

*All agencies seek to improve their mission performance. Enterprise architecture is a management practice to maximize the contribution of an agency’s resources, IT investments, and system development activities to achieve its performance goals. Architecture describes clear relationships from strategic goals and objectives through investments to measurable performance improvements for the entire enterprise or a portion (or segment) of the enterprise.*
Enterprise architecture is one of several practice areas that must be executed effectively to achieve improvements in agency mission performance and other measurement areas. EA helps to organize and clarify the relationships between agency strategic goals, investments, business solutions and measurable performance improvements - but, it is just one link in a chain of integrated practice areas. To achieve target performance improvements the EA practice must be strong and fully integrated with other practice areas including strategic planning, capital planning and investment control (CPIC), and program and project management.

EA practice integration means verifying the impact of enterprise architecture on measurable agency performance improvements. This is difficult. Two primary reasons exist: first, there are many links in the chain of integrated practice areas to generate performance improvements; and second, verifying direct relationships between an EA practice and actual agency performance improvements often requires a long period of time. To evaluate the impact of EA products and services, effective EA practice areas measure the value of architectural work products and management processes. This extends to business decisions across all integrated practice areas, monitoring stakeholder feedback and other value indicators to continuously improve the quality of EA products and services to enhance decisions and support business results.

This guidance helps architects to develop and use enterprise architecture to deliver value by:

- Describing the current and future state of the agency and its segments;
- Defining the desired results for an agency and priority segments;
- Determining what resources are used to achieve measurable performance improvements for an agency’s core mission areas and common or shared services;
- Leveraging business and information management resources across the agency;
- Developing a transition strategy to achieve strategic goals and objectives and target performance improvements; and
- Measuring the value of EA products and services to inform decisions in other practice areas and support business results.

Similarly, TOGAF expresses its expectations for enterprise architecture as follows [TOGAF09: pp.6-7]:

The purpose of enterprise architecture is to optimize across the enterprise the often fragmented legacy of processes (both manual and automated) into an
integrated environment that is responsive to change and supportive of the delivery of the business strategy.

Today’s CEOs know that the effective management and exploitation of information through IT is a key factor to business success, and an indispensable means to achieving competitive advantage. An enterprise architecture addresses this need, by providing a strategic context for the evolution of the IT system in response to the constantly changing needs of the business environment.

Furthermore, a good enterprise architecture enables you to achieve the right balance between IT efficiency and business innovation. It allows individual business units to innovate safely in their pursuit of competitive advantage. At the same time, it ensures the needs of the organization for an integrated IT strategy are met, permitting the closest possible synergy across the extended enterprise.

The advantages that result from a good enterprise architecture bring important business benefits, which are clearly visible in the net profit or loss of a company or organization:

• A more efficient IT operation:
  • Lower software development, support, and maintenance costs
  • Increased portability of applications
  • Improved interoperability and easier system and network management
  • Improved ability to address critical enterprise-wide issues like security
  • Easier upgrade and exchange of system components

• Better return on existing investment, reduced risk for future investment:
  • Reduced complexity in IT infrastructure
  • Maximum return on investment in existing IT infrastructure
  • The flexibility to make, buy, or out-source IT solutions
  • Reduced risk overall in new investment, and the costs of IT ownership

• Faster, simpler, and cheaper procurement:
  • Buying decisions are simpler, because the information governing procurement is readily available in a coherent plan.
  • The procurement process is faster – maximizing procurement speed and flexibility without sacrificing architectural coherence.
  • The ability to procure heterogeneous, multi-vendor open systems.

And lastly, DoDAF v2.0 asserts [DoDAF09: pp1-2]:

107
DoDAF V2.0 is the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate DoD managers at all levels to make key decisions more effectively through organized information sharing across Department, Joint Capability Areas (JCAs), Component, and Program boundaries. DoDAF V2.0 focuses on architectural data as information required by key DoD decision makers, rather than on developing individual products. The DoDAF-described Models described in this volume and Volume 2 are used to obtain and visualize data requirements. The framework also enables architecture content to be built that is “Fit-for-Purpose”, as defined and described in Section 1.4. DoDAF is one of the principal pillars supporting the responsibilities Department of Defense Chief Information Officer (DoD CIO) in exercise of his responsibilities for development and maintenance of architectures required under the Clinger-Cohen Act. DoDAF also explains guidance from OMB Circular A-130 and other appropriate DoD directives and instructions; this version of the Framework also provides guidance on the development of architectures supporting the development of Net-centric services within the Department.

DoDAF also serves as the principal guide for development of integrated architectures, as defined in DoD Instruction 4630.82, which states: “An architecture consisting of multiple views or perspectives facilitating integration and promoting interoperability across capabilities and among integrated architectures”. The term integrated means that data utilized in more than one instance in the architectural views is commonly understood across those views.

The vision for utilization of DoDAF is to:

- Provide an overarching set of architecture concepts, guidance, best practices, and methods to enable and facilitate architecture development in support of major decision support processes across all major Departmental programs, Military components, and Capability areas that is consistent and complementary to Federal Enterprise Architecture Guidance, as provided by OMB.

- Support the DoD CIO in defining and institutionalizing the Net-Centric Data Strategy (NCDS) and Net-Centric Services Strategy (NCSS) of the Department, to include the definition, description, development, and execution of services and through introduction of SOA Development.

- Focus on architectural data as information required for making critical decisions rather than emphasizing individual architecture products. Enable architects to provide visualizations of the derived information through combinations of DoDAF-described Models, and Fit-for-Purpose Views commonly used by decision-makers, enabling flexibility to develop those views consistent with the culture and preferences of the organization.
• Provide methods and suggest techniques through which information architects and other developers can create architectures responsive to and supporting Departmental management practices.

While each of these three frameworks identifies noble aspirations for enterprise architecture, in the end, none identify a methodology that assures that any of the anticipated benefits or goals of enterprise architecture can be realized. Achieving value is neither demonstrated nor assured. In the case of the FEA, a combinatorially exploding strategy for producing the Business Reference Model and the Performance Reference Model has impeded many efforts. In the case of TOGAF, there are at least 55 architecture artifacts composed from at least 204 architecture outputs of the Architecture Development Methodology (ADM). In the case of DoDAF v2.0, its scope has been broadened from just systems (solution) architecture to focus on enterprise architecture and now there exists 52 defined models (formerly called views). Additionally, a “process owner” has the latitude to select only a subset of the DoDAF specified models as well as add any other models/views deemed “fit-for-purpose.” What is currently absent from DoDAF is any practical direction as to what purposes are fit for enterprise architecture.

Zachman, FEA, TOGAF and DoDAF are not the only frameworks available to enterprise architects. At this juncture in the maturation of the enterprise architecture discipline, there currently exist a number of frameworks and tools employing frameworks. However, scrutiny suggests there is much less diversity than the number of

51 DoDAF v2.0 was approved for immediate use on 28 May 2009. It was sixteen months later on 17 September 2010 before IBM delivered Rational System Architect v11.4 that could generate DoDAF v2.0 compatible architecture products.
frameworks would indicate. Zachman’s framework\textsuperscript{52} establishes the conceptual metes and bounds for the enterprise architecture problem domain and nearly all other frameworks mimic Zachman’s abstractions or map themselves into Zachman’s matrix.

From open source and consortia, FEAF, TOGAF and DoDAF are the most visible frameworks and possibly the most widely used. FEAF and DoDAF are both public domain and available to anyone with Internet access. The use of both FEAF and DoDAF is required either by law or regulation for U.S. Government departments and agencies thus generating a reasonably large user community for both of these frameworks. DoDAF has been used as the basis for other frameworks including the United Kingdom’s Ministry of Defence Architecture Framework (MODAF), NATO’s NATO Architecture Framework (NAF), Canada’s DND/CF Architecture Framework (DNDAF), and Australia’s Defence Architecture Framework (DAF). Each of these derivative frameworks is very similar to DoDAF. The IDEAS Group\textsuperscript{53} is working to develop a data exchange format between architectures developed using DoDAF, MODAF, DNDAF, and DAF with the goal of exporting and importing architectures or architecture subsets between tools aligned to the different frameworks.

TOGAF appears to have a fairly large user community in the commercial sector, more so in Europe than in the United States. TOGAF has been and continues to be developed by

\textsuperscript{52} While many profess to follow Zachman’s framework, it is so encompassing that almost any attempt at architecture can be mapped to some part of Zachman’s framework. Therefore I tend to agree with other observations (including Zachman’s own description in [Zachman08]) and conclude that Zachman’s framework falls short of any practical application as a framework.

\textsuperscript{53} International Defence Enterprise Architecture Specification for exchange Group.
a vendor- and technology-neutral consortium comprised of over 350 members at the time of this writing.\textsuperscript{54}

**5.4.12 Other Frameworks**

Two open source or consortia frameworks have flickered to life but now appear to have been extinguished:

- Generalised Enterprise Reference Architecture and Methodology (GERAM) was developed in the 1990s by an International Federation of Automatic Control (IFAC)/International Federation for Information Processing (IFIP) Task Force, but does not appear to have evolved since GERAM version 1.6.3 was published in March 1999.

- Good enough Architecture Methodology (GAME) has all-but-vanished from the Internet. In the past, it was espoused within the U.S. Army as an alternative to what was perceived as DoDAF’s overly comprehensive approach.

Other frameworks exist and can be obtained from consortia or commercial vendors, usually at considerable cost. Some of these frameworks have been derived from either Zachman, Spewak, TAFIM, TOGAF or a hybrid of these approaches while others approach architecture from a different perspective. Typically, the framework *per se* is

\textsuperscript{54} A free copy of TOGAF can be obtained with an educational license, which has restricted use, but otherwise, TOGAF is not free.
not what an enterprise will purchase, but will purchase instead an architecture tool or architecture consulting service (or both) which will include the framework, or choice of frameworks, in the bundle. Examples of these other architecture frameworks are:

- alphabet [sic] is a German company that offers its *planningIT* [sic] enterprise architecture tool which alphabet [sic] classifies as a comprehensive solution for architecture-based information technology planning.

- ArchiMate, although not called a framework, provides an enterprise architecture modeling technique that ArchiMate calls a “language” that is based on a metamodel that conforms to TOGAF.

- Atos Origin offers its *Comprehensive, Landscaped, Enterprise Architecture Representation* (CLEAR) framework for enterprise architecture.

- Capgemini is a management consulting company active in The Open Group that has developed its *Integrated Architecture Framework* (IAF) that conforms to the TOGAF framework.

- Casewise has developed its *Corporate Modeler Suite* that supports enterprise architecture development and analysis.

- COBIT (Control Objectives for Information and related Technology) is a framework originally created by the Information Systems Audit and Control Association (ISACA) and the IT Governance Institute (ITGI) in 1996. COBIT version 4.1 was released in 2007.
• Gartner, a U.S. information technology research and advisory company, offers its Gartner Enterprise Architecture Process and its Gartner Enterprise Architecture Framework as part of Gartner’s consulting service. Gartner also publishes its annual Magic Quadrant for Enterprise Architecture Tools; the most recent being [Gartner09] that evaluates currently available enterprise architecture tools that meet Gartner’s evaluation criteria.

• IBM’s Rational System Architect (formerly Telelogic System Architect and originally Popkin System Architect) is currently the leading enterprise architecture tool according to Gartner [Gartner09]. Popkin System Architect was the original tool that supported the C4ISR Architecture Framework and DoDAF. This support continued with Telelogic System Architect and continues now with IBM’s Rational System Architect. Rational System Architect supports “standard frameworks for industry standard methodologies as well as a flexible framework manager to align with custom methodologies, including Zachman, TOGAF, DODAF, NAF, MODAF, and TMForum NGOSS eTOM/SID.”

There is a significant difference between the capabilities of the DoDAF and commercial variants of Rational System Architect.

---

55 TM Forum’s Solution Frameworks (NGOSS) Business Process Framework (eTOM) and Information Framework (SID). Originally founded as the OSI/Network Management Forum in 1988, the name changed to the TeleManagement Forum in 1998 and later was shortened to the TM Forum. TM Forum is a telecommunications trade association of over 700 companies in 75 countries. IBM is a member of TM Forum.
Architect. [Gartner09] only evaluates the commercial variant of *Rational System Architect*.

- Mega provides the Mega Modeling Suite that includes several tools used for enterprise architecture. The Mega suite can support TOGAF, DoDAF and NAF architectures.
- Metastorm offers its *ProVision* enterprise architecture tool that supports DoDAF, TOGAF, FEAF, the Zachman Framework and some other frameworks.
- Salamander offers its *M4* product for MODAF architectures and its *MooD* product for TOGAF architectures.
- Sparx Systems is an Australian company that offers a Unified Modeling Language (UML) based enterprise architecture tool.
- Stroma Software Ltd. is an English company that developed the Ownership, Business Processes, Applications, Systems, Hardware, and Infrastructure (OBASHI) methodology and framework.
- Sybase offers PowerDesigner for data modeling, information architecture, and enterprise architecture.
• TRAK is a general enterprise architecture framework. It was originally developed to depict the architecture of the London Underground. At the beginning, it was a rail-specific architectural adaptation of MODAF but has since been generalized and was released under open sources licenses in 2010.

• Troux Technologies offers a suite of products centered on its Troux Transformation Platform. The Troux Architect product (formerly Metis) includes a visual modeling environment that is augmented by a number of enterprise architecture metamodels and templates including: Troux Semantics (formerly the Metis Enterprise Architecture Framework or MEAF), the General Enterprise Model (GEM), the Activity Based Model (ABM), UML, and DoDAF. In addition, Troux Architect allows a modeler to extend one of these metamodels or create entirely new metamodels.

There also exists a number of other tools and services focused on business process management (BPM), business intelligence (BI) and portfolio management (PfM) that address at least part of the enterprise architecture problem domain while not identifying themselves as enterprise architecture tools.

5.4.13 Conclusion

It only took fifteen years from when the term enterprise architecture was coined in [Zachman82] until John Zachman proclaimed enterprise architecture as the “issue of the century” in [Zachman97b]. Yet in enterprise architecture’s journey from “pipedream to
mainstream” it has yet to have the desired effect for most of the enterprises that pursue this path. Some of the reasons for these disappointing results can be seen in part by understanding from the prior discussion how enterprise architecture got to where it is today. Architecture frameworks that have evolved from technical architecture framework to systems architecture framework to enterprise architecture framework have created comprehensive frameworks that attempt to accommodate all the levels of abstraction and manor of purpose implied by Zachman and, as a consequence, excel at none.

Since 1986 when the PRISM report [Hammer86] first paradoxically observed that there was already a compelling perception in many companies that there existed “problems which can only be addressed by the development of an architecture,” but also observing that “no two … organizations mean the same thing by architecture when they express the above belief,” a vast amount of time, money and effort has been directed at trying to mature the enterprise architecture discipline. In most cases a considerable amount of effort has been directed at specifying what artifacts must be produced towards the goal of developing an enterprise architecture. Yet the empirical research that has been conducted on enterprise architecture [Hammer96, Ross03, Weill04, Ross06, Ross07, Weill09] reveals that how enterprise architecture is developed and employed within the enterprise is a much stronger determinant for success than what enterprise architecture artifacts are produced. In short, how enterprise architecture is developed is more important than what enterprise architecture products are developed. The significance of this observation was noted in the PRISM report [Hammer86: pp.2-3]:
We have concluded that architectural efforts are attempts by I/S organizations to restore a sense of structure and order into a constantly changing world. The advent of powerful, affordable, and interconnectible distributed computing technologies combined in many cases with rapidly changing business conditions and environments are challenging some of the traditional technical and economic assumptions which have guided the actions of I/S organizations in the past. The resulting uncertainty has created confusion and anxiety, and has left organizations with choices which cannot be evaluated in a straightforward and analytical way. While architecture has been put forward as the method to reduce or eliminate this uncertainty, most of the activities conducted in its name do nothing of the sort.

Two decades later, Jeanne Ross, Peter Weill and David Robertson confirmed the observations of the PRISM report when they summarized the results of their empirical research in [Ross06, p.47]:

*Many companies attack the enterprise architecture exercise with lots of drawings and analysis of both existing and hoped-for systems capabilities. But massive analytical efforts do not focus resources on what matters. The key to effective enterprise architecture is to identify the processes, data, technologies, and customer interfaces that take the operating model from vision to reality.*

Architecture Frameworks are predicated on the assertion that if one collects enough data to populate the models described by the framework, then the enterprise is bound to be able to answer some useful questions. However, this assertion is not borne out in practice. This was revealed by the research of Jeanne Ross *et al.* when they determined that architecture frameworks have no positive correlation to architectural success in terms of enterprise profitability [Ross10]:

*My sense is that if architecture isn’t a top-down initiative, it’s a waste of time. Ultimately, those frameworks you mention might help people take high-level views and manage the details of implementation, but architecture has to move in that direction or it’s worthless. I have no empirical evidence, other than a survey of 103 companies in which just about every management practice we asked about was positively correlated with architecture maturity, which was correlated with profitability. The two exceptions were TQM and architectural frameworks.*
If the focus of enterprise architecture remains, as in Taylor’s day nearly a century ago, to improve the efficiency of the enterprise for the betterment of the enterprise and its stakeholders, then one would hope that architectural lessons would have been learned in the past four decades that lend credence to the assertion in Cobb’s Paradox that “we know why projects fail, and we know how to prevent their failure.” Nevertheless, the most popular architecture frameworks clearly reveal not only that seven centuries of lessons on Occam’s Razor have been ignored, but also contrary to Cobb’s Paradox, we really do not know how to prevent project failure with respect to employing enterprise architecture.
5.5 The MAPPMR Enterprise Architecture Metamodel

Intuitively, an enterprise architecture framework should improve the probability of successfully developing and maintaining a successful\textsuperscript{56} enterprise architecture, yet there is no real evidence to support this conclusion. Conversely, the recent empirical research [Ross03, Weill04, Ross06, Ross07, Weill09, Ross10] indicates that frameworks do not contribute to the architectural maturity of enterprises that excel at enterprise architecture. This being the case, one begins to suspect that frameworks may be more of a liability than an asset, especially in the case of frameworks like FEAF, TOGAF and DoDAF that impose a very large burden on the enterprise in terms of work to engineer them with little or no examples of how all this work should or would benefit the enterprise. Clearly, this is not the intent of the organizations and consortia that have labored long and hard to develop their architecture frameworks. However, there is a rising groundswell of opinion that—especially in regards to FEAF, TOGAF and DoDAF—the efforts required by these frameworks do not yield commensurate results. In other words, it is unlikely that employing these frameworks can yield a positive return on investment.

\textsuperscript{56} A successful enterprise architecture is one that has a positive return on investment.
On the surface, enterprise architecture frameworks seem like a good idea, but instead of providing concise directions that highlight the shortest path to success, these frameworks often provide unconstrained instructions that result in the effort becoming overwhelmed by the enormity of the task. Not surprisingly, these efforts can end in disillusionment and failure. The scope of enterprise architecture frameworks has grown to almost boundless proportions over the course of their evolution. As a consequence of this, enterprise architecture frameworks may be even less likely to be of ultimate benefit to the enterprise.

So it is with some reluctance that this research employs the term “framework.” Ideally, an enterprise architecture framework should aid the architect in achieving conceptual integrity while applying Occam’s Razor to eliminate all unnecessary aspects that do not contribute to the overall objective which Ross et al. has found to be the desired levels of business process integration and business process standardization obtainable within the resource constraints of the enterprise [Ross06].

Towards this goal, this research has developed an enterprise architecture metamodel named the Maintain Accountability, Produce Product, and Manage Resources (MAPPMR) enterprise architecture metamodel. MAPPMR was not derived from systems architecture, software architecture, nor was it obtained from an exhaustive attempt to include anything and everything conceivable. This MAPPMR enterprise architecture metamodel strives to enforce the conceptual integrity of the enterprise architecture and help to reduce complexity by excluding any modeling that can consume
much of the available resources but which does not contribute to the objective of answering enterprise business questions. This MAPPMR enterprise architecture metamodel and methodology (discussed in Section 5.6) reflects the enterprise focus on the enterprise dilemma and organizes the enterprise architecture development in a way that is hopefully resistant to distractions that would divert the focus from the primary enterprise architecture objectives of answering the desired business questions. The MAPPMR enterprise architecture metamodel defines the focusing framework for the enterprise architecture and identifies the taxonomy in terms of physical entities, conceptual entities, and relationships among these entities so that the core set of business questions can be answered.

In trying to temper the manifold complexities of decomposing an enterprise into a useful metamodel that would be advantageous for enterprise architects, this research aligned the MAPPMR enterprise architecture metamodel decomposition to the three fundamental core capabilities identified in Section 3.5. These three core capabilities, maintain accountability, produce product, and manage resources, also align to the past, present, and future stages of the enterprise continuum as depicted in Figure 12.

![Figure 12: Capabilities of the Enterprise Across the Enterprise Continuum](image-url)
The depiction of the enterprise continuum has been iconified in Figure 13 as the topmost abstraction of the MAPPMR enterprise architecture metamodel. This symbolizes the three core capabilities of the enterprise aligned with the past, present and future timeframes that correspond to the three core capabilities of maintain accountability, produce product, and manage resources. This approach provides a visual orientation for the users of the MAPPMR enterprise architecture metamodel.

![Figure 13: Topmost Abstraction of the MAPPMR Enterprise Architecture Metamodel](image)

The structure of the MAPPMR enterprise architecture metamodel was based on the Troux Semantics metamodel that is provided with Troux Architect\(^{57}\) from Troux Technologies,

\(^{57}\) Troux Architect was previously known as Metis. Metis was originally created by the Norwegian company, Metis AS, which was bought by Digital Equipment Corporation and later sold to American...
Inc. The primary motivation for using Troux Architect is that it is a general-purpose modeling tool that provides a rich metamodel in Troux Semantics, yet can be modified and extended as needed to suit the problem domain being modeled. Troux Architect is also a tool that allows the architecture team to employ The Six-step Problem-Focused Question-Based (PFQB) Architecture Process described in [Martin06], where central to the architecture development is determining the questions the architecture must answer and then developing the architecture so that it will, in-fact, answer those questions. The fundamental difference between this and FEAF, TOGAF, DoDAF and DoDAF-like frameworks is that in these the architecture team builds the products and populates the underlying model in hopes of being able to answer relevant questions, whereas with the MAPPMR approach the enterprise ascertains the relevant business questions and then builds a model that will answer those business questions.

The remainder of this section is a description of the decomposition of the MAPPMR enterprise architecture metamodel.

Figure 14 shows the topmost abstraction of the MAPPMR enterprise architecture metamodel which is subdivided into three containers that align to the three core capabilities of maintain accountability, produce product, and manage resources.
5.5.1 The MAPPMR Enterprise Architecture Metamodel for Maintain Accountability Core Capability

Figure 15 shows the topmost abstraction of the MAPPMR enterprise architecture metamodel alongside the decomposition of the maintain accountability core capability. The maintain accountability core capability includes the Data Domain, Document Domain, Financial Domain, Information Domain, Knowledge and Skill Domain and Location Domain.
Each of the domains shown in the maintain accountability core capability in Figure 15 includes entity types identified below as specified in the Troux Semantics metamodel [Troux11]:

**Data Domain**

*Data Component*
- Database Object
- Database Column
- Database Schema
- Database Table
- Database Type
- Database View
- Datastore
- Database
- File
Document Domain

Document Component
- Document
- Online Document
- Document Class

Financial Domain

Financial Component
- Standard Cost
- Standard Cost Element

Control Component
- Charter
- Contract

Information Domain

Information Component
- Concept
- Information Object
- Information Attribute
- Information Group

Demand Component
- Need
- Information Need
- Business Question

Knowledge and Skills Domain

Knowledge Component
- Knowledge Access
- Knowledge Element
- Knowledge Source
- Skill
- Formal Skill
- Skill Profile
- Training Program

Location Domain

Location Component
- Location
- Logical Location
5.5.2 The MAPPMR Enterprise Architecture Metamodel for Produce Product Core Capability

Figure 16 shows the topmost abstraction of the MAPPMR enterprise architecture metamodel alongside the decomposition of the produce product core capability. The produce product core capability includes the Application and Software Domain, Infrastructure and Hardware Domain, IT Patterns Domain, IT Product Domain, IT Service Domain, Organization Domain, Process Domain, Product and Service Domain and the Systems Domain.

![Figure 16: Metamodel Decomposition of the Produce Product Core Capability](image-url)
Each of the domains shown in the **produce product** core capability in Figure 16 includes entity types identified below as specified in the Troux Semantics metamodel [Troux11]:

**Application and Software Domain**

*Application Component*

Software

- Software Patch
- Deployed Software

Application

- Software Module
- Operating System
- Software Server

- Application Server
- Database Server
- File Server
- Web Server

- Virtual Machine Image
- Virtual Host

*Category (from General Domain)*

- APO Category

*Process Component (from Process Domain)*

*Function (from Process Domain)*

- IT Function (from Process Domain)

- Application Function

*Interaction Component*

- Message

**Infrastructure and Hardware Domain**

*Process Component (from Process Domain)*

*Function (from Process Domain)*

- IT Function (from Process Domain)

- Infrastructure Function

*Infrastructure Component*

- Hardware

- Cabinet
- Computing Hardware

- Computer

- Desktop
- Laptop
- Server

- Mainframe

- Network Device

128
Firewall
Load Balancer
Router
Storage Device
Backup Device
Tape Backup
Disk Array
Switch
SAN Switch

Network
Network Interface
Storage Area Network

**IT Pattern Domain**

*Category (from General Domain)*
Design Pattern Category

**IT Product Domain**

*Offer Component*

*Deliverable (from Product and Service Domain)*

*Product (from Product and Service Domain)*
IT Product
Software Product
Hardware Product

*Product Version (from Product and Service Domain)*
IT Product Version
Software Product Version
Hardware Product Model

**IT Service Domain**

*Offer Component*

*Deliverable (from Product and Service Domain)*

*Service (from Product and Service Domain)*
IT Service

*Catalog (from Product and Service Domain)*
IT Service Catalog

**Organization Domain**

*Organization Component*
Organization
Internal Organization
External Organization

129
Person
Position
Role
Control Component
Privilege
Interaction Component
Communication

Process Domain

Process Component
Function
   Business Function
   IT Function
Business Process
   Process
      Production Line
Role
Interaction Component
Process Interaction
   Process Control
   Process Input
   Process Mechanism
   Process Output
Process Data
Process Event
   Process End Event
   Process Intermediate Event
   Process Start Event
   Process Gateway

Product and Service Domain

Offer Component
Catalog
Deliverable
   Product
      Business Product
   Product Version
      Business Product Version
Service
   Business Service
Product Method
Product Module
Product Property
Control Component
5.5.3 The MAPPMR Enterprise Architecture Metamodel for Manage Resources Core Capability

Figure 17 shows the topmost abstraction of the MAPPMR enterprise architecture metamodel alongside the decomposition of the manage resources core capability. The manage resources core capability includes the Analysis Domain, IT Architecture Domain, Market Domain, Policy Domain, Resource Domain, Services Portfolio Management Domain, Strategy Domain, Timeline Domain and Transition Domain.
Each of the domains shown in the manage resources core capability in Figure 17 includes entity types identified below as specified in the Troux Semantics metamodel [Troux11]:

**Analysis Domain**

*Analysis Component*

Action
Analysis
Factor

Cause
External Factor
Implication
Issue
Finding
Recommendation
Risk
Trend

Control Component
Constraint

IT Architecture Domain

IT Architecture Component
Industry Standard
IT Architecture
IT Architecture Element
Logical Application
Logical Datastore
Logical Technology Item
Technology
Industry Standard

Market Domain

Demand Component
Benefit
Market Opportunity
Market Segment
Need
Prerequisite
Requirement
Scenario
Target Customer
Target User

Policy Domain

Policy Component
Design Principle
Enterprise Policy
Security Policy
Policy Rule
Security Policy Rule
Standard
Standard Candidate
Category (from General Domain)
Standards Category
Policy Exception

Principle
Architecture Principle

Resource Domain
Resource Component
  Resource
    Manpower

Services Portfolio Management Domain

Service Component
  Service Catalog
  Service Level Agreement
    Service Level Agreement Template
    Deployed Service Level Agreement

Strategy Domain

Strategy Component
  Goal
  Strategy
  Success Factor
  Vision

Offer Component
  Capability

Control Component
  Measure
    Key Performance Indicator
  Mission
  Value Chain

Timeline Domain

Timeline Component
  Timeline
  Timeline Event
  Timeline Milestone

Diagram
  Event
    Incident
  Lifecycle
  Milestone
  Phase
  Time
  Transition

Change Component
  Initiative
  Program
  Project
  Task

Control Component
Plan

Disaster Recovery Plan

5.5.4 Conclusion

This research has developed the MAPPMR enterprise architecture metamodel with the following goals in mind:

- Defining a focusing framework for the enterprise architecture that is the basis for identifying a taxonomy in terms of physical entities, conceptual entities, and relationships among these entities so that the core set of business questions can be answered.

- Reducing complexity by minimizing any modeling that can consume much of the available resources without contributing to the objective of answering the business questions,

- Reflecting the enterprise focus and organizing the enterprise architecture development in a way that is resistant to distractions that would divert the focus from the primary enterprise architecture objectives of answering the desired business questions,

- Aligning the MAPPMR enterprise architecture metamodel decomposition to the three fundamental core capabilities of maintain accountability, produce product, and manage resources.
The next section will present a strategy for populating the MAPPMR enterprise architecture metamodel so that it actually results in an enterprise architecture that represents what is desired for the enterprise and is thus capable of being a useful tool for evolving the enterprise towards its desired objective.
Since John Zachman first opined about the need for enterprise architecture in [Zachman82], the allure of enterprise architecture has grown stronger as technology has advanced. Motivated by anticipations of either technological obsolescence (glass half empty) or of untold windfall (glass half full), or both, an architectural gold rush began in the late twentieth century. Enterprise architecture became the vogue as enterprise after enterprise embarked on quests to find the alchemy that could catapult the enterprise to market dominance. Typically, efforts would begin with optimism and confidence in the nobility of the cause, yet few enterprise architectures would ever return value that approached the cost of the architecture effort itself, let alone find significant value for the enterprise.

In general, the architecture frameworks that emerged over the past two decades have failed to illuminate a path to success for those seeking enterprise improvement through architectural enlightenment. This is illustrated by the evolution of DoDAF. In order to accommodate enterprise architecture, DoDAF v2.0 doubled its viewpoints from earlier versions from four to eight, doubled the defined models/views from twenty-six to fifty-two, and now allows an unbounded set of user-defined models/views, yet declares that no
model/view is required except as desired by the “process owner” as deemed “fit for purpose.” We must be concerned that if there is a plethora of specified models/views, an unbounded possibility of other models/views, and no minimal set of models/views is required, then what actually constitutes an enterprise architecture is not established. It appears that like beauty, what constitutes enterprise architecture is in the eye of the beholder. Unfortunately, under scrutiny, these ill-defined approaches to architecture (enterprise or otherwise) generally deliver ill-defined and unproductive results. The purpose of this section is to define a method that is both purposeful and adequate for developing enterprise architecture that can provide value for the enterprise.

5.6.1 Philosophy on People and Process

Succeeding with an enterprise architecture requires the involvement of people adhering to a process with the ultimate goal of developing, maintaining and evolving the architecture of the enterprise. However, success can be elusive even for an effort that can justifiably argue that it has the right people following the right process. Having both the right people (people with the right responsibilities, authorities, knowledge and skills) and the right process are critically important, but involving the right people takes precedence as no amount of process will enable the wrong group of people to succeed at enterprise architecture. This research has witnessed a number of architecture efforts that have been challenged by what has been termed the “curse of the surrogate stakeholder.”
This situation arises when an organization decides it needs an enterprise architecture. Typically what occurs is that someone within the organization is tasked with making the enterprise architecture development happen and that person delegates the actual task of developing the enterprise architecture to some subordinate(s) or contractor(s). At the architecture effort’s kickoff meeting, the person tasked with developing the enterprise architecture presents the welcome and administrative remarks and then quickly departs for other “more pressing” tasks. What results is a room full of “stakeholders,” none of whom could be described as empowered stakeholders that possess the requisite responsibility and authority to make the decisions needed to begin meaningful enterprise architecture development. While each individual in the room may indeed be a stakeholder in the sense that something about the system or enterprise impacts them, none of these stakeholders have the authority, responsibility or perspective to make the decisions that will shape the resultant enterprise architecture. In essence this is a committee of surrogates that are neither authorized nor empowered to make the requisite decisions needed to develop the enterprise architecture. Unfortunately, this is a very frequent occurrence in U.S. government organizations. This observation is not meant to denigrate the individuals in the room as they are usually highly knowledgeable, skilled and motivated individuals that want to do right for the enterprise. However, these surrogate stakeholders are not the “right” individuals to do what needs to be done. In effect, this is a situation where anyone in the room can say no, but it takes everyone in the room to agree on yes and even then that is only a recommendation for some empowered
stakeholder that is not in the room, not involved in the discussions, and has not developed a “shared understanding” with those making the recommendations.

Detailed strategies for determining empowered stakeholders, organizing the stakeholder groups, and selecting the methods for leading groups of stakeholders towards a shared understanding of the enterprise consisting of the enterprise’s present state and the enterprise’s desired future state are the subject of ongoing research in multiple disciplines. [Conklin06], [Burlton01], and [Burton06] provide excellent insight into group dynamics and building a shared understanding within the enterprise. The disciplines of group dynamics and organizational development, while beyond the scope of this research, have nonetheless influenced the outcome of this research. Understanding and managing these issues are key to obtaining a successful enterprise architecture and cannot be ignored in practice. Thus these issues cannot be ignored here.

As for process, the software domain has struggled with process from its origins, yet every significant study reveals that people contribute much more to the success of the effort than the process, the tools or the techniques\textsuperscript{58} employed. Enterprise architecture suffers similarly. There is no such thing as a “just add people” process that will result in a successful enterprise architecture. Enterprises that succeed at enterprise architecture do so as a result of the people pursuing architecture, not as a result of the process employed in the pursuit of architecture.

\textsuperscript{58} For a discussion of this problem as it relates to software development see [Glass03].
This research describes a process that will help focus the architecture team’s efforts towards the goal of more quickly developing an enterprise architecture that answers business questions that are relevant to addressing the enterprise dilemma and improving the chances that the architecture effort will succeed in yielding value for the enterprise. However, process alone is never sufficient. The people executing the process will always be the dominating influence towards a successful outcome.

5.6.2 Redefining Enterprise Architecture

The definitions of architecture, software architecture, information technology architecture, and enterprise architecture that have been applied by the U.S. Government are discussed in Sections 2.4 and 2.5. While none of these definitions can be considered incorrect in a broad sense, the enterprise architecture definitions define enterprise architecture in an inclusive way that does not focus an enterprise architecture effort on the activities that will add the most value. This research has applied Occam’s Razor to develop a definition of enterprise architecture that emphasizes the activities that correlate most strongly with establishing the conceptual integrity requisite for enterprise architecture success followed by the development of business questions that yield valuable insights quickly and from which increased value can be derived over time.

James Martin established the purpose of enterprise architecture as follows [Martin06: p.3]:

141
Enterprise architectures are developed for the purpose of discovering more efficient operational approaches for a business venture. Enterprise architecture models are developed to give business managers a better understanding of the things the enterprise owns, operates, and produces so they can make better business decisions.

This statement of purpose for enterprise architecture establishes the context for the definition of enterprise architecture. Accordingly and within this context, this research has applied Occam’s Razor and redefines enterprise architecture as follows:

An enterprise architecture is a tool that assists the enterprise in addressing the enterprise dilemma. The purpose of enterprise architecture is to provide the enterprise a better understanding of the things the enterprise produces, owns and operates so the enterprise can make better decisions regarding sustaining or changing the enterprise. An enterprise architecture contains:

- The statement of purpose for the enterprise,
- The definition of the bounds of the enterprise,
- The description of the desired operating model(s) for the enterprise to include the desired levels of business process integration and business process standardization,
- The information technology principles for the enterprise,
- The core diagram for the enterprise,
- A set of enterprise business questions developed to help the enterprise address the enterprise dilemma as it evolves from its current state to the desired state
in concordance with the core diagram, its desired operating model(s) and in compliance with the enterprise principles.

The strength of this enterprise architecture definition is that it specifies a context for enterprise architecture that is separate and distinct from systems architecture. It also identifies the benchmarks needed to establish the requisite conceptual integrity at the beginning of the enterprise architecture effort and leads to an iterative methodology that can deliver value quickly and allow the value to grow incrementally. Another benefit is that this methodology is “failure tolerant” in that it is not an approach that fails to deliver value until it is substantially completed, thus avoiding the “big failure at the end” situation that arises from most enterprise architecture frameworks.

The following section outlines a focusing methodology for enterprise architecture development is consistent with and aligned to this definition of enterprise architecture.

5.6.3 Enterprise Architecture Development Process

James Martin developed The Six-step Problem-Focused Question-Based (PFQB) Architecture Process for architecture development in [Martin06]. This research extends Martin’s PFQB architecture process to specifically address enterprise architecture development. Figure 5 (in Section 5.3) shows the top-level abstraction for The Six-step Problem-Focused Question-Based (PFQB) Architecture Process. These six steps, as defined by Martin in [Martin06], expand to include the following activities:

Step 1: Problem Framing
• Business Analysis
  o Purpose Identification
  o Business Question Definition
  o Business Objectives Prioritization
• Conceptual Definition of the Problem
  o Business Objects Identification
  o Business Facts Identification
  o Conceptual Schema Definition
  o Architectural Framework Development
  o Conceptual Definition Validation

Step 2: Metamodel Development
• Metamodel Specification
• Analytics & Queries
• Model Prototyping

Step 3: Data Collection
• Survey Development
• Data Call

Step 4: Model Development
• Data Import
• Query Development
• Tool Integration

Step 5: Architecture Deployment
• Deployment Strategy
• View Development
• Training & Tutorials

Step 6: Architecture Utilization
• Business Analysis
• Prototype Testing
• Iteration
To tailor Martin’s six-step process specifically for enterprise architecture, the PFQB Architecture Process has been extended to include the additional activities shown in bold italics:

Step 1: Problem Framing

- Business Analysis
  - Purpose Identification
    - Define the purpose of the enterprise,
    - Determine the bounds of the enterprise,
    - Develop the desired operating model(s) for the enterprise to include the desired levels of business process integration and business process standardization,
    - Determine the information technology principles of the enterprise,
    - Develop the core diagram for the enterprise.
  - Business Question Definition
  - Business Objectives Prioritization
  - Conceptual Definition of the Problem
    - Business Objects Identification
    - Business Facts Identification
    - Conceptual Schema Definition
    - Architectural Framework Development
    - Conceptual Definition Validation

Step 2: Metamodel Development

- Metamodel Specification\(^{59}\)
  - Define the enterprise:
    - The statement of purpose for the enterprise,

\(^{59}\) The Appendix contains an example metamodel specification.
• The bounds of the enterprise,
• The desired operating model(s) to include the desired levels of business process integration and business process standardization,
• The information technology principles,
• The core diagram.

○ Define the enterprise architecture metamodel for the enterprise:
  • The purpose of the enterprise Architecture,
  • Explanation of the role of business questions in metamodeling,
  • The business questions that the enterprise architecture is being developed to answer,
    ○ The business questions implemented in this increment,
    ○ The business questions implemented in previous increments.

○ Define the enterprise architecture metamodel objects & relationships:
  • Enterprise architecture metamodel schema
  • Enterprise architecture metamodel taxonomy
  • Entities & relationships used in the enterprise architecture metamodel
  • Objects used in the enterprise architecture metamodel

○ Define the enterprise architecture methods

○ Define the enterprise architecture criteria (queries)

○ Define the enterprise architecture views
  • Analytics & Queries
  • Model Prototyping

Step 3: Data Collection
  • Survey Development
  • Data Call

146
Step 4: Model Development
- Data Import
- Query Development
- Tool Integration

Step 5: Architecture Deployment
- Deployment Strategy
- View Development
- Training & Tutorials

Step 6: Architecture Utilization
- Business Analysis
- Prototype Testing
- Iteration

5.6.4 Applying the Focusing Methodology for Enterprise Architecture Development

To demonstrate that the hypothesis is workable and that it presents a feasible approach for enterprise architecture development, the application of this focusing methodology for enterprise architecture development is illustrated with the development of an enterprise architecture. The remainder of this section will chronicle the development of an enterprise architecture applying the methodology presented in Section 5.6.3, employing the MAPPMR Enterprise Architecture Metamodel described in Section 5.5, and applying the core set of enterprise business questions described in Section 5.3.
The NASA\textsuperscript{60} Science Mission Directorate (SMD)\textsuperscript{61} has been chosen for this example owing to NASA’s extensive public outreach resulting in a wealth of information being available in the public domain. NASA SMD is one of four NASA directorates.

5.6.4.1 Before Determining How to Get There, First Determine the Destination

Martin defines Step 1 of his *PFQB Architecture Process* as *Problem Framing* and includes *Business Analysis* and *Conceptual Definition of the Problem* as the two primary activities for this step [Martin06]. Each of these two activities is in turn decomposed into sub-tasks. In order to specialize Martin’s approach for developing enterprise architecture, this research has identified activities that correlate highly to enterprise architecture success. As noted earlier, Jeanne Ross *et al.*, in [Ross06, p.47] identified what they consider the primary focus of enterprise architecture; this focus provides clues on where to begin an enterprise architecture effort:

*The key to effective enterprise architecture is to identify the processes, data, technologies, and customer interfaces that take the operating model from vision to reality.*

However, when beginning a new enterprise architecture effort, or when beginning an effort to refresh any existing enterprise architecture, there are a few benchmarks that

\textsuperscript{60} NASA is the United States National Aeronautics and Space Administration.

\textsuperscript{61} This enterprise architecture was developed solely from publicly accessible information to illustrate the enterprise architecture development process. Any resemblance to any actual enterprise architecture for NASA or NASA’s Science Mission Directorate is purely coincidental.
should be established prior to drilling down into processes, data, technologies, and interfaces. The activities that establish these benchmarks are:

- Define the purpose of the enterprise,
- Determine the bounds of the enterprise,
- Define the desired operating model(s) of the enterprise,
- Establish information technology principles for the enterprise, and
- Develop the core diagram for the enterprise.

These activities establish the conceptual integrity needed to establish the shared understanding within the enterprise. The following illustrates how the enterprise architecture problem might be framed for NASA’s SMD:

Step 1: Problem Framing

- Business Analysis
  - Purpose Identification
    - Define the purpose of the enterprise,
    - Determine the bounds of the enterprise,

---

62 Enterprise-architecture development is never a linear process. Any enterprise is a dynamic organization that employs rapidly evolving technology. Keeping pace with this change is always a challenge that requires that all aspects of the enterprise architecture be continually reevaluated to ensure that all remains in concordance. Also, since much of the enterprise is composed of wicked problems, an unwieldy characteristic of a wicked problem is that any solution for a wicked problem changes the formulation of the problem, thus begging for a new solution. The path to the future is never straight and narrow.
• Develop the desired operating model(s) for the enterprise to include the desired levels of business process integration and business process standardization,

• Determine the information technology principles of the enterprise,

• Develop the core diagram for the enterprise.

5.6.4.1.1 Define the Purpose of the Enterprise

This activity focuses on establishing the purpose of the enterprise itself as described in [Rudzki07: p.19] as the “… fundamental reason-to-be underlying the organizations existence” and should not to be confused with establishing the purpose of the architecture as described in [Martin06: p.116]. A clear statement of purpose for the enterprise needs to exist and is key to establishing the conceptual integrity for the enterprise and thus for the enterprise architecture. A top-down decomposition should actually begin at the top and this step is what defines the top-most starting point for enterprise architecture development. Any enterprise architecture effort that fails to articulate a clear and concise statement of purpose for the enterprise will likely need to revisit the step again in the future. Similarly, if the purpose of the enterprise changes, the enterprise will need to re-align the enterprise architecture to adapt to the new purpose.
In the example, NASA SMD is constrained by NASA’s overarching vision:

*To reach for new heights and reveal the unknown so that what we do and learn will benefit all humankind.*

To help achieve this NASA vision, the purpose of the NASA SMD enterprise is to explore the Earth, solar system and universe beyond; chart the best route of discovery; and reap the benefits of Earth and space exploration for society. To achieve this purpose, SMD employs:

*Space observatories to conduct scientific studies of the Earth from space to visit and return samples from other bodies in the solar system, and to peer out into our Galaxy and beyond. NASA’s science program seeks answers to profound questions that touch us all:*

- How and why are Earth's climate and the environment changing?
- How and why does the Sun vary and affect Earth and the rest of the solar system?
- How do planets and life originate?
- How does the universe work, and what are its origin and destiny?
- Are we alone?*

In aggregate, the purpose of the NASA SMD is intended to help SMD contribute its part to achieving the NASA science vision:

*Using the vantage point of space to achieve with the science community and our partners a deep scientific understanding of our planet, other planets and solar system bodies, the interplanetary environment, the Sun and its effects on the solar system, and the universe beyond. In so doing, we lay the intellectual foundation for the robotic and human expeditions of the future while meeting today's needs for scientific information to address national concerns, such as climate change and space weather. At every step we share the journey of scientific exploration*

---

63 http://www.nasa.gov/about/highlights/what_does_nasa_do.html

64 http://science.nasa.gov/about-us/
with the public and partner with others to substantially improve science, technology, engineering and mathematics (STEM) education nationwide.\textsuperscript{65}

5.6.4.1.2 Determine the Bounds of the Enterprise

Determining the bounds of the enterprise is important because there needs to be a clear identification of what is and is not the enterprise, especially with respect to multi-organizational enterprises\textsuperscript{66} or multiple enterprises that are identified within a single organization such as the NASA SMD example. Enterprise architectures can and probably should depict more than just the enterprise itself. Defining where the boundaries of the enterprise lie will help differentiate endogenous from exogenous influences on the enterprise. It will also allow the enterprise to focus on the endogenous influences over which it has control as well as staying aware of the exogenous influences, which often imply risk, might be mitigated, but probably cannot be directly controlled by the enterprise.

NASA has four mission directorates: Aeronautics Research, Exploration Systems, Space Operations and Science. This example NASA SMD Enterprise Architecture applies only to the Science Mission Directorate. Nevertheless, NASA SMD must continually strive to ensure that its enterprise architecture remains in concordance with the NASA Enterprise

\textsuperscript{65} http://science.nasa.gov/about-us/

\textsuperscript{66} This is what Linda Vandergriff calls a “complex venture” in [Vandergriff06].
Architecture. Therefore, the NASA SMD Enterprise Architecture must propagate all the constraints imposed by higher authority.

5.6.4.1.3 Define the Desired Operating Model(s) of the Enterprise

Once the purpose and bounds of the enterprise are established, then the desired operating model(s)\textsuperscript{67} can be defined for the enterprise as described in [Ross06] and [Weill09]. The purpose of this particular activity is to ascertain the necessary level of business process integration and business process standardization desired for the enterprise. Ross et al. depicted the operating model continuum in a quadrant shown in Figure 18 [Ross06]. Each quadrant represents either a low-low, low-high, high-low or high-high combination of business process integration and business process standardization. Figure 18 identifies some of the characteristics each combination implies. However, attempting to decide on the operating model(s) is much more difficult without establishing the associated principles upon which to base this decision. This is a “chicken or egg” problem since the principles chosen must align with the characteristics of the desired operating model. Care must be exercised to ensure that the desired principles chosen by the enterprise support the desired operating model(s) and \textit{vice versa}.

\textsuperscript{67} It is entirely possible to have multiple operating models corresponding to various subsets of the enterprise. Do not assume a homogenous solution will be appropriate for the entire enterprise unless such a solution best satisfies the principles established by the enterprise.
Characteristics of four operating models

<table>
<thead>
<tr>
<th>Coordination</th>
<th>Unification</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Shared customers, products, or suppliers</td>
<td>• Customers and suppliers may be local or global</td>
</tr>
<tr>
<td>• Impact on other business unit transactions</td>
<td>• Globally integrated business processes often with support of enterprise systems</td>
</tr>
<tr>
<td>• Operationally unique business units or functions</td>
<td>• Business units with similar or overlapping operations</td>
</tr>
<tr>
<td>• Autonomous business management</td>
<td>• Centralized management often applying functional/process/business unit matrices</td>
</tr>
<tr>
<td>• Business unit control over business process design</td>
<td>• High-level process owners design standardized processes</td>
</tr>
<tr>
<td>• Shared customer/supplier/product data</td>
<td>• Centrally mandated databases</td>
</tr>
<tr>
<td>• Consensus processes for designing IT infrastructure services; IT application decisions made in business units</td>
<td>• IT decisions made centrally</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Business process standardization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
</tr>
<tr>
<td>Few, if any, shared customers or suppliers</td>
</tr>
<tr>
<td>Independent transactions</td>
</tr>
<tr>
<td>Operationally unique business units</td>
</tr>
<tr>
<td>Autonomous business management</td>
</tr>
<tr>
<td>Business unit control over business process design</td>
</tr>
<tr>
<td>Few data standards across business units</td>
</tr>
<tr>
<td>Most IT decisions made within business units</td>
</tr>
</tbody>
</table>

Figure 18: Characteristics of Four Operating Models [Ross06: p.29]

For the NASA SMD Enterprise Architecture example, the directorate is comprised of four science domains: Earth Science, Heliophysics, Planetary Science, and Astrophysics.
The goal is to evolve to a high state of business process integration between the science domains, but NASA SMD does not necessarily require a high degree of business process standardization between the science domains. This puts NASA SMD in the upper left quadrant of Figure 18 in terms of desired operating model. Within a science domain, the goal is again for a high level of business process integration and a moderate to high level of business process standardization. The objective is to provide a common look and feel across all the science domains for NASA SMD’s customers and partners while enabling unique approaches to solving problems that are unique to each science domain. Therefore, each NASA SMD science domain would be placed in the upper right quadrant of Figure 18 in terms of the desired operating model for each of the four science domains.

5.6.4.1.4 Establish Information Technology Principles for the Enterprise

Like all undertakings, the likelihood of obtaining the desired outcome is heavily dependent on having a desired outcome. This may seem trite and obvious, but in practice “desired outcomes” are rarely identified in a way that can be coherently represented in the enterprise architecture.

All the major empirical studies of enterprise architecture [Hammer86, Weill04, Ross06] found that information technology principles were critical elements of successful enterprise architecture efforts. [Hammer86] first reported the significant role information
technology principles play towards the goal of successful enterprise architecture.

Thomas Davenport et al. concluded in [Davenport89]:

>We have studied the IT decision-making processes of more than 50 large organizations, many of whose IT efforts were lacking direction. A few companies, however, had articulated their basic philosophies about IT, and they seemed to be using technology more effectively. They expressed these philosophies through a set of IT management principles that summarized how the company would use IT to achieve its goals. Those principles then guided any technology decisions that arose over the next few years. If the decision was in keeping with the principles, it was also in keeping with the corporate strategy.

Principles are simple, direct statements of an organization’s basic beliefs about how the company wants to use IT over the long term. By translating the main aspects of a company’s business strategy into the language of technology managers, these principles bridge the communication gap between top managers and technical experts. This way, business strategy drives technical strategy, as conventional wisdom says it should.

Weill and Ross et al. also found a high degree of correlation between enterprise architecture success and having established information technology principles [Weill04, Ross06]. Peter Weill et al. noted in [Weill04: p27]:

>Study after study demonstrates that enterprises achieving superior business value from IT have a small number of clearly articulated IT principles. IT principles are a related set of high-level statements about how IT is used in the business. Once articulated, IT principles become part of the enterprise’s management lexicon and can be discussed, debated, supported, overturned, and evolved.

Information technology principles, when properly articulated by the enterprise, become the de facto “desired outcomes” which the enterprise can strive to achieve and collectively establish the role of information technology within the enterprise and thus will highly influence the decisions affecting the enterprise’s information technology infrastructure. Given a vision for how the enterprise desires to operate (what Ross et al. calls the “operating model” in [Ross06]) that is based on information technology
principles, enterprise architecture can guide the enterprise in its transition from today’s vision to tomorrow’s reality. Enterprise architecture will not tell the enterprise what the operating model should be. The enterprise must determine the desired operating model(s) as a prerequisite for a successful enterprise architecture effort.

Unfortunately, in practice it is extremely difficult to develop a concise set of information technology principles that qualify as “simple, direct statements of an organization’s basic beliefs about how the company wants to use IT over the long term,” [Davenport89] or as a “related set of high-level statements about how IT is used in the business” [Weill04]. Divining the set of principles that will guide the enterprise architecture effort is at the heart of the first wicked problem to be addressed. As fundamental as having guiding principles seems, it is often overlooked and certainly undervalued at the beginning of an enterprise architecture effort. Confoundingly, principles are not explicitly represented in the DoDAF or FEAF frameworks. Therefore, it is common for enterprise architecture efforts to bypass this key step in the rush to finding “the solution.” Identifying information technology principles at the beginning of an enterprise architecture development effort will help ensure that the enterprise addresses the big-picture issues that establish the requisite conceptual integrity for the enterprise.

For the NASA SMD Enterprise Architecture example, no IT principles for NASA or NASA SMD were found in the public domain. Instead, the twenty-one principles

---

68 DoDAF v2.0 documentation discusses “principles” numerous times, but does not appear to accommodate information technology principles in any of the fifty-two defined models in this framework. A work-around in DoDAF is to model principles as “rules” within the DoDAF metamodel (DM2), but then principles are not readily distinguishable from other rules, except maybe by a naming convention.
identified in TOGAF v9 [TOGAF09: pp269-280] will be applied. These TOGAF principles are:

**Principle 1:** Principles of information management apply to all organizations within the enterprise.

**Principle 2:** Information management decisions are made to provide maximum benefit to the enterprise as a whole.

**Principle 3:** All organizations in the enterprise participate in information management decisions needed to accomplish business objectives.

**Principle 4:** Enterprise operations are maintained in spite of system interruptions.

**Principle 5:** Development of applications used across the enterprise is preferred over the development of similar or duplicative applications which are only provided to a particular organization.

**Principle 6:** The architecture is based on a design of services which mirror real-world business activities comprising the enterprise (or inter-enterprise) business processes.

**Principle 7:** Enterprise information management processes comply with all relevant laws, policies, and regulations.

**Principle 8:** The IT organization is responsible for owning and implementing IT processes and infrastructure that enable solutions to meet user-defined requirements for functionality, service levels, cost, and delivery timing.

**Principle 9:** The enterprise’s Intellectual Property (IP) must be protected. This protection must be reflected in the IT architecture, implementation, and governance processes.

**Principle 10:** Data is an asset that has value to the enterprise and is managed accordingly.

**Principle 11:** Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations.

**Principle 12:** Data is accessible for users to perform their functions.

**Principle 13:** Each data element has a trustee accountable for data quality.
Principle 14: Data is defined consistently throughout the enterprise, and the definitions are understandable and available to all users.

Principle 15: Data is protected from unauthorized use and disclosure. In addition to the traditional aspects of national security classification, this includes, but is not limited to, protection of pre-decisional, sensitive, source selection-sensitive, and proprietary information.

Principle 16: Applications are independent of specific technology choices and therefore can operate on a variety of technology platforms.

Principle 17: Applications are easy to use. The underlying technology is transparent to users, so they can concentrate on tasks at hand.

Principle 18: Only in response to business needs are changes to applications and technology made.

Principle 19: Changes to the enterprise information environment are implemented in a timely manner.

Principle 20: Technological diversity is controlled to minimize the non-trivial cost of maintaining expertise in and connectivity between multiple processing environments.

Principle 21: Software and hardware should conform to defined standards that promote interoperability for data, applications, and technology.

5.6.4.1.5 Develop the Core Diagram for the Enterprise

Developing a core diagram for the enterprise is neither an easy nor straightforward task. Ross et al. noted that many enterprises struggle mightily with developing the core diagram [Ross06]. While there may be many architecture views, products, and artifacts in the enterprise architecture, the core diagram is the central focus of the enterprise architecture and the closest thing to “… a single and coherent model of an enterprise …” as suggested by Khoury [Khoury04]. The core diagram should capture the essence of the enterprise’s purpose, boundaries and operating model(s) and if it is not quite the single,
coherent model of the enterprise, then it is at least the single starting point and gateway into the other models that complete and complement the enterprise architecture.

For the NASA SMD Enterprise Architecture example, Figure 19 is the Core Diagram for the NASA SMD Enterprise Architecture.

![Figure 19: The NASA SMD Enterprise Core Diagram](image)

5.6.5 Develop and Answer Business Questions

The preceding five activities: define the purpose of the enterprise, determine the bounds of the enterprise, define the operating model(s) for the enterprise, establish information
technology principles for the enterprise, and develop the core diagram for the enterprise are critical to establishing the conceptual integrity for the enterprise and therefore the enterprise architecture. Together, these five activities bound the enterprise problem space. Once the enterprise problem space is established, it now becomes feasible to begin identifying the business questions the enterprise architecture must be able to answer.

The enterprise architecture, like a successful enterprise itself, is never finished in terms of its efforts. Therefore, an enterprise architecture effort must produce value prior to reaching some never-to-be-achieved state of completion. Any enterprise architecture development approach that is predicated on the premise that the enterprise architecture must be “completed” before any substantial value can be gained from the effort is doomed from the start. Furthermore, any enterprise architecture that has any hope of succeeding in the long run must succeed in some significant way in the short run before the users of the architecture become dissatisfied with the results (or absence of results) and lose interest. Therefore, it is critical that enterprise architecting efforts begin with a purpose and focus on delivering functionality by answering business questions from the very beginning.

Once the prerequisites described in Section 5.6.4.1.1 through 5.6.4.1.5 above have been accomplished and with the core set of business questions identified in Section 5.3 above and with the MAPPMR Enterprise Architecture Metamodel described in Section 5.5 above, it is now possible to begin developing an enterprise architecture model that will
answer the desired business questions. This part of the enterprise architecture development methodology is meant to establish the foundation upon which all the desired functionality of the enterprise architecture is based.

In the approach described in [Martin06], business questions define the purpose, scope and functionality of an enterprise architecture and are the de facto requirements for the enterprise architecture. Business questions are used to identify the entities (nouns), relationships (verbs) and processes (functionality) of the enterprise that need to be accommodated by the enterprise architecture model and will enable the enterprise architecture to function as desired (i.e., answer the business questions). The range of the business question domain is vast. In the beginning, the architecture effort should focus on enterprise products and production lines and attempt to develop the enterprise architecture model that will answer business questions identified in Section 5.3 above. The scope can then be expanded to address the following types of questions:

- What assets does the enterprise possess?
- Where is an asset?
- What function(s) does an asset perform?
- What does an asset connect to?
- What does an asset cost (acquisition, sustainment & replacement)?
- When does an asset need to be replaced (end of life)?
From an information-centric view of information management, this core of the architecture development methodology is produced as follows:

Define what problem the enterprise architecture is attempting to address focusing on some subset of the enterprise dilemma. This is best achieved by determining what knowledge is required, and this is best expressed as a set of business questions that need to be answered such that if the appropriate decision-maker knew the answers to this set of business questions then the decision-maker would have the requisite knowledge to make the necessary decisions. When this is completed next determine:

- What information is needed to answer the business questions identified above. This is achieved by determining the entities and relationships that need to be modeled that will produce the needed information. When this is completed next determine:

- What data are needed to derive the information needed to answer the business questions identified above. Of course, the data are based on the attributes associated with the entities and relationships that were identified in the preceding step. When this is completed next determine:

- The data sources that provide the data needed to derive the information needed to answer the business questions identified above. When this is completed the enterprise architecture model can be populated with data from the identified data sources and the enterprise architecture model can
be verified and validated to ensure it generates the answers to the business questions and provide the requisite **information** to the decision maker.

For the NASA SMD Enterprise Architecture example, the goal is to answer complex business questions relevant to the sound management of resources and assets in support of NASA SMD’s mission into the foreseeable future. In striving towards answering more complex business questions, the initial version of the NASA SMD Enterprise Architecture has the capability to answer the following simplex business questions:

- What is the NASA SMD strategic objective?
- What are the NASA SMD big questions?
- What are the NASA SMD focus areas?
- What are the NASA SMD science domains?
- What are the NASA SMD information needs?
- What are the NASA SMD science requirements?
- What are the NASA SMD programs?
- What are the NASA SMD missions?
- What are the NASA SMD mission locations and orbits?
- What are the NASA SMD studies?
- What are the NASA SMD Enterprise Architecture principles?
- What are the NASA SMD funding levels?
• What are the NASA SMD science products?

The NASA SMD Enterprise Architecture has the capability to answer the following 2nd Order business questions:

• What is the funding profile for a given mission?

• What is the funding profile for a given program?

• What is the funding profile for a given study?

• What is the total funding for a given focus area?

• What focus area does a given mission support?

• What Big Questions does a given mission support?

• What is the location for a given mission?

• What is the intended location for a given program?

• What missions are on or orbiting Mars?

The NASA SMD Enterprise Architecture has the capability to answer the following 3rd or higher Order business questions:

• What is the funding profile for a mission addressing a specific Big Question?

• What is the funding profile for all missions addressing a specific Big Question?
• What is the funding profile for all the Astrophysics missions/programs/studies?

• What missions support Big Question AQ3 and when do those missions reach end of life?

• Will there be a gap between the Hubble Space Telescope end of life and the James Webb Space Telescope initial operational capability?

To assist the architecture effort this research relies on the Maintain Accountability, Produce Product, Manage Resources (MAPPMR) Enterprise Architecture Metamodel Specification (MMS). The MMS captures critical information necessary to establishing and maintaining the conceptual integrity of the enterprise architecture effort. An example MMS for the NASA SMD Enterprise Architecture discussed here is included in the Appendix. This example shows an MMS at the version 1.0 level of maturity and includes all the information need for the Purpose Identification activities identified in Section 5.6.2.

The intended audience of the MMS is the architecting team and it is used to manage the evolution of the enterprise architecture metamodel. However, the MMS contains information that is appropriate for non-technical audiences as well. The MMS contains the following information:

• The purpose of the enterprise, to include:
  
  o The enterprise’s statement of purpose,
• A description of the bounds of the enterprise,

• A description of the desired operating model(s) to include an explanation of the desired levels of process integration and process standardization,

• The enterprise’s information technology principles,

• The core diagram(s) for the enterprise.

• The purpose of the enterprise architecture, to include:

  • The enterprise architecture statement of purpose,

  • The enterprise architecture business questions,

  • The enterprise architecture business questions implemented in the current increment of the enterprise architecture,

  • The enterprise architecture business questions implemented in previous increments of the enterprise architecture

• The enterprise architecture objects and relationships, to include:

  • The enterprise architecture metamodel schema,

  • The enterprise architecture metamodel taxonomy.

• The enterprise architecture methods.

• The enterprise architecture criteria (queries).

• The enterprise architecture views.
5.6.6 Iteration

Note that the last activity that Martin identifies in his Step 6 PFQB architecture process is *Iteration*. In the grand sense, it is true that the objective is to deliver a usable enterprise architecture tool to the users and then begin another iteration of the Six-Step PFQB architecture process. However, it is important to realize that iteration could and should be applied anywhere within the Six-Step process as required to aid the maturation of the enterprise architecture. The fundamental concept is that for any momentous undertaking to succeed, it must be accomplished with many small steps that in the aggregate result in big changes. These small steps allow for small, survivable failures that do not jeopardize the overall process. Of course, each of these small failures triggers an iteration to overcome the setback.

During the course of developing the current version of the enterprise architecture, ideas for the desired functionality in the succeeding version should be solidifying. As mentioned previously, the NASA SMD Enterprise Architecture MMS example is at the version 1.0 level of maturity. Thoughts on what should be included in future versions of the NASA SMD Enterprise Architecture include:

- A sensor to mission mapping. The conceptual schema includes sensor, but the data was not available to integrate this fully.

- A more rigorous representation of the phenomenologies that the sensors can observe. The architecture should be able to show what sensor gaps and overlaps exist in the observable spectrum.
• Reassess the principles and map those principles to technologies modeled in the architecture.

• Production lines need to be identified and represented with enough fidelity to manage their evolution and lifecycles.

• Map the science Requirements to big questions, missions, programs, etc.

• Include a complete list of science products.

• Higher fidelity funding profiles. The currently available funding profile aggregates expenditures that should be visible within the enterprise architecture.

5.6.7 Conclusion

While this focusing methodology for enterprise architecture development approach seems simple enough, it is complicated and confounded in practice by the challenges of wicked problems, complexity and the enterprise learning curve discussed in preceding sections. Conquering these challenges is precisely the goal of an enterprise architecture endeavor so that in the end, the decision makers can get accurate answers to the key business questions necessary for informed decision-making.

The empirical research on enterprise architecture has observed that successful enterprises use their enterprise architecture to focus on the big picture while the unsuccessful enterprises overlook the big picture and get mired in the details. Enterprise architecture
efforts should start simple, but start with a purpose and a focus on delivering functionality (answering business questions) from the very beginning. If the first iteration does not answer some set of business questions, then the effort is proceeding down the wrong path. Since the scope of the enterprise architecture can grow without bound, from the beginning it is important for the scope to remain small and focused, but the ultimate goals of the architecture (in the beginning, think of the end) cannot be ignored.

In order to be of value in the short term, enterprise architecture must focus on the problem space and not the solution space. Enterprise architecture should map the enterprise problem space to identify gaps and overlaps and not strive to be the solution architecture for the entire enterprise. Enterprise architecture should not usurp the functions of systems engineering. It should provide major support to these in terms of a focus on problem finding so that it identifies problems that can then be solved by systems architecture and systems engineering.

A conundrum facing the enterprise is to decide the course of action that is most appropriate for dealing with multiple wicked problems and intractable complexity. Therefore, a successful enterprise architecture must be maintained at least as quickly as the enterprise itself evolves. For this reason, a holistic enterprise solution is too expensive, too resource intensive, too time consuming, and too momentous to be achieved at a pace that equals the evolution of the enterprise. Furthermore, architecting a
point solution—even if it were feasible—ignores the fact that the enterprise wants and needs to thrive in a dynamic environment.

The MAPPMR Enterprise Architecture Metamodel and its accompanying focusing methodology is no guarantee that an enterprise architecture effort will succeed, but this enterprise architecture modeling approach is a starting point that recognizes enterprise architecture as a journey without end.
5.7 Addressing the Enterprise Dilemma

Sed fugit interea fugit irreparabile tempus,
singula dum capi circumvectamur amore.\(^{69}\)

\[ \sim \text{Virgil (70 BC-19 BC)} \]
from the poem \textit{Georgics}
\[ \text{29 BC} \]

This research is based on the assertion that the focus and scope of any enterprise architecture effort should be on addressing the enterprise dilemma. This entails making decisions on how to allocate the enterprise’s resources between sustainment and change. This is not a solitary decision, but a myriad of decisions—some of which are persistent and some of which will need to be reassessed frequently—that are needed to transform the enterprise into its desired state.

At Google’s 2011 Zeitgeist Americas Conference, Eric Schmidt, Google’s Executive Chairman and former CEO, and Larry Page, Google’s current CEO, were taking questions from the audience. The final question asked was “What is the biggest threat to Google’s continued success?

Larry Page quickly responded “Google” implying that he thought Google was its own biggest threat to continued success.

\(^{69}\) Literally, “But meanwhile it flees: time flees irretrievably, while we wander around, prisoners of our love of detail.”
Eric Schmidt added, “Large companies are their own worst enemy because internally they know what they should do, but they don’t do it.

Then Larry Page explained:

One of the interesting things that we’ve noticed is that companies correlate on decision-making and speed of decision-making. There are basically no companies that have good slow decisions. There are only companies that have good fast decisions. I think that’s also a natural thing as companies get bigger—they tend to slow down decision-making. And that’s pretty tragic.

If enterprise architecture is to be worthwhile, it must continually produce value in the form of facilitating “good fast decisions” noted by Schmidt. Static or “snapshot” architectures that focus on the “as-is” and “to-be” all too soon depict the “never-was” and “never-will-be” and are relegated to becoming dusty shelfware. Enterprise architecture efforts are momentous and costly and cannot be justified if the enterprise architecture output is to be of transitory value. To be of lasting value, an enterprise architecture must remain relevant and be insightful and the best strategy for making the enterprise architecture be relevant and insightful is to construct it so that it answers relevant and insightful business questions that help the enterprise make those “good fast decisions” on the allocation of the enterprise’s resources.

This research has focused on developing an enterprise architecture approach that emphasizes establishing the requisite conceptual integrity first, and then applied Occam’s Razor towards emphasizing the development of business questions that address the allocation of resources applied to the enterprise’s core capabilities. The intended result is to describe an enterprise architecture development methodology that yields continuing
value from its earliest beginnings as well as increasing value to the enterprise as the enterprise architecture matures over time.

It would seem to be ideal if there existed a complete set of business questions that every enterprise architecture could answer. However, a complete set of business questions would be no more helpful than Zachman’s framework when it comes to developing a useful enterprise architecture that will have a positive ROI. Enterprise architecture, like most things in the enterprise should deliver the most benefit for the least cost. As noted in Section 5.4.13, Ross’ discovery of a complete lack of correlation between architectural frameworks and architecture maturity highlights that employing a framework may be a detriment. Frameworks tend to include anything and everything that might be useful. As a result, there is no clear insight into what returns value and what does not. Neither is it clear that there can exist a master-set of business questions because an exhaustive set of business questions would be too costly to implement. Value to the enterprise comes from developing an enterprise architecture that is based on its own foundation of conceptual integrity and produces a shared understanding amongst the enterprise’s empowered stakeholders, and that these empowered stakeholders use the enterprise architecture to govern and evolve the enterprise.

In essences, the approach prescribed by this research is:

- Establish the conceptual integrity of the enterprise,
- Identify a manageable set of enterprise business questions and develop the enterprise architecture to answer those business questions,
Repeat.

This enterprise architecture approach has three distinct advantages:

1. The process focuses on establishing conceptual integrity at the beginning. Then it reassesses the conceptual integrity at the beginning of each iteration to ensure that the enterprise and its enterprise architecture stay aligned.

2. The process is repeated in short increments which has two major benefits:
   a. The process strives to produce something of value quickly and takes the user feedback as input for changes or additions in the next iteration.
   b. The process avoids the “big failure at the end” syndrome. Short delivery cycles prevent the situation where the architecture development team works for months/years at a time and delivers a product that fails to meet the needs of the stakeholders. This approach is not “failure-proof.” Quite the opposite, it fosters small failures from which the stakeholders learn and from which future enhancements are identified. The most difficult lesson to convey to fledgling enterprise architecture efforts is that it is impossible to “get it right the first time.” Organizations that espouse “failure is not an option” must soon come to grips with the paradox that failure is an unavoidable prerequisite to success. Enterprise architecture is never a “do it once and move on” proposition. The blame for this is predicated on the wicked problems, complexity, and the enterprise learning curves described above. Other contributors may be inherent in
the human fabric that weaves the shared understanding that the architecture teams develop in their attempt to develop the architecture of the enterprise. Tom Wujec brilliantly explained the perils faced by all design efforts in “The Marshmallow Challenge” problem presented at TED2010 in February 2010. This simple, but revealing exercise shows that the best design process is an iterative design process that learns from “little failures” yet minimizes the risk of a “big failure” at the end when the effort has run out of time and resources. Some architectural approaches recognize this paradox by adopting a process that employs repetitive periods when the architecture team is working en charrette

Repetitive charrettes is a strategy that recognizes that single-stroke attempts at developing architecture will not succeed. It is better to fail early and fail often in the journey, as Tom Wujec so aptly revealed, than to save a big failure for the end after a considerable investment of time and resources.

3. The process applies Occam’s Razor and models only what is needed to model to achieve the desired functionality of the enterprise architecture. Unlike most architecture frameworks, there is no underlying metamodel that will only work when all the data is provided.

---

70 This seven-minute video is available at http://www.ted.com/talks/tom_wujec_build_a_tower.html.

71 Literally “on the cart.” See http://en.wikipedia.org/wiki/Charrette for the derivation of this expression.
While this enterprise architecture approach stresses the importance of and identifies what is needed to establish the conceptual integrity, identifies the enterprise dilemma, identifies the enterprise core capabilities, provides a recommend starting point for enterprise architecture development, and identifies a core set of enterprise business questions, it must be left for each enterprise to determine the exact set of business questions that are needed to derive the requisite value from its own enterprise architecture effort and this must manifest itself as a shared understanding amongst the empowered stakeholders.

The purpose of this core set of enterprise business questions is to serve as a point of embarkation for the enterprise architecting effort. It is not intended to be an exhaustive set of business questions that is “just add data” to be ready for implementation, nor is it meant to be the “correct” set of business questions for any or all enterprises. Instead, this core set of enterprise business questions is intended to point the fledgling enterprise architecture effort at the products and production lines of the enterprise as the logical starting point to initiate the enterprise architecture development effort and hopefully to help focus the investigation on determining its own “right” set of enterprise business questions.

As the enterprise architecture development matures, the enterprise architecture should grow in scope so that it answers business questions that span the full spectrum of the enterprise’s core capabilities with the ultimate goal of helping the enterprise address the enterprise dilemma as applied to achieving the enterprise’s desired levels of business
process integration and business process standardization. Clearly, this cannot be achieved by simply developing an enterprise architecture that can only answer business questions about the produce product core capability. If the enterprise architecture is to be a resource management tool, then it needs to expand to cover the two other core capabilities of maintain accountability and manage resources as well.

At a mature state, the enterprise architecture should:

- Model every asset within the enterprise and identify which one (or more) of the three core capabilities each asset supports.
- Identify when an asset was acquired, and when that asset is expected to be retired.
- Identify what an asset cost to acquire, sustain, and maintain, and identify if these costs are at, above or below expectations.
- Identify if an asset is functioning at, above or below expectations.
- Depict the current and desired operating model(s) for the enterprise to include the current and desired degree of process integration and process standardization within the enterprise.
- Identify the gaps (shortfalls) and overlaps (surpluses) within the enterprise. Note that a surplus is not necessarily undesirable. Surpluses may be required either for fluctuations in seasonal demand, continuity of operations, or contingencies.
When the enterprise architecture achieves this level of maturity, it will have become a robust tool for monitoring and managing the enterprise’s resources. Knowing what fiscal, physical and functional gaps exist within the enterprise is central to addressing the enterprise dilemma. In the context of the NASA SMD enterprise architecture example, a sufficiently mature enterprise architecture would identify all the NASA SMD products, production lines, production line components and facilities and identify each assets’ expected end of life. All of NASA SMD products would be mapped to the NASA SMD big questions, focus areas, science domains, science requirements and information needs. The products and information needs would be mapped to NASA SMD programs, missions and studies. The objective would be to know the current and future status of NASA SMD in terms of how well it achieves its strategic objective in terms of producing products that satisfy the science requirements needed to answer the big questions both in the present and in the foreseeable future. Of equal importance is determining what does not achieve the NASA SMD strategic objective in terms of producing products that satisfy the science requirements needed to answer the big questions both in the present and in the foreseeable future. Identifying these performance gaps are key to identifying changes needed within the enterprise.

While the range of functionality for an enterprise architecture could be boundless, a mature NASA SMD enterprise architecture should be able to answer business questions concerning:
• A physical accounting for each NASA SMD asset, to include each asset's location, current operating condition, expected lifespan, and estimated replacement cost.

• A functional accounting of the NASA SMD enterprise to include: how well each asset performs, its capacity or throughput, its return on investment, how well the asset meets expectations, and how satisfied the consumer is with the product.

• A fiscal accounting of the NASA SMD enterprise to include: the commitment, allocation and obligation of all funds across the budget to the degree that value can be determined for each asset.
6. EVALUATION OF THE RESEARCH

We cannot discover what ought to be the case by examining what is the case. We must decide what ought to be the case.

~ Paul W. Taylor
Normative Discourse, 1961

6.1 Case Study Design

George Friedman & Andrew P. Sage developed a two-dimensional case study framework for systems engineering case study research. The Friedman-Sage case study framework was developed from the perspective of systems acquisition from a development contractor-supplier by a governmental organization [Friedman04]. The Friedman-Sage case study framework is represented by a nine-by-three matrix consisting of nine concept domains represented by the rows, and three responsibility domains represented by the columns of the matrix. The purpose of this case study framework is “to provide the general structure of a framework for systems engineering concepts and their illustration through related case studies” [Freidman04]. The Friedman-Sage Two-dimensional Framework for Systems Engineering Case Study Research is shown in Figure 20. The nine concept domains consist of six phases of the system engineering lifecycle (A
through F below) and three process and system management activities (G through I below):

A. Requirements Definition and Management

B. Systems Architecting and Conceptual Design

C. Detailed System and Subsystem Design and Implementation

D. Systems and Interface Integration

E. Validation and Verification

F. System Deployment and Post Deployment

G. Life Cycle Support

H. Risk Management

I. System and Program Management
James N. Martin adapted and extended the Friedman-Sage case study framework “…to address case studies for enterprise-level systems engineering” in [Martin06, p.182] as shown in Table 1. In extending the Freidman-Sage case study framework, [Martin06] noted that the acquisition-oriented Friedman-Sage case study framework is designed to handle traditional systems engineering within a program or project.” The Martin case study framework modifies the Friedman-Sage case study framework to expand the scope to include strategic- and investment-oriented enterprise-systems-engineering activities.
<table>
<thead>
<tr>
<th>Concept Domain</th>
<th>Responsibility Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Strategic Technical Planning</td>
<td></td>
</tr>
<tr>
<td>B. Capability-Based Planning Analysis</td>
<td>*</td>
</tr>
<tr>
<td>C. Technology and Standards Planning</td>
<td></td>
</tr>
<tr>
<td>D. Enterprise Requirements Definition and Management</td>
<td></td>
</tr>
<tr>
<td>E. Enterprise Architecture and Conceptual Design</td>
<td>*</td>
</tr>
<tr>
<td>F. Program and Project Detailed Design and Implementation</td>
<td></td>
</tr>
<tr>
<td>G. Program Integration and Interfaces</td>
<td>*</td>
</tr>
<tr>
<td>H. Program Validation and Verification</td>
<td></td>
</tr>
<tr>
<td>I. Deployment and Post Deployment</td>
<td></td>
</tr>
<tr>
<td>J. Program Life Cycle Support</td>
<td></td>
</tr>
<tr>
<td>K. Risk and Opportunity Management</td>
<td></td>
</tr>
<tr>
<td>L. Enterprise Evaluation and Assessment</td>
<td>*</td>
</tr>
</tbody>
</table>

concept domains to the three domains of principle activity consisting of Program Responsibility, Shared Responsibility and Enterprise Responsibility.

Martin labeled the Friedman-Sage case study framework the Traditional Systems Engineering (TSE) case study framework to indicate its scope and differentiate it from his extension, which he called the Enterprise Systems Engineering (ESE) case study framework [Martin06]. Table 2 aligns the Friedman-Sage concept domains with the Martin concept domains. Note that domains A through H of the Friedman-Sage case study framework relate to domains D through K of the Martin case study framework although the scope of the concept domains in [Martin06] has been extended beyond what was originally intended in [Friedman04]. Also, System and Program Management in the Friedman-Sage case study framework is replaced in the Martin case study framework with Enterprise Management and includes four new concept domains A through C, and L which are Strategic Technical Planning, Capability-Based Planning Analysis, Technology and Standards Planning, and Enterprise Evaluation and Assessment.
Table 2: Aligning the Martin Case Study Framework to the Freidman-Sage Case Study Framework

<table>
<thead>
<tr>
<th>Friedman-Sage Concept Domains</th>
<th>Martin Concept Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Definition and Management</td>
<td>D  Enterprise Requirements Definition and Management</td>
</tr>
<tr>
<td>C System and Subsystem Detailed Design and Implementation</td>
<td>F  Program and Project Detailed Design and Implementation</td>
</tr>
<tr>
<td>D Systems Integration and Interfaces (^{72})</td>
<td>G  Program Integration and Interfaces</td>
</tr>
<tr>
<td>E Validation and Verification</td>
<td>H  Program Validation and Verification</td>
</tr>
<tr>
<td>F Deployment and Post Deployment</td>
<td>I  Deployment and Post Deployment</td>
</tr>
<tr>
<td>G Life Cycle Support</td>
<td>J  Program Life Cycle Support</td>
</tr>
<tr>
<td>H Risk Assessment and Management</td>
<td>K  Risk and Opportunity Management</td>
</tr>
<tr>
<td>I System and Program Management</td>
<td>[replaced with Enterprise Management to include the 4 areas below]</td>
</tr>
<tr>
<td>A Strategic Technical Planning</td>
<td></td>
</tr>
<tr>
<td>B Capability-Based Planning Analysis</td>
<td></td>
</tr>
<tr>
<td>C Technology and Standards Planning</td>
<td></td>
</tr>
<tr>
<td>L Enterprise Evaluation and Assessment</td>
<td></td>
</tr>
</tbody>
</table>

6.2 Enterprise Architecture Case Study Framework

Similar to the Martin expansion of the original Friedman-Sage case study framework, a focus of this research has been to expand the Friedman-Sage and Martin case study frameworks by identifying enterprise architecture concept domains. Table 3 shows the eight enterprise architecture concept domains identified by this research and maps those

\(^{72}\) [Friedman04] labels this concept domain as “Systems and Interface Integration” in its Figure 2 (shown above as Figure 19) and as “Systems Integration and Interfaces” in the text in its Section 4.4.
concept domains to both the Friedman-Sage and the Martin case study frameworks concept domains.

This research expands the existing work of Friedman-Sage and Martin and compares and contrasts the eight enterprise architecture concept domains identified in Table 3 to three domains of principle activity consisting of Empowered Stakeholder Responsibility, Surrogate Stakeholder Responsibility, and No Stakeholder Responsibility as shown in Table 4. The purpose of this is to determine the degree of involvement of stakeholders within the enterprise. It is important to not only determine if the architectural prerequisites are being accomplished, but also to determine if these prerequisites are being accomplished by the appropriate stakeholders within the enterprise.
Table 3: Aligning the Enterprise Architecture Case Study Framework to the Martin Case Study Framework and the Freidman-Sage Case Study Framework

<table>
<thead>
<tr>
<th>Friedman-Sage Concept Domains</th>
<th>Martin Concept Domains</th>
<th>Enterprise Architecture Concept Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Requirements Definition and Management</td>
<td>D Enterprise Requirements Definition and Management</td>
<td>EA Enterprise Architecture</td>
</tr>
<tr>
<td>B System Architecture and Conceptual Design</td>
<td>E Enterprise Architecture and Conceptual Design</td>
<td>EA1 Enterprise Purpose Definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA2 Enterprise Boundary Definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA3 Enterprise Operating Model(s) Definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA4 Enterprise Information Technology Principles Definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA5 Enterprise Core Diagram Development</td>
</tr>
<tr>
<td>C System and Subsystem Detailed Design and Implementation</td>
<td>F Program and Project Detailed Design and Implementation</td>
<td>EA3.1 Enterprise Business Process Integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA3.2 Enterprise Business Process Standardization</td>
</tr>
<tr>
<td>D Systems Integration and Interfaces</td>
<td>G Program Integration and Interfaces</td>
<td></td>
</tr>
<tr>
<td>E Validation and Verification</td>
<td>H Program Validation and Verification</td>
<td></td>
</tr>
<tr>
<td>F Deployment and Post Deployment</td>
<td>I Deployment and Post Deployment</td>
<td></td>
</tr>
<tr>
<td>G Life Cycle Support</td>
<td>J Program Life Cycle Support</td>
<td></td>
</tr>
<tr>
<td>H Risk Assessment and Management</td>
<td>K Risk and Opportunity Management</td>
<td></td>
</tr>
<tr>
<td>I System and Program Management</td>
<td>[replaced with Enterprise Management to include the 4 areas below]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Strategic Technical Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B Capability-Based Planning Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C Technology and Standards Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L Enterprise Evaluation and Assessment</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EA6 Enterprise Business Question Development</td>
</tr>
</tbody>
</table>
Table 4: Enterprise Architecture Case Study Framework

<table>
<thead>
<tr>
<th>Concept Domains</th>
<th>Responsibility Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Architecture Concept Domains</td>
<td>Empowered Stakeholder Responsibility Domain</td>
</tr>
<tr>
<td>EA</td>
<td>Enterprise Architecture</td>
</tr>
<tr>
<td>EA1</td>
<td>Enterprise Purpose Definition</td>
</tr>
<tr>
<td>EA2</td>
<td>Enterprise Boundary Definition</td>
</tr>
<tr>
<td>EA3</td>
<td>Enterprise Operating Model(s) Definition</td>
</tr>
<tr>
<td>EA3.1</td>
<td>Enterprise Business Process Integration Determination</td>
</tr>
<tr>
<td>EA3.2</td>
<td>Enterprise Business Process Standardization Determination</td>
</tr>
<tr>
<td>EA4</td>
<td>Enterprise Information Technology Principles Definition</td>
</tr>
<tr>
<td>EA5</td>
<td>Enterprise Core Diagram Development</td>
</tr>
<tr>
<td>EA6</td>
<td>Enterprise Business Question Development</td>
</tr>
</tbody>
</table>

For an enterprise architecture concept domain to be associated with the Empowered Stakeholder Responsibility Domain, its activities must have been accomplished by a stakeholder or group of stakeholders that is empowered with the responsibility and authority to execute the actions associated with the enterprise architecture concept domain. If this occurs, then these empowered stakeholders must have been actively involved in accomplishing the end result of the enterprise architecture concept domain and have established a shared understanding of the outcome and how the outcome came to be.
For an enterprise architecture concept domain to be associated with the Surrogate Stakeholder Responsibility Domain, its activities must have been accomplished by a stakeholder or group of stakeholders that is not empowered with the responsibility and authority to execute the actions associated with the activity. For an outcome of an enterprise architecture concept domain to become final, it must be approved by some higher-level empowered stakeholder(s) that is acting on the recommendation of the surrogate stakeholder(s). If this occurs, then no empowered stakeholders were actively involved in accomplishing the end result of the enterprise architecture concept domain and no empowered stakeholders achieved a shared understanding of how the outcome came to be.

For an enterprise architecture concept domain to be associated with the No Stakeholder Responsibility Domain its activities must have been neither identified nor assigned to any stakeholder(s) to be accomplished. It is entirely possible that some stakeholder or group of stakeholders may assume the burden of accomplishing the activities associated with the enterprise architecture concept domain without having been explicitly tasked to do so. If this occurs, then no empowered stakeholders were actively involved in accomplishing the end result of this enterprise architecture concept domain and no empowered stakeholders achieved a shared understanding of how the outcome came to be.

Since developing enterprise architecture is a highly iterative undertaking, it is possible to start an enterprise architecture concept domain activity in one responsibility domain and
in the end discover that the activity has resided in different responsibility domains during different iterations.

This research has identified the following eight enterprise architecture concept domains:

- Enterprise Purpose Definition,
- Enterprise Boundary Definition,
- Enterprise Operating Model(s) Definition:
  - Enterprise Business Process Integration Determination,
  - Enterprise Business Process Standardization Determination,
- Enterprise Information Technology Principles Definition,
- Enterprise Core Diagram Development,
- Enterprise Business Question Development.

Five of the eight enterprise architecture concept domains are derived from the benchmarks identified in Sections 5.6.4.1.1 through 5.6.4.1.5 above. Enterprise Business Process Integration Determination and Enterprise Business Process Standardization Determination can be viewed as sub-domains of the Enterprise Operating Model(s) Definition and directly contribute to determining the appropriate-to-the-enterprise operating model(s). These seven enterprise architecture concept domains comprise the pre-requisites of the eighth enterprise architecture domain, Enterprise Business Question Development.
Development. Together, these seven enterprise architecture concept domains help establish the conceptual integrity for the enterprise and thus for the enterprise architecture. Efforts that omit these first seven enterprise architecture concept domains will likely have difficulty achieving the conceptual integrity needed to achieve a successful enterprise architecture.

6.3 Case Study Backgrounds and Descriptions

This research includes four case studies. First, the background and description of each of the four case studies is presented in Sections 6.3.1 through 6.3.4. Then, in Section 6.4, the findings for each case study concept domain are described in the context of each of the three responsibility domains.

6.3.1 Case Study One Background and Description

In June, 2001, Secretary of Defense Donald Rumsfeld issued a memo stating that one of his highest priorities was reliable, accurate and timely financial management information as the basis for making effective business decisions for DoD. In response, the Office of the Secretary of Defense (OSD) initiated a business transformation effort with the goal of transforming DoD’s business and financial management processes. As part of this effort, OSD established the Business Management Modernization Program (BMMP) under the oversight of six OSD Business Domains: Accounting & Finance, Acquisition, Human Resource Management, Installations & Environment, Logistics, and Strategic Planning &
Budgeting\textsuperscript{73} as well as the Domain Owners Integration Team (DO/IT) to manage DoD’s transformation. The first order of business for BMMP was to create the DoD Business Enterprise Architecture (BEA).

Version 1.0 of the BEA was published in March of 2003. BEA v1.0 did not have an enterprise perspective and pursued an approach that presented Domain-centric stovepipes. Recognizing this shortcoming, the Business Modernization and Systems Integration (BMSI) office and the Business Domains developed a set of process models called “value chains” that still adhered to the Domain stovepiped approach and contained much cross-process and cross-Domain redundancy. In February 2004, BMSI hosted the first of what was scheduled as a weekly “sub-processes and schedule” meeting to determine how to move forward with the value chain approach.

The Installations & Environment Domain had expended significant effort developing its value chain to reflect its ongoing business process re-engineering effort. However, no other Domain had spent any significant effort developing its own value chain and one Domain was unaware that BMSI had developed a value chain for that Domain.

However, the Domain Action Officers realized during that meeting that the value chains, as presented, continued to propagate a stovepiped set of business processes that possessed more similarities than dissimilarities. The Domain Action Officers reached a

\textsuperscript{73} The number of domains grew to seven in 2003 when Technical Infrastructure was added. However, Technical Infrastructure had no unique functional requirements that did not originate in another business domain and did not contribute in the architecture development process during the time of this case study. In 2004 the Accounting and Finance Domain and the Strategic Planning & Budgeting Domain merged to form the Financial Management Domain.
fundamental agreement that there did not need to be multiple business processes for maintaining accountability of DoD assets when one process with different sets of business rules could work for all classes of DoD assets.

In March 2004, the Domain Action Officers initiated a two-month series of workshops that integrated a set of processes into a DoD-wide Enterprise Business Process Model (EBPM) that included enterprise sub-processes for:

- Apportionment,
- Receipt & Acceptance,
- Commitments & Obligations,
- Receivables, Payables & Disbursements,
- Asset Accountability & Valuation.

Following these workshops, the Domain Action Officers continued to consolidate and refine the EBPM. Figure 21 shows the top-most level of abstraction for the EBPM. The EBPM tasks were then mapped to two top-level BEA capabilities of Manage Resources and Maintain Accountability. Tasks that fell within the Manage Resources capability were mapped to the lifecycle phases:

- Plan,
- Source or Acquire
- Execute,
- Make
- Deliver
- Operate (end use)
- Return, Separate, Terminate and Dispose.

Tasks that fell within the Maintain Accountability capability were mapped to tasks within the Manage Resources capability to emphasize that accountability must be maintained throughout the entire lifecycle of all assets. Shown within the Maintain Accountability capability are three specializations corresponding to the three distinct rule sets for human resources accountability, asset accountability and financial accountability.

Figure 21: Enterprise Business Process Model Framework
The “EBPM viewpoint” for BEA v2.3 was expressed as follows [Harrell05]:

The BEA EBPM ... represents the end-to-end business enterprise of the Department of Defense from the perspective of the lifecycle of all DoD resources with the focus on resource management and the interactions with the financial record, asset record, and [human resources] record. The purpose of the EBPM is to show how the DoD Business Enterprise Capabilities for Resource Management, Financial Accountability, Asset Accountability and Human Resources Accountability are integrated throughout the life-cycle of all resources.

Figure 22 illustrates abstractly how the Installations and Environment Domain Architecture (IEDA) intended to integrate its tasks in alignment with the EBPM framework. The task boxes with titles beginning with “I&E Domain Capability…” were Installations and Environment Domain unique tasks for which the Installations and Environment Domain had the lead in terms of process ownership. The other task boxes were either specializations of enterprise tasks or tasks for which another business domain had the lead in terms of process ownership.
Figure 22: Installations & Environment Domain Architecture Framework

Figure 23 is a miniature of the EBPM process model. To be legible, the process model was printed on twelve feet of 42-inch wide plotter paper. The top-level structure of the EBPM process model in Figure 23 is the same (albeit difficult to see) as the top-level abstractions shown in Figure 21 and Figure 22 containing the two capabilities of Manage Resources and Maintain Accountability as well as the lifecycle phases of Plan, Source/Acquire, Execute, and Return/Separate/Terminate/Dispose within the Manage Resources capability.
The BEA effort only had responsibility for the two capabilities, Manage Resources and Maintain Accountability. The Perform Mission capability (elsewhere referred to as Produce Product) was deemed outside the scope of the BEA and thus not reflected in the top-most abstraction of the EBPM. While the process to produce the output of an acquisition program is, in fact, fundamentally identical to producing any other product, it was not acceptable to depict anything within the EBPM that could be misconstrued as Perform Mission, as that was outside the scope of the BEA. Thus, the EBPM only represented Manage Resources and Maintain Accountability as these two capabilities fell under the responsibility of the “Business Enterprise” of DoD.

Figure 24 is the top-most functional decomposition of the EBPM and was the basis for the BEA OV-5 Activity Model.
6.3.2 Case Study Two Background and Description

In 2005, the National Geospatial-intelligence Agency (NGA) completed version two of the National System for Geospatial Intelligence (NSG) Enterprise Architecture. The NSG Enterprise Architecture was developed over a span of many months in accordance with DoDAF v1.0 using what was known at the time as Popkin System Architect. The DoDAF views produced for the enterprise architecture included: AV-1, AV-2, OV-1, OV-2, OV-3, OV-4, OV-5, OV-6c (six), the as-is SV-1 & SV-4, the to-be SV-1, SV-4
(seven), SV-5, SV-6, and the TV-1 and TV-2. In addition, there was a non-DoDAF Enterprise Conceptual Data Model (ECDM).

In 2006 NGA updated the NSG Enterprise Architecture to version 2.5 by issuing a new OV-5 Operational Activity Hierarchy Chart. In February 2006, the plan called for version 3.0 of the NSG Enterprise Architecture to be produced by October 2006 and include the following revised DoDAF views: AV-1, AV-2, OV-1, OV-2, OV-3, OV-4, OV-5, OV-6c, OV-7, SV-2, SV-5, TV-1, TV-2, as well as an updated ECDM. However, none of these revised architecture products were produced, and have not been produced thus far to the author’s knowledge.

A 2007 review of the NGA enterprise architecture efforts revealed that NGA had had multiple enterprise architecting efforts conducted by multiple organizations within NGA in the previous three years to include:

- NSG Enterprise Architecture v2.0 and the NSG Enterprise Architecture v2.5 OV-5 Operational Activity Hierarchy Chart described above.

- As-Is Systems View. This effort was new work in 2007 that had identified over 600 systems and was developing only the SV-1s and SV-2s for those systems. The impetus for this effort was to prepare NGA for consolidating into a new facility at NGA Campus East as required by the 2005 Defense Base Realignment and Closure (BRAC) Commission. The SV-1 and SV-2

---

74 See [DoDAF04] for a description of the DoDAF architecture views.
architecture artifacts from the NSG Enterprise Architecture v2.0 were deemed not suitable for this effort.

- Generalized Information Workflows. This was a graphical depiction of seven as-is NGA product pipelines organized by data origin. These product pipelines consisted of Aeronautical, Nautical/Hydrographic, Topographic/Terrestrial, Precise Positioning & Targeting, Geodesy & Geophysics, Geographic Names, and Imagery Analysis.

- Operation Capabilities. NGA had identified twelve operational capabilities that were decomposed into 40 lower-level operational capabilities. Part of NGA considered this the “official” list of NGA capabilities. Other offices within NGA did not consider this as an official list.

- Business Services Architecture Business Component Map. Five systems were decomposed and mapped to IBM’s Direct—Control—Execute framework, which was modified to Strategic Planning—Management—Execution.

- Integrated Enterprise Business Architecture (iEBA). Began as the NGA Business Architecture, which proposed a top-down enterprise decomposition depicting the enterprise as an entity-relationship diagram. Development of the iEBA was not completed. This may have been the intended approach for the NSG Enterprise Architecture v3.0, which never came to fruition.

- As-is Product Lines. NGA had identified five product lines (aeronautical, geospatial, intelligence, maritime, imagery) plus software tools.
• Business Architecture. This effort identified ten NGA lines of business and mapped those lines of business into the Federal Enterprise Architecture Business Reference Model decomposition.

In addition to these eight architecting efforts, NGA was pursuing the acquisition of a major modernization in information technology infrastructure known at the time as the GeoScout Program. GeoScout was envisioned as a single web-presence portal for all National System for Geospatial-Intelligence (NSG) products and information. The architecture for GeoScout was being developed by the GeoScout program and was only partially visible to the NSG Enterprise Architecture team and apparently unconstrained by the previously published NSG Enterprise Architecture.

In summary, between 2004 and 2007 no fewer than nine organizations within NGA had pursued or were pursuing enterprise-architecture-like efforts, but in the end NGA had no enterprise architecture that was being used for any purpose beyond historical documentation. This led one NGA manager to note that “NGA has lots of governance with limited governance impact.”

### 6.3.3 Case Study Three Background and Description

In 2006, a U.S. government organization referred to here as Case Study Three, or CS3, was pursuing enterprise architecture. CS3 had chosen to divide its labors and address enterprise architecture from multiple directions. CS3 had bifurcated itself organizationally and was pursuing a “mission” enterprise architecture and a “non-
mission” enterprise architecture not unlike DoD’s segregation of its mission
architecture(s) from its Business Enterprise Architecture in case study one above.

On the mission side of the enterprise, CS3 was pursuing a mission-focused enterprise
architecture that conformed to DoDAF v1.0. The scope of this effort was all mission
functionality within the CS3 enterprise. This effort was being led by an architecture
group organizationally lateral to the acquisition groups that were developing their own
systems architectures and acquiring systems that performed the missions. The enterprise
architecture group did not have any fiscal, functional or managerial authority over the
acquisition groups. This effort has continued up to the present and strives to maintain an
as-is depiction of the enterprise, a feasible (constrained) future-state of the enterprise, and
an unconstrained future dream-state of the enterprise.

On the non-mission side of the enterprise, CS3 was pursuing a non-mission-focused
enterprise architecture that was attempting to conform to the FEA Framework (FEAF).
The CIO for CS3 was on the non-mission side of the organization and was leading the
FEAF-based enterprise architecture effort. The scope of this effort was all-things non-
mision within the CS3 enterprise. The CS3 CIO did have some authority over non-
mision information technology acquisition and had security accreditation authority over
all CS3 information technology, both mission and non-mission. Therefore the CIO did
have insights into what information technology was included in the mission portion of the
enterprise. This effort stopped at the point where it attempted to map the leaf nodes of
the Business Reference Model (BRM) to the leaf nodes of the Performance Reference Model (PRM) and was overcome by the major combinatorics of the task.

Also on the non-mission side of the enterprise, an organization not under the authority of the CIO was responsible for compliance with OMB Circular No. A-123 titled Management’s Responsibility for Internal Control. The primary focus of this task was producing and maintaining the CS3 Management Control Plan (MCP). Included in the MCP was the collected list of “key processes” for the CS3 enterprise.

Top-level management did issue revised overarching vision, mission and capabilities descriptions for use across the enterprise.

6.3.4 Case Study Four Background and Description

In 2008, a U.S. government organization referred to here as Case Study Four, or CS4, began an effort to construct an enterprise architecture that was national in scope. CS4 was a DoD organization. However, the enterprise architecture it was attempting to create spanned multiple DoD enterprises (OSD and the Departments of the Army, Navy and Air Force) as well as several non-DoD organizations.

The enterprise architecture effort began at the behest of DoD, but the performers of the effort consisted entirely of contractor personnel. Initially, the scope was intentionally limited to a subset of one mission area. The effort was to depict the as-is and the to-be as prescribed by the known programs of record. It was envisioned that the scope would grow to include other mission areas in future iterations. Work began by attempting to
define the purpose of the architecture. The purpose and the boundaries of the enterprise were as specified in existing documentation, legislation and programs of record. The focus of the effort was on the present and on the known future as specified by the programs of record. The intended use of the architecture was to examine the current and known-future enterprise in search of gaps and overlaps that represented risks or opportunities to the missions of the enterprise. Since the effort was not a priori choosing a new future state, there were no new operational concepts, operating model(s), principles, integration, or standardization choices identified.

This effort did attempt to identify relevant business questions that could be applied to an enterprise architecture of the present and of the known future-state of the enterprise in hopes of identifying the risks and opportunities.

This enterprise architecture effort went through several developmental iterations that generated incremental sets of business questions, conceptual schemas, taxonomies. The enterprise architecture effort attempted to capture all this information in a metamodel specification. However, this effort struggled with finding current and accurate data with which to populate the model. Since no one at CS4 had either fiscal or functional authority over any other organization in the “enterprise,” requests for data went mostly unanswered. The effort did proceed with older (2-5 years) data, but the interest in the effort waned as it became apparent that obtaining current data was unlikely.
6.4 Case Study Evaluation

Each of the case studies examined in this section is an enterprise of the U.S. federal government. While every enterprise is unique and therefore must develop its own enterprise architecture to suit its own needs, there is nothing in the enterprise architecture case study framework or its concept domains and responsibility domains that render this case study framework any less-suited for the study of commercial or private enterprises.

The following Sections 6.4.1 through 6.4.12 contain a description of each enterprise architecture concept domain relevant to all three associated responsibility domains. Also included are evaluations of the four case studies with respect to how each accomplished (or did not accomplish) the activities from the eight enterprise architecture concept domains and the constraints of the relevant responsibility domain(s). As an overarching context, the following applies to all of the enterprise architecture concept domain—responsibility domain pairings described in the following sections:

**Enterprise Architecture (EA) Concept:** The enterprise architecture shall be established and shall prescribe in the enterprise core diagram the desired operating model(s) for the enterprise and specify the degree of enterprise business process integration and the enterprise business process standardization which complies with the set of information technology principles established for the enterprise.
6.4.1 Enterprise Purpose Definition

Enterprise Purpose Definition Concept: Establishing the fundamental reason-to-be underlying the organization's existence. A clear statement of purpose for the enterprise needs to exist and is key to establishing the conceptual integrity for the enterprise and thus for the enterprise architecture.

Enterprise Purpose Definition by Empowered Stakeholders Criteria: Empowered stakeholders issue a statement that clearly delineates the purpose of the enterprise. This statement of purpose forms the pinnacle of the top-down decomposition that expands into the enterprise architecture as purpose implies needs and as needs lead to requirements and as requirements transform into specifications in a coherent and traceable manner.

Enterprise Purpose Definition by Surrogate Stakeholders Criteria: Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for the enterprise statement of purpose to surrogate stakeholders.

Enterprise Purpose Definition by No Stakeholders Criteria: No stakeholder is assigned or assumes the responsibility for expressing the enterprise statement of purpose before the enterprise architecture effort begins. Higher echelon stakeholders believe that the enterprise statement of purpose has already been published, is obvious if not published, or are unaware of the importance of having a clearly articulated enterprise statement of purpose towards the goal of achieving architectural alignment. As a consequence, this activity is not performed.
6.4.2 Enterprise Purpose Definition Case Study Key Findings

**Case Study One Key Findings:** In Case Study One, the enterprise purpose definition fell under the No Stakeholder Responsibility Domain. DoD probably suffers from an abundance of statements of purpose. However, in the case of the BEA, there was no single authoritative statement of purpose for the DoD Business Enterprise written prior to the start of developing the EBPM (BEA version 2.2) that was endorsed by all the Domains. While some stakeholders are certainly more empowered than others, no *bona fide* empowered stakeholders existed within DoD’s BEA effort. DoD, like any other federal bureaucracy, has vast numbers of stakeholders. However, attempting to develop a transformational business enterprise architecture for all of DoD to include the Departments of the Army, Navy and Air Force where DoD does not have any direct control of funding across the enterprise ensures that no empowered stakeholder can ever exist. Within DoD’s BEA, there never existed an empowered stakeholder that had both decision authority and fiscal authority for implementing any aspect of the BEA.

In hindsight, the Domain Action Officers that developed the Enterprise Business Process Model (EBPM) crafted their own statement of purpose only after completing the first development cycle of the EBPM. This EBPM Viewpoint became the *de facto* statement of purpose for the EBPM (but not the entire DoD Business Enterprise) and established the top-most point from which all else within the EBPM was decomposed. The Domain Action Officers had arrived at a shared understanding of the EBPM statement of purpose during the process of developing the EBPM. However, the need for a clear and concise
statement of purpose became apparent when it came time to present the EBPM to other parties that had not actually participated in the EBPM development.

**Case Study Two Key Findings:** In Case Study Two, the enterprise purpose definition fell under the Empowered Stakeholder Responsibility Domain. NGA had a clear statement of purpose that satisfied this need. This statement of purpose was approved at the highest level within NGA and was reasonably clear and concise.

**Case Study Three Key Findings:** In Case Study Three, the enterprise purpose definition fell under the Empowered Stakeholder Responsibility Domain. CS3 had an excellent statement of purpose. The only real issue is that the statement of purpose was revised three times in three years. This becomes a problem if the enterprise architecture is tightly coupled to and decomposed from the statement of purpose then the effort required to realign to a new statement of purpose can be very time consuming.

**Case Study Four Key Findings:** In Case Study Four, the enterprise purpose definition fell under the Empowered Stakeholder Responsibility Domain. CS4 did not have dominion over the enterprise it was trying to architect. However, since the goal of CS4 was to analyze the current and known-future programs of record of the enterprise, the set of existing purposes for the various subsets of the enterprise sufficed.

### 6.4.3 Enterprise Boundary Definition

**Enterprise Boundary Definition Concept:** Defining the bounds of the enterprise is important because there needs to be a clear identification of what is and is not the
enterprise. Defining where the boundaries of the enterprise lie differentiates endogenous from exogenous influences on the enterprise and allows the enterprise to focus on the endogenous influences over which it has control while being aware of the exogenous influences which often imply risk, can be mitigated, but probably cannot be directly controlled.

**Enterprise Boundary Definition by Empowered Stakeholders Concept:** Empowered stakeholders issue a statement that clearly identifies the boundaries of the enterprise. This boundary definition establishes the edge of the enterprise and identifies the limits of control for the enterprise architecture.

**Enterprise Boundary Definition by Surrogate Stakeholders Concept:** Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for determining the enterprise boundaries to surrogate stakeholders.

**Enterprise Boundary Definition by No Stakeholders Concept:** No stakeholder is assigned nor does any assume the responsibility for determining the enterprise boundaries before the enterprise architecture effort begins. Higher echelon stakeholders believe that the enterprise boundaries have already been published, are obvious if not published, or are unaware of the importance of having clearly defined enterprise boundaries.

### 6.4.4 Enterprise Boundary Definition Case Study Key Findings

**Case Study One Key Findings:** In Case Study One, the enterprise boundary definition fell under the No Stakeholder Responsibility Domain. The BEA had a large number of
boundary issues that were neither adequately identified nor resolved. Since the concept of a DoD-wide BEA covered multiple enterprises (OSD, Joint Staff, Departments of the Army, Navy, Air Force, etc.), it was always challenging to identify where the authorities and accountabilities (fiscal, physical, functional) resided. As a consequence, the BEA effort focused on defining a goal state for the architecture, but had no significant leverage to incentivize any evolution towards that goal state.

**Case Study Two Key Findings:** In Case Study Two, the enterprise boundary definition fell under all three Responsibility Domains. External boundaries for the NGA enterprise were well established and under the control of the Empowered Stakeholder Responsibility Domain. However, boundaries within the NGA enterprise were in a state of flux as various groups within NGA were advocating “transforming” the business processes of the enterprise. These transformations all impacted how NGA defined its internal boundaries. Most of this realignment discussion was being conducted at the Surrogate Stakeholder Responsibility Domain as evidenced by the numerous enterprise-architecture-like efforts that were occurring simultaneously at NGA. Often these differences resulted in stalemates within the organization that transitioned to the No Stakeholder Domain when consensus could not be reached.

**Case Study Three Key Findings:** In Case Study Three, the enterprise boundary definition fell under the Empowered Stakeholder Responsibility Domain. There were boundary issues between CS3 and other organizations, but these were generally well known as the various parties negotiated where the new boundaries would lie.
Case Study Four Key Findings: In Case Study Four, the enterprise boundary definition fell under the Empowered Stakeholder Responsibility Domain. Since CS4 was just functioning as the architect and did not have a stake in any of the boundaries, there were no issues with either internal boundaries or the boundaries surrounding the enterprise.

6.4.5 Enterprise Operating Model(s) Definition, Enterprise Business Process Integration and Enterprise Business Process Standardization

Enterprise Operating Model(s) Definition, Enterprise Business Process Integration and Enterprise Business Process Standardization Concept: Once the purpose and boundaries of the enterprise are established, the desired operating model(s) can be defined for the purpose of ascertaining the necessary level of business process integration and business process standardization desired for the enterprise. A prerequisite of this step is determining the information technology principles that the enterprise wants to adopt. Paradoxically, a prerequisite for determining the information technology principles is knowing the desired operating model(s) for the enterprise.

Ascertaining the desired operating model(s) that reflects the desired balance of business process integration and business process standardization and is coherently based on the desired set of information technology principles is no trivial task. The empirical evidence suggests that most enterprises never succeed at this and the ones that do succeed are motivated by dire circumstances that threaten the continued existence of the enterprise. Nevertheless, adopting information technology principles, ascertaining the desired
balance of business process integration and business process standardization and then
deciding on the operating model(s) that is consistent with those principles and reflects the
desired integration and standardization becomes the foundation for a successful enterprise
architecture that cannot be achieved otherwise.

**Enterprise Operating Model(s) Definition, Enterprise Business Process Integration
and Enterprise Business Process Standardization by Empowered Stakeholders**

*Concept:* Empowered stakeholders determine the desired degree of business process integration and business process standardization as well as the information technology principles upon which these are based. Then the empowered stakeholders determine the desired operating model(s) for the enterprise such that the operating model(s) is consistent with the principles and achieves the desired degree of business process integration and business process standardization.

**Enterprise Operating Model(s) Definition, Enterprise Business Process Integration
and Enterprise Business Process Standardization by Surrogate Stakeholders**

*Concept:* Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for determining the business process integration, business process standardization, and the operating model(s) for the enterprise to surrogate stakeholders.

**Enterprise Operating Model(s) Definition, Enterprise Business Process Integration
and Enterprise Business Process Standardization by No Stakeholders**

*Concept:* No stakeholder is assigned or assumes the responsibility for determining the business process integration, business process standardization, and the operating model(s) for the
enterprise. If this is not addressed from the enterprise perspective, then it usually defaults to being determined on a stovepipe-by-stovepipe basis.

6.4.6 Enterprise Operating Model(s) Definition, Enterprise Business Process Integration and Enterprise Business Process Standardization

Case Study Key Findings

Case Study One Key Findings: In Case Study One, the enterprise operating model(s) definition, enterprise business process integration, and enterprise business process standardization activities fell under the No Stakeholder Responsibility Domain because it was not determined \textit{a priori}. It was perceived that business process integration and business process standardization were objectives of the BEA, but the extent and degree that would occur was not reflected before beginning the BEA v2.2 effort. As the EBPM was being developed, the desired balance of business process integration and business process standardization evolved. After the fact, the EBPM itself became the \textit{de facto} enterprise operating model.

Case Study Two Key Findings: In Case Study Two, the enterprise operating model(s) definition, enterprise business process integration, and enterprise business process standardization activities fell under the No Stakeholder Responsibility Domain. The National System for Geospatial Intelligence Enterprise Architecture had no explicit description for the enterprise operating model beyond the OV-1 Operational Concept Graphic. Other groups within the enterprise had struggled or continued to struggle with
trying to specify the desired balance of business process integration and business process standardization and to ascertain how to capture that balance in an enterprise operating model, but none of these were adopted at the enterprise level.

**Case Study Three Key Findings:** In Case Study Three, the enterprise operating model(s) definition, enterprise business process integration, and enterprise business process standardization activities fell under the No Stakeholder Responsibility Domain. Discussions, meetings and working groups about integration, standardization, and transformation occurred frequently and the need for some degree of all was often cited. However, there was little substantive output in terms of an enterprise perspective.

**Case Study Four Key Findings:** In Case Study Four, the enterprise operating model(s) definition, enterprise business process integration, and enterprise business process standardization activities fell under the No Stakeholder Responsibility Domain. The rationale for not performing these activities is justified since the architecture effort was to capture only the current and known-future state (programs of record) of the enterprise. It was anticipated that as the enterprise architecture matured it would assist CS4 in identifying future integration and standardization risks and opportunities that might necessitate a change to the future state of the enterprise. However, the effort was never meant to define a desired end-state of the enterprise, but to document the currently known end-state.
6.4.7 Enterprise Information Technology Principles Definition

**Enterprise Information Technology Principles Definition Concept:** The common finding amongst major empirical studies of enterprise architecture [Hammer86, Weill04, Ross06] highlights that information technology principles are critical elements of successful enterprise architecture efforts. As noted in Section 2.4 above, even [OMB97] recognized the need for information technology principles as the basis for sound enterprise architecture. These principles become the policies and ground rules for employing information technology within the enterprise. Without these policies and ground-rules, there is nothing upon which to base information technology decisions.

**Enterprise Information Technology Principles Definition by Empowered Stakeholders Concept:** Empowered stakeholders determine the desired information technology principles necessary to govern the enterprise.

**Enterprise Information Technology Principles Definition by Surrogate Stakeholders Concept:** Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for determining the desired information technology principles necessary to govern the enterprise to surrogate stakeholders.

**Enterprise Information Technology Principles Definition by No Stakeholders Concept:** No stakeholder is assigned or assumes the responsibility for determining the desired information technology principles necessary to govern the enterprise.
6.4.8 Enterprise Information Technology Principles Definition Case Study

Key Findings

Case Study One Key Findings: In Case Study One, the responsibility for determining the desired information technology principles necessary to govern the enterprise fell under the No Stakeholder Responsibility Domain. As the EBPM was being developed, information technology principles revealed themselves in the modeling process. Once the EBPM was constructed, these principles became the information technology principles of the EBPM and determined the desired balance of business process integration and business process standardization.

Case Study Two Key Findings: In Case Study Two, the responsibility for determining the desired information technology principles necessary to govern the enterprise fell under the No Stakeholder Responsibility Domain. The NSG Enterprise Architecture did not have any explicit description for the information technology principles. The structure of the NSG Enterprise Architecture implied adherence to certain information technology principles, but none were explicitly stated.

Case Study Three Key Findings: In Case Study Three, the responsibility for determining the desired information technology principles necessary to govern the enterprise fell under the No Stakeholder Responsibility Domain.

Case Study Four Key Findings: In Case Study Four, the responsibility for determining the desired information technology principles necessary to govern the enterprise fell under the No Stakeholder Responsibility Domain. As before, the rationale for not
performing this activity is justified as the architecture effort was to capture only the current and known-future state (programs of record) of the enterprise and not to develop new desired outcomes.

6.4.9 Enterprise Core Diagram Development

Enterprise Core Diagram Development Concept: The core diagram should capture the essence of the enterprise’s purpose, boundaries and operating model(s) while being consistent with the enterprise information technology principles. The goal is to have a single, coherent top-level model of the enterprise.

At this point it is important to recall the discussion in Sections 3.2.1, 3.2.2, and 3.2.3 above and reemphasize that one should not consider the activities outlined in this discussion as a linear process. All of these activities associated with Enterprise Purpose Definition, Enterprise Boundary Definition, Enterprise Operating Model(s) Definition, Enterprise Business Process Integration Determination, Enterprise Business Process Standardization Determination, Enterprise Information Technology Principles Definition and Enterprise Core Diagram Development are the beginnings of the System Design Process which is NP-Complete and also a wicked problem. No single linear pass through these activities will satisfactorily address all the issues implied by these activities. Making all the decisions implied by these activities and ensuring that these decisions are aligned is a very difficult (momentous) undertaking. As noted earlier, [Ross06: p.66] revealed that Delta Airlines required 60 iterations to complete their core diagram, and 60
iterations may be much closer to the mean number of iterations needed to do a satisfactory job than the one iteration that most organizations anticipate as sufficient.

**Enterprise Core Diagram Development by Empowered Stakeholders Concept:** Empowered stakeholders develop the core diagram for the enterprise. It is not clear that this activity can be successfully accomplished by any other group besides the empowered stakeholders.

**Enterprise Core Diagram Development by Surrogate Stakeholders Concept:** Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for developing the core diagram for the enterprise.

**Enterprise Core Diagram Development by No Stakeholders Concept:** No stakeholder is assigned or assumes the responsibility for developing the core diagram for the enterprise.

### 6.4.10 Enterprise Core Diagram Development Case Study Key Findings

**Case Study One Key Findings:** In Case Study One, the responsibility for developing the core diagram fell under the Surrogate Stakeholder Responsibility Domain. Since there were no identifiable empowered stakeholders, this activity was delegated to numerous surrogate stakeholders on numerous occasions. DoDAF architecture efforts usually start with a picture because the OV-1 Operational Concept Graphic is the first view to be developed in the Operational Viewpoint. BEA was no exception. The early OV-1 depictions for the BEA showed each Business Domain as its own stovepipe. As
the EBPM development matured, the high-level abstractions shown above in Figure 21, Figure 22, Figure 23, and Figure 24 emerged. None of these could have been drawn a priori.

**Case Study Two Key Findings:** In Case Study Two, the responsibility for developing the core diagram fell under the Surrogate Stakeholder Responsibility Domain. Like Case Study One, Case Study Two was DoDAF based and therefore had an OV-1 Operational Concept Graphic. Architects and graphic artists developed the graphic.

**Case Study Three Key Findings:** In Case Study Three, the responsibility for developing the core diagram fell under the both the Surrogate Stakeholder Responsibility Domain and the No Stakeholder Responsibility Domain. The mission side of the enterprise was applying DoDAF and therefore had an OV-1 Operational Concept Graphic that approximated a core diagram. The non-mission side of the enterprise made no attempt at developing a core diagram.

**Case Study Four Key Findings:** In Case Study Four, the responsibility for developing the core diagram fell under the Surrogate Stakeholder Responsibility Domain. The architecture team constructed a conceptual schema that represented the as-is and known to-be core diagrams and was the foundation for the metamodel specification.

6.4.11 Enterprise Business Question Development

**Enterprise Business Question Development Concept:** Enterprise business questions define the purpose, scope and functionality of an enterprise architecture and are the de
facto requirements for the enterprise architecture. Developing a set of enterprise business questions becomes the basis for defining what an enterprise architecture will do.

**Enterprise Business Question Development by Empowered Stakeholders Concept:** Empowered stakeholders identify the enterprise business questions that will be answered by the enterprise architecture.

**Enterprise Business Question Development by Surrogate Stakeholders Concept:** Empowered stakeholders (or higher echelon stakeholders) delegate the responsibility for identifying the enterprise business questions that will be answered by the enterprise architecture.

**Enterprise Business Question Development by No Stakeholders Concept:** No stakeholder is assigned or assumes the responsibility for identifying the enterprise business questions that will be answered by the enterprise architecture.

### 6.4.12 Enterprise Business Question Development Case Study Key Findings

**Case Study One Key Findings:** In Case Study One, the responsibility for developing enterprise business questions fell under the No Stakeholder Responsibility Domain. No attempt was made to identify enterprise business questions.

**Case Study Two Key Findings:** In Case Study Two, the responsibility for developing enterprise business questions fell under the No Stakeholder Responsibility Domain. No attempt was made to identify enterprise business questions.
Case Study Three Key Findings: In Case Study Three, the responsibility for developing enterprise business questions fell under the No Stakeholder Responsibility Domain. No attempt was made to identify enterprise business questions.

Case Study Four Key Findings: In Case Study Four, the responsibility for developing enterprise business questions fell under the Surrogate Stakeholder Responsibility Domain. Identifying and developing enterprise business questions was at the center of the enterprise architecture development effort in this case study.

6.5 Case Study Epilogue

6.5.1 Case Study One Epilogue

As of this writing, the BEA continues under the auspices of DoD’s Business Transformation Agency. BEA version 7.0 was released on 12 March 2010 and is available on the WWW at:


Shortly after completion of the EBPM v2.2, a new manager for the Business Management Modernization Program was appointed and work on the EBPM was terminated. The EBPM was removed from the BEA and only a few vestiges remain.
6.5.2 Case Study Two Epilogue

NGA has changed leadership, reorganized and initiated a new effort to develop an enterprise architecture using a ten-layer framework analogous to Zachman. One might presume that this alleviates the onerous constraints that Zachman’s five-layer framework imposes on the domain and range of enterprise architecture.

6.5.3 Case Study Three Epilogue

CS3 continues pursuing its enterprise architecture, at least for the mission side of the organization. At a recent architecture working group meeting, an attendee asked who the users of the architecture were. The only users for the architecture were the architects themselves.

6.5.4 Case Study Four Epilogue

CS4 suspended its architecture development effort. CS4 was undergoing a major reorganization resulting in a large staffing cut and may, in the near future, be disestablished.

6.6 Extending the Enterprise Architecture Case Study Framework

The purpose of this research has been to establish an enterprise architecture case study framework that identifies the relevant concept domains and responsibility domains.
instrumental in establishing and sustaining successful enterprise architecture efforts. The goal was to provide a framework in the context of the previously presented Freidman-Sage framework and Martin’s extension of the Freidman-Sage framework.

This research applied this enterprise architecture case study framework to four case studies and evaluated each case study within the context of the 24-cell two-dimensional framework. The integrated conclusion from these efforts is that of all the myriad of activities that could be performed in the cause of establishing an enterprise architecture, the eight enterprise architecture concept domains identified by this research provide useful insight into how well an enterprise has (or has not) established the requisite conceptual integrity needed to develop a successful enterprise architecture.

The focus of this research has been directed towards enterprises that is just starting (or re-starting) to develop enterprise architecture and are trying to ascertain from all the myriad of frameworks, views, models artifacts, etc., just what is important to do first and what may be deferred to later. The goal is to establish some modicum of enterprise architecture competence that provides a useful product that helps the enterprise make better use of its resources. Establishing this baseline threshold of architectural competence is requisite to achieving ongoing architectural success. Once this occurs and value is perceived, then the enterprise may wish to grow its architectural competence in manifold ways and should look to [Rouse09] on how to proceed with enterprise transformation.
7. CONCLUSIONS

Do not be too timid and squeamish about your actions. All life is an experiment. The more experiments you make the better.

~ Ralph Waldo Emerson 1842

7.1 Research Findings

This investigation of enterprise architecture began with a number of assumptions and presumptions that have been contradicted as the investigation progressed. From the embryonic beginnings of enterprise architecture, most prescriptions for architecting (FEAF, TOGAF, DoDAF, etc.) inferred that if followed diligently, the architecting process would lead to a “big success.” Therefore, in the beginning this research assumed that a DoDAF-based approach that integrated all the solution architectures would lead enterprise architecture practitioners to successful outcomes.

What was discovered during this research was that a “big success” cannot be achieved with a solutions-focused approach that lacks conceptual integrity driven down from the top. To be successful, an enterprise architecture must be an architecture of the enterprise; not the integration of all the architectures of the enterprise’s parts.
Enterprise architecture frameworks provide an almost inexhaustible to-do list for the architect, yet few of these to-do items focus the architect on what the empirical research indicates as critical to a successful enterprise architecture effort.

The motivations for enterprise architecture, be that in Frederick Winslow Taylor’s terms from the early 20th century [Taylor11: p.9]:

*The principal object of management should be to secure the maximum prosperity for the employer, coupled with the maximum prosperity for each employee.*

*The words “maximum prosperity” are used, in their broad sense, to mean not only large dividends for the company or owner, but the development of every branch of the business to its highest state of excellence, so that the prosperity may be permanent.*

or be that in James Martin’s terms from the early 21st century [Martin06: p.3]:

*Enterprise architectures are developed for the purpose of discovering more efficient operational approaches for a business venture. Enterprise architecture models are developed to give business managers a better understanding of the things the enterprise owns, operates, and produces so they can make better business decisions.*

are remarkably clear, concise and straightforward. Every enterprise wants to first survive, then thrive, and finally dominate its market, and enterprise architecture is perceived as a way to help an enterprise achieve these goals.

However, as much as all enterprises yearn for the one “right system” that will transform a struggling enterprise into a successful enterprise, as depicted in the allegorical illustration shown in Figure 25, there never has been any “easy road” to success. In fact, there is no convincing evidence that enterprises that pursue enterprise architecture succeed at a rate appreciably higher than those that do not pursue enterprise architecture, and many may be
paying a premium for which no return on investment will ever be realized. Yet the empirical evidence does show that for a small minority, great value can be realized from enterprise architecture.

The purpose of this research has been to discover what can make enterprise architecture a worthwhile endeavor. Along the way, this effort has sought ways to avoid the detours, diversions and distractions along the Road to Success exemplified by wicked problems, complexity and the enterprise learning curve in an attempt to focus the enterprise architecting effort on those aspects of enterprise architecture that can return value.
A motivational poster first issued by the National Cash Register Company was adapted to musical education and published in the October 1913 issue of The Etude [Etude13].

Figure 25: The Road to Success [Etude13]
7.2  **Original Contributions**

The goal of this research has been to develop a process that could increase the likelihood of an enterprise successfully creating an enterprise architecture while decreasing the risks and costs. The approach of this research has been to apply Occam’s Razor to minimize the activities and artifacts required to create an enterprise architecture that is of continuing value to the enterprise.

Toward that goal, this research concentrated on four research objectives. The results of this research include the following five original contributions:

- **Contribution One (Research Objective One): define the focus and scope of enterprise architecture.** The focus of enterprise architecture should be to help the enterprise address the enterprise dilemma. The scope of enterprise architecture should be to specify the degree of business process integration and business process standardization desired for the enterprise. Given this scope, enterprise architecture should be developed to answer enterprise business questions that help address the enterprise dilemma as a means of evolving the enterprise from its current state to its desired state. This differentiates enterprise architecture from systems architecture where enterprise architecture maps the problem space, but is not used to generate solution architectures. Generating solution architectures is the role of systems architecture. The key is that enterprise architecture should constrain the set of
acceptable solutions without specifying any specific solutions. Solutions are the prescriptions of systems architecture, not enterprise architecture.

• **Contribution Two (Research Objective Two): develop a core set of enterprise business questions.** To be of lasting value, an enterprise architecture must be able to perform some ongoing, useful and valuable function. Answering business questions can be the intended use as well as the source of value for an enterprise architecture. As such, Enterprise business questions become the *de facto* requirements for the enterprise architecture, and identifying a robust set of enterprise business questions is the quest of the enterprise architecture effort. This research contributes a baseline set of enterprise business questions that establishes the initial domain and range of the enterprise architecture effort and serves as the starting point from which to begin the enterprise architecting process of developing a robust set of enterprise business questions. Answers to these core enterprise business questions will provide valuable information and thus establish perceived value for the enterprise architecture effort quickly. This aids the fledging enterprise architecture team with developing more complex business questions based on the needs and priorities of the enterprise.

• **Contribution Three (Research Objective Three): develop an enterprise architecture metamodel.** This research developed the Maintain Accountability, Produce Product, and Manage Resources (MAPPMR)
enterprise architecture metamodel. The MAPPMR enterprise architecture metamodel is aligned with the defined focus and scope of enterprise architecture. The MAPPMR enterprise architecture metamodel identifies a baseline set of physical entities, logical entities, and relationships among these entities so that an enterprise architecture model can be developed rapidly to answer core business questions. The motivation of this effort has been to provide value quickly with a modest investment of resources, and then be grown over time.

• **Contribution Four (Research Objective Four): develop a focusing methodology for enterprise architecture development.** Simply knowing what must be done is not sufficient, knowing how to do what must be done is equally necessary to ensure that the effort will more likely achieve the necessary conceptual integrity needed to succeed. Before an enterprise architecture can be developed to answer enterprise business questions, there are decisions that must be made before the enterprise architecture effort begins. Towards this objective, this research identified a set of enterprise architecture prerequisites that must be resolved before enterprise architecture development begins. Failure to do so will result in the effort having to backtrack and reexamine the unresolved prerequisites. These enterprise architecture prerequisites require that the enterprise architecture effort ascertain:
• What is the purpose of the enterprise?

• What are the bounds of the enterprise?

• What are the desired operating models of the enterprise?

• What are the information technology principles of the enterprise?

• What is the core diagram that reflects the desired operating model(s) for the enterprise?

Once these prerequisites are completed, the enterprise architecture effort can begin focusing on developing the robust set of enterprise business questions need to help it address the enterprise dilemma.

The combined effects of these first four contributions have resulted in a new definition of enterprise architecture that has benefited from Occam’s Razor. An enterprise architecture is a tool that assists the enterprise in addressing the enterprise dilemma. The purpose of enterprise architecture is to provide the enterprise a better understanding of the things the enterprise produces, owns and operates so the enterprise can make better decisions regarding sustaining or changing the enterprise. An enterprise architecture contains:

• The statement of purpose for the enterprise,

• The definition of the bounds of the enterprise,
• The description of the desired operating model(s) for the enterprise to include the desired levels of business process integration and business process standardization,

• The information technology principles for the enterprise,

• The core diagram for the enterprise,

• A set of enterprise business questions developed to help the enterprise address the enterprise dilemma as it evolves from its current state to the desired state in concordance with the core diagram, its desired operating model(s) and in compliance with the enterprise principles.

• **Contribution Five: Develop an enterprise architecture case study framework.** This effort extends both the Freidman-Sage and Martin case study frameworks for systems engineering case study research, and provides an enterprise architecture case study framework that shares a common basis for evaluating enterprise architecture case studies in the same conceptual framework as prior system engineering case study research.

### 7.3 Recommendations for Further Research

The enterprise architecture discipline is still in its infancy. There remains much to learn about how to determine the degree of process integration and process standardization appropriate to an enterprise and how to align these with ever-evolving information
technology solutions in a way that upholds the information technology principles of that enterprise. Specific topics of research on enterprise architecture include:

- Investigations into approaches that require the enterprise to expend fewer resources will be of more benefit to the enterprise than approaches that require expending more resources to gather a greater number of artifacts and activities over time. Therefore, research into how to apply Occam’s Razor to the enterprise architecting process, as well as the enterprise architecture itself, will be of most benefit to the enterprise architecting discipline in the foreseeable future.

- Research into distinguishing enterprise architecture from systems architecture while keeping both in concordance will help accommodate a divide-and-conquer strategy for controlling architectural complexity. An enterprise needs both enterprise architecture and systems architecture to both manage its resources and design effective solutions that integrate and standardize all the various systems, systems-of-systems and services needed within the enterprise. Better architecting tools that help manage these various levels of abstraction needed to build the architecture(s) would be of great benefit.

- Additional research into extending the enterprise architecture case study framework beyond the rudimentary stages of early enterprise architecture
development. Extensions to the case study framework to address enterprise transformation would be highly beneficial.
APPENDIX: METAMODEL SPECIFICATION

This appendix contains an example of the Maintain Accountability, Produce Product, Manage Resources Enterprise Architecture (MAPPMR) Metamodel Specification (MMS) as applied to the NASA SMD Enterprise Architecture as described in Sections 5.6.3 and 5.6.4. The MMS captures critical information necessary to establishing and maintaining the conceptual integrity of the enterprise architecture effort. The MAPPMR MMS is updated during each iteration of an enterprise architecture development cycle so that over time the MMS grows to include the entire scope of the enterprise architecture.

The fundamental purpose of the Maintain Accountability, Produce Product, Manage Resources Enterprise Architecture (MAPPMR) Metamodel Specification (MMS) is to provide not only a snapshot of the current enterprise architecture, but also facilitate a history of past versions that can illuminate the evolving context of the enterprise. When an MMS is used with appropriately short development windows (and appropriately smaller development scope), little failures can be identified and remedied before they grow into big failures.
NASA Science Mission Directorate (SMD) Enterprise Architecture Metamodel Specification

05 December 2011

Prepared by
J. Michael Harrell
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scope</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Referenced Documents</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Defining the NASA SMD Enterprise</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>The Purpose of the NASA SMD Enterprise</td>
<td>4</td>
</tr>
<tr>
<td>3.2</td>
<td>The Bounds of the NASA SMD Enterprise</td>
<td>5</td>
</tr>
<tr>
<td>3.3</td>
<td>The Desired Operating Model(s) of the NASA SMD Enterprise</td>
<td>5</td>
</tr>
<tr>
<td>3.4</td>
<td>The Information Technology Principles of the NASA SMD Enterprise</td>
<td>5</td>
</tr>
<tr>
<td>3.5</td>
<td>The Core Diagram for the NASA SMD Enterprise</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Defining the Enterprise Architecture Metamodel for the NASA SMD</td>
<td>7</td>
</tr>
<tr>
<td>4.1</td>
<td>The Purpose of the NASA SMD Enterprise Architecture</td>
<td>8</td>
</tr>
<tr>
<td>4.2</td>
<td>The Role of Business Questions in Metamodeling</td>
<td>8</td>
</tr>
<tr>
<td>4.3</td>
<td>NASA SMD Enterprise Architecture Business Questions</td>
<td>9</td>
</tr>
<tr>
<td>4.4</td>
<td>NASA SMD Enterprise Architecture Business Questions Implemented in this Increment</td>
<td>10</td>
</tr>
<tr>
<td>4.5</td>
<td>Enterprise Architecture Business Questions Implemented in Previous Increments</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>NASA SMD Enterprise Architecture Metamodel Objects &amp; Relationships</td>
<td>12</td>
</tr>
<tr>
<td>5.1</td>
<td>Enterprise Architecture Metamodel Schema</td>
<td>12</td>
</tr>
<tr>
<td>5.2</td>
<td>NASA SMD Enterprise Architecture Metamodel Taxonomy</td>
<td>14</td>
</tr>
<tr>
<td>5.3</td>
<td>Troux Semantics Objects used in the NASA SMD Enterprise Architecture Metamodel</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>NASA SMD Enterprise Architecture Methods</td>
<td>19</td>
</tr>
<tr>
<td>6.1</td>
<td>Role of Methods in Metamodeling</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>NASA SMD Enterprise Architecture Criteria</td>
<td>19</td>
</tr>
<tr>
<td>7.1</td>
<td>Role of Criteria in Metamodeling</td>
<td>19</td>
</tr>
<tr>
<td>7.2</td>
<td>NASA SMD Enterprise Architecture Business Questions implemented as Criteria</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>NASA SMD Enterprise Architecture Views</td>
<td>20</td>
</tr>
<tr>
<td>8.1</td>
<td>Role of Views in Metamodeling</td>
<td>20</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Nature of Change</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>05 Dec 2011</td>
<td>1.0</td>
<td>Initial draft</td>
</tr>
</tbody>
</table>
1 Scope
This document specifies the detailed characteristics of the metamodel being developed for the NASA Science Mission Directorate (SMD) Enterprise Architecture based on the Maintain Accountability, Produce Product, Manage Resources (MAPPMR) enterprise architecture metamodel. The MAPPMR enterprise architecture metamodel is based on the Troux Semantics metamodel in Troux Architect v9.1.0.3638.

Entities, relationships, properties, symbols, methods, criteria and views that are NOT predefined in the Troux Semantics metamodel are specified in this document.

2 Referenced Documents
Metamodeling and Modeling with Troux Semantics: for a description of the Troux Semantics metamodel consult the document Metamodeling and Modeling with Troux Semantics that is provide with the Troux Architect software. Look for the file Metamodeling_and_Modeling_with_Troux_Semantics.pdf located in the directory C:\Program Files\Troux\Troux9.1\help. This metamodel specification identifies enterprise architecture metamodel components that are different from, or in addition to those already contained in the Troux Semantics metamodel.

3 Defining the NASA SMD Enterprise
The information in this section establishes the foundations for developing the NASA SMD Enterprise Architecture. Included here are the:

- Definition of the purpose of the NASA SMD Enterprise,
- The bounds of the NASA SMD Enterprise,
- The desired operating model(s) to include the desired levels of business process integration and business process standardization,
- The information technology principles for the NASA SMD Enterprise,
- The core diagram for the NASA SMD Enterprise.

3.1 The Purpose of the NASA SMD Enterprise
Using the vantage point of space to achieve with the science community and our partners a deep scientific understanding of our planet, other planets and solar system bodies, the interplanetary environment, the Sun and its effects on the solar system, and the universe beyond. In so doing, we lay the intellectual foundation for the robotic and human expeditions of the future while meeting today’s needs for scientific information to address national concerns, such as climate change and space weather. At every step we share the journey of scientific exploration with the public and partner with others to
substantially improve science, technology, engineering and mathematics (STEM) education nationwide

### 3.2 The Bounds of the NASA SMD Enterprise


### 3.3 The Desired Operating Model(s) of the NASA SMD Enterprise

NASA SMD is comprised of four science areas: Earth Science, Heliophysics, Planetary Science, and Astrophysics. The goal is to evolve to a high state of business process integration between the science areas, but NASA SMD does not necessarily require a high degree of business process standardization between the science areas. Within a science area, the goal is again for a high level of business process integration and a moderate to high level of business process standardization. The objective is to provide a common look and feel across all the science areas for NASA SMD’s customers and partners while enabling unique approaches to solving problems that are unique to each science area.

### 3.4 The Information Technology Principles of the NASA SMD Enterprise

- **Principle 1**: principles of information management apply to all organizations within the enterprise.
- **Principle 2**: Information management decisions are made to provide maximum benefit to the enterprise as a whole.
- **Principle 3**: All organizations in the enterprise participate in information management decisions needed to accomplish business objectives.
- **Principle 4**: Enterprise operations are maintained in spite of system interruptions.
- **Principle 5**: Development of applications used across the enterprise is preferred over the development of similar or duplicative applications, which are only provided to a particular organization.
- **Principle 6**: The architecture is based on a design of services, which mirror real-world business activities comprising the enterprise (or inter-enterprise) business processes.
- **Principle 7**: Enterprise information management processes comply with all relevant laws, policies, and regulations.
• **Principle 8**: The IT organization is responsible for owning and implementing IT processes and infrastructure that enable solutions to meet user-defined requirements for functionality, service levels, cost, and delivery timing.

• **Principle 9**: The enterprise’s Intellectual Property (IP) must be protected. This protection must be reflected in the IT architecture, implementation, and governance processes.

• **Principle 10**: Data is an asset that has value to the enterprise and is managed accordingly.

• **Principle 11**: Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations.

• **Principle 12**: Data is accessible for users to perform their functions.

• **Principle 13**: Each data element has a trustee accountable for data quality.

• **Principle 14**: Data is defined consistently throughout the enterprise, and the definitions are understandable and available to all users.

• **Principle 15**: Data is protected from unauthorized use and disclosure. In addition to the traditional aspects of national security classification, this includes, but is not limited to, protection of pre-decisional, sensitive, source selection-sensitive, and proprietary information.

• **Principle 16**: Applications are independent of specific technology choices and therefore can operate on a variety of technology platforms.

• **Principle 17**: Applications are easy to use. The underlying technology is transparent to users, so they can concentrate on tasks at hand.

• **Principle 18**: Only in response to business needs are changes to applications and technology made.

• **Principle 19**: Changes to the enterprise information environment are implemented in a timely manner.

• **Principle 20**: Technological diversity is controlled to minimize the non-trivial cost of maintaining expertise in and connectivity between multiple processing environments.

• **Principle 21**: Software and hardware should conform to defined standards that promote interoperability for data, applications, and technology.
3.5 The Core Diagram for the NASA SMD Enterprise

The NASA SMD Core Diagram is shown in Figure 1:

![NASA SMD Core Diagram]

Figure 1: NASA SMD Core Diagram

4 Defining the Enterprise Architecture Metamodel for the NASA SMD Enterprise

The information in this section establishes the purpose and scope of the NASA SMD Enterprise Architecture. Included here are the:

- The purpose of the NASA SMD Enterprise Architecture,
- Explanation of the role of business questions in metamodeling,
- The set of business questions that the NASA SMD Enterprise Architecture is being developed to answer,
- The NASA SMD Enterprise Architecture business questions implemented in this increment,
- The NASA SMD Enterprise Architecture business questions implemented in previous increments.
4.1 The Purpose of the NASA SMD Enterprise Architecture

The purpose of the NSAS SMD Enterprise Architecture is to aid NASA in managing its resources towards the goal of achieving NASA’s vision:

*To reach for new heights and reveal the unknown so that what we do and learn will benefit all humankind.*

To achieve this vision, NASA SMD explores the Earth, solar system and universe beyond; charts the best route of discovery; and reaps the benefits of Earth and space exploration for society.

This NASA SMD Enterprise Architecture is intended to help NASA SMD better manage its resources in its quest to making new discoveries about the Earth, our solar system and the universe beyond.

4.2 The Role of Business Questions in Metamodeling

Business questions define the purpose, scope and functionality of a model and are the *de facto* requirements for the model. Business questions are used to identify the entities (nouns), relationships (verbs) and methods (functionality) required in the metamodel that will enable the model to function as desired.

Step one is to identify simplex (1*st* order) business questions. Simplex business questions do not require a relationship between entities in order to derive an answer. The following are examples of simplex business questions:

*What are the research assets?*
*What are the decision support tools?*
*What are the application areas?*
*What are the model outputs?*
*What are the system models?*
*What are the science observations?*
*What are the data products?*
*What are the instruments?*
*What are the observatories?*
*What are the decisions?*
*What are the outcomes?*
*What are the benefits?*
*Who are the who?*

Once a representative set of entities has been identified from these simplex business questions, then more complex business questions (n*th* order) need to be decomposed into

---

1 A 1*st* order business question is comprised of one entity or one noun.
a series of compound (2nd order) business questions necessary to answer the complex business question. An example of a complex business question is:

*What research assets currently support an integrated system solution?*

In order to answer this complex business question, it is necessary to decompose the question into compound questions that can be combined to derive the desired answer. In this example, the complex question is a 12th order question since it requires establishing twelve entity-relationship pairings between thirteen entities in order to derive the answer to this complex question. This complex question decomposes as follows:

1. *What research assets are decision support tools?*
2. *What model outputs feed into these decision support tools?*
3. *What system models generate these model outputs?*
4. *What science observations feed into these system models?*
5. *What data products contain these science observations?*
6. *What instruments generate data for these data products?*
7. *What observatories [integrated system solution] carry these instruments?*
8. *What decisions do these decision support tools support?*
9. *What outcomes are generated by these decisions?*
10. *What benefits result from these outcomes?*
11. *Who is impacted by these benefits?*

### 4.3 NASA SMD Enterprise Architecture Business Questions

The goal of the NASA SMD Enterprise Architecture is to answer sophisticated and complex business questions relevant to the sound management of resources and assets into the foreseeable future. When mature, the NASA SMD Enterprise Architecture will be able to answer business questions like the following:

- What is the NASA SMD strategic objective?
- What are the NASA SMD big questions?
- What are the NASA SMD focus areas?
- What are the NASA SMD science domains?
- What are the NASA SMD information needs?
- What are the NASA SMD science requirements?
- What are the NASA SMD programs?
- What are the NASA SMD missions?

---

2 An n<sup>th</sup> order business question is comprised for two entities and an implied relationship, but no direct relationship. The order of a business question is then determined by the determining the number of actual entity-relationship pairings need to derive the answer.

3 A 2<sup>nd</sup> order business question is comprised of two entities and a direct relationship, or two nouns and a verb.
• What are the NASA SMD mission locations and orbits?
• What are the NASA SMD studies?
• What are the NASA SMD Enterprise Architecture principles?
• What are the NASA SMD funding levels?
• What are the NASA SMD science products?
• Who are the customers of the enterprise?
• What are the facilities used by the enterprise?
• What are the production lines used by the enterprise?
• What are the organizations within the enterprise?
• What are the products of a production line?
• What are all the production lines that produce a product?
• Who are the customers of the products of the enterprise?
• What production lines are at what facilities?
• What are the production line components of a given production line?
• When is the earliest end-of-life (EOL) for a production line component on a given production line?
• What is the minimum mean-time-between-failure of any production line component on a given production line?
• What is the minimum actual-time-between-failure of any production line component on a given production line?
• What production line components are acquired by a given acquisition program?
• What technology a given research and development program is developing?
• What are the production line components that contribute to producing a product?
  o What are the production line components of a given production line?
  o What are the products of a production line?
• At what facility is a product produced?
  o What production lines are at what facilities?
  o What are the products of a production line?
• At what facility is a production line component located?
  o What production lines are at what facilities?
  o What are the production line components of a given production line?
• Which acquisition program requires a technology being developed by a research and development program?
  o What technology is being developed by a research and development program?
  o What technologies are required by given acquisition program?

4.4 NASA SMD Enterprise Architecture Business Questions Implemented in this Increment

In striving towards answering more complex business questions, the initial version of the NASA SMD Enterprise Architecture has the capability to answer the following simplex business questions:
The NASA SMD Enterprise Architecture has the capability to answer the following 2nd Order business questions:

- What is the funding profile for a given mission?
- What is the funding profile for a given program?
- What is the funding profile for a given study?
- What is the total funding for a given focus area?
- What focus area does a given mission support?
- What Big Questions does a given mission support?
- What is the location for a given mission?
- What is the intended location for a given program?
- What missions are on or orbiting Mars?

The NASA SMD Enterprise Architecture has the capability to answer the following 3rd or higher Order business questions:

- What is the funding profile for a mission addressing a specific Big Question?
- What is the funding profile for all missions addressing a specific Big Question?
- What is the funding profile for all the Astrophysics missions/programs/studies?
- What missions support Big Question AQ3 and when do those missions reach end of life?
- Will there be a gap between the Hubble Space Telescope end of life and the James Webb Space Telescope initial operational capability?

### 4.5 Enterprise Architecture Business Questions Implemented in Previous Increments

There is no previous increment for this version.
5 NASA SMD Enterprise Architecture Metamodel Objects & Relationships

All objects shall have Name and Description properties (per default). Object properties are specified in this section. If no other properties are specified below, then the object will have only Name and Description properties.

5.1 Enterprise Architecture Metamodel Schema

The current NASA SMD Enterprise Architecture metamodel is shown in Figure 2:

![Diagram of NASA SMD Enterprise Architecture metamodel]

Figure 2: NASA SMD Conceptual Schema

Figure 3 and Figure 4 shows the current NASA SMD Enterprise Architecture schema as implemented:
Figure 3: NASA SMD Enterprise Architecture Schema

Figure 4: NASA SMD Enterprise Architecture Schema with Relationships
5.2 NASA SMD Enterprise Architecture Metamodel Taxonomy

The NASA SMD Enterprise Architecture metamodel taxonomy implemented through this version is shown in Table 1, Table 2, and Table 3 below:

Table 1: NASA SMD Enterprise Architecture Ontology – Maintain Accountability Entities

<table>
<thead>
<tr>
<th>DOMAIN</th>
<th>ABSTRACT TYPE</th>
<th>Enterprise Architecture Ontology for Maintain Accountability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Domain</td>
<td>Data Component</td>
<td>Database Object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database View</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Datastore</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>File</td>
</tr>
<tr>
<td>Document Domain</td>
<td>Document Component</td>
<td>Document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Document Class</td>
</tr>
<tr>
<td>Financial Domain</td>
<td>Financial Component</td>
<td>Standard Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard Cost Element</td>
</tr>
<tr>
<td></td>
<td>Control Component</td>
<td>Charter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contract</td>
</tr>
<tr>
<td>Information Domain</td>
<td>Information Component</td>
<td>Information Object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information Attribute</td>
</tr>
<tr>
<td></td>
<td>Demand Component</td>
<td>Information Group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Need</td>
</tr>
<tr>
<td>Knowledge Domain</td>
<td>Knowledge Component</td>
<td>Knowledge Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knowledge Source</td>
</tr>
<tr>
<td></td>
<td>Knowledge Domain</td>
<td>Skill</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Formal Skill</td>
</tr>
<tr>
<td></td>
<td>Skill Profile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Training Program</td>
<td></td>
</tr>
<tr>
<td>Location Domain</td>
<td>Location Component</td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Extraterrestrial</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In Transit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Earth Orbit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solar Orbit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other Orbit</td>
</tr>
<tr>
<td></td>
<td>Logical Location</td>
<td></td>
</tr>
</tbody>
</table>

Maintain Accountability Entities
Table 2: NASA SMD Enterprise Architecture Ontology – Produce Product Entities

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PRODUCE TYPE</th>
<th>Produce Product Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application &amp; Software</td>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>Architecture</td>
<td>Architecture</td>
<td>Architecture</td>
</tr>
<tr>
<td>Domain</td>
<td>Domain</td>
<td>Domain</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Infrastructure</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>IT</td>
<td>IT</td>
<td>IT</td>
</tr>
<tr>
<td>Organization</td>
<td>Organization</td>
<td>Organization</td>
</tr>
<tr>
<td>Product &amp; Service</td>
<td>Product &amp; Service</td>
<td>Product &amp; Service</td>
</tr>
<tr>
<td>Service</td>
<td>Service</td>
<td>Service</td>
</tr>
<tr>
<td>System</td>
<td>System</td>
<td>System</td>
</tr>
</tbody>
</table>

Produce Product Entities

MAPPMR, Enterprise Architecture Ontology, for Produce Product Entities
Table 3: NASA SMD Enterprise Architecture Ontology – Manage Resources Entities

<table>
<thead>
<tr>
<th>Domain</th>
<th>Abstract Type</th>
<th>MAPPMR Enterprise Architecture Ontology for Manage Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis Domain</td>
<td>Analysis Component</td>
<td>Study</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Study</td>
</tr>
<tr>
<td></td>
<td>Factor</td>
<td>Cause</td>
</tr>
<tr>
<td></td>
<td>External Factor</td>
<td>Implication</td>
</tr>
<tr>
<td></td>
<td>Issue</td>
<td>Issue</td>
</tr>
<tr>
<td></td>
<td>Finding</td>
<td>Recommendation</td>
</tr>
<tr>
<td></td>
<td>Risk</td>
<td>Risk</td>
</tr>
<tr>
<td></td>
<td>Trend</td>
<td>Trend</td>
</tr>
<tr>
<td>IT Architecture Domain</td>
<td>IT Architecture Component</td>
<td>Industry Standard Version</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IT Architecture Domain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IT Architecture Component</td>
</tr>
<tr>
<td></td>
<td>Logical Application</td>
<td>Logical Delegation</td>
</tr>
<tr>
<td>Market Domain</td>
<td>Demand Component</td>
<td>Demand</td>
</tr>
<tr>
<td></td>
<td>Market Opportunity</td>
<td>Market Opportunity</td>
</tr>
<tr>
<td></td>
<td>Market Segment</td>
<td>Market Segment</td>
</tr>
<tr>
<td></td>
<td>Need</td>
<td>Need</td>
</tr>
<tr>
<td></td>
<td>Prerequisite</td>
<td>Requirement</td>
</tr>
<tr>
<td></td>
<td>Requirement</td>
<td>Resource</td>
</tr>
<tr>
<td></td>
<td>Technology</td>
<td>Technology</td>
</tr>
<tr>
<td></td>
<td>Standards</td>
<td>Standards</td>
</tr>
<tr>
<td></td>
<td>Standards Category</td>
<td>Standards Category</td>
</tr>
<tr>
<td>Policy Domain</td>
<td>Policy Component</td>
<td>Design Principle</td>
</tr>
<tr>
<td></td>
<td>Enterprise Policy</td>
<td>Security Policy</td>
</tr>
<tr>
<td></td>
<td>Security Policy</td>
<td>Security Policy</td>
</tr>
<tr>
<td></td>
<td>Policy Rule</td>
<td>Policy Rule</td>
</tr>
<tr>
<td></td>
<td>Standard</td>
<td>Standard</td>
</tr>
<tr>
<td></td>
<td>Standard Candidate</td>
<td>Standard Candidate</td>
</tr>
<tr>
<td></td>
<td>Category (In General Domain)</td>
<td>Standards Category</td>
</tr>
<tr>
<td></td>
<td>Policy Exception</td>
<td>Policy Exception</td>
</tr>
<tr>
<td></td>
<td>Principle</td>
<td>Architecture Principle</td>
</tr>
<tr>
<td>Resource Domain</td>
<td>Resource Component</td>
<td>Resource</td>
</tr>
<tr>
<td></td>
<td>Storage</td>
<td>Storage</td>
</tr>
<tr>
<td>Services Portfolio Mgmt Domain</td>
<td>Service Component</td>
<td>Service Catalog</td>
</tr>
<tr>
<td></td>
<td>Service Catalog</td>
<td>Service Catalog</td>
</tr>
<tr>
<td></td>
<td>Service Level Agreement Entity</td>
<td>Service Level Agreement Template</td>
</tr>
<tr>
<td></td>
<td>Service Offering</td>
<td>Deployed Service Level Agreement</td>
</tr>
<tr>
<td>Strategy Domain</td>
<td>Strategy Component</td>
<td>Strategy</td>
</tr>
<tr>
<td></td>
<td>Strategy</td>
<td>Strategy</td>
</tr>
<tr>
<td></td>
<td>Mission</td>
<td>Mission</td>
</tr>
<tr>
<td></td>
<td>Capability</td>
<td>Capability</td>
</tr>
<tr>
<td></td>
<td>Measure</td>
<td>Measure</td>
</tr>
<tr>
<td></td>
<td>Key Performance Indicator</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>Timeline Domain</td>
<td>Timeline Component</td>
<td>Timeline</td>
</tr>
<tr>
<td></td>
<td>Timeline</td>
<td>Timeline</td>
</tr>
<tr>
<td></td>
<td>Timeline Event</td>
<td>Timeline Event</td>
</tr>
<tr>
<td></td>
<td>Timeline Milestone</td>
<td>Timeline Milestone</td>
</tr>
<tr>
<td></td>
<td>Diagram</td>
<td>Timeline Diagram</td>
</tr>
<tr>
<td>Transition Domain</td>
<td>Time Component</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>Event</td>
</tr>
<tr>
<td></td>
<td>Lifecycle</td>
<td>Lifecycle</td>
</tr>
<tr>
<td></td>
<td>Milestone</td>
<td>Milestone</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>Transition</td>
<td>Transition</td>
</tr>
<tr>
<td></td>
<td>Initiative</td>
<td>Initiative</td>
</tr>
<tr>
<td></td>
<td>Program</td>
<td>Program</td>
</tr>
<tr>
<td></td>
<td>Project</td>
<td>Project</td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Task</td>
</tr>
<tr>
<td>Control Component</td>
<td>Plan</td>
<td>Plan</td>
</tr>
<tr>
<td></td>
<td>Disaster Recovery Plan</td>
<td>Disaster Recovery Plan</td>
</tr>
</tbody>
</table>
5.3 Troux Semantics Objects used in the NASA SMD Enterprise Architecture Metamodel

The NASA SMD Enterprise Architecture metamodel is based on the Troux Semantic metamodel. Section 5.3.1 below identifies and describes the NASA SMD Enterprise Architecture metamodel object classes that have been implemented in NASA SMD Enterprise Architecture. Objects that are implemented in the current version of the NASA SMD Enterprise Architecture are highlighted in blue.

Section 5.3.2 below identifies the object—relationship—object triplets that are implemented in the model that extend the Troux Semantic metamodel.

5.3.1 Entities

The following Enterprise Architecture objects are used as-is from the Troux Semantics Metamodel and form the basis for the collection of Conceptual Entities in the Enterprise Architecture metamodel:

**Analysis Domain:**
- Analysis:
  - Study:

**Application & Software Domain:**
- not used

**Data Domain:**
- not used

**Document Domain:**
- not used

**Governance Domain**
- not used

**Financial Domain**
- Standard Cost:
  - Standard Cost Element:

**Information Domain:**
- Information Group:
  - Science Domain:
    - Need:
      - Information Need:
        - Focus Area:
          - Business Question:
            - Big Question:

**Infrastructure & Hardware Domain:**
- not used

**IT Architecture Domain:**
- not used
IT Patterns Domain: not used
IT Product Domain: not used
IT Service Domain: not used
Knowledge & Skill Domain: not used

Location Domain:
  Location:
  Extraterrestrial:
  In Transit
  Earth Orbit
  Solar Orbit
  Other Orbit

Market Domain:
  Requirement:
  Science Requirement:

Organization Domain: not used

Policy Domain:
  Architecture Principle:

Process Domain: not used

Product & Service Domain:
  Deliverable:
    Product:
      Business Product:
      Science Products:

Resource Domain: not used

Service Architecture Domain: not used

Services Portfolio Management Domain: not used

Strategy Domain:
  Goal:
    Strategic Objective:
  Vision:
  Mission:
5.3.2  Enterprise Architecture Metamodel Relationships

The following NASA SMD Enterprise Architecture metamodel relationships are modifications to or additions to relationships defined in the Troux Semantics metamodel:

No modifications or additions in this version.

6  NASA SMD Enterprise Architecture Methods

6.1  Role of Methods in Metamodelling

No methods are implemented in this version.

7  NASA SMD Enterprise Architecture Criteria

7.1  Role of Criteria in Metamodelling

Criteria are the way Troux Architect specifies queries in the metamodel.

7.2  NASA SMD Enterprise Architecture Business Questions implemented as Criteria

The following are the criteria that implement the NASA SMD Enterprise Architecture business questions identified in section 3 above:

7.2.1  CRITERIA NAME

No criteria are implemented in this version. All business questions specified for this version can be answered with the “Find Instances” function within Troux Architect.

7.2.1.1  BUSINESS QUESTION(s) ANSWERED

No criteria are implemented in this version. See Section 4.4 above for business questions that can be answered with the “Find Instances” function within Troux Architect.

7.2.1.2  IMPLEMENTATION

No criteria are implemented in this version.
8 NASA SMD Enterprise Architecture Views

8.1 Role of Views in Metamodelling
No views are implemented in this version.
REFERENCES
REFERENCES

http://www.cs.cmu.edu/afs/cs/project/able/www/paper_abstracts/wright-tosem97-revision.html


Department of Defense Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance Architecture Framework, version 1.0.


<table>
<thead>
<tr>
<th>Reference</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Carlock06]</td>
<td>Carlock, P. G. and Jo Ann Lane, “System of Systems Enterprise Systems</td>
</tr>
<tr>
<td></td>
<td>Engineering, the Enterprise Architecture Management Framework, and System</td>
</tr>
<tr>
<td></td>
<td>of Systems Cost Estimation,” USC Center for Systems and Software</td>
</tr>
<tr>
<td></td>
<td>Technique (BIAIT)—The New Horizon,” Data Base, Vol. 10, No. 4, pp.3-9,</td>
</tr>
<tr>
<td></td>
<td>Spring 1979.</td>
</tr>
<tr>
<td></td>
<td>Technique (BIAIT) Finding the Big Payoff Areas,” Proceeding s of the</td>
</tr>
<tr>
<td></td>
<td>Origin and Future,” United States Naval Institute Proceedings, Vol. 124</td>
</tr>
<tr>
<td></td>
<td>Design Is an NP-Complete Problem,” Systems Engineering, Vol. 4, No. 3,</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.sie.arizona.edu/sysengr/publishedPapers/NP-complete.pdf">http://www.sie.arizona.edu/sysengr/publishedPapers/NP-complete.pdf</a></td>
</tr>
<tr>
<td>[CIO01]</td>
<td>Chief Information Officers Council, A Practical Guide to Federal</td>
</tr>
<tr>
<td>[CIO99]</td>
<td>Chief Information Officers Council, Federal Enterprise Architecture</td>
</tr>
<tr>
<td></td>
<td>Framework, version 1.1, September 1999.</td>
</tr>
<tr>
<td></td>
<td>Executive Overview,” Technical Report CMU/SEI-96-TR-003, ESC-TR-96-003,</td>
</tr>
<tr>
<td></td>
<td>February 1996.</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.003.html">http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.003.html</a></td>
</tr>
<tr>
<td>[Clements96b]</td>
<td>Clements, Paul C., “Coming Attractions in Software Architecture,”</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.008.html">http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.008.html</a></td>
</tr>
<tr>
<td>[Conklin06]</td>
<td>Conklin, Jeff, Dialogue Mapping: Building Shared Understanding of Wicked</td>
</tr>
</tbody>
</table>


The Etude, Theodore Presser Company, October 1913.


http://www.cs.cmu.edu/afs/cs.cmu.edu/project/able/www/paper_abstracts/archmismatch-icse17.html


Gartner Hype Cycle for Enterprise Architecture 2010.
http://www.slideshare.net/Gartner/gartner-hype-cycle-for-enterprise-architecture-2010-4905726

http://www.seas.upenn.edu/~gaj1/metgg.html


263


<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>

http://portal.acm.org/citation.cfm?id=800078.802526


http://www.cs.toronto.edu/km/tropos/UniCon.pdf

http://www.cs.cmu.edu/~Vit/paper_abstracts/ProgCodif.html


http://books.google.com/

http://books.google.com/


http://www.zifa.com/
J. Michael Harrell was born in Shreveport, Louisiana and graduated from C. E. Byrd High School. He received his Bachelor of Science in Zoology from the University of Oklahoma and upon graduation received a commission in the United States Navy. While serving as a Surface Warfare Officer in the U.S. Navy, he graduated from the Armed Forces Staff College in Norfolk, Virginia, received a Master of Science in Computer Science from Old Dominion University, and served on the faculty of the U.S. Naval War College in Newport, Rhode Island. Since retiring from the U.S. Navy, he has been employed conducting and managing research and development under DARPA contracts at George Mason University, and assisting the U.S. Government develop enterprise and systems architecture while employed by Science Applications International Corporation, Teknowledge Corporation, Illinois Institute of Technology Research Institute (IITRI), The MITRE Corporation and The Aerospace Corporation.