Legacy Interface Adapter Design Modeling

By

Lonney J. Head

A MASTER OF ENGINEERING REPORT

Submitted to the College of Engineering at
Texas Tech University in
Partial Fulfillment of
The Requirements for the
Degree of

MASTER OF ENGINEERING

Approved

______________________________
Dr. A. Ertas

______________________________
Dr. T. Maxwell

______________________________
Dr. M. Tanik

______________________________
Dr. J. Woldstad

October 28, 2007
ACKNOWLEDGEMENTS

I would like to thank my family and friends for their support during the past year and one half. My wife Lori, son Douglas and daughter Rachel provided a substantial amount of support and encouragement throughout the process of completing this program. My children also challenged me to get the best possible grades, so that I could learn as much as possible and keep up with their scholastic achievements.

This report would not have been possible without the support of David Kirsch and Bryan Fox. These gentlemen were instrumental in the development of the legacy interface problem and solution domains. With their expert knowledge and help, it was possible to capture a variety of ideas and solutions in this area. I appreciated the opportunity to be part of their team during the entire Texas Tech Master’s Program.

I would like to express my thanks to the entire Texas Tech leadership staff and guest speakers. Professor Ertas and his Transdisciplinary Program Team are instrumental to the success of the Raytheon – Texas Tech cooperative University/Industry collaborative program. By selecting experts in the respective fields from all over the world and bringing them to Raytheon, I gained a tremendous amount of knowledge and made many new friends.

Finally, I appreciate the support provided by Raytheon managers, Mark Fuerst and Jim Cotterman for providing the means to complete the program on schedule.
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................ 1
ABSTRACT .............................................................. 5
LIST OF TABLES .................................................... 6
CHAPTER I
INTRODUCTION .................................................... 7
CHAPTER II
BACKGROUND .................................................... 8
  2.1 IASD Development ........................................ 8
    2.1.1 Component Oriented Axiomatic Design .......... 8
    2.1.2 Architectural Considerations ...................... 8
    2.1.3 Project Planning ..................................... 9
CHAPTER III
COMPONENT ORIENTED AXIOMATIC DESIGN ............ 10
  4.1 Introduction to the Axiomatic Design Approach ..... 10
  4.2 Defining Customer Needs ................................ 11
    4.2.1 Customer Needs and Constraints ................ 12
      4.2.1.1 Adapter Interface Needs .................... 14
      4.2.1.2 Functionality Needs ......................... 14
      4.2.1.3 Technology Advancement ..................... 15
      4.2.1.4 Test and Verification ......................... 15
      4.2.1.5 System Connectivity .......................... 15
      4.2.1.6 Reduce Total Ownership Costs (TOC) ....... 15
      4.2.1.7 Constraints Identified from Customer Needs 16
    4.2.2 Mature Domain ....................................... 17
    4.2.3 Functional Requirements and Design Parameters Decomposition ............................ 19
      4.2.3.1 Level 1 FRs/DPs ................................ 20
      4.2.3.2 Level 2 FRs/DPs ................................ 21
      4.2.3.3 Level 3 FRs/DPs ................................ 22
      4.2.3.4 Level 3 FRs/DPs (Revised and Grouped) ...... 25
    4.2.3 Resulting Module Definition ........................ 28
      4.2.3.1 Adapter Components ............................ 29
      4.2.3.2 Adapter Simulations ............................ 30
      4.2.3.3 Missing Components (Identification) ......... 32
      4.2.3.4 Missing Components (Developed) ............... 33
      4.2.3.5 Integration ...................................... 35
      4.2.3.6 Adding Components to Mature Domain ........ 37
      4.2.3.7 Software Product (Execution) ................ 38
      4.2.3.8 Axiomatic Design Concluding Comments ....... 38
CHAPTER IV
ARCHITECTURAL CONSIDERATIONS ..................... 40
  4.1 Introduction to the Architectural Design Approach . 40
DISCLAIMER

The opinions expressed in this report are strictly those of the author and are not necessarily those of Raytheon, Texas Tech University, nor any U.S. Government agency.
ABSTRACT

Technology is changing at a very rapid rate. Many deployed/fielded information systems are aging and becoming “legacy” systems that continue to operate and perform as required. However, as time goes by, maintainability of these systems is becoming an increasing issue. Also, the dissemination of information and data from the legacy systems to modernized systems is becoming more prominent within industry. With the technology evolution, there is a need to adapt legacy systems to modern architected systems.

The objective of this project is to educate the reader about available design considerations and processes to consider when developing an “adaptor” type interface with a legacy system. An interface adapter example is utilized throughout this paper to provide the reader with sufficient information to get started on their own design.

This paper is intended to visit several design topics and processes; component-oriented axiomatic design, architectural considerations, and project planning. Through the survey of these topics, the reader will have a framework and a model in which to get a head start. Many of these topics are cross disciplinary in nature and may be used with a variety of systems.
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Level 1 FR/DP Design Matrix</td>
<td>21</td>
</tr>
<tr>
<td>Table 2</td>
<td>Level 2 FR/DP Initial Design Matrix</td>
<td>22</td>
</tr>
<tr>
<td>Table 3</td>
<td>Level 2 FR/DP Decoupled Design Matrix</td>
<td>22</td>
</tr>
<tr>
<td>Table 4</td>
<td>Iteration 1 Level 3 FR/DP Initial Design Matrix</td>
<td>23</td>
</tr>
<tr>
<td>Table 5</td>
<td>Iteration 1 Level 3 FR/DP Design Matrix</td>
<td>24</td>
</tr>
<tr>
<td>Table 6</td>
<td>Final Iteration 1 Level 3 FR/DP Design Matrix</td>
<td>24</td>
</tr>
<tr>
<td>Table 7</td>
<td>Final Level 3 Design Matrix</td>
<td>27</td>
</tr>
<tr>
<td>Table 8</td>
<td>Checking Integrated Components</td>
<td>33</td>
</tr>
<tr>
<td>Table 9</td>
<td>IASD Event /Accomplishment / Criteria Table</td>
<td>81</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

Customers who operate existing legacy data processing systems often have a need to interface those existing legacy systems to new or replacement subsystems. Legacy systems often provide stable low cost processing, but they may need to interface with new/replacement subsystems.

These new or replacement subsystems by their nature utilize more current technology than the legacy system. However, attempting to modify the legacy system directly to interface with the newer subsystem causes an undesirable and often unaffordable ripple of change in the legacy system. Hence, a need arises for an Interface Adapter that fits between the legacy system and the newer subsystem. The adapter can absorb the ripples of change so the newer subsystem can be designed taking advantage of the newest available technology and the legacy system is either not impacted or minimally impacted by this interface modification.

This report leverages the axiomatic design of the Interface Adapter Software Design (IASD) project. The set of customer needs, Functional Requirements (FRs), Design Parameters (DPs) and Constraints applicable to an IASD were developed and will be utilized to the fullest extent. The full design process, description of customer needs (CNs), Functional Requirements decomposition process, associated Design Parameters and Requirements traceability matrix are described in Chapter III.

The vertical requirement decomposition method/tool is utilized to decompose the system design requirements. The first level requirements are developed to describe the highest level of the design description, and an initial/high level design matrix is developed and decoupled to start the hierarchical vertical decomposition approach. An iterative decomposition analysis approach is utilized to refine the various levels after analyzing coupling in the design matrices.

Although this paper is based around the IASD design using component oriented axiomatic design techniques, architectural considerations are leveraged in order to validate the design and to systematically develop the design to a level that could lead directly to implementation.
2.1 IASD Development

During the process of completing the Raytheon/TTU Master’s of System Engineering program, techniques learned throughout the curriculum are utilized throughout this report. In any design project, it is very important to understand the customer needs and the contractual obligations of the project. Although, one would normally start with project planning, this paper examines the design and architectural approaches first. The primary reason for this is to introduce techniques for better understanding the scope of the design prior to completely planning out the project. Also, it is important to understand how to start with an initial set of customer needs and decompose functional requirements and design parameters. Throughout this process, constraints are identified that limit the scope of the project.

2.1.1 Component Oriented Axiomatic Design

Component Oriented Axiomatic Design is an approach that utilizes specific processes to develop a system design to the component level while checking for missing components. Identifying missing components early in the design process is crucial to maintain cost, stay on schedule and enhance performance on the project. Also, missing a component may be detrimental to the success of the project. The earlier missing components are found during the design process, the less time and money that are spent later in the project lifecycle. The design team should consider keeping the customer informed throughout the design process or, better yet, have them actively participate on the team.

2.1.2 Architectural Considerations

The second approach described in this paper utilizes an architectural design approach that combines contextual, operational, logical and physical data to provide a base architecture. This approach, similar to the axiomatic design approach, uses customer needs and requirements as a basis for the
architecture of the system. The difference between the two methods is the detailed processes utilized to describe each part of the system. These details allow a smooth transition to system hardware and software development.

2.1.3 Project Planning

Project planning occurs throughout all stages of project and is considered an evolving process. However, it is important to put together an initial project master plan and integrated master schedule. This is essential to successful project start up and initial project execution. Although it is highly probable that changes will occur during project execution for various reasons, continual adaptation of the master plan and schedule are essential for project completion. The IASD project Integrated Master Plan (IMP) and Integrated Master Schedule (IMS) are developed in this paper so that the reader may gain an understanding of how to approach project planning in a systematic manner.
CHAPTER III
COMPONENT ORIENTED AXIOMATIC DESIGN

4.1 Introduction to the Axiomatic Design Approach

This report uses axiomatic design techniques to present such an Interface Adapter Software Design (IASD). The report iteratively develops a list of customer needs, Functional Requirements, Design Parameters and Constraints applicable to an IASD and analyzes the design steps pointing out desirable and undesirable characteristics.

Two design axioms provide a critical foundation. These axioms are:

- Independence Axiom
  - Maintain independence between functional requirements

- Information Axiom
  - Minimize the information content of the design

The major axiomatic steps are:

1. Identify customer needs
2. Develop functional requirements that satisfy the customer needs
3. Identify design constraints
4. Develop design parameters that satisfy the functional requirements
5. Iterate steps 2, 3 and 4 until three levels of requirements and design parameters are achieved and the Independence and Information Axioms are optimally satisfied.

The axiomatic design approach is presented in Figure 1.
4.2 Defining Customer Needs

The project objective is to design an interface between an existing legacy data system and multiple modernized data systems with specific interface requirements. This project utilizes the Component-Oriented Axiomatic Design (COAD) process to provide the best possible design decisions based upon customer needs and decomposed functional requirements. The customer vision is to, “provide a reliable, cost effective mechanism to interface a legacy data system to a modernized data system with minimum disruption to ongoing operations.”

The customer needs were identified through brainstorming sessions and reviewing past industry project experiences. The author examined various perspectives on similar projects such as customer motivations that drive development of system upgrades, the challenges that are faced integrating older
and newer technology and perspectives from lessons learned in development and integration experience. 

These needs remained constant during the design iterations.

Customer needs would normally be determined in projects of this category by a consortium of personnel including:

- System design engineers
- Legacy knowledgeable systems engineers
- New subsystem design team
- Software engineers and architects
- Network engineers
- Field personnel who operate or maintain legacy systems
- Program management, scheduling, budgeting, etc.

The scenario under examination is a legacy system in which a subsystem is identified and replaced. This new, replacement subsystem uses current communication technology and protocols but the legacy system operates older technology. The primary desire is to develop a testable adapter to translate the two communication protocols (old and new).

4.2.1 Customer Needs and Constraints

The following information describes an initial list of customer needs, known or agreed constraints and definitions/declarations.

1. Adapter shall access subsystem replacement.
2. Adapter shall interface legacy system to subsystem replacement using specified communication protocols.
3. Customer needs a subsystem replaced because manufacturers are not supporting aging parts of system and/or customer needs to migrate parts of system to newer and more flexible technology and make software based.
4. Customer needs more functionality but the legacy system contains subsystems that are functionally limited.

5. Limited funding exists for overhauling the entire system so focus is on subsystem replacement which is more cost effective.

6. Customer needs additional functionality added to the system.

7. The customer needs more current technology for the resulting functionality improvement.

8. Various stakeholders need to be able to test the new subsystem in a stand alone configuration (without access to the legacy system).

9. Various stakeholders need to be able to verify requirements of a new subsystem by having visibility to data-flow within the adapter.

10. Customer needs to utilize standardized technology to make development more cost effective and gain more functionality (get more for less).

11. The current legacy system is incompatible with newer COTS products.

12. The Customer needs data flow improvement to take advantage of increased bandwidth and capacity of communication equipment, (i.e. network switches).

13. Customer needs to reduce equipment footprint size to use less space.

14. Customer needs reduced power requirements.

15. Customer needs reduced heat generation.


17. Customer needs improved maintainability.

18. Customer needs system availability to be at least as good as the current system (24x7 with scheduled maintenance periods - 1000 hours or better continual uptime).

19. Adapter shall be considered part of legacy system.

20. If one subsystem channel’s connectivity is lost, the system considers all are lost.

21. If no response is received from the subsystem or connection is lost on any of the multiple communication channels, all communication with the subsystem is halted.
22. Legacy interface is TCP-IP socket based.

23. Secondary interface shall be Service Oriented Architecture (SOA).

24. Adapter shall be software based running on a standard computer platform.

25. Adapter shall maintain SOA connectivity with Naming Service to resolve single server.

26. Adapter shall provide a sustainable TBD commands per second commanding rate.

27. Adapter design shall scale to match available hardware.

28. For operational mode, adapter does not have to operate if legacy system is not available.

29. Publicly exposed data references in the adapter should be protected (not deleted).

30. Adapter shall trace events to provide diagnostic access to processing.

31. Test scenarios shall utilize the legacy interface of the adapter to test the adapter and subsystem.

The list of customer needs is then grouped into higher level categories. These categories are areas of interest that describe the higher level needs by the customer. This method also helps flush out constraints that may affect the overall design of the final system.

4.2.1.1 Adapter Interface Needs

-1) The adapter shall access subsystem replacement.

-2) The adapter shall interface legacy system to subsystem replacement using specified communication protocols.

-19) Adapter shall be considered part of legacy system.

4.2.1.2 Functionality Needs

4) Customer needs more functionality but the legacy system contains subsystems that are functionally limited.

6) Customer needs additional functionality added to the system.

10) Customer needs to utilize standardized technology to make development more cost effective and gain more functionality (get more for less).
12) The Customer needs data flow improvement to take advantage of increased bandwidth and capacity of communication equipment, (i.e. network switches).

4.2.1.3 Technology Advancement

3) Customer needs a subsystem replaced because manufacturers are not supporting aging parts of system and/or customer needs to migrate parts of system to newer and more flexible technology and make software based.

7) The customer needs more current technology for the resulting functionality improvement.

11) The current legacy system is incompatible with newer COTS products.

4.2.1.4 Test and Verification

8) Various stakeholders need to be able to test the new subsystem in a stand alone configuration (without access to the legacy system).

9) Various stakeholders need to be able to verify requirements of a new subsystem by having visibility to data-flow within the adapter.

30) Adapter shall trace events to provide diagnostic access to processing.

31) Test scenarios shall utilize the legacy interface of the adapter to test the adapter and subsystem.

4.2.1.5 System Connectivity

20) If one subsystem channel’s connectivity is lost, the system considers all are lost.

21) If no response is received from the subsystem or connection is lost on any of the multiple communication channels, all communication with the subsystem is halted.

4.2.1.6 Reduce Total Ownership Costs (TOC)

5) Limited funding exists for overhauling the entire system so focus is on subsystem replacement which is more cost effective.

13) Customer needs to reduce equipment footprint size to use less space.

14) Customer needs reduced power requirements.
15) Customer needs reduced heat generation.
16) Customer needs improved reliability.
17) Customer needs improved maintainability.

4.2.1.7 Constraints Identified from Customer Needs
18) Customer needs system availability to be at least as good as the current system (24x7 with scheduled maintenance periods - 1000 hours or better continual uptime).

22) Legacy interface is TCP-IP socket based.
23) Secondary interface shall be SOA.
24) Adapter shall be software based running on a standard computer platform.
25) Adapter shall maintain SOA connectivity with Naming Service to resolve single server.
26) Adapter shall provide a sustainable TBD commands per second commanding rate.
27) Adapter design shall scale to match available hardware
28) For operational mode, adapter does not have to operate if legacy system is not available
29) Protect publicly exposed data references in adapter.

Figure 2 shows the KJ diagram related to the Customer Needs described above. It is often helpful to view the relationships between the customer needs and the categories into one diagram for quick identification of customer needs and constraints.
4.2.2 Mature Domain

Using knowledge from experience and technology used in similar systems, the following diagrams were developed to show possible mature domains that could be incorporated into the design. These domains will be used to further develop requirements and design parameters. The platform technologies listed in Figure 3 and the COTS and Software technologies listed in Figure 4 show mature domain components that map to the IASD requirements. This provides a component relationship of the IASD requirement so existing mature domain components already available in the industry. These mature domains change over time due to the rapid changes in technology, so taking a snapshot in time may lead to changes down the road. As in the description of “legacy system”, today’s mature domains are tomorrow’s legacy domains.
The following Constraints emerged during the initial stage of the axiomatic design process. The constraints are derived through the analysis of customer needs and act of mapping them to the mature domain components categorized for this application. Constraint C10 emerged later in the design process.

C 1 Legacy interface shall be TCP-IP socket based.
C 2 Secondary interface shall be SOA.
C 3 Adapter shall be software based running on a standard computer platform.
C 4 Maintain SOA connectivity with Naming Service.
C 5 Provide a sustainable TBD commands per second commanding rate.
C 6 Provide for 24 hour, 7 day per week duty cycle with a MTBF (Mean Time Between Failure) of 1000 hours or better.
C 7 Scalable to match available hardware.
C 8 In operational mode, adapter does not have to operate if legacy system is not available.
C 9 Protect (do not delete) publicly exposed data references in adapter.
C 10* Event log service is available to all parts of the adapter for logging trace events.

These system constraints define the adapter’s functional bounds. They also define the inputs for which the adaptor is responsible.

4.2.3 Functional Requirements and Design Parameters Decomposition

The adapter is assumed to operate between the legacy and the new subsystem where it can translate data transmitted in either direction. This position also allows a legacy driver to be connected to this adapter for testing and operation without the need for the legacy system. Figure 5 shows the relationship between the legacy system, adapter (IASD) and a new subsystem.

Figure 5 Base IASD Design
As shown in Figure 1, Functional Requirements (FRs) are developed from Customer Needs (CNs) and Design Parameters (DPs) are developed from the FRs. Each FR captures the concise scope of functions for the given level. Keeping a focus on customer needs insures all system needs are accounted for.

In this case, DP development evolves easily from FRs with the perspective that a DP satisfies everything within the scope of the FR. When the Design Matrix is used to examine coupling, the scope of DPs is made much clearer. This exposes the DP to multiple FR mappings. This process reveals additional information about the FRs (i.e. definition is too broad, functionality is excessive, or that the definition is optimal). By iteratively going over a level of FR/DP development based on Design Matrix analysis, a more sound design is realized as more concise FRs/DPs are realized.

Note: In the next sections, the FR and DP numbering scheme is formed as follows; <Level_1> • <Level_2> • <Level_3>. The sub-level numbers are sequential for the entire level to reflect a FR or DP at the given level, not within the higher level’s FR/DP. This numbering scheme is preferable when working with design matrices, specifically when a matrix needs to be manipulated into lower triangular form.

4.2.3.1 Level 1 FRs/DPs

The following FRs are derived for the first level. The primary function of the adapter is to translate data between two interfaces and secondarily provide a user interface for testing the new subsystem.

FR 1 Interface a legacy system to an upgraded subsystem.

DP 1 Data Translator Adapter.

FR 2 Provide a user driven stand-in legacy driver for testing and verifying new subsystem.

DP 2 Legacy Interface Test Driver and Status GUI.
Table 1 Level 1 FR/DP Design Matrix

<table>
<thead>
<tr>
<th>Level 1</th>
<th>DP1</th>
<th>DP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FR2</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

This matrix shown in Table 1 represents the decoupled relationship between FR1/DP1 and FR2/DP2. This could have been a completely uncoupled matrix (Xs only in diagonal locations), except to satisfy the legacy interface test driver requirement, FR2; the GUI must have access to the Adapter.

4.2.3.2 Level 2 FRs/DPs

The following level 2 FR/DP pairs are derived by analysis of the customer needs. In this case, the foundational functionality emerged from brainstorming the logical separations. Since it is assumed the implementation of the adapter is completed using a standard software language, the level 2 DPs are envisioned as primary functional components in a software design framework.

- FR 1.1 Provide legacy interface to receive and transmit data with legacy system.
  - DP 1.1 Legacy Interface.
- FR 1.2 Provide Subsystem interface to receive and transmit data with the new subsystem.
  - DP 1.2 Subsystem Interface.
- FR 1.3 Translate/manage entire adapter functionality
  - DP 1.3 Adapter Manager
- FR 1.4 Data needs to be inspected and checked against state of data on subsystem interface to determine where a message should be routed (message dependent or independent of current state)
  - DP 1.4 Data Marshaller to determine routing to subsystem
- FR 2.5 Operate adapter via legacy driver without legacy system being available
  - DP 2.5 Legacy Driver / GUI Portal
Reworking the matrix into lower triangular form, the following decoupled version is revealed:

<table>
<thead>
<tr>
<th>Level 2</th>
<th>DP 1.3</th>
<th>DP 1.2</th>
<th>DP 1.1</th>
<th>DP 1.4</th>
<th>DP 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR 1.3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR 1.2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR 1.1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR 1.4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR 2.5</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

The Adapter Manager obviously has a lot of coupling to the rest of the system, but this is by intent to try to keep other parts focused on specific functions. This coupling and others may be reduced during the level 3 FR/DP analysis.

4.2.3.3 Level 3 FRs/DPs

FR 1.3.1 Configure adapter using a configuration input

DP 1.3.1 Configuration File

FR 1.1.2 Interface with Legacy System to transfer data to/from adapter

DP 1.1.2 Legacy Interface

FR 1.2.3 Interface with Subsystem to transfer data to/from adapter

DP 1.2.3 Subsystem Interface

FR 1.3.4 Manage conversion of data in both directions via defined protocols

DP 1.3.4 Adapter Manager Data Converter

FR 1.4.5 Route data to/from multiple channels on subsystem interface
DP 1.4.5 Data Marshaller

FR 1.2.6 Initialize when legacy interface is triggered to be active against legacy subsystem or test driver

DP 1.2.6 Legacy Interface Initializer

FR 1.3.7 Trace events

DP 1.3.7 Event Log

FR 1.2.8 Subsystem Interface will input configuration for a given number multiple channels

DP 1.2.8 Subsystem Interface Initializer

FR 1.5.9 Manage/coordinate initialization sequence of all adapter functions

DP 1.5.9 Adapter manager Initializer

FR 1.3.10 Sequence/coordinate fault recovery/response

DP 1.3.10 Adapter Manager Fault Handler

Table 4 shows the Design Matrix realized from the first iteration of FR/DP pairs.

<table>
<thead>
<tr>
<th>Level3</th>
<th>DP1.3.1</th>
<th>DP1.1.2</th>
<th>DP1.2.3</th>
<th>DP1.3.4</th>
<th>DP1.4.5</th>
<th>DP1.2.6</th>
<th>DP1.3.7</th>
<th>DP2.5.8</th>
<th>DP2.5.9</th>
<th>DP1.3.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1.3.1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.1.2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.2.3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.3.4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.4.5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.2.6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.3.7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR2.5.8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR2.5.9</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FR1.3.10</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Tight coupling between FR1.3.7 and DP 1.3.7 is created because of the event tracing via an event log. This FR/DP is removed from the matrix by defining a new constraint. The new constraint is added as (C10) to ensure this need is covered and to reduce coupling in the matrix. Since the adapter is a
software based application, a common software log service can be made available to all parts of the system. See the results below in Table 5.

Table 5 Iteration 1 Level 3 FR/DP Design Matrix

<table>
<thead>
<tr>
<th>Level3</th>
<th>DP1.3.1</th>
<th>DP1.1.2</th>
<th>DP1.2.3</th>
<th>DP1.3.4</th>
<th>DP1.4.5</th>
<th>DP1.2.6</th>
<th>DP2.5.8</th>
<th>DP2.5.9</th>
<th>DP1.3.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1.3.1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.1.2</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.2.3</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.3.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.4.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FR2.5.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FR1.3.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The level 3 design matrix is then manipulated toward lower triangular form to produce Table 6.

Table 6 Final Iteration 1 Level 3 FR/DP Design Matrix

<table>
<thead>
<tr>
<th>Level3</th>
<th>DP1.3.10</th>
<th>DP1.2.3</th>
<th>DP1.1.2</th>
<th>DP1.3.4</th>
<th>DP1.2.6</th>
<th>DP2.5.9</th>
<th>DP1.3.1</th>
<th>DP1.4.5</th>
<th>DP2.5.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1.2.3</td>
<td>X</td>
<td>_</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.1.2</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.3.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.3.4</td>
<td>X</td>
<td>_</td>
<td>X</td>
<td>_</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR1.2.6</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FR2.5.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FR1.3.1</td>
<td>X</td>
<td>X</td>
<td>_</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FR1.4.5</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FR2.5.8</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Before going further, the design matrix reveals coupling that is difficult to account for in the current level 3 FR/DP definitions. The FR/DPs involving the Legacy Interface and the Subsystem Interface are coupled to other parts of adapter. The single fault handler function is a contributor to this coupling as well as the data conversion being handled in one location.

The analysis shifts back to reexamining the level 3 FR/DPs to more carefully look where functions are performed against the level 2 DPs. The fault handling is uncoupled by breaking out the
respective types of fault handling required and assigning them to respective parts that deal with that portion of the adapter functions. Also, the data conversion is assigned to the respective interfaces so other parts of the adapter do not have to be a part of the specific data protocols.

4.2.3.4 Level 3 FRs/DPs (Revised and Grouped)

**Legacy I/F**

FR 1.1.1 Communicate via legacy protocol on a sequential data channel  
DP 1.1.1 Legacy protocol interpreter (built to match spec)  
FR 1.1.2 Initialize when triggered  
DP 1.1.2 Legacy interface initializer  
FR 1.1.3 Provide interface to legacy driver  
DP 1.1.3 Legacy Driver / GUI Portal

**SubSystem I/F**

FR 1.2.4 Communicate via subsystem I/F protocol  
DP 1.2.4 Subsystem protocol interpreter (built to match spec)  
FR 1.2.5 Initialize when triggered  
DP 1.2.5 Subsystem interface initializer

**Manager**

FR 1.3.6 Input adapter configuration  
DP 1.3.6 Configuration reader  
FR 1.3.7 Manage/coordinate initialization sequence  
DP 1.3.7 Adapter Initializer  
FR 1.3.8 Sequence/coordinate fault recovery/response
DP 1.3.8 Fault Handler

FR 1.3.9 Collect system state

DP 1.3.9 Status Collector

Marshaller

FR 1.4.10 Manage parallel data channels. Assign work to appropriate channel based on configuration and the previous work handled

DP 1.4.10 Data Dispatcher Channel Manager

FR 1.4.11 Store message data while waiting status from subsystem

DP 1.4.11 Waiting-Response-Queue

FR 1.4.12 Reject unknown transmission

DP 1.4.12 Data Dispatcher Xmit Handler

FR 1.4.13 Reject transmissions when triggered

DP 1.4.13 Data Dispatcher Xmit Handler

GUI

FR 2.2.14 GUI interface for legacy driver

DP 2.2.14 GUI Legacy System Application

FR 2.2.15 Display status of interface states/configuration

DP 2.2.15 GUI Application*

FR 2.2.16 Perform diagnostics

DP 2.2.16 GUI Application*

FR 2.2.17 Authenticate GUI user

DP 2.2.17 User Authentication

* These are defined further in the next section.
The Design Matrix shown in Table 7 represents the new level 3 version after altering the position of the rows and columns to get it into lower triangular form. Note that for readability, the following table has DPs as columns and FRs as rows and since the third level number of the FRs and DPs is unique, only the third level number is used. That is, DP 1.3.9 is listed as column 9 and FR 1.3.9 is listed as row 9.

### Table 7 Final Level 3 Design Matrix

<table>
<thead>
<tr>
<th>Level 3</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>9</th>
<th>11</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>8</th>
<th>12</th>
<th>13</th>
<th>10</th>
<th>7</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

The tight coupling revealed in this iteration is isolated to two respective parts of system, the Dispatcher and the GUI. The coupling in the Dispatcher’s functionality (FRs 10, 12, 13) is expected and understandable since all data routing is done at this point. These FRs may be collapsed into a single FR. The breakout of the fault handling after the first level of FRs/DPs enabled helps to determine where that functionality is best served. The two fault handling functions (FRs 12 and 13) may be collapsed together.

The coupling with the GUI in FRs 15-16 is expected since all FRs are served from a single GUI. Collapsing these FRs into one removes coupling in the diagram, however the best solution is to modify
the two respective DPs to clarify which function is for system status and which one is for diagnostic access.

The Adapter Manager’s FRs (7, 8) that require configuration and initialization are coupled with other components. This coupling should not be intrusive to the overall design of the system. All level 3 FR/DP pairs define the scope of a design that may now begin to transition to functional physical components in a logical software design.

4.2.3 Resulting Module Definition

Figure 6 represents major components that emerged from the FR/DP and Design Matrix analysis. The level 2 DPs drive the primary functional components in the design. The level 3 DPs reveal more of the functions within and the relationships between the level 2 DPs.

The configuration and initialization requirements are mapped to three other components as well as the Adaptor Manager as indicated by the Management Function (MF) subcomponents in the diagram.
4.2.3.1 Adapter Components

The association of major adapter components identified in Figure 6 may now be associated with more specific mature domains. This consists of using COTS products, middleware, and software languages and libraries as components of the adapter design.

- Management Function
  - Java package
  - XML (config file)
- Legacy Interface
  - Java package
  - TCP-IP
- Data Marshaller
  - Java package (algorithms)
- Subsystem Interface
  - Web Services
- Event Log
  - Java Logging
- Legacy Driver
  - Database to store data for test
- GUI Display
  - Java Swing for GUIs
4.2.3.2 Adapter Simulations

Simulations may be used to examine the relationships of components and begin to expose the design to the requirements and ensure completeness. The following scenarios are examined using collaboration diagrams to analyze the components and look for functionality completeness.

These scenarios were developed using the steps required for three threads of processing. These threads represent some basic processing paths. The “authentication” component was considered a missing component and is highlighted for discussion later. Figure 7 shows the collaboration diagram of the IASD initialization process.

![Collaboration Diagram (Initialization)](image)

Figure 7 Collaboration Diagram (Initialization)
All three collaboration diagrams (Figures 7-9) highlight the newly added “Authentication” component to show the relationship of the Authenticator in the respective system processes.

Figure 8 Collaboration Diagram (Legacy System Data Process)
Management Function (MF) initializes subsystem connection. Data flows to legacy system as a result. GUI drives data repetitively in a loop through adapter to new subsystem. Data flows back through adapter to GUI. Data cycle repeats.

4.2.3.3 Missing Components (Identification)

The components’ relationships are defined as publishers (P) and/or corresponding subscribers (S) in Figure 6. The objects are shown as the ordered DPs from the final level design matrix shown in Table 8. The “Emergent” column captures the DPs that do not have a corresponding component to publish or subscribe to.
Table 8 Checking Integrated Components

<table>
<thead>
<tr>
<th>Objects</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Legacy I/F</td>
</tr>
<tr>
<td>DP 1.1.1: Legacy Protocol interpreter</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.1.2: Legacy Interface initializer</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.1.3: Legacy Driver</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.2.4: Subsystem protocol interpreter</td>
<td></td>
</tr>
<tr>
<td>DP 1.3.6: Configuration reader</td>
<td>P</td>
</tr>
<tr>
<td>DP 1.3.9: Status collector</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.4.11: Waiting – response - queue</td>
<td></td>
</tr>
<tr>
<td>DP 2.2.14: GUI Legacy system application</td>
<td>S</td>
</tr>
<tr>
<td>DP 2.2.15: GUI Application</td>
<td>S</td>
</tr>
<tr>
<td>DP 2.2.16: GUI Application</td>
<td>S</td>
</tr>
<tr>
<td>DP 2.2.17: User Authentication</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.3.8: Fault Handler</td>
<td>P</td>
</tr>
<tr>
<td>DP 1.4.10: Data Dispatcher Channel Manager</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.4.12: Data Dispatcher Xmit Handler</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.4.13: Data Dispatcher Xmit Handler</td>
<td>S</td>
</tr>
<tr>
<td>DP 1.3.7: Adapter initializer</td>
<td></td>
</tr>
</tbody>
</table>

At this point, the publish-subscribe condition is broken and one emergent component is exposed. The component needed is an authenticator to validate the user access to the GUI. The missing Authenticator component is added and so the publish-subscribe conditions are satisfied.

4.2.3.4 Missing Components (Developed)

The missing component provides authentication of users accessing the system via the GUI to operate the legacy data driver and access diagnostic data. The authentication component is attached to the GUI since users will be required to have user ids and passwords entered for authentication. The missing component, “Authentication” is added to the component diagram (Figure 6) and is shown in Figure 10.
The authentication component is integrated into the system and the new functionality as shown in the sequence diagram represented in Figure 11. Since authentication involves validating user access and provides different access rules for testing, administrative, and trouble shooting, it will provide various levels of accessibility. These requirements can be covered using Lightweight Directory Access Protocol (LDAP). This protocol is a mature domain in itself and can be implemented by many different COTS products, typically web servers.
4.2.3.5 Integration

Integration includes the development of components within the design. The UML diagrams show in Figure 12 and Figure 13 the integration of the java classes that came from the design. These diagrams represent a start to the primary classes, operations, and attributes that will exist in the software system. The integration of these classes to domains like TCP/IP and Database are shown in the UML diagrams.

At this point the test cases may be used to check out functionality through the integration process. Low level unit tests may be developed to check out component functionality and interfaces. The functional requirements test cases validate overall system functionality as the system is further integrated.
Figure 12 UML Diagram (Legacy and GUI)
4.2.3.6 Adding Components to Mature Domain

The missing authentication component that provides system security is common to both software systems operations and testing functions. The selection of LDAP for this design is a potential component that may be added to the list of mature domains originally identified in Figure 4.

Figure 14 shows an update to Figure 4 (Mature Domains) with security added to the mature domain components for the missing component, “Authentication”.

Figure 13 UML Diagram (Dispatcher and Subsystem I/F)
4.2.3.7 Software Product (Execution)

The system execution phase of an application such as this Adapter begins when major component development is complete and full system testing may begin with the external interfaces. An application of this type may be deployed when all functional requirement test cases have passed test validation. Customer acceptance should come upon approval of all test results.

4.2.3.8 Axiomatic Design Concluding Comments

The Component-Oriented Axiomatic Design process provides a clear process to identifying design components. The paper, *Systematic Component-Oriented Development with Axiomatic Design*, [Togay, Dogru, Tanik, Tate] presented the concept with the COAD process of discovering missing components in the design as seen by the missing user authentication component in the Adapter design.

Security added to Mature Domain to cover the “Authenticator”
By utilizing the COAD process, perceived customer confidence is heightened because the results of the final design meet the customer needs. The process brings a focus on the important functions of a system and how multiple functions best relate to each other. With the design process steps unique to COAD, the designer may identify missing components that are required to meet defined functional needs without altering the design dramatically.
4.1 Introduction to the Architectural Design Approach

The purpose of this chapter is to illustrate an approach to architecting the IASD Product utilizing techniques that systematically develop the architecture through a series of distinct processes. Figure 15 shows the components used to develop the IASD architecture.

4.1.1 Contextual Architecting

The objectives of Contextual Architecting are:

- Gather contextual data for the architecture
- Organize and evaluate contextual information
- Define scope and boundaries for the system of interest
- Identify and prioritize quality attributes for the system of interest
The tasks associated with gathering Contextual Architecting are:

1. Gather interrogative data (why, who, what, how, where, and when)
2. Organize interrogative data
3. Perform review of interrogative data
4. Define scope and boundaries
5. Perform review of scope and boundaries
6. Identify and prioritize quality attributes
7. Perform review of quality attributes

4.1.1.1 Gathering Interrogative Data

When gathering interrogative data, it is important to ask, “why does this work need to be done?” and “who are the participants to do the work?” To illustrate why the work must be done for the IASD project, we first must understand the goal of the project. The goal of the project is to

“Provide a reliable, cost effective mechanism to interface a legacy data system to a modernized data system (or systems) with minimum disruption to ongoing operations.”

Figure 16 shows a breakdown of the primary goal (shown above) into subordinate goals that may be developed in detail. This is similar to the process used when decomposing requirements.
In order to accomplish the goals of the IASD project, participants may include:

- Program management, scheduling, budgeting, etc.
- Legacy knowledgeable systems engineers
- Operations & Maintenance (O&M) personnel
- Software engineers
- System architects
- Network engineers
- Legacy System O&M subject matter experts
4.1.1.2 Organizing Interrogative Data

Once the goals are established, it may be useful to construct a relationship matrix in order to map the goals to the participants on the project. An example of this type of a matrix is shown in Figure 17.

![Figure 17 Relationship Matrix – Goals vs. Participants](image)

**Figure 17 Relationship Matrix – Goals vs. Participants**

### 4.1.1.3 Reviewing Interrogative Data

During the process of reviewing the interrogative data, it is important to understand the constraints that exist in the architectural context. Examples of constraints associated with the IASD project are:

- Customer needs system availability to be at least as good as the current system (24x7 with scheduled maintenance periods - 1000 hours or better continual uptime).
- Legacy interface is TCP-IP socket based
- Secondary interface is SOA based
- Adapter shall be software based running on a standard computer platform
• Adapter shall maintain SOA connectivity with Naming Service to resolve single server
• Adapter shall provide a sustainable TBS (to be supplied) commands per second commanding rate
• Adapter design shall scale to match available hardware
• For operational mode, adapter does not have to operate if legacy system is not available
• Assume one input legacy interface and scaleable (up to 5) output interfaces

While constraints are important, it is important to look for opportunities (i.e. hardware and/or software reuse). By looking for opportunities, there may be cost, schedule, or performance possibilities that had not been considered initially.

4.1.1.4 Defining Scope and Boundaries

The scope for the project is to build an interface to adapt a legacy system to a more modern data system. For example, the adaptor provides an interface to one legacy system and up to five new (more modern) systems. Figure 18 illustrates an example of defining the scope of the project by drawing an oval around the area of interest. For this example, it is assumed that the external interfaces are geographically located within the United States.
4.1.1.5 Reviewing Scope and Boundaries

Once the project scope and boundaries are set, it is important to convey the information to the team on a continuing basis. It is very easy to allow “scope creep” into a project, so it is advised to keep the established scope and boundary information continuously communicated and available for review.
4.1.1.6 Identify and Prioritize Quality Attributes

Quality Attributes consist of key customer care-abouts. These Quality Attributes describe what the customer cares about. In this case, the customer is interested in on-demand availability, performance and maintainability. The resulting quality attributes are prioritized on customer needs and priorities which influence the architectural design of the IASD. Figure 19 shows an example of how quality attributes might be characterized and ranked by importance to the customer.

![Figure 19 Customer Ranked Quality Attributes](image-url)

**Figure 19 Customer Ranked Quality Attributes**
4.1.1.7 Perform Review of Quality Attributes

During the review of quality attributes of the system, it may be helpful to start thinking of key scenarios to prepare for the next section, “Operational Architecting”. For the IASD project, the assumed key scenarios are:

- Transmit data to new sites upon request
- Monitor data and command flow
- Query data holdings on legacy system

4.1.2 Operational Architecting

The three scenarios above are used as examples throughout this chapter in order to provide three separate examples of the operational architecting process.

The objectives of Contextual Architecting are:

- Using contextual information, discover and document key operational information for the system of interest
- Expand operational understanding by developing and reviewing operational artifacts
- Evaluate work against contextual artifacts

The tasks associated with gathering Contextual Architecting are:

1. Define an operational concept for a solution
2. Identify key scenarios and actors
3. Select use cases for expansion
4. Create use case graphics
5. Expand selected use case(s) into formal text
6. Create selected activity diagrams
4.1.2.1 Define an Operational Concept for the Solution

Operational Concept (i.e. Concept of Operations – ConOps) helps define how the IASD product is going to be used. The key attributes of an Operational Concept are:

1. Scope
2. Reference Documents
3. User-Oriented Operational Description
4. Operational Needs
5. System Overview
6. Operational and Support Environment
7. Operational Scenarios

4.1.2.2.1 Scope

The scope of the IASD project is to build an interface that adapts a legacy system to a more modern data system by providing an interface between one legacy system and up to five new systems. The purpose is to provide an ability to share data from a legacy data system with some defined number of data holdings to other data systems that have a more modern architecture (as previously described in the contextual architecting section). The objective is to design an interface that meets customer requirements for system availability and monitoring while having data available upon request by any one of five external sites. The first goal is to execute the program within cost, schedule and technical performance as specified in the contract. The second goal is to provide a functional, reliable, maintainable, useable and complete adapter as specified in the contract. The intended audience consists of Program Management,
Operations and Maintenance Staff, End Users and Clients. The overarching vision of the customer is to “Provide a reliable, cost effective mechanism to interface a legacy data system to a modernized data system with minimum disruption to ongoing operations”.

4.1.2.2.2 Reference Documents

Reference documents are useful to understand the existing legacy system, describe the new system and understand the contractual requirements of the project. As an example, below is a list of possible reference documents that may be available for a project like this one:

- Legacy Concept of Operations
- Legacy Interface Specification
- New Subsystems Concept of Operations
- New Subsystems Interface Specification
- Systems Requirements Specification (SRS)
- System Requirements Document (SRD)
- Customer Statement of Work (SOW)
- Customer Contract
- Others not specified

4.1.2.2.3 User Oriented Operational Description

When gathering key user oriented operational information, it is important to describe the user oriented operational information related to user activities, order of user operations, operational process procedures and the organizational/personnel structure. For the IASD project, the following is an example of the user oriented operational description:
• User Activities
  o Users (new subsystems) request list of data holding on legacy system
  o Users select data for retrieval and execute command to “get” data
  o Users monitor data flow through system
  o Users operate and maintain system

• Order of user operations
  o Query, order, get, and store data
  o Monitor system for normal operations & anomalous conditions
  o Proactively disposition anomalous condition and disposition user requests or problems
  o Store system and user statistical information; provide real-time system status, post anomalous condition status, and other data as specified by contract

• Operational process procedures
  o Interface adapter System Operations Procedures (SOPs)

• Organizational/Personnel Structure (refer to Figure 20, Customer Organization diagram)

Figure 20 Customer Organization
4.1.2.2.4 Operational Needs

Because the end solution for the IASD is an operational environment for end users, it is important to understand the customer’s specific goals and objectives as they relate to the operational environment. Although the new operational environment may have updated hardware, software, and/or processes, it is essential that both the new and legacy environments are characterized. The customer specific goals and objectives related to the new operational environment and the existing legacy system description are:

- **Customer Specific Goals and Objectives**
  - Provide high leverage customer hot buttons from contextual quality attributes
    - Functional, reliable, maintainable, useable and complete
  - Provide medium quality attributes
    - Internally consistent, scalable, accreditable, implementable, expandable, adaptable, and standards compliant
  - Provide low quality attributes
    - Loosely coupled, testable, modular and open

- **Objectives**
  - Interface that meets requirements for system availability and monitoring
  - Data available upon request to up to five external continental U.S. sites

- **Existing System (Legacy System)**
  - Consists of aged architecture, hardware and interfaces
  - System is updated as needed to combat obsolescence issues
  - “If it ain’t broke, don’t fix it” type of system & architecture
    - Performs its job, but has no ability to expand to include additional users without some kind of interface adapter system
4.1.2.2.5 System Overview

Although the system overview has been described earlier, it is a good idea to restate the system overview with respect to the concept of operations. Examples of the information one might be interested to specify are shown below.

- **Scope**
  - Build an interface to adapt a legacy system to a more modern data system
  - Provide an interface between one legacy system and up to five new systems

- **Interfaces**
  - Direct connectivity between Legacy system and adapter
  - Direct connectivity available for up to five new subsystems
  - Bi-directional data flow from any external interface (I/F)
  - Legacy protocol is on a sequential data channel
  - New subsystem protocol is a parallel set of data channels

- **System Capabilities**
  - Interface legacy system to new subsystems
  - Additional functionality
    - Provide legacy data to multiple output sources in parallel manner to multiple sites
    - Provide diagnostic capabilities for monitoring data flow, gathering statistical information and fault isolation capabilities

- **Goals and Objectives (high level - interrogatives)**
  - Reduce Total Ownership Costs (TOC)
  - Provide additional functionality
4.1.2.2.6 Operational and Support Environment

The operational and support environment definition is important for understanding the physical aspects of the new system. Some of the key operational and support environment information to consider are:

- **Facilities**
  - Facilities will be customer and client data centers located in the U.S.
  - Assume operational computer facility (customer furnished space, electrical, HVAC, etc.)

- **Equipment**
  - Standard, readily available computer systems & printers

- **Hardware**
  - Computers and printers provided by contractor; offices/workspaces provided by customer

- **Software**
  - Delivered with computer hardware, configuration managed on legacy site

- **Personnel**
  - Key operations personnel (System Administrators, Customer Care), maintenance personnel (Software, Hardware, Logistics), end users and clients

- **Operational Procedures**
  - Systems Operating Procedures developed and delivered with system

- **Support necessary to Operate the Deployed System**
  - Factory Support available for Problem Management, Software System Upgrades and Engineering Changes
4.1.2.2.7 Operational Scenarios

Developing operational scenarios is an excellent way to understand how the system should operate. It allows collaboration across the organization regarding the way in which the system should be operated in real time. For a project the size of the IASD, there would likely be more than a dozen different operational scenarios - possibly more. In order to give the reader an idea of how this section works, three separate scenarios are chosen:

1. Query data holdings on legacy system
2. Transmit data to new sites upon request
3. Monitor system status and data/command flow through adapter

Each scenario description generally has two parts, a descriptive section and a stress/failure section. The examples below illustrate the three scenarios listed above.

- Query data holdings on legacy system (1 of 3)
  - Subsystem sites may query legacy data holdings
    - Subsystem site executes query, query is received and managed by adapter, adapter sends request to legacy system (as would a normal legacy system query operation), and legacy system results are sent back to adapter and forwarded to subsystem requestor.
  - Stress/Failure Scenario
    - Subsystem channel connectivity lost/degraded, operations intervention required for fault analysis, customer care and problem resolution.

- Transmit data to new sites upon request (2 of 3)
  - Subsystem site may order data on an adhoc basis or by standing order
    - Subsystem site executes data request command, data request command managed by adapter, adapter sends request to legacy system, legacy
system retrieves and sends data to adapter, and adapter forwards data to subsystem site.

- Subsystem site executes a standing order command for a registered data type, standing order is managed by adapter, adapter sends standing order request to legacy system, legacy system stores the order, sends an acknowledgement to the interface adapter, the interface adapter sends the acknowledgement of the standing order to the subsystem site; concurrently, the legacy system checks to see if data holdings are available, if holdings available, data is sent to adapter, adapter sends data to subsystem user.

  - Stress/Failure Scenario
    - Subsystem channel connectivity lost/degraded, operations intervention required for fault analysis, customer care and problem resolution

- Monitor system status and data/command flow through adapter (3 of 3)
  - Data monitoring available to Operations and Maintenance personnel through a graphical user interface (GUI) into the interface adapter

    - Operations & Maintenance (O&M) personnel initiate system monitoring, GUI displays default system monitoring screen data, and monitor data flow for normal and anomalous conditions.

  - Stress/Failure Scenario

    Subsystem channel connectivity lost/degraded, operations intervention required for fault analysis, customer care and problem resolution, and customer care notifies subsystem users of status.
4.1.2.2 Identify Key Scenarios and Actors

An effective method to identify and describe key scenarios and actors is by the use of “use cases”. These use cases provide a description, goal, actor’s preconditions, triggers, flows, post conditions and supplemental requirements. Generally, the use case description is accompanied by a use case diagram. Use cases are noted as UC1, UC2 and UC3 (respective of the three scenarios described above).

UC1 – Query data holdings on legacy system

- Description
  - Subsystem user query’s Legacy System data holdings

- Goal
  - View Legacy System data holdings

- Actors
  - Subsystem User, Customer Care and Interface Adapter Operator

- Preconditions
  - Legacy, Interface Adapter and Subsystem sites operational

- Triggers
  - Query command executed from Subsystem user

- Flows
  - Basic
    - Subsystem user enters a query command, query is received by the interface adapter, interface adapter sends query command to Legacy system, Legacy system results are sent back to interface adapter and forwarded on to user, Interface adapter Operator monitors data flow.
  - Alternate
    - None
  - Exception
- Contact customer care operations if operations do not work, customer care works with interface adapter operator to investigate problem, problem is resolved, customer care operator contacts user, and query is executed again until request is satisfied.

- Post Conditions
  - Legacy, interface adapter and Subsystem sites operational

- Supplemental Requirements
  - None

Figure 21 shows the respective use case diagram describing UC1.
UC2 – Transmit data to new sites upon request

- Description
  - Subsystem user may order on an adhoc basis or by standing order

- Goal
  - Order and receive data requested

- Actors
  - Subsystem Client, Customer Care and Interface Adapter Operator

- Preconditions
  - Legacy, interface adapter and Subsystem sites operational

- Triggers
  - “Data Request” command executed from subsystem user
  - “Standing Order” command executed from subsystem user

- Flows
  - Basic
    - Subsystem user enters a data request command, data request command is sent to interface adapter, interface adapter sends data request command to Legacy System, Legacy system retrieves and sends data to interface adapter, and interface adapter forwards data to Subsystem User.
    - Subsystem user enters a standing order command, standing order command is sent to interface adapter, interface adapter sends standing order command to Legacy System, adapter sends standing order request to Legacy System, legacy system stores the order, sends an acknowledgement to the interface adapter, the interface adapter sends the acknowledgement of the standing order to the Subsystem user; concurrently, the Legacy system checks to see if data holdings are
available, if holdings available, data is sent to adapter, and adapter sends data to subsystem user

- Alternate
  - Subsystem contacts Customer Care to place Standing order

- Exception
  - Contact customer care operations if operations do not work, customer care works with interface adapter operator to investigate problem, problem is resolved, and customer care operator contacts user and query is executed again until request is satisfied

- Post Conditions
  - Legacy, interface adapter and Subsystem sites are operational

- Supplemental Requirements
  - None
Figure 22 shows the respective use case diagram describing UC2.

UC3 – Monitor system status and data/command flow through adapter

- Description
  - Data monitoring available to Operations and Maintenance personnel through a graphical user interface (GUI) into the interface adapter
- Goal
  - Monitor system status and data/command flow through adapter
- Actors
- Interface Adapter Operator

- Preconditions
  - Legacy, interface adapter and Subsystem sites operational

- Triggers
  - System Monitoring initiated by operator, fault information send to operator from system, status information sent to operator from system, data/command information sent to operator from system

- Flows
  - Basic
    - O&M personnel initiates system monitoring, GUI displays default system monitoring screen data, and monitors data flow for normal and anomalous conditions
  - Alternate
    - None
  - Exception
    - Subsystem channel connectivity lost/degraded, operations intervention required for fault analysis, customer care and problem resolution, and customer care notifies subsystem users of status

- Post Conditions
  - Legacy, interface adapter and Subsystem sites operational

- Supplemental Requirements
  - None
Figure 23 shows the respective use case diagram describing UC3.

![UC3 Pictorial Diagram](image)

**Figure 23 UC3 Pictorial**

### 4.1.3 Logical Architecting

- **Objectives**
  1. Define a logical organization for the system of interest
  2. Evaluate work against contextual and operational artifacts
- **Tasks**
1. Using operational information, create a decomposition of the operational content of your architecture
2. Aggregate the leaf nodes of your decomposition to create a logical solution approach
3. Create a domain collaboration diagram or logical block diagram describing your approach
4. Create a data flow diagram for your domains or blocks
5. Create one or more sequence diagrams from your logical perspective
6. Evaluate the results of your work using the quality attributes you have defined for your problem
7. Iterate the tasks to see if you can find a better approach

4.1.3.1 Decomposition of the Operational Content of the Architecture

When gathering interrogative data, it is important to ask, “why does this work need to be done?” and “who are the participants. Figure 24 is an example of how interrogative data may be represented to show the direct relationship between the use case sequence step and associated function or functions used to perform the activity.

Figure 24 shows the sequence/function relationship for Use Case UC1 – Query data holdings on legacy system.
<table>
<thead>
<tr>
<th>UC 1 Sequence</th>
<th>UC 1 Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Query data</td>
<td>1. Subsystem X</td>
</tr>
<tr>
<td>2. Receive data query</td>
<td>2. Management Function, Subsystem I/F</td>
</tr>
<tr>
<td>5. Send query</td>
<td>5. Management Function, Legacy Interface</td>
</tr>
<tr>
<td>7. Send query results</td>
<td>7. Legacy System</td>
</tr>
<tr>
<td>10. Log results</td>
<td>10. Event Log</td>
</tr>
<tr>
<td>11. Forward results</td>
<td>11. Management Function, Subsystem I/F</td>
</tr>
</tbody>
</table>

**Figure 24 UC1 Decomposition of Operational Content**

Figure 25 shows the Decomposition of Operational Content in a graphical format for Use Case UC1.

**Figure 25 UC1 Decomposition of Operational Content Diagram**
Figure 26 shows the sequence/function relationship for Use Case UC2—Transmit data to new sites upon request.

<table>
<thead>
<tr>
<th>UC 2 Sequence</th>
<th>UC 2 Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Query data</td>
<td>1. Subsystem X</td>
</tr>
<tr>
<td>2. Receive data query</td>
<td>2. Management Function, Subsystem I/F</td>
</tr>
<tr>
<td>5. Send query</td>
<td>5. Management Function, Legacy Interface</td>
</tr>
<tr>
<td>7. Send query results</td>
<td>7. Legacy System</td>
</tr>
<tr>
<td>10. Log results</td>
<td>10. Event Log</td>
</tr>
<tr>
<td>11. Forward results</td>
<td>11. Management Function, Subsystem I/F</td>
</tr>
</tbody>
</table>

**Figure 26 UC2 Decomposition of Operational Content**

Figure 27 shows the Decomposition of Operational Content in a graphical format for Use Case UC2.
Figure 27 UC2 Decomposition of Operational Content Diagram

Figure 28 shows the sequence/function relationship for Use Case UC3 – Query data holdings on legacy system.

<table>
<thead>
<tr>
<th><strong>UC 3 Sequence</strong></th>
<th><strong>UC 3 Function</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start Monitor</td>
<td>1. IA GUI, Management Function</td>
</tr>
<tr>
<td>2. Query system status</td>
<td>2. IA GUI, Management Function, Legacy I/F, Data Marshaller, Subsystem I/F, Event Log</td>
</tr>
<tr>
<td>3. Receive system status</td>
<td>3. IA GUI, Management Function, Legacy I/F, Data Marshaller, Subsystem I/F, Event Log</td>
</tr>
<tr>
<td>4. Query data information</td>
<td>4. IA GUI, Management Function, Event Log</td>
</tr>
<tr>
<td>5. Receive data information</td>
<td>5. IA GUI, Management Function, Event Log</td>
</tr>
<tr>
<td>6. Query command info</td>
<td>6. IA GUI, Management Function, Event Log</td>
</tr>
<tr>
<td>7. Receive command info</td>
<td>7. IA GUI, Management Function, Event Log</td>
</tr>
<tr>
<td>8. Query event log info</td>
<td>8. IA GUI, Management Function, Event Log</td>
</tr>
</tbody>
</table>

Figure 28 UC3 Decomposition of Operational Content
Figure 29 shows the Decomposition of Operational Content in a graphical format for Use Case 3.

Figure 29 UC3 Decomposition of Operational Content Diagram

With the decomposition of operational content for the use cases is complete, the FRs and DPs may be grouped with components as shown in Figure 30. This allows the development team to design the components around FRs and DPs that meet operational context needs.

<table>
<thead>
<tr>
<th>Legacy I/F</th>
<th>Data Marshaller</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR 1.1.1 Communicate via legacy protocol on a sequential data channel</td>
<td>FR 1.4.1 Manage parallel data channels. Assign work to appropriate channel based on configuration and the previous work handled</td>
</tr>
<tr>
<td>DP 1.1.1 Legacy protocol interpreter</td>
<td>FR 1.4.2 Store message data while waiting status from subsystem</td>
</tr>
<tr>
<td>FR 1.1.2 Initialize when triggered</td>
<td>DP 1.4.2 Waiting-Response-Queue</td>
</tr>
<tr>
<td>DP 1.1.2 Legacy interface initializer</td>
<td>FR 1.4.3 Reject unknown transmission</td>
</tr>
<tr>
<td>FR 1.1.3 Provide interface to legacy driver</td>
<td>DP 1.4.3 Data Dispatcher</td>
</tr>
<tr>
<td>DP 1.1.3 Legacy Driver / GUI Portal</td>
<td>FR 1.4.4 Reject transmissions when triggered</td>
</tr>
<tr>
<td>SubSystem I/F</td>
<td>DP 1.4.4 Data Dispatcher</td>
</tr>
<tr>
<td>FR 1.2.1 Communicate via subsystem I/F protocol</td>
<td></td>
</tr>
<tr>
<td>DP 1.2.1 Subsystem protocol interpreter</td>
<td></td>
</tr>
<tr>
<td>FR 1.2.2 Initialize when triggered</td>
<td></td>
</tr>
<tr>
<td>DP 1.2.2 Subsystem interface initializer</td>
<td></td>
</tr>
<tr>
<td>Manager</td>
<td></td>
</tr>
<tr>
<td>FR 1.3.1 Input adapter configuration</td>
<td></td>
</tr>
<tr>
<td>DP 1.3.1 Configuration reader</td>
<td></td>
</tr>
<tr>
<td>FR 1.3.2 Manage/coordinating initialization sequence</td>
<td></td>
</tr>
<tr>
<td>DP 1.3.2 Adapter initializer</td>
<td></td>
</tr>
<tr>
<td>FR 1.3.3 Sequence/coordinating fault recovery/response</td>
<td></td>
</tr>
<tr>
<td>DP 1.3.3 Fault Handler</td>
<td></td>
</tr>
<tr>
<td>FR 1.3.4 Collect system state</td>
<td></td>
</tr>
<tr>
<td>DP 1.3.4 Status Collector</td>
<td></td>
</tr>
</tbody>
</table>

Figure 30 Key Requirements Mapping to Functional Architecture
Figure 31 shows a representative design rendition of the IASD at the component level. It is not surprising that this diagram resembles the axiomatic component level diagram shown in Figure 6. The primary reason is that each approach uses the same input data (FRs, DPs, and CNs) as a base for the project development.

Figure 31 Aggregation of Leaf Nodes of Decomposition Diagram
Figure 32 shows the UC1 Query data holdings on legacy system collaboration/data flow block diagram. This diagram is useful to understand the step-by-step flow of operations corresponding to operational scenarios. Figures 32 – 34 correspond to UC1, UC2 and UC3 respectfully.
Figure 33 shows the UC2 Transmit data to new sites upon request collaboration/data flow block diagram.
Figure 34 shows the UC3 Monitor system status and data/command flow through adapter collaboration/data flow block diagram.
Figure 35 shows the UC1 Query data holdings on legacy system sequence diagram.

Figure 35 UC1 Sequence Diagram

Figure 36 shows the UC2 Transmit data to new sites upon request sequence diagram.

Figure 36 UC2 Sequence Diagram
Figure 37 shows the UC3 Monitor system status and data/command flow through adapter sequence diagram.

Figure 37 UC3 Sequence Diagram

4.1.4 Physical Architecting

- Objectives
  - Define a physical organization for the system of interest
  - Evaluate work against contextual, operational and logical artifacts
- Tasks
  1. Identify physical components that could contribute to your solution
  2. Define a physical approach for the solution
  3. Create a static view of the physical system (block diagram, concrete class diagram)
  4. Trade of approaches to quality attributes
4.1.4.1 Identify Physical Components that Contribute to Solution

- Interface adaptor
  - Servers (Management, Event Logging and Marshaller functional/physical components)
  - Computer terminal for GUI I/F for Operator
  - Legacy and Subsystem I/F Switches & Communications Gear

- External
  - Assume communications backbone is customer furnished equipment
  - Assume all Subsystem components are customer furnished

4.1.4.2 Define Physical Approach to Solution

Figure 38 shows the geographic view of the physical system in order to gain an understanding of how the IASD is deployed.

Figure 38 Geographic Static View of Physical System
4.1.4.3 Create a Static View of Physical System

Figure 39 shows a conceptual static view of the physical system. This view gives the design team an understanding of the physical components within the system.

4.1.4.4 Architecture Concluding Comments

Utilizing architectural design techniques and practical examples described in Chapter IV, a reader may apply these techniques to problems of a similar nature. The primary idea is to give the reader an idea of where to start when trying to interface a legacy system to a more modern system.
CHAPTER V
PROJECT PLANNING

3.1 Planning Overview

This chapter develops an approach to an Integrated Master Plan (IMP) and Integrated Master Schedule (IMS) for an Interface Adapter Software Design (IASD) Project example.

3.2 Engineering Process

The following section describes a process for developing an engineering implementation and execution process for the IASD project. To simplify the engineering process, many assumptions are made in order to bridge the gap between the outputs of the axiomatic design process to developing a full IASD program.

The following section describes a system engineering planning approach to building the IASD. The design process is presented as a collection of artifacts with a description of the process surrounding them. This process has seven major steps:

1. Business Strategy/Planning
2. Project Planning and Management
3. Requirements and Architecture Development
4. Product Design and Development
5. System Integration, Verification and Validation
6. Production and Deployment
7. Operations and Support

The seven steps illustrate a simplified product development lifecycle and are intended to provide a framework for developing a project plan and integrated master schedule.
3.2.1 Business Strategy/Planning

Business Strategy Planning involves the identification of a business opportunity and developing a strategy to pursue it. In the case of the IASD program, the customers asked to take a recent Axiomatic Design and put together a program execution plan to build an IASD. The artifacts include an Integrated Master Plan, Integrated Master Schedule and a list of Requirements with defined verification methods.

In reality, a program of this size could easily have five hundred to a thousand requirements, but for the purposes of learning the process, the author has scaled down the example to only those technical requirements developed during the Component Oriented Axiomatic Design project. The IASD requirements are broken out into five high level areas as shown in Figure 40.

![Figure 40 IASD Top Level Requirements Layout](image)

In general, a significant amount of time, effort and collaboration goes into developing an initial IMP and IMS for a project similar to the IASD project.

3.2.2 Project Planning and Management

Project Planning and Management is the stage where project requirements are refined and negotiated with the customer. It is important to determine how to meet customer needs within existing budget and schedule constraints. At this point during the planning process, and IMP and IMS have been developed and many key management activities are in full swing. The objective is that when a customer
proposal is completed, submitted and signed by the customer, preparation for program execution is started.

The IMP and IMS are considered living documents during the life of the project. Documents should be updated during the program execution process. Any customer driven requests that come after contract signing should be evaluated against the baseline cost and schedule. If changes are warranted, document updates should be made to reflect changes.

3.2.3 Requirements and Architecture Development

Requirements and Architecture Development are where the IASD system & product functional and physical architecting occurs. Further requirements decomposition and development are completed in this phase for preparation of the System Requirement Reviews (SRRs), Preliminary Design Reviews (PDRs) and Critical Design Reviews (CDRs). At this point in the program, the technical baseline should be established in preparation for product implementation.

3.2.4 Product Design and Development

Product Design and Development is the primary component design and software build stage of the IASD program. By now, the requirements and architecture are mature and the development teams are in full swing. This is also when the System Integrations Verification & Validation (IV&V) are preparing procedures, analysis software and test equipment.

3.2.5 System Integration, Verification and Validation

System Integration, Verification & Validation are accomplished after the design and development activities are complete. For the IASD project, it is assumed that there are two distinct software builds. Build 0 has approximately 70% of the capabilities/requirements completed and is designed to provide an initial release for phased risk reduction while providing IV&V with an initial release to develop test procedures and validate requirements. The remaining requirements are assumed to be delivered in Build 1. Build 1 is the first release to the customer site.
3.2.6 Production and Deployment

Production and Deployment is the primary stage for preparing and deploying the IASD product to the customer. Final Factory preparations are conducted and final documentation is completed for delivery to customer site. The IASD deployment includes a breakdown of the hardware at the Factory site, shipment to customer sites and re-installation at the customer site. Stage 6 is also the final transition from the Factory to the customer location.

3.2.7 Operations and Support

Operations & Support include the receipt of the IASD system from the Factory site, where training programs are developed for the new IASD operations and Support staff at the customer site. This is where the Integrated Logistic group takes control of the system as it relates to the upkeep of Hardware (sparing, etc.).

3.2.8 IMS/IMP Lesson Learned

During the planning stages of a project, it is common for a team to proceed into IMS development without completely following the IMP/IMS development process. When this happens, there is typically twice the number of IMP and IMS iterations. This extra effort may be avoided by completing the IMP (specifically, the Event Dictionary) before putting the IMS together.

3.3 Integrated Master Plan (IMP) Development

An Integrated Master Plan is generally split into three sections; Descriptive, Product, and Process Sections. The Descriptive Section describes in detail the overview of the program and its organization. The Product Section, describes the detailed product information related to the program. The Process section describes the key processes used to achieve program success.

3.3.1 IMP Descriptive Section

The IMP Descriptive Section may include the following subsections:
• Purpose and Identification
• Content and Organization
• Program Overview
• Purpose and Scope
• Background
• Development Approach
• Performance Measures
• Assumptions and Constraints
• Program Organization
• Work Breakdown Structure (WBS)
• Risk Management
• Earned Value Management System (EVMS)
• Contract Compliance and Subcontract Management
• Event Dictionary
• Terms and Conditions

3.3.2 IMP Product Section

The IMP Product Section may include the following subsections:

• Event/Accomplishment/Criteria Table
• IMP and IMS Maintenance Approach

3.3.3 IMP Process Section

The IMP Process Section may include the following subsections:

• Key Program Processes
3.4 Integrated Master Schedule (IMS) Development

In preparation for developing the IMS, it is very important to complete (at least final draft form) the IMP Product Section “Event/Accomplishment/Criteria Table. The IASD project IMP is used to develop the corresponding IMS. Table 9 shows an example of the IMP that is used to translate data and construct the corresponding IMS shown in Figure 41.

Table 9 lists the events identified for the IASD project. These events represent an integration of key milestones and deliveries into assessment points that support review of program accomplishments and evaluation of product and project maturity. There should always be a one-to-one mapping from the IMP to the IMS. Both the IMP and IMS are living documents throughout the lifecycle of a program and should be updated on a regular basis.

<table>
<thead>
<tr>
<th>Event</th>
<th>Accomplishment</th>
<th>Criteria</th>
<th>Description</th>
<th>IMS Finish</th>
<th>IMS UID</th>
<th>Assoc. CDRL</th>
<th>Resp. IPT</th>
<th>WBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01</td>
<td>A0000</td>
<td>Initial Requirements Review (IRR) Completed</td>
<td>Thu 6/7/07</td>
<td>81</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0100</td>
<td>SE – Systems Requirements Defined</td>
<td>Wed 2/21/07</td>
<td>82</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0101</td>
<td>Sys Req – Customer Needs (CNs) defined</td>
<td>Wed 2/7/07</td>
<td>83</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0102</td>
<td>Sys Req - Derived Requirements Defined</td>
<td>Thu 2/15/07</td>
<td>84</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0103</td>
<td>Sys Req – CONOPS Defined</td>
<td>Wed 2/21/07</td>
<td>85</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A02</td>
<td>A0200</td>
<td>SE – System Architecture Developed</td>
<td>Tue 3/13/07</td>
<td>86</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0201</td>
<td>Sys Arch – System Functions Defined</td>
<td>Mon 2/20/07</td>
<td>87</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0202</td>
<td>Sys Arch – System Functional Analysis Performed</td>
<td>Mon 3/5/07</td>
<td>88</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0203</td>
<td>Sys Arch – System Physical Analysis Performed</td>
<td>Wed 2/28/07</td>
<td>89</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A03</td>
<td>A0300</td>
<td>SE – Product Requirements Defined</td>
<td>Fri 4/6/07</td>
<td>91</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0301</td>
<td>Prod – Product Scenarios Generated</td>
<td>Tue 3/20/07</td>
<td>92</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0302</td>
<td>Prod – Product Requirements Defined</td>
<td>Mon 3/26/07</td>
<td>93</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0303</td>
<td>Prod – Product interface Requirements Defined</td>
<td>Mon 4/2/07</td>
<td>94</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0304</td>
<td>Prod – Product Verification and Validation Requirements Defined</td>
<td>Fri 4/6/07</td>
<td>95</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0400</td>
<td>SE – Product Architecture Developed</td>
<td>Tue 5/15/07</td>
<td>96</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A04</td>
<td>A0401</td>
<td>Prod Arch – Product Functions Defined</td>
<td>Tue 4/10/07</td>
<td>97</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A0402</td>
<td>Prod Arch – Product Functional Analysis</td>
<td>Wed 98</td>
<td>SE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Accomp</td>
<td>Criteria</td>
<td>Description</td>
<td>IMS Finish</td>
<td>IMS UID</td>
<td>Assoc. CDRL</td>
<td>Resp. IPT</td>
<td>WBS</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>---------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Performed</td>
<td>4/18/07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0403</td>
<td>Prod Arch – Product Physical Analysis</td>
<td>Perfomed</td>
<td>Thu 4/26/07</td>
<td></td>
<td>99</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0404</td>
<td>Prod Arch – Product Architecture Defined</td>
<td></td>
<td>Fri 5/4/07</td>
<td></td>
<td>100</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0405</td>
<td>Prod Arch – Product Architecture Review Conducted</td>
<td>Conducted</td>
<td>Mon 5/14/07</td>
<td></td>
<td>101</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0406</td>
<td>Prod Arch – Independent System Functional Review (ISFR) – (IR-6) Conducted</td>
<td>Conducted</td>
<td>Tue 5/15/07</td>
<td></td>
<td>102</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0500</td>
<td>SE – Security Requirements Analysis Completed</td>
<td></td>
<td>Thu 3/29/07</td>
<td></td>
<td>103</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0501</td>
<td>SEC – SSAA Developed</td>
<td></td>
<td>Tue 3/20/07</td>
<td></td>
<td>104</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0502</td>
<td>SEC – SSAA Reviewed/Released</td>
<td></td>
<td>Fri 3/23/07</td>
<td></td>
<td>105</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0503</td>
<td>SEC – SRTM Developed</td>
<td></td>
<td>Fri 3/16/07</td>
<td></td>
<td>106</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0504</td>
<td>SEC – SRTM Reviewed/Released</td>
<td></td>
<td>Tue 3/20/07</td>
<td></td>
<td>107</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0505</td>
<td>SEC – SSAA/SRTM Coordination Completed</td>
<td>Completed</td>
<td>Tue 3/27/07</td>
<td></td>
<td>108</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0506</td>
<td>SEC – Review Preparation Completed</td>
<td></td>
<td>Thu 3/29/07</td>
<td></td>
<td>109</td>
<td>SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0600</td>
<td>IRR – Meeting Conducted</td>
<td></td>
<td>Thu 6/7/07</td>
<td></td>
<td>110</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0601</td>
<td>IRR – System Issues and Risks Defined</td>
<td></td>
<td>Tue 5/29/07</td>
<td></td>
<td>111</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0602</td>
<td>IRR – CDRLs Delivered</td>
<td></td>
<td>Thu 5/31/07</td>
<td></td>
<td>112</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0603</td>
<td>IRR – IRR Kickoff meeting conducted with review team</td>
<td>Conducted</td>
<td>Fri 6/1/07</td>
<td></td>
<td>113</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0604</td>
<td>IRR – Meeting preparation completed</td>
<td></td>
<td>Thu 6/7/07</td>
<td></td>
<td>114</td>
<td>SE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 41 shows the IMS corresponding to Table 9. There is a one for one match in the IMS. The task durations are determined by each responsible team’s basis of estimate and the logical predecessor/successor relationships by the collaboration between the teams. It is important to remember that the IMP and IMS are only mocked up and have no relation to any project or program within Raytheon. They are simply used to illustrate the process setting up a project using these techniques.
Figure 41 IASD Integrated Master Schedule (IMS)

Although Figure 41 shows only an excerpt of the entire 262 line IMS, Figure 42 shows the entire “rolled up” high level project level IMS. The high level IMS shows two phases, Phase 0 and Phase 1.
This was done to illustrate how someone might set up an entire multi-phase project. Figure 41 and Table 9 relate to Phase IASD Phase 1 only.

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity #</th>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>291</td>
<td>IASD Program Phase 0</td>
<td>58 days</td>
<td>Mon 11/13/06</td>
<td>Wed 1/31/07</td>
</tr>
<tr>
<td>2</td>
<td>270</td>
<td>Axiomatic Design TTU Course at Raytheon</td>
<td>20 days</td>
<td>Mon 11/13/06</td>
<td>Sat 12/9/06</td>
</tr>
<tr>
<td>3</td>
<td>233</td>
<td>Estimated Axiomatic Design Development</td>
<td>38 days</td>
<td>Mon 12/11/06</td>
<td>Wed 1/31/07</td>
</tr>
<tr>
<td>4</td>
<td>53</td>
<td>IASD Phase 1</td>
<td>703 days</td>
<td>Thu 2/1/07</td>
<td>Mon 10/12/09</td>
</tr>
<tr>
<td>5</td>
<td>54</td>
<td>Initial Requirements Review (IRR)</td>
<td>Completed</td>
<td>Thu 2/1/07</td>
<td>Thu 6/7/07</td>
</tr>
<tr>
<td>6</td>
<td>77</td>
<td>Integrated Baseline Review (IBR)</td>
<td>Completed</td>
<td>Wed 5/30/07</td>
<td>Tue 9/11/07</td>
</tr>
<tr>
<td>7</td>
<td>109</td>
<td>PDR Meeting Conducted</td>
<td>96 days</td>
<td>Wed 9/12/07</td>
<td>Fri 12/28/07</td>
</tr>
<tr>
<td>8</td>
<td>131</td>
<td>CDR – System Critical Design Complete</td>
<td>48 days</td>
<td>Mon 12/31/07</td>
<td>Wed 3/5/08</td>
</tr>
<tr>
<td>9</td>
<td>155</td>
<td>Software Development Complete</td>
<td>137 days</td>
<td>Mon 3/6/08</td>
<td>Fri 9/12/08</td>
</tr>
<tr>
<td>10</td>
<td>159</td>
<td>Factory Acceptance Test (FAT)</td>
<td>Test Readiness Review (TRR) Completed</td>
<td>Thu 6/2/08</td>
<td>Fri 1/23/09</td>
</tr>
<tr>
<td>11</td>
<td>210</td>
<td>Pre-Ship Review (PSR)</td>
<td>Complete</td>
<td>Thu 4/2/09</td>
<td>Wed 7/8/09</td>
</tr>
<tr>
<td>12</td>
<td>234</td>
<td>Site Acceptance Test (SAT)</td>
<td>Test Readiness Review (TRR) Complete</td>
<td>Thu 7/9/09</td>
<td>Wed 9/16/09</td>
</tr>
</tbody>
</table>

**Figure 42 IASD High Level Integrated Master Schedule (IMS)**

Good project planning as illustrated above is key to project success. It is important to remember that the project IMP and IMS are living documents and should be kept as current as possible. As changes are incorporated into these documents, always consider briefing the customer and project team at the earliest possible convenience.
CHAPTER VI
SUMMARY AND CONCLUSIONS

This report successfully describes how axiomatic design, architectural and project planning techniques are leveraged to systematically develop and plan the Interface Adapter Software Design project. A set of customer needs, Functional Requirements, Design Parameters and Constraints applicable to this type of project were developed and used to the fullest extent possible.

The Component Oriented Axiomatic Design approach utilized specific processes to develop a system design to the component level while showing how a missing component may be discovered and incorporated early into the design process to minimize overall cost and schedule impacts. Secondly, an architectural design approach combining contextual, operational, logical and physical data was used to provide a base architecture. These detailed processes allow a smooth transition to system hardware and software development. Thirdly, the IASD project Integrated Master Plan and Integrated Master Schedule were developed so that the reader may gain an understanding of how to apply project planning in a systematic manner.

The Raytheon – Texas Tech University System Engineering program is an excellent choice when combining University led education with an industry specific flavor. The coursework in this program was directly applicable for a project such as the one described in this paper.
REFERENCES


2. [Tate, 2006] Fundamentals of Transdisciplinary Design and Process, November 9-11, 2006, Derrick Tate, Ph.D.


4. [Togay, Dogru, Tanik, Tate] Systematic Component-Oriented Development with Axiomatic Design, Togay, Dogru, Tanik, Tate


