

STRATEGIC PLANNING AND
TACTICAL OPERATIONS FRAMEWORK
DESIGN AND IMPLEMENTATION

A Professional Project Paper submitted

By

Kenneth R. Houghtling

To

Regis University

In partial fulfillment of
the requirement for the
degree of

MASTER OF SCIENCE
in
COMPUTER INFORMATION SYSTEMS

This thesis has been
accepted for the faculty of
School for Professional Studies by:

Mike Prasad
Advisor

Abstract

For the past 30 years, Information Technologies costs have outpaced the return on investment.

There are several factors that are attributable to runaway Information Technologies costs.

Perhaps the costliest among these factors is the lack of reusable assets within the computing infrastructure inventory. A prime example of this is the pervasive model of every new project funding and implementing a completely self-encapsulated operating environment. Each new project is required to address the provisioning hardware platforms and software services. This paradigm tends to be more prevalent in large and medium sized organizations. Often this scenario requires the unnecessary and redundant implementation of common services and hardware platforms to suit the needs of an individual project or application.

The goal of this project is to facilitate that collective source of knowledge by providing a standardized framework for documenting the existing and projected computing infrastructure and software services within an organization. This framework will include processes to manage the associated data. Additionally, the project will facilitate the development of a prototype application to manage the data within the scope of the framework. The resulting deliverables will facilitate a knowledge base making this information available to the strategic and tactical software life cycle community. In turn, this community can realize opportunities for standardization and reuse, and provide a firm target for delivery. Additionally, this information will empower the various teams involved with architecture, design, testing, application implementation and production to make better decisions across the “Software Development Life Cycle”.



March 1, 2006

Property of:

Kenneth R. Houghtling

Do Not Replicate without Author's Permission

CONTENTS

Abstract	i
Table of Figures	viii
List of Tables	x
CHAPTER I Introduction	1
Problem Statement.....	1
Review of Existing Situation.....	1
Endless Trail of One-off Solutions	3
Lack of IT communication with its own organization.....	4
Goals for This Project.....	9
Strategic Planning Tactical Operation Framework	10
Strategic View	11
Tactical View.....	11
Operational View.....	11
Change Management for the Framework.....	11
Current Strategic Planning, Tactical and Operation Information Flow.....	12
Desired Strategic Planning Tactical Information Flow	15
SPTOF Prototype Application	18
Issues & Barriers to Success.....	18
CHAPTER II Scope of Project	20
Review of Research.....	20
Review of existing solutions	20
Research Methodology.....	21
What Is Known & Unknown About This Topic?.....	22

Project’s Contribution to the Industry	22
Project Methodology	23
CHAPTER III Framework Definition and Design	27
Alignment with Industry Standards.....	27
High-Level Definition	28
Strategic Framework Focus.....	28
Architectural Layers	29
Categories	30
Services.....	30
Operational Framework Focus.....	33
Domain Specification	33
Application Specification	34
Domain Application Deployment.....	36
Strategic Projection	37
Tactical Portion	38
CHAPTER IV Change Control Process Definition and Design.....	41
Change Management Process Definition	42
Stakeholders for the process.....	43
Define Roles with the process scope.....	43
Strategic Planning Tactical Operations Framework Change Control Process Steps	44
Process Design Overview Diagram based on Process documentation.....	46
CHAPTER V Prototype Application Definition & Design	47
Supporting UML Artifacts	47
Security Functions Use Cases	48
Architect Function Use Cases	51

Architectural Layer Management	52
Category Information Management Use Cases	54
Service Information Management Use Cases.....	57
Architecture Domain Projections Use Cases.....	60
Developer Functions Use Cases.....	62
Developer Request.....	63
Locally Developed Application Management.....	63
Operations Functions Use Cases.....	66
Domain Management.....	67
Operations Request.....	69
Commercial Application Management.....	69
Reporting Functions Use Cases	72
Design Supporting Framework Database.....	75
Strategic Data Scheme	75
Operational Data Scheme.....	75
External Supporting Data Scheme	76
SPTOF DATABASE TABLE SPECIFCATIONS	78
Functional Design for Prototype Application.....	86
High-Level Programmatic Flow.....	89
Hosting Environment Description.....	91
High-Level Design Diagrams.....	93
Package Description: edu.regis.sptof.beans	94
Package Description: edu.regis.sptof.dao	95
Package Description: edu.regis.sptof.datastructs.....	95
Package Description: edu.regis.sptof.event	95

Package Description: edu.regis.sptof.security	96
Package Description: edu.regis.sptof.servlet	96
Package Description: edu.regis.utils	96
Graphical User Interface Design Specification	97
Web Site Flow.....	97
Architects’ Web Flow	98
Developer’s Web Flow	99
Operations Web Flow	100
SPTOF Web Site Wireframe Diagrams and Narratives.....	101
Login Page Wireframe.....	102
Architecture Main Page Wireframe.....	103
Developers Main Page Wireframe.....	104
Operations Main Page Wireframe	104
Asset Management Main Page Wireframe	105
New Asset Wireframe.....	107
Modify Asset Wireframe	108
Domain Projection Wireframe.....	109
Gap Analysis Wireframe	111
Request Form Wireframe	113
CHAPTER VI Review of the Deliverables	114
CHAPTER VII Historical Project Information.....	117
Project Initialization Incentives	117
Project Measures of Success	118
Project Management Details.....	119
Project Variables and Their Impact	125

Success and Failure Disclosure	126
Project Summary	127
CHAPTER VIII Review of the Deliverables.....	128
What was the learned from the project?.....	128
What should have been done differently?	129
Did the project meet expectations?.....	130
Project Next Steps beyond Its Current Scope.....	130
Conclusions and Recommendations.....	132
Summary of Project.....	132
References.....	135

Table of Figures

Figure 1: Change, Cause and Effect - Diagram	8
Figure 2: Current Information Flow - Diagram	14
Figure 3: Desired Flow with the Addition of SPTOF Framework - Diagram	17
Figure 4: Strategic Frame Layout Diagram	29
Figure 5: N-Tier Stratification Diagram	30
Figure 6: Strategic Framework Inter-Relationships Diagram.....	32
Figure 7: Domain Specification Diagram	34
Figure 8: Application Specification Diagram	35
Figure 9: Domain and Services Interrelationship Diagram	37
Figure 10: Strategic Projection Diagram	38
Figure 11: Tactical Perspective Diagram.....	39
Figure 12: Change Control Process Overview (Diagram).....	46
Figure 13: Security Function (Use Case Diagram).....	48
Figure 14: Architecture Functions (Use Case Diagram).....	51
Figure 15: Developers Functions (Use Case Diagram)	62
Figure 16: Operation Functions (Use Case Diagram).....	66
Figure 17: Reporting Function (Use Case Diagram)	72
Figure 18: SPTOF (Entity Relationship Diagram)	77
Figure 19: High-level Functional Flow Diagram.....	90
Figure 20: Hosting Environment Diagram	91
Figure 21: High-Level Class Diagram	94
Figure 22: SPTOF Application Web Site Topology Diagram.....	98
Figure 23: Architects' Web Site Flow Diagram	99

Figure 24: Developers' Web Site Flow Diagram	100
Figure 25: Operations Web Site Flow Diagram	101
Figure 26: Login Page Wireframe	103
Figure 27: Architects' Main Page Wireframe Diagram.....	103
Figure 28: Developers' Main Page Wireframe Diagram.....	104
Figure 29: Operational Main Page Wireframe Diagram.....	105
Figure 30: Asset Management Main Page Wireframe Diagram.....	106
Figure 31: New Asset Wireframe Diagram	107
Figure 32: Asset Modification Wireframe Diagram.....	108
Figure 33: Domain Projection Wireframe Diagram	110
Figure 34: Gap Analysis Wireframe Diagram	112
Figure 35: Request Form Wireframe Diagram	113
Figure 36: Gant Chart Legend	119
Figure 37: Project Plan Phase I, II, & III Diagram.....	120
Figure 38: Project Plan Phase IV Diagram	121
Figure 39: Project Plan Phase V Diagram	122
Figure 40 : Project Plan Phase VI Diagram	123
Figure 41: Project Plan Phase VII & VIII Diagram.....	124

List of Tables

Table 1: Schedule of Phases for the Project.....	24
Table 2: Role Definition	43
Table 3: Change Inputs, Outputs and Process Steps Narrative.....	44
Table 4: SPTOF Database Tables Specifications	78
Table 5: User Roles Definitions.....	86
Table 6: High-Level Environment Table.....	92
Table 7: Prototyping Platform Specification	93
Table 8: Schedule of Project Phase Plan and Deliverables.....	114
Table 9: Schedule of Measures of Success	118
Table 10: Application Results Summary	123

CHAPTER I

Introduction

Problem Statement

For the past 30 years, Information Technologies costs have outpaced the return on investment. Several factors are attributable to runaway Information Technologies costs. Perhaps the costliest among these factors is the lack of reusable assets within the computing infrastructure inventory. A prime example of this is the pervasive model of every new project funding and implementation of a completely self-encapsulated operating environment. Each new project is required to address the provisioning hardware platforms and software services. This paradigm tends to be more prevalent in large and medium-sized organizations. Often this scenario requires the unnecessary and redundant implementation of one-off hardware platforms and supporting services that suit the individual needs of a project or application.

Review of Existing Situation

The present day Information Technology organization is in constant pursuit of accomplishing more with fewer resources. Some of the advances in technology have enabled this goal. Computer technologies have become relatively inexpensive. In the past ten years, business and Information Technology professionals have begun to recognize that segregated functions within the software development life cycle encumber business processes and information flow across the enterprise and vertically.

In *The Biology of Business*, Andy Clark states, "Markets, companies and various forms of business organizations can all be usefully viewed through the lens of complex adaptive systems."

He then says that a market or company is self-organizing where "crucial interactions are not controlled or orchestrated by an overseeing executive, a detailed program or any other source of strict hierarchy" (Clippinger & Jossey-Bass, 1999, p. 47).

Processing and storage capabilities have increased 1000% in the past thirty-year period. Yet, the price for the equivalent hardware has dropped by 500%. User interfaces have evolved that bring practical computing capabilities to a large segment of the general commercial and casual computer markets. All of these advancements are truly a boon to productivity and have been well received, but at what cost? Has the investment into new technologies and the endless parade of processes and methodologies provided adequate return on investment? Jack Welch, former CEO of General Electric, asserted in his book, "Jack Welch Speaks: Wisdom from the World's Greatest Business Leader", that, "Information Technologies is the greatest disappointment in the past thirty years." (Lowe, April 2001, p. 65)

During the decade of the 1990's, U.S. firms invested over \$2.4 trillion on Information Technology assets, including computer hardware, computer software and telecommunications equipment. ("Measuring Information Technology and Productivity in the New Economy", 2002) These three assets accounted for more than 40% of private fixed investment in equipment and software in 2000. Businesses have been asking hard questions about the Return on Investment for this large cash outlay. This extremely telling statement supports the notion that trillions of dollars have been pumped into information technologies in the thirty-year span Mr. Welch refers to, and many organizations would be hard pressed to associate those expenditures with investments.

Endless Trail of One-off Solutions

Why do businesses continue to pump valuable capital into information technologies when there is no tangible return-on-investment demonstrated? Perhaps the most unfortunate and avoidable truth about Information Technology is its associated expense. Practical wisdom points to the trimming of costs wherever possible. One first area to address is the one-off solution issue that exists.

The term one-off can be defined as the, “I could not find one in the corporate asset repository, so I built one to keep the project moving”, paradigm. This definition is more often the rule and not the exception. Developers would leverage common service assets if they knew of their existence and specification. The lack of reusability in this case stems from the lack of knowledge.

In larger organizations, a vertical view of technologies in narrowly defined deployment domains compounds fuels the lack of communications pertaining to reusable assets. Architecture, development and operations communities focus on familiar technologies and working models. They do not concern themselves with alternative solutions beyond the borders of domains for which they are responsible. This forces organizations to deploy multiple hosting environments to facilitate narrowly focused solutions. This type of internal corporate isolation results in environments that duplicate services or components already provisioned in other environment. Therein lies the impetus for one-off-solutions

Senior management seems genuinely surprised to discover that applications designed in these one-off environments require vast amounts of resources to accommodate change or integration.

Additionally, these solutions often require extensive modification efforts to accommodate changes in the business logic or infrastructure layer of the design. One-off designs usually contain just enough differences in their implementation to require major re-tooling. Consequently, their uniqueness excludes them from benefiting from the work performed on similar projects utilizing the same business logic.

Lack of IT communication with its own organization

The root-cause analysis into one-off solutions reveals extensive implementations of disjoint and isolated solution sets. These are usually directly attributable to the lack of communicated knowledge. Many larger organizations are forced to maintain several implementations of application-hosting environments for just that reason.

Enterprise architecture teams try to bridge the communications gap by publishing strategies that provide guidance and direction. Normally a strategy is published in the form of an “Enterprise Architecture Framework”, which contains its own concepts, components, and methodologies to facilitate an architectural strategy message. (Pedro Sousa, 2005).

The strategic view is a valuable resource in any information technologies organization. It provides a projection of current and future computing environments. In its raw form, this information does not support a value add proposition to any design and development community. The real value-add proposition would be to translate this “Enterprise Architecture Framework” into a form that development and operations teams can comprehend and consume. If the information was accurately defined and translated, the development and operations communities can target future application and project designs towards the projected specification.

Another missing communications component of most enterprise architecture framework publications is the lack of a feedback mechanism. In larger organizations enterprise architecture strategies are released to the design, development, integration and operations organizations without understanding the full ramifications or effect on the information technologies environment at large. Architects rarely receive feedback on their architectural strategies unless there is a major negative impact to the organization. Without a feedback loop, it is impossible for an architecture team to understand the positive and negative impacts of the strategies it produces.

In most organizations, the development team often bears the burden of filling knowledge gaps between the current state of a specific and projected state of an environment. These individuals are usually the only source of institutional knowledge about their assigned environment from a developmental and deployment perspective. Their knowledge is limited to the services and components deployed in their environments. Their focus is on the software tools at their disposal to translate business requirements into software designs. These teams do not possess the horizontal or enterprise view of the architecture team. Consequently, these teams do not factor reusable services into their design, because they are unaware of the potential of reusability of their final product or other products they might leverage in their designs. An even more alarming fact is that these development teams often will duplicate resource services that pre-exist in another environment. Now the enterprise is forced to incur the cost of maintaining several application implementations that provide the same service. Their limited enterprise vision does not facilitate an understanding of the possibilities of reuse of an existing service from an environment outside their own.

Regrettably, the operations team is usually at the same disadvantage as the development teams in terms of lack of information. Operations teams are usually focused on the aspects of production support. They do not focus on the strategic view. For this team, the important issues are uptime and stability. These two concerns place the operations team in direct opposition with the architecture and development teams. New implementation of services and applications that consume or support those services means change. A production support team's worst enemy is change. Downtime and instability are often directly attributable to unmanaged change. Managed change is the most desirable middle ground for an operations team. Change management requires information.

An unfortunate truth is that an operation team is at times its own worst enemy. "The operations team focus is limited to understanding the cause and effect of change within the scope of a specific applications domain" (Clippinger & Jossey-Bass, 1999, p. 47). They are not focused on domains that might consume services outside of the domain they are supporting. The ideal situation would be having a direct communications link between the development communities across the enterprise. The link could provide guidance on the effects of any change. In most cases, reality renders this type of communication impractical in most organizations. There is simply no vehicle to communicate change or impact. The result of this communications gap is running applications rendered useless by one simple change.

For the operations team to be a successful partner in the software development life cycle, it requires information about current state and the projected state in the environment it manages. This information needs to be re-factored in terms of applications and software package specifications.

In practical terms, when an off-the-self package is purchased, an organization is buying some other company's architectural specification. Product like PeopleSoft, Siebel's On Demand, and JBOSS are leveraging other company's architectural challenges (Britton, 2001, p. 89). This is not an acceptable risk, unless the architecture and operations teams choose to ignore this reality.

This type of information allows operations to determine the delta factor involved with the proposed change and allows them to work with the deployment teams to anticipate and mitigate change issues prior to any implemented change.

To understand change strategies and their impact on all the teams involved with the software development life cycle, the reasoning of change must be understood. Change manifests itself as a function of a business reacting to market pressures. The market space a business lives in dictates its direction. Even if a business is out in front of its market competitors, the market is the chief influence on a business direction. As exemplified in the following diagram, business services change in small chunks to afford the businesses rapid adaptation to market pressures.

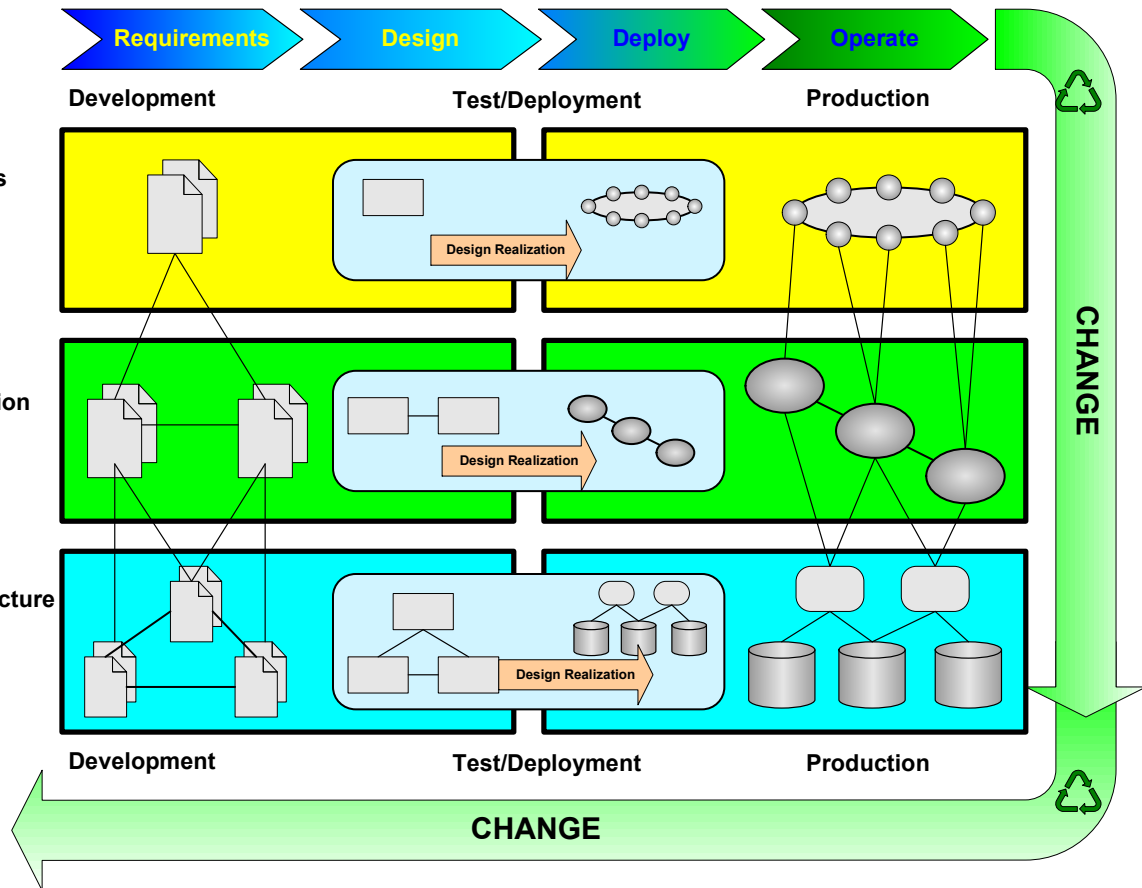


Figure 1: Change, Cause and Effect - Diagram

As the diagram progresses through the layers downward, any change to that layer has more impact due to the nature and the complexity of the layer. The lower level layers support the proceeding layers above. Changes in the lower-level layers can have a ripple effect throughout the entire diagram. Without an understanding of the change impact at these lower levels, drastic unwelcome results could be the result. Issues involving maintenance models are especially important in the infrastructure layer and all the layers above it.

The realities of the types of communications identified as having a gap are difficult to address. Without the proper facility to bridge, the gap organizations will continue to limp along producing applications that do not support the premise of reuse. Inefficiencies of one-off

implementations will drain overtaxed Information Technology budgets. Developers will continue to waste resources on developing services that already exist. They will not venture beyond the boundaries of their limited domain to investigate reuse opportunities. In addition, operations organizations will continue to live in a knowledge vacuum, installing patches and updates without understanding the full impact of doing it.

Goals for This Project

Not all services or platforms are sharable or scalable to meet the needs of an entire enterprise, but the knowledge of the implementation and management of these services is a valuable resource untapped by other internal organizations. The problem lies in the lack of a centralized repository of knowledge that facilitates the sharing of the aforementioned knowledge. From this knowledge source, an informed decision is possible for all the disciplines involved with the “Software Development Life Cycle”.

The goal of this project is to facilitate that collective source of knowledge by providing a standardized framework for documenting the existing and projected computing infrastructure and software services within an organization. This framework will include processes to manage the associated data. Additionally, the project will facilitate the development of a prototype application to manage the data within the scope of the framework. The resulting deliverables will facilitate a knowledge base making this information available to the strategic and tactical software lifecycle community. In turn, this community can realize opportunities for standardization and reuse and provide a firm target for delivery. Additionally, this information will empower the various teams involved with architecture, design, testing, application

implementation and production to make better decisions across the “Software Development Life Cycle”.

Strategic Planning Tactical Operation Framework

The project proposes a framework that defines a structured approach to the problem statement. It will provide logical formulation of data to be stored in a centralized knowledge repository. Data stored in this knowledge repository will include strategic, tactical and operation information. Each of these information areas will support the three disciplines primary involved in the problem statement (architecture, development, operations). The formalization and centralization of this data will facilitate simultaneous horizontal and vertical views of the available information. These views will support strategic and tactical domain information needs.

The implementation of a standard framework supporting the concept of reuse often locks an organization into an architecture that is a replica of that framework. The standards and guidelines used by the framework tend to become ingrained and rigid within the software life cycle community. However, if the organization does not follow a well-accepted development process that is common across projects, it realizes minimal benefit from previous efforts. The pattern of reuse is typically the domain of developers with good modeling skills. This framework proposes to extend this to architecture and operational disciplines within the same organization. (Scott W. Ambler, 2005).

Strategic View

The strategic view is chiefly the realm of the architecture team. It is comprised of information that translates the business vision into technological specifications for the future direction of a business. Within the scope of this effort, the strategic view specifies the components and application services allowing the business to realize its strategic vision.

Tactical View

A tactical view is the spanning view between the strategic and operational views. This is the view that provides the designers and developers with an inventory of the components, application services and the products deployed to facilitate them.

Operational View

The operational view supports the understanding that the products deployed are of a specific domain or of an enterprise standard operating environment. Applications include those internally developed, off the shelf packages and externally consumed services.

Change Management for the Framework

A change management governing body and accompany process will facilitate the integrity of the knowledge repository. The impetus behind this change management element of the project is to address communications and feedback loop inadequacies identified in the problem statement.

The governing body will consist of one or more member representatives from each of the three disciplines defined in the problem statement. Functions of this governing body include:

- Authorizing the publication of strategic information
- Authorizing changes to the tactical and operational data in repository
- Providing guidance on change management processes
- Processing change requests from various disciplines

The process will be comprised of a series of checks and balances. These checks and balances will serve to govern and control over the content and daily functionality this framework will support.

Current Strategic Planning, Tactical and Operation Information Flow

The current Strategic Planning, Tactical, and Operation information flow inadequacies require analysis to facilitate an understanding of the issues that contribute to the communications disconnects. To represent the current flow, a high-level narrative and supporting diagram that depicts the flow will be presented in this section.

For purposes of this discussion, three flow channels will be used in the flow narrative. Each channel will represent each of the three disciplines involved with the software development life cycle:

1. Architecture
2. Design/Development
3. Operations

The Architect information flow focuses on the research and definition of a strategic architecture. In a typical flow, once the strategic is completed, it is communicated to business partners as a strategic vision that aligns with the business vision. The design, development, and operations teams receive the same published strategy, but it is not typically published in a digestible form that these teams can leverage a practical way.

Design and development teams perform their normal function of translating business requirements into application and integrated solutions. During the research of components and services that meet the design requirements, the team is limited to the components and services of which they are aware. Additional components and services obtained in the development process are communicated to the operations organization in a form of an implementation request. This request is often just prior to deployment. If the technology is too divergent from the current target domain, a waiver must be obtained to approve the implementation of a new technology in an existing environment. If the technology is too radical, a one-off domain is implemented to support the new deployment.

The operations discipline tries to realize the architectural projection by implementing hosting domains that adhere to the strategic specification. This is often not possible due to the lack of technical information found in typical strategic architecture publications. The operations organization is tasked with trying to adhere to standards that may or may not support the future architectural specifications. Additionally, standards and stability must be maintained, while accommodating new application implementations. Without advanced warning, newer technologies will be shunned or encapsulated into one-off environments where adverse effects will be limited to that hosting domain.

There is no central source of knowledge in the current flow. Consequently, there is no source of knowledge to consult that would address any of the issues identified in this section. The following diagram depicts the current flow in support of the previous narrative.

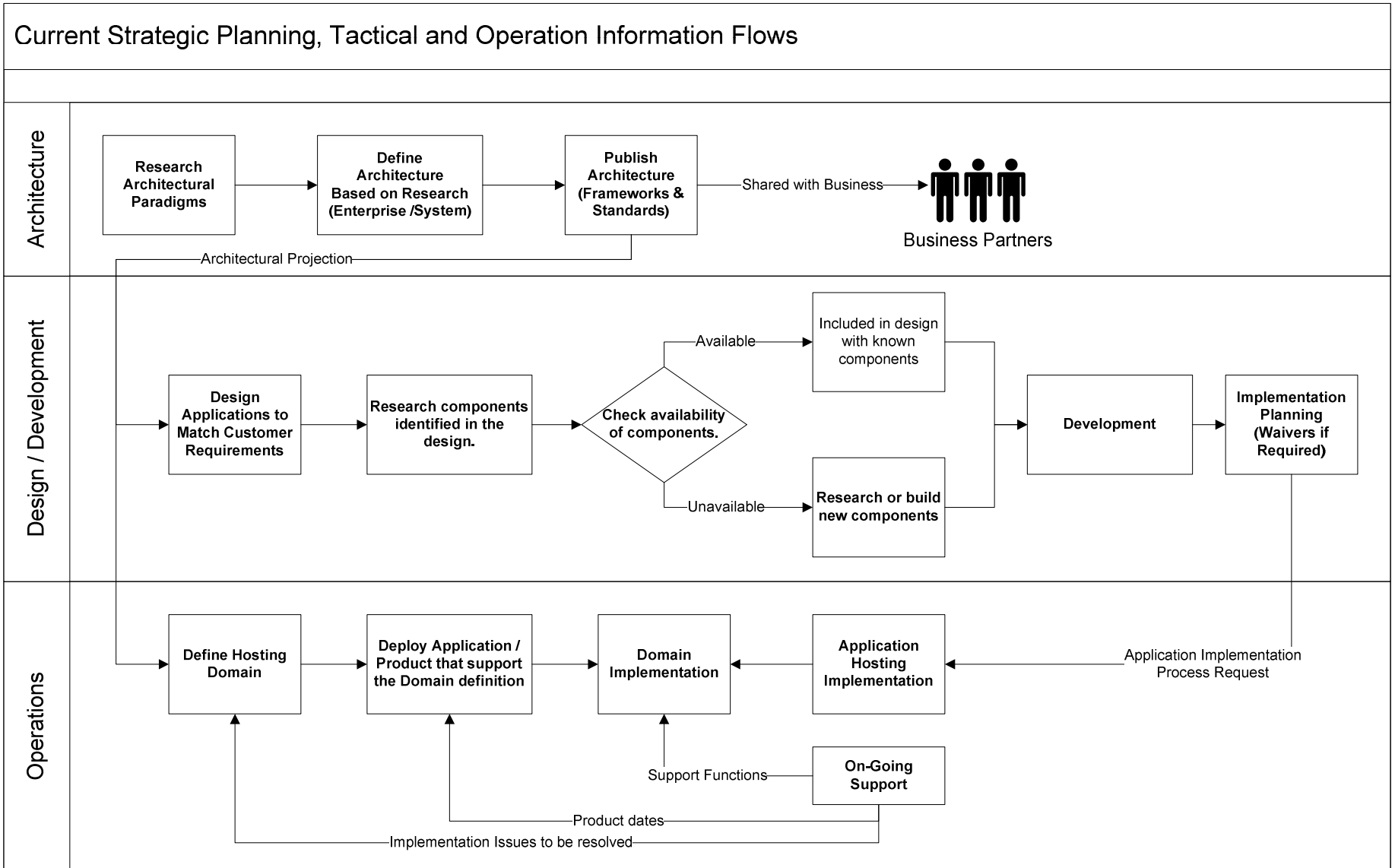


Figure 2: Current Information Flow - Diagram

Desired Strategic Planning Tactical Information Flow

This section provides information about desired flow in the form of a high-level narrative and supporting diagram. The desired flow will describe the Strategic Planning, Tactical, and Operation information flow that leverage the SPTOF framework to address the inadequacies identified in the previous section's analysis.

Within the scope of the architectural information flow, the SPTOF framework application provides processes that support the publication of a strategic architecture and transform that information into a digestible form that can be leveraged in a practical way by the design, development, and operations teams. The new flow promotes accountability for the architectural strategy published. The "Change Management Governance Board" will review strategic service and component specifications prior to their addition to the repository. In this way, the architecture team receives feedback on their strategic protections.

The additional SPTOF flow channel will provide the design and development teams with advance information about services and components from an actual and projected perspective. The perspective includes external views to other domains that could potentially provide a service to a current or future design. The framework will also provide a request and feedback mechanism for the architecture and operations teams. As service gaps are identified in the strategic and tactical specifications for a specific project, the designers can rapidly communicate this issue to the "Change Management Governance Board".

For the operations discipline, the SPTOF framework flow provides a means to communicate current and projected software deployment information. The operations teams can quickly identify cause and effect of upgrades and new applications implementation within the scope of a

hosting environment. As issues of functionality are identified, the operations team can bring these issues to the “Change Management Governance Board” for approval. They can also provide feedback to the architecture and design teams as to the practicality of a design or specification within the scope of a specific hosting domain. This service will potentially eliminate costly last minute redesigns and one-off implementations.

The implementation of the SPTOF framework “Change Management Governance Board” processes will require a more granular functional flow. These processes are documented later in this artifact. The following diagram depicts the desired flow in support of the previous narrative.

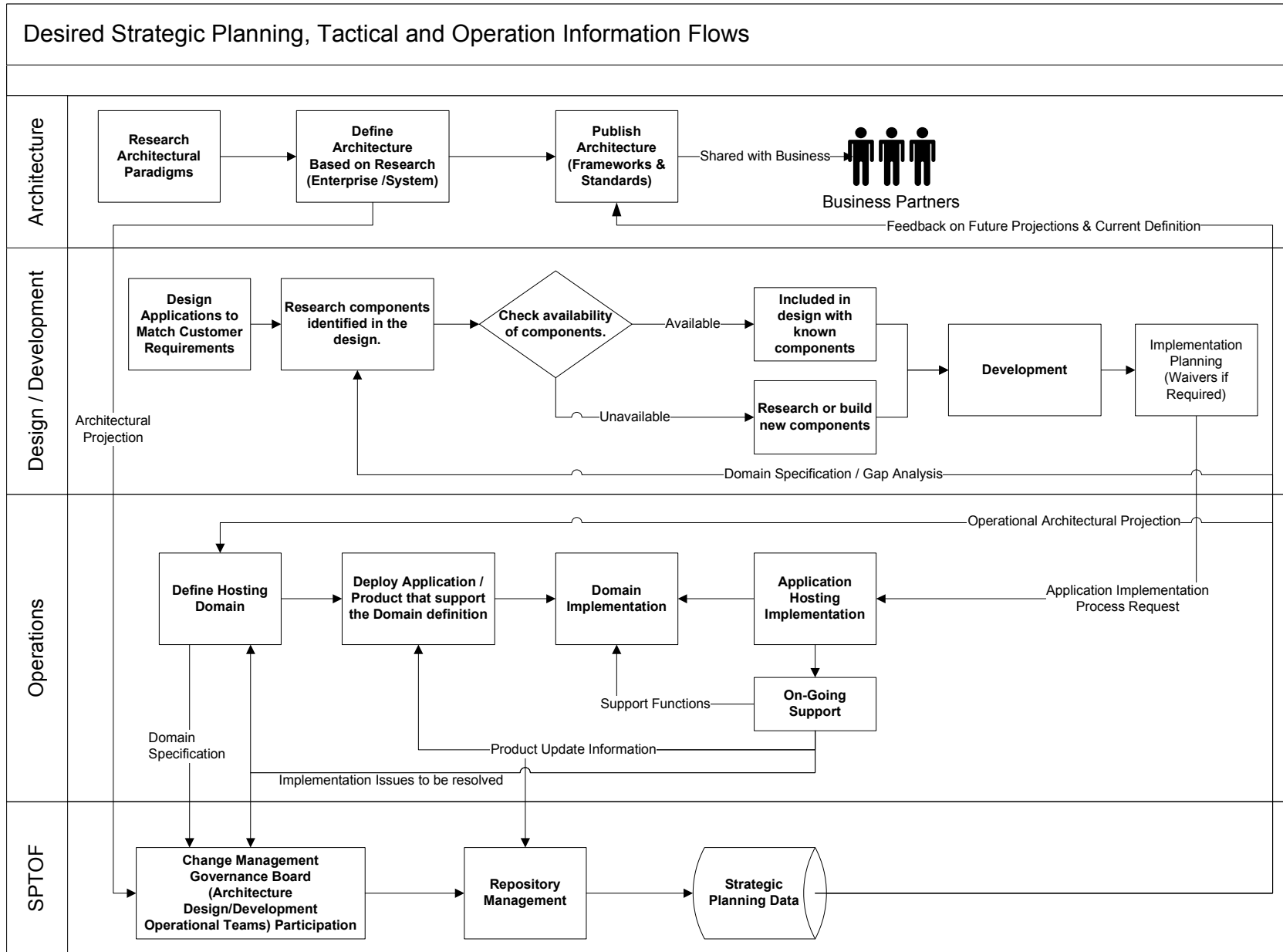


Figure 3: Desired Flow with the Addition of SPTOF Framework - Diagram

SPTOF Prototype Application

To properly support the tactical operations of the SPTOF framework “Change Management Process” and the facilitation of a centralized repository containing the supporting data, an application is required. This application must provide four basic tenets of functionality:

1. Heterogeneous user interface
2. Abstract the user community from direct interface with the data store
3. Secure the data preventing unauthorized access
4. Facilitate generalized reports that provide quick access to critical data

Due to the specific functionality of the SPTOF framework, this application will require localized development. Prior to acceptance of the framework, the customer desires a prototype implementation to evaluate the repository’s value.

Issues & Barriers to Success

Success of this project is dependent on many factors. The majority of these are political. Any implementation of the SPTOF framework will require buy-in from the three disciplines that will be participating in the “Change Management Governance Board” and the day-to-day tactical operations of the repository. If the architecture, design/development and operations organizations cannot agree to equally participate in the effort it will not be successful.

Most of the issues around appropriate participation are attributable to the culture that exists in the information technologies organization. In a typical IT organization these teams have had clearly defined areas of responsibility and control. In essence, software and processes cannot solve these entrenched cultural issues exclusively. Management sponsorship and acceptance of

the motives and methods that comprise the SPTOF framework is required. Management should take a stake in the integrity and accountability of SPTOF operations. In this way, management leads by example.

Management support is required to facilitate the alignment of the framework within their given organization. This includes the participation on the “Change Management Governance Board” until a future point in time when the board has gained acceptance and established the appropriate levels of authority to ensure its success. While the SPTOF framework makes provisions for role-based functionality, it does not dictate specific roles. This is the province of management and should be appropriately addressed by the management team.

The final issue to consider is the continued maintenance of the SPTOF framework and repository. Stagnant data is useless data. If the process of input and update are not followed at regular intervals, the data will become useless. The organization will lose faith in the SPTOF framework informational integrity. The teams involved with the management of SPTOF framework must be allotted time in their daily workflows to perform the necessary SPTOF activities to maintain the data. This includes the removal of inaccurate or outdated information.

CHAPTER II

Scope of Project

This projects' intent is to focus exclusively on the problem statement, which is fairly a broad topic of analysis and investigation. The business client for this project has narrowed the focus to include only the software development life cycle. The obvious implications of narrowing the scope are the limited focus of the analysis of existing solutions and trends in the industry. This section of the project documentation will provide information about existing solutions in this space and the alignment of the framework with industry standards in this area.

Review of Research

Numerous publications relate to topics covered in this project. All the available information tends to stay discipline specific. That is to say, documentation that addresses architecture standards and strategies tend to focus on the design and publication of the architectural strategy. The information presented may even venture into design/development communications, but it does not present the end-to-end view of working with the design and operations to ensure the strategic architectural vision can be achieved in the current or future production hosting environment.

Review of existing solutions

An industry offering in this space is provided by Flashmap Systems products, which offers software and graphical tools for technology portfolio management at the IT and Business Applications level. (Flashmap Systems Inc, 2005). This solution is focused on sharing a common terminology and interface for a global view of the enterprise technology portfolio.

Out of the box, Flashmap Systems solution does not provide a means of identifying multiple domains within a central repository. This makes for difficult research and comparisons.

Additionally, the Flashmap Systems solution focuses on the products with the scope of architectural layers and not at the service and component level the business client desires for this project.

Research Methodology

The research for this project was conducted under three approaches. The first approach was to conduct internal interviews to gather information about the current business problem. These interviews included the members of the three disciplines involved with the software development life cycle:

- Local and Enterprise Architecture teams
- Application / Integration Design And Development Teams
- Operations and production support teams

The problem statement found in this the document is based on summarized information gathered from those interviews. The problem statement was approved by all the stakeholders as correctly representing the issues requiring resolution within the scope of this project.

The second approach used included research into industry offerings that supported a solution set to the problem statement. This research included analysis from books, industry journals, industry periodicals, and whitepapers relating to the topic. Research resources chosen for this approach were based on the domain of each of three disciplines involved. Each of the individual domains approaches the solution to this problem differently. The best of breed for each segment of the research is used to develop a solution to address the problem statement.

Finally, the third approach includes gathering Use Case functional specifications from managers and practitioners of each of the three disciplines involved with the software development life cycle. This Use Case analysis includes functionality and data specifications required by each group. This information will be presented in the application, “Prototype Application Define & Design” section of this document.

What Is Known & Unknown About This Topic?

The horizontal view of the multiple discipline focus does not facilitate or yield much in the way of specific information relating to a cross discipline view within the framework. Each discipline has many resources of information pertaining to the specific methodologies and motivating processes that support individual focuses of that particular discipline within the scope of the software development life cycle.

What is lacking is information on how all the disciplines involved in the problem space can work together to facilitate a cross discipline solution to the problem. The lack of information on this topic stems from the compartmentalization of IT disciplines within the scope of the software development life cycle. There is some overlap of job function in the three disciplines involved, but that is limited to institutional knowledge. No readily available industry framework spans all three disciplines in a way that this project proposes.

Project’s Contribution to the Industry

If this project is properly designed and implemented, this strong framework definition could be leveraged by any organization. The supporting repository and accompanying application will be designed to support a self-determining strategic, tactical and operational model.

The real benefit that can be realized from the SPTOF framework is the standardization of a cross discipline view with a strategic, tactical and operation model. From the model, architects can gain a sense of which strategies are working in their enterprise and which are not. They take into account new and existing services that can be designed into business solutions.

Design and development teams can realize reuse opportunities at various levels of the enterprise. No longer will this team's vision be limited to a specific domain. Cross-platform sharing of services will eventually reduce the number of one-off solutions. It will also allow this team to focus on the business' needs and not on reinventing the wheel.

Operations teams will find it less difficult to raise issues with the other disciplines. The common view of the enterprise afforded to the operations teams will allow them to determine the ramifications of changes to be made to a specific domain. Additionally, the operations team is afforded a view the architectural projection of strategy. This will provide them advanced notice of changes projected for the production environment. Finally, as the number of one-off environments is reduced, the number of environments will also be reduced. This allows the operations team to be more proactive, rather than a reactive team.

Project Methodology

The project will be broken into eight phases. Each phase will support the ultimate goal of supplying a working prototype to the business for evaluation. Each of the phases will contain logically grouped tasks that support completion of that particular phase. The following table provides an inventory of the phase and the associated high-level tasks:

Table 1: Schedule of Phases for the Project

Phase	Description	Work Tasks
I	Analysis	<ul style="list-style-type: none"> • Gather information to define problem statement • Develop problem statement • Obtain approval from stakeholder for problem statement
II	Framework Definition	<ul style="list-style-type: none"> • Define framework to address problem statement • Document framework layout and business function • Document function of framework within the scope of the enterprise
III	Change Control Process Definition, Design	<ul style="list-style-type: none"> • Define purpose of change control process • Define roles within the scope of the process • Define the process flow for the change control process
IV	Application Definition and Design	<ul style="list-style-type: none"> • Collect Use Case data from client population • Document Use Cases in text and UML format including Use Case Diagrams • Define High-Level program flows • Define repository supporting data structures • Develop Entity Relationship Diagram defining the database structure • Define User interfaces based on Use Case information • Define High-Level reporting structures • Generate programmatic flow diagram

Phase	Description	Work Tasks
V	Application Construction	<ul style="list-style-type: none"> • Generate database scheme based on ERD • Generate data layer objects based on table structures defined in the database • Generate test cases for data layer objects • Generate business logic objects based on Use Case requirements • Generate test cases for business logic objects • Generate report logic objects based on Use Case requirements • Generate test cases for report logic objects • Generate User Interfaces based on Use Case requirements • Generate test cases for User Interfaces • Generate release management scripts and documentation
VI	Application Test	<ul style="list-style-type: none"> • Unit level tests on the following: <ul style="list-style-type: none"> ○ Database ○ Data Layer Access Objects ○ Business Logic Objects ○ User Interfaces • Integration Tests • System Level Tests • User Acceptance Tests
VII	Implementation	<ul style="list-style-type: none"> • Implementation of application into production hosting environment • Establish change control board (nominate members and assign roles) • Train end-user community on application usage • Train end-user community on the change processes • Initiate change control processes • Release application to end-user community

Phase	Description	Work Tasks
VIII	Written Report and Presentation	<ul style="list-style-type: none">• Generate written report in support of the project• Generate overview presentation of the project

The ultimate outcome of this project is to produce a change management process, centralize a knowledge repository, and a prototype that supports both. Each of the phases identified in the previous table supports that goal. The project will use a blend of the Comprehensive Delivery Processes governance model and the Agile eXtreme Programming paradigm to ensure that proper project accountability and the customers desires for rapid prototyping are addressed.

CHAPTER III

Framework Definition and Design

The SPTOF framework is modeled on the N-Tiered Architecture and SOA (Service Oriented Architecture) paradigms. Arguably, these two paradigms have been the driving forces behind the last fifteen years of object oriented computing. The framework strives to unify two computing focuses, strategy and implementation, by providing a mechanism to relate the products of each. The product of this unified focus a third product that can be leveraged by the Architecture, Application Integration / Design / Development, and Operations support teams.

SPTOF's design principles are founded on flexibility. It is intended to accommodate a wide range of business models. Essentially, the determination of the deployment and use of SPTOF is left to the organization. Consistency is the only real limiting factor. Once an organization determines the SPTOF framework alignment with the business, it is wise to maintain that standard usage paradigm throughout the enterprise.

This section of the document provides an overview of the definition and design of the SPTOF framework. It will document the foundations and intent of the framework.

Alignment with Industry Standards

As stated, the SPTOF framework is founded on the N-Tiered Architecture and SOA paradigms. It adopts many of the terms from both; consequently, the definition of SPTOF will leverage these terms and their respective definitions.

High-Level Definition

The framework is comprised of two foundational views of the software development life cycle. The strategic view facilitates a projected view of a particular computing domain or environment within that organization. The strategic view can be defined as the desired or “to-be” direction for that particular domain.

The next view found in the framework includes the operational view. This view is referred to as the “as-is” view. It provides an inventory of the current applications and hosting components deployed in support of said applications.

Individually, these two views provide specific horizontal and vertical views of a computing domain. These two views are factored together into a third product. This view is considered the tactical view. The tactical view provides a view of the domain that facilitates and an understanding of the available components and services that align with strategic architectural models. It also provides an understanding of what elements of an environment that are lacking.

Strategic Framework Focus

SPTOF strategic components are aligned with the N-Tiered models found in many application architectures. The strategic components can be equated as containers filled with subcomponents. This type of interrelationship makes the strategic view easier to comprehend and manage. The following diagram depicts the relationship between the various components of the strategic portion of the framework.

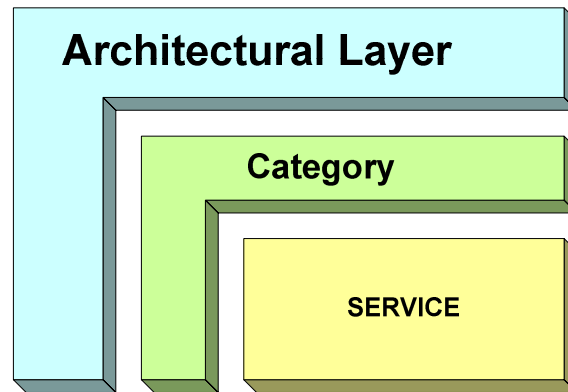


Figure 4: Strategic Frame Layout Diagram

As depicted in the diagram, the “Architectural Layer” serves as the base container, followed by the “Category” container, which provides a means to categorize the services that are associated with a particular architectural layer. The remaining container, the “Service” container, facilitates the services that are associated with an architecture layer.

Architectural Layers

An Architectural Layer is a model in which each layer takes on a specific function within the system. These layers are logical components and contain no functionality in and of themselves. Each layer takes on one high-level function. (McGovern, Ambler, Stevens, Linn, Jo, Sharan, 2003, p. 61). For example, the presentation layer in a given architecture encapsulates those services that provision presentational services, including:

- Web Server
- IVR (Integrated Voice Recognition)
- Portal services

The following diagram depicts a theoretical N-Tiered architectural computing model.

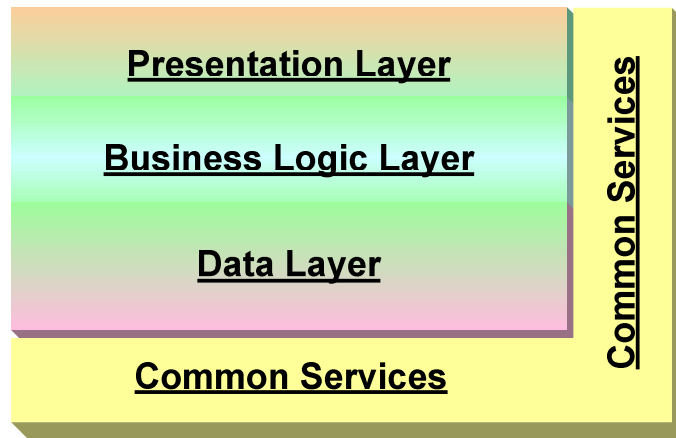


Figure 5: N-Tier Stratification Diagram

As this diagram progresses the model's layer dependencies and abstractions become clear. For example, the "Presentation Layer" does not directly interact with the "Data Layer"; it leverages the "Business Logic Layer" to proxy requests for transformed data. The "Common Services" and "Core Infrastructure" supports all layers. The framework will leverage this logic to establish collections of services that can be projected into a domain or associated with application services facilitation.

Categories

A category provides an intermediate container for services. Storage of services directly in the architectural layer does not provide for logical categorization of services. As services accumulate within the framework categories, they provide a necessary level of stratification to facilitate management and reporting frameworks.

Services

This component of the SPTOF framework is the essential element in generating strategic and tactical perspectives. A service is a software component that can be used as a part of single application implementation or as a part of the overall business process. A service encapsulates its

own state and the business data it is processing. Services can communicate through the service interface it exposes.

There is a great deal of confusion today, around SOA and its relationship to web services. The service component of SOA is not exclusively Web Services. While Web Services fit nicely into the definition of service previously provide, there are many examples of services that existed long before the conception of Web Services took hold. For example:

- Java's J2EE and RMI components
- CORBA Services
- Microsoft's OLE and COM services

All of the above exposed interfaces that are consumed externally or internally, this is not intended to be a slight on Web Services. Web Services are an extremely interesting and exciting new technology.¹ The important take-away is to understand services come in many forms and interface types. Many applications or software packages offer services that can be used to realize business process logic. That is the base tenant of the SPTOF framework.

The relationship found in the "Strategic Frame Layout Diagram" is a foundational view of the framework itself, but it is extensible. Categories and services can be associated with one or more Architectural Layers. The SPTOF framework provides the flexibility to allow the larger chunks of strategic information to be associated or disassociated with an architectural layer. As with

¹ SPTOF in its current implementation does not provide for integration with a UDDI (Universal Description, Discovery and Integration) servers. However, this is a definite opportunity in future implementations of the SPTOF framework.

most flexible implementations, broad brush movement of data poses some risk. Any category association relationship should be analyzed carefully prior to modification.

As with categories, services can enjoy the benefits of this flexible association paradigm. Services occur at a far more granular level of implementation. A single service can be associated with multiple categories and architectural layers. Again, any associated relationship should be analyzed carefully prior to modification. These interrelationships are depicted in the following diagram.

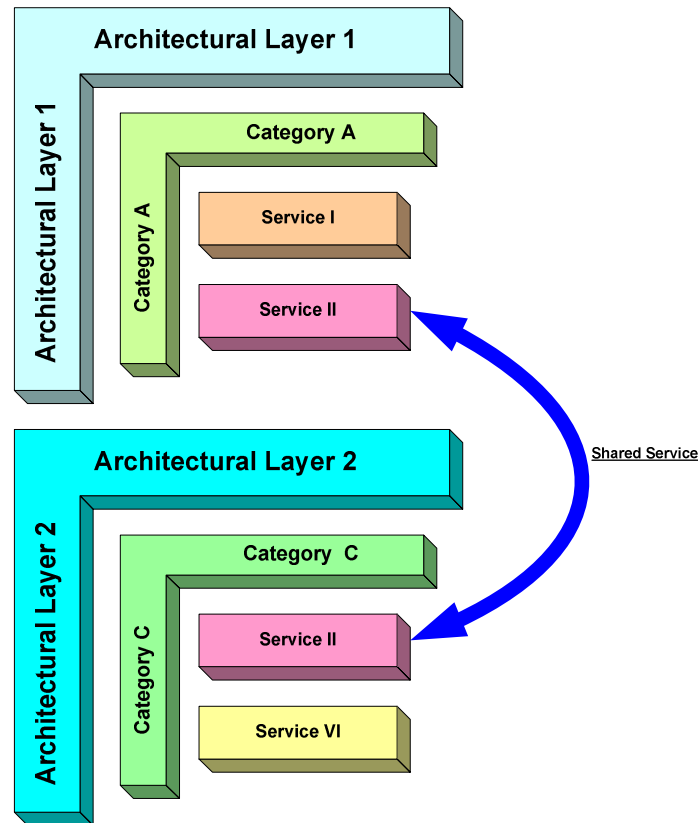


Figure 6: Strategic Framework Inter-Relationships Diagram

As depicted, “Service II” is associated with both “Architecture Layer 1 / Category A” and “Architecture Layer 2 / Category C”. In theory this could imply that “Service II” provides a

similar or dissimilar functionality depending on its interpreted implementation. The SPTOF framework implementation and maintenance teams should avoid this ambiguity. It is more desirable to uniquely define a service's functionality and corresponding name, rather than promote misinformation in the repository.

Clarity is imperative with establishing the SPTOF framework in any organization. Using clear and concise names and descriptions for strategic components stored within the repository will yield enormous benefits during daily use of the framework. As the strategic architecture publications tend to be ambiguous in nature, clarity for components found in the strategic portion of the framework has a greater significance.

Operational Framework Focus

At the other end of the framework's spectrum is the operational focus. The operations portion is comprised of two essential specifications:

- Domain specification
- Application specification

Together these two specifications are the foundation for the generation and operational view. The following subsections will outline these two specifications and their respective interrelationship.

Domain Specification

A domain is a conceptual container that defines a logical grouping of platforms or components that make up a hosting environment. Domains provide an anchor point for service projections

and application deployment specifications. The following diagram presents the domain container graphically:

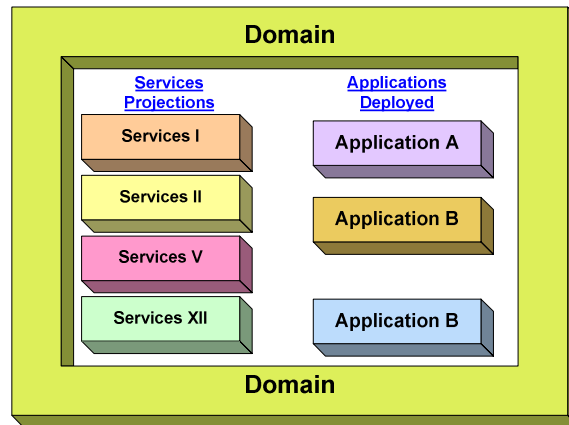


Figure 7: Domain Specification Diagram

Domain containers facilitate encapsulated views of the deltas between the strategic projections and the operational realities. The domain portion does not facilitate the concept of asset management. The SPTOF application is not intended to replace or directly integrate with current asset management implementations. It simply maintains a complementary view of the environment for applications and hosting servers.

Application Specification

The strict definition of the term application is, “A program or group of programs designed for end users. Software can be divided into two general classes: systems software and applications software. Systems software consists of low-level programs that interact with the computer at a very basic level. This includes operating systems, compilers, and utilities for managing computer resources.” (Webopedia, 2003). This definition holds true within the scope of the SPTOF

framework and is expanded to include those software and firmware components that facilitate consumable services.

Here again, the framework maintains an ambiguous stance. An organization could include operating systems, middleware applications, and external service provider, should it so desire. This is entirely up to the implementation and maintenance staff for the SPTOF framework. As with the strategic portion of the framework, consistency is essential. The definition should be clearly documented and communicated to the SPTOF user community.

Applications and their services will realize a natural association with architectural layers and categories based on the strategic specification previously defined. In some cases, an application will support or act as bridge between two or more architectural layers. This relationship is depicted in the following diagram:

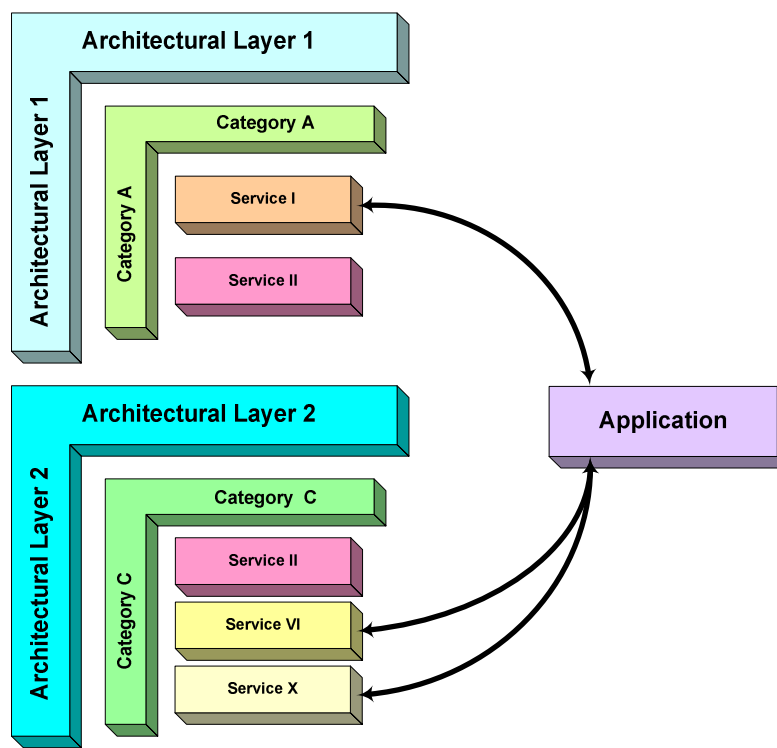


Figure 8: Application Specification Diagram

It is important to understand this concept. As previously stated, the strategic layer does not contain any tactical information. The association of services with an application bridges this knowledge gap between the strategic view and operational view.

The specification of an application must maintain accuracy down to the revision level. Often services or functionality is added or subtracted between release levels. In short, a service provided in revision 1.0 may have been removed from an application functional inventory in revision 5.1.4.

Vendor management information is an important part of an application's specification. It provides the interested party with information about application support and opportunities for updating functionality. In the prototype release, vendor information will be limited to contact information².

Domain Application Deployment

Once application specifications have been stored in the repository, the next step is to associate the application within a domain. This is accomplished by defining a deployment record. This is an association between an application specification record and domain specification record. These associations facilitate the view of the application and its associated services deployed into a particular domain. The following diagram illustrates this association.

² There are opportunities to integrate the SPTOF framework with a previous established vendor management system. The SPTOF frame is not intended to be used as a vendor management system.

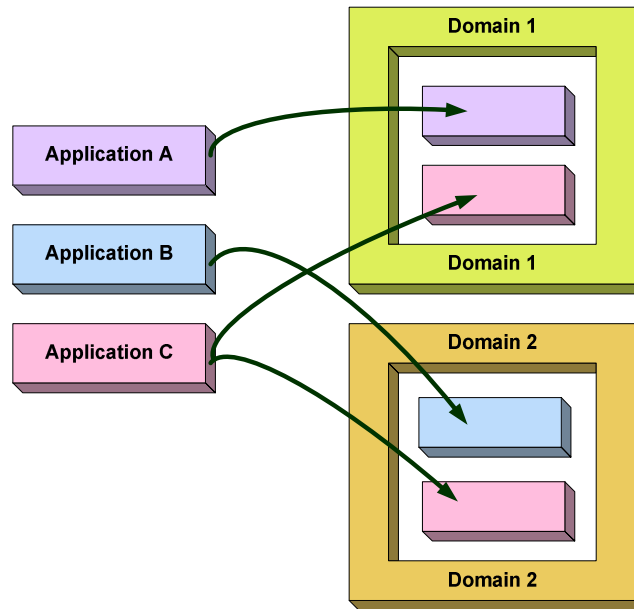


Figure 9: Domain and Services Interrelationship Diagram

As depicted in the previous diagram, the associative view comprises the larger operational view. Applications can be deployed in multiple domains supporting multiple business functions. Additionally, this view supports analysis of possible one-off implementations.

Strategic Projection

Within the context of the SPTOF framework, the strategic projection provides a vehicle for the architecture team to define a projected specification of which services should be deployed in a specific domain. The determination of which services are required is the responsibility of the architecture team in conjunction with the IT strategy publication.

The association of services determines the inheritance of architectural layers and categories that are represented in this view of the domain. The following diagram depicts the projection association of the services with a domain.

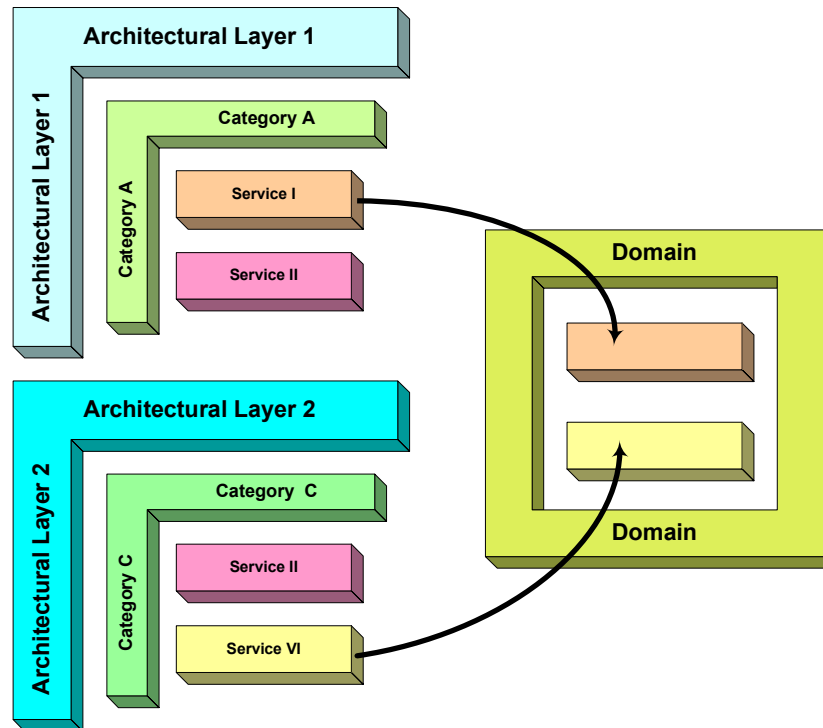


Figure 10: Strategic Projection Diagram

This view is a future one, but the tactical development teams can leverage this information to determine future directions of designs that support business visions and process.

Tactical Portion

The final portion of the SPTOF framework supports the tactical view. The tactical view is the product of comparing the strategic projection and the actual deployment inventory. The comparison yields different products. This is perhaps the greatest value of SPTOF framework. These products bridge the knowledge gap between strategic and operational views by creating a single tactical view.

The comparison is based on the relationship between services and the applications deployed providing those same services. The domain container is the focal point for the comparison. As previously stated, domain containers facilitate an encapsulation of view deltas between the two strategic projections and the operational realities. The following diagram depicts these relationships:

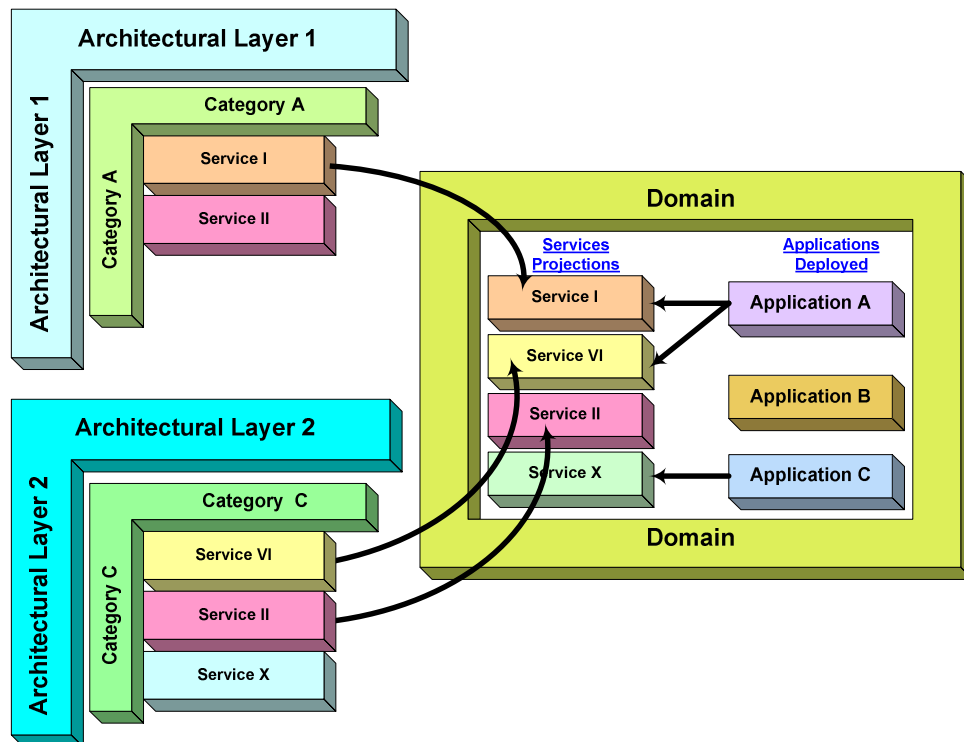


Figure 11: Tactical Perspective Diagram

Each of these two products can yield additional information useful to the software development life cycle teams in variety of ways. For the purposes of this document, just the primary products will be discussed.

The first product is a view facilitates an understanding of what applications are deployed in a specific domain and the services made available by the applications. The value to designers and development is clarity. These teams will no longer have to guess what services are deployed in

the particular environment. Additionally, developers are empowered to research services outside of their domain, thereby enabling them to realize reuse of external assets.

The second product is a view of the gaps between projections and real deployment inventories. If a service is projected into a domain that has no application deployed in support of that service a gap is identified. Once a gap is identified, costly delays in design and deployment can be avoided. Designer and developers can work with operations staff to get the proper services deployed. Again, there is always the opportunity to look in another domain for reuse alternatives for the desired services.

CHAPTER IV

Change Control Process Definition and Design

The change management process supports the actions of the governing body by insuring the integrity of the knowledge repository. The governing body will consist of one or more member representatives from each of the three disciplines defined in the problem statement. Functions of the body include:

- Authorizing the publication of strategic information
- Authorizing changes to the tactical and operational data in repository
- Providing guidance on change management processes
- Processing change requests from various disciplines

The process will be comprised of a series of checks and balances. These checks and balances will serve to govern and control the content and daily functionality this framework will support.

This section will detail the change management process, providing narratives and flow diagrams. The roles and responsibilities are defined from a generalist perspective and may require refinement during the implementation phase of the project.

Change Management Process Definition

Process Name: Strategic Planning Tactical Operations Framework Change Analysis

Purpose: This document focuses the processes to maintain a Strategic Planning Tactical Operations Framework environment. The Architecture, Development, and Operations Teams have the responsibility to maintain clear channels of communications to ensure the Development, Operations, Test, and Software Configuration environments accurately maintain standard specifications. This section of the document addresses the processes to maintain the integrity of that communication and the associated SPTOF application that maintains the data used to document the projected and current (SOE) standard operating environment.

Documentation supporting the various views of the SOE provides a description of services and software components deployed in the environment. This description includes a generalized overview as well as more detailed information. Intended audience for the SOE includes those individuals interested in the detailed specification. This documentation facilitates the technical needs of clarification or reference of the projected and current (SOE) standard operating environment.

Control Mechanisms: All change requests to modify the processes contained in this document should be routed to the SPTOF Change Control Board for proper consideration and approval.

Entry Conditions:

- New or upgraded components of the SPTOF specification are identified
- Components of the SPTOF framework require modification or removal

Exit Conditions:

- Change board approved changes applied to the current SPTOF Repository
- Revised *Current* SOE is published **–or–**
- New Projected SOE is published

Stakeholders for the process

The stakeholders for this process are senior members of teams that have direct input to or consume information about the standard operating environment. This information enables various teams to leverage the standard operating environment in meeting strategic or tactical goals of the business.

The stakeholders for this process include designated senior members of the following software lifecycle teams:

- Enterprise Architecture
- Application Architecture
- Application Design
- Application Development
- Application Test
- Software Quality Assurance
- Software Configuration Management
- Strategic Operations

Define Roles with the process scope

There are three main roles within the scope of the change management process. Each role represents and supports a discipline within the software development life cycle. The following table defines each role and provides a brief description of the area of interest within the context of the change management process.

Table 2: Role Definition

Role	Description	Function
Architect	Architects will that serve on the change control board represent the local and enterprise architecture teams.	Their focus is on the strategic vision for the business at large. They often will serve as the business vision representative.

Role	Description	Function
Design/Developer	This role represents the application design and development teams within the organization.	These representatives have a tactical focus, concerning themselves with the current implementation environment. Additionally, they have secondary focuses on near-term strategic projects. This aids them in determining a target for future application implementations.
Operations	The operations role represents the environmental support teams. This includes production, development, and testing environments.	Operations teams are focused on the aspects of production support. For this team, the important issues are application uptime and environmental stability. Operations will act as a check in the “check-and-balance” portion of the change management process.

Strategic Planning Tactical Operations Framework Change Control Process Steps

The change management processes are comprised of certain input, outputs and processing steps.

The following table and subsequent diagram facilitate a definitive example of the change management process.

Table 3: Change Inputs, Outputs and Process Steps Narrative

Inputs	Process Steps	Outputs
1 Message Notification for Change Request to the SPTOF Change Control Board: <ul style="list-style-type: none"> •Environment targeted for change •Submitter •Description of change •Date Required 	1 Message sent to SPTOF Change Control Board, including information about desired change. 2 Change Request initialization and assignment triggers new work for SOE. 3 All applicable input artifacts are gathered and documented. 4 Designated teams will provide analysis on the effects of the Change Request at the component level. ³	1. Published changes or enhancements for the prescribed environment into the supporting documentation. <ol style="list-style-type: none"> a. Revised existing document published. b. New document containing projected enhancements published.

³ The Change Management Governance Board will collaborate on the analysis steps of this process.

Inputs	Process Steps	Outputs
<ul style="list-style-type: none"> • Effected Systems or Services <p>2 Change Request Outputs:</p> <ul style="list-style-type: none"> • Justification • Requirements • Measures of Success • Additional facilities requested by the business • Fix identified for a problem within the SOE scope • Metrics analysis identifies additional capacity needs • Project alignment • Vendor events • Technology events • Strategic planning <p>3 Most recently published Architecture, Development and Operations SOE contributions.</p>	<p>4.1 Devise alternative Architectures.</p> <p>4.2 Select architecture from the list of component alternatives.</p> <p>5 Analyze the effects of the Change Request at the global level.¹</p> <p>5.1 Devise alternative Architectures.</p> <p>5.2 Select architecture from the list of global alternatives.</p> <p>6 All individual models are incorporated into a proposed SOE document.</p> <p>7 SPTOF Change Control Board chairs the High-Level SOE Design Team⁴ and facilitates the review of the proposed SOE⁵. The reviewing body⁶ is made up of the High-Level SOE Design Team and pre-designated members of the Architecture, Development and Operations community.</p> <p>8 Analyze the results of the review (<i>if necessary</i>).</p> <p>8.1 Return to Step (4) to re-model solution.</p> <p>8.2 Approved</p> <p>9 Next steps analyzed</p> <p>9.1 Revised SOE is published</p> <p>9.2 New SOE Project is initiated and Projected SOE is published</p>	

⁴ The Change Management Governance Board will consist, at a minimum, of representative(s) from the SPTOF Change Control Board.

⁵ The voting body will be established from the Change Management Governance Board and pre-determined reviewing body.

⁶ Reviewing body may differ depending on the scope and complexity of the Change Request.

Process Design Overview Diagram based on Process documentation

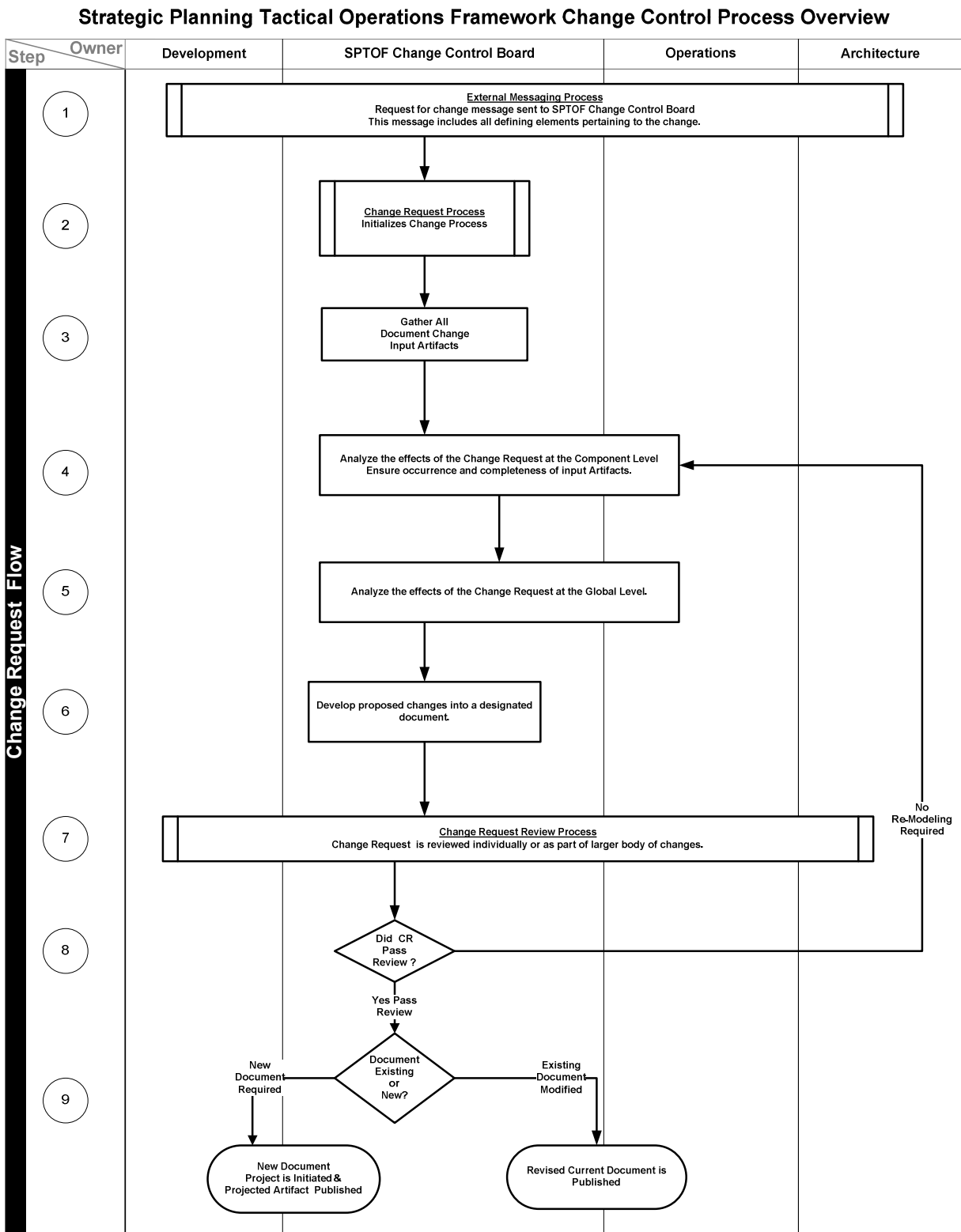


Figure 12: Change Control Process Overview (Diagram)

CHAPTER V

Prototype Application Definition & Design

The Strategic Planning Tactical Operations Framework requires a prototype application to support a proof of concept for the framework base functionality. This section documents the requirements, functional specification, and high-level design for this prototype application design.

Supporting UML Artifacts

The UML modeling information and supporting data provides a translation of the functionality requirements for the Strategic Planning Tactical Operations Framework. The Use Cases for this project fall into one of the five following categories:

- 1) Security Functions
- 2) Architectural Functions
- 3) Developer Functions
- 4) Operations Functions
- 5) Reporting Functions

Each section contains a Use Case diagram and the supporting Use Case text.

Security Functions Use Cases

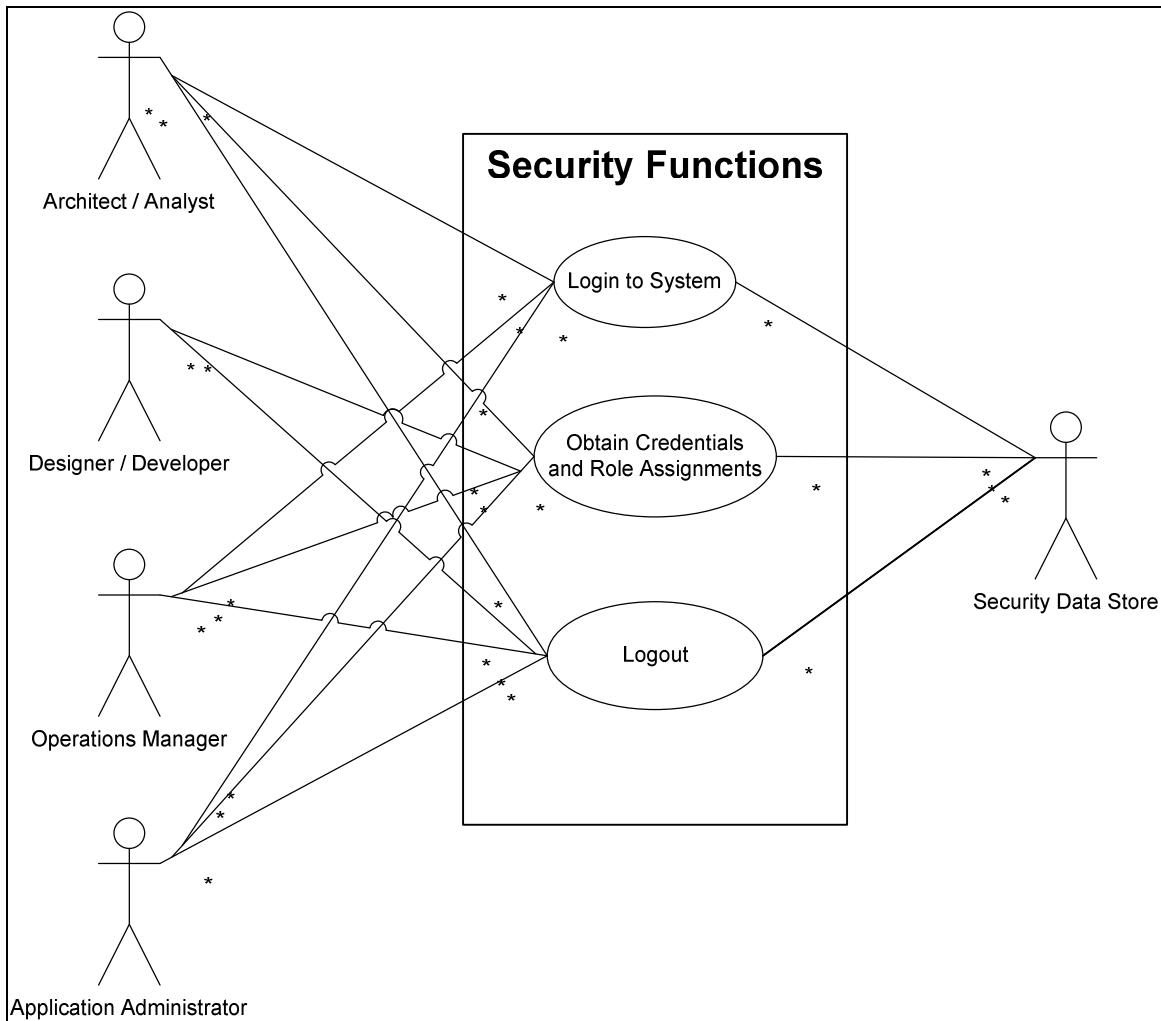


Figure 13: Security Function (Use Case Diagram)

Login Functions

Use Case Name:	Login to SPTOF Application
Actor(s):	Architects, Developers, Operations, Administration
Goal:	Identify and Authenticate User.
Trigger(s):	Actor desires access to SPTOF application.
Pre-Conditions:	Application session has not yet been established or previous session has timed out.
Output:	Actor gains authenticated access to the application.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Actor accesses application via assigned user interface.
2	System confronts actor with user identifier and password challenge.
3	Actor submits user identifier and password combination.
4	System authenticates actor to the system.
NOTES:	

Use Case Name:	System assigns user credentials to user session based on authentication information
Actor(s):	Architects, Developers, Operations, Administration
Goal:	Assign proper credentials to actor based on assigned role designated to actor-authenticated session.
Trigger(s):	Actor gains authenticated access to the application.
Pre-Conditions:	Actor's assigned credentials exist in the security repository.
Output:	Actor's session is assigned role-based credentials that are extracted from the security repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Actor's authenticated user identifier is compared to available assigned credentials in the security repository.
2	Actor's session is assigned role-based credentials that are extracted from the security repository.
NOTES:	

Logout Functions

Use Case Name:	Logout of SPTOF Application
Actor(s):	Architects, Developers, Operations, Administration
Goal:	Complete actor's session and nullify any session information.
Trigger(s):	Actor desires logout of the SPTOF application or session timeout occurs.
Pre-Conditions:	Valid Application session exists.
Output:	Actor is logged out of the application and any session information nullified
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Actor selects logout option from the user interface.
2	System verifies active session.
3	All session variables, actor credentials, and any session information are nullified.
4	Actor is presented an information message that the session has been terminated.
NOTES:	

Architect Function Use Cases

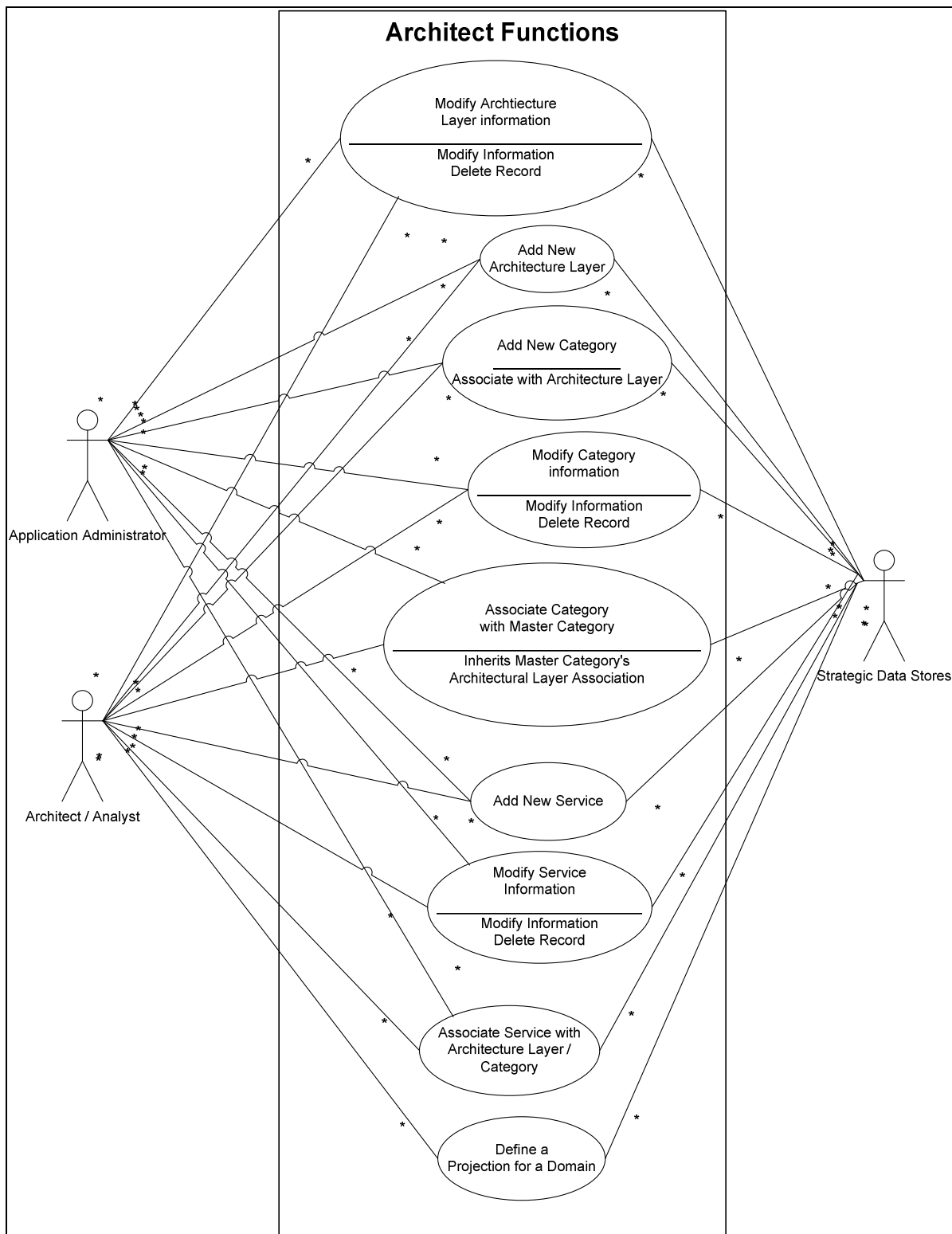


Figure 14: Architecture Functions (Use Case Diagram)

Architectural Layer Management

Use Case Name:	Add Architectural Layer
Actor(s):	Architects
Goal:	Instantiate New Architecture Layer.
Trigger(s):	Architecture Team determines that a new layer is required to support the logical architecture infrastructure.
Pre-Conditions:	Architecture Layer is researched, necessary changes are clearly defined and the architecture body has determined this layer adds values to the environment.
Output:	Architecture Layer defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Architecture Layer identified at conceptual level.
2	Architect defines the specifications for the layers.
3	Architecture governance board authorizes the layer for input.
4	Architects input the layer into repository.
NOTES: The Architectural Layer is the base component for the Architectural Layer / Category / Service relationship. The communications of the completion of this layer data input must be published to the SPTOF community.	

Use Case Name:	Modification Architectural Layer Information
Actor(s):	Architects
Goal:	Update Architecture Layer Information
Trigger(s):	Architecture Team determines that newer layer information is required to support the logical architecture infrastructure.
Pre-Conditions:	Architecture Layer modification information is clearly defined and the architecture body determines that the information pertaining to layer, adds values the environment.
Output:	Architecture Layer information is defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Architecture Layer update information is identified at conceptual level.
2	Architect defines the update specifications for the layers.
3	Architecture governance board authorizes the layer update for input.
4	Architects input the layer into repository.
NOTES:	

Use Case Name:	Delete Architectural Layer Information
Actor(s):	Architects
Goal:	Delete Architecture Layer Information
Trigger(s):	Architecture Team determines that existing layer information is no longer wanted or needed to support the architecture infrastructure.
Pre-Conditions:	Architecture Layer targeted is clearly defined and the architecture body determines that the layer information may be removed from the environment.
Output:	Architecture Layer information is removed from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Architecture Layer to be is identified.
2	Architecture governance board authorizes the layer for deletion.
3	Architect identifies and validates the layer for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Architectural Layer / Category Information Association
Actor(s):	Architects
Goal:	Facilitate the associating Architectural Layer to Category information. This association needs to support the architecture infrastructure specification.
Trigger(s):	Architecture Team determines that existing layer information needs to be associated with specific existing categories.
Pre-Conditions:	Architectural Layer and Category information must preexist prior to association.
Output:	Architecture Layer information is removed from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Architecture Layer / Category are identified.
2	Architecture governance board authorizes the association.
3	Architect identifies and validates the Architecture Layer / Category association.
4	Architect places the Architecture Layer / Category association in the repository and validates the association.
NOTES:	

Use Case Name:	Single Architectural Layer Information Report
Actor(s):	Architects
Goal:	Facilitate a report of single Architectural Layer information.
Trigger(s):	SPTOF User requires a quick view of Architectural Layer information.
Pre-Conditions:	Architectural Layer information must preexist in repository.
Output:	Single Architecture Layer information is reported based on record from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Architecture Layer record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report formation should contain the following minimum detail:	
<ul style="list-style-type: none"> • Architecture Layer Name • Architecture Layer Description • Architecture Layer / Category association 	

Category Information Management Use Cases

Use Case Name:	Add Category
Actor(s):	Architects
Goal:	Instantiate New Category
Trigger(s):	Architecture Team determines that a new category is required to support the logical architecture infrastructure.
Pre-Conditions:	Category researched, is clearly defined, and the architecture body has determined this category adds value to the environment.
Output:	Category defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category identified at conceptual level.
2	Architect defines the specifications for the category.
3	Architecture governance board authorizes the category for input.
4	Architects input the category into repository.
NOTES: The Category is the base component for the Category / Service relationship. The communications of the completion of this category data input must be communicated to the SPTOF community.	

Use Case Name:	Modification Category Information
Actor(s):	Architects
Goal:	Update Category Information
Trigger(s):	Architecture Team determines that newer category information is required to support the logical architecture infrastructure.
Pre-Conditions:	Category modification information is clearly defined and the architecture body determines that the category information adds value to the environment.
Output:	Category information is defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category update information is identified at the conceptual level.
2	Architect defines the update specifications for the category.
3	Architecture governance board authorizes the category update for input.
4	Architects input the category into repository.
NOTES:	

Use Case Name:	Delete Category Information
Actor(s):	Architects
Goal:	Delete Category Information
Trigger(s):	Architecture Team determines that existing category information is no longer wanted or needed to support the architecture infrastructure.
Pre-Conditions:	Category targeted is clearly defined and the architecture body determines that the category information may be removed from the environment.
Output:	Category information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category to be is identified.
2	Architecture governance board authorizes the category for deletion.
3	Architect identifies and validates the category for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Delete Category Information
Actor(s):	Architects
Goal:	Delete Category Information
Trigger(s):	Architecture Team determines that existing category information is no longer wanted or needed to support the architecture infrastructure.
Pre-Conditions:	Category targeted is clearly defined and the architecture body determines that the category information may be removed from the environment.
Output:	Category information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category to be deleted is identified.
2	Architecture governance board authorizes the category for deletion.
3	Architect identifies and validates the category for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Category / Architectural Layer Information Association
Actor(s):	Architects
Goal:	Facilitate the associating Category / Architectural Layer information. This association needs to support the architecture infrastructure specification.
Trigger(s):	Architecture Team determines that existing layer information needs to be associated with specific existing categories.
Pre-Conditions:	Architectural Layer and Category information must preexist prior to association.
Output:	Architectural Layer / Category information is associated in the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category / Architectural Layer records are identified.
2	Architecture governance board authorizes the association.
3	Architect identifies and validates the Category / Architectural Layer association.
4	Architect places the Category / Architectural Layer association in the Repository and validates the association.
NOTES:	

Use Case Name:	Single Category Information Report
Actor(s):	Architects
Goal:	Facilitate a report of single Category information.
Trigger(s):	SPTOF User requires a quick view of Category information.
Pre-Conditions:	Category information must preexist in repository.
Output:	Single Category information is reported based on record from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Category record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report formation should contain the following minimum information: <ul style="list-style-type: none"> • Category Name • Category Description • Architectural Layer / Category Association 	

Service Information Management Use Cases

Use Case Name:	Add Service
Actor(s):	Architects
Goal:	Instantiate New Service
Trigger(s):	Architecture Team determines that a new layer is required to support the logical architecture infrastructure.
Pre-Conditions:	The service is researched, clearly defined, and the architecture body determines this service adds value to the environment.
Output:	Service defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Service identified at conceptual level.
2	Architect defines the specifications for the layers.
3	Architecture governance board authorizes the layer for input.
4	Architect input the layer into repository.
NOTES: The Service is the base component for the Service / Category / Service relationship. The communications of the completion of this layer data input must be communicated to the SPTOF community.	

Use Case Name:	Modification Service Information
Actor(s):	Architects
Goal:	Update Service Information
Trigger(s):	Architecture Team determines that newer layer information is required to support the logical architecture infrastructure.
Pre-Conditions:	Service modification information is clearly defined and the architecture body determines that the service information adds values the environment.
Output:	Service information is defined and placed into the repository to be leveraged throughout the architecture projection and product realization analysis.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Service update information is identified at conceptual level.
2	Architect defines the update specifications for the layers.
3	Architecture governance board authorizes the layer update for input.
4	Architect inputs the layer into repository.
NOTES:	

Use Case Name:	Delete Service Information
Actor(s):	Architects
Goal:	Delete Service Information
Trigger(s):	Architecture Team determines that existing layer information is no longer wanted or needed to support the architecture infrastructure.
Pre-Conditions:	Service targeted is clearly defined and the architecture body determines that the layer information may be removed from the environment.
Output:	Service information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Service to be is identified.
2	Architecture governance board authorizes the layer for deletion.
3	An architect identifies and validates the layer for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Delete Service Information
Actor(s):	Architects
Goal:	Delete Service Information
Trigger(s):	Architecture Team determines that existing layer information is no longer wanted or needed to support the architecture infrastructure.
Pre-Conditions:	Service targeted is clearly defined and the architecture body determines that the service information may be removed from the environment.
Output:	Service information is removed from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Service to be deleted is identified.
2	Architecture governance board authorizes the layer for deletion.
3	Architect identifies and validates the service for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Service / Architectural Layer / Category Information Association
Actor(s):	Architects
Goal:	Facilitate the associating Service to Category information. This association needs to support the architecture infrastructure specification.
Trigger(s):	Architecture Team determines that existing layer information needs to be associated with specific existing categories.
Pre-Conditions:	Service and Category information must preexist prior to association.
Output:	Service information is removed from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Service / Architectural Layer / Category are identified.
2	Architecture governance board authorizes the association.
3	Architect identifies and validates the Service / Architectural Layer / Category association.
4	Architect places the Service / Architectural Layer / Category association in the repository and validates the association.
NOTES:	

Use Case Name:	Single Service Information Report
Actor(s):	Architects
Goal:	Facilitate a report of single Service information.
Trigger(s):	SPTOF User requires a quick view of Service information.
Pre-Conditions:	Service information must preexist in repository.
Output:	Single Service information is reported based on record from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Service record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report format will need to contain at minimum the following detail: <ul style="list-style-type: none"> • Service Name • Service Description • Service / Category Association 	

Architecture Domain Projections Use Cases

Use Case Name:	Associates Architectural Layer / Category / Service record with a Domain
Actor(s):	Architects
Goal:	Project an association of an Architectural Layer / Category / Service Record with a specific domain.
Trigger(s):	Architecture Team determines that an Architectural Layer / Category / Service Record should be projected into a domain to support the logical view of required functionality.
Pre-Conditions:	The predefined Architectural Layer / Category / Service Record is defined and validated prior to any operation.
Output:	A record of the Architectural Layer / Category / Service Record are defined as having a valid association with the specified domain.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	A new Architectural Layer / Category / Service Record are defined and can be projected into a specific domain.
2	Architect defines the need for this association.
3	Architecture governance board authorizes the association.
4	An architect inputs a record of the Architectural Layer / Category / Service Record association with the specified domain.
NOTES: This process occurs within larger chunks of operations functionality. The end user will not be tasked with putting each record into the repository individually.	

Use Case Name:	Deletion of the association of Architectural Layer / Category / Service Record with a Domain
Actor(s):	Architects
Goal:	Update association of Architectural Layer / Category / Service Record with a specific Domain Information
Trigger(s):	Architecture Team determines that a relationship between the Architectural Layer / Category / Service Record and a specific domain must be terminated.
Pre-Conditions:	A record of the Architectural Layer / Category / Service Record are defined as requiring deletion, and the record exists in the repository.
Output:	The relationship of the Architectural Layer / Category / Service Record defined as having a valid association with the specified domain is terminated.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	The need for terminating a defined relationship between an Architectural Layer / Category / Service Record and a specific domain is identified at the conceptual level.
2	Architect defines the need for this terminated association.
3	Architecture governance board authorizes the termination of this association.
4	An architect deletes the record of the Architectural Layer / Category / Service Record association with the specified domain.
NOTES: This process occurs within larger chunks of operations functionality. The end user will not be tasked with putting each record into the repository individually.	

Use Case Name:	Report on current Architectural Layer / Category / Service Record information for a specified domain.
Actor(s):	Architects
Goal:	Provide a report of current domain projection information.
Trigger(s):	Architecture Team requires a view of what Architectural Layer / Category / Service Record information is projected for a specific domain.
Pre-Conditions:	The name of the specific domain of interest is known for query facilitation.
Output:	Generate a report of current domain projection information.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Architect identifies the domain of interest.
2	Information gathered from the repository depicting the current association between Architectural Layer / Category / Service Record information and the specified domain is gathered.
3	A report of current domain projection information is presented to the architect.
NOTES:	

Developer Functions Use Cases

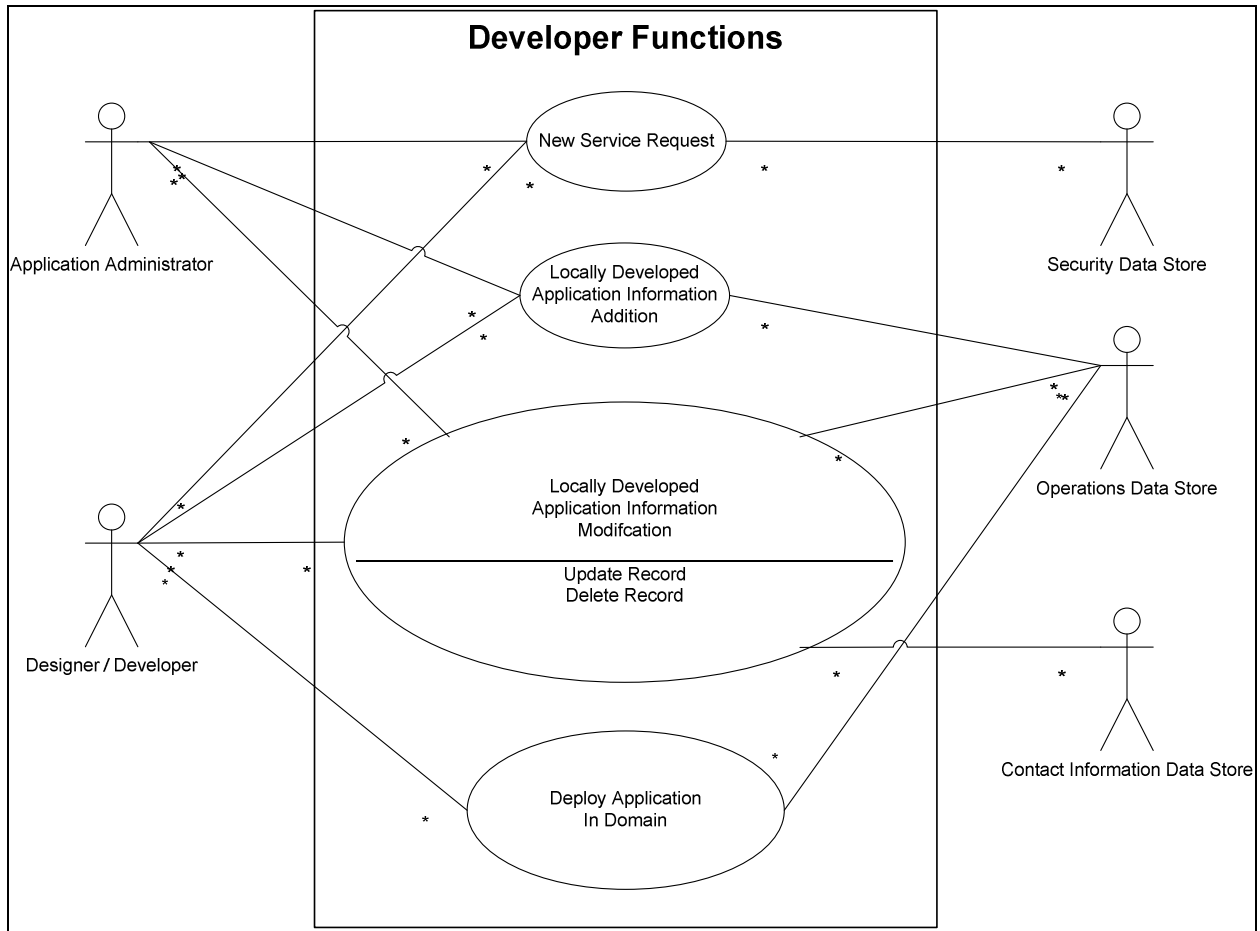


Figure 15: Developers Functions (Use Case Diagram)

Developer Request

Use Case Name:	Developer Request Facility
Actor(s):	Developer
Goal:	Provide a means for developers to request new services or application implementation.
Trigger(s):	Developer identifies a missing component or service gap in a given domain.
Pre-Conditions:	The Domain must exist or be defined by the operations team.
Output:	Completed request form is sent to the SPTOF change control board.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Developer defines a missing Architectural Layer / Category / Service Record or application that supports that record in a specific domain.
2	Developer defines the problem in a captured environment (a means of collecting the data).
3	Developer submits the information captured to the SPTOF change control board.
NOTES: This process occurs within large chunks of operations functionality. The end user will not be tasked with putting each record into the repository individually.	

Locally Developed Application Management

Use Case Name:	Add Locally Developed Application Information
Actor(s):	Developer
Goal:	Instantiate New Record for Locally Developed Application
Trigger(s):	Development Team determines that a Locally Developed Application will provide new services suitable for consumption.
Pre-Conditions:	All services provided by a Locally Developed Application are identified.
Output:	New Record for Locally Developed Application placed into the repository to be leveraged throughout a specific domain or the entire enterprise.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	All services provided by a Locally Developed Application are identified and categorized.
2	Development team decides on the validity of the application to provide reusable services.
4	Locally Developed Application Information is input into repository.
NOTES: The Architectural Layer is the base component for the Architectural Layer / Category / Service relationship. The communications of the completion of this layer data input must be communicated to the SPTOF community.	

Use Case Name:	Modification of Locally Developed Application Information
Actor(s):	Development
Goal:	Update Record for Locally Developed Application
Trigger(s):	Development Team determines that new Locally Developed Application information is required that will accurately depict the application or service(s) it provides.
Pre-Conditions:	Locally Developed Application information is clearly defined and the Development body determines that the record needs to be modified.
Output:	Updated information about a Locally Developed Application placed into the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	All services provided by a Locally Developed Application are identified and categorized.
2	Development team decides on the validity of the application to provide reusable services.
3	Locally Developed Application Information is input into repository.
NOTES:	

Use Case Name:	Delete Locally Developed Application Information
Actor(s):	Developer
Goal:	Delete Locally Developed Application Information
Trigger(s):	Development Team determines that existing Developed Application Information is no longer wanted or needed to support the Development infrastructure.
Pre-Conditions:	Locally Developed Application targeted is clearly defined and the Development body determines that the layer information may be removed from the environment.
Output:	Locally Developed Application information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Locally Developed Application is identified as a candidate for deletion.
2	Development team authorizes the layer for deletion.
3	Developer identifies and validates the layer for deletion.
4	Repository system validates the deletion.
NOTES:	

Use Case Name:	Single Locally Developed Application Information Report
Actor(s):	Developer
Goal:	Facilitate a report of single Locally Developed Application Information.
Trigger(s):	SPTOF User requires a quick view of Locally Developed Application Information.
Pre-Conditions:	Locally Developed Application Information must preexist in repository.
Output:	Single Locally Developed Application information is reported based on record from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Locally Developed Application record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report formation should contain the following minimum information: <ul style="list-style-type: none"> • Locally Developed Application Name • Locally Developed Application Description 	

Use Case Name:	Deploy Locally Developed Application Information in Specific Domain
Actor(s):	Developer
Goal:	Associate a given Locally Developed Application Information with a specific domain.
Trigger(s):	Development Team determines Locally Developed Application offers services that could be leveraged in a given domain or enterprise.
Pre-Conditions:	Locally Developed Application Information must preexist in repository. Specific domain record exists in the repository.
Output:	Locally Developed Application information is associated with specific domain.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Locally Developed Application record is identified.
2	Domain is selected for application association.
3	Developer defines Locally Developed Application associated with specific domain.
NOTES:	

Operations Functions Use Cases

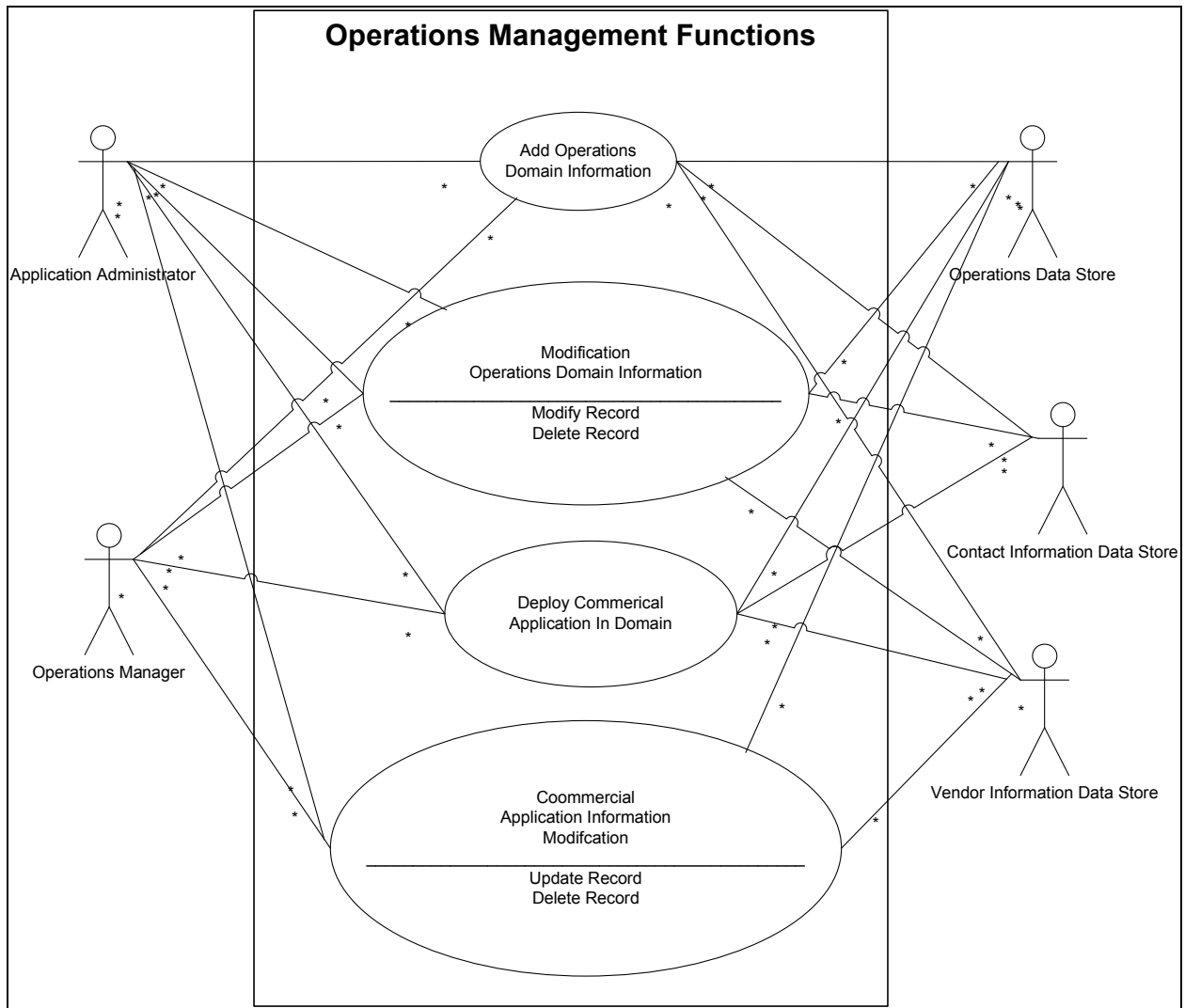


Figure 16: Operation Functions (Use Case Diagram)

Domain Management

Use Case Name:	Add Domain
Actor(s):	Operations
Goal:	Instantiate New Domain
Trigger(s):	Operations Team determines that a new domain is required to support the logical operations environment.
Pre-Conditions:	Domain is researched, clearing defined, and the Operations Team has determined this domain adds value to the business enterprise.
Output:	Domain defined and placed into the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Domain is identified at the conceptual level.
2	Operations Team defines the specifications for the new domain.
3	Operations Team and SPTOF governance board authorizes the domain for realization.
4	Operations Team inputs the domain information into repository.
NOTES: The new Domain realization must be communicated to SPTOF community.	

Use Case Name:	Modification Domain Information
Actor(s):	Operations
Goal:	Update Domain Information input into repository.
Trigger(s):	Operations Team determines that newer domain information is required to support the logical operations domain.
Pre-Conditions:	Domain modification information clearly defined and the Operations Team determines that the domain information adds value to the environment.
Output:	Domain information is defined and placed into the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Domain update information is identified at the conceptual level.
2	Operations Team defines the update specifications for the domain.
3	Operations Team and SPTOF governance board authorizes the domain update for input.
4	Operations input the domain into repository.
NOTES:	

Use Case Name:	Delete Domain Information
Actor(s):	Operations
Goal:	Delete Domain Information.
Trigger(s):	Operations Team determines that existing domain information is no longer wanted or needed to support the operations environment.
Pre-Conditions:	Domain targeted is clearly defined and the Operations Team body determines that the domain information may be removed from the environment.
Output:	Domain information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Domain is identified for deletion.
2	Operations Team and SPTOF governance board authorizes the domain for deletion.
3	The Operation team identifies and validates the domain for deletion.
4	Repository validates the deletion.
NOTES:	

Use Case Name:	Single Domain Information Report
Actor(s):	Operations
Goal:	Facilitate a report of single Domain information.
Trigger(s):	SPTOF User requires a quick view of Domain information.
Pre-Conditions:	Domain information must preexist in repository.
Output:	Single Domain information is reported based on record from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Domain record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report formation should contain the following minimum detail: <ul style="list-style-type: none"> • Domain Name • Domain Description • Application Deployment 	

Operations Request

Use Case Name:	Operations Request Facility
Actor(s):	Operations
Goal:	Provide a means for Operations team to request new services or application implementation.
Trigger(s):	Operations Team identifies a missing component or applications installation gap in a given domain.
Pre-Conditions:	The domain must exist or be defined by the operations team.
Output:	Completed request form is sent to the SPTOF change control board.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Operations Team defines a missing component or applications installation gap in support of a specific domain.
2	Operations Team defines the problem in a captured environment (a means of collecting the data).
3	Operations Team submits the information captured to the SPTOF change control board.
NOTES: This process will happen in large chunks of operations functionality. The end user will not be tasked with putting each record into the repository individually.	

Commercial Application Management

Use Case Name:	Add Commercial Application Information
Actor(s):	Operations
Goal:	Instantiate New Record for Commercial Application
Trigger(s):	Operations Team determines that a Commercial Application will provide new services suitable for consumption.
Pre-Conditions:	All services provided by a Commercial Application are identified.
Output:	New Record for Commercial Application placed into the repository to be leveraged throughout a specific domain or the entire enterprise.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	All services provided by a Commercial Application are identified and categorized.
2	Operations team decides on the validity of the application to provide reusable services.
4	Commercial Application Information is input into repository.
NOTES: The Domain is the base component for the Domain / Category / Service relationship. The communications of the completion of this domain data input must be communicated to the SPTOF community.	

Use Case Name:	Modification of Commercial Application Information
Actor(s):	Operations
Goal:	Update Record for Commercial Application
Trigger(s):	Operations Team determines that new Commercial Application information is required that will accurately depict the application or service(s) it provides.
Pre-Conditions:	Commercial Application information is clearly defined and the Operations body determines that the domain information adds values to the environment.
Output:	Updated information about a Commercial Application placed into the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	All services provided by a Commercial Application are identified and categorized.
2	The Operations team decides on the validity of the application to provide reusable services.
3	Commercial Application Information is input into repository.
NOTES:	

Use Case Name:	Delete Commercial Application Information
Actor(s):	Operations
Goal:	Delete Commercial Application Information
Trigger(s):	Operations Team determines that the existing Developed Application Information is no longer wanted or needed to support the Operations infrastructure.
Pre-Conditions:	Commercial Application targeted is clearly defined and the Operations body determines that the domain information may be removed from the environment.
Output:	Commercial Application information is removed from the repository.
MAIN SCENARIO <u>[All tasks and activities involved with accomplishing the stated Goal]</u>	
1	Commercial Application is identified as a candidate for deletion.
2	The Operations team authorizes the domain for deletion.
3	The Operations team identifies and validates the domain for deletion.
4	Repository system validates the deletion.
NOTES:	

Use Case Name:	Single Commercial Application Information Report
Actor(s):	Operations
Goal:	Facilitate a report of single Commercial Application Information.
Trigger(s):	SPTOF User requires a quick view of Commercial Application Information.
Pre-Conditions:	Commercial Application Information must preexist in repository.
Output:	Single Commercial Application information is reported based on record from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Commercial Application record is identified.
2	Report is generated based on the specification found in the notes.
NOTES: Report formation should contain the following minimum detail: <ul style="list-style-type: none"> • Commercial Application Name • Commercial Application Revision • Commercial Application Description 	

Use Case Name:	Deploy Commercial Application Information in Specific Domain
Actor(s):	Operations
Goal:	Associate a given Commercial Application Information with a specific domain.
Trigger(s):	The Operations team determines Commercial Application offers services that could be leveraged in a given domain or enterprise.
Pre-Conditions:	Commercial Application Information must preexist in repository. Specific domain record exists in the repository.
Output:	Commercial Application information is associated with specific domain.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Commercial Application record is identified.
2	Domain is selected for application association.
3	The Operations team defines a Commercial Application association with specific domain.
NOTES:	

Reporting Functions Use Cases

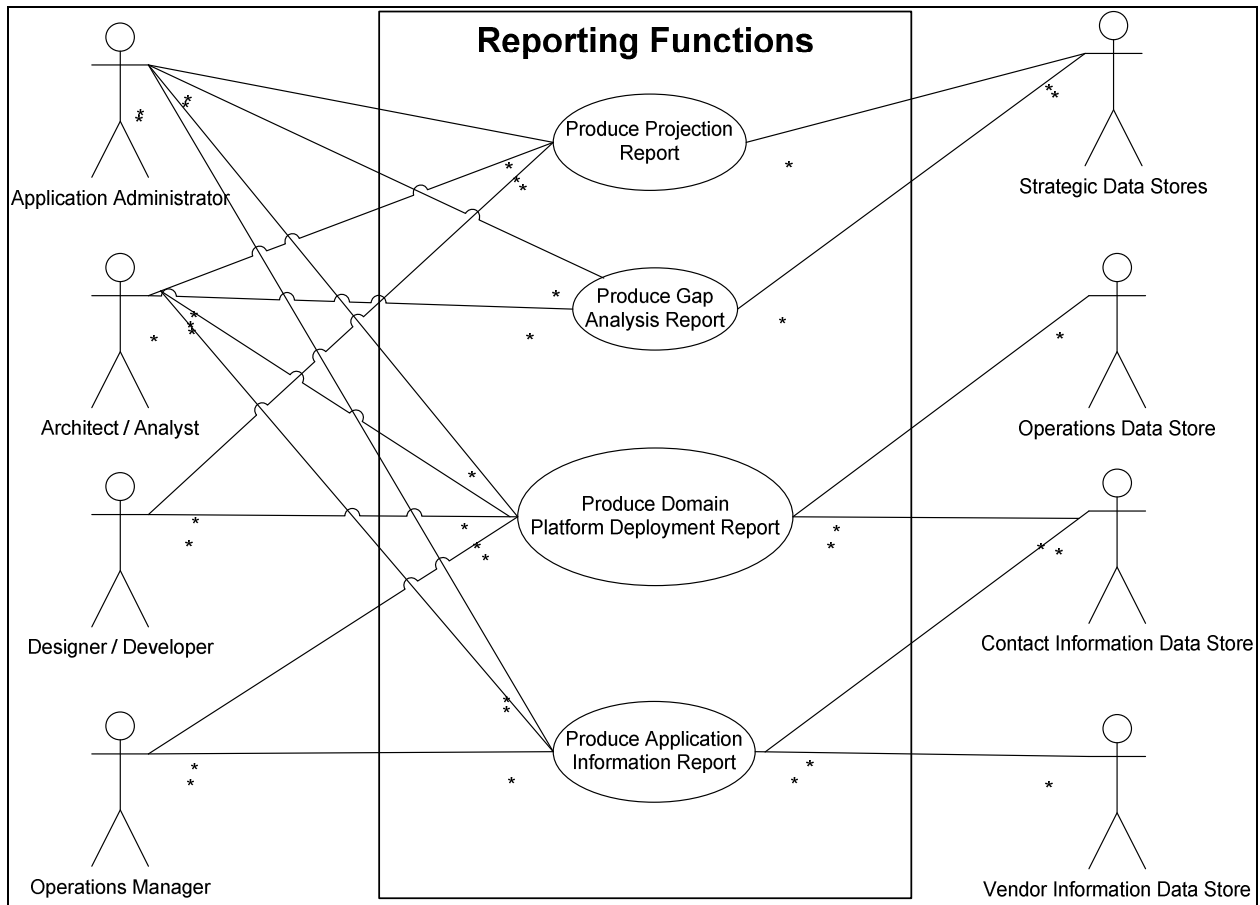


Figure 17: Reporting Function (Use Case Diagram)

Use Case Name:	Projection Report
Actor(s):	Architects and Developers
Goal:	Facilitate a report of single Architectural Layer, Categories and Service records projected into a specific domain.
Trigger(s):	Actor requires a quick report of projected Architectural Layer, Categories and Service records for a specific domain.
Pre-Conditions:	Domain projection information must exist.
Output:	Report of single Architectural Layer, Categories and Service record projected into a specific domain based on data from the repository.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Domain for projection is identified.
2	Report is generated based on the domain specification.
NOTES: Report format will need to contain at minimum the following detail: <ul style="list-style-type: none"> • Domain Name • Domain Description • Architectural Layer, Categories and Service record 	

Use Case Name:	Gap Analysis Report
Actor(s):	Architects, Developers and Operations
Goal:	Facilitate a report of the gap that exists between Architectural Layer, Categories and Service record projection and the physical application deployment record for the specific domain.
Trigger(s):	Actor requires a report of the gap analysis for a specific domain.
Pre-Conditions:	Domain projection and deployment information must exist.
Output:	A report of the gap that exists between the Architectural Layer, Categories and Service record projection and the physical application deployment record for the specific domain.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Domain for projection is identified.
2	Report is generated based on the domain specification.
NOTES: Report format will need to contain at minimum the following detail: <ul style="list-style-type: none"> • Domain Name • Domain Description • Architectural Layer, Categories and Service record • The Application that facilitates the Architectural Layer, Categories and Service specification • If there is no deployment specification, the report should identify this record as having a gap 	

Use Case Name:	Domain Application Deployment Report
Actor(s):	Developers and Operations
Goal:	Facilitate a report of the physical application deployment record for a specific domain.
Trigger(s):	Actor requires a report of the physical application deployment records for a specific domain.
Pre-Conditions:	Domain deployment record information must exist.
Output:	Report of the physical application deployment records for a specific domain.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Domain for deployment studied is identified.
2	Report is generated based on the domain specification.
NOTES: Report format will need to contain at minimum the following detail: <ul style="list-style-type: none"> • Domain Name • Domain Description • Application deployment specification report 	

Use Case Name:	Application Specification Report
Actor(s):	Developers and Operations
Goal:	Facilitate a report of a single application specification.
Trigger(s):	Actor requires report of a single application specification.
Pre-Conditions:	Application specification record information must exist.
Output:	Report of a single application specification.
MAIN SCENARIO [All tasks and activities involved with accomplishing the stated Goal]	
1	Application for study is identified.
2	Report is generated based on the application specification.
NOTES: Report format will need to contain at minimum the following detail: <ul style="list-style-type: none"> • Application Name • Application Revision • Application Description • Application Vendor Specification 	

Design Supporting Framework Database

At the heart of the SPTOF framework is a relational database. The database is designed to support the strategic and operational paradigms of the framework itself. This section of the document provides the specification for the structure and content of this database.

Strategic Data Scheme

The strategic portion of the SPTOF relational database supports the three primary elements of the strategic framework and their inter-relationships. The three main table elements include:

- Architectural Layers
- Categories
- Services

The architectural-layers and categories are supported by a single translation table. This translation is used to support the relationship between the services and architectural layers and categories.

A transition table in the strategic portion of the database supports the architectural projections functionality of the framework. This table supports the relation between the architectural layers, categories, services and the domain on which they are projected.

Operational Data Scheme

The operational portion of the SPTOF relational database supports the two primary elements of the operations functions for the framework. These two table elements support:

- Application Specification and Management table
 - Domain Specification and Management table
-

The application relationship to the services is supported by the prototype application uses a single translation table scheme. Another transition table captures the application deployment relationship to a domain deployment specification.

External Supporting Data Scheme

Three external tables are used in the prototype implementation of the SPTOF framework to support the prototype application. These tables support the major functional aspects of the application specification. The contact table supports the contact information requirements of the application and vendor information relationships.

The vendor table supports the minimal information requirements of vendor management for the SPTOF framework support application. Contact information is referenced from the contacts table. The vendor information is leveraged by the application table found in the operation data scheme.

The final table in the external support data scheme is the access table. This table supports the security functionality portion of the SPTOF framework. The access table leverages contact information referenced from the contacts table.⁷

An overview of the SPTOF framework, RDBMS (Rational Database Management System), is depicted in the following entity relation diagram. The diagram provides an overall view of the database and its internal structure.

⁷ ***Note: These three information sources will be replaced by commercial and existing data source services in the final production implementation. The current implementation tables will be discontinued at that time.***

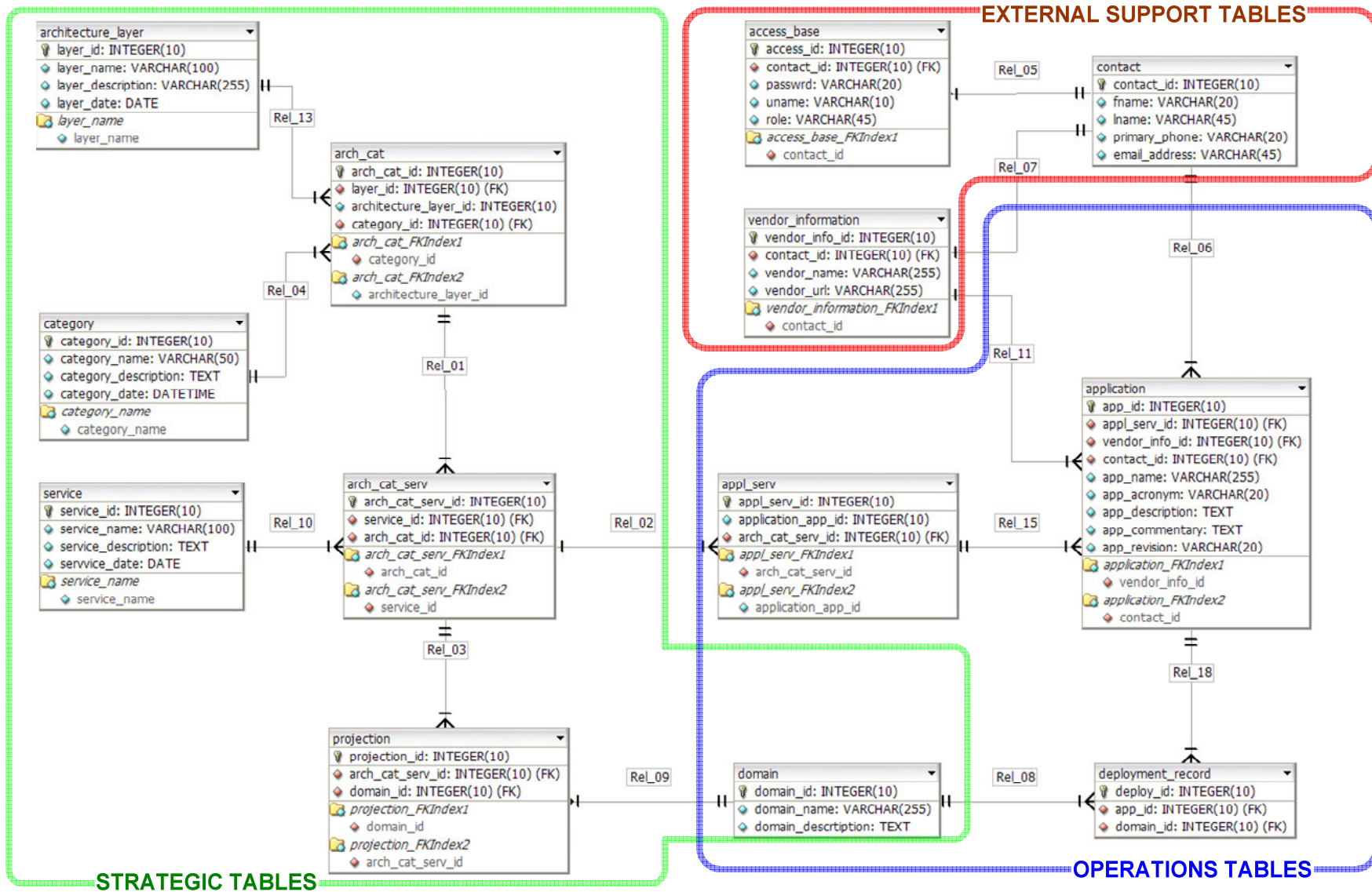


Figure 18: SPTOF (Entity Relationship Diagram)

SPTOF DATABASE TABLE SPECIFICATIONS

This section provides documentation for individual tables of the SPTOF framework relational database. Each table specification contains a table description, column specification, and primary and foreign key indicators. All tables have auto-incremented primary keys. None of the tables contain smart key associations at this time.

Table 4: SPTOF Database Tables Specifications

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
access_id	INTEGER(10)	PK	NN	UNSIGNED		
contact_id	INTEGER(10)		NN	UNSIGNED		
passwd	VARCHAR(20)		NN			
uname	VARCHAR(10)		NN			
role	VARCHAR(45)		NN			
IndexName	IndexType		Columns			
PRIMARY	PRIMARY		access_id			
access_base_FKIndex1	Index		contact_id			

appl_serv

Each application facilitates one or more services within the confines of this database. There are applications in real-world implementations that are self-contained and provide no external services. This database is focused on those applications that facilitate services that can be leveraged in real-world architectures and design. This table associates a specific application with one or more services.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
appl_serv_id	INTEGER(10)	PK	NN	UNSIGNED		
application_app_id	INTEGER(10)		NN	UNSIGNED		
arch_cat_serv_id	INTEGER(10)		NN	UNSIGNED		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	appl_serv_id
appl_serv_FKIndex1	Index	arch_cat_serv_id
appl_serv_FKIndex2	Index	application_app_id

application

Each application facilitates one or more services within the confines of this database. There are applications in real-world implementations that are self-contained and provide no external services. This database is focused on those applications that facilitate services that can be leveraged in real-world architectures and design. The application information is limited to bare bones specifications for this prototype.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
app_id	INTEGER(10)	PK	NN	UNSIGNED		
appl_serv_id	INTEGER(10)		NN	UNSIGNED		
vendor_info_id	INTEGER(10)		NN	UNSIGNED		
contact_id	INTEGER(10)		NN	UNSIGNED		
app_name	VARCHAR(255)		NN			
app_acronym	VARCHAR(20)		NN			
app_description	TEXT		NN			
app_commentary	TEXT		NN			
app_revision	VARCHAR(20)		NN			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	app_id
application_FKIndex1	Index	vendor_info_id
application_FKIndex2	Index	contact_id

arch_cat

This table serves as an association table for the architectural layers and category data.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
arch_cat_id	INTEGER(10)	PK	NN	UNSIGNED		
layer_id	INTEGER(10)		NN	UNSIGNED		
architecture_layer_id	INTEGER(10)		NN	UNSIGNED		
category_id	INTEGER(10)		NN	UNSIGNED		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	arch_cat_id
arch_cat_FKIndex1	Index	category_id
arch_cat_FKIndex2	Index	architecture_layer_id

arch_cat_serv

The SPTOF framework provides the flexibility to allow the larger chunks of strategic information to be associated or disassociated with an architectural layer. As with most flexible implementations, broad brush movement of data poses some risk. Any category association relationship should be analyzed carefully prior to modification. As with categories, services can enjoy the benefits of this flexible association paradigm. Services occur at a far more granular level of implementation. A single service can be associated with multiple categories and architectural layers.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
arch_cat_serv_id	INTEGER(10)	PK	NN	UNSIGNED		
service_id	INTEGER(10)		NN	UNSIGNED		
arch_cat_id	INTEGER(10)		NN	UNSIGNED		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	arch_cat_serv_id
arch_cat_serv_FKIndex1	Index	arch_cat_id
arch_cat_serv_FKIndex2	Index	service_id

architecture_layer

An Architectural Layer table stores data that supports a model in which each layer takes on a specific function within the system. These layers are logical components and contain no functionality in and of themselves.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
layer_id	INTEGER(10)	PK	NN	UNSIGNED		
layer_name	VARCHAR(100)		NN			
layer_description	VARCHAR(255)		NN			
layer_date	DATE		NN		0000-00-00	

IndexName	IndexType	Columns
PRIMARY	PRIMARY	layer_id
layer_name	Index	layer_name

category

A category provides an intermediate container for services. Storage of services directly in the architectural layer does not provide for logical categorization of services. As services accumulate within the framework categories, they provide a necessary level of stratification to facilitate management and reporting frameworks.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
category_id	INTEGER(10)	PK	NN	UNSIGNED		
category_name	VARCHAR(50)		NN			
category_description	TEXT		NN			
category_date	DATETIME		NN		0000-00-00 00:00:00	

IndexName	IndexType	Columns
PRIMARY	PRIMARY	category_id
category_name	Index	category_name

contact

The basic premise behind this table is to facilitate contact information. In larger implementations, this could be facilitated by an external link to a corporate electronic phone book or directory.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
contact_id	INTEGER(10)	PK	NN	UNSIGNED		
fname	VARCHAR(20)		NN			
lname	VARCHAR(45)		NN			
primary_phone	VARCHAR(20)		NN			
email_address	VARCHAR(45)		NN			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	contact_id

deployment_record

The deployment of specific applications in a given domain facilitates the understanding of what services exist for that specific domain. The precise inventory of what services are deployed in a given domain is an operational function. The differences between the projected services requirements and the physical deployment constitute a gap analysis study of those required services missing from the domain.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
deploy_id	INTEGER(10)	PK	NN	UNSIGNED		
app_id	INTEGER(10)		NN	UNSIGNED		
domain_id	INTEGER(10)		NN	UNSIGNED		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	deploy_id

domain

A domain is a conceptual container that defines a logical grouping of platforms or components that make up a hosting environment. Domains provide an anchor point for service projections and application deployment specifications.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
domain_id	INTEGER(10)	PK	NN	UNSIGNED		
domain_name	VARCHAR(255)		NN			
domain_descrtiption	TEXT		NN			
IndexName	IndexType	Columns				
PRIMARY	PRIMARY	domain_id				

projection

This table supports the relationship of strategic projection providing a vehicle for the architecture team to define a projected specification of which services should be deployed in a specific domain. The determination of which services are required is the responsibility of the architecture team in conjunction with the IT strategy publication. The association of services determines the inheritance of architectural layers and categories that are represented in this view of the domain.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	AutoInc
projection_id	INTEGER(10)	PK	NN	UNSIGNED		
arch_cat_serv_id	INTEGER(10)		NN	UNSIGNED		
domain_id	INTEGER(10)		NN	UNSIGNED		
IndexName	IndexType	Columns				
PRIMARY	PRIMARY	projection_id				
projection_FKIndex1	Index	domain_id				
projection_FKIndex2	Index	arch_cat_serv_id				

service

This component of the SPTOF framework is the essential element in generating strategic and tactical perspectives. A service is a software component that can be used as a part of a single application implementation or as a part of the overall business process. A service encapsulates its own state and the business data it is processing.

ColumnName	Data Type	Primary Key	NotNull	Flags	Default Value	AutoInc
service_id	INTEGER(10)	PK	NN	UNSIGNED		
service_name	VARCHAR(100)		NN			
service_description	TEXT		NN			
service_date	DATE		NN		0000-00-00	

IndexName	IndexType	Columns
PRIMARY	PRIMARY	service_id
service_name	Index	service_name

vendor_information

Vendor information is an essential component to this data. In many cases the indicator of which vendor's product has been deployed will also provide quick relational clues to the products capabilities. The vendor information can also be leveraged in the architectural or engineering investigations. For the purposes of the prototype, the address and other detail identification have been intentionally ignored.

ColumnName	Data Type	Primary Key	NotNull	Flags	Default Value	AutoInc
vendor_info_id	INTEGER(10)	PK	NN	UNSIGNED		
contact_id	INTEGER(10)		NN	UNSIGNED		
vendor_name	VARCHAR(255)		NN			
vendor_url	VARCHAR(255)		NN			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	vendor_info_id
vendor_information_FKIndex1	Index	contact_id

Functional Design for Prototype Application

The SPTOF framework prototype application is designed to support and manage a repository containing strategic and operational data. Management functions include all typical database management components of record creation, update, retrieval and deletion. In its initial prototype form, the SPTOF application supports only the client server paradigm, offering no external services for consummation.⁸ Users will establish secure browser sessions with proper identification and authentication information.

Users will be afforded access to the various operations via secured roles assigned to each user session. There are four roles identified for the user community. Each role defines the user access right within the scope of the application. The following table defines the role and its high-level function within the scope of the application.

Table 5: User Roles Definitions

Role	Function
Architect	Responsible for the management of strategic data sets including: <ul style="list-style-type: none">• Architectural Layers• Categories• Services• Domain Projections

⁸ Currently the application design does not include web services. In future versions of the production version of the application, the publication web services will be required.

Developer	Primarily a consuming role, the development role is focused on obtaining information about projected services and deployed applications within a given domain. The developer role is also assigned functionality to manage information about locally developed applications.
Operations	The operations role is assigned the primary roles of managing domain information and applications deployed in those domains. This includes: <ul style="list-style-type: none"> • Commercial Application/Middleware record management • Domain information Management • Commercial Application/Middleware deployment record management
Administrator	This role inherits all functionality associated with the three preceding roles. Additionally, these roles will be associated with user account management and application setup, startup, and shutdown.

The SPTOF framework application design is based on the N-Tier architecture paradigm. Its design is intended to abstract each supporting layer from the other layers in the design. This type of design supports component level implementation architecture, where each layer's independence affords loose coupling of design components. This loose coupling allows for maintenance or replacement of components in each architectural layer without major repercussions in any of the other layers.

The presentation layer supports web-based clients with which the application is designed to communicate. Web-based clients are stateless in design. They allow for application access from a multitude of client platforms without the overhead or tight coupling of a thick client. The only requirement of a client platform to access the SPTOF application is that client platform must

have a web browser installed that supports HTML (Hyper Text Markup Language) version 4.0 compliant content.

The Business logic layer will be hosted on a J2EE compliant middleware server. This server will support the JAVA 5.0 specification. All business logic processes are limited to and facilitated by the business logic layer. This layer will act as the intermediary component between the presentation and data access layers. This layer will support resources for security authorization and logging functionality.

The data access or persistence layer supports the access to the various data stores required by the application for operation. All current operational and external data stores are slated to be implemented on relational databases. Each data store will be supported via a JAVA data access object that will abstract low-level CRUD (Create, Read, Update, and Delete) operations from the upper layers of the application design.

Reporting functions for the initial prototype release will be limited to internal canned reports provided via the web-based implementation. Currently, the prototype will support individual asset reports and a domain gap analysis report⁹.

Aside from the previously mentioned CRUD operations, users will be offered request forms to facilitate the change management portion of the SPTOF framework specification¹⁰.

⁹ Ad hoc and complex reporting functions will be addressed by external off-the-shelf reporting tools when the production implementation is realized.

¹⁰ Request forms are emailed to a common email address. The production implement will address possibilities of address message handling via the use of external tools and email managers.

High-Level Programmatic Flow

The high-level programmatic flow of the prototype application facilitates the “Model-View-Controller” industry standard design pattern (Inderjeet Singh, Mark Johnson, & the Enterprise Team, 2002). The web-tier or presentation controller receives each incoming HTTP request and invokes the business logic layer to perform the requested operation. The results of the operation and content of the model invocation is processed into the next view to display. The controller generates the constructed view that is transmitted to the client browser for presentation. As the process request moves down the application architectural layers, the data will be validated and transformed into desired a format for storage or retrieval depending on the operation request.

All transactions will be logged to a centralized logging service. The granularity of logging is dependent on the error level of logging properties set at the application invocation and the complexity of the event or alarm occurrence.

The following diagram depicts a high-level overview of the components involved with the design of the SPTOF framework prototype application. The diagram will depict each component in relation to the architectural layer in which the component is located.

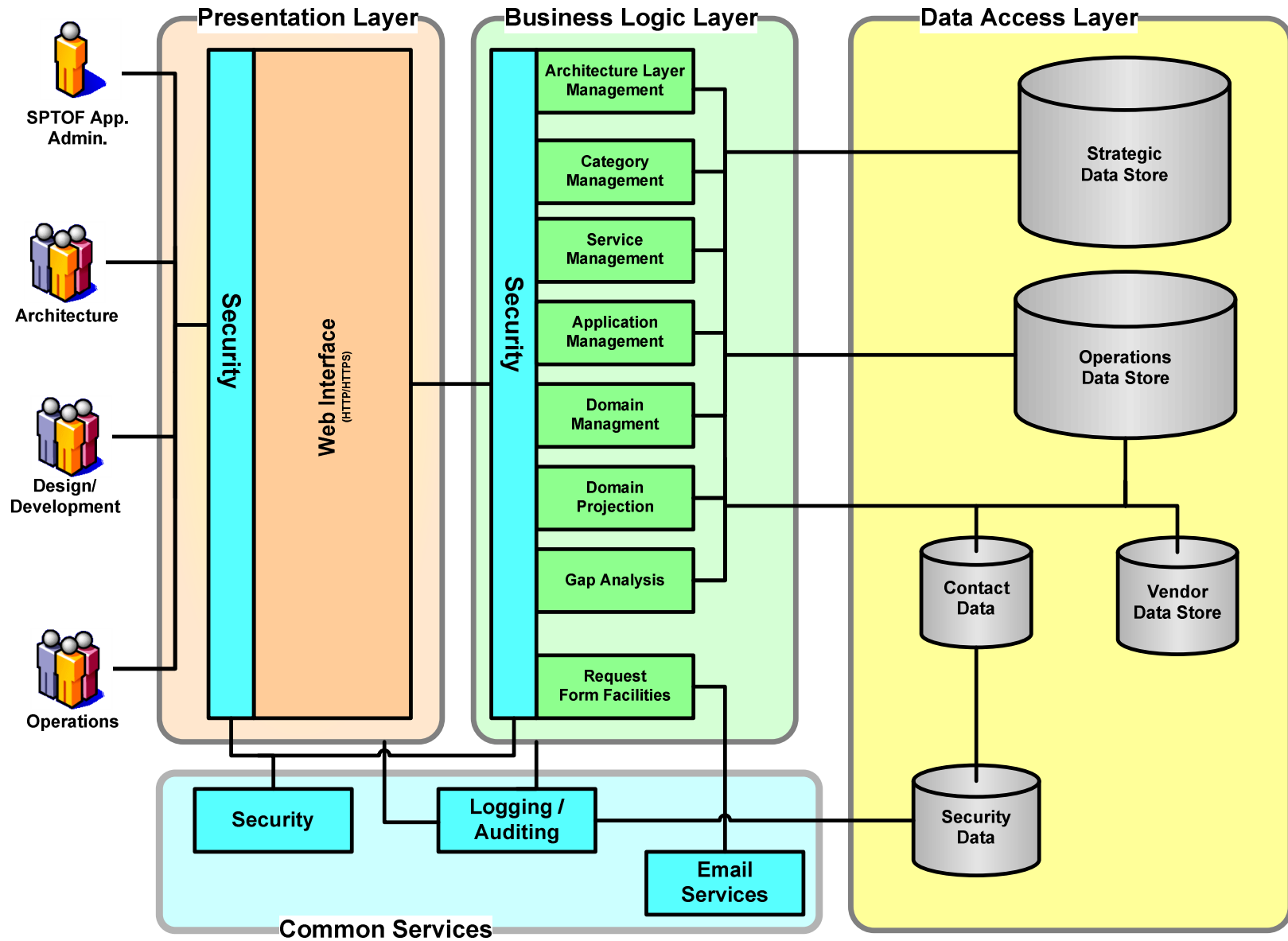


Figure 19: High-level Functional Flow Diagram

Hosting Environment Description

The hosting environment for the SPTOF framework prototype application requires three server implementations to support the proper functional environment. This section of the document describes the minimal SOE server requirements for the proper hosting and operation of the application. The following diagram depicts a high-level overview of the SOE for the SPTOF framework prototype application.

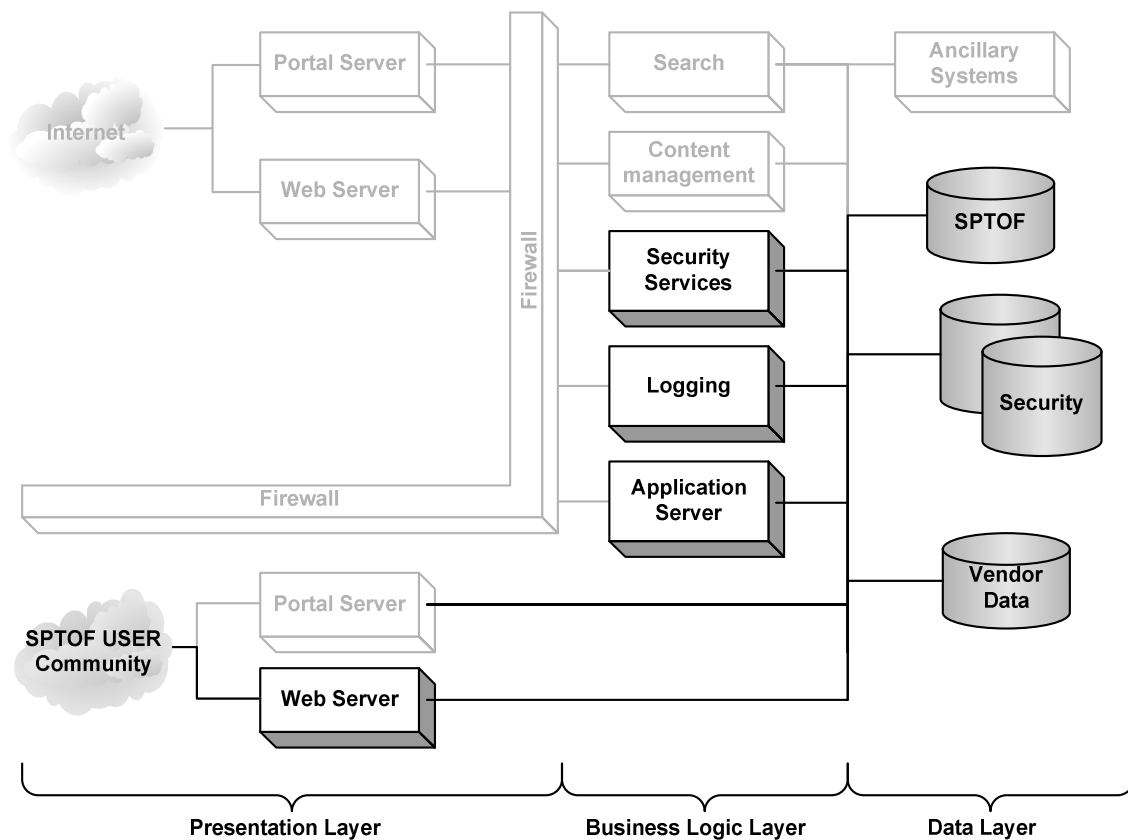


Figure 20: Hosting Environment Diagram

Those components bolded in the previous diagram depict the minimum components required for the SPTOF framework prototype application. The following table provides an inventory of the components, architectural layer, and their functional description.

Table 6: High-Level Environment Table

Architectural Layer	Component	Description
Presentation Layer	Web Server	A computer that delivers or serves up web pages content. A web page is a document created with Hyper Text Markup Language (HTML) that is part of a group of hypertext documents or resources available on the World Wide Web. Every web page is identified by a unique Uniform Resource Locator (URL).
Business Logic Layer	Application Server	The application is a JAVA/J2EE based application and requires an application server deployed that will support the JAVA/J2EE specification.
	Logging Service	Captures and persist an application transactions event messages or alerts.
	Security Services	Supports the identification, authentication, and authorization for the presentation and business logic layer.
Data Layer	SPTOF (operational data store)	Relational database hosted on a RDBMS that support JDBC (Java Database Connectors) access. All SPTOF operational and persistence data will be facilitated on this server.
	Security (data store)	Relational database hosted on a RDBMS that support JDBC (Java Database Connectors) access. Supports the identification, authentication, and authorization information persistence.
	Vendor (data store)	Relational database hosted on a RDBMS that supports JDBC (Java Database Connectors) access. Supports the vendor information persistence.

The platform for the prototype design, development and test is based on open source products. These products facilitate the component specified in the “High-Level Environment Table”. The following table provides a list of open source products that fulfill the component requirements.

Table 7: Prototyping Platform Specification

Architectural Layer	Component	Open Source Product(s)
Presentation Layer	Web Server	<ul style="list-style-type: none"> • JBOSS 4.0.1 (Jboss Application, 2005) • JSTL 2.0 (Jakarta Taglibs, 2002) • Ditchnet JSP Tabs (Todd Ditchendorf, 2005)
Business Logic Layer	Application Server	<ul style="list-style-type: none"> • JBOSS 4.0.1 (Jboss Application, 2005)
	Logging Service	<ul style="list-style-type: none"> • Log4J 4.1 (Log4j Logging, 2003)
	Security Services	<ul style="list-style-type: none"> • JBOSS 4.0.1 (Jboss Application, 2005)
Data Layer	SPTOF (operational data store)	<ul style="list-style-type: none"> • MySQL RDBMS 4.1.2 (Mysql Database, 2002)
	Security (data store)	<ul style="list-style-type: none"> • MySQL RDBMS 4.1.2 (Mysql Database, 2002)
	Vendor (data store)	<ul style="list-style-type: none"> • MySQL RDBMS 4.1.2 (Mysql Database, 2002)

High-Level Design Diagrams

The SPTOF framework prototype application is a JAVA/J2EE based application. This section facilitates a high-level view of the packages and classes that compose the application. This application is segregated into JAVA class packages, which contain various JAVA classes implemented to facilitate application functionality. The following diagram provides a high-level class diagram view of the packages designed to support the N-Tier design of the application. The diagram aligns the N-Tier design specification previously presented in this section.

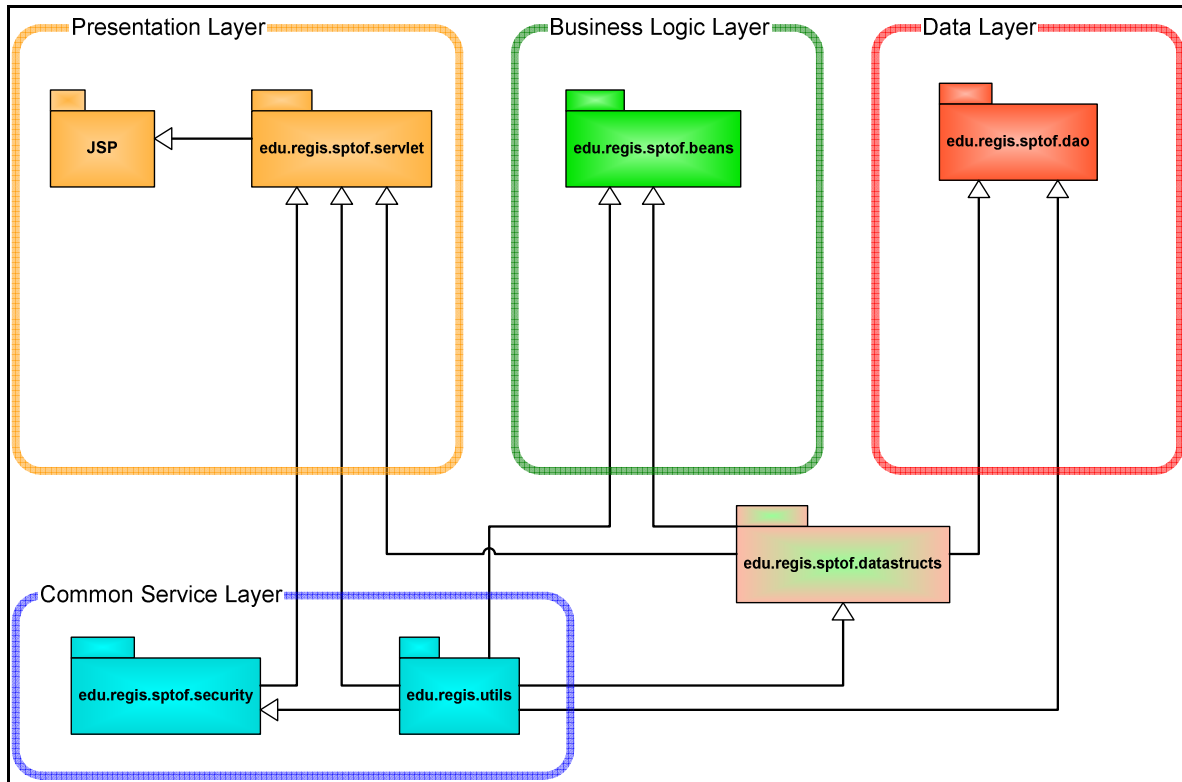


Figure 21: High-Level Class Diagram

As depicted in the previous diagram the presentation layer is supported by JSP (JAVA Server Pages) and servlets deployed in the “edu.regis.sptof.servlet” packages. The business logic is supported in the “edu.regis.sptof.beans” package. The access to data stores is facilitated by the “edu.regis.sptof.dao” and “edu.regis.sptof.datastruct” packages. Security and various utility elements are supported by the “edu.regis.sptof.security” and “edu.regis.sptof.datastructs” packages. The subsequent sections will provide an alphabetically sorted inventory of packages deployed to support the SPTOF application. Each package is documented with their relative function within the scope of the application.

Package Description: edu.regis.sptof.beans

This package supports all business logic for the application. Collectively, the classes in this package control the processing, transformation, and collation of SPTOF framework data. The

class specifications are aligned with the strategic, operational, and tactical paradigms of the framework. The primary function for the classes in this package is to act as a middleware conduit between the presentation layer and data access layer.

Package Description: edu.regis.sptof.dao

The DAO (Data Access Object) supports the “factory design pattern” (Erich Gamma, 1994) of implementing just-in-time transactional objects that contain data to be stored or retrieved from the RDBMS that supports the SPTOF application. These classes are intended to abstract the business logic classes away from being tightly coupled with the database.

Package Description: edu.regis.sptof.datastructs

The datastructs package is the contract or the construction component of the “factory design pattern” (Erich Gamma, 1994). It provides a template for the methods and objects that consume or produce data throughout the application. The class specifications are aligned with the strategic, operational, and tactical paradigms of the framework. Additionally, the security, contact, and vendor data structures are implemented to allow for their replacement when the application moves out of the prototyping phase and into production.

Package Description: edu.regis.sptof.event

This package supports the elements of navigation components for JSP and servlet operations. Providing abstract navigation allows for new JSP and servlets to be introduced into the application without significant modification to the application structure. There are property-files associated with the operation of these navigational components.

Package Description: edu.regis.sptof.security

The security package is a temporary component of the SPTOF framework application prototype. A commercial security package will be deployed in the final production version. This package has been deployed to support the portability of the application. This package supports the identification, authentication, and authorization operations for the presentation and business logic layers in the application. User credentials are populated after proper authentication has been completed.

Package Description: edu.regis.sptof.servlet

The servlets classes act as the proxy between the presentation layer and the bean classes in the business logic layer. The servlet class methods adhere to the servlet requirements prescribed by the JAVA/J2EE specification. The servlet classes have been extended to include session clearing and security methods.

Package Description: edu.regis.utils

The utils package contains classes that facilitate various utilitarian classes leveraged as common services throughout the application. Some of the notable classes include:

- Logging
- Exception Handling
- Property File Management
- String Management
- Tree Map Extensions
- Alerts Management

These utility classes are not tied to the main edu.regis.sptof package to promote reusability.

Graphical User Interface Design Specification

The graphical user interface for the SPTOF prototype application is designed for web-based technologies. Since these technologies are deployed in a stateless model, round trip data validation and large data record transmissions must be accounted for in each screen designed.

In this section, the basic web site navigation models will be presented. The flow and function will be represented in hierarchical diagrams. The site is designed around the role functions discussed in the “*Functional Design for Prototype Application*” section of this document.

The section that follows contains the web site flow presented in the form wireframe mock-ups. These wireframe diagrams represent the proposed web interface screens that will be realized in the prototype application. Each screen diagram will be preceded by a brief description of form and function.

Web Site Flow

The SPTOF application web site design contains a basic security model. Under this security model, no functional page access is granted until the user has properly authenticated the session.

The session must be associated with the proper security and role credentials prior to any interaction with the application. Any attempt to directly access a page prior to proper login operation will result in the session being redirected to the login page.

The web site consists of six primary page realms that support the three disciplines. Two generic pages support gap analysis reporting and request form functionality. The following diagram depicts the top-level layout for the application’s web site.

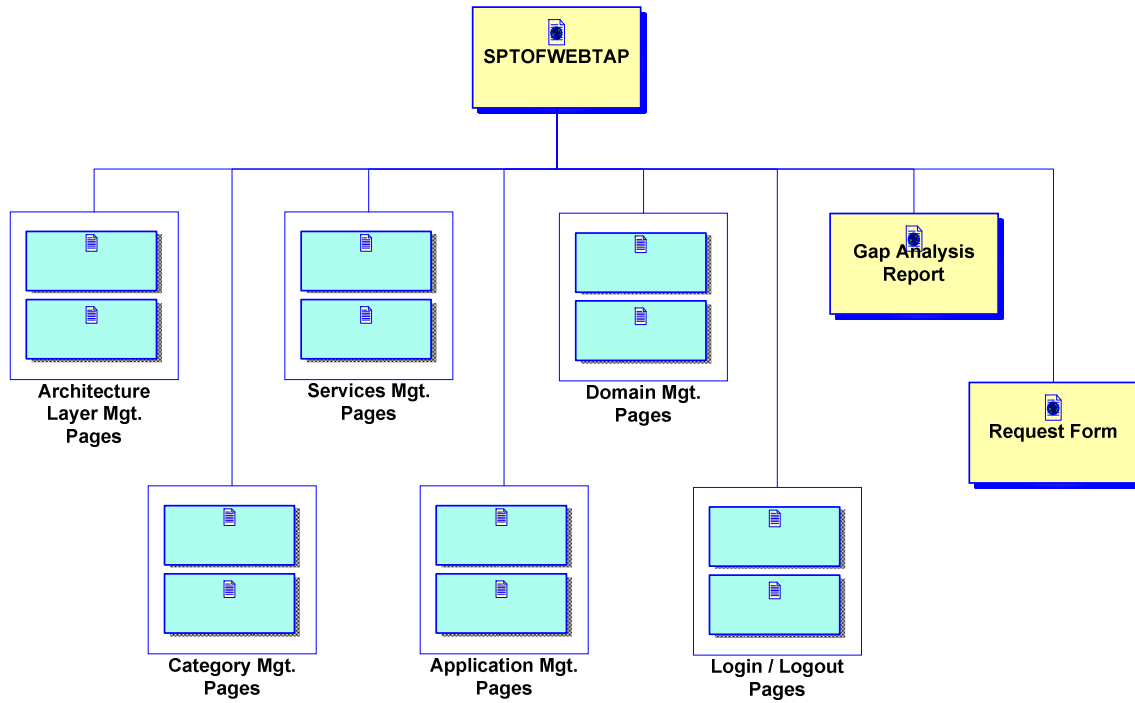


Figure 22: SPTOF Application Web Site Topology Diagram

Architects' Web Flow

The architects' web site flow supports the basic strategic information management requirements of the project. As depicted in the following web page hierarchy, the architect role will access to strategic management assets.

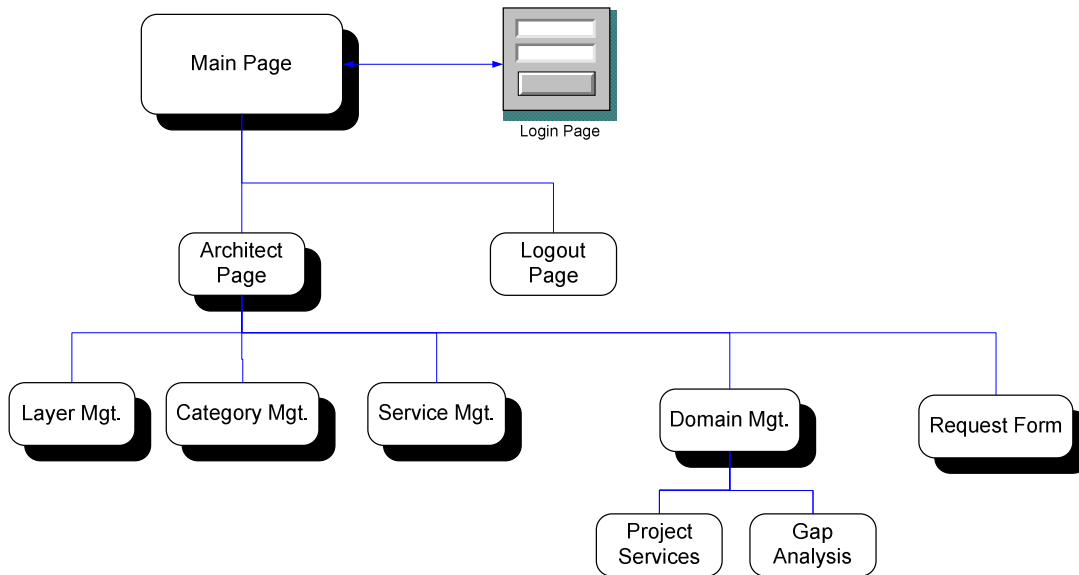


Figure 23: Architects' Web Site Flow Diagram

Additionally, the role will have limited access to domain management pages. Change request functionality is supported via the “Request Form” page.

Developer's Web Flow

The developer's web site flow supports the primary elements of tactical information consumption via the gap analysis reporting function. Developers will be provided with tools to manage locally implemented application assets.

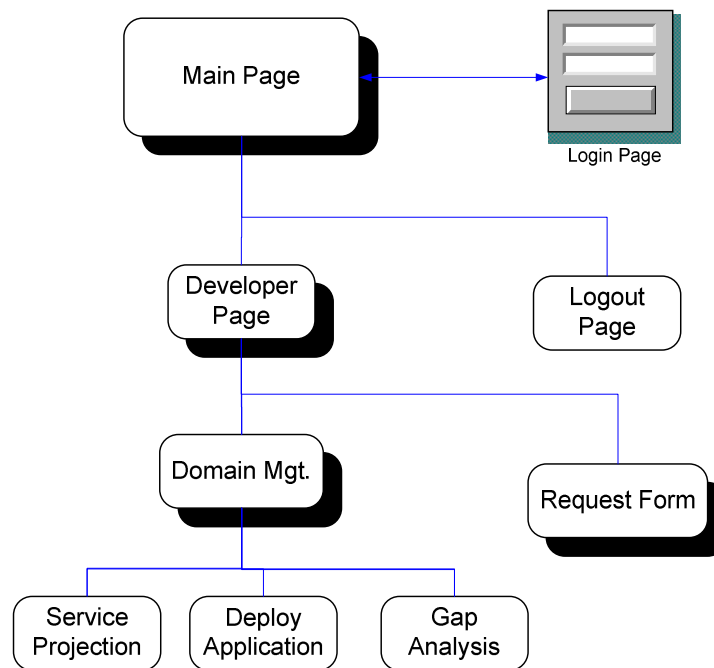


Figure 24: Developers' Web Site Flow Diagram

As depicted in the previous diagram, developers will be provided with application management tools and limited access to domain management pages. Change request functionality is supported via the “Request Form” page.

Operations Web Flow

The operations team web site flow supports aspects on operational information management including:

- Domain Management
- Commercial Application Deployment Management

As depicted in the following web page hierarchy, the operations role is afforded access to operations management assets.

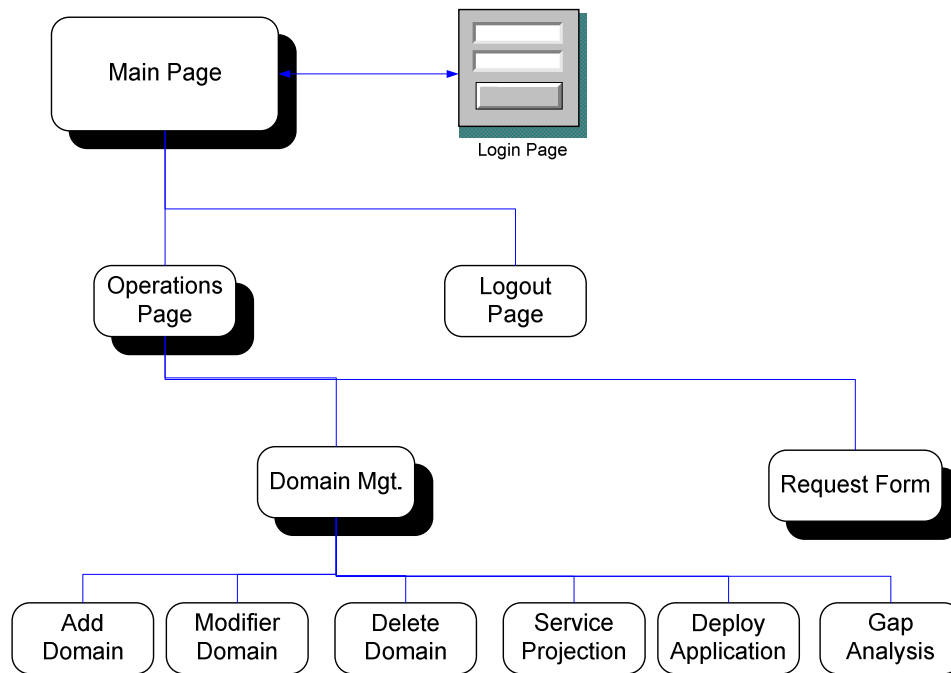


Figure 25: Operations Web Site Flow Diagram

The Gap Analysis page is provided to the operations team to facilitate knowledge about service and applications gaps that exist in the environments, which they manage. Change request functionality is supported via the “Request Form” page.

SPTOF Web Site Wireframe Diagrams and Narratives

The support for the layout of each page is based on the general functional flow of the SPTOF framework prototype application. This section provides wireframe diagrams that depict the project content forms and linking components for the application pages.

Page formats consist of eleven page types. The inventory of page type wireframes for this project includes:

3 * Main Page Types

4 * Asset Management Page Types

1 * Domain Project Page

1* Gap Analysis Report Page

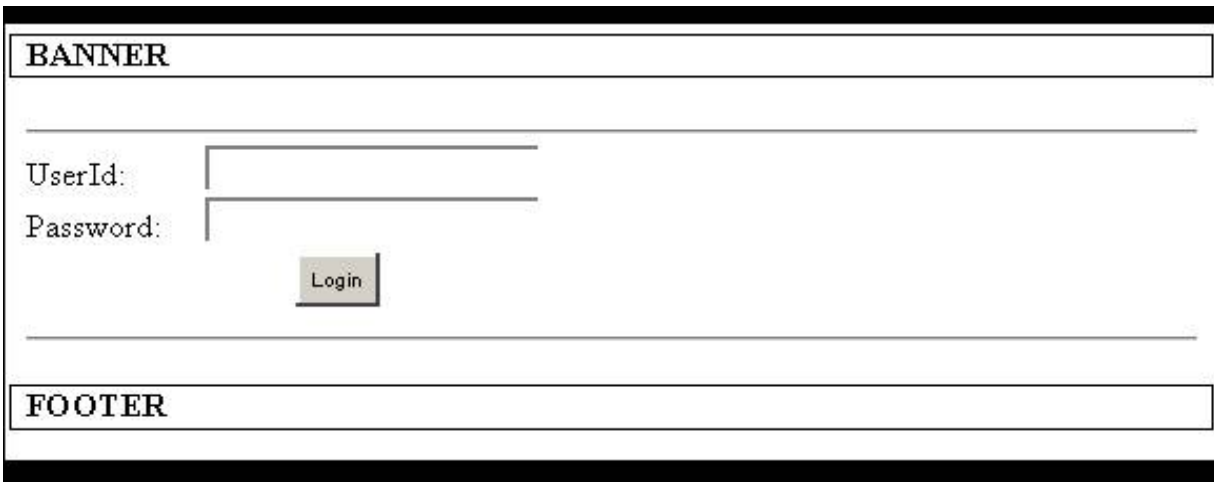
1 * Request Form

1 * Login Form

All pages will include a banner section and footer section to standardize the look-and-feel and to facilitate the standard navigation model for the web site.

Login Page Wireframe

This page is presented to the all roles as a login challenge. User must supply proper user credentials to gain access to the SPTOF framework support application. User identification and credentialing is completed via a round trip to the server at which time the user's identifier and password combination is validated and credentials are assigned to the session.

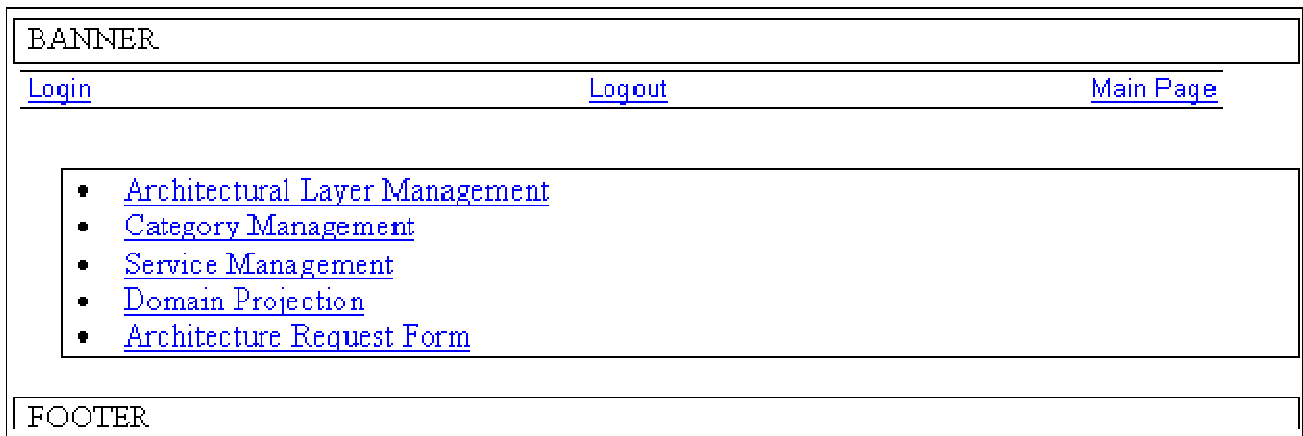


The wireframe shows a page layout with a 'BANNER' section at the top and a 'FOOTER' section at the bottom. In the center, there is a login form consisting of two input fields: 'UserId:' and 'Password:'. A 'Login' button is positioned below the 'Password:' field.

Figure 26: Login Page Wireframe

Architecture Main Page Wireframe

This page is presented to the architect role upon login. It serves as the home page for all architecture and strategic functions.



The wireframe shows a page layout with a 'BANNER' section at the top and a 'FOOTER' section at the bottom. Below the banner, there are three links: 'Login', 'Logout', and 'Main Page'. Below these links, there is a list of bulleted links: 'Architectural Layer Management', 'Category Management', 'Service Management', 'Domain Projection', and 'Architecture Request Form'.

Figure 27: Architects' Main Page Wireframe Diagram

To navigate to the various functions, the user clicks on the bulleted link. The “Main Page” link will redirect the users’ session to its parent main page. The “Login” and “Logout” links are provided for user role switch and session terminator, respectively.

Developers Main Page Wireframe

This page is presented to the developer upon login. It serves as the home page for all developer and tactical functions.

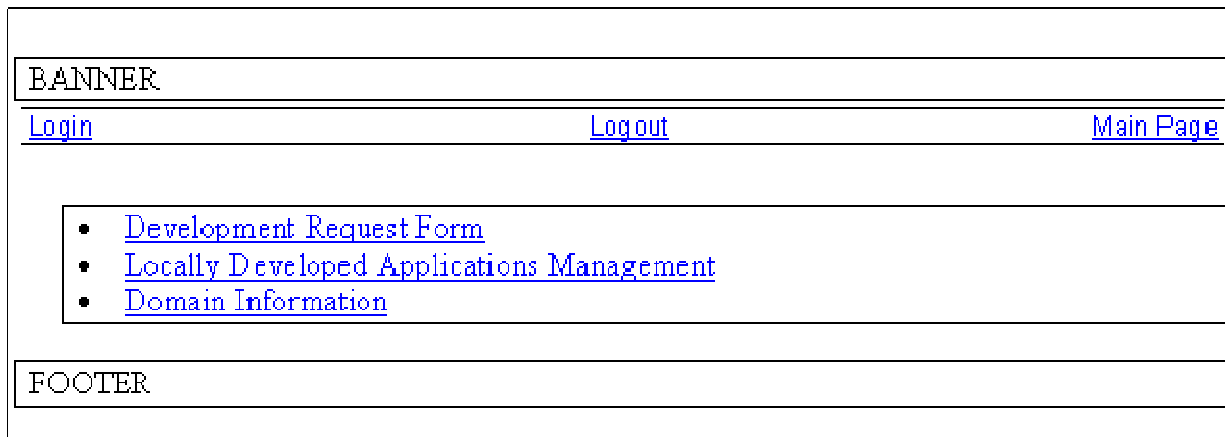


Figure 28: Developers' Main Page Wireframe Diagram

To navigate to the various functions, the user clicks on the bulleted link. The “Main Page” link will redirect the users’ session to its parent main page. The “Login” and “Logout” links are provided for user role switch and session terminator, respectively.

Operations Main Page Wireframe

This page is presented to the operations role upon login. It serves as the home page for all operations functions.

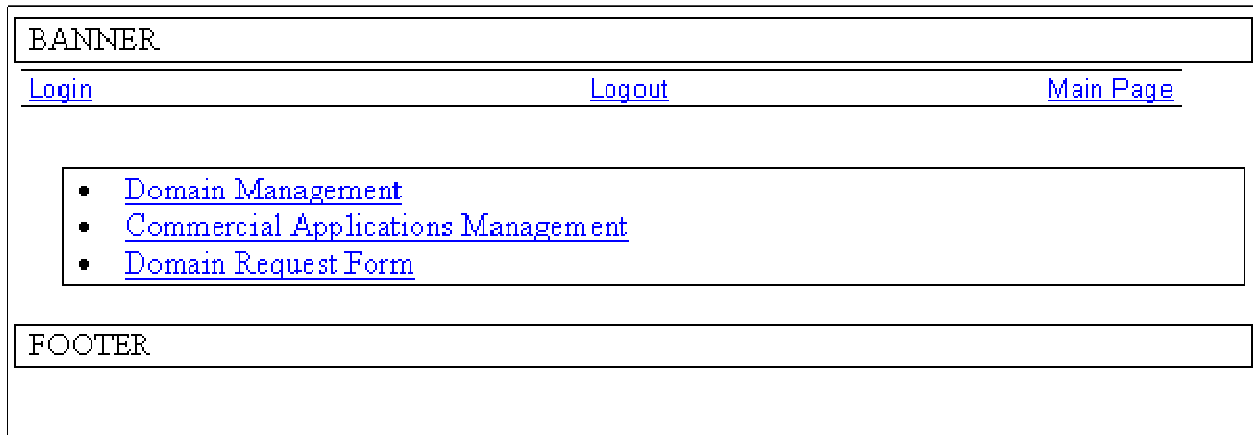


Figure 29: Operational Main Page Wireframe Diagram

To navigate to the various functions, the user clicks on the bulleted link. The “Main Page” link will redirect the users’ session to its parent main page. The “Login” and “Logout” links are provided for user role switch and session terminator, respectively.

Asset Management Main Page Wireframe

The asset management page wireframes are applicable in general forms that support the following functions:

- Architecture Layer Management
- Category Management
- Service Management
- Application Management
- Domain Management

The pages will be adjusted to reflect the customized content and data input needs of each function.

This page is presented as the top-level page for asset management functions in particular. The basic functions of data management are supported via the various submit buttons.

BANNER													
Login	Logout	Main Page											
<input type="button" value="NEW"/>	<input type="button" value="MODIFY"/>	<input type="button" value="DETAILS"/>	<input type="button" value="DELETE"/>										
Asset Name: (Highlight Asset for operation)													
<table border="1"><tr><td>Application or Business Logic Layer</td><td>▲</td></tr><tr><td>Arch Cats Association Layer</td><td></td></tr><tr><td>Common Services</td><td></td></tr><tr><td>Data Layer</td><td></td></tr><tr><td>Presentation Layer</td><td>▼</td></tr></table>				Application or Business Logic Layer	▲	Arch Cats Association Layer		Common Services		Data Layer		Presentation Layer	▼
Application or Business Logic Layer	▲												
Arch Cats Association Layer													
Common Services													
Data Layer													
Presentation Layer	▼												
FOOTER													

Figure 30: Asset Management Main Page Wireframe Diagram

As depicted in the previous diagram, the “NEW” button will direct the session to a new asset input formation. The rest of the submission buttons require the user to select an asset from the asset list prior to the operation. Failure to do so will result in a visual and audible warning message.

New Asset Wireframe

This page is presented as a result of the selection of the “NEW” button from the management functions page. The page functions as a data collection page for new asset submissions. Fields will be customized to meet the individual asset type it supports.

The wireframe diagram shows a web page layout for submitting a new asset. At the top is a 'BANNER' section. Below it are two large text input fields: 'Asset Name:' and 'Description:'. Underneath these is a table with two columns: 'Available Assets' and 'Selected Assets'. The 'Available Assets' column contains a list of asset types: Content Services, Customer Information Management, DB API, DBMS, Data Warehouse, Directory Services, Distributed Messaging Services, Enterprise Portal, Integrated Voice Response (IVR), and Middleware Application Server. The 'Selected Assets' column is currently empty. Between the two columns are three arrows pointing from left to right, indicating a move function. At the bottom of the table are two buttons: 'Submit Layer' and 'Cancel'. Below the table is a 'FOOTER' section.

Figure 31: New Asset Wireframe Diagram

In some cases assets have associations with other assets in the SPTOF framework repository. The associations will be facilitated through selections lists. Referring to the previous diagram, the user will select assets from the available assets lists and move them to the selected list. These associations will be submitted to the program upon operation of the submission button.

Typically, no text input fields may be left blank on any management forms for new asset submission. Failure to complete all text fields will result in a visual and audible warning message.

Modify Asset Wireframe

This page is presented as a result of the selection of the “MODIFY” button from the asset management functions page. The page functions to present existing record information and data collection for updating asset information submissions. Fields will be customized to meet the individual asset type they support.

BANNER	
Asset Name:	Application or Business Logic layer
Description:	The application layer or business logic layer houses the application proper, this means any domain specific knowledge components needed to power the functionality of the application reside at this layer.
Available Assets	Selected Assets
<ul style="list-style-type: none"> Content Services Customer Information Management DB API DBMS Data Warehouse Directory Services Distributed Messaging Services Enterprise Portal Integrated Voice Response (IVR) Middleware Application Server 	<ul style="list-style-type: none"> Middleware Application Server
<input type="button" value="Submit Layer"/> <input type="button" value="Cancel"/>	
FOOTER	

Figure 32: Asset Modification Wireframe Diagram

In some cases assets have associations with other assets in the SPTOF framework repository. The associations will be facilitated through selections lists. Referring to the previous diagram, the user will select assets from the available assets lists and move them to the selected list. These associations will be submitted to the program upon operation of the submission button.

Typically, no text input fields may be left blank on any management forms for modified asset submission. Failure to complete all text fields will result in a visual and audible warning message.

Domain Projection Wireframe

This page is presented as a result of the “Projection” button being selected from the domain management functions page. The page functions to facilitate the associations between domains and services projected into the domain from a strategic perspective. The associations will be facilitated through selections lists. Referring to the previous diagram, the user will select assets from the available assets lists and move them to the selected list. These associations will be submitted to the program upon operation of the submission button

BANNER		
Login	Logout	Main Page
Domain Name:	Bogus Asset	
Available Architect Layer / Category / Services		
Application or Bus in / Middleware Applicati / CORBA		
Application or Bus in / Middleware Applicati / DCOM/COM Server		
Application or Bus in / Middleware Applicati / EJB		
Application or Bus in / Middleware Applicati / Failure recovery (CI		
Application or Bus in / Middleware Applicati / J2EE Connector Archi		
Application or Bus in / Middleware Applicati / JDBC (Database Acces		
Application or Bus in / Middleware Applicati / JMS (Mess aging Servi		
Application or Bus in / Middleware Applicati / JNDI LDAP		
Selected Architect Layer / Category / Services		
Application or Bus in Middleware Applicati CORBA		
Application or Bus in Middleware Applicati DCOM/COM Server		
Application or Bus in Middleware Applicati EJB		
Application or Bus in Middleware Applicati Failure recovery (CI		
Application or Bus in Middleware Applicati J2EE Connector Archi		
Application or Bus in Middleware Applicati JDBC (Database Acces		
Application or Bus in Middleware Applicati JMS (Mess aging Servi		
Application or Bus in Middleware Applicati JNDI LDAP		
<input type="button" value="Submit Projection"/>		<input type="button" value="Cancel"/>
FOOTER		

Figure 33: Domain Projection Wireframe Diagram

Gap Analysis Wireframe

This page is presented as a result of the “Gap Analysis” button being selected from the domain management functions page. The page functions to facilitate a report of the tactical view for the selected domain. This is the product of comparing the strategic projection and the actual deployment inventory. The comparison yields to different products. This is perhaps the greatest value of SPTOF framework. This report bridges the knowledge gap between strategic and operational views by creating a single tactical view. The following diagram depicts the wireframe representation of this report.

BANNER				
Login		Logout		Main Page
<input type="button" value="Return"/>				
Domain Name:		Bogus Domainus		
Architectural Layer	Category	Service	Application Specification	
			Name	Revision
Application or Business Logic layer	Middleware Application Server	CORBA	GAP: No Application Deployed	
		EJB	GAP: No Application Deployed	
		Failure recovery (Clustering)	GAP: No Application Deployed	
		JDBC (Database Access)	MySQL Community Edition 4.1.2	4.1.2
		JMS (Messaging Service)	GAP: No Application Deployed	
		Load balancing (Clustering)	GAP: No Application Deployed	
		XML	GAP: No Application Deployed	
Common Services	Directory Services	Active Directory Server	GAP: No Application Deployed	
		LDAP	GAP: No Application Deployed	
		UDDI	GAP: No Application Deployed	
		Update Service Tester 11/30/2005	GAP: No Application Deployed	
		Email	GAP: No Application Deployed	
	Reporting	Ad Hoc Reporting	GAP: No Application	

Figure 34: Gap Analysis Wireframe Diagram

Request Form Wireframe

This page is presented as a result of the “REQUEST FORM” link being selected from any role’s main page. The page functions as a data collection vehicle for any new or updated asset elements. Upon submission, the form is routed to the change control board via an email interface.

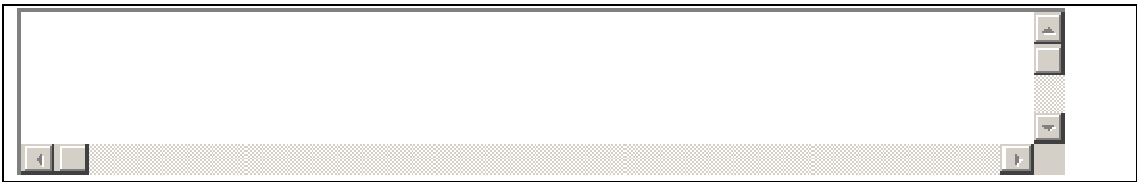
BANNER		
Login	Logout	Main Page
Developer Name: _____		
Title: _____		
<input type="text"/> First: <input type="text"/>	Last: <input type="text"/>	
Depart. Name: _____	Depart. ID: _____	Role: <input type="text"/>
Contact Information:		
Phone Number: _____	Email Address: _____	
Request Details: _____	Request Type: <input type="text"/>	
		
<input type="button" value="Submit Request"/>	<input type="button" value="Clear All Fields"/>	<input type="button" value="Cancel"/>
FOOTER		

Figure 35: Request Form Wireframe Diagram

CHAPTER VI

Review of the Deliverables

The following is a summarization of the project plan and deliverables by phase for the SPTOF project. The project plan is based on the “Schedule of Phases for the Project” found in the “Project Methodology” section of this document. Each of the phases contains a logical grouping of tasks that supports completion of that particular phase.

Table 8: Schedule of Project Phase Plan and Deliverables

Phase	Description	Work Tasks	Deliverables
I	Analysis	<ul style="list-style-type: none"> • Gather information to define problem statement • Develop problem statement • Obtain approval from stakeholder for problem statement 	<ol style="list-style-type: none"> 1. Problem statement and supporting narrative 2. Client approval of problem statements' accuracy
II	Framework Definition	<ul style="list-style-type: none"> • Define framework to address problem statement • Document framework layout and business function • Document function of framework within the scope of the enterprise 	<ol style="list-style-type: none"> 3. High-Level SPOTF framework narrative 4. Supporting diagrams for the framework narrative
III	Change Control Process Definition, Design	<ul style="list-style-type: none"> • Define purpose of change control process • Define roles within the scope of the process • Define the process flow for the change control process 	<ol style="list-style-type: none"> 5. Change control process definition 6. Role definitions 7. Change control process narrative of process steps 8. Process overview diagram

Phase	Description	Work Tasks	Deliverables
IV	Application Definition and Design	<ul style="list-style-type: none"> • Collect Use Case data from client population • Document Use Cases in text and UML format including Use Case Diagrams • Define High-Level program flows • Define repository supporting data structures • Develop Entity Relationship Diagram defining the database structure • Define User interfaces based on Use Case information • Define High-Level reporting structures • Generate programmatic flow diagram 	<ul style="list-style-type: none"> 9. Use Case diagrams 10. Use Case narratives 11. SPTOF repository database design artifacts 12. High-level program flows diagram and narrative 13. Class diagrams and supporting package descriptions 14. Application web site definition 15. Web page wireframe diagrams
V	Application Construction	<ul style="list-style-type: none"> • Generate database scheme based on ERD • Generate data layer objects based on table structures define in the database • Generate test cases for data layer objects • Generate business logic objects based on Use Case requirements • Generate test cases for business logic objects • Generate report logic objects based on Use Case requirements • Generate test cases for report logic objects • Generate User Interfaces based on Use Case requirements • Generate test cases for User Interfaces • Generate release management scripts and documentation 	<ul style="list-style-type: none"> 16. Database ERD 17. JAVA Classes supporting Data Access Objects and tests cases 18. JAVA Classes supporting Business Logic Objects and tests cases 19. JAVA Classes supporting Reporting Requirements Objects and tests cases 20. JSP Pages supporting User Interface Requirements Objects and tests cases 21. Ant Scripts for compilation and release 22. Release documentation

Phase	Description	Work Tasks	Deliverables
VI	Application Test	<ul style="list-style-type: none"> • Unit level tests on the following: <ul style="list-style-type: none"> ○ Database ○ Data Layer Access Objects ○ Business Logic Objects ○ User Interfaces • Integration Tests • System Level Tests • User Acceptance Tests 	23. Summary results for Unit level tests 24. Summary results for Integration level tests 25. Summary results for System level tests 26. Summary results for User Acceptance tests 27. Defect Report
VII	Implementation	<ul style="list-style-type: none"> • Implementation of application into production hosting environment • Establish change control board (nominate members and assign roles) • Train end-user community on application usage • Train end-user community on the change processes • Initiate change control processes • Release application to end-user community 	28. Release application to production team 29. Change control board initial meeting 30. Hold training meetings for end-user community 31. Establish prototype web site and notify end-user community of its online status
VIII	Written Report and Presentation	<ul style="list-style-type: none"> • Generate written report in support of the project • Generate overview presentation of the project 	32. Generate and publish project paper document 33. Generate and publish overview presentation for the project

CHAPTER VII

Historical Project Information

This section of the document captures historical information as a result of the execution of the project plan for the SPTOF framework project.

Project Initialization Incentives

This project is an accumulation of a body of work that has spanned the past eight years of my career. In every organization in which I have worked, architecture, development, and design, teams have had little or no communication outside of the normal software development life cycle communications paths.

In roles in which I served in all three disciplines, in every case without exception, each team was focused solely on their own issues. They did not partner with the other disciplines nor did they share information. If knowledge was shared, it was done so begrudgingly. It was simpler to build the asset within the scope of a given project rather than wait on other teams that would not share the projection dates of when we might see the service we were seeking. I can personally admit guilt about building one-off implementations based on my own impatience with communication gaps.

An opportunity to define a common framework to address this issue was afforded to me in the summer of 2005. The architecture organization of which I am a member decided to bridge this knowledge gap. We decided to support the various organizations internal and external to the Information Technologies organization by supplying them with strategic and operational framework that supports common environments. In return, these groups would share information

about their environments. Collectively, this data would be made available to the enterprise at large.

The goal for the project is much broader than information sharing. The drivers for this project include:

- Reduce the overall cost to business by reducing the number of one-off environments.
- Empower designers and developers to look for solutions beyond the boundaries of their domains.
- Enable architects and operations to translate strategic and operational views into tactical views.
- Quickly identify gaps in the strategic and operational views of specific domains.
- To facilitate timely resolution of the gaps identified.

Chief among all the driving incentives for this project is to prove that communications among the three disciplines is not only possible, but can unify teams into an effective organization that the business will trust.

Project Measures of Success

There are a number of success measures for this project. Each measure is aligned with the ultimate goal of having the prototype accepted as a value ad. The following table documents individual measures of success.

Table 9: Schedule of Measures of Success

#	Measurement	Justification
1	Accurately document the problem statement.	Without an accurate problem statement and accompanying understanding of the problem, no accurate solution is possible.

#	Measurement	Justification
2	Define a framework that is readily understood and accepted by the architecture, development, and operations disciplines.	The framework is the backbone of the solution. It must represent the solution set for each of the three disciplines.
3	Define a change control process that supports the interests of the architecture, development, and operations disciplines.	The change control process ensures buy-in from all the disciplines involve. Each discipline must accept and participate in the process for it to work correctly.
4	Implement a prototype application that supports the change control process.	The prototype application is the physical realization of theory behind the framework and change control process.
5	Ensure the prototype application that supports the needs of the architecture, development, and operations disciplines.	The prototype application must facilitate a tangible knowledge-base for information gathered from all three disciplines and support the transformation of that information into useful data.

Project Management Details

This section of the document captures a view of the project plan for the SPTOF project. It provides graphical representation of the project through Gant charts. Additional project comments and milestones are presented with the Gant charts for clarity. The project plan is based on the “Project Phase Inventory” found in the Project Methodology section of this document. Each of the phases will contain logical group tasks that support completion of that particular phase. Additional milestone and project issues are provided on an as needed basis. The following diagram depicts the legend key for the Gant chart diagrams:



Legend	
Task	
Milestone	

Figure 36: Gant Chart Legend

The following Gant charts depict the project life cycle.

ID	Task Name	Task Notes	Start	Finish	Duration	Q3 05		Q4 05			Q1 06	
						Aug	Sep	Oct	Nov	Dec	Jan	Feb
1	Phase I - Analysis	- Gather information to define problem statement - Develop problem statement - Obtain approval from stakeholder for problem statement	8/1/2005	9/16/2005	7w	█						
2	Problem Statement Approval	Submitted to clients 8/16/2006	8/22/2005	8/22/2005	0w	◆						
3	Phase II - Framework Definition	- Define framework to address problem statement - Document framework layout and business function - Document function of framework	8/22/2005	12/9/2005	16w	█						
4	Phase III - Change Control Process Definition, Design	- Define purpose of change control process - Define roles within the scope of the process - Define flow for the change control process	9/5/2005	9/30/2005	4w	█						
5	Project Put On Hold For Funding issues and Prioritization Issues		9/30/2005	9/30/2005	0w			◆				
6	Change Control Process Approved	Submitted to clients 9/30/2006	10/10/2005	10/10/2005	0w			◆				

Figure 37: Project Plan Phase I, II, & III Diagram

- The SPTOF project started in full during the first week of August 2005. Phase I and II were completed on time and as expected. At the outset of Phase III, a project was identified as having greater priority for the business. Senior management then decided to shift resources from the SPTOF project to the higher priority project starting October 1, 2005. This priority shift was deemed temporary until the higher priority project was back on track for completion.
- The Client did sign-off on the “Problem Statement” , and “Change Control processes” were approved by the Architect, Development, and Operations teams.

- An additional factor for the delay was my lack of funding for the Fall Eight Week 2 term. My company does not fully fund my schooling. I had exhausted my personal funds and educational assistance for the calendar year.

ID	Task Name	Task Notes	Start	Finish	Duration	Q3 05		Q4 05			Q1 06	
						Aug	Sep	Oct	Nov	Dec	Jan	Feb
1	Project Restarted	Funding and Prioritization Issues Address	12/2/2005	12/2/2005	.2w							
2	Phase IV - Application Definition and Design	- Collect Use Case data from client population - Document Use Cases in text and UML format including Use Case Diagrams - Define High-Level program flows - Define repository supporting data structures - Develop Entity Relationship Diagram defining the database structure - Define User interfaces based on Use Case information - Define High-Level reporting structures - Generate programmatic flow diagram	12/2/2005	12/26/2005	3.4w							
3	Peer review of design completed and approved	Submitted to peer group 12/5/2005	12/12/2005	12/12/2005	0w							◆

Figure 38: Project Plan Phase IV Diagram

- Resource conflicts with the higher priority project were resolved the first week of December 2005. The project team was released back to the SPTOF project at that time.
- The database design and application definition was completed just prior to the holidays. The design and definition was submitted and reviewed for design approval by a peer group. The approval to proceed was obtained 12/12/2005.
- The design and definition was completed in full on 12/26/2005.

ID	Task Name	Task Notes	Start	Finish	Duration	Q3 05		Q4 05			Q1 06	
						Aug	Sep	Oct	Nov	Dec	Jan	Feb
1	Phase V - Application Construction	- Generate database scheme based on ERD - Generate data layer objects based on table structures define in the database - Generate test cases for data layer objects - Generate business logic objects based on Use Case requirements - Generate test cases for business logic objects - Generate report logic objects based on Use Case requirements - Generate test cases for report logic objects - Generate User Interfaces based on Use Case requirements - Generate test cases for User Interfaces - Generate release management scripts and documentation.	12/12/2005	1/27/2006	7w							
2	Submitted Application & Database to Test group	Testing begins	1/30/2006	1/30/2006	0w							◆

Figure 39: Project Plan Phase V Diagram

- The database and application construction was completed during the month of January 2006.
- The final release of the SPTOF prototype application was submitted to the testing team 1/30/2006. The testing team functions include:
 - System Test
 - Integrated Test
 - User Acceptance Test

ID	Task Name	Task Notes	Start	Finish	Duration	Q3 05		Q4 05			Q1 06	
						Aug	Sep	Oct	Nov	Dec	Jan	Feb
1	Phase VI - Application Test	- Unit level tests on the following: - Database - Data Layer Access Objects - Business Logic Objects - User Interfaces - Integration Tests - System Level Tests - User Acceptance Tests	1/31/2006	2/14/2006	2.2w							■
2	Testing completed		2/14/2006	2/14/2006	0w							◆

Figure 40 : Project Plan Phase VI Diagram

- Testing was conducted during the first two week of February 2006. The test team completed the testing on the 2/10/2006. Their final testing results report was released to the design and development team 2/14/2006. The following is a summary of their findings:

Table 10: Application Results Summary

Test Designator	Critical Errors	Defects Requiring Re-Engineering	Minor Defects
Database Testing	0	0	0
System Level Tests	0	0	3
Integration Tests	0	0	2
User Acceptance Test	0	1	5

- The one test result that identified as a defect was a navigation problem with Login. The user wanted a logout action to redirect the user’s session back to the Login screen. The navigation change was made during the testing phase and user acceptance tests re-run on that function. The user acceptance team signed off on the testing 2/14/2006.

ID	Task Name	Task Notes	Start	Finish	Duration	Q3 05		Q4 05			Q1 06	
						Aug	Sep	Oct	Nov	Dec	Jan	Feb
1	Phase VII - Implementation	- Implementation of application into production hosting environment - Establish change control board (nominate members and assign roles) - Train end-user community on application usage - Train end-user community on the change processes - Initiate change control processes - Release application to end-user community	2/15/2006	3/15/2006	4.2w							
2	Implementation in Hosting environment complete	Application is ready for user community to consume.	2/16/2006	2/16/2006	.2w							
3	Change control board Identified 1st meeting	Kick off meeting	2/21/2006	2/21/2006	.2w							
4	Phase VIII - Written Report and Presentation	- Generate written report in support of the project - Generate overview presentation of the project	12/15/2005	3/1/2006	11w							

Figure 41: Project Plan Phase VII & VIII Diagram

- The project was turned over to the deployment team 2/15/2006. This team is responsible for deploying the database and java components in a production environment. This environment is accessible by the common user population inside the corporate firewall network infrastructure.
- Submitted a suitable copy of the application and database to advisor (Mike Prasad) 2/8/2006 for MSCIT credit.
- The user population is starting to use the product and is evaluating its viability for moving forward to a production version that a large community could use.
- The Change Control Board held their first meeting 2/21/2006. This team will be electing officers and members over the next few weeks. They plan to hold review meetings every two weeks at the outset. They will increase the frequency as necessary.

Project Variables and Their Impact

Throughout the course of this project there were quite a few notable project-management issues that occurred during the first five phases. Among the most notable are the following:

1. The project was halted for 10 weeks during which time project resources were refocused on a project with a higher priority for the business.
2. I personally ran out of educational funding during this time could not support the Fall 8WK2 term.
3. The enterprise architecture group decided to use an off-the-shelf security solution for all web-based applications. The security portion of the SPTOF design was dropped in favor of this new implementation, which is due in March 2006.
4. The Vendor management team requested that the SPTOF application use its repository rather than implement a one-off database of vendor data. The SPTOF framework team will be working with this team in the near future to realize this integration opportunity.
5. Contacts will be managed out of the corporate LDAP implementation. The LDAP team was not ready to support the implementation of the SPTOF framework at this time.
This integration will be phased in during the next quarter.

The issues defined in list items 3, 4 and 5 caused vendor management, contact management and the local security efforts to be dropped from the SPTOF prototype design. A skeletal structure was used in the prototype to ensure the prototype application could be demonstrated professionally and academically.

Success and Failure Discloser

The SPTOF project has realized successes and failures on many levels. Most of the failures were schedule related. The initial project for the project plan specified that the project be completed by December 2005. Business priorities and funding challenges severely altered that plan.

Additionally, I had personal goals for implementing a security and vendor management portions of the framework that would provide additional functionality to the prototype application.

Strategic and tactical concerns did not support the security and vendor management portions of the design. Consequently, I was forced to sub-out those components. In the next iteration of the project, existing assets will be used to support those functions. While this is not a failure from the perspective of reuse and eliminating one-off designs, it was a personal disappointment.

The most notable success for this project was the initial meeting of the change control board. The event represents a huge step forward in communication among the architecture, development, and operations disciplines. Their participation is essential for the long-term realization of the goals of this project.

Testing for the prototype was very successful. A single defect was identified, but this was a configuration issue and not a code defect. The problem was readily remedied and retested. Code release was flawless and the prototype application is currently being evaluated for long-term adoption.

Project Summary

The projects' main goals were achieved. The SPTOF framework and related change control process and prototype are currently being used. The user community has embraced the concepts of the framework and its supporting knowledge repository. The associated benefits will not be realized for the foreseeable future.

As with most development efforts, the end-user community is gathering lists of improvements and modifications for the prototype application. This is not unexpected and is quite complimentary. If the application was not showing value, the user community would simply ignore the application. Generally speaking, feedback is good news for the longevity of the SPTOF framework.

CHAPTER VIII

Review of the Deliverables

What was the learned from the project?

There are two types of learning that occurred during the course of this project, professional and academic. From the professional perspective, the possibilities of bringing three disciplines together and establishing communication was suspect at best. The actual results of sharing knowledge among the disciplines discussed are turning out better than anticipated. It must be stated, the SPTOF framework is currently implemented in an encapsulated environment where the culture is not a huge factor in its success. Even with the positive reception, it is clear that senior and middle-level management sponsorship is vital to the success of any implementation of the SPTOF framework. There have been minor issues in bringing the necessary teams together. Management support has been an essential component in addressing those issues.

Documentation and training materials were sparse throughout the life of the project. The minimum documentation was not a success factor in the initial phases of the project. The lack of detailed documentation did present problems in the release and training phases. The support and end-user community requires more in-depth documentation should the project go to the next level.

From an academic perspective there were a number of lessons learned. I found it personally rewarding and at the same time frustrating to operate in all the roles of the software development life cycle. I have acted in the various roles of architect, designer, developer, tester, project manager and operations support person but not simultaneously. The greatest challenge was the open source tools used to generate the prototype. Installation and configuration required far too

much of my time. This time could have been better spent on documentation and improving the design. However, if the development and testing environments were not properly provisioned the project would not be possible.

I learned quite a bit about the design processes and the capture of requirements and supporting use case information. The methods of capture were adequate, but not robust. A proper framework for client interviews and information capture would be desirable for future revisions of this project.

What should have been done differently?

The project really required more development resources with more experience in user interface design and the backend technologies used to build the application. The business logic and data access elements took only 30% - 40% of the development effort. Far too much time was spent on user interface design and implementation.

A pre-provisioned development and testing environment would have helped greatly; far too much time was spent on deploying configuring tools for the effort. As this was mostly an academic exercise, this situation was unavoidable. The cost of off-the-shelf development and implementation platforms and software was not feasible.

Iterative user feedback during the design and development phases would have eliminated some of the navigational problems found during testing. In the future, the end-user community should have access to hands-on reviews of the application at key strategic points in its development to ensure the design meets the user communities' expectations.

Did the project meet expectations?

The project met the expectations of the client community. Their interest was focused on the proof-of-concept. They wanted to see if SPTOF was practical and feasible as a process within the software life cycle community. The prototype appears to have been accepted from the proof-of-concept perspective. Admittedly, the client community is more interested in the change control process and the long-term success of the application. Additional functionality and integration with common services well-established in the environment might improve the application value.

Personally, I was optimistic about the possibilities for the SPTOF framework going into this project. I was interested in learning more about JAVA technologies and deploying more tools and JAVA extensions that would increase my professional knowledge. I am a great advocate of open source technologies, but I find my enthusiasm depleted somewhat. I was disappointed about the amount of effort required to match technologies and revisions with hosting environments that would function properly.

Upon reflection, I see several areas of the project to revisit and improve. I would be very interested in refining and improving the application into a valuable corporate asset. Overall, the project was a satisfying experience.

Project Next Steps beyond Its Current Scope

Going forward, this project would benefit from several newer technology implementations and integration opportunities incorporated into future releases. The clients have already made requests for additional functionality not currently available in the prototype release.

Fundamentally, the database would benefit greatly from the implementation of the hibernate package. Hibernate is a persistence service that stores JAVA objects in relational databases or provides an object-oriented view of existing relational data. This will require some refactoring of code at the data access layer, but the benefits realized will far out-weigh the investment in implementing the hibernate package.

The client community has expressed a strong interest in adding projected and retirement dates in the application deployment records. This would facilitate views of the domain that would allow for greater flexibility in planning new projects and operational schemes for update and shutdown of existing applications. Additionally, it will greatly increase the strategic views capabilities of the architecture team.

It is not cost effective to design and build a report generator in the SPTOF application itself. The current reports are limited to the coded JSP and the servlets that support the JSPs. Integrating a report generation engine into the design would empower the users to leverage canned reports, as well as, taking advantage of OLAP (Online Analytical Processing) and ad hoc reporting technologies.

A personal goal for the project is to investigate integrating the SPTOF framework with UDDI (Universal Description Discovery and Integration) services. It was never my intent to replicate any existing information repository and UDDI is well-established. However, the merging of the information found in SPTOF with UDDI information could be another benefit of the SPTOF application.

The integration of the vendor management and security packages is the highest priority at the present time. If the prototype is to be taken to a production ready implementation it is one of the base requirements prior to its implementation.

Conclusions and Recommendations

The SPTOF framework, change control process and prototype application in their current implementation states are good proof-of-concept assets. They demonstrate that with proper support and implementation communications can be achieved. However, this is only a proof-of-concept. The SPTOF framework needs to be taken to the level of production readiness to realize its full potential. Given the proper funding and resource allocation, the project has the potential to decrease IT costs and increase productivity of the software development life cycle community. Further development and refinement of the SPTOF framework is recommended.

Summary of Project

The premise that a communications gap exists between the members of the software development life cycle community is not a new concept. There have been many governance models and development processes implemented over the years that strive to bridge this gap. In relative terms, the SPTOF framework is a narrowly focused solution that addresses a few issues in that problem space. There is always a danger with this type of narrowly focused solution that important issues are being ignored. In brief, the SPTOF framework was not designed to be all things to all people.

An important concept to reiterate is that this framework design is targeted for medium to large organizations. Typically, communications gaps do not exist in smaller organizations. In smaller

organizations, just a few individuals fulfill several roles of the software development life cycle. The SPTOF framework is no magic bullet. To implement the framework requires more than a hosting platform and external technologies components; it requires sponsorship and commitment from the appropriate management teams.

The project was a rewarding one. From my point of view, the SPTOF framework serves as proof that filling the gap in the communications paradigm is not only possible, but plausible to address. It increased my understanding of just how difficult and detail intensive the end-to-end software development life cycle is within any given project. Additionally, I have a much better understanding and appreciation of the value of pre-design detail work. The design and development efforts were not nearly as complex as with past projects in which I have participated. As the sole resource for this project, I feel I worked harder on the STPOF project than any other project in which I have been involved. I truly hope the project moves forward, but if this project does not continue on, I still consider it a personal success story.

References

- (2005). *Flashmap Systems Products* [Brochure]. Author. Retrieved August 3, 2005, from Flashmap Systems Web site: Flashmap Systems, <http://www.flashmapsystems.com/index.htm>, 2006
- (Scott W. Ambler, 2005) Ben Brauer, HP., & Sean Kline, Systinet. (2005, February). *SOA Governance: A Key Ingredient of the Adaptive Enterprise*. Ken Houghtling: Adapted from “Figure 1. Service delivery diagram” presented at the Online Presentation.
- Chris Britton. (2001). In *IT Architectures and Middleware, Strategies for Building Large, Integrated Systems* (p. 89). Addison-Wesely Publishers.
- Definition of term “Application”*. (2003). Author. Retrieved December 23, 2005, from Webopedia Web site: <http://www.webopedia.com/TERM/a/application.html>
- Erich Gamma, Richard Helm. (1994). In *Design Patterns: Elements of Reusable Object-Oriented Software* (p. 109). Reading, Mass: Addison-Wesley Professional Computing Series.
- Flashmap Systems Inc. (2005). *Flashmap Systems Products* [Brochure]. Author. Retrieved August 3, 2005, from Flashmap Systems Web site: Flashmap Systems, <http://www.flashmapsystems.com/index.htm>, 2006
- Inderjeet Singh, Beth Stearns., Mark Johnson, & the Enterprise Team. (2002). 4.4 Web-Tier Application Framework Design. In *Designing Enterprise Applications with the J2EE TM Platform* (2nd ed., p. 150). Author.
- Jakarta Taglibs*. (2002). Author. Retrieved December 12, 2005, from Jakarta Taglibs Web site: <http://jakarta.apache.org/taglibs/>
-

Janet Lowe. (2001). *Jack Welch Speaks: Wisdom from the World's Greatest Business Leader*. John Wiley & Sons.

JBoss Application Server Information. (2005). Author. Retrieved January 1, 2006, from JBoss Application Server Web site: <http://www.jboss.com/developers/index>

John Henry Clippinger III, Jossey-Bass. (1999). *Decoding the Natural Laws of Enterprise*. In *The Biology of Business* (p. 47). . (John Henry Clippinger III, 1999)

Log4J Logging Services Information. (2003). Retrieved August 1, 2005, from Log4J- Logging Services Web site: <http://logging.apache.org/log4j/docs/>

Measuring Information Technology and Productivity in the New Economy. (2002). *World Economics*, 3(1).

MySQL Database Server Information. (2002). Retrieved July 1, 2005, from MySQL Database Server Web site: <http://www.mysql.com/>

Pedro Sousa, Carla Marques Pereira. (2005). *Enterprise Architecture Alignment Heuristics*. Retrieved January 30, 2006, from Enterprise Architecture Alignment Heuristics Web site: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmaj/html/heuristics.asp>

Scott W. Ambler, James Linn. (2003). In James McGovern. Michael E Stevens (Ed.), *The Practical Guide to Enterprise Architecture* (1st ed., p. 61). Author.

Scott W. Ambler, John Nalbone. (2005). *Extending the Rational Unfired Process*. In *The Enterprise Unified Process, Extending the Rational Unfired Process* (p. 204). Pearson Education.

Todd Ditchendorf. (2005). *Ditchnet JSP Tabs Taglib Information*. Retrieved December 12, 2005, from Ditchnet JSP Tabs Taglib Web site: <http://ditchnet.org/tabs>