Department of Veterans Affairs

Office of Product Development (PD)

VA Enterprise Target Application Architecture

V 1.0

June 12, 2012

Document Approval

_____ _____

Lorraine Landfried Date

Executive Director

OIT, Office of Product Development

Additional Signatures (please print name and title below signature):

_____ _____

Date

_____ _____

Date

_____ _____

Date

_____ _____

Date

Document Control

Amendment Record

Issue	Date	Author	Comments
1.0	6/29/2012	Jeffrey Mohr	Initial general distribution

The latest version of this document supersedes all other previous issues.

Distribution Record

Issue	Date	Recipients
1.0	6/29/2012	General distribution

Index of Acronyms & Abbreviations

Acronym	Definition
ACL	Access Control List
ADA	Application Data Architecture
API	Application Program Interface
ACID	Atomic, Consistent, Isolated, and Durable
ATO	Authority to Operate
BAM	Business Activity Monitoring
BI	Business Intelligence
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
BRM	Business Reference Model
B2B	Business-to Business
CEP	Complex Event Processing
CPIC	Capital Planning and Investment Control
COE	Center of Excellence
CDC	Centers for Disease Control
CA	Certificate Authority
CIIF	Common Information Interchange Framework
CRL	Certificate Revocation List
C&A	Certification & Accreditation
CCB	Change Control Board
CIO	Chief Information Officer
COTS	Commercial Off the Shelf
CCEVS	Common Criteria Evaluation and Validation Scheme
CCTL	Common Criteria Testing Laboratories
COI	Community of Interest
COR	Copy of Record
CCBS	Core Common Business Services (CCBS)
CCIS	Core Common Infrastructure Services
DAS	Data Access Services
DIS	Data Interchange Services
DM	Data Mart
DRM	Data Reference Model
DW	Data Warehouse
DBMS	Database Management System
DoD	Department of Defense
DS Access	DoD Self-Service Access Center
DAA	Designated Approving Authority
DR	Disaster Recovery

DAC	Discretionary Access Control
DCOM	Distributed Component Object Model
DNS	Domain Name Service
DLL	Dynamic Link Library
EDI	Electronic Data Interchange
EAI	Enterprise Application Integration
EA	Enterprise Architecture
ECDM	Enterprise Conceptual Data Model
ECM	Enterprise Content Management
ELDM	Enterprise Logical Data Model
ERAM	Enterprise Release Allocation Matrix
ESB	Enterprise Service Bus
ETP	Enterprise Transition Plan
ETP	Enterprise Transition Plan
ER	Entity Relationship
XML	Extensible Markup Language
ETL	Extract, Transform, and Load
FBCA	Federal Bridge Certificate Authority
FEA	Federal Enterprise Architecture
FISMA	Federal Information Security Management Act of 2002
FIPS	Federal Information Processing Standard
FPKIPA	Federal Public Key Infrastructure Policy Authority
FOC	Full Operational Capability
GAO	General Accountability Office
GUID	Global User Identification
GOTS	Government-Off-The-Shelf
GUI	Graphical User Interface
HIPAA	Health Information Portability & Accountability Act
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secured Socket
MQ	IBM WebSphere MQ Series
IV&V	Independent Verification and Validation
IaaS	Infrastructure as a Service
IOC	Initial Operational Capability
iEHR	integrated Electronic Health Record
ISA	Interconnection Security Agreement
ICD	Interface Control Document
IATO	Interim Authority to Operate
IA	Internal Affairs
J2EE	Java 2 Platform, Enterprise Edition
JDBC	Java Database Connectivity

JMS	Java Message Service
LDAP	Lightweight Directory Access Protocol
LoB	Line of Business
LAN	Local Area Network
LDM	Logical Data Model
MI	Major Initiative
MAC	Mandatory Access Control
MTBT	Mean Time Between Failure
MTTR	Mean Time To Recover
MOA	Memorandum of Agreement
MOU	Memorandum of Understanding
MOM	Message Oriented Middleware
MHS	Military Health System
NIAP	National Information Assurance Program
NIEM	National Information Exchange Model
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OMB	Office of Management and Budget
EBA	OneVA Enterprise Business Architecture
EDA	OneVA Enterprise Data Architecture
OLAP	On-line Analytical Processing
OLTP	On-line Transaction Processing
ODBC	Open Data Base Connectivity
ORC	Operational Research Consultants
OS	Operating System
ODMS	Operational Data Management System
ODS	Operational Data Store
PRM	Performance Reference Model
PIV	Personal Identity Verification
PII	Personally Identifiable Information
PDM	Physical Data Model
PDM	Physical Data Model
POC	Point of Contact
PIA	Privacy Impact Analysis
PTA	Privacy Threshold Assessment
PMAS	Project Management Accountability System
PHI	Protected Health Information
PKI	Public Key Infrastructure
PUB	Publication
RFID	Radio Frequency ID
RPO	Recovery Point Objective

RTO	Recovery Time Objective
RHEL	Red Hat Enterprise Linux
RA	Registration Authority
RMI	Remote Method Invocation
RPC	Remote Procedure Call
ROI	Return On Investment
Root CA	Root Certificate Authority
508	Section 508 of the American with Disabilities Act
SSL	Secure Socket Layer
SAML	Security Assertion Markup Language
SRM	Security Reference Model
SRA	Security Risk Assessment
ST&E	Security Test and Evaluation
SBU	Sensitive But Unclassified
SQL	Sequel Query Lane
SCBA	Service Based Component Architecture
SCRM	Service Component Reference Model
FEA SRM	Service Component Reference Model
SIP	Service Insertion Package
SLA	Service Level Agreement
SO	Service-Oriented
SOA	Service-Oriented Architecture
SOA-TF	VA Service Oriented Architecture Technical Framework
SOI	Service-Oriented Infrastructure
SOx	Service-Oriented Architecture or Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SLS	Single Level Security
SaaS	Software as a Service
SP	Special Publication
SAN	Storage Area Network
SMP	Symmetric Multi Processor
SORN	System of Record Notice
SRT	System Recovery Time
SDLC	Systems Development Life Cycle
TRM	Technical Reference Model
TCO	Total Cost of Ownership
TLS	Transport Layer Security
UDDI	Universal Description, Discovery, and Integration
UID	User Identity
VAAFI	VA Authentication Federation Infrastructure
VAMC	VA Medical Center

VPL	Validated Products List
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
WSDL	Web Service Definition Language
WSIA	Web Services for Interactive Applications
WSRP	Web Services for Remote Portlets
WSIL	WS-Inspection

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

This *OneVA Enterprise Application Architecture (EAA)* document provides direction to VA system designers and developers as to how VA application systems will be designed and built. This EAA supports the OneVA EA goals especially the goals of facilitating delivery of services to Veterans, and supporting IT portfolio management and capital planning and investment control. This document provides the rules – the building codes, by which the designers and developers will build systems.

1.1. Background

VA operates a large number of application systems and is in the process of developing many more – both to provide new capabilities and to modernize existing capabilities. These applications are organized into a number of Major Initiatives (MIs) and business areas. Further there are a number of projects working to develop and enhance these applications[1].

In addition, VA is moving into a new application development and operational environment which will move it away from the functional silos that result from existing application systems towards a much more integrated operational environment, where systems are integrated both across VA and between VA and Department of Defense (DoD) Military Health Services (MHS). Further, it has been a VA systems development goal to move from the development of monolithic systems to the development of systems based on a reusable set of services and the provision of a core common set of both business and infrastructure services to be used by all systems requiring the capabilities that they provide across VA. To provide for the use of common systems and services across VA and to allow for the sharing of information across the enterprise, VA has recognized the need for the full harmonization of its data resources – semantic harmonization meaning that a data element has the same meaning across VA, and syntactic harmonization meaning that the allowed values and validation rules for a data element are the same across VA.

The existing components of the OneVA Enterprise Architecture (OneVA EA) describe the systems to be built and what they must do (the OneVA Enterprise Business Architecture [OneVA EBA]) with the enterprise data that is available for them to use (the OneVA Enterprise Data Architecture [EDA]), and the basic tools and products that can be used to build systems and services (the OneVA Technical Reference Model [TRM]). This EAA describes how to use the tools and products allowed in the TRM and the data identified in the EDA to build the systems needed to meet the business requirements specified in the EBA. In terms of a city planning metaphor, the EBA is the zoning or land use master plan, the TRM is the list of building materials that are available and the EAA is the set of building codes that govern the construction of the structures.

[1] There are a range of definitions for the term application system. However, it makes little difference to this document whether a development effort is designated as being for an application or a subsystem. As used in this document application system refers to a system defined as an application in the Federal Information Security Management Act of 2002 (FISMA) and the list of VA applications is the list applications submitted as the VA FISMA submission to OMB.

1.2. Scope

The EAA provides direction as to how to develop systems and services; it does not describe what the systems should do, it does not describe the development process, the Systems Development Life Cycle (SDLC), or the artifacts that are required as part of the development process (they are described in ProPath); nor does it describe the management or approval processes required to implement those systems (those are described in the Project Management Accountability System [PMAS]). It provides the technical direction for developing systems and services. It describes the basic building blocks that are available to the systems developers and provides the rules for combining those building blocks into systems. Although it is not the intent of this document to repeat the guidance in other architectural, project management or other guidance documents, it does at times provide some background on how the information in some of those documents relate to application system and service design and development.

VA systems are to be developed based on a VA Service Oriented Architecture (VA SOA). The VA SOA is specified in the "*VA Service Oriented Architecture (SOA) Technical Framework,*" (SOA-TF) which provides a technical framework for the development and uses of services within VA. It describes the SOA-based concepts that will form the basis for the design, development and deployment of services within VA, and a conceptual and logical overview of the infrastructure that will be needed to support those services.This document will provide direction as to the technical designs that are developed. Direction in this document is mandatory and systems adherence to this direction will be reviewed at appropriate PMAS gate reviews.

1.3. Mandatory Use for All Projects

The SOA and other architectural directions specified in this document are for all new software development efforts and major system modernization or system enhancement projects at VA. All newly developed external interfaces will be implemented as services that will be made available to all authorized VA users. All new functionality will be developed and exposed as services. No new services will be developed which duplicate services that were previously developed by other VA systems. Reuse of previously developed services, where such previously developed services can reasonably meet the requirements, is mandatory for all efforts. Where an existing service does not quite meet the functional requirements, the developer of the existing service will enhance the functionality of the existing service.

New projects and projects providing enhanced functionality to existing systems are expected to follow the direction mandated in this document and all modernization efforts should be planned based on the direction mandated in this document. Legacy systems are expected to follow this direction as they are modernized and when any functional enhancements are added.

Legacy systems will migrate to the SOA as quickly as feasible. The first step will be to expose legacy functionality as services to increase the availability of legacy functionality to the SOA services that will be developed. All modernization efforts will replace monolithic legacy code with services that expose required functionality. For non-service based legacy functionality that is not scheduled to be modernized or which has been modernized in a non-SOA manner, a service façade will be developed to expose the functionality as services and then the code residing behind the service facade will be re-factored – replaced with native service code as the code is modified or replaced.

Recently modernized systems will not be replaced immediately, but should develop plans to follow these directions for the remainder of their modernization efforts and should develop plans for

adherence to these mandates for previously implemented modernized functionality. In general ongoing modernization projects should expect to align with these mandates as they do technology refreshment of the previously implemented functionality.

1.4. Time Frame for Implementation

The EAA described in this document is to be viewed as the VA "end state" architecture – the Target Architecture, and is to be viewed as direction for all VA development projects. However, since the EAA is the "end state" architecture, it is not expected that all projects will be able to migrate to this architecture immediately, but all projects are to take steps to implement the architecture as quickly as they reasonably can.

This EAA does not describe the "current state" or the time frame or manner in which projects will transition from their current state to the end state described in this document. The time frame for current projects to implement this architecture and degree to which it is used for current projects is a subject for the *Enterprise Transition Plan (ETP).*

This EAA is intended to accomplish two goals - one is to provide a long-term vision as to the ultimate architecture to be used to implement VA systems while the other is to provide near term guidance to those implementing systems today. An issue arises in that the architecture that is desired for the long-term, future end state may include elements that cannot reasonably be completely implemented by all projects in the near term (i.e., they rely on the completion of efforts or projects performed by entities outside of the development group). This architecture provides that operational direction and each MI, VA Business Area, and system development project needs to determine how completely and how quickly it can implement this direction. While it is expected that new projects will conform much more fully than minor expansions of legacy capabilities, projects will need to develop plans to transition to this architecture.

1.4.1. Long-Term Architecture Implementation

The definition of the long-term architecture is important even if it is never fully implemented as initially envisioned. Two factors need to be recognized. First, the future rarely unfolds as predicted, so changes that are expected may never occur; and second, later authors of the architecture may change the architectural direction prior to the its full implementation. In addition to technologies that mature and that must be included in future versions of the long-term architecture, there will be changes in functional requirements and constraints, as well as in accepted architecture tools, techniques, and principles.

The long-term architecture is intended to guide the development of applications, even if it will never be fully achieved. The architecture will be updated periodically and each of those updates may adjust the direction of the long-term architecture. This does not diminish either its importance or the need for projects to be mindful of it. The importance of designing for the long-term architecture is that during the course of a system design effort there will be numerous day-to-day design issues that will need to be resolved. Each of the design issues may be resolved in a manner that makes it easier to instantiate the long-term architecture or which add to be base of legacy or non-compliant systems that must be transitioned. By taking cognizance of the long-term architecture and making design decisions consistent with moving to that architecture, the organization comes closer to compliance with the architecture and makes any future transition or transformation that much easier.

While the long-term architecture provides a broad vision of the direction in which the enterprise should be moving, projects that must deliver functionality in the near future must be given specific guidance as to how the applications they are building today should be designed.

1.4.2. Short-Term Architecture Implementation

There will be a certain amount of work that will need to be completed before systems can be developed that are fully compliant with this architecture. Even though all elements of the architecture will not have been developed and will not be available for use by application developers, application designers and developers should use the direction provided in this document. This direction is mandatory for all VA projects and shall be implemented by both new and ongoing projects to the fullest extent possible and as soon as possible. This specifically includes the use of SOA services, the separation of business logic and data logic, use of specified virtual infrastructure elements as they become available, the development of application logical data models and supporting their harmonization into the VA Enterprise Logical Data Model.

The OIT Enterprise Data Management Principles and the OIT Strategic IT Principles describe the basic architectural and data management principles that are intended to guide the development of VA systems. While these principles provide long term guidance, in the short term, these principles will need to be followed to the extent feasible. Ongoing projects may not be able to fully comply with all of the principles immediately, but will need to comply to the fullest extent possible and as soon as feasible.

The technical architecture should be read and interpreted in a manner such that whatever is not prescribed is proscribed i.e., what is not specifically allowed is prohibited.

1.5. Relationship of the EAA to the Other EA Components

Technology exists in VA to meet mission requirements. The mission requirements and the future vision of VA functional organizations as to how those requirements should be met, is translated to systems and technology through the EBA and the PRM. These EA artifacts document the changes that will be required to implement the business vision and the system capabilities that will be needed to implement those new business processes. This EAA documents how those capabilities are to be provided and how the systems providing those capabilities will be implemented.

The OneVA EA will document the business' vision as to how VA will perform its missions in the future and identify the changes to the business processes that will be needed to implement those visions. The OneVA EA will identify the system and technology capabilities that will be required to implement those business processes and will group those capabilities into systems for assignment to the MIs and Business Areas for implementation. This application architecture has been developed to guide the implementation of the systems that VA needs to implement its modernized business processes in the most efficient and effective manner and to improve the effectiveness of those systems by improving their overall availability and reliability as well as by reducing their implementation and operation costs.

The OneVA EA consists of a large number of components, some of which are more directly related to this EAA and some of which are less are directly related. VA applications systems exist to meet VA's mission requirements. The EAA exists to ensure that those systems are designed and built in a common manner and that they are built in a way that they are supportable throughout their life cycle. The OneVA EA contains two types of documents, those specified by the OMB Federal Enterprise Architecture and which include the set of reference models specified by that model and a set of addition documents specific to VA.

1.5.1. OMB FEA Components of the OneVA EA

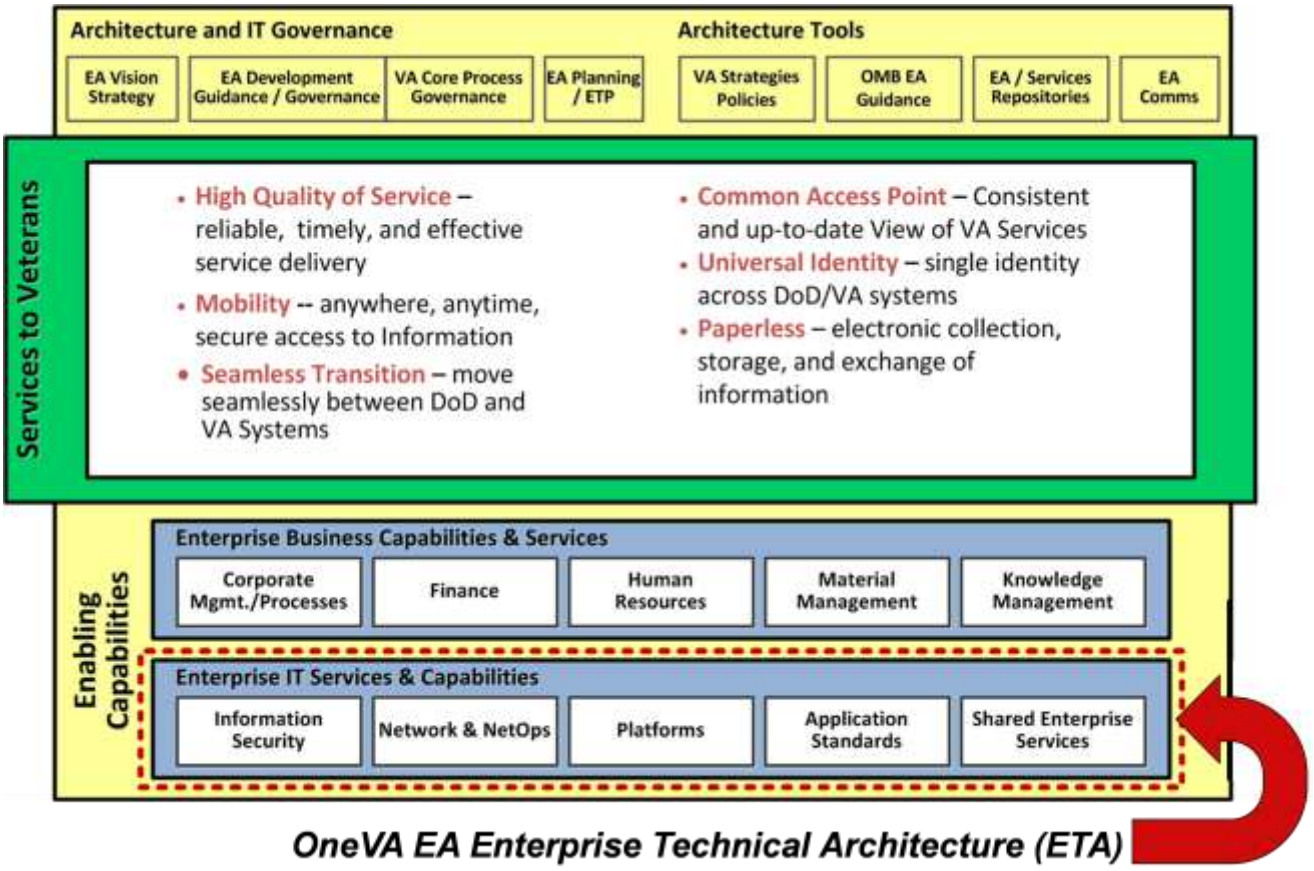

Figure 1 -- OneVA Enterprise Architecture

The components of the OneVA EA that are derived from the OMB FEA include:

1. **Enterprise Business Architecture (EBA)** – Describes the IT capabilities that are required to implement VA's senior management's vision of how VA should do business in the future.
2. **Business Reference Model (BRM)** – Is included in the EBA and describes the VA senior management's vision for how the VA should do business in the future and maps the VA business areas, MIs, and projects to the FEA Lines of Business (LoBs).
3. **Performance Reference Model (PRM)** – Provides specific, numeric measures of how IT system improvements (e.g., new or enhanced IT systems) will improve VA's mission performance (e.g., reduce Veteran homelessness).

4. **Service Reference Model (FEA SRM)** – Is an FEA Reference Model that identifies major service areas that VA IT systems support. These are major, very coarse grained functions such as payroll which would be implemented using a very large number of SOA services[2]. The OneVA EA does not include a separate FEA SRM, but rather, the information normally

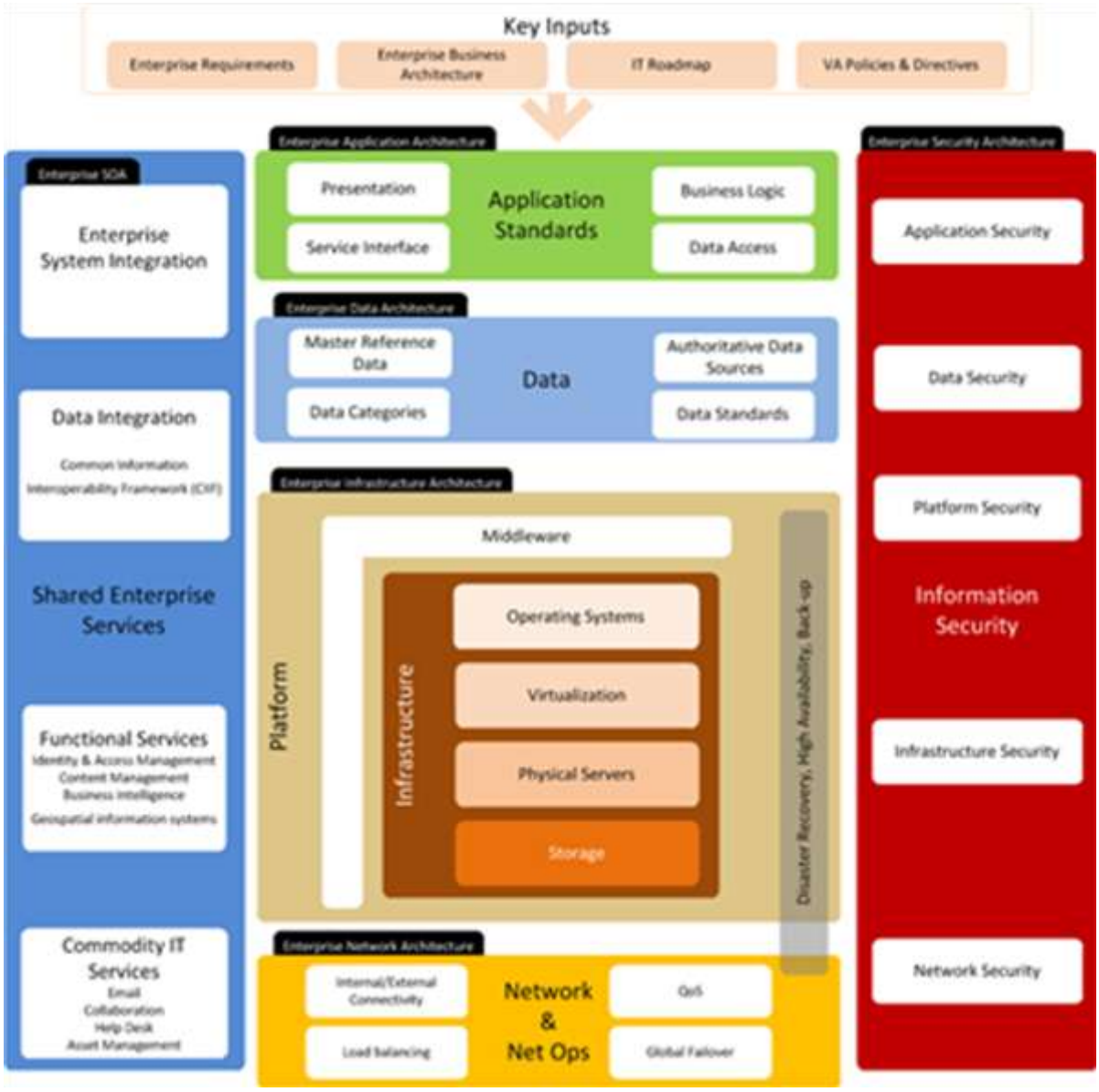

Figure 2 – OneVA Enterprise Technical Architecture

provided in the FEA SRM is provided on the OneVA BRM.**Technical Reference Model**

[2] Unfortunately both the FEA SRM and SOA literature use the term services, but mean very much different things. In this document the term "services" is qualifies as being either "SRM services" or "SOA services." When the unqualified term "services" is used, "SOA services" is the intended meaning.

(TRM) – Identifies the products and standards that can be used in the development of VA IT systems. The EAA limits the use of products beyond the restrictions on the products that are allowed by the TRM.**Data Reference Model (DRM)** – Identifies the major data subject areas about which VA collects data and identifies VA's major data repositories.

1.5.2. Components of the OneVA Not Related to the OMB FEA.

There are also a number of components of the OneVA EA that are not related to the OMB FEA. These include:

1. **Release Architecture (RA)** – Describes the product stack (a subset of the TRM) that is installed at each of the VA datacenters to support applications. It is the standard product stacks described in the RA, not the total list of products in the TRM, which application developers will use to develop VA applications.
2. **Enterprise Technical Architecture** – Describes the way information technology is to be deployed in VA and includes both the Enterprise Data Architecture and the Enterprise Application Architecture.
3. **Enterprise Data Architecture** – Includes the DRM, but also includes guidance on the use and sharing of data across VA.
4. **Enterprise Transition Plan (ETP)** – The plan describing how the VA will evolve from the current enterprise to the target enterprise.
5. **Security Reference Model (SRM)** – Describes the implementation of security mechanisms across the VA and the manner in which they are implemented. Both the manner in which the services are developed and deployed and the infrastructure on which they run will need to comply with Security Reference Model.

The EBA describes what the systems that VA builds must do; the PRM describes how well they do it (or the benefit to be obtained from building them); the TRM and RA describe the tools with which the systems can be built and operate and the EAA describes the ground rules for the development of the systems.

1.6. Relevance to Transition to Cloud Computing

Both the Federal Government and the VA have made the decision that the "cloud", whether commercial or government, will be the direction for Federal and VA computing and computing infrastructure.

> *"The Federal Government must be better prepared in the future. Beginning immediately, the Federal Government will shift to a "Cloud First" policy.*
>
> *The three-part strategy on cloud technology will revolve around using commercial cloud technologies where feasible, launching private government clouds, and utilizing regional clouds with state and local governments where appropriate.*
>
> *Cloud computing brings a wide range of benefits:*
>
> - ***Economical:*** *Cloud computing is a pay-as-you-go approach to IT, in which a low initial investment is required to begin, and additional investment is needed only as system use increases.*

- ***Flexible:*** *IT departments that anticipate fluctuations in user demand no longer need to scramble for additional hardware and software. With cloud computing, they can add or subtract capacity quickly and easily.*
- ***Fast:*** *Cloud computing eliminates long procurement and certification processes, while providing a near-limitless selection of services.*

When evaluating options for new IT deployments, OMB will require that agencies default to cloud-based solutions whenever a secure, reliable, cost-effective cloud option exists. To facilitate this shift, we will be standing up secure government-wide cloud computing platforms[3]*."*

This EAA does not directly address the issue of cloud computing (i.e., the location of the physical infrastructure used to process VA workloads) or any decision or plan to move VA computing infrastructure from VA datacenters to cloud computing facilities either run by government or commercial organizations. This EAA does not address the physical infrastructure – cloud, VA datacenter, or consolidated datacenter. However, a major goal of this EAA is to provide for the virtualization of the infrastructure and to hide the physical characteristics of the infrastructure from the applications and application designers and developers.

This EAA is very much oriented towards moving VA to Infrastructure as a Service (IaaS) in that it requires all infrastructure capabilities to be exposed as services to be used by the applications and SOA services. The EAA requires that physical infrastructure be exposed as services for use in the virtual sub layer. This is the basis for providing IaaS and then as the basis for moving VA infrastructure to the cloud environment, which then can be done transparently to the applications and the users. Providing IaaS is a key step in providing Software as a Service (SaaS).

Therefore, whether the computing infrastructure that is used to support VA SOA services and applications is provided from a VA datacenter or a Cloud computing facility should be completely transparent to application and service developers. The direction in this EAA to move to a much more standard, virtualized environment should make transition to a cloud environment much simpler and more straight forward. So, while this EAA does not require the use of the cloud, it does better position the VA for movement to a cloud – or other standard environments.

The actual mechanism by which the VA infrastructure is moved to the cloud, or even if it is, is not a decision to be made within the context of application or SOA services development, but will be made by infrastructure development and operations groups. Further, the decision to move to the cloud, or even the actual movement of VA infrastructure to the cloud, should be totally transparent to applications and SOA services developed in conformance to this EAA. Therefore, there is no need to discuss such a migration in detail in this EAA.

1.7. References

1. *OIT Strategic IT Principles*
2. *OIT Enterprise Data Management Principles*

[3] 25 Point Implementation Plan to Reform Federal Information Technology Management; Vivek Kundra, U.S. Chief Information Officer , December 9, 2010

3. *VA OneVA Enterprise Architecture*
    a. *OneVA Enterprise Business Architecture*
    b. *OneVA Business Reference Model*
    c. *OneVA Performance Reference Model*
    d. *OneVA Security Model*
    e. *OneVA Data Reference Model*
    f. *OneVA Technical Reference Model*
    g. *OneVA Enterprise Data Architecture*
    h. *OneVA Enterprise Transition Plan*
4. *VA Release Architecture*
5. *VA Service Oriented Architecture (SOA)—Technical Framework*
6. *VA Enterprise Target Application Architecture: SOA Layer Implementation Guide*

2. Principles

This EAA is based on a number of principles, some specific to this EAA and some more general. These more general principles relate to the development of the applications and the management of data by VA systems. The OIT IT Architecture and OIT Data Management Principles are included in this document and are being published as standalone directives. If there is a variation between the latest published version of those principles and the restatement of those principles in this document, the official, published version of the principles should be considered to be authoritative. The directives and policies are written as specific mandatory guidance without discussion as to the intent or implementation of the direction, this summary provides some discussion as to the implementation and meaning of the direction. This section does not include all elements contained in the various "principles" documents – only those most applicable to systems and services development. In particular, the summaries below do not include the discussions related to organizational responsibilities contained in those documents.

2.1. OIT Architecture Principles

The "target architecture" will serve as the basis for the design and development of new systems and updates to existing systems within VA. These principles are mandatory for the design, development, and operation of all systems within VA. However, because these principles will require changes in direction for many ongoing projects, there may be a transition period before all VA systems projects are able to fully implement these principles.

1. Systems shall be developed based on (this) VA EAA, which includes the VA Service Oriented Architecture (SOA).

This EAA describes how VA applications systems should be built. The VA Service Oriented Architecture (SOA) will serve as the basis for the development of VA applications. The VA SOA is to be used not just for exposing web services to external users, but as the core for the development of VA applications.

A Service Oriented Architecture (SOA) is an architecture in which business functionality is provided by a set of discrete, communicating "services" rather than by monolithic applications. VA has determined that using an SOA to provide internal and external applications access to data owned by and processing performed by VA applications is the best approach to sharing data, acquiring data from external sources, providing data to external users, and reducing redundancies across the

applications and business areas. The SOA is also viewed as the best approach for both sharing data and ensuring interoperability across applications as well as supporting cross functional integration and data sharing across functional silos. Both of these uses of an SOA are viewed as equally important aspects of this SOA.

The use of the SOA is intended to provide many operational benefits, including improving reuse of code, providing more current views of data to all who need it and improving business functionality, all while lowering costs and improving overall operational effectiveness. High levels of code reuse come from having a single service to perform any given function, rather than the more traditional use of code from a code library. The VA SOA goes well beyond many more traditional definitions of SOA as services are to be the basis for the development of all VA applications and not just for sharing information between systems.

As important as the SOA is to the development of VA systems, it is only one component of the VA EAA. The VA EAA includes not just the SOA, but the infrastructure needed to support the VA SOA. It describes not only how applications will be built using the SOA, but the infrastructure and other aspects of the architecture on which applications will be built and how they will be used in constructing VA applications.

2. ***VA systems shall be based on a tiered architecture, with a separation of code between the presentation logic layer, business logic layer, and the data logic layer on the server.***

A major aspect of the EAA is the separation of presentation logic, business logic, and data logic. The separation of business logic and data logic ensures that databases can be reorganized and redesigned without impacting the applications allowing for database redesigns to improve efficiency as usage of the database changes over the life of the system.

This also allows applications to be developed without regard to the physical structure of the underlying database(s) – hiding any knowledge of the physical structure of the databases from the business logic allows the database structure, and even the Database Management System (DBMS) that is used to support the database(s), to be changed without requiring any changes to the business logic (services or applications). Columns can be added to database tables, tables can be split, tables can be added, or data can be accessed from remote sources all without any changes to the business logic layer.

The separation of business logic and presentation logic allows multiple presentation services to use the same business logic service. This means that the same business logic service can be used to provide information to multiple types of devices (e.g., workstations, laptops, kiosks, tablets, or smart phones) with displays optimized for each particular device. The separation of business logic and presentation logic also allows the use of a single business logic service to support multiple presentation services, each aimed at a different audience. A single business service might access a patient record to retrieve the results of one or more tests. Separate presentation services might be employed to provide the results to the physician, the Veteran, or the Centers for Disease Control (CDC); each presentation service providing only that information relevant to the particular recipient and formatted for that recipient. This approach also supports a “defense in depth” approach to security, because the presentation services used to support “more public” access to data will be inherently less capable than other presentation services in terms of the types of information that they return and thus, further protect the “less public data” should the “more public” interface be compromised.

The converse is also true – where a single presentation service can be used to provide the user interface for multiple business services. A presentation service could be configured using XSL, XForms, and SVG standards. Each presentation template is a separate configuration of standard capabilities and can be used to display the output of multiple business logic services.

3. ***Data shall only be accessed through the defined data services layer promoting data reuse and supporting the Data Management policy.***

The first principle required the separation of business logic from data logic. This principle specifies how that separation is to be developed and maintained. Business logic will exist in a different layer in the architecture from the underlying data that it uses for processing. The architecture also specifies that the data shall be accessed through a well-defined set of data services that expose the data to the business services. As will be discussed below the data services will provide fully semantically and syntactically harmonized data based on the entities described in the Enterprise Logical Data Model (ELDM). The application owning the data will access the data through these formally defined data access services. Communication between the data logic services and the business logic services will be through the use of formal, semantically and syntactically harmonized messages. These same data access services will be made available to any other VA system requiring access to the data. Because all data access is through standard data services and messages providing the data to other applications requires no further effort than providing it to the application that owns the data.

4. ***No data shall be stored on the workstation or other end-user device.***

This principle prohibits the permanent storage of VA information on end-user devices. Temporary storage of information on end-user devices is allowed, such as temporary storage used to support display of information to the user or to serve as a cache for information collected using the device prior to that data being upload to VA servers. Permanent data storage on the workstations or other end-user portable devices is problematic for a number of reasons. First and foremost, storing data on an end-user device means that there will be a possibility of loss or compromise of the data should the end-user device be lost, stolen or compromised. Further, any enterprise data stored on the end-user device will need to be backed up and backing up data from thousands of end-user devices is inherently more complex than backing up the data from a limited number of enterprise storage devices. If the local copy of the data that is on the end-user device is the only copy of the data, then that data will not be available to other VA users. VA data must only be stored on VA servers, not end-user devices.

There may be some requirement for the use of mobile end-user devices where connection to the VA network is not possible. In those cases, data may be collected on the mobile end-user device, but 1) must be uploaded to VA servers as soon as feasible and 2) data input into a local device is not considered to be a change to VA data until such time as the data is uploaded and entered into a VA system. VA systems may have much more extensive data validation and acceptance processes including comparison of new data to data already stored on VA devices that is not available when data is input into a disconnected device, and thus, data accepted as valid by the mobile device may be rejected by the VA systems. Further, the synchronization of data entered into disconnected devices with updates to data stored on VA systems while the end-user device was disconnected is inherently problematical as updates may have been made to the VA system after the data was entered into the mobile device, but before it can be entered into the VA system.

5. ***The Web browser will be the user interface to VA applications. The web browser interface includes mobile code such as Java applets and may support "rich internet applications"***

which do not store or update the workstation disk storage. Non-browser based clients (i.e. fat clients) or any browser-based approach that stores data on the end-user device are not allowed.

Presentation logic will be stored and execute on VA enterprise servers. While presentation services will only provide information to an end-user device that the user of the device is authorized to see, the presentation service that provides the display of that data may have a need to access data that is more sensitive than the user of the end-user is allowed to see or more sensitive than can allowed to be transmitted to the end-user device.

In addition to not storing data on the end-user device, there should be no permanent storage of programs that need to be loaded on to the end-user devices as this makes the update of systems a major configuration issue and a major issue when the software needs to be updated. The preferred method of access is through a web browser so that the logic is updated every time that the page is accessed. Rich clients (e.g., Google Earth type applications) are allowed as they are downloaded and updated automatically based on data being stored on enterprise servers. VA "apps" for tablets and smart phones may also be used provided that they are downloadable and regularly updated through the web. The rich applications and "apps" should be configurable and able to be updated when the device software is updated.

6. ***The technical solution with the lowest enterprise Total Cost of Ownership (TCO) and / or best enterprise Return-On-Investment (ROI) that is able to meet VA requirements shall be chosen. TCO includes not just acquisition or development costs, but all costs associated with the design, development, deployment, and operations and maintenance of the system over the full period over which the system operates in the VA environment.***

While the cost of a solution is not per se an architectural issue, the lower total cost of common enterprise-wide solutions rather than project specific solutions is a major motivation for this EAA.

VA does not choose technology products for the sake of technology, but to meet documented business requirements in the most effective manner. Therefore, VA will select the lowest total cost solution to meeting its requirements. Lowest cost does not necessarily mean the lowest purchase price, or lowest initial price solution, but the lowest total overall cost solution. Total costs include not only initial acquisition or development costs, but also ongoing operations, maintenance and support costs for the life of the system. Overall costs of a technical solution includes all of the third party software that may be installed on a system, specifically including all costs to upgrade licenses of all software on mainframe or large-scale Symmetric Multi-Processors(SMPs) that are tied to the size of the machine rather than the usage of the software.

There are real costs associated with the introduction of any product into the VA environment. Total Cost of Ownership includes not just acquisition costs for the product or system but:

1) All acquisition costs
2) All operation and maintenance costs
3) All residual values for equipment etc. at the end of the system life
4) All disposal costs for any items that remain at the end of the system life
5) Potentially includes externalities (both costs and benefits) to the extent that they can be calculated to include:

a. User operational costs / savings – not just the VA data entry or application users, but Veterans usage as well

b. Improvements to mission operations to the extent that they can be quantified and monetized or to which a value can be assigned

A project team may raise issues with this direction which mandates solutions designed to achieve a global optimum, but, that does not happen to be locally optimal for that specific project. The project might rightly claim that it could do its job better, cheaper or faster, but for the requirements levied by this direction. However, in most cases the project's claims ignore factors outside the project's purview that increase VA costs, limit effectiveness or otherwise lower the overall value of the solution from an enterprise perspective. The project typically bases its evaluation of the costs of a solution on the costs that it incurs as part of the solution and ignores the externalities that the solution imposes upon the VA enterprise. Each time that a new product is introduced, VA incurs operations and maintenance costs for the life of the system. Further, when new products are introduced the purchase volumes of other products – products used throughout the enterprise - are reduced and volume discounts for the products are reduced. Thus, while the project's desired solution may reduce charges to the project, it may have much larger costs to the VA enterprise.

Traditionally, VA projects and contractors have selected systems architectures that both met mission requirements, with which the project team is most familiar, and which the project viewed as in some sense as being locally optimal, even if the locally optimum solution was more costly when costs are considered across the VA enterprise. This was especially true when many of the costs were not charged to the project, but rather paid for by the operations group out of their budgets. Lowest TCO is to be evaluated across the total VA enterprise and to include all costs. The requirement for the lowest TCO across the VA enterprise requires the use of standard solutions, because they reduce the systems engineering, security, management, operations, maintenance, and update costs. When standard architectures are used, these costs can be spread across large numbers of systems. When custom, non-standard solutions are used, additional costs are incurred in each of these areas for the individualized work that needs to be done to support that solution as well as the costs of obtaining a waiver from the standard solution.

Operations and maintenance costs are a part of the total systems cost. Therefore, VA systems must be designed for supportability – VA systems must be designed using the standard configurations of the standard products. This allows the use of standard systems builds and standard procedures markedly reducing the support costs for the systems. Further, fully testing systems so that they do not fail in production will also be critical.

Standard configurations of commodity servers provide the lowest overall cost solution. Therefore, standard configurations of commodity systems shall be used as the basis for VA applications systems and should be incorporated into all application designs, unless it can be shown that they cannot meet the mission requirements. Each VA datacenter may have an individual standard stack, but all applications running in a datacenter should use the standard stack for that datacenter.

While projects may request additions to the TRM for functional software that is unique to their functional requirements, projects are highly discouraged from requesting changes to the TRM for products to support the design, development, or operation of their system, but rather use the VA standard packages. The emphasis should be on the reduction in the number of redundant packages used within the VA – COTS products that can be used to meet similar requirements, even if there are slight technical differences between the products. Ideally there should be two COTS products on the

TRM able to meet any requirements. Two products assure a level of competition between vendors while still providing economies of scale in engineering and support.

7. ***VA solutions must be robust, scalable and adaptable to meet changing user requirements and demands, and will be designed to meet one of three (3) levels availability. These three levels include all scheduled and unscheduled downtime. The three levels are:***
    i. ***Mission critical (99.999% available),***
    ii. ***Mission important (99% available), and***
    iii. ***Generally important (95% available).***

VA applications and infrastructure shall be designed to one of three levels of availability – truly mission critical that essentially can never be completely unavailable, mission important that are allowed some minimal amount of total system downtime, and generally important that receive a standard level of support and availability. Clearly costs of the solution increase dramatically as the level of availability increases with the level of availability being requested.

Continuous availability means that the systems are available 24x7x365 with no scheduled or unscheduled downtime that makes the system functionality unavailable to the users. Individual components of the system can be expected to fail, but no such failure can be allowed to make the system unavailable to the users. The requirement that there be no system downtime visible to the user means that there can be no scheduled downtime for an entire mission critical system. This in turn requires that there be multiple instances of all hardware required to support the system, in at least each of two locations as well as redundant networks to assure that no single failure can take down the entire system. It will be possible to schedule downtime for individual components, for a full string of equipment, and if the system is instantiated at multiple sites – for all equipment at a site as long as a sufficient portion of the system remains operational that the user does not see the system as being unavailable. It is hard and expensive to provide 99.999% availability and this level of availability should only be requested where absolutely required. This level of availability requires that the entire system is never taken down at one time for update and at least one full string of equipment is always operational. No current VA systems operate at this level of availability and the current VA infrastructure is not designed to support this level of availability.

This goal will need to be reached through systems design and architecture, not just technology. However, this strategic principle cannot be satisfied with any approach that includes only a single copy of critical data or a single instance of any hardware component. To meet the 99.999% level of availability, it is likely that the system will require not just multiple strings of components at a single location, but multiple strings at two or more locations. As used in this section hardware component is a box, not a processor; so a solution consisting of a single large SMP would be considered as being a solution based on a single device and therefore, a solution that did not satisfy this strategic principle.

Much of the work to implement a working high availability solution will be based on a secure infrastructure and will be developed by VA infrastructure groups rather than as part of application development projects. However, even though the core development of the high availability infrastructure will not be the responsibility of the application development groups, the application designs will need to be cognizant of the need to operate in a high availability environment.

VA solutions being designed today are expected to remain operational for an extended period time, perhaps 20 to 30 years. VA systems have been experiencing natural increases in their workloads both because of increases in the number of veterans and the amount of processing needed to process the

benefits for each Veteran. Further, VA requirements have been changing rapidly in the face of changes to the mission, expansion of the mission, and newly legislated requirements and programs. These changes can lead to explosive growth in VA systems workloads and an urgent requirement to dramatically expand systems functionality. Therefore, VA can expect that its major systems will experience continuing workload and requirements growth. VA systems must be designed with the expectation that it is highly likely that after delivery that the nature of the system will change and that the system will be required to process arbitrarily large workloads. VA systems designs must therefore be scalable and able to grow to meet any requirements. Specifically, it must be assumed that the workload on any new VA system may grow to the extent that it cannot be supported by any single system. Therefore, all VA systems must be designed to operate using a cluster of loosely coupled systems. VA systems designs must meet the requirement for the lowest overall cost design, but must do that in a manner that provides a scalable system that can meet very large increases in workload.

VA systems designs need to be sufficiently robust to ensure that additional capabilities can be added over the systems life. This requires that the systems be designed in a highly modular manner so that individual modules can be added or modified without impacting the rest of the systems. The requirements are best met through a services SOA based design. Robust applications are a byproduct of the use of the SOA.

8. ***All commercial off-the-shelf (COTS) software used in VA solutions shall be from mainstream vendors who are able to provide the product across the VA enterprise over its full life cycle until it is removed from VA service.***

To provide the level of service that every Veteran has the right to expect, VA must use modern technology. However, because VA operates mission critical systems that must operate 24x7x365 the VA cannot afford the risks of the most bleeding edge technologies, even though they may appear to have some cost, speed or functional advantages. VA needs modern, leading edge technologies, but, technologies that are stable and that can be deployed across the VA. VA needs to be able to migrate to newer technologies as those technologies mature, and therefore must remain aware of those technologies and continue to track those technologies as they mature.

VA will only select reasonably mature technologies that have been proven in environments as large and as complex as VA. Further, VA solutions will only be based on products that are supportable and which the vendor is in a position to support at VA installations nationwide. Any VA organization planning the installation of solutions at VA locations other than the data centers will need to consider the supportability of the solution at the field locations. There needs to be an expectation that the product and the company that produces the product will last for the life of the system. Because of the long life of VA systems, it is recognized that products may go through many incarnations and even technical refreshments, and that companies may be bought or merged, but the expectation is that the capability that is acquired will continue to be available and upgraded over the life of the system. The requirement that VA solutions include only mainstream products from mainstream companies mitigates against technologies from smaller/startup companies.

While support would normally be acquired through the product vendor as the product vendor may use third parties to provide support. Open source software will also need to be support across the organization. This support may be provided by external commercial organizations (e.g., Red Hat in the case of Linux) or not for profit organizations (e.g., OSEHRA in the case of open source VISTA).

In addition, it may be a VA organization that commits to providing support for an open source product.

The requirement that VA solutions be mainstream and state-of-the-art means that products that are at, or even approaching, the end of their life should not be included in VA systems that are designed to meet new requirements. A key measure that a technology is no longer state-of-the-art is that the user base is no longer expanding, and that new versions of the product are only sold to previous customers and that companies using the product only use it to support legacy applications and not new development.

There is another aspect to the mainstream requirement included in this principle. It must be recognized that every COTS product incorporated into a system will require updates to new versions multiple times during the life of the system. Therefore, if a COTS or Government-Off-The-Shelf (GOTS) product is used in any VA system or project, it should be implemented in such a way that its upgrade requirements are as independent – or at least as loosely coupled, from the upgrade of other products as feasible so that the number of products that must be upgraded concurrently is as small as possible. In addition, because at some point in the systems life cycle it may become necessary to replace the product, making the loose coupling of COTS packages even more important. Further, because there will necessarily be a number of upgrades to each COTS and GOTS products over the life of the system, those products should not be modified, and any tailoring or custom code added to the system must be implemented in such a way that changes will not need to be re-implemented when the underlying product is upgraded. In general, this means that custom code should only interact with such products through vendor supplied Application Program Interfaces (APIs) or exits that the vendor guarantees will be supported through future versions. Where VA requires more significant changes to a product, VA should endeavor to get the vendor to make the changes to the core product, incorporate those changes into the standard distribution, and support those changes through future releases of the product.

To protect VA interests should a vendor face financial difficulties, a copy of COTS products' source code should be held in escrow by a third party. This practice is known as "code vaulting." This ensures that if a COTS product vendor goes out of business, VA would have a copy of the source code as a basis for future maintenance efforts.

9. All COTS software must be included in the VA Technical Reference Model (TRM) prior to its incorporation into any VA solution.

All products – hardware or software, need to be in the OneVA TRM prior to being used in a VA IT system or in a VA IT system development effort. However, only functional COTS software products will be of relevance to VA software designers and developers since the infrastructure elements will be designed, selected, acquired and implemented by other groups within OIT and specified for use by application designers and developers. Getting a product included in the TRM is the result of a process that reviews the product and the need for the product – not only that the product provides a capability required by VA, but that the product does not duplicate functionality already available within VA and is consistent with the existing VA infrastructure. The goal of this process is to limit the number of products providing similar capabilities within VA.

While there is justifiable concern with using the TRM to limit the acquisition of products for use in the VA, this does not limit the need to 1) perform thorough technical evaluations of any product prior to its introduction into the VA environment and 2) there are costs associated with the increase in the number of products throughout the VA and these costs cana and should be included in the cost

evaluation of any proposal that includes products not in the TRM. Any product introduced into the VA environment forces the VA to incur support costs for that software for the life of that software. Currently, these costs are externalized to the support organizations. Including the full life cycle costs of products and the full range of technical and support issues that they may cause is fully permissible. Further, it is equally permissible for VA to issue services only task requests that require vendors to build systems to the standard installed infrastructure.

10. ***Security shall be designed into all software, hardware platforms, and network infrastructure, and will provide for Mandatory Access Controls, single sign-on and auditing. National Information Assurance Program (NIAP) certification of products is desirable.***

Security and privacy and therefore the protection of information are critical to VA's continued ability to collect and manage information. Therefore, security must be a critical consideration in the design and development of all VA systems. Security must be considered at each stage of the design process and must be a primary consideration in the selection of a solution.

Mandatory Access Control (MAC) refers to a type of access control by which the operating system constrains the ability of a *subject* or *initiator* to access or generally perform some sort of operation on an *object* or *target*. In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, etc. Subjects and objects each have a set of security attributes. Whenever a subject attempts to access an object, an authorization rule enforced by the operating system kernel examines these security attributes and decides whether the access can take place. Any operation by any subject on any object will be tested against the set of authorization rules (aka *policy*) to determine if the operation is allowed. A database management system, in its access control mechanism, can also apply mandatory access control; in this case, the objects are tables, views, procedures, etc.

With mandatory access control, this security policy is centrally controlled by a security policy administrator; users do not have the ability to override the policy and, for example, grant access to files that would otherwise be restricted. By contrast, Discretionary Access Control (DAC), which also governs the ability of subjects to access objects, allows users the ability to make policy decisions and/or assign security attributes. (The traditional Unix system of users, groups, and read-write-execute permissions is an example of DAC.) MAC-enabled systems allow policy administrators to implement organization-wide security policies. Unlike with DAC, users cannot override or modify this policy, either accidentally or intentionally. This allows security administrators to define a central policy that is guaranteed (in principle) to be enforced for all users.

Evaluations of COTS products are conducted through the NIAP Common Criteria Evaluation and Validation Scheme (CCEVS). The NIAP is an activity jointly managed by the National Institute of Standards and Technology (NIST) and National Security Agency (NSA) and staffed by personnel from those agencies[4].

[4] The Common Criteria Evaluation and Validation Scheme, http://niap.bahialab.com/cc-scheme/defining-ccevs.cfm.

The NAIP CCEVS Validation Body, hereafter referred to as the Validation Body, is managed and staffed by the NSA. The Validation Body assesses the results of a security evaluation conducted by a Common Criteria Testing Laboratories (CCTL) within the scheme and when appropriate, issues a Common Criteria certificate. The certificate, together with its associated validation report, confirms that an IT product or protection profile has been evaluated at an accredited laboratory using the Common Evaluation Methodology for conformance to the Common Criteria. The certificate also confirms that the IT security evaluation has been conducted in accordance with the provisions of the scheme and that the conclusions of the testing laboratory are consistent with the evidence presented during the evaluation.

The Validation Body maintains a NIAP *Validated Products List* (VPL) containing all IT products and protection profiles that have successfully completed evaluation and validation under the scheme. The VPL includes those products and profiles successfully completing similar processes under the schemes of authorized signatures to the Arrangement on the Mutual Recognition of Common Criteria Certificates in the Field of Information Technology Security.

Because of the length of time that it takes for a product to complete evaluation and the fact that only products that are expected to be successfully validated are submitted, the use of products that have been accepted for evaluation and for which the evaluation has started can be used.

11. Secure information sharing is an essential part of the VA Architecture. Information shall be "shared by rule and withheld by exception" with all authorized users with a need to know.

A major thrust of this application architecture and the VA Data Architecture is to remove information sharing barriers so that current, accurate, timely information is available to any person or system within VA with a need to know when and where and in the form that it is needed. All information collected or generated anywhere in VA is a VA asset to be shared amongst all VA systems subject to only to security, privacy, need to know and other legal restrictions. The primary goal of many of the MIs is to promote sharing of information within VA and to make that information available when, where, and in the form desired.

Any information collected by any VA system is VA information and is intended to be shared with all VA systems subject only to security, privacy or other legal restrictions. It is incumbent upon the steward for any data collected, generated, stored or maintained within VA to ensure that the data is accessible and available to all VA users or systems that have a requirement and are authorized to see that data.

VA must transition from an environment where an OIT organization views itself as the owner of information and its mission to protect that information to one in which each organization views itself as the steward for the authoritative instance of corporate information and its own core mission as getting that information to where it is needed.

All information is corporate information for which individual program offices are the steward. The core mission is to share the information and to ensure that all who need it are provided access.

12. All communications between architecture layers shall be based on formal messages or published Application Program Interfaces (APIs).

The capabilities provided by each layer in the architecture are formally defined as services and communications between these services is through the mechanism of formal messages. The formal mechanism for communication between layers in the architecture shall be through formal, well-

defined messages. In some cases there will be communication between actual services, sometimes between services and service facades placed in front of systems or software, that is not inherently services based, and sometimes – particularly at lower layers of the hierarchy, based on formal APIs. The intent is that all communication between services and between services and non-service capabilities will be through standard, well defined interfaces and not through special programming of modification of lower level programs of capabilities. Services at multiple layers in the hierarchy are to be loosely coupled and this is best accomplished when communications is through formal mechanisms as required here.

13. All access to Personally Identifiable Information (PII) and Personal Health Information (PHI) shall be logged and is subject to audits. Appropriate controls shall be implemented and enforced.

VA protects two types of information:

- **Personally Identifiable Information (PII)** – Any information about an individual that can reasonably be used to identify that individual that is maintained by VA, including but not limited to, education, financial transactions, medical history and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, telephone number, driver's license number, credit card number, photograph, finger prints, biometric records, etc., including any other personal information which is linked or linkable to an individual.
- **Protected Health Information (PHI)** – Individually Identifiable Health Information that is under the control of VHA, as VA's only Covered Entity under Health Information Portability & Accountability Act (HIPAA). PHI is health (including demographic) data that is transmitted by, or maintained in, electronic or any other form or medium, and relates to 1) the past, present, or future physical or mental health, or condition of an individual; 2) provision of health care to an individual; or 3) past, present, or future payment for the provision of health care to an individual, and that identifies the individual or for which there is a reasonable basis to believe it can be used to identify the individual. If the information identifies or provides a reasonable basis to believe it can be used to identify an individual, it is considered individually identifiable health information.

It is vital to the VA mission that the VA collect and store both PII and PHI; to maintain public trust, the VA must show that it can protect this information. Protection of PII and PHI includes not just implementing the security controls to limit access to the data, but also to be able to track who has had access to specific data. This means being able to reconstruct who had access to data requires that requests for access to data be logged. VA can only show that access to PII and PHI is appropriate if such access can be audited after the fact, which again can only be done if the accesses are logged.

2.2. Data Management Principles

This section provides a summary of VA Enterprise Data Management Principles as they relate to development of systems in VA. Areas of the Data Management Principles not directly related to the development of VA application systems or the operation of those application systems are not included in the summary below. This summary of those principles does not include a discussion of the organizations responsible for the activities described below.

1. Enterprise Logical Data Model

a. There shall be a VA ELDM that shall identify each "enterprise" entity, an "enterprise" entity being an entity that contains at least one attribute (data element) that might be of use outside of the system in which it is created or stored. Any data that enters or leaves a system is considered to be data used outside of that system. The phrase "might be of use outside of the system," although vague, should be construed as broadly as possible. In general, all data should be in the ELDM except data completely internal to a system. Using this phrase allows for the possibility that there might, in some rare cases, be data, which is visible external to a system (e.g., appears as input or output to the system) which could not be of use elsewhere in VA.

The VA ELDM will serve as the basis for the definition of a set of syntactically and semantically harmonized messages. These messages will have to have the same syntax and semantics across VA; and since these messages will be the sole mechanism for communication between SOA services across the enterprise, they will need to encapsulate all information that flows between VA systems and between VA systems and end-user or non-VA (i.e., external) systems. By definition any data that flows from external e.g., to end-users, other VA systems, or external systems is enterprise data and needs to be represented in the ELDM. Strictly speaking, only data that is visible external to a VA system is included in the ELDM; therefore, data that is wholly internal to a system would not normally be included in the ELDM, although those data elements would be included in the project logical data model. The reason for the broad interpretation of the concept of enterprise data is that the boundaries of the application today may change as the system is redesigned using SOA services, particularly if those services are then exposed for more general use across VA, VA business area or MI. If this application exposes services across VA or to systems in a business area or MI, data that had been internal would now become enterprise data as it would be exposed to services outside of the application.

b. The ELDM shall provide harmonized, standard definitions of all entities and attributes, and it is these enterprise definitions that shall be the basis for data exchanges between systems. It shall be the responsibility of each application to convert from its internal data definitions to the enterprise definitions for communication with enterprise services or other systems.

The ELDM identifies all enterprise data collected, created, stored and processed by VA systems. There will only be a single definition for each data entity and data element and that definition will be used by all systems across VA. A single definition means that when two systems use data elements with the same name, the data elements have the same syntax and semantics and when they use data elements with different names they really do refer to different items – i.e., the syntax, semantics, or both are different between the two data elements. The use of distinct namespaces can be used to assure uniqueness of data element names that are local to a system even though the data names are used elsewhere in the enterprise. This does not obviate the need to harmonize data element names that are used outside of a single system.

This syntactic and semantic harmonization will form the basis for VA Standard Messages that are discussed in this EAA. These VA Standard Messages will be the basis for data exchanges between VA systems and will be the only messages that SOA services will accept. Each application system will be responsible for transforming messages and data from the application's own internal format to the VA Standard. This places the responsibility for the transformation with the group with the greatest capability to perform that transformation – the developers for that application that have the detailed understanding of the details of that application's data syntax and semantics. Each application

development group will need to develop only a single transformation – to and from the VA Standard, not, potentially, once for every application with which the application communicates.

c. For each enterprise entity, there shall be one business organization designated as the Functional Steward for that entity, and one OIT organization that shall be designated as the Technical Steward for that entity.

All data collected, created, stored, and processed by VA systems is VA data and is owned by VA. However, there will be two organizations that will be responsible for the management of the data – the Function Steward and Technical Steward. The roles of these two stewards are described below.

i. Functional Data Steward

The functional steward represents the "owner" of the entity. The "owner" of the entity is the business organization responsible for collecting the data and ensuring its accuracy, timeliness and consistency. The Functional Steward is responsible for changes to the data. For example, in a pharmacy system, the Functional Steward would be responsible for approving changes to the "formulary" – the list of drugs that the pharmacy stocks. The Functional Steward cannot be an OIT organization. In addition, the Functional Steward is responsible for identifying the applicable security, privacy, records management and records retention requirements – with records retention requirements subject to approval by National Archive and Records Administration (NARA). The steward is also responsible for designating the domain management rules which govern processes for creating and approving new, valid data values.

The Functional Steward along with the Technical Steward shall have the responsibility for sharing the data with all VA applications and organizations with a need for the data in accordance with applicable security and privacy laws, regulations, and policies (both PII and PHI).

The Functional Steward shall determine with whom data may be shared, while the Technical Steward will be responsible for the technical manner in which the data is shared. While the desire is to share data as widely as possible, the Functional Stewards and the Technical Steward shall determine the rules for access and update of data based on the need to minimize risk of any inadvertent or unauthorized disclosure; subject to any applicable laws, regulations, or policies etc. that impact who may see / access the data; how or by whom the data may be changed; archiving and retention requirements for the data; as well as any other aspects or data access, update, assurance etc.

ii. Technical Data Steward

The Technical Steward is an OIT organization and is the owner of the database(s) or data stores used to store the data and the systems and services used to access the data. The Technical Steward decides on the physical storage of all VA data (e.g., if it is stored in a single or multiple databases) and ensures the enforcement of consistent data use by all application developers as designated by the Functional Stewards and that it is consistent and in compliance with applicable security, privacy, records management, and retention requirements – with records retention requirements subject to approval by National Archive and Records Administration (NARA).

It is the responsibility of the Technical Steward to ensure update of the authoritative instance and approve creation of any necessary performance-related copies. Application developers who need to create a copy of the authoritative data shall generate a request to the Technical Steward for access to the OLTP system that contains the COR. Any request to create a copy of the COR shall include documentation of the mechanism to be used to update the authoritative instance of the data (e.g., a

direct system interface, service request etc.). In no instance shall a change to the copy be considered to be effective unless and until the COR for that data has been changed.

The Technical Steward for VA data shall have the responsibility for providing physical extracts / summaries of that data to populate a VA Enterprise Data Warehouse (EDW) or VA data marts. Functional Stewards will serve as advisors to VA regarding the validity of extracts or summaries as designed.

2. **Authoritative Instance of the Data**

    a. **A single instance[5] of each data element (attribute in an entity) in the ELDM will be designated as "Authoritative," and will serve as a unique and unambiguous source of data to be shared operationally across all systems in the enterprise.**

A Veteran cannot have a single unique address in VA, but may have many. The Veteran may have a:

- Permanent (Street) Address.
- Permanent Mailing Address.
- School Address.
- Temporary Address.

Each is valid, but each has a different meaning and must be represented by a different name. While many systems may use any of these addresses, each would have a different name and a single unique value across VA. The only way that a data element used by multiple systems can have a single value across all of those systems is for there to be one unique, unambiguous version of the truth – the one which when it is changed, the change is recognized across VA. This one unique, unambiguous version of the truth is called the "Authoritative" instance of that data element. The "authoritative" instance of the data is also known as the Copy of Record (COR).

While the principle states that there is an authoritative instance of each data element, normally the authoritative instance of all data elements (attributes) contained in an entity will be stored in the same database. However, because there is not a one-to-one relationship between entities in the logical data model and tables in the physical data model / physical database design, the principle is stated in terms of data elements (attributes). For example, it would be foolish for the authoritative instance "*first name*" and "*last name*" to be in different databases or OLTP systems.

    b. **The authoritative instance resides in a specific Online Transaction Processing (OLTP) data base referred to as a "Copy of Record" (COR). It is the single, unambiguous "truth" for VA. The database that contains the COR is defined as part of the data architecture, and does not change as data is updated or as new records are created. An Operational Data Store (ODS), Data Warehouse (DW), Data Mart (DM), or other data store optimized for Online Analytic Processing (OLAP) may not house the COR for any data.**

[5] Instance: a case or occurrence of anything *or* an example put forth in proof or illustration

The COR for a data element will be housed on the OLTP system that is the primary system responsible for the collection of the data. If multiple systems collect the data, the Data Architecture will identify one of the OLTP systems to house the COR. Only an OLTP system can house the COR as other versions of the data – those in an ODS, DW or DM or any other OLAP version of the data may be based on transformations of the data that are updated based on data extracted from the OLTP system after some delay. Because none of the copies other than those in an OLTP system are optimized for update, there is necessarily a delay between the time the OLTP system is updated and when the data is extracted, transformed, and then loaded into the OLAP system(s).

The COR must reside in a single fixed system, at a fixed location, so that other systems needing the information know where to find the one, true value of the data element. If the COR cannot be assured of remaining in one fixed location then, the location must be stored in an accessible registry, so the COR can be found by any system needing to access the current value of the data element.

c. Copies of the authoritative instance may be made by other OLTP systems for performance purposes with permission and under the control of the Technical Steward, but no update to any such copy or creation of new records in such a copy shall be considered to be effective unless and until the authoritative instance has been successfully updated.

The COR will necessarily be maintained by a single system at most a small number of locations (i.e., a second location if the system is configured as a high availability system). Large numbers of other systems may require access to the data and may not be able to tolerate the delays inherent in remote access to the data. Such systems are allowed to make copies of the COR – the authoritative instance of the data, but updates that they make to a copy of the data are not considered to be official until the COR has been updated. It is the responsibility of the system with the copy of the data to ensure that the COR is updated. If the application that had updated the copy of the data does not update the COR, the COR would not have the latest updates to the data and other systems requiring access to the data would also not be able to get the updates.

3. Separation Between Data Logic and Business Logic

a. There shall be a complete separation between business processing and the use of the data and data manipulation, data access and delivery services such that the business logic has no visibility into the physical structure of the data.

As stated above in the discussion of the IT Principles there will be a complete separation between the business logic and the data logic. That discussion will not be repeated here, but the sections below provide a somewhat more detailed discussion of the issue.

b. Business logic shall only access data through well-defined data access services such that physical changes to the physical data base design (e.g., replacement of the Data Base Management System [DBMS], increase or decrease in the number of tables or number of columns in a table, etc.) has no impact on the business logic and requires no changes to the business logic.

The data layer will expose a set of data services that access the physical data and that present data to the data logic services accessed by the business logic services. Data exposed by a single data service may include data from multiple tables, multiple databases, or even multiple DBMS'. This allows for changes to the underlying data without impact to the business services and allows the data structures to be extended or restructured at will.

c. Separation of business logic and data logic requires that no SQL, SQLnet, Java Database Connectivity (JDBC), or Open Database Connectivity (ODBC) etc. is allowed at the business logic level. Any product or technology for which knowledge of the DBMS product being used (e.g., table names, or column names etc.) is not allowed at the business logic level. Capabilities such as those provided by PL/SQL or T-SQL are allowed at the data layer, but are not allowed at the SOA services layer.

Because business services cannot have any knowledge of the underlying data structures there can be no SQL, SQLnet, or other products that inherently require knowledge of the underlying data structures in the business logic. Data logic services will be explicitly dependent on the data structures and may need to change with every change to the underlying data structures. The data layer services can be developed using PL/SQL or T-SQL or other DBMS specific languages as appropriate for the DBMS being used and allowed by the TRM. Stored procedures are specifically approved for use at the data layer and may be accessed from the business logic layer. While the stored procedure will have knowledge of the underlying data structures, this information cannot be made visible from the business logic layer. This provides the flexibility to allow the data services to perform the functions that are necessary to build the data services to be able to provide the data to the business logic services and maintain the separation between the business logic services and the data logic services.

2.3. Enterprise Application Architecture Principles

An application is a collection of services that automate business functions and provide capabilities and structure of software for the business users. Each such application will either be a COTS / GOTS product or a set of custom developed services.

The VA EAA will be based on a series of layers and components each of which provides a unique programming and development environment as well as unique capabilities.

VA business applications will be designed and developed based on the following principles which will be expanded upon subsequently as follows:

- **Services based design and development –** All applications will be implemented as a series of atomic or composite services. Development of monolithic systems will not be allowed.
- **SOA Services will reside in the SOA Services Layer –** This EAA is based on a multi-layer design that isolates functionality to a series of dedicated layers. The SOA will be implemented as the topmost layer of the hierarchy and will access capabilities provided by the functionality at the lower layers.
- **N-tier SOA Layer application designs –** Applications will be built using SOA services which will be provided based on the "*n-tier"* design approach in the SOA Services Layer that separates browser-based clients, portal servers, application servers, and data servers, etc., to separate tiers of the architecture, client/server architecture is not allowed, except for COTS solutions that have received waivers.
- **Core Common Services –** Some services will be designated as Core Common Business Services (CCBS) or Core Common Infrastructure Services (CCIS). These will be built once and used by all projects needing the capabilities that they provide. Their use is mandatory.
  - **Core Common Business Services** – Are functional services (e.g., correspondence, case management, workflow, or adjudication etc.) the CCBS are high level functions that may

be composed of a large number of SOA services and perform a relatively standard function or which can be parameterized and is useful across a large part of the VA.

- **Core Common Infrastructure Services** – Are infrastructure or technical services (e.g., ID Management) that are needed across large number of VA applications. The CCIS may be SOA services based, but not necessarily as they may provide infrastructure capabilities that the SOA services need to operate.

- **Service reuse –** Wherever feasible existing services will be reused and new services will not be developed if existing services are available that provide the required functionality.
- **Distributed systems design –** All VA applications will be designed to scale out (i.e., run on larger numbers of small systems) rather than scale up (i.e., run on larger and larger SMP systems) and will be designed to operate on a series of loosely coupled commodity systems.
- **Access to applications –** User access to applications is permitted only via the VA provided infrastructure (Internet or intranet). Systems access to other applications will be through service requests mediated by the Enterprise Service Bus (ESB).
- **Secure messaging service –** Only approved message paths shall be used and no direct access (for example, SQL) by users to the internal databases is permitted. There will be no access to applications which bypass VA applications or security infrastructure.
- **Access to data housed in VA data stores –**Applications will the access the authoritative instance of data rather than using local copies unless they are needed for performance. No changes to copies of authoritative data will be considered to be final unless and until the authoritative instance has been successfully updated.
- **Applications shall use Data Marts (DMs) rather than OLTP systems for complex queries –** Data will be extracted from application OLTP systems that are optimized for update into an EDW. The EDW will be used to generate a series of local DMs. These DMs shall be used to support complex queries from the applications so as not to burden the OLTP systems with this workload.

2.4. Overview of the VA Service Oriented Architecture

Some consider an SOA to be an abstraction layer that provides agile business services to the user while hiding the underlying complexity of the technology. However, an SOA is much more than just an abstraction layer. It provides a very different approach to architecting solutions than has been used previously and requires a mind shift from designing monolithic (and ever growing) stovepipes to designing reusable components.

2.4.1. SOA Services versus Web Services

There is general agreement in the literature that the term service as used in SOA is distinct from the term service as used in "Web services," which are services specified by and provided based on a specific set of standards – usually with a name in the form WS-XXXX. As used in this paper, the term "Web Services" specifically refers to services provided in compliance with that set of standards. As used in this architecture, services is a more general term that includes Web services, but also includes services that are not compliant with those standards.

"Web services" as that term is most often used will be used primarily for interactions with users and systems external to VA. Initially at least, a major use of services implemented as part of this SOA

will be to expose functionality of legacy systems and to serve as a basis for the modernization of component systems and thus, more properly fall under the rubric of Service-Oriented Integration (SOI). The VA SOA covers both the provision of services to provide interoperability between systems in different business areas as well and to support integration of systems and providing interoperability of systems within a single business area.

The approach taken in this SOA is to publish a set of services that will be accessible to various classes of users. The SOA addresses the need for very fine grained services that might be used to support communications between very closely related applications wholly within a business area, to sharing across VA and DoD, as well as with organizations outside of VA.

Services exposed external to VA and DoD should conform to applicable Web services or other required standards[6]. These services will be relatively coarse grained and will provide relatively major functional capabilities. Services that are only exposed within a business area will likely be much finer grained and are less likely to conform to Web Services standards. The notion is that services will be used for two purposes 1) to share information and functionality between applications and 2) to serve as the basis for the development of applications. The notion is that services exposed external to a component will be relatively coarse-grained services – services that perform a reasonable amount of work. These services may be composed of a number of fine-grained services that execute within the application. Since services are also viewed as a mechanism to simplify applications and to speed application development, services may need to be very lightweight and to be used where there is a very high degree of trust between the service provider and service consumer – a much higher degree of trust than typically exists between web service consumers and web service providers.

2.4.2. SOA versus Traditional Integration Architectures

Fundamentally, in an effective SOA implementation, integration is a byproduct, rather than an enabler, of the architecture. This difference distinguishes SOA from other distributed computing architectures that came before it, such as client/server or traditional n-tier. Both of those earlier architectures specified logical tiers and the connections between them, using middleware as the glue that created connections between systems in a tier and across tiers – and the more tiers, the more "glue", typically with a separate software product mediating communication between each tier. In the previous n-tier architectures any combination of data across systems either occurred "at the glass" (i.e., on the workstation screen) or through complex middleware that would need to do conversions of data definitions between disparate data from multiple systems on the fly. In these architectures the glue is required to perform the complex transformations of data required to allow integration across independently developed processing silos. This is a core problem with traditional Enterprise Application Integration (EAI) implementations where potentially there would be a need for n*(n-1) data transformations. The middleware used in these architectures must also orchestrate the processing which is hard-coded into the applications and which must be modified every time that

[6] It should be noted that not all external interfaces will be able to be exposed as services as changes to external services will need to be based on agreement of both the VA and the external party. For practical reasons many external interfaces will need to be support based on their current formats and protocols because of an inability to ensure that the organization with which the interface is maintained will change its interface.

business processes change. It is the continual addition of layer upon layer of this control logic – each layer being added to handle some new condition which has arisen or to allow reuse of code for a new function – that makes many legacy applications so very complicated.

Because the business process control has been externalized from the services, external Business Process Management tools (e.g., workflow engines or Business Process Orchestration tools) may be used to control process and workflow. When such tools are used, the business process can be specified in a standard manner using the Business Process Execution Language (BPEL) – a Web services based standard for specifying business process flows. Many tools exist which can perform process orchestrations written in BPEL. The standards for BPEL are still evolving and there are differences in implementation.

While the formal use of BPEL would require the use of the full Web services stack, this is not recommended for high volume and relatively static workflows. Many of the BPEL enabled products are able to allow input of process workflows / orchestrations in the BPEL language while allowing the underlying services to be based on protocols other than those specified by the Web services stack.

Major goals of the implementation of the SOA include:

- To provide information and functional sharing across VA.
- Reduce the implementation of duplicative interfaces with external organizations and the implementation of the same capabilities in multiple VA applications.
- Foster greater reuse of existing services that reduce cost and maximize application efficiencies.
- All modernized applications will be implemented as a collection of services and that services are not just used to share information between applications or between agencies.
- Allow Major Initiatives to rationalize and modernize those systems without impacting users of the information.
- Allow the application systems and databases to be updated, merged, and / or rationalized.

The basic tenets of the SOA include:

1. An SOA is an architecture in which business functionality is provided by a set of discrete, communicating ‘services’ rather than by monolithic applications. Applications are assembled and business processes are performed through the orchestration of business services. Individual business services may be included in multiple orchestrations.
2. The SOA is oriented towards the sharing of information between VA applications, between VA and DoD applications, and between VA applications and outside organizations.
3. Data will not be stored in the service in the SAO Services Layer, but will remain in the data layer and will be accessed by services that expose the data at the VA level.
4. The VA SOA is a federation of services and infrastructure that exists at the VA level with component SOAs that support services that are only exposed within a component and the component infrastructure that supports those services. The VA Business Areas and MIs have flexibility in the implementation of their SOAs but must federate with the VA SOA. Further, certain functions such as interfaces to external organizations are reserved for the VA SOA.
5. All data is VA data and shall be shared with all VA users authorized to see the data and a need to access the data.

6. The steward for the authoritative instance of the data shall provide services that allow other VA users access to the data.
7. Each service will support interface(s) to the ESB using one or more of the above mechanisms. Services are not limited in the number of the specified mechanisms that they can support.
8. Service security will be provided by a combination of digital signatures and security domains. The VA SOA distinguishes between authentication and authorization. Service requests will be authenticated using digital signatures upon entry to a security domain and therefore, will not require re-authentication while remaining in the security domain. Authorization will be performed by the application that is the steward for the data. All user access to data will be authorized by the application from which the data is being accessed.

2.4.3. Service Centricity

An SOA replaces the traditional integration-centric mentality with a service-centric mentality. In this view, the services form the crux of the application. Because the services are discrete units of functionality and contain minimal flow or control logic – because the process flow (workflow) has been externalized and the only flow or control logic required is the flow and control logic internal to the service – the business analyst can compose the discrete services to support business processes and thereby manage business logic in a declarative fashion using a process choreography or workflow tool. These tools allow the business analyst to combine existing services, manual processes, and newly defined processes into a processing stream to support a business process. The definitions for the newly defined processes can then be provided to programmers for implementation. Using this paradigm, all of the control logic governing the order in which services are performed and the circumstances under which are governed is in the workflow engine or business process choreography tool, and therefore, is not internal to the service. This allows reuse of services without the intertwining of business logic for different functions that is common in most legacy applications.

Business process choreography deals with the control and sequencing of processes in a workflow, a workflow designed to accomplish some business function. An SOA can be used to support business workflows and to tie services to business functions, which ensures that there will be a business owner for each service and that the services support the business needs. Each atomic service will have an owner. Each time a set of services is choreographed or orchestrated a new service is created which will have an owner. That service can then be included in further orchestrations.

The technical attributes of the technology underlying these functional services (making sure they are available, scalable and otherwise meet the service-level agreements and other policies that apply to them) can be managed separately from their functional and business process attributes. The act of integration in this world-view moves to the process level, where business analysts and business process architects connect services to one another as they compose processes. The technical infrastructure that enables this composition to take place therefore no longer centers on integration in the sense of connecting systems or applications together, but rather on building and supporting the services that the business needs.

It is the goal of this SOA that all modernized applications will be implemented as a collection of services and that services are not just used to share information between applications or between agencies. Service interfaces will be provided for legacy systems to allow the functionality inherent in legacy applications to be accessed as services. In the context of this SOA, an application is a collection of software services that automates a business function. The approach for collecting and

grouping these software services depends on the specifics of the business functions being automated and the relationships and interfaces between those functions.

It is recognized that many modernized applications will be implemented either in whole or in part using COTS products. Virtually all of the major vendors of business oriented, functional COTS are migrating their products to SOAs, or at least exposing their products' functionality as services. While the goal is to move to a full SOA where all modernized applications are deployed as services, the extent to which COTS functionality is or can be exposed, as services will be dependent upon the degree to which the COTS vendors move their products to SOAs. There is no intention that COTS software be modified to conform to the SOA. If a COTS package does not expose functionality as services, then a service interface may be placed in front of the COTS application to expose its functionality as a set of services for external consumption, just as is done for legacy mainframe systems.

Applications are to be based on a loosely-coupled, autonomous, services oriented processing model. Atomic services are defined as "small", discrete programs that do a "small" piece of discrete work, which then may be composited into larger services to provide significant functional capabilities. Services are either atomic services or composite services. Composite services are orchestrations that include at least two services, each of which may be atomic or composite services. Services can be combined with control logic to build applications. The control logic needed to combine atomic services into composite services and atomic or composite services into applications is contained in the orchestration logic external to the functional services. Services are designed to be used "on demand" by many applications.

Applications will consist of a set of loosely coupled services that will be used both internally and exposed to users external to the application through a number of mechanisms including a secure messaging service which will be mediated by an Enterprise Service Bus. The secure messaging mechanism and ESB exist at lower levels in the stack and are described as parts of those layers.

The requirement for message based communication does not extend to communication between internal components of a COTS application or between a COTS application and its proprietary data store. COTS applications are not required to replace direct API or SQL calls to databases maintained only for those COTS applications. COTS applications will communicate with other portions of the enterprise through the messaging service and will access services that exist in the enterprise in the same manner as custom developed applications. Interfaces to COTS applications will be through services exposed on the ESB. This requirement can be met by placing an adapter between the COTS application and the messaging service.

Services external to an application communicate and cooperate via the network by calling other services through a message-based interface. The only access to services external to an application is through the message interface. The internal details of the service must be hidden from users of the service so that the internal structure of the service, the platform on which the service is hosted, or the sources of data for the service can change without impacting any applications calling the service.

Services allow developers to focus on application specific business and presentation logic without having to worry about the mechanisms by which the services are provided. The service hides the complexity of the infrastructure and allows incompatible technologies to co-exist and cooperate. The limited access points provided by a service allow for tightly controlled security. Each service is documented, tested, and certified by security once. However, privacy compliance reviews are conducted each time a service is reused to ensure privacy compliance requirements are met across the

full data and data usage life cycle. Services can be upgraded and scaled independently because of this loose coupling.

2.4.4. Service Contracts

One major aspect of an SOA is the concept of a service contract. A service contract can be thought of as fulfilling the role of the Interface Control Document (ICD) and Memorandum of Understanding (MOU) in traditional systems as it specifies the connection between the service provider and the level of service that the consumer will receive. However, the service contract goes much further than the ICD as it specifies all aspects of the manner in which the service will be provided and consumed, not just the flows of information across the interface. This specifically includes aspects of the service included in Service Level Agreements (SLAs) such as performance, availability, accessibility, etc. The service contract is a specification of the functionality of the service, but should not specify the manner in which the service provider will provide the service. In fact, the service provider is free to change any aspect of the way in which the service is implemented as long as the service contract continues to be honored. The details of the implementation of a serv ice should be hidden from the consumer of the service and the consumer of the service should not make any assumptions nor have any dependencies on the precise manner in which the service is implemented.

Web Services Definition Language (WSDL) is a standard form of machine-readable service contract that can be entered into a directory and be discovered by service consumers. Services contracts as used in this document go beyond the bare bones service contracts specified by the WSDL standard contracts which are designed to be machine-readable and serve as a mechanism for systems to locate and access services without human intervention.

3. EAA Layered Model

The EAA will be based on a series of layers with each layer and its sub-layers containing services or capabilities of a given type. This section provides an overview of the layered model and the next sections describe each of the individual layers.

3.1. Basis of the EAA Layered Model

The VA EAA both provides the building codes that describe the manner in which VA systems will be implemented and the environment in which they are run. It builds on modern directions in systems design and incorporates those directions in a formal way into the design and development of VA systems. It is based on a number of principles and constraints that reflect those major directions in architecture. These include:

1. The EAA shall be defined as a series of layers and sub-layers, with each layer or sub-layer containing services of a single type (e.g., application, messaging, systems management etc.).
2. Communication between services within a layer or sub-layer shall be through formal messages, with the format and protocol of those massages being dependent upon the specific layer and sub-layer at which the services reside.
3. Application systems shall be designed based on a series of services running in a virtual environment.
4. The virtual environment shall be able to be mapped to one or more physical environments.

5. Transformations will be provided at each layer to transform the virtual services that are visible to developers into the physical services provided by VA infrastructure or operational environments.
6. The transformations shall allow services running in the physical environment to be changed or replaced without impact to the services running in the virtual environment.

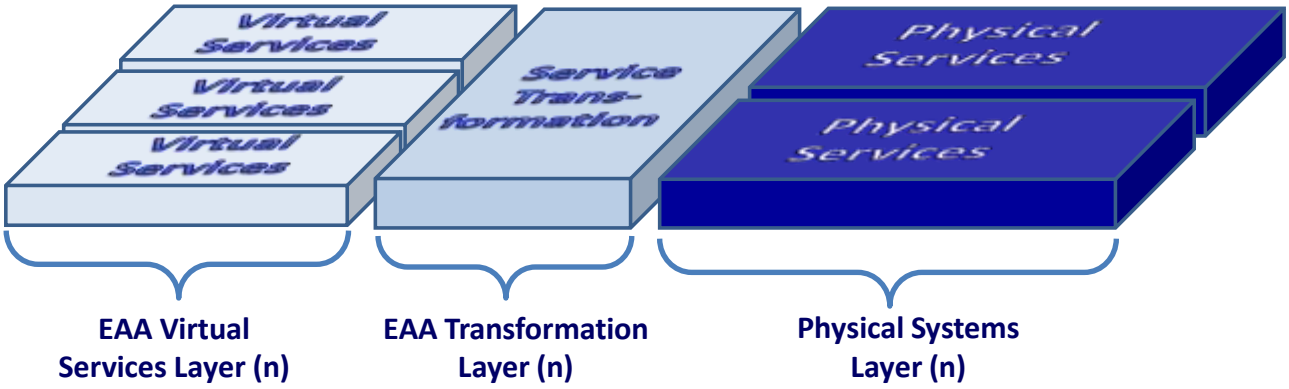

Figure 3 – Virtual and Physical Layers in EAA

7. Application development projects shall only develop code at the highest layer of the layered model described below (the SOA Services Layer), but may access, either explicitly or implicitly, virtual services at the next lower levels.
8. The Virtual environments are expected to be much more stable than the physical environments in that the physical environments are expected to change over time (e.g., because of new software releases, new hardware models, or because the physical sub layer is modernized).
9. Virtual layers may be sliced into a series of sub-layers with each sub-layer containing a proper subset of the services appropriate to that layer. For example, the SOA Services Layer includes isolated sub-layers for presentation, business logic, and data services where the business logic services .
10. There shall only be a single interface between VA and any system external to the VA and this interface will be shared by all VA systems need access to that external system.

3.2. Definition of the EAA Service Layers

The EAA is composed of eight layers:

- **SOA Services Layer** – Is the level in which application code resides. Applications are to be written using an SOA-based approach, rather than a monolithic application style. SOA services will communicate using messages that are defined in the level below. The SOA Services Layer will include a data services sub-layer that provides logical access to data. The physical sub layer contains both native services and the VA legacy applications. The transformation sub layer provides service facades that expose VA legacy system functionality as services for the modernized services to access.

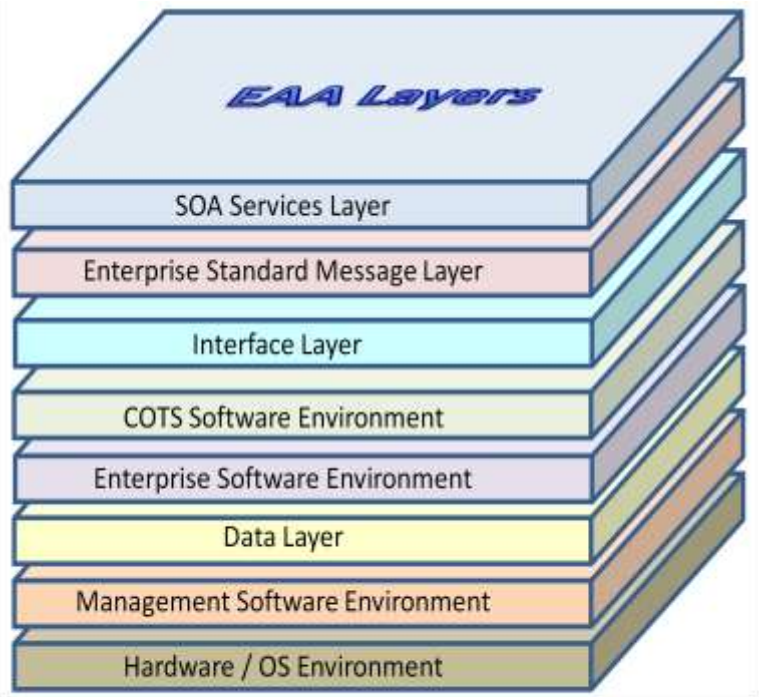

Figure 4 – Overview of EAA Layers

- **Enterprise Standard Message Layer** – Defines the logical messages that are used to transmit data. All messages at the virtual sub-layer shall use fully harmonized VA standard messages. The transformation sub-layer transforms VA standard messages to the logical formats with which VA communicates with the external world. The form, format and protocol of the messages with external entities will vary depending upon where the communicating services reside. For example, NIEM may be used to communicate between VA business services and services in other federal agencies or HL7 may be used between internal[7] and external health applications. The Standard message Layer is a logical layer which deals with logical message formats such as HL7, NIEM, EDIFACT etc., and not physical transports such as IBM WebSphere MQ Series (IBM MQ) or HTTP etc. **Interface Layer** – provides an SOA interface to external systems and devices. Neither external interfaces nor physical devices (e.g., bar code readers, MRI scanners) will provide VA SOA compliant interfaces. The interface layer provides service facades for each external system or physical device. Since there will only be a single interface between VA and any external organization, so any SOA service facade built for an external interface will be shared with all VA systems requiring access to the external system. While the Standard Message Layer transforms Standard Messages to external protocols, the interface layer provides the physical interface over which those messages are exchanged.
- **COTS[8] Software Layer** – includes functional software, software that satisfies business requirements of the application and is specific to that application. This layer specifically does not include COTS software that can be used on an enterprise basis or which provides general, non-application specific capabilities.

[7] While all internal messages will be based on the fully harmonized standard VA messages with data element names and definitions from the Enterprise Logical Data Model, in the health care arena those data element names and definitions could be based on HL7.

[8] As used in this document, the term COTS refers both to Commercial-Off-The-Shelf software and Government-Off-The Shelf (GOTS) software.

Enterprise Software Environment – houses much of the middleware and large-scale COTS products – including both functional COTS packages and systems oriented COTS such as workflow management, knowledge management, and document management software. It is the middleware level where the physical message passing software resides as well as all other middleware-like software, including Enterprise Service Bus(es) (ESBs) etc.

- **Data Layer** – The data layer includes access to the physical data layer and includes logical constructs such as the location of the Copy of Record (COR) for the data and physical constructs such as the physical databases and DBMS'. The object is to maintain sufficient separation between the physical database design and the data services visible to the SOA services such that the physical database design (e.g., columns in a table, the number of tables, and even the DBMS being used) can change without impact to the data services that exist at the SOA services level.
- **Management Software Layer** – Includes all non-operating system software not included in any of the layers described above. This would include a wide range of operating system-like services such as Identity Management Services, Backup and Recovery Services, and other such system oriented services not provided as part of the operating system or by the operating system vendor.
- **Hardware / OS Level** – Describes the most basic physical environment in which the application software will run. Each VA datacenter or cloud computing services provider will implement a standard physical stack and the transformation layer will provide the mapping between the virtual image that applications are designed and deployed to and the physical environment in which they will actually run.

3.3. Primary Layers, Sub-Layers, and Towers

The primary layers used to define the EAA were identified in the section above. Each of those primary layers is divided into a series of sub-layers providing additional information about the primary layer. These stacks of sub-layers form a series of stacks that are described in the section below.

3.3.1. Virtual Stack, Transformations, and Physical Stack

Each of the layers described above is divided into the three sub-layers illustrated in Figure 5. The stacks are:

- **Virtual Stack** – Is the stack visible to application developers and the environment in which they develop systems. This will be a standard set of services that is independent of the physical environment in which an application is deployed.
- **Transformation** – Describes the mechanism by which the services in the virtual stack are mapped to the services in the physical stack. An example of a transformation would include a "service façade" placed in front of a legacy system to make the legacy monolithic system appear to the logical services as a modernized service.

- **Physical Stack** – Is the physical environment in which the systems operate. The same virtual services may operate in a number of different physical environments or a physical environment that changes over time as new versions of software or new hardware is introduced.

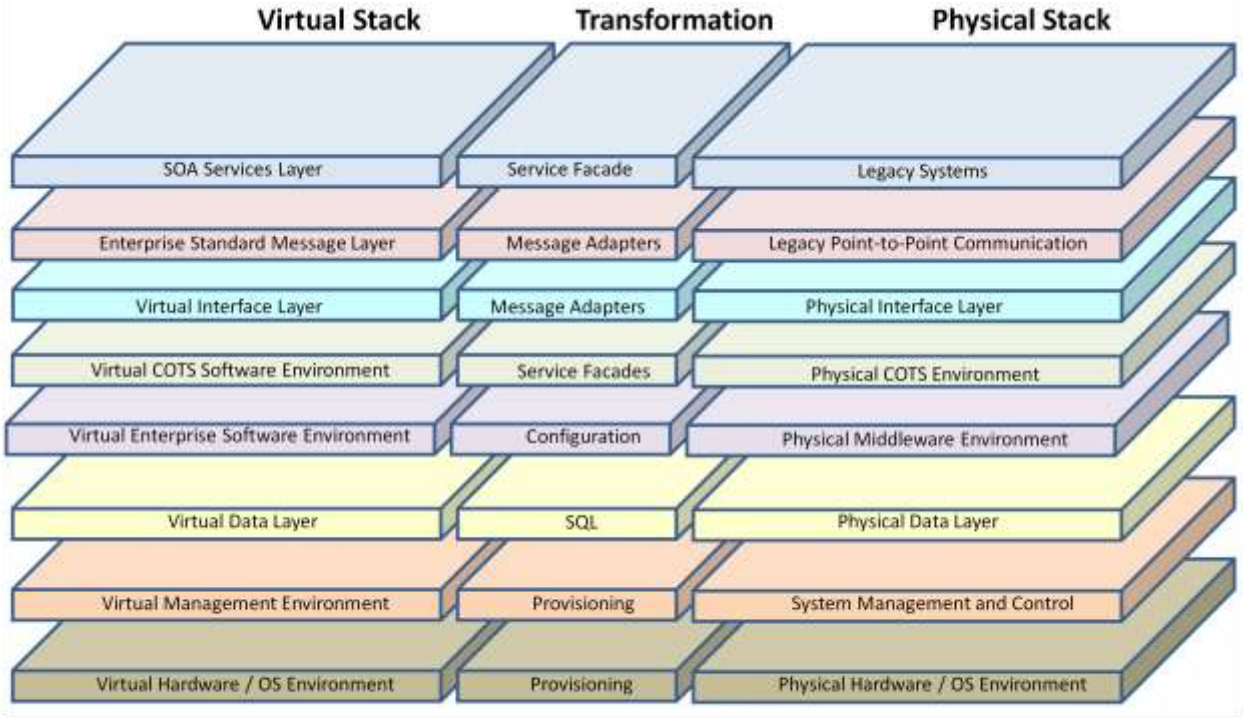

Figure 5 – Overview of the Virtual, Transformation, and Physical Stacks

The details describing the sub-layers making up each of the stacks will be described in the sections below. The definition of the Virtual Stack, the Physical Stack are independent of one another and the Transformation Stack provides the mapping between the between the two. A core feature of this EAA is that since the Virtual Stack is defined independently of the Physical Stack, the Virtual Stack – the SOA services can be designed and developed in a "green field" environment. The SOA services can be designed as a new start, independent of the legacy systems and depend upon the Transformation Stack to provide the mapping between the modernized and the legacy systems.

The Physical Stack describes the current physical environment – including both legacy elements and newly-built, modernized, components although partitioned and described in a somewhat different manner than typical. Although the description of the current environment may differ in form from the way it has traditionally been described, this is to better align it with elements in the Virtual Stack.

One major issue that arises in introducing new capabilities or architectural elements, such as an SOA, is the need for ensuring the interoperability of the new elements with existing systems. While the legacy system operates and the new system could operate once it was developed, the issue occurs when parts of the legacy system must interoperate with parts of the new system. Further complications occur when the legacy system and new systems are based on different technologies and / or design and different development philosophies.

The introduction of the Virtual Stack is intended to provide a "green field" for the development of new systems and new capabilities without the need for continuing the technical direction used for the legacy systems, and to shield the environment in which the new systems are being developed from the existing technologies and designs used in the legacy systems. Of course the newly developed systems built using the new technologies must interoperate with the legacy systems to provide enhanced functional capabilities. The emphasis here is on interoperability rather than full integration with the legacy systems. The transformation layer is intended to provide the mapping between the virtual layer and the physical layer to provide the required interoperability. The transformation layer provides the loose coupling between the modernized virtual environment and the legacy physical environment. The transformation layer will be different at each layer in the stack and will be more or less complex depending on the degree of difference between the virtual and physical layers.

This EAA does not provide much detail as to the nature and structure of the layers in the Physical Stack. It is not the purpose of this EAA to provide a full description of the current legacy environment, but a full understanding of the current physical environment will be required to develop the transformation layer services. The lower layers of the physical stack are also not described in

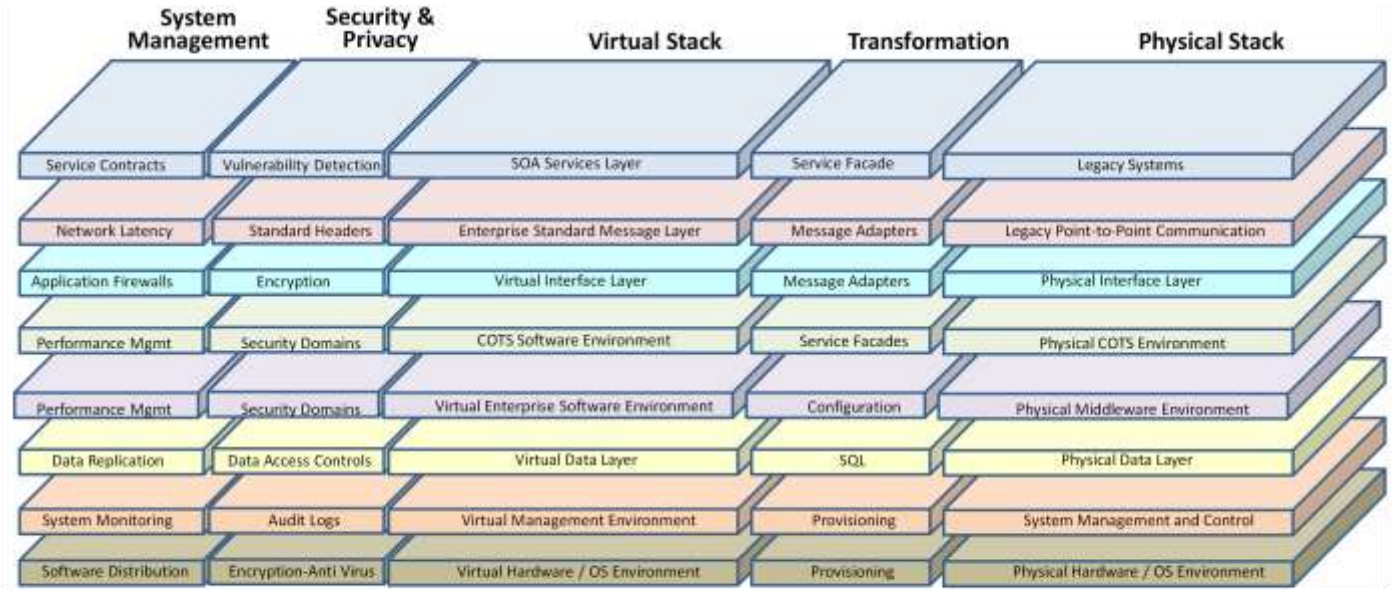

Figure 6 – Overview of the Security and Privacy and Systems Management Stacks

great detail as they relate to the physical infrastructure and are the responsibility of organizations other than the application development organization and may change frequently.

3.3.2. Security and Privacy and Systems Management Stack

In addition to the three stacks described in Figure 5, there are two additional stacks that include sub-layers in each of the primary layers (Figure 6).

- **System Management** – Services used to manage the services at each level of the stack.
- **Security and Privacy** – Services used to provide the security and integrity at each level of the stack. This can range from physical security, such as firewalls at the lowest level of the stack, to digital signatures on service messages at higher levels of the stack.

These two stacks exist and are implemented to some extent at each layer, albeit differently at each layer. For example, the security services provided at the SOA services level are fundamentally different than the security services provided at the Hardware / Operating Services Layer.

3.3.3. Implementation of the Services Stack

Previous sections described the layers and some of the sub-layers in the EAA. While the subject of this document is the application architecture, it includes not just the application functional logic, but the specification of the environments used to support those applications and the parts of the environment that are visible to the application developer. Each of the environments (the layers of the model) needs to be specified and implemented – just not necessarily by the application developers. In fact, only a small part of the environment will be developed by the application development projects. Application development projects, whether developing new applications and updates to existing applications will only develop code at the SOA Services Layer, message adapters at the Enterprise Standard Message layer, and service facades for the Virtual Interface Layer and not develop code at any lower level without a specific waiver or without specific permission from the head of the VA systems development organization.

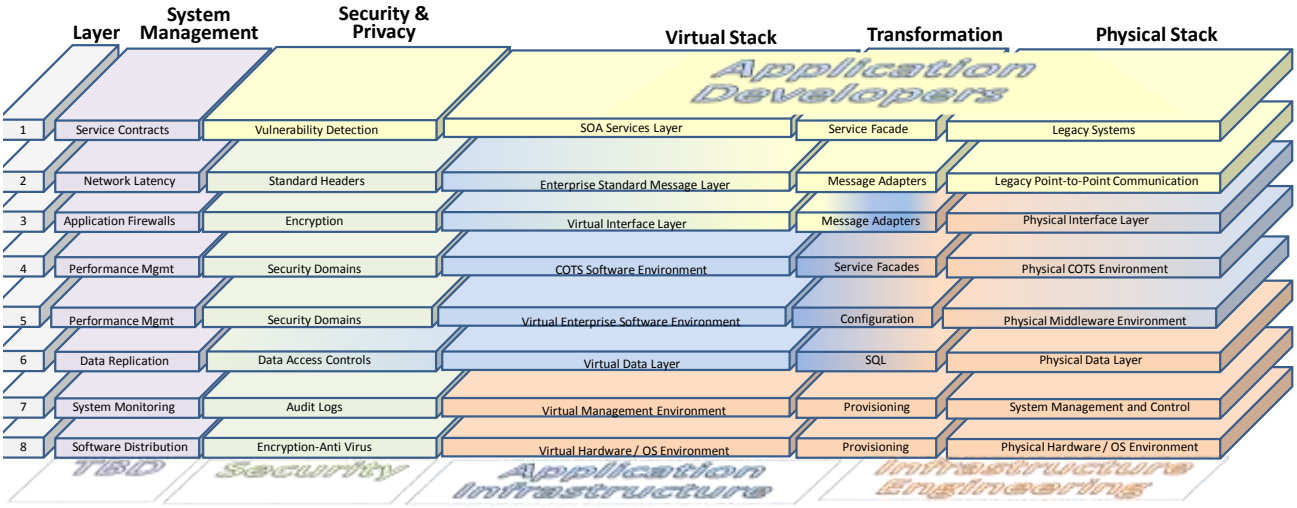

Figure 7 – Responsibility for Implementation of the Services Stack

Software implementing all layers of the virtual, physical, and transformation sub layers below the SOA Services Layer will developed by specialized groups within the application development organization or by other groups such as the infrastructure group or by security. Responsibility for development of the sub-layers is described in

. The traditional application development project teams would only be responsible for the development of the services in the top layer – the SOA services stack. This includes:

- Development of SOA services in the virtual services sub-layer.
- Update and maintenance of the of the legacy systems.

- Design and development of the service facades that serve as the transformation between the newly developed SOA services and the legacy systems that are still in operation.

The layers below the SOA Services Layer are essentially one type of infrastructure or another used to support the SOA services or the layers below it. As such, they will be built by infrastructure groups[9] either in the application development group and the infrastructure and system development and operations groups. Some sub-layers may be developed and managed by a mixture of the application developers and infrastructure developers. In some cases it is not yet determined which group would support the development of the sub-layer. In the case of the security stack, some aspect of which will impact each layer of the stack and the organization responsible for implementing the security controls could both vary between layers and within a layer. For example, developers will be responsible for ensuring that code does not contain known vulnerabilities, but a security group would be responsible for auditing the code to ensure that there are no vulnerabilities. In many instances groups other than the security group will be responsible for the development of the security services and capabilities at each layer.

3.3.4. Relationship Between the EAA Stacks and OneVA Artifacts

Earlier sections of this document discussed the general relationship between the EAA and other OneVA artifacts. This section relates the some specific aspects of the OneVA EA and the EAA stacks that are the basis for this EAA.

3.3.5. Relationship Between the EAA Stacks and the TRM

The VA TRM provides a list of products and standards that are allowed for use in the design and development of VA systems. It is meant to be prescriptive in that products that are not included in the TRM may not be used in VA systems unless and until they are incorporated into the TRM.

Where the TRM provides the list of materials that can be used to build a system, this EAA provides the "building codes" that are used to control the building of systems for use in the enterprise and describe how the various allowed materials can be used. While there are a large number of products listed in the TRM, most of these will not be visible to the application development teams for one of three reasons. First, they may exist in one of the physical sub-layers and thus not be directly visible in the virtual sub-layers to which the application systems and services developers have access. Second, they exist at lower levels of the stacks that are not directly visible to the systems and service developers operating at the higher levels of the stack. Or, three they might not be used in the standard virtual or physical layers and thus, are not used in VA applications. Because the virtual layers are instantiated through a set of standard layers, these standard layers will contain only a very small subset of the products in the TRM. Thus, only a very small portion of the products that are allowed by the TRM will be used in VA solutions.

Application developers will develop in a virtual environment without regard to the technology products that are used to support that environment. The datacenters will identify the physical stack that they will support and that provide the infrastructure to support the services at the virtual layer.

[9] Because organizations change over time, there is an attempt to avoid the use of organization names, although some are used for clarity.

The application developers will see a standard virtual environment, regardless of the physical environment that is used to at a specific operational site. The developers will see a standard, common set of infrastructure services and environments that will be developed to provide the capabilities from the lower levels.

The use of a standard set of virtual environments not only shields the application developers from the physical environment in which their applications will run, but greatly reduces their dependencies on the physical environment and their need to request changes to those physical environments. The use of formal services interfaces between the SOA layer applications and services and the infrastructure capabilities more readily allows changes to the underlying COTS packages while minimizing impact to the SOA services.

3.3.6. Relationship the EAA Stacks and the Release Architecture

Each VA datacenter will specify a single stack of products that the datacenter will support and that it can rapidly provision as the need arises. These stacks can be expected to be a very small subset of the total set of products included in the TRM. The direction to VA system development and enhancement projects should not expect that any technologies, other than those specified in the RA will be supported or available to them. This is a very explicit recognition that the development and publication of the technology stacks in the RA will effectively prohibit the use of conflicting products that a project has a desire to use – even though those conflicting products may be allowed by the TRM. The products that are in the stack for each of the datacenters will be published in the RA which will be updated as the products in the stacks for each of the datacenters is updated.

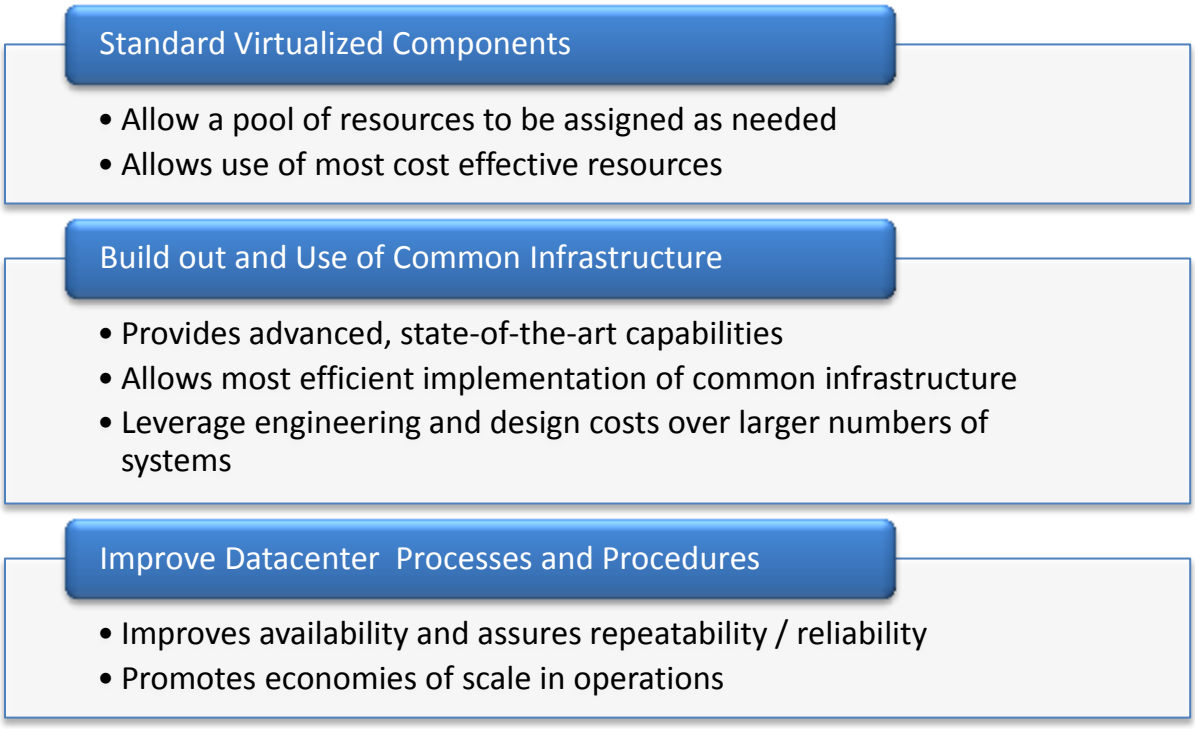

Figure 8 – Advantages in Use of Standard Technology Stacks

The products in the datacenter stacks will cover the range of IT products that are required to support the development and operation of modern application systems. As such these products will exist in multiple layers in the product hierarchy – i.e., in different layers of the EAA as defined below. The

products in the technology stacks specified in the RA will meet the requirements for the individual layers in the EAA and will be part of the physical environment for the appropriate levels. Use of the products specified in the RA will be mandatory for all VA development efforts – whether performed inhouse or by a contractor to VA. Deviations from the products in the approved stacks can be expected to cause VA to incur significant costs and should be avoided.

It is expected that each of VA datacenters will support a somewhat different stack of products to provide the required infrastructure services at each level of the stack. These products will be identified in the RA and exist in the physical stack. The transformations used to hide the physical products in the RA stacks from the services in the virtual stack can be used to provide a uniform set of services – a common set of service facades so that each of the datacenter-specific technology stacks appears the same.

4. Application Architecture Layers

The VA EAA is described in terms of eight primary layers, each with three primary sub-layers describing a virtual environment, a physical environment, and a set of transformations between the two. Each of the sub-layers may in turn be further sub-divided. Each of the layers and sub-layers are described in this section.

4.1. Layer 1 – SOA Services Layer

Layer 1, the SOA Services Layer, is the layer in which the VA functional (i.e. business) applications reside, and which includes all of the functional logic needed to implement the systems and services that are required to meet VA business requirements. The functional logic includes presentation services, business services, and data services. Layers below this layer provide infrastructure and application support for the functional logic that runs in this layer. This layer is the layer in which application programmers code the business, presentation, and data services that compose the application. As noted in Figure 9, the SOA Services Layer, like all other layers is composed of a virtual services layer, a transformation layer, and a physical services sub-layer. Each of these sub-layers later in this section.

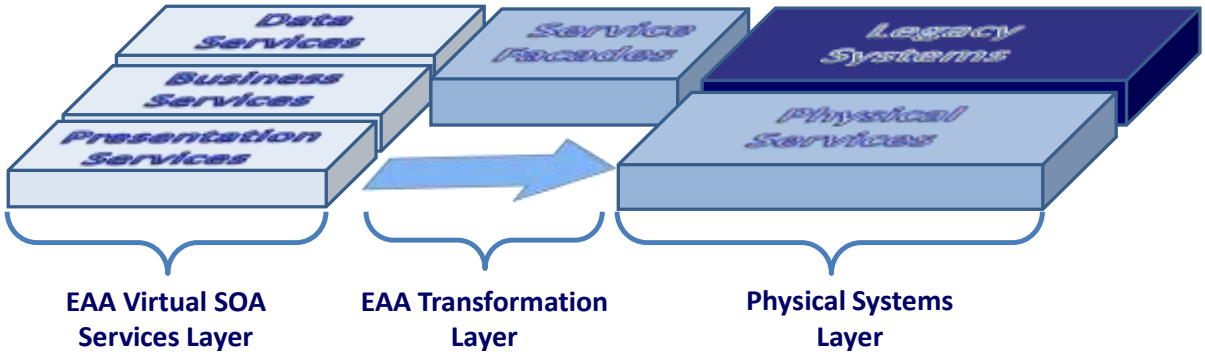

Figure 9 – SOA Services Layer

The virtual services in the SOA Services Layer are implemented as services as per the VA SOA. An overview of the VA SOA is provided in this section with a more detailed description provided in the VA Service Oriented Architecture (SOA): Technical Framework. The Virtual Services Sub-Layer is just that, a layer that contains virtual specifications of services – absent the systems / middleware capabilities that would be required to allow the services to run. An application in the Virtual SOA Services Sub-Layer would include an orchestration of a set of services contained in that layer. The

services and orchestration would be specified using standard notations, such as Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL), without reference to the actual middleware product that will provide the BPMN and BPEL execution. The capabilities of the products that are used to allow the actual execution of the services and orchestration are specified in the lower levels of the architecture.

4.1.1. Virtual SOA Services Sub-Layer

The Virtual Services Sub-Layer is the layer in which new services are developed and in which they operate. It can be viewed as a green field in which new services can be developed without specific reference to the existing applications. The transformation layer will provide the interface between the new services developed in the Virtual Services Sub-Layer interface to the legacy systems through the transformation sub-layer. Services in the Virtual Services Sub-Layer are developed based on the VA SOA.

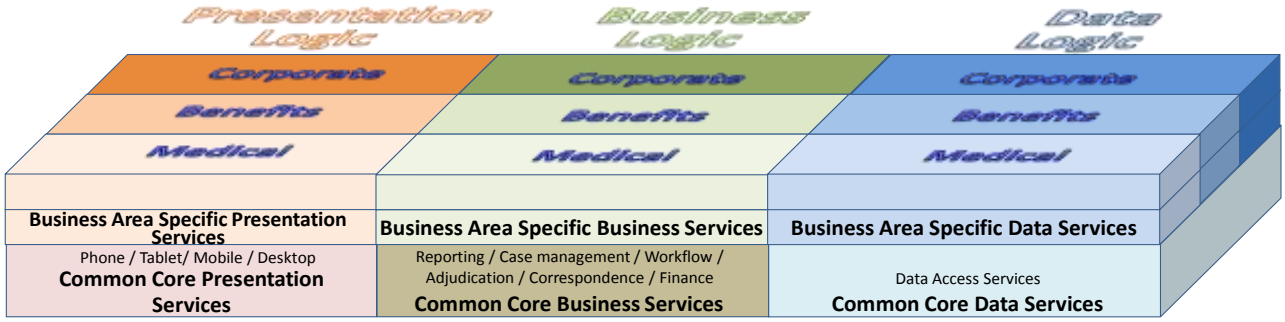

Figure 10 – SOA Services Sub-Layers

4.1.2. Presentation Logic, Business Logic, and Data Logic Sub-Layers

This section describes the presentation logic, business logic, and data logic sub-layers. All services developed by applications developers will be in these sub-layers.

4.1.2.1. Presentation Logic

As used in this section, external user specifically refers to people that provide or receive information from the system. Presentation services take information produced in the business logic sub-layer and provide it external users and systems or receive information from those users or systems.

4.1.2.1.1. Isolation of Information

One role of the presentation logic is to allow information developed by a single business logic service to be provided to users through a wide variety of user interface devices including workstations, laptops, tablets, smart phones, kiosks or other user interface devices and to have the output optimized for each type of device. Conversely, the presentation logic could be used to allow a single presentation service to provide the user information from a number of business logic services.

A single business service might deliver its results through multiple presentation services because of the different information needs of the recipients as well as the differences in the devices used to access the data. For example, while a single business logic service may process a patient medical record or the results of a test or series of tests, different presentation services might provide that information to the doctor, the patient or other interested parties on any of a number of devices.

A presentation layer service may only support users at one security or access level or who require access to the same types of PII or PHI. This approach is also necessary to allow there to be an underlying set of services at the business layer that perform processing using data more sensitive than can be released to the user. For example a calendaring system may only show a patient when a doctor has open time for appointments or when that patient is scheduled to see that doctor, other users may be able to see when the doctor is in the hospital or not, while others may be able to see exactly what the doctor has scheduled and who the doctor is scheduled to see. But all of the information is presented based on the same underlying business information.

A key factor related to the design of this layer is the use of defense in depth to ensure that even were an external user to penetrate one or more devices in the presentation layer, the processors in the business logic layer would still be protected. Where an application has implemented a portal to service external users that has already been certified and accredited by security, this portal should be used for interfaces that provide sensitive or privacy data to external users.

4.1.2.1.2. **End-User Interface**

The presentation logic implements the end-user interface which may be through a web browser on a laptop or an "app" on a tablet or smart phone. It focuses on the general interaction with the end user and addresses areas such as screen formatting, window management, input handling, mouse handling, and display-related data transformation. The services implemented at this level will be based on the following principles:

- The Web browser will be the only user workstation / laptop interface to VA applications.
- Fat clients (that is, non-Web based clients) are not allowed for workstation type devices.
- Special web-based apps may be used for non-workstation / laptop end user devices.
- All presentation services will provide a common look and feel to the users subject to the constraints of the user device that is being used.
- To the maximum extent possible COTS products shall be configured to resemble the VA common user interface look and feel.
- All user interface services shall comply with Section 508 of the Americans with Disabilities Act (508).

4.1.2.2. Business Logic

The Business Logic Layer includes the core computational services that perform the bulk of the work for the application systems. It is at this level that the core business processes will be performed and all of the services except for externally exposed Web services, presentation services, and external interfaces will reside.

All VA services-based business logic will reside in the business logic layer. The applications that reside in the business logic sub-layer are implemented as services and can be either COTS-based or custom developed. Non-services based COTS packages reside at a lower level of the architecture, so it is only the native COTS services, service interfaces (i.e., service facades) to the COTS packages or services facades supplied for the COTS package that reside at this layer. These services implement the complete set of business rules and business processing required by the application. Business applications that run in this sub-layer are developed and / or integrated from reusable services and will be available to be orchestrated into more complex composite services.

Modernized applications will be developed to be compliant with the SOA and expose all interfaces as services in a native manner or will migrate to that approach over time. It is important that all future increments and releases of these systems conform to the SOA. Business logic services will be implemented at this level. Applications being built at this level will not interact with non-component applications directly, but will do so through the presentation or external systems interface layer services.

Business applications employ the following design concepts and associated guiding principles:

- All business logic is implemented as either atomic or composite services.
- Atomic services are not composed of other services.
- Composite services are composed by orchestrating atomic and / or composite services.
- Business services will not directly access data and will not rely on any knowledge of the physical structure of the data as the physical structure of the data may change without notification to the application.
- All service requests and service responses will be based on standard messages as described in the message processing layer.
- In the short term, it shall be the application's responsibility for converting message contents from the application specific formats to standard message formats and from standard message formats to the application specific formats.
- In the longer term, the application's native message and data formats shall be based on the standard selected for communication with other services.

4.1.2.3. Data Logic

The data layer will be maintained in the SOA with special services providing data access. In a non-SOA environment, there is a concept of an application owning a database and controlling access to that database, this distinction can be lost in an SOA. Since an application is a collection of services, some of the services will be accessing data, others will be processing data, and others will be validating potential changes to the database. Any of these services may be used in a number of applications, so as the boundaries of an application are relaxed, the need for well-defined services for data access must be strengthened. There must be strict controls as to which services can read specific databases tables and which services can create, update, or delete records. Much of this work will need to be performed in the Data Layer, specifically those that create, delete, or edit physical records. However, some of the work is performed in the Data Logic Sub-Layer of the Virtual Services Layer. This would include receiving data from the data layer and presenting this data to the business logic services. It may also include a variety of data validation and editing processes to assure that data that is supplied to the database is valid.

The collection of information from external sources, the analysis and processing of this information and the development of a logical record will be done in this sub-layer, while the physical changes to the data base will be performed in the Data Layer. The results of a test may be viewed and processed as a single logical record, but be stored in multiple records in multiple tables and even in multiple databases and DBMSs in multiple locations. The processing of the logical record occurs at the Data Logic Sub-Layer of the Virtual Services Layer while the services that perform the actual table updates are at the Data Layer described below.

The data layer manages access to the data and is based on the use of services to expose data. The data layer will be responsible for separating the business logic from the data and for hiding the physical structure of the data from the business logic services. The data layer will be developed based on the following principles:

- There will be a separation between data logic and business logic and only data logic will be able to directly access the data.
- Services at the SOA Layer will have no knowledge of, or access to, the physical databases.
- The physical structure of the data may be changed including the choice of DBMS, table partitioning, or physical record structure. This will be reflected in Data Layer services, not SOA Layer services
- The Technical Steward for each instance of authoritative data shall expose services that will make that data available to all authorized users in VA.
- Data services that expose data outside of the application will expose data based on standard data definitions regardless of the data definitions used in the physical database or within the application.
- The Technical Steward for an instance of authoritative data will only be required to expose a single service to provide read or update to the data. This service shall expose all data elements that are part of the data class / entity of the ELDM.
- Any updates requested by a data update service will be required to pass all application edits and controls prior to being allowed to update the authoritative instance of the data.

4.1.3. Common Services

One major aspect of the EAA is the use of common services – services that can be used by applications other than the application for which they were developed. There are two aspects to the specification and development of common services, services that can be used by applications throughout VA and services that can be used within a single business area, but cannot or should not be used outside of the business area in which they are developed. Therefore, there will be three classes of services that are developed:

- Application specific services.
- Business area specific services.
- VA-wide services (core common services).

4.1.3.1. Application Specific Services

Application specific services are presentation, business, or data services that are used only within a single application. These services are highly specialized and are applicable only to the application. Because the goal is to maximize service reuse, the application specific services are the least desirable class of services and the goal is to promote as many services as possible to business area or core common service classes. However, because there is some overhead in promoting services to the higher levels, services can only be promoted if there is a legitimate likelihood that the services will be of use to others outside of the application system.

4.1.3.2. Business Area Specific Services

Business area specific services are services that are useful and can be reused in other applications in the specific business area. For example, there will be some services in the health area that are specific to health care and access PHI that for any of a number of reasons may not be appropriate for use in other areas, but which may be of use to multiple applications within the health care arena.

4.1.3.3. Core Common Business Services

Core common services are those services that perform functions that are of use, or are potentially of use, across VA. Initial candidates for CCBS include:

- Reporting.
- Case Management.
- Workflow.
- Correspondence.
- Adjudication.
- Finance.
- Military Information.
- Personal Information
- Contact History
- Verification and Notification

Each of these are high level, coarse grained services and therefore, will be composed of a number of smaller, more fine grained services. Once any of these CCBS are developed, their use will be mandatory for any applications with a need for the capabilities provided by those services. No project should implement any capabilities related to these CCBS except as part of the single implementation of these Core Common Business Capabilities and then only for VA-wide use.

Other functions or capabilities may be added to the list of CCBS over time as the need for other enterprise-wide common business services are identified. As such functions or capabilities are identified; they will be added to the list and will be prohibited for further development by projects. In addition, as projects make plans to develop capabilities for their own use, those capabilities may be designated as CCBS and then must be developed to meet enterprise, not project, requirements.

The CCBS are being described in the SOA Services Layer as they replace Business Logic SOA services, but the CCBS will normally reside in the Software Service Layer which contains most of the COTS and other major software services.

4.1.4. SOA Services Transformation Layer

The SOA Services Transformation Layer connects the Virtual SOA Services Layer to the Physical Services / Legacy Systems Sub-Layer and thus, there will be two components to this transformation layer, a set of service facades that expose the legacy systems and a pass through that maps services in the Virtual SOA Services Layer to native services to native services in the Physical SOA Services layer.

Generally, there should be a one-to-one mapping between services in the Virtual Services Sub-Layer and the Physical Services Sub-Layer. There are no service facades with the services in the Physical Services Sub-Layer as the services already provide a services interface and should be executable based on the capabilities that are provided at the lower levels.

4.1.4.1. Access to Legacy System Functionality

The Legacy Systems Facade Sub-Layer provides a series of service facades in front of specific VA legacy application systems to allow the services that are developed to interact with and interoperate with the legacy systems.

Legacy systems and services that do not conform to the SOA will be exposed through a service facade – a series of adapters that access the legacy data and business logic but appear as SOA services to other SOA services. The details of where the service adapters / service facades reside and how they are implemented are not the topic of this layer in the discussion, but the service adapters should appear to the SOA services as SOA services and any complexity related to the attachment to the legacy systems is to be on the legacy systems side of the adapter so as to be invisible to the SOA services.

A service facade is a program that presents a services interface to the SOA services – communicates with SOA services via VA Standard Messages and communicates with the legacy applications through an API. The service facade looks like a service to the SOA services in the Virtual Sub-Layer and like a legacy application to the legacy application in the Physical Sub-Layer. Further, once the service facade is in place the legacy system can be modified, or even replaced with services, without impact to the SOA services that access the legacy capabilities through the service facade. This replacement of legacy code behind a service facade with SOA services code is called “refactoring.”

Code refactoring is "disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior" undertaken in order to improve some of the nonfunctional attributes of the software. Typically, this is done by applying series of "refactorings", each of which is a change in a computer program's source code that does not modify its functional operation.

This refactoring works particularly well in an SOA environment since an SOA service is like a black box that is specified in terms of the inputs to the service, the outputs from the service, and a transformation i.e., the functional relationship between the service inputs and the outputs. The service contract is a functional description of what the service does, but not how the service performs the transformation or provides the functionality. The user is not aware whether the service provides the functionality by means of a single atomic service or a set of composite service, or whether that changes over the life of the system. Since a service facade appears to be a service, the users of the service, the facade, have no visibility as to whether the code behind the facade is a legacy system or modernized SOA services.

4.1.4.2. Access to Legacy System Interfaces

Although the long term goal would be for all external interfaces to be through the External Interface Layer, current legacy systems links with external systems will be retained and not be required to go through the External Interface initially , as would new links for modernized systems or links that are implemented as services. Legacy interfaces will be access through the service facades that are used to provide access to the legacy applications.

4.1.4.3. Access to Physical Services

Functionality implemented as SOA services will be directly accessible by services being developed in the virtual layer. The access will be without service facades or adapters.

4.1.5. Physical Services / Legacy Systems Sub-Layer

This sub-layer includes two components, the physical services sub-layer that contains the services that have been written to the virtual services running in the virtual services layer and the legacy systems sub-layer that contains the legacy systems.

4.1.5.1. Physical Services Sub-Layer

The Physical Services Sub-Layer contains the physical services that have been implemented, are operational, and which comprise the modernized application systems. This sub-layer includes the same sub-layers as the Virtual SOA Services Sub-Layer including:

- Presentation Logic, Business Logic, and Data Logic Services.
- Application Specific, Business Area, and Core Common Services.

4.1.5.1.1. Physical Presentation Sub-Layer Services

Portals are an increasingly common mechanism for applications to communicate with end-users. These portals can serve as the host for the presentation level services. Presentation services contained in such are portal are normally implemented as portlets. Portlets can be used to allow information from multiple SOA services or applications to present information on a single screen or to allow information entered into one sections of a screen to be input to multiple applications and services. Portal and Portlets lets are the preferred long term physical implementation of the presentation level services. However, prior to the full development of portals and the ESB Presentation Sub-Layer Services will need to be implemented without use of these capabilities

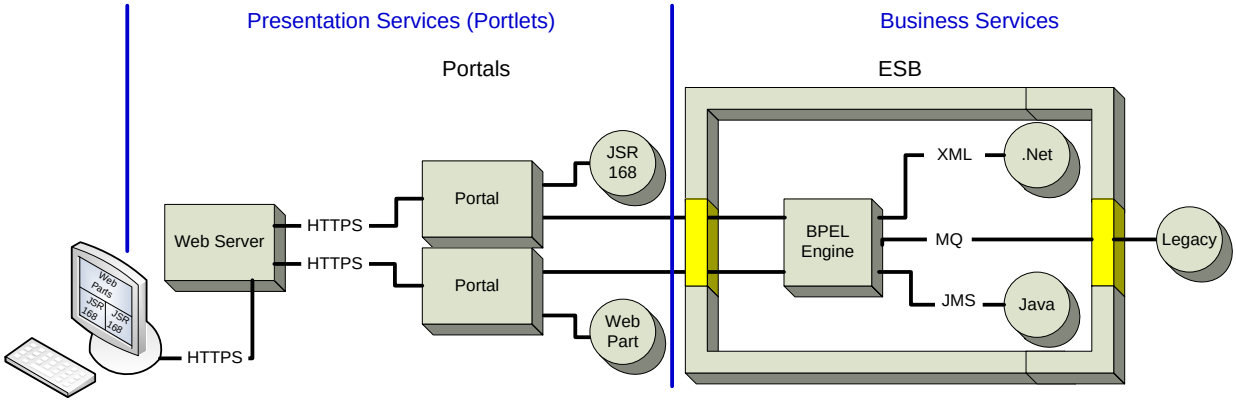

Figure 11 -- Use of Portal, Portlets, and the ESB

4.1.5.1.1.1. Use of Portlets

Portlets are to the Presentation Sub-Layer as services are to the Business Sub-Layer, as they provide the ability to develop reusable presentation services that can be used to display information on multiple screens. Portlets also allow information – both inputs and outputs, to be shared between multiple screen areas.

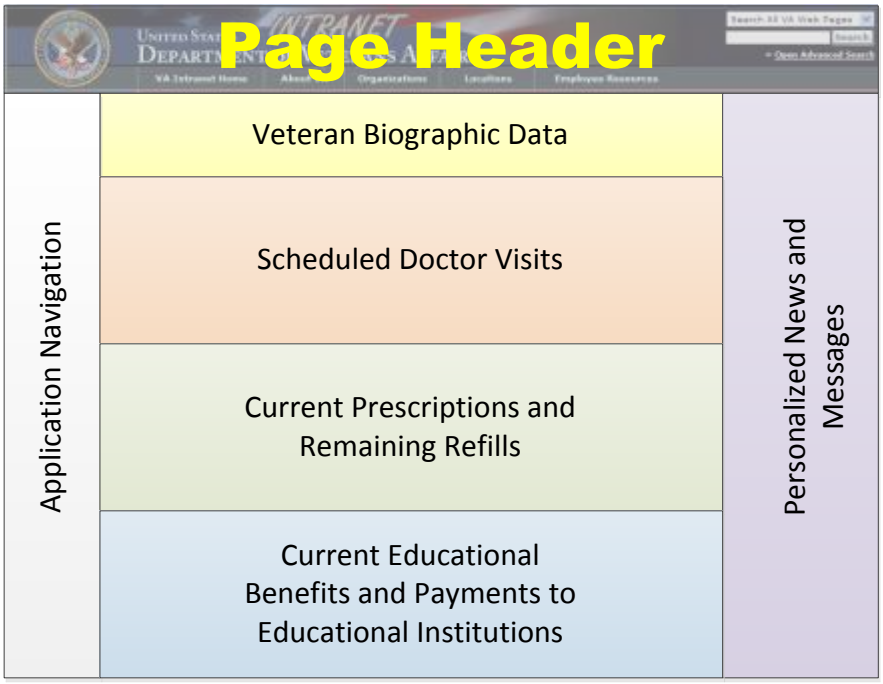

Figure 12 -- Example Use of Portals Displaying Information from Multiple Services

Figure 12 shows a sample web page with two standard areas (the Page header and the Application Navigation) and five areas specific to the Veteran, the:

- Veteran Biographic Data.
- Scheduled Doctor Visits.
- Current Prescriptions and Remaining Refills.
- Current Educational Benefits and Payments to Educational Institutions.
- Personalized News and Messages.

Information entered to identify the Veteran in the Veteran Biographic Data is used by all of the other Veteran specific areas. Using portals information is entered once and shared by all portlets on the page and by the individual portlets with the business logic services that support them. Further, the output of one portlet can be used as input to the other portlets on the page.

4.1.5.1.1.2. Portlet Standards

Web Services for Remote Portlets (WSRP) is an OASIS-approved network protocol standard designed for communications with remote portlets. The WSRP specification defines a web service interface for interacting with presentation-oriented web services. Initial work was produced through the joint efforts of the Web Services for Interactive Applications (WSIA) and Web Services for Remote Portlets (WSRP) OASIS Technical Committees.

The WSRP specification does not make any statements as to implementation. Java's portlet specification, JSR 168, and WSRP are not competing technologies. JSR 168 may be used to define a portlet, and WSRP may be used to define a portlet's operations to remote containers. JSR 168 portlets and WSRP may be used together to define a portlet and to provide remote operations. Similarly, .NET portlets may be created for use with WSRP. WSRP can be used to allow JSR 168 portlets to communicate with non-portal (i.e., ESB-based) Web services.

4.1.5.1.2. Physical Business Logic Services

Generally, there should be a one-to-one mapping between services in the Virtual Services Sub-Layer and the Physical Services Sub-Layer. There are no service facades with the services in the Physical Services Sub-Layer, as the services already provide a services interface and should be executable based on the capabilities that are provided at the lower levels.

The name Physical SOA Services Layer is used because it is the physical layer associated with the Virtual SOA Services Layer. The Physical SOA Services Layer represents the functional code (presentation logic, business logic and data logic) of the modernized applications without reference to the middleware or technology environment that is required to support the legacy applications.

4.1.5.2. Legacy Systems Sub-Layer

This EAA does not describe the legacy application environment and therefore, does not describe the contents of the Legacy Systems Sub-Layer, only to note that it exists and contains the non-SOA services legacy systems that provide the vast bulk of VA systems now and will through a transitional period. Since the EAA mandates that new development be based on SOA services, not extensions to the legacy systems, the EAA does not provide direction for this sub-layer. While there will be some ongoing development in the Legacy Systems Sub-layer, this should be confined to ongoing maintenance and minor enhancements to the legacy systems. Larger enhancements to the legacy systems should be based on SOA services.

4.2. Layer 2 – Enterprise Standard Messaging Layer

An SOA is an architecture in which business functionality is provided by a set of discrete, communicating "services" rather than by monolithic applications. The communication between services is accomplished through the exchange of messages. The nature and form of those messages is the subject of this section. Stemming the flow of ad hoc communications between systems and requiring each system to understand the syntax and semantics of every other system is the role of a formal messaging system. Semantic harmonization ensures that data exchanged between applications has the same meaning in both applications. Syntactic harmonization means that the data has the same meta data, such as allowable values and ranges and edit and validation rules etc., in both systems. The messaging system will be composed of the following elements:

- A messaging architecture that establish how the messaging system will be used by VA applications.
- A set of standard messages used across the enterprise whose data element names and data element semantics are defined by the Enterprise Logical Data Model (ELDM) being implemented as part of the Data Architecture. This use of a standard set of messages based on the ELDM data names and semantics provide the syntactic and semantic harmonization that is required to allow systems to work together.
- Reduce the number of formats and types that are used to transmit information between VA applications.
- Built on messaging infrastructure that ensures the authenticity and delivery of each message. The infrastructure used to ensure the authenticity and the delivery of messages is handled at lower levels of the hierarchy. The Messaging Layer is strictly a logical layer that talks to the form, format, and semantics of the message without regard to the physical structures at lower

levels that are used to ensure reliable delivery of the message. The messaging infrastructure used to ensure the reliable delivery is described in the lower levels of this EAA.

- A series of standards, policies and procedures that mandate use of the messaging system. The actual standards, policies and procedures are outside the scope of this EAA. This EAA only mandates that they exist and that standard formats are defined and used.

4.2.1. Virtual Enterprise Standard Messaging Sub-Layer

The Virtual Messaging Layer describes the messages that are passed between services in the VA SOA. These messages may be independent of the external formats that may be mandated for messages or the formats of the messages that are currently transferred between VA applications. In VA SOA, data transferred as part of a service is in a standard format using enterprise standard data definitions. These enterprise standard data definitions provide semantic harmonization and can be used natively by the applications or through translation performed by an adapter developed by the applications. The development of data standards is discussed in VA Directive XXXX "OIT Enterprise Data Management Principles" and the VA Enterprise Data Architecture.

4.2.1.1. Logical Messaging Hub

There are two approaches to services messaging. First, applications may perform point-to-point messaging in which each application can send information directly to any other application, but must know the format, syntax, and semantics of the messages traversing that particular link. If *n* is the number of applications, there could be as many as *n*(n-1)* individual connections, each with a separate format and distinct syntax and semantics. Second, and more common in an SOA, messages are mediated a messaging hub. This approach is highly scalable since there is no message semantic transformation in the hub – it is performed by the applications through an adapter so only standard messages are passed and each application can send a message to any other application knowing its own internal formats and the standard formats.

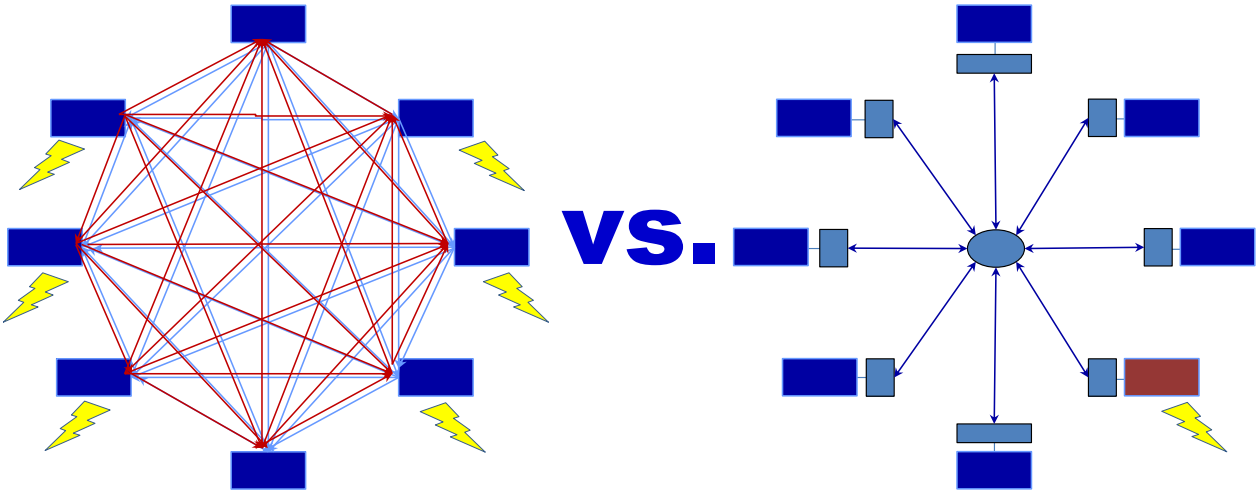

Figure 13 – Point-to-Point Messaging vs. Use of a Message Hub

Further complicating the situation, each application may have its own external interfaces which use yet another set of formats, syntax, and semantics. When a messaging hub is used there would be one or more nodes specifically designated as external interfaces that localize external interfaces, allow the elimination of redundant interfaces – where more than one VA system maintains an interface to a

single external system. This also localizes the conversion from VA internal formats to external message syntax and semantics.

While at the virtual level, there is no concern as to the physical transport for the messages and thus there is no concern as to whether there is a physical hub or point-to-point communication, the VA EAA Messaging Layer is based on the use of standard messages as if there were a logical hub independent as to whether an actual hub is used. The discussion of the use of adapters to convert the syntax and semantics of each of the application's native messaging formats to the standard is discussed in the Message Transformation Sub-Layer described below.

4.2.1.2. Use of the ELDM as the Basis for Messages Between Systems

The VA Messaging System shall be based on the use of standard messages. The messages will be semantically and syntactically harmonized based on the VA ELDM. The ELDM contains metadata on all data that is visible external to a VA application – whether it is input or output via a user interface or an interface to an internal or external system. The ELDM does not contain metadata for any data elements that are wholly internal to an application and which are unlikely to be exposed outside of the application. Therefore, the ELDM will describe all data that is to be exchanged external to a single application – which is the main purview of the EAA messaging layer.

VA internal messages will be based on the entities documented in the ELDM and with one message being used for each entity that includes all attributes for that entity. This reduces the number of messages that each application will need to support. Should there be significant demand for messages with less content then contained in the full entity, the entity should be split in the ELDM. One would expect that much of the data in the ELDM would correspond to data elements defined in the standard message formats that are currently used by VA systems (e.g., Health Level Seven International (HL7), the global standard for interoperability of health information technology or the Nation Information Exchange Model (NIEM), which is becoming increasingly used to exchange data between Federal agencies and with the Federal Government and state and local entities), and data definitions from those sources may, but not necessarily will, be incorporated into the VA Standard Message set.

4.2.2. Enterprise Standard Messaging Transformation Sub-Layer

The Messaging Transformation Sub-Layer provides a mapping between the syntactically and semantically harmonized, standard messages that flow between services and the physical message formats used by VA's legacy systems and systems external to VA.

All messages exchanged between VA services or between VA systems and external systems interfaces shall be based on the use of the standard messages. Where an application system's internal data representation does not conform to the standard data element definitions or message formats, the application will be required to develop adapters from the current internal data definitions and formats to the enterprise standard data definitions and formats. These message adapters will be integrated with the adapters used to expose legacy system functionality as services. For the most part, exposing the legacy systems as services will be through the messaging adapters and, for the most part, the messaging adapters will be the services adapter. Fortunately, only one such adapter need be developed per application system, rather than one for every other system with which the system communicates.

While the most desirable end state would be for all VA applications to use the standard data definitions as their internal data representations, this would require the redesign and modernization of

all VA systems, something that would not be financially or technically feasible. The next most desirable end state would be for all VA systems to implement adapters and for all messages between VA systems use the standard messages. It will also be infeasible to define *a priori* all messages that will be required. This is also infeasible from a financial and technical perspective. What is feasible is that as services are developed the:

- Standard syntactically and semantically harmonized data elements are defined.
- Standard messages are defined based on the syntactically and semantically harmonized data elements.
- Adapters are developed for the legacy systems with which the services need to communicate using the newly defined standard messages.

The approach is to adopt a "just-in-time" approach to the definition of standard messages and the development of the required adapters. That is, the effort is undertaken only when and where there will be a service being developed to consume the messages.

4.2.3. Physical Enterprise Standard Messaging Sub-Layer

The Physical Messaging describes the physical form, format, syntax, and semantics of the messages that pass between VA systems and between VA systems and external systems.

This layer deals with logical messages between services and deals with the format of the message that the service sees and not details of the physical transport of the message. This layer deals with logical formats such as HL7, not transports such as HTTP, Simple Object Access Protocol (SOAP), or IBM MQ. More physical aspects of the message transport are handled by the Message Oriented Middleware Sub-Layer of the Enterprise Software Layer (4.5.6.4 Message Oriented Middleware Sub-Layer and 4.8.3.3 Physical Networks). Because this layer deals with logical messages between services, the messages may never be sent over any communication channel, as the communicating services may be operating within the same system.

4.2.3.1. Communication with External Systems

The Physical Messaging Sub-Layer supports the physical message formats that in many instances will be very much different from the semantically harmonized, standard messages formats used for communication between VA SOA services. This would include the need for VA to communicate with external organizations using standard messages in formats specified and over interfaces based on previously agreed to Interface Control Documents (ICDs) and agreements as to the form and format of the messages that will traverse them.

It is assumed that VA will have little or no influence on external organizations with which it communicates, and that the external interfaces with those organizations will need to remain as they are. Therefore, interfaces with external systems will remain in whatever format they have been implemented. The Physical Messaging Sub-Layer will support these external interfaces.

The Physical Messaging Sub-Layer will also include all of the standard message formats that are used within VA to including, but not limited to:

- NIEM.
- HL7.
- EDIFACT.

4.2.3.2. Communication Between Legacy VA Systems

Legacy VA systems do not communicate using the syntactically and semantically harmonized, standard messages that pass between VA SOA services. Legacy systems communicate using the full range of mechanisms from shared databases and shared files to formal messages in standard formats such as HL7. While it is the goal of this EAA that all VA applications be based on the use of SOA services and standard, semantically harmonized messages, legacy systems will continue to use their current messaging for communication amongst themselves. Communications adapters, described in the transformation sub-layer above, will be used to transform the messages in these legacy system formats to the VA standard, semantically harmonized messages.

4.2.3.3. XML as a Wire Format

XML is the format de rigueur for communications with Web services and all Web services are based on the use of, or assume the use of, XML. As noted earlier, services in the sense of an SOA are not Web services, and thus there will be no requirement to use XML for transmission of information between services, particularly between services provided by closely related applications. Further, NIEM conformance does not require use of XML for the message transport, although it does require that the message format be defined in XML in an XSD.

XML is the preferred "wire format" for the messages sent between services where either the service provider or service consumer is outside of VA. XML formatted messages can be used within VA or even between related applications, but in those cases tradeoffs must be made between the additional flexibility that XML provides and the performance impacts of its use. The overhead required to convert, format, un-format and reformat and reconvert between binary data and variable length, character based XML messages make use of XML impractical for high volume, high performance applications.

Web services are services that expose services over the web and use a specific set of protocols including SOAP, WSDL, UDDI, SAML, and XML etc.; are designed to allow people and systems to consume services from providers unknown to them. The use of XML and higher-level protocols promote interoperability between consumers and providers. Since XML is self-describing, it allows communications and interoperability between consumer and provider without prior negotiation of data formats or data syntax. However, there is still a need for agreement on the semantics of the data being exchanged. This agreement on the semantics – the data element names and definitions – is provided by VA standard VA messages or be such standards such as HL7 or NIEM.

However, since SOA denotes an architectural philosophy rather than a set of standards, many of the benefits of an SOA can be obtained through higher performance transports and mechanisms than through "Web services". Further, many of the benefits, such as improved interoperability, that are only available through the full "Web services" protocol stack may be of minimal utility within an application or even within a component. For example, SAML, which supports security in services transactions between parties unknown to each other may be dramatic overkill for services who can be assured of the identity of its users by much simpler, less expensive means.

There is also significant concern that the heavy protocol stack used for "Web services" is less efficient in support of very high volume transaction processing environments. The trade-off between the use of Web services and native messaging is a trade-off between interoperability and generality and performance. Production organizations with very large numbers of service accesses coming from internal users, and stringent performance requirements – bordering on real time requirements, readily sacrifice some interoperability (which they may not require) for performance.

4.3. Layer 3 – Interface Layer

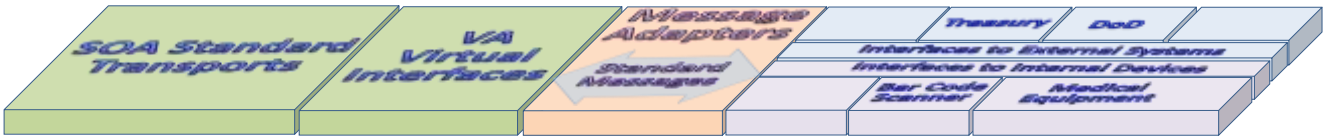

Figure 14 – Interface Layer

The Interface Layer describes interfaces at the protocol level – the Interface Control Document (ICD) and Interconnection Security Agreement (ISA) level. It is above the physical link level, but below the transport level. The interface layer describes interfaces between VA services / systems and external systems and between VA services and special devices – i.e., devices other than end user devices that provide a graphical user interface. These special devices could be anything from a barcode reader to an MRI scanner. The characteristic of both of these types of interfaces is that they are very much different from the standard VA SOA services messages formats, and they are beyond VA's power to change. The Interface Layer services present the VA SOA services with a standard message interface regardless of whether the interface is with an external system or a special device. For external systems this transformation is at the transport level – not the message syntax and semantic level, which occurs at the Standard Message Level. For devices, the Interface Layer converts the device data formats to VA compatible information streams. The Interface Layer is used to convert the native formats of both of these types of interface to message streams that VA services can handle.

Many devices may come with special purpose driver to application software to control this device. If the device vendor provides a full application to support the device then a SOA service facade would be placed in front of the application to allow it to talk to VA SOA services.

One goal of this SOA is that there will only be one interface between VA and any external system and for all VA systems or services to access the external system through that single external interface.

4.3.1. Virtual Interface Sub-Layer

The Virtual Interface Sub-Layer provides a service based interface using VA standard messages making the interfaces appear to be standard services. This applies to both interfaces to external applications. The interfaces appear to be standard SOA services.

4.3.2. Interface Transformation Sub-Layer

The Interface Transformation Sub-Layer is basically a service facade that is used to transform the physical characteristics of the system or device interfaces to the SOA services that will be visible to the SOA services and applications.

4.3.3. Physical Interface Sub-Layer

The Physical Interface Sub-Layer is the physical interface to the system or the device and takes whatever form the external system or device provides.

4.3.3.1. System Interfaces

The external systems interfaces are typically fixed and specified by an ICD with the external party to the interface. For the purposes of this EAA it is assumed that these interfaces are fixed and cannot be

unilaterally changed by VA. These interfaces may use transports used within VA, or may use other transports – transports not used within VA. These transports will continue to be used, but will not be visible to services or applications within VA.

4.3.3.2. Device Interfaces

Vendors of devices to which VA systems might need to attach typically provide either device drivers to allow systems to both control their devices and to receive data from the devices. In addition to the device drivers that vendors provide, they may also provide full applications to control or allow use or access to their devices. These applications or the device may be accessed through a vendor provided API. It is these device drivers, applications, or APIs are accessed to control the device and to receive information from the device or to send information to the device.

4.4. Layer 4 – COTS Software Sub-Layer

The COTS Software includes two classes of non-systems oriented COTS software. First, there is functional COTS software – COTS software that directly supports the VA mission e.g., patient management systems, drug interaction tracking etc., and second there are more general enterprise COTS software that can be used in multiple functional areas e.g., workflow management, collaboration software, document management etc.

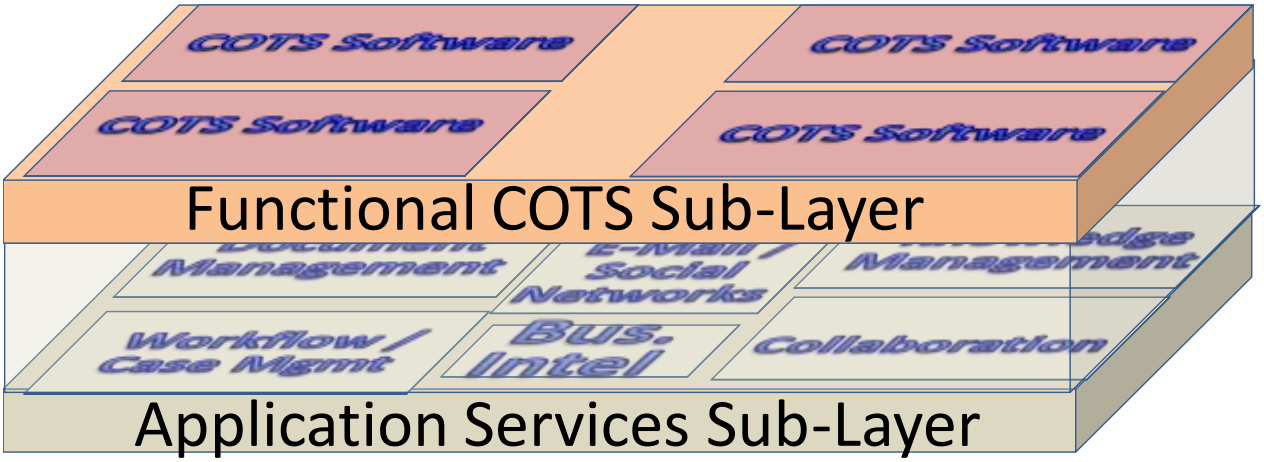

Figure 15 – COTS Software Layer

4.4.1. Virtual COTS Software Layer

The Virtual COTS Software Layer expose the COTS software to the modernized applications and VA users.

4.4.1.1. Functional COTS Software Sub-Layer

The non-SOA service based functional COTS software will reside in the Physical Enterprise Software Sub-Layer with minimal modifications. However, few, if any, functional COTS packages are installed as standalone packages with no interaction with other VA applications or services. Interfaces to functional COTS packages will be exposed as SOA services rather than as custom APIs and will be accessed through the use of standard VA, semantically harmonized, messages. This will require adapters since none of the COTS packages will natively use VA Standard Messages.

VA currently uses a wide variety of functional COTS Software. Functional COTS software, as distinguished from COTS systems software, directly supports the performance of the VA mission rather than the operation of VA IT Systems. The ESB and MOM software discussed elsewhere in this section are examples of VA system software.

It is not the purpose of this EAA to identify the functional software packages that are currently in use in VA or that may be needed in the future, but only to establish the place for this class of software in the architecture and the manner in which it will integrate with or interoperate with other components of this architecture. Any functional COTS package will need to be included in the TRM prior to its use as part of a VA application system.

4.4.1.2. Application Services Sub-Layer

Figure 16 – Application Services

Application Services includes a number of capabilities that are normally based on large complex COTS packages including:

- **Knowledge Management** – Comprises a range of strategies and practices used in an organization to identify, create, represent, distribute and enable adoption of insights and experiences. Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organizations as processes or practices.
- **Document Management** – Is a computer system (or set of computer programs) used to track and store electronic documents and / or images of paper documents. It is usually also capable of keeping track of the different versions created by different users (history tracking). The term has some overlap with the concepts of content management systems. It is often viewed as a component of enterprise content management systems and related to digital asset management, document imaging, workflow systems and records management systems.
- **Collaboration Software** – Is computer software designed to help people involved in a common task achieve goals. The design intent of collaborative software is to transform the way documents and rich media are shared to enable more effective team collaboration. Collaborative software helps facilitate action-oriented teams working together over geographic distances by providing tools that aid communication, collaboration and the process of problem solving. Additionally, collaborative software may support project management functions, such as task assignments, time-managing deadlines, and shared calendars.
- **Workflow / Case Management** – A workflow management system is a computer system that manages and defines a series of tasks within an organization to produce a final outcome or outcomes. Workflow Management Systems defines different workflows for different types of jobs or processes. At each stage in the workflow, one individual or group is responsible for a specific task. Once the task is complete, the workflow software ensures that the individuals

responsible for the next task are notified and receive the data they need to execute their stage of the process.

- **E-Mail / Social Networks** – Encompass much more than just the transmission of text based e-mail messages, but include a wide variety of message oriented software that is used to exchange information between individuals or between groups of people.
- **Business Intelligence (BI)** – Mainly refers to software used in identifying, extracting and analyzing business data, including current and predictive views of business operations. Common functions of business intelligence technologies are reporting, online analytical processing, analytics, data mining, process mining, complex event processing, business performance management, benchmarking, text mining and predictive analytics. Business intelligence aims to support better business decision-making. Business Intelligence is dependent on a data warehouse or data marts for its data, BI software normally includes significant data analysis and visualization capabilities.
- **Complex Event Processing (CEP)** – Refers to a set of software products that manage the actions of a set of event creators and a set of event consumers. The creator, which is the source of the event, only knows that the event has occurred. Consumers are entities that need to know the event has occurred; they may be involved in processing the event or they may simply be affected by the event. The CEP software correlates events and manages the actions that the consumer takes in response to the events. Event consumers typically subscribe to some type of middleware event manager. When the manager receives notification of an event from a creator, it forwards that event to all registered consumers. The benefit of event-driven software is that it enables large numbers of creators and consumers to exchange status and response information in near real-time. CEP interacts very well with several aspects of the SOA including BPM, BAM, and service orchestration.

One of the common aspects of software at this sub-layer is that all of the software becomes more valuable the more widely that they are used in an organization. Ideally, there would be one instance of each class of software for use across VA, but it is more likely that individual packages will be used in broad areas of VA, but not throughout all of VA.

As noted above, the capabilities in this area are provided by large, complex COTS packages. Systems interfaces to these packages would be exposed as services like any other COTS packages, but user access to the capabilities would be provided through their native interfaces.

4.4.2. COTS Software Transformation Sub-Layer

4.4.2.1. Functional COTS Software Transformation Sub-Layer

Functional COTS software is to be exposed to VA SOA services as SOA services using standard VA messages. While more and more COTS vendors are providing service interfaces, not all are and one cannot assume that such a service interfaces will be available from the COTS package vendor. Further, even where the COTS vendors provides an SOA interface, the messages accepted by those SOA services are unlikely to be syntactically and semantically harmonized with VA standard messages.

The transformations performed at the Functional COTS Software Sub-Layer will be to:

1) Provide an SOA facade to the COTS vendor provided APIs.

2) Define message formats semantically and syntactically harmonized with VA standard messages.
3) Provide syntactic, semantic, and transport harmonization with VA standard messages and transports.

The transformation will include the development of adapters to be placed between the functional COTS package and the VA SOA services. The adapters will interface with the functional COTS application through the COTS vendors' standard APIs provided with the functional COTS package without change to the internal code or internal structure of the functional COTS application. Changes to the functional COTS application should be minimized to the extent feasible.

4.4.2.2. Applications Services

The Transformation Sub-layer for the Application Services will be a service facade to provide a services interface. Many of the applications included in this sub-layer present a services interface and those that do not provide a more traditional APIs. Those APIs can be exposed through a service facade.

4.4.3. COTS Software Physical Sub-Layer

4.4.3.1. Functional COTS Software Sub-Layer

The Physical Enterprise Software Environment – Functional COTS Sub-Layer includes the COTS software package and whatever custom code that has been added at the APIs to allow the package to most fully meet VA requirements and to integrate with existing VA systems. While it would be highly desirable for the all new functional COTS packages to integrate with other VA applications and software packages through a services interface, it is possible that some of the interfaces will be required to be through the APIs provided with the COTS package. While this is allowable when required, such interfaces should only be used when a service interface would not meet user requirements, or for those cases in which such services are not technically feasible.

All customization of the functional COTS software should be performed by code interacting with the product through vendor supplied APIs rather than through modifications to the software package.

4.4.3.2. Application Services Sub-Layer

The Applications Services Physical Sub-Layer consists of a set of large, very complex COTS software packages that will be installed and be made available within VA.

4.5. Layer 5 – Enterprise Software Environment

The Enterprise Software Layer is the layer in which the major software products that provide the operational environment needed to support the services and logical messaging described in Layers 1 and 2. It is the software at this level that enables the software running at the higher levels.

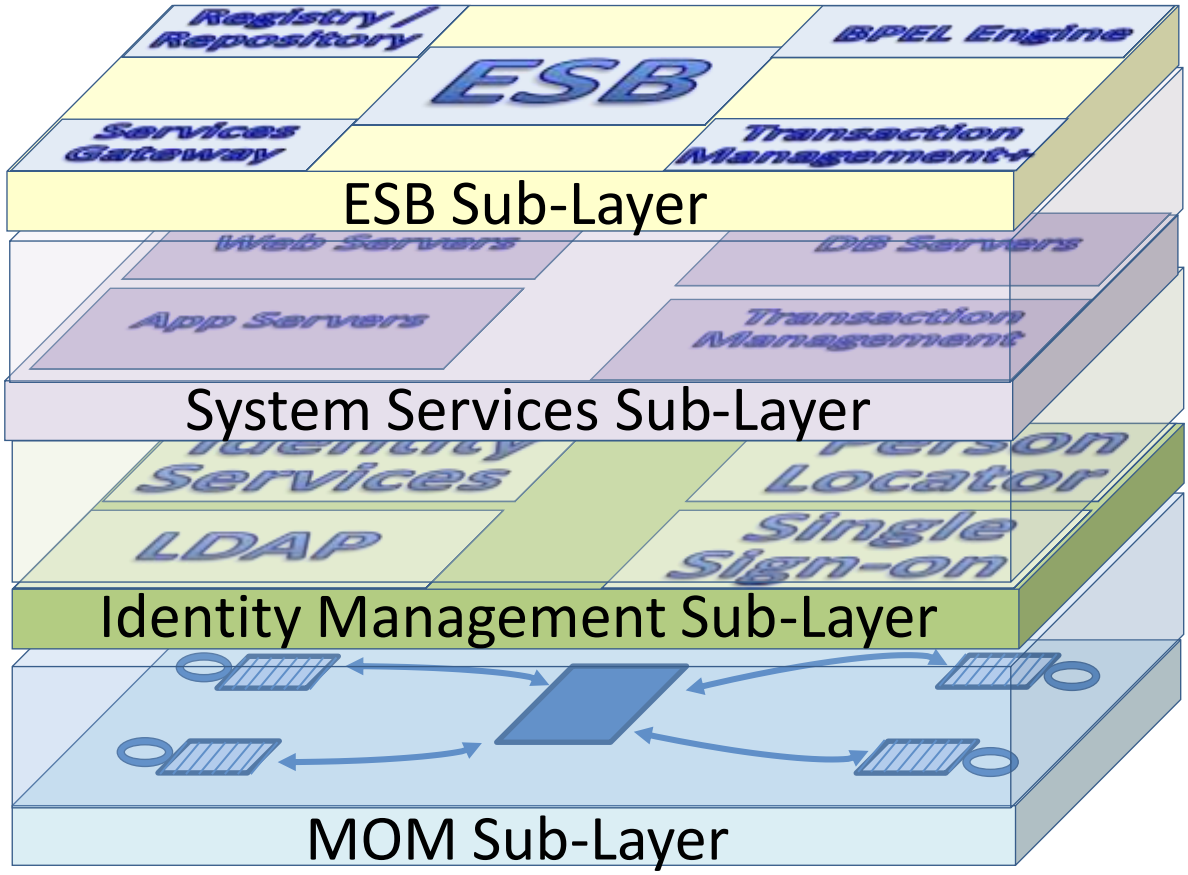

Figure 17 -- Enterprise Software Layer

The Enterprise Software Environment consists of four sub-layers, the:

- Enterprise Service Bus (ESB) Sub-Layer.
- System Services Sub-Layer.
- Identity Management Sub-Layer.
- Message Oriented Middleware (MOM) Sub-Layer.

4.5.1. Core Common Infrastructure Services

Just as the CCBS were business functions that would be developed once and used throughout VA, the CCIS are infrastructure services that are to be used throughout VA. Most of the CCIS reside in this layer of the EAA layered Model. Initial candidates for the CCIS include:

- Enterprise Single Sign-On.
- Enterprise Identity Management.
- Enterprise Identity Resolution.
- Enterprise Messaging.

Unlike the CCBS whose value is merely directly proportional to the number of applications that make use of them, the value of the CCIS rises much more rapidly. It has long been noted that value of

many network-based systems rises with the square of the number of elements in the network. Single sign-on provides very limited value if only one or two systems use the single sign-on capability, but, has much greater utility if all systems use the capability. Then users need only enter a single USERID and PASSWORD to log onto any of the systems to which they are allowed access.

This version of the EAA does not mandate the use of these services at this time since the CCIS have not yet been built and thus are not available for use, nor are there definitive dates for their completion. However, as the CCIS are implemented and deployed their use will become mandatory for new system deployments and major enhancements to existing systems.

4.5.2. Virtual Enterprise Software Environment Sub-Layer

The Virtual Enterprise Software Environment Sub-Layer describes the interface to the software environment that will be seen by the application services. The services made available in the Virtual Enterprise Software Environment are independent of the specific enterprise software that is installed as part of the Physical Enterprise Software Environment Sub-Layer. For example, while a specific set of products will be installed to support the enterprise, the integrated Electronic Health Record (iEHR), Vista Modernization, and VA Benefits may each choose different products as the basis for their ESBs or MOM. Regardless of the specific COTS product(s) that are selected the service interface provided to the users should be the same. Maintaining a common set of virtual services may require that some very specialized, unique features of specific products not be used.

4.5.3. Enterprise Service Bus Sub-Layer

The Enterprise Service Bus Sub-Layer will define a standard set of ESB Services that are to be provided regardless of which vendor product(s) are used as the basis for any of the ESBs. The characteristics of the ESB, particularly as they relate to capabilities that it must provide to support the SOA are described in detail in The VA Service Oriented Architecture (SOA): Technical Framework. While many of the products that may be selected as the basis for one or more of the ESBs may provide capabilities beyond those in the standard set, the standard set is defined to ensure that a specified set of capabilities are available to SOA services regardless of which ESB or ESB vendor products are used. This not only allows COTS products to be changed over the life of the system but, allows SOA services developed to operate in one environment to operate in other VA environments. This, in turn, allows VA Core Common Services to be replicated and to operate anywhere in VA rather than requiring data to be sent to a remote location to allow use of a Core Common Service.

Figure 18 – ESB Capabilities

The ESB is a logical entity that provides a number of capabilities necessary for the implementation of the SOA. The ESB will need to support services being provided in each layer of the multi-tier SOA Layer (Layer 1) of this architecture and may be distributed across each of the sub-layers of that architecture. In previous types of architectures, the software in the ESB's place in the architecture was the magical glue that performed all of the integration (e.g., data integration) functions between

systems. In this SOA, the VA ESB is federated with the possibility of ESBs existing at multiple levels including the project, MI or business area, and enterprise. This allows the separation of services and the ability for projects, MIs, or business areas to support local services. Also, orchestration mechanisms may exist at multiple locations in the architecture. Because some services may only be exposed within an application, or a closely related group of applications, orchestration may need to occur within that application or application group.

In addition to orchestration and service execution, the ESB will provide other capabilities including authentication and authorization, messaging and format transformations, and registry and repository functions etc. As a logical construct, the ESB potentially includes capabilities of multiple hardware and software vendors possibly with components distributed across the enterprise. ESBs may be instantiated within applications that are implemented based on services.

4.5.3.1. Information Integration

The integration of information flows between systems is accomplished through architectural means rather than through a complex piece of software. Unlike previous integration approaches that were product-based – the integration in this SOA is through the architecture, not through products.

One major reason that previous architectures were unable to provide the level of integration desired was that they depended upon a piece of software to accomplish complex data and systems integration and the complexity of this software rose with the square of the number of systems to be integrated. SOAs are not conceptually new; they have been implemented for over thirty years. What is new is that there are now industry accepted standards for communications between application systems and an understanding that data integration must be done in the applications, not in an external piece of software.

Figure 19 – Use Message Adapters to Enable VA Standard Messages

Ideally, in the SOA much of the data integration work has been removed because communication between services is based on data definitions contained in data classes from the ELDM. The core of data integration comes from the process or the syntactic and semantic harmonization of the data as part of the definition of the VA Standard Messages. While this removes the data integration from the both the SOA and the EAA, it does leave it in the Data Architecture. The level of effort required to achieve the require syntactic and semantic harmonization of the data cannot

For the foreseeable future many of the data exchanges will not be based on native use of the ELDM data classes. Instead, there will need to be a mechanism that will perform data translation and transformation. This data transformation cannot be done on a point-to-point basis. It must be done from the application's native data definitions to the standard data definitions and from the standard data definitions to the data definitions required by other applications. Using this approach each application need only transform and / or convert data once, and this transformation can be used by any application that has an adapter to allow it to talk using the standard definitions. As noted earlier, these semantic data transformations are not part of the ESB, but will be the responsibility of the applications. Each application will need to convert from its native data definitions to and from the enterprise native definitions as specified in the ELDM. The applications will perform this

transformation through a series of adapters, one for each application or one for each group of applications using common messaging syntax and semantics.

The ESB will need to be able to support a hierarchy of services, each layer of services being built upon the underlying layers. These layers will range from very fine-grained functional services to very high-level functional services. High-level functional services will be composed of lower level services and the ESB will need to efficiently support the large number of subsidiary service calls that result from the invocation of the single high-level service. Security considerations, such as identity management, authentication, and authorization will influence the permitted granularity of the services.

4.5.3.2. ESB Functions

There are a number of possible definitions as to the capabilities that the ESB will provide from not much more than a simple BPEL engine to a complex set of capabilities including a variety of communication capabilities and other transformation capabilities. As used in this EAA, those communications functions and transformations have been allocated to other layers and sub-layers in the architecture.

The ESB in this EAA contains virtually all of the capabilities directly related to the services including such service-related capabilities as the service registry / repository. The ESB also provides process and transaction management provides a higher layer of services dealing with how business processes are executed. Process and transaction management capabilities include:

- **Orchestration –** Refers to composing services into an overall process, for example, a higher-level service, which is internally made up of a series of lower-level (finer-grained) processes sequenced together.
- **Choreography –** Refers to ways of defining how multiple applications collaborate to accomplish an overall business objective by exchanging messages in a peer-to-peer style. There is no particular central point of control. Rather, execution of an overall business flow is shared and distributed across all the applications participating in the collaboration.
- **BPEL Engine / Services Processing Engine** – Business Process Execution Language (BPEL) is a formal mechanism that can be used to specify business process services and products exist that can generate BPEL from graphical specifications of service flows written in Business Processing Modeling Notation (BPMN) and that can directly execute the BPEL. The VA EAA and SOA does not require that services be specified in BPMN or BPEL, but can be written in other languages such as Java or C#, and thus, a Java-based or C# application server can serve as the Services processing engine.

- **Transaction Management –** In the context of an SOA, a transaction can be a long running series of services. "Two-phase commit" transactions are sometimes referred to as Atomic, Consistent, Isolated, and Durable (ACID) transactions. Because an SOA

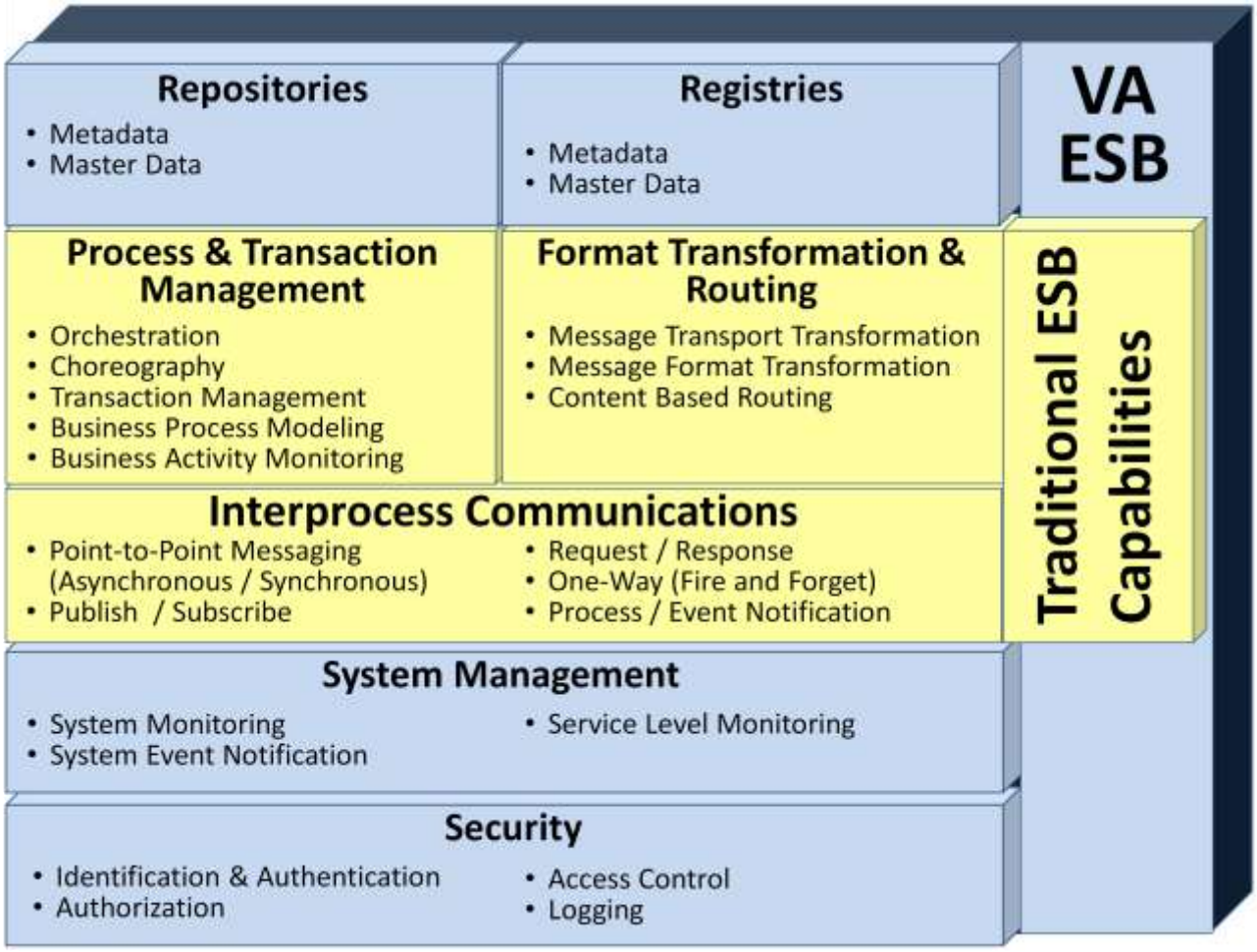

Figure 20 – VA ESB Functions

"transaction" may run for minutes, hours, days, or even longer, locking databases and implementing a two phase commit for database updates may not be feasible. In an SOA backing out a transaction is normally accomplished through the execution of one or more compensating transactions to put the databases in the proper state.

- **Registry / Repository** – Services descriptions will be maintained in both human and machine-readable formats. All services (other than those only accessible within an application) will be held in a central repository or repositories (e.g., one for machine-readable and one for human-readable). A "registry" is a place where official records are kept. It is an authoritative store of information that relates to a particular task at hand. Registries store metadata that relate to a particular asset without actually containing those assets. The store that actually contains those assets is the repository. So, while a registry simply records official information that relates to an asset, the repository stores the assets themselves. While a registry might be the right place to store a Service definition and

associated policy metadata, the repository is the right place to store service artifacts such as models, maps, shared keys, and transformational schemas.

- **Services Gateway** – Each ESB will define a security domain requiring authentication and authorization upon entry to the domain. Further, authentication is not required when accessing services in the security domain, although each service may, but is not required to perform, individual service authorizations. Every request entering the domain will have been authenticated and provided authorization to enter the domain, but not necessarily authorization to execute all services in the domain. The Services Gateway is a gateway that provides access to a services security domain. It is at the Services Gateway that the authentication and domain authorization is performed.

4.5.3.3. Services Security Domains

NIST Special Publication (SP) 800-33 defines a security domain as "a set of active entities (persons, processes, or devices), their data objects, and a common security policy." More concretely, a security domain is a set of networked hosts protected by one or more security perimeter devices from unconstrained access by other network components. Security domains, with their accompanying perimeters, provide the basis for placement, and configuration of firewalls, Virtual Private Networks (VPNs), and remote access protection devices.

The security gateway is the component of the infrastructure that authenticates the service requests. The security gateways provide authentication by verifying the digital signature of the incoming service requests. The security gateway apply security policies and serve as an application (level 7) firewall that can block service requests that are not well formed or from users that are not allowed access to the security domain. The security gateway can also be used to inspect the contents of the service requests and, for XML based services, can verify that the service request is well formed and conforms to the XSD for the service.

An ESB can provides two sets of security gateways; protecting the "front door" – the links from external sources and the "back door" – the links other VA ESBs. The security gateways on the verify signatures for services seeking to enter the ESB – either from outside sources or service requests coming from VA and DoD applications, and signs service requests and responses that it sends – responses to either other VA or DoD applications or external service requests . In addition, the ESB logs all authentication events. The security gateways are also able to perform both event and full payload logging. However, payload logging will only be performed when specifically requested. Such requests should be made as sparingly as possible both because of the very large amount of data that can be generated as well as the security and privacy concerns related to the generation of such large amounts of potentially personally identifying data.

4.5.4. System Services Sub-Layer

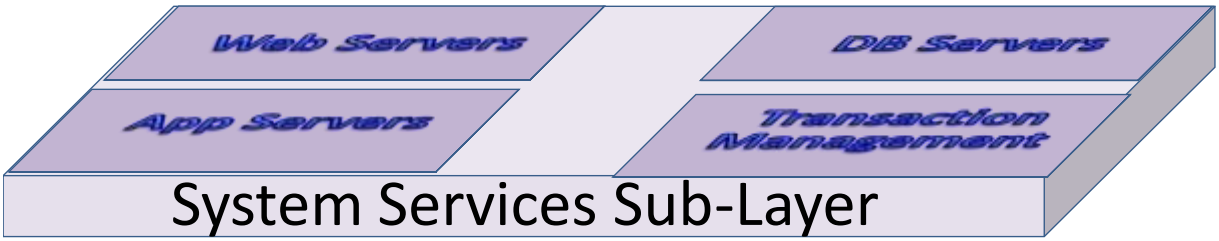

Figure 21 -- System Software Services

The System Services Sub-Layer exists primarily in the Physical Sub-Layer as it provides the core systems services that allow the presentation, business, and data services to operate and includes software such as:

- Web Servers (e.g., Apache or IIS).
- App Servers (e.g., WebSphere App Server or JBoss App Server).
- Database Server.
- Transaction Managers (e.g., Tuxedo or CICS).

Transaction processing allows multiple individual operations to be linked together automatically as a single, indivisible transaction. The transaction-processing system ensures that all required data records are locked and that either all operations in a transaction are completed without error, or none of them are. If some of the operations are completed but errors occur when the others are attempted, the transaction-processing system "rolls back" *all* of the operations of the transaction (including the successful ones), thereby erasing all traces of the transaction and restoring the system to the consistent, known state that it was in before processing of the transaction began[10]. If all operations of a transaction are completed successfully, the transaction is committed by the system, and all changes to the database are made permanent; the transaction cannot be rolled back once this is done. While historically transaction managers were standalone products, but now is more commonly bundled in with other products – commonly the MOM products described below.

Because of the low level "system" functions provided by software in this sub-layer, there really is not virtual layer and the physical software is visible to the application designer or developer, but not to the user. While a Virtual Sub-Layer could be defined with a Transformation Sub-Layer to the Physical Sub-Layer, this abstraction would not add much value as the Virtual Sub-Layer cannot be service based as the servers at this layer support the services. While it would be ideal to be able to develop a virtual layer that would enable the identity of the Application Server or the Web Server, but this layer would provide little advantage, additional overhead, and is not likely to be successful.

4.5.4.1. Identity Management Services

Identity Management Services are a mix of services, some visible to the users, and others that should be transparent to the user. For example, with single sign-on it is what the users do not see that is important – they do not see separate logons for each application that they attempt to use. The operation of single sign-on is strictly behind the scenes and operates transparently to the application and the user. The core of the Identity Management services is a directory or repository that contains identity information and a secure mechanism for applications to query that repository and securely obtain the results. The identity management repository in normally developed based on the Lightweight Directory Access Protocol (LDAP) which is accessible to programs and applications.

[10] Transaction Management Systems are less important in an SOA environment than in legacy transaction processing systems since the orchestration engines assume some of the responsibilities performed by the transaction monitors. Further, because (as described in the VA Service Oriented Architecture (SOA): Technical Framework) orchestrated services may be very long running and thus, makes locking databases for the duration of the service may not be feasible, meaning the compensating services rather than rollback may be the approach to be taken in the event a service fails.

The LDAP protocol and its associated message set will be the standard method for applications to access directory information. In this case, the virtual layer and the physical layer will be the same and will be the LDAP message set.

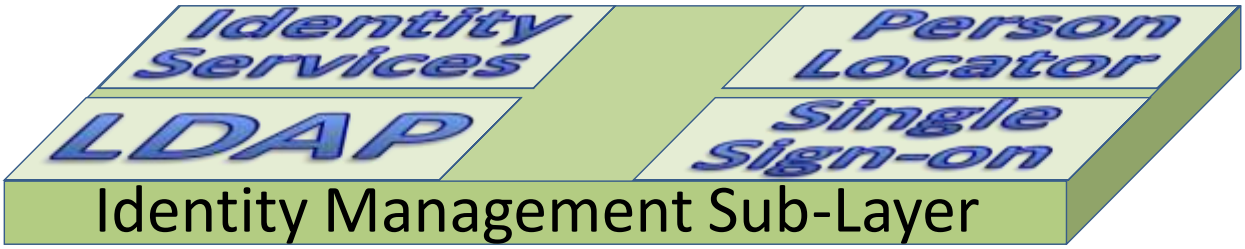

Figure 22 – **Identity Management Services**

In addition, to the basic directory management and single sign-on capabilities there are a number of other identity management capabilities that can be provided such as capabilities to:

- Locate individuals.
- Disambiguate individual information based on a less than complete description of the person.
- Locate all record associated with a person.

A centralized access management service will support the following three authorization mechanisms:

(1) Attribute Based Access Control (ABAC) – is a fine-grained authorization, using a collection of valid user attributes retrieved from multiple data sources or a meta-directory to dynamically validate if an entity (e.g. human user) should be authorized to access protected content. The process of validation is dynamic in the sense that it cross references Values of these user attributes against the ABAC policies each time a new authorization session is initiated. If validation passes, then a user is granted access to the resource accordingly, otherwise, access is denied.

(2) Role Based Access Control (RBAC) – is an authorization model that maps a set of permissions directly to a role. Users are directly linked to these roles instead of permissions. This method reduces the administrative burden of individually managing user privileges. A person would have a single identity, but might at various times have multiple roles. For example, a doctor at a VAMC who is also a Veteran might sign on to a system as a doctor to enter prescription information for a patient and later sign on to that same system as a "patient" to renew a subscription of his own. The doctor would be recognized as the same individual (the same identity) but with different roles.

(3) Identity Based Access Control (IBAC) – is a traditional authorization model that assigns permissions directly to users. An example of this is adding a user to a document's access control list (ACL) and granting that user "read" access. This method is useful for controlling access to objects when the rules are relatively straightforward and align well with simple policies and smaller user-bases.

VA is in the process of identifying an enterprise-wide enterprise identity management solution that would be deployed for use across the VA enterprise. Until such time as the enterprise solution is

deployed the services described in this sub-layer of the architecture will not be available for use by new services or applications.

Although a specific enterprise-wide identity management service has not yet been designed, an Identity Management Segment has been identified and VA is in the process of developing the OneVA Identity Services Segment Architecture. OneVA Identity Services Segment Architecture[11] will adhere to seven core tenets:

- **Identity-centric** – Identity Services will necessarily reflect an architectural approach which is identity-centric – this will provide the most consistency and flexibility. Identity Services must be generic and all-encompassing – they cannot reflect a single business process or application
- **Roles as first class identity assets** - party roles must be modeled at the intersection of identities, entitlements, and organizational structures and managed as part of the broader identity management lifecycle
- **Role-specific authentication** – authentication mechanisms must reflect the levels of risk and the granularity of the resources associated with that risk, without overburdening the individual, and should apply to both parties in an interaction
- **Integrated identity data silos** –identity data integration approaches that combine the benefits of meta-directory and virtual-directory technologies, allied with tooling to assist with data reconciliation
- **Federated** – as a transition tactic, when integration is not yet possible and feasible, a federated approach may be used for mediation of relationships at the heart of identity management, which in turn depends on managing and brokering the trust that underpins those relationships
- **Shared identity services** – identity management capabilities must be delivered as distributed infrastructure services, which are defined according to clear contracts which are enforced through policies
- **Policy-based and service-oriented** – there is a need to authorize access to business functions and information at the level of each service using policy-based approaches to the definition and enforcement of access control requirements

The Identity Services segment architecture has a number of characteristics which are essential if it is to provide an identity management foundation for the long term and which is capable of supporting the broad array of business requirements in an incremental fashion. These include:

- It is based on a clear separation of identity management concerns, with identity management capabilities delivered as a set of distributed infrastructure services, underpinned by a services repository and ultimately an integrated identity data repository.

[11] OneVA Identity Services Segment Architecture Handbook

- Resources access these services through policy-based mediation, which also serves to control the monitoring and audit functions required to mitigate risk, and enforce compliance, and demonstrate auditable logs.
- Identity data is managed throughout its lifecycle, from core data maintenance through to provisioning and de-provisioning, by a set of processes implemented using automated workflow and process management technologies, to increase efficiency, enforce consistency, and facilitate integration of identity management and business processes.
- Open standard protocols and data formats bridge the gaps between the layers to facilitate interoperability between the architectural components and the broader IT infrastructure.

Users of VA systems may obtain PKI-based credentials from VA Authentication Federation Infrastructure (VAAFI) certificate providers. The VAAFI Credential Providers are government or non-government entities that supply login IDs and passwords or PKI certificate credentials to the end-users. As of the date of this document VA has two certificate providers:

- DoD Self-Service Access Center (DS Access) – a credential provider which issues the DS Logon credential.
- Operational Research Consultants (ORC) – a civilian company partnered with the VA.

4.5.4.2. Message Oriented Middleware Sub-Layer

The MOM services will be based on the transport of VA standard, syntactically and semantically harmonized messages suing a standard set of messaging services. These messages were defined at Layer 2 of the EAA – the Logical Message Layer. Transformation of non-standard messages will be performed in the Transformation Sub-Layer. The services to be provided include:

- Point-to-Point Messaging.
- One Way (Fire and Forget with Guaranteed Delivery).
- Publish / Subscribe.
- Request / Response.
- Process – Event Notification.
- Content-Based Routing[12].
- Message Transport / Format Transformation.

Message-oriented middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocols. The middleware creates a distributed

[12] Content based routing is dependent upon the messaging system to inspect the contents of the message and take action accordingly. The Government Accounting Office (GAO) is requiring agencies to encrypt messages within an agency security domain and even across the datacenter floor. This will of course make content based routing unavailable.

communications layer that insulates the application developer from the details of the various operating system and network interfaces. APIs that extend across diverse platforms and networks are typically provided by MOM.

MOM provides software elements that reside in all communicating components of an architecture and typically support asynchronous calls between applications. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism. MOM can either be a COTS product, for example:

- IBM MQ.
- Oracle AQ.
- Oracle (BEA) Tuxedo.

or locally developed software such as MDWS within VA.

Virtually all ESB implementations include not just the ESB service management software, but also incorporate some MOM capabilities to the extent that most major ESB vendors also include, or are based on or require the use of, proprietary messaging software – although they may provide access to or interoperate with, a variety of other vendor MOM products.

The most basic capability of the ESB will be to transfer messages between services. This can either be based on synchronous or asynchronous communication approaches. Messages can either be guaranteed delivery or not. Guaranteed delivery assumes that messages are persistent and will eventually be delivered even if the receiving system is not operational at the time that the message is sent.

The sub-layer will provide the core message transmission service for the VA EAA in contrast to the Layer 2 – Logical Messaging Layer. The Layer 2 – Messaging Layer includes logical messages and specifically deals with the messaging protocols that are used (e.g., HL7 or NIEM). This sub-layer contains the core messaging capabilities (e.g., Guaranteed Delivery) on which the Layer 2 – Messaging Layer depend. The MOM messaging systems provided at this sub-layer provide, at a minimum, the following capabilities.

- **Point-to-Point Messaging** – Ensures that only one receiver consumes any given message. If the channel has multiple receivers, only one of them can successfully consume a particular message. If multiple receivers try to consume a single message, the channel ensures that only one of them succeeds, so the receivers do not have to coordinate with each other. The channel can still have multiple receivers of a single type to consume multiple messages concurrently, but only a single receiver consumes any one message.
- **One Way (Fire and Forget with Guaranteed Delivery)** – A message is sent from a single sender to a single or multiple receivers. This Interprocess Communication method is built upon the point-to-point communications described above. However, at each step in the transfer the message is copied to persistent storage and the message is not considered to be successfully delivered until processed (e.g., moved to local storage) by the receiver.
- **Publish / Subscribe** – Allows multiple receivers to register (subscribe) to receive notification of some event or to receive updates to some data store. It has one input channel that splits into multiple output channels, one for each subscriber. When an event is published into the channel, the *Publish-Subscribe Channel* delivers a copy of the message to each of the output

channels. Each output channel has only one subscriber, which is only allowed to consume a message once. In this way, each subscriber only gets the message once and consumed copies disappear from their channels.

- **Request / Response** – Is composed of a pair of one-way messages. In this case the original sender sends a message to the original receiver. The receiver performs some work and then creates a second message, which it sends to the original sender. Request / reply consist of two independent messages and is not a simple "ack" of the original message.
- **Process – Event Notification** – Allows a request to be registered to be informed when some process completes or fails or when some specified event occurs. This allows for asynchronous processing as it allows a process to be notified when previously initiated processing is complete.
- **Content-Based Routing** – The Content-Based Router examines the message content and routes the message onto a different channel based on data contained in the message. The routing can be based on a number of criteria such as existence of fields, specific field values, etc. When implementing a Content-Based Router, special caution should be taken to make the routing function easy to maintain, as the router can become a point of frequent maintenance. In more sophisticated integration scenarios, the Content-Based Router can take on the form of a configurable rules engine that computes the destination channel based on a set of configurable rules. Content-based routing can cause additional security and privacy issues as payload data is processed and the results of that processing are be logged. Therefore, both security and privacy reviews will need to consider any content-based routing rules as well as the log data that is produced.
- **Message Transport / Format Transformation –** Provides communication services that support different message transports (e.g., HTTP vs. COTS messaging) or formats (e.g., SOAP HTTP vs. J2EE RMI) and converts between them as required. This message transport / format transformation is a syntactic transformation in that it transforms the physical form of the message and is not a semantic transformation which is performed in the communication adapters associated with specific applications or groups of applications.

Semantic data transformation is not performed by the ESB or the MOM, but is performed by a data transformation adapter that is owned by the application. The message transport / format transformation capabilities assume that the messages passing use the standard data definitions specified based on the ELDM.

By making the transformation of local data formats to the ELDM standard ensures that each application only needs to understand two data formats, its own internal data format and the enterprise standard. Further, the owner of the application best understands the application's internal formats and is best able to do the transformation. This allows the ESB and MOM sub-layers to be developed and maintained by an enterprise services group without requiring knowledge of the detailed application formats.

Further, performing the data transformations in the ESB or MOM sub-layers could potentially require n*(n-1) transformations and seriously reduce scalability as the number of applications passing data through the ESB increases.

4.5.5. Enterprise Software Environment Transformation Sub-Layer

4.5.5.1. Enterprise Service Bus Transformation Sub-Layer

The Enterprise Service Bus Transformation Sub-Layer transforms the physical capabilities of the ESBs as implemented. Virtually all of the vendors of ESB products provided the capabilities mandated by the applicable standards and these services can be provided directly without further transformation. However, to the extent that the Virtual ESB Sub-Layer specifies services beyond those available through the applicable standards, these services will need to be implemented for each deployed ESB. The additional services, beyond those required by the applicable standards or those provided by the specific product(s) that are being used will be implemented in this sub-layer.

4.5.5.2. System Services

Because of the lack of a Virtual Systems Services Sub-Layer, there will be no System Services Transformation Sub-Layer. The software in the Physical Systems Services Sub-Layer will be presented directly.

4.5.5.3. Identity Management Services

In the target state the Transformation Layer for the Identity Management Services will be lightweight, as most of the identity management services are provided directly i.e., the virtual and physical layers are very similar. While the LDAP based directory will be implemented in the physical layer, the physical LDAP message set will also be the message set / service calls that are exposed at the logical layer. VA developed identity management services will be developed and exposed as services meaning that the virtual sub-layer will conform to the physical sub-layer.

However, until there is such a single integrated identity management system used across all systems there will need to be much more work to be done in the transformation layer. Currently there are a large number of identity management systems being used, each with its own identity repository, and the transformation layer will need to provide more uniform access to the identity information in these various identity repositories and to be able to present a single uniform view of a person's identity prior to the deployment of the single integrated identity management system. This will potentially require the correlation or alignment of roles between multiple systems so that common roles across systems can be identified and individuals can be provided appropriate access rights across the range of systems.

4.5.5.4. Message Oriented Middleware Transformation Sub-Layer

The MOM Transformation Sub-Layer transforms VA standard messages to messages that can be transported by the MOM COTS software. This will primarily be a transport transformation since the MOM software transports messages regardless of their content or format. The transformation layer will also support providing any of the services identified in the virtual layer not provided by the COTS software.

4.5.6. Physical Enterprise Software Environment Sub-Layer

The Physical Enterprise Software Environment Sub-Layer housed the physical software required to provide the capabilities discussed in the sections above.

4.5.6.1. Enterprise Service Bus Sub-Layer

Multiple ESBs are likely to be developed and deployed throughout VA, each instantiation providing some or all of the capabilities described above. When an ESB is developed to support a single

application, functions such as "publish / subscribe" messaging or full registry services would not normally be required. ESBs supporting larger parts of the enterprise would be expected to support all of the capabilities described above. ESBs may be implemented using one or more vendors' products and the selection of vendors and products may vary across VA subject to VA efforts to reduce the number of different products used across VA.

4.5.6.2. System Services Sub-Layer

The Systems Services Physical Sub-Layer consists of a set "system-like" software packages, such as web servers and application servers, which will be installed and be made available within VA.

4.5.6.3. Identity Management

The long term target would be for there to be a single, integrated identity management solution across all VA system users – or perhaps on for Veterans, one for internal VA users (e.g., employees and contractors), and possibly one for other stakeholders. There would need to be interoperability in that each VA systems would accept the identity information from the appropriate sources (i.e., there will clearly be systems that employees can access but are not in general accessible to Veterans).

In the nearer term, there is unlikely to be a single integrated identity management system used by systems across the VA. What is more likely, and the direction in which the VA is currently heading, is to recognize that there will be a relatively large number of identity management systems as the integration is occurring at the MI and business area level, not the enterprise area. This would lead multiple sources of identity which then must be made to interoperate – typically through some more centralized hub or broker, that allows systems that are built to use one source of identity to validate the identities of persons whose identity is managed by another source of identity information or to correlate identities between multiple sources of identity information.

The Identity Management Physical Sub-layer will consist of a variety of software, both COTS and custom developed. The core identity management repository / repositories will be COTS products, typically LDAP compliant COTS products. In addition, to the directory / repository there needs to be either a token based system (e.g., Kerberos) in which all applications can accept the tokens (e.g., most major COTS application are "Kerberized") or the Identity Management system must install agents on each server to mediate access to applications. While this software will operate in the physical sub-layer, it will not be visible to the users.

4.5.6.4. Message Oriented Middleware Sub-Layer

The Physical Enterprise Software Environment Message Oriented Middleware Sub-Layer contains the MOM packages that are used in VA. Multiple COTS MOM packages are currently in use within the as well as a number of VA developed applications that perform similar functions. While these products will continue to be used, at least for a period of time, the number of such products should decline over time. These COTS packages will be used as the transport for the VA Standard Messages sent as described in Layer 2 (Section 4.2).

4.6. Layer 6 – Data Layer

The things that VA collects data about and the basic data that it collects will remain relatively constant. What will change is how VA collects and distributes the data – its applications and how it stores and accesses the data – its databases and other data assets. The Data Layer implements the OneVA Enterprise Data Architecture (EDA) and the OIT Enterprise Data Management Principles

(EDMP); it is the services running in the Data Layer of the EAA that implement that data architecture and the data management principles and that access the physical databases.

The EDA contains the Enterprise Conceptual Data Model (ECDM) that describes the subject areas in which VA has an interest. The EDA also contains the ELDM that identifies the data elements and entities (classes) and the metadata that is collected about each one. The ELDM is the critical artifact in achieving semantic harmonization of data as described in Section **Error! Reference source not ound.** below.

4.6.1. Relationship Between Services in the SOA and the Data Layer

Data services exist in the Layer 1, the SOA Services Layer in the Data Services Sub-Layer, and in Layer 4, the Data Layer. This section describes the differences in the services at the two layers.

4.6.1.1. SOA Services Data Sub-Layer vs. Data Layer

SOA Services implement the business rules that are the raison d'être for the application as they are the core of the application processing. The implementation of the business logic is kept distinct from the data logic as well as distinct from the control logic. The Data Services Sub-Layer in the SOA Services Layer will be based on the specific data elements in the ELDM as well as additional internal data elements needed to operate the system. The Data Services Sub-Layer will also be explicitly dependent upon the definitions of those data elements. Basically, the business rules are used to acquire values for data elements, transform the values of data elements, and transmit values of the data elements. Each of these operations is based directly on the precise specification of the business rules and the data element definitions. Were either of these to change the other would necessarily need to change.

A major function of the business rules is the validation of input prior to the data stores being updated. This will include performing all required edits and ensuring that data updates are valid before they are added to the database. While data can be accessed without the execution of complex business rules – other than those necessary to ensure that security and privacy requirements are met, data update will require the execution of the business rules, not only to perform edit checks of the input data but to ensure that the new values of data elements being updated are consistent with the values of data elements already in the data store.

The Data Services Sub-Layer in the SOA Services Layer does not have access to the physical data stores and will be independent of the physical structure of the underlying data. The Data Services Sub-Layer in the SOA Services Layer are logical services that deal with the data at the logical level.

4.6.1.2. Data Layer Services

The data logic in the Data Layer provides the mapping between the logical data elements and the physical instantiation of those data elements. Data Layer services access the physical data stores and databases and are explicitly dependent upon the structure of those databases and data stores.

A service will ask for some data, neither the SOA Services – Data Services Sub-Layer nor the logical model specifies how or where the individual data elements are stored. The Data Layer will specify where and how each data element is stored. Data logic specifies how the requested record is assembled, regardless of in how many tables or even DBMSs that the data is actually stored. By providing the fully assembled record to the SOA service in response to a service request, the Data Layer insulates the SOA Services from the physical data structures and allows the physical structure of the data to be modified without requiring changes to the business logic. The data logic will be

required to change with every change to the underlying physical structure. Data logic may be implemented as stored procedures within the DBMS or in separate services.

The Data Layer will also be the home for non-service oriented data access and COTS products used to provide custom or specialized data access such as Business Intelligence (BI) systems, records and document management systems, statistical analysis and reporting systems, and other specialized data analysis and data management services. The capabilities of these products would be provided in native form rather than through a series of service facades.

4.6.2. Data Layer Description

The sections below described the Virtual Data, Data Transformation, and Physical Data Sub-Layers.

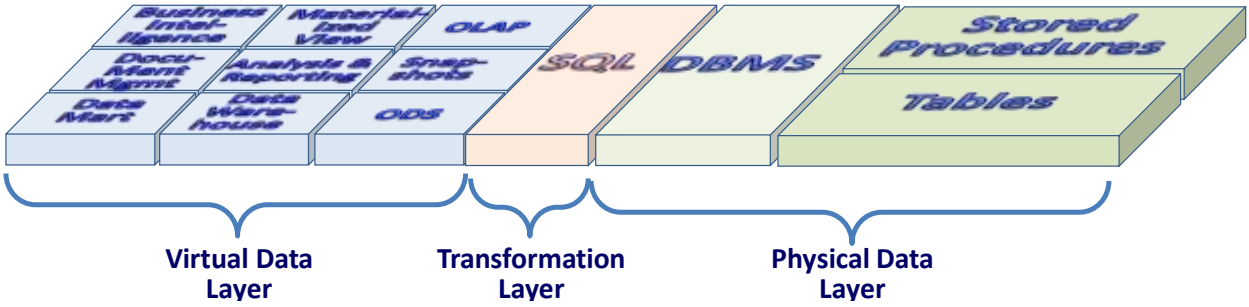

Figure 23 -- Data Layer Sub-Layers

4.6.2.1. Virtual Data Sub-Layer

The Virtual Data Sub-Layer Data logic specifies how the requested record is assembled, regardless of in how many tables or even DBMSs that the data is actually stored. The Virtual Data Sub-Layer provides the SOA data services an interface to the physical database. By providing the fully assembled record to the SOA service in response to a service request, the Data Layer insulates the SOA Services from the physical data structures and allows the physical structure of the data to be modified without requiring changes to the business logic in the SOA Services Layer.

The Virtual Data Sub-Layer also provides “materialized views,” A materialized view is a database object that contains the results of a query. It is a local copy of data located remotely, or is used to create summary tables based on aggregations of data from multiple tables.

In a database management system following the relational model, a view is a virtual table representing the result of a database query. Whenever an ordinary view's table is queried or updated, the DBMS converts these into queries or updates against the underlying base tables. A materialized view takes a different approach in which the query result is cached as a concrete table that may be updated from the original base tables from time to time. This enables much more efficient access, at the cost of some data being potentially out-of-date. It is most useful in data warehousing scenarios, where frequent queries of the actual base tables can be extremely expensive.

In addition, because the view is manifested as a real table, anything that can be done to a real table can be done to it, most importantly building indexes on any column, enabling drastic speedups in query time. In a normal view, it's typically only possible to exploit indexes on columns that come directly from (or have a mapping to) indexed columns in the base tables; often this functionality is not offered at all.

The Virtual Data Sub-Layer will also be the home for non-service oriented data access and provides access to custom or specialized data analysis products such as Business Intelligence (BI) systems,

records and document management systems, statistical analysis and reporting systems, and other specialized data analysis and data management services. The actual COTS products would be housed in the Physical Data Sub-Layer.

Because they are extracts of operational data from OLTP systems, ODS', data warehouses, and data marts exist at this sub-layer and are fed by the Extract Transform and Load (ETL) processes from the underlying OLTP data stores.

4.6.2.2. Data Transformation Sub-Layer

The Data Transformation Sub-Layer is a fairly lightweight sub-layer providing the link between the data services exposed by the Virtual Data Sub-Layer with the physical database accesses in the Physical Data Sub-Layer. The Data Transformation Sub-Layer would either provide SQL access to the physical tables or perform ETL functions to move OLTP data from the physical database to data warehouses or data marts.

4.6.2.3. Physical Data Sub-Layer

The Physical Data Sub-Layer contains the physical DBMS', databases, and data analysis tools that hold, access, and analyze VA's data. This is the level at which the Database Administrator's (DBA's) and deals with the physical DBMS product that is used to support the databases, the physical databases, and the tables and meta-data that is used to support the physical storage of the data.

The Physical Data Sub-Layer takes into account the facilities and constraints of a given database management system. The Physical Data Sub-Layer implements the Physical Data Model that includes all the database artifacts required to create relationships among tables or achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables, or clusters.

The Physical Data Sub-Layer is explicitly dependent upon the commercial DBMS that is being used and the physical database and table structures that are in use, and of necessity change based on any change to the underlying physical data structures that are being used. Any stored procedures associated with the database would exist at this level.

4.7. Layer 7 – Management Software Environment

The Management Software Layer includes "quasi" systems software, which while not part of the operating system, is software that many would either consider to be part of the operating system or at least very close to the operating system. The management software may either provided by the operating system vendor or, more likely, to be provided by third party vendors because one function of this software is to make multiple, possibly diverse systems to appear to be a single system. Because of the "quasi" system nature of the software, this software is not normally visible to application developers or users.

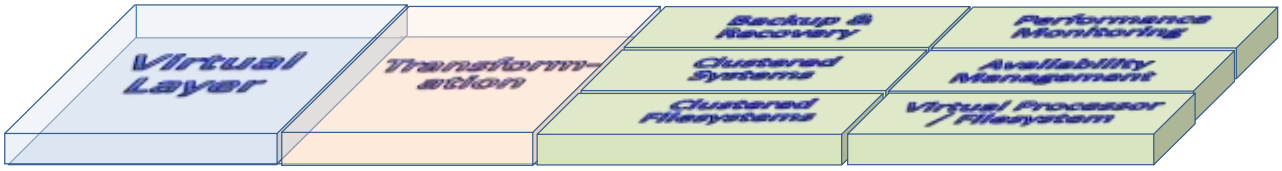

Figure 24 – Management Software

The Management Software includes:

- Backup and Recovery.

- Clustered Systems / Clustered Filesystems.
- Performance Monitoring.
- System Monitoring.
- Availability Management.

4.7.1. Virtual Management Software Environment Sub-Layer

The virtual layer is extremely lightweight, to the extent of being non-existent, as the Management Software is, for the most part, designed to be transparent and provide a virtual environment.

4.7.2. Management Software Environment Transformation Sub-Layer

Because there is no Virtual Management Software Environment Sub-Layer, there is no transformation sub-layer.

4.7.3. Physical Management Software Environment Sub-Layer

The Physical Management Software Environment Sub-Layer consists of a series of products that control the environment in which the applications operate. Software at this layer includes:

- **Backup and Recovery** – To perform backups of application and data and to allow recovery of the system in the event of a disaster or the need to roll back files. This is in addition to any data replication or snap copies of data made by the storage systems or the DBMS.
- **Clustering** – Is used to make multiple devices appear to be one higher capacity device, either to provide higher availability or higher capacity, or both, depending on how the systems are configured and the way in which they are used. Clustered processors are used to make multiple processors as a single larger processor or to provide redundancy. Clustered filesystems are used to allow filesystems installed on multiple systems to appear as a single filesystem.
- **Performance Monitoring** – A set of tools used to measure the utilization, service levels and responsiveness of system components to allow systems personnel to ensure that the system will meet performance goals.
- **System Monitoring** – Manage and monitor systems and system resources. The systems monitoring software typically require endpoints on each of the systems being monitored.
- **Availability Management** – Ensure that system resources and to reassign processor resources when there is a system or component failure.

4.8. Layer 8 – Hardware / OS Environment

The Hardware / Operating System (OS) Environment includes the physical devices and operating system and other device specific software required to allow those devices to operate. The Virtual Hardware / OS Environment Sub-Layer provides a virtual environment in which higher level services operate. The Physical Hardware / OS Environment Sub-Layer describes the actual physical environment in which VA systems operate. The Hardware / OS Environment includes sub-layers for communication networks, processors, and storage.

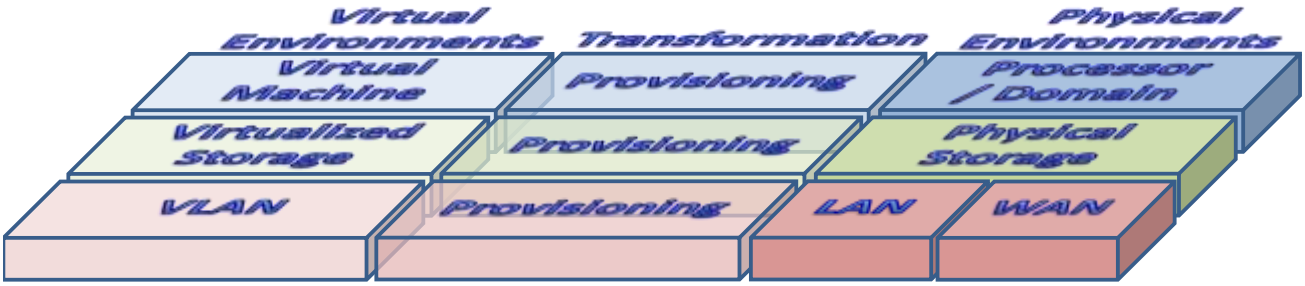

Figure 25 -- Hardware / OS Environment

4.8.1. Virtual Hardware / OS Environment Sub-Layer

The Virtual Hardware / OS Environment is the representation of the physical environment that the program sees and programs to.

4.8.1.1. Virtual Processors

Best practice in computing center operations dictates the use of standard configurations that minimize variability across the enterprise. Therefore, VA systems will be based on a relatively small set of standard, virtual configurations to be used throughout the enterprise reducing operational complexity. Reducing the number of configurations both reduces overall costs and improves reliability by simplifying the environment and operations. VA users will be provided virtual systems images for use, and VA operations will assign the applications' virtual systems to physical systems. The virtual systems will be defined as a limited, preset number of physical configurations consisting of both a standard set of hardware configurations and software images.

Ideally, all VA systems would be developed using a series of single processor, single thread processors. Using this model each major application component of a system would be supported by a separate, single processor system. However, this is not a realistic target environment as there are considerations that applications developers need to be aware. Therefore, the basic Virtual Machine (VM) will be a single processor, multi-thread, multi-core processor. Individual system components (e.g., web servers, application servers, and data base servers) will be implemented in separate VMs. However, there are some significant limitations to this processing model, notably the limitations on system capacity and the lack of system redundancy. Therefore, the basic processing will be clusters of single processor, multi-thread, multi-core processors. Clustered processors improve both scalability and availability – scalability by providing the ability to add processors as needed to meet capacity and performance requirements and availability by ensuring the availability of processing resources in the event of the failure of any single device failing by ensuring that there is no single point of failure.

Microsoft Windows Server and Red Hat Enterprise Linux (RHEL) are the two allowed operating systems in the virtual environment and will carry over into the physical environment.

4.8.1.2. Virtual Networks

The applications will see a dedicated, flat virtual network where both all processors in the application and all processors and / or devices that a processor needs to access are directly accessible. Network partitioning, network devices, such as firewalls, switches, routers, proxy servers and intrusion detection devices etc. will not be visible to the user applications, but will need to be included in the physical network. The user systems will be able to see the virtual network dedicated to the transport

of user data and will not see the VLAN(s) or physical networks dedicated to systems management or data backup etc.

4.8.1.3. Virtual Data Storage

Each application will be provided what appears to be one of more dedicated filesystems, each of unlimited size, each installed on a local device internal to the processor.

Storage services include not just storage devices but the infrastructure required to support and operate those devices. This storage will be fully virtualized and servers will have no knowledge of the actual device on which its files are stored. Further, files may be moved to higher performance or lower cost storage as required to optimize enterprise operations.

4.8.2. Hardware / OS Environment Transformation Sub-Layer

The transformation of the virtual devices in the virtual environment to the physical devices in the physical environment is accomplished through the provisioning process. Provisioning is a set of actions to prepare a server, network, or storage unit with appropriate systems, and data and software configuration, and make it ready for operation. Typical tasks when provisioning a server, network connection and storage unit are: select a device from a pool of available devices, load the appropriate software (operating system, device drivers, middleware, and applications), appropriately customize and configure the system, device, and the software to create or change a boot image for the server or device allocation tables and then change its parameters, such as IP address, IP Gateway. This provisioning includes:

- Assigning VMs on physical processors, either on individual processors or in a domain or region on a larger Symmetrical Multiprocessing Processing (SMP) system
- Assignment of physical switch ports, IP addresses, and firewall rules etc. to instantiate a VLAN onto the physical network infrastructure
- Allocation of LUNs and disk storage to instantiate a virtual filesystem in physical disk systems

4.8.3. Physical Hardware / OS Environment Sub-Layer

The description of the Physical Hardware and OS Environment will be necessarily limited because the actual definition of the physical environments is beyond the scope of this EAA. VA systems engineering and the teams at each of VA datacenters will be responsible for defining and deploying the actual physical environments at each of the VA datacenters to which the virtual environments will be mapped. Because of this there will be no specification of the physical environments in this EAA, except to say the goal being that, because new applications and major enhancements to existing applications will be designed be deployed on standard systems, that the number of physical configurations will hopefully be extremely limited.

4.8.3.1. Physical Servers

The physical servers will be Intel based commodity servers running Windows Server or Red Hat Enterprise Linux (RHEL).

4.8.3.2. Physical Storage

Physical storage will be based on a number of shared Storage Area Network (SAN) units that house storage for large numbers of VA applications. The SAN units will provide disk mirroring and remote backup and replication of application storage.

4.8.3.3. Physical Networks

The VLANs described above will be mapped to physical LANs and WANs.

4.9. Security & Privacy Tower

Assuring the ability to secure VA applications and protect both the PII and PHI that they store. Implementation of an SOA creates new vulnerabilities and different avenues for attack than are present in the development of monolithic, or even distributed, applications. It is the intention of this section to discuss architectures, mechanisms, and approaches to closing the holes and lessening the vulnerabilities across all layers of the architecture, but special emphasis is placed on security related to the use of and SOA and ESB.

SOAs are based on the definition of a set of common services that are used by all applications and which typically do not differentiate between calling applications. Since services are called and processing is done outside of the calling application, it is quite possible that the service will be provided from a process in a domain other than the one in which the service consumer resides. Because of the nature of services and SOAs, an application may be unaware of the security domain from which a service it consumes is provided, or even if, at some point, the service is moved from one security domain to another. Any flow of information between security domains introduces potential vulnerabilities. While multiple mechanisms and protections can be included on top of the services, this introduces a layer of cost, processing and inefficiency.

Because of the sensitivity of the information processed by VA systems, it will be critical for the SOA to be developed in a manner consistent with established VA Security Architectures so as to ensure that vulnerabilities are not introduced by the introduction of the SOA and to keep additional complexity and processing to a minimum. A key to ensuring the required level of security of the VA enterprise will be the very careful integration of the SOA with the existing security architectures.

This EAA describes the system development environment in terms of a series of layers, each layer providing a different type of capability. Because different capabilities are provided in each layer are different, the threats and vulnerabilities for each layer will be different, and therefore, the security mechanisms applicable to each layer will also be different. This section discusses security mechanisms that are applicable to each layer.

The security discussion provided here is not intended to supersede or replace any security guidance, directives, processes, or procedures, but only to describe how security mechanisms will be applied at each layer in the architecture.

The discussion below is not intended to be a complete discussion of security. This EAA is intended primarily towards application designers and developers and thus, concentrates primarily towards the aspects of security that would be visible to application designers and developers.

4.9.1. SOA Services

Since the SOA will be an integral part of the VA enterprise, the security mechanisms used to support the SOA will need to be integrated with the security mechanisms for the rest of the enterprise. Because the SOA provides new access paths to systems and data it follows that these paths must be protected to the level that the rest of the enterprise is protected, lest the SOA provide new less

protected paths to current systems. One side effect of the use of an SOA is to remove applications boundaries and to allow access to code that once resided inside of an application and was thus, inaccessible. That this code can now be exposed as a service and made accessible introduces vulnerabilities that require protection. The SOA also provides increased mechanisms for external access (e.g., internet access) to component functionality. This too introduces vulnerabilities that must be defended against.

New applications will be built based on a set of services built in accordance with the SOA. Service-based interfaces will be built for legacy systems to allow service requests to be used as the primary interface to those systems. This will inextricably tie the security and access control approaches used to support the SOA with those that support the legacy and modernized systems.

The approach that will be taken to providing services security includes providing authentication and authorization to service requests to control access to the services and ensuring that the coding of the services does itself does not introduce additional vulnerabilities.

The authentication and authorization services will be provided by the ESB Sub-Layer and are discussed below. The approach to ensuring that the services themselves do not introduce system vulnerabilities will be based on 1) the training of designers and developers of design and coding practices that are known to introduce security vulnerabilities into a system. If "design patterns" are descriptions of "best" design and development "practices", these known vulnerabilities are "anti-patterns" – design patterns that if followed are ensured to introduce vulnerabilities.

Vulnerability scanning software will be used to scan the code for all services prior to deployment to ensure that the code does not introduce any known software vulnerabilities to the system.

4.9.2. Enterprise Standard Messages

Enterprise Messages are logical messages, not physical messages. At the virtual sub-layer they are VA standard, syntactically and semantically harmonized messages. At the physical sub-layer they are messages that conform to one of a number of external standards (e.g., HL7 or NIEM etc.). Neither of these message formats are dependent upon the transport layer, so security mechanism, such as encryption are not part of this layer – although they will be included at lower sub-layers.

Security at this level includes the use of standard headers for messages to ensure the auditability of the messages. The standard messages will include information as to the identity of the requestor of all information and the user session that originated the request – regardless of how many services or systems through which the request has passed.

Further, electronic signatures can be used to ensure the authenticity of messages that have been sent, to enforce non-repudiation (ensuring that the sender cannot claim to have not sent the message), and to ensure that messages have not been modified in transmission. This is done by encrypting a hash of the message using the sender's private key. Decrypting the message using the sender's public key and checking the hash ensures the authenticity and accuracy of the message. Encryption with the senders private key ensures that they message was sent by the sender.

4.9.3. Interface Layer

The interface layer includes two types of interfaces – physical devices attached to VA systems and interfaces to external systems. Security for physical devices will be provided by a number of mechanisms. First, physical access to the devices will be restricted by their physical location and the access controls inherent in access to the locations of the devices. Each physical device will be

attached to a VA system and access to devices will be managed and secured by access to those systems.

Interfaces to external systems will be of two types – web services interfaces and non-web services (other) interfaces. The web services interfaces will be secured using the web services standards including, but not limited to WS-Security and SAML. Both of these mechanisms require X.509 certificates and digital signing of messages.

Non-web services include both all legacy interfaces and new, non-web services mechanisms. A basic assumption regarding external interfaces is that VA cannot unilaterally change existing interfaces, and that new interfaces will need to be developed based on agreement with the external parties. The legacy interfaces are what they are, and each interface has a mechanism by which VA system that is attached to the interface verifies the party on the other end of the interface. Whatever that mechanism is will continue to be used since VA has no ability to change the mechanism.

For new external interfaces, the strong desire would be for the interfaces to be based on digitally signed encrypted messages. The digital signatures and the encryption should be based on X.509 certificates issued by a certificate authority subordinate to the Federal Bridge or cross-certified with the Federal Bridge. ISAs, ICDs, and MOUs would be required for new external interfaces.

4.9.4. COTS Software

The two classes of COTS software present very different security issues. The Functional COTS software will typically be used only within a small segment of the VA population and then used only for a specific purpose. Further, the amount of data contained in a specific Functional COTS is likely to be somewhat limited. The Application Services COTS packages present a very much different problem in that they are very widely used and contain large aggregations of data, making them very tempting target for compromise.

4.9.4.1. Functional COTS

The functional COTS are a series of diverse COTS packages similar only in that they provide functionality that VA has determined would be more efficient to buy than to custom build. The functional COTS software packages will need to be integrated with VA single sign-on solution and an SOA facade will need to be constructed for those COTS packages that do not natively provide an SOA interface.

No changes should be made to the COTS software but, if changes are required for either technical or programmatic reasons, all such changes should be made through the services that they expose of through their standard APIs. If the continuing need to maintain custom changes to COTS software were not in itself sufficient reason to not allow such changes, such changes also have the potential to introduce security issues that were not present in the COTS software as delivered by the vendor.

4.9.4.2. Application Services

Application Services also includes a wide range of COTS software the define subsystems that the user will interact with and work within. However, the software at this sub-layer provides particular issues related to security and privacy, but especially privacy, as much of the software at this sub-layer deals with the collection, aggregation and organization of information. Types of software at this sub-layer that pose particular security and privacy issues include:

- Business intelligence.

- Knowledge management.
- Document management.

These support the collection and access of information and:

- E-mail.
- Social networks.
- Collaboration software.

These support the distribution and sharing of information. The issue with both from a security and privacy perspective is that while there are legitimate uses for all of this software and legitimate reasons why VA not only allows, but encourages, the use of this software, they create new collections of information and allow the rapid assembly and dissemination of large amounts of information.

The primary security mechanisms will be access controls and identification and authentication. Access to any software at this sub-layer will initially be controlled through USERIDs and passwords, but once an approach is deployed, through the identification and authentication capabilities provided by the VA single sign-on capabilities. However, access control only keeps unauthorized persons from accessing VA data, but does nothing to stop persons with legitimate access from accessing data beyond that which they need to do their job or from disseminating information that they access as part of their job.

VA policies and directives related to the implementation of security and privacy controls and the dissemination of VA information are in place which prohibit the unauthorized access or dissemination of VA information or any PII or PHI related to a Veteran, and VA employee, or any other person about which VA collects and stores information.

The discussion above pertains directly to software installed on VA computers. Users of VA systems and computers typically also have access to personal / non-VA e-mail accounts and personal access to one or more social networks. While VA policies and directives certainly should restrict any dissemination of information through these non-VA systems, it is beyond the scope of this EAA to discuss restrictions on the use of non-VA systems.

VA systems will collect log records showing who accessed the systems and what actions they took. These audit log records will be available to determine who accessed information and to be able to take action against anyone who violates VA security and privacy policies and directives.

4.9.5. Software Services

The Software Services Layer consists of a number of distinct sub-layers, each including a different type of software, and each with different threat profiles and thus, requiring different security mechanisms to protect against those threats.

4.9.5.1. ESB

The "*VA Service Oriented Architecture (SOA) – Technical Framework*" provides an extensive discussion of SOA and ESB security. That discussion will not be repeated here. Only a brief summary of that discussion is provided.

A security domain may consist of one or more hosts, one or more LANs at a site or a group of networks connected via a network provider or backbone, and is usually implemented as either a physical network segment or a private VLAN. It is distinguished by the fact that security devices at

one or more access chokepoints mediate access to hosts or network components within the domain from any host or network component outside the domain. The security perimeter is the set of all access points, such as routers or modems, through which the hosts and network components within the domain are connected to networks or other components outside. Forcing all accesses into the domain from entities outside, and vice versa, to pass through security devices at the access points protects the domain and enforces a policy at the security perimeter. Typically, there will also be physical protection at a security perimeter. The ESB is implemented as a security domain

A "domain of trust" based security architecture requires a high level of perimeter security to guard against intrusions from outside of the domain and allows a high level of trust between all processes within the domain. This perimeter security is normally provided by a security gateway and because many of the messages are XML based, the security gateway is usually serves as an XML firewall.

The operational concept for the SOA infrastructure is that it will be implemented is that the ESB will be implemented as a single security domain instantiated as a logically isolated network segment (i.e., either a physically isolated segment or a VPN segment). The presentation and interface servers will be on a separate VPN or network segment from the business logic servers and other ESB components. Further, authentication will be performed on entry to the network segment (security domain) that provides the ESB capabilities. The SOA framework is based on a federation of such security domains (isolated network segments).

4.9.5.2. Systems Services

The Systems Services Sub-Layer includes a set of quasi system software which provide the bare systems their personality – i.e., whether the server will be a web server, application server, or data server etc. These systems are not accessed by VA users, but by VA systems and computer operations personnel. Security is provided through the same identification and authentication and access control mechanisms as operating systems software.

4.9.5.3. Identity Management

Identity Management software and data is perhaps the most sensitive data stored on any system. Identity Management software provides the "keys to the kingdom" inasmuch as the credentials and the information stored in the system are sufficient to provide access to any program or data on the system. Identity Management software needs to be protected like other Operating System software from unauthorized use and access. The Identity Management data stores will be fully encrypted to ensure that the information contained in them, including identity information on all VA personnel and every Veteran, is protected. Access to the Identity Management stores must be restricted to the extent feasible with the USERIDs and PASSWORDs very actively managed and changed frequently.

4.9.5.4. Message Oriented Middleware

The MOM layer is logical the transport layer for the enterprise messages. In recent years the General Accountability Office GAO has been insisting that messages be encrypted not just on public networks, but on local networks internal to the datacenter as well. Messages into and out of the MOM system can be encrypted both at the message (local link) layer – the MOM layer, and at the physical network layer. Encryption at the MOM layer includes encryption of the body of the message, but not the header to allow the message to be routed for deliver. It can be further protected by applying a digital signature to the message – whether the message is encrypted or not.

4.9.6. Data Layer

The data layer software contains the physical DBMS', the databases that they manage and other associated data management software. The security mechanisms provided at this level are primarily the security mechanisms provided by the DBMS to control access to the data. Modern DBMS', and certainly those in use in VA, provide a range of security mechanisms including ACLs, row or view security controls, encryption of "data at rest", multi-factor authentication, etc. These capabilities are internal to the DBMS and are intended to restrict access to the database or parts of the database.

Further, no users will be allowed direct access to the data layer services; access to the data layers services will be through the presentation layer or business layer services. Direct access to the DBMS will also be protected as operating system or other operating system-like software. The DBMS servers should be isolated network segments or VLANS to allow network security capabilities to the provide further security for the DBMS and associated databases.

4.9.7. Management Software

This layer includes "operating system-like" software such as backup and recovery software, cluster files systems etc. and will need to be protected in the same manner as the operating systems in the layer below.

4.9.8. Hardware / Operating System

The Hardware and Operating System Layer is the lowest layer in the model. It is here where there are access control and identification and authentication mechanisms to restrict access to the operating systems. In addition, to the software access control that is provided at this layer, like other layers, physical access control is also provided. Much can be done to a server if physical access is achieved, so access control at this level includes restricting physical access to the servers.

The other aspect of Hardware / Operating System is the isolation to of functions (e.g., application web server, web server, or database server) must be on different physical / virtual servers.

4.10. Systems Management

The System Management Tower describes the mechanisms that are provided at each level to manage the services provided at that layer. The EAA layer model describes a series of virtual services that are provided at each layer in the architecture. The services described in the Systems Management Tower are the management services that the developers of the services in the corresponding virtual services layer can assume will be available to support the management and reporting of services at their level.

4.10.1. SOA Layer

The SOA Services Layer supports presentation logic, business logic and data logic services. Each of the services should be supported by a specific Service Level Agreement (SLA). SLAs are discussed in The VA Service Oriented Architecture (SOA): Technical Framework.

In addition to the specification of the SLAs and the Service Contracts, there will be a mechanism to monitor the performance of the services.

There are two sets of products that will be used to measure service performance. First, there will be Business Activity Monitoring (BAM) software that is part of the ESB functional software, which is intended to build an OLAP database containing information on the results of the services and the business processes that they support. This software provides a high level view of SOA performance

in terms of business process oriented measures. BAM software measures the use of services – e.g., number of times a service is executed and length of time an orchestration takes to complete to provide measures of the performance of the business processes that the services support. A description of the BAM software is provided in The VA Service Oriented Architecture (SOA): Technical Framework.

The other type of software that can be used to monitor service performance is special purpose software designed for SOA monitoring and management. This software is able to measure service performance and SLA management. Products providing service monitoring and management capabilities look similar to the more tradition system management software products except that they are oriented towards the monitoring of SOA services rather than systems. Like the system management software, these products will require "endpoints" on each server that support processing of the service.

Only systems management software running at this level would even potentially be visible to the applications developers. Since the layers below the SOA Services Layer are not directly visible to application developers, the systems management software supporting those layers are also not visible to application developers.

4.10.2. Enterprise Messaging

The Enterprise Messaging Layer deals with logical messages between services, not the physical transport of information. The SOA management software should report the message latency times as part of the service response times as the processing of the service will also include the time required to pass between services that are included in a composite service. Since the services that are described at this level are all logical services, the network measurements will also be logical measurements.

4.10.3. Interface Layer

At this layer there is a differentiation between the interfaces supporting web services and those supporting non-web services interfaces. For web services, application firewalls are available which not only operate as OSI Level 7 firewalls, but can perform content inspection of XML packets to ensure that the message is well formed and generate log records for the authorization decisions. Logging of traffic over non-web services would also be performed.

4.10.4. COTS Software Layer

Systems management for the COTS software is general systems management and monitoring to assure COTS application availability and performance management to assure that there is sufficient system capacity to provide acceptable levels of performance. Performance management includes workload characterization and forecasting, resource analysis, and capacity analysis and forecasting.

4.10.5. Enterprise Software Layer

The Enterprise Software Layer contains a number of sub-layers, each supporting a different class of software. System management at each of these levels includes monitoring to ensure availability of the services and resource / usage monitoring to measure workload, resources, and capacity.

System monitoring and management are used to ensure the availability of the servers and systems. This will normally be done using endpoints for the installations' Integrated Service Management software.

The other major aspect of system management of this level is performance monitoring and capacity planning. It is not at the SOA service layer that resource measurement is performed, but at the enterprise software layer – i.e., the application servers on which non-BPEL services execute and the ESB elements used to support the SOA services. Box level performance management is performed at a lower layer and determines whether the workloads are being adequately balanced across processors and when processors must be added to the pool. At this layer performance management includes the determination of the resource requirements of services or applications to allow workload forecast to be developed.

4.10.6. Data Layer

Systems management at the Data Layer is primarily provided by the DBMS, enterprise backup and recovery software, and Integrated Service Management software. Modern DBMS' provide a wide range of management software support tools for management of the databases and optimizing their performance. Both the DBMS and the enterpriser backup and recovery software as well as the DBMS software provided backup and recovery of the databases.

4.10.7. Management Software

The Management Software Layer contains the Integrated Systems Management Software. These tools install endpoints on each instance of the operating system to provide a wide range of management services from monitoring the overall availability of the systems, providing management functions for processors, storage, and networks and displaying information on the operation of each of the system components through a single integrated console.

4.10.8. Hardware / Operating System

Systems Management functions at the Operating Systems Layer are primarily software provision and distribution and patch management. Common systems management software runs at the Management Software Layer. Software provisioning loads standard images and standard products on servers and other devices and applies patches to already deployed systems.

5. Application Data Architecture

Obviously the design and development of systems and services clearly depend on the underlying data and the data architecture used to manage that data. This section describes the Application Data Architecture (ADA). This section is not a replacement for the EDA but extracts and summarizes data architecture principles and applies them to the development of systems and services.

The things that VA collects data about – e.g., Veterans' education, home loans, healthcare and the basic data that it collects e.g., Veteran's names, educational status, results of medical test etc. stays relatively constant. What new systems and services will do is change is how VA collects and distributes the data – its applications and how it stores and accesses the data – its databases and other data assets. While VA may always collect the same information, the system in which the data is processed may change over time, the DBMS they are stored in may change and within a DBMS the table layouts may change over time. The Application Data Architecture (ADA) describes those things about VA's data that are likely to remain relatively invariant and the rules for managing and sharing that data.

The EDA contains artifacts that describe the subject areas in which VA businesses have an interest. The EDA contains a number of artifacts, the most important of which is the ELDM that identifies the

data elements and entities (classes) and the metadata that is collected about each one. One key concept that forms the basis for the ADA is that there will be a "single version of the truth" in VA – there will be a single authoritative source for each data element. Another of the ADA's key concepts is the separation of data logic and application logic. Because of the separation of business logic and data logic, applications should have no knowledge of the physical data structures within which the data is stored, which allows those data structures to be changed without impacting the business logic. All data access is to be through very light weight services. These services isolate business logic from data logic and provide a mechanism for hiding the physical data structures from the applications. The data architecture also differentiates between OLTP data stores and data warehouses and DMs. Only OLTP data stores can be authoritative. VA needs to be able to present a single version of the truth both for use internal to VA and when reporting information outside of VA. This requires that DMs be federated and that they all receive their data from a single source so that all queries related to a single set of data are answered in the same way regardless of from where the query is made.

Data architecture can exist at both the enterprise and the project level. This ADA is primarily oriented towards the development of systems and services and the sharing of information between those systems and thus, provides direction as to the management of data rather than artifacts describing the data. While, the discussion in this section deals primarily with data architecture at the enterprise level, it is assumed that each project will develop a data architecture for their project and that the project data architectures will implement the concepts discussed in this section so as to allow the projects to meet their responsibilities.

5.1. Relationship of VA ADA to the VA DRM and the OneVA EDA

Where VA DRM is concerned with the data that VA maintains, how it is defined, and how it is / can be shared, this ADA is concerned with the rules and frameworks under which applications access data and the rules for accessing and maintaining that data. Where the DRM is vitally concerned with the entities and data element definitions that are contained in the ECDM and ELDM, this ADA does not deal with such topics. This ADA assumes that complete and consistent ECDMs and ELDMs have been developed as part of the EDA and are available, or that their development will be completed in the near future. This ADA deals with acquiring, maintaining, and providing that data – it provides the basis for the development of application systems. It is not concerned with the functional content of those systems.

This ADA provides mandatory direction as to how data is to be made available to applications and how applications access that data.

5.1.1. Relationship of VA ADA to VA DRM

The main purpose of the Office of Management and Budget OMB Federal Enterprise Architecture (FEA) DRM is to enable effective data management, information sharing and data reuse across agencies and is described by OMB as follows:

"The DRM is a flexible and standards-based framework to enable information sharing and reuse across the federal government via the standard description and discovery of common data and the promotion of uniform data management practices. The DRM provides a standard means by which data may be described, categorized, and shared. These are reflected within each of the DRM's three standardization areas:

- ***Data Description** – Provides a means to uniformly describe data, thereby supporting its discovery and sharing.*

- ***Data Context –*** *Facilitates discovery of data through an approach to the categorization of data according to taxonomies. Additionally, enables the definition of authoritative data assets within a Community of Interest (COI).*
- ***Data Sharing –*** *Supports the access and exchange of data where access consists of ad hoc requests (such as a query of a data asset), and exchange consists of fixed, reoccurring transactions between parties. Enabled by capabilities provided by both the Data Context and Data Description standardization areas "*

Unlike the other four reference models, the DRM is a high-level model covering data management and exchange, and is implemented by creating a number of data artifacts to perform the multiple functions specified in the DRM. These artifacts include:

- Conceptual and LDMs.
- Data Asset Catalog.
- Information Exchange Repository.
- Other artifacts as the need arises in the future.

5.1.2. Relationship of VA ADA to OneVA EDA

The OneVA EDA consists of a set of artifacts and a set of rules for how data is to be managed, shared, and protected. The artifacts included in the Enterprise Data Architecture will include:

- Enterprise Conceptual Data Model (ECDM).
- Enterprise Logical Data Model (ELDM).
- Data Asset Catalog.
- Information Exchange Matrix.
- VA Common Glossary, Vocabulary, or Lexicon
- VA Information Taxonomy and Ontology.

Project PDMs are not normally included in an Enterprise Data Architecture, as they are project artifacts. The VA ADA does not itself contain artifacts, but is the subset of the EDA containing information of importance and interest to application and service developers.

5.1.3. Relationship of VA ADA to iEHR CIIF

The iEHR Common Information Interchange Framework (CIIF) is a mechanism designed to allow the exchange of syntactically and semantically harmonized information between VA and DoD health systems – although neither CIIF or iEHR use those terms. Both the CIIF and the ADA have similar goals in that they provide the basis for developing common standard data element and message syntax and semantics across multiple systems and as such, address many of the same issues. Figure 26 illustrates the cross walk between the components the CIIF and the ADA. The components of the ADA are described in more detail in Section 5.6 where the data artifacts required by the ADA are described. The development of the required ADA artifacts as part of the larger VA Data Architecture effort will support the development of the CIIF and facilitate the interchange of information across VA systems and between VA systems and DoD systems.

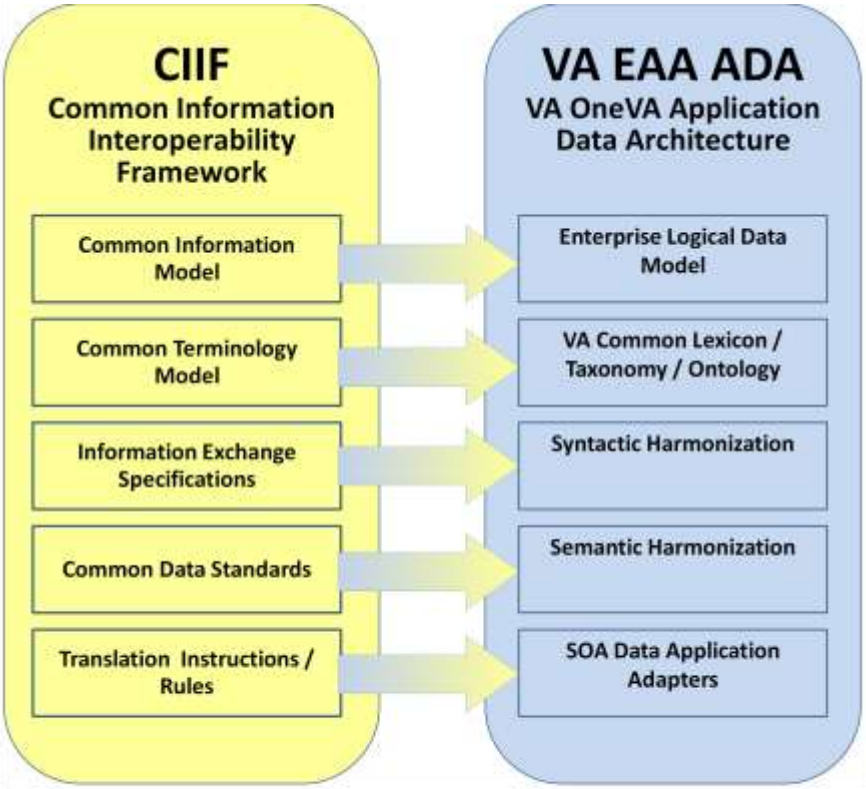

Figure 26 -- Relationship between VA ADA and iEHR CIIF

5.1.4. Relationship Between the VA ADA and the VA SOA

There is a very close relationship between the OneVA ADA and the VA SOA. The ADA summarizes the rules and policies for the management of data within VA as specified in the OneVA EDA, but it will be the VA SOA that provides the mechanism for that sharing. As an example of the close relationship between the EDA and the SOA is that the SOA is based on the use of standard, semantically and syntactically harmonized messages and standard data elements. While this semantic and syntactic harmonization is of critical importance to the SOA, the work to perform the required harmonization will be done as part of the EDA.

Since the EDA describes the rules for the management and sharing of data in VA, the application and service developers will need to abide by these rules. These rules will place constraints on the development of the applications and services. The rules will also place responsibilities on the Data Stewards application and service developers.

The VA Enterprise Data Management Principles require that there be a functional and technical data steward for each entity who are responsible and for performing the data management and technical support for the data items.

5.1.4.1. Functional Data Stewards

The Functional Steward represents the “owner” of the entity. The “owner” of the entity is the business organization responsible for collecting the data and ensuring its accuracy, timeliness, and consistency. In addition, the Functional Steward is responsible for identifying the applicable security, privacy, records management and records retention requirements – with records retention requirements subject to approval by National Archive and Records Administration (NARA). The steward is also responsible for designating the domain management rules which govern processes for creating and approving new, valid data values.

The Functional Steward and the Technical Steward shall have the responsibility for sharing the data with all VA applications and organizations with a need for the data in accordance with applicable security and privacy laws, regulations, and policies – both PII and PHI.

5.1.4.2. Technical Data Stewards

The Technical Steward is the owner of the database(s) or data stores used to store the data and the systems and services used to access the data. The Technical Steward decides on the physical storage of all VA data (e.g., if it is stored in a single or multiple databases) and ensures the enforcement of consistent data use by all application developers as designated by the Functional Stewards and that it is consistent and in compliance with applicable security, privacy, records management, and retention requirements – with records retention requirements subject to approval by NARA.

5.1.4.3. Data Stewards Responsibility for Data Sharing

The Functional Steward shall determine with whom data may be shared, while the Technical Steward will be responsible for the technical manner in which the data is shared. While the desire is to share data as widely as possible, the Functional Stewards and the Technical Steward shall determine the rules for access and update of data based on the need to minimize risk of any inadvertent or unauthorized disclosure; subject to any applicable laws, regulations, or policies etc. that impact who may see / access the data; how or by whom the data may be changed; archiving and retention requirements for the data; as well as any other aspects or data access, update, assurance etc.

5.1.4.4. Single Authoritative Instance of all Data

One key concept that forms the basis for the EDA and the EDMP is that there will be a "single version of the truth" in VA – there will be a single authoritative source for each data element.

A single instance of the data asset containing the data element values related to each entity will be designated as "Authoritative" and will serve as a unique and unambiguous source of data to be shared operationally across all systems in the enterprise. This authoritative instance is sometimes referred to as a "Copy of Record" (COR). It will be the single, unambiguous "truth" for VA. COR data must be either perpetual or effective-dated. The COR will exist at the OLTP site where the data is maintained for the following reasons:

- Because they are snapshots of OLTP perpetual and effective-dated data, Online Analytical Processing (OLAP) (i.e., Data Warehouses or DMs) can never be designated authoritative.
- Reference data is a special class of perpetual data with a very low update frequency and high access rates. "Reference" data (e.g., lookup tables) may be replicated across the enterprise provided that there is a publish/subscribe mechanism to publish changes across the enterprise.

Copies can be made by other systems for performance purposes, but changes to those copies cannot be considered to be effective until the authoritative instance is updated. The EDA will identify the authoritative instance for each of the entities in the ELDM.

5.1.4.5. Separation of Business Logic and Data Logic

Another of the EDA's key concepts is the separation of data logic and application logic. Because of the separation of business logic and data logic, applications should have no knowledge of the physical data structures within which the data is stored, which allows those data structures to be changed without impacting the business logic. All data access is to be through very light weight services. These services isolate business logic from data logic and provide a mechanism for hiding the physical data structures from the applications.

The ADA also differentiates between online (OLTP) data stores, Operational Data Stores (ODS'), and Data Warehouses (DWs) and Data Marts (DMs). Only OLTP data stores can be authoritative.

5.2. Single Logically Integrated but Physically Distributed Database

VA data must be made available to all within VA who require it, and since data will be accessed through a series of data services that hide the physical structure of the databases in which the data is stored from the business services that use that data, there is no reason that all data a business service accesses need be co-located. All VA data assets should be viewed as being stored in a single logically integrated, but physically distributed database. That is, while the VA data should be able to be viewed as being contained in a single logically integrated database – a single database at a single location, even though the data may be stored in a number of discrete physical databases distributed throughout VA. Data services can be constructed to provide access to data from multiple physical data based and provided to its consumers as if all of the data came from a single local database.

5.3. Classification of Data and Data Stores

5.3.1. Temporal Classifications of Data

Database tables are classified based on the temporal nature of their information – (a) "perpetual," meaning current, (b) "effective-dated," meaning historical, and (c) "snapshot," or point-in-time/range-of-time. A snapshot table may exist for either a perpetual (point) or effective-dated table (range). Classifications are elaborated as follows:

- **Perpetual Table** – Has only the most current information related to a given data entity. Because any value in a perpetual table is "now," a perpetual table may not contain a date in the logical primary key. Replicas of perpetual tables may be updated asynchronously provided that the latency period is small.
- **Effective-Dated Table** - (Sometimes called a historical table) is a temporal audit trail of information, such as a transaction file. An effective-dated table must contain a date in the logical primary key. Replicas of effective-dated tables may be updated asynchronously provided that the latency period is small.
- **Snapshot Table** - Has the information related to a given data entity as of given points in time, which point(s) is/are typically branded into the table name. A snapshot of a perpetual table is as of a given date time or range of date times.

5.3.2. Types of Data Stores

There are several types of data stores that will be implemented across the VA enterprise. They are each distinct, are designed to meet different needs, and each should only be used for the purposes for which it was designed.

- **Online Transaction Processing (OLTP)** – Are databases, typically in Boyce Codd Normal Form (BCNF), which are optimized for transaction processing, primarily update but some query IF it supports update. OLTP data stores may be either perpetual or effective-dated, but never snapshots. Effective-dated tables will contain only enough historical information to meet the OLTP workload.
- **Operational Data Stores (ODS)** – Are a BCNF version of OLTP data which serves as a staging area for information to be loaded into a data warehouse and/or DMs. The ODS is a replica of the perpetual and effective-dated OLTP. The ODS may contain enough effective-dated information to meet reporting and analytical needs, which may be considerably more data than at the OLTP sites. ODSs do not contain snapshot information.

- **Enterprise Data Warehouse (EDW)** – Is the system that provides access to snapshots (daily, weekly, etc.) of enterprise information in an integrated format in BCNF. EDW data stores are strictly snapshot in nature. The EDW does not contain perpetual or effective-dated information. The EDW may use effective-dated information to apply updates to its snapshots. The EDW is an amalgam of data from ODSs, integrated and harmonized across domains wherever necessary. The EDW shall be maintained in BCNF. The EDW may be a single physical entity – a physical EDW that houses all of the extracted information in one location or it can be a logical construct – composed of a number of physical data warehouses that while they are harmonized are separate data stores, each in a different part of the organization. In this EAA no distinction will be made between the two possible EDW constructs and the single term EDW will be used.
- **Data Marts (DM)** – Are systems that provide query access to snapshots of information that have been summarized for specific purposes. DMs are sometimes called "subject" databases. If DM data stores cross domains, the source information must be drawn from the EDW. DM data stores contain the same temporal mix of information as the EDW. As used here, the term "DM" is a term of art whose meaning is only as defined in this section. Any subject area models that alter the physical format of the BCNF data are considered DMs. These DMs are typically implemented on a star schema. DMs may be implemented at the EDW or within the domains. Any process that extracts data from the EDW will be considered to be a DM. This specifically includes extracts that are used to support high volume query requirements as well as management reporting requirements. As used here the term "DM" refers to two types of data stores, each used for a distinct purpose. These are:
    1. **Online Analytic Processing and Reporting Data Stores** – These data stores are relatively static, are updated on a regularly scheduled basis (typically once a day or once a week), and are used to support generation of complex analyses and regular, point-in-time management reports.
    2. **High Performance Specialized Query Data Stores** – These very specialized databases are designed to meet specific high volume queries. They are specialized subsets of a large more complex data store, may be distributed across VA, and are used to support VA operational query requirements. The High Performance Specialized Query Data Stores correspond to "materialized views" in which the query result is cached as a concrete table that may be updated from the original base tables. This enables much more efficient access and is most useful where frequent queries of the actual base tables are extremely expensive. In addition, because the view is manifested as a real table, anything that can be done to a real table can be done to it, most importantly building indexes on any column, enabling drastic speedups in query time.

5.4. Building Data Warehouses and Data Marts

Application Domains (Domains) are a collection of application systems that use a common or related set of information and are closely related. There may be multiple domains in a single VA Business Area or MI. The EDW is the place where cross domain information is integrated. The EDW is a snapshot integrated version drawn from the ODSs. From there data can be distributed to DMs, which are tailored for access by decision support users. While the EDW is designed to manage the bulk supply of data from its suppliers (e.g. ODSs), and to handle the organization and storage of this data,

DMs focus on packaging and presenting selections of the data to end-users, to meet specific management information or high volume query needs.

All DMs are completely independent of one another. They have independent feeds of update information from the EDW. They may be fielded on separate IT resources and their ETL feeds do not compete for machine resources.

The definition and operation of the EDW and each of the DMs will be based on the following set of organizational responsibilities:

- Domains are responsible for operating VA OLTP systems as required to meet their mission. OLTP systems will:
    - Be managed by the individual domains
    - Incorporate all business logic
    - Perform all updates to COR information (i.e., the authoritative instance of information)
    - Are the basic operational systems
    - The operation of these OLTP systems is not part of these principles.
- The domains are responsible for replicating in an ODS all of their OLTP data stores in BCNF using VA standard data definitions. This ETL function should occur within a very short amount of time – with less than 10 minutes latency being typical. The information in the ODS will be available to the EDW and to DMs. The domain can use publish / subscribe mechanisms to move data from the ODS to the EDW/DMs or the EDW/DMs may elect pull data from the ODS.
- The content of the domain's ODS will be based on a governance agreement with the enterprise data management group. The governance agreement will constitute a LDM in the form of an Entity-Relationship Diagram (ERD) for the domain's portion of the enterprise data.
- The organization(s) responsible for building, deploying, and operating the EDW will assimilate from the various ODSs the domains' contributions to the EDW.
- The domains are responsible for extracting BCNF data from the EDW to form DMs to meet their separate reporting missions and high performance query processing. The domains may extract and summarize data from the EDW to form DMs in accordance with their own rules provided:
    - Each "*unique summary*" (subset of the facts and / set of business rules) must have a unique name

- Any summary structures and their associated names are coordinated with enterprise data management group in its governance role.

- The domain will extract, align and cleanse the data in its ODS as necessary, based on enterprise data management group's guidance, prior to making the data available to the EDW

- Facts extracted from an OLTP database will be staged in an ODS database prior to being loaded into the EDW

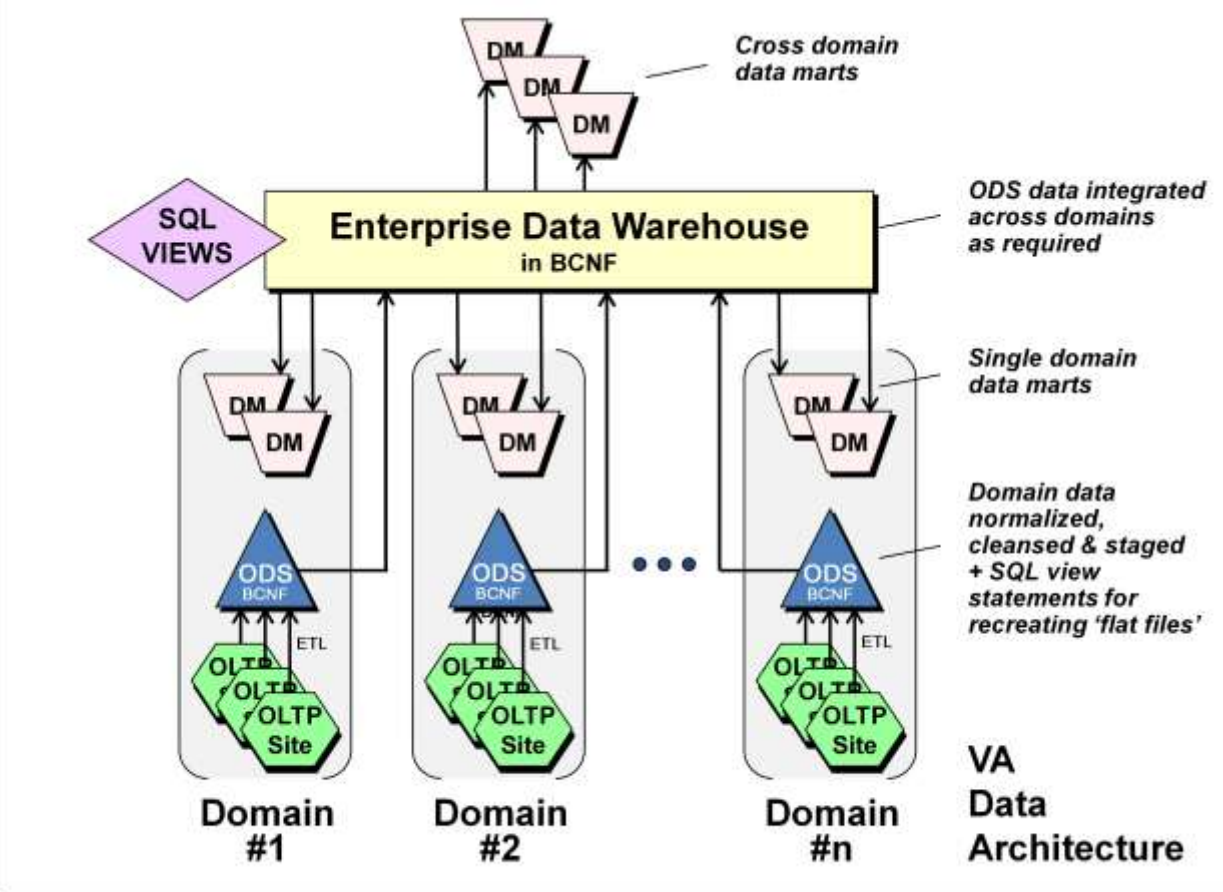

Figure 27 -- Building the Enterprise Data Warehouse and Data Marts

- Any VA organization may develop cross-domain DMs by extracting data from the EDW as long as it observes the naming conventions for summary structures.

- The domain will be responsible for creating and maintaining an asynchronous replica of all data in one or more ODS. The ODSs will conform to the approved LDM. The domain will be responsible for transforming OLTP information from the PDM into the governed LDM format. The transformation will include both normalizing the data and cleansing

the data (e.g., converting local data values to the standard codes etc.). Normalization will include transforming the data to BCNF.

- Any subject area models that alter the physical format of the BCNF data are considered DMs. These DMs are typically implemented on a star schema. DMs may be implemented at the EDW or within the domains. Any process that extracts data from the EDW will be considered to be a DM. This specifically includes extracts that are used to support high volume query requirements as well as management reporting

However, there may be some very limited number of very high priority data feeds that are needed throughout VA or a portion of the VA in as timely a matter as possible. These specific data feeds will need to request a waiver from the above process. For those data feeds, the data will be distributed to all users directly without their waiting for the data being loaded into to the EDW and then extracted from the EDW and sent to the DMs. These exceptional data feeds will be identified by specific waivers. These specified data feeds would then be provided to all DMs that required the data, regardless of the domain in which they reside. This direct path to the DMs would only include data feeds that had been extracted to the ODMS and sent to the EDW.

In the target state, OLTP data will be used and maintained in the form and normalized to the degree that makes the most sense for a given production application. Full BCNF normalization is the preferred format for OLTP data. However, performance is a key factor in OLTP data repositories and often de-normalizations are required to meet performance goals.

Each domain will also use ETL toolset(s) to maintain on one or more RDBMS platforms an ODS. The ODS data will be in BCNF format based on a domain LDM that has been approved by EDME and maintained by the EDME data governance group. The domain will perform the necessary ETL operations to asynchronously replicate changes from the OLTP's data repositories (in the project or domain PDMs) to the ODS's LDM format.

This will take the form of transactional replication and synchronization will be maintained within a ten-minute latency window. The domain will trickle-feed the changes to the ODS, and make those change feeds available to all authorized subscribers, including but not limited to the EDW and any DMs that reasonably require the information. This is shown in Figure 21.

The EDW may either ping the ODSs to obtain domain information or arrange to receive trickle feeds, or both. The EDW shall have access to all levels of information placed in the ODS regardless of security, privacy, or source restrictions, but will be responsible for enforcing and / or promulgating any security, privacy, or source restrictions and assuring that any recipients of the data or DMs that are created.

The EDW will be responsible for integrating BCNF information across domains, to the extent that such integration may be necessary. Any changes to the domain LDMs that affect the EDW will be coordinated with the enterprise data management group. The enterprise data management group will maintain a site with EDW LDM and PDM information, only to the extent that such metadata differs from the ODS metadata. In other words, it is not necessary or advisable for COR metadata to be stored more than once.

DMs may be created by any domain for valid purposes, provided that information is drawn from the ODS or the EDW. DMs may store information in whatever format is most appropriate to the given

purpose. DMs are responsible for ensuring that all of the information that they disseminate is in accordance with the EDME approved copies of record and metadata information.

5.5. Information Sharing

Sharing of information is an increasingly important element of the VA mission. It is imperative to effectively exchange information across VA and with DoD Military Health Services (MHS). The Information Sharing Strategy establishes the vision and goals for information sharing, while paving the way for more detailed initiatives to document the implementing actions necessary to achieve these goals and realize the vision.

Information sharing is defined as, "making information available to participants (people, processes, or systems)." Information sharing includes the cultural, managerial, and technical behaviors by which one participant leverages information held or created by another participant. The vision for information sharing is to deliver the power of information to ensure mission success through an agile enterprise.

The future state is one in which transparent, open, agile, timely and relevant information sharing occurs. There are several actions that the needs to take to move from its current state to full implementation of the information sharing to the vision. The information sharing goals form an environment that will:

- **Promote, encourage, and motivate sharing –** Successful information sharing necessitates a mindset where information is continually shared as a normal course of work.
- **Achieve an extended enterprise –** The extended enterprise refers to all internal and external participants required to ensure mission success.
- **Strengthen agility, in order to accommodate unanticipated partners and events –** Though it is important to proactively plan for information sharing with anticipated partners and events, it is also critical to prepare for unanticipated partners and events.
- **Ensure trust across organizations –** A cornerstone of information sharing is trust – trust in the partner organizations including, but not limited to, their policies, procedures, systems, networks, and data.

The Information Sharing Environment is based on the concept of a Shared Space as a separate area to be used by participating entities – MIs, projects, applications or VA and partners etc., to place shareable services and data in a manner that ensures appropriate security. At a conceptual level, this definition gives VA the opportunity to further articulate the mechanics of its how data and services are shared.

The Shared Space may be both logical and physical. It can be physical in the sense that there may be a physical area in which copies of data assets are stored – for example for sharing of data with the MHS such as at North Chicago. It can be logical in the sense that the shared space does not require that physical data assets be placed in the shared space. The shared space may just contain a repository of services that can be used to access data assets stored in various locations. For example, with iEHR some data may be stored in VA repositories and some in DoD repositories, but access may need to be provided to both VA and DoD systems.

5.6. Data Artifacts Related to Application Development

The EDA contains a number of artifacts that are of interest of the application developers.

5.6.1. Data Asset Catalog

The purpose of the Data Asset Catalog is to store information about the Enterprise Data Assets used to accomplish the mission of the VA. It is essential to the mission of information sharing that the description of the Enterprise Data Assets be comprehensive, not only to support the discovery of data contained within data assets, but to facilitate the information sharing in support of the mission. As the Data Architecture matures, the elements that make up the data asset descriptions within the Data Asset Catalog will continue to be refined, expanded and captured. The catalog draws on descriptive information attributes to balance access and security issues such as privacy, data quality, authoritative source indicators and temporal and geographic coverage, as well as management and stewardship responsibilities. The descriptive information enables efficient access control to mission data, service or process. This balances the protection of the data asset with the dissemination of the information to mission users. A key element in effective information sharing is the traceability mapping and accountability of information assurance of the Data Asset Catalog information.

The Data Asset Catalog will include information each of the databases in VA. The data asset catalog will include and identification for each database, the data that is stored in the data base, the points of contact for the databases and other metadata about the data elements in the data base such as security and privacy restrictions on the data and the source of the data.

FEA DRM defines a data asset as a managed container for data. Examples include a relational database, a Web site, a document repository, or a data service. The purpose of the Data Asset Catalog is to store information about the enterprise data assets used to accomplish the mission of the VA. It is essential to the mission of information sharing that the description of the enterprise data assets be comprehensive, not only to support the discovery of data assets, but also to facilitate information sharing in support of the mission.

The catalog draws on descriptive information attributes to balance access and security issues such as privacy, data quality, authoritative source indicators and temporal and geographic coverage, as well as management and stewardship responsibilities. The descriptive information enables efficient access control to mission data, service, or process.

5.6.2. Information Exchange Matrix

The Information Exchange Matrix (IEM) details information exchanges and identifies “who exchanges what information, with whom, why the information is necessary, and how the information exchange must occur.” Information exchanges express data relationships with a focus on the specific aspects of the information flow and the information content.

An IEM may be used to represent the following concepts.

- **Information exchange –** Information exchange is an act of exchanging information between two distinct operational nodes and the characteristics of the act, including the information element that needs to be exchanged and the attributes associated with the information element (e.g., Scope), as well as attributes associated with the exchange.

- **Information element –** Information element is a formalized representation of information subject to an operational process (e.g., the information content that is required to be exchanged).

5.6.3. Data Models

Enterprise Data Models are products of the Enterprise Data Architecture and include a conceptual and logical representation of real-world objects and events that represent the inherent properties of data independently of hardware, software or machine performance considerations. There are three levels of data models that are developed – the conceptual, logical, and physical models. These range from the most abstract to the most physical. The conceptual and logical models are produced at the project, MI / Business Area, and enterprise level. The physical model is not normally produced at the enterprise level and there are no plans to produce a VA Enterprise Physical Data Model (PDM).

5.6.3.1. Enterprise Conceptual Data Model (ECDM)

The ECDM represents the overall structure of the enterprise information, independent of any software or data storage consideration. The ECDM describes the data and information that supports program and business operations. It contains data objects and a grouping of related data objects called subject areas and gives a formal representation of the data needed to run the enterprise. The subject areas and data objects are used to categorize and describe enterprise data, thus supporting the discovery of enterprise data and information sharing products based on the business functions.

This is a very high-level model that shows subject area taxonomies, stewardship assignments, policies and business rules, sources of record and sources of reference.

5.6.3.2. Enterprise Logical Data Model (ELDM)

The ELDM is a further description of the data that supports the implementation of business processes. It identifies objects, attributes, relationships, cardinality and keys. The ELDMprovides a more detailed view of the data needed to run the enterprise.

The ELDM is a high-level model detailing subject areas and super types from the ECDM. The ELDM shows semantic definitions and relationships. It includes subject areas, entities and their attributes.

An entity is a thing of significance about which the organization wishes to hold information. The model shows data attributes grouped into fully normalized entities and the relationships among these entities. One or more data elements or attributes describe an entity, and the values of those elements describe individual instances of the entity. An attribute defines one of the pieces of information that is included in an entity. The data elements are the instantiation of those attributes in the physical data implementation. The ELDM is a logical representation of real-world objects and events that represent the inherent properties of data independently of hardware, software or machine performance considerations. It includes subject areas, entities and their attributes. The model shows data attributes grouped into fully normalized entities and the relationships among these entities. The data model also includes the specification of valid values for reference data such as country code tables. To use a common analogy, the data model is equivalent to an architect's building plans.

5.6.3.3. Physical Data Model

A Physical Data Model (PDM) is a representation of a data design that takes into account the facilities and constraints of a given database management system. Typically, it is derived from the Logical Data Model (LDM) and may include all the database artifacts required to create relationships among tables or achieve performance goals, such as indexes, constraint definitions, linking tables,

partitioned tables, or clusters. While the LDM drives the development of the PDM, there is not a one-to-one correspondence between the LDM and the PDM as the physical tables need to be designed and implemented to ensure performance goals. Thus, a single entity in the LDM may be implemented as one of more tables in the PDM and there may be a number of the table included in the PDM to support performance or for housekeeping or other internal functions that do not appear in the LDM.

The PDM is a data base implementation model and is not normally captured at the Enterprise Level and there are no plans to develop a VA Enterprise PDM. The PDM will be a major artifact of the project data architecture and will be developed by each application development project.

5.6.4. Data Harmonization

A major factor underlying the development and successful use of this EAA is the harmonization of data across VA. Data harmonization requires that the syntax, semantics, and metadata agree for data elements across VA. Data Harmonization across VA is an important factor in the development and operation of this EAA. Common data element definitions will be critical to the development of both core common services and services that are able to be reused across VA as common data definitions will be a prerequisite to the use of common services and the fully harmonized data definitions are the basis for VA Standard Messages.

Data harmonization needs to include several types of harmonization including:

- Semantic Harmonization
- Syntactic Harmonization
- Meta Data Harmonization

Each of the above harmonizations will need to be performed. There is a fourth harmonization that is possible, format harmonization, but this will not be require to be performed *a priori* because the ESB will perform format transformations.

Data harmonization is based on the ELDM that identifies all of the enterprise data that will be of interest across the VA enterprise. Much of the work performed in constructing the ELDM will be the semantic harmonization of data to define data entities and attributes (data elements) that can be across VA. Ideally, data would be harmonized across both VA and the MHS, but this cannot occur until there is harmonization of data across VA.

5.6.4.1. Semantic Harmonization

Semantic harmonization of data ensures that data elements with the same name have the same meaning and that data elements that have different names have different meanings. As systems choose data element names there is a high probability that the same name will be used for different quantities. Systems that use the same name for different quantities cannot communicate unless and until they can agree on the meaning of the data that they exchange.

For example, one may ask how many patients are on a floor of a particular VAMC. Admitting might respond with the number of persons admitted and assigned to the floor, whether they had reached the floor or not; the nurses might respond with the number of people physically present and not count those that are off the floor for tests or surgery; housekeeping and food service might give yet a different answer. It is not that any of these answers is wrong, they are just answers to different

questions and if data is to be shared across the enterprise, each must be represented by data elements with different names.

One approach to assuring unique data names that are used internal to a variety of systems is for the names in each system to exist in a separate namespace. In general, a namespace is a container for a set of identifiers (names), and allows the disambiguation of homonym identifiers residing in different namespaces. Namespaces usually group names based on their functionality. Namespaces can be confined to a single system or a small group of systems, but data names that are exposed to multiple systems must be in a namespace accessible to each of the systems and therefore must be harmonized across those systems. Therefore, the use of distinct namespaces does not resolve the issue of the need for common data element names for data elements or messages that will be used by systems across the VA.

5.6.4.2. Syntactic Harmonization

Syntactic harmonization refers to the form of the data. Data must not just have the same meaning, but the meta data related to it must also match. For example, are names sent as <FIRST, MI, LAST> or <LAST, FIRST, MI> etc.? However, because XML is self-describing it may seem able to resolve the syntactic harmonization issues, but 1) the high cost of transforming data to and from XML makes its use problematical for sending data between systems problematical and 2) it does not resolve all of the syntactic harmonization issues – i.e., are there limits on the number of characters in a first or last name? To be able to share data, the data validation rules must be the same across the enterprise so that data that is entered into one system will not fail validation checks in another. In addition, other aspects of the data elements must match, such as allowable values and validation rules must also match

Since the OIT VA Enterprise Management Principles and Application Architecture require that there be a single authoritative instance for all data and no updates of copies of the authoritative instance are valid until the authoritative instance is updated. But, what this also required is that once one system accepts a data update it will pass all data edits and validations for the authoritative instance of the data. But the data once validated is validated, it must be able to pass the data validations of all systems that use that data.

5.6.4.3. Format Harmonization

Format refers to the physical form of the data – i.e., whether a number is represented in binary format, as ASCII character representing numbers, or an XML string with the value (i.e., <VARIABLE NAME = “data string”>. Depending upon the nature of a service, the format of the messages may vary. Web services will use SOAP over HTTP for message transport which will require XML based messages, communications between VA systems may be through a commercial messaging product and be for example JMS over IBM WebSphere MQ, or between closely related services messages may just be binary data sent over a TCP/IP link. The ESB will be able to perform message transport and message format transformations, but is the ESB is not used for communication between services than both sender and receiver will need to use the same transport and message format.

5.6.4.4. Metadata Harmonization

Meta data harmonization means that there can only be one set of metadata for any data element. Metadata includes information about a data element, such as timeliness or accuracy requirements, retention requirements, restrictions on access or use and location of the authoritative instance / COR

etc. It is all of the information needed to manage the data element and must be consistent across the VA enterprise.

5.6.5. Common Glossary, Vocabulary, or Lexicon

A glossary, vocabulary or lexicon is a list of words used in a specialized field with their associated definitions. The term glossary is used throughout VA and a large number of glossaries exist within VA including one for EA. Because of the large number of glossaries that are used across VA, the term lexicon will be used in this section to avoid confusion with the numerous glossaries that are in use.

Developing and using a common vocabulary is critical to effective and efficient information sharing. Without a common understanding of the words used to define data, their definitions, meanings and context, information sharing between two or more parties will be difficult, expensive and prone to errors that drive up mission risks. VA Lexicon mechanism for supporting a common vocabulary across the enterprise, and is defined as follows:

"A repository of organized terms (acronyms, words, compound words, and phases) providing a single, explicit and context driven definition for each word or phrase and, if necessary, supporting annotations."

The Lexicon will be used as the controlled vocabulary for describing data assets in the Data Asset Catalog and artifacts that are contained in the Information Exchange Repository in order to support information discovery.

Efforts to develop a common vocabulary will go far beyond the requirements for the standardization needed to harmonize the project LDM into an Enterprise Logical Data Warehouse. The development of the VA common vocabulary and common lexicon will be needed to support the data warehouse efforts. The development of the a data warehouse and data marts, where each "*fact*" that is extracted from the OLTP systems and each unique summary that is derived from the data warehouse will need unique names, will drive the need for the common lexicon and vocabulary.

This harmonization will occur through the development and definition of a VA Information Taxonomy and VA Information Ontology. A taxonomy is a hierarchical list of information while the ontology provides a much richer set of information about the data elements, and therefore will be more challenging to develop.

The Lexicon provides a common set of definitions for terms that are used across VA. While the definition of a standard set of terms is important, that set of terms will not be broad enough or rich enough to support the harmonization of all VA enterprise data elements. While VA lexicon will include standard terms and their definition – but the terms will be at much too high a level to differentiate between the variations in definitions that will be necessary to support the development of unique data element names across VA. Further, a glossary or lexicon is typically an alphabetical list of terms that does not provide any notion of structure or the hierarchy of data elements.

5.6.6. VA Information Taxonomy

Taxonomy refers to the classification of things arranged in a hierarchical structure or classification scheme. Typically this is organized by supertype-subtype relationships, also called generalization-specialization relationships, or less formally, parent-child relationships, typically indicated by the phrase 'is a kind of' or 'is a subtype of'.

In terms of data harmonization, the information taxonomy is important because it provides a hierarchical structure to the data. For example, many VA may store numerous addresses for a Veteran. Whereas a glossary or lexicon could list the definition of each, the taxonomy includes the hierarchical relationships between them. The taxonomy provides a mechanism for organizing data and finding the information that is required.

The taxonomy contains more information than the glossary or lexicon, but only a limited amount of additional information. If other kinds of relationships between concepts are also included, a taxonomy is extended into an ontology. An ontology includes a taxonomy.

5.6.7. VA Information Ontology

The data described by an ontology is interpreted as a set of "individuals" and a set of "property assertions" which relate these individuals to each other. An ontology consists of a set of axioms which place constraints on sets of individuals (called "classes") and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided.

An ontology is much more than just a list of data elements, even a hierarchical set of data elements. The additional information is not only “meta” data in the traditional sense, but rather data about the structure of the data elements, their relationships, constraints on the values that they can take and information that can be implied about the data elements.

Common components of ontologies include definition of:

- **Individuals** – Instances or objects (the basic or "ground level" objects).
- **Classes** – Sets, collections, concepts, classes in programming, types of objects, or kinds of things.
- **Attributes** – Aspects, properties, features, characteristics, or parameters that objects (and classes) can have.
- **Relations** – Ways in which classes and individuals can be related to one another.
- **Function terms** – Complex structures formed from certain relations that can be used in place of an individual term in a statement.
- **Restrictions** – Formally stated descriptions of what must be true in order for some assertion to be accepted as input.
- **Rules** – Statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.
- **Axioms** – Assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.
- **Events** – The changing of attributes or relations.

The ontology takes the elements identified in the taxonomy and provides information as to the relationships between them and how they interact. Clearly, developing a taxonomy is a first step

towards the development of the ontology, but the ontology augments the taxonomy with much additional information – both metadata about the data elements, the structure of the data elements, and the other information concerning data relationships.

The ontology represents meaning as a series of relationships between entities and constraints on the entities. This is similar to the relationship data in an Entity Relationship (ER) diagram that is at the core of the ELDM, but the ontology usually expresses a much larger number of relationships than would be in the ER diagram. Also, the ontology does not contain the structural information contained in the physical data model.

The ontology provides the basis for the harmonization of VA data by providing semantic information about each of the data elements. However, it is unlikely that full ontological information will exist about VA data elements prior to the completion of the ELDM and therefore, the process of harmonizing VA data can be used to support the development of ontology for VA data.

6. Application Design Considerations

This section describes a number of aspects of application design and development.

6.1. Portfolio Management

In a stovepipe environment each VA element – VA business area, MI, or project etc. would make the decision as to what systems or services it would develop based on its own priorities and resources. As the OneVA EA becomes more and more a part of VA IT planning, this SOA places much higher emphasis on common business and infrastructure services, and the Capital Planning and Investment Control (CPIC) process pushes VA more towards a portfolio management approach to systems development. The VA portfolio management processes as they mature will be the base for the management of SOA services across the VA enterprise and deciding which enterprise services will be developed and who will develop them.

The definition of the CCBS and the CCIS, their mandatory on all projects, and the general requirement for the reuse of existing services rather than the development of new services brings a new focus to portfolio management. Projects will not be free to specify, design and develop the services that they desire, but are required to use enterprise services (i.e., the CCBS and CCIS) or other previously developed services.

There will need to be coordination between projects to ensure that related projects cooperate in the identification of services that they require, so that service requirements result in a smaller number of more generally useful services than a larger number of single use services. Relatively general purpose, or multi-use, services should be developed as a series of fine grained services so that minor customizations to allow the services to meet the requirements of each project can be achieved through variations in the orchestrations rather than customizations of the code. The design and development of services to be used by multiple systems will require a certain amount of cross project coordination. The closer two projects are, the more likely they will be able to share services.

6.2. Design for 24x7x365 Operation

ATruly mission critical applications are required to be designed to operate on a 24x7x365 basis with 99.999% (five 9s) availability – essentially zero (0) planned and unplanned downtime. VA IT principles summarized at the beginning of this document specify that VA solutions must be designed to meet one of three (3) levels availability. These guidelines specify that mission critical systems

provide 99.999% availability including all scheduled and unscheduled downtime per year. The design of the system shall be such that there is no single point of failure within a site or within the network. This will require that each system will be designed with multiple strings of equipment at each of a minimum of two sites. The use of multiple strings at each site ensures that there is multiple of each resource at each site so that no single component failure can cause the system to fail at the site. The requirement for the system to operate at two sites ensures that the system can continue to operate and to be available in the event that one of the two sites is destroyed or otherwise becomes unavailable. This in turn will require that current data be maintained at each of the two sites to allow operations at one or both sites at all times. Further, there must be at least two, completely independent, routes through both the local and wide area networks so as to assure that at least one of the two sites is accessible at all times. For wide area networks, two network vendors should be used. The cost of providing two fully redundant systems and each of two sites with at least quasi-real-time data update and redundant networks will be very expensive to provide. Because of the cost and the need to upgrade the VA infrastructure to provide such high availability, requests for such a high level of availability should only be made based on verified functional requirements. The technology infrastructure required use multiple strings of equipment within a site or the use of systems at multiple sites are not the subject of this ADA. The discussion below is intended to provide background for application developers as well as to make them aware of the need to allow for this requirement in their application designs.

6.2.1. Modes of Operation

There are three major modes of operation that can be used to meet the high availability requirements – active – active and active – passive, and a hybrid approach that might best be termed active – "quasi" active. In the active – active mode of operation both sites are used to support and workload is balanced between the two sites. This approach allows both sites to process workload, but requires that data be updated at both sites concurrently with each site passing updates to the other. If it is possible that the same record would need to be updated at both sites at the same time this will be difficult as there would need to be a locking mechanism to operate across both sites – which will degrade performance markedly. With very high volume databases where the same record can be updated at both sites, either requires locks that operate over geographic distances and thus may have a serious performance impacts or sophisticated synchronization that may require significant program involvement and in many cases may not be feasible.

In an active – passive mode of operation, one site is the active site and processes all workload and the other is a passive site – one that is continuously updated so that it always has a current copy of the data, but one which is processing no workload. This approach is much simpler to implement, but results in half of the equipment being idle at all times.

The hybrid active – "quasi" active of the two approaches is to divide the workload so that all updates are processed at one of the two sites, but both sites can be used to process query workloads. This approach avoids the major issue of active – active, concurrent writes at the two sites and the major issue of the active – passive, half of the equipment sitting idle at all times. However, this approach is dependent upon being able to direct the update workloads to the "active" site.

The determination as to the approach to be used in any specific application design will be the prerogative of the application design team; but, the Architecture Principle of Lowest Total Overall Cost and 24x7x365 operation must be followed.

6.2.2. Data Synchronization across Sites

There are a several mechanisms that can provide more or less access to current copies of data at more than one location. The need for reasonably current copy of the data at multiple sites occurs in the active – "quasi" active approach to high availability as well as when an application requires a local copy of data for which it does not house the authoritative instance of that data. Where performance requires an application to maintain a local copy of data for which it does not hold the authoritative instance, there will need to be a mechanism for updating each of the remote copies. Each of these approaches has its advantages and disadvantages. The potential approaches are listed below roughly in order of the latency in the availability of data for use at the other site. It is the choice of the application designer as to which of the approaches is used for a given application.

- **Synchronous writes at each location** – This provides truly synchronous systems, as each write operation is performed at each sites and must complete before the write is considered to be complete at any site. Read operations can proceed independently at both sites as they both have complete, current copies of the data. This approach is not only the most expensive, but places what many consider to be an intolerable performance penalty on the system as the need for messages to transit both to and from each site. The transit time for the messages will be approximately 1 millisecond for every 90 miles of separation between two datacenters.
- **Quasi-synchronous writes at each location** – Data is written synchronously to an auxiliary installation close to the location at which the data is being written. This second location would need to be close enough to the primary installation to as to not impose significant latency, but far enough away that any disaster that destroys the first center does not destroy the auxiliary center. The auxiliary datacenter then writes the data asynchronously to the second datacenter. There only needs to be minimal capabilities at the auxiliary datacenter – some storage to buffer results while they are being transferred and sufficient intelligence to know that the data has been safely stored at the second site. This approach avoids the performance impacts of the previous approach, but introduces a minor amount of latency between the two sites – i.e., the amount of time that it takes for the remote write to be completed at the second site. Both of these approaches ensure that no data is lost in the event one of the sites fails. Implementing this approach requires that such an auxiliary installation for each of the two primary data centers.
- **Asynchronous writes between each site** – Data is written asynchronously to the remote sites. This approach is less costly than the previous two, but the sites can be out of synch by one or two writes. In the event of a failure, if a write had been performed to the primary database, but had not yet been sent to the second datacenter, the two sites will be slightly out of synch. Writes that have been sent will be received by the other datacenter, even if the sending datacenter is destroyed before the write message is received, but there is a small window in which a write will have been completed at one site and the update message has not yet left the site. This delay would likely be at most a few milliseconds. With asynchronous writes the remote site will always be one or two "writes" behind the site at which the original "write" is performed. This delay is likely to be in terms of milliseconds rather than seconds.

6.3. Disaster Recovery

VA has mission critical systems that store vast amounts of information that could not be recreated were it to ever be lost. VA has a disaster recovery plan and a disaster recovery site. However, as

applications move to the active-active mode of operations, there will be less need for a formal disaster recovery site as data will automatically be stored at two geographically remote sites. Both sites will have a complete set of the data for the application and sufficient resources to operate the application in the event of a disaster at one of the sites – or even just the unavailability of one of the sites.

Although the need for the formal disaster is reduced with active-active operations, there will still be a need for the traditional disaster recovery process, with backup copies of data being stored at a remote site and plans to instantiate the applications at a remote site. Even with active-active operations, backup copies of the data will be required even though they will not actually be used for disaster recovery. With active-active and the virtual real time transfer of database changes an error in one data base will quickly propagate to the other. The backup copy of the data will be required in the event that there are errors in the database and it needs to be restored to an earlier consistent state.

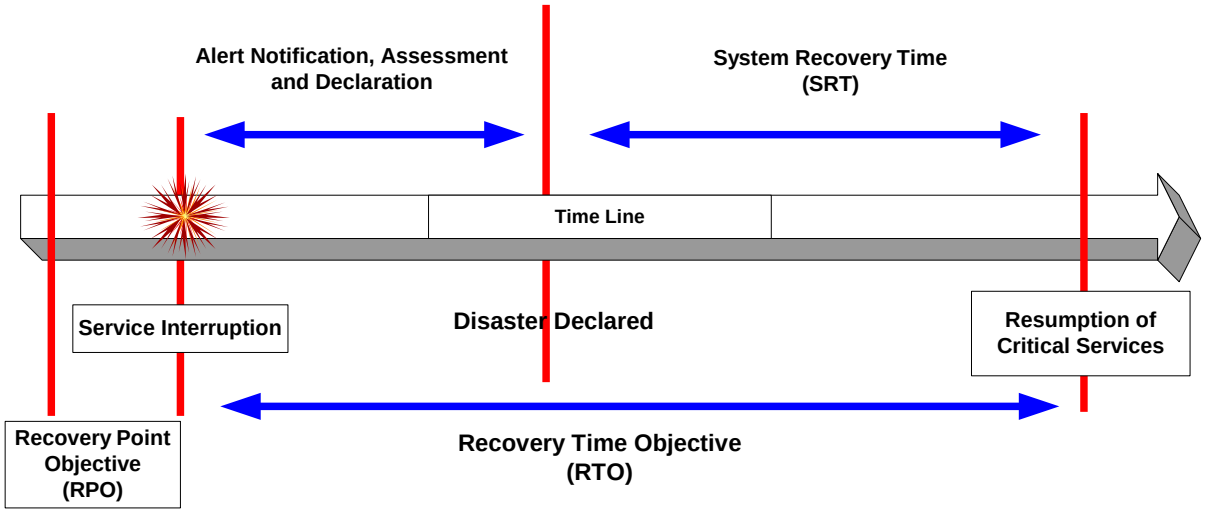

Figure 28 – Disaster Recovery Time Line

6.3.1. Basis for Recovery

Recovery time and data currency are key components in determining the level of service a business process requires in the event of a major disruption. These are quantified as the Recovery Point Objectives and the Recovery Time Objectives.

6.3.1.1. Recovery Point Objectives

The Recovery Point Objective (RPO) is defined as the point in time to which recovered data is restored. This is the maximum time before the outage to the point in time when the disaster occurred for which data will be recovered. The Recovery Point Objective is measured starting from the time of the disaster event backward to when the applications and data are preserved off site from the production-computing center, enabling applications and data to be restored for production functions on the recovery system. The copy or backup of data must be physically removed off site from the production-computing center in order for the backup operation to be considered complete. See **Figure 53** for a representation of RPO prior to the service interruption.

6.3.1.2. Recovery Time Objectives

The Recovery Time Objective (RTO) is the period following the major disruption within which application systems and data that support the business processes must be restored. RTO extends from the point when the disaster or service interruption occurs to the point when the recovery system is

operational and ready to resume production. RTO is a business requirement that, essentially, describes how long the business can tolerate being without the functionality of the application systems supporting its business process. As shown in **Figure 28**, the RTO incorporates the following two timeframes:

- The combination of Alert Notification, Damage Assessment, and Disaster Declaration timeframes.
- The System Recovery Time (SRT).

6.3.2. Disaster Recovery for Active – Active Systems

For VA mission critical systems for which there is a 24x7x365 operational requirement, the Recovery Point Objectives will be to the end of the last committed transaction for which a response has been sent to the user. With active architecture the database at the remote site will be updated whenever a transaction completes at the primary site. While this process will start at the time that the transaction is completed, it will take some number of milliseconds for the message to be sent to the second site and for the results to be sent to the user. Were failures to occur during this interval, the second site will not have been updated, and the user will not have been informed that the transaction had completed. In this instance the user will just resubmit the transaction to the second site, as he will have seen the first attempt to execute the transaction fail i.e., he did not get a response.

For VA mission critical systems for which there is a 24x7x365 operational requirement the RTO will be a matter of seconds – the amount of time to switch operations to the second site. With the active-active architecture, declaring one site unavailable and switching all work to the alternate site will not be a difficult process, but should not be done lightly as there is a process that will need to be followed and data will need to be recovered at the "failed" site. The Alert Notification will occur almost instantly – as soon as the systems management tools detect that the system is no longer responding. The System Recovery Time will be on the order of seconds – even if one of the sites has been running in passive mode it will only take seconds to make that system the primary.

The reason that a failover is not to be done lightly is that once a site is declared to be down, all database updates will be done at the second site and will not be replicated to the first site in real time. Therefore, once service is restored at the failed site the database will need to be recovered and brought to a current consistent state. The amount of time that this operation will take will depend on the amount of time that the failed site was not operating.

6.3.3. Disaster Recovery for Non-Active – Active Applications

Disaster Recovery for applications not operating in a 24x7x365 mode and that do not maintain a near real time copy of its data at two or more sites; will be performed as it has been at VA. A backup copy of the application's data will be generated and moved to an offsite location where operations would be restarted in the event of a disaster at the applications primary processing site.