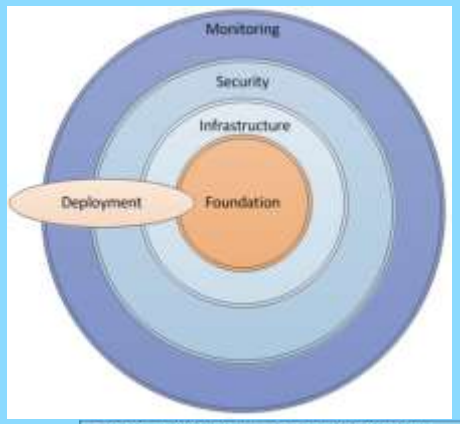
Digital Aviation
Reference Architecture

Guidance for Development Teams
Version 1.5

Sept 16, 2016

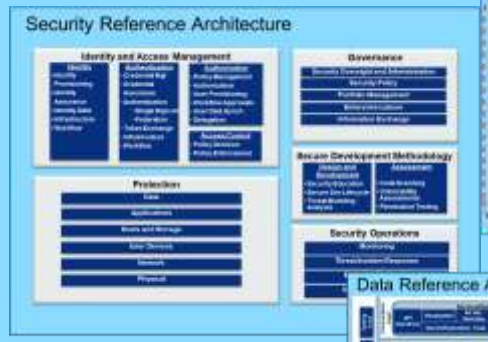
Digital Aviation Reference Architecture Guidance

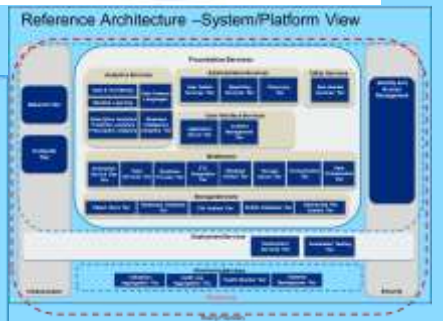
Purpose: "***Provide Digital Aviation development teams with Reference Architecture and Design Guidance that enables rapid, high quality, cost effective development, deployment and maintenance of software products.***"

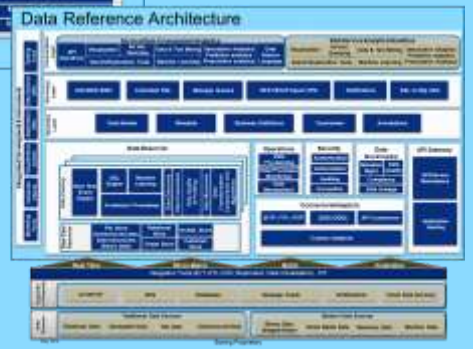


Digital Aviation Reference Architecture Guidance materials are organized into to the following areas:

- Enterprise Architecture
- System/Platform Architecture
- Data Architecture
- Security Architecture
- Mobile Architecture
- Common UI/UX
- Software Development Process (SAFe)
- Quality Assurance and Test Process
- Quality Attributes

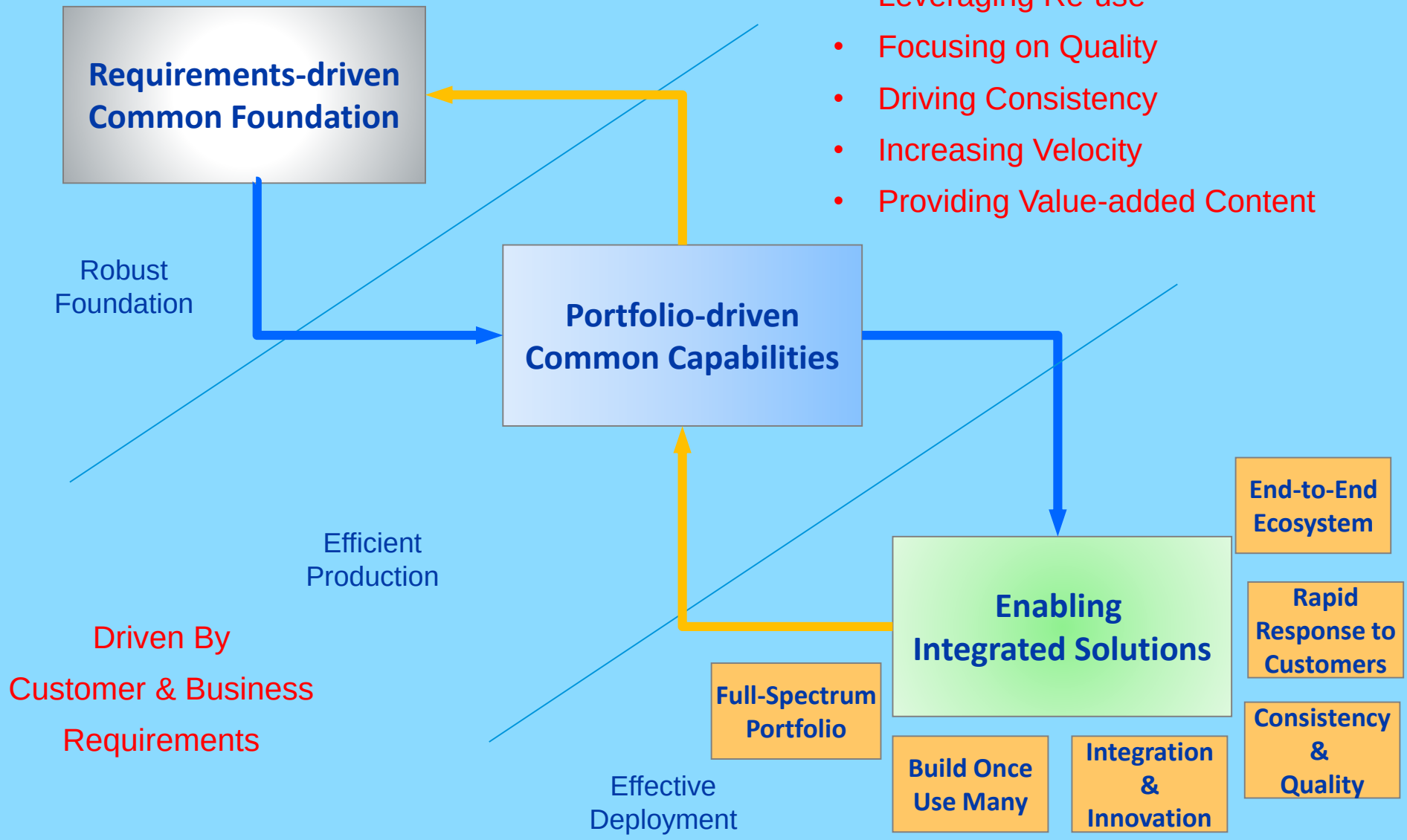






Reference Architecture is a Strategic Enabler…

- Leveraging Re-use
- Focusing on Quality
- Driving Consistency
- Increasing Velocity
- Providing Value-added Content



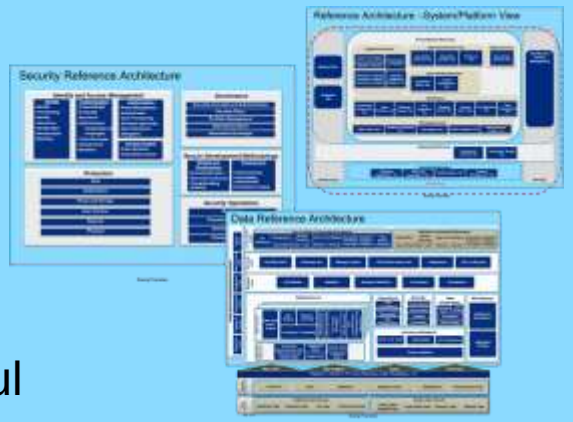
Reference Architecture – Approach to Common Services

Define/Specify Approach

- Reference Architecture provides guidance on **using common approaches** for design, development and deployment

Make Use of Platform Managed Services

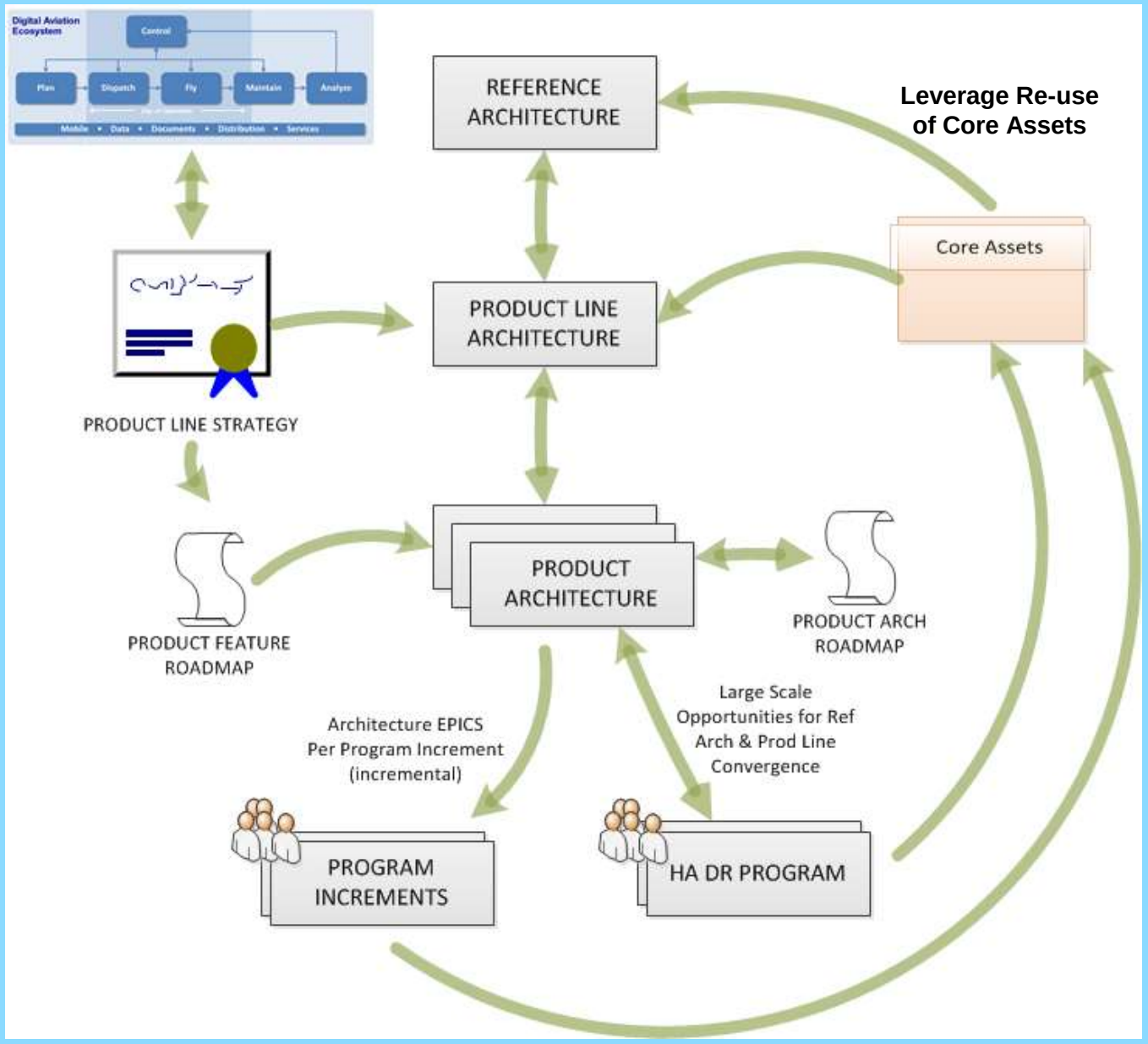
- Reference Architecture provides guidance on **equivalent services** within each platform that could meet project requirements



Leverage Capabilities from Digital Aviation Projects

- Reference Architecture will provide guidance on useful “Common” capabilities within existing Digital Aviation product that could be re-used in new projects
- Evaluate project requirements to determine if re-use of existing common components “as is” meets the need
- Determine if refactoring of an existing component can meet the need rather than initiating a new effort to create a similar component

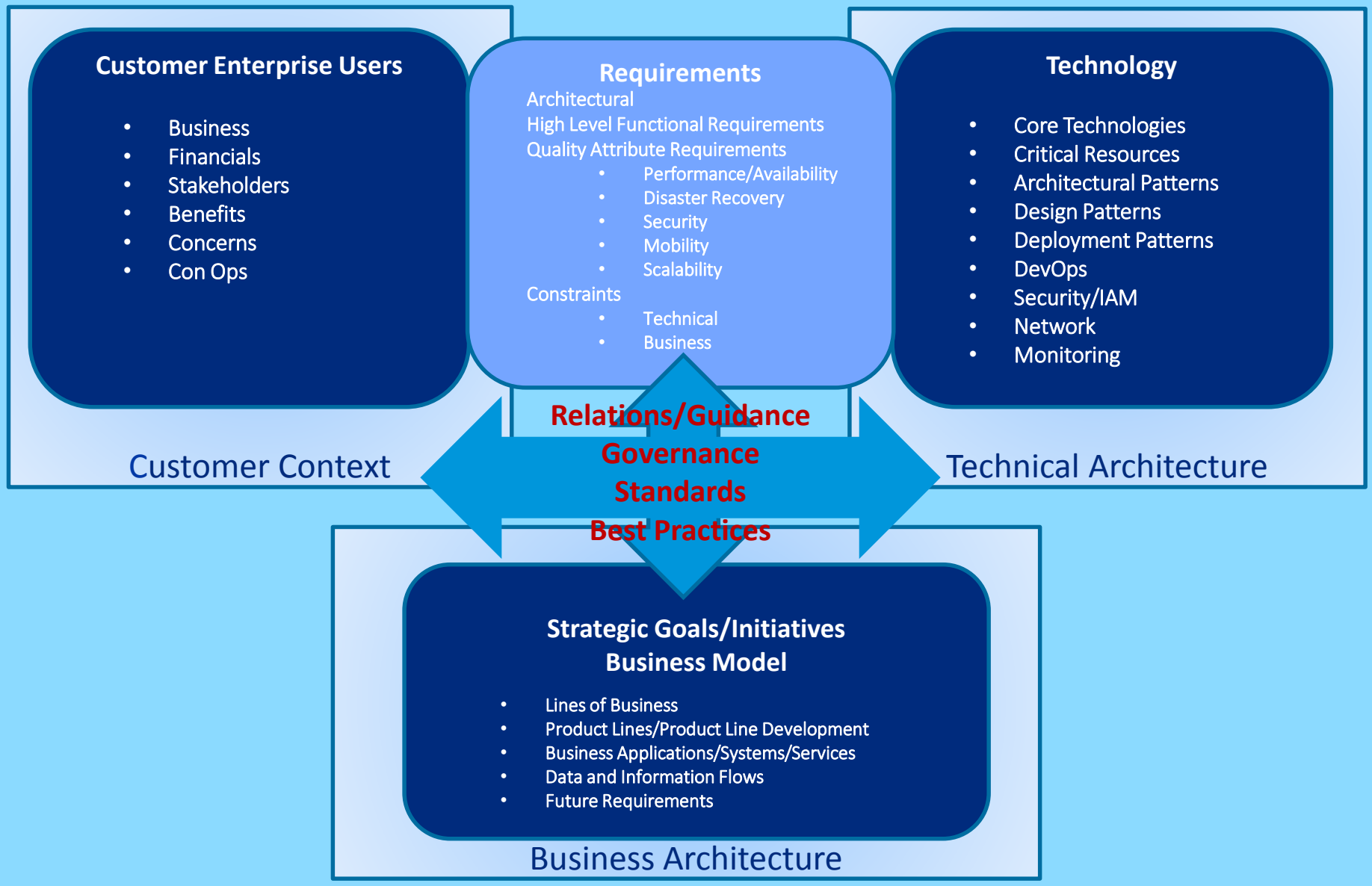
Synchronize with Product Architecture



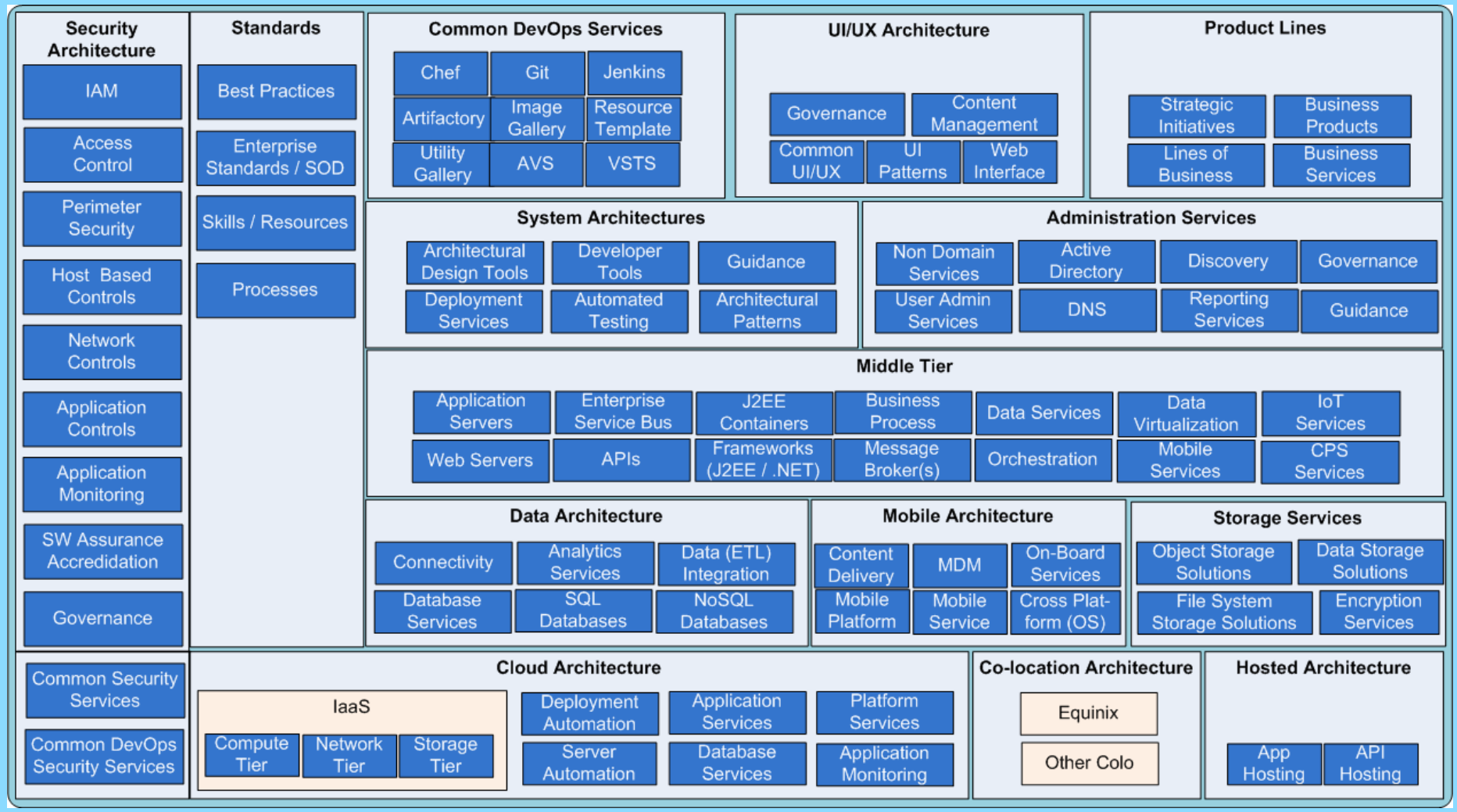
Enterprise Reference Architecture

Return

Enterprise Reference Arch – DA Organization View



Enterprise Reference Architecture Overview



Enterprise Reference Architecture Summary

Description

- Digital Aviation's Enterprise Reference Architecture is the sum of all architectural guidance that is formally provided to internal and external development teams building applications for the Digital Aviation portfolio.
- Where specific architecture decisions or system components have been identified in this document, it is expected that development teams will make use of those components or architecture unless a formal waiver has been granted by management.
  - High level concepts presented in architectural diagrams that cover structure and key non-functional requirements
  - Specific uses cases based on application deployment patterns to provide further clarity on how to interpret the guidance
  - Technical guidance organized/customized by platform (such as specifying a version of a specific component that is optimal for one platform vs. another)
  - Structures for communicating key architectural decisions made by the development teams for each product to empower efficient and appropriate governance.
  - Reference Architecture IS NOT a cookbook for designing or building applications. More detailed information will be available for actual design and development work.
  - Reference Architecture IS general design guidance on the key technical elements that are expected to be present in any application, new or refactored

Digital Aviation Reference Architecture Governance

- Digital Aviation's Enterprise Reference Architecture plays an important role in driving standardization and integration across the DA Portfolio of Products
- Ensuring that guidelines are understood and adhered to is a significant part of the Reference Architecture Organization's mandate
- New, refactored or re-hosted software products are expected to adhere to the guidance provided by the Reference Architecture Organization to the extent possible based on the nature of the project, the funding provided and the direction of Product and Technical Management.
- All Digital Aviation projects will be evaluated on their compliance with the published Reference Architecture based on the following criteria:

Score	Guidance
1	Architecture for product/product line is independently developed with little or no commonality or coordination with DA Reference Architecture
2	General DA Reference Architecture patterns were followed but more than 50 percent of components were at variance with DA Architecture Guidance
3	General DA Reference Architecture patterns were followed but more than 25 percent of components were at variance with DA Architecture Guidance
4	General DA Reference Architecture patterns were followed with only incidental (and approved) variance with DA Architecture Guidance
5	Architecture is in full alignment with published Reference Architecture guidance for Digital Aviation.

System Platform Reference Architecture

Return

System/Platform Reference Architecture Summary

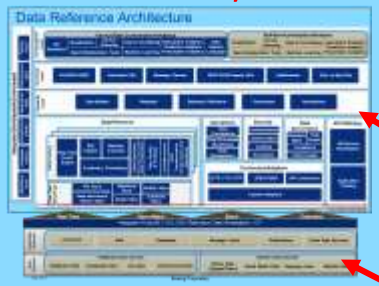
Description

- System/Platform Reference Architecture is comprised of the framework and components that are used to deploy Digital Aviation software products.
- The following views are provided for reference:
  - System/Deployment Overview
  - System/Deployment Details for Azure Deployment (Microsoft Cloud)
  - System/Deployment Details for AWS Deployment (Amazon Cloud)
  - System/Deployment Plans for HP Helion Openstack Deployment (Boeing Internal Cloud)

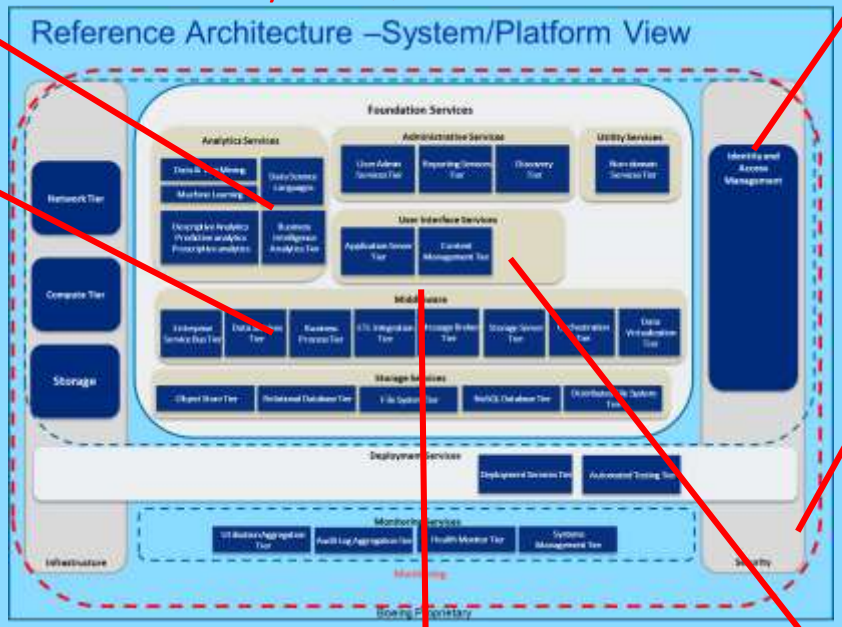
NOTE: There are additional onboarding guidance materials within this section for each platform, notably Azure. This material is dynamic and changes rapidly as new development efforts move forward. The Reference Architecture Team will periodically refresh this guidance material and work with the development teams to make them away of changes as they occur.

Reference Architecture Domains

Data/Analytics

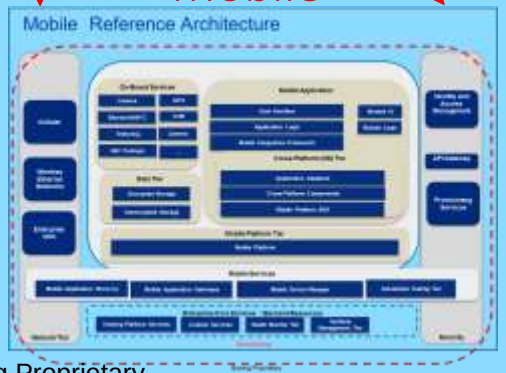


System/Platform

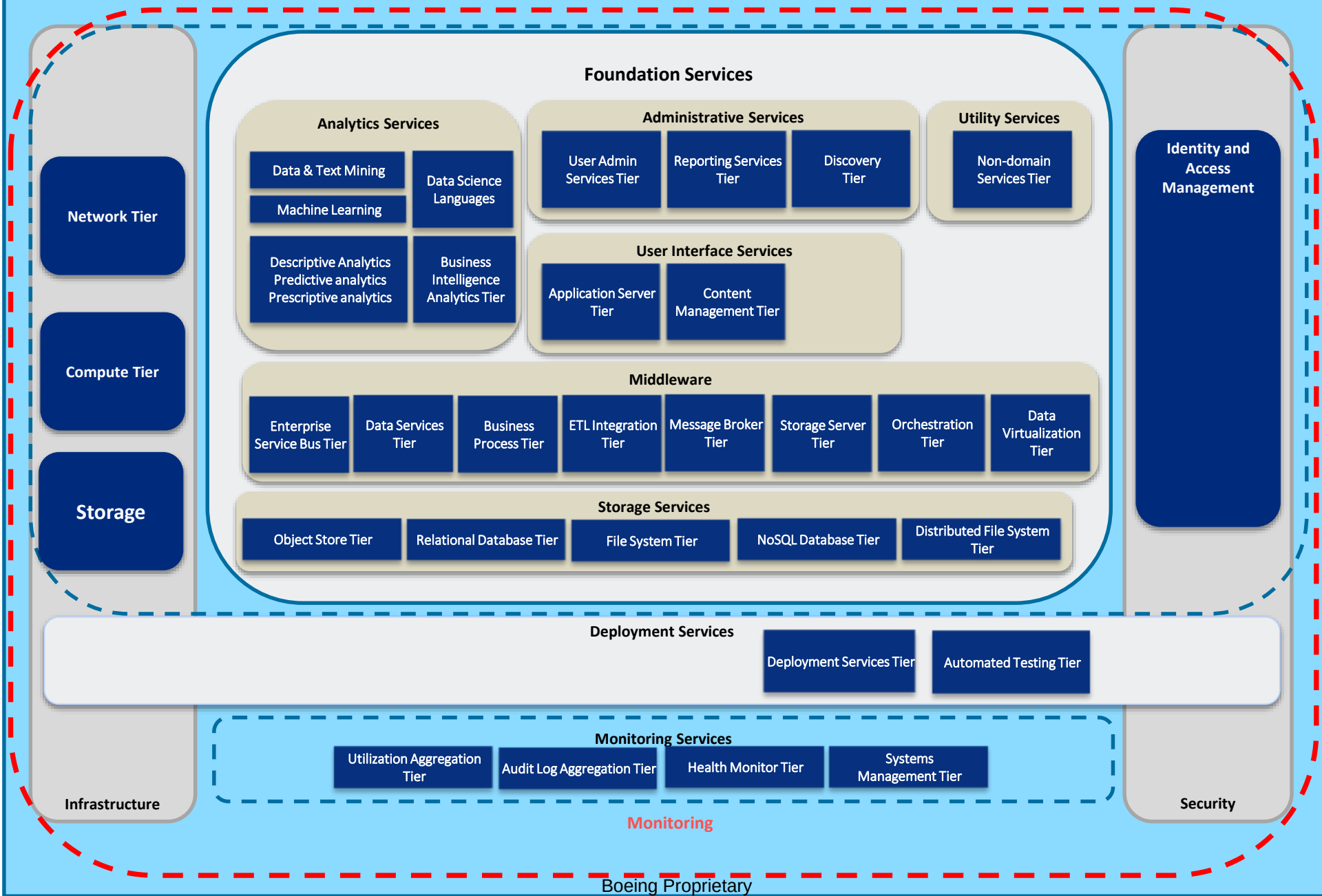


Security

Mobile



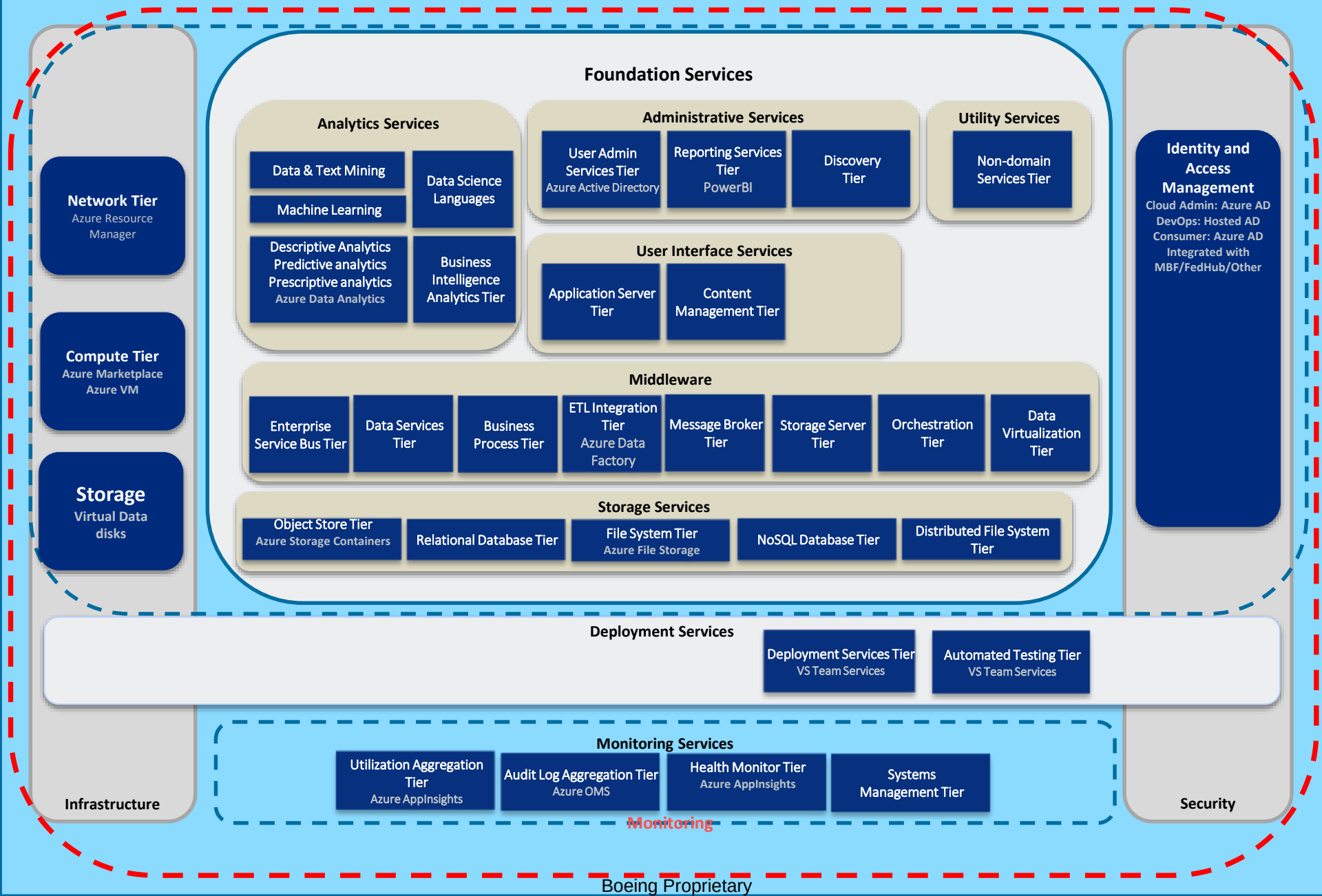
Reference Architecture –System/Platform View



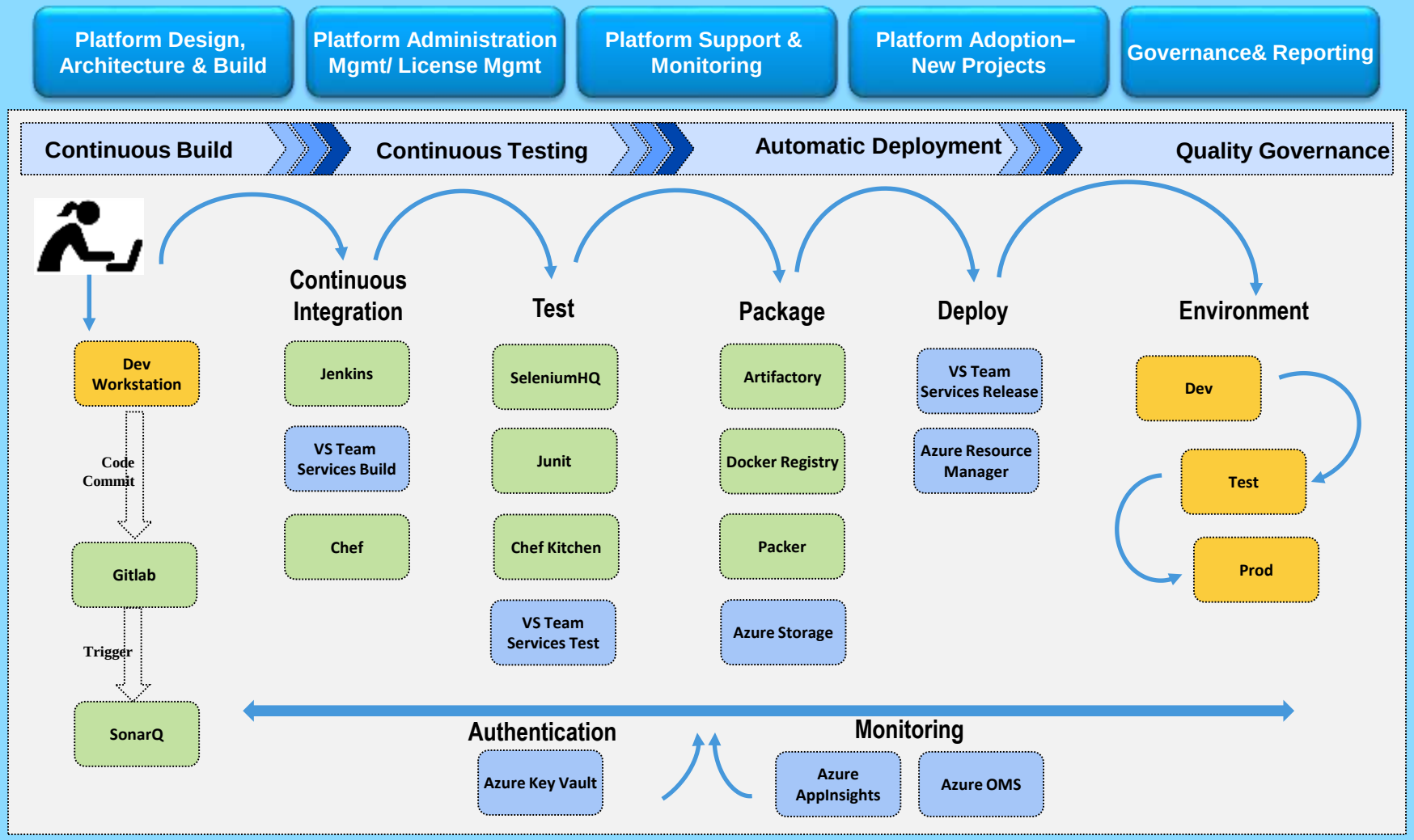
Azure – Current Boeing Utilization

- Azure has been designated as a primary Cloud Deployment Platform for Digital Aviation Products
- Initial Azure Deployment and Testing efforts are underway
  - Multiple projects are in work on the Azure Platform
  - The DA Azure Architecture is in a formative state
  - This guidance represents a “current state” overview for the DA use of the Azure Platform
  - The DA Azure DevOps team has produced an **Azure Onboarding Guide** that is the authoritative source for up-to-date Azure deployment information for DA Dev Teams
  - Select sections of this guide are presented within this document as general guidance and for context

Reference Architecture –System/Platform View - Azure



Shared Services to Drive Scale – Azure/Toolbox POC



Azure Onboarding Introduction

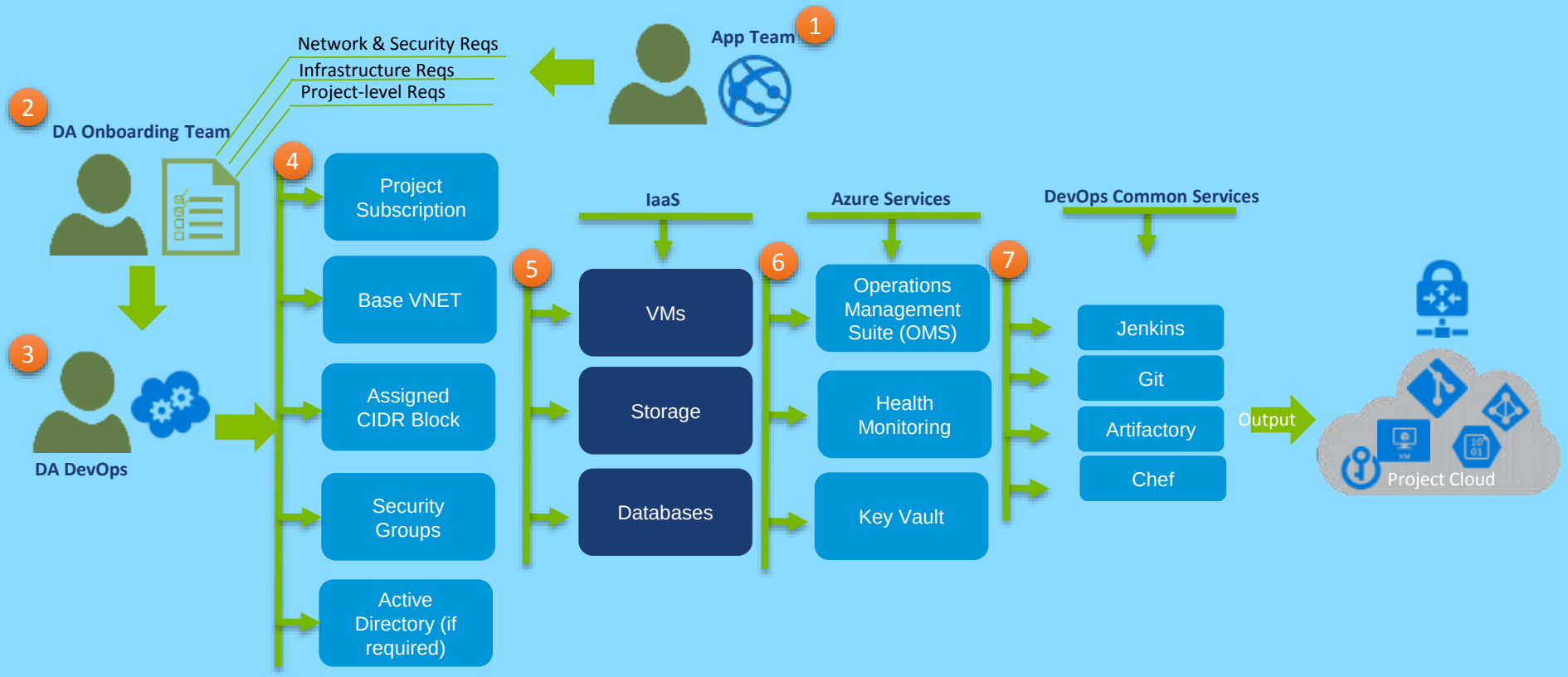
- DA Azure environment architecture
- Project subscription model
- Approved list of Azure Managed Services
- Allowable data types in Azure
- Project classification categories
- Training options

- Production onboarding steps
  - Guided high-level steps to host applications on the Azure environment
  - Security Assessment requirements

NOTE: Selected Content Pages from the DA Azure Onboarding Guide follow – please refer to the full Azure Onboarding guide for additional details.

Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Azure
Project Build Out – Base Infrastructure & Services in Azure



Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Approved Azure Managed Services

Category	Service	Description	Use in DA Azure
Identity Management	Azure Active Directory	IAM for managed services	DevOps AuthN/AuthZ specifically for managed services
Monitoring and Reporting	Operational Management Suite	Audit log monitoring portal	DevOps audit log collection, dashboard, and alerts
Monitoring and Reporting	Azure AppInsights	Health and monitoring portal	DevOps health monitor (e.g., are services running and are they performing).
Monitoring and Reporting	PowerBI	Business intelligence tool	DevOps monitoring reports based on OMS and AppInsights data
Monitoring and Reporting	Azure Automation	Automation tool for repetitive and/or complex administrative tasks	Standard responses to alerts (e.g., shut down an open port, on-demand backup)
Code Repo	VS Team Services Code	Code repository (Git)	Code repository for bootstrap infrastructure scripts
Work Item Management	VS Team Services Work	Agile backlog management	Backlog management
Build Automation	VS Team Services Build	Continuous build	Continuous build
Test Automation	VS Team Services Test	Integrated test framework	Test automation
Release Management	VS Team Services Release	Deployment orchestration	Deployment orchestration
Storage	Azure Storage	Storage containers (blob, tables, queues)	Data storage, VM storage, VM images, recovery artifacts, docker registry, oss software for build
Storage	Azure File Storage	NAS as a managed service	In support of Azure scale sets
Deployment	Azure Marketplace	Software (OS, application, plugins, components) pre-vetted for Azure environment	Base OS images (e.g., CentOS)
Deployment	Azure Resource Manager	Deployment automation through infrastructure as code	Deployment of network and VMs
Security	Azure Security Center	Security monitoring	Security monitoring
Security	Azure Key Vault	Secure key management in the cloud	Secure key management in the cloud
Mobile	Notification Hub	Send push notifications to mobile devices	Send push notifications to mobile devices

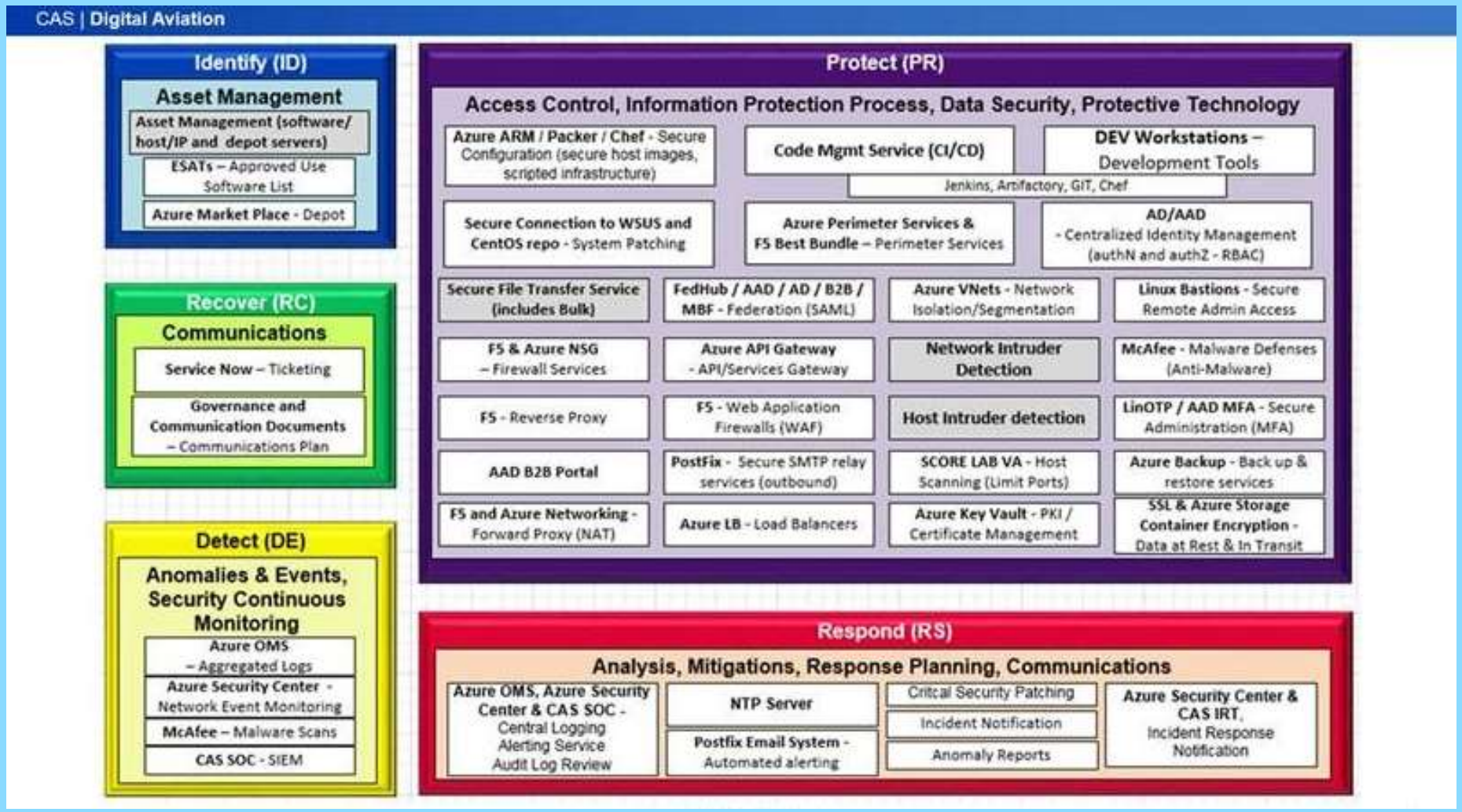
Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Data Classification – What Data can be hosted on Azure?

Data Type	Description	Allowed in DA Azure?
Boeing Public Information	Boeing information that has been authorized for public release through approved release processes or creation processes.	Yes
Material Non-public	As defined in PRO-1752, information that could affect the price of a security and hence a reasonable investor would consider it important in deciding to purchase or sell a security.	No
Boeing Proprietary Information	Information, regardless of form, in which the company has rights and that has actual or potential economic value. Boeing proprietary information does not include other party proprietary information. Boeing proprietary information is used synonymously at Boeing with the term “Trade Secret.”	Yes
Export NLR Information	EAR, no license required.	Yes
Export Licensed Information	EAR or ITAR with specific license and or program need-to-know restrictions.	No
Proprietary Information of Others	Marked and/or unmarked information of a non-Boeing party to which Boeing has an obligation to protect under law or by contract. “Proprietary Information of Others” is also referred to as “Non-Boeing Proprietary Information” and “Third-Party Proprietary Information.”	Yes
Non-sensitive PII	Any information about a person that may be used to identify, contact or locate him or her.	Yes
Sensitive PII	Personal information that could cause significant harm to an individual or to Boeing if it is not properly protected or used, is not collected for a lawful purpose, or has been lost or disclosed to unauthorized persons. Sensitive information includes data that can be used to commit fraud or identity theft, as well as information that has been specifically designated as sensitive by applicable law and regulations. The regulations pertaining to sensitive personal information vary greatly by jurisdiction and change often.	No - Only passwords
Highly Sensitive/Highly Regulated PII	Sensitive PII has been determined to present a particularly high impact to individuals or to the company if lost, compromised or misused	No
CUI	Controlled unclassified information (CUI) in a new category of unclassified information that replaced the various categories used for sensitive but unclassified information. CUI is unclassified information relating to the interests of the U.S. Federal Government or outside entities it believes should be protected.	No
Boeing Non-public Information	Information that has been created by Boeing but has not been marked with an information type or supplemental control. It does not have specific sharing restrictions and is not Boeing public information.	No

Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Azure POC Implementation of Boeing Security Controls



NOTE: Refer to the Security section of this document for more details regarding security controls

Azure Environment Architecture

The DA Azure environment consists of three main components

- Common Services

  - Where all common services like Git, Jenkins, and Artifactory are hosted

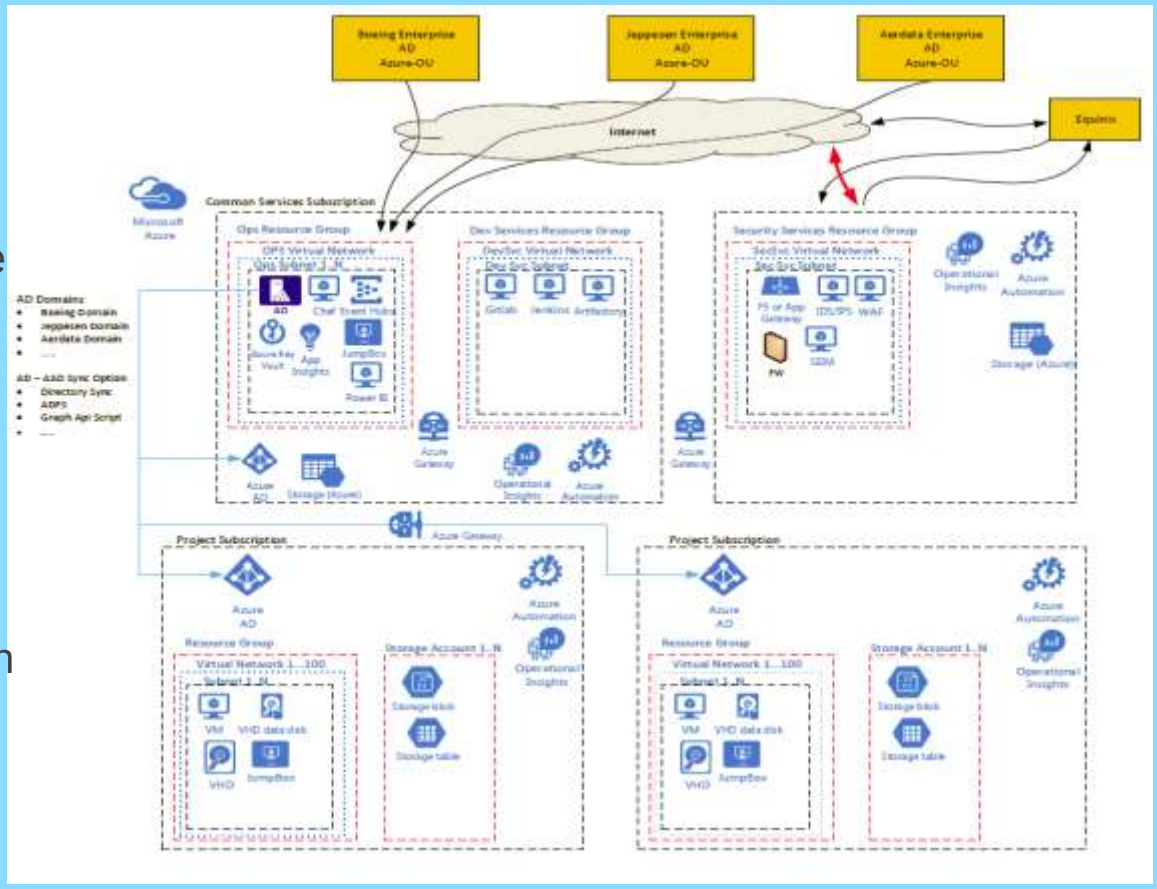
- Common security layer

  - Where all security services are hosted

  - All traffic coming in and out of the environment will pass through the common security layer

- Project-level subscription

  - Each application will have its own project-level subscription



Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Azure Onboarding Guiding Principles

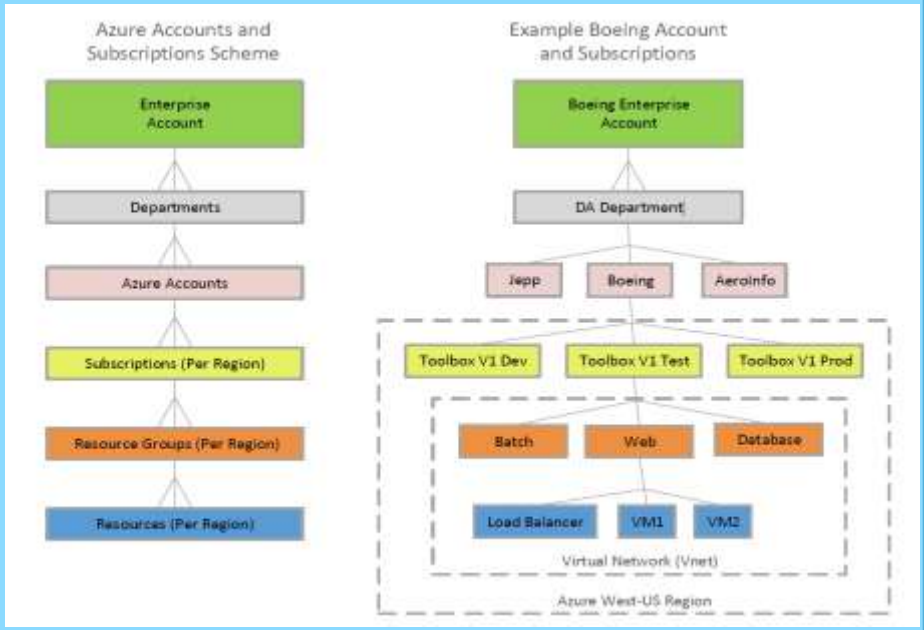
- Azure vetted or managed services are used when possible
- Digital Aviation managed services are used when possible
- All activity is monitored for security and auditing purposes
- All network traffic passes through a common DMZ
- The 80/20 rule is used when constructing new common services
- Project teams are able to deploy updates at their own tempo
- Project teams are able to support and monitor their own products from IaaS up
- The DAA does not attempt to restrict the technical definition of what a product is

Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Azure/DA Subscription Model

Boeing Azure Enterprise Agreement

- Common Services has its own subscription
- Individual projects have their own subscription
- Projects control access to their own subscription
- Resource usage/costs can roll up for easy accounting
- Each project pays for its own consumption of Azure resources



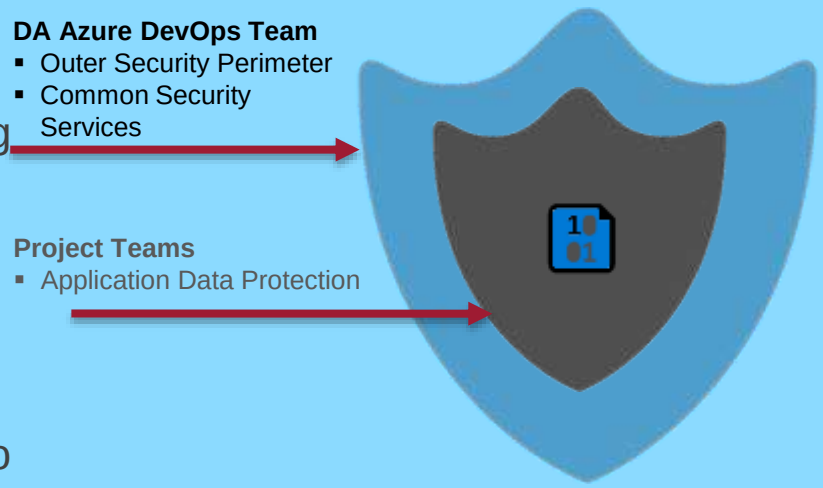
Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Azure FFU/JRA* Security Assessment

- The DA Azure environment has gone through an FFU certification for the outer security perimeter and key infrastructure components
- The certification of the Azure environment application means that teams will be able to leverage much of the previous work, thus requiring less effort on their part
- All applications onboarding to Azure will still be required to go through FFU or JRA

  - Primary focus of security assessment is on <u>application data protection</u>
- The artifacts are largely the same between the two processes; however, anticipated flow times vary greatly between the two (and will vary with application complexity and scale, and priority)

  - Std flow time for FFU: 96 business days
  - Std flow time for JRA: 40 business days

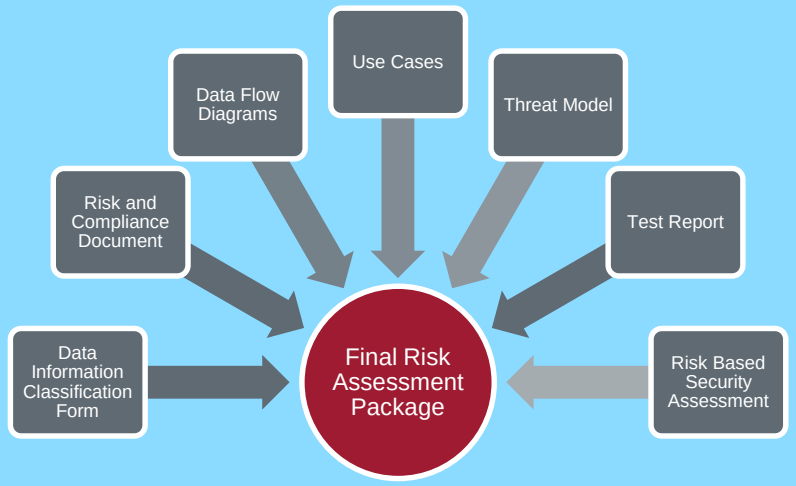


*** FFU: Fitness For Use – Boeing Security Assessment**
JRA: Jeppesen Risk Assessment

Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

Application FFU: Kickoff and Intake Deliverables

- Onboarding forms
  - Intake form
  - FFU – IS IRE form
  - P100s (business case) document
  - Executive overview 4-square
  - *P140 - Enterprise architecture required - required for project exit
  - *P370 - Enterprise architecture required - required for project exit
- Kickoff (information gathering, ensure FFU process needed)
- Data focal and security focal assigned : Security business partner questionnaire required



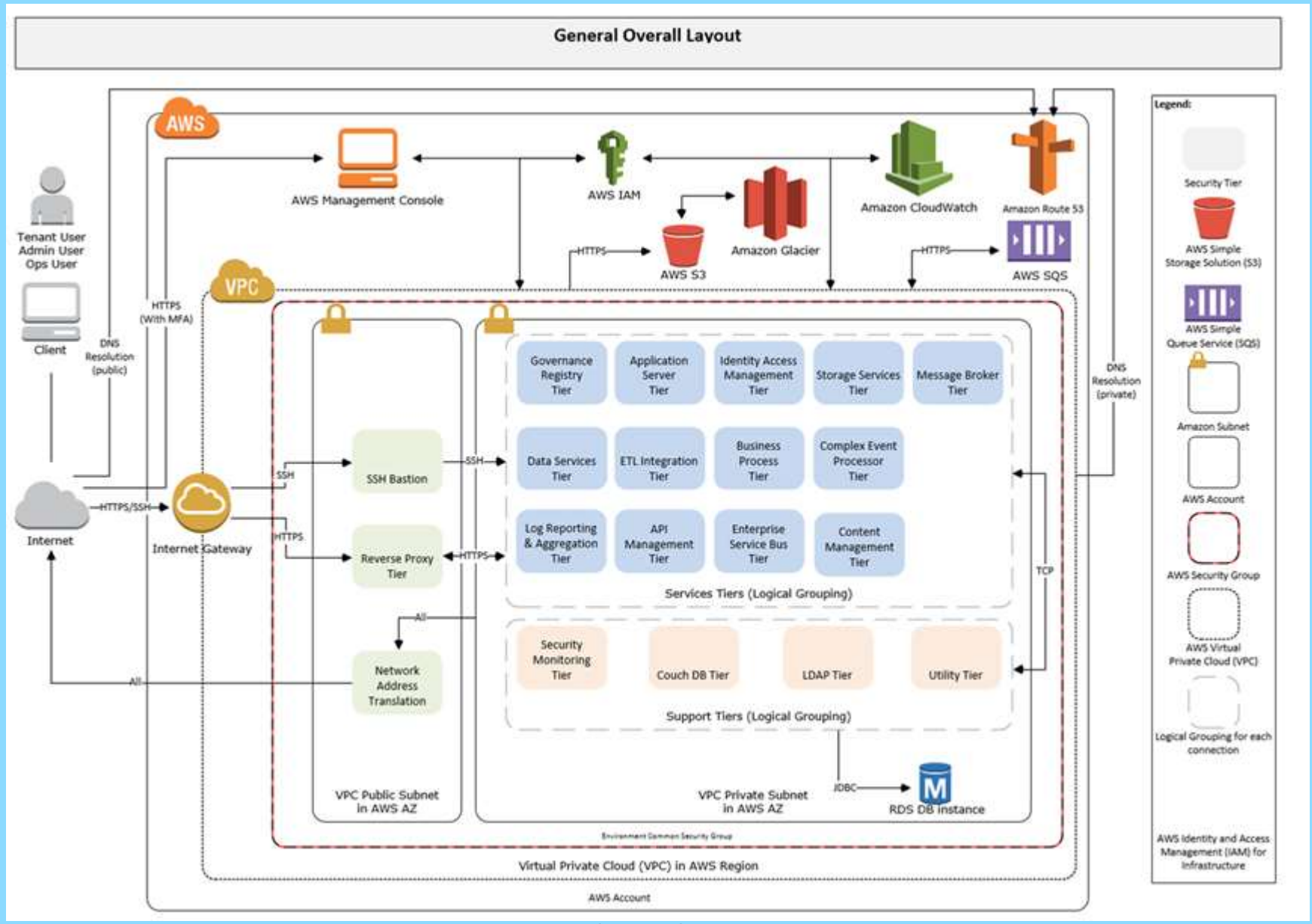
Source: Boeing Digital Aviation Azure Onboarding Guide V1.1

AWS (Amazon Cloud) – Current Boeing Utilization

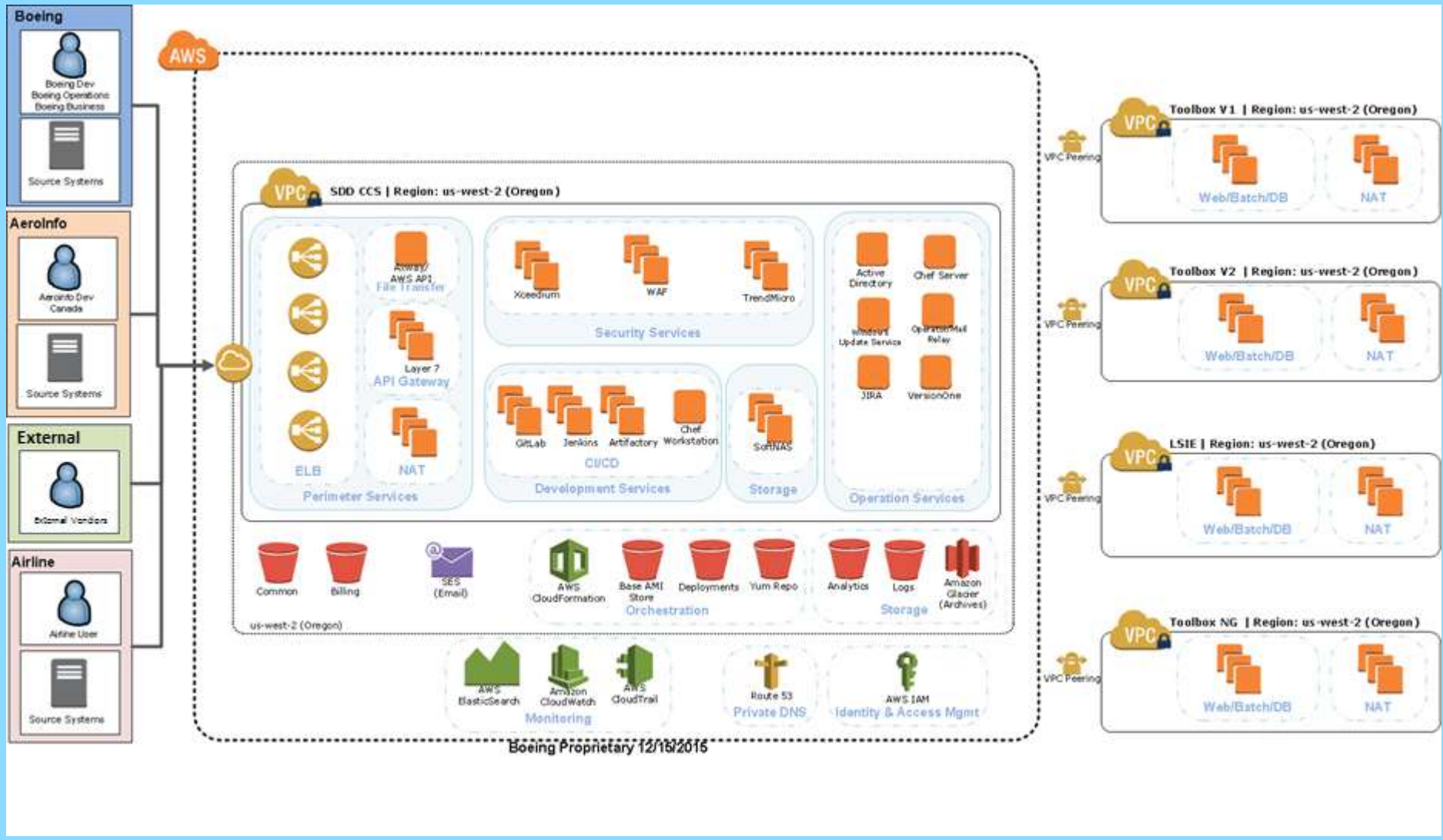
- The Digital Aviation Platform (DAP) for Digital Aviation is built on the Amazon AWS Platform.
- DAP has evolved over the past few years and is in operational/production mode currently supporting several Maintenance-related customer-facing applications
- The AWS platform (non-DAP) is also being used by various other Digital Aviation initiatives (Boeing and Jeppesen), supporting both customer-facing production applications as well as a number of work-in-process projects in data and analytics

NOTE: Additional material on AWS projects within Digital will be provided in future versions.

Current Digital Aviation Platform – AWS Example



Digital Aviation Product Deployment Example - AWS



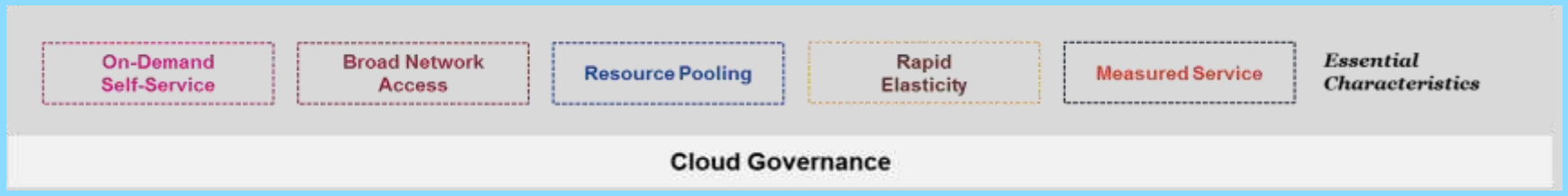
Boeing Proprietary 12/15/2015

Boeing Internal Cloud – Work in Process Plans

The Boeing Internal Cloud hosting solution design is currently based on HP Managed Private Cloud. It would be customized to Boeing needs (created based on specific Digital Aviation requirements) and utilize existing Data Center facilities

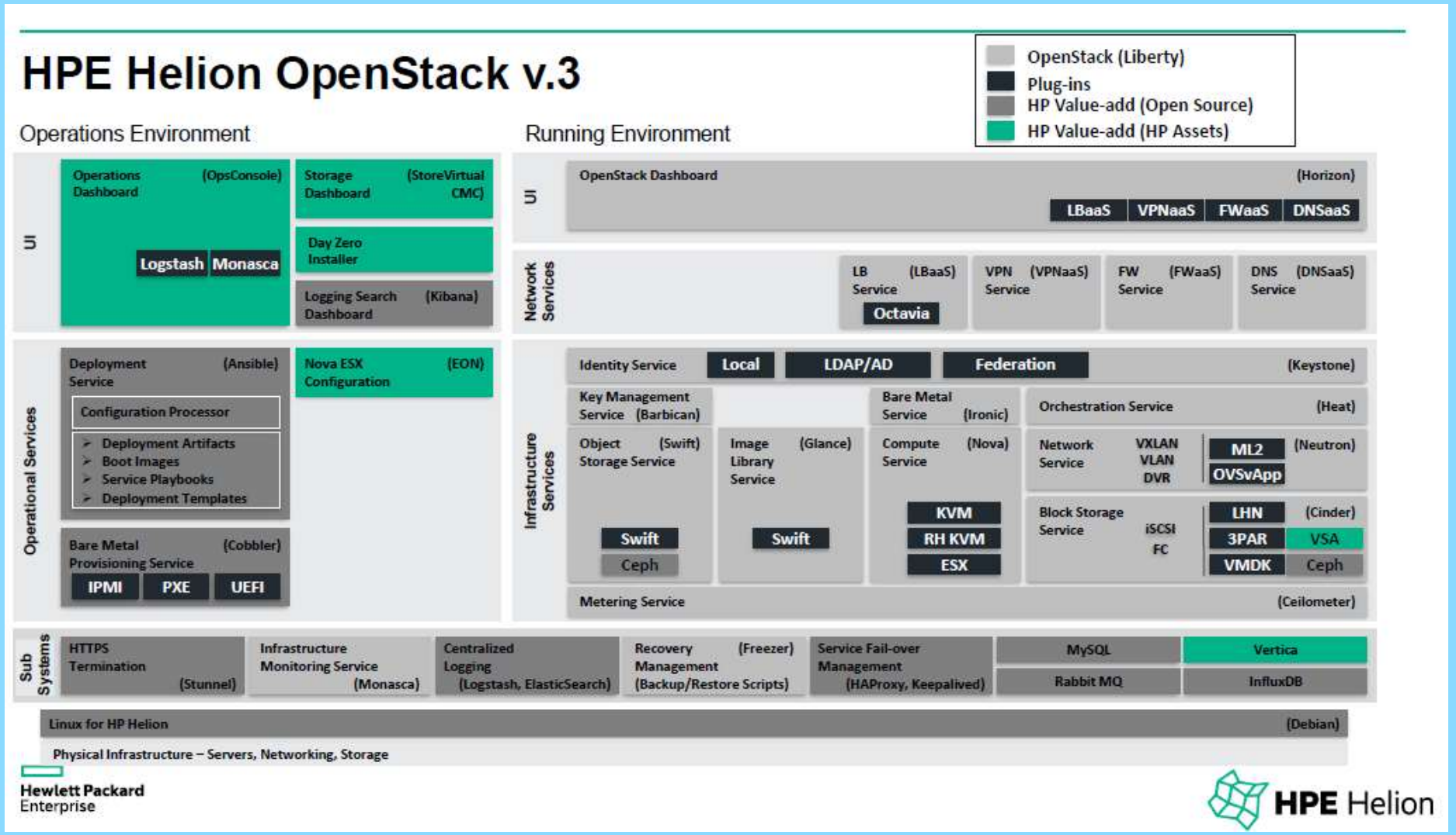
Planned Services

- Application Platform Service
- Cloud Template Service
- SDK Service
- Simple Storage Service
- Software Defined Networking
- Elastic Block Storage Service
- Provisioning Service
- Auto Scaling Service
- Load Balancing Service
- Web Application Firewall Service
- Container Registry Service
- Container Service
- Monitoring Service
- Planning and Audit Service
- Identity and Access Management
- Inventory Management Service
- Environment Configuration Service
- Certificate Management Service
- Code Deployment Service
- Code Pipeline Service
- Tracking Service
- Batching Service
- Managed Service for Oracle and SQL



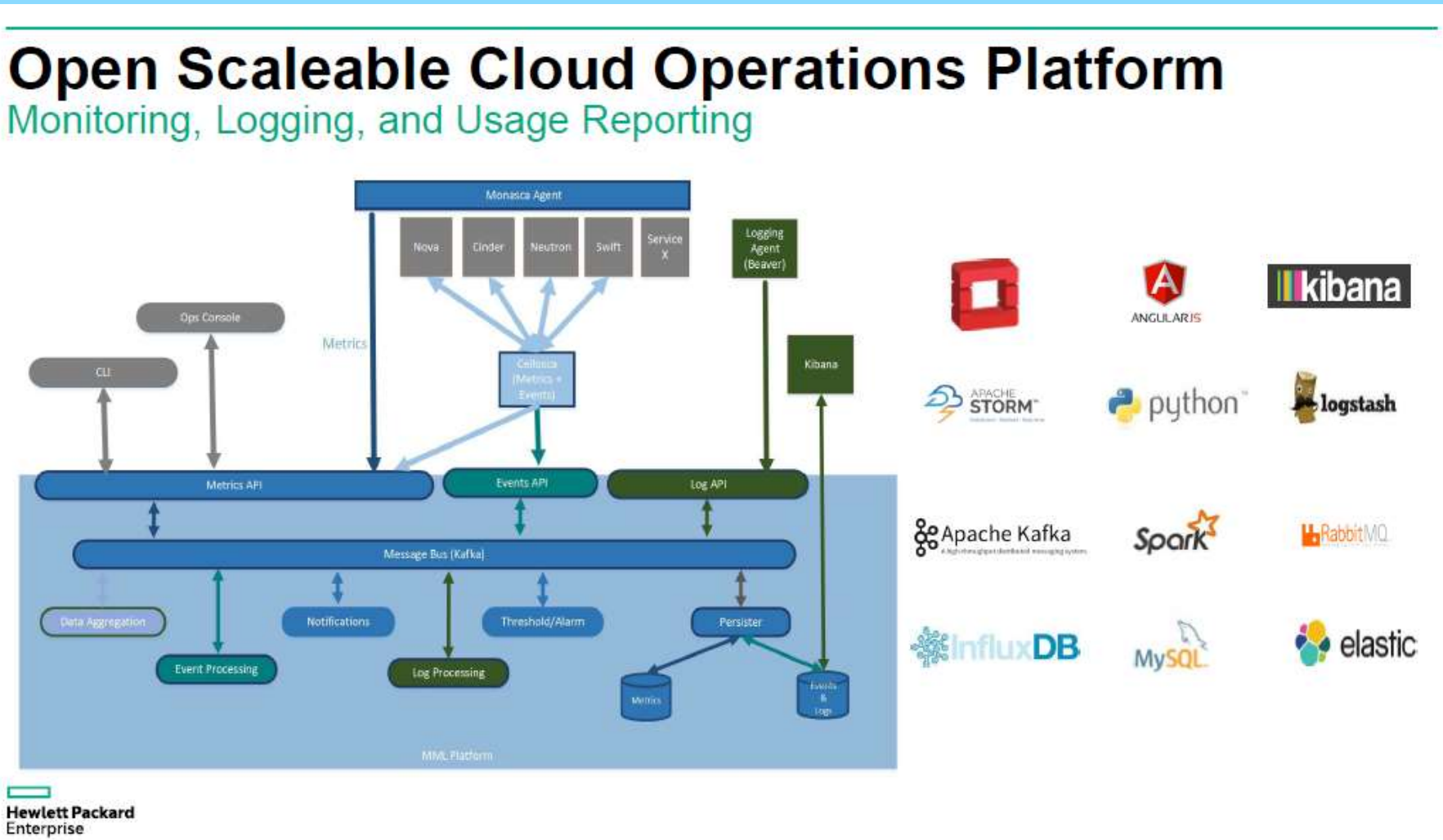
Boeing Internal Cloud – Planned Technology

HP Helion OpenStack – Architecture/Component Overview



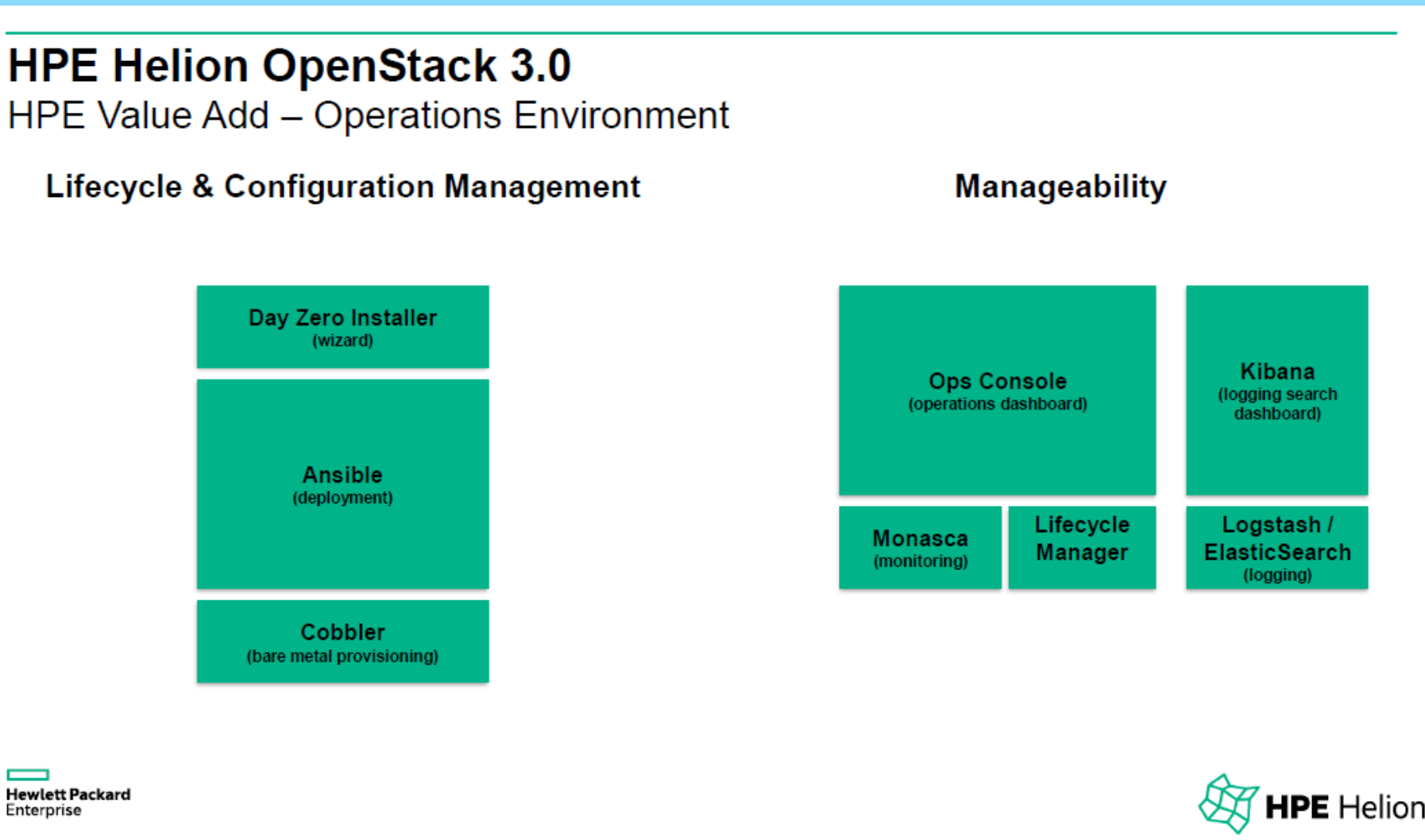
Boeing Internal Cloud – Planned Technology

HP Helion OpenStack – Architecture/Component Overview



Boeing Internal Cloud – Planned Technology

HP Helion OpenStack – Architecture/Component Overview



Data Reference Architecture

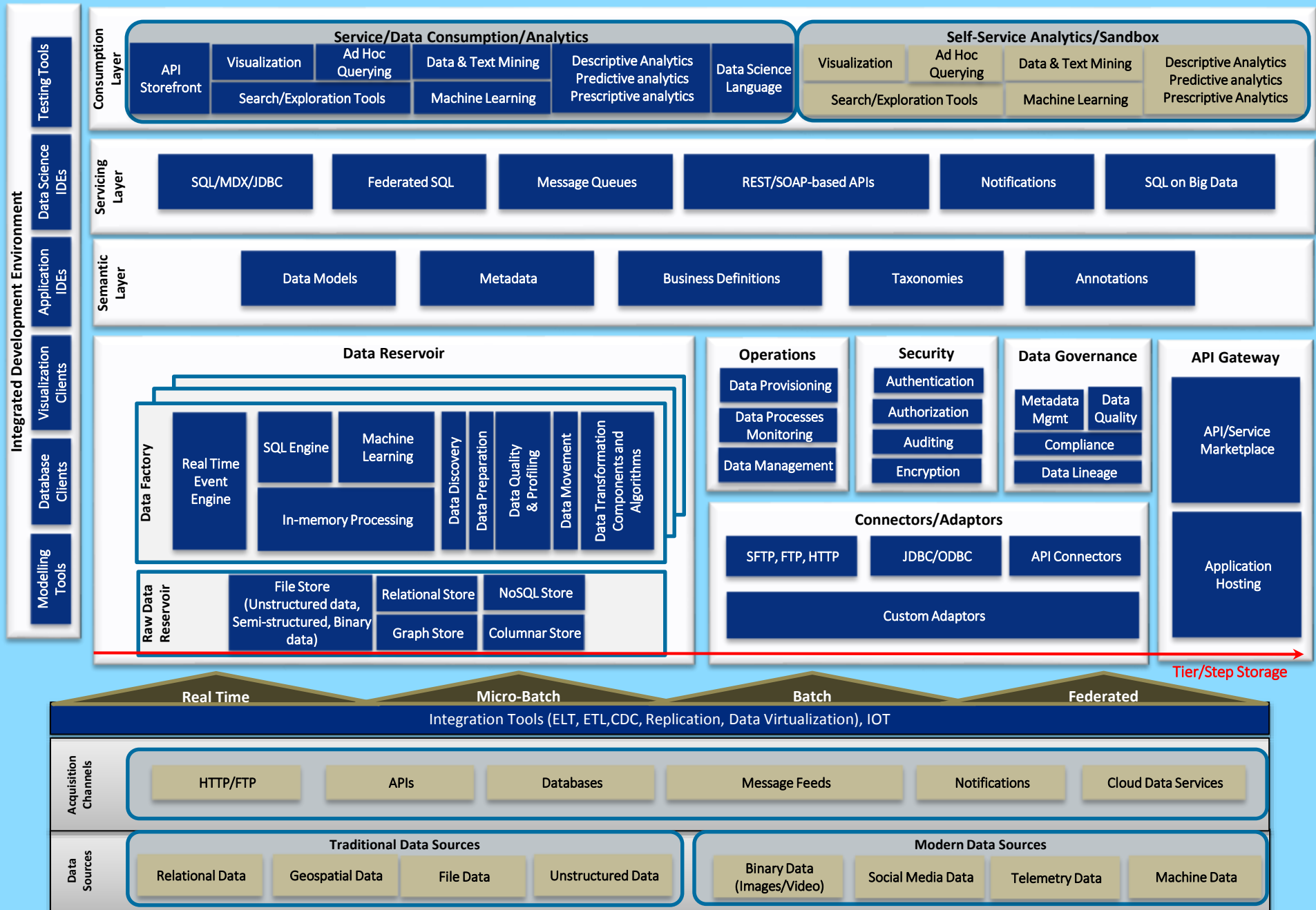
Return

Data Reference Architecture Summary

Description

- The Data Reference Architecture provide guidance on data and analytics-related tools, techniques and technologies that are judged most applicable to Digital Aviation's portfolio of product.
- The following views are provided for reference:
  - Data Reference Architecture Overview
  - Data Architecture View for Azure Deployment (Microsoft Cloud)
  - Data Architecture View for AWS Deployment (Amazon Cloud)
  - Data Architecture View for HP Helion Openstack Deployment (Boeing Internal Cloud)
- Architectural Goal for Data Reference Architecture is to make use of common tools and standardized data sources whenever possible:
  - Reduces complexity of implementing content security
  - Leverages existing Data sources and capabilities (HELIX examples and reference architecture provided)
  - Ensures consistency of data across the full application portfolio
  - Provides for reduced time to market for product integrations across portfolios and with external sources of content (such as airlines or vendors)

Data Reference Architecture



Data Reference Architecture – Azure Version

Azure
Non-Azure Custom

Integrated Development Environment
Selenium, HP Load Runner, HP Quality Center
I-Python Notebook, R-Studio Web
Eclipse, UnixShell
Tableau Desktop, D3 (Browsers)
MySQL Workbench, Oracle SQL Developer
Sybase Power Designer, ER Studio

Consumption Layer
Service/Data Consumption/Analytics: Azure API Management; Zoomdata; Tableau; Power BI; Pentaho; Datameer; Azure Machine Learning; jupyter (multiple data science Languages); R-Studio
Self-Service Analytics/Sandbox: Digital Aviation Advanced Analytics Framework Components, Commercial Products (On-demand data science functionality)

Servicing Layer
Cisco Information Server; Azure Service Bus; Azure Notification Hub; Apache Drill; Spark SQL; Presto

Semantic Layer
Tamr; Data Catalog; From other tools within Data Factory ingested with Custom Data Models/Taxonomies such AIT (Aviation Information Taxonomy)

Data Reservoir
Data Factory: Stream Analytics; Spark Streaming; Storm; Spark SQL; HD Insight; Datameer; Tamr; Trillium; Pentaho; Paxata; Informatica DQ; Azure Data Factory
Raw Data Reservoir: Blog Storage; SQL Database, Document DB, SQL DataWarehouse, Table Storage; Azure Redis (Caching); Azure Backup (Archival)

Operations
?
Splunk; ELK
Log Analytics

Security
Azure AD
DAP LDAP

Data Governance
Trillium
Informatica DQ
Falcon
?

API Gateway
Azure API Management
DAP (Digital Aviation Platform)

Connectors/Adaptors
From tools within Data Factory

Tier/Step Storage

Real Time | Micro-Batch | Batch | Federated

INT.
Cisco Information Server; Azure Event Hub; Azure Data Factory; Informatica; Flume; Goldengate; Twister; Kafka; Azure IoT Hub; Custom Components

Acquisition Channels
HTTP/FTP; APIs; Databases; Message Feeds; Notifications; Cloud Data Services

Data Sources
Traditional Data Sources: Relational Data; Geospatial Data; File Data; Unstructured Data
Modern Data Sources: Binary Data (Images/Video); Social Media Data; Telemetry Data; Machine Data

Data Reference Architecture – AWS Version

AWS
Non-AWS Custom

Integrated Development Environment

Selenium | HP Load Runner | HP Quality Center
I-Python Notebook | R-Studio Web
Eclipse | UnixShell
Tableau Deaktop | D3 (Browsers)
MySQL Workbench | Oracle SQL Developer
Sybase Power Designer | ER Studio

Consumption Layer

Service/Data Consumption/Analytics
- AWS API Gateway
- Zoomdata
- Tableau
- AWS Quicksight
- Pentaho
- Datameer
- AWS Machine Learning
- jupyter (multiple Data Science Languages)
- R-Studio

Self-Service Analytics/Sandbox
- Digital Aviation Advanced Analytics Framework Components, Commercial Products (On-demand data science functionality)

Servicing Layer
- Cisco Information Server
- AWS API Gateway
- AWS SQS
- AWS SNS
- Apache Drill
- Spark SQL
- Presto

Semantic Layer
- Tamr
- From other tools within Data Factory ingested with Custom Data Models/Taxonomies such AIT (Aviation Information Taxonomy)

Data Reservoir

Data Factory
- Kinesis Streaming
- Spark Streaming
- Storm
- Spark SQL
- EMR
- Datameer
- Tamr
- Trillium
- Pentaho
- Paxata
- Informatica DQ
- Data Pipeline

Raw Data Reservoir
- S3 (EMRFS)
- Redshift Aurora Titan DB DynamoDB
- AWS Elasticache (Caching)
- Glacier (Archival)

Operations
- ?
- Splunk
- ELK
- Cloudwatch

Security
- AWS IAM
- AWS Directory Service
- DAP LDAP

Data Governance
- Trillium
- Informatica DQ
- Falcon
- ?

API Gateway
- AWS API Gateway
- DAP (Digital Aviation Platform)

Connectors/Adaptors
- From tools within Data Factory

Tier/Step Storage

Real Time | Micro-Batch | Batch | Federated

INT.
- Cisco Information Server
- Kinesis Firehose
- Informatica
- Flume
- Goldengate
- Twister
- Kafka
- AWS IoT
- Custom Components

Acquisition Channels
- HTTP/FTP
- APIs
- Databases
- Message Feeds
- Notifications
- Cloud Data Services

Data Sources

Traditional Data Sources
- Relational Data
- Geospatial Data
- File Data
- Unstructured Data

Modern Data Sources
- Binary Data (Images/Video)
- Social Media Data
- Telemetry Data
- Machine Data

Data Reference Architecture – HP Helion OpenStack

Helion
Helion Certified

Integrated Development Environment
Selenium | HP Load Runner | HP Quality Center
I-Python Notebook | R-Studio Web
Eclipse | UnixShell
Tableau Desktop | D3 (Browsers)
MySQL Workbench | Oracle SQL Developer
Sybase Power Designer | ER Studio

Consumption Layer
- Service/Data Consumption/Analytics: CTERA API Gateway; Zoomdata; Tableau; Quicksight; Pentaho; Datameer; Machine Learning; jupyter (multiple Data Science Languages); R-Studio
- Self-Service Analytics/Sandbox: Digital Aviation Advanced Analytics Framework Components, Commercial Products (On-demand data science functionality)

Servicing Layer
Cisco Information Server | CTERA Portal | Swift APIs | ? | Apache Drill | Spark SQL | Presto

Semantic Layer
Tamr | From other tools within Data Factory ingested with Custom Data Models/Taxonomies such AIT (Aviation Information Taxonomy)

Helion Content Depot

Data Factory
- Kinesis Streaming
- Spark Streaming
- Storm
- Tibero
- Hortonworks
- Datameer
- OpenDataSoft
- Trillium
- Riak KV
- Ceilometer
- Informatica DQ
- IRIS Big Data DB

Raw Data Reservoir: Object Storage (Swift) | HP StoreOnce | Image Service (Glance) | HP Data Protector | Block Storage (Cinder)

Operations
Horizon | Monasca | ELK | Freezer

Security
ID Services | Directory Service | DAP LDAP

Data Governance
Trillium | Informatica DQ | Falcon | Magnitude

API Gateway
CTERA APIs

Connectors/Adaptors
From tools within Data Factory

DAP (Digital Aviation Platform)

Tier/Step Storage

Real Time | Micro-Batch | Batch | Federated

INT.
Cisco Information Server | Kinesis Firehose | Informatica | Flume | Goldengate | Twister | Kafka | Helion IoT | Custom Components

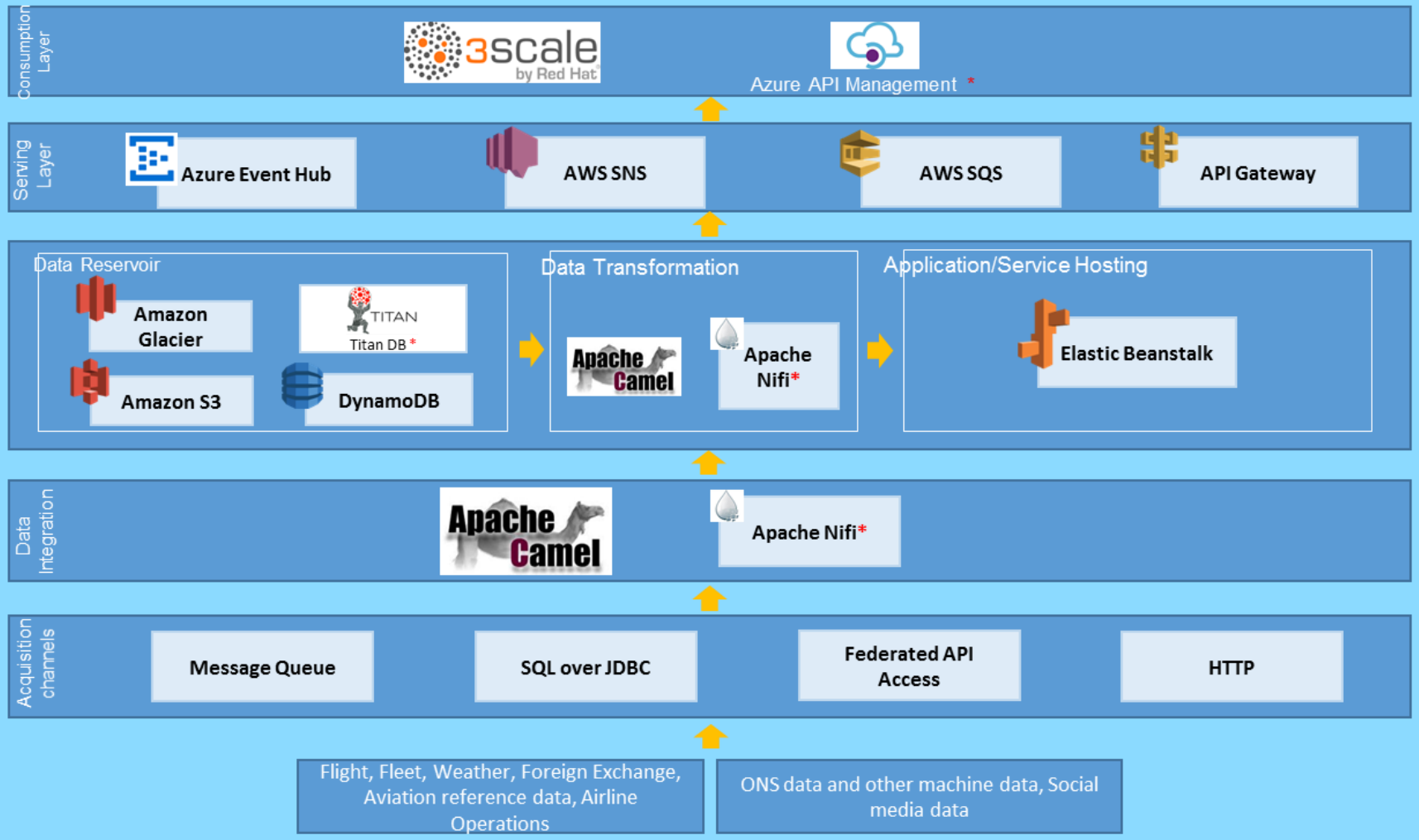
Acquisition Channels
HTTP/FTP | APIs | Databases | Message Feeds | Notifications | Cloud Data Services

Data Sources
- Traditional Data Sources: Relational Data | Geospatial Data | File Data | Unstructured Data
- Modern Data Sources: Binary Data (Images/Video) | Social Media Data | Telemetry Data | Machine Data

HELIX – Data Services Capability for DA

- The HELIX initiative underway within Digital Aviation is developing data content and data services capability with both the Azure and AWS (Amazon) platforms
- As a “real world” instantiation of the Data Reference Architecture, HELIX can offer Digital Aviation projects the following:
  - APIs to access common “high demand” data sources from both Digital Aviation and external sources (such as Weather data)
  - Data Services Capability that can be deployed in various configurations to allow Azure or Amazon based applications to connect to and retrieve data from the various data patterns that are required to support the Digital Aviation portfolio of products
- The material in this document is intended to make development teams aware of the HELIX initiative. Where HELIX capabilities can meet project requirements, the HELIX team should be contacted for specific guidance on how to leverage their capabilities for Azure and/or AWS (Amazon) product deployments.

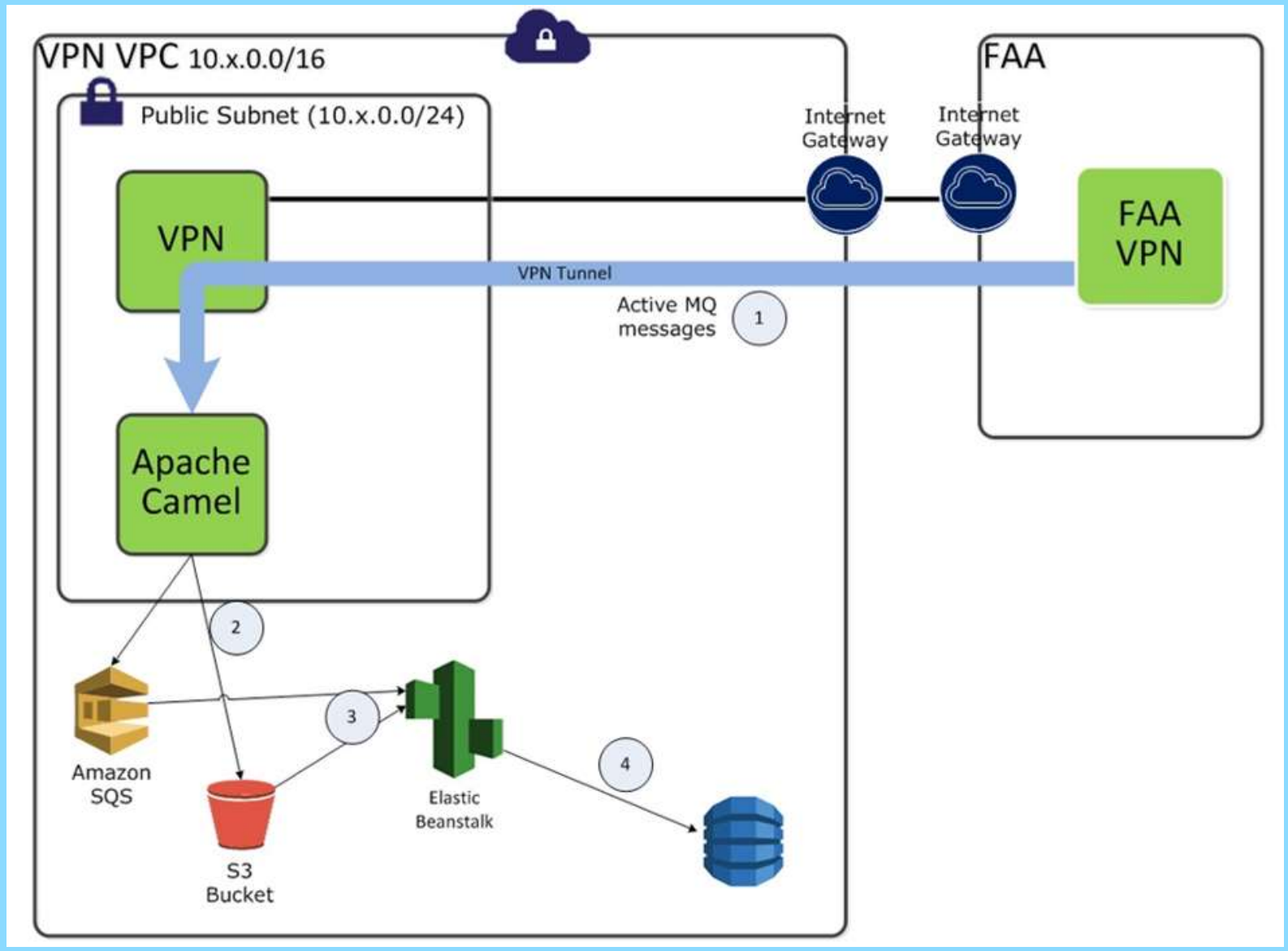
HELIX - Data Architecture – Current View



* Tentative/In work

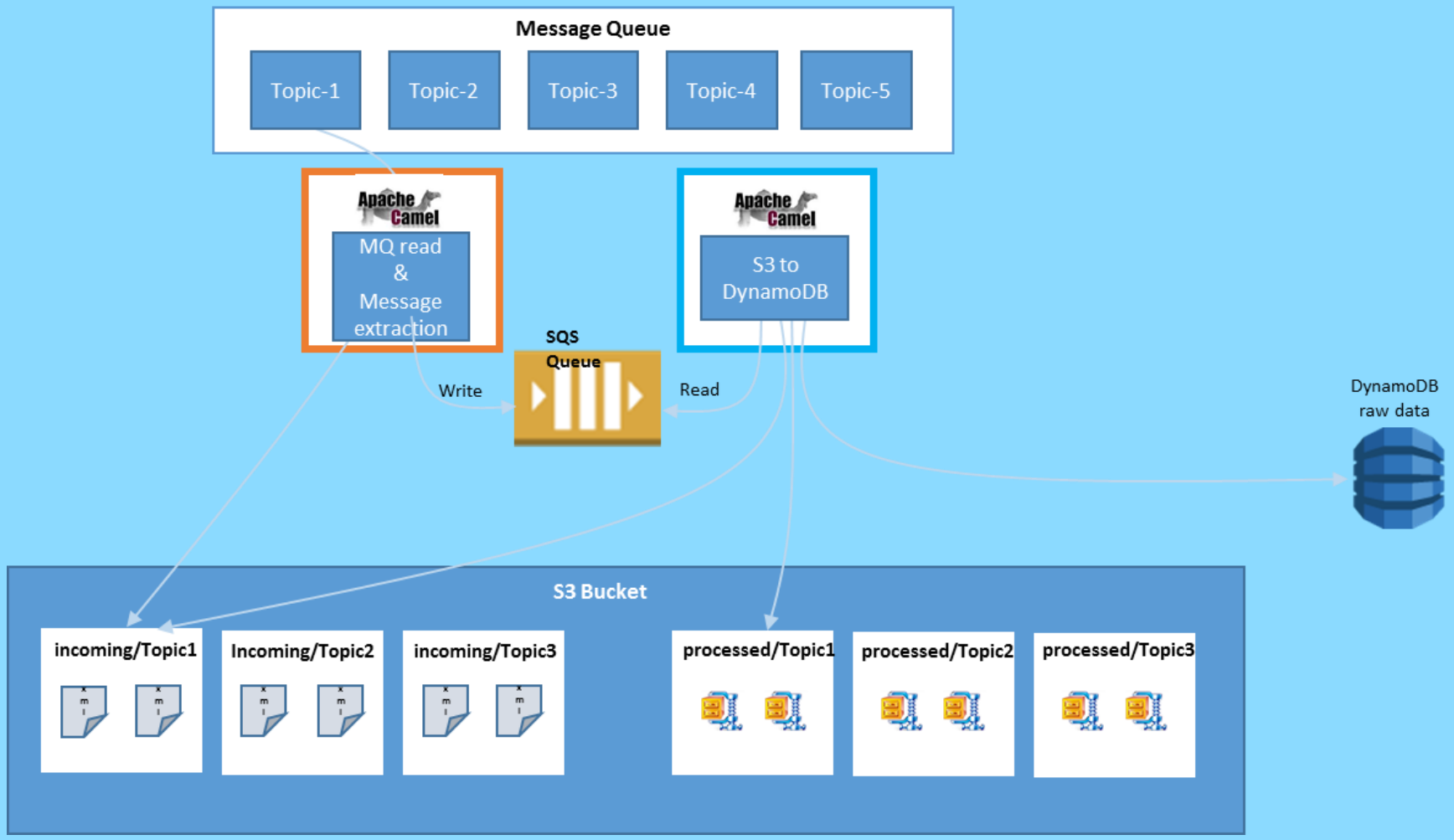
Source: Helix team: Ken Cyr/Ganesh M

HELIX – Real-Time Data Flow Example



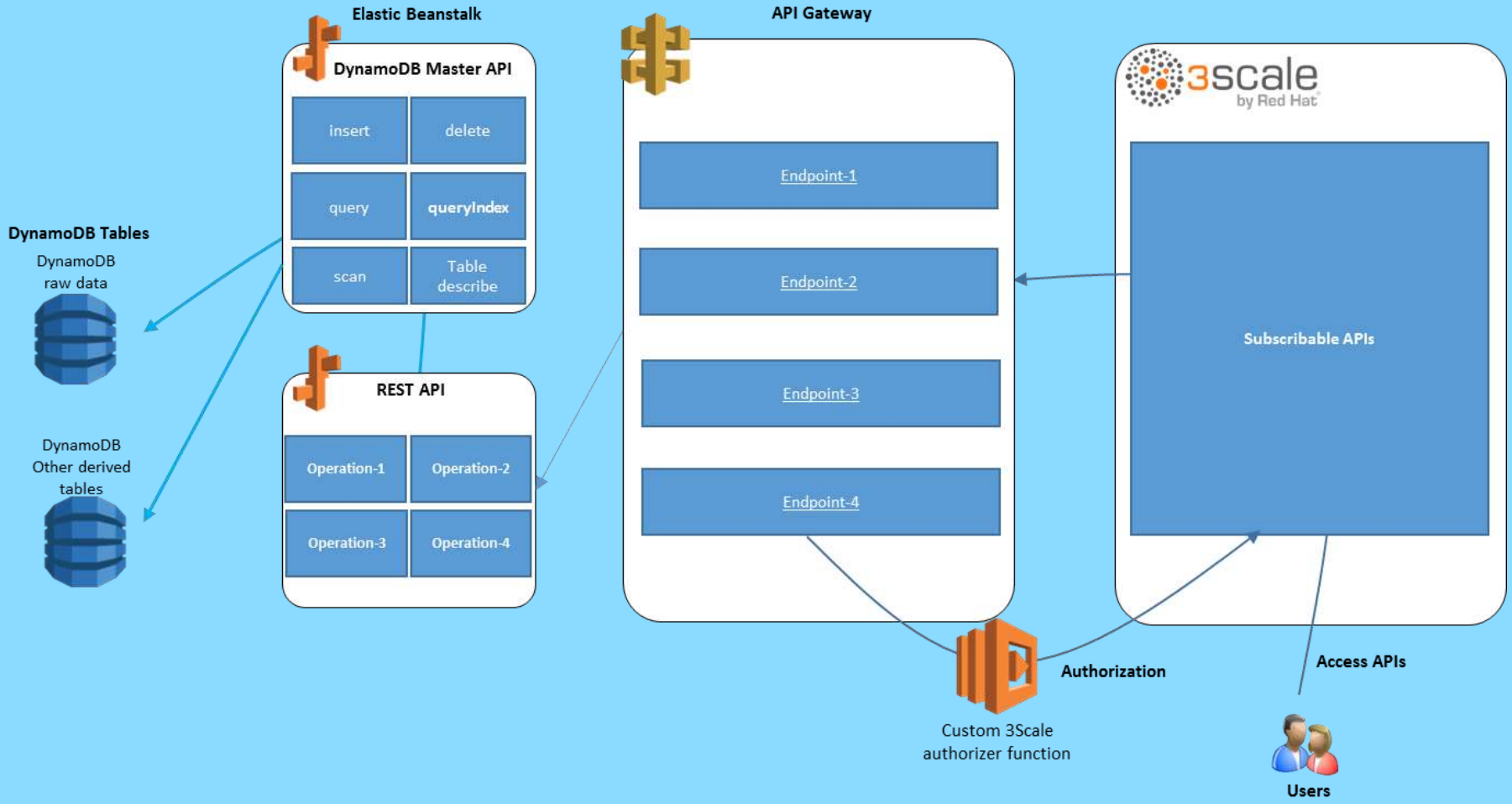
Source: Helix team: Ken Cyr/Ganesh M

HELIX – Incoming Feed Processing Example



Source: Helix team: Ken Cyr/Ganesh M

HELIX – API Creation And Deployment



Source: Helix team: Ken Cyr/Ganesh M

Security Reference Architecture

Return

Security Reference Architecture Summary

Description: Security Reference Architecture provides guidance on the key aspects of security that need to be understood by teams developing, refactoring or re-hosting Digital Aviation Portfolio Products. Both Cloud and Non-cloud deployments are anticipated, and the development teams are strongly encouraged to make use of the guidance, common services and capabilities provided by the deployment platforms and the Digital Aviation Technical Development organization, rather than attempting to develop security capabilities independently.

Highlights:

- Requirements for the security reference architecture are primarily driven out through:
  - NIST Cyber Security Framework
  - CAS Security Assurance Levels (SAL) Framework
  - Critical Security Controls (CSC, v6.0)
  - Boeing InfoSec requirements, coupled with industry standards for cloud computing
- Security requirements will be satisfied by a combination of technical, administrative, and operational controls – this guidance will assist in the implementation of these controls
- Controls are designed, implemented, and supported by the cloud platform vendor, Boeing's managed environment security team, and the Digital Aviation application development teams

General Security Principles

Risk Based - The level of protection should be specific and appropriate to the asset at risk.

- Business demands that security enables business agility and is cost-effective. This is best determined by maintaining an active risk management process.

Usability – Cyber security should be user transparent and not cause users undue extra effort.

- Security controls that are hard to use, cause productivity loss or are otherwise disruptive are often ignored, disabled, or abrogated, leaving resources vulnerable.

Simplicity – Systems & Services should be as simple as possible while retaining functionality.

- Complexity is an enemy of security, making it difficult to analyze risk and detect threats. Security mechanisms must be pervasive, simple, scalable, and easy to manage.

Security by Design - Security should rely on specific, proven controls rather than obscurity.

Context - Assume context at your peril.

- Security solutions designed for one environment may not be transferable to effectively work in another.

Security Design Principles

Devices - All devices must be capable of maintaining their security policy on an un-trusted network.

- The proliferation of small, mobile devices demands that they carry their cyber security with them as they transit different environments.

Defense in Depth - Greater security is obtained by layering defenses.

Least Privilege - Principals (People, Things, Processes, etc.) should be granted only the rights necessary to perform authorized tasks.

Access Control – All resources of value should be protected by scalable access control mechanisms.

- Access control policies should be centrally managed by resource owners.
- Identity Management, Authentication, and Access Control Decisions should operate as globally as possible.
- Authorization controls should be enforced as close to the resource as possible.

Communication - Devices and applications must communicate using open, secure protocols.

- In today's over-connected world, no unencrypted transmission can be assumed to have any level of security

Comprehensive Security Framework

- **A security framework provides an organizing model for security functions. This makes it easier to determine omissions and eliminate duplications.**
- **A good security framework provides a holistic view of security capability including people, processes and technology.**
- **The technology portion of a security framework can provide an authoritative source for a comprehensive set of security services.**
- **We use the NIST Cybersecurity Framework (CSF) which was developed by an international group representing hundreds of organizations and coordinated by US NIST.**

NIST Cyber Security Framework (CSF)

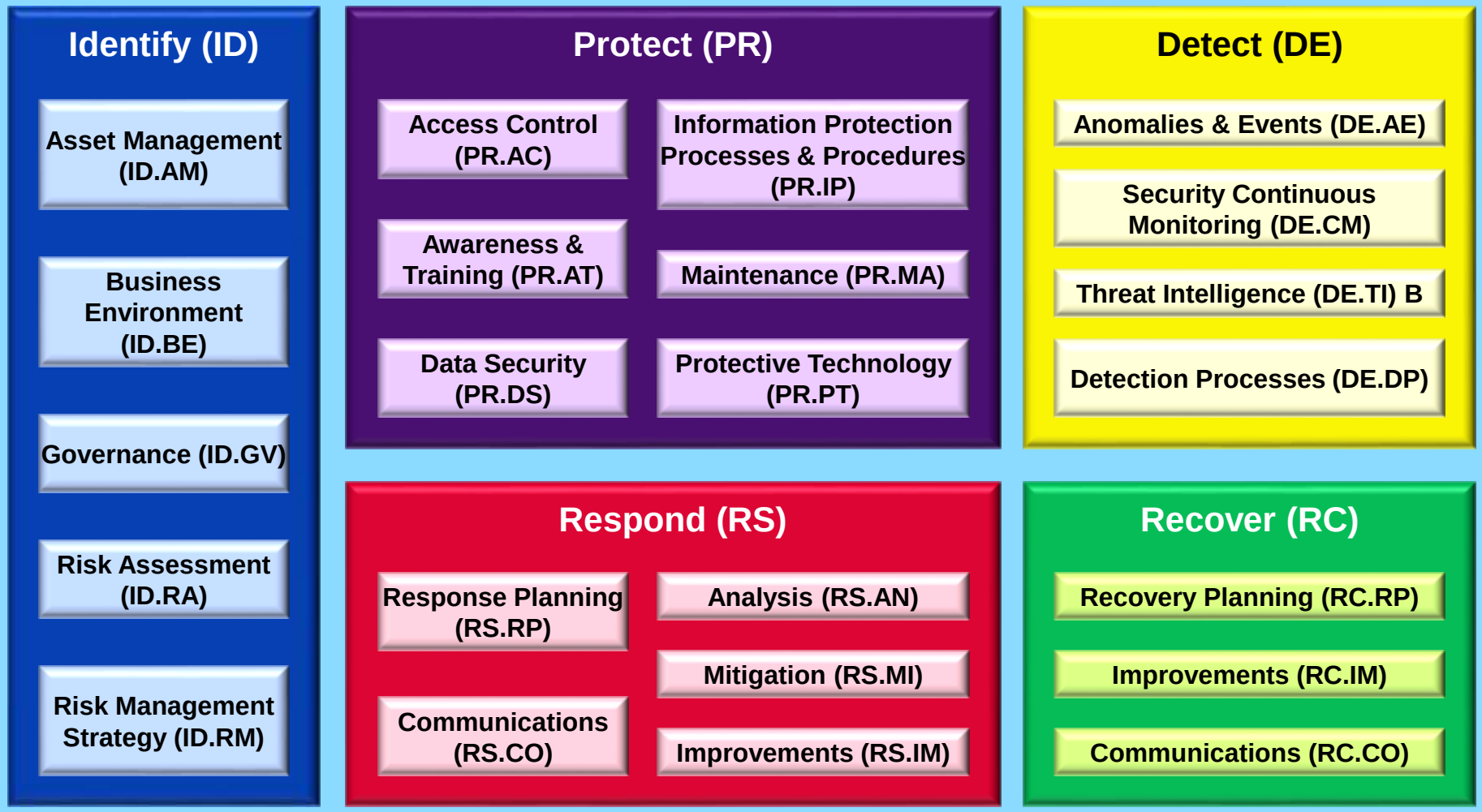
NIST Cyber Security Framework (US Presidential EO 13636)

- Risk based approach
- Traceable to US government standards
- Potential audit standard for most US government agencies
- Mappings to compliance standards (NIST 800-53, ISO 27002, COBIT, ISA, etc.)
- Automated tools in development, early tools available from NIST

Five “Functions” for Managing Risk

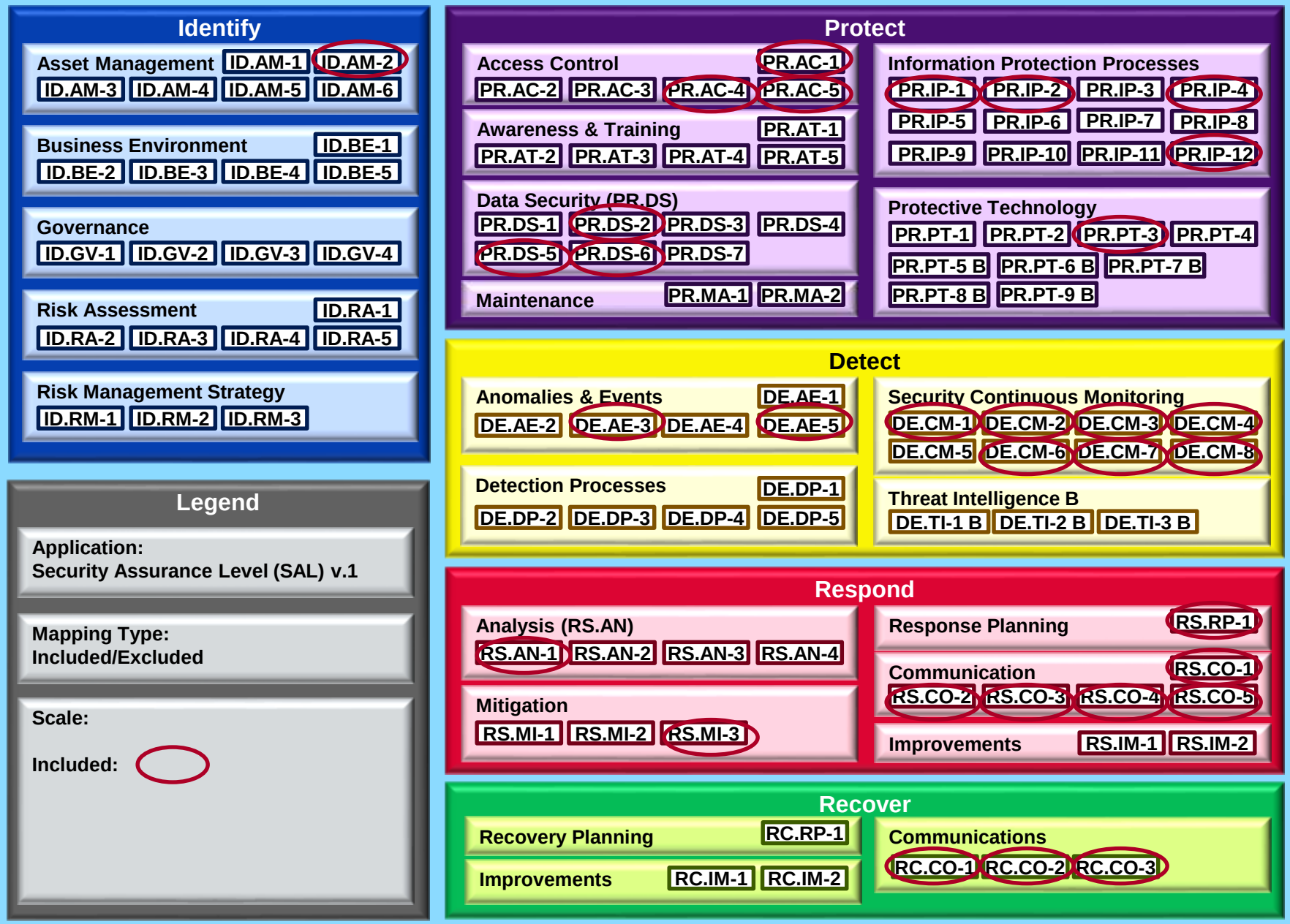
- **Identify**, **Protect**, **Detect**, **Respond**, and **Recover**
- Functions broken Down into Categories and Sub Categories
- NIST Standard Color Coding, and Tagging for easy reference and communication across government and industry

NIST Cyber Security Framework (Top Level Functions)



Based on US NIST Cybersecurity Framework

Security Assurance Level Mapping to Cyber Security Framework



Common Security Controls for Infrastructure as a Service

Identify (ID)

Asset Management

- Asset Management (software/ host/IP and depot servers)
- Approved Use Software List
- Depot/Distribution Servers

Recover (RC)

Communications

- Incident Response/ Management Process (ticketing)
- Communication Plan

Detect (DE)

Anomalies & Events, Security Continuous Monitoring

- Aggregated Logging/SIEM
- Vulnerability Scans
- Malware Scans
- Network Event Monitoring

Protect (PR)

Access Control, Information Protection Process, Data Security, Protective Technology

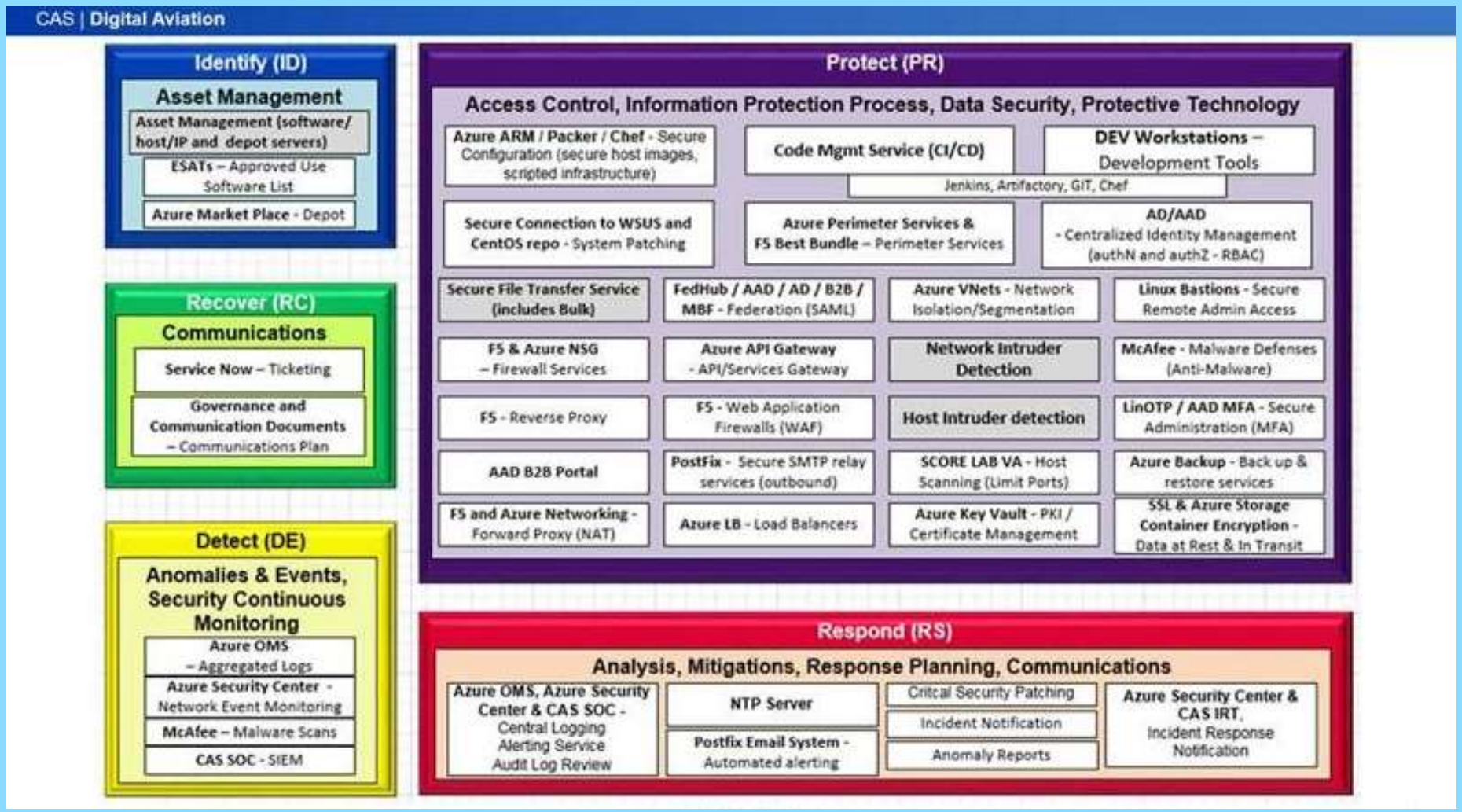
- Secure Configuration (secure host images, scripted infrastructure)
- Code Mgmt Service
- (CI/CD) (Jenkins, Artifactory, GIT and Orchestration tool like Continuum)
- Development Tools
- Config Mgmt, Provisioning (Chef, Cloud Foundry, etc.)
- System Patching Tool/Process (secure connectivity to WSUS/ CentOS repositories)
- Perimeter Services
- Centralized Identity Management System (authN and authZ - RBAC)
- Secure File Transfer Service (includes Bulk)
- Federation (SAML)
- Network Isolation/ Segmentation
- Secure Remote Access Channels (Admins)
- Firewall Services
- API/Services Gateway
- Network Intruder detection
- Malware Defenses (Anti-Malware)
- Reverse Proxy
- Web Application Firewalls (WAF)
- Host Intruder detection
- Secure Administration (2FA)
- Portal
- Secure SMTP relay services (outbound)
- Host Scanning (Limit Ports)
- Back up & restore services
- Forward Proxy (NAT)
- Load Balancers
- PKI / Certificate Management
- Secure Data (at rest & in transit)

Respond (RS)

Analysis, Mitigations, Response Planning, Communications

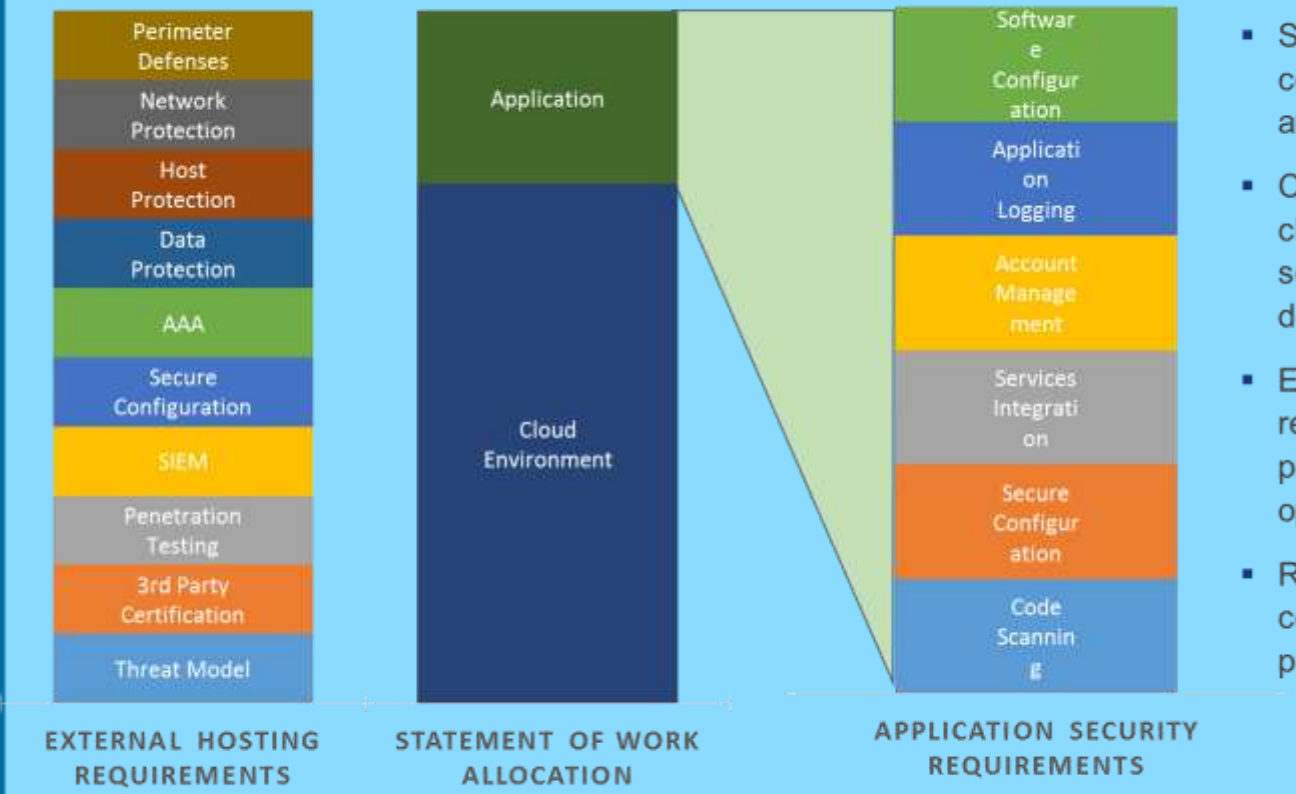
- Central Logging & Alerting Service
- NTP, synchronized time sources for log analysis
- System Patching Tools/ Process
- Incident Response Notification
- Audit Log Review Procedure
- Automated alerting
- Incident Notification
- Anomaly Reports

Azure Implementation Example
Boeing Security Controls



Security Assurance Levels – Implementation Concepts

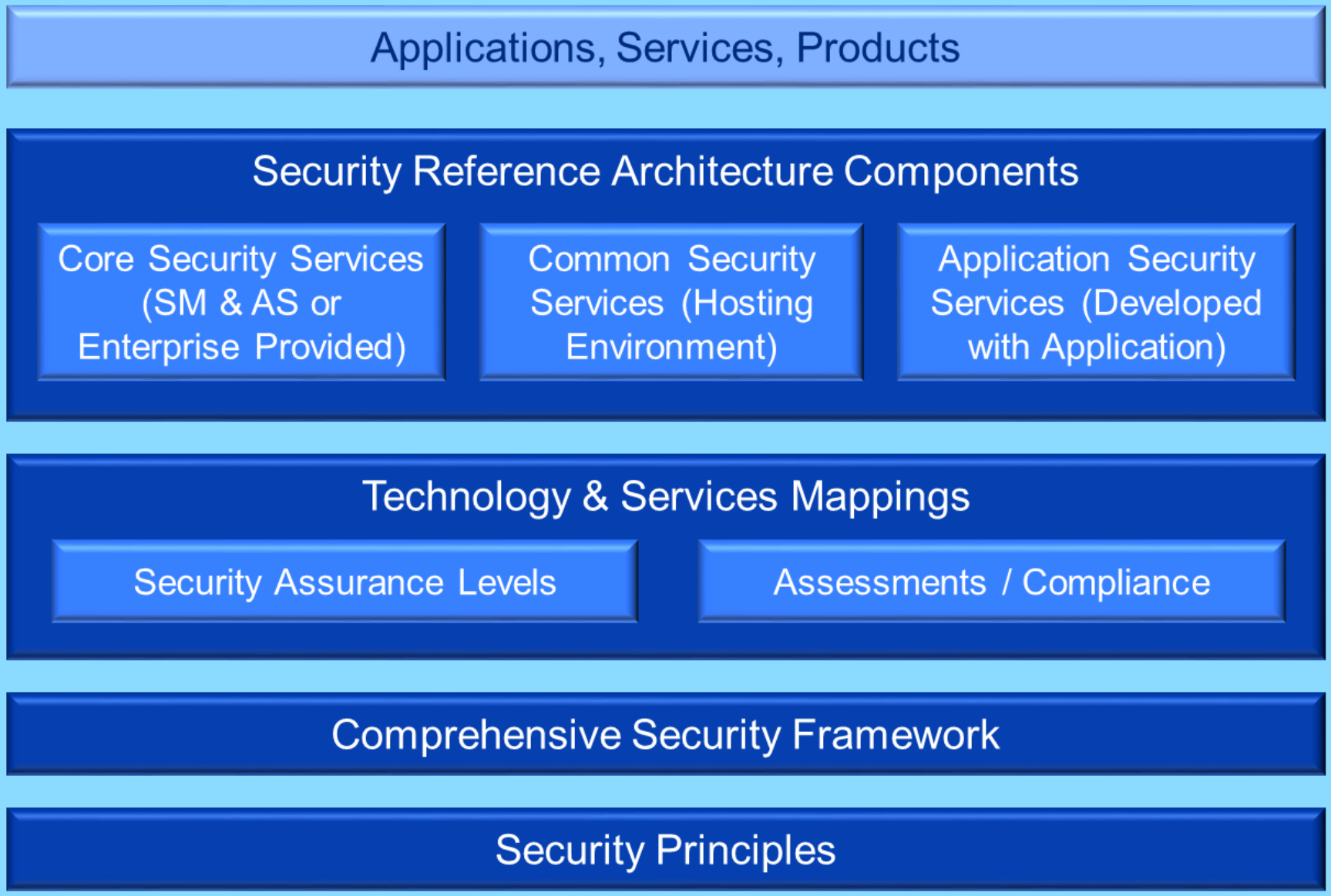
Cloud Hosting Environment
Pre-deployed security controls



- Security requirements are satisfied by a combination of technical, administrative, and operational controls.
- Controls are provided by a combination of cloud vendor, managed environment security team, and the application development team.
- Environment provides the vast majority of required security controls as part of a prescriptive implementation pattern and operational support process.
- Risk Assessment becomes a validation of conformance to an already approved pattern.

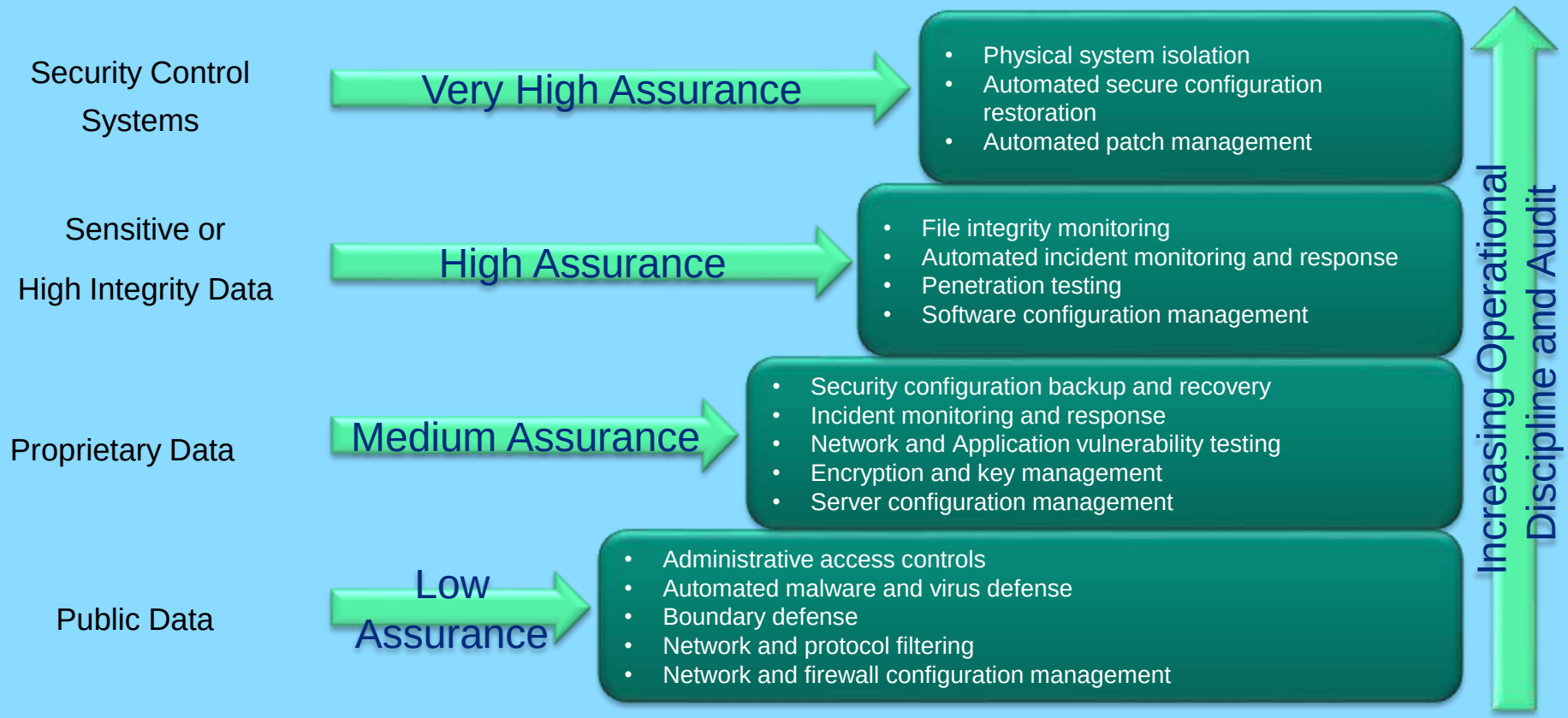
Environment Should Prescribe the Majority of Security Controls

General Security Architecture



Security Protection Framework

Security Assurance Levels



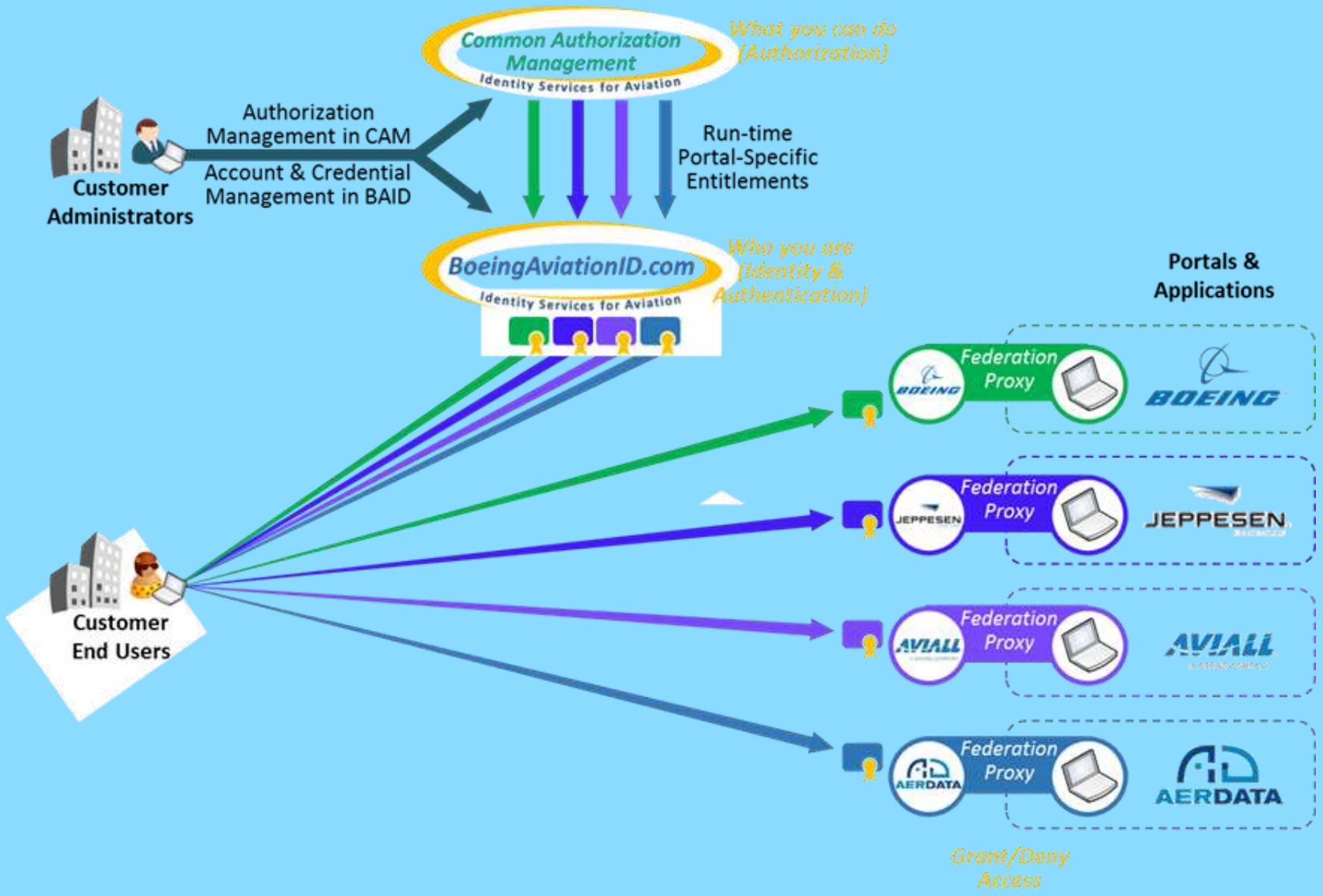
Common Services Mapping to SAL

	Required Common Services		SAL Mapping to CSC
1	Secure Configuration (secure host images, scripted infrastructure)		CSC 3-1, CSC 3-2, CSC 11-1, CSC 11-2
2	System Patching Tool/Process (secure connectivity to WSUS/CentOS repositories)		CSC 4-5
3	Secure Remote Access Channels (Admins)		CSC 3-4
4	Malware Defenses (Anti-Malware)		CSC 8-1, CSC 8 Supplemental
5	Perimeter Defenses		CSC 12-1, CSC 12-8, CSC 9-6, CSC 18-2, CSC 12 Supplemental
	a.	Firewall Services	CSC 12-1, CSC 12-8
	b.	Reverse Proxy	CSC 12-5
	c.	Portal	CSC 12-5
	d.	Forward Proxy (NAT)	CSC 13-7
	e.	Load Balancers	CSC 12-5
	f.	Web Application Firewalls (WAF)	CSC 9-6
	g.	API/Services Gateway	CSC 12-5
	h.	Secure SMTP relay services (outbound)	CSC 9-5
6	Secure Administration (2FA)		CSC 5-1, CSC 5-2, CSC 5-3, CSC 5-4, CSC 5-6, CSC 5-8, CSC 11-4, CSC 16-11
7	Back up & restore services		CSC 10-1, CSC 10-2, CSC 10-3
8	Central Logging & Alerting Service		CSC 6-1, CSC 6-2, CSC 6-3, CSC 6-4, CSC 6-5, CSC 6-6
9	Incident Response/Management Process (ticketing)		CSC 6-6, CSC 19-4
10	Secure Data (at rest & in transit)		CSC 13-2, CSC 14-2, CSC 16-14
11	Federation (SAML)		CSC 16 Supplemental
12	Centralized Identity Management System (authN and authZ - RBAC)		CSC 16-1, CSC 16-4, CSC 16-5, CSC 16-6, CSC 16-7, CSC 16-9
13	Network Isolation/Segmentation		CSC 2-4, CSC 14-1
14	Intruder detection		CSC 3-1, CSC 12-3, CSC9-2
	a.	Host	CSC 3-1
	b.	Network	CSC 12-3
	c.	Host Scanning (Limit Ports)	CSC 9-2
15	PKI / Certificate Management		Supplemental
16	Code Mgmt Service		CSC 18
	a.	Continuous Integration / Continuous Deployment (CI/CD) (Jenkins, Artifactory, GIT and Orchestration tool like Continuum)	CSC 18
17	Development Tools		CSC 18
	a.	Config Mgmt, Provisioning (Chef, Cloud Foundry, etc.)	CSC 18
18	NTP		CSC 6-1, CSC 9-5
19	Secure File Transfer Service (includes Bulk)		CSC 14-4
20	Asset Management (software/host/IP and depot servers)		CSC 2-1

Expectations by Security Assurance Levels

	Low	Medium	High	Very High
Host Based Controls				Remove unnecessary srvs Digitally Sign Archive Logs Bi-weekly log anomaly rpts Air gapped systems Automated Patch Mgmt against expected baseline Dedicated Administration Workstation (football) Subcomponent Level Assessments Dedicated & encrypted log file server
			Auto Scan tool report unapproved SW File Integrity Checking Tool Automatic port scanning	HIDS Automated Analytics of Admin Use Web Application Firewall
		Encrypted host drives containing sensitive data Automate Admin Acct Reconciliation Encrypted Database containing sensitive data Disable unused port, protocols & services Encrypted backups for sensitive data System, Security & Audit Log enabled all hosts Logs retained on remote Sync Server	Re-Image compromised Hosts Host Assessment & Mitigations Tested recovery from backups Automatic backups Separate Logical Hosts for critical services (DNS, File, Mail, etc.) Log/alert on Admin Acct Changes Audit Log Policies	Formal Change Mgmt Synchronized clock for DNS Vulnerability Testing Business Continuity Plan
	Approved Software Listing/SW Depot Remote Administration via encrypted protocols Login with Non Privilege User Automated anti-virus/spyware/malware on all hosts	Administration via Bastion/Jump Server Access Control Plan & Review Only use Admin Acct tasks requiring it Patch Mgmt Process		Hardened, Standard Host Images MFA for Admin Access Automated anti-virus signature updates
Network Controls			Document changes to baseline config	Network segmentation
		Formal Change Mgmt Data encrypted in transit to/from infrastructure components		Automated change detection & alerting
	IDS Outbound Filtering Proxy	Mgmt of Network devices with 2 factor authN and encryption Private subnets restricted to DMZ address space		Blacklisting/Whitelisting DDOS protection enabled
Ops/ Alerting/ Govern		Monitor Alert on Admin Acct Changes Insure verbose logging at Boundary devices Monitor Change detection alerting Key Safeguarding Process	Monitoring of System, Security & Audit Logs Cert Alerting & Renewal CA Trust Process	SIEM enabled Incident Response Key Mgmt Process
Apps				Analytics on users typical usage patterns No group accounts permitted
			Accept SAML using 2 Factor AuthN	
		Multi factor AuthN for access to sensitive data Provision accounts using a central ID store Software/Hardware tokens for app admins	Review System Acct & Disable unused Timeout inactive user sessions Blockpoint include scan & CVE checking	Accept SAML Assertions Disable and Revoke accounts Penetration Testing
	Assessment of data to determine sensitivity Password files encrypted & restricted	Monitor and disable dormant accounts Complex passwords		Lock out account after exceed failures reached

Enhanced Access Control – Identity Management



Mobile Reference Architecture

Return

Mobile Reference Architecture Summary

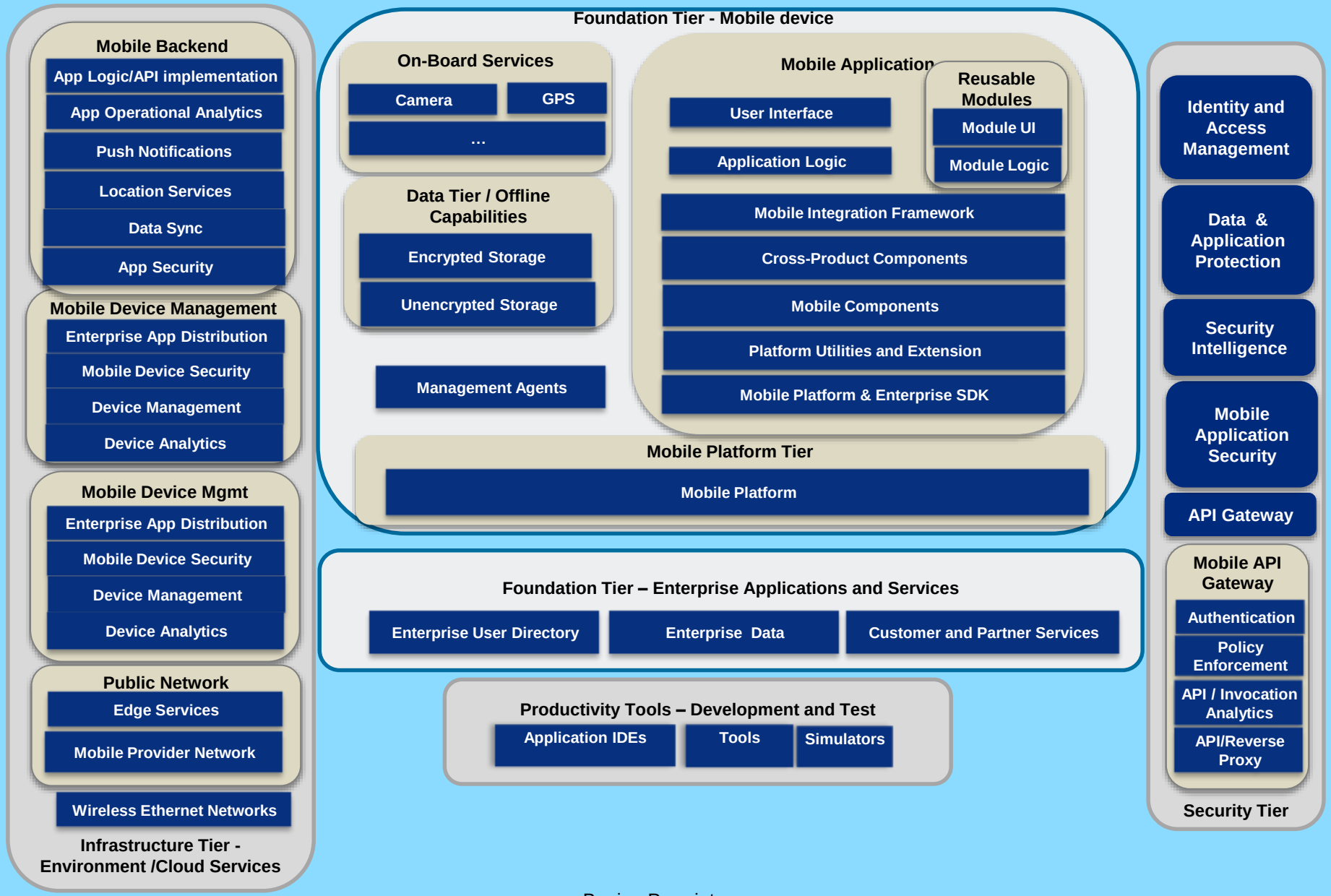
Description: Mobile Reference Architecture defines design guidance for creating mobile applications that will meet Digital Aviation guidelines for cloud or data center hosting and integration with Enterprise resources.

Highlights:

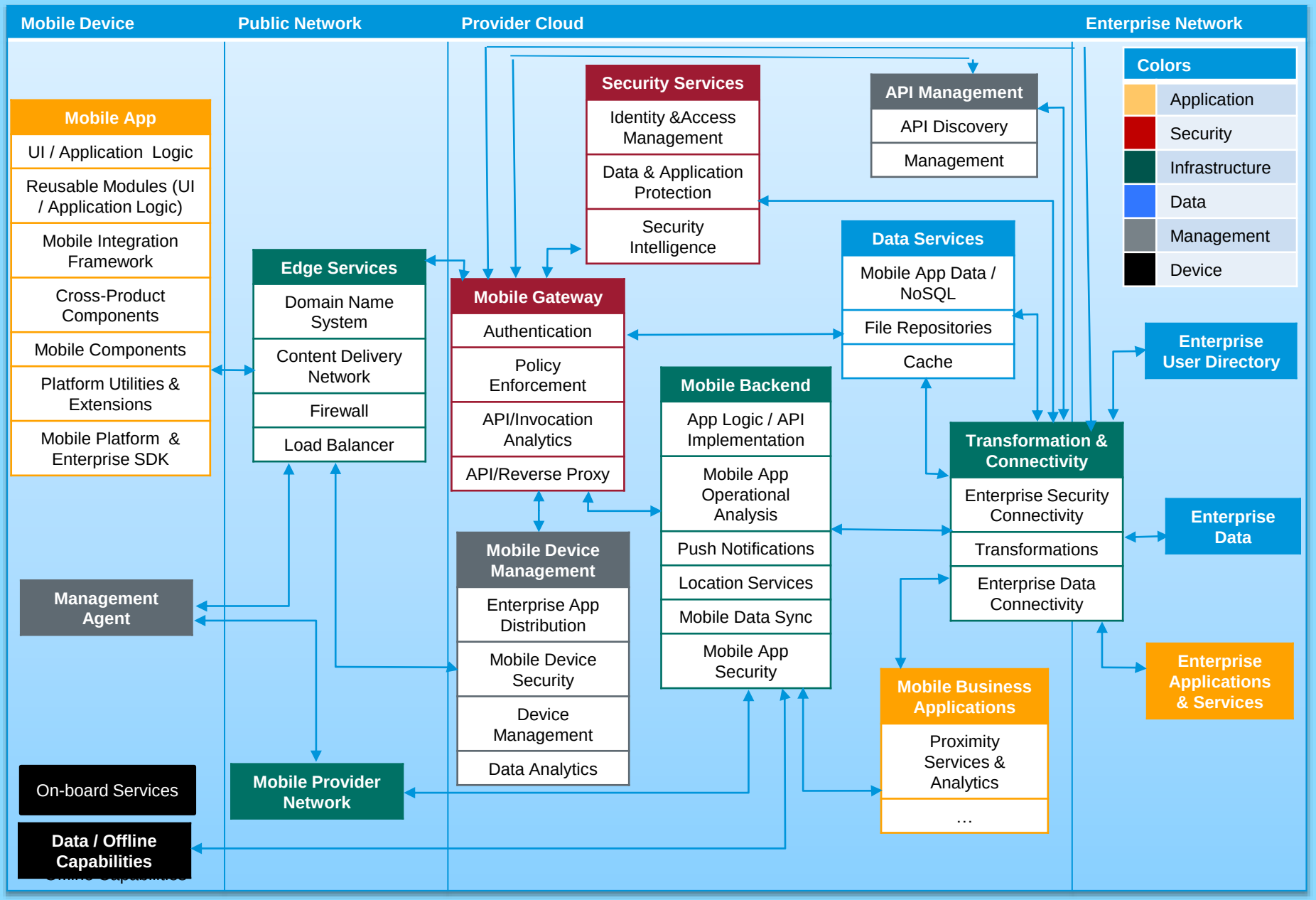
- Cover mobile application development on multiple platforms (OS) such as iOS, Android, Windows
- Define principals of UI/UX development for both Native and Browser-based applications
- Satisfy security requirements on all levels: Mobile Device authentication and Data Storage, protected and authorized access to enterprise resources, provisioning
- Comply with different multiple enterprise environments: internal data centers and cloud (Azure, AWS, Helion)
- Provide reliable connections and traffic support through cellular and wireless channels
- Review and recommend productivity tools, i.e. MobXP

  MobXP is a toolset which is already widely used on multiple projects and has demonstrated ability to significantly improve productivity. It concentrates on Mobile Specific Capabilities to support Source Code Access, Build, Unit and Compliance Testing, Signing, Release Management and Deployment...

Mobile Reference Architecture – Static Tier View



Mobile Reference Architecture – Environment/Allocation View



Mobile Reference Architecture - Component Definitions

Reusable Application Modules encapsulate and package UI and application logic in order to permit their reuse across applications on the same platform.

Mobile Integration Framework (MIF) facilitates the module and component integration within the application to increase ability of constructing the applications from “large building blocks”. It may help to envision the MIF as a very thin and simple bus (ESB).

Cross Product Components - (sometimes referenced as Cross-Platform components) implement core DA capabilities and make them available to a wide range of products. These products include mobile and desktop systems, server, and the real-time operating systems used by OEMs who incorporate Digital Aviation data and services into their products. Cross-Product (Platform) components enable business capabilities across different markets and devices. Examples of these components are the Terminal Charts and Common Content Delivery Library.

Mobile Components capture business logic and functions for mobile platforms only. There are 2 types of implementation:

- **Native** – written in a native language and potentially reusable across all applications on the same platform.
- **MADP** (Mobile Application Development Platform such as Xamarin or Crdova) based. These components are written in a “neutral” language and reusable across all applications developed using the MADP. An advantage of this approach is leveraging the common programming model being used on the applications.

Mobile Reference Architecture - Component Definitions cont.

Platform Utilities and Extensions - APIs and libraries that extend the functionality of the platform for the application and components. They support reuse and standardization of DA and 3rd party libraries.

Examples:

- Adobe Mobile SDK for analytics
- OpenSSL for content delivery certificate management
- Boost geospatial libraries
- eXtremeDB for cross-platform databases

Extensions will generally be cross-platform. Platform utilities will generally be native.

Mobile Platform and Enterprise SDKs - represent the programming interfaces and libraries made available to applications.

Examples of SDKs:

- IP Networking, Bluetooth, Multi-peer connect (sharing and service access)
- Downloading, background downloading (content delivery)
- Keychain, Password Vault (registrations)
- Notifications (application)

Mobile Reference Architecture - Component Definitions cont.

Edge Services - services needed to connect the mobile device and its apps to the right mobile gateway through the Internet using Wi-Fi or mobile provider networks.

Mobile Provider Network – provider of wireless communications.

Mobile Gateway - usually a single entry point for mobile apps; Services: authenticate, enforce (security) policies, capture API analytics, provide API reverse proxy capabilities to the backend

Mobile Backend provides runtime services including operational analytics, push notifications, location services, data synch from the offline storage to the data service, app security (in addition to the gateway security), i.e., detect the app/app store tampering.

Mobile Device Management includes: app distribution (if not provided through app store, device security, device management (i.e, install policies, wipe out the device if lost or stolen), capture device analytics

Mobile Business Applications - industry specific capabilities, proximity service, campaign management, business analytics, workflow support

API Management - manage service catalog, advertise available services endpoints, provide keys to access APIs, API versioning, …

Data Services - mobile app data/NoSQL (JSON), file repositories, cache (store data so client requests can be served without going to the enterprise backend)

Mobile Reference Architecture - Component Definitions cont.

Security services (defined in depth in the Security domain section):

ID and Access management to provide access to cloud resources,

App and Data Protection enables security as part of the development, delivery and execution of mobile apps.

Security Intelligence helps to detect and defend against threats through analysis of events and logs and correlation.

Transformation and Connectivity: Enables secure connection to enterprise systems and the ability to filter, aggregate, or modify data

Enterprise User Directory: Provides storage for and access to user security information

Enterprise Data: One or more systems of record, i.e., transactional data or data warehouses

Enterprise Applications and Services: Enterprise business processes and logic. This tier may also include other "ground" services from the Customer, Partner, or Manufacturer

Mobile Reference Architecture - MobXP Overview

Federated Source Management

Perforce

GIT-fusion

Configuration managed code transfer from off-shore teams

Create build

Distributed Development

A federated source control repository allows teams to share code, common components, and UX artifacts

Bamboo – continuous integration Env.

Cross-platform IDE, called Xamarin, allows for significant code reuse across platforms (iOS, Win, Android)

Integrated build environment allows for automated check-in, builds and auto-signing of Enterprise builds

Automated Upload to Mobile Gallery

Automated Upload to AirWatch , TestFlight

With Cross-Platform & eCom Projects

Project Planning

Automatic Task and Defect tracking improves project statusing

TestFlight

airwatch

Mobile Gallery

Administered installs

Role based access to each type of build

VersionOne or Jira

Defects and Enhancements

Automated Testing reduces time and resources required, improves quality (Eggplant, Monkeytalk, Perfecto Mobile, Adobe Analytics …)

Test

Unit, Automated, Functional, Performance

Dev/Test

Demo

Prod

Automated Testing Tools

User Interface/User Experience Reference Architecture

Return

User Experience and User Interface Reference Architecture Summary

Description: User Experience and User Interface Reference Architecture provides guidance on how UI design should be implemented. The Digital Aviation initiative “Digital Experience” provides important guidance as well as templates to support this effort.

Assumptions:

1. Digital Aviation will retain responsibility for business and user requirements
2. Development Teams (internal, external or hybrid) will be responsible for creating products including User Interface

Guidance Materials available for Development Teams:

1. User interface design standards for web and mobile applications
2. Standard components intended for generation of consistent user interfaces such as Bootstrap templates and style sheets based on the Digital Experience guidelines and processes
3. Wireframe drawings showing approximate layout and work processes
4. Other requirements and performance indicators as necessary and appropriate

User Experience and User Interface Reference Architecture

Boeing/Digital Aviation

- Understanding users, tasks, and environment
- Understanding competitive environment relevant to UX
- Regulatory constraints on product UX
- Defining task process flow of proposed product
- Defining how task process flow of proposed product integrates with other Boeing products
- Basic interaction design
- Low fidelity mockups and prototypes
- Definition of human in the loop KPI and user stories
- User Validation
- Product acceptance testing
- Behavioral analytics
- In service tests
- UX research

Common Tools

- UI Standards
- Brand management
- Bootstrap templates
- Style sheet
- Content management system

Governance

- Compliance reviews
- Revisions to standard
- Periodic update to templates, style sheets, and standards

Development Team

- Implementation of standards
- Implementation of requirements
- Development of technical system design
- Development of user interface design.
- Usability testing

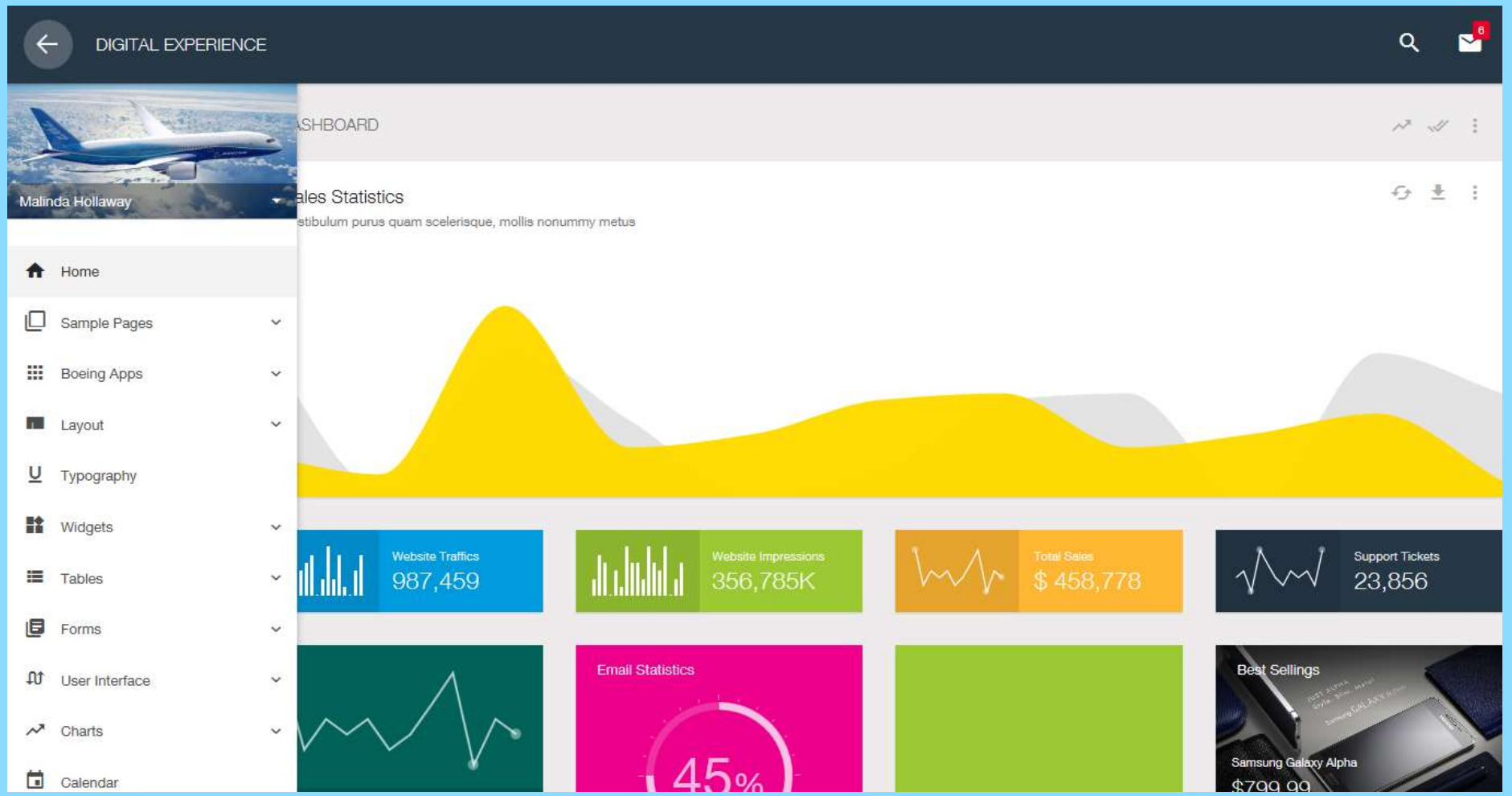
Re-Host/Re-Factor Categories and Effort Required

Software Architecture Transition Type	UI Transition Type	User Interface Design Effort	User Experience Design Effort	Comments
1. Lift and Shift: Software is moved to cloud with little if any change	**No change to application UI**	**User interface is not changed**	**User experience is not changed**	**Some work may be required to evaluate changes in things like response time.**
2. Remodel: Existing application components are modified to integrate with current	**A. Back end elements are changed to take advantage of cloud capabilities but UI is not changed**	**User interface is not changed**	**User experience is not changed**	**Some work may be required to evaluate changes in things like response time.**
	B. Functionality is exactly the same as in current application.	**Simple re-skin. UI work to choose new widgets, colors, etc.**	**Some work required to ensure that changes improve or at least do not degrade user experience.**	**While the intent may be to do a simple re-skin, project discipline will be required to limit scope creep. For example, pressure will be intense to add things like responsive design.**
	C. Functionality remains the same, but some changes are made to take advantage of current technology; e.g., add responsive design for mobile web application.	**Re-skin with technology updates. Implement new UI capabilities using current design standards.**	**Understanding of users and tasks required to implement technical changes in a way that best supports users.**	**Again, project discipline will be required to prevent scope creep. Frequently, programs will want to make changes to address problems reported by customers.**
3. Rebuild: Application is completely redesigned	**A. New system is largely redesigned to provide same functionality as existing system. Some adaptations are made to leverage new technology. Other changes realize high-value opportunities.**	**Execute new UI capabilities and new functions using current design standards.**	**Understanding of users and tasks required to implement technical changes in a way that best supports users.**	**This is only slightly less work than a full redesign (3B). Still need to do all the same work to understand the users and their tasks and what the design needs to do to support them.**
	B. Redesigned to completely new requirements.	**Execute new UI capabilities and new functions using current design standards.**	**Full system design process required. As with any other full re-design, the new system will behave very differently than prior system. UX support is needed to ensure it meets users' needs.**	

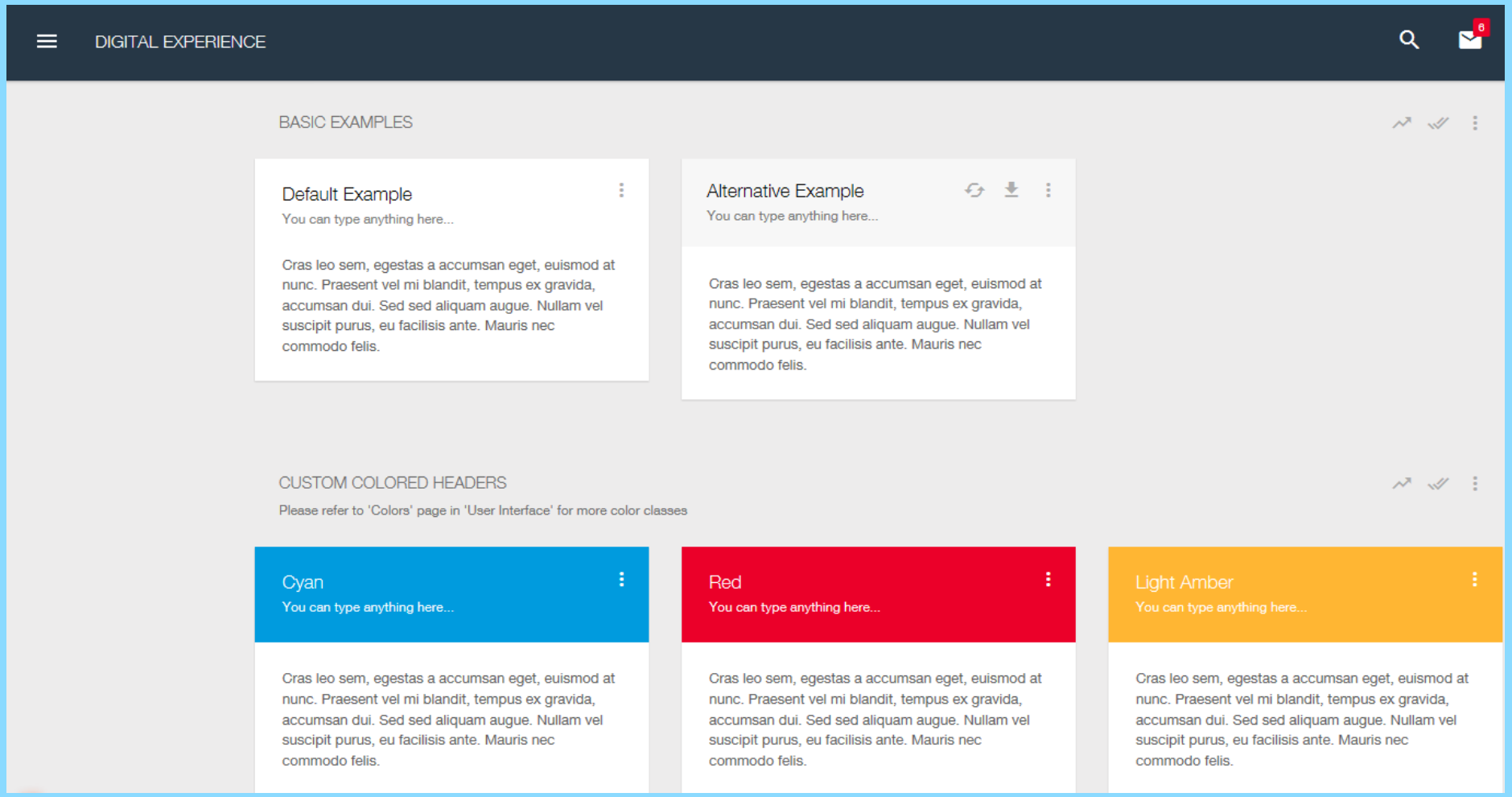
Boeing Digital Experience Website (Boeing Internal)

http://design.web.boeing.com/digitalexperience/#/home

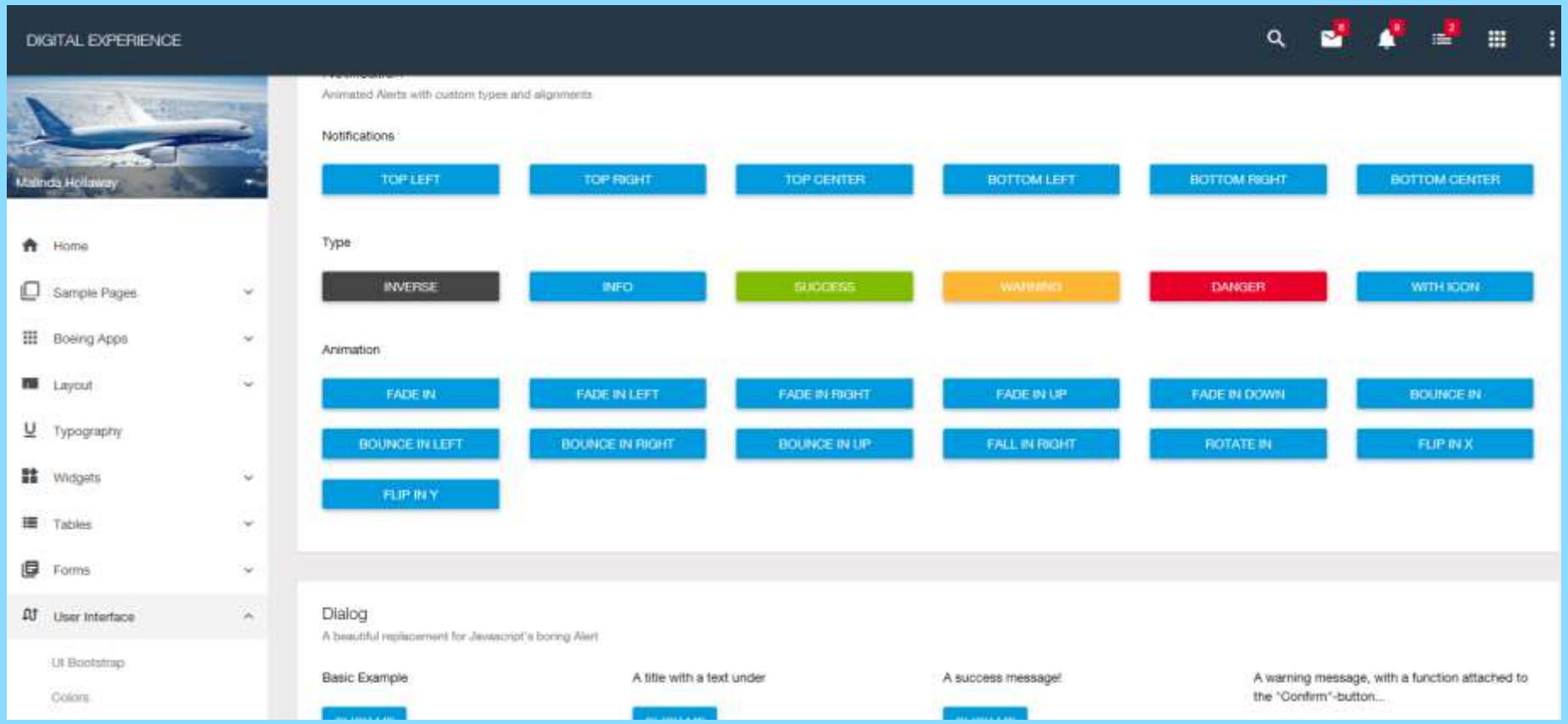
UI/UX Development Guidance Example



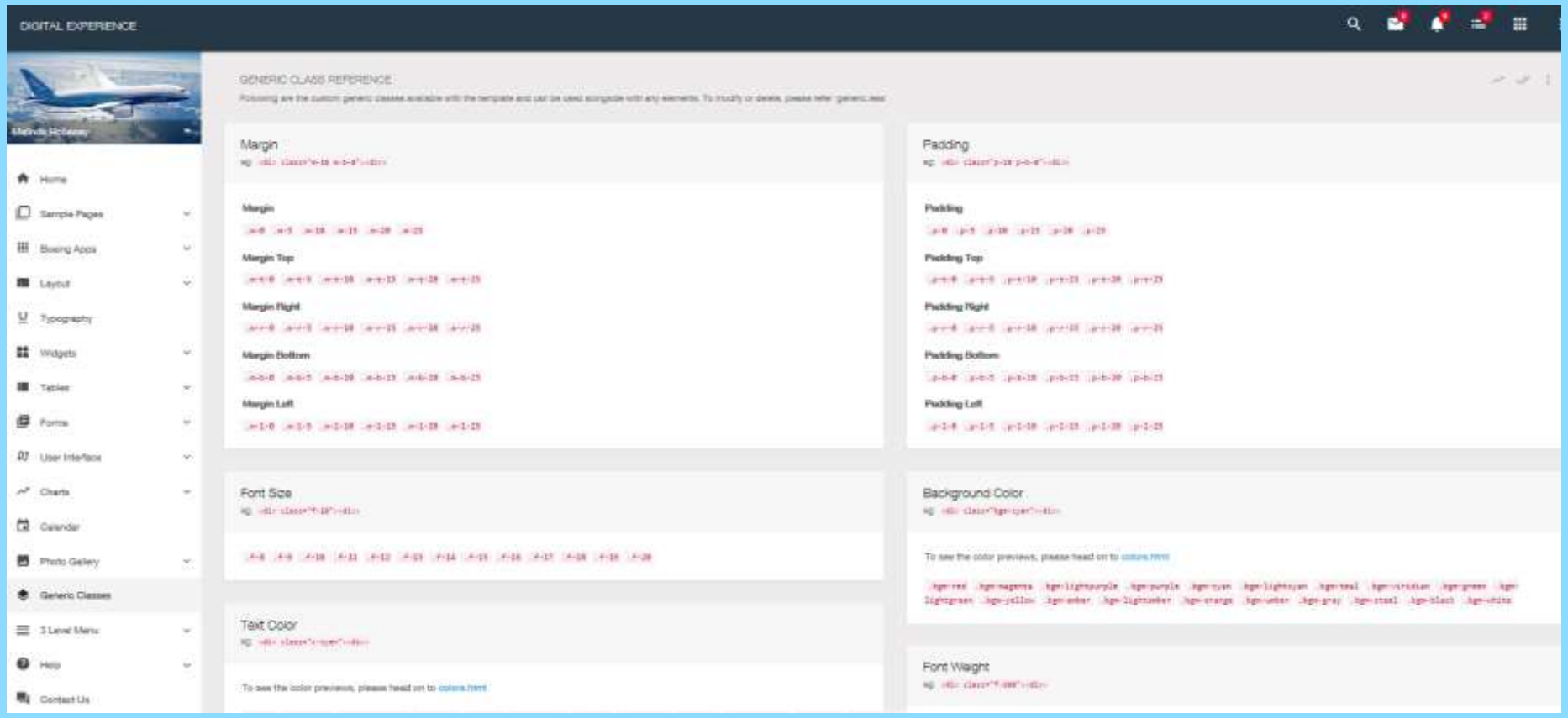
UI/UX Development Guidance Example



UI/UX Development Guidance Example



UI/UX Development Guidance Example



Product-specific User Interface Standards Example (Boeing Internal)

STYLEGUIDE

TOOLBOX STYLEGUIDE

COLORS

The Toolbox color palettes were selected from the Boeing-approved color palette to invoke a sense of stability and reliability. Both of these traits are particularly important when it comes to maintenance data.

How Toolbox Uses Color

The use of colors in design can often make or break the design. That said, it is important to the Toolbox UX Team that we are consistent with the Boeing brand, while designing a modern, compelling user experience.

USAGE :
The color classes can be used on `<div>,<p> and <span>` elements. Use the classname defined below each color, in the class attribute for the elements

```
<div class="bg-steel"></div> OR <span class="bg-fire"></span> OR <p class="bg-magenta"></p>
```

CLASSNAME : bg-black

CLASSNAME : bg-steel

CLASSNAME : bg-umber

CLASSNAME : bg-fire

CLASSNAME : bg-magenta

CLASSNAME : bg-red

CLASSNAME : bg-orange

CLASSNAME : bg-amber

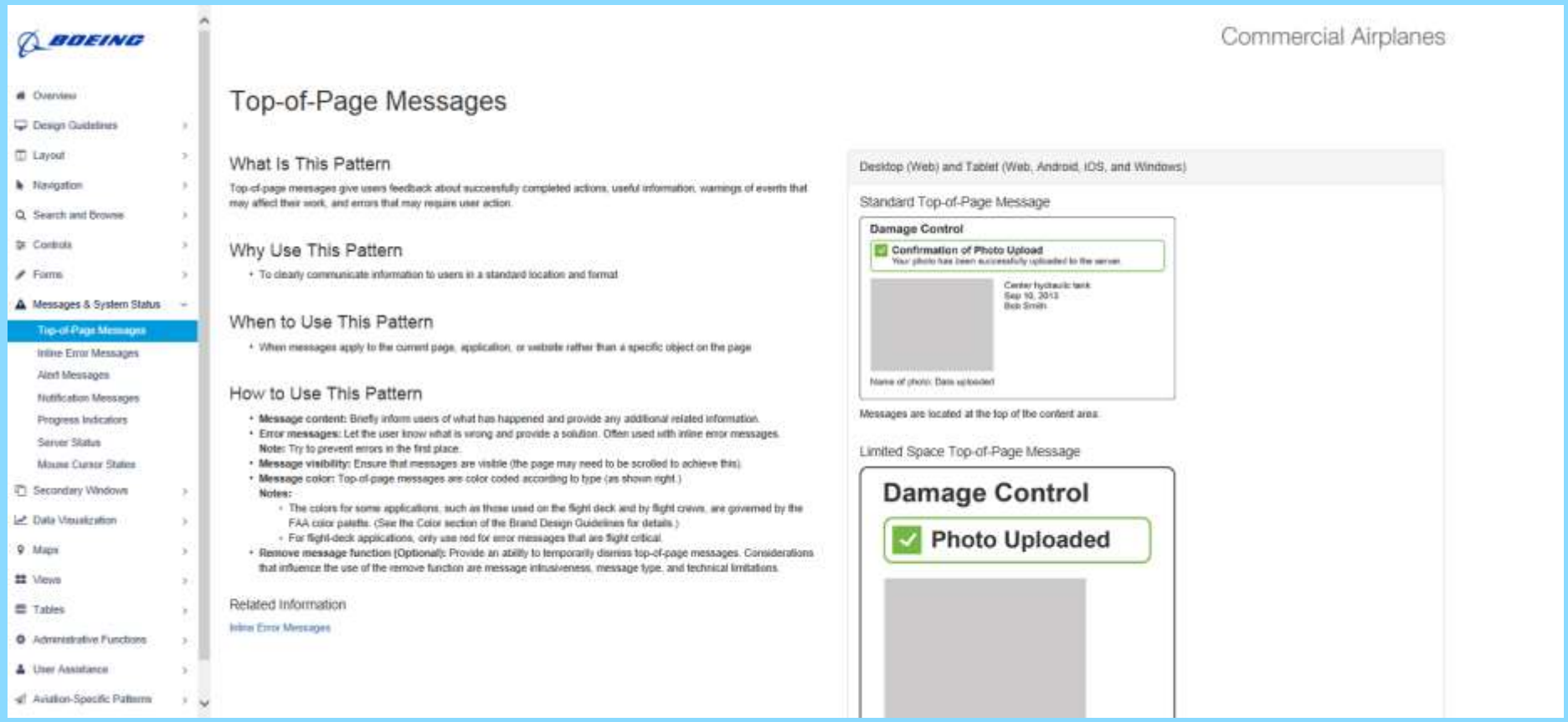
CLASSNAME : bg-yellow

Boeing CAS User Interface Standards (Boeing Internal)

http://csweb01.nw.nos.boeing.com/csdev/barrett/uui/#/landing

User Interface Standards Example

User Interface Standards Example



User Interface Standards Example

BOEING

Commercial Airplanes

- Overview
- Design Guidelines
- Layout
- Navigation
- Search and Browse
- Controls
- Forms
- Messages & System Status
- Secondary Windows
  - Modal Window
  - Modeless Window
  - Transitory Window
- Data Visualization
- Maps
- Views
- Tables
- Administrative Functions
- User Assistance
- Aviation-Specific Patterns
- Additional Patterns

Transitory Window

What Is This Pattern

Transitory windows are used to display information that may be related to the information displayed on the main screen. Users can interact with the main screen while the transitory window is open, but the transitory window may close.

Why Use This Pattern

- To display temporary information (e.g., alerts and notifications)
- To quickly communicate or obtain information using limited space

When to Use This Pattern

- When supplementary information needs to be displayed

How to Use This Pattern

- **Opening the window:** Display when the user clicks, hovers, or taps a link, button, or image to open an in-context window.
- **Resizing the window:** Do not allow users to resize transitory windows.
- **Hover:** Do not dismiss the transitory window when the cursor is hovering over it.
- **Close the window:**
  - When the user clicks outside the window.
  - When the user interacts with the window, that is, makes a selection.
  - When the user presses the Escape or Back key.
  - After a specified amount of time.
- **Timed window dismissal:** Dismiss the window after 3 seconds. Dismiss it after 5 seconds if the transitory window is at the edge of the screen, the message contains more than a few words, or it includes a link.

  Note: Do not dismiss windows that the cursor is hovering over.
- **Audible cues:** Play an audio cue just before the secondary window is displayed.

Related Information

Modal Window, Modeless Window

Desktop and Tablet (Web)

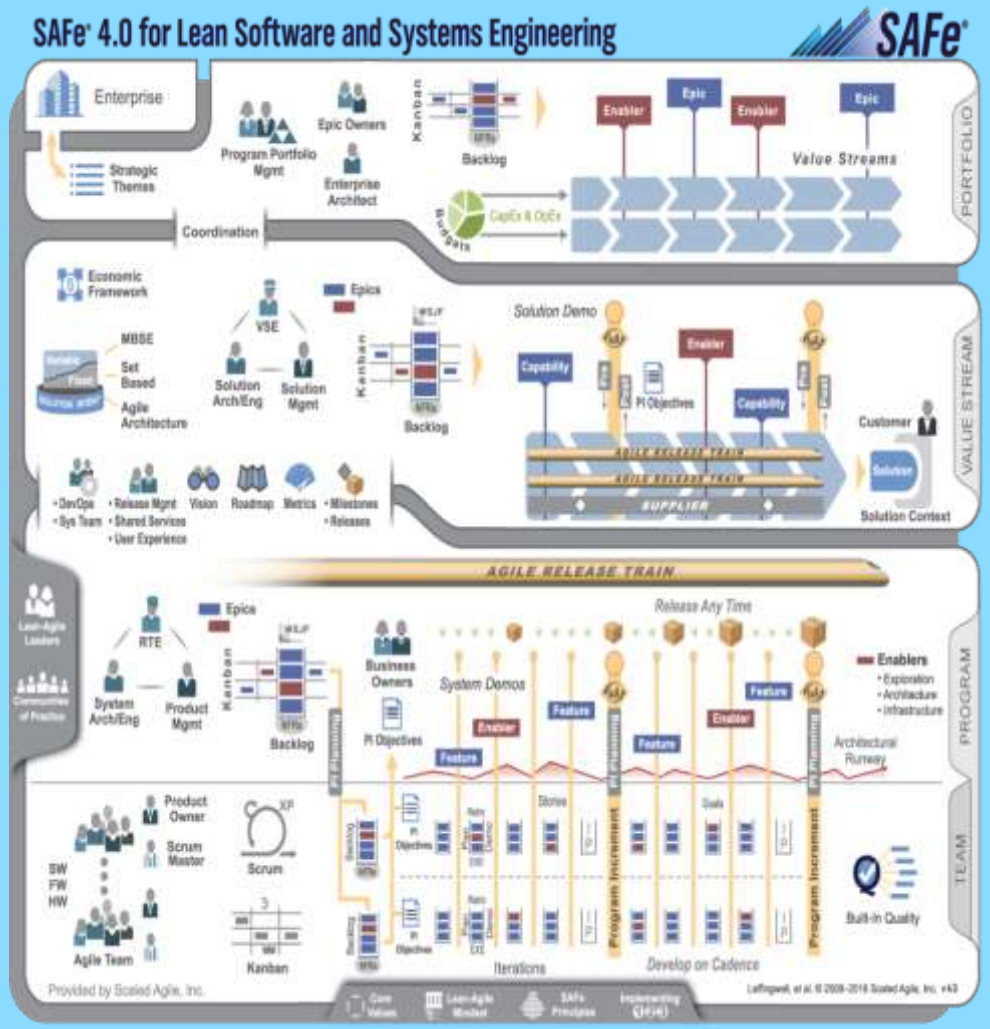
Tablet (iOS)

Development Process/Scaled Agile Framework (SAFe)

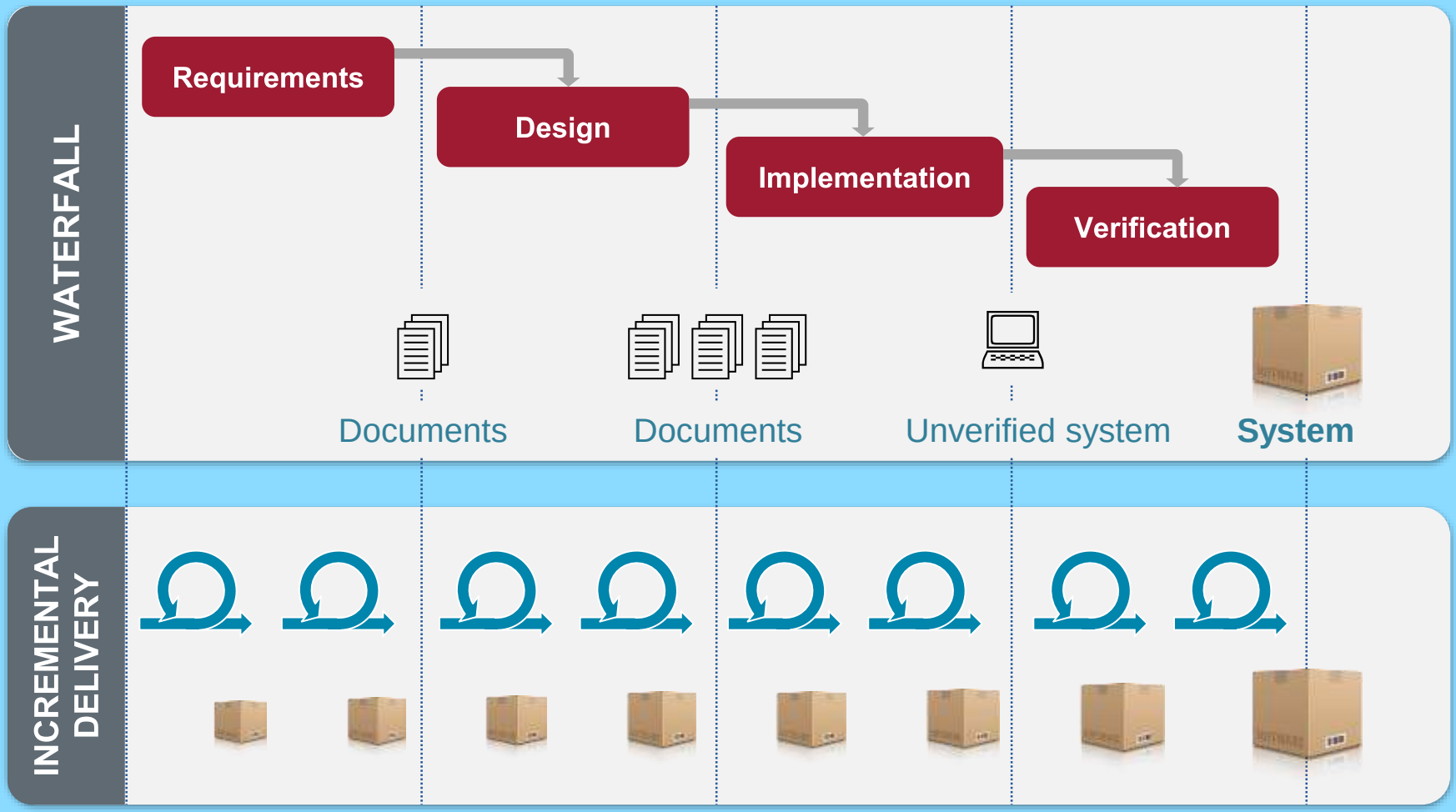
Return

Software Development Process Guidance

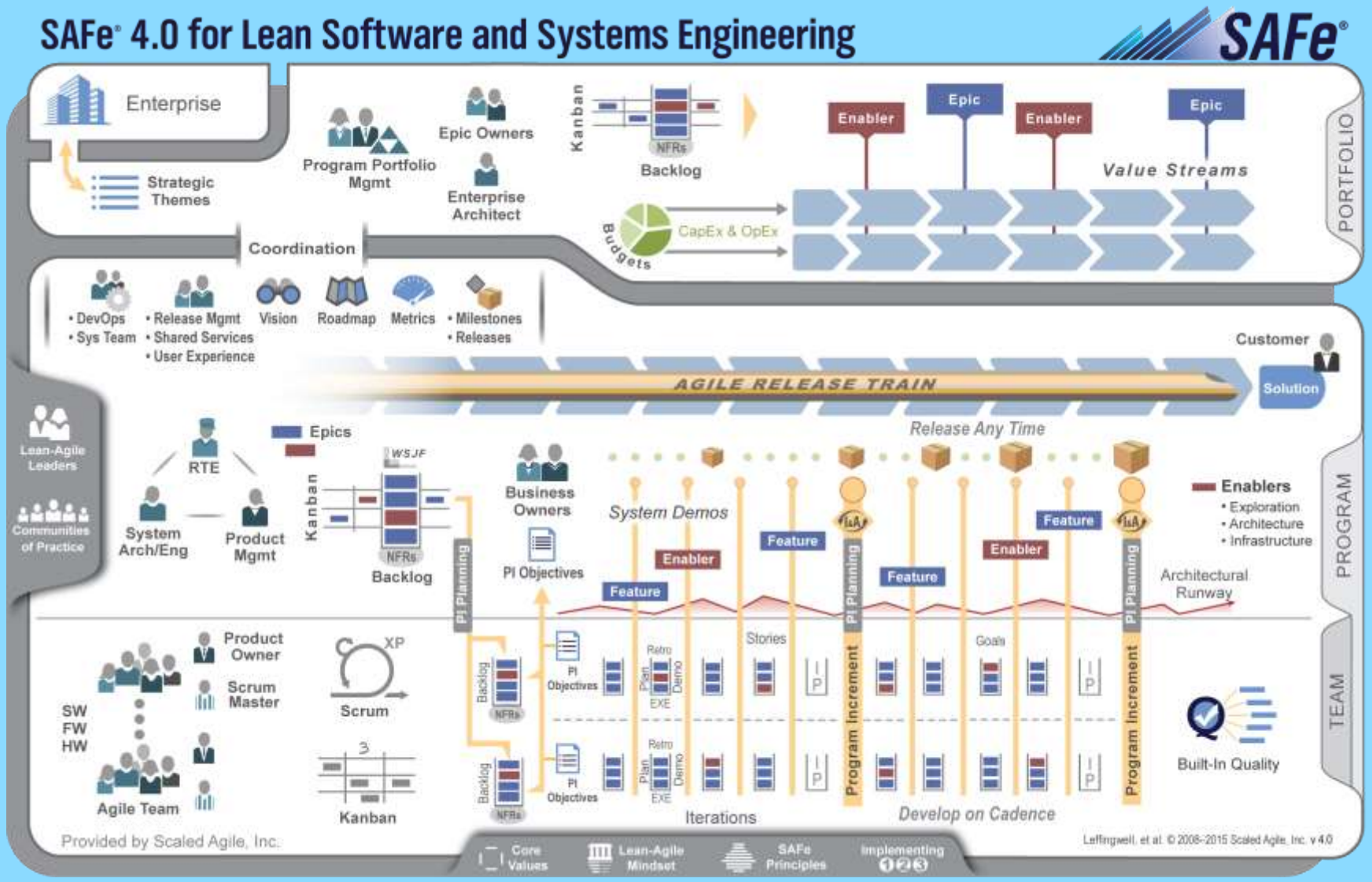
- Digital Aviation Standard Development Teams utilize the Scaled Agile Framework (SAFe)
  - Technical practices (Agile Architecture, DevOps, Model Based Systems Engineering)
  - Alignment with Pragmatic Marketing
  - Influence evolution of SLiM
- Software Development is aligned with Digital Aviation‘s Program and Portfolio Management through the Portfolio and Value Stream provisions within SAFe
  - Lean-agile budgeting
  - Prioritizing portfolio backlog
- Key areas of focus:
  - Program Portfolio Management
  - Alignment with Digital Solutions
  - Technical practices



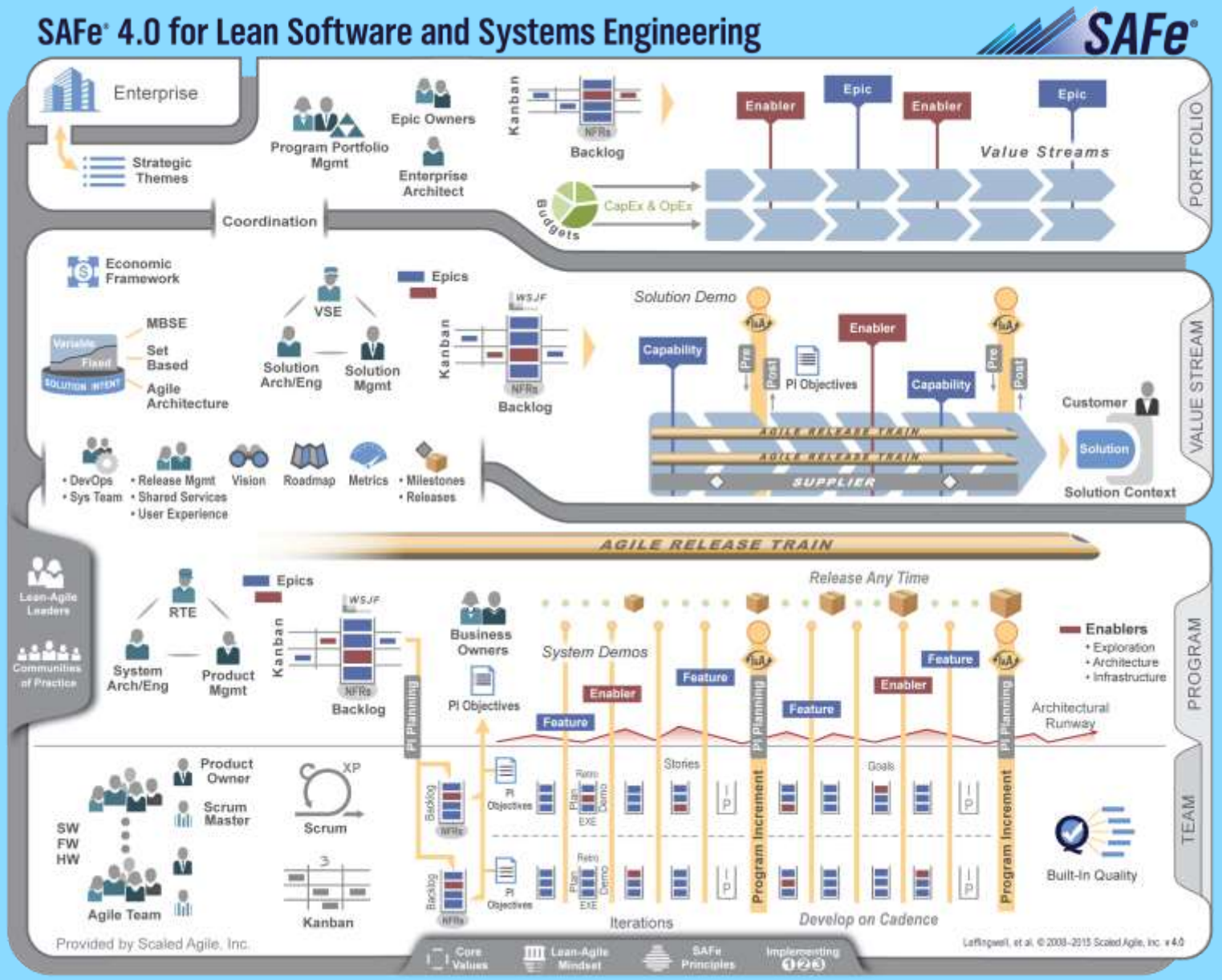
Build Incrementally to Accelerate Value Delivery



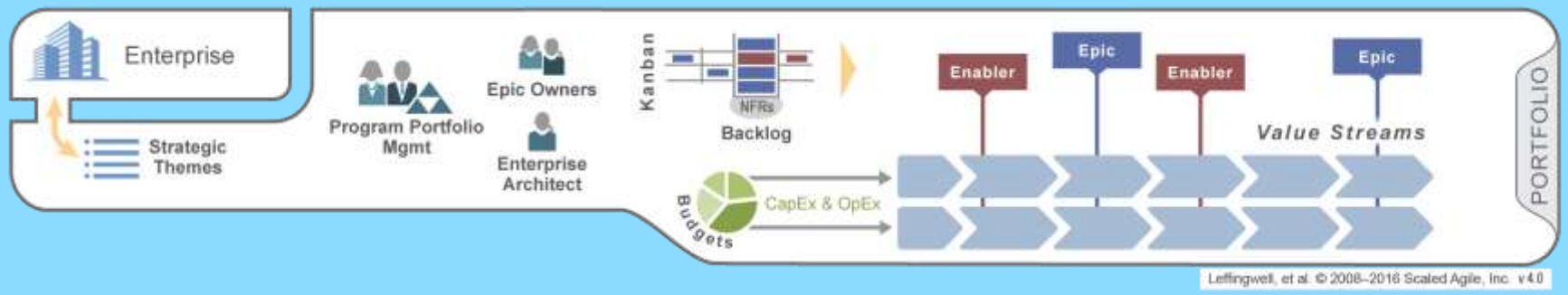
SAFe for midsize solutions



SAFe for large solutions



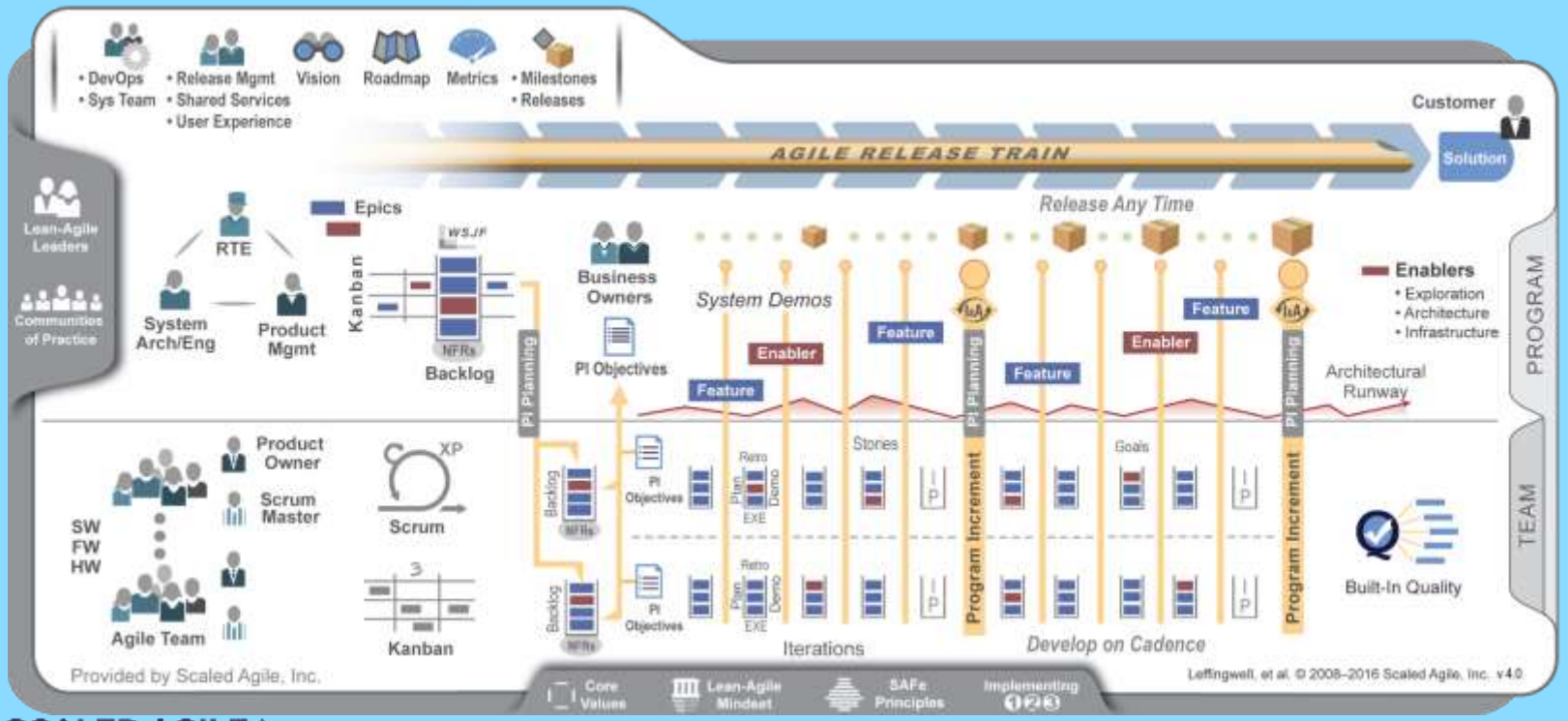
Organize the Portfolio around value



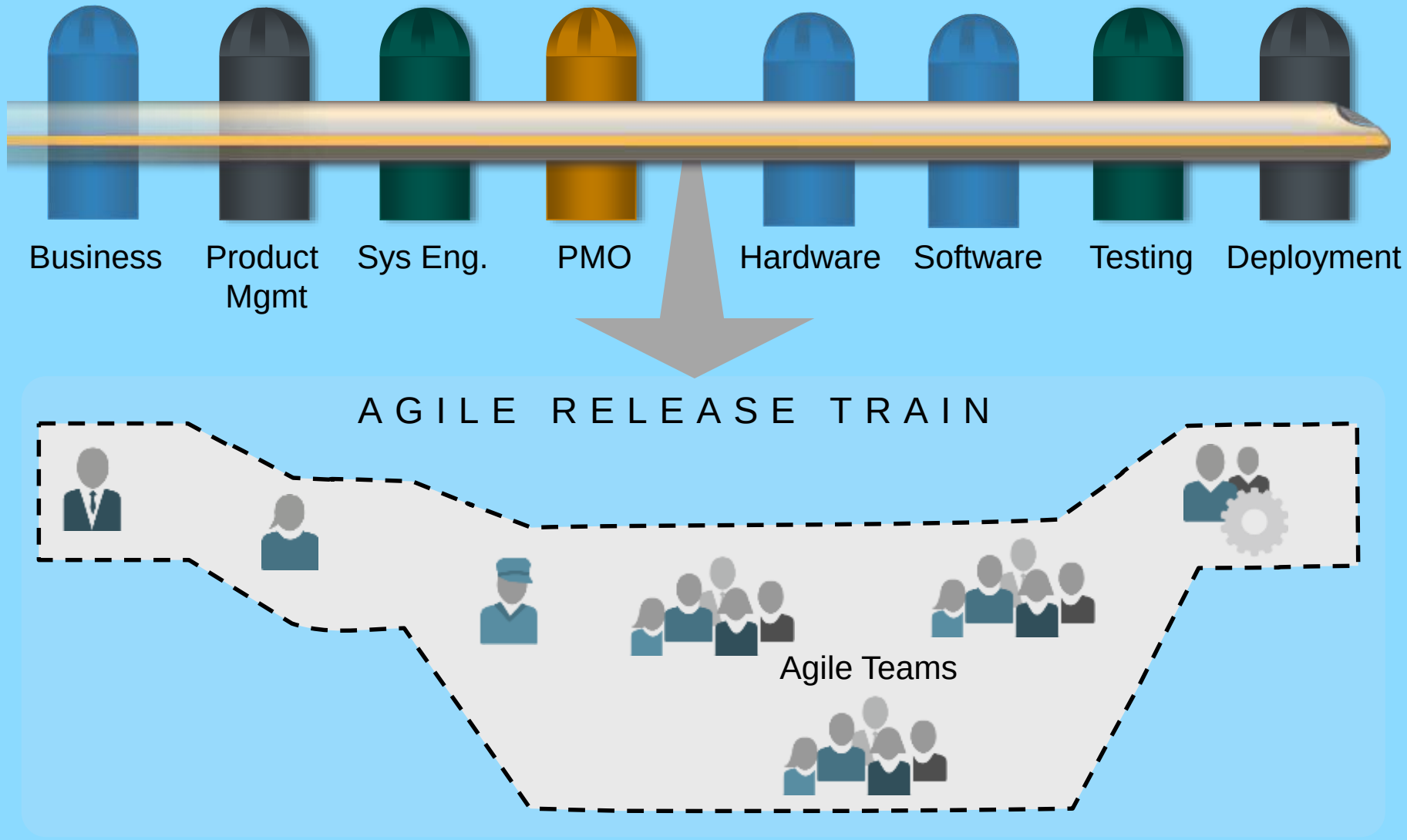
- Identify and organize around Value Streams
- Communicate enterprise strategy with Strategic Themes
- Empower decision makers with Lean-Agile Budgeting
- Provide visibility and governance to cross-cutting initiatives with Kanban

Build value with Agile Release Trains

- Align to a common mission
- Apply cadence and synchronization
- Communicate with Vision, Roadmap, architectural guidance
- Apply objective Milestones to measure progress



The ART “takes a systems view”



Agile Teams Power the Train

Five to nine team members, three roles

Scrum Master

- Runs team meetings, drives Agile behavior
- Removes impediments, protects the team from outside influence
- Attends Scrum of Scrum meetings

Product Owner

- Defines and accepts stories
- Acts as the customer for developer questions
- Works with product management to plan PIs

Agile Team

- Creates and refines user stories and acceptance criteria
- Defines/builds/tests/delivers Stories
- Develops and commits to Team PI Objectives and Iteration plans

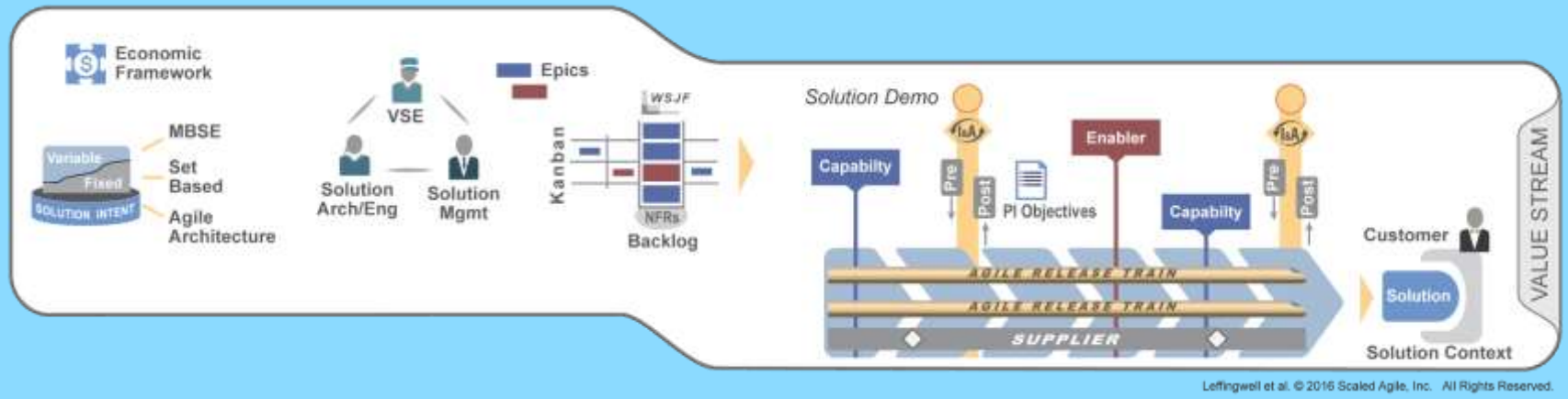
Synchronize with PI Planning

Future product development tasks can't be pre-determined. Distribute planning and control to those who can understand and react to the end results.

— *Michael Kennedy,* Product Development for the Lean Enterprise

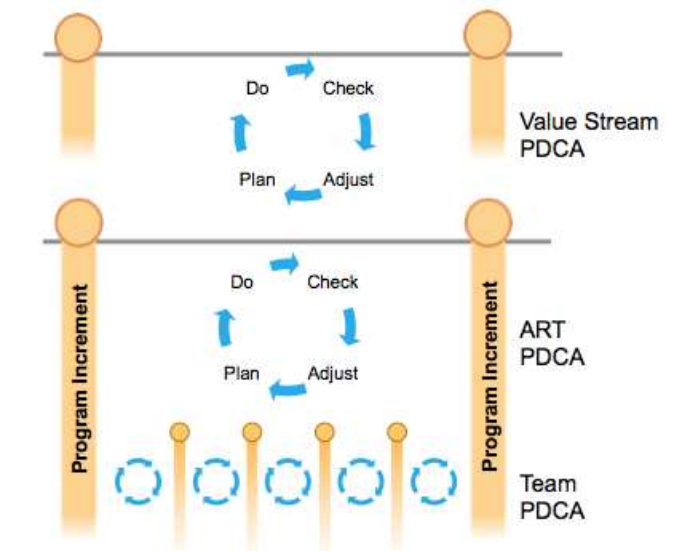
- All stakeholders face-to-face (but typically multiple locations)
- Management sets the mission, with minimum possible constraints
- Requirements and design emerge
- Important stakeholder decisions are accelerated
- Teams create—and take responsibilty for—plans

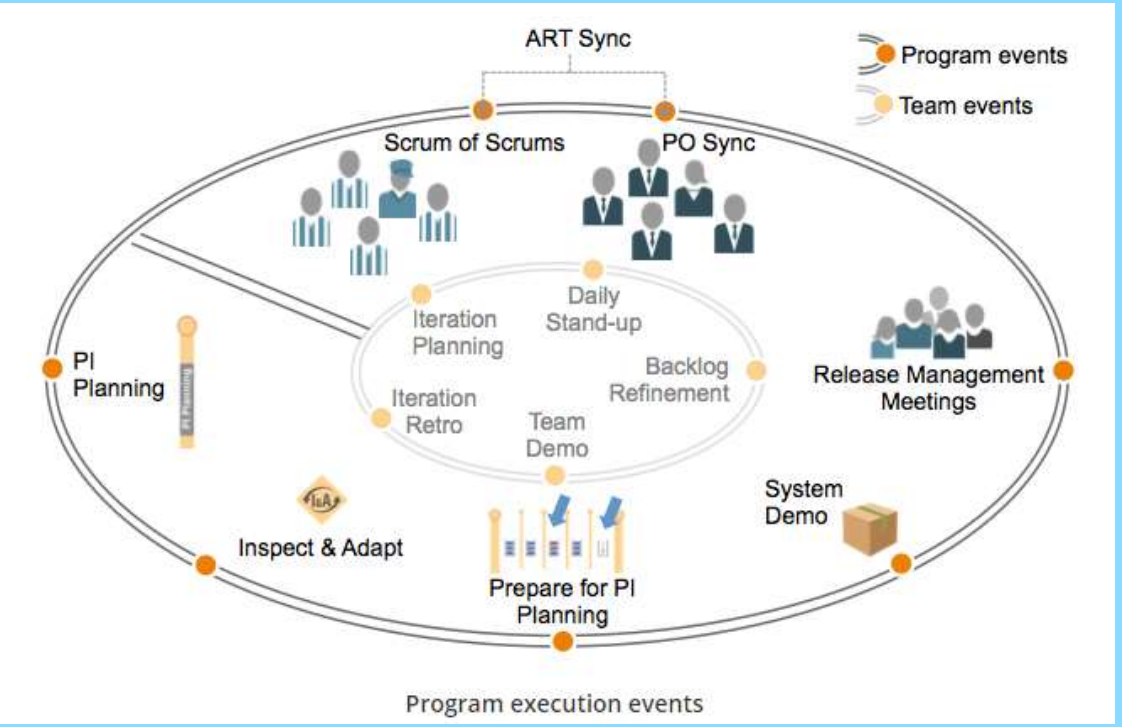
Coordinate large Value Streams



- Apply cadence and synchronization
- Establish governance with Value Stream roles and an Economic Framework
- Manage fixed and variable Solution Intent
- Manage the flow of Capabilities with the Value Stream Kanban
- Frequently integrate and validate Customer Solutions

SAFe Implementation

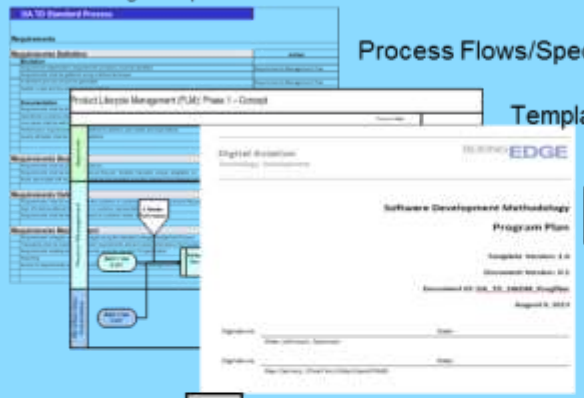




Digital Aviation Software Development Process

Constraints: CMMI, PMBP, ISO, etc.

Process Principles
- Industry best practices
- Boeing best practices



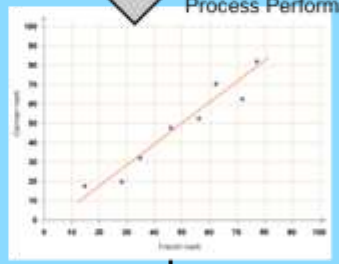
Process Flows/Specs

Templates/Examples

Implemented by

Accessed by

Development execution measured by Process Performance Metrics



Feedback

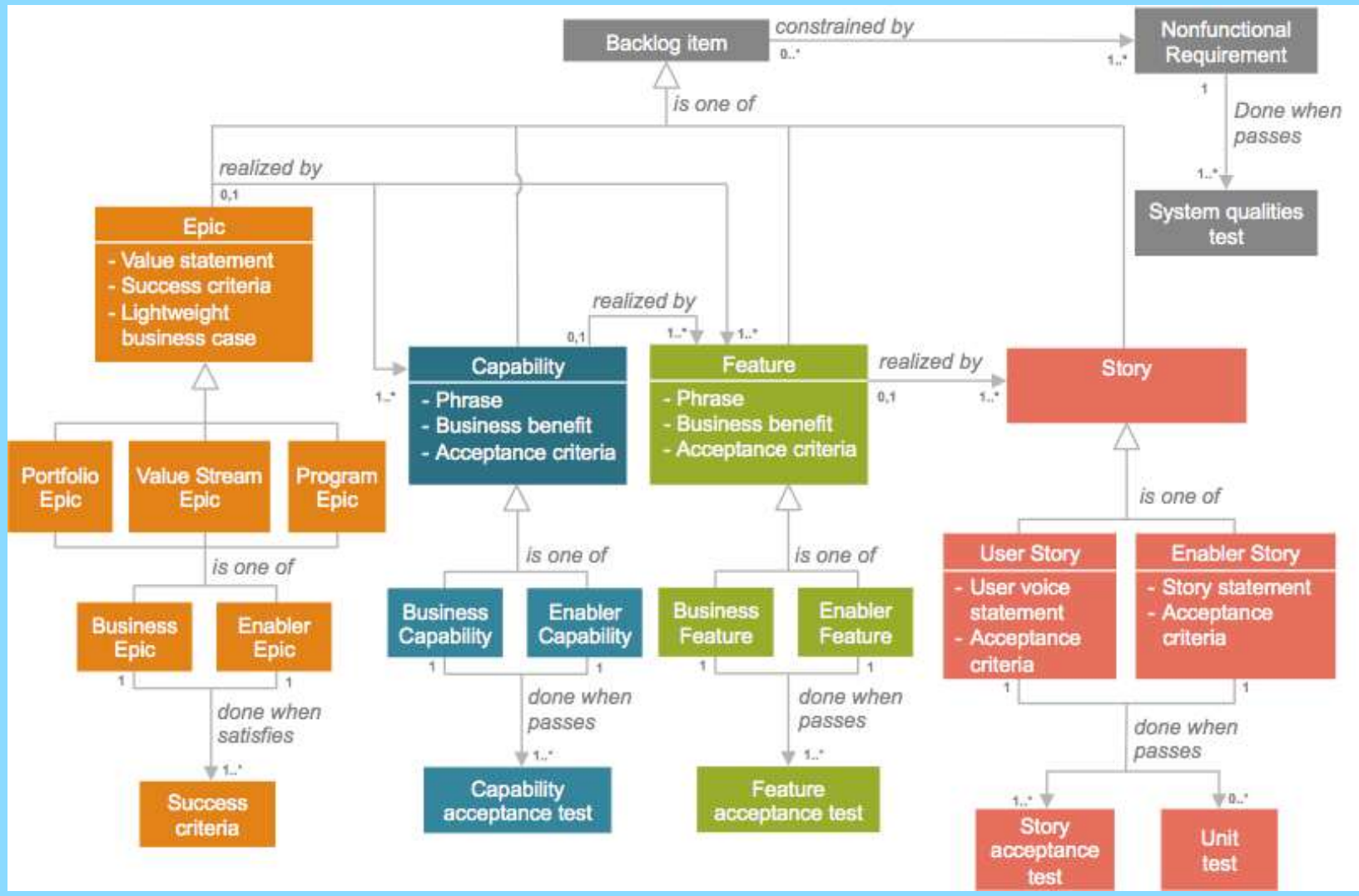
Process maturity measured by

Supported By

REQS	ARCH	CM	BUILD	TEST MGT	TEST AUTO
Contour	Enterprise Architect	JIRA Perforce	Bamboo	QualityCenter	Unified Functional Testing / QuickTest Pro

Feedback

Requirements Model – Scaled Agile Framework



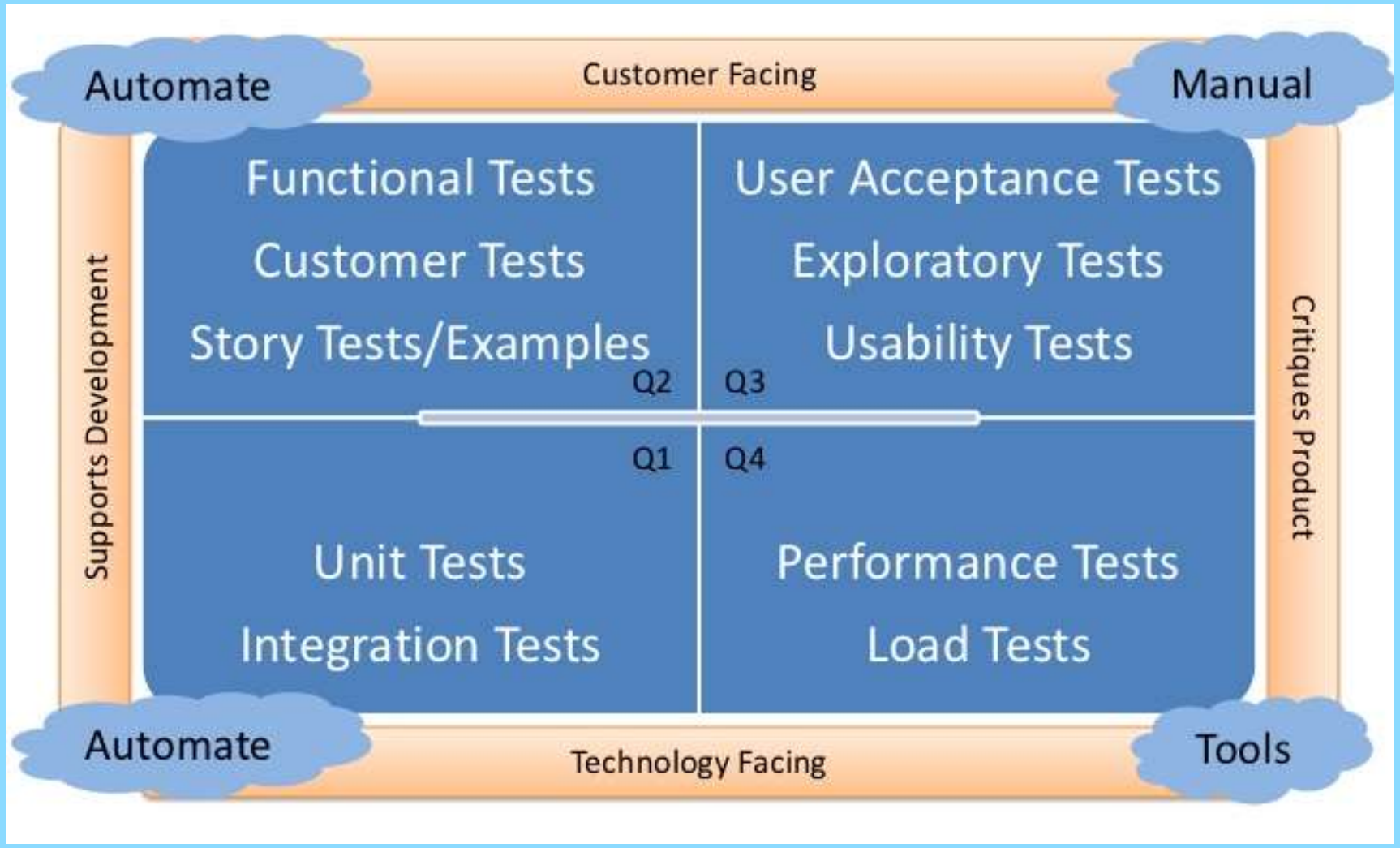
Quality Assurance and Test Process

Return

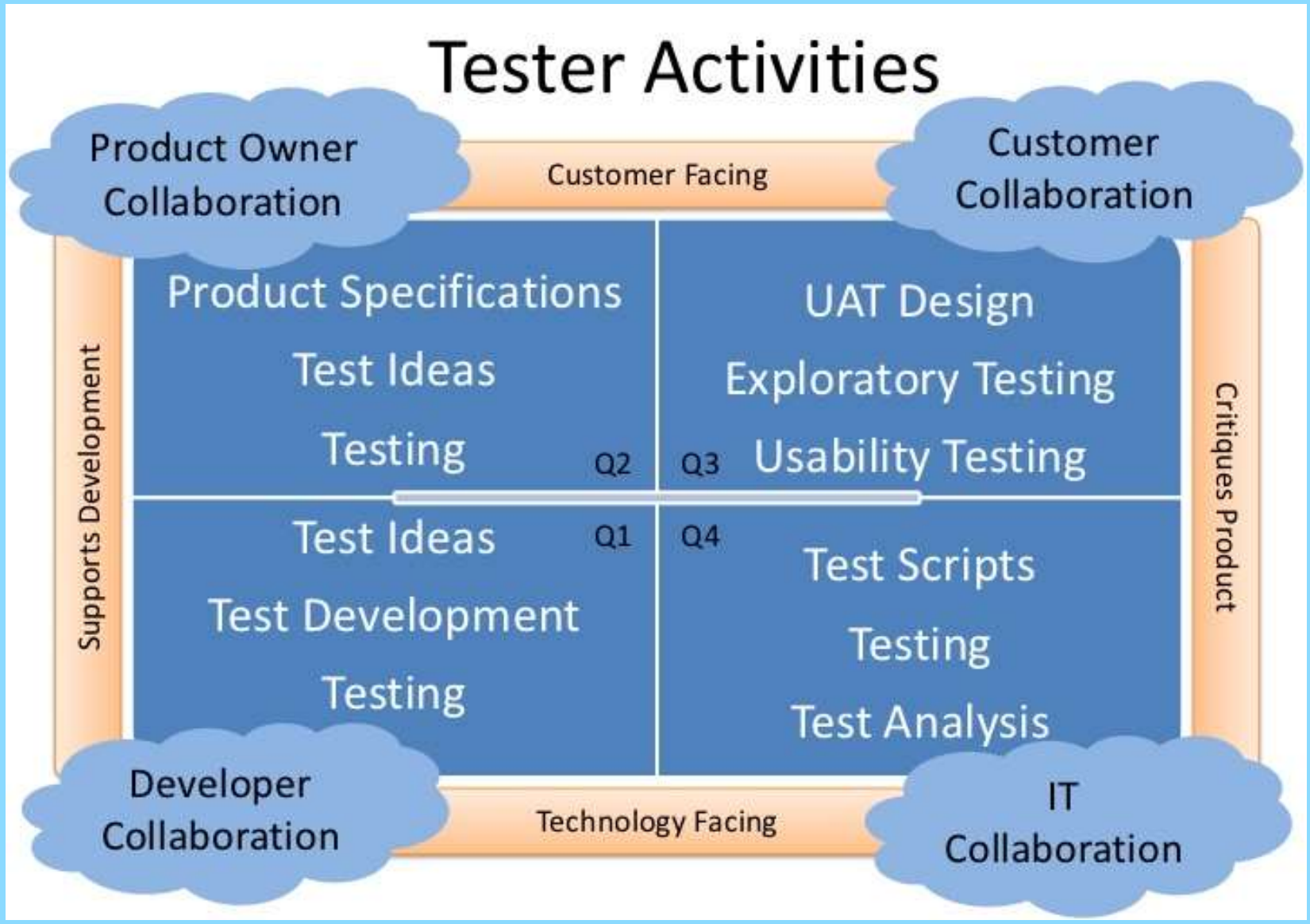
Test Approach Summary

- Development: Continuous Integration – test at the lowest level possible
  - Unit test
  - Functional test
  - Application integration test
  - Acceptance test
- Build
  - Develop build test harness (all unit tests pass)
  - Build Smoke Test
    - Combination of application test types
  - Establish clear rules/communication to address fixes
- Promote to Integration and Test
  - Full regression testing
  - System Test (includes integration, cross-application)
  - Acceptance Test – executes user-scenario test types (end-to-end)
  - Performance and Scalability tests

Agile Testing Concepts



Agile Testing Concepts



Quality Attributes

Return

Quality Attributes - Overview

Guidance on Quality Attributes is provided in the following categories:

- Availability
- Performance
- Robustness
- Cybersecurity
- Interoperability
- Extensibility
- Operations
- Sustainment
- Data
- Acquisition

Quality Attributes - Availability

• **Extensibility**	Adding features, and carry-forward of customizations at next version/upgrade. There are three classes of Extensibility; White, Black and Grey box. White box has two sub-classes, Open and Glass. Open being the direct modification of the source code and Glass being modification of the system purely by extending the existing source code without modification of the original. Black box typically found in data driven systems extends systems only through the use of their open API's and Grey box is a hybrid of the two that allows programmers to modify the extentions/API's.	The concept of this measure is first the definition of the style the system software is adopting and then measuring its success through inspection of the design. The object of the measure is dependent upon the style but in all cases it is focused upon the amount of change to the core source code necessary to support change due to functional and non-functional grown and the deployment requirements of the customer. The method will be a the change metrics from the project/program over time.
• **Adaptability**	This quality includes the attributes of software expandability, flexibility and upgradeability	The concept of this measure is similar to Extensibility but deals with the amount of change needed in the software as built to support its deployment to the customer. The object is any the number and magnitude of software changes to the system required to meet the customer's needs and thus the degree and magnitude that the implementation changes from its baseline. The methods will be the metrics from the change and configuration management processes.
• **Maintainability**	Characteristic of design and installation, expressed as the probability that an item will be retained in or restored to a specified condition within a given period of time.	The concept of this measure is reflective of how well the system was designed for maintenance over its lifetime. The object of the measure is both the effort to effect change and the degree the design isolates the change only to the target system feature or capability. The methods of this effort is the percentage of sustaining or development effort that is apportioned to the core system versus to the development of new features and capabilities.

Quality Attributes - Performance

• **Usability**	The ease of use and learnability of software application, website, device, or process	The concept that is being measured is the ease and speed in which a user can learn the system. The object is the user's ability to resolve questions about how to use a feature or function of the system intuitively and the method is through observations of how quickly the user ascertains the information needed to understand the same and how quickly he is able to apply the information to complete his task.
• **Capacity**	Loads, current and forecast, steady-state and peak	The concept that is being measured is the capacity of the system to support a given workload and the behavior of the system when that workload changes. The objects of measurement are the functions in the 3 tiers of the architecture from the presentation tier and the efficiency in which it displays its data, through the logic tier and its ability to manage message loads (in the case of a message orientated architecture) to the data tier and its ability to effectively retrieve and store data. As a result this quality will require several measurement methods.
• **Efficiency**	Resource consumption for given load	The concept that is being measured is the efficiency in which the system consumes available computing resources. The objects of measurement are those resources that are finite. The method is therefore are the initial footprint (idle) of the system at the presentation and logic tiers and then how the computing resources are used from system idle to peak load and back again.

Quality Attributes - Performance

• **Effectiveness**	Resulting performance in relation to effort	The same concepts are being measured here as in Efficiency however the object is how effectively the system allocates additional resources when needed as in load balancing or allocating more compute resources in an elastic infrastructure. The measures are both the granularity in which additional resources are allocated and any latency the system incurs when allocating more compute (the system having to wait (degrade) because the resources are not yet available).
• **Resource Constraints**	Processor speed, memory, disk space, network bandwidth, etc.	The concept of this measure is the sensitivity of the system to the operating environment. The objects are the characteristics of the compute needed by the system in order for it to optimally function. The objects of this measure are the number of and type of processors, memory and storage and the network profile. The measures are reflective of the type of system, is it computationally intensive or principally marshalling I/O?
• **Response time**	The response time perceived by end user is the interval between (a) the instant at which a user at a device enters a request for a response and (b) the instant at which the last character of the response is received	The concept of this measure is the rational perception of latency. As a rule of thumb humans perceive latency as a delay of greater than 100ms. Rational latency however factors in expected system behavior i.e. the time one would expect for the letter I typed on my keyboard to appear on the screen is 100ms however if I am requesting a list of data that I know is long I may be willing to wait up to a second without being informed it is taking longer. The object therefore is a set of quantified measures against classes of functions or capabilities in the system for instance the capability of logging onto the system involves several functions: Entry of Authentication Credentials, Look-up on LDAP, set user-id permissions or present error message or pass user onto to requested feature. Each of those functions introduces some latency and added all together will determine the response time of the log-on capability thus the measure is an aggregate. First Define a list of principle features (classes) of the system to measure and then define the expected response time against each class.

Quality Attributes - Robustness

• **Robustness**	Capability of a computer system to cope with errors during execution or the ability of an algorithm to continue to operate despite abnormalities in input, calculations, etc.	The concept of this measure is dealing with the ability of the system to run without interruption. This necessitates strategies to trap errors before they do harm to functions or data. The objects of this measure are not how many are trapped rather they are how many are not and thus how many system failures occur. Breaking this down similarly as capacity into the 3 tiers of the system we can define how effective the tactics must be i.e. at the presentation layer the objective is to capture any error at the point of entry providing responsive feedback to the user before they request the system to perform a function. At the logic layer defining the factors that will cause an algorithm to failure and then preprocessing those factors to ensure that the error is trapped before the algorithm fails. Finally at the data layer and ensuring the integrity of the data in transit and at rest. The methods will be how many function, feature, or system failures are logged and performing root cause analysis to determine how they were introduced.
• **Safety**	Capability of a life-critical system to behave as needed, even when components fail	Concept of this measure is to reduce the risk that any one component will cause the system to be inoperative. For systems that are not life-critical this quality is represented by other qualities that determine availability. For systems that are life-critical the FAA advisory for System Safety Analysis and Assessment for Part 23 Airplanes should be referenced.

Quality Attributes - Robustness

• **Survivability**	Capability of a system, subsystem, equipment, process, or procedure to continue to function during and after a natural or man-made disturbance	The concept of this measure is to quantify the approach taken to ensure that there are no single points of failure in the system design. The object of the measure are testable outcomes from those designs i.e. testing the system's response to failing over a web, application or data server and assessing the degree in which the failure impacts the user. From the user's perspective the amount of work lost, how much has to be redone is the observable measure. The impact is defined in the systems recovery/recoverability objective or overall system Service Level Agreement. An overall objective is to be the least disruptive as possible. The methods for measuring this quality are the system's inability to survive in part or whole a component failure and tested both through destructive testing (shutdown a component while the system is running) and forensic analysis when the system has incurred an outage in production. The second method is to analyze the impact to the user sessions, what work is lost, what is their state after the component failure, when informed to refresh their session what state are they in and where do they stand moving forward and how many users are affected.

Quality Attributes - Cybersecurity

• **Privacy**	System protects sensitive data from unauthorized access	The concept of measurement here is seemingly binary in nature, either you can or cannot access data you are not authorized to access. What is observable to the end user should either be nothing, they don't know the data exists, and/or an error message stating the information requested is not accessible based upon their authority level. Those appear to be a concern or measure of usability rather than Privacy per se, Privacy is still a binary measure for effectiveness. As for Efficiency the efficiency of the design and performance of the code to determine if you are authorized though specific to the privacy requirement are measures of performance. Finally the robustness measure, the breadth in which the system is able to determine the privacy of data evaluates if the system has the awareness of how separate data together creates a privacy requirement i.e. Person Name and Telephone Number apart may not be considered sensitive information however together they are. Is it therefore fair to state that the object of this measure is the completeness of the privacy design, that is both the determination of the data's privacy needs and the effectiveness in which it is protected from unauthorized access? If it is, can completeness be quantified or is it mostly subjective ergo qualitative?
• **Computing Security**	System software is designed with mechanisms that prevent unauthorized changes or access to computing resources, and has automated detection and self correcting	The concept of this measure is the auditability of the systems CM process from the infrastructure through the layers of the architecture to the application software release process, including Mobile if applicable. The object of the measure is the accuracy of the CM process, are all changes logged and approved. The methods are an audit of the approved change requests against the changes in the infrastructure or application based upon change period scan data; for a given period scan for changes to infrastructure and application software against approved change requests.

Quality Attributes - Cybersecurity

• **Application Security**	Full application lifecycle security practices: A Design-in Security strategy, Authentication, Authorization and compliance, Entitlement processes, Segregated policy points, Penetration testing, and Monitoring, alerting and auditing capabilities	System and process design evidence and metrics. See "Security SDLC" tab
• **Data Security**	Data is secured to the level required by its sensitivity in every point of the Data Lifecycle.	The concept of this measure is the degree in which the design protects the data at every point of its lifecycle. See Tactics for matrix on data sensitivity. The objects are the application of hardware and software based data protection approaches
• **Information Security**	The designs the aggregation of data into information providing segregation of data where the sensitivity of the data warrants. System provides techniques for detecting and managing irregular data access such as mining or bulk exports	The concept of this measure is to demonstrate in design and testing the ability of the system to defend information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. This is different from data security in that it must address that the aggregation of data has greater sensitivity. The objects of this measure are the results from the inspection of the how robust the design is in defining the data relationships sensitivity (see Tactics on Data Sensitivity) and the effectiveness of controlling the information assets at the appropriate level and the efficiency or the lack of constraints that information security places on the system resources. The method of measurement will be number of and resolution of findings in an assessment of the design. In addition measuring the effectiveness of the software to implement the design through White/Black Hat testing.

Quality Attributes - Cybersecurity

• **Network Security**	Request and data transit is secure i.e. https	The concept of this measure is the security of the overall network design to prevent unauthorized access to any node on that network. The object of this measure are the security design and implementation that prevents a malicious user from being able to exploit weakness in the design for unauthorized access to a node on the network. The method of measurement will be number of and resolution of findings in an assessment of the design. In addition measuring the effectiveness of the software to implement the design through White/Black Hat or Penetration testing.
• **Information Assurance**	Information assurance is the process of adding business benefit through the use of Information Risk Management which increases the utility of information to authorized users, and reduces the utility of information to those unauthorized.	The concept of this measure includes the enumeration and classification of information assets and their sensitivity. Through risk analysis the appropriate tactics (need to brainstorm) are applied in the system design, threat modeling and test plan. The object of this measure starts with the enumerated and classified list of information assets. Tactics are then defined to address any vulnerabilities weighing the risk of either loss, corruption or deleting of the assets against the cost of implementing and maintaining designs to protect them. The methods of measurement are the results of the risk assessment and cost of the software design along with the test results that verify their implementation. An additional measure is formal certification by a third party auditor should it be warranted.

Quality Attributes - Interoperability

• **Open Standards**	Syntactic, Semantic, Cross-Domain Interoperability for information exchange	Uses open or industry standards where applicable
• **Network Topology**	The arrangement of various deployment elements (e.g., links, nodes)	The concept of this measurement is the effectiveness of the logical design to meet the system's requirements and the efficiency of the physical design and the overall design's ability to meet the performance and scalability requirements of the system. The object of the measures are the outcome of performance and scalability testing that demonstrate those concepts. The measures will be latency and survivability results.
• **Operability**	To keep a piece of equipment, a system or a whole industrial installation in a safe and reliable functioning condition, according to pre-defined operational requirements	The concept of this measurement is the discoverability and health monitoring for all of the nodes of the topology. The objects are the accuracy of the topology and the duration at which the nodes health is evident and modifications to the topology or nodes are discovered.

Quality Attributes - Interoperability

• **Backward Compatibility**	Products designed for the new standard can receive, read, view or play older standards or formats, then the product is said to be backward-compatible	The concept of this measure is the degree in which change impacts the system or its users. The object of this measure is the ration of changes that have collateral impacts versus those that do not. The method will be based upon the number of correlated changes required because of changes to other components. It is important to categorize changes that are being made to take advantage of new functionality as these should not count against this measure.
• **Platform Compatibility**	Runtime libraries allow re-use of code and provide abstraction layers which allow the same high-level source code to run on differently configured hardware	The concept of this measure is the degree in which change impacts the platform or its users. The object of this measure is the ration of changes that have collateral impacts versus those that do not. The method will be based upon the number of correlated changes required because of changes to the platform services as well as the changes to the platform that are necessary due to its deployment on other hardware. It is important to categorize changes that are being made to take advantage of new functionality as these should not count against this measure.

Quality Attributes - Extensibility

• **Extensibility**	Adding features, and carry-forward of customizations at next version/upgrade. There are three classes of Extensibility; White, Black and Grey box. White box has two sub-classes, Open and Glass. Open being the direct modification of the source code and Glass being modification of the system purely by extending the existing source code without modification of the original. Black box typically found in data driven systems extends systems only through the use of their open API's and Grey box is a hybrid of the two that allows programmers to modify the extensions/API's.	The concept of this measure is first the definition of the style the system software is adopting and then measuring its success through inspection of the design. The object of the measure is dependent upon the style but in all cases it is focused upon the amount of change to the core source code necessary to support change due to functional and non-functional grown and the deployment requirements of the customer. The method will be a the change metrics from the project/program over time.
• **Adaptability**	This quality includes the attributes of software expandability, flexibility and upgradeability	The concept of this measure is similar to Extensibility but deals with the amount of change needed in the software as built to support its deployment to the customer. The object is any the number and magnitude of software changes to the system required to meet the customer's needs and thus the degree and magnitude that the implementation changes from its baseline. The methods will be the metrics from the change and configuration management processes.
• **Maintainability**	Characteristic of design and installation, expressed as the probability that an item will be retained in or restored to a specified condition within a given period of time.	The concept of this measure is reflective of how well the system was designed for maintenance over its lifetime. The object of the measure is both the effort to effect change and the degree the design isolates the change only to the target system feature or capability. The methods of this effort is the percentage of sustaining or development effort that is apportioned to the core system versus to the development of new features and capabilities.

Quality Attributes – Extensibility

• **Portability**	The usability of the same software in different environments	The concept of this measure are either the constraints or amount of change necessary to deploy the software to the target environments. The objects are the constraints the software places on the runtime environment from the OS that it prescribes to the middleware or other commodity software required to operate the system in the environment. The methods of this measure will be inspection of the deployment documentation and any prescription beyond capacity requirements.
• **Scalability (horizontal, vertical)**	Capability of a system, network, or process to handle a growing amount of work in a capable manner or its capability to be enlarged to accommodate that growth	The concept of this measure is the degree in which the software responds to the demands placed on it by its operation (internal) influences. The objects of the measure are the methods used for the software to utilize additional compute capacity through internal or external means such as load balancing or additional allocated infrastructure. The methods of this measure are performance testing results that demonstrates the software utilizing resources to meet load demands within its latency and performance specifications or SLA's
• **Reusability**	Capability for a segment of source code to be used again to add new functionalities with slight or no modification	The concept of this measure is the reuse of system features or capabilities in the product development process. The object of this measure is the catalog of assets, their modularity and documentation most importantly purpose description and conditions of use. The method of measurement is the evidence of reuse in metrics generated from the build or runtime environment.

Quality Attributes – Operations

• **Accessibility**	The degree to which a product, device, service, or environment is available to as many people as possible	The concept of this measure is availability of the software to end users. There are two considerations of availability, the measures for concurrent users and the human-computer interaction that support users with special needs. The objects are then the design of the client server and network, the approaches and methods used to accommodate special needs such as Java Access Bridge. The method of the measure is the test results of the design for scalability and principles, design software.
• **Recovery / recoverability**	(e.g., mean time to recovery - MTTR) and recovery time objective - RTO	The system design should be configurable for recovery at the data layer. The system design should be stateless where only uncommitted work is lost while ensuring the correctness and validity of the data
• **Disaster recovery**	Process, policies and procedures that are related to preparing for recovery or continuation of technology infrastructure which are vital to an organization after a natural or human-induced disaster. Also known as Business Continuity.	The concept of this measure is the ability to recover from a man-made or natural disaster whereby the system cannot run on the infrastructure from which it was deployed. The objective is to measure the time and completeness of the recovery from detection to reinstatement of the system. There three types of objectives: 1. Preventive measures - controls and designs aimed at preventing an event from occurring, 2. Detective measures - controls and designs aimed at discovering unwanted events and 3. Corrective measures - controls aimed at correcting and restoring the system after an event occurs. The methods of measurement will be the Recovery Point Objective (RPO) related to data loss and Recovery Time Objective (RTO) related to availability of the system. Both are measured in time from when detection occurs until the system is fully available at established SLA's.
• **Resilience**	Capability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation and tolerate unpredictable or invalid input.	The concept of this measure is the software's ability to either protect itself from data that would cause a fault or deal with situations such as low memory without failing. The objects of this measure are the coverage of the design principles for known fault areas and inspection for their application in the code along with the test designs. The methods of this measure are the metrics from the test's execution and the number of findings found and resolved in the source code peer reviews.

Quality Attributes - Sustainment

• **Quality**	Defined by faults discovered, faults delivered, fault removal efficacy	The concept of this measure are defined in the Block 1 & 2 metrics out of DA's Quality Organization (see notes and SDLC Process Flow). The object of this measure is to ensure that the processes in place are producing the level of quality required in DA's program and project KPI's. (need a reference). The method of these measures should be provided from the DA Standard tools (JIRA for program/project and development management, Contour for requirements management and Quality Center (ALM) for test management and defect reporting)
• **Stability**	Property such that components over time will not need changes	The concept of this measure is to determine the fragility of a component by looking at its dependencies and how it has abstracted their interfaces. The object of this measure is to make the component as robust as possible while also providing stability for those components that are dependent upon it. The method of its measure are its design. The criticality of the module to the overall system is its complexity as measured by the number of dependencies it has with the following inputs: - Components which invoke module - Components which module invoke - Number of variables in both of the prior cases - The number of Global Data References Over time these inputs will change making the component more or less critical to the system. This information can be used in the Product Risk Analysis (See SDLC Process Flow) The analysis of the design should focus on the implementation of tactics used to make it more stable in the system.

Quality Attributes - Sustainment

• **Supportability a.k.a. Serviceability**	Capability to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis, and provide hardware or software maintenance in pursuit of solving a problem and restoring the product into service	The concept of this measure is the ability of the technical support personnel to install, configure and monitor the systems operation resulting in more efficient product maintenance and reduction of operational costs and maintaining business continuity. To that end the object of these measures are the breadth of services supporting these areas, the depth in which they are supported and the ease in which they can be performed. The method of this measure are the coverage of capabilities and the lower operations support costs afforded by the system owner.
• **Testability**	The degree to which a software artifact (i.e., a software system, software module, requirements- or design document) supports testing in a given test context	The concept of this measure is to ensure that a. the requirement is written in a testable manner and b. that the developer uses an approach that will meet the requirements with the minimal amount of code. The objective is to measure how well the system meets the functional and non-functional requirements. The methods of measure are the maturity (clarity and completeness) of the requirements and the Block 1 measurements for pre and post release defects

Quality Attributes - Data

• **Confidentiality**	Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information.	The concept of this measure is the ability of the system to implement the appropriate level of data security as defined on the tactics slide. The objective is to prove that they have been met through non-destructive testing of the system's authorization controls. The methods of this measure are the set of test cases written expressly to test confidentiality and the use of external teams to attempt to access the data using tests that known security vulnerabilities have been blocked.
• **Integrity**	System takes safeguards to ensure both the correctness and validity of data including correctness checks at the point of entry and integrity checks of interface and data at rest	The concept of this measure is the system's ability to ensure the accuracy and consistency of data over its entire lifecycle. The object is to ensure that the data is stored exactly as intended through the physical and logical software and in some cases hardware designs. The method of measurement is the application of database design principles for referential and entity integrity to implement ACID properties. Tests performed on the application will test for Data Integrity thus both the thoroughness of the tests and tests results will provide evidence.
• **Availability**	Is the requirement that data and processes be protected from denial of service to authorized users.	Data and System Availabilities are highly important characteristics to consider for all Digital Aviation software development projects. Each project will specify availability requirements, usually in terms of “9s” – 4 9s = 99.99% availability, for example, equates to less than 53 minutes of downtime in an entire year. Development teams need to understand the availability requirements for their projects and ensure that end-to-end design decisions and architecture support those requirements. There are risks of over engineering for too many 9s or failing to meet customer needs with too few 9s.
• **Lineage**	Is the requirement that the data source be understood and any modification to the authoritative data be disclosed	The concept of this measure is the completeness of information either documented or embedded about the data's lineage. The objects are then the system design and data architecture documentation and where appropriate the ability of the UI to divulge the data's origin and transformations. The methods of this measure are inspection of the design and audibility of the data's history.

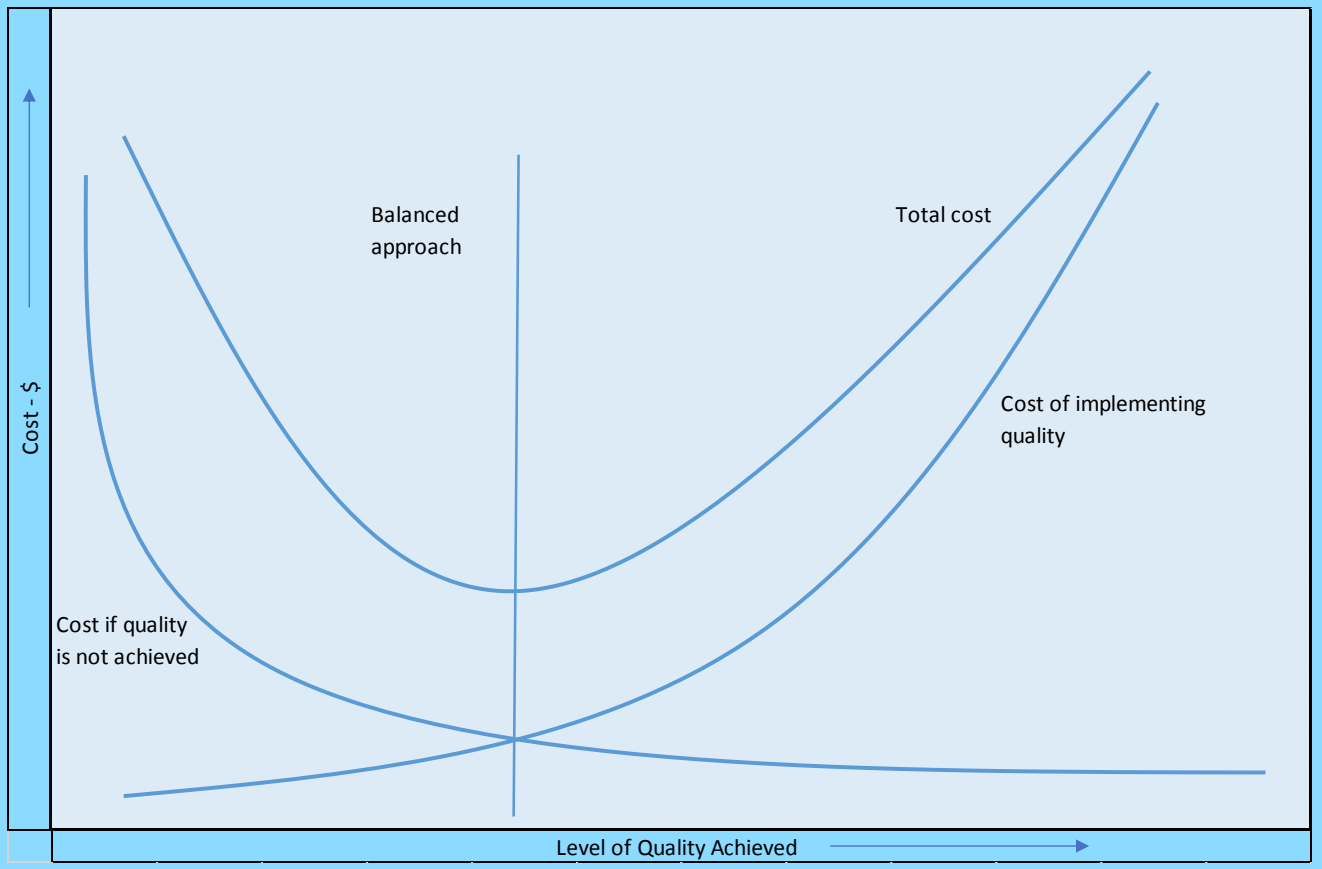
Quality Attributes - Acquisition

• **Affordability**	Cost/Price	The concept of this measure is the degree in which the system's design economizes resources. Resources are labor, time, computing, storage and other i.e. Software Licensing costs. The objects are the costs in dollars, estimated and then actuals. These are balanced against the potential and then actual revenue from the software. The measure is the margin between them, or profit, potential or earned.
• **Certification**	The confirmation of certain characteristics of an object, person, or organization	The concept of this measure is the degree in which we can reduce our risk by insuring our software partners are who and what they claim to be by assessing various industry sources of certification. The object will be dependent on the partners claims i.e. an Cloud Infrastructure as a Service supplier should be able to produce certification that they have been audited and tested as a secure data center. The method is inspection of the certificate agencies reports on the partner measured against our requirements.
• **Configuration Management**	Process for establishing and maintaining consistency of a product's performance, functional and physical attributes with its requirements, design and operational information throughout its life	The concept of this measure is the rigor in function and process of the documentation and software configuration management. The object are the managed objects of the program or project and adherence to a set of controls that ensure that the only what is approved and tested moves to production, ergo minimizes defects that impact the customer. The method is the collection of and root cause analysis of any post-release defects, the certification that the software objects that are moved to production are known by the customer and that the production documentation represents productions current state.

Quality Attributes - Acquisition

• **Compliance**	Conforming to a rule, such as a specification, policy, standard or law	The concept of this measure is to provide certification that we are compliant with the policies, standards and laws for which the software will be used. The objects of the measure are the certifications and their economic value to the project. The method of the measurement is the valuation of the certificate which may be absolute, i.e. the product has no value without it or partial, the portion of the market sector available if certification is received against the efficiency of getting the certificate.
• **Escrow**	The deposit of the source code of software with a third-party escrow agent	The concept of this measure is meeting the requirement whereby a third part must act as an agent for the software provider so as to ensure that the same does not have access to data that is classified as sensitive by the customer e.g. if Boeing was the software provider but the system was to be used for both Boeing and Airbus sensitive data a third part would be contracted to provide configuration management and operations services for its deployment. The objects of this measure is dependent on the degree in which the third part must ensure that software cannot leak any data to the software provided. The methods will be process inspection of the third party and threat analysis and penetration testing performed by an independent service provider.
• **Legal and licensing issues or patent-infringement-avoidability**	End-user license agreement (EULA) or software license agreement is the contract between the licensor and purchaser, establishing the purchaser's right to use the software	Clearly documented and communicated agreements along with process enforcement of the same or evident
• **Open source**	A development model promotes a) universal access via free license to a product's design or blueprint, and b) universal redistribution of that design or blueprint, including subsequent improvements to it by anyone	The designs maximize the use of commodity OSS

Quality Attributes – Balanced Approach



- Quality much be achieved with careful attention to the total cost required.
  - Saving money initially, but failing to meet quality requirements for Digital Aviation Customers will likely result in higher total costs in the long run
  - Designing applications with quality in mind will result in efficient achievement of the required quality attributes

Digital Aviation Reference Architecture

For further information, please contact:

Scott D. Taylor
Management Lead – Digital Aviation Reference Architecture
Senior Manager, Software Engineering
Jeppesen, a Boeing Company
Email: scott.d.taylor2@jeppesen.com
303-328-4415 - Office
303-332-7022 - Mobile