Enterprise Architecture Principles

Revised - 12/6/2017

Security

Principles	Rationale
Assess risk across the entire system, not only within a particular layer.	Utilize the ‘defense in depth’ approach. Risk and security must be understood and applied across the whole system and not just within a specific layer.
Balance risk, asset value and cost to protect within the context of approved security policies.	Security can't be appropriately applied without an understanding of the risk, including existing threats and impacts, as well as the "value" of what is being secured.
Include both prevention measures and detection and response functions.	Failures will occur and perfect security is impossible to achieve, so it is important to balance prevention measures with detection and response functions.
Build security into the entire product lifecycle.	Security is best accomplished if built into the entire product lifecycle (design, deployment, operation, and end of life) and not "bolted on" afterwards.
Consider people, process and technology in making security decisions.	Security is comprised of people, process, and technology, and done well needs to take all three into consideration.

User Experience

Principles	Rationale
Prioritize user impact in development and selection efforts.	Understand your users and their needs and make that a priority for design decisions.
Optimize for the entire user journey and experience.	Ensure that all touchpoints of the user journey are optimized for a great user experience across all channels and devices for all users.
Incorporate user feedback throughout the design, testing, and implementation process.	Continually test designs with users to ensure effectiveness, efficiency, and satisfaction.
Ensure the accessibility and mobility of products whenever possible.	Make Interactive systems equally operable by all: • Regardless of circumstances or limitations • On all common devices (computer, laptop, tablet, phone).

Applications

Principles	Rationale
Minimize customization and in-house development.	Favor SaaS, then COTS solutions before considering investments in customization and development efforts. Consider total cost of ownership, time to market, vendor lock in and other criteria when making build/buy decisions.
Select and build applications that meet multiple needs and can support multiple organizations.	Applications should deliver functionality that can be used in multiple organizations. Avoid the duplication of effort and unnecessary expense of redundant implementations.
Select and build applications that include shareable components, preferably using APIs.	Use services from other applications when available. Expose unique functional capabilities to other applications as services.
Build and evaluate applications considering institutional principles and policies.	Organizations should have confidence that application teams have proven the effectiveness and security of their solutions. Users should have confidence that their interactions with applications will not harm them.
Invest in applications that comply with contemporary development, operations and support practices.	Applications designed for the cloud (cloud native, 12 factor) can more easily take advantage of cloud scaling, automation, DR and monitoring capabilities.

Middleware

Principles	Rationale
Use middleware solutions for common, shared application functions.	Middleware provides services to other software as opposed to implementing business functions directly. Using middleware services as supporting components to the functional capabilities of applications can simplify development and support portability.
Use enterprise shared services whenever possible.	Shared services for access management, logging and other common needs reduce duplication of effort, help achieve economies of scale and can improve quality. Give preference to services that provide full management and operation capabilities to application teams in order to minimize redundant investment in staff, skills, and computing resources.
Use shared services that work in multi-tenant environments.	Give preference to shared services that are able to support multiple applications using the smallest number of instance implementations.

Interoperation

Principles	Rationale
Build and use reusable APIs to exchange data	APIs are the preferred method of moving

between systems.	information between systems.
Prefer open standards.	Open standards ease interoperation, facilitate broader adoption and reduce vendor lock-in.
Document interoperation interfaces.	Interfaces must be well documented and freely available. Interfaces must be documented using standard languages.
Select tools and products that have multiple implementations.	There should be multiple vendor or open source implementations for vendor-supplied interfaces.
Use an API versioning system to manage API changes and indicate compatibility levels.	Minimize version changes to provide stability. A change in syntax or semantics requires a new version. Manage and document your API lifecycle.

Data

Principles	**Rationale**
Source systems should export data in a single format.	Source systems should provide data in only one format.
Transform data the least number of times and into the smallest number of different formats.	Process once, reuse many times. Data transformation for common data assets is performed the smallest number of times, ideally once.
Obtain data only when needed in order to maximize data currency.	Obtain data from other systems only when needed, except when coordinated snapshots are needed for consistency such as fiscal year closing. Make it timely.
Document data element descriptions and meaning.	All data assets must be documented with descriptions and easily available to members of the Harvard Community.

Infrastructure

Principles	**Rationale**
Use infrastructure and services that enable virtualization, abstraction, elasticity, and automation.	Make every effort to leverage cloud infrastructure first. Continuously improve Cloud solutions and empower customers to take advantage of the full benefits of the Cloud. Facilitate evolution with the technology to achieve greater value in both time and cost. Provide the highest quality level of service to encourage universal Cloud adoption and buy-in. Provide the means for migrating to a Cloud infrastructure.
Reuse common capabilities and automate repetitive processes.	Focus on using architecture patterns to achieve efficient results, modularity and enterprise-wide standardization. Empower the customer to take advantage of Cloud capabilities. Favor AWS-native over vendor agnostic solutions except where ITSM-specific services are required (e.g. monitoring, logging, alerting, centralized configuration management etc.). Be open to SaaS integrations. Encourage innovation

	and experimentation.
Use infrastructure and services that enable developers and administrators to manage application performance, cost and operational risk.	Provide expertise and offer services that enable the customer to make well-informed decisions and actively manage their applications. Provide dashboards that simplify viewing performance and cost information and tools that streamline configuration changes.
Ensure infrastructure services offer appropriate levels of security, configurability, resiliency and recoverability.	Align customer applications with Harvard's IT direction. Provide foundational services to customers that improve application quality, delivery and reliability. Ensure the Cloud resources provide resiliency to customer applications. Align to ITSM practices.

Network

Principles	**Rationale**
Leverage open and established standards whenever possible.	Be open - leverage open and established standards and discourage the use of proprietary protocols or narrow implementations.
Identify failures modes and design accordingly.	Provide seamless recovery from failure. Design and expect failure; routine failure should not impact availability.
Control access using identity rather than network address.	Use a meaningful identity - Users and applications should be permitted through their identity and system and not their current address.
Enable self-service.	Provide systems and controls to give end users flexibility and control over their resources.