

## **Enterprise IT Architecture Principles**

**Purpose:** A logically consistent and easily understood set of principles that guide the engineering of APU's enterprise\* information technology and services. These principles serve as a starting point for a more detailed enterprise architecture, which should include practical patterns within each discipline. The principles are one factor among several that provide guidance for IMT decision making, and do not represent the sum of all possible requirements for solution determination.

The principles assume the following drivers affecting strategic design of future systems:

- *Agility:* Design that leads to sustainable success in the midst of constant change is critical to meeting organizational goals within resource limits.
- *Accessibility:* Real-time services and information resources are expected to be accessible from wherever and whenever they are needed.
- *Security:* Making sure only the right people can get to resources is a pervasive challenge which requires central services and distributed controls.

### **Principle Summary:**

1. **Open Standards** - Agility requires the use of open standards for process, data, interface, and information exchange.
2. **Modular Design** - A solution's overall functionality can be decomposed into smaller functional building blocks.
3. **Adaptable Interfaces** - The design of a service doesn't assume a fixed input or output interface.
4. **Reusable Services** - Loosely coupled applications and technology form services to meet common needs.
5. **Leveraged Data** - Data is an enterprise asset that should be leveraged across the organization over time.
6. **Dynamic Security** - Solutions interpret security by plugging into centralized identity and policy services over secure channels.
7. **Internet Delivery** - The Internet is the common distribution channel for enterprise information services.

Synergy of Principles: Open standards, modular design, and adaptable interfaces enable rapid assembly of reusable services which leverage common data for secure Internet delivery of applications and information.

\* **Scope:** These principles apply to enterprise services and data: those services which are broadly used across the university, support core functions, or interact with essential enterprise data.

## **Principle Detail:**

### **1. Open Standards** - Agility requires the use of open standards for process, data, interface, and information exchange.

*Definition:* We have proposed the use of the International Telecommunication Union (ITU-T) Definition of Open Standards<sup>1</sup>:

"Open Standards" are standards made available to the general public and are developed (or approved) and maintained via a collaborative and consensus driven process. "Open Standards" facilitate interoperability and data exchange among different products or services and are intended for widespread adoption.

Other elements of "Open Standards" include, but are not limited to:

- *Collaborative process* – voluntary and market driven development (or approval) following a transparent consensus driven process that is reasonably open to all interested parties.
- *Reasonably balanced* – ensures that the process is not dominated by any one interest group.
- *Due process* - includes consideration of and response to comments by interested parties.
- *Intellectual property rights (IPRs)* – IPRs essential to implement the standard to be licensed to all applicants on a worldwide, non-discriminatory basis, either (1) for free and under other reasonable terms and conditions or (2) on reasonable terms and conditions (which may include monetary compensation). Negotiations are left to the parties concerned and are performed outside the Standards Development Organization (SDO).
- *Quality and level of detail* – sufficient to permit the development of a variety of competing implementations of interoperable products or services. Standardized interfaces are not hidden, or controlled other than by the SDO promulgating the standard.
- *Publicly available* – easily available for implementation and use, at a reasonable price. Publication of the text of a standard by others is permitted only with the prior approval of the SDO.
- *On-going support* – maintained and supported over a long period of time.

The opposite of an Open Standard is a “proprietary standard”, which may have become a *de facto standard* due to popularity. De facto standards are only as open as their owner allows, and thus do not have all the same benefits of Open Standards.

*Impact:* The use of exclusively Open Standards is an ideal, and the existence of this principle provides a way to measure value and risk which will be interpreted independently in each situation. An organization which seeks out solutions, services and development models which are based on Open Standards will reduce the risk of external and internal change, and increase the durability of investments. Open Standards allow for solution durability because the technology specification is independent from the individual implementation. Will the applications you deploy today work on devices beyond your current consideration? Will data created by an application today be accessible even if the application

---

<sup>1</sup> ITU is a specialized agency of the United Nations. ITU-T definition: <http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html>

vendor goes out of business? If the applications and devices your organization adopts adhere to Open Standards, then the likelihood of positive answers to these questions increases. Open Standards enable agility, allowing an organization connect and automate processes that transcend technologies, platforms, languages and customizations.

*Example:* The best example of a technology built on Open Standards, is the Internet. The answer to the success of the Internet lies primarily in the interest of its participants to achieve a common goal of connectivity based on the adoption and commitment to Open Standards.

**2. Modular Design** - A solution's overall functionality can be decomposed into smaller functional building blocks.

*Explanation:* Architecture is often defined as the discipline of breaking apart of a system into its separate parts, and understanding the relationship between them. If one cannot break apart a system into individually useful parts, then modular design has not been followed. Such systems are referred to as monolithic. Essential to this concept of modular design is that each component should only perform one conceptual function. In some cases a module may have a few functions, but they serve the conceptual intent. Design targets should meet the most generic need related to the function, so that the potential for reuse increases.

*Impact:* Long-term benefits include: less time to market, reduced cost, flexibility, and extensibility. Flexibility is gained by modifying existing services to meet changing needs by replacing modules instead of the whole system. Components within modular design have a small "surface area", allowing for system functionality to be extended with less effort and risk, than if the system was built in a monolithic way. We call this benefit, extensibility. As stated above, building a system with modular components is not accidental: it is a result of solid design and engineering. In the short run, efforts require more up-front time until a library of reusable components is available.

*Availability Implications:* Modular design is also a primary attribute leading to scalability. As demand varies, components can be added or removed to alter supply as needed. Furthermore, if there is demand variance between different services, modular components can be shifted between services making the most of every resource.

**3. Adaptable Interfaces** - The design of a service doesn't assume a fixed input or output interface.

*Explanation:* For the purposes of this principle, an interface has two definitions: 1) the means by which the user interacts with an application; or 2) the means by which a service interacts with another service. Interfaces are abstract and independent from a specific device or system, in some cases allowing the user or a participating system to negotiate which to use. Whether it is a custom interface for a device, or on-the-fly data format conversion, interfaces should be adaptable. In software, a common example of this principle is the separation of business logic from display. Along with Open Standards, this principle is what allows the building blocks created with modular design to be interchangeable.

*Impact:* Because a service is not designed around a single user interface, the solution can be made available to new devices without a complete re-design. For example, a web application can be delivered to a cell phone by adding another interface plug-in, eliminating the need to re-write the entire application. In the case of a service interacting with another service, adaptable interfaces allow the

chaining of applications together to rapidly deliver new aggregated services as business needs arise. This fosters new levels of integration, and sets the table for Reusable Services.

**4. Reusable Services** - Loosely coupled applications and technology into reusable services to meet common needs.

*Explanation:* Instead of solutions being designed for use in a single context, they are designed for repeated use in various areas. This is inclusive but not limited to the formal software framework of "service oriented architecture" (SOA), which applies this principle to software design. In SOA, commonly used software functions are real-time executable over a network by many applications over an agreed upon interface. Applications become an aggregation of several smaller applications. Common building blocks then become the means to meet "new" needs without starting from scratch. How many things can you make with a Lego™ set?

*Impact:* This leads to fewer point solutions, which each require independent maintenance and development costs over time. Instead, maintenance of common needed functions feed common services. Time to market exponentially decreases as basic services are available on which to rapidly assemble larger services.

**5. Leveraged Data** - Data is an enterprise asset that should be leveraged across the organization over time.

*Explanation:* Often data is seen as tied to a specific application, rather than an independent enterprise asset. This leads to duplication of resources and process. Duplication of data can lead to decreased integrity of the information. Data should be stored where it can be leveraged the most. Even within a single application, data should be stored where it can be used, not buried in the code itself. Enterprise data should be centrally stored and highly accessible. Regardless of stewardship or original context, enterprise data should be able to be leveraged in the future by the enterprise.

*Impact:* If data was viewed an asset, then it would be stored in a place that's accessible. It would also inspire an enterprise data model, an inventory of the assets, including descriptions of the types of data available and how it might be able to be used. It would trace the data movement within the organization realizing that the movement of data represents process. It would expose duplication of effort, when the same data was moved back and forth from one location, person, or system to another, or even collected for a second time.

**6. Dynamic Security** - Solutions interpret security by plugging into centralized identity and policy services over secure channels.

*Explanation:* Access to services and information are dynamic interpretations of global policy, local choice and personal context.

Increasingly, there is a need for security to be inherited from several sources:

- Enterprise Wide Policy – rules for data, access, retention, logging etc.
- Local Choice – users or groups picking who can gain access to something they have published
- Personal Context – an environment has security or privacy that is inherited from preferences or method of access

**Identity & Access Management:** A solution should dynamically interact with identity and policy services rather than duplicating security management. At minimum, enterprise solutions need to dynamically authenticate against enterprise authentication services. Unless local security is required to match the desired level of risk, systems should not require independent authentication tokens (e.g. username & password). Flexibility of business rules to protect privacy of information should be considered. The “leveraged data” principle applies to identity data as well.

**General Security:** Transport and data level security need to be built-in, and not an afterthought for solutions going forward. Solutions which do not address transport level security from the start may imply ignorance in other security design issues leading to greater vulnerability throughout. Software should be thoroughly tested for security vulnerabilities due to poor programming conventions and bugs.

*Impact:* The impact of being able to have the right people access the right resource is obvious. Dynamic security provides a way to combine enterprise wide policy and local decisions into a seamless environment which balances risk and convenience. The reuse of common identity data for authentication and interpretation of authorization for systems reduces duplication and cost. Services which have designed basic data and transport security into their solution reduce risk without additional costs. If applications don't have built-in security, the organization has to wrap an additional security layer around the application. Finally, solutions which have been engineered with security in mind reduce the organizational risk and cost of recovery for system compromises and exploits.

**7. Internet Delivery** - The Internet is the common distribution channel for enterprise information services.

*Explanation:* Though not all enterprise applications are delivered over the Internet, the ability to do so should be inherent in their design. The nature of enterprise information services, is that they need to be broadly consumed. The Internet has become the inexpensive common distribution channel for an increasing number of services that IT departments deliver. The network has become the computer, and services should be designed to be supplied and consumed over the public Internet from the beginning.

An Internet delivered solution is one which solely depends on standard Internet Protocols, as defined by the Internet Engineering Task Force. If a solution uses proprietary delivery mechanisms while making use of the public Internet, then it doesn't inherit all the benefits of this principle.

*Impact:* Low cost global delivery of service to customers using commodity devices. Furthermore, the direction toward “software as a service” and service oriented architectures implies that increasingly, an organization's set of information solutions will be an aggregation of software from several different service providers. The ability to mix and match “in house” services with external software supplied over the Internet provides an incentive to make sure new enterprise services are designed for the Internet.

**Implied Rule:** Our primary target interface for new enterprise applications is the Web Browser.

The most practical and common method for Internet delivery is targeting the web browser for application interface. Products which do not target the Web Browser from the start often end up retrofitting it poorly, because their design didn't allow for such a change in interface.

The primary reasons for targeting web are:

- Universal Access - the application and associated data stay in one place, but are accessible from anywhere allowed
- Platform Independence - application should work on multiple operating systems, browsers, and devices
- Lower cost of maintenance - no software to install or maintain on client devices

Any application claiming to be web-based which does not have these benefits is probably not truly web-based, but only web launched. Any application which has these benefits, but is not web-based (e.g. Java Rich Client Applications) could be considered where richer interface or functionality is needed.