HUIT Architectural, Design and Engineering Values

From HUIT Architecture Advisory Group

Contents

[hide]

Background

Part of coming together as a team involves developing a shared set of views about how the organization's tasks are to be accomplished. HUIT already has high-level values to guide our work. This article contains:

1. A set of recommended documentation for engineering projects. What matters is the content and thought processes as opposed a particular form and length of output.
2. An articulation of a set of technical values/principles HUIT engineers and others can use as they make decisions about the technologies they will buy, develop or otherwise incorporate into their systems, whether they are software components or key infrastructure elements.

This article is not intended as a rulebook since each system is different and will respond to a different set of priorities. That said, the article is intended to influence thinking about what characteristics of elements we acquire or develop that should be emphasized. It is assumed that requirements for the system have been established, documented and understood as input to the design process - in short, do the engineers know the problem/need that they are attempting to address with their efforts?

Objective and Benefits

This article is part of a larger effort to incorporate larger HUIT-wide objectives and values into the local engineering/design process. We propose these values with the objective of improving our services and our efficiency, both today and over the long haul. By imbuing the engineering environment with these values and technical requirements we hope these values will help us overcome HUIT weaknesses in terms of policy, procedure and approach in a way that allows HUIT employees to deliver better services faster, with lower cost, and greater reliability and agility. We anticipate that they will:

- Help remove roadblocks and bureaucracy where cost does not outweigh the benefit.
- Encourage practices that help people focus on the delivery of new and/or improved services.
- Reduce the trend of projects creating additional unneeded technical debt in the interest of short term success that carry high payback costs.

The greatest value in approaching development using the approach of clearly identifying key factors and describing their relative weights and documenting results that will influence development and operations is what may be learned in the process of doing the work by the team. It helps foster a clear view of what considerations are most important for the project and ensuring that everyone is ‘on the same page’. This knowledge should improve not only the engineering decisions taken in isolation but also the quality of the decisions in the context of the larger HUIT organization and the university.

Application of the Principles

While these principles primarily relate to the engineering process, they can most successfully be applied when there is broad buy-in from the entire organization. As a result, this section describes how the principles may be relevant at various times and in different parts of the organization.

Engineering

This document is not a recipe book. It is intended to help engineers ask questions about the software they are building or evaluating for use in their systems and provide some guidance on HUIT-wide principles for engineering. No one principle overrides all the others, the system/software under evaluation should be evaluated across all the design and system characteristics elements described using the perspective of the characteristics suggested. That does not mean that they all should receive the same weight. How strongly each is weighted is a local decision based on local factors. This process here is not just about a central set of values but using a more rigorous and well understood approach. The result of this evaluation should be documentation that includes:

1. A complete requirements document as suggested in the HUIT Engineering Project Requirements Checklist
2. An analysis/evaluation of architectural alternatives.
3. An analysis evaluation of technologies and where appropriate, a make versus buy analysis.
4. A detailed list of project assumptions and risks and mitigations.
5. A general description of the concepts of operation of the system.
6. Business case based on cost analysis or needed new capability.

Management/Senior Leadership

Engineering groups will need the support of leadership to take the time to do some of this work up front. They may also need training and other support. Investment in these areas will pay dividends in terms of efficiency of operation and more reliable systems when they are deployed.

PRC Review Process

There is nothing new in this document with regard to the process save the enumeration of the type of information that would be generally sought in any type of project review/approval process. If the documentation suggested here is completed, it should be easy to move through the process.

Non PRC/ITCRB Related Projects

Not all projects will go through the PRC/ITCRP process. In those cases, it may be more rather than less important to carefully review the principles in this document and apply them intelligently to your projects since there will not be an extra opportunity for input.

The Engineering Process

A key engineering value is that of following a known process for our work. Different efforts may place different emphases on different aspects of the process flow below, but the idea is a clearly articulated plan that is understandable, flexible and predictable. There are several elements of importance in the process that are captured in the following areas. The exact form; one or many documents, web pages, etc. does not matter. What is important is the thought process for each of the following areas:

1. Requirements - The HUIT Engineering Project Requirements Checklist details areas to be considered in a requirements document.
2. Assumptions and Constraints - critical to success and maybe one of the most important when the project has been completed. It can help readers understand why certain decisions were made and why the system behaves a certain way. It may be the hardest to fill in because assumptions are not always obvious. For example, since most of us are on campus most of the time we may think in terms of access from the ‘Harvard’ network. In some cases, this may be a reasonable assumption, in others it may be problematic. It is up to the business and engineering teams to carefully articulate these during the project so that they are understood by all - including end users. It will be one of the sections most carefully scrutinized in the review process since this area has had little to no visibility in the past. Constraints often have a significant impact on projects as well. For example, is a specific time frame or budget already been set, or are the types of individuals to be used on the project limited or otherwise constrained?

3. Technology Evaluations and Make versus Buy (trade analyses) - we should always be on the lookout for new technologies or approaches to our work. At the beginning of a project it makes good sense to spend a bit of extra time coming up for air and seeing if technologies, approaches or solutions have changed since the last time they were evaluated. This is also the time to see what the real costs are associated with purchase of a technology (assuming it is available) are versus developing in house.
4. Architectural Description - What are the main building blocks of the system, how do they relate to each other and to other systems? This is also an appropriate place to document issues and approaches to the deployment architecture(s) used.
5. Engineering Project Plan - This need not be the details of each engineer for each sprint anticipated between project inception and first roll out. It should be a general description of the amount of software, documentation and test engineering resources expected for the duration of the project and a reasonable description of the work they will be performing.
6. Operational Approach/Model - or concept of operations. This describes the system from the users perspective. This is not intended to be as comprehensive/weighty as found in other organizations (like the military). It should give the reader a pretty good view of what it would feel like to interact with the system. Smaller systems will have a very small description.
7. Risks and mitigation - the idea is to enumerate what you consider to be key risk areas and whether they are things you can impact or not. Try and describe what you can do to anticipate these areas.

Design and System Characteristics for All Elements

Factors in this section apply to all hardware and software elements we use in our systems whether they are developed by us, purchased from a third party or an open source solution.

Software/System Interface Standards

Open standards are strongly preferred to proprietary ones. Just because a well known language or protocol is used in an interface it does not mean that the interface is an open one. For example, a standard protocol like HTTP might be used to carry data that is not understood except by software from a particular vendor. Another, more subtle example is when a vendor indicates support for a protocol such as SNMP, but places objects in the PDUs that are customized for that particular vendor, or worse are opaque.

While documentation can mitigate a bit of this, preference should be given to products that support open, widely available interfaces supported by more than one vendor.

A key factor to consider when considering an API or other interface is that of change control. If a single vendor can, at will, change an interface then there may be legitimate cause for concern.

Note that these values apply not only to management interfaces but to all common services the software implements whether the software is open source, purchased from a third party or developed by HUIT.

Manageability/Instrumentation and Logging

Protocol requirements described above are one aspect of the manageability domain. Other characteristics are also important:

- Instrumentation that covers the full range of functions of the system through these standard interfaces is critical. To the greatest extent possible, instrumentation should cover the full range of configuration parameters.
- While Web/Graphic interfaces have a place they do not scale thus the requirement for full control via standard machine to machine interfaces is not diminished by the presence of a GUI. In addition, the machine to machine interface must be fully functional include at least as much input validation as provide GUI.
- Systems should have a facility for use of different interfaces for monitoring and control than are used for service delivery.
- Logging through standard mechanisms is also important and the system should provide facilities for the control of of log-level over the network.
- All of these control functions should be over secured channels.
- Open standards informed by generality are preferred to either proprietary ones or ones that use standard protocols but with proprietary data objects.
- To the extent possible, the fewer the protocols and data definition languages, configuration languages, etc. in our environment the better.
- A goal is to have only one configuration method per 'service element'. A service element could be a router, firewall, web server, operating system, etc. The granularity of the definition of the service element is within the purview of the management system.
- It is also a goal to have a single data collection system per service element.
- Good instrumentation is a prerequisite to but not sufficient for effective monitoring and control. A management plan should be created by operational personnel when deploying systems that includes how the systems will be monitored when deployed and the management software that will be used to provide the monitoring functions.

Support for Client Type Diversity

Systems that provide an equal user experience regardless of the users platform are preferred. Techniques for supporting client diversity include (but are not limited to) the following:

Tailored Application Behavior

Server applications should be able to gather information about their clients and alter their behavior to best serve the capabilities of different clients. For example:

- A web application can alter the formatting of its content in response to queries from mobile browsers or traditional desktop browsers.
- A SSH daemon can perform a protocol negotiation with a client and select an encryption method appropriate to that client's capabilities.
- A SMTP daemon can announce the optional protocol extensions it supports, enabling a client to take advantage of them if available.

It is a violation of the principle of supporting client/platform diversity to significantly disadvantage one platform over another. For example if a service has a basic set of requirements, under these values, then all platforms should have the same level of service.

Implementation of Multiple Transports and Protocols

Services should support multiple transports and protocols whenever feasible. For example:

- A website can publish the same content in both a RSS and an Atom feed; since both protocols are commonly supported, it is inappropriate to arbitrarily choose one over the other.
- A SMTP daemon can support LMTP as well as SMTP; while SMTP support is a basic prerequisite for an Internet mail relay, the addition of LMTP serves the smaller population of hosts whose circumstances are not well suited to SMTP.

Synchronization of State

When a system cannot be extended or enhanced to support diverse clients, it should synchronize its state with other systems that can themselves support diverse clients. For example:

- A proprietary database application may require a unique, proprietary client for access; in such a case it is appropriate to implement a connector or a translation layer enabling data replication between the proprietary database and a different database that supports a wider range of clients

- A GUI application may require a specific workstation architecture or operating system; in such a case it is appropriate to convert any output files into formats that can be read by other applications with comparable feature sets, rather than forcing collaborators to use the same workstation architecture or operating system.

Deployment Flexibility

Systems that do not assume a specific hardware and network configuration give us more flexibility in terms of our use of different approaches/technologies to ensure predictable performance and reliability. A simple example is a system that allows elements to be moved across different servers with minimal impact. In those cases where there are assumptions they should be understood and documented so those characteristics can be accommodated.

Integration with other Systems

There are several dimensions to 'integration'

- Integration with other like systems - for example in the management software domain, or in the domain of databases or calendar and mail systems.
- Integration with other elements of a system/service of which the system or technology under investigation is a part.
- Methods of integration - systems/technology vendors often cite interoperability or integration with competitive vendors/products or services. By looking at the method(s) of integration it is possible to understand the range of capabilities and difficulty a particular vendor supports via their integration approaches. Each project will have different requirements - the key is to understand the challenges/opportunities different technologies/vendors present in this area.

For HUIT to have maximum flexibility, there should be technology/application-specific and appropriate interfaces that work well across the above dimensions. For example, different mail and calendar systems are an inevitability. Our goal should not be to force a particular solution in this example (or in other like cases), rather we should work to provide effective integration technologies, whether acquired or developed at Harvard. This will help improve the ability of our customers to more effectively collaborate.

Availability/Redundancy

The challenge here is to strike the right balance between what one would like, and what we can afford. Consequently, the first element to consider is whether the requirements have been carefully thought out and articulated regarding availability and redundancy. In some cases, the vendor used for a system will limit options. For some systems the ability to keep users up when there is a failure is important. In this case, saving/sharing session information among the availability components is a key value.

Maintainability

Maintainability has several dimensions:

- Has the system been designed to that it can be deployed in a variety of topologies and configurations. Can it be configured so that some elements of the production environment may be upgraded while the other elements continue to serve requests without degradation of service. Related to this is the ability to test one or more systems in the environment that have had new software or other changes made using production traffic.
- Has the database (or other information repositories) been constructed with an eye toward extension and migration/schema updates?
- Modularity - to the extent possible, has the system been constructed so that sub-systems can be upgraded without taking the entire system off-line.
- Has information used for configuration/operation been externalized so it can be centrally coordinated?

Security

Like management, security is something that is part of the system design and deployment architecture, not added at the end. It should be factored into all aspects of the engineering process including how the systems are deployed. Many of our systems contain or process sensitive data. Because of this there are extra controls that are 'baked into' or environment. Care should be taken so that this does not impede development or testing. For example, we should avoid inclusion of sensitive data on any of our development environments. We should also strive for use of primary security infrastructures as opposed to developing new ones. For example, if a decision is not to use our PIN system, or the LDAP infrastructure on which it depends, there should be a compelling reason.

Characteristics for Hardware and Software Elements Acquired from Third Parties

In addition to the values/considerations that should apply to all systems deployed in HUIT, there are a set of characteristics we should consider when by use software developed by a third party.

These considerations apply equally to systems that are open source solutions or from commercial sources.

Support

The issue is not what is in the contract. Commitments to address a trouble ticket are nice, but what really matters is how engaged and responsive are the providers when you have a problem. How quickly can fixes be created, tested and deployed? How quickly can feature and bug fix requests be addressed?

Openness/Cost of Integration

How well with the system and all of its APIs work with the rest of the environment. All systems in HUIT are part of a larger ecosystem that includes server, network, application, middleware and other elements that must all work together for the services we provide to work correctly. How well a third party solution provides interfaces that are standard will impact our ability to integrate that system effectively and with reasonable cost.

Avoid Single-Source Solutions

Sometimes a critical element of technology or a particular function/performance set is available from only one vendor. There may be times when business requirements force us down this path, but we should do so only after a full analysis and understanding of the risks. Being tied to a particular vendor can inadvertently increase costs or force us to change course.

Vendor Flexibility and Responsiveness

How willing is a vendor to make special releases, and how fast can they turn them around. When you have a problem late in the day, how long does it take to get a useful response? The ‘useful’ is important. Most large vendors will respond per their agreement but they seldom make guarantees about getting patches done or even proposing a solution. The thing of value here is concrete action that solves a problem in the minimum of time.

Make Versus Buy Analysis and Overall Cost

There are no rules - let the numbers speak for themselves. When conducting the costs of technology or products, consider the entire lifecycle costs including development, deployment, consulting, hardware, network, licensing, storage, maintenance contracts, operational costs, flexibility, etc. In fact most of the other dimensions identified in this article.

Additional Engineering Values

1. HUIT values experimentation and investigation. It is helpful when people challenge their own assumptions and investigate alternatives and other approaches in order to improve our services and our efficiency.
2. We encourage talking with the Operations people and across all groups that have an interest in the project. We want to avoid engineering in isolation and ensure that people that are impacted have sufficient input. In short, we want to avoid decisions are made with local optimizations without sufficient consideration of larger impact.
3. The other side ensuring good input is making decisions even where there is no consensus. While consensus is nice, sometimes to make progress, not everyone will have everything they way they like.
4. Checking our work and information. Fact checking is important and helps ensure that we do not propagate misinformation.
5. The role and value of processes. There is a tendency to circumvent processes, often because they are cumbersome, duplicated, or simply do not work well. We should:
    1. Encourage critical evaluation of our processes and procedures to ensure there has been a reasonable cost benefit analysis done so that we are clear we have to have the process.
    2. Look for alternatives - often there are many different ways to achieve some goal (e.g., a certain level of security). Is the method used by the procedure/process the most effective?
    3. When there is a problem, try to work with those involved to come up with an improvement.
    4. In those cases where improvement is not possible, rather than circumvent the process - rethink you approach to see if there is a better way to achieve the results you require.
6. Everywhere, we want to encourage transparency. In all that we do, we should be open and honest, this includes letting people know about our assumptions, risks, limitations, etc. This means communications up and down our hierarchy as well as between peers.
7. Some of our process problems might be helped with greater automation. A lot of the delay (and error) in our systems is due to the high percentage of manual work.
8. In many of the service groups, there is no person charged with helping the customers. For example the people making routing, firewall, server, etc. changes are the same systems engineers tasked with keeping systems running. One approach to help address this might be to encourage groups to have a resident expert (perhaps on a rotating basis) that interacts with customers and is temporarily relieved of day to day efforts.

9. The fact that we do not keep sensitive data out of our development and test environments wastes a great deal of engineering time and costs $$. Sensitive data should be keep out of, or removed from the development and test environments.
10. The engineering organization needs people not only with specialized technical skills, but who are also generalists in the sense that they understand the environments they are building software for.
11. We have to take time to do the right thing, not only keep the environment running. For example, that means doing a complete root cause analysis for significant failures.

Getting Help

The approaches and values suggested in this document are a departure from what people are familiar with. If you have any questions, please send mail to:

HUIT Enterprise Architecture

Retrieved from "https://wikis.fas.harvard.edu/huitarch/HUIT_Architectural,_Design_and_Engineering_Values"