HUIT Enterprise Systems Review Summary Recommendations and Priority Actions

Jon Saperia

Version 1.3

February 14, 2013

Table of Contents

HUIT Enterprise Systems Review - Summary Recommendations and Priority Actions

Executive Summary

Since the June 2012 publication of the PIN systems review, two additional studies have been completed: iSites published in November 2012 and the recently completed Email and Active Directory review. In each study the approach was to broadly understand key elements of the system and identify conditions that increase the risk of service outage, impairment, or other significant concern. A list of recommendations to address identified risks was included in each review.

While each of the three systems and the services they provide are different, the reports show an overlapping set of conditions that threaten stability, cost effectiveness, function and user experience.

This report consolidates recommendations from those reports creating a basis for actions in a coordinated cross-HUIT plan[1]. The conditions that increase risk and recommendations to remediate them in the more than 70 pages of detail in the previous reports, may be consolidated into 5 areas:

1. Understanding our environment - In almost every dimension, we are 'flying blind' due to lack of data and analyses of the data so that we may chart an effective technology roadmap and service our customers most efficiently.
2. Service Management - how we define, provision, monitor, manage and report on services we provide.
3. Enterprise Architecture/Technical Principles and direction. This also includes understanding the technologies we have and their current and recommended applications.
4. Operational and engineering principles for our development and operations environments.
5. HUIT's technical teams and organizational structure.

Recommended Action Priorities

> *"One of the important implications of technical debt is that it must be serviced, i.e., once you incur a debt there will be interest charges.*

[1] This report does not recap the detail listing of conditions in each system that led to this consolidated set of recommendations. For those that wish to review the details, each reports is referenced in Appendix A.

> *If the debt grows large enough, eventually the company will spend more on servicing its debt than it invests in increasing the value of its other assets.[2]"*
>
> *- Steve McConnell*

There are a number of actions, that if implemented, would be a downpayment on HUIT's technical debt, the result of which would be improved service reliability, functions and user experience. Additional detail on each of the recommended actions below is found in the Priority Actions section.

1. **Understanding our Environment** - We should begin active data collection from our environment that will tell us about our users, access technologies, and use patterns. Of equal importance, HUIT should assess the technologies we currently employ and the services each technology supports. From this we will be able to recommend appropriate technologies for services in a globally coordinated fashion as opposed to the localized approach that is our default. These evaluations should be made prior to pilots or significant commitments to technologies, especially when those technologies/products tend to 'lock' us into specific platforms. For example, given the high and growing majority of non-Microsoft mobile devices in the Harvard environment is Microsoft Office 365 the most effective way to provide email and calendaring functions? In the same vein, is the Microsoft Office Suite the most cost effective and user friendly way to provide the whole of the Harvard community with office productivity applications?

2. **Integrated Service Management System** - Uncoordinated configuration changes are the root cause of many of the service outages experienced by the systems studied in the last 9 months. Examples include changes to our WiFi address space that resulted in an iSite outage during the fall 2012 startup or the frequent outages across services that are the result of uncoordinated ACL changes in our hardware and software firewalls, the most recent of which was on February 1. A new PIN Server was unable to function because of an out of sync clock due to a configuration error. This outage was yet another example of the need for integration, not only of functions such as configuration and monitoring but also across dependent services. In this case, the configuration error of NTP (Time Service) impacted the PIN system and in turn applications that depended on the PIN system[3]. The system we construct must be able to

---

[2] http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx

[3] This failure highlights another issue. The specific problem had to do with customizations to existing systems to change the used NTP port that were not applied because we do not keep these configurations in a source code control system and the result of using a standard port was not compatible with our content switch configuration.

not only services and their constitutive sub-systems but also monitor in a coordinated fashion all of these elements so that rapid fault isolation and root cause analysis is possible. Together these actions will reduce the number of failures we have, and when they do, reduce their duration.

Since configuration is cornerstone of an effective integrated service management system. For this reason, the starting place for our efforts should be an immediate effort to create an automated system with an initial goal of providing some improvement in the area of system deployment and configuration.

3. **Enterprise Architecture** - HUIT should begin an ongoing effort to identify and infuse awareness of new technologies, operational approaches, and architectures across the organization. This effort would also help to ensure cross HUIT coordination at a technical level as well as integration with University-wide efforts where appropriate.

   Previously cited HUIT studies have shown how we deploy our services and that complexity is foundational to their stability, cost, ability to scale to meet increased demand, and fault isolation. New technologies, approaches to security, and service management techniques have emerged since some of these systems were conceived. The recommended action is to consider new trends that include not only cloud-based alternatives but also new management practices, approaches to security and other factors. The output of this effort would be one or more prototype example designs that can be used for new service development work or serve as a platform for the migration of existing systems.

   This effort is not a substitute for local engineering judgment and control. It is intended to avoid duplication of effort, create a common approach to the way we engineer our services and acquire, develop, deploy and manage the technologies important to our services.

4. **Operational Principles** - How we operate our systems is as important as the systems themselves. We should establish and follow guidelines for keeping production software and systems current. Additionally, we should treats these systems as ‘code’ where we include configuration control principles commonly used in the software engineering discipline to the management of change of our deployed systems. Our actions on these systems must become more automated if we are to achieve any kind of scale, cost-effectiveness, and reliability.

Finally, our current approach to develop, test, stage and deploy production systems must be simplified. A more streamlined system will increase efficiency and will reduce the number of 'roll-backs' of changes we experience in production systems. The envisioned streamlined system will expose new code to the rigors of a production environment sooner in a controlled way which will reduce 'surprises' when code is placed in full production mode.

5. **Organizational Experimentation** - The fact that we have many stovepipes within and across HUIT is understood and has also been described in the previous documents. They add cost, complexity, and reduce overall effectiveness. The question is what to do about this while minimizing organizational churn? While ITIL has something valuable to contribute in this regard, the recommended action to address these issues is to do some experimentation with new projects to see if we can identify a nucleus of principles around which the technical organization may evolve. In particular, we should select one or more new projects and create teams that would be more service focused. The approach of selecting one or two test teams as opposed to making large organizational changes will be less disruptive to operations and allow more nimble experimentation which should produce better results.

Report Details

The remainder of this paper is divided into two sections. The first contains recommendations that are general enough to apply to more than one of the previously studied systems and are of sufficient criticality that they merit attention in a short time frame. The second section presents a set of criteria that were used to identify a subset of priority actions in areas of most critical need and which do not require large amounts of additional resources or significant organizational change..

Consolidated Recommendations

This section presents a consolidated list of recommendations from the previous reports. In most cases, the recommendations apply to more than one of the systems that were studied.

Understanding Our Environment

> *"If we could first know where we are and whither we are tending, we could better judge what to do and how to do it."*
>
> *- Abraham Lincoln*

To the extent there is a unifying principle behind recommendations and actions in this document, it is that we do not know where we are and have not charted a course to a

desired state. These conditions and states exist across most of the dimensions of our environment: from high-level principles like an appreciation of our users and their changing needs to an understanding of how our organizational stove piping tends to hide information. We are often either, unaware of, or unwilling to take advantage technical options available to us. Finally, we are often in the dark when it comes to understanding how changes to one element in a system impacts the rest of that or other systems.

We must develop systems, procedures, organizational and other structures, that will let us know where we are at any moment. We should do this, not as a discrete exercise, but as an inherent part of how we conduct the business of design, development and operation of HUIT services.

Existing Technologies - Use and Relationships

Because of our stove-pipe organizational structure and systems, we lack a global understanding of the technologies currently in use and which ones should be used to accomplish certain tasks and which ones should not. Previous investigations show that individual groups produce services based on the technology set they have without a full understanding of the larger technical environment. For example, there are multiple technologies that can be used to provide access control to applications and the data they operate on. We should develop recommendations about which technologies should be used for specific functions and services to address this and ensure the information is widely understood by the entire organization. This is especially true when the system has broad impact as in the case of the previously cited email example.

Customers

In addition to understanding and documenting requirements of systems our customers request, we need to understand problems they are attempting to solve more broadly. This improved understanding will allow us to engineer systems that more efficiently (for our customers and HUIT) address current and future needs.

Access Trends and Technologies

We should have an ongoing set of measurements in our systems that help us understand:

- Usage patterns.
- Periods of peak demand (beyond the obvious we know about).
- The technologies clients use to access our systems - specifically:
  - The operating systems and applications that are used to access systems.
  - Mode of access - such as desktop device, mobile, tablet, etc.

- Ownership of the devices - a significant percentage of the platforms used to access HUIT based services are not University owned, they are personal devices used to access university systems either remotely or while at a campus facility, This should be factored in from security, performance, and other perspectives.
- What features are valued most and how they are used.

Current Systems - Data Collection and Analysis

In addition to monitoring for all the characteristics in the Access Trends and Technologies area above, we should continuously collect data that we can use to project:

- Future demand.
- Changes in use patterns and access - for example when there are topology changes that impact the performance of our services.
- Fault Patterns.
- Interactive performance - latencies, overall responsiveness.
- Service Level capacity and which elements in the delivery of the service are approaching thresholds of performance.
- Configuration changes through the system and how they impact performance, reliability and system wide latencies.

HUIT Procedures and Policies

All over HUIT, people are frustrated with our procedures and policies. While they are often in place for sound reasons such as: safeguarding security, ensuring operational stability, or meeting University or governmental regulations, many seem to have significant side effects such as:

- Adding significant delay for what are obvious (at least to many involved) change requests.
- Abdication of personal responsibility.
- Inhibiting experimentation, investigation, and innovation.

To address this condition of frustration and inefficiency, we should:

1. Identify extant policies and procedures, and for each; identify the problem/ issue that was to have been addressed by it.

2. Review the polices for effectiveness, cost, alternative approaches and need with respect to the issue it was conceived to address. It would not be surprising to discover that, in many cases, the problem that was to be solved may be more simply solved or addressed by other means. This set of issues is unlikely to be fully addressed outside the context of organizational change, but some improvement should be possible absent large-scale organizational change.

3. Put a mechanism in place so that new policies and procedures are independently reviewed from a broad perspective for impact, alternatives, and effectiveness ***before*** implementation. This review should include the line people that will carry the burden of the process or procedure as they may be the people best able to identify side effects and better alternatives, or even if the procedure is being proposed as a result of some secondary or tertiary issue that may be better solved directly.

New Technologies

HUIT should also work to understand the environment external to our world. This includes not only University-wide technology application and development but also new technologies that may be more effective than the ones currently in use.

Integrated and Automated Service Management

A good deal has been written about 'service management' and depending on the source there are different emphases. The following sub sections are the foci of service management recommendations for HUIT.

This report assumes progress will continue to be made in the service incident and change management process. Indeed those processes can not be effective without the data that an automated service management system recommended in these pages can provide.

For our purposes, service management is both a process and a set of software. It spans:

1. **User requirements and service level agreements** - How we collect requirements for our services and report on our success in the delivery of those services.
2. **Engineering** - Translating the requirements in engineering to operational systems.
3. T**hird party requirements** - Part of the process of engineering a solution is an examination of technologies that may be used to implement the service. To the extent we outsource part of our services or use Commercial or Open Source technologies, we should ensure that these technologies also meet our requirements for elements in our service management system - as well as

overall architectural requirements also described in this report. In particular, do they offer management interfaces that are open and follow known standards?

4. **Implementing a Service Management System** - key recommendations for characteristics to avoid and require of an automated service management system for HUIT.

Requirements and Service Level Agreements

HUIT should develop and implement a requirements gathering process that includes our constituents and documents their expectations. This is not a static process done once and then forgotten until the next funding cycle for a project. It is an ongoing part of our daily operations to ensure we do not become disconnected from our users. The process should also provide engineering groups with the opportunity to innovate and propose new approaches by including rapid prototyping as input to refined requirements.

It is not surprising that we do not have a coordinated data collection environment. A contributing factor is that we do not have identified and clearly articulated service level agreements and the associated metrics/objectives that would be used to verify service levels. It is the definition of service level agreements and objectives for services that will drive the identification of the types of data that must be collected in a coordinated fashion.

Service Level Reporting

For each element of software/hardware used in the delivery of one or more services, engineering should identify details of the instrumentation that will be used, in the case it already exists, or created where it does not exist. These 'low level objects' become the data elements that are combined to verify service level objectives in the Integrated Service Management System.

Engineering - Translating The Requirements for Service Management

After high-level requirements have been collected, engineering investigates technologies at hand, as well as those that may be potentially useful to create an implementation that meets both the user visible functional requirements and requirements for effective service level reporting.

Third Party Provider Requirements

HUIT has begun to use an increasing range of third party providers to help deliver services. While the results have been mixed, the trend toward further outsourcing is likely to continue, at least in some areas. To the extent that third party providers are used, we should apply similar requirements for service levels, management information and responsiveness that we apply to ourselves. This includes the requirement that third party providers allow HUIT access to at least some real time information when we have outsourced some or all of a service.

In the case of software/hardware systems we use in the implementation of our services, from databases and middleware to the servers on which these elements and the higher level software depend, we should ensure that the technology meets our ability to collect data for service level management in a manner that works with our overall approach, as opposed to a closed system whose data is difficult to integrate with data from other systems. Most of the larger vendors have closed systems that are impediments to the development of the integrated and automated service management system envisioned in this section. These vendors include Microsoft, Cisco and Oracle.

Implementation of an Automated Service Management System

The market is full of equipment and software vendors that claim to have excellent service management software. These products are invariably fatally flawed by one or more critical factors, including:

- Require the addition of proprietary monitoring software on deployed devices.
- Does not support existing standards.
- Manages only a small portion of the service environment such as the operating system, one or more network elements, database, middleware, or specific application.
- Management functions are limited, e.g., only supports configuration or fault management.
- Is limited to a single vendor, for example supports Cisco but not Juniper or F5.
- Has a proprietary data storage approach and/or API that make integration with out data elements problematic.

What has all this to do with the recommended service management system beyond enumeration of characteristics we should avoid in a HUIT system? It points to key characteristics of the system we require:

1. Our service management software should be driven to the greatest degree practical by our need to deliver integrated services to our customers and manage all the hardware and software elements that are constitutive elements of that service.
2. Our services should be expressed and reported in terms our users understand and the software system should allow the separation of the service from specific technologies used to implement a service. It should allow us to 'plug in' one or more different technologies and roll them up. For example, how reliable are our Mail services in aggregate? This as opposed to how well did we do with a Google or Microsoft based service. From the aggregate, we can zoom in on the details.

3. As with the services the system is designed to manage, the requirements for this software system should be listed in a technology independent fashion - as opposed to a set of product features from a particular vendor[4]. Engineering's responsibility is to translate these requirements into the particular technologies that are selected at any point in time.
4. The system we construct should be flexible enough so that it can be adapted to evolving technology without hiring armies of consultants or re-engineering the system.

Service Provisioning and De-provisioning Processes

Integrated service management requires that we have a clear process and software to allow us to deploy services across our environment as opposed to configuring one element at a time[5].

Automation and Process Control

Whether we are talking about event/alarm escalations, out of policy configuration changes, or the addition of more resources to adjust to increasing demand, the complexity of our environment has demonstrated that a manual approach to configuration/service provisioning is error prone, costly, and under-responsive. Our Integrated Service Management system should have facilities for automation of all types of actions (with human confirmation in cases that are configured - policy). This facility will also include configurable escalation facilities to ensure that certain events do not fall through the cracks.

Integration of Data

Only through the integration of all types of data will we be able to address conditions that have led to a number of outages and service impairments reported in the previous studies. It is recommended that the following types of information be integrated in an automated service management system:

- Service level data (service definitions, technologies that implement the service, metrics, network, server and application elements that support the service).
- Global topology and inventory of systems and software.
- Performance/accounting data.

---

[4] Understanding a broad set of capabilities from multiple vendors is always instructive.

[5] It is understood that in the end we will have a series of configuration actions that take place on potentially many devices. The key point is that we need to think of services as an integrated set of systems working together to provide a service and services need to be provisioned and de-provisioned with this in mind. Our failure to do this is the source of numerous outages documented in the previous reports.

- Configuration data - including access to historic information as well as planned/staged changes at a service level and individual element level.
- Fault and service level data and history.
- Data related to security including access controls to the information in the system.
- Policies/rules that govern system data collection, analysis, reporting, etc.

User Role Diversity

In the section on organization, stove pipes are discussed. These stove pipes increase system complexity and contribute to failure and time to correct faults when they occur. To address this, the automated service management system should be HUIT-wide in terms of scope. A number of types of users such as operating systems administrators, network and middleware administrators, application engineers and development should have access to the system. The scope of the information will be controlled by configurable access rules. This diversity of users suggests that we also have an adaptable interface suited to the different tasks different types of users will have.

Enterprise Architecture

Enterprise architecture is an umbrella for a number of recommendations designed to better coordinate our technical efforts and impart new alternatives and ways of thinking.

Architectural Principles for HUIT

A broad set of principles that guide our development and deployment approaches should be developed. They will act as guideposts for the engineering community to use in the countless decisions made as each service is developed, deployed and enhanced. Topics include, but are not limited to:

- **Understand the problem to be solved** - we want to ensure that the technology is appropriate to the problem. That can only be accomplished if we understand problem set and available technologies well, locally and in the context of the system as it relates to the larger HUIT/University environment.
- **Service definitions and technology** - separate service definitions from the potentially many technologies that could be used to implement them. Whether we are talking about mail, calendaring, firewalls, identity managers, routing and load balancing systems, and a host of other simple and complex systems HUIT provides; we should first understand what the users want and need before selecting a

technology and moving forward. Once we know what we need from a service perspective, then we can evaluate the technologies using a variety of factors to identify which is the best match for our needs.

- **Considering alternative/new technologies** - We often only consider technologies at hand or those that are currently believed to be endorsed by management. When we do our evaluation of technologies (against service/user needs) we should make a point of investigating alternatives that may be available that we have not used before.
- **Systems components should integrate well** - Most of the systems we create could potentially include many components including HUIT-custom elements, open source and commercial options. How well a component integrates with other elements of the system as well as how well it might integrated with other systems across HUIT, the University and beyond are important considerations. This extends beyond the previously mentioned service management requirements already noted. It also applies to how well the system will integrate with operational elements. For example, with a university wide program/system for access management.
- Cross organization impact - We should always consider all options in a thorough technology/design effort. That said, we want to make sure we are not making a local optimization that will cost the larger HUIT organization and the University more in the long run.
- Consider technical debt - not all technical debt is bad. If there is 'technical debt' incurred in the decision, make sure that all parties understand the consequences and that the decision makes sense in a local and global context. People should be prepared to pay the 'interest'.

Software and Hardware Selection

We require more critical thinking in our technology selection process. There has been a tendency to select what has thought to be the easiest[6] choice. HUIT should foster an environment where our engineering groups are encouraged to push back and ask questions - even against specific recommendations/decisions only recently made. In addition to the intended benefit of better technical decisions for all of HUIT, as opposed to local optimizations, we may be able to begin to address the general

[6] Easy could be what people thought management supported, what they were familiar with, etc.

feeling that management does not listen to technical people. With this freedom will come the responsibility for, and investment in, the engineering choices made.

The criteria we have used for hardware and software selection recently has left some alternatives unexplored, particularly in the realm of open source software (see architectural principles above). In addition to functional, cost and performance requirements, additional factors we should consider when making technology decisions are in the following list. Some of the items come from other areas in this report.

- What is the problem we are attempting to solve and how well does the specific piece of technology address the problem?
- Manageability - we want services that can be effectively managed and this should be part of the design and service level requirements. This includes general manageability requirements as well as facilities for logging and integration into the proposed service management system described in the previous section. This requirement for manageability applies to each element we include in our environment.
- Openness of the system - closed systems can be deceptively appealing. We should consider the cost of integration with other technologies, support, and risk to future systems ability to evolve as a result of being 'locked in' to a specific vendor, technology or product.
- Flexibility/Expandability - how easy or difficult is it for us to make modifications to the element, and what are the costs and logistics associated with such changes.
- Vendor/Technology Independence - The relative weight given to this or any other criteria will change in some circumstances. As a general rule, we should avoid technologies/vendors that push us toward dependence on their solutions (which includes most of the vendors we deal with). Related to this is the issue of migration. Vendors change their product mix and sometimes new products are added or whole product lines are discontinued as in Cisco's recent decision to stop providing load balancing solutions. To the extent we can look into the future we should ask ourselves - if we have to abandon this platform, how painful will it be?
- Support and Maintainability - There is a misperception that large established vendors will give us better support than smaller or open source alternatives. In terms of responsiveness this is often not the case. These larger vendors are often

'safe' choices but in the long-run may contribute to our problems. When looking at products, we should evaluate not only our history with extant vendors but also examine how responsive alternative vendors are. Responsive is not managing the process which is what larger vendors often do. Responsive is providing a timely solution to what drove the support call in the first place.

- Large organizations often produce more complex solutions - part of the criteria we use when evaluating a technology should be: how easy it it for HUIT to understand and manage?
- How easy/hard/costly is it for HUIT to find people to work with a specific technology?

Reliance on Third Parties

After completion of the three studies that led to this report, one is left with the impression that HUIT believes the best way to solve many of its cost, personnel, and service related problems is through greater efforts to acquire whole solutions from large vendors, or purchase entire services in the cloud.

Third party solutions should certainly be an element of our technical strategy, but it should not be thought to be the silver bullet that will solve our cost structure, personnel, and other technical challenges.

Whether we are thinking about placing key HUIT systems in the cloud or simply acquiring/developing a system for internal use, we should consider the strategic importance of a solution, need for integration, customization, responsiveness and other factors as we consider our technical options.

Design and Complexity

> *"A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away."*
>
> *Antoine de Saint-Exupery*

Our systems and environment have become increasingly complex over the years without a corresponding improvement in reliability, or in some cases, performance or function. While there are many causes for this, including staff attrition of those that created the systems, or systems that have undergone one or more technology changes, etc., complexity increases risk of service outage or impairment and inhibits our ability to rapidly diagnose, repair and prevent failures. As we evolve existing systems, or make plans for new ones, we should evaluate whether there are simpler,

more robust architectural approaches that could applied than those we have used in the past.

As with other architectural principles, and service level management concerns noted previously, we should make sure to understand the requirements ***and assumptions*** on which the system is based before attempting a (re)design. Key factors include:

- What are the scale requirements now and in the future for the system? That is, we should not be asking what is the current and anticipated scale[7] in isolation of the current deployment model. We should also consider the facilities the architecture provides for the addition of more resources via different topologies or architectures.
- The level of availability/redundancy that the business requires and HUIT can afford.
- Topologic complexity - one of the difficulties with some of the extant systems is that they are so topologically complex (even within a single facility) that trouble shooting and addition of new computing resources are problematic. We should balance needs against complexity.
- Cloud based systems - deployment of services in the cloud can be a cost effective solution for some of our systems. We should understand that in some cases, this approach will add to rather than reduce overall complexity. For example, when messages must first pass through the HUIT infrastructure before being routed to an external environment for processing. This is not an argument against using this approach for our services, but a recommendation that we consider how that impacts our service management and architecture when we do so.

Complexity Related to Security

Our objectives and requirements related to security have been realized in our deployment of technology, policies and procedures in such a way that management of security concerns has become time consuming and errors in the process often result in service failures (e.g., when an ACL on a host based firewall is not set correctly).

This is such a pressing issue that security is one of the key areas mentioned in the priority actions section at the end of this document. See the section on security.

[7] However it is appropriate to specify scale such as transactions per unit time, simultaneous users, etc.

Architectural Team - Architectural Currency

The activities recommended in this enterprise architecture area should be ongoing. They should be conducted as part of an informal technical advisory group that will keep the principles up to date, act as a sounding board for major initiatives, and investigate and recommend new technologies for possible application to HUIT services.

Operational Principles

Just as there are key architectural principles that should guide our systems design, so too should we develop and support principles that are useful to the operation and maintenance of our systems. Operational principles, like the architectural ones should be used as guideposts rather than absolutes. As with all our decisions, when we vary from the principles, we should understand and document the reasons for doing so.

Software Currency and Alignment Across Systems

Any system that HUIT supports should be maintained consistent with best practices for production systems. Specifically, we should have a more aggressive program of software maintenance so that our revisions do not become so out of date that either they are not supported by the vendors that supplied them or support is more problematic because we are not current.

This is not to suggest that we exist at the bleeding edge of vendor release schedules. There are conditions where it makes good sense NOT to upgrade to a new release as in the case where a required feature is not supported or performance or reliability are reasonably questioned. These conditions should be the exception rather than a rule.

A reason often given for not keeping releases current is that there are other priorities. Evidence suggests that when we do not keep our systems current, the technical debt we incur has a high and unpredictable interest rate.

To the extent we find it is to cumbersome or time consuming to keep production software current (and this includes all elements of a service from operating systems to high level software of the servers and network elements) we should investigate why this is the case. It may be that our methods and procedures for upgrade, or the architecture for the systems themselves require improvement.

Separate Engineering From Production Environments

One of the greatest impediments to effective engineering within HUIT is the cumbersome processes that surround our development systems. In some cases this is because sensitive data has been allowed to flow down from production environments thus properly imposing additional security requirements on the development and test systems. To the extend this mingling has been allowed to exist, immediate steps should be taken to isolate and remove sensitive data - *even in advance of critical projects the team has on its plate*.

Once the development and test environments are 'clean' from a security perspective, they should be treated as development systems and should be almost wholly under the control of the development organizations. Procedures for change and maintenance can be tailored to suit this environment increasing productivity, and HUIT resources consumed by process inappropriate to a development can be freed to perform more useful tasks.

There is a side benefit of creating a closer connection between development and the servers and other infrastructure elements on which it depends. Too often application engineers are kept in a bubble and do not know much about the distributed environment or software systems on which their applications depend. Modern engineering techniques can isolate them from some of these considerations, but the degree to which they are kept too much in a bubble, they will not understand their environment and are more likely to design and implement systems that do not operate as efficiently as they could.

Levels of Support

While the HUIT FY13 rates offer a variety of service levels, there is consensus that many rates are high. HUIT should investigate creating lower-cost options for different service levels (since not all services have the same level of business criticality). This investigation should proceed along with understanding existing costs[8].

Infrastructure as Code

The documentation and configuration of all the elements of our infrastructure and the software that rests on it should be in a source code controlled environment. The current practice of using Remedy[9] as both a trouble ticket system and way to propagate certain configuration changes is inefficient and error prone. We should begin the migration from both software and process/policy perspectives (see the section on Service Provisioning and De-provisioning and the entire Integrated Service Management Section for additional details) to one where all details of the configuration of our services are in a source code control system that can be (re)deployed without requiring (yet still permitting) manual intervention . Operational procedures should be developed to support the availability and integration of these new technologies.

Root Cause Analyses and Continuous Improvement

Our ability to improve our systems performance and reliability is significantly impended by a number of factors:

---

[8] For example, the FY13 charge for 'Monitoring-Ping' is $41.67 per server. Without details of all that is included in this, it is hard to evaluate how cost-effective is it - but on its face, this seems high.

[9] There is currently an investigation on a new trouble ticket system. It is not clear to some what problem the new system is designed to address and what the criteria for success are for the new system.

- Our tendency to reboot to solve a majority of problems.
- The lack of a consistent HUIT-wide logging and reporting infrastructure that is integrated.
- Lack of a service management system.
- Over compartmentalization of data and access to it. For example OS information, network details, database details, middleware and application layer data are all often collected by different departments and not integrated.
- Under valuing preventing a problem down the road. This is seen as a luxury that can not be afforded.
- Architectures that do not facilitate 'live' upgrades.

While the service management group does record 'root causes' for major outages, this is mostly a documentation function. HUIT should foster an environment where the technical people can work together cooperatively to understand persistent problems, their root causes and come up with solutions to them as opposed to simply feeding the process. This may come only with organizational change - see the New Organizational Structure(s) section.

The root cause analysis is incomplete without a action section at the end of the analysis. That is: the specific steps that will be taken to reduce the likelihood of a similar failure in the future for this or any other system that relies on the failed element or elements.

Testing, Simulation and Environment Levels

In addition to the separation of the development and testing environment from the stage and production environments recommended above, we should develop new approaches to testing that would better prepare our software for the rigors of production operation. It is when software is placed in production service that we often uncover problems that have to that point been obscured. We should begin to develop architectures that allow production testing on live data with real traffic. The idea is that we have sufficient capacity and traffic isolation capabilities in our deployment architectures so that when we wish to test a new component, a production system (or group of servers) is migrated to that new component. It is run with a small, then increasing amounts of traffic until it reaches full load. In the event of a problem, the system(s) with the new elements can be taken out of the server rotation without risking the capacity of that site[10] to deliver the service under test.

[10] The most obvious examples of site are 1 Summer and 60 Oxford Streets. The general principle would apply to a cloud based environment where multiple sites were used for reliability purposes.

Automation

Many of our procedures are manual and error prone (see the previous discussion on Infrastructure as Code). New systems are needed for this approach, but we should pay close attention to the adaptation of new procedures to new systems as they are deployed or many of the benefits of the new systems will not be realized.

Distribution of Control

HUIT must find new ways to securely distribute control of elements to those that most closely understand the needs of the system. For example, while a central group may have considerable database, network or middleware expertise, that does not mean that they must be the ones making the changes in these components. Indeed, the application developers that use a middleware server such as JBOSS may have a better understanding of the changing HEAP requirements of a middleware server in certain traffic patterns than the middleware experts do. Distributed control has worked well in many environments and we should actively pursue that approach in HUIT.

Organization and Personnel

> *"...organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations."*
>
> *- Melvin E. Conway*

There are many opportunities for improvement that an organization as large and complex as HUIT will have. This section is only focused on those aspects directly related to risks associated with our service offerings.

Organizational Stovepipes

These stovepipes exist not only across the main departments in HUIT but within them as well. Previous sections have described the 'data islands' that are often created as a side effect of these stove pipes and how they make fault isolation, root cause analysis, and other functions problematic.

Previous reports have recommended teams based on services as opposed to specific technologies (e.g., networks, servers, applications, etc.). While there are basic elements of this in our current environment, our organizational structure seems to inhibit these ad hoc teams from functioning as desired. At the same time, a wholesale reorganization would be problematic. Two approaches in combination might improve this condition.

1. Create a new team structure on new projects that are amenable to this approach and keep the team outside the current organizational structure. This requires more than putting a group of people together. Ideally, they will have control of virtually everything they need to develop or restructure a system from building access, control of network and computing resources, source code

control systems, implementing security and legal requirements for data privacy, etc.

2. Use the new software systems for Integrated Management as they become available as a way to drive change (see A HUIT-Wide Automated Service Management System).

Rebalancing Our Values

Employee frustration with processes and the feeling that technical input was not sufficiently valued as it moved up our deep organizational structure was noted earlier in this report. We have to do a better job of appreciating and fostering, technical excellence up and down our hierarchy. The focus has to move from policies, procedures and organizational boundaries to 'doing the right thing'. Said differently we should encourage thinking globally (for the global good) and acting locally with less hypersensitivity to organizational boundaries.

Staff Retention and Advancement

We continue to loose exactly the people we wish to retain and it is having an accelerating impact. Monitoring the situation is not sufficient, new ways have to be found to directly address issues. This is not a communication problem where either we have not communicated values or instructions to the organization or have not asked for input. Demonstrable actions that show how previously expressed concerns are being addressed will be the best action to improve morale.

A frequently expressed concern is that even though people feel like they have a full plate, they are often asked to chip in and help on one organizational initiative or another while maintaining performance on the rest of their job responsibilities. For staff to effectively contribute to initiatives, including the ones in the following section, they must be freed up to do the work.

Priority Actions - Implementing the Recommendations

The depth and breadth of issues previously identified are such that remediation would be a multi-year project even if we were not operating in a resource constrained environment. The action items in this section, that expand discussion from the executive summary, were chosen using a number of criteria:

1. **Cost** - To make meaningful progress, incremental resources will have to be applied to these priority actions. Given a flat budget, that may mean that some other 'priorities' may have to be deferred. That said, the items proposed in this section are included because they are believed to be achievable with modest investment.
2. **Interest payments** - related to cost are the 'interest payments' we much make in a particular area of technical debt. In some cases, the interest we pay is in increased down time, or frequency of outages. In short, if we do not address a

particular area how risky is it, and/or how much will it cost to address in the future.

3. **Practicality** - A number of factors fell under the umbrella of practicality in addition to those cost when making selections for the small number of projects in this actions section. These include:
   (a) Immediate disruptiveness to the organization.
   (b) Level of effort/mind share required versus the benefit result.
   (c) Likelihood of success.
   (d) Sustainability - how likely is it that the beneficial result of the action can be sustained.
4. **Overall benefit** - Leverage points/criticality - The actions proposed in this section are thought to not only address one or more specific issues but also impact our operations broadly.
5. **Technical Exemplars** - Is it possible that some or all of the result of the proposed action could help other activities?

In short, the following projects are deemed to be low-hanging fruit, though while not free, they can provide meaningful lasting benefit to HUIT in furthering its mission.

A HUIT-Wide Automated Service Management System

> *"If you want to teach people a new way of thinking, don't bother trying to teach them. Instead, give them a tool, the use of which will lead to new ways of thinking."*
>
> *- Richard Buckminster Fuller*

Lack of a HUIT-Wide Automated Service Management System contributes to the cost of operations (high-labor content) and long-term technical debt. The technical debt interest payments include higher cost for services deployed and increased outages and operational costs. Creating a service management system may be one way to have a positive impact on the organization as a whole, increasing overall effectiveness. It can also address one of the problems we have; solving the same problem over and over again, rather than solving a particular issue once and then moving on.

Recommendations for an Automated and Integrated Service Management System detailed some of the key characteristics we should incorporate and avoid in such a system. There is no third party solution, set of consultants, or open source code that individually will create what is needed. This is as true for HUIT as it is and has been for other educational, government and commercial environments. What we can do is create a plan and a vision that will lead to incremental improvements in the shorter term and a significant improvement further down the road.

The action recommended here is that we immediately begin with the cornerstone problem, coordinated configuration management of our infrastructure and the

services that rely on it. A separate plan will be produced over the coming months with details on the approach and the resources required for success.

It is hoped that a side benefit of this approach will be "a tool, the use of which will lead to new ways of thinking."

Success Criteria

Deliverables:

1. An initial software deliverable that coordinates ACL configuration control across multiple systems.
2. A road map for the implementation of the larger service management system described in this document.

Benefits:

1. Reduced down time as a result of configuration errors.
2. Improved operational efficiency as a result of automation.

HUIT-Wide Technical Principles

The enterprise architect should lead a small group empowered to create recommendations in key areas identified in this and earlier reports, topics include:

- Technology recommendations for services.
- HUIT Architectural Principles.
- Separating services from technologies.
- Vendor and technology selection criteria.
- Understanding service dependencies - many services we offer are in turn dependent on lower level 'infrastructure services' like the DNS, NTP, Routing, etc.

  Understanding dependencies of our distributed systems and planning for them is something that needs to be infused into all our systems.
- Architectural Relationships between Inter and intra HUIT systems and technologies.
- Manageability objectives and requirements.
- Understanding our customers and technologies.

This team will continue on an ad hoc basis to provide technical input to new proposals for expansion or revision of existing services, or development of new services.

Success Criteria

Deliverables:

1. An architecture web site for posting the information the team develops.

2. Documentation on each of the topics above.
3. Introductory meetings with or presentations to key organizations to walk through the material and start an ongoing discussion.

Benefits:

1. Better technical decisions on our projects.
2. Less overlap of function between systems.
3. Better fit between our solutions and customer needs.
4. Simpler more effective architectures are less costly to keep running and will have better uptime.

Deployment Architectures Review and Recommendations

Detailed studies should begin as soon as possible in two critical areas. Input would come from the team working on the architectural principals since these activities are related.

Security/Deployment Architecture

The requirements for the protection of data and the need to secure our systems and services is not in question. What we must do is once again look at and understand the requirements. Given the evolution of technology, we should evaluate alternatives to our cumbersome and error prone systems given the clearly stated requirements. The end result of this would would be a report that re-articulates requirements, includes recommendations for new hardware and software architectures, procedures and systems that would meet security requirements more efficiently.

Cloud, Disaster Recovery and Availability

New opportunities for architectures present themselves when services are operated in the cloud. This has significant implications for our disaster recovery approach, indeed is raises the question about whether a separate site for disaster recovery is needed. Certainly, how we construct systems for high availability will also be impacted. The team should begin a study to evaluate options and recommend architectures as appropriate for several different types of services. The recommendations should also identify approaches relevant to disaster recovery.

Success Criteria

Deliverables:

1. Delivery of the security/deployment architecture recommendation.
2. Delivery of the cloud, disaster recovery and availability recommendations.

Benefits:

1. Fewer outages as a result of security configuration errors.

2. More effective, less costly monitoring and trouble shooting due to reduced complexity.
3. Reduced overall cost of operations for services that can use these new architectural approaches.

New Organizational Structure(s)

> *"You never change things by fighting the existing reality. To change something, build a new model that makes the existing model obsolete."*
>
> *- Richard Buckminster Fuller*

As noted earlier, wide sweeping organizational changes are not the focus of this document. That said, it is clear that our current structure that has inherited the combined issues from its predecessor organizations. A scalable way to address this in a non-disruptive way would be to create a 'test team' working on a new project free from the existing organization as noted earlier in this report.

This team would be free to innovate not only with technology but also with effective ways of organizing and cost effectively delivering a new service. Work in this area could begin as soon as management and the potential team approve.

Success Criteria

Deliverable:

> Creation of a single team to build a new service (or a fundamental redesign of an existing service) and then successful delivery of that service.

Benefits:

1. Provides an example that may be emulated for efficient organization of services within HUIT.
2. Can be used to help reduce morale concerns as a concrete example of attempting to resolve issues.

Timing

Time, function and resources are the three variables in the delivery of any project or set of projects of this kind. Once at least two of the variables are known, the other will become obvious.

Appendix A - HUIT Systems Study Reports

- Analysis and Recommendations - Network, Systems, Software Deployment, Monitoring, Management and End-to-End Perspectives of the PIN Systems, Version 1.2, Jon Saperia, 7/11/2012.

- iSites Services - Risks and Recommendations, Jon Saperia and David McElroy, Version 1.3, November 15, 2012.

- HUIT Email And Active Directory Status Review, Version 1.2, January 23, 2013.