

IT@JH
Information Technology Project Management
Procedures and Guidance
Version 0.3

Overview	3
What is a Project?	3
Predictive vs. Adaptive	3
Table 1: Comparison of Methodologies	4
Common Mistakes in Project Management.....	6
IT@JH Project Planning: A Guide	6
Stage 1: Initiation.....	6
Initiation - Business Need	7
Initiation - Concept.....	7
Initiation – Goals and Strategic Implications.....	7
Initiation – System Sponsors, Stakeholders and Governance	8
Initiation – Approval Process	8
Initiation – Funding	8
Initiation – Inter-dependencies and interfaces	8
Initiation – Assumptions and Constraints	8
Initiation – Risk.....	9
Stage 2: Planning.....	9
Planning - Project Manager and Customer Liaison	10
Planning - Scope	10
Planning – Goals and Objectives	11
Planning – System Requirements.....	11
Planning – Budget(s)	12
Planning – Contract(s)	12
Planning – Planned iterations (sub-stages)	12
Planning – Resource allocations	12
Planning – Schedule(s).....	12
Planning – Change control plan	13
Planning – Interdependencies and interfaces	13
Planning – Risk management plan	13
Planning – Training plan	13
Planning – Communications plan	13
Stage 3: Execution	14
Stage 4: Testing.....	14
Testing - Test Environment.....	14
Testing - Test Plan.....	15
Testing – Systems	15
Testing - Acceptance.....	15
Testing - Integration	16
Testing – Load, Stress and Failure Mode	16
Testing - Security	16
Testing - Regression	16
Testing - Assessment of Documentation and Training	17
Stage 5: Completion.....	17
Attachment 1: Systems Design and Deployment/Common Technical Concerns	18
Confidentiality	18
Authentication	18
Authorization	18

Access logging	18
Transmission Data Security.....	18
Device Security	18
Server Security	19
Portable Device Security.....	19
E-mail.....	19
Data Change Logging.....	19
Data Error Handling/Monitoring	19
Data Retention.....	19
Data States (Preliminary, Final, Amended, Addendum)	19
Dictionary Synchronization	20
Availability.....	20
System Fault Tolerance	20
OS Patches.....	20
Antivirus Protection	20
Administrator account management.....	20
Remote System Access	20
Change Control Approval	21
Code Management.....	21
Change Deployment Mechanism	21
Business Continuity Plan.....	21
Disaster Recovery Plan.....	21
Downtime Procedure.....	22
Remote Site Recovery and Operation	22
Data Backup (and backup data security).....	22
System Failover (in clustered environments)	22
Application Environment Backup.....	22
Network Protocol Standards.....	23
Network Topology	23
Bandwidth Consumption.....	23
System Management/Monitoring Software	23

Overview

This document establishes a common framework for managing information technology projects at Johns Hopkins. It includes definitions, processes, checklists and things to watch for when managing or participating in an IT-related project.

What is a Project?

The standard definition for “project” is a temporary endeavor undertaken to create a unique product, service or result. An example would be the Manhattan Project. A wide range of scientific, technical and operational professionals and staff were sent to New Mexico to develop an atomic weapon based on theoretical advances that had already been made. After the bomb was successfully completed, the Manhattan Project ended, although many of the individuals involved would continue to conduct similar research at Los Alamos.

This standards and guidance applies to major initiatives involving, for instance, at least several hundred work hours and staff efforts across IT and customer departments. As a rule of thumb, every project has a beginning and foreseeable, if not precisely defined, end. Thus, on-going system support would not generally be considered a project, unless something new is developed. As with the Los Alamos example, there may be on-going activity afterwards, but the defined project itself has concluded.

Another attribute of a project is that it has scope. Not everything related to context or direction of a project will be within the scope of that project. Limiting project scope and managing necessary changes in scope are among the most important factors for project success. How project methodologies handle changes in scope will be considered in greater detail below.

A project includes the following stages:

- Initiation
- Planning
- Execution
- Testing and monitoring
- Completion

Most projects include several iterations of these stages with several stages running in parallel. Even so, a project manager should be able to define stages for each iteration of a project. Templates and checklists document are designed to assist project managers at each stage.

Predictive vs. Adaptive

Project management methodologies can generally be placed upon a continuum from “predictive” to “adaptive.” Projects that have from the outset detailed requirements and resource allocations and maintain rigorous change controls throughout are

generally predictive. In contrast, adaptive often called “agile” projects are more self-managed, intensely iterative and rely heavily on face-to-face communications.

Any successful project will have both adaptive and predictive elements. And as it passes through its stages, a project may shift from predictive to adaptive or vice versa. Project managers should ensure that planning methodologies address project goals and development conditions.

For instance, adaptive methods generally require routine face-to-face (or other real time) communications between members of the project team and with the customer. If such communications are not frequent enough, there may need to set more rigorous requirements and reporting methods, thus becoming more predictive.

Choosing a methodology is not principally a matter of personal preference or taste. Rather, it is an attempt to locate the right tool for the job after a realistic assessment of the situation. Either approach can work, yet current research indicates that each is better suited for specific conditions:

Table 1: Comparison of Methodologies

Predictive	Adaptive
One or two modules (“big bang”)	Several discrete modules
Many dependencies between modules	Decomposable (reversible) modules
Structured project review meetings	Daily developer communications
Release once (or twice) but right	Release early, release often
Customer review panel	Frequent customer contacts
Rigorous change control	Short development and deployment windows
Structured developer roles	Flexible developer activities
Hierarchical management	Self-managed development teams
Deployment-oriented	Development-oriented

The predictive approach is well entrenched, but it suffers from two significant drawbacks – inaccurate forecasting and a too common disconnect between planning and execution. Rarely does a project face issues in quite the way the original planners envisioned. And even the most thorough project plan may be left on the shelf by the implementation team. Adaptive methods seek to sidestep these problems by substantially shortening deliverable time horizons (e.g. 4-6 week development modules) and replacing detailed project plans with frequent communications (usually in the form of daily updates) between developers and nearly as frequent communication with users.

Like many academic institutions, Hopkins is not a rigidly hierarchical organization (although certain Hopkins entities may be), and its business drivers and objectives are rarely defined clearly. One might therefore expect that an adaptive approach to

projects would be most successful here. Yet those looking to use adaptive methods should consider a few factors instrumental for success in the adaptive mode.

The first is related to how the system is to be deployed. Agile approaches are best for systems that have several modules set for a sequential roll-out. Each module becomes a mini-project of its own, with limited objectives and a short time horizon. Contrast this with the “big bang” approach where inter-dependent modules are bundled together and released all at once. Effectively coordinating interdependencies and development sequences generally requires a more predictive model.

A related consideration concerns customer tolerance for “good enough” early iterations of a system. Adaptive technologies generally follow the 80/20 rule whereby 80 percent of the functionality can be generated with 20 percent of the effort. What usually follows from that is an early deployment to a production environment. This is perhaps the most cost effective method for clarifying how the remaining 80 percent of effort will be applied to the last 20 percent of the system’s functionality. Yet for high assurance or safety requirements, 80 percent may not be good enough for production.

While system development and deployment cycles can be adjusted with good planning and careful testing, the third major success factor for adaptive systems may be the most difficult to achieve. Agile development compensates for less detailed planning by requiring real-time and frequent communication between the principals. This usually means daily face-to-face meetings, conference calls or the like (although creative uses of a project management tools like Confluence or SharePoint can be an effective substitute for face time). Moreover, there must be regular communication between developers, testers, project managers and with customers and users. Many of our projects are short-staffed on the customer side, and this can severely undermine the ability to adapt to changing circumstances and requirements.

If ready and continuous customer participation is required for agile success, few agile projects would succeed here. Even so, it is questionable whether even the most structured and predictive model can overcome an inert customer. The most important thing is to build in regular customer communication into both predictive and adaptive plans.

The Role of Product Selection

There is some question as to whether product selection – specifically third party production application -- is part of an overall project or itself a project. In some instances, product selection is exploratory and open-ended. Vendors are provided with some basic guidance, and vendor proposals form the basis for product requirements and specifications. For these cases, it is probably best to treat product selection as a separate project from design and implementation.

Common Mistakes in Project Management

The following factors, if left unchecked, are detrimental to even the best planned projects.

- Unclear project goals and objectives
- Unrealistic deadlines
- Staffing allocations inadequate in numbers or unsuitable in skills or expertise
- Inadequate or unclear project management authority
- Lack of solid business sponsorship
- Ad hoc testing
- Failure to emphasize training and usability

IT@JH Project Planning: A Guide

Whether a project team chooses to use predictive, adaptive or some combination of both, this approach to project management is equally applicable. This is a stage-by-stage approach that can be used for nearly every IT project, from those of entirely new software developed in-house, vendor developed with or without customization, or an upgrade of a current system. The approach loosely follows several project management methods and seeks to avoid obscurity and jargon. Each stage will eventually include several templates and checklists. These are also being produced for the SharePoint, Confluence and JIRA issue systems.

By maintaining some consistency across multiple projects, we can measure success specific to a project and across multiple projects. And while we expect to see some diversity in templates, it is important that projects follow the basic structure of this guide.

Throughout a project lifecycle, it is critical to continuously re-assess documentation, including plans, budgets and schedules. The evaluation process should include stakeholders and project team. No good project plan stays on the shelf during deployment and completion.

Stage 1: Initiation

The **initiation** stage determines the nature and scope of the project. It defines the reasons for developing or deploying a system (often in the form of statement of a solution to a problem). It identifies the customer, contracting vehicles, interfaces, governance, financial costs, opportunity costs, constraints, project management approach and risk. The artifact of this process is the **Project Business Case** (PBC).

As will be emphasized later, all projects must have a knowledgeable IT project manager and a customer liaison. In many cases these roles are defined in the next stage planning, so what is needed at this stage are IT project and customer "initiators." In some cases, these initiators may become managers, but often not.

They are responsible for putting the project on a sure footing early by doing the following:

- Identify business need
- Develop a conceptual design of the proposed system
- State goals and strategic implications of the system (e.g. impact on safety, security, simplification, capital budget criteria, etc.)
- Define system and project sponsorship, stakeholders and governance
- Define system and project approval process
- Identify funding sources
- Enumerate interdependencies and interfaces
- Identify constraints
- Consider assumptions and risk
- Address return on investment

No matter the project methodology, all projects should consider and document these factors at the earliest stages. These often change some over the life of the project. For example, sponsorship may shift from one business unit to another. Still, radical shifts in several of these factors are often indicators of trouble in a project.

The PBC in an appropriate format (usually defined by customer entity) must be signed and approved by identified system and project sponsors prior to approval of a project plan.

Initiation - Business Need

A business need can almost always be translated into a problem statement. It is important to clearly define the problem, why it is a problem, how the problem arose and how it is likely to evolve in the absence of an intervention.

Initiation - Concept

How would a proposed system address or solve the business need? Workflow diagrams are especially helpful for communication purposes.

Initiation – Goals and Strategic Implications

IT@JH maintains a current Strategic Plan, and new systems are assessed in terms of the fundamental strategic mission of the organization and the institution. It is not enough merely to build a system that provides a benefit. That project must fit within the overall direction of institutional information technology. Strategic planning is generally not choosing between good and bad systems, it is choosing the better of two or more good ones. It is therefore incumbent on sponsors to understand strategic direction, justify the project in terms of organization mission (the seven S's) and establish governance and funding models appropriate to the type of project. There are also opportunity costs to allocating knowledge resources to a certain project and not another, and these should be addressed.

Initiation – System Sponsors, Stakeholders and Governance

It is best to clearly articulate these at the outset. Information systems must be sponsored by an organizational unit of Johns Hopkins (e.g. and academic department or division, Johns Hopkins Hospital department, entity like Johns Hopkins Healthcare or the Clinical Practice Association). There are also individuals or groups that are not sponsors yet may be affected by the proposed system. Governance bodies are usually pre-existing management structures or committees, but may in some cases be ad hoc for purpose of the proposed system. In many cases, governance includes a formal approval process.

Initiation – Approval Process

Approval by bodies such as the Clinical Systems Advisory Council (CSAC), or the Financial and Administrative Systems Advisory Council (FASAC) might also be required. In cases where clinical data is being displayed or exchanged between systems, review by the Clinical Data and Documentation Committee (CDDC) is required. Clear understanding of approval status will be necessary before the expenditure of institutional resources.

Initiation – Funding

Funding sources should be identified for initial and ongoing system costs. In many cases, capital budget approval is required. A substantial portion of the budget should be set and provision made for cost over-runs in capital and operating components.

Initiation – Inter-dependencies and interfaces

If the successful deployment of a system requires a substantial upgrade or change in infrastructure, that concern should be addressed in this section and in the risk section below. Interfaces are generally data or application layer interdependencies with current or planned JH systems. Major interfaces such as those with ADT, EPR or HopkinsOne are important and generally manageable, especially when the system is a receiver feed. There is more risk in bi-directional interfaces and those interfaces with less well supported systems.

Initiation – Assumptions and Constraints

Assumptions are like interdependencies in that system success hinges on certain factors outside a project team's control. Assumptions, however, can cover a broader array of factors than just systems or technologies in the Hopkins environment (e.g. regulatory or accreditation requirements, etc.) Constraints are factors that may limit deployment of the system or impede system or project success.

Initiation – Risk

Every project involves risk. There is the risk that the technology will not work as planned or needed, that the system will be too expensive or complex, or that users will opt for another approach to solving the business case. There are also project-related risks, such as loss of key personnel, defeated assumptions, etc. In the PBC, it is more important that risk factors be identified than mitigated. Control measures will be discussed in project plans and throughout the lifetime of project. And while many risk factors can be identified early, many more only become apparent as the project proceeds.

Stage 2: Planning

The initiation phase considers the system as a whole. It defines the context in which a project may be initiated. The first stage of the project itself is the **planning** stage. Of course, planning continues throughout the life of a project, yet it still critical that an overarching plan be established at the outset.

The project plan incorporates and annotates the Project Business Case and structures management of a project to meet some or all of the goals enumerated in the PBC. This means that one PBC may generate a single project with several planning/execution iterations, or it may spawn several independent projects each with its own project management structure. The choice to split a system into several discrete projects will be discussed in more detail below.

A **Project Plan** (PP) starts at a broad systems level:

- Updated Project Business Case (PBC)
- Designation of Project Manager (PM) and Customer Liaison (CL)
- Statement of Scope
- Goals and Objectives

At this point it is time to begin the difficult and time-consuming work of planning out the system. Since it is possible that a System may involve several projects, systems requirements and designs may cross over between projects. Where the project is not principally concerned with developing or deploying a system, you can substitute “project” for “system” requirements and design. While such project oriented documentation is unlikely to be as extensive as those for a fully realized system, it is always a good idea to begin with a healthy understanding of stakeholder requirements and an account of project design.

- System Requirements
- System Design

Now we work back to the more project specific considerations. It is tempting to divide budget and contract matters from design, some cross-over is necessary. For good or ill, vendor and stakeholder issues often communicated through budgets, schedules and contracts. It is critical that these match up with requirements and

design expectations and documentation. When disputes arise it is often because of an inconsistency between a requirement and related budget or contractual language.

- Budget(s)
- Contract(s)
- Planned iterations (sub-stages)
- Resource allocations
- Schedule(s)

The following components are routinely neglected in many project plans. They may seem like second order considerations, but time and again the details here are the differences between successful and unsuccessful projects.

- Change control plan
- Interdependencies and interfaces
- Risk management plan
- Training plan
- Communications plan

What many people consider a project “charter” is largely found in the PP (although there are some elements in the PBC). Again whether one is using a predictive or adaptive methodology, the PP should be complete before any work is commenced. The PBC operates at the highest level of abstraction and at a systems level, the PP is also an over-arching document, yet it is not usually so general and pertains specifically to project(s) related to the system. It is not, however, the end of planning as specific stages and iterations regarding execution, testing and completion are considered in sections below.

One of the main differences between a predictive and adaptive PP is in how each treats project scope. An adaptive system would allow for a more open-ended account in the PP, and more rigidly define scope for each sub-stage iteration. Predictive systems are more likely to tightly define scope in the PP and require formal change management so as to avoid scope “creep.”

Planning - Project Manager and Customer Liaison

Each project requires management accountability on both the technical and customer sides. In most cases, the PM will be an IT professional. In some instances, project management accountability will vest with business (not IT) management. If so, a technical project lead should still be designated.

Planning - Scope

This is one of the most important factors for success. In predictive models, scope is defined rigorously as the precise business processes and systems that will be developed or modified and related execution and testing. For adaptive systems, there will likely be a general statement of scope and criteria for incorporating new

knowledge and objectives into the project. For example, part of the scope could be to evaluate whether a messaging component should be added to the system.

Planning – Goals and Objectives

Having defined scope, goals and objectives should flow naturally. There should be only a few goals, while there could be several more objectives. Objectives should be measurable and time limited.

Planning – System Requirements

Requirements can vary based on project context and styles of project leadership. Usually the system, not the project, is the focus of a requirements document. Gathering requirements can be Project managers should ensure that requirements are stated clearly and communicated to all members of the project team. There are a number of methods for developing requirements and these may be specific to customer or organizational context. There are several things to look for when writing requirements.

- Requirements analysis is essential to success of each related project
- Customers find it difficult to write requirements that properly reflect their needs
- Requirements are often negotiable in terms of cost, schedule and risk
- Requirements analysis can involve a number of techniques, and user involvement is valuable
- Requirements will evolve in detail (and likely change) over system development life cycle
- It is important to document and track agreements, interpretations and decisions.

Good requirements have the following characteristics:

- Verifiable
- Unambiguous
- Complete
- Consistent
- Traceable
- Concise
- Use standard constructs.

One of the defining characteristics of adaptive methodologies is that requirements definition is on-going and can change based on the findings of each iterative stage. It is therefore incumbent on project managers to maintain criteria for requirements change, authority and articulation.

Planning – System Design

Design is such a complex topic that it deserves a paper on its own. *Systems Design and Deployment/Common Technical Concerns, Attachment 1* provides a good checklist for basic design issues. It is strongly urged that systems that require sophisticated design approaches adopt a coherent design methodology. Johns Hopkins is in the process of compiling design documents and methodologies for model purposes.

For projects that require substantial development or customization, you should generally spend at least as much time designing the system as coding.

Planning – Budget(s)

The budget may be specific to the project or cover the entire system. It may be divided between capital and operating resources, cut across multiple years, and/or organizations.

Planning – Contract(s)

The contracts should be in place prior to project initiation and statements of work incorporated into scope. Review with counsel attachments and checklists required in contracts at the earliest stages. For example, JHHS require a HIPAA Security capability checklist. It is best to review this with vendors early in the procurement process rather than wait until contract negotiations are well underway.

Planning – Planned iterations (sub-stages)

Project plans sets forth tasks, and in this model tasks are categorized and developed according to iterative project stages (see Stage 3: Execution below). It should be possible to identify at the outset all of the early parallel and sequential sub-stage iterations. Later sub-stages may be less certain, but there should be some forecast of potential iterations and spin-off projects that remain within project scope.

Planning – Resource allocations

Tasks, iterations, resources and scheduling are all inter-linked. Changes in any of these will have an impact on the rest. The principal concern here is identifying staff or facilities with unique capabilities (e.g. interface specialists, enterprise DBA's, network architects) to ensure that these resources will be available when needed.

Planning – Schedule(s)

These can vary widely in form, content and levels of assurance. In general scheduling begins with customer deadlines or product release schedules, and then involves filling in the gaps. This can be problematic for complex projects with a number of iterations.

Planning – Change control plan

Both predictive and adaptive methodologies must implement change control processes, the only difference may be the level of rigor. Change control in predictive environments amounts, in many cases, to a change in requirements, with reverberations in budgets, schedules, etc. It is therefore critical that changes be clearly stated and agreed to by stakeholders. Change in adaptive models is often more fluid, and in these cases documentation through journals and changes in iterative plans is usually sufficient.

Planning – Interdependencies and interfaces

The diversity and complexity of the Hopkins environments produces a tangle interfaces and manifold dependencies across the institution. Every production system must address the problem of interfaces. Vendors are often ill-equipped to handle the scope and complexity of our interface requirements. Familiarity with HL7 and XML is common, but many systems that use standard integration tools vary enough to make interfaces complex and difficult to develop. User interfaces and patient identity are also persistent interface issues here, and project managers should review JHM standards on patient context and data presentation.

Planning – Risk management plan

Projects never go precisely as planned. The risk management plan is the opportunity for the pessimists in our midst to point to where things can go wrong in the project and in the system as a whole. The simplest way to write this is to identify each major advance that the system represents over current practice, foresee what can go wrong and outline a Plan B.

Planning – Training plan

Project managers often address training too late and training is therefore rushed and uncoordinated. Training involves a “who, what, how,” analysis, all of which are important. It is important to understand that training often has its own hardware, software, connectivity and user interfaces. In other words, it is often a system of its own and should be planned for as a system, including requirements and design.

Training assessments are an important component of testing, described in detail in the training section below.

Planning – Communications plan

The communications plan consists of a table that identifies stakeholders and project milestones. The plan then designates communication methods for each type. It may be prudent to communicate certain technical release information to developers via e-mail, and leave it on a project Website for other stakeholders who do not require immediate notification.

Stage 3: Execution

The execution stage includes implementation of the plan. It is tempting to think of this as the period where things are actually done. The picture is more complex. The execution phase of a project consists of a series of iterative sub-stages. These sub-stages are defined by a limited number of achievable tasks that can be completed sequentially or in parallel. Each sub-stage includes a shortened project plan or through a journal or wiki that includes the following elements:

- Stage scope – objectives and deliverables from the stage. It also includes tolerance levels for risk of sub-standard, late or expensive deliveries
- Stage context – issues that led to this stage, related current stages and dependencies are also documented
- Task allocation and responsibilities – the PM may assign management responsibilities to team members. Team members are usually encouraged to identify tasks and introduce them into the stage plan
- Stage schedule – timelines for completion
- Stage boundaries – concerns the completion and near completion of stage as well as conditions that exceed tolerance levels defined in the scope.

The primary difference between predictive and adaptive methodologies is often evident at this stage. Adaptive approaches will usually have voluminous stage plans that change rapidly and are largely self-managed. Predictive systems stage plans will often resemble task orders. These are carried out according to the pre-determined project plans.

Stage 4: Testing

Testing programs are based on scenarios. Testing generally consists of the following components:

- Systems
- Acceptance
- Integration
- Load, Stress and Failure Mode
- Security
- Regression
- Documentation and Training Assessments

Testing plans should be developed early and often, and testing should be conducted throughout the system lifecycle. Avoid waiting until the last minute to develop a testing plan.

Testing - Test Environment

Errors can be introduced into systems at any level including, hardware, operating system, network, software infrastructures, software applications, or configurations. Changes at any level should be subjected to testing prior to production use. The

degree of testing should be determined according to cost, risk and potential impact. It is strongly recommended that industry standard testing tools be used for most, if not all, of the testing stages below.

Testing is ideally conducted in a completely separate environment. This allows the test environment to be used for changes at all levels. At a minimum, testing should be conducted with complete separation of the test data and application software, from the production data and application software.

Testing - Test Plan

Testing plans are multi-dimensional. In the sections below we consider types of testing (e.g. systems, regression, security). Within each of these testing types, a general method is followed. The items below compose a reasonable test cycle for each testing area:

- *Identification* -- identify business processes and transactions
- *Test Objectives* – what knowledge will be gained by undertaking a testing protocol
- *Test Scenarios* -- description of the situation/processes being tested
- *Test Data* -- fields, templates, files, documentation, etc.
- *Expected or required results*
- *Entrance criteria* – establishes programming or system maturity necessary to begin testing cycle
- *Setup and execution procedures* -- detail of how the test cycle is conducted
- *Exit criteria* – verifies against scenarios and expected results and indicates that testing cycle can be terminated
- *Responsibility and Accountability*

Testing – Systems

It is first necessary to test the system according to criteria of performance. In other words, does the system perform as expected in a relatively stable, hot house environment? While this is often the first cycle of tests run, it is critical that systems testing be done throughout the development lifecycle.

System testing is similar to what is known as function testing and generally involves:

- Application functions
- Databases
- API's and internal interfaces
- Logging and reporting

Testing - Acceptance

While system testing is intended to ensure correct technical operation, acceptance testing is intended to ensure that the system does what the user needs it to do in

an acceptable (practical and efficient) fashion. In other words, it tests whether the system works as expected.

Acceptance testing should always be performed by user representatives, and should be based upon scenarios which represent realistic task combinations.

Testing - Integration

Integration introduces the intricacies of the Hopkins environment into the testing regime. It begins with introduction to near-production hosts, placement in data centers, configuration behind firewalls and network devices, introduction of access control (e.g. AD or Siteminder), Hopkins and external interfaces and data feeds. Many of the same systems tests performed in step one are run again in a simulation of Hopkins conditions, albeit usually at lesser loads.

Testing – Load, Stress and Failure Mode

There is a certain logic to the order of testing. First one tests that application alone in hothouse conditions – does it work according to specification? Then we ask -- does the system meet customer requirements? Having satisfied those, it is time to introduce the system to a simulation of the Hopkins environment. Now the system is ready to be broken, and the purpose of stress testing (like security testing below) is to test the limits of systems tolerance, whether it fails well or badly and how it recovers. Cycles of testing for applications in operating in degraded mode are particularly useful.

Loadrunner and similar tools can provide automated stress testing on systems components.

Testing - Security

Security testing is a combination of systems and stress testing. It focuses on the three main components of security – confidentiality, integrity and availability. The primary concerns include insecure coding practices that result in buffer overflows, SQL injections or the like, poor key management and escalation of privileges. Testing administrative access can indicate vulnerability to insider or multi-vector attacks. It is generally better to test the application layer and database layers in separate scenarios and then integrate tests.

OS and Web server configurations should be tested with NESSUS or a similar vulnerability tool, and browser interfaces scanned for common SQL injection defects.

Testing - Regression

Regression testing refers to the confirmation that changes in one part of a system have not caused errors to appear in other parts. Typically the parts known to have

changed receive substantial system testing. The scope of regression testing needs to be consciously determined based upon an assessment of cost, risk and impact. The risk component requires an understanding of the system's construction sufficient to assess the potential for interactions. At a minimum, the "complete system" should be exercised, in its most common modes, after changes at any level.

Testing - Assessment of Documentation and Training

Documentation and training are critical to the success of any system. It is not enough to simply have documentation and a training plan. Both of these must be tested with users, administrators and stakeholders well in advance of deployment. Focus groups accompanied with surveys should be used to assess these both in form and substance.

It is strongly encouraged that nearly all documentation and training materials be made available to Hopkins users on the Web and that there be made space for comment by users throughout training and implementation phases of the project.

These components should also be incorporated into the change control process.

Stage 5: Completion

For a project to be a project it must end. At IT@JH we provide a short yet formal process for project closure. Project Managers must certify project closure by checking against PP objectives, assessing project timeliness and success and communicating next steps. Project completion usually requires a formal sign-off by Project Manager, Customer Liaison and Project Sponsor(s).

Attachment 1: Systems Design and Deployment/Common Technical Concerns

Confidentiality

Authentication – who is the person trying to use the system

- Should systems be required to use a centrally supported authentication system (like RACF, or JHED, Active Directory, or perhaps Site Minder).
- If using an independent authentication system, what about:
 - ensuring that users lose access to the system when they leave the institution.
 - A process for granting and managing the authentication
 - Password length and complexity standards, expiration policy.
 - Use of common user identifiers (so that users would be identifiable by the same “name” in all access control and logging datasets).

Authorization – who can see what data. Under HIPAA (and general good security practices) a rule of “minimum necessary” should be employed.

- How is policy set for data access?
- Does the system use role based authorization? With a common (with other systems) definition of role?
- Do the settings for each user correspond to settings for the same data and user in other systems?
- Is a policy needed on query access to the system – under HIPAA it is easy for data to be “misused”.

Access logging – who has seen what data

- Is access logging a HIPAA requirement, or something implied by our HIPAA Privacy statement?
- Should access logging be required?
- Should logs be centrally stored? consolidated? Included in “on-line” access reporting of EPR?
- At what level of granularity should access be logged?

Transmission Data Security

The level of protection of data during transmission, like most security issues, will be dependent upon risk assessment. Encryption of data during transmission is one tactic using technologies like HTTPS, IPSEC, VPN and is typically required for data traveling on the internet.

Device Security

- Workstation/Application Inactivity Timeout/Password protection
- Storage of protected health information in local files

Server Security

Servers which hold large aggregations of data require additional protection, typically including physical location in secured areas and careful use of administrative accounts and passwords. Currently, the protection of stored data is typically provided by access controls on the application and server. Encryption of stored data is an option.

Portable Device Security

Access to patient data and other sensitive data on PDAs and other portable devices will be restricted to authorized individuals through appropriate policies, procedures and access control mechanisms. All portable devices will incorporate hardware and/or software features and functions necessary to ensure that patient and other sensitive data cannot be accessed by unauthorized individuals in the event of loss, theft or misplacement of the device.

E-mail

Inherently insecure due to forwarding, routing and "clear text" transmission protocols.

Integrity

Data Change Logging

All writing/editing/changing of clinical data must be recorded either as an intrinsic part of the applications database or in external log files which are retained.

Data Error Handling/Monitoring

Data errors must be reported and handled in some way. Typically this involves system logs which are regularly reviewed, or the automated "push" of error condition information to a person (for example by page). A common type of error is the failure of an "interface transaction" due to an unanticipated or impermissible data content. Such failures should not result in the un-noticed loss of data.

Data Retention

An explicit data retention policy should be stated in the project documentation. Where projects are subject to imposed data retention policies (either internal or regulatory) these should be clearly referenced. Few information systems have unlimited lifetimes. The ability to separate the data from the application for long term storage must be considered.

Data States (Preliminary, Final, Amended, Addendum)

Many applications, and clinical processes, have well defined states related to the life cycle of a document/record. The treatment of clinical record states must conform to institutional policy, and medical records management best practice. Controls should be incorporated into user applications which enforce correct record management.

Dictionary Synchronization

Many systems throughout the institution shared reference data, such as a lists of credentialed providers, or locations, or laboratory test codes. Systems should plan for the ongoing synchronization of such data. In cases where data is received from another system containing unknown values, mechanisms should be in place to allow dictionary reconciliation without loss of the data in the affected record.

Availability

System Fault Tolerance

System fault tolerance is available at many levels of modern systems. These include: disk; disk array control; power supply; network card, application server, web server, cluster node, etc. Many of these are now automated (switching occurs automatically) and highly reliable. The selection of fault tolerant technologies for a project must be determined by a cost, risk, impact assessment. Any fault tolerance technology which is employed, must be tested. Some of these technologies are highly dependent upon configuration. History has demonstrated that the use of untested fault tolerance technologies causes more faults than it prevents.

OS Patches

Host/server systems are subject to ever increasing levels of attack. Operating system patches to close known vulnerabilities are now a common occurrence. Plans must be in place to maintain the OS patch level (and version/build level) of operating systems as part of a general defence strategy. (see ICSC web site for additional guidance)

Antivirus Protection

Host/server and client systems are subject to ever increasing levels of attack. Plans must be in place to maintain anti-virus software as part of a general defense strategy. (see ICSC web site for additional guidance)

Administrator account management

Remote System Access – for users and maintainers by various means: VPN, Web application, remote control software, terminal services. Remote access essentially means access from less protected environments, across less protected communication channels. As such there is a higher level of risk, requiring a higher

level of controls. At a minimum, encryption of data streams is required. For high capability administrator accounts, additional controls may be required (stronger authentication). For vendor support access, controls must be in place so that we retain control over systems changes, and data access.

Change Control Approval

All changes (hardware, software, infrastructure services) to production systems must pass through a change control process. The purpose of the process is to ensure the safety of the change. Typically, change control processes require change review and approval cycles by different personnel than those authoring the change. Change control is generally tied to the successful completion of appropriate testing, and includes: a description of the change, a risk assessment, and a 'back-out' process.

Code Management

Application code is a core resource for any information system project. This includes the code written by the application developers, and any associated configuration files. These sources of code must be maintained in a state capable of being "backed-up" and "restored". Code management tools are frequently used to retain versions of software source code (down to the individual component level) and to control editing and assembly of components during the development process.

Change Deployment Mechanism

The introduction of new software for an application potentially requires a coordinated distribution of the software, across multiple machines. Server based systems are generally easier to coordinate (due to the small numbers, and central control). "Fat clients" (with large distributions needed to each client workstation) are the most difficult. The machinery for deployment of changes must be understood for each application, and should be tested before roll-out of the application. Typical tools involve "push" technologies (like Microsoft SMS), or automated installers.

Business Continuity Plan

Users of all systems must be prepared for the possibility that the system may unexpectedly become inoperative. Procedures should be predefined for continuing with the core mission, in the absence of the information system. Procedures may vary based upon the duration of the outage. (see "IT Disaster Recovery and Business Continuity Policy").

Disaster Recovery Plan

In situations where an information service has become interrupted, a Disaster Recovery Plan defines the steps to be taken to restore the service. The plan requires an assessment of cost, risk, and impact. Different recovery plans may apply to different disaster scenarios. Plans may range from “fix in place, or live without” to “restore at a prepared off-site location”. (see “IT Disaster Recovery and Business Continuity Policy”).

Downtime Procedure

Downtime procedures differ from Business Continuity Plans because the outages are planned rather than unexpected. Typically, the downtime procedure operate on the expectation of a limited duration outage, and with the opportunity to mitigate impact though increased staffing or other support mechanisms. User notification, and the timing of the outage are usually key issues.

Remote Site Recovery and Operation

One Disaster Recovery strategy which uses a prepared location with a complete set of alternate hardware and connectivity as the recovery platform. When this strategy is employed, it must include the routine testing of the recovery sequence at the recovery site.

Data Backup (and backup data security)

Data backup is the most basic of all disaster recovery techniques. It requires copying all data needed to operate a system on a regular basis. The copy technique and its frequency are determined by the degree of acceptable data (transaction) loss. Modern database systems with “backups” and “journals” can recover up to the last transaction in many instances. Backup and recovery (especially recovery) must be periodically tested. Backup media which contains sensitive information (including any patient data) must be protected just as any other data source. A mix of physical and electronic controls may be used. Backup media should be cycled through locations, separate from the hardware, for additional safety.

System Failover (in clustered environments)

Although a form of fault tolerance, which is addressed elsewhere, clustered systems typically present a greater technical challenge. Failover within a cluster of basic file and other operating system services is usually not a problem. However, successful failover of complex applications and database engines requires local testing and high level of vendor support and experience.

Application Environment Backup

System backups need to include all elements of the environment in order to allow system recovery. In the past his has primarily meant data backup and application

source code backup. As environments become more complex so does the requirement. Examples: a development environment with specific versions of software libraries might be needed in order to recompile an application; firewall setting may have been established specifically for an application; applications may depend on receiving "certificates" from other servers which match "stored keys".

Network Protocol Standards

(see ICSC Site for network standards)

Network Topology

Modern networks utilize zone defenses and layered protection. Network architects should be consulted about the placement of any application on the network. Applications requiring any special connectivity (other than low volume, wired, TCP/IP connectivity) should emphasize these requirements when developing their network use plan.

Bandwidth Consumption

Vendor (and developers) should be asked to estimate their bandwidth footprint on the network during the planning process. If possible it should be measured. This information should be supplied to the network architects.

System Management/Monitoring Software

Depending upon the nature of the system there may be opportunities to link the system into existing system management consoles. The system may also have the ability to generate alerts to support personnel, and in some cases to be self repairing. Plan to monitor the system. The goal is to know about problems before the users call.