Why non-functional requirements can mean the difference between the success and failure of your IT project

Karen Carpenter, Sr. IT Director, PMP

Thomson Reuters, 1455 Research Blvd, Rockville, MD 20850; karen.carpenter@thomsonreuters.com, jk.carpenter01@comcast.net

ABSTRACT

Most IT project teams focus on return on investment, the user experience, performance, and features that will delight their customers. Equally important and often overlooked are the non-functional requirements: Disaster Recovery, Information Security, and Software Selection. In this session, we review why these non-functional requirements are essential, and provide practical advice and tips to apply throughout the project life cycle.

Are you confident that your systems would stand up if a disaster struck? Companies evoke disaster recovery (DR) procedures for a variety of reasons: severe weather, maintenance, software/hardware bugs, or environmental issues. Ensure your project design includes systems and infrastructure to support these activities and has the processes and personnel in place when needed.

Could your company withstand a sustained cyber-attack? In today’s internet driven economy, Information security is often a topic of the evening news. It is paramount that your project team understands and designs a solution that protects your customers and your company. Include infrastructure design, security policy, and sensitive information reviews throughout the project life cycle.

Do you have access to the right software solutions and expertise? Your company continually strives to cut costs by increasing demands on data analytics and computers to solve complex problems. Software selection must consider skills, licensing models, scale, security, DR, and supportability.

At the end of the session, you will understand the fundamental importance of these non-functional requirements and have real world examples and checklists for use on your next IT project.
This paper is designed to provide practical background and helpful materials for project managers that are new to IT project management and want to insure that non-functional requirements are understood and considered during the project life cycle. Development teams focus on designing the best User Interface (UI), and providing a product rich in features, that meet the functional requirements outlined in the project scope document. Often overlooked, are the less exciting topics of Disaster Recovery, Information Security, and Software Selection. Although this paper focuses on these three non-functional requirements, others are equally or perhaps more important for your particular project and are listed at the end of the paper.

DISASTER RECOVERY

If Disaster Recovery scenarios are incomplete, manually intensive, time-consuming, poorly documented, or not thoroughly tested, your IT system may not function adequately during a variety of situations, including a severe weather event or data center disaster. In addition, executing routine deployments and system maintenance is much more complex and costly. Use the below questions during the design and build phases of your project to help insure you include time and tasks in support of delivering a sound disaster recovery capability.

**Design Phase.** Early in the design phase of your IT project, be sure that you ask the right questions of both the system and software architects. Examples include:

1. What is the Recovery Time Objective (RTO)? The time the user can be without the system will define the requirements for appropriate infrastructure and software restoration.

2. What is the Recovery Point Objective (RPO)? The recovery point objective determines how much data loss your users are willing to accept. How frequently is data updated? Can the data be re-processed? Is the data entered online directly by the users?

3. Will the Resiliency design support these Objectives? For example, if your users cannot endure more than 1 hour of downtime, consider keeping the data in your secondary site in-sync via automated methods.

4. Will the DR design utilize a live-live model? A live-standby model? Will the developed application software of your project keep the data in-sync? Will you leverage a third Party solution?

5. What teams will need to be involved in the failover activity? Do they have the appropriate level of resources?

6. What are, if any, the licensing requirements for your third Party software? Have these costs been factored into the project plan?
**Build Phase.** Once the design is agreed and the infrastructure is ready for the application installation, the technical teams should confirm that the failover design is still valid, and supports the original RTO and RPO objectives. Questions to consider at this phase of the project relate to the actual failover activity.

1. What methods will you implement to switch the systems to the DR site? Do all components failover at the same time?

2. What teams need to be involved in the failover activity? Have you secured resources knowledgeable in the type of failover you want to test? Who will authorize failover? How will support teams initiate failover?

3. What are the training and documentation requirements? Have you included these activities in your project plan?

4. What activities will result in the need to failover to the DR environment (application or data center maintenance, patching, deployment, or severe weather)? Who will authorize failover? Are appropriate support models in place?

5. In the event of a full datacenter failure, what is the importance of this application compared to other applications? Do you need to register the application with the data center operations team?

6. Has a failover test been scheduled or conducted and documented? What is the expected length of time for the failover activity?

Armed with the answers to these questions and an understanding of the DR requirements and design will help insure that your system is ready for both routine activities as well as unforeseen events.

**INFORMATION SECURITY**

Requirements related to information security affect numerous aspects of your IT project, including Software Applications, Data Centers, Networks, Firewalls, and Servers. This paper focuses on common security considerations when developing application software. Having a general understanding of the following concepts will insure that you include appropriate tasks in your project plan to secure your Company’s application and protect your customer.

**Architecture and Design.** During the design phase of the project, include the following topics in whiteboard sessions and understand the security policies of the organization. Discuss and understand sensitive information, application access
controls, software design best practices, Open Source software, patching, anti-virus and security scanning policies.

**Sensitive Information.** There are various types of sensitive information, each with a different security classification or potential impact to your application and ultimately your company. For simplicity, public information is available without restriction, and non-public information requires controls. Personally Identifiable Information (PII) describes data that is specific to an individual, with Social Security number the most often given example. During the application design phase of the project, understand the implications and corporate policies for sensitive data in order to insure that proper controls are in place. For example, non-public or PII data may require data encryption of both in flight and at rest data. For systems that hold sensitive company information, the company security policy may recommend additional access restrictions for externally facing web sites. In addition, if you are working on a project where data is housed outside of the United States, the country may have laws about storing data outside of their geographic location.

**Application Access.** Security policies may require password rules, may limit generic user accounts or elevated privileges. Insure server builds have user accounts and password policies that support the company’s security policy. Examples include removing guest accounts, locking down administrator/root accounts, and implementing strong passwords to avoid unwanted access to your systems. Two-factor authentication is a common method used for insuring a valid user is attempting to gain access to your application. Two-factor authentication is the requirement for two or more pieces of information to confirm your user’s identity.

**Software Design Best Practices.** To insure that application software is of the highest quality, include security-scanning activities at early stages in the project delivery, perhaps after completing a majority of components. These scanning activities will uncover security vulnerabilities, such as Cross Site Scripting¹, SQL injection², Two Factor Authentication, and elevated privileges.

**Cross Site Scripting.** Also known as XSS, is one of the most prevalent forms of security attack today and affects the end user instead of the actual application. An email with an embedded link that redirects a user and executes malicious software is an example of cross site scripting.

**Sql Injection.** SQL injection is a type of security attack that exposes the back-end database via a web application. The attacker is able to submit a command against the database not originated by the calling application.

**Open Source Software.** Open Source Software³ is publically available, and can be freely used and shared. Find a full definition of Open Source at http://www.opensource.org/docs/definition.php.
Most security teams recommend central registration and security scans of Open Source. Check the policies and confirm that your development team registers their Open Source usage and confirms this usage via Open Source Scanning. Common examples of Open Source software include Apache License 2.0, GNU General Public License, and Mozilla License 2.0.

**Antivirus and Patching.** To protect your application from unwanted access, malware or virus, many companies require up to date antivirus and anti-malware software. Antivirus software protects your applications from virus or malware attacks and provides a method for centrally deploying emergency responses for Zero-day viruses. A Zero-day virus is a virus for which the vendor does not have a fix or prevention in place. Software or Operating System Patches provide updates, enhancements or fixes to known vulnerabilities or bugs. It is important to plan for patch deployment and ongoing patch management prior to going live with your new application or product. Patching activities often require outage windows and server restarts, which can be quite disruptive for internet facing applications. Planning the frequency, maintenance window, and related lower-level environment confirmation prior to production deployment, needs to be included in the project plan and ongoing support model. Combined with an appropriate DR strategy, administrators apply security patches without affecting system uptime.

**Security Scanning.** Scanning your application source code, networks, databases and firewalls is paramount to insuring that your application does not have security vulnerabilities. Depending on the physical implementation of your servers, the sensitivity of your data, and the company policies, include appropriate security scanning and/or testing activities in your project plan. There are multiple types of security scans to consider, each with a different goal.

**Vulnerability Access (VA) Scanning.** VA scanning of an application flags potential risks for unwanted access to an internet facing application via a variety of methods: open ports, missing patches, or software coding vulnerabilities. Particularly important for externally facing internet applications, however, many security experts recommend vulnerability scanning for internal systems.

**Static Scanning.** Static scanning of the application binaries identifies code that could exploit weaknesses or is malicious.

**Dynamic Scanning.** Dynamic scanning examines the live application via application entry point or URLs (commonly used for web applications) in a manner that simulates a hacking attempt.

**Open Source Scanning.** Scanning an application for Open Source supports the requirement for registering your Open Source usage and identifying which open source libraries are active in your products.
**Penetration Testing.** Penetration testing is a non-destructive exploitation test used to determine the feasibility of a particular set of attack vectors and identify higher-risk vulnerabilities that result from a combination or particular sequence of lower level exploits. Simulated hackers identify vulnerabilities that may be difficult or impossible to detect with scanning software. There are four types of penetration tests:

1. Server level tests check patch levels and the lockdown state of the server.

2. Network device tests check Switch, Router, LB, Proxy devices patch levels, and that configuration is to company standard.

3. Firewall tests target the review of firewall rules, overall health and configuration.

4. Database test – look to identify unauthorised access flaws, check patch status, security configuration of database and hardening status.

**Government Regulations.** The Federal Information Security Management Act (FISMA) is United States legislation that defines a comprehensive framework to protect government information, operations and assets against natural or man-made threats. FISMA was signed into law as part of the Electronic Government Act of 2002. Your data centers may need to be FISMA compliant. FISMA requires each federal agency to develop, document, and implement an agency-wide program to provide information security for the information and information systems.

In summary, Information Security activities affect all phases of an IT project, from design through to deployment and ongoing operations. The financial and brand damaging consequences of a cyber attack can be devastating. Properly planning for security activities keeps your project within schedule and budget.

**SOFTWARE SELECTION**

The IT industry is constantly evolving, with new technologies and capabilities introduced into the market place quite regularly. With all of the new and oftentimes exciting software choices, it is important that the IT project manager understands the organizations desired software stack, corresponding skill sets and/or preferred vendors.

**Software Stack.** During the initial design phase of your software project, consider the following questions related to other non-functional requirements: concurrency, error handling, auditing and reporting, capacity, accessibility, and legal & regulatory.

1. Will it support your information security requirements?
2. Will it support the DR requirements and RTO/RPO objectives?

3. Will it provide the needed availability?

4. Are you selecting the right version of the software? For example, the Standard vs. Enterprise version may have different functionality.

5. Will it provide proper error handling? Do you have special auditing or reporting requirements?

6. Will it provide sufficient capacity? How many concurrent users will your configuration support? Will the software scale appropriately? Will the software meet the performance requirements?

7. Do you require special accessibility for people with disabilities?

**Skills and Standards.** Companies are always looking to save costs. By selecting the company’s standard software components (Operating System, Database, Load Balancer, Monitoring Tool, etc.) the technology organization can leverage resources and standard IT offerings to achieve these scale and cost savings objectives. In addition, implementing standard processes and procedures for various routine activities (Failover, monitoring, patching, access controls, etc.) improves response time and supportability of your applications during incidents and routine maintenance.

1. Do you have the right skill sets in your organization?

2. Do you have restrictions on where your support personnel reside?

3. Will your systems have appropriate portability? Would you be able to move to different architecture?

**Licensing/Existing Agreements.** In addition to technical assumptions about the desired software stack, understand the current vendor relationships and existing licensing agreements in place at the organization.

1. Are existing agreements/discounts in place?

2. Are you considering a different usage that may have license implications? For example, is the software licensed for a Live/Live DR scenario? Are the server specifications beyond memory limits of the license?

3. Is the Open Source Software usage subject to any restrictions?

4. Does the company have an unbalanced reliance on a particular vendor?
5. Are there government regulations to consider? Does your data center meet FISMA compliance requirements?

6. Are there restrictions or laws about housing data outside the country of original?

**Third-Party Software.** Software bought from a Third Party often eliminates the need to develop custom in-house applications. While desirable from a schedule and cost perspective, it is important to understand the consequences related to cost, DR and information security.

1. Will the Third-Party software meet the desired RTO/RPO objectives?
2. Will externally facing API’s create a dependency on a particular vendor?
3. Is there an Open Source option?
4. Does the Software pass Security Scanning requirements?

**OTHER COMMON NON-FUNCTIONAL REQUIREMENTS**

**Non-Functional Requirements (NFRs).** In addition to the non-functional requirements mentioned in the preceding sections, the below NFRs are also important.

**Accuracy & Precision.** The accuracy and precision of data.

**Accessibility.** The ability of the system to support users with disabilities.

**Auditing & Reporting.** The ability to log (or audit) data changes, system changes, configuration changes and report on these activities.

**Availability.** Also called “up time” of the system, and measures the time the system is available for use.

**Backup & Recovery.** The RTO and RPO requirements will affect the frequency, location, hardware, methodology, type of backup, storage and retention.

**Capacity.** The application or system must meet processing and computing power during normal operation, a DR event, and during periods of significant or excess load.

**Concurrency.** Design the systems to accommodate the highest number of simultaneous users.

**Configurability.** There are many environments utilized when developing an application. They may include Development, Test, Quality Assurance (QA), Pre-
production, Production and DR. Configurability allows software to move from environment to environment with simple configuration changes.

**Error handling.** Proper error handling results in the appropriate number and type of warning and alarms, with notifications automatically generated.

**Legal & Regulatory.** The system must adhere to legal requirements or government regulations.

**Licensing.** License the software and hardware in accordance with vendor requirements and end user agreements.

**Maintainability.** Software written with maintenance and support considerations accounted for is easier and more cost effective to support.

**Performance.** The system is designed to meet the performance requirements of its users.

**Redundancy.** Additional systems will prevent service disruption.

**Reliability.** Systems built with quality software on dependable hardware.

**Scalability.** Designing a system that has growth or scale potential.

**Security.** The protection of data, systems, and information against threats and inappropriate access.

**Supportability.** The ease in which operations’ teams can support the system.

**CONCLUSION**

By understanding and planning for activities in support of Non-Functional Requirements, project teams will deliver IT projects that meet the needs of the company and their customers. By incorporating related activities throughout the project life cycle, project teams will avoid delays or cost impacts due to security exposures, incompatible software, or incomplete disaster recovery capabilities. The result will be a solution that aligns with your organization’s technology policies and products that are secure, available, and supportable.
REFERENCES


3“The Open Source Definition”, http://opensource.org/docs/osd


5FISMA website http://csrc.nist.gov/groups/SMA/fisma/index.html