Transforming Business Requirements into IT Application Solutions & System Implementations

Rohit Parashar[1],
M. Tech,
Department of Computer Science and Engineering,
Ganga Institute of Technology and Management,
MD University, Haryana, India

Dr. Yashpal Singh[2]
[2]Associate Professor,
Department of Computer Science and Engineering,
Ganga Institute of Technology and Management,
MD University, Haryana, India

Each company's business has its own specific or unique requirements to implement; and no one application software system can meet and satisfy the business requirements of organizations. Organizations have involved major of its IT taskforce to integrate other party Vendor's software products or customizable commercial products into their own requirement specific computer systems. Understanding the business requirements bit by bit is the base for creating an application solution. This study delivers a course roadmap for developing system solutions that satisfy business requirements via business requirements analysis, systemdesign& architecture, developing &implementation, testing and validation, and system deployment. It is not specific to any specific system production. However, the motivations behind this paper are the practical issues faced by the industry and what should be done as pre-step to incorporate system solutions.

I. INTRODUCTION

As briefed before, one of the critical challenge for any organization is always been how to design and build software application to meet a company's business requirements. Long term Factors to be considered importantly. Factors such as - long term company goals and vision, budget, time, resource constraints, and degree of utilizing off-the-shelf software products to decrease both development cost and time have increased the complexity of building the systems.

Organizations like pharmaceutical, biotech or computer software systems which are primarily and heavily built on the regulatory norms and regulations, have helped companies to enhance productivity, accuracy, reliability, consistency, authenticity, and compliance with regulatory agencies to such a high level that was never achieved before. Building the system from scratch is not required. It reduces lot of personnel's efforts if the business requirements are analyzed and managed effectively. There are lots of commercial products or software solutions that can meet some or most of your potential business needs. For Example Software like SAS, Regulatory companies like biotech/pharmaceutical Company are using it and integrating their software products in the total architecture system. Though, it is still the most prominent challenge to meet the all business requirements from such integrated architectures. As per the standard practice, IT Companies create utilities and tools to streamline the business process and requirements. Using off-the-shelf software products is advantageous because most of these products are generally compliant with the relevant regulatory code. The important factor always remains the effective business requirements analysis. Any gap or failure in identifying the problem to be solved accurately and completely becomes root cause of system development failures in later stages. Therefore system development shall comply with all business requirements. Business problems should be studied and evaluated carefully in the context of long term strategic planning apart from the system solution being developed for any specific business criteria. Implementing solutions to immediate business purposes might cause considerable loss to the organization in the long run. This majorly happens when the business expands so fast that it triggers the discarding of legacy system or overhauls it.

This article is justifying the plan of software system project management by building the clinical data analysis and reporting system for biotech/pharmaceutical companies from the viewpoint of system development life cycle, which covers business requirements analysis, system architecture, system design, implementation, testing and validation, and system deployment, and in this order. The development life cycle schema described in the paper is not specific and can be referred to all scales and types of software development extending from small utility tools to large systems.

II. BUSINESS REQUIREMENTS ANALYSIS

Understanding the targeted business is the foundation to implement any system solution for its business requirements. A business requirement is what the application needs to do for business; what it takes to continue running the business through the IT application; and what can be improved to application solution to run the business better.

Requirements analysis is an amalgamation of business and information technology. Business analysts are responsible in both business and IT area to bridge the gap between the different domains. The main tasks of business analysts are:-

1. Identifying key stakeholders,
2. Expertise on business process,
3. Leading and coordinating the communications needed to manage the business needs
4. Documenting and organizing the business process and needs,
5. Analyzing the business needs and defining the business requirements

6. Communicating the business requirements to an entire team for agreement.

Stakeholder Analysis

The business analysts have to understand the sources of requirements and how to access them to produce information and analyze business requirements. The stakeholders being the primary source are individuals, team, or organization with interests in or concerns about the business. Examples of stakeholders are customers, end users, partners, management, business policies, regulatory agencies, and others such as business domain experts who understand the business. Concerns are drawing interests in parallel that are specific to the system's development, its operation or any other aspects that are important to stakeholders.

The stakeholder analysis focuses at understanding the responsibilities and needs, achieving agreement on business problems and solutions. The stakeholder analysis is considered as the first step at an organizational level in strategic planning activities. An initial stakeholder can be based and decided as per the business process. However identifying stakeholders' interests, impact level, and relative priority is not an easy task. The stakeholder generally belongs to a business group. It is recommended to identify representatives of stakeholders. The representatives are accountable for providing the opinions of the groups they represent.

It is important to measure and access stakeholders' importance and influence. Only after the understanding of stakeholders' needs and expectations, System analysts can manage the expectations to ensure a successful project implementation. Failing to meet any need or expectation of any stakeholder at a critical time of Software Development Life Cycle may ruin a project. Analysis requires skills to develop alliances with the stakeholders for agreement on what the problems and the proposed solutions are. Usually you have to compromise to reach agreement. Both internal project entities and external interfaces should be measured.

For example - To build a statistical analysis and reporting system for clinical trial studies - the key stakeholders will depend upon organizational structures and corporate culture. Typically the stakeholder group may include: statistical programmers, biostatisticians, clinicians, medical writers, IT groups, and regulatory bodies. The other internal stakeholders include the management members in the statistical programming team. Base on the business process which is discussed in the next section, the external stakeholders could cover the clinical database management team, report publishing team, and other teams who may request data analysis such as scientists. Although external, the database management team and publishing team are important stakeholders. Any inappropriate data inconsistency or exotic formatted data from the database management team could cause troubles to the statistical programmers. Similarly disagreed analysis reports produced by the statistical programmers would be rejected by the publishing team.

Business Process Analysis

Understanding the business process is an important process in the business requirements analysis. The aim of business process analysis is to create solutions of problems to improve the processes for effectiveness and efficiency. Business process presents how a company delivers products. Management hierarchy shows the responsibilities and reporting relationships within the company on the other hand. Compare to the static administrative organizations, business process is dynamic and often cross departmental lines. Thus the efforts to improve the quality and efficiency of a business process are likely to depend on the collaboration of stakeholders from different departments.

The business process may include business activities and the relationships among them such as dependencies. Gathering the data necessary for business process analysis requires a great deal of efforts. This is especially the case when a cross-functional collaboration becomes necessary. While collecting business process data people start to learn a lot more about how their business is actually done and begin to think about how to do it better. The data gathered are used to create a process map or flowchart, which is a graphic representation of the sequence of activities. The process map may be in the form of a cross-functional flowchart that includes relations among these activities. The business process of clinical data management, analysis, and reporting in the biotech/pharmaceutical industry can be described as collecting and processing Case Report Form (CRF) data (and other supporting data), storing data in a database, analyzing the data, and generating reports. The process can be illustrated as a business model. A top level view of the business process is shown in Figure 1.

Based on the business process of clinical data management, analysis, and reporting, some biotech/pharmaceutical companies organize their business management teams under one umbrella. Multiple departments can be found under the same management, such as database management, biostatistics, and programming teams. The administrative organization like this would make it easier to reach agreement on interfaces among business models and acceptance of business processes and requirements across the teams. This is easy to understand because administrative organizations should align with business process at least at certain degree or level.

Analysis of the relationships among the business models facilitates the system architecture and design. The CRF data source affects both the data management system and the database system which in turn affects the analysis system.

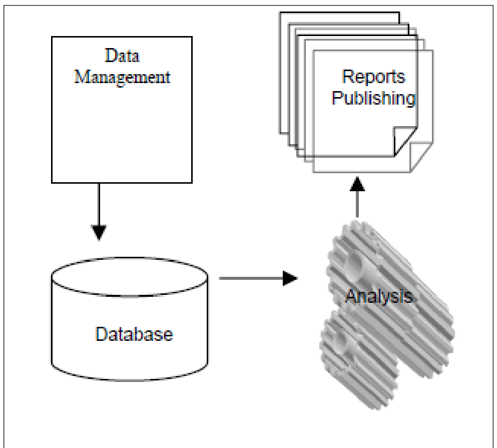

Figure 1: Clinical Data Process

Collecting and Documenting Business Needs

Identifying the stakeholders and analyzing the business process are the initial steps for understanding the business requirements and it should be documented. Various practices exist for requirements elicitation, gathering, and documentation. Major techniques for collecting business needs include

1. Interviews,
2. Brainstorming,
3. Questionnaires,
4. Prototyping,
5. Informal Use Case Analysis,
6. User stories/ Change Requests.

Discussing requirements with users, going through relevant documents, knowledge of the stakeholders' vision about the long term, developing alternative approach, and designing process diagrams serves well for your business requirements collection. Any sample business process can be affected by multiple business conditions. These conditions detail the efforts to gather the requirements for supposed business requirements and specifications.

There may be just an idea at beginning to start the business, Data might need to be collected by asking questions such as how well does this part of the business process operates; what could lead to setbacks from the proposed process, is there any redundancy that is not adding any value to the process instead consuming the project budget; Can the better results be achieved with less cost and with alternate approach; are the business process or technology better serving and meeting feasibility of business requirements?

There are two major reasons for conducting business requirements analysis. Firstly is to increase and advance the quality and efficiency of the business. Secondly is to reduce the cost of accomplishing the suitable outcome. No matter what the goal is there to achieve, the result is always weighed against the costs of obtaining it. The businesses requirements are constantly changing constantly with new or advanced technologies are emerging. Sometimes it is necessary to bring an outsider business analyst to help with constantly changing business requirements analysis in case that specific department has not been able to absorb the external resources or knowledge.

In the regulated industry, business rules must be taken into consideration during the requirements analysis. Business rules are the specialized form of logic that expresses a constraint about the way that a system or the people using it behave. These rules are guided by a variety of elements including regulatory agencies, industry standards, business expertise, or common sense. The activities in each of the business models should be depicted as much in detail as possible, including the business rules. The result of requirements elicitation is a list of requests or needs that are described textually and graphically with given priority relative to one another. A description of when and how the requirements need to be addressed is recommended.

WRITING BUSINESS REQUIREMENTS DOCUMENT

A BRD or business requirements document may consist of, but not limited to, the following:

1. Objectives of Business
2. Stakeholders
3. Organizational Processes
4. Scope of Problem or identified business requirements
5. Scope of Improvements
6. Suggested Solutions

This document forms kind of the legal contract between the client and the software developer in commercial software development. The document should be well organized, Comprehensive and rich sufficiently. The business requirements document must be agreed to by all the stakeholders most importantly. There is no standard about the context and volume the document should detail. The level of the detail requested in a requirements document depends on the size of the software to be built, the interface to other systems, the stage in requirements gathering, the level of domain and the technology expertise, the cost, and so on. The business requirements document for huge applications or large systems is arranged in a hierarchy or phase wise generally. It may contain sub-business requirements, and many functional requirements and non-functional requirements. Functional requirements illustrate what the system should perform and non-functional requirements describe conditions that must be satisfied. The specification in the business requirements document should be categorized built on priorities, long and short term vision, budget, time taken to implement, and others.

Some functional requirements may be mandatory and others may be optional. This stands true especially when there are lists of requirements and only few are taken for implementation due to budget or development time constraint. The goal of prioritization of the requirements is to serve the business purpose aligning with the department and corporate goals. When business requirements are conveyed to and agreed by the entire project team, the stakeholders, mainly business analysts, management, system architects may decide to calculate software alternate approach. The major decision is generally based on build vs. buy. The reasons for buying a software package are less time to implement, proven reliability and performance, less technical development staff, lower cost, and future upgrades provided by the vendor. The reasons for in-house development include meeting unique requirements, meeting constraints of existing systems and technology, minimizing changes in business procedures and policies, and developing internal resources and capabilities.

Practical Example – Statistical Estimation: It is a common practice that statisticians and statistical programmers write specifications for programming; and then write programs for generating statistical analysis results based on the specifications. Before the results can be used for any estimation or specification the programs must be validated as illustrated in Figure 3.

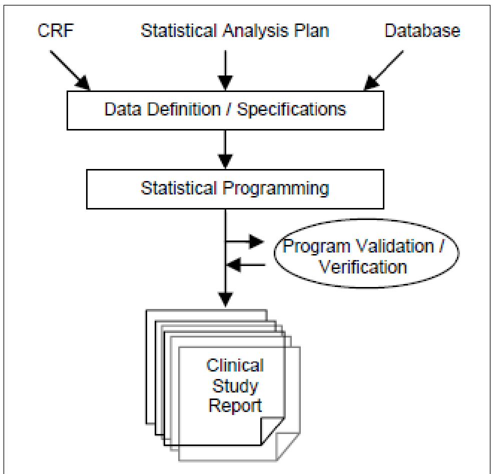
CRF
Statistical Analysis Plan
Database
Data Definition / Specifications
Statistical Programming
Program Validation / Verification
Clinical Study Report

Figure 3. Clinical statistical programming process

SAS is an excellent software system for data analysis and reporting for statistical programmers. However, the following is a list of identified requirements that programmers have to handle by themselves:

- Data definition table (DDT). Programmers and statisticians should specify specifications to define data before can be programmed using SAS.
- Program version control. Versioning the programs becomes the best choice for tracking the changes and preventing accidental change or loss of the programs. Programmers write programs and modify the programs several times during the process of SAS program development.
- Program validation or verification. Programmers need tools to manage the process of validating the programs to make sure that the programs they write do what they are supposed to do and do it correctly.
- Standard programs. Many programs that programmers write for analyzing data for one case can be used for another case with or without modification. It would be a great saving of resource and risk reduction if these programs can be standardized.

It is appropriate from the viewpoint of management to establish a dedicated group for dealing with issues like those listed above. These are just examples of business needs that statistical programmers require.

Data Definition Table (DDT): The DDT is a specification document that defines all datasets and variables to be included in BRD or case report tabulations (CRT) and analysis. Sometimes variable definition table (VDT) is used as synonym. The variables in the DDT are defined based on the type of case report form (CRF), statistical analysis plan (SAP) and data schema. Figure 4 shows simple process of defining data.

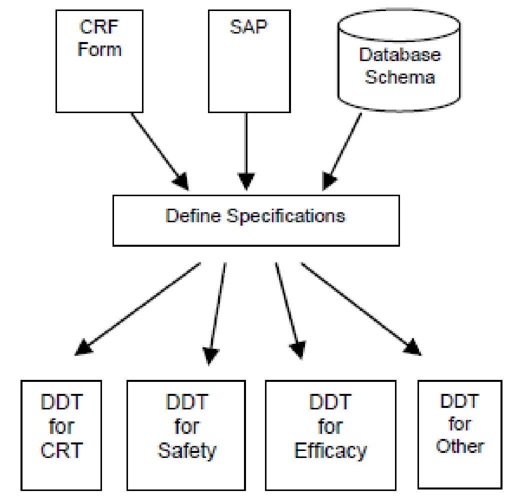

Figure 4. Data definition process

DDT is used as the basis for both generating and validating CRTs and analysis data sets. Statisticians define the DDT usually; but programmers are the owner of DDT. The requirements for the DDT usually include data set names and labels, and variable names, labels, types, formats, Sign off on the algorithms and derivations and derivation methods. Designing and architecting the DDT make a great difference and challenging in meeting the programmers prerequisites in terms of flexibility, accuracy, and reusability.

Program Version Control: Program version control supports programmers tracking program changes. The minimum functionalities for version control need tracking when a program is changed and what is changed thus, preventing accidental change to the program. A more powerful requirement should contain the program modified before or on a specific date as the new version. It is not required to write the own software to meet your version control business need. Several different companies offer adequate version control software products. A full version of version control software allows for functionalities such as branching, merging, and comparison in addition to the basic functionalities.

Program Validation: Program validation is a quality control process of clinical trial data analysis. It verifies that a program complies with required quality control procedures e.g. SOPs and guidelines; and it also attempts to prove that a program produces correct results. Although there are several ways to verify a program, one of the most commonly used processes in the biotech/pharmaceutical industry is double programming, which is also, called independent programming. A validation programmer writes his/her own statistical program independently based on the same specifications used by the original programmer, and compares the result with that of the original programmer. Although this kind of validation cannot absolutely guarantee the correctness of the statistical program, it reduces the risk of making mistakes to the minimum. Practically there is no other better way or good tools to use to do it at this time. For each trial data, programmers may write from dozens to several hundreds of statistical programs. These programs are often modified and validated several times before they can be finalized for production. How do the programmers and validators track the modifications and validations and get the most updated summary of the validation status for a specific clinical study during this process?

Standard Programs: There are at least two ways to write standard programs. The simplest way is to write standard program templates. You copy a template program and modify it to fit your program need. This is a good practice when your needs for different clinical trials are similar but not exactly same and the difference varies among trials. However a lot programs such as those required for generating demographic or disposition tables don’t change among trials; and they may be incorporated into your system as true standard programs. To use a true standard program each time you just call the program for generating required table, listing, or graph. The advantages of using the standard programs over using templates are that standard programs are easier to use; and they are validated hence saving a lot of resource writing validation programs each time it is used. Your business needs, resource availability, long term commitment, and other factors may be taken into consideration for an effort to standardize your programs.

III. SYSTEM ARCHITECTURE

To define the system means to translate stakeholder needs into a meaningful description of the system to be built based on business requirements. In fact a key factor in the success of the system architecture is the extent to which it is linked to business requirements. It is a good practice to derive the characteristics of the architecture directly from the high-level business requirements.

The system architecture defines the structure of a system, which covers its software elements, their external visible properties, and the relationships among them. This means that architecture specifically exclude certain information about those elements that are not related to their interaction. Thus, architecture is an abstraction of a system that omits details of elements that do not affect how they use, are used by, and relate to other elements. In large projects, the system elements are almost certainly subdivided. Every software system has architecture because it consists of elements and relations among them. The behavior of each element is part of the architecture as the behavior can be observed from the viewpoint of another element. The behavior is what allows elements to interact with each other. This does not mean that the exact behavior and performance of every element must be documented in all circumstances; but to the extent that an element’s behavior influences how another element must be written to interact with it or influences the acceptability of the system as a whole. The system architecture forms the backbone for building a successful software system. Architecture represents a common vehicle for communication among the system's stakeholders. Conflicting goals and requirements are mediated during system architecting. The system architecture must be based on the business requirements, which is called the requirement-driven architecture. During the system architecture and design, the business analysts and system designers have to investigate the pros and cons of different interactions among the systems from a business perspective, especially from the viewpoint of the cost and user’s requirements. A view is a representation of a whole system from the perspective of a related set of concerns. A viewpoint is a specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views. The relationship between viewpoint and view is analogous to that of a template and an instance of that template. Each stakeholder of a software system such as user, project manager, coder, analyst, tester, and so on is concerned with different characteristics of the system that are affected by its architecture. Each of their views provides a different perspective and design handle on a system. The chosen set of views together show the entire architecture and all of its relevant properties.

The typical progression of system architecting is from business to technology, using business requirements analysis to properly align with all pertinent concerns from high level overview to lower level detail. The concerns and requirements of the stakeholders are continually referred throughout the process. In case of improving the existing system, each of these progressions has to be made distinctively for the existing environment and the target environment. The architect may choose to develop pertinent business and technical architecture views of both the existing system and the target system, which establishes what elements of the current system must be carried forward and what must be removed or replaced. The final product of the system architecture is a system architecture document which contains the viewpoints, relevant views, and information that applies to more than one view to give a description of the system. Both natural and graphical languages are used to depict the architecture.

IV. SYSTEM DESIGN

System architecture and system design have different focuses. Often people are confused about them; and the two terms have been used interchangeably. System architecture is concerned about the system’s structures dealing with system elements, their external visible properties, and the relationships among them omitting details of elements known to them. Functional requirements are partitioned during the architecture phase; whereas the functional requirements are accomplished during the design phase. Every property or attribute of each element of the system is described in detail in the system design. However system design is based on the framework of the system architecture document.

The output of the system design is the software design description (SDD). The SDD is a representation of a software system created to facilitate analysis, planning, implementation, and decision making. It is a blueprint or model of the software system. The components are usually classified as user interface component, problem-domain component, task-management component, data-management component, and others so that they can be considered, implemented, changed, and tested. Decisions are made about what programming language(s), tools and libraries, database models, data structures, and major algorithms will be used for the system solution during the system design phase. The design description model used to represent a software system can be expressed as a collection of design entities; each possesses properties and relationships. Entities can exist as a system, subsystems, data stores, modules, programs, and processes. Organization of the SDD into separate design views facilitates information access. The decomposition description view shows the partition of the system into design entities; dependency description view reveals the relationships among entities and system resources; interface description view lists everything a designer, programmer, or tester needs to know to use the design entities that make up the system; and the detail description view depicts the internal design details of an entity.

Again for illustration purpose, the status manager is used as an entity. Its attributes are:

1. Identification: status manager.
2. Type: process.
3. Purpose of existence: to establish communication channel with users; and process users' inputs and return results.
4. Function: serving as the processing center for the system to be built, status manager takes inputs from program manager, programmer, or validator; and parses them and commands the entity messenger to perform its clients' requests; and then it processes the data returned by messenger and sent the results to its clients.
5. Subordinates: entity status manager does not have any child entity.
6. Dependencies: entity status manager depends on entity messenger to store information to or retrieve information from entity validation database; it also depends on the messenger to obtain program information from the file system.
7. Interface: the entity interacts with program manager, programmer, validator, and messenger. The method of interfacing is to be determined (TCP/IP).
8. External resources: operating system and network.
9. Data and data structure: the entity mainly uses arrays as its data structure
10. Processing: the entity uses program date, validation program date, and most recent validation completion date to make decision on a program validation status. If the most recent validation completion date is missing the status is "validation needed"; else if the most recent validation completion date is greater or equal to program date the status is "completed"; else if the most recent validation completion date is less than program date the status is "revalidation needed". The programs validation status for a clinical study in terms of percentage of completion for each type of programs such as data sets, tables, listings and graphs is computed by number of programs completed divided by total number of programs in the category. A summary of overall status and detailed status for each program is computed and sent to program manager.
11. Designing and planning integration testing at design phase allows risks to be addressed early in the development cycle. The output from software design phase should ensure their testability. Integration testing (also called module testing) addresses modules' interfacing together in a correct, stable, and coherent manner.

V. IMPLEMENTATION

Software implementation is governed by software architecture and design documents ensuring meeting business requirements. During coding and testing, developers must validate that all requirements are satisfied and all features are implemented. Good naming convention makes it easier to maintain source code; and it also reduces confusion during software development and testing. For variable, method, class, package, database and other naming conventions for coding, software engineers are recommended to follow relevant coding conventions. Forgetting initializing data is quite common in coding. The good practice is checking input parameters for validity, initializing variables as they declared, paying special attention to counters and accumulators, and checking the need for re-initialization. Unit testing is typically conducted by the development team and specifically by the programmer who coded the unit.

VI. TESTING AND VALIDATION

Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. It not only carries intent of finding defects but also explores the status of the benefits and the risk associated with release of a software system. Validation is a newer concept. It ensures that the system is implemented according to the relevant requirement documents such as business requirements, system architecture, design, etc. Testing applies to all applications including their components, modules, and subsystems. Testing includes unit, integration, system, system integration, and user acceptance testing.

Each development phase is linked to a corresponding testing phase. The testing phases are given the same level of management attention and commitment as the corresponding development phases in terms of planning, allowing any risks to be addressed early in the development cycle. The output from the development phases can be viewed by the testing team to ensure their testability. Acceptance testing is considered during business and system requirements analysis phase; system testing is considered during system architecture; integration testing is planned during system design; and unit testing is planned and implemented during system implementation.

UNIT TESTING: Unit testing represents the lowest level of testing. It is conducted to ensure that reliable program units (or software components) are produced that meet their requirements. Unit testing is typically conducted by the development team and specifically by the programmer who coded the unit. “Independent” observation of the testing process is likely to be performed by the project leader or by another member of the project team who will be expected to countersign the appropriate test result record form to confirm that the correct procedures were followed during the testing.

INTEGRATION TESTING: The purpose of integration testing (also called module testing) is to demonstrate that the modules which comprise the Application Under Test (AUT) interface and interact together in a correct, stable, and coherent manner prior to system testing. Integration testing is typically a black box testing conducted by the development team and involves independent observation of the testing process. Testing issues associated with integration testing such as test planning and review of testing requirements and high-level test design should be considered during the design phase.

SYSTEM TESTING: The objective of system testing is to establish confidence that the AUT will be accepted by its users. System testing is conducted by the test team. System testing should employ black box testing techniques and will test the high level requirements of the system without considering the implementation details of the component modules.

SYSTEMS INTEGRATION TESTING: Systems integration testing is to provide confidence that the AUT is able to interoperate successfully with other specified software systems and does not have an adverse effect on other system in the live environment, or vice versa. It is possible that the testing tasks performed during system integration testing may be combined with system testing, particularly if the AUT has little or no requirement to interoperate with other systems. System integration testing should deploy black box techniques.

USER ACCEPTANCE TESTING: The purpose of the user acceptance testing is to confirm that the AUT meets its business requirements and to provide confidence that the system works correctly and is usable before it is formally released to the end user. User acceptance testing is conducted by one or more user representatives with the assistance of the test team. User acceptance testing should employ a black box approach and the user representative will test the AUT by performing typical tasks that they would perform during their normal usage.

VII. SYSTEM DEPLOYMENT

To install a system and maintain and support it, a description of how the parts of the system fit together is a must. It is a good practice to develop some form of deployment model. Deployment models force you to think about important deployment issues long before you must deliver the actual system. When determining how to model the deployment architecture for a system, try to consider fundamental technical issues such as existing systems that your system needs to integrate with, robustness of your system, connection and interaction between your system and existing system, middleware including the operating system and communications approaches that your system uses, hardware and software your users directly interact with, approaches to monitor the system once it has been deployed, security of your system; collect critical information for anyone involved in development, installation, or operation of the system; and plan and perform training on deploying, maintaining, and supporting the system.

The system installation information shall include the required hardware and other constraints (e.g., minimum memory requirements), detailed instructions for the installer, and any additional steps that are required prior to the operation of the system (e.g., registering the software). The type of software to be installed and the expected level of expertise of the installer shall be considered when writing installation instructions. In some cases, the installation planning shall include defining the order of installation at several sites. It could also define one or more configurable options that are to be handled in the installation process.The system release manager is responsible for packing the system for deployment, testing, and actually deploying the system. It is not the end of the system development after the system is deployed. Many issues will rise when it is put in use; and they will be put back into the software development life cycle. The development life cycle for the system ends when the system is obsolete and uninstalled.

VIII. CONCLUSION

The goal of system development is to meet the business requirements. A development effort is initiated with the identification of business requirements for a system to be developed, whether it is a new effort or a change to all or part of an existing application. Potential approaches shall be based upon the business requirements and any data pertinent to the decision to develop or acquire the system, using resource information, budget data, the availability of third party or existing reusable software products, and others. This phase in the development lifecycle is the most critical an often the most overlooked. Without adequate attention paid to stakeholder and business needs the subsequent steps may and will likely be inappropriately directed.

Software project management planning requires the collection and synthesis of a great deal of information into coherent and organized software project management planned information (SPMPI) based on the software life cycle process (SLCP). This activity details the project organization and assign responsibilities. Standards, methodologies, and tools for management, quality, evaluation, training, documentation, and development shall be specified; project budget and staffing shall be allocated; and procedures for scheduling, tracking, and reporting shall be defined. Issues such regulatory approvals, required certifications, user involvement, subcontracting, and security shall be considered.

An iteration of the system development includes requirements analysis, system architecture, design, implementation, testing and validation, and system deployment. The system architecture must be a requirement-driven architecture. The system development life cycle includes maintenance of the system. It never stops until the system is obsolete and uninstalled.

REFERENCES

[1] ANSI/IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems –Description. The Institute of Electrical and Electronics Engineers, Inc.

[2] Bass, Len; Clements, Paul; &Kazman, Rick. 2003. Software Architecture in Practice, Second Edition. Boston, MA: Addison-Wesley.

[3] Carnegie Mellon Software Engineering Institute. How Do You Define Software Architecture? http://www.sei.cmu.edu/architecture/definitions.html. Last Modified: 15 February 2006.

[4] Chang, Tony and Soloff, Dana. 2005. "A Simple Solution for Managing the Validation of SAS® Programs That Support Regulatory Submissions". SAS Conference Proceedings: SUGI30.

[5] Clements, Paul; Garlan, David; Little, Reed; Nord, Robert and Stafford, Judith. 2003. "Documenting Software Architectures: Views and Beyond". Proceedings of the 25th International Conference on Software Engineering (ICSE.03).

[6] IEEE Std 1016-1998. IEEE Recommended Practice for Software Design Descriptions. The Institute of Electrical and Electronics Engineers, Inc.

[7] Smith, Larry. 2000. "Project Clarity through Stakeholder Analysis". CrossTalk, The Journal of Defense Software Engineering. Vol.13 No.12 Pages 4-9.

[8] Sommerville, Ian, and Sawyer, Pete. Requirements Engineering: A Good Practice Guide. New York: Wiley, 1998. 64.

[9] Weinberg, Gerald M. Quality Software Management, Vol. 1, Systems Thinking. New York: Dorset House, 1992. 155.