

4.5 SDLC Common Roles & Responsibilities

According to NYS Project Management Guidebook, in SDLC, the team members have the following roles & responsibilities:

(a) The Project Team

— It consists of a Project Manager and a variable number of Project Team members who are responsible for planning and executing the project. Team members specific to the System Development Lifecycle are described below.

(b) The Facilitator

— The one who leads sessions to identify business requirements and issues, keeps session's focused and productive, draws out issues and ideas from all participants, and maintains clear and open communications within the session.

(c) The Business Analyst

— The one that effectively leads discussions with the Customers to determine the business requirements, participates in preparing the data and process models, prepares module specifications, test data, and user documentation materials, assists in prototyping activities, and develops strategies for testing and implementation.

(d) The Database Administrator

—The one who is responsible for providing and maintaining database administration policies and procedures, approving and executing database scripts, performing database tuning activities, and transforming a pictorial representation of the system data into physical database tables that support the final system.

(e) The Data/Process Modeler

— The one that develops and maintains data and process models to represent the business information needs in the area under study, develops and defines the data dictionary, validates models with the Customers, and participates in prototyping.

8.0 Overcoming Requirements Modeling Challenges

Consequently, we may have understand that very often requirements modeling efforts are undermined by the project team environment – it is common to discover that an organization's culture isn't conducive to effective software development efforts or project stakeholders do not understand the implications of their decisions. In this paper, I had identify common challenges, or at least issues which are perceived as challenges, that many development teams face when it comes to requirements modeling which include:

(1) But We Gave You Exactly What You Asked For

The requirements gathering phase is rather an iterative process. By so doing, the risk of giving the stakeholder what they ask for, rather than what they really need, is increased. The requirements definition phase of a software project is never the self-contained function implied by many software development life-cycle models. Thus, it is not enough to obtain the stakeholders' requirements once and assume that they are correct.

(2) I Know That I Think I Know What Our Requirements Are

This risk is a very common occurrence. It is characterized by not involving project stakeholders throughout the development effort. By not taking necessary steps to assure that we understand the requirements, we are inviting project rework that will result in schedule delays and cost overruns. However, typically this requirements risk will not be identified until stakeholder testing or implementation.

(3) Overboard Assumptions

The risk of assuming that developers and customers have the same thoughts about system requirements is like assuming that we all agree on political or religious issues. At this stage of development, there will almost certainly be negative impacts on both project cost and schedule. These issues are often not identified until the customer first sees the application.

9.0 Recommendation — Strategies to Mitigate Requirements Risks

In this paper, I also discuss potential solutions for dealing with those challenges which encompass:

(1) Develop and Follow Sound Processes and Procedures

The problem comes when the process that I follow and the process others in the organization follow is not the same. This lack of consistency leads to confusion, misunderstandings, development delays, and cost overruns.

An important step to mitigating requirements risks is for the organization to develop and strictly follow sound processes and procedures relative to requirements engineering. These processes and procedures should include direction to developers in the following areas:

- Requirements elicitation.
- Requirements analysis.
- Documentation of requirements.
- Requirements verification, review, and approval.
- Configuration control of requirements.
- Requirements traceability.

The organization should also ensure that roles and responsibilities relative to these processes and procedures are clearly defined.

(2) Incorporate Requirements into All Software Life Cycles

By now, the need and value of having organizationally accepted software lifecycle models and methods should be well established. From our research and knowledge of various life-cycle models, they all include a requirements analysis, requirements management, or stakeholder requirements phase. By assuring that the life cycle(s) approved for usage within the organization require proper levels of requirements administration, the risk associated with projects relative to requirements will be reduced.

interpretations based on the background of the various stakeholders.

Box-based documentation uses geometric symbols to represent various aspects of the system requirements. This approach is designed with the software engineer in mind and is generally more comfortable for the software engineer to follow and understand. This approach is traditionally more difficult for end users to understand because of the learning curve associated with this type of documentation.

Graphic-based documentation is the most recent of the three documentation forms. It was developed to support object-oriented development and design techniques. This method uses geographic symbols to represent the actual objects within a system. As with the box-based methods, graphic-based documentation is tailored more toward the developer than to the end user's understanding.

We recommend that some combination approach be adopted, experience using a combination of graphics and text. This approach helps the users insert themselves into the process or requirement being documented. It also is easily understood by the developers and serves as a useful tool to both elicit and validate requirements.

(6) Validate Software Requirements

As requirements are documented, the stakeholders should validate them. Requirements validation is an essential step to ensure that requirements are properly understood and documented. Often the act of requirements validation will uncover requirements issues that can be discovered in no other way. This mitigation strategy goes hand in hand with the need for dedicated stakeholder involvement.

(7) Hold Formal Requirements Reviews

Studies indicate that document reviews greatly reduce the errors in critical documents. Errors, inconsistencies, ambiguities, and confusion can be greatly reduced by holding formal reviews of software requirements documents. Formal reviews will greatly improve the quality of software requirements documents. Teams performing requirements management activities should be trained in sound review methods.