IT Project Management (Finals) - Jaish Khan

Table of Contents

---

1. Project Planning

Overview

Project planning is the foundation of successful software development. It involves understanding stakeholders, organizing teams, and creating comprehensive documentation to guide the project's execution.

1.1. Stakeholders and Users

A **stakeholder** is anyone with interest in the software completion. A **user** is someone who will use the software to perform tasks.

Difference between Stakeholder and User

While all users are stakeholders, not all stakeholders are users. Understanding this distinction is crucial for proper requirement gathering and project scope definition.

1.2. Roles in Software Development Team

The software development team consists of:

- *Business Analysts*: Requirements gathering and user communication
- *Designers* and *Architects*: Technical solution planning
- *Programmers*: Code implementation
- *Testers*: Verification and validation

Team Communication

Effective communication between team members is crucial. Regular stand-ups and team meetings help ensure everyone is aligned with project goals and deadlines.

1.3. Project Management

> It is the application of knowledge, skills, tools, and techniques to project activities to satisfy stakeholder needs and expectations.

> **Project Manager** serves as the direct link to the client and must manage expectations, ensuring alignment between client desires and what is practically

> achievable.

Project Manager Responsibilities

- Identifying users and stakeholders
- Determining stakeholder needs
- Team coordination
- Task assignment
- Knowledge management
- Scope definition
- Stakeholder communication

> **Critical Project Manager Tasks**
>
> The project manager must maintain a balance between stakeholder satisfaction, team productivity, and project constraints (time, budget, scope).

1.4. Vision and Scope Document

A project manager's tool to demonstrate to the stakeholders that the project team understands their needs and will address them in the software.

This document should include

1. **Problem Statement** → includes the *background* of the project, the *stakeholders* involved, the *users*, the *risks*, and any *assumptions* that are being made.
2. **Vision** → includes the vision *statement*, a *list of features*, the *scope* of each phased release, and any *features* that will not be developed.

> **Document Template**
>
> A good vision and scope document should be concise yet comprehensive, typically 10-15 pages long, focusing on what's needed rather than how it will be built.

> **Documentation Risks**
>
> Insufficient or outdated documentation can lead to project failure. Keep all documents up-to-date and easily accessible to the team.

1.5. Project Plan

The project plan defines all the work that will be done on a project and who will do it.

- **Statement of Work (SOW)** → Describes all the products that will be produced, including a list of features, a description of each deliverable, and an estimate of the effort involved for each product.
- **Resource List** → A list of all people, hardware, rooms, and anything else required for the project with an indication of availability and cost.
- **Work Breakdown Structure (WBS), Estimates, and Project Schedule** → A hierarchical breakdown of all the tasks needed to produce the deliverables, with effort estimates for each task, and a schedule of when each task will be completed.
- **Risk Plan** → A list of all the risks that could threaten the project, the probability of them occurring, the impact they would have, and a plan to mitigate them.

---

2. Estimation

> Project manager must set expectations about the time required to complete the software among the stakeholders, the team, and the organization’s management. If those expectations are not realistic from the beginning of the project, the stakeholders will not trust the team or the project manager.

Estimation Purpose

The goal of estimation is not to predict the exact completion date, but to provide a reasonable range based on known factors and assumptions.

2.1. Wideband Delphi Process

The Wideband Delphi technique offers a structured approach to estimation. In this process, a team of 3-7 members works through several rounds of individual estimation and group discussion to reach consensus. Here's the detailed process:

1. **Team Selection**:
    - Project manager selects the estimation team (3-7 members)
    - A moderator is chosen who understands the Delphi process
    - Ideally, the moderator has no stake in the outcome
    - Project manager should be part of the estimation team, not the moderator
2. **Kickoff Meeting**:
    - Team members review and understand the Delphi process
    - Everyone reads the vision and scope document
    - Team becomes familiar with project background and needs
    - Team brainstorms and documents assumptions
    - Generate a WBS with 10-20 tasks
    - Agree on estimation units
3. **Individual Preparation**:
    - Each team member independently generates estimates
    - Members document effort estimates for each task
    - Additional assumptions are noted as needed
4. **Estimation Session**:
    - Team works toward consensus on effort for each WBS task
    - Members fill out estimation forms individually
    - Session divided into rounds of revision based on group discussion
    - Individual numbers are not discussed
    - Moderator collects forms and plots effort sums on a line

    - Team resolves disagreements, often by adding assumptions
    - Process continues until estimates converge or team won't revise further
5. **Task Assembly**:
    - Project manager collects final estimates
    - Compiles final task list, estimates, and assumptions
6. **Results Review**:
    - Project manager reviews final task list with estimation team
    - Team validates final compilation

> **Best Practice**
>
> The Wideband Delphi process works best when team members have diverse experience levels and backgrounds, leading to more balanced estimates.

2.2. Alternative Estimation Methods

- PROBE (Proxy Based Estimating)
- COCOMO II (Constructive Cost Model)
- Planning Game (from XP)

2.3. Review Processes

> **Review Purpose**
>
> Reviews help catch issues early, reduce costs, and improve overall project quality.

2.3.1. Types of Reviews

> 1. **Inspections** → moderated meetings in which reviewers list all issues and defects they have found in the document and log them so that they can be addressed by the author.

The goal of the inspection is to repair all of the defects so that everyone on the inspection team can approve the work product.

Inspecting Meetings:

1. A work product is selected for review and a team is gathered for an inspection meeting.
2. A moderator is chosen to moderate the meeting.
3. Each inspector prepares for the meeting by reading the work product and noting each defect.

4. A defect is any part of the work product that will keep an inspector from approving it.
5. Discussion is focused on each defect, and coming up with a specific resolution. It's the job of the inspection team to do more than just identify the problems; they must also come up with the solutions.
6. The moderator compiles all of the defect resolutions into an inspection log.

> 2. **Deskchecks** ⟶ a simple review in which the author of a work product distributes it to one or more reviewers.

The author sends a copy of the work product to selected project team members. The team members read it, and then write up defects and comments to send back to the author.

- Unlike an inspection, a deskcheck does not produce written logs which can be archived with the document for later reference. Deskchecks can be used as predecessors to inspections.

> 3. **Walkthroughs** ⟶ an informal way of presenting a technical document in a meeting.

Unlike other kinds of reviews, the author runs the walkthrough: calling the meeting, inviting the reviewers, soliciting comments and ensuring that everyone present understands the work product.

- Walkthroughs are used for someone who does not have the technical expertise to review the document.
- After the meeting, the author should follow up with individual attendees who may have had additional information or insights. The document should then be corrected to reflect any issues that were raised.

> 4. **Code Reviews** ⟶ a special kind of inspection in which the team examines a sample of code and fixes any defects in it.

In a code review, a defect is a block of code which does not properly implement its requirements, which does not function as the programmer intended, or which is not incorrect but could be improved.

- It's important to review the code which is most likely to have defects. This will generally be the most complex, tricky or involved code software that only one person has the expertise to maintain or code that implements a highly abstract/tricky algorithm or a difficult object, library or API or code written by someone who is inexperienced or has not written that kind of code before, or written in an unfamiliar language.

> 5. **Pair Programming** → a technique in which two programmers work simultaneously at a single computer and continuously review each others' work.

Two programmers sit at one computer to write code. Generally, one programmer will take control and write code, while the other watches and advises.

2.4. "Project Needs" Identification

> **Needs Assessment**
>
> A thorough needs assessment at the start of the project can prevent many common problems from occurring later in the development cycle.

Key Areas

1. Stakeholder Analysis → Identify all stakeholders, Document their interests and influence and Determine communication needs
2. Resource Assessment → Required technical skills, Hardware and software needs, Training requirements and Budget constraints
3. Risk Evaluation → Technical, Business, Resource or Schedule risks
4. Success Criteria → Measurable objectives, Quality metrics, Performance indicators and Acceptance criteria

---

3. Why Projects Fail

⚠ **Key Issues**

Most project failures stem from leadership gaps, poor planning, estimation issues, and communication breakdowns.

3.1. Planning Problems

1. **Lack of Leadership** → It takes more than a talented and motivated team to make a successful project. It shows itself in the team members suffering from: Tunnel vision, Over-reliance on gut instincts and Repeated false starts in the project.
2. **Mid-Course Correction** → A change in project priorities throws the team into disarray. This usually comes from a lack of understanding of the scope of the project.
3. **Detached Engineering Team** → There is an artificial wall between the people who build the software and those who need it.

ⓘ **How to Fix**

- Use a vision and scope document to define the needs of the users and stakeholders.
- Use a project plan to keep every informed about how those needs will be met.
- Use risk planning to keep the plan realistic.

3.2. Estimation Problems

1. **Padded Estimates Generate Distrust** → Programmers add extra time to their estimates and Managers quickly figure this out, and start to question individual estimates.
2. **Self-Fulfilling Prophecy** → A project manager under pressure simply imposes a deadline, and creates unrealistic estimates that meet it.

ⓘ **How to Fix**

- Adopting a repeatable estimation process like Wideband Delphi can help fix them.
- By writing down assumptions, the team can handle risks without padding their time and even avoid the risks altogether.

- It reduces padding and increases honesty through transparency, by letting the team correct each other in an open meeting.

3.3. Scheduling Problems

1. **Working Backwards From a Deadline** → PMs approach a non-negotiable deadline for a project by working backwards; by shortening the tasks or cutting them entirely until everything fits.
2. **Misunderstood Predecessors** → PM does not take the time to understand how tasks depend on each other. Problems are then discovered partway although the project one task can't be started because it depends on another.

ⓘ How to Fix

- They can be avoided by adopting good planning and estimation practices and creating a project schedule.
- Schedule techniques like critical path analysis can help spot problems early on.

3.4. Review Problems

1. **Problems Are Found Too Late** → Preventable defects in the software that aren't caught until late in the project.
2. **Big, Useless Meetings** → A project manager who has previously been burned by problems that were found too late is determined to avoid falling into the same trap
3. **The Indispensable "Hero"** → One "critical" person is seen as the clear top programmer, and all important work is sent through him.

ⓘ How to Fix

- Reviews can catch defects early, when they are cheaper to fix.
- A review meeting only includes the people necessary for the work to be done.
- Reviews – especially code reviews – can help the "hero" spread his expertise and knowledge.

3.5. Requirements Problems

1. **Iteration Abuse** → Iteration can be a useful tool, but it is often abused. The team uses iteration as a "guessing game".
2. **Scope Creep** → After the programming has started, users and stakeholders make changes. Each change is easy to describe, so it sounds "small" and the programmers agree to it. Eventually, the project slows to a crawl.

> **How to Fix**
>
> - The team can adopt software requirements engineering practices to write down most of the changes before the work begins.
> - A change control process gives them a handle on the few changes that remain.

3.6. Programming Problems

1. **Haunted by Ghosts of Old Problems** → Programmers find that old bugs suddenly reappear without warning.
2. **Broken Builds** → The programmers deliver a build which does not work – and the testers can't even begin to test it.
3. **Spaghetti Code** → Maintaining old code is the least desirable programming job in many organizations.

> **How to Fix**
>
> - Get control of the source code with version control software like Subversion
> - Use unit tests and test-driven development to increase the quality of the build
> - Use refactoring to keep the code readable

3.7. Testing Problems

1. **Requirements Haven't Been Implemented** → The team delivers software with missing behavior or even entire features
2. **Obvious Bugs Slipped Through** → Inexperienced testers are expected to just "bang on the software"
3. **"But It Worked For Us!"** → When a product is not tested in all environments in which it will be used, the tests will be thrown off

> **How to Fix**

- Software testers must be involved in every stage of development.
- Test planning must be given adequate time on the schedule.
- Sufficient budget must be provided for a testing environment.

THE END