Chapter 3, Project Organization and Communication
Lecture Outline

• Project Definition
• Project Organization
• Roles
• Tasks & Activities
• Work Product & Deliverables

• Focus of this lecture
  • Understand project management concepts from the developer’s perspective
How it should go

Requirements Analysis

Design

Implementation

System Testing

Delivery and Installation
How it often goes

Requirements Analysis

D

E

L

A

Y

Bananaware ripens with the Customer

Vaporware
Project Definition

• A **project** is an undertaking, limited in time, to achieve a set of goals that require a concerted effort

• **A project includes**
  • A set of deliverables to a client
  • A schedule
  • Technical and managerial activities required to produce and deliver the deliverables
  • Resources consumed by the activities (people, budget)

• Focus of **project management**
  • Administer the resources
  • Maintain accountability
  • React to change
  • Make sure, the goals are met.
Simple Object Model of a Project

- Deliverables
- Schedule
- Activity
- Resource
Laws of (Software) Project Management

• Projects progress quickly until they are 90% complete
  • Then they remain at 90% complete forever
• If project content is allowed to change freely, the rate of change will exceed the rate of progress
• Project teams detest progress reporting because it manifests their lack of progress
• Murphy’s law:
  • “When things are going well, something will go wrong”
  • “When things just can’t get worse, they will”
  • “When things appear to be going better, you have overlooked something.”
Refinement of the Model

Diagram: An entity-relationship diagram showing the relationships between various software engineering concepts. The diagram includes entities such as Project, Resource, Schedule, Work Breakdown Structure, Work Package, Organizational Unit, Role, Participant, Staff, Department, Team, Set of Work Products, Work Product, Internal Work Product, Project Deliverable, Outcome, Activity, Task, and Project Function. The diagram highlights the relationships and dependencies among these entities, such as production and consumption roles, and describes the dependencies and responsibilities within the software engineering model.
Dynamic Model of a Project

- **Definition**
  - do/Define Scope

- **Start**
  - do/Assign Tasks

- **Steady State**
  - do/Develop System

- **Termination**
  - do/Deliver System

- **Scope Defined**

- **Tasks Assigned**

- **System Done**
Project Organization

- A project organization defines the relationships among resources, in particular the participants, in a project

- A project organization should define
  - Who decides (decision structure)
  - Who reports their status to whom (reporting structure)
  - Who communicates with whom (communication structure)
Example of a Communication Structure

```
Management : Team
  + communicateDecision()
  + communicateStatus()

UserInterface : Team
  + communicateStatus()

Database : Team
  + communicateDecision()
  + communicateStatus()

Control : Team
```

UserInterface, Database, and Control: Team communicate with Management: Team.
Reporting vs. Communication

- Reporting supports project management in tracking project status
  - What work has been completed?
  - What work is behind schedule?
  - What issues threaten project progress?

- Reporting along the hierarchy is not sufficient when two teams need to communicate
  - A communication structure is needed
  - A participant from each team is responsible for facilitating communication between both teams
  - Such participants are called liaison
Hierarchical Project Organization

Basis of organization:
Complicated information and control flow across hierarchical boundaries

A wants to talk to B: Information Flow
A wants to make sure B does a certain change: Controlflow
Peer-To-Peer Communication

Basis of organization:
Nonlinear information flow across dynamically formed units

A wants to talk to B: Simple Information Flow

A wants to make sure B does a certain change: Simple Controlflow
Role

- A role defines a set responsibilities ("to-dos")
- Examples
- **Role: Tester**
  - Write tests
  - Report failures
  - Check if bug fixes address a specific failure
- **Role: System architect**
  - Ensure consistency in design decisions and define subsystem interfaces
  - Formulate system integration strategy
Roles

• Each member may assume multiple roles
• Role types
  • Management roles
    • Project manager, team leader ...
  • Development roles
    • System architect, object designer, implementor (development engineer), tester ...
  • Cross-functional roles
    • API engineer, document editor, configuration manager, tester ...
• Consultant roles
  • Client, end-user, application domain specialist, solution domain specialist ...
Responsibilities are assigned to Roles, Roles are assigned to People

“To Do” List for the Project

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8
- Item 9
Possible Mappings of Roles to Participants

- **One-to-One**
  - Ideal but rare

- **Many-to-Few**
  - Each project member assumes several "hats"
  - Danger of over-commitment
  - Need for load balancing

- **Many-to-"Too-Many"**
  - Some people don't have significant roles
  - Lack of accountability
  - Loosing touch with project
Task

- A **task** describes the smallest amount of work tracked by management
- Typically 3-10 working days effort

- Tasks descriptions
  - Role
  - Work product
  - Start date
  - Planned duration
  - Required resources.
Example: Tasks for building a House

- Survey
- Excavate
- Buy Materials
- Lay Foundation
- Build Outside Wall
  - Install Interior Plumbing
  - Install Interior Electrical
  - Install Wallboard
  - Paint Interior
  - Install Flooring
  - Install Roofing
  - Paint Exterior
  - Install Exterior Doors
  - Install Exterior Plumbing
  - Install Exterior Electrical
  - Install Exterior Siding
- Request Permits
Survey, Excavate
Example: Tasks for building a house

START → Survey → Excavate → Buy Material → Lay Foundation → Build Outside Wall → Install Interior Plumbing → Install Interior Electrical → Install Wallboard → Paint Interior → Install Flooring → Install Interior Doors → Install Roofing → Paint Exterior → Install Exterior Doors → Install Exterior Plumbing → Install Exterior Electrical → Install Exterior Siding → FINISH
Tasks and Work Packages

• A task is specified by a work package
  • Description of work to be done
  • Preconditions for starting, duration, required resources
  • Work products to be produced, acceptance criteria for it
  • Risks involved

• A task must have completion criteria
  • Includes the acceptance criteria for the work products (deliverables) produced by the task.
Work Products

- A work product is a visible outcome of a task
- Examples
  - A document
  - A review of a document
  - A presentation
  - A piece of code
  - A test report
- Work products delivered to the customer are called **deliverables**
Task Sizes

• Tasks are decomposed into sizes that allow monitoring
  • You may not know how to decompose the problem into tasks at first
  • Depends on the nature of work and how well task is understood.

• Finding the appropriate size is crucial
  • To-do lists from previous projects
  • Each software development activity identifies more tasks and modifies existing ones.
Activities

• Major unit of work
• Culminates in a major project milestone:
  • Scheduled event used to measure progress
  • Internal checkpoints should not be externally visible
  • A project milestone usually produces a baseline
• Activities are often grouped again into higher-level activities with different names:
  • Phase 1, Phase 2 ...
  • Step 1, Step 2 ...
• Allows separation of concerns
• Precedence relations can exist among activities
  • Example: “A1 must be executed before A2”
Example: Activities for Building a House

START

Survey → Excavate → Buy Material → Lay Foundation

Request Permits

Build Outside Wall

Install Interior Plumbing → Install Interior Electrical → Install Wallboard

Install Roofing

Paint Interior

Install Interior Doors

Install Exterior Doors

Paint Exterior

Install Exterior Plumbing → Install Exterior Electrical → Install Exterior Siding

Install Flooring

Finish
Example: Activities for Building a House

Establish Foundation

Request Permits

Build Outside Wall

Finish Interior

Install Plumbing

Install Electrical

Install Wallboard

Paint Interior

Finish Interior

Paint Exterior

Install Exterior Plumbing

Install Exterior Electrical

Install Exterior Siding

Install Exterior Roofing

Install Exterior Doors

Install Interior Doors

Install Flooring

Install Roofing

Paint Exterior

Finish Exterior

Install Exterior

Finish Exterior

Establish Foundation

Request Permits

Build Outside Wall

Finish Interior
Examples of Software Engineering Activities

- Planning
- Requirements Elicitation
- Analysis
- System Design
- Object Design
- Implementation
- Testing
- Delivery
Associations between Tasks, Activities, Roles, Work Products, and Work Packages

- **Work Package** describes **Unit Of Work**
- **Unit Of Work** results in **Work Product**
- **Activity** has 1 **Unit Of Work**
- **Task** has many **Unit Of Work**
- **Role** has 1 **Unit Of Work**
Schedule

• Mapping of tasks onto time with dependencies specified
Schedule – Gannt chart
Schedule – PERT chart

Storage subsystem system analysis
1  5d
Nov 13  Nov 19

Storage subsystem object design
2  5d
Nov 20  Nov 26

Storage subsystem test plan
5  10d
Nov 27  Dec 10

Storage subsystem implementation
3  15d
Nov 27  Dec 17
Summary

- Projects are concerted efforts towards a goal that take place within a limited time.
- Project participants are organized in terms of teams, roles, control relationships, and communication relationships.
- An individual can fill more than one role.
- Work is organized in terms of tasks assigned to roles and producing work products.
Outline

• Concepts and terminology
• Communication events
  • Planned communication
  • Unplanned communication
• Communication mechanisms
  • Synchronous communication
  • Asynchronous communication
• Communication activities
A Communication Example

From an Airplane Crash report:

"Two missile electrical boxes manufactured by different contractors were joined together by a pair of wires."

![Diagram of communication example](image)
A Communication Example (continued)

Thanks to a particular thorough preflight check, it was discovered that the wires had been reversed."
After the Crash...

... "The postflight analysis revealed that the contractors had indeed corrected the reversed wires as instructed."
“In fact, both of them had.”
Communication is critical

• In large system development efforts, you will spend more time communicating than coding
• A software engineer needs to learn the so-called soft skills:
  • Collaboration
    • Negotiate requirements with the client and with members from your team and other teams
  • Presentation
    • Present a major part of the system during a review
  • Management
    • Facilitate a team meeting
  • Technical writing
    • Write part of the project documentation.
Communication Event vs. Mechanism

**Communication event**
- Information exchange with defined objectives and scope
- *Scheduled*: Planned communication
  - Examples: weekly team meeting, review
- *Unscheduled*: Event-driven communication
  - Examples: problem report, request for change, clarification

**Communication mechanism**
- Tool or procedure that can be used to transmit information
- *Synchronous*: Sender and receiver are communicating at the same time
- *Asynchronous*: Sender and receiver are not communicating at the same time.
Modeling Communication

Communication Event is supported by Communication Mechanism

- Planned Event
- Unplanned Event
- Synchronous Mechanism
- Asynchronous Mechanism
Planned Communication Events

Problem Definition
• Objective: Present goals, requirements and constraints
• Example: Client presentation
• Usually scheduled at the beginning of a project

Project Review: Focus on system models
• Objective: Assess status and review the system model
• Examples: Analysis review, system design review
• Scheduled around project milestones and deliverables

Client Review: Focus on requirements
• Objective: Brief the client, agree on requirements changes
• The first client review is usually scheduled after analysis phase.
Planned Communication Events (cont’d)

**Walkthrough** (Informal)
- Objective: Increase quality of subsystem
- Example
  - Developer informally presents subsystem to team members (“peer-to-peer”)
- Scheduled by each team

**Inspection** (Formal)
- Objective: Compliance with requirements
- Example
  - Demonstration of final system to customer (Client acceptance test)
- Scheduled by project management
Planned Communication Events (cont’d)

Status Review

- Objective: Find deviations from schedule and correct them or identify new issues
- Example
  - Status section in regular weekly team meeting

Brainstorming

- Objective: Generate and evaluate large number of solutions for a problem
- Example
  - Discussion section in regular weekly team meeting.
Planned Communication Events (cont’d)

Release

- Objective: Baseline the result of each software development activity
- Examples:
  - Software Project Management Plan
  - Requirements Analysis Document
  - System Design Document
  - Beta version of software
  - Final version of software
  - User Manual
- Usually scheduled after corresponding activity (“phase”)

Postmortem Review

- Objective: Describe Lessons Learned
- Scheduled at the end of the project
Unplanned Communication Events

Request for clarification

• The bulk of communication among developers, clients and users
• Example: A developer may request a clarification about an ambiguous sentence in the problem statement.

From: Alice
Newsgroups: vso.discuss
Subject: SDD
Date: Wed, 2 Nov 9:32:48 -0400

When exactly would you like the System Design Document? There is some confusion over the actual deadline: the schedule claims it to be October 22, while the template says we have until November 7.

Thanks, -Alice
Unplanned Communication Events

Request for change

- A participant reports a problem and proposes a solution
- Change requests are often formalized when the project size is substantial
- Example: Request for additional functionality

Report number: 1291    Date: 5/3    Author: Dave
Synopsis: The STARS form should have a galaxy field.
Subsystem: Universe classification
Version: 3.4.1
Classification: missing functionality
Severity: severe
Proposed solution: …
Unplanned Communication Events

**Issue resolution**

- Selects a single solution to a problem for which several solutions have been proposed
- Uses issue base to collect problems and proposals.
Synchronous Communication Mechanisms

- If they require both sender and receiver to be available at the same time
  - Smoke signals
  - Hallway conversation
    - Supports: Unplanned conversations, Request for clarification, request for change
      + Cheap and effective for resolving simple problems
        – Information loss, misunderstandings are frequent
  - Meeting (face-to-face, phone, video conference)
    - Supports: Planned conversations, client review, project review, status review, brainstorming, issue resolution
      + Effective for issue resolution and consensus building
        – High cost (people, resources), low bandwidth.
# Meetings

<table>
<thead>
<tr>
<th>Header information identifying the meeting and audience</th>
<th>When and Where</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 1/30</td>
<td>Date: 1/30</td>
<td></td>
</tr>
<tr>
<td>Start: 4:30 P.M.</td>
<td>Start: 4:30</td>
<td></td>
</tr>
<tr>
<td>End: 5:30 P.M.</td>
<td>End: 5:30</td>
<td></td>
</tr>
<tr>
<td>Room: WH, 3420</td>
<td>Room: WH, 3420</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Desired outcome of the meeting</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Objective</td>
<td></td>
</tr>
<tr>
<td>Resolve any requirements issues that prevent us from starting prototyping.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action items to be reported on</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Status [Allocated Time: 15 minutes]</td>
<td></td>
</tr>
<tr>
<td>Dave: State of command parsing code</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issues scheduled to be discussed (and resolved) during the meeting</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Discussion items [Allocated Time: 35 minutes]</td>
<td></td>
</tr>
<tr>
<td>3.1 How to deal with arbitrarily formatted input data sets?</td>
<td></td>
</tr>
<tr>
<td>3.2 How to deal with output data?</td>
<td></td>
</tr>
<tr>
<td>3.3 Command parsing code (modifiability, backward compatibility)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The wrap-up period is the same for all meetings</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Wrap up [Allocated Time: 5 minutes]</td>
<td></td>
</tr>
<tr>
<td>4.1 Review and assign new action items</td>
<td></td>
</tr>
<tr>
<td>4.2 Meeting critique</td>
<td></td>
</tr>
</tbody>
</table>
Meetings

Open-ended meetings take more time than necessary.

This objective is difficult to achieve and cannot be verified.

Lack of context: what were Dave’s action items?

Lack of content: what are the current issues in each of these activities?

<table>
<thead>
<tr>
<th>When and Where</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date:</strong> 1/30</td>
<td><strong>Primary Facilitator:</strong> Peter</td>
</tr>
<tr>
<td><strong>Start:</strong> 4:30 P.M.</td>
<td><strong>Timekeeper:</strong> Dave</td>
</tr>
<tr>
<td><strong>End:</strong> open</td>
<td><strong>Minute Taker:</strong> Ed</td>
</tr>
<tr>
<td><strong>Room:</strong> WH 3420</td>
<td></td>
</tr>
</tbody>
</table>

1. **Objective**
   Resolve open issues

2. **Status [Allocated Time: 15 minutes]**
   Dave: Dave’s action items

3. **Discussion items [Allocated Time: 35 minutes]**
   3.1 Requirements issues
   3.2 Design issues
   3.3 Implementation issues

4. **Wrap up [Allocated Time: 5 minutes]**
   4.1 Review and assign new action items
   4.2 Meeting critique
Meetings

**Header information identifying the meeting and audience**
- **When and Where**
  - Date: 1/30
  - Start: 4:30 P.M.
  - End: 6:00 P.M.
  - Room: WH 3420
- **Role**
  - Primary Facilitator: Peter
  - Timekeeper: Dave
  - Minute Taker: Ed
  - Attending: Ed, Dave, Mary, Peter, Alice

**Verbatim from agenda**
1. Objective
   ... 

**Summary of the information that was exchanged**
2. Status
   ... 

**Record of issue discussion and resolution**
3. Discussion
   3.1 Command parsing code is a 1200–1300 line if statement. This makes it fairly hard to add new commands or to modify existing commands without breaking backward compatibility with existing clients.
   Proposals: 1) Restructure the command parsing code by assigning one object per kind of command. 2) Pass all command arguments by name. The latter would make it easier to maintain backward compatibility. On the other hand, this would increase the size of the commands, thus increasing the size of the command file.
   Resolution: Restructure code for now. Revisit this issue if backward compatibility is really an issue (the calling code might be rewritten anyway). See Al[1].
   ...

   Discussion of the other issues omitted for brevity

**Additions and modifications to the task plan**
4. Wrap up
   Revisit command parsing code. Emphasis on modularity. Coordinate with Bill from the database group (who might assume backward compatibility).
   ...

   Other action items and meeting critique omitted for brevity
   ...
Meetings

Yesterday, in our four-hour meeting, we agreed to postpone the vendor selection.

No, we agreed to use our existing vendor.

I thought we agreed to cancel the whole project.

We might need some clarity on this.

Four more hours should do it.
Meetings

LET'S HAVE A LITTLE PREMEETING TO PREPARE FOR THE MEETING TOMORROW.

WHOA! DO YOU THINK IT'S SAFE TO JUMP RIGHT INTO THE PREMEETING WITHOUT PLANNING IT?

OKAY, LET'S GET THIS PRELIMINARY PREMEETING MEETING GOING.

YOU THINK YOU'RE FUNNY, BUT YOU'RE NOT.
Asynchronous Communication Mechanisms

- **E-Mail**
  - Supports: Release, change request, brainstorming
  - Ideal for planned communication and announcements
  - E-mail taken out of context can be misunderstood, sent to the wrong person, or lost

- **Newsgroup**
  - Supports: Release, change request, brainstorming
  - Suited for discussion among people who share a common interest; cheap (shareware available)
  - Primitive access control (often, you are either in or out)

- **World Wide Web (Portal)**
  - Supports: Release, change request, inspections
  - Provide the user with a hypertext metaphor: Documents contain links to other documents.
  - Does not easily support rapidly evolving documents.
### Mechanisms for planned events

<table>
<thead>
<tr>
<th>Problem definition/ Brainstorm</th>
<th>Project/ Client Review</th>
<th>Status Review</th>
<th>Inspection/ Walkthrough</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hallway</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newsgroup</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WWW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Mechanisms for planned events

<table>
<thead>
<tr>
<th>Hallway</th>
<th>Meeting</th>
<th>Email</th>
<th>Newsgroup</th>
<th>WWW</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td><img src="image2.png" alt="Icon" /></td>
<td><img src="image3.png" alt="Icon" /></td>
<td><img src="image4.png" alt="Icon" /></td>
<td><img src="image5.png" alt="Icon" /></td>
</tr>
</tbody>
</table>

- **Problem definition/Brainstorm**
- **Project/Client Review**
- **Status Review**
- **Inspection/Walkthrough**
- **Release**
## Mechanisms for unplanned events

<table>
<thead>
<tr>
<th>Request for clarification</th>
<th>Change request</th>
<th>Issue resolution / Decision Making</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hallway</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newsgroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WWW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Mechanisms for unplanned events

<table>
<thead>
<tr>
<th></th>
<th>Request for clarification</th>
<th>Change request</th>
<th>Issue resolution / Decision Making</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hallway</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Email</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newsgroup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WWW</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Concepts and terminology
• Communication events
  • Planned communication
  • Unplanned communication
• Communication mechanisms
  • Synchronous communication
  • Asynchronous communication
• Communication activities
Typical Initial Communication Activities in a Software Project

• Understand problem statement
• Join a team
• Schedule and attend team status meetings
• Join the communication infrastructure.
Understand the Problem Statement

• The problem statement is developed by the client
  • Also called scope statement
• A problem statement describes
  • The current situation
  • The functionality the new system should support
  • The environment in which the system will be deployed
  • Deliverables expected by the client
  • Delivery dates
  • Criteria for acceptance test.
Join a Team

• During the project definition phase, the project manager forms a team for each subsystem
• Additional cross-functional teams are formed to support the subsystem teams
• Each team has a team leader
• The responsibilities of the team and the responsibilities each member must be defined to ensure the team success.
Attending Team Status Meetings

- Important part of a software project: The regular team meeting (weekly, daily,...)
- Meetings are often perceived as pure overhead
- Important task for the team leader:
  - Train the teams in meeting management
    - Announce agendas
    - Write minutes
    - Keep track of action items
  - Show value of status meeting
  - Show time-saving improvements.
Join the Communication Infrastructure

- A good communication infrastructure is the backbone of any software project
  - Web-Portal, e-mail, Newsgroups, Lotus Notes
- Learn to use the appropriate communication mechanism for the information at hand
  - The appropriateness of mechanisms may depend on the organizational culture.
- Register for each communication mechanism which is used by the software project
  - Get an account, get training
- Questions to ask:
  - Are meetings scheduled in a calendar?
  - Does the project have a problem reporting system?
  - Do team members provide peer reviews in meetings or in written form?
Summary

- **Communication Events**
  - Planned (stipulated by the schedule)
  - Unplanned (driven by unexpected events)

- **Communication Mechanisms**
  - Asynchronous communication mechanisms
  - Synchronous communication mechanisms

- **Important events and mechanisms in a software project**
  - Weekly meeting
  - Project reviews
  - Online communication mechanisms:
    - Discussion forum, email, web (Wiki)