ICT - 4231
OBJECT ORIENTED SOFTWARE ENGINEERING

LECTURE - 3

CHAPTER – 3 : PROJECT ORGANIZATION AND COMMUNICATION

Moinul Islam, IIT, JU

CONTENTS

TEXT BOOK

- Object-oriented Software Engineering: Using UML, Patterns, and Java; 3rd Edition; Bernd Bruegge, Allen H. Dutoit

INTRODUCTION

- Software engineering is a collaborative endeavor that involves participants from diverse backgrounds, including domain experts, analysts, designers, programmers, managers, technical writers, graphic designers, and users.
- Due to the complexity of systems, no single participant can fully comprehend or control all aspects, necessitating mutual dependency. Changes in the system or application domain require participants to update their understanding promptly, highlighting the critical need for accurate and timely information sharing.
- Communication in software engineering varies based on the supported activity. Participants convey status in regular meetings and document it in minutes. Project status is communicated to clients during reviews, while requirements and design use models and corresponding documents.
- Spontaneous exchanges address crises and misunderstandings through calls, messages, conversations, and ad hoc meetings.
- In larger projects, increased communication time can impact technical activities. To manage this, organizing projects into teams and utilizing formal and informal channels for information sharing is crucial.

INTRODUCTION (A ROCKET EXAMPLE)

- On June 4, 1996, 30 seconds after liftoff, the Ariane 501 rocket, the first prototype of the Ariane 5 series, exploded due to a software error.
- The main navigational computer experienced an arithmetic overflow, leading to a shutdown and a fatal loss of control. The backup computer had already shut down with the same exception.
- The incident resulted from inadequate testing of the alignment software, which had not been tested with an actual trajectory. The navigation system, reused from Ariane 4, had been flight-tested without failure in the previous model.
- The alignment calculations, critical for trajectory accuracy, were not properly communicated between the component and system teams, leading to the catastrophic failure.

AN OVERVIEW OF PROJECTS

System models are not the only information needed when communicating in a project. For example, developers need to know -

- Who is responsible for which part of the system?
- Which part of the system is due by when?
- Who should be contacted when a problem with a specific version of a component is discovered?
- How should a problem be documented?
- What are the quality criteria for evaluating the system?
- In which form should new requirements be communicated to developers?
- Who should be informed of new requirements?
- Who is responsible for talking to the client?

AN OVERVIEW OF PROJECTS (CONT.)

Although these questions can be relatively easy answered when all participants share a coffee break in the afternoon, the development of large software systems usually does not succeed with such an ad hoc approach. From a developer's perspective, a project consists of four components:

- **Work product:** This is any item produced by the project, such as a piece of code, a model, or a document. Work products produced for the client are called deliverables.
- **Schedule:** This specifies when work on the project should be accomplished.
- **Participant:** This is any person participating in a project. Sometimes we also call the participant project member.
- **Task:** This is the work to be performed by a project participant to create a work product.

AN OVERVIEW OF PROJECTS (CONT.)

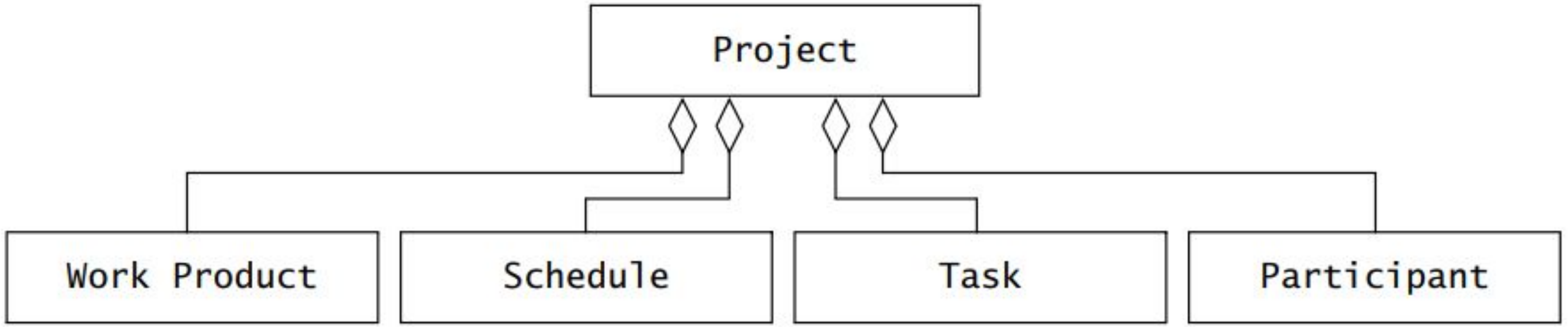


Figure 3-1 Model of a project (UML class diagram).

AN OVERVIEW OF PROJECTS (CONT.)

- **From a dynamic point of view, a project can be in any of several phases shown in Figure 3-2.**

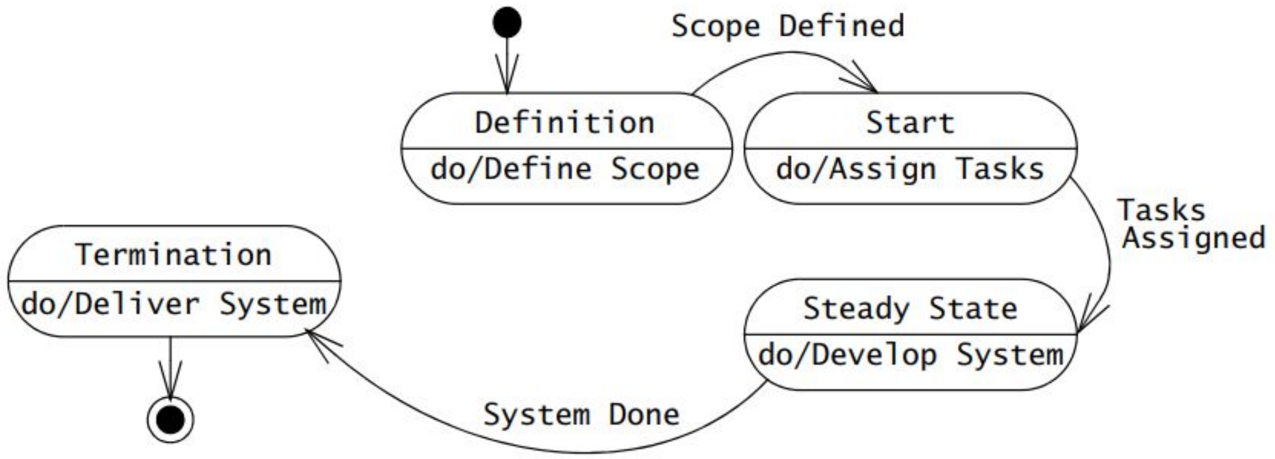


Figure 3-2 States in a software project (UML state machine diagram).

AN OVERVIEW OF PROJECTS (CONT.)

- **Project definition phase:** In this phase, the project manager, potential client, and key team member, the software architect, are involved. The focus is on understanding the software architecture, especially subsystem decomposition, and key project details like schedule and resource requirements. This phase produces three key documents: **the problem statement, the initial software architecture document,** and **the initial software project management plan.**
- **Project start phase:** the project manager sets up the project infrastructure, hires participants, organizes them in teams, defines major milestones, and kicks off the project. During the project definition and project start phases, most decisions are made by the project manager.

AN OVERVIEW OF PROJECTS (CONT.)

- **Project steady state phase:** The participants develop the system. They report to their team leader, who is responsible for tracking the status of the developers and identifying problems. The team leaders report the status of their team to the project manager, who then evaluates the status of the complete project. Team leaders respond to deviations from the plan by re-allocating tasks to developers or obtaining additional resources from the project manager. The project manager is responsible for the interaction with the client, obtaining formal agreement and renegotiating resources and deadlines.
- **Project termination phase:** the project outcome is delivered to the client and the project history is collected. Most of the developers' involvement with the project ends before this phase. A handful of key developers, the technical writers, and the team leaders are involved with wrapping up the system for installation and acceptance and collecting the project history for future use.

AN OVERVIEW OF PROJECTS (CONT.)

Communication within a project occurs through planned and unplanned events.

Planned communication includes:

- problem inspection
- status meetings
- peer reviews
- client and project reviews
- releases

Unplanned communication includes:

- requests for clarification
- requests for change
- issue resolution

Planned communication helps disseminate information that targeted participants are expected to use. Unplanned communication helps deal with crises and with unexpected information needs. All three communication needs must be addressed for project participants to communicate accurately and efficiently.

PROJECT ORGANIZATION CONCEPTS

Project organization concepts includes the following concepts:

- Project organization
- Roles
- Tasks and work Products
- Schedule

PROJECT ORGANIZATION

- An important part of any project organization is to define the relationships among participants and between them and tasks, schedule, and work products. In a team-based organization (Figure 3-3), the participants are grouped into teams, where a team is a small set of participants working on the same activity or task.
- We distinguish teams from other sets of people, in particular groups and committees.
- **A group**, for example, is a set of people who are assigned a common task, but they work individually without any need for communication to accomplish their part of the task.
- **A committee** is comprised of people who come together to review and critique issues and propose actions.

PROJECT ORGANIZATION

Project participants interact with each other. The three major types of interaction in a project are:

1. **Reporting:** This type of interaction is used for reporting status information. For example, a developer reports to another developer that an API (Application Programmer Interface) is ready, or a team leader reports to a project manager that an assigned task has not yet been completed.
2. **Decision:** This type of interaction is used for propagating decisions. For example, a team leader decides that a developer has to publish an API, a project manager decides that a planned delivery must be moved up in time. Another type of decision is the resolution of an issue.
3. **Communication:** This type of interaction is used for exchanging all the other types of information needed for decision or status. Communication can take many flavors. Examples are the exchange of requirements or design models or the creation of an argument to support a proposal. An invitation to eat lunch is also a communication.

PROJECT ORGANIZATION

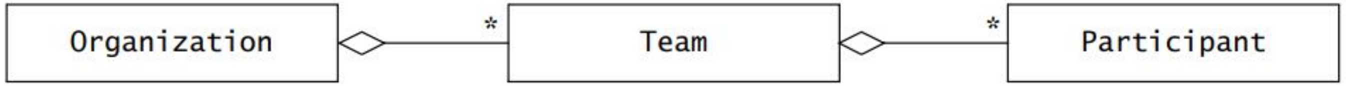


Figure 3-3 A team-based organization consists of organizational units called teams, which consist of participants or other teams (UML class diagram).

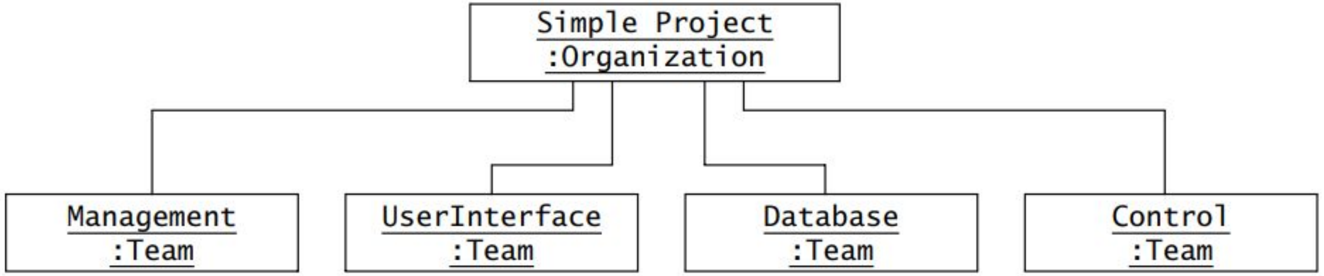


Figure 3-4 Example of a simple project organization (UML instance diagram). Reporting, deciding, and communicating are all made via the aggregation association of the organization.

PROJECT ORGANIZATION

- **Hierarchical Organization:** We call the organization hierarchical if both status and decision information are unidirectional; that is, decisions are always made at the root of the organization and passed via the interaction association to the leaves of the organization. Status in hierarchical organizations is generated at the leaves of the organization and reported to the root via the interaction association. The structure of the status and decision information flow is often called the reporting structure of the organization.

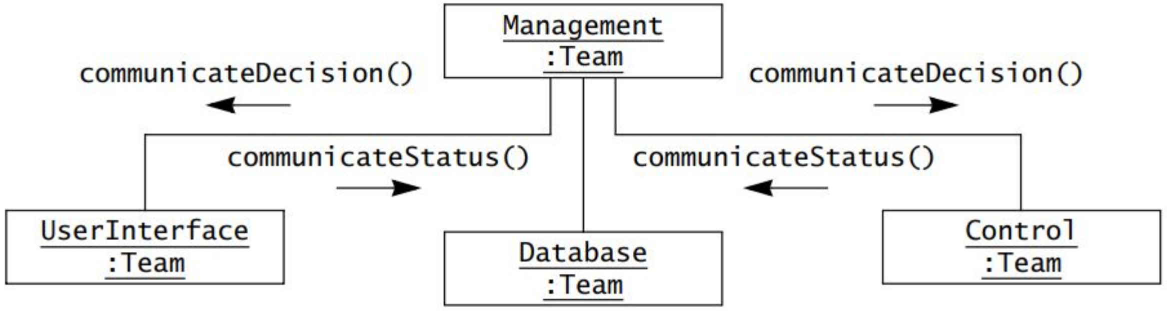


Figure 3-5 Example of reporting structure in a hierarchical organization (UML communication diagram). Status information is reported to the project manager, and corrective decisions are communicated back to the teams by the team leaders. The team leaders and the project manager are called the management team.

PROJECT ORGANIZATION

- **Liaison-based communication structure:** Effective in software development, this approach involves designated liaisons facilitating direct communication between teams, overcoming delays and information distortion often associated with the established reporting structure. Cross-functional teams play a key role, and team leaders extend their responsibility to ensure smooth information flow.

PROJECT ORGANIZATION

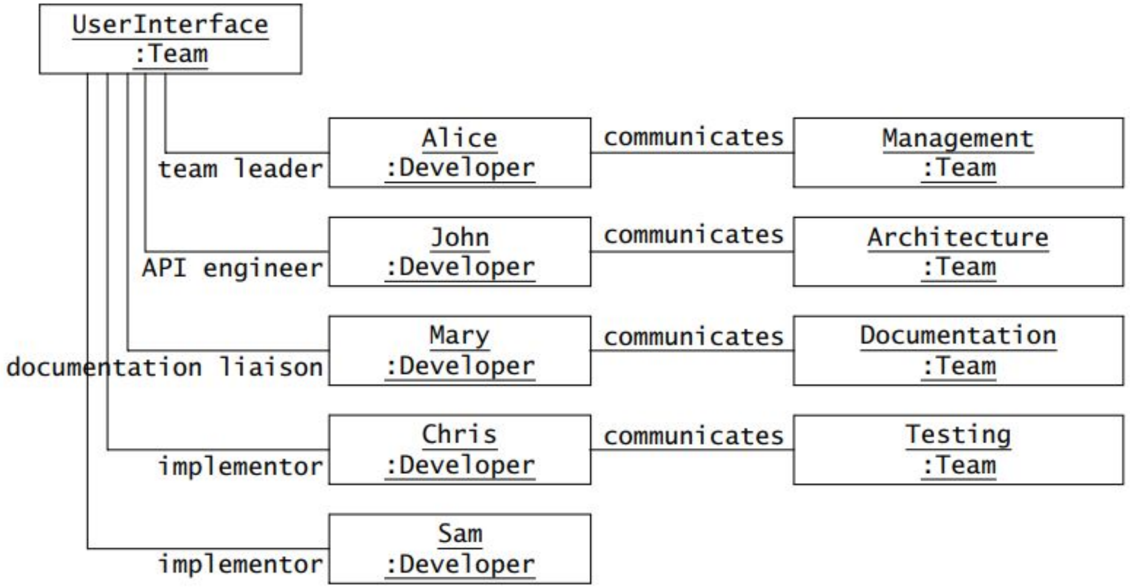


Figure 3-6 Examples of a liaison-based communication structure (UML object diagram). The team is composed of five developers. Alice is the team leader, also called the liaison to the management team. John is the API engineer, also called the liaison to the architecture team. Mary is the liaison to the documentation team. Chris and Sam are implementors and interact with other teams only informally.

ROLES

- A role defines the set of technical and managerial tasks that are expected from a participant or team. In a team-based organization, we assign tasks to a person or a team via a role.
- For example, the role of tester of a subsystem team consists of the tasks to define the test suites for the subsystem under development, for executing these tests, and for reporting discovered defects back to the developers.
- In a software project we distinguish between the following four types of roles:

1. management roles,
2. development roles,
3. cross-functional roles, and
4. consultant roles

ROLES

1. **Management roles** (e.g., project manager, team leader) are concerned with the organization and execution of the project within constraints..
2. **Development roles** are concerned with specifying, designing, and constructing subsystems. These roles include the analyst, the system architect, the object designer, the implementor, and the tester. Table 3-1 describes examples of development roles in a subsystem team.
3. **Cross-functional roles** are concerned with coordination among teams. Developers filling these roles are responsible for exchanging information relevant to other teams and negotiating interface details. The cross-functional role is also called liaison. The liaison is responsible for disseminating information along the communication structure from one team to another. There are four types of liaisons: **API engineer, document editor, configuration manager, tester.**
4. **Consultant roles** are concerned with providing temporary support in areas where the project participants lack expertise. The users and the client act in most projects as consultants on the application domain. Technical consultants may bring expertise on new technologies or methods. Non-technical consultants can help to address legal and marketing issues. We distinguish the following types of consultant roles: **client, end user, application domain specialist, solution domain specialist.**

ROLES

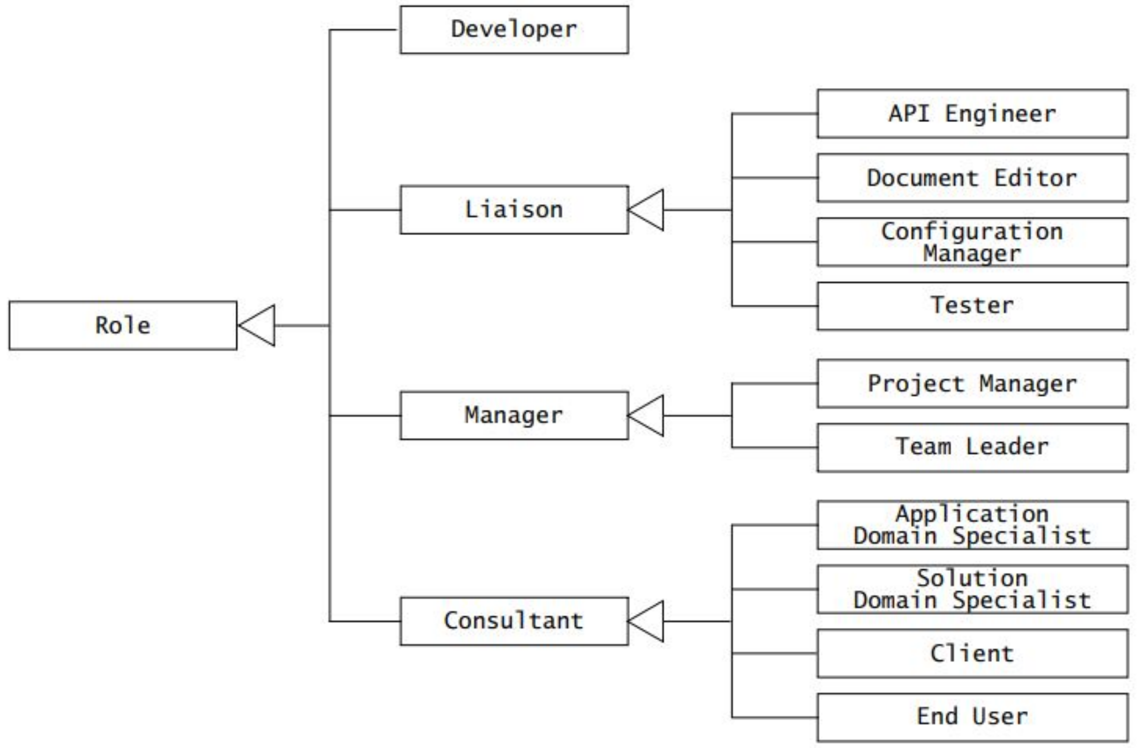


Figure 3-7 Types of roles found in a software engineering project (UML class diagram).

ROLES

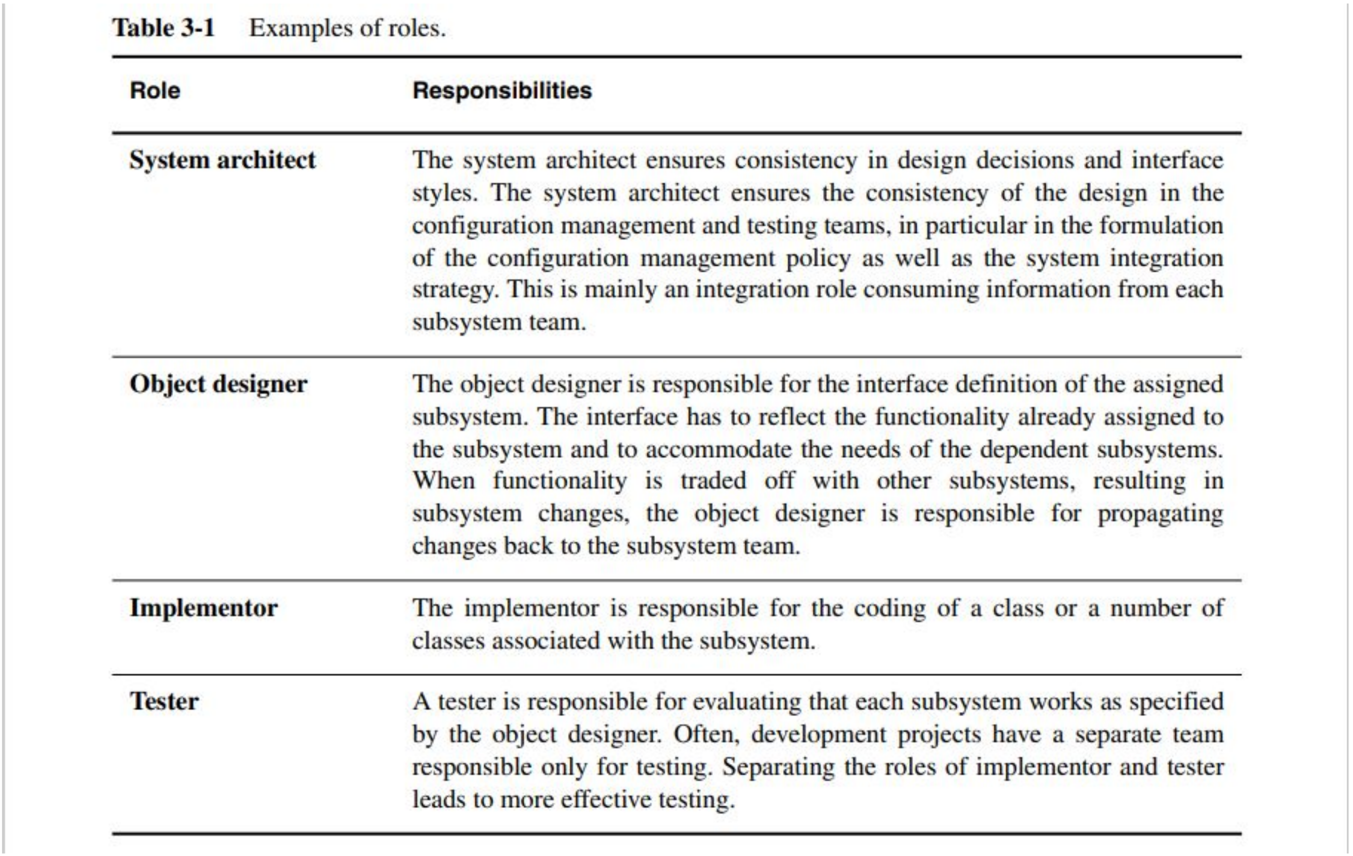


Table 3-1 Examples of roles.

Role	Responsibilities
System architect	The system architect ensures consistency in design decisions and interface styles. The system architect ensures the consistency of the design in the configuration management and testing teams, in particular in the formulation of the configuration management policy as well as the system integration strategy. This is mainly an integration role consuming information from each subsystem team.
Object designer	The object designer is responsible for the interface definition of the assigned subsystem. The interface has to reflect the functionality already assigned to the subsystem and to accommodate the needs of the dependent subsystems. When functionality is traded off with other subsystems, resulting in subsystem changes, the object designer is responsible for propagating changes back to the subsystem team.
Implementor	The implementor is responsible for the coding of a class or a number of classes associated with the subsystem.
Tester	A tester is responsible for evaluating that each subsystem works as specified by the object designer. Often, development projects have a separate team responsible only for testing. Separating the roles of implementor and tester leads to more effective testing.

TASKS AND WORK PRODUCTS

- A **task** is a well-defined work assignment for a role. Groups of related tasks are called activities. The project manager or team leader assigns a task to a role. The participant who is assigned the role carries out the task, and the manager monitors its progress and completion.
- A **work product** is a tangible item that results from a task. Examples of work products include an object model, a class diagram, a piece of source code, a document, or parts of documents. Work products result from tasks, are subject to deadlines, and feed into other tasks.
- Any work product to be delivered to the client is called a **deliverable**. The software system and the accompanying documentation usually constitute a set of deliverables.
- Work products that are not visible to the client are called **internal work products**.
- The specification of work to be accomplished in completing a task or activity is described in a **work package**. A work package includes the task name, the task description, resources needed to perform the task, dependencies on inputs (work products produced by other tasks) and outputs (work products produced by the task in question), as well as dependencies on other tasks.

TASKS AND WORK PRODUCTS

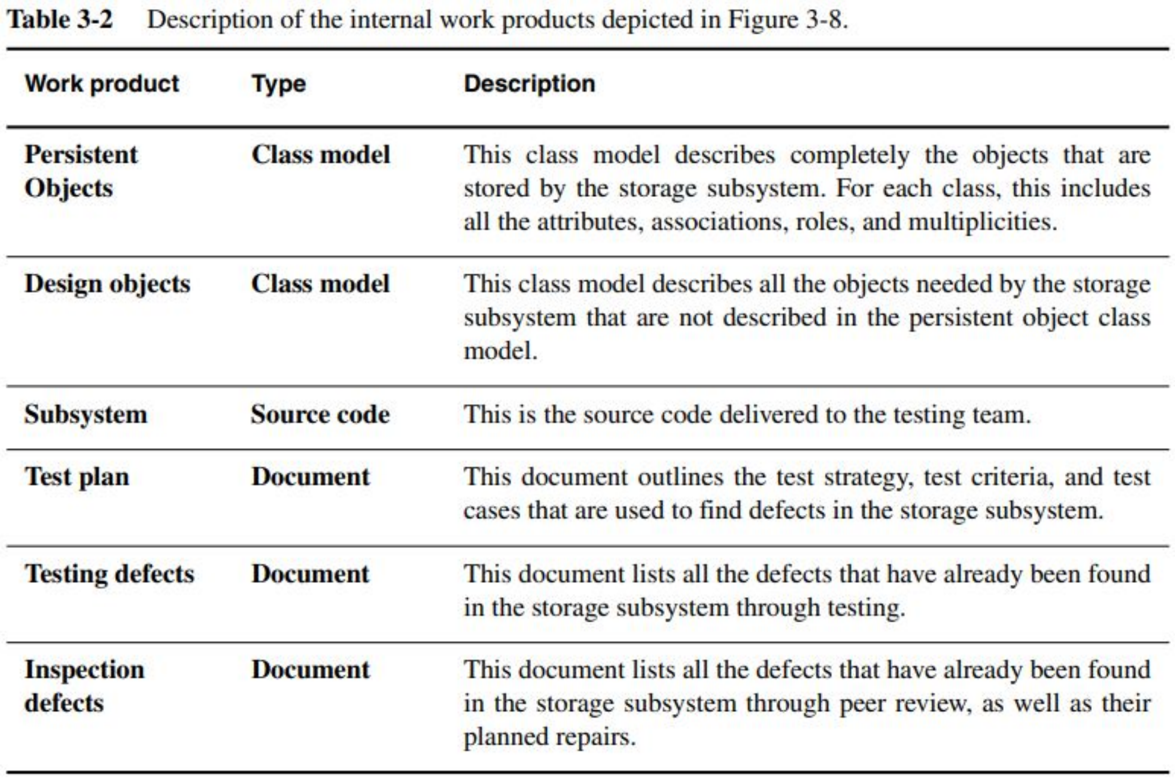


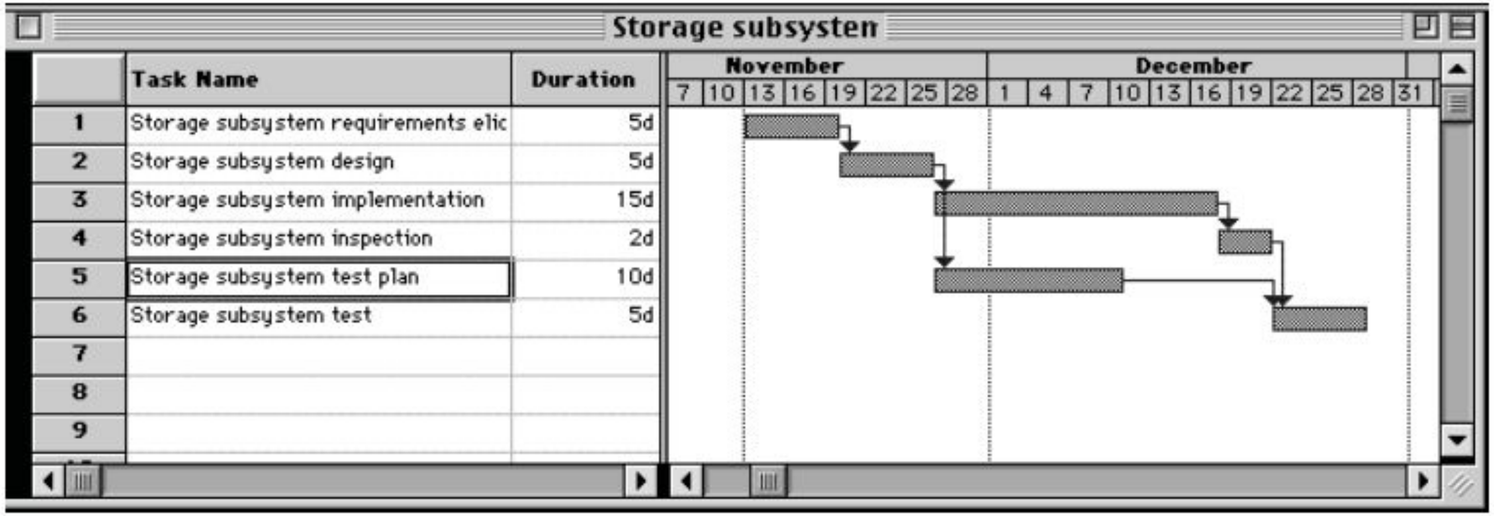
Table 3-2 Description of the internal work products depicted in Figure 3-8.

Work product	Type	Description
Persistent Objects	**Class model**	This class model describes completely the objects that are stored by the storage subsystem. For each class, this includes all the attributes, associations, roles, and multiplicities.
Design objects	**Class model**	This class model describes all the objects needed by the storage subsystem that are not described in the persistent object class model.
Subsystem	**Source code**	This is the source code delivered to the testing team.
Test plan	**Document**	This document outlines the test strategy, test criteria, and test cases that are used to find defects in the storage subsystem.
Testing defects	**Document**	This document lists all the defects that have already been found in the storage subsystem through testing.
Inspection defects	**Document**	This document lists all the defects that have already been found in the storage subsystem through peer review, as well as their planned repairs.

SCHEDULE

- A **schedule** is the mapping of tasks onto time: each task is assigned start and end times. This allows us to plan the deadlines for individual deliverables. The two most often used diagrammatic notations for schedules are **PERT** and **Gantt charts.**
- A **Gantt chart** is a compact way to present the schedule of a software project along the time axis. A Gantt chart is a bar graph on which the horizontal axis represents time and the vertical axis lists the different tasks to be done. Tasks are represented as bars whose length corresponds to the planned duration of the task.
- A **PERT chart** represents a schedule as an acyclic graph of tasks. The planned start and duration of the tasks are used to compute the critical path, which represents the shortest possible path through the graph. The length of the critical path corresponds to the shortest possible schedule, assuming sufficient resources to accomplish, in parallel, tasks that are independent. Moreover, tasks on the critical path are the most important, as a delay in any of these tasks will result in a delay in the overall project. The tasks and bars represented in thicker lines belong to the critical path.

SCHEDULE



	Task Name	Duration
1	Storage subsystem requirements elic	5d
2	Storage subsystem design	5d
3	Storage subsystem implementation	15d
4	Storage subsystem inspection	2d
5	Storage subsystem test plan	10d
6	Storage subsystem test	5d
7		
8		
9		

Figure 3-10 An example of schedule for the database subsystem (Gantt chart).

SCHEDULE

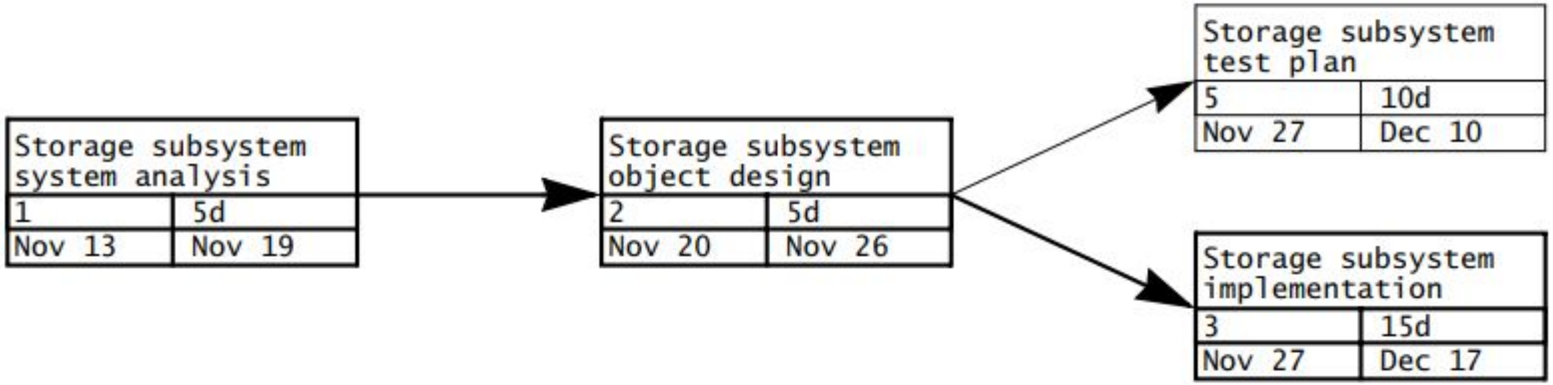

Storage subsystem system analysis
1
5d
Nov 13
Nov 19
Storage subsystem object design
2
5d
Nov 20
Nov 26
Storage subsystem test plan
5
10d
Nov 27
Dec 10
Storage subsystem implementation
3
15d
Nov 27
Dec 17

Figure 3-11 Schedule for the database subsystem (PERT chart). Thick lines denote the critical path.

ORGANIZATIONAL ACTIVITIES

The activities of a developer when joining a project organization and its communication infrastructure. The activities include

- Joining a Team
- Joining the Communication Infrastructure
- Attending Team Status Meetings
- Organizing Client and Project Reviews.

JOINING A TEAM

- During the project definition phase, the project manager identified a **subsystem team** for each subsystem in the initial decomposition of the software architecture.
- Additionally, **crossfunctional teams** (e.g., architecture team, integration team) are formed to support the subsystem teams.
- Each team has a team leader who was also already selected during the project definition phase. An important activity during the project start phase is now the assignments of participants to teams.
- Each subsystem team also has to nominate a liaison to the crossfunctional teams to facilitate information transfer among teams. Table 3-6 depicts an example of role assignment to participants for the database team of the OWL project.

JOINING A TEAM

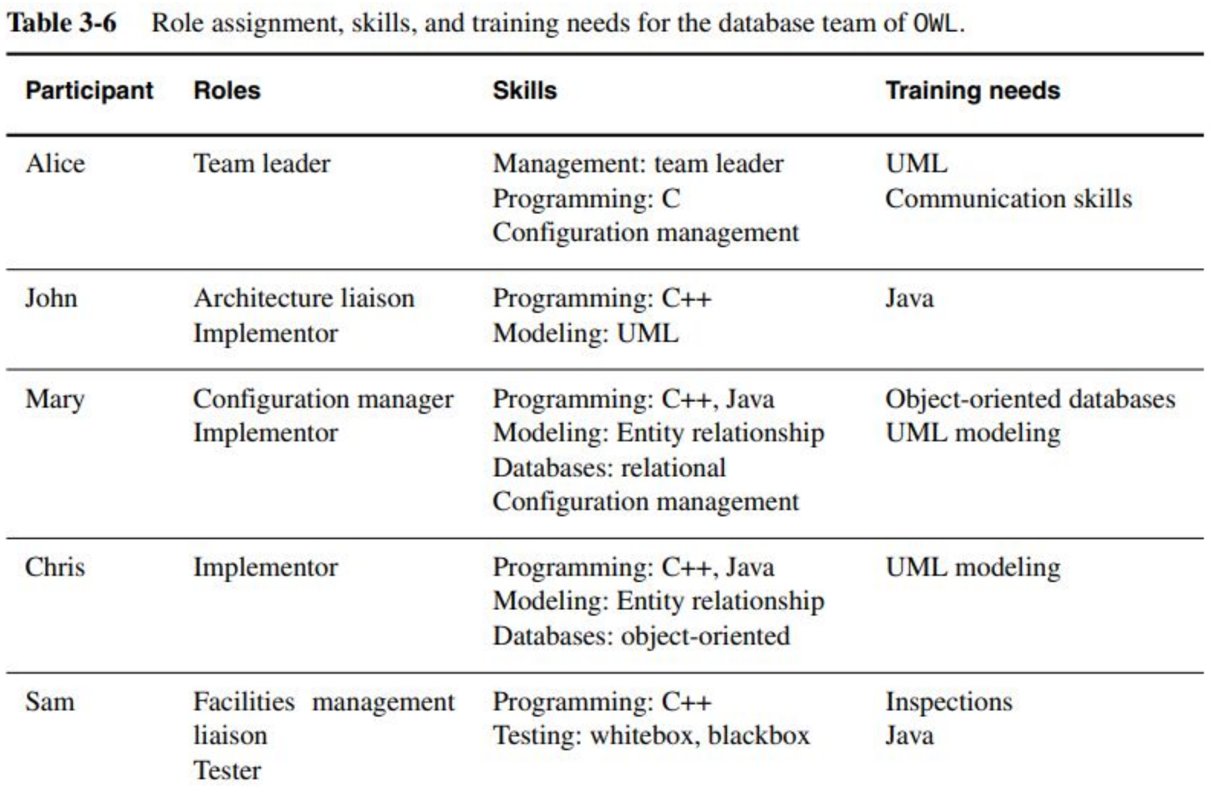


Table 3-6 Role assignment, skills, and training needs for the database team of OWL.

Participant	Roles	Skills	Training needs
Alice	Team leader	Management: team leader Programming: C Configuration management	UML Communication skills
John	Architecture liaison Implementor	Programming: C++ Modeling: UML	Java
Mary	Configuration manager Implementor	Programming: C++, Java Modeling: Entity relationship Databases: relational Configuration management	Object-oriented databases UML modeling
Chris	Implementor	Programming: C++, Java Modeling: Entity relationship Databases: object-oriented	UML modeling
Sam	Facilities management liaison Tester	Programming: C++ Testing: whitebox, blackbox	Inspections Java

JOINING THE COMMUNICATION INFRASTRUCTURE

Two sets of forums are created to support project and team communication, respectively. Members subscribe to all project forums and to their team's forums. Project forums include:

- **Announce.** Major events (e.g., review agendas, releases) are announced by management by posting to this forum.
- **Discuss**. Project-level requests for clarification and requests for change are posted in this forum.
- **Issues.** Open issues and their current state are posted in this forum. All project members can post to this forum and read its documents.
- **Documents.** The latest versions of the project deliverables (e.g., Requirements Analysis Document, System Design Document) and other internal project documents (e.g., Software Project Management Plan) are posted in this forum.
- **Equipment list.** This forum contains descriptions of equipment and its status (e.g., availability, current borrower). Only the equipment manager can post to this forum.

ATTENDING TEAM STATUS MEETINGS

- The weekly team meeting is a critical component of a software project, serving as a platform for status reviews, brainstorming, and issue resolution. The first meeting is particularly important as it introduces team members to formal meeting roles and procedures.
- During the first meeting, management takes the opportunity to explain meeting procedures and motivate team members. The goal is to train participants by example, encouraging discussion about procedures. The facilitator's role is emphasized, focusing on increasing meeting efficiency without imposing decisions. Team members are taught keyword phrases for standard situations to maintain focus.
- Role rotation is implemented by management to build redundant skills and enhance information sharing, despite the short-term drawback of participants not quickly maturing into roles. The long-term benefit is seen as a healthy investment in effective communication and collaboration.
- Teams are responsible for assigning meeting roles and posting them in their respective forums. The facilitator posts the initial draft of the agenda, and the minute taker posts minutes before the status meeting. Team members can comment on the agenda and minutes, promoting collaboration.
- Acknowledging that meeting roles and procedures may be perceived as overhead, management invests time at the project's start to illustrate their benefits. A systematic review of early meeting agendas and minutes is conducted, suggesting time-saving improvements for facilitators and minute takers. The focus is on capturing action items and unresolved issues in minutes for efficient communication.

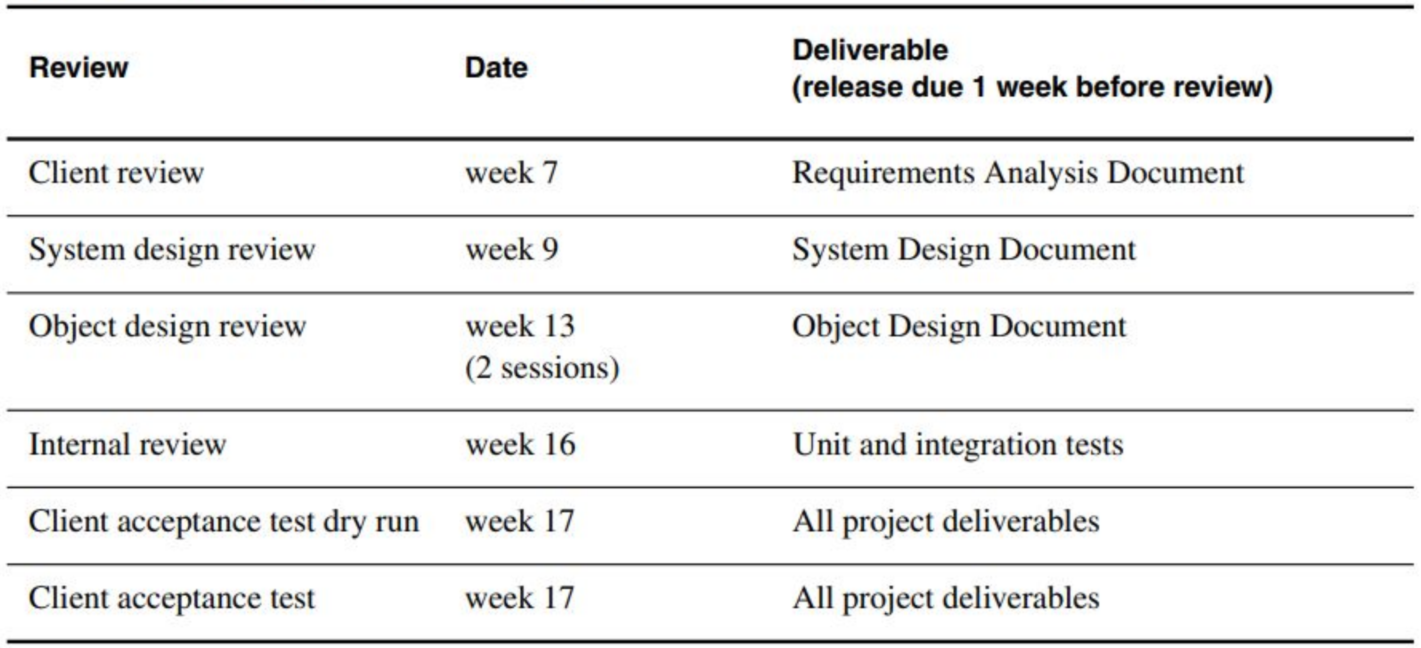
ORGANIZING CLIENT AND PROJECT REVIEWS

- **Client reviews** are conducted after the release of the requirements analysis document and after the delivery of the system.
- **Project reviews** are conducted to review the system design documents, the detailed object design, and the test. A project review may also be conducted before delivery as a dry run for the client acceptance test.
- Management also introduces procedures for organizing reviews:
  1. The deliverables being reviewed are released one week1 prior to the review.
  2. Shortly after the release, management publishes a draft agenda listing presentation topics for each team. The initial draft of the agenda is posted in the Announce forum.
  3. Candidate presenters reply to the original agendas and refine the presentation topic. Management modifies the agenda based on the replies.
  4. Presenters submit their slides by replying to the agenda and including the slides in the reply. The management collates the slides before the presentation and updates the agenda.

ORGANIZING CLIENT AND PROJECT REVIEWS

Project management schedules all reviews during the planning phase (Table 3-7).

Table 3-7 An example of a review schedule.



Review	Date	Deliverable (release due 1 week before review)
Client review	week 7	Requirements Analysis Document
System design review	week 9	System Design Document
Object design review	week 13 (2 sessions)	Object Design Document
Internal review	week 16	Unit and integration tests
Client acceptance test dry run	week 17	All project deliverables
Client acceptance test	week 17	All project deliverables

THANK YOU