**Chapter three** : **software project management**

**This chapter will discuss the following concepts:**

**3-1 Introduction**

**3-2 People**

**3-3 The Product**

**3-4 the process**

**3-5 the project**

3-1 Introduction

 Effective software project management focuses on the four P’s: people, product, process, and project. The order is not arbitrary.

Software project management is an essential part of software engineering. Projects need to be managed because professional software engineering is always subject to organizational budget and schedule constraints. The project manager’s job is to ensure that the software project meets and overcomes these constraints as well as delivering high-quality software. Good management cannot guarantee project success. However, bad management usually results in project failure: the software may be delivered late, cost more than originally estimated, or fail to meet the expectations of customers.

The success criteria for project management obviously vary from project to project but, for most projects, important goals are:

1. Deliver the software to the customer at the agreed time.

2. Keep overall costs within budget.

3. Deliver software that meets the customer’s expectations.

4. Maintain a happy and well-functioning development team.

**3-2 People**

 In fact, the “people factor” is so important that the Software Engineering Institute has developed a people management capability maturity model (PM-CMM), “to enhance the readiness of software organizations to undertake complex applications.

The software process (and every software project) is populated by people who can be categorized into one of five categories:

1. ***Senior managers***: who define the business issues that often have a significant influence on the project.

2. ***Project (technical) managers:*** who must plan, motivate, organize, and control the practitioners who do software work.

3. ***Practitioners:*** who deliver the technical skills that are necessary to engineer a product or application.

4. ***Customers:*** who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.

5. ***End users***: who interact with the software once it is released for production use.

The “best” team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the over-all problem difficulty.

There are three generic team organizations:

1-**Democratic Decentralized (DD)**: This software engineering team has no permanent leader. Rather, "task coordinators are appointed for short durations and then replaced by others who may coordinate different tasks." Decisions on problems and approach are made by group consensus. Communication among team members is horizontal.

2-**Controlled Decentralized (CD)**: This software engineering team has a deﬁned leader who coordinates speciﬁc tasks and secondary leaders that have responsibility for subtasks. Problem solving remains a group activity, but implementation of solutions is partitioned among subgroups by the team leader. Communication among subgroups and individuals is horizontal. Vertical communication along the control hierarchy also occurs.

3-**Controlled Centralized (CC):** Top-level problem solving and internal team coordination are managed by a team leader. Communication between the leader and team members is vertical.

**3-3 The Product**

"Product" refers to the estimation of the cost, time and effort required to produce the finished software product, before  a software project is planned, the  product  objectives  and  scope  should  be  established, technical  and  management   constraints  should   be  identified. After the processing, the final product should be built and should be delivered in time to the client.

 The ﬁrst software project management activity is the determination of software scope. Scope is deﬁned by answering the following questions:

1-***Context***: How does the software to be built ﬁt into a larger system, product, or business context and what constraints are imposed as a result of the context?

2-***Information objectives:*** What customer-visible data objects are produced as output from the software? What data objects are required for input?

3-***Function and performance***: What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

**3-4 the process**

The generic phases that characterize the software process—deﬁnition, development, and support—are applicable to all software. The problem is to select the process model that is appropriate for the software to be engineered by a project team. Wide array of software engineering paradigms were discussed:

• The linear sequential model

• The prototyping model

• The RAD model

• The incremental model

• The spiral model

• The WINWIN spiral model

• The component-based development model

• The concurrent development model

• The formal methods model

• The fourth generation techniques model

The project manager must decide which process model is most appropriate for:

1. The customers who have requested the product and the people who will do the work.
2. The characteristics of the product itself.
3. The project environment in which the software team works.

**3-5 the project**

In order to manage a successful software project, we must understand what can go wrong (so that problems can be avoided), understand the critical success factors that lead to good project management, and develop a commonsense approach for planning, monitoring and controlling the project.

There are ten signs that indicate that an information systems project is in risk:

1. Software people don’t understand their customer’s needs.

2. The product scope is poorly deﬁned.

3. Changes are managed poorly.

4. The chosen technology changes.

5. Business needs change [or are ill-deﬁned].

6. Deadlines are unrealistic.

7. Users are resistant.

8. Sponsorship is lost [or was never properly obtained].

9. The project team lacks people with appropriate skills.

10. Managers [and practitioners] avoid best practices and lessons learned.