Software Engineering Fundamental Revision

CHAPTER ONE

- Definition of software
- Software Engineering vs. Computer Science
- Failure curve of Hardware and software
- Software Myths
- Software development life cycle (SDLC)
- Software Development methods

Metaphor for Software Development

- **Software** (set of instruction + design diagrams+ necessary databases)
- **Software Engineering** : Creation and design of the software .
- **Computer Science**: Broad approach to the study of the principles and use of computers that covers both theory and application

Characteristics of Software

software has characteristics that are considerably different than those of hardware:

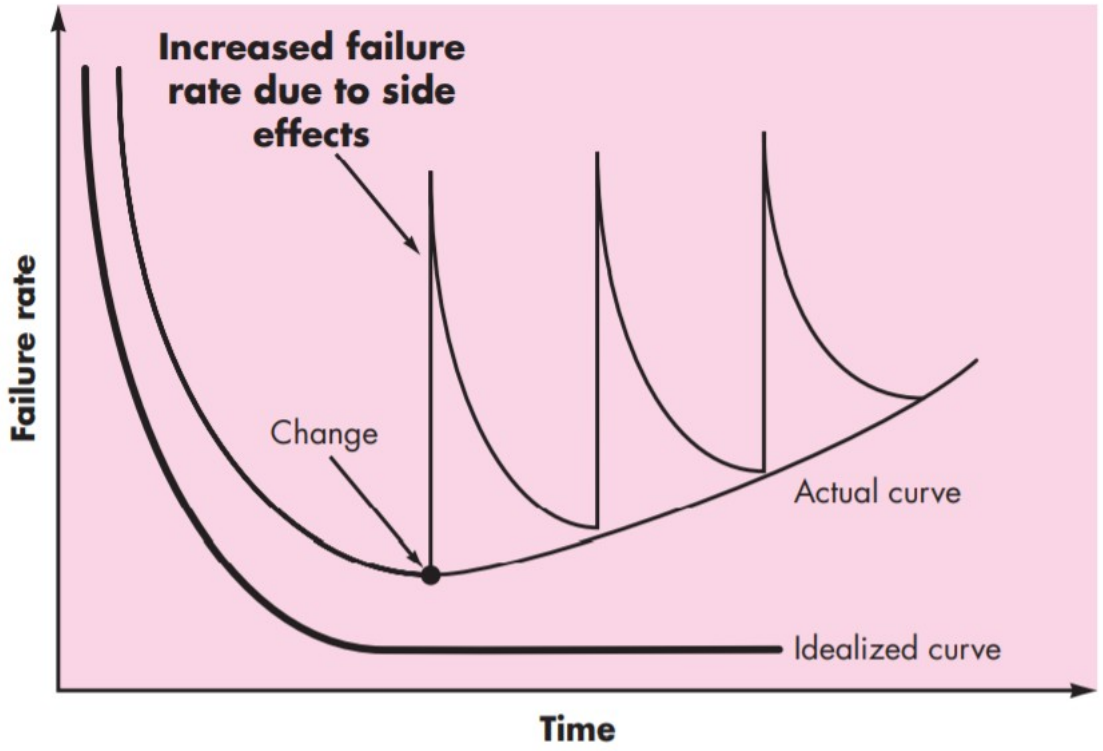
1. Software is developed ; it is not manufactured (*Both process focuses to achieve high quality, for hardware can introduce quality problems that are nonexistent for a software*)
2. Software doesn't "wear out."

3. Software is custom build

Characteristics of software

FIGURE 1.2

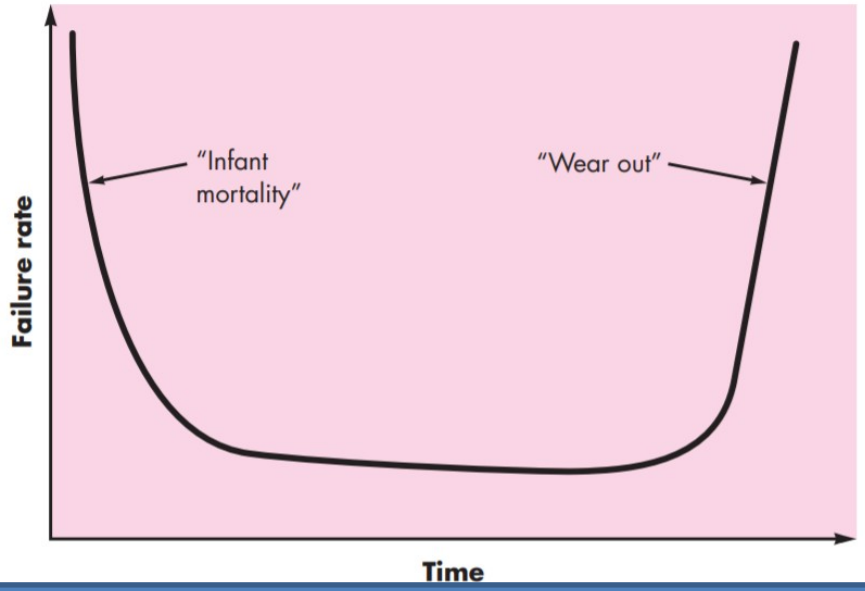
Failure curves for software



2. Software doesn't wear out but it does deteriorates

Characteristics of Software

FIGURE 1.1
Failure curve for hardware



2. Software doesn't wear out but hardware does

Software Myths

Management myths

Myth: ***We already have a book that's full of standards and procedures for building software***. Won't that provide my people with everything they need to know?

Reality: Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable?

Software Myths

Management Myths

Myth :If we get **behind schedule**, we can add more programmers and catch up

Reality :new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort

Software Myths

Management Myth

Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

Software Myths

Customer myths

Myth: A **general statement** of objectives is sufficient to begin writing programs—we can fill in the details later.

Reality: an ambiguous "statement of objectives" is a recipe for disaster.

Software Myths

Customer

Myth: Software **requirements continually change**, but change can be easily accommodated because software is flexible.

Reality :When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly

Software Myths

Practitioner's myths.

Myth: Once we write the program and get it to work, our job is done

Reality: Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time

Software Myths

Practitioner's myths

Myth: Until I get the program "running" I have no way of assessing its quality

Reality: Quality focuses from at the time of requirement analysis

Software Myths

Myth: The only **deliverable work product** for a successful project is the working program

Reality: *A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software support*.

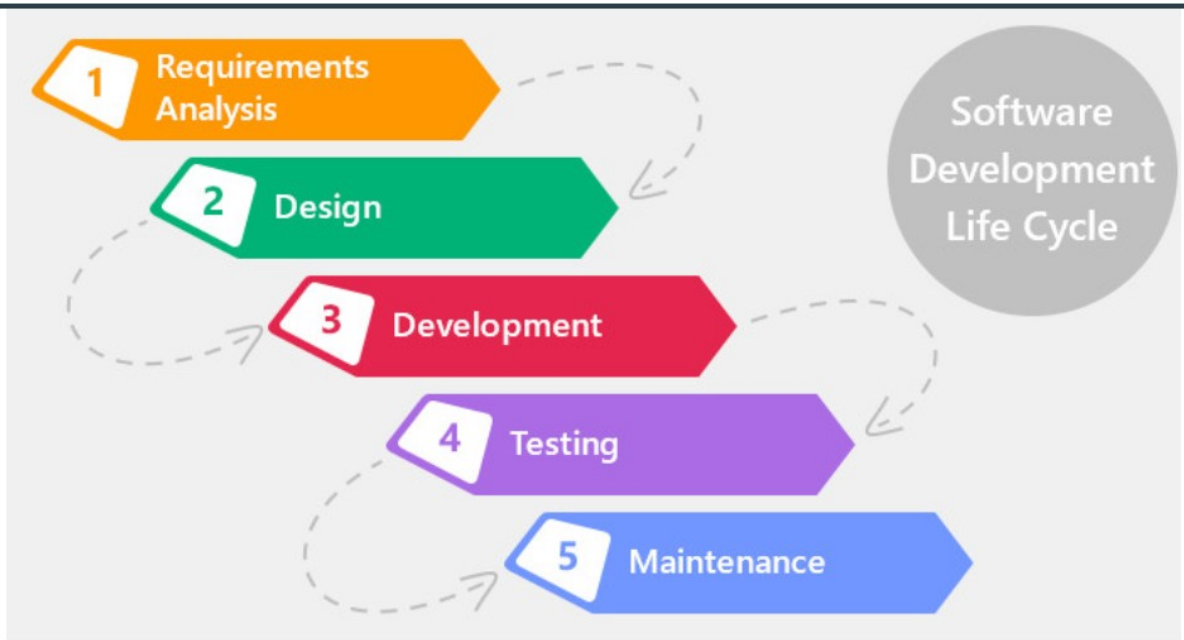
Software Myths

Practitioners Myths

Myth: Software engineering will make us **create voluminous and unnecessary** documentation and will invariably slow us down.

- Reality: Software engineering is not about creating documents. It is **about creating a quality product**. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

Software Development Life Cycle (SDLC)



What are the Software Development Life Cycle (SDLC) phases?

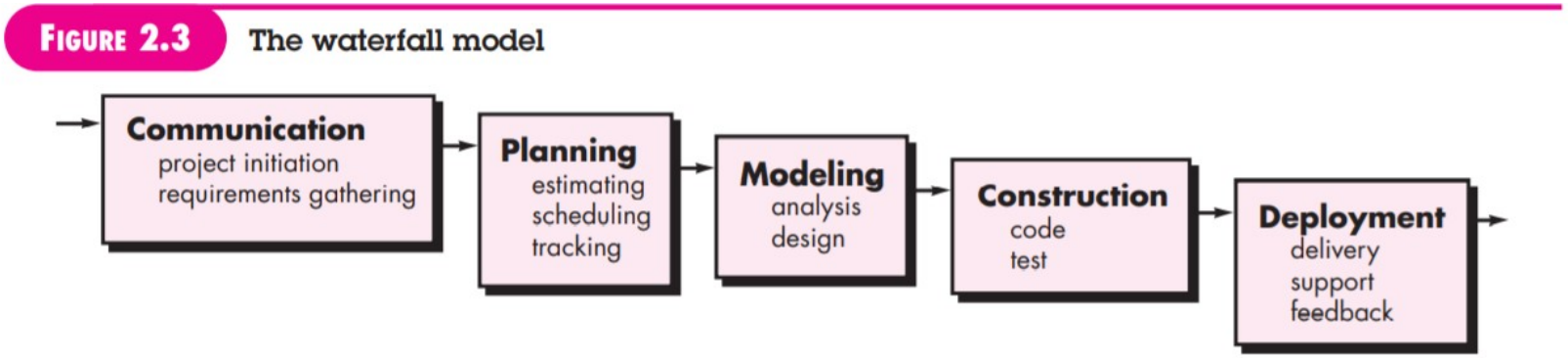
Published on August 23, 2017

Process Model of Software Engineering

1. Waterfall model
2. Incremental model
3. Prototype model
4. Spiral model
5. RAD model
6. Agile (Framework->Scrum)

Software Process model

Waterfall model



Requirement are well understood

Software process model

Incremental model

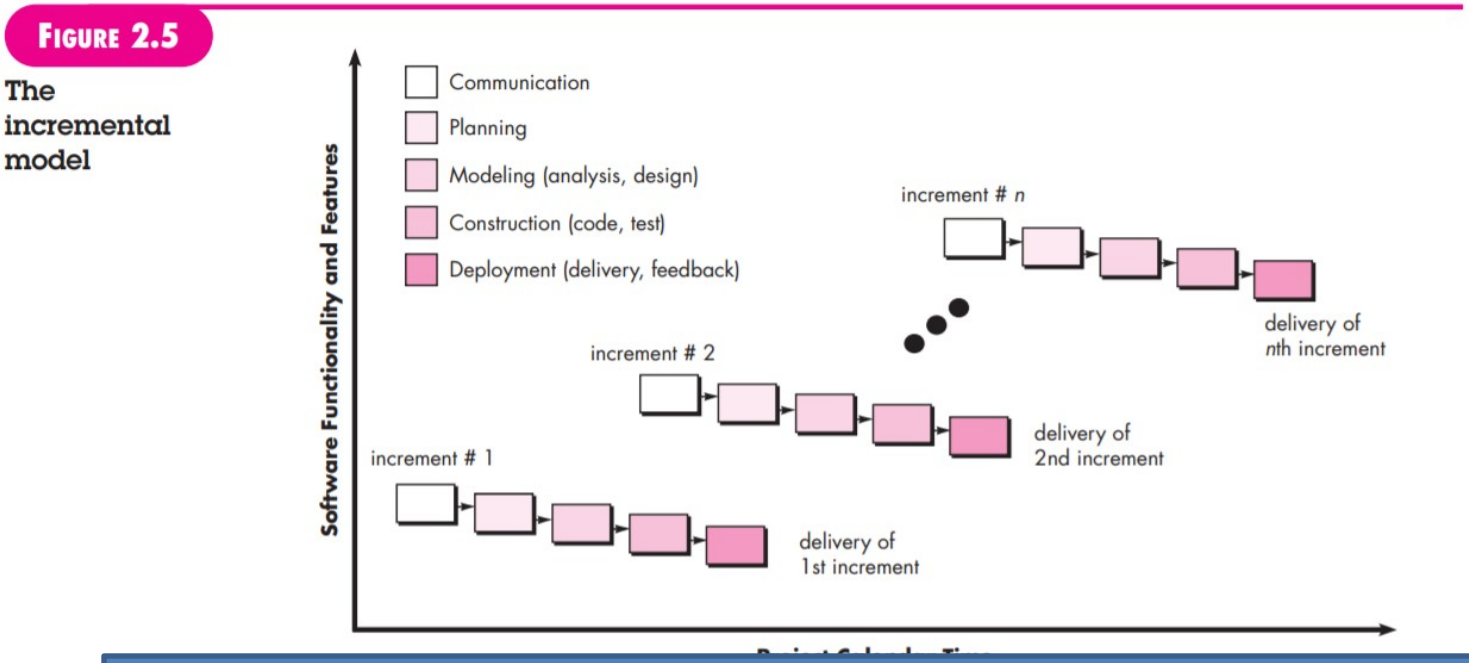


FIGURE 2.5
The incremental model

When an incremental model is used, the first increment is often a core product

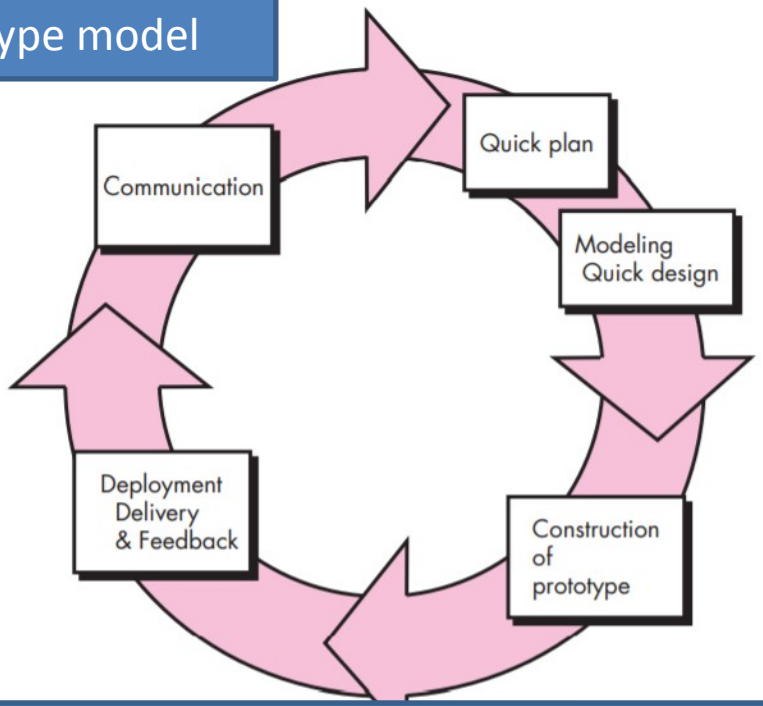
IF there may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software

Software Process model

FIGURE 2.6

The prototyping paradigm

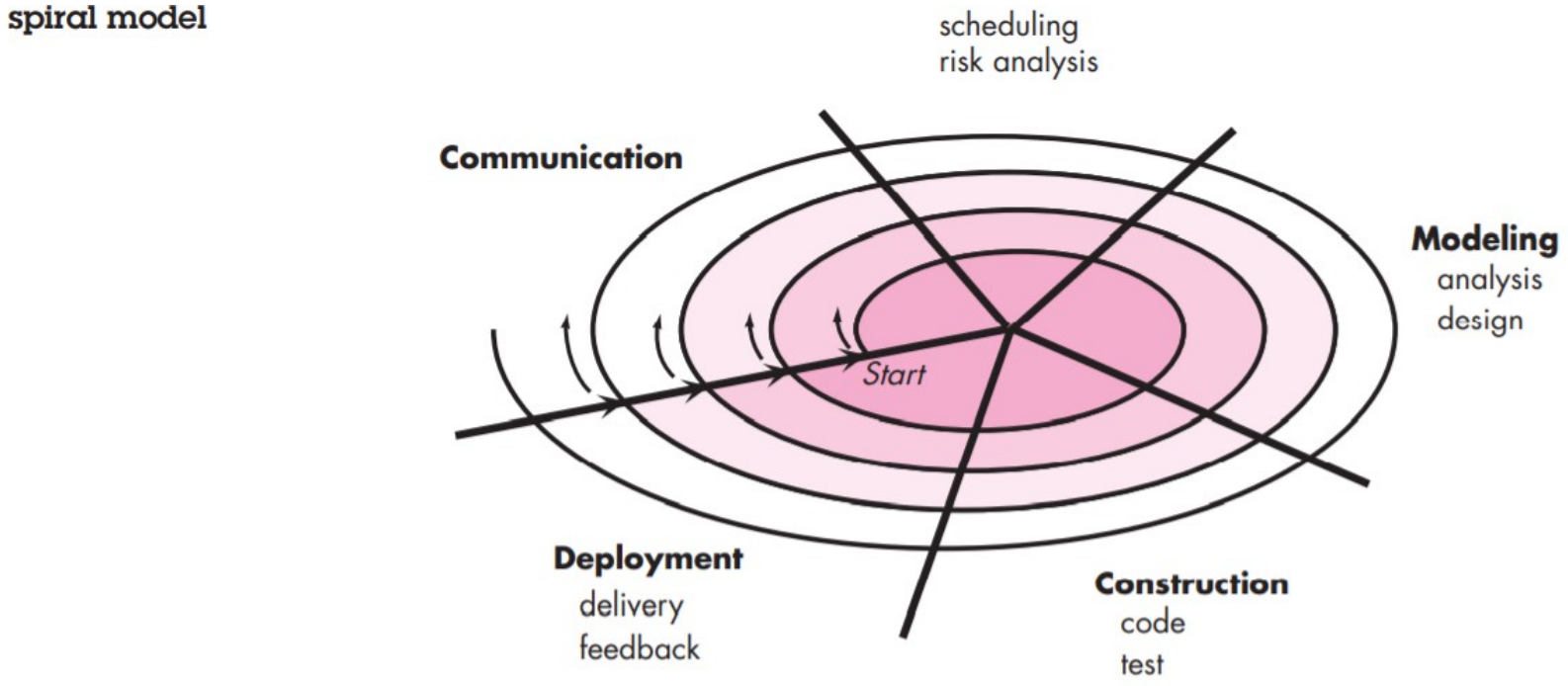
Prototype model



Often, a customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. In other cases, the developer may be unsure of the efficiency of an algorithm

Software Process Model

Spiral Model

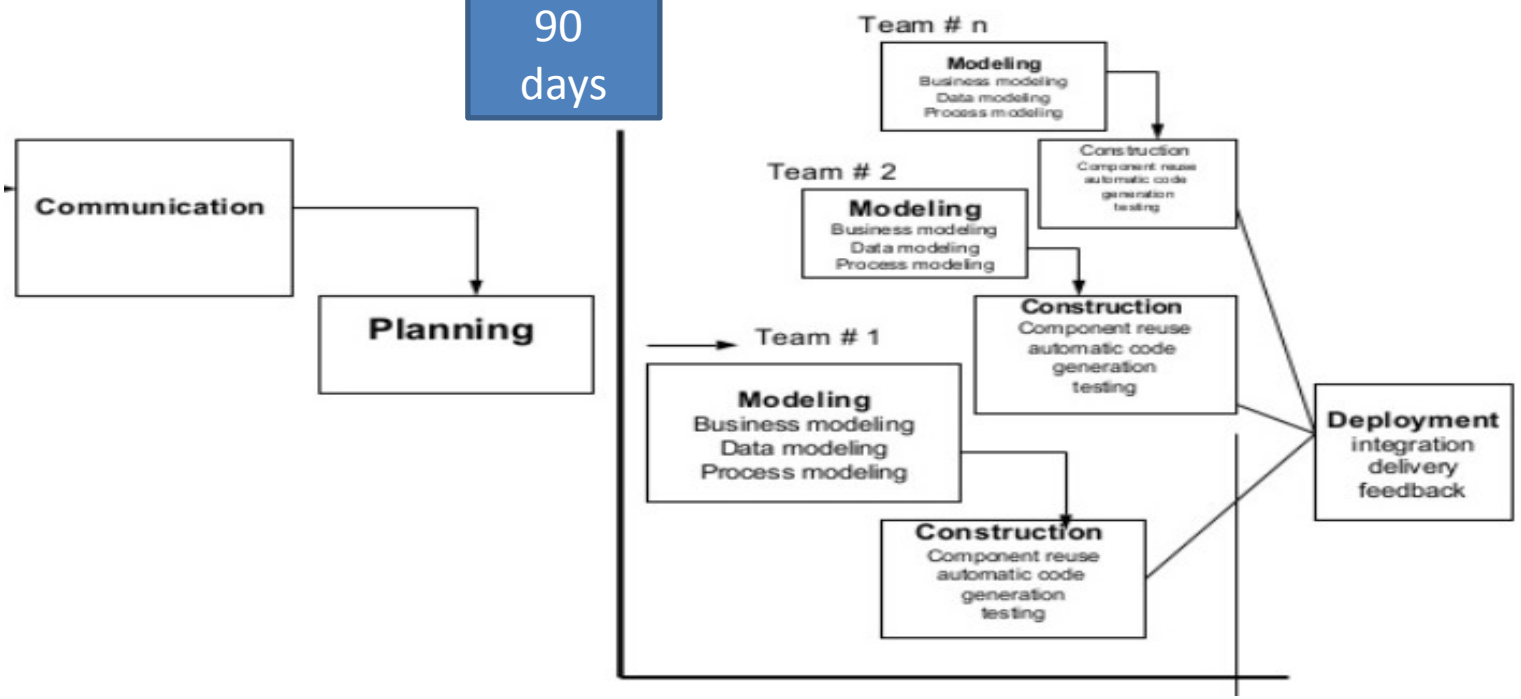


The spiral development model is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems

Software process model

Rapid Application Development

60 to 90 days



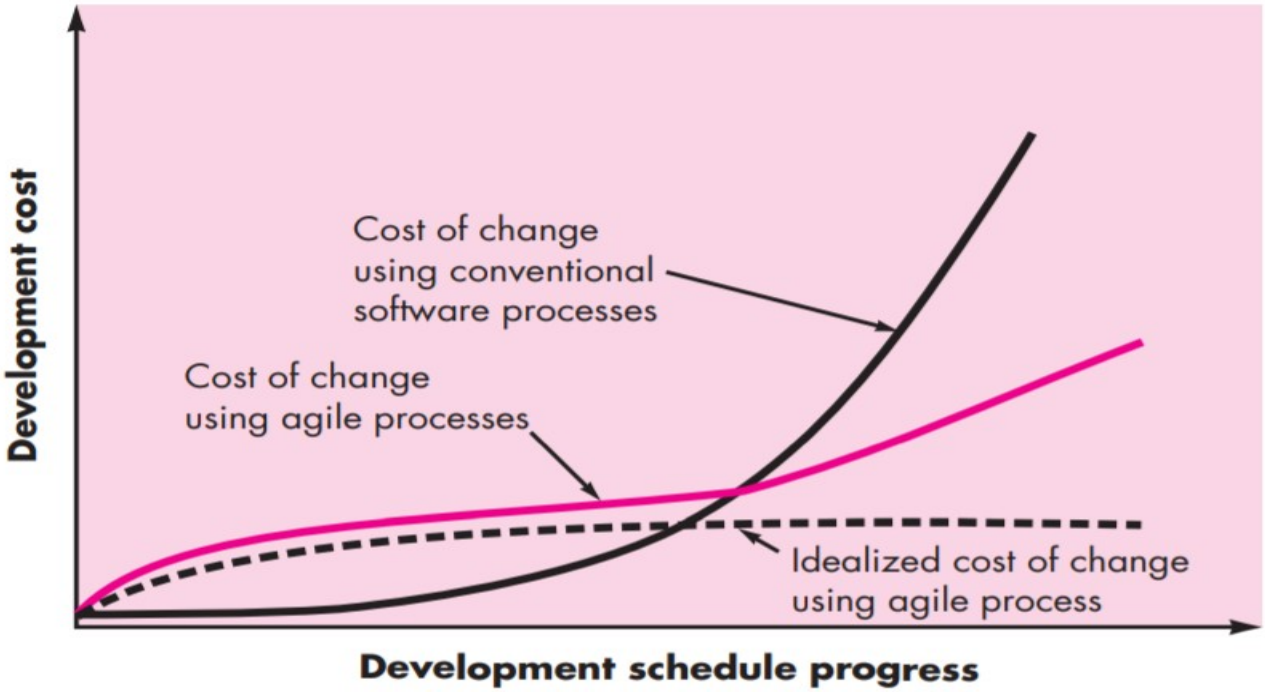
Software process model

Agile

- Agility has become today's buzzword when describing a modern software process.
- An agile team quickly respond to changes.
- Change is what software development is very much about.
- Agility has **12 principles** and **four guidelines**

Software process model

Agile Process



Software process model

Scrum

- It is one of the **popular framework** for the agile development
- Other popular frameworks are Kanban, Extreme Programming (XP)

Software process model

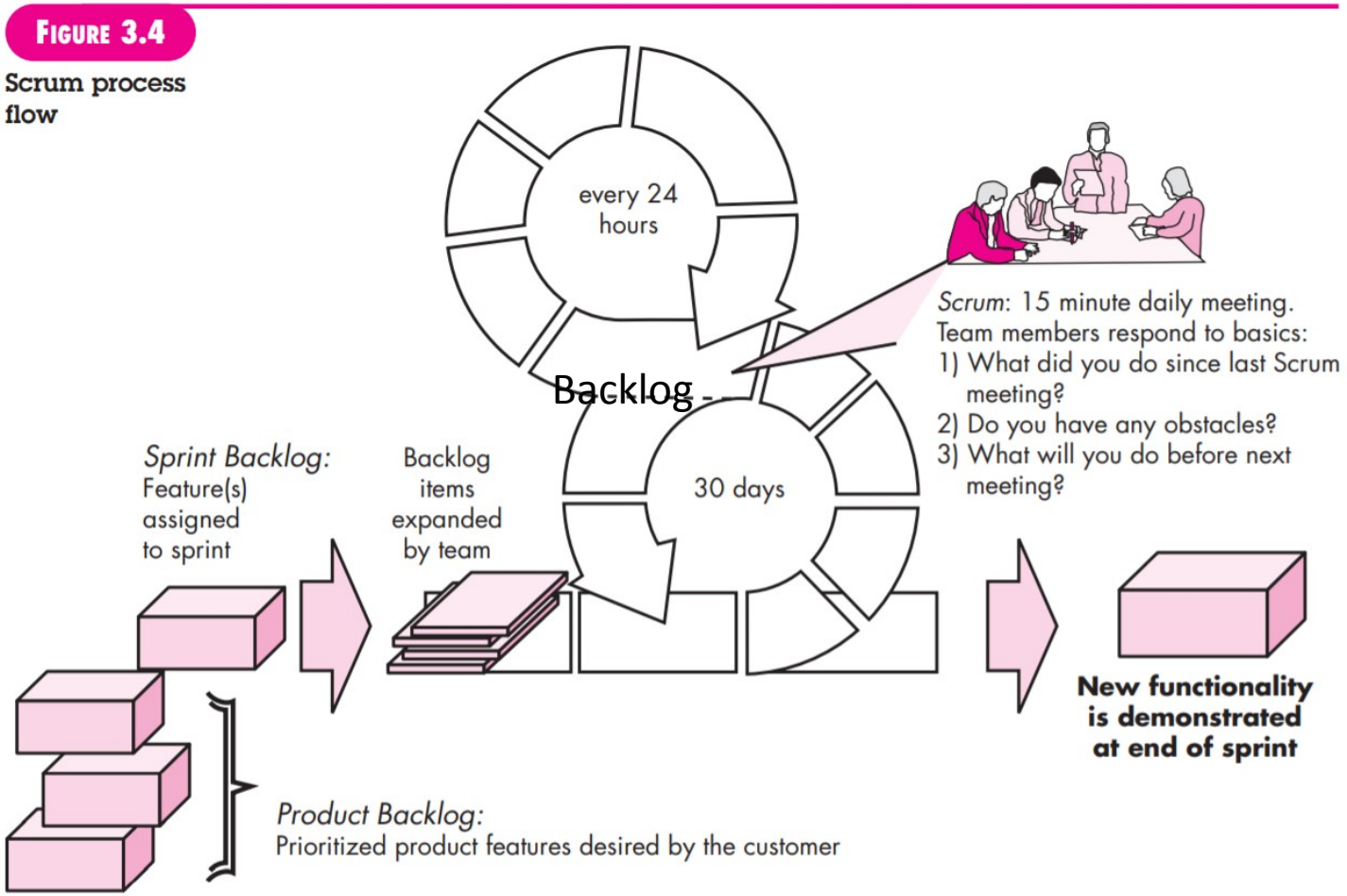
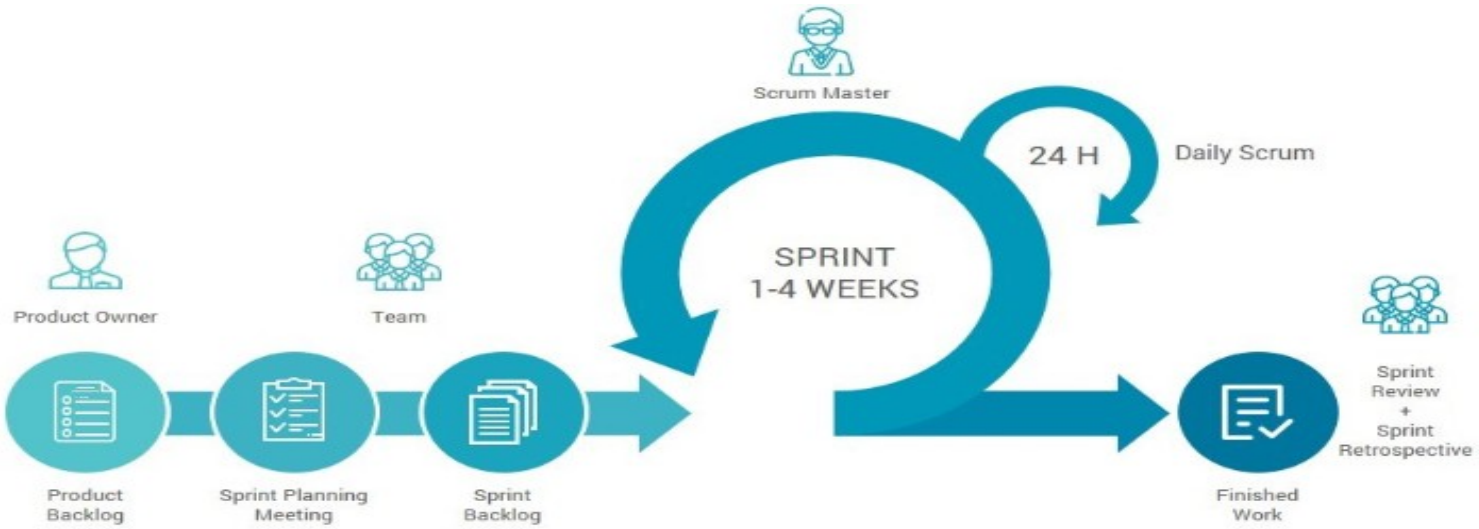


FIGURE 3.4
Scrum process flow

Software process model

Scrum process



Scrum Meeting

Scrum

Backlog :

- Items can be added to the backlog at any time (this is how changes are introduced).
- The product manager assesses the backlog and updates priorities as required.

Sprint:

- consist of work units that are required to achieve a requirement defined in the backlog that must be fit into a predefined time-box(typically 30 days).

Scrum

Scrum Meeting

- Scrum meetings are short (typically 15 minutes) meetings held daily by the Scrum team.

Three key questions are asked and answered by all team members

1. **What did you do since the last team meeting?**
2. **What obstacles are you encountering?**
3. **What do you plan to accomplish by the next team meeting?**

SCRUM

- A team leader, **Scrum master**, leads the meeting and assesses the responses from each person.
- The Scrum meeting helps the team to uncover potential problems as early as possible.

Demos:

- Deliver the software increment to the customer so that functionality that has been implemented can be demonstrated and evaluated by the customer.
- Demo may not contain all planned functionality, but rather those functions that can be delivered within the time-box that was established

Process model

Unified process

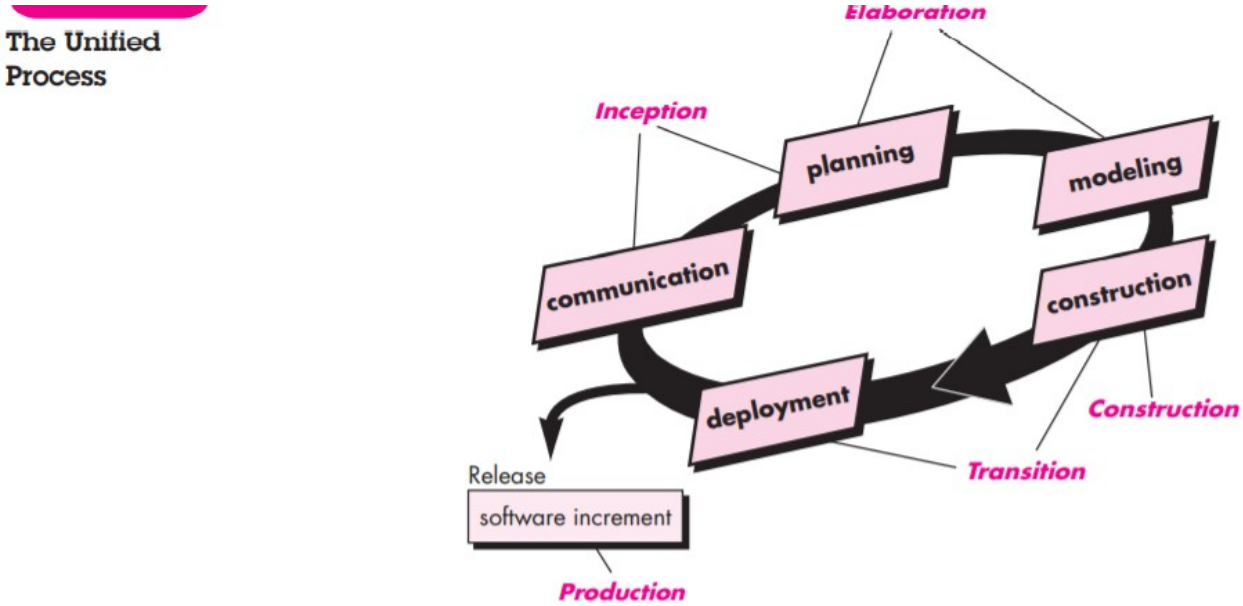


Fig : The Unified process

FOUR P’s

- Effective software project management focus on Four P’s
    - **People**
    - **Product**
    - **Process**
    - **Project**

Four P's

People

- Highly **skilled manpower** is needed for the software industry
- Software industry follows the ***People Management capability Maturity Model (PM-CMM)*** *to enhance the readiness of software organization.*
- ***PM-CMM*** *defines the key practice areas for software: recruiting, performance, training, selection, compensation*

Four P's

Product

- Before a project can be planned**, product objective and scopes** should be established
- Only then we can establish **accurate estimation of cost, effective project management, risk assessment, scheduling**.

Four P's

Process

An effective software process provides the framework from which comprehensive plan for software development is established.

Four P's

Project

- We conduct planned and Controlled software project for one primary reason – it is the only way to manage the complexity.
- Many software projects have failed due to poor management of the project
- In order to avoid the project failure a software project manager and the software engineer must develop the commonsense approach for planning, monitoring and controlling the project.