CMPE 331

A software includes → Instructions (provide desired features)
Data Structures (enable the program to adequately manipulate info)
Documents (describe the operation)

Differences Between Software and Hardware

Software is developed or engineered; it is not manufactured in the classical sense.

It is harder to manage software projects.

The industry is moving toward component-based cons. most software continues to be custom built.

Software doesn't wear out.

Changing Nature of Software

System Software
Application Software
Engineering/scientific Software
Embedded Software

Legacy Software

It is sometimes hard to cope with old big softwares
↓
What should we do: Support core business functions
Have longevity and business critically
(no short time solutions)

Do not allow poor quality
↳ poor documentation
poor testing
⋮

182 [illegible]

When we make a change in software what shold be consider

Adaptive
Perfective
Corrective

} These are also the 3 major reasons for any software maintenance.

Software Products

Generic Products → Stand-alone systems that are marketed and sold to any customer who wishes to buy them.

↓ owned by Software Developer

Ex: PC software
Systems for dentists

Customized Products → qualified by specific customer

↓ owned by Customer

Ex: Embedded control systems
Air traffic control
Traffic monitoring

Essential attributes of good Software
↓
Maintainability
Dependability and Security
Efficiency
Acceptability

Software Process Activities

Software specification
Software development
Software validation
Software evolution

General issues that affect software
↓
Heterogeneity
Business and social change
Security and trust
Scale

The software engineering methods and tools used depend
on the type of app being developed
the requirements of the customer and
the background of the development team

Application Types 1

Stand-alone
Interactive transaction-based → Apps that execute on a remote computer and are accessed by users from their own PCs.
Embedded control

Application Types 2

Batch Processing → that are designed to process data in large batches.
Entertainment
Systems for modelling and Simulation → developed by scientists and engineers to model physical processes or situations.

Application Types 3

Data Collection
Systems of systems

Web Software Engineering → Software reuse
Incremental and agile development
↓
Service-oriented Systems
Rich Interfaces

Software Engineering Ethics

Issues of professional responsibility
↓
Intellectual property rights
Computer misuse
Confidentiality
Competence

Software Process Descriptions

Process descriptions include → specifying data model
designing UI
Products
Roles
Pre and post-conditions

Plan-driven processes → where all of the process activities are planned in advance and progress is measured against this plan.

Agile Processes → planning is incremental and it is easier to change the process to reflect changing customer requirements

Software Process Models

The waterfall model

↓

Plan-driven model. Separate and distinct phases of specification and development

Phases

Requirements analysis and definition

System and software design

Implementation and unit testing

Integration and system testing

Operation and maintanance

The main drawback of this model is the difficulty of accommodating change after the process is underway. A phase has to be complete before moving onto the next phase.

It is difficult to respond to changing customer requirements. Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

This model is mostly used for large systems engineering projects where a system is developed at several sites

Incremental development

↓

The whole model is divided into various builds. Multiple development cycles take place here. Cycles are divided up into smaller, more easily managed modules

Benefits

The cost of accomodating changing customer req. is reduced.

It is easier to get customer feedback on the development work that has been done.

More rapid delivery and deployment of useful software to the customer is possible

Problems

The process isn't visible ⇒ If systems are developed quickly, it isn't cost effective to produce documents that reflect every version of the system.

System structure tends to degrade as new increments are added.

Integration and Configuration

↓

The system is assembled from existing configrable components. May be plan-driven of agile. Reused elements may be configured to adapt their behavior and functionality to a user's requirements.

Types of Reusable Software

* Stand-alone application systems
* Collections of objects that are developed as a package to be integrated with a component framework. Ex» .NET
* Web services that are developed according to service standards and which are available for remote invocation.

Stages → Requirements specification
Software discovery and evolution
Requirements refinement
Application system configuration
Component adaptation and integration

Advantages and Disadvantages

* Reduced costs and risks as less software is developed from scratch.
* Faster delivery and deployment of system.
* But requirements compromises are inevitable because of the dependancy of the used application so system may not meet real needs of users
* Loss of control over evolution of reused system elements

Design Activities → Architectural Design → identify overall structure
Database Design
Interface Design
Component selection and design

Testing Stages → Component Testing
System Testing
Customer Testing → Testing with customer data.

Benefits of prototyping → Improved system usability
A closer match to user's real needs
Improved design quality
Improved maintainability
Reduced development effort.

Prototypes are normally undocumented
The prototype structure is usually degraded through rapid change

Incremental Delivery → Dev. and delivery is broken down into increments.
User req. are prioritised.
Once the dev of an increment started the req. are frozen though req. for later increments

Incremental Delivery Advantages → Early increments act as a prototype to help elicit req.

Lower risk of overall project failure.

The highest priority system services tend to receive the most testing.

↓

Problems

As requirements aren't defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all inc.

Process Improvement

* The process maturity approach, which focuses on improving process and project management and introducing good software engineering practice.

* The agile approach, which focuses on iterative dev. and the reduction of overheads in the software process.

Process Imp. Activities

Measurement
Analysis
Change

Agile Methods

Scrum → An agile process that allows us to focus on delivering the highest business value in the shortest time.

Extreme Programming

Adaptive Software Development (ASD)

Dynamic System Development Method (DSDM)

Scrum Characteristics

Self-organizing teams

Product progresses in a series of month-long sprints

Requirements are captured as items in a list of product backlog

No specific engineering practices prescribed

Uses generative rules to create an agile environment for delivering projects

Sprints

Scrum projects make progress in a series of sprints

Target duration 1 month

Product is designed, coded and tested during the sprint.

No changes during the sprint.

Scrum Framework

Roles: Product Owner, ScrumMaster Team

feedback Meeting
↑

Ceremonies: Sprint Planning, Sprint Review, Sprint Retrospective & Daily Scrum Meeting

Artifacts: Product Backlog, Sprint Backlog and Burndown Chart

Product Backlog ↓

* A list of all desired work on the project.
* List is prioritized by the product owner.

Sprint Backlog ↓

* Created only by Team Members
* Each item has it's own status
* Should be updated every day
* Team can add or subtract items from the list.

Sprint Burn Down Chart

Depicts the total Sprint Backlog hours remaining per day.

Shows the estimated amount of time to release.

Ideally should burn down to zero to the end of the sprint.

Actually is not a straight line

Can bump up

Scrum

↓

Pros

- * Completely develped and tested features in short iterations.
- *Simplicity of the process
- *Clearly defined rules
- *Increasing productivity
- *Self-organizing
- *Each team member carries a lot of responsibility
- * Improved communication
- * Combination with Extreme programming

Cons

- * Undisciplined hacking (no written doc)
- *Violation of responsibility
- * Current mainly carried by the inventors

System Perspectives

External Perspective → you model the context or env. of the system.

Interaction Perspective → you model the inter. between a system and its env.

Structural Perspective → you model the organization of a system

Behavioral Perspective → you model the dynamic behavior of the system and how it responds to events

UML Diagram Types

Activity Diagrams
Use Case Diagrams
Sequence Diagrams
Class Diagrams
State Diagrams

Context Models

Context Models are used to illustrate the operational context of a system.
They show what lies outside the system boundaries
Social and organisational concers may effect the sys. bound.

Process Perspective → Process models are related how the system being developed is used in broader business processes.

Interaction Models

Modeling user interaction is important.
Modeling system to system int. highlights the communication prob. that may rise.
Use case diagrams and sequence diagrams may be used for interaction modelling

Use Case Modelling

Each use case represents a discrete task that involves external interaction with a system.
Actors in a use case may be people or other systems.

Sequence Diagrams → model the interactions between the actors and the objects within a system.
* Shows the sequence of interactions that take place during a particular use case

Structural Models

Structural models may be static models, which show the structure of the system design.
Structural Models may be dynamic models, which show the organization of the system when it is executing ([illegible])

Behavioral Models

They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
These stimuli are two types:

Data → Some data arrives that has to be processed by the system
Events → Some event happens that triggers system processing

Data - Driven Modeling

They are controlled by the data input to the system.
Shows the sequence of actions related with processing input data and generating an associated output.
They are particularly useful during the analysis of requirements
They can be used to show end-to-end processing in a system.

Event-Driven Modelling

Real time systems are often event-driven, with minimal data processing.
Shows how a system responds to external and internal events.

State Machine Models

Shows system states as nodes and events as arcs between these nodes.

Model Driven Engineering

The principal outputs of the development process are models rather than programs.
The programs that execute on a hardware/software platform are then generated automatically from the models.

Pros	Cons
Allows system to be at higher levels of abstraction. It is cheaper to adapt systems to new platform.	Models for abstraction may not be right for implementation. Developing translators for new platform may cost more than expected.

Model Driven Architecture → It uses a subset of UML models to describe a system.
Models at different levels of abs. are created.
It is a high level, platform independe model.