# Software Engineering Practice and Experience

## Unit 1

### Software Lifecycle

Dealing with complexity : process models/ process model evolution

Software engineering process models

Software engineering methodologies

Technical engineering practices (CVS etc…)

Quality Engineering practices

In general, the development and maintenance of software involves:

Specification

Production

Validation

Evolution

Software process Models

Providing a structure to the development of large software systems.

Sometimes processes are referred to as paradigms; a set of steps that encompasses methods, tools and procedures.

The following points apply to the paradigm:

* Independent of the specifics of how each step is accomplished
* May be explicitly designed to support a specific tool/technique
* Informal/heavily formalised.
* May be enforced by tool/management

Quality is the end result of good processes, methods and tools.

#### Processes

The foundation for software engineering. Enables everyone to be aware of the steps involved in producing software. Template for software development

Hold the technology layers together by enabling an abstraction of various stages. Removing the detail enables all levels of the company to manage and understand the glue that holds the components together.

Rational/timely development of software.

Key process areas

* What the company decides is important
* Basis of management control of SP
* Context in which milestones are established
* What technical methods are applied
* How work is actually produced (reports etc…)
* Change management
* QA

#### Methods

How do we execute the instructions the process is telling us to carry out? Technical how-to build software

* Requirements analysis
* Design
* Program Construction
* Testing
* Support

#### Tools

Tools automate support for the process and methods. CASE tools

* Code production tools
* Data modelling tools
* UML tools
* Configuration management tools (Revision control)

#### Advantages of process models

1. Checking that all tasks are completed
2. Checking all tasks are done in the right order
3. Discourages coding without spec
4. Ensures that quality is checked before the start
5. Encourages the development of test plans
6. Helps to ensure that tools are used for the job
7. Coordinates people / handing over
8. Project planning and management becomes easier
9. Facilitates setting of milestones and deliverables

#### Disadvantages of process models

1. Fixed models may encourage rigidity in development mechanisms and consequently discourage innovation
2. If model is inappropriate, it may cause the product to be worse.

Models encourage rigidity and discourage innovation (could be a good thing to discourage hacking)

Process models

* Linear sequential models
  + Waterfall
  + Enriched waterfall
  + V lifecycle
* Prototyping models
* RAD
* Evolutionary Models
  + Incremental
  + Spiral
  + Concurrent Development
* Component based development

Provides a technical framework for OO systems. Allows components to be reused and is evolutionary in nature. Components stored in a class library and distributed accordingly. Saves significant time and effort.

##### Waterfall Model

Winston Royce. Output of each stage is a document.

* Analysis
* Design
* Code Generation
* Testing
* Final product

Certainty at every stage. We know what we have achieved at every stage and we know what we have to achieve. Output very reliable

Obviously does not deal with changes in requirements.

##### Enriched Waterfall

Adding feedback from one phase to the next phase. For this to work development work has to start as late as possible in the cycle. You can then feedback continuously until a usable requirements spec is produced.

Waterfall models are unsuitable where:

* There is uncertainty in the application domain
* Research is needed
* Internet based / ecommerce based systems where version cycle is very short

On the other hand it ensures quality because all work must be QA – ed before the progression to the next phase.

##### V lifecycle

Testing is planned before any of the coding starts.

High level design phase : system architecture + integration test plan

Low level design phase: how to code the components + unit testing.

Clearly separates product and process.

The spec addresses what the whole system should do, but this is decomposed into small units/ modules of code.

* Straightfoward to use
* Specific deliverables
* Test plans at all stages
* Suited for projects where requirements are well understood

However is quite inflexible.

* Inflexible
* No visible prototype after unit testing
* Does not address problems encountered in testing

#### Alternatives to the linear sequential models

The principal problem with waterfall style process models is their rigidity / inflexibility.

Assuming the customer knows what they want can sometimes be very inaccurate.

A solution for this is prototyping..allowing the client to view the system early on and point out possible problems.

##### Prototyping Process models

Screen layouts; Non-functioning mockups of sample screens and report layouts.

Throwaway prototypes. The system is developed very quickly and refined in frequent discussions with the client. Prototype does not contain the system architecture but allows the user to figure out what he needs

Evolutionary prototypes…incrementally developed. Prototype to be built into the finished product.

Disadvantages of prototyping; Design may be of poorer quality and the resulting system may be harder to maintain.

##### Incremental Development Prototyping

* Set out requirements for complete system
* Set out development of part of the system
* Choose what users need the most in this part of the system
* Implement it (takes less time since only a part is implement it)
* Deliver
* Iterate until budget is used

Product is built as separate components over time.

###### Advantages

* Early delivery
* Live testing
* More likely to deliver software in time
* Priority is delivery of what the user actually needs
* Can structure the system since requirements are known
* Can structure development, since complete outline is known

###### Disadvantages

Difficult to know when to stop development

The quality of the system may be compromised due to piece by piece development

##### RAD

A strategy for system development that relies on a high degree of user participation and exploits modern high powered software tools to shorten the lifecycle of information system development.

RAD Lifecycle:

* Requirements planning
* User design
* Construction
* Cutover

###### Requirements planning

Focus on solving business problems and information requirements. Seek input from users.

###### User design

Prototyping cycle. Experienced users prototyping with experienced developers. CASE tools are used to build prototypes which undergo user evaluation

###### Construction

Code generated from CASE prototypes and users validate the new system.

###### Cutover

System testing, user training and introduction of the system.

###### Advantages/disadvantages

* High degree of user participation
* Systems are delivered more quickly
* RAD projects are cheaper than traditional counterparts
* RAD systems more closely related to current needs of business
* Speed of development means software engineering practices not so sound
* Heavy reliance on user commitment and participation can push up cost for client
* Long term modification of system can be problematic due to lack of standards
* RAD only works for certain types of system: not appropriate to mission critical apps (plane crash anyone? )

###### Agile and Extreme programming

Both subsets of RAD.

Agile methods are all built on incremental development. (Incremental prototypes)

Ideal for small to medium sized business systems. Focus on the code rather than the design as are principally intended to deliver working software quickly

Examples of agile methods

Scrum

Crystal

ASD

DSDM

Feature driven programming

Extreme programming

Scrum

The scrum approach is where the phases overlap

Scrum roles

Scrum master maintains processes

Product owner looks after the interests of the company developing the product

Team including the developers

Regular meetings in Scrum

Questions:

What have you done since last meeting?

What are you planning to do next?

What problems are stopping you from doing the next thing?

Crystal

Invented pre 1988 by Alistair Cockburn

Crystal sapphire/Crystal diamond/Crystal clear/Crystal yellow

The more complicated the system to be produced the more formalized the crystal methodology to be used becomes. The larger the project the more solid the documentation.

Roles in a crystal clear system

* Sponsor
* Senior designer
* Programmer

Roles in a crystal orange system

* Architect
* Sponsor
* Business analyst
* Project manager

Crystal methodology 7 points

* Frequent delivery
* Continual feedback
* Constant communication
* Safety
* Focus
* Access to end users
* Automated tests and integration

Adaptive Software Development

Allows for continuous change to the process and consequently is considerably different compared to normal models.

Speculate

The stage when the project is initiated and the adaptive planning cycle takes place.

Collaboration

Planning balanced against unpredictable aspects. All stage requirements are gathered and mini specs generated

Learn cycles

This is when the components are implemented and tested. Formal technical reviews based on the idea that making mistakes based on false assumptions = improved knowledge and gain greater command of domain to avoid mistakes in future.

###### DSDM – Dynamic Systems Development Method

Similar to scrum. Is split into three main phases:

* Pre project phase
* Project life-cycle phase
  + Feasibility study
  + Business study
  + Functional Model iteration
  + Design and build iteration
  + Implementation
* Post project phase (maintenance and enhancements)

###### DSDM Principles

1. User involvement
2. Empowered project team
3. Focus on frequent delivery of products
4. Emphasis on delivering a system that addresses business needs
5. Iterative and incremental development
6. The changes during development are reversible
7. The high level scope and requirements should be baselined before the project starts
8. Testing is carried out throughout the lifecycle
9. Cooperation among stakeholders

###### DSDM Roles

* Executive Sponsor
* Visionary (person who starts the project)
* Ambassador User (represents the user community)
* Advisor User (updates everyone on the daily progress of the project)
* Project Manager (individual manages the project)
* Technical coordinator
* Team Leader (keeps the team together)
* Developer (interprets system requirements and builds them)
* Tester
* Scribe (gathering and recording the requirements decided in every workshop along with all agreements made)
* Facilitator (facilitating communication between users)

###### DSDM Techniques

* Timeboxing
* Must Could Would (priorities in project)
* Prototyping
* Testing
* Workshop
* Modelling
* Configuration management

##### Feature Driven Development

FDD consists of five activities. These are listed below:

###### Develop Overall Model

High level walkthrough of the system

###### Build Feature List

List of features (components of functionality)

###### Plan By Feature

Ordering of features and assigning them to chief programmers

###### Design by Feature

Chief programmer designs sequence diagrams for each feature.

###### Build by Feature

The class owners develop the actual code

#### Extreme Programming

Extreme programming is the most widely used and known agile method. Involves the following key practices:

* Customer involvement in both the development and management of the project.
* Pair programming
* Emphasis on simplicity
* Constant testing at all stages. Test driven development
* Small releases
* Based around a metaphor

**Iteration Planning Meeting**

Customer writes requirements as stories. Each story is broken down into simple stories. Each story is written onto a card and distributed to the development team. Customer explains the scenario to development team.

Tasks depend on the length and size of the ‘story’. The developers are able to estimate the size of their story at this stage, and a schedule is calculated based on the size of the story.

Once the estimates are done, the customer prioritises the stories based on this information.

##### System as a metaphor

Metaphor should describe the approach to the solution. Places an emphasis on a product vision, which helps everyone understand the basic components of the product.

* A common vision for the developer and customer.
* Identifies the structure of the problem to be addressed and establishes a vocabulary for the objects and relationships between them.
* Provides a basis for system architecture
* Brings out issues, problems and solutions

##### XP Iteration

###### Planning

The programmers choose the tasks they want to do. A minimal set of functionality that provides business benefit is then implemented.

###### Doing

The customer writes the acceptance tests, which the implemented stories must satisfy.

Test driven development (generally felt that the time involved writing the test code and the system code is the same as the time to develop the code).

Process of developing the test code helps the programmer to think about the system code.

System is constantly refactored to meet changing requirements. As soon as task is complete it is implemented into the system.

###### Design Guidelines in XP

Simple design

Constant refactoring

Customer is always available and part of the team

###### Coding Guidelines in XP

Code must be written to agreed standards

All code is PAIR programmed, and only one pair integrates code at a time

The integration tests should take place often

Collective code ownership should be used

No overtime!

###### Testing Guidelines in XP

All code must have unit tests

Code must pass all unit tests before it can be released

When a bug is found, new tests are created to test for re-emergence of the bug

There is always user involvement in test dev and validation

Acceptance tests must be run often

The frequent use of automated test environments (Junit?) used on all component tests

###### Pair programming

Two programmers sit at one computer: One person programming whilst the other is keeping a higher-level viewpoint. They typically switch roles a few times per session.

* High/low level views of the system
* Non programmer is constantly reviewing code.
* Mutual training is built in. Spreads knowledge across the team
* Reduces ego
* No loss of productivity
* Peer reviews

###### Refactoring

Involves reworking class structure/details to make further changes easier.

Objectives of refactoring

* Cleaning up design
* Refining class hierarchies
* Deleting unnecessary code
* Reversing the problem of design decay

Process of refactoring

* Do a refactoring exercise
* Run unit tests to ensure behaviour has not changed
* Iterate until design is ok

All within the framework of passed unit tests.

Reasons for refactoring

Improves the design

Makes the system easier to understand / modify

Improves performance

Solves bugs

XP advantages

Moves programmers around

Spreads knowledge

Reduces dependence on key individuals

Supports inexperienced programmers

All tests developed before the code is written

XP disadvantages

Emphasizes the programmer as a ‘hero’

May lead to failures if you have too many inexperienced programmers

Not all people work well in pairs

Refactoring = less structured code

Priorities are constantly being revised

Process may be seen as undisciplined from a management perspective.

### CMI/CMMI

#### Evolutionary Software process models

##### Incremental Model

##### Spiral Model

##### Components Based Development

#### Methodologies

##### Waterfall based methodologies

###### SA/SD

The first generally used methodology. A complete process and displays the difference between phases by using different notations.

DFD

###### SSADM

A partial process; Data oriented; requirements stages use data flow/logical data modelling. Later versions introduce relational DA, ERD and data catalogues. Feedback is discouraged by strict deliverable idea.

###### Automating methodologies

Case tools