

Chapter 1

Chapter 1 An Introduction to Software Engineering



Learning Objective

...to give an appreciation for and to introduce *software engineering* and to place it into context with the *system engineering process, process models* and some of the fundamentals to *project management*.



Frederick T Sheldon
Assistant Professor of Computer Science
Washington State University

Software Engineering

Designing, building and maintaining large software systems

Objectives

- To define software engineering and explain its importance
- To discuss the concepts of software products and software processes
- To explain the importance of process visibility
- To introduce the notion of professional responsibility

Topics covered

- Software products
- The software process
- Boehm's spiral model
- Process visibility
- Professional responsibility

Software engineering

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development
- Software engineering expenditure represents a significant fraction of GNP in all developed countries

Software costs

- Software costs often dominate system costs. The costs of software on a PC are often greater than the hardware cost
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs
- Software engineering is concerned with cost-effective software development

Software products

Generic products

Stand-alone systems which are produced by a development organization and sold on the open market to any customer

Bespoke (customized) products

Systems which are commissioned by a specific customer and developed specially by some contractor

Most software expenditure is on generic products but most development effort is on bespoke systems

Software product attributes

Maintainability

It should be possible for the software to evolve to meet changing requirements

Dependability

The software should not cause physical or economic damage in the event of failure

Efficiency

The software should not make wasteful use of system resources

Usability

Software should have an appropriate user interface and documentation

Importance of product characteristics

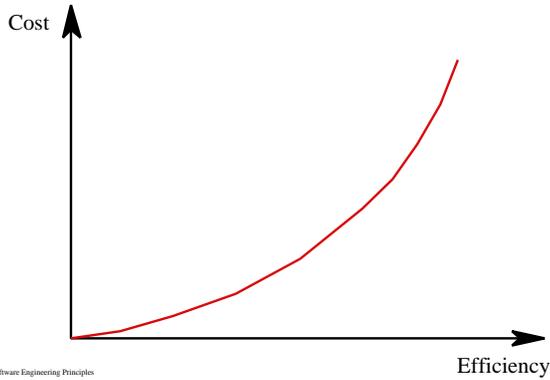
The relative importance of these characteristics depends on the product and the environment in which it is to be used

In some cases, some attributes may dominate

In safety-critical real-time systems, key attributes may be dependability and efficiency

Costs tend to rise exponentially if very high levels of any one attribute are required

Efficiency costs



The software process

Structured set of activities required to develop a software system

- Specification*
- Design*
- Validation*
- Evolution*

Activities vary depending on the organization and the type of system being developed

Must be explicitly modeled if it is to be managed

Process characteristics

Understandability

Is the process defined and understandable

Visibility

Is the process progress externally visible

Supportability

Can the process be supported by CASE tools

Acceptability

Is the process acceptable to those involved in it

Process characteristics

Reliability

Are process errors discovered before they result in product errors

Robustness

Can the process continue in spite of unexpected problems

Maintainability

Can the process evolve to meet changing organizational needs

Rapidity

How fast can the system be produced

Engineering process model

Specification - set out the requirements and constraints on the system

Design - Produce a paper model of the system

Manufacture - build the system

Test - check the system meets the required specifications

Install - deliver the system to the customer and ensure it is operational

Maintain - repair faults in the system as they are discovered

Software process models

Normally, specifications are incomplete/anomalous

Very blurred distinction between specification, design and manufacture

No physical realization of the system for testing

Software does not wear out - maintenance does not mean component replacement

Generic software process models

The waterfall model

Separate and distinct phases of specification and development

Evolutionary development

Specification and development are interleaved

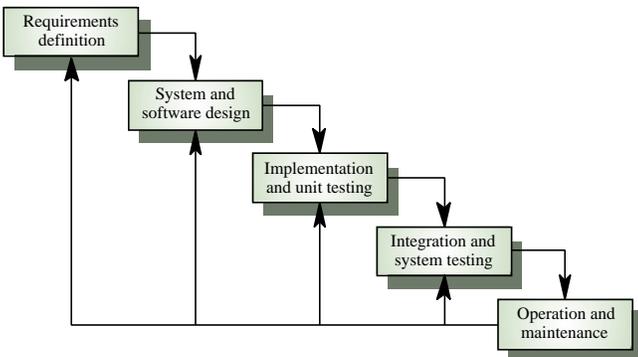
Formal transformation

A mathematical system model is formally transformed to an implementation

Reuse-based development

The system is assembled from existing components

Waterfall model



Waterfall model phases

Requirements analysis and definition

System and software design

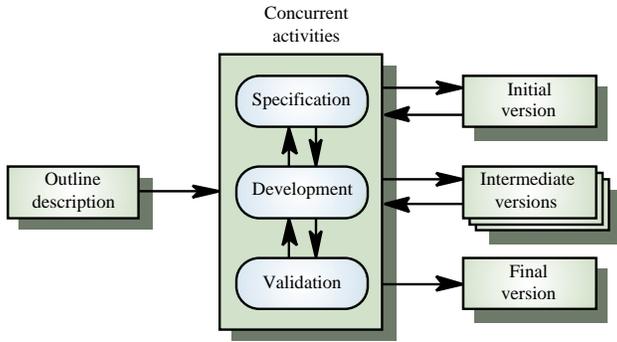
Implementation and unit testing

Integration and system testing

Operation and maintenance

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

Evolutionary development



Evolutionary development

Exploratory prototyping

Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements

Throw-away prototyping

Objective is to understand the system requirements. Should start with poorly understood requirements

Evolutionary development

Problems

- Lack of process visibility
- Systems are often poorly structured
- Special skills (e.g. in languages for rapid prototyping) may be required

Applicability

- For small or medium-size interactive systems
- For parts of large systems (e.g. the user interface)
- For short-lifetime systems

Risk management

Perhaps the principal task of a manager is to minimize risk

The 'risk' inherent in an activity is a measure of the uncertainty of the outcome of that activity

High-risk activities cause schedule and cost overruns

Risk is related to the amount and quality of available information. The less information, the higher the risk

Process model risk problems

Waterfall

High risk for new systems because of specification and design problems

Low risk for well-understood developments using familiar technology

Prototyping

Low risk for new applications because specification and program stay in step

High risk because of lack of process visibility

Transformational

High risk because of need for advanced technology and staff skills

Hybrid process models

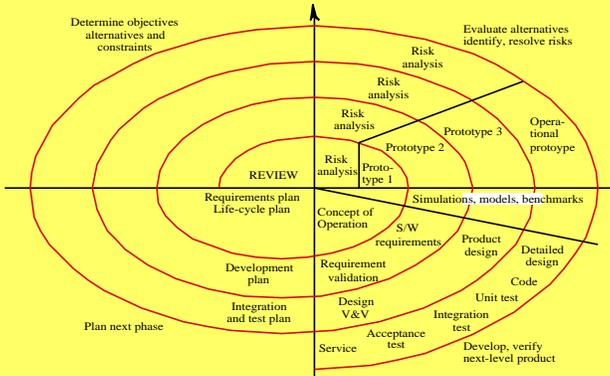
Large systems are usually made up of several sub-systems

The same process model need not be used for all subsystems

Prototyping for high-risk specifications

Waterfall model for well-understood developments

Spiral model of the software process



Phases of the spiral model

Objective setting

Specific objectives for the project phase are identified

Risk assessment and reduction

Key risks are identified, analyzed and information is sought to reduce these risks

Development and validation

An appropriate model is chosen for the next phase of development.

Planning

The project is reviewed and plans drawn up for the next round of the spiral

Template for a spiral round

Objectives

Constraints

Alternatives

Risks

Risk resolution

Results

Plans

Commitment

Quality Improvement

Objectives

Significantly improve software quality

Constraints

Within a three-year timescale
Without large-scale capital investment
Without radical change to company standards

Alternatives

Reuse existing certified software
Introduce formal specification and verification
Invest in testing and validation tools

Risk Management

Risks

No cost effective quality improvement possible
Quality improvements may increase costs excessively
New methods might cause existing staff to leave

Risk resolution

Literature survey
Pilot project
Survey of potential reusable components
Assessment of available tool support
Staff training and motivation seminars

Results, Plans and Commitment

Results

Experience of formal methods is limited - very hard to quantify improvements
Limited tool support available for company standard development system.
Reusable components available but little reuse tool support

Plans

Explore reuse option in more detail
Develop prototype reuse support tools
Explore component certification scheme

Commitment

Fund further 18-month study phase

Catalogue Spiral

Objectives

Procure software component catalogue

Constraints

Within a year

Must support existing component types

Total cost less than \$100,000

Alternatives

Buy existing information retrieval software

Buy database and develop catalogue using database

Develop special purpose catalogue

Catalogue Spiral (continued)

Risks

May be impossible to procure within constraints

Catalogue functionality may be inappropriate

Risk resolution

Develop prototype catalogue (using existing 4GL and an existing DBMS) to clarify requirements

Commission consultants report on existing information retrieval system capabilities.

Relax time constraint

Catalogue Spiral (continued)

Results

Information retrieval systems are inflexible.

Identified requirements cannot be met.

Prototype using DBMS may be enhanced to complete system

Special purpose catalogue development is not cost-effective

Plans

Develop catalogue using existing DBMS by enhancing prototype and improving user interface

Commitment

Fund further 12 month development

Spiral model flexibility

Well-understood systems (low technical risk) - Waterfall model.

Risk analysis phase is relatively cheap

Stable requirements and formal specification.

Safety criticality - Formal transformation model

High UI risk, incomplete specification - prototyping model

Hybrid models accommodated for different parts of the project

Spiral model advantages

Focuses attention on reuse options

Focuses attention on early error elimination

Puts quality objectives up front

Integrates development and maintenance

Provides a framework for hardware/software development

Spiral model problems

Contractual development often specifies process model and deliverables in advance

Requires risk assessment expertise

Needs refinement for general use

Process visibility

Software systems are intangible so managers need documents to assess progress

However, this may cause problems

- Timing of progress deliverables may not match the time needed to complete an activity

- The need to produce documents constraints process iteration

- The time taken to review and approve documents is significant

Waterfall model is still the most widely used deliverable-based model

Waterfall model documents

Activity	Output documents
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

Process model visibility

Process model	Process visibility
Waterfall model	Good visibility, each activity produces some deliverable
Evolutionary development	Poor visibility, uneconomic to produce documents during rapid iteration
Formal transformations	Good visibility, documents must be produced from each phase for the process to continue
Reuse-oriented development	Moderate visibility, it may be artificial to produce documents describing reuse and reusable components.
Spiral model	Good visibility, each segment and each ring of the spiral should produce some document.

Professional responsibility

Software engineers should not just be concerned with technical considerations.

They have wider ethical, social and professional responsibilities

No clear rights and wrongs about many of these issues

- Development of military systems

- Whistleblowing

- What's best for the software engineering profession

Ethical issues

- Confidentiality

- Competence

- Intellectual property rights

- Computer misuse

Key points

Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products

Software products consist of programs and documentation.

Product attributes are maintainability, dependability, efficiency and usability

The software process consists of those activities involved in software development

Key points

The **waterfall model** considers each process activity as a discrete phase

Evolutionary development considers process activities as concurrent

The **spiral process model** is risk-driven

Process visibility involves the creation of deliverables from activities

Software engineers have ethical, social and professional responsibilities
