

Software Engineering: The Process

Doro Edi, ST., M. Kom.
k_doroedi@yahoo.com

These courseware materials are to be used in conjunction with *Software Engineering: A Practitioner's Approach*, 6/e and are provided with permission by R.S. Pressman & Associates, Inc., copyright © 1996, 2001, 2005

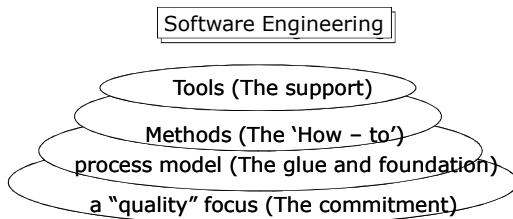
1

What is Software Engineering ?

- ▶ Software engineering is the establishment and sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines (Fritz Bauer,1969)
- ▶ Software engineering is [1] the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and [2] the study of approaches as in [1] (IEEE,1993)

* SEPA 6th ed, Roger S. Pressman

Software Engineering – a layered technology



* SEPA 6th ed, Roger S. Pressman

Why SE ?

- ▶ To get the right software and to make the software right
- ▶ Complexity of software
 - Domain problem: Business Rule
 - Data size: Digital and Non Digital
 - Solution: Algorithm
 - Place or Sites



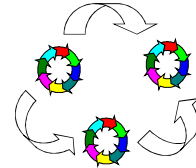
How should SE be applied ?

- ▶ There are 2 things to be considered in SE:
 - Product = Software:
 - Programs
 - Documents
 - Data
 - Process of how the software is build:
 - Management process
 - Technical process



Product of SE

- ▶ Product is obtained through stages of development = Software Development Life Cycle (SDLC)
- ▶ Examples of life cycles (SDLC):
 - Waterfall model
 - V model
 - Spiral model
 - Fountain model
 - Prototyping



Process of SE

- ▶ Management process includes:
 - Project management
 - Configuration management
 - Quality Assurance management



Process of SE (2)

- ▶ Technical process, described as methods to be applied in a particular stage of the s/w development life-cycle
 - Analysis methods
 - Design methods
 - Programming methods
 - Testing methods
- ▶ Technical methods are leading to paradigms



When should SE be applied ?

- Pre-project
- Project Initiation
- Project Realisation
- Software Delivery & Maintenance



Who is involved ?

- Manager
 - Project Manager
 - Configuration Manager
 - Quality Assurance Manager
- Software Developer:
 - Analyst
 - Designer
 - Programmer



Who is involved ?

- Support
 - Administration
 - Technical Support for Customer
 - Welfare



What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering.

* Software Engineering 7th ed, Ian Sommerville

What is the difference between software engineering and system engineering?

- ▶ System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.
- ▶ System engineers are involved in system specification, architectural design, integration and deployment.

* Software Engineering 7th ed, Ian Sommerville

What are the costs of software engineering?

- ▶ Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- ▶ Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- ▶ Distribution of costs depends on the development model that is used.

* Software Engineering 7th ed, Ian Sommerville

What are software engineering methods?

- ▶ Structured approaches to software development which include system models, notations, rules, design advice and process guidance.
- ▶ Model descriptions
 - Descriptions of graphical models which should be produced;
- ▶ Rules
 - Constraints applied to system models;
- ▶ Recommendations
 - Advice on good design practice;
- ▶ Process guidance
 - What activities to follow.

* Software Engineering 7th ed, Ian Sommerville

Work orientation*

Need to be :

Target Oriented
Process Oriented
Quality Oriented

Questions to ask and be answered

- ▶ What is the problem
- ▶ What entities needed to solve the problems
- ▶ How will the entity (solution) be realised
- ▶ What approach used for detecting errors
- ▶ How to support the entity over time in relation to corrections, adaptations, and enhancements.

Software phases

- ▶ *Definition Phase* : information, functions/performances, behaviours, interfaces, constraints, and validation criterias
- ▶ *Development Phase* : data structure, function implementation, interface character, design programming, and testing
- ▶ *Support Phase* : error correction, adaptation, enhancement, prevention/re-engineering

Requirement justifications *

- ▶ *Necessity* : Need to have, nice to have, need not to have
- ▶ *Feasibility* : technically, economically, operationally

A Common Process Framework

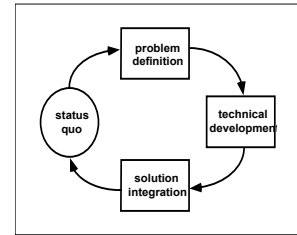
Framework activities
tasks
milestones & deliverables
QA checkpoints

Umbrella Activities

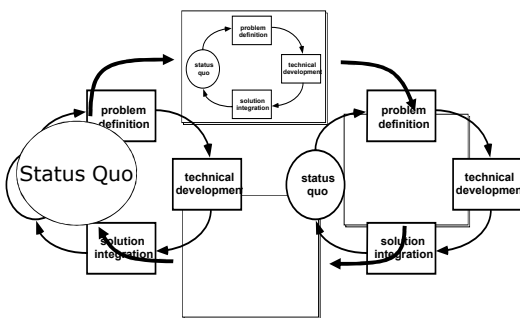
Umbrella Activities

- › Software project management
- › Formal technical reviews
- › Software quality assurance
- › Software configuration management
- › Document preparation and production
- › Reusability management
- › Measurement
- › Risk management

Problem Solving Loop Model



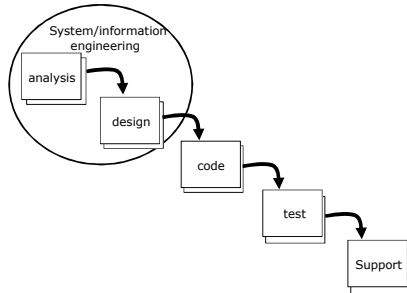
Problem Solving Loop Model



Problem Solving Loop Model

- › Difficult to contain activity neatly
- › Cross talks within and across stages
- › Tasks (and degree of rigor) for each activity will vary based on:
 - the type of project (an "entry point" to the model)
 - characteristics of the project
 - common sense judgment; concurrence of the project team

The Linear (Waterfall) Model



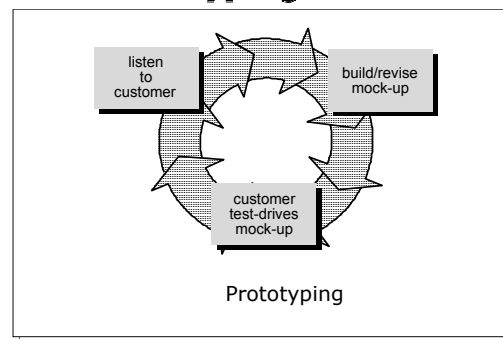
The Linear (Waterfall) Model

- › *Requirements Analysis* : information, function, behaviour, performance, and interface -> Deliverable : SRS Document
- › *Design* : data structure, software architecture, interface representation, algorithm. Deliverable : Software Config
- › *Code generation* : deliverables : program code
- › *Testing* : logical and functional
- › *Support* : error correction, adaptation, enhancement

The Linear (Waterfall) Model

- › *Changes can create confusion*
- › *Difficult to accommodate early stage natural uncertainty*
- › *Working version can only be obtained later on*

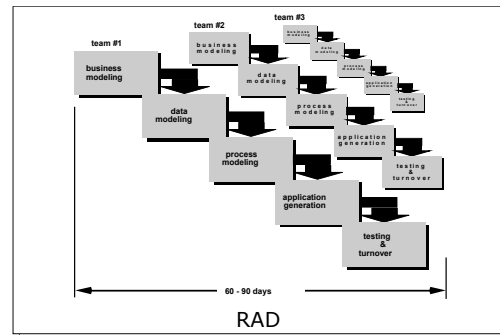
Prototyping Models



Prototyping Models

- ▶ *Prototype serves as 'first system'*
- ▶ *Can make customer demands of a few fixes to make a working product*
- ▶ *Implementation is compromised to get a prototype quickly.*
- ▶ *Need to agree that prototype only serves as mechanism for defining requirements*

RAD Models



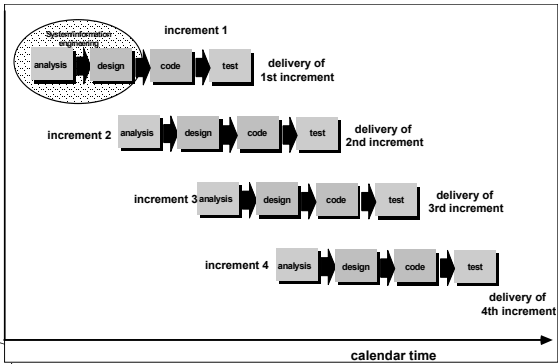
The RAD Model

- ▶ *Business Modelling* : BPM (what info needed, who generate it, who use it,etc), BPR
- ▶ *Data Modelling* : data objects, attributes, and relationship
- ▶ *Process Modelling* : add, modify, delete, retrieve data object
- ▶ *Application Generation* : reuse and /or create reusable components
- ▶ *Testing and turnover* : newly created components

The RAD Model

- ▶ *Requires substantial resources for large projects*
- ▶ *Requires commitment of developers and customers.*
- ▶ *Not suitable for high risk and performance applications (high degree of interoperability)*

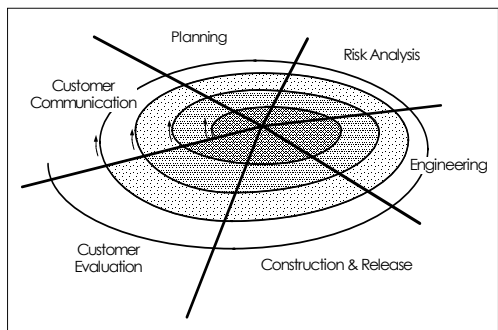
The Incremental Model



The Incremental Model

- ▶ First increment = core product
- ▶ Subsequent increments = additional features and functions
- ▶ Useful for under staff situation to meet dateline

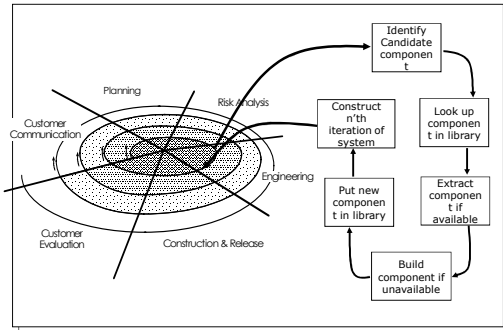
An Evolutionary (Spiral) Model



The Spiral Model

- ▶ Start from center clockwise
- ▶ Each pass = next stage of project : eg. Concept development, product development, product enhancement
- ▶ Realistic approach to large scale system development
- ▶ Better understanding and reaction to risk
- ▶ Requires expertise in risk assessment
- ▶ Not widely used

Component Based Model



The Component based Model

- process to apply when reuse is a development objective
- *Unified Modelling Language (UML) for defining components to use and interfaces that connect the components*

The 4th Generation Techniques

- Ability to specify software characteristics at high level
- Includes : code generation, report generation, data manipulation, HTML generation, etc
- Offers credible solutions to software problems
- Reduction amount of design and analysis
- Requires more analysis, design, and testing of software

Still Other Process Models

- Concurrent process model—recognizes that different part of the project will be at different places in the process
- Formal methods—the process to apply when a mathematical specification is to be developed
- Cleanroom software engineering—emphasizes error detection *before* testing

The Primary Goal: High Quality

Remember:

High quality = project timeliness

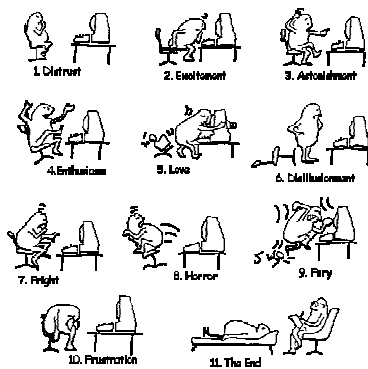
Why?

Less rework!

CMM-I (Capability Maturity Model – Integrated)

*

- ▶ *Initial* : ad hoc process, sometimes chaotic, success depends on individuals
- ▶ *Repeatable* : basic PM to track cost, schedule and functionality, posses necessary process disiplin to repeat earlier success.
- ▶ *Defined* : process are documented, standardized and integrated
- ▶ *Managed* : detailed measures and product quality are collected. Both process and products are understood and controlled using detailed measures.
- ▶ *Managed* : detailed measures and product quality are collected. Both process and products are understood and controlled using detailed measures. Success is probable
- ▶ *Optimizing* : continuous process trough feedback. Success is always



What is CASE (Computer-Aided Software Engineering)

- ▶ Software systems that are intended to provide automated support for software process activities.
- ▶ CASE systems are often used for method support.
- ▶ Upper-CASE
 - Tools to support the early process activities of requirements and design;
- ▶ Lower-CASE
 - Tools to support later activities such as programming, debugging and testing.

* Software Engineering 7th ed, Ian Sommerville

What are the attributes of good software?

- ▶ The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- ▶ Maintainability
 - Software must evolve to meet changing needs;
- ▶ Dependability
 - Software must be trustworthy;
- ▶ Efficiency
 - Software should not make wasteful use of system resources;
- ▶ Acceptability
 - Software must be accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

** Software Engineering 7th ed, Ian Sommerville*

What are the key challenges facing software engineering?

- ▶ Heterogeneity
 - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- ▶ Delivery
 - Developing techniques that lead to faster delivery of software;
- ▶ Trust
 - Developing techniques that demonstrate that software can be trusted by its users.

** Software Engineering 7th ed, Ian Sommerville*

Professional and ethical responsibility

- ▶ Software engineering involves wider responsibilities than simply the application of technical skills.
- ▶ Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- ▶ Ethical behaviour is more than simply upholding the law.

** Software Engineering 7th ed, Ian Sommerville*

Issues of professional responsibility

- ▶ Confidentiality
 - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- ▶ Competence
 - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

** Software Engineering 7th ed, Ian Sommerville*

Issues of professional responsibility

- ▶ Intellectual property rights
 - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- ▶ Computer misuse
 - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

* Software Engineering 7th ed, Ian Sommerville

ACM/IEEE Code of Ethics

- ▶ The professional societies in the US have cooperated to produce a code of ethical practice.
- ▶ Members of these organisations sign up to the code of practice when they join.
- ▶ The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

* Software Engineering 7th ed, Ian Sommerville

Code of ethics - preamble

- ▶ Preamble
 - The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.
 - Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

* Software Engineering 7th ed, Ian Sommerville

Code of ethics - principles

- ▶ PUBLIC
 - Software engineers shall act consistently with the public interest.
- ▶ CLIENT AND EMPLOYER
 - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- ▶ PRODUCT
 - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

* Software Engineering 7th ed, Ian Sommerville

Code of ethics - principles

- ▶ JUDGMENT
 - Software engineers shall maintain integrity and independence in their professional judgment.
- ▶ MANAGEMENT
 - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- ▶ PROFESSION
 - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

* Software Engineering 7th ed, Ian Sommerville

Code of ethics - principles

- ▶ COLLEAGUES
 - Software engineers shall be fair to and supportive of their colleagues.
- ▶ SELF
 - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

* Software Engineering 7th ed, Ian Sommerville

Ethical Dilemmas

- ▶ Disagreement in principle with the policies of senior management.
- ▶ Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- ▶ Participation in the development of military weapons systems or nuclear systems.

* Software Engineering 7th ed, Ian Sommerville

Questions??



56