Software Engineering

Oleh :
Ir. I Gede Made Karma, MT

Software

- What is Software ?
- Software Characteristics
- Software Component
- Software Applications
- What's wrong with software development ?
- Software Myths

What is Software?

- Definitions:
  - *Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system* **(IEEE Standard Glossary of Software Engineering Terminology, 1990)**
- Software is designed and built by software engineers.
- Software engineers have a moral obligation to build reliable software that does no harm to other people.
- Software engineers view computer software, as being made up of the programs, documents, and data required to design and build the system.
- Software users are only concerned with whether or not software products meet their expectations and make their tasks easier to complete.
- Software is both a product and a vehicle for developing a product.
- Currently, most software is still custom-built.

Software Characteristics

- Unique product (no series production)
- Does not wear out
- Invisible
- Flexible, therefore easy (**!?**) to modify
- A young technology that is not yet mature
- Limits of complexity constantly extended
- **Linked with hardware**

Software Component

- Build using a programming language
- Important characteristic: reusability

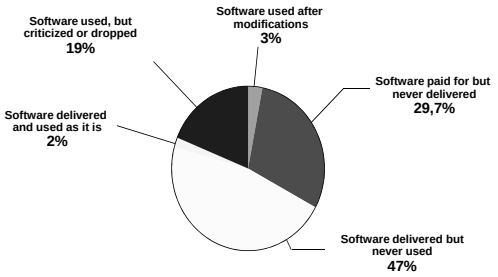
Software Applications

- System software
- Real-time software
- Business software
- Engineering and scientific software
- Embedded software
- Personal computer software
- Web-based software
- Artificial intelligence software

What's wrong with the S/W Development ?

- Software crisis
  - Software failures receive a lot more publicity than software engineering success stories.
  - The software crisis predicted thirty years ago has never materialized and software engineering successes outnumber the failures.
  - The problems that afflict software development are associated more with how to develop and support software properly, than with simply building software that functions correctly.
- Software problems
- Decision: Software Engineering

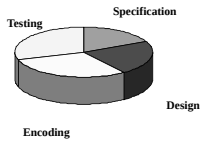
Software Problems



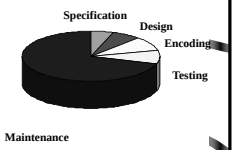
1982: Nine DOD contracts amounting to $6.8 million
(source: GAO, quoted in CMU/SEI-**93**-EM-8)

Software Problems (2)

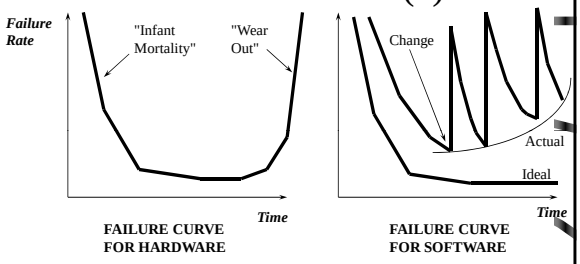
Distribution of effort : what is believed



Distribution of effort: what happens



Software Problems (3)



FAILURE CURVE FOR HARDWARE

FAILURE CURVE FOR SOFTWARE

** Software Engineering, Module 1, Richard Conn, University of Cincinnati, May 1993*

Software Myths: Clients' point of view

Myths:

- A general statement of objectives is enough to get going. Fill in the details later.
- Project requirements continually change, but change can be easily accommodated because software is flexible.

Reality:

- Poor up-front definition of the requirements is *THE* major cause of poor and late software.
- Cost of the change to software in order to fix an error increases dramatically in later phases of the life of the software.

Software Myths: Developers' point of view

Myths:

- Once a program is written and works, the developer's job is done.
- Until a program is running, there is no way to assess its quality.
- The only deliverable for a successful project is a working program.

Reality:

- 50%-70% of the effort expended on a program occurs after it is delivered to the customer.
- Software reviews can be more effective in finding errors than testing for certain classes of errors.
- A software configuration includes documentation, regeneration files, test input data, and test results data.

Software Myths: Management's point of view

Myths:

- Books of standards exist in-house so software will be developed satisfactorily.
- Computers and software tools that are available in-house are sufficient.
- We can always add more programmers if the project gets behind.

Reality:

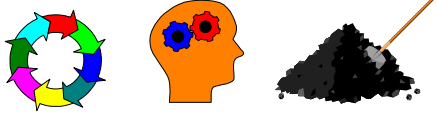
- Books may exist, but they are usually not up to date and not used.
- CASE(**) tools are needed but are not usually obtained or used.
- "Adding people to a late software project makes it later." -- *Brooks*

Software Engineering

- What is SE ?
- Why SE ?
- How should SE be applied ?
- Product of SE
- Process of SE
- When should SE be applied ?
- Who is involved ?

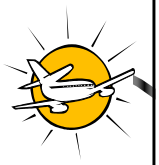
What's SE ?

- *Software Engineering* adalah teknologi yang harus digunakan oleh setiap orang yang akan membangun software, dengan melalui serangkaian proses, menggunakan sekumpulan metode dan alat bantu (*tools*) (Pressman, 1997)



Why SE ?

- Untuk mendapatkan software yang benar dan untuk membuat software menjadi benar
- Software adalah sesuatu yang kompleks dalam hal:
  - *Domain problem: Business Rule*
  - *Data size: Digital and Non Digital*
  - *Solution: Algorithm*
  - *Place or Sites*

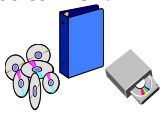


Why SE ? (2)

- Software harus benar (***correct)***:
  - Berdasarkan *business rule*
  - Sejalan dengan segala sesuatu dan semua pihak yang terkait
- Pembangunan software harus dikelola dengan baik untuk memelihara kebenarannya (*correctness*)

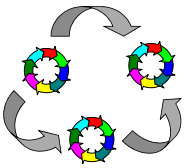
How should SE be applied ?

- 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

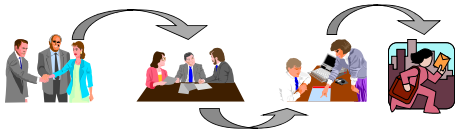
- Management process includes:
  - Project management
  - Configuration management
  - Quality Assurance management

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 ? (2)

- Support
  - Administration
  - Technical Support for Customer
  - Welfare

Software Process

What is Behind the Names???

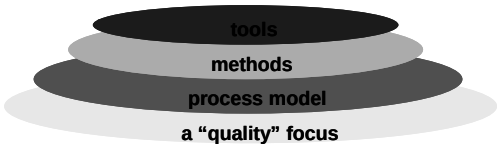
The IEEE Definition of Software Engineering:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)

- Software engineering:
  - **Software process**
  - **Technical Methods**
  - **Automated Tools**

A Layered Technology

Software Engineering



What Does Software Engineering Do???

- **Definition (What ???)**
  - What information is to be processed,
  - What function and performance are desired,
  - What system behavior can be expected,
  - What interface are to be established,
  - What design constraints exist, and
  - What validation criteria are required to be define a successful system.
- System or information engineering (10)
- Software project planning (3, 5, 6 and 7)
- Requirements analysis (11, 12 and 21)

What Does Software Engineering Do???

- **Development (How ???)**
  - How data are to be structured,
  - How function is to be implemented within a software architecture,
  - How procedural details are to be implemented,
  - How interfaces are to be characterized,
  - How the design will be translated into a programming language, and
  - How testing will be performed.
- Software Design (13-16 and 22)
- Code Generation and Software Testing (17, 18 and 23)

What Does Software Engineering Do???

- **Maintenance (Change)**
  - Associated with error correction,
  - Adaptions required as the software's environment evolves, and
  - Changes due to enhancements brought about by changing customer requirements.
- Correction : change the software to correct defects.
- Adaptation : modification to the software to accommodate changes to its external environment.
- Enhancement : extends the software beyond its original functional requirements.
- Prevention : makes changes to computer programs so that they can be more easily corrected, adapted and enhanced.
  - Software reengineering.

What Causes SW Projects to Fail?

- Unrealistic plans, based on optimistic estimates
- Ineffective tracking of performance
- Volatile requirements
- Risks

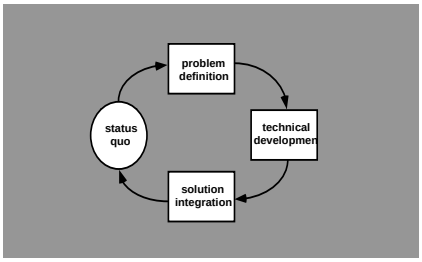
But, Why do We Let it Happen?

- People tend to be risk averse when there is potential of loss
- People are unduly optimistic in their plans and forecasts
- People prefer to use intuitive judgment rather than quantitative models

Controlling Human Nature

- Documenting the way work is performed
- Provide guidance and quantifiable criteria where possible
- Record decisions and the data used to make them
- Analyze the results and improve the process where possible
- Learn - individually and organizationally

Process as Problem Solving



The Process Model: Adaptability

- the framework activities will always be applied on every project ... BUT
- the 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 Primary Goal: High Quality

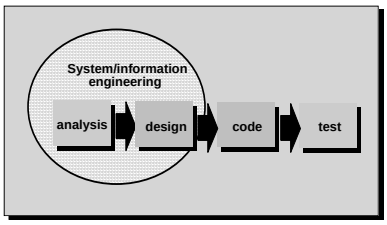
Remember:

High quality = project timeliness

Why?

Less rework!

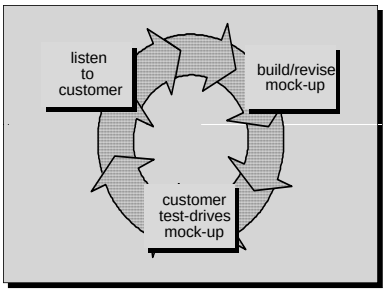
The Linear Model



Linear Models - Problems

- Change handling during the process
- Requires that all requirements are stated clearly at the beginning of the process
- Working version is delivered at the end of the process cycle; mistakes at earlier stages may be disastrous
- “Blocking States”

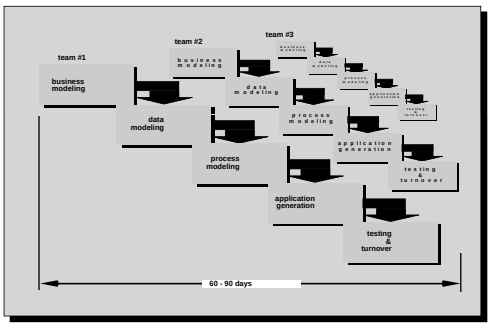
Iterative Models - Prototyping



Prototyping - The Problems

- There is a “working version” of software before the requirements for the overall quality and maintainability are satisfied.
- Implementation compromises, made to create a quick “working version” often become a part of the final version.

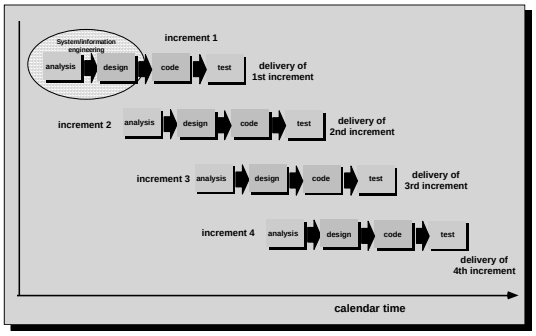
Iterative Models - RAD



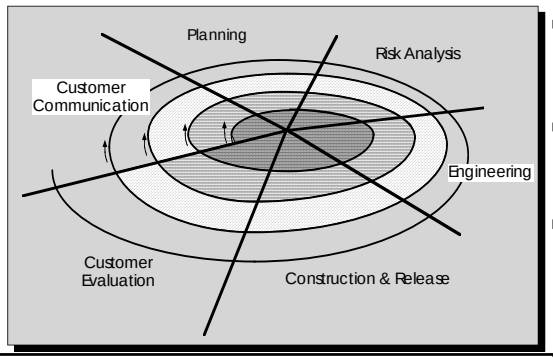
RAD - The Problems

- For large, but scalable projects, requires significant human resources
- Requires customers and developers willing to work in a rapid development environment
- If the requirements can not be modularized, this approach may not be suitable
- If fine-tuning is needed, this approach may not be suitable

Evolutionary Models - The Incremental Model



Evolutionary Models - Spiral Model



Spiral Model – The Lifecycle of SW Product

- Concept Development Projects
- New Product Development Projects
- Product Enhancement Projects
- Product Maintenance Projects

Spiral Model - Characteristics

Advantages

- application in large systems and software
- used well as a risk reduction mechanism

Disadvantages

- controllability (demands high risk assessment and expertise)
- has not been applied as much (little history)

Still Other Process Models

- **Formal methods**—the process to apply when a mathematical specification is to be developed
- **Cleanroom software engineering**—emphasizes error detection *before* testing
- **4GT** (fourth generation techniques) — automatic code generation