Session #2

SOFTWARE ENGINEERING PROCESS

Software Engineering in Practical Approach: an Experience Sharing

Jurusan Teknik Informatika Politeknik Elektronika Negeri

ITS SURABAYA, Sept., 20-22, 2011

BAHTIAR H. SUHESTA

IT Practitioner, Writer-preneur, and Founder of An-Nabwah Group

ITS Institut Teknologi Sepuluh Nopember

ANNABWAH

Topics

- 7 Principles
- The Software Process
- The Process Flow
- The Software Process Model
- The Types of Software
- Discussion

7 Principles *)

- The Reason It All Exists
- Keep It Simple
- Maintain the Vision
- What You Produce, Others will Consume
- Be Open to the Future
- Plan Ahead for Reuse
- Think!

)* Adapted from David Hooker.

The Reason It All Exists

- A software system exists for one reason: *to provide value to its users.* All decisions should be made with this in mind.
- Before specifying a system requirement, before noting a piece of system functionality, before determining the hardware platforms or development processes, ask this question: "Does this add real value to the system?"
  - If the answer is "No," then DON'T DO IT!
- All other principles support this one.

Keep It Simple

- Software design is not a haphazard process. There are many factors to consider in any design effort.
- *All design should be as simple as possible, but no simpler.*
- This facilitates having a more easily understood and easily maintained system.
- The more elegant designs are usually the more simple ones. But, simple does not mean "quick and dirty."

Maintain the Vision

- *A clear vision is essential to the success of a software project..*

What u Produce, Others will Consume

- *Always specify, design, and implement knowing someone else will have to understand what you are doing.*
- The audience for any product of software development is potentially large.
- Specify with an eye to the users. Design, keeping the implementers in mind. Code with concern for those that must maintain and extend the system.
- Someone may have to debug the code you write, and that makes them a user of your code.
- Making their job easier adds value to the system.

Be Open to the Future

(thd system, teknologi & perubahan requirements)

- A system with a long lifetime has more value.
- Today, software lifetimes are typically measured in months. True “industrial-strength” software must endure far longer.
- Systems must be ready to adapt to these and other changes. And systems do this successfully are those that have been designed this way from the start.
- *Never design yourself into a corner.* Always ask “what if” and prepare for all possible answers. This could very possibly lead to the reuse of an entire system.

Plan Ahead for Reuse → Library Man (keseluruhan s/w, komponen, algoritma, metode)

- Reuse saves time and effort.
- The reuse of code and designs has been proclaimed as a major benefit of using object-oriented technologies.
- There are many techniques to realize reuse at every level of the system development process. . .
- *Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated.*

Think!

- *Placing clear, complete thought before action almost always produces better results.*
- When clear thought has gone into a system, value comes out.
- Applying the first six principles requires intense thought, for which the potential rewards are enormous.

How It All Start

- When you work to build a product or system, it's important to go through a series of predictable steps.
- That is a road map that helps you create a timely, high-quality result.
- The road map that you follow is called a "software process."

The Software Process

- At a detailed level, the process that you adopt depends on the software that you’re building.
- One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a website.

The Software Process

- There are a number of software process assessment mechanisms that enable organizations to determine the “maturity” of their software process.
- However, the quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.

The Software Process

- A software process is a framework for the *activities*, *actions*, and *tasks* that are required to build high-quality software.
- The software process, or sometimes is called as Software Development Process or Software Development Life Cycle (SDLC), is often resulted from a long-time best practice process of the company.

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands.

Pemilihan Software Process Model

- Tergantung banyak hal dengan mempertimbangkan kelebihan dan kelemahan masing-masing model
- Salah satu yang mesti dipertimbangkan adalah jenis aplikasi / perangkat lunak yang akan dibangun.

Project-based vs Product-based

Project-Based	Product-Base
User-driven	Developer-driven
Produce-to-order	Produce-to-stock
Special product for user(s)	Product mass
Tailor made	Ready made
Project cost	Investment cost
Limited time and cost	Unlimited time and cost
Huge customization	Limited customization
Berakit-rakit ke hulu dan juga ke tepian ☺	Berakit-rakit ke hulu, berenang ke tepian

Project or Product-based?

- Pilih *Project-based*, jika:
  - Tidak punya banyak modal (investasi)
  - Ada pesanan dari user / kustomer; dan banyak kustomer semisal (berpotensi *repetitive-order*)
  - Perangkat lunak berpotensi bisa dibuat produk
- Pilih *Product-based*, jika:
  - Punya modal untuk investasi
  - Memiliki ide perangkat lunak yang prospektif dan bersifat *product-mass* ke depan
  - Berdamai dengan waktu
  - Lebih rawan dijiplak/dicopy (copy protect, HAKI)

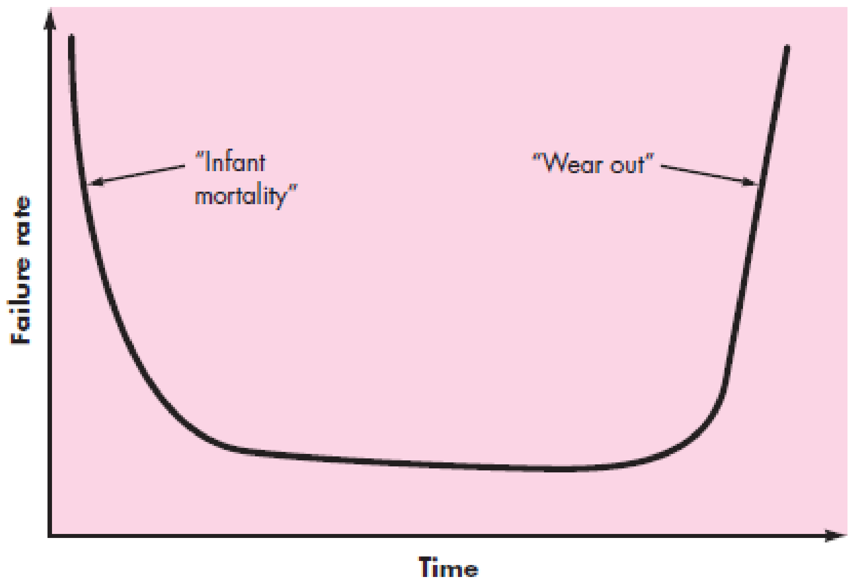
Pure-Software vs HW-Embedded

Pure-Software	Software w/ HW-Embedded
Only Software installed in the infrastructure	Software integrated with hardware should be installed
Virtual deliverable	Physical deliverable
Not-easy to deliver (user's perspective)	More easy to deliver (user's perspective)
Software problem	Software & hardware problem
Not-easy in pricing	More-easy in pricing
Software security	Software & Hardware security

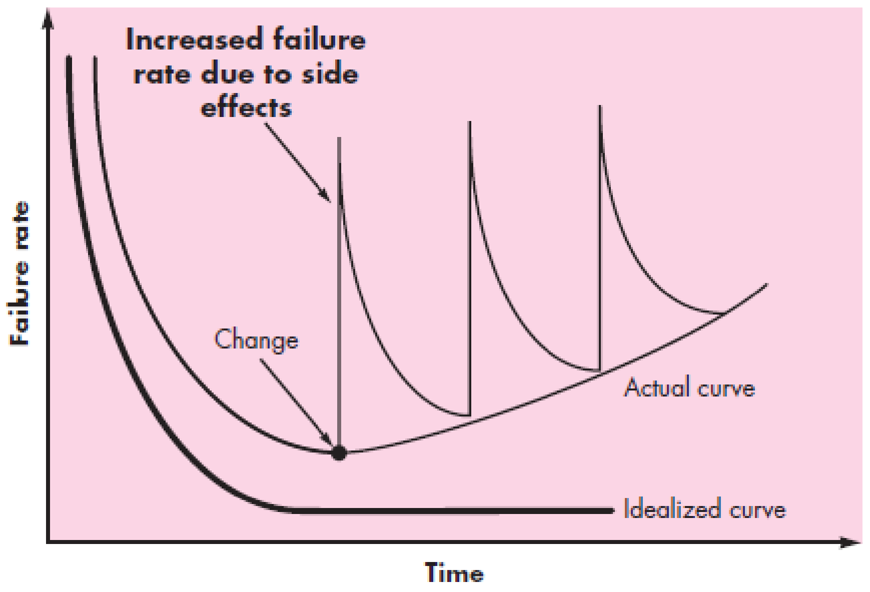
Pure-Software or HW-Embedded?

- Depend on the competency → HW-Embedded needs HW-experts.
- HW-problem sometime cannot be predicted.
- HW-replacement is cost-expensive.
- The characteristic of their failure is different.
- HW has spare-parts. But SW has not.

Pure-Software or HW-Embedded?



Failure curve for hardware



Failure curve for software

Discussion