Supplementary Slides for

Software Engineering: A Practitioner's Approach, 6/e

Part 1

Software Engineering: A Practitioner's Approach, 6/e

Chapter 1
Software and Software Engineering

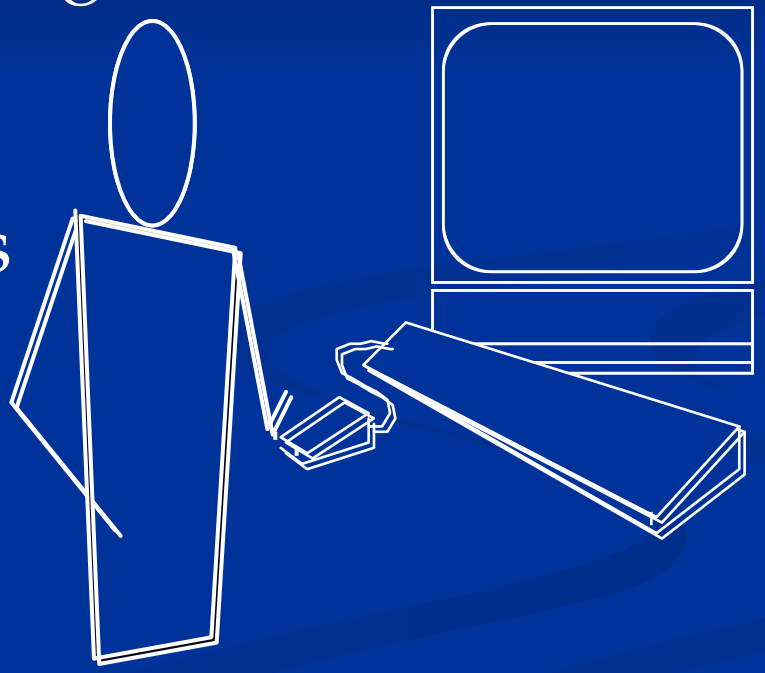
Software's Dual Role

- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects communications (e.g., networking software)
  - Helps build other software (e.g., software tools)

What is Software?

Software is a set of items or objects that form a "configuration" that includes

- programs
- documents
- data ...

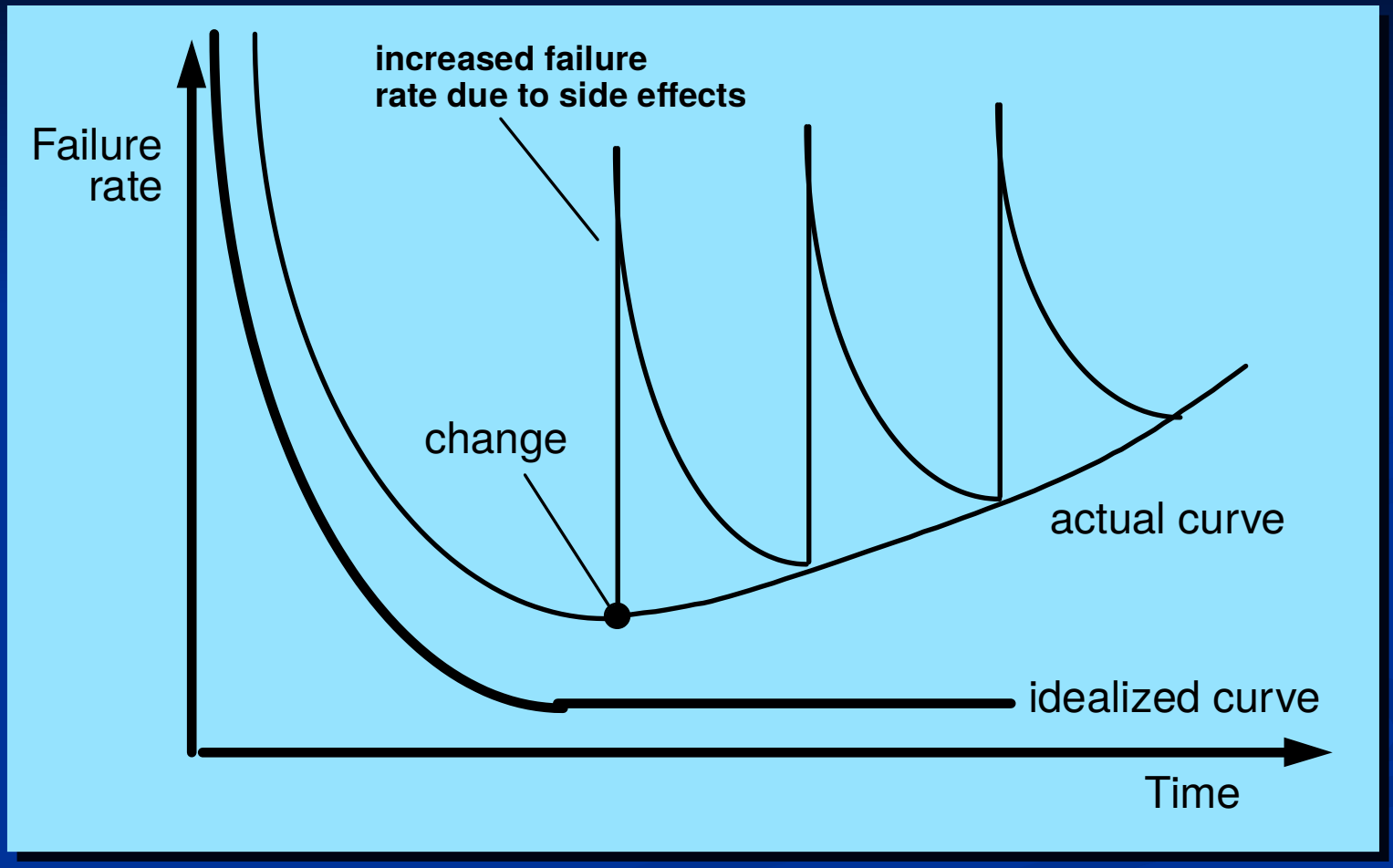


What is Software?

- software is engineered
- software doesn't wear out
- software is complex

- Software is developed or engineered , it is not manufactured in the classical sense.
- Software doesn't wear out.
- Although the industry is moving toward component-based construction, most software continues to be custom built.

Wear vs. Deterioration



Software Applications

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Software—New Categories

- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—"free" source code open to the computing community (a blessing, but also a potential curse!)
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

Legacy Software

Why must it change?

- software must be adapted to meet the needs of new computing environments or technology.
- software must be enhanced to implement new business requirements.
- software must be extended to make it interoperable with other more modern systems or databases.
- software must be re-architected to make it viable within a network environment.

Generic process framework for Software engineering

- **Communication :** collaborate with customers for requirement gathering.
- **Planning:** software project plan , tasks, risks, resourses & schedules.
- **Modeling** : for better understanding requirements.
- **Construction**: coding & testing.
- **Deployment:** software delivered to customer & f/b evaluated.

Umbrella Activities

- **Software project Tracking & control. : Against Project Plan.**
- **Risk management :** may affect outcome of product.
- **Software quality assurance :** ensure quality of product.
- **Technical reviews:** try to uncover errors before they propagate.

Umbrella Activities

- **Work product preparation and production :** documents, logs , forms & lists.
- **Measurement:** defines & collects process, project & product measures.
- **Software configuration management :** manage the effects of change.
- **Reusability management:** achieve reusability of components.

Software Engineering Practices

1. **Understand the problem**(Communication & analysis)
2. **Plan a solution** (Modeling & software design)
3. **Carry out the plan**(Code generation)
4. **Examine the result for accuracy** (testing & quality assurance)

Understand the problem

- Who has a stake in the solution to the problem.?
- What are the unknowns ? (Data, functions, features)
- Can the problem be compartmentalized?
- Can the problem be represented graphically?

Plan a solution

- Have you seen similar problems before?
- Has a similar problem been solved?
- Can subproblems be defined?
- Can you represent a solution that leads to effective implementation?

General principles

1. The Reason It All Exists : to provide value to users.
2. Keep It Simple Stupid:
3. Maintain the Vision : Conceptual integrity is important.
4. What you produce others will consume.
5. Be open to the future.
6. Plan ahead for reuse.

Software Myths

- **Managements Myths**

- ***We already have a book that's full of standards for building s/w .*** Wont that provide my people with every thing they need to know.

- If we get behind schedule , we can add more programmers & catch up.

- If I decide to outsource the s/w project to a third party , I can just relax & let that firm build it.

Customer Myths

- A general statement of objectives is sufficient to begin writing programs- we can fill in the details later.

- Software requirements continually change , but change can be easily accommodated because s/w is flexible

Practioner's Myths

- Once we write the program and get it to work , our job is done.
- Until I get the program running I have no way of assessing its quality.
- The only deliverable work product for a successful project is the working program.
- Software engg will make us create voluminous & unnecessary documentation & will slow us down.