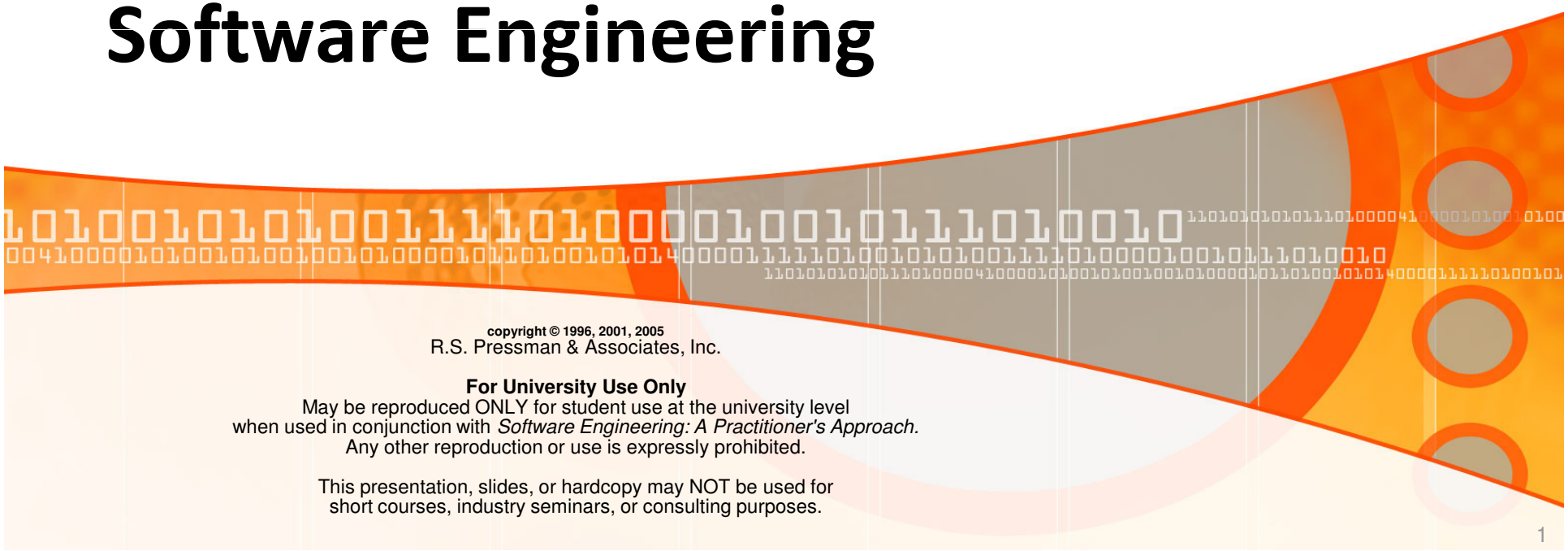
Part 1
Introduction to Software Engineering



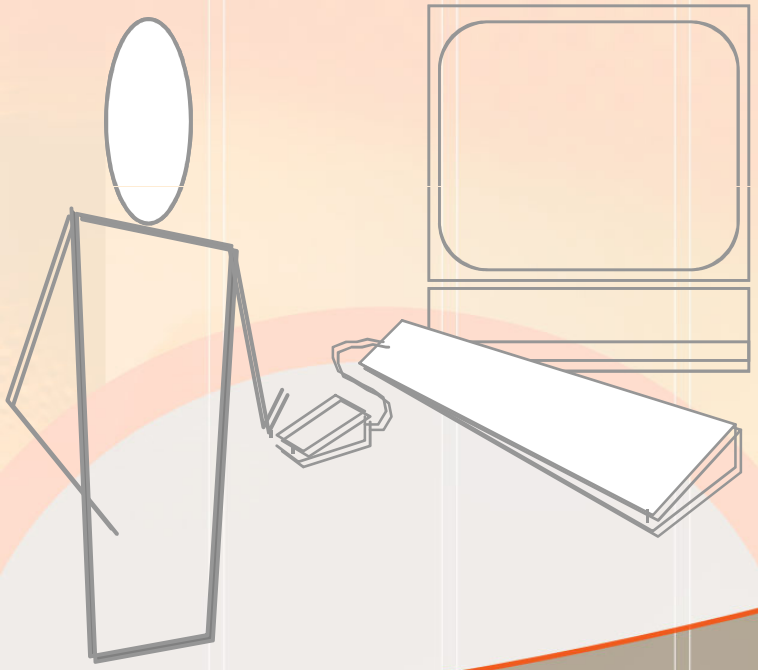
Software's Dual Role

- *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

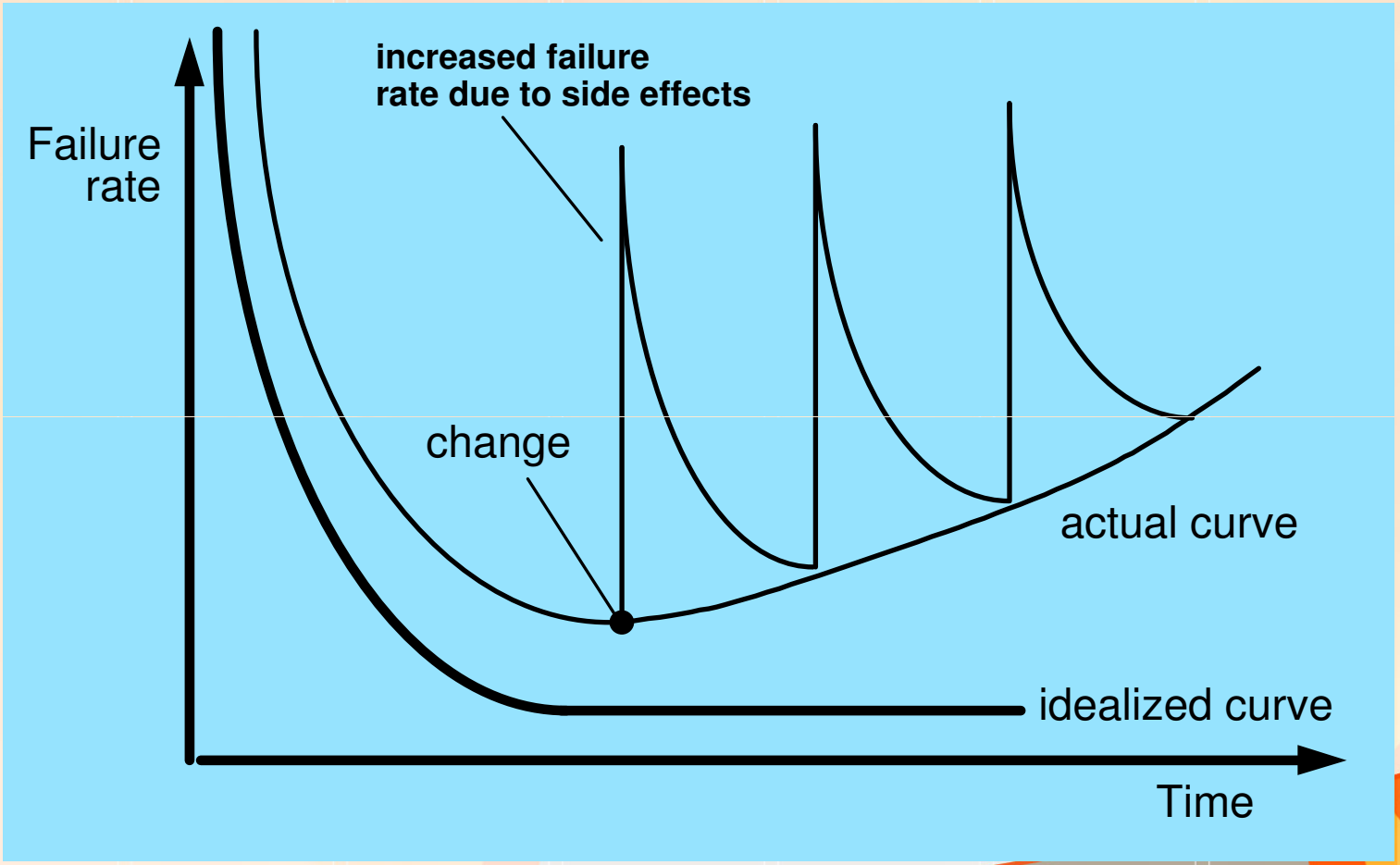
- programs
- documents
- data ...



What is Software?

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

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!)*
- *Also … (see Chapter 32)*
  - *Data mining*
  - *Grid computing*
  - *Cognitive machines*
  - *Software for nanotechnologies*

Legacy Software

- *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.*

Why must it change?

E-Type Systems

- *E-Type Systems:*
  Software that has been implemented in a real-world computing context and will therefore evolve over time

Software Evolution

- *The Law of Continuing Change (1974): E-type systems must be continually adapted else they become progressively less satisfactory.*
- *The Law of Increasing Complexity (1974): As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.*
- *The Law of Self Regulation (1974): The E-type system evolution process is self-regulating with distribution of product and process measures close to normal.*
- *The Law of Conservation of Organizational Stability (1980): The average effective global activity rate in an evolving E-type system is invariant over product lifetime.*
- *The Law of Conservation of Familiarity (1980): As an E-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behavior to achieve satisfactory evolution.*
- *The Law of Continuing Growth (1980): The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.*
- *The Law of Declining Quality (1996): The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.*
- *The Feedback System Law (1996): E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.*

Software Myths

- *Affect managers, customers (and other non-technical stakeholders) and practitioners*
- *Are believable because they often have elements of truth,*
- *but …*
- *Invariably lead to bad decisions,*
- *therefore …*
- *Insist on reality as you navigate your way through software engineering*

Management Myths

- *"We already have a book of standards and procedures for building software. It does provide my people with everything they need to know …"*
- *"If my project is behind the schedule, I always can add more programmers to it and catch up …"*
  (a.k.a. "The Mongolian Horde concept")
- *"If I decide to outsource the software project to a third party, I can just relax: Let them build it, and I will just pocket my profits …"*

Customer Myths

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

- *"Project requirements continually change but this change can easily be accommodated because software is flexible …"*

Practitioner's Myths

- *"Let's start coding ASAP, because 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 engineering is baloney. It makes us create tons of paperwork, only to slow us down …"*