
Advanced Software Engineering **(CS6401)**

Autumn Semester (2022-2023)

Dr. Judhistir Mahapatro
Department of Computer Science and
Engineering
National Institute of Technology Rourkela

Introduction to Software and Software Engineering (Lecture-1)

Organization of this Lecture:

- ▶ Nature of software
- ▶ Nature of software projects
- ▶ What is Software Engineering?
- ▶ Programs vs. Software Products
- ▶ Software Process
- ▶ Introduction to Life Cycle Models
- ▶ Summary

Software

- **Instructions** (computer programs) that when executed provide desired features, function, and performance
- **Data structures** that enable the programs to adequately manipulate information, and
- **Descriptive information** in both hard copy and virtual forms that describes the operation and use of the programs.

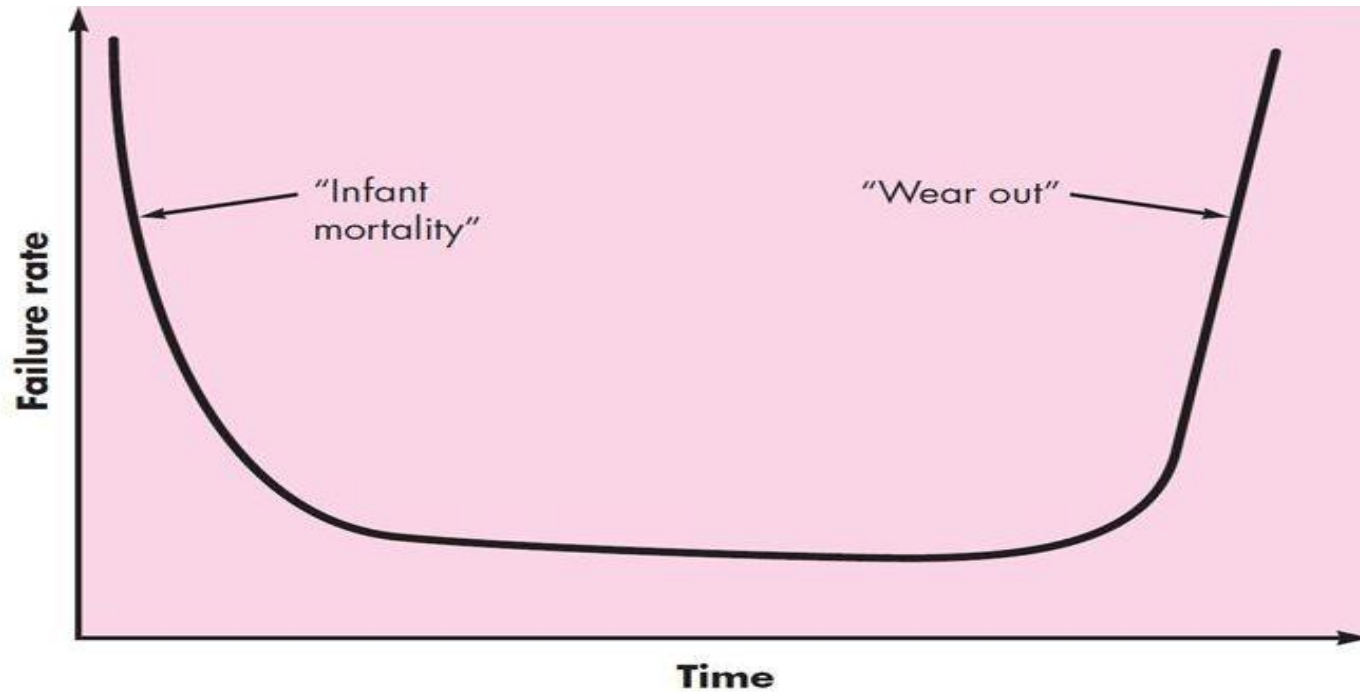
Characteristics

- **Software is developed or engineered; it is not manufactured in the classical sense.**
 - Similarity exist between software development and hardware manufacturing, but two activities are different.
 - High quality is achieved through good design, but hardware manufacturing introduces quality problems that are easily corrected for software
 - Relationship between people applied and work accomplished is entirely different
 - Both activities require the construction of a product, but approaches are different
 - Software costs are concentrated in engineering; software project management is different from hardware

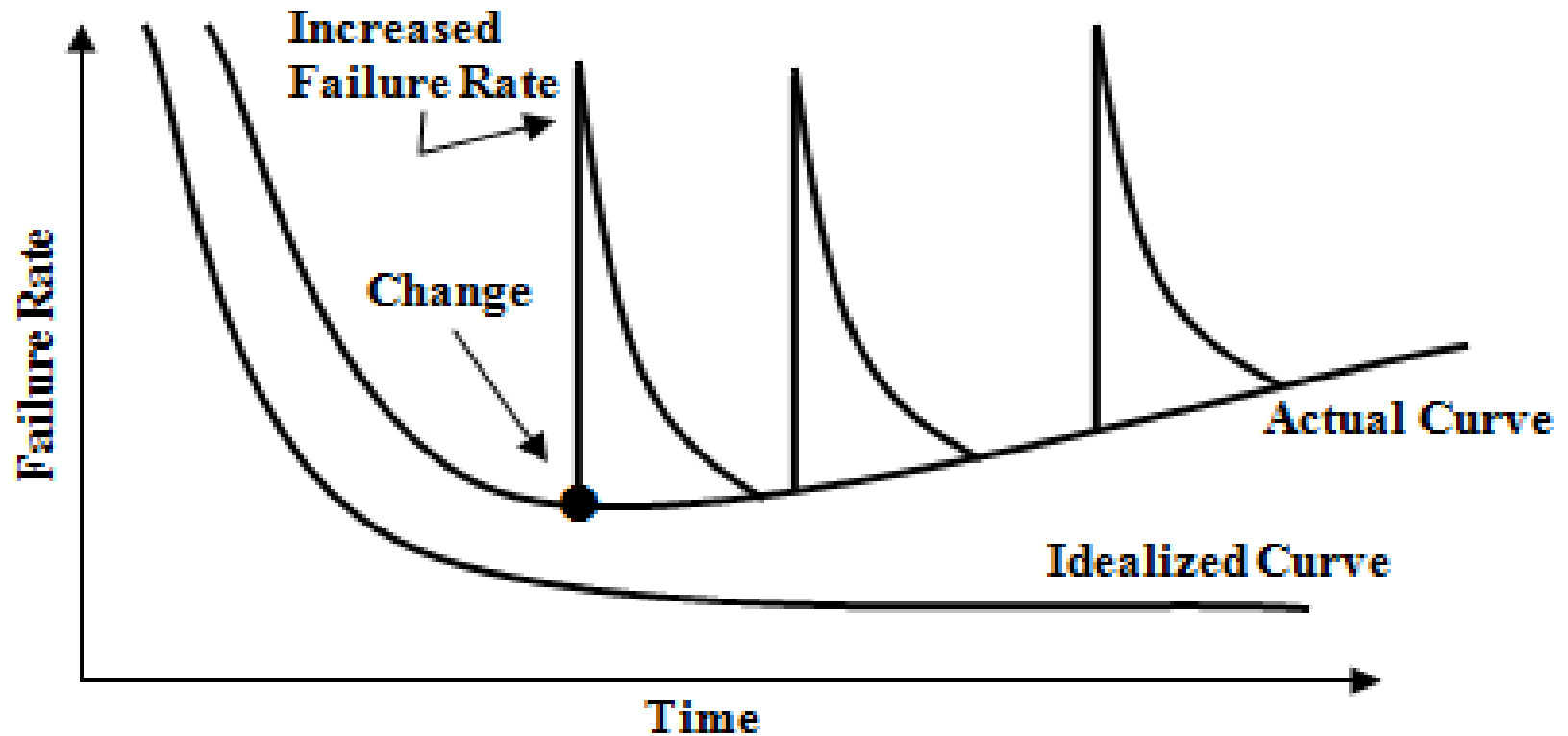
Characteristics

- **Software does not wear out.**
 - Hardware failure rate resembles **Bathtub curve**
 - Relatively high failure rates early in its life
 - Often attributable to design or manufacturing defects
 - Failure rate drops to steady-state level due to correction of defects
 - **Failure rate rises** as hardware components suffer from cumulative **effects of dust, vibration, and temperature** etc.
 - But software is not susceptible to those environmental maladies.
 - Failure curves of hardware and software are given in the next slide.

Failure curve for hardware



Failure curve for software



- Software does not wear out. But it does deteriorate.
- There are no spare parts.

Characteristics

- **Although the industry is moving toward component-based construction, most software continues to be custom built**
 - Reusable components have been created so that engineer can concentrate on the truly innovative elements of a design
 - Example: Screws used by mechanical engineers
 - It has only begun to be achieved on a broad scale.

Nature of software projects

- Ubiquitous, used in variety of applications – Business, engineering and scientific applications
 - System Software,
 - Application Software,
 - Scientific Software,
 - Embedded Software,
 - Web applications etc.
- Simple to complex, public usage (example, railway ticket reservation system), single function to enterprise-wide (example, payroll), one location to distributed, batch or real-time, informational to mission-critical etc.



Major challenges in large projects

- Developing large/complex software application is very challenging
 - Effort intensive
 - High cost
 - Long development time
 - Changing needs for users
 - High risk of failure, user acceptance, performance, maintainability



Successful software system

- Software development projects have not been always successful (compared to other engineering disciplines).
- When do we consider a software application successful?
 - Development completed
 - It is useful
 - It is usable
 - It is used



Reasons for failure

- Schedule Slippage – not released at appropriate time
- Cost over-runs
- Does not solve user's problem
- Poor quality of software (may not be maintainable)

Ad hoc software development results in such problem:

- No planning of development work (e.g., no milestones are defined)
- Deliverable are not defined
- Technical incompetency of developers
- No control over review (Review the progress of the system because enough money is spent in it and keep updated with the recent trends)
- Poor understanding of the user requirements
- Poor understanding of the cost and effort by both developer and user

Engineering: other disciplines

- Large projects common and successfully done
 - building bridges and dams
 - Power plants, Missiles
- Engineering a solution:
 - to design and develop an artefact that meets specifications effectively, cost effectively and ensuring the quality using scientific principles



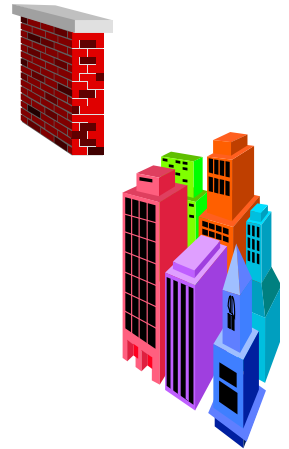
Engineering:

- Requires well-defined approach: repeatable, predictable
- Large projects requires managing the project itself
 - Manage people, money, equipment schedule
 - Scale makes big difference: compare building a house, 2-storeyed, or 100-storeyed
- Need a systematic approach and management
- Quality extremely important: relates failures, efficiency and usability
 - People willing to pay for quality.



What is Software Engineering?

- Engineering approach to develop software.
 - Building Construction Analogy.
- Systematic collection of past experience:
 - techniques,
 - methodologies,
 - guidelines.



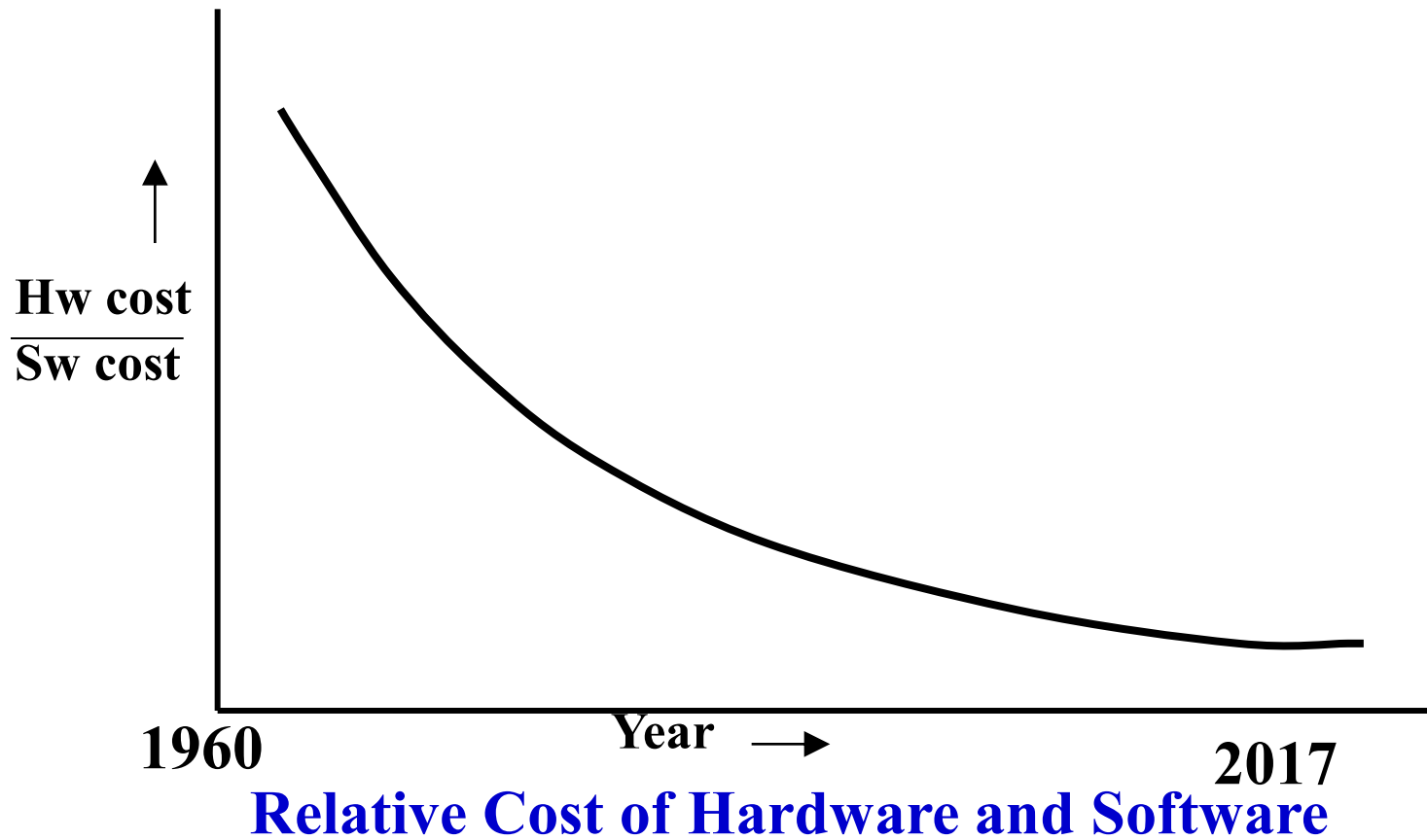
Engineering Practice

- Heavy use of past experience:
 - Past experience is systematically arranged.
- Theoretical basis and quantitative techniques provided.
- Many are just thumb rules.
- Trade-off between alternatives
- Pragmatic approach to cost-effectiveness

Software Crisis

- Software products:
 - fail to meet user requirements.
 - frequently crash.
 - expensive.
 - difficult to alter, debug, and enhance.
 - often delivered late.
 - use resources non-optimally.

Software Crisis (cont.)



Factors contributing to the software crisis

- Larger problems
- Lack of adequate training in software engineering techniques
- Increasing skill shortage
- Low productivity improvements

Software Crisis (cont.)

- Software Engineering appears to be available to tackle the software crisis.

Programs versus Software Products

• Usually small in size	• Large
• Author himself is sole user	• Large number of users
• Single developer	• Team of developers
• Lacks proper user interface	• Well-designed interface
• Lacks proper documentation	• Well documented & user-manual prepared
• Ad hoc development.	• Systematic development

Computer Systems Engineering

- Computer systems engineering:
 - encompasses software engineering.
- Many products require development of software as well as specific hardware to run it:
 - a coffee vending machine,
 - a mobile communication product, etc.

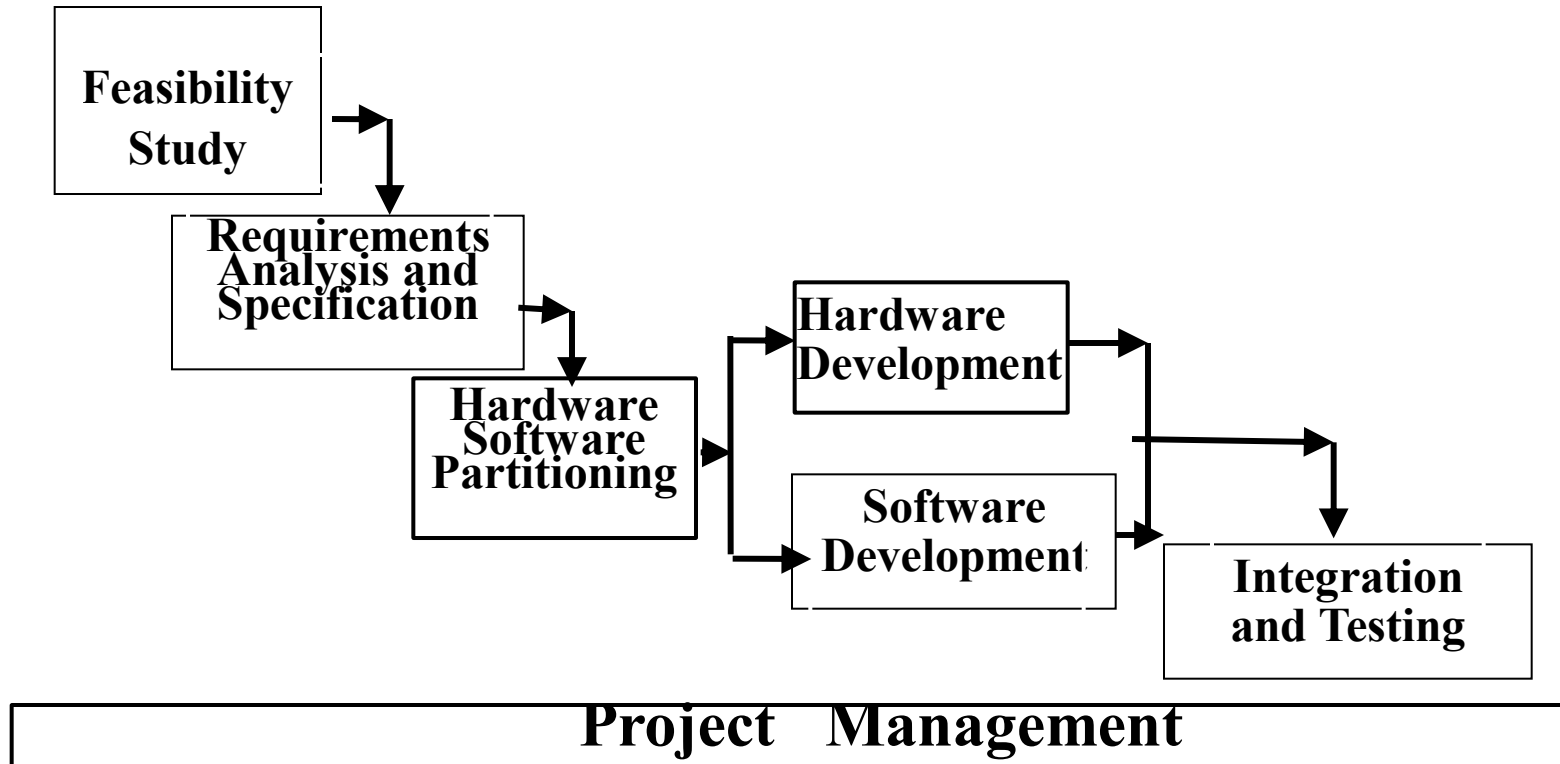
Computer Systems Engineering

- The high-level problem:
 - deciding which tasks are to be solved by software
 - which ones by hardware.

Computer Systems Engineering (Cont..)

- Often, hardware and software are developed together:
 - Hardware simulator is used during software development.
- Integration of hardware and software.
- Final system testing

Computer Systems Engineering (CONT.)



Large projects:

- Deliverables: architecture, model, structure diagram, electrical cabling layouts
- Standards, regulations, conventions need to be followed (contribute towards success of the software)
- Continuous monitoring is in place to have progress in its developments
- It is difficulty to assess the progress because it does not have any physical dimensions (in case building you can see the storeyed developed to till date)



Software Projects

- Software is different from other products
- Cost of production concentrated in development (once it is developed its copy will be installed in various places)
- Maintenance consists of making corrections and enhancing or adding functions
- Progress in development is difficult to measure



Apply engineering approach:

- Hence planning and control even more important in software development
- Attempt estimate cost/effort
- Plan and schedule work
- Involve user in defining requirements
- Identify stages in development
- Define clearly the milestones so that the progress can be measured
- Schedule reviews both for control and quality
- Define deliverables
- Plan extensive testing

Note: We need to put all of these in the form of process and we use this as an engineering approach.



Job of software developer is difficult

- Dealing with users
 - ill-defined requirements (what they expect from the software is not clearly defined)
 - Concern with ease-of-use and response time
- Dealing with technical people
 - Concerned with coding, databases, file structures etc
- Dealing with management
 - Concerned with return on their investment
 - Cost-benefit analysis
 - Schedule

Note: Balance the expectations of people who are associated with this project.



For success in large software developments, it is important to follow an engineering approach, consisting of a well-defined process.



Software Process

- Process consists of steps/activities to be carried out in a particular order
- Software process deals with both technical and management issues
- Consists of different types of process
- Process for software development:
 - produces software as end-result
 - multiple such processes may exist
 - a project follows a particular process



Process types

- Process for managing the project
 - defines project planning and control (supervising the plan)
 - effort estimations made and schedule prepare
 - resources are provided
 - feedback taken for quality assurance
 - monitoring done
- Process for change and configuration management
 - Resolving requests for changes
 - Defining versions and their compositions
 - Release control



Process types

- Process for managing above processes themselves
 - Improving the processes based on new tools and techniques
 - Standardizations and Certifications (ISO, CMM)



Multiple processes

- A large software development company may have multiple development processes
 - For client-server based conventional applications (example, sales processing, payroll etc.)
 - For enterprise-level (ERP) projects based on packages and customization
 - For data-warehousing/decision-support type
- The company may have many projects going under each category. They may refine these processes to suit the type of projects.



Step in a process

- Each step has a well-defined objective
- Requires people with specific skills (e.g., design of a database, you require certain skills to carry such activity)
- Takes specific inputs and produces well-defined outputs



Step in a process

- Step defines when it may begin (**entry criteria**) and when it ends (**exit criteria**)
- Example: The phase exit criteria for the software requirements specification phase:
 - Software Requirements Specification (SRS) document is complete, reviewed, and approved by the customer.
- A phase can start:
 - only if its phase-entry criteria have been satisfied.



Process step..

- Uses specific techniques, tools, guidelines and conventions.
- Step must be executed as per project plan that gives duration, effort, resources, constraints, etc.



Process step..

- It must produce information for management so that the corrective actions can be taken
 - For example, **effort estimated** may be felt not appropriate, in order to deliver within the **stipulated time**, you may need **more people** to complete it. Such information should be produced by each step for the management
 - For example, adding more resources

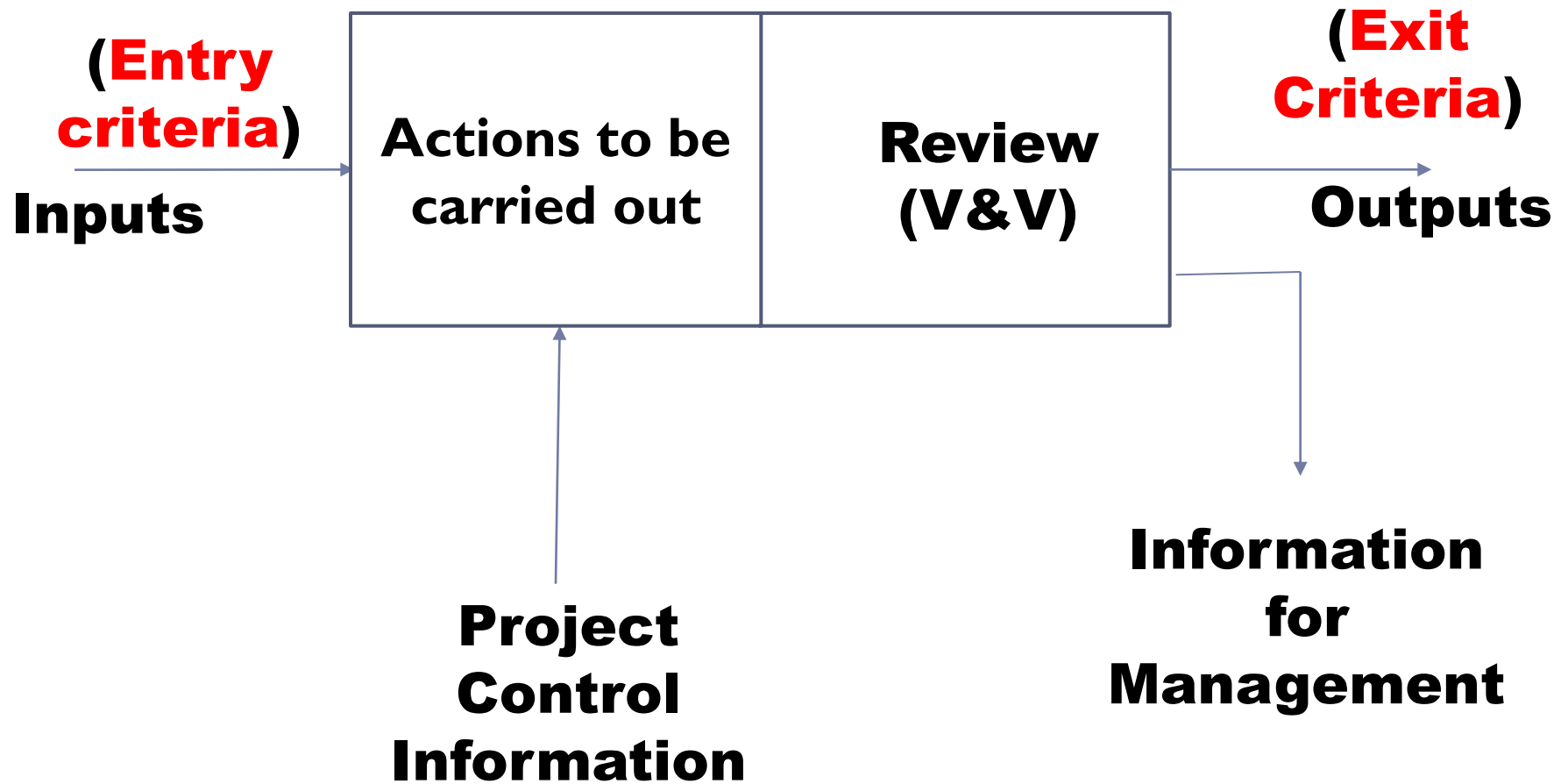


Process step..

- A step ends in a review (V&V)
 - **Verification:** check consistency of the outputs with inputs (of the step)
 - **Validation:** Check consistency with the user needs (meets the user specification)



Process step



Characteristics of a Good Process

- Should be **precisely** defined – no ambiguity about what is to be done, when, how, etc.
- It must be **predictable**-can be repeated in other projects with confidence about its outcome
 - **Predictable with respect to effort, cost:**
 - Project 1:** Web-based library application done by 3 persons in 4 months
 - Project 2:** Guest house booking, similar in complexity should also take about 12 person months.
 - We can extrapolate the projects.



A Good process

- **Predictable for quality:** with respect to number and type of defects, performance
 - Predictable process is said to be under statistical control, when actual values are close to the expected values.
- It supports **testing** and **maintainability**
 - Maintenance by third party
 - Follow standards, provide necessary documentation
 - This characteristic differentiates between prototype and product
- Facilitates **early detection** of and **removal of defects**
 - Defects add to project cost
 - Late detection/correction is costly



A Good process

- It should **facilitate monitoring** and **improvement**
 - Based on feedback
 - Permit use of new tools and technologies
 - Permit measurements



Software Life Cycle (or software process)

- series of identifiable stages that a software product undergoes during its life time:
 - Feasibility study
 - requirements analysis and specification
 - design
 - coding
 - testing
 - maintenance

Life Cycle Model (CONT.)

- a descriptive and diagrammatic model of software life cycle
 - identifies all the activities required for product development,
 - establishes a precedence ordering among the different activities,
 - Divides life cycle into phases.

Life Cycle Model (CONT.)

- Several different activities may be carried out in each life cycle phase or step.
 - For example, the design stage might consist of:
 - structured analysis activity followed by
 - structured design activity.

Why Model Life Cycle ?

- A written description:
 - forms a common understanding of activities among the software developers.
 - helps in identifying inconsistencies, redundancies, and omissions in the development process
 - Helps in tailoring a process model for specific projects

Why Model Life Cycle ?

- Processes are tailored for special projects.
 - A documented process model
 - helps to identify where the tailoring is to occur.

Life Cycle Model (CONT.)

- The development team must identify **a suitable life cycle model**:
 - and then adhere to it.
- Primary advantage of adhering to a life cycle model:
 - development of software takes place in a systematic and disciplined manner.

Life Cycle Model (CONT.)

- When a program is developed by a single programmer
 - he has the freedom to decide his exact steps.
- When a software product is being developed by a team:
 - there must be a precise understanding among team members as to when to do what,
 - Otherwise, it would lead to chaos and project failure.

Life Cycle Model (CONT.)

- A software project will never succeed if:
 - one engineer starts writing code,
 - another concentrates on writing the test document first,
 - yet another engineer first defines the file structure
 - another defines the I/O for his portion first.

Life Cycle Model (CONT.)

- When a life cycle model is adhered to,
 - the project manager can at any time fairly accurately tell,
 - at which stage (e.g., design, code, test, etc.) of the project is.
 - Otherwise, it becomes very difficult to track the progress of the project
 - the project manager would have to depend on the guesses of the team members.

Life Cycle Model (CONT.)

- This usually leads to a problem:
 - known as the 99% complete syndrome.

Summary

- A fundamental necessity while developing any large software product:
 - adoption of a life cycle model.

Summary

- Adherence to a software life cycle model:
 - helps to do various development activities in a systematic and disciplined manner.
 - also makes it easier to manage a software development effort.

Reference

- ▶ R. S. Pressman, *Software Engineering A Practitioner's Approach*, McGraw Hill Publications , 2006
- ▶ R. Mall, *Fundamentals of Software Engineering*, Prentice Hall of India , 2014
- ▶ I. Sommerville, *Software Engineering*, Pearson Education, Asia , 2006
- ▶ P. Jalote, *An Integrated Approach to Software Engineering*, Narosa , 2006