Software Engineering

Software Process

Lecture Learning Objectives

- Understand software process.
- Understand why software process is important.
- Appreciate process-driven Software Engineering.

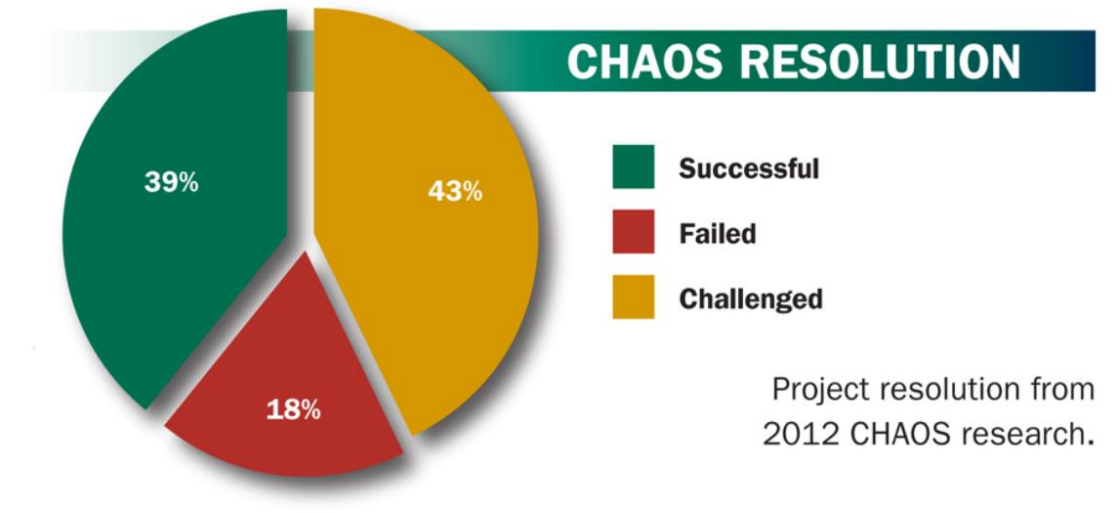
The Problems with Software

- Software problems are often complex and difficult to solve.
- Software solutions require unusual rigor and hard work.
- Software requirements are often ambiguous, not well-defined.
- Software size and complexity grow exponentially over time.
- Large projects experience more problems than small ones, due to team size, communication and skill.

Customer’s View

- Software systems often do not meet customer needs.
- Software often fails.
- Software takes too long to develop.
- Software costs too much.
- Software performance is unpredictable.
- Customers do not have visibility into progress.
- Software often has quality problems.
- Software maintenance is difficult, costly and error prone.
- Software from one system is seldom used in another system even when similar functions are required.

State of Software Project

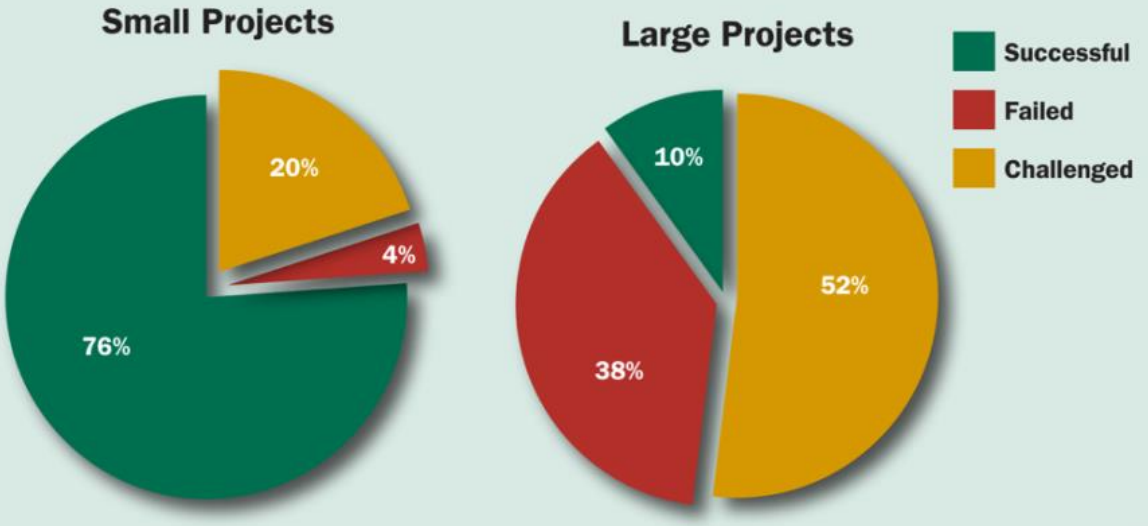


- *Succeeded*: on time, on budget and with all functions include
- *Challenged*: Late and/or over budget and/or less functional than planned.
- *Failed*: cancelled or delivered and never used.

Failure rates by project size

CHAOS RESOLUTION BY LARGE AND SMALL PROJECTS

Project resolution for the calendar year 2012 in the new CHAOS database. Small projects are defined as projects with less than $1 million in labor content and large projects are considered projects with more than $10 million in labor content.



The underlying causes

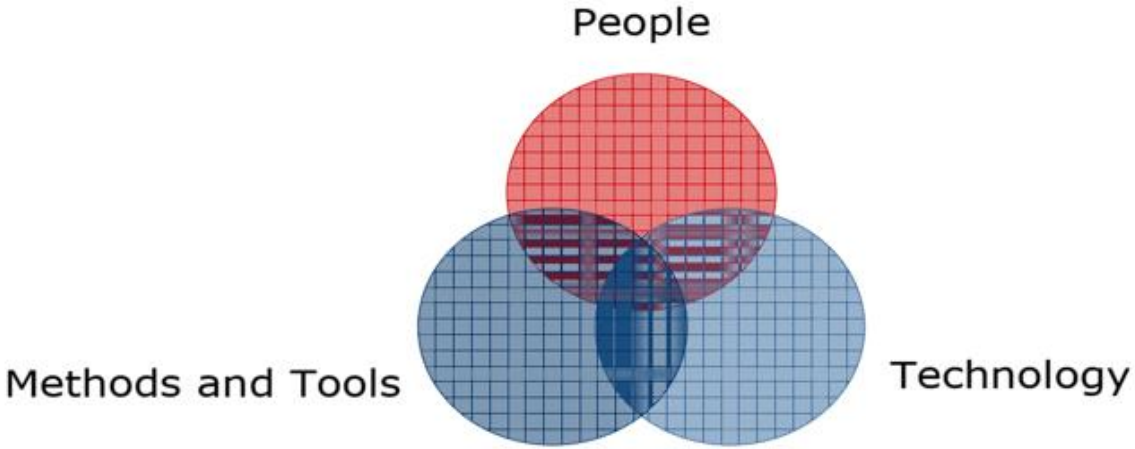
- The inability of organizations to deal with the increase in size and the complexity of software products.
- The tendency of organizations to continue using archaic programming languages and obsolete practices.
- Failure of education institutions to adjust curricula for modern software practices.
- A shortage of personnel trained in software engineering. Failure of organizations to understand the importance of software development processes.
- The difficulty in measuring intangible products (can not touch or see).

Why Software Process?

- The quality of a software product is governed by the quality of the process used to develop and maintain it.
- To improve the quality of the product, one must improve the quality of the process that creates the product.
- By focusing only on a software product, you can not solve the size and complexity (scalability) issues and do not have the knowledge of how to do it better.
- By focusing on the software process and understanding every step along the development path, you can predict the quality of the product, the project trends, and be able to repeat what you have done in future work.

Definition of Process

The logical organization of people, technology, methods and tools into work activities designed to produce a high quality end result.



Process Perspective

- People:
  - Must have the skill, training, and motivation necessary to do the work. They must be managed in a way that will increase their effectiveness.
- Technology:
  - Must be selected to enhance the business and support the product needs.
- Methods & Tools:
  - Must be defined to guide people and the application of technology in the business of providing products and services to the customers.

Definitions

- Process (What, what's next)
  - A series of actions, changes, or functions organized in a structured step-by-step way to achieve an end-result.
- Method (How to)
  - A complete set of rules that establishes a precise and repeatable way of performing a task and arriving at a desired result.
- Methodology
  - A collection of methods, procedures, and standards that defines an integrated synthesis of software engineering approaches to the development and maintenance of a product.
- Tools (automated)
  - An automated or semi-automated device to support process, method and methodologies.

Software Process

- Process is a set of activities, methods, and practices that are used in the development of software.
- The basic element of process is the process unit which can be defines as an activity uniquely identified to accomplish a specific task.
- Process can be described using the ETVX notation (Entry-Task-Validation-Exit):
  - Each process unit requires a Entry condition (input), a Task to do something (task can be a procedure, method), a Validation (measurement), an Exit condition (Output) to define the results.

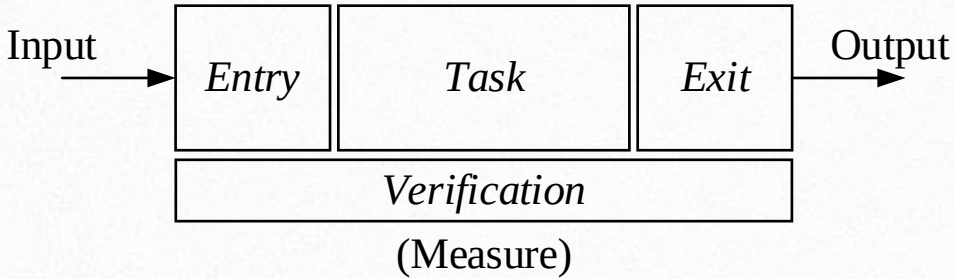
Process Unit

Entry: Condition to be met before a task can begin.

Task: What to be done (who, what, when)?

Validation: Measure the task to ensure it meets requiremers.

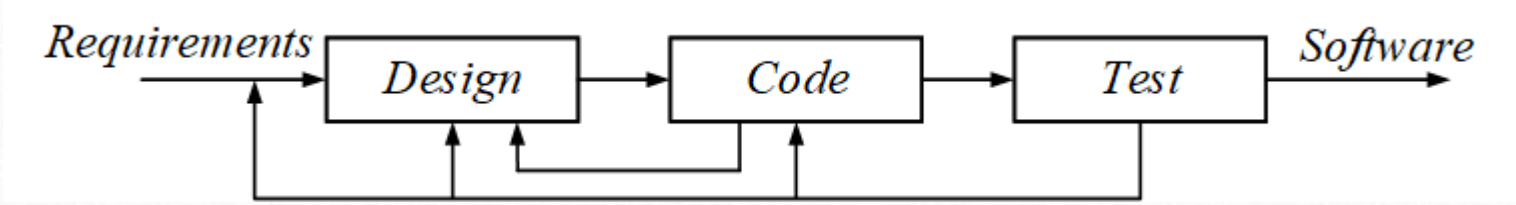
Exit: End-Results.



The ETVX Model

- The simplest process description is the ETVX notation where:
  - 'E' is the entry criteria which must be satisfied before a set of tasks can be performed.
  - 'T' is the set of tasks to be performed.
  - 'V' stands for the verification & validation process to ensure that the right tasks are performed.
  - 'X' stands for the exit criteria or the output of the tasks.
- If an activity fails in the validation check, corrective action is taken or a rework is ordered.
- This notation can be used in any development process. Each phase in the process can be considered as an activity and structured using the ETVX model. If required, the tasks can be further subdivided and each subtask can be further structured using the ETVX model.

Example of Process



Process Unit	Design	Code	Test
Entry	Approved Requirements	Approved Design	Approved Code
Task	Perform Design	Implement Code	Conduct Tests
Verification	Resources, Errors, Design, Document	Resources, Errors, Code, Document	Resources, Errors, Code, Document Test Suite
Exit	Approved Design	Approved Code	Approved Software

The Defined Process

- Software process is a critical factor in software engineering. Software quality and productivity improvements can be obtained by defining a quality process and measuring the results.
- A Defined process is a set of proven successful processes (best practices) approved for use in an organization to produce quality products on a predictable basis.
- A Defined process is also a communication aid to ensure that each team member understands the activities to be carried out in performing the job.
- A Defined process can support management to plan all project activities, and monitor and measure the quality of the development process before and during the creation of the software product.

Defined Process

- For a Defined Process be considered fully in place or institutionalized, it must be:
  1. Defined
  2. Documented
  3. Trained
  4. Used (Practiced)
  5. Measured
  6. Verified (Independently)
  7. Continuously improved (Has been in use at least 6-12 months)

The Process-Driven Development

- Process-driven software development begins by defining the development process for a project, based on proven successfull processes from prior experiences.
- After the process has been defined, software engineers must be trained to follow the defined process.
- As software engineers execute the process, managers can monitor and measure the execution to ensure that the process is being followed and the results meet the desired quality requirements.

The Process-Driven Development

- By having measurements, deficiencies may be found. When this occurs, the process is modified to correct errors or enhance it, and reinstalled and used for software development.
- When usage brings good results, the defined processes can be placed in a Process Asset Library (PAL), so they can be reused for similar projects in the future.

Why Process Driven Engineering Now?

- Problems with software:
  - Hardware costs are stabilizing and decreasing.
  - Software costs are increasing.
  - **Requirements are often changing.**
  - Software size keeps growing larger and more complex.
  - Software quality needs improvement.
  - Many major problems can be traced to software.
- Process driven engineering is designed to:
  - Control costs and schedules predictably.
  - Respond to changing needs.
  - Be scalable from small to large systems.
  - Produce quality products predictably.

To Control Costs and Schedules

- To control cost and schedule, software process must be defined according to these principles:
  1. People who do the work estimate and plan that work.
  2. Methods and relevant data are used to plan, track, and manage the work.
  3. The progress of the work is regularly tracked.
  4. When progress falls behind, the problem causes are promptly identified and resolved.
  5. When requirements change, all involved tasks must be re-estimated and the project plan revised.
  6. Risks are anticipated and managed.

Respond to Changing Needs

- Requirements often change in software projects and project managers must be responsive to new requirements while continuing to meet prior cost and schedule commitments by:
  1. Examining each proposed change to understand its effects on the project plan.
  2. Pay particular attention to how each change will impact completed work, including the requirements, design, implementation, verification, and testing activities.
  3. Estimate all cost and schedule consequences of making needed changes.
  4. Where cost and schedule implications are significant or where they exceed the currently approved plan, get management approval before proceeding.

Be Scalable to Size of Projects

- A defined process must follow principles and practices that are suitable for the size of the project. To be highly scalable, a defined process must meet these criteria:

  1. Must use robust and precise methods at all levels, especially at the systems, hardware, and software-engineer levels.

  2. Must use management system for technical and management program decisions, based on the knowledge and judgment of the development-level engineers and anyone else who has relevant information.

  3. Must consistently use data that is derived from accurate, precise, and auditable process and product measurements.

Produce Quality Products

- Quality process will consistently produce quality products while a poor-quality process will generally produce low-quality products.
- A problem in software development is many software people do not receive adequate training on process-driven techniques and are only trained in programming techniques.
- When software people produce a small functional program that works well, they feel they have proven their techniques, continuing to use the program for larger-scale work. For this reason, many larger software projects run a significant risk of getting poor-quality results.
- Unless software people are willing to follow well-defined and proven processes, they will waste their time in fixing errors and reworking failed projects.

Learning From Success

- All modern sciences are based on learning from prior effective experiments and practices.
- Software development is a learning process, and unless this learning is defined and preserved, the knowledge is lost.
- Software engineers should define, use, and continually improve their own processes and learn from other's experience.
- Competent engineers know what has been successful and do not waste time with processes that have produced unsatisfactory results.
- Quality work does not happened randomly. Quality must be planned, measured, tracked, and managed. When it follows a well-defined quality process, defects are normally reduced by at least by ten times (order of magnitude).

Defect Prevention

- To produce quality work, the main goal of the defined process is to prevent defects before they are introduced into the software.
- To produce quality software products consistently, the process must have the goal of removing all defects before testing. The objective of testing then is to verify and validate the product – not to fix defects.
- To ensure an effective and high-quality process, the defined process must have quality measures.
- To manage a quality process effectively, every team member must support and follow the defined quality process.

Defect Prevention

- Removing defects before testing is an essential element of a quality management programs. The following principles of quality management are based on the following facts:
  - It costs more in money and time to build and fix defective products than it would have taken to build them properly the first time.
  - It costs more to fix a defective product after delivery to users than it would have cost to fix it before delivery.
  - It costs more and takes longer to fix a product in the later testing stages than in the earlier design and development stages.
  - It costs more and takes longer to fix requirements errors in the design, implementation, testing, and operation phases than in earlier requirements development.
  - The least expensive and most efficient plan is to prevent the defects from happening altogether.

The Quality Program

- The requirement that everyone must participate in a quality program is a key principle in process-driven software engineering.
- Quality work results only from consistent striving for perfection. All team members must strive to produce defect-free work. Any undisciplined work by any individual can be a source of defects, and their work must be measured and controlled. If it is not, high-quality products simply will not be produced.
- In addition to development team members, all support groups and managers must also follow the defined quality process to ensure the quality of the product will be defect-free.

Summary

- Process is a set of activities, methods, and practices that are used in the development of software.
- The quality of a software product is governed by the quality of the process used to develop and maintain it.
- To improve quality of a product, one must improve the quality of the process that creates the product.
- A defined process is a set of proven successful processes (best practices) approved for use in an organization to produce quality products predictably.
- By following a defined process, organizations can:
  - Control costs and schedules predictably.
  - Respond to changing needs.
  - Be scalable from small to large systems.
  - Produce quality products predictably.

Assingment

1. Software process?
2. Why software process?
3. Process description and process unit?
4. Process Driven Software engineering?
5. Underlying causes of software development issues?