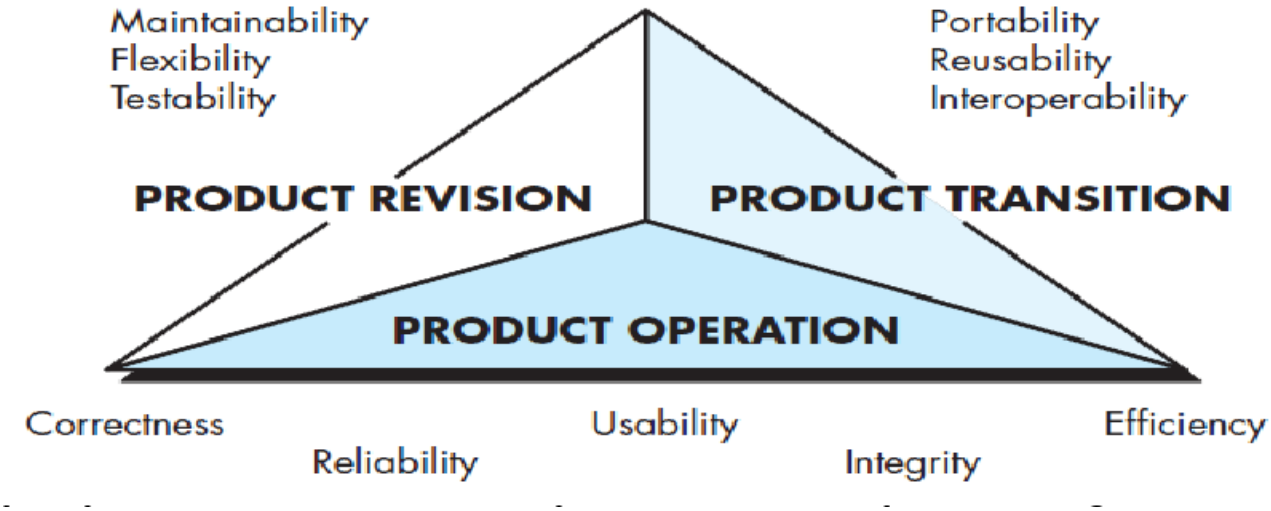
MSS Module 5 Important Topics

> ***Table of contents***

Software Quality Dilemma

McCall’s quality factor



- McCall, Richards, and Walters propose a useful categorization of factors that affect software quality.
- These software quality factors focus on three important aspects of a software product: its operational characteristics, its ability to undergo change, and its adaptability to new environments

Good Enough software

- Good enough software delivers high-quality functions and features that end users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.

The Cost of Quality

- The cost of quality includes all costs incurred in the pursuit of quality or in performing quality-related activities and the downstream costs of lack of quality.

Elements of Software Quality Assurance

- Standards
  - The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents.
  - Standards may be adopted voluntarily by a software engineering organization or imposed by the customer or other stakeholders.
- Reviews and audits

    - Technical reviews are a quality control activity performed by software engineers for software engineers
    -
- Testing
    - Software testing is a quality control function that has one primary goal—to find errors.
- Error/Defect collection and analysis
    - The only way to improve is to measure how you're doing.
    - SQA(Software quality assurance) collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them
- Change management
    - Change is one of the most disruptive aspects of any software project
    - SQA ensures that adequate change management practices have been instituted Education
- Education.
    - The SQA organization takes the lead in software process improvement and is a key proponent and sponsor of educational programs.
- Vendor management.
    - The job of the SQA organization is to ensure that high-quality software results by suggesting specific quality practices that the vendor should follow (when possible), and incorporating quality mandates as part of any contract with an external vendor.
- Security management.
    - SQA ensures that appropriate process and technology are used to achieve software security
- Safety
    - SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
- Risk management
    - SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established

---

SQA Tasks

- The Primary purpose of the SQA group is to assist the software team in achieving a high-quality end product

- The Software Engineering Institute recommends a set of SQA activities that address quality assurance planning, oversight, record keeping, analysis, and reporting
- The tasks of an SQA (Software Quality Assurance) group include
    1. **Preparing an SQA Plan:**
        - Develop an SQA plan for the project.
        - Review the plan with all stakeholders.
        - Govern quality assurance activities based on the plan.
        - Identify evaluations, audits, reviews, applicable standards, error reporting procedures, work products, and feedback mechanisms.
    2. **Participating in Software Process Description:**
        - Collaborate in developing the project's software process description.
            - A software process description is a detailed document that outlines the steps, activities, procedures, and guidelines involved in a specific software development process
        - Review the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other aspects of the software project plan.
    3. **Reviewing Software Engineering Activities:**
        - Review software engineering activities to ensure compliance with the defined software process.
        - Identify, document, and track deviations from the process.
        - Verify that corrections to deviations have been implemented.
    4. **Auditing Software Work Products:**
        - Audit designated software work products to verify compliance with the defined software process.
        - Review selected work products, identify deviations, document and track them.
        - Verify that corrections to deviations have been made.
        - Periodically report the results of work to the project manager.
    5. **Handling Deviations:**
        - Ensure that deviations in software work and work products are documented.
        - Establish and follow a documented procedure for handling deviations.
        - Deviations may be related to the project plan, process description, applicable standards, or software engineering work products.
    6. **Recording Noncompliance and Reporting:**
        - Record instances of noncompliance.
            - Noncompliance refers to the failure to adhere to or meet specified requirements, standards, regulations, or established processes

- Report noncompliance to senior management.
- Track noncompliance items until they are resolved.

SQA Goals

- Requirement’s quality.
  - The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- Design quality.
  - Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements
- Code quality.
  - Source code and related work products must conform to local coding standards and exhibit characteristics that will facilitate maintainability
- Quality control effectiveness
  - A software team should apply limited resources in a way that has the highest likelihood of achieving a high-quality result

---

Software Process Improvement(SPI), SPI Process

- Software Process Improvement (SPI) means a few important things.
  - It suggests that we can clearly define the key parts of a good software process.
  - It implies that we can evaluate how well a company currently makes software by comparing it to these key parts
  - it indicates that we can come up with a useful plan for making things better.

Approaches to SPI

- An organization can choose one of the many SPI frameworks. An SPI framework defines
  - a set of characteristics that must be present if an effective software process is to be achieved
  - a method for assessing whether those characteristics are present
  - a mechanism for summarizing the results of any assessment
  - a strategy for assisting a software organization in implementing those process

characteristics that have been found to be weak or missing.

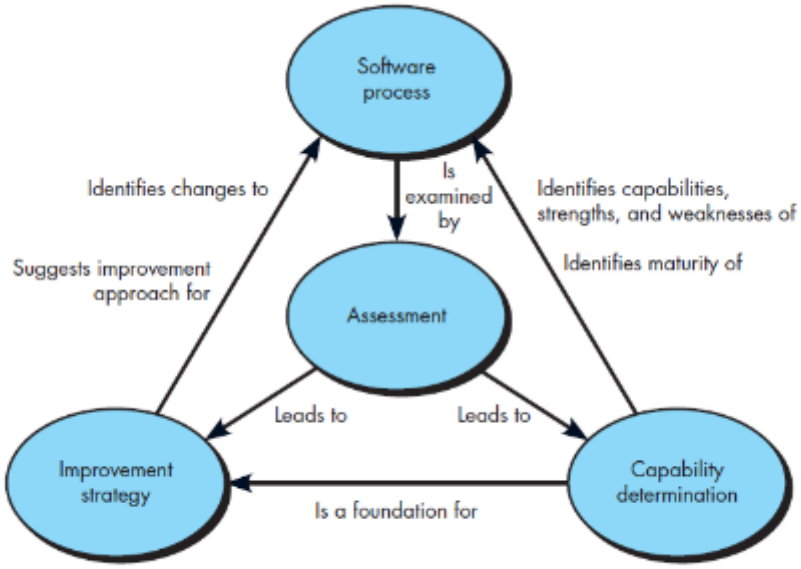


Fig: Elements of a SPI framework

SPI Support constituencies

- Quality certifiers:
  - Their approach is examine a well-defined set of characteristics that allows them to determine whether the process exhibits quality.
- Formalists
  - This group wants to understand process workflow.
  - To accomplish this, they use process modelling languages (PMLs) to create a model of the existing process and then design extensions or modifications that will make the process more effective.
- Tool advocates.
  - This group insists on a tool-assisted approach to SPI
- Practitioners
  - They have a practical, realistic, and hands-on approach to their work.
  - They may not use highly structured and rigid process models or tools to execute their tasks.
  - They carefully plan and measure progress for each specific project.
- Reformers
  - The goal of this group is organizational change that might lead to a better software process.
  - They tend to focus more on human issues and emphasize measures of human capability.
- Ideologists

- Ideologists have a greater interest in a process that would support reuse or reengineering

---

Cloud-based Software

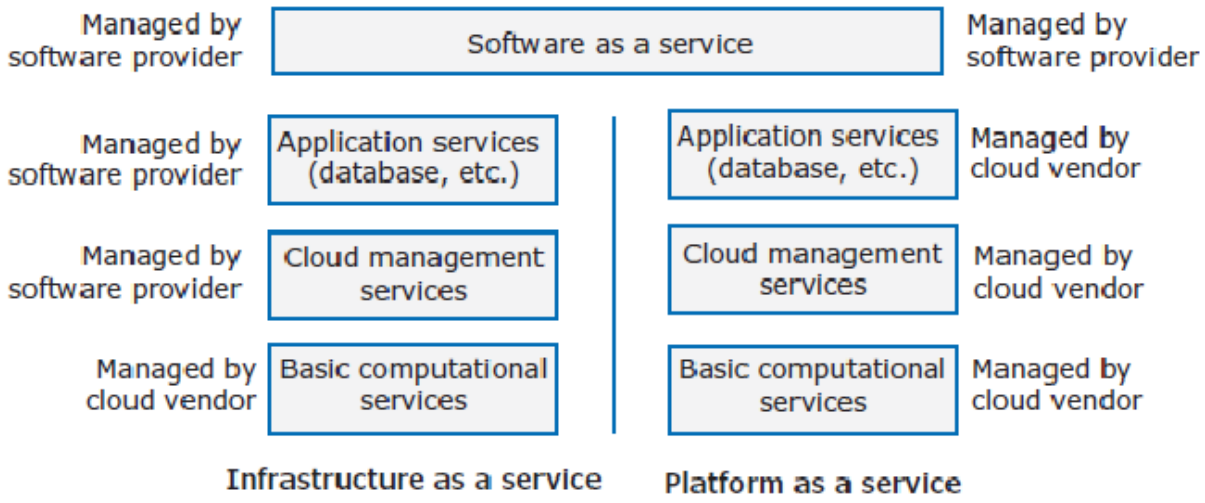
- Cloud-based computing allows users access to software applications that run on shared computing resources via the Internet
- These computing resources are maintained in remote data centres dedicated to hosting various applications on multiple platforms.

Benefits of cloud

- Cost
  - You can avoid initial capital costs of hardware procurement
- Startup time
  - You dont have to wait for hardware to be delivered before you can start work. Using the cloud you can have servers up and running in a few minutes
- Server choice
  - If you can find that the servers you are renting are not powerful enough, you can upgrade to more powerful systems.
  - You can add servers for short term requirements such as load testing.
- Distributed development
  - If you have a distributed development team, working from different locations, all team members have the same development environment and can seamlessly share all information.

Everything as a service

Figure 5.6 Management responsibilities for SaaS, IaaS, and PaaS



Infrastructure as a service (IaaS)

- This is a basic service level that all major cloud providers offer.
- These infrastructure services may be used to implement virtual cloud based servers.
- The key benefits of using IaaS are
  - You don’t incur the capital costs of buying hardware
  - You can easily migrate your software from one server to a more powerful server.
- Using the cloud provider’s control panel, you can easily add more servers if you need to as the load on your system increases

Platform as a service (PaaS)

- This is an intermediate level where you use libraries and frameworks provided by the cloud provider to implement your software.
- These provide access to a range of functions, including SQL and NoSQL databases.
- Using PaaS makes it easy to develop auto-scaling software

Software as a service (SaaS)

- Your software product runs on the cloud and is accessed by users through a web browser or mobile app
  - Examples include mail services such as Gmail, storage services such as Dropbox, social media services such as Twitter
- If you are running a small or medium-sized product development company, it is not cost effective to buy server hardware.

- If you need development and testing servers, you can implement these using infrastructure services.

Virtualisation and containers

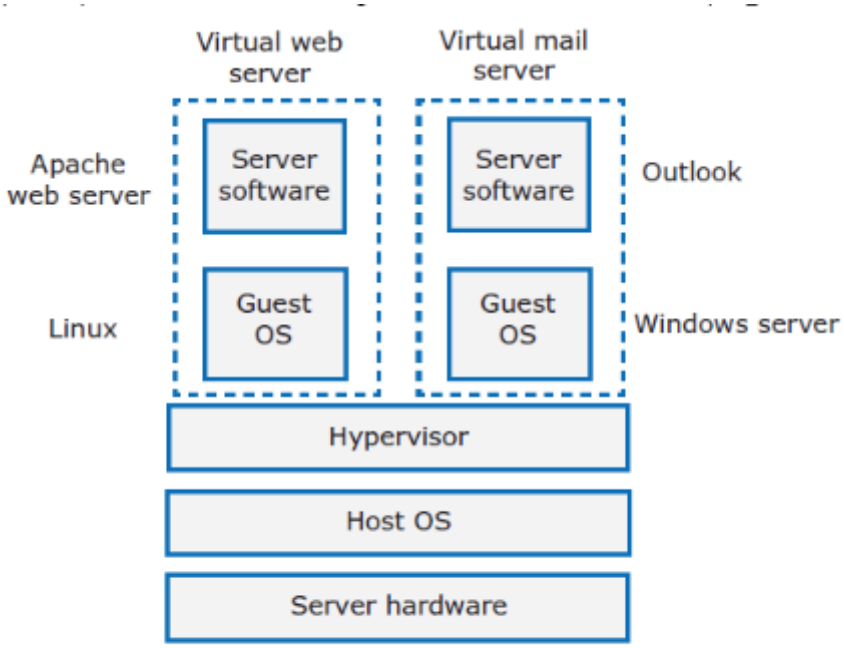


Fig: Implementing a virtual server as a virtual machine

Virtual Server

- All cloud servers are virtual servers.
- A virtual server runs on an underlying physical computer and is made up of an operating system plus a set of software packages that provide the server functionality required.
- The general idea is that a virtual server is a stand-alone system that can run on any hardware in the cloud.
- Virtual machines (VMs), running on physical server hardware, can be used to implement virtual servers
- **Hypervisor** provides a hardware emulation that simulates the operation of the underlying hardware
    - A **Hypervisor** is like a virtual simulator for your computer's hardware. It pretends to be the actual hardware so that different operating systems or software can run on it, even if they are not really using the physical hardware directly.

Containers

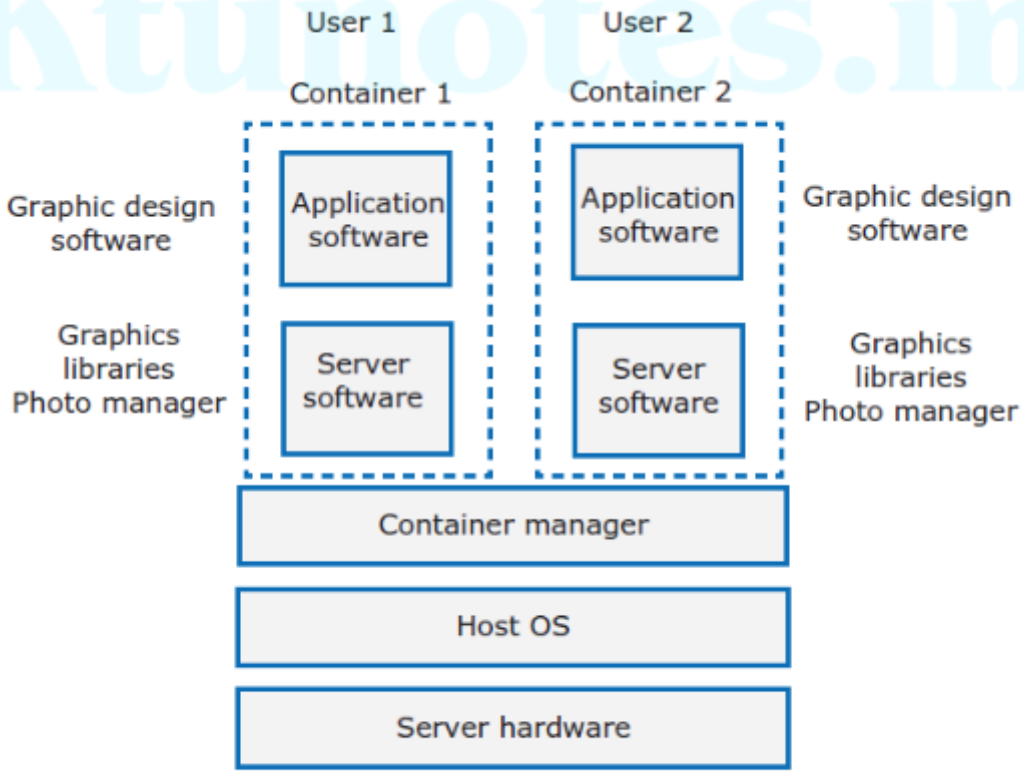


Fig: Using containers to provide isolated services

- If you are running a cloud-based system with many instances of applications or services, these all use the same operating system, you can use a simpler virtualization technology called **containers**
- **Containers** are an operating system virtualization technology that allows independent servers to share a single operating system.
- Using containers dramatically speeds up the process of deploying virtual servers on the cloud
  - Containers are usually megabytes in size, whereas VMs are gigabytes.
  - Containers can be started up and shut down in a few seconds rather than the few minutes required for a VM
  - Many companies that provide cloud-based software have now switched from VMs to containers because containers are faster to load and less demanding of machine resources

---

Microservices architecture

Microservice

- A service should be related to a single business function.

- Instead of relying on a shared database and other services in the system, services should be completely independent, with their own database.
- They should also manage their own user interface.
- **Microservices** are small-scale, stateless services that have a single responsibility.

Microservice Architecture

- Software products that use microservices are said to have a **microservices architecture**
- Microservices architecture is an architectural style— a tried and tested way of implementing a logical software architecture.
    - For web-based applications, this architectural style is used to implement logical client-server architectures, where the server is implemented as a set of interacting microservices.
- Microservices are self-contained and run in separate processes. In cloud based systems, each microservice may be deployed in its own container.
    - This means a microservice can be stopped and restarted without affecting other parts of the system.
    - If the demand on a service increases, service replicas can be quickly created and deployed.
- Microservice architecture aims to address 2 fundamental architecture
    - When a monolithic architecture is used, the whole system has to be rebuilt, retested, and re-deployed when any change is made.
        - This can be a slow process, as changes to one part of the system can adversely affect other components. Frequent application updates are therefore impossible
    - As the demand on the system increases, the whole system has to be scaled