Topics to be Discussed

- IOT (Internet of Things)
- Artificial Intelligence
- AR & VR (Augmented reality & Virtual reality)

IOT - INTERNET OF THINGS

The term IoT, or Internet of Things, refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves.

Applications of IoT

- Wearables
- Smart Home Applications
- Health Care
- Smart Cities
- Agriculture
- Self-driving cars etc.

Wearables

Wearable technology is a hallmark of IoT applications and probably is one of the earliest industries to have deployed the IoT at its service. We happen to see Fit Bits, heart rate monitors and smartwatches everywhere these days.

- Smart Watch
- Gaming Simulator
- Smart Shoes
- Fitness tracker
- Smart clothing etc.

HEALTH CARE

- REMOTE PATIENT MONITORING

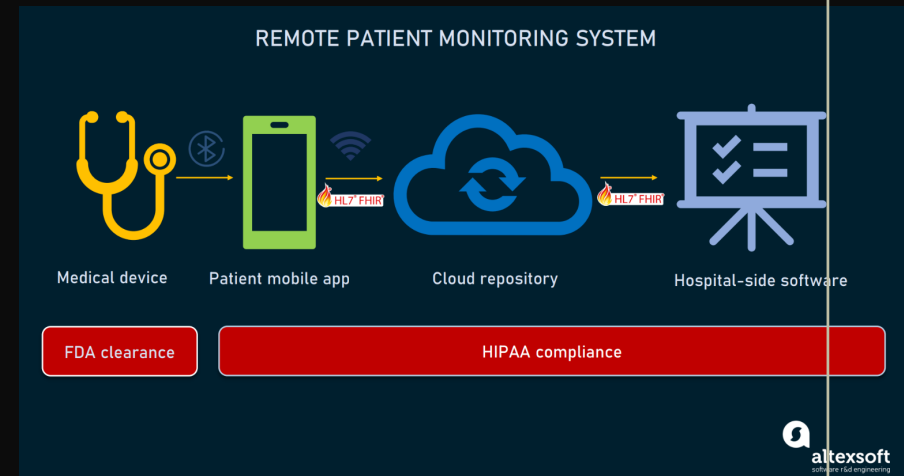
DISTANT PATIENT CHECKING IS THE MOST WIDELY RECOGNIZED USE OF IOT GADGETS FOR MEDICAL CARE. IOT GADGETS CAN CONSEQUENTLY GATHER WELLBEING MEASUREMENTS LIKE PULSE, CIRCULATORY STRAIN, TEMPERATURE, AND MORE FROM PATIENTS WHO ARE NOT GENUINELY PRESENT IN A MEDICAL SERVICES OFFICE, DISPENSING WITH THE REQUIREMENT FOR PATIENTS TO GO TO THE SUPPLIERS, OR FOR PATIENTS TO GATHER IT THEMSELVES.

- GLUCOSE MONITORING

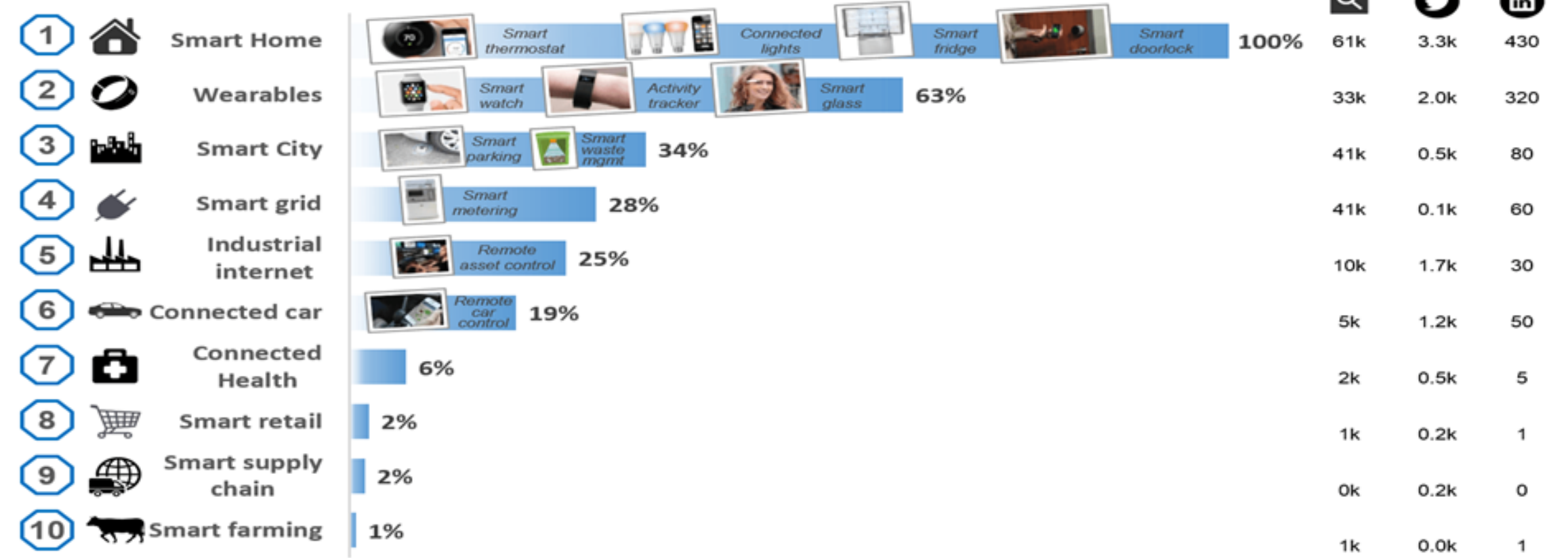
IOT GADGETS CAN ASSIST WITH TENDING TO THESE DIFFICULTIES BY GIVING PERSISTENT, PROGRAMMED OBSERVING OF GLUCOSE LEVELS IN PATIENTS. GLUCOSE CHECKING GADGETS DISPENSE WITH THE NEED TO KEEP RECORDS PHYSICALLY, AND THEY CAN CAUTION PATIENTS WHEN GLUCOSE LEVELS ARE HAZARDOUS.

- HEART-RATE MONITORING

LIKE GLUCOSE, CHECKING PULSES CAN BE TESTING, IN ANY EVENT, FOR PATIENTS WHO ARE AVAILABLE IN MEDICAL SERVICES OFFICES. INTERMITTENT PULSE CHECKS DON'T PREPARE FOR QUICK CHANGES IN PULSES, AND ORDINARY GADGETS FOR PERSISTENT HEART OBSERVING UTILIZED IN CLINICS EXPECT PATIENTS TO BE CONNECTED TO WIRED MACHINES CONTINUALLY, HINDERING THEIR VERSATILITY.



IoT Analytics – *Quantifying the connected world*



1. Monthly worldwide Google searches for the application 2. Monthly Tweets containing the application name and #IOT 3. Monthly LinkedIn Posts that include the application name. All metrics valid for Q4/2014.
Sources: Google, Twitter, LinkedIn, IoT Analytics

SELF DRIVING CARS

- It is a driverless automated ground vehicle which senses the environment and operates with very less or no human input.
- Technologies used: Sensors, Cameras, Radar and Artificial Intelligence to travel between destinations.

SMART HOMES

- A home equipped with lighting heating and electronic devices that can be controlled remotely by phone or computer.
- Technologies used: Zigbee and Z-wave

AR & VR

- **AR – Augmented Reality**

Augmented Reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual information. It is merely adding to the user's real-life experience.

Example: Navigation(GPS), Medical(MRI)

- **VR – Virtual Reality**

Virtual Reality (VR) is a simulated experience that can be similar or completely different from the real world. These immersive simulations can create almost any visual imaginable for the person using equipment.

Example: Oculus Rift, Google Cardboard

MR & XR

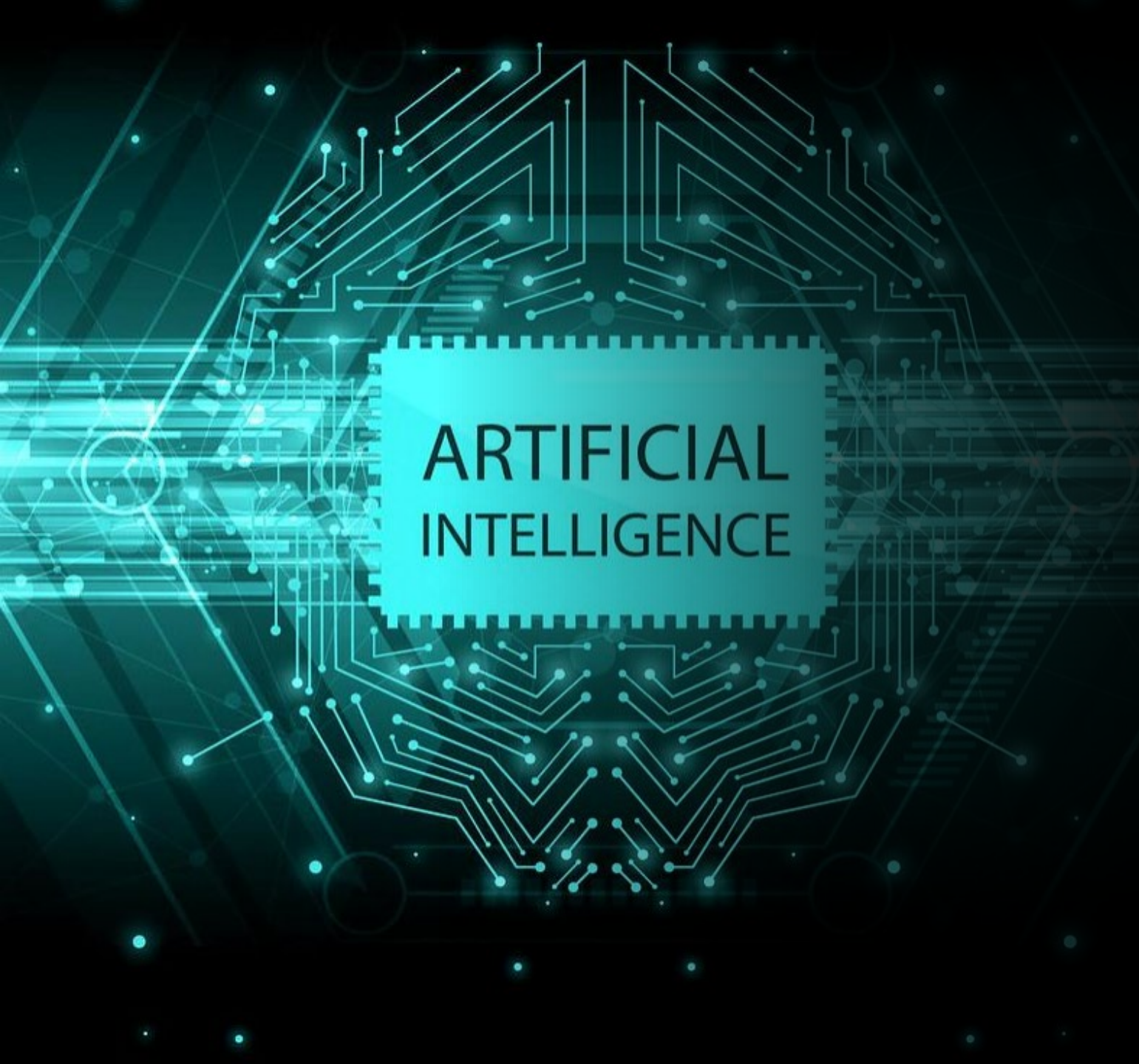
- **MR – Mixed Reality**

Mixed reality (MR) is a graphical interface that combines real-world and computer-generated components in an interactive perception. MR is simply an enhanced, more interactive version of AR.

- **xR – eXtended Reality**

All real and virtual worlds created by computer technologies and wearables are referred to as extended reality (xR). The letter 'x' in xR stands for a variable that can be any letter.

AR	VR
The system augments the real-world scene	Completely immersive virtual environment
In AR User always have a sense of presence in the real world	In VR, visual senses are under control of the system
AR is 25% virtual and 75% real	VR is 75% virtual and 25% real
This technology partially immerses the user into the action	This technology fully immerses the user into the action
AR requires upwards of 100 Mbps bandwidth	VR requires at least a 50 Mbps connection
No AR headset is needed.	Some VR headset device is needed.
With AR, end-users are still in touch with the real world while interacting with virtual objects nearer to them.	By using VR technology, VR user is isolated from the real world and immerses himself in a completely fictional world.
It is used to enhance both real and virtual worlds.	It is used to enhance fictional reality for the gaming world.



•Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.

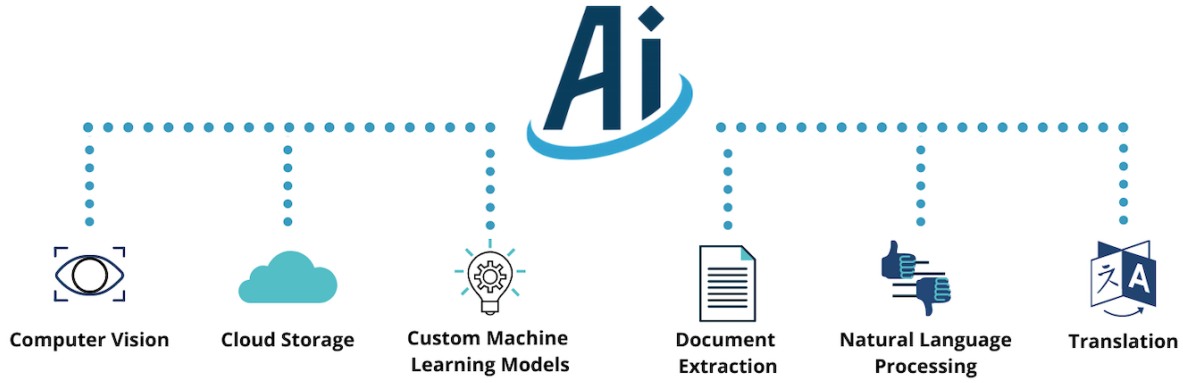
•In the most basic sense, Artificial intelligence (AI) is a tool that makes it possible for machines to learn from experience, adjust to new inputs and perform human-like tasks.

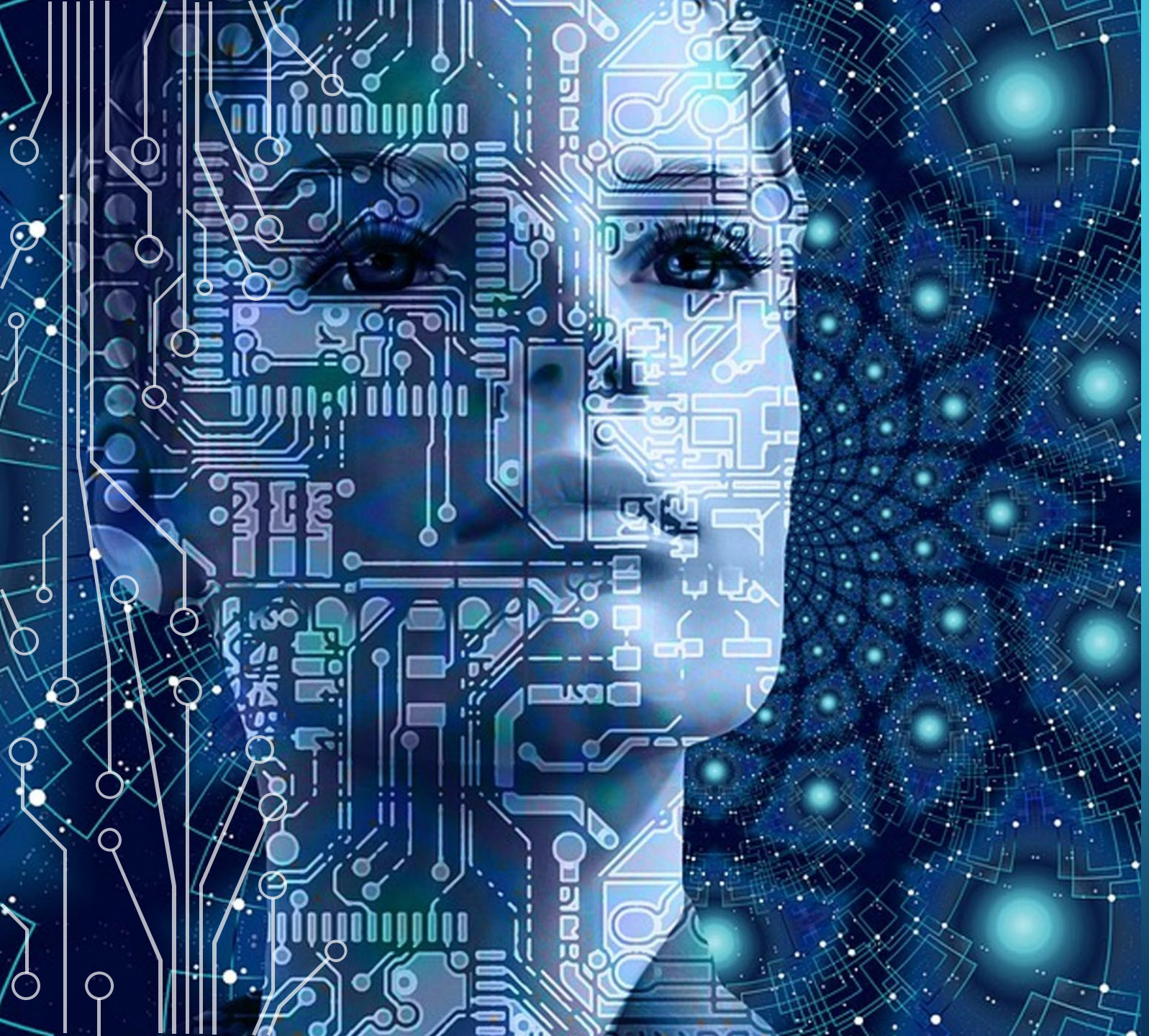
•AI is getting used across different industries including finance and healthcare.

•Examples: Siri, Alexa.

FEATURES OF ARTIFICIAL INTELLIGENCE

- Computer Vision
- Cloud Storage
- Custom Machine Learning Models
- Document Extraction
- Natural Language Processing
- Translation





HOW AI WORKS?

AI systems work by combining large sets of data with intelligent, iterative processing algorithms to learn from patterns and features utilize a whole series of techniques and processes, as well as a vast array of different technologies.

THANKYOU

AVINASH KUMAR MADDINI
SARANG MEHTA
JAHNAVI NEKKANTI
HEMANTH REDDY MEDA
TEJASWI RAVIPATI
ARCHANA PRAKASH NIKAM
ARTH MODI
MADAS KRUPANAND HRIDAY RANJAN
LAKSHMI GEETHIKA NAINALA

Software Engineering

Group 4 : Testing

CSE-5325

Heer Patel

Jaykumar Dhirubhai Patel

Kabir Amit Oza

Naga Guru Aishwarya Pannala

Naveena Pachava

Pooja Bhupendrakumar Patel

Rohit Ravindra Padwal

Tirth Nareshbhai Patel

Venkata Sai Phaneendra Padala

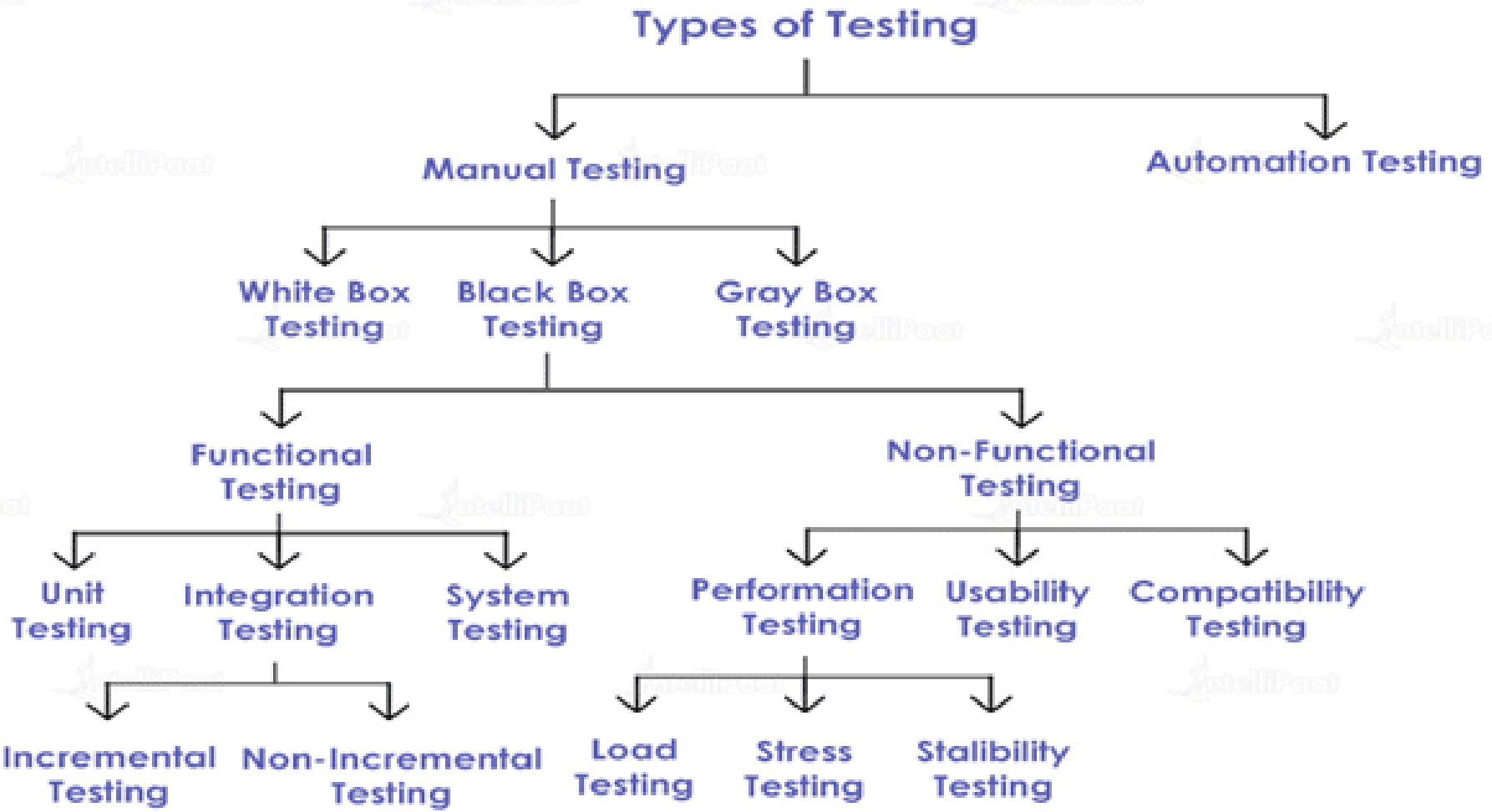
UTA

What is testing?

- Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

Why do we need testing in the software industry?

- Helps in saving money
- Security
- Quality of the product
- Satisfaction of the customer
- Enhancing the development process
- Easy while adding new features
- Determining the performance of the software


Types of Testing
Manual Testing
Automation Testing
White Box Testing
Black Box Testing
Gray Box Testing
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performation Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Load Testing
Stress Testing
Stalibility Testing

Automation Testing

Benefits

- Quick and effective
- Better long-term ROI
- Transparent and meticulous

Drawbacks

- Additional costs required
- Lack of human input on usability and UI
- Tools' limitations and in-built issues

Manual Testing

Benefits

- True-to-life testing
- Thorough design review
- Lower short-term investment

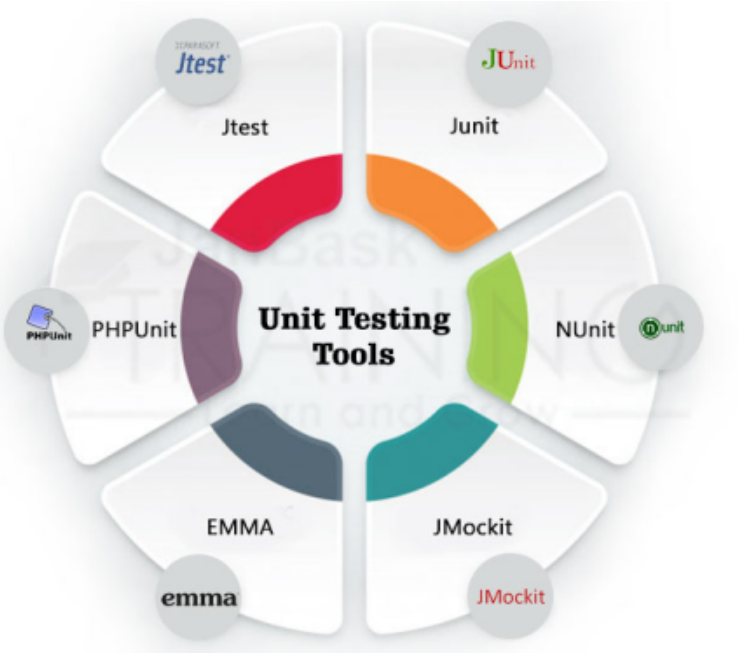
Drawbacks

- Increased risk of a failure
- More resources and time required
- Not everything can be tested manually

Unit Testing

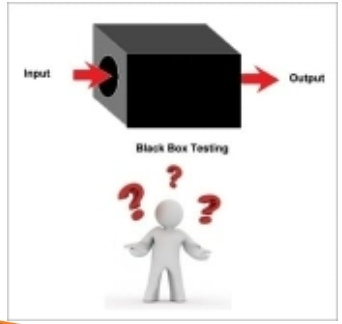
- Component tested in isolation from the rest of the system

- **Why?**
- Foundation for other testing like integration and system testing
- **What to test?**
- smallest units of code such as functions, procedures, subroutines, subprograms
  Methods, classes
- **When and who?**
- Frequently done during code development by code developers



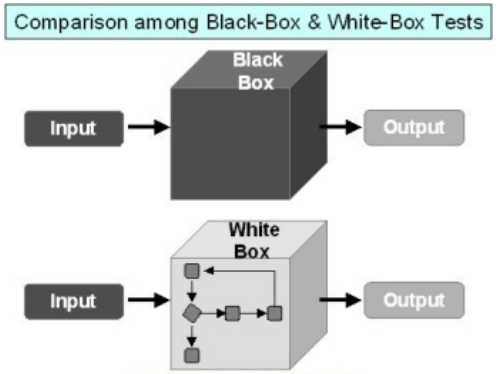
Black box testing

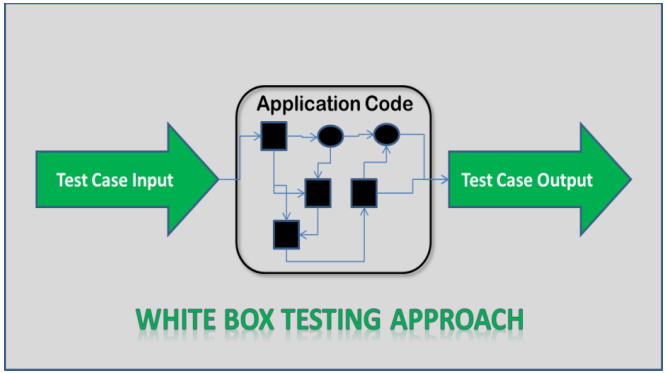
- Definition : Analyses the functionality of a software/application without knowing much about the internal structure/design
- Mainly focuses on input and output of software applications
- It is called behavioral testing
- The majority of test case are covered so find most of bugs
- Testers do not require technical knowledge, programming, or IT skills
- Code is unknown to tester



White box testing

- Internal structure, design and coding of software are tested
- Other name : Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing
- It is functional testing.
- Code should be known to tester in order to perform this test





Security testing

- Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders
- Security Tests is to identify all possible loopholes and weaknesses of the software system
- helps developers to fix the problems through coding

Example Test Scenarios for Security Testing:

- A password should be in encrypted format
- Application or System should not allow invalid users
- Check cookies and session time for application
- For financial sites, the Browser back button should not work.

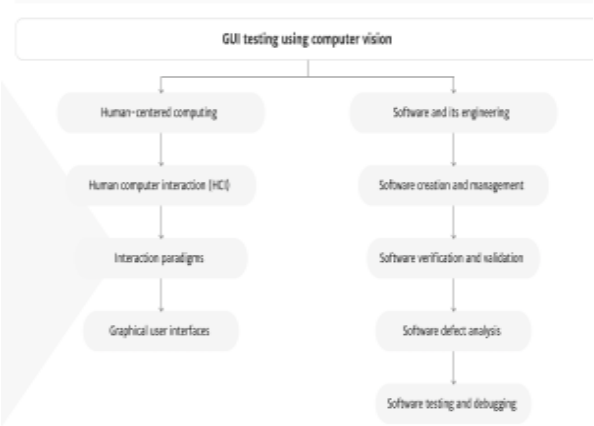
Let's get to know about some tools for security testing

- AFL – code analyzer
- Wireshark – network analyzer

What is UI testing, and why is it important?

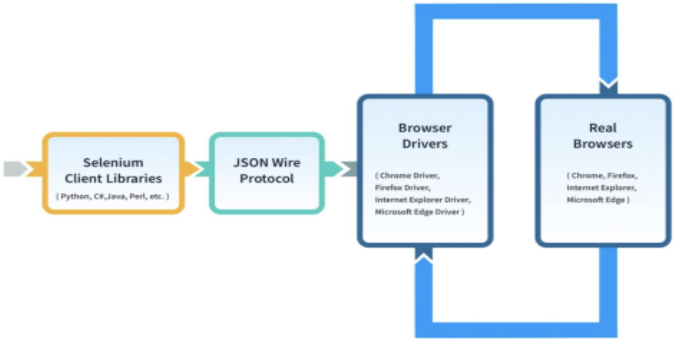
UI testing, also known as GUI testing, is a technique for testing the features of any software that a user will interact with usually involves testing the visual components to ensure they meet the outlined requirements, both functionality, and performance.

Computer Vision used in automation testing?
Computer vision is a scientific discipline that allows machines to 'visually' analyze their environment. Images/videos usually form the core of this process.



Tools in UI Testing :

Selenium :- it is an open-source tool for UI testing
How selenium works



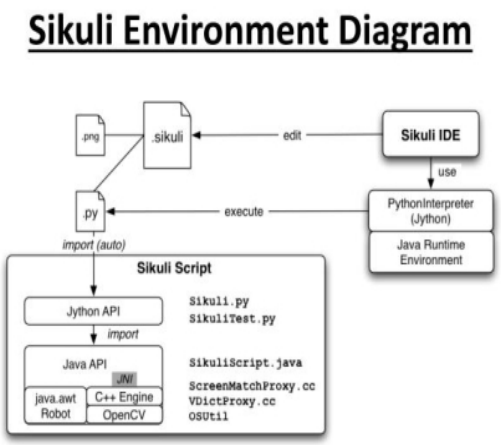
Which model is best for UI Testing

Why Sikuli tools are much better than selenium

Why UI Testing Is automation?
Automating UI tests **helps minimize the time and human effort required to test an application's UI**. Instead of having a tester validate every single aspect of the UI, test scripts are created for corresponding user scenarios

About Sikuli UI testing

- Sikuli automates anything you see on the screen of the user desktop computer running Windows, Mac, or some Linux/Unix. It uses image recognition powered by OpenCV to identify GUI components.

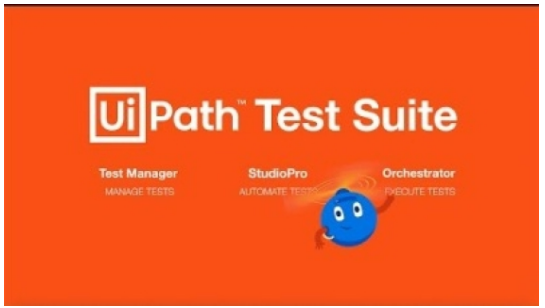


Other tools are there used for UI testing.

- Watir.
- Sahi.
- AutoIT.
- TestComplete .
- UI Path.

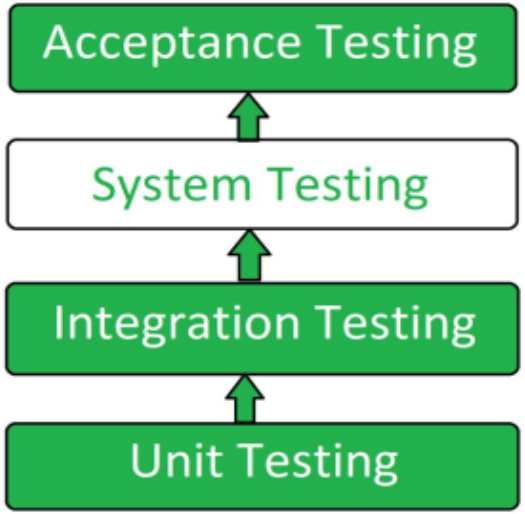
Interesting Facts:- **Why UI testing is regression testing.** Regression testing is a form of software testing that repeatedly executes functional and non-functional tests. These available tests frequently. UI is still looking good once a new feature has been implemented. In addition, it is a verification procedure that looks for any new flaw or fault in the current software.

What is RPA (Robotics Process Automation?
Video below



System Testing

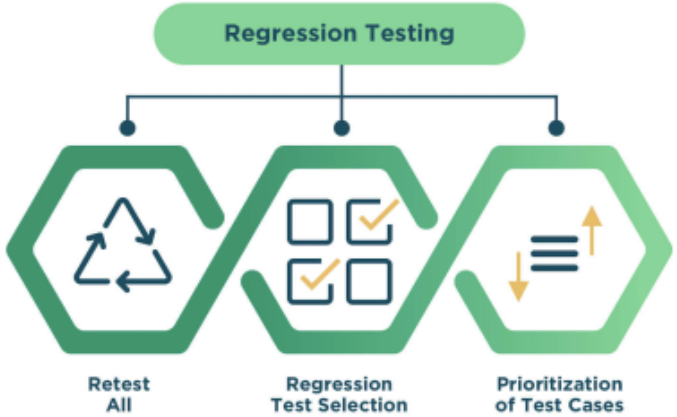
- What is system testing?
  - Testing of a complete and fully integrated software product.
  - In simple words testing how the external peripheral in-order to check how components interact with each other.



How Many types of System testing are there?

Regression Testing

- Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine



will ensure that issues that are already fixed do not occur again

- Automation tools can be used for this testing

This ensures that any bug fixes or enhancements that are done do not impact the exist in **Regression in Agile**

- Sprint Level Regression
- End to End Regression

Benefits of Regression Testing

- It improves the quality of product
- Thisg functionality of the product

Smoke and Sanity Testing

SMOKE TESTING

- Smoke Testing is a software testing process that determines whether the deployed software build is **stable or not**. It is also known as "**Build Verification Testing" or "Confidence Testing."**

1. **When do we do smoke testing**
   If new functionalities of software are developed and integrated with existing build.

2. **Who will do Smoke Testing**
   It is performed by QA engineers/QA lead.

3. **Why do we do smoke testing?**
   it ensures the correctness of the system in initial stages.

4. **How to do Smoke Testing ?**
   It is usually done manually but we can do the same through automation.

Summary:
Smoke test activity is the final step before the software build enters the system stage.

SANITY TESTING

- Testing done in a development environment on the code to ensure the correctness of the application before releasing build to QA, this is known as Sanity testing.
- It is usually narrow and deep testing.
- It checks the functionality works roughly as expected.

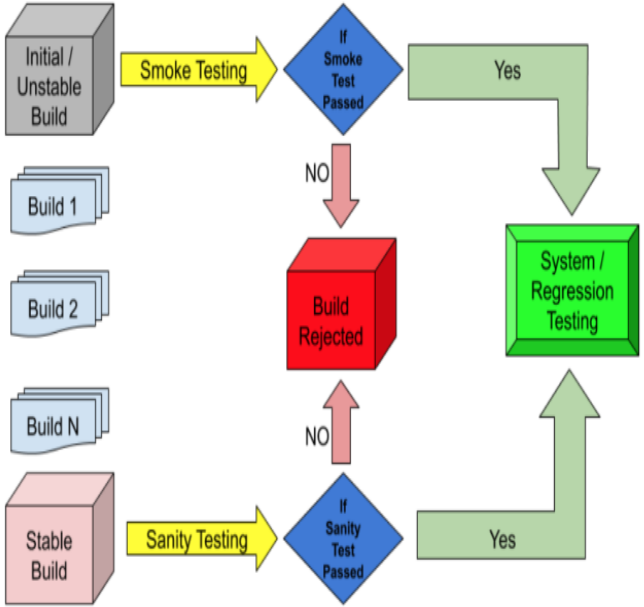
FUNCTIONAL TESTING : validates the software system
against the functional requirements/specifications.

Purpose of functional testing ?
Test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Summary:
basic test to quickly evaluate whether a claim or the result of a calculation can possibly be true.

Key Difference of Smoke and Sanity

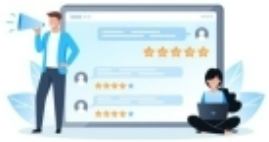
Smoke Test	Sanity Test
• Smoke Testing has a goal to verify “stability”	• Sanity Testing has a goal to verify “rationality”.
• Smoke Testing is done by both developers or testers.	• Sanity Testing is done by testers.
• Smoke Testing verifies the critical functionalities of the system.	• Sanity Testing verifies the new functionality like bug fixes.
• Smoke testing is a subset of acceptance testing	• Sanity testing is a subset of Regression Testing.
• Smoke testing is documented or scripted.	• Sanity testing isn’t.
• Smoke testing verifies the entire system from end to end.	• Sanity Testing verifies only a particular component.
• Smoke testing is like General Health Check Up.	• Sanity Testing is like specialized health Check Up.



Alpha Testing

- Done on application towards the end of development process when the product is almost in a usable state
- Carried out before release of the software
- Both white-box and Black-box testing are involved.
- Reveals the bugs that were not noticed during previous testing.
- Errors are quite easily detected and resolved during
- Some functionalities get missed for testing as the software is still under the development phase

Beta Testing



- Performed by real users of the software in real environment
- Carried out after release of the software
- Black-box testing is involved.
- Improves the quality of software with end user's feedback
- Increase customer satisfaction
- Not all the reviews are beneficial for software product

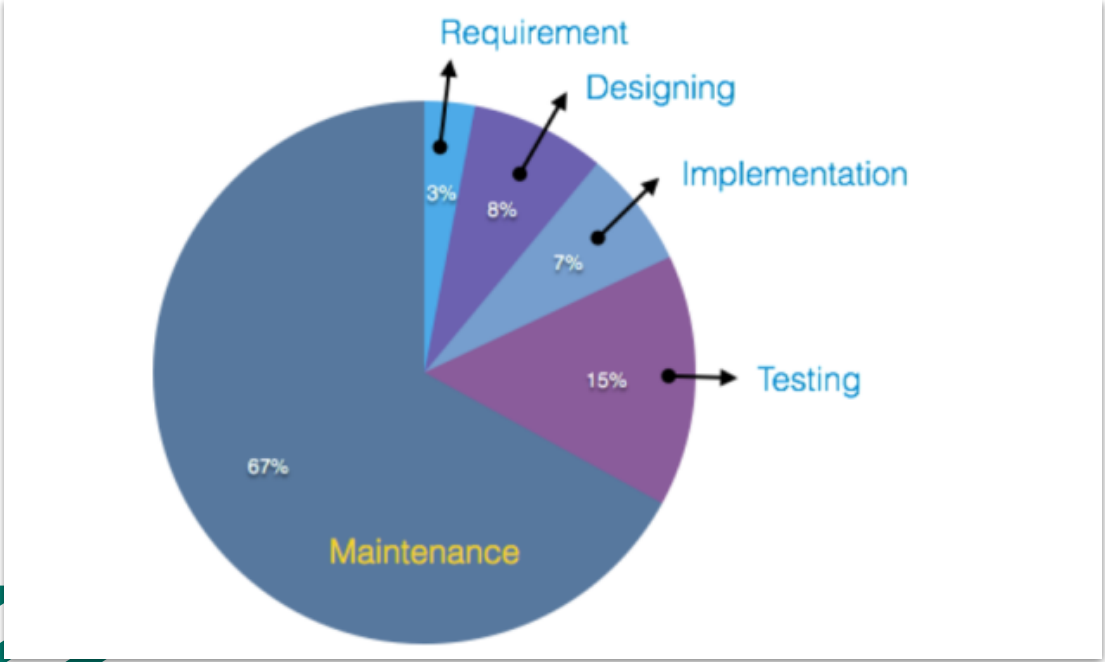
Thank You

Maintenance

Lemuel Rivera-Báez
Adit Shah
Shubham Sharma
Mohammad Firoz Rangwala
Rahul Kumar Sharma
Anirudh Poluri
Kavya Rayala
Shirisha Velineni
Kathankumar Shah


PREVENTION
PLANNING
DEVELOPMENT
FIXING
OPTIMIZATION
ANALYSIS
BUG
DEFECTS
TESTING
SOFTWARE
MAINTENANCE
PERFORMANCE
MAINTENANCE
ISSUE
IMPROVE
ADAPTIVE
CORRECTION
ENHANCEMENT
LIFE-CYCLE
FUNCTIONALITY
PROCESS
FAULTS

Cost of Maintenance:



Real-world factors affecting Maintenance Cost:

- The standard age of any software is considered up to 10 to 15 years.
- Older software, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced software's on modern hardware.
- As technology advances, it becomes costly to maintain old software.
- Most maintenance engineers are newbie and use trial and error method to rectify problem.
- Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
- Changes are often left undocumented which may cause more conflicts in future.

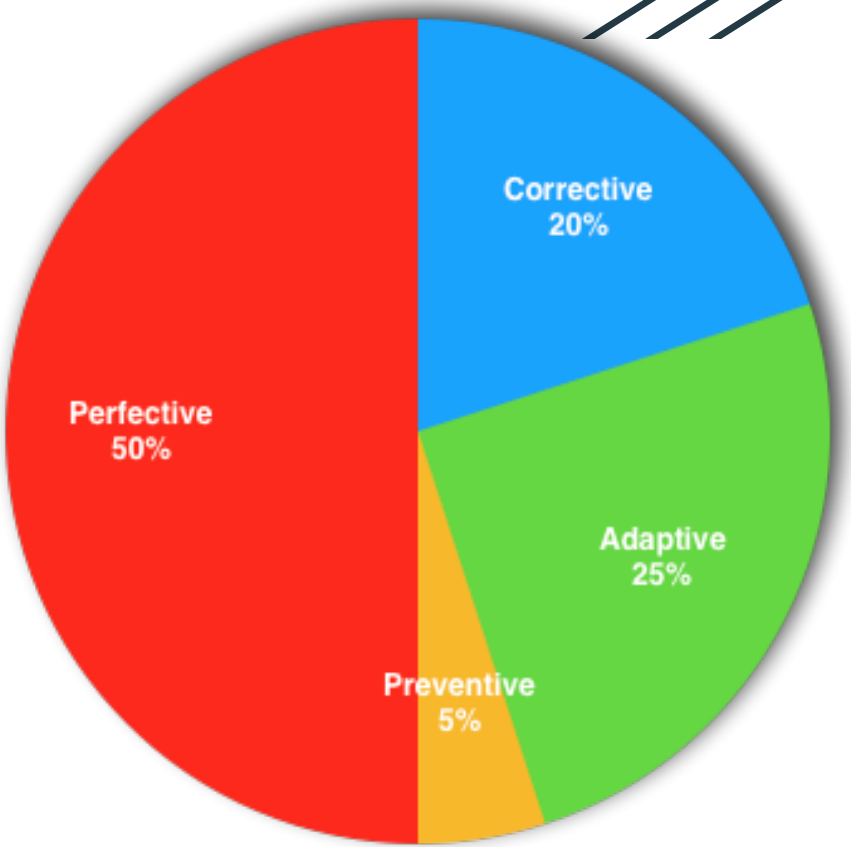
Software-end factors affecting Maintenance Cost:

- Structure of Software Program.
- Programming Language.
- Dependence on external environment.
- Staff reliability and availability.

Kinds of Maintenance Activities

1. **Corrective Maintenance**: Corrective maintenance is performed to rectify and repair faulty systems and equipment. The purpose of corrective maintenance is to restore systems that have broken down
2. **Adaptive Maintenance:** Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment.
3. **Perfective Maintenance**: Modification of a software product after delivery to improve performance or maintainability. Adapting to changing user requirements if often deemed to be perfective maintenance.
4. **Preventive Maintenance:** Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

Distribution of maintenance activities



Growth of Maintenance Problems

1975: ~75,000 people in maintenance (17%)

1990: 800,000 (47%)

2005: 2,500,000 (76%)

2022: ??

Shift in type of Maintenance over time

- **Introductory stage:** Priority on **User Support** that includes answering User questions and resolving problems that can be resolved by reading the Product Documentation. Usually this level of support is provided by End User's own internal resources, but may be provided by a third party.
- **Growth stage:** Basically **Fault Correction** stage. Includes analysis and modification or addition to repair, correct or circumvent a Fault in the Software, so that the Software operates substantially in accordance with the specifications for the Software.
- **Maturity**: Focus on **Enhancements** including modifications or improvements made to the Software which improve performance, capacity or provide additional functions to the Software.
- **Decline**: Implement **Technology Change**. This whole process helps in creating new markets and new market structures, and destroying obsolete markets.

Major causes of Maintenance Problems

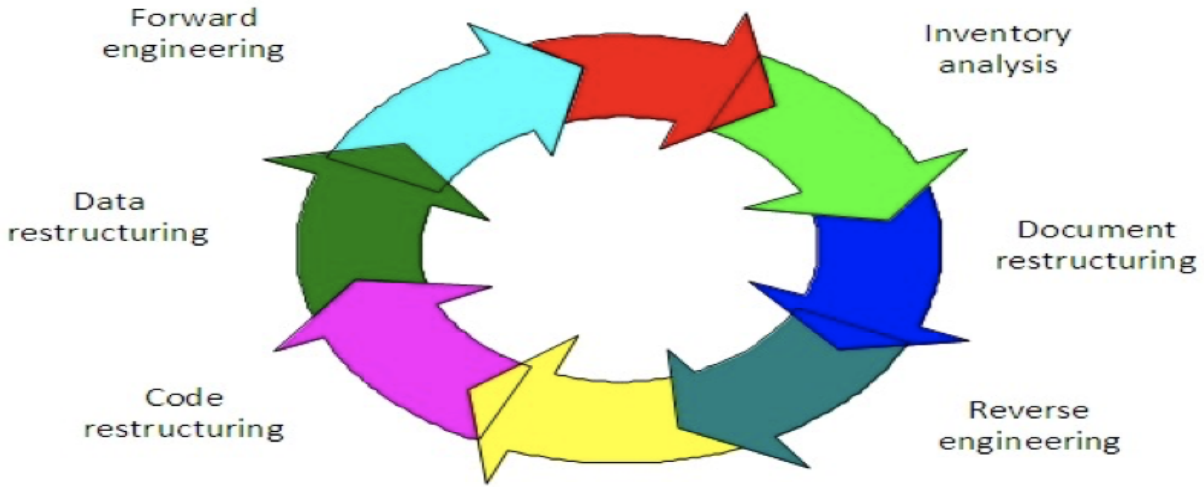
•**Unstructured code**: The **code is written as a single whole block**. The whole program is taken as a single unit. It is harder to do changes in the program.

•**Insufficient domain knowledge**: The lack of domain knowledge makes it difficult to apply the right methods as well as to judge their performance properly. In fact, the application of domain knowledge must be pervasive throughout the process in order for it to be effective.

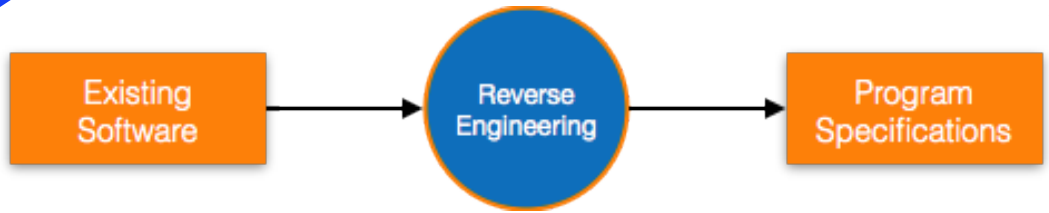
•**Insufficient documentation**: Life Cycle documents are not always efficiently produced even as part of a development project.

Software Re-engineering

When we need to update the software to keep it to the current market, without impacting its functionality, it is called software re-engineering. It is a thorough process where the design of software is changed and programs are re-written.



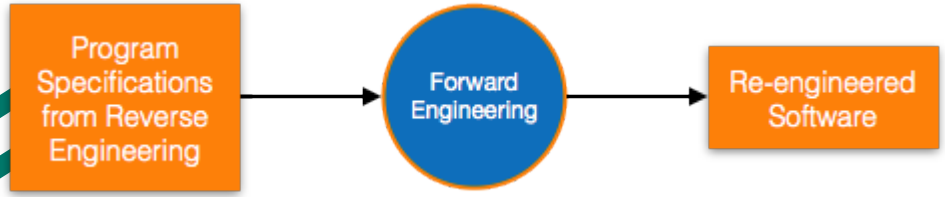
REVERSE ENGINEERING



PROGRAM RESTRUCTURING

It is a process to restructure and re-construct the existing software. It is all about re-arranging the source code, either in same programming language or from one programming language to a different one. Restructuring can have either source code-restructuring and data-restructuring or both.

FORWARD ENGINEERING



What is Refactoring?

- Restructuring techniques that allow us to change certain defects of the code.

Bad Smell Categories

- Bloaters
- Change Preventers
- Dispensables
- Encapsulators
- Couplers

Bad Smell Examples

- Long Method
- Large Class
- Primitive Obsession
- Long Parameter List
- Data Clumps
- Switch Statements
- Lazy Class
- Duplicate Code

Programming Plan:

- A program fragment that corresponds to a stereotypical action.

Programming Beacon:

- A key feature that typically indicates the presence of a particular structure or operation.

As-needed Strategy:

- Reading the program text from beginning to end like a piece of prose and forming hypotheses based on the presented information.

Systematic Strategy:

- Forming an understanding of the system through a systematic top-down study of the program text.

Tools to ease perceptual processes:

- Reformats the code to make it easier to analyze and understand.

Tools to gain insight of the static structure of programs:

- They allow us to see the basic structure of the code for the program and how the different components are connected.

Tools to gain insight in dynamic behavior:

- They allow us to monitor program executions. They are similar to debuggers.

Tools that inspect version history:

- Allows us to identify components that are changed very often.

Gap Model of Service Quality

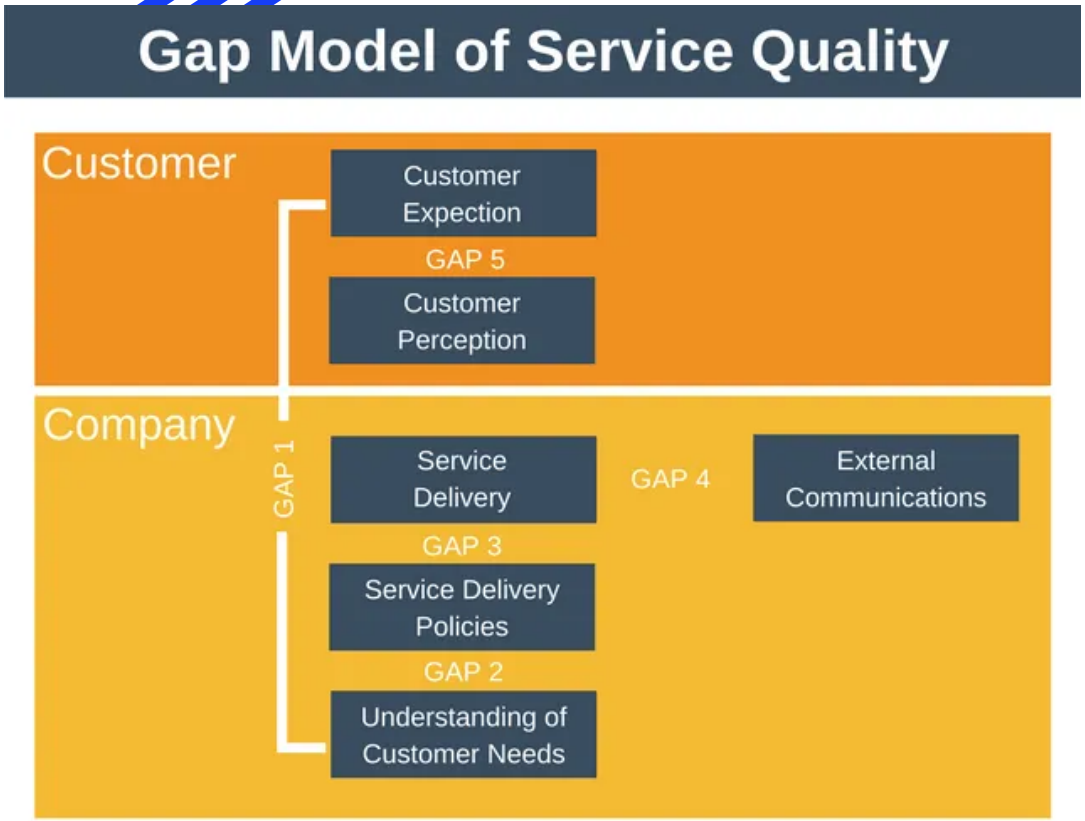
What is Gap in terms of service quality:

- The Gap Model of Service Quality (aka the Customer Service Gap Model or the 5 Gap Model) is a framework which can help us to understand customer satisfaction.

Aim:

- The aim of this model is to: **Identify the gaps between customer expectation and the actual services provided at different stages of service delivery**.

Service Gaps Model



Gap 1:Knowledge gap

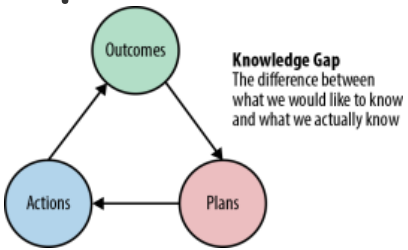
Gap 2: The Policy Gap

Gap 3: The Delivery Gap

Gap 4: The Communication Gap

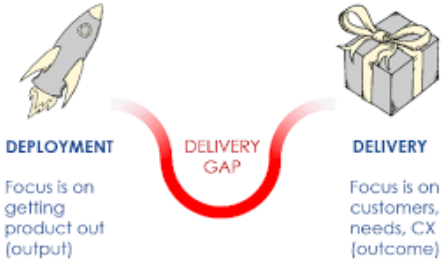
Gap 5: The Customer Gap

The Knowledge Gap



The Policy Gap

The Delivery Gap



The Communication Gap

The Customer Gap

Analyzing software evolution data

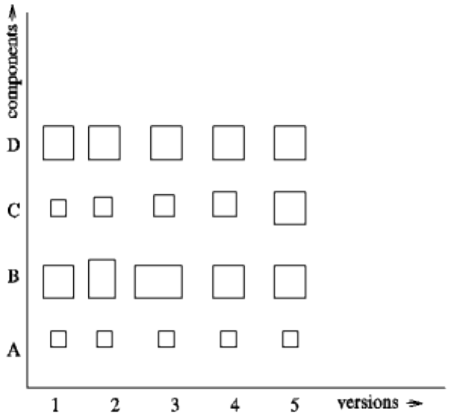
The evolution of software appears to follow a fairly predictable pattern. We may use this knowledge to try to forecast how a system will evolve in the future by looking at how it has evolved so far. Then, using information from the past, we decide what to do next. We can pick which components to reengineer by looking at components that have changed a lot recently, for example. The presumption is that components that have changed significantly in the recent past will likely change again in the near future.

- **Version-centered approaches** compare versions of a system with the aim of revealing *when* (i.e., in which version) a particular change occurred

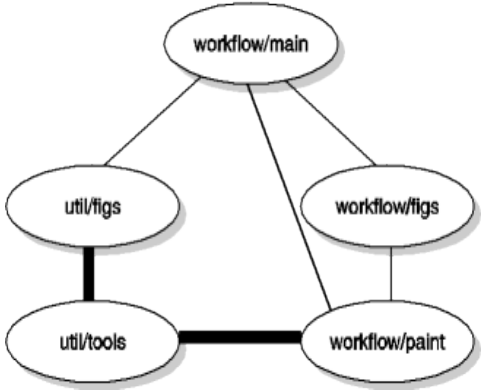
- **History-centered approaches** are concerned with revealing *what* the changes were and *where* these occurred, by summarizing the evolution according to a particular point of view.

Examples of both history-centered and version-centered evolution

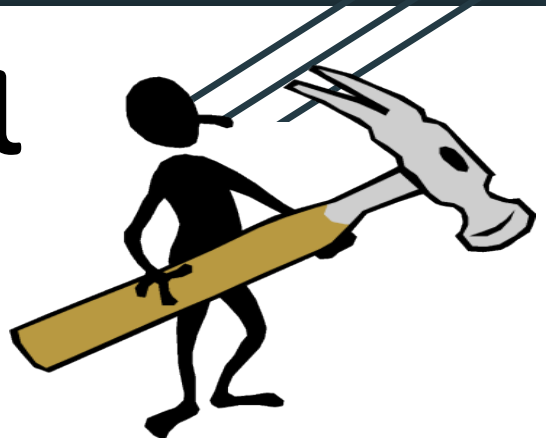
Version-Centered Analysis



History-Centered Analysis



Maintenance control



- Configuration control:
  - Identify, classify change requests
  - Analyze change requests
  - Implement changes

- Fits in with *iterative enhancement model* of maintenance (first analyze, then change)

- As opposed to *quick-fix model* (first patch, then update design and documentation, if time permits)

Indicators of system decay

- Frequent failures
- Overly complex structure
- Running in emulation mode
- Very large components
- Excessive resource requirements
- Deficient documentation
- High personnel turnover
- Different technologies in one system

Thank you for your time!

People Management

SE2: Group 6

Sandeep Yadav

Rohan Shetty

Avadhut Talbar

Kshitija Shete

Suchitra Subramani

Vaishali Vanjari

Lavanya Velagala

Introduction

- People management is the process of training, motivating and directing employees to optimize workplace productivity and promote professional growth.

- People have different goals

- People and productivity

- Coordination of work

- Importance of informal communication

Co-ordination Mechanisms

Why are co-ordination plans needed?

- To ensure productivity
- To have efficiency
- To maintain leadership
- To avoid Redundancy
- To avert inflexibility

Mintzberg Co-ordination mechanisms

- **Simple structure** - Simple structure organizes a vertical pyramid with lines of authority. Key characteristics include direct supervision of subordinates and organic organization . Authority is concentrated at the peak in the person of the CEO.
- **Machine bureaucracy (Standardization of work processes)** – This is a formalized management structure with a high degree of specialization where senior management takes major decisions . Machine bureaucracy is much more common in large, established companies than start-ups, young businesses, or small-to-medium enterprises
- **Divisionalized form (Standardization of work products)** - Large organizations with diversified products create divisions to handle related activities. The divisions can act with a high degree of autonomy to address their problems, while the central part of the organization concentrates on the big picture.
- **Professional bureaucracy (Standardization of worker skills)** - Professional bureaucracy relies on highly qualified professionals to carry out the work with a high degree of independence. It achieves its coordination of functions by standardizing the skills and qualifications required to carry out the work of a particular position.
- **Adhocracy (mutual adjustment)**-In project-based companies where projects are all different, formal structures limit the required flexibility. Adhocracy has characteristics such as a lack of formal structure combined with a variety of highly skilled employees. In this teams are self-organizing with matrix structures that allow effective vertical and horizontal sharing of authority based on competence and the conciseness.

Management Styles

Two dimensions in managing people

– Relation directedness:

This concerns attention to an individual and his relationship to other individuals within the organization.

– Task directedness:

This concerns attention to the results to be achieved and the way in which these results must be achieved.

Types of Management Styles

1. Separation style:
This management style is usually most effective for routine work. Efficiency is the central theme.

2. Relation style:
This style is usually most effective in situations where people have to be motivated, coordinated and trained.

3. Commitment style:
This is most effective if work is done under pressure.

4. Integration style:
This fits situations where the result is uncertain. The work is explorative in nature and the various tasks are highly interdependent.

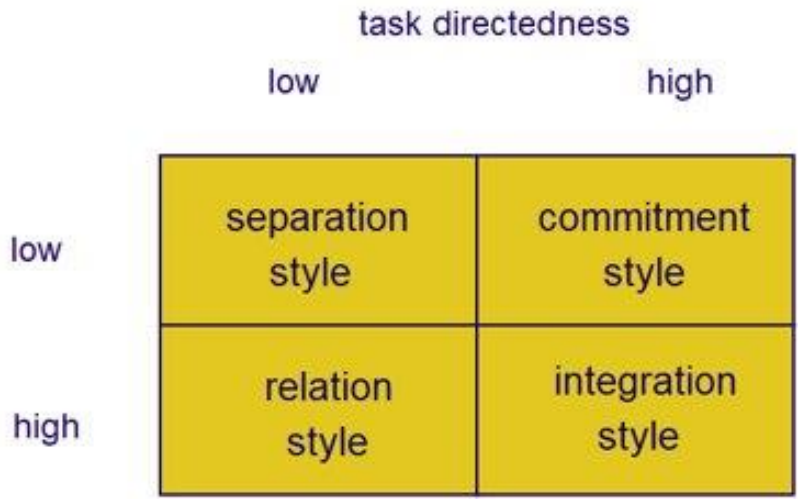
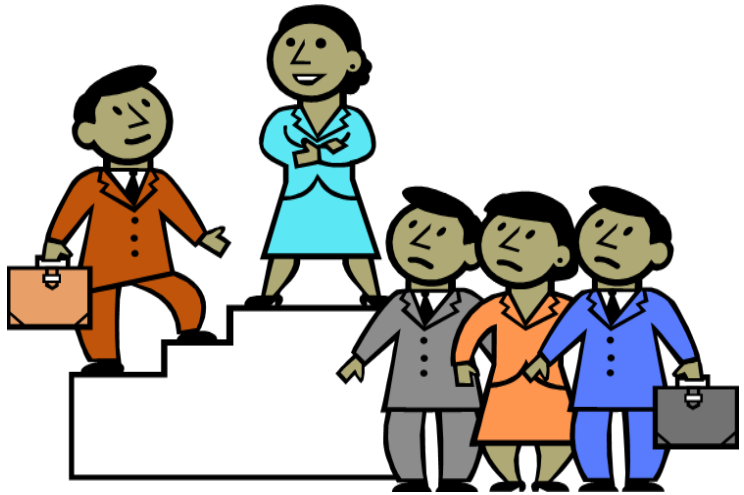


Figure: Four basic management styles, cf (Reddin, 1970)

Hierarchical Organization:-

- Most common type of organizational structure.
- High-Ranking people are at apex and Low-Ranking at the bottom.
- Chain of commands from top to bottom.



Advantages:-

- Clear lines of communication
- Clear understanding of employee roles and responsibilities.

Disadvantages:-

- Structure of unequal treatment
- Communication barriers

Matrix Organization:-

- Reporting to multiple leaders.
- Three types of matrix management: weak, balanced, strong.
- Mostly followed by small businesses.

Advantages:-

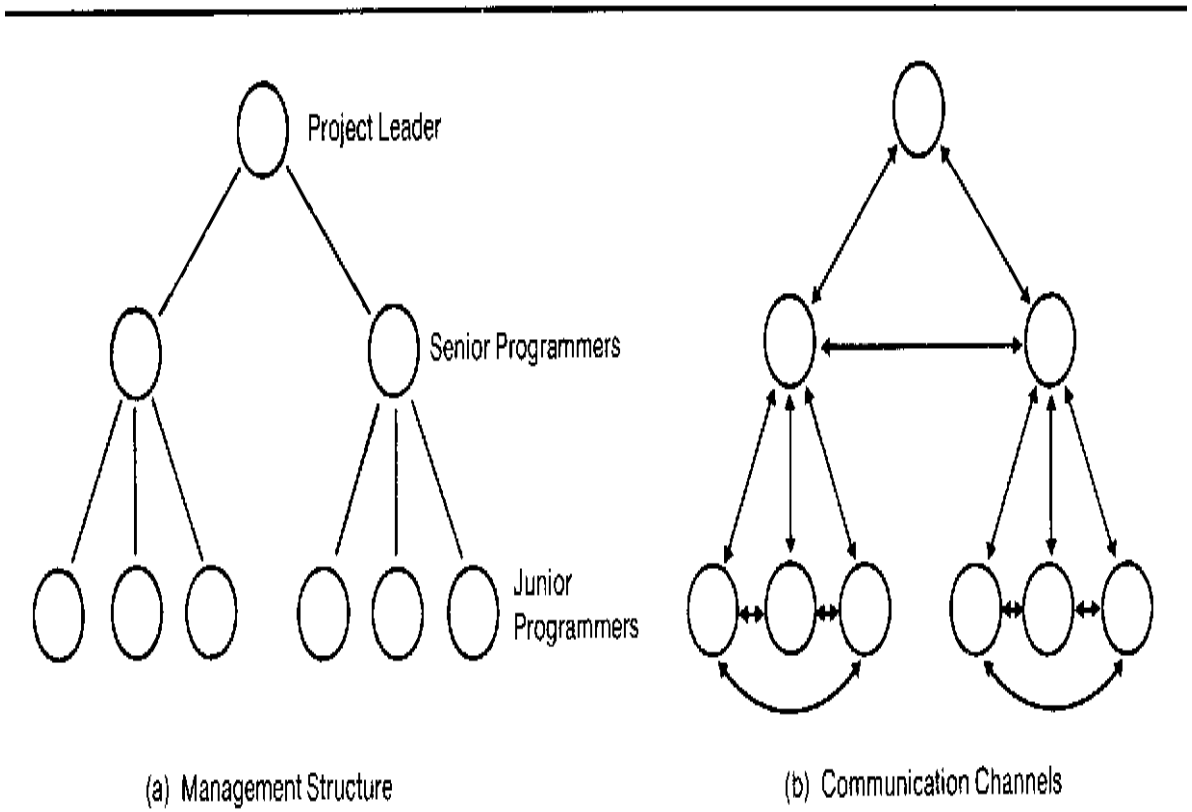
- Clear project objectives.
- Efficient use of resources.

Disadvantages:-

- Complex reporting style.
- Conflicting Guidance.

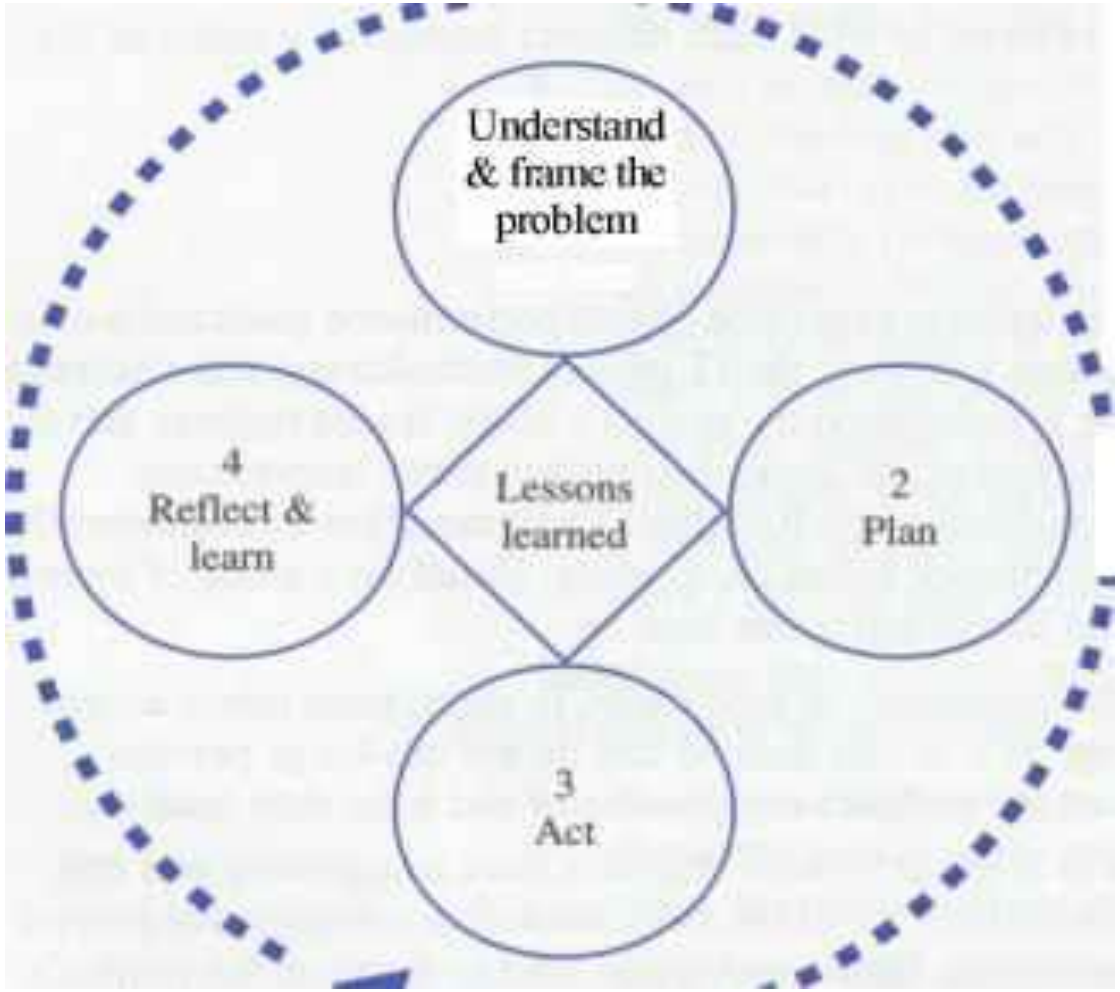
Chief Programmer Team

- The kernel of chief programmer team consists of three significant people :
  - 1)The chief programmer
  - 2)Assistant
  - 3)Librarian
- The most experienced persons act as chief programmer and deputy chief programmer, respectively.
- At the other end of the scale, one or two trainees can be assimilated and get the necessary on-the-job training.
- A trainee may well act as the team's librarian.



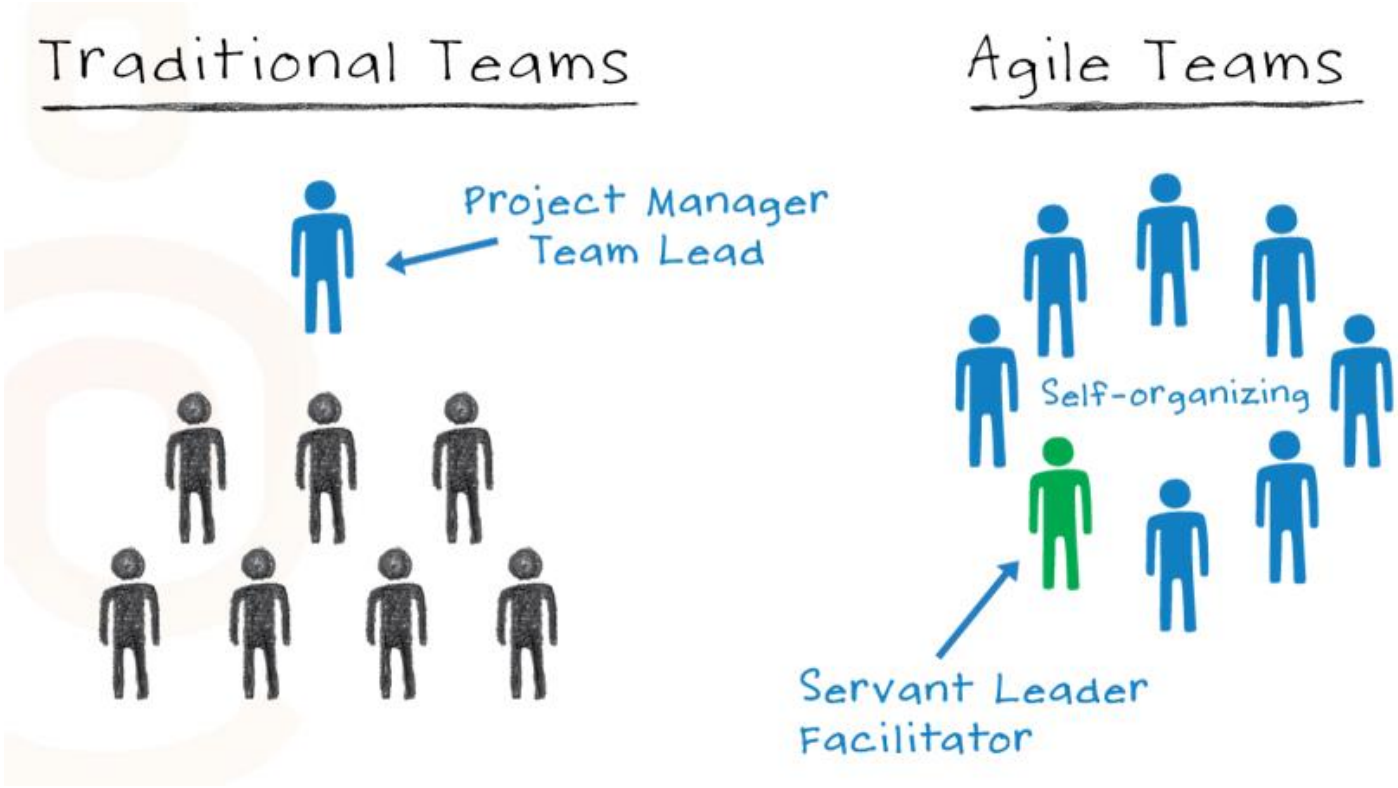
SWAT Team

- SWAT team as a software development version of a project team in which both task and relation directedness are high.
- A SWAT team typically builds incremental versions of a software system.
- As in the chief programmer team, the leader of a SWAT team is like a foreman in the building industry: he is considered to be both a manager and a co-worker.
- Team motivation is very important in a SWAT team.



Agile Team

- Cross-functional group
- Independent
- Mentorship
- Shared skillset
- Self-disciplined



Open-Source Software Development

- Core Team

  1) Experienced Software developers

  2) Responsible for changes in the software

- Co-developers

  1) Experienced Software developers

  2) Responsible for bug fixing and review of the code

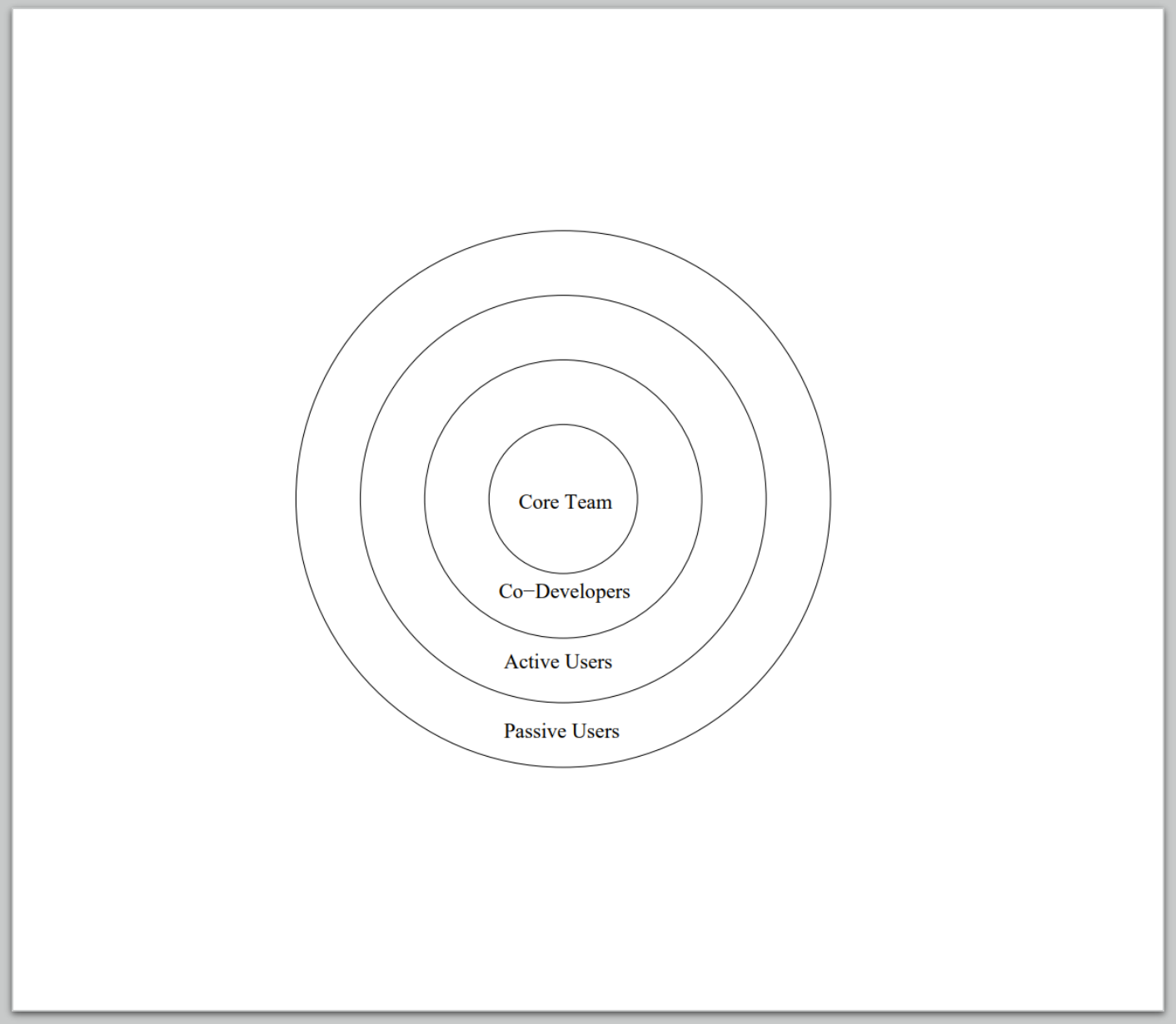
- Active users

  1) Users of the recently released system

  2) Requests new features and reports

- Passive users

  1) Merely users of the released stable software.



General Principles for Organizing a Team

- Use fewer, and better, people
- Try to fit tasks to the capabilities and motivation of the people available
- It is wise to select people such that a well-balanced and harmonious team results
- Someone who does not fit the team should be removed
- Do not pursue –

  The Reverse Peter Principle

  The Paul Principle