PACE
UNIVERSITY
Seidenberg School of Computer Science and Information Systems

CS631G Software Verification

INCIDENT AND PROBLEM MANAGEMENT

(W11)

Class instructor: Yuri Chernak, PhD

Outline

Section	Outline
Part I. Introduction	• ITIL: Incident and Problem Management Overview • ITIL Terminology • IEEE Std. 610 Definitions • Incident vs. Problem Management • Defect vs. Incident Management
Part II. Incident Management	• Why is incident management important? • Incident management process • The 3 types of incident management • Levels of Incident Management • Incident Management KPIs
Part III. Defect Management	• Defect Life Cycle • Reporting Software Defect • Defect Severity vs Defect Priority • Writing Defect Report • Defect Metrics
Part IV. Problem Management	• Problem Management Purpose • Value to Business • Process Activities and Techniques • Pareto Analysis Explained

Class Objectives

The objective of this class is to explain the ITIL functions – Incident and Problem Management, parts of the Service Operation phase of the ITIL Framework.

At the end of this class you will:

- Understand ITIL and IEEE terminology related to the Incident and Problem Management processes;
- Understand the objectives and importance of these functions;
- Understand the step-by-step workflow of the Incident Management.
- Understand Software Defect Management common practices.
- Get familiar with the Problem Management function.

Part I

Part I.
Introduction

What is ITIL?

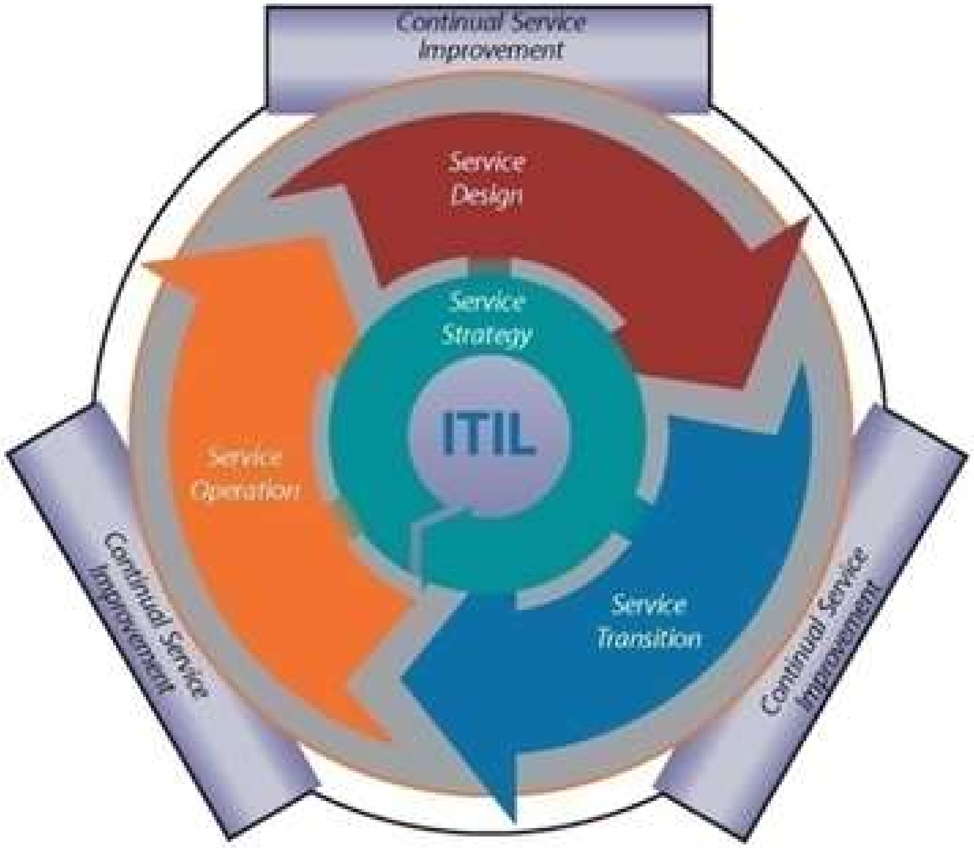
- IT organizations are continuously challenged to deliver better IT services at lower cost in a turbulent environment.
- Several management frameworks have been developed in the IT Industry to cope with this challenge.
- The IT Infrastructure Library (ITIL) is the most common framework that focuses on aligning IT services with the needs of business.
- The ITIL framework was originally developed in the 1980s in the UK at the same time when the CMM framework was being developed in the US.
- ITIL is maintained by the itSMF (the IT Service Management Forum) who publishes new ITIL editions (current ITIL 4 Edition, 2019).

ITIL Lifecycle Phases

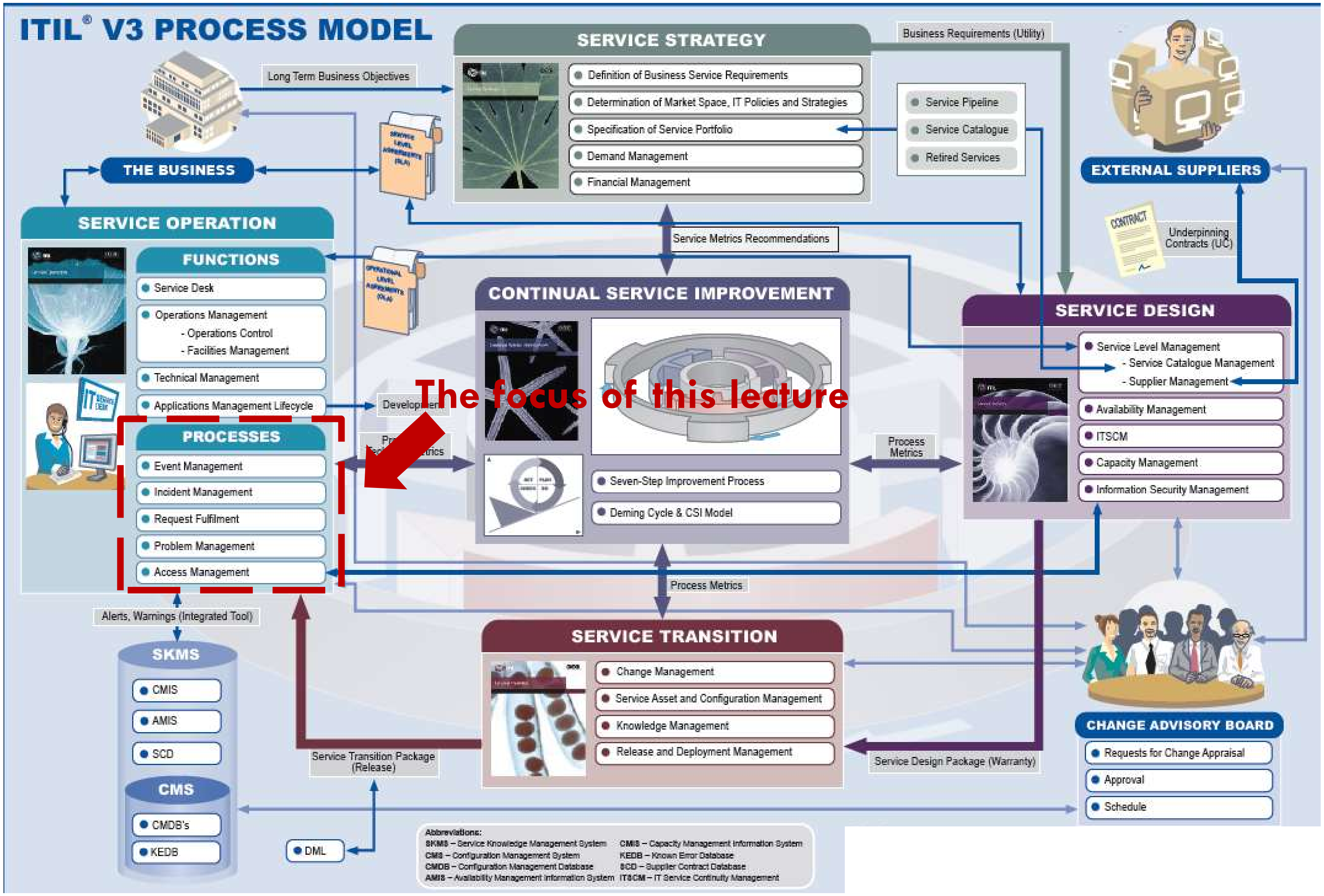
ITIL defines the Service Lifecycle as five phases:

- Service Strategy
- Service Design
- Service Transition
- Service Operation
- Continual Service Improvement

The focus of this lecture is Service Operation, which includes Incident Management and Problem Management.


Continual Service Improvement
Service Design
Service Strategy
ITIL
Service Operation
Service Transition
Continual Service Improvement
Continual Service Improvement

The ITIL Process Model


ITIL® V3 PROCESS MODEL
SERVICE STRATEGY
Definition of Business Service Requirements
Determination of Market Space, IT Policies and Strategies
Specification of Service Portfolio
Demand Management
Financial Management
Business Requirements (Utility)
Service Pipeline
Service Catalogue
Retired Services
Long Term Business Objectives
THE BUSINESS
SERVICE LEVEL AGREEMENTS (SLA)
EXTERNAL SUPPLIERS
CONTRACT
Underpinning Contracts (UC)
SERVICE OPERATION
FUNCTIONS
Service Desk
Operations Management
- Operations Control
- Facilities Management
Technical Management
Applications Management Lifecycle
PROCESSES
Event Management
Incident Management
Request Fulfilment
Problem Management
Access Management
OPERATIONAL LEVEL AGREEMENTS (OLA)
Service Metrics Recommendations
CONTINUAL SERVICE IMPROVEMENT
Seven-Step Improvement Process
Deming Cycle & CSI Model
Development
Process Metrics
Process Metrics
SERVICE DESIGN
Service Level Management
- Service Catalogue Management
- Supplier Management
Availability Management
ITSCM
Capacity Management
Information Security Management
Alerts, Warnings (Integrated Tool)
SKMS
CMIS
AMIS
SCD
CMS
CMDB's
KEDB
DML
SERVICE TRANSITION
Change Management
Service Asset and Configuration Management
Knowledge Management
Release and Deployment Management
Service Transition Package (Release)
Service Design Package (Warranty)
CHANGE ADVISORY BOARD
Requests for Change Appraisal
Approval
Schedule
Abbreviations:
SKMS – Service Knowledge Management System
CMS – Configuration Management System
CMDB – Configuration Management Database
AMIS – Availability Management Information System
CMIS – Capacity Management Information System
KEDB – Known Error Database
SCD – Supplier Contract Database
ITSCM – IT Service Continuity Management

The focus of this lecture

ITIL and IEEE Terminology

ITIL Definitions	IEEE Std. 610 Definitions
An incident is an unplanned interruption (or potential interruption) to or quality reduction of an IT service. **A major incident** is the highest category of impact for an incident. A major incident results in significant disruption to the business. **Service-Level Agreement** (SLA) A service-level agreement (SLA) is a contract between a service provider and its customers that documents what services the provider will furnish and defines the service standards the provider is obligated to meet. **Incident Management** is the process responsible for managing the lifecycle of all Incidents. The primary purpose of Incident Management is to restore normal IT service operation as quickly as possible. **Problem** is defined as the cause of one or more incidents. It is an underlying concern that, when resolved, can prevent incidents from occurring or recurring in the future.	**Fault.** (1) A defect in a hardware device or component; for example, a short circuit or broken wire. (2) An incorrect step, process, or data definition in a computer program. *Note:* This definition is used primarily by the fault tolerance discipline. In common usage, the terms "error" and "bug" are used to express this meaning. **Failure.** The inability of a system or component to perform its required functions within specified performance requirements. *Note:* The fault tolerance discipline distinguishes between a human action (a mistake), its manifestation (a hardware or software fault), the result of the fault (a failure), and the amount by which the result is incorrect (the error). Fault is exposed as a Failure of software.

Incident and Problem Management Explained

Reactive Measure

Incident Management is the process responsible for managing the lifecycle of all Incidents irrespective of their origination.

The goals for the Incident Management process are to:

- Restore normal service operation as quickly as possible;
- Minimize the adverse impact on business operations;
- Ensure that agreed levels of service quality are maintained.

Proactive Measure

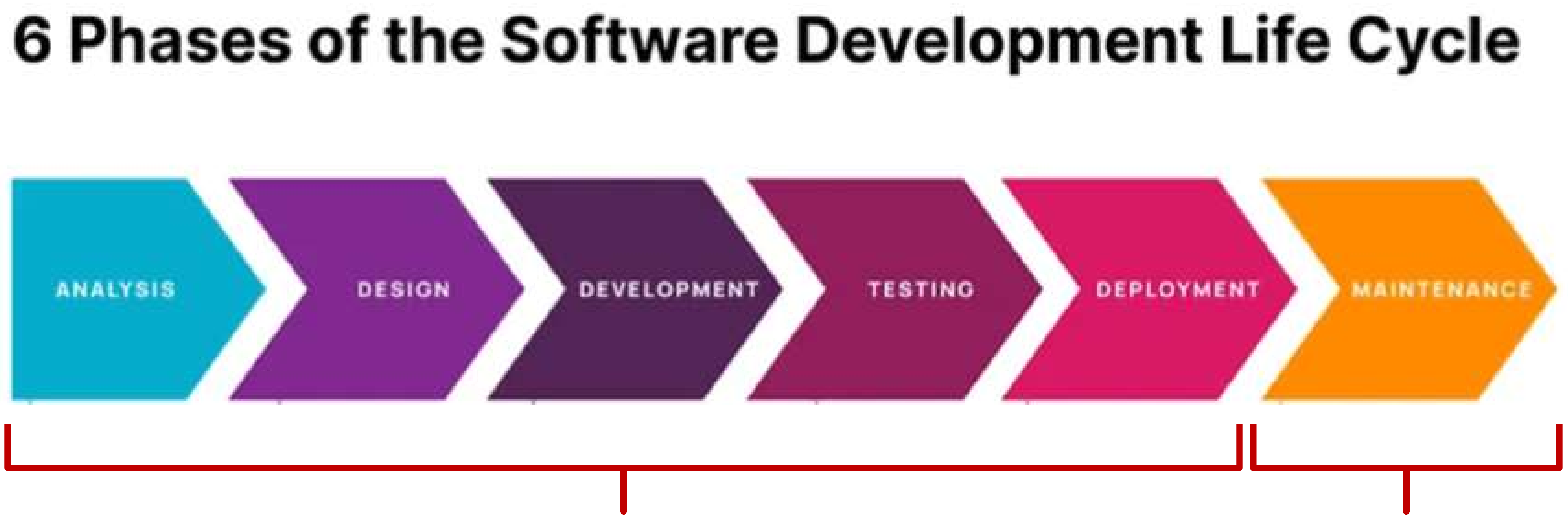
Problem Management is the process responsible for managing the lifecycle of all problems.

The goals of Problem Management process are to:

- Prevent problems and resulting incidents from happening to eliminate recurring incidents;
- Minimize the impact of incidents that cannot be prevented.

Defect vs. Incident Management Explained

Defect Management is the process of identifying, documenting, and tracking defects (bugs or issues) in a software product. It is an important part of the software development process that ensures defects are identified and addressed in a timely manner.



6 Phases of the Software Development Life Cycle

Defects are reported during the Software Development Cycle, most commonly found and reported during the Testing phase.

Incidents are reported from the Production Environment.

Part II

Part II.
Incident Management

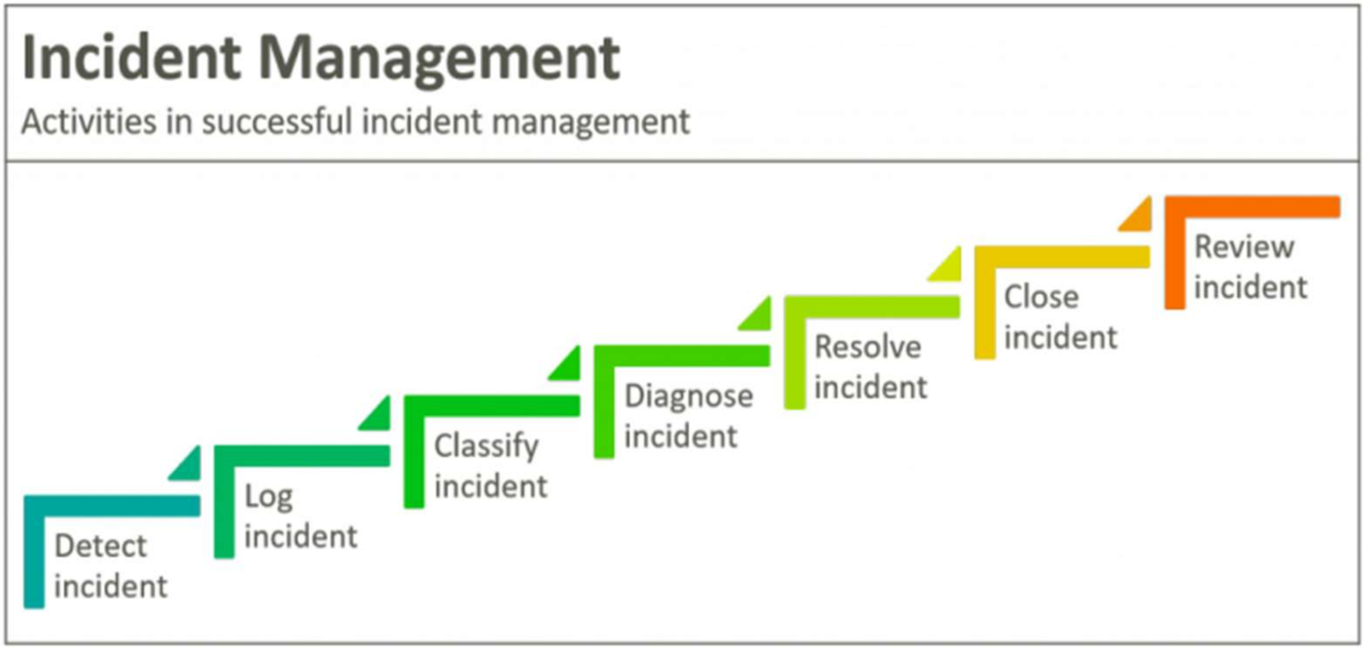
Why is Incident Management Important?

Incident management is a critical component of IT Service Management (ITSM), and the Information Technology Infrastructure Library (ITIL) provides a comprehensive framework for handling incidents effectively.
This approach contributes to enhanced service quality and stability:

Minimizing Downtime
ITIL incident management emphasizes a quick response to incidents. By promptly addressing and resolving issues, downtime is minimized, ensuring that IT services remain available and uninterrupted.
Improved User Satisfaction
Users value timely resolutions to their IT issues. Effective incident management ensures end-users experience minimal disruptions, increasing satisfaction and confidence in IT services.
Compliance and Reporting
Incident management in ITIL emphasizes documentation and reporting, which is crucial for compliance purposes and helps analyze trends, identify recurring issues, and implement preventive measures.
Risk Mitigation
Incidents often highlight potential risks in IT systems. Effective incident management helps identify these risks early, allowing IT teams to implement preventive measures and reduce the likelihood of future incidents.
Cost Efficiency
By swiftly resolving incidents and preventing their escalation, ITIL incident management contributes to cost efficiency. Avoiding prolonged downtime and business impact leads to financial savings and preserves organizational reputation.

Incident Management Process

- Successful Incident Management relies on having a clear understanding of what the customer agreed to or is willing to tolerate regarding the duration and handling of any particular incident.
- This is usually defined in service level agreements (SLAs) or contracts, which include timelines for responding and resolving incidents based on some criteria, usually priority, as a function of impact and urgency.
- The Incident Management process, as defined in ITIL, follows the steps:



Incident Management Process (Cont'd)

1. **Detect the incident:** Incident detection usually happens in one of two ways: A user reports a service issue and the service provider validates it as an incident.
2. **Log the incident:** The service provider logs the incident. This should register it in a system for purposes of proper management, including: Assigning the right handler to the incident Tracking the handling progress, particularly the timelines
3. **Classify the incident:** In the incident classification phase, the service provider categorizes the incident in terms of: Type Impact, as in who and what is affected Urgency, or the speed required for resolution Priority, with regard to business and customer perspectives Classification is useful for accelerating the process of identifying: Who should handle the incident What model, if any, is best suited Whether existing workarounds can be used.
4. **Diagnose the incident:** During incident diagnosis, the service provider investigates in order: Identify what has gone wrong, Determine the fastest way to recover normal service. Diagnosis can be done by one person (handler) where the symptoms relate to a previously known and documented incident.
5. **Resolve the incident:** Incident resolution refers to when the solution is applied.
6. **Close the incident:** Once the incident is resolved, formal incident closure of the record takes place.
7. **Review the incident:** During the incident review, sometimes known as an incident postmortem, the process owners or management may review how the incident was handled to determine what was done right and what went wrong.

The 3 Types of Incident Management

1. ITSM

ITSM teams are traditionally responsible for end-to-end management of IT services within an organization. In terms of incident management, ITSM teams strive to restore normal service operation as quickly as possible after an incident occurs, minimizing impact on business operations. They do this through established processes for incident identification, logging, categorization, prioritization, investigation, resolution, and closure. This approach tends to be more reactive, dealing with incidents after they've occurred.

2. Site reliability engineering (SRE)

SRE teams take a somewhat different approach to incident management. While they certainly address incidents as they occur, they also place a great emphasis on preventing incidents from happening in the first place. This involves designing systems to be robust and resilient, and continually measuring and improving system reliability. SRE teams often operate under a service level agreement that specifies a certain level of system uptime, and they aim to maintain system reliability within these agreed parameters.

3. DevOps

DevOps teams address incident management with a focus on continuous delivery and infrastructure as code. Incidents are often seen as opportunities for improvement, and the team's response will typically involve not only resolving the immediate problem, but also adjusting the development and deployment processes to prevent similar incidents in the future. This might involve making changes to the code, updating automated tests, or enhancing monitoring and alerting capabilities.

In summary, ITSM teams focus on aligning IT services with business needs and tend to be more reactive. SRE teams aim to build robust systems and prevent incidents from occurring. DevOps teams view incidents as opportunities for improvement and aim to adjust their processes to prevent recurrence. Each approach has its strengths, and many organizations will use a combination of these strategies to manage incidents effectively.

Levels of Incident Management

In order to better serve the customers or users, services are divided into Tiers or Levels. As a result, a common support structure revolves around a three-layered system. Tier 1 support or Level 1 IT support, followed by Tier 2 & Tier 3 tech support.

Tier 1/L1: Line staff who are the subject matter experts for assessing, planning and monitoring Incident Management for their functional organization and specific technology platform. They function as contact people between the different departments for a specific process and may be responsible for the design of processes within their own departments.

Tier 2/L2: More in-depth technical support than tier 1. Tier 2 support personnel may be more experienced or knowledgeable on a particular product or service. Additionally, Tier 2 may be able to provide onsite troubleshooting and/or resolution. Specialized departments (i.e. Networks, Servers, Video) will provide Tier 2 Support in their respective areas of expertise.

Tier 3/L3: This is the highest level of support in a three-layered technical support model responsible for handling the most difficult or advanced problems. It is synonymous with L3 support denoting expert troubleshooting and resolution methods.

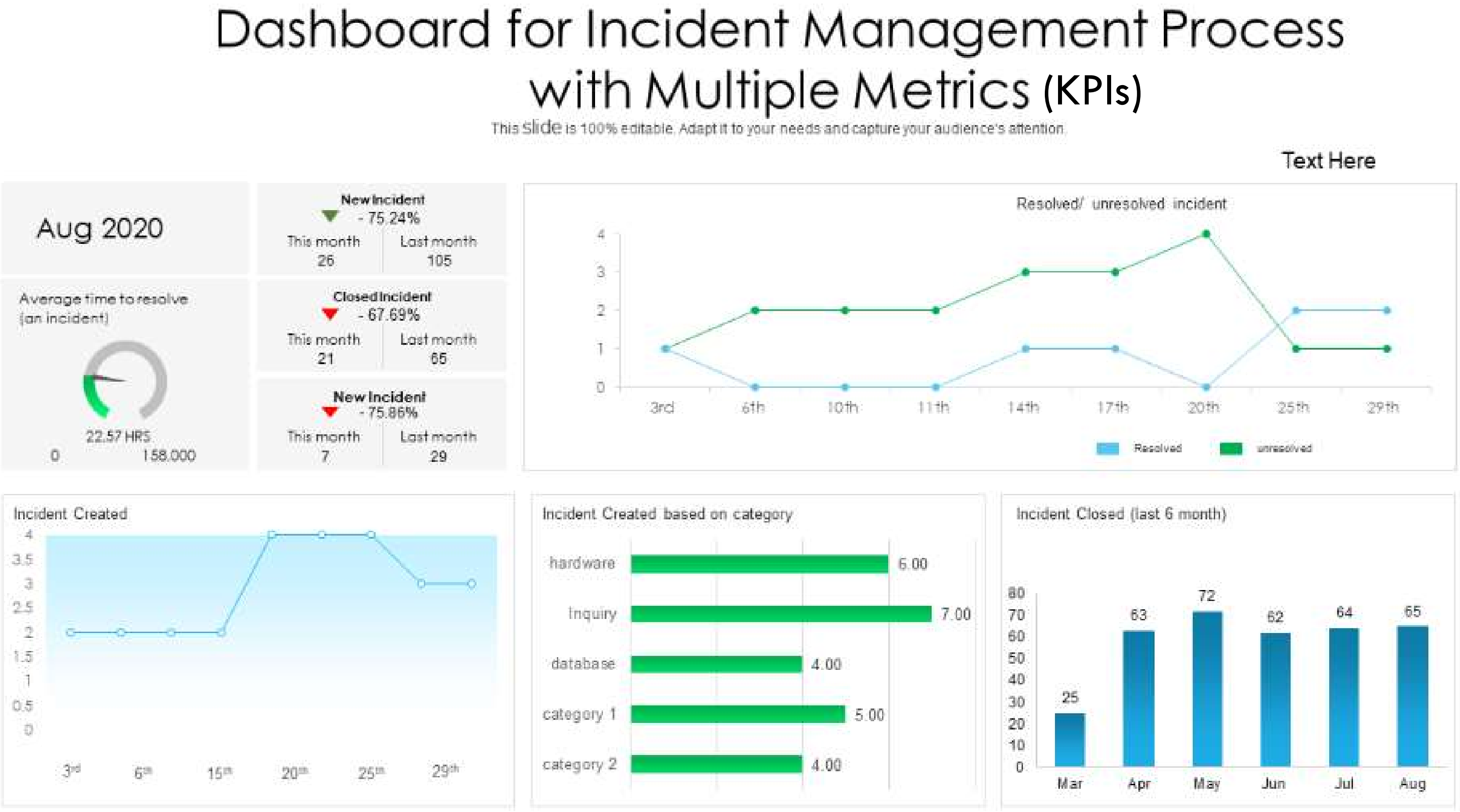
Incident Management KPIs

- Production incidents is a critical concern for Organization’s Management.
- To provide visibility into the state of the production stability, Production Management department collects metrics (KPI) and analyzes them on regular management calls.

Examples of Key Performance Indicators (KPIs):

1. **Incident resolution time:** Measures the time taken to resolve incidents from the moment they are reported.
2. **Resolution effectiveness:** number of incidents reopened.
3. **Incident trends:** Tracks the frequency and patterns of incidents over time.
4. **Customer self-service:** Number of self service tickets via a customer portal verses tickets created by the Service Desk.
5. **First call resolution rate:** Evaluate the percentage of incidents resolved during the initial contact with the service desk.

Example: Incident Management Dashboard



Dashboard for Incident Management Process with Multiple Metrics (KPIs)

This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here

Aug 2020

Average time to resolve (an incident)

22.57 HRS

0 158,000

New Incident
- 75.24%

This month	Last month
26	105

Closed Incident
- 67.69%

This month	Last month
21	65

New Incident
- 75.86%

This month	Last month
7	29

Part III

Part III.
Defect Management

Reporting Software Defects

- The most important mission of testers is finding and reporting software defects.
- Defects are found in deliverables produced by other team members, e.g., business analysts (requirements) or developers (source code).
- Once a defect is found, it needs to be communicated to a party who is the owner of the deliverable.
- However, it is not always evident where a fix should be applied, and some investigation of the defect origin is commonly needed.
- Hence, defect reporting is a communication process where a given defect changes hands from the person who found it to the next party who investigates it, then to the party who fixes it, etc. *See a defect life cycle on the next slide.*
- Once the defect is fixed, it goes back to the Tester for re-testing and closing.
- On large projects, the number of new defects reported daily can be from 15 to 30 defects. On such projects effective defect management becomes a critical activity:
  - Defect Management = New Defect Prioritization + Defect Status Tracking

Example: Defect Management Life Cycle

Each defect goes through a stages of the life cycle shown in the example chart.

Stages include:

New: When a defect is logged and posted for the first time. It's state is given as new.

Assigned: After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. It's state given as assigned.

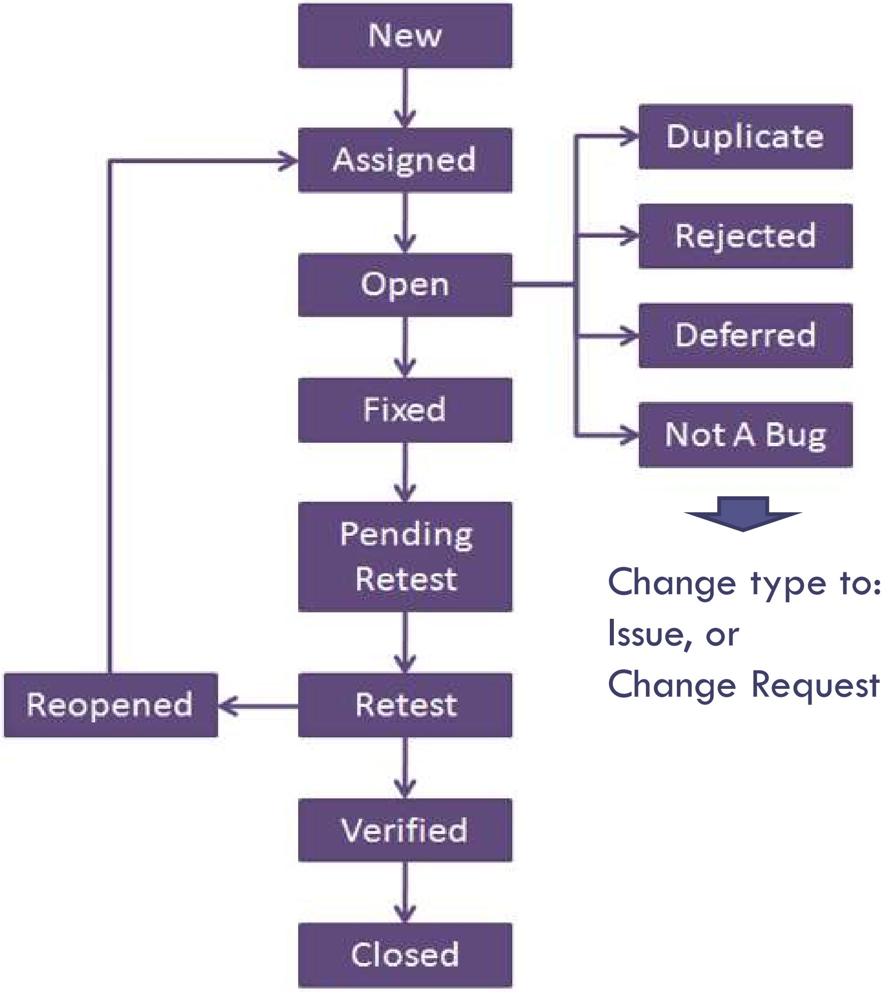
Open: At this state the developer has started analyzing and working on the defect fix.

Fixed: When developer makes necessary code changes and verifies the changes then he/she can make bug status as 'Fixed' and the bug is passed to testing team.

Pending retest: After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.

Retest: At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

Verified: The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "verified".



Change type to:
Issue, or
Change Request

Bug / Defect Lifecycle

Defect Management Life Cycle (Cont'd)

Stages include (continued):

Reopen: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again.

Closed: Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.

Duplicate: If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to "duplicate**".**

Rejected: If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to "rejected".

Deferred: The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are **priority** of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

Not a bug: The state given as "Not a bug" if there is no change in the functionality of the application. For example: If a customer asks for some change in the look and feel of the application like change of color of some text then it is not a bug but just some change in the look of the application.

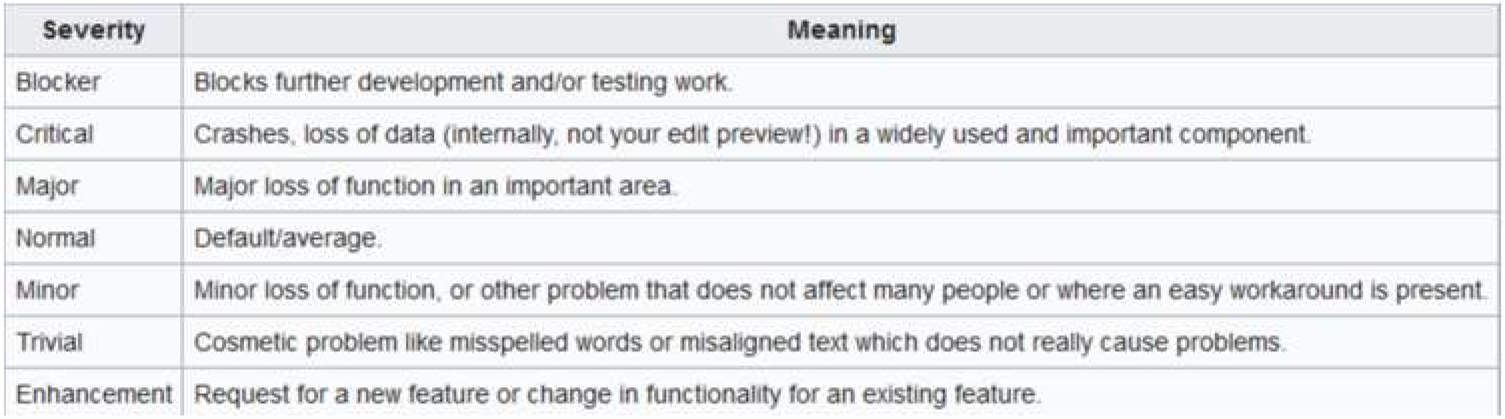
Defect Severity vs. Defect Priority

Defect Priority (Bug Priority) indicates the importance or urgency of fixing a defect:

- **High priority** for tasks committed for the current sprint, or that need to find an owner who can start working on them soon.
- **Normal priority** for tasks that are not critical for the current sprint or candidates for a next sprint.
- **Low priority** for tasks that we can live without, usually sitting in the backlog, sometimes added to a sprint.

Defect Severity is the degree of impact that a defect has on the development or operation of a component or system.

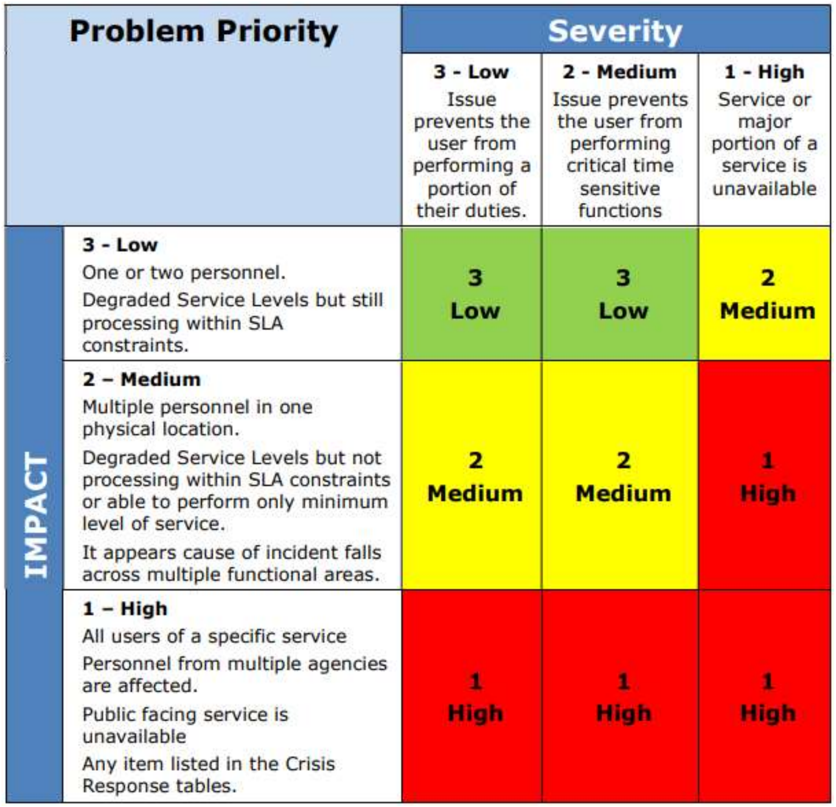
Severity Common Classification



Severity	Meaning
Blocker	Blocks further development and/or testing work.
Critical	Crashes, loss of data (internally, not your edit preview!) in a widely used and important component.
Major	Major loss of function in an important area.
Normal	Default/average.
Minor	Minor loss of function, or other problem that does not affect many people or where an easy workaround is present.
Trivial	Cosmetic problem like misspelled words or misaligned text which does not really cause problems.
Enhancement	Request for a new feature or change in functionality for an existing feature.

Defect Severity vs. Defect Priority Dependency

Severity and Priority are often correlated, i.e., Priority is assigned based on analyzing the Severity level. However, in the defect report form they are separate management fields.



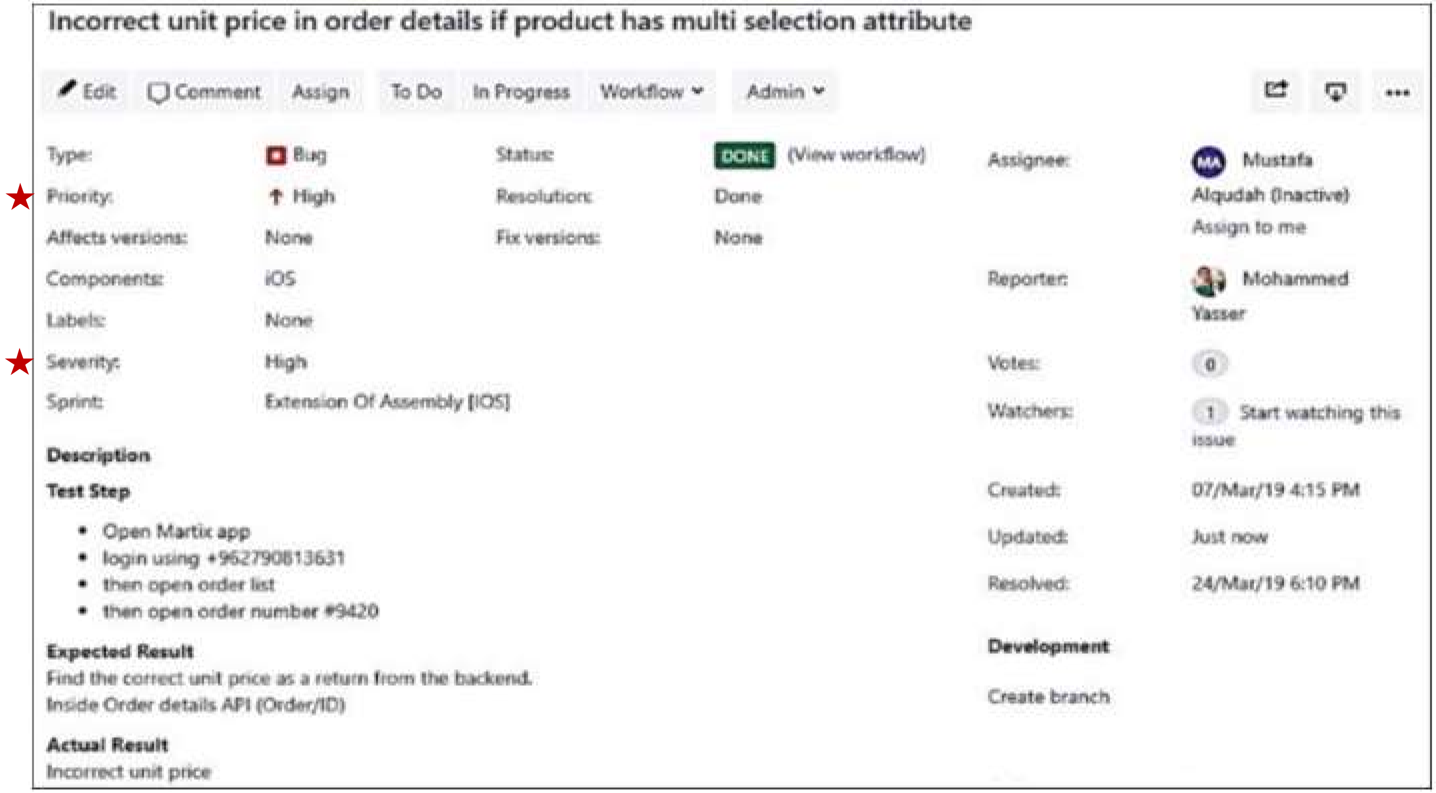
Problem Priority		Severity		
		3 - Low Issue prevents the user from performing a portion of their duties.	**2 - Medium** Issue prevents the user from performing critical time sensitive functions	**1 - High** Service or major portion of a service is unavailable
IMPACT	**3 - Low** One or two personnel. Degraded Service Levels but still processing within SLA constraints.	**3** **Low**	**3** **Low**	**2** **Medium**
	2 – Medium Multiple personnel in one physical location. Degraded Service Levels but not processing within SLA constraints or able to perform only minimum level of service. It appears cause of incident falls across multiple functional areas.	**2** **Medium**	**2** **Medium**	**1** **High**
	1 – High All users of a specific service Personnel from multiple agencies are affected. Public facing service is unavailable Any item listed in the Crisis Response tables.	**1** **High**	**1** **High**	**1** **High**

Tips to Write Effective Defect Reports

Tips	Descriptions
It's all in Defect Title	The defect title should always be meaningful and self-descriptive enough to tell about the defect. Keep the title short & should properly convey the defect summary without further reading. Each defect should be clearly identifiable by the defect title.
Steps To Reproduce The Defect	Software defects are very sensitive & they occur when there is a condition in a software product which does not meet a software requirement or end-user expectations. While reporting the defect, you should add the proper steps to reproduce the defect.
Be Specific	The defect description should be to the point and do not write down the essay about the problem.
Defect Linking	Add the references to the specifications or other related defect number while adding the defect. For example, in Jira defects can be linked to their related user stories.
Don't take business decisions, only pass the Information	You should be objective while reporting defects. You should add the actual result in defect & the expected result if it is specified in the requirement specification document. Don't take business decisions if anything is not clear , ask for additional information, nut do not pass any kind of judgments by your own.
Capture ***Severity*** and ***Priority*** of defects	Severity & Priority defect attributes play a vital role in the defect management. The **Severity** describes the impact of the bug on application functionality. The **Priority** indicates how quickly a bug should be fixed (*see next slide*).

Example: Jira Bug Report

Management fields on a defect report form are frequently customized to better support the project needs.



Incorrect unit price in order details if product has multi selection attribute

Type:	Bug	Status:	DONE (View workflow)	Assignee:	Mustafa Alqudah (Inactive) Assign to me
★ Priority:	High	Resolution:	Done		
Affects versions:	None	Fix versions:	None		
Components:	iOS			Reporter:	Mohammed Yasser
Labels:	None				
★ Severity:	High			Votes:	0
Sprint:	Extension Of Assembly [IOS]			Watchers:	1 Start watching this issue

Description

Test Step

- Open Martix app
- login using +962790813631
- then open order list
- then open order number #9420

Expected Result

Find the correct unit price as a return from the backend.
Inside Order details API (Order/ID)

Actual Result

Incorrect unit price

Created:	07/Mar/19 4:15 PM
Updated:	Just now
Resolved:	24/Mar/19 6:10 PM

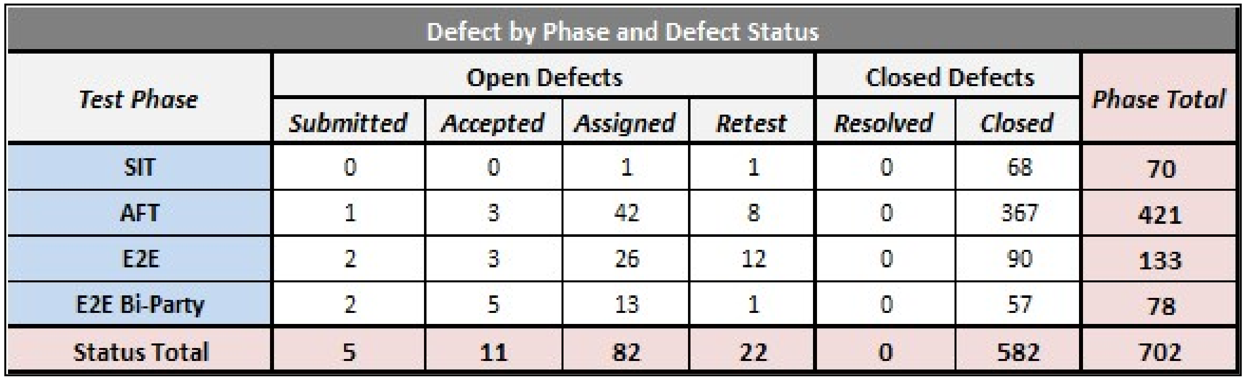
Development

Create branch

Examples of Defect Metrics

Producing and communicating defect metrics can help a test manager and project stakeholders assess how effective the testing is and when it can be completed.

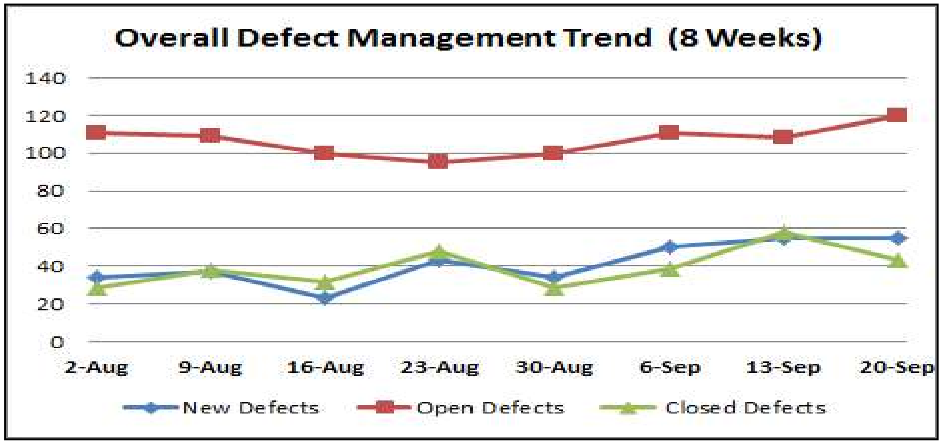
Distribution of defects by test type and status.



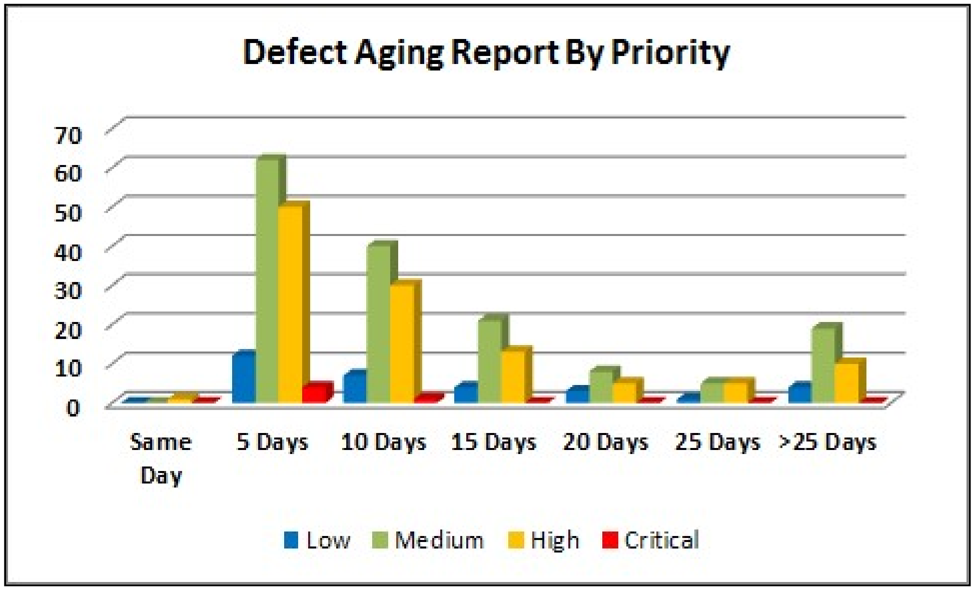
Defect by Phase and Defect Status

Test Phase	Open Defects				Closed Defects		*Phase Total*
	Submitted	*Accepted*	*Assigned*	*Retest*	*Resolved*	*Closed*	
SIT	0	0	1	1	0	68	70
AFT	1	3	42	8	0	367	421
E2E	2	3	26	12	0	90	133
E2E Bi-Party	2	5	13	1	0	57	78
Status Total	5	11	82	22	0	582	702

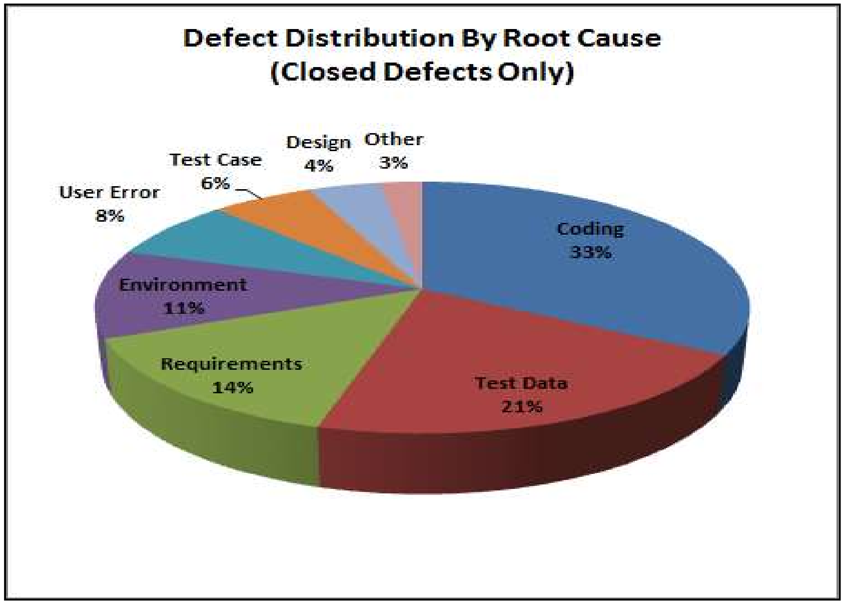
Trend of defects by status.



Examples of Defect Metrics (continued)



The **defect aging** report can help manage commitments to investigate and fix defects.



The **defect root-cause** report can help understand the main reasons why defects were introduced.

Part IV.
Problem Management

What is Problem Management?

- **Problem Management** is a core component of the ITSM framework, and is the process for identifying and managing root causes and potential IT incidents.
- Problem Management identifies and manages problems using preventative methods and identifying underlying causes to help prevent future issues. With a structured workflow for diagnosing root causes and fixing problems, it helps eliminate recurring incidents and minimize the impact of unexpected disruptions.
- Problem Management makes it possible to identify the root cause of service-affecting problems, determine the resolution to those problems, and can likewise help prevent issues before they occur.
- It is also responsible for ensuring that the resolution is implemented through the appropriate control procedures, especially Change Management and Release Management.
- Problem Management will also maintain information about problems and the appropriate workarounds and resolutions, so that the organization is able to reduce the number and impact of incidents over time.
- In this respect, Problem Management has a strong interface with Knowledge Management, and tools such as the Known Error Database will be used for both.

Problem Management Benefits

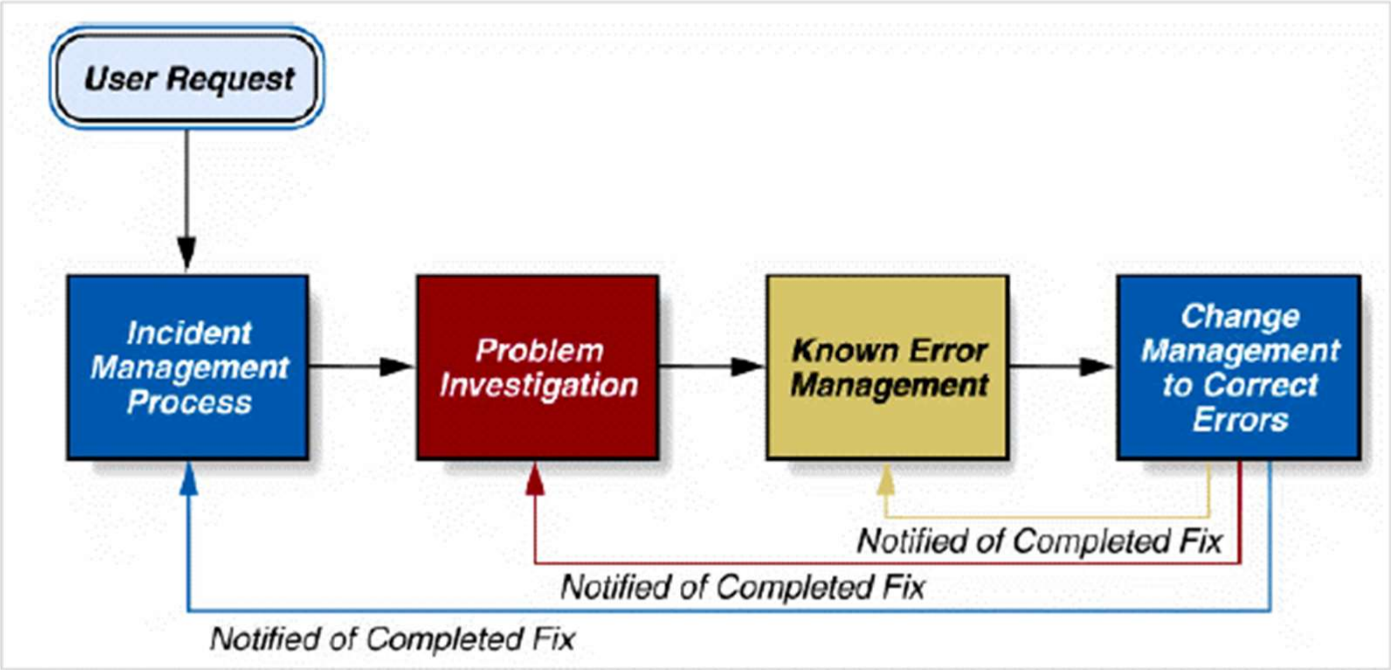
ITIL Problem Management is a crucial aspect of IT service management that aims to detect and remove the root cause of problems that affect IT services.
By finding and eliminating the root cause of problems, problem management can have several benefits when executed correctly.

The primary benefits include the following:

- Continuous service improvement
- Avoiding costly incidents
- Increased productivity
- Decreased time to resolution
- Increase customer and employee satisfaction
- Speed up service restoration
- Minimize service disruptions

Problem Management vs. Other Processes

- Problem Management focuses on determining the root cause of a problem, and on using the Change Management process to correct the root cause.
- The following chart shows the relationship between Incident Management, Problem Management, and Change Management processes for user requests.


User Request
Incident Management Process
Problem Investigation
Known Error Management
Change Management to Correct Errors
Notified of Completed Fix
Notified of Completed Fix
Notified of Completed Fix

Problem Management vs. Other Processes (Cont'd)

Problem Management vs Knowledge Management

The management of knowledge is the creation of a repository of documentation and solutions that include incidents and problems. Knowledge management is used to help in the solving of problems. When a known error is documented, a single click generates a Known Error Article in the knowledge base—saving time and effort in fixing recurring issues in the future.

Problem Management vs Incident Management

Problems are potential causes of one or several incidents. While problem and incident management overlap in a few disciplines, there are key differences.

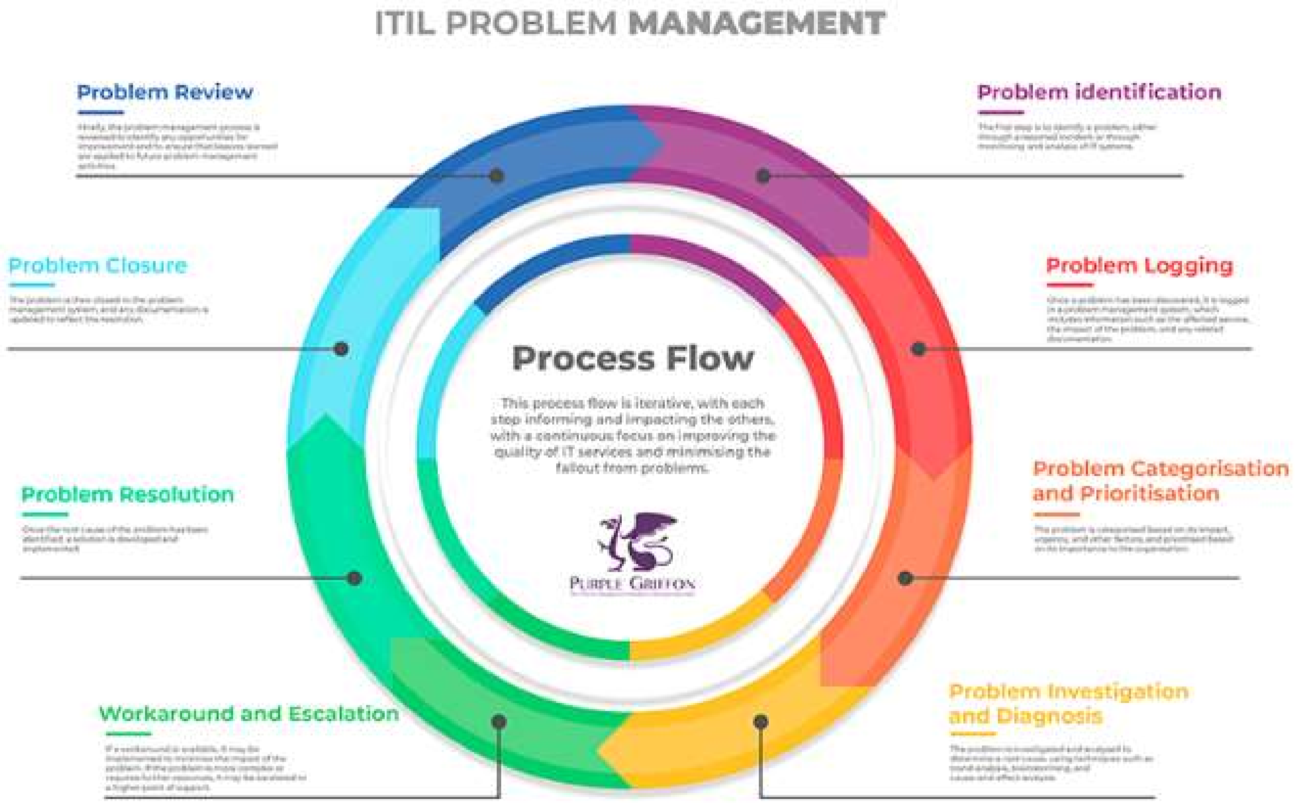
Incident management has a shorter timeline, and the goal is primarily to resolve the incident and return services back to their former state. Problem management is a bit more complex; it can take longer and looks to identify what lies beneath an incident, why the incident occurred, and what can prevent the incident from occurring again.

Problem Management vs Change Management

Change management describes a process in which changes are planned, tracked, and released without any service disruption. In the event that a change causes a disruption, the change is analyzed during a Problem Management process. Change Management provides a systematic approach to control the life cycle of all changes, facilitating beneficial changes to be made with minimum disruption to IT services.

ITIL Problem Management Process Flow

The Problem Management process flow is iterative, with each step informing and impacting the others, with a continuous focus on improving the quality of IT services and minimizing the fallout from problems.



ITIL PROBLEM MANAGEMENT

Problem identification

The first step is to identify a problem, either through a reported incident or through monitoring and analysis of IT systems.

Problem Logging

Once a problem has been discovered, it is logged in a problem management system, which includes information such as the affected service, the impact of the problem, and any related documentation.

Problem Categorisation and Prioritisation

The problem is categorised based on its impact, urgency, and other factors and prioritised based on its importance to the organisation.

Problem Investigation and Diagnosis

The problem is investigated and analysed to determine a root cause, using techniques such as trend analysis, brainstorming, and cause and effect analysis.

Workaround and Escalation

If a workaround is available, it may be implemented to minimise the impact of the problem. If the problem is more complex or requires further resources, it may be escalated to a higher point of support.

Problem Resolution

Once the root cause of the problem has been identified, a solution is developed and implemented.

Problem Closure

The problem is then closed in the problem management system, and any documentation is updated to reflect the resolution.

Problem Review

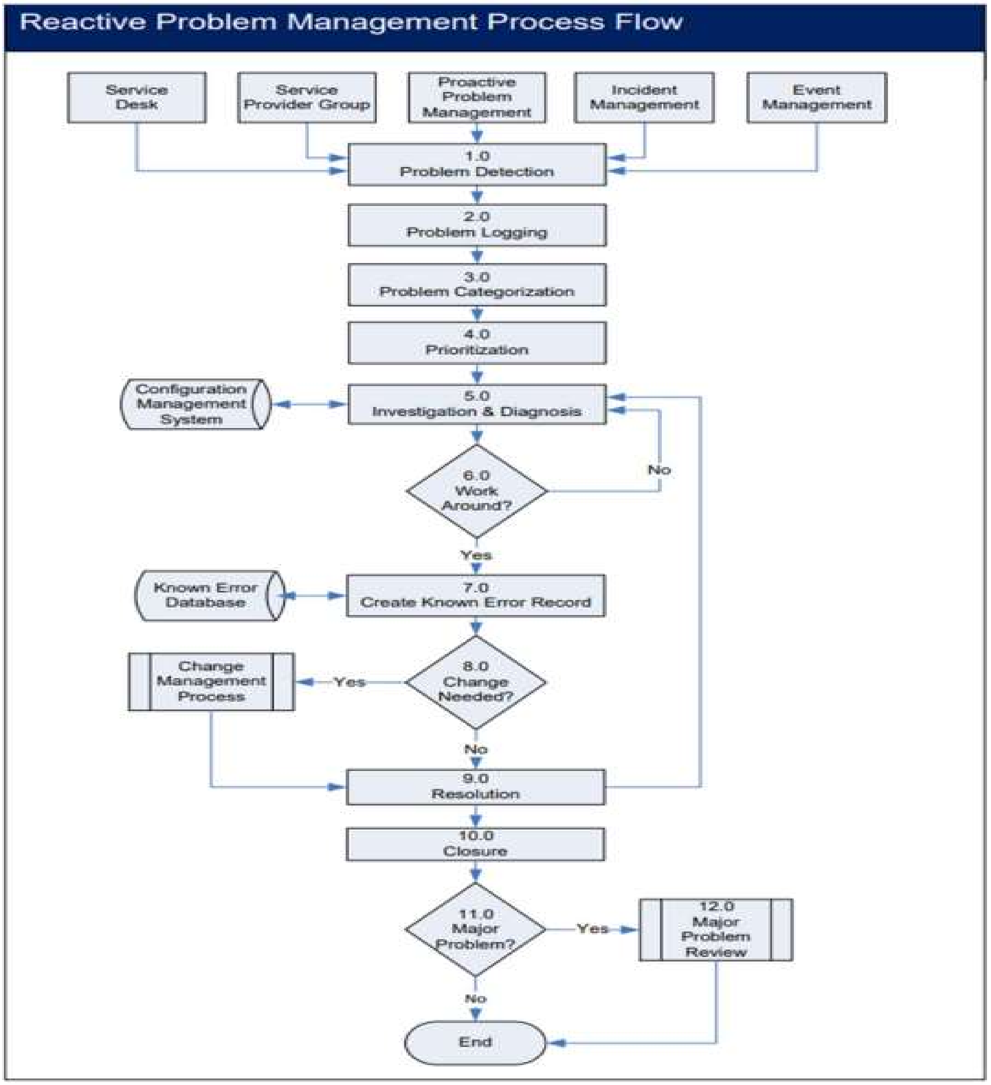
Finally, the problem management process is reviewed to identify any opportunities for improvement and to ensure that lessons learned are applied to future problem management activities.

Process Flow

This process flow is iterative, with each step informing and impacting the others, with a continuous focus on improving the quality of IT services and minimising the fallout from problems.

ITIL Problem Management Process Flow

The steps of this process flow are explained in the next slide.



Reactive Problem Management Process Flow

Service Desk

Service Provider Group

Proactive Problem Management

Incident Management

Event Management

1.0 Problem Detection

2.0 Problem Logging

3.0 Problem Categorization

4.0 Prioritization

Configuration Management System

5.0 Investigation & Diagnosis

6.0 Work Around?

No

Yes

Known Error Database

7.0 Create Known Error Record

Change Management Process

Yes

8.0 Change Needed?

No

9.0 Resolution

10.0 Closure

11.0 Major Problem?

Yes

12.0 Major Problem Review

No

End

Problem Management Process Steps

The process flow of Problem Management in ITIL 4 consists of the following steps:

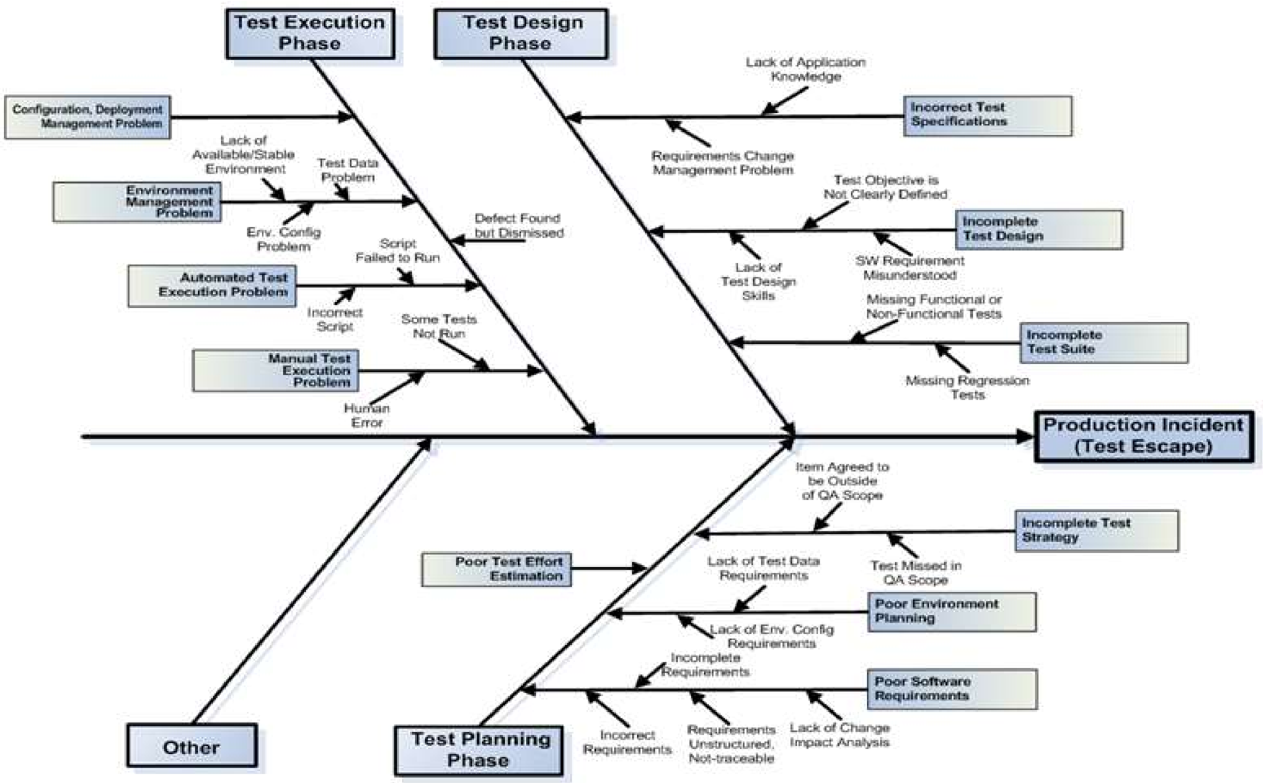
Problem Identification: The first step is to identify a problem, either through a reported incident or through monitoring and analysis of IT systems.
Problem Logging: Once a problem has been discovered, it is logged in a problem management system, which includes information such as the affected service, the impact of the problem, and any related documentation.
Problem Categorization and Prioritization: The problem is categorized based on its impact, urgency, and other factors, and prioritized based on its importance to the organization.
Problem Investigation and Diagnosis: The problem is investigated and analyzed to determine a root cause, using techniques such as trend analysis, brainstorming, and cause-and-effect analysis.
Workaround and Escalation: If a workaround is available, it may be implemented to minimize the impact of the problem. If the problem is more complex or requires further resources, it may be escalated to a higher point of support.
Problem Resolution: Once the root cause of the problem has been identified, a solution is developed and implemented.
Problem Closure: The problem is then closed in the problem management system, and any documentation is updated to reflect the resolution.
Problem Review: Finally, the problem management process is reviewed to identify any opportunities for improvement and to ensure that lessons learned are applied to future problem management activities.

Problem Management Techniques

In ITIL 4, several techniques can be used to manage problems, including:

1. Trend analysis: This involves analyzing patterns and trends in incidents and problems over time to detect potential issues and areas for improvement.
2. Brainstorming: This involves gathering a group of stakeholders and subject matter experts (SME's) to generate ideas and potential resolutions for a problem.
3. Root cause analysis: This involves investigating a problem to identify its primary cause, using techniques such as cause-and-effect analysis, fishbone diagrams (*see next slide*), and the 5 Why's method.
4. Pareto analysis: This involves using the Pareto principle (also known as the 80/20 rule) to identify the most significant problems and prioritize them for resolution.
5. Known error database: This is a database that contains information about known errors and their solutions, allowing problem managers to quickly identify and resolve problems that have occurred before.
6. Workarounds: This involves implementing interim solutions to minimize the impact of a problem on the business while a permanent solution is developed and executed.

Root-Cause Analysis Model Example



Testing-Related Causes of Production Incidents

Test Execution Phase
Test Design Phase
Configuration, Deployment Management Problem
Lack of Available/Stable Environment
Test Data Problem
Environment Management Problem
Env. Config Problem
Defect Found but Dismissed
Script Failed to Run
Automated Test Execution Problem
Incorrect Script
Some Tests Not Run
Manual Test Execution Problem
Human Error
Lack of Application Knowledge
Incorrect Test Specifications
Requirements Change Management Problem
Test Objective is Not Clearly Defined
Incomplete Test Design
Lack of Test Design Skills
SW Requirement Misunderstood
Missing Functional or Non-Functional Tests
Incomplete Test Suite
Missing Regression Tests
Production Incident (Test Escape)
Item Agreed to be Outside of QA Scope
Incomplete Test Strategy
Poor Test Effort Estimation
Lack of Test Data Requirements
Test Missed in QA Scope
Poor Environment Planning
Lack of Env. Config Requirements
Incomplete Requirements
Poor Software Requirements
Incorrect Requirements
Requirements Unstructured, Not-traceable
Lack of Change Impact Analysis
Other
Test Planning Phase

Once production incidents are classified by causes, we can perform Pareto Analysis and produce a chart showing main causes of incidents.

Pareto Analysis Explains

What is Pareto Analysis?

- Pareto Analysis is a technique used for decision making based on the Pareto Principle. Pareto Principle is based on 80/20 rule which says "80% of impacts are due to 20% of causes". It emphasizes that a major number of issues are created by a relatively smaller number of underlying causes.
- Pareto Principle is also called the law of "The **Vital Few** and **Trivial Many**". It is a prioritization tool that helps to find **"VITAL FEW"** and **"TRIVIAL MANY"** causes.
- **Vital Few** means many problems come from a relatively small number of causes that will be addressed in process improvement.
- **Trivial Many** refer to a large number of remaining causes result in very few problems.
- Pareto Analysis can be applied literally in any scenario we see around in our day-to-day life as well.

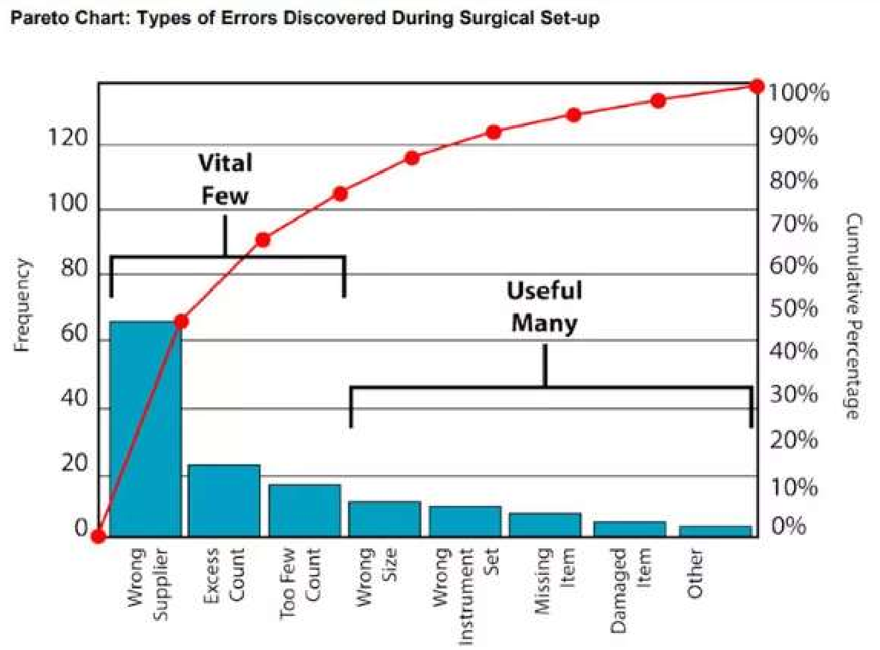
Some Examples:

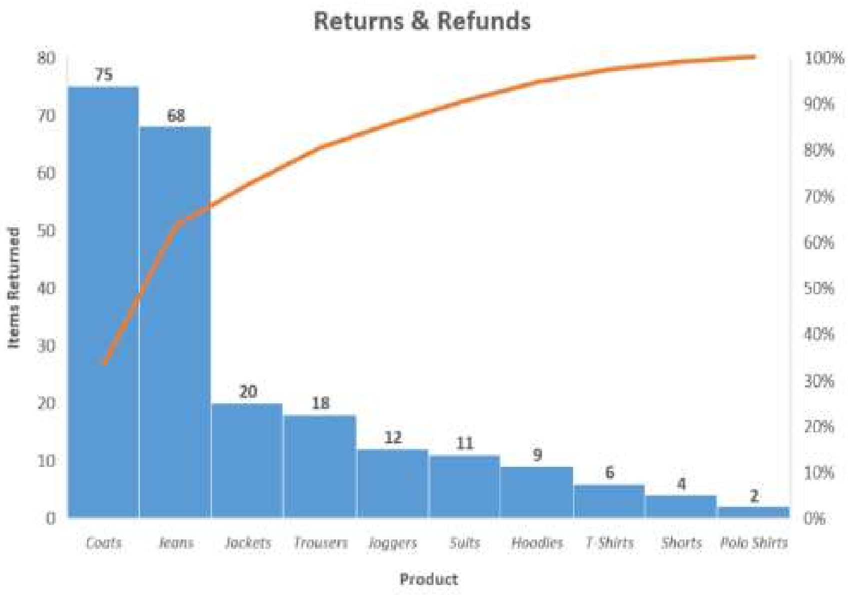
20% of drivers cause 80% of accidents.
20% of clothes in the wardrobe are worn 80% times.
20% of household items consume 80% of electricity.
20% of all people in the world receive 80% of all income.

Examples of Pareto Charts

80% of errors (*vital few*) are caused by only two factors: Wrong Supplier, Excess Count.

80% of returns (*vital few*) are caused by only two products: Coats, Jeans.


Pareto Chart: Types of Errors Discovered During Surgical Set-up
Frequency
Vital Few
Useful Many
Cumulative Percentage
Wrong Supplier
Excess Count
Too Few Count
Wrong Size
Wrong Instrument Set
Missing Item
Damaged Item
Other


Returns & Refunds
Items Returned
Coats
Jeans
Jackets
Trousers
Joggers
Suits
Hoodies
T-Shirts
Shorts
Polo Shirts
Product