Software Engineering
Unit - V

Compiled by
M S Anand

Department of Computer Science

Software Engineering
Introduction

Text Book(s):

1. “Software Engineering: Principles and Practice", Hans van Vliet, Wiley India, 3rd Edition, 2010.
2. “Software Testing – Principles and Practices”, Srinivasan Desikan and Gopalaswamy Ramesh, Pearson, 2006.

Reference Book(s):

1. “Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling” by By Jennifer Davis, Ryn Daniels, O’ Reilly Publications, 2018
2. “Software Engineering: A Practitioner’s Approach”, Roger S Pressman, McGraw Hill, 6th Edition 2005
3. “Software Engineering”, International Computer Science Series, Ian Somerville, Pearson Education, 9th Edition, 2009.
4. “Foundations of Software Testing ", Aditya Mathur, Pearson, 2008
5. “Software Testing, A Craftsman’s Approach ", Paul C. Jorgensen, Auerbach, 2008.
6. IEEE SWEBOK, PMBOK, BABOK and Other Sources from Internet.

Software Engineering
Software Engineering in a Global Environment

<u>Global Environment</u>

<u>Traditional Team</u>

- ➢Social group of individuals <u>collocated</u>
- ➢Tasks are independent
- ➢Activities to achieve common goals

<u>Global Software Development team</u>

- ✓Group of knowledge workers located in <u>various parts of the globe</u> developing commercially viable software
- ✓Use virtual teams
- ✓Same goals and objectives
- ✓Differentiated by distance
- ✓Linked by communication technologies

Global Software Development – Characterised by

Geographic distance

Linguistic distance

Cultural distance

Temporal distance
(The amount of time that separates an individual's present time and a target event in the future)

Types of development

Collocated Development

- Housed within walking distance

Multisite Development

- Members are distributed across sites

Global Development

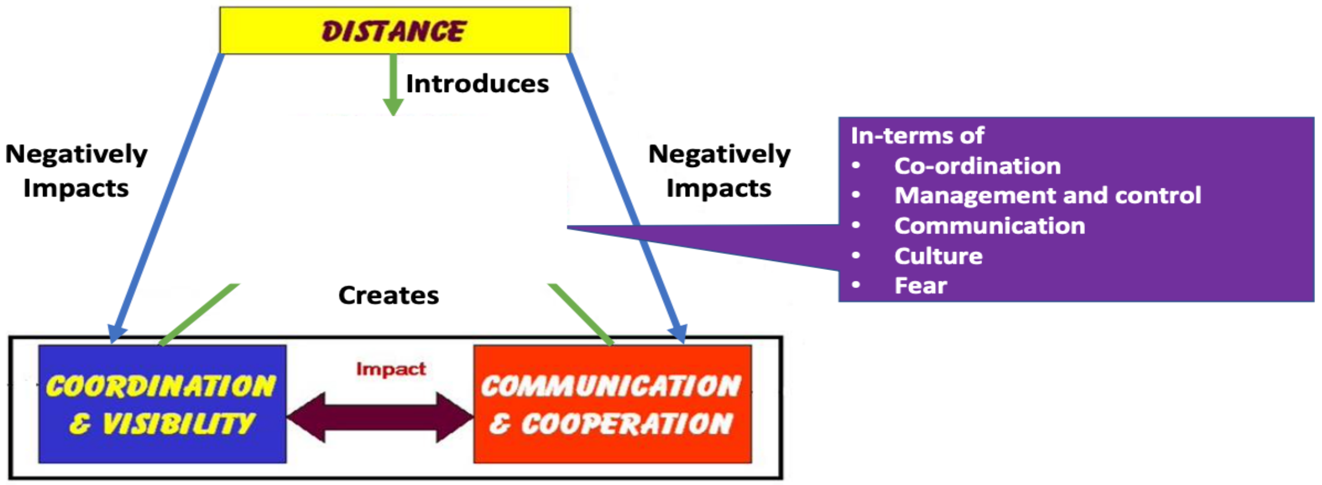
- Distribution of members of multisite are spread across Centers/Sites exceeds frontiers of a country

Software Engineering

Rationale and Challenges for Global Software Development

✓Cost savings by cost arbitrage
✓Taking advantage of time difference to extend productive hours leading to faster delivery
✓Larger and global pool of developers
✓Locating developers closer to markets and customers
✓Taking advantage of diversity of stakeholders knowledge and experiences
✓Leverage of best practices

Challenges



Geographical distance challenges

<u>Communication</u>
Effective information exchange (Less informal exchange, different languages, different domain knowledge)
Build a team (Cohesiveness, removing “them and us”, trust)

<u>Coordination</u>
Task awareness (Shared mental model)
Sense of urgency (Perception)

<u>Control</u>
Accurate Status information (Tracking, stop blaming)
Uniform Process (different tools and techniques)

<u>Cultural</u>
Social background (directed, people orientation to task orientation)
Attitude to authority, national culture, cultural distance
Projection (different tools and techniques)

Overcoming Geographical distance challenges

Evolve process for effective Global Software Engineering

Collaboration Models

- Body leasing or Team leasing
- Team extension
- Project Teams

Teaming models

- Assembly line
- Construction – The whole team should be in place from day One.
- Engineering – Horizontal, Vertical and Matrix type of structure
- Garage (no oversight)

Note:

Tuckman’s model – Forming, Storming, Norming, Performing and Adjourning.

Overcoming Geographical distance challenges

Note:

An **assembly line** is a manufacturing process (often called a *progressive assembly*) in which parts (usually interchangeable parts) are added as the semi-finished assembly moves from workstation to workstation where the parts are added in sequence until the final assembly is produced. By mechanically moving the parts to the assembly work and moving the semi-finished assembly from work station to work station, a finished product can be assembled faster and with less labor than by having workers carry parts to a stationary piece for assembly.

Overcoming Geographical distance challenges

Specific Practices

Project Management Practices

- Common ground
- (De)coupling of work
- Collaboration readiness
- Technology readiness
- Empowerment
- Cultural awareness
- Communication

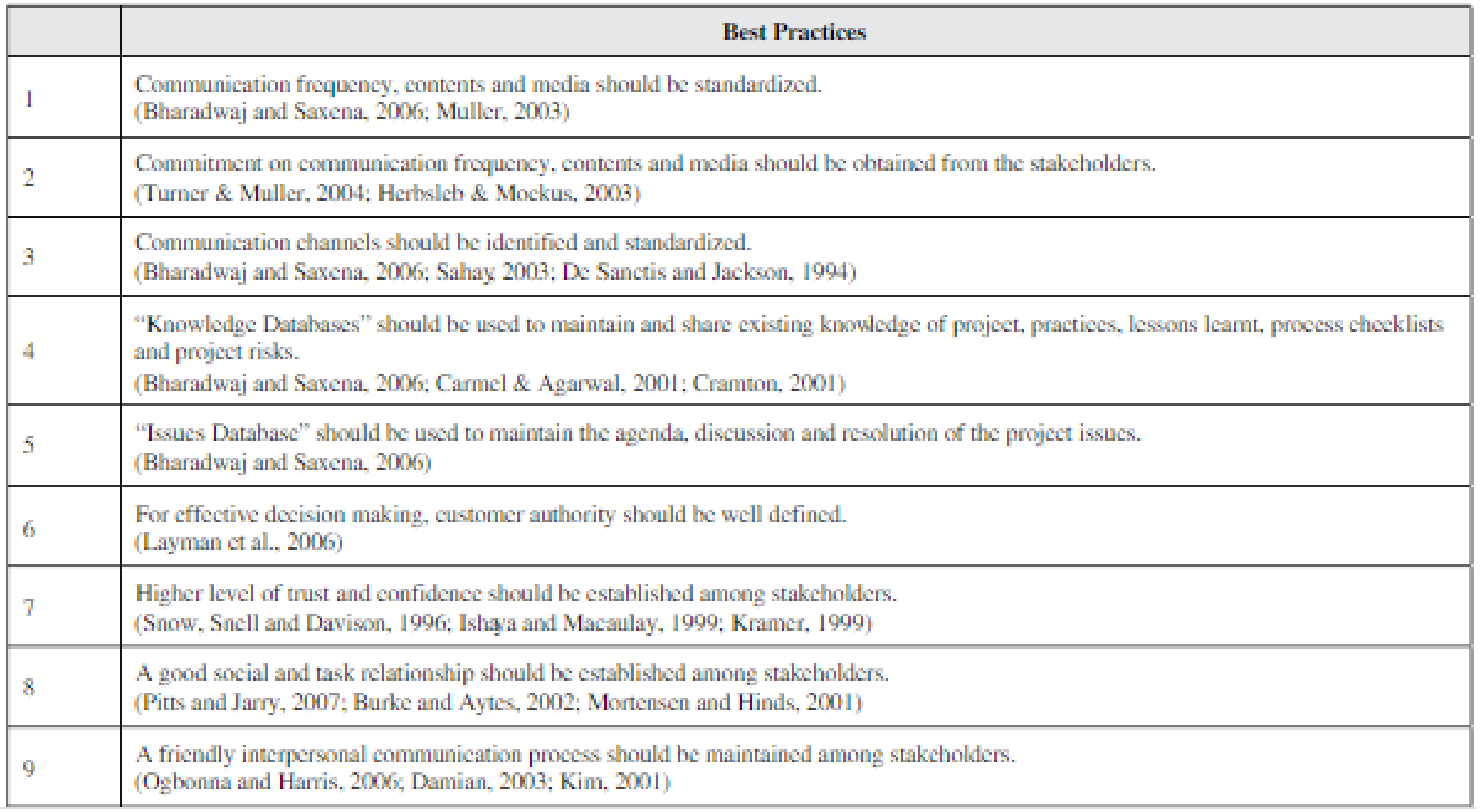
Communication Practices

Software Engineering

Overcoming Geographical distance challenges

Communication Practices

Table 1. Best practices of the "effective communication"

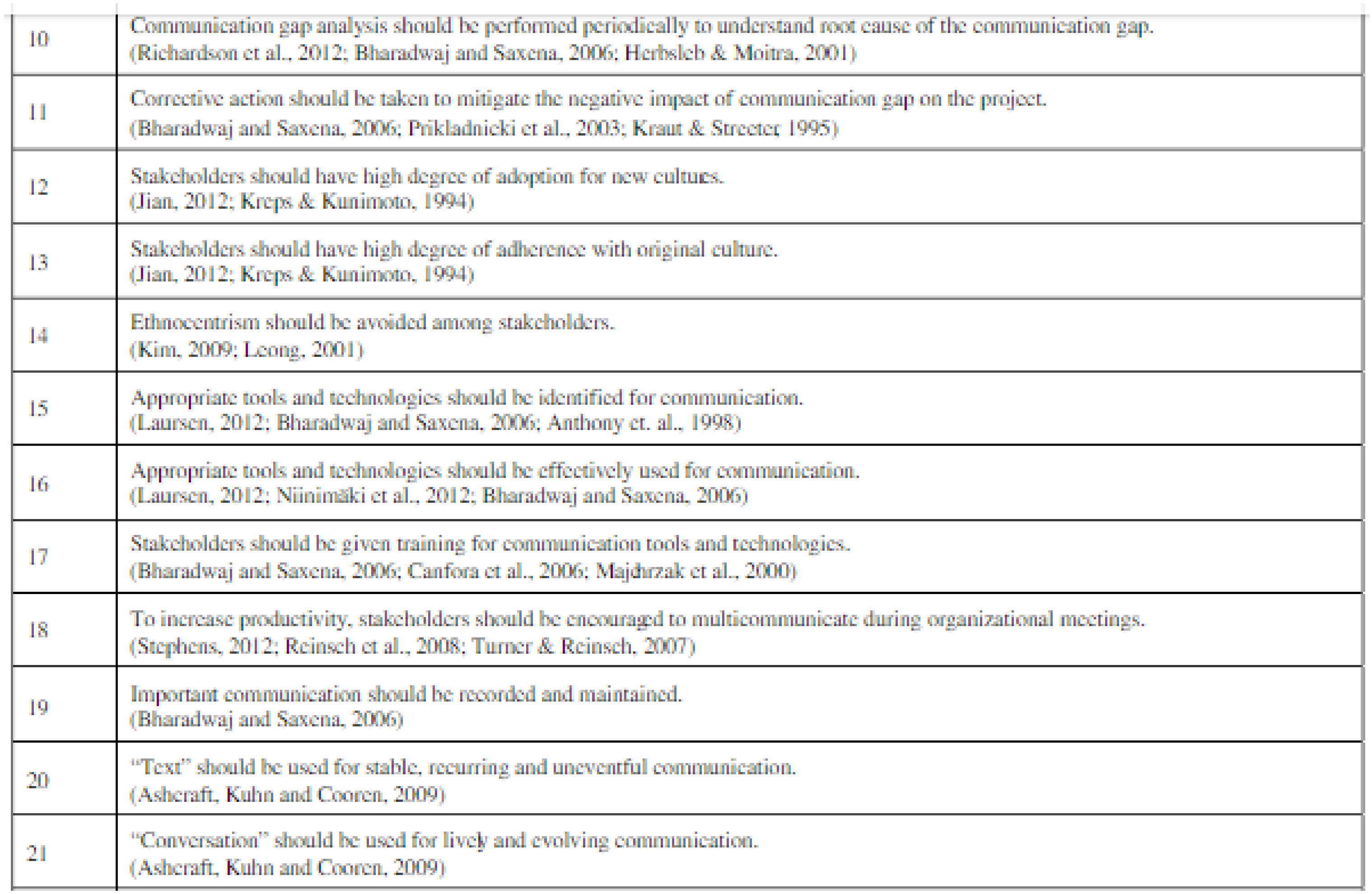


	Best Practices
1	Communication frequency, contents and media should be standardized. (Bharadwaj and Saxena, 2006; Muller, 2003)
2	Commitment on communication frequency, contents and media should be obtained from the stakeholders. (Turner & Muller, 2004; Herbsleb & Mockus, 2003)
3	Communication channels should be identified and standardized. (Bharadwaj and Saxena, 2006; Sahay 2003; De Sanctis and Jackson, 1994)
4	"Knowledge Databases" should be used to maintain and share existing knowledge of project, practices, lessons learnt, process checklists and project risks. (Bharadwaj and Saxena, 2006; Carmel & Agarwal, 2001; Cramton, 2001)
5	"Issues Database" should be used to maintain the agenda, discussion and resolution of the project issues. (Bharadwaj and Saxena, 2006)
6	For effective decision making, customer authority should be well defined. (Layman et al., 2006)
7	Higher level of trust and confidence should be established among stakeholders. (Snow, Snell and Davison, 1996; Ishaya and Macaulay, 1999; Kramer, 1999)
8	A good social and task relationship should be established among stakeholders. (Pitts and Jarry, 2007; Burke and Aytes, 2002; Mortensen and Hinds, 2001)
9	A friendly interpersonal communication process should be maintained among stakeholders. (Ogbonna and Harris, 2006; Damian, 2003; Kim, 2001)

Software Engineering

Overcoming Geographical distance challenges

Communication Practices



10	Communication gap analysis should be performed periodically to understand root cause of the communication gap. (Richardson et al., 2012; Bharadwaj and Saxena, 2006; Herbsleb & Moitra, 2001)
11	Corrective action should be taken to mitigate the negative impact of communication gap on the project. (Bharadwaj and Saxena, 2006; Prikladnicki et al., 2003; Kraut & Streeter 1995)
12	Stakeholders should have high degree of adoption for new cultures. (Jian, 2012; Kreps & Kunimoto, 1994)
13	Stakeholders should have high degree of adherence with original culture. (Jian, 2012; Kreps & Kunimoto, 1994)
14	Ethnocentrism should be avoided among stakeholders. (Kim, 2009; Leong, 2001)
15	Appropriate tools and technologies should be identified for communication. (Laursen, 2012; Bharadwaj and Saxena, 2006; Anthony et. al., 1998)
16	Appropriate tools and technologies should be effectively used for communication. (Laursen, 2012; Niinimäki et al., 2012; Bharadwaj and Saxena, 2006)
17	Stakeholders should be given training for communication tools and technologies. (Bharadwaj and Saxena, 2006; Canfora et al., 2006; Majchrzak et al., 2000)
18	To increase productivity, stakeholders should be encouraged to multicommunicate during organizational meetings. (Stephens, 2012; Reinsch et al., 2008; Turner & Reinsch, 2007)
19	Important communication should be recorded and maintained. (Bharadwaj and Saxena, 2006)
20	"Text" should be used for stable, recurring and uneventful communication. (Ashcraft, Kuhn and Cooren, 2009)
21	"Conversation" should be used for lively and evolving communication. (Ashcraft, Kuhn and Cooren, 2009)

Hacking

- Historically, used constructively in terms of cleverness or ability to move ahead in spite of obstacles

- Entails some form of excellence

- Currently, gets used more often with criminality in terms of gaining unauthorized access to systems and resources

- Criminal Hacking: Using psychology to trick user into unknowingly giving information or data to the hacker
  - Social Engineering

Hacker v/s Software Engineer

Hacker

- Solves problem with non standard approaches or exploits weakness
- Needs novelty, attracted by challenges and toys
- Weakly motivated by conventional rewards

Software Engineer

- Takes a problem, fits it within criteria, to meet performance benchmark, exceed reliability requirements, make it aesthetically pleasing, meet arbitrary regulations
- Creative and original thinking given constraints

Types of hackers

Hackers use deep understanding and knowledge of computer systems and software. They look to exploit weaknesses of the system

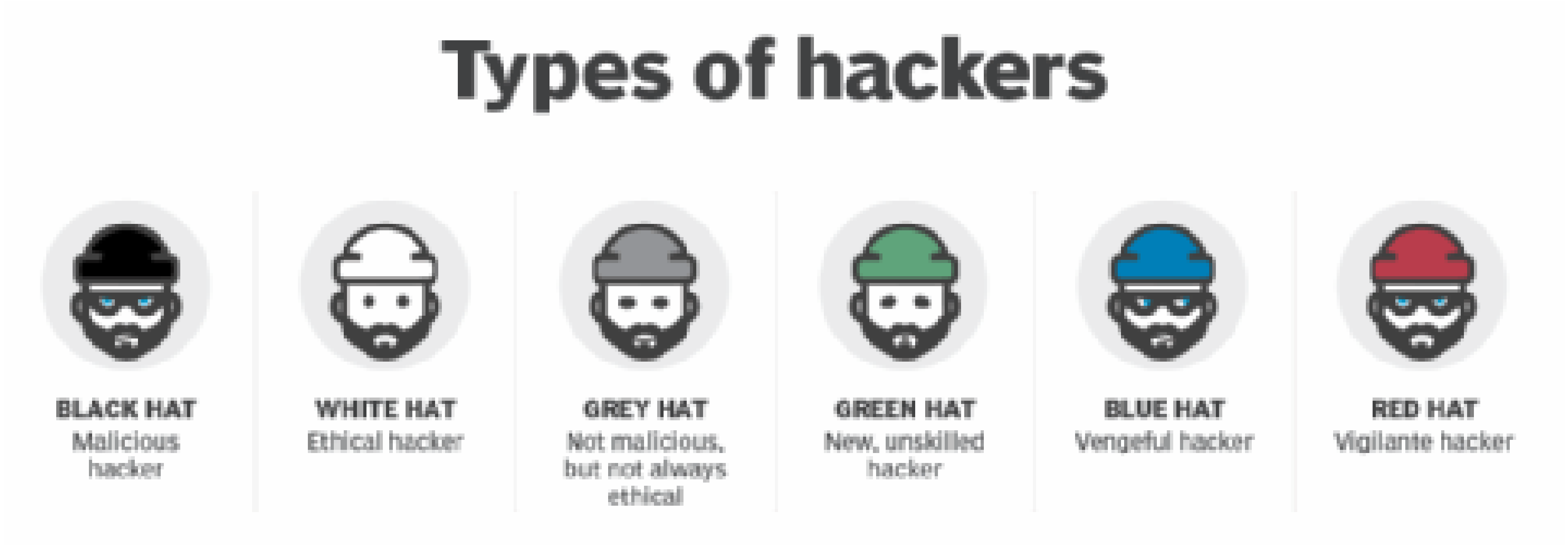
Types of hackers
White Hat Hackers:
strive to improve security of organization’s security systems by finding vulnerable flaws

- Prevent Cybercrime

Black Hat Hackers:
subvert the technology, for stealing something valuable or for other malicious reasons

- Disrupting reputation, corporate espionage, nation-state hacking

Types of hackers



Hacking v/s Software Engineering

Hacker tries to stretch what's possible or what exists
Once something is functional and successful, hacker moves on to another problem
Engineer picks a problem makes it fit within a limiting criteria

Hacking involves poking around approaches towards a solution
If a solution is achieved, harder to backtrack
Engineering involves crafting a solution understanding why and considering the best practices

Necessary to hack and engineer at different times

Software Engineering
Ethics in Software

Ethics, also called moral philosophy is the discipline concerned with what is morally good and bad and morally right and wrong.

As computer scientists and software engineers you will be faced with making difficult decisions.

- Personal ethics (Integrity, Objectivity, Professional Competence, Confidentiality)

- Professional ethics
    - ➢Reuse of source code
    - ➢Reuse of copyrighted materials
    - ➢Enforcing strength of passwords
    - ➢Knowledge of flaw or bug
    - ➢Design of potentially dangerous software feature
    - ➢Discovery of security flaw

Often, there are clear cut laws to dictate the decision

Ethics

Personal Ethics

- Values or viewpoints an individual uses to influence and guide his or her personal, social, or professional behavior
- Values are reinforced and strengthened from childhood

Business Ethics

- Values or viewpoints which a business and “individuals when doing business” use to influence and guide behavior
- Focus on behaviors pertaining to work

Professional Ethics

- Principles that govern the behavior and judgment while performing his/her profession.

Ethical frameworks

Considering Ethics as a framework

•Reason we make a choice matters

•Most people have an intrinsic sense of what is “right”

Utilitarian Framework

•Cost analysis based on risks, costs and probabilities of outcomes

•Includes consideration of legal violations (through liability, cost of fines, risk of jail)

Framework of Individual Rights

•Can’t put a price of life

•People are entitled to certain rights: right to dignity, right to freedom, property rights

•One’s actions shouldn’t trample others

Ethical frameworks and principles

Golden Rule

- Treat others as you want to be treated
    - Example: documenting known bugs and important assumptions

- **Principle of Beneficence**: decision making to do what is right and good; prioritize doing good

- **Principle of least Harm**: deals with situations when no choice is beneficial

- **Principle of Respect for Autonomy**: allowing people to make decisions that apply to their situations without interference

- **Justice Ethical**: focus on actions that are fair to those involved

Software Engineering
Ethics

- **Codes of Ethics**
  - Guiding principles or boundaries in understanding the difference between Right/Wrong that can be used by individuals acting as free moral agents to make choices to guide their behaviour
    - Integrity.
    - Objectivity.
    - Professional competence.
    - Confidentiality
    - Professional Behaviour (Cost/Work/ Use and Abuse of resources)
    - No Plagiarism and respect for Intellectual Property

- **Codes of Conduct / Practice**
  - How an employee will need to behave in the workplace on a day-to-day basis
  - a specific set of professional behaviours and values the professional must know and must abide by
    - Equality and fairness
    - Empathy and respect
    - Performing to one's best capability
    - Obedience to the law
    - Privacy
    - Following standard practices and standards
    - Portraying realistic competence levels & should contribute in the areas of competence
    - Non-Plagiarism

Code of conduct - examples

Intel's code of conduct is here.

Philip's code of conduct

Placing the customer first, and upholding patient safety, quality and integrity always, goes to the heart of Philips' purpose. This is why we: design and deliver safe, effective and reliable products, solutions and services. adhere to and maintain the effectiveness of the quality management system.

Twitter's code of conduct

Treat others with dignity and respect. Refrain from demeaning or discriminatory behavior and speech. Be mindful of your surroundings and of your fellow participants.

Software Engineering
Ethics

Public: software engineers act consistently with public interest
Client and Employer: Act in a manner that is in best interests of their client or employer and consistent with public interest
Product: Products and Related modifications meet highest professional standards
Judgement: maintain integrity and independence in professional judgment
Management: subscribe to and promote an ethical approach to management
Profession: advance the integrity and reputation of the profession consistent with public interest
Colleagues: fair and supportive of their colleagues
Self: lifelong learning and promote ethical approach

Ethics

Intellectual Property: property that includes tangible creations of human intellect

Types: patents, copyrights, trademarks, trade secrets

- Patents: focuses on methods to do something (utility patents), look-and-feel (design patents)
- Copyrights: focus on expressions of implementations to protect original work if authorship

Important to engineers and companies

Provides protection of investment
Patent/copyright infringement can be costly

IP is an asset

- Patents have value and can be traded between companies

Licencing

•Licenses provide rules and guidelines for others to use your work

•Licensing is not giving its rights away, as copyrights are your own to have

•Open source licenses help others to contribute to work or project without seeking permission

•GNU – General Public License is a free software license that allows software to be modified or redistributed without any restrictions or compulsory payments for the licensed code. The GPL license is used by developers to ensure that their code does not become proprietary when modified.

•BSD license - a low restriction type of license for open source software that does not put requirements on redistribution.

•MIT license - is a permissive license, meaning it carries very few restrictions in these areas.

IT Service Management (ITSM)

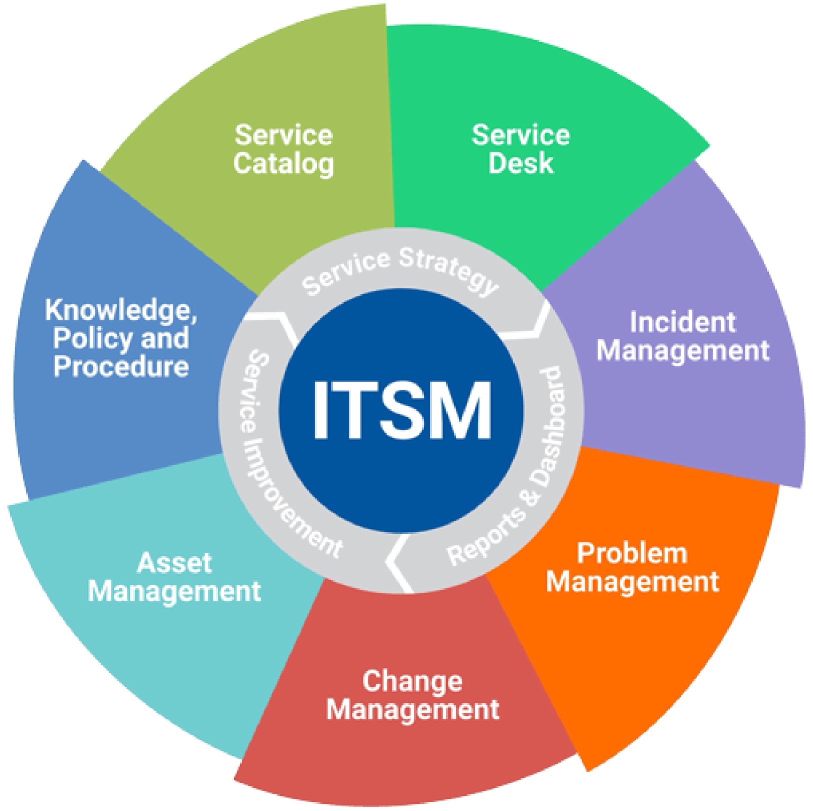
What is IT Service Management (ITSM)?

IT service management -- often referred to as ITSM -- is simply how IT teams manage the end-to-end delivery of IT services to customers. This includes all the processes and activities to design, create, deliver, and support IT services.

The core concept of ITSM is the belief that IT should be delivered as a service. A typical ITSM scenario could involve asking for new hardware like a laptop. You would submit your request through a portal, filling out a ticket with all relevant information, and kicking off a repeatable workflow. Then, the ticket would land in the IT team's queue, where incoming requests are sorted and addressed according to importance.

Software Engineering
IT Service Management (ITSM)

Due to their day-to-day interactions with IT, people often misconstrue ITSM as basic IT support. On the contrary, ITSM teams oversee all kinds of workplace technology, ranging from laptops, to servers, to business-critical software applications.



Applications in the IT infrastructure use ITSM processes

Applications developed using SDLC and/or being deployed in an organization typically will form part of or will influence the business processes of the organization

Applications are deployed by IT operations/Infrastructure management teams using ITSM processes

- Production acceptance process for deployment

IT Infrastructure: consists of equipment, systems, software and services

- Used, in common, across an organization
- Required to develop, test, deliver, monitor, control or support IT services

IT Systems Management and IT Operations

IT Systems Management: how processes and service are administered to ensure that IT infrastructure <u>provides stable and responsive IT environment</u>

- Management of processes associated with IT infrastructure to deliver the right set of services at right quality at competitive costs

IT Operations: support activities that the IT department needs to support within a large organization

- These applications impact the business, offer IT services and are deployed by IT operations or infrastructure management teams
- Product acceptance process
  - Consistent and successful deployment of application systems
  - Support increasing deployment frequency
- Product acceptance process would ensure capacity is planned, a robust streamlined deployment process

Software Engineering

IT Operations Activities can include

- Routine and day-to-day operational tasks and maintenance, performing data back-ups, restoring system after service outage/update and would also look at organizations disaster recovery plans

- Configuring and tuning servers and other configurable infrastructure components to optimize their performance

- Allocating IT resources where they are needed to promote effective service delivery

- Monitoring and measuring performance of IT infrastructure
    - Including IT organization's security posture

- Developing operational metrics to evaluate performance of key processes and services, manage software license compliance and conduct infrastructure audits to verify that security and performance targets are met

IT Infrastructure systems management

IT Infrastructure consists of number of physical devices

- Servers, Networks, Disk Storage, Desktop Computer
- Software products such as Databases

IT Infrastructure Systems Management is about how an IT organization manages IT services and provides a stable and responsive IT environment

- Supports and furthers business of the organization

IT Infrastructure systems management - objectives

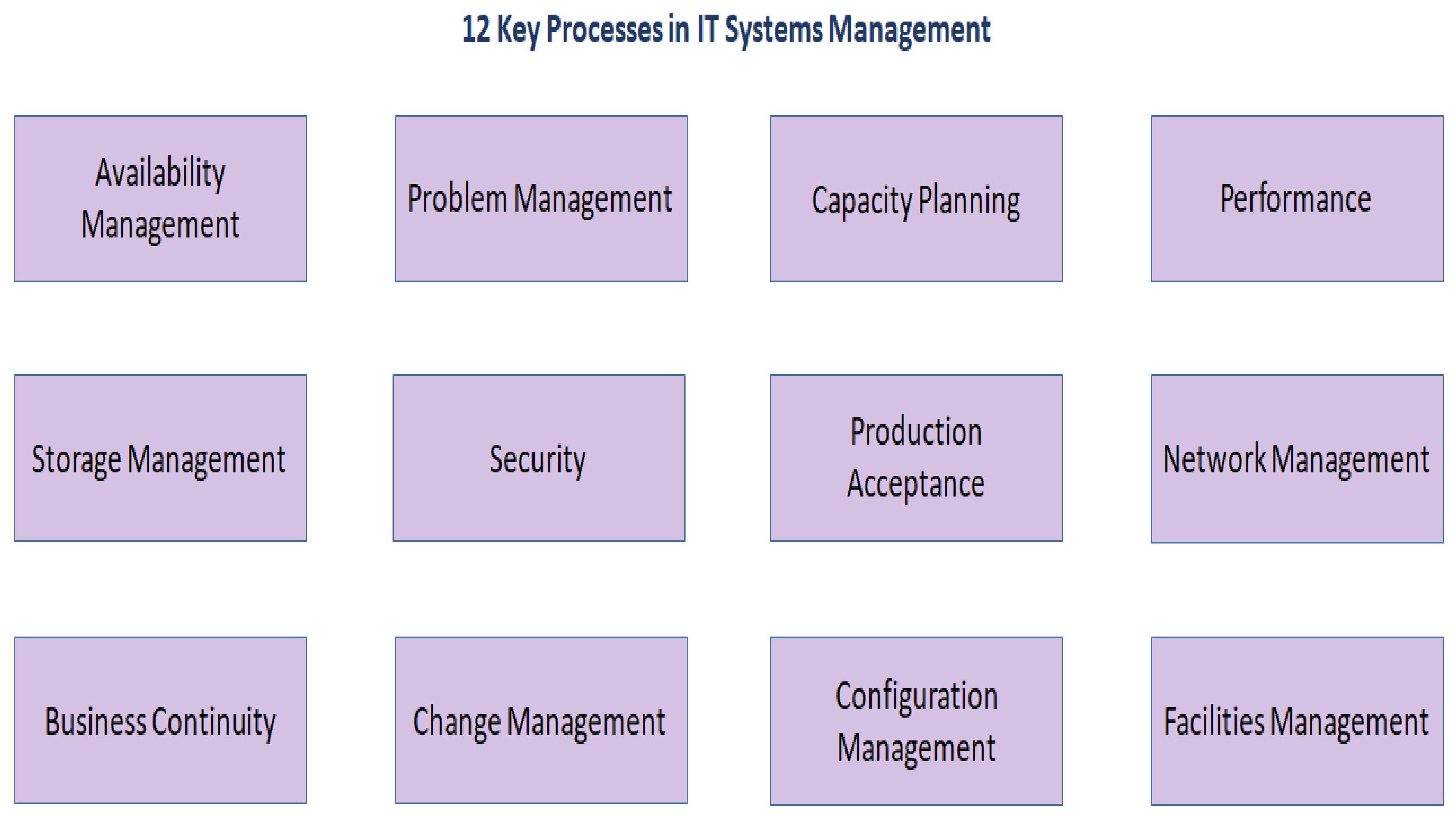
✓Provide Stable(Available) and responsive IT infrastructure

- **Stability**: Systems are always up and accessible as scheduled, 24x7
  - ➢Measure: % of uptime, % of Downtime, MTBF, MTTR
- **Responsiveness**: How quickly the jobs are processed and completed
  - ➢Measure: Throughput, Average Turn Around Time

✓Predictable Support and Service costs
✓Service costs that scales with Business
✓Reduction of cost of IT Management
✓Increased flexibility and responsiveness to business needs
✓Improved productivity and customer satisfaction
✓Improved security, reliability and availability of IT infrastructure
✓Ability to integrate existing technologies and add new technologies

IT Infrastructure systems management - objectives

Developing organization capabilities to efficiently and effectively manage enterprise IT



12 Key Processes in IT Systems Management

Availability Management	Problem Management	Capacity Planning	Performance
Storage Management	Security	Production Acceptance	Network Management
Business Continuity	Change Management	Configuration Management	Facilities Management

Software Engineering

ITSM processes

What is ITSM?

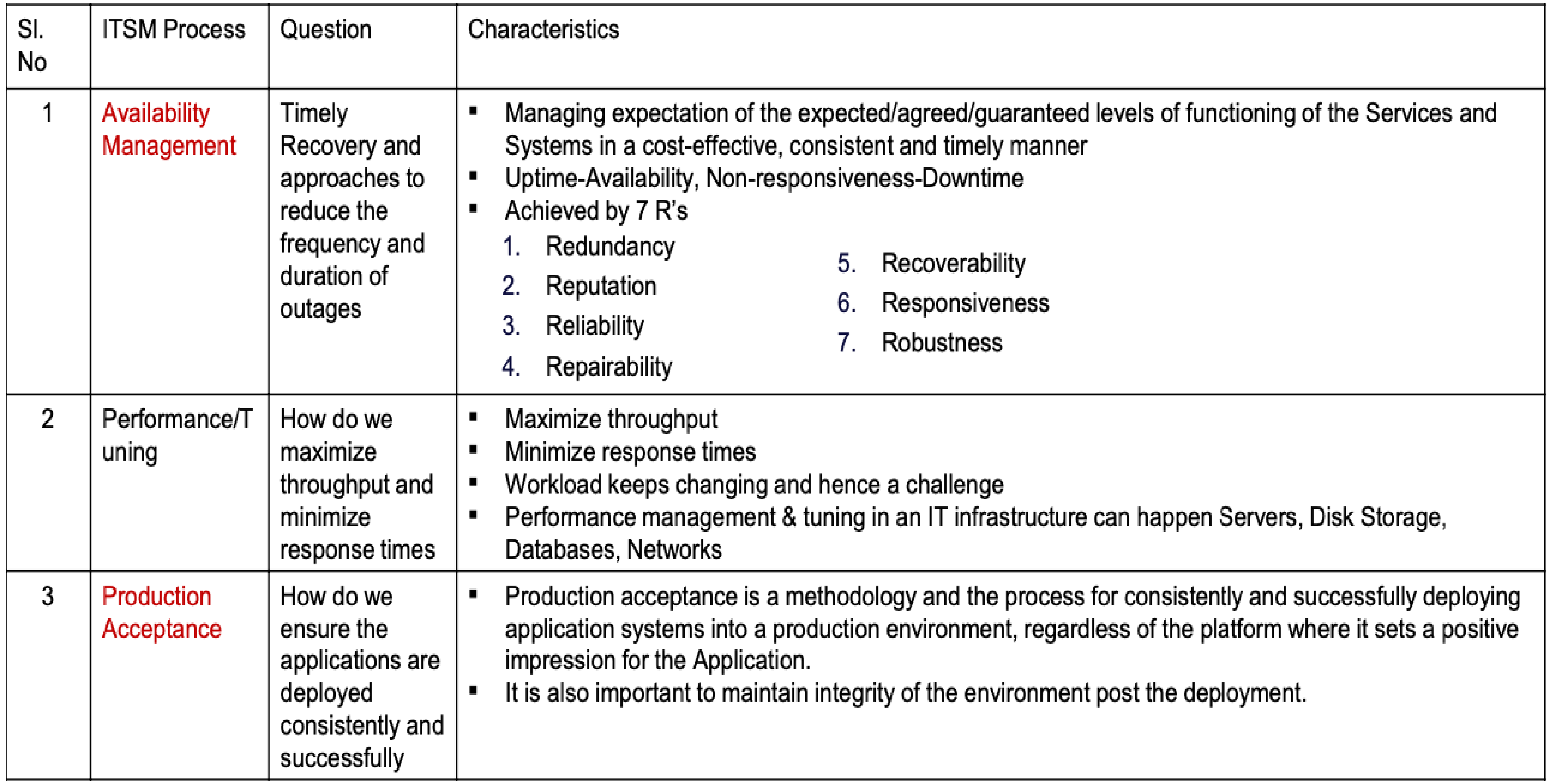
•IT Service Management, or ITSM, refers to the entirety of activities directed by policies, organized and structured in processes and supporting procedures, that are performed by an organization to design, plan, deliver, operate and control information technology services offered to customers.

•“IT Services” refers to the application of business and technical expertise, and offering services which enable organizations in creation, management, optimization, or to provide access to information and business processes

•Organization's IT infrastructure will support firm’s business and information systems strategy

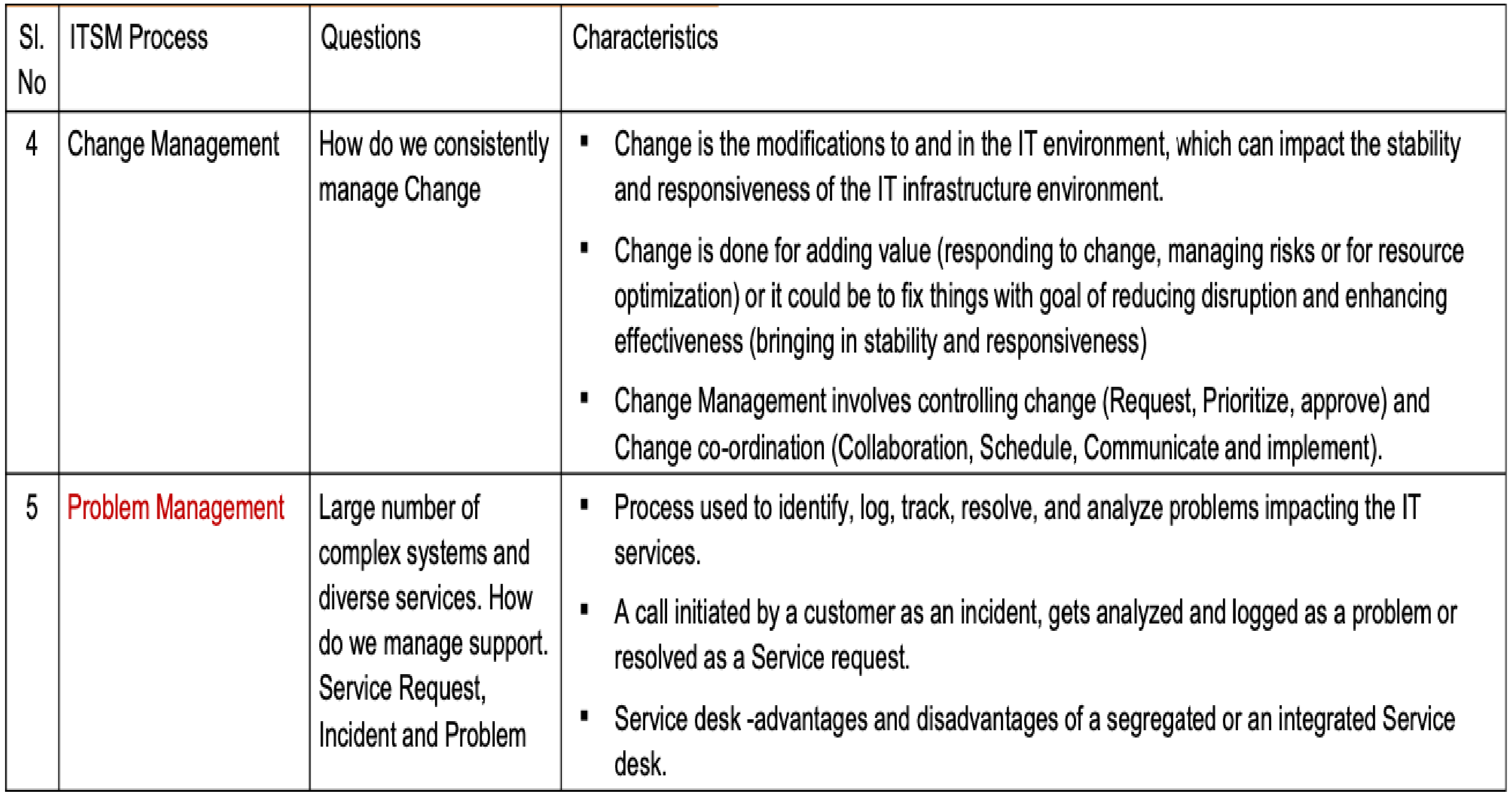
Software Engineering

ITSM Processes



Sl. No	ITSM Process	Question	Characteristics
1	Availability Management	Timely Recovery and approaches to reduce the frequency and duration of outages	▪ Managing expectation of the expected/agreed/guaranteed levels of functioning of the Services and Systems in a cost-effective, consistent and timely manner ▪ Uptime-Availability, Non-responsiveness-Downtime ▪ Achieved by 7 R's 1. Redundancy 2. Reputation 3. Reliability 4. Repairability 5. Recoverability 6. Responsiveness 7. Robustness
2	Performance/Tuning	How do we maximize throughput and minimize response times	▪ Maximize throughput ▪ Minimize response times ▪ Workload keeps changing and hence a challenge ▪ Performance management & tuning in an IT infrastructure can happen Servers, Disk Storage, Databases, Networks
3	Production Acceptance	How do we ensure the applications are deployed consistently and successfully	▪ Production acceptance is a methodology and the process for consistently and successfully deploying application systems into a production environment, regardless of the platform where it sets a positive impression for the Application. ▪ It is also important to maintain integrity of the environment post the deployment.

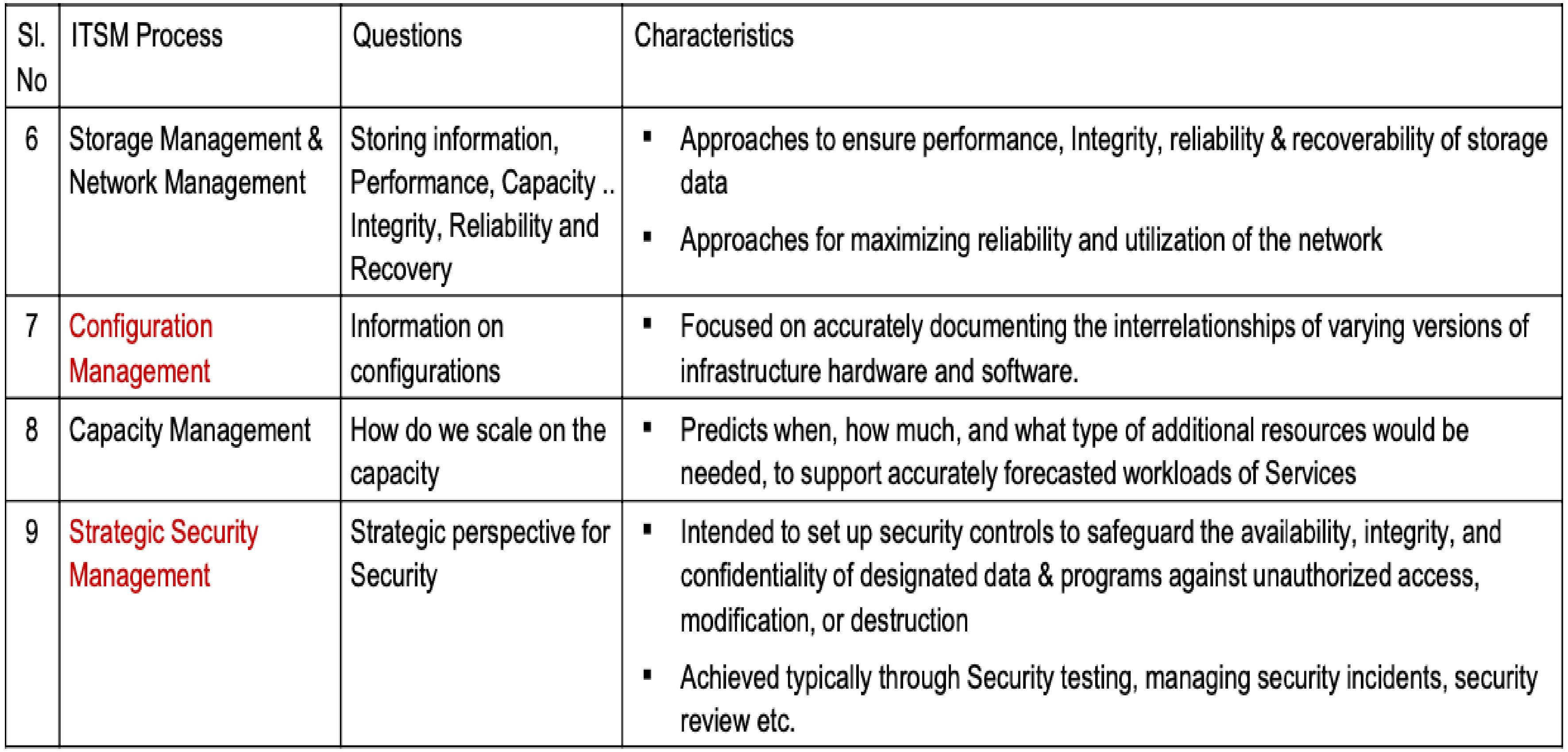
ITSM Processes …



Sl. No	ITSM Process	Questions	Characteristics
4	Change Management	How do we consistently manage Change	▪ Change is the modifications to and in the IT environment, which can impact the stability and responsiveness of the IT infrastructure environment. ▪ Change is done for adding value (responding to change, managing risks or for resource optimization) or it could be to fix things with goal of reducing disruption and enhancing effectiveness (bringing in stability and responsiveness) ▪ Change Management involves controlling change (Request, Prioritize, approve) and Change co-ordination (Collaboration, Schedule, Communicate and implement).
5	Problem Management	Large number of complex systems and diverse services. How do we manage support. Service Request, Incident and Problem	▪ Process used to identify, log, track, resolve, and analyze problems impacting the IT services. ▪ A call initiated by a customer as an incident, gets analyzed and logged as a problem or resolved as a Service request. ▪ Service desk -advantages and disadvantages of a segregated or an integrated Service desk.

Software Engineering

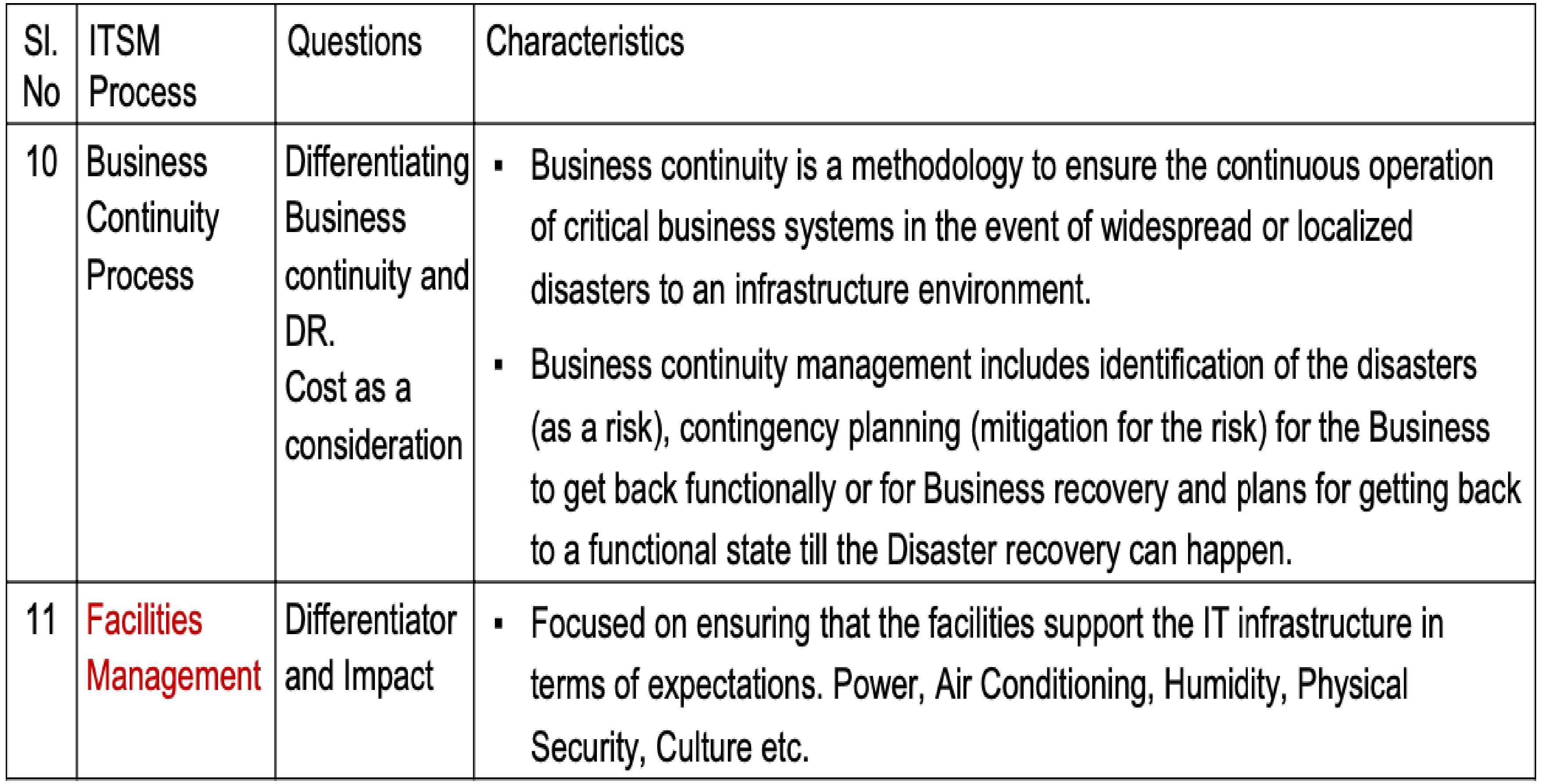
ITSM Processes …



Sl. No	ITSM Process	Questions	Characteristics
6	Storage Management & Network Management	Storing information, Performance, Capacity .. Integrity, Reliability and Recovery	▪ Approaches to ensure performance, Integrity, reliability & recoverability of storage data ▪ Approaches for maximizing reliability and utilization of the network
7	Configuration Management	Information on configurations	▪ Focused on accurately documenting the interrelationships of varying versions of infrastructure hardware and software.
8	Capacity Management	How do we scale on the capacity	▪ Predicts when, how much, and what type of additional resources would be needed, to support accurately forecasted workloads of Services
9	Strategic Security Management	Strategic perspective for Security	▪ Intended to set up security controls to safeguard the availability, integrity, and confidentiality of designated data & programs against unauthorized access, modification, or destruction ▪ Achieved typically through Security testing, managing security incidents, security review etc.

Software Engineering

ITSM Processes …



Sl. No	ITSM Process	Questions	Characteristics
10	Business Continuity Process	Differentiating Business continuity and DR. Cost as a consideration	▪ Business continuity is a methodology to ensure the continuous operation of critical business systems in the event of widespread or localized disasters to an infrastructure environment. ▪ Business continuity management includes identification of the disasters (as a risk), contingency planning (mitigation for the risk) for the Business to get back functionally or for Business recovery and plans for getting back to a functional state till the Disaster recovery can happen.
11	Facilities Management	Differentiator and Impact	▪ Focused on ensuring that the facilities support the IT infrastructure in terms of expectations. Power, Air Conditioning, Humidity, Physical Security, Culture etc.

ITSM frameworks

Number of ITSM frameworks which businesses can use.

Some frameworks are targeted at specific industries

Popular Frameworks

- **IT Infrastructure Library (ITIL)** (V4): framework of best practices for delivering IT services
- **Business Process Framework**: designed for telecommunications service providers
- **COBIT** (Control Objectives for Information and Related Technologies): IT governance framework
- **FitSM**: simplified, streamlined service management framework aligned with ISO/IEC 20000
- **ISO/IEC 20000**: considered the international standard for IT service management and delivery

ITIL – IT Infrastructure Library

What is ITIL?

•ITIL is a framework or set of IT Service management best practice processes that align IT services with business needs

•Approaches that have worked on some scenarios, for selection, planning, delivery, maintenance and overall lifecycle of IT services

•It helps businesses manage risk, strengthen customer relations, establish cost effective practices and build stable IT environment that allows for growth, scale and change

•ITIL describes best practices which are not organization/technology specific

- •Can be applied by organization for establishing integration with organization strategy, delivering value, and maintaining minimum level of competency

•Organization to establish baseline from which it can plan, implement and measure

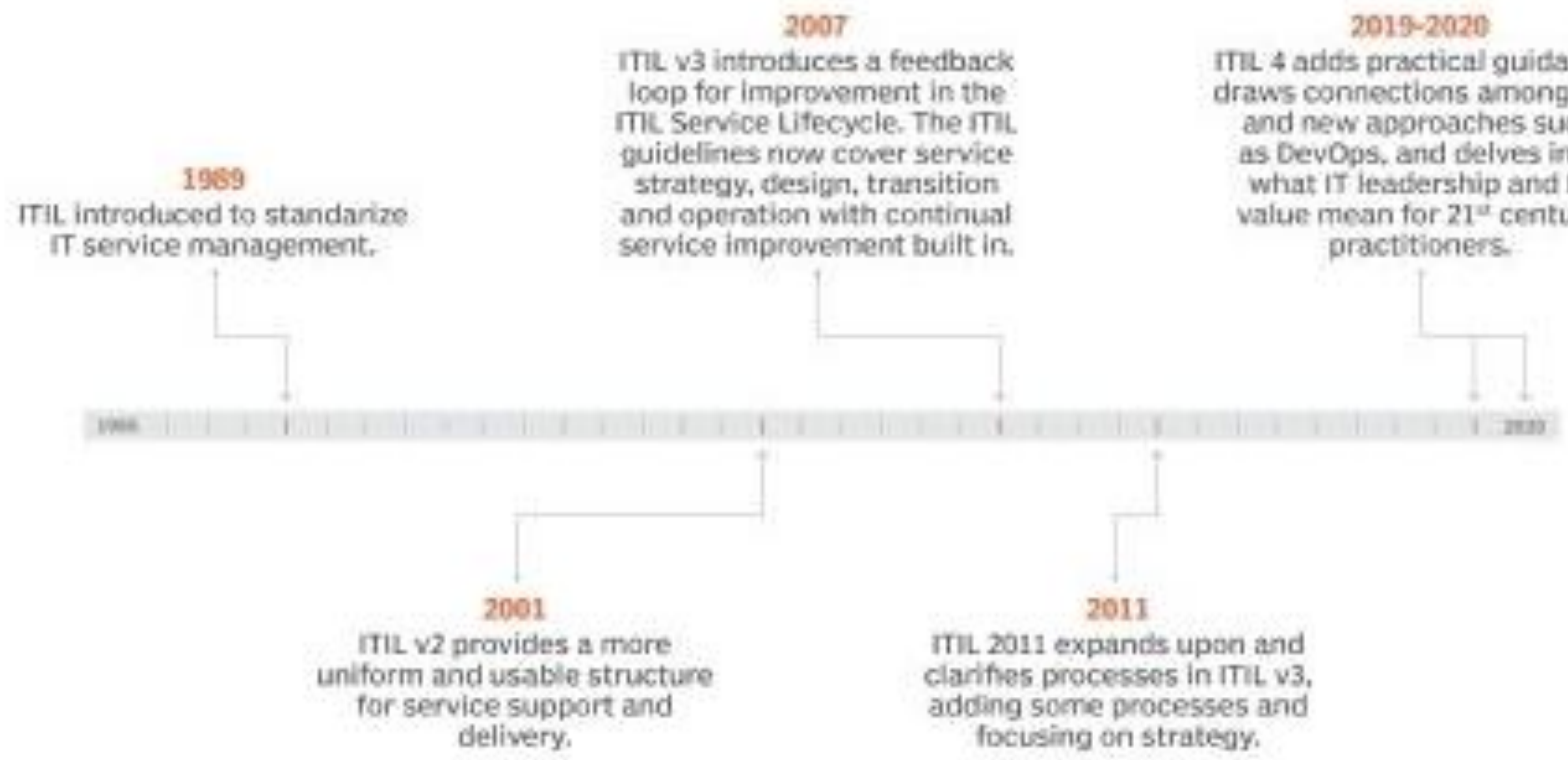
- •Demonstrate compliance and measure improvement

•Refined over the years and provides guidance for service management becoming de facto standard framework of best practices

Origins of ITIL

- •ITIL started as process improvement initiative to improve quality of products and services
- •In 1986, British Government's Centralized Telecommunications and Computing Agency formally sponsored program to promote improved management of IT services
  - •Team of 40 IT experts established a framework of best practices
- •Due to growth of books regarding ITIL, in 2000, a more condensed ITIL version 2 came in
  - •Released 7 books where Service Support and Service Delivery were prominent
- •In 2007, ITIL V3 consisting of 26 processes and functions, grouped into 5 volumes around the concept of service lifecycle
- •In 2019, ITIL V4 came in which focuses on business and technology while working with agile, DevOps and digital transformation

Software Engineering
ITIL evolution



1989
ITIL introduced to standarize IT service management.

2001
ITIL v2 provides a more uniform and usable structure for service support and delivery.

2007
ITIL v3 introduces a feedback loop for improvement in the ITIL Service Lifecycle. The ITIL guidelines now cover service strategy, design, transition and operation with continual service improvement built in.

2011
ITIL 2011 expands upon and clarifies processes in ITIL v3, adding some processes and focusing on strategy.

2019-2020
ITIL 4 adds practical guidance, draws connections among ITIL and new approaches such as DevOps, and delves into what IT leadership and IT value mean for 21st century practitioners.

What is the basic objective of ITIL?

ITIL constitutes practical approaches for quality management of IT services and infrastructure processes with focus on automating processes, improving service management and integrating the IT department into business.

Some of these objectives are:

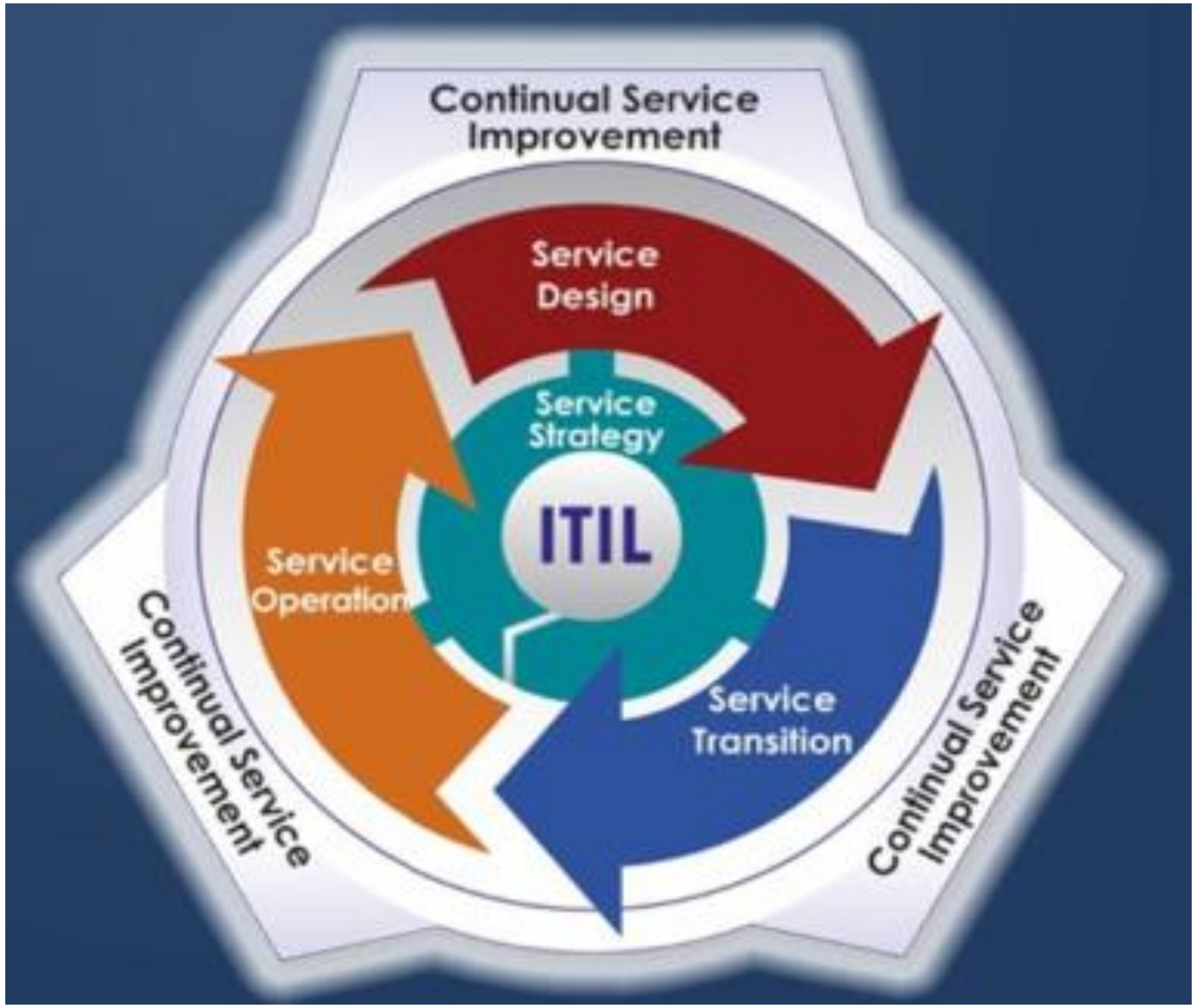
- **Best Practice Guidance**
  - Methodology of what works in actual practice derived from practitioners around the world
- **Non Proprietary**
  - Not a single vendor view and do not have to pay to apply in your organization
- **Comprehensive**
  - Captures all of the essential service support and services delivery processes, and integrates them to work together

ITIL -3 : IT Service Lifecycle

Strategy that supports service organization in examining the service opportunities proactively

Service Strategy: Understands organizational objectives and customer needs and provides strategic guidance for investments in services
Includes service value definition, business case development, service assets, market analysis and service provider types
Service Design: turns strategy into a plan for delivering business objectives and technology
Service Transition: develops and improves capabilities for introducing new services into supported environments
Relates to delivery of services required by business into live/operational use
Service Operation: Manages services in supported environments.
Provides best practice for achieving delivery of agreed levels of service
Continual Service Improvement: incremental and large scale improvements to services

ITIL -3 : IT Service Lifecycle


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

ITIL V4 - Holistic Service Management

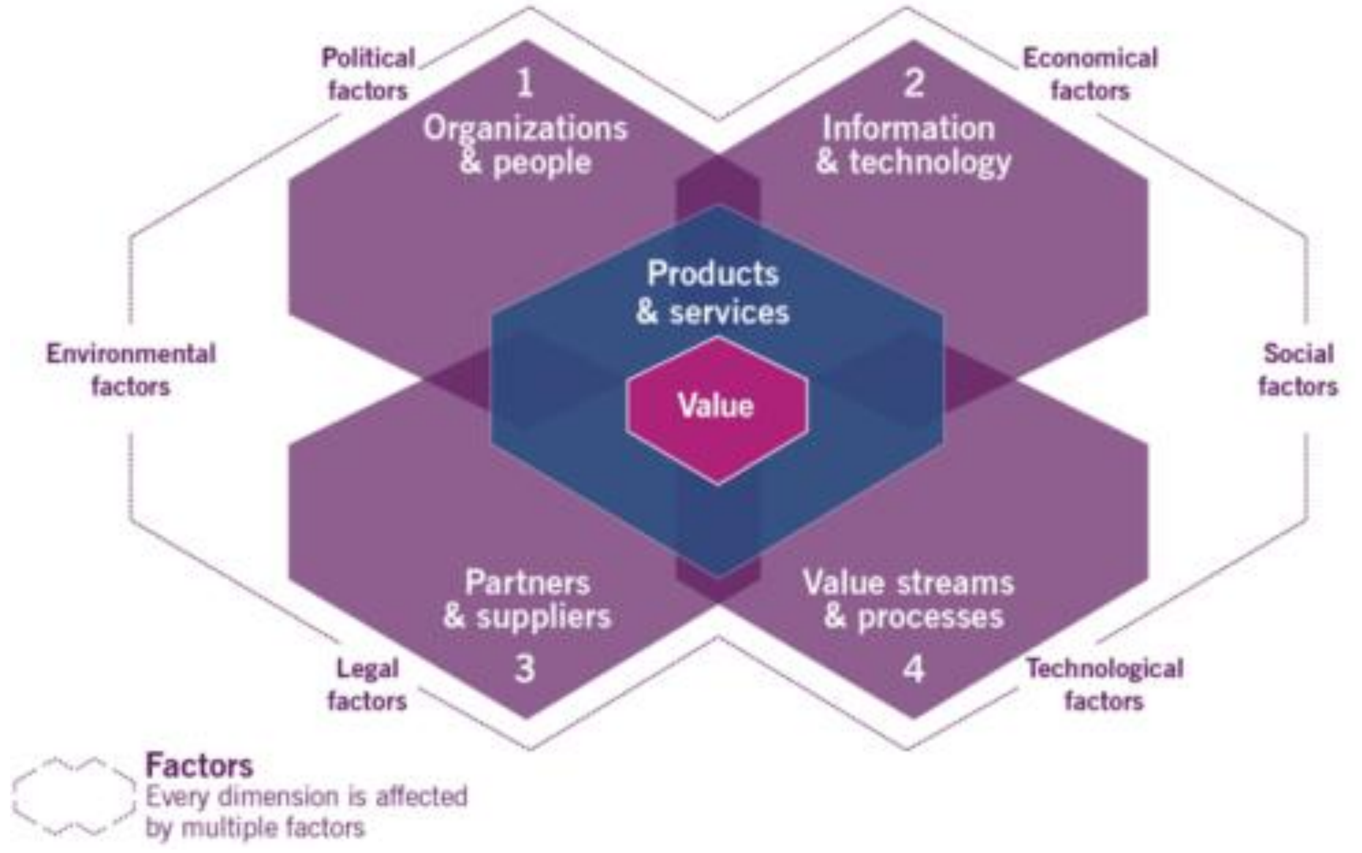
ITIL V4 facilitates value to customers and stakeholders by looking at service management in 4 dimensions

- **Organizations and People**: organization needs culture that supports objectives and the right level of capacity and competency among workforce
- **Information and Technology**: includes the information, knowledge and technologies required for management of services
- **Partners and suppliers**: organizations relationships with other businesses in design, deployment, delivery, support and continual improvement
- **Value streams and processes**: various parts of the organization work in integrated and coordinated way is important to enable value creation

Expands from processes to practices by factoring in culture, technology, information and data management

34 management practices; various types of guidance, such as key terms and concepts, success factors, key activities, information objects, etc.

ITIL V4 – The 4 dimensions



More details at:
https://blog.ifs.com/2022/07/itil4-organizations-and-people/

ITIL V4 - Holistic Service Management …

ITIL v4 defines Service Value System, built around Service value chain

The ITIL service lifecycle (ITIL v3) has been replaced with the ITIL service value system (SVS) and the service value chain within it

How Does the Service Value Chain Work?

There are six kinds of activities in the Service Value Chain. Each one represents a broad activity type into which *any and all* of your current (and future) service management activities will fit:

PLAN – All types of planning, at all levels

ENGAGE – Any and all interactions with people who are *external* to the service value chain (employees, customers, management, partners/suppliers)

DESIGN AND TRANSITION – Business analysis and development of new and improved services

OBTAIN/BUILD – Any new resource brought into the value chain is sourced via obtain/build

DELIVER AND SUPPORT – Provisioning services and providing help and information

IMPROVE – Improvements at all levels

Software Engineering

ITIL V4 - Holistic Service Management …

Guiding Principles of ITIL V4

- ➢Focus on value
- ➢Start where you are
- ➢Progress iteratively with feedback
- ➢Collaborate and promote visibility
- ➢Think and Work holistically
- ➢Keep it simple and practical
- ➢Optimize and automate

Reference:
https://blog.ifs.com/2021/05/itil4-focus-on-value/

Lot more details can be found in the attached document.

Common Myths concerning implementation of ITIL

- ➢You must implement all ITIL or no ITIL at all
- ➢ITIL is based on infrastructure management principles
- ➢ITIL applies mostly to data center operations
- ➢Everyone needs to be trained on ITIL fundamentals
- ➢Full understanding of ITIL requires purchase of library
- ➢ITIL processes should be implemented at one time
- ➢ITIL provides detailed templates for implementation
- ➢ITIL framework applies only to large shops
- ➢ITIL recommends tools to use for implementation
- ➢There is little need to understand ITIL origins

Visit the following links for more details:
https://www.informit.com/articles/article.aspx?p=175932&seqNum=179
https://www.informit.com/articles/article.aspx?p=175932&seqNum=180

Software Engineering

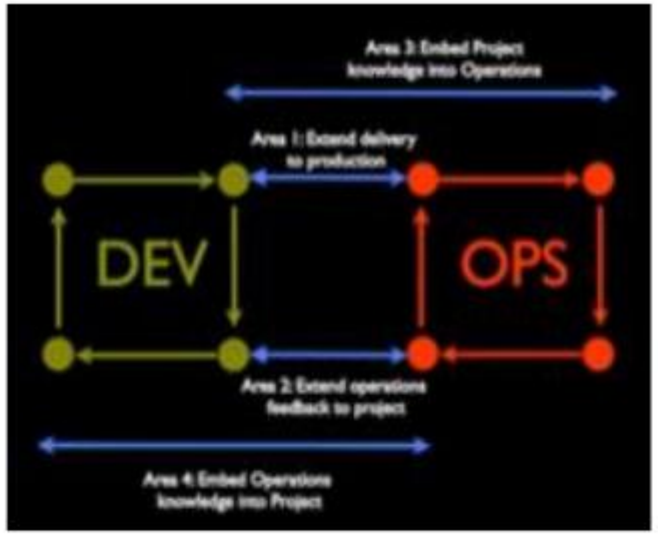
DevOps - Introduction

An introductory video:
https://www.google.co.in/search?q=what+is+DevOps&source=lnms&tbm=vid&sa=X&ved=2ahUKEwj48IOm-q37AhVtSGwGHR1vAecQ_AUoA3oECAIQBQ&biw=1366&bih=625&dpr=1#fpstate=ive&vld=cid:d4f77e34,vid:0yWAtQ6wYNM

Software Engineering

DevOps - Introduction

- DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver (Deploy and Support) applications and services at high velocity and become more effective
- <u>It is the blending of the terms development and operations</u>
- Represent a collaborative or shared approach to the tasks performed by a company’s **Application Development** teams (following SDLC) and **IT operations** teams
- The term DevOps is also used as an operational philosophy that promotes better communication and knowledge between the above two Dev and Ops teams

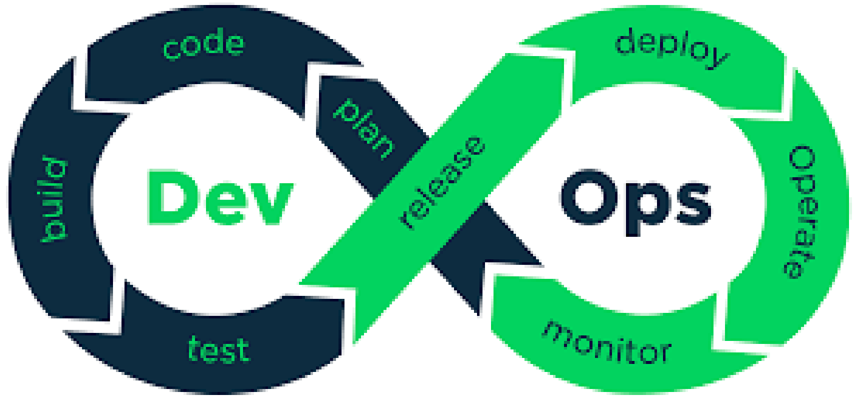

Area 3: Embed Project knowledge into Operations
Area 1: Extend delivery to production
DEV
OPS
Area 2: Extend operations feedback to project
Area 4: Embed Operations knowledge into Project

What is DevOps?

- DevOps looks to make the enterprise processes faster, more efficient, and more reliable with the intention of increasing the **business value**

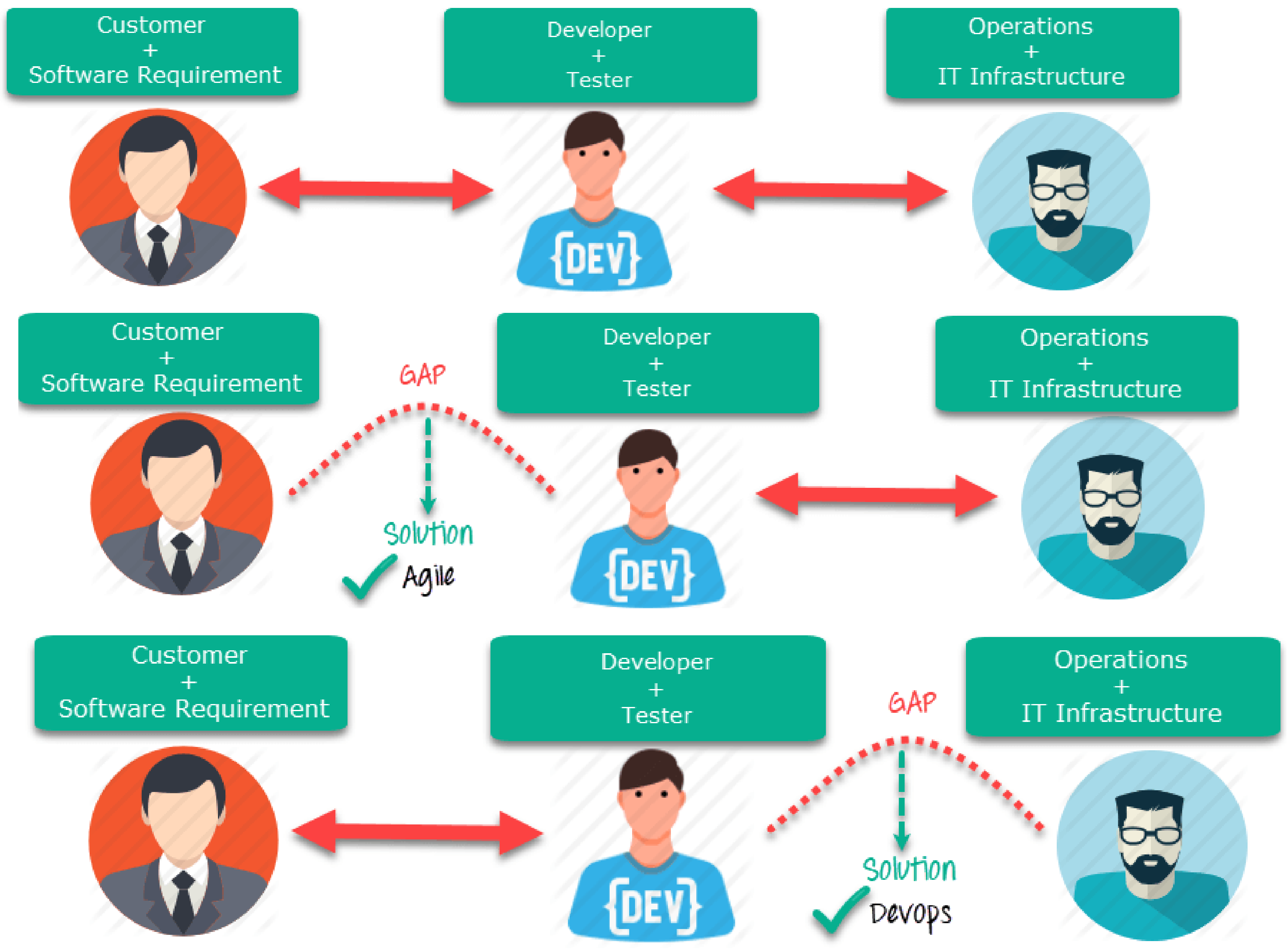
- Repetitive manual labor (error prone) is removed whenever possible

- DevOps in a sense is an extension of the agile way of doing things where you can deliver functionality incrementally and at a faster rate



Which Agile manifesto principle do you think DevOps follows?

Software Engineering
Agile v/s DevOps

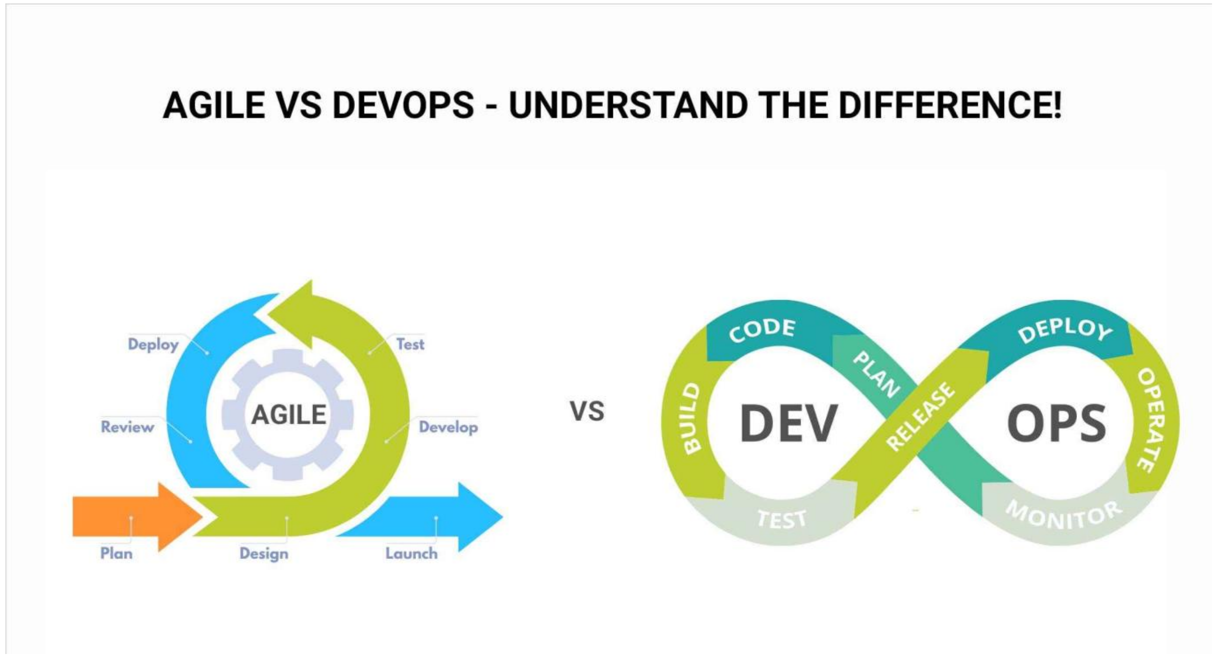


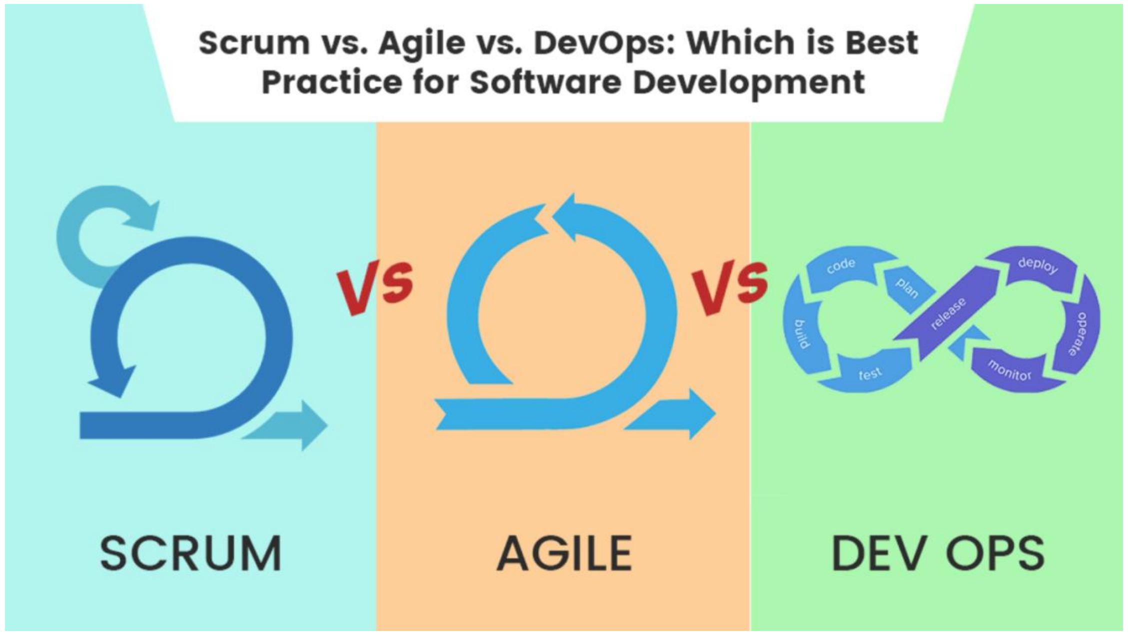
Customer + Software Requirement
Developer + Tester
Operations + IT Infrastructure
Customer + Software Requirement
GAP
Solution
Agile
Developer + Tester
Operations + IT Infrastructure
Customer + Software Requirement
Developer + Tester
GAP
Solution
Devops
Operations + IT Infrastructure

Software Engineering

What is DevOps?

- Deployment systems are maintained by DevOps engineers
- Can make the deliveries not only at the end of Scrum cycles with a periodicity of two to four weeks but even faster (could even be many times a day) and be more efficient




Scrum vs. Agile vs. DevOps: Which is Best Practice for Software Development
VS
VS
code
plan
deploy
build
release
operate
test
monitor
SCRUM
AGILE
DEV OPS

Software Engineering
DevOps: Concepts and terminologies

Recap some of the terminologies you have already learnt which are also used in DevOps:

- SDLC (Waterfall model, Agile approach)
- Operations Methodologies (ITSM, ITIL)
- SCM terminologies associated with Development, Release and Deployment concepts

VERSION CONTROL

- Version control systems like Git record changes to files or sets of files stored within the system
- Developers can make changes individually or within groups and save these changes through commits or revisions within the system in one way or another
- Ability to commit, compare, merge & restore past revisions to objects to the repository allows for richer cooperation and collaboration within and between teams and minimizes the risk

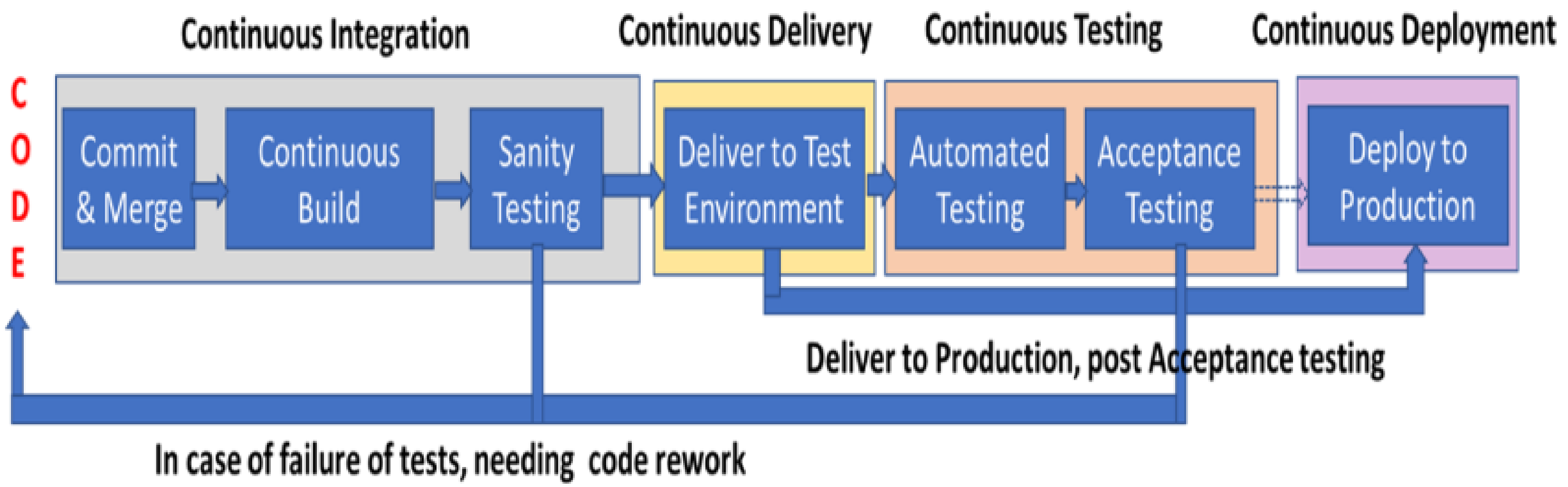
DevOps pipeline

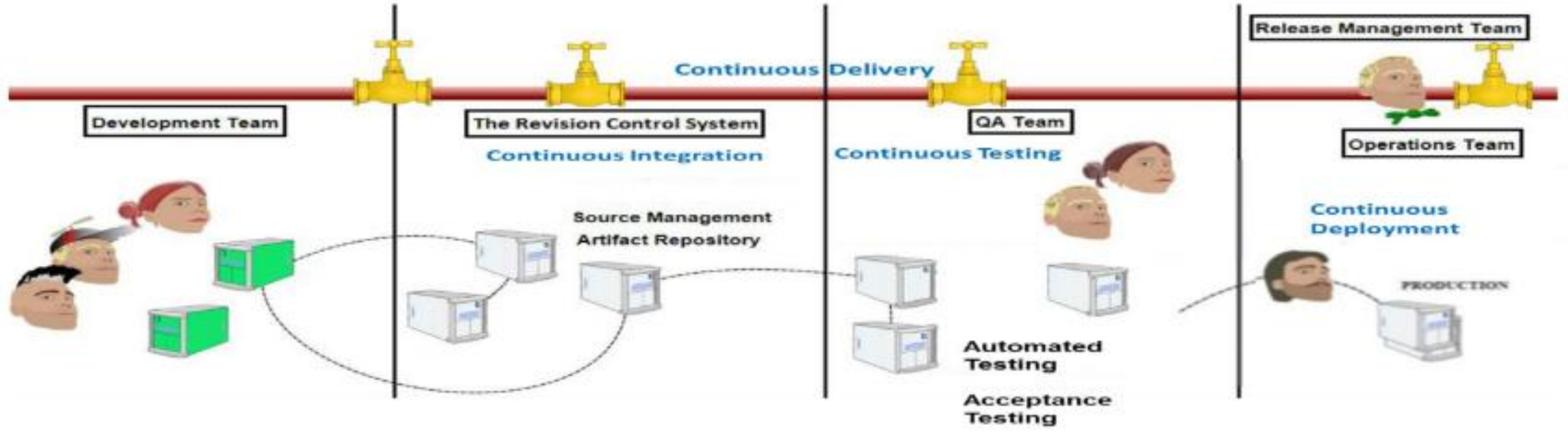
What is the DevOps pipeline?

A DevOps pipeline is a set of automated processes and tools that allows both developers and operations professionals to work cohesively to build and deploy code to a production environment.

DevOps: Concepts and terminologies

DevOps PIPELINE



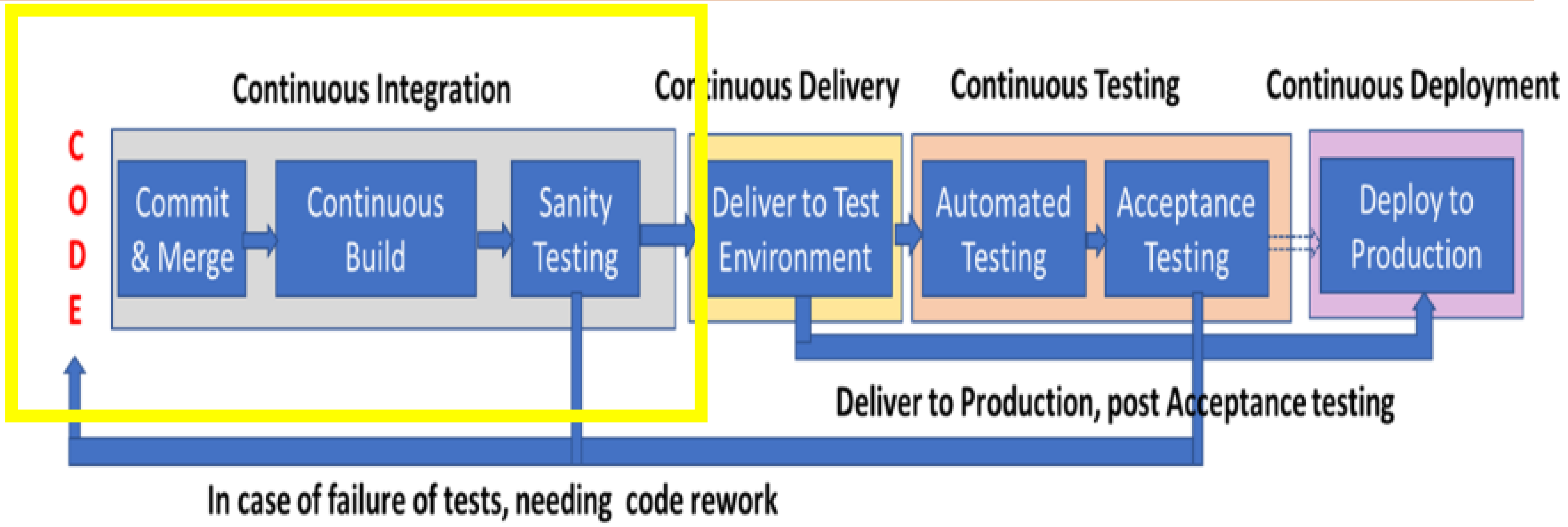


DevOps: Supporting pillars

Four common themes or pillars of DevOps

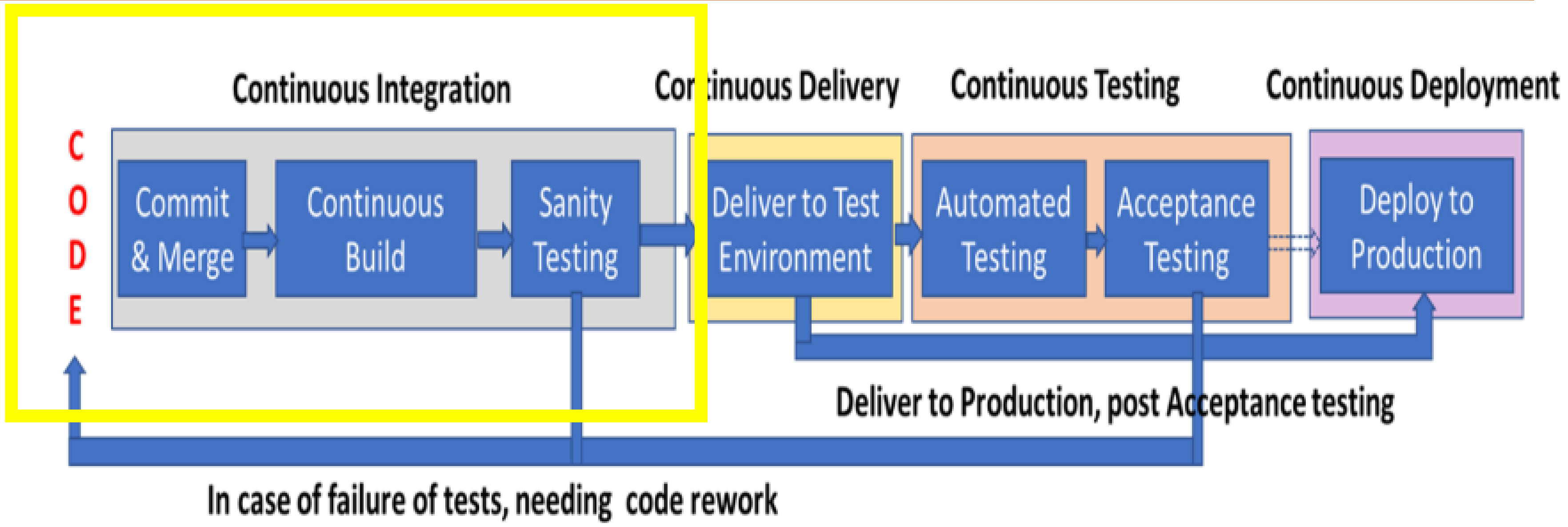
- **Collaboration**
- **Affinity**
- **Tools**
- **Scaling**

DevOps Pipeline: Continuous Integration



- **Continuous Integration (CI)** involves continuously merging the code written by developers with a mainline or “master” branch, <u>frequently throughout the day</u>
- Developers take a snap shot or branch code from a master branch and work on their individual components and check-in and merge the changes back into the mainline
- Merging of the code can happen multiple times through the day or in some periodicity and during merge if they find a merge conflicts, its resolved and then the code is merged

Software Engineering
DevOps Pipeline: Continuous Integration

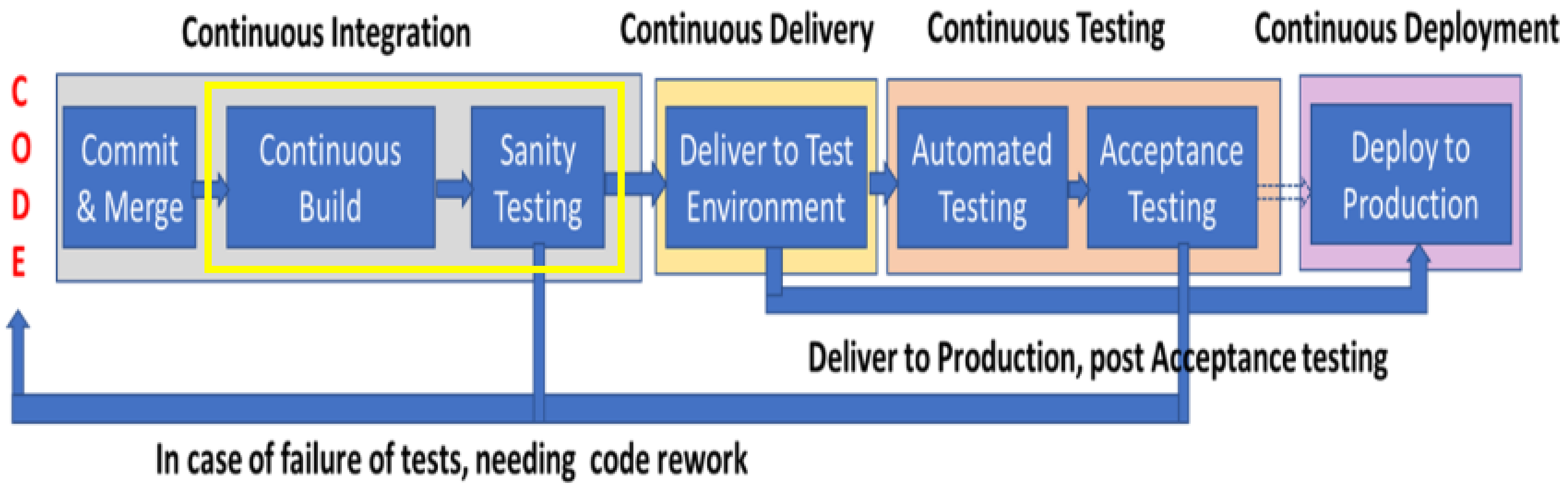


Benefits of CI:

- Change sets are small and hence, work for resolving conflicts would be small & incremental
- Supports detecting of issues early
- Avoids difficult integrations, increased visibility and communications
- Reduced debugging time

What would happen if code is not merged frequently?

DevOps Pipeline: Continuous Integration



Continuous Build

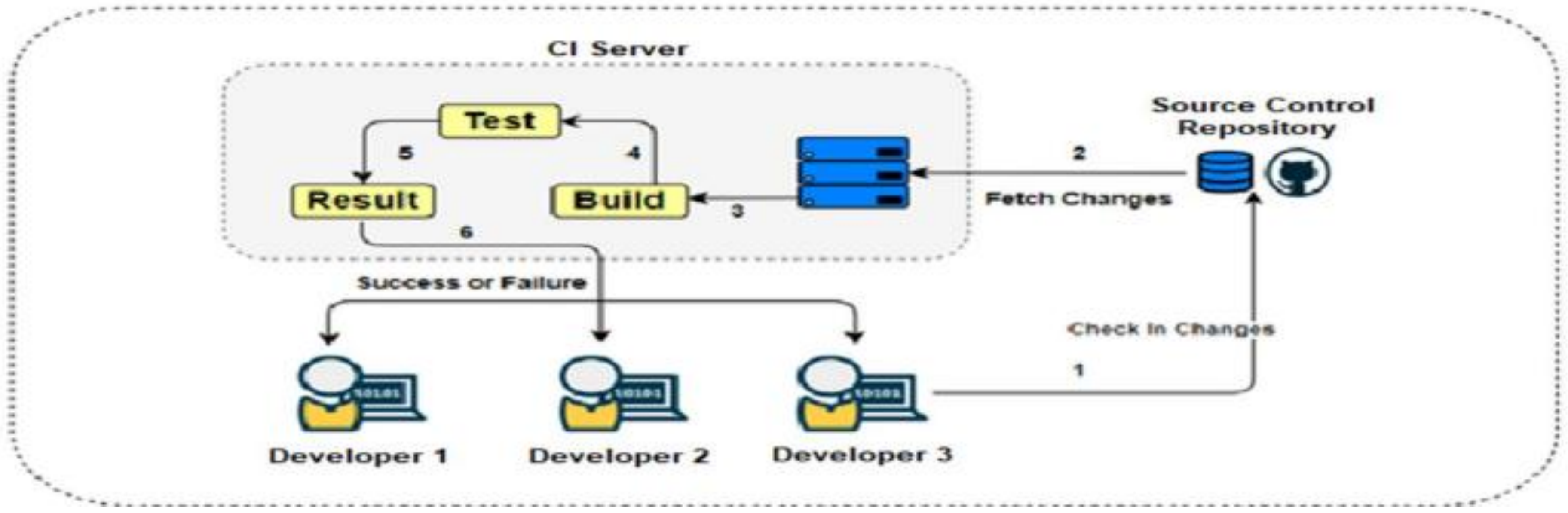
- Build involves compiling, handling dependencies and involves generating an executable product
- This may also involve running some static code analysis tools like Lint

Sanity testing

The executable built, is run with some sanity tests to ensure that the total compilation is successful, the dependencies are all resolved and the executable is in an executable state.

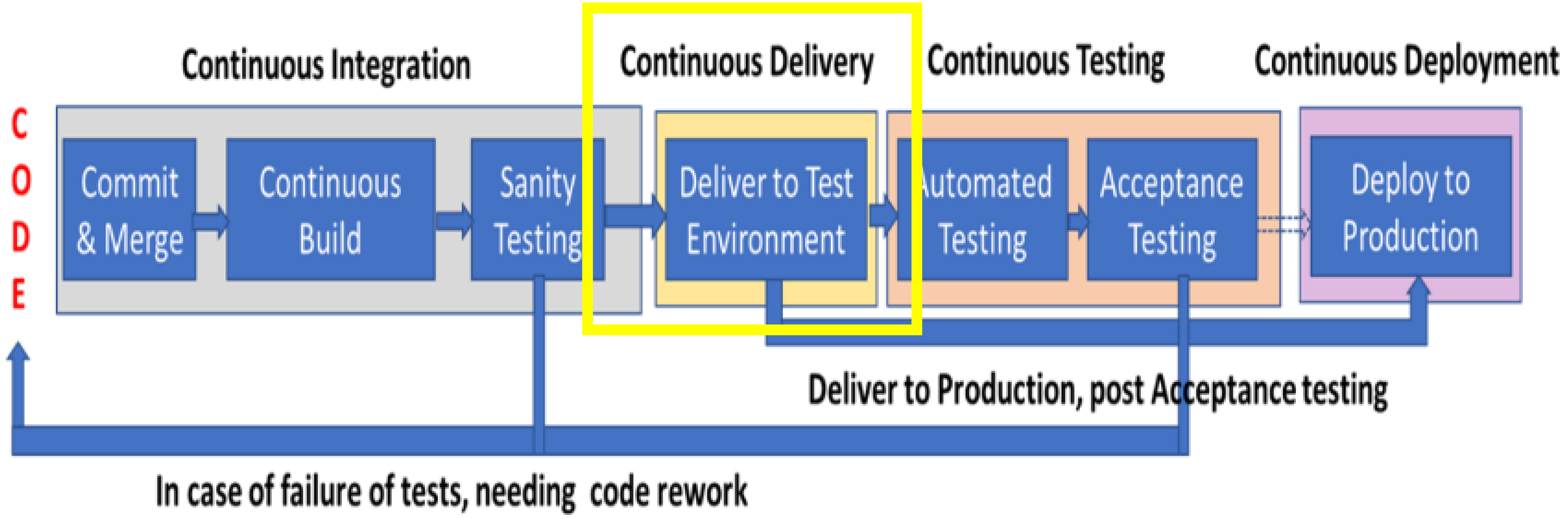
DevOps Pipeline: Continuous Integration

Sanity testing



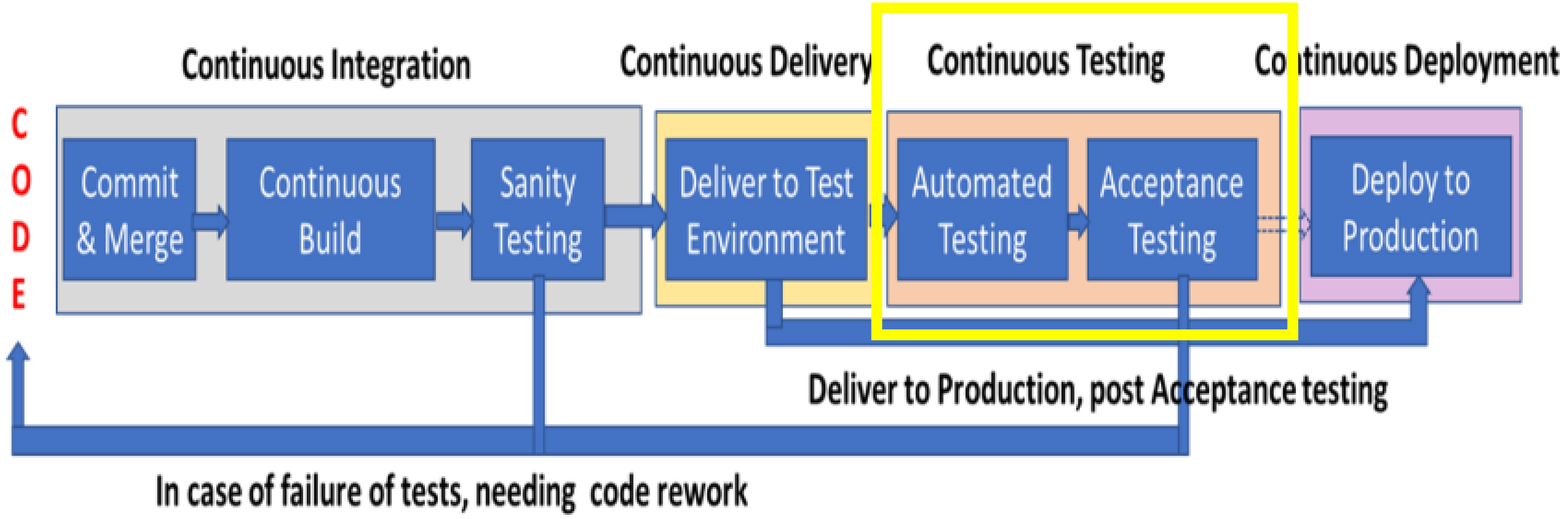
Software Engineering

DevOps Pipeline: Continuous Delivery



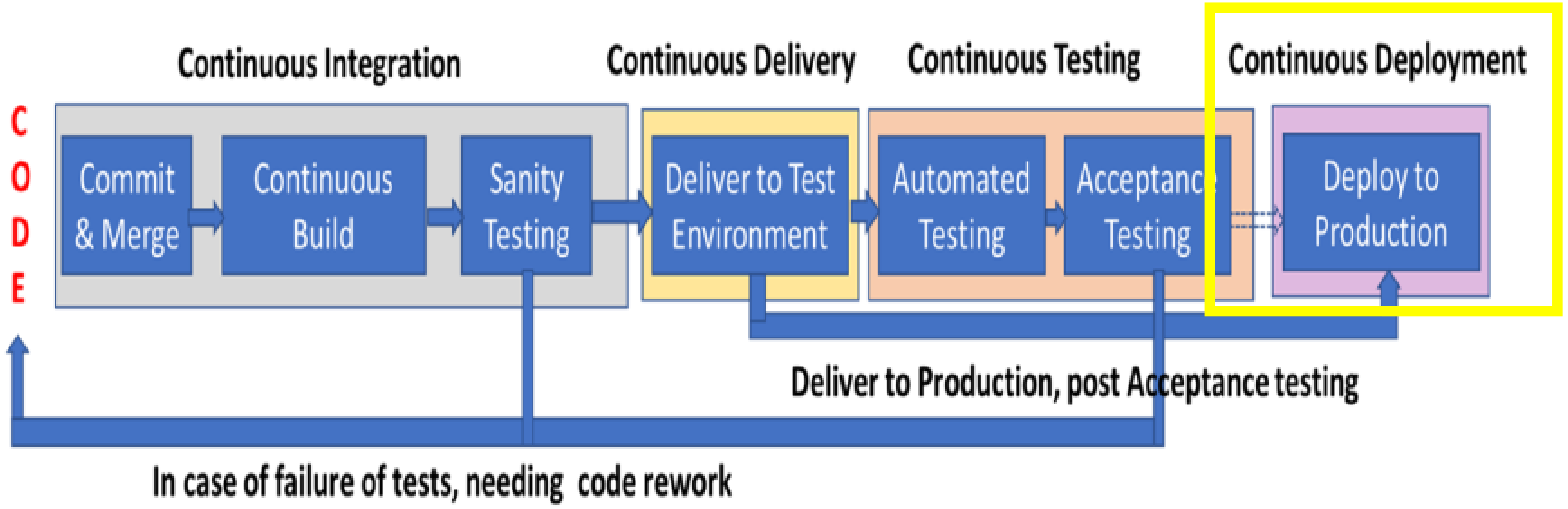
- **Continuous Delivery** is the frequent shipping of code to a given environment
- Integrated bits are ready to be deployed & this supports release of the new changes to your test teams/customers quickly in a sustainable way
- <u>Process may be triggered by the last step in the continuous integration process</u>
- Drives the Business outcomes like deploy on demand, and supports faster time to market and of higher quality and may use tools like Jenkin Plugins

DevOps Pipeline: Continuous Testing



- **Continuous testing** is the process of executing <u>predominantly automated tests </u>for validating the code to ensure that the application works as envisaged and is free of bugs or defects and can be continuously deployed
- Designed to be executed with minimum wait times and provide the earliest possible feedback & support detection (or prevention) of the risks
- Entry criteria for testing by the QE process groups would be successful completion of the unit and integration tests
- In the QA environment, these could include some static code analysis and validation of both functional requirements and non-functional requirements

DevOps Pipeline: Continuous Deployment



- Continuous Deployment (CD) is a software release process
- Uses the delivery mechanisms for deploying the automated testing validated product, immediately and autonomously to a production environment
- Typically used in highly mature DevOps teams
- Significant number of times, validated integrated components are batched together and are then deployed into customer environment

DevOps v/s ITIL

Compare Agile and DevOps methodologies

DevOps is a culture and approach which looks to remove the silos of Development activities of building a product and the Operations activities of deployment, support and upkeep. This also focusses on increasing the velocity of deployment of products facilitating it by communication, collaboration, automation, usage of tools and integration among IT professionals.

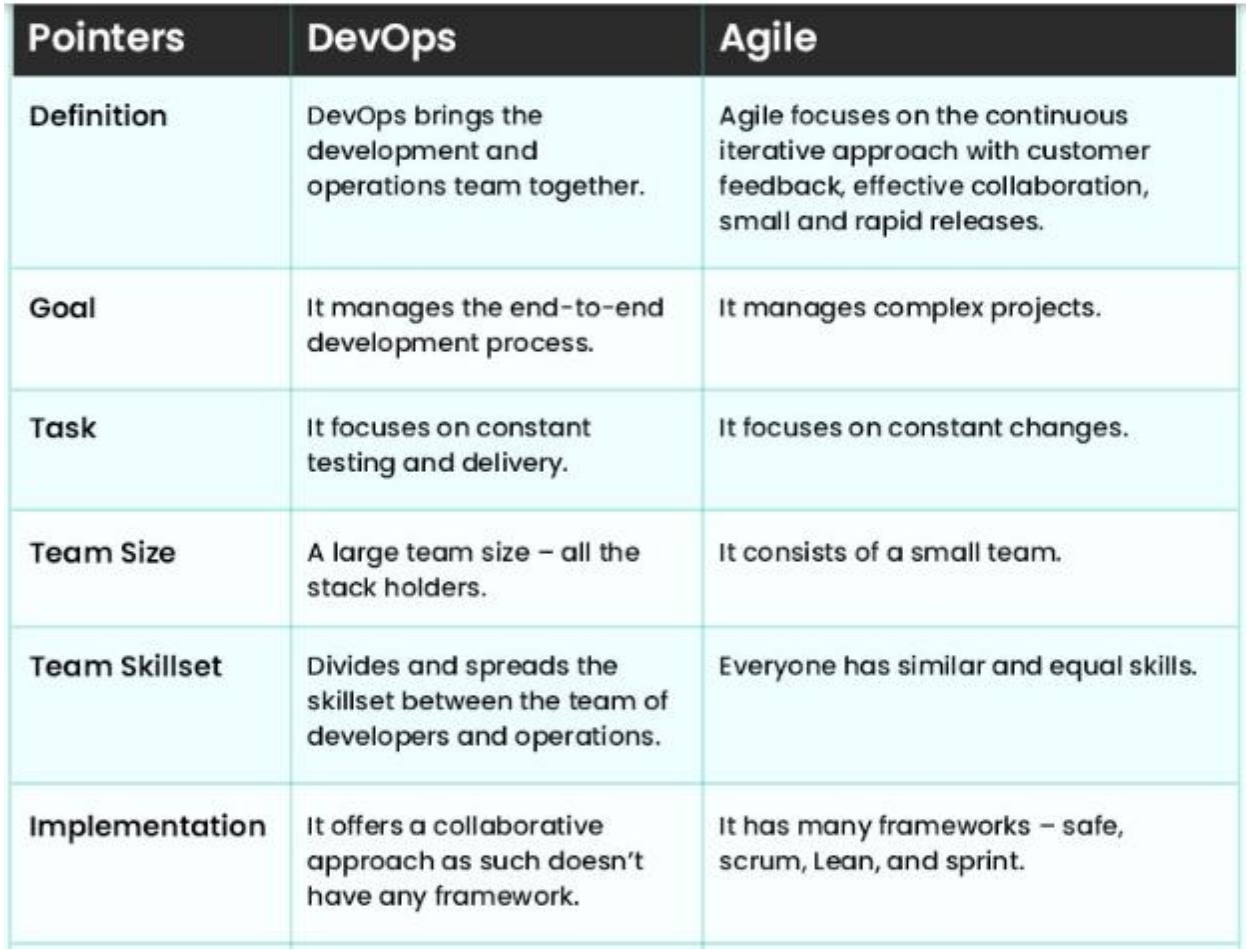
Agile is a methodology or a type of iterative SDLC which as a process, supports changes, ensures more collaborations between developers and other stakeholders, reduces the planning overhead and delivers products or part of products periodically say once 2-4 weeks.

DevOps focuses on constant testing and delivery while the Agile process focuses on constant changes.

The target area of Agile is Software development whereas the Target area of DevOps is to give end-to-end business solutions and fast delivery.

DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.

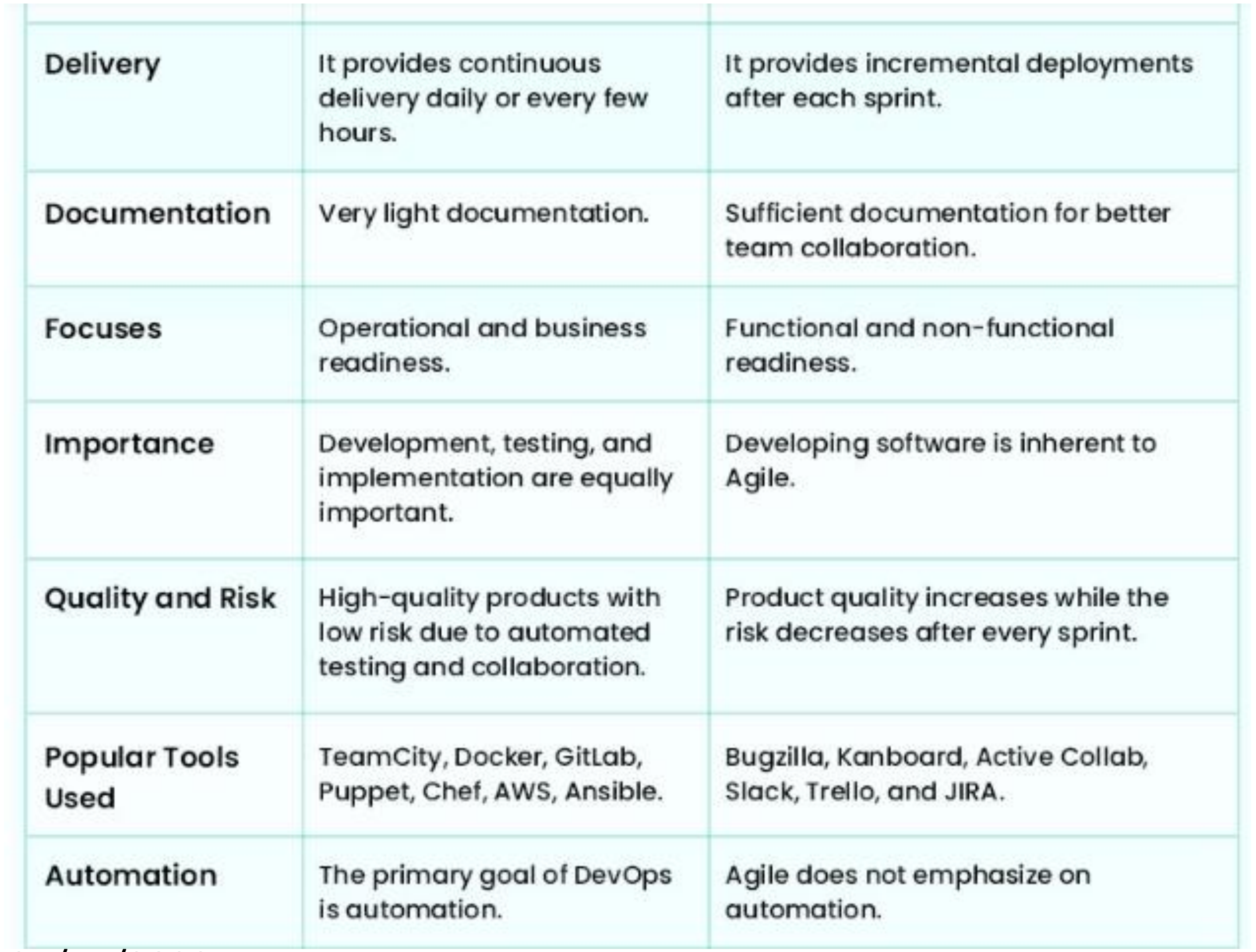
Agile v/s DevOps



Pointers	DevOps	Agile
Definition	DevOps brings the development and operations team together.	Agile focuses on the continuous iterative approach with customer feedback, effective collaboration, small and rapid releases.
Goal	It manages the end-to-end development process.	It manages complex projects.
Task	It focuses on constant testing and delivery.	It focuses on constant changes.
Team Size	A large team size – all the stack holders.	It consists of a small team.
Team Skillset	Divides and spreads the skillset between the team of developers and operations.	Everyone has similar and equal skills.
Implementation	It offers a collaborative approach as such doesn't have any framework.	It has many frameworks – safe, scrum, Lean, and sprint.

Software Engineering

Agile v/s DevOps



Delivery	It provides continuous delivery daily or every few hours.	It provides incremental deployments after each sprint.
Documentation	Very light documentation.	Sufficient documentation for better team collaboration.
Focuses	Operational and business readiness.	Functional and non-functional readiness.
Importance	Development, testing, and implementation are equally important.	Developing software is inherent to Agile.
Quality and Risk	High-quality products with low risk due to automated testing and collaboration.	Product quality increases while the risk decreases after every sprint.
Popular Tools Used	TeamCity, Docker, GitLab, Puppet, Chef, AWS, Ansible.	Bugzilla, Kanboard, Active Collab, Slack, Trello, and JIRA.
Automation	The primary goal of DevOps is automation.	Agile does not emphasize on automation.

Software Engineering
DevOps - Pillars

Recap

<u>Four common themes or pillars of DevOps</u>

- ➢**Collaboration**
- ➢**Affinity**
- ➢**Tools**
- ➢**Scaling**

Software Engineering
DevOps - Pillars

- **Collaboration** is the process of working towards a specific outcome through supporting interactions and the input and support of multiple people & groups

- Effective collaboration involves
  - Theory of Mind
  - Equal participation
  - Communication

- <u>Theory of Mind</u>
  is the ability to recognize ones perspective and understanding that others have distinct and different perspective based on their own context

DevOps Pillars - Collaboration

Considering different aspects of getting people to work together more effectively could be by having:

- A shared goal
- Empathy of a collaborative culture through effective communication
- Teams are effective when based on relationships built on trust, empathy & reciprocity
- Where hierarchies are less emphasized

DevOps Pillars - Affinity

- There needs to exist a strong relationship between individuals, teams and departments along with collaborative relationships to ensure the Dev and Ops groups work together

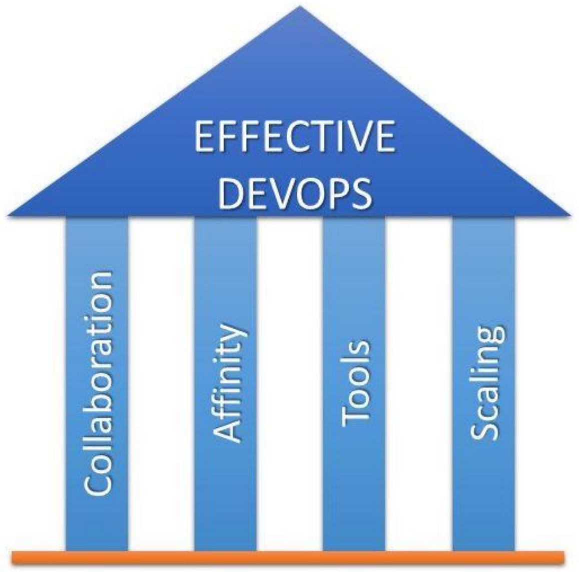
- This is achieved by having teams with **strong affinity** between them
  - **Affinity** is the measure of the strength of the relationship between individuals, teams, business units, and even companies

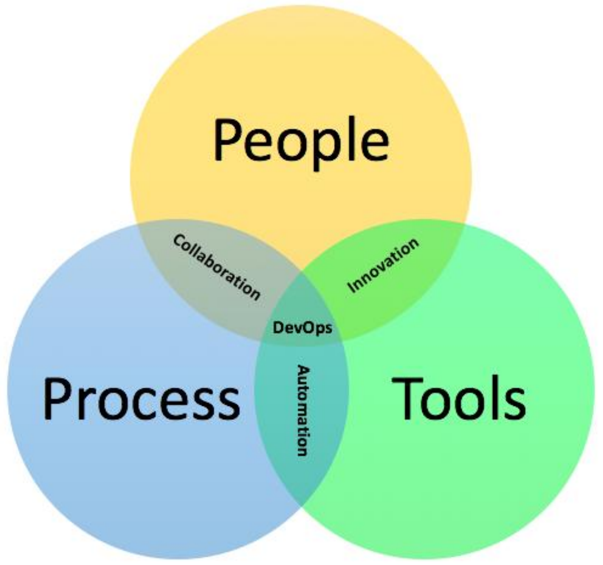
  - **Building Affinity** is the process of building these inter-team relationships, navigating differing goals or metrics while keeping in mind shared organizational goals, and fostering empathy and learning between different groups of people

- Affinity is achieved by having:
  - Shared Values
  - Consistent team culture
  - Team cohesion

DevOps Pillars - Affinity

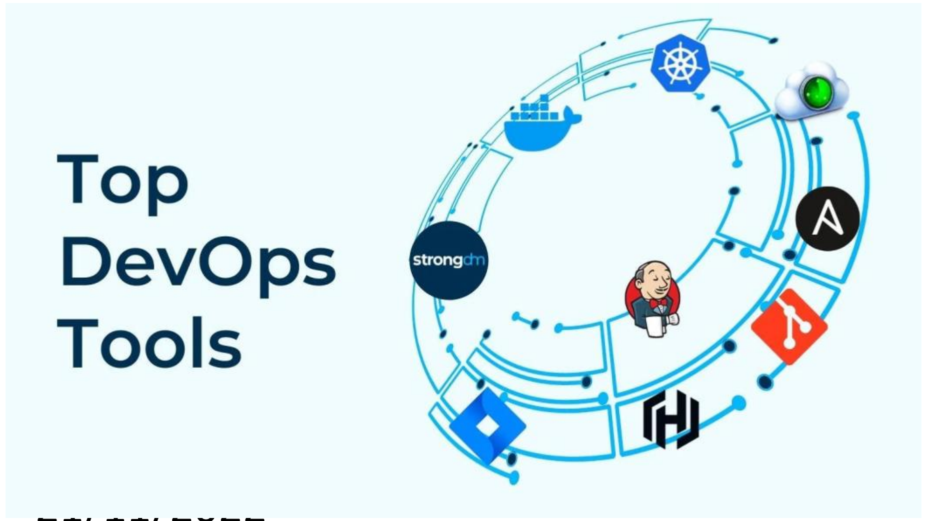
- Affinity can be measured by several factors like
  - ➢ Shared Time
  - ➢ Intensity of the relationships
  - ➢ Reciprocity of shared stories
  - ➢ Reciprocity of support

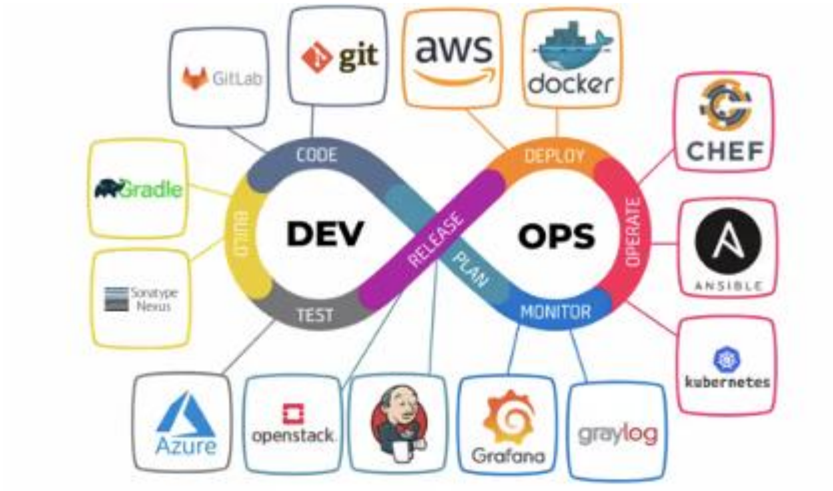




Software Engineering
DevOps Pillars - Tools

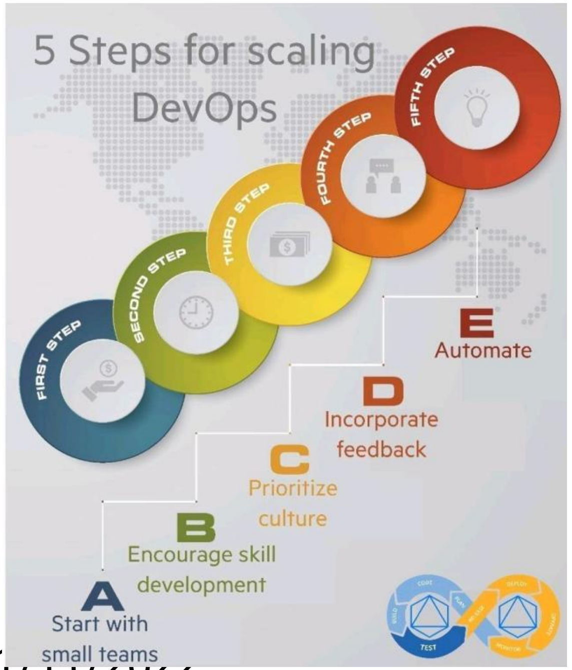
- **Tools** are important in the DevOps environment as they act as an accelerator, driving change based on the current culture and direction
- Tools & their ease of use impacts the acceptance and proliferation of specific aspects of the culture as they become the common language between teams
- If the cost of collaboration is high, not investing in tools (or worse, investing in poor tools) raises this cost
- Examples of tools : System provisioning tools, Build Tools, Automation tools, Testing tools, Monitoring tools, Log extraction tools, Deployment tools etc.

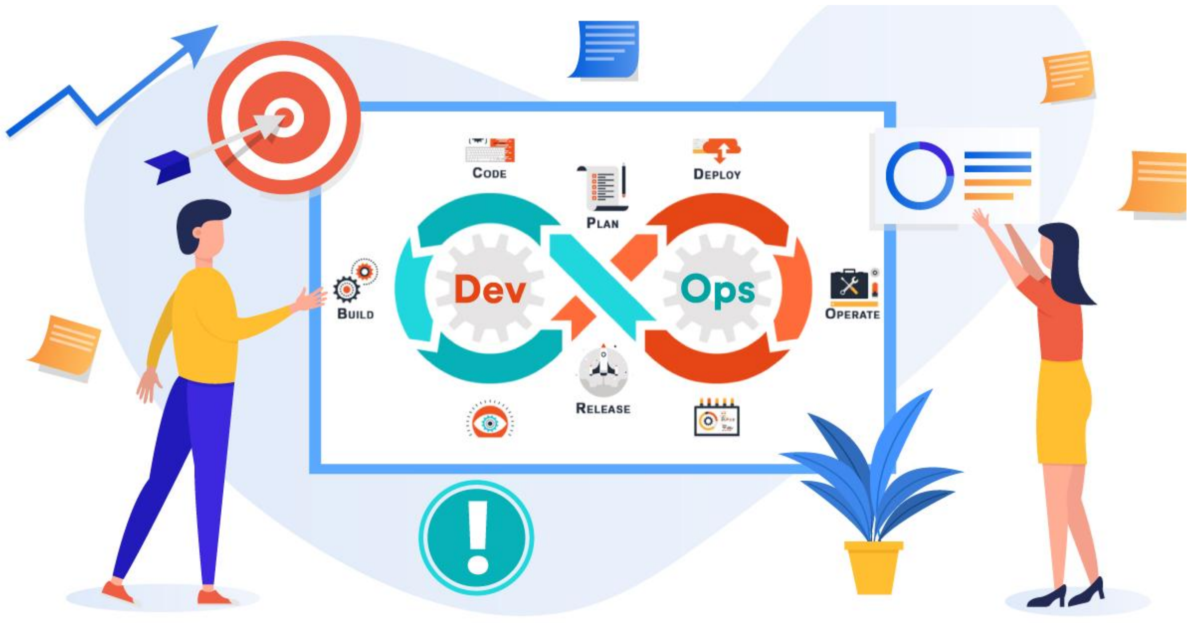




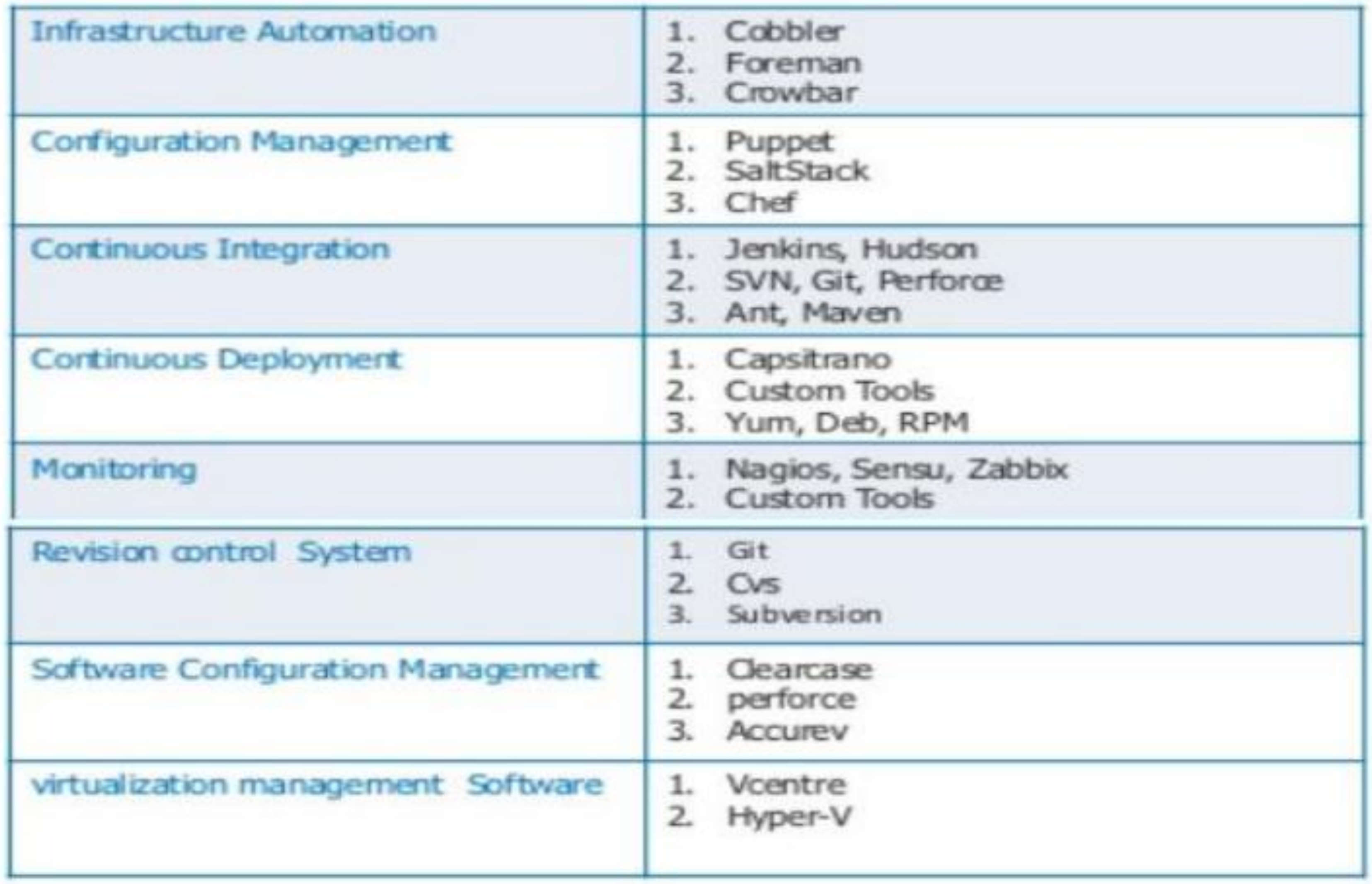
DevOps Pillars - Scaling

- **Scaling** has a focus on the processes and pivots that organizations must adopt throughout their lifecycles
- Scaling takes into account how the other pillars of effective DevOps can be applied throughout organizations as they grow, mature, and even shrink in large organizations
- Scaling could be for *organization, infrastructure, teams (hiring, retention, outsourcing), complexity, workload*





Software Engineering
Tools used in DevOps



Infrastructure Automation	1. Cobbler 2. Foreman 3. Crowbar
Configuration Management	1. Puppet 2. SaltStack 3. Chef
Continuous Integration	1. Jenkins, Hudson 2. SVN, Git, Perforce 3. Ant, Maven
Continuous Deployment	1. Capsitrano 2. Custom Tools 3. Yum, Deb, RPM
Monitoring	1. Nagios, Sensu, Zabbix 2. Custom Tools
Revision control System	1. Git 2. Cvs 3. Subversion
Software Configuration Management	1. Clearcase 2. perforce 3. Accurev
virtualization management Software	1. Vcentre 2. Hyper-V

DevOps v/s ITIL

You can find some very interesting information about this topic at:

https://www.atlassian.com/itsm/itil/devops-vs-itil

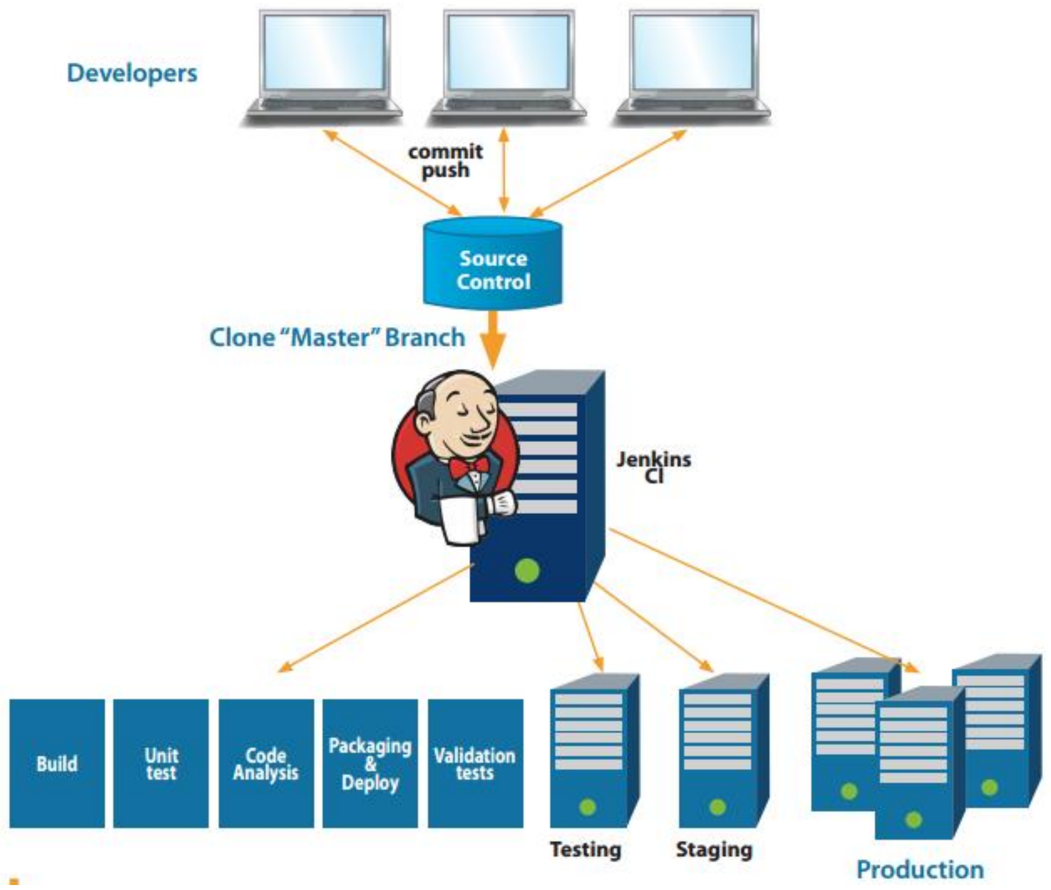
What is Jenkins?

A quick look at Jenkins is here.

What is Jenkins?

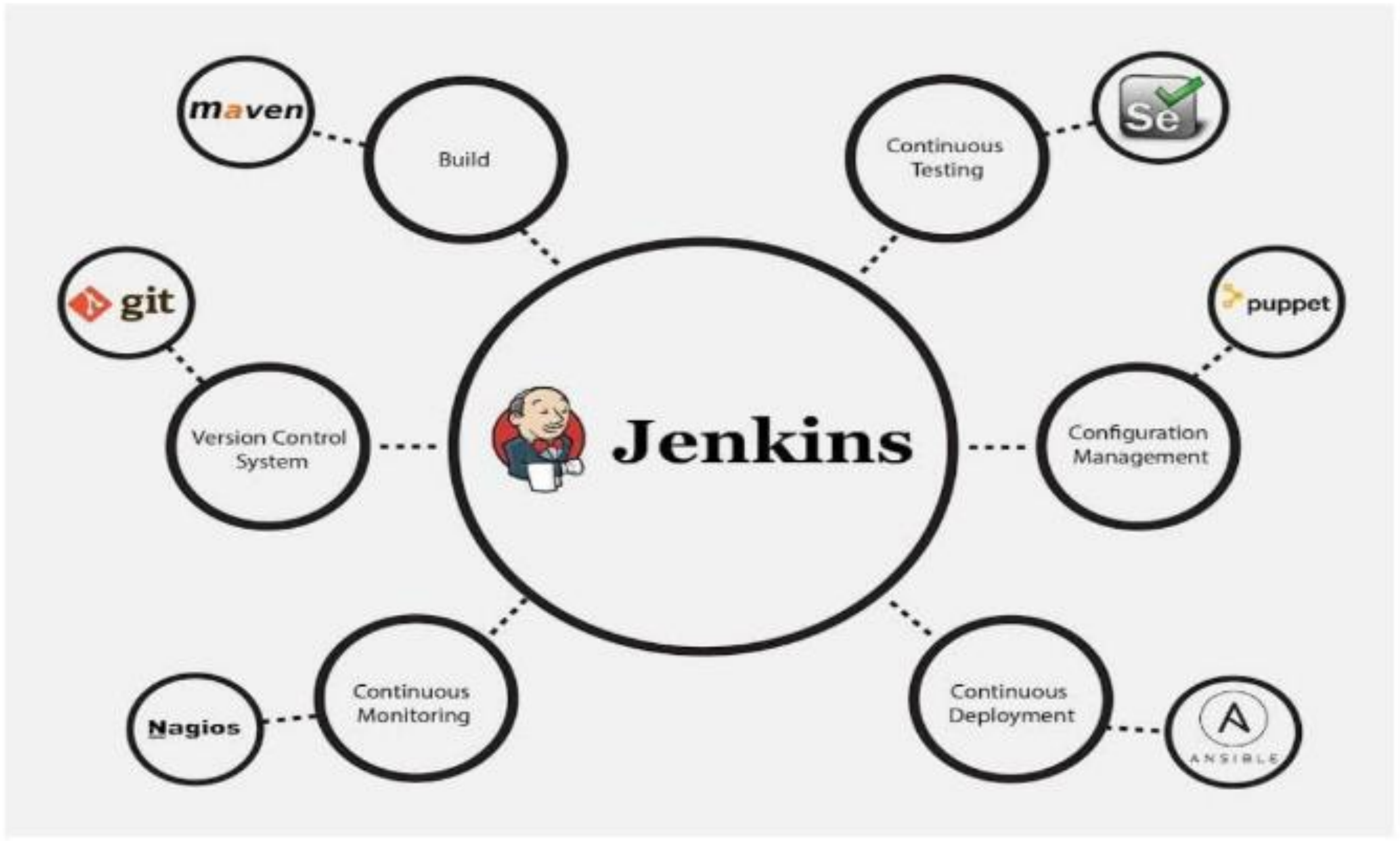
Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose.

- Generates test reports
- Integrates with many different Version Control Systems
- Pushes to various artifact repositories
- Deploys directly to production or test environments
- Notifies stakeholders of build status
- It’s plugins allow integration of various DevOps stages
- Supports UI Customization



Software Engineering
Jenkins

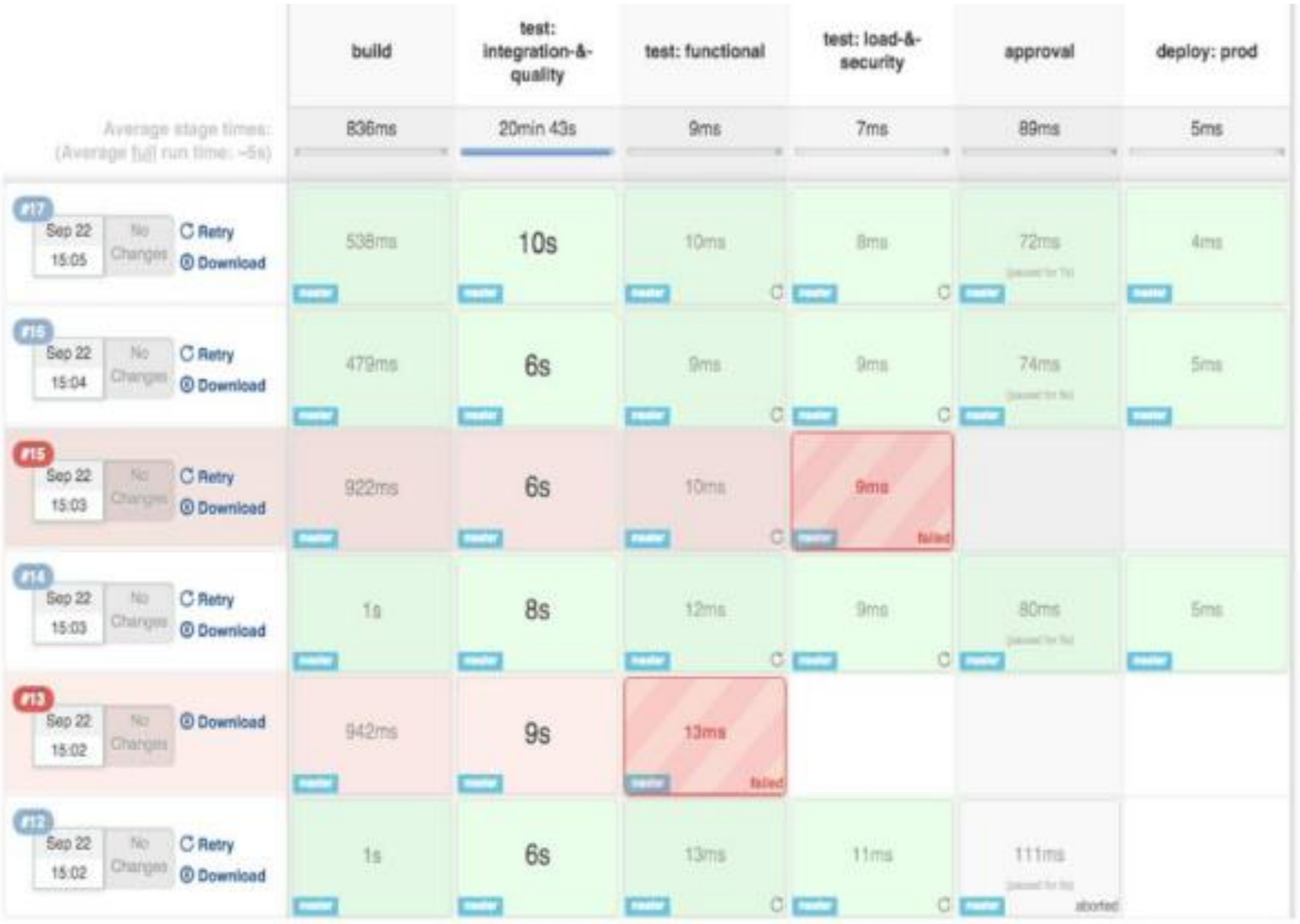
Jenkins plugins allow integration of various DevOps stages



Software Engineering

Jenkins – Pipeline

- Each stage in the pipeline is represented by a cell
- The column headers represent the name of the job that is being run at that stage in the pipeline
- The green cells indicate that the stage/job is successful, red cells indicate that the job has failed

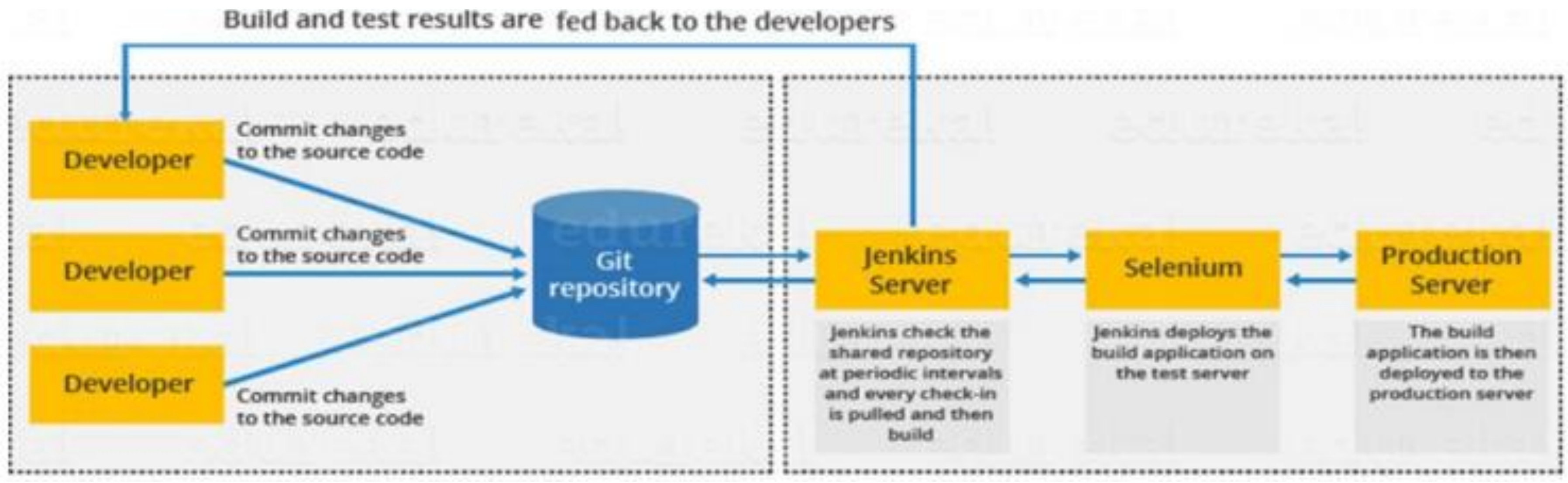


	build	test: integration-&-quality	test: functional	test: load-&-security	approval	deploy: prod
Average stage times: (Average full run time: ~5s)	836ms	20min 43s	9ms	7ms	89ms	5ms
#17 Sep 22 15:05 No Changes Retry Download	538ms	10s	10ms	8ms	72ms	4ms
#16 Sep 22 15:04 No Changes Retry Download	479ms	6s	9ms	9ms	74ms	5ms
#15 Sep 22 15:03 No Changes Retry Download	922ms	6s	10ms	9ms failed		
#14 Sep 22 15:03 No Changes Retry Download	1s	8s	12ms	9ms	80ms	5ms
#13 Sep 22 15:02 No Changes Download	942ms	9s	13ms failed			
#12 Sep 22 15:02 No Changes Retry Download	1s	6s	13ms	11ms	111ms aborted	

Jenkins – Continuous Integration

It involves the following steps:

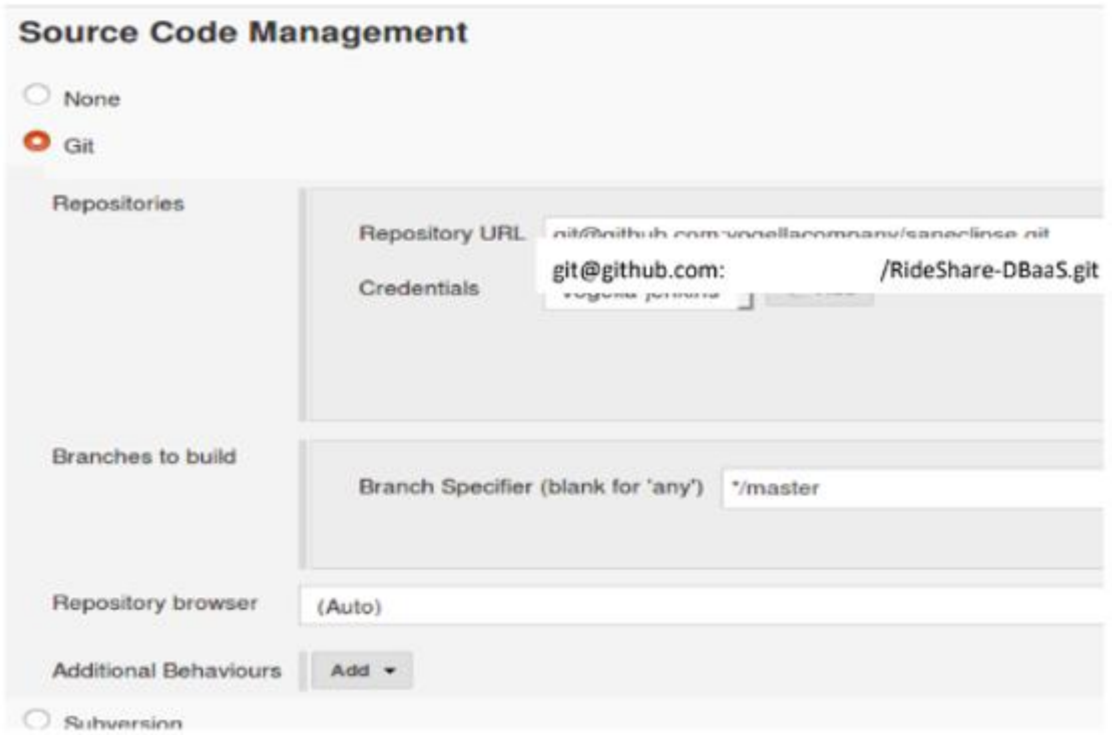
- Developers are required to commit changes to the source code in a shared repository several times a day or more frequently
- Every commit made in the repository is then built
- This allows the teams to detect the problems early
- There are several other tools that depend on Continuous Integrations tool. Like -
    - Deploying the build application on the test server
    - Providing the concerned teams with the build and test results.


Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

Software Engineering

Jenkins – Continuous Integration Illustration

The code is being sourced from a Github repository, and the job will be run on that repository, based on some Build Trigger.



Source Code Management

- ○ None
- ● Git

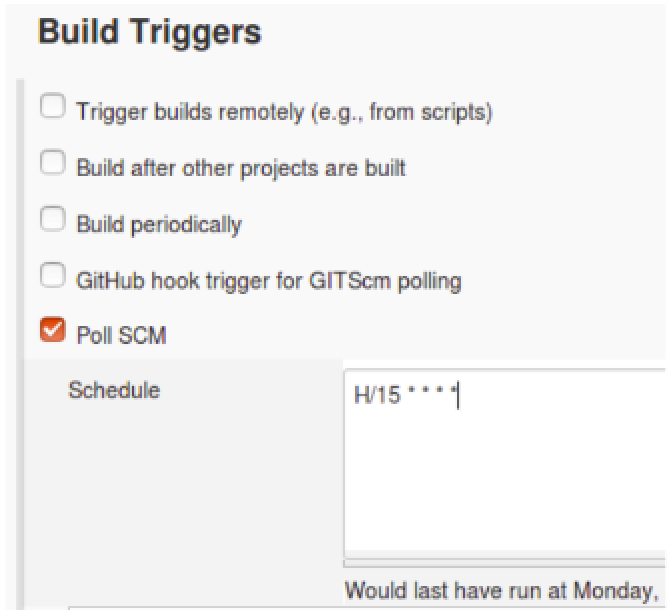
Repositories
- Repository URL: git@github.com: /RideShare-DBaaS.git
- Credentials

Branches to build
- Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours: Add ▾

- ○ Subversion



Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☑ Poll SCM

Schedule: H/15 * * * *

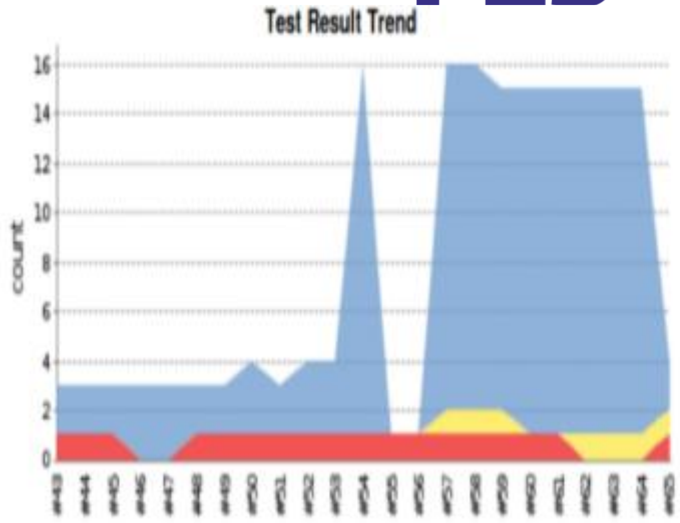
Would last have run at Monday,

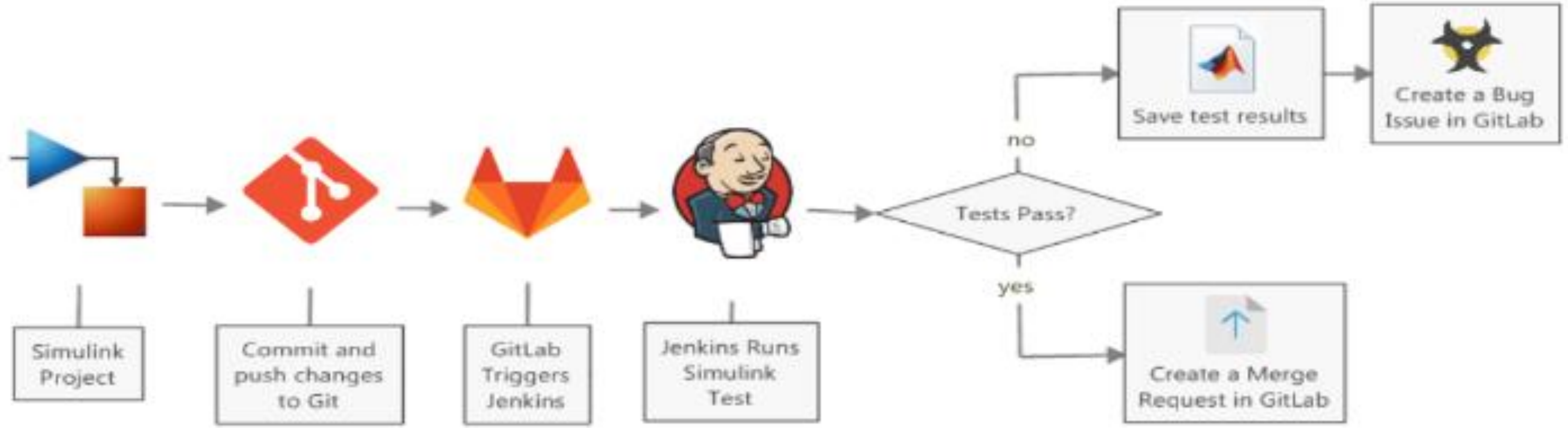
Software Engineering

Jenkins – Continuous Testing

- Automated continuous testing facilitates higher velocity of deployment. This could as seen for the Selenium based tests or using something like the MathWorks Simulink Project which helps in collaboration and Automation with source code control

- Jenkins can schedule tests to run at a specific time and also displays statistics like number of tests executed, time taken, failures, error messages, test results and trends etc.

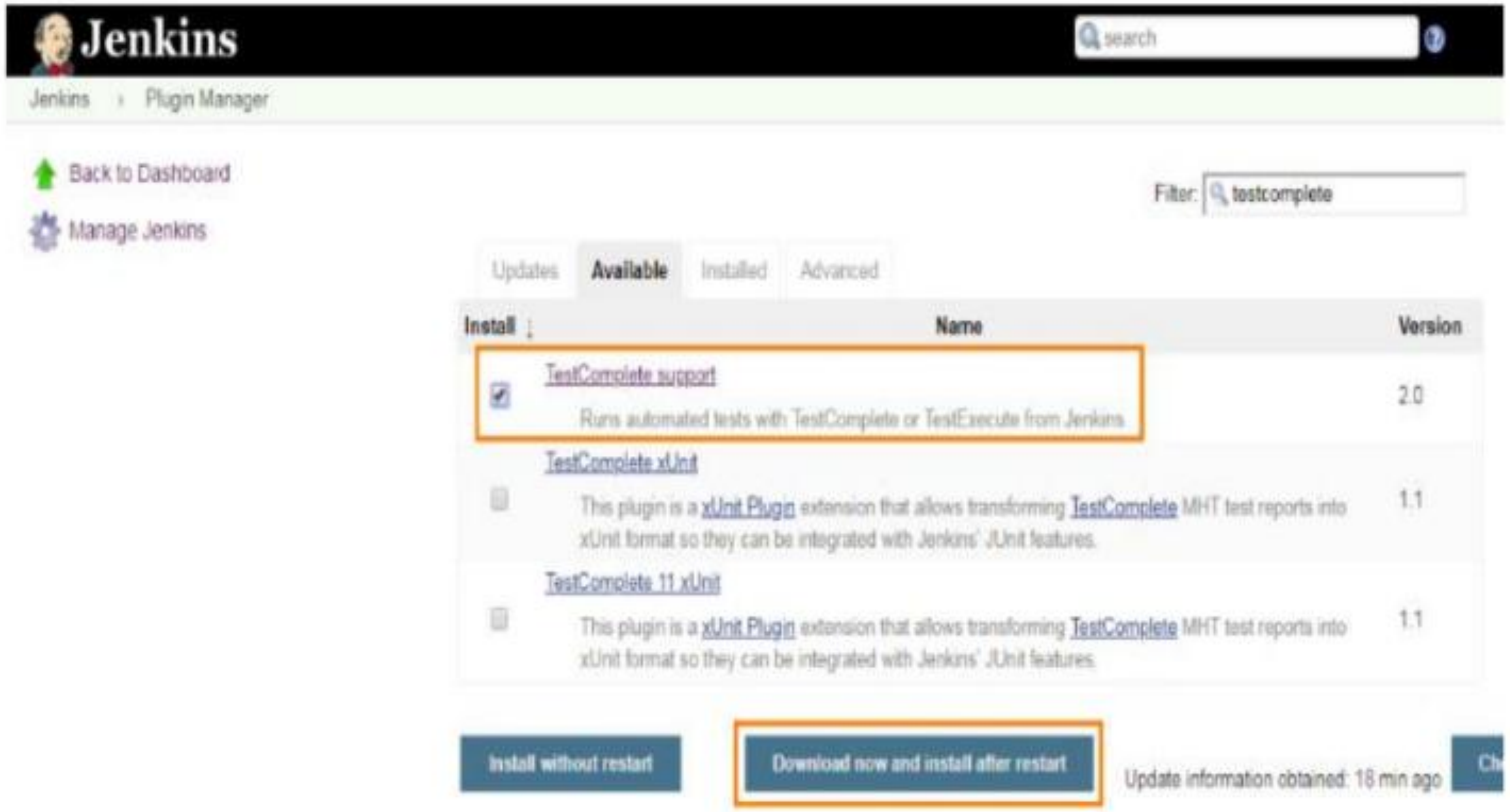



Simulink Project
Commit and push changes to Git
GitLab Triggers Jenkins
Jenkins Runs Simulink Test
Tests Pass?
no
yes
Save test results
Create a Bug Issue in GitLab
Create a Merge Request in GitLab

21/11/2022

Jenkins – Continuous Testing Illustration

A Plugin called TestComplete is being used to run the automated tests as a part of the CI pipeline.



Install	Name	Version
✓	TestComplete support Runs automated tests with TestComplete or TestExecute from Jenkins	2.0
☐	TestComplete xUnit This plugin is a xUnit Plugin extension that allows transforming TestComplete MHT test reports into xUnit format so they can be integrated with Jenkins' JUnit features.	1.1
☐	TestComplete 11 xUnit This plugin is a xUnit Plugin extension that allows transforming TestComplete MHT test reports into xUnit format so they can be integrated with Jenkins' JUnit features.	1.1

Install without restart
Download now and install after restart
Update information obtained: 18 min ago
Ch

THANK YOU

M S Anand
Department of Computer Science Engineering

anandms@yahoo.com