UPES

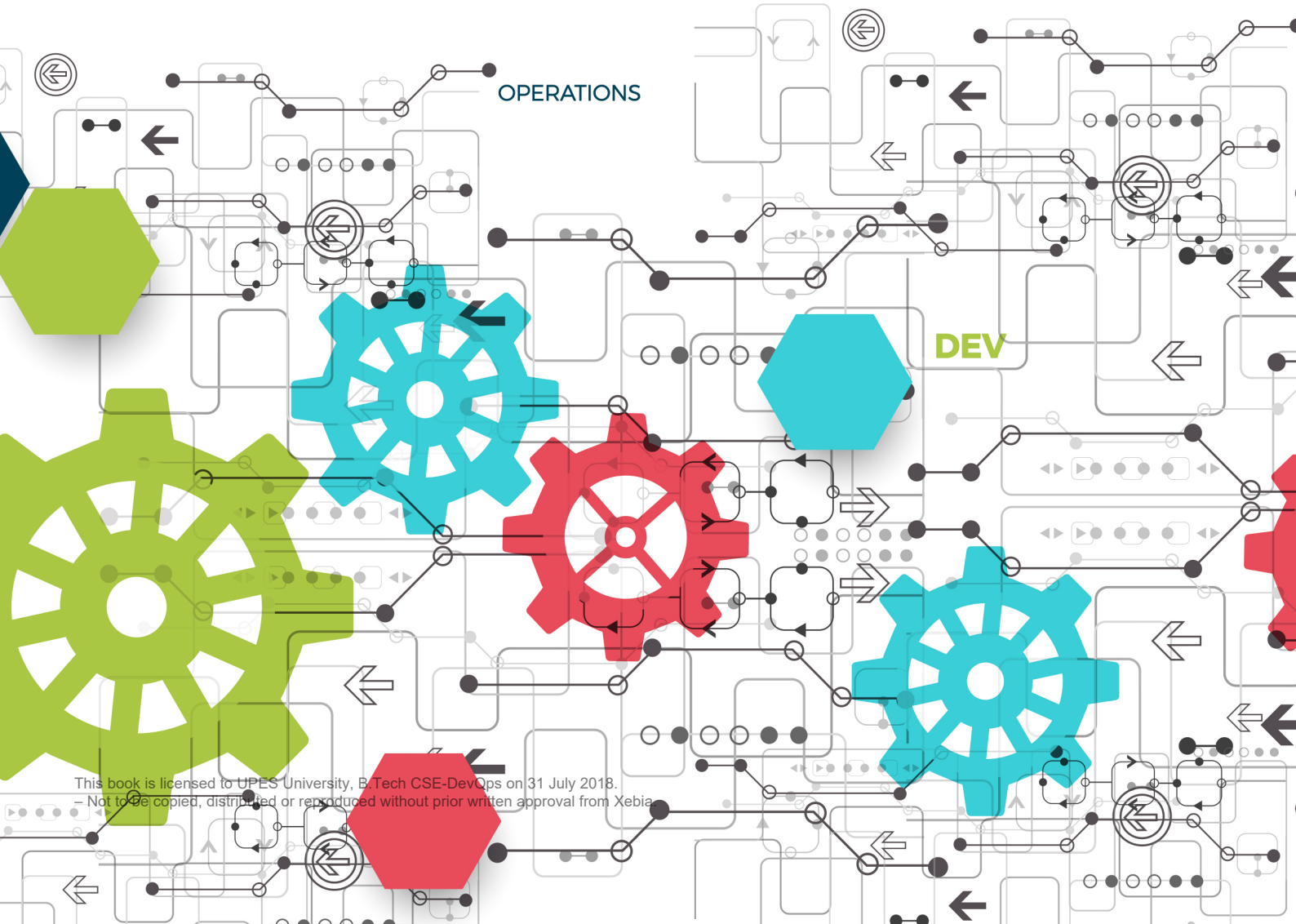
Xebia

B.Tech Computer Science and Engineering in DevOps

DEVOPS OVERVIEW

Semester **01** | **Student Handbook**

Release 1.0.0



Copyright & Disclaimer

B. TECH CSE with Specialization in DevOps

Version 1.0.0

Acknowledgements

We would like to sincerely thank the experts who have contributed to and shaped B. TECH CSE with Specialization in DevOps. Version 1.0.0

SME

Rajagopalan Varadan
A tech enthusiast who loves learning and working with cutting-edge technologies like DevOps, Big Data, Data science, Machine Learning, AWS & Open stack

Course Reviewers.

Aditya Kalia | Xebia
Maneet Kaur | Xebia
Sandeep Singh Rawat | Xebia
Abhishek Srivastava | Xebia
Rohit Sharma | Xebia

Review Board Members.

Anand Sahay | Xebia

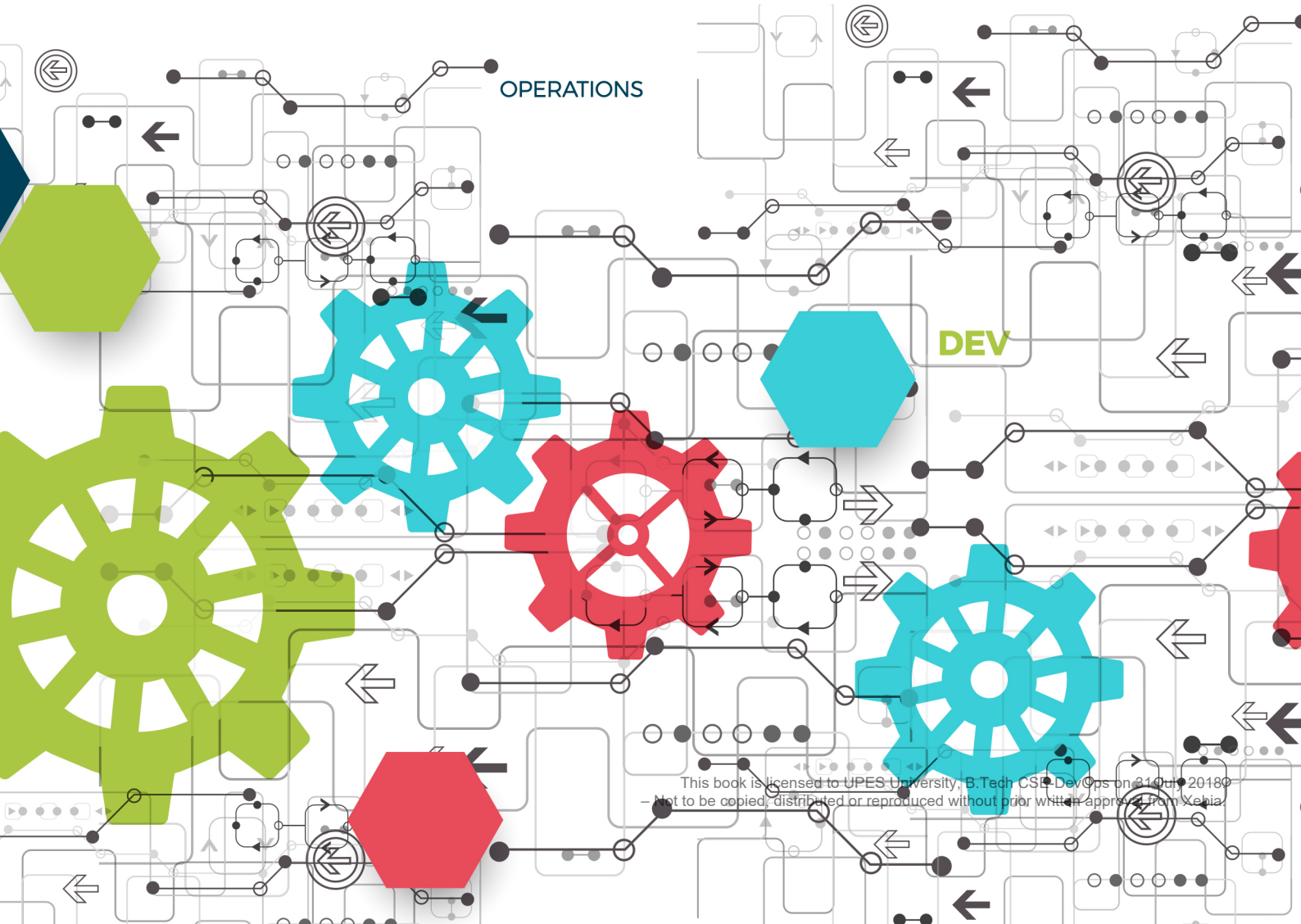
Xebia Group consists of seven specialized, interlinked companies: Xebia, Xebia Academy, XebiaLabs, StackState, GoDataDriven, Xpirit and Binx.io. With offices in Amsterdam and Hilversum (Netherlands), Paris, Delhi, Bangalore and Boston, we employ over 700 people worldwide. Our solutions address digital strategy; agile transformations; DevOps and continuous delivery; big data and data science; cloud infrastructures; agile software development; quality and test automation; and agile software security.

ODW is dedicated to provide innovative and creative solutions that contribute in growth of emerging technologies. As a learning experience provider, ODW strengths include providing unique, up to date content by combining industry best practices with leading edge technology. ODW delivers high quality solutions and services which focus on digital learning transformation.

B.Tech Computer Science and Engineering in DevOps

DEVOPS OVERVIEW

MODULE 5
CAMS (Culture, Automation, Measurement and Sharing)



Contents

MODULE 5

CAMS (Culture, Automation, Measurement and Sharing)

You will be talking discussing CAMS model in this module.

Module Learning Objectives

At the end of the Module you would be able to learn the following

- Explain CAMS model of DevOps
- Enumerate in detail about the culture, automation, measurement and sharing, the processes & tools used by organization for DevOps
- Elaborate the importance of test driven development and configuration management
- Explain the role of automation and the use of infrastructure automation
- Perform root-cause analysis for mitigating issues
- Understand the need for building a culture that promotes and facilitates organizational learning

Module Topics

The following topics that will be covered in the module:

1. CAMS - Culture
2. CAMS - Automation
3. CAMS - Measurement
4. CAMS - Sharing

5. Test-driven development
6. Configuration Management
7. Infrastructure Automation
8. Root Cause Analysis
9. Blamelessness
10. Organizational Learning

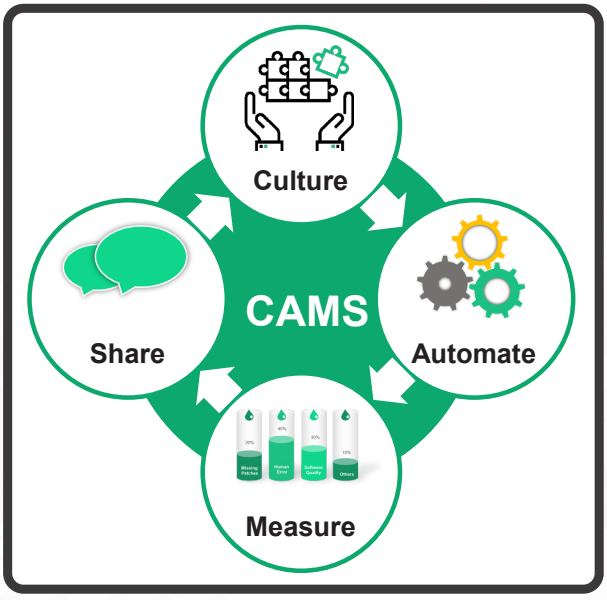
Reiterate the topics you will be covering in the module.

1. CAMS - Culture, Automation, Measurement and Sharing

CAMS model was popularized by Gene Kim, the founder of the phoenix project.

CAMS is an acronym that embodies the core values of DevOps, such as

- Culture
- Automation
- Measurement
- Sharing

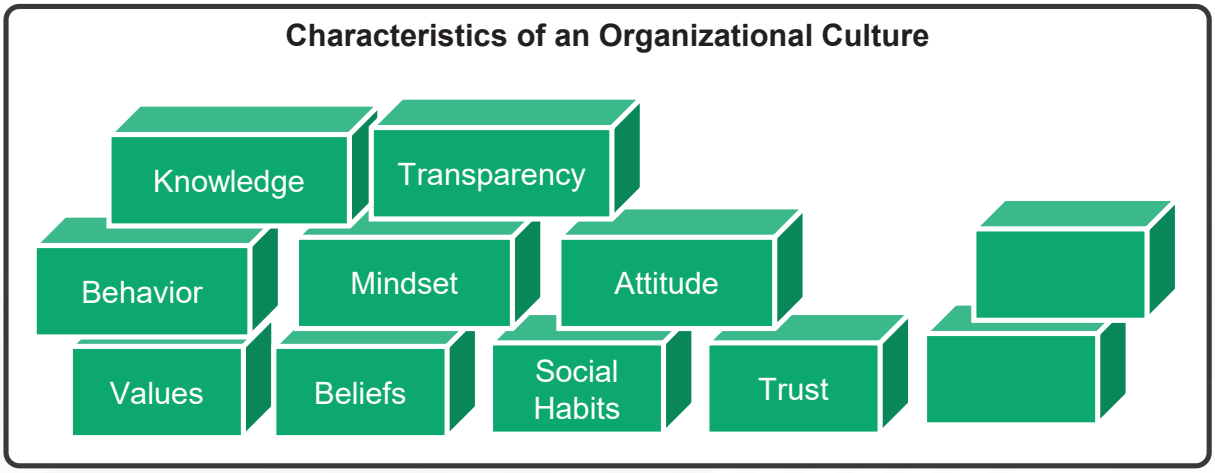


CAMS model was popularised by Gene Kim, the founder of the Phoenix project

CAMS is an acronym for Culture, Automation, Measurement and Sharing. These four elements are the foundational elements of DevOps and the role of each element is explained in detail in the upcoming slides.

2. Culture

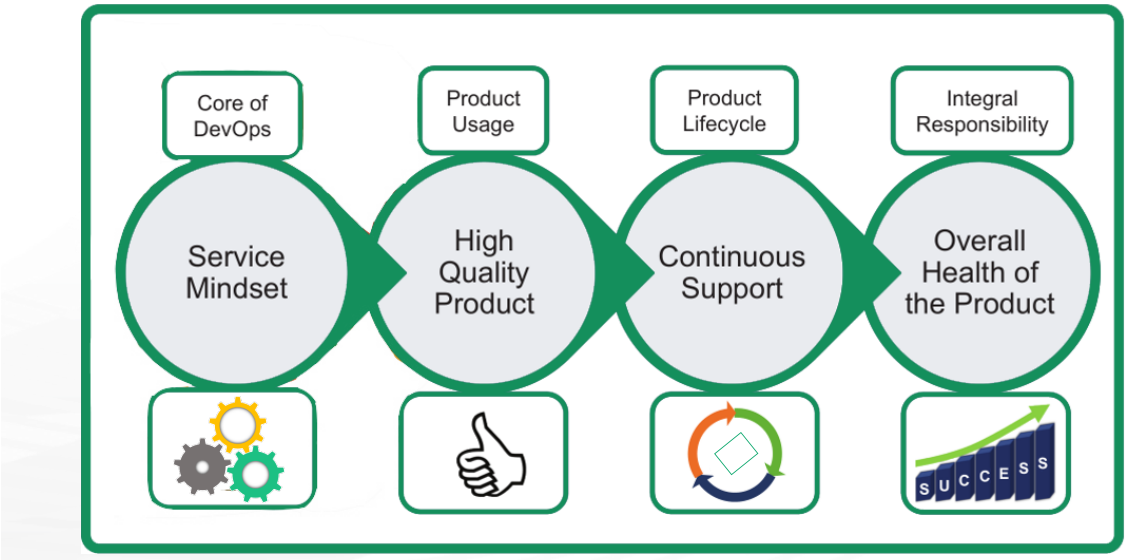
The following values form the crux of an organizational culture.



The culture of an organisation is its core values, which influence the actions and the behaviour of every team member. Right from a Janitor to a senior executive, everyone in the organization needs to work towards a common goal of high performance.

2.1. DevOps Culture - Service Oriented Culture

DevOps culture emphasizes on offering high quality service.



Adapting to a DevOps culture means transforming into an organization that focuses on delivering a high-quality service. Service not only means making the product available to the customer, but also it is to ensure that the product is in the best of its shape and it constantly meets the requirements of the customers. To offer better service, teams should have the knowledge of how the product will be received and used by its customers. This will enable the teams to offer continued support service even after the product has reached the customers.

A DevOps organization's primary motive is to ensure the performance of the product. The performance is tracked throughout its lifecycle to ensure that it continues to offer value to its customers. This is critical because the DevOps team in an organization is responsible for the different business processes, it supports and consistently works without any problems throughout the product life-cycle.

Service mindset is a critical aspect of DevOps culture, since the products developed with service mindset will meet the following conditions:

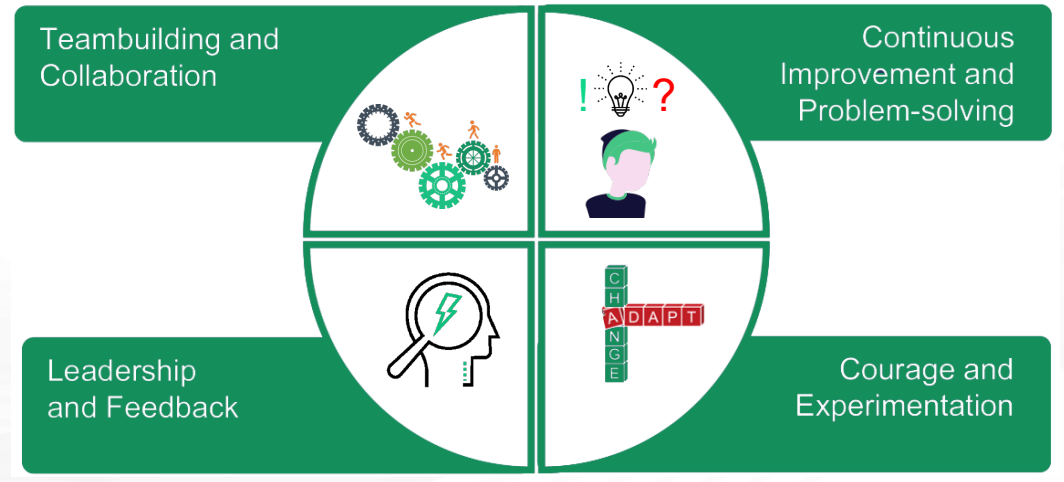
- Quality is given utmost importance right from the source.
- Early discovery of errors, especially right at the activity where these are created.
- Solving problems is easier and cheaper.
- Happy customers.

Establishing a DevOps culture - DevOps teams are driven by certain aspects in order to focus on delivering a high-quality service to customers. They are an atmosphere that promotes continuous learning, autonomy to experiment, a product thinking mindset, an engineering culture & focus on quality.

The Organizations should focus on creating an environment that propagates the above aspects.

2.2. Cultural aspects of DevOps

DevOps culture is a combination of following aspects



The primary differentiator of DevOps culture is, building a collaborative environment that delivers products or services at a faster pace and better quality than a traditional system. The people, processes and tools facilitate in creating a culture that works towards delivering the customer value.

Team-building and collaboration:

Much like any team-sport, the success of an organization is born out of the team effort. Every player has a key role to play where open communication and collaboration is critical. A transparent, decentralized organization where smooth information dissipation is critical to the success of an organization.

Continuous Improvement:

Delivering high-quality service to its customers is the primary goal for DevOps implementation. The teams need to constantly look for new ways to innovate and improve to ensure they avoid any inefficiencies or delays that occur during the service-delivery. The first step towards this is to ensure that the teams implement processes that identify lag or areas for the improvement. This could be in terms of cost of production, the quality of a product, the time taken to solve the issues etc.

Experimentation:

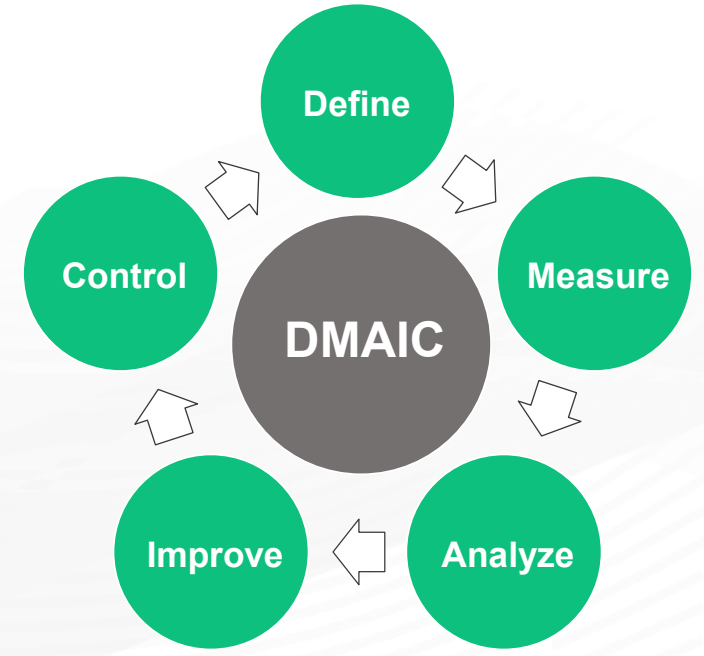
Implementation of DevOps for an organization is a transition into an open, collaborative environment, free from its traditional systems. Building an environment that allows room for failure is important for organizations. This allows them to try new methods, processes, tools to see which works best for them. By constant trial and error method, the organizations learn and find the best methods that deliver high- quality service and product.

Effective leadership & feedback systems:

DevOps is a standalone process that bring changes in the organization as soon as its implementation. It is a culture that needs to imbibed by every member of the organization. Each individual needs to take the ownership of their deliverables and responsibilities to ensure that the organization progresses towards to DevOps organization. Excellent feedback systems are critical since they help create awareness about pitfalls that occur and take immediate action. A good feedback is important for continuous improvement

2.3. Continuous Improvement and Problem Solving

The two methods used are Kaizen method & DMAIC method



Kaizen mindset is primarily about identifying the root problems, analyzing and solving them and sharing the knowledge gained.

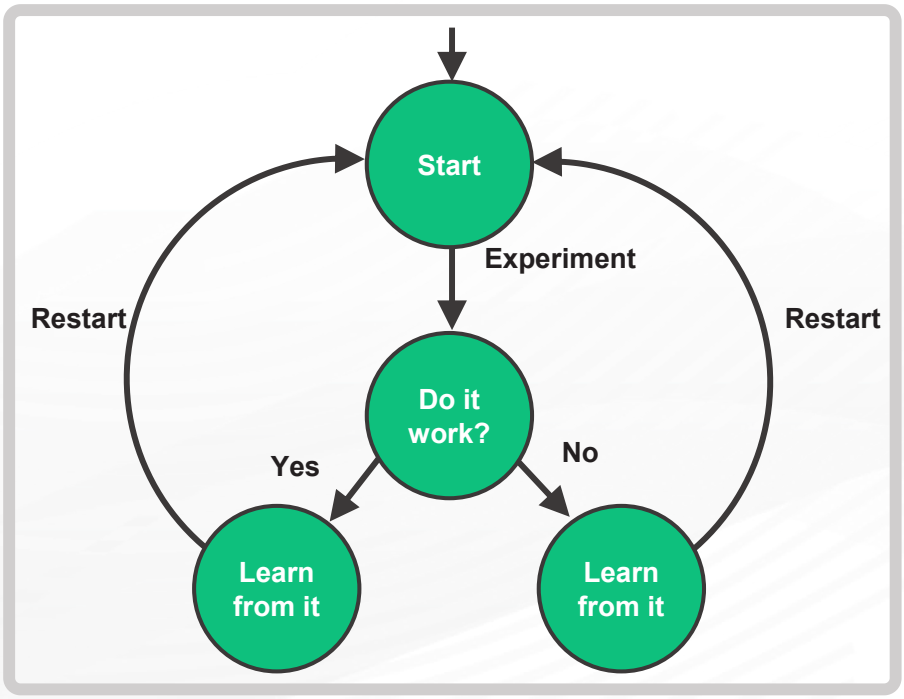
- All the organizations encounter different types of problems with regard to teams, processes, service etc.
- It starts with identifying the problems and coming up with strategies to fix these identified issues.
- Identify the root cause of the problem, solve them and identify a long-term strategy to completely mitigate the issue and ensure it is not repeated.
- The process of problem-solving and the knowledge gained is solved internally within the organization for the others to see and hear.

DMAIC Method

- You cannot resolve what you cannot define. Define the problem first and ensure that all parties agree on the same.
- Review internally, collate data regarding the problem and validate the information gathered.
- Analyze the collated data and test the issue.
- Come up with strategies to fix the issues and make amendments to improve the process.
- Create control systems and share the knowledge gathered in the process.

2.4. Courage, Experimentation and Learning

Experimentation and learning form the crux of innovation.

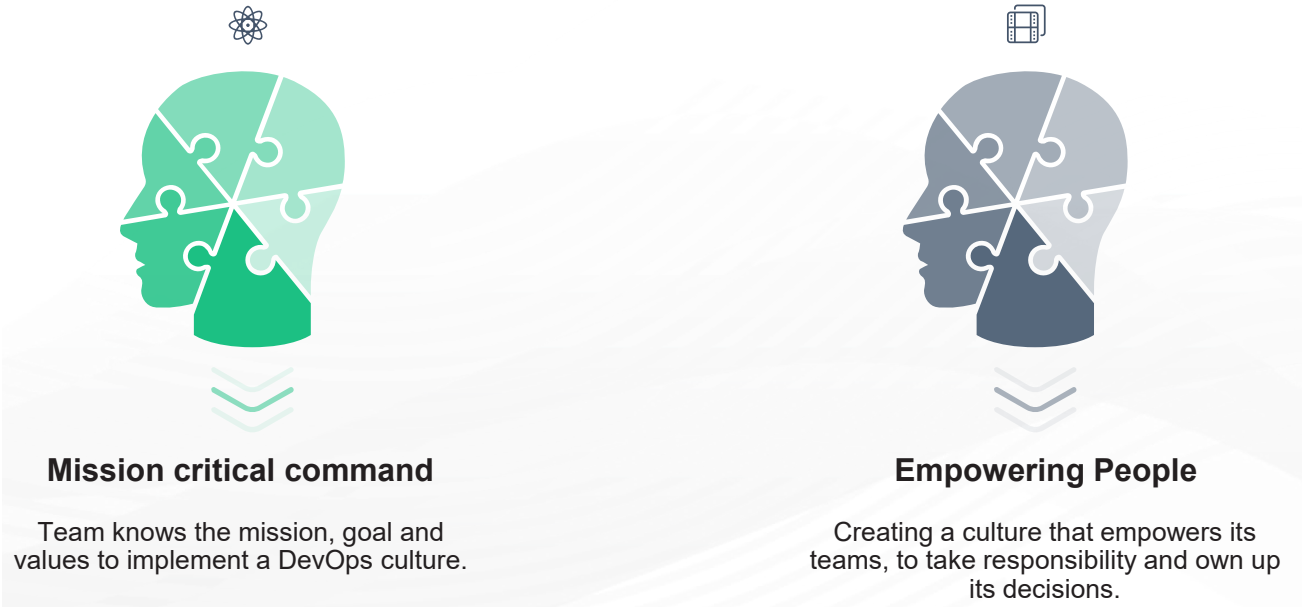


Experimentation is not merely trial and error. Experimentation involves adopting best practices to find out ways to optimize the existing processes of the organization, which eventually delivers value to the customers. It is about identifying a problem and building a hypothesis and testing it in a controlled environment. Experimenting in a live environment can create complications and that's why it is important to take calculated risks and resolve issues. All hypothesis must be tested well in advance once it is tested and stamped to be stable.

As an organization, having an environment that is courageous and knowing the full implications of failed experiments is important for DevOps implementation. A DevOps team should be courageous enough to carry out the experimentation. Because, the experiment may yield the desired result or fail completely. Courage gives the ability to fail fast and recover fast from failures. Experimentation is the best way to identify what will work and what won't. Ultimately, this ensures quality at source.

2.5. Leadership & Feedback system

The leadership has to be nurturing and encouraging enough to facilitate ownership among teams.



Mission critical command is to lead in an environment that is highly unpredictable and uncertain. Operating in such an environment requires extreme discipline, ability to make decisions quickly, work independently and in a union. This type of commanding and leadership is most critical for a DevOps environment where there is a lot of uncertainty within the operating environment.

An effective leadership will help overcome the following roadblocks on the way to effective collaboration:

- Lack of trust
- Fear of conflict
- Avoidance of accountability

- Lack of commitment
- Inattention to rules

Many conflicts arise within an organization due to various issues, such as lack of trust among team members, poor commitment, to push away responsibility to someone else and none of them taking responsibility etc. These issues can be mitigated only through constantly striving to create an environment and culture that nurtures trust. Team members need to be empowered through giving them responsibility and ensuring that failure is tolerated and supported.

Feedback system:

Feedback is one of the key tools that leaders use and stimulate. Giving and receiving feedback is the basis for all improvement and development of teamwork. It is vital for both leaders and team members to learn and practice giving and receiving feedback in a respectful manner.

The steps involved in giving and receiving feedback are as follows.

To give feedback:

- Describe the concrete observations.
- Explain what it does to you
- Wait and listen to clarifying questions
- Give concrete suggestions OR recognition/incentive

To receive feedback:

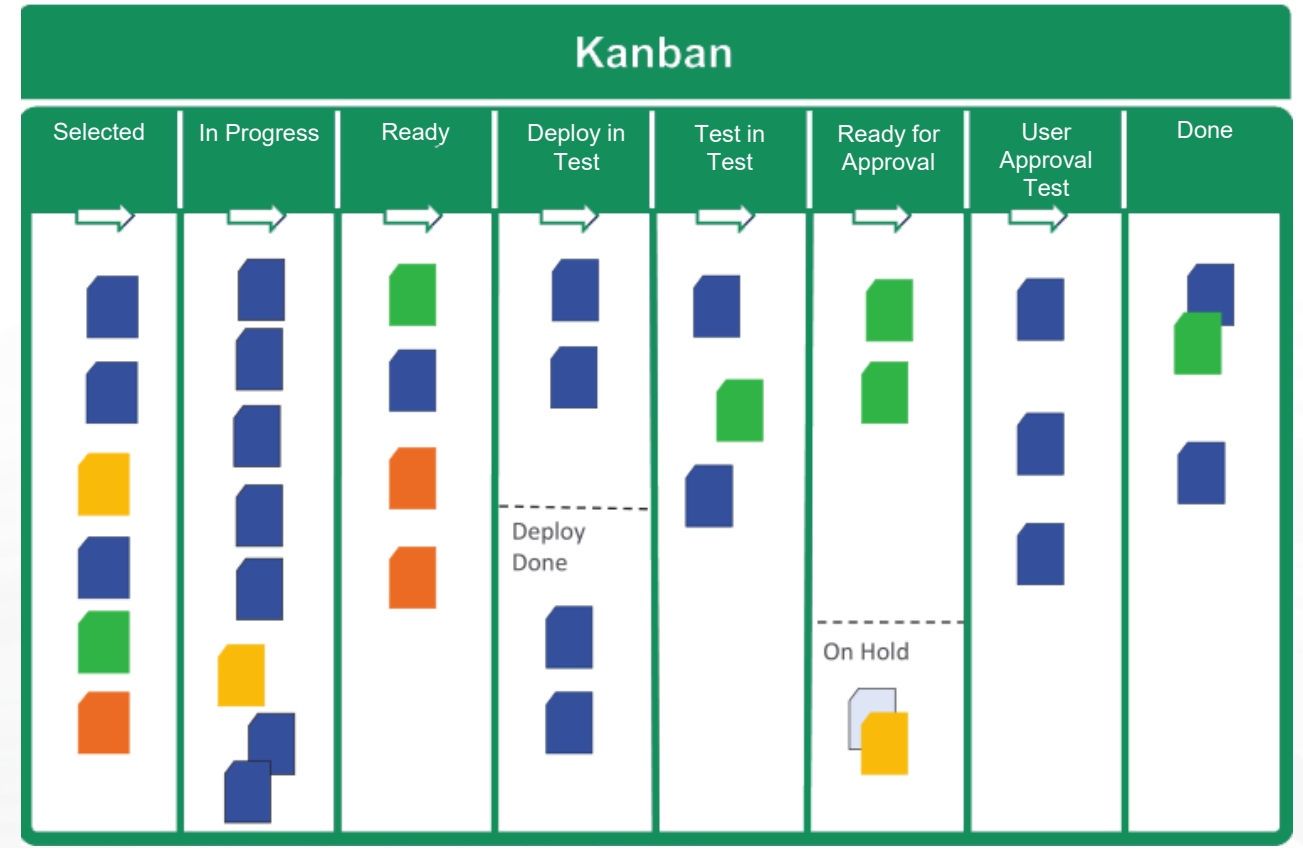
- Listen without interruption
- Avoid discussions or excuses
- Check if there is clear understanding
- Recognize other person’s position
- Thank him/her
- Determine whether the feedback is applied

An effective feedback helps foster an open atmosphere in which experimentation, team building, and improvement become part of the culture.

2.6. Team building and collaboration

Team building and collaboration form the important aspects of building a healthy DevOps organization.

Visual Tools are used by the teams to manage and collaborate with the team.



As per Katzenbach & Smith, “A team is a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually responsible.”

People joining a DevOps team often come from different technical teams. Therefore, they do not really fit the definition of a team. These technical teams have people with similar skills. However, they do not always have the same purpose or goals. They might be working together with other people (outside their group) to achieve the enterprise’s goals. Therefore, when transitioning from a technical team to a DevOps team, it is essential to give proper attention to team building.

Collaboration is one of the important success factors for effective team work. Collaboration means working together to achieve a common goal. If not properly implemented, collaboration will bring its own negative effects.

Team building is not a one-time activity but an effort that has to be taken as part of an organizational building. It is critical for creating a collaborative culture that facilitates trust and a healthy work environment. In order to facilitate team building across the organization certain tools are recommended. Mostly visually managed tools are very helpful in such cases.

Visual management helps to:

- Identify work and impediments
- Communicate important information
- Show how to perform a task
- Show planning and priorities

For example, visual boards are used to track the activity of each team member and have a live status of what is happening in an organization, at any given point in time. There are different types of the Boards such as a Day board, a week board and an improvement board.

The Day board is used to manage the progress of work on a daily basis. The management can be done using the simple “To Do-Doing-Done” setup or a more complicated approach.

The Week board is used to ensure the work is planned and agreed. It is also a place where a team should discuss their Key Performance Indicators (KPIs). Such a discussion is preferred on a daily basis. However, if it is impossible, the discussion should happen at least once a week. The board also serves as a repository for comments from customers and employees.

The Improvement board is where all the problems or improvements initiatives are collected that need to be solved or carried out. It is the place where the team members should communicate, and share solutions and learnings. It also shows the progress on problem-solving.

One of the tools for visual management is Kanban. Kanban is a signaling system that identifies when a new work can enter the system. It helps establish single-piece-flow to ensure flow of value and keep the team free from getting overloaded with work.

What did you Grasp?

1. Which of the following is not a cultural aspect of DevOps?

   A) Team-building and collaboration
   B) Continuous Improvement
   C) Experimentation
   D) Continuous Delivery

2. Which type of mindset is core of DevOps culture?

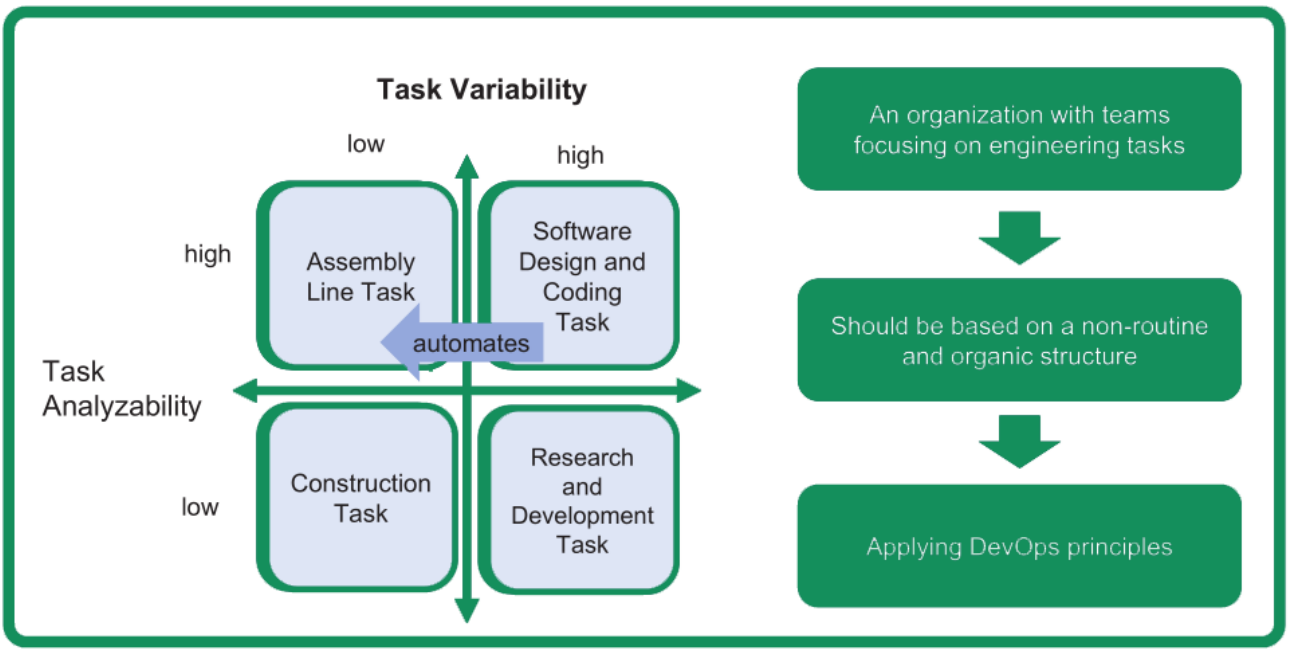
   A) Service mindset
   B) People mindset
   C) Skill mindset
   D) Automation mindset

3. Which of the following techniques is best suited for continuous improvement?

   A) Kanban
   B) Kaizen
   C) Scrum
   D) Automation

3. CAMS - Automation

Technology advancements are leading to automation of routine tasks.



Job surveys conducted across the sectors have started showing the shift towards automating routine jobs and how today there is a heavy need for skills to automate. Today, automation is applicable to different industry sectors such as manufacturing, healthcare, transportation and IT.

In a software delivery lifecycle, many of the tasks are repeated and become routine. On handing over the process to the operation team as well, we identify many tasks that are routine and redundant. Testing, routine checks can all be automated to improve the efficiency of the entire delivery process.

Barring non-routine tasks such as research, all other routine jobs such as traditional, standard processes which have little or no variation can be automated. When processes are automated, it becomes a standard and the inefficiencies are removed from the process. This also avoids any wastages that may have occurred and reduces the overall cost of software delivery.

The figure above, shows the task classification quadrant by Charles Perrow. The quadrant is based on two dimensions: Tasks Analyzability and Task Variability. Their distinction:

- Task Analyzability is defined as the extent to which, when an exception is encountered, there are known analytical methods for dealing with it.
- Task Variability is defined by the number of exceptions to standard procedures encouraged in the application of a given technology.

Many tasks to deliver software has a relatively low task variability and a high task analyzability or the software delivery process can be transformed using tasks with low variability. A value stream analysis can help to determine where processes can be improved and help to determine to identify tasks with high analyzability and low variability.

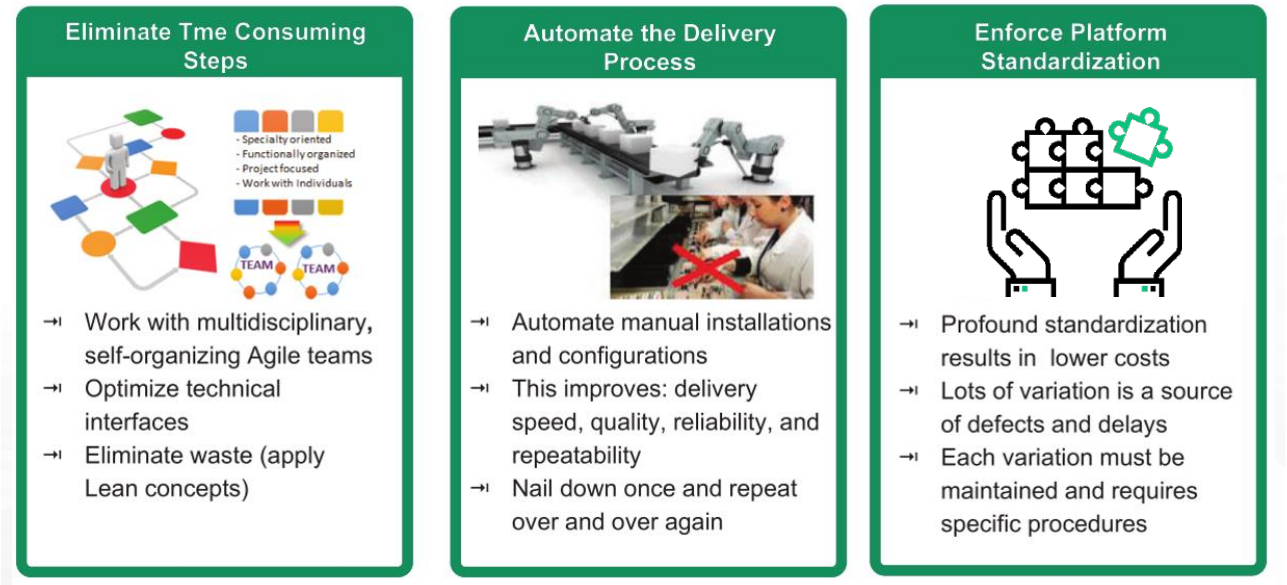
First, the routine tasks need to be identified and then automated. Some routine tasks are scripted test execution, deployment execution and code compilation. Automation helps delivery teams to focus on non-routine tasks that have a high level of variability.

Automation of routine tasks will result in a shift, where the Software Delivery team members will mainly execute tasks with high variability and high analyzability. In other words, engineering tasks. This requires an organization to be based on an organic structure, with largely autonomous, multidisciplinary, self-organized, engineering teams. Organizations need to apply DevOps principles in order to bring value to the table.

Automated processes are repeatable. Cost of process execution is minimized. Joan Woodward defined technical complexity as the extent to which a production process can be programmed so that it can be controlled and made predictable. In other words, automation results in predictable and standardized outcomes. These concepts can be applied to software delivery processes.

3.1. Delivering high value - DevOps way

DevOps focuses on delivering customer value through high quality service delivery



Technology adoption and the new processes continue to lead to many different types of tasks from time to time. Review the entire software delivery lifecycle and the organizational processes that can help identify the inefficiencies and fix them using automation.

The priority of using Automation and DevOps is to deliver supreme business value to the customers, the organization serves. Any hindrance that affects this needs to be eliminated.

Development of technology has resulted in a more and more type of tasks which can now be automated with ease. Automation, combined with other DevOps principles ensure teams can focus on the delivery of value. This desired focus is first of the 14 principles listed in the Agile Manifesto: “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software”. Teams should focus on delivering business value, and everything which hinders that should be

eligible for elimination. Automation can be applied to many activities in the software delivery process. As a result, DevOps teams can focus more on the engineering tasks required to deliver valuable software. In addition, they can utilize a software delivery process which delivers new software features continuously using an optimized flow.

Modern software development tools embrace everything as code concepts, API's, DSL, and scripts like first class citizens. At the same time, in the traditional operations space, more and more components can be defined with code, like software defined networks. These trends combined make it possible to define more and more artifacts with code. Teams which strongly focus on engineering tasks and automation will handle most delivery artifacts as code. Not just the software they are writing, but also the software configuration, test specifications, documentation and other software delivery tasks.

Many of the manual processes that exist between development and operation teams and the end users are carried out smoothly by DevOps as it employs many lean methodologies. By applying lean methodologies, process deficiencies, communication delays, missing information, manual errors, etc. can be avoided. Lean methodologies rely on continuous improvement, which are achieved by implementing continuous practices. With the use of integration technologies, DevOps processes can be automated. Integration brings together the tools used by different teams like development, operations, engineering and QA teams. Integrations helps in achieving a sync among the teams, which results in faster delivery of better quality software.

To optimize the software delivery processes, Continuous Delivery practices strongly rely on automation.

3.2. Continuous Delivery

Faster	Cheaper	Better
Human intervention is less due to automation saving time	Manual errors are expensive	Automation helps detect errors early
Automated execution eliminates need for manual meta tasks	Automation leads to standardization	Automation leads to better feedback loop and eliminates waste
Human-to-machine interaction time is less	More repeatable than manual task execution.	Measurement-driven evaluation of software features.

Revisit the definition of continuous delivery and how Automation facilitates continuous delivery of service.

Automation facilitates continuous delivery of services or products. Continuous delivery helps to optimize software delivery eliminating any wastage. This has been achieved by automating the routine tasks and focusing on solving the problems through an engineering approach. This has further led to the faster delivery of products, reliability, stability and better product or service delivery.

According to Jez Humble, “Continuous Delivery is about putting the release schedule in the hands of the business, not in the hands of IT. Implementing Continuous Delivery means making sure your software is always production ready throughout its entire lifecycle – that any build could potentially be released to users at the touch of a button using a fully automated process in a matter of seconds or minutes.

This in turn relies on comprehensive automation of the build, test and deployment process, and excellent collaboration between everyone involved in delivery – developers, testers, DBAs, systems administrators, users, and the business. In the world of Continuous Delivery, developers aren’t done with a feature when they hand some code over to testers, or when the feature is “QA passed”. They are done when it is working in production. That means no more testing or deployment phases, even within a sprint (if you’re using Scrum).”

Continuous Delivery Automation:

Automation of the software delivery process is one of the core means to deliver faster, cheaper, and better delivery.

Faster due to:

- Automated task execution does not depend on the availability of humans, so the wait time is eliminated.
- Automated task execution requires no human-to-machine interaction time.
- Automated task execution reduces the need for (manual) meta tasks (like a review of an installation document).

Cheaper due to:

- Automated task execution is more reliable than manual task execution. Manual execution errors are expensive and may not be detected immediately.
- Automated task execution is more repeatable than manual task execution.
- Automated task execution requires no human effort (which is more expensive than machine time).
- Automated task execution requires standardization, based on minimal required variations.
- Every variation requires specific procedures and maintenance.

Better due to:

- Automation results in predictable, standardized, reliable, and repeatable outcomes.
- Automation eliminates manual task execution errors.
- Automation (with test automation) results in faster feedback loops, defects are detected as early as possible.
- Automation enables measurement driven evaluation of software features delivered.

What did you Grasp?

1. According to DevOps principles, it is advisable to automate highly variable and highly analyzable tasks.
   A) True
   B) False

4. CAMS - Measurement

Measurement is fundamental for continuously offering value and improvement. Tracking important metrics and offering feedback helps improve the process. Three ways feedback model

Measurement and monitoring have to be integrated into your day to day processes as part of DevOps implementation.

The First Way focuses the entirety of the system and ensuring that none of the defects at a local level affect the overall flow, performance or stability. This method ensure there is one single flow of feedback.

The Second way is about creating bidirectional feedback loops of moving from left to right and right to left. For this to happen, the feedback loops should be shorter in order to receive and send feedback sooner. This method helps in having a deeper understanding of the process.

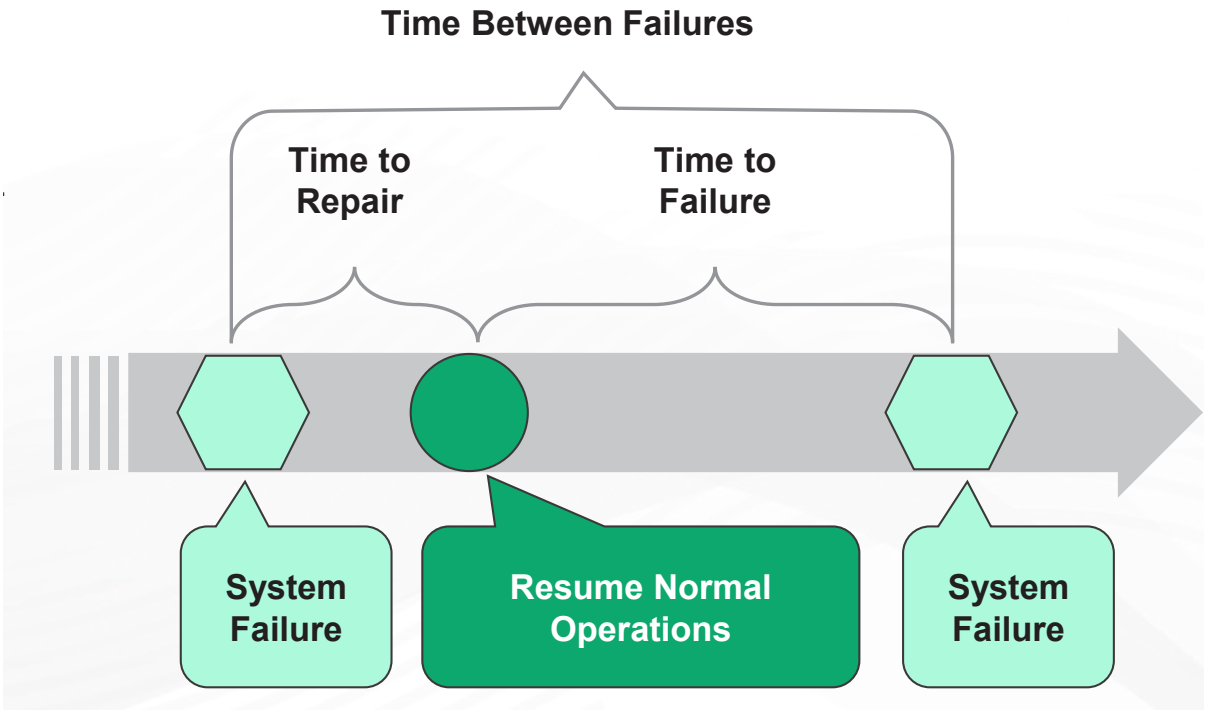
The Third way of measurement is to create an environment of continuous learning, feedback and room for failure. In this method, we build an MVP, receive feedback and makes changes accordingly. It rewards teams for risk taking despite the failure.

4.1. Metrics used for tracking

It is important to choose the right metrics to ensure there is no inefficiency.

The different performance metrics used are:

- Measurement of throughput using deployment frequency and response time.
- Measurement of stability using MTTR .



An organization's performance has to be tracked from time to time to ensure there is no slack, inefficiencies and drop in output. Just like how certain parameters indicate good health in a human being, certain performance metrics have to be tracked to ensure sound health of an organization.

Both throughput and stability are critical metrics that need to be tracked to ensure that software delivery is efficient.

The two important factors that need to be tracked during measurement of throughput is deployment frequency and response time. Deployment frequency is tracked by ensuring there is continuous delivery and employing version control practices. Using version control where all IT artefacts can be traced back to save time and cost for the organization and in addressing issues. Version control also helps in reducing the lead time where routine tasks are automated.

MTTR- Mean time to recovery

As DevOps is implemented more and more in organizations, MTTR should get lower and lower. Systems should function stable in order for MTTR to be lower. The time taken to recovery will have

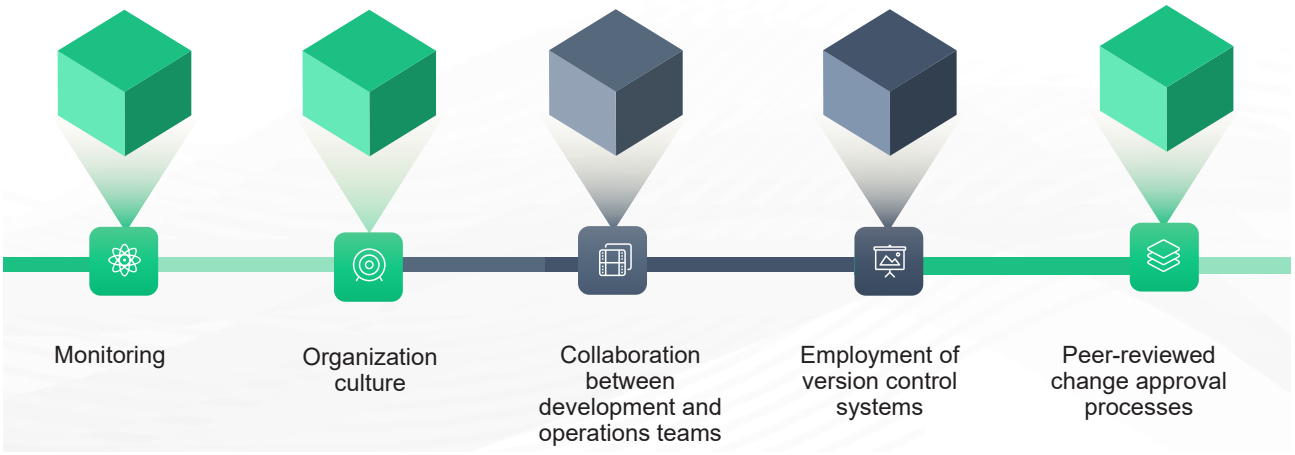
direct impact on the customer retention. If MTTR from a failure is negligible, the systems will be up and running even before the customers notice it. On the other hand, unstable products will have more MTTR, which causes disinterest among the customers. So teams have to focus on recovery aspect more than failure. With automated tests and deployment environments, the time taken to fix an issue is usually a matter of minutes or seconds. MTTR is a major driver for the following:

- Both the teams and product will be resilient and fault-tolerant.
- Faster and better feedback mechanisms that result in faster delivery of error-free products.
- Less dependencies between teams and systems, hence failures are handled locally.
- Usage of better monitoring systems.
- Fixes are done as quickly and easily as a rollback.

Again version control is one of the ways to control errors since it helps the application to be switched back to the previously used working mode. This saves a lot of time while fixing bugs or issues. Monitoring your IT systems also help towards detecting failures and to ensure a healthy working application. Different tools are used for the same

4.2. Performance Predictors

In addition to the performance metrics, there are few predictors that ensure your organization's IT performance is on track



For an organization to be successful and most importantly consistent, the management has to ensure that every department, function, member have to work in union. There are many predictors that give us an insight into health of an organization. Especially for DevOps we need to ensure that there is a healthy organizational culture that promotes collaboration and team-work. Puppet Labs in its State of DevOps report, 2014, has dealt in detail with IT performance predictors. According to the report, the top 5 predictors are:

Peer-reviewed change approval process:

The 2014 State-of-DevOps Report concludes that IT performance throughput is lower when approval of changes is done by outside members rather than the team that make the changes (for example the Change Approval Board (CAB). When the team, responsible for the changes, is made accountable for the quality of the changes through peer review, the performance throughput increases.

Version control for all production artifacts:

Version control is a correlating practice for three of the performance metrics (deployment frequency, lead time for changes, MTTR), which probably means that it is wise to invest in if you want to increase your IT performance. Good version control provides a Single Source of Truth (SSOT). SSOT refers to the concept where certain data has only one official source to be used by data consumers for the true current version of that data. Good version control means when incidents occur, it's easier to look for the root causes and roll-back to the last good state, reducing the time to recover.

Version control also promotes learning and sharing knowledge between teams. Especially, when version control isn't limited to application code, but also includes system and application configurations and deployment script.

Proactive monitoring:

Organizations that work with teams which practice extensive and proactive monitoring of their own products and services, are able to solve problems faster and accelerate their continuous improvement. When logging events, such as failures, are primarily the responsibility of dedicated monitoring teams (for example, an Operations Bridge), IT performance suffers.

High-trust organizational culture:

DevOps is all about culture, and the report shows that a high-trust culture predicts better performance and that bureaucratic and fear-based cultures do the opposite.

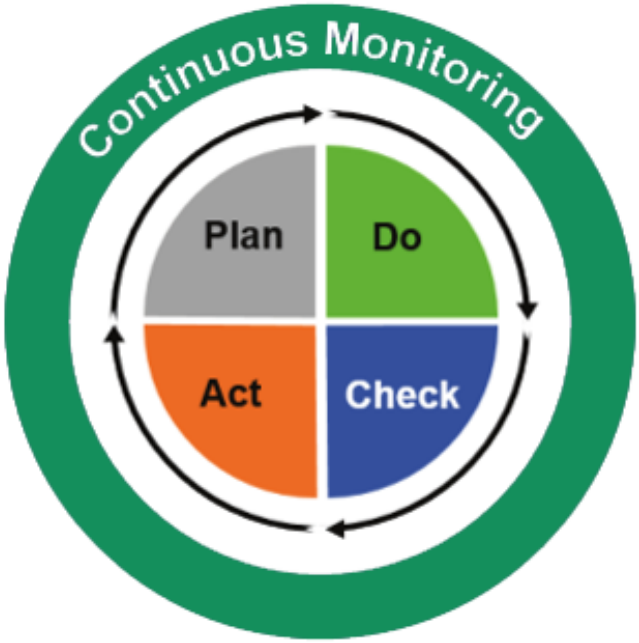
Win-win relationship between Dev and Ops:

Breaking down the barriers in siloed organizations and building strong teams based on DevOps principles benefit IT performance greatly.

Good monitoring systems need to be designed to ensure performance is on track. We'll now learn about continuous monitoring.

4.3. Continuous Monitoring

Continuous monitoring is the practice of tracking compliance and risk issues that may arise in an organization from time to time.



In traditional organizations, where teams worked in silos, monitoring was managed by a separate team within operations to track and monitor all the key parameters. However, this might not be applicable to organizations that have adopted DevOps. In a DevOps environment, there is a need for faster systems that have integrated DevOps and QA teams.

For any organization to have good monitoring processes, it needs a clear strategy towards the same. First, you identify all the parties whose data needs to be monitored, create specification, necessary tools to collate data logs and perform data analytics to implement a strong monitoring system.

Monitoring is not just the infrastructure, we also monitor the application, the platform, log aggregation and also monitoring the overall business health.

4.3.1. Monitoring for DevOps

An organization that has adopted DevOps, not only monitors the infrastructure like traditional organizations but also tracks the multiple facets of the business to stay in tune with the customer's performance and needs.

A well-designed Monitoring set-up will raise a continuous improvement cycle which is the very purpose of DevOps. DevOps looks at continuously delivering high-quality service to the client and the monitoring helps them to track their pitfalls from time to time resulting in a continuous improvement. The Tools are used for monitoring the Data and any issues detected are directly communicated to the respective product to address.

Since DevOps is about an overhaul of the entire organization, we look for multiple ways to track the customer data. Social media is often one of the easiest ways to learn and understand customer behaviour.

Monitoring dashboards are created for being a single view of all issues. DevOps dashboards are an ideal way to monitor different kinds of administrational tasks. These dashboards keep continuous and up-to-date track of critical parameters like uptime, load time, API calls, CPU process threads, memory usage, etc. As the dashboard keeps updating, the teams can find out down times, if any, even before the customers become aware of it. Some of the types of DevOps dashboards are:

- Business performance dashboard - Gives a picture related to customers. The number of customers, average usage of product, revenue, active business hours, etc.
- End-user and app performance dashboard - Gives a picture on the performance of the application. Keeps track of the performance changes every time there is a change in code or any fix is done.
- Change events dashboard - Monitoring the event changes that happen on the underlying infrastructure.
- Alerts dashboard - An important dashboard that tracks alerts. Alerts are raised whenever there is an issue. Alerting systems need to be intelligent enough to indicate really abnormal events, that are meaningful to the users.

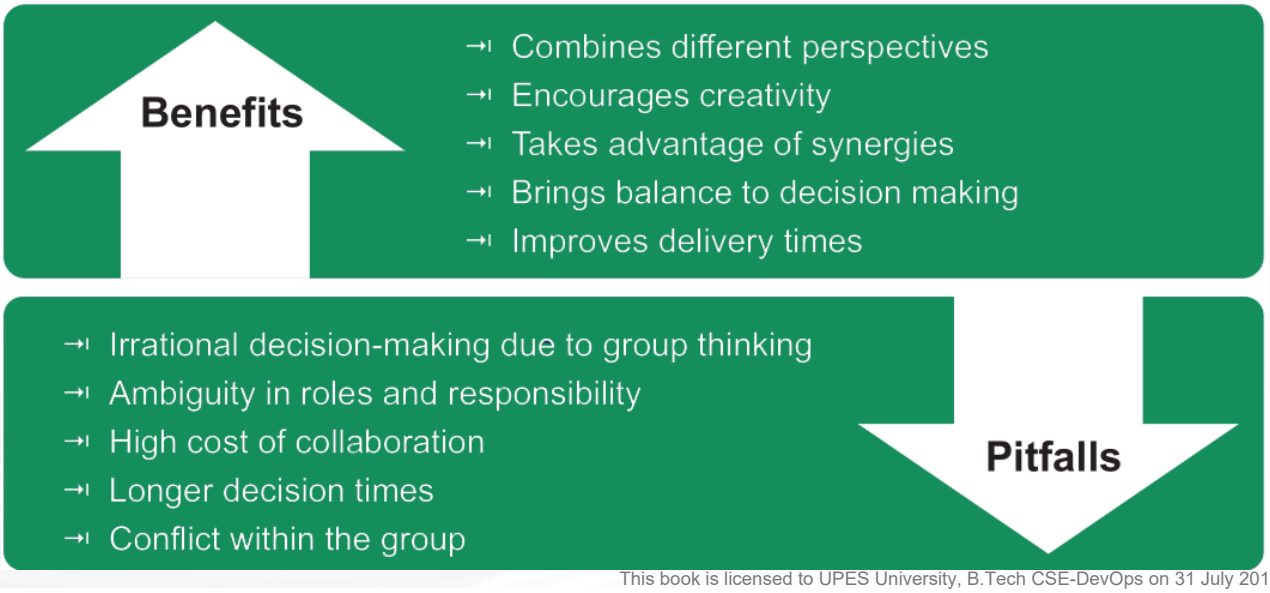
What did you Grasp?

1. Which of the following is NOT a way of continuous monitoring?
   - **A) Alerting incidents to the teams directly**
   - **B) Incorporating service management data**
   - **C) Multiple points of truth**
   - **D) Track social media**

2. Which of the following is true about performance predictors?
   - **A) Version control systems is not mandatory for tracking performance**
   - **B) Need of peer reviews and change approval processes**
   - **C) Development and operations teams need to work independently of each other**
   - **D) Continuous monitoring will result in frequent changes and will not be helpful deriving a good result**

3. Which of the following is important metric for efficiency?
   - **A) Throughput**
   - **B) Automation**
   - **C) Stability**
   - **D) Both a and c**

5. CAMS - Sharing



Sharing is one of the most important aspects of the CAMS model. DevOps by definition is the collaboration of development and operations and sharing is fundamental to this. Implementing DevOps involves a constant interaction and most importantly knowledge sharing. Every team within an organization has encountered failures with regard to the people, processes, tools, projects etc. This knowledge has to be shared within an organization among the team members to ensure successful implementation of DevOps.

A blame-free environment is required for effective sharing. People do make mistakes and they tend to hide their mistakes, because of the fear of getting blamed or punished. Blameless culture is the culture of sharing.

5.1. Blamelessness

Collaboration, transparency and cross-functional learning form the crux of DevOps culture.

DevOps culture is about team-building, collaboration and innovation. The Organizations need to create a culture of taking accountability and an environment where team effort is the norm. In DevOps, you create an environment where people can experiment and fail. In such a scenario, a blame game should be avoided for the benefit of the team and organization. This practice is possible only through creating a culture that has room for failure.

Matt Stratton, Senior Solutions Architect at Chef, had a customer who developed what's known as the "blameless post-mortem." According to Matt, "The thing is, people are always going to make mistakes. You can't fix that. People are human. So stop trying to fix them. Instead, we fix systems. That's what we mean when we say we have a blameless post-mortem,". The true advantage of blameless post-mortem is that there is room for each member to think about what went wrong. This way, the problem identification becomes the focal point, instead of blaming people for problems created.

What did you Grasp?

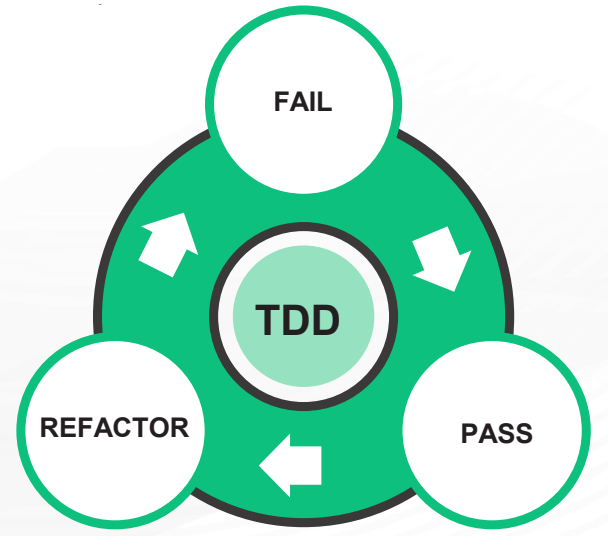
1. Which of the following is NOT a benefit of sharing?

   A) High cost of collaboration
   B) Delivery time improvement
   C) Improved creativity
   D) Longer decision times

Group Discussion

6. Test Driven Development

It is an approach to development in which the test cases are written first and followed by code just enough pass the test cases developed.



DevOps emphasizes automating as much as you can. Automation ensures continuous delivery. Achieving the CAMS formula is a means to achieve continuous delivery. Continuous delivery largely depends on testing, as testing helps in delivering bug-free, reliable software to the customers. Test-driven development is one of the ways of developing and delivering quality software that makes use of automation.

Test driven Development is an approach to development in which automated test cases are written first to test the desired improvement or a new function. Then minimal code is written to pass the test. Once the test gets passed, new tests are written when new functionalities are added. The code is then refactored to acceptable standards. Thus, TDD follows a very short, repeated development cycles.

The steps involved in TDD are:

- Write a test
- Run all tests and see if the new one fails
- Write minimal code
- Run tests
- Refactor code
- Repeat

Benefits of Test-Driven Development are modularized, flexible and extensible code, no unnecessary code which saves a lot of time, a minimal working code in shorter time and then test cases to be extended to cover the edge cases and greater confidence in the code.

6.1. TDD – Categories of Tests

Unit Tests	A smallest part of the code (Often a class or its method) is tested with substitutes such as mock objects or method stubs.
Integration Tests	The Individual modules of software are combined together and tested as a group.
Functional Tests	A quality assurance test that evaluates the overall functionality of the code and compares the actual results with expected results.
Regression Test	It ensures the new changes made to the code have not created any faults.
End-To-End Tests	It is an overall test that includes Unit tests, Functional tests and Integration tests etc

Different types of testings are used in each stage of development. Some of the methods of tests are unit tests, integration tests, functional tests, regression test and end to end tests.

Unit test cases are used for testing classes or methods while functional test is used for testing the functionality of the code and tested against the actual expectation.

What did you Grasp?

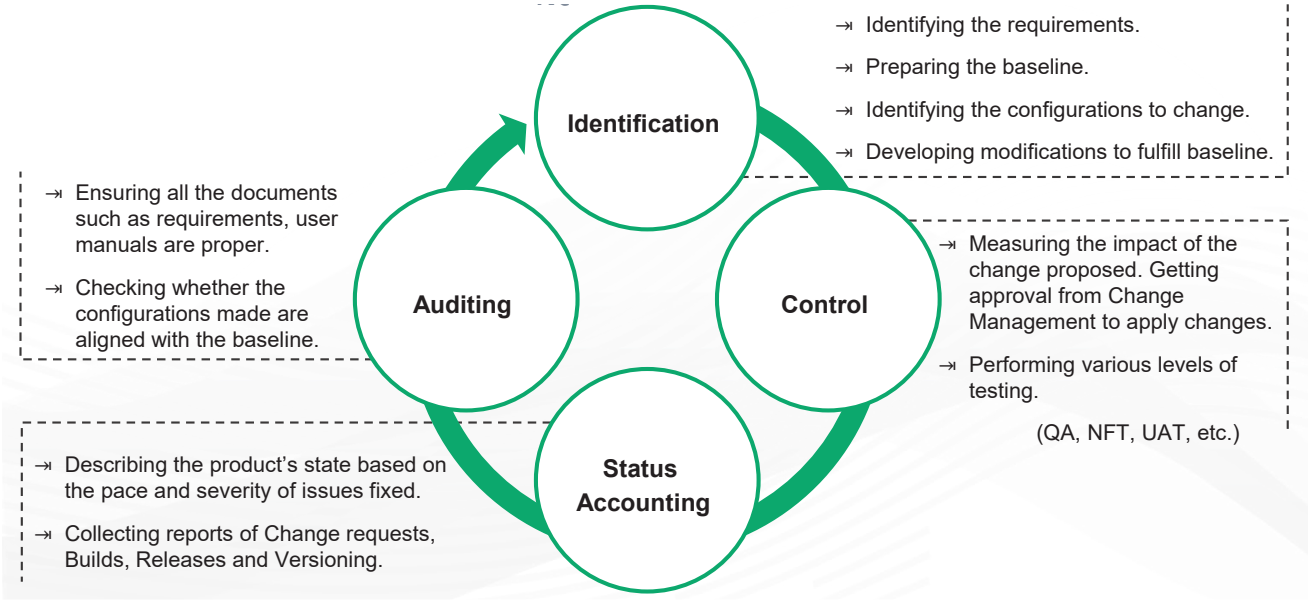
1. Which of the following does not belong to the testing category?

   A) Unit test
   B) Acceptance test
   C) Characteristics test
   D) Functional test

2. Which of the following is a valid statement about a functional test?

   A) Individual modules are collated and tested as a group.
   B) A quality assurance test that ensures the overall functionality of the software.
   C) A test that validates new additions and identifies faults.
   D) Checks whether the product meets customer's requirements.

7. Configuration Management



Configuration management is a control mechanism that is used for managing and controlling a product's performance and other aspects such as operational, functional and design attributes. It is used in managing the software lifecycle of a product.

A configuration item (CI) is the fundamental unit of any configuration management system, that can be a service component, infrastructure element, or any other other item. A CI can be a full-fledged

service, that is inclusive of hardware, software and documentation or it can be a single program or else a minor hardware. Management of CI is essential for the successful delivery of services. A CI can have:

- Classification
- Attributes
- Status value
- Relationships
- Owner

CIs differ in terms of complexity, size and type.

Configuration management tools are essential for maintaining large scale environments in a desired state. These tools aid in monitoring the environment continuously. These tools also ensure that the infrastructure is configured to correct specifications. These configuration tools are also helpful in eliminating version control issues.

The important features of configuration management tools are as follows:

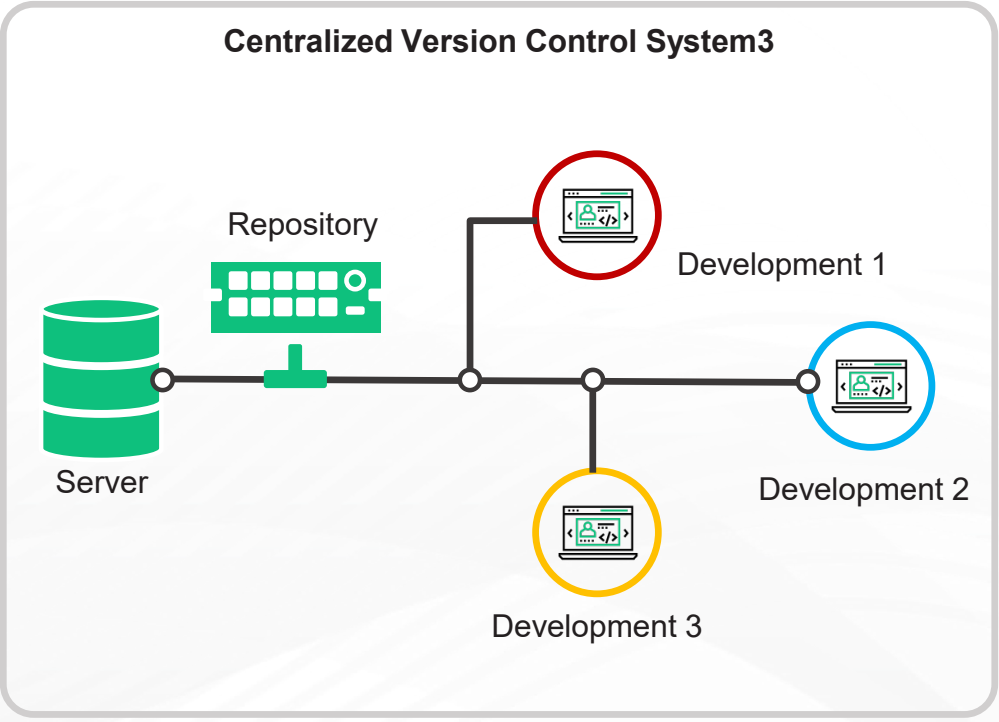
1. These tools keep the data about the versions and builds of the software.
2. For efficiently managing the builds and releases
3. Effective baselining, i.e., the configuration items that make up a release
4. To maintain the access controls

Some of the common configuration management tools are:

- Chef - Configuration management tool that helps in provisioning, configuring and managing an enterprise's servers.
- AWS Opsworks - Configuration management service that uses Chef to automate server configuration, deployment, and management across your Amazon compute.
- Puppet - An open-source configuration management tool that provides a standard way of delivering and operating software irrespective of the platform upon which it runs.
- Ansible - Open-source automation engine that automates software provisioning, configuration management, and application deployment.
- SaltStack - Infrastructure and application management package.

8. Source Code Management - Version Control

- Record to maintain the changes made to the source code - Who? What? When?
- All the changes to the source code, documentation, user manual, etc., should be tracked and versioned.
- Versioning allows the change management to easily rollback to a previous working version, in case of an unsuccessful change.
- Moreover, it enables the development and existence of multiple versions of the product that will allow the customers to take their own time to migrate.
- Git, Mercurial, VSTS are some of the most common source version control tools on the market.
- Semantic Versioning (Major . Minor . Patch) is adopted widely.



Source code management (SCM) is critical to keep a record of the changes made to the source files. These systems will be up-to-date with the information of 'who' changed 'what' and 'when' did the change happen. Every organization will have a source control system, that has a set of tools to manage it. There will be a centralized source code repository, and each and every developer should have a good knowledge about the source control system in order to efficiently build and release the software.

Version control is the way to manage the source code effectively and keep track of the changes made to the source code. It is a principle employed to track, document and monitor every piece of code written i.e. it used to manage changes that occur with every code commit. This method helps developers track the code, especially when multiple people are contributing at different points in time. This also helps roll back to the earlier version if there are any errors.

Git, Subversion and Mercurial are some of the most commonly used version controls tools used globally. Version control systems (VCS) are chosen based on code structure, sharing patterns, how the code is built, the size of the codebase and the ways developers work and share the code.

Different versions of the software should be named by a global standard called 'Semantic Versioning' (SemVer), which is a three-component naming system, which takes the format **Major . Minor . Patch**. Version increment is done to:

- MAJOR version when incompatible API changes are made to the software.
- MINOR version a functionality is added in a backwards-compatible manner.
- PATCH version when backwards-compatible bug fixes are made.

For example, open any application on your mobile play store. You will find the version of the application. So if an app is updated, a bug fix might have been done, or a new functionality might have been added or else API changes have been made.

8.1. Base Lines

Baselines are primarily used as milestones in product/software development that categorises and groups changes that will be delivered in a change request. The changes to be delivered becomes a baseline on thoroughly reviewing all the changes. A baseline is used to identify one particular configuration of a software. A baseline can be recreated at any point in time in future.

A baseline is any intermediate stage of the software that is recorded/saved and approve at certain points in time. It serves to provide a fixed reference point for change management.

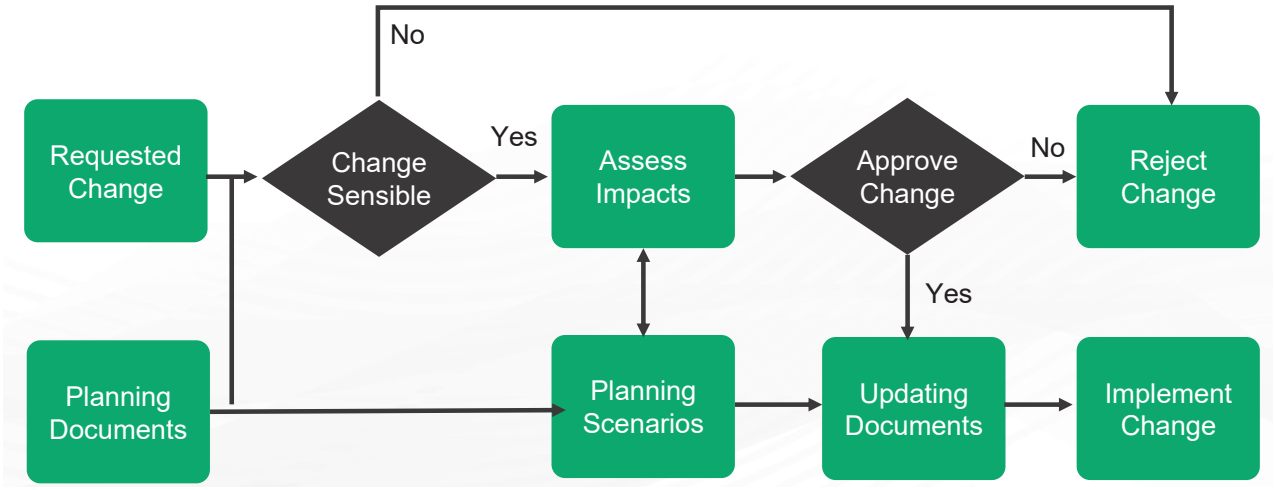
There are different types of baselines that are used as a milestone in product building. They are functional baseline, developmental baseline, allocated baseline and product baseline. On completing every change request, it has to be verified and allocated in the system accounting phase.

Branching:

One of the important features of version control systems is that it is possible to have changes isolated and kept on to a separate line of development, which is called 'Branch'. Branches are important while adding new features to the code, without disturbing the master or main line of development. If new functionality is added to the main branch, there is a possibility that it will get broken or a possible failure of previously existing functionality. Especially, when multiple developers are working on the same code, conflicts may arise. To avoid this, developers will commit their changes to the branch. Once all changes are committed and the newer version is tested and found to be stable, it will be merged with the main development branch.

Version control systems can mark particular revisions (e.g. a release version), so you can at any time recreate a certain build or environment. This process is known as tagging.

8.2. Change Control



The Change control is a process that involves a step by step process that follows the submission of a change request. This is primarily employed to minimise the changes in the product and the effective utilisation of the resources.

Change Control is part of Change Management. The Change Control is a sequence of record, assess, plan, build/test, implement and close.

Record: The change request is initiated by placing a formal request. This request is then categorized and sent for analysis. Based on the propensity of the request, it is thoroughly analyzed to ensure that it makes technical and business sense. Once the request is approved, a plan is made to implement the change. Once all the parameters are analyzed, the change is applied, built and implemented.

The product with implemented changes are sent for the client approval and once approved, the request comes to a closure.

What did you Grasp?

1. Approval from change management is done during which of the steps in configuration management?

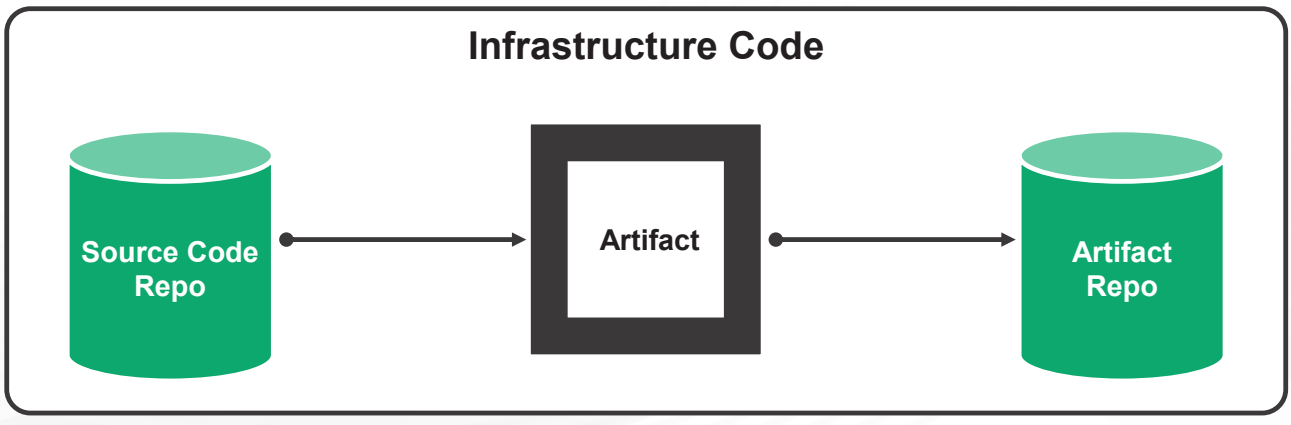
   A) Identification
   B) Control
   C) Status accounting
   D) Auditing

2. Which of the following is the CORRECT way of representing a version?

   A) Minor.Major.Patch
   B) Patch.Major.Minor
   C) Major.Minor.Patch
   D) None of the above

9. Infrastructure Automation

This is process of managing Datacenters through the computer readable files instead of configuring your hardware interface.

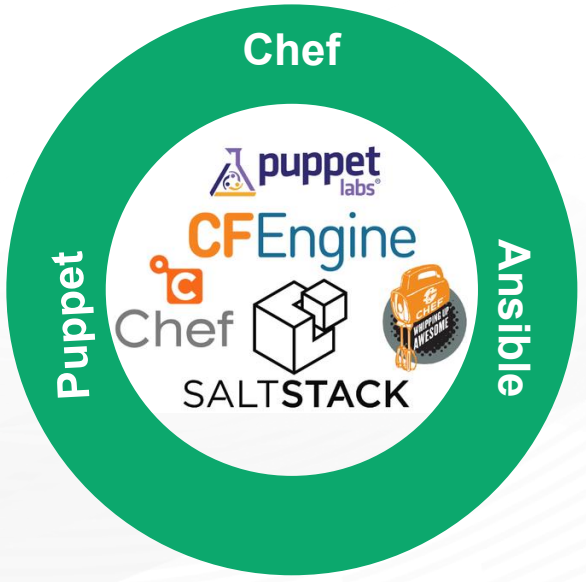


Automation is primarily done when organizations are planning to scale and cater to a larger base. Having this in mind, automation is done with dynamic infrastructure. It is done primarily provisioning and setting up environment, allowance for scaling resources dynamically, moving applications to new age cloud, start deploying application in a hybrid environment and support different environments.

Current conditions, applicable need to built to be configurable, immutable, independent and reuildable. For this, automating infrastructure, application and testing will help and create great value.

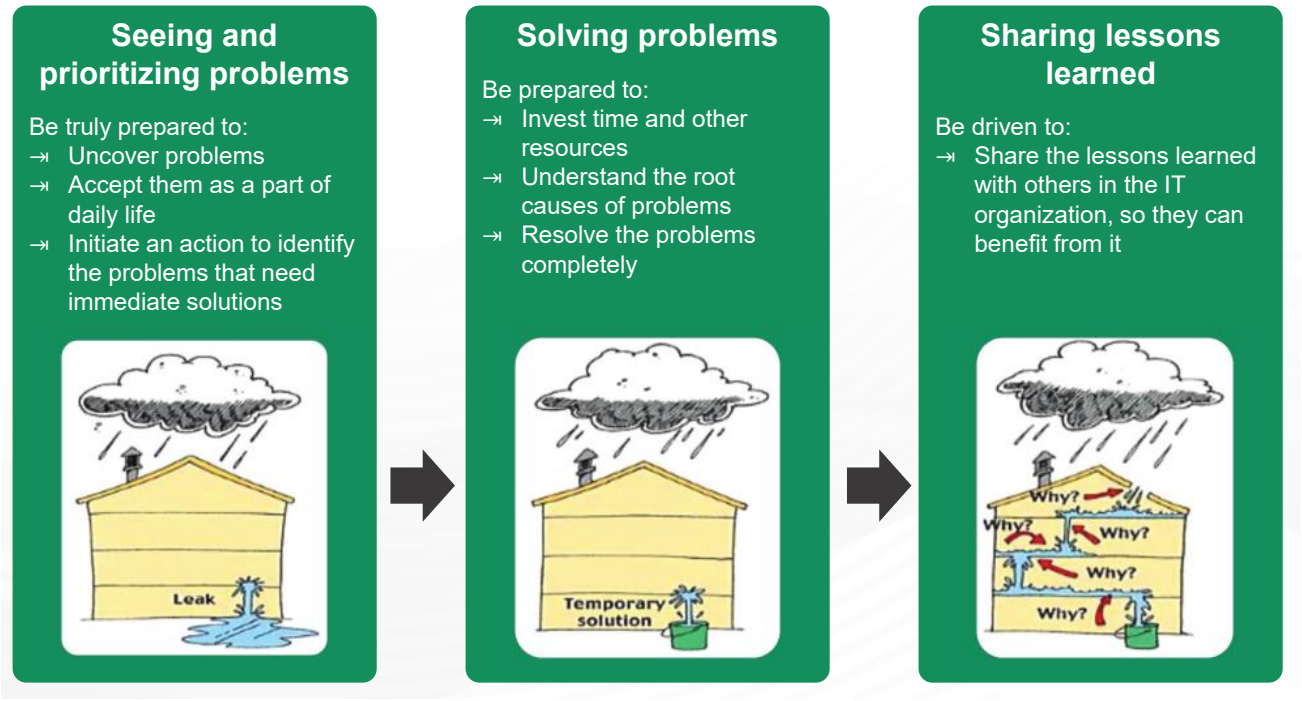
To enable faster deployments, infrastructure is treated as software, i.e., code. By treating infrastructure as code, the management processes become easier and the same tools and processes that are used for code management can be used for infrastructure management as well. Effectively, by considering infrastructure as code, you use code to manage configurations and automate provisioning of infrastructure in addition to deployments. These tools and processes include the ones that we learnt in this module: continuous integration, code review, version control and automated testing.Thus infrastructure changes are done in a fast and reliable manner. Example for these management tools include: Puppet, Ansible, etc.

9.1. Infrastructure Automation Tools



Infrastructure automation has gained significance as part of the DevOps movement and some of tools used are Chef, Puppet and Ansible.

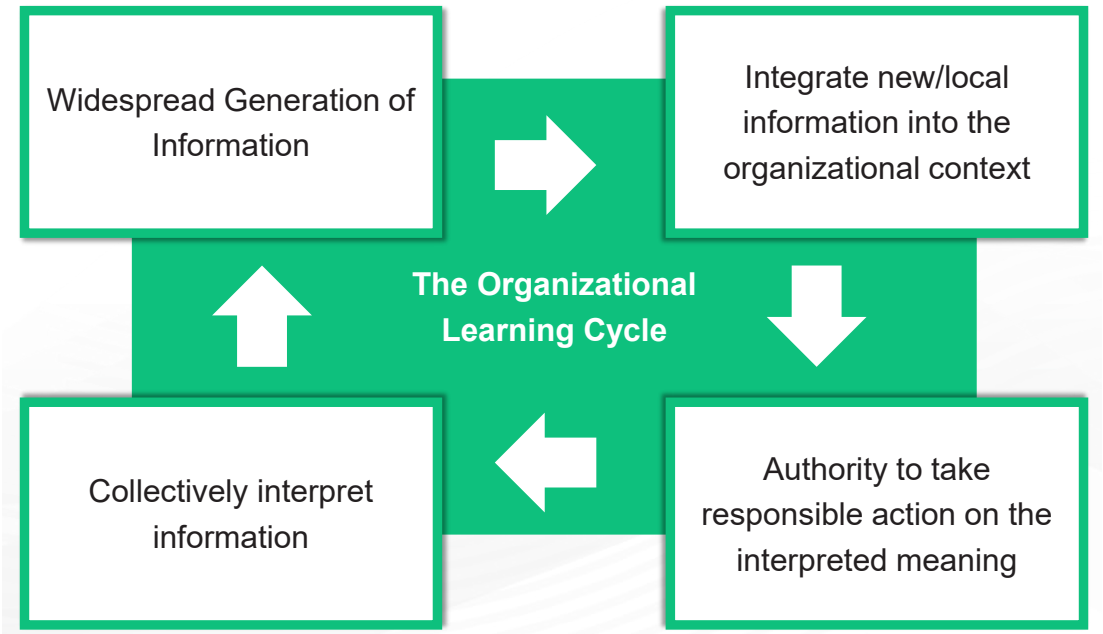
10. Root Cause Analysis



The Root-cause analysis is a very important practice that the organizations follow to mitigate issues and ensure that they are not repeated. The Root-cause analysis also ensures that the companies build long-term, sustainable mechanisms to tackle an issue.

- Organizations need to thoroughly analyze the landscape and identify existing problems.
- They also need to review each process completely and diagnose the sensitive areas where the problems can occur.
- Once the issues are identified, analyzing the root-cause of the issue is critical to finding a permanent, long-standing solution.

11. Organizational Learning



For any organization to grow and sustain it is imperative to innovate, experiment and learn. This is very true for a company that has implemented DevOps. As a culture, the company has to be open to learning and unlearning on a constant basis and this is the only way for organizations to stay ahead of the competition.

This can be done in the following ways:

- Creating a multidisciplinary team to facilitate shorter feedback circles and quicker learning.
- Hosting knowledge sessions to learn from each other and from the knowledge experts.
- Organizing events such as Developer Day, Meetups for people to participate and learn from others.
- Creating Innovation labs within the company where the individuals can apply MVP principles and try new ideas.

What did you Grasp?

1. Which of the following is used for root-cause analysis?

   A) Kaizen
   B) MVP
   C) Agile
   D) DevOps

In a nutshell, we learnt:

1. We have learnt in depth about the DevOps CAMS model and each of the components that make DevOps a success.
2. We also learnt about the test-driven development and the different types of testing and its applicability.
3. About how organizations focus on organizational learning and the different methods of identifying and solving problems.

Take note of the key points covered in this module.

Release Notes

B. TECH CSE with Specialization in DevOps

Semester One -Year 01

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide and Mock exams.

Current Release Version.
1.0.0

Current Release Date.
2 July 2018

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Feb 2019