Collabera®

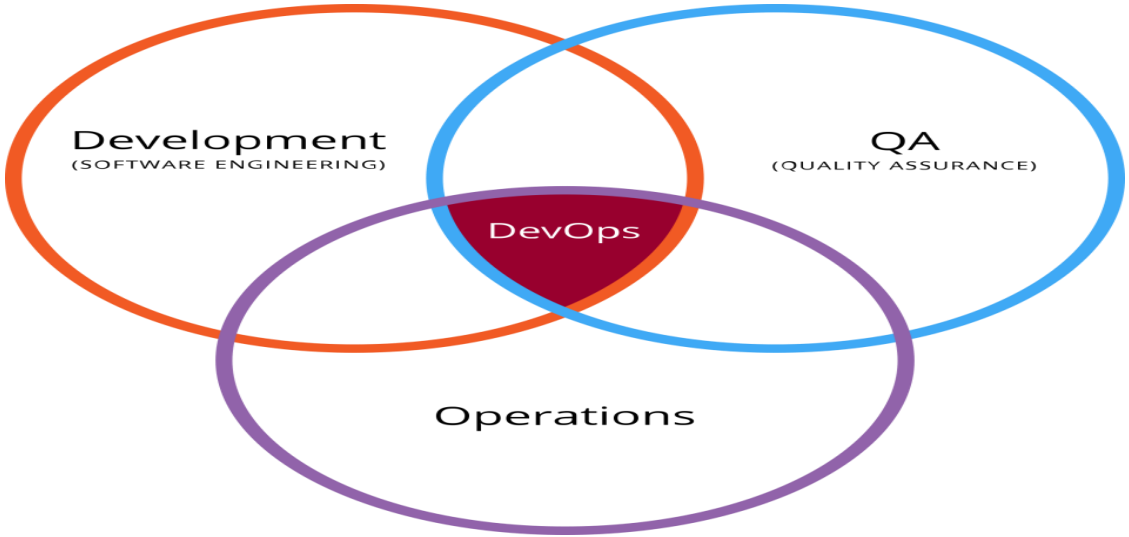
Value. Accelerated.

Collabera®

TACT 360° Training

Collabera

Introduction To DevOps



Topics For This Module:-

- Define Devops
- What's and Why's of Devops
- History of Devops
- SDLC models,Lean,ITIL,Agile
- Devops Stakeholders
- Devops Goals
- Important terminology
- Devops perspective
- Devops and AgileDevops Tools
- Configuration management
- Continuous Integration and Deployment

Define DevOps

- DevOps (development and operations) is an enterprise software development phrase used to mean a type of agile relationship between Development and IT Operations.

What's and Why's of DevOps

- The goal of DevOps is to change and improve the relationship by advocating better communication and collaboration between the two business units.
- DevOps practices and procedures lead to smoothing out the typically 'bumpiest' aspects of software development and deployment.

What's and Why's of DevOps

- By pulling infrastructure setup and awareness earlier in the development cycle, surprises from environmental differences are significantly reduced.
- By establishing consistent, reliable and repeatable automated deployments, human error and the need for 'rockstar firefighters' are reduced.

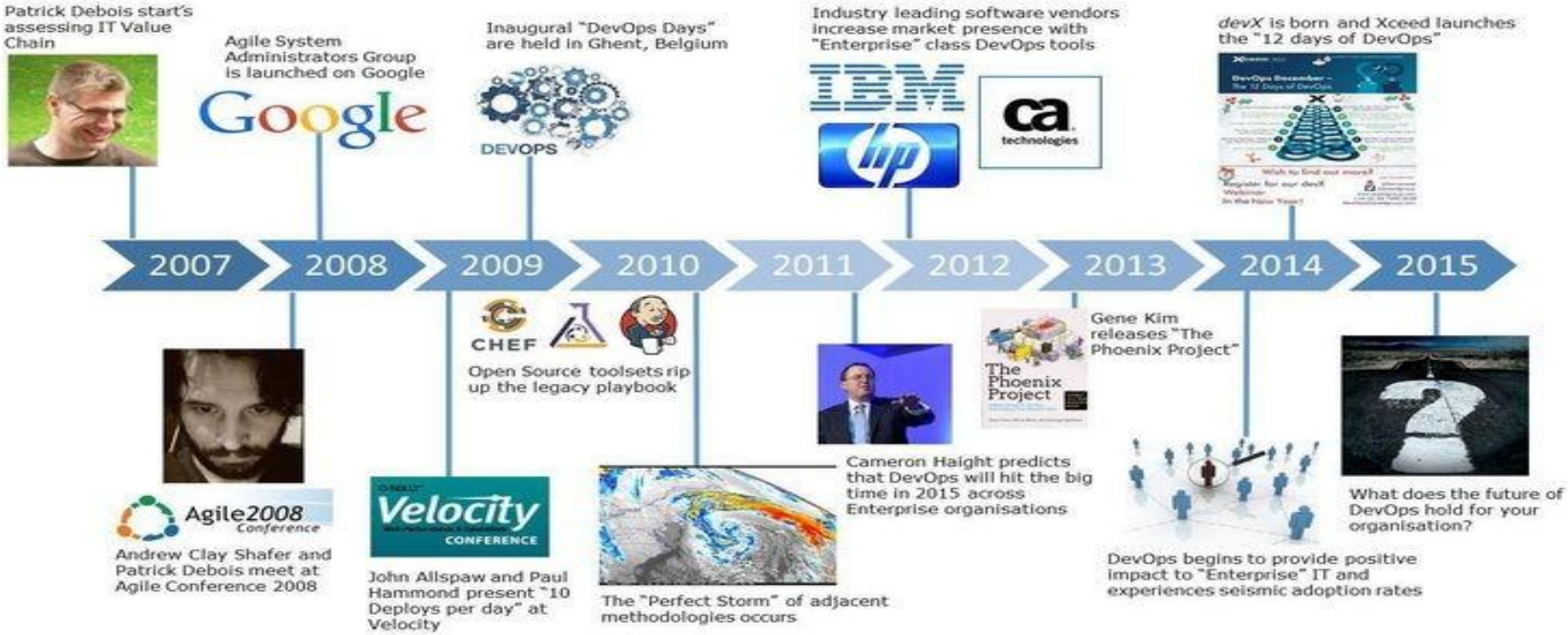
What's and Why's of DevOps:-

- In many ways, DevOps is about ensuring quality at all stages.

History Of DevOps

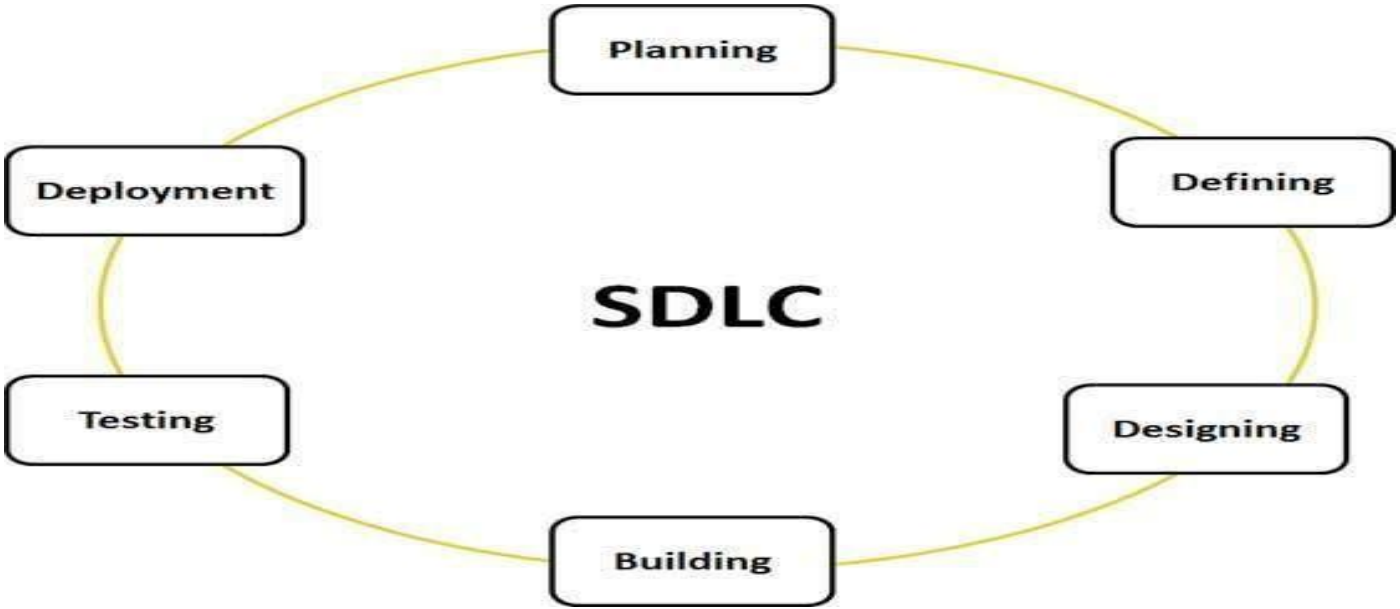
- The genesis of DevOps comes from an increasing need for innovation on the systems side of technology work. The DevOps movement inherits from the Agile System Administration movement and the Enterprise Systems Management (ESM) movement.
- DevOps arose as a reaction against the silos and inflexibility that were resulting from existing practices, which probably sounds familiar.
- DevOps emerged from a “perfect storm” of these things coming together. The growing automation and toolchain approach fed by more good monitoring and provisioning tools, the need for agile processes and dev/ops collaboration along with the failure of big/heavy implementations of ITSM/ITIL – they collided and unconsciously brought together all three layers of what you need for the agile movement (principles, process, and practices) and caught fire.
- Since then it has developed further, most notably by the inclusion of Lean principles by many of the thought leaders.

The History of DevOps



SDLC (Software Development Life Cycle):-

- Software Development Life Cycle is a process for designing, developing and testing enterprise applications to ensure high quality.

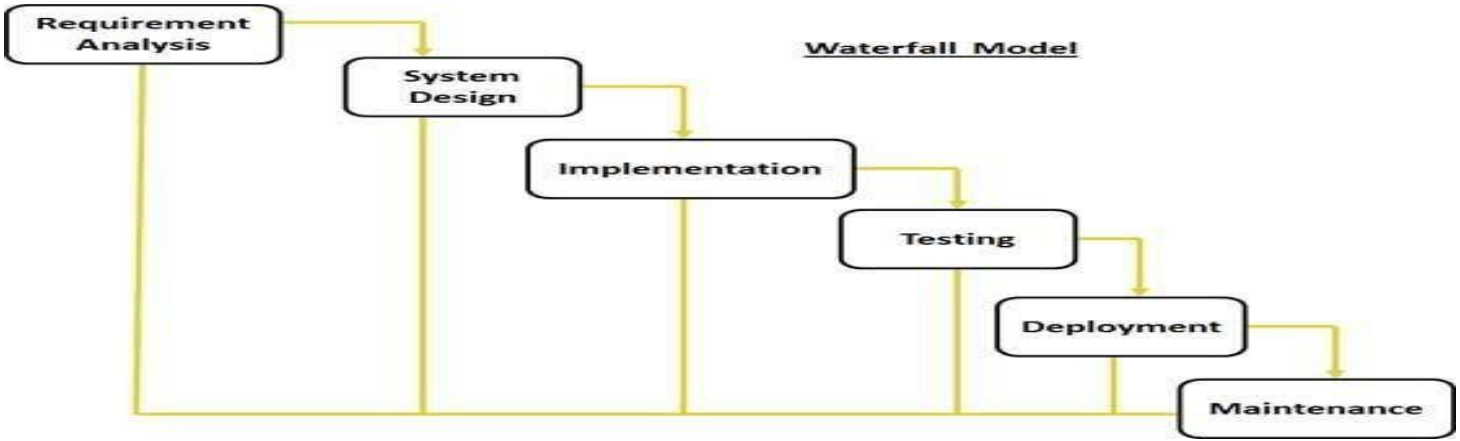


SDLC Models

- Waterfall Model
- Iterative Model
- RAD Model
- V Model
- Prototype Model

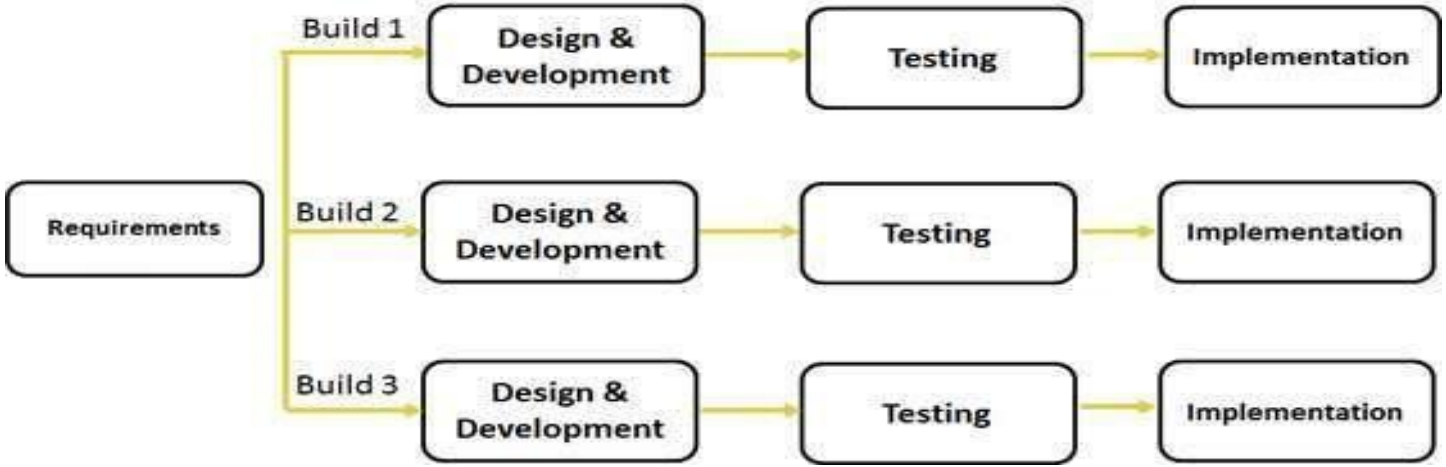
SDLC Models (Waterfall)

- The Phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases.
- The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.



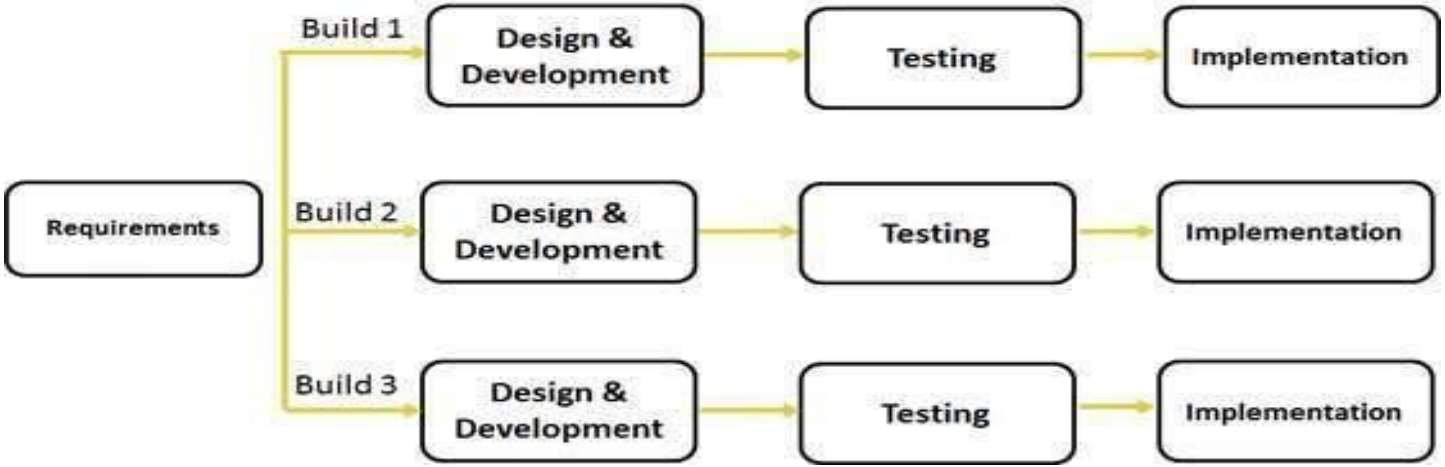
SDLC Models (Iterative)

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.



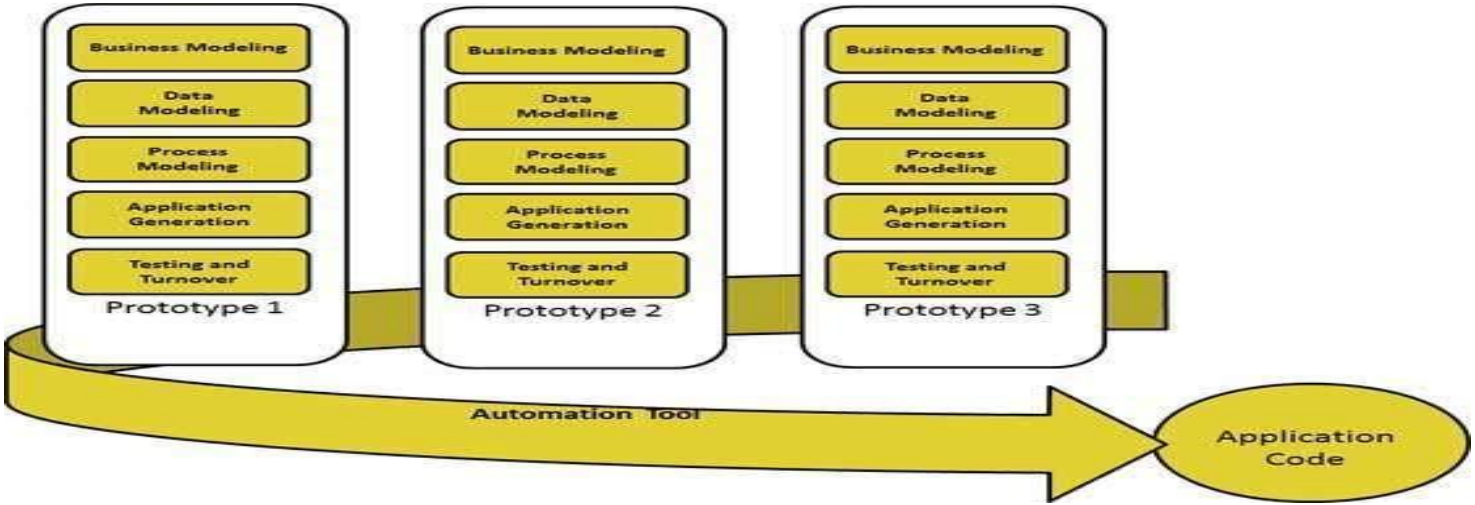
SDLC Models (Iterative)

- Resources with needed skill set are not available and are planned to be used on contract basis for specific iterations.
- There are some high risk features and goals which may change in the future.



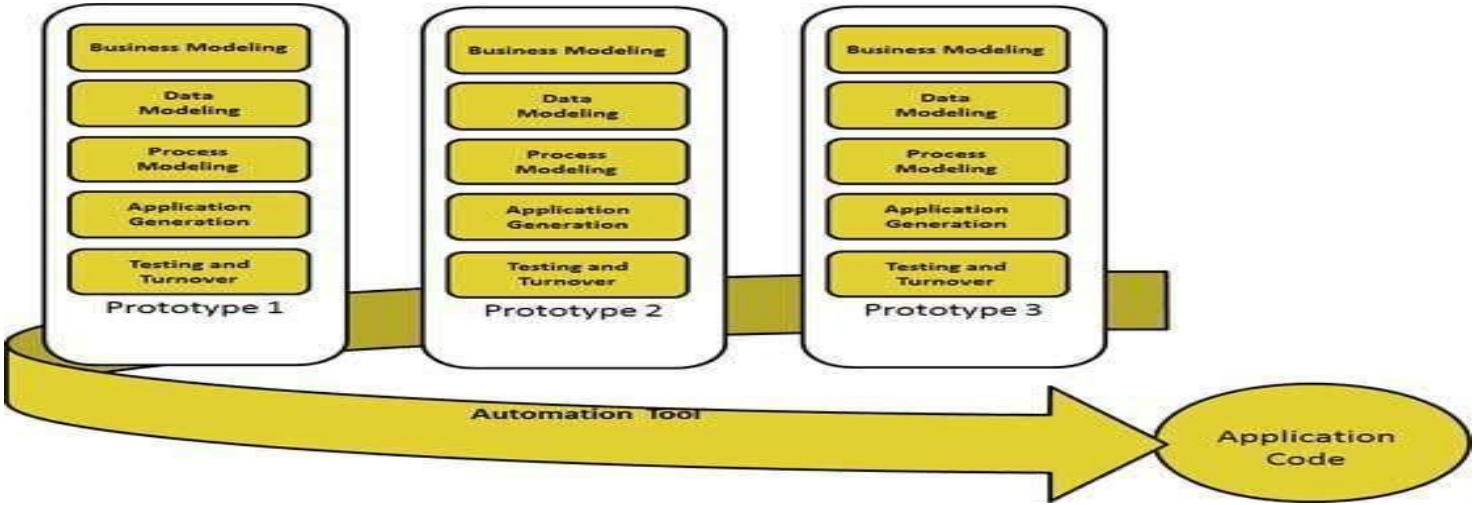
SDLC Models (RAD)

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
- It should be used if there is high availability of designers for modeling.



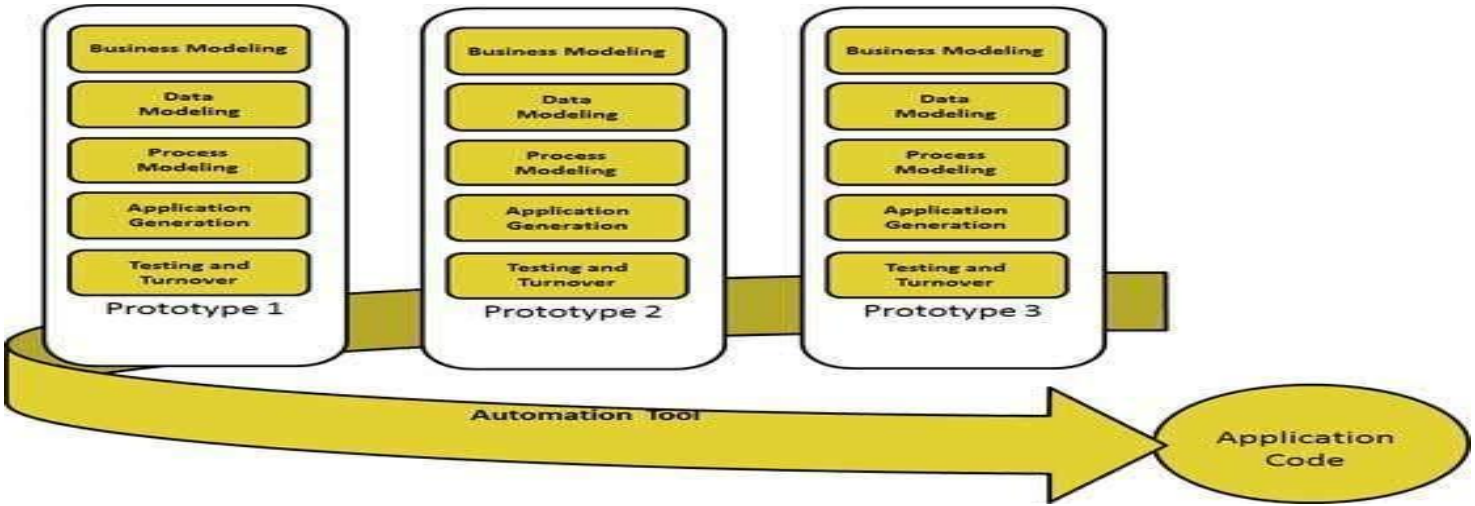
SDLC Models (RAD)

- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.



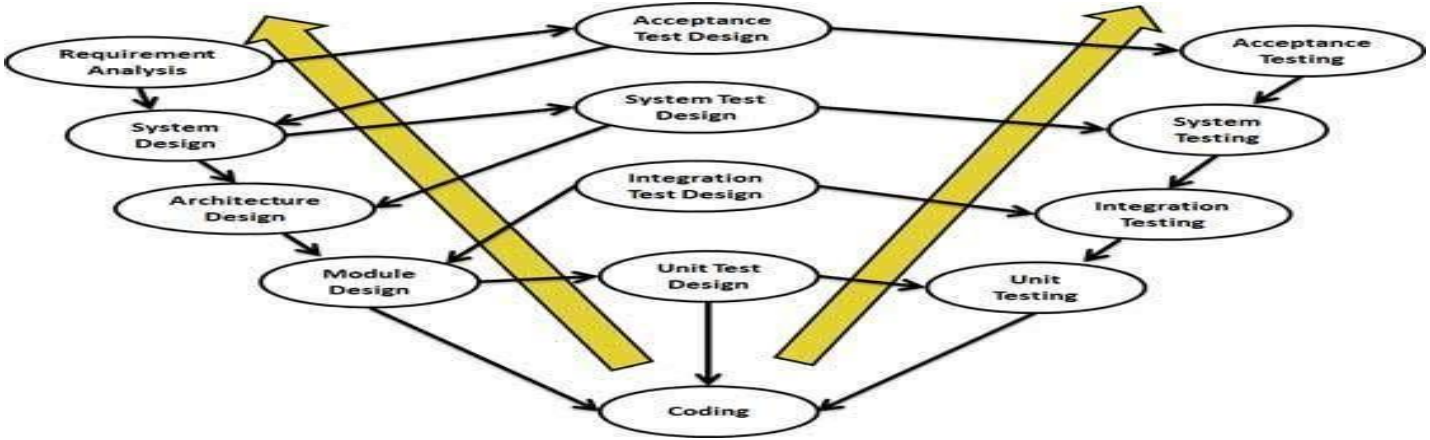
SDLC Models (RAD)

- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.



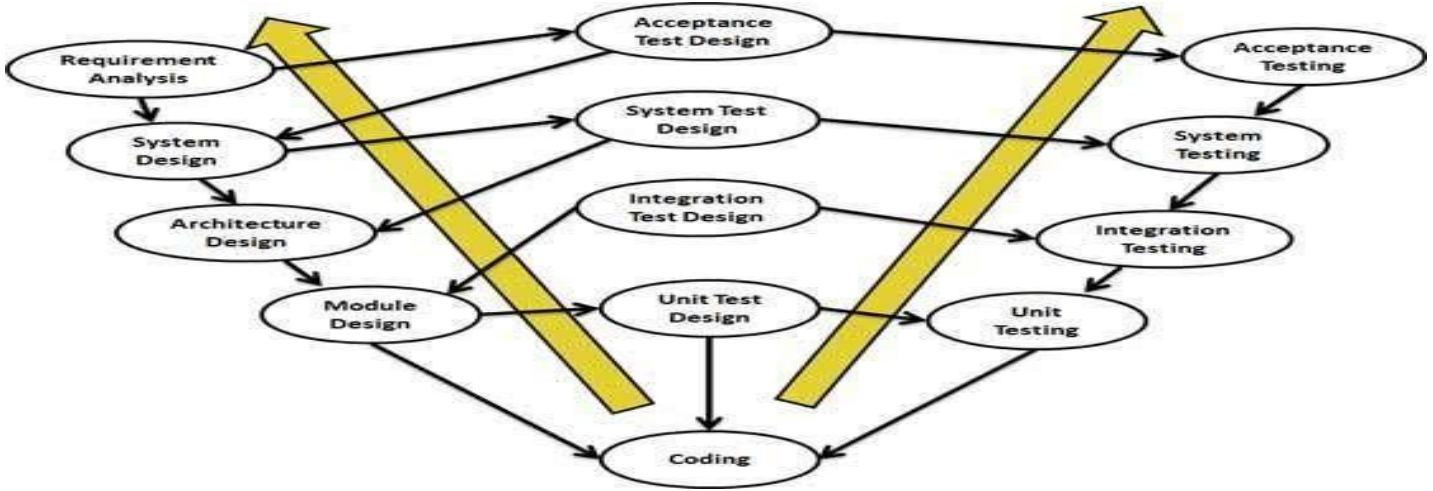
SDLC Models (V)

- The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model.
- V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage.



SDLC Models (V)

- This means that for every single phase in the development cycle there is a directly associated testing phase.
- This is a highly disciplined model and next phase starts only after completion of the previous phase.



SDLC Models (Prototype)

- The prototyping approach is used in the requirement gathering and in the analysis phase to capture the exact requirement of the proposed system.
- After the requirements are frozen, the remaining phases of the development process needs to be executed to complete the development of the software system.

SDLC Models (Prototype)

- An e-commerce website, such as shopping site is an example where you can implement the prototyping approach. You can develop the prototype of the various web pages of the shopping site such as catalogue page, product order page etc., and present it to the customer for approved.

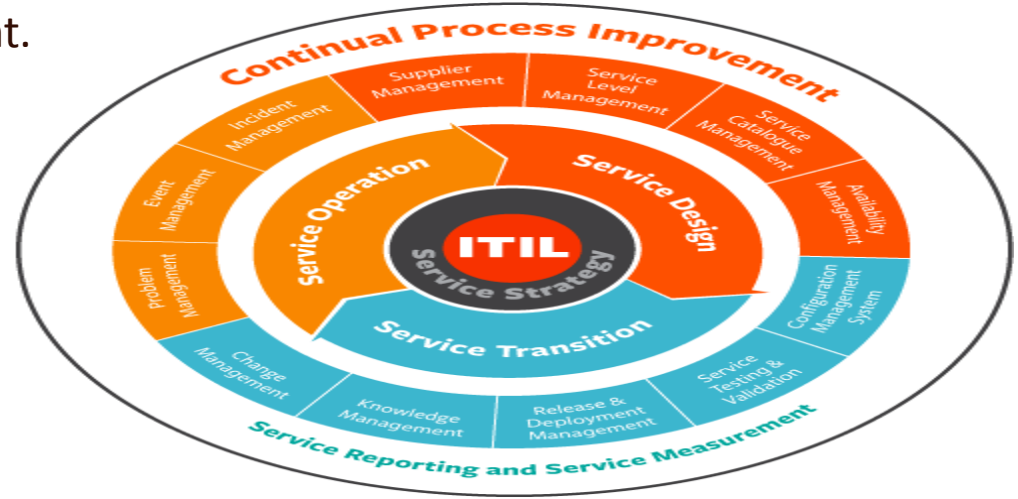
If the customer approves the prototype of the site, requirements are stated again and the design of the web site is initiated. If the customer does not approve the web site, the development team revisits the prototype and resubmits it to the customer for approval. This process continues until the prototype is approved.

Throwaway prototypes: Prototypes that are eventually discarded rather than becoming a part of the finally delivered software. Examples of throwaway prototypes include screen mock-ups and story boards.

Evolutionary Prototypes: prototypes that evolve into the final system through iterative incorporation of user feedback.

ITIL

- ITIL a.k.a. Information Technology Infrastructure Library helps individuals and organizations use IT to realize business change, transformation and growth.
- The ultimate goal of ITIL is to improve how IT delivers and supports valued business services.
- Every organization delivers a service or product. For every service or product, the ITIL framework helps manage delivery, industrialization, support, and consumerization from inception to retirement.

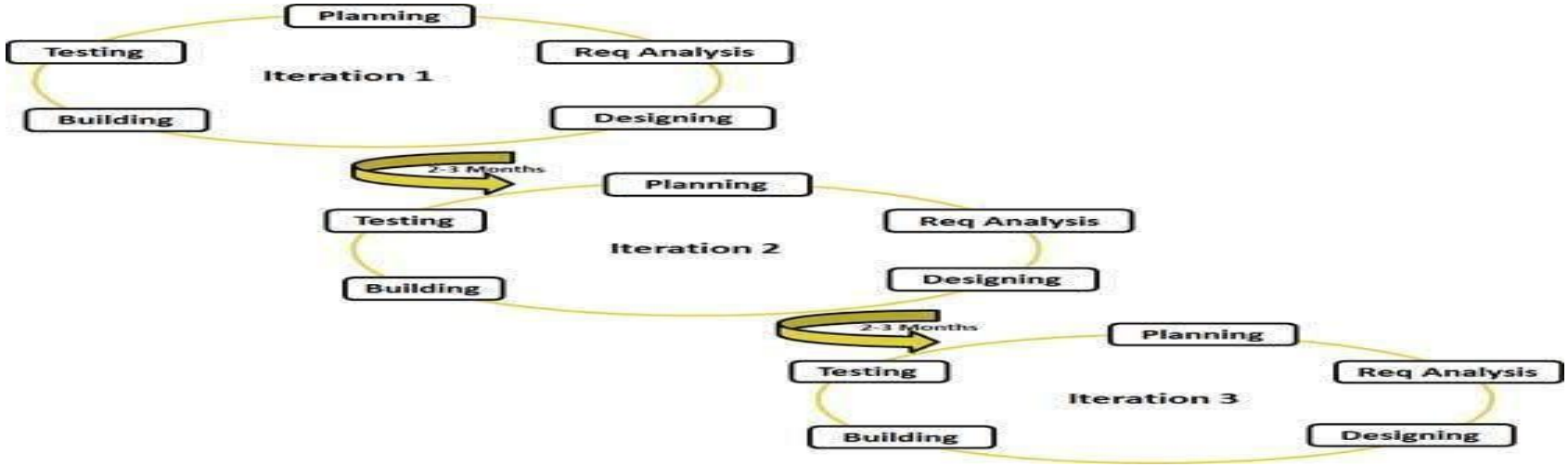


ITIL (Advatages)

- Stronger alignment between IT and the business.
- Improved service delivery and customer satisfaction.
- Reduced costs through improved use of resources.
- Greater visibility of IT costs and assets.
- Better management of business risk and service disruption or failure.
- More stable service environment to support constant business change.

Agile

- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds.These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.



Agile

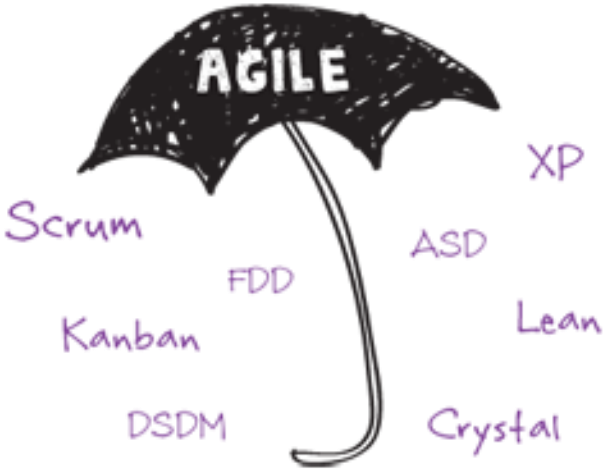
- Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
- At the end of the iteration a working product is displayed to the customer and important stakeholders.

Agile (Agenda)

- **Individuals and interactions -** in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software -** Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.
- **Customer collaboration -** As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change -** agile development is focused on quick responses to change and continuous development.

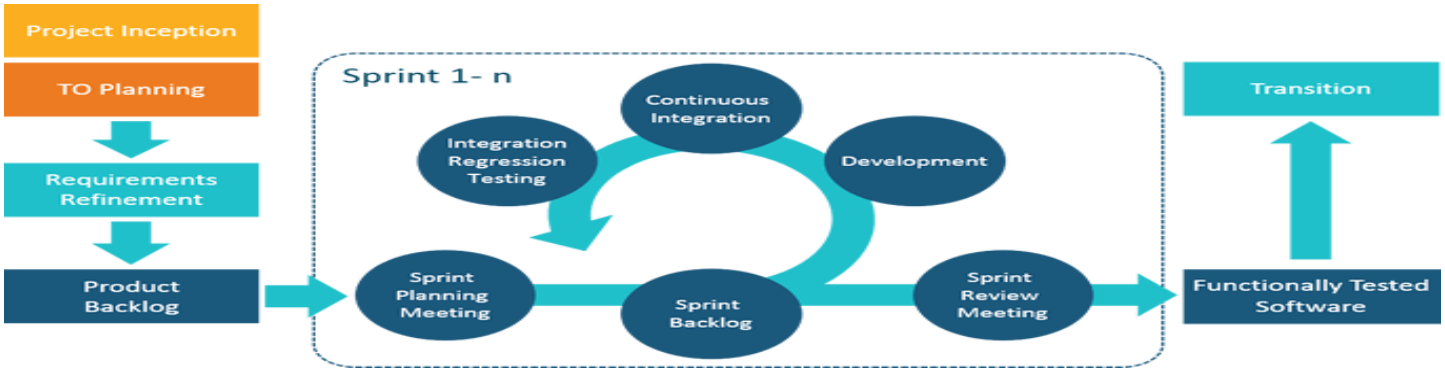
Agile (Types)

- Scrum
- Kanban
- Lean
- Extreme Programming (XP)



Agile (Types - Scrum)

- Scrum is a lightweight agile project management framework with broad applicability for managing and controlling iterative and incremental projects of all types.
- Scrum has garnered increasing popularity in the agile software development community due to its simplicity, proven productivity, and ability to act as a wrapper for various engineering practices promoted by other agile methodologies.

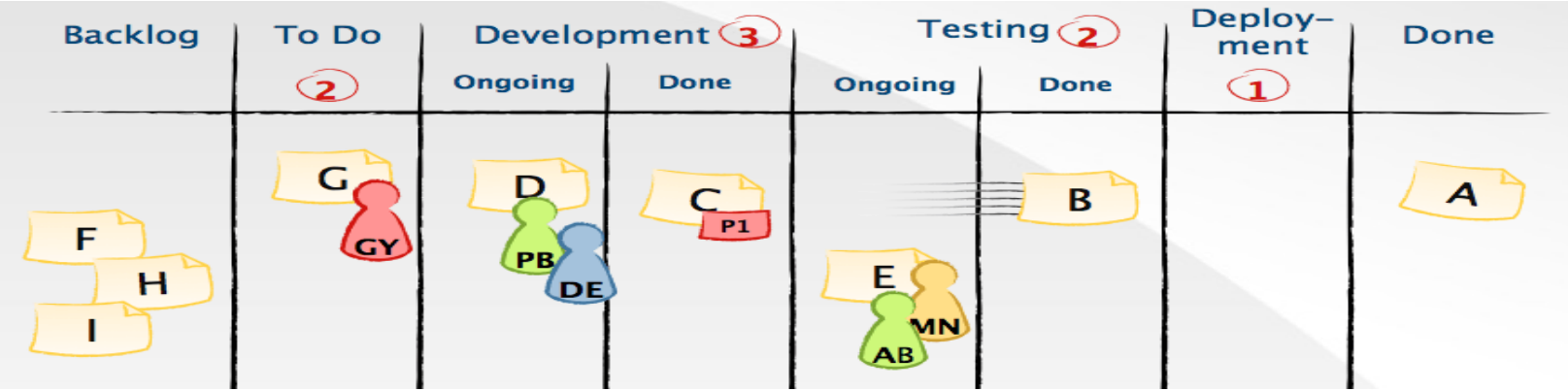


Agile (Types - Scrum)

- The Product Backlog consists of features, bug fixes, non-functional requirements, etc. – whatever needs to be done in order to successfully deliver a working software system.
- With priorities driven by the Product Owner, cross-functional teams estimate and sign-up to deliver “potentially shippable increments” of software during successive Sprints, typically lasting 30 days.
- Once a Sprint’s Product Backlog is committed, no additional functionality can be added to the Sprint except by the team.
- Once a Sprint has been delivered, the Product Backlog is analyzed and reprioritized, if necessary, and the next set of functionality is selected for the next Sprint.
- Scrum methodology has been proven to scale to multiple teams across very large organizations with 800+ people.

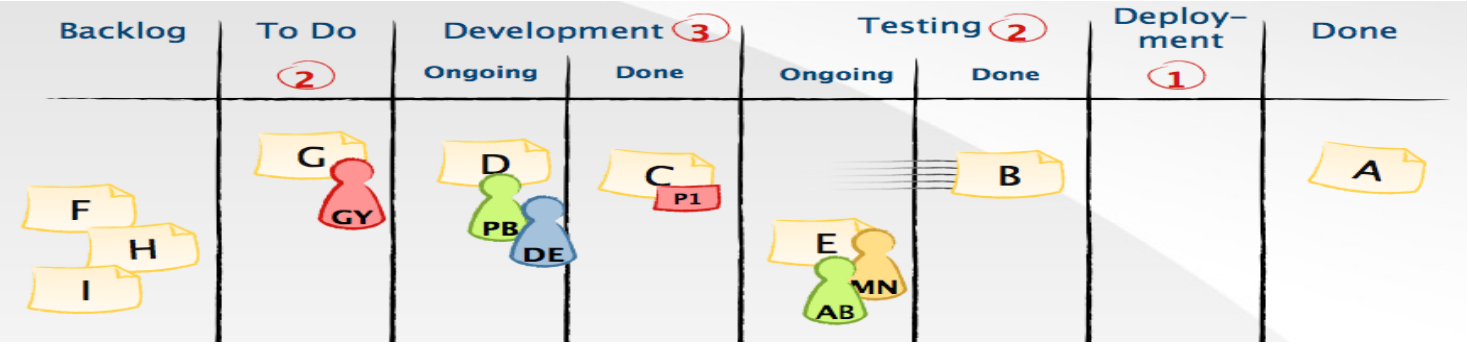
Agile (Types - Kanban)

- The Kanban Method is used by organizations to manage the creation of products with an emphasis on continual delivery while not overburdening the development team.
- Like Scrum, Kanban is a process designed to help teams work together more effectively.



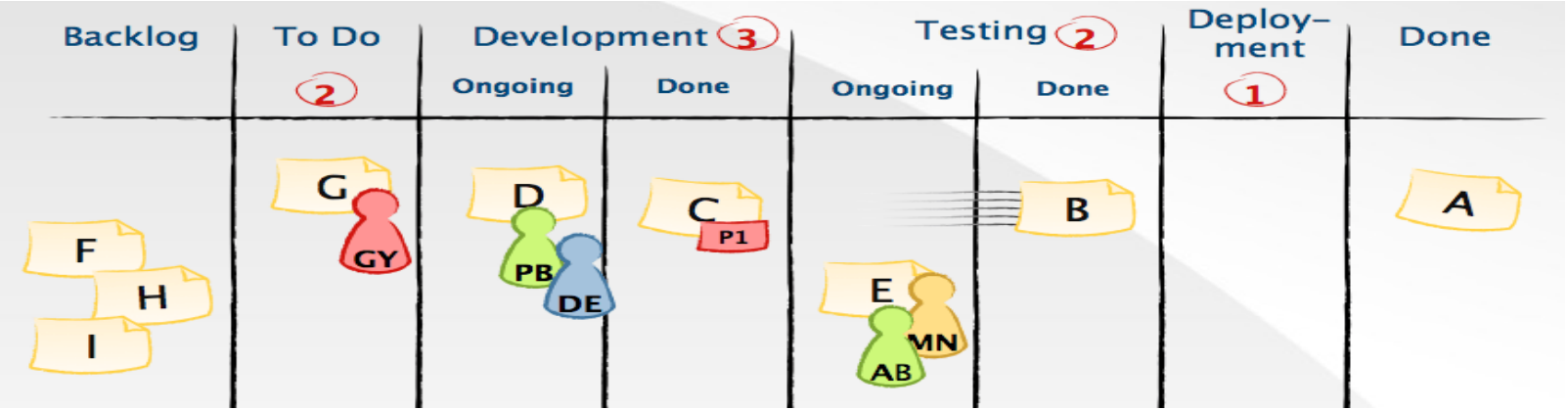
Agile (Types – Kanban: Principles)

- **Visualize what you do today (workflow):** seeing all the items in context of each other can be very informative
- **Limit the amount of work in progress (WIP):** this helps balance the flow-based approach so teams don ‘t start and commit to too much work at once.
- **Enhance flow:** when something is finished, the next highest thing from the backlog is pulled into play



Agile (Types – Kanban)

- Kanban promotes continuous collaboration and encourages active, ongoing learning and improving by defining the best possible team workflow.



Agile (Types – Lean)

- Lean Software Development is an iterative agile methodology originally developed by Mary and Tom Poppendieck.
- Lean Software Development owes much of its principles and practices to the Lean Enterprise movement, and the practices of companies like Toyota.

Principles of Lean Thinking

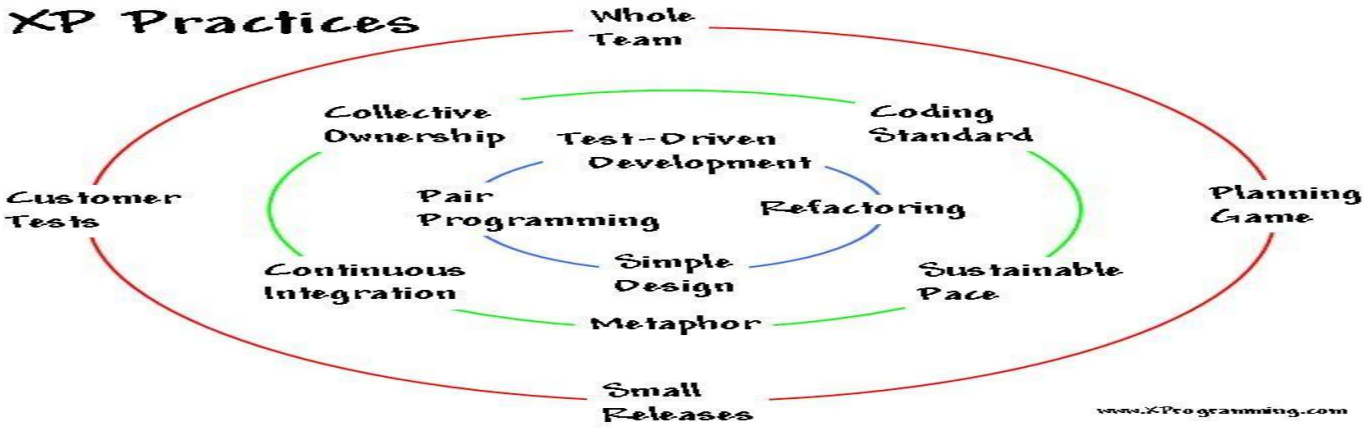
1. Eliminate Waste
2. Increase Feedback
3. Delay Commitment
4. Deliver Fast
5. Build Integrity In
6. Empower the Team
7. See the Whole

Agile (Types – Lean)

- Lean Software Development focuses the team on delivering Value to the customer, and on the efficiency of the "Value Stream," the mechanisms that deliver thatValue.
- It emphasizes the speed and efficiency of development workflow, and relies on rapid and reliable feedback between programmers and customers.
- Lean uses the idea of work product being "pulled" via customer request.
- It focuses decision-making authority and ability on individuals and small teams, since research shows this to be faster and more efficient than hierarchical flow of control.
- Lean also concentrates on the efficiency of the use of team resources, trying to ensure that everyone is productive as much of the time as possible.
- Lean methodology eliminates waste through such practices as selecting only the truly valuable features for a system, prioritizing those selected, and delivering them in small batches.
- It concentrates on concurrent work and the fewest possible intra-team workflow dependencies.
- Lean also strongly recommends that automated unit tests be written at the same time the code is written.

Agile (Types – Extreme Programming)

- XP, originally described by Kent Beck, has emerged as one of the most popular and controversial agile methodologies.



Agile (Types – Extreme Programming)

- XP is a disciplined approach to delivering high-quality software quickly and continuously. It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks.
- In XP, the “Customer” works very closely with the development team to define and prioritize granular units of functionality referred to as “User Stories”.
- The development team estimates, plans, and delivers the highest priority user stories in the form of working, tested software on an iteration-by-iteration basis.
- In order to maximize productivity, the practices provide a supportive, lightweight framework to guide a team and ensure high-quality software.

DevOps Stakeholders

- A fundamental philosophy of DevOps is that developers, operations staff, and support people must work closely together on a regular basis.
- An implication is that they must see one other as important stakeholders and actively seek to work together.

DevOps Stakeholders

- A common practice within the agile community is "onsite customer," adopted from Extreme Programming (XP), which motivates agile developers to work closely with the business.
- An implication is that they must see one other as important stakeholders and actively seek to work together.
- Disciplined agilists take this one step further with the practice of active stakeholder participation, which says that developers should work closely with all of their stakeholders, including operations and support staff--not just business stakeholders.
- This is a two-way street: Operations and support staff must also be willing to work closely with developers.

DevOps Goals

- Improved deployment frequency.
- Lower failure rate of new releases.
- Shorten lead time between fixes.
- Maximize predictability, efficiency,security and maintainability.
- Improved Communication, Collaboration and Integration.

Important Terminology

- **Continuous Deployment** - Deliver every successful build to production.
- **Continuous Delivery** - Every build can potentially be delivered to production.
- **Deploy** - Deploy code to production without releasing.
- **Release** - Make functionality available to end users.
- **Push** - Deployment and infrastructure changes are pushed from a central place to the servers.
- **Pull -** Each server queries a central place for deployment and infrastructure changes, and apply the changeslocally.
- **Deployment Pipeline** - At an abstract level, a deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users.

DevOps Perspective

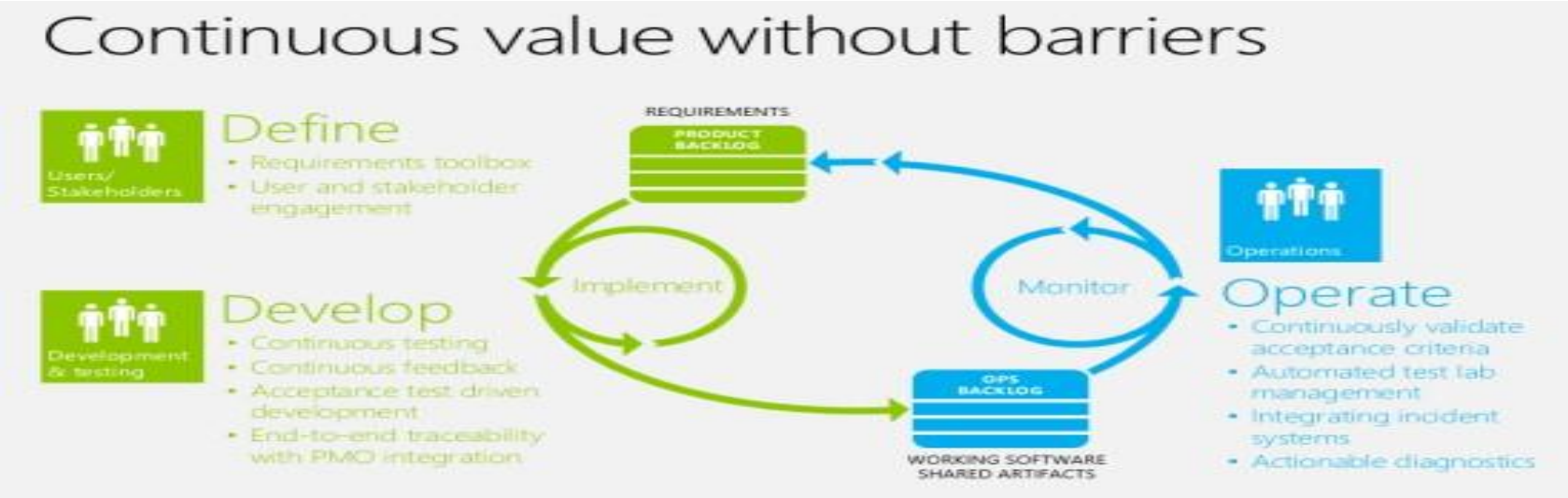
- Why DevOps?
  - How do we become a High Performing Culture?

- Leads to:-
  - Better products and services.
  - Increased customer satisfaction.
  - Improved profitability.

- DevOps is not:-
  - A person, role, or team.
  - A fix for IT.

DevOps & Agile

- DevOps & Agile complement each other to deploy working functionalityinto production faster.



DevOps & Agile (Key Aspects)

❑ Continuous involvement of the Ops team

- A fundamental requirement of DevOps is that the Ops team is continuously engaged with the development team throughout the life cycle of solution development.
- Ops should participate right from the visioning stage to understand the business vision, the epics, and the release timelines.
- They should also contribute to determining the solution's technical and schedule feasibility.

❑ Product owner

- The product owner (PO) is the face of the business for the development team.
- The PO has a vision for the product and has a complete insight into the functional requirements that must be met.
- However, if the non-functional requirements (NFRs) are not well understood and articulated by the PO, the development team will not be able to take them into account while creating the architecture and building the final solution.

DevOps & Agile (Key Aspects)

❑ Product backlog

- Typically a product backlog focuses on epics and stories related to the functional requirements.
- The PO and the Dev team are well trained to brainstorm, break down epics, and document the functional requirements.
- However, often the NFRs are not well specified in the backlog.
- In addition to the functional requirements, the backlog should describe items such as the following:-
    - Performance requirements.
    - Tech requirements related to deployment and support.
    - Requirements to develop the guidelines for rapid rollback and roll forward.
    - Security/firewall requirements

DevOps & Agile (Key Aspects)

❑ Ops representation in the team

- The Agile development team is cross-functional and self-organizing. Should this team then include an Ops person too? Maybe -- this depends on how cross- skilled the team members are.
- Typically in a start-up IT organization, some of the developers or testers would also be responsible for deployment and Level-3 (bug fixing) support.
- In such cases the requirements related to deployment and support would be well discussed in planning and review meetings.
- However , in large organizations, operations would need a large team dedicated to take over completed code from multiple Dev teams and deploy them. In such cases, job rotation could be a useful proposition.
- That is, some of the developers could play the Ops role for certain period of time, and some of the Ops team members with the right aptitude could be made part of the Dev team for a certain period of time.This way the Ops side would be well represented during the development cycle.

DevOps & Agile (KeyAspects)

❑ Definition of Done

- Another key lever to involve Ops in the development cycle is to weave in Ops- related aspects in the Definition of Done.
- Along with the standard coding, testing, and documentation elements, validation of the code in the deployment platform (e.g., a mock production box), specific support instructions as part of the documentation, and a dry run of these instructions should also be included in the Definition of Done.
- Here again the inputs from the Ops team are crucial.

❑ Sprint planning and daily stand-up

- Sprint backlog planning and daily stand-ups should pay attention to the Ops needs while prioritizing backlog items and discussing progress.
- The sprint backlog should include specific line items related to securing the necessary tech platforms for mock deployment and other such coordination activities.
- •It is a good idea to include the Ops team during sprint planning and in selected daily stand-ups where the team would discuss Ops aspects.
- Any dependency on infrastructure providers and system integrators should be considered at this stage using inputs from the Ops team.

DevOps & Agile (Key Aspects)

❑ Sprint review

- When the Ops team is continuously involved in the development cycle, it goes without saying that they should be part of the sprint review as well.
- The Dev team should demonstrate the Ops-related features of the solution. Clearly, all sprint reviews may not include Ops-related features.
- However, if the Ops team is part of the demos, they have an opportunity to see what is coming up and provide inputs for the subsequent sprints to improve the product and include Ops requirements as well.
- Here again, if one or more of the Dev team members represent Ops, it is easy to get this alignment.
- If Ops is a separate team, the Dev team has to make sure to get the Ops team for product demos.

DevOps & Agile (Key Aspects)

❑ Scrum of Scrums

- When multiple Scrum teams work on a solution, the integration of the output of each team has to be carefully planned and executed.
- Each Scrum team should take into account the Ops requirements and build features in alignment with the requirements.
- The product owners should have a view of the final product, how it will be developed through multiple Scrum teams, and where and how it will be deployed.
- They should involve the Ops teams to provide specific inputs for each Scrum team.
- At Scrum of Scrum events, the POs should come together and validate that the Scrum teams in fact include these in their plans and demos.

DevOps & Agile (Key Aspects)

❑ Plan alignment

- DevOps and Agile complement each other well and help the business and release teams plan the annual release calendar.
- With continuous engagement and collaboration with the development team, the Ops team gets to know which functionality will be coming out when.
- With that insight, and using the sprint completion pattern, the Dev and Ops teams should be able to predict with reasonable accuracy the potential release dates.
- Eventually they should strive to align the release schedule with the sprint plans.
- And when this alignment happens, the support team would be able to move completed functionality to production faster and at shorter intervals -- and a key benefit of Agile is realized!

DevOps & Agile (KeyAspects)

❑ Metrics

- To measure how DevOps would help faster releases, a few leading and lagging indicators such as the following could be used.
- Industry statistics demonstrate that organizations using DevOps and Agile are able to make multiple releases to production in a single day.
    - Release date adherence percentage.
    - Percentage increase in the number of releases.
    - Time taken for release to production.
    - Defects attributable to platform/support requirements.
    - Percentage of NFRs met.

❑ Conclusion

- When Agile and DevOps come together, the IT organization can see tangible outcomes.
- DevOps is much larger and involves multiple dimensions including business, IT, and vendor stakeholders.

DevOps Tools

- **Version Control**
  - TFS
  - GIT
  - SubVersion
- **Build & Deployment**
  - TFS
  - Jenkins
- **Monitoring**
  - SCOM
  - Nagios
  - Pingdom, StatusCake, PagerDuty, VictorOps
- **Configuration Management**
  - Powershell DSC
  - Puppet
  - Chef
  - Ansible

Configuration Management

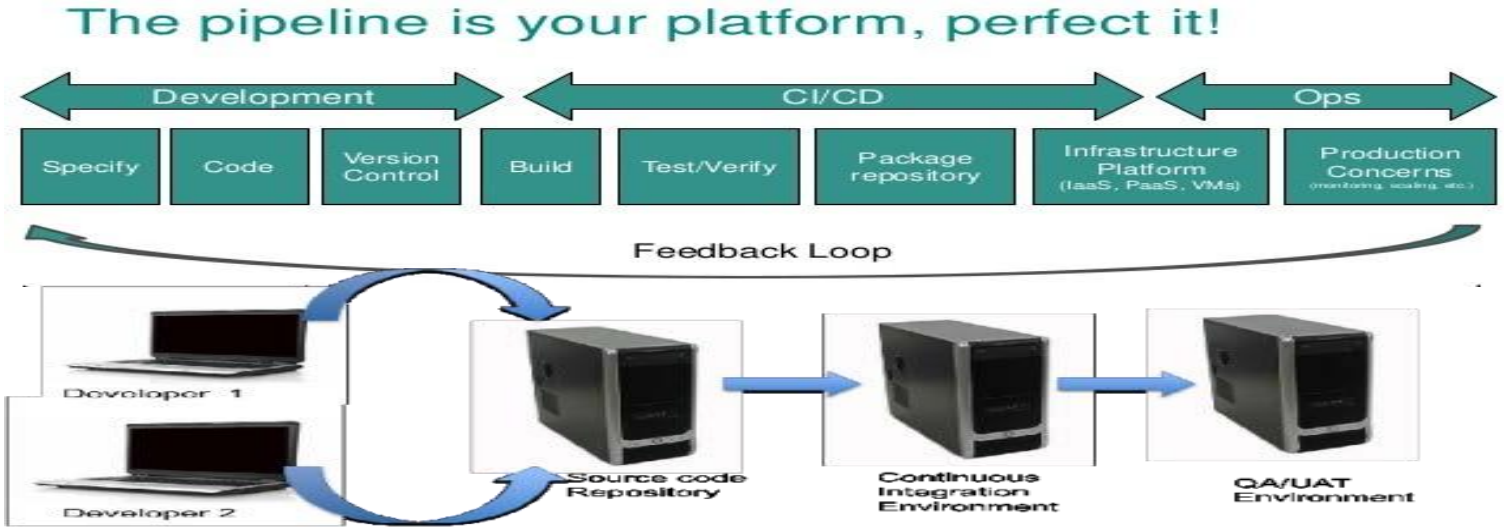
- Configuration management (CM) is the detailed recording and updating of information that describes an enterprise's hardware and software.
- Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted and doesn't rely on the tacit knowledge of the development team.
- Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted and doesn't rely on the tacit knowledge of the development team.

Configuration Management (Key Benefits):-

- Increased efficiencies, stability and control by improving visibility and tracking.
- Cost reduction by having detailed knowledge of all the elements of the configuration which allows for unnecessary duplication to be avoided.
- Enhanced system reliability through more rapid detection and correction of improper configurations that could negatively impact performance.
- The ability to define and enforce formal policies and procedures that govern asset identification, status monitoring, and auditing.
- Greater agility and faster problem resolution, thus giving better quality of service.
- Decreased risk and greater levels of security.
- More efficient change management by knowing what the prior structure is in order to design changes that do not produce new incompatibilities and/or problems.
- Faster restoration of service. If you know the desired state of the configuration and how they are interrelated, recovering the live configuration will be much easier and quicker.

Continuous Integration & Deployment

- CI is the process of executing the software Build-Deploy-Test (BDT) cycle frequently with minimal manual intervention with the underlying principle as one of constant communication and feedback among team members.
- CD builds upon the earlier concept by providing fast, automated feedback on the *correctness* and *production readiness* of your application every time there is a change to *code, infrastructure, or configuration*.



For technical support:

Call : +91 75730 27611

E-mail : tactsupport@collabera.com

Visit us at : http//:www.collaberatact.com

collabera