Software Development

Application Lifecycle Management

When creating an application, we outline the entire life cycle with a few key concepts referred to as Application Lifecycle Management (ALM). These concepts are not limited to the development of the application, and are concerned with everything from decision making to test frameworks, maintenance schedules and processes, and developer resources / customer security.

- Governance - How will decisions be made that impact the future of the application?
- Requirements - What does the application need in order to succeed, what are the key features and use cases?
- Resources - How will the application handle key resources and security? Who will be able to access certain features, who will have visibility into the source or implementation of the application?
- Development - How will we approach the process of designing, building, testing, and deploying the application? (Using Agile?)
    - This stage is where the Software Development Life Cycle (SDLC) is focused. In contrast the ALM is focused on the entire llife cycle of the application, before, during, and after the development process, all the way to the impact these processes have on the end user.
- Testing - How will we manage testing new contributions to the application to validate that it behaves as expected, prior to releasing new builds to end-users?
- Maintenance - How do we plan to maintain the application, how will we upgrade existing features and deploy new ones?

Software Development Life Cycle

The Software Development Life Cycle is mostly concerned with the development of the application. In general, the development of an application follows these steps

- Planning - Requirement Analysis based on feedback from users, marketing teams, and sales departments which helps to outline the basic requirements needed for the project to remain feasible from an economic and business perspective
- Defining - Further defining the requirements derived from the Planning phase, to more specifically outline the next steps for project success via a Software Requirement Specification (SRS) that provides an in-depth look at each requirement
- Designing - The SRS is either improved upon to describe project architecture and design, or a Design Document Specification (DDS) is created to do the same. These documents are reviewed by stakeholders to determine the best course of action moving forward.

- Building - Following coding guidelines and best practices as close as possible, the actual development begins on the project using the SRS and/or DDS
- Testing - The applicaiton is tested using the new features or fixes that were implemented as a result of previous stages in the SDLC
- Deploying - Once tests are passing, the new or upgraded application is deployed to the end users who then provide necessary feedback for the next iteration of the SDLC

There are many models for software development -

- Waterfall
- V Model (Waterfall 2.0)
- Iterative
- Spiral
- Agile

Continuous Integration

Continuous Integration (CI) often used to streamline testing new contributions to a repository by verifying builds, test cases, and merges to aid in ALM and SDLC. By automating building the application, we ensure that the change is valid in respect to the tools and technologies that assemble the end product. We then test this build with a series of Unit / Integration / Functional test cases, and when all of these pass we proceed to attempt a merge of the changes into the current code base. This process ensures that the changes being made are improvements on the current status of the application, and not degredations. This process also ensures that changes are more frequently merged into a common repository or branch, whereas without CI we risk building up a large set of changes that could otherwise be difficult to merge into the current code base. CI helps to avoid this complexity and in-turn reduce development time required to make changes to the product, while simultaneously ensuring that bugs are realized and fixed earlier in development rather than later.

Continous Delivery

Continuous Delivery (CD) is the delivery of the changes to a remote repository, where this delivery takes minimal effort on the behalf of development teams which enable them to focus their work on the improvement of the application without having to manually deliver their changes each time. In order for delivery to be effective, it's important to have a good CI process in place that protects the code base from unwanted bugs or breaking changes. The delivery process can include deploying the application to a testing environement where more tests are performed to validate the changes made.

This step can optionally automate deployments also, or there can be an additional system that handles Continuous Deployment strategies independent from repository / source code management.

Continous Deployment

Continuous Deployment (CD) is the deployment of changes deliverd by a Continuous Delivery system. This process can either be independent from or part the of Continous Delivery strategy. Once tests from all previous stages in the CI / CD pipeline are passing, the changes are deployed to production where the end users can benefit from the new features or upgrades made in this iteration of development.

CI / CD Pipeline

The CI / CD pipeline has several meaning that are all context dependent for the application

- CI and Continuous Delivery
- CI, Continous Delivery, and Continuous Deployment
- CI and Continous Delivery utilizing a system that also handles deployments to production

In general, the CI / CD pipeline refers to the use of automation tools to streamline the development of an application.

Some CI / CD tools are listed below. Some of these focus on CI, Continuous Delivery, Continuous Deployment, or any combination of the three.

- Jenkins
- Buildbot
- CircleCI
- GitLab CI
- Bamboo
- Codeship
- Octopus Deploy
- Deploybot
- AWS CodeDeploy

Resources

RedHat - Application Lifecycle

RedHat - CI / CD

Tutorialspoint - SDLC

[Wikipedia - SDLC]

[Guru99 - CI tools]

Revision #1
Created 2022-04-03 12:20:31 UTC by Shaun Reed
Updated 2022-04-03 14:57:38 UTC by Shaun Reed