White paper
Cisco public

AI / ML Powered DevOps

January 2021

Contents

Problem Statement and Challenges

Service providers and enterprise customers are migrating from hardware based network functions to software based network functions using virtualization and container technologies. A key driver of speed, quality and reliability is alignment across the board amongst the vendors whose software components are integrated. DevOps is required to deliver integrated software applications at a high velocity generally not attainable through the traditional software management processes.

In order to achieve operational efficiency, reliability and rapid innovation at scale, Web Companies created Cloud-Native Architecture, a software centric approach with openness and interoperability. Well established and well tested mechanisms for rapid innovation in software are enabled by Continuous Integration (CI) and Continuous Deployment (CD). Advancements in Artificial Intelligence/Machine Learning (AI/ML) have paved the way to significantly improve the productivity of personnel while simultaneously reducing risk. In this paper, we show the use of AI/ML to improve productivity, reliability and quality of DevOps processes.

Thus, software distribution and delivery in a Cloud Native heterogeneous vendor environment is critical for any consumer in order to be agile, bring innovations in a short time to market and obtain a cost competitive advantage. Meeting the carrier grade requirements and complexities involved in automating the DevOps process due to multi-vendor environments, rigid processes, compliance and security concerns are slowing down the adoption. AI/ML can now assist service providers and enterprises to tackle the aforementioned complexities.

The vision of “Move fast and build things” or “Continuous Innovation” is now closer to reality than ever before. An acceleration in innovation at the infrastructure level can have an exponential impact on the rate of innovation for products that depend on it.

Proposal

Our proposal to advance the automation of DevOps backed by real world implementation includes AI/ML as the core of the strategy. AI/ML Powered DevOps is a comprehensive SaaS based application to enhance the DevOps capabilities that exist in your environment.

At the core of this proposal, we introduce:

1. AI/ML powered DevOps capabilities based on cloud-native microservice architecture that can be consumed as a SaaS.
2. A multi-phased approach to deploy these capabilities towards an Autonomous DevOps in your environment.

Based on our incubation work, we have developed a cloud-native microservice Architecture with AI/ML core capabilities to transform any Automated DevOps framework into an Autonomous one. The AI/ML powered DevOps has three major components of capabilities for easier and measurable rollout.

1. Information Discovery
2. Intelligent Automation
3. Risk Management

AI/ML Powered DevOps

A vertical integration of software can be achieved in the infrastructure world by enlisting vendors to deliver software builds that run on commodity hardware in a consistent manner. Envision a platform that brings together these varied components in a homogenous way through a declaration of software component version numbers, release information in a format that can be machine translated, performance characteristics that are quantifiable and compatibility matrices digitized.

A standard DevOps process is shown in Figure 1. This paper covers how AI/ML powered DevOps capabilities can be leveraged in any Automated DevOps Platform. For an introduction to implementing a software delivery and automation framework for virtualized network functions, please refer to [1].

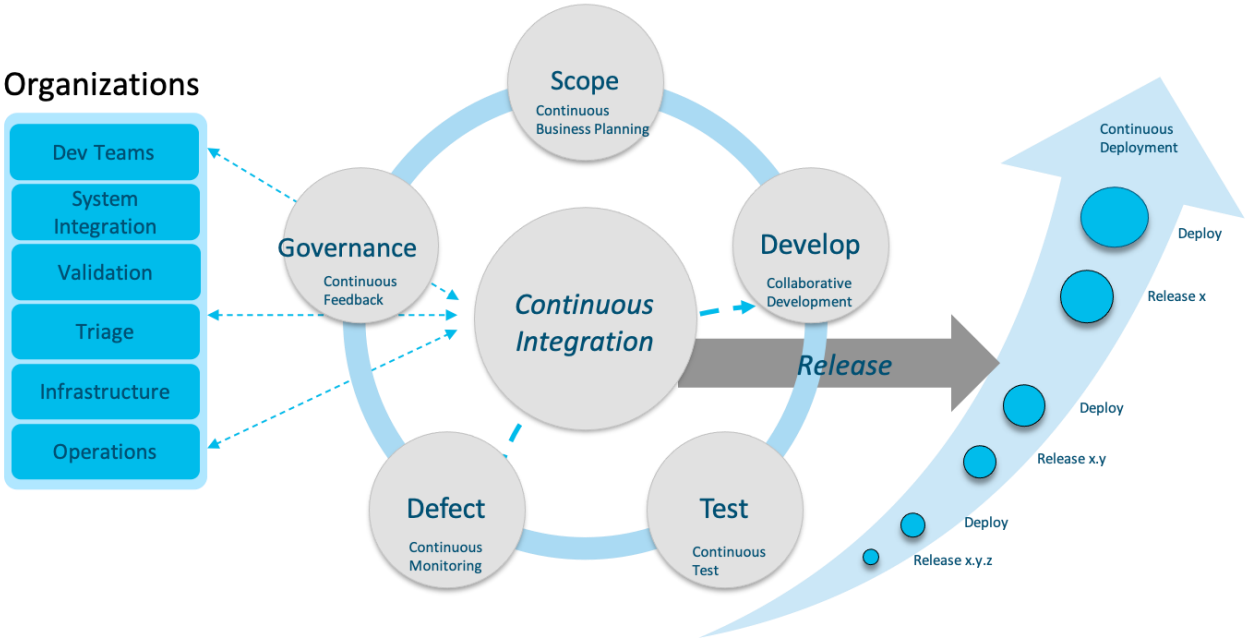

Figure 1.
The DevOps Process

A journey towards Autonomous DevOps requires a phased approach as shown in Figure 2. Many of you are in various phases of this journey. So, irrespective of your situation, the introduction of AI/ML can bring positive outcomes.

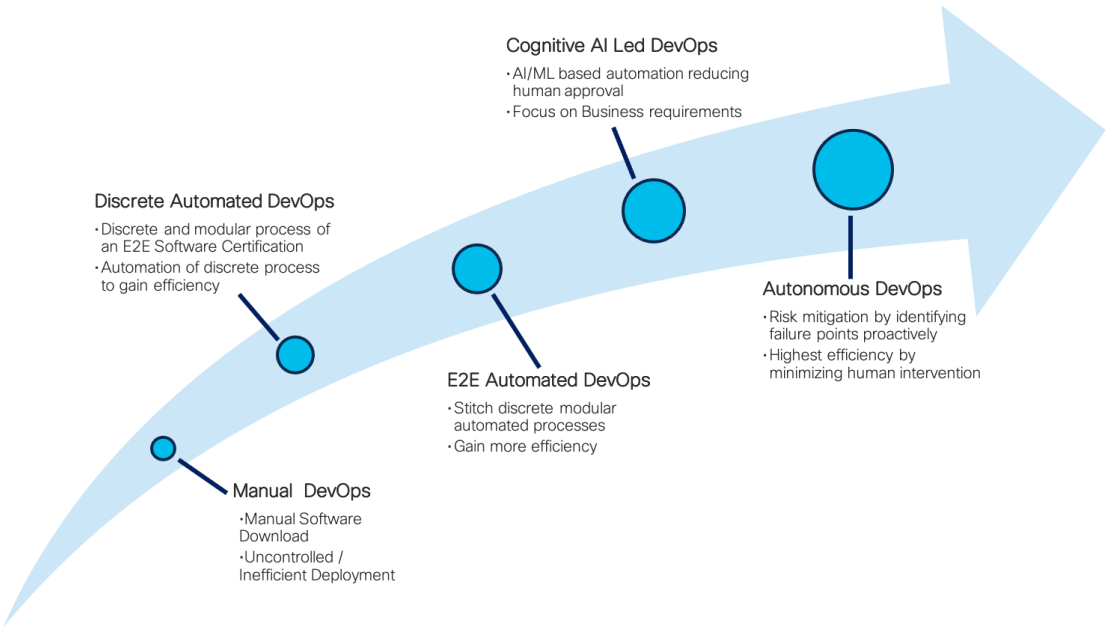

Figure 2.
Phased approach to DevOps Automation

The DevOps phases illustrated in Figure 2 are explained below:

Manual DevOps

In this phase, a detailed process for software development and delivery between various development, operations and business organizations are created. Most of the communication will be verbal and/or written documentation. Most of the implementations will be manual, manual software builds, manual testing, manual release process etc.. Very few customers stop at this phase and usually we see this in an initial transformation journey of an organization.

Discrete Automated DevOps

As the transformation journey continues, different operations are automated in the DevOps process, but still there could be many manual steps in transitioning from one operation to the next.

End-to-End Automated DevOps

In this phase, End-to-End processes are automated. In addition, end-to-end visibility and insight is provided to track status.

Cognitive AI/ML led DevOps

Even with end-to-end automation, we noticed high error rates during testing and in production. Test teams under time constraints choose a representative set of test cases that contain duplicates oftentimes leading to the omission of required test cases. Such omissions lead to higher error rate in production. During this phase, AI/ML Engines are leveraged either for Discrete or End-to-End Processes to improve reliability, quality and efficiency.

Autonomous DevOps

In what can be termed as the final frontier, the DevOps platform takes in the feedback from the production environment to self-drive the various outcomes of DevOps processes like software build, certification and eventual deployment of the software in production.

AI/ML Powered DevOps Architecture

Based on the innovative work in a large 5G mobility provider, we propose an AI/ML DevOps Architecture to meet the vision of Autonomous DevOps. The core of this Architecture is depicted below in Figure 3.

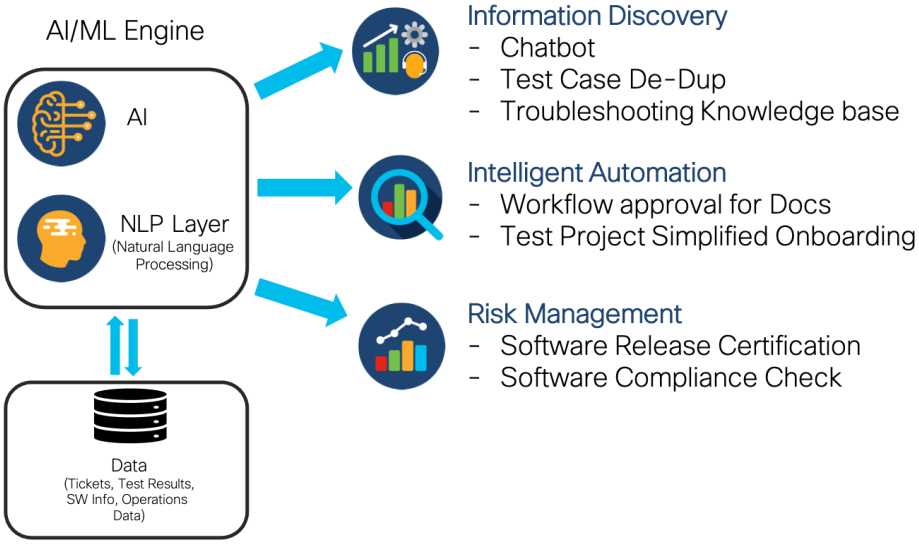

Figure 3.
AI/ML Framework for DevOps Automation

Information Discovery

Any reasonably complex environment poses the ever-present challenge of information discovery. In a cloud-native environment, the complexity is compounded by the participation of multiple vendors, their technologies and associated artifacts. The implications of such complexity are explained below.

Demands on Human Time

In a multivendor environment which has thousands of staff, with roles of developers, testers, release managers, executives, and so on, it becomes very expensive to provide support for resolving issues in the Development and Production environments. A traditional approach to the problem of lack of knowledge or the ability to discover existing knowledge is to engage with humans who may possess this knowledge. A subtlety often overlooked is identifying the person who has this knowledge leading to handovers from one person to another. A handover often results in loss of information. Providing quick and easy access to information for well understood scenarios and engaging with humans for new and complex scenarios is an approach that can reduce the traditional high touch approach.

Vertical Search Engine

A Cloud-native Architecture with a multi-vendor environment has unique datasets around protocols, product naming, technical capabilities etc. An ML Engine needs to be built specific to this environment in what is sometimes called as a Vertical Search Engine. The ML engine should have the following capabilities backed by secure APIs that can be integrated programmatically.

- **Data Management.** The core of any ML capability is the data. In the context of data management, the ML engine should have:
  - Standardized and flexible interfaces for ingesting data from a variety of environments.
  - Data processing infrastructure to author processing and training workflows with flexible topologies and reliable execution.
  - The ability to scale operations to very large data sets to ensure that algorithms developed will work at production scale while simultaneously supporting low latency for serving purposes.
- **Relevance Engine.** A core capability underlying several use cases is computing relevance of objects in a search space to a query object. The relevance engine should support:
  - Flexible and complex query representations, from simple free text queries to structured queries with different components, filters, user feedback, etc.
  - A flexible ranking framework that allows the deployment of a variety of models and signals for ranking the candidates.
- **Aggregate Analysis.** Several operations on the data are of an aggregate nature, rather than per object (test case, document, etc.), and we need a flexible and scalable infrastructure to support such analyses.
  - **Duplicate Detection:** To ensure high fidelity of the data used by the ML engine, we weed out near-duplicates in the data. The presence of near-duplicate objects leads to bad user experience, and also confounds other analysis on the objects.
  - **Statistical analysis:** A variety of statistical computations are performed on objects to estimate various properties of the objects, such as usage, centrality, PageRank, summary statistics, and similar, to improve the relevance of results.
- **Conversational Engine.** Enabling users to self-solve problems and queries is one of the top automation use cases to reduce support costs while improving user satisfaction, and conversational agents or chatbots are rapidly becoming the technology of choice for these use cases. Chatbot solutions should include the following capabilities:
  - Flexible infrastructure to quickly define new intents, workflows, responses, and actions and train the engine to accurately identify intents and follow workflows.
  - Pluggable backend, with API access to allow easy integration of a variety of capabilities, from simple database operations to complex relevance and feedback use cases.

- **Feedback Loop.** It is very rare to find the ideal data set for a particular task, and the usual compromise is to bootstrap using related data sets or rule-based systems. Once a feature is deployed, the data from the usage of the feature yields valuable insights that help to improve the feature and can also be used to obtain high-quality data for training/tuning the models. In order to enable this, the ML engine should have:
    - Mechanisms for the capture and long term storage of different kinds of user feedback, both implicit and explicit, at scale. This includes search behavior, chatbot conversations, etc.
    - Mechanisms to automatically transform feedback data into training data for ML models, and to continuously retrain/tune models and deploy to production.

We have developed an ML engine with the capabilities described above and have configured and integrated it into the software ecosystem of the service provider. A single custom built search engine can be applied for multiple outcomes. Some of the outcomes we were able to leverage are discussed below.

Chatbot

An alternate approach to the classic "search using keywords" is an ML based conversation engine that can help in multiple ways as depicted in Figure 4. Frequently asked questions identified based on the analysis of the repository of past Service Requests (SRs) and conversations on interactive communication channels can be codified. Questions that do not fall into this category can be converted into a classic search by extracting the key aspects of the topic. If the resulting set of documents do not help with the resolution, an escalation path to a human is established through the creation of a service request. The entire context of the question along with all actions taken by the user are captured in the service request to assist the human. The service request identifier is shared with the user for tracking purposes and any further engagement with the human agent.

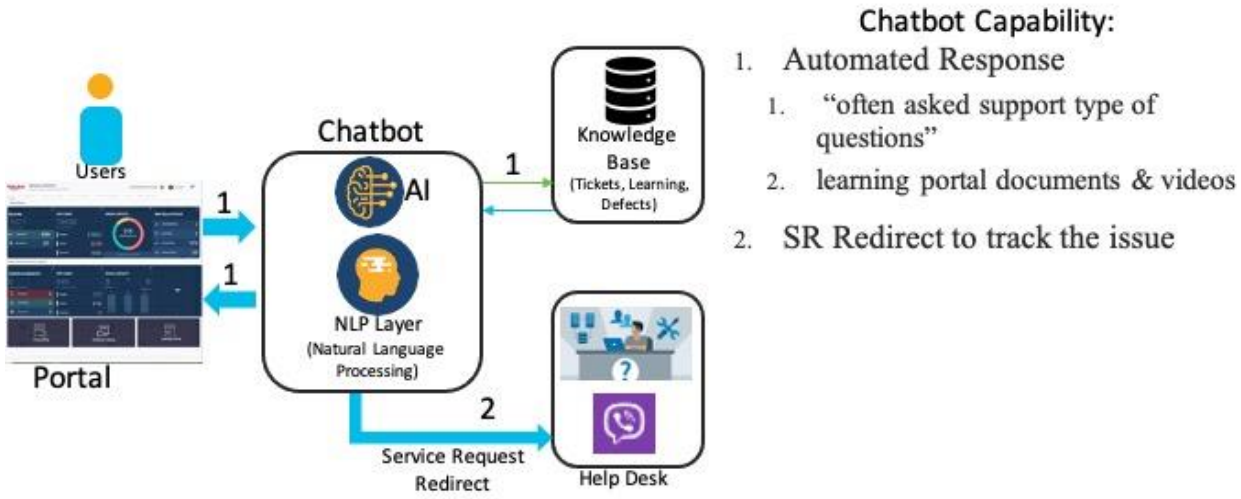

Figure 4.
An Example AI/ML Chatbot

Test Case De-Duplication

Test cases that are used for testing a particular software could be written by multiple people. A test project consisting of a unique set of test cases aims to test if a product works as intended by the user. If there are duplicate test cases in a test project, the effectiveness of the test project is greatly reduced. Test case de-duplication helps with removing these duplicates and keeping the data set unique. If individual users are asked to find and remove duplicates, it will become a tedious and error prone task for them. As test cases are added periodically, the ML engine scans the test cases regularly to identify and help remove duplicates greatly improving the efficiency of testing. In addition, potential duplicate test cases could be suggested at the time of test case creation to avoid the duplication.

Troubleshooting Knowledge Base

An efficient interface to a Knowledge Base of Build Environment, Test Data, Issues and Defects, Features and Functions, owners and stakeholders with full search and suggestion capability is required in any environment. In a multi-vendor environment, the need is heightened due to a large variety of document types and formats, often with disparate structures and language characteristics, and user heterogeneity in terms of roles, goals, behavior, cultures, and so on. Such an interface could be in the form of a 'Chatbot' or 'Web interface' to search the knowledge base with suggestions. The results have to be ranked and perhaps personalized. This capability is very similar to that of the 'search engines' provided by web companies and greatly improves efficiency by providing accurate information from multiple trusted data sources. As an added benefit, in the future, Operations teams can proactively identify issues by leveraging such a functionality.

Intelligent Automation

DevOps implementations require many functional groups working together to develop and deploy the complex Cloud Native based software solutions. Multiple complex processes are implemented as part of various standards like ITIL and TOGAF to meet the business requirements. Automating these processes are critical for a successful outcome. We will show the examples of how the AI/ML search engine was leveraged to improve the efficiency of a complex process without compromising the quality aspect.

Test Project Onboarding

Another example of AI/ML playing a key role in DevOps automation is for the Testing process. Testing a software solution is probably the most important phase before software deployment in DevOps. It requires unique skills and can be a costly exercise in the absence of proper planning.

With a minimal yet more representative number of high-quality test cases obtained by de-duplication; it is possible to automatically onboard a test project that is attached to a release request. The purpose of this stage is to plan and execute test cases to assess the release on all parameters that are important to the release manager. To have a **high-quality** release, it is important that **all** relevant test cases are executed. To have a **fast** release cycle, it is important that **only** relevant test cases are executed. We have designed AI/ML-based test case search and ranking mechanisms to accelerate this stage. In the system there are about 70 thousand cases, say the user wants to know the test cases that deal with "BGP summary", there are only 65 of them across vendors. The AI/ML test case search helps in finding and displaying them.

Workflow Approval for Release Certification

The evolution of Automation for processes has evolved from Robotic Process Automation to Intelligent Automation as depicted in Figure 5. There are numerous approvers in the process making it very tedious and time consuming. These approvals of compliance data and corresponding documents are required as part of a rigorous compliance standards Service Providers and large Enterprises are held to. ML based workflow approval relies on NLP to check the data provided by the vendors. This eliminates the need for human approvers after a period of supervised learning.

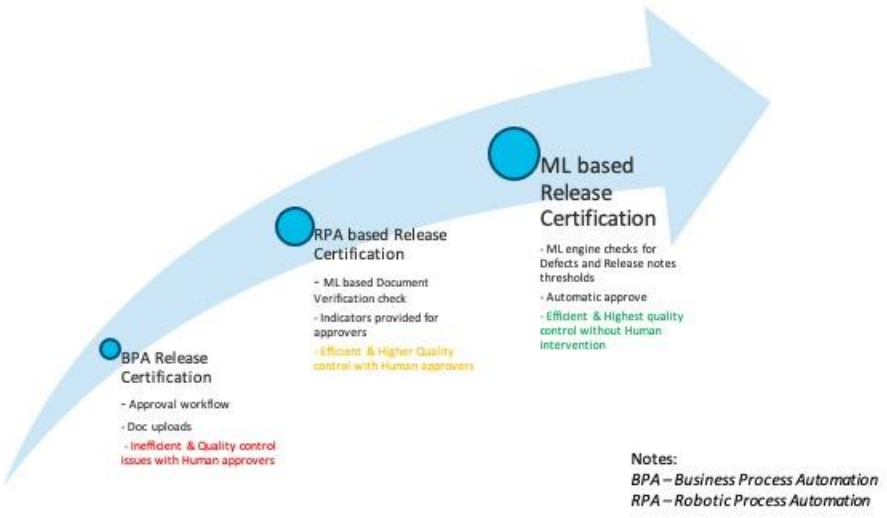

Figure 5.
Evolution of Process Automation

Risk Management through Insights and Collaboration

Artificial Intelligence for IT Operations (AIOps) is about streamlining and automating IT monitoring. AIOps helps IT Ops teams regain control over IT Production environments, quickly detect and repair problems, and prevent major outages. Even though this paper focused on DevOps, many of the AI based use cases apply for IT production environment in a predictive way.

As we enter into IT production environments, AI/ML powered DevOps assists in reducing the risk through insights and collaboration. We cover two examples, namely, Software Release Certification and Software Compliance. These examples are illustrative of autonomous DevOps, where we can implement a loopback system between Development and Production environments.

Software Release Certification

> "Program testing can be used to show the presence of bugs, but never to show their absence!"
>
> **– Edsger W. Dijkstra** [2]

The entire testing process is a simulation of the real-world usage of the system with the stated goal of eliminating if not minimizing failures in production. When such failures aka incidents occur, it's critical to fold that feedback into the testing process. Closing this loop brings the testing process closer to the real world and further advances the stated goal. Software Release Certification relies on Site Reliability Engineering (SRE) Loopback.

In the operational world, all known incidents are recorded for the dual purposes of tracking and analysis (sometimes called retrospectives or post-mortems). The details of the incident are captured in incident management systems with descriptions of the failure, steps taken to diagnose/remedy and the root cause analysis. Along with such descriptions, the telemetry and logs are also attached.

The testing world has a lot of parallels to the operational world in terms of information capture. Documenting testing scenarios involves test case descriptions, steps to reproduce the problem and root cause analysis. System and application logs are attached to complement the textual descriptions. Using AI/ML, test cases that are similar to incidents in production can be grouped and categorized allowing the test engineers to validate and enhance existing test cases. Incidents that are not similar or part of any known testing clusters can then be used to create new testing scenarios.

The telemetry and configurations from the production systems can add tremendous value to the tuning and benchmarking of releases prior to a production rollout. In addition, regressions from an established baseline can be detected well before discovering them in the real world.

Software Compliance

Software compliance refers to how well an application obeys the rules in a standard. It is an audit process. Many of your organizations have to comply with certain standards in order to run your business. In an environment with heterogeneous vendors, it is an extremely complex task to make vendors meet the standards, or your organization meeting the vendor-initiated audits.

An example of vendor audit compliance is license and contracts entitlement. In addition, this can evolve into a license optimization that can help in contract renegotiations and renewals. The AI/ML engine could provide a comparison of the vendor's software recommendation to your production deployment by extracting the key attributes of the configuration from trusted sources of information specified by the vendor. Efficient use of licenses is a classic optimization problem well suited for statistical machine learning. Using the duration of active use of licenses, the category of usage and the time of usage, ML models can suggest schedules that optimize usage.

Another example is Security compliance as part of DevSecOps. Whether it is API security or container security, the compliance data from vendors have to be extracted and compared with production implementation. A collection of all the data viewed with real-time Analytics and deploying our compliance AI/ML engine helps in identifying the risks proactively.

Case Study

We implemented AI/ML Powered DevOps for a greenfield advanced 4G/5G Service Provider in Japan. The CTO's expectation was that the business outcome from DevOps automation would result in cost savings in product testing certification without compromising quality. In essence the two clear objectives are the following.

1. Improved Code Quality
2. Accelerated Code Releases

In this setting, we can broadly think of two categories of problems. Some are "Potential Issues" that **can arise** in production. Potential Issues are captured in **Release Notes** as Known Issues and as **Defects** during the software development lifecycle. Defects are created when a software fails in test environment while running test case(s) and may become Potential Issues in production. On the other hand, **expressed** problems are those that **have happened** in production. These are usually described in **tickets**. Tickets are issues that are created when a software fails in production.

Decision Points

An analysis of the release process at the provider to understand the dependencies and bottlenecks revealed three important stages where a high-quality decision helps to improve both quality and speed.

- **Release** request creation

  Accuracy of Artifacts provided with software releases increased drastically with AI/ML Document check. The quality of these documents(artifacts) improved with the AI/ML check. This also helped in more accurate and faster release certifications. Many releases did not have a release manager, or they were not qualified to review all these artifacts.

- **Test project creation and execution**

  AI/ML based Project creation helped in reusing the projects previously created for similar software certification. In addition, the ML selected better and higher quality test cases for execution.

- **Release Certification**

  AI/ML based correlation between various datasets that were collected helped in accurate and faster release certifications for example, pass/failure counts in test results versus defects found in production etc.

KPIs

In a 6-month time frame of implementing AI/ML capabilities, the following KPIs were captured.

Release KPIs

1. 75+ vendors with 350+ Software packages.
2. Number of releases: 950+ releases addressed and 400 certified for deployment.
3. Compliance: More than 75% release-certified software deployed in production environments.

AI/ML specific KPIs

AI/ML specific KPIs that were achieved in this time frame are listed below.

1. AI/ML Duplicate Reduction: More than 60% of the Test Cases were deprioritized from databases.
2. AI/ML Chatbot: Provides support for over 2500+ users in case creation and resolution.
3. AI/ML Automated Test Project creation reduced a 12-step selection process to a 2-step process.
4. AI/ML-based Document Check: Reduces cognitive effort during the release process by auto-checking and summarizing documents for the reviewer; on average, there are 7 documents per release request.

Conclusion

With a SaaS based AI/ML Powered DevOps Application, customers can accelerate their innovation by rapidly deploying complex heterogeneous vendor solutions. AI/ML enables easy access to trusted information, provides actionable insights while mitigating risk allowing Service Provider and Enterprise customers to be in compliance. Customers can begin their AI/ML usage at any stage of the DevOps automation journey they are on currently. A secure cloud based application provides the benefit of enabling customers to tap into learnings across a wide variety of critical infrastructure deployments. The network effect of such learning compounds over time.

Authors

Vijayakumar Raghavendran, Distinguished Engineer, CX CTO, Cisco Systems Inc

Ranjani Ram, Principal Engineer, CX Center, Cisco Systems Inc

Santhosh Srinivasan, Co-Founder and VP of Engineering, Peritus.AI, Inc.

Goutham Tholpadi, Data Scientist, Peritus.AI, Inc.

Contributors:

Nathan Sowtskey, Principal Engineer, CX CTO, Cisco Systems Inc

Arun Arunachalam, Principal Engineer, CX Center, Cisco Systems Inc

Robin Purohit, CEO, Peritus.AI, Inc

Ahmed Khattab, Principal Architect, CX Engineering, Cisco Systems Inc

References

1. An Efficient Software Delivery Framework for 5G Environment by Vijay Raghavendran (https://www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/cto-cdaf.pdf)
2. Notes on Structured Programming by Prof.dr. Edsger W Dijkstra (https://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF)

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.