WHITE PAPER

CI CD FOR CCM PLATFORMS

When it comes to the latest market trending technology and process adoptions like Cloud, Artificial Intelligence, Agile and Lean based Software Development; Customer Communication Management (CCM) platforms must take a different route based on various assessments, as the platform's way of software development is niche as compared to the conventional programmatic software development. This is no different for Continuous Integration / Continuous Deployment or Delivery (CI / CD) implementation in an Agile based Software Development for a CCM platform. This paper provides the technique (tailored for CCM) to implement CI / CD orchestration in CCM space.

Infosys®
Navigate your next

Introduction

Continuous Integration and Continuous Deployment / Delivery (CI / CD) is a technique or process of automating various phases in an Agile SDLC for an early and resilient software release with an effort of Combined Engineering. The conventional adoption technique of CI / CD is applicable for projects having various activities like coding, creating build, packaging, code quality check, deploying etc.

Customer Communication Management (CCM) is a platform to Create and manage Document / Communication Generation, and Distribution and Interactions via Multi / Omni-channel offering. These platforms have design tool to create Business Document Template (BDT), Admin or Manage Component to package the BDT into CCM product's proprietary file formats.

Creating Build, Code Analysis and Quality checks can't be performed on these proprietary packages and Templates by the traditional DevOps / CI tools as they won't support or be compatible with them. This makes Automated CI orchestration difficult in CCM space.

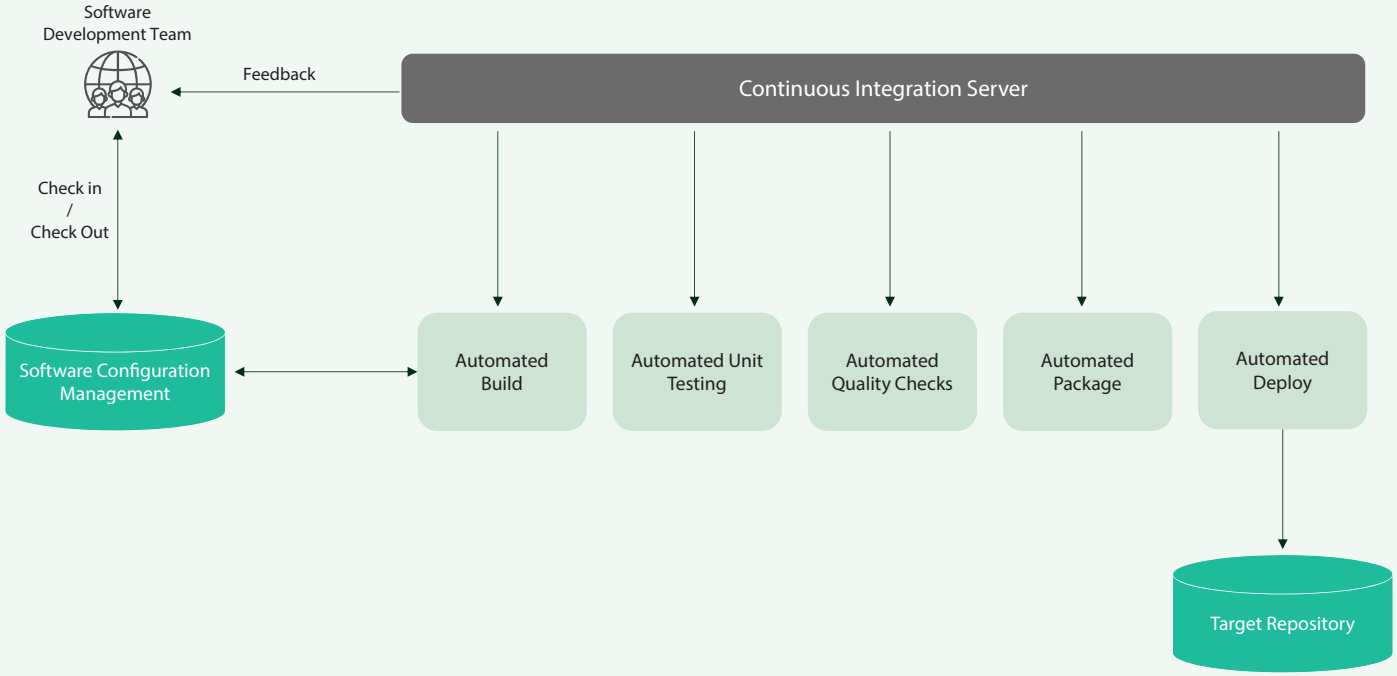
CI & CD workflow – Ideal vs. CCM

In typical IT projects the various source code components developed by various developers are stored, maintained and versioned in repositories (centralized or distributed). These repositories become the holistic place for the source code to be version controlled and made ready for the 'Build' as the next stage in Continuous Integration Workflow.

CI tools like Jenkins, Hudson are employed to automate the workflow of continuous integration starting from:

- Polling the software configuration management database
- When changes identified in source code, triggering build and executing the test cases
- Perform quality check, package and deploy to target repository
- Sends feedback to the Development team at any point in the workflow (based on the configuration)

Here is a representation of typical automated CI workflow:



Though source control management repositories are used in CCM projects, they haven't been leveraged for a CI / CD workflow, this is because the CCM development journey cannot fit into an ideal CI / CD workflow and in this document, we can see how these repositories can be utilized for CCM project with a tailored CI / CD workflow and methodologies.

CI and CD workflow in CCM Space

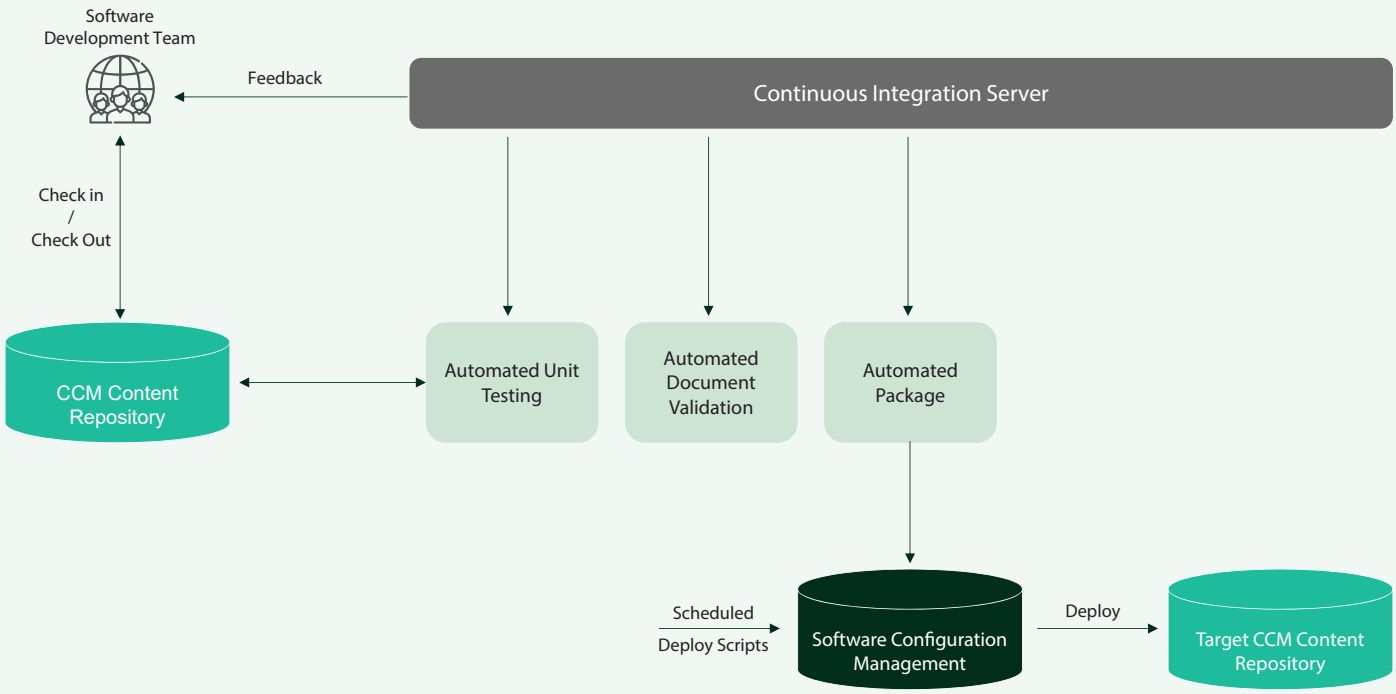
Considering the above, a different paradigm needs to be considered for CI/ CD adoption for CCM projects. Let's see a possible CI/CD automation workflow in a CCM portfolio.

To start with the SCM repository, check in and check out of code into SCM may not be required at the initial stage of the CI / CD workflow. All mature CCM platforms have their own Content repository (CR - Database for CCM) where they store all the information of the environment like BDTs developed, their versions and metadata. Along with these they also hold other information like the configuration of data source, user management and batch jobs.

In a template development, once the BDT is complete, they have their own workflow for approval of the BDT post where document versioning will be done. A DB (Database) monitoring can be employed to monitor the CCM Content Repository for a specific table which holds the information on BDT versions. Once updates are monitored on this table, hooking technique can be used to notify the other systems on the update or a periodic polling can be done from other systems to check for any updates done in the DB.

After this step with available test data, unit testing can be performed to ensure the compilation (Document Assembly) and execution (Document Publish) happens without returning any errors. As a next step, document validation can be executed by document comparison tools which could send report on the output discrepancies based on its comparison with the expected output. There can be feedback (report) send to developers for an approval to perform the next step or if there are zero discrepancies the next step can be automatically initiated.

In this step, CCM scripts can be employed to export the BDT and upload to SCM repository and scripts can be employed to import the package to target CCM repository and these can be triggered based on event schedules.



Overcoming CI / CD Challenges in CCM space

Having the workflow described, there could be challenges involved in bringing in the automation tools in the various stages. As the typical IT automation tools cannot be utilized for unit testing, packaging and other activities, so custom tools must be developed suiting for the CCM platform utilized, and the below stages in CI / CD will be addressed with such custom tools.

Unit Testing (Proofing) - Utility can be built by consuming the CCM platform's API which publishes the output document. Publishing the document ensures there is no error in compilation and execution of the BDT.

Document Validation - Utility can be built from scratch or use SDKs of mature document comparison software which could generate reports. These report captures the discrepancies in the actual output from expected output.

Packaging and Build - Extended script of CCM Migration scripts can be used to export the document as a package.

This stage could be before unit testing too which depends on the type of CCM platform.

One example is OpenText Exstream were unit testing is done post creating the package.

Deployment - Similar like packaging script extended importing scripts will allow to import the package into target CCM environment.

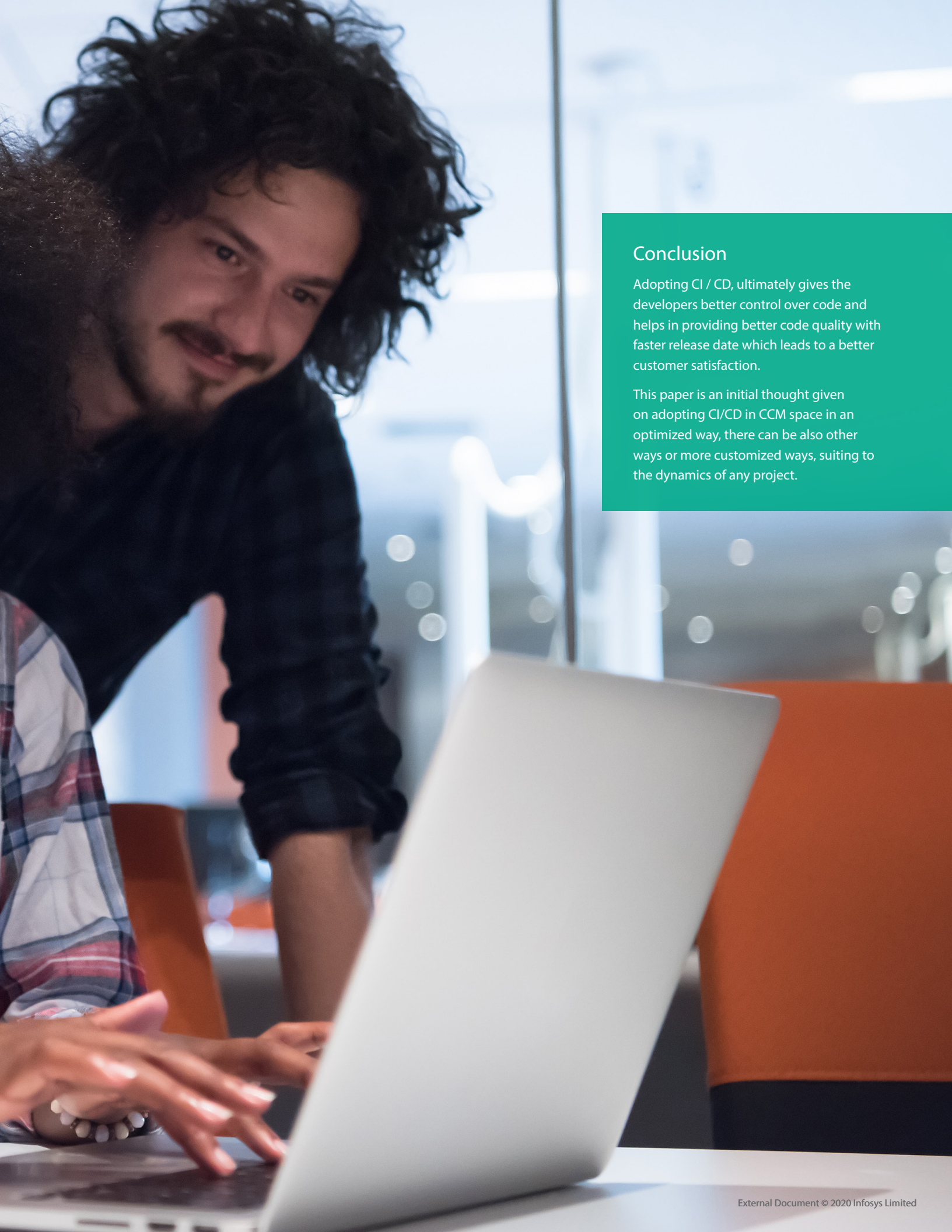
Opportunities / Benefits of CI - CD in CCM

1. Quick time to market
2. Increased team transparency and accountability
3. Smaller code changes - Integrating various small pieces of code (minor changes in various BDTs) at one time which helps to maintain business as usual without disrupting the existing code
4. Fault isolations - Fault identified BDTs can be isolated and specifically reverted
5. Easy maintenance and updates

Best practices

Below are some practices that can be followed to achieve a better CI / CD orchestration:

1. Approval workflow and versioning of BDT can be effectively utilized in CCM products, having steps like code review on functional and performance before approval
2. Packaging of BDT can be done for individual BDTs instead of wrapping multiple BDTs into one package. Having individual packages in SCM allows us to track and compare source code for complex projects and allows us to revert changes of specific BDT instead of reverting the code of the overall release / sprint.



Conclusion

Adopting CI / CD, ultimately gives the developers better control over code and helps in providing better code quality with faster release date which leads to a better customer satisfaction.

This paper is an initial thought given on adopting CI/CD in CCM space in an optimized way, there can be also other ways or more customized ways, suiting to the dynamics of any project.

About the Author

Diwakar Subramani
CCM Consultant - Digital Experience, Infosys

A CCM Consultant with the Digital Experience unit of Infosys, Diwakar Subramani has around six years of experience in Information technology. He has extensive experience in analysis, design and implementation of Customer Communication Management solutions. He holds a Post-Graduation in Data Science from International Institute of Information and Technology, Bangalore, India.

For more information, contact askus@infosys.com

Infosys®
Navigate your next