ADDTEQ

About Us

Addteq's aim is to support software development teams with right technologies, processes and resources to Release software frequently. We bring-in end to end solutions to manage their Software Change and Configuration Management (SCCM) needs. We center our products and services around this fundamentally unique approach towards Software Release Management.

Our key offerings:

1. Atlassian Products
2. Cloud hosting for all your Software development applications
3. Services
   - Process consulting in Agile Continuous Delivery
   - Build, Release & deployment automation
   - Implementation Services around Atlassian products
   - Migration of source code repositories
   - Integration of systems from different vendors

1-888-9ADDTEQ

facebook.com/addteq

twitter.com/Addteq

www.addteq.com

CaseStudy

System integration, SCM best practices and automation for huge productivity gains.

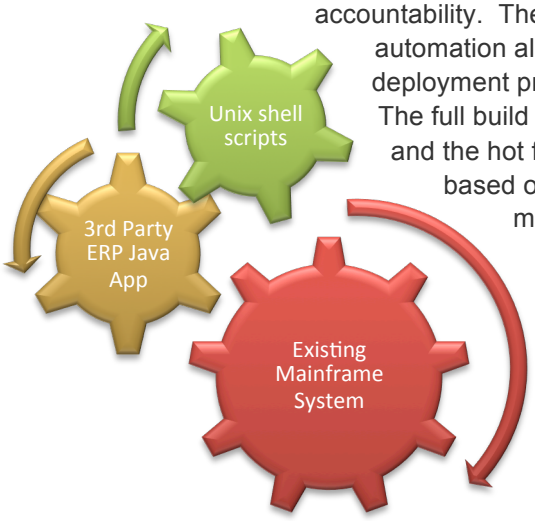
Company

The US government entity, responsible for managing financial information of all the government agencies of the state

Challenge

This client received a pre-built enterprise resource planning (ERP) application from another vendor, developed and built at the vendor site. This new ERP application was a Java based application which need to work along with their old mainframe software. The new Java based application would enable information to be submitted online and be automatically inserted in the database rather than having to be manually submitted. The ERP software was packaged as EAR files, had a database component to it, and had Unix shell scripting modules which were developed at the client site and packaged differently than the EAR files.

Their development solution did not involve automating the build and deployment process. There were frequent issues of failed builds on the manual deployment process as it lacked strict SCM policies that promote accountability. The absence of deployment automation also unnecessarily lengthened deployment process and increased costs. The full build deployment took over 8 hours and the hot fix migration took 3-4 hours based on the number of hot fix to be migrated.



The biggest challenge the organization was facing was to get their development teams to adapt to the new technologies. The team was occupied with the daily product issues and couldn't manage to drive change or learn new skills to solve their build and deployment challenges. As it went live it

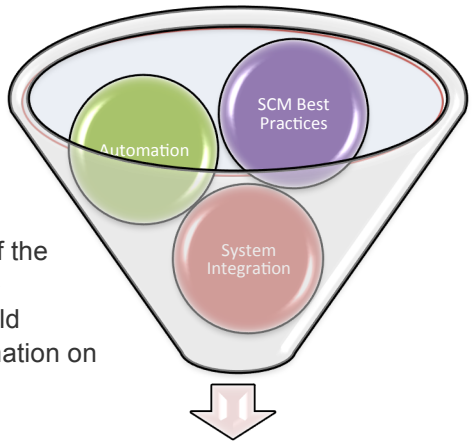
failed and for 6 months the project was "dead in the water" due to these issues.

Approach

We divided the migration task into UNIX scripts migrationand Java code migration.As UNIX script migrations benefits were immediate, idea was to automate them first, in order to free up a majority of employees so they could focus on the automation of the Java code migration.

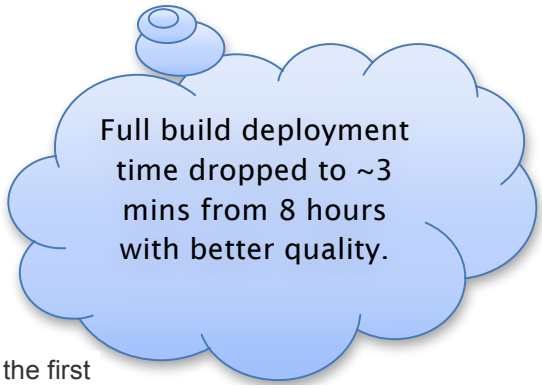
Solution

We decided to number every build; every package delivered by the vendor would go through this numbering and messaging system for the code. The build would then be moved to the build repository (SFTP server) dedicated to store the "ready to deploy" builds. The meta-data of the build along with the build number was stored in the CM database for easy retrieval of information without going through multiple build locations. This was also extremely resourceful while debugging build failures. This also helped in linking the builds directly on the CM website through the database information.



Automation assisted in simplifying the deployment script. The location of the package to be extracted depended on the application name and module name. Based on the information passed to the deployment script, it would automatically place the content in the correct location and update information on CM website along with the other meta-data like:

- Who ran the deployment
- Automation time
- Number of files involved
- Affected database pieces
- Affected environment variables
- Information on Servers that need to act to complete the deployment



Outcome

The error rate of deployments was brought down to 0.001% from 60% within the first 6 months of implementation. The deployment time for hotfix migration was dropped from 3 hours to 15 seconds. The deployment time of full build was dropped from 8 hours to 3 minutes and 15 seconds. Effectively, the development teams were able to focus on areas where they are skilled and overall productivity of the department increased drastically.