CI ツールの導入による
開発生産性と品質の向上

東京海上日動システムズ（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

古澤　直人

2014 年　東京海上日動システムズ（株）入社
アプリケーション開発部 所属

■ 論文要旨 ■

筆者は、エンジニアとして保険商品を扱う契約管理システムの内製開発を担当している。システム開発では手作業による定例的な作業が多く、人的ロードとコストが掛かる。筆者が担当するプロジェクトにおいても大量のプログラム資源反映、テスト環境サーバの再起動など、多くの手作業がある。手作業プロセスを自動化することで、ロード削減や人的ミスを防止できると考え、プロジェクトへの継続的インテグレーション（Continuous Integration、以下 CI と略）ツールを導入した。この結果、開発生産性の向上と、環境セットアップにおける障害削減により、開発品質を向上することができた。また、ツールの導入は３年目以下の若手メンバー４人で、導入検討・サーバ調達・ジョブ構築・導入作業までのすべてを担当した。本稿では、自ら設計・開発・テストまで行うことで気付けた、CI ツール導入における期待と効果、および導入時のポイントについて論ずる。

■ 論文目次 ■

１．はじめに ……《 3》
１．１　当社の概要とアプリケーション開発部について
１．２　SP システム開発プロジェクトの概要

２．システムの開発プロセスの一般的な問題点 ……《 3》
２．１　CI の概要
２．２　CI ツール「Jenkins」の概要

３．SP システム開発における開発プロセスと問題点 ……《 4》
３．１　SP システムの開発・配備プロセス
３．２　SP システムⅡ期の開発・配備プロセスの問題点と分析

４．CI ツールの導入と効果 ……《 6》
４．１　SP システムⅢ期開発における Jenkins のジョブ構成
４．２　導入後の効果

５．おわりに ……《 8》
５．１　CI ツール導入時のポイント
５．２　今後の展望
５．３　所感

■ 図表一覧 ■

図１　Jenkins画面イメージ図 ……《 4》
図２　SPシステムⅡ期開発の作業フロー ……《 4》
図３　SPシステムジョブ構成図 ……《 6》
図４　CI構築作業と手作業による作業工数累計の比較 ……《 7》
図５　テスト環境トラブル件数の比較 ……《 8》

表１　SPシステムⅡ期開発の作業内容と作業時間 ……《 5》
表２　SPシステムⅡ期開発の作業フローの問題点 ……《 5》

１．はじめに

１．１　当社の概要とアプリケーション開発部について

東京海上日動システムズ株式会社（以下、弊社とする）は 1983 年９月に設立され 10 本部 30 部から成り、東京海上グループの情報システムの開発・保守・運用を担う。所属するアプリケーション開発部は「高いシステム開発力を背景とした新技術の展開」と「内製開発を通してシステムの内部に踏み込んだ提案・設計ができる人材を育成する」ことを目的に、内製開発を担当している。

１．２　SP システム開発プロジェクトの概要

SP システムとは、自動車保険などとは異なる、少量多品種の保険商品を対象とした契約管理システムである。金融系特有のメインフレームからの脱却を図り、オープン系で開発している。SP システムの開発は保険商品ごとに５段階に分けて、順次サービスインを開始している。本取り組みはⅢ期開発における結合テスト工程から導入したものである。Ⅲ期時点での開発規模は 130 万ステップ以上で、資源数も 3,000 本以上である。

２．システムの開発プロセスの一般的な問題点

システム開発においては多くの定例作業・手作業が存在する。SP システム開発においても、プログラム資源リリース、テスト環境サーバ再起動など、大規模システムであるが故に多くの作業があり、かつ本格的に自動化・効率化着手はできておらず、作業のほとんどが人手で行われていた。担当者が作業に大きな工数を掛けてしまっていること、人的ミスが頻発していることなど、たくさんの課題がある中で、抜本的な対策として「手作業のプロセスそのものを自動化する」ことこそが最善と判断し、継続的インテグレーション（Continuous Integration、以下 CI と略）ツール導入の検討を始めた。

２．１　CI の概要

CI とは、システム開発における作業（ソフトウェアの結合、テスト環境へのリリース、テストサーバの再起動・監視など）を自動化するツールのこと。自動化することで、人手を介すことなく１日に何度も、再帰的に作業を実行できるようになる。CI を導入することで、プログラム資源の修正・変更によって発生したビルドエラーや、サーバ再起動時のエラーなどを早期に発見でき、エラーがどの変更によって引き起こされたのかを特定しやすくなる。結果としてシステムの品質向上や手戻り工数の削減が期待できる。

２．２　CI ツール「Jenkins」の概要

本プロジェクトにおいては、自動化の実現に向けて Jenkins（図１参照）を導入した。Jenkins とはオープンソースの CI ツールの一つで、「ビルド」「デプロイ」「サーバ再起動」「テスト実行」などの一連の作業をジョブとしてスケジューリングして自動化できるツールである。Jenkins を採用した理由は、以下の３点である。[1]

①オープンソースで提供されており、安価に導入できる

②実績が豊富で、構築に必要な情報を得やすい

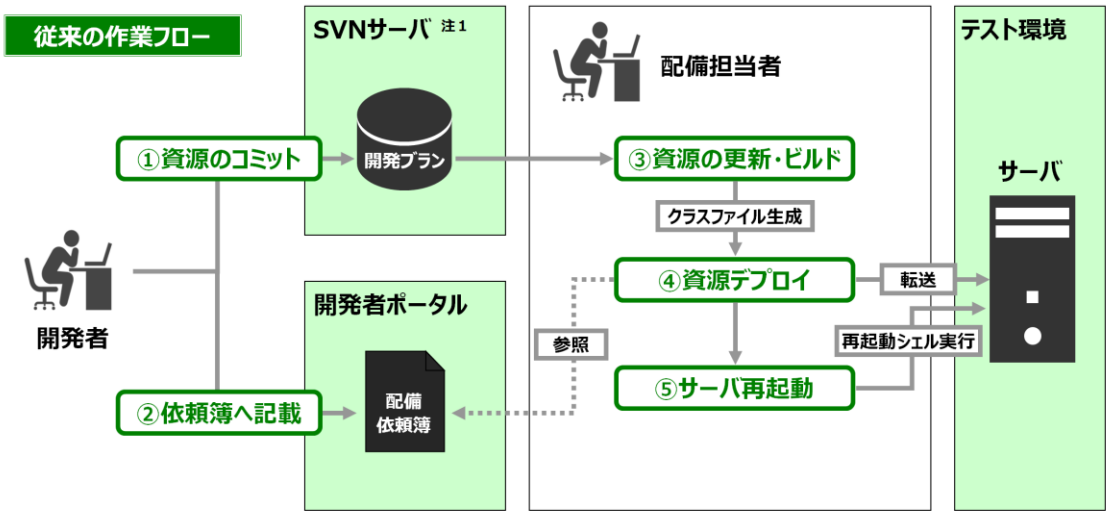
③プラグインの開発が活発で、必要な機能を個別に追加できる

図１　Jenkins 画面イメージ図

３．SP システム開発における開発プロセスと問題点

３．１　SP システムの開発・配備プロセス

SP システム開発における従来の開発者および配備担当者の作業フローを図２に示す。



注１．SVN（Apache Subversion）とは、プログラム資源を構成管理するためのソフトウェアである。

図２　SP システムⅡ期開発の作業フロー

SP システムでは、ビルドによって生成されたクラスファイルを個別にサーバへ配備している。プログラム資源が修正された場合は差分のある資源のみを再ビルドし、サーバへデプロイする。開発者と配備担当者の作業内容と作業時間は表１のとおりである。

担当	作業名	作業内容	作業時間
開発者	①資源のコミット	作成、修正したプログラム資源をSVN上の開発ブランチにコミットする。	数分（1資源あたり）
	②依頼簿への記入	配備希望資源名と配備先サーバパスを開発者自身がポータル上の配備依頼簿に記載する。	数分（1資源あたり）
配備担当者	③資源の更新・ビルド	開発環境上の資源を更新・ビルドし、クラスファイルを生成する。	1時間（1日あたり）
	④資源デプロイ	配備依頼簿を参照し、対象資源をテスト環境サーバへ1ファイルごとにFTP転送する。多い日には100資源以上になることもある。	2時間（1日あたり）
	⑤サーバ再起動	サーバ上の再起動用シェルスクリプトを一つずつ順番に実行し、ログを監視する。対象のサーバは9個あり、それぞれを1日に数回再起動する	30分（1日あたり）

表１　SP システムⅡ期開発の作業内容と作業時間

３．２　SP システムⅡ期の開発・配備プロセスの問題点と分析

従来の開発・配備プロセスで発生した問題点を整理したところ、表２の結果となった。問題点の多くは「開発者と配備担当者間の認識齟齬」や、「手動で行う際の作業ミス」が原因となっており、CI ツールによる自動化で改善が期待できると分析した。特に「資源の更新・ビルド」「資源デプロイ」は、作業に要する時間が多いこと、開発者への確認が必要なこと、クリティカルなミスが頻発していることなどから、自動化による効果が大きいと考えた。更に「依頼簿への記入」については、すべてを自動化することはできないが、Excel の配備依頼簿を廃止することで記載漏れ、記載ミスがなくなり、作業時間の削減につなげることができると考えた。

担当	作業名	作業実施中に起こった問題点	自動化の効果
開発者	①資源のコミット	依存関係のある資源が全量コミットされておらず、ビルドエラーが発生した。	×
	②依頼簿への記入	配備依頼簿への記載漏れ、記載ミスにより、必要な資源が正しく配備されなかった。	△
配備担当者	③資源の更新・ビルド	週に1度のペースでビルドエラーが発生し、開発担当者への照会が必要となった。担当者が不在の場合には更に多くの時間を要した。	◎
		開発したプログラムを1資源ごとに作業スペースへ引き込む際、引き込み漏れや更新漏れが発生し、正しくプログラムがビルドされなかった。	
		結合テスト環境、システムテスト環境、オーナーテスト環境など、環境のバリエーションが増加した結果、誤って別の環境のプログラムをビルドしてしまった。	
	④資源デプロイ	プログラム資源を一つ一つ手作業で引き上げを行う際に、引き上げ漏れが発生し、正しいプログラムが配備されなかった。	◎
		誤って配備対象ではないプログラムをテスト環境に配備してしまった。（その結果、テストサーバでエラーが発生しシステムトラブルとなった）	
		結合テスト環境に配備予定だったプログラムを、誤ってオーナーテスト環境に配備してしまい、オーナーテスト環境でトラブルが発生した。	
		その日に配備を行ったプログラムをExcelの配備一覧として管理していたが、作業後に一覧を更新する作業を失念してしまった。	
	⑤サーバ再起動	結合テスト環境のサーバを再起動しようとしたところ、誤ってオーナーテスト環境のサーバを再起動してしまい、オーナーテストができない時間が発生した。	○
		サーバの再起動は定期的に行う必要があるが、担当者の急な休暇などで本来の時間に再起動作業が実施されず、テストの際に資源の修正が反映されていなかった。	

表２　SP システムⅡ期開発の作業フローの問題点

４．CI ツールの導入と効果

４．１　SP システムⅢ期開発における Jenkins のジョブ構成

前項で述べた問題点・課題を解決するために SPⅢ期で構築したジョブ構成を図３で示す。テスト環境へ引き上げたい資源を、開発者が配備用 SVN ブランチへコミットし、その後「資源の更新」「ビルド」「資源ごとの配備先サーバパス算出」「資源デプロイ」「サーバ再起動」のすべての作業を Jenkins が自動で実行する。これにより、配備依頼のあった資源を人の手を介さずに、テスト環境へ引き上げることができる。

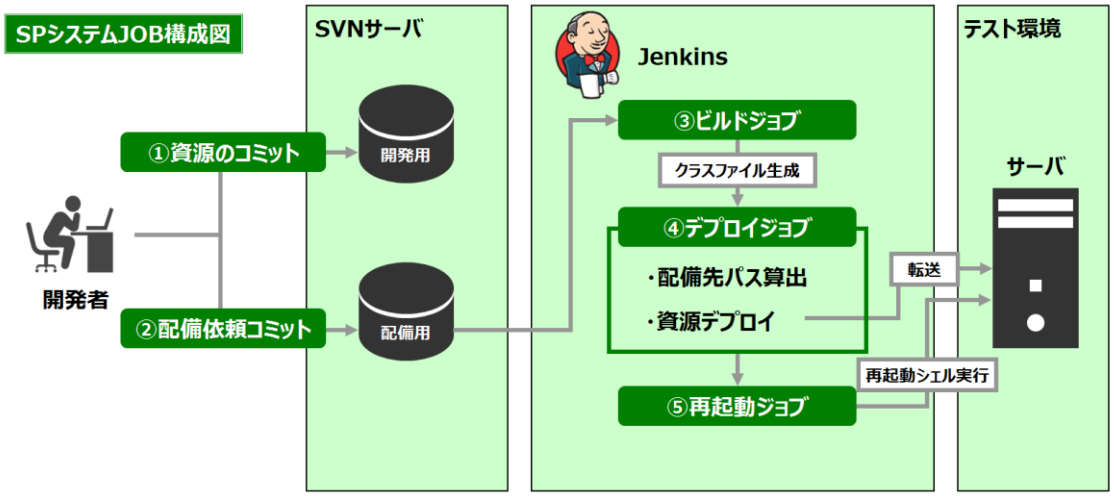


図３　SP システムジョブ構成図

（１）ビルドジョブ

配備用ブランチへコミットされた資源を更新し、ビルドするジョブである。生成されたクラスファイルは、Jenkins 内のワークスペースに格納される。配備資源を生成するジョブ以外にも、単体で１時間おきに実行するビルドジョブも構築した。これにより、配備用ブランチでビルドエラーを常に監視できるようになった。

（２）デプロイジョブ

ビルドジョブによって生成されたクラスファイルを、テスト環境サーバへ FTP 転送するジョブである。コミットログから開発ブランチ上のパスを取得し、配備先サーバ上のパスを算出する。Jenkins のプラグインはクラスファイル単位の配備に対応していないものが多く、また SP システムはサーバ上のディレクトリ構成が複雑なため、デプロイジョブは Java によるスクラッチ開発で作成した。これにより、誤った配備先に引き上げたり、引き上げ自体を忘れたりすることがなくなった。

（３）再起動ジョブ

資源配備が終った後に、SSH でテスト用サーバに接続しシェルスクリプトを自動実行し、サーバを再起動するジョブである。Jenkins の TextFinder プラグインを用いてログを監視し、再起動の成功・失敗を判断する。これにより、予定どおり再起動されるようになった。

３つのジョブにより、表１にある「資源の更新・ビルド」「資源デプロイ」「サーバ再起動」の作業すべてが不要となり、配備担当者はブラウザから各ジョブの実行結果を確認するのみとなった。

４．２　導入後の効果

CI ツールの導入によって配備作業・サーバ再起動作業が自動化され、「定例作業・手作業工数の削減」「テスト環境トラブル件数の削減」の効果が得られた。

（１）定例作業・手作業工数の削減

従来、配備作業は開発者が順に担当し、効率化はしてきたものの、日によっては１日の５割以上を費やしていた。自動化によって工数はほぼゼロとなり、開発者はプログラム開発やテストに集中できるようになった。

図４は、CI ツール導入に掛かった工数と手作業による配備作業に掛かった累計工数を比較したグラフである。初めての導入だったこともあり、各ジョブの初期構築に 12 人日を要したが、導入後は基本的に工数が掛からない。一方で手作業による配備作業を続けていくと、テスト実施日を重ねるごとに累計工数が増加していき、おおよそテスト開始後 24 日時点で累計工数が逆転する。SP システムではテスト工程を５ヶ月（100 営業日）以上実施しているため、非常に大きな工数削減効果がもたらされている。

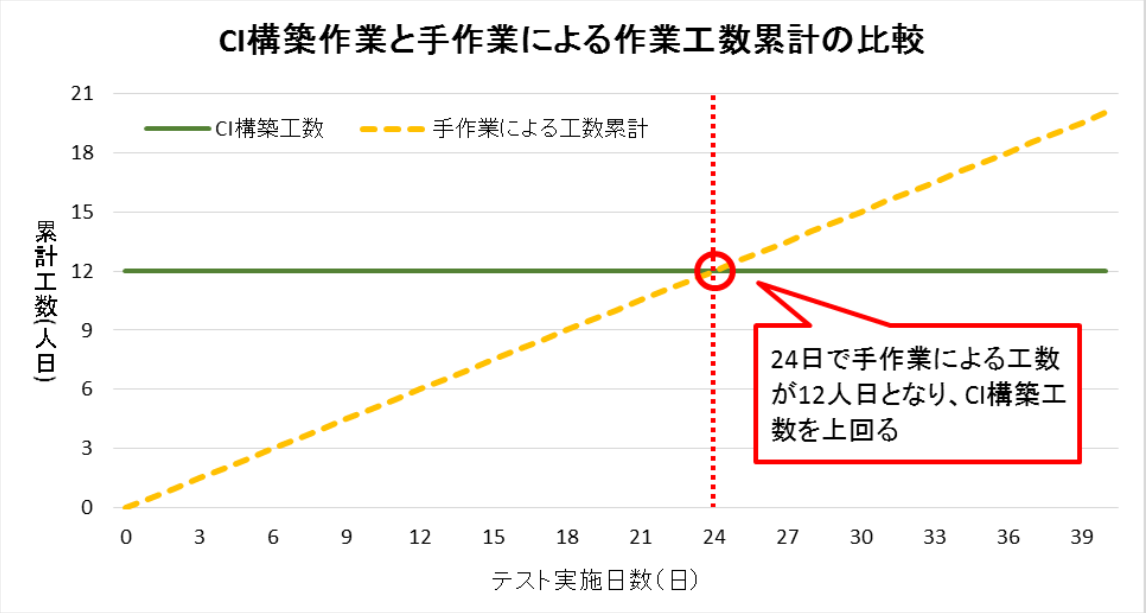


図４　CI 構築作業と手作業による作業工数累計の比較

（２）テスト環境トラブル件数の削減

配備作業から手作業が排除されたことでテスト環境不備によるトラブルの件数が減り、予定していたスケジュールどおりにテストができるようになった。

図５は人的ミスによるシステムトラブル件数の推移である。CI ツール導入前後でトラブル件数が８分の１に減少しており、作業品質面でも非常に大きなメリットをもたらし

ている。なお、CI 導入後３ヶ月間に発生した２件は、初めて取り組んで、ノウハウもない中で生じたものであり、ツールを改修した以降にトラブルは発生していない。

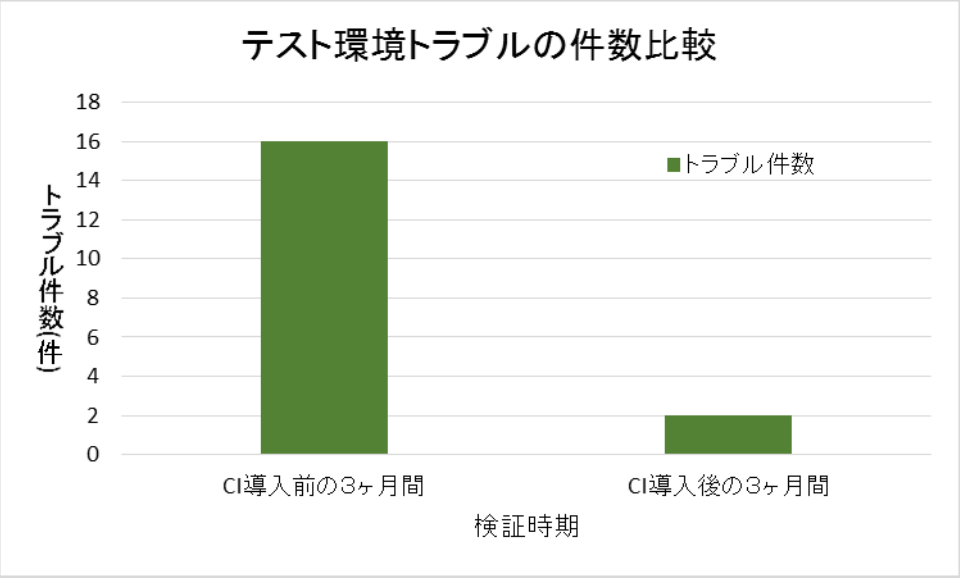


図５ テスト環境トラブル件数の比較

５．おわりに

５．１　CI ツール導入時のポイント

今回 SP システムでは、Ⅲ期開発からの導入となったため、既存の開発プロセスに合わせてジョブを構築する必要があった。新規プロジェクトであれば CI ツールを前提とした開発プロセスを組むことができ、導入コストが更に小さくなる。CI ツールは開発初期から導入したほうが効果が大きいといえる。

また、開発途中から導入する場合には、システムの特性やプロジェクトの実情を整理・分析し、適応していく必要がある。SP システムには、クラスファイル単位の資源配備に対応するために、デプロイジョブを既存のプラグインではなく、スクラッチ開発によって実現した。このように、システム固有の特性を個別にカスタマイズして吸収させることで自動化の効果も更に大きくなると考える。このような観点から適切に判断しないと、場合によっては CI ツールの導入前よりもトラブルやエラーが増え、開発者に掛かる負担が増えてしまう恐れがある。そのような事態に陥らないよう、CI ツールを導入する際には対象システムを理解することに加えて、各作業内容を十分に整理したうえでジョブを設計する必要がある。

５．２　今後の展望

今回 SP システム開発に導入し、大きな効果を得た CI ツールを適切に利用するために、判断基準を明記したガイドラインなどを作成し、今後も対象プロジェクトや自動化の対象範囲を拡大していきたいと考えている。

（1）テストの自動実行による更なる品質向上

ビルドからサーバの再起動までの配備作業を今回自動化した。本来であればCIを実現するためにはテストの自動実行まで行う必要があると考えている。今後、jUnit [注2]による単体テストの自動実行・カバレッジ集計や、Selenium[注3]を用いた画面テストの自動実施を実現し、常にアプリケーションが正しく動く状態を維持することで、SP システム開発の品質を更に向上させたい。

（2）他プロジェクトへの展開

SP システムで実現された例からも分かるように、業務を効率化するツールの導入は、工数削減、品質向上の双方で多大なメリットを感じることができた。今後は他プロジェクトも広く導入を検討すべきだと考えている。導入に向けて、プロジェクトごとに異なる特性を整理し、何を、どのように自動化するのが最適かを分析し、コンサルティングする専門のチームを結成して、CI ツールを始めとした業務効率化ツールの全社展開を実現していきたい。

５．３　所感

今回のCIツール導入は、アプリケーション開発部３年目以下のメンバー４人のみで、かつ、すべてを内製で実現させることができた。若手中心での取り組みだったため苦労する点も多々あったが、業務改善への強い思いと、新たな仕組みへの好奇心、自分たちの手で作り上げる楽しさがあったからこそ達成できたと考えている。今後も、現状に満足することなく技術力を高め、自らが担当するシステムの品質を高めることに貢献していきたい。

また、社内の開発プロセスにはまだまだ多くの手作業が残されており、自動化による効率化の余地が多くあると感じている。自動化すべきものと、自動化すべきではないものを適切に判断し、効果的にツールを設計・導入していくことで、更なる工数削減・品質向上につなげたいと考えている。

参考文献

［1］佐藤聖規、和田貴久、河村雅人、米沢弘樹、山岸啓：“Jenkins実践入門”，技術評論社（2011）

注２．jUnitとは、Javaプログラムの単体テストを行うためのフレームワークである。
注３．SeleniumとはWebのUIテストを自動化するためのツールである。