31SQiP 研究会　特別講義　レポート

作 成 日：　2015 年　12 月 18 日
書記氏名：　齋藤　伸介

日時	2015 年 12 月 18 日（金）　10:00 ～12:00
会場	(一財)日本科学技術連盟・東高円寺ビル　2 階　講堂
テーマ	「積極的なソフトウェア構成管理の活用方法」
講師名・所属	長沢 智治氏（アトラシアン株式会社　エバンジェリスト）
司会者	猪塚　修氏（横河ソリューションサービス株式会社　ソリューション技術本部）
アジェンダ	・ソフトウェア構成管理の概要 －広義と狭義のソフトウェア構成管理 －ソフトウェア構成管理の構成要素 －ソフトウェア構成管理と変更依頼管理、ビルド自動化 ・ソフトウェア構成管理システム －中央集中バージョン管理 －分散バージョン管理 －ソフトウェア構成管理システムの導入のポイント ・期待に応えるソフトウェアのためのソフトウェア構成管理 －ビジネスニーズとソフトウェアの品質の変化 －積極的なソフトウェア構成管理
アブストラクト	ソフトウェア構成管理は、ソフトウェア開発に常に寄り添い、活用されてきました。ソフトウェアへのビジネスニーズの変化と共に、ソフトウェアの品質への期待値も変わり、ソフトウェア開発現場も変わってきました。 今回は、ソフトウェア構成管理をテーマとし、ソフトウェア構成管理の過去、現在を俯瞰的な視点で見ていきながら、構成管理を積極的に活用している現場で行われているプラクティスやブランチ戦略なども見ていきます。

今回の特別講演では「積極的なソフトウェア構成管理の活用方法」と題してアトラシアン株式会社　エバンジェリストの長沢智治さんから、講演を頂きました。

長沢さんは、長く IT 業界においてプロセス改善コンサルタントとして活動され、また「ソフトウェア品質知識体系ガイド」「C#実践開発手法」など、著作も多数執筆されています。

今回の講演では、ソフトウェア構成管理というものが、単なる成果物のバージョン管理というものではなく、ビジネスも目標を達成の為に、積極的に活用していくにはどうするか、という斬新な視点から、非常にわかりやすい講演をしていただきました。ありがとうございました。

＜講演内容＞

◆ソフトウェア構成管理の概要

・広義と狭義のソフトウェア構成管理

ソフトウェアの構成管理とは、ソフトウェア開発における成果物の変更をコントロールすることである。現在の開発において生じる複雑な多岐にわたる成果物は、人手でのコントロールには限界があり無駄を生じている。、このため、ツールなどで行うことで、人は「意図したとおりのソフトウェアを作る」という本来の責務に注力できるようになる。

ソフトウェアの開発の現場では「様々な要求への対応」「各種タスクの並行作業」「開発拠点の分散化」「確実で適切なリリース」といった非常に複雑な業務が要求されている。

こういった「複雑さの軽減」のために、変更依頼をトリアージ（優先順位の意志決定）し、優先順位に沿って要求の変更と機能追加／バグ改修を行い、リリースする、という変更管理のプロセスを適切に運用する必要がある。

狭義のソフトウェア構成管理は、このプロセスにおけるソースコードなどの成果物のバージョン管理を指すが、広義のソフトウェア構成管理は、変更管理の流れ全体を指し示す。

・ソフトウェア構成管理の構成要素

ソフトウェア構成管理の構成要素は①バージョン制御②ワークスペース制御③ビルド／リリース制御④プロセス制御、の４つである。

ソフトウェアのバージョンを管理では「自分が再現したい状況を正しく再現できること」が求められる。それ故、「①バージョン制御」で管理されるリポジトリだけでなく、状況が再現される「②ワークスペース制御」も重要である。

また近年のソフトウェア開発では、自動ビルドにより適切なリリースを目指すケースが増えているが、この場合も「③ビルド／リリース制御」で正しい情報が正しく取りだされた上で、ビルド／リリースが行われることが重要である。

また、多数のチームが同時並列的に開発を行うケースも増えているが、こういうケースでは、バージョン管理～自動ビルド～リリースまでのプロセスを適切に運用するための「④プロセス制御」が重要となっている。

・ソフトウェア構成管理と周辺構成

ソフトウェア開発のプロセスにおける基本要素は、①人／役割②動機／行動③成果物、の３つである。
この３要素は「プロセスの視点」でみると、仕様変更などの「動機／行動」によって「人／役割」がアサインされて作業されることで「成果物」が作成される、という流れである。

しかし、「成果物の視点」でみると、その成果物を作成した作業をしたのが「人／役割」であり、かつ、成果物が作成された理由が「動機／行動」である、ということで、プロセスの前後関係や動機が不確かになる、というケースが多い。プロセス改善をバージョン管理ツールの導入だけで実現しようとする場合がこのケースにあたり、なぜ機能追加が必要になったか、という本来の動機や変更の理由がわからなくない、という状況に陥りがちである。

この問題を解決するためにある「サポートシステム」として、「(変更の) 動機」、「(それを作業した) 人」までを管理する変更依頼管理のシステム (IssueTrackingSystem:ITS) があり、バージョン管理 (SCM)、リリース管理 (CI) と組み合わせて利用されている。これによって、複雑さが軽減され、いつでも構成を識別／公開することが可能になる。

◆ソフトウェア構成管理システムの仕組み

・ソフトウェア構成管理の特徴
　ソフトウェアのバージョン管理の中心は「リポジトリ」と「ワークスペース」である。
　「リポジトリ」で成果物の格納、成果物のバージョン化、成果物構成・関係のバージョン化、が行われる。かつての構成管理では、ファイル１つ１つの版管理であったが、近年の構成管理では複数ファイルの依存関係も１つの「変更セット」として管理され、構成管理では「変更セットのトランザクション」を管理するものになっている。
　「ワークスペース」では、安全な成果物の「識別」、「取得・更新」、「利用と引き渡し」が行われる。
　ただ、実際には、「上書き」や「ファイル名を変えてコミットした場合」などのケースでは、正しく成果物を再現するには、ある程度人手によるプロセスも不可欠である。
　また、より積極的に構成管理を活用するケースでは、ブランチを作成し、成果物をマージしてチェックアウト、チェックイン、という使い方になる。こういう場合は、チェックインポリシーを策定する必要がある。
　また、ソースコードの識別・再現だけでなく、ビルド／リリースの構成の識別も重要であり、これによって、バグの特定が容易になる。
　ソフトウェア構成管理のプロセス管理の適切な運用は、構成管理の属人性を排除し、ミスや手間の軽減につながる。その為の手段としては、構成管理の運用ポリシーを策定し、たとえば「バグは ID が与えられていないとコミットできない」というような運用ルールを定める必要がある。

・中央集中バージョン管理
　中央集中バージョン管理（CVCS）は、中央に単一のリポジトリを置くもので、管理・制御の集中化が容易となる。例としては、SVN、Clear Case、TFS などがある。
　正しいものが置いてある場所が、常に中央のリポジトリである、という点が明確なシンプルなシステムだが、そのため、常にリポジトリに接続する必要がある。

・分散バージョン管理
　分散バージョン管理（DVCS）は、ワークスペース毎に local のリポジトリを持ち、それらの分散した local リポジトリを中央の remote リポジトリが管理する形態であり、Git、Mercurial などがある。local リポジトリがある為、ワークスペース毎に独自にバージョン管理を行うことができるので、オフラインの作業環境でも対応できる、という利点がある反面、プロセスの制御が複雑で制御しにくい、という欠点もある。

・ソフトウェア構成管理システムの導入のポイント
　CVCS、DVCS のどちらの構成管理を導入すればよいか、という指針は、以下の通り。
　－「頻繁な変更」「地理的に分散」「複雑なブランチ戦略」「他システムとの連携が複雑」「ソーシャルコーディング」・・・これらのキーワードで一つでも当てはまるなら、DVCS が適している。
　－「変更は緩やか」「リポジトリ常時接続可」「堅実な構成管理」「定型的ブランチ戦略」「他システム連携は緩い」「メンバーが構成管理に慣れていない」・・・これらのキーワードが全て当てはまるなら、CVCS が適している。
　近年では、DVCS を採用するケースが増えているが、特に DVCS が適合するケースは、以下の通り。
　－「反復」「フィードバック」の多いアジャイルプラクティスを行う場合
　－それぞれの目的別に複数のブランチを作って管理するモダンブランチ戦略を採る場合
　　（Pull Request が可能な GithubFlow などがよく用いられている）
　－DVCS に対応した、ITS、CI システムを使った、“広義のソフトウェア構成管理”を行う場合

◆期待に応えるソフトウェアのためのソフトウェア構成管理
－ビジネスニーズとソフトウェアの品質の変化
　売上増などのビジネスニーズを満たすために IT を積極的に活用することを、近年では DevOps（ビジネス駆動ムーブメント）と称している。DevOps を実現するために重要なことは、ソフトウェア成果物の「共同所有」という概念である。特に近年では、インフラのコード化が進み運用側での成果物も増えているため、開発・運用双方の成果物を互いに共有する必要性が高まっている。故に、ソフトウェア構成管理は、開発・運用の協調のための情報インフラとしても重要になってきている。

ー積極的なソフトウェア構成管理

ソフトウェア構成管理はソフトウェア開発の複雑さを軽減させるが、それだけでなく、コミュニケーション、コラボレーションを高める役割も担っている。例えば、コード単位の情報は開発者には理解できても経営層には理解できないので、ソフトウェア構成管理で経営層に理解できる“粒度”まで調整する必要がある。

その他、アイデアがビジネスの価値になるまでの流れを重視する「Flow of Value」や、従来の構成管理より、各自の知見を集めた集合知を活用する「ソーシャルコーディング」といった開発手法を実践する上でも、ソフトウェア構成管理が活用されてきている。

◆質疑応答

Q:DevOps の話の中で、成果物の「開発と運用の共同所有」の重要性が指摘されていたが、「共同所有」することでの具体的なメリットや事例などを聞かせて欲しい。

A:近年は、クラウド環境を利用したサービスなどでは、開発と運用の壁がなくなる「ボーダーレス化」が進んでいる。ここでは、開発側はインフラの制約を考慮する必要があり、一方、運用側もアプリケーションの制約を考慮する必要がある。こういうケースにおいては双方の成果物の「共同所有」が非常に重要になっている。

また、オープンソースなどのソーシャルコーディングの現場では、成果物の共同所有が重要なのは当然だが、共同所有することで、今までに無かった新しい知恵を出し合っての改善が進められている。

以上