オープンソース Git を活用した FA 統合開発環境における複数人並行開発の実現

岩村　慎太郎

近年、市場における消費者ニーズの多様化に伴い、短いサイクルで新しい商品が次々と生み出されている。製造業各社は、新しい商品を製造するために、新たな生産設備の増設や既存設備の段取り替えパターンの追加などの対応を行っている。

こうした中、生産設備の制御システムの複雑化や、制御ソフトウェアの大規模化が進んでいる。また商品リリースの短いサイクルに対応するため、生産設備の開発期間の短縮も必要になり、制御システムの開発を複数の開発者にて分担するケースが増えてきている。このようなケースにおいて、制御ソフトウェアを1つにまとめ管理する必要があるが、重複した変更の整合や変更履歴の管理などの課題があった。

本稿では、このような課題解決に向け、オムロンのFA統合開発環境であるSysmacStudioとオープンソースのバージョン管理システムであるGitの連携技術を確立し、複数人による制御ソフトウェア開発の変更管理工数および変更作業のミスによる後戻り工数を削減し開発期間を短縮できることを確認した。また、完成した制御ソフトウェアを流用し、類似する生産設備の制御ソフトウェア開発（派生開発）についても、本技術を適用し開発期間を短縮できることも確認した。

Concurrent development by multiple developers in an integrated factory automation development environment using an open-source system “Git”

Shintaro Iwamura

In recent years, consumer needs in the market have diversified, and new products have been successively produced in a short cycle. Manufacturing companies have installed new felicities or added new production patterns to existing systems to manufacture new products.

The control system of production facility has become more complex and the scale of control software has increased. Also, in order to respond to a short cycle of product release, it is necessary to shorten the development period of the production facility, and the number of cases in which the development of the control system is shared by multiple developers is increasing. In such a case, it is necessary to collectively manage the settings and control program, but there have been problems such as matching of duplicate changes and management of change history.

This article describes the cooperating technology that we developed for SysmacStudio which is Omron's integrated factory automation development environment, and an open source version management system Git. We confirmed that it can shorten the development period by managing changes of software developed by multiple developers, and reducing the backward man-hour caused by mistakes. It was also confirmed that the development period could be shortened by using the completed control software and applying this technology to the control software development (derivative development) of similar production facilities.

1. まえがき

オムロンは、生産設備の制御に必要なセンサ（Input）、出力機器（Output）、コントローラ（Control）に加えて、ロボットやセーフティ機器を含めたFA（Factory Automation）システム全体を1つのアプリケーションソフトウェアで開発可能なFA統合開発環境SysmacStudioを提供しており、生産設備を制御する機器の設定および制御プログラムなど

連絡先：岩村　慎太郎　shintaro.iwamura@omron.com

の制御ソフトウェアを一元管理できるようにしている。SysmacStudioでは、この一元管理された制御ソフトウェアをプロジェクトデータと呼んでいる。

近年の生産設備の高度化・複雑化に伴い、それぞれの制御機器に精通したスペシャリストが集まり1つの生産設備を開発するケースや、生産設備の開発期間を短縮するため、大規模化するプログラマブルロジックコントローラ（PLC）の制御プログラムの開発を複数人で行うケースが増えてきた。複数人で1つの生産設備を並行して開発する場合、マスタとなる生産設備のプロジェクトデータのコピーを各開発者に配布し、都度、各開発者がSysmacStudioの編集機能を用い変更内容をマスタのプロジェクトデータに反映する作業が必要になる。

しかしながら、各開発者の変更内容をマスタに反映するには、変更箇所の特定を行い、他の開発者との変更箇所の重複の有無の確認、重複して変更された箇所の取捨選択、および変更の見直しを行う必要があり、この作業に多くの工数を要することが課題となっていた。加えて、この作業が手作業で行われるため、開発者の判断・操作ミスで制御ソフトウェアに不具合が混入し、開発の後戻りの原因にもなっていた。

一般的に、ソフトウェア開発を複数人で行う場合、開発するソフトウェアの変更管理を行うため、バージョン管理システムの導入が広く普及している。バージョン管理システムとは、開発するソフトウェアのソースコードファイルやドキュメントファイルの変更日時、変更者、変更内容など変更履歴を管理するためのシステムであり、ファイル単位で変更履歴を保存することができる仕組みである。また、複数人で1つのファイルを同時に編集しても、編集の重複がない場合は自動的に各編集内容を1つのファイルに反映できる自動マージ機能など、効率よくファイルの変更の集約、管理ができるようになっている。

そこで、このバージョン管理システムを活用し、複数人による生産設備の制御ソフトウェア開発における課題解決に取り組むこととした。SysmacStudioのプロジェクトデータをバージョン管理システムで効率良く、かつ正確に変更管理できるようにするため、プロジェクトデータの更新に関わる3つの技術課題を抽出し解決策を検討した。

本稿では、これらの技術課題に関する具体的な解決手段に加え、バージョン管理システムと連携するFA統合開発環境により、プロジェクトデータの変更管理工数を削減できた具体的事例についても説明する。なお本技術は弊社商品SysmacStudio Ver.1.20.0のチーム開発オプションの機能として2017年10月にリリースしている。

2. 課題

プロジェクトデータには、制御プログラムなどテキスト形式で保存されたデータの他、表示器（HMI）で使用するグラフィックデータなど、様々なデータ形式のファイルが含まれる。また、FAシステムの1つの制御機能がプロジェクトデータの複数のファイルに分割し保存されている場合や、プロジェクトデータ内で定義されるデータの冗長性を排除するため、ファイル間で相互にデータを参照する場合もある。

こうしたプロジェクトデータについて、SysmacStudioを使用せずにバージョン管理システムで自動マージ機能などによりファイル更新を行うと、データの参照関係が壊れてしまう可能性がある。また開発者がバージョン管理システムの機能を用いてファイルの差分を直接確認してもデータの内容が理解できないため正しくマージを行うことができず、ファイルを直接編集するとプロジェクトデータを壊してしまう可能性もある。

そこで、SysmacStudioとバージョン管理システムを連携させることで、上記のような問題の発生を未然に防ぐプロジェクトデータの変更管理の仕組みについて検討した。この仕組みの実現にあたり、次の3つの技術課題の解決に取り組んだ。

2.1　クライアントツール機能の実現　バージョン管理システムの操作には、一般的には専用のアプリケーションソフト（クライアントツール）を利用する。クライアントツールは、バージョン管理システムの運用に必要な機能を備えており、管理するソフトウェアについてファイル単位での編集や編集の履歴、その差分を表示することもできる。しかしプロジェクトデータには開発者が直接理解できないファイルも含まれるため、プロジェクトデータの生成元であるSysmacStudioで、ファイル毎の差分を意識することなくプロジェクトデータの変更箇所、変更履歴が確認できる必要がある。そこでバージョン管理システムのクライアントツール機能をSysmacStudioへ統合することが1つ目の技術課題である。

2.2　プロジェクトデータの比較・マージ機能の実現　開発者はクライアントツールを使用して、修正中のファイルとバージョン管理システムに登録されているマスタファイルと比較し、その差分情報を含む新たなマスタファイルを登録することができる（比較・マージ機能）。しかし、この機能が利用できるのはテキスト形式のファイルのみであるため、表示器のようにグラフィック形式など複数の形式のファイルを含むプロジェクトデータはバージョン管理システムに差分情報を登録することができない。そこで、バージョン管理システムと連携しSysmacStudioでプロジェクトデータの比較・マージ機能を実現することが、2つ目の技術課題である。

2.3　プロジェクトデータの整合性の保証　プロジェクトデータにおいて、ファイル間で相互にデータを参照し整合性の保証が必要なものについては、バージョン管理システムによる不正な変更を防止することや、不正なデータの変更を検知し、必要に応じてバージョン管理システムの過去の履歴から正常なデータを復元することが必要になる。このプロジェクトデータに含まれるデータの整合性に関わ

る異常の検知と、正常データへの復元が3つ目の技術課題である。

3. 技術内容

本稿ではソフトウェア開発の現場で広く普及しているバージョン管理システムの一つである Git[1] と SysmacStudio との連携技術について説明する。Git は分散型のバージョン管理システムであり、セキュリティの制約からネットワークが使用できない生産現場でも利用可能な特徴を持つものである。

3.1　ウインドウズシェル拡張機能を用いたアプリケーションソフトウェアの統合　SysmacStudio において、バージョン管理システムの操作機能を実現するため、Git のクライアントツール（TortoiseGit[2]）との連携技術を検討した。具体的には、SysmacStudio からウインドウズシェル拡張機能を通して TortoiseGit の機能を呼び出すことで、Git の機能を利用できるようにした（図1）。ウインドウズシェル拡張機能とは、Windows OS が提供するエクスプローラ上に存在するファイルやフォルダのコンテキストメニューなどを独自に変更し拡張する機能である。このウインドウズシェル拡張機能を利用することで、TortoiseGit のコンテキストメニューを SysmacStudio に統合した（図2）。これにより、SysmacStudio から TortoiseGit の機能を直接使うことができるようになった。

また、ウインドウズシェル拡張機能を使うことで、常に最新の TortoiseGit の機能を SysmacStudio から利用できるようになった。TortoiseGit と SysmacStudio をウインドウシェル拡張機能のインタフェースで結合することにより、それぞれのアプリケーションソフトウェアは相互に影響を与えることなく独立して機能強化をはかることができる。実際にオムロンがバージョン管理システムと連携した SysmacStudio を発売してから Git、TortoiseGit 共に何度も更新されており、リビジョングラフ機能などの TortoiseGit の新機能や Git 本体の機能の改善を SysmacStudio ですぐに活用できている。オムロンとしてはこの間 TortoiseGit や Git の進化に追従するための SysmacStudio の改造は不要であった。

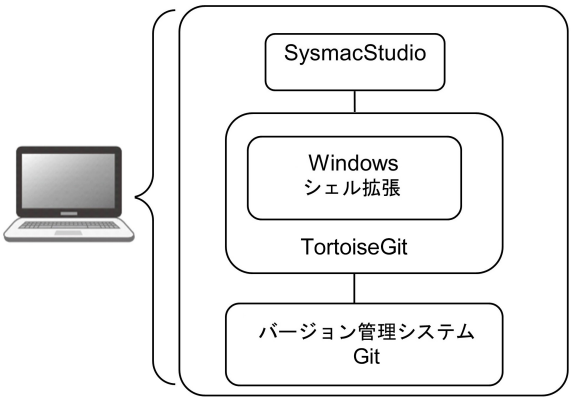

図1　バージョン管理システムとの連携構成

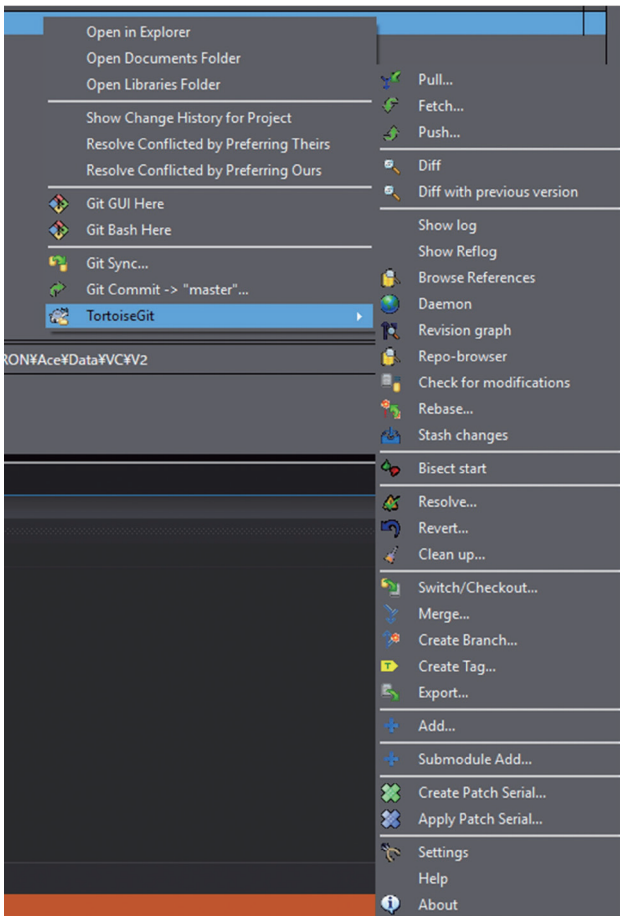

図2　SysmacStudio上のTortoiseGit コンテキストメニュー

3.2　開発者が直感的に操作できる差分の表示とマージ

バージョン管理システムから取得したプロジェクトデータの差分情報から装置を構成する制御機器の設定や制御プログラムの変更内容を確認し、マージできるようにするため、制御機器単位でビジュアルに表示比較できる機能を SysmacStudio で実現した。

本稿ではこれら比較機能の中で、ラダープログラムの比較、表示機の画面データの比較について説明する。

ラダープログラムは、PLC で一般的に使われているプログラム言語であり、リレーによる論理回路を記述するために考案されたものである。現在では複雑な演算などを行うための命令語なども使用できるようになっている。

従来、この論理回路を図示表記するラダープログラムは、変更の差異を見分けることが難しかったが、回路要素単位で差分を抽出することによって、回路要素単位で比較できるようにした（図3）。回路要素単位での比較を回路の接続順（パワーフロー）に沿って入力側と出力側の2方向から差異を検索し、その差分結果を比較することで実現した。これによって、現場で変更したラダープログラムの内容を詳細に比較できるようになった（図4）。

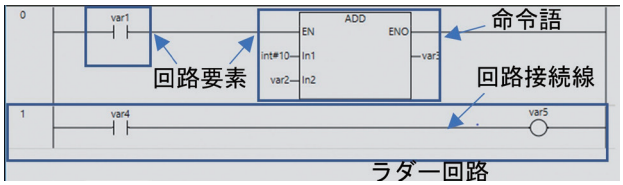

図3　ラダーエディタのデータ構成

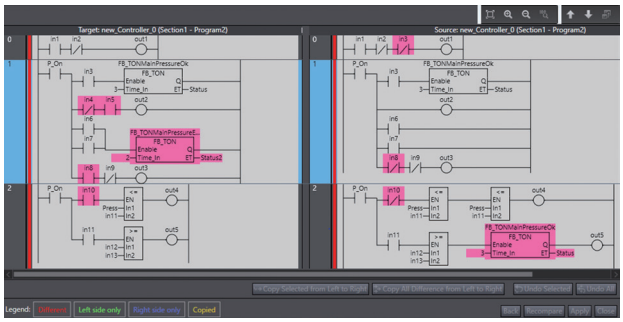

図4　ラダーエディタの比較画面

SysmacStudioは、表示器の画面を編集するPageEditorという機能を持っている。PageEditorで編集する画面データには、画面に配置するボタンなどの形状、色などのデータや、ボタンの状態に応じて切り替わる文字データなど、複数の形式のデータから構成される。SysmacStudioはこれらのデータおよびデータの関係を階層的に管理している。そこで、SysmacStudioはバージョン管理システムから取得したデータの差分情報を階層化したデータの並びで整理し、PageEditorの画面から全てのデータ差分詳細を簡単に把握できるしくみを実現した。例えば、画面上に配置されたボタンに表示する文字列が変更された場合、PageEditorの画面上でボタンに関連するデータに変更があることを表示し、その後、文字列の詳細を表示する画面に表示を切り替え、差異の詳細が把握できるようにした（図5）。

この、PageEditorの比較機能は現在（2018年11月現在）特許出願中である。

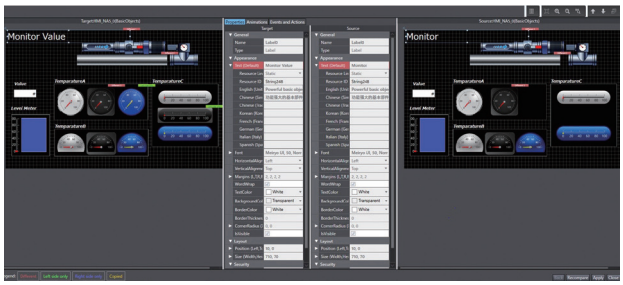

図5　PageEditorの比較画面

これらにより、バージョン管理システムと連携し、開発者がデータの差分を直感的に理解し正確にマージする機能をSysmacStudioで実現することができた。

3.3　MD5ハッシュを用いたデータ間の整合性保証

バージョン管理システムによるプロジェクトデータの整合性に関わる異常を検知するために、データ間の整合性を保証する方法を検討した。

相互に依存関係のあるデータの整合性を担保するために、依存関係のあるデータを1つのデータ群として扱う方法をMD5ハッシュを使って実現した。MD5ハッシュは任意の長さの元データを元に128ビットの値を生成するハッシュ関数であり、MD5ハッシュによって生成されたデータを見ることでデータが変更されたかが分かる仕組みである。ハッシュ値は広く一般にWebのデータのやり取りにも使われている技術であり、MD5、SHA-1、SHA-2といった種類があり、セキュリティや、ファイルの同一性の担保に用いられている。今回の用途はファイルの内容が完全一致しているかのチェックであるので、計算時間が短く、サイズの小さいMD5ハッシュを採用した。

依存関係があるデータは、複数のデータを1つのデータとしてまとめたMD5ハッシュ値を作り、データが更新される度にハッシュ値を更新するしくみとした（図6）。これをプロジェクトデータとして保存することよって依存関係のあるデータの一部分だけ変更した場合に、MD5ハッシュ値のデータの不整合を検出することで、プロジェクトデータの不整合を検出できるようになった。SysmacStudioでプロジェクトデータの読み出し時に不整合を検出した場合、検出した対象のデータを表示することで、ユーザはバージョン管理システムを使って変更履歴から正常データへの復元が可能になった（図7）。

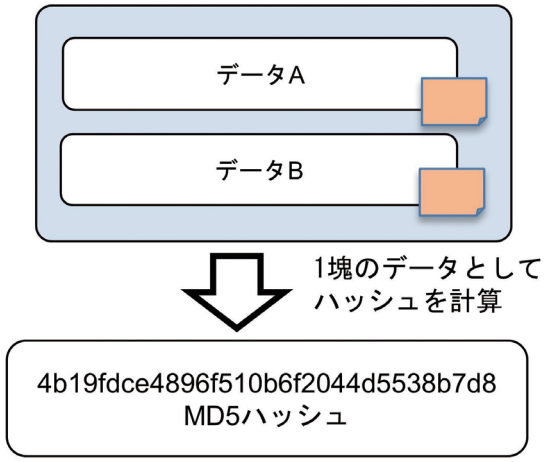

図6　MD5ハッシュの生成

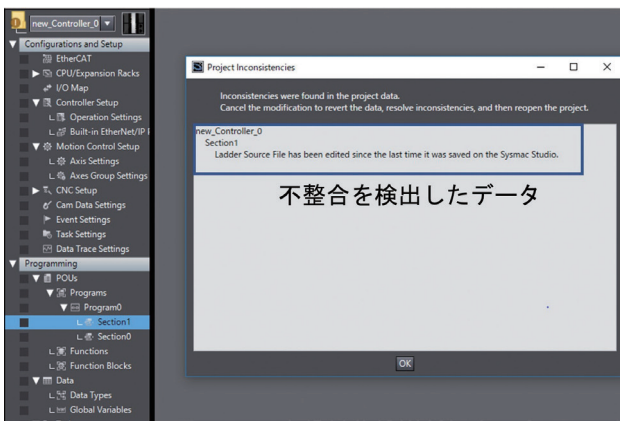

図7　データの不整合検知UI

4. 技術の検証

バージョン管理システムGitと連携したSysmacStudioを利用して、実際の複数人による生産設備の開発ができること、および変更管理にかかる工数が削減できていることを検証した。検証のためのユースケースは、複数人開発を実施している複数の顧客に対して、生産設備の開発の進め方についてヒアリングを実施し、その結果に基づき定義した。

検証内容について説明する前に、まず、分散型バージョン管理システムGitのシステム構成と運用について簡単に説明する（図8）。

分散型バージョン管理システムは各ユーザのパソコン内にローカルリポジトリと言われる保存領域と、ユーザ間でデータを共有するためのリモートリポジトリを持つ。リポジトリにはバージョン管理対象のプロジェクトデータとその変更履歴情報が格納される。各ユーザは任意の作業タイミングでローカルリポジトリとリモートリポジトリの同期をとることができる。ローカルリポジトリでの変更を他のユーザと共有するときはリモートリポジトリへの反映（プッシュ）する操作を、他のユーザの変更をローカルリポジトリに反映するときはリモートリポジトリからの取り込み（プル）する操作を行う。ユーザはローカルリポジトリを使って作業し、ある程度作業がまとまったところでリモートリポジトリに反映する使い方ができる。

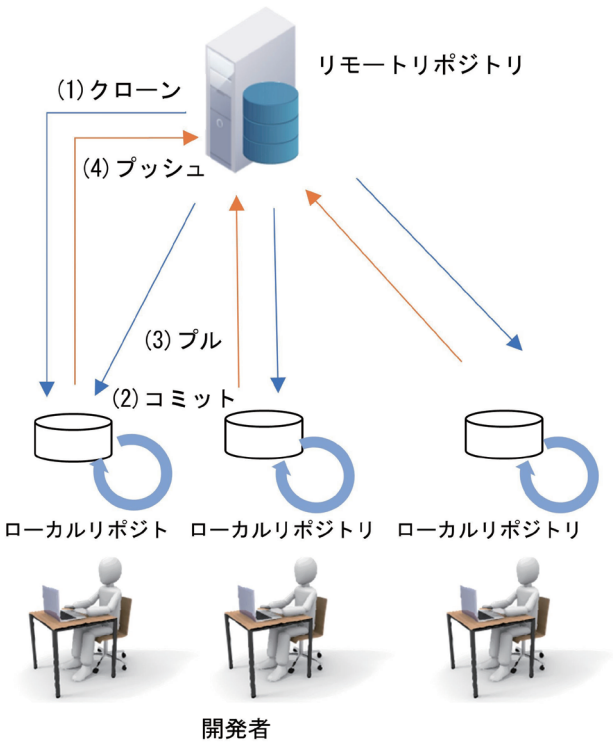

図8　分散型バージョン管理システム

4.1　バージョン管理システムの運用　実際の運用形態には、リモートリポジトリの共有方法の違いにより次の3つの形態があり、ユーザはいずれかの形態を選択する。

- リモートリポジトリをパソコンの共有フォルダで共有する
- Git専用のサーバを構築し運用する
- インターネット上のGitサーバサービスを活用して運用する

SysmacStudioでは、以上の3つの形態すべてをサポートできることを実際のサーバを運用して検証した。これによって、開発者は自社の生産設備の開発に適した方法を選択してバージョン管理システムを運用することができる。

4.2　複数人による生産設備の開発　バージョン管理システムを使用して2名の開発者（開発者XとY）が生産設備の開発を進めるフローを以下に示す。

はじめに、プロジェクトデータの編集の分担を決める。SysmcStudioのバージョン管理機能は、複数人で1つのプロジェクトを同時に編集しリモートリポジトリへプッシュする際に、Gitのオートマージ機能によって他の開発者の変更内容をマージする機能を持っている。他の開発者が同じデータを同時に編集していた場合、変更をマージしようとすると、変更内容に競合が発生しマージすることができなくなる。この競合の発生を防ぐために以下のように分担する。

開発者X：SysmacStudioプロジェクトの構成・設定全般と、データ型、グローバル変数、ラダープログラム（ProgramX）の編集を担当

開発者Y: ラダープログラム（ProgramY）の編集を担当

リポジトリとブランチの関係は図9の通りである。

各開発者は各開発者のパソコンのローカルリポジトリ上で作業用ブランチの作成し、プログラムの変更を行う。変更が終わった後に、ローカルリポジトリの変更をリモートリポジトリにプッシュして変更を反映する。リモートリポジトリを介して各開発者のパソコンのローカルリポジトリのデータをマージする。

詳細な操作方法はSysmacStudio プロジェクトバージョン管理機能 スタートアップガイド参照[3]。

社内で検証した結果、図9に示すフローを繰り返すことによって、複数人で並行して、装置を開発できることを確認した。また、表1のように従来のバージョン管理システムを利用しない生産設備の開発に比べてバージョン管理システムを利用した場合、繰り返し行う変更管理の作業を大幅に短縮できることを確認した。

表1　検証結果

ユースケース	バージョン管理システムなし	バージョン管理システムあり
対象のプロジェクトを見つけるまでの時間（変更箇所特定）	1分	10秒
データ比較までにかかる時間（変更箇所の重複の確認）	1分	5秒
プログラムデータのマージ時間（重複して変更された箇所の取捨選択）	2分	30秒

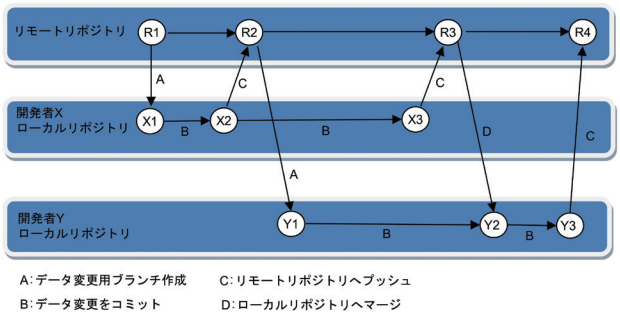

図9　複数人開発のフロー

4.3　複数人による派生生産設備の開発　次に、実際の顧客の装置で複数人開発を応用した派生開発についてバージョン管理システムとの連携の効果を検証した。派生開発とは、生産装置のバリエーションの増加に伴う、装置のハードウェア構成が異なる複数の派生装置のプログラムを共通化して開発する方法のことである。

検証内容は包装機のシステムのプロジェクトデータを複数の開発者で派生開発した場合の実施例である。

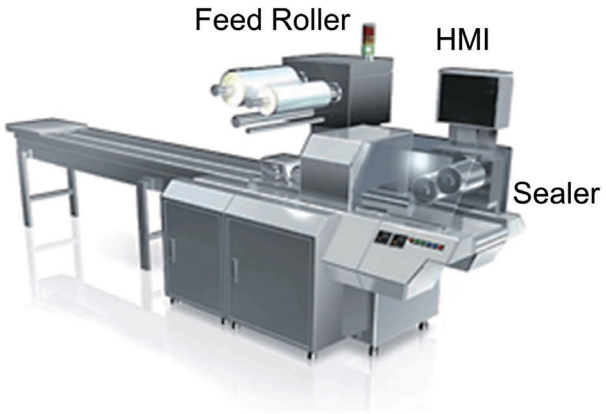

図10　包装機のシステム

この顧客は包装機のシステムをそれぞれの顧客の要望に合わせて一品一様の仕様で制作することを得意としている大手包装機メーカーである。すべての包装工程を含む基本構成から、顧客に不要な機能を省き、顧客が求める機能や仕様を組み合わせて一台の装置を開発している。一品一様ということもあり、ユーザの要求に合わせて装置を開発していくと、装置が複雑になり、複数人で開発すると誰がどこを変更したのか把握することやプロジェクトデータのマージなどの変更管理工数が大きくなっていた。複数人で開発しているが、生産性がなかなか上がらないことが課題であった。また、装置が大規模になるに従って不具合が発生した場合の原因特定に時間がかかるようになっていた。

そこで今回、バージョン管理システムとの連携機能を実装したSysmacStudioを採用し、実際の生産設備の開発で運用した。

上の図10のような包装システムを以下の表2条件で開発した。

表2　分担構成（開発者　5名）

分担構成	人数
SysmacStudioプロジェクトの構成・設定全般	1
制御プログラム	2
表示機画面	2

装置の制御システムには、オムロン製の以下の機器が採用された。

1. PLC NJシリーズ CPUユニット
2. NXシリーズ I/Oシステム
3. ACサーボシステム 1Sシリーズ
4. プログラマブルターミナル（表示機）NAシリーズ

この開発案件が完了した後、包装機メーカーの開発者より以下の評価・フィードバックがあった。

このバージョン管理システムを使って開発する前は、プロジェクトデータをSysmacStudioのエクスポート機能で1つのファイルにまとめ、ファイル名に日付をつけて保存、管理していた。トラブルの際などはプロジェクト間の照合機能を使って比較していた。この方法では、都度ファイルを開いて照合・比較するため、特定の修正箇所の履歴を時系列で確認する作業に時間がかかり、トラブルの原因箇所のプログラムを特定する作業に大変な時間と手間がかかっていた。

今回SysmacStudioのバージョン管理システムの導入によって過去の履歴から原因となる履歴を探し、瞬時に比較結果を表示することで原因箇所の特定時間が従来の1/4以下に短縮できた。その結果、開発工程におけるファイルのバージョン管理工数が半分以下になった。

5. むすび

SysmacStudioで複数人開発を実現するために、バージョン管理システムGitのクライアントツールTortoiseGitとウインドウズシェル拡張技術を利用して、GitとSysmacStudioの連携を実現した。

また、FA統合開発環境の様々なデータ形式に対して、ビジュアルな比較技術を実現し、ユーザが直感的にデータを比較・マージできるようになった。

さらに、MD5ハッシュを利用して、SysmacStudioのプロジェクトデータに含まれるデータの間にある整合性を保証する技術を確立した。

この確立した技術を複数人開発に適用することで技術の有効性を確認した。また、複数人開発だけではなく装置の派生開発についても今回確立した技術が有効であることを確認した。

実際に装置開発の現場においてプロジェクトデータの管理工数を半減することができ、本技術を実装したSysmacStudioの導入がグローバルで拡大している。

今後も、オープンソースの技術進化を活用して、お客様のものづくり革新に貢献していきたい。

参考文献

1）https://git-scm.com/（参照 2018-11-19）
2）https://tortoiseGit.org/（参照 2018-11-19）
3）オムロン. SysmacStudio プロジェクトバージョン管理機能スタートアップガイド, SBCA-112

執筆者紹介

岩村　慎太郎　Shintaro　Iwamura
インダストリアルオートメーションビジネスカンパニー　商品事業本部
コントローラ事業部　第2開発部
専門：ソフトウェア工学