一般論文

製造支援システム開発の効率化に向けた取り組み

神戸事業所　開発部　開発第3課
黒田　巧、橘　耕平

1. まえがき

製造支援システム（以下、本システム）とは、設計図書や部品表など三菱電機㈱電力システム製作所での電子基板設計に関連するデータを一元管理し、製品開発業務の生産性向上と品質確保を実現するものである。本システムは、運用開始から15年以上が経過した現在も機能拡充を重ねている。現行のシステムは、海外サードパーティー製のパッケージソフトウェアに独自機能を追加して構築しており、機能数は約250、プログラムステップ数は約1500kLに達している。当社は本システムの構築当初からソフトウェア開発を担当しており、機能拡充開発を効率化するために様々な改善を進めてきた。

また、現行パッケージソフトウェアの製品サポート終了に伴い、パッケージソフトウェアを更新する新システムへの移行開発が始まった。移行開発では、独自追加した機能を含めシステムの全機能を更新後のパッケージソフトウェアに対応させる必要があり、移行開発の効率化に取り組んでいる。更に、現行システムに対するユーザーニーズに応えるため、図1に示すように移行開発と同時に機能拡充開発も継続している。

本稿では、本システムの機能拡充開発と移行開発で実施した開発効率化の取り組みを紹介する。

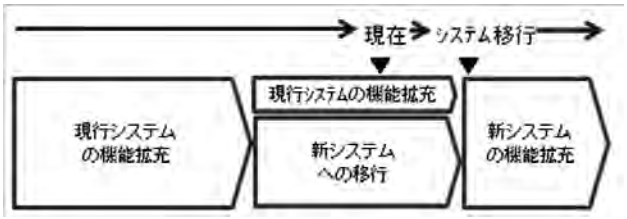

図1. 本システム開発の状況

2. 機能拡充開発の効率化

本章では、機能拡充開発の効率化に向けた課題と対策について述べる。

2.1　機能拡充開発の概要

機能拡充開発は、ユーザーニーズにもとづく新機能の追加と既存機能の改造を行い、図2に示すように、(1)パッケージソフトウェアのカスタマイズ開発と、(2)パッケージソフトウェアに格納したデータを活用するアプリケーション開発がある。

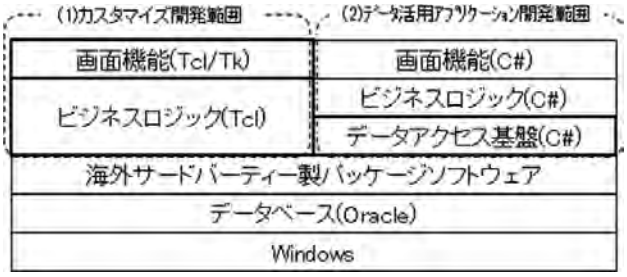

図2. ソフトウェア構成と機能拡充開発の対象

(1) カスタマイズ開発

パッケージソフトウェアに本システム独自の機能を追加する。パッケージソフトウェアの仕様により、プログラミング言語にはTcl/Tk(注1)を使用する。

(2) データ活用アプリケーション開発

パッケージソフトウェアに格納したデータを利用するGUIアプリケーションやバッチ処理アプリケーションを開発する。アプリケーション開発には、C#、VisualBasic、Javaなどのプログラミング言語を利用できるが、パッケージソフトウェア内のデータ操作には、ベンダー独自仕様のクエリー言語(注2)を使用する。

2.2　機能拡充開発の課題

パッケージソフトウェアに合わせた機能開発を行うため、以下の課題が存在した。

(1) Tcl言語でのカスタマイズ開発の効率化

(2) データ活用アプリケーション開発の効率化

(注１) スクリプト言語Tclと、GUIツールキットTk。
(注２) 問い合わせ言語(query language)。代表的なクエリー言語として、リレーショナルデータベースを操作するSQLがある。

(1) Tcl言語でのカスタマイズ開発の効率化

Tcl言語による開発は、パッケージソフトウェアベンダー提供の開発環境を使用する。しかし、この開発環境はテキスト編集以外の機能を持たないため、熟練者においてもコーディングミスによる手戻りが頻発した。また、開発したプログラムは、パッケージソフトウェア内のデータベースに格納されるため、構成管理ツールによる管理が困難であった。そのため、複数の開発者が同じプログラムを同時に編集した際、先に編集した内容が上書きされる問題が度々発生した。

(2) データ活用アプリケーション開発の効率化

パッケージソフトウェアのデータを操作するために、ベンダー独自のクエリー言語を使用する。しかし、当クエリー言語は複雑で、使用にはパッケージソフトウェアの仕様を理解する必要があり、開発者が限定されていた。

2.3 課題の対策

前述の各課題を解決するため、以下に示す対策を実施した。

(1) 統合開発環境の自社開発
(2) データアクセス基盤の構築

(1) 統合開発環境の自社開発

Tcl言語でのカスタマイズ開発の効率化対策として、市販ツールやフリーソフトウェアの導入を検討したが、我々のニーズには適さなかった。そこで、図3に示すように本システム開発専用の統合開発環境を開発した。

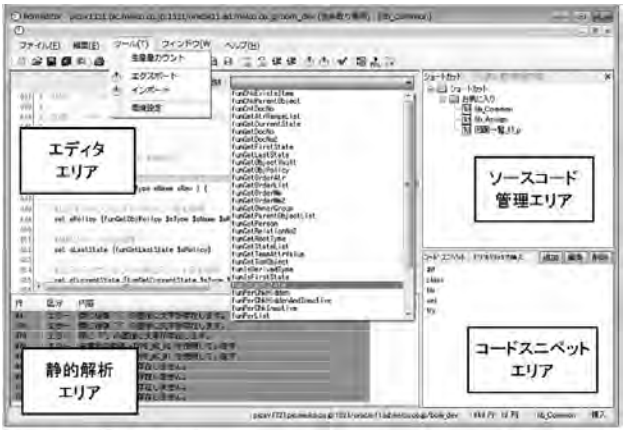

図3. 統合開発環境

統合開発環境には、表1に示す機能を実装することで、プログラミング作業の効率化を実現した。

表1. 統合開発環境の主な機能

機能名	説明
エディタ機能	キーワードをハイライト表示しプログラムの可読性を向上する。
入力補完機能	キーワードなどの単語入力を補完し、タイプミスの削減とキー入力を省力化する。
検索機能	システムの全プログラムを対象とした検索が可能で、影響先の抽出作業を省力化する。
静的解析機能	本システム開発で頻発していたコーディングミスの内容を整理し、コーディング時の事前検出を可能とする。おもなチェック項目を表2に示す。
構成管理機能	複数の開発者によるプログラムの同時編集で、編集内容が消失することを防止する。
コーディング支援(コードスニペット)機能	使用頻度の高いコードブロック(Tclの構文など)を事前登録しておき、コーディング時にコードブロックを呼び出して貼り付ける。その結果、タイプミスを削減し、プログラミング作業を省力化する。
お気に入り機能	作業中のプログラムをお気に入りに登録することで、次回作業開始の時間を削減する。

表2. 静的解析のチェック項目(一例)

チェック項目	説明
ブロック開始終了チェック	{ } ブロック、[ ] ブロックの開始と終了が対であるかチェックする。
“” チェック	“(ダブルクオート)の開始と終了が対であるかチェックする。
未定義変数使用チェック	未定義変数を使用していないかチェックする。

(2) データアクセス基盤の開発

データ活用アプリケーション開発の効率化対策として、図4に示すように開発者がベンダー独自仕様のクエリー言語を意識することなくプログラミングできるよう、.NET Framework(注3)のクラスライブラリとしてデータアクセス基盤を開発した。データアクセス基盤を利用することで、プログラミング作業の効率化を実現し、ソースコードの可読性を高めることができた。

(注3) Microsoft社が開発したWindowsアプリケーションの開発・実行環境。

図4. データ活用アプリケーション開発の変化

3. 移行開発の効率化

本章では、移行開発の効率化に向けた課題と対策について述べる。

3.1 移行開発の概要

移行開発は、更新後のパッケージソフトウェアに対応させるため、図5に示す機能を対象としたプログラム移植を実施する。一般的にパッケージソフトウェアの利用に際しては、パッケージソフトウェアの機能に業務を合わせることが多い。しかし、本システムの移行開発では現行システムとの互換性を優先させるため、すべての機能を新システムへ移植することとした。

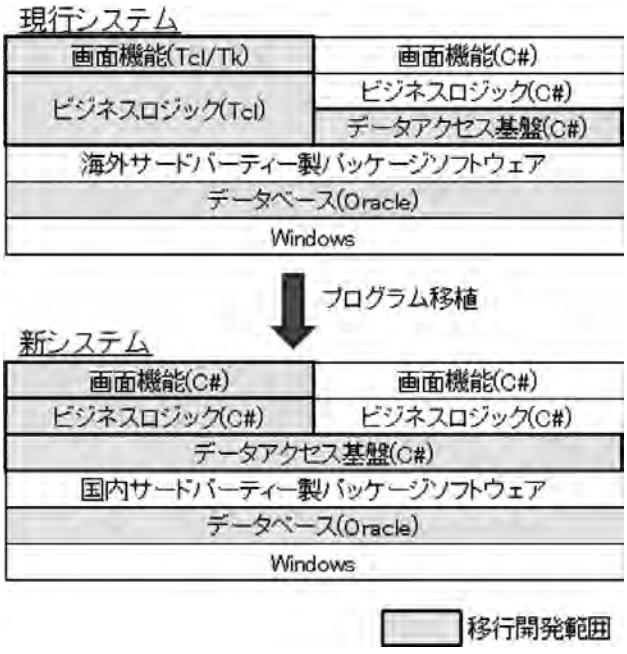

図5. 移行開発におけるプログラム移植対象

パッケージソフトウェア更新後の機能拡充開発では、C#言語を使用する。そのため、Tcl言語で開発した既設プログラムをC#言語で作り直す必要があった。また、ベンダー独自仕様のクエリー言語で実現したデータアクセス基盤も、更新後のパッケージソフトウェアが提供する言語で再構築する必要があった。

3.2 移行開発の課題

移行開発は規模が大きく、以下の作業効率化が課題であった。

(1) プログラム移植作業の効率化
(2) 回帰試験の効率化
(3) プログラム説明書作成の効率化

(1) プログラム移植作業の効率化

移行開発では、表3に示すようにTcl言語で開発した285kLステップのプログラムをC#言語で書き換える必要があり、移植作業に時間を要する。

表3. 移行開発の対象

現行システムでの開発言語	移行対象ステップ数	開発対象
VB、C#	686kL	パッケージソフトウェアに依存する約10%のプログラムを修正。
Tcl	285kL	すべてをC#に変換。

(2) 回帰試験の効率化

移行開発では、複数の機能を同時に移植する。プログラム移植の過程で機能間の影響評価が欠かせず、回帰試験に時間を要する。

(3) プログラム説明書作成の効率化

移行開発では、機能の更なる共通化を進めている。開発者は、設計書に記載された共通関数の仕様を参照し、プログラムを移植する。そこで、プログラミング作業の効率化に向けて、プログラム説明書(関数リファレンス)を用意することとした。しかし、プログラム説明書の作成には多大な時間を要する。

3.3 課題の対策

前述の課題を解決するため、以下に示す対策を実施した。

(1) プログラム移植作業の自動化

(2) 回帰試験の自動化
(3) プログラム説明書作成の自動化

(1) プログラム移植作業の自動化

プログラム移植作業の効率化対策として、Tcl言語からC#言語へのプログラム移植を自動化するツール(以下、プログラム移植ツール)を開発した。

プログラム移植ツールを利用することで、移植対象コードの約68%を自動変換した。

表4. プログラム移植の自動化(17年度実績)

項目	値
移植後コード量(C#)	139.4kL
自動変換量(C#)	95.2kL
自動変換比率	68.3%

以下に、プログラム移植ツールの機能を説明する。

① Tclプログラム変換機能

TclプログラムをC#プログラムに変換する機能である。表5に示すTclコマンドとC#メソッドの対応表、及び構文解析によってプログラムを変換する。

表5. TclコマンドとC#メソッドの対応(一例)

Tclコマンド	C#メソッド
^concat$ {0} {1}	{0}.AddRange({1})
linsert {0} {1} {2}	{0}.InsertRange({1}, {2})
join {0} {1}	string.Join({1}, {0})

プログラム変換においては、Tcl言語とC#言語でコマンドの種類が異なるため、変換不可能なパターンも存在し、手作業での修正が必要となる。そのため、変換できない項目については、ToDoコメントの挿入と意図的に当該コードでコンパイルエラーが発生する状態とした。これにより、手作業での修正漏れを防止する。

② GUI変換機能

Tcl/Tk画面をC#画面に変換する機能である。図6に示すように、Tcl/Tkのエクスポートファイル(注4)から部品の種類、サイズ、座標、文字サイズ、文字色などの情報を抽出し、C#画面(Windowsフォーム)のデザイン部分を定義するファイルを生成する。

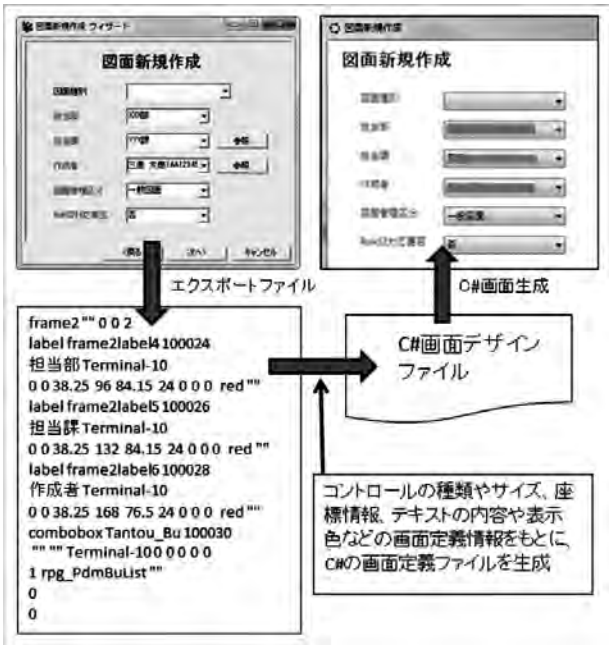

図6. GUI変換例

③ プラグイン機能

Tclプログラムは、任意のタイミングで外部プログラムファイルをロードし、外部プログラムに記述した関数を参照し、実行することができる。本機能は図7に示すように、C#上で外部プログラムを呼び出す機能を提供する。

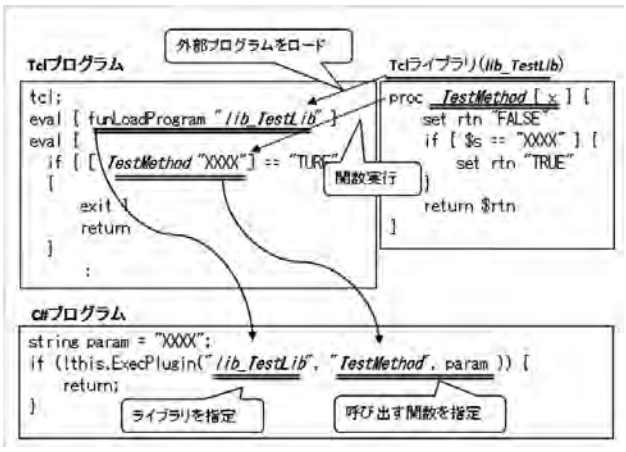

図7. プラグイン処理実装例

本機能によって、C#でもアプリケーションで使用する外部プログラムを事前に参照定義する必要を無くし、外部プログラムの呼び出しを可能とした。その結果、プログラム移植時に関数呼び出しインターフェースの記述変更が不要となり、移植作業量を軽減した。

(注4) あるアプリケーションのデータを、他のアプリケーションが解釈できるデータ形式として出力したファイル。

(2) 回帰試験の自動化

回帰試験の効率化対策として、図8に示すようにテストコードとVisualStudio(注5)の単体テスト機能、及びJenkins(注6)を活用して試験を自動化する仕組みを構築した。

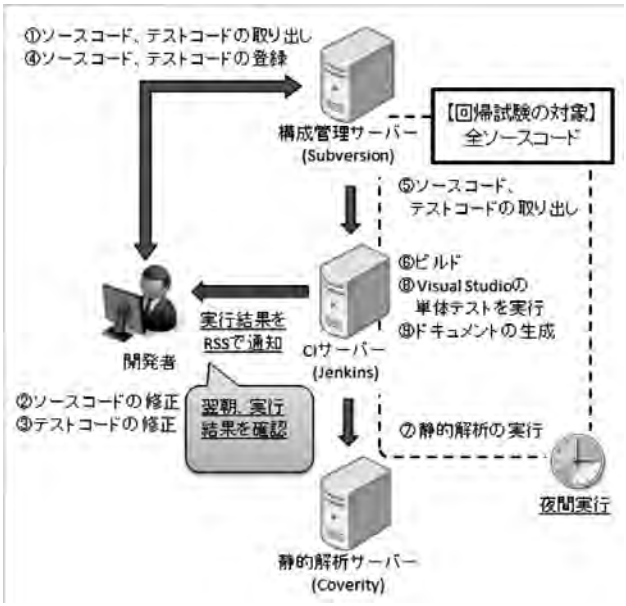

図8. 回帰試験の自動化

本環境では、夜間に回帰試験を自動実行し、試験結果は、RSS(注7)を用いて開発者にメールで通知される。開発者は翌朝、回帰試験の実行結果を確認し、回帰試験で発生したエラーを是正する。また、Coverityによる静的解析も自動化の対象とし、回帰試験作業を効率化した。

(3) プログラム説明書作成の自動化

プログラム説明書作成の効率化対策として、図9に示すようにプログラム説明書の作成にSandcastle(注8)、ソースコード修正に対応したプログラム説明書の反映にJenkinsを活用した。これにより、プログラム説明書作成を自動化した。

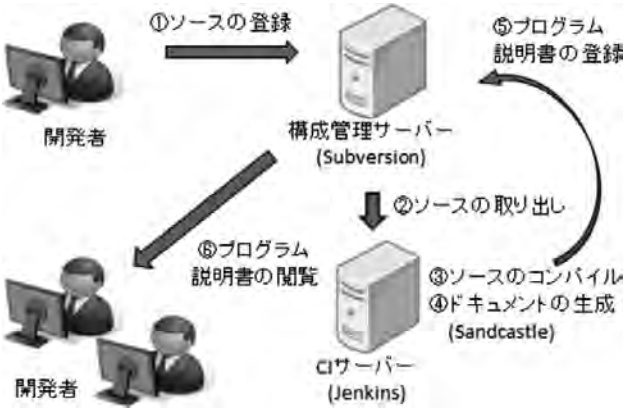

図9. プログラム説明書作成の自動化

4. むすび

本稿で紹介した取り組みにより、電子基板の製造支援システムの機能拡充開発と移行開発を効率化した。また、当社ではパッケージソフトウェアの利用事例が少ないため、機能拡充開発と移行開発で得たノウハウを「ソフトウェアパッケージ利用時チェックシート」として整理し、社内で情報共有した。

働き方改革が、ますます進む中、作業の機械化／効率化は必達である。今後も作業のツール化を進め、作業効率化に努める所存である。

(注５) Microsoft社が提供するWindows環境における統合開発環境。

(注６) 継続的インテグレーションツール。開発プロセスの自動化等に利用される。

(注７) Rich Site Summaryの略称。ウェブサイトの更新情報を配信するための文書フォーマット。

(注８) ドキュメントコンパイラ。ソースコードに記述したコメントから、関数リファレンスを作成する。

執筆者紹介

黒田　巧　　クロダ　タクミ

1998年入社。主に三菱電機神戸地区向け社内システムのソフトウェア開発に従事。現在、神戸事業所開発部開発第3課。

橘　耕平　　タチバナ　コウヘイ

1993年入社。主に三菱電機神戸地区向け社内システムのソフトウェア開発に従事。現在、神戸事業所開発部開発第3課。