ソフトウェア工学 122−12
（1999. 3. 19）

ソフトウェア再利用環境 EvoMan における構成管理機能

佐々木 幹郎　田村 直樹　柳生 理子
三菱電機株式会社 情報技術総合研究所

ソフトウェア構造の安定化、ソフトウェア開発プロセスの効率化、開発者間の情報共有を実現するためのソフトウェア開発支援環境 EvoMan の開発を行っている。EvoMan は、要求の変化が断続的に発生するような開発に対して有効な「発展型開発」モデルに基づいた支援ツールである。「発展型開発」は、系列を持った製品開発の際に発生するような、少しずつ機能が異なるものを継続的に開発する場合に顕著に見られるソフトウェア開発の形態である。EvoMan では、システムに対する変更の管理、開発プロセスの自動化、Web ベースの情報蓄積・共有機能を備える。本稿では特に EvoMan のベースとなる構成管理機能について述べる。

Configuration Management in Software Reuse Environment: EvoMan

Mikio Sasaki　Naoki Tamura　Riko Yagiu
Information Technology R&D Center, Mitsubishi Electric Corp.

We are developing a software development environment "EvoMan" that aims to accomplish following items: 1) making structure of software system stable, 2) making software development processes more efficiently, 3) sharing knowledge and information about software system among developers. EvoMan is a support environment for "Evolutionary Development" life cycle model. it is sufficient for software development with occurring intermittent changes of requirement, and useful for growing functions of a software system gradually, such like developing a family of products. EvoMan provides facilities to manage changes in software, to control change processes, and to share information through WWW. This paper describes about EvoMan's software configuration management function.

1　はじめに

顧客要求の高度化・複雑化が進む中、ソフトウェア生産性および品質の向上が求められている。これらはソフトウェア開発における長年の課題であり、開発の現場や研究機関などでさまざまな取り組みが行われている。ソフトウェアの再利用技術は、上記の課題に対して有効な回答として期待されており [1]、様々なドメインで事例の蓄積が成されている過程である。

「発展型開発モデル」はソフトウェア開発の一形態として提案されている。発展型開発とは、一度開発されリリースされた製品を様々な顧客の要求に応じて拡張することにより、ソフトウェアの工期の短縮や品質向上を目指したソフトウェア開発モデルである。本稿ではこの発展型開発についてソフトウェア再利用の観点から議論し、本開発を支援するためのツールである EvoMan について紹介する。

以降の章ではまず発展型ソフトウェア開発とその課題について述べ、その後 EvoMan の狙い、具体的な機能について述べる。

2　ソフトウェア再利用と発展型開発

2.1 ソフトウェア再利用と問題点

これまで、ソフトウェアの品質向上や開発期間短縮などソフトウェア開発に対する要求に応えるため、ソフトウェアを再利用する技術が提案され、さまざまな取り組みが行われてきた [2]。しかし、一部の汎用的なソフトウェア部品を除き、効果的な再利用が行われていないという指摘もある [3]。

ソフトウェア開発では、システムに対する顧客要求の変化や基盤環境の変化により、ソフトウェアシステムに対する変更が頻繁に発生する。このような要求の変化や、出荷後の保守活動や修正によるシステムの変更は避けられないものである [4]。特に、開発期間が長期にわたるような大規模なシ

ステム開発では、運用・保守期間のみならず、開発の間にも顧客の要求が変更され、それに伴って生じる機能の変更や拡張を開発システムに行わなければならない場合も多い。

このような開発において、あらかじめ要求定義に従って再利用可能なソフトウェア部品を用意しておくことは困難である。

2.2 発展型ソフトウェア開発

顧客要求の揺れや変化に対応するための開発モデルとして「発展型開発(Evolutionary Development)」のアプローチが提案されている[5]。

発展型ソフトウェア開発では、図 1に示すようにソフトウェアのライフサイクルを、最初のシステム（初期システム）の開発から、その標準システムを変化させながら繰り返し開発や保守を行い、最終的にその役割を終えるまでと捉えている。

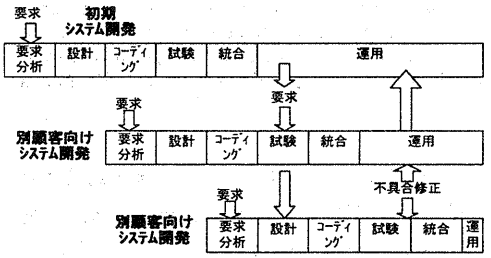

図 1 発展型開発モデル

発展型開発の例として、シリーズ製品における開発のような、最初に標準となるシステムを開発しておいて、それを元に異なる複数の顧客向けに必要機能の追加などのカスタマイズを行い提供するという形態がある。

発展型開発の利点としては、次の事項がある。

- 基盤となるシステムを顧客に提示する事で顧客の要求を効果的に収集でき、要求の揺れを少なくする事ができ修正点が明確になる
- 基盤となるシステムをそのまま部品として利用するため、比較的高品質なシステムが期待できる
- 複数の改良システムの並行開発が容易

その一方で、このアプローチに従ったソフトウェア開発を行う上で、以下の技術的課題がある。

- 並行開発や、下流からの要求変更のためにソフトウェア部品やシステムに対して変更が発生するため、版管理作業が複雑化する
- 類似する部品およびシステムが多くなり、どの部品を利用すればよいのかがわかりにくい
- 変更の発生が、他の製品（リリースされたシステム）や、他の作業者の開発作業に影響を及ぼす可能性が高い
- 蓄積されるソフトウェア部品はアプリケーションに依存したものが多い
- 部品の利用にドメイン知識が要求される。また、使い方を習得するのに時間がかかる

3 EvoMan の狙い

EvoMan は発展型のソフトウェア開発に着目し、2.2で述べた課題を解消し、効率の良い開発を行えるよう開発作業を支援することを目的としている。

EvoMan では、次の３つのアプローチをとる。

- ソフトウェア構造の安定化
- 開発プロセスの定義・安定化
- 情報・知識共有

以下、順に説明する。

3.1 ソフトウェア構造の安定化

発展型ソフトウェア開発では、標準システムやそこから派生したシステムを部品として再利用する。このような開発を進める上で必要な事は、変更が行われてもシステムに対する影響を低減し、できるだけ部品として再利用できる部分が多くな

るよう、システムが構造的に安定している事が重要である。それには、要求が変化した際に追随して変更する個所の特定が容易で、変更量が少ないものである必要がある。これを実現するための技術としてフレームワーク化[6]、デザインパターンの利用[7]などが提案されている。

また、構造を安定化させるためには、開発のすべての過程で、要求をトレースし、プログラムにどのような影響が及ぼされ、どのような変更が行われたのかを計測する事が重要となる。

3.2 ソフトウェア開発プロセスの安定化

システムや部品に対する変更が無秩序に行われると、変更に関する情報が失われたり、同じものを複数作成していたりする場合がある。これらの問題を防ぐためには、変更のプロセスを含め、効率の良いソフトウェア開発プロセスを定義し、安定して運用できるようにすることが重要である。

また、ソフトウェア開発では、作業見積りの誤りや、元になるソフトウェアに対する障害の発見等様々な要因により開発プロジェクトが破綻する場合がある。これらの問題を防ぐためにも開発プロセスの定義は必要である。

開発プロセスを安定化させるために、次の項目がある。

- 開発プロセスの自動化による効率化
- プロジェクト状況の的確な把握

3.3 知識共有

アプリケーションドメインにおいて、それぞれの開発者が持つ知識やノウハウはドキュメント化されにくく、これらの知識を伝達・共有する事は難しい。ソフトウェア開発者間の知識共有を行うためには、これまでドキュメント化が困難であった情報を自動的に抽出したり、開発者に簡易的に入力させたりする仕組みの提供や、再利用に向けたドキュメントの整備を行う必要がある。

さらに、得られた情報を蓄積し参照可能にしたり、他の開発者に通知するなどの工夫が必要である。

4 必要とされる機能

4.1 開発基盤としての構成管理機能

「発展型開発」に見られるような複数製品の並行した開発や、度重なる製品リリースが発生する場合、どの製品がどのバージョンのプログラムにより構成されていたのかといった情報を管理する事が人手では難しくなる。

これらの作業は、構成管理活動として定義されている[8]。ソフトウェア構成管理活動は、ソフトウェアの開発プロセス全体に渡る活動であり、そして開発の過程で生成されるあらゆる成果物とその変更に関わる活動である。構成管理活動の項目を表1に示す。

表1 構成管理活動の項目

活動項目	内容
識別	構成管理の対象要素を定義する作業
変更管理	構成管理要素に対する変更を管理する作業。変更内容についての情報管理や、必要な場所へのリリース、通知といった情報伝達が含まれる
審査	変更が正しいものかどうかを審査する作業
状況報告	変更情報の状況を報告する作業

EvoManにおいても構成管理は、ソフトウェア開発の基盤重要な支援対象となっている。EvoManでは、図 2に示すように、ソフトウェア開発全体に関わる成果物とそれに対する変更を管理する構成管理機能を基本機能として、その上に再利用のための検索やプロセス自動化、情報共有といった機能を追加していくようなシステム構成となっている。

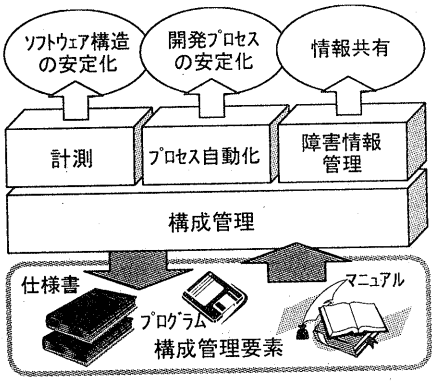

図 2 基盤としての構成管理機能

4.2 ソフトウェア構造安定化と計測

「発展型開発」において、最初に開発される標準システムが、ある程度部品化再利用を想定して作成されたものであっても、2.2で述べたように不具合の改修も含めて変更が余儀なくされる場合が多い。

最初のシステムがそのまま再利用される場合、システムの多くの部分で修正が行われるが、適切な分析が成されればその修正の割合は、システムの開発が進むにつれ少なくなっていく事が期待できる。このように、ソフトウェア構造の安定化を考えた場合、White-box 型の再利用部品をできるだけ Black-box 型の再利用へと移行する事が重要となる。また、コスト効率の面でも、ソフトウェア部品をそのまま利用する Black-box 型の再利用が有利である。

システム開発において、再利用が行われているかどうか評価するためには、システムにおけるソフトウェア部品の再利用状況の割合を計測しておく必要がある。利用されているソフトウェア部品に対して、修正を伴ったものなのか、修正を伴わないものなのかを計測する。また、システム内のどのプログラムが頻繁に修正されているのかを定量的に測定し、構造として問題がないかどうかを評価するための指標とする。

4.3 不具合情報の管理と情報共有

発展型開発では、最初に開発されたシステムを次の開発でも利用する。最初のシステムに対して行われた不具合に対する修正は、次のシステムが既にリリースされいている場合はそのシステムに対しても反映される必要がある。また、複数の製品開発が並行して行われている場合は、他の開発者にその情報を提供する必要がある。

このためには、先に述べた構成管理と、開発者間の情報蓄積・共有の機能が重要になる。

5 実装と課題

これまでに、障害情報共有化のための試作開発などを行ってきた [9]。現在は、構成管理機能と計測の一部分を実装し、運用を行いながら他の機能の開発を進めている。

5.1 構成管理機能

構成管理にあたっては、プロジェクトの構成管理計画を作成し、表 1の各項目を定めた。開発プロセスの同定と各プロセスで出力される成果物を定義した。構成管理要素間の依存関係を定義し、単純な依存関係に基づく影響分析機能を追加した。

構成管理機能の画面例を図 3に示す。

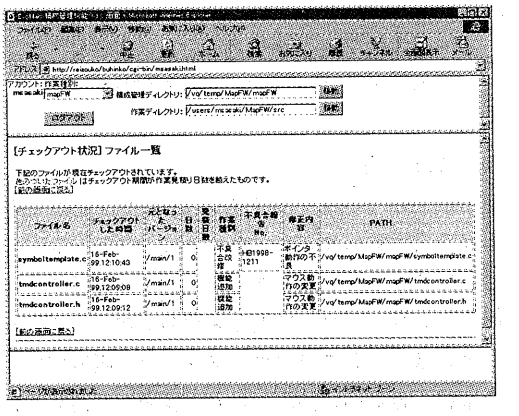

図 3 構成管理情報画面

5.2 計測

計測機能は構成管理機能との組み合わせによって実現される。

4.2で述べたように、システムを構成する各プログラムが修正され、利用されているかどうかを計測する。そして各プログラムの修正頻度に関する情報を収集する。

また、プログラムの修正には、機能を追加した事による修正や障害解決のための修正などその修正の理由により種類が分かれる。再利用性を計測したい場合には部品の機能的な変更を伴わない不具合修正は取り除く必要がある。これを切り分けて計測する必要があるため、プログラムのチェックインの際に、修正理由を選択させることで、この修正の発生理由が機能追加によるものなのか、障害の修正によるものなのかも合わせて記録するようにしている。

5.3 情報共有

必要なソフトウェア部品の検索や作業状況の参照など、開発者間で共有されるべき情報は多い。EvoMan では、電子メールによる通知や Web サービスにより提供することで、情報の共有を行えるようにしている。構成管理機能に関して参照可能な情報として表 2のようなものがある。

表2 情報共有

内容	ｻｰﾋﾞｽ	内容
作業状況	WWW	構成管理要素のチェックアウト状況を Web サービスで参照可能
障害情報	WWW	障害情報の発生と対処状況を参照可能
変更情報	Eﾒｰﾙ	チェックインされたプログラムに関する情報を関係者に通知

5.4 プロセス自動化

構成管理のための変更手続きをはじめとして、開発組織にはさまざまな開発のための運用ルールやノウハウが存在する。EvoMan は、図 4に示すように、従来開発者が持っていたこれらの開発ノウハウを組み込み、作業効率化を促進する事を目標としている。

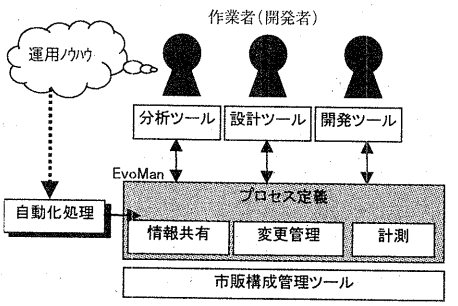

図4 運用ノウハウの組み込み

例えば、構成管理活動の一つに「審査」という過程があるが、開発組織によっては作業効率化のために、プログラムを変更した際にその変更が適切なものかどうかの判定を、プログラムのチェックインの際に試験プログラムを動かして通過すれば審査の過程を通過する事にするといった機能がプロセスの自動化の機能である。

5.5 システム構成

EvoMan を用いたソフトウェア開発のためのシステム構成を図 5に示す。

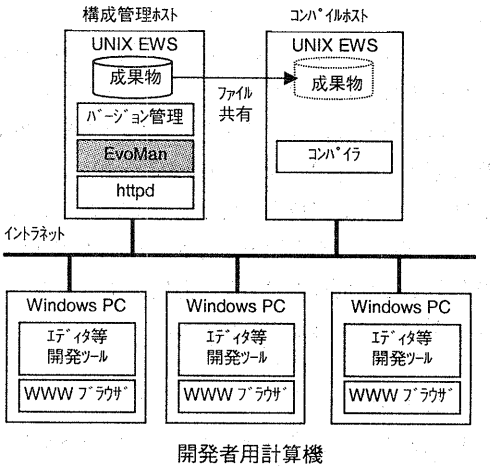

図 5 開発システム構成

EvoMan システム自体は Web アプリケーションとして構築されている。構成管理機能自体も Web インタフェースを通して提供される。

6 関連技術

ソフトウェア再利用技術についての取り組みはこれまでに様々な形で行われてきている。

再利用部品のアプリケーションドメインへの依存性については、ドメイン分析を行いソフトウェアの構造を見直し、またその設計情報を再利用する方法などが提案されている [10]。EvoMan が支援する発展型開発では、標準システムが存在し、開発者自体がある程度ドメイン知識を持つことを前提にしている。また、標準システムの作成にあたってドメイン分析を行うとより効果的であると考えている。

ソフトウェア再利用では、その効果を享受するにはソフトウェア部品をそのまま用いるような Black-box 型の再利用が望ましいとされている [7]。しかし、実際には汎用的なソフトウェア部品以外の整備は困難である場合が多く、部品に対して変更が発生する。我々は EvoMan をこれまでソフトウェア開発の現場で無意識のうちに行われてきた再利用のプロセスを定義し、Black-box 型再利用の割合を増加させるための支援ツールとして捉えている。

また、ソフトウェアのライフサイクルプロセスについては、従来のウォーターフォールモデルに対する反省から、スパイラル型 [12] の開発モデルや RAD(Rapid Application Development)、プロトタイピングなどの技術が提案されている。これらは主にプロジェクト管理を行う上でリスクの低減を目的とされたものである。発展型開発はスパイラル型の一つであるとも言える。

7 まとめ

以上のように、発展型ソフトウェア開発を効率良く行うために、構成管理機能をベースとした開発支援環境である EvoMan を提案した。

EvoMan は、繰り返し行われるソフトウェアシステムに対する変更を記録・管理し、その変更から再利用の状況を計測するための情報を蓄積したり次の開発のために情報を利用する事を可能にする。また、複数人での開発作業を円滑に行うための情報共有機能、開発プロセスの自動化機能などを備える。

現在、EvoMan は基本部となる構成管理機能を実装し、組み込み機器向け制御ソフトウェアの開発に適用中である。今後は、実システムへの適用・評価を行いながら、部品検索や部品理解のためのドキュメント作成支援など、ソフトウェア部品を効果的に運用するための技術や、より高度な分析・情報共有機能の開発を行う予定である。

1) Griss, M. L., and Wosser, M.: "Making Reuse at Hewlett-Packerd," IEEE Software, Vol.12, No.1, pp.105-107 (Jan. 1995)
2) Prieto-Diaz, R.: " Status Report: Software Reusability," IEEE Software (May 1993)
3) Biggerstaff, T., and Ritcher, C., "Reusability Framework, Assessment, and Directions," IEEE Software (Mar. 1987)
4) Boehm, B. W.: " Software Engineering Economics," Prentice-Hall (1991)
5) Dorfman, M.: "Requirement Engineering," *Software Requirement Engineering Second Edition*, IEEE Computer Society Press (1997)
6) Calder, P., and Linton, M.: " The Object-Oriented Implementation of a Document Editor," OOPSLA 92 Conference Proceedings, pp.154-165 (Oct. 1992)
7) Gamma, E., et.al.: "Design Patterns: Elements of Reusable Object-Oriented Software," Addison Wesley (1994), 本位田 真一他監訳, "オブジェクト指向における再利用のためのデザインパターン," ソフトバンク (1995).
8) IEEE Std828-1990: "Software Configuration Management Plans"
9) 柳生、田村、佐々木:"S/W 部品庫 EvoMan における品質管理機能," 情報処理学会研究会報告(98-SE-120-19), pp.133-140 (1998)
10) Prieto-Diaz, R., and Arango, G.: " Domain analysis and software systems modeling," IEEE Conputer Society Press (1991)
11) Poulin, J. F.: "Measuring Software Reuse," Addison Wesley (1997)
12) Boehm, B. W.: "A Spiral Model of Software Development and Enhancement," in ACM SIGSOFT Software Engineering Notes, pp.14～24 (1986)