ソフトウェア工学 86－1
（１９９２．７．２０）

プロセス指向による交換ソフトウェア自動試験環境

岩本 康　立元 慎也　白石 智
NTT交換システム研究所

本稿では、まず、交換ソフトウェア開発の現状とその効率化へ向けての課題を整理し、交換ソフトウェアプロセスの規定と、それを支援するプロセス指向SDEの必要性について述べている。ついで、プロセス構造化、対象ドメイン指向などの、プロセス構築のための方法論に基づいたプロセス指向SDEの実現法として、Process Integrationを提案し、プロセス自動化のための要求条件の整理と、それを実現するSDEの構成について述べている。最後にその適用例として、試験プロセスの自動化を支援する個々のツールを紹介し、それらを統合するプロセス指向SDEの具体的構築法を示している。

Process Oriented Automatic Testing Environment for Switching System

Yasushi IWAMOTO　Shinya TACHIMOTO　Satoshi SHIRAISHI
NTT Communication Switching Laboratories

This paper proposes methodologies for creating process oriented SDE. First, problems of a switching system software development are clarified. And needs of process oriented SDE are expressed. Then, the Process Integration is shown as an implement for process oriented SDE based on process structuring and domain oriented concept proposed as methods for creating a switching system·software process. And requirements for automatic testing are given and the SDE structure which meets these requirements is shown. Finally, as an example of SDE , the tools which support automatic testing and process oriented SDE which integrate these tools are shown.

1．はじめに

ここ十数年、ソフトウェア工学の面からソフトウェア開発に関する研究が行われ、数多くの開発作法や方法論が提案されている。特に最近では、プロセスの表現法を定めるプロセス中心の方法論[1]が注目されており、プロセスプログラミング[2]をはじめとして、エキスパートシステムによるAI方面からのアプローチ[3]、プロセス再利用を考慮したRe-engineering[4][5]など、様々な観点から研究が進められている。

プロセス中心の方法論に基づいた開発形態（プロセス指向ソフトウェア開発）が実現されると、開発作業そのものが明確に表現されるため、経験的作業の継承や、作業内容の評価／改善が容易に行なえるようになり、ソフトウェア開発の目標の二本柱である生産性の向上と品質の向上に大いに貢献できると考えられる。

本稿では、まず、交換ソフトウェア開発の現状を整理し、交換ソフトウェアプロセスの規定と、それを支援するプロセス指向SDEの必要性について述べる。ついで、プロセスの規定法とプロセス指向SDEの構築法を示す。最後に実際の交換プロセスへの適用法について述べる。

2．プロセス指向SDEの必要性

本章では、交換ソフトウェアと、その開発環境の特徴と問題点をあげ、これらを解決し、効率的開発への課題を整理するとともに、プロセス指向SDEの必要性について述べる。

2.1 交換ソフトウェア開発の現状

交換ソフトウェアは、リアルタイム、高信頼、大容量等の厳しい要求条件を満たすために、開発作業、プロダクトともに大規模とならざるを得ない。また、その作業は経験者のノウハウに依存するところが大きい。さらに、長期間にわたって機能追加を繰り返しながら、維持管理していく必要がある。このような特徴から、開発や維持管理時には、少数の経験者の下に経験の浅い人々が大勢集まって、次々と変化する環境の中で、試行錯誤の作業を行なっているのが現状である。

2.2 効率的開発への課題

2.1で述べた問題の根本的原因として、プロダクトを作成することに注目するあまり、開発作業の整理を疎かにしていた点と、開発環境がシーズ先行で更新されていくため、開発者が受け入れられない点の二点が考えられる。その結果、誰が、何を使って、どのように作業をすればよいかが規定されないままとなっている。つまり、この問題を解決するには、開発作業の三大要素である、人、環境、作業内容の関係を明確に整理し、ソフトウェアプロセスとして記述することが重要である。この整理には、プロセスに関する要求条件をまず明確にする必要があるが、その抽出にはプロダクトに対する要求条件からのフィードフォワードとプロダクトからのフィードバックによって開発対象となるシステムの特徴を十分考慮することが重要である。このようにソフトウェア工学の成果によるシーズと、対象となるシステムのプロセス要求条件からのニーズとを融合させて交換システムに適したソフトウェア開発プロセスの体系化を行なうこと（図1）が重要な課題である。このようなプロセスを構築するには、これをサポートする環境の役割が重要である。特に、様々な厳しい要求条件に基づいて開発される交換システムにおいては、特有の支援環境の構築は必須の課題であり、プロセスと密接に関係をもったプロセス指向SDEの構築が重要である。

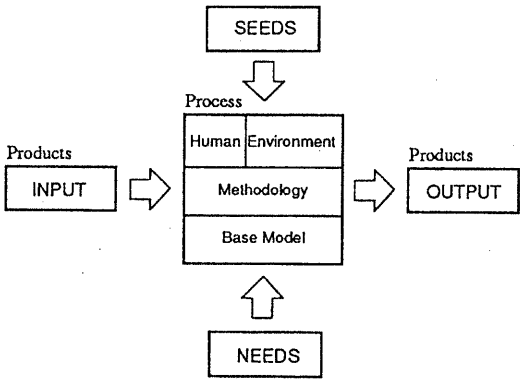

図1　ソフトウェアプロセス

3．プロセス指向SDEの構築

本章では、プロセスのベースモデルの選定と、プロセスを実際に規定するための手法[7][8]およびプロセス指向SDEの構築法[9]について述べる。

3.1 ソフトウェアプロセスモデル

交換ソフトウェアのプロセスおよびプロダクトの特徴の中で、長期にわたるライフサイクルをもつ点、多くの機能追加・変更が行なわれる点およびノウハウが整理されていない点などを考慮すると、プロセスおよびプロダクトを整理して、その内容を明確に規定し、これらを積極的に再利用することで、作業の効率化を図ることが重要である。このような点から、プロセスのベースモデルとして、ソフトウェア

の部品化と再利用を主眼としているRe-engineeringのモデルが適していると考えられる。このモデルでは、プロセスは、Forward-engineeringとReverse-engineeringを繰り返すループに時間軸を与えることで螺旋状のパイプで表現され、そのパイプの中をプロダクトが加工されながら流れていくモデルで表される。知識は各プロセス毎に作成されるプロダクト部品およびプロセス部品と、これらのさらに上位に位置する知識でプロセス生成のための人間的思考内容の集まりであるメタ知識の三階層から構成されている。

そこで、知識の部品化による収集という概念に基づいてプロダクトとプロセスの部品化を表現したRe-engineeringのモデルをベースとして、交換ソフトウェア開発プロセスの詳細化を進めていく。

3.2 ソフトウェアプロセスの規定法

3.2.1 プロセス構造化

プロセスを明確に整理するには、プロセスに関するシーズとニーズを取り入れ、開発プロセスの性質を分析することで、開発環境のベースとして素直に反映させる必要がある（図1）。

プロセスモデルに基づいて、実際の交換ソフトウェアプロセスを導出するには、プロセスをさらに詳細にする必要がある。そのプロセスで扱うプロダクトの分割単位や、作業の流れにしたがって、階層的にサブプロセスに分割していくことで、構造的なプロセスができあがる。各階層のソフトウェアプロセスは、図2のモデルのように、問題解決と形式変換の相互実行で実現されると考えられる[6]。このモデルにしたがって、プロセスを問題解決部分と、形式変換部分に明確に分離した環境を構築し、その環境に添った作業内容を記述することでプロセス詳細化への手順を導くことが可能となる。

Forward-engineeringのプロセスでは、階層的に分割された単位毎に、環境と人との関係を明確にし、自動化できるものと、人に依存するものとに整理する。その中で自動化できるものについては、それを支援するツールの環境設定条件や、実行手順等を明確に記述したシナリオを作成し、それを逐次実行するインタプリタを作成することで、自動化が実現する。また、そこで得られたシナリオが、プロセス部品となる。人間に依存する作業は、それを支援するツールの操作手順などを明確に記述することで、プロセス部品となる。Reverse-engineeringのプロセスでは、Forward-engineeringにおいて得られたプロセス部品を、プロセスの構造との関係に基づいて、階層的に整理することで、各階層レベルでのプロセス再利用単位を管理することができる。

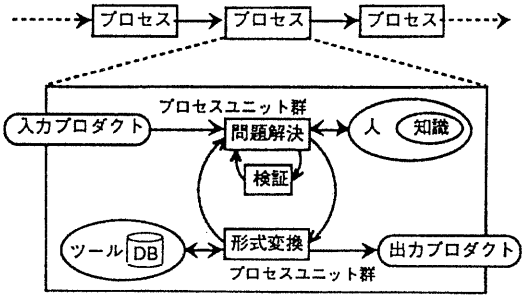

図2 ソフトウェアプロセスモデル

3.2.2 対象ドメイン指向

対象ドメイン指向とは、対象となるシステムを特定することで、そのシステムのハードウェアやプログラム構造に適合したプロセスを追及するとともに、そのプロセスにおける開発作法やツール類を導入しやすいハードウェアやプログラム構造を追及する概念である。対象システムを広くカバーしたプロセスやツール等は、全システムに共通する部分しかサポートできないため、詳細化レベルや機能レベルは低いものとならざるを得ない。しかし、対象システムの特徴をプロセスに反映させるとともに、プロセスを考慮したプロダクト構造をとることによって、プロセスの詳細化やツールの機能の向上が見られ、痒いところに手の届いた環境が得られることになる。

また、この考え方では、個々の作法やツール等に対して、そのサポート範囲を明確に規定することが要求される。これによって、そのプロセスが有効な範囲が明確になるばかりでなく、他のシステムへの移行の際に、修正すべき部分も明確になる。つまり、特定のシステム開発に適合したプロセスを追及することは、他システム開発への再利用を促進することになる。

3.3 プロセス指向SDEの構想

3.3.1 要求条件

プロセスの実行を支援するプロセス指向SDEを構築する上では、以下のような要求条件が考えられる。

（1）プロセス実行制御機能

プロセスの階層化によって最下位層の個々のプロセスを制御する上位層のプロセスが記述される。このようなプロセスの実行を制御する機能を実現することで、プロセスの自動化範囲が拡張される。

（2）ツールHMIの統一

これによって、開発環境の利用手法が明確になり、ツールの自動実行の際のシナリオ（自動化可能なプロセスを計算機が理解できる形式にまとめたもの）の記述法などが統一され、プロセスの部品化が促進される。

（3）対象ドメイン指向の適用

ツールが対象とするドメインを明確に規定することで、その適用範囲を明かにする。これによって、ツールの部品化がなされ、システムの仕様変更が行なわれた際に、システム依存のツール部品を交換することで、新たな開発環境が構築される。

（4）ツール間情報伝達方式の規定

これによって、ツールの協調動作が可能となり、これまで人間が介在してツール動作のタイミングをとっていた作業がシナリオ上で実現され、自動化が可能となる。

3.3.2 基本技術

前項で示した要求条件を実現するためのプロセス指向SDEにおける基本技術を以下に示す。

プロセス構造化によって、詳細に記述されたプロセスは、その個々のプロセスを実行するツールを作成することで、自動化が実現される。しかし、単純に個々の詳細なプロセス対応にツール化したのでは、プロセス間のインタフェース部分の考慮が欠けることになる。そこで、こうした個々のプロセスを順次実行させ、各種ツールを連動させる仕組みが必要である。これをProcess Integration方式（図3）と呼ぶ。

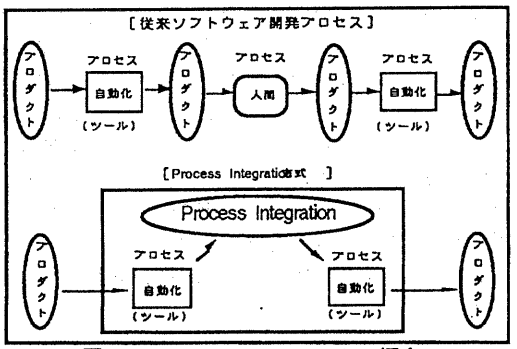

図3　Process Integration 概念

Process Integration方式の基本機能として、Process DriverとIntegration Platformの2機能が考えられる。シナリオとしてプロセスの内容が明記されたProcess Driverの指示に従って、Integration Platformがその配下のツールを制御し、プロセスを自動的に実行する。

このように、Process Driver、Integration Platform、各種ツール、そして対象システムによって、プロセス指向SDEは構成される（図4）。Process Driverは規定されたフォーマットで記述されたプロセスシナリオであり、プロセスの単位毎にIntegration Platformへ実行内容を送出する。プロセスシナリオには、プロセスとしてのツール動作のタイミング規定、ツールのオペレーション規定、入出力プロダクト規定、ツール動作条件規定等が記述される。また、Integration Platformはツールの連動を制御する。具体的には、配下に位置するツールとのインタフェースを規定することにより、Integration Platformのバックエンドにてツール環境の違いを吸収し、協調動作させる。また、各ツールの状態を一括管理することで、ユーザより指定されたプロセスシナリオのツールの実行順序チェックを行う。また、各連動ツール相互の動作タイミングの吸収を行う。また、状態矛盾の場合は次プロセスシナリオの実行への移行判定と、ツールの初期化などのエラー処理を行う。

以上のようなProcess Integration方式の実現により、3.3.1で挙げた要求条件が満たされ、以下のような効果が得られる。

（1）ユーザは、各ツールの組み合わせやオペレーションを考慮するだけで、ツール間の連動規定を意識する必要がない。

（2）各ツールを部品化することで、Process Driverの記述内容と、Integration Platformのデータ変更によって、新たな開発環境が容易に構築できる。

（3）各ツールを連動可能とすることで、開発環境全体を捉えたプロセスの自動化が実現される。

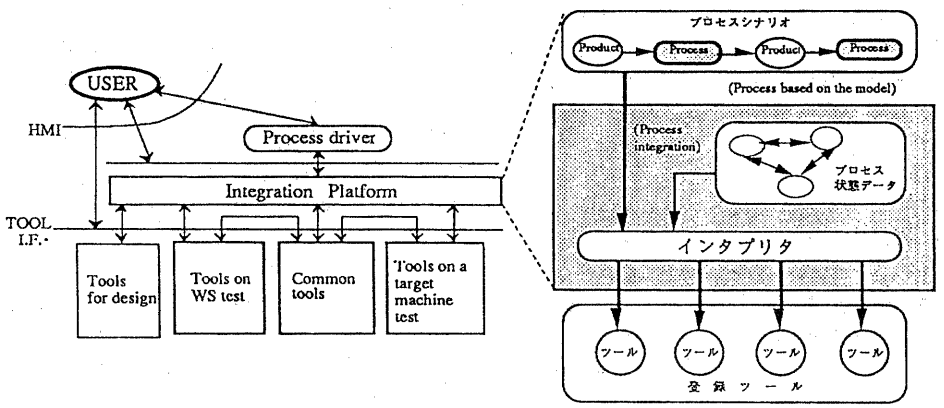

図4　Process oriented SDE

4．プロセス指向SDEの実現

本章では、交換ソフトウェア開発の中でも特に試験工程に注目し、プロセス指向SDEの構築のための要求条件を対象ドメイン指向の観点から整理するとともに、交換システムに密着したプロセス指向SDEを構成するツールを紹介する。

4.1 要求条件

交換ソフトウェア開発におけるプロセス指向SDE構築のための要求条件として、自動試験環境、WS上における試験、交換システム特有の機能、アーキテクチャ依存機能、プロセス部品化再利用を挙げる。

4.1.1 自動試験環境

人間が介在する作業を極力減らし、自動化することで、試験の効率化を図る。これには、試験内容を記述した手順書を理解し、各種ツールを制御するIntegration Platformが必要となる。

4.1.2 WS上における試験

交換システムでは、論理的処理に加え、各種ハード制御を含めたリアルタイム処理の確認が重要である。そこで、ターゲットマシンを用いたデバッグが必須となる。最近のプロセッサの飛躍的向上により、WSが比較的容易に入手できるようになり、分散環境でのソフトウェア開発が注目されつつある。

そこで交換システムにおいても、論理的処理などはできるかぎりWS上で試験を行い、リアルタイム処理などターゲットが必要な試験のみ、ターゲット上で行うことで、WS上での分散環境における試験が可能となり、試験効率の向上が期待できる。

要求条件としては、WS上におけるターゲット環境の擬似ツール、WS環境とターゲット環境でのツールの共用化およびHMIの統一などが挙げられる。

4.1.3 プロトコル試験機能

交換システム特有の機能として代表的なものに、信号授受の際に複雑なプロトコル仕様が各階層にあり、試験時には、準正常動作を含めると、極めて大量の試験項目を消化する必要がある点が挙げられる。これに対処するためには、各プロトコルレイヤ単位の試験を行うことが効果的である。そこで、このようなレイヤ毎に分割されたプロトコル信号を発生させる仕組みが必要となる。

4.1.4 アーキテクチャ依存機能

交換システムに要求される条件として、長期にわたる維持管理フェーズにおける機能追加が容易であること、超多重処理をリアルタイムにこなすために高速処理が可能であること、高信頼性を保持するために障害処理が充実していること等が挙げられる。

このような条件から、CPU、OS、ソフト構造などは特別なものを用いる必要がある。これに対処するため、このような特別なアーキテクチャに対応した機能が必要となる。

4.1.5 プロセス再利用

一度作成したプロセスを再利用することで、プロセスの効率化が図れる。具体的には、ソフトウェア変更時のデグレードチェック機能などを充実することで、プロセスの有効活用が可能となる。

4.2 プロセス指向SDEの実現[10]

前項で示した要求条件を満たすためのツールとして、表1のようなものが挙げられる。こうした要求条件を十分に満たし、高機能なツールとして実現するには、多かれ少なかれシステムに依存する機能や構造をとる必要があり、対象ドメイン指向の概念が重要になる。

以下にこれらの概要について述べる。

表1 ツール一覧

要求条件	対応ツール
[自動試験]	シミュレータ
[機械作業の自動化] プロトコル試験機能 アーキテクチャ依存機能	 イベントジェネレータ 試験結果収集／解析
[プロセス再利用]	リグレッションテスタ

4.2.1 自動試験環境をサポートするツール

3.3.2項で示したIntegration Platformそのものである。ユーザが記述した試験シナリオを解釈し、逐次実行する。その際には、次項以降に挙げるような、Integration Platform配下に存在する様々なツールの実行制御を行う。各ツールのHMIはこのIntegration Platformで統一され、また、ツール間のインタフェースも統一される。図5にSDE全体の概観を示す。

4.2.2 WS上における試験をサポートするツール

WS上での試験を行う際に必須の機能として、ターゲットのシミュレート機能が挙げられる。具体的には、ターゲットのプロセッサを擬似するCPUシミュレータ、OSを擬似するOSシミュレータ、ハード装置を擬似するハードシミュレータ等が挙げられる。図6にその概要を示す。これによって、単体試験から、論理処理確認の結合試験までをWS上でカバーすることが可能となり、マシン時間、開発ロケーションに制約されることなく、効率的に試験が行えるようになる。

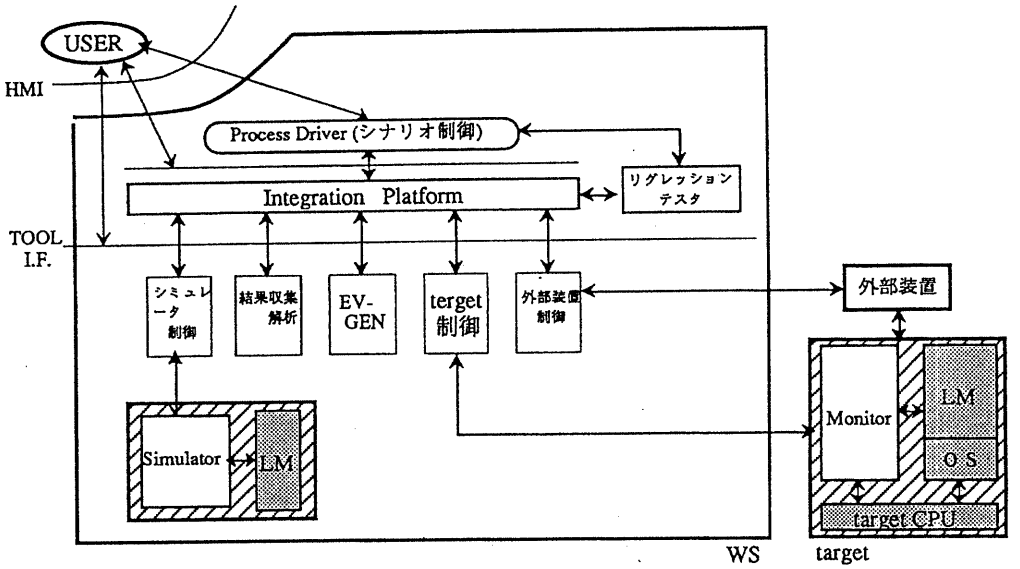

図5　交換ソフトウェアSDEの概観

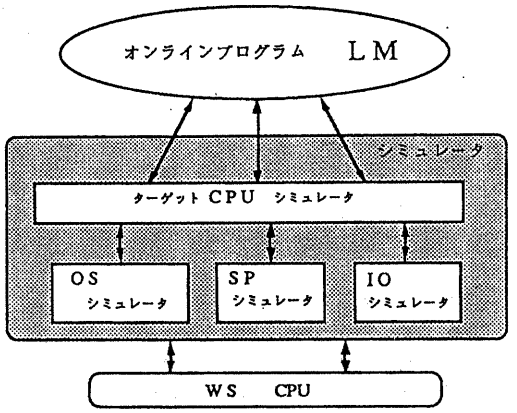

図6　シミュレータ概要

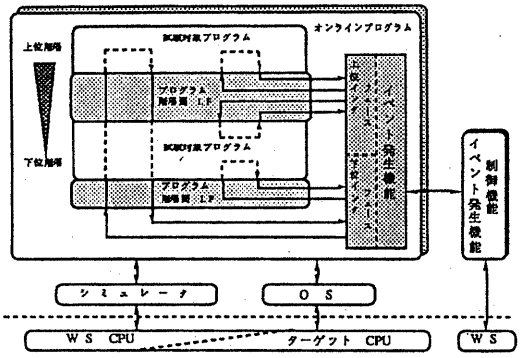

図7　イベントジェネレータ概要

4.2.3　プロトコル試験機能をサポートするツール[11]

交換システムでは、試験対象プログラムに対して関数コールや各種プロトコル毎のイベントを発生する機能が要求される。従来は試験対象プログラムに対してターゲットマシン外よりハード擬似呼装置を利用して結合試験を行っていた。これに加え、該当アプリケーションプログラムのイベント挿入インタフェースを明らかにし、各インタフェースからイベントを発生させるツールを作成することによって、プログラムの任意のポイントにおいてイベント発生が可能となり、多様なプログラム単位での試験が可能となる（図7）。

4.2.4　アーキテクチャ依存機能をサポートするツール

試験時における重要な作業として、試験結果データの収集と解析が挙げられる。このデータは、システムのアーキテクチャに大きく依存するため、これをサポートする特有のツールが必要となる（図8）。

データ収集ツールは、対象プログラム内にトレーサを埋め込み、時系列にデータを収集する方式と、あるプログラム状態をダンプし、一括して収集する方式とが考えられる。どちらの方式においても、収集すべきデータを特定するためには、ツール自体がアーキテクチャを十分理解しておく必要がある。

結果解析ツールは、収集されたデータを解析し、結果を判定できる形態に加工し、さらに正解値と照合することで合否の判定を行うツールである。これにより、メッセージシーケンスや状態遷移などをビジュアルに表示し、自動的に判定することができる。

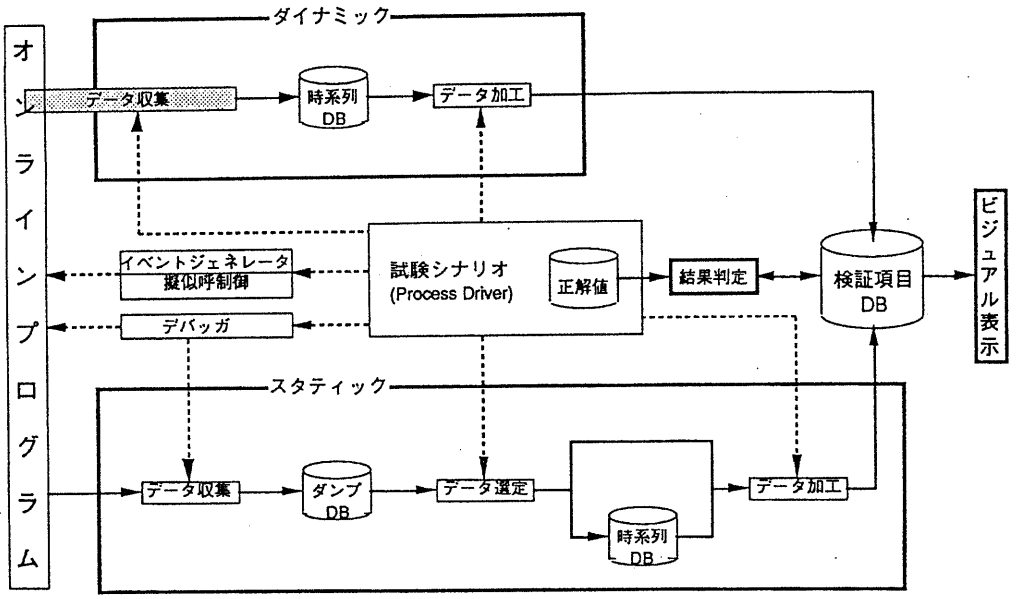

図8　試験結果解析全体概要

4.2.5　プロセス再利用をサポートするツール

自動試験環境が実現されると、試験を行う手順は、全てツールが理解できるシナリオとして記述される。このシナリオは一つの詳細な試験プロセスと位置づけられる。これを積極的に再利用することによって試験効率の向上が期待できる。有効な再利用法として、交換ソフトウェアにおいて頻繁に行われる機能追加の際に、既存機能のでグレードチェックが考えられ、これを実現するツールとしてリグレッションテスタが挙げられる（図9）。

リグレッションテスタは前項までに挙げた全てのツールが協調動作することで実現される。基本機能としては、上記各機能に加え、試験項目／シナリオ管理機能が挙げられ、これらの機能を統合することで実現される。

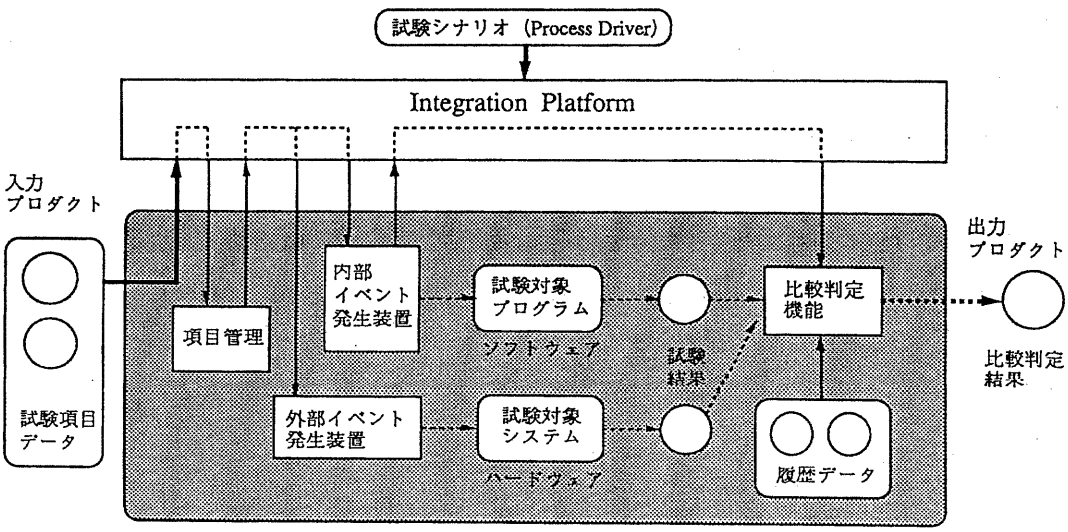

図9　自動試験機能を適用したリグレッションテスタ

5．まとめ

今回の報告では、交換ソフトウェア開発におけるプロセス導出のための方法論に基づいて、プロセスを支援するプロセス指向SDEを提案した。その内容をまとめると以下のようになる。

（1）交換ソフトウェア開発の効率化には、明確なプロセスの規定とそれを支援するSDEの構築が重要である。

（2）交換ソフトウェアプロセス構築法としては、プロセス構造化、対象ドメイン指向の考えが重要である。

（3）Process Integration方式を適用した開発環境構築により、自動化が実現できる。

（4）システムに依存した機能は、対象ドメイン指向開発環境によって実現できる。

（5）プロセスをシナリオ化して管理することによって、自動化および再利用が可能となる。

（6）交換システムSDEを構成する主なツールには、シミュレータ、イベントジェネレータ、試験結果収集／解析、リグレッションテスタ等があげられる。

今後は、プロセス指向SDEを構築するとともに、実際の開発に適用し、生産性、汎用性の面から評価を行う予定である。

参考文献

1) Watts S. Humphrey : "Characterizing the Software Process: A Maturity Framework", IEEE Software, Mar. 1988, pp.73-79

2) Osterweil, L. : "Software processes are software too", Proc. of 9th ICSE, pp.2-13

3) 山口, 落水："事例ベース推論とモデル推論の相互作用に基づくソフトウェアプロセスモデル獲得支援環境", 知能ソフトウェア工学の動向と展望シンポジウム論文集, pp.75-81, 1992

4) 立元, 白石："Re-engineeringを適用した交換ソフトウェア開発プロセスに関する一考察", SSE 90-110, pp.75-81

5) 立元, 白石："Re-engineeringを適用した交換ソフトウェア開発プロセスにおける信頼性改善の一手法", 信学全大春季B-525, 1991

6) 白石："開発プロセスに着目した交換プログラム開発支援手法に関する一考察", 信学全大春季B-515, 1990

7) 岩本, 白石："交換ソフトウェア自動試験に関する一考察", 信学全大春季B-515, 1990

8) 立元, 白石, 黒田, 清野："交換ソフトウェアプロセスに関する一考察", SSE92-6 pp.1-6

9) 岩本, 立元, 白石："交換ソフトウェア用プロセス指向SDEプラットフォームの検討", 信学全大秋季, 1992 投稿予定

10) 立元, 白石, 黒田："交換ソフトウェアプロセス定義法および評価法の検討", 信学全大秋季, 1992 投稿予定

11) 高田, 白石, 岩本, 鹿内："交換ソフトウェア試験高度化の検討", 信学全大秋季, 1992 投稿予定