三菱電機技報 84巻5号(通巻948号)

Changes for the Better

MITSUBISHI

三菱電機技報

Vol.84 No.5

2010 5

特集「ソフトウェア開発環境」

eco Changes

家庭から宇宙まで、エコチェンジ

ISSN 0369-2302 CODEN:MTDNAF

三菱電機技報 Vol. 84 No.5 2010

目　次

特集「ソフトウェア開発環境」

Software Development Environment

特許と新案

表紙：ソフトウェア開発環境

三菱電機は，製品・システム開発におけるソフトウェア開発の増加をうけ，ソフトウェアの高品質化と開発の効率化を目的に，ソフトウェア開発プロセスの改善，効率的なソフトウェア開発を支援する開発環境の整備，ソフトウェアプロダクトラインによる開発量の縮減に取り組んできた。

表紙は，"01"の数字やパソコンでソフトウェアを，ミーティングの様子は快適で確実な開発環境を，地球に降り注ぐ光や地球を取り巻くラインによって，製品に組み込まれ広く利用されるソフトウェアやそれによってもたらされる情報の流れをそれぞれ表現した。

巻/頭/言

ソフトウェア開発環境特集号に寄せて

Foreword to Special Issue on Software Development Environment

西田正吾
Shogo Nishida

私の研究分野は，ヒューマンインタフェース（HI），ヒューマンコミュニケーション（HC）である。私自身は，もともとは“システム屋”であり，大規模システムの制御や運用，計画等の手法の研究を行ってきたが，“システムの規模の増大に伴う複雑性管理の問題”を解決するにあたって，“人間と機械の接点において，人間は人間の得意なことを，また機械（コンピュータを含む）は機械の得意なことを行い，それぞれの結果をうまく統合する”ことで解決したいと考え，人間に関わる研究に携わるようになった。つまり，システム屋の立場から，HI，HCの分野の研究に取り組んできたわけである。

そのような観点で“ソフトウェアの開発プロセス”をとらえると，“設計における意図の伝達”“他者が設計したものの理解支援”“設計における上流工程の支援”など多くの問題を内包している。一例として，“設計における意図の伝達”の問題を取り上げると，通常，組織としてソフトウェア開発を行う場合には，大きなものでは100人を超えるプログラマが開発に関わることになるが，開発チームとテストチームの間のデバッグのフェーズにおける意図の伝達は大きな問題である。また，システム稼働後は，メンテナンスやシステム改良は保守チーム（通常は開発チームに比べ，かなり小規模になる）へ手渡されるのが普通であるが，開発チームと保守チームの間のソフトウェアに関する意図の伝達は，非常に重要かつ困難な問題となる。このような問題は，一人のスーパーマンが，概念設計からコーディング，テスト，メンテナンスまですべてをやることができれば存在しなくなるものであるが，組織としてソフトウェアを開発する場合には避けて通れない。

そこで，このような問題点を踏まえたうえで，ソフトウェア開発環境における必要機能について，特に筆者が重要であると思っているものをリストアップしてみる。

①人にとって見やすい，理解しやすい視覚的な開発環境の構築
②通常はブラックボックスになってしまう設計意図の記述や伝達が可能な枠組みの提供
③プロジェクトマネージメントの向上のための方策，例えば，開発過程のモニタリング，アラーム機能の充実，チーム内の認識や理解のギャップ減少のためのコミュニケーション支援手法など
④ソフトウェア資産の再利用による生産性の向上
⑤急速な技術進歩への対応が可能なソフトウェア開発環境の整備，特に端末やソフトウェア環境が変わっても容易に対応が可能な開発環境の構築が重要
⑥概念設計などの上流工程の支援並びに上流工程とプログラミングを行う下位工程の連携

このような機能実現の一つの有力な方法が，対象を限定したうえで，ソフトウェア開発環境をあるモデルに基づいた枠組みの上で構築することであろう。この場合，ある程度対象を絞って，その性質も取り入れた形でうまく構造化することが重要である。また，設計者は制約があることも認識して，その枠組みを利用する必要がある。特に，対象の選び方と構造化の度合いのバランスをうまくとることが必要で，対象についてはビジネス規模等も考慮した上で，うまく選ぶことが重要であると思われる。

最後に，ソフトウェア開発に関する組織風土作りについて触れておきたい。以前から，ソフトウェア製作能力に関しては日米格差ということが言われているが，私自身は長年にわたる日本の“ものづくり”におけるハード優先の考え方がそのベースにあるのではないかと感じている。つまり，日本においては，ソフトウェアの価値，特に“ものづくり”における位置付けがハードウェアに比べてどうしても低く見られがちである。しかし時代は変わり，これからはソフトを制するものが“ものづくり”を制する時代が来ようとしている。このような時代においては，ソフトウェア開発者を鼓舞する意味でも，ソフト重視の組織風土を創っていく必要があると思う次第である。

◆大阪大学　副学長　　Vice President, Osaka University

巻頭論文

ソフトウェア開発環境の現状と展望


山下昭裕*　木槻純一**

Software Process Improvement and Development Environment

Akihiro Yamashita, Junichi Kitsuki

要　旨

社会システムから家電製品に至るまで，機能の高度化が進展するとともに，製品投入サイクルも短期化しており，従来ハードウェアで実現されていた機能が幅広くソフトウェアで実現されるようになっている。これに伴って，製品・システム開発におけるソフトウェア開発の位置付けが重みを増している。こうした観点から，三菱電機は，従来ソフトウェアの高品質化と開発の効率化を目指して，ソフトウェア開発プロセスの改善，効率的なソフトウェア開発を支援するソフトウェア開発環境の整備，ソフトウェアプロダクトラインによる開発量削減などに取り組んでいる。

この特集号では，ソフトウェアの生産性向上，高品質化に向けた，ソフトウェア開発環境に関する三菱電機の取組みについて，次の4つのポイントに整理して紹介する。

(1)　要件定義・仕様設計の高品質化

ビジネスモデルに基づいて要件定義を行うとともに，UML(Unified Modeling Language)(注1)による仕様記述によって，漏れの防止と仕様の高品質化を図る。

(2)　ソフトウェアの実装効率化

形式的な仕様記述と，分野別のソフトウェア基盤を活用し，実装の効率化とともに，実装誤りを防ぐ。

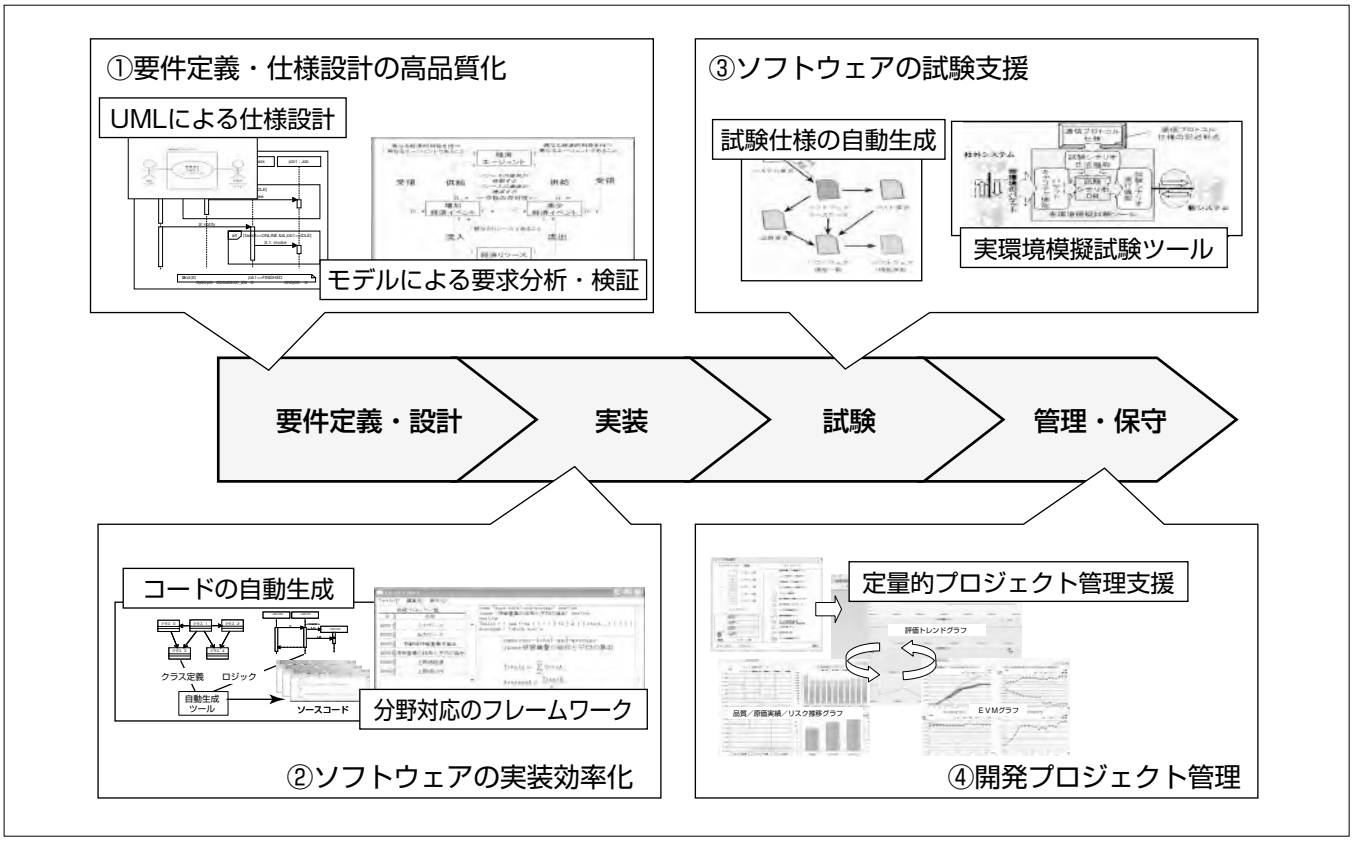
(3)　ソフトウェアの試験支援

仕様記述から，試験条件を自動抽出するとともに，自動試験環境を構築して試験の効率化を進める。

(4)　開発プロジェクト管理

ソフトウェア開発プロジェクトを“見える化”する。

（注1）　UMLは，Object Management Group Inc.の登録商標である。



三菱電機におけるソフトウェア開発環境への取組み

家電製品から大規模な社会インフラシステムに至るまで，三菱電機の幅広い製品群で，機能の実現・製品開発におけるソフトウェアの比重が高まっている。高品質のソフトウェアを効率よく開発するために，ソフトウェア開発の上流である要件定義・設計から，ソフトウェアの実装，ソフトウェア試験，保守開発に至る各段階に対して，開発プロセスの改善とソフトウェア開発環境の改良を進めている。

*設計システム技術センター長　**情報技術総合研究所 マルチメディア技術部門統括

1. まえがき

社会システムから家電製品に至るまで，機能の高度化が進展するとともに，製品投入サイクルも短期化しており，きめ細かい制御を実現するため，従来ハードウェアで実現されていた機能が幅広くソフトウェアで実現されるようになっている。また，インターネット化の進展に伴い，従来のクローズなシステムと異なるソフトウェア構造，ユーザーインタフェース(UI)を持つシステムが増加している。こうした観点から，ソフトウェア開発の効率化を目指して，ソフトウェア開発プロセスの改善，効率的なソフトウェア開発を支援するソフトウェア開発環境の整備が必要となってきている。

2. ソフトウェア開発環境の動向

事業競争力の強化のためには，製品の差別化・付加価値向上が必要であり，ソフトウェアの重要性はますます高まっている。

それに伴い，ソフトウェアが大規模化，複雑化し，また事業のスピードアップのために短期開発のニーズも強く，ソフトウェアの開発力強化が危急の課題となっている。一方，電力・交通などの社会インフラ，携帯電話・AV機器など生活必需品となったソフトウェアの品質は重要となり，ソフトウェアトラブルが社会生活に多大な影響を及ぼすことも多くなっている。

こうしたソフトウェアの課題に対し，経済産業省はIPA／SEC((独)情報処理推進機構 ソフトウェア・エンジニアリング・センター)を2004年10月に設立し，ソフトウェア開発力強化に取り組んでおり，三菱電機も設立時から参画している。

ソフトウェア開発力強化には，プロセス，技術，人のバランスよい強化が必要である。

(1) プロセス

プロセスとは，ソフトウェアを開発するための手順や工程，成果物，進め方に関する基本的な考え方を定義したものである。プロセスを改善する手法・モデルとして，CMMI(Capability Maturity Model Integration)[1]やISO/IEC15504，共通フレーム2007(Software Life Cycle Processes：SLCP)などがあり，アセスメント手法も整備され普及している。またプロジェクト管理面ではPMBOK(Project Management Body of Knowledge)[2]によって管理手法が体系的にまとめられている。

(2) 技術

要件定義から設計，実装，テストの全工程にわたって，モデリング手法が整備され実用化されている。分析や設計段階ではUMLによるモデリング手法が実用化されており，モデル検証の研究もされつつある。一方，ソフトウェアの再利用技術としては，SPL(Software Product Line)型開発によって製品ロードマップに沿った再利用型開発手法が実用化されつつある。

(3) 人

人材育成の観点から，ETSS(Embedded Technology Skill Standards)などのスキル標準，キャリアアップ体系が整備され，企業への導入が加速されつつある。

3. 三菱電機の取組み

三菱電機でも，1980年代からエンタプライズ系のソフトウェア開発力強化活動を推進してきたが，組込み系ソフトウェアにおいて**2章**で述べた課題が大きくなってきたことから，両分野を合わせたソフトウェア開発力強化活動を2004年から全社的に展開している。この活動には，ソフトウェアを開発する事業所だけでなく，ソフトウェア開発を担当するグループ会社，ソフトウェア開発の先進技術を研究する研究所・技術センターも参画している。

この活動では，開発力強化のねらい(施策)として“フロントローディング化”に注力している。フロントローディング化とは，開発工程の上流設計段階で品質を作り込む手法である。フロントローディング化によって，上流での課題解決による手戻り削減と開発期間短縮が図れる。

三菱電機では，フロントローディング化を次の2段階で進めている(**図1**)。

(1) 第一段階

ソフトウェアの開発プロセスに着目し，従来，テストなどの後工程で発見していた不具合を要件定義や設計など上流工程で検出する活動である。手法としては，CMMIやISO15504プロセスモデルに従ったプロセスの定義と，定量データの収集・評価(見える化)を行う。設計不具合は設計レビューで検出し，実装の不具合は静的解析技術を使って検出する。

(2) 第二段階

ソフトウェア設計技術に重点を置く。不具合を検出するのではなく，不具合の混入を防止する技術である。そのために，モデルベースのソフトウェア設計・開発や，製品分野に合わせたソフトウェア共通プラットフォームの開発・利用やSPL手法を活用している。

第一段階，第二段階ともに，開発に携わる人の育成が重

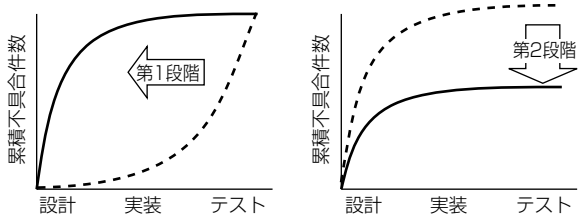


図1．フロントローディング化

要であり，スキル教育だけでなく，特にソフトウェアプロジェクトリーダーの育成に力を入れている。

4章以降に，ソフトウェアの開発環境に関する三菱電機の取組みを次の4つのポイントに整理して述べる。

⑴ 要件定義・仕様設計の高品質化

ビジネスモデルに基づいて要件定義を行うとともに，UMLによる仕様記述によって，漏れの防止と仕様の高品質化を図る。

⑵ ソフトウェアの実装効率化

形式的な仕様記述と，分野別のソフトウェア基盤を活用し，実装の効率化とともに実装誤りを防ぐ。

⑶ ソフトウェアの試験支援

仕様記述から，試験条件を自動抽出するとともに，自動試験環境を構築して試験の効率化を進める。

⑷ 開発プロジェクト管理

ソフトウェア開発プロジェクトを“見える化”する。

4. 設計の効率化・支援

ソフトウェアの開発で，システムが果たすべき機能・仕様を早い段階で確定・検証することが，ソフトウェア開発全体の効率化に寄与する。近年，オブジェクト指向によるソフトウェア要求分析・設計が広く採用されており，そのための仕様記述方法として図形言語であるUMLが使われる。特に，要求分析段階ではユースケース図が，設計段階ではクラス図，シーケンス図，状態マシン図などが使われている。さらに，対象分野に適した要求仕様モデルを使うことによって，要件定義の容易化とともに，仕様の完全性を検証する取組みや，仕様段階でソフトウェアの動作を模擬して検証するモデルベース開発などの取組みも広がってきている。

4.1 要件定義の高品質化・効率化

ソフトウェアの要件定義では，UMLのユースケース図を使う。しかし，ユースケース図は記述の自由度が高いため，記述レベルが書き手によって異なったり，要件定義に漏れが生じたりする可能性がある。そこで，記述規則やそれに対応したツールを用いて仕様定義を行ったり，機能モデルに基づいて仕様記述を行うことで，要件定義の高品質化を図る。

この特集号では，要件定義の高品質化への三菱電機の取組みとして，システム化の対象をREA(Resource Event Agent)モデルを活用してモデル化し，システム機能仕様を明確にする手法を紹介している[3]。

また，設計品質の向上を目的として，UMLで記述された設計情報を網羅的に検査するツールを開発した。このツールは，“形式手法”の一つであるモデル検査技術を採用しており，オートマトン／時相論理／グラフ理論を用いて，ある範囲のシステムの動作を網羅的に検査する。このツールによって，サーバ間のメッセージシーケンス，Webサービスの呼出しシーケンスや通信プロトコルなどの仕様を高品質化することができた[4]。

4.2 要件定義・仕様設計における今後の課題

性能・可用性・信頼性といった非機能要求を形式的に記述することは，従来の手法では難しい。

この課題に対して，三菱電機インフォメーションシステムズ㈱(MDIS)も参加した非機能要求グレード検討会が，非機能要求を明確化するためのガイドラインと要求項目一覧表[5]などを公表している。これは非機能要求を，可用性，性能・拡張性などの6種に分類し，各要求の水準を段階的に数値化することで，システムの非機能要求を“見える化”する。

ただ，こうした非機能要求を，詳細仕様設計や実装に反映する方法とソフトウェア開発環境については今後の課題となっている。

5. ソフトウェアの実装効率化

5.1 ソフトウェアの再利用

ソフトウェアの再利用による開発効率化は，ソフトウェア開発の初期から様々に試みられてきた。

⑴ 流用

最も簡単であるが，単純な流用では，派生コードが複数できてしまうなど，版の管理や保守に問題が生じやすい。また，流用を繰り返すことでコードが複雑化し，品質や性能が低下するとともに機能拡張の開発効率が低下することもある。

⑵ 共有ライブラリ

複数のプログラムから使われる機能を，ライブラリとしてまとめて提供する。文書処理など共通性の高い分野では有効である。一方，システムごとに要求仕様が異なるビジネスロジックなどではライブラリ化は進んでいない。

⑶ フレームワーク

ライブラリでは，全体の流れをソフトウェア開発者が決める。これに対して，全体の流れをフレームワークとして共通化し，システム固有の処理部分だけを開発する手法が出現した。例えば，Webサーバの処理は，クライアント側からのページ要求に対して，HTML(Hyper Text Markup Language)ページを作成して返送することである。そこで，要求の受付と返送の処理をフレームワーク化し，ページ作成部分だけを開発すればよいようにしたのが，Apache Strutsなどである。

⑷ ソフトウェアプロダクトライン(SPL)

家電などの組込みソフトウェアの場合，仕様の異なる多くの機種に向けて短期間で開発する必要がある。ここで使われるのが，SPLである。対象分野に合わせてソフトウェアを機能領域に分割し，各領域で異なった仕様を実現する

コンポーネントをあらかじめ用意する。実際の機種開発では，用意されたコンポーネントを組み合わせるか，一部だけを新規に開発してソフトウェアを開発する。

この特集号では，空調機器制御ソフトウェアの事例[6]，などを紹介している。

5.2 ソフトウェアの自動生成

モデルを活用して形式的に仕様を記述することで，ソフトウェア仕様の品質を向上させるだけでなく，コード自体を仕様から自動的に生成し開発を効率化することができる。UMLクラス図からクラスのスケルトンコードを生成するだけでなく，UIのような定型的なソフトウェアについては，専用ツールや定義ファイルを用いることで，ほぼプログラミングなしにソフトウェアを開発することができる。この特集号でも，クラス図とシーケンス図を用いて，Webアプリケーションのビジネスロジックを自動的に生成する取組み[7]を紹介している。

また，分野対応にソフトウェアを自動的に生成する手法も検討しており，複雑な科学技術計算ソフトウェアを，数式エディターを使って開発する事例[8]も紹介する。

将来に向け，Webブラウザの機能拡張と同様に，分野対応のフレームワークとアドオンモジュールとを組み合わせてソフトウェアを構築する手法についても研究を進めている(**図2**)。この手法では，アドオンモジュールを機器定義データから自動的に作成し，最小限の操作手順だけを簡易言語で指定することで，開発量を大幅に削減できる可能性がある。

6. ソフトウェアの試験支援

ソフトウェア開発で最も時間とコストを要するのは，試験である。**図3**に，経済産業省が公開した組込み機器開発における工数比率[9]を示す。全工数の半分がシステム設計とソフトウェア開発であり，ソフトウェア開発の3分の2を試験工数が占めていることが分かる。

実装については，既存コードの流用，コードの部品化や仕様からの自動生成などによって，開発量を削減することが可能となってきている。一方試験については，一部のコードの変更が全体の動作に影響を与える可能性があるため，機能追加や変更のたびに，すべての試験を再実行する必要がある(回帰試験，Regression Testing)。このため，コードの生成を自動化しただけは，ソフトウェア開発コストを大幅に低減することはできない。

そこで三菱電機では，次のような試験の自動化・効率化にも取り組んでいる。

(1) システム仕様から試験仕様，試験条件，試験プログラムなどを自動生成する仕組みの構築
(2) 試験をシナリオに従って実行するとともに，試験結果を自動的に生成する仕組みの構築
(3) プラントなどの監視制御システムの試験を，実機なしに実行するためのプラント・シミュレータの構築

この特集号では，Webアプリケーションのセキュリティをアプリケーションソース(Webページデータ…

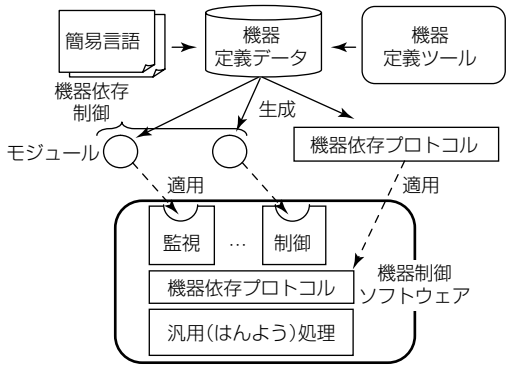


図2．モジュールによる機器への対応，機能拡張

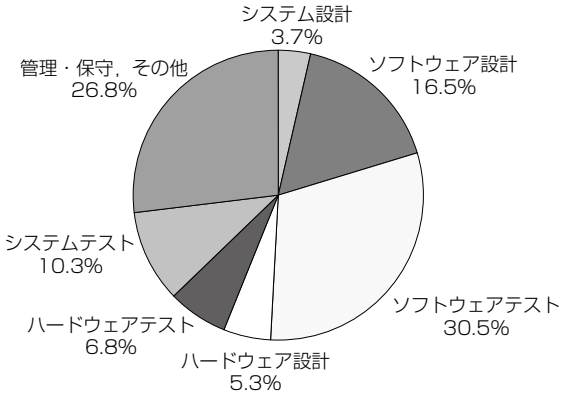


図3．組込み機器開発における工数比率[9]

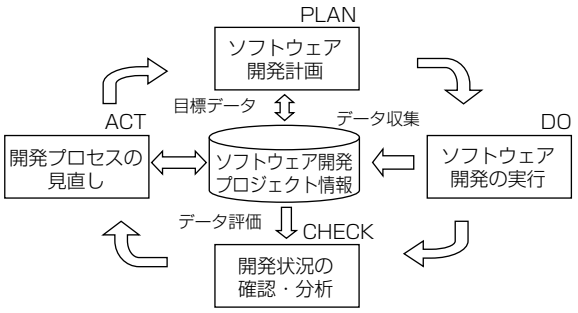


図4．開発プロジェクト管理のPDCAサイクル

るリスクを最小化することである。このため，プロジェクトの各段階で，仕様書やコードの開発量，レビューの実施状況・指摘件数，試験の密度・網羅率やバグ発生状況などを定量的に収集し，これらのデータを評価・分析することによって，工程の遅延，コストオーバーラン，品質の劣化などをプロジェクトの途中段階で“見える化”することができる。

この特集号では，MDISが開発した，CMMIレベル4相当の定量的プロジェクト管理システム“P-Support”[12]について紹介している。このシステムは，プロジェクト・ポータルから計画・実績データを自動的に取り込み，プロジェクトの品質／コスト／工程／リスクの評価を行う。また評価結果やそのトレンドを，プロジェクトダッシュボード上に“見える化”しており，これによって，管理負荷を増やすことなく，経営者から開発者までプロジェクトの状況を容易に把握し，問題への迅速な対処が可能となっている。

8. むすび

ソフトウェアの高品質化と低コスト化に向けたソフトウェア開発の動向と，三菱電機における取組みについて述べた。組込みソフトウェアから大規模システムに至るまで，ソフトウェア機能の増大と複雑化による開発規模の増加は，今後も継続していくと考えられる。また，これに合わせて，ソフトウェア開発の上流から下流まで新たな手法が導入されていくと考えられる。三菱電機におけるソフトウェア開発量も毎年増加しており，今後もソフトウェア開発の改善に向けて，開発プロセス・手法の改良と開発環境の改善を継続していく。

参考文献

(1) Carnegie Mellon University Software Engineering Institute：成功するソフトウェア開発(CMMによるガイドライン)，㈱オーム社（1998）
(2) Project Management Institute：A Guide to the Project Management Body of Knowledge-Fourth Edition，Project Management Institute （2008）
(3) 篠崎　衛，ほか：システム構築上流設計手法の改善，三菱電機技報，**84**，No.5，279～282（2010）
(4) 上野浩一郎，ほか：高信頼性を実現するシステム仕様検査技術，三菱電機技報，**84**，No.5，295～298（2010）
(5) 非機能要求グレード検討会：システム基盤の非機能要求に関する項目一覧（2010）
http://www.nttdata.co.jp/nfr-grade/result.html
(6) 大河原　繁，ほか：空調機器制御ソフトウェアの再利用開発，三菱電機技報，**84**，No.5，311～314（2010）
(7) 原田雅史，ほか：Webシステム開発フレームワーク，三菱電機技報，**84**，No.5，271～274（2010）
(8) 井上勝行，ほか：数値計算プログラムのビジュアルな構築環境，三菱電機技報，**84**，No.5，299～302（2010）
(9) 経済産業省：2008年版組込みソフトウェア産業実態調査報告書―プロジェクト責任者向け調査―
http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2008software_resarch/project_houkokusho.pdf
(10) 河内清人，ほか：ハイブリッドセキュリティ診断技術，三菱電機技報，**84**，No.5，291～294（2010）
(11) 川崎将人，ほか：システム試験の効率化／高精度化技術，三菱電機技報，**84**，No.5，287～290（2010）
(12) 坂田賢志，ほか：定量的プロジェクト管理支援システム“P－Support”の開発と試行―定量データに基づくプロジェクト状況の把握への取組み―，三菱電機技報，**84**，No.5，283～286（2010）

特集論文

Webシステム開発フレームワーク

原田雅史* 河村美嗣*
倉持和彦*
石井 洋*

Application Development Framework for Web-based Systems

Masafumi Harada, Kazuhiko Kuramochi, Hiroshi Ishii, Yoshitsugu Kawamura

要 旨

Webシステムが，技術革新によって従来の情報系システムから企業などの基幹系システムの構築に用いられるようになり，顧客から高品質・低コスト・短納期での構築が求められるようになった。これらの要求を満たすために，三菱電機では早期からフレームワークの開発に取り組んでおり，これまで事業本部対応で，“BizFrame”“TRIWALQ”及び“MIWESTA（MDIS Web Development Standard）”の開発を進めてきている。これらは，多数のWebシステム構築に使用され10～30%程度の生産性向上を達成している。

当社のフレームワークは，生産性向上や品質向上を実現するために“実行環境”だけでなく，ソースコード自動生成ツールなどの“開発環境”や“開発プロセス”も合わせて提供している。また，事業特性に合わせてベースとなるフレームワークに対して様々な機能強化を実施した“実行環境”を開発している。

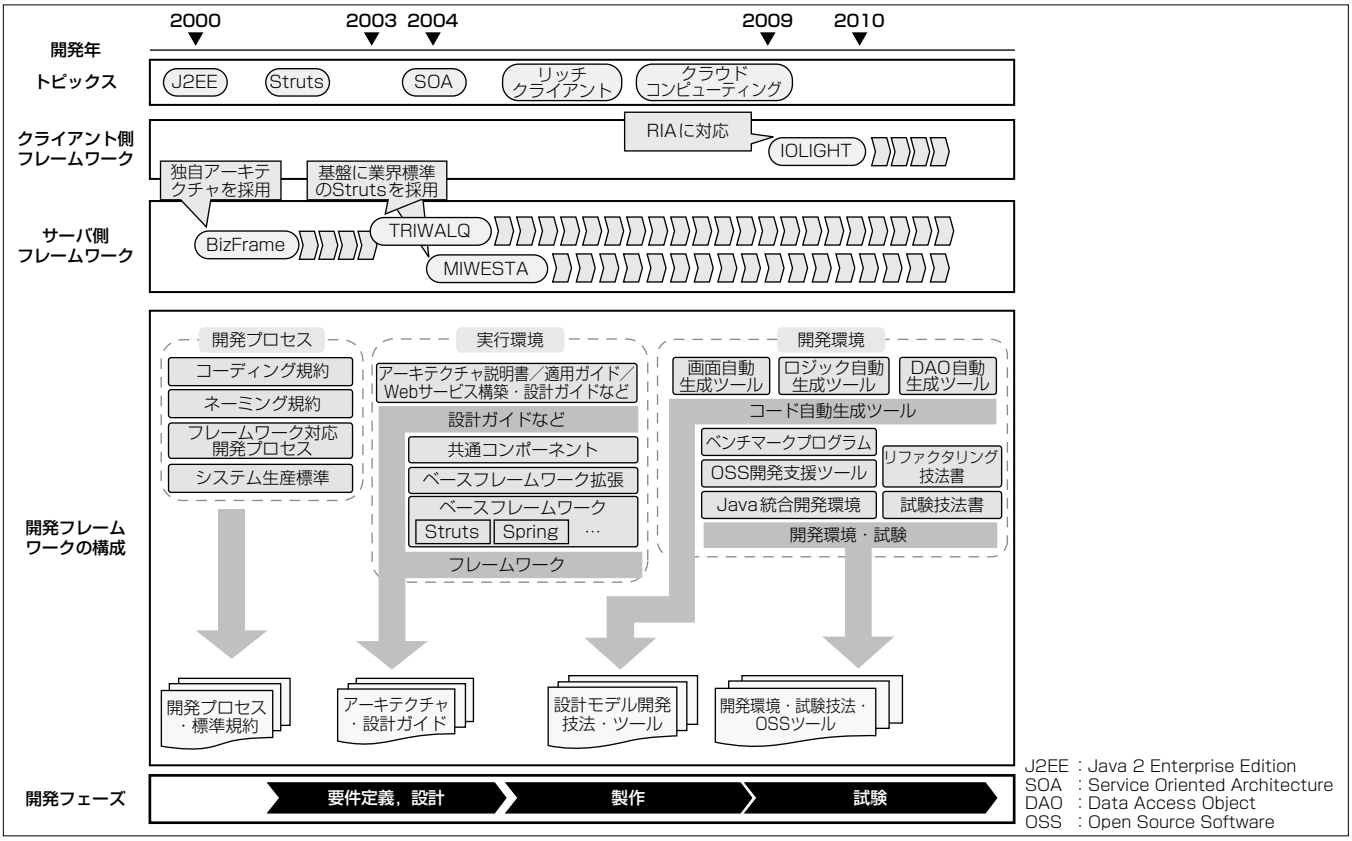
本稿では，次の3つの取組みについて述べる。

(1) CoC（Convention over Configuration）(注1)概念を適用して，設定情報の記述量を削減可能なサーバ側の開発フレームワーク

(2) UML（Unified Modeling Language）(注2)で記述された設計文書を入力とし，生産性向上を実現するソースコード自動生成ツール

(3) RIA（Rich Internet Application）技術を適用して，クライアントサーバシステムと同等の応答性能を実現するクライアント側の開発フレームワーク

（注1） 命名規則などに従うことで，フレームワークに対する設定情報の記述量を削減する概念。
（注2） UMLは，Object Management Group Inc. の登録商標である。



開発フレームワークのロードマップと構成

当社では，Webシステムの黎明（れいめい）期から開発フレームワークに着目し，システム構築時の生産性・品質・保守性を向上するために継続的に開発に取り組んでいる。当初は独自アーキテクチャであったが，業界標準フレームワークであるStrutsを取り入れオープン化を実現した。フレームワークを含む“実行環境”，開発支援ツールなどを含む“開発環境”及びフレームワークに対応した“開発プロセス”を提供することで，各開発フェーズでシステム開発を強力にサポートしている。

*情報技術総合研究所

1. まえがき

開発フレームワークを使用してWebシステムを構築する際に，更なる生産性向上や品質向上を実現するためにフレームワークに代表される“実行環境”だけでなく，それと密接に連携する“開発環境”及び“開発プロセス”も合わせて提供している。これらには，設計・開発ガイドや各種技法書などのドキュメント類も完備しており，はじめてフレームワークを使用するプロジェクトに対して導入しやすくしている。また，事業特性に合わせてベースとなるフレームワークに対して様々な機能強化を実施した“実行環境”を開発している。

本稿では，次の3つの取組みについて述べる。

(1) CoC概念を適用し，肥大化する設定ファイルの記述量を削減したサーバ側の開発フレームワーク

(2) 上流からのフロントローディングによって生産性及び品質向上を実現するために，業界標準の設計言語であるUMLで記述された文書を入力として，Java(注3)ソースコードを自動生成するツール

(3) RIA技術を適用したクライアント側の開発フレームワーク

(注3) Javaは，Sun Microsystem, Inc. の登録商標である。

2. CoC概念を適用したフレームワーク

2.1 Apache Struts設定情報の課題

Apache Struts(以下“Struts”という。)は，Java言語を用いたWebシステムを構築する際に用いるデファクトスタンダードなオープンソースのアプリケーションフレームワークである。**図1**に示すように，Strutsは，ユーザーのボタン押下などによるWebブラウザからのリクエスト(URL)を，Strutsが提供するActionServletが受け付ける。ActionServletは，Struts設定ファイルを参照し，リクエストに対応するActionクラスの呼出し及びリクエストパラメータを格納したFormクラスのインスタンスを作成する。Actionの実行が終わるとActionが指定した論理的なとび先に対応するJSP(Java Server Pages)にフォワードする。

このように設定ファイルにはURLとActionクラスの対応，ActionクラスとFormクラスの対応，JSPへのフォワード情報を保持する。そのため，画面数や画面上に配置されるボタン数が増加すると，それに対応するActionも増え設定ファイルも比例して大きくなる。近年，大規模な業務システムにもStrutsが利用されるようになり，大規模開発におけるStrutsの設定情報が肥大化することで，作成や管理・保守作業が困難になり，問題視されている。

2.2 Struts設定情報削減機能の提供

Struts設定情報削減機能は，設定情報の肥大化の原因となっているAction定義，Form定義，フォワード定義の削減を目的としている。提供する機能は次の4機能からなる。

(1) Action名やForm名の命名規約定義

ユーザーが指定したパッケージ配下に存在するActionクラス及びFormクラスに対して，自動でアクション名やフォーム名を命名する機能である。

(2) Action，FormやJSPの自動マッピング

CoC概念を適用して，Actionに対応するFormやJSPのマッピングを自動で行う機能である。自動マッピングに対応する場所にActionクラス，Formクラス，JSPを配置することによって，設定なしで動作させることが可能になる。

(3) アノテーションによるアクション定義

Action，Formの自動マッピング機能でマッピングしたForm以外のFormを利用したい場合に，Actionクラスのソースファイルに対してアノテーション(注釈)を付加することで，Actionに対応するForm名を指定する。

(4) 物理パスによる呼出し

Actionクラスが次遷移先の論理名を指定する代わりに，JSPなどのURLを直接指定する機能である。

2.3 Struts設定情報削減機能の有効性確認

画面数10，ボタン数15のサンプルアプリケーションを題材にしてStruts設定情報削減機能を用いたものと用いていないものを作成し，コード量を比較した。その結果を**表1**に示す。

Javaコードがアノテーション定義の増加によって4.3%増加したが，設定ファイルは70.5%削減できた。さらに，設定ファイルの詳細は，**表2**のとおりとなった。

Struts設定ファイルであるstruts-config.xmlの記述量が230行から22行と，削減率90.4%に達しており，設定情報削減機能が有効に機能していることが確認できた。

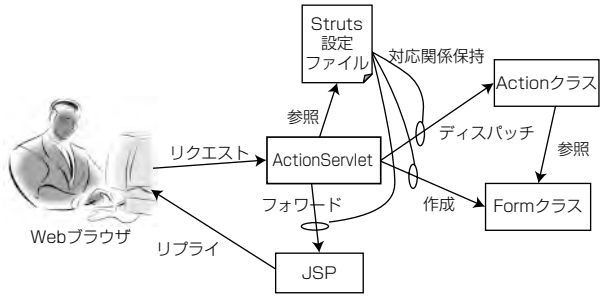


図1．Strutsのアーキテクチャ

表1．アプリケーション作成コード量

種類	非利用(行)	利用(行)	削減率(%)
Java	1,692	1,764	−4.26
JSP	480	418	12.92
設定ファイル	295	87	70.51

表2．設定ファイルコード量比較

ファイル名	非利用(行)	利用(行)
/WEB-INF/web.xml	65	65
/WEB-INF/struts-config.xml	230	22

3. UMLを入力とするソースコード自動生成ツール

3.1 ソースコード自動生成の課題

Webアプリケーションの開発では，オープンソースのアプリケーションフレームワークを利用してアプリケーション構造を，画面を担当するプレゼンテーション層，業務ロジックを担当するビジネスロジック層，データベース接続を担当するデータアクセス層の3層に階層化することが一般的となっている。

当社はこれまで，階層化された各層に対してソースコード自動生成ツールを適用することで，開発効率の向上を図ってきた。プレゼンテーション層・データアクセス層では，定型処理が多いため，設計情報（画面・データベース）から，サンプルアプリケーションの例で内部処理を9割程度生成可能という高い自動生成率を実現していた[(2)]。一方，ビジネスロジック層では，アプリケーションごとに異なる処理が多く，内部処理の自動生成が困難であるという課題があった。

3.2 UML図を入力とするソースコード自動生成機能の提供

3.1節で述べた課題を解決するため，UML図を入力としてソースコード自動生成を行うツールを開発した。UML図とは，ソフトウェア仕様を図式化する統一された手法であり，クラス図やシーケンス図からなっている。自動生成ツールの動作を**図2**に示す。自動生成ツールは，まずクラス図を解析することで，クラスやメソッドの定義情報を読み取る。次にシーケンス図を解析することで，メソッドの内部ロジックを生成する。

3.2.1 クラス図の解析機能

クラス図には，クラスの構成情報が含まれている。自動生成ツールは，クラスの構成情報からクラス名，属性情報（名前・型など），メソッド情報（名前・戻り値の型・引数名・引数の型など）といったクラスのひな型を生成するために必要な情報を取得する。これらの情報からシーケンス図を読み込む準備とソースコードのひな型の生成を行う。

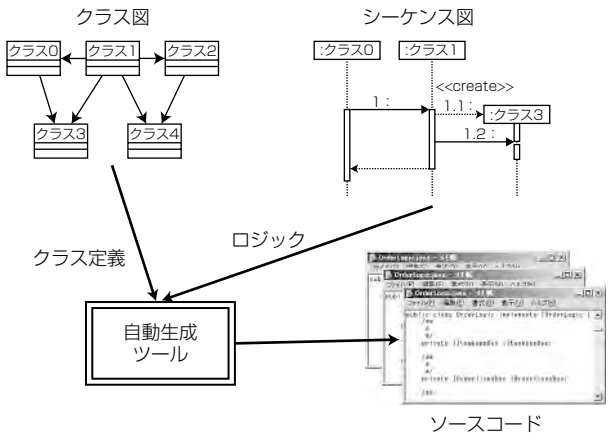


図2．自動生成ツールの動作

3.2.2 シーケンス図の解析機能

シーケンス図には，オブジェクト間のメッセージの流れが時系列順に表現されている。自動生成ツールは，シーケンス図に含まれる全メッセージから，送信元がビジネスロジッククラスであるメッセージを取得し，取得したメッセージをオブジェクト間のメソッド呼出しに変換する。また，通常のメッセージだけでなく，createメッセージやreturnメッセージは，それぞれnew文，return文に変換する。また，シーケンス図は処理の分岐・繰り返しなども表現することができるため，それぞれif文・while文に変換する。このようにして，シーケンス図からメソッド内部ロジックを生成することができる。シーケンス図から生成したロジックを，クラス図から生成したソースコードのひな型と組み合わせることで，設計情報を反映したソースコードを自動生成する。

3.3 ソースコード自動生成機能の有効性確認

ビジネスロジッククラス数3のサンプルアプリケーションを対象として，ビジネスロジック層のソースコードを自動生成し，出力したファイルの行数を比較することで評価した。コード全体に対する自動生成ツールが生成したコードの割合を**表3**に示す。

自動生成ツールが生成したコードは279行であり，全体の93%であった。このように，UML図を入力とすることで，コードのひな型だけでなくメソッド内ロジックも自動生成が可能となり，コードの大部分が自動生成できることが確認できた。これによって，設計情報を実装に反映でき，ソフトウェアの開発生産性と品質の向上への貢献が期待できる。

4 RIAクライアント開発フレームワーク

4.1 WebシステムにおけるWebブラウザの課題

初期のWebシステムでは，データの入力効率が低いマウス操作と，操作ごとに発生する画面の書換えを伴うサーバとの通信による通信データ量の増加が課題となっていたため，クライアントサーバシステム（以下“C／Sシステム”という。）からの移行は限定的なものであった。近年Ajax（Asynchronous JavaScript[注4]＋XML（eXtensible Markup Language））の登場によって，課題となっていた入力効率の改善と画面の書換えを伴わないサーバとの非同期通信による通信データ量の削減が可能になったことで，Webシステムへの移行が進んでいる。

（注4） JavaScriptは，Sun Microsystem, Inc. の登録商標である。

表3．ソースコード自動生成率

	行数（行）	割合（%）
自動生成ツール	279	93
開発者	20	7
合計	299	100

しかし，C／Sシステムは，通信プロトコル上の制約から社内LAN(Local Area Network)での利用が前提であったのに対して，WebシステムではHTTP(Hypertext Transfer Protocol)を用い，社内LANと比べて低速な広域網(インターネット)を利用して構築するケースが大半である。このため，C／Sシステムでは問題とならなかった大量のデータ通信を伴う業務システムでの応答性能の確保が課題となる。

4.2 RIA技術を適用したデータバッファリング機能の提供

RIA技術は，クライアントでの音声や動画といったリッチなコンテンツに注目が集まりがちであるが，業務システムでは処理ロジックを実装できる点が重要である。このことによって，これまでWebブラウザだけでは実現できなかったことが可能になった。

今回，RIA製品の一つであるSilverlight(注5)を利用し，インターネット上のクライアントとサーバとの間で大量のデータ通信を伴う業務システムに対して，応答性能を確保するためのデータバッファリング機能を試作し，その有効性を検証した。

データバッファリング機能とは，非同期通信機能を利用して，ユーザーが入力操作中にバックグラウンドでクライアントがサーバと通信を行い，業務データを先読みしクライアントのメモリ上にデータをバッファリングするものである。

(注５)　Silverlightは，Microsoft Corp. の登録商標である。

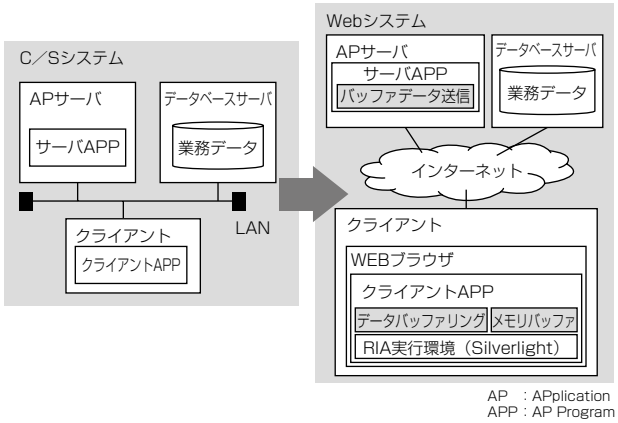
4.3 データバッファリング機能の有効性確認

試作したデータバッファリング機能の有効性を確認するための検証システム(**図３**)を構築し，C／Sシステムと同等の応答性能を達成できることが確認できた。この結果に基づき，実際の業務システムを模したクライアント50台によるサーバ上のデータ2,000件を繰り返し処理するシミュレーションを実施した(**表４**)。シミュレーションの結果，C／Sシステムでは全体の処理時間が313秒(**表４①**)であったのに対し，データバッファリング機能を有効にしたWebシステムでは294秒(**表４③**)となり，データバッファリング機能によって運用形態でもC／Sシステムと同等の応答性能を達成可能なことが確認できた。

今後は，RIAクライアントの開発フレームワーク化に向け，データバッファリング機能など共通機能のライブラリ化，既存のC／Sシステムからの移行ガイドラインなどのドキュメント整備を行う。また，事業側からのニーズに基づいたライブラリや開発支援ツール類の充実化など，フレームワーク整備に向けた取組みを実施する予定である。

5. むすび

当社では，Webシステムが基幹系システムに用いられるようになってからフレームワークの開発に取り組んでおり，これまでに社内外の様々なシステム構築に適用し，生産性向上，品質向上及び保守性の向上を実現してきた。現在，クラウドコンピューティングの台頭によって，情報システムは新しいパラダイムシフトを迎えようとしており，クラウド環境上でのフレームワーク構築にも取り組んでいる(4)。引き続き，最新のシステム構築技術を取り入れた“実行環境”“開発環境”及び“開発プロセス”を三位一体で提供することによって，高い生産性や品質を目指していく。



AP　：APplication
APP：AP Program

図３．RIA技術を適用したデータ検証システムの構成

表４．バッファリング機能による処理時間の比較

	C/Sシステム ①	Webシステム バッファリングなし②	Webシステム バッファリングあり③
データ１件当たりのダウンロード時間	0.3秒	3秒	
データ１件当たりの処理時間	5～10秒		
データバッファ数	0		3
データ：2,000件 端末　：50台 処理時間	313秒	411秒	294秒

参考文献

(1) 原田雅史，ほか： MVCアーキテクチャを実現するアプリケーションフレームワーク“BizFrame”と適用事例，三菱電機技報，**77**，No.7，459～462（2003）
(2) 川口正高，ほか：オープン環境のシステム構築を高品質・短納期で実現するWebシステム開発標準“MIWESTA”，三菱電機技報，**81**，No.7，489～492（2007）
(3) 倉持和彦，ほか：Apache Struts設定削減機能の試作評価，情報処理学会，第71回全国大会，6A-6（2009）
(4) 倉持和彦，ほか：Strutsアプリケーションのパブリッククラウド(Google App Engine)への移行に対する考察，情報処理学会，第72回全国大会，6B-6（2010）
(5) 河村美嗣，ほか：UMLを入力とするソースコード自動生成ツールの開発，情報処理学会，第72回全国大会，6B-3（2010）

特集論文

ソフトウェア要求定義の品質向上技術

石井俊直*
細谷泰夫**
石原　鑑*

On Development and Validation of Software Requirement Specifications

Toshinao Ishii, Yasuo Hosotani, Akira Ishihara

要　旨

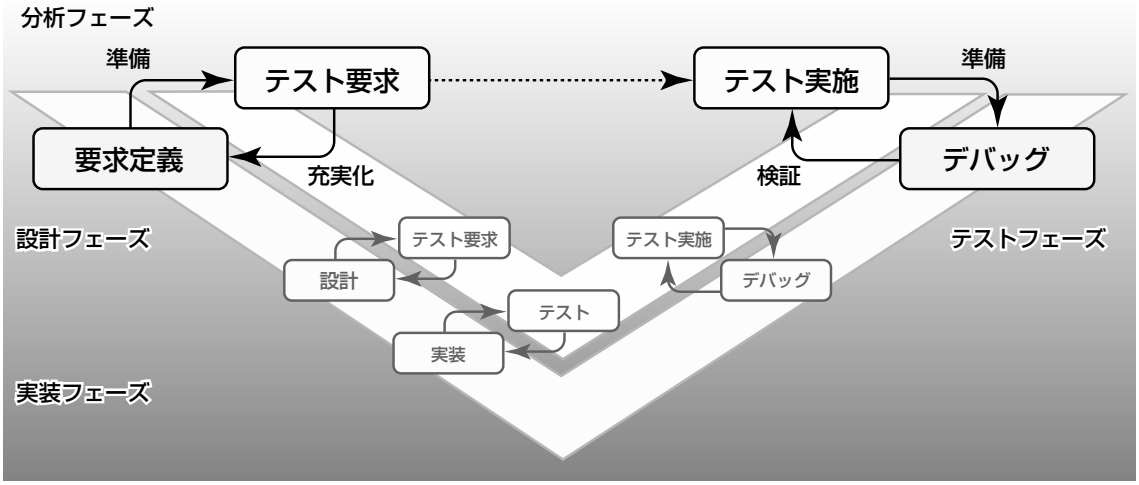
近年，ソフトウェアはあらゆるシステムで使われるようになった。そのためソフトウェアへの要求は多様化し，要求の規模も大きくなっている。ハードウェアの制御を中心とした組み込み分野のソフトウェアでも，この傾向は見られている。このため，ハードウェアが持つ機能の観点からソフトウェアへの要求を仕様化する従来のアプローチだけでは，システムへの要求を十分に把握することが難しい場合が多くなりつつある。本稿では，こうした問題に対応する一つの取組みとして，組み込みシステムの分野でソフトウェアの要求定義にシステムを使って行う業務の観点を積極的に取り入れた要求開発の取組みについて述べる。

この業務観点からソフトウェアの要求定義を導く第一歩は，一般的なオブジェクト指向ソフトウェア設計(OOSE)と同様，業務でアクタ(ユーザー)とシステムの間で起こる振る舞いを表すユースケースの分析である。その後，ユースケースを利用しテスト要求を分析することで，テストの準備を前倒し実施するとともに要求定義にそのフィードバックを行うようにした。上流でテスト要求の分析を行うことで，網羅性向上などソフトウェアの要求定義が充実化される。また要求定義とテスト実施の関連を確かなものとし，デバッグと要求定義の直接的関係を明確にするねらいもある。

この要求開発プロセスは新しい取組みである。実際にプロジェクトに適用すると課題も多く見つかるが，要求開発への有効な作用を含め，次に挙げることを含め多くのことが確認された。

(1)　ユースケース分析結果が多面的に利用価値が高い。
(2)　テスト設計のために用意されたガイドラインがソフトウェア要求開発で有用である。
(3)　直交法[1]やCFD(Cause Flow Diagram)[2]などのテスト設計技法がソフトウェア要求開発でも役に立つ。

本稿では，要求定義を行う分析フェーズにおけるテスト準備とそれによる要求定義充実化の取り組みについて述べる。



テストの並行実施による開発力向上

テスト要求や実施を開発のすべてのフェーズに取り入れるプロセス

実装フェーズをテストで駆動するプロセスが一般的になりつつあるが，ソフトウェア開発のその他のフェーズもテストに関する活動を平行実施することで開発力が向上する。

*先端技術総合研究所　**通信機製作所

1. まえがき

本稿では，社内のある組織を対象に行ったソフトウェア要求仕様作成ガイドラインの改訂と適用の経験をもとに，ユースケースやテスト観点の分析をソフトウェア要求定義フェーズで活用する技術について述べる。

ソフトウェアは，ビジネス課題の計算機システムによるソリューションととらえることができる。すなわち，ある計算機システムがユーザーや他の計算機システムと情報のやりとりをすることでビジネス課題が解決されるが，その枠組みを定義し，また情報のやりとりを実現するのがソフトウェアであると考えるのである。例えば，携帯電話機は最先端の計算機システムだが，その中の通話プログラムは，遠くにいる相手と会話をするという課題を解決している。計算機を実際に動かすプログラムのみならず，システム実現方式などの設計もソフトウェアである。課題解決に適した設計は，具体的な計算機システムより本質的なものであるから，設計をソフトウェアから除外する理由はない。設計とプログラムは，解決のレベルが異なっているだけである。ソフトウェアは一般的にこのようなものなので，その開発では様々なレベルで課題とシステム，そしてその境界を定義することが必要になる。

ソフトウェア開発を分析フェーズ，設計フェーズ，実装フェーズ，テストフェーズに分けると，本稿の対象は主に分析フェーズで，特にソフトウェアの要求定義を話題とする。したがって，課題・境界・システムというソフトウェアモデル化の枠組みで，ソフトウェアの要求定義をどのように進めるべきかが，またより具体的に，要求定義にユースケース(UC)やテストの観点がどのような役割を担うかが本稿全体で議論する課題である。

ソフトウェア要求定義プロセスを定めるには，そもそも課せられている制約とプロセス選択の基準とを分析しておくことが必要である。ここでは，ソフトウェア要求定義フェーズへの主な制約として，次の点を考慮した。

⑴ ソフトウェア要求開発の上流工程では，モデル化の視点がビジネス課題寄りになり計算機システムは高いレベルで抽象化されていること

⑵ このたびソフトウェア要求開発プロセス改革を適用した組織では従来，機能を中心とした開発プロセスが主流であったこと

またプロセス選択の基準としては次の点を考慮した。

①ソフトウェア機能の定義に結びつきやすいこと

②品質の定義が容易になること

③ソフトウェア要求定義の妥当性が確認しやすいこと

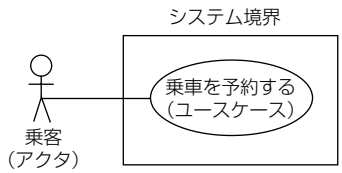
①と②は制約⑵と関係している。③は，ソフトウェア要求仕様レベルの不具合の流出防止やテスト工程の効率化などと関連する重要な基準である。

2. ユースケース ー課題領域から見たソフトウェア要求のモデルー

業務観点からのソフトウェア要求定義の第一歩は，一般的なオブジェクト指向ソフトウェア設計(OOSE)[3]と同様，業務でアクタ(ユーザー)とシステムの間で起こる振る舞いを表すUCの分析を採用した。UCは**図1**に示すようなもので業務が達成されるまでの振る舞いを表したものである。振る舞いと言っても，ソフトウェア要求レベルのUCでは，ユーザーの作業というよりは，起こるべきこと，言い換えるとアクタとシステムの意図を物語風に簡潔に記述したものである。

このUCによって記述した開発対象のモデルには，課題・境界・システムが形式的にはっきりと表れる。アクタが一つのUCの物語で達成することが，すなわち課題に相当する。システムはこの段階では中身がほとんど定義されない一つの箱でしかない。そして境界は，物語におけるアクタとシステムの間でやりとりされる情報という形で明らかにされる。

このようなソフトウェアUCは，ビジネス課題領域からシステムを分析した結果であり，**図1**上に示した図や**図1**下の物語風の文書で記述される。この文書もある形式に則(のっと)って記述されたものであって，自由奔放に記述されたものではない。こうした形式化はソフトウェア開発のあらゆるフェーズで重要であり，ツールによるサポートを可能にするのに必要な要件となるが，本稿ではこの問題にはこれ以上触れない。さて，この段階でシステム内部についてはほとんど定義が行われないが，システムが全体としてビジネス課題にどのように作用すべきかが定義される。そして，**3章**以降で詳しく述べるが，この物語に表れた名詞や動詞がデータや機能としてシステムの内部を形作っていくことになる。



ゴール	乗車を予約する
主アクタ	乗客
	1．乗客は乗車システムにログインする。
	2．システムは，乗車希望条件入力画面を表示する。
	3．乗客は，希望条件を入力する。
	4．システムは，希望条件に合う運行予定を表示する。
	5．乗客は，表示された運行予定から乗車予約希望を選択する。
	6．システムは，乗車予約を行う。
	7．システムは，乗車予約情報を表示する。
拡張シナリオ	4a.乗車希望に見合う運行予定がない場合…

図1．ユースケースの記述例

3. 機能の定義

一般的なオブジェクト指向設計プロセス[3]では，UC分析結果をもとにして，より形式的なモデルへの詳細化が行われる。こうしてできたモデルは分析モデルと呼ばれる。分析モデルを導く代表的な手法にバウンダリ分析がある。しかしながら，我々はこの方法を採用しなかった。代わりに，UC分析結果をもとにソフトウェアが実現すべき機能一覧を作成することとした(**図2**)。ここで機能とは，およそUCの記述に現れた動詞に対応するものである。例えば，UCに“システムはアクタに○○を表示する。”という記述があれば“○○表示機能”が要求機能の一つとなるという具合である。

このようにしたのは，この定義スタイルが，プロセス変革を適用した組織が従来多く採用してきたスタイルに近いことが理由の一つである。機能項目化のあと，“□□機能”がすでに確立されたコンポーネントとして再利用可能なものであれば，その再利用を行うことができる。また“機能”というスタイルに経験豊富なエンジニアはよりスムーズにその経験を生かすこともできる。一方でいったん機能として定義したところで，さらにOOSEの手法を適用していくことが妨げられるものでもない。

また，機能として分解することで**4章**で述べるテスト観点に基づく分析を加えることが容易になる。このテスト観点を使った分析は，振る舞いレベルにとどまったままで要求仕様を充実化することに効果が大きい。

UCから機能を抽出すると，機能は自然にUCに束ねられて項目化される。**4章**で，テストへの要求について述べる中でこの性質を利用する。このほかにも，UCに束ねられることで，個々の機能が利用されるコンテキストが具体的になり，ソフトウェアへの品質要求がより明確になるといったメリットもある。

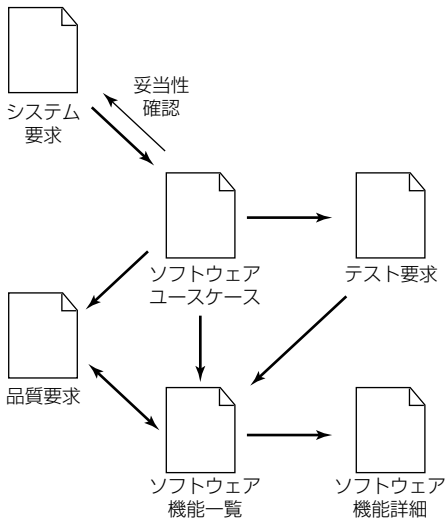


図2．UCを中心としたソフトウェア要求仕様の詳細化

4. テスト要求の分析とフィードバック

ソフトウェア要求定義のフェーズでテストへの要求分析を前倒しに実施する具体的メリットには，次の二つがある。

①プログラムのテスト準備の早期開始

②ソフトウェア要求定義の妥当性向上，要求漏れの低減

開発プロセスのVモデル(注1)に加え，早期のテスト準備などの要素を明記したWモデル(注2)の提案者スピルナーは，Wモデルの本質が“テストと開発の同時平行性”であると述べている[4]。言い換えると，開発のどのフェーズであっても，フェーズによって役割こそ変わるが，テストと開発はいつも同時並行的に行われるのがWモデルに則った開発プロセスである。要求定義の段階でWモデルのプロセスを具体化するため，テスト観点分析を導入した。テスト観点とは，ソフトウェアやシステムを実行する際の環境，結果，実行の部分などのコンテキストから，テスト設計で利用するために特定されたものと我々は定義している。ここで例として，仮想的なモバイル端末のテスト観点分析を行ってみよう。端末には初期化という実行の部分があるだろう。その過程で接続しているネットワークの種類や容量にバリエーションがあると考えられる。また，モバイル端末のバッテリーの状態や，接続されている周辺機器も同様にバリエーションがある。このような分析を行うと，初期化，ネットワークタイプ，ネットワーク速度，接続周辺機器などのテストすべき観点が明確化されてくる。

実際にテスト観点分析を行う場合には，テスト専門家が作成していたテスト設計のためのガイドラインに，テスト観点を特定するためのヒントが散りばめられ有用であった。テスト観点の分析は，UCのシナリオに沿って検討を進める形で進めた。一つ一つの機能はUCシナリオに比べると断片的であるため，コンテキストがイメージしにくい。UCシナリオを活用することで，テスト観点の母集団としてより網羅性の高いものを見いだすことができた。

テスト観点を分析した結果は，マインドマップ(注3)の形でまとめた。さらに，テストすべき処理フローやそのための条件について分析を進めると，要求仕様の妥当性確認テストにつながる。テスト観点分析結果は，このようにテストへの準備としての役割を持つが，さらに分析結果をフィードバックすることで，ソフトウェア要求の充実化がなされる。ソフトウェア要求開発の立場からは，このことの方が重要と言える。はじめにUCをもとに行うソフトウェア要求分析は，ソフトウェアの振る舞いのうち，典型的な業

(注1) V字の底に実装を置き，左側に要求定義と設計を，右側にテストを記して表すモデル。一方向に開発が進むウォーターフォールモデルにおける対応関係を示している。
(注2) 本稿扉ページの図で示したような，Vモデルに平行な二番目のVを持つモデル。
(注3) マインドマップは，Buzan Orqanization Ltd. の登録商標である。

務シナリオが中心とならざるを得ない。これに対し，ソフトウェア要求開発では，それ以外を含めた様々な例外的振る舞いについての要求仕様化も行われるべきである。テスト観点分析は，業務観点の分析で漏れがちな例外的状況への要求を特定するために効果があった。

5. プロジェクトへの適用

最後に，実際の開発プロジェクトへの適用について簡単に触れる。改訂したソフトウェア要求仕様書作成ガイドラインは，現実に行われていることを整理したものというよりは，一つのあるべき姿をもとに作成したものという色彩が強い。そこでまず，規模や対象がこの新しい考え方に合っているプロジェクトを選択し，プロセスエンジニアが積極的にプロジェクトとかかわりながら，全面的に新しいプロセスでプロジェクトのソフトウェア要求開発を進めるという試みを行った。このほかに組織全体への説明会も実施したが，プロセスエンジニアリング側のプロジェクトへの直接的かかわりは，新しい方法の普及に不可欠であるように思われた。

テスト観点分析を実施した結果，UCごとにシナリオ数，テストケース数，処理フロー数を比較することでUC単位で要求仕様書の充実度を推定することができた。テスト観点分析前と後を定量的に比べると，UCシナリオ数と比べおよそ1.7倍のテストケース，シナリオ数で３％程度の不可欠なUCシナリオを新たに見いだすことができた。また，テスト観点分析からおよそ1,000通りのリソース競合関係を特定し，結果的にはこの中のおよそ３％が開発ソフトウェアに特有の競合処理定義が必要であったが，それらすべてをテスト観点分析をもとに定義することができた。

このほかの取組みとして，新しいガイドラインの部分適用も試みた。これは，徐々に浸透させていくという戦略であるが，テスト観点分析は取組み自体が新しいものであるため，この領域がいわば“更地”となっていることが多く，部分導入の差し込み口として適していた。新しいプロセスを組織レベルで導入するには，通常段階的な移行プロセスが必要となる。ところが，今までやってきたことを変えることはなかなか難しいもので，移行が進まないことはよく見られる。今回の取組みでは，更地の開拓の方がすでにいろいろ積み上げのあるところを変えることよりもスムーズに進む傾向が見られた。このことから，有用な新しい技術をてこにした移行は，一つのパターンになるだろう。

6. むすび

UCによる分析を中心としたソフトウェア要求仕様開発の新しい取組みについて述べた。この取組みで採用した開発プロセスは，ソフトウェアへの要求を洩(も)れなく見つけることに重点を置き，そのため様々なテクニックを使いソフトウェアの振る舞いについて追求するものである。

要求を的確に獲得することは，顧客に満足いただけるシステムを提供するための基本である。ここで述べたプロセスを普及・発展し，顧客により高いレベルの満足を提供できるよう，今後も活動を進めていきたいと考えている。

参考文献

⑴ 立林和夫：入門タグチメソッド，日科技連出版社（2004）
⑵ 松本正雄，ほか：ソフトウェア開発検証技法，電子情報通信学会（1997）
⑶ イヴァー・ヤコブソン，ほか：UMLによる統一ソフトウェア開発プロセス―オブジェクト指向開発方法論（Object oriented selection），翔泳社（2000）
⑷ Spillner,. A，et al.：Software Testing Practice：Test Management，Rocky Nock Inc.（2007）

特集論文

システム構築上流設計手法の改善

篠崎　衛*
相馬仁志*
武曽　徹*

Improvement of Business Architecture Design Approach

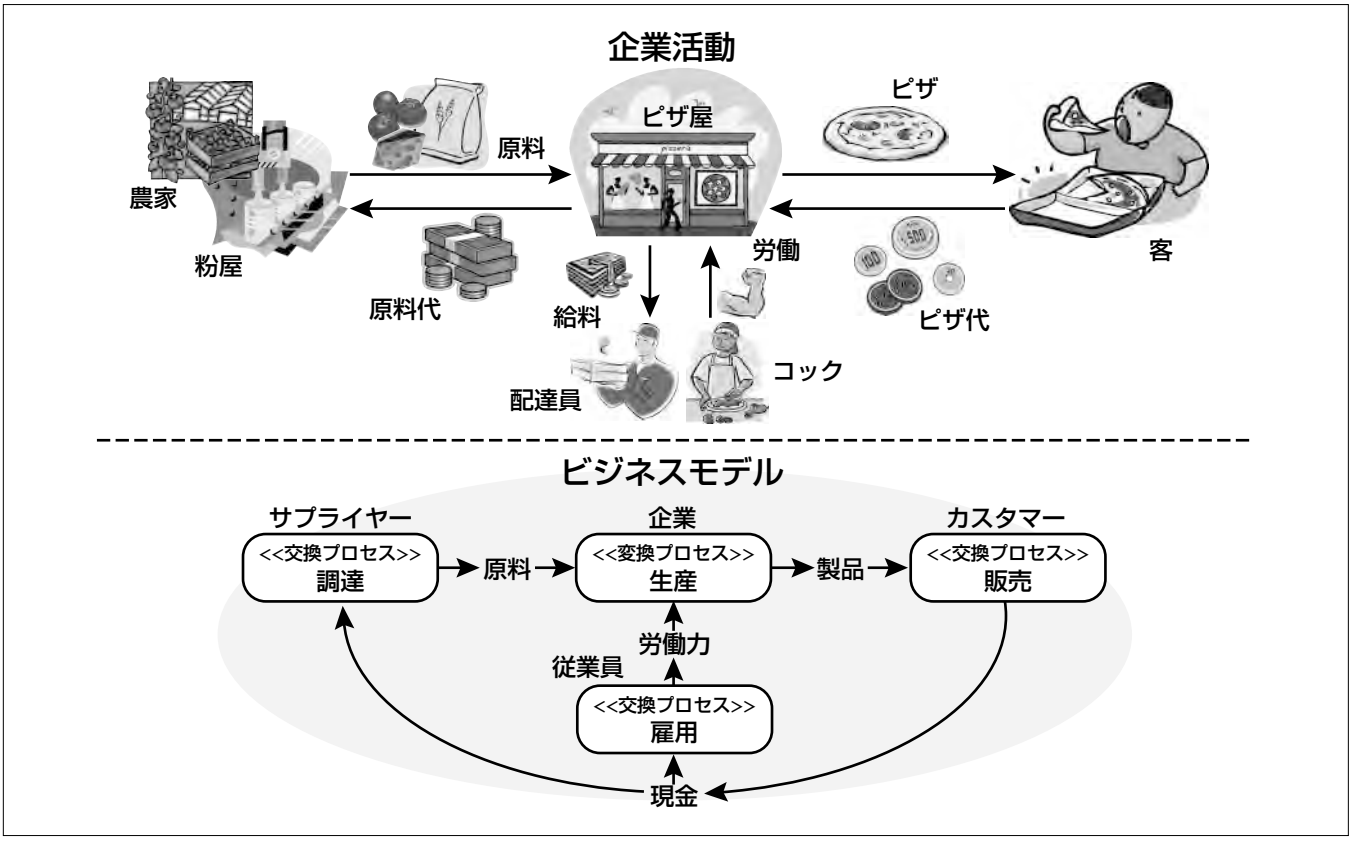
Mamoru Shinozaki, Hitoshi Soma, Toru Muso

要　旨

競争の激化によって市場の競争原理に変化が生じており，これからは知恵の競争へ突入するといわれている。すなわち，すべての従業員が，環境の変化や課題を感知し，それぞれの立場で考え，軌道修正を加えながら適応していく必要がある。そのためには，ビジネスの統一的な理解が必要で，その仕組みとしてビジネスモデリングに注目が集まっている。システム構築事業でも，顧客のビジネスの理解と課題の共有，解決策の共創で，ビジネスモデリング技術の向上は重要な取組みの一つであり，三菱電機でも日々改善を加えている。

本稿では，検証方法の一例として，当社で取り組んでいるREA（Resource Event Agent）モデルを活用したモデルの品質向上の手段と適用事例について述べる。

REAモデルは，Pavel Hrubyによる，ビジネスモデリングに特化したモデル記法であり，企業活動領域で使われている意味ごとに構造やルール（構造パターン）を提供している。REAでは，これら構造パターンの制約に従ってモデリングすることによって，大きな抜けや誤り，解釈の相違を防ぐことができるという。当社では，“技術者のレベルによらず，一定の品質を確保すること”を可能とするため，REAモデルの基本的な制約をもとに，ビジネス要求定義時に作成するモデルの検証を試みている。その結果，機能不足や外部とのインタフェースの存在を推定できることなどの有効性を確認した。更なる展望としては，モデル検証方法の汎用（はんよう）化と充実化，事例の蓄積による検証方法の拡充，整合性検証支援環境，知識ベースへの展開などが考えられる。



ピザ屋の商売（企業活動）とバリューチェーン（ビジネスモデル）

ピザ屋がピザを売って収入を得るという商売にかかわる業務の流れ（上の絵）を，REAモデル記法に従って抽象的に表現したバリューチェーン（下の絵）を示す。バリューチェーンは，企業が扱うリソースが，だれと交換され，どのように変換されるかに注目して表現したものである。

*情報技術総合研究所

1. まえがき

上流設計手法におけるビジネスモデリングの課題と，解決手段としてのREA構造パターンの活用について，実際のプロジェクトに適用した事例を含めて述べる。

2. 背景と課題

当社で実施している上流設計手法の概要と，ビジネスモデルを知識ベースとして活用する上での課題について述べる。

2.1 ビジネスモデルの活用状況

当社では，顧客ビジネスの理解と課題の共有，解決策としてのITソリューションの共創を目指し，システム構築手法を整備し，日々改善を加えている。システム構築手法は，EA(Enterprise Architecture)の手法をベースとし，経営戦略から運用保守までをサポートしている。また，上流設計手法によるBA(Business Architecture)は，ソリューションとしても提供している。また，社内での業務改善プロジェクトにも活用することで，ビジネスアーキテクトの育成に努めている。

2.2 モデリングにおける課題

システム構築手法の中で使用するモデリング技法は，特に限定せずに目的に応じて選択し使用する(Unified Modeling Language：UML(注1)，Business Process Modeling Notation：BPMN，Supply-Chain Operations Reference-model：SCOR(注2)，Event-driven Process Chain：EPCなど)。しかしこれらのモデル記法は，汎用性が高いことが災いして，でき上がったモデルに対する解釈が，読む人間の経験や技術レベルによって異なってしまうことが多い。これらの現象の原因として，モデリング技術者の業務ドメイン知識不足や，モデリング記法自体に，業務ドメイン知識やルールを明確に表現する手段が提供されていないことなどが知られている。

当社でも，ビジネスモデルを蓄積し，知識ベースとして活用することで生産性向上を試みているが，これらの課題が普及の障害となっている。すなわち，“経験豊富な技術者が作成したモデルを読んで，擬似的に経験を積む”“ほかの人が作成した既存モデルを再利用することによって，モデリング時間を削減する”などの事例が増えてこない。

(注1) UMLは，Object Management Group Inc. の登録商標である。
(注2) SCORは，Supply Chain Council Inc. の登録商標である。

3. 解決策

3.1 REAモデルの活用

REAモデルとは，ビジネスモデルを記述するために用いられ，企業のビジネス(活動の仕組みや組織)の構造を記述するために表記やルールを定めたものである。REAモデルが他のモデル記法と最も異なる点は，オントロジ(基本概念や語彙(ごい)の体系)とメタモデル(モデル構造の意味をモデルによって表現したもの)を導入したことである。オントロジによって，ビジネス(経済活動)の観点からモデルの意味を説明できるとともに，メタモデルによって見落としなく記述でき，かつ一貫性をある程度保証することができるとしている。具体的には，企業の経済活動を“保有している資産(リソース)の価値を増減させること”と定義し，企業活動にかかわる人や組織・機械(エージェント)，発生する変化や状況(イベント)との関係を，企業活動領域で使われている意味ごとに，構造やルール(構造パターン)を提供している。REAでは，これらのパターンの制約に従ってモデリングすることによって，大きな抜けや誤り，解釈の相違を防ぐことができる。

当社では，ビジネスモデルの品質向上を目的に，REAモデルを活用したモデルの整合性検証を試みている。

3.2 モデルの整合性検証手段

ここで，検証方法の一例として，ビジネスプロセスモデルの検証方法と事例について述べる。

(1) REA構造パターンへのマッピング

はじめに，検証対象のビジネスプロセスモデルを，REA構造パターンに沿って表現する。構造パターンとは，実際に起きる経済的な活動を表現するための構造を定義したモデル(メタモデル)である。ビジネスモデルは，次の基本パターンの組合せのみで表現できる。

①REA交換プロセスパターン
②REA変換プロセスパターン

図1に交換プロセスパターンのメタモデルを示す。

次に，検証で着目するポリシー及び制約を示す。

①企業は“交換”と“変換”によってリソースの価値を増やしたり減らしたりする。
②交換(exchange)は，企業が他のエージェントからリソースを受け取り，その見返りとして，そのエージェントにリソースを与えることである(取引)。

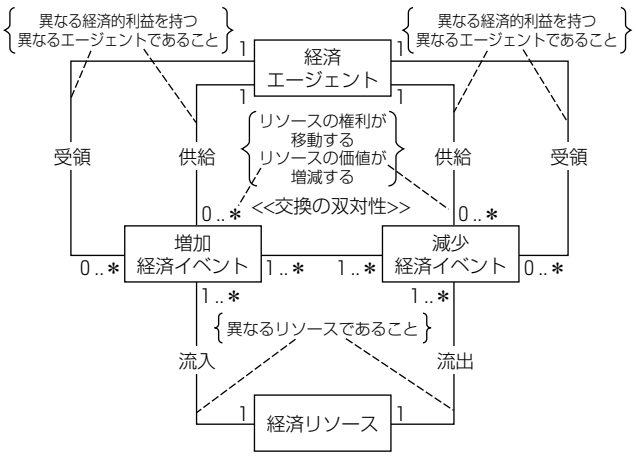


図1．REA交換プロセスパターン(メタモデル)(3)

③変換(conversion)は，企業が新しいリソースの生産や，既存のリソースの変更のために，リソースを使用又は消費するプロセスソースに対する権利の受渡しを行うことである(生産)。

④企業が，その制御下にあるリソース全体の価値を増やしたい場合，通常はいくつかのリソースの価値を下げなければならない。

⑵ REAバリューチェーンへのマッピング

次に，REA構造パターンで表現したプロセスをつなげて，バリューチェーンを構成できるか否かを検証する。バリューチェーンとは，企業が計画，監視，制御したいリソースの価値に，影響を与えるプロセスのつながりを矢印(リソースバリューフロー)で表現したものである。**図2**にバリューチェーンのモデルを示す。バリューチェーンを作成する上で，着目するポリシー及び制約を示す。

①ビジネスプロセスは，そのプロセスが制御するリソースに価値を与える。

②ビジネスプロセスは，他のプロセスによって生成されるリソースを活用する。

③フローは，いずれかのプロセスの中で始まって，いずれかのプロセスの中で終わらなければならない。

④ビジネスプロセスは，流入するリソースフローと流出するリソースフローの両方を持たねばならない。

4. 適用と評価

ここでは，検証手段を実際の業務システム要求定義に適用した結果について述べる。なお，モデルは単純化してある。

4.1 適用プロジェクトの概要

評価対象としたのは，情報提供サービス事業のシステム化プロジェクトである。顧客の要求に応じて，顧客に設置済みの機器から必要なデータを収集し，作成したレポートを提供するサービスを実施している。顧客数の増加に伴い，データ収集とレポート作成を自動化することになった。さらに，Web上での閲覧環境も提供するシステムを構築する。

このプロジェクトに対して，あるビジネスアーキテクトが作成したビジネスプロセスモデルを**図3**に示す。

次に，このモデルを検証するために実施した手順を示す。

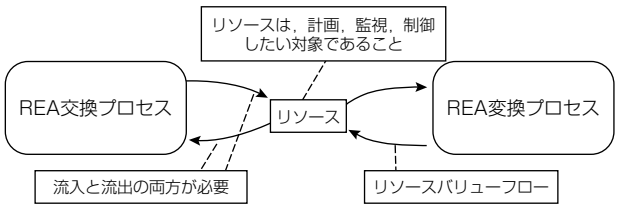


図2．バリューチェーン(メタモデル)[3]

4.2 REA構造パターンへのマッピング

対象とするビジネスプロセスをREAを使って表現すると，**図4**に示すモデルになる。既存のモデルとの対応は次のようになっている。

①“収集”と“提供”は交換(取引)プロセスであり，“加工”は変換(生産)プロセスである。

②REAでは，収集，加工，提供プロセスには，それぞれ，保管，供給，決済のイベントが記述される。

REAモデルへのマッピングによって，一般的に，他業務でも発生し得ると考えられる次の傾向が読み取れた。

①リソースの供給者と受領者が同一の場合には，必要なイベント(すなわち機能)が省略される。

②記憶域，データなど目に見えないリソースが使用されたり，消費されない(ように見える)リソースを使用したりする場合に，必要なリソースが省略される。

4.3 REAバリューチェーンへのマッピング

次に，バリューチェーンを作成すると，**図5**に示すモデルとなる。バリューチェーン内では，リソース“現金”と

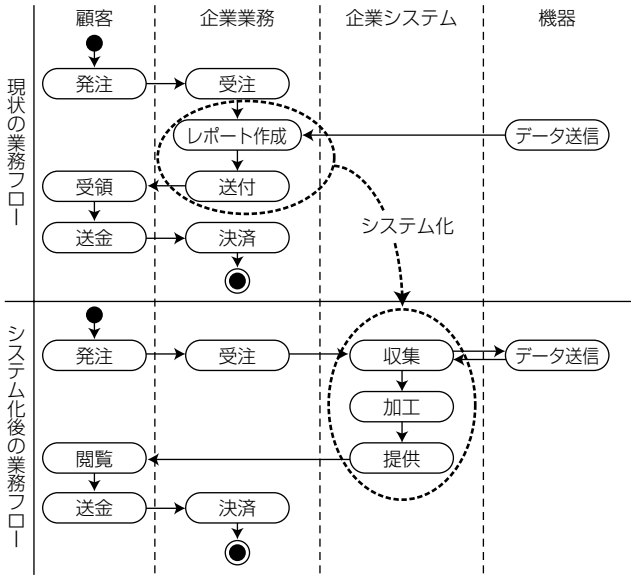


図3．評価対象のビジネスプロセスモデル

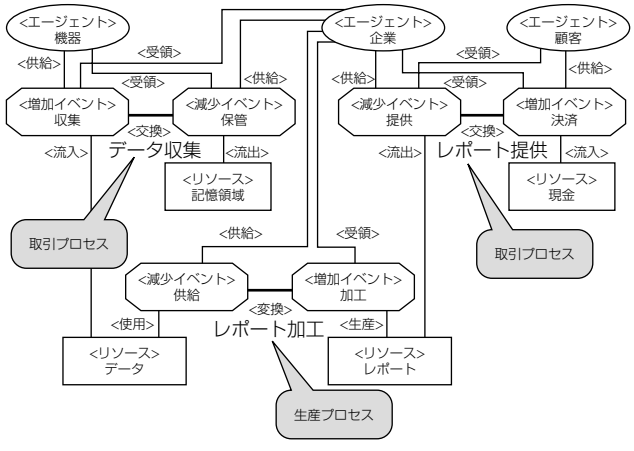


図4．REA構造パターンにマッピングした後のモデル

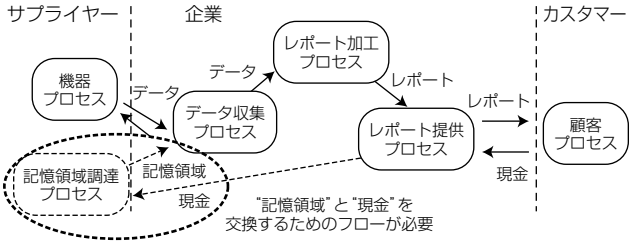


図5．REA構造パターンにマッピングした後のバリューチェーン

“記憶領域”に対するフローが完結していない（点線で囲んだ部分）。すなわち，現在検討している業務範囲外に，現金から記憶領域へ交換しているプロセスが存在するはずである。したがって，現在検討中のシステムには，“現金及び記憶領域に対するデータを外部と交換するためのインタフェースが必要になることを考慮しておくべき”となる。

4.4　評価と考察

検証によって，次の知見が得られた。

⑴　企業活動の意味の理解が深まる

プロセスごとに目的（生産／交換），利害関係（何をだれと）など，手段ではない本質が明確になる。

⑵　気付きにくいリソースもあぶりだされる

見えないリソース（労働力，計算能力，保管空間など）が，REAのルールを守ることによって，モデル上に自然と表現されるようになる。

また**図6**に，今回事例として取り上げたモデルを基に，より一般化したプロセスを示す。**図6**から，サービス業でも，企業活動に必要なリソース（労働力）を調達し，付加価値をつけ（能力や知識），顧客に提供するという製造業と類似した活動を行っていることが分かる。したがって，ある業務のある抽象化レベルでは，プロセスモデルを標準化することが可能であることが分かる。したがって，業種共通のプロセスモデルを蓄積し，モデリング時にテンプレートとして活用することによって，一定の品質を確保できる仕組みを提供できるようになる。

5. むすび

ビジネスモデリングにおける課題“モデルに対する解釈が，読む人間の経験や技術レベルによって異なる”の解決策の一つとして，REAモデルの制約を基にビジネスプロセスモデルを検証する手段と事例を示した。事例によって，陥りやすい誤り（考慮不足の点やモデリング対象範囲外とのインタフェースなど）を検出可能なことを示し，検証手段の有効性を確認した。さらに，この手段によって，モデリング上の共通項をテンプレート化する一例を示し，知識ベースとしての活用可能性を示した。

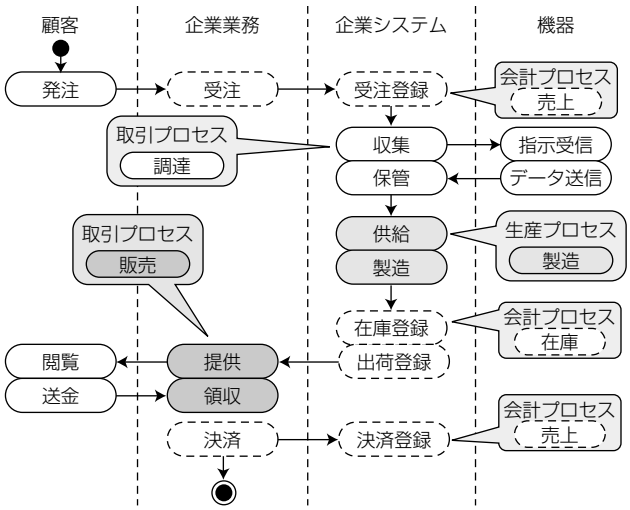


図6．ビジネスプロセスモデルのテンプレート

更なる展望としては，モデル検証方法の汎用化と充実化，事例の蓄積による検証方法の拡充，整合性検証支援環境，知識ベースへの展開などが考えられる。

参考文献

⑴　松岡泰正，ほか：お客様との共創を目指したエンタープライズアーキテクチャ（EA）に基づくソリューションサービス，三菱電機技報，**79**，No.4，241～246（2005）

⑵　相馬仁志，ほか：業務分析による業務・システム改善事例とその評価，情報処理学会第71回全国大会（2009）

⑶　Hruby，P.，et al.：ビジネスパターンによるモデル駆動設計，日経BPソフトプレス（2007）

⑷　依田智夫：REAとビジネスパターン入門，ITpro　日経BP（2007）
http：//itpro.nikkeibp.co.jp/article/Watcher/20070828/280551/?ST=system

特集論文

定量的プロジェクト管理支援システム“P-Support”の開発と試行 —定量データに基づくプロジェクト状況の把握への取組み—

坂田賢志*
藤原良一*
岩切　博**

Development and Trial of Quantitative Project Managemet Support System

Takashi Sakata, Ryoichi Fujihara, Hiroshi Iwakiri

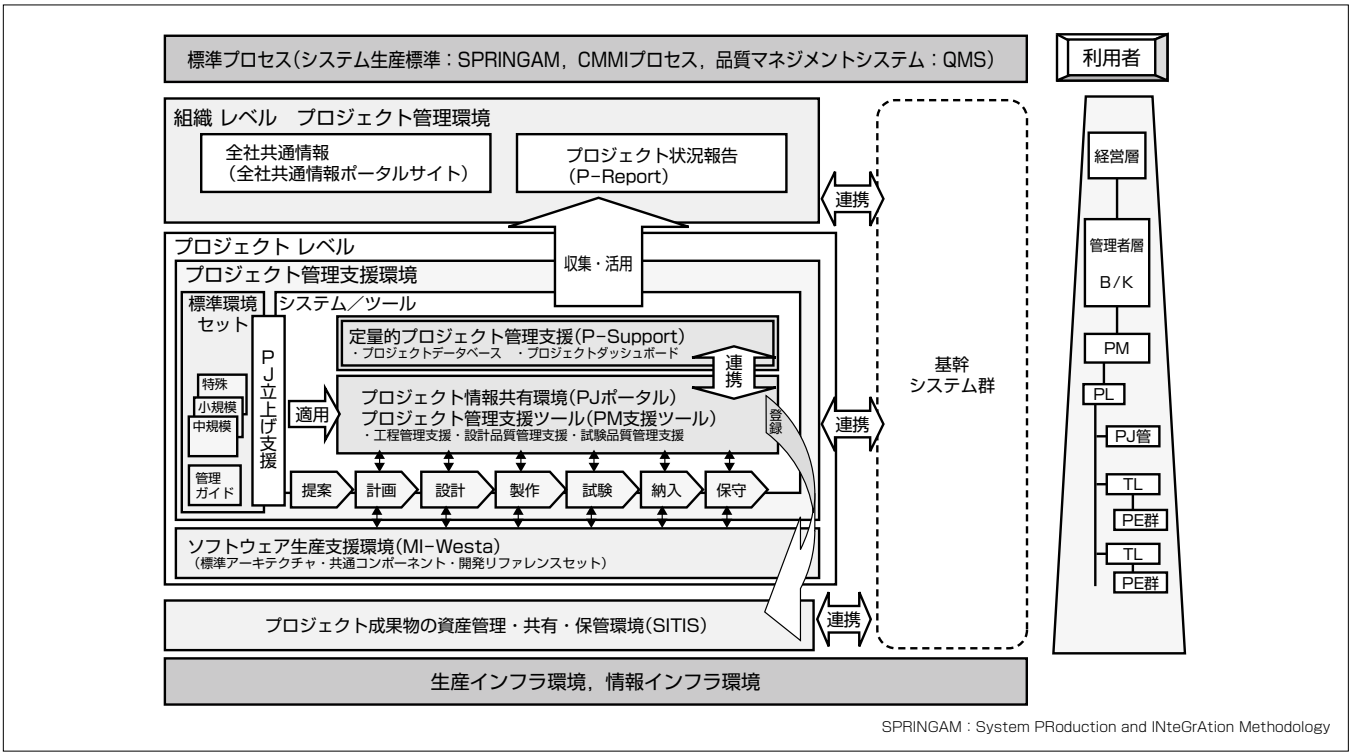
要　旨

システム生産活動の品質・生産性の改善を目的としたプロジェクト管理情報システム(PMIS)のプロジェクト管理支援環境として，ISO9001対応の品質マネジメントシステム(QMS)を基に，CMMI(Capability Maturity Model Integration)(注1)レベル3相当の標準プロセス，情報共有環境(PJポータル)やプロジェクト管理支援ツール(PM支援ツール)を強化してきた。

さらに，組織とプロジェクトの品質・生産性の継続的な改善をねらい，プロジェクトの状況をよりタイムリーかつ客観的に把握し，対策をフィードバックできるCMMIレベル4相当の定量的プロジェクト管理の標準プロセスを整備した。従来のプロジェクト管理方法では，プロジェクト状況を組織の管理者層やプロジェクトマネージャー(PM)が把握するために，人手でデータを収集しPM支援ツールを使って分析・評価報告を行っていた。よりタイムリーにプロジェクト状況を把握するために，プロジェクトのデータ収集や分析の負荷を軽減し，早期に問題点の検出が行えるシステム“P-Support”の開発が不可欠であった。

P-Supportは，PJポータルのプロジェクトの計画・実績データを自動的に取り込み，プロジェクトの品質／コスト／工程／リスクの評価を行う。また，評価の変動(トレンド)とその詳細を一覧表とグラフで表示する。これによって，プロジェクトメンバーの管理負荷をかけずに定量データを蓄積し，管理者やPMがプロジェクト状況の変化をタイムリーにとらえることができる。P-Supportの試行を通じ，プロジェクト管理作業の共通化・均質化によって組織レベルでのプロセス改善とプロジェクト効率化の適用効果を確認した。今後，全社展開を進めるが，より効果的な機能改良や蓄積データを見積り・計画業務の精度向上に利用したいとの要望に対応することで，組織やプロジェクトの品質・生産性の継続的な改善活動の普及・定着を図っていく。

(注1)　米カーネギーメロン大学ソフトウェア工学研究所が公表した，組織の能力成熟度レベルを5段階で評価・改善する能力成熟度モデル。システム開発を行う組織がプロセス改善を行うためのガイドライン。



SPRINGAM：System PRoduction and INteGrAtion Methodology

MDISプロジェクト管理情報システム(PMIS)の構成

PMISとは，システム生産活動を円滑に進めるためのプロジェクト管理／ソフトウェア開発支援環境の総称で，プロジェクト管理／ソフトウェア生産支援環境のほかにも開発プロセスなどの会社規則や，生産・情報インフラ環境が含まれる。

*三菱電機インフォメーションシステムズ㈱　**三菱電機㈱ 本社

1. まえがき

システム生産活動の品質・生産性の改善を目的に，生産活動の継続的な改善をねらい，プロジェクト管理情報システム(PMIS)を整備してきた。

PMISは，次の5つの機能群から構成されている。

(1) 標準プロセス

システム生産標準：SPRINGAM，CMMI・ISO9001準拠の品質マネジメントシステム：QMS

(2) 組織レベルのプロジェクト管理環境

①プロジェクト状況報告システム(P-Report)

②全社共通情報ポータルサイト

(3) プロジェクトレベルの支援環境

①プロジェクト管理支援環境

(a) プロジェクト情報共有環境(PJポータル)

(b) プロジェクト管理支援ツール(PM支援ツール)

②ソフトウェア生産支援環境(MI-Westa)

(4) プロジェクト成果物管理・共有・保管環境(SITIS)

(5) 生産インフラ環境，情報インフラ環境

イントラネット，電子メール，ネットワーク環境など

従来からCMMIレベル3相当のデータに基づくプロジェクト管理を実施してきたが，プロジェクトの大規模・短期化・複雑化に伴い，プロジェクトの状況をよりタイムリーかつ客観的に把握し，迅速に対策をフィードバックできるCMMIレベル4相当の定量的プロジェクト管理を導入した。しかし，従来の管理方法では，データ収集及び分析の負荷が大きくなることが予想された。そこで，定量的プロジェクト管理の適用を推進するため，管理負荷を低減しつつ定量的プロジェクト管理が行える定量的プロジェクト管理支援システム“P-Support”の開発を行った。

2. システムのねらい

P-Supportの開発を行うにあたり，利用部門の要求を整理し，次のシステム化方針を設定した。

(1) 現状のプロジェクト管理手法を踏襲する。

(2) 実施メンバーの管理負荷を低減する。

(3) 職制で担当しているプロジェクトの品質(Q)／コスト(C)／工程(D)／リスク(R)の状況を俯瞰(ふかん)してタイムリーに把握できる。

これらを実現するために，プロジェクトレベルの定量的プロジェクト管理，職制レベルのプロジェクト状況の見える化，データの自動収集の整備方針を設定した(**図1**)。

次に，整備方針に基づくシステムのねらいについて述べる。

2.1 プロジェクトレベルの定量的プロジェクト管理

プロジェクトの状況変化をタイムリーに把握し，問題点の検出を早期に行うためのねらいを次の3点とした。

(1) プロジェクトの計画と実績データに基づきQCDRごとにグラフなどを活用して，定量的かつビジュアルに状況の把握ができる。

(2) 従来より活用しているPM支援ツールにデータ連係することで，より詳細な分析ができる。

(3) 蓄積データを基に，見積・計画業務の精度向上を支援できる。

2.2 職制レベルのプロジェクト状況の見える化

個別プロジェクトの状況を職制で横通しで見ることで，組織的なリソースの投入などの対策を早期に行えるように，次の2点をねらいとした。

(1) 職制の全プロジェクト状況を総括的に把握できる。

(2) 計画と実績の差異及び実績の変化による問題プロジェクトの早期発見，及び改善対策効果の確認ができる。

2.3 データの自動収集

プロジェクトメンバーが生産活動に専念できるように，生産活動で発生する作業成果物からデータを抽出する。具体的には，次の2点とした。

(1) プロジェクトの生産環境に蓄積された作業成果物から定量データを自動収集できる。

(2) 定量データは，QCDRに関する定量データを対象とする。

3. システムの特徴

先に述べた整備方針を基に，プロジェクトの定量的プロジェクト管理に対しては，プロジェクトダッシュボード(詳細表示)機能，職制でのプロジェクト状況の見える化に対しては，プロジェクトダッシュボード(一覧表示)機能，データの自動収集としてデータの自動収集機能とそのデータを蓄積するプロジェクトデータベースを整備した(**図2**)。

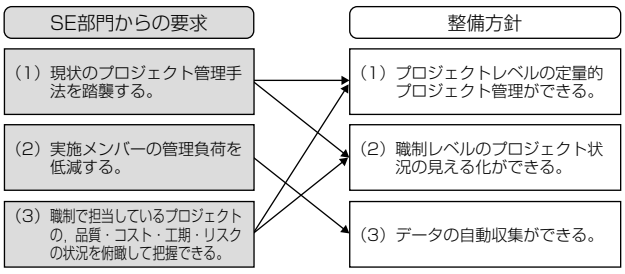


図1．SE部門からの要求と整備方針

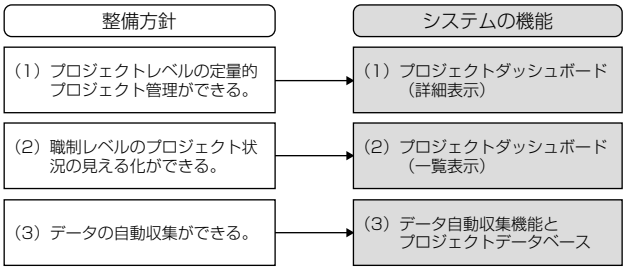


図2．整備方針とシステムの機能

これらのシステムの機能概要を**図3**に示す。

3.1 プロジェクトダッシュボード(図3(1)(2))

品質／コスト／工程／リスクを対象に定量的にプロジェクトを評価し，その状況を表示する機能で，一覧表示と詳細表示の2種類の画面機能からなる。

3.1.1 プロジェクトダッシュボード(一覧表示)

プロジェクトダッシュボード(一覧表示)画面を**図4**に示す。P-Supportは，計画データとPJポータルから自動的に取り込んだ実績データから，評価基準を基にQCDRごとにA～Eのランク評価を実施する。一覧表示画面では，部門内のプロジェクトのQCDR評価トレンド(前回評価と最新評価とその変化)を視覚的にとらえやすいようにデザインしている。表示したいプロジェクトの条件や表示順序，及び表示する評価項目はユーザーごとに設定できるので，ユーザーの注目する評価視点でプロジェクト状況を総括的に見ることができる。また，登録プロジェクトとユーザーIDの両方に参照／更新／削除の権限管理機能(事業部／部／課レベル)を実装しているので，部門ごとにプロジェクトデータのセキュリティが確保されている。

プロジェクトのA～Eのランク評価は，次の基準値で判定している。

①品質(Q)　：設計ステップ(レビュー指摘率)
　　試験ステップ(誤り検出率)
　　の計画(上下限管理限界)値からの逸脱率
②コスト(C)：EVM(Earned Value Management)工数差異予測値
③工程(D)　：EVM日数差異予測値
④リスク(R)：(a)件数とコンティンジェンシー額
　　又は(b)処置期限最大遅れ日数

このように評価の共通化，基準値の明確化を行うことで，プロジェクト評価の効率化・均質化を図っている。なお，これらの基準値は，部門ごとに設定することも可能である。

3.1.2 プロジェクトダッシュボード(詳細表示)

図5のプロジェクトダッシュボード(詳細表示)画面は，一覧表示画面上で選択したプロジェクトの詳細状況をグラフ表示する。この画面は，最大4つのグラフパーツを配置できる画面を，ユーザーIDごとに3画面ずつ設定することができる。今回のシステム化では，品質／コスト／工程／リスク及び評価推移を見る次のグラフパーツを整備した。

(1) 評価トレンドグラフ(QCDR評価の推移を表示)
(2) EVMグラフ(PV(Planed Value)／EV(Earned Value)／AC(Actual Cost)グラフ，SV(Schedule Variance)／CV(Cost Variance)グラフ，工数予測グラフ，日数差異予測グラフ)
(3) 品質グラフ(開発ステップ別誤り検出率を表示)
(4) 原価実績グラフ(費目別計画・実績推移積上グラフ)
(5) リスク推移グラフ(件数とコンティンジェンシー額の推移)

各画面ページのレイアウトや配置したグラフパーツ情報も，ユーザーIDごとに設定することができる。

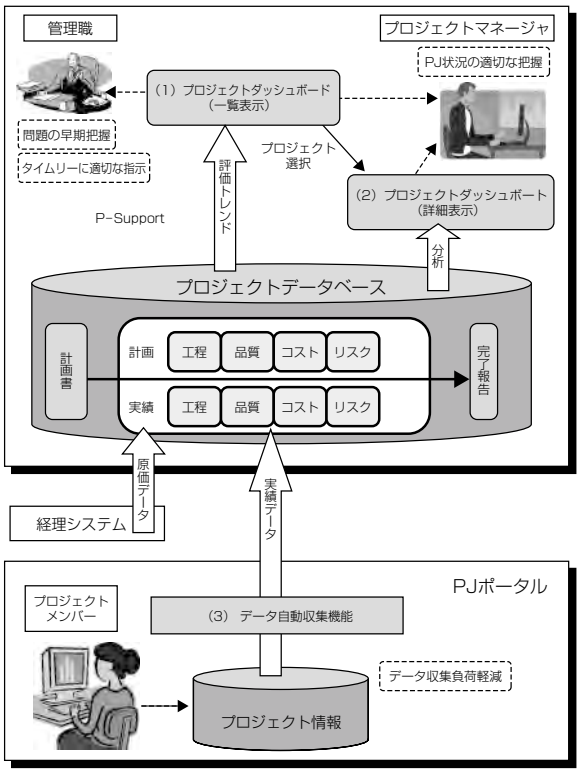


図3．システム機能概要

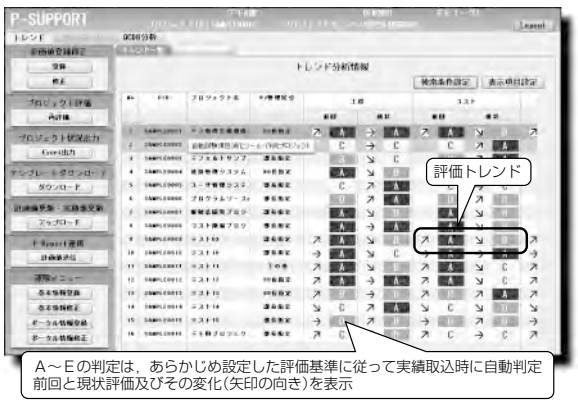


図4．プロジェクトダッシュボード(一覧表示)

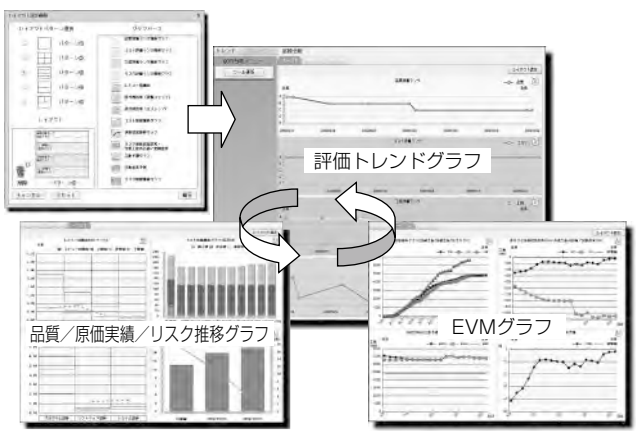


図5．プロジェクトダッシュボード(詳細表示)

3.2 データ自動収集機能(図3(3))

PJポータルで共有している作業成果物から，品質／コスト／工程／リスクに関するデータを抽出し，定期的にプロジェクトのデータを管理するデータベース(プロジェクトデータベース）に自動アップロードすることができる。

3.3 その他の支援機能

(1) 詳細品質分析支援機能

プロジェクトデータベースの品質データ(計画・実績データ)をPM支援ツールと連携させ，より詳しい品質分析を可能とする。

(2) プロジェクト完了報告書作成支援機能

プロジェクトデータベースのデータから，計画データと実績データを完了報告書に自動転記し，報告書作成の効率化を図る。

(3) データアップロード機能

PJポータルを利用していない遠隔プロジェクトなどを持つ組織でもP-Supportを活用できるように，プロジェクトの計画書や作業成果物からプロジェクトデータベースにデータをアップロードできる。

4. P-Supportの利用形態

P-Supportは，プロジェクトメンバーに負荷をかけずにプロジェクト状況を把握することをねらい，プロジェクト情報共有環境(PJポータル)と連係して作業成果物から実績データを自動収集し，計画データと比較してプロジェクトの評価が行えることを目指している。

具体的なP-Supportのプロジェクトでの活用シナリオについて次に述べる。

(1) プロジェクト計画の設定

プロジェクトの計画フェーズでは，プロジェクト特性とプロジェクトのQCDR計画データを登録する。

(2) プロジェクトの実行状況の把握

①プロジェクトメンバーは，プロジェクトの作業成果物をPJポータルで共有する。PJポータルの情報と経理システム内のプロジェクトのコスト実績は，日次で自動的にプロジェクトデータベースに取り込まれる。

②管理者及びPMは，ダッシュボードの一覧表示機能使って，部門のプロジェクトのQCDR評価トレンドを総括的に把握する。プロジェクトの詳細状況を把握したい場合は，詳細表示機能を使ってプロジェクトごとの状況を把握する。

③より詳細な分析を行いたい場合は，PM支援ツールにデータ連携して，プロジェクトの品質分析を行う。

(3) プロジェクト全体の評価

プロジェクトの完了フェーズでは，プロジェクトデータベースの計画／実績データを基にプロジェクトプロセスを分析し，教訓や改善点を含んだプロジェクト完了報告を作成する。他のプロジェクトへの教訓の水平展開を目的に，プロジェクト成果物の資産管理環境に蓄積する。

5. P-Supportの試行と今後の課題

P-Supportの全社展開を目的に，2009年11月から実プロジェクトでの試行を開始した。

試行部門やプロジェクトは，工程短縮や出荷後品質予測など，品質・生産性向上を目的に，試行を推進している。

その結果，次の効果が得られている。

(1) P-Supportによって，プロジェクト管理作業が共通化・均質化でき，生産活動の効率化（生産性向上）が図れる。PJポータルとの連係によって定量データの収集が容易になることで，データの活用につながり，今後組織データとしてのフィードバックも可能となる。

(2) プロジェクトダッシュボードによる状況の見える化で，QCDRのポイントを素早く抑えたフォローや対策が可能となる。

これによって，システム化のねらいである，現状のプロジェクト管理手法の踏襲，プロジェクトメンバーの負荷軽減，タイムリーなプロジェクトの状況把握と問題プロジェクトの早期発見に関して達成できる見通しがついた。

一方，試行作業を通じて，システムに対する課題も認識できている。システムの目的・ねらいと照らして整理する。

(1) “データの自動収集”に関しては，より管理負荷を減らすユーザーインタフェースにする。特に品質管理に関するインタフェースファイルの種類が多く，シンプルなインタフェースにする改善が必要である。また，プロジェクトで独自に活用している作業成果物からデータを取り込む改良を行うことで，標準様式への変換の手間を省く。

(2) “職制レベルのプロジェクト状況の見える化”に関しては，QCDR評価だけでなく，想定される課題の洗い出しとその対応策がリコメンドされる機能が必要である。

(3) “プロジェクトレベルの定量的プロジェクト管理”に関しては，プロジェクトダッシュボードの詳細表示を，もっと多角的な側面でグラフが見られるように改良する。

6. むすび

定量的プロジェクト管理プロセスの全社展開・普及に向けて，中小規模(特に小規模)プロジェクトでの適用に関しては，利便性の改善やシステム活用のトレーニングなど，継続した改善が必要である。ユーザーインタフェースの改善やプロジェクト評価の多角化，見積り・計画プロセスの支援など，組織の品質・生産性向上につながる機能拡充を，プロジェクトの試行結果を踏まえ，継続して行う計画である。

今後もシステム生産活動の品質・生産性改善を通して，顧客満足度向上につながるように継続的なプロセス改善を推進していく所存である。

特集論文

システム試験の効率化／高精度化技術

川崎将人*
大塚　亮*
後沢　忍*

System Testing Technology for Emulating Production Environment

Masato Kawasaki, Ryo Otsuka, Shinobu Ushirozawa

要　旨

近年，金融や交通などの社会を支える情報システムの障害によって，社会全体に多大な影響を与えることが多くなっている。これらの障害は，旧システムから新システムへの移行時に発生するケースが多いことから，障害の要因はシステムが稼働する環境(＝実環境)と試験環境の差異にあると考えた。そこで，実環境を流れる通信データを試験データとして用いることで，実環境に近い状況を再現できる試験を実現することをねらいとする。

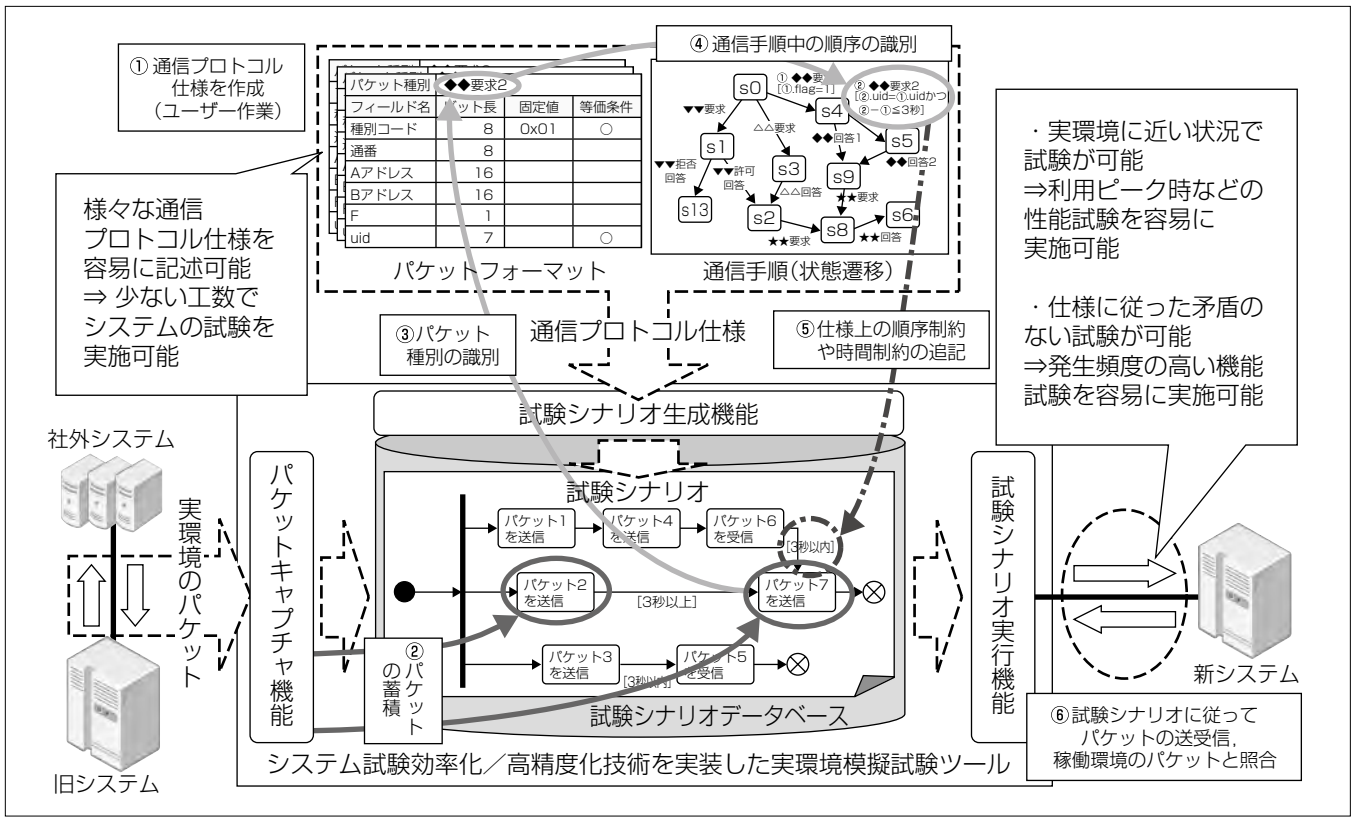
しかし，実環境の通信データを試験環境の新システムに入力しても，出力される通信データは実環境と異なるのが通常である。例えば，送受信の順序やタイミング，パケット数に差異が生じる。その際，試験対象のシステムに不具合があると一概にはいえない。不具合の有無を判別するためには，差異のある通信データが仕様に適合するかを判断する必要がある。そこでこの技術では，仕様への適合性を判別し，通信データの差異を吸収した試験を行えるよう，次の開発を行った。

(1) 仕様上の順序制約や時間制約に従ってパケットの送受信を行うとともに，稼働時と等価なパケットを受信したかを検査する試験シナリオ実行機能

(2) (1)で実行可能な試験シナリオとなるよう，キャプチャしたパケット間に仕様上の順序制約や時間制約を付加する試験シナリオ生成機能

(3) (2)で付加する順序制約や時間制約などの通信プロトコルの仕様の記述に特化した記述形式

この技術によって，少ない工数で様々なシステムの試験が可能になるとともに，実利用に即した機能試験及び性能試験が可能となる。今後は，異常ケースや負荷の追加によって試験精度を更に向上させられるよう，生成した試験シナリオを編集する機能を開発する。



実環境模擬試験ツールの利用手順及び動作概要の説明図

①あらかじめユーザーは，所定の形式に従って，システムが扱う通信プロトコル仕様を記述する。②ユーザーは，既存のパケット キャプチャツールを用いて，実環境の通信データを保存する。試験シナリオ生成機能は，③パケットフォーマットを読み込み，蓄積したパケットの種別を判別し，④通信手順を読み込み，パケットの種別を基にして通信手順中の順序を識別し，⑤識別できた順序と付随する時間制約をパケット間に記録する。⑥試験シナリオ実行機能は，試験シナリオの順序制約と時間制約に従ってパケットの送受信及び実環境のパケットとの照合を行う。

*情報技術総合研究所

1. まえがき

近年，金融や交通などの社会を支える情報システムの障害によって，社会生活に多大な影響を与えることが多くなっている。これらの障害は，旧システムから新システムへの移行時に発生するケースが多いことから，障害の要因はシステムの実環境と試験環境の差異にあると考えた。

一方，システム移行は，長い保守期間中に，ハードウェアなどの寿命によってプラットフォームのみのリプレースが行われることが多い。その際，リプレース後も外部から見て同じ動作を保証する必要がある。つまり，システムの主要なインタフェースであるネットワークで，ネットワークを介してやりとりされる通信データが外部のシステムから見て一致することを保証する必要がある。

そこで，実環境と試験環境の差異を埋めるため，実環境を流れる通信データを利用して試験することを考える。すなわち，実環境を流れる通信データを試験環境の新システムに入力し，出力される通信データが実環境の通信データと等価であるか確認する。これによって，実環境に近い状況を再現し試験することをねらいとする。

しかし実環境の通信データを試験環境の新システムに入力しても，出力される通信データは実環境と同じではないのが通常である。例えば，送受信の順序やタイミング，送受信するパケット数などの差異が発生する。しかしながら，その差異だけで一概にシステムに不具合があるとはいえない。不具合の有無を判別するためには，通信データの差異が仕様に適合するかを判断する必要がある。そこで，通信データの差異が仕様に適合するかを自動的に判断する実環境模擬試験ツールを開発した。

2. システム試験効率化／高精度化技術の実現

2.1 実環境模擬試験ツールの概要

この技術を実装した実環境模擬試験ツールによって，実環境と試験環境の通信データの差異が，仕様に適合するかを自動的に判断することができる。そのためには，プロトコル仕様の順序制約及び時間制約に従って，キャプチャしたパケットを送信し，試験対象から出力されるパケットを受信の上，実環境のパケットと比較し一致しているか検査する必要がある。さらにそれには，試験対象システムが扱う通信プロトコル(＝アプリケーションプロトコル)の仕様を解釈し，キャプチャしたパケット間に仕様上の順序制約及び時間制約を付加する機能が必要である。また，このツールが仕様を解釈できるよう，ユーザーが所定の形式に従って仕様を記述する必要がある。次に，実環境模擬試験ツールの実現方式について述べる。

2.2 実現方式の概要

図1に，実環境模擬試験ツールの機能構成要素を示す。

(1) パケットキャプチャ機能

実環境を流れるパケットをすべて受信し，試験シナリオデータベースに蓄積する機能。Wireshark(1)(注1)などの既存ソフトウェアによって実現されている。

(2) 試験シナリオデータベース

試験シナリオを蓄積する。試験シナリオの構成要素には，送信するパケット，受信を期待するパケット，順序制約情報，時間制約情報などがある。

(3) 通信プロトコル仕様の記述形式

通信プロトコルの仕様記述に特化している。

(4) 試験シナリオ生成機能

(3)の形式で表された仕様を解釈し，試験シナリオデータベースに蓄積されたパケットを解析して，パケット間に仕様上の順序制約情報や時間制約情報を追記する。

(5) 試験シナリオ実行機能

試験シナリオ中の順序制約情報や時間制約情報に従ってパケットの送受信を行うとともに，稼働時と等価なパケットを受信したかを検査する。

次に，特に重要な(3)(4)(5)について詳細に述べる。

(注1) Wiresharkは，Combs, Gerald C.の登録商標である。

2.3 通信プロトコル仕様記述形式

通信プロトコルの仕様には，パケットフォーマットと通信手順の二つがある。

2.3.1 パケットフォーマット

パケットフォーマットの入力イメージを**図2**に示す。

パケットフォーマットは，TCP(Transmission Control

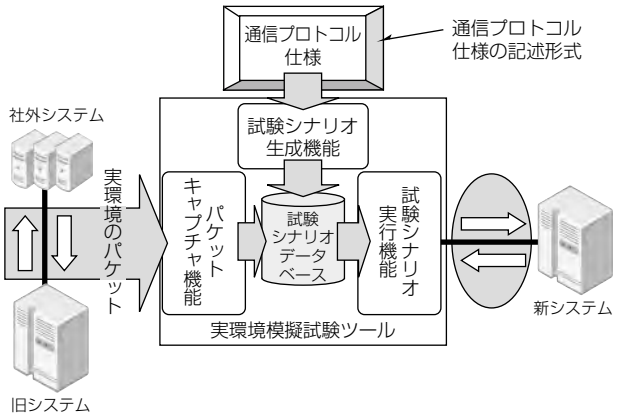


図1．実環境模擬試験ツールの機能構成要素

パケット種別	◆◆要求2		
フィールド名	ビット長	固定値	等価条件
種別コード	8	0x01	○
通番	8		
Aアドレス	16		
Bアドレス	16		
F	1		
uid	7		○

図2．パケットフォーマットの入力イメージ

Protocol)などの下位プロトコルのパケットのペイロードを基にして，アプリケーションプロトコルのパケットのフィールド値を解析するための仕様を記述する。**図2**に示すように，パケット種別ごとに表を持ち，表は少なくとも，フィールド名，ビット長，固定値，等価条件の列を持つ。パケット種別はパケットが持つフィールドの集合を識別するためのものである。フィールド名は，パケットが持つフィールドの名前を示す。ビット長は，フィールド値を格納するのに必要なビット数を示す。固定値は，ビットパターンで表されたパケットから，その種別を識別する際に使用するもので，特定のパケット種別に該当するためには，ビット長によってパケットを分割後にそのフィールド値が所定の値になることを定義する。等価条件は，実環境と試験環境のパケットを比較する際に使用するフィールドを指定する。

2.3.2 通信手順

通信手順の入力イメージを**図3**に示す。

通信手順は，送受信されたパケット間の依存関係を解析し，パケット間に順序制約と時間制約を記録するために使用する。アプリケーションプロトコルと1対1で定義される。

通信手順の基本構造は状態遷移モデルであり，特にプロトコル状態マシンと呼ばれるモデルを採用している。プロトコル状態マシンは通信路における状態遷移を表しており，TCPの仕様記述にも使用されている[2]。

プロトコル状態マシンにおける“状態”とは，通信路から観測可能な状態を表す。例えば，“パケットAを送信後”のような状態を表す。プロトコル状態マシンにおける“遷移”とは，パケットの送受信イベントによる状態の変化を表す。例えば，“パケットAを送信前”から“パケットAの送信”イベントによって“パケットAの送信後”へと遷移する。

プロトコル状態マシンの形式は，OMG(Object Management Group)が規定するUML(Unified Modeling Language)(注2)のように標準化された形式や，Alfaro[3]らのように独自に規定した形式もある。この技術では，一般的な状態遷移モデルに，次の二つの記述制約を追加した形式を使用する。一つ目の制約として，遷移を引き起こすためのイベントには，パケットフォーマットで規定したパケット種別を指定する。二つ目の制約として，状態遷移する際の条件を表すガード条件に対し，次の3種類の組合せという制約を課す。

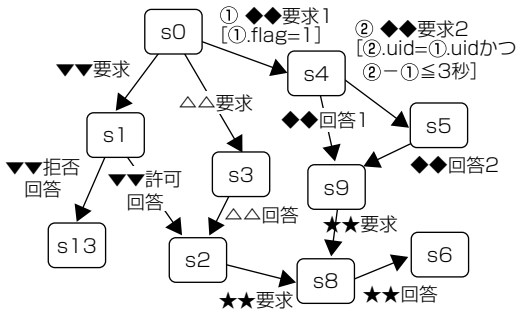


図3．通信手順の入力イメージ

(1) 送受信するパケットのフィールド値と，所定の定数との不等式。例：**図3**中，遷移①の(①.flag=1)。

(2) 送受信するパケットのフィールド値どうしの不等式。例：**図3**中，遷移②の(②.uid=①.uid)。

(3) 送受信するパケットの送受信時刻どうしの演算結果，及び時間差を表す定数との比較演算による不等式。この制約は，時間制約を表している。
例：**図3**中，遷移②の(②−①≦3秒)。

複数の制約が組み合わされている場合には，それらの制約をすべて満たすこと，すなわち論理積を表すものとする。

(注2) UMLは，Object Management Group Inc.の登録商標である。

2.4 試験シナリオ生成実現方式

試験シナリオ生成機能の概要を**図4**に示す。

試験シナリオ生成は，①パケット種別の解析，②パケット間の依存関係の解析，③順序制約と時間制約の追記の，三つの手順で行う。このうち②が試験シナリオ生成処理の要となる処理である。

2.4.1 パケット種別の解析

パケット種別解析処理の概要を**図5**に示す。

パケット種別の解析処理は，蓄積したパケットが，どのパケット種別に該当し，どのようなフィールドと値を持つかを識別するための処理である。処理のポイントを次に示す。

(1) パケットをフィールドに過不足なく分割できるか？

(2) 分割したフィールドの値が，固定値カラムの値で示した値すべてに一致するか？

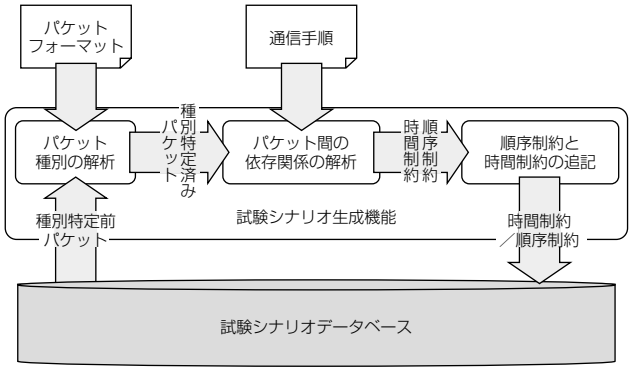


図4．試験シナリオ生成機能の概要

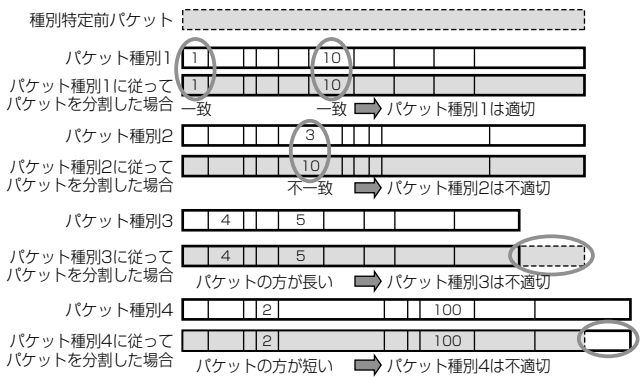


図5．パケット種別解析処理の概要

なお，可変長のパケットに対応する場合や，下位プロトコルのパケット複数個からアプリケーションプロトコルのパケットを得る場合でも，この技術を拡張することで容易に対応することができる。

2.4.2 パケット間の依存関係の解析

パケット間の依存関係解析処理と制約追記処理の概要を**図6**に示す。

TCPで接続の確立から切断までがそうであるように，一つの送受信シーケンスは，通信手順における初期状態からの遷移の系列(＝状態遷移パス)に相当する。そこで，一つの状態遷移パスを表すデータとしてトークンを使用する。**図6**に示すように，トークンは記録したパケットと，通信手順における遷移の対応付けのリストを持つ。

このような前提の下，パケット間の依存関係を解析していく。依存関係を解析する上での基本動作は，次々にパケットを取得し，次の2点を解析する処理である。

(1) 取得したパケットがどのトークンに属するか？(**図6**中でトークンと対応付けの間の実線に相当)
(2) 取得したパケットが通信手順上のどの遷移に該当するか？(**図6**中の破線矢印に相当)

なお，状態遷移の終了状態に到達した場合には，パケットをトークンに対応付ける対象とはしない。これはTCPにおけるコネクションの切断に相当する。また，取得したパケットが，状態遷移の最初の遷移を起こすイベントと一致する場合には，トークンを新たに生成する処理を行う。これはTCPにおけるコネクション確立処理の開始に相当する。

2.4.3 順序制約と時間制約の追記

パケット間の依存関係を解析後，識別された順序制約とそれに付随する時間制約を試験シナリオに追記する。**図6**では，処理①～④の実線矢印で示した処理が該当する。

図6の処理①は，順序制約の追記処理である。トークンが保持する隣り合う対応付け(**図6**中の対応付け①②)がそれぞれ指すパケット(**図6**中のパケット①②)の間に，順序制約情報(**図6**中の順序制約①)を追記する。

ガード条件の追記処理のポイント3点を，**図6**を用いて次に述べる。それぞれ**2.3.2項**の(1)～(3)に対応する。

(1) 定数値との比較の場合は，フィールド値制約情報(フィールド値制約①③)として試験シナリオに追記(処理②)。
(2) フィールド値同士の比較の場合（例えば遷移②の条件“②.uid=①.uid”)は，先行する他の遷移(遷移①)に対応するパケット(パケット①)の具体的なフィールド値（uid=5と仮定する)に変換し，フィールド値制約情報（フィールド値制約②)として試験シナリオに追記(処理③)。
(3) 時間制約の場合は，対応するパケット同士(パケット①②)の間に時間制約情報(時間制約①)を追記(処理④)。

2.5 試験シナリオ実行実現方式

生成された試験シナリオは，UMLアクティビティ図に

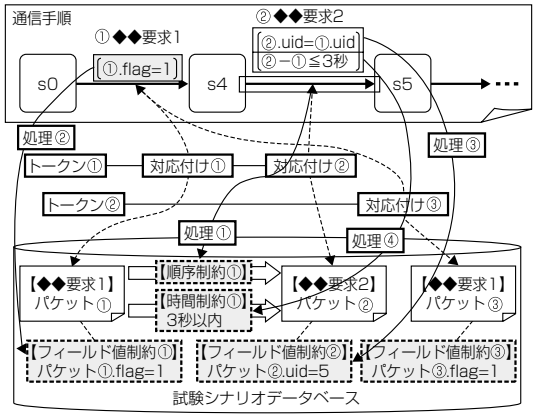


図6．依存関係解析処理と制約追記処理の概要

似たデータ構造を持っている。すなわち，蓄積されたパケットがアクション，順序制約が制御フロー，時間制約がガード条件に対応する。そのため，紙面の都合上，説明は省略するが，**2．4節**の結果出力される試験シナリオを基に，UMLアクティビティ図のデータ構造に容易に変換できる。

したがって，試験シナリオ実行時の基本動作は，UMLアクティビティ図のデータ構造で，開始ノードから終了ノードに至るまでのパスをたどる動作になる。さらにこの技術では，次の二つを行う。

一つ目は，パケットを送受信した際に，その時刻を記録する。これは，後に送受信するパケットの送受信時刻が時間制約を満たすか検査するためである。

二つ目は，受信パケットのアクションでは，パケットの受信を待機するだけでなく，パケットを受信した際には試験シナリオ中のパケットと比較するとともに，時間制約を満たすか検査する。比較時には，**2.3.1項**で示したパケットフォーマットで，等価条件カラム値に“○”を記述したフィールドを対象として比較を行う。対象となるフィールドすべてについてフィールド値が一致した場合に限り，パケットは等価であると判断する。

3. むすび

この技術によって，少ない工数で様々なシステムの試験が可能になるとともに，実利用に即した機能試験及び性能試験が可能となる。今後は，異常ケースの追加，負荷の追加によって試験精度を更に向上させられるよう，生成した試験シナリオを編集する機能を開発する。

参考文献

(1) Wireshark：http://www.wireshark.org/
(2) RFC 793-Transmission Control Protocol：http://www.faqs.org/rfcs/rfc793.html
(3) Alfaro，L. de，et al.：Inteface automata，ACM SIGSOFT Software Engineering Notes，26，Issue 5，109～120（2001）

特集論文

ハイブリッドセキュリティ診断技術

河内清人*
藤井誠司*

Hybrid Security Assessment Technology

Kiyoto Kawauchi, Seiji Fujii

要　旨

近年，相次ぐWebアプリケーションからの情報漏えい事件を受け，Webアプリケーションの脆弱(ぜいじゃく)性を検査するWebアプリケーションセキュリティ診断サービスの重要性が広く認識されている。診断サービスを効率化する上で，サイト内で自動到達可能な範囲(診断可能範囲)の拡大，及び診断項目の一層の充実が求められている。

この課題を解決するために，三菱電機ではWebアプリケーションの静的解析と動的解析を組み合わせたハイブリッドセキュリティ診断技術の研究開発に取り組んでいる。この技術の特長は次のとおりである。

(1)　Webアプリケーションを漏れなく診断

Webアプリケーションを静的解析し，ページ抽出と各ページへの到達経路決定を行うページ間依存性解析で，従来技術では到達できなかったページも診断可能

(2)　業界標準の診断項目(OWASP Top 10)を自動診断

診断データ入力時と正常データ入力時との応答の類似性から脆弱性有無を判定するページ類似性判定によって，従来自動診断が困難であった項目を自動化することで実現

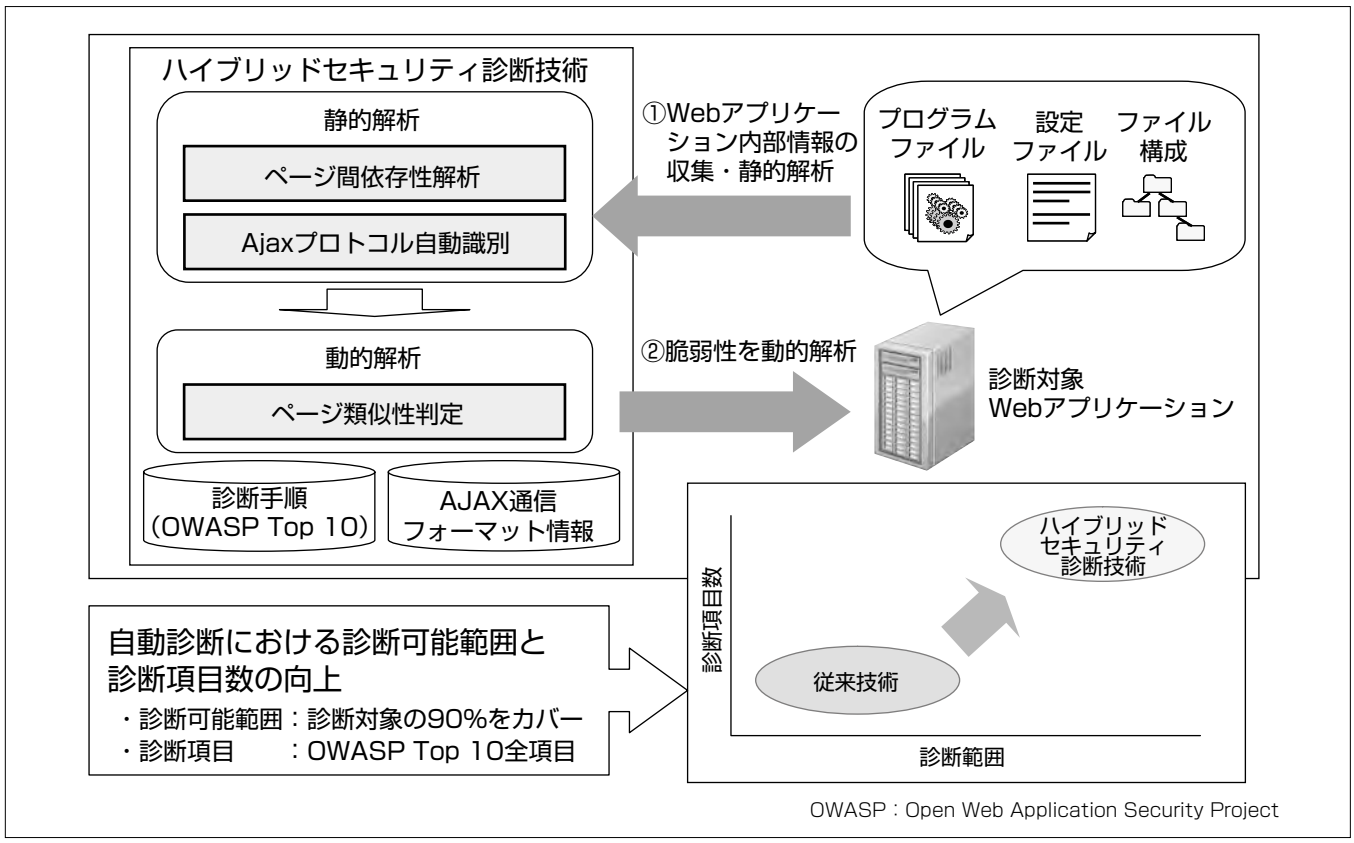
(3)　Ajax(Asynchronous JavaScript＋XML)(注1)への対応

Webアプリケーション内の情報からAjaxプロトコルを自動識別し，Ajaxに対する自動診断を実現

本稿ではまずハイブリッドセキュリティ診断技術について述べ，次に，SaaS(Software as a Service)型(注2)Webアプリケーションセキュリティ診断サービスへこの技術を適用した場合の実現形態について述べる。

(注1)　ブラウザ上のJavaScriptとWebアプリケーションが非同期に通信を行い，ページを動的に更新する技術。なお，JavaScriptは，Sun Microsystems, Inc. の登録商標

(注2)　ソフトウェア機能をサービスとして提供する形態



ハイブリッドセキュリティ診断技術の概念図

ハイブリッドセキュリティ診断技術は，Webアプリケーションの静的解析と動的な診断を組み合わせることで，診断範囲と診断項目数を向上させる。ページ間依存性解析は，Webアプリケーション内の各ページへの正しい到達経路を決定する。Ajaxプロトコル自動識別は，使用されているAjaxに対応した診断メッセージを生成する。ページ類似性判定は，従来脆弱性有無の自動判定が困難であった診断項目の自動化を可能にし，OWASP Top10全項目の診断を実現する。

*情報技術総合研究所

1. まえがき

今や，企業がインターネット上にWebアプリケーションの動作するWebサイトを公開し，顧客に対して様々なサービスを提供する形態が当たり前になってきている。顧客の個人情報など機密性の高い情報も大量に扱うようになるにつれ，Webアプリケーションは不正アクセスの格好の標的となり，昨今Webアプリケーションの弱点(脆弱性)を悪用した情報漏えい事件が相次いでいる。

そのため，インターネットに公開するWebアプリケーションに脆弱性が含まれていないか検査を行うWebアプリケーションセキュリティ診断を実施することは今や不可欠であり，三菱電機情報ネットワーク㈱(MIND)をはじめ，各社から同診断サービスが提供されている[(1)]。

Webアプリケーションセキュリティ診断を実施する上で，自動診断ツールによる効率化は不可欠である。そのため，自動診断技術には診断可能範囲の拡大，及び診断項目の一層の充実が求められている。

当社では，Webアプリケーションセキュリティ自動診断における診断範囲の拡大と診断項目の拡充を目指し，ハイブリッドセキュリティ診断技術の開発を行っている。

2. ハイブリッドセキュリティ診断技術

Webアプリケーションセキュリティ診断サービスで用いられる診断ツールは，Webアプリケーションを実際に動作させることで脆弱性の有無を診断する動的な診断方式が一般的である[(2)]。

診断ツールは，攻撃を模擬した異常なリクエストをWebアプリケーションに入力し，それに対するWebアプリケーションの応答を観測することで，脆弱性が含まれているかどうかを検査する。

それに対し，ハイブリッドセキュリティ診断技術は，Webアプリケーションの静的解析と動的な診断のハイブリッドであり，これによって次の特長を実現する。

(1) Webアプリケーションを静的解析し，Webアプリケーション上の全ページを抽出後，さらに各ページへの到達経路決定を行うページ間依存性解析で，従来技術では到達できなかったページも診断可能

(2) 異常なリクエスト送信時の応答と，正常なリクエスト送信時の応答に含まれるページ類似性に基づいた脆弱性判定技術によって，業界標準の診断項目(OWASP Top10 [(3)])のうち，誤検知の可能性が高く，従来専門家による手動診断で実施していた項目を自動化

(3) Webアプリケーション内部の情報からAjaxフレームワークを自動認識し，各フレームワーク固有の通信プロトコルに準拠した診断メッセージを生成することで，Ajaxによって呼び出されるWebアプリケーション機能への診断を実現

2.1 ページ間依存性解析による診断可能範囲拡大

従来の診断ツールは，Webアプリケーション全体を診断するため，ページ内のリンクやフォームを辿(たど)り，Webサイトを巡回して診断対象ページの探索を行う。

しかし，ページ間の依存関係によって，巡回時に表示されないページが診断対象から漏れるという課題があり，Webアプリケーションを網羅的に巡回することはできていない。

例えば，**図1**に示すようなショッピングサイトを考える。このWebアプリケーションでは，“カートに追加”操作を行っていない限り，“決済”ページで決済用フォームが表示されない。ツールがカートに商品を追加する前に決済ページを巡回すると，ツールはそのようなフォームが存在することを認識できず，診断対象から漏れてしまう。

つまり，Webアプリケーション内のページをすべて診断するためには，ページ間のリンクで表される依存関係以外の隠れた依存関係を見つけなければならない。従来のように，Webページ上に現れる情報だけを用いてこの隠れた依存関係を特定することは困難であった。

ハイブリッドセキュリティ診断技術では，Webアプリケーション内部のプログラムファイルや設定ファイルを静的に解析することで，各ページ間の隠された依存関係を特定するページ間依存性解析によってこの課題を解決する。既存ツールを用いた評価結果からこの課題を解決することで，診断対象の90%程度まで診断可能範囲を拡大できると考えられる。

ページ間依存性解析は**図2**のように行われる。まず，Webアプリケーションが格納されているディレクトリの構成や，設定ファイルの情報から，Webアプリケーションとして用意しているページの一覧を取得する。次に，各ページを処理するプログラムファイル内の実行コードを解析し，セッション変数などプログラムファイル間で共有されるデータの設定・参照関係を調査することで，ページ間の依存関係を明らかにし，依存関係を満たすよう各ページへの到達経路を決定する。

これらの処理によって，ページ間に張られたリンク以外

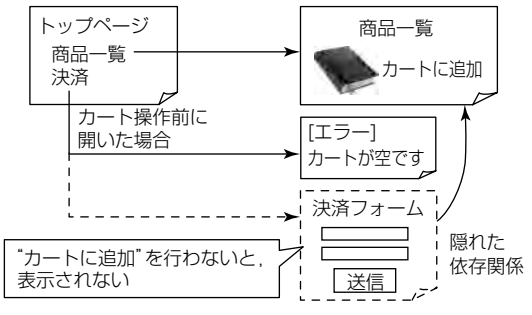


図1．隠れた依存関係が存在するサイトの例

の隠れた依存関係を明らかにでき，Webアプリケーション内のページを漏れなく診断することが可能となる。

2.2 診断前後のページ類似性に基づいた脆弱性判定

セッション管理の不備など，OWASP Top 10の診断項目の中には，異常な入力を与えたあとのWebアプリケーションからの応答がエラー画面かどうか判定しなければならないものがある。

従来は“エラー”などのキーワードが応答HTML(HyperText Markup Language)に含まれているか確認したり，レスポンスのステータスコードがWebアプリケーション内でのエラー発生を示しているか確認したりする手法がとられてきた。

しかし，Webアプリケーションが常にこれらの条件に一致するエラー応答を返すとは限らず，脆弱性有無の判断を誤る場合があり，診断結果を専門家が確認する必要があった。

この課題を解決するため，エラー発生時に表示される画面が正常にアクセスした場合と大きく異なるという特徴に着目し，異常な入力を与えた際の応答ページと，正常にアクセスした場合の応答ページの類似性を測定することでエラーかどうかを判定する方式を開発した。

ページ類似性の判定は，**図3**のように行われる。まず，比較対象となる二つの応答ページに対し，正規化処理が行われる。正規化とは，各ページに対して一種の整形を行う処理であり，判定精度を向上させることが目的である。

整形後のページは，テキスト比較によってそれらの差分が求められる。差分の大きさが閾値(しきいち)を超えた場合に，類似性が失われた，すなわちエラーが発生したとみなす。

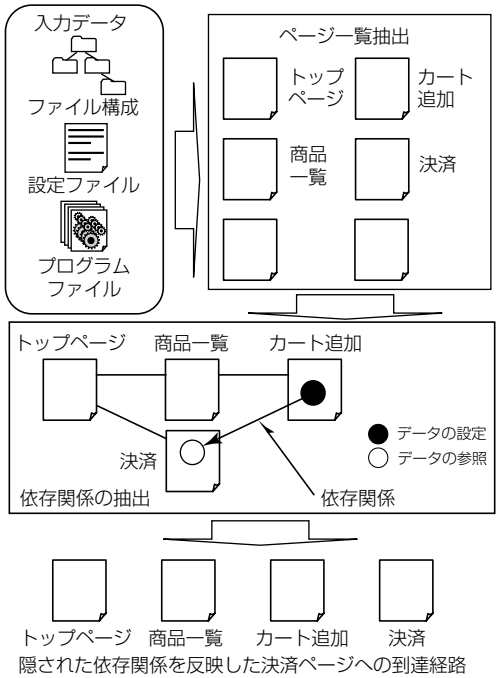


図2．ページ間依存性解析処理

2.3 Ajaxプロトコル自動識別

近年Webページの見映えや操作性を向上させるため，Ajaxと呼ばれる技術が急速に普及してきている。Ajaxは，ブラウザ上のJavaScriptとWebアプリケーションが非同期に通信を行い，表示されているWebページを動的に更新する技術である。

Ajaxでは，JavaScriptとWebアプリケーション間は自由なプロトコルでデータを送受信することが可能である。したがって，Ajax通信で呼び出されるWebアプリケーションの機能を診断するためには，サイトごとに異なるAjaxのプロトコルに準拠した診断メッセージを生成できなければならない。

この課題を解決するため，**図4**にあるように，診断対象Webアプリケーション内部で使用されているライブラリ名や設定ファイルの内容からAjaxフレームワークを特定し，その情報に基づいてAjax通信に対する診断メッセージを自動生成する技術を開発した。

3. SaaS型Webアプリケーション セキュリティ診断サービスへの適用

従来の診断サービスは，専門的な技術者がツールを駆使して行うのが一般的であった。そのため，高額なサービスとならざるを得ず，セキュリティ予算が潤沢ではない利用者は，Webアプリケーションのごく一部分のページだけしか診断を受けることができなかった。

そこで近年，診断プログラムによる自動診断のみを提供するSaaS型のWebアプリケーションセキュリティ診断サ

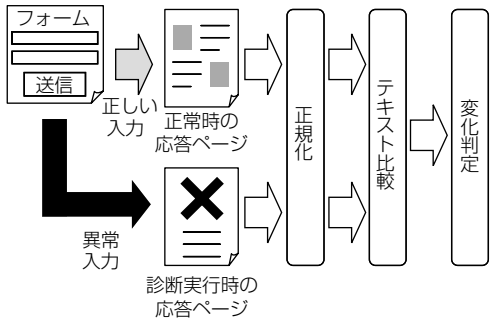


図3．ページ類似性判定

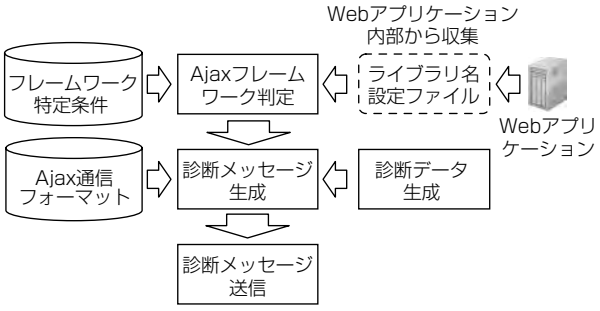


図4．Ajaxプロトコル自動識別

ービスが登場し，脚光を浴びている。このサービスは，診断にかかる人手を徹底的に排除することで，サイト全体を安価に診断できることを特長としている。

ハイブリッドセキュリティ診断技術をSaaS型Webアプリケーションセキュリティ診断に適用することで，既存サービスよりも診断項目数や診断対象サイトの診断範囲に優れた診断サービスを提供することが可能となる。ハイブリッドセキュリティ診断技術をSaaS型Webアプリケーションセキュリティ診断サービスに適用した場合のシステムイメージを**図5**に示す。

この診断サービスの流れは次のようになる。

(1) 診断サービスを希望するユーザー(Webサイト管理者)は，サービスプロバイダの提供するWebポータルにアクセスし，診断の申込みを行う。申込み完了後，Webポータルから構成情報収集プログラムをダウンロードし，診断対象のWebサーバ上で実行する(**図5**①，②)。

(2) 構成情報収集プログラムは，Webサーバ内で動作し，**2．1節**及び**2．3節**で述べた処理に必要な情報を収集する。収集した結果は，構成情報ファイルとして出力される。

(3) Webサイト管理者は出力された構成情報ファイルを回収し，必要ならば内容を確認後，ユーザーの使用しているパソコンから，Webポータルにファイルをアップロードする（**図5**③，④)。

(4) Webポータルにアップロードされた構成情報ファイルは，ハイブリッドセキュリティ診断サーバに入力され，解析が行われる。ハイブリッドセキュリティ診断サーバは解析の結果得られた情報に基づいて診断を実施する(**図5**⑤)。

なお，構成情報収集プログラムが直接ファイルを診断サーバにアップロードせず，このように，ユーザーが構成情報ファイルを回収し，ユーザーにファイルをアップロードさせる手順をとったのは，次の理由による。

①Webサーバの動作しているDMZ(DeMilitarized Zone)とインターネットとを分離するファイアウォールでは，通常Webサーバから外部への接続は許可していない。

②ユーザーに診断サーバにアップロードされる情報に機密情報が含まれていないことを確認する機会を与える。

4. むすび

本稿では，Webアプリケーションセキュリティ診断の自動化における診断可能範囲の拡大，及び診断項目の一層の充実を目指したハイブリッドセキュリティ診断技術について述べ，近年注目されているSaaS型Webアプリケーションセキュリティ診断サービスへの応用について述べた。

この技術はWebアプリケーションセキュリティ診断への適用を目指して開発を進めてきたが，その要素技術はWebアプリケーションのシステムテストにも適用可能と考えられる。今後はこれらの分野も視野に入れ，引き続き技術開発を進めていく予定である。

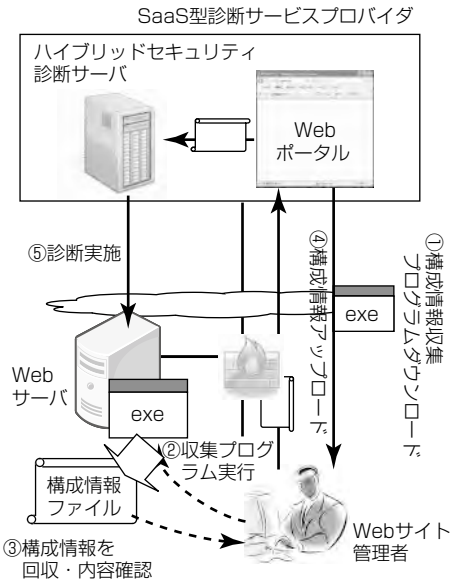


**図5．SaaS型Webアプリケーション
セキュリティ診断サービスへの適用**

参考文献

(1) 三菱電機情報ネットワーク：“Webアプリケーションセキュリティ診断サービス”
http://www.mind.co.jp/service/security/managed/application.html

(2) 河内清人，ほか：統合セキュリティ診断ツールに対するWeb診断機能の拡張，第66回情報処理学会全国大会（2004）

(3) Open Web Application Security Project(OWASP)：OWASP Top Ten Project
http://www.owasp.org/index.php/Category：OWASP_Top_Ten_Project

(4) Hoffman，B.，ほか： Ajaxセキュリティ，ISBN978-4-8399-2842-1（2008）

特集論文

高信頼性を実現するシステム仕様検査技術

上野浩一郎*
磯田　誠*
市原利浩*

System Specification Verification Tool for Highly-dependable System

Koichiro Ueno, Makoto Isoda, Toshihiro Ichihara

要　旨

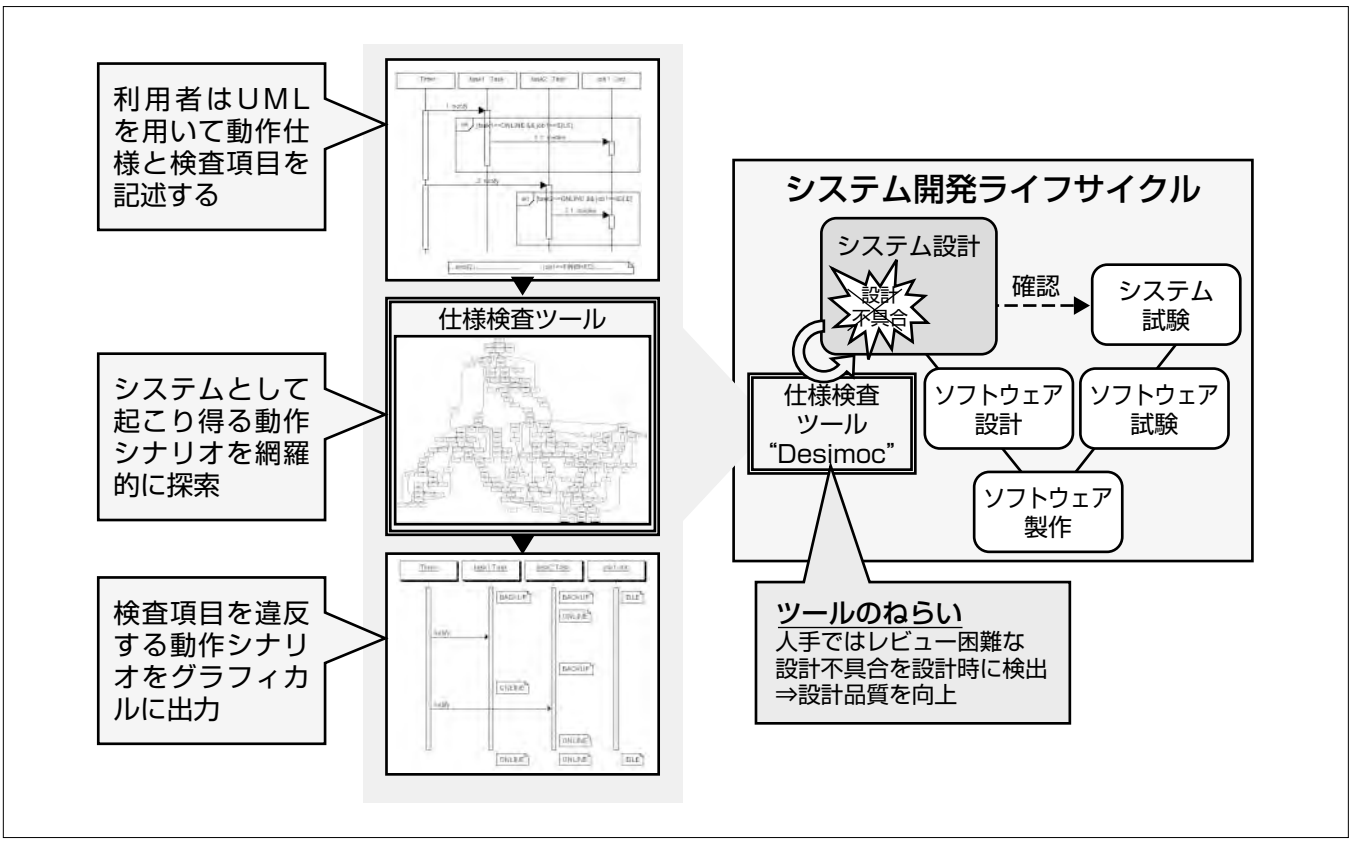
近年，社会活動が情報システムへの依存度を増しているため，情報システムに対して一層の信頼性が求められている。しかし従来のシステム開発では，システム仕様を人手でレビューしており，設計不具合が設計工程から流出する可能性があった。さらにシステム試験では，流出した設計不具合を含む仕様を正しいとして試験するため，不具合を検出できない危険がある。したがってシステムの高信頼性を実現するには，システム仕様中の設計不具合を網羅的に摘出する必要がある。

そこで三菱電機では，設計品質の向上を目的に，システム仕様を網羅的に自動検査するツール“Desimoc”を開発した。システム設計者は，Desimocに対してシステムの動作仕様と，システムが満たすべき検査項目を入力する。Desimocは入力された動作仕様を基に，システムとして起こり得る動作シナリオを網羅的に探索し，検査項目に反する動作シナリオを検出する。Desimocで扱う検査内容を次に示す。

①到達可能性（初期状態から特定状態に到達できる）
②安全性（デッドロックや無限ループがない）
③活性（いつか必ず特定状況が発生する）
④公平性（各々の並行処理が必ず進む）

Desimocが網羅的に動作シナリオを探索するため，人手のレビューでは検出困難な設計不具合も検出可能である。

今回，Desimocを開発済みシステム仕様に対して試行適用し，マニュアル運用で回復すべき稀（まれ）な異常動作を検出できた。この試行によってDesimocで設計不具合をなくし，システムの信頼性が向上することを確認した。



システム仕様を検査するツール“Desimoc”

Desimocは，人手ではレビュー困難な設計不具合をシステム設計時に検出するためのツールである。Desimocの利用者はUML（Unified Modeling Language）を用いて，検査対象システムの動作仕様と検査項目を記述する。Desimocはこれらを入力して，システムとして起こり得る動作シナリオを網羅的に探索する。そして，検査項目を違反する動作シナリオをグラフィカルに出力する。設計者はDesimocが出力した動作仕様から設計不具合を修正する。

*情報技術総合研究所

1. まえがき

近年，社会活動は情報システムに依存し，障害が発生した場合の経済的かつ人的損失の影響は非常に大きい。しかしシステムが大規模化・複雑化した結果，開発時に不具合を洩(も)れなく発見することが従来に比べ困難になっている。

これに対して経済産業省は，システムに関係する組織が遵守すべき“情報システムの信頼性向上に関するガイドライン”[(1)]を公表した。このガイドラインでは，高信頼性を実現する手段として“形式手法”の適用を推奨している。

当社は，情報システムを高信頼化するために，形式手法の一つであるモデル検査技術を用いて，システム設計段階で仕様の正しさを検査するツール“Desimoc”を開発した。本稿ではDesimocの概要と適用例について述べる。

2. 仕様検査技術

2.1 形式手法とモデル検査

形式手法とは，計算機科学における論理学や離散数学に基づいて，システムの仕様を記述し検証する技術の総称である。本稿で述べる仕様検査ツールDesimocは，形式手法の一つであるモデル検査技術を採用している。モデル検査とは，オートマトン／時相論理／グラフ理論を用いて，システム動作を網羅的に検査する技術である。モデル検査を適用できる対象例を次に示す。

①システムを構成するサーバ間のメッセージシーケンス
②Webサービスや分散オブジェクト間の呼び出しシーケンス
③通信装置と対向装置間の通信プロトコル仕様

モデル検査技術の実システム開発への適用は欧米が先行しているが，最近では国内でもこの技術に取り組み始めている。例えば宇宙航空研究開発機構では，宇宙機の制御ソフトウェアの検査に適用している[(2)]。また産業技術総合研究所では，モデル検査の理論，利用プロセス，ツールなどの体系をMCBOK2008(Model Checking Body Of Knowledge)として公開した[(3)]。

2.2 仕様検査ツールのねらい

本稿で述べる仕様検査ツールは，モデル検査技術を用いて，システム設計時に仕様の正しさを検査するツールである。従来のシステム開発では，システム仕様を人手でレビューしていた(**図1**)。これでは設計不具合が下流工程に流出する可能性があった。設計不具合が信頼性にとって問題なのは，流出した設計不具合を含む仕様を正しいとしてシステム試験するため，試験でも不具合を検出できない危険が増す点である。仕様検査ツールのねらいは，設計品質を向上させることである。仕様検査ツールを用いたシステム開発では，ツールが仕様を網羅的に検査するので，設計不具合が下流工程へ流出することを抑止できる。

3. 仕様検査ツール“Desimoc”

3.1 ツール機能

Desimocは，**図2**に示す①②③の順に動作する。

(1) 仕様モデルの入力(**図2**①)

開発者が作成した“仕様モデル”を入力する。仕様モデルとは，システムがどのように振る舞うかを記述した“動作仕様”と，システムが常に満たすべき制約である“検査項目”である。検査可能な項目は次の4種類である。

①到達可能性(初期状態から特定状態に到達できる)
②安全性(デッドロックや無限ループがない)
③活性(いつか必ず特定状況が発生する)
④公平性(各々の並行処理が必ず進む)

(2) 動作シナリオの探索(**図2**②)

(1)で入力した動作仕様に対して，実行タイミングの組合せによって様々な動作シナリオが起こり得る。これらの動作シナリオを，Desimocは重複なく網羅的に探索する。

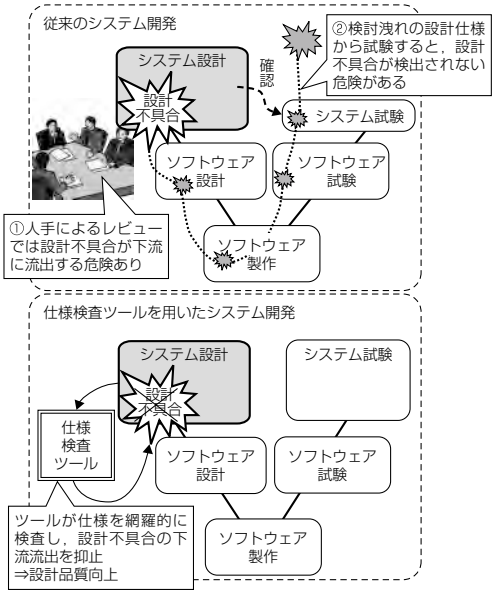


図1．仕様検査ツールのねらい

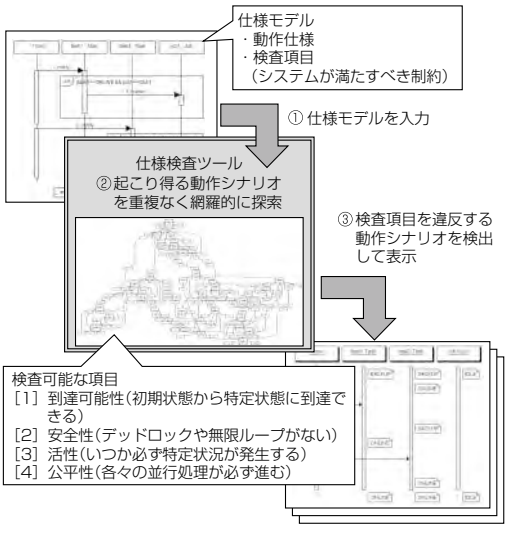


図2．仕様検査ツール“Desimoc”の動作概要

“動作シナリオを重複なく網羅的に探索”する点が，システム動作を確率的に模擬するシミュレーションとの違いである。

⑶ 検査項目違反の動作シナリオの出力(**図2**③)

⑵で探索した動作シナリオのうち，⑴で入力した検査項目を違反する動作シナリオを検出して，グラフィカルに表示する。

3.2 利用方法と効果

Desimocの利用方法と効果は，**図3**に示すように三つある。

⑴ 設計不具合の修正

システムでは様々な動作シナリオが起こり得るため，すべてを人手でレビューするのは困難である。Desimocが出力する検査項目を違反する動作シナリオを確認することによって，稀なタイミングでしか発生しない動作の検討洩れを抑止し，設計品質を向上させる。

⑵ 異常時の回復仕様の追加

検査項目を違反する動作シナリオの中には，例えばネットワークの回線障害など，それが発生しないことを前提にできない内容がある。この場合，検査項目を違反する動作シナリオに対して，システム設計で“異常時の回復仕様”を追加し，この仕様に対してシステム試験で“異常系の試験ケース”を作成する。Desimocは網羅的に検査項目を違反する動作シナリオを出力するので，異常系の試験ケースの網羅性が向上し，試験品質を向上させる。

⑶ 拡張開発時の仕様確認

情報システムの拡張開発時，仕様モデルの動作仕様は拡張のため変更されるが，検査項目はシステムとして本質的なため不変であることが多い。この場合，仕様変更した動作仕様と，前回開発の検査項目をDesimocに入力して，検査項目を違反する動作シナリオがないことを確認すべきである。このように，仕様変更に伴う悪影響の有無を設計時に確認することによって，安全なシステム移行を実現する。

3.3 Desimocの特長

従来の仕様検査ツールの利用者は，**図4**に示すように，固定的に埋め込まれた汎用(はんよう)記法に則(のっと)って，仕様モデルを記述し，出力された動作シナリオを解釈する必要があった。この場合，記法が汎用的過ぎるため，開発者にとってツールを使いこなすハードルが高かった。

Desimocの特長は，入力する仕様モデルと出力する動作シナリオの記法を，開発者が担当する分野の特徴を反映した記法に交換してプラグイン可能な点である。このようにプラグイン可能な記法をドメイン特化記法と呼ぶ。特定分野向けに記述しやすく読みやすいドメイン特化記法を，Desimocにプラグインして利用することによって，開発者とツール間のギャップを解決する。

4. 適 用 例

4.1 適用対象

仕様検査ツールDesimocの効果を確認するために開発済みシステムに試行適用した。対象はすでに稼働中の社会インフラシステムであり，システム仕様書からいくつか仕様を抜粋して，Desimocを用いて仕様検査した。次に，説明のために単純化した仕様を用いてDesimocの入出力例を紹介し，適用結果について述べる。なおDesimocの入出力例は，この適用対象向けに新たに発案した“分散システム向けのドメイン特化記法”に基づいている。

4.2 入力する仕様モデル

図5は，検査対象とした“タイマによるジョブ起動”に対する仕様モデルである。動作仕様として，Timerはtask1とtask2にnotifyを送信し，その際の内部状態に応じてtask1又はtask2がjob1にinvokeを送信する。検査項目は“job1はFINISHED”になることである。次に記法について述べる。構成要素間のメッセージ送受信は，UML[注1)(4)]シーケンス図(**図5**①)で記述する。各構成要素の状態遷移はUML状態マシン図(**図5**②③)で記述する。検査項目は，UMLシーケンス図中にノート(**図5**④)で記述する。検査項目における“end(2)”とはメッセージ番号2のnotifyが実

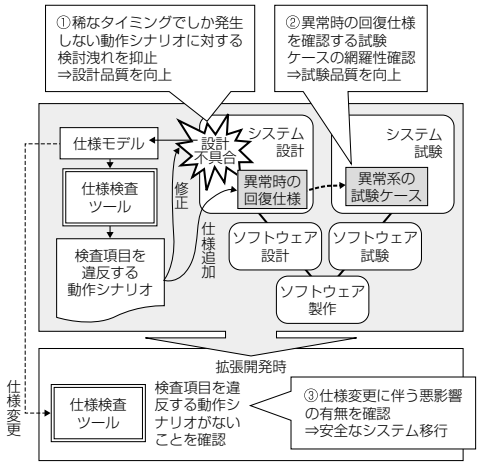


図3．仕様検査ツール“Desimoc”の適用効果

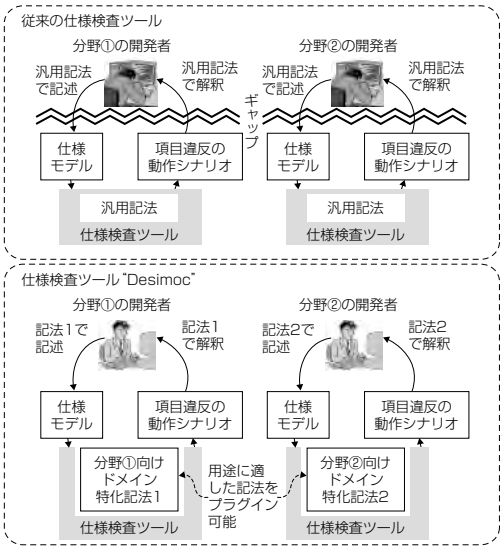


図4．仕様検査ツール“Desimoc”の特長

行終了した時点を示す。したがってこの検査項目は“メッセージ番号2のnotifyが実行終了した後はいつか必ずjob1がFINISHEDになる”ことを意味している。このようにメッセージ番号を使って検査項目を記述可能な点がこのドメイン特化記法の特長である。**図5**に示したend(実行終了)以外にも“メッセージ送信時点”“メッセージ受信時点”を指定可能である。

図5の仕様モデルでは，Taskの内部状態がBACKUPとONLINEを任意のタイミングで遷移(**図5**②)し，この内部状態に依存してjob1に対するinvoke送信が決定する(**図5**①)。このため状態に応じて様々な動作シナリオが起こり得る。なお**図5**①のシーケンスはすべて同期メッセージなのでメッセージ送受信の順番は固定である。一方，非同期メッセージの場合は送受信のタイミングが非決定的なので更に多くの動作シナリオが起こり得る。Desimocはこのような非同期メッセージの検査にも対応している。

(注1) UMLは，Object Management Group Inc. の登録商標である。

4.3 出力する動作シナリオ

図6は，**図5**の仕様モデルで起こり得る動作シナリオのうち，検査項目を違反する動作シナリオの一例である。この動作シナリオは，Timerがnotifyを送信した際，task1とtask2がいずれもBACKUPであるため，job1にinvokeを送信せず，job1の内部状態がIDLEのままとなっている。job1がFINISHEDではないので検査項目を違反していることが分かる。なお**図6**のドメイン特化記法は，UMLシーケンス図に状態遷移を組み合わせて表示するものに拡張している。具体的には，メッセージ送受信(**図6**①)と，各構成要素の内部状態(**図6**②③④)を組み合わせて表示する構成を採用した。

4.4 適 用 結 果

開発済みシステムへの試行適用によって，システムの高信頼化に次の2点で役立つことを確認した。

(1) 適用対象システムはすでに長期稼働中で，仕様が成熟していたため，新たな設計不具合は検出しなかった。ただしDesimocを用いることによって，マニュアル運用でシステム異常を回復すべき稀(まれ)な動作シナリオを検出できることを確認した。これらは，運用マニュアルの改善につなげる。

(2) モデル化記法として，汎用(はんよう)的で自動検査できないUMLではなく，この適用対象に合わせたドメイン特化記法を発案して用いた。このドメイン特化記法を用いて様々な仕様変更を記述して実験することによって，拡張開発時の仕様変更による影響の有無を容易に確認できることを確認した。

5. む す び

システムの信頼性を向上させる仕様検査ツールDesimocについて，その概要と適用例について述べた。今回は，開発済みシステムに対する試行を述べたが，現在は新規開発システムに対して適用評価を進めている。

高信頼なシステム実現のため，次に必要なのはプログラム品質の向上である。今後は，プログラムの実行シナリオを網羅的に探索し，稀にしか発生しないケースを試験可能とするツールを開発する。これらのツールによって，設計から試験のライフサイクル全体で信頼性向上を図っていく。

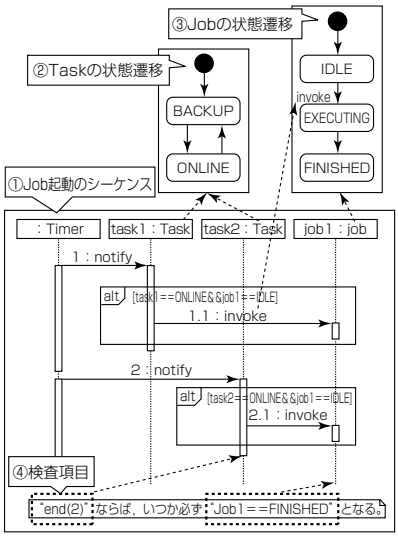


図5．仕様検査ツール“Desimoc”の入力例

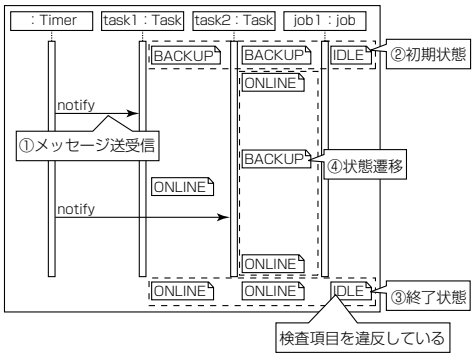


図6．仕様検査ツール“Desimoc”の出力例

参 考 文 献

(1) 経済産業省：情報システムの信頼性向上に関するガイドライン第2版（2009）
http://www.meti.go.jp/press/20090324004/20090324004-4.pdf

(2) 宇宙航空研究開発機構：「ソフトウェア独立検証と有効性確認」技術の研究
http://stage.tksc.jaxa.jp/jxithp/

(3) 西原秀明，ほか：MCBOK2008：ソフトウェア開発のためのモデル検査知識体系，産業技術総合研究所（2009）
http://unit.aist.go.jp/cvs/tr-data/PS2009-009.pdf

(4) OMG：Unified Modeling Language
http://www.uml.org

特集論文

数値計算プログラムのビジュアルな構築環境

井上勝行* 小林康祐***
河合克哉** 梶 正弘***
北村操代*

A Visual Environment for Building Arithmetic Calculation Software

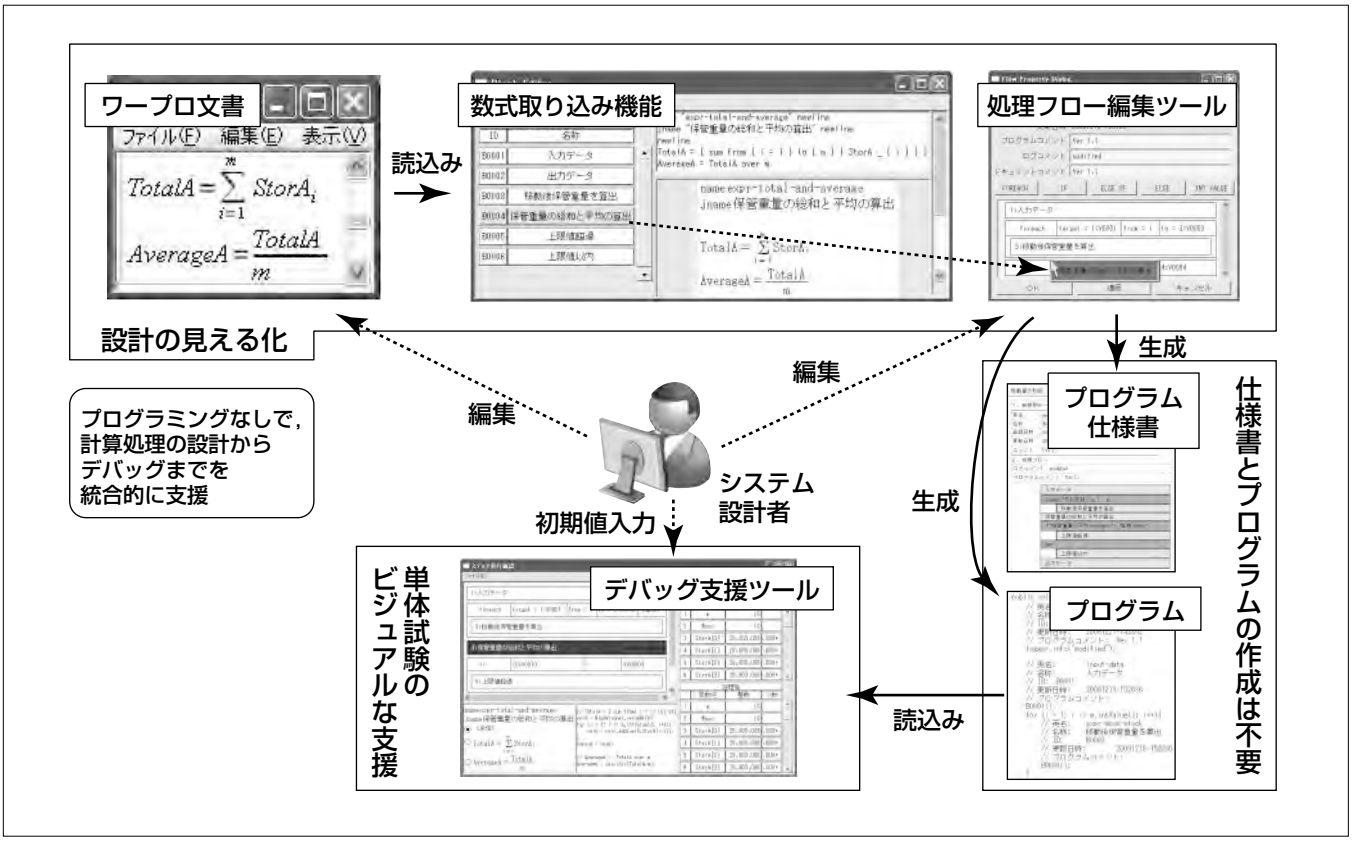
Katsuyuki Inoue, Katsuya Kawai, Misayo Kitamura, Yasumasa Kobayashi, Masahiro Kaji

要 旨

一般に，監視制御システム，工場の運転管理システムなどの産業用システムは，センサなどで取得した数値データを用いて複雑な計算をする機能を備えていることが多い。こうした数値計算機能は，年々高度化・大規模化している。また計算の内容は，法令の改正などの外的要因によってもしばしば変更される。これらの理由から，数値計算機能の構築における生産性の向上は，今後ますます重要となる。

三菱電機では，産業用システムにおける数値計算機能をビジュアルに構築できる数値計算プログラム構築環境を開発した。この構築環境は，数値計算で用いる数式を汎用(はんよう)のワープロソフトウェアで作成した文書から取り込むことができ，それらの実行順序をユーザーがビジュアルに指定するだけで，数値計算プログラムとその仕様書を生成する。またこの構築環境は，数値計算のプログラムコードを見なくても数式のレベルでデバッグできるようにするビジュアルなツールも提供しており，数値計算の設計からデバッグまでを統合的に支援する環境となっている。この構築環境を利用することで，プログラミングの知識がほとんどない技術者でも数値計算機能を構築できるようになり，生産性の大きな向上を実現できる。また同様に，この構築環境が生成する仕様書を見ることで，プログラムで実際に実行される計算の内容を詳細に把握できるようになる。

本稿では，数値計算プログラム構築環境の開発コンセプトと，おもな機能・ツールについて述べる。



数値計算プログラム構築環境による数値計算機能の構築

ユーザーが，汎用のワープロソフトウェア上で入力した数式をこの構築環境へ取り込み，それらの実行順序をこの構築環境上でビジュアルに指定するだけで，数値計算プログラムとその仕様書を生成させることができる。また試験・検証工程用に，数値計算の外部仕様をバッチ処理的に確認するツール，及び数値計算の実行をビジュアルに再現するツールも提供している。

*先端技術総合研究所(工博) **同研究所 ***電力システム製作所

1. まえがき

一般に，監視制御システム，工場の運転・管理システムなどの産業用システムは，センサなどによって取得する数値データを用いて複雑な計算をする機能を備えていることが多い。当社では，こうした数値計算の機能をビジュアルに構築できるプログラム構築環境を開発した。この構築環境を利用すると，汎用のワープロソフトウェアで入力した数式に，実行順序をビジュアルに指定するだけで，数値計算プログラムとプログラム仕様書を生成できる。

本稿では，数値計算プログラム構築環境の開発コンセプトと，おもな機能・ツールについて述べる。

2. 数値計算機能の構築における課題

産業用システムにおける数値計算の内容は，システムの運用が始まってからもしばしば改修される。その原因として，関連する法令の改正，プラントの設備・機器の更新，仕様についてのプログラマの誤解，仕様の漏れなどが挙げられる。そして改修の際には，システム設計者(以下，設計者という。)が仕様書を正しく改訂し，それをプログラマが適切に解釈してプログラムを正確に書き換え，試験技術者が試験の必要な範囲を特定して試験を実施する必要がある。このように，数値計算の改修には工数がかかる。このため，数値計算機能をより容易に構築・改修できることが求められる。

また，プラント技術者，設計者などシステム関係者のほとんどは，プログラミングの知識を持たないのが普通である。このため，これらの関係者が数値計算の詳細を確認する際には，プログラム仕様書に頼ることになる。したがって，プログラムで実行する計算の内容が細大漏らさずプログラム仕様書に記述されていることを保証する必要がある。

3. 数値計算プログラム構築環境

3.1 開発コンセプト

当社では，数値計算の内容をプログラミングしなくても，設計者がビジュアルに構築できるプログラム構築環境が必要と考えた。その構想は，数値計算を構成する数式を汎用のワープロソフトウェアで入力し，それらの実行順序をユーザーがビジュアルに指定することで，プログラム構築環境がプログラムとプログラム仕様書を生成するというものである。この構想によれば，数値計算をプログラミングしなくても，汎用ワープロ上でのビジュアルな数式のレベルで数値計算プログラムを作成でき，さらにプログラムと仕様書の内容が完全に一致することを保証できる。

この構想に基づいて，数値計算プログラムのビジュアルな構築環境を開発・実用化した。この構築環境では，数値計算の処理を処理フローと呼ぶモデルで表現した。**図1**に示すように，処理フローは，処理ブロックを要素とする構造化フローチャートとして数値計算処理を表現したものである。また処理ブロックは，任意個の数式を一つのかたまりとして表現する。処理フローは構造化に対応しているので，処理ブロックの逐次実行と条件分岐だけでなく，数列の各要素に同じ処理ブロックを適用するなどの処理を，繰り返し構造を用いて簡潔に表現できる。

この構築環境を用いて数値計算機能を構築することによって，**2章**で述べた二つの課題を解決できるようになる。

3.2 数値計算のビジュアルな定義

3.2.1 ワープロ文書からの数式の取り込み

この構築環境は現在，OpenOffice.org Writer(以下“Writer”という。)文書からの数式の取り込みに対応している。Writerは，近年急速に普及が進んでいるフリーなオフィス スィートOpenOffice.org[(1)]のワープロソフトウェアで，その標準文書形式は，ISO／IEC 26300規格のODF(Open Document Format)[(2)]に準拠している。また，Writerの数式入力機能はテキスト形式での入力に対応しており，例えば**図2**でTotalAを算出している数式のテキスト表現は，“TotalA = Sum from {i = 1} to {m} {StorA_i}”である。

この構築環境は，このテキスト形式で表現された数式をWriterの文書ファイルから抽出・解析して，プログラムを生成するべき数式と変数の情報を取得する。例えば，**図2**に示すWriter文書を取り込んだ場合，**図3**に示す処理ブロックエディタに取り込んだ数式の内容が表示される。

処理ブロックエディタの左側には，取り込み済みの処理ブロックのIDと名称が列挙され，右側にはユーザーが選択した処理ブロックの内容が，テキスト表現(上段)とワープロ表現(下段)で表示される。一方，変数ビューアには，

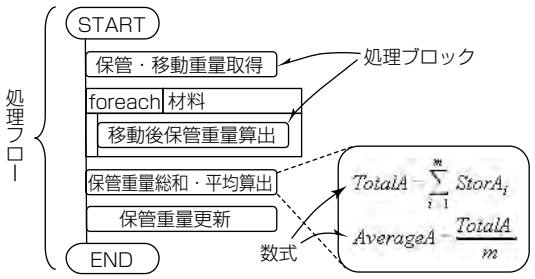


図1．数値計算処理のモデル

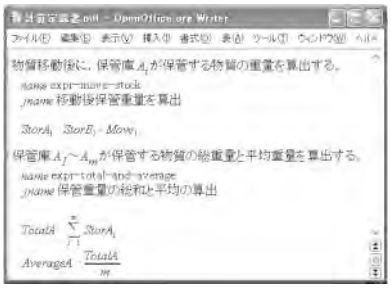


図2．OpenOffice.org Writer文書の例

変数名のテキスト表現，生成するプログラムでの変数名，変数の型(整数，又は有理数)などが一覧表示される。

3.2.2 数式実行順序のビジュアルな指定

Writer文書から数式と変数を取り込んで処理ブロックと変数を構築環境へ登録したのち，処理ブロックの実行手順を処理フローとして定義することによって，この構築環境がプログラムとプログラム仕様書を生成する準備が整う。

処理フローを編集するためのビジュアルなツールの画面を**図4**に示す。この画面下半分の領域へ，処理フローの構成要素をドラッグ・ドロップすることで，処理フローを構造化フローチャートとして作成・編集できる。処理フローの構成要素とは，処理ブロックエディタ上の処理ブロック，**図4**中段にボタンが並んでいる繰り返し構造(FOREACH)，条件分岐(IF，ELSE-IF，ELSE)，及び変数ビューア上の変数と整数定数(INT-VALUE)である。例えば**図4**は，処理ブロックエディタから処理ブロック“保管重量の総和と平均の算出”をドラッグして，“移動後保管重量を算出”とIF要素の間にドロップ(挿入)しようとしている場面である。なお，矩形(くけい)“foreach”の右方にある“1：V0001”，“3：V0003”などは，変数ビューアからドロップした変数のIDである。

3.3 数値計算プログラムの生成

この構築環境は，処理フローを単位として数値計算のプログラムを生成できる。現在は，Java(注1)言語に対応しており，各処理フローは一つのJavaクラスとして生成される。このクラスでは，長けた計算ライブラリを用いて計算を行うため，けたあふれが発生することはない。

(注1) Javaは，Sun Microsystems, Inc. の登録商標である。

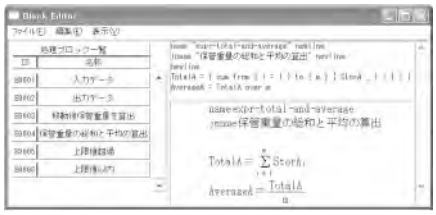


図3．処理ブロックエディタの画面例

図4．処理フロー編集ツールの画面例

3.4 プログラム仕様書の生成

この構築環境は，処理フローの詳細な仕様を記述したプログラム仕様書も生成できる。現時点では，Writer文書形式によるプログラム仕様書の生成に対応している。

プログラム仕様書には，処理フローの名称，更新日時などの管理用データ，処理フローの構造化フローチャート，使用する変数の一覧，各処理ブロックで実行される数式などが記述される。プログラム仕様書の生成例を**図5**に示す。

3.5 設計データの管理

この構築環境では，ユーザーが登録した処理フロー，処理ブロック，変数などのデータを，設計データと総称している。ユーザーは，これら設計データを保存するディレクトリを指定・切替えできる。これによって，数値計算を適用するシステムを随時切り替えて，数値計算機能を構築できる。

また，処理フロー中で使用する処理ブロック，処理ブロック中で使用する変数といった，設計データの使用関係を抽出する機能を備えている。これによって，ほかの設計データが使用している設計データをユーザーが削除できないようにして，設計データの整合性を維持する。

なお設計データは，名称，ID，最終更新日時などの管理用データを埋め込んだテキスト形式のファイルである。このため，設計データのバージョン管理に使用するツールを，様々な汎用ツールの中からユーザーが自由に選択できる。

4. デバッグ支援ツール

数値計算の構築では，ユーザーがその動作を詳細に確認できることが重要である。この構築環境では，単体試験をバッチ処理として実行する動作確認ツールと，処理フローの任意の時点での状況を再現するステップ実行ツールを提供しており，動作確認とデバッグの作業を強力に支援する。

4.1 動作確認ツール

動作確認ツールは，処理フローの動作をバッチ的に確認するためのツールである。ユーザーが処理フローを指定して動作確認ツールを起動し，入出力定義ファイルを指定す

図5．生成したプログラム仕様書の例

ると，動作確認を実行できるようになる。入出力定義ファイルとは，動作を確認するケース(以下，テストケースという。)ごとに，処理フローの実行開始時と正常終了時の変数値を，ユーザーが事前に記述しておいたものである。ユーザーは，任意の数のテストケースを入出力定義ファイルに記述できる。

動作確認ツールは，処理フローからプログラムを生成して，テストケースごとに変数値に初期値を設定して実行し，実行終了時の変数値を正常終了時のものと照合する。照合した結果は，テストケースごとに**図6**に示す動作確認結果詳細ウィンドウに表示される。このウィンドウには，正解値(正常終了時の値)と出力値(実行終了時の値)が，変数ごとに一覧表示され，出力値が正解値と異なる変数(**図6**では，変数StorA[10])の情報がハイライト表示される。

動作確認に用いる入出力定義ファイルは，一つの処理フローに対して必要なだけ作成・保存できる。このため，ユーザーは処理フロー，又は処理ブロックを更新するごとに，過去の動作確認で使用した入出力定義ファイルを用いて，機械的に回帰テストを実施できる。

4.2 ステップ実行ツール

動作確認ツールが実行結果に注目するのに対し，ステップ実行ツールは，処理フロー実行の任意の瞬間を再現するツールである。ステップ実行ツールは，処理フローの実行時に記録しておいた詳細な履歴情報を基に，ユーザーが指定する処理ブロックなどの実行前後での変数の値を表示する。

例えば**図7**左上部に表示している処理フローでは，ユーザーが指定した処理ブロックである“保管重量の総和と平均の算出”を反転表示している。そして，この処理ブロックの内容のワープロ表現が**図7**左下部に，中央下部には，生成したプログラムが表示される。さらに，**図7**右上部には，処理ブロック“保管重量の総和と平均の算出”実行前の変数の状態が一覧表示され，また右下部には実行後の状態が表示される。

再現する瞬間の指定では，処理ブロック以外にも，処理ブロック内の数式を選択したり，繰り返し構造の何回目の繰り返しを再現するかを指定したりもできる。さらに，繰り返し構造全体を選択して，その実行前後の状態を表示させることもできる。このように，ステップ実行ツールを利用すれば，処理フローの実行を詳細に追跡することも，大きな粒度で俯瞰(ふかん)的に確認することも容易である。

5. むすび

産業用システムの発達とともに，その数値計算機能も高度化・大規模化している。また数値計算の内容は，法令の改正などの外的要因によってもしばしば変更される。これ

図6．動作確認結果詳細ウィンドウの表示例

図7．ステップ実行ツールの画面例

らのことから，数値計算機能の構築における生産性向上は，今後ますます重要となる。

本稿で述べた数値計算プログラム構築環境は，数値計算で用いる数式をワープロ文書から取り込み，プログラムとプログラム仕様書を生成する。また，数値計算のデバッグをビジュアルに支援する強力なツールも提供しており，数値計算の設計からデバッグまでの作業全体を統合的に支援するビジュアルな環境となっている。この構築環境を利用することによって，プログラミングの知識がない技術者でも数値計算機能を構築できるようになり，生産性の大きな向上を実現できる。また同様に，プログラムを見なくても，この構築環境が生成する仕様書を見るだけで，プログラムで実際に実行される計算の内容を詳細に把握できるようになる。

この構築環境は，エネルギー関連分野の工場運用支援システムへ適用される予定である。今後は，この構築環境の適用によって削減できる開発量の検証を進めていく。

参考文献

(1) OpenOffice.org—the free and open productivity suite,
http://www.OpenOffice.org/
(2) OASIS Open Document Format for Office Applications(OpenDocument),
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

特集論文

車両情報制御システムのソフトウェア開発プラットフォーム“PLATINA”

辰巳尚吾*
浅井陽介*
渡邉亮一**

"PLATINA"：Software Platform for Train Integrated Management System

Shogo Tatsumi, Yosuke Asai, Ryoichi Watanabe

要　旨

車両情報制御システムは，列車に搭載される機器全体を監視・制御する分散組み込みシステムであり，車両の仕様の多様性に追従した機能の柔軟性，拡張性を実現する上で，ソフトウェアが重要な役割を担っている。車両情報制御システムでは，機種ごとの仕様の差異が大きいため，従来は機種個別のソフトウェア開発が必要となっていた。

複数の機種にまたがるソフトウェア開発の効率化と品質向上のために，“PLATINA（PLatform for Train Integrated management system Nucleus with Advanced technology）”を構築した。PLATINAは，ミドルウェア，開発ワークフロー，開発ツールなどを含むソフトウェア開発プラットフォームである。

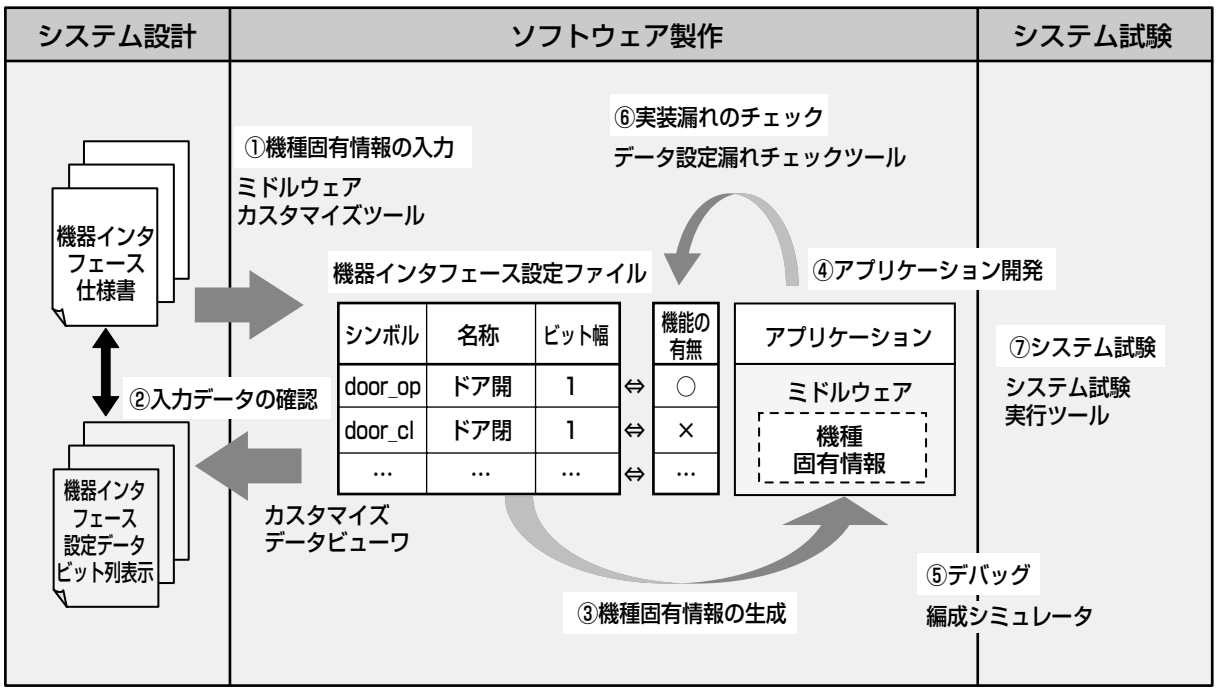
PLATINAを構築することにより，以下を実現した。

(1) 機種間で共通に利用する機能と変動する機能を分析した結果に基づき，ソフトウェア，開発ワークフローを再構築した。

(2) 共通に利用する機能をミドルウェアとして実現し，ミドルウェア，及び，その上位のアプリケーションの再利用を可能とすることにより，新規ソフトウェアの製作量を削減した。

(3) 開発ワークフローにおける各作業を支援するツール群を整備し，作業の効率化を図るとともに人為的ミス混入の可能性を低減した。

これらの総合的な取組みによって，開発効率の向上と信頼性の確保が達成できると考える。



“PLATINA”におけるソフトウェア開発ワークフローと開発ツール

PLATINAは，機種間で共通に利用される機能をミドルウェアとして提供するとともに，ソフトウェア開発ワークフロー（図中の①～⑦）と，これらを支援する開発ツール群（ミドルウェアカスタマイズツール，カスタマイズデータビューア，編成シミュレータ，データ設定漏れチェックツール，システム試験実行ツール）を提供する。これによって，ソフトウェアの機種間での再利用が促進され，新規製作量が削減される。また，ツールによって人為的なミスの可能性が低減される。このため，ソフトウェア開発の効率化と信頼性の確保が達成される。

*先端技術総合研究所　**同研究所（情報科学博士）

1. まえがき

車両情報制御システムは，列車に搭載される機器全体を監視・制御する分散組み込みシステムであり，機能の柔軟性，拡張性を実現する上で，ソフトウェアが重要な役割を担っている。

これまでの車両情報制御システムの開発では，機種間で機能の類似性があるものの，列車の構成，接続する機器の多様性から，ソフトウェアを毎回新規作成する部分が大きな割合を占めていた。近年，車両情報制御システムに接続される機器，要求される機能はますます拡大しており，ソフトウェアの製作量が増加している。このため，ソフトウェア開発効率と品質の向上が重要な課題となっている。

そこで三菱電機は，これらの課題に対応して，ミドルウェア，開発ワークフロー，開発ツールを含むソフトウェア開発プラットフォームPLATINAを構築した[(1)(2)(3)]。PLATINAを構築するに当たり，次の取組みを行った。

(1) 機種間で共通に利用する機能と，変動する機能を分析し，ソフトウェア，開発ワークフローを再構築する。共通に利用する機能をミドルウェア化し，ソフトウェアの機種間における再利用を促進することによって，新規作成量を減らす。

(2) ミドルウェアを核とした開発ツール群を構築する。

本稿では，車両情報制御システムの構成について述べたあと，(1)を実現するための車両情報制御システムの分析，及び，この分析に基づき設計したミドルウェアの構成とその特徴について述べる。そして，(2)の開発ツール群についても述べる。

2. 車両情報制御システムの構成

車両情報制御システムは，演算ノード，表示器及びそれらを接続する伝送路で構成される(**図1**)。個々の演算ノードは，車両に搭載されるブレーキ，ドア，空調などの各機器と伝送，デジタル入出力などを用いてデータを授受する。

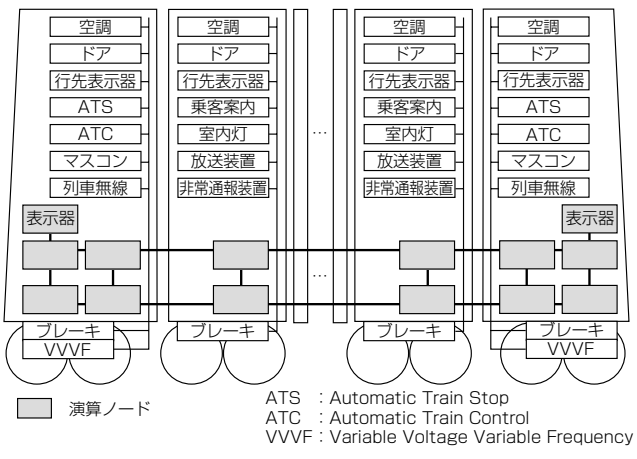


図1．車両情報制御システムの構成例

列車内に複数設置された演算ノードは，伝送路で接続され，ネットワークを構成する。車両情報制御システムは，このネットワークを利用して，列車全体の監視・制御を実現する。

車両情報制御システムのネットワークは，信頼性を確保するために二重系で構成され，冗長性を確保している。演算ノードは対で配置され，演算機能が冗長化される。演算ノード，伝送路のいずれか一箇所の故障に対しては，故障箇所を迂回(うかい)する経路が存在するよう構成される。

3. 車両情報制御システムの変動性と共通性

ここでは，車両情報制御システムの変動性，共通性について述べる。

3.1 変動性

3.1.1 列車構成

車両情報制御システムの機種間の要求仕様，システム構成などに関する変動性について分析した結果，列車構成の変動性が，機種間でのソフトウェア再利用を妨げる要因の一つであることが分かった。列車構成とは，列車全体の編成，車両，機器の構成である。編成は，複数の車両の集合である。車両には，複数機器が搭載される。車両は，搭載される機器の種類，数に応じて決まる属性を持つ。

列車構成の機種間での変動は，演算ノード間の伝送パケットのフォーマット，機器と演算ノード間で授受する伝送パケットのフォーマットに影響を与える。これは，実時間性を実現するために設けたパケットの容量制限を満たすように，列車構成に応じてパケットのフォーマットを最適化する必要があるためである。また，列車構成が変化すると，演算ノードで実行されるソフトウェアが参照するデータの位置や，演算ノードの種類による処理実行の要否などが変化する。

3.1.2 演算ノード上のデバイス

機種によって演算ノードが入出力に用いるデバイスや，不揮発メモリデバイスが変更されることがある。この場合，デバイスドライバの変更が必要となる。

3.2 共通性

3.2.1 冗長構成と制御方法

共通性として，車両情報制御システムが監視制御する機器との伝送における冗長構成とその制御方法，演算ノードの冗長構成とその制御方法，編成の併合・分割時の処理方法など，インフラ部分に関する制御方法が挙げられる。

3.2.2 アプリケーション機能の共通化

変動性として挙げた列車構成を抽象的に表現することによって，アプリケーション機能を機種間で共通化できるものがある。例えば，パンタグラフは，機種によって，搭載位置，個数は異なる。しかしながら，個別のパンタグラフの上げ下げの制御方法や，状態の監視方法は機種間で共通化できる場合がある。また，その処理を実施する演算ノー

ドの指定も，“パンタグラフを搭載しているノード”という抽象的な指定方法をとることによって，共通化できる。

3.2.3　アプリケーション機能の枠組みの共通化

異なる機種間でも共通して利用されるアプリケーション機能が存在する。例えば，監視制御対象の機器との伝送データに応じて，あらかじめ決められた条件によって故障を検知し，それらを記録する故障検知機能，保守時に監視制御対象の機器に対して試験動作を指示し，結果を収集する機能などである。

これらの機能で，個別の条件や試験動作の内容は機種が異なれば変更される可能性があるが，それらを組み合わせてアプリケーション機能を実現する枠組みは共通に利用可能である。

4. ミドルウェア適用によるソフトウェアの再利用

3章で述べた車両情報制御システムの変動性・共通性の分析に基づき，機種間の共通機能をミドルウェアとして構築した。ここでは，この構成と，ミドルウェア適用によるソフトウェアの再利用について述べる。

4.1　ソフトウェア構成

車両情報制御システムのソフトウェアは，ミドルウェアとアプリケーションで構成する。

⑴　ミドルウェア

ミドルウェアは複数の機種で共通に利用される“共通部”と，機種ごとに変化する“機種固有情報”で構成される。共通部は，データ層，インタフェース処理層，ドライバ層から構成され，機器伝送，車両間伝送，CPU(Central Processing Unit)間伝送などの基本機能を持つ。また，ミドルウェアの管理するデータにアクセスするための共通のインタフェースとして，データアクセス関数がアプリケーションに提供される。機種固有情報は，列車構成情報や，演算ノードと機器間で共有されるデータのフォーマット情報などを含む。

⑵　アプリケーション

車両情報制御システムのアプリケーションは，ミドルウェアが提供するデータアクセス関数を用いて，演算ノード間，及び演算ノードと機器間で授受するデータにアクセスして実現される。

4.2　ソフトウェアの再利用

ミドルウェア適用によって，複数機種を順次開発する際にソフトウェアの製作量が減る理由について，アプリケーションの機種間における再利用，及びミドルウェア化による再利用の観点から述べる。

4.2.1　アプリケーションの機種間での再利用

従来，機種ごとに変動する機種固有情報はアプリケーションが個別に保持していた。このため，異なる機種を開発する際，機種固有情報の変更が必要であり，関連するアプリケーションすべての修正が必要であった。

一方，ミドルウェアは，上記機種固有情報を一元管理し，アクセス方法を標準化する(**図2**)。この仕組みを利用して実装されたアプリケーションは，機種固有情報を持たない。このため，異なる機種を開発する際の機種固有情報の変更が，アプリケーションに及ばないため，アプリケーションが再利用可能となる。

4.2.2　ミドルウェア化による再利用

機器との伝送における冗長構成とその扱い方，演算ノードの冗長構成とその制御方法，編成の併合・分割時の処理方法は，機種間で共通に扱える機能である。ミドルウェアがこれらの機能を実現することによって，ミドルウェアそのものと同時に再利用される。

車両情報制御システムでは，監視制御対象を制御する系の切替え時や，編成状態が確定するまでの過渡的な状態であっても安定して動作することが求められる。これらの状況では，想定すべき状況が多岐にわたるため試験ケースが複雑になり，試験漏れの発生や，試験時間の増大が問題となる。

この機能をミドルウェアとして実現することによって，ミドルウェアに付随する試験ケースによって試験漏れの発生を抑制できる。また，これらの機能を新規実装する必要がなくなり，不具合混入の可能性が低減する。

5. ソフトウェア開発環境

車両情報制御システムの開発ワークフローは，システム設計，ソフトウェア製作，システム試験から構成される。ミドルウェア適用に合わせ，ソフトウェア製作段階に後述するツール群を開発し，適用できるようにした。これによって，開発の効率化と品質の確保が可能となる。

次に開発ワークフローに含まれる主な作業を示す。また，各作業で適用するツールもあわせて示す。

①機種固有情報の入力(ミドルウェアカスタマイズツール)
②入力データの確認(カスタマイズデータビューア)
③機種固有情報の生成(ミドルウェアカスタマイズツール)
④アプリケーション開発

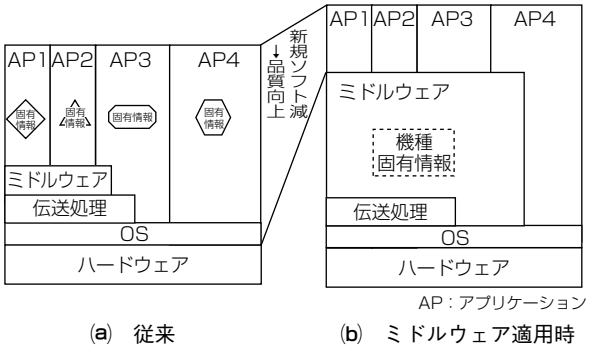


図2．ミドルウェア適用によるアプリケーション ソフトウェアの新規製作量削減

⑤デバッグ（編成シミュレータ）
⑥実装漏れのチェック（データ設定漏れチェックツール）
⑦システム試験（システム試験実行ツール）
次に，各ツールについて述べる。

5.1 ミドルウェアカスタマイズツール

ミドルウェアカスタマイズツールは，機器インタフェース仕様をデータ化し，機種固有情報の定義データを自動生成するツールである。

表形式による入力作業であり，ソフトウェアに詳しくない担当者でもデータ入力が可能である。また，入力データの一覧性を高め，入力間違いを防止する。

5.2 カスタマイズデータビューア

カスタマイズデータビューアは，ミドルウェアカスタマイズツールで入力したデータを，仕様書と同様のビット列表示で出力するツールである。ミドルウェアカスタマイズツールへの入力データと仕様書の内容を照合することによって，入力漏れやビットのずれなどの人為的ミスを低減できる。

5.3 編成シミュレータ

編成シミュレータは，汎用パソコン上で実機と同じアプリケーションを動作させ，デバッグを可能とするものである。

実機を用いてデバッグをする場合，処理タイミングが正確である一方，複数人での並行開発に限界があった。

アプリケーションは，複数の演算ノード上に搭載され，それらが連携して機能が実現されることが多い。このため，編成シミュレータは，編成に属する複数の演算ノードの動作を模擬する。

図3に編成シミュレータの構成を示す。車両情報制御システムのソフトウェアにミドルウェアを適用したので，演算ノードと機器間で共有されるデータへのアプリケーションによるアクセス方法が統一される。このため，シミュレータ用ミドルウェアを用意し，実機と同一のデータアクセス関数を提供することによって，汎用パソコン上で実機と同じアプリケーションを動作可能とした。

5.4 データ設定漏れチェックツール

アプリケーションは，ミドルウェア上に定義されたシンボルを用いて演算ノードと機器間で共有されるデータにアクセスする。データ設定漏れチェックツールは，シンボルを用いたデータ書き込み処理がソースコード上にあることを調査する。これによって，アプリケーションによるデータの設定漏れを検出し，人為的ミスを排除する。

5.5 システム試験実行ツール

車両情報システムが実現する機能の増大に伴い，試験項目数も増大しているため，試験操作時に混入する人為的ミスの削減と，試験実施の効率化の必要性が高まっている。

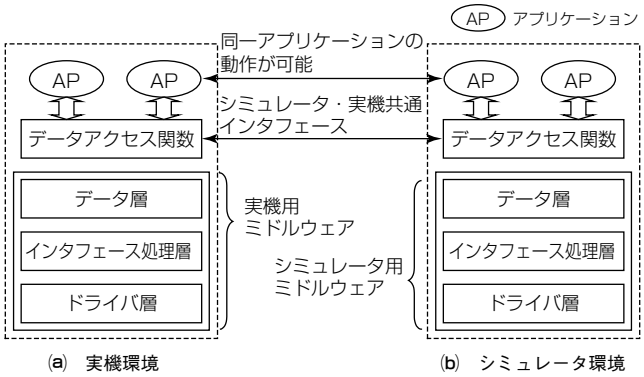


図3．編成シミュレータの構成

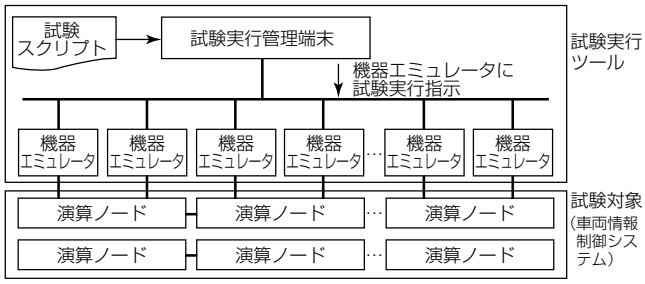


図4．システム試験実行ツールの構成

システム試験実行ツールは，与えられた試験スクリプトに従って，試験実行管理端末が機器エミュレータに対して動作を指示するツールである（**図4**）。

システム試験実行ツールは，車両情報制御システムが監視制御する機器の入出力を模擬する機器エミュレータと，それらの入出力動作を制御する試験実行管理端末から構成される。機器エミュレータに対する試験操作を，人手を介さずに実行できるため，試験操作における人為的ミスを排除するとともに，試験工数が削減される。

6. むすび

車両情報制御システムの機種間での共通性，変動性に基づき設計したミドルウェア，それを利用した開発ワークフロー，及びミドルウェアを核とした開発ツール群について述べた。

これらの総合的な取組みによって，ソフトウェアの生産性，品質の向上が達成できると考えられる。

参考文献

(1) 吉田　実，ほか：列車情報管理装置のソフトウェアプロダクトライン，三菱電機技報，**77**，No.7，479～482（2003）
(2) 小島泰三，ほか：戦略的再利用に基づくソフトウェア開発，三菱電機技報，**80**，No.10，647～650（2006）
(3) 辰巳尚吾：列車統合管理装置のソフトウェア開発技術，システム制御情報学会セミナー（2005）

特集論文

組み込み機器向けUI設計ツール“Edamame”

中川隆志*
岡本啓嗣**
小中裕喜***

User Interface Development Tool for Embedded Systems "Edamame"

Takashi Nakagawa, Hirotsugu Okamoto, Hiroki Konaka

要　旨

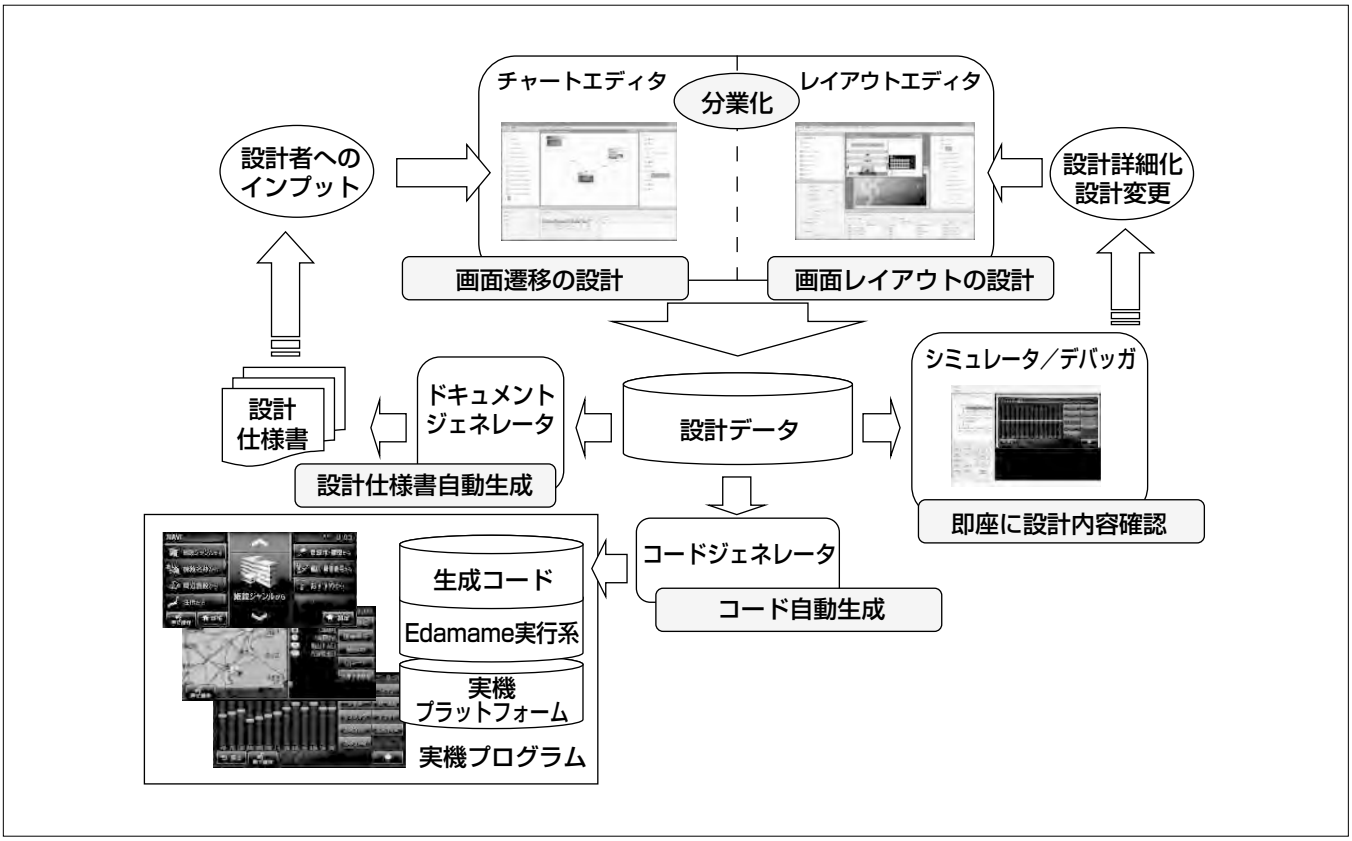
近年，カーナビや家電機器など様々な組み込み機器におけるUser Interface（UI）の高度化，開発期間の短縮に伴い，組み込みソフトウェア開発におけるUI開発の生産性向上が重要な課題となっている。三菱電機では，この課題解決をねらった開発環境として，UI設計ツール“Edamame（Embedded system Design Architecture with Model-based Approach and Middle-ware Environment）”の開発を行っている。

Edamameは，その前身である“NINA（Navigator for Interface of Application）”から，SCO（State Chart Object）というUI設計データの部品化による再利用を促進するための機構を引き継ぎ，さらに，年々肥大化するUI設計仕様に対応するため，分業によって同時並行的な開発を可能とする設計モデルへの拡張，及び連続的な動きや透明から徐々に浮かび上がるといった高度な表現機能を実現したものである。

Edamameによって大規模なUI設計の再利用性を高めるためには，UI設計データの部品化の粒度や，様々なスキルを持つ作業者間での分業範囲，成果物の受渡しの流れといった開発プロセスをどのように定めるかがポイントとなる。

現在Edamameはカーナビ開発に適用されているが，この開発における部品の再利用回数を測定したところ，部品の再利用が促進されていることが確認できた。

本稿では，UI設計ツールEdamameの機能の概要を述べるとともに，大規模なUI設計での設計モデルと設計の部品化の方針，及び開発プロセスの例について述べる。



UI設計ツールEdamameの構成とUI設計開発の流れ

UI設計ツールEdamameは複数の機能モジュールから構成される。レイアウトエディタではUI画面のレイアウトを設計し，チャートエディタではUI画面間の遷移やUI部品の制御を設計する。次にシミュレータ／デバッガによって，設計したUIの動作を確認し，修正が必要な場合にはエディタに戻り修正する。設計時には，ドキュメントジェネレータによる文書自動生成によって，最新の文書を閲覧することが可能である。コードジェネレータによって設計データから実機用ソースコードが自動生成され，Edamame実行系と結合することで，実機用のUIソフトウェアが完成する。

*先端技術総合研究所（エネルギー科学博士）　**同研究所　***三田製作所（工博）

1. まえがき

近年，カーナビや家電機器など様々な組み込み機器におけるUIの高度化，開発期間の短縮に伴い，組み込みソフトウェア開発におけるUI開発の生産性向上が重要な課題となっている。当社では，この課題解決をねらった開発環境として，UI設計ツールEdamameの開発を行っている(1)(2)。

Edamameは，その前身であるNINA(3)から，UI設計データの部品化による再利用を促進するための機構を引き継ぎ，さらに，年々肥大化するUI設計仕様に対応するため分業によって同時並行的な開発を可能とする設計モデルの拡張，及び連続的な動きや透明から徐々に浮かび上がるといった高度な表現機能を実現したものである。

本稿では，UI設計ツールEdamameの概要と，大規模なUI設計での設計モデルと開発プロセスの例，及びこれらの適用効果について述べる。

2. UI設計ツール Edamame

2.1 Edamameの特長

Edamameでは，設計データの再利用の促進及び，開発の効率化を実現するために次の特長を備えている(**扉図**)。

(1) 設計データの部品化

巨大なひとかたまりの設計データを，小さな独立した設計データのかたまり(部品)の組合せとして構築する。また，この部品をユーザーが自在に作ることを可能とする。

個々の部品が持つ機能や振る舞いを単純で明確なものにすることで，部品単位での再利用を図ることが容易となる。これらの部品は，チャートエディタ，レイアウトエディタによって構築される。

(2) モデル設計と設計文書の自動生成

部品はステートチャートに基づく図や，レイアウト図として設計する。このように設計データを図的に定義することで，その部品の振る舞いを，設計者本人以外でも容易に把握することができる。Edamameではドキュメント生成機能によって，設計データから設計仕様書のような設計文書を自動生成することができる。この機能は，設計者にとっては，従来の手書きによる文書作成の手間が省けるというメリットだけでなく，設計者本人以外にとって，実際の設計と完全に一致した設計文書がいつでも参照可能となり，部品の再利用に大きな効果を上げることができる。

(3) 設計即実行

シミュレータによって，設計データは部品単位で設計直後にその場で実行し動作を確認できる。これによって設計の誤りをいち早く発見することができる。Edamameでは，コンパイルやリンクといった従来のプログラミングで必要であったプロセスを経ることなく，即座にシミュレータで実行することが可能なため，設計－実装－デバッグといった繰り返しを効率よく実施できる。また，デバッガを使うと，あとで述べるSCO内のイベントハンドラを対象に，ブレイクポイントの配置やステップ実行，変数の参照・編集も可能である。

(4) コード生成とEdamameランタイム

Edamameでは，設計情報から，実機上で動作可能なコンパクトなソースコードを生成できる。生成したコードは，C++言語で記述されたEdamameランタイムライブラリと結合して，Edamame上のシミュレータの動作と完全に一致した挙動を再現できる。

このEdamameランタイムライブラリは，Windows(注1)，Linux(注2)など多様なプラットフォームに対応している。

(注1) Windowsは，Microsoft Corp.の登録商標である。
(注2) Linuxは，Linus Torvalds氏の登録商標である。

3. 設計モデルと開発プロセス

UI設計は，設計作業の面から見ると，画面デザインと制御ロジック設計の二つに大きく分類できるため，Edamameではこのような画面デザインと制御ロジック設計の成果物を，レイアウト部品とSCOの二つのタイプの部品として構築する。

次に，レイアウト部品，SCOのそれぞれの特性と，大規模なシステムにおける開発プロセスについて述べる。

なお，画面デザインの設計者を画面デザイナ，制御ロジックの設計者をプログラマと呼ぶ。

3.1 レイアウト部品

画面デザインを定義した部品をレイアウト部品という。レイアウト部品には，テキストやピクチャーといったEdamameにあらかじめ組み込まれている数種類の基本部品や，他のレイアウト部品，あとで述べるUI-SCOを配置することができる。**図1**にレイアウトエディタを使ったレイアウト部品の設計例を示す。

レイアウト部品は，部品サイズに応じた矩形(くけい)の上に，直接他の部品を配置することによって，その部品の見映えを直感的に，かつ，特別なプログラミングスキルを持つことなく定義することができる。

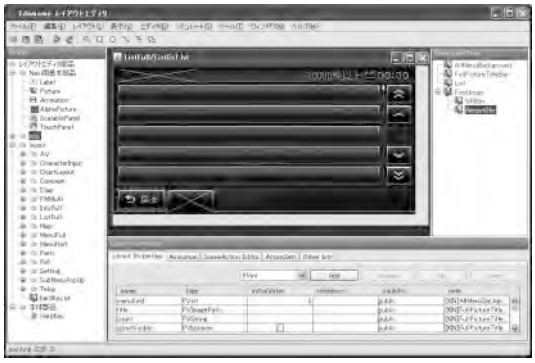


図1．レイアウトエディタの画面例

Edamameでは，頻繁に利用される定型的な表示効果を，プログラミングすることなく実現できる機構（アニメーション）を設けている。レイアウト部品には，外部から設定可能なプロパティを設けることができるが，このプロパティ値に応じて，そのレイアウト内部に配置されている部品を指定の位置に移動させたり，色を変えたり，変形させたり，透過度を変えるといった，プロパティ値とその値における制御内容をアニメーションとして複数定義可能である。

さらに，プロパティ値を時々刻々変えるタイムライン定義が可能である。このアニメーションとタイムライン定義を組み合わせることによって，画面の下部から上りながらスライドインしてくるメニューや，徐々に透明になって消えていくボタンなどの高度な表現を，レイアウト部品単体で実現可能である。

3.2 SCO

複数の表示状態と，それらの間の遷移を設計可能なUI部品である，UI-SCOの概念を**図2**の上部に示す。

UI-SCOは，ユーザーが定義可能な動きのある表示部品であり，複数の表示状態を持つことができる。例えば，**図2**で，UIScol1は，選曲，再生の二つの表示状態を持っている。選曲，再生のそれぞれの表示状態には，Layout3やLayout4といったレイアウト部品を保持している。UIScol1は，ユーザーの操作やシステム内部の状態の変化などのイベントを受け，レイアウト部品を切り替えることによって，部品の見え方を制御している。このようなイベントを受けて表示状態を切り替えるものをイベントハンドラといい，このイベントハンドラには，簡単なスクリプト（プログラム）を定義することも可能である。SCOを定義するためには，チャートエディタを用いる（**図3**）。

一方，UI-SCOと異なり，各状態における表示部品を保持せず，SCO全体で一つのレイアウト部品の制御を行う制御部品としてControl-SCOがある。**図2**の下部にControl-SCOの概念図を示す。Control-SCOは，先のUI-SCOと同様にイベントハンドラを保持できることから，イベントを受けて，制御対象であるレイアウト部品に対して様々な表示制御を行うことが可能となる。**図2**におけるControlSco1は，Layout1に対する制御を行っており，停止中，再生中のそれぞれの状態が変わる際に，イベントハンドラに定義されたLayout1の画像の変更や文字列の変更といった制御を行う。

SCOやレイアウト部品の特長は，設計者が作成した部品を，他の部品に張り込むことができる。例えば，**図2**では，Layout1がLayout4の一部に張り込まれている。このように，画面中の小さな部品から，より大きな表示領域を持つ部品，アプリケーション全体にまで，様々な階層の設計を部品化し，組み合わせることを可能としている。また，複数の画面に配置した部品の機能拡張や障害修正を行うと，その効果はすべての配置先に水平展開される。

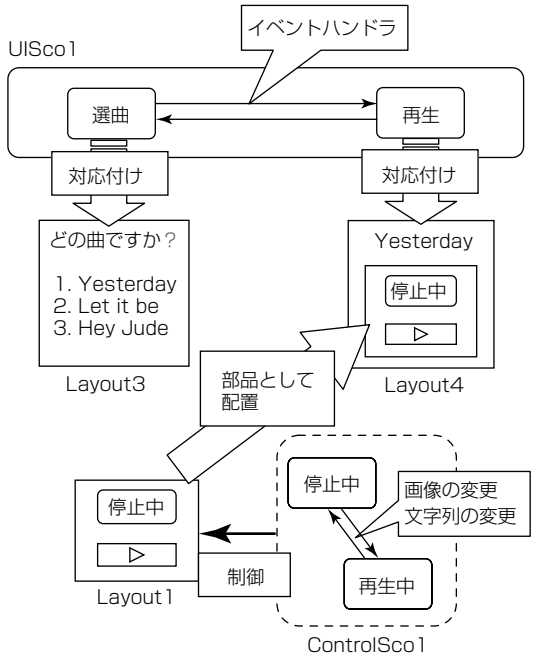


図2．モデル構造の例

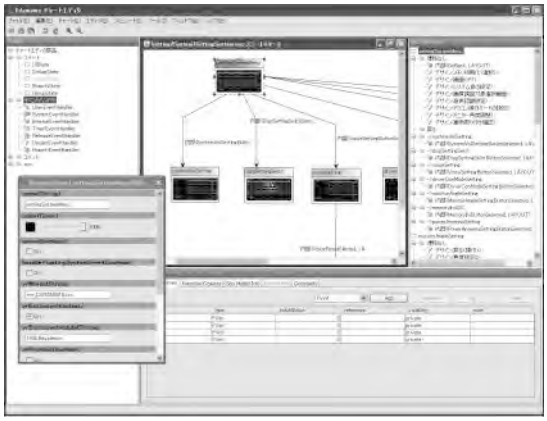


図3．チャートエディタの画面例

3.3 分業のための開発プロセス

UI設計の再利用性を高めるために，UI部品として適切な粒度を定めることと，それぞれの出現頻度に応じた作業担当を定める必要がある。まず，UI部品の粒度に応じて次の4種類の分類を行った。

(1) 小部品

ボタンのように基本的な機能を提供するUI部品。アプリケーション全体で頻繁に出現する。

(2) 中部品

ボタンが縦や横に並ぶリストのような，アプリケーション中のある範囲で共通的に利用されるUI部品。小部品ほどではないものの，しばしば出現する。

(3) 画面部品

待ち受け画面，メニュー画面といった，アプリケーションレベルでの画面を表すUI部品。出現頻度は低い。

⑷ アプリケーション

アプリケーション全体における画面フロー。画面部品がどのような順序で表示されるかを設計する。ナビや，オーディオ，設定といった単位で作成される。

このように粒度から分類されたUI部品を，次のように画面デザイナとプログラマは分担して設計を行う。ここでは**図4**の例に基づき，開発プロセスについて述べる。

⑴ 第1段階

小部品であるボタン部品は，レイアウト部品としては画像とテキストなど数点の部品を保持する単純な部品であるが，これを制御するControl-SCOは，配置箇所に応じて多くの表示バリエーションを実現するなどの汎用(はんよう)性を考慮した設計が必要である。そのため，小部品については，スキルの高いプログラマが開発初期に集中して担当することが望ましい。この段階で，部品単位の挙動をシミュレータで確認する。

⑵ 第2段階

第1段階の成果を利用して，中部品，画面部品を画面デザイナが作成する。中部品，画面部品になると，画面の数に比例して大量に必要になることや，仕様が頻繁に変更されることを踏まえ，あまり汎用性を重視した高度な部品を設計するのではなく，各画面の表示内容に対応したレイアウト部品を量産することが求められる。また，定型的な表示効果については，アニメーションとタイムライン定義を組み合わせて実現する。

⑶ 第3段階

第2段階の成果をプログラマが引き受け，それぞれのレイアウト部品に対応した制御ロジックを，Control-SCOとして定義してゆく。この段階で，画面内の挙動をシミュレータで再現することが可能となる。

⑷ 第4段階

第3段階で作成された画面を，プログラマがUI-SCOの画面間遷移として取り込む。このとき，画面間の遷移時に必要な様々なイベントハンドラの定義も一緒に行い，このUI-SCOをアプリケーションとして作り上げていく。

3.4 Edamame適用による効果

Edamameを大規模なUI設計を必要とするカーナビの開発に複数機種にわたって適用し，再利用性，開発効率について評価した。

この結果，平均1部品当たり3.7回，また，ボタン部品のような汎用性の高い小部品については，100回以上再利用が行われていた。この再利用の回数は各機種でもおおむね同じであった。比較的小さな部品を中心に，効率良く再利用されていることがわかる。

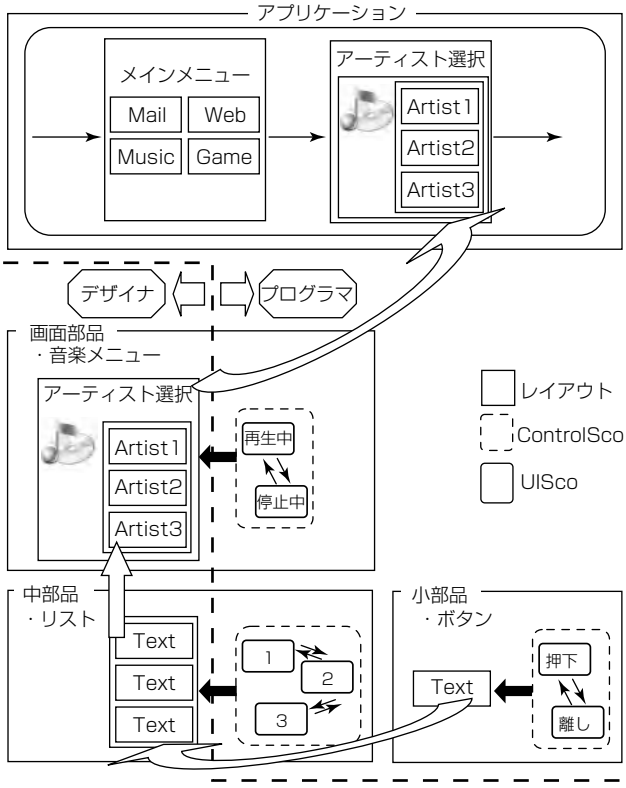


図4．部品の分割と役割分担

さらに，従来のソースコードによる開発と比べると，シミュレータを活用した問題の早期発見による手戻り防止効果も確認できた。

4. むすび

UI設計ツールEdamameの概要と，大規模なUI設計での設計モデルと開発プロセスの例について述べた。そして，カーナビのUI開発にこれらを適用したところ，小さな部品を中心に，効率良くUI設計が再利用されていることを確認した。今後は，カーナビに加え，他のUI設計が高度化するような組み込み機器へ展開を図る計画である。

参考文献

⑴ 岡本啓嗣，ほか：スキルに応じて作業分担可能なUI設計ツール，電気学会全国大会(3)，154～155（2007）
⑵ 岡本啓嗣，ほか：UI設計ツールを用いた開発における効率的運用，情報処理学会FIT2008 第7回情報科学技術フォーラム（2008）
⑶ 小中裕喜，ほか：階層的部品定義に基づく組み込み用UI設計ツール，組み込みソフトウェア工学シンポジウム2002，情報処理学会研究報告，2002－SE－139，7～8（2002）

特集論文

空調機器制御ソフトウェアの再利用開発

大河原　繁*
白川智也*
長峯　基**

Reuse Framework for Air-conditioner Software Development

Shigeru Okawara, Tomoya Shirakawa, Motoi Nagamine

要　旨

近年，空調機器の製品機能の多様化と制御方式の高度化に伴い，空調機器を制御するための制御ソフトウェアは複雑化してきている。空調機器の制御方式は，顧客ニーズへの対応や性能向上など高度化する要求にこたえるため，既存の制御方式を継承しながら改良が重ねられてきた。空調機を制御するソフトウェアの開発は，既存製品のソースコードを流用して機能を追加・変更する方法で行われており，高度化する要求にこたえるためには，流用の繰り返しによるソフトウェア構造の複雑化を防止し，効率的かつ高品質な開発の実現が課題となっている。

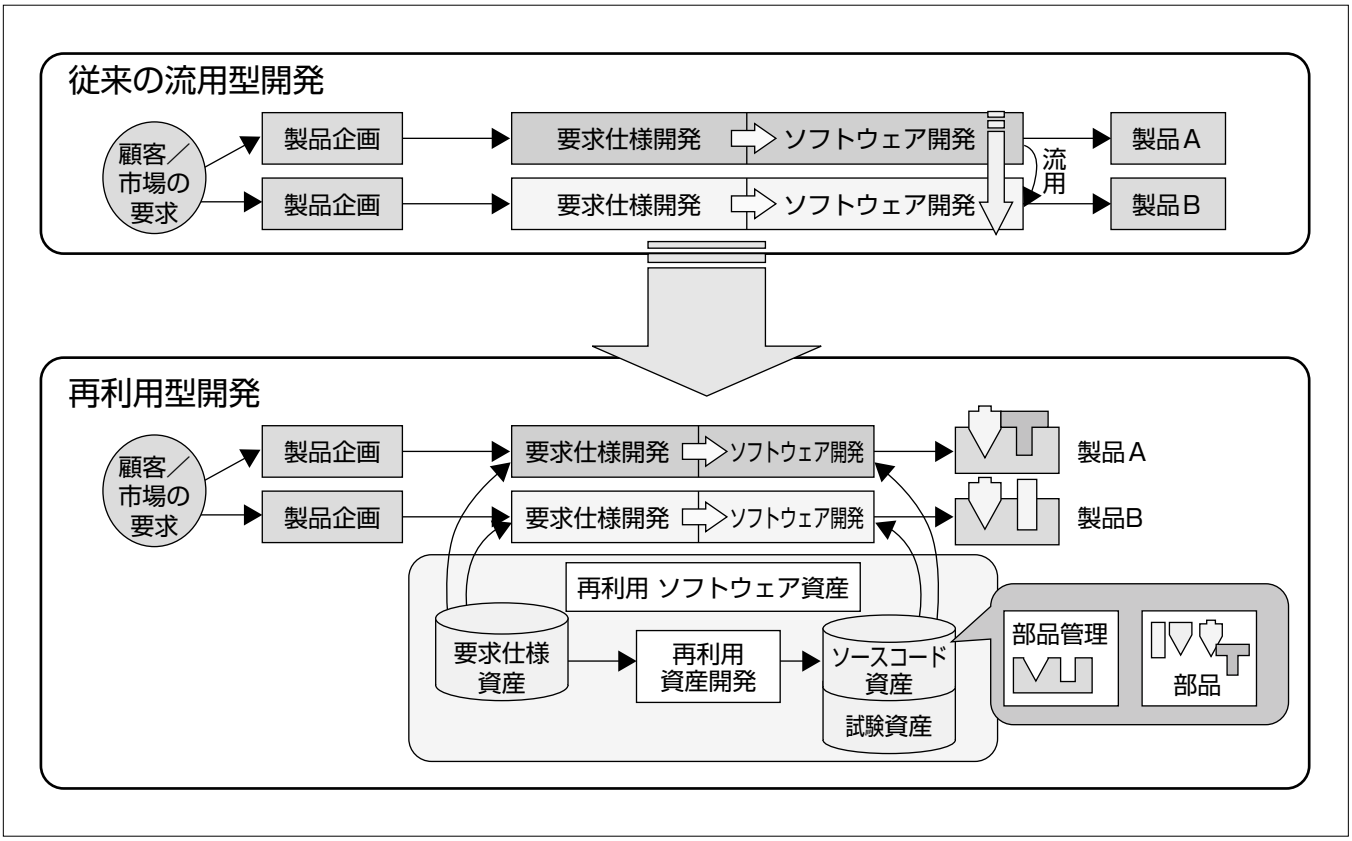
本稿では，空調機器の室外機に搭載される制御ソフトウェアを対象とした再利用型開発実現の取組みについて述べる。従来型開発における課題に対し，次のような取組みを行うことで，ソフトウェア部品化に基づく再利用への基盤整備を進めた。

(1)　要求仕様書で機能分類などの論理構造を規定することによって，機能分割する方法を標準化した。

(2)　ソフトウェアの基本構造を要求仕様書と対応させるとともに，機能部品の作成方法を標準化した。

(3)　機能部品の試験設計方法の標準化，及び試験実施の自動化を実現するとともに，要求仕様書に対応した再利用可能な試験資産を構築した。

これらの取組みの結果，生産性の向上，試験の網羅度の向上といった改善効果を得ることができた。



従来の流用型開発と再利用型開発

従来行われていた流用型開発では，新しい製品（製品B）を開発する際に，既存の製品（製品A）のソースコードを流用・改造していた。再利用型開発では，機種間で共通に使うことができる再利用ソフトウェア資産を用意しておき，製品Aも製品Bもこれらの資産を組み合わせて開発される。再利用ソフトウェア資産には要求仕様資産，ソースコード資産及び試験資産が含まれる。

*設計システム技術センター　**静岡製作所

1. まえがき

近年，空調機器の製品機能の多様化と制御方式の高度化に伴い，空調機器を制御するための制御ソフトウェアは複雑化してきている。空調機器の制御方式は，顧客ニーズへのタイムリーな対応や性能向上など高度化する要求にこたえるため，従来の製品で実績がある制御方式を継承しながら改良が重ねられてきた。この結果，過去10年間で制御機能項目数は約2倍に増加し，制御ソフトウェアの規模増大や複雑化が進んできた。

こうした中で，高い品質を維持しながらより効率的に制御ソフトウェアを開発することが求められており，その手段としてソフトウェア再利用の重要性が増してきている。本稿では，空調機器の室外機に搭載される制御ソフトウェアを対象とした再利用型開発実現の取組みについて述べる。

2. 空調機器制御ソフトウェア

2.1 制御ソフトウェアの位置付け

今回開発の対象とした空調機器制御ソフトウェアの位置付けを**図1**に示す。空調機器の室外機には圧縮機，電子膨張弁，ファン，四方弁といったアクチュエータがあり，コントローラからこれらを制御することで，冷房や暖房といった機能を実現する。コントローラに搭載されたマイコンが，ROMに格納された制御ソフトウェアを実行することで制御を行う。

2.2 制御ソフトウェア開発における課題

従来，制御ソフトウェアの開発は既存製品のソースコードを流用して機能を追加・変更する方法で行われてきた。機能を追加・変更する際には，ソースコードを詳細に調査して追加・変更箇所や変更影響箇所の特定を行っており，ソースコードの大規模化・複雑化に伴い，これら作業の効率化が課題となっていた。

このようなソフトウェア開発へのニーズに対して，要求仕様書とソフトウェア共通の機能分類を規定して各機能の実現箇所を明確化すること，機能間の独立性を高めて新規機能を容易に実装できると同時に従来と同じ機能は再利用できることなどが課題であった。

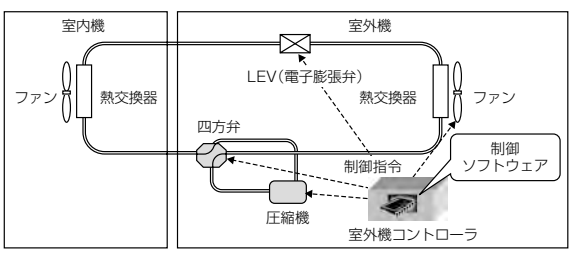


図1．制御ソフトウェアの位置付け

3. 施　策

2.2節で述べた課題解決のための施策として，要求仕様からシステム試験まで一貫した部品化による再利用型開発環境を確立した。その流れを**図2**に示す。次に，各施策について詳細に述べる。

3.1 分　析

空調機器制御ソフトウェアの要求仕様を分析して，空調機器制御の論理構造を抽出した。その結果，空調機器制御の構成要素が大きく①状態遷移，②基本制御，③補正制御に分類されることが分かった。この分析結果を基に，ソフトウェア部品化に基づく再利用型開発環境の確立に取り組んだ。

3.2 要求定義

ソフトウェア部品化に基づく再利用型開発環境の確立のためには，ソフトウェア作成のための入力情報になる要求仕様書で，部品化に対応した機能分割やデータ定義をする必要がある。しかしながら，従来の要求仕様書は，機能分割ルールが規定されておらず，また機能間で使用するデータは未定義なものも存在した。そのため，部品化に対応した機能分割に基づく仕様記述ができていなかった。そこで，空調機器制御ソフトウェアの要求仕様の分析結果を基に，要求仕様書を体系的に機能分割やデータ定義可能な章構成に変更し，その章構成で要求仕様を記述するためのテンプレートを作成した(**図3**)。

図3のテンプレートに従い要求仕様書を記述することで，ソフトウェア部品化と対応する要求仕様書を実現するとともに，定形化した記述をすることでモレ・ムラがない仕様記述を可能にした(**図4**)。

3.3 設計／実装

長期間保守しても複雑化などの劣化が生じにくいソフト

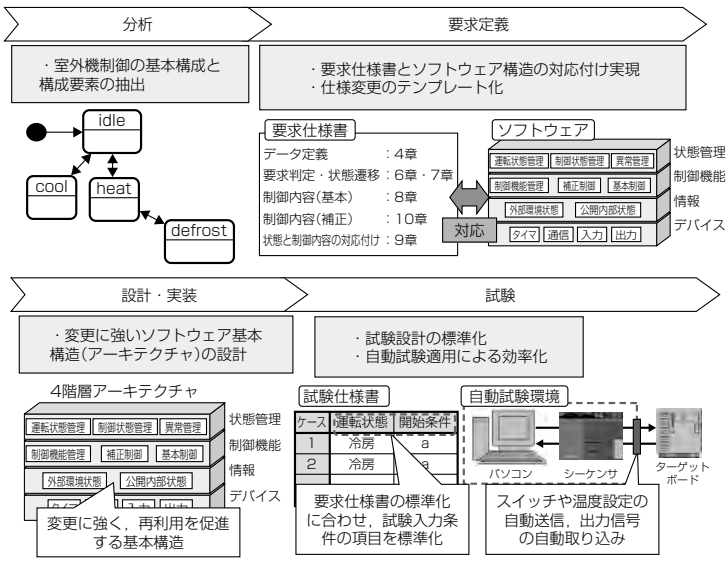


図2．再利用型開発環境の確立の流れ

ウェア構造を実現するために，4階層からなるソフトウェア・アーキテクチャを定義した(**図5**)。従来のソフトウェアは階層化が明確でなかったため，機能追加や変更を繰り返した結果，機能間の相互依存が存在するなど構造が複雑化していた。階層間は定義されたインタフェースを通じて参照するようソフトウェア部品の設計・実装ルールを規定することで，無秩序な参照を防止しソフトウェアの複雑化を抑制した。

制御機能は，ソフトウェア部品と部品管理部に分けることで，ソフトウェア部品を動的に組み合わせて空調機器としての制御機能を実現できるようにした。ソフトウェア部品は，冷房起動制御や高圧保護といった空調機器制御における一つの機能を実装したものである。部品管理部は，各ソフトウェア部品からの出力値を統合する役割を持つ(**図6**)。部品登録テーブルには，製品を構成するために必要なすべてのソフトウェア部品が登録される。部品管理部は，部品登録テーブルに登録されたソフトウェア部品を呼び出して実行し，各部品の出力を統合して，空調機器全体としての出力値を決定する。

各部品の単位は，空調機器全体としての制御の状態(冷房中，暖房中など)と，圧縮機や膨張弁などのアクチュエータの状態(起動中，停止中など)の組合せに対応する一つの制御機能(冷房起動制御等)として規定した。制御機能を部品化する粒度を統一することによって，開発者による部品化のばらつきをなくし，ソフトウェア部品単位での再利用を容易にした(**図7**)。

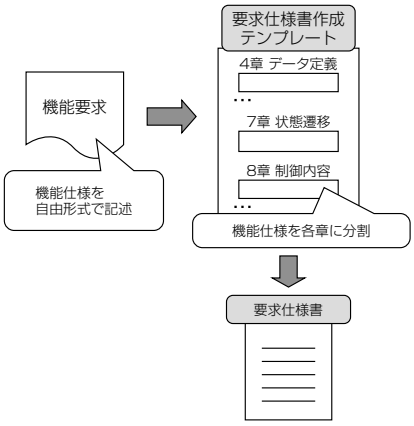


図3．要求仕様書作成テンプレートのイメージ

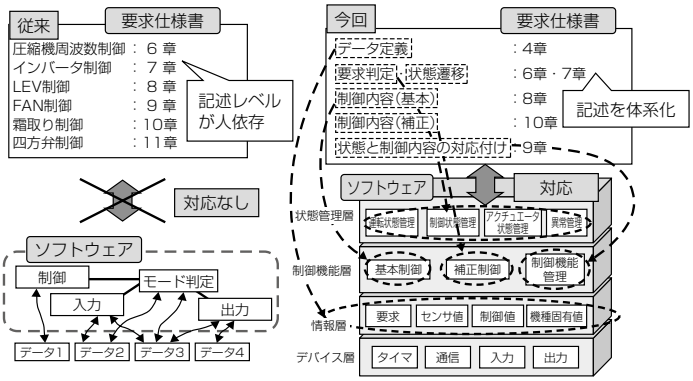


図4．要求仕様書とソフトウェアとのトレーサビリティ

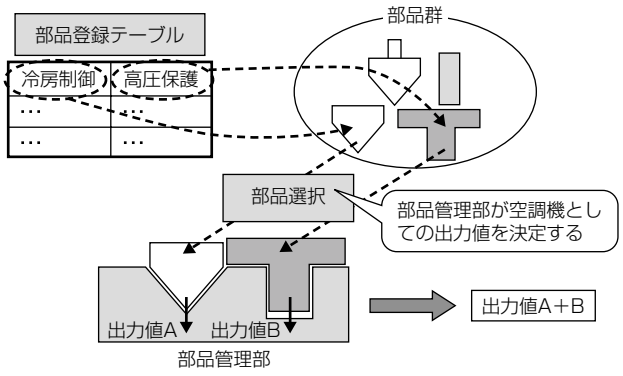


図6．制御部品管理部と部品の連携

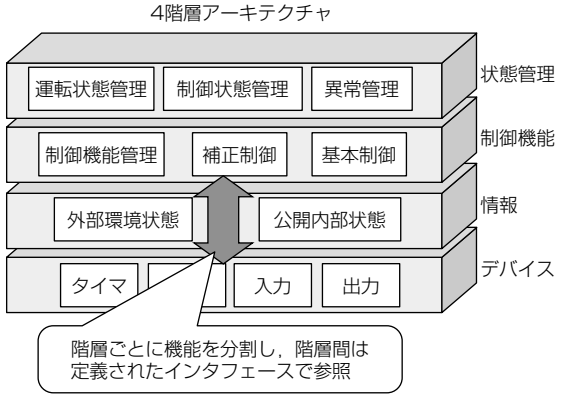


図5．ソフトウェア・アーキテクチャの定義

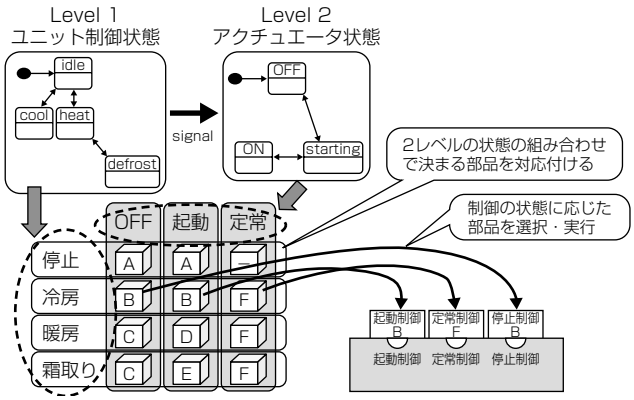


図7．部品化単位の標準化

3.4 試　　験

3.4.1 試験設計の標準化

従来のシステム試験は，要求仕様書から抽出する試験入力条件が試験設計者ごとにばらついていた。そこで今回は，標準化された要求仕様書の記載項目をそのまま試験入力条件として抽出できるようパターン化し，試験入力条件抽出のばらつきを削減した(**図8**)。

また従来は，入力条件を組み合わせて試験ケースを作成する際に，試験設計者の経験と勘で入力条件を組み合わせていたため，組合せの漏れが存在していた(二つの入力条

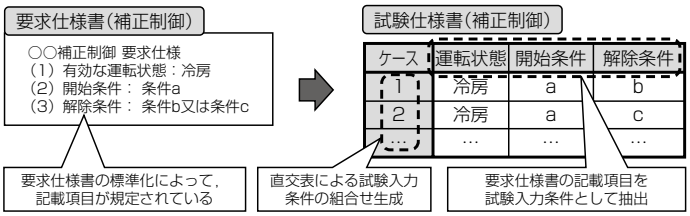


図 8．試験設計の標準化

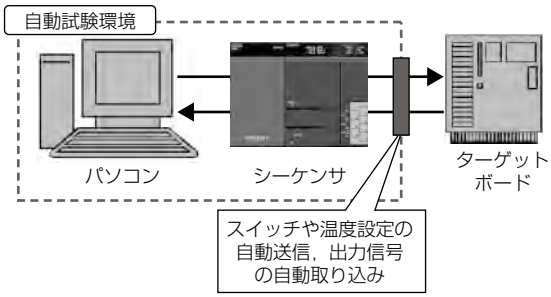


図 9．試験実施の自動化

図10．試験シナリオのイメージ

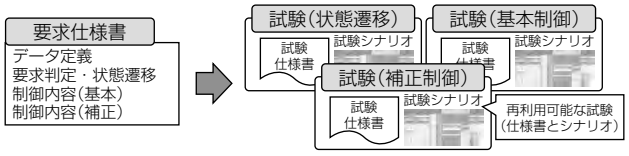


図11．試験資産の再利用

件の組合せ網羅度65%)。そこで今回は，直交表を活用して入力条件の組合せを作成することで，組合せの漏れを削減した。直交表は品質工学に基づいて考案された表であり，試験設計に直交表を活用することで，二つの入力条件の組合せ網羅度を100%保証することができる。また，専用ツールを使用することで，効率的に入力条件の組合せを作成した(**図 8**)。

3. 4. 2　試験実施の自動化

部品化による再利用型開発では，システム試験で，部品の組合せごとの繰り返し検証が必要になる。しかし，従来は人手による試験実施のため，繰り返し検証をするのに膨大な工数が掛かり非効率であった。そこで今回は，具体的な試験手順や試験データを記載した試験シナリオを再利用し，システム試験実施を自動化する環境を構築した(**図 9**)。

図 9 の自動試験環境は，次の流れで試験実施する。

(1)　**図10**に示す試験シナリオを入力とし，パソコンに接続されたシーケンサで空調機器のセンサやアクチュエータの模擬信号を作り，これをソフトウェアが搭載された制御ボードに送信する。

(2)　自動試験環境のパソコンで試験結果のデータを自動収集し，リアルタイムで期待値と一致するか判定する。

(3)　試験結果と期待値との判定結果を記載した試験成績書を自動生成する。

3. 4. 3　試験資産の再利用

図 8 に示した試験設計の標準化によって試験仕様書を作成し，その試験仕様書を基に**図10**に示した試験シナリオを作成することで，要求仕様書に対応した再利用可能な試験資産を構築した(**図11**)。

4. 効　　果

再利用型開発環境を空調機制御ソフトウェアの製品開発に適用することで，次の効果が得られた。

(1)　要求定義からシステム試験までの生産性を40%向上

(2)　ソースコード全体の複雑度を35%削減

(3)　システム試験における二つの試験入力条件の組合せ網羅度を100%実現

5. む　す　び

要求仕様から試験まで一貫した部品化による再利用型開発環境を構築することができた。これによって，従来は要求仕様から試験まで属人的に実施されていた機能分割を標準化することができた。また，見える化されていなかったソフトウェア構造を，部品化をベースとした整理されたソフトウェア構造に再構築することで，見える化を実現した。これまでに述べた取組みの他の事業分野への展開を進めている。