昭和37年9月15日第三種郵便物認可　三菱電機技報　77巻7号(通巻866号)　2003年7月25日発行(毎月1回25日発行)

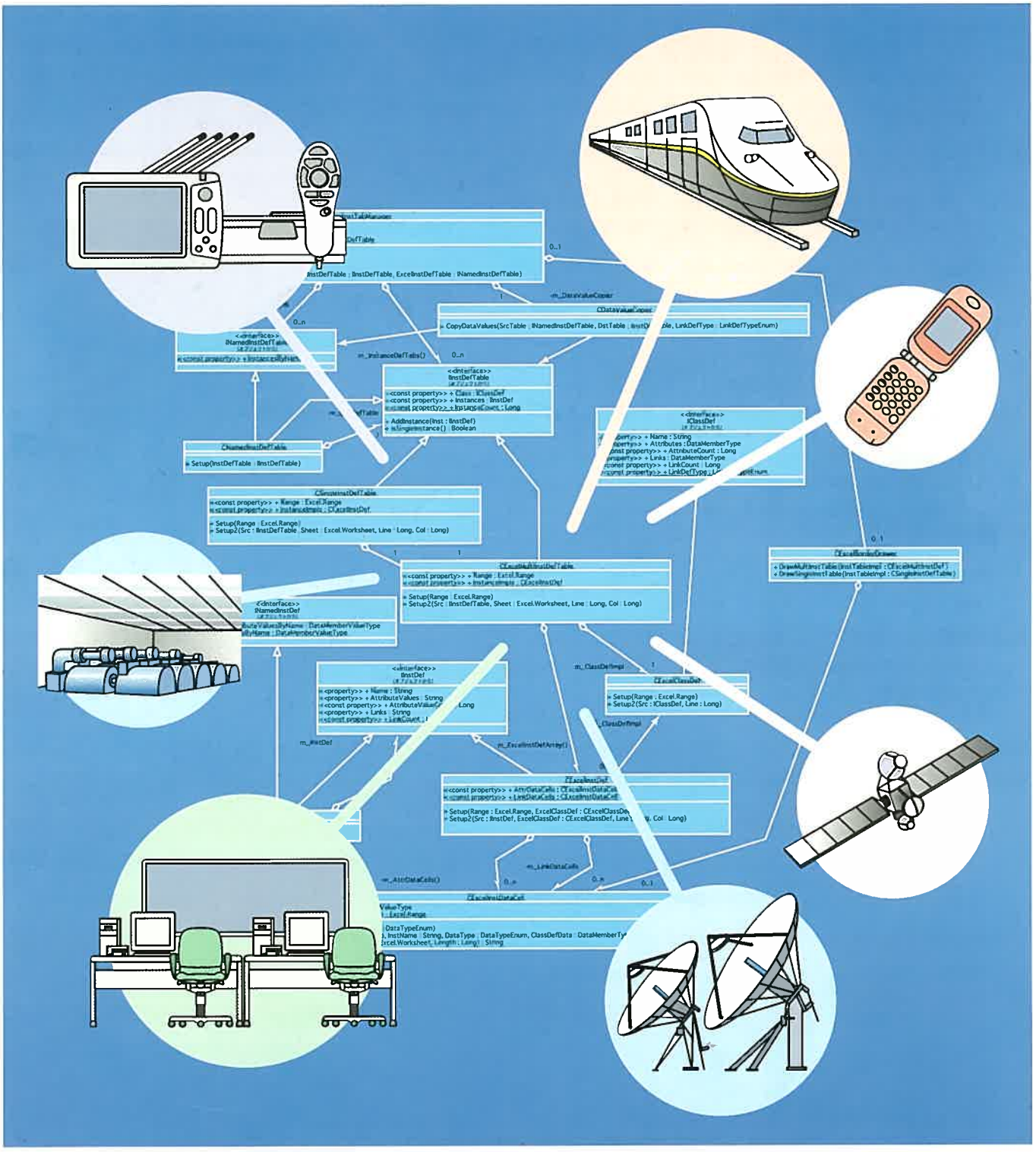
Changes for the Better

MITSUBISHI

三菱電機技報 Vol.77 No.7

特集「ソフトウェア開発プロセス・手法の革新」

2003 7



ISSN 0369-2302 CODEN:MTDNAF

三菱電機技報 Vol. 77 No.7 2003

目　次

特集「ソフトウェア開発プロセス・手法の革新」

特許と新案

スポットライト

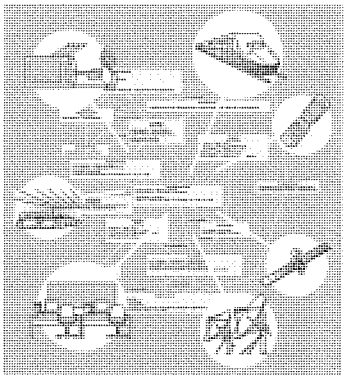
汎用熱・流体解析ソフトウェア"MelTHERFY""TherfBENCH"

表紙

ソフトウェア技術による製品群の価値の創生

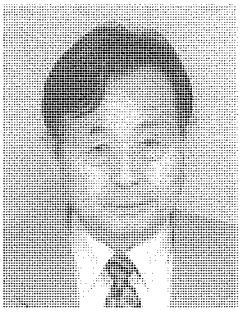
IT(Information Technology)技術の向上，また，その経済社会への浸透に伴い，社会インフラの整備，企業の経営革新，また個人の多様なライフスタイル実現など，ソフトウェアの比重はますます高くなり，付加価値創生の源泉がハードウェアからソフトウェアに移りつつある。

三菱電機では，人工衛星，監視制御，鉄道，携帯電話，カーナビ等の各分野に対し，ソフトウェア開発におけるプロセス及び手法の革新により，生産性と品質の向上を実現し，製品の付加価値を高め，お客様に満足いただく製品作りに取り組んでいる。



巻/頭/言

ソフトウェア開発力は企業技術力の重要ファクタ



東京電機大学理工学部

教授　小泉寿男

ソフトウェア開発力は，いまや企業技術力を示す重要ファクタになってきている。ソフトウェアは，情報システム構築目的やパッケージソフト開発にとどまらず，産業機器，情報家電等のほとんどすべての製品に搭載され，その量はますます増大し，製品機能，品質，開発期間と市場投入タイミングを左右して，製品競合力を握るようになった。

ソフトウェア開発力と組織の関係については，組織のソフトウェア開発能力評価モデルのCMM(Capability Maturity Model)によく表れている。CMMは，組織がソフトウェア開発に必要な能力のレベルを成熟度として表したモデルであり，300以上に及ぶ評価観点の得点を基に，評価認定機関がレベル１からレベル５の５段階で評価する。

CMMは，ソフトウェア開発組織を評価の対象としているが，ソフトウェア開発力を企業の技術力を継続的に発展させる重要ファクタであり重要技術基盤の一つであると考え，組織的アプローチの充実を図るべきであろう。この理由は次のようである。

(1)　ソフトウェア開発力は一人一人の技術力と組織力の結合

良いソフトウェアの開発には，既存技術を分析してどこにどのような最新技術を取り入れていくかが基本であり，このためには，一人一人の技術力と組織的な企画，行動の結合が不可欠となる。ソフトウェア開発生産性上最も効果的な再利用技術においても，再利用可能なソフトウェアアーキテクチャ設計と再利用対象コンポーネントの蓄積・更新は一人一人の技術力と組織力を必要とするからである。近年，中国など海外へのソフトウェア開発委託においても，委託側の技術力と組織的方策がその成否を決めている。

(2)　開発プロセス改善の継続的フィードバックループ

どのような製品開発においても，プロセスの改善の成否は，目標の設定，課題・改善事項の抽出，実行結果の分析とフィードバックの繰り返しによって決まる。ソフトウェア開発力においても，目標設定と継続的フィードバックループの充実には，企業レベルの推進，支援を必要とするからである。

(3)　技術力の目標設定

一人一人の技術力を向上し，組織力とするためには，ソフトウェア開発における現状の組織技術力の分析，到達すべき目標の設定が不可欠であり，その施策には企業レベルの視点が必要とされるからである。

以上のようにソフトウェア開発力を企業技術力の重要ファクタと見てその向上を図るには，まず一番目は，“技術者の相互連携と組織的活動のビジブル化”である。組織の技術力は，一人一人の技術力アップが原点であり，組織支援による相互研鑽(けんさん)で更に向上する。この組織技術力と実際の開発プロジェクト実績との相互評価は有効であろう。

二番目は，“技術者教育”であり，上記のような目標設定に基づく施策が重要となる。

三番目は，これらの活動が“企業レベルの適切なステアリング”により，改善の施策が常にとられることである。

この特集号の発行が，ソフトウェア開発プロセス・手法の革新を促進し，さらに，ソフトウェア開発力を考察する機会になることを期待したい。

巻頭論文

ソフトウェア開発プロセス・手法の革新への取り組み



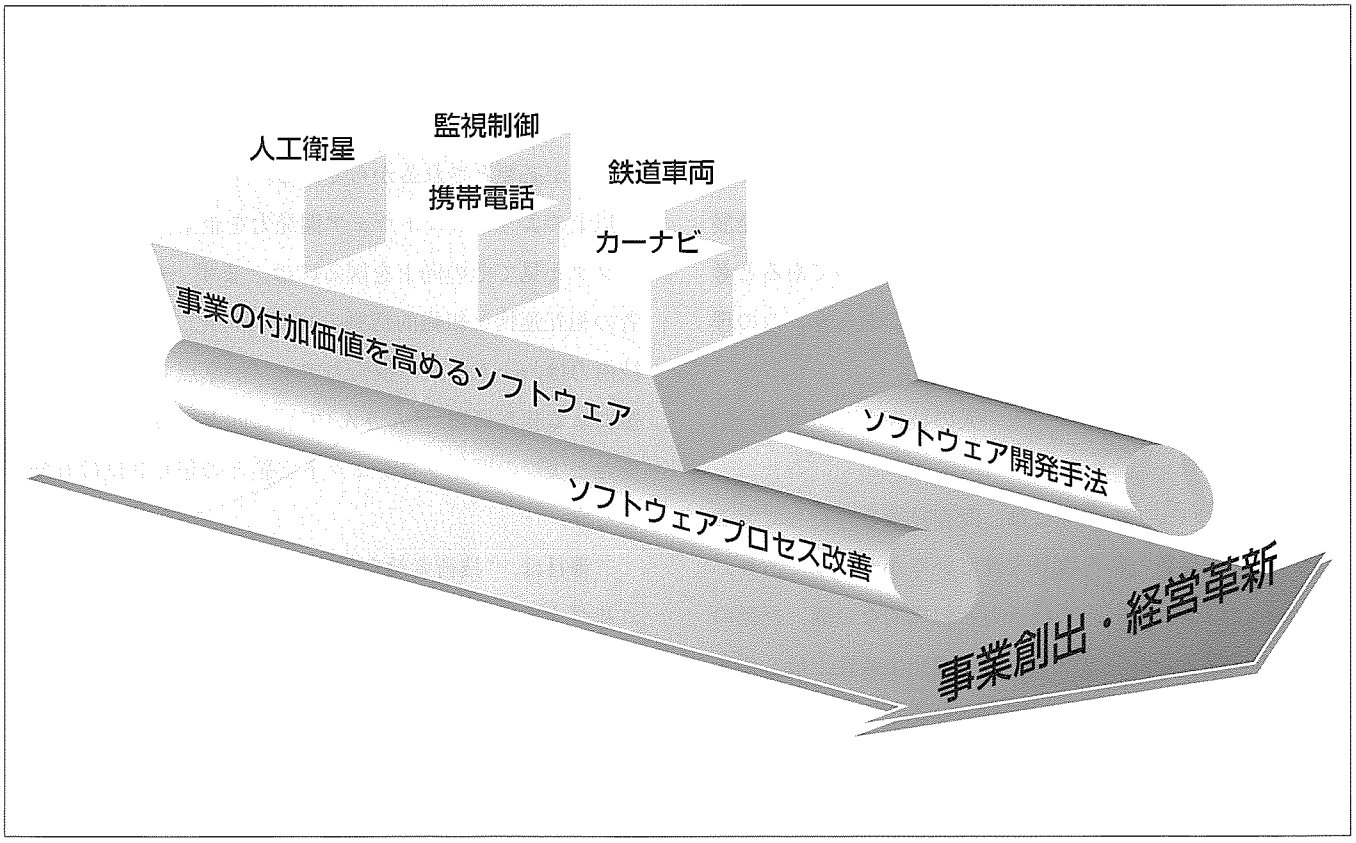
木村廣隆* 坂井英明** 堀池 聡***

要 旨

IT(Information Technology)技術の向上，また，その経済社会への浸透に伴い，社会インフラの整備，企業の経営革新，また個人の多様なライフスタイル実現など，あらゆる分野においてソフトウェアの比重はますます高くなり，付加価値創生の源泉がハードウェアからソフトウェアに移りつつある。その結果，ソフトウェアの規模は増大・複雑化し，短期間に多くの人数を投入する開発スタイルをとらざるを得ない状況が起きている。このような状況への対応には，ソフトウェアプロセスと開発手法の両面から，生産性と品質の向上に取り組むことが不可欠である。

三菱電機において，プロセス面では，これまでの問題解決型のアプローチだけではなく，本来あるべきソフトウェア組織の姿を定義し，体系的なプロセス改善に向けた活動を実施している。一方，ソフトウェアをより効率良く実現するには，既に作られたソフトウェアを再利用する技術が不可欠である。ただし，単なるソフトウェア部品の再利用ではシステム開発を効率的に進めることはできない。戦略的に再利用を進める仕掛けが必要である。ソフトウェア開発手法としては，分野対応のフレームワークを中心とする再利用技術に注力している。

当社では，ビジネス分野，さらには携帯電話，カーナビや多くの監視制御等のシステム製品に対し，ソフトウェアプロセスと開発手法両面のアプローチをとり，ソフトウェア開発の強化に取り組んでいる。



ソフトウェア開発のイメージ

ソフトウェアプロセス改善，手法の革新の両輪で，各事業の中核である"ソフトウェア自体"を高品質化，さらには生産性を高め，事業創出，企業の経営革新等に寄与している。

*設計システム技術センター長 **同センター ソフトウェア応用技術部 次長 ***先端技術総合研究所 システム構築技術部 主席技師長(工博)

1. まえがき

経済社会，また個人生活の隅々まで，IT技術が浸透し，ソフトウェアが提供する利便性や信頼性などその品質の高さは不可欠となり，また，ソフトウェアの規模は増大・複雑化しつつある。

近年，欧米だけでなく，インド，中国などのアジア諸国においても，IT技術に直結したサービス産業を自国の戦略産業と位置付け，プロセス改善活動の推進を国家規模で実施している。さらに，調達サイドの動きも大きく，既に欧米においてはプロセス改善を積極的に実施している企業からの調達を優先し，また，日本国政府においても欧米と同様に調達先のプロセス改善を評価する動きにある。

一方，開発手法の面では，ソフトウェア再利用を戦略的に進める仕掛けが必要であり，再利用の機構が作り込まれた分野対応別のフレームワークが望まれる。

2. プロセス・手法革新の背景，課題

あらゆる分野においてソフトウェアの比重が高くなるにつれ，ソフトウェア品質に問題が生じたときの社会的影響は大きく，品質向上は最優先課題である。また，製品に関する価格競争が激化しており，ソフトウェアを適切な開発コストで実現する技術が求められている。ソフトウェア開発では，コード生成だけではなく，設計，ドキュメント製作やソフトウェア試験にも多くのコストを費やしており，これらに要するコスト低減も重要なテーマとなっている。

一方，ソフトウェア領域における急激な技術進歩を製品に反映するために，市場への新機種投入サイクルが短くなってきている。それに伴い，ソフトウェアをより短期間で開発する技術が必要とされている。製品によっては，開発期間の遅延により市場投入へのタイミングを逸してしまい，その開発が無意味になりかねない状況が起こり得る。これらの課題は相互に関連しており，従来からソフトウェア開発に伴う課題ではあるが，近年ますますその深刻さを増している。

3. プロセス改善への取り組み

3.1 ソフトウェアプロセス改善の取り組み内容

従来，品質上・工程上の問題を解決するためには，工程内，また市場における問題に対して，問題の発生原因を突き止め，それを解決する施策を立案して実行するといった現場解決型の帰納的アプローチを実施してきた。このアプローチは個人の持つノウハウを組織のノウハウとして構築することができるとともに特定の問題の解決には極めて有効であるが，工期／工数の大幅な削減といった抜本的な改善については必ずしも有効ではなく，当社では，SW-CMM(注1)[1]などのソフトウェアプロセス改善手法が活用するベストプラクティスモデルを演繹(えんえき)する取り組み方法も併せて適用開始している。

ソフトウェアプロセス改善とは，ソフトウェア開発における企画，開発，運用，保守等の様々なプロセス(工程と呼ばれることもある。)にかかわる諸問題を明確にし，より良い状態へと改善していくことである。当社では，ソフトウェアプロセス改善活動に先立ち，社外のリードアセッサを招聘(しょうへい)したソフトウェアプロセスアセスメントも行っており，ソフトウェアの品質・生産性向上をねらいにした改善活動を全社的に展開している(図1)。

また，従来からプロセス改善を自力で進められるように，ステップアップガイドを作成し，ソフトウェア組織がソフトウェアプロセスを改善し，その能力を向上させるためのガイドラインを提供している。このガイドラインは，SW-CMM等の参照モデルへの対応を行い，プロセスを改善するための重要な8つの項目(仕様管理，工程管理，外注管理，品質保証・品質管理，構成管理，組織管理，見積計画・原価管理，技術管理)を定義している。また，組織がそのプロセスを改善するために必要なステップと行動を体系的に示しているため，このガイドが提供している評価欄を利用して現在の組織の状態を診断することができる。ガイドラインに沿って各項目における自部門の状態を採点することによって，今後の改善活動のターゲットを明確に設定することが可能となる。これらの活動の成果として，市場におけるソフトウェア品質の向上やプロジェクトの計画と実績の差異が少なくなっているという効果を得ている。

3.2 政府，国際規格への対応

最近，調達の面でもプロセス改善活動等に対する実績を評価することがうたわれ，ベンダーの情報システム開発プロセスについて，客観的評価基準設定の必要性が指摘された[2]。また，IEC(国際電気標準会議)やISO(国際標準化機構)などにおいて国際規格の整備が進むにつれて，海外案件を中心に規格対応を納入要件に掲げる事業者が増えており，それらへの対応を積極的に実施している。

(注1) CMMは，Carnegie Mellon大学の商標である。

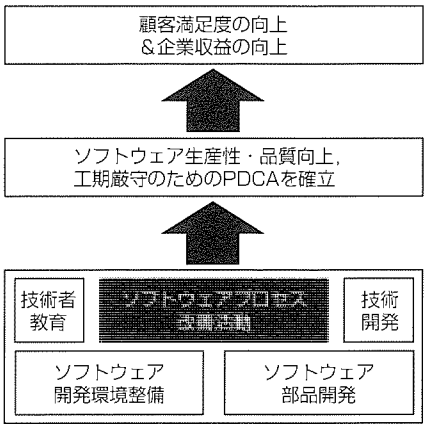


図1. ソフトウェアプロセス改善活動

3.3 定量的プロセス管理

組織としての開発プロセスを明確に定義し，ソフトウェア開発という活動を可視化することにより，ソフトウェア開発プロセスに発生する問題点を逸早く発見し，それを直ちに是正する仕組み作りが重要である。そのためには，客観的な定量的データを収集・分析し，コントロールサイクルを回す必要がある(図2)。製品の品質・生産性を計測して初めて，品質・生産性について議論することが可能となる。

客観的な尺度としてのデータとしてはソフトウェア製品の品質と生産性という2つの視点があり，重要なソフトウェア評価尺度は，製品／プロジェクトがどのように変わりつつあるのかを示す尺度である。

当社では，これまで，進捗(しんちょく)管理，障害管理，構成管理，デザインレビュー管理などソフトウェア開発管理の活動をサポートするツールの開発と社内普及活動を行ってきた。これらの個別の管理手法を統合し，スケジュール，コスト，品質，リスクといったプロジェクトのすべての状況をタイムリーに把握し，多角的な分析結果をプロジェクト関係者や経営層に提供するための情報インフラ“ソフトウェア開発管理システム”の開発及びそれを利用したプロセス改善に取り組んでいる。

この特集では，プロセス改善状況として，三菱電機インフォメーションシステムズ㈱(MDIS)と当社鎌倉製作所の事例を紹介する。MDISでは，プロジェクトマネジメントポータルサイト構築に当たり，プロセスデータベース環境を整備し，プロジェクトに関する事例や，生産・品質統計情報などの計測データ等をMDIS全社から閲覧可能としている。

また，鎌倉製作所では，ソフトウェア開発プロセスのうち，下流工程に重点を置いて，ソフトウェア開発手法の改善とそれをサポートするためのツール施策を推進している。

3.4 試験(テスト)

品質と生産性を両立させる対策として，試験工程に注目する必要がある。試験は多くの工数が必要となる工程であり，さらに開発スケジュールの終盤で行われるため，コストとスケジュールの点から見てリスクの大きな工程と言える。費用の観点から言うと，バグ発見のための費用とバグ修正のための費用がかかるが，バグが発見される工程が後になるほどこのコストは急激に増加する。また，同じバグであっても，規模が大きくなるに従い，修正費用は増加する。そこで当社では，試験の最初の段階である単体試験を重要視し(図3)，検出すべきバグを確実に検出し，後の試験工程にバグを流出させないとの考えから単体試験の作業品質・効率を向上させる施策に注力し，試験工程の生産性向上に取り組んでいる。

この特集では，当社が開発したMicrosoft Visual C++用の試験支援環境の特長及び機能について紹介する。この環境はC言語及びC++言語を対象としており，この環境を使用することで，単体試験及び組合せ試験における作業の大幅な自動化が可能となり，ソフトウェアの開発効率が向上する。単体試験支援環境を用いることにより機械的で時間を要する作業が大幅に省力化され，最も重要な試験手続きの作成に集中することができる。また，自動的にカバレッジが測定されることにより試験の充足度の測定及び終了判定の基準を得ることが可能となる。さらに，いったん試験手続きを作成すれば，機能改修等を行った場合の回帰試験の手間も大幅に削減できる。

4. ソフトウェア再利用性の向上

2章で述べたソフトウェア課題解決のため，ソフトウェア開発手法としては実績のあるソフトウェアを効果的に再利用する技術が大きな役割を果たす。

4.1 ソフトウェア再利用技術の概要

ソフトウェアの再利用技術は，部品ベースで再利用し全体ソフトウェアを構成する技術と，同一仕様のソフトウェアを異なるOSやCPUを利用したプラットフォーム上で動作させるため全体コードを再利用する技術に分けられる(図4)。

部品ベースの再利用技術は，サブルーチン，モジュール，オブジェクトと発展してきた。1980年代初期から注目されてきたオジェクト指向技術は，モデル化技術としてのUML(注2)やJava(注3)等のプログラミング言語として標準化

(注2) UMLは，OMGの商標である。

(注3) Javaは，米国Sun Microsystems，Inc.の米国及びその他の国における商標又は登録商標である。

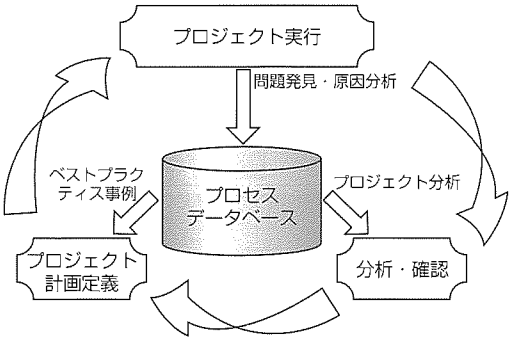


図2．プロセス改善のサイクル

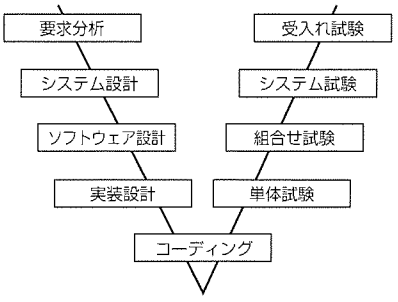


図3．ソフトウェア開発プロセス

が進みつつある。しかしながら，オブジェクト指向技術を単に道具として使用し場当たり的で偶然に頼ったソフトウェア部品の再利用ではシステム開発を効率的に進めることはできない。再利用を戦略的に進める仕掛けが必要であり，再利用の機構が作り込まれた分野対応フレームワークの開発に大きな期待が寄せられている。

分野対応フレームワークによる戦略的再利用技術は，カーネギーメロン大学の ソフトウェア工学研究所を中心に研究開発が進められており，SPL(Software Product Line)と呼ばれている[3]。この特集では，分野対応フレームワークとSPLを同義として扱う。

一方，全体コードを再利用するポーティング(移植)技術は，膨大な資産として存在するレガシーソフトウェアを新しいプラットフォーム上で有効に利用するため必要となる。また，良質なソフトウェアの流通のため，適用プラットフォームの拡大手段としても有効である。

4.2 分野対応フレームワーク

4.2.1 戦略的再利用技術

分野対応フレームワークは，ソフトウェア部品群やツールに加え，対象分野のソフトウェアをどのように構成するかを規定することで戦略的に再利用可能な形にまとめた枠組みである(図5)。

分野対応フレームワークによるシステム開発の特色を以下に挙げる。

(1) 対象領域の分析とカスタマイズ

分野対応フレームワークでは，その分野に対する綿密な要求分析・機能分析を行い，製品又は機種ごとの共通点・変更点を明らかにしなければならない。共通点についてはフレームワークに再利用の手段を組み込み，変更点についてはカスタマイズを効率良く実現するためのツールが必要である(図6)。

(2) ライフサイクルに対する再利用のサポート

ソフトウェア開発は，プログラムコードの作成だけではなく，ドキュメント作成や試験等というプロセスを経て達成されるものである。プロセスの各段階で知的な生産活動がなされて初めて，成果物は再利用対象となる。試験の場合では，例えば変数がとり得る範囲の境界値など，各分野で特徴となる試験ケースがある。このようなケースを含めた試験仕様がソフトウェア生産のための知的資産であり，製品系列間で再利用できれば，効果的な試験が行える。したがって，戦略的再利用を更に進めるため，分野対応フレームワークは開発プロセスの各段階での知的生産物を利用できる構造でなければならない。SPLではこれらの知的生産物はコア資産と呼ばれている。

(3) 開発コスト

分野対応フレームワークの開発コストは，機種系列の開発を進める中で，トータルコストとして見る必要がある。

トータル開発コスト＝

初期開発コスト＋Σ機種ごとのカスタマイズコスト

まず，分野分析のためのコストやカスタマイズツールを含めたフレームワークの初期開発コストが必要とされる。そのコストは，再利用を配慮せずに開発した場合のコストを超える可能性がある。しかし，同じ分野での機種開発を繰り返していく中で，資産の戦略的再利用によりカスタマイズコストを抑えることができる。結果として，初期投資

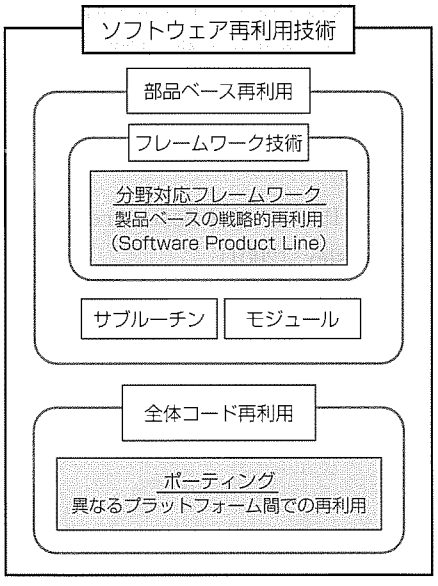


図4．ソフトウェア再利用技術の全体像

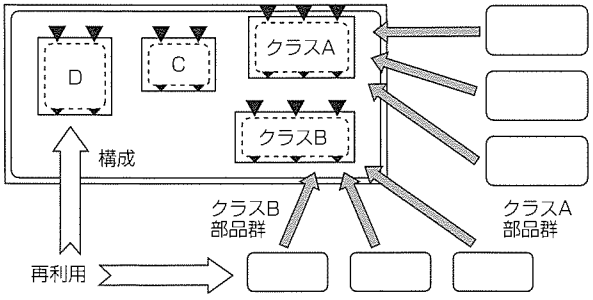


図5．フレームワークの構成イメージ例

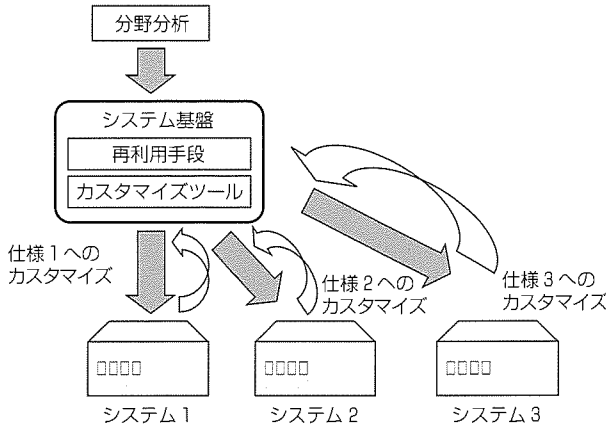


図6．戦略的再利用とカスタマイズによるシステム開発

を回収し，トータルの開発コストを低減することになる。

4.2.2 開発事例

この特集では，以下の分野を対象としたフレームワーク技術を掲載している。

まず，エンジニアリングからオンライン実行系までを一貫して支援する監視制御向けフレームワークを紹介する。この技術は1990年代から電力・公共向けに開発を行い，その後，ビル，宇宙等の分野へと展開している。

次に，携帯電話，カーナビ，列車統合管理装置などの組込みシステムに対する分野対応フレームワークの開発事例を紹介する。組込みシステムへのソフトウェア適用が急速に拡大しており，生産性と品質向上のため，分野対応フレームワークの開発は急務である。それぞれの対象分野を分析することにより，製品ごとの共通部を再利用し，カスタマイズ対象を効率的に生産するフレームワークとしている。

また，業務システムの大規模化・複雑化も増すばかりであり，効率的な開発が不可欠である。この分野では，クライアント／サーバによる基幹業務システム開発と，さらにWebを用いたシステム開発を紹介する。

なお，SPLに関するワークショップが2000年から開催され，自動車，航空機，携帯電話等，多方面での研究開発や適用事例が報告されている[4]。

4.3 ポーティングによる再利用

4.3.1 ポーティング技術

ソフトウェアのポーティングは，プログラムコードを利用する技術とバイナリーコードを利用する技術に分けられる。前者では，プラットフォーム間の差異を，例えばAPI(Application Program Interface)のエミュレーションにより埋める。後者では，ソフトウェアによるバイナリートランスレーションにより命令コードを変換する機能を提供する。バイナリートランスレーションは，特にプロセッサの世代交替におけるソフトウェア再利用を容易にする。

4.3.2 開発事例

この特集では，プログラムコード利用のポーティング技術として，Windows(注4)向けプログラムのUNIX(注5)環境上への移植のために用いられるAPIのエミュレートライブラリ開発を紹介する。バイナリーコード利用としては，トランスレーションの高速化技術と組込みシステム適用への課題について紹介する。

(注4) Windowsは，米国Microsoft Corp.の米国及びその他の国における登録商標である。

(注5) UNIXは，米国及びその他の国におけるThe Open Groupの登録商標である。

5. むすび

品質の高いソフトウェアを効率良く作るには，プロセス，手法両面の革新が重要であり，これにより，ソフトウェア開発組織の成熟度は向上し，また，安定したソフトウェアの再利用が可能となる。

当社は，ソフトウェア開発におけるプロセス及び手法の革新により，提供する製品の付加価値を高め，お客様に満足いただく製品作りに貢献していく所存である。

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) 経済産業省商務情報政策局情報処理振興課：ソフトウェアプロセス改善に向けて～SPIへの今後の取り組み～（2002－4）

(3) Clements, P., et al.：Software Product Lines, Addison Wesley（2001）

(4) Donohoe, P., ：Software Product Lines；Experience and Research Directions, Kluwer Academic Publishers（2000）

組織のソフトウェア開発成熟度を向上させるプロセス改善

佐々木　誠*
真野哲也**
小林正幸**

要　旨

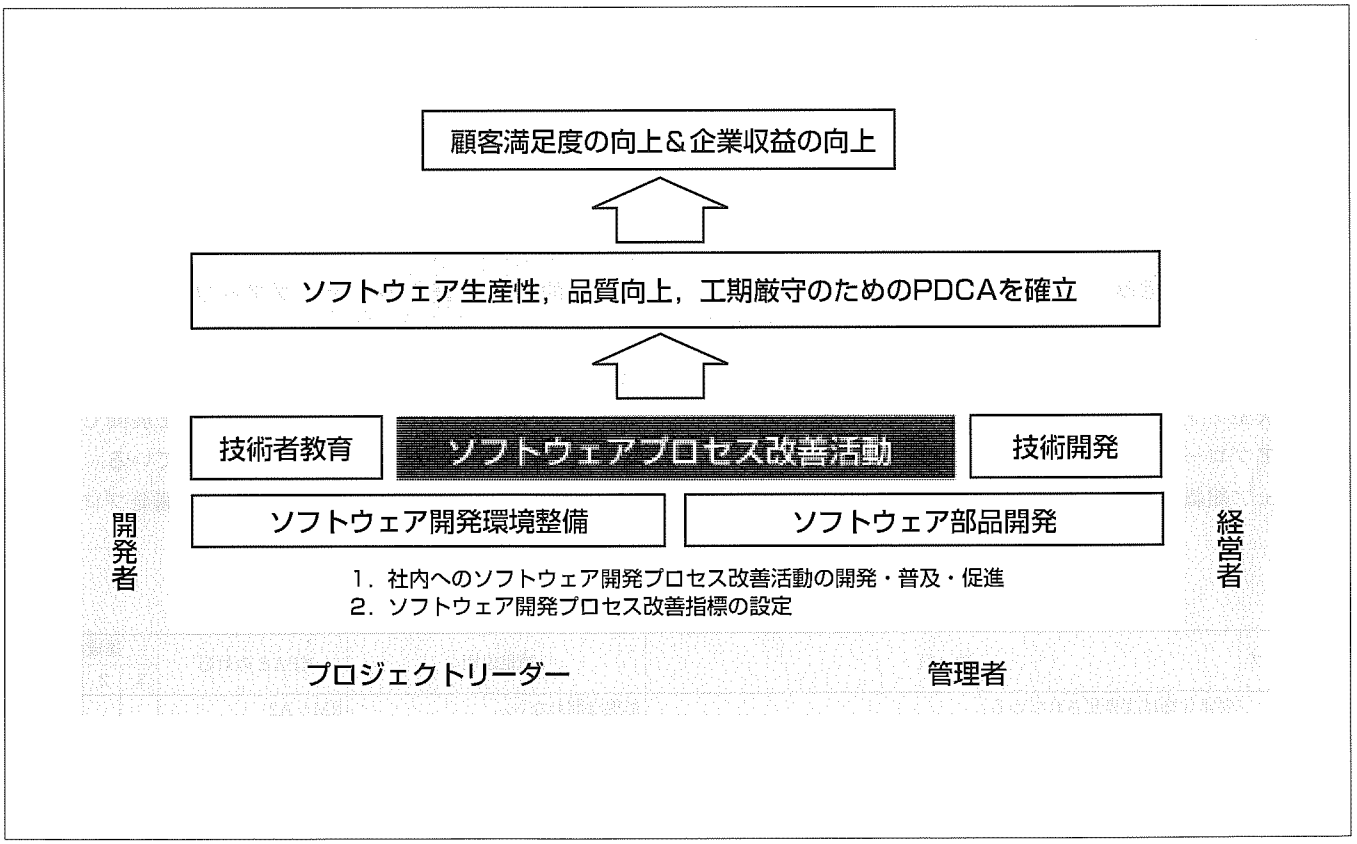
ユビキタス社会の到来によりIT(Information Technology)技術の普及が進み，IT製品やITソリューションの効率的な提供が企業にとってますます重要になっている。IT技術を支えるソフトウェア開発も，中国，インドなどアジア諸国のソフトウェア開発労働力の活用，CMM(注1)(Capability Maturity Model)などに代表されるプロセス改善活動が一般的になるなど，その取り巻く環境は大きく変わりつつある。

また，三菱電機では，ソフトウェアはソフトウェア製品の形で，又は製品に組み込まれた形で当社製品の広い範囲に適用されており，大小様々な規模の開発プロジェクトチームにおいてソフトウェア開発を実施している。これらのプロジェクトチームを円滑にかつ効率的に推進させるために，様々な活動を展開している。

本稿では，顧客満足度や企業収益の向上を達成するために当社が実施している技術者教育，ソフトウェアプロセス改善活動，技術開発，ソフトウェア開発環境整備，ソフトウェア部品開発といった諸活動のうち，ソフトウェアプロセス改善活動に着目してソフトウェアプロセス改善活動に関する実践状況を紹介する。

当社では幾つかの事業所や関係会社においてソフトウェアプロセス改善活動に既に取り組んでおり，ソフトウェアプロジェクト管理の徹底やソフトウェア構成管理，ソフトウェア品質保証を事業所や関係会社の置かれた状況に合わせて実践している。これらの活動の結果，CMMレベル3の認定を受けた部門もあり，確実に改善が進んでいる。

(注1)　CMMとCapability Maturity Modelは，米国Carnegie Mellon大学の商標である。



顧客満足度及び企業収益を向上させる活動

ソフトウェアの生産性向上・品質向上・工期厳守を常に達成できるようにするために，開発者，プロジェクトリーダー，管理者，経営者が一丸となって組織の力を充実させることが重要である。そのためには，標準開発プロセスを設定するソフトウェアプロセス改善活動，標準開発プロセスや設計技法／試験技法を開発者に定着させる教育，ソフトウェアに関する新技術開発や開発環境整備，部品開発などを継続的に実施していかねばならない。

*生産技術部　**設計システム技術センター

1. まえがき

ソフトウェアはソフトウェア製品の形で又は製品に組み込まれた形で当社製品の広い範囲に適用されており，大小様々な規模の開発プロジェクトチームが活動している。計画どおりに開発が進まず工期の延期，工数の追加投入，開発範囲の縮小などを起こしているプロジェクトも一部にはあり，開発技法の革新，管理技術の向上など本質的な改善策に期待が寄せられている。

また，ソフトウェア製品又はソフトウェア組込み製品の発注側も，受注側が期待にこたえる開発能力又は開発管理能力を備えているかどうかを見極める必要に迫られている。

こうした情勢の中で，管理能力の国際標準として，CMM[(1)]などが注目されるようになってきた。実際の商談においてCMMに沿ったプロセス改善の取り組みが要請されるなど，現実的な課題となりつつある。

2. ソフトウェアプロセス改善への取り組み

2.1 従来の取り組み

これまでは，品質上・工程上の問題を解決するためには，工程内の問題の発生，市場における問題発生をきっかけにして，問題の発生原因を突き止め，それを解決する施策を立案して実行するといった問題解決型のアプローチを実施してきた[(2)]。このアプローチは，個人の持つノウハウを組織のノウハウとして構築することができるとともに，特定の問題の解決には極めて有効である。しかし，昨今要求されている顧客満足度向上，工期の大幅な短縮化，工数の大幅な削減といった抜本的な改善については，必ずしも有効ではない。

2.2 現在の取り組み

IT型社会の到来に伴って，これまで以上に顧客満足度の向上及び企業収益の向上を実現していくために，従来の問題解決型アプローチだけでなく，改めて総合的な観点での改善アプローチが必要との認識に立ち，本来あるべきソフトウェア組織の姿を定義し，それに必要な改善を順番に実施していくような取り組み方法を開始している。

1990年代から，ソフトウェアプロジェクト管理の重要性に認識し，“プロジェクト崩れ”を防止するために，現状のソフトウェア開発部門の体質を診断できる仕組みを構築してきている。また，その当時カーネギーメロン大学ソフトウェア工学研究所で考案されていたCMMも参考にして，次の項目からなるステップアップガイドを作成している。

①仕様管理　②工程管理
③外注管理　④品質保証・品質管理
⑤構成管理　⑥組織管理
⑦見積計画・原価管理　⑧技術管理

なお，**表1**では，①の仕様管理の一部を抜粋してステップアップガイドの内容を示している。当社では，このステップアップガイドを現在も改定しており，CMMなどのモデルに従ったプロセス改善を実施するまでもない部門や，ソフトウェアの開発規模が小さい部門においても適用しており，自らがアセスメントしソフトウェア開発における弱点を見いだしている。表のステップアップガイドは組織がソフトウェアプロセスを改善するために必要なステップと行動を体系的に示しており，この表の評価欄を利用してその組織の状態を診断することもできる。また，KPA(Key Process Area) 欄には対応するCMM Ver1.1のキープラクティクスを示し，CMMによるソフトウェアプロセス診断にもすぐに移行できるようにしている。

3. ソフトウェア開発プロセス改善活動の実施・普及

改善活動による効果を早期に達成するために，モデルとなる部署を設定し，その部署の特徴などを考慮しつつ改善活動を展開している。

具体的には，**表1**のステップアップガイドやCMM，CMMI，ISO／IEC TR 15504を用いて，アセスメントを実施し，その部署の強み／弱み，改善の機会を明確にしている。このアセスメント結果を基に改善活動を展開して，再度アセスメントを実施し改善結果を確認している。各事業所やグループ内関係会社に，ソフトウェア開発プロセス改

表1．ソフトウェア開発プロセス改善のためのステップアップガイド(抜粋)

分類	ステップ	項　目	補足説明	KPAとの対応	達成状況			
					3	2	1	0
仕様作成	1	要求仕様は文書化されている。	形態は問わない。	RM－A2				
		仕様変更が生じた場合でも，確実に仕様書へ反映している。		RM－A2				
	2	仕様書は決められた作成要領書に基づき作成している。		SPE－ACT2，ACT3				
		上位の仕様から階層的に仕様書を作成している。		SPE－ACT2，ACT3				
		レビュー結果は確実に仕様書に反映している。		SPE－ACT2，ACT3				
	3	仕様書の内容(記載漏れ，記述方法等)は分析され改善を行っている。		SPE－ACT2				
		各仕様はトレーサビリティをとりながら展開している。		SPE－ACT10				
	4	要求を明確な設計仕様に変換するためのプロセス／手法がシステム的に利用できる仕組みがある。	決められたプロセス／手法がだれでも利用できる仕組みがある。	RM－A3 SPE－ACT2				

(注)　上表の達成状況欄に達成度合いを4段階（0～3）で評価した結果を記載することとしている。
(記号説明)　RM：要件管理　SPE：ソフトウェア成果物開発　A：実施能力　ACT：実施活動

善活動を展開していくために，改善プロセスの手順を提示し，具体的な改善活動を推進している。

また，必要に応じてリードアセッサによるCMMのレベルの認定取得活動も実施している。

3.1 ソフトウェアプロセス改善の手順化

ソフトウェアプロセス改善活動をより広く社内に推進するために，ISO/IEC TR 15504に記載されているプロセス改善の手順[3]を基にして，次からなる手順を用意している。

(1) 組織ニーズの検討

経営幹部が認識している解決すべき課題を把握し，それを改善活動のゴールに展開する。ゴールには経営的指標の改善を挙げるのが一般的である。

(2) プロセス改善活動の開始

改善活動の対象範囲の特定と改善活動のスケジュールの立案を行う。

(3) プロセスアセスメントの準備及び実施

プロセスアセスメントを実施し，現状のギャップ(課題)を明確にする。

(4) アセスメント結果の分析及び活動計画の策定

明確になったギャップを改善するための実施策を，対象部署の特徴を十分に考慮した上で計画する。また，改善の結果として得られる定量的なゴール(目標)についても設定する。

(5) 改善の実行

計画した改善策の実行とその状況を監視する。

(6) 改善の確認

設定したゴールを達成できているかを確認する。

(7) 改善目標の維持

改善成果を関係部門に水平展開する。

(8) 実施状況の監視

組織全体における改善実施状況を確認する。

3.2 プロセス改善の実施

これまでのアセスメントの経験から，CMMが要求している事項に比較して多かれ少なかれ次の項目に課題があると認識しており，これを解決するための改善活動を実施している。

- ソフトウェア品質保証
- ソフトウェアプロジェクト計画・管理
- ソフトウェア構成管理

(1) ソフトウェア品質保証に関する改善活動

当社では，ISO9001：2000年度版による改善活動の一環で，品質マネジメント活動に取り組んでおり，開発計画の立案時に機種又はプロジェクト単位に個別品質管理計画書の作成を義務づけている。この個別品質管理計画書を重要視しており，この計画書に管理の対象となる開発プロセスやドキュメントなどの成果物を明記し，その状況をホールドポイントでフォローできる仕組みを構築している。また，開発プロセスやドキュメントなどは，機種やプロジェクトの特徴に合わせて，所定のルールの下でカスタマイズできるように工夫している。

また，これらの品質保証活動を強化するために，必要に応じて品質保証担当組織(Quality Assurance Group：QAG)を設置して部署レベルで品質保証活動を統括している。

(2) ソフトウェアプロジェクト計画・管理に関する改善活動

PMBOK(Project Management Body of Knowledge)に記載されている体系を参考に，成果物の規模，スケジュール，工数，費用につき合理的な計画が立案できるようガイドや実績が計画から著しくずれた場合の是正処置方法を作成し，各プロジェクトで利用している。また，リスク管理については重要視しており，プロジェクト開始時点においてプロジェクトのリスクを抽出できるようチェックリストを用意し，抽出されたリスクがどのように推移しているかが管理フォローできる仕組みも構築している。

(3) ソフトウェア構成管理に関する改善活動

プログラムの修正による新たな障害の発生，旧版のプログラムのリリースといった問題も散見されており，これらを解決するために，ISO/IEC TR 15846を参考にし次の内容を記載した構成管理計画の作成活動を展開している。

- ソフトウェア構成管理の開始時期
- ソフトウェア構成品目の識別
- ソフトウェア構成制御の手順(変更妥当性の確認と変更状況の追跡性を確保)
- ソフトウェア構成管理の体制(構成管理者の任命と役割の付与)
- ソフトウェア構成管理で利用する様式
- ベースラインへの登録，ベースラインの変更
- ベースラインの監査

また，ソフトウェアの規模やソフトウェア構成管理業務量を勘案して，ソフトウェア構成管理環境の構築にも取り組んでいる。

4. ソフトウェア開発プロセス改善指標の設定

プロセス改善の効果を特定する場合は，一般的には経営への貢献度を表す顧客満足度，不具合発生状況，生産性(生産効率)，費用，工期などを組み合せて利用している(図1)。図に改善前，改善後の状況をプロットすることで，改善度合いを確認できる。

当社では，さらに，事業の置かれた状況にかんがみ，きめ細かな管理を行うために，ソフトウェア生産効率(生産性)を次の算式のとおり分割し，この算式に示す各指標の計測も行っている。

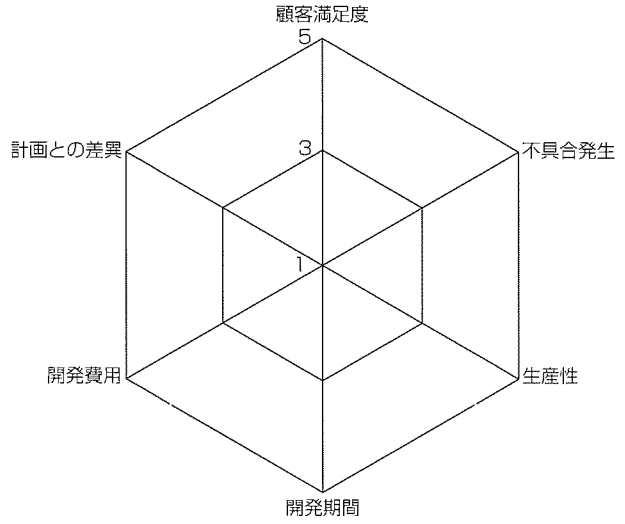


図1．プロセス改善の効果測定例

$$\frac{\text{生産高}}{\text{ソフトウェアコスト}}=\frac{\text{ソフトウェア外部委託費}}{\text{ソフトウェアコスト}}\times\frac{\text{生産量}}{\text{ソフトウェア外部委託費}}\times\frac{\text{新規開発量}}{\text{生産量}}\times\frac{\text{生産高}}{\text{新規開発量}}$$

（ソフトウェア生産効率）　（ソフトウェア外部依存率）　（ソフトウェア外部生産性）　（流用率）　（新規性評価）

各指標の評価の観点は次のとおりであるが，当該事業(製品)に応じて，指標を取捨選択している。

(1)　ソフトウェア生産効率(生産性)

計画した成果を得るまでに投入したソフトウェア開発コストと生産高の比で，ソフトウェア開発プロセスの効率性を示している。

(2)　ソフトウェア外部依存率

外部への依存の度合いが事業の特徴を考慮し適切な範囲内に収まっているかを評価する。

(3)　ソフトウェア外部生産性

外部へ委託しているソフトウェアの生産性が適切なものか否か，効率に問題がないかを評価する。

(4)　流用率

当該事業(製品)においてどの程度新規機能を開発しているか，これにより事業(製品)の成熟度を評価する。

(5)　新規性評価

新規開発量の生産高への寄与度，いわゆる新規機能が製品訴求度に与える影響を評価する。

5. むすび

ソフトウェアプロセス改善活動を既に実践しており，社外のリードアセッサを招聘(しょうへい)したソフトウェアプロセスアセスメントも必要に応じて行っており，ソフトウェアの品質・生産性向上をねらいにしてこれらの改善活動を全社的に展開している。これらの改善活動の結果，CMMレベル3の認定を受けた部門もあり，確実に改善が進んでいる[4]。並行して，プロセス改善活動を効率良く行うためにCMMなどのリードアセッサの育成も行っている。

これらの活動の成果として，市場におけるソフトウェア品質の向上やプロジェクトの計画と実績の差異が少なくなっているという効果を得ている。

今後は，プロジェクト計画作業を更に効率化していくために，プロジェクト計画時の参考となる良好事例(ベストプラクティクス)の作成や管理すべきWBS(Work Breakdown Structure)数の爆発を防ぐ意味を含めてWBSのテンプレートの作成を行う予定にしている。また，ソフトウェアプロセス改善だけでなく，技術者教育，技術開発，ソフトウェア開発環境の整備，ソフトウェア部品開発にも引き続き取り組んでいく。

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)　経済産業省ソフトウェア開発・調達プロセス改善協議会：ソフトウェアプロセスの改善に向けて～SPI活動への今後の取り組み～，経済産業省（2002）

(3)　ISO/IEC TR 15504-7(TR X 0021-7)：ソフトウェアプロセスアセスメント第7部　プロセス改善の手引き

(4)　芝田　晃，ほか：ITソリューションを支えるプロジェクトマネジメント(ISO9000，CMM)，三菱電機技報，**77**，No.4，291～294（2003）

ソフトウェア開発を成功に導くためのプロセス管理

山田裕子* 佐々木俊昌**
安田光宏* 藤原良一***
長江雅史*

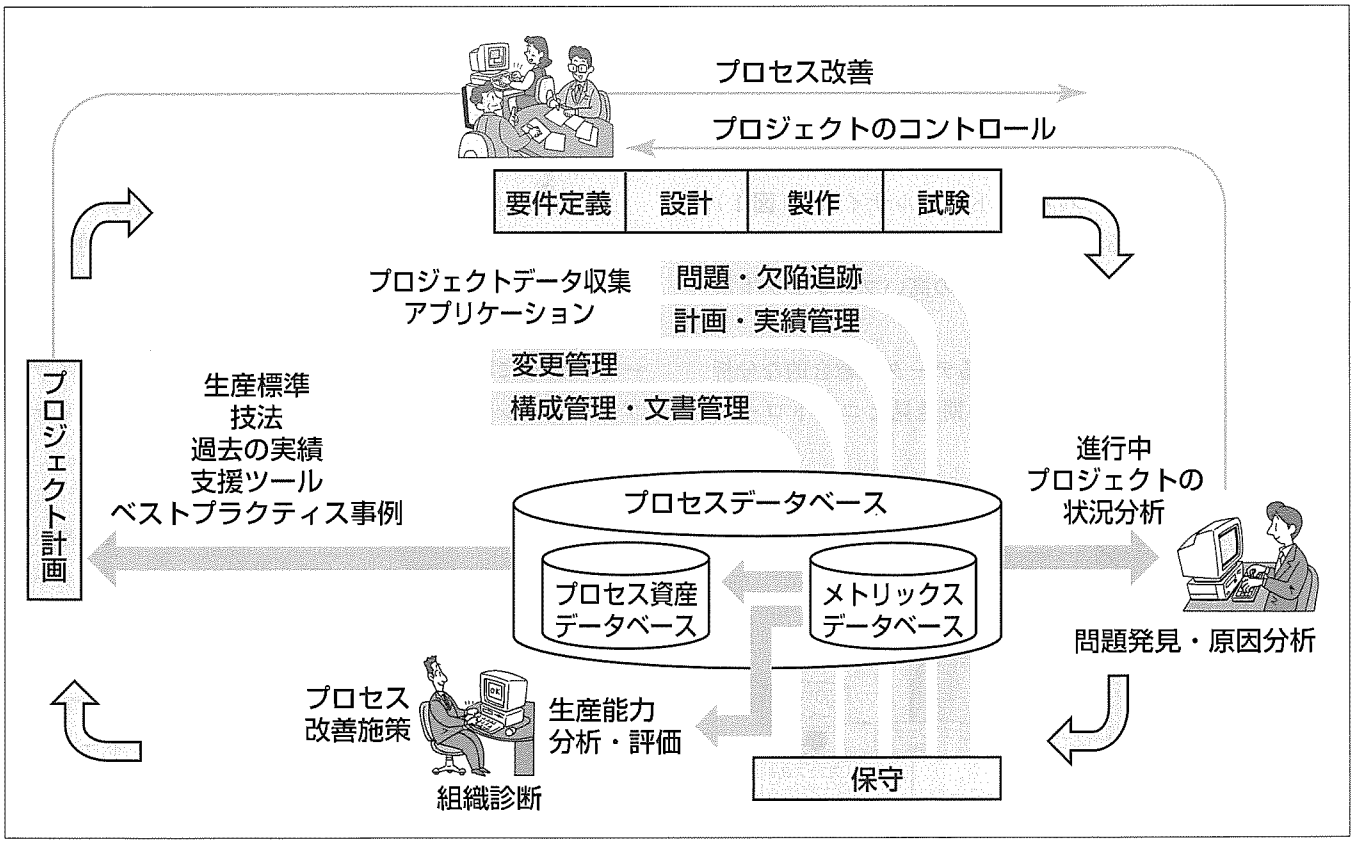
要 旨

情報システムの開発のみならず組込みシステムの開発においても，インターネット対応機能のサポートなどにより，急激な開発規模の増大と開発期間の短縮という変化に直面している。製品開発をビジネスとしての成功に導くためには規模が増大したソフトウェア開発の飛躍的な生産性の改善が求められるが，実際の開発の現場では，開発対象のプログラムコードが小規模だったころのプロジェクトマネージャー個人の能力に依存した管理形態が引き続き行われているケースも散見される。

このような背景から，三菱電機では，組織としての開発プロセスを明確に定義し，ソフトウェア開発という活動を可視化するための情報インフラを整備し，それを用いてソフトウェア開発プロセスに発生する問題点をいち早く発見し，的確なタイミングでコントロールする仕組み作りと，その普及・展開活動を進めている。

本稿では，この開発プロセスの標準化とその定着を円滑に進めるためのソフトウェア開発管理システムの構想とその整備・適用状況について述べる。

今後，更に統合されたマネジメントシステムの確立を目指し，活動を推進していく。



ソフトウェア開発のライフサイクルと開発管理をサポートする情報インフラ

ソフトウェア開発の現場で使用されている各種管理ツールから開発活動に関するデータを収集し，プロセスデータベースに蓄積する。このデータは，開発完了後に，次期類似開発のプロジェクト計画の精度の向上や実施したプロセス改善策の評価に利用される。また，プロセス改善の成熟度の高い組織では，現在進行中のプロジェクトの状況診断にも活用しており，大規模プロジェクトに発生する問題への速やかな対策の実施を可能とする。

*三菱電機㈱ 設計システム技術センター **同社 鎌倉製作所 ***三菱電機インフォメーションシステムズ㈱

1. まえがき

当社設計システム技術センターでは，これまで，進捗(しんちょく)管理，障害管理，構成管理，デザインレビュー管理などソフトウェア開発管理の活動をサポートするツールの開発と社内普及活動を行ってきた。これらの個別の管理手法を統合し，スケジュール，コスト，品質，リスクといったプロジェクトのすべての状況をタイムラグなく把握し，多角的な分析結果をプロジェクト関係者や経営層に提供するための情報インフラ“ソフトウェア開発管理システム”の開発及びそれを利用したプロセス改善に取り組んでいる。

2. ソフトウェア開発管理システム

2.1 システムのねらい

ソフトウェア開発プロジェクトの管理は，これまで，プロジェクトマネージャーがプロジェクトの状況を示すデータを集め手元ファイルで集約して分析・管理を行っていた。しかし，プロジェクトが大規模化し開発要員が100名を超えることも珍しくなくなった現在，この方法ではプロジェクト状況データを漏れなく収集できなかったり，集計は完了したが集計に時間がかかったため人員追加投入が遅れてしまったり，重大な品質問題の発生が放置されたりといった事態にもなりかねず，従来手法での管理は限界となっている。

ソフトウェア開発管理システムは，プロジェクトの状況把握をサポートするためのシステムであり，収集したデータを活用し，以下の3つのコントロールサイクル(**図1**)の形成をサポートすることをねらいとする。

(1) 過去の実績データに基づく精度の高いプロジェクト計画の立案(図の①)

(2) 開発の各フェーズで収集されるデータの総合的な分析機能の提供による，問題の早期発見とコントロール(是正)(図の②)

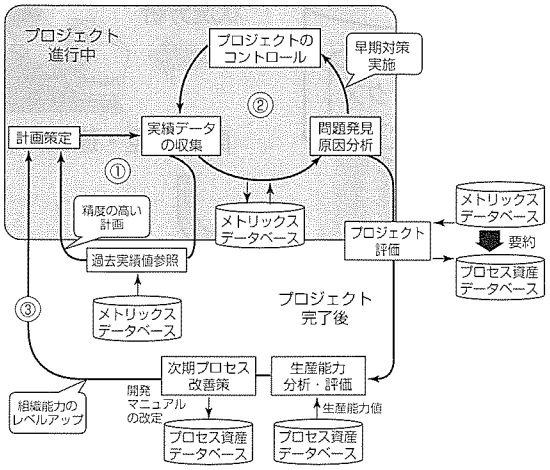


図1．ソフトウェア開発管理の3つのサイクルの形成

(3) 計測データを用いた組織の生産能力の客観的評価と，その継続的なレベルアップ活動による安定したソフトウェア開発組織の育成(図の③)

2.2 システム構成及び機能概要

ソフトウェア開発管理システムのシステム概要を要旨ページの図に示した。構成要素，機能は大きく以下の4つに区分される。

(1) プロセスデータベース

ソフトウェア開発プロセスの各種状況を記録・格納するデータベースであり，プロセスデータベースは，メトリックスデータベースと，プロセス資産データベースから構成される。

(a) プロセス資産データベース

プロセス資産データベースは，生産標準(標準開発マニュアル)や所属組織の生産指標(標準工数，品質目標値等)や事業所内の過去のプロジェクトの情報(ベストプラクティス)を格納するデータベースである。

(b) メトリックスデータベース

メトリックスデータベースは，プロジェクトを診断するための計測データ(メトリックス，例えば，要求仕様件数，プログラムコードサイズ，消化工数，障害発生件数など)を収集し，その履歴をヒストリカルに蓄積する。

プロジェクト完了時に，メトリックスデータベースに格納された実績データは要約され，プロセス資産データベースの所属組織の生産能力標準値に反映される。

(2) プロジェクトデータ収集アプリケーション

ソフトウェア開発プロセスの日々の状況を示すメトリックスを収集するクライアントとして動作する。定期的に開発状況(消化工数，成果物実績など)を収集するタイプのアプリケーションと変更管理ツール，問題・欠陥追跡管理ツールのように事象の発生によりデータを収集するタイプのアプリケーションがある。**図2**に，データ収集アプリケーションとその収集メトリックスについて例を示した。

(3) 進行中プロジェクトの状況分析機能

プロジェクトの進捗をモニタする進度モニタ機能や，障害発生状況の分析機能など，現在進行中のプロジェクトの状況について，メトリックスデータベースに格納されている生の計測データや分析結果を提供して，プロジェクトの問題を発見したりその原因分析をサポートする機能である。

この分析結果は，プロジェクトメンバー，マネージャー，上級管理層がいつでも閲覧可能である。これにより進行中のプロジェクトで発生している問題を早期に発見し，遅れなくプロジェクトをコントロールすることが可能となる。

(4) 生産能力分析・評価機能

プロジェクトの完了後に，プロジェクト進行中に収集した履歴データから組織の能力分析のためのデータを抽出し，

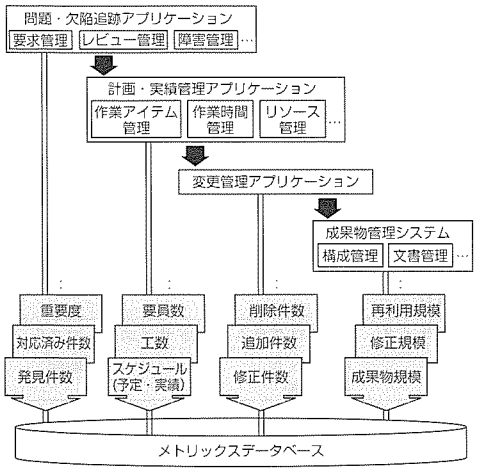


図2．データ収集アプリケーションと収集メトリックス

組織ごとに生産能力の分析・評価結果を提供する機能である。プロセス改善策の評価等に利用し，組織の継続的な生産能力の向上を図る。

2.3 課　題

ソフトウェア開発管理システムの各機能を構成する各アプリケーションとプロセスデータベースの接続には以下の課題を解決する必要がある。

(1) 既存アプリケーションの活用

当社の各事業所では，ソフトウェア開発の各フェーズ，各活動ごとに管理をサポートするアプリケーションを既に開発し適用済みである。また，これらのツールは，各事業所が生産する製品に最適となるように設計されている。そのため，ソフトウェア開発管理システムとして高度に統合化された一貫性のある市販システムへのリプレースはせず，各事業所が現在利用中のツールをクライアントとして最大限活用する形とし，それらのクライアントから収集されるソフトウェア開発活動に関する収集メトリックスの違いやアプリケーションが動作するプラットフォームの違いを吸収して統合する必要がある。

(2) 測定メトリックスの差異・変更に対する対応

幅広い製品群を製造対象としているため，事業所ごとに測定・分析すべきメトリックスに差異が存在する。また，ソフトウェア開発プロセスの改善が進めば，改善目的や改善対象も変化し，それに伴い測定・分析すべきメトリックスも変化する。測定・分析すべきメトリックスへの変化に対し，即座にかつ柔軟に対応できるシステムである必要がある。

2.4 接 続 方 式

2.3節で述べた課題を解決するために，ソフトウェア開発管理システムは，分散オブジェクト技術の1つであるWebサービス技術を利用して，各機能をネットワーク上のソフトウェア部品としてインプリメントすることとした。図3は，Webサービスでソフトウェア開発管理システムの各機能を接続した場合の構成図である。“プロセスデータベース”はWebサービスプロバイダーとしてサービスを利用するためのインタフェースを公開登録する。“プロジェクトデータ収集アプリケーション”“進行中プロジェクトの状況分析機能”“生産能力評価・分析機能”はいずれもWebサービスリクエスタとして動作し，Webサービスブローカーを介してプロセスデータベースのシステムインタフェースを問い合わせ，利用する疎結合(Loose coupling)で接続する。

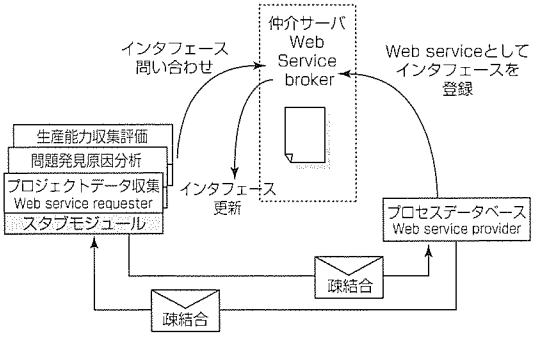


図3．Webサービスでシステムを構成した場合の各機能の接続

このような構成とすることにより，アプリケーションからプロセスデータベースへの利用要求は，プロセスデータベースのデータ構造に変更が生じた場合も，開発キットが生成するスタブモジュールで吸収可能である(必要に応じ，自動でのスタブモジュールの更新も可能である。)。また，Webサービスのメッセージの規格はSOAP(Simple Object Access Protocol)であり，データの記述はXML(Extensible Markup Language)を利用して行う。そのため，収集メトリックスに組織ごとの差異が存在する場合についても容易に対応可能となる。さらに，同じメトリックスを収集・分析するアプリケーションが事業所ごとに異なる場合の接続も可能となる。

3. 適 用 事 例

これまで述べてきたようなソフトウェア開発プロセスの診断や改善サイクルにプロジェクトの状況を示すメトリックスデータを集約しプロジェクト管理の精度を向上させるための環境整備状況につき社内事例を紹介する。

3.1 良い事例情報のプロセス資産データベースへの集約

三菱電機インフォメーションシステムズ㈱(MDIS)では，プロジェクトマネジメントポータルサイト構築の際に，プロセスデータベース環境を整備した(図4)。

プロセス資産データベースには，MDISのソリューション提供のノウハウを集約したシステム生産標準SPRINGAM(System Production and Integration Methodology：標準開発マニュアル)[1]，技法，支援ツールなどの情報とともに，過去のプロジェクトに関する良い事例や生産・品質統計情報などの計測データが蓄積されており，ベストプラクティスとしてMDIS全社から閲覧可能となって

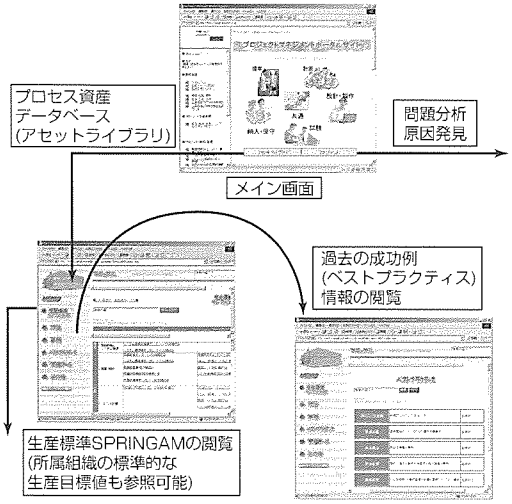


図4．プロセス資産データベース操作画面

いる。

このプロジェクトマネジメントポータルサイトによりMDIS全社としての生産・管理技法の標準化とそのレベルアップを図るとともに，自社の計測値とプロジェクトの生産性・品質目標とを比較することにより，開発者に生産能力の位置付けを客観的に把握してもらい，ソフトウェア開発プロセス改善への動機付けとすることを目的としている。

この活動も含めたプロセス改善活動を行い，2002年12月に公共システム事業分野で，SW-CMM(Software Capability Maturity Model)のレベル3を達成し，フィールド品質も約65%改善した。

3.2 現在進行中のプロジェクトへのメトリックス管理の適用

三菱電機の鎌倉製作所では，IT(Information Technology)推進活動であるKDI(Kamakura Digital Innovation)の4つのサブプロジェクトのうちの1つKDI-s(KDI for Software)として1999年にソフトウェア開発プロセスの改善活動を開始した[2]。

KDI-sでは，ソフトウェア開発プロセスのうち，下流工程に重点を置いて，ソフトウェア開発手法の改善とそれをサポートするためのツール施策を推進している。この活動の一環として，プロセスデータベースに測定したメトリックスデータを集約し多角的な分析を行う環境を構築した。この環境を，プロジェクトの診断や組織の生産能力のレベル測定に活用している。測定メトリックスとしては，表1に示した基本的なメトリックスを選択した。基本的なメトリックスを測定対象としたのは，条件の異なるプロジェクトにも適用可能とすることを目的としている。

鎌倉製作所では，ソフトウェア開発プロセスの改善活動サイクル(改善施策の設定→プロジェクトの計測→効果分析)への適用のみならず，現在進行中のプロジェクトの状況診断にもメトリックスによる管理を適用しており，開発後半に多量の障害を発生する可能性のあるプロジェクトを

表1．プロセス改善効果測定に使用したメトリックス

管理対象	測定メトリックス		
	設　計	製　作	試　験
進捗管理	・完了スケジュール(予定及び実績)	・プログラムコードサイズ	・完了スケジュール(予定及び実績) ・発生障害数
品質管理	・完了スケジュール(予定及び実績) ・デザインレビュー指摘事項数	・静的解析ツールの指摘事項数(※)	・完了スケジュール(予定及び実績) ・発生障害数
予算管理	・設計文書サイズ ・コスト ・期間	・プログラムコードサイズ ・コスト ・期間	・コスト ・期間

(※)　ソフトウェアプロセス改善活動の結果，製作フェーズの品質管理指標として測定メトリックスに追加

テストフェーズ初期に傾向把握し，発見することにより，開発完了までに立て直すための仕組みを確立した。

4. むすび

当社のソフトウェア開発は家電品から宇宙科学分野まで多分野に及んでおり，また，その規模もまちまちである。当初メトリックス計測によるソフトウェア開発管理は標準化が困難ではないかとの意見も多かった。しかしながら，鎌倉製作所での適用経験から，開発対象分野，開発規模に依存しない基本的なメトリックスでの計測であっても十分に活用可能との確信を得ることができた。

また，納期に追われるソフトウェア技術者の中には，管理のための状況データの入力の必要性に疑いを持つ人も少なくない。しかし，多くのプロジェクトの過去事例情報が集約され客観的なデータを整備することにより，見積り精度が上がり無理なプロジェクト計画が減少することや，技術者個人の能力を客観的に評価できる仕組みが整備されることなど，ソフトウェア技術者自身にとってのメリットも大きい。

上記のように，計測メトリックスの標準化，技術者のメトリックス計測への理解など解決すべき課題も残されているが，すべてのソフトウェア開発事業所への適用を目標に，ソフトウェア開発を成功に導くための各種管理ツールの統合化とプロジェクト計測によるプロセス管理技術の普及と定着を推進していく。

参考文献

(1)　中前雅之：システム生産標準の改善によるプロジェクトマネジメント技術力の向上，プロジェクトマネジメント学会　春季研究発表大会予稿集，73～76 (2002)

(2)　Sasaki, T., et al.：KDI-s：An Environment to Support Software Process Improvement Method, Proceedings of 2002 International Symposium on Empirical Software Engineering, II, Poster and Research Demonstration Session, 15～16 (2002－10)

ソフトウェア単体試験支援技術

藤本卓也* 小松 理**
松井聡一* 田口弘一***
岩垣征樹*

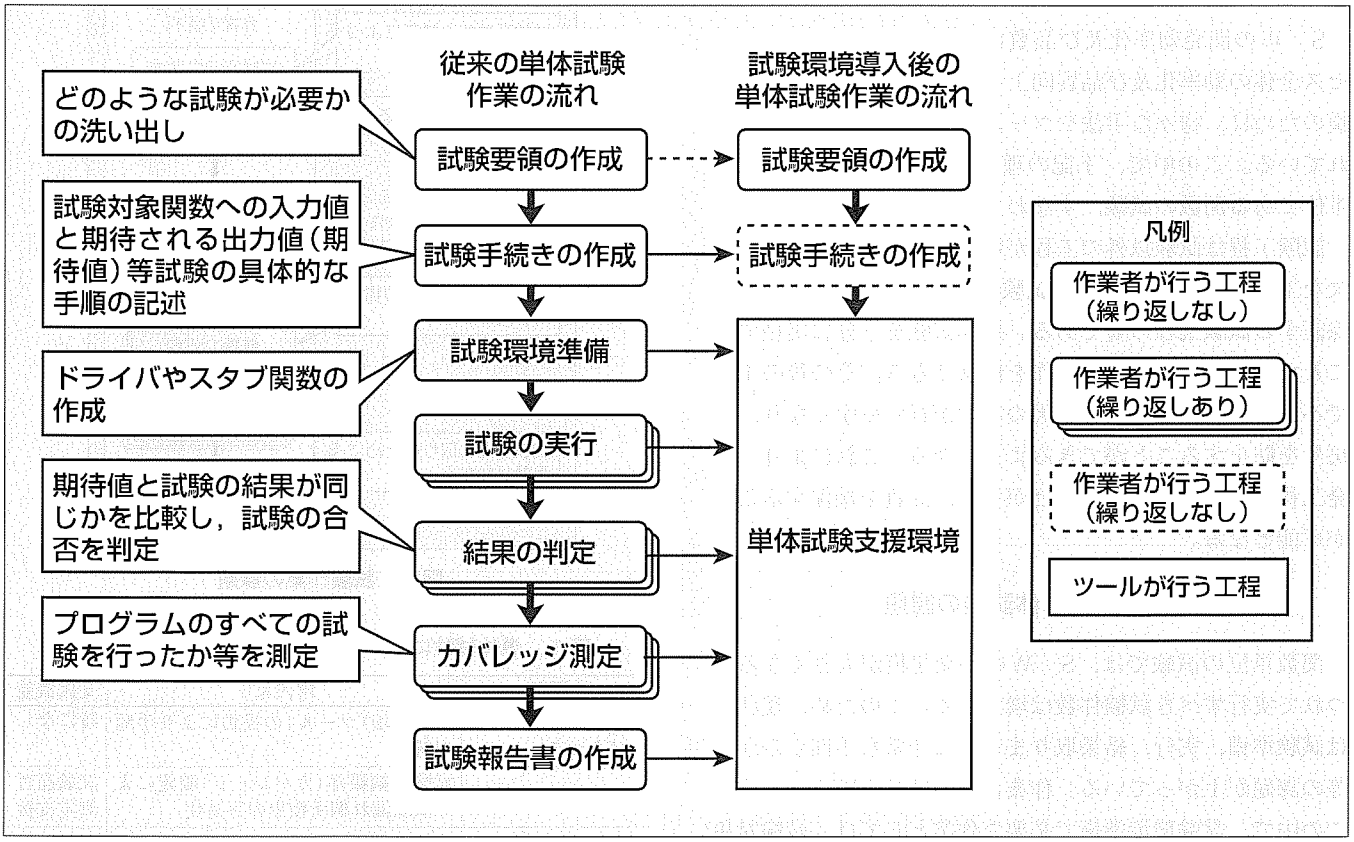
要 旨

種々の製品に組み込まれるソフトウェアの開発では，製品に対するユーザーの多様な要求にこたえるために製品の機能をソフトウェアで実現する傾向が強まり，年々開発すべきソフトウェア規模が拡大している。しかし，同じくユーザーの要求にこたえる製品を速やかに市場へ投入するため，製品の開発サイクルはますます短くなり，ソフトウェアの開発期間の短縮が求められている。この状況において，品質を維持しつつソフトウェアを効率良く開発していくことは非常に困難になってきている。

品質維持と開発効率化を両立させる対策として，ソフトウェアの最小構成単位に対する試験となる“ソフトウェア単体試験工程”に注目し，単体試験の作業を支援する環境を構築した。単体試験は試験の要(かなめ)であり[1]，この作業を自動化することで作業効率を上げることができ，単体試験の質を向上させることで品質の維持及び単体試験以降の手戻りの減少を図ることが可能となる。また，この作業の支援は，ソフトウェアの製造時だけでなく，機能変更等によりソフトウェアの改修が行われた際の回帰試験の省力化にも有効となる。

本稿では，三菱電機が開発したMicrosoft Visual C++(注1)用の試験支援環境の特長及び機能について紹介する。この環境はC言語及びC++言語を対象としており，この環境を使用することで，単体試験及び組合せ試験における作業の大幅な自動化が可能となり，ソフトウェアの開発効率が向上する。

(注1) Microsoft，Microsoft Windows，Microsoft Visual C++及びMicrosoft Excelは，米国Microsoft Corp.の米国及びその他の国における登録商標である。



従来(手作業による)と支援環境を使用した単体試験作業の比較

単体試験支援環境を用いることにより機械的で時間を要する作業が大幅に省力化され，最も重要な試験手続きの作成に集中することができる。自動的にカバレッジが測定されることにより試験の充足度の測定及び終了判定の基準を得ることが可能となる。また，いったん試験手続きを作成すれば，機能改修等を行った場合の回帰試験の手間も大幅に削減できる。

*三菱電機㈱ 設計システム技術センター **同社 稲沢製作所 ***三菱電機エンジニアリング㈱

1. まえがき

現在，当社が製造，販売する製品でソフトウェア(以下“S／W”という。)を搭載していないものは皆無と言っても過言ではない。なぜなら，ユーザーからの高機能，多様な要求にこたえ，競合他社との差別化を図るため，製品の機能を柔軟性の高いS／Wで実現する割合が大きくなっているからである。このため，S／Wの開発量は急激に増え続けている。しかし，その一方で，製品の開発サイクルは，ユーザー要求，差別化という同じ要因から短縮せざるを得ない状況にある。

このような事態に対して，S／W開発の効率化を図り，開発期間を短縮しつつ，品質を維持向上させる仕組みを構築・展開することが急務となっている。これに対する一つの方策として試験の最初の工程である“S／W単体試験工程”に注目し，単体試験工程の作業を支援する環境の開発を行った。単体試験工程で十分な試験を実施することでその後の工程での不具合発生を防止し，S／W開発の期間短縮と品質向上とを両立することが可能となる。また，S／Wの製造時だけでなく，機能変更等によりS／Wの改修が行われた際の回帰試験の省力化にも有効である。

2. 単体試験作業の必要性

S／Wの開発効率化及び品質向上には，S／W開発プロセス全体の効率化及び品質向上を行う必要がある。この支援のために，様々な手法やツールが利用され，活動が行われている。この中で，下記の理由から，S／Wの最小構成単位である関数の試験，すなわち単体試験に注目した。

試験工程は試験以外の工程がいかに効率化されても不可欠な工程であり，特に単体試験はS／Wの機能の正しさを保証する試験工程の要である。単体試験を十分に実施することで関数ごとの機能の動作を保証できる。その後の工程で不具合が発見されても原因の究明が行いやすくなり，手戻りを防止することができるようになる。これにより，開発工程全体の短縮を図ることができ，品質を確保することが可能となる。

3. S／W単体試験の課題

関数単位の試験では，S／Wの開発規模が大きくなるにつれて実行すべき試験件数は増大する。このため，現状では試験準備，実行，結果取りまとめに非常な手間がかかる等の課題が上がっている。作業ごとの課題を図1に示す。この中で，試験環境準備で必要な作業としては，試験対象関数を呼び出すための試験ドライバ関数の作成や，試験に応じて試験対象関数が呼び出す試験スタブ関数の作成がある。

4. 試験作業の課題を解決するために必要な仕組み

図1の課題を解決するために，解決策を立てた。また，この解決策を実施するために，従来の単体試験作業を分析し，必要な支援機能を提案し，試験支援環境として実現した。表1にこの解決策と試験機能を示す。表では，試験手順のばらつきという課題に対しては支援機能を示していないが，この課題に対しては，表で示した支援機能を含んだ試験支援環境を提供することで解決されると考える。

実現した支援機能の詳細を以下に示す。以降，個々の試験項目を“試験手続き”と呼び，一連の試験手続きを組み合わせた試験のまとまりを“試験シート”と呼ぶこととする。

4.1 試験準備支援

試験の準備の工程では次の作業を行っている。

- 試験シート作成
- 試験手続き作成
- 試験ドライバ関数作成
- スタブ用ダミー関数，ダミー変数作成

そこで，これらの作成作業を効率化するために，次の機能を実現した。

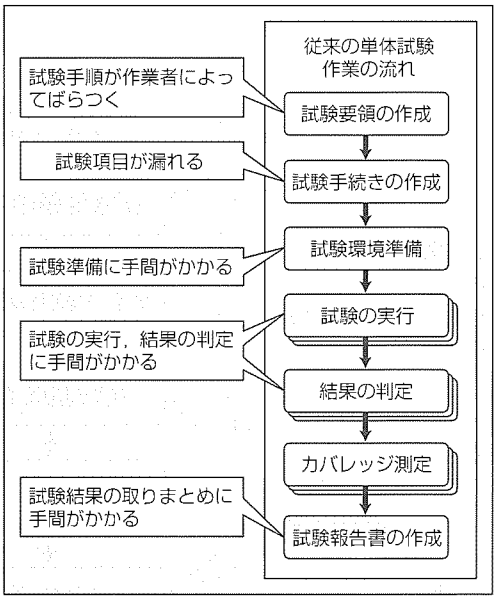


図1．試験作業の課題

表1．単体試験における課題への解決策と支援機能

課題	解決策	支援機能
試験手順のばらつき	試験環境(ツール)の提供により手順を統一	特になし
試験項目の漏れ	試験の網羅率(カバレッジ)測定により試験漏れ個所検出の容易化	試験品質測定支援
試験準備の手間	試験手続き，ドライバ及びスタブの作成支援により試験準備時間を短縮	試験準備支援
試験実行(値設定や確認)の手間	試験の自動化により試験を高速に実行(試験者の手間を一掃)	試験実行支援
試験結果取りまとめの手間	試験結果一覧や試験報告書の自動作成により取りまとめ時間を短縮	試験管理支援

- 試験シートのテンプレートの作成機能
- 類似試験手続きの作成支援機能
- 試験ドライバ関数の自動作成
- スタブ用ダミー関数，ダミー変数の自動作成

4.2 試験実行支援

試験の実行及び試験の結果の判定の工程では次の作業を行っている。

- ブレークポイントの設定等によるプログラム実行制御
- 値の設定
- 値の確認

そこで，これらの作業を効率化するために，試験シートに上記の内容が形式的に記述でき，その内容を自動実行する機能を実現した。

プログラムの実行制御においては，プログラムの修正が発生しても値の設定や確認を行う位置が特定可能な仕組みが必要である。また，試験の実行制御が柔軟に行えるように特定の条件が成立したときのみブレークする機能も重要である。

- 試験の一時停止機能
- 複数の試験の連続実行

また，値の設定／比較の記述を容易にするためには次の機能が必要である。

- マクロによる値の設定，比較
- 文字列の代入，比較
- 設定値，期待値での変数の使用
- 確認時の小数点以下の比較けた数指定
- 論理式による確認

4.3 試験結果の文章化支援

試験の判定及び試験報告書の作成の工程では次の作業を行っている。

- 試験合否の記入
- 試験進捗(しんちょく)の記入
- カバレッジの記入

さらに，試験全体の管理のため，次の作業を行っている。

- 試験全体の進捗記入
- 試験全体のカバレッジ記入

そこで，これらの作業を効率化するために，次の機能を実現した。

- 試験合否の自動作成
- 試験の進捗報告自動作成
- 試験カバレッジ状況報告自動作成
- 試験全体の進捗，カバレッジ状況報告自動作成

4.4 試験品質測定支援

試験作業では，試験作業自体の品質を向上させるために，試験ごとにカバレッジを測定し，その結果を試験手続きの作成に反映させることが重要である。

そこで，試験を実施した後，C0カバレッジ(全実行ステートメント中の通過ステートメントの比率)を測定する機能を実現した。

5. 試験支援環境

5.1 試験環境の概要

4章で抽出した機能を実現したS／W試験支援環境を構築した。図2は構築した試験環境の概要である。この試験環境は現在，Microsoft Windows(注1)用のS／Wのみならず，組込みS／Wの開発に適用されているMicrosoft(注1)社製のS／W統合開発環境であるMicrosoft Visual C++(以下“VC++”という。)と，表計算用アプリケーションであるMicrosoft Excel(注1)(以下“Excel”という。)を利用している。

この試験環境の特長は，S／Wの試験作業において，試験の手続きを試験シートとしてExcel上に記述するとこの試験手続きに従って単体試験をVC++上で自動実行し，その試験結果を試験シートに追記することで試験結果報告書としてまとめるところにある。試験手続き及び試験報告書のフォーマットは，Excelのマクロ機能を使用することでカスタマイズ可能である。

この試験環境の対象言語はC言語及びC++言語である。また，検証レベルはC／C++言語の論理的な検証である。このため，組込みS／Wの開発においてこの試験環境を使用した場合，実機を用いて試験を別途実施する必要がある。

5.2 デバッグ作業との連携

この試験環境はVC++のデバッガを利用している。このため，試験結果から試験対象のプログラムに誤りが検出された場合，そのプログラムのデバッグ作業が試験環境と同じ環境で実施できる。そこで，試験作業とデバッグ作業を連携させるため，図3に示すように，試験で誤りが発生した試験の確認箇所で試験を一時停止の指定が可能になるような仕組みを入れた。作業者は，この仕組みを利用して，誤りが発生した箇所，又はそれより手前の箇所まで試験を

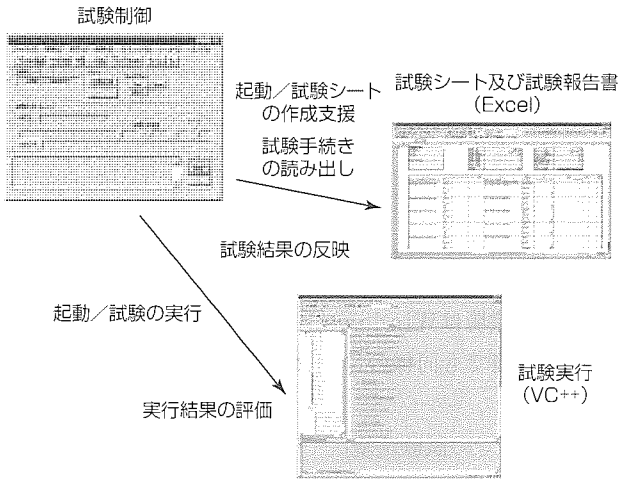


図2．試験環境の概要

試験手続き	合否	ポーズ
試験手続き設定 1		
試験手続き設定 2		
試験手続き確認 1	OK	
試験手続き確認 2	NG	ポーズ
試験手続き設定 3		

誤りが発生した時点に一時停止するように設定し，そこまでプログラムを実行して，その後，デバッグ作業に入る。

図3．デバッグ作業との連携

実行させる。その後は，VC++の機能を利用してプログラムのデバッグを行う。この仕組みを利用することで，デバッグのために，従来，試験で誤りが発生するための条件を再現する作業が効率化される。

6. 適用事例

この試験環境を応用した事例を以下に2つ示す。両者とも，これにより，試験の効率及び品質が向上した。特に，回帰試験の効率が大幅に向上した。

6.1 適用事例1

車両用の空調機のS／W開発のモジュール試験においてこの試験環境を適用した。従来VC++のデバッガ環境で作業者が試験仕様書を見ながら手作業で行っていた次の作業をExcel上に試験シートに記述した。

(1) ブレークポイントの設定
(2) ブレークポイントまでの実行
(3) 値の設定／確認

この試験シートの内容に基づいて試験を自動実行し，試験報告書を自動生成した。

6.2 適用事例2

エレベーターのS／W開発の機能組合せ試験においてこの試験環境を適用した。VC++上にエレベーターのS／Wを搭載したシミュレーション環境を用意し，これとこの試験環境を連携させた。シミュレータ上でのボタンの押下等の操作作業や，ランプの点灯などの確認作業をExcel上に形式的に記述し，従来シミュレータ上で作業者が手作業で確認していた作業を自動化した。この際，試験手続き間の間隔をブレークポイントの設定条件を設定することで制御している。また，作業がシミュレーション環境のGUI上で行った操作を記録してこの内容を試験シートに反映させる仕組みを作成することで，試験シートの作成効率を向上させた。

7. 他の開発環境での試験環境構築の取り組み

この試験環境はWindows上の試験環境であるが，次の他の開発環境でも同様のコンセプトの試験環境を構築している。

(1) UNIX上での試験環境
(2) Cygwin上での試験環境
(3) CPUシミュレータを実行環境とした試験環境
(4) ICE(In Circuit Emulator)を実行環境とした試験環境

8. 効果と今後の課題

(1) 試験の効率化

試験の自動化機能により値の設定や確認等の試験作業の効率が向上する。特に回帰試験においては，試験を自動実行するだけになるため，効率が大幅に向上する。また，試験手続きの作成支援機能，ドライバ，スタブの自動作成による試験環境準備機能，試験結果一覧，試験報告書の自動作成機能により，試験の準備，結果のまとめ，及び管理作業の効率が向上する。

(2) 試験作業の質の向上

試験作業の自動化により，値の設定，確認等の誤りが削減される。また，試験手順の統一により，試験作業の個人差が低減する。さらに，C0カバレッジ測定機能により試験漏れが防止される。

(3) S／W品質の向上

試験効率の向上により，限られた期間でより多くの試験項目が実施可能となり，これに伴いS／W品質が向上する。また，試験作業の質が向上することで，S／Wの安定的な品質が確保可能となる。

(4) デバッグ作業の効率化

試験作業者とデバッグ作業者が同一の場合，試験とデバッグがVC++上でシームレスに実施可能となる。その場合，試験環境を用いることでデバッグ時の不具合状態が容易に再現可能となる。

9. むすび

S／W開発の試験工程のうち最初の工程である単体試験工程に着目して，この工程の作業を支援する単体試験支援環境を開発した。この環境の使用により単体試験作業の省力化・効率化を実現し，S／W開発の効率化，期間短縮に寄与するとともに単体試験の充足度を高め，S／W品質を確保し，後工程及び市場での不具合発生を防止するための手段の一つを提供することができた。今後は，この単体試験支援環境に対して試験設計支援や障害管理ツールとのリンク等の機能拡充，組合せ試験への対応を行っていく。

参考文献

(1) Beizer, B.(小野間　彰，ほか訳)：ソフトウェアテスト技法(原書名：Software Testing Techniques, Second Edition)，日経BP出版センター (1994)

鉄道システムにおける開発プロセスの監査活動

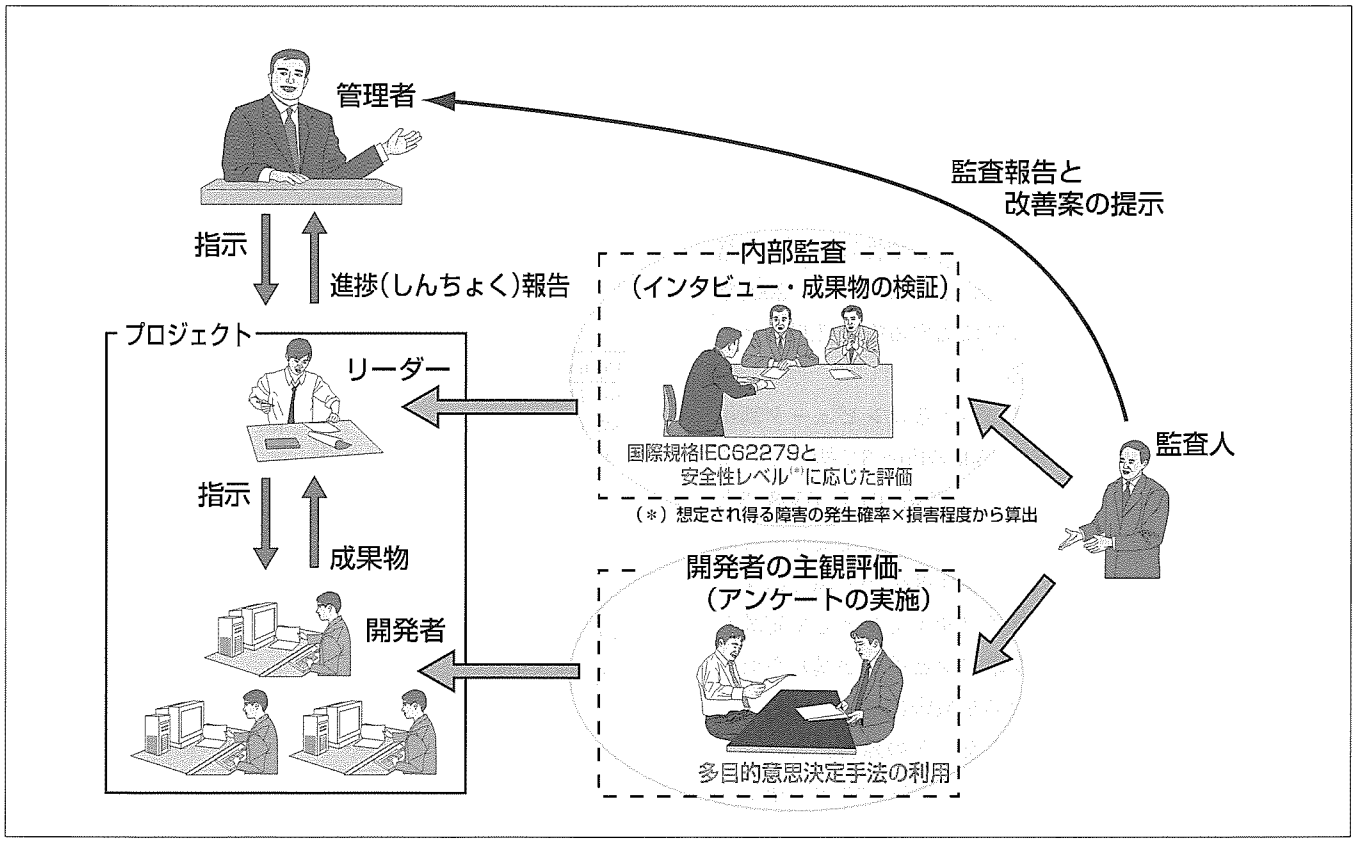
高橋　理*
石岡卓也*
駒谷喜代俊*

要　旨

近年，鉄道システム技術に関する国際規格の整備が進むにつれ，海外案件を中心に規格対応を納入要件に掲げる鉄道事業者が増えている。また，従来は独自の発展を遂げてきた国内市場にも、今後，海外メーカーが規格対応を武器に参入してくる可能性もあり，こうした動向を看過できない状況にある。

本稿では，まず，鉄道信号のソフトウェアに関する安全性規格として2002年秋に発行されたIEC62279の概要について述べる。IEC62279は，ソフトウェアの開発ライフサイクルと安全性レベルに応じて定められる技術要件を規定していることを特徴としている。また，このIEC62279への対応を円滑に進めるために実施している社内プロジェクトの内部監査活動について紹介する。監査は，定期的又は各工程の節目などにプロジェクトリーダーなどにインタビューする形式で行っており，単なる現状評価にとどまらず，規格を生産性向上に対する指針と考え，実現可能な改善策を話し合う場として活用している。

一方，国際規格などを利用した生産性向上や開発プロセス改善の活動は，開発者に新たな作業を強いることもあるためトップダウンに押し進めることが必要であるが，最近になって，その成否は現場開発者の現状プロセスに対する満足度にも依存することが分かってきた。そこで，現状の開発プロセスに対して，開発者が主観的に持つ改善必要性の程度を定量化する手法について紹介するとともに，それが改善策の立案に有効であることを併せて示す。



国際規格への対応と生産性向上の取り組み

ソフトウェア開発プロジェクトのリーダーに対してインタビュー形式で行う監査活動と開発者に対する満足度評価を実施することにより，国際規格への対応とそれを利用した生産性向上を効率良く進める。

*先端技術総合研究所(工博)

1. まえがき

従来，日本国内の鉄道に関する技術標準は，国内の鉄道事業者やメーカーなどによって作成・実施されており，国内市場においてはこれで十分であったために，国際規格に対する積極的な取り組みは余りなされてこなかった。

ところが，近年，IEC(国際電気標準会議)やISO(国際標準化機構)などにおいて国際規格の整備が進むにつれて，海外案件を中心に規格対応を納入要件に掲げる鉄道事業者が増えている。また，WTO(世界貿易機構)の調達協定により，国際規格と国内規格の整合化が義務づけられたため，今後は規格対応を武器にした海外メーカーの国内市場参入の可能性もあり，こうした動向を看過できない状況にある。

そこで，本稿では，鉄道システムを取り巻く国際規格の動向について説明するとともに，規格対応に関する取り組みの一例として，鉄道システム向けソフトウェアの開発プロセスに対して実施している内部監査活動について述べる。さらに，開発プロセスの新しい改善アプローチとして，現状の開発プロセスに対する開発者の主観評価を考慮した改善策の決定方法に関する研究成果について紹介する。

2. 鉄道信号システムに関する国際規格

国際標準化を行っている機関にはISOやIECがあり，ソフトウェア全般を対象とした開発プロセスのアセスメントに関する国際規格としてはISO15504TRがよく知られている。一方，鉄道電気設備の規格については，IECに設置されたTC9と呼ばれる専門委員会で審議されている。以前は，製品規格が審議の主流であったが，最近では，鉄道システム全体としての規格やマネジメント・安全性に関する規格も数多く審議されるようになっている。このうち，2002年秋には，鉄道信号のソフトウェアに関する安全性規格として，IEC62279が発行されている。

IEC62279は，一般産業分野におけるコンピュータ制御機器を対象にした国際規格IEC61508のPart.3で扱っているソフトウェアに関する部分(日本国内でもJIS C 0508-3として規格化済み)をベースに，鉄道信号の技術要件を反映したものである。ソフトウェアの要件抽出から保守に至る過程を**図1**に示すような開発ライフサイクルとして定義した上で，安全性確保のための要求事項とそれが満たされていることを明らかにするプロセスを厳密に規定している。また，対象とする装置・システムを，その障害によって生じ得る被害の程度と発生確率を基に全く安全性を考慮しなくてもよいレベル(レベル0)から極めて高い安全性を必要とするレベル(レベル4)に分類し，要求される安全性レベルに応じた安全性技術の適用を規定している。

表1及び**表2**は，安全性レベルに応じて必要とされるドキュメント成果物やその検証・試験方法を示したものである。表中において，HR(Highly Recommended)は利用が強く推奨される技術及び手法，R(Recommended)は利用が推奨される技術及び手法，－は特に推奨のない技術・手法であることを示し，対象とする装置が安全を脅かすものではない場合にはソフトウェア要件やプログラムコードと総合試験結果などの成果物だけを作成すればよい一方で，安全性レベルが高くなるほど厳密なドキュメントと多方面からの安全性検証が求められる。また，高い安全性レベルを要求される場合には，設計者と検証者はそれぞれ所属する組織が独立でなければならないといった要件なども加わる。

3. 国際規格への対応と生産性向上の取り組み

前章に述べたように，いまや国際規格への対応は不可欠になりつつあるが，規格対応に当たっては新たなドキュメントを作成しなくてはならない場合もある。また，必要な情報を持つドキュメントが既にあるにもかかわらず，規格

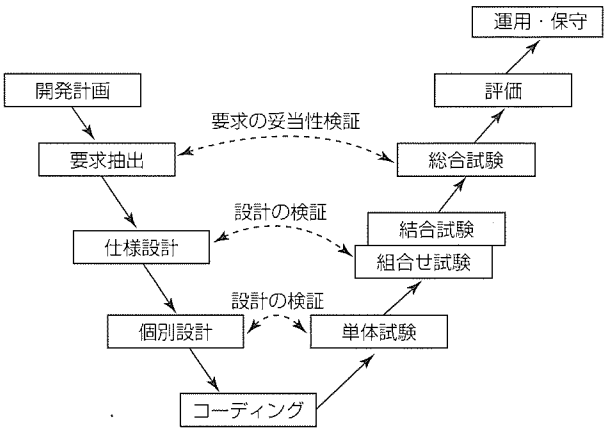


図1．ソフトウェア開発ライフサイクル(Vモデル)

表1．安全性レベルとドキュメント体系

Documentation	SIL 0	SIL 1／2	SIL 3／4
SW Planning Doc.	R	HR	HR
SW Requirement Doc.	R	HR	HR
SW Design Doc.	－	HR	HR
SW Module Doc.	－	HR	HR
Source Code and Doc.	R	HR	HR
SW Test Report	－	HR	HR
SW／HW Test Report	－	HR	HR
SW Validation Report	R	HR	HR
SW Assessment Report	－	HR	HR
SW Maintenance Records	R	HR	HR

表2．安全性レベルと検証方法

Technique	SIL 0	SIL 1／2	SIL 3／4
Formal Proof	－	R	HR
Probabilistic Testing	－	R	HR
Static Analysis	－	HR	HR
Dynamic Analysis and Test	－	HR	HR
Metrics	－	R	HR
Traceability Matrix	－	R	HR
SW Error Effect Analysis	－	R	HR

の定める体系と1対1に対応していないために新たに作成してしまう場合もあると考えられる。

そこで，三菱電機は，鉄道システムを対象とした国際規格対応を円滑に進めるため，複数の社内プロジェクトに対する監査を実施し，ドキュメント体系の整理や規格対応評価などの活動を行っている。監査は，定期的又は各工程の節目において，プロジェクトリーダー及び製作・品質管理部門の責任者にインタビューする形式で行っている。また，国際規格は開発上で遵守すべき規約であると同時に生産性向上に対する指針と考えることもできることから，監査の場では，単なる現状評価を行うだけでなく，現場の開発者とともに実現可能な改善策について話し合うことに主眼を置いている。そして，監査終了後には，監査結果及び改善策を管理者あてに送付し，管理者によるトップダウンな改善作業を促している。

この章では，これまでに実施してきた監査活動の内容とその結果として得られた成果の幾つかを紹介する。

3.1 仕様書項目と記述内容の整理

国際規格では，製品の開発計画から出荷後の保守に至るライフサイクルの各段階で作成すべき成果物が明確に定義されており，それに従って，数多くのドキュメントを整備する必要がある。一方，多くの開発組織は，たとえ国際規格と内容や名称の点で1対1には対応していなくても，国内規格や社内独自の規格に沿って開発を行っていることも多く，そういった場合には，国際規格の要求する項目を既にある程度は満足していると考えられる。

そこで，IEC62279の要求するドキュメント類と過去の類似製品開発の際に作成してきたドキュメント類の対応づけを行うとともに，その充実度に関する評価を行った。この結果，プロジェクト計画と単体試験に関して新たなドキュメント整備の必要があるが，他の工程については現状のままで対応できることが分かった。一方，規格対応を性急に対処しようとすると，不足する項目を単純に追加することになり，結果として同じ項目を複数のドキュメントに重複して記述することがあったり，プロジェクトごとに独自に項目の追加や削除を行うことによって今後の成果物流用に差し支えることもあると予想される。そこで，各ドキュメントに記述する項目についての整理も行っている。

3.2 実績収集

IEC62279やISO15504などの国際規格では，適切なプロジェクト計画とそれに従った管理・運営が行われているプロジェクトが望ましい姿とされ，プロジェクトの開始段階で，作業量の見積りやリソース配分，リスク分析などを行うことが必要とされている。これらの作業を精度良く行うためには，過去に実施した類似製品の工程実績を基に調整するのが良いと考えられるが，過去の製品開発では適切な情報収集が行われていなかったり，収集している情報の種類がまちまちであった。そこで，今後のプロジェクトにおける作業見積りの精度向上を目的として，仕様書レビューの指摘件数や，ソフトウェア試験の件数及び障害件数などの情報収集を始めることにした。

3.3 情報の横通し

複数のプロジェクトに対して監査を進めていると，個々のプロジェクトや開発者が工夫のとれた開発を進めていたり，ツールをうまく使いこなしていることがある。また，監査者や管理者のトップダウンによる作業指示は開発者の反発を招くこともあり得るが，他のプロジェクトが実際にやっている作業であれば，比較的スムーズに受け入れられる傾向がある。そこで，このようなプロジェクトごとの工夫をベストプラクティスと考え，他のプロジェクトに対して紹介する活動も併せて行っている。

4. 開発者主観の評価による改善作業の決定

国際規格を利用した生産性向上や開発プロセス改善の活動においては，その結果として開発者に新たな作業を強いる場合もあるため，第三者監査などによる客観評価に従ってトップダウンに活動を押し進めることが必要である。ところが，近年，改善活動の成否は，実際の活動を行う開発者の現状プロセスに対する満足度に大きく依存し，現場の開発者が不満に感じている作業に対する改善であるほどその効果が大きいことが分かってきた．そこで，この章では，開発者の主観評価を基にボトムアップな改善を進めるアプローチとして，現状の開発プロセスに対して開発者が持つ改善必要性の程度を定量化する手法を紹介する。

この章に紹介する手法は，開発者の主観評価を定量化する手法として，階層化意思決定法(Analytic Hierarchy Process：AHP)の改良型モデルを用いる。AHPは，評価構造を評価基準や代替案からなる階層構造に表現した上で各項目の一対比較によってそれぞれの相対的重要度を決定する多目的意思決定手法の1つであり，他の意思決定手法に比べて，実施が容易で手順が理解しやすく，定性的なデータを扱うこともできるという利点を持っている。

最初に，改善活動に着手する開発工程の順位を決定することを目的に，4つの開発工程を代替案とする階層構造を**図2**のように構成した．また，階層構造には，問題を評価する上で必要となる評価基準を過不足なく入れる必要がある。そこで，評価基準の第1層には，ソフトウェア開発プロセス評価モデルとして広く認知されているプロセス成熟度モデル(Capability Maturity Model：CMM)(注1)において，最初の到達目標となるレベル2に定義されている6つのKPA(Key Process Area)を配置し，さらにそれを分かりやすく書き下したサブ評価基準を作成した。

(注1) CMM及びCapability Maturity Modelは，米国Carnegie Mellon大学の商標である。

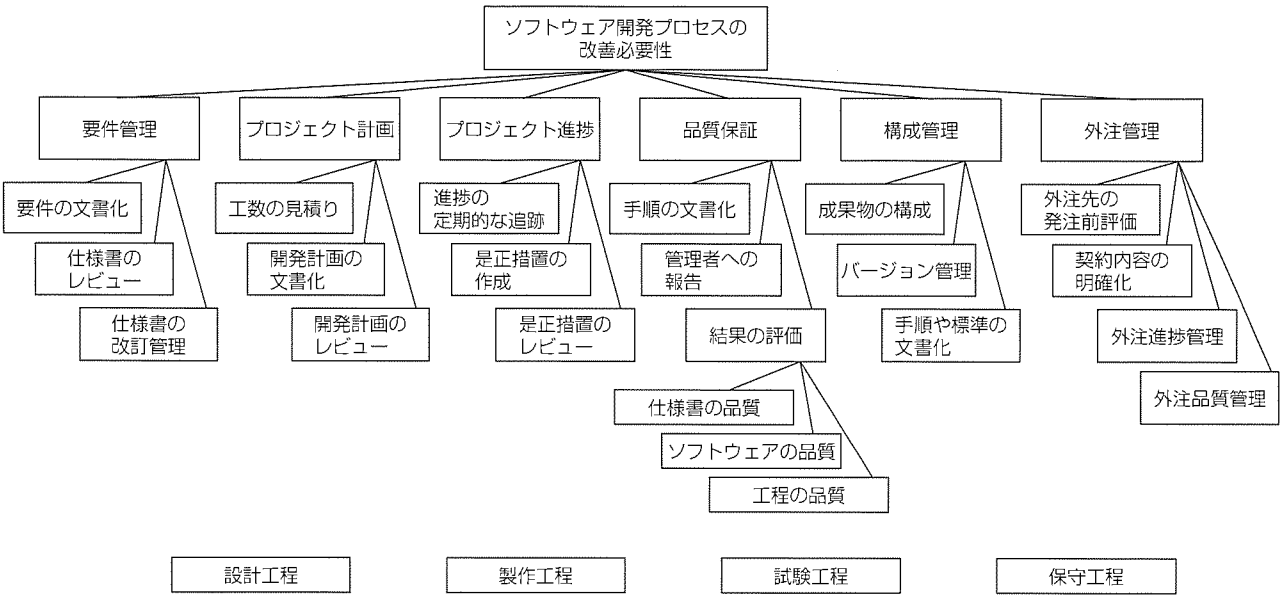


図2．開発プロセスの改善必要性を評価するための階層構造

そして，開発組織の中ですべての工程を横断的に見る立場にある中堅の開発者を被験者として，評価手法の内容と手順について説明した後，抽象的な用語の説明を行いながら，この階層構造の同一階層にあるすべての評価基準及び代替案の対について，1つ上の親評価基準の下での改善必要性の比を7つの選択肢から選んでもらった。このとき，開発者の一対比較を円滑に進めるため，各評価基準項目の“改善された状態”をイメージしやすくする目安テーブルを用意したり，回答困難な比較に対しては無理に回答してもらうのではなく回答の得られた項目の値から推定するなど，開発者の負荷を軽減する工夫を行った。

この手法をある開発者Aに適用した結果を**表3**に示す。第1行目に第1層の評価基準重要度を，第2行目以下に各評価基準の下での代替案の重要度を示す。これらの重要度はそれぞれ値が大きいほど改善の必要性があることを表し，開発者Aは製作工程と試験工程に対する改善必要性を強く感じていることを示している。また，この結果は，開発者の意識とおおむね一致するとの評価を得ることができた。

このようにして，開発者の開発プロセスに対する改善必要性評価を定量化することができる。また，各開発者への適用結果を比較することにより，互いの考え方の違いが明らかになるとともに，改善策立案のための合意形成の材料として利用できることが分かった。

表3．開発者Aの評価結果

	要件	計画	進捗	品質	構成	外注	総合
	0.23	0.17	0.08	0.39	0.05	0.08	1.00
設計	1.31	1.42	4.38	2.25	1.00	2.40	2.02
製作	3.00	2.28	1.57	4.90	4.38	2.26	3.50
試験	2.36	3.80	5.22	4.65	2.69	3.08	3.81
保守	1.62	2.69	1.27	2.25	2.69	0.72	2.00

5. むすび

以上，鉄道システムを取り巻く国際規格の動向について述べるとともに，規格対応の取り組みとして実施しているソフトウェア開発工程に対する監査活動と開発者の改善必要性に対する主観評価を用いた改善策立案について紹介した。この監査活動によるトップダウンアプローチと開発者主観を評価するボトムアップアプローチを組み合わせることによって，効率良い規格対応化と開発生産性の向上を図ることができると考えられる。

参考文献

(1) 電気学会鉄道技術標準化調査委員会編：規格戦略時代の鉄道技術標準化，電気学会技術報告，No.887（2002）
(2) 平尾裕司，ほか：鉄道信号の国際安全性に関する考察，電子情報通信学会技術報告，FTS2001-72（2001）
(3) 高橋　理，ほか：階層化意思決定法を用いたソフトウェア開発プロセスの改善必要性評価，システム制御情報学会論文誌，**16**，No.7（2003）

監視制御システム向けフレームワーク

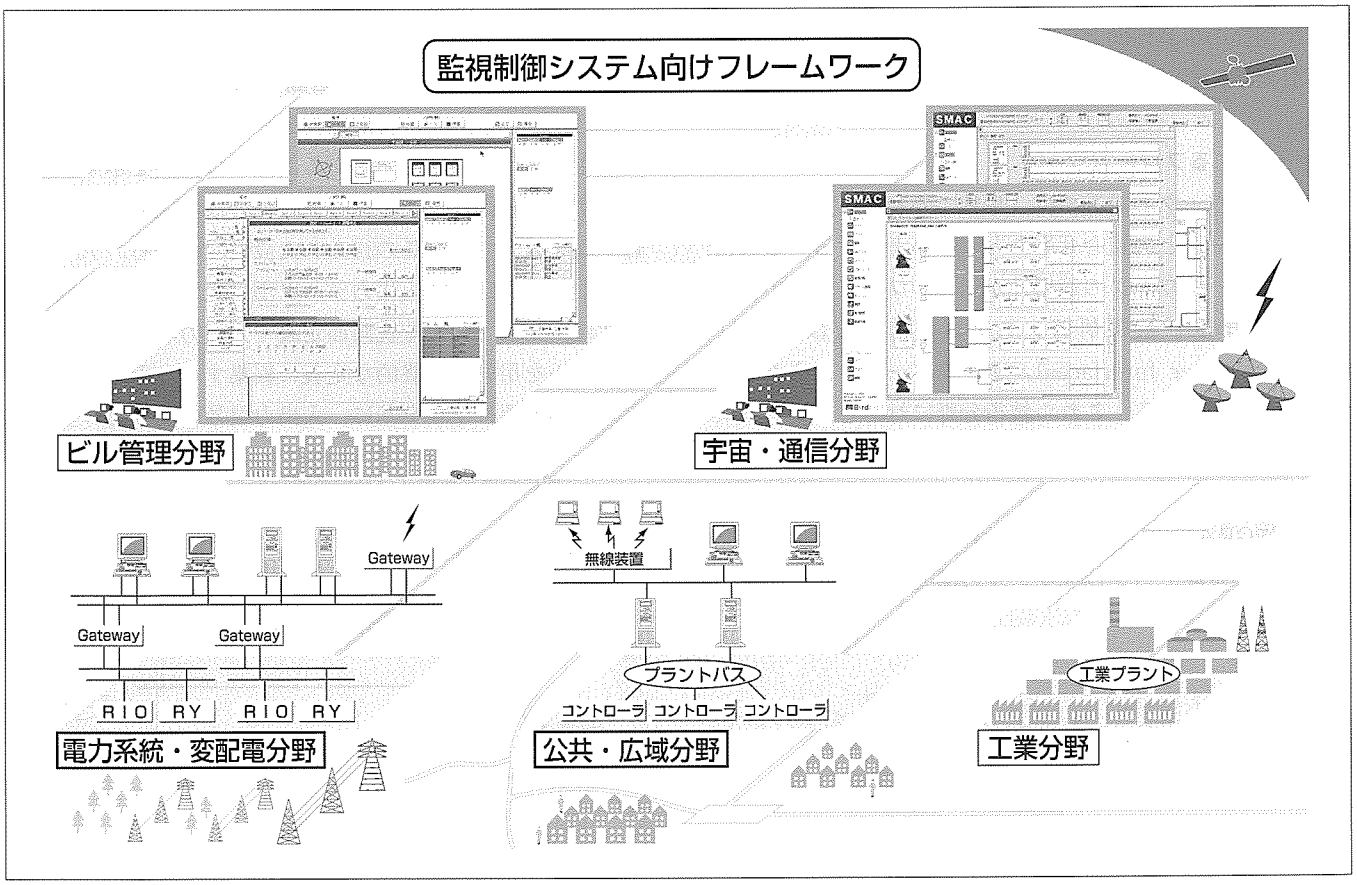
小島泰三* 山口 崇†
野里貴仁**
千葉裕二***

要 旨

監視制御技術は，産業分野の基盤を構成する技術である。例えば，電力システムや公共プラント等においては，その設備の状態を監視し必要に応じて制御機器を遠隔操作することで，システムの健全なサービスを維持する。また，工業プラントでは，シーケンサや工業計算機により生産ラインを管理する。さらに，近年，インテリジェント機器をつないだホームネットワークが提案されているが，その実現には，監視制御の技術が必要となる。

三菱電機では，電力，公共，工業分野等，様々な分野において，様々な種類の監視制御システムを供している。近年，システム機能の高度化によりソフトウェアが複雑化する一方，開発期間が短くなっており，そのソフトウェア開発の生産性が問題となっている。監視制御システムのソフトウェア生産性向上を目的に，監視制御システム向けの分野対応フレームワークの開発を行った。分野対応フレームワークは，単なる再利用可能なライブラリモジュールの集まりではない。対象分野のソフトウェアをどのように構成するかといった分析設計の結果を，ライブラリやツール群に加え，利用規則のような形式で規定することで，再利用可能な形にまとめあげた枠組みである。

本稿では，フレームワーク開発の遷移と適用例を紹介することで，各種分野に対応できる監視制御向けのフレームワークと，それにより達成したソフトウェア再利用について示す。



監視制御システム向けフレームワークの各種分野への展開

監視制御システム向けフレームワークは，監視制御システムのソフトウェアに関する分析・設計をプログラムコードやツール及び開発手順の形で再利用可能としたソフトウェア開発の枠組みの一種である。監視制御システムの機能は，様々な分野で導入されている。そのソフトウェア開発の枠組みを規定することで，ソフトウェア開発の生産性を向上させる。

*先端技術総合研究所(工博) **同研究所 ***稲沢製作所 †通信機製作所

1. まえがき

監視制御システム向けフレームワークは，監視制御システムのソフトウェアに関する分析・設計の結果をプログラムコードやツール及び開発手順の形で再利用可能としたソフトウェア開発の枠組みの一種である。フレームワークとしてソフトウェア開発に枠組みを規定することでソフトウェア開発の生産性を向上させる。

本稿では，当社で開発している監視制御システム向けフレームワークの一つとして，GRASS(Generation based Reconfigurable Architecture for SCADA Systems)フレームワーク[1]について概要を説明する。また，このフレームワークを用いて開発している分野対応システムを紹介する。

2. 監視制御システム向けフレームワーク“GRASS”

GRASSは，分散型の監視制御システムを対象とし，C++言語で記述したクラスライブラリ，システムを構成する標準のプロセス群，及び，それらプロセスの動作を定義するツール群からなる再利用可能なソフトウェアシステムである。GRASSの設計では，様々な対象分野に対して適用できること，また，同一分野での個別システムごとのソフトウェア開発の生産性向上も目標に設定した。以下では，これら目標がGRASSにおいてどう達成されているかを説明する。

2.1 成長するフレームワーク

フレームワークの設計では，単独での利用に加え，他のフレームワークとの組合せもあらかじめ考慮する。これにより，フレームワークの適用性が増し，ソフトウェア再利用性が高まる。監視制御システムの構築では，GUI(Graphical User Interfaces)，データベース等，様々な実装技術が用いられる。これら技術分野に対応したフレームワークが存在し，これらの組み合わせ方もフレームワークとなる。

図1にGRASS開発の変遷を示す。まず，監視制御オフライン系の定義データ作成ツールの開発において，GUIフレームワークの開発が始まった(GhostHouse：1991)。次に，公共プラント向け監視制御システムにおいて，GhostHouseをベースに，オンライン系のGUIをフレームワーク化した(SEVEN：1995)。そして，電力システム向け開発において，GUIに加え，データサーバと，そのエンジニアリングツールのフレームワーク化に着手した(GRASS：1996)。その後，工業プラント，昇降機監視と実用化し，現在，衛星追跡管制，衛星通信管理での実用化開発を行っている。これと並行し，既に実用化していたシステムをWebブラウザを用いたGUIで拡張し(DOOARS：1997)[2]，さらに，Java(注1)機能の付いた携帯電話によるユーザーインタフェースの開発も進行中である(MILK：2002)[3]。

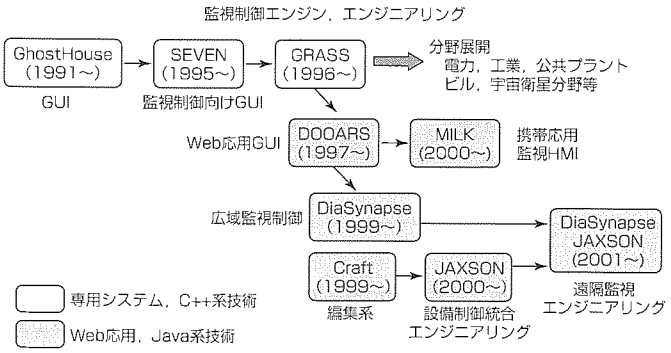


図1．フレームワーク開発の変遷

フレームワーク開発を継続的に行うことで，そのソフトウェア再利用性は段階的に向上する。一連の実用化開発では，フレームワークの分野に対する依存部／非依存部を見直すことで，フレームワークを整理した。その際，先の適用において依存部として開発していた部分について，一般化できることが明らかとなる。これがフィードバックされ，他の分野に適用する際に，その部分が再利用される。また，これまでの適用では実現されていなかった機能が追加されることがある。その際，一般化して実装すれば，既開発システムに対しても容易に導入できる。これにより，フレームワークの再利用が進む。

2.2 システム定義データの自動生成・カスタマイズ

GRASSの特長は，設備定義を基とした諸定義データの自動生成とカスタマイズにある。図2に概要を示す。

まず，回路図やシステム構成図等の設備図面を用いて設備定義を入力する。この得られた設備定義から，デフォルトの監視画面定義や入出力データ項目定義を自動生成する。

次に，生成された監視画面定義や入出力項目定義を，編集ツールを用いて最終的な定義に編集する。そして，得られた入出力項目定義を基に，入出力信号の割り付け表や集

(注1) Javaは，米国Sun Microsystems，Inc.の米国及びその他の国における商標又は登録商標である。

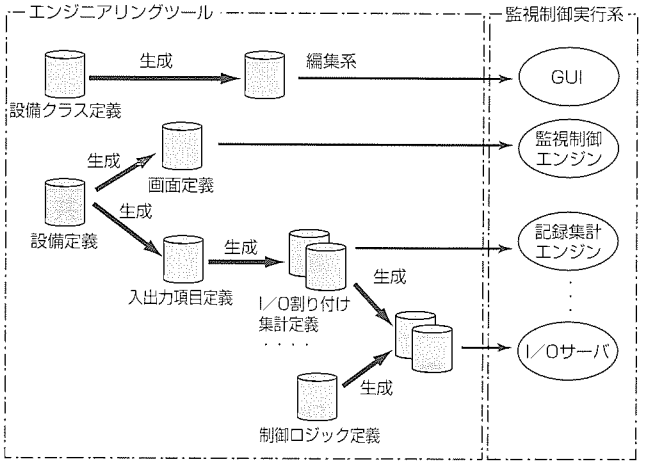


図2．システム定義データの自動生成

計定義，また入力信号に対してアラームを発する等の監視処理定義を生成する。監視制御システムを対象システムごとに適合させるための作業量を最小化するとともに，自動生成により誤りの混入をも最小化する。

2.3 実行系アーキテクチャ

実行可能なデータ構造やスクリプト言語を外部から与えることで動的な振る舞いをカスタマイズできるソフトウェアを一般的にエンジンと呼ぶ。GRASSでは，カスタマイズ性を向上させるため，様々なソフトウェアをエンジン化している。図3に，GRASSを用いた監視制御システムの基本的なプロセス構成例を示す。

図において，監視制御対象の設備の情報は，I／Oサーバ経由で，監視制御エンジンに入力される。監視制御エンジンでは，値の範囲チェックや事故判定等の処理が実行され，その結果はGUIプロセスや他の応用プログラムに渡される。また，監視制御エンジンでは，状態変化をトリガーとして制御スクリプトが実行される。GRASSでは，監視制御処理を定義データにより柔軟にカスタマイズするため，HGSと呼ぶスクリプト言語を開発した。そして，監視対象の設備のクラス定義における処理の記述，図的編集系を用いて入力する制御シーケンスや分野特化の特殊な事故判定処理をこのスクリプト言語のプログラムに変換し実行することで，各種分野に対する適応性を高めた。

3. 適 用 例

3.1 昇降機統合監視システム“MELAMS”

昇降機統合監視システムMELAMS(注2)は，大規模ビル向けのエレベーター，エスカレーターの監視システムである。図4にMELAMSのシステム構成を示す。図のように，このシステムは，監視端末，サーバ，昇降機との通信をつかさどる群管理モニタリング(GM)が汎用LANでつながる。

(注2) MELAMSは，三菱電機㈱の登録商標である。

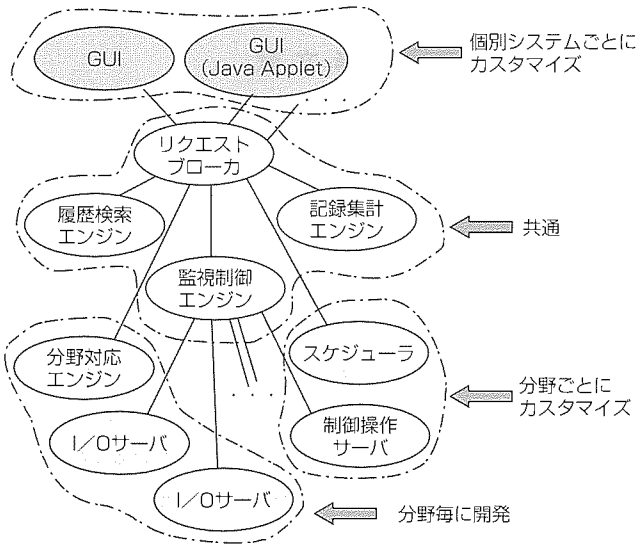


図3．実行系プロセス構成例

昇降機の情報は，群管理コントローラ(GC)からGMを介して取得する。ユーザーインタフェース機能としては，エレベーターの位置やかごの状態を示す立面図や平面図，故障等の発生状態を示すアラーム一覧，各種制御パラメータや運転モードを設定するための制御画面等を提供する。図5に画面例を示す。

MELAMSの開発に当たっては，昇降機分野特化の開発として，群管理コントローラとの通信機能，昇降機監視用のデータ生成ツール，MELAMS特有の設定処理を行う専用のソフトウェア部品を新たに開発した。一方，監視制御エンジンや履歴検索エンジン等については，分野依存処理は，すべてスクリプト言語で吸収できている。

システム定義データの作成では，GRASSの標準方式を，MELAMS向けに拡張している。MELAMS内部では，エレベーターのかご，バンク，フロア等を設備としてモデル化している。個別システムの構築では，昇降機の台数やビルの階床数等の情報から，画面定義，設備定義等を生成する。大規模なシステムの場合，顧客の好みや特殊仕様等により各種カスタマイズが必要となるが，標準構成のシステ

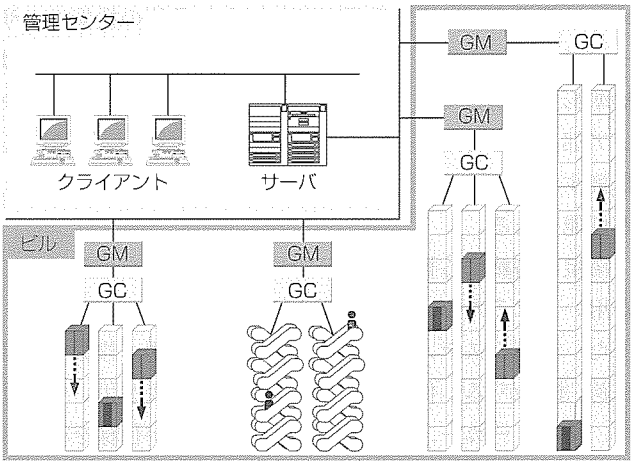


図4．昇降機統合監視システムの構成

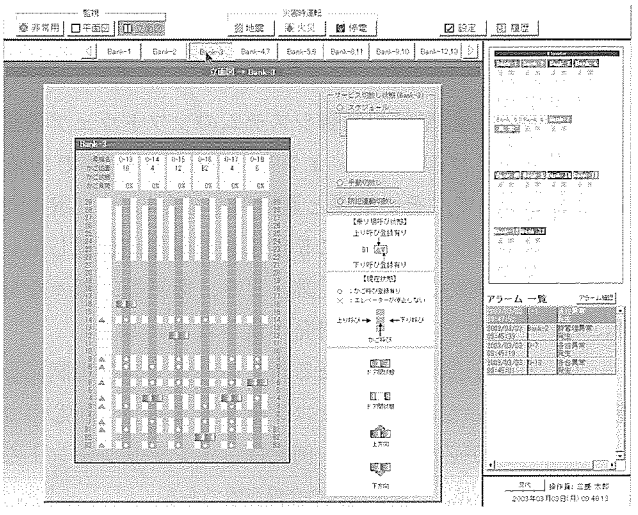


図5．監視画面例(MELAMS)

ムの場合には，生成後の編集作業は発生しない。

3.2 衛星追跡管制システム“BirdStar”

衛星追跡管制システムBirdStar(注3)は，人工衛星の監視，姿勢や軌道等の制御操作を行う地上局システムのソフトウェア基盤として開発中のシステムである。このシステムは，監視制御システム向けフレームワークを分野向けにカスタマイズしたものとして位置付けられており，個別のシステム開発は，BirdStarに対して，システムごとに要求される機能を付加することで達成する。**図6**にBirdStarのシステム構成を示す。図において，衛星管制装置，衛星データ解析装置，地上局監視制御装置の開発にはGRASSの監視制御機能，衛星運用計画装置にはGRASSに含まれるエンジニアリングツール作成のためのサブフレームワーク，そして，軌道運用解析装置にはGRASSで利用しているGUIフレームワークを利用している。

図7に衛星管制装置の画面例を示す。**図5**と比較して分かるように，画面レイアウトや画面に表示される内容は，分野に依存するものの，個々の表示部品は共通のものが用いられている。適用対象分野が異なっても，システムの目的は，設備の状態監視と制御であり，同様である。このため，設備構成図による監視やイベント一覧等の機能は，サーバでの内部処理は同様であり，また，表示方法の大枠は，標準形式を用意できる。GRASSではこれを標準ソフトウェア部品として用意してあり，この仕様に従いシステムを構築すれば，適用分野に左右されないソフトウェア再利用が可能となる。

衛星に搭載する機器は，目的に応じて様々な種類のものが用いられるため，設備情報を標準化することが難しい。また，衛星システムの開発は一社では閉じず，複数の会社に跨(またが)る。例えば，テレメトリ項目表を用いて，顧客，衛星メーカー，管制システムメーカーが仕様をやり取りする。このため，GRASSの適用においては，個々のテレメトリ情報を単独の機器と見なし，これからGRASSに必要な定義データを生成することで対応した。

4. むすび

本稿では，分散型監視制御システムのソフトウェア構築を支援するアプリケーションフレームワークであるGRASSについて，その概要と適用例を紹介した。GRASSは，開発に着手してから既に5年以上の歳月が経過している。その間，電力，公共，工業，ビル，衛星等，様々な分野において実用システムの開発へ供されている。

一方，利用形態が全く異なる新しい分野での監視制御機能の利用については，新たなコンセプトに基づく新しいフレームワークが必要となる。例えば，インテリジェントな組込み機器を活用するシステムでは，従来の監視制御システムとは異なるエンジニアリング手法が必要となる(4)。また，セキュリティシステムでは，監視対象が固定的な設備ではなく，異なるモデリングが必要となる。今後は，これら新しい応用について検討し，フレームワークの開発を行っていく予定である。

(注3) BirdStarは，三菱電機㈱の登録商標である。

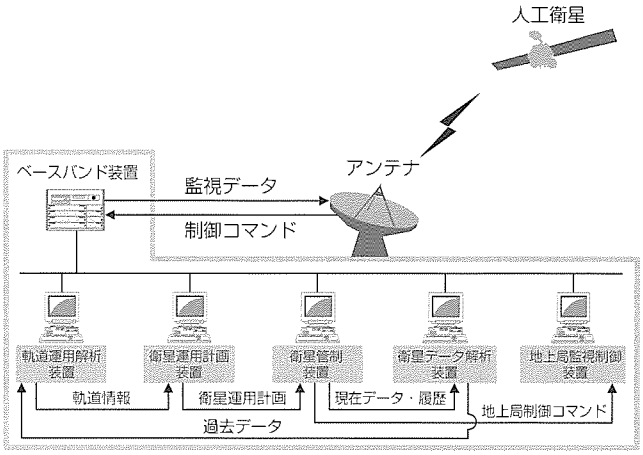


図6．衛星追跡管制システムの構成

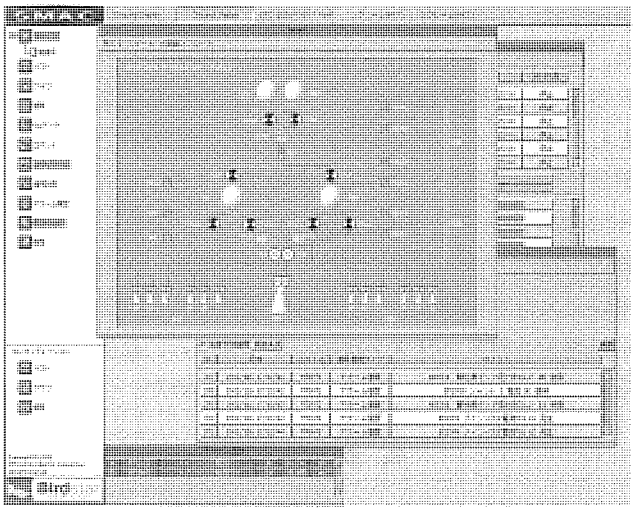


図7．監視画面例(BirdStar)

参考文献

(1) 小島泰三，ほか：監視制御システム向けアプリケーションフレームワーク，電気学会論文誌(C)，**119**，No.10，1274～1282（1999）

(2) 佐藤正行，ほか：公共プラントWeb応用監視制御システム，三菱電機技報，**74**，No.12，735～738（2000）

(3) 北村操代，ほか：シンクライアント方式による携帯応用監視制御システムの実現，情報処理学会研究報告，2003-MBL-24，2003，No.21，45～52（2003）

(4) 石原　鑑，ほか：インターネット応用監視制御フレームワーク“DiaSynapse/JAXSON”，三菱電機技報，**76**，No.9，604～608（2002）

基幹業務システム開発における ソフトウェア再利用技術への取り組み

上野浩一郎* 阿波道雄**
倉持和彦* 浦井哲哉***
熊井秀憲**

要　旨

近年，企業活動は情報システムへの依存度を高め，情報システムが大規模化している。このような大規模システムを開発するには，一般に次のリスクがある。①開発作業量が膨大なため開発期間が長期化。②開発者が多数なためコミュニケーションギャップが拡大。③多様な開発者が参加するためプロジェクト内で技術スキルが均質でない。

他方，大規模開発では大量のアプリケーション間で類似部分が多い傾向があるため，標準化・共通化してプロジェクト内でソフトウェア再利用を図れば，開発の品質と効率を高められる側面もある。

そこで三菱電機は，基幹業務システムという大規模開発において，上記3つのリスク低減と再利用促進を目的に以下の取り組みを実施している。

(1)　ソフトウェアアーキテクチャ主導の開発

ソフトウェアアーキテクチャは開発対象構造の概観であり，この設計で共通部分を明確化して重複開発を抑止する。

(2)　ソフトウェアフレームワークの再利用

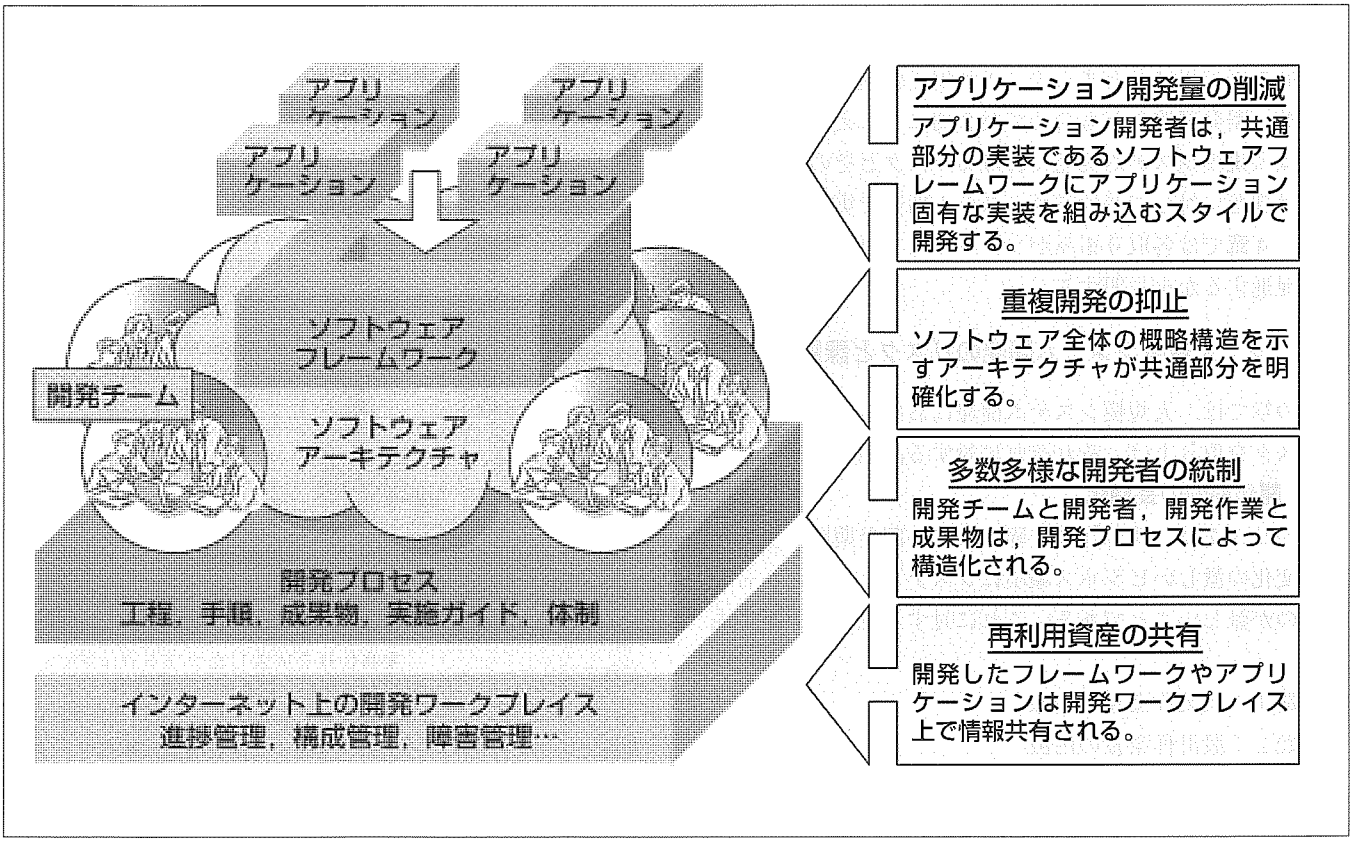
ソフトウェアフレームワークは共通部分の実装であり，これにアプリケーション固有な実装を組み込むスタイルによりアプリケーション開発量を削減する。

(3)　開発プロセスの標準化

アーキテクチャを反映した具体的な開発プロセスにより，多数多様な開発者を統制した再利用開発を実現する。

(4)　開発ワークプレイスの運営

多拠点に分散する多数の開発者向けにインターネット上で構成管理サービス等を提供して，再利用資産を共有する。



ソフトウェア再利用技術への取り組み

①ソフトウェアアーキテクチャ“開発者間で開発対象を共通認識するためにソフトウェア構造を概観したもの”，②ソフトウェアフレームワーク“あるソフトウェア群に共通する骨組みを実装したソフトウェア”，③開発プロセス“開発上の工程／手順／成果物／体制等を定義したドキュメント”，④開発ワークプレイス“システム開発に必要な進捗（しんちょく）管理や構成管理などのサービスをインターネット上で提供する環境”の取り組みを実施している。

*三菱電機㈱ 情報技術総合研究所　**同社 社会eソリューション事業所　***三菱電機インフォメーションシステムズ㈱

1. まえがき

近年，社会や企業の活動は情報システムへの依存度をますます高め，このため，情報システムが大規模化する傾向にある。三菱電機は，業務系やエンジニアリング系を含めて様々な大規模システムを開発しており，筆者等はクライアント／サーバによる基幹業務システムを開発している。このシステム開発は，実現性検討を含め期間約4年間，ピーク時要員約100人以上の規模である。このような大規模システム開発は，小規模システム開発と比較すると①開発作業量が膨大，②多数の開発者が参加，③多様な開発者が参加という特徴があり，一般にその成功率は低くなる。

他方，大規模システム開発では大量のアプリケーション間で類似部分が多い傾向があるため，これらを標準化・共通化してプロジェクト内でソフトウェア再利用を図れば，開発の品質と効率を高められるという側面もある。システム開発規模に従い，ソフトウェア再利用による削減コストを大きくできる可能性がある。

以上により，大規模システム開発では，上記で指摘した3つのリスクを低減しつつソフトウェア再利用を促進することが求められている。リスクを低減する取り組みにより開発成果物／開発作業／開発環境の標準化を達成し，これによりソフトウェア再利用の土台を確立できる。なお，本稿では，ソフトウェア再利用技術を，プログラムコードの直接的な再利用にとどまらず，再利用を促進する設計，開発手順，開発環境なども含めて総合的にとらえている。**2章**では大規模システム開発におけるリスクとその課題，**3章**では課題に対して基幹業務システム開発で実施した取り組み，**4章**では各取り組みがソフトウェア再利用をどのように促進するかを説明する。

2. 大規模システム開発のリスクと課題

この章では，大規模システム開発における以下の代表的なリスクを取り上げ，それぞれに対する課題を説明する。

2.1 開発期間の長期化

このリスクは，開発作業が膨大なため開発期間が長期化し，変化の激しいビジネス環境にシステムリリースを合わせるのが難しいことである。これに対する課題は以下である。

課題1：実装作業量の削減

課題2：設計作業量の削減

上記の課題に対する取り組みは**3.2節**で説明する。

2.2 開発者間のコミュニケーションギャップ

このリスクは，参加開発者が多数なためコミュニケーションギャップが拡大することである。これに対する課題は以下である。

課題3：開発作業を分担できるような開発対象の分割

課題4：開発者間でソフトウェア全体イメージを共有

課題3と4に対する取り組みは**3.1節**で説明する。

課題5：他チームへの作業依存性が低いチーム構成

課題5に対する取り組みは**3.3節**で説明する。

課題6：多拠点に分散した開発者間での成果物共有

課題6に対する取り組みは**3.4節**で説明する。

2.3 プロジェクト全体で技術スキルが均質でない

このリスクは，技術力や専門分野が多様な開発者が多数参加し，プロジェクト全体で技術スキルが均質でないことである。これに対する課題は以下である。

課題7：開発者に左右されない成果物品質の確保

課題8：スキルを考慮したチーム員構成

上記の課題に対する取り組みは**3.3節**で説明する。

3. 大規模システム開発に対する取り組み

この章では，前章で指摘した課題に対して基幹業務システム開発で実施した取り組みを説明する。

3.1 ソフトウェアアーキテクチャ主導の開発

ソフトウェアアーキテクチャとは，開発者間で開発対象を共通認識するためにソフトウェア構造の概観を示したドキュメントである。例えば**図1**と**表1**は，筆者等が取り組んでいる基幹業務システム開発におけるソフトウェア群を“層”の視点で示したものである（フレームワークについては**3.2節**参照）。**図1**と**表1**は，ソフトウェアが各々の責務を持った5個の層で構成されていることを示している。

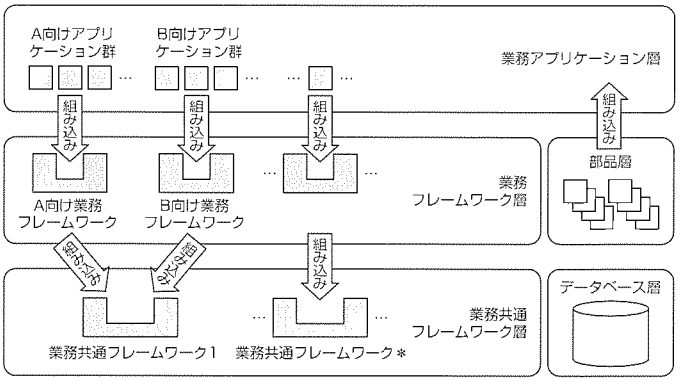


図1．ソフトウェアの層構成

表1．各層のソフトウェア説明

業務アプリケーション	業務仕様を実装したソフトウェア。業務ごとに画面と帳票データを持つ。
業務フレームワーク	共通化した業務仕様を実装したソフトウェア。共通する画面構造や帳票データを提供する。
業務共通フレームワーク	標準化したシステム機能を実装したソフトウェア。データベースへのアクセス機構や画面管理機構を提供する。
部　品	頻出する小粒度の汎用機能を実装したソフトウェア。必要に応じて業務アプリケーションに組み込まれる。
データベース	業務データを格納するデータベース。

このような層分割は，大規模なシステムを開発可能な単位に適切に分割している(課題3の解決)。さらに，開発者間でシステムを議論する際，その議論対象のソフトウェア層を明らかにしてその責務を明確化することにより，ソフトウェア全体イメージを共有できる(課題4の解決)。また，このアーキテクチャは，後述するフレームワークと開発プロセスの土台に位置付けられる。

3.2 ソフトウェアフレームワークの再利用

ソフトウェアフレームワークとは，ある層で開発しなければならない多数のソフトウェア群に共通する骨組みを実装した再利用可能なソフトウェアである。例えば**図1**の業務フレームワークは，多数の業務アプリケーション群をグループ化してそれらに共通する骨組みを実装したものであり，業務にかかわる画面構造や帳票データなどを提供している。業務アプリケーションの開発者は，自分が開発するアプリケーションに特有な仕様だけを実装して，それを業務フレームワークに組み込むことになる。また**図1**の業務共通フレームワークは，データベースへのアクセス機構や画面管理機構などのシステム機能を標準化し，これらをMVC(Model，View，Controller)構造に基づいて提供している(**図2**)。MVC構造は，業務アプリケーションを①データとその処理を行うModel，②Modelを画面表示するView，③ユーザー操作の応答制御を行うControllerの3つに分割することにより，ユーザーインタフェースを柔軟に変更可能とする。

このような各種フレームワークに基づく開発スタイルは，ソフトウェアの骨組みに該当する実装コードを再利用している(課題1の解決)。また，フレームワークの利用に際しては，専用の支援ツール群を準備することにより，実装作業の負荷を軽減している。さらに，ソフトウェアの基本的な骨組み，例えば**図2**に示すようなMVC構造をフレームワークが設計済みの形で提供するので，業務アプリケーション開発者はその骨組みに準拠することにより設計結果も再利用していることになる(課題2の解決)。

3.3 開発プロセスの標準化

開発プロセスとは，開発上の工程／手順／成果物／実施ガイド／体制などを定義したドキュメントである。システム一般に対する開発プロセスの業界標準として“統一プロセス”[1]がある。これは様々なシステム開発に対応可能なように，工程は網羅的，手順は汎用的に記述されている。このため各プロジェクトへの適合性は低い。そこで今回，大規模開発に適用可能なように工程を取捨選択するとともに，ソフトウェアアーキテクチャで定義した2つのフレームワークの再利用を前提にした手順を具体化した。プロジェクトに最適化したこのような具体的な開発プロセスにより，多様な開発者の成果物品質を一定基準以上に確保することができる(課題7の解決)。

また**図3**は，この開発プロセスが定義するチームとチーム員の構成方法を説明している。この開発プロセスは，ソフトウェアアーキテクチャが定義する5層(**図1**参照)を反映したチームを定義し，さらに，これらのチーム間をまたがって調整するアーキテクチャチームと開発環境支援チームを定義している。各チームの依存性は，自らの下位層に対応するチームに限定されているので独立性の高いチーム構成となっている(課題5の解決)。そして，各チームごとに工程／手順／成果物／実施ガイド／役割一覧を定義している。役割一覧とは，そのチームで必要な役割(例えばアーキテクト，設計者，実装担当者等)を列挙したものである。この役割には必要な技術スキル群(例えばオブジェクト指向設計技術，データベース実装技術等)を定義しているため，スキルを考慮してチーム員を構成可能である(課題8の解決)。

3.4 開発ワークプレイスの運営

開発ワークプレイスは，システム開発に必要な進捗管理，構成管理，トラブルチケット管理，障害管理，それらを統

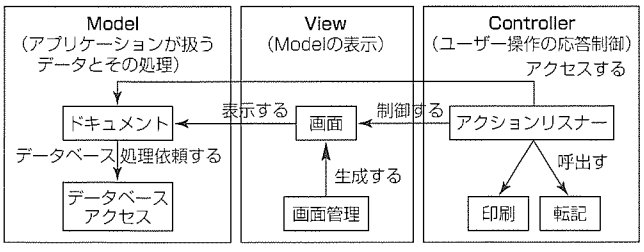


図2．業務共通フレームワークが提供するModel，View，Controller構造

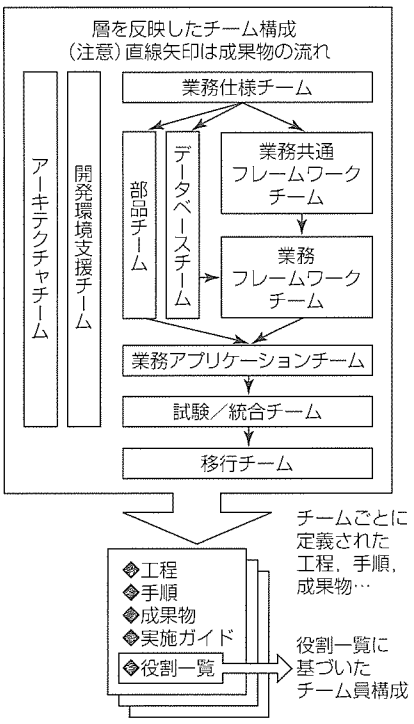


図3．開発プロセスが定義するチームとチーム員の構成方法

合するポータルのサービスをインターネット上でWebにより提供する開発支援環境である[2]。大規模システム開発では多数の開発者が各地の拠点に分散し，さらに，作業によっては拠点を移動しなければならない。このような状況でも，各開発者はブラウザでプロジェクトの全開発情報にアクセスすることが可能である(課題6の解決)。例えば構成管理を担当する三菱電機製品DEPOSIT(注1)[3]は，ソフトウェアや仕様書をインターネット上で一元管理しつつ，各工程とユーザーごとに作業空間を提供して分散開発を実現している(図4)。

なお，同様なサービスを提供するインターネットサイトとして，オープンソース開発向けのSourceForge.jp[4]がある。ただしこれは，オープンソース向けなので全開発情報をインターネット上に積極的に公開しており，企業での開発スタイルとは異なっている。特に企業における大規模システム開発では，顧客／主契約者／ソフトウェア外部委託業者という様々な役割を持った人々がアクセスするので，それらのアクセス制御が特に重要である。

3.5 各取り組みの関係

この節では，上述した4つの取り組みの相互関係について説明する。ソフトウェアアーキテクチャは全開発者が共有すべきであり，すべての取り組みの前提である。しかし，ソフトウェアアーキテクチャは，概念を表したドキュメントに過ぎない。そこで，ソフトウェアアーキテクチャに則って共通部分のソフトウェア実装を与えるものがソフトウェアフレームワークである。ただソフトウェアフレームワークはソフトウェアであり，それを利用した開発方法を示唆するものではない。そのため，開発者又は開発チームが作業する前に，利用するソフトウェアフレームワークに基づいた開発プロセスが必要となる。そして，実際に開発プロセスを実践して分散したチーム間で協調作業するには，インターネット上での開発ワークプレイスが必要になる。

4. むすび

本稿では今回実施した取り組みを紹介した。最後に，これらの取り組みが再利用を促進することを説明する。①ソフトウェアアーキテクチャは，全体的な共通部分を明確化して，開発チーム間の重複開発を抑止する。②ソフトウェアフレームワークは，共通部分の実装を提供し，それにアプリケーション固有な実装を組み込む開発スタイルにより，アプリケーション開発量を削減する。③開発プロセスは，プロジェクト固有なアーキテクチャに基づいて，多数多様な開発者を統制した再利用開発を実現する。④開発ワークプレイスは，多拠点に分散する多数の開発者向けにインターネット上で構成管理サービス等を提供して，プロジェクト全体で再利用資産を共有する。

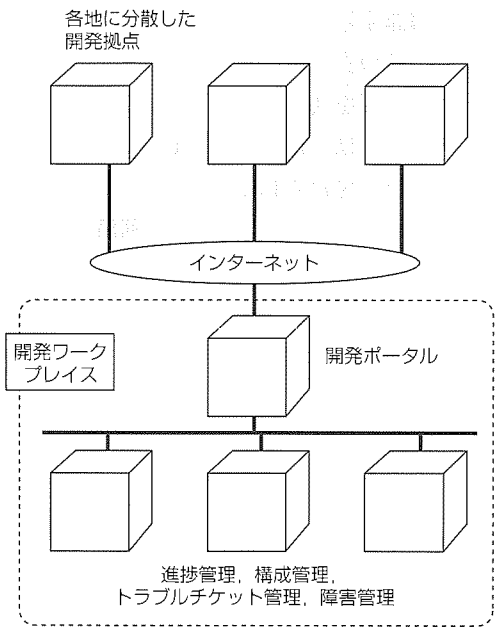


図4．開発ワークプレイス

なお本稿では言及しなかったが，リスク低減を目的にウォーターフォール型開発ではない“反復型開発”，設計や実装だけでなく仕様の再利用促進を目的に“ユースケース駆動”も取り組み中である。今後も，大規模システム開発に対する総合的な取り組みを継続していく予定である。

(注1) DEPOSITは，三菱電機㈱の登録商標である。

参考文献

(1) Jacobson，I.，ほか：UMLによる統一ソフトウェア開発プロセス，翔泳社（2000）
(2) 電波新聞（23 Oct 2001）：電力会社向けの情報システム構築，ポータルサイトで一元管理—三菱電機，http://www.shimbun.denki.or.jp/backno/01102303.html
(3) ソフトウェア部品庫DEPOSIT：http://www.MitsubishiElectric.co.jp/service/deposit/
(4) SourceForge.jpのURL：http://sourceforge.jp/

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

原田雅史* 鷲津 忍**
松田昇平* 武井篤志**
土屋 隆*

要 旨

近年，プラットフォームに依存しないJava(注1)技術の進展，Webアプリケーションサーバミドルウェアの出現などにより，Webコンピューティング技術は，インターネット／イントラネット上でWebサーバを用いて情報発信や情報共有を主体としたシステムから消費者向けEC(Electronic Commerce)システム，さらにはミッションクリティカルな基幹系システムにまで適用されるようになった。

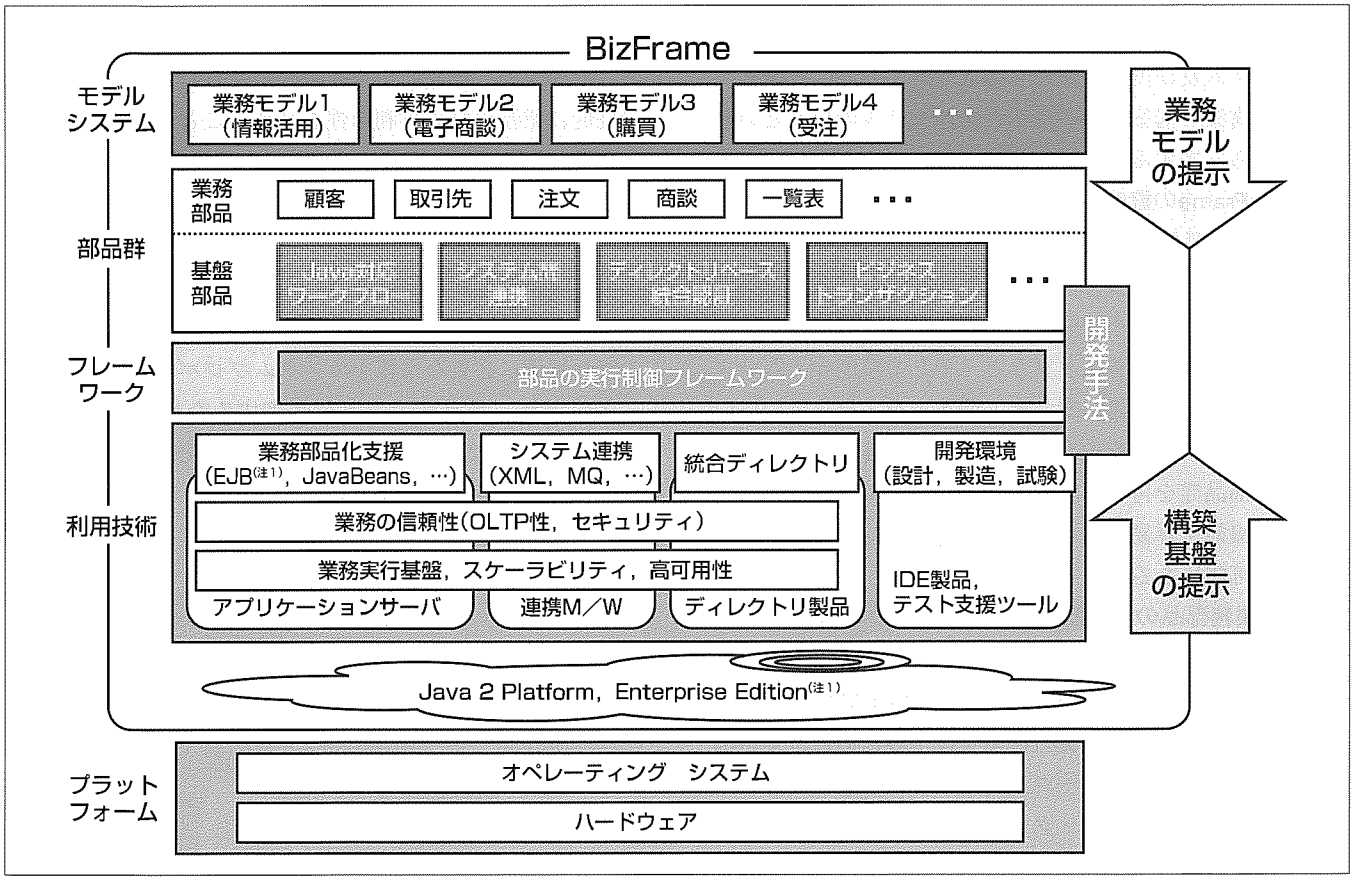
このようなWebアプリケーションを開発する際の課題として，一般に次の項目が指摘されている。

(1) 開発期間が概して短い。
(2) Java，Webアプリケーションサーバなどに関する経験豊富な技術者の確保が難しい。
(3) 設計，開発にオブジェクト指向技術が必要とされる。
(4) システムの要件が固まらない，又は変更が頻繁に行われる。

今回，これらの課題を解決するために，開発言語にJava，ミドルウェアとしてWebアプリケーションサーバを使用する3階層システムの構築を，MVC(Model，View，Controller)アーキテクチャを実現することにより支援するアプリケーションフレームワーク“BizFrame(注2)”を開発した。

本稿では，BizFrameの概要と主な機能，BizFrameの基盤である部品の実行制御用フレームワークの構成と動作概要，及びBizFrameを用いた開発プロセス，そして最後にBizFrameの適用事例について紹介する。

(注1) Java，EJB(Enterprise Java Beans)，J2EE(Java 2 Platform，Enterprise Edition)は，米国Sun Microsystems，Inc.の米国及びその他の国における商標又は登録商標である。
(注2) BizFrameは，三菱電機㈱の登録商標である。



アプリケーションフレームワークBizFrameの構成と位置付け

Webアプリケーションを構築するためには，様々な新しい技術要素が必要である。J2EE(Java 2 Platform，Enterprise Edition)に対応したミドルウェアを使用することで，Webアプリケーションはプラットフォームから独立になる。現在，アプリケーションフレームワークBizFrameを開発し，構築基盤の提示と業務モデルの提示という両方向からWebコンピューティングに取り組んでいる。

*三菱電機㈱情報技術総合研究所 **三菱電機インフォメーションシステムズ㈱

1. まえがき

BizFrameは，特定のビジネスドメインに依存せず，開発言語にJavaやWebアプリケーションサーバを使用した3階層（プレゼンテーション層，業務ロジック層，データベース層から構成）システムの構築を，汎用的に支援するアプリケーションフレームワークである。

BizFrameを開発した背景は，三菱電機及び関連会社のシステム構築案件が従来のクライアント／サーバシステムに代表される2階層システムからJavaやWebアプリケーションサーバを使用した3階層システムへ移行しつつあり，短期間で高品質な3階層の業務システムを構築する必要があったためである。

既に，BizFrameは三菱電機及び関連会社の複数プロジェクトで実際に適用され，生産性や品質の向上に寄与している。

2. BizFrameの概要と主な機能

2.1 BizFrameの概要

BizFrameは，WebアプリケーションサーバなどのISV（Independent Software Vendor）製品に関する利用技術，基盤部品や業務部品などの部品群，それらを実行制御するフレームワーク，業務システムの雛型（ひながた）であるモデルシステム及び開発手法から構成されており，3階層システムの構築を構築基盤の提示と業務モデルの提示という両方向から支援することが可能である。

2.2 BizFrameの機能

2.2.1 各種部品群の提供

BizFrameは，基盤部品としてアプリケーションで共通に利用可能で粒度の小さいログ出力，データベースアクセス，画面の遷移順序チェックなど，そして粒度の大きいワークフロー，統合認証，システム間連携などを提供している。また，事業部側を中心に業務部品の整備を行っている。これらの部品群を再利用することでシステム構築に要求される基盤機能を容易に実現でき，生産性や品質の向上を図ることが可能になる。

2.2.2 業務部品の実行制御フレームワークの提供

BizFrameでは，業務部品などのいわゆるソフトウェア部品を開発し，それらを組み合わせてWebアプリケーションを開発する。この開発方法を支援するために，業務部品の実行制御フレームワークを提供している。このフレームワークは，業務オブジェクト（例えば，注文伝票など）が業務フローに沿って状態が変化していく点（起票→上長承認済み→提出完，など）に着目し，それの状態遷移に基づいて業務部品の実行制御を行う。つまり，業務フローに従って業務部品の振る舞いをプログラムコードで記述するのではなく，あらかじめ定義したテーブルを参照しながら，業務部品を実行制御することで，それらの再利用性とアプリケーションの仕様変更への対応を高めるのがねらいである。さらに，次の2つの機能を持っている。

(1) MVCアーキテクチャの実現

実行制御フレームワークでは，MVCアーキテクチャに沿った3階層システムを実現する。MVCアーキテクチャは，GUI（Graphical User Interface）を持つアプリケーションを開発するためのデザインパターンである。このパターンは，Model（業務ロジック），View（画面），Controller（制御部）を分離することで，Modelの独立性を高め，ViewやControllerの変更に対してModelへの影響度を小さくすることを目的としている。そのメリットは，次のとおりである。

- Modelの再利用性が高まる。
- Model，View，Controllerの並行開発が可能になり，生産性向上を図ることができる。

(2) EJB（Enterprise Java Beans）コンテナによるトランザクション制御

実行制御フレームワークでは，トランザクション管理をEJBコンテナに委譲し，CMT（Container Managed Transaction）機能を用いてトランザクション管理を行っている。この方式は，トランザクション境界（commit，rollback）の指定を業務部品のプログラムコード中に記述する必要がない。そのため，部品同士を組み合わせる自由度が高くなり，最終的に業務部品の再利用性を高めることが可能になる。

2.2.3 業務フロー定義用のGUIツールの提供

BizFrameでは，Microsoft社のテクニカル ドローイングツールである“Visio(注3)”を用いた業務フロー定義用GUIツールを提供している。このツールで業務オブジェクトの状態遷移に着目した業務フローを定義し，さらに，業務部品の実行制御に必要な情報をカスタムプロパティに記述することにより，業務部品を実行制御するためのテーブルを自動的に生成できる。状態遷移図の表記方法は，UML（Unified Modeling Language）のステートチャート図を採用している。このツールを用いた業務フローの定義例を図1に示す。

（注3） Visioは，米国Microsoft Corp.の登録商標である。

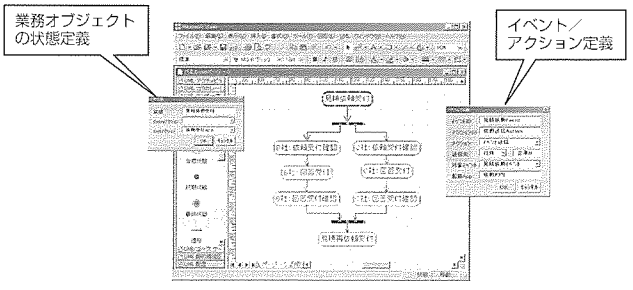


図1．GUIツールを使用した業務フローの定義例

3. 実行制御フレームワーク構成と動作概要

3.1 実行制御フレームワークの構成

BizFrameの中で業務部品などをテーブルに基づいて実行制御し業務アプリケーションをMVCアーキテクチャで実現する中核的な責務を果たすのが，実行制御フレームワークである。このフレームワークは，図2に示すように，業務アプリケーションを次の三つの部分から構成する。

(1) Model：業務ロジックをサーバ側の部品化技術であるEJBを用いて実装したJavaのクラス(業務部品)

(2) View：JSP(Java Server Pages)(注4)で作成した表示用画面ファイル

(3) Controller：入力データの解析，業務部品の実行制御，画面の出力制御などを行うサーブレット，EJB，Java BeansなどのJavaのクラス及び実行制御用テーブル

実行制御フレームワークでは，フレームワークが提供するクラスと開発者が作成するクラスを明確に分離することで，アプリケーション開発時の初期段階での立ち上がりを加速する。

3.2 実行制御フレームワークの動作概要

実行制御フレームワークは，図2に示すように，WWWブラウザからの要求に対して次のように動作する。

①，②，③：利用者がボタン押下など何らかのイベントを発生させることで，HTML(Hyper Text Markup Language)のフォーム形式の要求がWWWブラウザから送信される。メイン・サーブレットは，それを解析して，オブジェクトID，イベント種別などを取り出し，パラメータをデータ・テーブルに格納し，処理決定を呼び出す。

④：処理決定は，業務オブジェクトIDをキーにして，状態管理テーブルから検索した業務オブジェクトの状態とイベント種別をキーにして，処理決定テーブルから要求に対する処理名を決定する。

⑤，⑥：処理実行は，上記で決定された処理名をキーにして，処理定義テーブルから実行する業務ロジックを検索し，それをパラメータにして業務ロジックハンドラを呼び出す。

⑦，⑧，⑨：業務ロジックハンドラは，Modelである業務ロジックを実装したアクション部品を呼び出す。アクション部品は，業務データベースへのアクセスを実装したデータアクセス部品を呼び出す(注5)。業務ロジックの実行結果は，データ・テーブルに格納される。次に，業務ロジックハンドラは，状態管理を呼び出し，実行結果に応じて業務オブジェクトの状態を変更し，その情報をテーブルに書き込む。

⑩，⑪，⑫：最後に，メイン・サーブレットは画面出力制御を呼び出し，ViewであるJSPが実行され，WWWブラウザにHTMLで記述された結果を返す。表示に必要な処理の結果はデータ・テーブルから取得する。

なお，実行制御用テーブル間の関連を図3に記す。

(注4) JSP(Java Server Pages)は，米国Sun Microsystems, Inc.の米国及びその他の国における商標である。

(注5) 業務ロジックとデータアクセスの実装を分離することにより，部品の再利用性が高くなる。

4. BizFrameを用いた開発プロセス

BizFrameを適用して業務アプリケーションを開発する場合も，標準的な開発プロセスである要件定義→業務分析→設計→実装→試験というステップを踏む。この章では，BizFrameを用いた開発プロセスの特徴的なポイントについて述べる。

設計フェーズでは，開発する業務部品の再利用性を高め生産性を向上させるために，次の方針で業務ロジックの部品化を行う。

- 業務ロジックを共通化して部品化を行う。
- 業務ロジックを意味のある機能単位に分割する。
- 分割した機能単位の組合せで業務ロジックを実行する。

ここでの機能単位が業務部品の候補となる。具体的には，画面の入出力項目と業務データベースのデータ項目の対応を整理して，複数の画面に現れる処理を共通機能として切り出し部品化を行う。

次に，実行制御フレームワークは，実行制御用のテーブルに基づいて業務部品を呼び出し，業務ロジックを実行す

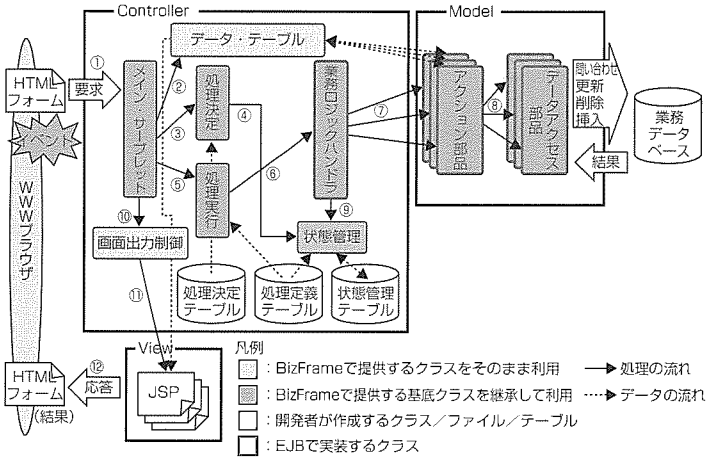


図2．部品の実行制御用フレームワークの構成と動作概要

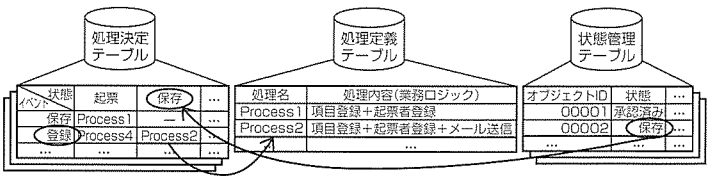


図3．実行制御用テーブルの関連

るため，設計フェーズでは，

- 業務オブジェクトの状態に着目した業務フローの定義
- 処理決定テーブル及び処理定義テーブルの作成

を行う。**2.2節**で述べたGUIツールを使用して業務フローを定義することによりこれらの実行制御用テーブルは自動生成することが可能なので，設計者の作業は前者に注力でき，作業負荷が軽減される。

実装フェーズでは，提供されている基盤部品のカスタマイズや設計フェーズで切り出した業務部品のJavaでの実装を行う。

5. BizFrameの適用事例

BizFrameは，電子商談システム“プロキオン”(Procurement Cyber Communication System)を始めWeb-EDIシステムなど多くのシステム構築に適用されている。この章では，プロキオンについて適用結果を紹介する。

5.1 電子商談システムプロキオンの概要

プロキオンの目的は，当社グループ内の資材調達業務に関して，取引先間での競合による資材原低，オンラインでの商談・見積り・情報提示により調達業務における情報の発信・取得・登録のスピードアップを図ることである。そのために，全社統一の取引先データベースを構築し，競合見積り機能，公文・案内機能，逆オークション機能を持ち，図面などの電子データ添付機能を備えている。システム構成を**図4**に示す。プロキオンでは，WWWブラウザ，電子メールなどを使用して，グループ内の設計部門，資材部門の担当者及び取引先の担当者の間で資材調達業務を行う。

5.2 BizFrameの適用評価

(1) 生産性

生産性を厳密に評価するためには，同じシステム構築に対してBizFrameを適用した場合と適用しなかった場合で比較を行う必要がある。しかし，現実のシステム構築においては不可能なため，特定の機能に関してBizFrameを使用せずにプロトタイプを開発し，その開発量から全体のボリュームを見積り，実際にBizFrameを適用した開発量との比較を行った。その結果，見積り予測42.0kLineに対し実際の開発量は30.0kLineで開発量が約28%削減され，生産性向上を実現した。また，システム設計からシステム試験まで5か月という短期間で完了させることができた。この要因は，フレームワークの適用により並行開発を行うことができ，業務部品設計時に共通化を進め，再利用性を高めたことの効果である。

(2) 仕様変更に対する柔軟性

開発を進めていく上で仕様変更は度々発生したが，新しい機能追加でない場合(例えば，承認者の追加など)，既存部品を組み替えるだけで対応可能であった。つまり，プログラムコードを修正するのではなく，処理定義表の修正だけで処置が完了した。

(3) その他

使用しているミドルウェア(Webアプリケーションサーバ，データベースなど)に依存した処理(注6)をBizFrameが提供する基底クラスで吸収することで，プログラマが開発したクラスで個別に対応する必要がなく，生産性，品質の向上に寄与した。

以上のことから，BizFrameがアプリケーションフレームワークとしての成果を十分に発揮していると考えられる。

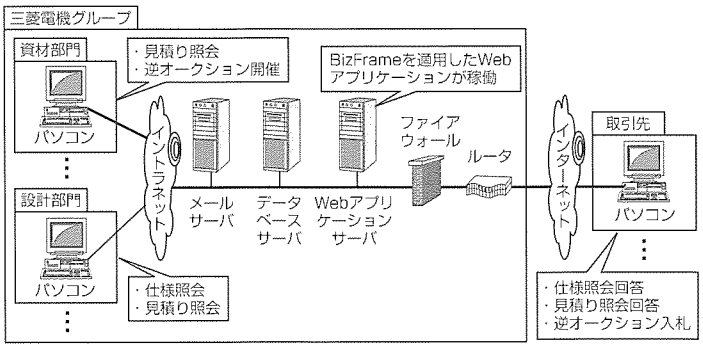


図4．電子商談システム“プロキオン”のシステム構成

6. むすび

Webコンピューティングを取り巻く環境は技術進展が著しく，例えば，Webサービス技術，ポータル技術，ビジネスプロセス管理(Business Process Management：BPM)を用いたシステム間連携技術などが最近注目を浴びている。今後，これらの新しい技術を用いたシステム構築が行われるようになる。

本稿では，Java，Webアプリケーションサーバを使用した3階層システムの構築をターゲットとしたアプリケーションフレームワークを紹介したが，さらに，上述した新しい技術に対応可能なアプリケーションフレームワークを検討し，開発していく必要がある。

今後とも日々革新する情報技術をキャッチアップし，新しいソリューションとして提供していく所存である。

参考文献

(1) Ralph E. Johnson：パターンとフレームワーク，共立出版 (1999-6)

(2) Fondatao Inc.：コンポーネントモデリングガイド，ピアソンエデュケーション (2001-9)

(3) John Cheesman，ほか：コンポーネントベースソフトウェアのための開発プロセス，ピアソンエデュケーション (2002-5)

(注6) 特定の文字に対するWWWブラウザでの文字化けへの対処，データベースとのコネクションのクローズ処理への対処など。

車載情報端末フレームワーク

上川哲生*
下谷光生**
橋本浩二***

要　旨

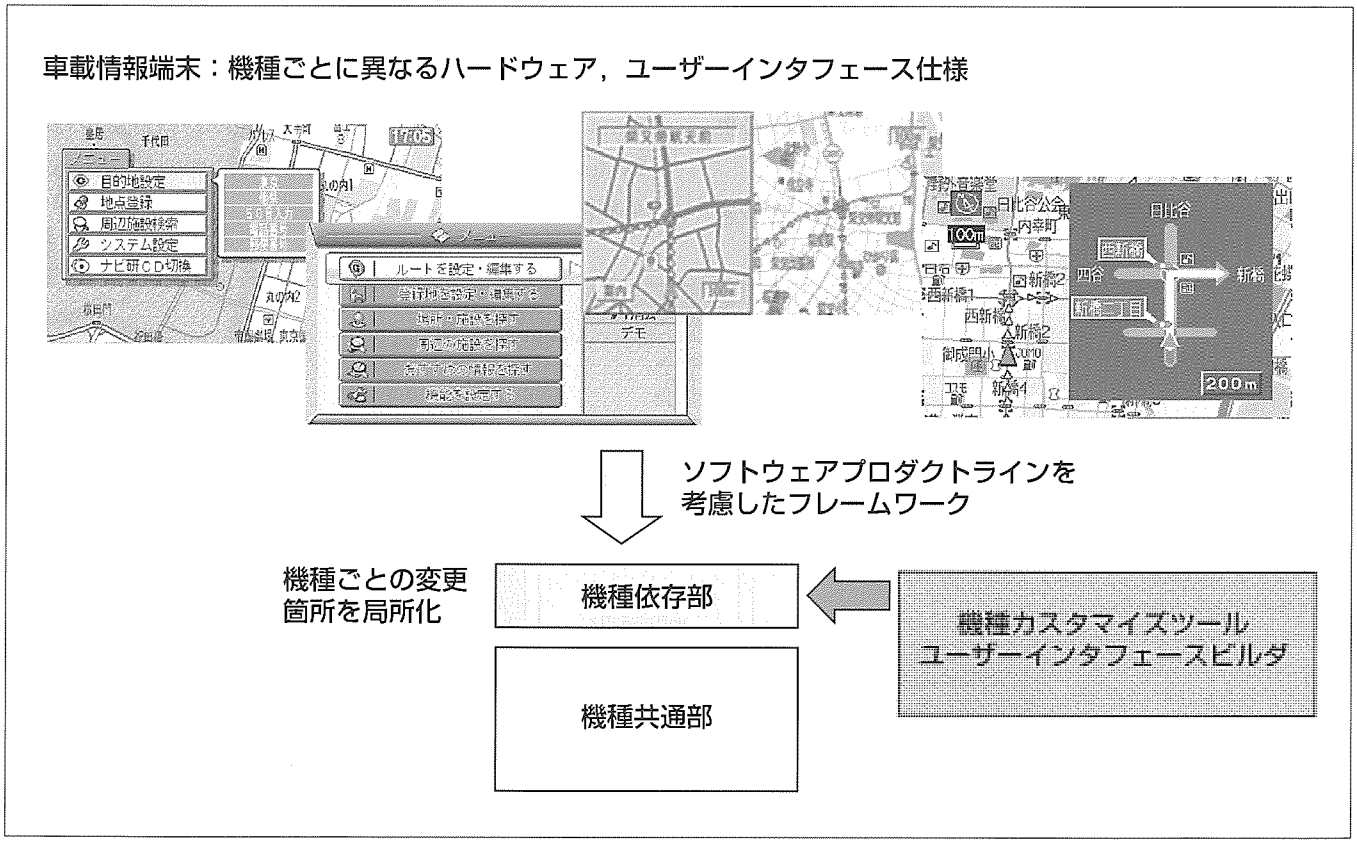
カーナビゲーションシステムに代表される車載情報端末は，1980年代初期に発表されて以来，飛躍的な進歩を遂げてきた。1990年初期のカーナビゲーションシステムは自車位置検出と地図表示のみの機能であったが，その後数年で目的地までの推奨経路探索と音声と交差点拡大図による経路案内機能が基本機能となり，現在ではVICS(Vehicle Information and Communication System)や携帯電話経由で車外の情報を取得できる車載情報端末へと進化し続けている。

車載情報端末の多機能化に伴い，そのプログラムコード量も増加し，開発コストも加速度的に膨れ上がってきている。また，開発する各機種ごとに，ユーザーインタフェース仕様やハードウェア仕様が異なり，機種ごとに異なるソフトウェアを開発する必要がある。

そこで，三菱電機では，従来の職人芸的な組込みシステムソフトウェア開発を見直し，一連の機種ソフトウェア開発において開発コストを抜本的に削減することのできるフレームワークを開発した。

さらに，ハードウェアプラットフォームの違いに依存しないJava(注1)アプリケーションを実行できるようにして，オープンな車載情報端末を実現した。サードパーティや個人ユーザーは自由にJavaアプリケーションを作成し，当社の車載情報端末上で実行させることができる。また，このフレームワークでは，Javaにナビゲーション拡張API(Application Program Interface)を設けており，Javaアプリケーションから車載情報端末のナビゲーション機能を呼び出すことができる。

(注1)　Java及びJavaに関連する商標は，米国Sun Microsystems, Inc.の米国及びその他の国における商標又は登録商標である。



車載情報端末のソフトウェアフレームワーク

カーナビゲーションシステムに代表される車載情報端末では，各機種ごとにハードウェア仕様，ソフトウェア仕様が大きく異なる。そこで，ソフトウェアプロダクトラインを考慮したフレームワーク設計が有効となる。機種の違いによって変更を受けるソフトウェア部分を機種依存部，変更を受けない部分を機種共通部として分離し，機種ごとの変更箇所を局所化する。さらに，機種カスタマイズツールや，ユーザーインタフェースビルダを用いることにより，機種ごとのカスタマイズを容易にしている。

*先端技術総合研究所　**自動車機器開発センター　***三田製作所

1. まえがき

カーナビゲーションシステム[1]に代表される車載情報端末の市場は，2002年に国内で年間200万台を超え，急速に拡大している。市場の拡大に伴う製品の高機能化で，そのソフトウェアのプログラムコード量も増加し続けている。プログラムコードの規模は1機種当たりで300万ステップを超え，従来の職人芸的な組込みソフトウェア開発手法[2]での開発は困難になってきている。

そこで当社では，従来の開発方式を見直し，機種開発時のソフトウェア生産性を抜本的に向上することを目指して，車載情報端末フレームワークの再構築を行った。

本稿では，構築した車載情報端末フレームワークについて述べる。

2. 車載情報端末におけるソフトウェアプロダクトライン

自動車のデザインがメーカーや車種によって異なるように，車載情報端末の仕様もまた車種によって異なる。そこで，車載情報端末におけるソフトウェア開発では，一連の機種開発において，ソフトウェア生産性が最適になるように開発を進める必要がある。

ソフトウェアプロダクトラインでは，機種ごとに変更が必要とされるソフトウェアの変更箇所を局所化することが望ましいとされる。これは，複数のソフトウェアモジュールに変更箇所が分散すると，モジュール間のインタフェース仕様の変更や，複数の開発チームによる協調作業が必要となるからである。

そこで，まずは新機種開発時にソフトウェア変更が必要となる箇所を局所化するために，**図1**に示すような車載情報端末のソフトウェアリファレンスモデルを作成した。リファレンスモデルでは，機種によって変更が必要な箇所を依存部として分離している。機種によって変更が必要な箇所を以下に挙げる。

(1) ユーザーインタフェース
画面仕様，操作仕様

(2) 搭載アプリケーション
DVD(Digital Versatile Disc)ビデオ再生機能，Webブラウザ，メーラー音声認識，占い，電卓など搭載するアプリケーション

(3) 地図フォーマット
採用する地図のフォーマット

(4) 入力デバイス
リモコン入力，タッチパネル入力，ハードキー入力など使用する入力デバイス

(5) ストレージデバイス
CD-ROM(Compact Disc-Read Only Memory)，DVD-ROM，ハードディスクドライブなど，使用するストレージデバイス

(6) グラフィックスデバイス
使用するグラフィックスチップ

(7) 車両情報
車内LAN(Local Area Network)非接続，CAN(Controller Area Network)接続，MOST(Media Oriented Systems Transport)接続，IDB(Intelligent Transportation Systems Data Bus)-1394接続，オリジナルバス接続など，車両情報の入手手段

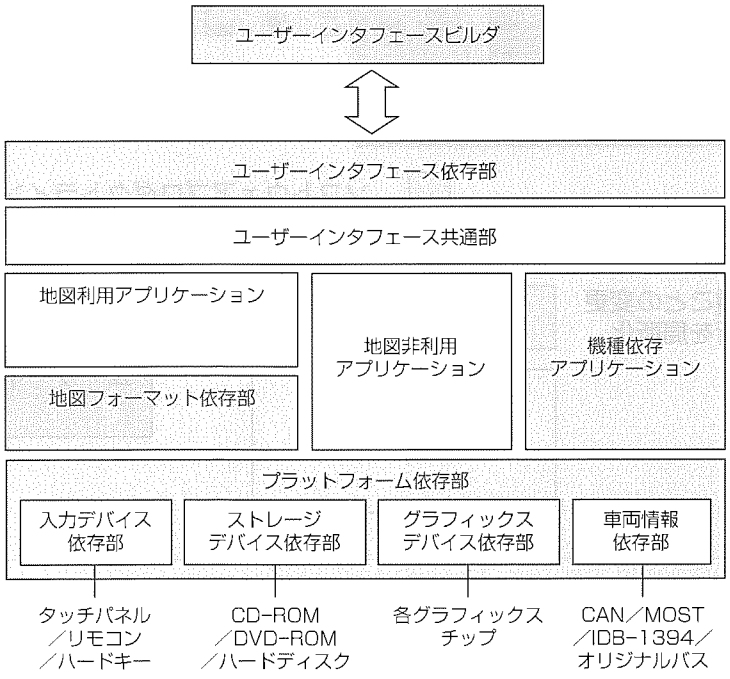


図1．車載情報端末ソフトウェアリファレンスモデル

このソフトウェアリファレンスモデルに従ってフレームワークを再構築した。このリファレンスモデルに従ったフレームワークでは，機種によるソフトウェア変更箇所が局所化されているため，ソフトウェア生産性を飛躍的に向上することができる。

3. ソフトウェア生産性向上の取り組み

さらに，このフレームワークでは，以下に挙げるような技術の導入により，ソフトウェア生産性の更なる向上を目指した。

(1) オブジェクト指向設計の導入

C++によるオブジェクト指向設計[3]により，ソフトウェアの生産性と再利用性を高めた。

(2) コンポーネント指向設計の導入

オブジェクトより粒度の高いコンポーネント指向[4]の設計により，コンポーネントの組合せによって機種によるアプリケーション構成の違いを変更できるようにした。

(3) パソコン環境での開発

実機環境とパソコン開発環境の違いを吸収するミドルウェア層を設けることにより，パソコン上でのソフトウェア開発を可能とした。実機用のOS(Operating System)にはパソコン環境と親和性が良いマイクロソフト社のWindows CE(注2)を採用した。

システム試験からソフトウェア改修をすべて同一の開発パソコン上で行えるため，デバッグサイクルの短縮を図ることができる。また，ハードウェアが未だ完成していない段階からパソコン上で開発できるため，ハードウェアとソフトウェアのコンカレント開発が行える。**図2**にパソコン開発環境画面の例を示す。キーボードによるキー入力，又はリモコンダイアログボックスのマウスでのクリックにより，操作を行うことができる。

(4) ユーザーインタフェースビルダの適用

仕様決定のためのプロトタイピングと，ユーザーインタフェース部のソフトウェア自動生成のために，**図3**に示すようなユーザーインタフェースビルダを開発した。このユーザーインタフェースビルダを使用することにより，最も仕様変更量の多いユーザーインタフェース部を，ソフトウェアコーディングすることなしに，容易に構築することができる。

4. Javaによるオープンな車載情報端末

携帯電話の普及率が高まり，車内において外部の情報やコンテンツをインターネット経由で取得するテレマティックスサービスへの関心が高まっている。しかし，現状の車載情報端末では，メーカーごとに全く異なるプラットフォームの製品を販売しているため，その上で動作するアプリケーションに全く互換性がない。インターネット上でアプリケーションコンテンツを流通させるためには，どのメーカーの車載情報端末上でも動作するオープンな車載情報端末の実現が必要である。

そこで，当社では，サン・マイクロシステムズ社が開発したJava[5]を車載情報端末上のアプリケーション実行環境として採用した。Javaの特長である，プログラム開発の自由度が高い，アプリケーションを実行する前に不正なプログラムがないかチェックする高いセキュリティ機能，アプリケーションの実行がナビゲーションシステム本来の機能を阻害することはない，といった点を重視した。

特に，Javaアプリケーション自身がハングアップしてもナビゲーションシステム自体の動作に影響を与えないという点が，OS上のネイティブアプリケーションと比べると，高信頼性が求められる車載環境においては優れている。また，Javaの最大の特長はマルチプラットフォームであり，車載情報端末のOSや，メーカーのハードウェアとい

(注2) Windows CEは，米国Microsoft Corp.の米国及びその他の国における登録商標である。また，Microsoft Corp.が開発した機器組込用OSである。

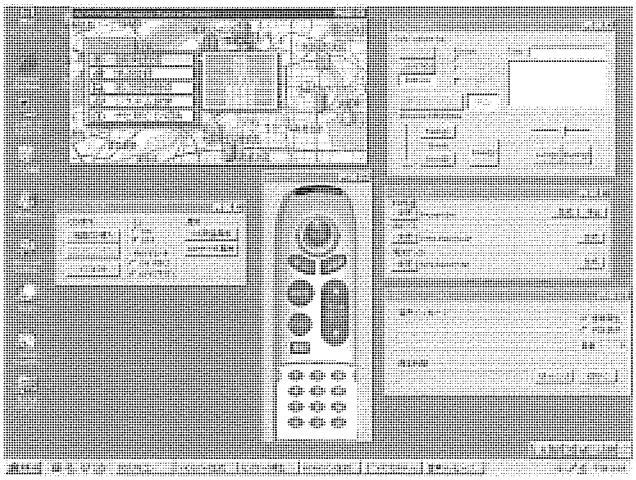


図2．パソコン開発環境画面例

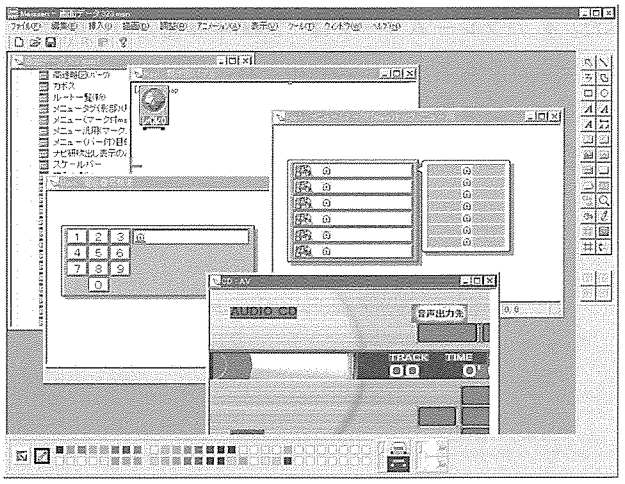


図3．ユーザーインタフェースビルダ

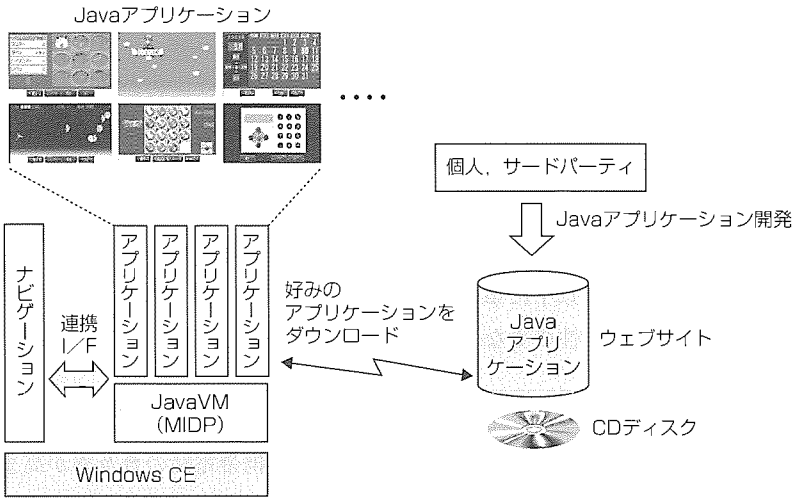


図4．Javaによるオープンな車載情報端末の実現

ったプラットフォームの違いに関係なく，同一のアプリケーションを実行できる。

当社の車載情報端末では，モバイル情報機器向けのMIDP(Mobile Information Device Profile)準拠のJavaバーチャルマシン(Java VM)を搭載した。さらに，ナビゲーション機能と連携するためのナビゲーション連携APIも用意した。ナビゲーション連携APIを用いると，自車位置の取得，目的地の設定，目的地情報の取得，指定位置の地図表示，指定メニュー画面表示，車両情報の取得，音声合成出力などのナビゲーション機能をJavaアプリケーション側から呼び出すことができる。

当社は，これらのナビゲーション連携APIとアプリケーション開発用のSDK(Software Development Kit)を一般に公開しており，**図4**に示すように，サードパーティや個人ユーザーが自由にJavaアプリケーションを作成し，車載情報端末上で動作させることができる。

5. むすび

以上，開発した車載情報端末フレームワークについて述べた。このフレームワークは，2000年春発売の市販カーナビゲーションシステムに初適用され，現在では，当社のカーナビゲーションシステム全機種への適用が完了している。

このフレームワークの適用により，1機種当たりのソフトウェア開発コストは1/3となり，そして，1機種当たりの開発期間は1/2となり，大幅にソフトウェアの生産性を向上することができた。

また，2002年にはこのフレームワークへJavaを導入し，サードパーティや個人ユーザーが自由にアプリケーションを作成して実行できる環境を実現した。

各車載情報端末メーカーが同様にJavaを採用しナビゲーション連携APIを統一することにより，インターネット上をJavaアプリケーションのコンテンツが自由に流通されるテレマティックス時代の到来を願っている。

参考文献

(1) 横内一浩，ほか：車載ナビゲーションシステム，三菱電機技報，**73**，No.10，709～713（1999）
(2) 高木敏宏，ほか：VxWorksの概要と開発環境Tornadoの実際，Interface，93～102（2001-12）
(3) J.ランボー，ほか：オブジェクト指向方法論OMT，トッパン（1992）
(4) 青山幹雄：コンポーネント指向開発，bit，**32**，No.3，3～7（2000）
(5) Yu Feng，ほか：J2MEワイヤレスJavaプログラミング，アスキー出版（2002）

組込み用ブラウザ・メーラソフトウェアの展開

山中　弘*　西川正治***
佐々木幹郎*　田中功一*
中　邦博**

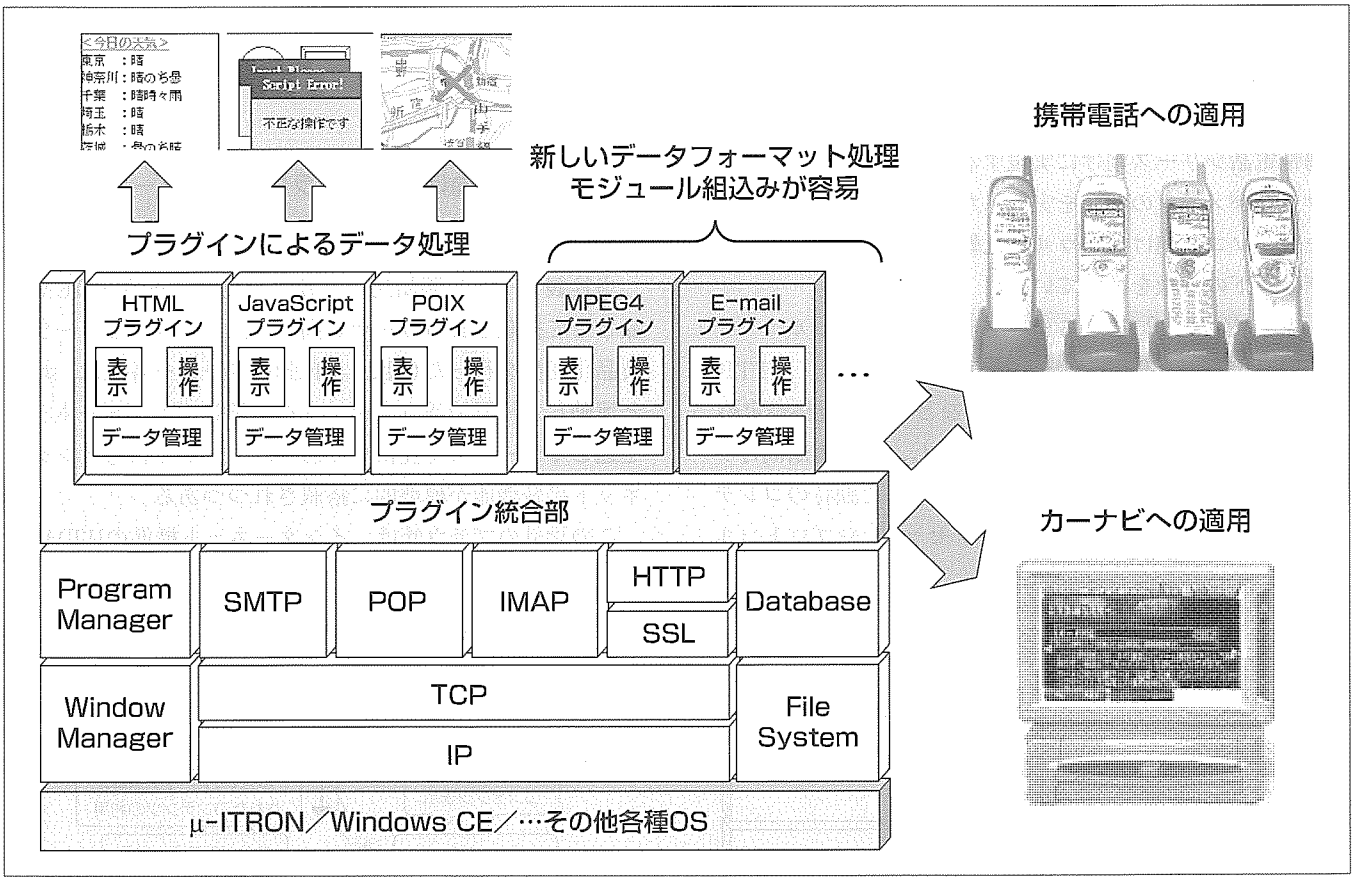
要　旨

携帯電話向けブラウザ及びメーラ開発において，開発規模が大きく様々な機種に展開可能なコンテンツデータ処理モジュールに着目し，流用可能なソフトウェア・コンポーネントとしての設計・実装を試みた。設計においては，組込みソフトウェアの分野で機種依存性を強くする要因となっている要求仕様・ハードウェアの相違に柔軟に対処可能とすることに留意した。

また，実際に，開発したモジュールをカーナビ向けブラウザ及びメーラ開発に流用し評価を行った。カーナビ向けブラウザでは，処理するコンテンツ記述言語をHTML3.2相当に拡張したほか，携帯電話ハードウェアよりも大きなメモリ・画面サイズへの対処を行った。基本的なデータ構造及び処理構造を変えることなく流用が可能であり，コンポーネントとしての有効性を実証できた。

更なる再利用性向上を目指し，個々のデータフォーマット処理モジュールのデータ管理・表示・操作機能分割と，プラグイン構造設計によるデータフォーマット処理とモジュール統合処理の分割を行い，分割モジュール間のインタフェースを明確化するブラウザ再設計を実施した。プラグイン設計においては，製品出荷後の障害対応容易化をねらいとし，組込み機器での動的プラグイン実現も考慮に入れた。

再設計したモジュール構成により，更に広範囲の機器への適用や，ブラウザ・メーラ以外のデータ処理アプリケーションへの適用をねらう。



組込み機器向けブラウザ最終設計イメージと適用計画

再設計した組込みブラウザは，各種データフォーマット処理部と統合処理部とを分離・インタフェースを明確化したプラグイン構造を持ち，サポートするデータフォーマットの取捨を容易化する。各データフォーマット処理部はデータ管理・表示・操作機能を分離しており，要求仕様と実装環境の相違に伴う変更を局所化する。実用化対象としては，携帯電話，カーナビなどを計画中である。

*情報技術総合研究所　**モバイルターミナル製作所　***三田製作所

1. まえがき

組込み機器の分野では，短い開発期間でバグのない高信頼のソフトウェア開発が求められる。競争の激化から，組込みソフトウェアは年々高機能化・複雑化してきており，開発効率の向上が急務である。

開発効率向上には，ソフトウェアの部品化と流用は有効な手段の一つである。しかしながら，組込み機器の分野では，機種ごとに多様に要求仕様が変化すること，厳しいハードウェアリソース制約を満たしつつ十分な性能を実現しなければならないことなどから，対象機種独特の設計・実装が行われ，別の機種への流用が難しくなる場合が多い。

今回は，携帯電話向けブラウザ・メーラ開発において，開発規模が大きく様々な機種に展開可能な処理モジュールに着目し，要求仕様・ハードウェアの相違に柔軟に対処可能とする開発を目指した。実際にそのモジュールをカーナビ向けブラウザ開発に流用評価し，再利用性の更なる向上を試みたので紹介する。

2. 携帯電話向け組込みブラウザ及びメーラの開発

2.1 携帯電話向けブラウザの概要

1990年代末から，携帯電話キャリア各社は，コンテンツ・メールなどのインターネットアクセスサービスを開始しており，現在販売されているほとんどの携帯電話には組込みブラウザ・メーラが搭載されている。これを三菱電機製携帯電話シリーズ向けに実現するため，今回，組込みブラウザ及びメーラの開発を実施した(図1)[1]。

一般パソコン向けブラウザと比較し，携帯電話向けブラウザは以下の特徴を持っている。

(1) キャリアごとに異なるデータ形式・プロトコル

HTML(HyperText Markup Language)[2]／HTTP(HyperText Transfer Protocol)[3]などの標準規格が定められているパソコン向けブラウザに対し，携帯電話向けブラウザでは，キャリア各社ごとに定められた独特のコンテンツ記述言語(基本はHTMLのサブセット)やプロトコルに対応しなければならない。

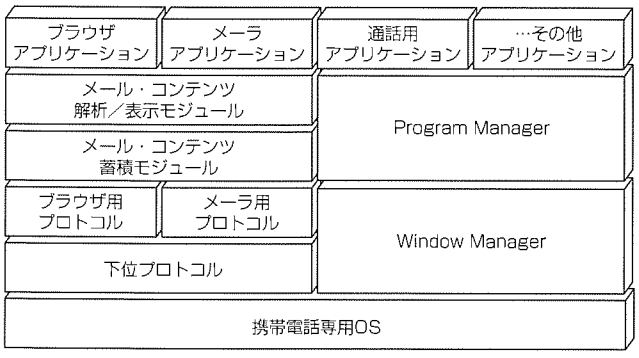


図1．携帯電話向けブラウザ周辺モジュールの構成

(2) 少ないハードウェアリソース

携帯電話向けソフトウェアは，ROM(Read Only Memory)・RAM(Random Access Memory)のサイズや，CPU(Central Processing Unit)の性能など厳しいハードウェアリソース制約を満たしながら，要求機能の提供と十分なユーザーレスポンスを実現しなければならない。

2.2 開発方針

開発規模が特に大きく機能的に他の機種開発にも展開可能なコンテンツデータを解析・表示するモジュール：ブラウザエンジン(図1のメール・コンテンツ解析／表示)に着目し，流用による開発効率化を実現するために，下記の開発方針を設定した。

(1) コンテンツ記述言語に対する柔軟性

各キャリアのコンテンツ記述言語の基となっているHTMLを対象に，基本的なデータ構造・処理構造を構築し，タグやタグ属性の名前変更，機能の追加・削除に容易に対応可能とする。

(2) ハードウェアに対する柔軟性

当時の開発対象機種のハードウェア制約(RAMサイズ・画面サイズ)にとらわれず，RAMが増減した場合，画面サイズが変更された場合にも容易に対処可能な設計を行う。サイズ変更による影響を考慮すべきデータタイプ・定数定義値を明確化し，これらの再定義を行えば基本データ構造・処理構造に影響を与えることなく変更が可能な実装を目指した。

3. カーナビ向け組込みブラウザ及びメーラの開発

3.1 カーナビ向けブラウザの概要

当社製カーナビ製品向け，及び自動車メーカーOEM製品として搭載されるカーナビ向けに，ブラウザとメーラの製品化開発を行った(図2)。自動車メーカー各社は，地図と連動した位置情報コンテンツを整備し，インターネットを通してサービスを提供しており，カーナビにもインターネット接続機能が標準的に搭載されつつある。

この設計のブラウザは，インターネット標準のHTML／

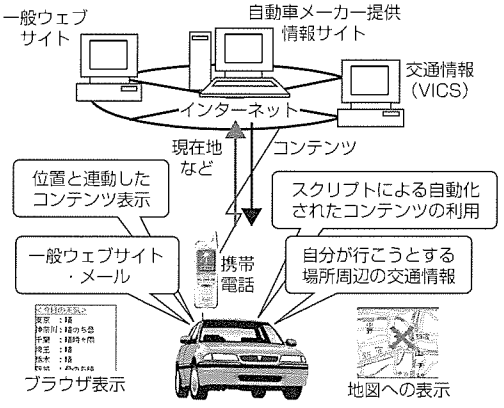


図2．カーナビ向けブラウザのシステムの概略

HTTPに準拠するだけでなく，位置情報記述方式として標準化が進んでいるPOIX(Point Of Interest eXchange language)[4]に対応した位置情報交換機能や，JavaScript[5]によるスクリプト処理機能を新規に追加した。

3.2 携帯電話向けブラウザエンジンの流用

携帯電話向けブラウザエンジンをカーナビ向けブラウザのHTML解析処理に流用することで，開発期間の短縮を図った。

以下の開発作業を実施した。

(1) HTML3.2相当のコンテンツ記述への対応

携帯電話向けブラウザエンジンが解析して持つコンテンツ各要素の情報は，HTML3.2相当のものが最初から準備されており，解析処理をほぼそのまま流用することができた。

(2) 処理可能コンテンツサイズの拡大

携帯電話向けブラウザは，使用可能RAMリソースの制約から処理可能なコンテンツサイズにも制約を付けた実装が行われていた。カーナビでは，使用可能RAMリソースが増えるため，コンテンツサイズ制約をなくす設計とした。開発ライン数1千行弱のメモリ管理処理の変更，データタイプのサイズ・定数定義値の変更を実施したが，基本的内部データ構造は特に変更することなく流用可能であった。

(3) 描画処理のカーナビ向け移植

カーナビ実装環境は，高機能な描画処理用コンポーネントモジュールが提供されている。カーナビ各アプリケーションのユーザーインタフェースを統一するという前提の下，ブラウザエンジンが解析して生成したコンテンツ要素データを描画コンポーネント向けのデータに変換する必要があった。携帯電話の描画処理構造は流用できず，新規開発となった。

(4) 位置情報記述解析への対応

新規開発であったが，位置情報記述言語POIXがHTMLに近い言語仕様を持つため，携帯電話向けブラウザエンジンの解析処理構造を流用した設計・実装を行うことができた。

3.3 評　価

開発規模実績で評価すると，携帯電話向けブラウザエンジンの95%のソースコード流用が可能であった。これは，カーナビ向けブラウザエンジン全体の55%に当たる。

また，開発工数実績で評価すると，カーナビ向けブラウザエンジンを新規に開発した場合の予測工数に対し，実際の開発では63%の工数削減を実現した見積りとなる。ブラウザエンジン部では，各種タグ・タグ属性などの組合せ，不正記述コンテンツ，端末リソースを多く消費するコンテンツなど，多くのケースを考慮する必要があり，携帯電話向け開発では，品質確保に大きな工数を要した。これに対しカーナビ向け開発では，次の2つの携帯電話向けブラウザ開発当初の設計目標達成により，品質の確保されたデータ構造・処理構造をほぼそのまま流用することができ，品質確保作業を大きく削減することができた。

(1) 字句解析処理部分がHTML3.2への対応だけでなく，位置情報記述に関する解析にも用いることができ，柔軟にタグセットの交換が可能であったこと。

(2) メモリ制約・画面サイズが大きく変更されたカーナビ上でも，基本的なデータ構造・処理構造を変更する必要がなかったこと。

4. 再利用性の向上

4.1 カーナビ開発における問題点

カーナビ向けブラウザ開発への流用を通し，以下の問題点が挙げられた。

(1) 実装環境の表示・操作処理部の相違

表示処理・操作処理部は，実装環境により大きく異なる。携帯電話開発当初は，実装環境それぞれの低レベル描画API(Application Program Interface)層で相違を吸収することを想定していたが，カーナビのように実装環境の描画コンポーネントを利用しユーザーインタフェースを統一するという開発要件には対処できなかった。このため，カーナビ開発では，携帯電話向けブラウザエンジン中の描画・操作処理依存部の調査・抽出作業を実施しなければならなかった。

(2) 処理可能なデータフォーマットの追加

コンテンツ中に存在する画像やサウンドなどの埋め込みデータフォーマットは様々であり，要求仕様に応じ多様なデータフォーマットへの対応が求められる。実際カーナビ向けブラウザ開発では，携帯電話でサポートしていなかった画像データフォーマットやJavaScriptなど，コンテンツ埋め込みデータの処理モジュールを追加した。携帯電話向けブラウザエンジンでは，埋め込みデータ処理構造が独自のモジュールに依存した実装となっており，処理モジュールの追加において依存部の調査・抽出作業を実施しなければならなかった。

以上のカーナビ開発における問題点のフィードバックを受け，更なる再利用性の向上を図るために，次節以降に述べるモジュール構造の再設計を試みた。

4.2 MVC(Model／View／Controller)設計手法の適用

MVC設計手法[6]とは，データ管理・表示・操作それぞれの機能を独立したモジュールModel／View／Controllerに分割し，インタフェースを明確化し，それぞれの要求仕様変更に伴う作業を局所化する方法である。

組込みブラウザへの適用において，各データフォーマット処理モジュールそれぞれをMVC構造とした。これによって，実装環境の表示・操作系に依存する変更作業は，それぞれView／Controllerモジュールに集約することが可能である。また，機種ごとの仕様相違が少ないデータ処理などのModelモジュールについては，手を加えることなく他

の機種に横展開することができる。

4.3 組込み向けプラグイン構造の設計

新しいデータフォーマット対応を容易化するため，個々のデータフォーマット処理部とモジュール統合部を分離し，相互間に幾つかのインタフェースを規定する。この規定インタフェースに則ったモジュールを作成することにより，容易に新データフォーマット処理を組み込んだり，不要なデータフォーマット処理を削除したりすることが可能なプラグインモジュール構造を構築した(図3)。

データフォーマット処理部とモジュール統合部とのインタフェース規定に当たっては，将来的にインターネットからプラグインモジュールをダウンロードし，データ処理機能を自動拡張することのできる動的プラグイン機能を実現することを考慮に入れた。動的プラグインによる製品の機能向上とともに，これまでの組込みソフトウェア開発における大きな課題の1つであった製品出荷後に発覚した障害対応容易化をねらいとしている。

組込み機器向けの動的プラグイン実現においては，ハードウェアリソース制約から以下の問題が存在する。

(1) 組込み機器では，プラグインモジュールの保存領域が小さく抑えられてしまう。このため，ダウンロードしたモジュールを単に追加保存していくだけでは，簡単に領域不足を引き起こしてしまう。

(2) 新しいプラグインを追加して実行するとき，CPUに与える負荷や必要なメモリサイズなどが分からない。このため，ブラウザが既存のプラグインと新規追加したプラグインを同時に実行できるかどうかの判定が難しい。

これらの問題に対し，次のインタフェース規定と機能を，各々のプラグイン処理部とモジュール統合部に持たせることにより対処した[7),(8)]。

(3) プラグインの実行に，呼出し期限，呼出し回数，処理データサイズなどの制限を設け，制限を超えたプラグインを自動的に削除する。

(4) コンテンツ表示時に必要／不要なプラグインを把握し，新しいプラグインのダウンロードが必要なときに保存領域があふれてしまう場合には，不要プラグインを削除する。

(5) 各々のプラグインのある動作状態におけるメモリ・CPU負荷状況通知手段，及びプラグインの動作状態変更手段についてのインタフェースを規定する。モジュール統合部側では，プラグイン側に設けたインタフェースを用いてメモリ・CPU負荷を分散させることにより，プラグイン競合実行を実現する。

5. むすび

携帯電話向けに開発した組込みブラウザエンジンをカーナビ向けに流用することにより，開発効率化を実現した。携帯電話向けブラウザエンジンの開発に当たっては，機種

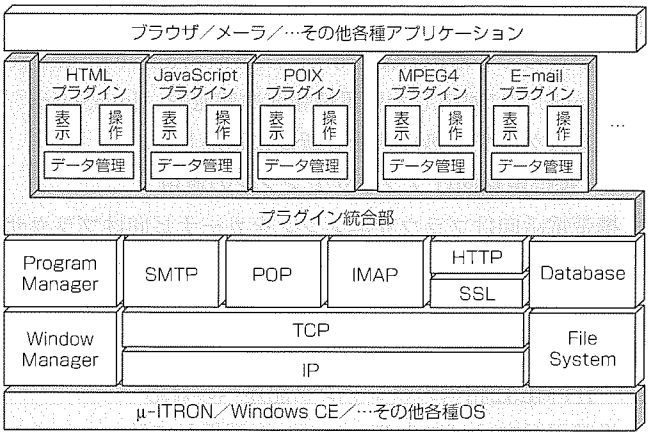


図3．再設計後の組込み機器向けブラウザの構成

シリーズ化や他のキャリア向けへの展開を目標にハードウェア性能やコンテンツ記述言語の相違に柔軟に対処可能な設計を目指したが，カーナビ向け開発で設計内容の有効性を実証できた。

また，カーナビ向けブラウザ開発で挙げられた問題点に対処し，MVC設計手法導入によるデータ管理・表示・操作処理変更の容易化，プラグイン構造導入によるインターネット上の様々なデータフォーマット対応の容易化を図った。現在，その設計に基づくプロトタイプ作成と評価を実施中である。

今後は，評価に基づき，新設計による組込みブラウザの実用化を進めるとともに，プラグイン構造導入のもう1つの目標である，インターネットからプラグインモジュールをダウンロードし，データ処理機能を自動拡張する動的プラグインの実現を計画している。

参考文献

(1) 山中　弘，ほか：携帯電話向け組み込みブラウザの開発，第60回情報処理学会全国大会（2000-3）

(2) World Wide Web Consortium(W3C)：HyperText Markup Language(HTML)（1999）

(3) World Wide Web Consortium(W3C)：HTTP-Hypertext Transfer Protocol（2000）

(4) モバイル標準化検討委員会(MOSTEC)：POIX：Point of Interest eXchange language Version 2.0（1999）

(5) Flanagan, D.：JavaScript：The Definitive Guide, 4th Edition, O'Reilly（2001）

(6) Gamma, E. et al.：Design Patterns：Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing（1994）

(7) マットモラン：Embedded System Plug-in Re-configuration Preference, 情報処理学会FIT2002（2002）

(8) 山中　弘，ほか：組込ブラウザにおけるプラグイン競合実行制御，情報処理学会FIT2002（2002）

組込み用UI設計ツールをベースとしたUI設計開発

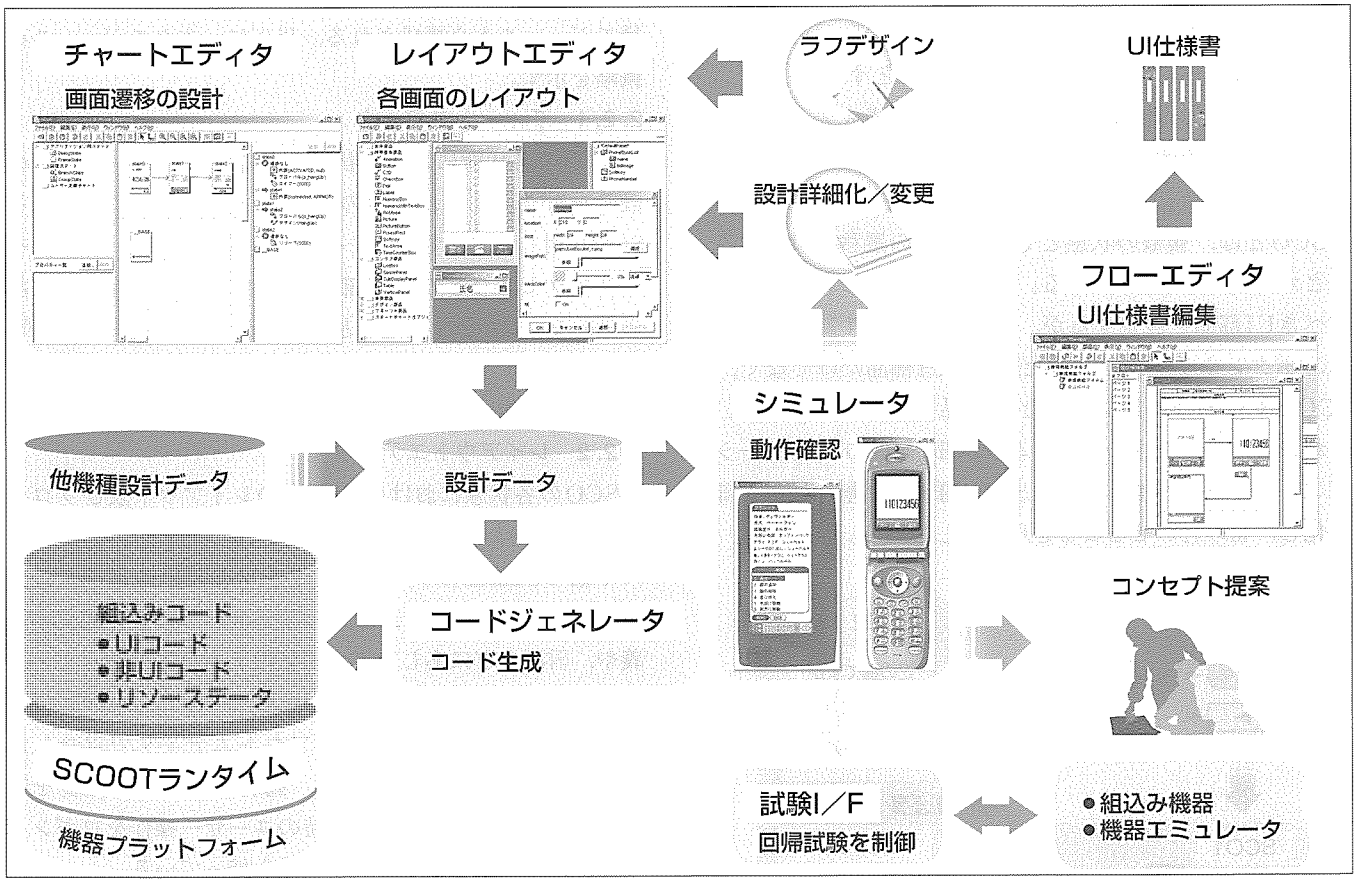
小中裕喜* 浮穴朋興***
中川隆志**
小船隆一***

要 旨

携帯電話を始めとする組込み機器において，ユーザーインタフェース(UI)は重要な差別化要因であり，機器の高度化・多機能化に伴う仕様の拡張やユーザー嗜好(しこう)に応じた仕様変更が日々行われている。そしてこれらが，UIソフトウェアの開発コスト急増を招いている。

そこで三菱電機では，UI設計開発の効率化を目的として，“組込み用UI設計ツール”の開発を行ってきた。UI設計の部品化，階層化，再利用性向上を図るため，このツールでは，UIの複数の状態とそれらの間の状態遷移，及び各状態におけるUI部品のレイアウトを定義可能なステートチャートオブジェクト(State Chart Object：SCO)という概念を導入している。SCOの概念をベースとすることにより，このツールでは，組込み機器本体から各種アプリケーション，そしてアプリケーション中で用いられる表示部品に至るまで，すべて同一の枠組みで設計し，それらを組み合わせてシミュレートすることを可能としている。また，このツールは，上流工程のUI設計にとどまらず，UI仕様書作成から実機向けコード生成，回帰試験実行に至るまでシームレスにサポートし，UIソフトウェアの生産性及び品質の向上に寄与する。

本稿では，組込み用UI設計ツールの概要と，このツールをベースとした組込み機器UIの設計開発，そしてその適用例として，携帯電話開発における取り組みについて述べる。



組込み用UI設計ツールの構成とUI設計開発の流れ

組込み用UI設計ツールは，ラフデザイン段階から詳細設計まで幅広く用いることが可能である。エディタ群を用いて設計したデータは，即座にシミュレートして動作確認を行うことが可能であり，設計への迅速なフィードバックを容易とする。また，シミュレーション結果からのUI仕様書作成や回帰試験，設計データからの実機コード生成などをサポートすることにより，トータルなUI設計開発コストの低減を図る。

*先端技術総合研究所(工博) **同研究所(エネルギー科学博) ***モバイルターミナル製作所

1. まえがき

携帯電話を始めとする組込み機器のUIは，機能の増大やデザインの多様化，開発期間の短縮などに伴い，その生産性向上が大きな課題となっている。

大画面上でポインティングデバイスによる直接操作が可能なデスクトップ上のUIと異なり，多くの組込み機器では，画面が小さく，操作手段も限られるため，シーンを切り換えながら操作するようなUIを提供することが一般的である。また，各シーンにおいても，フォーカス状態などに応じて部分的に細かく表示を切り換えることが多い。このようなシーンや部分表示の切換えに伴う表示状態の組合せの爆発的な増大は，従来のGUI(Graphical User Interface)ビルダや状態遷移図などを用いた設計アプローチの適用を困難なものとしていた。

本稿では，上記課題を解決するためにUI設計の階層化・部品化をサポートする組込み用UI設計ツール[(1)]の概要を説明する。また，このツールをベースとしたUI設計開発と，当社における携帯電話開発への適用について述べる。

2. 組込み用UI設計ツール

このツールは，表示の切換えを伴うUIの階層的モデリングに適したSCOという概念をベースとする。以下，SCOの概念と，それを核としたツールの構成要素について，それぞれ説明する。

2.1 SCO

SCOとは，複数の状態とそれらの間の遷移を設計することが可能なUI部品である。各状態にはその状態で表示すべきUI部品を設定するが，このとき，図1に示すように，ボタンやラベルといった既製のUI部品だけでなく，SCOとして新たに設計されたUI部品を利用してもよい。これにより，画面中のカスタム表示部品からアプリケーション，そして機器本体に至るまで，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。

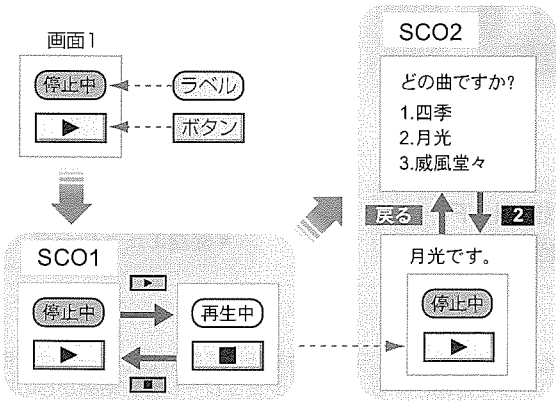


図1．ステートチャートオブジェクト(SCO)

SCOの概念に基づいたUI設計の部品化について，その特徴を以下に示す。

(1) 専用のプログラミングが必要な既製UI部品とは異なり，既製UI部品を組み合わせて部品SCOとすることにより，既製UI部品と類似したルックアンドフィールを保ちながら，新たな機能を持ったUI部品を容易に作成することができる。

(2) 複数のアプリケーションで共通して用いられる一連のシーンを共通SCOとして作成し，共用することにより，アプリケーション間での操作性の統一を図ることが容易となる。

(3) SCOの設計を修正すれば，すべての使用場面にその修正が反映される。すなわち，仕様変更に伴う作業が局所化される。

(4) 表示状態の組合せは，表示を構成するSCOの状態の組合せに還元される。したがって，各SCOにおいては，表示状態の組合せを直接扱う必要がなく，それぞれ固有の状態空間を持つUI部品として，局所的に設計することが可能である。

2.2 ツール構成

要旨の図に示すように，このツールは，SCOの設計をGUIベースで行うエディタ群と，設計データを利用する各種モジュールから構成される。以下に，主なモジュールの概略を示す。

2.2.1 チャートエディタ

SCOの各状態における状態遷移などのイベント処理の設計を行う。また，既製のUI部品と同様のプロパティをSCOに対して定義する。イベント処理において，データベースアクセスなどの非UI部との連携やそれに伴うUI部品の動的なプロパティ設定などの動作が必要な場合，Java[(注1)]のサブセットをベースとした簡易スクリプト言語で記述することが可能である。

2.2.2 レイアウトエディタ

SCOの各状態におけるUI部品のレイアウト及びプロパティ設定を行う。あるUI部品のプロパティがデータベースや他のUI部品のプロパティを常に参照するように設定することも可能であり，SCOによるカスタム表示部品の定義や，言語などに応じた表示の切換えが容易となっている。

2.2.3 シミュレータ

チャートエディタ及びレイアウトエディタで設計したSCOの動作をシミュレートし，動的なUI仕様の確認を容易にする。シミュレーション時のユーザー操作をロギングして，後で再生するリプレイ機能も持っている。

2.2.4 フローエディタ

シミュレーション結果からのUI仕様書作成を支援する。

(注1) Javaは，米国 Sun Microsystems, Inc. の米国及びその他の国における商標又は登録商標である。

作成されたUI仕様書の画面イメージをリプレイ機能を用いて自動的に更新することも可能である。

2.2.5 コードジェネレータ

設計データからC++プログラムコードを生成する。UI部品のレイアウト情報だけでなく，SCOの状態遷移やスクリプト記述を含むイベント処理に対応したコードも生成する。生成されたコードをこのツールの提供するランタイムと組み合わせ，機器上で動作させる。移植性を考慮して，コード生成系及びランタイムは，それぞれ機器独立部と依存部に分離されている。

2.2.6 試験インタフェース

シミュレータでリプレイされたユーザー操作を，実際の組込み機器又は機器エミュレータに通知するとともに，ツール側の画面遷移と機器側の画面遷移の比較試験を実行する。

3. UI設計開発

組込み用UI設計ツールは，UIのラフフロー設計からコード生成に対応した詳細設計まで，幅広く対応している。このツールのアウトプットには幾つかのレベルが考えられ，それぞれに対応したツールの適用フェーズが存在する。**表1**に，その一例と，各フェーズにおける参加ユーザー，アウトプットの利用度，各フェーズに必要なデータ／開発項目を示す。

フェーズ1のラフフロー設計の段階では，暫定的な画面イメージなどを表示するデザイン用部品やシミュレート時に簡易操作で発生可能な仮想的イベントを用いて，紙芝居的なシナリオを設計する。後続のフェーズでは、暫定的な画面遷移やレイアウト設計をより具体的なイベント処理やUI部品レイアウトで置き換えながら，順次設計を詳細化する。設計データに関しては，構成管理を行うとともに，シミュレータを利用しながらフェーズごとにレビューを行う。

フェーズが進むに従って，設計データの利用度も大きくなり，コード生成に達した時点でツールの導入効果は最大となる。一方，フェーズ3と4に移行するには，ツール側のカスタマイズや機器側の動作環境の整備なども合わせて行うことになる。ツールの導入に際しては，機器開発のリソースやスケジュールなどを考慮しながら適切なレベルのアウトプットを設定するとともに，フェーズに応じた参加ユーザーからなる機動的な開発体制の編成を行うことが重要である。

4. 携帯電話への適用

4.1 W-CDMA機におけるUI仕様書作成

携帯電話W-CDMA(Wideband-Code Division Multiple Access)機のUI仕様書作成にこのツールの適用を試みた。これは**表1**のフェーズ2に相当する。

UI設計においては，W-CDMA機の各種アプリケーションに対応して，それぞれSCO(アプリケーションSCO)を作成した。また，W-CDMA機本体に対応したSCOを作成するとともに，アプリケーションの起動や切換えなどの制御を行うアプリケーションマネージャー部品を用意して，本体SCOに配置することにより，画面の中でアプリケーションSCOを切り換えながら表示できるようにした。

各アプリケーションSCOのUI設計に用いる既製UI部品としては，ボタンやラベル，静止画，アニメーション表示などの基本表示部品，スクロールやリスト表示などが可能なコンテナ部品がある。これらに加えて，今回は，時刻入

表1．設計フェーズとアウトプット

フェーズ		フェーズ1	フェーズ2	フェーズ3	フェーズ4
アウトプットレベル		ラフフローレベル	表示操作仕様書レベル	アプリケーション連携レベル	コード生成レベル
参加ユーザー	デザイナ	○	△	×	×
	仕様検討者	△	○	×	×
	アプリケーション設計者	×	△	○	×
	プログラマ	×	×	△	○
	設計ツール担当者	×	△	○	△
アウトプットに必要なデータ／開発項目		ラフな画面イメージ／テキスト	ビットマップ・文字列リソース，画面サイズやキー種別などの機種情報	アプリケーション管理を含むUI部品のカスタマイズ	実機側UI／非UI分離，ツール側非UI部の実装，実機側ランタイムの移植
再現度	スナップショットレベル	○	○	○	○
	動的表示操作	×	△	○	○
	実機レベル(非UI部を含む)	×	×	×	○
コード生成／実機へのデータ流用	UI部のみ	×	△	○	○
	非UI部との連携まで	×	△	△	○
他機種へのデータ流用	ビットマップ・文字列	△	○	○	○
	UI部品レイアウト／プロパティ	×	△	○	○
	イベントハンドラ	△	○	△	△
回帰試験適用	シーンIDレベル	○	○	○	○
	ビットマップレベル	△	△	△	○

力や日付表示，カウントダウン，アイコン付きメニューなど，20種以上の部品SCOをあらかじめ用意した。また，“はい／いいえ”の選択や暗証番号のチェック，音量の調節などは共通SCOとして設計しアプリケーションSCOに組み込んで用いることにより，UI設計の効率化を図った。

以上のように，作成された設計データをシミュレートし，その結果をフローエディタでまとめ，UI仕様書を作成した。UI仕様書はPDF(Portable Document Format)形式で出力される。

また，従来の紙の仕様書に加えて，シミュレーションを用いたレビューが可能となった結果，従来なら開発後期で発覚していたような課題を早期に検出し，仕様の誤解や検討不足による手戻りを未然に防止することが可能となった。

4.2 PDC機における自動試験

このツールの適用フェーズがコード生成レベルに至らない場合，機器側の組込みソフトウェアは人手でコーディングされる。このツールは，そのような機器側UIソフトウェアに対して，入力された設計データと同一の画面遷移を行うことを，自動的に比較照合する回帰試験にも利用できる。

回帰試験は次のとおり実現される。まず，試験用データとして，UI仕様に基づく画面遷移を表す設計データ(SCO)と，ユーザー操作を記録した操作データをツール側に用意する。次に，試験インタフェースを介して，シミュレータと組込み機器又は機器エミュレータを接続する。そして，シミュレータで操作データをリプレイすることにより，あるべき画面遷移をツール側で再現する。それと並行して，操作データを時々刻々機器側に通知することにより，機器側でもユーザー操作を再生し，画面を遷移させる。そして，ツール側の画面遷移と機器側の画面遷移を試験インタフェースが比較し，結果を記録する。

画面遷移の照合については，個々のシーンを特定するシーンIDによる比較と，実際に画面に出力されるビットマップによる比較の2つのレベルを選択できる。**図2**にビットマップレベルでの回帰試験の照合例を示す。図の左側がツール側で期待される画面，中央が機器側で得られた画面，右側が比較結果及び不一致時のビットマップの差分イメージである。

携帯電話PDC(Personal Digital Cellular)機開発において，UIソフトウェアの回帰試験にこの自動試験機能の適用を試みた。試験用データの構築においては，機器エミュレータ側での操作及び画面遷移データをツール側の操作データと設計データに自動変換する機能を設け，これを利用した。この結果，試験は機器側でデータを採取した時点が基準となるものの，試験用データ構築コストは極めて小さくなった。また，あるべき画面遷移がSCOの設計データに変換され，図的に表現されるので，画面照合不一致時の

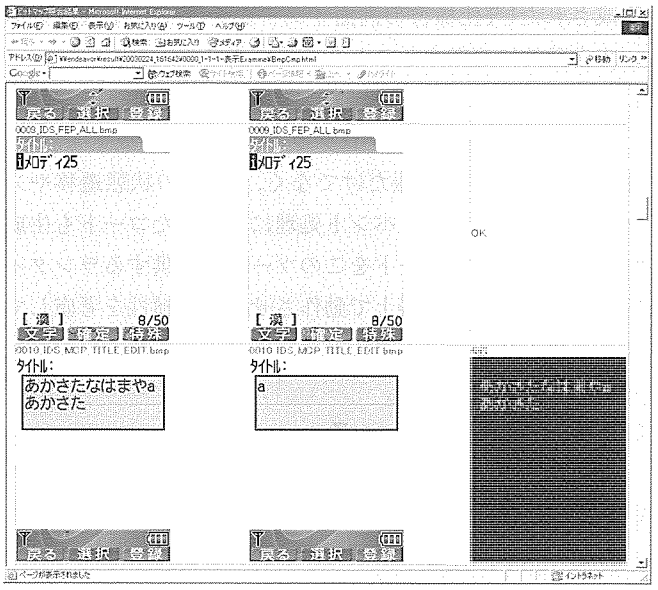


図2．回帰試験におけるビットマップレベルの照合結果

原因調査が容易となった。

現在，多数の試験データを繰り返し自動実行し，ソフトウェア改変による品質低下の検出を行っている。また，画面遷移だけでなく，通信網とのデータ通信などについても，期待されるデータと実データを照合する機能を設け，併せて試験を行っている。

5. むすび

組込み機器におけるUIの生産性向上を目的とした組込み用UI設計ツールの基本概念とその構成，そして，このツールを用いたUI設計開発と，携帯電話への適用例について述べた。

このツールが提供するフレームワークを用いれば，ラピッドプロトタイピングによって，デザイナ，開発者，顧客の間で“動く仕様書”を共有・確認しながら，簡易な上流設計から実機レベルに至るまで，手戻りを最小化しつつ連続的に設計を進めることが可能である。また，SCOの概念に基づくUI設計の部品化及び機種内・機種間の設計再利用により，設計コストの削減や操作感の統一が容易となる。さらに，UI仕様書の更新機能やコード生成機能，回帰試験などのサポートにより，仕様変更に伴うオーバーヘッドを軽減させることが可能である。

今後，このツールの携帯電話への適用を更に加速するとともに，他事業分野への水平展開を推進する予定である。

参考文献

(1) 小中裕喜，ほか：階層的部品定義に基づく組込用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)

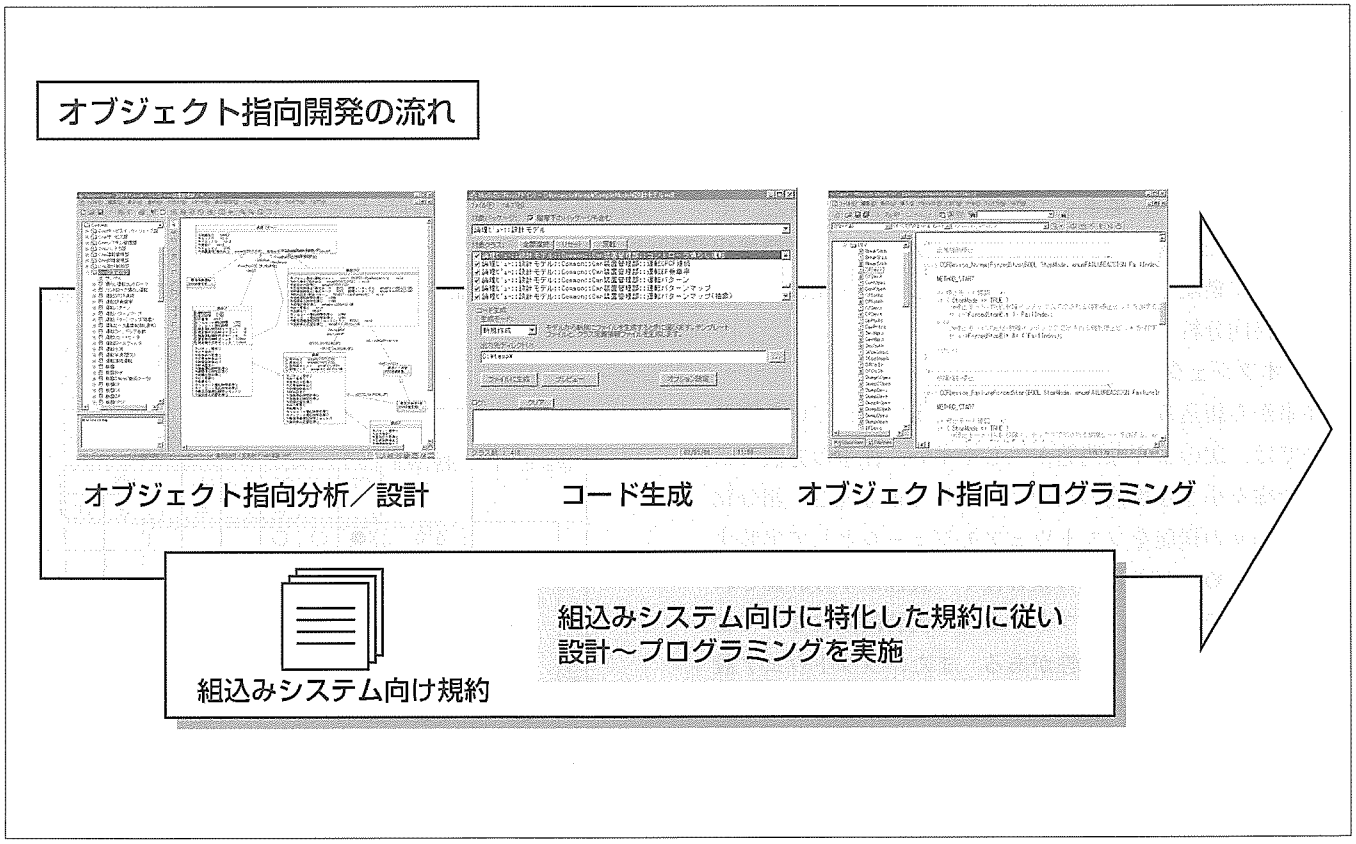
車両空調機器における オブジェクト指向技術適用

風間由美子* 渋谷康雄**
小林 敦* 木村芳美***
大河原 繁*

要 旨

近年，ビジネス分野に限らず，組込み分野においてもソフトウェアの大規模化・多様化が進んでいる。こうした中，仕様のバリエーションへの対応(抽象化・部品化)，多様化する要求への対応(柔軟性)，ソフトウェア保守性の向上といった設計の改善を進めるための基盤技術として，オブジェクト指向開発が注目されている。三菱電機では，鉄道車両用空調制御装置の組込みソフトウェア開発を対象としてオブジェクト指向開発手法を適用するとともに，組込みソフトウェア開発向けのC言語用オブジェクト指向開発環境を構築した。オブジェクト指向分析／設計では業界標準のUML(Unified Modeling Language)でモデルを記述するが，多くの組込みソフトウェアでは実装にオブジェクト指向言語ではないC言語を用いる。実装言語としてC言語を用いる場合，オブジェクト指向設計のアウトプットである設計モデルからプログラムコードへのマッピングは，C++やJava(注1)のようなオブジェクト指向言語を用いる場合に比べて単純ではなく，設計と実装との間で表現される内容にギャップが生じやすい。そこで，C言語においても簡潔で信頼性の高いオブジェクト指向プログラミングを可能にするコーディング規約を策定し，コーディング規約に従ったプログラミングを支援するマクロを開発した。また，UMLの設計モデルからC言語のプログラムコードを生成するツールを開発して，作業の効率化を図るとともに，分析／設計から実装まで一貫したオブジェクト指向開発を可能にした。

(注1) Javaは，米国Sun Microsystems, Inc.の米国及びその他の国における商標又は登録商標である。



オブジェクト指向開発のイメージ

オブジェクト指向分析／設計では，業界標準のUMLでモデルを記述する。C言語において簡潔で信頼性の高いオブジェクト指向プログラミングを可能にするコーディング規約を策定し，併せてUMLの設計モデルからC言語のプログラムコードを生成するツールを開発した。これにより，分析／設計から実装まで一貫したオブジェクト指向開発を可能にした。オブジェクト指向開発により，仕様の変更，機能拡張に強い設計が可能となる。

*三菱電機㈱ 設計システム技術センター **同社 冷熱システム製作所 ***三菱電機エンジニアリング㈱

1. まえがき

近年，組込みシステムの大規模化・多様化が急速に進み，その一方で，開発期間の短縮が求められる傾向にある。こうした環境の中，ソフトウェア設計の継続的な改善によって開発を効率化していくことが従来にも増して重要になってきている。こうした中，仕様のバリエーションへの対応(抽象化・部品化)，多様化する要求への対応(柔軟性)，ソフトウェア保守性の向上といった設計の改善を進めるための基盤技術として，オブジェクト指向技術が注目されている。

本稿では，鉄道車両用空調制御装置の組込みソフトウェア開発を対象にしたオブジェクト指向開発手法の適用と，適用に際して構築した開発支援環境について紹介する。

2. オブジェクト指向開発手法

2.1 オブジェクト指向技術の適用

当社では，鉄道車両用空調制御装置の組込みソフトウェア開発に対して，オブジェクト指向開発手法を適用した。この製品には次のような特徴がある。

- 空調機器としての基本動作は同じであるが，機器構成や細部の動作が機種ごとに異なる。
- 製品世代間で基本的な仕様が継承されるため，ソフトウェアの流用が多い。

このような特徴を持った製品のソフトウェア開発を効率化するため，ソフトウェアモジュールの標準化が従来から進められてきた。しかし，製品の高機能化とソフトウェア規模の増大に伴い，標準化の継続が難しくなってきた。そこで，多様な客先要求に対応しながらソフトウェア標準化を長期的に継続していくことを目的として，開発にオブジェクト指向分析／設計手法を適用した。

2.2 オブジェクト指向開発の利点

従来から組込みソフトウェアで多く用いられている開発方法では，次のような問題が生じやすい。従来の方法では，要求仕様を小さな機能の集合とみなして細分化し，細分化された個々の機能をソフトウェアモジュールとして実装する。このため，要求仕様の整合性や一貫性が十分評価されないまま開発が進み，ソフトウェアの実装段階になって初めて問題が発覚する危険性がある。また，ソフトウェア設計の面では，機能の重複による冗長さや，追跡困難な依存性がモジュール間で生じやすく，再利用性を維持・促進するための継続的な保守が難しいという問題がある。

これに対してオブジェクト指向開発では，ソフトウェアの機能をオブジェクトによる協調動作として解釈し直し，個々のオブジェクトの仕様をクラスとして実装するという特徴がある。このため，概念レベルでモデリングを行うオブジェクト指向分析においては，要求仕様の不整合や冗長さを早い段階で発見して仕様を見直すことが容易である。また，コーディングに近いレベルでモデリングを行うオブジェクト指向設計においては，クラス間依存関係の管理と抽象化などを用いたソフトウェア構造の整理が容易で，ソフトウェア標準化のように長期的な保守が必要になる開発の手法として適している。

2.3 ソフトウェア標準化

空調機器の制御ソフトウェアでは，強冷や弱冷などの運転領域，圧縮機や送風機などの各機器の制御，これら各機器に適用する運転パターン・運転制約条件・保護条件等の構成要素を協調的に組み合わせることで，空調機器としての機能を実現している。ここでは，例として，空調機内の各機器に対する運転パターンの仕様を示して説明する。**図1**に示すように，運転パターンは，運転領域と各機器に対する組合せ表の形で表現される。

この組合せの考え方は，空調機器の仕様として基本的な考え方であり，機種間で共通である。そこで，組合せの仕組みだけを抽出したクラス群を作成して，機種間で再利用可能な共通の枠組みとしている。この枠組みは，組合せの方法と，組合せ対象になる部品が持つべきインタフェースを規定しており，ソフトウェア標準化の共通基盤になる。

また，組合せの対象となる個々の運転領域，機器及び運転パターンのクラスを，それぞれ標準化されたインタフェースに従って作成した。これらのうち，汎用的な仕様を持つものは共通部品として機種間で再利用し，機種特有の動作をするものは機種専用の部品としている。

このように，共通基盤の上に共通部品と個別機種用の部品を必要に応じて組み合わせることによって，個別機種向けのソフトウェアを構成できるようにした。考え方としては，**図2**に示すように，ある機種Aで必要になる共通部品

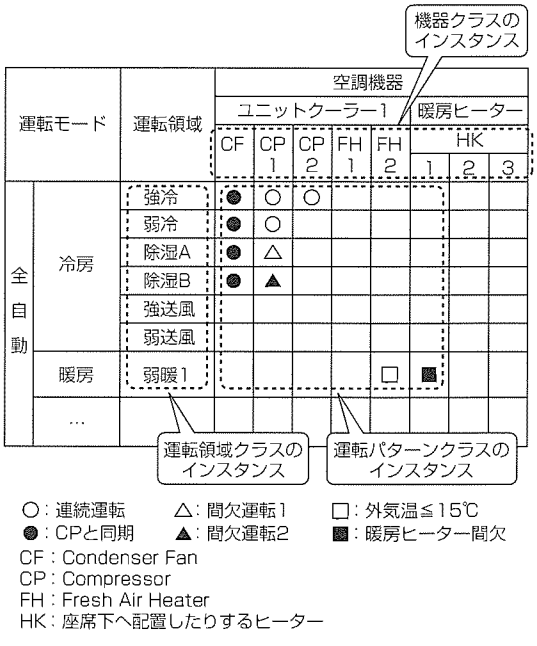


図1．空調機器仕様の例

及び専用部品(クラス)を選択し，そのクラスのインスタンス群を定義することによってソフトウェアを構成する。

3. 開発支援環境

3.1 開発支援環境の構成

オブジェクト指向分析／設計手法の適用を進めるため，従来から開発言語として用いられているC言語を対象としたオブジェクト指向開発支援環境を構築した。C言語はオブジェクトの概念を直接表現できないため，そのための仕組みと，設計モデル要素をプログラムコードへ反映させるルールが必要である。そこで，C言語によるオブジェクト指向開発を可能にするコーディング規約と開発支援ツールを開発し，設計モデル作成からプログラミングまでの一貫した作業環境を実現した。

開発作業の各ステップにおいて使用するツールを**表1**に示す。分析と設計では，オブジェクト指向分析／設計ツールであるRational Rose(注2)(以下"Rose"という。)でモデリングを行う。次に，開発支援ツールで生成したプログラムコードを基にコーディングを行う。テスト及びデバッグは，パソコン上ではMicrosoft Visual C++(注3)を，ターゲット上ではICEを用いて行う。

3.2 コーディング規約

コーディング規約は，比較的小規模な組込みソフトウェアを対象として，オブジェクト指向の設計モデル要素をC言語で表現する方法を規定するとともに，記述を支援するマクロを提供することでC言語によるオブジェクト指向の実装を可能にしている(**図3**)。組込みソフトウェアの特徴を考慮して，設計モデルやコーディングに一定の制約を設けることで，簡潔で効率良い記述を可能にしている。この規約は次のような特徴を持っている。

(1) インスタンスを1つだけ持つクラスと複数のインスタンスを持つクラスを区別して扱う。組込みソフトウェアではインスタンスが1つしか存在しないクラスが多く現れる。このようなクラスは，インスタンスを識別する必要がないため，特に区別して扱うことで関数呼出し時のオーバーヘッドを軽減する。

(2) クラスのインスタンスをすべて静的な変数として定義する。組込みソフトウェアでは，実行時に現れるインスタンスの個数が決まっている場合が多いため，静的な定義を行うことで，メモリリークや断片化といった実行時に発生する問題を回避できる。

(3) インタフェースの継承による操作の多相性をサポートしており，C++言語の仮想関数に類似した記述が可能になっている。また，継承や多相性に関係して起こり得るプログラムコード上の記述ミスをコンパイル時に検出できる。例えば継承した操作を実装する際に起こり得る関数シグニチャの不一致や，関数呼出し時のインスタンスと関数の不適切な組合せを検出できる。

3.3 開発支援ツール

C言語によるオブジェクト指向開発の支援ツールとして，クラス定義ファイル生成ツールと，システム構成ファイル生成ツールの2つを開発した。

3.3.1 クラス定義ファイル生成ツール

クラス定義ファイル生成ツールは，**図4**のようなRose上で記述したUMLのクラス図から**3.2節**で述べたCコーディング規約に沿ったC言語のスケルトンコードを生成するツールである。スケルトンコードとは，設計モデルから機械的に生成可能なクラス定義部分のプログラムコードであり，処理ロジックのコードを追加記述することで最終的なプログラムコードとなる。元々RoseにはC++やJavaなどのオブジェクト指向言語に対するスケルトンコードを生成する機能があるが，C言語のコードを出力する機能はないため，Roseの拡張インタフェース機能を使ってクラス定義情報を取得し，そこからC言語のスケルトンコードを出力

(注2) Rational Roseは，米国Rational Software Corp.の登録商標である。

(注3) Microsoft，Microsoft Visual C++，Microsoft Excelは，米国Microsoft Corp.の米国及びその他の国における商標又は登録商標である。

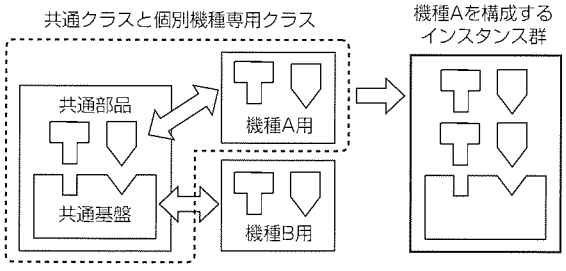


図2．クラスの選択とシステム構成の考え方

表1．開発ツール

開発作業	使用ツール
分析	Rational Rose
設計	Rational Rose
コーディング	クラス定義ファイル生成ツール システム構成ファイル生成ツール Microsoft Visual C++
テスト・デバッグ	Microsoft Visual C++(パソコン上) ICE(ターゲット上)

```
void CItemGroup_Save (OBJECT (IStream) out)
{
    METHOD_START          下線部がオブジェクト指向
                          支援マクロの使用部分
    LONG i;
    WITH (out).Write (this->ItemsCount);

    for (i=0; i<this->ItemsCount; i++) {
        OBJECT (IItem) item=this->Items [i];
        WITH (item).Save (out);
    }

    this->Modified=FALSE;
}
```

図3．規約に従ったプログラムコード例

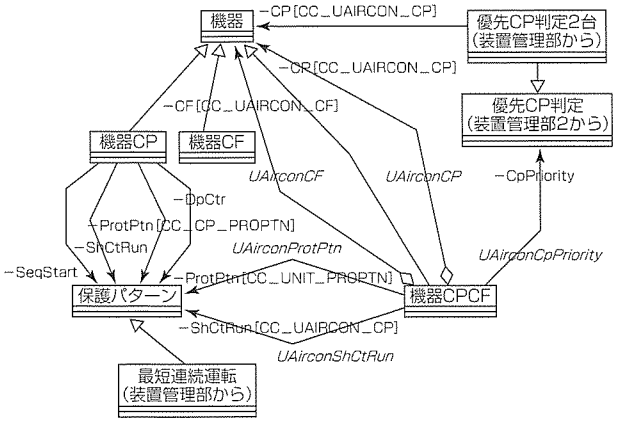


図4．クラス図

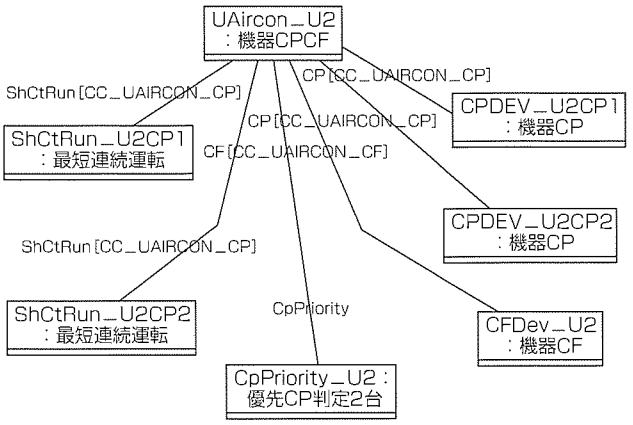


図5．オブジェクト図

する機能を実現した。

クラス定義ファイル生成ツールは，設計モデル修正時の再生成にも対応している。つまり，ユーザーが一度コード生成を行って，処理ロジックのコードを記述した後にRoseの設計モデルを修正し，再度コード生成をした場合，再生成したコードに，前にユーザーが実装したコードが引き継がれる。このツールを使用することにより，コーディング作業を効率化するとともに，設計モデルとプログラムコードの一貫性を機械的に保証することで，モデリングを中心にした開発が可能となった。

3.3.2　システム構成ファイル生成ツール

システム構成ファイルは，システムに存在する静的なインスタンスの定義(ソフトウェア部品の組合せ)を記述するC言語のプログラムコードであり，インスタンスが持っている属性の初期値，インスタンス間の参照関係を表すデータの初期値をこのファイルで記述する。車両用空調制御装置の組込みソフトウェア開発では，従来このファイルを約8,000行のC言語のプログラムコードとして手作業で作成していたため，システムとしての可読性や作成時の作業効率の悪さが問題となっていた。システム構成ファイル生成ツールは，これまで手作業で行ってきたシステム構成ファイルの作成を半自動化するツールである。Rose上で記述した**図5**のようなインスタンス間の参照関係を表すオブジェクト図からMicrosoft Excel[注3](以下"Excel"という。)のフォーマットのシステム構成テーブルを生成し，さらに，ユーザーがExcel上でインスタンスの属性初期値と参照関係を入力した後，システム構成テーブルからシステム構成ファイルを生成する(**図6**)。システム構成テーブルにおけるインスタンスの参照関係の入力では，参照先として有効なインスタンスのリストから選択すればよい。システム構成ファイル生成ツールにより，従来C言語のテキストで記

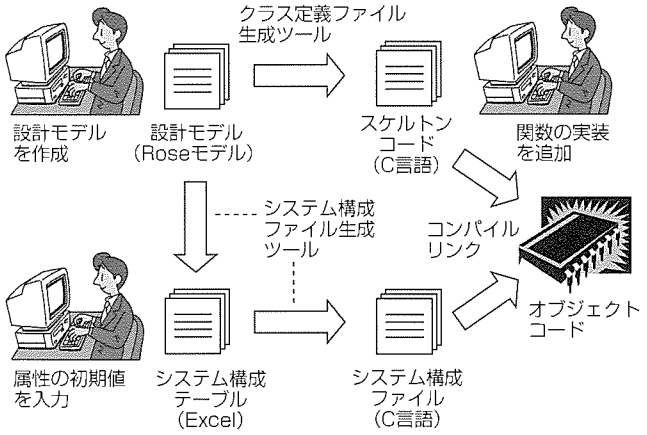


図6．開発支援ツールを用いた開発の流れ

述していたシステム構成をExcelの表上で単に数値を入力するか又はリストから選択すればよくなった。システム構成ファイル生成ツールはクラス定義ファイル生成ツールと同様に設計モデルの修正時の再生成にも対応しており，設計モデルを修正した後に再生成しても，前に入力した値は再生成されたシステム構成テーブルに引き継がれる。システム構成ファイル生成ツールにより，煩雑な記述作業を省力化するとともに，記述ミスを防止し，信頼性を向上させることが可能となった。

4. むすび

本稿では，車両空調制御装置の組込みソフトウェアを対象にしたオブジェクト指向技術の適用と，その開発を支援する環境であるコーディング規約及びコード生成ツールについて紹介した。組込みソフトウェアの分野においてもオブジェクト指向開発手法は普及が進んでおり，今後も，適切な開発環境の整備を進めながら開発現場への展開を図っていく。

列車情報管理装置のソフトウェアプロダクトライン

吉田　実*　重枝哲也***
辰巳尚吾**　角南健次***
遠藤義雄**

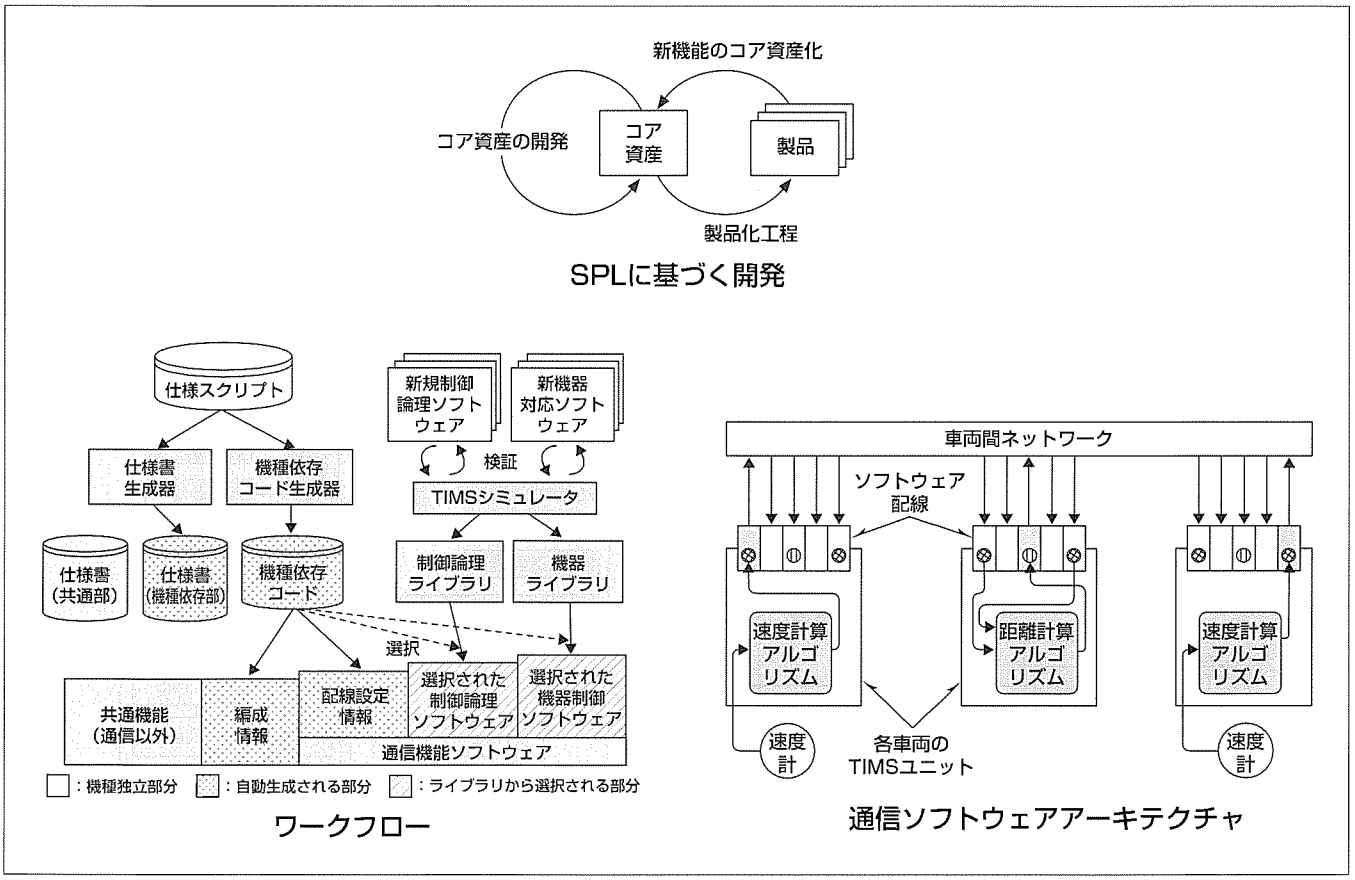
要　旨

列車統合管理装置(Train Integrated Management System：TIMS)は，列車全体の機器を監視・制御するシステムであり，ソフトウェアが重要な役割を担っている。TIMSは，高い信頼性とハードリアルタイム性が要求される分散システムである。そのソフトウェアは機種ごとの差異が大きく，従来は機種個別の開発が必要となっていた。

TIMSのソフトウェア開発にソフトウェアプロダクトライン(SPL)という考え方を取り入れ，複数の製品を効率的に製作するため，ソフトウェアの開発方法を含め再構築した。まず，SPLの考え方に基づき詳細なドメイン分析を行い，TIMSの基本的な考え方を整理し，共通する部分と機種により変化する部分を切り分けた。その分析に基づき，共通部と機種依存部を分離可能なソフトウェアアーキテクチャの設計及びそれを実現する共通部ソフトウェア，設計・製作の一部を自動化するコード生成ツールの構築を行い，個々の機種の設計・開発で必要になる工程を効率化した。さらに，顧客ごとの要求分析を円滑に行えるよう標準機能仕様書を作成した。この仕様書は，機能ごとに分類されており，標準機能としての考え方と特定の機種に適用するに当たって変更できる項目が記述されている。

また，従来の実機ハードウェア上での開発は，大きな設置スペースを必要とし，個々のソフトウェア開発者が日常の開発で扱うには問題があった。そこで，開発工程で実機とその設置スペースの確保の問題を軽減し，開発者によりよい環境を提供し，効率的な試験ができるよう，パソコン上で動作するTIMSシミュレータを開発した。

SPLを基本にしたこれらの総合的な取り組みにより，開発効率の大きな向上と信頼性の確保が達成できると考える。



TIMSのソフトウェアプロダクトライン

SPLでは，共通のコア資産から複数の機種の製品が製作される。コア資産は，初めから再利用することを念頭に開発される。製品開発で共通性の高い機能が実現されれば，コア資産に取り込む(上図)。再利用性を高めるため，共通部と機種依存部を切り分けたソフトウェアアーキテクチャ(右下図)と機種依存部における作業を一部自動化するワークフロー(左下図)を構築した。自動化により人為的なミスの可能性を抑えることができ，また，実績のあるソフトウェアモジュールを再利用することができ，信頼性向上に寄与する。

*先端技術総合研究所(工博)　**同研究所　***交通システム事業所

1. まえがき

TIMSは，列車全体の機器を監視・制御するシステムであり，ソフトウェアが重要な役割を担っている高い信頼性が要求される分散ハードリアルタイムシステムである。複数のTIMS機種間では，機能の類似性は高いものの，列車の構成，機器や列車運用の多様性から，ソフトウェアは毎回作り直しになっている。今回，ソフトウェアプロダクトライン[1]の考え方に基づき，TIMSの製品群を初めから分析し直し，要求機能の本質的に変わらない部分と変更される部分について検討した。その結果に基づき，ソフトウェアアーキテクチャ，開発のワークフロー，基本となるソフトウェア，仕様書の構成と内容など，ソフトウェア自体とソフトウェアを開発する仕組みを再構築した。

本稿では，TIMSのソフトウェアプロダクトライン適用について紹介する。

2. TIMSとソフトウェアプロダクトライン

2.1 TIMSの特徴

TIMSの製品系列には，機器の監視を主機能とした列車モニタ，列車の制御も行うTIS(Train control Information management System)，列車全体でブレーキ力を適切に制御する機能などの高度な制御を行う狭義のTIMSがあるが，本稿では，これらをTIMSと総称することにする。列車には多様な機器が多数搭載され，それらはリアルタイムの監視・制御が必要である。**図1**にTIMSの構成と監視・制御対象機器の一部の例を示す。

衝突防止などの安全性についてはATS(自動列車停止装置)などの保安装置が担うものの，モータ・ブレーキ制御，デッドマン(運転士の異常を監視する)など重要な制御も行っており，高い信頼性と安全性が求められるシステムである。今日の過密ダイヤでは1つの列車の障害が多くの列車の運行に影響するため，高い稼働率も必要である。さらに，複数の編成が運行時に結合する併結時には，それぞれ一体となって動作している2つの分散リアルタイムシステムを動的に結合する必要がある。

TIMSに要求される機能は，概略レベルでは比較的共通性が高いものの，車両数，機器の種類・数や配置，機器の通信インタフェース，顧客ごとの列車運用に対する考え方が異なるため，毎回，作り直しと試験が必要になっていた。

2.2 ソフトウェアプロダクトラインとTIMSの適合性

SPLは，最近注目されているソフトウェア開発手法である。共通性のある複数のソフトウェア製品を1つのプロダクトラインから効率的に製作するための総合的な取り組みである。TIMSは，以下の理由でSPLに特に適していると考えられる。

(1) 要求仕様の機種ごとの差異は大きいが時間軸では比較的安定している。また，新機能の導入ペースが緩やかである。

(2) 年間受注件数が適切な範囲内にある。

(3) 高い信頼性が要求される。

SPLを採用することで再利用の範囲と粒度が大きくなり，その結果，ソフトウェア開発効率と製品の信頼性向上が期待できる。

2.3 ソフトウェアプロダクトラインの立ち上げ

まず，TIMSという製品をSPLの考え方に基づいて詳細

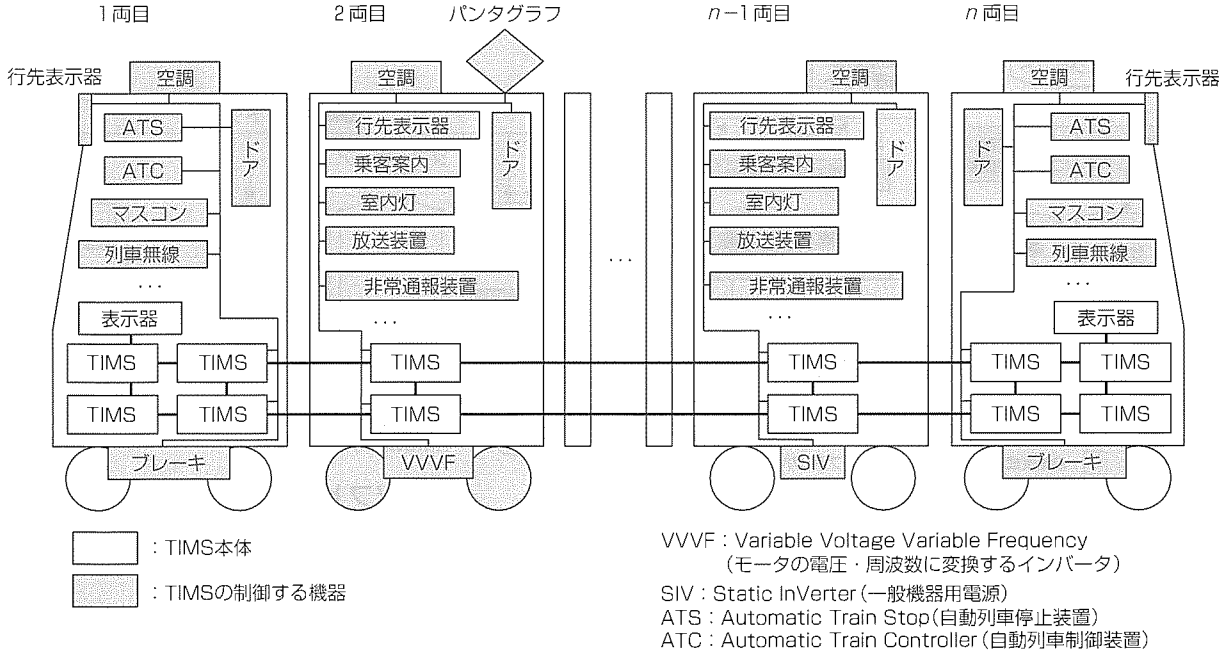


図1．TIMSの構成例

に再分析することから始めた。分析の初期段階で，顧客ごとに同じ概念が別の用語で呼ばれていることが分かった。そのため，最初に共通機能を考えるために，用語を整理し，分析中は必要に応じて用語の見直しを行った。また，ユースケース分析，標準機能分析をSPLの視点で行った。これらの分析は，ある特定の顧客，特定の機種でなく，広く各機種を調査する必要があるため困難な作業となったが，キーマンが継続的に参加し分析作業を進めた。この分析作業に基づき，3章のソフトウェア関連のコア資産が生まれた。

分析は，UML(Unified Modeling Language)などを用い，オブジェクト指向設計を採用した。これにより，モジュール性の高い堅牢(けんろう)なソフトウェア設計・製作と変更ポイントの分離が容易になった。

なお，柔軟で再利用可能なソフトウェアは，その柔軟性と引き換えに，わずかながら計算資源とメモリを余分に消費する場合が多い。一方，ハードウェア性能の制約が大きい場合，性能チューニングが行われることになるが，たいていの場合ソフトウェアの可読性や再利用性が犠牲になってしまう。TIMSのSPLでは，再利用性が高く，保守が容易なソフトウェアを作るために，能力の限界に近づきつつある現在のハードウェアを強力な新ハードウェアで置き換えることとし，ソフトウェアのコア資産開発と並行してハードウェアの開発も進めた。

3. コア資産

コア資産は，SPLにおいて製品を生み出すための共通の基盤になるものである。コア資産は，ソフトウェアそのものだけでなく，製品開発で共通に必要になるものすべてが含まれる。再利用可能なソフトウェア自体はもちろんのこと，ソフトウェアアーキテクチャ，標準仕様書やその他の文章，ソフトウェアプロセス，ツールなども含まれる。さらには，組織の在り方，計画の立案法なども含む広範な概念である。以下の節では，TIMSのSPLの特長的な点について概要を説明する。これらの総合的な取り組みにより開発効率が向上できると考えている。

3.1 ソフトウェアアーキテクチャ

TIMSは，従来ハードウェアが行っていたことを少しずつソフトウェアに取り込みながら，新しい機能を徐々に追加してきた。例えば，単純な車両間配線をTIMSのネットワークで置き換えることで，車両間の引き通し線の数を大幅に削減できた。また，監視制御する機器が増えることで，列車の統合的な監視制御，ナビゲーションや車上試験などの運用支援機能が可能になった。

分析の結果，TIMSのソフトウェアで最も基本的かつ変化しやすい部分は従来ハードウェアで行われていた車両間配線をソフトウェアで実現している部分であることが判明した。車両間の配線は，現在は車両間ネットワークを通じて通信されるパケット中のデータとして実現されている。しかし，設計者は，パケットの厳しい容量制限と要求されるデータ量，許容遅延時間，信頼性のトレードオフを毎回の設計で行う必要があった。また，その設計に基づいてソフトウェアも作り直されていた。このアーキテクチャの特長は，これらをソフトウェア配線としてとらえることである。伝送遅延，伝送量など要求が異なっても，線として抽象化することで，線を利用するソフトウェアは汎用性が高くなる。

ソフトウェア配線が接続されるのは，アルゴリズムクラスと呼ばれるTIMSの最小機能を実現するソフトウェアモジュールである。ある1つの機能は，アルゴリズム又は複数のアルゴリズムの組合せで実現される。

図2に配線の例を示す。速度計は列車に2個付いており，その出力はそれぞれの速度計算アルゴリズムの入力になる。速度計算アルゴリズムが計算した速度の出力は，ネットワーク上を配線されたかのように，距離計算アルゴリズムの入力につながれる。

従来は，制御プログラムが必要な情報を直接アクセスしていた。このため，機器の配置や数の変更の影響を直接受けていた。配線方式では，アルゴリズムと機器の入出力は，配線によって間接的に結び付けられる。アルゴリズムの配置と配線は機種に依存した情報であり，3.3節で説明するコード生成ツールによって作られる。この方式によりアルゴリズムは入出力対象から切り離され，再利用性が高くなる。

3.2 標準機能仕様書

SPLでは，製品の機能の変更可能な範囲を適切に定めることが重要である。この範囲が狭すぎると，製品の適用範囲を限定することになり，顧客の一般的なニーズに対応できなくなるおそれがある。逆に，広すぎると，効率的な機能の取扱いが難しくなり，SPLが破綻(はたん)する原因にもなる。

今回，ドメイン分析の結果に基づき，TIMSの各機能それぞれに対して標準として何が求められているか検討しな

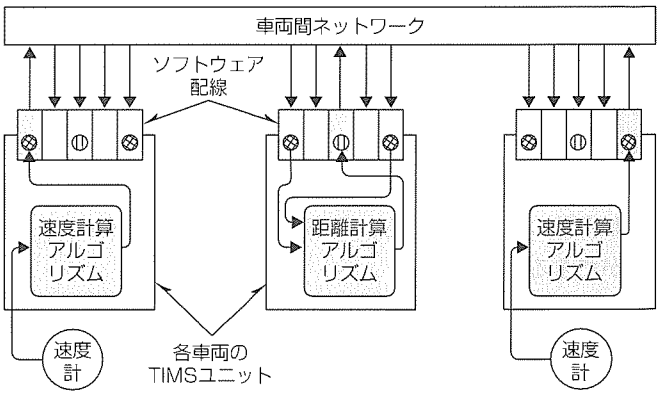


図2．ソフトウェア配線によるアルゴリズムの接続例

がら，標準機能仕様書を作成した。標準機能仕様書は，個々の機種で変更になる接点名，パラメータの値などを個別に指定できるスタイルになっている。また，根本的に考え方が異なり，1つの仕様書にまとめると理解が難しくなる機能は，複数の仕様書に分割した。

なお，標準機能仕様書は製品に対する顧客の要求を明確にすることが第一の目的であるので，標準機能仕様書とその機能を実現するソフトウェアの実装は，1対1に対応するとは限らない。

3.3 設計・製作のワークフローとコード生成ツール

従来，設計者が設計書を作り，それをソフトウェア開発者が読んでソフトウェアを製作していた。これを，図3に示すワークフローに変更した。設計者は，ツールを用いて電子的な設計データである仕様スクリプトを作る。仕様スクリプトは，編成情報，配線情報，アルゴリズム選択／パラメータ設定情報などから構成され，コード生成ツールにより，機種対応部のソフトウェアになる。このワークフローにより，開発が効率化されるとともに，設計からソフトウェアへの変換が自動化されるため，仕様からの転写ミス，仕様の誤解などの人為的なミスを防ぐことができる。

3.4 シミュレータ

組み込みソフトウェア開発で，実機ハードウェア用のソフトウェアがそのまま実行・検証できるシミュレータが注目されてきている。TIMS開発においても，従来実機ハードウェア上で製作・試験を行ってきたが，開発効率を高めるため，パソコン上で動作し，設計，製作，試験の大部分に適用できるシミュレータを開発した。特に，設置スペースが大きく列車全体のハードウェア確保が困難であったため，シミュレータは大きな効果を発揮すると期待される。

このシミュレータは，ソフトウェアが実用的な速度で動く範囲で，可能な限り実機に近い動作をするように設計されている。ただし，非決定的なタスクの実行タイミングや実行性能に関するシミュレートは正確でなく，また低レベルのハードウェア処理は直接シミュレートしていない。そのため，実機上での検証作業は大きく削減できるが，全く不要になるわけではない。

また，1つのTIMSの総CPU数は60個を超えることがあるため，スケーラビリティにも配慮した。システム構成が大きい場合でも負荷の重い全機能同時実行のソフトウェア検証ができるよう，パソコン複数台での実行をサポートしている。したがって，1台のパソコンでも，編成車両数と同じ台数のパソコンでも，列車全体をシミュレートすることができる。

なお，SPLは機種当たりのソフトウェア開発を効率化することが目的であり，機種ごとのシミュレータ構築に時間

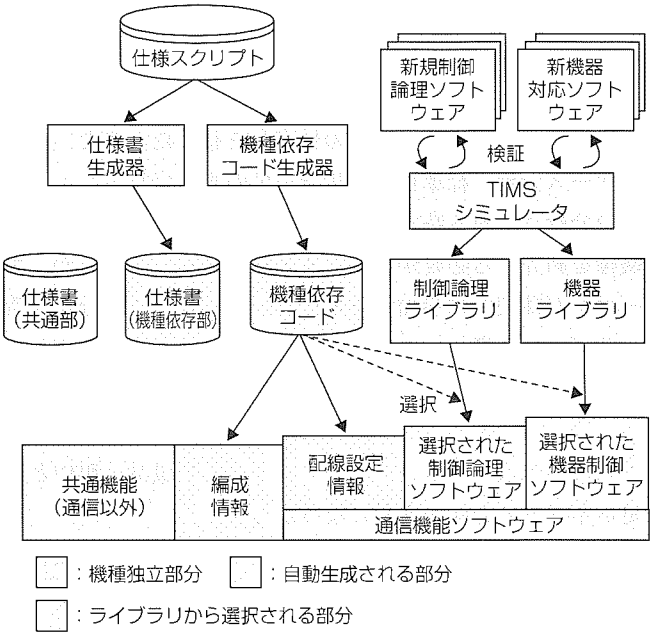


図3．設計・製作ワークフロー

がかかるとシミュレータの意義は大きく損なわれる。しかし，変化するのは機器と制御の方法が主であり，TIMS自体のハードウェアはバリエーションが多くはない。シミュレータはTIMSハードウェアを忠実にシミュレートするように作成するだけでよく，TIMS上で動作するプログラムや機器のエミュレータなどは実機用と同じものが使える。したがって，シミュレータを利用するためだけに必要になる機種ごとの作業はほとんどない。

3.5 試験フレームワーク

オブジェクト指向設計とシミュレータの導入により，効率的な単体・統合試験が可能になった。クラス単体試験のためのフレームワークを用意し，コードを修正したときは瞬時に試験できるようにし，常に試験をしながらインクリメンタルに開発できるようにした。

4. むすび

本稿では，SPLという考え方に基づいたTIMSの製品開発法の再構築について概要を紹介した。SPLの適用により，仕様決定やソフトウェア製作が容易になり，実績があるソフトウェアが再利用できることで信頼性が高くなることが期待できる。また，初めてTIMSを導入する場合でも，標準機能を使うことで，運用等のノウハウを含め，これまでの実績のある機能を自然かつ円滑に利用できる。

参考文献

(1) Clements, P., et al.：Software Product Lines — Practices and Patterns —，Addison-Wesley（2002）

WindowsAPIのUNIX環境へのPorting技術

佐藤重雄* 河井弘安**
金田典久*
高山茂伸*

要 旨

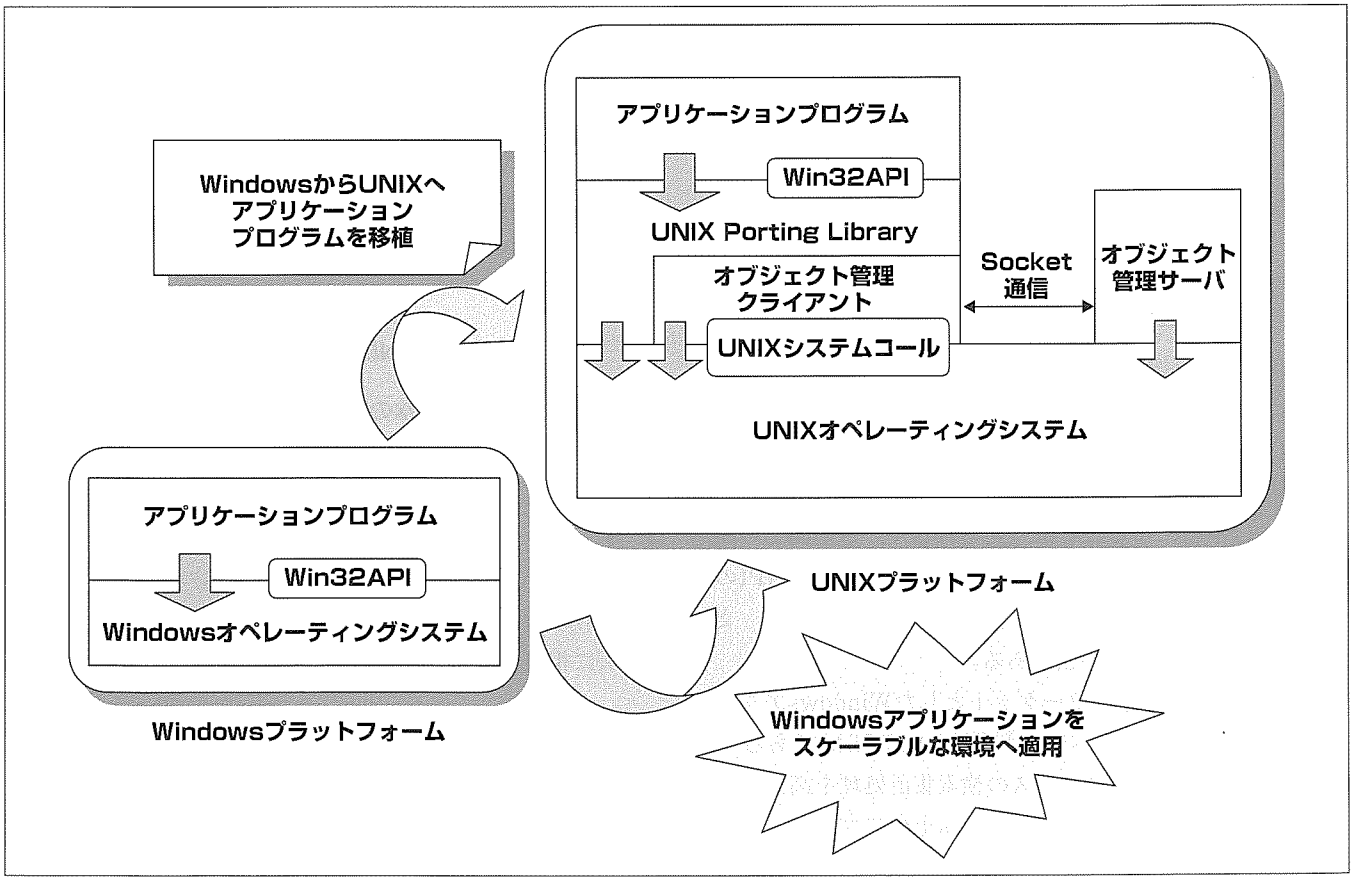
Microsoft社のOS(Operating System)であるWindows(注1)の普及により，パソコンプラットフォームにおいて，Win32API(Windows 32bit Application Program Interface)と呼ばれるWindowsが提供するインタフェースを使用したアプリケーションが作成されてきた。一方，UNIX(注2) OSは，プロセッサ，メモリの拡張性が高く大規模システムに向いており，Windows上で開発されたアプリケーションプログラムをそのような大規模システムで使用するためにはUNIX環境に移植(Porting)する必要がある。

ところが，Win32APIはWindowsが提供するインタフェースであるため，UNIX環境に移植するためには，プログラムコードの大幅な修正が必要となり，開発コストを要するという問題があった。

そこで，Windows上の製品をUNIX環境に移植する開発において，既存コードを極力再利用し，開発期間を短縮するために，Win32APIをUNIX環境上でエミュレートするライブラリを開発した。このライブラリは，Win32APIで提供する機能をUNIX上で提供するライブラリである。

本稿では，Win32APIをUNIX上でエミュレートするために開発したUNIX Porting Library，及びWindows上でオブジェクトとして扱われているイベントなどを管理するオブジェクト管理機能の仕組みについて述べる。また，実際にWindowsからUNIXに移植を行ったアプリケーションに対して，その効果を評価した結果についても述べる。

(注1) Windowsは，米国Microsoft Corp.の米国及びその他の国における商標又は登録商標である。
(注2) UNIXは，米国The Open Groupの登録商標である。



WindowsアプリケーションのUNIX環境への移植

Win32APIを使用して作成されたアプリケーションプログラムをUNIX環境に移植する場合にWin32APIの機能をUNIX上でエミュレートするシステムを実現した。このシステムでは，UNIX Porting Library，及びオブジェクト管理機能の提供により，プログラムコードのオリジナル部分には手を加えることなく移植を実現できる。

*三菱電機㈱情報技術総合研究所 **三菱電機インフォメーションテクノロジー㈱

1. まえがき

Windows上で動作するアプリケーションは，Win32APIと呼ばれるアプリケーションインタフェースを使用して作成される。このWin32APIは，ファイル，プロセスなどの制御を行う機能を提供するWindowsのインタフェースである。

Win32APIを使用して作成されたアプリケーションをプロセッサ，メモリを大量に使用する大規模なシステムに適用するためには，WindowsよりもスケーラビリティをもつUNIX環境に移植することが行われている。ところが，Win32APIとUNIX OSが提供するAPIは異なるため，通常，プログラムコードの大幅な修正が必要となっていた。

本稿では，Windows上のアプリケーションDIAPRISM(注3)をUNIX環境へ移植する開発の際に使用した手法，及びその評価結果について述べる。

2. UNIXへの移植

2.1 課　題

Windowsのアプリケーションを UNIX環境に移植する場合，以下の2つの方法が考えられる。

(1) Windowsアプリケーションのプログラムコードを直接UNIX用のプログラムコードに書き換える方法

(2) WindowsのAPIをUNIX上でエミュレーションし，アプリケーションプログラム自体には極力手を加えない方法

(1)の方法では，アプリケーション自体の動作や実現方式などを詳細に理解した上で，プログラムコードの書換えを行う必要がある。しかし，多くの開発者によって開発された大規模プログラムを短期間かつ少人数で移植を行うためには，アプリケーションプログラム自体をUNIX用に書き換えるという方法は現実的には不可能である。

一方，(2)の方法は，アプリケーションプログラム自体をUNIX用に書き換える方法に比べて，アプリケーションプログラムとOSの間にエミュレーション層があるために性能的には不利になることは考えられるが，短期間での開発を実現することが可能である。このようにWin32APIのエミュレーションをUNIX上で実現する試みとしては，文献(1)に述べられている手法がある。

今回UNIXへの移植のターゲットとしたWindowsアプリケーションは，データベース製品DIAPRISMである。DIAPRISMは，データベースの検索集計処理を高速に実行するシステムであるが，Windows上のアプリケーションであるために，テラバイト規模の大規模システムに対しては適用が困難であるという問題があった。そのため，WindowsよりもシステムのUNIX上で動作することが早急に望まれていた。

(注3) DIAPRISMは，三菱電機㈱の登録商標である。

そこで，DIAPRISMをUNIX上へ短期間で移植するために，上記(2)の方法，すなわち，WindowsのAPI(Win32API)をUNIX上でエミュレーションする方法により移植を実現した。文献(1)で述べられている手法は，移植するWin32APIの範囲を限定した実現方式であり，DIAPRISMが使用しているすべてのWin32APIが対象となってはいない。そのため，本稿では，別のエミュレーション方法により移植を実現した。

2.2 方　針

移植の対象とするWin32APIは，DIAPRISMにおいて使用されているAPI及び機能に限定するものとする。

移植に当たっては以下の方針で行った。

(1) アプリケーションプログラムコードのオリジナル部分には極力手を加えず，Win32API部分以下をライブラリの形で実現する方法を採る。

(2) アプリケーションの全機能が実現可能となるようにWin32APIの移植を行う。つまり，Windows上のアプリケーションからWin32APIの関数が呼ばれた場合と，UNIX上に移植したアプリケーションからWin32APIの関数が呼ばれた場合に，ライブラリが提供する機能には差が生じないようにする。

(3) 実行性能は，同じプロセッサ数，同程度のクロック数，実装メモリ環境において，Windowsで実行した場合とほぼ同程度の性能を実現する。また，UNIXの特性を生かすために，プロセッサ数の増加に応じて性能が向上する実現方式とする。

図1に，この手法での全体システム構成を示す。UNIX Porting Libraryの部分が今回の移植において開発した部分である。

3. UNIX移植システム

3.1 WindowsとUNIXの相違点

Win32APIは，その機能から以下の2つに分類される。

(1) UNIX OSに同等の機能を持つシステムコールが存在するAPI

(2) WindowsのAPIであり，UNIX OSのシステムコールに対応できないAPI

(1)に相当するAPIは，**図2**に示すように，Win32APIの

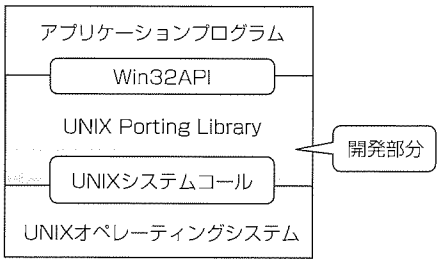


図1．システム構成

機能をライブラリの形式(UNIX Porting Library)として実装し，そのライブラリ内で単純にシステムコールに対応させることにより移植を実現できる。表1の分類では，メモリやレジストリに関する操作がこの場合に相当する。

一方，(2)に相当するAPIでは，UNIX Porting Libraryだけでは単純に対応することはできない。特に，ファイル，プロセス，イベントなどはWindowsではWindowsオブジェクトとして扱われ，そのオブジェクトを識別する識別子(HANDLE)はUNIX OS上には存在しない。したがって，HANDLEを扱うAPIに対しては考慮が必要となる。また，プロセス-スレッド間の同期に関するAPIもUNIX OSには単純に対応するものがない。

そこで，今回の移植に当たっては，オブジェクト管理機能を開発することにより，HANDLE，及びプロセス-スレッド間の同期を実現した。オブジェクト管理機能の設計に当たっては，以下の方針の下に実施した。

- すべてのWindowsオブジェクトは，その識別子であるHANDLEによって扱えるようにする。
- プロセス-スレッド間の同期をとる関数においては，Windowsと同様に，種類の異なるオブジェクトが複数混在しても扱えるようにする。

3.2 オブジェクト管理機能

オブジェクト管理機能は，オブジェクト管理サーバとオブジェクト管理クライアントから構成される(図3)。オブジェクト管理機能では，表1に記述するWindowsオブジェクトを管理する。

オブジェクト管理サーバは，システムに1つだけ存在するデーモンプロセスとして実装し，すべてのオブジェクトの管理を行う。オブジェクト管理クライアントは，Win32APIを使用するプロセスにそれぞれライブラリの形式でロードされ，オブジェクト管理サーバと通信を行う。オブジェクト管理サーバとオブジェクト管理クライアント間の通信はソケット通信を用いて実装し，クライアントはスレッドごとにサーバと通信を行えるようにする。

図4のフローチャートは，Unix Porting Libraryとオブジェクト管理機能を用いて，オブジェクトを生成するWin32APIを実装した処理について示したものである。

表1．Win32APIの分類

Win32APIの分類	オブジェクト管理機能	
	使用の有無	管理するオブジェクト
同期	使用する	イベント／セマフォ／ミューテックス
プロセス・スレッド	使用する	プロセス／スレッド
プロセス間通信	使用する	ファイルマッピング／名前なしパイプ／名前付きパイプ
ファイル	使用する	ファイル(ディレクトリ)
メモリ	使用しない	なし
レジストリ	使用しない	なし
セキュリティ	使用しない	なし
RPC	使用しない	なし
文字列操作	使用しない	なし
時間関数	使用しない	なし
デバッグ関数	使用しない	なし

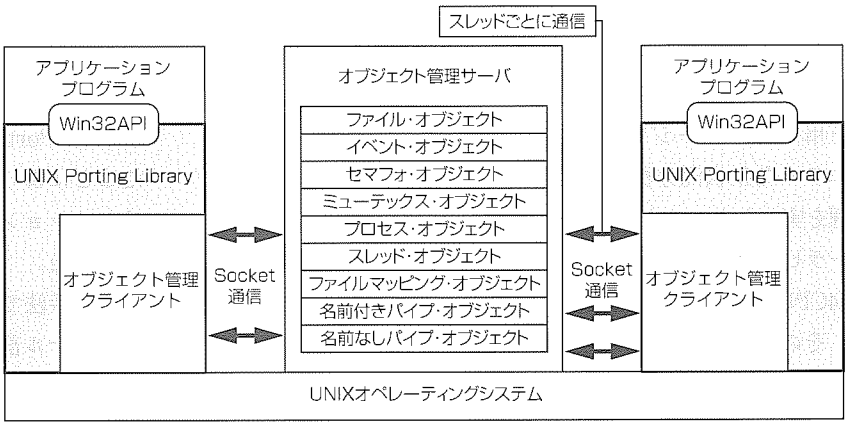


図3．モジュール構成

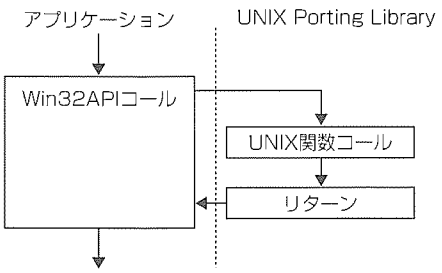


図2．単純に移植できるWin32API動作例

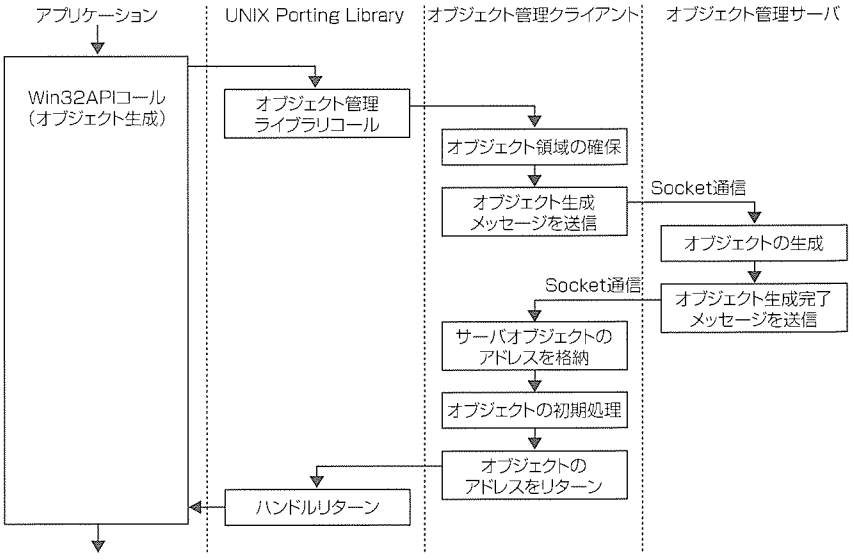


図4．オブジェクトを生成するWin32API動作例

オブジェクトをこのようにオブジェクト管理機能を用いてライブラリの形で実現することにより，プログラムコードのオリジナル部分には手を加えずに移植が可能となり，プログラムコードの再利用性を向上させることができる。

4. 効果（評価結果）

4.1 機能評価

この手法により移植を行ったDIAPRISMにおいて，Windows版が持つすべての機能が正常に動作することを評価するために，Windows版での評価に使用している全試験セットをUNIX版DIAPRISM上で実行した。その結果，すべての機能が正常に動作することを確認し，この手法によりDIAPRISMが使用するWin32APIの機能がUNIX上で実現されたことが検証された。

4.2 性能評価

この手法により移植を行ったDIAPRISMのUNIX上での性能評価を実施した。評価は，大規模システムでのスケーラビリティを検証するために，hp superdome(注4)を使用してWebサイトアクセスログ分析の検索・集計性能を計測した。

計測モデルとしては，1日5,000万ページビュー相当規模のWebサイト1年分に当たる約10億件のアクセスログを用いて，ページアクセス数分析，リンク元URL分析，エラー発生状況分析など，典型的な8種類の検索・集計処理を実行した。また，プロセッサ数を変化させた場合のスケーラビリティを評価するために，4CPU，8CPU，16CPU，64CPUの環境でそれぞれ計測を実施した。

その結果，4CPU環境ではWindows版と同程度の性能が達成されること，また，**図5**に一例を示すように，64CPUまで，プロセッサ数の増加に比例して，検索処理における単位時間処理件数が向上することが確認できた。

しかしながら，一部の検索・集計処理については，十分な性能が達成できないことも判明した。その原因は，並列・パイプライン処理の同期制御を行うWin32APIのエミュレーションにおいて，API呼出しのたびに，オブジェクト管理サーバとオブジェクト管理クライアント間のソケット通信が発生するためである。この問題を解決するため，上記同期処理のAPI呼出し回数が特に多い一部分に関しては，プログラムコードのオリジナル部分の一部を直接修正することにより性能の向上を図った。この修正は，全体の

(注4) superdomeは，ヒューレット・パッカード社の商標である。

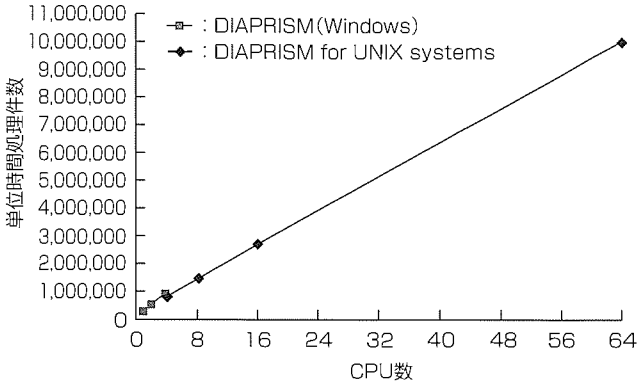


図5．“Webログ分析”性能比較

1%以下であるため，書換えは極力少なくすることができる。

以上の結果から，Windows環境からUNIX環境へのDIAPRISMの移植において，UNIX版DIAPRISMは，

(1) Windows版と同程度の性能が得られること
(2) 大規模なハードウェア構成でも，DIAPRISMがスケーラビリティを実現していること

が確認できた。

5. むすび

Unix Porting Libraryとオブジェクト管理機能を開発したことにより，WindowsのAPIを使用して作成されたアプリケーションプログラムをUNIX環境に移植する一手法を確立した。

この手法では，アプリケーションのプログラムコードのオリジナル部分には極力手を加えないため，Windows対応のプログラムコードの再利用性を高めることが可能となる。また，この手法を用いることにより，UNIX環境が持つスケーラビリティを活用でき，Windowsアプリケーションを大規模システムに適用する手段となり得ることが確認できた。

今後は，この手法を他のシステムに適用した評価を実施し，オブジェクト管理機能の強化を進めていく所存である。

参考文献

(1) Sven M. Paas, et al：Win32API Emulation on UNIX for Software DSM，2nd USENIX WindowsNT Symposium（1998）

ソフトウェア資産活用に有効なバイナリートランスレーション技術

平岡精一* 山崎弘巳**
近江谷康人*
西川浩司*

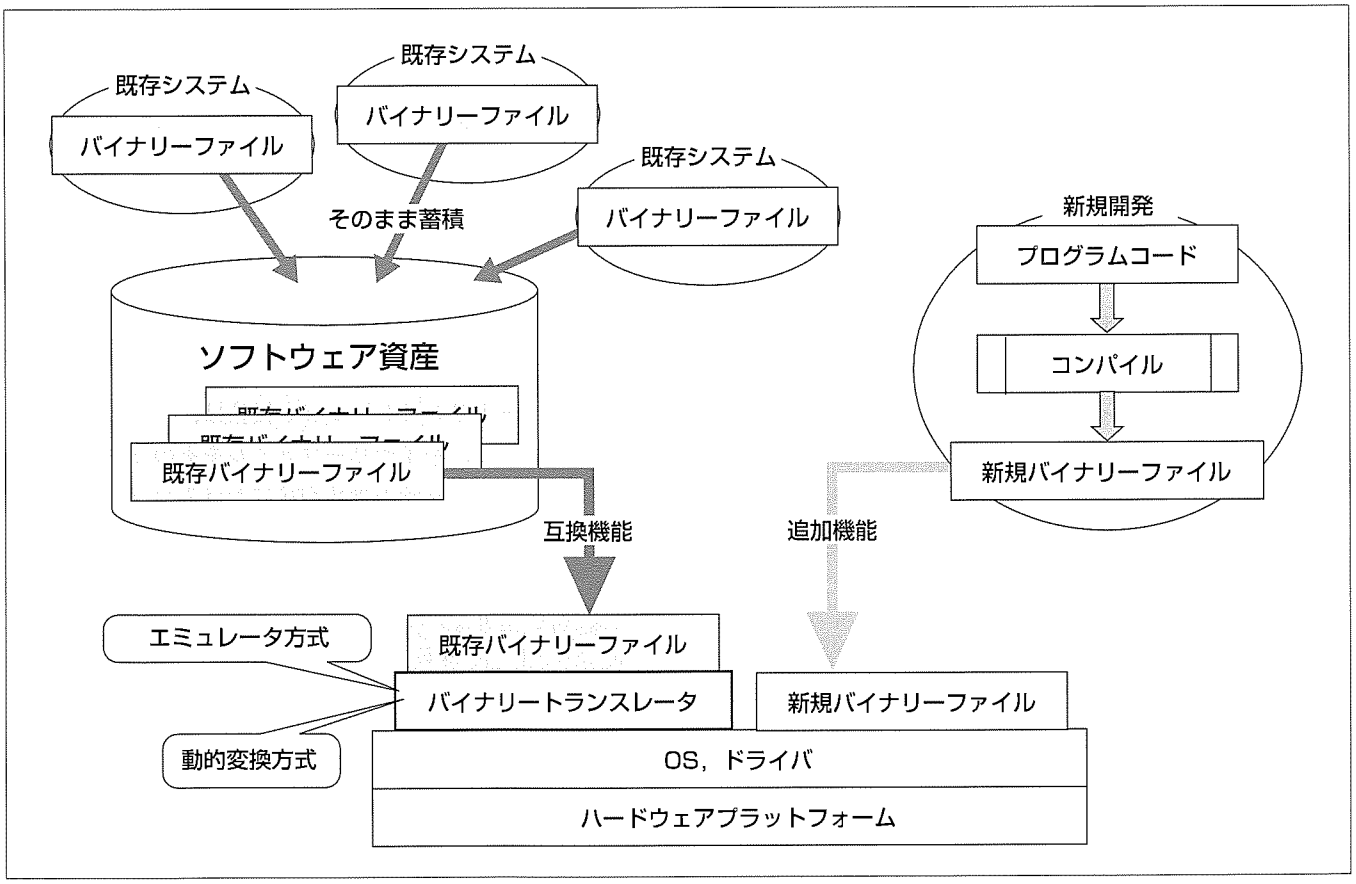
要　旨

近年，システムの高機能化に伴い，ソフトウェアの開発量は飛躍的に増大してきており，組込み機器においても，ソフトウェアを資産として蓄積し再利用していくことが必要となってきている。

既存ソフトウェアを新しいハードウェアにポーティングし資産として継承・活用するには，プラットフォームに互換機能を備えるか，プログラムコードから再コンパイルして新たなバイナリーファイルを生成・評価する必要がある。

バイナリートランスレーション技術は，ソフトウェア的にプラットフォームの互換機能を提供する手法である。バイナリートランスレーション技術には，①命令を逐次的に解釈・処理するエミュレータ方式，②動作中に命令列を一括翻訳・保存し保存した命令列を実行する動的変換方式，③オフラインで命令列を一括翻訳・保存し保存した命令列を実行する静的変換方式の3種類がある。ソフトウェア再利用に適しているエミュレータ方式と動的変換方式は，実行性能の高速化が課題である。エミュレータ方式では，複数の命令間で解釈と処理を並列動作させることで解決できる。動的変換方式の組込みシステムへの適用には，翻訳により生成するバイナリーファイルのコンパクト化が重要である。

今後は，組込みシステムのソフトウェア資産活用へのバイナリートランスレーション技術の適用に向けて，組込み用マイコンの特性に合わせたコンパクトで高速な処理方式の実現に向けた技術開発が必要である。



バイナリートランスレーションによるソフトウェア資産活用

既存システムで使用していたソフトウェアのバイナリーファイルをそのままの状態で資産として蓄積する。従来からの互換機能は，ソフトウェア資産として蓄積したバイナリーファイルをバイナリートランスレータで実行し，追加機能のみを新規開発する。

*情報技術総合研究所　**鎌倉製作所

1. まえがき

半導体技術の著しい進歩によるマイクロプロセッサの処理性能の大幅な向上により，従来ハードウェアで行ってきた処理をソフトウェアによって行うようになってきている。このため，組込みシステムにおいてもソフトウェアの開発量は増大の傾向にあり，既存ソフトウェアを資産として再利用する必要が出てきている。

バイナリートランスレーション技術は，アーキテクチャの異なるハードウェアプラットフォーム用に生成したバイナリーファイルを直接実行することにより，ソフトウェア資産の再利用を実現する技術である。

本稿では，バイナリートランスレーション技術の高速化技術と，組込みシステムに適用していく上での技術課題に関して述べる。

2. バイナリートランスレーション技術

2.1 分　類

バイナリートランスレーション技術は，エミュレータ方式，動的変換方式，静的変換方式の3種類[1]に分類できる。これらの特長を整理すると下記のようになる。

(1) エミュレータ方式

命令を逐次的に解釈・処理する。

(2) 動的変換方式

逐次的に解釈・処理した結果に基づき抽出した実行頻度の高い命令列を一括翻訳・保存し，保存した翻訳済み命令列を実行する。

(3) 静的変換方式

命令列を解析して一括翻訳・保存し，保存した翻訳済み命令列を実行する。翻訳はオフラインで行う。

静的変換方式はスタンドアロンのツールを用いて翻訳を行う必要があり，エンドユーザーに負担がかかる[1]。それに対して，エミュレータ方式と動的変換方式は，アーキテクチャの異なるハードウェアプラットフォームのバイナリーファイルを自動的に処理する方式であるため，ソフトウェアの再利用には適していると言える。

2.2 エミュレータ方式の高速化

エミュレータの処理は，主として，命令フェッチ，デコード，オペランド生成，演算，格納という複数の処理によって構成される。エミュレータでは，各処理結果同士の依存関係が強く，分岐・アドレス計算・データ読み出し待ちなどのオーバーヘッドにより，マイクロプロセッサの持つ並列性(スーパースケーラ機能やパイプライン処理)を活用できず性能が出ないのが一般的である。その課題を解決する手段として，処理間の因果関係の弱い複数命令のエミュレーション処理を並行して行いプロセッサの限界性能を引き出す方式[2]がある。

この方式の概念を図1に示す。エミュレータ処理において，命令フェッチ／デコードの処理とオペランド生成／演算／格納の処理に別の演算ユニットを割り付ける。命令Kのオペランド生成／演算／格納の処理を実行中に次の命令である命令$K+1$の命令フェッチ／デコードの処理を実行する。このような処理を連続して実行することにより，命令フェッチ／デコードの処理時間を隠蔽(いんぺい)し，全体の実行時間を短縮し，処理の高速化を実現する。

2.3 動的変換方式

2.1節に述べたとおり，動的変換方式は，実行時に命令列を一括変換して，それ以降は保存した変換コードを実行することで高速処理を実現する。

バイナリーファイルのすべてを変換すると，変換コードを保存するのに膨大なメモリスペースが必要となる。プログラムの処理には“処理の90%の時間は10%のコードで実行される”という経験則(90／10則)で言われる局所性がある[3]。この特性を利用しよく実行される一部のパス(高頻度パス)を変換・保存することで，高速性を維持しつつ，コンパクト化を実現できる。

図2に動的変換方式の概念を示す。バイナリーファイルは，まずエミュレータ方式のインタプリタで処理を行いながら，よく実行される一部のパス(高頻度パス)を抽出する。次に，抽出した高頻度パスを変換し，コードキャッシュと呼ばれるメモリ上の保存領域に変換コードを保存する。次回以降，変換済みのパスを実行するときは変換コードを実行する。実行回数が閾(しきい)値を超えない低頻度パスは変換されない。このようなメカニズムにより，高速かつコンパクトな実行環境が実現する。

3. 組込みシステムへの適用

3.1 バイナリートランスレーションの現状

バイナリートランスレーション技術は，コンピュータシステムにおいて，バイナリーファイルをそのまま使用することによりプロセッサの世代交代におけるソフトウェア移

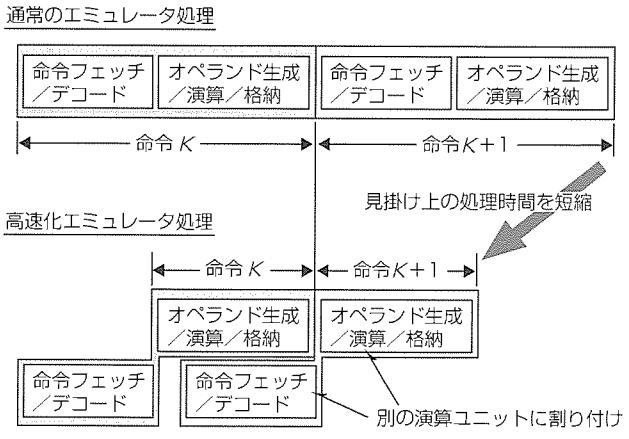


図1．エミュレータの高速化

行性を容易にするために培われてきた技術である。古くはメインフレームやワークステーションの機種切り換えに使用されてきた。最近では，メインフレームやオフィスコンピュータのプログラム資産をPCサーバ上で動作させるシステム，防衛用途組込みコンピュータ，パソコンの互換プロセッサチップに適用されている。

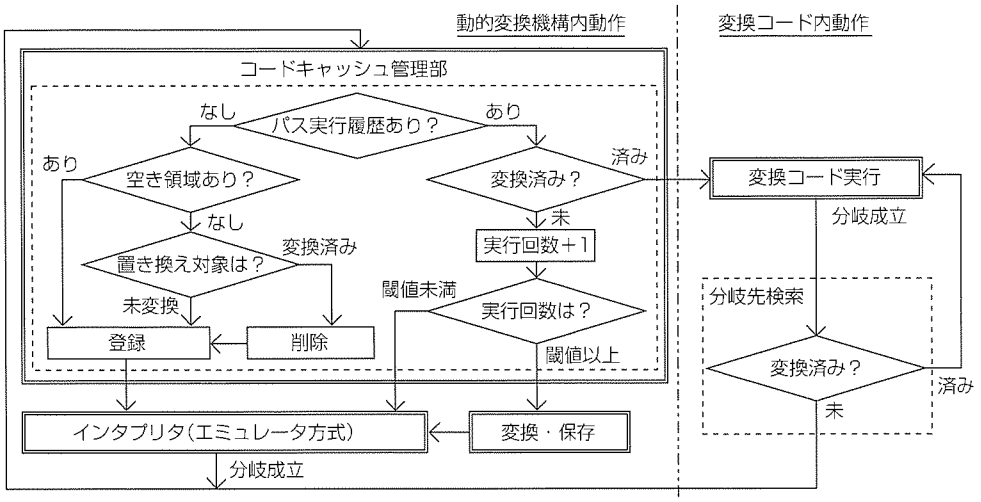


図2．動的変換方式

3.2 動的変換方式を適用する場合の課題

エミュレータ方式は，歴史もあり，高速化や小型化の方式が数多く提案されており，組込みシステム特有の課題はないと考えられる。一方，動的変換方式は，主として大容量のメインメモリ，キャッシュメモリを持つワークステーションやパソコンをターゲットに技術開発されてきており，組込みシステムへの適用に向けて解決すべき特有の技術課題が存在する。

(1) コードキャッシュのミス率

動的変換方式バイナリートランスレータの評価システムを構築し，コードキャッシュの容量とミス率の関係を測定した。**図3**はアプリケーションプログラムとしてSPECInt95の囲碁プログラム“go”(プログラムサイズ=250kバイト)を実行したときの測定結果である。図中のIPItは生成した命令列の変換率であり，1命令を変換したときに生成される命令数を示している。なお，IPIはInstructions per Instructionの略である。

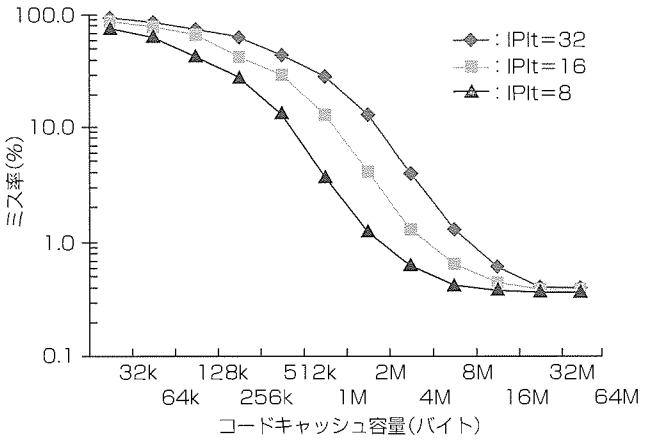


図3．コードキャッシュの容量とミス率

図4は，**図3**のミス率から性能に換算した結果である。図中のIPIxは実行する命令列の変換率であり，変換前の1命令を実行するために必要な命令数を示している。

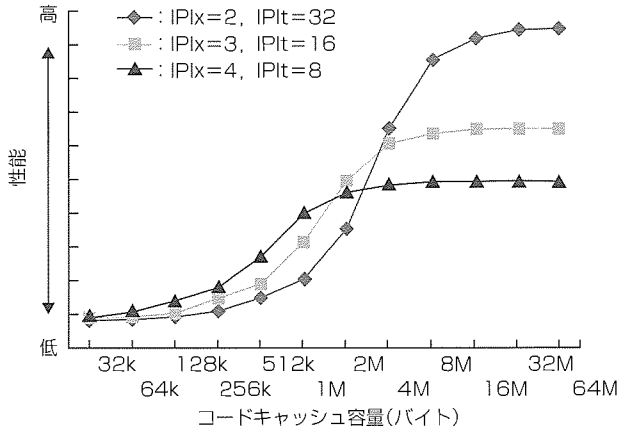


図4．コードキャッシュの容量と性能

コードキャッシュの容量が大きいときには，生成するコード量は気にせずに高速な命令を使用した速度優先の最適化が高い性能を得るが，逆に，コードキャッシュの容量が小さいときには，コード量優先の最適化が性能的にも効果あることが分かる。組込みシステム内において，コードキャッシュに割当て可能なメモリサイズから変換における最適化方針を導出する技術が必要である。

(2) キャッシュメモリのミス率

メモリシステムにはキャッシュメモリがあり，このミス率が性能に大きく影響する。**図5**は，コードキャッシュのヒット率が100%の状態において，高頻度パスのサイズを変化させて性能を測定した結果である。評価プラットフォームにはパソコンを使用した。一次キャッシュ又は二次キャッシュにヒットした状態ではエミュレータ方式よりも高い性能であるが，二次キャッシュをミスする状態ではエミュレータ方式よりも性能が低い。

プログラムサイズ，プログラムの動作特性，変換率などから高頻度パスのサイズを評価しエミュレータ方式か動的変換方式を選択する技術が必要である。

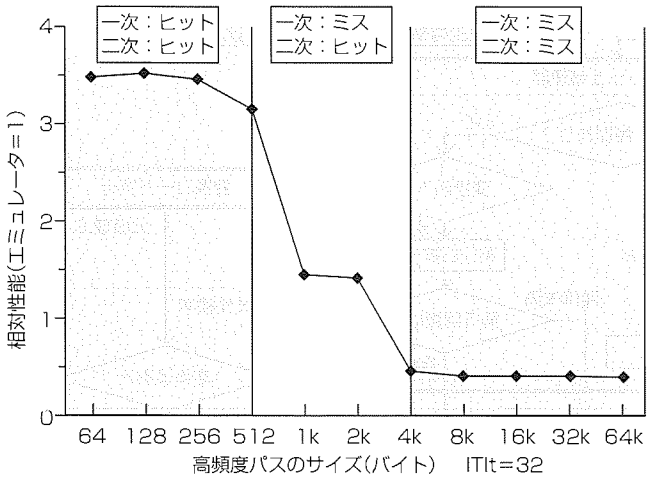


図5．キャッシュメモリの影響

4. 適 用 分 野

4.1 プロセッサ代替

マイクロプロセッサの開発には多大なコストがかかる。このため，半導体プロセスの進化に合わせて新規のマイクロプロセッサを開発することは，製品価格を高騰させ，市場競争力を低下させる場合も出てくる。一方，蓄積されているソフトウェア資産には，バイナリーファイルのレベルで互換性があり，そのまま使用できることが望ましい。

この問題を汎用マイクロプロセッサとバイナリートランスレーション技術により互換システム(図6)を構築することで解決できる。

形態(1)ではOSはポーティングし，アプリケーションはソフトウェア資産を使用したものである。ソフトウェアの開発量は多くなるが，高い性能が得られるというメリットがある。形態(2)はOSまで含めてすべてのプログラムについてソフトウェア資産を使用したものである。アドレス変換や例外検出をソフトウェアで実現するには，性能が低い欠点があるが，OS依存性がないためポーティングとシステム試験の費用面でメリットがある。

4.2 ソフトウェア開発・検証環境

パソコン用プロセッサは，組込み用マイコンに対して，周波数で単純に比較しても10倍以上の高い処理性能を持っている。この性能差を利用し，バイナリートランスレーション技術によりパソコン上に実システムと同程度の処理速度を持つソフトウェア開発・検証環境(図7)を構築できる。

ハードウェア開発の工期に依存しないので，システム試験の早期着手が可能になる。また，解析機能やマルチプロセッサ構成のように，従来のソフトウェア環境では作りにくかった検証環境を構築できる。正確で定量的なシステムの動作情報を基に不具合現象や性能ボトルネックなどを解析することで，処理性能や信頼性といった製品目標を早期に確認することができる。

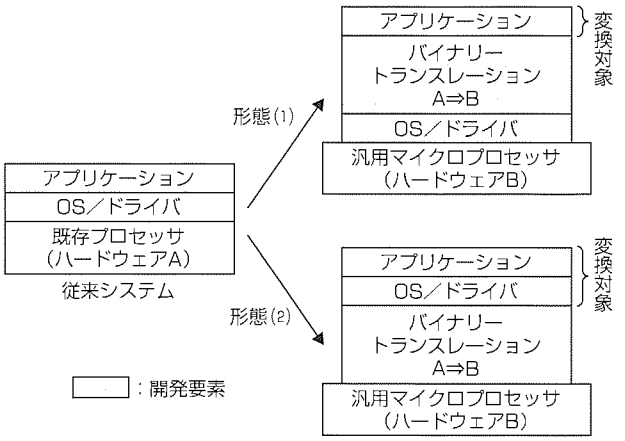


図6．互換システム

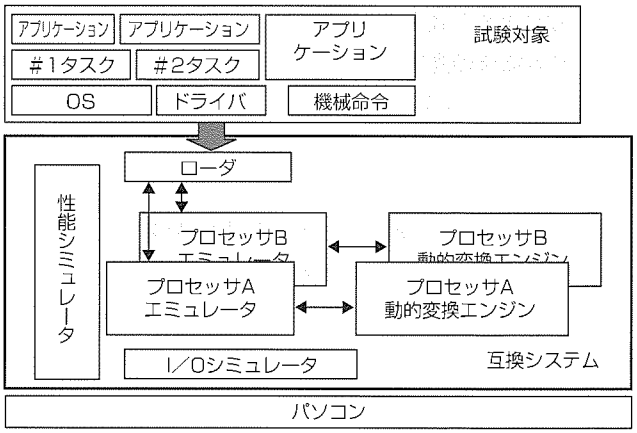


図7．ソフトウェア開発・検証環境

5. む す び

組込みシステムにおけるソフトウェアの開発量はますます増加する傾向にある。そのため，ソフトウェアの再利用による生産性向上の要求がより一層高まってくる。

バイナリートランスレーション技術は，バイナリーファイルをソフトウェア資産として再利用できる有効な技術である。組込みシステムへ適用するために，マイコンの特性に合わせたコンパクトで高速な処理方式の実現に向けた技術開発が必要である。

参 考 文 献

(1) Erik, R., et al.：Welcome to the Opportunities of Binary Translation., IEEE Computer, **33**, 40～45 (2000)

(2) 近江谷康人：命令エミュレーション方法，公開特許広報，特開2002-182928 (2002)

(3) Hennessy, J. L., et al.：Computer Architecture：A Quantitative Approach, Morgan Kaufmann Publishers, Inc. (1996)

生産ライン自動運用装置　特許第3169079号(特開平3-287360)

発明者　森　一之，築山　誠，福田豊生

この発明は，生産計画における工程の処理予定時刻に基づいて生産ラインを自動運用する装置に関するものである。

従来の生産ライン運用装置は，装置の遊び時間をなくすことを目的としている。したがって，生産計画で予定された納期である製品完成時刻に製品を完成させるとは限らなかった。この発明は，上記の問題点を解決するためになされたもので，生産計画で予定された製品完成時刻に製品を完成することができる生産ライン自動運用装置を得ることを目的とする。

この装置は，まず，生産ラインの主な工程における材料各々の処理開始時刻と処理完了時刻が記述された生産スケジュールを生産ライン上の材料各々の仕掛り状態に変換し，材料各々についてこの仕掛り状態と所定時間における生産ライン上の材料各々の仕掛り状態を比較することにより，材料各々の評価値を演算する。そして，この評価値に基づいて生産ライン上の設備の運用方法を決定し，この運用方法に従って生産ライン上の設備を運用する。

したがって，この発明にかかわる生産ライン自動運用装置は，材料各々の各工程において，生産計画からのずれの状態を大局的に判断し，生産ラインを運用する制御を行うので，生産計画で予定された製品完成時刻に製品を完成させることができる効果がある。また，各設備の運用の決定は，各設備の材料の仕掛り情報だけを使用する。したがって，この装置を各設備に配置することにより，生産ラインを早く自律的にかつ各設備単位で分散して行うことができるので，短時間で運用制御できる効果がある。

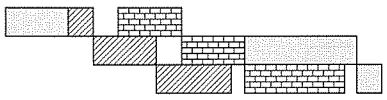


図1．計画スケジュール

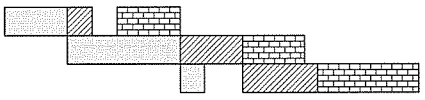


図2．従来手法(先入れ先出し)によるライン運用結果

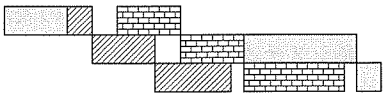


図3．この発明によるライン運用結果

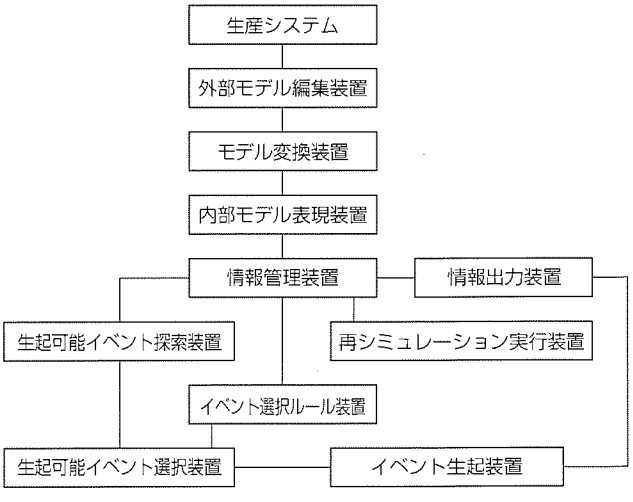
生産計画シミュレーション装置　特許第2608085号(特開平1-199756)

発明者　森　一之，築山　誠，福田豊生

この発明は，生産システムの操業スケジュールを離散事象シミュレーションによって作成するスケジューラに関するものである。

従来のスケジューラは，シミュレーションを実行させるためには生産システムのモデルをユーザーがプログラミング言語で記述する必要があった。また，シミュレーション実行後，シミュレーションの条件を一部変更してスケジュールを変更したいときには最初からシミュレーションを再実行する必要があった。

この発明は，上記の問題点を解決するためになされたものである。まず，ユーザーはグラフィックエディターによる生産システムの外部モデル編集装置により生産システムモデルの外部モデルを作成する。この生産システムの外部モデルを計算機のシミュレーションモデルである内部モデルへ自動的に変換する。これにより，ユーザーはプログラミングすることなしにシミュレーションのモデルを容易に作成できる効果がある。この発明の装置は，内部モデルとライン運用ルールであるイベント選択ルールに基づいて離散事象シミュレーションを行い，その結果を生産スケジュールとして出力する。さらに，この発明の装置は，情報管理装置が各イベントの生起時刻と完了時刻を記憶しているので，得られたシミュレーション結果の任意の時刻から条件を変えて再シミュレーションができる効果がある。



生産計画作成装置　特許第2885406号(特開平2-95549)

発明者　森　一之，築山　誠，福田豊生

この発明は，この装置とユーザーとの共同作業により生産計画を作成するための生産計画作成装置に関するものである。

従来の離散事象システムシミュレータを用いた生産計画装置は，生産計画の変更／追加が難しく，生産計画を初めから立て直さなければならないという問題があった。また，作成された生産計画が必ずユーザーの要求(納期を守る計画など)に合うとは限らないという問題があった。

この発明は，上記の問題点を解決するためになされたものである。この装置に対して，ユーザーの指示，生産システムの情報又は稼働中のシステムからの情報を入力するための情報入力手段と，この情報入力手段の出力に基づいて生産システムのモデルと生産計画及びその制約を所定の形式で表現する表現手段と，この表現手段により表現された生産システムのモデルに基づいて生産計画をシミュレートするシミュレーション手段と，作成した生産計画を生産における制約を満足しつつ生産計画を編集(ロットの移動，コピー，削除)する手段，並びにこの生産計画に生産計画における機会の空き時間及びプロセスの空き時間の前向き探索（図1）又は後ろ向き探索(図2)を行い，新たな生産計画を追加する生産計画する手段を備える。

このような構成により，生産計画の作成中又は作成後の生産計画の編集(ロットの移動，コピー，削除)，追加を簡単に行うことができ，その結果，特急生産の納期が確定した場合のロットの投入時期，さらにロットを投入した場合の製品完成時期の明確化を図ることができる効果がある。

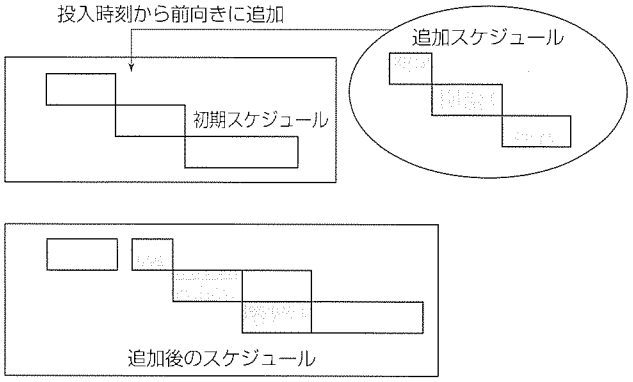


図1

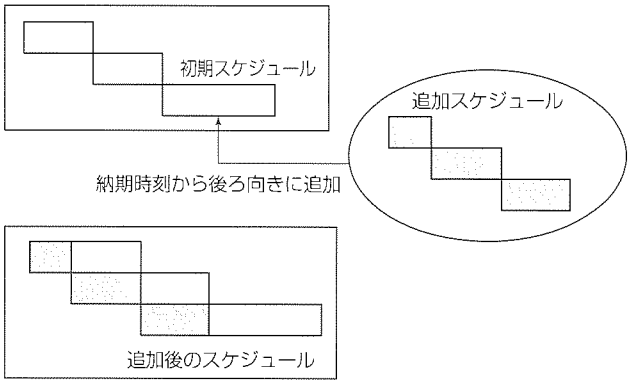


図2

〈本号記載の商標について〉
本号に記載されている会社名，製品名はそれぞれの会社の商標又は登録商標である。

〈次号予定〉三菱電機技報 Vol.77 No.8 特集「IT宇宙インフラ」

三菱電機技報編集委員
委員長　井手　清
委員　小林智里　長谷川裕　堤清英
桑原幸志　村松洋　松本修
浜敬三　田島範一　中川博雅
中島克人　部谷文伸
黒畑幸雄　山木比呂志
事務局　松本敬之
本号取りまとめ委員　坂井英明
堀池聡

三菱電機技報　77巻7号　2003年7月22日　印刷
(無断転載・複製を禁ず)　2003年7月25日　発行
編集人　井手　清
発行人　松本敬之
発行所　三菱電機エンジニアリング株式会社　e-ソリューション&サービス事業部
〒105-0011　東京都港区芝公園二丁目4番1号
秀和芝パークビルA館9階　電話(03)3437局2692
印刷所　株式会社　三菱電機ドキュメンテクス
発売元　株式会社　オーム社
〒101-0054　東京都千代田区神田錦町三丁目1番地
電話(03)3233局0641
定価　1部735円(本体700円)　送料別

URL　http://www.MitsubishiElectric.co.jp/giho/
三菱電機技報に関するお問い合わせ先　cep.giho@ml.hq.melco.co.jp

スポットライト

汎用熱・流体解析ソフトウェア “MelTHERFY” “TherfBENCH”

MelTHERFY(メルサーフィ)(注1)は，三菱電機 先端技術総合研究所で開発された歴史ある汎用熱・流体解析ソフトウェアです。

三菱電機 設計技術センターにおいて開発された専用のプリポストプロセッサであるTherfBENCH(サーフベンチ)(注1)の登場でモデル作成が容易になり，解析結果を回路網上に直接表示(アニメーション表示も可能)できるようになりました。

■製品の特長

回路網法は，有限要素法などの領域型解法に比べて格段に計算時間が早いという特長を持っていることから，熱設計者にとって，限られた時間の中で多くのパラメータ解析を行うことができます。

MelTHERFYは既に三菱電機情報ネットワーク㈱(MIND社)からパッケージソフトとして外販していますが，TherfBENCHもMIND社から販売することになりました。両ソフトウェアともWindows(注2)対応で一般に普及しているパソコンで十分動作可能ですので，導入も容易です。

(注1)MelTHERFY，TherfBENCHは，三菱電機㈱の登録商標です。
(注2)Windowsは，Microsoft Corp.の登録商標です。

■計算例

ここでは，水道管内に流れるお湯の過渡熱解析の例(図1)を示します。

図2は，発熱源が関数変化する場合のあるノードの温度変化をグラフ表示させた計算例です。このような解析も計算時間は1分程度です。

TherfBENCHは部品エディターとネットワークエディターで構成され，三次元解析も，ネットワーク接続機能を使うことで可能です。

■今後の取り組み

詳細な熱分布をCFD(Computational Fluid Dynamics)で解析するには数時間～数十時間を要する場合が多く，解析できる回数には制限があります。CFDによる詳細計算結果から支配的熱経路の特定を行い，その結果からコンパクトモデルとして熱回路を作成し，回路網法で設計パラメータの解析を実施する方法が注目されてきています(三菱電機 設計技術センターにおけるテンプレートシステムアプローチ)。熱設計者にとってMelTHERFYは，短時間に精度良い解析ができる大変便利なツールであると言えます。

MIND社では，お客様からのニーズを受けて，パッケージソフトの販売だけでなく様々なカスタマイズやシステム解析及び解析支援をさせていただきます。

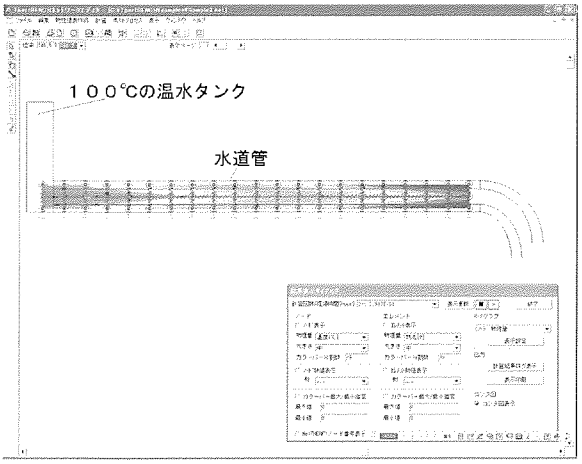


図1．計算例1：水道管内のお湯の流れ解析

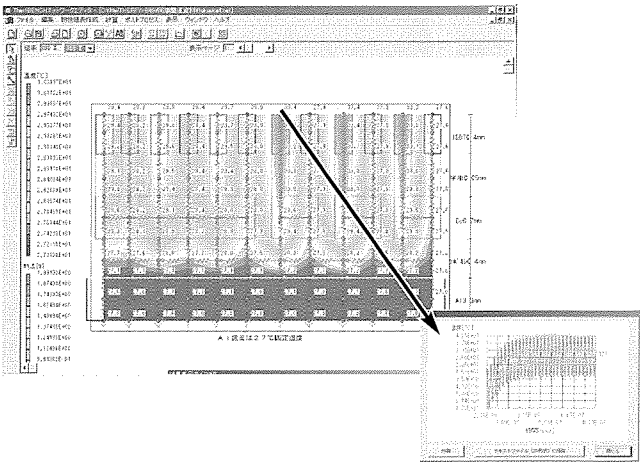


図2．計算例2：過渡熱解析

住　所：〒661-0001 尼崎市塚口本町8丁目1番1号
会社名：三菱電機情報ネットワーク㈱　お問い合わせ先：西日本アプリケーション本部IT応用技術課
TEL 06-6494-0371　FAX 06-6497-9627
URL http://www.mind.co.jp/service/index.html