ホストCOBOLシステムのパソコン活用の実現

（確信の時，積年の夢が現実へと変わった）

富士火災海上保険（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

林　　達　也

1982年　富士火災海上保険（株）入社
　　自動車保険システム担当
1991年　データベース化プロジェクト担当
1994年　ホスト運用担当
1996年　損害調査システム担当
2001年　現在 システム部営業支援システム
　　所属　営業支援端末システム担当

■ 論文要旨 ■

損害保険の自由化という規制緩和に伴って，外資系保険会社・生保系損保子会社などが続々と参入する中，激化する競争の中で戦っていくために特色のある保険商品を早期に市場投入するニーズが高まっている．こんな中システム部門では，新しい保険商品のシステム開発や顧客ニーズに応えた保険商品へのシステム変更が次から次へと対応を迫られる戦国時代に入ってきている．特に，保険料計算システムについては，過去の開発経緯と開発環境・開発言語の違いからホスト，パソコン，ＷＥＢの３重持ちの状況を招き，改定の都度，開発工数がかかり開発効率の面からも非効率な状況が続いていた．よって今回，開発言語が全く違うこれら３つのシステムをホストのＣＯＢＯＬ言語を使用して１つのソースにて稼動させることにし，大幅な開発効率と開発費用の削減を実現した．同時に，従来からのホスト系ＣＯＢＯＬシステム担当者の業務知識と高い技術力をオープン系システム開発の世界でも活かせることとなった．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　当社概要

当社は 1918 年に設立された「日本簡易火災保険株式会社」がその前身であり，以来 84 年にわたり中小企業・個人を顧客基盤として保険商品・サービスを提供している損害保険会社である．損害保険商品ラインアップはほとんどすべての分野をカバーしているが，中でも自動車保険は収入保険料の約６割を占める主力商品である．

システム部門では，ほとんどの保険商品事務処理をシステム化しており，保険申込書を処理する商品系のホストシステムを中心に，最近ではオープン系システムも手がけ，顧客や損害保険代理店との情報基盤となるＷｅｂ系システムも稼動させている．また，代理店向けには保険販売を助けるパソコン用システムの開発提供も行っている．開発体制はホスト系システムは社内要員による開発を行っており，オープン系システムについては外部業者に開発を委託している割合が高い．

１．２　近年の業界動向

損害保険の自由化という規制緩和に伴って，特に自動車保険の分野では外資系保険会社をはじめとし生保系損害保険子会社，あるいは異業種などが続々と参入している．保険商品についても各社各様の特色あるものを投入してきており競争は激化の一途をたどっている．当社においても他社との差別化を図るために新しい保険商品の開発や既存保険商品の改定を繰り返している．競争に勝ちぬくためには，顧客ニーズに応えた商品開発は勿論のこと商品販売の基盤を支えるシステム開発につき，そのスピードや外注コストが大きな課題となっている．

２．保険料計算システムの現状と課題

２．１　保険料計算システムとは

保険料計算システムとは，保険会社の商品の価格を算出するシステムである．自動車保険では，自動車の種類や運転者の年齢など約 100 種類にもおよぶ項目により保険料が決定する．その算出結果に至っては無限ともいえるパターンが存在する．保険料計算の前提となる項目間の整合性チェックや項目自身の正当性チェック機能についても保険料計算システムの機能の一部である．

顧客ニーズに応える自動車保険は，顧客の生活スタイルや自動車の使用方法や自動車に付属する様々な装置の違いにより多様化しており，顧客により保険料計算の方法は異なり，今や人の手で計算ができる代物ではなくなっている．言い換えれば，保険料計算システムがないと自動車保険の販売は不可能な領域にまで来てしまっているのが現状である．

２．２　保険料計算システムの現状

保険申込書に記入された保険内容が正しい内容なのか，またその内容に見合った保険料をお客様より頂いているかといった保険申込書計上チェックのためのホストシステムは当社のシステム部門創生期より存在し，人の誤りをシステムが発見する必要性の中から保険料計算システムは保険申込書計上チェックの一機能として生まれてきた．今では，保険締

結前に顧客に対して保険内容を説明する際の保険料の試算を行う時にも必要なシステムとなっている．

もともと保険料計算システムは，保険申込書の計上時のホストでのチェックシステムに組み込まれていたもので，バッチシステムとして存在していた．現在では，ホストオンライン処理の中で同様に保険申込書の計上時チェックの一機能として稼動している．システムの開発言語はＣＯＢＯＬで，現在でもホストバッチシステム・ホストオンラインシステムで稼動している．

近年になってパソコンが普及すると，営業社員や保険代理店から対面販売時に保険料を計算するツールの要望を受けパソコンで保険料を計算できる保険料計算システムが登場した．このシステムは，パソコン用の開発言語であるＶｉｓｕａｌＢａｓｉｃ（以下ＶＢ）やＶｉｓｕａｌＣ＋＋（以下ＶＣ＋＋）などで開発されており，計算結果はホストの保険料計算システムと同じなのだが，実態は全く違うシステムとして代理店のパソコンに導入している．

最近では，インターネットを使った保険販売や保険料試算が業界各社においてもラインアップされ，当社においてもホームページでの保険料試算が始まった．Ｗｅｂシステムで稼動する保険料計算システムもＶＣ＋＋で開発されているがパソコンのそれとは違い，あくまでもＷｅｂサーバで稼動する保険料計算システムで，やはり実態の違うものであった．

このような背景の中，当社には３つの保険料計算システムが存在することとなった．１つはホストのＣＯＢＯＬ資産，１つはパソコンのＶＢ・ＶＣ＋＋資産，１つはＷｅｂサーバのＶＣ＋＋資産である．パソコンのＶＢ・ＶＣ＋＋資産については，社内外の約 13,000 台のパソコンに搭載されている．

２．３　保険料計算システムの課題

保険会社の商品価格を取り扱う保険料計算システムは，保険システムの心臓部と言っても過言ではない．また，保険システムの膨大な機能の中で最も利用者が多いサブシステムである．言い換えれば保険料計算システムは，「まちがい」が許されないシステムである．従ってこのシステムの開発において特徴的なのは，テスト工数を通常のシステム開発に比して多く設定する必要があるということである．

保険料計算システムは，保険商品の発売時には利用者全員の手元に準備されていなければならない．保険商品の多様化に伴い，手計算では保険料を導けない状況であり，このシステムがなければ保険商品を販売できないからである．よって当社が保有する３つの保険料計算システム，ホスト，Ｗｅｂ，パソコンは同時期にリリースしなければならない．

特に，パソコンに搭載している保険料計算システムは，パソコンのハードディスク上に配置し，スタンドアローンのパソコンにおいても利用できるシステムであり，このシステムの変更を行う場合は，差し替え用のソフトウェアを配布する必要がある．パソコンでの利用者は，主に保険代理店であるが未だネットワーク接続をしていないパソコンも多数存在するため，約 13,000 枚に上るＣＤ－ＲＯＭを作成し配布するという方法をとっている．

大量のＣＤ－ＲＯＭをプレスし配布するためには，約２週間の期間が必要で，パソコンの保険料計算システムは，ホスト・Ｗｅｂと同時に開発をスタートするものの，発売の２週間前にはシステム開発の完了を迎える必要があり，必然的に開発期間は短くなっている．

最も歴史の古いホストの保険料計算システムは，社内での内製を実現している．一方，

Ｗｅｂ・パソコンの保険料計算システムは，歴史が浅くオープン系システムの技術を必要とするため内製ではなく，この方面の技術力を持った外部業者に開発を委託している．

外部業者に委託していることによる問題点は３つある．ひとつは，内製するよりもコストが大きくかかるということ．ひとつは，少しプログラムに手を入れたいと思っても，今日明日というわけにいかず，それなりの開発期間を要してしまうこと．最後に，外部業者は，保険のプロフェッショナルではないため，プログラムの品質を高い水準で保つことに非常な労力を要することである．具体的には，仕様のやり取りが頻発したり当方のユーザーテストに時間と労力を必要以上に掛けてしまっているということである．

すなわち以上のことをまとめると，保険料計算システムの課題は，絶対に「まちがい」を起こさないためにテストに大きな工数を必要とすること，同じシステムを開発言語を変えて３つ作成しなければならないこと，パソコンシステムに許される開発期間が総じて短いこと，外部業者に頼っていることによりコストがかかっていること，さらに思惑通りに開発が進まないこと，である．

３．ホスト資産と人的資源の有効活用

３．１　システム資産統合への模索

保険料計算システムの課題を解決するために，保険料計算システムをなんとか一つのプログラムソースで，ホスト・Ｗｅｂ・パソコン上で稼動させる方法はないかと模索していた訳だが，プログラムソースの統一は開発言語の統一を意味するので，ホスト・Ｗｅｂ・パソコンの全てのシステムで稼動する開発言語を探すことが必須であった．

また一方では，ホスト系システム一辺倒からオープン系へとＩＴ化が進むシステム開発の世界において，社内の技術者はＣＯＢＯＬでのホスト開発に日々追われ，新しい技術や開発言語を習得する余裕もなく，やむを得ずオープン系のシステムは外部業者へ委託せざるを得ない状況が続いていた．社内では保険業務に精通し高い技術を持つ社内の開発要員をオープン系システムの世界で活かす方法はないか，という課題が持ち上がっていたのも事実である．

３．２　ホスト資産の有効活用

社内技術者をオープン系システムの世界で活かすためには，新しい技術や開発言語を習得させる方法があるが，育成のためには一定のコストと時間を要する．また，ホスト開発要員である社内技術者をオープン系システムの開発要員にシフトすると，この間，補完要員としてホスト開発要員を補充する必要も生じてくる．これでは外部業者委託の問題はクリアできるが，保険料計算システムの統合はできない．保険料計算システムの統合つまり開発言語の統一を実現し社内技術者をオープン系システムで活かすためには，ホストで稼動するＣＯＢＯＬ資産がそのままオープン系で稼動するというコンセプトしか考えられなかった．

ほんとうに，そんなことが現実的に可能なのであろうか？ベンダーへのヒアリング，インターネットを使用しての調査を行った．浮かび上がってきたのは，２つの製品であった．いづれの製品もＣＯＢＯＬプログラムソースをパソコン上でコンパイルすることにより，ホストで稼動していたＣＯＢＯＬ資産がパソコン上で稼動するというもので，当社の目指すコンセプトに合致したものであった．しかし，信じる気にはなれなかった．

３．３　人的資源の有効活用

社内技術者が従来通りＣＯＢＯＬのプログラム開発を行うことで，ソフトウェアがホストで稼動するだけでなくＷｅｂでもパソコンでも稼動するということならば，社内技術者は，ホストシステムやオープン系システムを意識することなく同一のプログラムソースで開発が可能となる．社内技術者は，新たに開発言語を習得する必要もなく，これまでの高度な保険知識を有したままでオープン系システムを手がけることができるわけである．さらに，外部業者への委託コストの問題，開発効率の問題をも一気に解決することとなる．

４．システム概要と開発過程

４．１　ＣＯＢＯＬ言語支援製品の選定

ＣＯＢＯＬソースをオープン系の世界でコンパイルし，Ｗｅｂやパソコンで実行可能な実行ファイルを生成できるＣＯＢＯＬ言語支援製品は，前述の２製品であった．当社としての新しい取組みであり，将来に渡っての使用が予想されるため，社内標準化の意味合いからも１製品に絞り込みを行うこととした．

今回の製品選定においては，２つのＣＯＢＯＬ支援製品で生成したソフトウェアが利用者側パソコン上で問題なく稼動することを基本的な選定基準におき，応用面では処理性能，リソース消費量，システム開発時における製品自体の使い勝手を検証した．

具体的な選定方法は，プロトタイプシステムを構築することにより２製品の双方で，実行ファイルを生成し，この実行ファイルを呼び出す親プログラムをＣ言語で作成しレスポンスタイムやリソース消費量などをレポートした．プロトタイプシステムは，保険料計算システムの計算ロジック部分（以下　計算エンジン）を抜き出して作成した．

結果としては，実行レスポンスに約４倍の差が出たのをはじめ，実行ファイルのサイズや稼動保証パソコンＯＳの種類など，**表１**で示すように，ほとんどの比較項目で一方の製品（製品Ａ）が他方を上回った．

また，この製品は製品自身のユーザーインターフェイスについても，コンパイルの手順が簡易で理解しやすかったり，ソースエディターも色別に修正箇所の判別が容易であるといった優位点があり，迷うことなく，製品を絞り込むことができた．ほんとうに動くのだろうかという不安と期待が確信へと変わった．

表１　ＣＯＢＯＬ支援製品比較表

	製品Ａ	製品Ｂ
ＣＰＵ使用時間（／件）	１００ｍｓ	４７０ｍｓ
メモリ使用量	３Ｍｂ	３Ｍｂ
実行ファイル容量	０．９Ｍｂ	１．３８Ｍｂ
動作可能ＯＳ		
Ｗｉｎｄｏｗｓ９５	○	○
Ｗｉｎｄｏｗｓ９８	○	○
ＷｉｎｄｏｗｓＭｅ	○	×
ＷｉｎｄｏｗｓＮＴ３．５１	×	○
ＷｉｎｄｏｗｓＮＴ４．０	○	○
Ｗｉｎｄｏｗｓ２０００	○	×
ＷｉｎｄｏｗｓＸＰ	○	×
ＷｅｂでのＡｃｔｉｖｅＸ化	可能	可能

４．２　プログラムソース取込運用

４．２．１　運用管理システムの必要性

当社システム部門では，オープン系システム担当部署とホスト系開発担当部署が組織的に分かれており，保険料計算システムにプログラム修正が発生した場合，ホスト系開発担当部署で修正完了したＣＯＢＯＬソースは間違いなくオープン系システム担当部署に取り込み，同一のプログラムソース内容を保証しなければならない．なぜなら，ホスト・Ｗｅｂ・パソコンの３つのシステムが同じ計算結果を返さないとなると，社内外の保険料計算事務を混乱させることになるからである．

パソコン用ソフトウェアは，ＣＤによる配布を行っているため，リリース日以前の２週間以上前にＣＤにソフトウェアを焼き込む．一方，ホストやＷｅｂのプログラムはリリース前日にプログラムの入れ替えを行う．つまり社内外でのリリース日はホスト・Ｗｅｂ・パソコンで同時期であるがプログラムソースの確定時期は，パソコンのプログラムの方が２週間以上早い時期となる．すなわち，ホストでの開発が完了しパソコン側に取り込んだ後，リリース日までの期間，ホストの保険料計算システムは，まだ修正を行うことが可能であるが，同一のプログラムソース内容を保証するために，ホストにおけるシステム修正は完全に抑止する必要がある．

また，ホスト系開発担当に一元化している対象プログラムを日常より意識させることには無理があり，同様にオープン系開発担当もホスト系でのプログラムメンテナンスを認識することにも無理がある。しかし，ホスト系オープン系の両者が認識を持つことによって，リリース時期のずれや取り込み漏れを防ぐことができる．すなわち，一元化の対象プログラムは，何らかの方法でホスト系オープン系両者に認識をもたせてメンテナンスやリリースが行われる仕組みが必要である．

ホストにおける開発要員は，数十人にもおよび今回の保険料計算システムはプログラム本数にして 330 本以上に上るため，担当部署や開発要員同士のコミュニケーションでプログラムソースの同一性を保証することは現実的でないため，何らかの運用管理システムによる機械管理が必要と判断した．

４．２．２　運用管理システムの構築

ＣＯＢＯＬソースの運用管理システムを構築する目的は，リリースするソフトウェアのソース内容を同一に保つことにある．従来より，ホストシステムではＣＯＢＯＬプログラムをプログラム管理システムにて運用を行っている．特にソース管理に関しては，リリース済みの本番資産と開発中のテスト資産に分かれ，開発者は，テスト資産環境にてシステム開発を行っている．リリース時には，本番資産環境へプログラムを移行することによりリリースを迎え，一つの開発が終了する．プログラム修正を始める場合は，本番資産環境からテスト資産環境へプログラムを移行し，プログラム修正を開始する．移行に際しては，厳重に管理された開発案件管理ナンバーを不正防止の鍵として運用している．

パソコンやＷｅｂで稼動するＣＯＢＯＬプログラムを作成するためには，パソコン上でＣＯＢＯＬソースをコンパイルし実行形式ファイルを生成する必要がある．これを行うのが，前述のＣＯＢＯＬ支援製品であるが，これらの作業は，この製品を使用して複数のプログラムを一度にバッチ形式で処理することができ，作業としても大した作業ではない．

問題は，いかにホストのプログラム管理システムからパソコンの世界に同一のプログラ

ムソースを移行するかである．

ホストのプログラム管理システムからのプログラムソース取込に関しては，取込対象プログラムが現在ホストで修正中なのか，パソコン・Ｗｅｂ側へ取り込みはいつ行うのか，いつ行ったのか，などを管理するための取込資源管理システムが必要であると判断した．これは，ホストでのプログラム修正が始まったことや，取り込みが行われたことによるホスト側プログラムソースの修正抑止，さらに取り込み漏れや２重取込などの危険性を回避するためのシステムである．

さらに，最新のプログラムソースを取り込むために，常にコンパイル結果ソースを自動的に保存する機能も必要であった．

このシステムの機能は，

① 取込対象プログラムに対して，本番資産環境からテスト資産環境へのプログラム移行を常に抑止する機能．

② 本番資産環境からテスト資産環境にプログラム移行する際，パソコン・Ｗｅｂ担当者が取込資源管理システムに対して移行抑止を解除する機能．

③ 取込対象プログラムの状態を常に監視する機能

④ コンパイル結果ソースを自動保存する機能

の４点である．

このシステムは，ホストのプログラム管理システムにサブシステムとして組み込む形で構築を行った．ホストのプログラム管理システムは，ＣＬＩＳＴで構成されたＰＦＤメニューでできており，今回の各種機能は，これらに対してシステム修正を行うことにより実現している．

４．２．３　アセンブラプログラムの取扱

ホストの保険料計算システムのプログラムは，ほとんどがＣＯＢＯＬであるが，一部アセンブラプログラムも存在する．アセンブラプログラムは，そのままではパソコン・Ｗｅｂ用に実行形式ファイルにすることは不可能なので，やむを得ずＣＯＢＯＬへの再コーディングを行った．プログラム本数としては８本であった．

また，当社では項目コード値に対応する項目の日本語表現や保険料率数字など，システム内でコンスタントで使用する文字列や数字とそれに対応するコードについて，定型のアセンブラプログラムにてコード対応テーブルを作成しシステム内で共有化する手法をとっており，こちらもアセンブラプログラムとなっていた．このプログラムについては，定型のアセンブラソース（以下　アセンブラテーブル）となっており，ソースコンバージョンの仕組みを構築することにより，ＣＯＢＯＬソース化を実現した．

ホストの取込資源管理システムからプログラムソースを取り込んだ後，アセンブラテーブルについては，常にソースコンバートを行うことによりＣＯＢＯＬソース化しプログラムソースの同一性を保証した．

４．２．４　プログラム取込システム全体像

富士通ホストのホスト取込資源管理システムからＣＯＢＯＬソース管理サーバへの取込ルートは，富士通社製ソフトウエアのＤＵＥＴを使った．ＤＵＥＴは富士通社のワークステーションマネージャ（ＷＳＭＧＲ）上で起動しホストの順編成ＰＳファイルをパソコン

で扱えるＴＥＸＴ形式ファイルとしてホストよりデータ取込を行うソフトウエアで，今回は，ホスト取込資源管理システムからＤＵＥＴを起動することにより自動取込を実現した．

ホスト開発は，従来よりＴＳＳ端末を使用していたが，最近では端末がパソコンへと移行し，ホストファイル資産をパソコンのハードディスクに簡単に落とし込むことが可能となっており，これらの機能を持つワークステーションマネージャ（ＷＳＭＧＲ）やＤＵＥＴは，ホスト系とオープン系を連携する基盤ソフトウエアであり，今回のプログラム取込システムの情報連携基盤となっている．

プログラム取込システムの全体像を**図１**に示す．

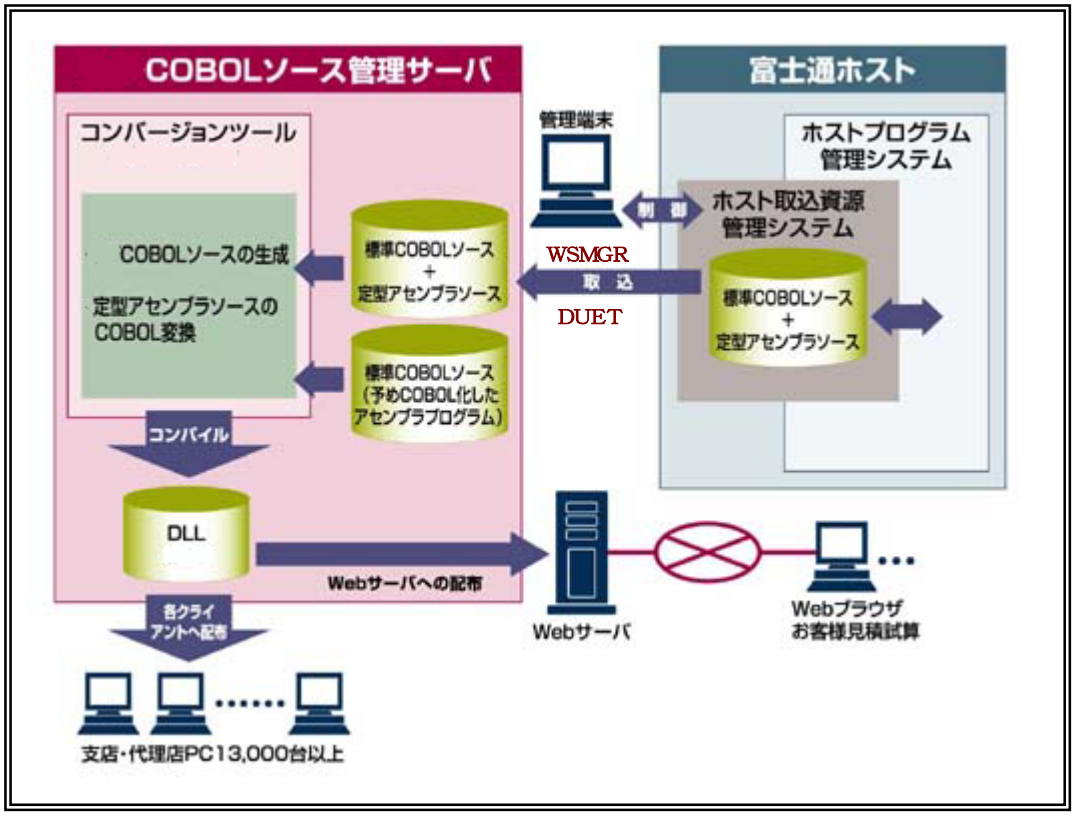


図１　プログラム取込システムの全体像

４．３　パソコンシステムの改造

パソコンで稼動している保険料計算システムは，画面，項目チェック，計算エンジンの３つの構成要素により構築されている．今回，ホストの項目チェックと計算エンジンを組み込むにあたり，まず，項目チェックおよび計算エンジンロジックを画面ロジックから切り離しを行い，インターフェイスエリアを介してのやり取りで動作できるようにしホスト計算エンジンおよび項目チェックロジックの置き換えを実施した．画面ロジックについてはＣＯＢＯＬ化できないため，そのままの形で継続利用することとした．パソコンのプログラム構造を**図２**に示す．

画面より入力されてくる項目群をホストプログラムのインターフェイスに合わせるためには，項目コードをパソコンシステム用のそれからホストシステム用のものへ変換する必要があったため，インターフェイス部分にはコード変換ロジックを持たせた．

画面，項目チェック，計算エンジンの３つのロジックを分割したことにより，将来，各種の情報携帯端末や携帯電話向けシステムにて保険料計算システムを稼動させる場合には，画面ロジックの開発とインターフェイスロジックの開発のみで迅速に対応できる体制が整ったと判断している．

Ｗｅｂに関しては，パソコンで動作するＣＯＢＯＬソースを持ち込んで，Ｗｅｂ用にコンパイルし実行形式ファイルを作成するだけで問題なく動作した．

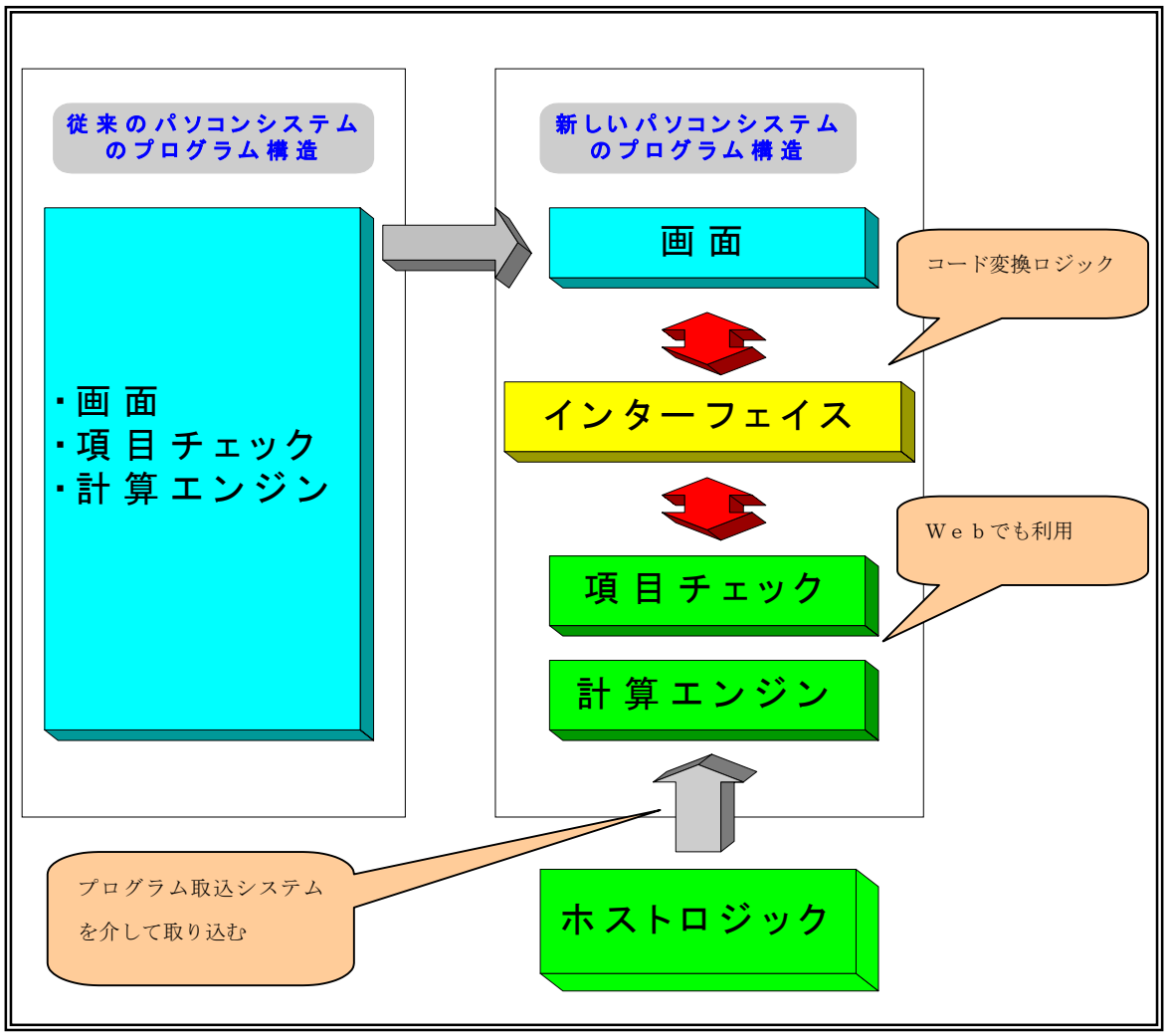


図２　パソコンのプログラム構造

４．４　ホストシステムの改造

ホストロジックをそのままの状態でパソコンシステムへ持ち込むことをコンセプトにする場合，どうしてもホストロジックを改造する必要が生じた．

ホストシステムは，オンラインシステムであるためパソコンシステムと同様に画面ロジックを抱えており，この画面ロジックと項目チェックロジックが微妙に同居しているプログラムが存在した．

また，パソコンシステムでは画面操作中に索引する仕掛けにしている各種マスターファ

イル類につき，ホストロジックでは項目チェックロジックの中で索引しており，このロジックをパソコンシステムに持ち込むと無駄な負荷がかかるため邪魔なロジックであった．

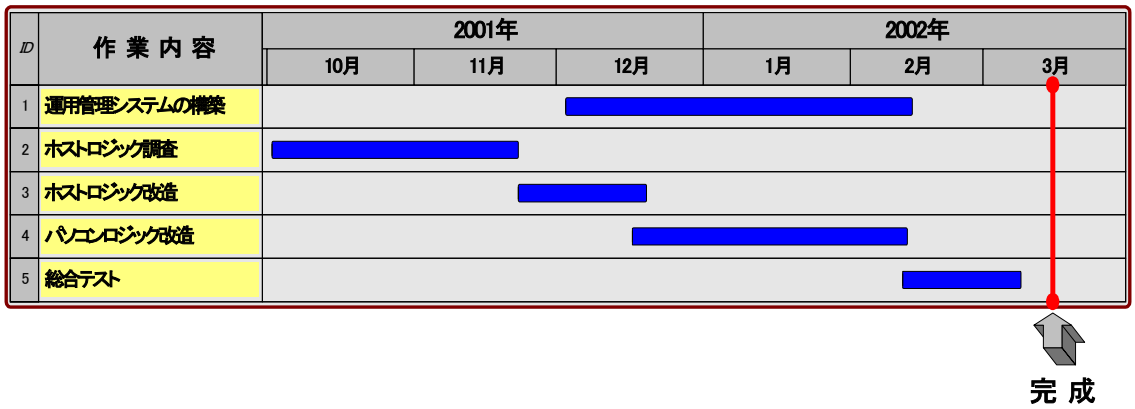
これらのロジックは，ホストには必要であるがパソコンでは必要のないロジックであったため，パソコンに取り込んだ際にはこれらのロジックが動作しない仕組みを予めホストのロジックに対して加えた．これは，ホストからパソコンへソースを取り込んだ後にパソコン側のソースに手を加えることがないようにソース一元化の原則を将来に渡って遵守するために必要なことであった．修正を加えたプログラムは取り込みを前提としている約330 本のプログラム群の内，７本であった．

今回の開発で最も苦労したのがこの７本のプログラムを見つけ出す作業で，修正を加えるべきロジックを明確にするために，約 330 本すべてのプログラムロジックを机上で確認した．１人で約２か月弱の作業であった．モジュール同士の受け渡しエリアの内容を踏まえた上でロジックを追いかけていく作業は，分担して行えるものではなく孤独で地道な作業であった．

４．５　開発の工程

表２に示すように，全開発工程は約５か月であった．前段階でのホストロジック調査を確実に行ったおかげで総合テストは短い期間で完了に持っていくことができた．11 月から12 月にかけて行ったホストロジック修正については一足先に 12 月中にホスト側への本番リリースを行った．パソコンロジック修正に関しては，その半分以上を単体テストに費やした．Ｗｅｂに関しても，同時期に完成している．

表２　開発工程表



４．６　展開

今回のＣＯＢＯＬプログラムにはランタイムソフトが必要であり，約 13,000 台の保険料計算システムが乗るパソコンに配布を行った．配布の方法は，新しい自動車保険の保険料計算システムと同じＣＤ－ＲＯＭに収納し３月末には展開を終えた．本ランタイムソフトを配布しておけば今後のＣＯＢＯＬソフトウエアはソフトウエアのみの配布で問題なく稼動することとなる．

５．共有一元化の効果

５．１　共有一元化の効果

ホスト・パソコン・Ｗｅｂでのロジックの共有一元化が実現したことで，３重開発がなくなり，特にパソコン・Ｗｅｂについては外部業者へ業務委託を行う必要がなくなりコスト面で年間約 2,000 万円強の削減が実現した．

開発効率の面では，ホストロジックを利用したことで保険業務に精通し高い技術を持つ社内の技術者が開発する保険ロジックを共有化でき，プログラム精度が高位に保たれパソコンシステムのテストフェーズでの手戻りが皆無となった．

従来よりオープン系システム担当は，外部業者が納品したソフトウエアをテストすることにほとんどの力を注いできており，特に保険料計算に関しては，間違えられないというプレッシャーと時間がないという焦りに追われていたが，この状況より開放された．

また，もともとホストロジックがモジュール化されていたことから，必然的にパソコンのロジックもモジュール化されたこととなり，一つ一つのプログラムが小さくなったためネットワークでのプログラム配布が格段に楽になった．これは，今回の開発の副産物である．

６．今後の展望

６．１　他の保険種目への展開

今回の開発でＣＯＢＯＬ化を行ったのは自動車保険の保険料計算システムであるが，損害保険会社が扱っている保険は自動車保険だけではない．また，ホストシステムで既にシステム化されている保険種目も多い．保険料率の改定の頻度や収入保険料に関しては，自動車保険が最も多く保険料計算システムの一元化効果は大きいが，これ以外の保険種目に関してもコスト削減や開発効率の面からも自動車保険と同様に一元化することとした．

運用管理システムに関しては，保険種目単位のプログラム群を別々に管理できる仕組みを既に組み込んでおり，他の保険種目を対応したとしても問題なく運用できる．また，今回の開発により，一元化の手順は確立しているので，自信を持って火災保険・傷害保険と順次一元化を進めて行く予定である．

６．２　ＣＯＢＯＬロジック利用の有効性と限界

今回の開発でＣＯＢＯＬソースのロジックをパソコンやＷｅｂサーバで稼動させることが可能であることが証明された訳だが，単純にどんなシステムでもＣＯＢＯＬで稼動させればよいということでもないであろう．システムにおける様々な処理で稼動に耐えれるだけのレスポンスが得られるのか，すべてのロジックがホストと同様の処理ロジックで稼動させることができるのかを検証する必要がある．

今回の一元化ロジックになかった代表的なロジックは，ファイルの入出力ロジックである．ファイル入出力ロジックは，ホストのロジックと同様のもので稼動させることはできず，たとえホストロジックをオープン系システムに持ち込んだとしてもコーディングを変更する必要がありロジック一元化となり得ない．

ファイルの入出力に関して，データベースアクセスロジックにつきパソコンで稼動させ

るべくプロトタイプＣＯＢＯＬプログラムを作成しレスポンス調査を行ったが，**表３**のように同一のロジックを持ったＶＢプログラムよりレスポンスは劣った．

表３　ＤＢアクセス比較表

	VB	COBOL
1回目	2．19s	18．02s
2回目	1．87s	5．44s
3回目	1．21s	4．16s
4回目	1．43s	5．22s
5回目	1．32s	5．11s
平均値	1．60s	7．59s

このような検証により，ホストＣＯＢＯＬロジックのオープン系システムへの持ち込みやオープン系システムでのＣＯＢＯＬ開発では，特にデータベース入出力ロジックのあるシステムは，持ち込みを行ったとしてもコーディングの変更を余儀なくされる点とレスポンス確保の点で問題があると判断した．

しかし，データベース入出力のない内部処理ロジック部分については，レスポンスの点でもＣＯＢＯＬが優勢であり，多くのＣＯＢＯＬ開発要員を活かすためにも，この部分のプログラムは，ＣＯＢＯＬで開発すべきと判断した．

オープン系システム開発においてＣＯＢＯＬ開発要員を活かしつつレスポンスも最高のシステムを作成できる方策は，パソコン・Ｗｅｂともに内部処理ロジック部分についてはＣＯＢＯＬで開発し，それ以外のデータベースファイルアクセスや画面の作成については，従来のオープン系開発言語にて開発する方法がベストと判断している．システムのモジュール化を前提に開発を進めると，この方法は現実的であり今後の開発方針としたい．

７．おわりに

日々システム開発に追われている状況の中においても，常にコスト削減・開発の効率化・信頼性の向上を念頭におきシステム開発における問題点の発掘を行うことで，「不可能では」と固定観念化していたことが，現実性を帯びてくるものである．日頃の仕事の中でも何か疑問に感じることがあれば，すぐに調べてみることが大切であると改めて感じた．

今回の開発も，きっかけはインターネット検索であった．「なんとかならないか」と感じた時に即時に調査できるインターネットの威力をまざまざと見せつけられた感があった．

インターネットに限らず，情報のアンテナを常に張っておくことは，システム開発に従事する者にとっては欠かしてはいけないことであると痛感している．

一方，情報の精査を常に怠らないことも重要である．情報に踊らされて，何でも良く見えてくることは，よくあることである．その情報は，正しいのか，裏側に何か隠された事実はないか，常に考慮しておくことが情報を上手に使いこなすためには不可欠である．従って，情報を入手した場合には，必ず裏をとる習慣をつけておく必要がある．今回，プロトタイププログラムを作成したり，レスポンスやリソース消費量の調査を行うことにより，間違いのない判断ができたのではないかと思っている．

本システムは，現在，何の問題もなく順調に稼動しており，これを礎として当社におけ

るオープン系システムのひとつのモデルが確立したわけで，このモデルを発展させていくことが，今回の開発の真の成功を導くと考える．