

# 開発ツールによる品質と作業効率の向上

## Improvement of Software Quality and Productivity Using Development Tools

あらまし

経営層や現場の課題解決に应运つづけてきた情報システムは、長年の改修を重ね巨大化・複雑化してきた。効率的なIT投資を行う上で、現行資産の有効活用やシステムの早期稼働、改修に当たってのテスト工数の削減などが不可欠となってきている。

そこで、富士通は、このようなシステム開発を実施させるため、アプリケーション開発のライフサイクル全体を踏まえた体系的な開発環境（SDAS）を提供している。

本稿では、SDASによるWebアプリケーション開発における体系立てた開発ツールについて、オープン標準の統合開発環境「Interstage Apworks」、テスト支援「SIMPLIAシリーズ」、ドキュメントリバース「仕様書工房」、および各種プラットフォームに対応した「NetCOBOL」を紹介する。

### Abstract

Information systems, which continue to help on-site management and staff solve problems, have become enormous and complicated after many years of modification. Effective use of current resources, quick system operation, and reduced test work-hours have been essential for efficient IT investment. To implement such system development, Fujitsu has provided the System Development Architecture & Support facilities (SDAS) systematic development environment based on the entire life cycle of application development. This paper introduces some systematic development tools for SDAS-based Web application development. Specifically, it introduces an integrated development environment based on the open standard Interstage Apworks, the SIMPLIA series testing support tool, reverse engineering tool PROSPECS, and a COBOL compiler called NetCOBOL for various platforms.



白取知樹  
(しらとり ともき)

ミドルウェアプラットフォーム事業部 所属  
現在、Apworksの開発に従事。



殿村方規  
(とのむら まさき)

SDAS推進統括部 所属  
現在、SIMPLIAの開発に従事。



佐々木孝次  
(ささき こうじ)

富士通ソフトウェアテクノロジーズ アプリケーションコンポーネント事業部 所属  
現在、ソースコード解析ツールの開発に従事。



阿保谷英夫  
(あばたに ひでお)

ミドルウェアコンポーネント事業部 所属  
現在、COBOL開発環境の開発に従事。

## ま え が き

近年、経営層や現場の課題解決を早期に実現するに当たって、巨大化・複雑化した現行システムに手を入れることは避けられず、新しい最適化システムへの追加機能の組込み方法や改修範囲を見極め、開発量やテスト工数の削減を図ることが必要となってきた。

また、現行資産の有効活用、プロジェクトの小口化による早期稼働の実現、開発のスピードアップに伴うドキュメント保守の効率化など、ものづくりにおけるツールによる品質と作業効率の向上が求められている。

さらに、オープン化の加速によって、J2EEに準拠したWebアプリケーションとそれを実現するフレームワークが注目される中で、Eclipseをはじめとしたオープンソースの活用などに見られるように開発ツールにも業界標準の流れが押し寄せている<sup>(1)</sup>。

このような開発ツールを取り巻く環境の変化の中で、アプリケーションを設計・実装・テスト・保守といった開発のライフサイクル全体でとらえた場合、開発言語やマルチプラットフォーム対応なども踏まえて体系立てた品揃えが品質と作業効率の向上につながる。

本稿では、SDASによるWebアプリケーション開発における体系立てた開発ツールを、以下の中核製品で具体的に紹介する。

- (1) 統合開発環境 “Interstage Apworks”
- (2) テスト支援「SIMPLIAシリーズ」
- (3) ドキュメントリバース「仕様書工房」
- (4) 各種プラットフォームに対応した“NetCOBOL”

## Interstage Apworks

近年における開発期間の短期化とシステムの複雑化により、統合開発環境を用いた作業効率化への期待がますます高まっている。統合開発環境による作業効率化の手法には大きく二つある。

## (1) 定型的な処理の自動生成

定型的な処理を行うコードは毎回新たに作成するよりも品質確保済みのものを再利用した方が開発量やテスト工数を削減できる。そのようなコードを統合開発環境が自動生成することで作業効率を向上させることができる。フレームワークを用いてアプリ

ケーション構造を標準化し、その構造に従って規格化されたコードを自動生成するのが最も効果を発揮する利用方法である。

## (2) 開発手番の削減

ビルドやデバッグといった繰り返し行う操作には手間をかけたくない。そのような操作を少ない手番で行えるようにし、全体的な作業時間を短縮させることが統合開発環境に求められている基本的な要件である。

また、作業効率化のために統合開発環境そのものの習熟期間を短縮したいという要望もある。近年ではオープンソースのEclipse<sup>(2)</sup>が開発環境として急速に普及しており、すでに業界標準の地位を確立したと言える。習熟期間短縮のためにこのような業界標準を採用し、利用者が使い慣れた操作性を提供してほしいという声も非常に強い。業界標準を採用していればサードパーティのソフトハウスにも受け入れてもらいやすいという実態もある。

作業効率化を実現することと、業界標準の操作性を提供すること、これが現在の統合開発環境に求められるものである。本章ではSDASの統合開発環境であるInterstage Apworks<sup>(3)</sup>において、これがどのように実現されているかを紹介する。

## 統合開発環境 “Interstage Apworks”

Interstage Apworks (以下、Apworks) は業界標準であるEclipseを採用した統合開発環境である(図-1)。

ApworksはEclipseをベースに、WebアプリケーションやEJB (Enterprise JavaBeans) といった

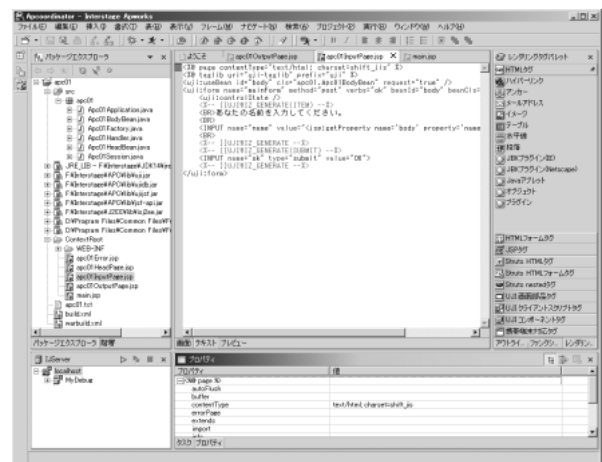


図-1 Interstage Apworks  
Fig.1-Interstage Apworks.

J2EE準拠のアプリケーション開発やAppletなどのJavaクライアントアプリケーションの開発を行うための機能を提供している(図-2)。

開発ライフサイクルからとらえた場合には、開発の各フェーズに合わせて以下の機能を取りそろえている。

- ・要求分析・設計のためのUMLモデリングツール
- ・ビジュアル編集を行うためのHTML/JSPエディタ、GUIビルダおよび電子フォームデザイナー
- ・Interstage Application Server向けに最適化したJ2EEアプリケーションのデバッグ機能や運用環境への配備機能
- ・COBOLアプリケーションの開発機能(COBOLエディタ、COBOLデバッガ)

また、ベースとなるEclipseは一般に公開されているものをそのまま用いるのではなく、富士通で動作検証を行い、検出した不具合を修正した上で組み込んでいる。これにより利用者にはより安定した品質のEclipseをお使いいただくことができる。

ここではApworksが提供する各種機能の中から、自動生成と作業手番の削減というキーワードに関連するものをいくつか紹介する。

## (1) ウィザード

EclipseにはJavaのパッケージやクラスのひな型を生成するウィザードが標準で提供されている。

Apworksではこのほかに例えば以下のものを自動生成するウィザードを提供している。

- ・WebアプリケーションやEJBのひな型
- ・Appletのひな型
- ・HTMLファイルやJSPファイル
- ・J2EEアプリケーションで用いられる各種定義ファイル

これらのウィザードはSDASのフレームワーク(Interstage Application Framework Suite)にも対応しており、フレームワークを用いた開発においてより一層の生産性向上を実現するものである。

## (2) テンプレート

テンプレートとはある処理を行うコードをあらかじめ登録しておいたものである。テンプレートの挿入操作により、そのコードをソースファイル内の任意の位置に埋め込むことができる。頻繁に現れる処理をテンプレート化して利用すれば、利用者がコードを記述する手間が軽減され、コーディング時間が短縮されるとともに単純な入力ミスを防止する効果も得られる。Eclipseにはif文やfor文といったJavaの構文入力を簡単に行うためのテンプレートが標準で提供されている。Apworksではこれに加えて、EJBの参照処理やCOBOLの定型処理といった実際の業務でよく使われる処理をテンプレートとして提供している。利用者が独自のテンプレートを登録す



図-2 Apworksの各種機能  
Fig.2-Key product features of Interstage Apworks.

ることも可能である。

(3) 効率的なデバッグ

Apworksにはデバッグ用途のアプリケーションサーバを同梱している。このアプリケーションサーバのインストールおよび各種設定は、Apworksのインストール時に自動的に行われる。また、WebアプリケーションやEJBを実行するには事前にアプリケーション資産の配備という作業が必要になるが、デバッグ時にはこれもデバッガが自動的に行ってくれる。利用者は面倒な作業を一切行うことなく、デバッグボタンを押すだけでWebアプリケーションやEJBをデバッグできる。

このように、Apworksは業界標準のEclipseをベースに、業務アプリケーション開発を効率化する機能を強化しており、開発現場で実際に求められている要件にマッチした統合開発環境としての価値提供を目指している。

最新のオープン標準への対応

Eclipseは現在バージョン3系が主流になりつつある。Apworksも次版ではEclipseのバージョン3系をベースにする予定である。これからも最新の業界標準に対応しつつ、十分な品質を確保した上で作業効率化を実現するための各種機能を提供し、利用者に安心してお使いいただける統合開発環境たるべく製品を強化していく。

SIMPLIAシリーズ

SDASではメインフレーム時代から、開発プロジェクトで必要とされるいくつかのツールを作成・汎用化して“SIMPLIAシリーズ<sup>(4)</sup>”と呼ばれる開発支援ツール群を提供してきた。本章では、とくにテスト作業の効率化の観点からテスト自動化ツールについて紹介する。

なお、ここではJavaによるServlet/JSP画面のWebアプリケーションを想定し、MVCモデル(図-3)に従って、Model、View、Controllerの各ブロックから構成するものとして、各ブロックで適用するツールを説明する。

(1) 各ブロックの中身のテスト - 単体テスト

ControllerブロックとModelブロックのうち、プログラミングがなされた部分のテストを対象に、二つの製品を提供している。

“SIMPLIA/JF Kiyacker”は、コーディング規

約の作成とJavaソースの規約チェック、およびバグや性能レビューの支援機能を提供する。一般的なツールでは、単にチェック機能を提供するものが多いが、実際のプロジェクト適用時に問題となるのは、開発要員に規約を周知徹底させる方法である。本ツールでは、標準で提供している規約のカスタマイズが可能で、作成した規約をHTMLファイルとして出力することができる。これを配布することにより開発要員に規約を周知させることができる。

“SIMPLIA/JF JudgePruefer”は、ドライバとスタブ(テスト対象のプログラムを動作させるために作成する、呼出し用のプログラムと、テスト対象から呼び出されるが、まだ完成していないプログラムの代わりとなる代替プログラム)の自動生成により、Javaプログラムのテストを自動化する機能を提供する。オープンソースの単体テスト用フレームワークであるJUnitでも同様のことは可能であるが、テスト仕様書との連携が乏しく、適用に際しては工夫が必要になる。本ツールでは、テスト仕様書を作成することができ、このテスト仕様に基づいてドライバとスタブの生成と自動実行・結果判定を行うため、テスト内容のレビューやテスト結果の管理がしやすく、実際の開発作業スタイルに沿った支援ができる。

以上のテストツールは、最近のEclipseベースの開発スタイルに適合すべく、順次Eclipse連携を強化している。

(2) ブロックをつなげてテスト - 結合テスト

ControllerブロックとModelブロックを結合したサーバサイドのテストでは、“SIMPLIA/JF JudgePruefer”を適用するものとし、MVC全体のブロックを結合したテスト支援として、Internet

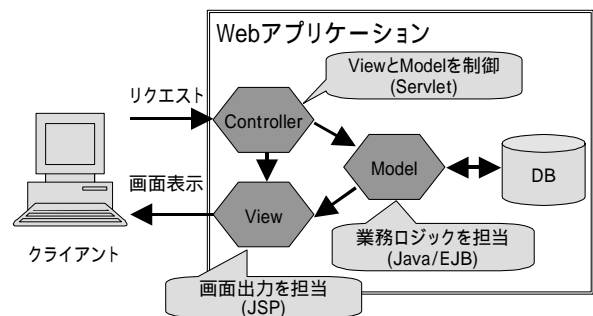


図-3 MVC (Model-View-Controller) モデル  
Fig.3-MVC model.

Explorerの画面からのリグレッションテスト自動化を行う“SIMPLIA/TF-WebTest”を提供している。また、結合テスト仕様書からのテスト支援を目的とした、「Webアプリケーションテスト自動化ツール」も開発中（2006年1月時点）である。

以下、「SIMPLIA/JF JudgePruefer」と、「Webアプリケーションテスト自動化ツール」の詳細を紹介する。

SIMPLIA/JF JudgePruefer

SIMPLIA/JF JudgePrueferは単体テストおよびサーバサイドテストの自動化が可能であり、以下の機能から成る。

(1) 実行機能

テストケースを作成することにより、ドライバとスタブの自動生成とテスト自動実行ができる。テストケースはメソッドの引数や戻り値、ファイルやDBの前状態と後状態などを指定することで作成できる。

(2) 検証機能

テストの自動実行後に、テストケース作成時に指定したメソッドの戻り値、変数/ファイル/DBの後状態などを実際の実行結果と比較して、テスト結果の成功/失敗を判定する。

単体テストにおいては、ドライバとスタブの自動生成により、クラス単位または関係するクラスを結合した状態でのテストが可能になり、ドライバとスタブ作成の手間を省くことができる。また、実行網羅率を確認することにより、ホワイトボックス観点から不足するテストケースを追加することができる。結合テストにおいては、ドライバの自動生成とDBの前状態の自動設定/後状態の自動検証により、従来困難であった、DBの状態に依存したサーバサイドのテストを自動化することができる。

このように利用者はテストケースを作成するだけで自動的にテストができるため、テストケースの作成に専念することができ、テスト品質の向上に役立つ。実際の適用事例としては、ユーザインタフェース設計～システムテスト工程が3箇月というプロジェクトにおいて、例外処理を除く実行網羅率100%に近い単体テストの実現と、結合テスト以降の仕様変更時のリグレッションテスト自動化によるデグレード（機能の後退）防止が実現できたという事例がある。

Webアプリケーションテスト自動化ツール

画面からのテスト作業は、負荷テストを除いてツールによる支援が難しく、従来どおり人手でのテスト作業が多い。クライアントからのリグレッションテストの自動化をねらった製品が他社からも出されているが、初回のテストの効率化には結びつかず、効果があるのは2回目以降のリグレッションテストにおいてである。また、テストケースの抽出も人手で行っており、テストの質についても属人的になっている。このような課題解決の取組みとして、現在開発を進めているのが、クライアントからのWebアプリケーションテストの自動化ツールである。

結合テストにおける機能テストでは、業務仕様を考慮する必要があるが、業務仕様は文章で記述されることが多く、そのドキュメントフォーマットも多様であるため、ツールによる解析が難しい。しかし、画面遷移図および画面項目定義はそれを記述するツール、またはファイルから情報抽出することが比較的容易であり、テストケースとなる画面遷移パターンも生成しやすい。本ツールでは、IBM社のUMLモデリングツールである、Rational XDE形式で保存された画面遷移図および画面項目定義から画面遷移テストのテストケースを生成し、Microsoft社のExcelのテスト仕様書として作成することができる（図-4）。

テストケースとしては、画面遷移図の開始から終了へのパスの中で、画面項目定義の情報をもとに、以下のテスト観点から生成する。生成されたテスト仕様は、Excelシート上でテストケースの追加/削除が可能である。

(1) 入力データ（正常/異常）

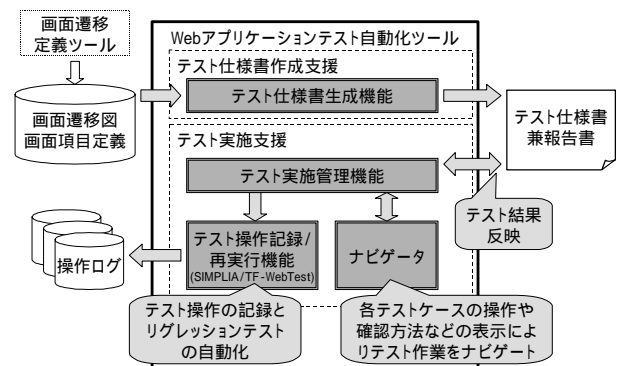


図-4 Webアプリケーションテスト自動化ツール  
Fig.4-Automated testing tool for Web application.

- (2) 表などの繰返し項目の画面表示件数 (0/1/N)  
 (3) 画面遷移図内のループの実行回数

各テストケースの画面項目定義の各項目の型から値の候補を自動生成する。とくに性別などのコードが決まっている項目は、コード定義値を表示して選択するだけで入力可能である。

このようにして作成されたテスト仕様書をもとにテストを行う際には、「テスト実行ナビゲータ」(以下、ナビゲータ)と呼ばれるテストをナビゲートする画面を表示し、1テストケースずつテスト作業を支援する。ナビゲータの画面では、テスト仕様書から1テストケースを抽出して、テスト前に行うべき作業、およびテスト画面上の入力操作とその結果の確認内容を表示する。利用者はこのナビゲータに従ってテスト作業を行い、テスト結果をナビゲータ画面に入力する。このテスト結果はテスト仕様書に反映され、テスト仕様書兼結果報告書として使用できる。

さらに“SIMPLIA/TF-WebTest”と連携することにより、テスト操作を記録・保存しておき、リグレッションテスト時には自動的に再実行を可能にする。

### 仕様書工房

現在のアプリケーション開発では、UML、フレームワーク、ソースプログラム自動生成などの機能を持つ開発環境を使用することが一般的である。ソースプログラムの生産性は、これらの開発環境を使用することにより飛躍的に向上した。しかし、その反面でドキュメントが存在しないプログラムが増加しているという問題がある。これはドキュメントの作成に当たって次のような問題を抱えているからである。

- (1) 開発期間の短期化で、現在作業中のプログラム修正が優先され、ドキュメントの作成が後回しになる。  
 (2) 頻繁に発生する仕様変更により、ソースプログラムとドキュメントの同期の維持が難しい。

このようなことから、システム開発の中ではドキュメントの作成が軽視される傾向にある。

ドキュメントの必要性は、新規開発時よりシステム稼働後の保守時のほうが大きい。保守は既存ソースプログラムを変更する作業を行うため、変更作業

の前に既存ソースプログラムを理解する必要がある。また、年月が経つと、開発担当者と保守担当者が違うことが多く、保守担当者はソースプログラムを理解することに苦労する。さらに、ソースプログラムを理解する手掛かりとして関連するドキュメントを参照するが、これらのドキュメントはソースプログラムと同期して更新されていない場合が多い。

このようなドキュメントに関する問題を解決するのがドキュメントリバースツールである。ドキュメントリバースツールを使用する利点は三つある。

一つ目は、正確なドキュメントを作成できることである。現在稼働中のソースプログラムを入力とするため、生成されるドキュメントの信頼性は高い。

二つ目は、ソースプログラムしかない状態からドキュメントを作成できることである。他人が作成したソースプログラムやオープンソースプログラムの理解に効果を発揮する。

三つ目は、ドキュメントの作成時間を短縮できることである。納品物を作成するときや、プロトタイプ開発のようにソースプログラムを作成しながら内部仕様を決めていくやり方に効果大きい。

ドキュメントリバースツール「仕様書工房」

ドキュメントリバースツールはいくつか存在する。その多くは、ソースプログラムを入力とし、コメントの抽出ルールに従ってソースプログラムのコメントを反映したドキュメントを自動生成する。

富士通では「仕様書工房<sup>6)</sup>」というドキュメントリバースツールを開発・商品化している。仕様書工房の仕様は次のとおりである。

- (1) 入力言語：Java, C/C++, C#, VisualBasic (以下、VB)  
 (2) 生成種類：47種類～69種類(言語により異なる)  
 定義情報(クラス一覧、メソッド一覧など)  
 説明書(クラス説明、メソッド説明など)  
 参照情報(クラス参照一覧、変数参照一覧など)  
 構成情報(クラス階層図、メソッド呼出図など)  
 統計情報(ファイル統計、メソッド統計など)  
 差分情報(ファイル差分、メソッド差分など)  
 (3) 生成形式：印刷, HTML, Word, CSV

仕様書工房は、ドキュメントの作成と、ソースプログラムを理解する局面で使える。その特徴は、作業を支援するフォームエディタとソースブラウザと

いうツールを備えている点である。利用者はドキュメントの体裁を標準化していることが多い。この体裁を合わせるための機能がフォームエディタである。フォームエディタはドキュメントの形式を定義するツールで、矩形域、罫線、コメント、ソース解析結果などをマウス操作で配置する作図ツールである。操作しやすく、印刷イメージを把握しやすいのが特徴である（図-5）。

もう一つのソースブラウザは、ソースプログラムを参照するツールである。ソースプログラムの予約語、識別子、コメントが色分けして表示されるウィンドウ、クラスやメソッドの定義一覧が表示されるウィンドウ、参照情報や構成情報が表示されるウィンドウで構成されている。これを使うことでソースプログラムの可読性が向上する。また、表示を「コメントキーワード設定」に切り替えると、抽出キーワードとドキュメントへ反映するコメントが色分けして表示される。これを使うことで簡単にソースプログラムから抽出するコメントを定義できる。

ロジックの抽象化へ

ソフトウェア開発は大きく「分析」、「設計」、「実装」、「テスト」の工程に分けられるが、現在のドキュメントリパースツールで生成するドキュメントは「実装」工程で作成されるソースプログラムに近いレベルのドキュメントである。今後は、「分析」、「設計」工程レベルのドキュメントを生成するために、ソースプログラムから意味のあるロジックを抽出する技術を確立し、抽象度の高いドキュメントを生成することが課題である。

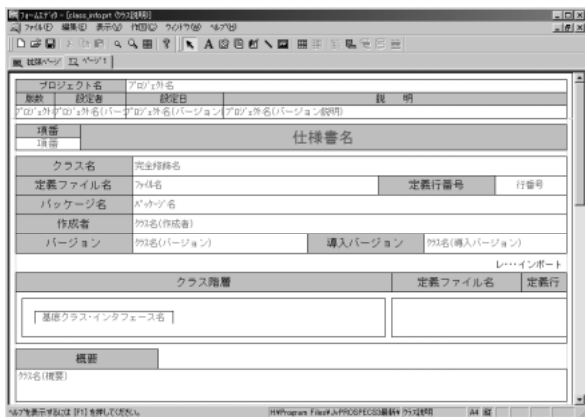


図-5 仕様書工房（説明書）  
Fig.5-Example of PROSPECS “Class description”.

NetCOBOL

オープンシステムの基盤システム構築においては、言語をJava、VBにするのかまたはCOBOLにするのかなどの議論はつきもので、各言語それぞれの特長を考慮する必要がある。Javaならインターネットに対応し、柔軟なシステム構築ができる。一方、COBOLならメインフレーム、オフコンで動作している既存資産を流用することができる。これからのシステム構築では、現状の資産、構築しようとするシステムの目的に合わせて適材適所に言語を選ぶことが重要であり、フロント系業務はJavaで、ビジネスロジックは既存資産を流用してCOBOLで構築する事例も増えている。

富士通が提供するCOBOL開発環境“NetCOBOL®”は、COBOLによる開発を設計からテストまで統合的に支援するとともに、業界標準であるEclipseを採用した統合開発環境“Interstage Apworks”と連携して、JavaとCOBOLを利用したシステム開発を短期間で効率良く構築できる。また、Microsoft社が提供する.NET Frameworkベースのシステム構築も、.NETの統合開発環境Visual Studio.NETでVB、C#と同様にシステム開発を行うことができる。NetCOBOL統合開発環境は、設計・プログラミング・テスト・保守といった開発サイクル全般を支援している。本章では、NetCOBOL統合開発環境が提供しているコンパイラ、デバッガなどの基本機能のほか、画面・帳票編集、テスト支援、ドキュメント作成支援機能を紹介する。

(1) プロジェクトマネージャ

NetCOBOL開発環境のメインツール「プロジェクトマネージャ」では、開発資産の依存関係を管理しながら、コンパイラ、デバッガなどの各種支援ツールと連携操作ができる。従来仕様のアプリケーションからオブジェクト指向COBOL仕様のアプリケーションまで、効率良く開発できる。プロジェクトマネージャを利用した開発イメージを図-6に示す。

(2) コンパイラ

国際規格に準拠し、新規格COBOL2002で導入されたオブジェクト指向機能、主要な業界標準仕様、富士通メインフレーム、オフコンと共通の富士通標準仕様を実現している。

コード系は、シフトJIS、EUC、Unicode、

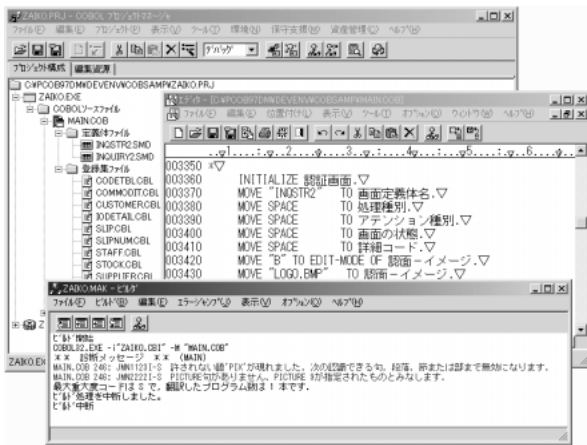


図-6 プロジェクトマネージャを利用した開発イメージ  
Fig.6-Development image by use of Project Manager.

EBCDIC-JEFに対応しており、他製品との連携を含めた広範囲で統一された文字コードの利用が可能となる。

### (3) デバッグ

以下の豊富な対話型デバッグ機能により、テスト作業の効率化、トラブルの早期解決が可能である。

- ・自由度の高い中断点設定
- ・データ内容監視
- ・データ変更中断
- ・副プログラム単独でのテストなど

また、運用時に詳細なデバッグ情報を出力する診断レポートを装備しており、トラブルシューティングに費やす調査時間を短縮できる。

### (4) 画面・帳票編集

ディスプレイ（画面）のどこに何を入力・表示するかを設定した「画面定義体」、プリンタでどこに何を印刷するかを設定した「帳票定義体」および帳票定義体に重畳する「オーバーレイ定義体」を使用することにより、COBOLによるきめ細かい画面帳票アプリケーションを作成することができる。また、画面・帳票定義体を、COBOLプログラマが使い慣れたWRITE文やREAD文で、通常のファイルを扱うのと同じように使用できるので、アプリケーションの短期構築を実現できる。

### (5) 開発・保守支援 (SIMPLIA/COBOL支援キット)

以下の開発/保守を支援する様々な機能を提供し

ている。

#### ・テスト支援

変換ツールや対話型エディタなどにより、テストデータ作成工程の削減、既存データを活用した効率の良いデータ作成が可能である。

#### ・ドキュメント作成支援

画面・帳票設計書、ファイル設計書、アプリケーション構造設計書、モジュール設計書などの保守ドキュメントを開発資産から自動生成する。

#### ・プラットフォーム間の移行支援

メインフレーム・オフコン・オープンシステム間のデータ/ソース流通を支援する。

## む す び

富士通におけるJavaやCOBOLでのアプリケーション開発は年々増加傾向にあり、特定のベンダ技術に縛られることなくマルチプラットフォームに通用する体系立てられた開発ツールは、ますます重要な位置付けとなる。

また、設計工程の情報を実装やテストに有効活用するなどMDA ( Model Driven Architecture ) をはじめとした新技術やSOA ( Service Oriented Architecture ) に基づく開発手法なども踏まえ、より実践的な開発ツールの提供、当社技術の優位性追求を進めていく。

### 参考文献

- (1) パッケージソフト&ソリューション総覧2005 . 日経コンピュータ, 2005年8月22日号 PR別冊 .
- (2) Eclipse.org .  
<http://www.eclipse.org/>
- (3) Interstage Apworks製品ホームページ .  
<http://interstage.fujitsu.com/jp/apworks/>
- (4) SIMPLIA製品ホームページ .  
<http://software.fujitsu.com/jp/simplia/>
- (5) 仕様書工房ホームページ .  
<http://jp.fujitsu.com/fst/services/frontier/kobo/>
- (6) NetCOBOLホームページ .  
<http://software.fujitsu.com/jp/cobol>