マルチ・プラットフォーム時代の 2200/IX/CS 統合開発環境

Integrated Development Environment for 2200/IX/CS Series in Era of Multiple Platform

元 山 裕 之

要 約 企業システムが，メインフレーム，UNIX，さらに Windows といったマルチ・プラットフォーム環境に展開されるようになって久しい．そこで稼働するアプリケーションも一様ではなく，使用するプログラミング言語や開発環境も様々である．これは選択肢の拡大というユーザ・メリットの反面，開発方法や管理方法の多様化・複雑化に繋がり，開発・保守の管理コスト増大や生産性低下の要因となっている．

本稿では，マルチ・プラットフォーム時代の開発環境が持つべき要件を挙げ，その具体的な実装として，2200/IX/CS 向けに開発した統合開発環境「CSCEAD」の，基本的な仕組みや重要機能の概要を説明している．

Abstract A long time has passed since the Enterprise systems were deployed on the multiple platform environments including mainframe, UNIX and Windows systems. The application programs running on the multiple platform environments are not uniformed, written in different programming languages and developed on the different development Environments. This brings the benefits of a wider range of options available to customers, but results in the increase in diversification and complexity of both and configuration management method, which becomes a factor increasing the management cost of development and maintenance, and decreasing the productivity in development activities.

This paper shows the requirements for the development environment in the ere of multiple platforms, and explains the outline of the fundamental mechanism and the key functions of the Integrated Development Environment“ CSCEAD ”which was developed for 2200/IX/CS Series as a definite implementation.

1. はじめに

開発環境とは何か．狭義には，エディタ，コンパイラ，リンカおよびデバッガを含む開発基本ツールの集合体である．これらの開発ツール群は，開発時に必須の基本ツールであり，特に個人の開発生産性の向上に重点を置いていることから，個人による単独の開発作業には充分であるが，複数人からなるチーム開発では充分とは言えない．チーム開発では，排他制御を考慮した構成管理，変更管理，進捗管理などの総合的な管理支援機能や，上流工程の設計支援機能も加えた開発支援ツールが必要となる．企業における開発作業はチーム作業となるのが常であり，大規模な開発では開発支援ツールも必須となる．

本稿の対象とする開発環境は，広義に捉えた開発環境，すなわち「チーム開発を支援する総合的な機能体」を指す．日本ユニシス（以下，当社）のメインフレームであるシリーズ 2200，HMP IX シリーズ，CMP CS シリーズ（以下 2200/IX/CS）の 3 GL アプリケーション開発環境の現状を把握し，マルチ・プラットフォーム環境で新開発環境に求められる要件，開発へのアプローチ，機能やメリットを順に説明する．

2. 開発環境の現状と問題点

最近 20 年間のシリーズ 2200 の 3 GL アプリケーション開発環境で，ユーザに影響を与えた大きな変化を振り返ってみる．エディタは行モードのエディタ（@ED，@CTS）から，全画面モードのエディタ（@IPF）に変化した．コンパイラは各言語独立（@ACOB，@FTN）から，UCS コンパイラ（@UCOB，@UC…）となりコンパイラのバックエンドを共通化した．リンカは BM（Basic　Mode：基本モード）で静的連結必須（@MAP）から，EM（Extend　Mode：拡張モード）で動的連結を用い，コンパイル後即実行が可能になった．デバッガは，事後ダンプ（@PMD）の解析から，会話型デバッガ（@PADS），さらに GUI デバッガ（OADE-Win）へと変化している．大規模システム開発用の開発支援ツールとしては SDASS，TSX Ⅱ，IDES を提供してきた．

最後に挙げた，開発支援ツールに焦点をあてる．これらは，すべて当社の独自開発ツールであり，次の観点から開発された．

1） 開発成果物の管理

一般に言われる「構成管理」や「変更管理」などの機能である．開発支援ツールの基本機能であり，ライン管理機能やプログラム更新機能を有している（SDASS，TSXII，IDES）．

2） 開発プロジェクトの管理

開発/修正作業を人の活動面から捉え，作業の漏れや遅れがないように支援する機能である（SDASS-ADMS，TSX-ADMS，IDES/2200 プログラム管理）．また，許可された人が，許可された開発や修正のみ実施できるよう，セキュリティ面の機能も有している．

3） 品質の維持/向上支援

ある構成要素への修正の影響を把握して，不整合の発生を未然に防ぐ影響分析機能である（IDES/2200 プログラム管理）．またテストの網羅性を検証することでテスト品質を保証している（TRTR-C，IDES/ITEST）．

4） 開発生産性の向上

実際にプログラムを作成/修正する担当者を支援する機能であり，パラメタによる COBOL ソースの生成機能（PASE 1100，SDASS/GCOB），や JSP（Jackson Structured　Programming）に基づくビジュアルな手法で，構造化プログラム作成支援を行う機能（IDES C/S パック）を提供している．

しかしながら，これらの開発支援ツールは，シリーズ 2200 向けアプリケーションしか開発できず，今日の UNIX，Windows といったマルチ・プラットフォーム環境では，次のような問題点が見えてくる．

- 他のプラットフォームと管理手法が異なり，全社統一の管理基準を設定することが困難．
- あるプラットフォームで利用可能な開発支援機能が，別のプラットフォームでは利用できない．
- ソースに対応する仕様書などの成果物が何のリンクもなく別々に管理されているため，相互に関連するプログラムのソースや仕様書を検索することが困難．

・全ての指示が文字入力（CUI）であり，作業効率が悪い．Windows では，GUI が常識である．
・全ての工程で，高価なシリーズ 2200 ホスト資産を多量に消費．

3. 新開発環境の要件

上述した開発環境の現状と問題点を踏まえ，マルチ・プラットフォーム時代の開発環境が具備すべき要件を洗い出した．

1） プラットフォームに依存しない管理手法

開発・保守作業の中で発生する開発成果物の管理や進捗管理などは，ターゲット・プラットフォームに依存することなく，各プラットフォーム共通に実行できる機能である．ここで，ターゲット・プラットフォームとは開発したアプリケーションの稼働するプラットフォームを意味する．一方，コンパイル，連結編集およびデバッグ作業などは，固有のターゲット・プラットフォームに依存せざるを得ない部分であるが，極小化し明確に切り分けることにより，管理機能をターゲット・プラットフォーム非依存とすることができる．

2） ISV ツールの有効利用

Windows の世界では，開発環境向けに各種ソフトウェアが提供されている．各ツールは狭い範囲の分野を対象としたものから，統合開発環境として広い分野を対象としたものまで各種存在し，日進月歩である．ユーザの選択肢は多く，望むものを選ぶことができる．これらをうまく利用することで，開発・保守の生産性を向上させることが期待される．そのためには，開発・保守の実務が Windows 上で行えることが大前提となる．

3） Windows の GUI によるプログラム開発

シリーズ 2200 のソース・プログラムは INFOConnect UTS 画面（ホスト端末画面）から IPF 等のエディタを起動して作成・保守するのが従来からの手法であった．この方法では，利用者はシリーズ 2200 の知識，特にエディタのコマンドを知っている必要があった．このことはシリーズ 2200 を全く知らない要員を活用しにくい原因の一つになっていた．ソース・プログラムのエディット指令，ビルド（コンパイルおよび連結編集）指令，デバッグ指令などを Windows の GUI 操作によって直感的に行うことができれば，シリーズ 2200 以外の開発環境に慣れている要員であっても，比較的短期間に作業習熟を図ることができる．

4） 既存開発環境からの移行支援

IDES 等の開発・保守環境を使用しているユーザの資産（プログラム，管理情報等）を，新開発環境へ移行するための支援機能が必要である．また，ユーザ独自の開発・保守環境を設定している場合にも，ソース・プログラムの解析により，新開発環境への移行を支援できるようにする必要がある．

5） チーム開発環境としての基本機能

次に挙げる機能群は，チーム作業を考慮した開発環境として備えていなければならない基本機能である．

・構成管理

開発中のシステムを構成するソフトウェアを識別し，それぞれのソフトウェアのバージョンや変更手続きを管理している．過去の任意のバージョンの状態を再現可能である．

・更新の排他制御

多数のチーム・メンバが混乱なく作業を進めるために必要な機能である．チェックイン・チェックアウト機能により，誰かが更新中の部品を誤って他人が変更してしまう等の事故を防止することができる．

・並行開発（ライン管理）のサポート

更新の排他制御機能にかかわらず，例えば計画保守と緊急保守の並行作業を可能にする機能である．これは「サイドライン」の考え方を導入し実現している．サイドラインに対し，本流のラインを「メインライン」と呼ぶ．

・チーム内に発生する各種アクティビティの管理

チーム内にどのようなアクション項目があり，何が終わっていて何が未完了かを把握し，全体として遅れを予防し，作業の漏れをなくすために必要な機能である．

4. 新開発環境「CSCEAD」の開発

当社提供のソフトウェア「TeamFactory」（チーム開発支援ツール）および「OADE-Win」（シリーズ 2200 用プログラム開発支援ツール）を利用することにより，前章のいくつかの要件を満たすことができる．新開発環境「CSCEAD」（CMP Server Cooperated Environment for Application Development）は，無関係に存在するこれらのソフトウェアを連携させ，不足部分を開発した統合開発環境である．新開発環境「CSCEAD」の柱となる基本的な仕組みを説明する．

4.1 3 階層モデル

CSCEAD は，役割に応じて三つのプラットフォームに機能を分散し，それらの連携によって目的を達成する仕組みとした（図 1）．

1） 構成管理サーバ

ターゲット・プラットフォームによらず，全てのプログラムソース，コンパイル結果，連結編集結果，関連ドキュメントを集中管理し，単一の管理機能を提供する．ここには以下の機能が用意されている．

・構成管理機能

・変更管理機能

・アクション管理機能

2） ビルドサーバ

実際のビルド（コンパイルおよび連結編集）を行うには，ビルド結果の実行可能プログラムが稼働するターゲット・マシンが必要である．メインフレーム（2200/IX/CS），UNIX，および Windows が該当し，これらを「ビルドサーバ」と呼ぶ．後述の開発クライアントからビルドの指示を行ったとき，ビルドに必要なソース・プログラムなどを実際のビルド・サーバに自動転送し，自動ビルドを行う．

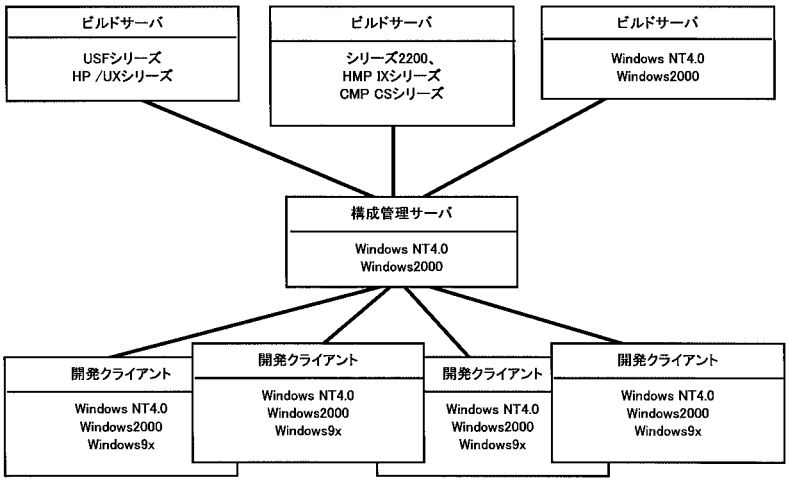

図 1　3 階層モデル

3）　開発クライアント

個々のチーム・メンバが操作する Windows である．ここにユーザ・インタフェース部が搭載され，ここから開発・保守に必要な全ての操作ができる．

4.2　成果物管理モデル

Windows や UNIX の世界では，ソフトウェア部品は，それらを組み合わせて最終的に作成されるプログラム（通常は EXE ファイルや DLL ファイル．これを最終成果物と呼ぶ）ごとに管理するのが普通である．Visual Studio や JDK などの開発環境では，このような管理単位を「プロジェクト」として扱っている．「プロジェクト」による管理手法は，Visual な管理ツールを用いることにより，その構成が直感的に把握可能になるメリットがある．

一方，シリーズ 2200 に代表されるメインフレームの世界では，プロジェクト単位でなくファイル（シリーズ 2200 ではエレメント）のタイプあるいはサブタイプによって分類管理されてきた．すなわち，ソース・ファイルや OM（Object Module）・ファイル，ZOOM（Zero Overhead Object Module）・ファイルなどによる分類管理となる．このような括りによる管理手法は，プログラム・ファイルを見ただけでは内部の構造が分かりづらいという問題があるが，多くのユーザで，既に確立していることも事実である．そこで CSCEAD は以下の考え方を採用した（図 2）．

- 構成管理サーバ：コンポーネント単位にプロジェクトを構成し，その単位で管理する．UNIX でのコンポーネントは，一つの make に相当する．2200/IX/CS では，一つの実行可能プログラム（ABS，ZOOM）に相当し，コンポーネントにはそのプログラムを構成するすべてのソース（プログラム，登録集，連結編集用シンボリック，プロジェクト情報）が含まれる．
- ビルドサーバ：従来のプログラム・ファイル単位による管理を踏襲し，現行の運用方法を変更することなく接続できるようにする．

なお CSCEAD に於いては，ビルドサーバ上のソース・ファイルは，あくまでビルドのための一時的なファイルであり，正のソース・ファイルは構成管理サーバ内にある．これに対し，ビルド結果である REL/OM や ABS/ZOOM はビルドサーバ上で管理される．

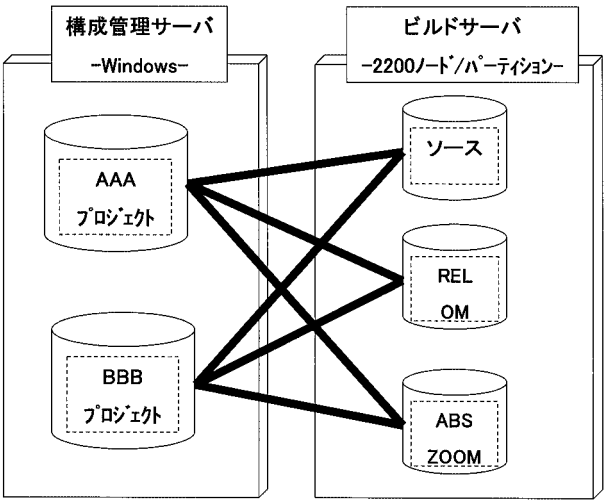

図 2　成果物管理モデル

4.3　ワークベンチ機能

アプリケーション開発作業は，エディット/ビルド（コンパイルおよび連結編集）/デバッグの三種類の作業を一連の作業として繰り返すことである．従来のシリーズ

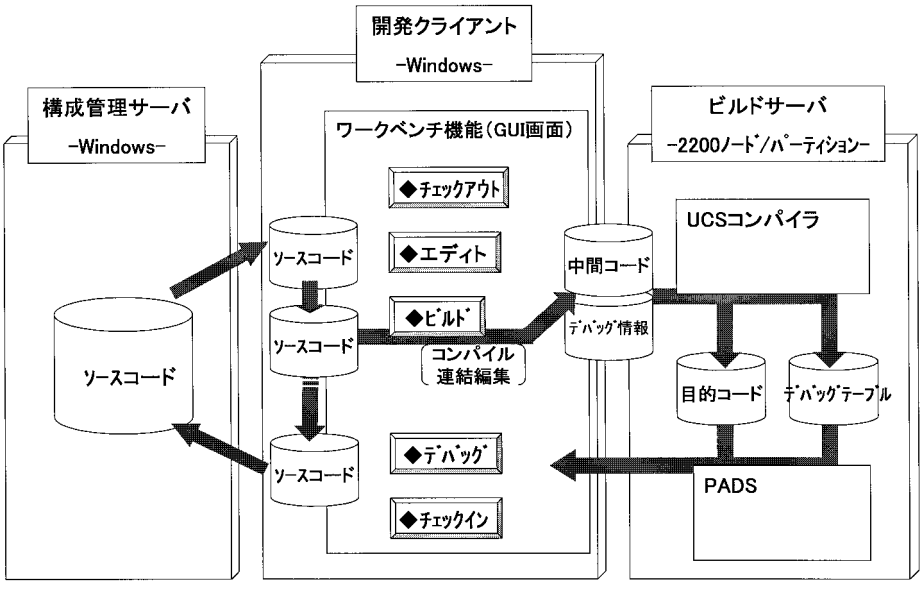

図 3　ワークベンチ機能の仕組み

2200 の開発環境では，これらの作業は UTS 専用端末から CUI ベースで別プロセッサを呼び出す必要があり，作業の中断を余儀なくされていた．CSCEAD ではソースコードを構成管理サーバからチェックアウト機能で取り出し，開発クライアント用 Windows 環境に置くことにより作業を進めることができる．ユーザ・インタフェースは GUI となり，これら三種類の作業を相互連携させることによって，作業に連続性を持たせることが可能となった．作業完了後，チェックイン機能で構成管理サーバにソースコードを戻す（図 3）．

ワークベンチ機能の中のビルドやデバッグでは，開発クライアント用 Windows とビルドサーバ用 2200 ノードを連携させている．コンパイルでは開発クライアントで中間コードまで生成した後，ビルドサーバにファイル転送し，UCS コンパイラのバックエンドに渡す．コンパイルの続きと連結編集は 2200 ノードで実行される．ビルド時のメッセージ等の情報は Windows のワークベンチ GUI 画面に表示される．デバッグでは，Windows のワークベンチ GUI 画面で指示した内容を解釈し，2200 ノードの PADS から情報を入手し，Windows のワークベンチ GUI 画面に表示している．

5. 新開発環境「CSCEAD」がもたらすもの

CSCEAD のソフトウェア構成は次に示す通りである（図 4）．商品としての CSCEAD は破線部分であるが，OADE-Win および TeamFactory を加えて始めて機能する．これらを合わせた全体を「CSCEAD」の開発環境と呼ぶ．

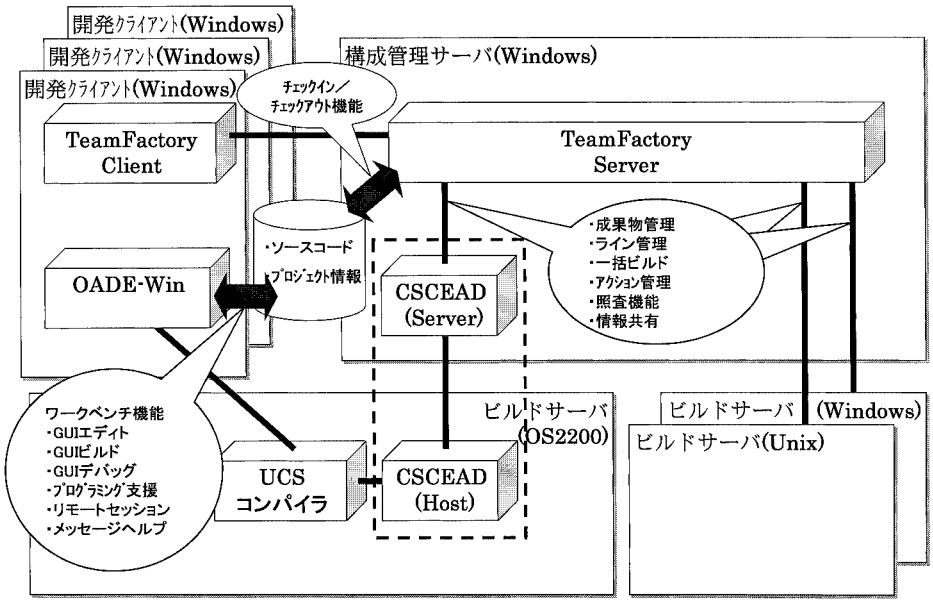

図 4　新開発環境「CSCEAD」のソフトウェア構成

次にいくつかの重要な機能をピックアップし，その概要とメリットを述べる．説明の中では，構成要素である CSCEAD，OADE-Win，および TeamFactory の区別はしていない．

5.1 成果物管理およびライン管理

エクスプローラ風の画面で，システムの階層構造を任意に定義し，バージョン管理が可能である．成果物個々を別々に管理するのではなく，構造を含め成果物相互の整合性を維持しながらバージョン管理をしているので，例えば本番環境へのリリース・レベルから対応するソースや設計書を特定することが容易にできる．過去の任意のバージョンを復元可能であり，バージョン間の差分やリスト等の出力機能も装備している．成果物はチェックイン/チェックアウト方式の排他制御で管理しており，チーム開発に対応している．ファイルの実体は，差分で保管しておりディスク容量を節約できる．

また，特定時点の対象システムあるいはサブシステムを「ベースライン」として確定保存することができる．一つのベースラインから複数のラインをブランチして並行開発を可能としている．下位ラインの管理者は上位ラインの管理者により設定された作業範囲内の構成要素だけを更新でき，作業終了後上位ラインにマージすることができるので，目的別の作業を並行して矛盾なく行える（図 5）．さらに変更履歴を視覚的に表示でき，一目瞭然となる．

異種プラットフォームの成果物を同じ操作で一元管理できるので，管理方法が共通化されるメリットがある．

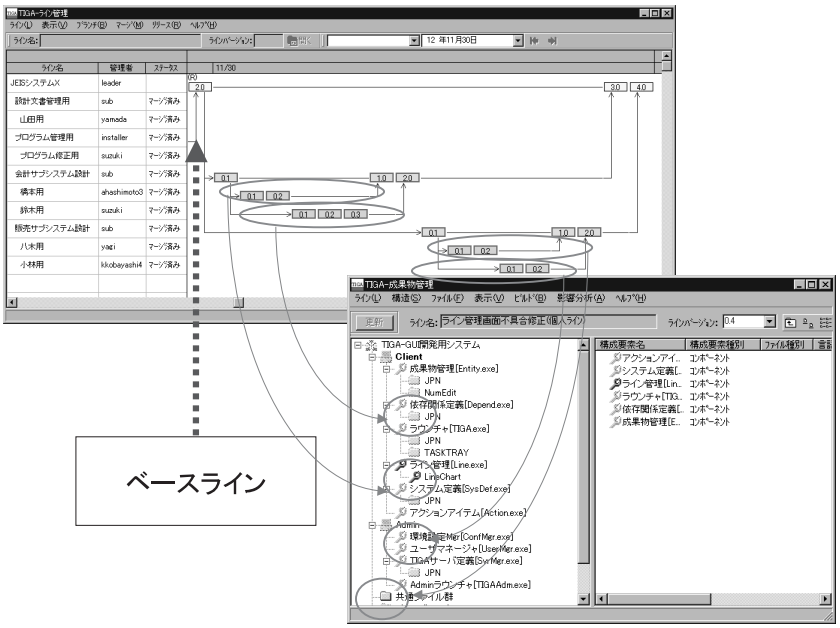

図 5　成果物管理，ライン管理画面例

5.2 アクション管理

サイドラインあるいは個人ラインとして派生したラインに関し，そのライン上での開発作業の進捗状況を視覚的に確認できる．チーム内の進捗状況がだれにでも把握できるようになる．また，チェックインやチェックアウト，ブランチ，マージなどのタ

イミング毎にメールが自動送付され，作業指示などコミュニケーションが円滑になる．また作業漏れが無くなり，品質向上に繋がる．これらはアクション管理画面（図 6）で，常時確認可能である．

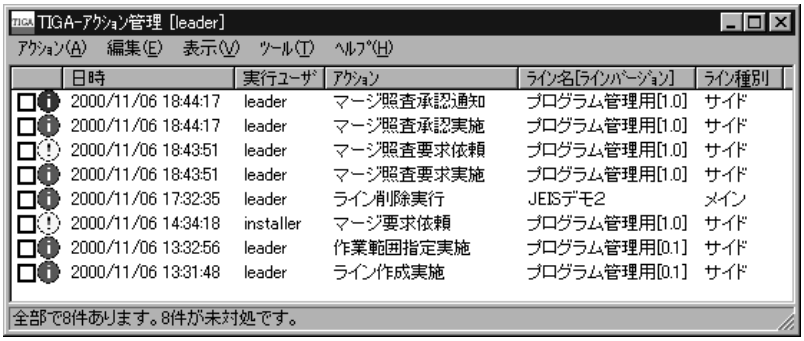

図 6　アクション管理画面例

5.3　プログラム作成支援，設計支援

5.3.1　オブジェクト指向プログラミング支援

UCS　OOCOBOL は，従来の UCS　COBOL の機能向上版で，開発時の GUI 機能を強化し，オブジェクト指向機能（次期 COBOL 規格の先取り）を追加している．Java や C++等オブジェクト指向言語を使う場合，開発上流工程のオブジェクト指向分析・設計ツールを用いることが常識となっている．UML で記述した設計情報から，Java や C++のソース・コードを生成する機能も一般的である．Rational 社の Rose は，この分野における代表的なツールであるが，OOCOBOL ソース・コード生成機能はない．当社では，Rose で作成した設計情報から OOCOBOL ソース・コードを生成する機能を開発した（図 7）．

オブジェクト指向による設計ができるが COBOL は知らない，あるいは逆に COBOL は知っているがオブジェクト指向設計ができないというケースにおいても，本機能をこれらの技術者の繋ぎ役として利用することが可能である．

5.3.2　構造化プログラミング支援

当社の IDES C/S パックの木構造図エディタで，プログラム構造を描くことができる．これは，モジュール仕様書の一部であり，他の管理情報と合わせて「仕様書ファイル」と呼ばれる．IDES　C/S パックは，「仕様書ファイル」をベースに，「ソース・コード生成」機能で必要なソース・プログラムを作り出すことができる．木構造図エディタは JSP に近い記法を用いており，これを用いることによってプログラム・ロジックの組み立て方が均質化される効果がある．このことはレビューやデバッグの手順が均質化されることにも繋がり，生産性の向上及び品質の向上に寄与している．CSCEAD 環境においても，IDES　C/S パックが共存でき，生成されたソース・プログラムを OADE-Win プロジェクトに含め，構成管理サーバに保存できるようになった．

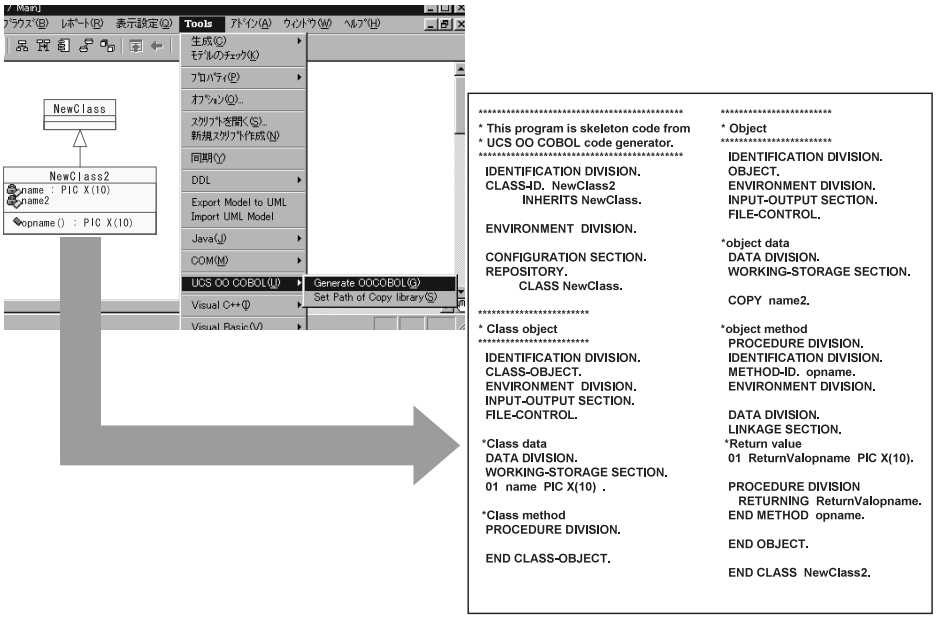

図 7　OOCOBOL Generator

5.4 GUI デバッグ

GUI デバッグ機能は，トレース/トラップ/ステップ実行など，UNIX や Windows 上のデバッグツールと同等のビジュアルな機能を有する（図 8）．シリーズ 2200 のバッチおよびデマンドさらにトランザクション・プログラムも対象としており，動的実行をしながらデバッグもできる．従来の@PADS ではトランザクションに関して，エラー時の静的デバッグ情報をもとにした事後分析しかできなかった．ダンプ解析に慣れた人にも分かり易いように，ダンプの表示機能もある．デバッグ作業はエディッ

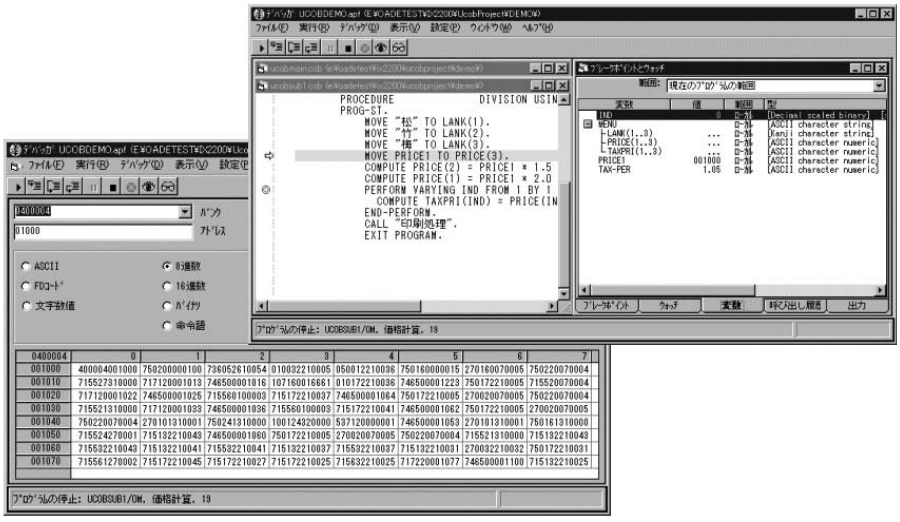

図 8　GUI デバッグ

ト作業とともに，プログラム開発作業の中でも大きな比重を占める作業であり，GUI操作により生産性向上が期待できる．

5.5 一 括 ビ ル ド

チーム内の各開発メンバがソースの作成・修正を終えた後に行うビルド（コンパイルおよび連結編集）作業は，ソースコードの正しさや連結編集でエラーがないことの確認を目的として，ワークベンチ機能の「GUI ビルド」で行う．これに対し，開発プロジェクト・チームを構成する全メンバの開発・テスト作業が終わった後に，当該システムあるいは全社で定めたコンパイル・リンクの標準に従ってビルドし直すことが必要である．この工程は構成管理サーバの GUI を用いて，管理者が行うもので「一括ビルド」と呼ぶ．

図 9 は，一括ビルドの環境設定画面である．詳細な設定により，クリーンな最終成果物を構築し，成果物の品質保証を可能にしている．

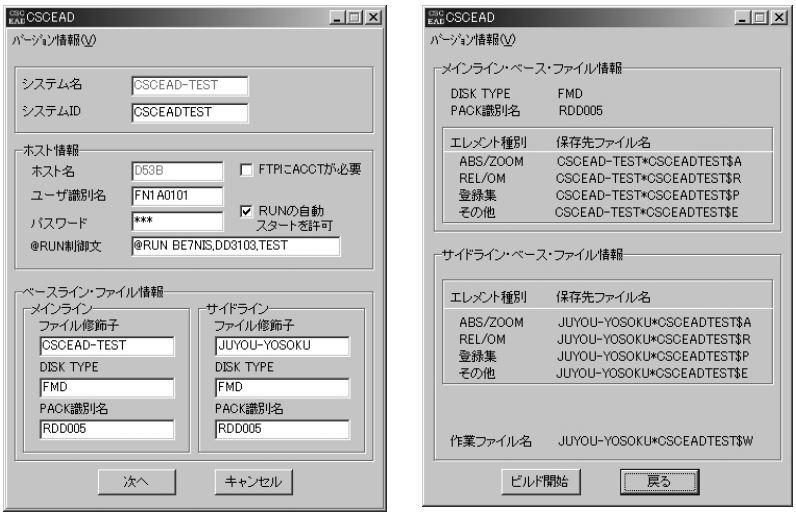

図 9　一括ビルドの環境設定画面

5.6 移 行 支 援

シリーズ 2200 上に存在するソース・プログラム，JCL 等の既存資産を CSCEAD に移行するための支援機能を提供する．CSCEAD では，構成管理サーバの成果物管理の単位はプロジェクトである（4.2 節参照）．COBOL ソース・プログラム，および登録集を解析し，呼び出し関係や登録集の使用をビジュアルに表示するとともに，プロジェクトの自動生成まで行うことができる（図 10）．また，移行作業中にも開発・保守作業が継続できるよう，移行完了まで旧開発環境での開発・保守を可能とし，移行最終段階で旧開発環境での変更分を CSCEAD に反映できるよう工夫されている．

特に開発時のメンバーが居らず，構成管理が不充分なケースからの移行では，現状のプログラムを解析する作業負荷は大きい．本機能を使うことにより現状把握が容易になり，さらに充分な構成管理機能を使える CSCEAD に移行できるため，将来にかかる保守コストの観点からもメリットは大きい．

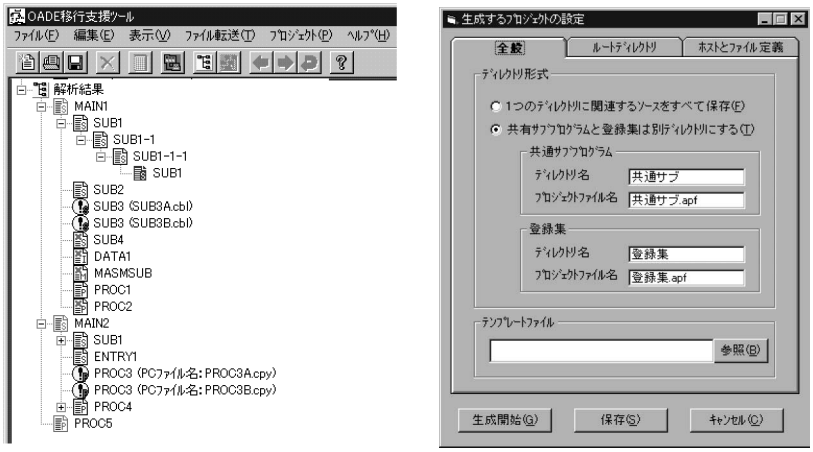

図 10　COBOL ソースコード解析，プロジェクト自動生成画面

6. お わ り に

CSCEAD の開発では，今後の開発環境がどうあるべきか理想を追求し，要件定義してきた．単なる，IDES の後継モダナイズ版という位置付けではない．IDES で資産を管理している場合や，独自の方法で管理している場合にも，既存プログラム資産を CSCEAD 開発・保守環境に継承できるよう工夫しているので，2200/IX 上に既存のプログラム資産をもつ全ユーザに受け容れられるものと考えている．

大規模システムのライフサイクルを見ると，開発に加えて長期間の保守作業が発生するのが常である．CSCEAD 導入による費用対効果を試算したところ，CSCEAD は長期に渡り保守が発生する大規模システムで，特に効果が大きいことがわかっている．

また，将来オープン環境への移行を一つのオプションとして想定している場合にも，ビルドサーバが換わるだけで，基本的な管理方法を変えなくて済むため，移行負荷が少ないというメリットを受けることができる．

参考文献
［1］ Open Application Development Environment(OADE) for Windows-User Guide (78483989)
［2］ Object Oriented COBOL Compiling System Functional Specification（＃20008091）
［3］ 統合開発環境支援システム IDES 解説書プログラム管理運用編（431100361-1）
［4］ 統合開発環境支援システム　IDES 解説書単体テストモジュール系･網羅率測定編（431101000-0）
［5］ 橋本　篤，チーム開発支援モデルと TeamFactory，UNISYS 技報　第 68 号，2001, pp.106～123

執筆者紹介　元　山　裕　之（Hiroyuki Motoyama）
1978 年横浜国立大学工学部卒業．同年日本ユニシス（株）入社．主として，UNISYS シリーズ 2200 の言語プロセッサの開発，保守業務に従事．現在，社公システム一部社公ソリューション室に所属．SC 22/COBOL WG 委員．