ソフトウェア工学　94－1
（1993．9．21）

ＣＡＳＥ統合化技術を用いたソフトウェア開発・管理方法

藤井　諭　　中村　智法　　旭岡　裕子　　大西　伸幸

松下通信工業（株）
〒226横浜市緑区佐江戸町６００番地

あらまし　　ソフトウェアの品質確保と生産性向上を目的として、ＣＡＳＥの開発と導入が活発に行われている。ＣＡＳＥの適用にあたっては、課題に対して最適なツールを選択し、かつユーザの使えるレベルにカスタマイズすることが重要である。また、開発プロセスに合わせて複数のツールを柔軟に組み合わせ出来る必要がある。

この理由から、国際標準化動向に基づくＣＡＳＥ統合化技術を開発し、ツールのカスタマイズと柔軟な選択・組み合わせを可能とした。このＣＡＳＥ統合化技術をベースとして、上流工程、下流工程、開発管理の各工程において、ＣＡＳＥ適用によるソフトウェア開発・管理方法を開発した。また３つの工程での事例を通して、開発現場でＣＡＳＥ適用の効果を得るためのポイントを考察した。

和文キーワード　　ＣＡＳＥ、統合化、エンカプシュレーション、ソフトウェア開発、ソフトウェア管理

Software Development and Management Method using CASE Integration Technology

Satoru FUJII　Tomonori NAKAMURA　Yuko HINOOKA　Nobuyuki Onishi

Matsushita Communication Industrial Co.,Ltd.
600,Saedo－cho,Midori－ku,Yokohama,226 Japan

Abstract　　To obtain quality and improve productivity of software, we are avtively developing and introducing CASE. We have to select certainly best tools for the problem and customize for the users. And we must be able to combine some kind of tools flexibly to development processies.

By this reason, we developed CASE integration technology based on international CASE standardization activities, and we could customize and combine many kind of tools. Using this CASE integration technology, we developed software development and management mehtods for upper development process, lower development process and development management process. With these subjects in three processes, we considered the points to get applied effects of CASE at software development sections.

英文key words　　CASE, Integration, Encapsulation, Software development, Software Management

1．はじめに

製品に占めるソフトウェアの位置づけが大きくなり、その品質確保と生産性向上が重要さを増している。これを目的としてのＣＡＳＥの開発と導入が活発に行われている。しかし期待ほどには顕著な効果はまだ出ているとは言えず、ＣＡＳＥの適用による良い事例の作成が現状での課題となっている。我々は今までの経験から、ＣＡＳＥの使い方は一律なものではなく、使う部署や工程に合わせてカスタマイズできないと、期待する効果は得られないと考えている。

現在この背景のもと、ソフトウェア開発支援システムＳＤＳＳ(Software Development Support System)[1] の成果を発展させた、ＣＡＳＥ統合化技術の構築に取り組んでいる。まず、開発環境の国際標準化案ＰＣＴＥ[2][3] をベースとし、使う工程に適したＣＡＳＥ統合化技術の開発を推進している。さらにこのＣＡＳＥ統合化技術をベースとして、プロジェクトに最適なソフトウェア開発・管理環境を作成し評価する取り組みを行っている。これらの取り組みを通じて、システム商品開発における品質確保と生産性向上を推進している。

本報告では、ソフトウェア開発の上流工程、下流工程、開発管理の各工程において、ＣＡＳＥ統合化技術をベースとするソフトウェア開発・管理方法を開発した結果について述べる。また工程毎の３つの事例を通して、開発現場でＣＡＳＥ適用の効果を得るためのポイントについて考察する。

2．ＣＡＳＥ統合化の考え方

2．1　ＳＤＳＳの位置付け

我々は既に、マイコンシステム分野を対象としたソフトウェア開発支援システムＳＤＳＳを開発し、開発現場への適用を行ってきた。ＳＤＳＳはソフトウェア開発の基本設計からテストまでの広い工程にわたって支援機能を持っている。ホストをＵＮＩＸワークステーションとし、端末をパソコンとしてマンマシンインタフェースはＷＩＮＤＯＷＳで構成される。

機能としてはまず、ソフトウェア開発の社内作業標準に基づくドキュメントの標準様式を設定することができる。作成されたドキュメントの中の必要な情報を、他工程のドキュメントにコピーする自動伝達機構も持っている。ドキュメントはホストマシンで集中管理し、複数開発者によって同時に参照して分担作成するといった、並行開発が可能である。主に以下の機能から構成されている。

(1)文書編集
　ＷＩＮＤＯＷＳ上でテキスト、表、図を組み合わせドキュメント編集する機能
(2)ＨＣＰチャート編集
　ＷＩＮＤＯＷＳ上でＨＣＰチャートを編集する機能
(3)Ｃソースコード生成
　上記ＨＣＰチャートからＣソースコードを生成する機能
(4)ＨＣＰチャート生成
　ＣソースコードからＨＣＰチャートを自動生成する機能
(5)テスト支援
　テストケース設定、スタブ／ドライバ生成、カバレージ集計、テスト成績集計の機能の組み合わせ
(6)プログラム構造設計書生成（ＰＳ書生成）
　Ｃソースコードのヘッダ情報からプログラム構造設計書を生成する機能

しかしこれらの機能ですべての課題をカバーできるわけではなく、ユーザ固有のツールや市販ツールと効率良く組み合わせて行く必要がある。ＣＡＳＥ統合化技術は、ＳＤＳＳで開発した成果を新しい技術の流れ[1]に適合させ発展させることをねらいとする。上記の機能はアプリケーション・ツールのひとつと位置づけ、ＣＡＳＥ統合化の中で市販ツール等と組み合わせることによってより効果的な活用をはかる。

2．2　ＣＡＳＥ統合化のモデル

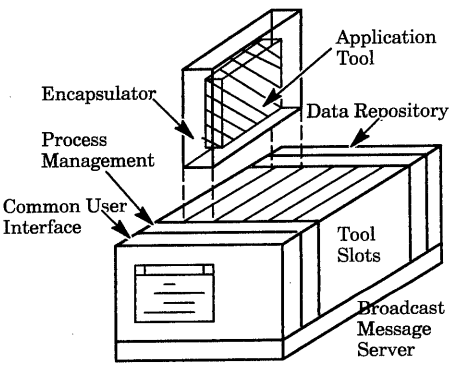


図１　CASE統合化のモデル（トースターモデル）

ＰＣＴＥで提唱するＣＡＳＥ統合化のモデルは制御統合、プレゼンテーション統合、データ統合の３軸の統合から構成される。実体としては図１に示すトースターモデルと呼ばれるものである。[2]

我々はこのモデルに基づく制御統合とプレゼンテーション統合を具現化したＨＰ社のＳｏｆｔＢｅｎｃｈを、ＣＡＳＥプラットフォームとして採用している。[4][5]

アプリケーションツールはEncapsulatorでカプセル化し、Tool slotsに差し込んで使用する。ツールスロットにアプリケーションツールを差し込むための作業を、エンカプシュレーション（カプセル化）とよぶ。エンカプシュレーションのプログラミングによって、ウインドウ、メニュー、ボタン等の部品の実装を少ない工数で実現できる。

ツール同士はＢＭＳ(Broadcast Message Server)を介し、メッセージ通信によって連動させることができる。このＢＭＳを用いてツール間コミュニケーションを実現でき、開発作業の自動化をはかることができる。またCommon user interfaceによって、マンマシンインタフェースを柔軟に開発することができる。

２．３　ＣＡＳＥ統合化の方法

ＣＡＳＥ統合化はツールのカプセル化と、ツール間コミュニケーションによって実現することができる。

まずツールのカプセル化の方法を、ＳＤＳＳ機能の一つであるＨＣＰプリントマネージャを例に述べる。図２に示すＨＣＰプリントマネージャを実現する手順は、次の通りである。

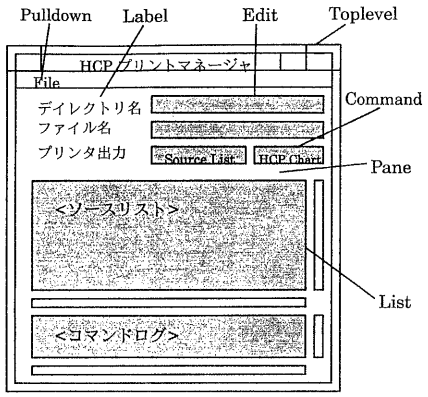


図２　エンカプシュレーションの例

(1)対象ツールをＵＮＩＸコマンド化する
(2)プリミティブ・オブジェクト（実体）を貼り付ける土台となる、次の４種類のマネージャ・オブジェクトを選択定義する
[Toplevel] 基本部でありこの上にTransient、Pane、Pulldown等のマネージャを定義
[Transient] ユーザ定義ダイヤログボックス用
[Pane] プリミティブ・オブジェクトの貼付け用
[Pulldown] メニューの貼付け用であり、ネストすることでサブメニューを構成
(3)プリミティブ・オブジェクトを定義する
Label,Command,Toggle,List,Editなど
(4)ＵＮＩＸコマンドの呼出しを定義する

次に、ツール間コミュニケーションの手順を述べる。図３に示すように、ＳＤＳＳテスト支援ツールからメッセージを送信し、ＢＭＳを介してＨＣＰプリントマネージャまたはSoftBenchエディタを起動させる方法を例に示す。ＳＤＳＳテスト支援ツールでモジュールテストを行っている途中で、ＨＣＰチャートプリンタによるプログラム構造図の出力を起動可能とし、またソースコードを修正したければSoftBenchエディタを起動する事を可能とする。

実現方法は次のようになる。
(1)ツールクラスへＨＣＰプリントマネージャ、テスト支援ツールをツール登録
(2)テスト支援ツールにＨＣＰプリントマネージャ起動用のメニュー、メニュー項目を追加
(3)テスト支援ツールにＨＣＰプリントマネージャ、SoftBenchエディタ起動用のイベント、関数を作成
(4)テスト支援ツールにＨＣＰプリントマネージャを起動するためのメッセージ送信処理を追加
(5)ＨＣＰプリントマネージャにテスト支援ツールから送信されるメッセージに対応するイベントを追加

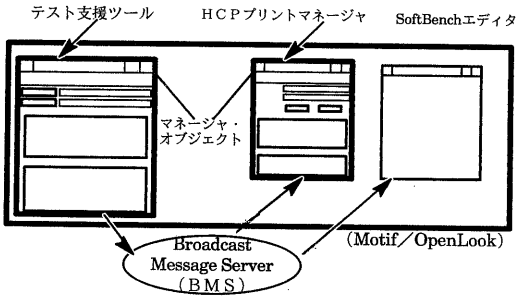


図３　ツール間コミュニケーションの例

ＢＭＳを介してツール間でやりとりするメッセージの、主な構成要素は次のものである。
[Message－Type] 実行要求(Request)、正常処理(Notify)、異常処理(Failure)のメッセージ種別指定

［Tool−Class］基本メッセージセットであり、EDITクラスの場合softedit、emacs、viなど任意に設定可能

［Command］他のツールへの起動メッセージを定義

この例で実現したＳＤＳＳテスト支援ツールとＨＣＰプリントマネージャ、SoftBenchエディタとの間の、ＢＭＳメッセージ内容を表１に、ツール間コミュニケーションのメッセージ・マトリクスを表２に示す。ＢＭＳへのメッセージの中で、ＳＤＳＳテスト支援ツールからＨＣＰプリントマネージャに対して"ＰＲＩＮＴ"というコマンドを、SoftBenchエディタに対して"WINDOW"というコマンドを定義し、実現している。

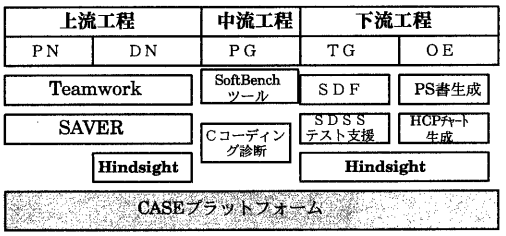
表１　ツール間メッセージ

受信ツール / BMSメッセージ	ＨＣＰプリントマネージャ	SoftBenchエディタ
Message−Type	Notify	Request
Tool−class	NULL	EDIT
Command	PRINT	WINDOW

表２　メッセージ・マトリクス

受信側 / 送信側	ＳＤＳＳテスト支援ツール	ＨＣＰプリントマネージャ	SoftBenchエディタ
ＳＤＳＳテスト支援ツール	−−−	PRINT	WINDOW
ＨＣＰプリントマネージャ		−−−	
SoftBenchエディタ			−−−

２．４　ＣＡＳＥ統合化のツール構成例



PN～OE　：SDEM90の大工程名

図４　CASE統合化によるツール構成例

ＣＡＳＥ統合化の全体のツール構成例を図４に示す。ＣＡＳＥプラットフォーム上にアプリケション・ツール群を配置する。上流、中流、下流の各工程に対し実績のあるアプリケション・ツールをユーザ要望、独自開発、市販品にかかわらず混在させて配置する。図はその一例であり、適用プロジェクトを分析し、改善目的に有効なツールを選択する。選択したツール群を、ツールのカプセル化とツール間コミュニケーションによってまとめ、ＣＡＳＥ統合化を実現していく。

３．上流工程における取り組み

３．１　背景

上流工程への取り組みでは、主に仕様書の品質と再利用性の向上のための改善が期待されている。従来の方法はシステムの要求仕様が文章主体であり、あいまいで全体が見えにくい。またシステムの仕様と作りとの対応がとりにくいため、細部での漏れや矛盾に対するチェックがしにくい。その結果仕様書、設計書の再利用性に乏しくなり、次回も仕様書から作り直す場合が多い。

この課題の改善のために、ＣＡＳＥの適用による開発方法論の導入を推進している。開発方法論としてはリアルタイムＳＡ／ＳＤ手法を採用した。リアルタイムＳＡ／ＳＤ手法を採用した理由は、方法論として確立して世の中の使用実績も多く、ＣＡＳＥツールが豊富にあることである。また我々の開発対象がほとんどリアルタイム・システムであるため、本手法を選択した。

今回は交通システム関連の、比較的規模の小さいシステムの開発に適用した。

３．２　取り組み内容

リアルタイムＳＡ／ＳＤ手法がスムーズに適用できるよう、従来の設計項目との整合性を考慮したドキュメント体系を考案した。表３に設計工程のドキュメント体系を示す。

ＣＡＳＥには市販のリアルタイムＳＡ／ＳＤツール(SAVER)を用いた。表３のドキュメント体系はこのＣＡＳＥのみではカバーできないため、他のドキュメントツールとの併用が不可欠となる。基本仕様書はユーザにもわかるよう、従来通りのテキスト形式とした。メインとなる機能仕様書は、すべてＣＡＳＥで作成した。しかし視覚的表現を必要とする画面仕様、帳票仕様、物理的要件の多いファイル仕様およびプロトコルを記述する通信制御仕様はＣＡＳＥでは書きずらいため、従来のドキュメント形式を併用した。プログラム構造仕様書は機能仕様書から作るため、ほとんどをＣＡＳＥで記述することとした。

これによって、従来の開発手順との整合のとれる手法導入を可能とした。コントロールフローダイアグラム（ＣＦＤ）、データフローダイアグラム（ＤＦＤ）、データディクショナリ（ＤＤ）、ミニスペック（ＭＳ）の一貫したつながりにより、仕様漏れのない記述ができるようにした。マルチタスク処理に対してはＤＦＤとＣＦＤを併用して表現し、最上位階層のレベルでタスク構造が判るように記述した。ＭＳは論理的に記述することで統一し、プログラム構造設計工程でのモジュール分割のしやすさを考慮した機能定義を行った。

表３　設計工程のドキュメント体系

工程	名称	記述内容	記述形式
システム設計	基本仕様書	システム概要、基本仕様、システム構成、システム規格、使用条件	従来通り
機能設計	機能仕様書	ソフトウェアの機能構成	ＤＦＤ、ＣＦＤ
		システムの入出力仕様	ＤＦＤ、ＤＤ ＣＦＤ
		システムの動作手順	ＳＴＤ
		機能の詳細、エラー処理方法	ＭＳ
	画面仕様書	画面一覧、画面遷移	従来通り
	帳票仕様書	帳票イメージ	従来通り
	通信制御仕様書	通信手順、データ形式	従来通り
	ファイル仕様書	ファイルの構成と内容、コード種別、データ長	従来通り
プログラム構造設計	プログラム構造仕様書	タスクの名称、機能	従来通り
		タスク関連図、タスク制御、タスク処理方法	ＤＦＤ、ＤＤ、ＣＦＤ、ＭＳ
		モジュール関連／構成、呼出し手順	ＳＣ
		モジュールの機能	ＭＤＳ

注）ＤＦＤ：データフロダイアグラム　ＣＦＤ：コントロールフローダイアグラム　ＤＤ：データディクショナリ　ＳＴＤ：状態遷移図
ＭＳ：ミニスペック　ＳＣ：ストラクチャードチャート
ＭＤＳ：モジュールスペック

３．３　評価結果

当初ＣＦＤ、ＤＦＤ、ＤＤと言った方法論独特の記述法は、始めての部署にはわかりにくいと言った課題があった。そこで、担当者全員が共通に認識できるための導入教育を実施して対応した。導入時のオーバーヘッドを理解でき、導入意欲のあるプロジェクトであったことが、スムーズな導入の要因となった。方法論は慣れが大切であり、実際に使って行く中でメンバーへの理解を深めることができた。

評価のポイントとしては次の３点に着目している。

1)ドキュメントの判りやすさ（仕様と作りの対応）

2)ドキュメントのバグ（漏れ、矛盾、あいまいさ）

3)ドキュメントの再利用性

現在は機能設計がほぼ終了した段階であり、1)と2)で機能設計工程に期待した効果は出ている。ＣＡＳＥの持つ同時参照機能、相互検証機能、検索機能を用いることによる、初期段階での誤りや漏れの防止、レビューの容易化といった改善効果は日常的にあらわれている。またＭＳの使い方によりシステムの仕様と作りとの対応が取りやすくなった。

一方で、適用部署ではリアルタイムＳＡ／ＳＤ手法導入がはじめてであるこから、機能設計での工数は明らかに増加した。特に、システム設計時にＭＳにより処理内容の詳細化を行うのに時間がかかった。しかし、機能設計時に処理手順を論理的に抑えることができるため、次工程（プログラム構造設計）での時間はかなり短縮が見込まれる。[8]
3)の評価はまだできないが、システムの仕様と作りとの対応が取りやすくなったことで、次回の再利用範囲が増えると予想される。

３．４　上流工程への今後の取り組み

本取り組みでは、リアルタイムＳＡ／ＳＤツール以外のツールとの連携は特に行ってない。次のステップとしては、下流工程や開発管理のためのツールとのツール間コミュニケーションによるＣＡＳＥ統合化が考えられる。

世の中でオブジェクト指向手法の実用化の試みが活発な中で、今回はあえてリアルタイムＳＡ／ＳＤ手法を採用した。オブジェクト指向手法はまだ方法論としては発展途上にあり、ＣＡＳＥツールの完成度も低い。例えばＯＭＴ手法[9]　を用いる場合、オブジェクトモデル図、状態図、事象トレース図、データフロー図などをサポートするＣＡＳＥツールとして実用に耐えるものはまだない。設計手法の適用にＣＡＳＥツールは不可欠なものと考えている。

また、オブジェクト指向ＣＡＳＥを適用する場合、今回と同様に理解のしやすいドキュメント体系を考案して適用する必要がある。その時には、このリアルタイムＳＡ／ＳＤツールによる取り組み内容が、共通に役立つと考えている。

４．下流工程における取り組み

４．１　背景

マイコン組み込みシステムの場合、ターゲットの使用ＣＰＵに合わせた開発環境が必要となる。リアルタイム処理の場合は、リアルタイムＯＳ上でアプリケーションを動かすシミュレータも必要となる。処理時間やメモリーの制約に対しては、それを満足するソフトウエア性能の実現も必須条件となる。

したがってＣＡＳＥ環境を設計する時点で、対象システムに対しどの工程までＵＮＩＸ上で開発し、どの工程からターゲット上で開発するかを定義する必要がある。

本事例では、交通システムの一部を開発するためのＣＡＳＥ統合化環境の構築を行った。プログラミング工程を中心にホスト上のＣＡＳＥ統合化環境で開発し、モジュールテストを終えたプログラムをリアルタイムＯＳシミュレータに渡して結合テストを行い、それ以降はターゲット上に移して開発するよう分担した。

４．２　取り組み内容

図５に今回開発した、プログラミング工程をサポートするＣＡＳＥ統合化システムのツール間遷移図を示す。ツールには、SoftBenchエディタ／ビルダ、ＵＮＩＸデバッガ、コーディング規約との整合をチェックする「Ｃコーディング診断システム」(市販品)、ＳＤＳＳテスト支援ツール、ＰＳ書生成ツール、プログラム構造や複雑度の検証に使用するHindsight(市販品)を選定した。これらのツールをカプセル化とツール間コミュニケーションによって統合化し、図５の流れに構成した。

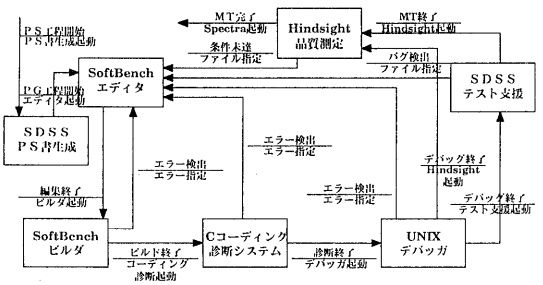


図５　CASE統合化によるツール間遷移図

この構成によると、SoftBenchエディタで作成したソースコードをビルダにかけるとＣコーディング診断システムでチェックがかかり、設定されたコーディング規約に合わない項目が指摘され、SoftBenchエディタ上で指摘箇所を参照して修正することができる。規約をクリアしたソースコードがＵＮＩＸデバッガにかけられる。デバッグを終えたソースコードは、必要に応じてＳＤＳＳテスト支援やHindsightにかけ、品質評価レポートが出力できる。このように定形的な作業手順はツール連動で自動化しながら、必要なツールを開発者が選択して使用できる構成としている。

４．３　評価結果

従来は、開発するシステムに合わせた開発環境の構築にはツール開発に多大の労力を必要としていた。ＣＡＳＥ統合化技術により独自ツールと市販ツールを開発作業の流れに合わせて柔軟に組み合わせる事ができるようになり、ツール開発の工数は大幅に削減できた。これによりＣＡＳＥへの投資に対するコストパフォーマンスを向上することができた。

改善効果としては次のものがある。

(1)ＣＡＳＥへのコストパフォーマンスの向上
(2)適用部署に適した開発プロセスを柔軟に構築
(3)ツールの連動によるデータの流用、手続きの簡略化、作業漏れの防止
(4)ＧＵＩの統一化による使いやすさの向上

今後の課題としては次のものがある。

(1)実運用評価による改良
(2)クロス開発環境との整合による適用工程の拡張

５．開発管理における取り組み

５．１　背景

小容量交換機ソフトウェアのソースファイルのアクセス管理に対し、ＣＡＳＥの適用を行った。このソフトウェアの場合、モデルチェンジが繰り返されるため、８割が改造・流用、２割が新規開発といった割合になることが多い。したがって、既存のファイル、プログラム構成を再利用した修正作業が開発のメインとなる。

今回の適用部署においては、下記の取り組みの強化が求められていた。

(1)修正の反映漏れによるアクシデントの防止の仕組み
　複数開発者が異なる時間に同一ファイルにアクセスし修正する場合の反映漏れの防止
(2)共通エリアに最新のモジュールを集める仕組み
　システムテスト用の共通エリアに最新モジュールを漏れなくそろえる仕組み

５．２　取り組み内容

上記の課題を解決できるＣＡＳＥツールとして、適用部署で既存の市販の構成管理ツール（ＳＤＦ）を用いることとした。これによる、ファイル・アクセスコントロールの実現を以下のように行った。

図６に示すように、プロジェクトアカウント（project）の下に、資産領域（複数開発者が開発したソフトウェアを集結）、プロジェクトワークスペース（共通エリアであり、最新版を登録しテスト時のmakeで参照）、個人ワークスペース（開発担当者毎の作業エリアで資産領域からチャージイン／チャージアウト）を設定する。

資産領域へのプロジェクトの登録は本来プロジェクトリーダが行うが、その負担を軽減するために、シェルスクリプト化したコマンドによってシステム管理者ができるようにした。忙しいプロジェクトリーダや開発者に負担にならずに使えるよう、提供するコマンドは実際に必要となるものだけに絞り込んで実現した。

システム管理者、プロジェクトリーダ、開発者の役割分担は表４のように設定し、使用法をマニュアル化した。

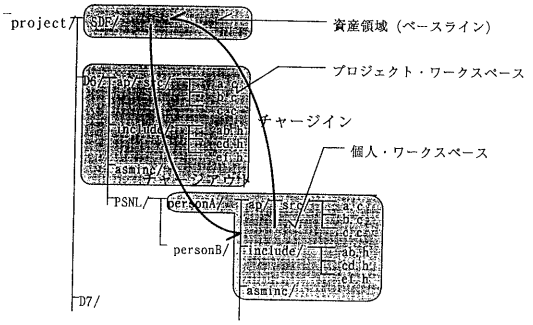


図６　ファイルアクセスコントロールの仕組み

表４　アクセスコントロールの役割分担

役　割	担　当　作　業
システム管理者	資産領域の設定、プロジェクトワークスペースの設定、プロジェクトメンバーの登録
プロジェクトリーダ	プロジェクトワークスペースの共通ファイルの管理、ファイルの凍結とシステムテストの指示、バージョンのＦＩＸと保管
開発者	ファイルの持ち出し、ファイルの参照、ファイル間差分チェック、ファイル登録

５．３　評価結果

試作したシステムの長所としては、

(1)従来の開発環境／運用形態に大きな変更を強いることなく、ＣＡＳＥの使用が可能なった。

(2)コマンドの絞り込みとシェルスクリプト化によりユーザの負担を減らして実現した。

(3)共通エリアの状況把握、ファイル持ち出し、ファイルを持ち出す際の排他制御が、容易に行えるようになった。

(4)次期商品の開発管理への適用のメドをつけることができた。

今回は適用部署の事情もあり、ファイルアクセスコントロールのみに絞って実現した。したがってＣＡＳＥ活用としては十分なものではない。次のステップとして以下の展開をして行けば、ＣＡＳＥによる開発管理の効果をより上げることができると考える。

(1)ＳＤＦのmake機能のカスタマイズによるアセンブラへの拡張

(2)ＣＡＳＥ統合化による品質管理ツールとの連携

６．おわりに

まず、ＣＡＳＥ統合化技術の考え方と実現方法について詳細に述べた。そして、ＣＡＳＥ統合化技術をベースとして、上流工程、下流工程、開発管理の工程に分けてＣＡＳＥ適用によるソフトウェア開発・管理方法を開発した結果を述べた。

上流工程では、仕様の明確化、仕様上の漏れや矛盾の防止を目的としてＣＡＳＥの適用を行った。この工程ではＣＡＳＥとしてリアルタイムＳＡ／ＳＤツールを適用したが、設計手法の活用を主体に、ツールは手法を使い易くするための補助手段として用いた。ユーザや管理者から見た仕様の判りやすさを重視し、既存ドキュメントとの組み合わせによる新しいドキュメント体系を考案し試行した。漏れや矛盾の防止の点では手法の特長とＣＡＳＥの検証機能を有効活用した。その結果、仕様のあいまいさは取り除かれ、またプログラム構造設計のしやすい機能設計を実現できた。わかりやすいドキュメント体系を考案し、手法とＣＡＳＥの特長を生かすことがポイントであった。

下流工程への取り組みでは、開発ターゲットに合った開発環境にすることが重要となる。そのためには開発ターゲットに有効なＣＡＳＥツールを各作業内容毎に選択して組み合わせ、作業の流れに合わせてツール間コミュニケーションをとることがポイントである。本事例においては、７種のツールのカプセル化とツール間コミュニケーションを開発することによって適用部署に適した開発プロセスを実現

した。コーディング規約に基づく品質の確保、定形作業の簡略化と作業の漏れの防止、およびツールの操作性の向上に有効である。

開発管理への適用事例では、ファイルアクセスコントロールの実現を行った。ＣＡＳＥ適用によってプロジェクトリーダや開発者に負担が生じないよう、コマンドの絞り込みとシェルスクリプト化によって最小限のコマンドで実現した。また、それぞれの立場からの役割分担を明確化し、使用するコマンドを分類してマニュアル化した。ここでのポイントは、初めてのユーザに違和感なく簡単に使えるよう使用法をカスタマイズすることであった。

３つの適用事例で示したように、対象とする工程によってまたシステムやユーザによって、ＣＡＳＥの適用内容も適用のポイントも異なっている。ＣＡＳＥ適用の効果を上げるには画一的な取り組み方ではうまくいかない。適用プロジェクトにおいて課題が何かを見極めた上で、ＣＡＳＥによる開発環境を設計する必要がある。従って効果を得るためには、相応の時間と労力を必要とする。

またＣＡＳＥによる開発環境の設計にあたっては、今後の発展が予想されるＣＡＳＥ統合化の流れに適合できるツールか否かを、選択の基準として考慮すべきである。

謝辞

ＣＡＳＥ適用にあたり指導、助言をいただいた当社情報システム事業部の川里隆主任技師、伊藤伸一氏、渡辺敏広氏、および当社通信システム事業部の小野悦子氏をはじめとする関係者に感謝します。

参考文献

[1]藤井他：ソフトウェア開発支援システムＳＤＳＳ、情報処理学会「ＣＡＳＥシンポジウム」(1989)

[2]ECMA:ReferenceModelforFrameworksofSoftwareEngineeringEnvironments, NIST(Dec1991)

[3]鯵坂：解放型ＣＡＳＥプラットフォーム、コンピュタソフトウェアVol.10,No.2、日本ソフトウェア科学会(Mar1993)

[4]藤井他：ＣＡＳＥ統合化技術を用いたソフトウェア開発・管理支援方法、情報処理学会第４６回全国大会(1993)

[5]BrianD.Fromme：HPEncapsulator:BridgingtheGenerationGap, HEWLETT-PACKARD JOURNAL(June 1990)

[6]PaulT.WardandStephenJ.Mellor：StructuredDevelopmentforReal-TimeSystems、PrenticeHall/YourdonPress(1985)

[7]DerekJ.Hatley,ImtiazA.Pirbihai(立田監訳)：リアルタイム・システムの構造化分析、日経ＢＰ社(1989)

[8]藤井他：ソフトウェア開発支援システムＳＤＳＳへの設計支援導入の検討、電子情報通信学会春期大会(1992)

[9]J.Rumbaugh他(羽入田監訳)：オブジェクト指向方法論ＯＭＴ、トッパン(1992.4)