ソフトウェア開発支援システム SDSS における CASE 統合化

藤井 諭[†,*] 千葉雅弘[†]
高柳雄一[†] 中村智法[†]

マイコン組み込みシステムのソフトウェア開発において，開発の作業標準は定められていたが，成果物と直接つながらないため，普及はあまり進まない状況にあった．特に開発ドキュメントの作成は，従来から手作業が主体で，プログラミングやテストの作業とは遊離していた．設計工程からテスト工程までの一貫支援によって，開発作業標準の普及とドキュメント作成の効率化をはかることをねらいとして，ソフトウェア開発支援システム SDSS (Software Development Support System) の開発を行った．5年間にわたり，SDSS の現場への適用評価と，改良開発を繰り返した．その結果，延べ300ユーザ数，ピーク時で83プロジェクト，100ユーザ数の使用実績を得た．さらに NIST/ECMA が提唱する CASE 標準化案に基づき，SDSS と市販ツールの融合連携による，CASE 統合化システムを開発した．その結果，適用プロジェクトに固有のツールと SDSS ツールの組み合わせを，少ない工数で実現できた．また CASE 統合化技術の中で，SDSS をアプリケーション・ツールとして組み込み可能とし，将来への拡張性を確保した．

CASE Integration in Software Development Support System SDSS

Satoru Fujii,[†,*] Masahiro Chiba,[†]
Yuichi Takayanagi[†] and Tomonori Nakamura[†]

Standards of software development works were already decided for microcomputer systems, but these were used little because these did not linked with development products directly. Especially, software documents were almost handmade sets and separated to programming and testing works. We tried to develop SDSS (Software Development Support System) supporting consistently from design process to testing process, to improve propagation of standards of development works and effectiveness of document production. We repeated to estimate SDSS toward many projects and to improve development in 5 years. As a result, we obtained 300 users totally, 83 projects and 100 users within a certain period. Still more, depend on CASE standards proposed by NIST/ECMA, we developed CASE integration system by combining SDSS and marketing tools. As a result, we gave effect to combine tools suitable for users in a short time. We completed SDSS combinable with another tools in CASE integration technology, and kept enlargement for future.

1. はじめに

ソフトウェア開発支援システム SDSS (Software Development Support System) は，マイコン組み込みシステムの開発を中心に，開発作業標準 SDEM (System Development Engineering Methodology) の普及とドキュメント作成の効率化をはかることによって，ソフトウェア開発の生産性，品質の向上を目指したものである．その背景には，ソフトウェア開発量の増大に伴い，ソフトウェアの生産性，品質の改善が急務とされる状況において，開発は従来から手作業が主体で，改善活動が思うように進んでいなかったことが上げられる．

1986年より社内の重点テーマの一つとして SDSS の企画，立案を開始した．1988年に最初のバージョンを開発し，社内のソフトウェア開発現場への適用を開始した．以来，ユーザの声を反映させながら，5年間にわたって約8回の改良開発を繰り返してきた．また，専任の運用支援グループによって，導入教育，プ

† 松下通信工業(株)技術本部
Corporate Engineering Division, Matsushita Communication Industrial Co., Ltd.
* 現在，松江工業高等専門学校
Presently with Matsue National College of Technology

ロジェクトごとの個別サポート，ユーザからの改善要望のとりまとめ等を行ってきた．

その結果，5年間で延べユーザ数で約300名，ピーク時で83プロジェクト，100ユーザの使用実績を得た．また，開発作業標準の全社的な普及にも効果があった．

さらに，SDSS の持つ課題を CASE 統合化によって解決するため，1992年より NIST/ECMA が提唱する CASE 標準化案[1),2)]への SDSS の融合をはかり，実プロジェクトへの適用評価を行った．

本稿では，2章でソフトウェア開発支援システム SDSS の開発内容，3章で SDSS の運用評価，4章で CASE 統合化への取り組み内容，5章で CASE 統合化の適用評価について述べ，6章でまとめる．

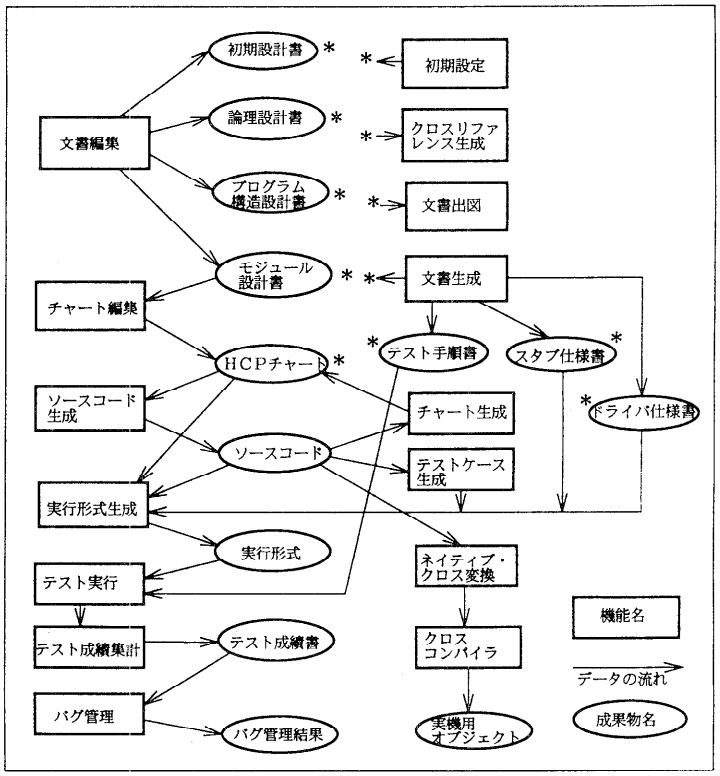


図 1　SDSS の構成
Fig. 1　Construction of SDSS.

2. ソフトウェア開発支援システム SDSS の開発

2.1 SDSS の全体構成[3)]

SDSS は，ソフトウェア開発の設計からテストまでをサポートする一貫支援システムである．

図1に SDSS の構成を，機能と成果物の関係で示す．機能を四角で，成果物を楕円で，データの流れを矢印で表す．まず「初期設定」により，仕様書，設計書の標準となる階層構成，様式，シンボルをカスタマ

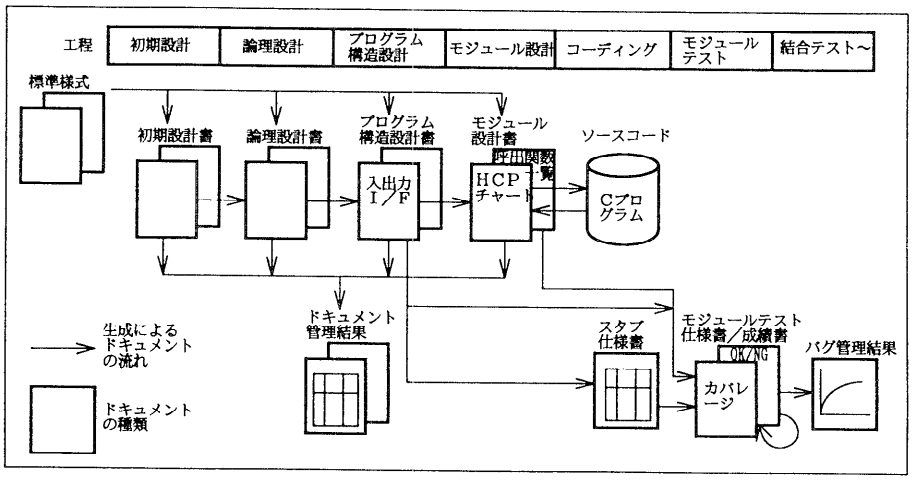


図 2　自動生成によるドキュメントの流れ
Fig. 2　Flows of automatic document production.

イズして設定する．この設定に基づき，「文書編集」で各設計工程の設計書を穴埋め式で作成できる．「チャート編集」はHCPチャート[4),5)]専用のエディタであり，これによって作成された「HCP チャート」は，「ソースコード生成」によりCソースコードに自動変換される．逆に「チャート生成」を用いると，HCP チャートを自動生成することができる．「文書生成」を用いると，設計書をもとにテスト手順書，ドライバ仕様書，スタブ仕様書を生成でき，これらを使って「実行形式生成」，「テスト実行」，「テスト成績集計」，「バグ管理」の機能によりプログラムの検証と品質測定が可能である．

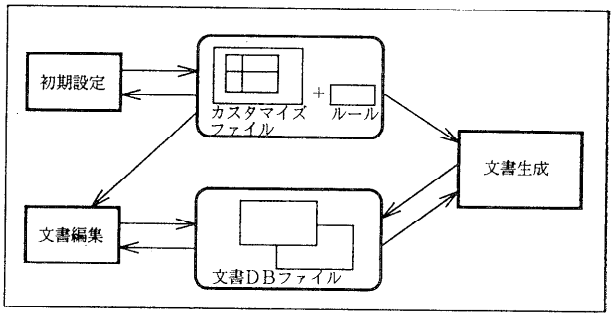


図 3 文書生成の機能関連図
Fig. 3 Relationship of functions for document production.

システム全体は，ホスト側がUNIX ワークステーション，端末がパソコンで，イーサネットで複数のホスト，端末間を結ぶネットワーク構成としている．図1に楕円形で示した「成果物」の管理は，ホストマシン上で一括して行っている．マンマシンインタフェースは，端末の MS-Windows プログラムで実現している．

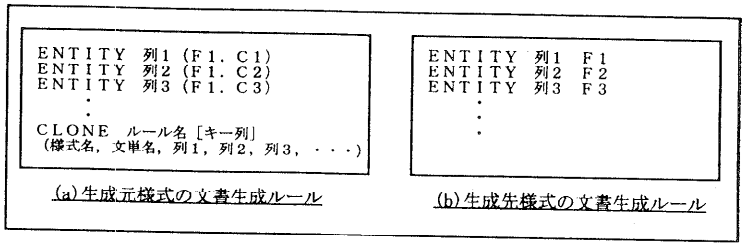


図 4 文書生成ルール
Fig. 4 Production rules of documents.

2.2 SDSS の特徴

SDSS の特徴の一つは，テキスト，表，論理図，図形の「枠情報」を単位とする，ドキュメント間の情報伝達，自動生成機構である．

図2に SDSS の持つ，自動生成によるドキュメントの流れを示す．「標準様式」から各設計書への伝達は「初期設定」を介して行う．上流設計書から下流設計書の生成は「文書生成」によって，「ドキュメント管理結果」の生成は「クロスリファレンス生成」によって，実現している．

設計書・ソースコード間の整合をとるための機能として「チャート編集」，「ソースコード生成」，「チャート生成」があり，「HCP チャート」と「ソースコード」間の相互変換の自動化を実現している．

設計書からの「モジュールテスト仕様書」，「スタブ仕様書」の生成は，「文書生成」で行う．

開発ドキュメントの情報伝達，自動生成機構を実現する三つの機能について，以下で詳細に述べる．

（1） 文書生成機能[6)]

「初期設定」，「文書編集」との関連を**図3**に示す．

まず「初期設定」により，使用するドキュメントの枠の種類，位置関係，領域を設定する．設定した様式に対して文書生成ルールを設計し，様式とともにカスタマイズファイルに登録する．

「文書生成」を実行する場合，「文書編集」を用いて生成すべき文書の内容を事前に作成しておくことが必要である．

文書生成ルールの構成を**図4**に示す．（a）は生成元様式の生成ルールである．ENTITY 宣言によって，表のエンティティ名と枠番号 (Fx)，列番号 (Cx) を定義する．CLONE 宣言の中で，生成先の様式名，文書単位名と生成元で複写の対象となるエンティティ名を定義する．（b）は生成先様式の生成ルールである．ENTITY 宣言によって生成元のエンティティ名（列x）と生成元枠番号（Fx）との対応を定義しておく．

図5は，上記のルールを用いた文書生成の実行例を示す．文書 DB ファイルの(A)項目一覧は，カスタマイズファイルの(a)項目一覧用紙から登録し，

ルールA：[(a)項目一覧用紙の「モジュール」，「No.」，「ケース名」の情報を，(c)MT 仕様用紙の「モジュール」，「No.」，「ケース名」の枠へ複写する]

を持つ．

(B)外部仕様は(b)モジュール外仕（外部仕様）用

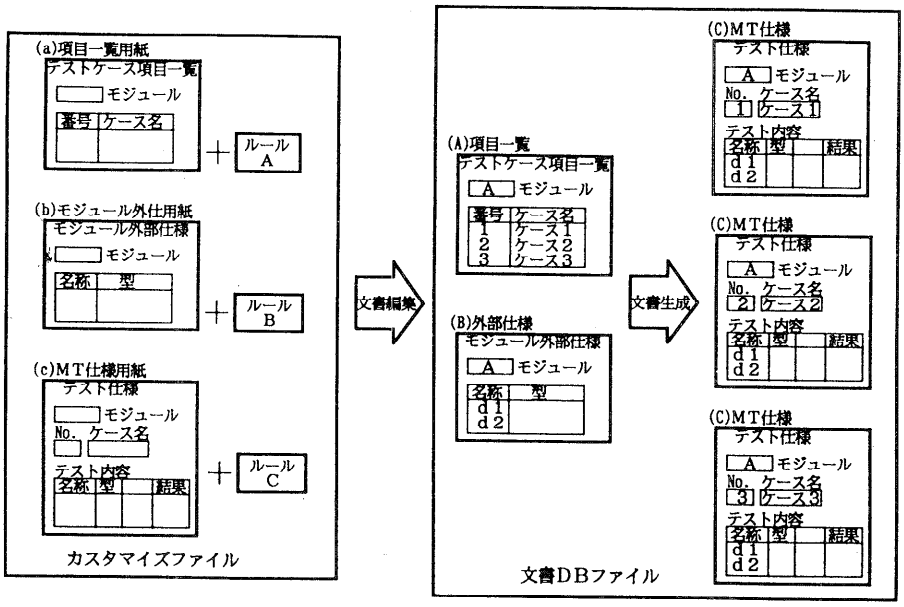


図 5　文書生成の実行例
Fig. 5　Execute example of document production.

紙から登録し，
ルールB：[(b)モジュール外仕用紙の「モジュール」，「名称」，「型」の情報を，(c)MT 仕様用紙の「モジュール」，「名称」，「型」の枠へ複写する] を持つ．

また(c)MT 仕様用紙は
ルールC：[生成元様式の生成ルールから指定されたエンティティ名に対応する枠番号] を持つ．

以上より，(A)のテストケース項目一覧を生成元としてルールAを実行し，(B)のモジュール外部仕様を生成元としてルールBを実行すると，(C)のテスト仕様を，三つのケースに分けて同時に生成することができる．

(2)　チャート関連機能[7]

チャート関連機能は「チャート編集」，「ソースコード生成」，「チャート生成」の機能で構成され，「HCP チャート」と「ソースコード」のデータを自動的に双方向変換する．これにより，プログラム設計書とソースコードの分離を防止することが可能となる．

データ「HCP チャート」は，「チャート枠」と称する枠情報としてシンボル番号，コメント情報，ソースコード，次へのポインタ等を要素とする構造体で扱う．テスト支援機能を実現するために，この構造体に「チャート編集」で，HCP チャートのボックス番号，プローブ関数の埋め込みを行う．

(3)　テスト支援機能[8]

テスト支援機能は，プログラムまたはモジュールのテスト作業を効率化するための機能群を総称したものである．テスト支援機能に相当する部分の構成の詳細を**図 6** に示す．

この中の「Cソースコード」は，「テストケース生成」で HCP チャートに変換し，プログラムの分岐や繰り返しの条件に着目したテストケースを生成する．また HCP チャートの各シンボルに対応したプローブ関数を，C言語で挿入する．このプローブ関数に対して，非分岐/分岐/繰り返しの区分と，構造化の順で割り振ったボックス番号の，二つの引き数を与える．

生成されたテストケースに対して，「テスト実行」でテストデータを与える．さらにプローブ関数の実行時に，そのボックス番号，通過経路，実行回数，通過回数などのデータを収集する．これらの結果を「テスト成績集計」で整理し，**図 7** のテスト成績書を出力する．このテスト成績書の中のテストケースの内容は，前述の自動生成したテストケースと対応し，その設定数，完了数，完了率がモジュールごとに集計される．カバレージ[9]の計算は例えば，

命令網羅率　C0＝実行ボックス数/全ボックス数
判定網羅率　C1＝通過経路数/全経路数

で行う．図7の情報以外で，未通過ボックスや未通過

経路も同時に出力できるため，不具合の検出等に活用できる．

このようにテスト支援機能は，HCP チャートを用いたホワイトボックステストでの定量測定を特徴としている．

3. SDSS の運用評価

3.1 運用結果

表1 に SDSS の使用プロジェクト数を一覧で示す．1992年6月時点で，合計で83プロジェクトに達した．使用機能としては文書編集が35プロジェクトと最多で，二位はチャート生成の31プロジェクトであった．この時の使用ユーザ数は100人であった．5年間での延べ使用ユーザ数は約300人である．

3.2 効果

（1） 開発作業標準の普及

社内での開発作業標準に合わせて，プロジェクトとしての共通フォーマットを初期設定し，チームとして共通に端末から使用することで，標準化促進に有効であった．5年間を通じて SDSS の存在が，開発作業標準の全社的普及に効果的であった．

（2） ソフトウェア・ドキュメント作成の効率化

ソフトウェアの仕様書，設計書の作成のための，初期設定，文書編集，文書生成およびクロスリファレンス生成のツール群の利用実績が高く，ソフトウェア・ドキュメント作成の効率化に有効であった．

（3） 設計書とソースコードの不一致の改善

チャート編集，ソースコード生成，チャート生成の組み合わせによる設計書とソースコードの不一致を減

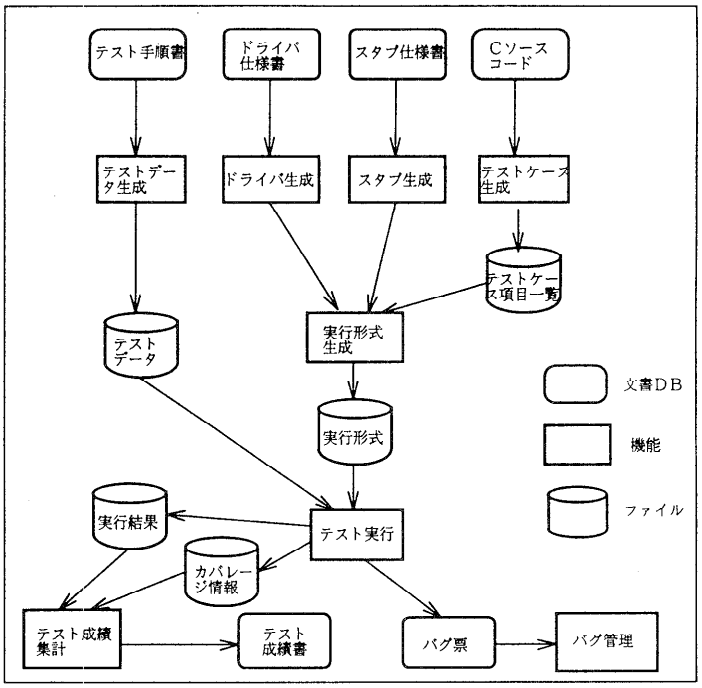


図 6 テスト支援機能の構成
Fig. 6 Construction of test support functions.

ﾌﾟﾛｸﾞﾗﾑNo.	プログラム名	ﾌﾟﾛｸﾞﾗﾑ略称	プログラム機能概要
1.1	ファイルサイズ印刷	fsize	コマンドラインで指定されたファイルのサイズを印刷する。ファイルがディレクトリであればそこにあるすべてのファイルのサイズを印刷する。

カバレージ（網羅率）

C0（命令網羅率）	C1（判定網羅率）	C2（条件網羅率）	M0（関数網羅率）	M1（呼出関係網羅率）
95%	88%	100%	100%	100%

モジュール成績一覧表

対象モジュール				テスト日付			テストケース			テスト成績				
項番	ﾓｼﾞｭｰﾙ番号	モジュール名	モジュール略称	属性	開始日	終了日	設定数	完了数	完了率	C0	C1	C2	M0	M1
1	1.1.1	ファイルサイズ印刷メインモジュール	main	関数	93/5/10	93/5/15	5	5	100%	100%	100%		100%	100%
2	1.1.2	ファイルのサイズ印刷	fsize	関数	93/5/10	93/5/15				100%	100%		100%	100%
3	1.1.3	ディレクトリ内のファイルのサイズ印刷	dirwalk	関数	93/5/10	93/5/25	1	1	100%	91%	80%	100%	100%	100%

図 7 プログラム成績書
Fig. 7 Example list of test records.

表 1 使用プロジェクト数
Table 1 Number of user projects.

プロジェクト数 / 機能	完了プロジェクト	使用中プロジェクト	使用予定プロジェクト	合計
文書編集	13	11	11	35
文書生成	3	6	3	12
チャート生成	13	8	10	31
テスト支援	0	2	3	5
合計	29	27	27	83

らす改善効果があった．

システムの納期が迫っている場合に，開発作業標準に従わず，修正はまずソースコード上で行い，テスト終了後に設計書に反映させることが多い．このような場合，チャート生成が多用され，ソースコードからの設計書の逆生成が有効であった．

3.3 システムの課題

（1） テスト支援機能

テスト支援機能の使用実績が少なかった原因は，本機能がモジュール単位のホワイトボックステストに有効であることと，納期に追われる現場ではブラックボックステストが主体となっていることとの不整合にあった．しかしバグが発生した場合はホワイトボックステストは不可欠であり，本機能をより使いやすくする必要がある．

（2） 他ツールとの連携

SDSS の環境へ，使用 CPU に合ったデバッガや品質測定などの市販ツールを個々に組み込むには，手間がかかる．特に，ツール間の連携，マンマシンインタフェースの統一に相応の改造工数を必要とする．

（3） 設計手法のサポート

構造化手法[10]，オブジェクト指向手法[11]等の設計手法を使用することができない．これらの手法をサポートする CASE ツールと，連携させる仕組みが必要となる．

（4） コストパフォーマンスの向上

ツールの機能の追加開発に要する費用は高価である．CASE ツールは対象ソフトウェア，使う人によって効果のばらつきが大きい．ツールの開発投資に見合う効果の確証を得ることは，かなり難しい問題である．

これらの課題を解決する技術として，CASE 統合化技術への高い期待があった．そこで，上記の四つの課題を解決する手段として，CASE 統合化技術への取り組みを行った．

4. CASE 統合化技術への取り組み

4.1 CASE 統合化の考え方

NIST/ECMA が提唱する CASE 統合化のモデルは図 8 に示すトースターモデルと呼ばれるもので，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．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 Soft-Bench[12]を，CASE プラットフォームとして採用した．

アプリケーションツールはカプセル化し，Tool slots に差し込んで使用する．Tool slots にアプリケーションツールを差し込むための作業を，Encapsulation と呼ぶ．Encapsulation 用のライブラリを用いたプログラミングで，ウィンドウ，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．

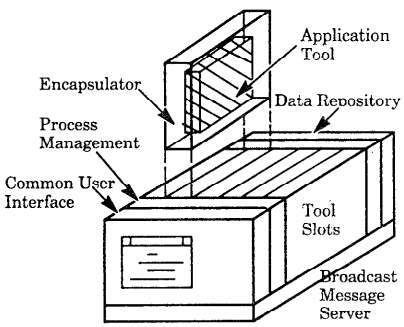


図 8 CASE 統合化のモデル（トースターモデル）
Fig. 8 Model of CASE integration (Toaster model).

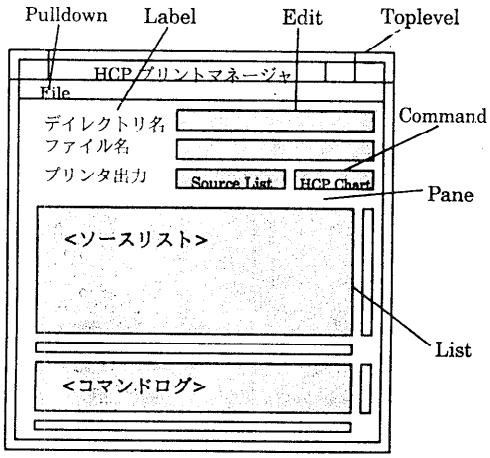


図 9 Encapsulation の例
Fig. 9 Example of encapsulation.

4.2 CASE 統合化の方法

CASE 統合化はツールの Encapsulation と，ツール間コミュニケーションによって実現できる．

Encapsulation の方法を，SDSS 機能のひとつである HCP プリントマネージャ（図1のチャート生成と文書出図をあわせたもの）を例に述べる．Encapsulation の例を**図 9** に示す．実装の手順は次のとおりである．

（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] ユーザ定義ダイアログボックス用

[Pane] プリミティブ・オブジェクトの貼り付け用

[Pulldown] メニューの貼り付け用であり，ネストすることでサブメニューを構成

（3）Label, Command, Toggle, List, Edit などのプリミティブオブジェクトを定義する．

（4）UNIX コマンドの呼び出しを定義する．

次に，ツール間コミュニケーションの手順を**図 10** の例によって述べる．SDSS のテスト支援ツール（図6の機能をまとめたもの）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示す．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 プリントマネージャによる HCP チャートの出力，または SoftBench エディタによるソースコードの修正を可能としている．

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，テスト支援ツールを起動するためのメッセージ送信処理を追加

4.3 CASE 統合化への取り組み例[13)]

CASE 統合化によるツール構成例を，**図 11** に示す．CASE プラットフォーム上にアプリケーションツール群を配置する．上流工程において，SDSS にはなかった

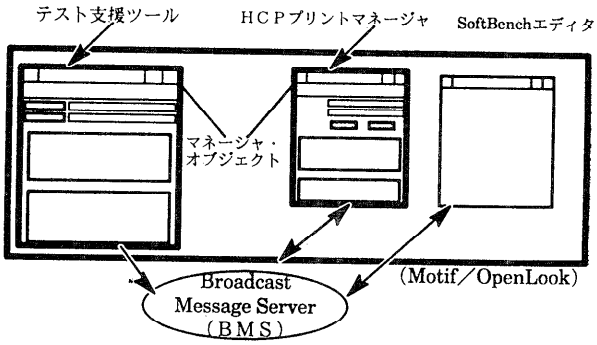
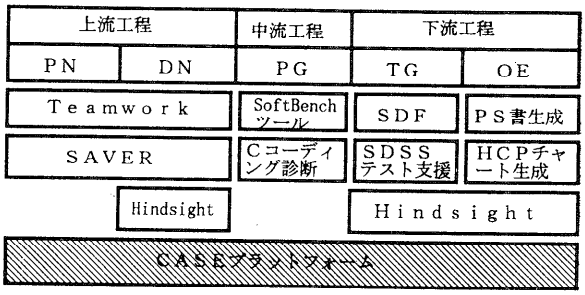


図 10 ツール間コミュニケーションの例
Fig. 10 Example of tool's communication.



PN:Planning DN:Design PG:Programming TG:testing
OE:Operation and Evaluation

図 11 CASE 統合化によるツール構成例
Fig. 11 Example of tool's communication by CASE integration.

設計手法のサポートを，市販の構造化設計ツール（Teamwork, SAVER）の組み込みにより可能としている．中流工程，下流工程においても，SDSS ツール（SDSS テスト支援，PS 書生成，HCP チャート生成）と市販ツール（SoftBench ツール，Cコーディング診断，SDF，Hindsight）との融合が可能となり，ツール開発に対するコストパフォーマンス向上に役立っている．

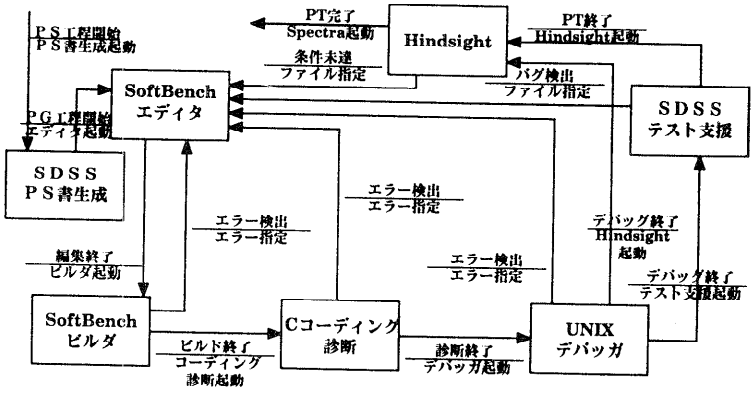


図 12 CASE 統合化によるツール間遷移図
Fig. 12 Transition diagram of tools by CASE integration.

CASE 統合化の取り組み例として，マイコン組み込みソフトウェア開発用の，CASE 統合化環境について述べる．**図 12** に，今回開発した CASE 統合化による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チェックをする「Cコーディング診断システム」，「SDSS テスト支援」，Cソースコードからプログラム構造設計書を生成する「PS 書生成」，プログラム構造や複雑度の検証に使用する「Hindsight」を選定した．これらのツールを，Encapsulation とツール間コミュニケーションによって統合化した．

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た後，Cコーディング診断システムでチェックがかかり，コーディング規約に合わない項目への指摘が行われる．指摘項目を選択クリックすると，SoftBench エディタに指摘箇所が呼び出され，効率的な修正が可能である．コーディング規約を合格したソースコードは，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，テスト支援ツールまたは Hindsight にかけることで，品質評価レポートを出力できる．

プログラムテストを完了したプログラムは，使用マイコンに適合するクロス開発環境 Spectra（市販ツール）に引き渡す．

5. CASE 統合化の適用評価

5.1 適 用 結 果

適用プロジェクトは2プロジェクトである．一つは中流〜下流工程，もう一つは上流工程を中心に CASE 統合化の適用を行った．

5.2 効　　果

（1） 市販ツールとの融合による開発コスト低減

ユーザ要望の市販ツールとの融合連携を，Encapsulation とツール間コミュニケーションのプログラミングにより，短時間で実現できた．これにより，アプリケーション・ツールを新たに開発するための，開発コストの低減に効果のあることがわかった．

（2） 設計手法との融合

設計手法をサポートするため，市販の構造化設計ツールとの融合をはかった．特に，SDSS のチャート関連機能との融合連携を行い，構造化手法での設計内容を，プログラミング工程で利用可能とした[14]．

（3） 標準化動向との整合

世界標準となりつつある CASE 統合化の流れの中で，SDSS をアプリケーション・ツールの一つとして位置づけ，将来への拡張性を確保した．

5.3 CASE 統合化の課題

（1） CASE プラットフォームを搭載するワークステーション等の設備がまだ少なく，CASE 統合化の適用できる現場プロジェクトに制限があるため，より一層の整備が必要である．

（2） 対象とするシステムの規模，CPU の種類などによって支援ツールの種類は異なり，リポジトリーの統合はツールメーカに依存してまだ難しい状況にあるため，マイコンシステム開発に適合するリポジトリー標準化の進展が望まれる．

6. おわりに

ドキュメント作成の効率化と，開発作業標準の普及をはかることをねらいとし，ソフトウェア開発支援シ

ステム SDSS を開発した．SDSS のツール群を用いることで，設計工程からテスト工程までの一貫支援を可能とした．また SDSS の，現場での運用評価に取り組み，延べ約 300 ユーザ数，最大時で 83 プロジェクト数，100 ユーザ数の使用実績を得た．

さらに NIST/ECMA が提唱する CASE 標準化案に基づき，SDSS と他ツールを融合連携した，CASE 統合化システムを開発した．これにより，ユーザに適したツールの組み合わせを少ない工数で実現できることがわかった．また SDSS を，CASE 統合化技術の中で，組み合わせの自由なアプリケーション・ツール群として位置づけ，将来への拡張性を確保することができた．

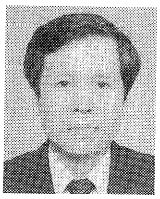
SDSS によって実現した上流から下流までの一貫支援の考え方は，国際標準化に基づく CASE 統合化技術を通じて，今後はプロセスプログラミングの実現手段として，発展させることが可能である．そのためには CASE 統合化の適用プロジェクトを広げ，適用分野，開発規模と，有効なプロセス，ツールの関連を示すデータを積み上げる必要がある．

謝辞 SDSS の開発，評価に協力いただいた，松下通信工業(株)の波里純次，森田真奈美，旭岡裕子，小又富士子，大津瑞枝，大西伸幸の各氏，および貴重な意見をいただいたユーザの方々に感謝する．

参 考 文 献

1) 鯵坂：開放型 CASE プラットフォーム，コンピュータソフトウェア，Vol. 10, No. 2 (1993).
2) ECMA : Reference Model for Frameworks of Software Engineering Environments, NIST (1991).
3) 藤井，加賀，山口，高柳，上田，平林，清岡：ソフトウェア開発支援システム SDSS，情報処理学会「CASE 環境」シンポジウム，pp. 65-72 (1989).
4) 花田：プログラム設計図法，(株)企画センター (1989).
5) 米田ほか：HCP チャートを用いたソフトウェア開発支援システム，NTT 研究実用化報告，Vol. 36, No. 11 (1987).
6) 加賀ほか：ソフトウェア開発支援システム SDSS における文書生成機能，第 37 回情報処理学会全国大会論文集，5 M-2 (1988).
7) 山口ほか：ソフトウェア開発支援システム SDSS におけるチャート編集機能，第 37 回情報処理学会全国大会論文集，5 M-3 (1988).
8) 伊藤ほか：ソフトウェア開発支援システム SDSS におけるテスト支援機能，第 38 回情報処理学会全国大会論文集，1 M-4 (1989).
9) Mycrs, G. J.：ソフトウェア・テストの技法，長尾・松尾（訳），近代科学社 (1992).
10) Ward, P. T. and Mellor, S. J.: *Structured Development for Real-Time Systems*, Prentice Hall/Yourdon Press (1985).
11) Rumbaugh, J. ほか：オブジェクト指向方法論 OMT，羽生田（監訳），トッパン (1992).
12) Fromme, B. D.: HP Encapsulator : Bridging the Generation Gap, *Hewlett-Packard Journal* (June 1990).
13) 藤井，中村，旭岡，大西：CASE 統合化技術を用いたソフトウェア開発・管理方法，情報処理学会研究報告，93-SE-94, pp. 1-8 (1993).
14) 藤井，清岡：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，SD-3-2 (1992).

(平成 6 年 4 月 7 日受付)
(平成 6 年 10 月 13 日採録)



藤井　諭（正会員）

1948 年生．1969 年松江工業高等専門学校電気工学科卒業．同年松下電器産業(株)入社．松下技研(株)にて主に音声認識の研究に従事．1987 年より松下電器東京研究所および松下通信工業(株)にて CASE の研究開発に従事．1994 年より松江工業高等専門学校助教授，現在情報工学科に所属．ソフトウェア工学の研究に従事．電子情報通信学会会員．

千葉　雅弘

1955 年生．1974 年宮城県立米谷工業高校卒業．同年松下通信工業(株)入社．ソフトウェア開発／生産技術の研究開発に従事．

高柳　雄一

1964 年生．1987 年早稲田大学理工学部応用物理学科卒業．同年松下電器産業(株)入社．1989 年より松下通信工業(株)勤務．ソフトウェア開発／生産技術の研究開発に従事．

中村　智法（正会員）

1964 年生．1987 年東京農工大学工学部電子工学科卒業．1989 年同大学院工学研究科電子工学専攻修了．同年松下通信工業(株)入社．ソフトウェア開発／生産技術の研究開発に従事．