1M-4

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

伊藤信昭[1)]、岩見彰一[1)]、田辺晴彦[1)]、中村雅之[1)]、清岡弘[2)]、藤井諭[3)]
[1)]松下システムエンジニアリング(株) [2)]松下通信工業(株) [3)]松下電器産業(株)東京研究所

１．はじめに

ソフトウェア開発支援システムＳＤＳＳは、ソフトウェア開発の生産性、信頼性を向上させるために開発されたものである。[1][2] ＳＤＳＳはソフトウェア開発における各工程でドキュメントを正確に作成し次工程で有効活用することにより、生産性・信頼性の向上を目指している。本報告では、ＳＤＳＳにおけるテスト支援の基本機能について述べる。

２．位置付け

テスト支援は、プログラム構造設計書とモジュール設計書を利用してモジュールテスト仕様書を作成し、これに基づいてテストを自動的に行うものである。

３．構成

図１に機能関連図を示す。テスト仕様書からテストデータとテストドライバを生成するテストプロ生成機能。下位モジュールの動作を表現するスタブ仕様書からスタブを生成するスタブ生成機能。ＨＣＰチャートを基にテスト用のソースコードを生成するテストソース生成機能。被テストモジュールの下位モジュールを記述した呼出関数一覧から情報を抽出する呼出関数登録機能。必要なソースコードを集め実行形式を生成する実行形式生成機能。テストを行い結果をテスト成績書に自動的に書き込むテスト実行機能。テストの状況を集計するテスト成績集計機能。自動テストと同一環境で被テストモジュールのHCPチャートを表示しながらデバッグ出来るチャートデバッガ。

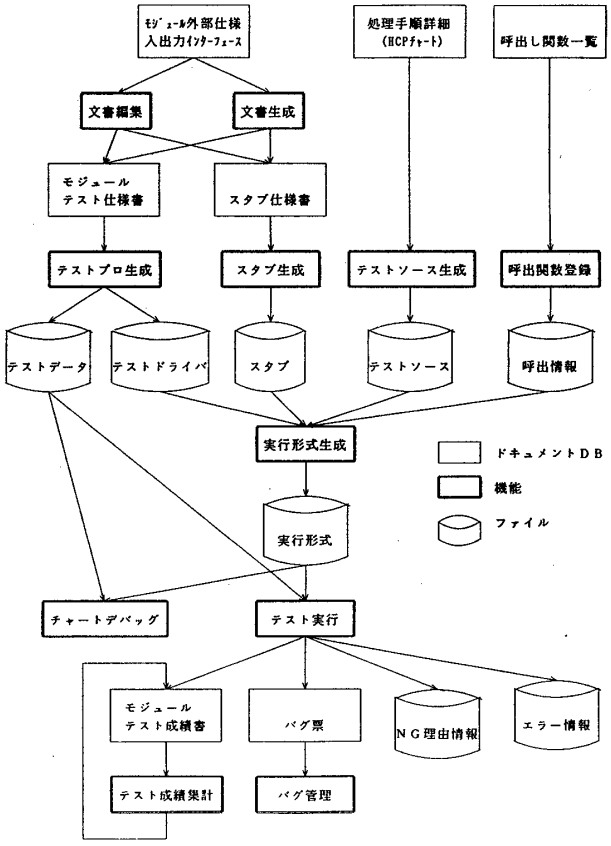

図１　機能関連図

４．特徴

(１)スタブ／ドライバの自動生成

従来モジュールテストを行う上で、テスト環境としてスタブ及びテストドライバをプログラミングする作業の手間が大きな障害であった。本機能はあらかじめ作成してあるドキュメント（スタブ仕様書/テスト仕様書）に基づいて自動的にスタブ/ドライバを生成することが出来るので、モジュールテストを容易に行うことが出来る。

(２)テスト良否の自動判定及びテスト成績書自動生成

あらかじめ決められたテスト条件をシステムが自動的に抽出し結果の良否を判断し、テスト成績書に書き込むので誤りのないテスト結果が得られる。

(３)再テストが容易

作成済のドキュメントに基づいてテストを行うため、モジュールの誤りを修正し再テストする場合も、自動的にテスト環境を生成し容易に再テストすることが出来る。

Testing Aid of the software development support system SDSS
Nobuaki Ito, Shouichi Iwami, Haruhiko Tanabe, Masayuki Nakamura.[1)]Matsushita System Engeneering Co.,Ltd.
Hiroshi Kiyooka.[2)]Matsushita Communication Industrial Co.,Ltd.
Satoru Fujii.[3)]Tokyo Research Labo.,Matsushita Electric Industrial Co.,Ltd.

5．機能概要

テスト支援は、種々のモジュールに対し、与えられたテスト条件より環境を自動生成し、テストを行い、自動判定結果を成績書として出力する。その中で、外部条件を与えて行うブラックボックステストと、内部の処理の経過を詳細に調べるホワイトボックステストを説明する。この機能は、図１のテスト実行に当たる。

（１）ブラックボックステスト

図２に示すテスト仕様書は、被テストモジュールのインターフェースデータに対する条件を記述している。テスト支援は、これに基づき初期値設定を行い、被テストモジュールの起動を行い、終了時に予測結果判定を行う。このとき実行結果が予測結果の条件に適合していれば、図３の様にテスト成績書にＯＫを書き込み、不適合ならＮＧを書き込む。下位モジュールをスタブ仕様書で定義している場合は、ダミーモジュールをリンクし、指定された動作をシュミレーションする。

また、結果がＯＫと判定されたケースに対し、カバレージが自動的に測定され、テスト成績書に書き込まれる。テスト用のソースコードとしてプローブ関数を埋め込んだものを生成することによってこれを実現している。

（２）ホワイトボックステスト

モジュールテストと言えども，全てブラックボックステストで行える訳ではなく、被テストモジュールの実行中に他のプロセスからの影響を受ける場合のテストをすることがある。そのために、テスト仕様書には被テストモジュールの実行途中にインターフェースデータが変化する旨を記述することが出来るようになっている（図２「箱　の　回目」の表現）。これは、（１）で述べたプローブ関数で実現している。また、テストケースに対しモジュール内のどのパスを通過すべきかをチェックするよう記述することも可能である。

6．おわりに

以上モジュールテストの支援機能を述べたが、ボトムアップテストであれば下位モジュールをリアル化することにより結合テストに拡張使用することが可能である。

今後、実機デバッグツールとテスト支援との連携により、更に下流工程のテストを支援することを計画している。

参考文献

[1] 藤井他：「ドキュメントの有効利用に着目したソフトウェア開発支援システム」　情報処理学会　34回全国大会(1987)

[2] 藤井他：「ソフトウェア開発支援システムＳＤＳＳの全体構成」　情報処理学会　37回全国大会(1988)

テスト手順

N:1ﾊﾞｲﾄｺｰﾄﾞ数値
S:1ﾊﾞｲﾄｺｰﾄﾞ文字列

作成日	1988.10.30	作成者	松下太朗
更新日		更新者	

モジュール名称	演算処理	略称[S]	operate

ケースNo [N]	2	ケース名	前の演算子が＋の場合

テスト内容	
実行順序	
打ち切り時間(分)	
判定条件	加算が実行されるか
箱　の　回目	

名称	型 [S]	略称 [S]	配列数 [N]	データの意味	初期値	予測結果
演算経過	extern int	memory		演算経過を保持する	10	1010
前の演算子	extern char	prev_op		１つ前の演算子	'+'	'-'
数値	para int	number		演算を実行する数値	1000	
演算子	para char	operator		次の演算を行う演算子	'-'	
戻り値	int	operate		0:正常終了 1:０による除算の実行 2:入力エラー		0

図２　テスト仕様書

テストケース項目一覧

N:1ﾊﾞｲﾄｺｰﾄﾞ数値
S:1ﾊﾞｲﾄｺｰﾄﾞ文字列

作成日	1988.10.30	作成者	松下太朗
更新日		更新者	

モジュール No[N]	2	モジュール名称	演算処理	略称[S]	bperate

ケース NO [N]	ケース名称	日付（YY/MM/DD）		結果
		開始日	終了日	
1	前の演算子がない場合	88/11/22	88/11/28	OK
2	前の演算子が＋の場合	88/11/22		NG
3	前の演算子が－の場合	88/11/22	88/11/30	OK
4	前の演算子が＊の場合	88/11/22	88/11/30	OK
5	前の演算子が／の場合（正常）	88/11/22	88/11/30	OK
6	前の演算子が／の場合（異常：０で割った場合）	88/11/22	88/11/30	OK
7	前の演算子が異常の場合	88/11/22	88/11/30	OK

カバレージ	C0	91%	C1	88%

未実行ボックス	3	全ボックス数	11個
未通過分岐	1-3	全分岐数	8個

ボックス NO.	ヒット数 5	10	15
1	*****|**7		
2	*****|*6		
3			
4	****4|		
5	*1		
6	**2		
7	*1		
8	***3		
9	*1		
10	*****|5		
11	*****|5		

分岐	ヒット数 5	10	15
1- 2	*****|5		
1- 3			
1- 4	***3		
1- 5	*1		
1- 6	**2		
1- 9	*****|*6		
6- 7	*****|**7		
6- 8	***3		

図３　テスト成績書