ソフトウェア工学 140－9
（2003．3．6）

PSPをもとにした形式手法有効性確認コースの設計

鈴森寿之† 海谷治彦† 海尻賢二†

VDMを学習している初学者がVDMの有効性を客観的に確認する方法を提案する。有効性は以下の順で確認する。(1) 特定ドメインを想定した小規模プログラム演習を数回繰り返えす。(2) 演習遂行に従い、VDMを段階的に導入する。(3) 各演習のプロセス・プロダクトのデータをそれぞれ測定する。(4) 測定データを基に欠陥除去等の変化を測定者に示す。

VDM over PSP: A Method for Confirming The Usefulness of A Formal Method

Hisayuki Suzumori,† Haruhiko Kaiya† and Kaijiri Kenji†

We propose a method for confirming the usefulness of a formal method during its learning course. A beginner of VDM can confirm its usefulness in the following way. First, He or she practices several number of exercises for software development. These exercises are designed for reflecting a specific problem domain, and techniques in VDM are introduced gradually in the series of the exercises. Second, process and produce data of software development are recorded in each exercise, and they are measured by several metrics. Third, by observing the valiation of a value measured by each metrics during the series of exercises, He or she can confirm the usefulness of suitability of VDM for him or her objective.

1. はじめに

The Vienna Development Method(以下VDMと記述)はプログラムの挙動を簡単な数学で記述しその正当性を確認できる設計法である。また、IFAD社が専用の解析ツール[6]を提供しているため、ソフトウェア開発の効率化・高品質化に貢献するものと思われる。しかし、一般に形式手法による設計は、UML等の図式言語を利用した設計よりもソフトウェア開発作業が困難になる場合がある[5]。その理由は、テスターやユーザのフィードバックに頼らずにプログラムの挙動を明確に書くのが難しいからである。特に、全ての状況で正しいプログラムの挙動を記述するのは不可能といってもよい。さらには、VDMを含む形式手法では、実装言語と同様の形式言語の記述を行うため、開発者にとって二度手間と感じる場合も多い。以上のように、形式手法による設計は欠点や不利な点ばかり目につき、その有効性を利用者に納得してもらうのは難しい。

そこで、本稿ではVDMを学習している初学者が自分が学んでいるVDMが確かに有効だということを客観的に確認するための方法を作成する。

† 信州大学 工学部
Faculty of Engineering, Shinshu University

提案する方法ではソフトウエア開発のデータを収集分析する必要がある。そこで、我々はソフトウエアのベンチマーク技法であるThe Personal Software Process[1](以下PSPと記述)をデータ収集分析に利用することにした。PSPに基づくデータ収集に慣れるには、それなりの訓練が必要となる。我々が提案する方法では、VDMの段階的な導入によるソフトウェア開発の変化に着目するため、PSPの習熟度の差による変化の影響を分離したい。よって、本方法の利用者はPSP訓練を事前に受けているものとする。

以下に本稿の概要を述べる。まず、PSPをベースにVDMを扱えるように拡張する。そして、VDMが有効であるといわれている部分に対してメトリクスを提案する。さらに、VDMが不必要だということを確認するメトリクスも提案する。これは、形式手法の導入により、開発に時間がかかったり、品質が向上するどころか悪化するなども状況も考えうるからである。そして、現状のプロセスとVDMを導入したプロセスをメトリクスを通し比較する方法で有効性と不必要性を確認する。

VDMに慣れていない開発者がVDM全体をすぐに開発に導入するのは困難である。そこで、本方法では、VDMを段階的に導入する方法と、VDMの利用ガイドも提供する。具体的には、通常の開発プロセスからVDMの利用をガイドに従い問題解決を進めるこ

とで、段階的に方法論を導入する。そして、導入過程のプロセス・プロダクトのデータをもとにVDMが自分にとって有効かどうかを確認することができる。

2. PSP

ソフトウエア技術者の仕事は、決められたコストとスケジュールに従って品質の高いソフトウエアを作ることである[2]。PSPは個人レベルの計画立案、品質管理の方法を提示し、自分のプロセスを如何に計測し如何に分析するかの枠組みを示す。PSPはプロセスが定義され開発をガイドするプロセススクリプト、作業途中で作るべきプロダクトをガイドするテンプレート、自分のプロセスを測定、分析するための帳票からなる。定義されたプロセス、テンプレートの規範に従うことで技術者は自分の行なってることをより理解して、自分の作業を管理し、自己分析し、改善していく枠組みを得る。PSPは個人レベルの比較的小規模なソフトウエア開発をターゲットとしており、その開発プロセスは直線的なウォーターフォールモデルである。そしてPSPは無理なくPSPの実践を導入し訓練できるように段階的に導入する枠組みと問題シリーズが用意されている訓練コースがある[1]。

2.1 PSPの品質管理

PSPの品質管理は欠陥を管理することに焦点を当てている。欠陥を記録し、それにに基づき欠陥を予防する戦略と時間記録、規模記録と合わせ作り込まれた欠陥を効率良く除去する戦略をとる。そして、メトリクスを使い行ったことを評価し改善する。

PSPの欠陥除去戦略は欠陥は作り込まれるものとしそれを如何に効率良く取り除くかということを考える。PSPでは効率の良い欠陥除去法としてレビューを導入する。設計、コーディングのあとにレビューをそれぞれ行う。

PSPで導入される欠陥予防戦略は二つある。効率良くレビューを行うためにコードレビューチェックリストを導入される。これをカスタマイズする方法と設計欠陥を少なくするために設計テンプレートと言われるものを導入する。

設計テンプレートは設計の表記品質の向上をはかった完全な設計記述の要素であるテンプレートの集合である。設計テンプレートの分類を表1に示す。PSPでは設計については特別な設計手法に依存しない表記についてのみ規定されている。

- 機能仕様テンプレート
  クラス構造、メソッドの外部仕様と事前条件を書く
- 操作シナリオテンプレート
  プログラムの使われ方を書く
- 論理仕様テンプレート
  各メソッドの動作を書く
- 状態仕様テンプレート
  プログラムの状態遷移を書く

表1 PSPテンプレート分類

オブジェクト仕様テンプレート	内部	外部
静的	論理仕様テンプレート	機能仕様テンプレート (継承クラス構造)
動的	状態仕様テンプレート	機能仕様テンプレート (ユーザインタラクション), 操作シナリオテンプレート

表2 欠陥型標準

型番号	型名	説明
10	文章	コメント, メッセージ
20	構文	スペリング、区切り、誤字、命令形式
30	ビルド, パッケージ	変更管理, ライブラリ, 版管理
40	アサインメント	宣言, 名前の重複, 範囲, 制限
50	インターフェース	手続き呼出しと参照, 入出力, ユーザ書式
60	チェッキング	エラーメッセージ, チェック不良
70	データ	構造, 内容
80	機能	論理, ポインタ、ループ、再帰、計算、機能欠落
90	システム	構成, タイミング, メモリ
100	環境	設計, コンパイル, テスト, 他の支援システムの問題

2.2 PSPで記録されるプロセスデータ

PSPにおいて一回の開発プロセスで記録されるデータは、各工程の時間記録、欠陥記録、ソフトウエアの規模記録、計画立案で行った見積り記録である。とくに欠陥記録は表2で分類されたものを使い欠陥を分類する。そしてその欠陥が作り込まれた工程除去した工程、修正に係った時間を記録する。

3. 形式手法

3.1 形式手法の利用法

形式手法は以下のように使うことが出来る[3][5]。

- Model
  プログラムの挙動のある部分を抜きだし単純化して記述し問題を簡単にする。
- Design
  図示し、プログラムの構造を書く方法論とは違い

プログラムの挙動を簡単な数学で簡潔に書くことが出来きる。そして挙動を理解した上でプログラムの構造を作ることが出来る。段階的に抽象化モデルから詳細化することが出来る。

- Verification
  形式的に書かれた仕様にプログラムがしたがっているか示すことが出来る。例えば、証明。
- Validation
  自分達が意図していることをプログラムがしていると言うことの信頼性を向上させることが出来る。例えば、ツールサポートにおける仕様実行によるテスト。

プログラムの挙動を数学を使う表記と記述した結果を解析出来るのが特徴である。形式手法は設計手法であり、設計の品質を向上させることで製品の品質を向上させる手法である。

3.2 VDM

VDM は 1973 年、IBM Vienna Laboratory で生まれた[3] プログラムの仕様を数学を使い記述できる形式手法の一つである。VDM の記述言語である VDM specification langrage(以下 VDM-SL と記述) はモデル指向言語である[3]。データ型を定義し、それを操作する関数で構成されている。主な記述機能として、関数を事前条件、事後条件で記述する方法、関数の挙動を明確に書く方法、データ型の不変条件の記述する方法がある。今回、IFAD の ToolBox[6] を使い VDM を利用する。ToolBox は VDM-SL を解析する手助けをするツールである。

3.3 ToolBox

Tool Box には主に以下の機能がある。

- Syntax check
  VDM-SL の構文エラーをチェックする。
- Type check
  型の整合性を静的にチェックする。
- Interpriter and Debugger
  インタプリタ, ソースレベルデバッカが付いている。ここでは動的なチェックが行える。データの不変条件の動的チェックと関数の事前条件の動的チェックをするかしないかの選択をそれぞれ設定出来る。VDM command を列挙したスクリプトを使うことができる。
- Integrity Examiner
  ランタイムエラーが起こりそうな場所を見付ける解析。そしてランタイムエラー が起こらない事を確認した integrity propaties を生成する。 interpriter で行う動的チェックをある程度自動化するもの。
- Test Facility (Systematic Testing)
  あらかじめ用意した一連のテストを呼出してテストカバレッジインフォメーションを 表示する。
- Automatic Code Generator
  VDM のコードから C++のコードを生成する。
- Dynamic Link Facility
  インタプリタで VDM-SL と C++とのダイナミックリンクを行える。

表 3 PSP 2.1 プロセス

始め	計画立案
	設計
	設計レビュー
↓	コーディング
	コードレビュー
	コンパイル
	テスト
終り	事後分析

表 4 VDM over PSP プロセス

始め	計画立案
	設計
	設計レビュー
	コードレビュー (VDM-SL)
	Syntax check(Use Tool)
↓	Type check(Use Tool)
	Valdation(Use Tool)
	コーディング
	コードレビュー
	コンパイル
	テスト
終り	事後分析

4. VDM over PSP

PSP を基にした VDM の有効性確認コースを、我々は “VDM over PSP” と呼ぶこととする。本節ではコースの概要と手段を述べる。

今回ベースとする PSP 2.1 の開発プロセスは表 3 である。設計工程で設計テンプレート表 1 を使い設計をし、レビュー工程ではそれぞれレビューチェックリストを使いレビューを行う。

4.1 VDM を PSP の何処に組み込むか

PSP は計画立案、品質管理に着目した開発プロセスである。VDM は設計手法であるので品質管理に組み込む。そして PSP の品質管理の規範を総て組み込まれた PSP2.1 を本研究のベースラインプロセスとする。

まず、VDM の品質管理について、そして VDM を導入したことによる欠陥記録の変更について、評価メトリクスの再定義について順に述べる。

4.2 VDM の品質管理

4.2.1 VDM の欠陥除去戦略

VDM と品質管理を追加した開発プロセスのタスクを表 4 に示す。

設計工程では設計を逐次 VDM-SL で記述して行く。設計レビュー工程は PSP の設計レビューそのままである。VDM-SL をツールに掛けるために設計レビューの後コードレビューを行う。PSP2.1 同様に欠陥除去工程ではそのタスクで欠陥を全て除去するように努力

表 5　VDM を使ったテンプレート分類

オブジェクト仕様テンプレート	内部	外部
静的	explicit function 記述	データ型記述、データの不変条件記述
動的	状態仕様テンプレート	implicit function 記述,pre-condition 記述, ユーザシナリオテンプレート

表 6　設計テンプレートを使った Validation の手順

工程	使用する設計テンプレート	使用する Tool の機能
関数個々の動作チェック	機能仕様テンプレート	Interpriter
		Interpriter と関数の事前条件の動的チェック
		Interpriter, 関数の事前条件動的チェックと不変条件チェック
仕様全体のチェック	操作シナリオテンプレート	Interpriter
		Interpriter と関数の事前条件の動的チェック
		Interpriter, 関数の事前条件動的チェックと不変条件チェック

する。Validation は Tool を使った仕様実行にとどめる。proof などは行わない。

4.2.2　VDM の欠陥予防戦略

VDM での欠陥予防戦略は PSP と同様にレビューチェックリストと設計テンプレートを使う。そして、設計テンプレートを利用し Validation での実行データを生成する指針にする。設計テンプレートに VDM-SL を適用するために以下の要素に分ける。

- implicit function 記述, 関数の事前条件記述
  関数の外部仕様記述
- explicit function 記述
  関数の動作記述
- データ型記述、データの不変条件記述
  データ構造記述

これと VDM-SL で表記出来ない部分を併せて VDM での設計テンプレートとする。それを表 5 に示す。

設計テンプレートを使った Validation の手順を表 6 に示す。関数の外部仕様記述とデータの不変条件から適当なテストデータを選んで関数個々の動作確認を行う。関数個々の確認が終ったならば操作シナリオテンプレートで記述したシナリオに従いテストデータを入れる。そのことで全体の動作を確認する。

表 7　欠陥記録の取扱

工程	欠陥型番号 10〜40	欠陥型番号 50〜100
計画立案		
設計		
設計レビュー		
コードレビュー (VDM-SL)	VDM-SL コーディング欠陥	設計欠陥
Syntax check(Use Tool)		
Type check(Use Tool)		
Validation(Use Tool)		
コーディング		
コードレビュー		
コンパイル	実装言語コーディング欠陥	
テスト		
事後分析		

表 8　再定義した COQ メトリクス

評価 COQ	評価 COQ=設計レビュー時間+ コードレビュー (VDM-SL, 実装言語) 時間/総開発時間
失敗 COQ	失敗 COQ=Syntax check 時間+Type check 時間+Validation 時間+コンパイル時間+テスト時間/総開発時間
総 COQ	総 COQ = 評価 COQ+失敗 COQ
A/F 比	A/F 比 = $\frac{\text{評価 COQ}}{\text{失敗 COQ}}$

4.2.3　欠陥記録の扱い

欠陥記録は表 2 で述べた要素を持つ。ここで欠陥記録の作り込み欠陥に注目する。記録された作り込み欠陥から表 7 のように区別する。表の設計欠陥と実装のコーディング欠陥が従来記録されていた欠陥である。VDM-SL 欠陥の部分が新たに増えた欠陥である。

4.2.4　評価メトリクスの再定義

特別にメトリクスを再定義しなければならないものは COQ に関してである。COQ は時間的なコストを評価するメトリクスである。各欠陥除去工程の時間を評価コスト、失敗コストに分けて時間的な経済性を計る。新たに追加された工程がどちらに入るか判断しなければいけない。

他は工程が従来より多くなったことと、欠陥の扱い方を欠陥記録の中から VDM-SL コーディング欠陥を除いたものだけで評価すれば良い。

COQ の失敗コスト・評価コストの分類で Validation は失敗コストに入れる。なぜなら Validation はテストに似るからである。新たに追加された設計レビュー、コードレビューは明かに評価コストである。Syntax check,Type check は実装言語のコンパイルと同様に失敗コストとする。再定義されたメトリクスの式は表 8 のとおりである。

以上のものを使い VDM over PSP の評価メトリクスとする。表 4 のプロセスで開発を行い、設計では

表 9　確認する有効性とメトリクスの対応

確認する VDM の利点	確認メトリクス
設計欠陥を Validation 工程までで除去する。後の工程に残さない。	設計欠陥抽出率
要求を網羅し設計を詳細に書くようになる	設計欠陥作り込み率
設計欠陥を容易に除去できる	設計欠陥除去影響力

表 5 を記述しレビュー工程ではそれにあったレビューチェックリストを使いを行い、再定義されたメトリクスで評価を行う。以上が PSP 2.1 に対し VDM を組み込んだ VDM over PSP である。

4.3　VDM over PSP の段階導入

VDM over PSP のベースラインプロセスは 4.1 節で記述したように PSP2.1 を使う。VDM の段階導入は 4.2.2 節で記述した VDM を使った設計テンプレートで分割した VDM-SL の要素に従い 3 つに分ける。この段階を経て従来の設計方法から少しずつ形式手法にならしていき、設計方法を移行する。

- VDM over PSP 0: 現状プロセス
  PSP2.1 を使い開発をする。
- VDM over PSP 1: 外部仕様記述
  データ型記述、データの不変条件記述,implicit function 記述, 事前条件記述を行う。
- VDM over PSP 2: 内部仕様記述
  上記に加え、explicit function 記述を加える。
  完全に VDM-SL で設計をする段階。
- VDM over PSP 3: Validation
  上記に加え、Tool を使い Validation する。
  VDM-SL で解析を行う段階。

設計テンプレートを段階的に変更して行くこととなる。VDM-SL のコードレビューは Validation 行うとき追加する。

5.　有効性確認メトリクス

本稿では VDM の有効性に疑問を持つ人が客観的に VDM の有効性を確認するためにプロセスとプロダクトのデータを利用する。これらのデータをもとに、形式手法の有効性を反映するメトリクスを提案する。したがって有効性を確認するために品質に着目したメトリクスを提案する。

5.1　確認する設計手法の有効性とメトリクス

表 9 が確認する形式手法の有効性とメトリクスである。VDM の品質向上戦略は表 9 の有効性が現われるように努力することである。

表 10　形式手法の不必要性と確認メトリクスの対応

形式手法の不必要性	確認メトリクス
生産性の低下	グラフでの生産性の傾きを最小 2 乗法で求める
設計欠陥の上昇	グラフでの設計欠陥の傾きを最小 2 乗法で求める

5.2　有効性確認メトリクスの詳細

PSP で定義される欠陥除去率と欠陥除去影響力の二つのメトリクスを拡張する形で定義する。

- 設計欠陥抽出率
  設計欠陥のみに着目し、作り込まれた全体の設計欠陥のうちある工程で除去出来た割合を示す。
  $\frac{\text{工程で除去した設計欠陥}}{\text{工程で除去した設計欠陥}+\text{全体の設計欠陥数}} \times 100$
- 欠陥除去影響力
  比較したい 2 工程に置ける、時間当たりに除去された設計欠陥の割合。工程 I と単体テストとの比較の式を示す。
  $\frac{\text{設計欠陥/時間 (工程 I)}}{\text{設計欠陥/時間 (単体テスト)}}$
- 設計欠陥作り込み率
  設計欠陥のみに着目し、作り込まれた全体の設計欠陥のうち何処で作り込まれたかの割合を示す。
  $\frac{\text{工程で作り込んだ設計欠陥}}{\text{全体の設計欠陥}}$

5.3　形式手法の不必要性確認メトリクス

表 10 が確認する不必要性と確認メトリクスである。

5.4　形式手法の不必要性確認メトリクスの詳細

- 生産性の低下
  VDM を導入したことで生産性が低下したかどうかを確認するためにいままで行ったコースウエアの問題の生産性をグラフで示す。生産性の式は以下に示す。
  時間当たりの生産性 $= \frac{\text{新規作成・修正 LOC 数}}{\text{総開発時間}} \times 60$
- 設計欠陥の上昇
  VDM を導入して総設計欠陥数がどのように変化したかを確認するためにいままで行ったコースウエアの問題のそれぞれの総設計欠陥数をグラフで示す。

5.5　メトリクスの見方

5.1 節と 5.3 節で提案した評価メトリクスに対し見方を示す。メトリクスは問題シリーズを全て解いたあとに順に並べグラフ化して比較する。

5.5.1　設計欠陥抽出率

表 9 で述べた、設計欠陥は Validation 工程までででほぼ除去出来ているかどうかを評価する。

Validation を導入することでコーディング以降にとりこぼしていた設計欠陥が少なくなれば良い。Validation 工程の設計欠陥除率がその効果を最も良く表す。

VDMで表記することの効果を考えると近似的であるが、その分だけ設計欠陥の除去が可能になったということである。

大きく設計欠陥の除去取り残しが減ればあればそのひとにとってはVDMは設計欠陥除去効果があるといえる。ないか少なければその人にとってはVDMでの設計欠陥除去効果はそのひとにとっては必要がない。

5.5.2 設計欠陥作り込み率

表9で記した有効性のうち、設計判断が設計工程で出来ているかどうかを評価する。

ほとんど作り込むのは設計とコーディングの2工程である。VDMを導入するに従い,設計欠陥の作り込みがコーディングで少なくなれば良い。

先と同様に大きくコーディングでの設計欠陥の作り込みが少なくなればVDMで設計をすることは判断しなければならない設計上の問題を設計で行うために効果があるといえる。変化が少なければあまり導入しても意味がないといえる。逆にコーディングでの作り込みが増えていればVDMを導入したことが足枷となっていると言える。

5.5.3 設計欠陥除去影響力

表9で述べた設計欠陥を容易に除去できるかどうかを確認する。

これはValidationでの欠陥除去の時間的経済性を示してくれる。各除去工程と比較して効果が大きければ将来、実際導入するときにValidationの時間を増やすことを検討すべきである。逆に少なければ減らすことを検討すべきであろう。あり得ないだろうがテストと変わらないのであればVDMを導入するのは明らかに余計だろう。

5.5.4 生産性の低下

これは5.3節で述べた不必要性確認メトリクスである。

ベースプロセスよりも段階導入するに従い、大きく生産性が低下してVDMの利点よりも無視できないならばこのひとにとってVDMは有効ではない。逆に、低下がVDMの効果と比べて許容範囲内ならばVDMは有効であるだろう。

5.5.5 設計欠陥の上昇

これは5.3節で述べた不必要性確認メトリクスである。

ベースプロセスと比べ段階導入するに従い、設計欠陥が減少するどころか増加したならばその人がVDMになれていか、VDMは余計なものである。この場合、VDMは欠陥予防としては有効ではない。逆に、低下しているならVDMは欠陥予防として有効であるだろう。

6. VDM over PSPとそのメトリクスの考察

この節では上記で提案したものに関して、主なものの理由付けを記述する。まず、6.1節で、4.2.3節において欠陥記録を分類し取り扱った根拠とその理由について記述する。6.2節で、5.1節において記述した有効性を確認することが何故メトリクスから行えるかの理由付けを記述する。

6.1 欠陥記録の分割理由

PSPの品質管理は2.1節で述べた。PSPの品質管理は欠陥を管理する。しかし、PSPでは記録される欠陥は実装言語の構文エラーであるコーディング欠陥と設計欠陥を一緒に記録し同じ欠陥として扱われてメトリクスが定義されている。どれぐらい除去出来たかが問題となるので、設計品質を欠陥からはかるには設計欠陥か実装言語のコーディング欠陥か区別する必要がある。それにValidationを導入しToolを本格的に使うとVDM-SLのコーディング欠陥が新たに現われる。その欠陥はVDM-SL特有の欠陥であり最終的な生産物であるプログラムには残らない。その欠陥を含めてPSPにあるメトリクスを計算しても最終製品の品質評価にはならない。

そこで、欠陥型標準をつかい分類する。欠陥型標準は型番号が大きくなるほど複雑な欠陥になっておりWatts S. Humphry[2]曰く

> 型番号10〜40はより簡単なコーディング関連の欠陥で、型50〜100はより複雑な設計関連の欠陥と言える。

これより表7のように分けた。

6.2 有効性確認メトリクスの根拠

有効性確認メトリクスについては5節で述べた。それは設計欠陥の除去に着目し設計品質向上を評価するものである。設計欠陥の原因は以下のものがある[2]。

- 設計の誤り
- 設計はどうあるべきか分かっていたのに単純なミスをおかした場合
- 要求された内容を誤解する
- 設計の表現のとり間違い

欠陥記録からは単純に欠陥があったことを記録するのみで何が原因か分からない。それが上記の設計欠陥原因のどれに当てはまるのか明確でない。そこで、PSPは一人で行う比較的小規模で要求がきっちり決まったものを開発するプロセスであるということから、ここでは設計表現のとり間違いは無く、要求が誤解さ

れることが無いとし、記録される設計欠陥は単純に設計の誤りであるか単純なミスであると仮定する。

また、設計品質は以下の2つがある[1)]。

- 表記品質
- 内容品質

今回、確認する方法論である形式手法では表記品質を向上させる要素はプログラムの挙動を書けると言う部分であり、内容品質を向上させる要素は仕様を解析出来るという部分である。上記の仮定より欠陥から見た有効性確認は表記品質は考慮しない。設計品質を向上させるのは単純に設計の誤りか単純なミスに注力して欠陥を除去することである。

7. 例

本稿の確認コースを使う場合の例を示す。これは思考実験であり、実際の例ではない。

研究で形式手法を使う必要性がある学生がいたとする。形式手法の記述法はプログラム言語のようだから、簡単にあるていど学習出来た。しかし、形式手法を実開発で使うモチベーションが上がらない。だが、研究テーマに関連して、ある程度プログラム開発での利点などを実際に行ってみて把握しなければいけなかった。

そこで、本稿の有効性確認コースを使うこととなった。実際にVDMをプログラム開発に適用したことがなくても段階導入とガイドラインに従えば良い。さらには、有効性確認メトリクスである程度何処がどう良くなり、悪くなるのかが客観的に示される。

問題を用意しなければいけないがこの学生はPSPに付属の問題シリーズを使った。PSPの問題シリーズは個人向けは9個用意されている。今回はVDM over PSP 0,1,2,3それぞれ順に2,2,2,3と割り振り行った。

段階導入とガイドラインに従い問題を解き、それぞれメトリクスを計算し記録する。終了時にそれぞれのメトリクスに対し計算結果を並べてグラフ化し変化を見た。

図1をみると、コーディング工程における、設計欠陥の作り込みは段階的に少なくなってきている。5.5.2節に記述した見方によると、これはVDMを使うことで前よりも設計判断を設計で行うようになったことを表す。

図2をみると、Validationまででほとんどの設計欠陥を除去出来ており、Validationは設計欠陥の全体の約10%を除去できている。さらにはテストでは数%しかのこっていない。Validationを導入する前と比べると著しい改善である。5.5.1節の見方によると、この学生にとって、VDMの導入により設計欠陥除去は早い段階で除去する効果があると言える。

図1 設計欠陥作り込み率

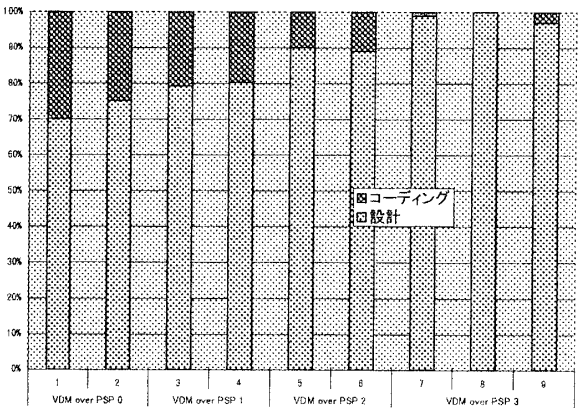

図2 設計欠陥除去率

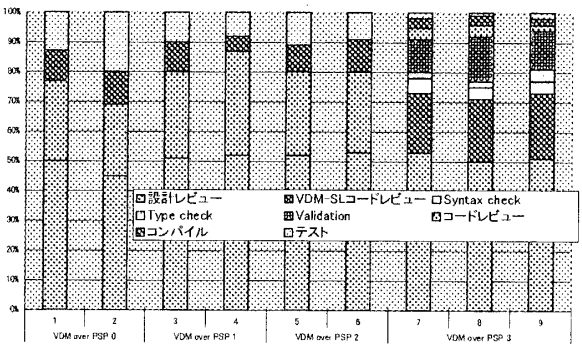

図3 設計欠陥除去効果

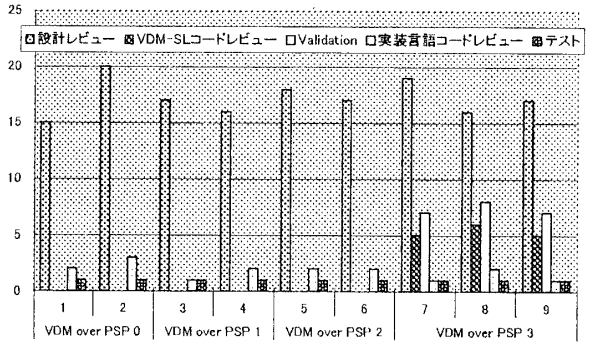

図3をみると時間効率としては設計レビューには全くかなわず、コードレビューよりも多少良い程度である。5.5.3節の見方によると、validationよりも設計レビューで設計欠陥を取るよう努力したほうが良いこととなる。

図4をみると生産性は著しく減っている。最終的には最初の半分になっている。最小2乗法での計算結果

図 4　生産性

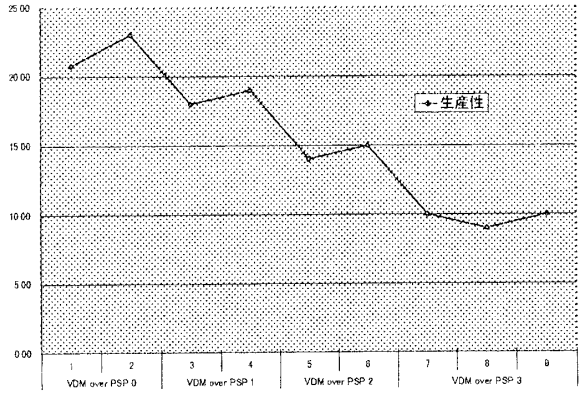

図 5　KLOC あたりの設計欠陥数

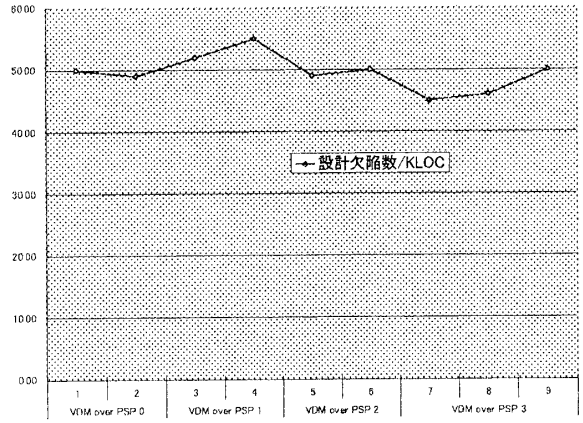

表 11　確認コースの結果

確認メトリクス	効果
設計欠陥作り込み	あり
設計欠陥除去率	あり
設計欠陥除去効率	悪い
生産性	低下
KLOC あたりの欠陥数	変化無し

の値は -1.752 である。5.5.4 節の見方によると、生産性をあげるという効果はこの場合 VDM にはない。逆に低下させる。

図 5 を見ると、設計欠陥の数は変わらない。最小 2 乗法の計算結果は-0.467 である。多少低下している様であるが微々たるものである。5.5.5 節の見方によると、この学生にとって VDM は設計欠陥を予防する効果はないといえる。

結果をまとめると表 11 のようになる。このひとにとっては生産性は低下するが、最終的なプログラムの品質をあげるのには効果がある。

この学生は普段の簡単なプログラム作成には使えないが時間を掛けてでも品質をあげなければいけないようなプログラムに対しては効果があると納得したようである。

8. おわりに

PSP をもとに PSP の品質管理部分に対し VDM を扱えるように拡張した開発プロセスを提案した。そして、記録したプロセスデータをもとに形式手法の解析による設計欠陥除去効果をはかるメトリクスとその見方を提案した。そのメトリクスを利用し形式手法の有効性を確認するためのコースのために段階導入を提案した。これは PSP で作業を管理し、測定し、自己分析を行うことを習得した人が改善策を模索するための一助となるものである。このコースウエアでは設計品質の表記品質、内容品質のうち表記品質をはかるメトリクスが提案できていない。さらには、形式手法の Validation の規範が詳しく示されていない。

今後の課題として形式手法の Validation の明確な規範の導入。今回作成した確認コースの適用実験をする。将来的には世の中の方法論の有効と言われていることに対し確認メトリクスを提案し、それを使い自分自身にあった方法論を客観的に選択できるようにしたい。

謝　辞

VDM のアカデミックライセンスを寄与して頂いた IFAD 社に感謝致します。また、ライセンス取得の際、E-Mail での質問に答えて下さった Anne Berit M. Nielsen 氏にお礼申し上げます。

参考文献

1) Watts S. Humphrey 著 ソフトウエア品質経営研究会訳「パーソナルソフトウエアプロセス技法」共立出版株式会社.
2) Watts S. Humphrey 著 PSP ネットワーク訳「パーソナルソフトウエアプロセス入門」共立出版.
3) John Fizgerald,Peter Gorm Larsen 著「Modelling Systems」Cambaridge University Press.
4) 荒木啓二郎, 張 漢明共著 「プログラム仕様記述論」オーム社出版局.
5) jonathan jacky 著「the way of Z」Cambaridge University Press.
6) IFAD 著 「VDM-SL Toolbox User Manual」IFAD 社.
7) 井上克郎、松本健一、飯田 元 著 「ソフトウエアプロセス」 共立出版社.