ソフトウェア品質技術の開発と適用

Development and Deployment of Software Quality Assurance Techniques

森　俊樹　■MORI Toshiki
櫻庭　紀子　■SAKURABA Noriko
中野　隆司　■NAKANO Takashi

一般に組込みソフトウェアの分野では，開発規模が飛躍的に増大するなか，製品出荷後のソフトウェアの不具合も増加して発生しており，製品開発の現場では，ソフトウェア技術者がデバッグやテストにかなりの時間を費やしている実情がある。

東芝では，このような状況に対処するために，上流から下流までの一貫した品質管理の枠組みとして“Wモデル”を提案し，上流工程における品質可視化技術の開発，及びテスト技術の体系化に基づくテストガイドの開発などに取り組んでいる。

With the exponential growth in the scale of development of embedded software, an increasing number of software defects are being detected inside products after shipment. Software engineers therefore often spend considerable time on debugging and testing.

In response to these circumstances, Toshiba has proposed "W model" as a framework for consistent quality management through the product development process. This includes quality visualization for upstream activities and a software testing body of knowledge for downstream activities.

1　まえがき

一般に組込みソフトウェアの分野では，開発規模が飛躍的に増大するなか，製品出荷後のソフトウェアの不具合も増加して発生しており，製品開発の現場では，ソフトウェア技術者がデバッグやテストにかなりの時間を費やしている実情がある。東芝でも組込みソフトウェアの品質確保は大きな課題であり，積極的に対応を進めている。

しかし，ソフトウェア品質は目に見えないため，外部から正確な状況を把握しづらく，品質に関する適切な診断や品質向上のための適切な対応が難しい。また，開発規模の増大に伴い，網羅的なテスト・検証の実施がますます困難になっている。そこで，当社は，品質可視化技術やテストガイドなどのソフトウェア品質技術を開発・適用することにより，製品及び開発プロセスの品質に関する診断，品質向上のための施策立案，テスト以降のフェーズの支援などを実施し，製品開発におけるソフトウェア品質の向上を目指している。

2　ソフトウェア品質の課題

最近のソフトウェア開発を取り巻く環境の変化は大きく，特に，組込みソフトウェアにおける開発規模の飛躍的な増大とTime-to-Market(注1)の大幅な短縮の結果，**図1**に示すように，多くの品質上の課題が発生している。

要求分析や設計などの上流工程では，仕様変更に対する

(注1)　製品を市場に投入するまでにかかる時間。

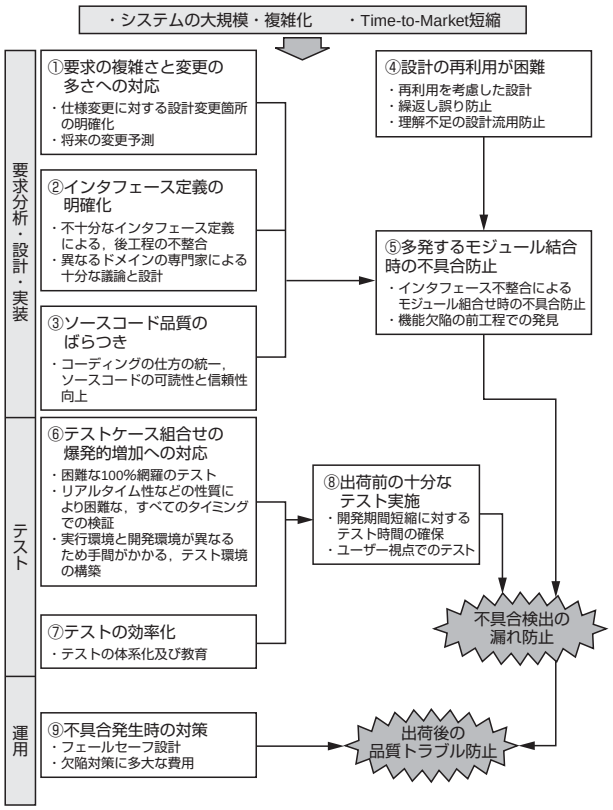

図1．ソフトウェア品質の課題 ― ソフトウェア品質の主要な課題を整理し，九つに分類した。

Major problems of software quality

設計変更の箇所があいまい，インタフェースの定義が不十分なため後工程で不整合が生じる，設計の再利用が進まないため新規開発により新たなエラーが発生する，設計を流用する際に使用上の制限事項を把握しきれない，などの課題がある。実装フェーズでは，ソースコードの品質にばらつきがあり，可読性の低いコードも存在する。テスト工程に先行する上流工程での品質問題は目に見えないため見過ごされがちであり，後工程になってモジュールを組み合わせたときに発覚することが多い。

テスト以降のフェーズにおいては，システムの大規模・複雑化により，テストケースの組合せの数が爆発的に増大しており，100％網羅的にテストするのは実質的に困難である。特に，組込みソフトウェアの開発では，リアルタイム性などの性質により，すべてのタイミングを検証することが難しく，更に，実行環境と開発環境が異なるため，テスト環境の構築にも手間がかかる。また，テストの体系化や教育が不十分なため，テストを効率的に実施できないという問題もある。

3　ソフトウェア品質技術の開発と適用

ソフトウェア開発は，多くの作業が連鎖して成り立っている複雑なプロセスであり，どこからでも欠陥が混入する可能性がある。つまり，ソフトウェアの品質問題には上流から下流まであらゆる工程が関係しており，したがって，切れ目のない，一貫した品質管理の枠組みが必要となる。

上流工程では，品質の“見える化”による，欠陥の混入防止と早期発見が重要である。ソフトウェア品質は目に見えないため，外部から正確な状況を把握しづらく，つい楽観的に考えがちである。しかし，複雑な作業を人手で行っていれば，確率的には，どこかでエラーが発生する可能性が高い。そして，欠陥が混入してから除去されるまでの滞留時間が長ければ長いほど，品質に及ぼす影響が大きくなる。“人がかかわれば，必ずエラーが発生する”という原則のもと，品質を可視化して欠陥の混入を未然に防止し，万一混入した場合は早期に発見して除去することが重要である。また，欠陥の混入を未然に防止する別のアプローチとして，設計の再利用や作業の自動化など，なるべく人がかかわらないように工夫することも重要であろう。例えば，自動車業界では，ツールベンダーに働きかけて，制御系設計ツールでモデリングした制御モデルからC言語のソースコードを自動生成できるようにしていることも，そうした取組みの一例である。

テスト以降のフェーズでは，テストを効率的に実施して，適切な品質保証を行う必要がある。しかし，現実には，テストケースの組合せの数が爆発的に増大する一方，納期は短くなっていて，十分なテストが実施できないケースもある。テストケースの組合せの数が爆発的に増加することへの対処として，リスク分析やユーザー運用記録などに基づくテストケースの優先度付け，直交表の利用などが考えられる。また，テストを効率的に実施するために，テスト技法とプロセスを体系化して，適切なテスト教育を行うことも重要である。更に，抜本的な解決策として，テスト工程をボトルネックとみなして，製品開発プロセス全体を最適化するようなアプローチも必要だろう。最近のテストファーストなどの考え方も，ある意味でそうしたアプローチの一つとみなすことができる。

当社は，上流から下流まで一貫した品質管理の枠組みとして，レビュー，インスペクション，静的解析，テストなどを中心とするV&V（Verification and Validation：検証及び妥当性確認）技術を基本とした，Wモデル（**図2**）を提案している。これは，従来のV字モデルに並行して品質管理プロセスを付加したモデルであり，上流工程の品質可視化技術，ソースコード品質保証のための静的解析技術，市場への不具合流出を防止するテスト技術から構成される。

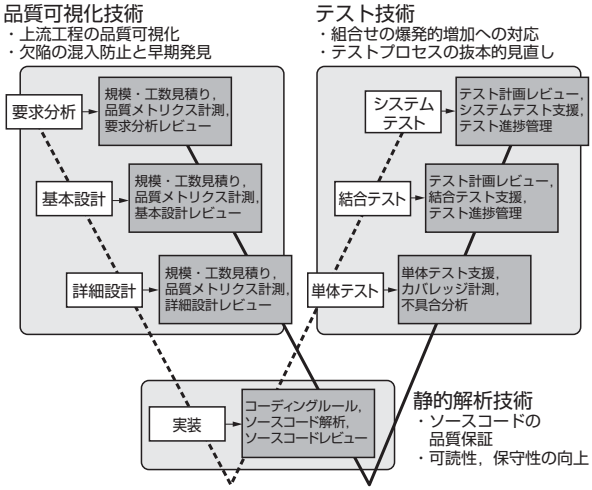

図2．品質管理の枠組み（Wモデル） ― 従来のV字モデル（破線）に，品質管理プロセス（実線）が並行している。

Framework for software quality management (W model)

以下の章では，最近の当社の取組みとして，品質可視化技術の開発と適用，及びテスト技術の体系化を目指したテストガイドの開発について述べる。

4　品質可視化技術の開発と適用

ソフトウェアの品質可視化は，狭義にはソフトウェアの品質状況を“目に見える形”で表現することと定義されるが，当社は，以下の三つの目的を達成するための活動を総合して，品質可視化ととらえている。

⑴　ソフトウェア品質の定量的な把握　　ソフトウェアの品質状況を数値として把握する。

⑵　定量的データに基づく品質の判断　　品質に関する定量的な値から“品質の良否”を判断し，その問題点を抽出する。

⑶　品質改善の方策の導出　　⑵で品質が悪いと判断された場合に，改善策を導き出す。

これらの品質可視化活動は，一般的には⑴から⑶へと段階的に導入されると考えられる。また，実際に品質可視化活動をソフトウェア開発プロジェクトに適用するためには，以下の機能が必要である。

・品質可視化活動の詳細なプロセス

・データ測定値やプロセス情報などを格納・蓄積するデータベース

品質可視化活動の枠組みを**図3**に示す。現在，当社は⑴のソフトウェア品質の定量的把握を中心に，品質可視化の要素技術の開発を進めている。

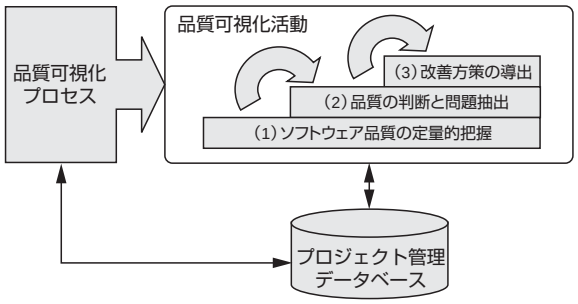

図3．ソフトウェア品質可視化の枠組み ― ソフトウェア品質可視化は3ステップの可視化活動，可視化プロセス，データベースから構成される。

Framework for software quality visualization

4.1　ソフトウェア品質定量化の流れ

ソフトウェアの品質の定量化を把握するためには，品質可視化プロセスに以下の作業が含まれる。

⑴　品質可視化の目的の明確化

⑵　品質メトリクスの設定

⑶　データ測定値の選択と収集

ここでの“品質メトリクス”とは，ソフトウェアの品質状況を判断するための指標値である。例えば，単体テストの不具合密度は品質メトリクスの一例であり，“単体テストで発見された不具合数（個）/開発規模”のようにデータ測定値から導出される。

ソフトウェア開発においては，**図4**に示すように，開発ライフサイクル全体を通して品質の定量化が実施されることが望ましい。

このような品質の定量化への取組みは，様々な組織で実施

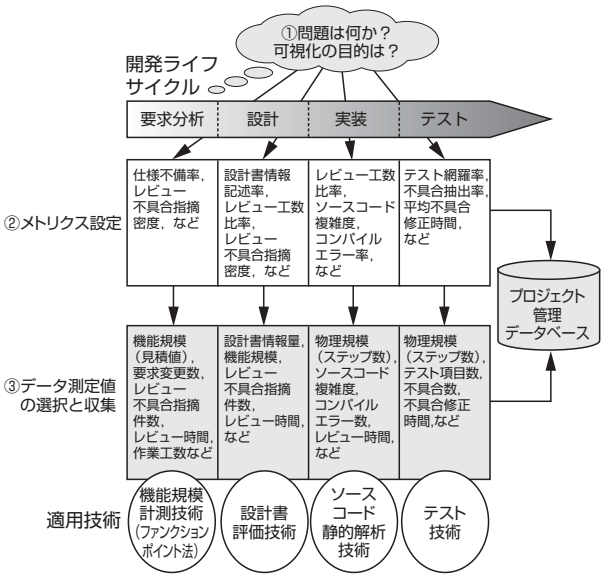

図4．品質定量化の流れ ― 開発ライフサイクルを通して品質の定量化が行われる。

Process of software quality metrics measurement

されているが，現実的には以下の課題が顕在している。

⑴　上流工程での可視化が不十分　　不具合数の測定などは主にテスト工程を中心に実施されている。テスト工程で発見された不具合が，上流工程である分析や設計工程のミスに起因している場合，手戻り作業に多大な時間とコストを費やすこととなる。このように課題を解決するためには，上流工程で適切に品質状況を把握することが必要となる。

⑵　定量化の目的が不明確　　実際にソフトウェアの品質可視化を実践している組織においても，品質可視化の目的，品質メトリクス，データ測定の間の関係があいまいになっているケースが多い。そのため，測定するデータ項目が多過ぎる，測定結果が開発プロジェクトにフィードバックされていない，などの問題を引き起こしている。

当社では，このような課題解決のために，設計書評価リスト及びメトリクスガイドを開発した。

4.2　設計書評価リスト

ソフトウェア開発の実装工程以前は，プログラムを実行して品質を確認することができない。そのため，設計工程までは主に設計書やテスト仕様書などのドキュメントが品質可視化の元情報となる。そこで当社は設計工程で作成される設計書に着目し，その記述の十分性を数値化して設計品質を表すために，設計書評価リストを開発した。

設計書評価リストは，開発に必要な事項が設計書にどの程度記載されているかを1,000点満点で評価する表計算ソフ

トウェアを利用したツールであり，以下の特長を持っている。

⑴　アンケート形式の階層化された質問項目
⑵　2種類の評価観点（第三者評価，設計者用評価）
⑶　質問項目の属性や配点の重み付けがカスタマイズ可能
⑷　回答結果をグラフィカルに表示

実際に，同一の要求仕様に対して複数者が作成した設計書を対象に，設計書評価リストの第三者評価を試行した。その結果，総合点や必須記述項目の記述率の差異などが確認できた。設計書評価リストは，評価を繰り返し実施して結果を追跡することで，設計品質の変化の把握が可能であると考えられる。また，設計工程のレビューポイントでこの評価を実施することで，上流工程から品質を作り込む意識を定着させるという副次的効果も期待できる。

4.3　メトリクスガイドによる品質可視化支援

ソフトウェアの品質を定量的に把握するために，やみくもにデータを測定・収集しても，その効果は得られない。それは，品質可視化の目的－必要な品質メトリクス－必要なデータ測定値，の三者の関係があいまいなことで，可視化活動が最終的に品質の改善につながらないためである。この問題に対して当社は，GQM（Goal-Question-Metrics）手法を用いたメトリクスガイドを開発し対応している。GQM一覧の事例を**図5**に示す。

メトリクスガイドの主な構成は以下のとおりである。

⑴　各開発工程での可視化の目的（Goal），把握したい事項（Question），品質メトリクス（Metrics）の一覧
⑵　各品質メトリクスの概要，定義，使用用途，分析方法，可視化例などの説明
⑶　各品質メトリクスを表現するために必要なデータ測定項目の説明

メトリクスガイドは，ソフトウェア開発部門での品質可視化プロセスの検討で利用されることを想定している。ガイドの活用により品質可視化の目的，品質メトリクス，データ測定項目の間の関係が明確になり，可視化プロセスの検討が効率的に行われることが期待される。また，メトリクスガイドはソフトウェアの品質可視化に必要なメトリクスとデータ測定項目をほぼ網羅しており，メトリクス一覧（95種類）やデータ測定項目一覧（96種類）なども備えており，自部門で測定可能なデータ項目からどのような品質メトリクスが把握可能なのかを調べるなど，逆引き的な利用も可能である。

実装			
Goal	Question分類	Question	Metrics
実装の進捗度合い	現在の進捗	要求された機能が実装されているか	実装割合
		テストは実施されているか	単体テストの有無
			結合テストの有無
		デバッグは終了しているか	未修正件数
		デバッグ後の再テストは終了しているか	修正確認テストの未実施件数
	進捗の遅れ	進捗は遅れていないか	実装の遅れ
実装の内容に問題はないか	エラーメッセージがないか	コーディング規約違反が検出されていないか	コーディング規約違反件数
	構造上の問題はないか	偏りがあったり，分割に問題がある箇所はないか	長すぎる関数
			複雑すぎる関数
		同じコードが複数の箇所で実装されていないか	多数の類似した関数
実装プロセスに問題はないか	開発者による評価に問題はないか	テストは十分にされているか	実施されたテスト項目数の十分性
			テストカバレージ率
		デバッグは終了しているか	未修正件数
			修正確認テスト未実施件数
		デバッグ後の再テストはなされているか	修正確認テストの実施割合
	頻繁に変更されていないか	度重なる変更がないか	変更回数（要求管理）
			変更回数（ソースコード）
			変更量（ソースコード）

図5．メトリクスガイド― メトリクスガイドにはGQMの一覧が記載されている。
Software metrics guide

5　テスト技術の体系化

ソフトウェアのテストは，ソフトウェア品質保証に関する“最後の砦（とりで）”と言われている。しかし，近年のシステムの大規模・複雑化，Time-to-Marketの短縮，顧客要求の多様化により，出荷前に十分なテストが行えずトラブルが発生するという問題が挙がっている。その要因としては，ソフトウェアテストが効率的に行えていない，テストケースの組合せの数が爆発的に増加しているなどが挙げられる。また，これらの原因をたどると，次のような課題につながる。

⑴　テストのノウハウが蓄積されていない。
⑵　テストスキルが属人化している。
⑶　テストプロセスが標準化されていない。

すなわち，ソフトウェアテストの体系化や教育が不十分ということが問題である。

そこで当社は，ソフトウェアテストのプロセスや技法など，テストを実施するために必要な知識を体系化し，テストガイドを開発して，テスト教育の準備を進めている。ソフトウェアテストの体系化にあたっては，技術的な面とプロセス的な面から行い，“テストガイド（技法編）”，“テストガイド（プロセス編）”としてドキュメント化した。

5.1　テストガイド（技法編）

テストガイド（技法編）では，文献調査に基づき66種類のテスト技法の分類とその解説を記載した。技法の分類一覧，若しくは索引（五十音順）からリンク（印刷した場合は，ページ番号）をたどることにより，すばやく情報を取り出せるようにしてある。テスト技法を実際に活用する場合は，参考情報

などを閲覧する必要はあるが，技法の目的や概要などテストに携わる者として最低限知っておくべき事項を中心に記載してある。すなわち，テスト技法の知識を補うハンドブック的な利用を想定している。

テスト技法は様々な観点で分類でき，互いに独立というものではないが，今回，テスト技法を七つの観点で分類を行った。テスト技法の分類を**図6**に示す。

文献調査などから抽出した各種テスト技法を，KJ法などを用いて整理し，“V字モデルに基づくテストのレベル”，“ISO9126（国際標準化機構規格9126）の品質特性に関するテスト技法”，“テストのやり方によるテストの種類”，“静的・動的による分類”，“テストケースの生成技法”，“テスト計測・評価の技法”，“受入れ確認”の七つのカテゴリーに分類した。

一般的にソフトウェアテストとは，プログラムやシステムを動かして動作を確認すること（“動的テスト”と呼ばれる）を指す場合が多いが，プログラムやシステムを実際に動かさない“静的テスト”もテスト技法の一種として含めている。静的テストの方が，誤り箇所が一目瞭然（りょうぜん）であるため，不具合の原因であるバグ発見の効率が高くなり，修正コストも安くなるというメリットがある。一方で，静的テストは最終成果物（プログラム本体）の品質を保証するものではなく，動的テストでないと発見できないバグもあるため両方のテストを実施することが望ましい。

更に，テストガイド（技法編）では，**図7**のように，個々のテスト技法に関して，“別名”，“英語名”，“概要”，“詳細説明”，“関連するテスト技法”，“参考文献”，“関連ツール”などを解説している。

5.2　テストガイド（プロセス編）

テストガイド（プロセス編）では，まず，推奨するテストプロセスを定めて，各工程におけるポイントや実施上のノウハウを整理した。また，テストに関する基礎知識として，テストの定義，用語，心構えなどをまとめた。更に，テストガイド（技法編）に掲載のテスト技法をどのような場面で使用すればよいかもガイドの中で示した。

テストの作業はテストケースを実行するだけではない。テストケースを実行する前のテスト計画やテスト設計，テスト準備，テストケース実行後の報告，品質管理，進捗（しんちょく）管理などのプロセスも重要である。ガイドでは，これらプロセスにおいて，どのような点に注意をして何を行わなければならないのかを明記してある。

特に，テスト計画段階でテストの目的と目標を明確にし，テスト設計にて，目的に応じたテスト技法を選択することが重要である。また，全体的な視点，個々のテストレベルでの計画，及びテストの実施が重要と考えている。上流工程とテスト技術との対応関係を**図8**に示す。

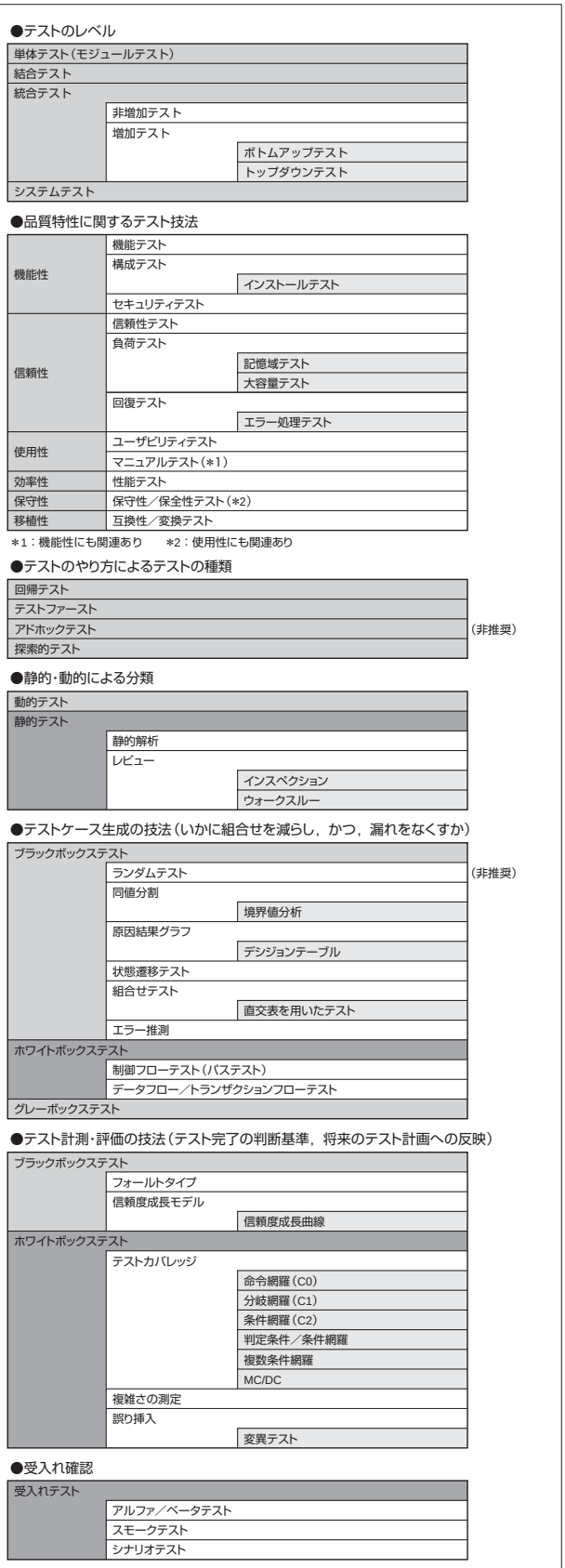

図6．テスト技法の分類 — 66種類のテスト技法を七つのカテゴリーで分類した。

Classification of software testing techniques

名　称	MC/DC
別　名	
英語名	Modified Condition/Decision Coverage
概　要	プログラム中の入口及び出口のすべてのポイントを少なくとも一度は実行し，プログラム中の判定でのすべての条件のとりうる結果を少なくとも一度は実行し，プログラム中のすべての判定のあらゆる結果を少なくとも一度は実行し，各々の条件が判定の結果に独立して影響を与えることを示すようにテストケースを作成する技法。

詳細説明	プログラムの条件判断部に注目したテストカバレッジの一種で，判定条件／条件網羅の改良版の技法。判定条件／条件網羅より網羅的であり，複数条件網羅より少ないテストケースで網羅できるとされる。判定条件・条件網羅との違いは，判定内の各条件による部分判定（例①での（A&B）など）の扱いにあり，判定内の部分判定の評価中にも状態が変わりうるシステム（リアルタイム性が要求される組込みシステムなど）で高い信頼性が求められる場合（航空宇宙関係など）にMC/DCが用いられる。なお例①を例②のように判定部を分解した場合は，判定条件・条件網羅とMC/DCは同じになる。
	
	例①
	if ((A&B)||c){
	　DO_SOMETHING;
	}
	
	例②
	if (A){
	　if (B){
	　　DO_SOMETHING;
	　} elsif (C){
	　　DO_SOMETHING;
	　}
	} elsif (C){
	　DO_SOMETHING;
	}
	
	ある条件が判定の結果に独立した影響を持つことは,その条件以外のすべての条件を固定したうえで，その条件を変更することで，判定の結果が変わることにより示される。例①においては次のようになる。
	
	①A（A変化，B・C固定）：TestCase1（True），TestCase2（False）
	②B（B変化，A・C固定）：TestCase3（True），TestCase4（False）
	③C（C変化，A・B固定）：TestCase5（True），TestCase6（False）
	
	TestCase1＝TestCase3のためTestCase3を消去できる。よって，五つのテストケースを実行すれば，100 %となる

関連するテスト技法	テストカバレッジ（↑），制御フローテスト（パステスト） 単体テスト，統合テスト，システムテスト全般
参考文献	（1）John Joseph Chilenski and Steven P. Miller, “Applicability of modified condition/decision coverage to software testing”, Software Engineering Journal 1994 （2）D0-178B

TestCase	A	B	C	if部
1	TRUE	TRUE	FALSE	TRUE
2	FALSE	TRUE	FALSE	FALSE
3	TRUE	TRUE	FALSE	TRUE
4	TRUE	FALSE	FALSE	FALSE
5	FALSE	FALSE	TRUE	TRUE
6	FALSE	FALSE	FALSE	FALSE

TestCase	A	B	C	if部
1	TRUE	TRUE	FALSE	TRUE
2	FALSE	TRUE	FALSE	FALSE
4	TRUE	FALSE	FALSE	FALSE
5	FALSE	FALSE	TRUE	TRUE
6	FALSE	FALSE	FALSE	FALSE

図 7．テスト技法の解説例 ― 各テスト技法の目的や概要など最低限知っておくべき事項を中心に記載してある。

Description of software testing technique

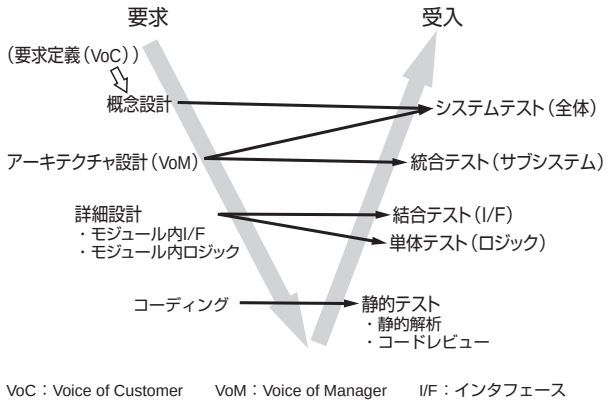

図 8．上流工程とテスト技術の対応関係 ― 上流工程の結果が，どのように検証プロセスで利用されるかを示している。

Relationship between upstream and software testing

6　あとがき

ここでは，ソフトウェア品質の主要な課題を整理し，それに対処するためのソフトウェア品質技術の枠組みとして，従来のV字モデルに並行して品質管理プロセスを付加した“Wモデル”を提案した。特に，最近の当社の取組みとして，品質可視化技術及びテストガイドの開発を取り上げ，ソフトウェア品質可視化活動の枠組みの紹介，その要素技術である設計書評価リストとメトリクスガイドの開発，テスト技術の体系化に基づくテストガイドの作成などについて述べた。

今後は，品質可視化の要素技術を実際の品質管理活動で適用することで，より効果的な品質可視化が実現できるよう改良を進めていく。また，テストガイドに基づくテスト教育の整備やコンサルティングなどの社内展開を行っていくとともに，個別のテスト技術のスキルアップにも努めていきたい。最終的には,個々のソフトウェア品質技術を高めていくことで，Wモデルに基づく支援体制を整備し，当社のソフトウェア品質の向上に貢献したい。

文　献

⑴ Roger S. Pressman（監訳：西 康晴，ほか）．実践ソフトウェアエンジニアリング．日科技連出版，2005，p.307－333.

⑵ John McGarry，ほか（監訳：古山 恒夫，ほか）．実践的ソフトウェア測定．（株）構造計画研究所，2004，272p.

森　　俊樹　MORI Toshiki

ソフトウェア技術センター 技術開発担当参事。ソフトウェア品質に関する技術開発に従事。ASME，精密機械工学会会員。

Software Engineering Center

櫻庭　紀子　SAKURABA Noriko

ソフトウェア技術センター 技術開発担当主務。ソフトウェアの品質管理技術，見積り技術の開発に従事。情報処理学会会員。

Software Engineering Center

中野　隆司　NAKANO Takashi

ソフトウェア技術センター 技術開発担当。ソフトウェア開発の品質技術開発に従事。情報処理学会，品質管理学会会員。

Software Engineering Center