ソフトウェア品質向上への取組み
Software Quality Improvement Techniques

平山　雅之 HIRAYAMA Masayuki
植木　克彦 UEKI Katsuhiko
宮本　隆一 MIYAMOTO Ryuichi
岡安　二郎 OKAYASU Jiro
岸本　卓也 KISHIMOTO Takuya
藤巻　昇 FUJIMAKI Noboru
増岡　範雄 MASUOKA Norio

ソフトウェアが身の回りのあらゆるところで広く利用されるようになり，より高い品質がソフトウェアに求められるようになってきている。ソフトウェアの品質向上には，ソフトウェア設計過程の改善とともに，実装されたソフトウェアのテストや検証，あるいは検出した不具合の確実な除去などの過程の革新が不可欠である。当社では，テスト項目重要度の評価による効率的なテスト手法の開発や，シミュレーションを活用したテスト環境の構築，更にはWeb技術を活用した不具合情報の共有促進など，テスト・検証，不具合管理の革新に向けた技術の開発・普及を推進している。

Software systems have recently proliferated greatly and become a pervasive presence both in the life of individuals and in society at large. Accompanying the expansion of software use, it is essential to ensure the high quality of software. Sufficient software testing, verification, and fault elimination are the most important techniques for improving software quality.

This report focuses on techniques related to software testing, verification, and fault management with examples of their application.

ソフトウェア品質向上への取組み——概論
Overview of Software Quality Assurance Approach

1 まえがき

ソフトウェアが様々な領域で活用されるのに伴い，その品質が大きくクローズアップされている。特に近年のネットワーク技術の進化により，様々な周辺装置と組み合わされたシステムとしての品質も注目されている。

単純な機器も，ソフトウェアを利用することによって従来にないきめ細かな機能を備えるようになったが，これらのソフトウェアは，様々な動作条件や利用シーンを考慮した複雑な論理構造を内部に含んでいる。こうしたソフトウェアではテスト・検証技術を活用して，様々な条件やシーンでの動作を確実に保証することが求められる。また，ソフトウェアのテスト・検証で検出された不具合は，確実に修正することが必要になる。このための不具合管理技術も，ソフトウェアの品質を向上していくうえでの不可欠な技術である。

2 ソフトウェア品質向上に関する課題

ソフトウェアエンジニアリングは，ソフトウェア品質保証に関しても様々なソリューションを提供してきた。しかし，近年，急激な速度で変化しているソフトウェア開発のなかで，特に以下の点が新たな課題となりつつある。

⑴ ソフトウェア開発量の増大　ソフトウェア技術の進化により，様々なことがソフトウェアでできるようになってきている。これに伴い，ソフトウェアの規模や複雑さが飛躍的に増大している。この結果，テストや検証での確認事項が多くなり，また，テストを実施する条件や環境も多くのバリエーションが必要になるなど，ソフトウェアのテストや検証が従来以上に難しくなっている。

⑵ 開発期間の短縮　ソフトウェアを含むシステムの製品寿命は，急速な技術革新とともに短くなる傾向がある。このため，ソフトウェアの開発期間も従来に比べると短くなり，テストや検証に充当できる時間も短かくなっている。

⑶ 製品バリエーションの多様化　最近のソフトウェア開発では，標準ソフトウェアに対するバリーエション付加により，様々な顧客に対応する進化型開発が主流となっている。進化型開発では，特定バージョンで発生した不具合に関する修正が多くの関連バージョンに影響する。このため，バージョン間の不具合情報の共有などが大きな課題になる。

3 ソフトウェア品質向上の枠組み

当社ではソフトウェアの品質保証を，① 設計・実装工程での品質作り込み，② テスト・検証による確実な動作保証，の二つのアプローチを柱にして進めている。

⑴ 設計・実装工程での品質作り込み　当社では品質作り込みを，ソフトウェアの設計や実装などの各段階で品質をより確実なものにする技術として位置づけている。実際の開発では，設計・実装を含むすべての開発ステップで，開発確認作業としてデザインレビューを実施し品質作り込みを進めている。

⑵ テスト・検証による確実な動作保証　テスト・検証では，後述する選択的テスト手法やシミュレーションを活用したテストにより，効率的に重要な不具合を確実に除去する方式を採用している。また，Webを活用した不具合管理システムにより，検出した不具合の確実な修正とフォローを実現している。

4 ソフトウェア品質向上に関する研究とその適用

4.1 最適なソフトウェアテスト項目の設計

短時間で効率的なテストを行うには，① 適切なテスト項目の抽出，② 適切なテストデータの用意，が重要である。この実現に向けて，選択的テスト手法の研究と実用化を進めている。この手法では，ソフトウェアが提供する機能に重要度を付け，重要度に応じてテスト項目の詳細度を設定する。重要な機能に対しては，様々な動作パターンを含む詳細なテストを行う。一方，それほど重要でない機能は，基本動作だけを簡便にテストする。これによって，テストの効率化と品質の重点チェックを実現している。

4.2 シミュレーションを利用したテストの効率化

近年のソフトウェアは，様々な機器や関連ソフトウェアと連携して動くものがほとんどである。こうしたソフトウェアのテストでは実際の周辺機器，周辺ソフトウェアをテスト環境として用意する必要がある。しかし，この方式では，テスト環境の整備にコストが掛かり，また，テスト環境整備がテスト着手時までに準備できないなど問題点も少なくない。これに対処するため，周辺機器も含むテスト環境をコンピュータ上のシミュレータとして用意し，早期のテスト着手を可能とする仮想テスト技術を研究し，実際の製品開発に投入している。

4.3 Webを利用した不具合情報の共有化

テストで発見した不具合情報を，いかに迅速に開発担当者へフィードバックするかは大きな課題である。また，複数の異なるバージョンを同時に開発，出荷する場合などは，バージョンをまたがった不具合情報の共有が重要になる。当社では，Web技術を利用して不具合情報を共有化するツールPRISMY$_{TM}$(PRoject Information Sharing and Managing sYstem)を開発し，製品開発の中で利用している。

5 あとがき

ソフトウェアのテスト・検証，不具合管理は，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，先に述べたように，近年のソフトウェア開発では，これらの作業が極めて難しくなっている。ソフトウェアエンジニアリング分野でこれまで研究されてきた技術を応用し，現場ニーズに合ったテスト・検証技術を開発，提供し，いっそうの品質向上に努めたい。　（平山）

選択的テスト手法によるテストの効率化

Selective Testing Method

1 まえがき

ソフトウェアのテストは，一般に単体テスト，結合テスト，システムテストの三つに分けられる。この中でソフトウェアシステムが提供する機能が当初のシステム仕様に合致しているかどうかを判定するのがシステムテストである。近年のソフトウェアシステムが提供する機能は膨大になっており，システムテストでテストすべき機能も増加の一途をたどっている。

従来のシステムテストでは，機能やコードレベルでのテストの網羅率を向上させる考え方が中心であった。しかしながら，近年の肥大化したソフトウェアシステムでは，網羅率を念頭に置くとほとんど無限に近いテスト項目が必要になる。こうした状況を解決する手だてとしては，いかに意味のあるテスト項目をテストするかが重要になる。ここで意味のあるテスト項目とは，いかにシステムに内在する不具合を的確に検出できるかということと同義である。つまり，多くのテスト項目の中から，フィールドでの影響がより深刻な不具合を検出する率の高いテスト項目をよりすぐって，効率よくテストする手法が有効であると考えられる。当社では，こうしたテストの考え方を選択的テスト手法と呼び，この手法の具体的実現方式の研究・実用化を進めている。

2 選択的テスト手法

2.1 概要

選択的テスト手法は，① 機能の優先度決定，② テスト仕様書の作成，③ テスト計画の作成，の三つのフェーズから構成される。その概要を**図1**に示す。

まず，ソフトウェアシステムが提供する機能にテスト優先

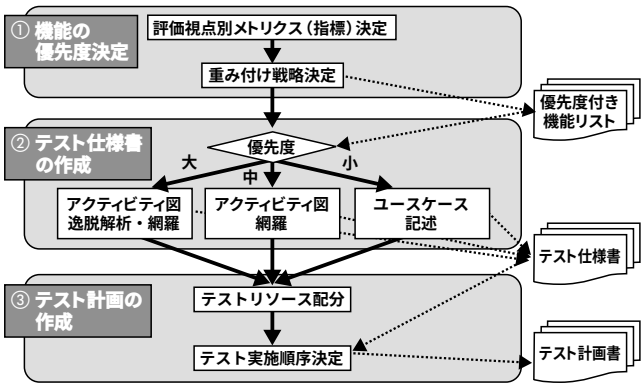


図1．選択的テスト手法の流れ　選択的テスト手法では，テスト対象機能の優先度決定，テスト仕様書作成，テスト計画作成の三つのフェーズにより，効率的なテストを実現する。
Outline of selective testing method

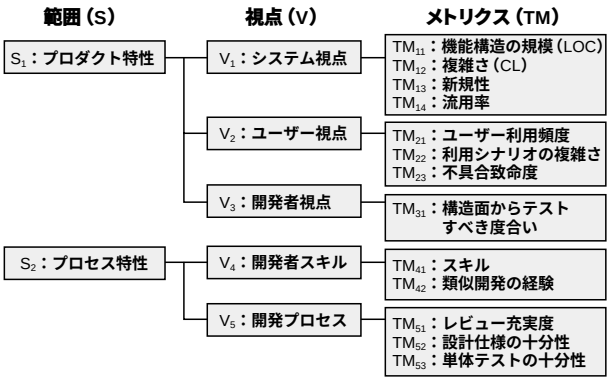


図2．優先度付けのための評価観点　テスト項目は，プロダクト，プロセスのそれぞれの視点から優先度付けを行う。
Evaluation viewpoints of testing priorities

度を付与する。そして，この優先度に応じてテスト項目の詳細度を変えたテスト仕様書を作成し，重要な機能ほど詳細にテストする。また同時に，優先度を参考に，テストの実行順序やリソース配分を最適化し制御する。つまり，与えられたシステムテスト期間やリソースを有効に使ってフィールドでの影響が深刻な不具合を確実に除去することで，より短い期間で最大の信頼性を得ることを目指している。

2.2　機能優先度付け

2.2.1　考え方　深刻な不具合はテストフェーズで確実に検出し，フィールドでの発現を防ぐ必要がある。このため，影響の大きい不具合がより多く存在する箇所を重点的にテストすることが求められる。選択的テスト手法では，ソフトウェアシステムが提供する各機能に対して，ユースケースの情報などを分析する。そして，機能の利用頻度やフィールドでの影響度など複数の視点から評価して，機能ごとにテストに関する優先度付けを行う。

2.2.2　優先度付けのための評価視点　ソフトウェアの不具合に関する基本動作は，不具合の混入，発現，影響の三つである。これをソフトウェア開発に照らし合わせると，不具合混入はプロセスに起因し，不具合の発現や影響はプロダクト側から把握することができる。このため評価の範囲（視点）をS_1：プロダクト特性，及びS_2：プロセス特性の二つに分けて考える（**図2**）。

⑴　**プロダクト特性**　プロダクト特性は，プロダクトとしての対象ソフトウェアのどの部分を重点的にテストすべきかを決定する特性であり，ソフトウェアとして実現される機能ごとに決定される。それには次の三つの視点がある。

V_1：システム視点
V_2：ユーザー視点
V_3：開発者視点

システム視点では，システムを構成する各部分の新規性や流用度合いなどを評価する。

ユーザー視点では，ソフトウェアの利用や保守のしやすさの観点から評価する。また，開発者視点では，機構の複雑さや機能の重要性などから見て，どの部分を重点的にテストすべきかを評価する。

⑵　**プロセス特性**　プロセスは不具合の混入に大きな影響を持つと考えられる。プロセス特性では，個々の機能の開発過程を，以下の二つの視点から評価する。

V_4：開発者スキル　どのようなスキルをもった開発担当者が開発したか。
V_5：開発プロセス　どのようなプロセスで開発を進めたか。

2.2.3　評価尺度とテスト戦略

⑴　**評価尺度**　テスト優先度付けでは，各機能の上述のそれぞれの視点から見たテスト優先度を定量的に評価する。このため，上述の各視点V_iごとに，これを計測，評価するための尺度（TM_{ij}）を用意する。例えば，システム視点V_1に関しては，TM_{11}：機能構造の規模（LOC：Line Of Code），TM_{12}：複雑さ（CL：サイクロマティック数），TM_{13}：新規性，TM_{14}：流用率，などを尺度として，それぞれ1～10の値をつける。

⑵　**テスト戦略**　実際のテストでは，最終的にどの機能あるいはどの部分に重点を置くかを判断する必要がある。例えば，エンドユーザーが重視する機能に重点を置く場合もあれば，開発担当者がテストしたほうが良いと考える機能を重点的にテストする場合もあり，先に付けた尺度値を複数視点から総合的に判断しなければならない。これをテスト戦略と呼ぶ。テスト戦略はテストにおいてどの視点を重視するか，どの尺度を重視するかを考慮して，尺度に重み付けを行うことで実現する。

例えば，ある機能Kの評価尺度TM_{ij}の値をX_{ij}，その評価尺度の重みをW_{ij}とする。テスト優先度は次の式

で計算される（ここでW_{ij}は重み係数で，$0 \leqq W_{ij} \leqq 1.0$の任意の値を設定する）。

テスト優先度（機能K）$= \Sigma W_{ij} \times X_{ij}$

あるテストでユーザー視点（V_2）を重視する戦略を採用する場合には，ユーザー視点に関する尺度であるユーザー利用頻度（TM_{21}），不具合致命度（TM_{23}）などの重み係数を1.0に近い値にする。また同時に，開発者視点（V_3）などに対する係数を0.2程度にする。

2.3 テスト仕様書作成

先に決定したテスト優先度によって対象を以下の三つのタイプに分け，それに応じてテスト仕様書の作り方を変える。

Type-1：優先度大の機能　この機能については，特に詳細にテストすることが求められるため，正常動作及び正常動作から逸脱する場合の利用シナリオを基に，その機能の致命的な不具合を想定してテスト仕様を作成する。

Type-2：優先度中の機能　基本的な正常動作の利用シナリオを網羅する形でテスト仕様を作成する。

Type-3：優先度小の機能　この機能は，それほど重きをおいたテストを求められないため，基本仕様書に記述された動作が正常に動作することを中心に確認する。

このようにして，テスト仕様の段階でテスト優先度を考慮し，テストの濃淡をつけるアプローチを採用している。

2.4 テスト計画作成

テストの実施にあたっては，テスト戦略に従ったテストチーム編成，テストサイクルの設定，テストケース振分けなどが必要になる。選択的テストでは，採用するテストケースをテストサイクルごとに入れ替え，また，どのようなテストリソース（要員など）をどの部分のテストにあてるかなどを含む工程の最適化を行って，テスト効率の向上を図る。

例えば，テストケース選択手法としては，すべてのテストケースを実施するフルテストと，優先度が高い項目だけを実施するクイックテストの二通りのテスト方式を用意している。新規機能組込み直後やサンプルリリースなどのマイルストーンではクイックテストを行い，通常はフルテストを行うなど，テスト作業を最適化する。

3 適用事例

3.1 適用概要

以下，選択的テスト手法を情報通信関係の組込みシステムに適用した事例を紹介する。

この例では，まずシステムが提供する個別機能の優先度付けを行い，これに基づきクイックテスト，フルテストを混ぜる形で選択的テスト手法を適用した。

機能優先度付けでは，システムの提供する約120の機能に対して，プロダクト特性を重視し，V_1：システム視点，V_2：ユーザー視点，V_3：開発者視点から優先度付けを行った。この結果，例えば，サブシステムCについては，優先度大のテスト項目が247，優先度中のテスト項目が117，優先度小のテスト項目が643となった。これらより，サブシステムCに対するクイックテストでは，優先度大に相当する247項目，フルテストでは優先度大～小まですべてを含めた1,007項目をテスト項目として抽出した。

3.2 適用評価

サブシステムCのクイックテスト，フルテストの状況を**表1**に示す。テスト期間中，サブシステムCではクイックテストを延べ4回，フルテストを2回実施した。クイックテスト，フルテストのそれぞれに要した期間は，平均で1.0日及び5.5日となった。また，表に示すように，この延べ4回のクイックテストでテストしたテスト項目数は988項目で，20件の重大バグが検出できた。一方，延べ2回のフルテストでは合計で2,014項目を試験し，6件の重大バグを検出した。この適用では，選択的テスト手法を利用したクイックテスト，フルテストの実施について，以下のような効果が確認できた。

⑴ 選択的テスト手法を活用することで，システムの重要な部分を重点的にテストすることができる。特にクイックテストではテスト期間の短縮効果が大きい。

⑵ 不具合検出については，表中の不具合検出率を見ると，クイックテストのほうがフルテストより大きくなっており，重大不具合をより効率的に検出できる。

⑶ クイックテストとフルテストを組み合わせてテストすることで，まんべんなくかつ重要なところは深くテストすることができ，システム全体の確実な品質保証が可能となる。

表1．適用結果
Application results

項　目	クイックテスト	フルテスト
テスト項目数　（項目）	247	1,007
繰返し回数　（回）	4	2
延べテスト項目数（項目）	988	2,014
重要不具合検出数　（件）	20	6
不具合検出率	0.020	0.003
テスト期間　（日）	1	5.5

4 あとがき

ここでは，ソフトウェアテストの効率化を実現する選択的テスト手法を紹介した。適用事例にも示したとおり，この手法によって，システムの重要な機能に関する不具合のより少ない時間での検出が実現でき，高品質のソフトウェアのスピーディな提供が可能になる。

（平山／植木／宮本）

シミュレーションを利用したテストの効率化

Testing Using Simulation Technique

1 まえがき

ソフトウェア組込み製品は複数の専用ハードウェアとソフトウェアのユニットから構成されており，外界とやり取りしながら機能する。その開発には，ソフトウェアだけではなく電気や機械など異なる分野の技術が必要であり，複数のグループによるハードウェア／ソフトウェア並行開発の形が採られる。このため，組込み製品開発のテストには以下のような問題がある。

⑴ 高額な試作テスト環境　　テスト用の試作機は，量産機とは異なり非常に高価であり，開発に十分な数を確保するには相当額の投資が必要となる。

⑵ ユニット開発遅れの全体への波及　　特定のユニットの開発に遅れが生じると，関連するユニットのテストに着手できず，結果としてシステム全体の開発に遅れが生じる。

⑶ 困難な結合テスト　　各ユニットの動作が十分検証されていない状態で結合テストを行うと，問題が生じた場合にどのユニットに問題があるのか，あるいは純粋に結合に起因する問題なのか否かなどの判断が難しい。

このため，他のユニット開発の進捗（しんちょく）からは独立に，たとえハードウェアがなくてもソフトウェアのテストが行えるような環境が求められている。

2 シミュレーションを利用したテスト環境

2.1 アプローチ

前述のような問題の解決には，ソフトウェアのテストで必要になる他のユニット，及び外界をシミュレーションすることによって，ソフトウェア的に開発環境上に実現する方式が有効である。これにより，他ユニットの開発状況の影響を受けずに，ソフトウェアをテストすることが可能になる。この方式のイメージを，図3に示す。

シミュレーションするユニットの種類に応じて，以下のようなシミュレーションのタイプがあり，これらのシミュレーションを必要に応じて組み合わせることにより，テスト環境を構築する。

⑴ 外界のミュレーション　　組込み製品が情報や物をやり取りする外界の環境をシミュレーションする。冷蔵庫を例にとれば，利用者の操作や庫内の温度，外気温などを生成する。

⑵ ハードウェアのシミュレーション　　外界とのやり取りを実際に行うアクチュエータやセンサをシミュレーションするとともに，ソフトウェアを実際に動作させるハードウェアなどもシミュレーションする。冷蔵庫を例にとれば，外界の温度を読み取る温度センサ，ドアセンサ，温度を制御するコンプレッサなどがハードウェアに当たる。また，冷蔵庫のソフトウェアを搭載するCPU，メモリ，周辺回路などもハードウェアである。

⑶ OS/ドライバのシミュレーション　　ハードウェアをプログラム制御するための専用ソフトウェアであるドライバや，ソフトウェアを機器上で動作させるためのOS（基本ソフトウェア）をシミュレーションする。いずれも，ソフトウェア開発環境上では動作しない，機器専用のソフトウェアである。

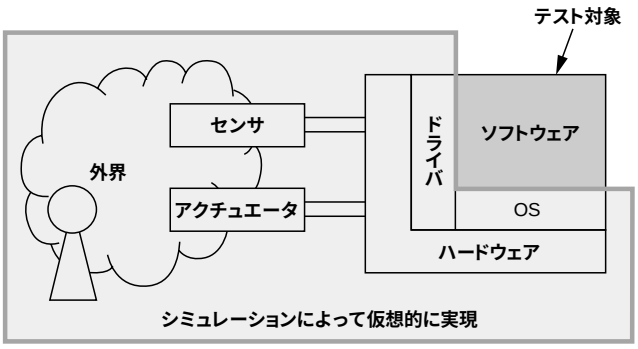


図3．シミュレーションを利用したテスト環境　テスト時に不足している様々な部分をシミュレータで補ってテストを実施する。
Testing environment using simulation technique

2.2 効果と実現上の課題

シミュレーションを利用したテスト環境を構築して利用することにより，以下のような効果が期待できる。

⑴ テスト環境が安価　　基本的には開発用の計算機上にソフトウェアとしてテスト用の環境を構築するため，試作機などハードウェア開発コストが非常に小さくなる。また，必要な台数だけ用意することができる。

⑵ 他ユニットの開発が遅れても，テストを進められる　　他のユニットがなくても，論理的なレベルのテストを進めることが可能になる。性能的な面についても，ある程度の予測を立てることができるようになる。

一方で，シミュレーションを利用したテスト環境の実現にあたっては，実際に（ソフトウェア）シミュレータを開発しなければならない。従来，シミュレーション利用のテストでは，これが障害となることが少なくなかった。このため当社では，テスト環境を構成するシミュレータをパターン化し，短期間でテスト環境が構築できるようにして，シミュレーションを利用したテストの実施を支援している。

このアプローチでは，テスト環境自体は製品種類ごとに，ほぼ汎用のものとして利用でき，テスト環境開発コストの大幅な削減が可能になる。

以下では，シミュレーションを利用したテストによってテストの効率化を実現した事例として，冷蔵庫テスト環境，携帯電話テスト環境の2例を紹介する。

3 事例 ― 冷蔵庫用テスト環境

3.1 冷蔵庫制御ソフトウェアのテストの特長

近年の冷蔵庫はユーザーの声を反映した様々な機能を備えており，その分だけ制御ソフトウェアも複雑化している。このため，冷蔵庫制御ソフトウェアの開発においてもテストが重要になっている。冷蔵庫制御ソフトウェアはコンプレッサをはじめとするハードウェアを，庫内温度や外気温などの様々の状態に応じてきめ細かに制御しなければならない。しかし，ハードウェアや庫内温度，外気温など環境のあらゆる状況を実際に作り出してテストすることは極めて難しい。このため，冷蔵庫テスト環境においては，開発用計算機で外界及びハードウェアをシミュレーションすることによって，制御用ソフトウェアの論理的なテストを行う。これにより，より多くのバリエーションのテストを可能にし，制御ソフトウェアの信頼性向上を実現している。

3.2 冷蔵庫用テスト環境

冷蔵庫制御ソフトウェア用テスト環境は，冷蔵庫システムの特徴を基に，外部環境シミュレータ，ハードウェアシミュレータ及び内部状態モニタから構成される。テスト環境全体の構成を図4に，また，ツールの外観を図5に示す。

⑴ 外部環境シミュレータ　　外界のシミュレータとして，スイッチ類をグラフィカルなインタフェースによる操作が可能な操作パネルGUI（Graphical User Interface）としてソフトウェア的に実現している。更に，温度変化や製氷皿の回転のシミュレーションも実現しており，実際に冷蔵庫として機能している状況と同様のテストが可能になっている。

⑵ ハードウェアシミュレータ　　ソフトウェアとハードウェアとは，RAM（Random Access Memory）を経由して情報のやり取りを行う。このRAMエリアを仮想的に開発用計算機上に構築して，ハードウェアのシミュレーションを実現し，上記の外界のシミュレータとも接続している。

⑶ 内部状態モニタ　　グラフィカルなモニタを用いて，ソフトウェアの内部状態やハードウェアの動作状態を見やすく表示することにより，動作確認などの作業効率を更に高いものにしている。

既に，当社の冷蔵庫制御ソフトウェアのテストでこの方式を採用しており，開発の早期段階から制御ソフトウェアをテスト・評価することが可能になっている。その結果，不具合の大幅な削減など，30％近い製品品質の向上に貢献している。

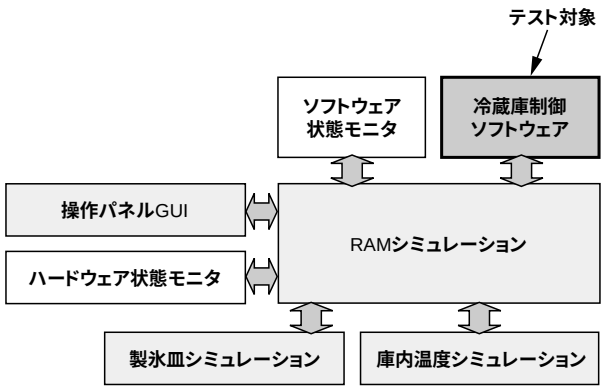


図4．冷蔵庫テスト環境の構成　冷蔵庫テスト環境では，外部状態などのシミュレータ及びシステムのRAMシミュレータなどを利用して，制御ソフトウェアのテストを実現している。
Testing environment for refrigerator control software

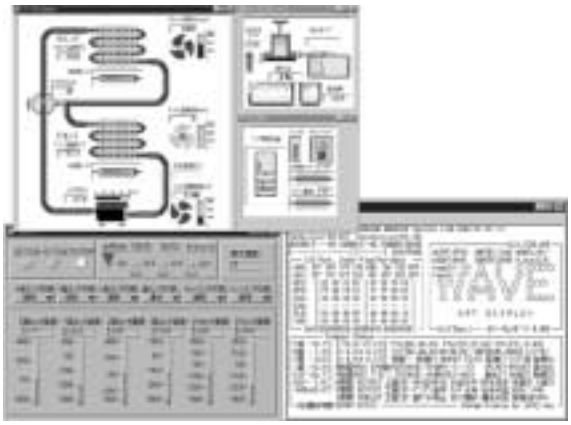


図5．冷蔵庫テスト環境の概観　冷蔵庫向けのシミュレーションテスト環境では，GUIによって冷蔵庫内部の動作や，外界の条件，ソフトウェア内部の状況をモニタできる。
Overview of refrigerator simulation testing environment

4 事例 ― 携帯電話テスト環境

4.1 携帯電話ソフトウェアのテストの特長

携帯電話は，通信機能をつかさどるプロトコル部と，通話処理や電話帳，メール機能などのサービスを提供するユーザーインタフェース部から構成されている。これらのソフトウェアは，いずれもリアルタイムOSを介してマイコン上で動作する。携帯電話の需要は急激に増加しており，製品の早期リリース実現に向けてシステムテストの効率化が重要になってきている。

4.2 携帯電話通信テスト環境

このため，当社ではシミュレータを用いて携帯電話通信プロトコル部を効率的にテストする環境を開発し，利用している。このテスト環境では，外界，OS，周辺タスクをシミュレーションし，制御用ソフトウェアの論理的な動作を開発用計算

機だけでテストすることを可能としている(図6)。

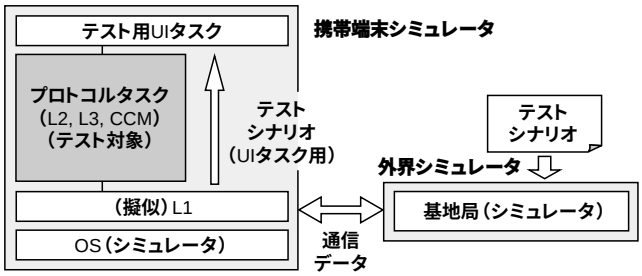
⑴ 外界シミュレーション　スクリプト形式のテストシナリオを用いて，携帯電話基地局の動作をシミュレーション実行する。携帯電話本体のプロトコルと基地局との間の通信データを模擬し，フィールドでのテストと同等の環境を整える役割を持つ。

⑵ OSシミュレータ　マイコン上のRTOS(Real Time OS)上で動作する携帯電話の制御ソフトウェアのテストを容易にするために，パソコン(PC)上にRTOSシミュレータ(RTOS-Sim)を用意し，その上で，携帯電話の制御ソフトウェアを動作させる方式を採用している。

実際のテストでは，RTOS-Sim上にテスト対象となる携帯電話のプロトコル部ソフトウェアを載せ，テストデータ投入用のテスト用UI(User Interface)部から操作データを投入する。また，基地局側との通信データはテストシナリオの形で投入し，携帯電話の一連の動作を論理的に検証することを可能としている。

テスト時の動作確認は，プロトコルタスクから出力される通信データを基地局シミュレータで表示し，テストケースから期待されるデータと一致していることを確認して行う。

このテスト環境を用いることにより，高価なテスト用基地局が不要になり，また開発の早期段階で通信プロトコルソフトウェアの試験ができるようになった。



L　：Layer
CCM：Communication Control Manager

図6．携帯電話テスト環境の構成　携帯電話のテストでは，OSや基地局などをシミュレータによって実現し，プロトコル部などのテストを行う。
Testing environment for cellular phone system

5 あとがき

シミュレーションを利用したテスト環境の概要とその事例を紹介した。このようなテスト環境を利用することにより，ソフトウェアのテストを開発の早い段階で行うことが可能になり，ソフトウェアの品質と開発効率の向上が実現できる。

今後は，更に多くの組込み機器開発への効果的な適用を目指して，活動を進めていきたい。　(岡安／岸本)

Webベース不具合管理システム PRISMY™

PRISMY™ Web-Based Problem Management System

特集❷

1 まえがき

不具合管理は，ソフトウェア開発のテスト工程で検出された不具合の確実なフォローと除去を保証する活動である。しかし，近年の開発形態の変化に伴い，従来の不具合管理の効率と効果の限界が表面化している。当社では新たな開発形態への適応に向けて，Webベースの不具合情報管理システムPRISMY™を開発し利用している。

2 不具合管理の課題

近年のソフトウェア開発は，多くの分散拠点で様々な機種を並行開発する形をとっており，不具合管理についても，効率とその効果を中心に，以下のような課題の解決が望まれている。

⑴ 効率性：マルチサイトの分散開発に対応できない
開発規模の急速な増大に伴い，社内外を問わず地理的に分散した環境(マルチサイト)での開発やテストが一般的になった。こうした分散開発環境では開発担当者への不具合情報のフィードバックに遅延が生じ，また管理者がタイムリーに状況把握することが困難になる。

⑵ 効果性：マルチバージョンの同時開発に対応できない
製品のtime to market短縮が強く求められるため，同じ製品系列の複数バージョンを同時開発する機会が増えている。また，同じソフトウェアIP(Intellectual Property)が様々な製品に組み込まれる機会も増えている。各バージョンでは出荷時期や製品戦略に違いがあり，同じ現象の不具合でも対応のタイミングや方法が異なってくる。このため，最終的にすべてのバージョンでもれなく対応が完了したことを確認することが難しくなっている。

3 PRISMY™による不具合管理の高度化

PRISMY™は次の三つの特長を持ち，これによってテストフェーズの不具合管理の高度化を実現している。

⑴ Webとe-mailによるマルチサイト開発への対応
⑵ 不具合データベース(DB)によるマルチバージョン不具合管理への対応
⑶ 不具合情報分析機能による効率的なフィードバック

3.1 マルチサイトにおける不具合管理の効率化

ある不具合が発見されてから最終的に修正の結果が確認されるまでに要する時間の多くは，次の作業を待つ滞留時

間であると言われている。特にマルチサイトの場合，各サイトから連絡された不具合情報を，窓口担当者が設計者に中継する部分で滞留が発生しやすい。

PRISMY$_{TM}$では，Webとe-mailを組み合わせることで不要な滞留時間の削減を実現している（図7）。

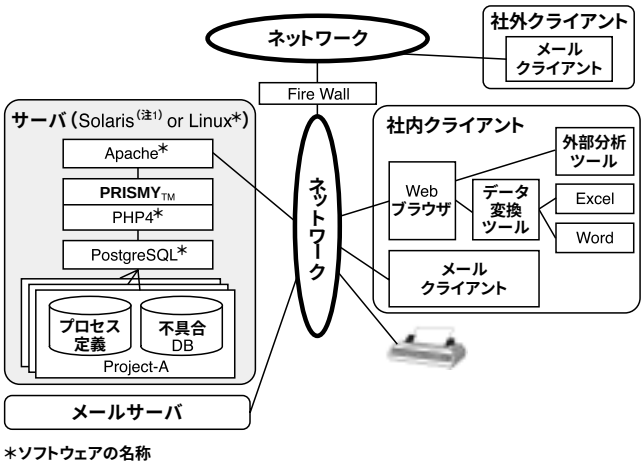


図7．PRISMY$_{TM}$のシステム構成　PRISMY$_{TM}$はWebやe-mailを活用して，社内外の様々な開発サイトの不具合情報を共有化する。
System configuration of PRISMY$_{TM}$

発見した不具合は，社内外のクライアントからWebブラウザ若しくはメールを利用してPRISMY$_{TM}$サーバに登録される。不具合が登録されると，あらかじめ指定されたメールアドレスに次の作業を促すメールが自動送信される。滞留時間の最短化は，この対応情報の登録と自動メールの送信を，不具合が決着するまで繰り返すことにより実現される。PRISMY$_{TM}$では，情報共有の手段としてWebとメールの二つの仕組みを利用し，社内外を問わず様々なサイトからの報告を自動的に情報共有することができる。

この結果，携帯電話ソフトウェアへの適用では，従来の不具合シートなどの紙を利用した管理と比較して，1件当たりの不具合対応時間は約60％短縮できた（図8）。

3.2　マルチバージョンに対応した効果的な管理

マルチバージョンに対応する不具合管理の難しさは，①プロジェクトをまたいだ情報共有を迅速・確実に実施しなければならない，②バージョン間での不具合への対応タイミングや修正方法の違いをもれなく管理する必要がある，の2点にある。特に，プロジェクトをまたがると管理主体があいまいになりやすく，もれのないフォローが難しくなる。この点を考慮すると，効果的な管理を実施するには，次の2点が重要になる。

⑴　バージョン間で不具合情報が効率的に交換できること（情報のexportとimport）

⑵　ある不具合の対応状況をバージョン間でもれなく追跡できること（トラッキング）

PRISMY$_{TM}$ではプロジェクトごとに不具合DBを作成するが，図9のように，不具合DB間でデータが交換できる仕組みを提供することで⑴を実現している。また，個別の不具合の対応状況が，各バージョン横並びで一覧し確認できるので，もれなく決着までフォローでき⑵を効率的に行える。

（注1）　Solarisは，米国SunMicrosystems社の商標。

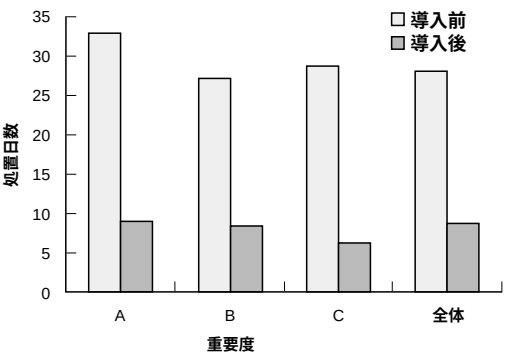


図8．不具合対応時間の改善　PRISMY$_{TM}$の導入によって，不具合対応に要する時間が格段に短縮された。
Improvement of troubleshooting cycle

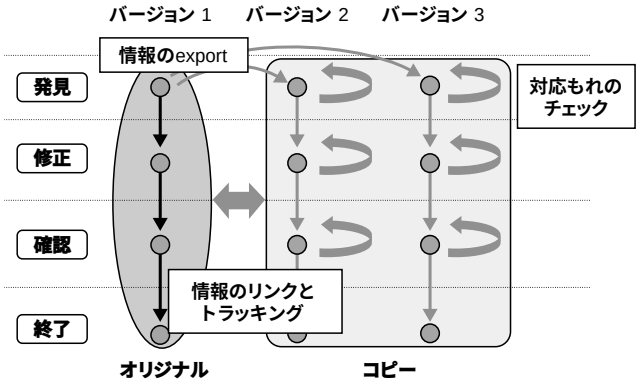


図9．マルチバージョン間の不具合管理　PRISMY$_{TM}$を利用することで，マルチバージョン間の不具合情報の円滑な処理も可能になる。
Fault management in multiversion software

3.3　不具合データの活用

不具合データを蓄積することにより，そのデータを利用した定量的な不具合管理が行える。PRISMY$_{TM}$は，Web上での多角的な不具合情報の集計や不具合予想曲線作成など，タイムリーな状況把握支援機能を備えている（図10）。また，集計では，例えば処置の進行状態や不具合の重要度をモジュールごとに確認するマトリクス分析により，信頼性が低いモジュールの発見や定量化，重点的レビューを可能にして

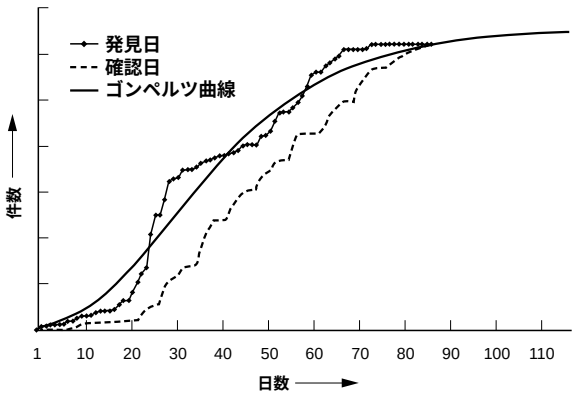


図10．不具合予測曲線　PRISMY$_{TM}$では不具合予測曲線と連動した不具合管理や，不具合収束時期判定などもできる。
Example of bug estimation curve

いる。また不具合予想曲線は，リリース時期の見積りやテスト戦略作成などに用いられている。重要なのは，デシジョンを行うプロジェクト管理者自身の手で分析が短時間で行える点である。従来は，不具合管理の責任者に分析を依頼してから結果を入手するまでに数日を要したため，フォローの頻度が低く，タイムリーな対策を打ち出しにくかった。PRISMY$_{TM}$によって，数分間で同等の分析が可能になり，定量的なフォローを毎週若しくは数日置きなど任意の頻度に設定できるようになっている。

4　部門のプロセスへの適合

マルチサイトやマルチバージョンといった複雑な形態の不具合管理にはツールは不可欠である。しかし，ツールが十分に効果を発揮するには，ツールは部門やプロジェクトにマッチした不具合管理プロセス（体制，帳票，フロー）を扱えなければならない。PRISMY$_{TM}$では図7に示したように，不具合管理のフローや帳票の形式などを，プロセス定義データとしてプロジェクトごとに個別に設定できる。そのため，非常に汎用性が高く，利用部門のニーズにマッチした不具合管理を提供できる。

5　あとがき

PRISMY$_{TM}$を使った不具合管理で，マルチサイトやマルチバージョンを扱う開発環境に対応した定量的な管理が実施できるようになった。今後は構成管理などと連携した，より効果的な不具合管理環境を構築していきたい。（藤巻／増岡）

文　献

(1) M.Hirayama, et al. "Generating Test Items for Checking Illegal behaviors in Software Testing". Proc. of 9th Asian Testing Symposium, Dec. 2000, IEEE.
(2) 平山雅之，ほか．“機能モジュールに対する優先度に基づいた選択的テスト手法の提案”．電子情報通信学会技術研究報告SS2001-6，p.1－8.
(3) Roger S. Pressman.（飯塚悦功監訳）実践ソフトウェア工学．日科技連出版，2000-10.


平山　雅之　HIRAYAMA Masayuki
研究開発センター　システム技術ラボラトリー主任研究員。ソフトウェアの信頼性保証，テスト・検証技術の研究に従事。情報処理学会会員。
System Engineering Lab.


植木　克彦　UEKI Katsuhiko
研究開発センター　システム技術ラボラトリー研究主務。ソフトウェアのテスト・デバッグ技術の研究に従事。情報処理学会会員。
System Engineering Lab.


宮本　隆一　MIYAMOTO Ryuichi
東芝デジタルメディアエンジニアリング（株）モバイルグループ チームマネージャ。
携帯電話端末ソフトウェア評価業務に従事。
Toshiba Digital Media Engineering Corp.


岡安　二郎　OKAYASU Jiro
研究開発センター　システム技術ラボラトリー研究主務。ソフトウェアの設計・テスト技術の研究に従事。情報処理学会会員。
System Engineering Lab.


岸本　卓也　KISHIMOTO Takuya
（株）オーイーシー　エンジニアリング事業本部 開発・設計第1グループグループ長。
家電機器IT化の中心となるホーム端末の開発に従事。
OEC Corp.


藤巻　　昇　FUJIMAKI Noboru
研究開発センター　システム技術ラボラトリー研究主務。ソフトウェア開発プロセスの改善技術の研究に従事。情報処理学会会員。
System Engineering Lab.


増岡　範雄　MASUOKA Norio
モバイルコミュニケーション社 日野モバイル工場 モバイルソフトウェア設計部主務。移動通信端末ソフトウェアの技術企画・技術管理業務に従事。
Hino Operations－Mobile Communications