ソフトウェアのテスト管理システム

Test Management System for Effective Software Development

河村　透　　小笠原　秀人
■KAWAMURA Toru　■OGASAWARA Hideto

ソフトウェアの開発において，最終工程となるテスト工程のいっそうの効率化が求められている。高機能化及び複雑化に伴ってソフトウェアの規模やテストすべき機能が増える一方で，ソフトウェアの開発期間は更なる短縮を迫られる状況にあり，テスト工程は二律背反の状況に置かれている。このような状況下では，テスト対象の品質状況を把握して，実施するテストの数，種類，期間，及びテストの結果を正確に管理することが必要である。

東芝は，ソフトウェアのテスト管理作業を効率化するために，テストの項目と結果を一元管理し，管理するうえで注意すべきポイントを可視化する機能を持つテスト管理システムを開発した。

The increasing complexity and sophistication of software in recent years has given rise to a need for more effective software development. This is particularly true in the testing process, the final stage of software development. While the number of items to be tested has increased with the expansion of large-scale software systems, software development periods have been becoming shorter year by year. In this contradictory situation, it is necessary to comprehensively understand the quality of software undergoing testing through appropriate management of the testing process by monitoring the numbers, categories, schedule, and results of the tests.

To improve the efficiency of test management of software, Toshiba has developed a test management system incorporating several functions to both unify the management of test items and visualize warning points requiring attention.

1　まえがき

近年，大規模化かつ複雑化するソフトウェア開発において，テスト工程の質の良否が，製品の品質を左右する重要な鍵の一つになっている。テスト工程では，ソフトウェアの規模に応じてテスト担当者の人数が増加してくるとテストマネージャーが必要になり，テストチームを運営するうえでテストを管理することが求められる。テスト管理の内容として，主に次の四つが挙げられる。

(1)　テストスケジュールの管理　　あらかじめ実施するテストの数，種類，実施期間，及び担当者を決め，テスト実施中はその進捗やテストの合格状況を収集し，期間内に目標品質に到達するかを判断する。

(2)　不合格テストの管理　　テストして不合格になった項目に対し，その不合格数の傾向や，不合格テストの期日までの修正状況を確認する。不合格テストの内容の分析や修正の管理は，専用の管理システムを利用して，効率的に実施する。

(3)　テスト項目の管理　　市場で不具合が発見されることを未然に防ぐため，テスト自体の品質を高めるようにテスト項目を管理する。ここでは機能や要件を基に抽出したテスト項目を保管するとともに，テストの種類（機能テスト，ストレステスト，回帰テストなど）や対象となる機能が網羅されているかを考慮し，内容が妥当か精査する。テスト結果によってはテスト項目の内容修正や追加を検討する。

(4)　テスト結果の管理　　実施したテスト数や検出した不合格数に応じて今後の対策を検討する。不合格箇所や出現頻度と，テスト対象に追加，修正された機能の範囲を照らし合わせて，テスト対象の品質を補強する部分を確認する。またソースコードの規模とテストの総数の比率から，テスト数が妥当かを検証する。

ここでは，ソフトウェアのテスト管理を効率的に実施するため，東芝が開発しているテスト管理システムの概要と活用方法について述べる。

2　ソフトウェアのテスト工程における課題

高機能化が進む組込みシステムの開発では，高い品質が求められる。そのためには，十分にテストを遂行し，検出した不具合には早期に対策を施して，期間内の品質確保に努める必要がある。これらを踏まえてテスト管理を行ううえでの課題として，次の3点が挙げられる。

(1)　データ集計作業の負荷増加と問題の表面化の遅れ

一般的なテスト管理手段として，テスト担当者が実施した結果を帳票ツールに記録する方法がある。記録されたテスト結果は，分析のためにテストマネージャーが集計し，

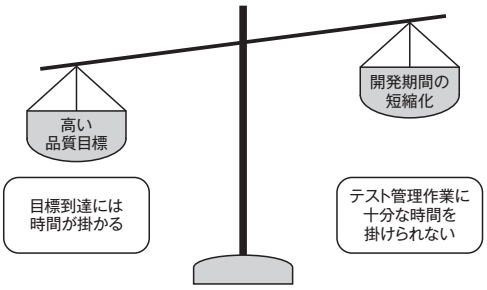


図1．組込みシステムの高機能化と短納期のバランス ― システムの高機能化と短納期の状況下にあるテスト工程では，いかに効率よくテストをこなしていくかが製品品質の鍵となる。

Balance between quality and short delivery time

グラフ化などによって，品質や進捗を確認する。高品質が求められる組込みシステムの開発では，問題をいち早く表面化させるためにデータの集計作業は欠かさず実施する必要がある。

しかし近年の短期開発において，テスト期間も同様に短縮を迫られており，そのような状況下ではテスト結果の記録や保管が主となってしまい，テスト実施中に集計や分析を頻繁に行うことが難しい。また，品質データの集計は，対象の規模とテスト担当者の人数の増加に応じて手間が増える傾向にある。テストマネージャーにとって，これらのデータを集計，分類し，分析することが大きな負担になる。

テスト結果を適時にかつ適切に集計，分析できないと，開発途中に対象の品質を把握できず，問題の表面化の遅れにつながる。品質と期間のバランスを取りながらよりいっそうのテスト効率化を図るため，効率的なテスト管理が望まれている（**図1**）。

(2) テスト状況の分析の不備　　実際のテスト管理作業では，しばしばテスト状況をうまく分析できていない場合がある。テストマネージャーのスキルによっては，目的が不明確なまま，ただテスト結果の合計値を評価している場合も少なくない。個人のスキルに左右されないテスト管理が実施できる仕組みが求められている。

(3) 不明確なテストの終了判定　　ソフトウェア開発では，新規に機能を追加したり，既存機能を修正したりするのに伴って，それまで正常であった機能が動作しなくなる“機能デグレード”がしばしば発生する。一度テストを行ってもテストが完了しない状況が起こりうるため，テストを再び行うリグレッションテストが必要になる。しかし短期間のテストでは，リグレッションテストを行う回数には限度がある。どの程度行うべきかを判断するには，データの分析によって品質が安定しているか否かを判断することが求められる。

3　テスト管理システムの開発

2章で述べたように，品質や，期間の面で切迫したテスト工程では，テスト項目の内容や各テスト担当者からのテスト結果などを一元的に把握して，適時かつ適切なテストを実施できる仕組みが求められる。近年，テスト関係者の負担を減らすものとして，テスト項目とテスト結果を一元管理するテスト管理システムが活用され始めている。

当社は，テスト工程の更なる効率化を目指したテスト管理システムを開発している。このシステムは，テスト情報であるテスト項目とテスト結果を専用サーバで一元管理する。テスト担当者はWebブラウザを経由して，サーバにあるテスト情報の入力や閲覧がいつでも可能になる。

3.1　テスト管理システムの機能

当社が開発したテスト管理システムは，次のような機能を持っている。

3.1.1　テスト項目とテスト結果の保管と共有　　テスト項目とテスト結果は専用サーバに登録して保管する。データの入力にはWebからの直接入力だけでなく，表計算ソフトウェアのデータを一括して入力する方法も用意しており，今まで帳票ツールを使ってきたユーザーでも容易に利用できる。入力された内容はシステムに即時に反映され，Webブラウザからいつでも確認できる（**図2**）。テスト担当者及び開発者が同じデータを参照して，実施したテストの成果やテスト項目の内容をレビューすることができる。テスト項目はフォルダツリー上に区分けして管理することが可能であり，モジュールや開発・テストチーム単位で管理できる。

3.1.2　品質グラフの表示　　システムに蓄積されたデータを元に，不合格になったテストの検出状況（**図3**）や，テストの実施計画と進捗をグラフで即時に確認できる。グラフはフォルダツリーの各フォルダ単位で集計が可能であり，フォルダ構成をモジュールや開発・テストチーム単位で構成すれば，各々のテスト状況を確認できる。

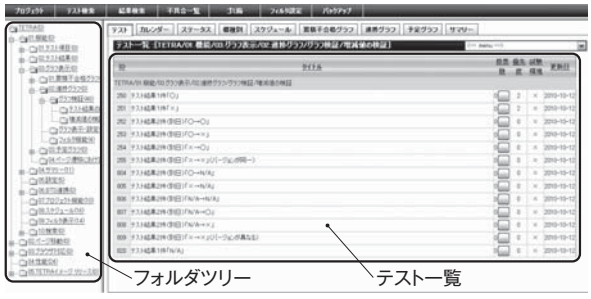
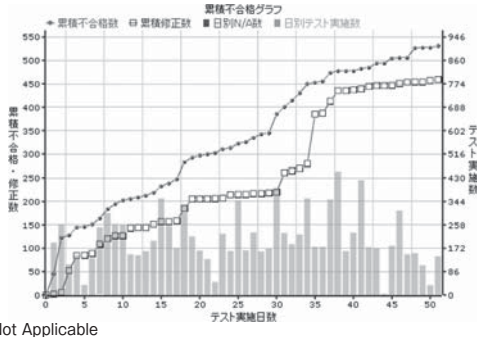


図2．テスト管理システムの画面例 ― 入力された内容は即時に反映されるので関係者間で共有できる。全てのテスト項目とテスト結果のフォルダはツリー状に配置され，テスト全体を把握できる。

Example of test management system display

一般論文



N/A：Not Applicable

図3．テスト不合格数の累積グラフ ― 不合格となったテストの数を日ごとに集計し，テスト実施数と不合格テストの出現具合を確認できる。入力したテスト結果は即時に反映される。

Stacked chart of test faults

3.1.3　テスト成績書の生成

一般的に，テストの結果は成績書にまとめて保管されることが多い。このシステムでは，蓄積されたテスト情報から自動的に成績書を作成する機能があり，成績書のテンプレートを事前に用意しておけば，システム内のデータを任意の成績書に記載することも可能である。

図4に示すように，従来の帳票ツールを用いたテスト管理と比べ，このシステムではテスト結果を一元管理し，集計の手間が発生しない。また，システムから最新のテスト状況をいつでも参照できるので，現状をいち早く把握し，迅速な対応を取りやすいメリットがある。

3.2　テスト管理システムによる問題の発見

開発したテスト管理システムは，テスト工程で活用できる専用

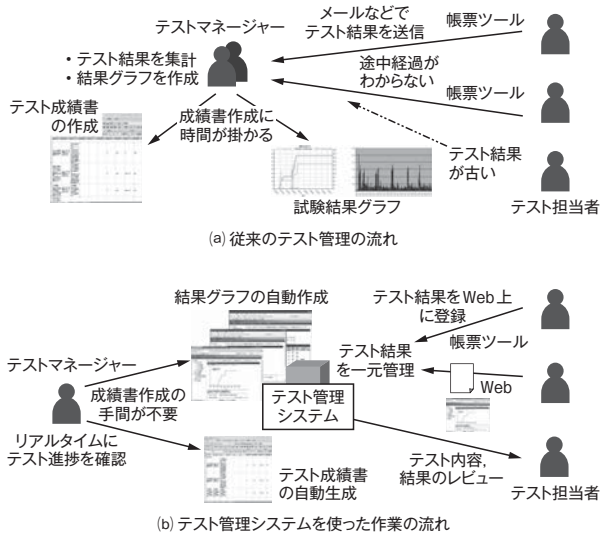


図4．テスト管理システムと従来のテスト管理の比較 ― 従来の帳票ツールによるテスト管理と比べ，システムに直接テスト結果を登録するため，集計の手間が発生しない。

Comparison of test management system and conventional test management method

グラフを用意している。これらのグラフは，テスト結果を登録するだけで即時に表示でき，次に示すような注意すべきポイントを可視化することによって効率的なテスト管理を可能にする。

3.2.1　問題箇所の可視化

ソフトウェアの不具合には，一部の要因に伴う不具合が80％を占めるという“パレートの法則”と呼ばれる特性がある。これを踏まえ，このシステムではツリー上のフォルダごとに不合格数を集計しグラフ化することで，不合格の割合で大半を占める箇所を確認できる（**図5**）。

また，不具合の出現状況の判別も今後の対策のうえで重要になる。このシステムではテスト進捗を示すグラフで，不具合の修正状況を，その後のテスト結果を元に確認することができる。この修正状況の推移から，修正に時間が掛かっている箇所や品質が不安定な箇所を特定できる。

例えば，**図6**(a)に示すテスト実施日とその不合格状況のグラフを確認すると，数日前の変更により，それまで合格になっていたテスト項目が不合格になって現れている。直前の影響範囲を確認し対応することで，テストの合格率は再び向上すると予測される。一方，図6(b)では，日ごとのテスト実施数を示す棒グラフから，テストは継続的に実施されているが合格率は上がらない状況が続いている。この場合は図6(a)のような一過性の不具合と異なり，すぐにテストの合格率を向上させることが難しく，ソフトウェア設計や，コーディングの中身，テスト内容などを精査する必要がある。

3.2.2　機能デグレードの可視化

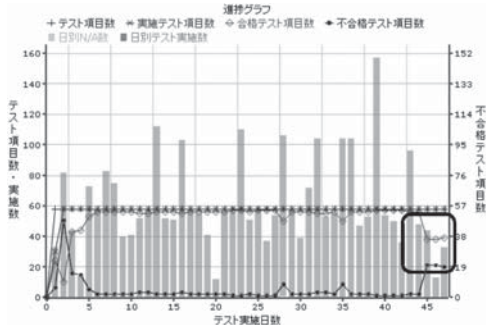
不具合修正に伴う機能デグレードに対しては，リグレッションテストの実施管理が必要になる。このシステムではリグレッションテストの管理機能として，複数回のテスト結果の入力と，機能デグレードを可視化する機能を備えている。各テスト項目とテスト実施日の表からテスト結果を入力することで，同じテスト項目に対し，複数回実施したリグレッションテストの結果が閲覧できる。

図7のように機能デグレードが起きている箇所がわかるよう

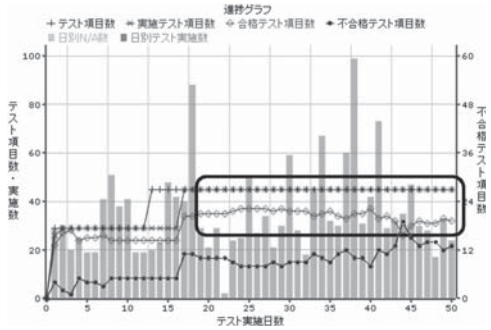
フォルダ名	テスト項目数	テスト進捗 未実施テスト項目数	不合格テスト項目数	合格テスト項目数	N/Aテスト項目数	進捗グラフ	テスト実施結果 テスト実施数	不合格検出数	デグレード検出数
/	0	0	0	0	0		0	0	0
/1.テスト一覧表示	45	0	13	32	0		1762	75	37
/10.サマリー	10	0	0	10	0		402	10	8
/11.その他	6	0	1	5	0		144	12	9
/2.テスト切り替え表示	2	0	0	2	0		39	0	0
/3.カレンダー表示	3	0	0	3	0		49	8	2
/4.ステータス表示	2	0	0	2	0		39	0	0
/5.種別表示	58	0	19	39	0		2881	128	105
/6.スケジュール表示	16	0	1	15	0		501	16	10
/7.累積不合格グラフ	6	0	1	5	0		91	53	1
/8.進捗グラフ	20	0	2	18	0		174	13	1
/9.予定グラフ	77	0	4	73	0		4457	216	27
合計	245 件	0 件	41 件	204 件	0 件		10539 回	531 回	200 回

図5．実施テストと不合格数の可視化 ― 不具合の検出状況を棒グラフ化して一覧表示する。不合格になったテストが集中している箇所を確認できる。

Numerical visualization of executed tests and faults



(a) 急性的な不合格数の増加



(b) 慢性的な不合格発生の高止まり

図6. 不合格テスト項目数の推移 — 合格テスト項目数の推移を確認すると，(a)では合格数が直前に急激に減少しているが，一過性の問題である可能性がある。これに対し(b)では，合格数は長い間向上していないことから仕様，設計，コーディング内容，テスト内容などを根本的に見直す必要がある。

Transitional changes of test faults

図7. テスト結果と機能デグレードの確認 — 実施したテスト項目と日付の表からテスト結果が確認できる。機能デグレードが発生していると，テスト結果欄に表示される。

Test results display for checking degraded functions

に表示し，該当するテスト項目のデグレード回数を一覧にして**表1**のように表示することもできる。一覧のテスト項目と不具合修正・機能追加状況を照らし合わせ，影響を受けやすいテスト項目を確認し，次回のリグレッションテスト実施方針の検討に役だてることができる。

3.2.3 信頼度成長曲線による品質安定傾向の可視化

テスト結果に対して，品質が十分と言える状態を定義することは難しい。質の良いテストを十分な数だけ実施したかは感覚的にもつかみづらいが，記録したテスト結果を元に不具合が再発する可能性があるかを統計的に可視化する方法がある。再発確率を計算し，安定の可否を判断する統計的手法として，このシステムではゴンペルツ曲線などの“信頼度成長曲線[1]”を用いる。過去のテスト結果を元に，今後何日間でどのくらいの量の不具合が出現するかを，その軌跡から予想できる（**図8**）。

一見して，不具合のないモジュールが，何かの修正タイミングで不具合が起きる可能性があり，その頻度が多いと信頼度推定曲線は発散した弧を描き，不具合の発生が収束していない状況であることがわかる。この状況を回避するには，開発の早い段階からリグレッションテストを実施したり，新たなテスト項目を実施したりして，継続して機能デグレードが起きていないことを確認し，品質の安定に努めることが重要である。

3.2.4 テストの予定と進捗確認

短期間で効率的なテストを実施するためには，事前にテスト計画を作成し，テストを計画どおり着実に実施していく必要がある。このシステムではテスト計画を立案するとともに，計画と実施状況を照らし合わせてグラフ化する機能を持っている（**図9**）。

テストの進捗管理では，テスト項目の実施が計画よりもはるかに進んでいる状況が必ずしも良好とは言えない。テストを途中で省略したり，実施方法を誤ったりしている可能性もあ

表1. 不合格や機能デグレードの多いテストト項目

Test items with numerous faults and degraded functions

No.	ID	タイトル	フォルダ	テスト実施数	不合格検出数	デグレード検出数	最新結果
1	200	所定のフォーマットにて記入されたcsvシートを登録する	TETRA/01.機能/01.テスト項目/ファイル入力/データ登録	10	6	1	○
2	176	テストをコピーする(異なるフォルダに)	TETRA/01.機能/01.テスト項目/Web入力/テスト操作	15	4	3	○
3	26	テストフォルダのコピーをする	TETRA/01.機能/01.テスト項目/Web入力/フォルダ操作	15	4	3	○
4	23	テストを削除する	TETRA/01.機能/01.テスト項目/Web入力/テスト操作	22	4	2	○
5	716	スケジュールを更新する(開始日・終了日・担当者・バージョンの変更)	TETRA/01.機能/06.スケジュール	6	4	1	○
6	199	テストの入力文字数を多くして，テスト項目を入力する	TETRA/01.機能/01.テスト項目/Web入力/テスト登録	12	3	2	○
7	497	シングルクォート(')を含んだテスト項目	TETRA/01.機能/01.テスト項目/Web入力/テスト登録	10	3	2	○
8	516	テスト項目にフィルタをかけた状態でグラフを描画する	TETRA/01.機能/03.グラフ表示/02.進捗グラフ/フィルタ機能	9	3	2	○
9	198	特殊文字の入ったデータをcsvシートで出力する	TETRA/01.機能/01.テスト項目/ファイル入力/データ取得	9	3	2	○
10	201	所定のフォーマットにて記入されたcsvシートで更新する	TETRA/01.機能/01.テスト項目/ファイル入力/データ登録	8	3	2	○

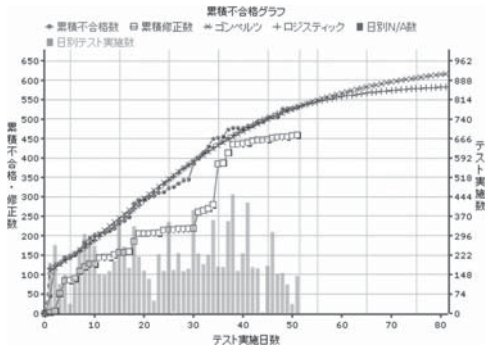


図8. 信頼度成長曲線の例 — 不具合の検出頻度を統計的に分析して，今後の不具合の検出傾向を予測する。

Example of software reliability growth curves

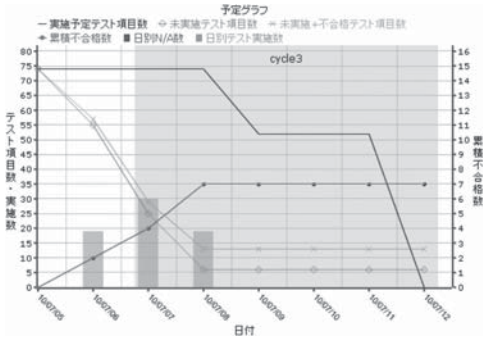
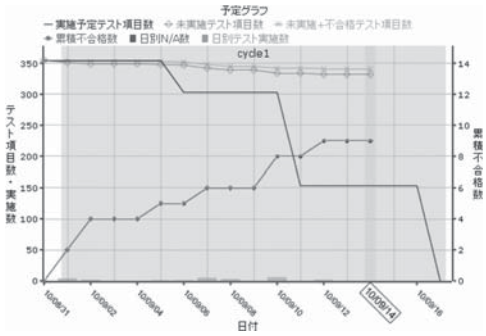


図9. テストの予定と実施状況のグラフ化 — 予定と実施の乖離（かい り）と，不具合の検出状況から，テストが想定どおり実施されているか推考する。

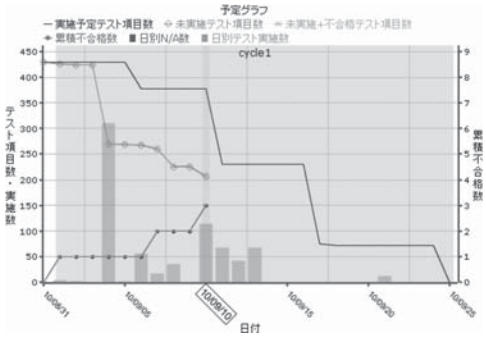
Example of test schedule and status chart

る。特に気を付けなければならないのは，次に述べるテスト予定と実績のパターンである。

図10(a)のパターンは，テスト項目の実施が予定より遅れ，不合格になるテスト項目が継続的に出現している状況である。テスト項目の実施で苦労している要因には，修正による影響の広さ，仕様の漏れ，根本的な処理方式の不良などが挙げられ，



(a) テスト項目の消化が予定より遅れているパターン



(b) テストの実施が予定よりはるかに進んでいるパターン

図10. テスト計画で注意すべきグラフの例 — (a)では，テストの進捗が遅く，不合格になったテスト項目数が予定を超えており，対象の品質に問題がある状況とわかる。(b)では，進捗も良く，不合格になったテスト項目数も想定より少ない。対象の品質が良い場合もあるが，テストの内容が適切でないため不具合を検出できていない可能性がある。

Examples of test schedule warning patterns

断続的にテストが不合格になり，関連するテスト項目が実施できないなどが考えられる。

一方，図10(b)のパターンでは，テスト項目の実施は予定よりもはるかに進み，不合格になるテスト項目も少ない。これは，テスト対象の品質が高い可能性もあるが，開発の序盤で実施したテスト項目が洗練されておらず，適切に問題を検出できていない可能性も考えられる。

3.3 適用効果

テスト管理システムを利用することで，テストマネージャーは従来よりも集計作業の負荷を減らして適切にテストの実施状況を把握できるとともに，曖昧（あいまい）だったテスト終了判定も信頼度成長曲線を用いて品質が安定しているか判断できるようになる。また，このシステムが提供する各グラフは，必要最小限のテスト結果情報を用いて表示できるため，テスト担当者に余計な負担を増やすことはない。

このシステムを適用した開発プロジェクトでは，3.2節で述べた問題のポイントの早期発見に役だてるとともに，テスト結果を記録するテスト管理プロセスの確立や，開発メンバー，テスト担当者などを交えたテスト項目の内容のレビューにも効果を上げている。今後はテスト管理の効率化の程度を定量的に測定し，短期間でのソフトウェア品質の安定にどの程度寄与できるかを確認する予定である。

4 あとがき

ここでは，ソフトウェアのテスト管理に必要な作業について触れ，作業の効率化のために当社が開発したテスト管理システムと，その活用方法について述べた。

テスト工程はソフトウェアの品質向上にとって最後のとりでと言われており，テスト工程の良否が製品の品質を左右する。作業の効率化とともに，品質向上及びテスト効率化の両面でテスト管理システムの適用を推進していく。

文 献

(1) 山田 茂. ソフトウェア信頼性モデル — 基礎と応用. 東京，日科技連出版，1994，202p.

河村 透 KAWAMURA Toru
ソフトウェア技術センター プロセス・品質技術開発担当主務。ソフトウェアテスト技術の開発に従事。
Process and Quality Technology Group

小笠原 秀人 OGASAWARA Hideto
ソフトウェア技術センター プロセス・品質技術開発担当主査。ソフトウェアテスト技術の開発に従事。IEEE，情報処理学会，電子情報通信学会，プロジェクトマネジメント学会会員。
Process and Quality Technology Group