ソフトウェアテストシンポジウム 2006
JaSST'06: Japan Symposium on Software Testing 2006

Web システムテスト・ルネッサンス
－高品質と短納期の両立へ向けて－

林田 智也† 小林 敏幸‡

†横河情報システムズ株式会社 カスタマーサポート部 〒180-8750 東京都武蔵野市中町 2-9-32
‡株式会社 間組 経営企画本部 企画部 情報システム室 〒105-8479 東京都港区虎ノ門 2-2-5
E-mail: †tomoya.hayashida@jp.yokogawa.com, ‡koba@hazama.co.jp

あらまし 本研究は，富士通株式会社のユーザ会であるリーディングエッジシステム研究会（以下，LS 研）にて，2004 年 4 月～2005 年 3 月までの期間において研究した内容をまとめなおしたものである．

Web システムにおけるテストについて，納期を意識するあまり省略され，品質が低下している現状から，「高品質」と「短納期」を両立させることを目的として，「設計のテストとして有効なレビュー（手戻りの抑制）」と，「テスト工程の標準化」という 2 つのアプローチからテストを見直し，設計のテストに有用な「レビューガイドライン」と，テスト実施に有用な「テスト実施ハンドブック」「テスト項目抽出ツール」を作成した．

キーワード Web システム，レビュー，アプリケーションテスト，インフラテスト

Test Renaissance of Web Application Systems
－for Coexisting of "High quality" and "Short delivery date"－

Tomoya Hayashida† Toshiyuki Kobayashi‡

† Customer Support Dept, Yokogawa Information Systems Corporation

2-9-32 Nakamachi, Musashino-shi, Tokyo 180-8750, Japan

‡ Data Processing & Information System Dept, Hazama Corporation

2-2-5 Toranomon, Minato-ku, Tokyo 105-8479, Japan

E-mail: †tomoya.hayashida@jp.yokogawa.com, ‡koba@hazama.co.jp

Abstract The test of Web application systems is being omitted because of considering the delivery date. And the quality of the Web application systems has been decreasing by omitting the test. In such a current state, we reviewed the test of Web application systems from two approaches for coexisting of "High quality" and "Short delivery date". Those are "Review which is effective as the test of the design (control of the return in the process) " and "Standardization of the test process".

As a result, we developed "Review guideline", "Test execution handbook" and "Test item extraction tool". In addition to this, we executed the questionnaire about those outputs. From the result of that, we conclude that "Review guideline" is useful for the test of the design, and "Test execution handbook", "Test item extraction tool" are useful for the test execution.

Keyword Web application systems, Review, Test of Web application systems, Infrastructure test

1. 研究の背景

近年，ビジネスの多様化にともない，Web システムのような，「利用者」「場所」「時間」を問わない利便性の高いシステムが主流となりつつある．また，社会変化の速度が早まる中で，ビジネスで勝ち残るためには，意思決定のスピード向上は必要不可欠な要素である．これは，企業経営の根幹を担う「情報」を取り扱う企業情報システム開発においても同様であり，「高品質」「低コスト」に加えて「短納期」であることが当然のように求められている．

そのような現状において，開発プロセスについては多様化が進む一方，品質保証面での確認作業であるテストについては，納期厳守を最優先とするあまり，レビューに代表されるような静的なものや通常テストと呼ばれる動的なものが軽視，省略されがちである．その結果，本来意図していた品質を満たせず，「カットオーバー遅延」「費用増加」「システム稼動後のトラブル発生」という悪循環が生まれている．

こうした現状を打開するために「テストを確実に実施しながらも，短納期への希求に応えるにはどうしたら良いか」という問題意識の元に研究をおこなった．

2. 研究のアプローチ

研究に当たり，品質向上とともに Web システム開発に強く求められがちな「短納期」という観点に着目し，「設計のテストとして有効なレビュー（手戻りの抑制)」と，「テスト工程の標準化(テストの無駄を省く)」との 2 つのアプローチからグループ分けをし，各々の現状の課題と原因を探り，その解決策を研究することとした．これは，その解決策が品質の保証と納期を守ること（短期間での開発）を可能にし，Web システムのテスト方法の確立につながると考えたためである（表 1)．

また，LS 研が企業ユーザの集まりであることから，単なる研究で終わることなく，実業務に適用可能な成果物を作成することを最終目的とした．

表 1　各グループの研究目的

	レビュー
研究観点	「設計のテストとして有効なレビュー(手戻りの抑制)」
研究目的	レビューの手法を整理し、重要性を再認識する。
	レビューの指針となるノウハウをまとめ、内容の標準化を図る(属人化を抑制する)。

	テスト
研究観点	「テスト工程の標準化」
研究目的	品質を落とすことなく、テスト工程の短縮を実現する。
	テストのノウハウをまとめ、内容の標準化・充実を図る(属人化を抑制する)。

3. 研究成果

3.1. 手戻りの抑制を実現する「レビューガイドライン」

3.1.1. レビューにおける問題認識

上流工程における要件の漏れ，仕様の欠陥が「手戻り」やテストのやり直しにつながり，「短納期」の実現を阻害する要因になる．また，Web システムの品質を大きく左右することにもなり得る．

そこで，まずレビューを「ソフトウェアの品質を高めるために，客観的な視点を交えて，開発成果物の記載内容を確認し，改善の項目を探す作業」，すなわち要件定義の工程も含めた「設計のテスト」と定義した．

その上で，研究メンバ各社においてレビューが有効に機能できていないことによると思われる，開発途中での要件の漏れや仕様の欠陥による「手戻り」が発生している事例から，Web システム開発におけるレビューの 5 つの課題を抽出した（表 2)．

3.1.2. レビューにおける研究の成果

前述の 5 つの課題を解決し，レビューを有効に機能させることが，開発途中での要件の漏れ，仕様の欠陥による「手戻り」の抑制につながる．このため，課題の解決策を以下の 2 つとした．

①レビューの手法を整理し，重要性を再認識する

②レビューの指針となるノウハウをまとめ，内容の標準化を図る（属人化を抑制する）

・業務要件以外の要件漏れを抑制する

・Web システムで重要となる要件を整理する

具体的には，「ピアレビュー」を中心にレビュー手法をまとめ（10 種)，目標に応じてレビュー手法を選定できるように体系化をおこなった．あわせて，レビュー時にレビュイ・レビュアの双方が遵守するべき 16 種の「レビューの原則」と，レビュー実施時に発生する 21 種の症例（トラブル）と 67 種の解決策を「レビューに関するトラブルシューティング」としてまとめた．

また，開発の上流工程におけるレビューのノウハウとして，要件定義段階では要件を業務要件と業務運用要件の 2 つに分類して定義した（表 3)．

その上で，確認項目をシステム間で共有することができる業務運用要件を中心に，過去の経験やトラブル事例から 203 の確認項目を「要件チェックシート」としてまとめ，レビュー内容の標準化を図った（表 4)．

加えて設計段階でも同様に，設計の各工程におけるレビューポイントを整理し，実際におこなっている確認作業を洗い上げ，259 項目に集約化した「設計チェックシート」を作成した（表 5)．各々のチェックシートは Web システム開発で重要となる確認項目について，特に注意を促すようにしている．

実際にメンバ内で「要件チェックシート」を使用し，既存の要件定義書のチェックをおこなった．システムにもよるが，概ね **30～40 項目**について記載漏れが発見された．記載漏れが「手戻り」つながり，開発スケジュールに大きく影響を与えていた可能性もあることから，有効性を確認できた．

これらの研究成果を含めて，最終的に手戻りの抑制を実現する「レビューガイドライン」としてまとめた．

表 2　レビューの 5 つの課題

No.	課題
1	仕様の漏れを抑えきれていない
2	効果的におこなわれていない
3	納期との関係から省略されている
4	形式化している （承認するだけの場になっている）
5	成果物の仕上げ工程になっている （自己レビューができていなく、修正議論の場になっている）

表 3　要件の分類

分類	内容
業務要件	エンドユーザのシステム要件そのもの
業務運用要件 (非業務要件)	業務要件を満たすためにシステムが確保すべき要件

表 4　要件チェックシート

大分類	中分類	小分類	確認項目	要件	Webで重要な項目	実施者		
						ユーザ	ベンダ	共同
性能要件	レスポンス	アプリ	画面切り替え時の許容レスポンスタイムを確認する。		○			
		サーバ	サーバへの更新、問合せ、ファイルダウンロード時の許容レスポンスタイムを確認する。		○			
		処理時間	バッチの処理時間(日次、週次、月次、年次)を確認する。					
			処理開始の					
		システム負荷	最大同時ク ンスタイムを確認		○			
		タイムアウト	タイムアウトの設定時間を確認する。		○			
	アクセス	2重アクセス	複数端末からの同一人物アクセスの可否を確認する。		○			
			同一端末からの複数人物アクセスの可否を確認する。		○			
		最大ユーザ数	最大利用ユーザ、将来の想定最大利用ユーザ数を確認する。		○			
		アクセス制限	アクセスコントロール(人数制限)の実施可否ならびに人数を確認する。		○			
セキュリティ要件	アクセス管理	権限管理	システム利用権限ルールの有無及び内容を確認する。		○			
		物理的アクセス	対象機器が他と明確に分類された場所に設置されているかを確認する。		○			
			設置部屋への入退室管理の確認をする。		○			
		リモートログオン	リモートログオン実装の有無確認をする。		○			
			利用ツールの制限(特定)や、操作端末の特定を行う。		○			
		不正アクセス	不正アクセスの監視有無、および連絡先、連絡方法を確認する。		○			
	なりすまし対策	画面ロック(端末)	実装の有無、実装方法、ロックをかけるまでの許容時間を確認する。		○			
	ID	構成要素	組合せルール(アルファベット、数字、記号、桁数)を確認する。		○			
		形態	ユーザのID保有形態(固有ID or 共有ID)を確認する。		○			
		失効処理方法	ユーザID失効時の処理(無効 or 削除)を確認する。		○			
		管理	ユーザ管理者の特定、管理IDを含めたID管理ルールを確認する。		○			
	パスワード	構成要素	組合せルール(アルファベット、数字、記号、桁数)を確認する。		○			
		有効期限	利用可能期間を確認する。		○			
		履歴	過去のパスワード履歴の管理実行有無を確認する。		○			
		管理	パスワード管理ルールの確認を行う。		○			
	認証	方法	認証実施の有無や認証方法(ID＋PW、バイオメトリックス等)の確認をする。		○			
		タイミング	どのタイミングで認証(OS起動 or アプリ起動)を行うか確認する。		○			
	ログ	収集実施	ログ収集を行うかどうかを対象(アプリ、ハード等)とともに確認する。		○			
		種類	アクセスログ、エラーログなどの種類を確認する。		○			
		内容	ログとして残すべき内容(ID，時間、結果、変更内容)の確認をする。		○			
		容量	ログを残しておく容量を確認する。		○			
		保有期間	ログを残しておく期間を確認する。		○			
		バックアップ	バックアップメディア種別を確認する。		○			
		ログの取り扱い	容量オーバー時のログの上書きを認めるか、削除するか確認する。		○			
		収集方法の変更	ログの変更、ログ収集停止を認めるか確認する。		○			
		監視	ログ監視の実施有無、および通知方法、通知先を確認する。		○			
		ツール	ログ解析ツールの要否について確認する。		○			
	ウィルス対策	ウィルス対策ソフト	ユーザ側で導入が義務付けられているかどうか確認する。		○			
		定義ファイル更新	定義ファイルの更新方法およびタイミングの確認をする。		○			
		定期スキャン	定期スキャンの実施可否およびタイミングを確認する。		○			
		検知	ウィルスを検知した場合の検知方法、通知先、利用ツールを確認する。		○			
		対策方法	ウィルスを検知した場合の対処方法を確認する。		○			

Webシステム開発で重要となる項目を整理

表５　設計チェックシート

大分類	中分類	小分類	確認項目	確認結果	SLA確認	Webで重要な項目	対象ドキュメント例	実施工程		
								UI	SS	PS-PG
方式	方式設計	インフラ	方式設計書の内容を確認する。				方式設計書	○		
		セキュリティ	セキュリティポリシーを確認する。			○	セキュリティポリシー	○		
業務機能	全般	要求仕様の実装確認	ユーザが要求するすべての機能が設計書に網羅されているか確認する。					○	○	○
	画面	画面レイアウト設計	プロトタイプでの顧客[illegible]確認する。			○	htmlなど動作可能資産	○		
		画面機能	エラー時の画面遷移[illegible]				画面遷移定義	○		
	帳票	帳票レイアウト設計	出力帳票のレイアウトを確認する。				帳票レイアウト定義	○		
		帳票機能	帳票出力方式が明記されているかを確認する。(PDF,ブラウザ印刷機能など)			○	方式設計書	○		
	データベース	データモデル設計	正規化がされているかを確認する。(3次正規化を実施したか？正規化後に正規化崩しをしているか？)				テーブル関連図 テーブル定義	○	○	○
		DBアクセス機能	排他についての記述があるか確認する。				DBアクセス部品設計書	○		
	ファイル	ファイル定義	必要なファイルが定義されているか、分割は正しいかを確認する。				ファイル一覧	○	○	○
		項目定義	必要なデータ項目がすべて明記されているか確認をする。				ファイル定義	○	○	○
		その他	中間ファイルについても定義がされているかを確認する。				ファイル定義		○	○
	その他入出力	コード	区分・コードのプログラムからの参照方法の定義を確認をする。				部品設計書	○	○	
		メモリ	メモリサイズの見積もりはされているかを確認する。(必要メモリサイズ、ヒープサイズ、スタックサイズなど)			○	クラス定義	○	○	○
		ログ	ログ出力機能は盛り込まれているかを確認する。				方式設計書	○	○	○
	機能	ユースケース	機能概要の内容は要求仕様を網羅しているかを確認する。			○	ユースケース一覧	○		
		クラス設計	パッケージの構成は明記されているか、わかり易く分割されているかを確認する。			○	パッケージ定義		○	○
		機能詳細設計	機能単位の処理の流れが正しく記述されているか。			○	シーケンス図		○	○
		共通部品	業務共通、システム共通部品は洗い出されているか、過不足はないかを確認する。			○	業務共通機能一覧 業務共通機能定義		○	○
		機能の実現方法確認	計算結果の算出方法の確認をする(計算結果時、画面表示時、印刷時の端数処理方法)。						○	○
	他システム連携		関連システム間のインターフェイスは明確か確認する。						○	○
	運用		運用(日時、月次、年次)日程が明確になっているか確認する。						○	○
	コーディング	クラス定義	関数名が定義されていることの確認をする。				プログラムソース			○
		データ定義	データの記憶領域の妥当性を確認する。				プログラムソース			○
		データ参照	配列の添え字の操作を確認する。(定義、範囲チェック)				プログラムソース			○
		計算エラー	計算結果の格納先オブジェクトの大きさが問題ないかを確認する。				プログラムソース			○
		比較エラー	NULL判定をおこなっているかを確認する。				プログラムソース			○
		制御の流れ	分岐の抜けが正しいことを確認する。				プログラムソース			○
		インターフェイス	呼び出し先メソッド内でのパラメータの更新がないことを確認する。				プログラムソース			○
		入出力エラー	ファイルのアクセス権限の付与のチェックを実施していることを確認する。				プログラムソース			○
		DBアクセス	トランザクション制御(行ロック、commit、rollback)が正しいことを確認する。				プログラムソース			○
		条件判定	判定条件の適合性と論理の矛盾がないことを確認する。				プログラムソース			○
		転送	ノード間の転送データを確認する。				プログラムソース			○
		処理漏れ冗長	必要なロジックの漏れがないことを確認する。				プログラムソース			○
		例外処理	処理例外(Exception)の処理に問題がないことを確認する。				プログラムソース			○
		規約	コーディング規約の遵守しているか確認する。				プログラムソース			○
		セション	セション管理はされているかを確認する			○	プログラムソース			○
性能	性能		画面遷移時のレスポンス要件を満たせるか確認をする。		○	○		○		○
セキュリティ	セキュリティ-外部	機密保持	個人情報の特定という脅威にさらされないよう、無条件で全てのURLからのCookieを受け入れる設定にしないようになっているかを確認する。			○				○
		認証	パスワードで設定可能な値は、他者から類推しづらいように文字数や文字パターンなどで制限しているかを確認する。			○				○
	セキュリティ-内部	アクセス権限	要件に基づき、利用者毎に公開可能なフォルダへのアクセス権限設定を行っているかを確認する。			○				○
		実行権限	フォルダ毎に適切な実行権限を設定されているかを確認する。(スクリプトなど)			○				○
構成	構成	インターフェイス	ハードウェアや通信とのインターフェイスが要求仕様通りになっているかを確認する。			○		○		
		ブラウザ	必要なプラグインに対応したブラウザかを確認する。			○		○		○
	サーバ設定	タイムアウトの考慮確認	設定値は妥当かを確認する。			○	デザインシート	○		

3.2. テスト工程の標準化を実現する「テスト実施ハンドブック」と「テスト項目抽出ツール」

3.2.1. テスト実施における問題認識

研究メンバの実業務の中でのテスト実施時における主な課題として，次の 3 点が挙げられた．

- 必要なタイミングで必要なテストができていない
- テスト対象やテスト方法が分からない
- 問題発生時にインフラの問題なのかアプリケーションの問題なのか分かりにくい

これらの課題が発生する問題として，Web システムのテストにおける標準化や，テスト手法やテスト方法のナレッジ化がうまくなされていことが原因なのではないか，ということで認識が一致した（表 6）．

3.2.2. テスト実施における研究の成果

問題認識を基に，必要なタイミングで必要なテスト項目を実施できるようにテストをカテゴリ分けし（表 7），対象や工程を標準化し，テストの分業化にも対応した「テスト実施ハンドブック」（以下，「ハンドブック」）と実際のテスト項目からテストのやり方（テスト仕様）をテスト仕様書として出力できる「テスト項目抽出ツール」（以下，「ツール」）を作成した．

テスト項目の抽出に関しては，Web システムでありがちな多種多様な基盤にも対応できるように，システムとして本質的なテスト項目のみを洗い出し,テスト方法（実施要領・確認項目）の記述欄に，代表的な基盤における例を複数挙げることで，標準化を図った．加えて，システム開発におけるインフラ構築とアプリケーション開発との分業化にあわせ，テスト工程も分業化（インフラとアプリケーション）が可能になるよう，テスト項目・確認内容を重点的に検討している．「ハンドブック」はテストの進め方や完了基準について，読みやすいように全 20 ページにまとめた．「ツール」は 191 のテスト項目全てに対して，具体的なテスト内容・方法を記述し，即実用可能なように標準フォーマットでの仕様書出力をサポートしている（表 8,図 1）．

最終的には，後述の実践検証での結果や実際に使用してみての使い勝手，確認項目や実施要領の読みやすさ・理解しやすさ等，研究メンバの意見を反映してより使いやすく，より実用的なものへ改善をおこなっている．

表 6　テスト実施における問題認識

課題	原因
テストをどのタイミングで実施すれば良いのか分からない。 ・（不安解消のために）同じテストを何回も繰り返している。 **→テスト工数の増大**	・Webシステムのテストをおこなう際のフローが明確になっていない。 ・Webシステムのどの工程でどのようなテストをおこなうべきかが明確になっていない。
実施すべきテスト項目やテスト方法が分からない。 ・テスト担当者の経験よって、実施するのにかかる時間や実施内容にばらつきがでる（属人化している）。 ・テスト漏れが起こる。 **→品質の低下**	・Webシステムのテストに対して作業項目・作業内容のノウハウが充分に蓄積されていない。
システム構成要素が多く、問題の切り分けに時間がかかる。 ・アプリケーションのテスト時に、 「通信ができない」 「データベースが起動をしない」 等の、ハードウェア・ミドルウェアのトラブルかアプリケーションのトラブルか判断がつきにくいエラーがよく発生する。 ・パフォーマンス問題で、どこがボトルネックなのか切り分けにくい。 **→テスト工数の増大**	・Webシステムにおいて導入するハードウェア、ミドルウェアの部分の設定等を保証するテスト（以下、インフラテスト）が充分でない。

表 7　テストのカテゴリ分け

対象	テスト区分	概要
インフラ	単体	システムを構成するハードウェア単体での設定・動作がしようと合致するかを検証する。
	結合	システムを構成するハードウェアの連携確認、及び障害時の動作検証や運用手順の検証をおこなう。
アプリケーション	単体	画面単位で仕様に合致するかを検証する。
	結合	サブシステム単位で仕様に合致するかを検証する。
全て	システム	システムを構成する全ての要素全体として仕様と合致するかを検証する。

表 8　テスト項目抽出（インフラ単体テスト）

大項目	中項目	テスト項目	具体例・注意事項	作業・確認事項	効果・リスク
インフラテスト	システム構成	疎通ツール等によるOS, ミドルウェアのアクセス確認ができるか？	サンプルアプリケーションの動作確認、ポートアクセス（Telnet、ブラウザ、その他）、 ホストアクセス（Ping）、 ファイアウォールの通信通過の確認（ファイアウォールを挟んでのポートアクセス）、 冗長なポートが無いかを確認する。 PING、DNSを確認する（双方向）。 メールの送信が可能か確認する。	対象のマシン・ポートに対して、通信が可能か確認する。	仕様を保証する。 正常にメール機能が動作することが確認できる。 ほぼ100%で接続問題が発見され、問題の早期切り分けに貢献している。
インフラテスト	事前確認	サーバ時刻は正しいか？	NTPサーバを使用している時は設定が正しくされているかを確認する。 関連するサーバ全ての時間が正しい事を確認する。 直ちに時刻同期をおこない、コマンドが正常終了することを確認する。	サーバ本体の時刻と、標準時刻が一致していることを確認する。 サーバ時刻と標準時間がほぼ同一であることを確認する。	日時による処理が正しく行われること、ログの日時が標準とされる日時であることで、障害時の判断が的確に行える。 実行時刻に起因する処理（EBなどは時間による締切が存在する）によるエラーを未然に防ぐことができる。 障害時等における、時刻の判断基準を明確にすることで、障害時の復旧がスムーズに行える。
インフラテスト	障害通知	障害発生の時、障害検知が仕様通り機能するか？	障害時のエラー内容がシステムに正しく保存されることを確認する。（ユーザーからの連絡なしにシステムでエラー内容を確認できる） （例）障害が起こったときにどのような障害なのか、障害対応内容をログに書き出すなどでシステム上確認できる方法があるのか。	システムに対し、サーバシャットダウン/ミドルウェア停止/ネットワーク切断などの障害を人為的に発生させ、障害検知のシナリオどおりの動きで、システム内メッセージ定義どおりの通知がされるかを確認し、システム要件定義を満たすかどうかを確認する。	障害発生時、それを検知する仕組みを確認しておけば、検知できずに復旧までの調査の時間が増大することを防ぐことができる。
インフラテスト	完全性＆否認防止（後から内容を否定できないこと）	正しいデジタル署名を使っているか？	指定した通信がHTTPSで可能になっているか（443のポートが開いているか）。 HTTPでの通信が不能になっているか。	システムで使用するソフトウェア（プラグイン、OCX）やHTMLがデジタル署名されているか確認する。	利用者にセキュアな通信環境を提供できるようになる。
インフラテスト	不正アクセス	パケットフィルタリングは適切か？（ルータ、ファイアウォール等）	ルータ等のパケットフィルタリング機能設定が適切かどうか確認する。	アクセス制御が確立されていることを確認できる。	フィルタリング間違えによる問題は、アプリテスト時に顕在化することが多い。その際、フィルタリング異常に突き当たるのに時間がかかる事が経験上多い。先に解決しておけば、全体として工数が少なくてすむ。
インフラテスト	パフォーマンステスト	同時接続可能数は適切に設定されているか？	ミドルウェア毎に設定項目が異なるので各ミドルウェアのマニュアル/説明書を確認すること。	Webサーバ、各種ミドルウエアの設定、ルータの設定、ファイアウオールの設定、アプリケーションサーバ、DBサーバ等の同時接続可能設定が適切か確認する。	運用後に接続ユーザが増えた場合に安定して稼動することを確認できる。 システムテスト時に無用なトラブルの発生を防げる。 テスト工数の削減が可能。
インフラテスト	パフォーマンステスト（Webサーバ）	Keep-Aliveの設定は適切に設定されているか？	Keep-Aliveの設定が長いと、クライアントの画面が閉じられていても、セッションが生きた状態になり、メモリリークの原因になる。	設定ファイル内のKeep-Aliveパラメータ設定が適切か確認する。	Keep-Aliveによる処理速度への影響をなくし、安定した処理速度が確保できる。
インフラテスト	パフォーマンステスト	Timeoutの設定は適切か？	サーブレットのタイムアウト値がデータベースアクセスへのタイムアウト値より大きくなっていることを確認する。	DBの接続とリクエストとセッションが仕様通りか、その大小関係に矛盾が無いかを確認する。	システムのパフォーマンスを最大限発揮できるようになる。 大小関係に矛盾があるとDBが処理終了しても、先にWebがタイムアウトしたりして利用者に処理結果を返せないことが生じる。利用者は処理結果が返ってこないので、繰り返し処理依頼をしてしまう。システムの負荷は単純に利用者の繰り返し回数に比例して増加する為、Timeoutの設定ミスだけでシステム過負荷が簡単に起きてしまう。
インフラテスト	パフォーマンステスト	OS, ミドルウェアのログの出力設定は適切か？	ログの出力先が正しいか（UNIX系ならアプリ /var のようなテンポラリー領域をしているか？）、ログの出力量が適切か（特定のサイズで過去のログの消去を行っている）等を確認する。	指定のログがすべて設定通りに出力されていることを確認する。	問合せ時、障害発生時に障害内容を確認することができる。 ベンダに問合せする際、ダンプ情報は必須。

テスト仕様書	システム名	作成:	検証:	サブシステム名	プログラム名
アプリケーション単体					

テスト内容	確認事項	具体例	効果・リスク	テスト結果	テスト日付	再テスト結果	再テスト日付
アプリケーションでリソースのリークが発生していないか？	処理を何度か流して、メモリリークが発生しないか確認する。	同一画面を複数端末で連続実行して、メモリの使用率をマシン上で確認する。メモリ、容量の増加やログ出力量を確認する。	パフォーマンス劣化、マシンの負荷増大を事前に防ぐことができる。				
ネットワークが切断された場合、仕様どおりの処理が正しくおこなわれるか？	LANケーブルの接続を切断し、仕様で定義された処理どおりの動作が実施されるか確認する。	パソコンのLANケーブルを抜いて、ブラウザ⇔HTTPサーバ⇔APサーバ⇔ DBサーバの3つのネットワーク間の確認する。	ネットワークに障害が発生した場合に、ユーザが状況を認識できる。またデータに影響がでないことを保証することができる。リカバリ作業時の復旧範囲の特定に役立つ。				
エラー発生時の画面の動作が対応できているか？	エラーを発生させ、その際の画面の動作が、仕様で定義されている動作をするか確認する。	エラー画面への遷移と、エラーメッセージを確認する。	エラー発生時に、ユーザが原因と対処法を把握することができる。				
障害発生時に決められたリランポイントから正しく処理が開始できるか?	仕様で決められたポイントで復旧できるか確認する。	システムを強制的に停止して、復旧する。データの保存範囲、トランザクションの保存範囲を確認する。	障害が発生した場合に、ユーザが作業していた直後まで問題なく復旧できることを確認することができる。				
Cookieデータの書き込み／削除のタイミングと内容は適切か？	仕様どおりのタイミングで書き込み、削除がおこなわれているか確認する。	画面読み込み時に前のCookie削除をおこない、ボタン押下時にCookie書き込みしているか確認する。また、ログオフ時のCookie削除をしているか確認する。	Cookie削除のタイミングによる情報漏洩を防ぐことができる。				
ソース改ざんなどによる不正データに対応できているか？	パソコン上に一旦保存し、HTMLやスクリプトを改竄した状態で送信してきたリクエストを検知してエラー処理できるかを確認する。	社内システムの場合、そこまで要求されない場合もあるが、社外向けシステムの場合は、要求されるなくても提案する必要はある。	システムに対するセキュリティやデータ整合性を保証することができる。				
URL入力による直接遷移が適切か？	仕様で定められた箇所以外のURL直接参照が不可能なことを確認する。	URL直接参照時にはエラー画面を表示する仕様であれば、エラー画面が表示される事を確認する。	URL直接入力による不正な画面遷移による入力や参照権限を持たない利用者による不正なデータ参照を防ぐことができる。				
メール送信が正しい宛先、内容で正常におこなえるか？	メール送信のトリガーを起動し、正しい内容と送り先にメールが送信されることとを確認する。	メールに余計な内容が含まれていないか、送信対象外の人に送信しようとしていないか確認する。	メールによる情報漏洩や、重複メールや想定しない大量メール送信によるサーバの負荷増大を防ぐことができる。				
ユーザ権限による表示・入力可能項目が適切か？	複数の権限の異なるユーザで同画面の操作をおこない、表示された画面の表示・入力可能項目に仕様どおりの制限がかかることを確認する。	特別な権限をもったユーザが参照・入力できる画面項目を、権限をもっていないユーザが参照不可・入力不可であることを確認する。	ユーザの権限に応じた動作を確認でき、権限を越えた不適切な動作をなくすことでシステムに対するセキュリティを保証することができることができる。				
ユーザ権限による画面遷移に対応できるか？	全てのユーザ権限のパターンで認証をおこない、画面遷移を確認する。	特別な権限が必要な画面が、その権限を持つユーザだけに表示されることを確認する。(ポータル画面など)	不適切な画面遷移をなくすことでシステムに対するセキュリティを保証することができる。				
サニタイジングをおこなっているか？	入力項目に特殊記号を入力し、チェックがかかることを確認する。	", ', (,), 半角カナ, <>, &などを記入する。シェルやDB、OSのコマンドも試す。	特殊記号が入力されることによるシステム異常を防ぐことができる。				
文字化けに対応できているか？	仕様で定義された文字出力になるかを確認する。	ブラウザのエンコード設定をサーバーの設定と異なる内容にする。	文字化けによる不具合の抑制、およびアクセサビリティの確認することができる。				
フォームの各入力フィールドのデフォルト値が適切か？	初期画面を表示させ、表示されているデフォルト値が正しいことを確認する。	デフォルト値がある場合にはその値が、ない場合には、何も値がセットされていないことを確認する。	デフォルト値が誤っていた場合に起因する後処理への悪影響を防ぐことができる。				
再入力の際、入力方法が考慮されているか？	同じ項目に対して複数回同一内容を入力したときに、仕様どおりの動作になっているか確認する。	オートコンプリートの利用およびコピーできない項目を確認する。	情報漏洩を防ぐことができる。				
入力項目タブ移動が適切か？	入力項目のカーソルをタブキーで移動させ、仕様どおりにカーソルが移動をしていることを確認する。	SHIFT＋TAB（反対回り）も確認する。	キーボード使用時のユーザの使いやすさを確保することができる。				
月末日の計算が正しくできているか？ 月末日の入力・更新がおこなえるか？	月末日の入出力ができるかを確認する。最終日およびその＋－1日の入力をおこなう。	各月の最終日。最終日＋1を確認する。特に閏年の時の2月29日の入力可能であることと、閏年以外の2月29日が入力不可であること。4,6,9,11月の31日が入らないこと。	月末日の特別処理に起因する悪影響を防ぐことができる。				
キー連打に対応できているか？	すべての各画面・各項目について、ボタンやリンクにフォーカスが当たった状態で、キーボードの連打をおこない、重複処理または誤処理、異常終了となっていないことを確認する。	Macのときは特に注意する。(Enterキー押下することで処理が流れる可能性あり)	キーボードの連打をされても処理を重複しておこなわず、1回分の処理だけをおこなうことを確認することで、重複データの登録などを防ぐことができる。				
二重押下による重複処理に対応できているか？	画面ごとに持っているボタンの二重押下をおこない、重複処理をおこなっていないことを確認する。	ログとDBで発番処理がうまくできているか。(同じ番号が二重押下によって発番されていないこと)	連続押下されても、処理を重複しておこなわず、1回分の処理だけをおこなうことを確認することで、重複データの登録などを防ぐことができる。				
同時利用者数が増えても要求どおりのレスポンスを得られるか？	想定最大利用者で処理のレスポンスを計測し要件の範囲内であることを確認する。	可能なら負荷テスト用のツールを使用する。共有資源（メモリ、ファイルIO、etc）を利用している箇所で資源の不要な使用をしていないか注意する。	要件のレスポンスを保証することができる。また、ボトルネックを把握することで利用者が増えた際のチューニングのポイントをあらかじめ把握することができる。				

図 1　テスト仕様書（アプリケーション単体）

実際に作成したツールを使用して，システムの品質を保証しつつ,「短納期」への対応ができるかをモデルシステムで実践検証した．評価は「アプリケーション単体テスト」でおこなった．これは,「テストの中で一番多くの要員がアサインされ，テスト工程の中で，最も工数が発生する」というメンバの共通認識から,単体テスト工程の時間が削減されることによる効果は非常に大きいと考えたためであり，また，単体時に質の良いテストをおこなっていれば,後工程は本来の役割(要件に対するテスト）に専念できるようになり，効果を最大限に発揮できると考えたためである．

上記を踏まえて，実践検証時における評価の重点ポイントを，以下の 3 点とした．

・テスト項目抽出数
・不具合検出率
・テスト仕様書作成時間

これらは,「テスト項目抽出数」と「不具合検出率」の向上が「品質の向上（確保)」に寄与し，同時に「テスト仕様書作成時間」の短縮が「納期の短縮」を実現するという点で，研究目的が達成できるかどうかの重要な指標である．

評価結果は，テスト項目の漏れを防ぎ，不具合の検出率を向上させ，品質を上げる効果が得られた．その上で，テスト仕様書作成時間を 63%も短縮し，短納期に対しても有効なものであった（表 9・10・11)．

表 9　テスト項目抽出数

	全体	PJT管理	ﾚﾋﾞｭｰ管理	初期開発
項目抽出数	86件	45件	41件	-
テスト実施数	45件	26件	19件	30件

表 10　不具合検出率

	全体	PJT管理	ﾚﾋﾞｭｰ管理	初期開発
検出数	11件	6件	5件	6件
検出率	24.4%	23.1%	26.3%	20.0%

表 11　テスト仕書作成時間

		全体	PJT管理	ﾚﾋﾞｭｰ管理
項目抽出時間	見積	3～6H	1.5～3H	1.5～3H
	研究	2H	1H	1H
仕様書作成時間	見積	5～10H	2.5～5H	2.5～5H
	研究	4H	2H	2H

「ツール」がテストの具体的方法まで提示しているため，スムーズにテストをおこなうことができ，作業時間についても，45 件のテストが 8 時間（4 人×2 時間）で終了し，仕様書作成とあわせた全体の時間（14 時間）が，実開発の仕様書作成時間（16 時間）よりも短い結果となった．つまり，本研究成果を取り入れたテスト工程は，従来のテスト工程での仕様書作成中に完了してしまうということである．この実践検証で特筆すべき点は，実践検証当日に初めてモデルシステムを見ただけの研究メンバが，仕様書作成およびテスト実施において，この様な効果を得られた点にある．

4. 提言

「高品質」と「短納期」の相反する 2 つのキーワードを両立させるために，システム開発の上流工程である要件定義段階からのレビューの徹底と，最終工程であるテスト工程の効率化の両面から研究を進めてきた．

特に Web システムは，ベースとなっている HTML が一般に普及し始めてから比較的新しく，企業システムに本格的に導入されて始めて，日が浅いため，ホスト型・C/S 型のシステムに比べ，レビューを含めたテストに関してナレッジ化が充分にされていないと言える．本研究では，Web システム開発に必要なレビューやテストについてナレッジ化し，活用しやすい形にまとめたが．これにより今までのテストを見直し（ルネッサンスし),高品質と短納期の両立に一役を担うことを切に望む．

また本研究は，Web システム開発における設計のテストとしてのレビューも含めたテストの，実現可能な在り方の模索であった．残念ながら，一年間という限られた研究期間であったため，現状では大きく 2 つの課題が存在する．1 つは要件からのテスト項目の自動抽出が実現されていない点，もう一方は技術の進化が速い現代において，いかにノウハウの陳腐化を防ぐかという点である．

1 つ目の課題は,「人間の考えや判断」をルール化できるかという点で非常に難問ではあるが，レビュー成果物とテスト成果物を少しずつでもつなげていくことが，解決策の 1 つとなると考えている．

2 つ目の課題の解決は，常に利用し，フィードバックし続けること，またそのような環境（文化）を作りだしていくことが解決策になると考える．

この 2 つの課題を克服したとき，本研究の成果物はより標準化され，普遍的なテストの在り方に発展することができると考える．このことは，各企業にて業務をおこなう中で，研究メンバがそれぞれ自分自身に役立つような形で解決すべき課題でもある．

文　献

[1] Web システムのテスト／検証方法　－最後の砦はあなたが守る－：2003 年度研究成果報告書（第 8 分冊)，ﾘｰﾃﾞｨﾝｸﾞｴｯｼﾞｼｽﾃﾑ研究会，2004 年 5 月

[2] Web システムのテスト／検証方法　ガイドライン　～ 誰にでもすぐに使えるテスト指針 ～：2003 年度研究成果報告書　活用ツール，ﾘｰﾃﾞｨﾝｸﾞｴｯｼﾞｼｽﾃﾑ研究会，2004 年 5 月

[3] Cem Kaner, Jack Falk, Hung Quoc Nguyen・テスト技術者交流会[訳]：基本から学ぶソフトウェアテスト，日経 BP 社，初版 5 刷，2003 年 3 月 27 日

[4] Karl E. Wiegers，大久保　雅一　監訳：ピアレビュー，日経 BP ソフトプレス，初版 1 刷，2004 年 3 月 1 日

[5] ＠IT Development Style 第 6 回読者調査結果　開発プロセスの採用状況（複数回答可)：http://www.atmarkit.co.jp/fjava/devs/survey/surney0401/survey0401.html

[6] 決定版！開発プロセス大全：日経ＩＴプロフェッショナル，2003 年 7 月，20 ページ

[7] 民間向け ＩＴシステムのＳＬＡガイドライン：社団法人電子情報技術産業協会，2004 年 10 月

[8] トラブルの根本原因を絶つ　レビューを見直せ：日経コンピュータ，2004 年 3 月 22 日号（№596)，50-73 ページ

[9] Karl E. Wiegers，渡部　洋子　監訳：ソフトウェア要求-顧客が望むシステムとは，日経 BP ソフトプレス，初版 1 刷，2003 年 7 月 14 日

[10] プロジェクト実態調査：日経コンピュータ，2003 年 11 月 17 日号（№587)，52-63 ページ