第５分科会（1 グループ）

テストの自動化による工数削減
Workload reduction by automated software testing

主査　　高橋寿一（ソニー株式会社）
副主査　増田聡　（日本アイ・ビー・エム株式会社）
研究員　蓮沼龍一（日立建機ビジネスフロンティア株式会社）
　　木村佳織（株式会社 NTT データ）
　　小椋健司（TIS 株式会社）

（敬称略　順不同）

1. 概要

ソフトウェアが世の中に浸透するにつれて、その重要性は日増しに高まっており、求められる品質も高くなっている。しかし、ソフトウェアの開発においては、規模が拡大している一方で、期間は短縮される傾向にあり、品質の作り込みが大変困難となっている。そこで我々研究員は、工数削減、品質向上を目的としたテストの自動化にテーマを絞り、各種文献やセミナーへの参加を通して自動化テストに対する調査を行った。そして、「スモークテストの自動化は効果的である」という調査情報を基にスモークテストに対して自動化テストツールの適用を試みたところ、ツールを適用しなかった場合と比較して、工数削減を実現することができた。

Abstract

As software infiltrates the world, the importance rises day by day, and the required quality rises. The size of the software is also getting grow up, however, software development cycle tends to shorten and it is difficult to keep high quality of software. Therefore we narrowed down a theme to automation of a test aimed for testing workload reduction, quality improvement, and an automation test conducted an investigation through various documents and participation to a seminar. And we compared it when we tried an application of an automation test tool for a smoke test based on investigation information that "automation of a smoke test was effective" when we did not apply a tool. And then we achieved workload reduction.

2. はじめに

近年、ソフトウェア開発は規模の拡大と開発の短納期化が著しい。例えば、組込みソフトウェア産業に従事するソフトウェア開発者は１５万人、開発規模は２兆円、また、組込みソフトウェア開発対象は多岐にわたり、個々の規模も平均５０万行、新規開発で平均１８万行と欧米と比べても大規模開発が目立つ。一方、その開発期間は１年未満であることが殆どであり、４０％は半年未満である（付録 1-1)。このような規模の拡大や開発期間の短縮は、これまで担当者独自のスキルで補ってきた品質確保を困難にさせつつあり、ソフトウェア品質を高水準に保つことをより

困難にさせている。これらの状況の打開策として、テスト作業の改善は有力な方策であり、その中でもテストの自動化は１つの有力な手段であると考えられる。この事を裏付ける動向として、ソフトウエアテストツールベンダー最大手のマーキュリーの米国売上高（図１)の推移を挙げる。

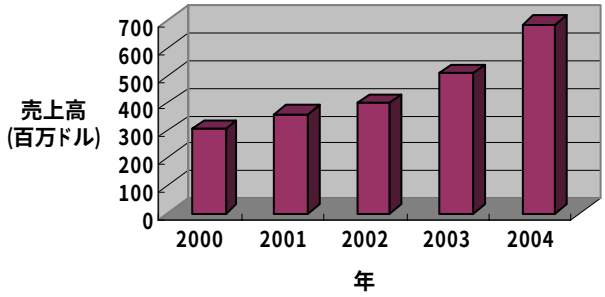


図１　マーキュリー米国本社売上高推移

マーキュリー・インタラクティブ・ジャパン株式会社　会社概要

過去に日本の製造業が、一定の品質を確保するために製造工程に順次ファクトリーオートメーションを導入した時代があったが、それに類似する動きがソフトウェア開発でも押し寄せてきているのではないだろうか。

3. テスト自動化の重要ポイントの調査

テストの自動化における問題点について討議したところ、以下のような点が挙がった。

- どの部分を自動化の対象にして良いか分からない。
- 自動化してみたが、スクリプトの保守性が低く、効率化出来なかった。
- テストツールの選定方法が分からない。

これらの問題点を解決すべく、書籍、セミナー、Web サイトを通じて自動化に関する知識を収集し、重要なポイントを以下にまとめた。

3.1. 目的の明確化

テストの自動化を計画するにあたり、まずはテストの目的を明確化することが重要である。自動化は手段であり、目的ではないため、目的を明確化し、解決策として自動化が最適か判断することが重要である。なお、一般的なテスト自動化の目的には以下のようなものがある。

表 1 一般的なテスト自動化の目的

区分	目的	補足
品質向上	･カバレージ向上 ･品質の数値化 ･確実な欠陥再現	テスト自動化の本命といえる。
工数削減	･繰り返しテスト ･パフォーマンステスト ･スモークテスト ･回帰テスト ･進捗管理工数	・ テストスクリプトの保守性の問題で達成できないケースもある。 ・ 単純作業を減らすことでモチベーション向上に有効という声もある。
納期短縮	･時間外テスト実行	

3.2. テスト自動化の準備作業

テストの自動化に際しては、以下のような準備作業が必要となる。

3.2.1. テスト設計

まずは自動化に依らないテスト全体を設計する。その後、それに基づきテストツール、自動テスト対象範囲、自動テスト方法の検討を行う。この順番が逆になると、テスト自動化自体が目的になってしまう危険があるため注意が必要である。なお、テストケースの作成においては、テストケースの粒度を詳細化しておくことが重要である。テストケースの粒度が粗いと、スクリプトを書きながらテストケースを再検討するという手間が生じてしまう。

3.2.2. テストツールの選定

テスト設計と並行してテストツールの選定を行う。ツールの機能次第では実施出来ないテストもあるため、プロジェクト早期段階で行うことが望ましい。また、テストツールの環境構築や機能習得に対する工数を減らすために、利用するテストツールの標準化やマニュアル整備を行っておくと更に効率が高められる。

3.2.3. テスト自動化戦略の立案

以下の事項を考慮して、自動化の対象範囲と方法を検討する。

- **テストプロセスの成熟度**

テストプロセスの成熟度とテスト自動化のレベルには相関関係があり、組織のテストプロセスの成熟度を考慮したテスト自動化計画を作成することが重要である。なお、「自動ソフトウェアテスト」（参考文献 [1]）には、テストプロセスモデル TMM に応じた適切な自動テスト方法が掲載されている（付録 3-1)。

- **プログラミングスキル**

後述するが、テスト自動化の成否を決める鍵の 1 つとして、テストスクリプトの保守性が挙げられる。レベルの高い自動化を行うには保守性の高いスクリプトを作成することが不可欠であり、そのためにはプログラミングスキルが必要となる。

- **仕様変更の多さ**

仕様変更が多いアプリケーションはテストスクリプトの修正に工数を要するため、自動化には向かない。仕様変更が少ない部分から自動化を行うと良い。

- **ソフトウェアの種類**

ソフトウェアの種類によって自動化が容易なものと困難なものが存在する。例えば、グラフィカル・ユーザー・インターフェース(GUI)や音声ファイルは変更が発生しやすく、少し変更が加わるとスクリプトの再作成が必要となるため、自動化に向かない。しかし、Web アプリケーションの GUI 部分は HTML（テキスト）ベースの比較が可能なため、不一致の場合もエラー箇所が分かりやすく、自動化に向いていると言われている。一方、API は変更が少なく、変更があったとしてもスクリプトの修正で対応出来る場合が多いため、自動化に向く。JUnit、NUnit 等による自動化が広く一般的に使用されていることからも実証されていると言える。

3.2.4. 実施するテスト種類の決定

- **パフォーマンステスト**

人手で出来ない量のテストを実施するため、ツールによる自動テストに向いている。既に Web

アプリケーションのパフォーマンステストにおける自動化は広く行われている。

- **セキュリティテスト**

サニタイジングやクロスサイトスクリプティング対応等、人手で実施し難い量のテストを実施するため、自動化に向いている。

- **回帰テスト**

GUI の回帰テスト自動化については賛否両論であるが、相応のスキルが必要という点では意見が一致している。そのため、他のテストの自動化より難易度が高いという前提で取り組む必要がある。

- **スモークテスト**

スモークテストとはビルドの受け入れテストのことで、ビルド時に毎回実行する。これに関しても賛否両論である。回帰テストの前にまずスモークテストの自動化を行うべきという意見もあれば、スモークテストは問題のあるプロジェクトのみで有効という意見もある。ただし、ラショナル統一プロセス（RUP）のようにデイリービルドを前提としたテストプロセスを実行する場合は繰り返す回数が多くなるため、自動化は必須である。

3.2.5. 優先順位の設定

全てのテストを自動化することは出来ないし、出来たとしても効率の向上に繋がるとは限らない。優先順位の高い機能や品質特性から自動化するとよい。

3.2.6. テスト環境の構築、テストツールの機能習得

テスト環境の構築、テストツールの機能習得にはそれなりの時間を確保する必要がある。テスト開始前に十分な時間が取れるようにスケジューリングすることが望ましい。

3.3. 保守性を考慮したスクリプト作成

テストスクリプトの保守性はテスト自動化成否の鍵の 1 つと言える。以下のような点に留意しながら、プログラミングを行う必要がある。

3.3.1. 共通モジュール化

同じコードが複数のモジュールに存在すると保守性が落ちるため、共通モジュール化する。

3.3.2. データドリブン

パラメータはテキストファイルや Microsoft Excel 等に外出しすると、保守性が高くなる。ツールによって使用できる外部ファイルの形式が異なるため、ツール選定時に考慮する。

3.3.3. シナリオベース

GUI アプリケーションの場合、１アクション毎にスクリプトを作成すると、スクリプト本数が増加し、保守性の低下を招く。業務シナリオベースでまとめることでスクリプト本数を減らせば、保守性の低下を防ぐことが出来る。

4. 調査結果に基づいた自動テスト実施

スモークテストは自動化テストに適しているという調査結果に基づき、スモークテストの自動化を試みることにした。あるアプリケーションに対して、テストツール適用した場合と、適用しない場合で同じテストケースを実施し、工数比較を行って自動化の有効性を検証した。

4.1. テスト設計

4.1.1. 対象システム

テスト対象とする「ホテル予約システム」の概要は、以下の通りである。

表 2 システム概要

項目	内容
機能	Web を通じて、ホテルの予約、照会、取消、変更が行える。
種類	Spring フレームワークを用いた Web アプリケーション。
規模	2.7Kstep、画面数 11
想定する環境	OS: Microsoft Windows 2000/ XP Professional JDK: J2SDK1.5.0 WebAP サーバ: Apache Tomcat 5.5.x（x はバージョン番号） DB: HSQLDB1.8.0.4 総合開発環境: Eclipse SDK 3.1.x
ユースケース	5 つ （ログオン、予約登録、予約確認/取り消し、ユーザ登録/削除、ログオフ）
作成目的	Spring フレームワークを拡張した自社のフレームワークの機能を確かめるためのサンプルアプリケーション。

4.1.2. 実施対象のテスト

開発ではウォーターフォールモデルアプローチを採用している（付録 4-1）。そこで、スモークテストとして結合テストのテストケースを実施することにした。結合テストの概要、およびそのテスト方法は表 3 の通りである。また、今回はデイリービルドを前提としたテストプロセスを定義し、第 1 段階のテストである、ユースケース単位の動作を確認するテストケースに絞ってスモークテストを行うことにした。

表 3 結合テストの概要

テスト概要	第 1 段階	ユースケース単位の動作を確認する。
	第 2 段階	業務単位（ユースケース間連携）の動作を確認する。
テスト方法	第 1 段階	画面からの入出力により動作確認を行う。（順々にリクエストを確認していき、最後のリクエストを確認することで一ユースケースの動作確認を完了とする。前のリクエストから続けて実行する方法をとるため、スタブは作成しない。）
	第 2 段階	第1段階の操作を業務単位（複数ユースケース）で実行する。

4.1.3. テストツールの選定

Selenium IDE を適用することにした。選定理由は「付録 4-2」の通りである。

4.1.4. テストケースの作成

- **テストケース抽出のための観点挙げ**

テストケースの抽出漏れを防ぐために、テストケース抽出のための観点をまとめた（付録 4-3）。

- **テストケース表の作成**

付録 4-3 のテストケースの抽出観点全てを網羅するように、テストケースを抽出した(付録 4-4)。

● スモークテスト対象ケースの選定

作成したテストケースの中から、特に重要度が高いテストケースを選定し、スモークテストで実施することにした。重要度を定める項目、および検討する内容は以下の通りとした。

表 4 重要度の設定

項目	検討内容
ユースケース	・実行頻度の高いユースケースか。 ・重要な処理、システムの中心となる処理を含んでいるか。
テスト観点	・バグが発生しやすい処理を含んでいるか。

表 4 の通り、ユースケースとテスト観点に対して重要度を定め（付録 4-5)、以下のテストケースに対してテストスクリプトを作成することにした。

・ 予約登録ユースケースの処理ロジック（DBアクセスがある場合）

また、スモークテストは必要最低限の品質を確保することを目的としているため、確認する項目は「画面項目」、「画面遷移」、「メッセージ表示」、「ダイアログ表示」とし、「エンティティ」に関する確認は対象外とした。

4.2. テスト実施

4.2.1. テストスクリプトの作成

下表のポイントに注意しながら、テストスクリプトを作成した。

表 5 テストスクリプト作成時のポイント

項目	概要
共通モジュール化	共通する処理単位でスクリプトを切る。
データドリブン	画面操作、実行結果が同じになる場合は、入力データ部分のみを置き換えればよいため、入力データを一覧にまとめる。
シナリオベース	比較結果がエラーになった場合もテストを続行できるように、比較コマンドは verify を使用する。

4.2.2. テストケースの実行

全く同様のテストケースを、テストツールを適用した場合と、適用しない場合で実施した。

- テストツールを適用した場合
  1. 作成した全テストスクリプトを順に実行し、結果を保存する。
  2. 1で保存した実行結果を目視で確認し、テストケース表に結果を記入する。
- テストツールを適用しない場合
  1. テストケース表の項番に沿って順々にテストケースを実施し、「画面遷移」「画面項目」「メッセージ表示」「ダイアログ表示」に関する確認項目を確認する。
  2. 実行結果をテストケース表に記入する。

なお、受け入れ可能かどうかのみ判定すればよいため、画面キャプチャやログ等のテスト証跡は残さなくてよいとした。

4.3. 結果の評価

テスト作業工数、およびテスト実施結果は以下の通りとなった。なお、バグ検出までは実際に行ったため実測値であるが、故障処理の工数値に関しては過去事例を基に算出しており、故障処理の作業としてはバグ修正作業と再試験作業を想定している。

表 6 テスト作業工数

	テスト準備		テストスクリプト作成	テスト実施
適用した場合	テストツールの習得	18.0h	17.0h	0.75h
	テスト環境の構築	3.0h		
適用しない場合	テスト環境の構築	2.0h	―	6.25h

表 7 テスト実施結果

	テストケース	バグ件数	故障処理工数
適用した場合	58 件	14 件	6.7 日
適用しない場合		14 件	7 日

表 7 の通り、ツールを適用した場合も適用しなかった場合も、共に 14 件のバグが発生した。全てのバグが回収されるまでデイリービルドを実施すると定義すると、それぞれの場合の工数は以下のようになる。

- ツールを適用した場合

  18.0 + 3 .0+ 17.0 + 0.75＋(0.75×7) = 44.0
- ツールを適用しない場合

  2.0 + 6.25＋(6.25×7) = 52.0

以上の結果より、ツールを適用した場合は、適用しなかった場合と比較して 8.0 時間削減された。また、ツール適用の 2 回目以降は、テストツールの習得に要する工数が減少し、テストスクリプトの作成効率も高くなることからも、ツールを適用しない場合と比べて非常に効果的であることがわかった。

4.4. 今後の課題

① テスト準備工数の削減

テストツール利用手順書等のマニュアルがなかったため、サイト情報を元にしてテストツールの機能やスクリプトの記述方法を学習した結果、テスト準備工数が膨らんでしまった。今後はマニュアルを整備することで、ツールの環境設定や機能習得に要する工数を削減したい。

② テストケースの選定

スモークテストによって品質向上を図るには重大なバグを発見する必要があるため、スモークテストで実施するテストケースの選定は非常に重要である。今回はユースケースとテスト観点の

重要度によりテストケースを選定したが、今後はシステム的なリスクの観点も含めたテストケースに改善することで、スモークテストによる品質向上に繋げたい。

③ 正確な検証結果

テストケースを実施してバグ件数を出す段階までは実施できたが、バグ修正以降の作業に関しては推測値となってしまった。今後はバグ回収作業を実際に行って、正確な検証結果を出したい。

5. まとめ

我々は書籍、セミナー、Web サイトを通じて得たテストの自動化に関する情報を生かして、スモークテストに対する自動化を試みた。その結果、スモークテストの自動化は工数削減をもたらすことがわかった。しかし、仕様変更が多いアプリケーションに対するテストように自動化に向かないテストもある。したがって、自動化を成功させるためには、十分なテスト設計によりテストの目的を明確化した上で自動化が有効かどうか判断し、目的に沿った自動化をする必要がある。

参考文献

[1]エルフリード・ダスティン、ジェフ・ラシュカ、ジョン・ポール：自動ソフトウェアテスト，ピアソンエデュケーションジャパン

[2] Cem Kaner、James Bach、Bret Pettichord，ソフトウェアテスト 293 の鉄則，テスト技術者交流会，2003

[3]高橋寿一：知識ゼロから学ぶソフトウェアテスト，翔泳社，2005

[4]高橋寿一、湯本剛：現場の仕事がバリバリ進む ソフトウェアテスト手法，技術評論社，2006

[5]加藤大受、町田欣史、大西建児、湯本剛、他：ソフトウェア・テスト PRESS Vol.3，技術評論社，2006

参考セミナー

[1]加藤大受：回帰テストの自動化によるテスト効率の改善について ～ユーザシナリオベースによる Web アプリケーションのテスト自動化～，2005

[2]加藤大受：テスト自動化に関する勉強会～テスト自動化についての概要～，2006

[3]加藤大受：テスト自動化に関する勉強会～テスト自動化について～，2006

[4]湯本剛：テスト自動化の前に押さえておきたい３つのポイント，2006

[5]西康晴：テストの改善、テストによる改善，2006

[6]高橋寿一：プロジェクト成功のためのソフトウェアテストの勘所 ～テスト自動化の理論とその実践～，2006

[7]藤本隆宏：ものづくり論とソフトウェア（ソフトウェア品質シンポジウムにて），2006

参考サイト

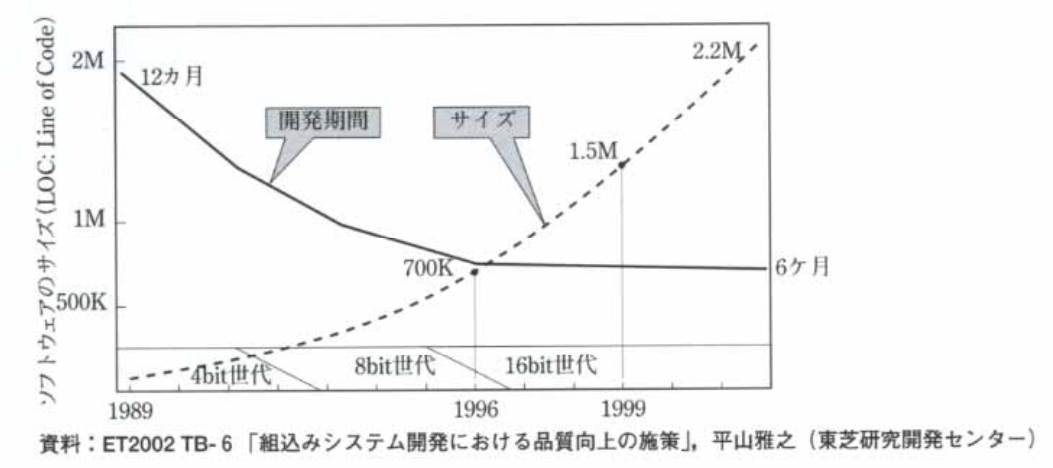
[1]JUnit：http://www.junit.org/

[2]CemCarner：http://www.kaner.com/articles.html

[3]高橋寿一：知識ゼロから学ぶソフトウェアテスト，http://blog.so-net.ne.jp/juichi/

付録

付録 1-1　携帯電話のプログラムサイズの推移(*1)



資料：ET2002 TB-6「組込みシステム開発における品質向上の施策」，平山雅之（東芝研究開発センター）

(*1)情報サービス産業白書 2005

付録 3-1　テスト成熟度と自動ソフトウェアテスト成熟度のレベル１～５(*2)

レベル	TMM	自動ソフトテスト
1	初期―テストは混沌としたプロセスである。テストの定義は誤っており、デバッグと区別されていない。テストはコーディングが終わった後、アドホック（行き当たりばったり）に行われる。テストとデバッグがソフトウェアのバグだしで交互に行われる。テストの目的はソフトウェアが動作することとされている。ソフトウェア製品は品質保証無しでリリースされる。リソース、ツール、適切に訓練されたスタッフが不足している。このタイプの組織は、おそらくソフトウェア工学研究所で開発されたCMMのレベル1であろう。この水準では成熟度ゴールはない。	このレベルでの自動ソフトウェアテストは、「偶然の自動化」と言われる・。レベル1では、自動テストは全く実施されていないか、あるいはアドホックベースで実施されるだけである。自動テストツールは実験的に使用されるかもしれない。捕獲/再生ツールを用いる自動テストの場合、ツールが作成したスクリプトのみでしかスクリプトの記録と再生をしていない。再利用容易性や保守容易性のために、スクリプトが手直しされることはない。自動スクリプトが設計されることもなく、開発標準に従うこともない。成果物であるスクリプトは再利用できず、保守も難しく、書くソフトウェアのビルドごとに再作成しなくてはならない。このタイプの自動化ではテストコストが125%以上に増加する。－－－ある事例では、手動テストに比べて各テストサイクルで150%増しとなっている。（第2章の「ケーススタディ：テスト自動化の」）
2	フェーズ定義―テストはデバッグと区別され、コーディングに続くフェーズと定義されている。計画された活動にも関わらず、レベル2でのテスト計画は、テストプロセスが未成熟なため、コーディングが終わった段階で案内される。たとえば、全てのテストは実行ベースでコードに依存するとの認識がレベル2であり、それゆえ、コードの完成した後でのみ計画される。この成熟段階のテストの主なゴールはソフトウェアが仕様にあっていることを示すこととされている。基本的なテスト技法と手法は適用されている。このTMMのレベルでは、テスト計画立案がソフトウェアライフサイクルの後の方で起こるので、品質問題が多く発生する。さらに、この重要な問題に対してﾚﾋﾞｭｰプログラムが実施されることはなく、欠陥が要件定義や設計フェーズを伝播し、コードに混入する。いまだに、コード作成後テストや実行ベーステストが主要なテスト活動だと考えられている。	「このレベルでは、テストは計画対象活動になっている。これは活動の完了をコミットするものである。プロジェクト計画ツールはプロジェクトマネージャによるテスト活動の定義や、テストプロセスに対する時間、資金、リソース、スタッフの配分を支援する」。 このレベルの自動ソフトウェアテストは「付随的な自動化」と呼ばれる。レベル2では自動テストスクリプトは手直しされるが文書化された標準や反復容易性もない。 このレベルで使用されるツールのタイプには、プロジェクト計画ツール、捕獲/再生ツール、シュミレータおよびエミュレータ、構文解析および意味解析ツール、デバッグツールなどがある。 新規プロジェクトへの自動テストツールを導入する計画がなく、プロセスも遵守されていない。テスト設計や開発標準も存在しない。自動テストツール利用の検討では、テストスケジュールやテスト要求事項が考慮されておらず、信頼性がない。レベル1と同じく、この種のテスト自動化は投資し駅が大きなものでゃなく、実際はテスト作業を増やしてしまう。
3	統合―テストは単にコーディング終了後に行うフェーズではなく、むしろ全てのソフトウェアライフサイクルと統合される。レベル2で捕獲したテスト計画スキルを前提として、組織が作り上げられる。レベル2で獲得したテスト計画スキルを前提として、組織が作り上げられる。レベル2とは違って、レベル3では（TMMでのテスト計画立案は）要求定義フェーズから開始され、1種のV字モデルに従うライフサイクル全体に渡って継続される。 テストの目的は、ユーザと顧客のニーズに基く要求事項に関わるものとして確立されており、テストケース設計と成功基準に利用される。テスト組織が存在し、テストは専門的活動と認識されている。技術訓練の組織体がテストに焦点を当てている。主要テスト活動を支援する基本的なツールがある。しかし、このレベルの組織は、品質管理の中でレビューが重要な役割を果たすと言う認識をもち始めて入るが、公式レビューについては確立されておらず、レビューはライフサイクル全体に渡っては実施されていない。プロセスと製品属性を適格化する際のテスト測定プログラムはまだ確立されていない。	この成熟度レベルでは「意図的な自動化」と呼ばれる。レベル3では、自動テストはよく定義され、よく管理されている。テスト要求事項やテストスクリプト自体はソフトウェア要求しようや設計文書から論理的に育成される。 児童テストスクリプトはテスト設計と開発標準に基いて作成されるが、自動テストプロシージャのレビューについて、テストチームはまだ実施していない。自動テストの再利用容易性や保守容易性も高まる。このレベルの自動テストで、投資利益が報われ始め、回帰テストの2回目の段階で損益分岐点に達する（第2章の「ケーススタディ：テスト自動化の価値測定」を参照）。このレベルで使用されるツールのタイプには要求管理ツール、プロジェクト計画ツール、捕獲/再生ツール、シュミレーションおよびエミュレーションツール、構文解析および意味解析ツール、デバッグツールが含まれる。
4	管理と測定―テストは測定されて定量化されたプロセスである。開発プロセスの全フェーズでレビューはテスト活動と品質管理活動として認識されている。ソフトウェア製品は信頼性、ユーザビリティ、保守容易性などの品質特性に対してテストが実施される。全プロジェクトからテストケースが収集され、テストケースデータベースに収容され、テストケースの再利用と回帰テストに利用される。欠陥は記録され、重大度レベルが付与される。このレベルでのテストプロセスでは良くある欠陥は、欠陥予防思想の決六や、テスト関連メトリクスの収集、分析、復旧での自動化支援が不十分である点に由来していることが多い。	このレベルのテスト成熟度は「進んだ自動化」と言われ、本書で述べるATLMの側面を覆う採用すれば、達成できる。このテスト成熟度レベルはレベル3を完全に実践した上で、主にリリース後の欠陥追跡と言う重要な要を加えたものである。欠陥は捕獲されると、直接、修正、テスト作成、回帰テストプロセスに戻される。ソフトウェアテストチームは製品開発をとうごうするかなめの役割を担い、テスト技術者とアプリケーション開発者は協同して働いていて製品がテスト要求事項を充足するようになる。どのようなソフトウェアのバグでも早期に捕獲されれば、修復コストを低く抑えることが出来る。このレベルで用いられるテストツールは、レベル3で述べたものに加えて、欠陥および変更追跡ツール、テストプロシージャジェネレータ、コードレビューツールがある。
5	最適化、欠陥予防、品質管理―TMMのレベル1～4の成熟度ゴールを達成しているインフラストラクチャにより、テストプロセスは定義され、管理されていると言われ、そのコストや有効性がモニターできる。レベル5のテストでは、精密に調整するメカニズムがあり、継続的改善が行われる。欠陥予防と品質管理が実践されている。テストプロセスは統計的標本や確信度、信用性、信頼性を測定して進められる。テストツールを選択し、評価する方法が確立している。自動ツールはテストケースの実行と再実行を完全に支援しており、テストケース設計、テスト関連アイテムの保守、欠陥収集や分析、テスト関連メトリクスの収集、分析、応用を支援している。	本書で述べるATLMのｶﾞｲﾄﾞﾗｲﾝを組み込み、適切なツールを効果的な方法で利用すれば、TMMのレベル5に到達することが出来る。このレベルで利用されるツールはレベル4で述べたものに加えて、テストジェネレータ、複雑度または規模測定ツール、カバレージおよび頻度分析ツール、欠陥分析と欠陥予防での統計ツールがある（各レベルでの述べたツールについては第3章で詳細を述べており、例は付録Bに示している）。

(*2)自動ソフトウェアテスト：参考文献[1]

付録 4-1　工程名称と各工程で確保する品質

工程	確保する品質
要件定義	業務･システム要件が漏れなく定義されている。
基本設計	外部仕様が漏れなく設計されている。
詳細設計	内部仕様が漏れなく設計されている。
製造	詳細設計工程で定義したプログラム仕様通りにプログラムが記述されている。
単体テスト	詳細設計･製造工程で定義した通りにプログラムが動作する。
結合テスト	基本設計工程で定義した通りにソフトウェアが動作する。
総合テスト	要件定義工程で定義した通りにシステムが動作する。

付録 4-2　テスト自動化ツール選定要件

選定項目	概要
システム種別への対応	Web アプリケーション対応のテストツールである。
スクリプトの作成容易性	Web ブラウザ上のユーザ操作を記録しスクリプトを作成できる。
スクリプトの保守性	スクリプトを修正できる。
スクリプトの言語	一般的な言語である。
価格	オープンソースである。

付録 4-3　テストケース抽出観点表

項番	範囲	大観点（☆は必須。その他は該当する項目がある場合に参照すること。）	小観点	テスト概要	テスト方法	確認項目					
						画面レイアウト	画面項目	メッセージ表示	ダイアログ表示	画面遷移	エンティティ
1	SI1(リ)	処理ロジック(☆) (リ) リクエスト単位 (ユ) ユースケース単位 (画面) 画面単位	正常入力データ	全ての正常ケースに対し、仕様通りに動作することを確認する	仕様で定義された全ての正常ケースを抽出し、それぞれのケースに対応した入力データ（DBや外部ファイル、項目の入力値など）のリクエストを実行する (※)入力データによって画面の表示状態や画面項目の値、遷移先など出力データが異なる場合、その全てを網羅すること		○	○	○	○	
2			エラー入力データ	全てのエラーケースに対し、仕様通りにエラーを返すこと、適切に後処理が行われることを確認する	仕様で定義された全てのエラーケースを抽出し、それぞれのケースに対応した入力データ（DBや外部ファイル、項目の入力値など）のリクエストを実行する		○	○	○	○	
3		DBアクセスがある場合	正常入力データ	DBに登録されている項目と画面に入力/表示する項目の整合性がとれていることを確認する	DBに登録されている項目値を入力データ/検索データとしたリクエストを実行する		○	○	○		
4			エラー入力データ	DBに登録されていない項目が画面に表示されないこと、入力するとエラーになること、を確認する	DBに登録されていない項目値を入力データ/検索データとしたリクエストを実行する		○	○	○		○
5			正常入力データ	全ての正常なDBアクセス処理（追加、更新、削除、参照）に対し、仕様通りに動作すること、データの受け渡しが正確に行われDBが正しく更新されること(参照以外)を確認する	DBが正常に稼動している状態で、DBが正しい結果を返すような入力データを使って、DBアクセス処理（追加、更新、削除、参照）を実行する		○				○
6			エラー入力データ(DAOエラー)	全てのDAOエラーとなるDBアクセス処理（追加、更新、削除、参照）に対し、仕様通りにエラーを返すこと、DBが更新されないこと、処理後のDBに問題なくアクセスできることを確認する	DBが正常に稼動している状態で、DAOエラーを戻す入力データを網羅的に抽出し、DBアクセス処理（追加、更新、削除、参照）を実行する (※) 不正なSQL文の実行、他の要求が参照しているデータのdelete など		○	○	○		○
7		外部ファイルの読込み･書出しがある場合	正常入力データ	全ての正常な外部ファイルへの読込み･書出し処理（追加、更新、削除、参照）に対し、仕様通りに動作すること、データの受け渡しが正確に行われ外部ファイルの内容が正しく更新されること(参照以外)を確認する	外部ファイルが正常な状態で、外部ファイルが正しい結果を返すような入力データを使って、外部ファイルへの書込み･読込み処理を実行する		○				○
8			エラー入力データ	全てのエラーとなる外部ファイルへの読込み･書出し処理に対し、仕様通りにエラーを返すこと、外部ファイルが更新されないこと、処理後の外部ファイルに問題なくアクセスできることを確認する	外部ファイルが正常な状態で、エラーを戻す入力データを網羅的に抽出し、外部ファイルへの読込み･書出し処理を実行する (※) 他の要求が参照しているデータのdelete など		○	○	○		○
9		入力値チェック（単項目）(☆)	正常入力データ	全ての入力項目に対する入力値チェック(正常ケース)を確認する	全ての項目に対し、正常入力データのリクエストを実行する (※)境界値などパターン網羅はUTで確認済みのため不要。1つの正常パターンに対し、1つの異常入力データを網羅すればよい。		○				
10			エラー入力データ	全ての入力項目に対する入力値チェック（全てのエラーケース）を確認する	全ての項目に対し、異常入力データのリクエストを実行する (※)境界値/特殊値/異常値のパターン網羅はUTで確認済みのため不要。1つのエラーパターンに対し、1つの異常入出力データを網羅すればよい。		○	○	○		
11		入力値チェック（相関項目）	正常入力データ	相関関係にある項目が、連動して動作すること、値を設定する順番を変えた場合に動作がおかしくならないこと、を確認	相関関係にある項目に対し、仕様で定義されている正常処理を満たす入力データのリクエストを実行する		○				
12			エラー入力データ	相関関係にある項目に対するエラー処理を確認する	相関関係にある項目に対し、仕様で定義されている全てのエラーケースを満たす入力データのリクエストを網羅的に抽出し、実行する		○	○	○		

付録 4-4　テストケース表

分類別ID	観点	概要	説明
S02_03	処理ロジック、DBアクセス	空室検索画面(SC0201)のリクエスト処理確認(1)	空室検索リクエスト、空室一覧確認画面ページ遷移リクエストの処理ロジック、画面遷移を確認する
	入出力項目		共通シナリオ／共通入力データ／共通確認項目
入力項目	画面からの入力（入力データに記載）		共通シナリオ：S02_01の項番1を実行し、「空室検索画面」が表示されている。
出力項目	画面出力（確認項目に記載）		共通入力データ：データパターンA

項番	試験項目 観点	試験項目 内容	試験条件(環境)& 入力データ	確認項目 観点	確認項目 内容	確認日	確認者	判定	確認したV/R
1	正常入力データ	空室検索条件に一致するレコードが、1件以上5件未満の場合、全ての検索結果が空室検索結果に一度に表示されることを確認する。	空室検索条件に以下の値を設定し、「検索」ボタンを押下する。 都道府県　「神奈川」 チェックイン　「2007/4/10」 チェックアウト　「2007/4/12」 宿泊料金の下限　「制限なし」 宿泊料金の上限　「制限なし」 客室種別名　「シングル」 喫煙種別名　「禁煙」 予約時宿泊利用客室数　「1」	画面項目	空室検索画面(SC0201)の空室検索条件に、入力した検索条件がそのまま表示される。 また、空室検索結果の画面項目が「別シート：SC0201 空室検索結果画面」の通りである。				
2	正常入力データ	空室検索条件に一致するレコードが、26件以上30件以下の場合、空室検索結果に検索結果の最初の5件と、ページ遷移リンクが表示されることを確認する。	空室検索条件に以下の値を設定し、「検索」ボタンを押下する。 都道府県　「北海道」 チェックイン　「2007/4/10」 チェックアウト　「2007/4/12」 宿泊料金の下限　「制限なし」 宿泊料金の上限　「制限なし」 客室種別名　「シングル」 喫煙種別名　「禁煙」 予約時宿泊利用客室数　「1」	画面項目	空室検索画面(SC0201)の空室検索条件に、入力した検索条件がそのまま表示される。また、空室検索結果には条件に合致したホテルのうち、最初の5件が表示される。 空室検索結果の上部に、以下のページ遷移リンクが表示される。(下線部分が有効リンク) 5ページ戻る 前 1 2 3 4 5 6 (1～6のページ番号) 次 5ページ進む				
3	エラー入力データ	空室検索条件に一致するレコードが、1件も存在しない場合、空室検索画面にエラーが表示されることを確認する。	空室検索条件の「宿泊料金下限」に100000円を設定し、「検索」ボタンを押下する。 （他の項目は項番1と同値）	画面項目	空室検索画面(SC0201)の空室検索条件に、入力した検索条件がそのまま表示される。 また、空室検索結果に何も表示されない。				
				メッセージ表示	空室検索画面(SC0201)に、 「検索条件に一致する空室情報がありません。検索条件を変更してください。」 と表示される。				
4	正常入力データ	空室検索結果の「次」リンクを押下した場合、検索結果の表示が変わることを確認する。	項番2で指定した検索条件で「検索」ボタンを押下する。 検索結果が表示された後、「次」リンクを押下する。	画面項目	空室検索画面(SC0201)が次の通りである。 ・空室検索条件が変化しない。 ・空室検索結果に、最初に表示されていた5件の次の5件（6～10件）が表示される。 ・空室検索の下に表示されるページ数が「2/6」である。 ・空室検索結果の上下にあるページリンクのうち、「前」「1,3,4,5,6（2以外のページ番号）」「次」のリンクが有効、それ以外は無効の状態である。				

付録 4-5　重要度の設定

● ユースケースに対する重要度の設定

	ログオン	予約登録	予約確認/取り消し	ユーザ登録/削除	ログオフ
実行頻度が高いユースケースか？	◎	◎	○	○	○
重要な処理を含んだユースケースか？	○	◎	○	○	△

● テスト観点に対する重要度の設定

	処理ロジック			入力チェック（単項目）	入力チェック（相関項目）
	DBアクセスも外部ファイルもない場合	DBアクセスがある場合	外部ファイルがある場合		
バグが発生しやすい処理を含んでいるか？	△	◎	○	○	◎