テスト技術者資格制度

Advanced Level シラバス日本語版
テストマネージャ

Version 2012.J04

International Software Testing Qualifications Board

Advanced Level Test Manager Sub Working Group: Rex Black (Chair), Judy McKay (Vice Chair), Graham Bath, Debra Friedenberg, Bernard Homès, Kenji Onishi, Mike Smith, Geoff Thompson, Tsuyoshi Yumoto; 2010-2012.

改訂履歴

◆ ISTQB®

バージョン	日付	摘要
ISEB v1.1	2001 年 9 月 4 日	ISEB Practitione シラバス
ISTQB 1.2E	2003 年 9 月	EOQ-SG による ISTQB Advanced Level シラバス
V2007	2007 年 10 月 12 日	テスト技術者 Advanced Level シラバス、2007 版
D100626	2010 年 6 月 26 日	2009 年に承認された変更箇所の反映と資格種別ごとの各章の分離
D101227	2010 年 12 月 27 日	書式変更と文字校正の反映
D2011	2011 年 10 月 31 日	シラバス分割のための変更、LO に対応する内容変更 BO の追加
Alpha 2012	2012 年 2 月 12 日	D2011 に対して受領したすべての NBs コメントの反映
Beta 2012	2012 年 3 月 26 日	アルファ版に対して随時実施された監修結果の反映
Beta 2012	2012 年 4 月 7 日	GA のためのベータ版リリース
Beta 2012	2012 年 6 月 8 日	NB への文書編集済みバージョンのリリース
Beta 2012	2012 年 6 月 27 日	EWG および用語集コメントの反映
RC 2012	2012 年 8 月 15 日	リリース前バージョン - 最終 NB 編集箇所の反映
GA 2012	2012 年 10 月 19 日	GA リリースのための最終編集

◆ JSTQB®

バージョン	日付	摘要
Version 2012.J01	2013 年 8 月 26 日	ISTQB Advanced Level Syllabus Test Manager Version 2012 の日本語翻訳版
Version 2012.J02	2013 年 10 月 19 日	「ソフトウェアテスト標準用語集（日本語版） Version 2.2.J01」への対応 ・4 章で使用している用語「混入フェーズ」を「フェーズ内阻止」に変更 ・2 章で使用している用語「デグレード」を「リグレッション」に変更
Version 2012.J03	2014 年 3 月 20 日	・1.5 テスト実装の節において、説明を一部変更。
Version 2012.J04	2021 年 2 月 19 日	・1 章で使用している用語「低位レベルのテストケース」および「高位レベルのテストケース」をそれぞれ「ローレベルのテストケース」、「ハイレベルのテストケース」に変更 ・2.3 節で使用している用語「方法論的アプローチ」を「系統的アプローチ」に変更 ・2.4 節で使用している用語「方法論的戦略」を「系統的戦略」に変更 ・全体的に文意が変わらない範囲での誤記修正

目次

謝辞

このドキュメントは、International Software Testing Qualifications Board Advanced Level Sub Working Group - Advanced Test Manager（Advanced Test Manager 作業部会）のコアメンバである Rex Black (Chair), Judy McKay (Vice Chair), Graham Bath, Debra Friedenberg, Bernard Homès, Paul Jorgensen, Kenji Onishi, Mike Smith, Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto が執筆した。

本コアチームは、レビューチームおよびすべての国の国際部会のメンバによる提案と意見に感謝したい。

本 Advanced Level シラバス完成時の、Advanced Level Working Group（Advanced Level 作業部会）のメンバは以下のとおりである（アルファベット順）。

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès（Vice Chair）, Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith（Chair）, Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto

次のメンバが、シラバスのレビュー、意見表明および投票に参加した。

Chris van Bael, Graham Bath, Kimmo Hakala, Rob Hendriks, Marcel Kwakernaak, Rik Marselis, Don Mills, Gary Mogyorodi, Thomas Mueller, Ingvar Nordstrom, Katja Piroué, Miele Posthuma, Nathalie Rooseboom de Vries, Geoff Thompson, Jamil Wahbeh, Hans Weiberg.

このドキュメントは、2012 年 10 月 19 日に開催された ISTQB®の総会で正式に発行された。

0. 本シラバスの紹介

0.1 本書の目的

本シラバスは、テストマネージャ向けの国際ソフトウェアテスト資格 Advanced Level のベースとなる。ISTQB®は、本シラバスを次の趣旨で提供する。

1. 各国の委員会に対し、各国語への翻訳および教育機関の認定の目的で提供する。各国の委員会は、本シラバスを各言語の必要性に合わせて調整し、出版事情に合わせてリファレンスを修正することができる。
2. 試験委員会に対し、各シラバスの学習目的に合わせ、各国語で試験問題を作成する目的で提供する。
3. 教育機関に対し、コースウェアを作成し、適切な教育方法を確定できるようにする目的で提供する。
4. 受験志願者に対し、試験準備（研修コースの一部、または独立した形）の目的で提供する。
5. 国際的なソフトウェアおよびシステムエンジニアリングのコミュニティに対し、ソフトウェアやシステムをテストする技能の向上を目的とする他、書籍や記事を執筆する際の参考として提供する。

ISTQB®では、事前に書面による申請があった場合に限り、第三者がこのシラバスを先に定めた以外の目的での使用を許諾することがある。

0.2 概要

Advanced Level は、次の 3 つの独立したシラバスで構成している。

- テストマネージャ
- テストアナリスト
- テクニカルテストアナリスト

『Advanced Level シラバス概要』[ISTQB_AL_OVIEW]には、次の情報を記載している。

- 各シラバスのビジネス成果
- 各シラバスの概要
- 各シラバスの関係
- 知識レベルの説明（K レベル）
- 付録

0.3 試験のための学習の目的

学習の目的はビジネス成果を支援し、Advanced Level テストマネージャ認定を取得するための試験問題作成を行うために使用する。基本的に本シラバスのすべての箇所は K1 レベルで試験対象となる。つまり、受験志願者は、用語や概念について認識し、記憶し、想起することになる。このため、関連する章の最初には、K2、K3、K4 レベルの学習の目的のみを記載する。

1. テストプロセス – 420 分

用語

終了基準、テストケース、テスト終了作業、テスト条件、テストコントロール、テスト設計、テスト実行、テスト実装、テスト結果記録、テスト計画作業、テスト手順、テストスクリプト、テストサマリレポート

「テストプロセス」の学習の目的

1.2 テストの計画作業、モニタリング、およびコントロール

TM-1.2.1　(K4)システムのテストニーズを分析して、テスト目的を達成するテスト活動およびテスト成果物を計画する。

1.3 テスト分析

TM-1.3.1　(K3)トレーサビリティを確保し、テスト目的、テスト戦略、およびテスト計画に基づいて定義されたテスト条件の完全性と一貫性をチェックする。

TM-1.3.2　(K2)テスト条件を指定する詳細度に影響を与える可能性がある要因および、詳細にテスト条件を指定することの長所と短所について説明する。

1.4 テスト設計

TM-1.4.1　(K3)トレーサビリティを使用し、定義されたテスト条件に基づいて設計されたテストケースの完全性と一貫性をチェックする。

1.5 テスト実装

TM-1.5.1　(K3)リスク、優先順位付け、テスト環境とデータ依存性、および制約を使用して、テスト目的、テスト戦略、およびテスト計画に対して完全性と一貫性のあるテスト実行スケジュールを作成する。

1.6 テスト実行

TM-1.6.1　(K3)トレーサビリティを使用し、テスト目的、テスト戦略、およびテスト計画との完全性と一貫性という観点で、テスト進捗をモニタリングする。

1.7 終了基準の評価とレポート

TM-1.7.1　(K2)終了基準に対する正確なレポート作成と評価を支援するために、テストプロセスにおける正確でタイムリーな情報収集が重要であることを説明する。

1.8 テスト終了作業

TM-1.8.1　(K2)テスト終了作業における 4 つのグループの作業を要約する。

TM-1.8.2　(K3)プロジェクトの振り返りを実行して、プロセスを評価し、改善する領域を発見する。

1.1 イントロダクション

ISTQB® Foundation Level シラバスでは、次の活動を含む基本的なテストプロセスについて記述している。

- 計画とコントロール
- 分析と設計
- 実装と実行
- 終了基準の評価とレポート
- 終了作業

Foundation Levelシラバスでは、プロセスの活動は理論的には順番に行われるが、重複したり同時に行われたりする場合もあると記述している。通常、システムとプロジェクトの状況に合わせて、これらの主要な活動を調整することが求められる。

Advanced Levelシラバスでは、プロセスの改良と最適化をさらに実行し、ソフトウェア開発のライフサイクルとの適合性を向上させ、テストのモニタリングとコントロールを効率的に推進するために、これらの活動の一部を独立したものとして考えている。現在、各活動は、次のように分けて考えている。

- 計画、モニタリング、およびコントロール
- 分析
- 設計
- 実装
- 実行
- 終了基準の評価とレポート
- 終了作業

1.2 テストの計画作業、モニタリング、およびコントロール

本節では、テストを計画、モニタリング、およびコントロールするプロセスに焦点を当てる。Foundation Level で説明したように、これらの活動はテストマネジメントの役割である。

1.2.1 テスト計画作業

各テストレベルにおいて、テスト計画作業はそのレベルのテストプロセスの開始時に始まり、そのレベルの終了作業が完了するまで、プロジェクト全体を通じて継続する。そこでは、テスト戦略で特定された使命や目的を達成するために必要な、活動とリソースの特定を行う。また、テスト計画作業では、プロジェクトのガイド、計画順守の判断、および目的の達成の評価に使用されるメトリクスを収集し追跡する方法も特定する。計画作業段階で有用なメトリクスを決定することにより、ツールの選択、トレーニングのスケジュール、および文書化ガイドラインの作成が可能となる。

テストプロジェクト用に選択された 1 つまたは複数の戦略は、計画作業段階で行うべきタスクを決定するために役立つ。たとえば、リスクベースドテスト戦略（第 2 章を参照）では、リスク分析を使用して、識別したプロダクトリスクの軽減とコンティンジェンシープランの支援を行うために必要な活動を軽減できるように、テスト計画作業プロセスをガイドすることができる。セキュリティに関して問題になりうる深刻な潜在的欠陥を多数識別した場合は、セキュリティテストを開発して実行するために、膨大な工数を費やす。同様に、設計仕様で深刻な欠陥が日常的に見つかることが判明した場合、テスト計画作業プロセスでは、設計仕様の静的テスト（レビュー）を追加する可能性がある。

リスク情報は、さまざまなテスト活動の優先度を決定するために使用する場合もある。たとえば、システム性能に関するリスクが高い場合は、コードが統合され使用可能になるとすぐに、性能テストを実行することがある。同様に、対処的戦略を採用する場合は、探索的テストなどの動的テスト技法のテストチャータとツールを作成する計画が必要となることがある。

さらに、テスト計画段階では、テストマネージャが、採用するテストレベル、各レベルの目標と目的、および各レベルで使用するテスト技法などのテストに対するアプローチを明確に定義する。たとえば、ある航空関連システムのリスクベースドテストでは、リスクアセスメントにより、必要とされるコードカバレッジのレベル、およびそのレベルに応じた使用すべきテスト技法を規定している。

テストベース（たとえば要求仕様やリスクなど）、テスト条件、およびそれらを満たすテストとの間には、複雑な関係がある場合がある。これらの成果物の間には、多対多の関係があることが多い。テストの計画作業、モニタリング、およびコントロールを効率的に実装するには、これらの関係を理解する必要がある。また、成果物間の関係の理解に応じて、ツールを決定する場合もある。

開発チームが作成した成果物とテストチームが作成した成果物との間にも、関係性がある場合がある。たとえば、システム設計者による詳細設計仕様の要素、ビジネスアナリストによるビジネス要件、およびテストチームが定義したテスト成果物との間の関係を、トレーサビリティマトリクスで追跡することが必要になる場合がある。ローレベルのテストケースを設計し使用する場合、計画段階で定義した要件が存在することがある。この計画段階では、テストケースの作成が始まる前に、開発チームが作成した詳細な設計ドキュメントが承認される。アジャイルライフサイクルに従う際には、テスト開始前にチーム間で情報を伝達するために、非公式な情報伝達セッションを利用する場合がある。

テスト計画では、スコープの範囲外のフィーチャを明確に識別することに加えて、（リスク分析に基づき適宜）スコープに含むソフトウェアの特定のフィーチャを一覧にする場合がある。プロジェクトに適した形式への準拠および文書化のレベルに応じて、スコープに含む各フィーチャを、対応するテスト設計仕様に関連付けることがある。

また、この段階で、テストマネージャからプロジェクトアーキテクトへの、初期のテスト環境仕様の定義、必要なリソースの可用性の検証、環境を設定する要員にその作業を割り当てていることの確認、およびテスト環境をすべて揃えて提供するために必要な作業や、コスト、納品スケジュールの把握を行うための要求事項がある場合がある。

最後に、外部とのすべての依存関係、および関連するサービスレベルアグリーメント（SLAs）を識別し、必要に応じて、初期に連絡をとる必要がある。依存関係の例としては、外部グループへのリソース要求があり、他プロジェクト（1 つのプログラム内で活動している場合）、外部のベンダーまたは開発パートナー、開発チーム、データベース管理者への依存などもある。

1.2.2 テストのモニタリングとコントロール

テストマネージャが効率的にテストをコントロールするには、テストスケジュールとモニタリングフレームワークを確立し、テスト成果物とテストリソースを計画と比較して追跡できるようにする必要がある。このフレームワークには、テスト成果物とテスト活動のステータスを計画と戦略目的に関連付けるために必要な、詳細な測定と対象を含める必要がある。

小規模であまり複雑でないプロジェクトでは、テスト成果物とテスト活動を計画と戦略目的に関連付けるのは比較的容易な場合があるが、一般的にこの関連付けを行うには、より詳細な目的を定義する必要がある。この目的には、テスト目的とテストベースのカバレッジをそれぞれ満たす測定と対象を含めることができる。

テスト成果物とテスト活動のステータスをテストベースに関連付ける場合、プロジェクトとビジネスのステークホルダにとって理解可能で適切な方法で行うことが、特に重要である。これを実現するために、テスト条件を介して他のテスト成果物をテストベースに関連付け、テスト条件およびテスト条件のグループに基づいて対象を定義し進捗を測定するという手段を用いることができる。トレーサビリティを適切に構成（トレーサビリティのステータスをレポートできることを含む）することにより、開発成果物、テストベース、およびテスト成果物の間に存在する複雑な関係を、より透明性の高い分かりやすいものにできる。

ステークホルダからモニタリングを求めている詳細な測定値と対象が、システムの機能または仕様に直接関連しない場合がある。特に、公式なドキュメントがほとんど、またはまったくない場合がそうである。たとえば、仕様がシステム機能の観点で定義されているにも関わらず、ビジネスステークホルダが運用上のビジネスサイクルに対するカバレッジの達成に、より興味を持つ場合がある。プロジェクトの初期段階でのビジネスステークホルダの関与は、これらの測定と対象を定義するのに役立つ。これらの測定と対象は、プロジェクト期間中に優れたコントロールを行うのに役立つだけではなく、プロジェクト全体を通じてテスト活動を推進し、テスト活動に良い影響を与えるために役立つ。たとえば、ステークホルダの求める測定と対象が、テスト進捗を正確にモニタリングすることを促進するため、これらの測定を背景にテスト設計とテスト実装の成果物の構築、およびテスト実行スケジュールのよりよい構築をもたらす結果につながる場合がある。また、これらの対象は特定のテストレベルに対するトレーサビリティを提供し、さらに異なるテストレベルにわたってトレーサビリティの情報を提供するのに役立つ可能性もある。

テストコントロールは、継続的な活動である。この活動には、計画と実際の進捗との比較、および必要に応じた是正措置の実装を含む。テストコントロールでは、必要に応じてテスト計画作業を再検討するなど、使命、戦略、および目的を満たすようにテストをガイドする。コントロールデータに対する適切な対応は、詳細な計画作業情報に基づいて行う。

テスト計画ドキュメントおよびテストコントロール活動の内容は、第 2 章に記載している。

1.3 テスト分析

Foundation Level シラバスでは、テストの分析と設計を一緒に検討すると記述しているが、Advanced Level シラバスではそれらを別々の活動として考える。ただし、これらの活動は、テスト設計成果物の作成を推進するために、並列的、統合的、または反復的な活動として実装できることを認識する必要がある。

テスト分析は、「何を」テストするかをテスト条件の形式で定義する活動である。テスト条件は、テストベース、テスト目的、およびプロダクトリスクを分析することにより、識別可能である。また、成功のための詳細な測定および対象（たとえば終了基準の一部）と見なすことも可能であり、成功のためのテスト目的、および他のプロジェクトまたはステークホルダの基準を含む、テストベースおよび定義された戦略目的にまで遡ることができる必要がある。さらに、テスト条件は、テスト設計およびそれ以外のテスト成果物を作成した際には、これらの成果物を追跡できる必要もある。

特定のテストレベルのテスト分析は、そのレベルのテストベースが確立されれば、すぐに実行可能である。公式なテスト技法およびそれ以外の一般的な分析技法（たとえば分析的リスクベースド戦略や分析的要件ベースド戦略など）を使用すると、テスト条件を識別できる。テスト条件では、テストレベル、分析を行うときに使用できる情報、および選択された詳細レベル（つまり、文書化の粒度）に応じて、値または変数を指定する場合もあれば、そうでない場合もある。

テスト条件を指定するための詳細度合いを決定する際には、次のようなさまざまな要因を考慮する必要がある。

- テストレベル
- テストベースの詳細度と品質

- システムまたはソフトウェアの複雑性
- プロジェクトリスクとプロダクトリスク
- テストベース、テスト内容、およびテスト方法間の関係
- 使用するソフトウェア開発ライフサイクル
- 使用するテストマネジメントツール
- テスト設計およびその他のテスト成果物に関する、詳細度、および文書化のレベル
- テストアナリストのスキルと知識
- テストプロセスおよび組織自体の成熟度（成熟度が高いほど、高い詳細度合いが必要となることもあれば、低い詳細度合いが可能になることもあることに注意）
- コンサルテーションのために他のプロジェクトステークホルダを利用できる可能性

詳細にテスト条件を指定すると、多数のテスト条件を作成する傾向がある。たとえば、一般的な e コマースアプリケーション用の 1 つのテスト条件「チェックアウトのテスト」があるが、詳細なテスト条件ドキュメントでは、この条件は、サポートする各支払い方法に対してそれぞれ 1 つの条件があり、対象の各宛先国に対してそれぞれ 1 つの条件があるなど、複数のテスト条件に分かれる場合がある。

詳細にテスト条件を指定する場合の利点には、次のようなものがある。

- 他のテスト成果物（たとえばテストケースなど）を、より柔軟にテストベースおよびテスト目的に関連付けできる。これにより、テストマネージャはより適切で詳細なモニタリングおよびコントロールが可能となる。
- Foundation Level で説明したように、テストベースが確立されてすぐに、さらに可能であればシステムアーキテクチャと詳細設計が使用可能となる前に、より上位のテストレベルのためにプロジェクトの早期に実行することで、欠陥の防止に貢献する。
- テスト成果物を、ステークホルダが理解できる用語で説明できる（多くの場合、テストケースおよびそれ以外のテスト成果物は、ビジネスステークホルダにとって何も意味せず、実行されたテストケースの数などの単純なメトリクスは、ステークホルダのカバレッジ要件に対して無意味である）。
- 他のテスト活動だけではなく、他の開発活動にも影響を与え、活動を導くのに役立つ。
- テストの設計、実装、および実行において、詳細な測定と対象をより効率よく網羅し、より最適化した成果物を生成する。
- あるテストレベルにおける、より明確な水平トレーサビリティのためのベースを提供する。

詳細にテスト条件を指定する場合の短所には、次のようなものがある。

- 時間がかかる場合がある。
- 環境が変わった場合、保守性を維持するのが困難になる場合がある。
- チーム全体で、形式化のレベルを定義し実装する必要がある。

次の状況では、詳細なテスト条件の仕様が、特に効果的な可能性がある。

- 開発ライフサイクル、コストや時間の制約、またはその他の要因に対応するために、チェックリストなどの簡易なテスト設計文書化方式を使用している。
- 公式な要件、またはそれ以外の開発成果物が、テストベースとして、ほとんど、あるいはまったく使用できない。
- プロジェクトが大規模、複雑、または高リスクであり、単純にテストケースを開発成果物に関連付けることでは提供できないモニタリングおよびコントロールのレベルを必要としている。

テストベースを容易に、直接テスト設計成果物に関連付けできる場合、低い詳細度でテスト条件が指定可能である。次のケースでは、その可能性がより高くなる。

- コンポーネントレベルのテスト
- テスト内容とテスト方法との間に単純な階層関係が存在する、複雑性の低いプロジェクト
- テストを定義するためにユースケースを活用できる受け入れテスト

1.4 テスト設計

テスト設計は、何かを「どのように」テストするかを定義する活動である。テスト戦略およびテスト計画で識別したテスト技法を使用して、識別したテスト条件またはテストベースを段階的かつ詳細に作成することによりテストケースの識別を行う。

テストモニタリング、テストコントロール、およびトレーサビリティで使用するアプローチに応じて、テストケースはテストベースおよび定義された目的に直接（または、テスト条件を介して間接的に）関連付けられる場合がある。これらの目的には、戦略目的、テスト目的、および他のプロジェクトまたはステークホルダの成功のための基準を含む。

所定のテストレベル用のテスト設計は、テスト条件を識別し、ローレベルまたはハイレベルのテストケースを作成するために、十分な情報が揃った後で、テスト設計のために採用したアプローチに従って実行できる。上位テストレベルのテストでは、テスト設計が初期のテスト分析に続く独立した活動となる可能性が高い。一方、下位テストレベルのテストでは、テスト分析とテスト設計を統合された活動として実行する可能性が高くなる。

また、実行することが必要なテストを、反復的アプローチを使用して作成する場合、通常はテスト実装で行う一部のタスク（たとえばテストデータの作成など）をテスト設計プロセスに統合する可能性がある。実際に、このアプローチでは、テスト条件のカバレッジを最適化して、その過程でローレベルまたはハイレベルのテストケースを作成できる。

1.5 テスト実装

テスト実装では、テストアナリストがその活動中にテストを体系立てて優先順位付けをする。公式に文書化された説明では、テスト実装とは、テスト設計が具体的なテストケース、テスト手順、およびテストデータとして実装する活動である。IEEE 829 [IEEE829]標準に準拠している一部の組織では、テストケース仕様で入力とそれに関連する期待結果を定義し、テスト手順仕様でテスト手順を定義している。より一般的には、各テストの入力、期待結果、およびテスト手順は、まとめて文書化する。また、テスト実装には、（たとえばフラットファイルまたはデータベーステーブルなどに）格納したテストデータの作成も含む。

テスト実装には、テストチームがテスト実行を開始する準備が整っていることの最終チェックも含む。このチェックには、必要なテスト環境、テストデータ、テストコードが整っていることの確認（一部のテスト環境の確認テストおよびコード受け入れテストの実行）、およびすべてのテストケースを記述し、レビューし、実行の準備が整っていることの確認を含む。また、対象のテストレベルの明示的および暗黙的な開始基準のチェックも含む場合がある（1.7 節を参照）。テスト実装には、テスト環境およびテストデータの詳細な説明の作成を含むことも可能である。

テスト実装時に実行する作業の詳細度合いと関連する複雑性は、テスト成果物（たとえばテストケースやテスト条件など）の詳細度に影響される場合がある。場合によっては、特に回帰テストで長期間再利用するためにテストをアーカイブに保管する際に、テストを実行するテスト担当者にかかわらず信頼性を確保し、一貫したテストが実施されるように、テスト実行手順の詳細な説明を提供することがある。法規制への適用が求められる場合、適用する標準にテストが適合しているエビデンスを示す必要がある（2.9 節を参照）。

テスト実装では、手動テストおよび自動テストの実行順序を、テスト実行スケジュールに含める必要がある。テストマネージャは、特定の順序、または特定の装置でテストを実行することを必要とするリスクや優先度などの制

約を、慎重にチェックする必要がある。テスト環境やテストデータの依存関係を理解し、確認しておかなければならない。

早期のテスト実装には、いくつかの短所が存在する場合がある。たとえば、アジャイルライフサイクルを使用すると、イテレーションからイテレーションの間にコードが劇的に変化し、実装作業の多くが陳腐化する場合がある。アジャイルのような変化しやすいライフサイクルを適用しない場合でも、イテレーティブまたはインクリメンタルなライフサイクルでは、イテレーションの間に重大な変化があるので、スクリプト化したテストが信頼できなくなる、または頻繁な保守が必要となる場合がある。プロジェクトの後期になってさえも要件が頻繁に変わるような、適切にマネジメントされていないシーケンシャルライフサイクルの場合も同様である。膨大な工数を要するテスト実装に着手する前に、ソフトウェア開発ライフサイクル、およびテストが可能なソフトウェアフィーチャかどうかを理解することが賢明である。

早期のテスト実装には、いくつかの長所が存在する場合がある。たとえば、具体的なテストでは、テストベースに従って記述している場合、ソフトウェアの正しい動作状態の稼働例を提供する。ビジネスドメインの専門家は、抽象的なビジネスルールの検証よりも容易に、具体的なテストの検証を行う可能性が高く、それによりソフトウェア仕様のさらなる弱点を識別する可能性がある。このような検証されたテストにより、必要とされる動作の明確な説明を、ソフトウェア設計者およびソフトウェア開発者に提供できる。

1.6 テスト実行

テスト実行は、テスト対象が受け渡され、テスト実行の開始基準を満たしたときに始まる。テストは、テスト実行の前に設計するか、または少なくとも定義する必要がある。ツールは、特にテストマネジメント、欠陥追跡、および（該当する場合）テスト実行の自動化のために、用意される必要がある。メトリクス追跡を含むテスト結果追跡が機能し、追跡されたデータをすべてのチームメンバが理解する必要がある。テスト実行結果記録および欠陥レポートが、使用でき公開される必要がある。テスト実行の前にこれらを確実に用意することにより、テスト実行を効率的に進めることができる。

テストはテスト手順に従って実行すべきであるが、テスト担当者にはある程度の自由裁量も与えられる。これは、テスト中に観察された興味深い振る舞いや追加したテストシナリオを確実にテストで網羅するためである。少なくとも部分的に対処的なテスト戦略に従っている場合、ある程度の時間を、経験ベースおよび欠陥ベースの技法を使用したテストセッションのために確保する必要がある。もちろん、このような記述されていないテストで検出したすべての故障について、故障の再現をするために、記述されているテストケースからの差異を説明すべきである。自動テストの場合は、逸脱することなく、定義した手順に従ったテストとなる。

テスト実行におけるテストマネージャの主な役割は、テスト計画に従って進捗をモニタリングし、必要に応じて、使命、目的、および戦略という点でテストを成功に導くため、コントロール活動を開始し実行することである。そのために、テストマネージャは、テスト結果からテスト条件、テストベース、最後にはテスト目的へと遡るトレーサビリティ、およびテスト目的からテスト結果へと進むトレーサビリティを使用できる。このプロセスについては、2.6 節で詳細に説明する。

1.7 終了基準の評価とレポート

テスト進捗のモニタリングとコントロールに関する文書化とレポートについては、2.6 節で詳細に説明する。

テストプロセスの観点からは、終了基準の評価とレポートを行うために必要な元となる情報を提供する効率的なプロセスを確実に実装することが重要である。

その情報の要件の定義とその情報を収集することは、テストの計画、モニタリング、およびコントロールの一環である。テストマネージャは、テスト分析、テスト設計、テスト実装、およびテスト実行中に効率的な評価およびレポートを促進するため、これらの活動を担当するテストチームのメンバが必要な情報を正確かつタイムリーに提供できるようにする必要がある。

レポートに要求される頻度と詳細度合いは、プロジェクトと組織によって異なる。これについてはテスト計画フェーズで検討し、関連するプロジェクトステークホルダとの協議が必要となる。

1.8 テスト終了作業

テスト実行が完了すると、主要なアウトプットを集めて次工程の担当者に引き渡すか、保管する必要がある。これらの作業全体が、テスト終了作業である。テスト終了作業は次のように大きく 4 つに分けられる。

1. テスト完了チェック － すべてのテスト作業が実際に完結したことを確認する。たとえば、計画上のすべてのテストは実行済みか、あるいは意図的にスキップしているべきである。判明したすべての欠陥は、修正し確認テストを行っているか、将来のリリースで対応するか、あるいは永続的な制約として許容するかがはっきりしているべきである。
2. テスト成果物の提供 － 価値のある成果物を、それを必要とする人々へ届ける。たとえば、判明している欠陥で改修が延期または現状で許容されているものをシステム利用者やサポート担当へ通知する。また、テストケースとテスト環境を保守テストの担当者に提供する。回帰テストセット（自動または手動）は、文書化し、保守チームに提供するべきである。
3. 学習した教訓 － （テストプロジェクト内およびソフトウェア開発ライフサイクル全体で）振り返りミーティングを主催するか、それに参加し、重要な教訓をドキュメントにまとめ、優れた点を繰り返して過ちを繰り返さないようにする。またプロジェクト計画で対応すべき解決できない問題を明らかにして計画を立てられるようにする。たとえば次のようなことを検討する。
    a. 品質リスクの分析セッションにおいて、十分に幅広く他部門にわたるユーザを招集できたか？たとえば、予期しなかった欠陥の偏在が後になって見つかったことを確認することにより、今後のプロジェクトの品質リスクの分析セッションに召集するユーザの代表は、どの部分まで広げて招集すべきかが分かるようになる。
    b. 見積りは正確だったか？見積りと実績が大幅に違っていた場合、今後の見積りの活動ではその違いを潜在理由とともに明らかにする必要がある。たとえば、テストを効果的に行っていなかった、見積りが本来必要な値より少なすぎたなど。
    c. 欠陥の原因分析および影響分析の傾向と結果は何か？たとえば、遅い段階で発生する変更要求が分析や開発の品質に与えた影響を評価する。たとえば、欠陥が早期に見つかればコスト効率が向上し、時間が節約できる可能性のある早い段階のテストレベルを省略してしまうというような、悪い慣習の兆候を見つける。そして、欠陥の傾向が新しい技術、スタッフの交換、スキルの欠如などと関係があるかを確認する。
    d. プロセス改善の余地はあるか？
    e. 今後の計画で対応すべき、予期しなかった計画との差異は存在したか？
4. 構成管理システムで、結果、ログ、レポート、その他のドキュメント、および成果物を保管する。たとえば、テスト計画およびプロジェクト計画の両方を計画用のアーカイブに保管し、これらを使用したシステムおよびバージョンが明確になるように関連付ける。

このようなタスクは重要であるにもかかわらず、多くの場合、実施すらされないため、テスト計画の一部として明示的に盛り込んでおく必要がある。
一般的に、このようなタスクの 1 つまたは複数を省いてしまうことがある。よくある理由として、チームの配置転換や解散が時期尚早だったことや、それ以降のプロジェクトに対するリソースやスケジュールへプレッシャーがかかったことや、チーム全体が燃え尽き症候群になってしまったことなどが挙げられる。受託開発のような、契約の下で進められるプロジェクトでは、契約に必須となるタスクとして指定しておくべきである。

2. テストマネジメント – 750 分

用語

レベルテスト計画、マスターテスト計画、プロダクトリスク、プロジェクトリスク、品質リスク、リスク、リスク分析、リスクアセスメント、リスク識別、リスクレベル、リスクマネジメント、リスク軽減、リスクベースドテスト、テストアプローチ、テスト条件、テストコントロール、テストディレクタ、テストの見積り、テストリーダ、テストレベル、テストマネジメント、テストモニタリング、テスト計画、テストポリシー、テスト戦略、ワイドバンドデルファイ

「テストマネジメント」の学習の目的

2.2 状況に応じたテストマネジメント

TM-2.2.1　(K4) ステークホルダ、状況、およびソフトウェア開発ライフサイクルモデルを含むソフトウェアプロジェクトまたはプログラムのニーズを分析し、最適なテスト活動を識別する。

TM-2.2.2　(K2) ソフトウェア開発ライフサイクル活動と成果物がテストに与える影響、およびテストがソフトウェア開発ライフサイクル活動と成果物に与える影響を理解する。

TM-2.2.3　(K2) 経験ベースのテストおよび非機能テストに関するテストマネジメントの問題を管理する方法を説明する。

2.3 リスクベースドテストとその他のテストの優先度付けと工数配分のアプローチ

TM-2.3.1　(K2) リスクベースドテストでリスクに対応するさまざまな方法を説明する。

TM-2.3.2　(K2) プロダクトリスク分析のためのさまざまな技法を、例を示して説明する。

TM-2.3.3　(K4) プロダクト品質リスクを分析、識別、および評価し、主要なプロジェクトステークホルダの観点に基づいて、リスクとその評価されたリスクレベルの概要を説明する。

TM-2.3.4　(K2) 識別したプロダクト品質リスクを、ライフサイクルとテストプロセス全体を通じて、評価したリスクレベルに応じて、軽減しマネジメントする方法を説明する。

TM-2.3.5　(K2) テストの選択、テストの優先度付け、および工数の割り当てに関するさまざまなオプションの例を示す。

2.4 テストドキュメントとその他の成果物

TM-2.4.1　(K4) 提供されたテストポリシーとテスト戦略のサンプルを分析し、これらのドキュメントに対して完全性と一貫性のあるマスターテスト計画書、レベルテスト計画書、およびテスト成果物を作成する。

TM-2.4.2　(K4) 所定のプロジェクトに対して、プロジェクトリスクを分析し、適切なリスクマネジメントオプション（軽減、コンティンジェンシープラン、移転、受け入れなど）を選択する。

TM-2.4.3　(K2) テスト戦略のテスト活動に対する影響を、例を示して説明する。

TM-2.4.4　(K3) 適用可能な標準化団体の利用可能なテンプレートを採用して、組織、ライフサイクル、およびプロジェクトのニーズに合ったテスト成果物のための文書化の規範とテンプレートを定義する。

2.5 テストの見積り

TM-2.5.1　(K3) 所定のプロジェクトに対して、適用可能なすべての見積り技法を使用して、すべてのテストプロセス活動の見積りを作成する。

TM-2.5.2　(K2) テストの見積りに影響を与える可能性がある要因を理解し、例を示す。

2.6 テストメトリクスの定義および使用

TM-2.6.1　(K2) 一般的なテスト関連のメトリクスを説明し比較する。

TM-2.6.2　(K2) テスト進捗モニタリングのさまざまな側面を比較する。

TM-2.6.3　（K4）未対応のリスク、欠陥ステータス、テスト実行ステータス、テストカバレッジステータス、および確信度合いの観点で、テスト結果を分析およびレポートし、プロジェクトステークホルダがリリースを決定するための判断材料となる的確な情報と提案を提供する。

2.7 テストのビジネスバリュー

TM-2.7.1　（K2）品質コストを決定する 4 つのカテゴリのそれぞれについて例を示す。
TM-2.7.2　（K3）品質コストをベースに、他の定量的および定性的要素を考慮して、テストの価値を見積り、見積った価値をテストステークホルダに伝える。

2.8 分散テスト、アウトソーステスト、およびインソーステスト

TM-2.8.1　（K2）分散テスト、アウトソーステスト、およびインソーステストのチームスタッフ戦略を、適切に使用するために必要な要因を理解する。

2.9 業界標準適用のマネジメント

TM-2.9.1　（K2）ソフトウェアテストに関する標準の出典とその用途についてまとめる。

2.1 イントロダクション

Advanced Level で、テストプロフェッショナルのためのキャリアの専門化が始まっている。本章では、テストプロフェッショナルがテストリーダ、テストマネージャ、およびテストディレクタの職位に進む際に必要となる知識の領域に焦点を当てる。本シラバスでは、さまざまな組織がこのような職位の人々の職名と職務レベルを、さまざまに定義することを考慮し、これらのプロフェッショナルをテストマネージャと総称する。

2.2 状況に応じたテストマネジメント

マネージャの中心的な職責は、リソース（要員、ソフトウェア、ハードウェア、インフラストラクチャなど）を確保して活用し、付加価値をもたらすプロセスを実行することである。ソフトウェアマネージャと IT マネージャの場合、そのプロセスの多くは、ソフトウェアやシステムを内部または外部で使用するために提供するプロジェクトまたはプログラムの一環である。テストマネージャの場合、そのプロセスはテスト、特に Foundation Level シラバスおよび本シラバスの第 1 章で説明した基本的なテストプロセス活動に関わっている。テストプロセスは、プロジェクトまたはプログラム全体の成功に対する貢献によってのみ（または、より深刻な故障を防ぐことによって）価値を付加できるので、テストマネージャはそれに基づいて、テストプロセスを計画しコントロールする必要がある。別の言い方をすれば、テストマネージャは、他のステークホルダ、その活動（たとえばテストが行われるソフトウェア開発ライフサイクルなど）、およびその成果物（たとえば要求仕様など）のニーズと状況に応じて、他の関連する活動と成果物を含むテストプロセスを適切に準備する必要がある。

2.2.1 テストステークホルダについて

テストステークホルダとは、テスト活動、テスト成果物、または最終システムや最終成果物の品質に関心を持つ人々である。このステークホルダの関心は、テスト活動への直接的または間接的な関与、テスト成果物の直接的または間接的な受理、あるいはプロジェクトやプログラムにより作成された成果物の品質に関する直接的または間接的な影響として現れる。

テストステークホルダは、プロジェクト、プロダクト、組織、およびその他の要因によってさまざまであるが、次のような役割を持っている。

- 開発者、開発リーダ、および開発マネージャ。これらのステークホルダは、テスト対象のソフトウェアを実装し、テスト結果を受け取り、多くの場合その結果に基づいて対応する必要がある（たとえば報告された欠陥の修正など）。
- データベースアーキテクト、システムアーキテクト、および設計者。これらのステークホルダはソフトウェアを設計し、テスト結果を受け取り、多くの場合その結果に基づいて対応する必要がある。
- マーケティングアナリスト、およびビジネスアナリスト。これらのステークホルダは、ソフトウェアに搭載する必要があるフィーチャ、およびそれらのフィーチャに備わっている品質レベルを決定する。また、多くの場合、必要なテストカバレッジの定義、テスト結果のレビュー、およびテスト結果に基づく意思決定に関与する。
- 上級管理職、プロダクトマネージャ、およびプロジェクトスポンサ。これらのステークホルダは多くの場合、必要なテストカバレッジの定義、テスト結果のレビュー、およびテスト結果に基づく意思決定に関与する。
- プロジェクトマネージャ。これらのステークホルダは、プロジェクトを成功へと導くための管理を担当する。プロジェクトを成功させるためには、品質、スケジュール、フィーチャ、および予算における優先度のバランスをとる必要がある。また、多くの場合、テスト活動に必要なリソースを獲得し、テストマネージャと協力してテストの計画とコントロールを行う。
- テクニカルサポート、顧客サポート、およびヘルプデスクのスタッフ。これらのステークホルダは、提供されたソフトウェアのフィーチャと品質によりメリットを受けるユーザと顧客をサポートする。

- 直接的および間接的なユーザ。これらのステークホルダは、ソフトウェアを直接使用するか（つまり、エンドユーザ）、またはソフトウェアが提供またはサポートするアウトプットを利用したりサービスを受けたりする。

テストステークホルダに関する詳細は、[Goucher09]の第 2 章を参照されたい。

このステークホルダのリストは、包括的なものではない。テストマネージャは、プロジェクトまたはプログラムに関する特定のテストステークホルダを識別する必要がある。また、ステークホルダとテストとの関係の本質、およびテストチームがステークホルダのニーズに応える方法について理解する必要がある。すでに説明したテストステークホルダを識別するだけでなく、テストマネージャは、テストに影響を与えたり、テストから影響を受けたりするその他のソフトウェア開発ライフサイクル活動、および成果物を識別する必要がある。これを怠ると、テストプロセスの有効性および効率性が最適化できない場合がある（2.2.3 節を参照）。

2.2.2 その他のソフトウェア開発ライフサイクル活動と成果物

ソフトウェアテストは、テスト活動以外で作成された 1 つ以上の成果物の品質を評価する活動であるので、通常一連の幅広いソフトウェア開発ライフサイクル活動の中に位置づけられる。テストマネージャは、Foundation Level シラバスで説明したように、その他の活動とその成果物がテストに与える影響、およびテストがその他の活動とその成果物に与える影響を理解して、テスト活動を計画しガイドする必要がある。

たとえば、アジャイル開発手法を使用する組織では、開発者は多くの場合、テスト駆動開発を実行し、自動化したユニットテストを作成し、継続的にコードを（そのコードのテストとともに）構成管理システムに統合する。テストマネージャは開発マネージャと協力して、テスト担当者をこれらの活動に統合し、連携して活動するようにしなければならない。テスト担当者は、ユニットテストのカバレッジと効果を増加させるための提案を行い、ソフトウェアとその実装について深い理解を得るために、ユニットテストをレビューする。また、状況を評価して独自の自動テスト、特に機能の回帰テストを、構成管理システムに統合できる。[Crispin09]

テスト活動、その他のテストステークホルダ、ソフトウェア開発ライフサイクルの諸活動、および成果物の間の具体的な関係は、プロジェクト、選択したソフトウェア開発ライフサイクル、およびその他のさまざまな要因に応じて異なるが、テストは次の項目に密接に結びつき、関係している。

- 要求エンジニアリングおよび要求管理。テストマネージャは、要件の変更を認識し、この変更に対応してテストコントロール活動を実行することに加えて、スコープおよびテスト工数の見積り時に要件を考慮する必要がある。テクニカルテストアナリストとテストアナリストは、要件のレビューに参加する必要がある。
- プロジェクトマネジメント。テストマネージャは、テストアナリストおよびテクニカルテストアナリストと協力し、スケジュール要件とリソース要件をプロジェクトマネージャに伝える必要がある。テストマネージャは、プロジェクトマネージャと協力し、プロジェクト計画の変更を理解し、テストコントロールアクションを実行して、これらの変更に対応する必要がある。
- 構成管理、リリース管理、および変更管理。テストマネージャはテストチームと協力して、テスト対象を配布するプロセスとメカニズムを確立し、テスト計画の中にそれらを取り込む必要がある。テストマネージャは、テストアナリストおよびテクニカルテストアナリストに対して、ビルド検証テストを作成し、テスト実行中のバージョン管理を確実に行うように指示することが可能である。
- ソフトウェアの開発と保守。テストマネージャは開発マネージャと協力して、欠陥マネジメント（第 4 章を参照）に参加することに加えて、各テストリリースの内容と日程など、テスト対象の配布を調整する必要がある。
- テクニカルサポート。テストマネージャはテクニカルサポートマネージャと協力して、テスト終了作業時にテスト結果を適切に提供する必要がある。これにより、リリース後にプロダクトのサポートに関わる

人々が、既知の故障および回避策を認識できる。さらに、テストマネージャはテクニカルサポートマネージャと協力して、テストプロセスを改善するために、運用時の故障を分析する必要がある。
- 技術ドキュメントの作成。テストマネージャは技術ドキュメントマネージャと協力し、テストを行うためのドキュメントをタイムリーに提供することに加えて、これらのドキュメントに記載された欠陥をマネジメントしなければならない。

すでに説明したように、テストステークホルダを識別することに加えて、テストマネージャは、テストに影響したりテストの影響を受けたりする、その他のソフトウェア開発ライフサイクル活動および成果物を識別する必要がある。これを怠ると、テストプロセスの有効性および効率性が最適化できない場合がある。

2.2.3 テスト活動とその他のライフサイクル活動の連携

テストは、使用されるソフトウェア開発モデルにかかわらず、プロジェクトの不可欠な要素である。このモデルには、次のようなものがある。
- シーケンシャルモデル（ウォータフォールモデル、V モデル、W モデルなど）。シーケンシャルモデルでは、所定のフェーズ（たとえば要件、設計、実装、ユニットテスト、統合テスト、システムテスト、受け入れテストなど）のすべての成果物と活動が完結してから、次のフェーズが始まる。テスト計画、テスト分析、テスト設計、およびテスト実装は、プロジェクト計画、ビジネス分析と要求分析、ソフトウェア設計とデータベース設計、およびプログラミングと重複した形態で進行する。ただし、正確な重複度合いは、対象となるテストレベルに応じて異なる。テスト実行は、Foundation Level シラバスと本シラバスで説明したテストレベルに従って、シーケンシャルに進行する。
- ラピッドアプリケーション開発（RAD）、ラショナル統一プロセス（RUP）などのイテレーティブモデルまたはインクリメンタルモデル。これらモデルでは、実装されるフィーチャは一緒にグループ化し（たとえばビジネスの優先度またはリスクに従うなど）、フィーチャの各グループに対して、成果物と活動を含むさまざまなプロジェクトフェーズが発生する。このフェーズはシーケンシャルに行うこともあれば、重複した形態で行うこともあり、イテレーション自体がシーケンシャルな場合、または重複している場合がある。プロジェクトの開始時では、上位レベルのテスト計画とテスト分析を、プロジェクト計画とビジネス分析または要求分析を使用して並列的に実行する。詳細テスト計画、テスト分析、テスト設計、およびテスト実装を、各イテレーションの開始時に重複した形態で行う。多くの場合、テスト実行時には、テストレベルの重複が発生する。各テストレベルはできるだけ早期に開始し、より上位の後続するテストレベルが開始した後にも継続して行う場合がある。
- スクラム、エクストリームプログラミング（XP）などのアジャイル。これらは、イテレーションが非常に短い（通常 2～4 週間）イテレーティブライフサイクルである。各イテレーションの成果物と活動が完結してから、次のイテレーションが開始する（つまり、イテレーションがシーケンシャルである）。テストはイテレーティブモデルと同様に進行するが、（さまざまなテストレベルで）テスト実行が開発活動と重複することを含み、さまざまなテスト活動が高い度合いで開発活動と重複する。テスト活動を含むすべてのイテレーションの活動は、次のイテレーションを開始する前に終了する必要がある。アジャイルプロジェクトでは、テストマネージャの役割は通常、直接的なマネジメント上の役割から技術的な専門家またはアドバイザとしての役割に変化する。
- スパイラル。スパイラルモデルは、実現可能性の確認、および設計と実装の決定を行うために、プロジェクトの早期にプロトタイプを使用して試行する。また、ビジネス優先度とテクニカルリスクのレベルに基づいて、プロトタイプを使用する試行の順番を選択する。これらのプロトタイプに対してテストを行い、未解決の技術的問題が残っている部分を特定する。主な技術的問題が解決されると、プロジェクトはシーケンシャルモデルまたはイテレーティブモデルに従って進行する。

テスト活動とライフサイクルを適切に連携させるために、テストマネージャは組織で使用するライフサイクルモデルを詳細に理解する必要がある。たとえば、V モデルの場合で、ISTQB が示す基本的なテストプロセスをシステムテストレベルに適用すると、次の内容に沿ったものとなる。

- プロジェクト計画と同時にシステムテスト計画が始まり、システムテストの実行と終了が完了するまで、テストコントロールを継続する。
- システムテスト分析および設計は、要求仕様から、システムおよびアーキテクチャ設計仕様（上位レベル）を経て、コンポーネント設計仕様（下位レベル）までのプロセスと並行して行う。
- システムテスト実装活動は、システム設計時に開始する場合があるが、これらの活動の大部分は通常、コーディングおよびコンポーネントテストと同時に実行する。この場合、システムテスト実装活動への取り組みは、システムテスト実行の開始数日前まで継続してしまいがちである。
- システムテストの実行は、システムテスト開始基準にすべて合致（または条件付き免除）したときに始まる。これは最低でもコンポーネントテストが完了し、コンポーネント統合テストも完了していることを意味する。システムテストの実行は、システムテスト終了基準に合致するまで継続する。
- システムテスト終了基準の評価およびシステムテスト結果のレポートは、システムテスト実行を通じて行う。これは通常、プロジェクトのデッドラインが近付くほど頻繁となり急を要するものとなる。
- システムテスト終了作業は、システムテスト終了基準に合致しシステムテスト実行の終了を宣言してから始まる。ただし場合によっては、受け入れテストが終わりすべてのプロジェクト活動が終了するまで遅れることがある。

イテレーティブライフサイクルまたはインクリメンタルライフサイクルでは、必要に応じて同じタスクを実行するが、タイミングと範囲が異なる場合がある。たとえば、プロジェクトの開始時にテスト環境全体を実装できる必要はなく、現在のイテレーションに必要な部分のみを実装する方が効率的な場合がある。イテレーティブライフサイクルモデルまたはインクリメンタルライフサイクルモデルのいずれかを使用する際には、計画作業が前倒しされ、基本的なテストプロセスのスコープを前倒しすることが可能となる場合がある。

各プロジェクトで行う計画フェーズに加えて、テスト実行とレポートも、チームが使用するライフサイクルに影響を受ける場合がある。たとえば、イテレーティブライフサイクルでは、次のイテレーションを開始する前に完全なレポートを作成し、イテレーション前のレビューセッションを実行するのが効果的な場合がある。各イテレーションをミニプロジェクトとして取り扱うことにより、前のイテレーション時に発生した事象に基づいて、チームは修正と調整を行うことができる。イテレーションは短期間で時間的制約がある場合があり、このレポートとアセスメントにかける時間と工数を削減することは合理的な場合がある。ただし、この作業は、テストの進捗全体を追跡し、できるだけ迅速にすべての問題領域を識別できるように実行する必要がある。あるイテレーションで発生したプロセスの問題は、解決する対策をとらなければ、次のイテレーションにすぐに影響し、再発することもある。

テストとその他のライフサイクル活動を連携させる方法に関する一般的な情報は、テスト戦略の中に取り込む場合がある（2.4.2 節を参照）。テストマネージャは、テスト計画やプロジェクト計画において、各テストレベルに対して、およびソフトウェア開発ライフサイクルとテストプロセスの任意の選択した組み合わせに対して、プロジェクト固有の調整を行う必要がある。

組織、プロジェクト、および成果物のニーズに応じて、Foundation Level シラバスで定義されているテストレベルに加え、次のような追加のテストレベルを定義できる。

- ハードウェアーソフトウェア統合テスト
- システム統合テスト
- フィーチャ相互作用テスト
- 顧客プロダクト統合テスト

各テストレベルには、次のような項目を明確に定義する必要がある。

- テスト目的（達成可能な目標を伴う）
- テストのスコープとテストアイテム
- テストベース（このベースのカバレッジを測定する手段を伴う、たとえばトレーサビリティ）
- 開始基準および終了基準

- テスト成果物（結果レポートを含む）
- 適用可能なテスト技法（これらの技法を使用して、適切なカバレッジを確保する方法を伴う）
- 測定指標およびメトリクス（カバレッジ測定など、テスト目的、開始基準と終了基準、および結果レポートに関連するもの）
- テストツール（特定のテスト作業に適用できる場合）
- リソース（たとえばテスト環境など）
- 責任を負う要員およびグループ（テストチーム内またはチーム外）
- 組織、規制、およびその他の標準への準拠（該当する場合）

本章内の以降で説明するように、ベストプラクティスは、これらの要素をすべてのテストレベルにわたって明確に定義し、同様のテストの異なるレベル間で発生する無駄で危険なギャップを避けることである。

2.2.4 非機能テストのマネジメント

非機能テストの計画を行わない場合、リリース後にシステムで、深刻な、ときには壊滅的な品質問題が検出される可能性がある。ただし、多くの種類の非機能テストは費用がかかるため、テストマネージャはリスクと制約に基づいて、実行する非機能テストを選択する必要がある。また、非機能テストにはさまざまな種類があり、特定のアプリケーションに、これらのすべてのテストが適しているわけではない。

テストマネージャは、計画のすべての考慮事項に対応できる十分な専門知識を持っていない場合があるので、テスト計画の責任の一部を、非機能テスト活動に割り当てられたテクニカルテストアナリスト（場合によってはテストアナリスト）に委任する必要がある。テストマネージャは、次の一般的な要因を考慮するようにアナリストに指示する必要がある。

- ステークホルダの要件
- 必要なツールの使用
- テスト環境
- 組織的要因
- セキュリティ

詳細は、Advanced Technical Test Analyst シラバス[ISTQB ATTA SYL]の第 4 章を参照されたい。

テストマネージャのその他の重要な考慮事項は、非機能テストをソフトウェア開発ライフサイクルに統合する方法である。すべての機能テストが完了してから非機能テストを開始するのは一般的に誤りであり、重要な非機能的欠陥の検出が遅れる結果になる可能性がある。非機能テストは、リスクに応じて優先度付けし、順番を決める必要がある。多くの場合、テストの早い段階、または開発中でも、非機能的なリスクを軽減する方法がある。たとえば、システム設計時にユーザインターフェースのプロトタイプの使用性をレビューすることで、システムテストの最後に検出した場合には大きなスケジュール上の問題を引き起こしたかもしれない使用性の欠陥を、非常に効果的に特定できる。

イテレーティブライフサイクルでは、変更およびイテレーションのペースが速いので、洗練されたテストフレームワークの構築を必要とする、信頼できる非機能テストに集中するのが困難になる可能性がある。単一イテレーションのスケジュールよりテストの設計および実装活動の方が時間のかかる場合は、イテレーションから外れた独立した活動として整理する必要がある。

2.2.5 経験ベースのテストのマネジメント

経験ベースのテストは、他のテスト技法では見逃す場合がある欠陥を効率的に検出し、これらの技法の完全性をチェックする役割を果たすことでメリットを提供するが、テストマネジメントに関する課題も存在する。テストマネージャは、経験ベースの技法、特に探索的テストのメリットに加えて、これらの課題を認識する必要がある。典

型的に結果記録を軽量にすること、およびテストの事前準備を最小限にすることを考慮すると、テスト中に達成するカバレッジを決定するのは困難である。テスト結果を再現するには、特に複数のテスト担当者が関与する場合、特定のマネジメント上の注意が必要になる。

経験ベースのテスト、特に探索的テストをマネジメントする 1 つの方法は、テストセッションと呼ばれる 30～120 分の短い時間にテスト作業を分割することである。この時間を区切ることにより、セッション内に完了するように作業を制限して絞り込み、一定レベルのモニタリングおよびスケジュールを提供する。各セッションは、テストチャータをカバーする。テストマネージャが、文書または口頭でこのテストチャータをテスト担当者へ伝える。テストチャータにより、テストセッションでカバーするテスト条件を示す。これにより、テストへの集中力が高まり、複数の担当者が同時に探索的テストを実行している場合、重複を防ぐことができる。

経験ベースのテストをマネジメントするもう 1 つの方法は、従来的な事前に設計したテストセッションにこのような自律的かつ自発的なテストを統合することである。たとえば、テスト担当者には事前に定義したテストで明示した手順、入力、および期待結果以上のものを探求する権限（および時間の割り当て）を与えることができる。また、テスト担当者には、事前に定義したテストの実行日の前後および当日に、日常的なテストの一環として、このような自律的テストセッションを割り当てることも可能である。このようなテストセッションで、欠陥または今後のテストのための対象領域を識別した場合、事前に定義したテストを更新する。

探索的テストセッションの開始時に、テスト担当者は、テストのために必要なセットアップ作業を確認し実行する。セッションでは、テスト担当者はテストするアプリケーションについて学習し、適用される技法およびアプリケーションについて学習した内容に従ってテストを設計して実行し、すべての欠陥を調査して、ログにテスト結果を記録する（テストの再現性が求められる場合は、テスト担当者はテストの入力、活動、およびイベントもログに記録する必要がある）。セッションの終了後、報告を行う場合があり、これにより後続のセッションの方向性を設定する。

2.3 リスクベースドテストとその他のテストの優先度付けと工数配分のアプローチ

テストマネジメントの普遍的な課題は、テストを適切に選択、割り当て、および優先度付けすることである。つまり、無限にあるカバーすべきテスト条件と条件の組み合わせの中から、テストチームは有限の条件セットを選択し、テストケースで各条件をカバーするために割り当てる適切な工数を決定する必要がある。さらに、実行するテスト作業の有効性と効率性を最適化するために、作成したテストケースを優先度に基づいて順序付けする必要がある。テストマネージャは、リスクの識別と分析を他の要因と一緒に使用することで、この問題の解決に役立てることができるが、互いに影響し合う制約と変動要因により、妥協を伴う解決策が必要となる場合がある。

2.3.1 リスクベースドテスト

リスクとは、悪いまたは望ましくない結果やイベントをもたらす可能性である。リスクはいつでも発生する可能性があり、これが、顧客、ユーザ、参加者、ステークホルダのプロダクト品質またはプロジェクト成功に対する認識に悪影響を与える。潜在的な問題の主な影響がプロダクト品質におよぶ場合、このような潜在的な問題を品質リスク、プロダクトリスク、またはプロダクト品質リスクと呼ぶ。潜在的な問題の主な影響がプロジェクト成功におよぶ場合、このような潜在的な問題をプロジェクトリスク、または計画リスクと呼ぶ。

リスクベースドテストでは、ステークホルダが参加して行われるプロダクト品質リスク分析において、品質リスクを識別し評価する。テストチームは、テストを設計、実装、および実行して、品質リスクを軽減する。品質には、顧客、ユーザ、およびステークホルダの満足度に影響するフィーチャ、動作、特性、および属性を総合的に含む。従って、品質リスクとは、プロダクト内に品質問題が存在する可能性がある状況である。システムの品質リスクの例としては、レポートでの不正な計算（正確性に関する機能リスク）、ユーザ入力に対する応答の遅れ（効率性

と応答時間に関する非機能リスク)、画面とフィールドの分かりにくさ(使用性と分かりやすさに関する非機能リスク)などがある。テストで欠陥が明らかになった場合、欠陥の存在を周知させ、リリース前にその欠陥に対応する機会を提供したことにより、テストで品質リスクを軽減したことになる。テストにより欠陥が検出されなかった場合、テスト条件の下でシステムが正しく動作したことを確認することにより、テストで品質リスクを軽減したことになる。

リスクベースドテストでは、プロダクト品質リスクを使用してテスト条件を選択し、それらの条件に合わせてテスト工数を割り当て、作成されたテストケースを優先度付けする。リスクベースドテストには、収集するドキュメントにおける重要性の種類と度合いによるバリエーション、および適用する公式度合いによってさまざまな技法が存在する。また、リスクベースドテストには、明示的または暗黙的に、テストにより品質リスクのレベル全体を引き下げる、特にリスクレベルを受け入れ可能なレベルにまで引き下げるという目的がある。

リスクベースドテストは、次の 4 つの主な活動で構成される。
- リスク識別
- リスクアセスメント
- リスク軽減
- リスクマネジメント

これらの活動は重複するが、次の項でこれらの活動について説明する。

最大限の効果を上げるために、リスク識別とリスクアセスメントには、すべてのプロジェクトとプロダクトのステークホルダの代表者を含める必要があるが、現実のプロジェクトでは一部のステークホルダが他のステークホルダの代理人として行動する場合がある。たとえば、一般向けのソフトウェア開発の場合、潜在顧客のごく一部をサンプルとして採用し、ソフトウェアの使用に大きな悪影響をおよぼす潜在的な欠陥を識別するよう依頼することがある。このような場合、潜在顧客のサンプルは、最終的な顧客全体の代理になる。テスト担当者は、プロダクト品質リスクと故障に関する特殊な専門性のため、リスク識別およびリスクアセスメントプロセスに積極的に関わる必要がある。

2.3.1.1 リスク識別

ステークホルダは、次の技法の 1 つまたは複数を使用して、リスクを識別できる。
- 専門家へのインタビュー
- 第三者によるアセスメント
- リスクテンプレートの使用
- プロジェクトの振り返り
- リスクに関するワークショップ
- ブレインストーミング
- チェックリスト
- 過去の経験の活用

リスク識別プロセスでは、ステークホルダのサンプルを広範囲に取り込むことにより、重要なプロダクト品質リスクを数多く検出しやすくなる。

リスク識別は多くの場合、副産物を生み出す。つまり、プロダクト品質リスクではない問題を検出する。例としては、プロダクトまたはプロジェクトに関する全体的な疑問点または問題、要求仕様や設計仕様などの参照ドキュメントの問題などがある。また、プロジェクトリスクは多くの場合、品質リスクの識別時に副産物として識別するが、これはリスクベースドテストの主な焦点ではない。ただし、プロジェクトリスクマネジメントは、リスクベースドテストだけなく、すべてのテストにおいて重要であるので、2.4 節でさらに詳しく説明する。

2.3.1.2 リスクアセスメント

リスク識別を行った後はリスクアセスメントを開始し、識別したこれらのリスクを調査する。特に、リスクアセスメントでは、それぞれのリスクを分類して、それぞれのリスクに関連する可能性や影響を判定する。各リスクのその他の特性の評価、またはリスク所有者の割り当てなども行われる。

リスクの分類では、それぞれのリスクを、性能、信頼性、機能性などの適切なタイプに分ける。一部の組織では、ISO9126 [ISO9126]（ISO 25000 [ISO25000]に改訂中）の品質特性を使用してリスクを分類しているが、多くの組織ではその他の分類体系を使用している。多くの場合、リスク識別で使用するのと同じチェックリストを、リスクの分類でも使用する。一般的な品質リスクチェックリストが存在しており、多くの組織はこれらのチェックリストをカスタマイズしている。チェックリストに基づいてリスク識別を行う場合、リスクの分類は識別時に行うことが多い。

リスクのレベルを決定する場合、通常はリスクアイテムごとに、リスク顕在化の可能性および顕在化した際の影響を評価する。リスク顕在化の可能性は、テスト中のシステムに潜在的な問題が存在する可能性と解釈される。言い換えれば、この可能性とは、技術的なリスクレベルのアセスメントである。プロダクトリスクおよびプロジェクトリスクの可能性に影響する要因には、次のようなものがある。

- 技術およびチームの複雑性
- ビジネスアナリスト、設計者、プログラマの個人的な問題およびトレーニングの問題
- チーム内の衝突
- 供給者側との契約の問題
- チームの地理的な分散
- レガシーアプローチ対新しいアプローチ
- ツールと技術
- 劣悪なマネジメントまたは技術的なリーダーシップ
- 時間、リソース、予算、およびマネジメントのプレッシャー
- 早期からの品質保証活動の欠如
- 高い変更率
- 高い初期欠陥率
- インターフェースと統合の問題

リスク発生の影響は、ユーザ、顧客、またはその他のステークホルダに対する影響の重要度と考えることができる。プロジェクトリスクおよびプロダクトリスクの影響に対する要因には、次のようなものがある。

- 影響を受けるフィーチャの使用頻度
- ビジネス目標の達成に対するフィーチャの重要性
- イメージの悪化
- 業務の喪失
- 財政的、環境保護的、社会的損失または責任の可能性
- 民事上または刑事上の法的拘束
- ライセンスの喪失
- 妥当な回避策の欠如
- 故障の判明による否定的な評判
- 安全性

リスクのレベルは、定量的または定性的に評価できる。可能性および影響が定量的に確認できれば、2 つの値を掛け合わせることで、定量的なリスク優先度値を計算できる。ただし通常、リスクのレベルは定性的にしか確認できない。つまり、可能性が「非常に高い」、「高い」、「中程度」、「低い」、「非常に低い」などと表現できるだけで、可能性を具体的な精度を持つパーセンテージとして表現することはできない。同様に、影響が「非常に高い」、「高い」、「中程度」、「低い」、「非常に低い」とは表現できるが、完全なまたは正確な金額で影響を示す

ことはできない。だが、この定性的なリスクレベルのアセスメントが、定量的なアプローチより劣っていると考えるべきではない。実際、リスクレベルの定量的アセスメントを不適切に使用すると、ステークホルダの実際のリスクに対する理解の程度とリスクマネジメントの度合いを誤ったものにしてしまう。リスクレベルの定性的アセスメントは多くの場合、乗算または加算を通じて総合的なリスクスコアを算出するために、組み合わせて使用する。この総合的なリスクスコアは、リスク優先度値として示す場合があるが、順序尺度としての定性的かつ相対的な評価点として解釈すべきである。

さらに、リスク分析が広範囲で統計的に正しいリスクデータに基づいている場合を除き、リスク分析はステークホルダの主観的な認識における可能性と影響に基づくことになる。プロジェクトマネージャ、プログラマ、ユーザ、ビジネスアナリスト、アーキテクト、テスト担当者は通常、異なる見識を持っているため、それぞれのリスクアイテムに対するリスクレベルについて意見が異なる場合がある。リスク分析プロセスでは、何らかのコンセンサスに達する方法を含める必要があり、最悪の場合でも、合意したリスクレベルを確立しなければならない（たとえば、管理職の指示、またはリスクアイテムの平均値、中央値、または最頻値の採用により合意するなど）。さらに、リスクの評価点がテストの順序付け、優先度付け、および工数の割り当てに対する意味のあるガイダンスになるように、リスクレベルが正しい範囲に割り当てられていることをチェックする必要がある。そうでなければ、リスクレベルはリスク軽減活動のガイドとして使用することができない。

2.3.1.3 リスク軽減

リスクベースドテストは、品質リスク分析（プロダクト品質リスクの識別と評価）で始まる。この分析が、マスターテスト計画およびその他のテスト計画の基礎になる。計画での指定に従って、リスクをカバーするように、テストを計画、実装、および実行する。テストの開発と実行に伴う工数は、リスクのレベルに比例する。つまり、リスクが高いと、より精緻なテスト技法（ペアワイズテストなど）を使用し、一方、リスクが低いと、あまり精緻でないテスト技法（同値分割法やタイムボックスを使った探索的テストなど）を使用する。さらに、テストの開発および実行の優先度は、リスクのレベルに基づく。一部の安全関連標準（たとえば FAA DO-178B/ED 12B、IEC 61508 など）では、リスクのレベルに基づいたテスト技法やカバレッジの程度を規定している。さらに、リスクのレベルは、プロジェクト成果物（テストを含む）のレビューの適用、独立性の度合い、テスト担当者の経験の度合い、および確認テスト（再テスト）と回帰テストの実行度合いの決定に影響を与える。

プロジェクトの期間中、テストチームは、一連の品質リスク、または判明している品質リスクのリスクレベルを変更する追加情報を、常に認識している必要がある。テストの調整をもたらす品質リスク分析の調整を、定期的に行うべきである。これらの調整は、少なくとも主なプロジェクトマイルストンで行う必要がある。調整には、新しいリスクの識別、既存のリスクレベルの再評価、およびリスク軽減活動の有効性の評価を含む。例を挙げると、要件フェーズ中に要求仕様に基づいてリスク識別セッションとリスクアセスメントセッションを行った場合でも、設計仕様が完成したときに、リスクを再評価する必要がある。別の例を挙げると、テストでコンポーネントに予想以上の欠陥が含まれることが分かった場合、この領域における欠陥の可能性が予想以上に高いと結論付け、その可能性およびリスクの全体のレベルを上昇させるように調整する。その場合、このコンポーネントのテスト量を増加させることになる。

また、プロダクト品質リスクは、テスト実行開始前に軽減することが可能である。たとえば、リスク識別時に要件に関する問題を検出した場合、プロジェクトチームは軽減活動として、徹底的な要求仕様のレビューを実施できる。これにより、リスクのレベルを軽減できる。つまり、より少ないテストで、残存する品質リスクを軽減できる可能性がある。

2.3.1.4 ライフサイクルにおけるリスクマネジメント

リスクマネジメントは、ライフサイクル全体で行うのが理想である。組織にテストポリシードキュメントやテスト戦略ドキュメントがある場合、プロダクトリスクおよびプロジェクトリスクをテストでマネジメントする包括的なプロセスを記述し、リスクマネジメントをテストのすべての段階にどのように統合したり、どのように影響をおよぼしたりするかを示す必要がある。

リスクに対する認識がプロジェクトチーム内に浸透している成熟した組織では、リスクマネジメントを、テストだけではなく、さまざまな段階で行う。重要なリスクは、特定のテストレベルで早期に対処するだけではなく、初期のテストレベルでも対処する（たとえば、性能を主要な品質リスク領域として識別した場合、性能テストをシステムテストで早期に開始するだけではなく、性能テストをユニットテストおよび統合テストでも実行する）。成熟した組織はリスクを識別するだけではなく、リスクの原因、およびリスクが顕在化した場合の結果も識別する。発生する欠陥に対しては、リスクの原因をより深く理解し、早い段階で欠陥を防ぐプロセスを改善するために、根本原因分析を行う。リスクの軽減は、ソフトウェア開発ライフサイクル全体を通じて行う。リスク分析では、十分な情報が与えられ、関連する活動を考慮して、システム動作分析、コストベースのリスクアセスメント、プロダクトリスク分析、エンドユーザリスク分析、および損害賠償リスク分析を行う。また、リスク分析では、テストチームがテスト以外でも、プログラム全体のリスク分析に参加し影響をおよぼす。

ほとんどのリスクベースドテスト手法には、リスクのレベルを使用してテストの順序付けおよび優先度付けを行うための技法を含んでいる。これにより、テスト実行時にもっとも重要な領域を早期に網羅し、もっとも重要な欠陥の検出を確実に行うことができる。場合によっては、高リスクのテストをすべて、低リスクのテストよりも先に実行したり、厳格なリスク順にテストを実行したりする。これは「縦型探索」（depth-first）とも呼ばれる。また別の場合は、「横型探索」（breadth-first）とも呼ばれるサンプリングアプローチを使用して、識別したすべてのリスクアイテムからテストのサンプルを選択することもある。この方法では、リスクに基づいて選択に重みを付け、同時に、すべてのリスクアイテムを少なくとも 1 回はカバーするようにする。

リスクベースドテストを縦型探索と横型探索のどちらで進めても、すべてのテストを実行しないうちにテストに割り当てた時間を消費してしまう可能性がある。リスクベースドテストでは、その時点における残りのリスクレベルについてテスト担当者が管理部門にレポートし、管理部門でテストを延長するか、あるいは残りのリスクをユーザ、顧客、ヘルプデスク／テクニカルサポート、運用スタッフに移転するかを決定できる。

テスト実行時、もっとも洗練されたリスクベースドテスト技法（もっとも公式度が高い、または重いやり方である必要はない）は、プロジェクト参加者、プロジェクトマネージャとプロダクトマネージャ、管理職、上級マネージャ、およびプロジェクトステークホルダに、リスクの残存度合いに基づいたソフトウェア開発ライフサイクルのモニタリングとコントロールを可能にする。このモニタリングとコントロールは、リリースの決定を含む。このためには、テストマネージャは各テストステークホルダが理解できる方法で、リスクに関するテスト結果を報告する必要がある。

2.3.2 リスクベースドテストの技法

リスクベースドテストには、さまざまな技法がある。これらの技法のいくつかは、まったく非公式なものである。例としては、探索的テストでテスト担当者が品質リスクを分析するアプローチがある[Whittaker09]。この技法はテストをガイドするのに役立つが、欠陥の影響ではなく欠陥の可能性に過剰に注目してしまう場合があり、クロスファンクショナルなステークホルダからの入力は含まれない。さらに、このようなアプローチは主観的であり、個々のテスト担当者のスキル、経験、および好みに依存している。そのため、これらのアプローチが、リスクベースドテストの利点を完全に実現することはほとんどない。

コストを最小化するのと同時に、リスクベースドテストの利点を獲得するために、多くの実務者は軽量のリスクベースのアプローチを採用している。これらのアプローチは、非公式アプローチの反応の早さや柔軟性と、より公式なアプローチの権限や合意形成をあわせ持つ。軽量なアプローチの例としては、実用的リスク分析とマネジメント（Pragmatic Risk Analysis and Management： PRAM）[Black09]、体系的ソフトウェアテスト（Systematic Software Testing： SST）[Craig02]、プロダクトリスクマネジメント（Product Risk Management： PRisMa）[vanVeenendaal12]などがある。一般的にこれらの技法は、リスクベースドテストの通常の属性に加えて、次の属性を持っている。

- 特に効率性の問題が重要となる業界でのリスクベースドテストに関する経験に基づいて、時間とともに進化する。
- 初期のリスク識別およびリスクアセスメントにおいて、ビジネス的および技術的観点の両方を代表する、クロスファンクショナルなステークホルダによるチームの広範な関与が前提となる。
- プロジェクトのもっとも早い段階で導入した場合、および品質リスクを軽減するために最大に作用するオプションの場合、およびリスクを最小化するようにリスク分析の主な成果物と副産物がプロダクトの仕様と実装に影響を与える場合に、最適化している。
- 生成された出力（リスクマトリクスまたはリスク分析テーブル）を、テスト計画とテスト条件、および後続するすべてのテストマネジメント活動とテスト分析活動のためのベースとして使用する。
- すべてのレベルのテストステークホルダにとっての残存リスクが何か、ということが分かるようなテスト結果の報告に役立てることができる。

これらの技法の一部（たとえば SST など）では、リスク分析への入力として要求仕様が必要となるので、要求仕様が提供される場合以外は使用できない。その他の技法（たとえば PRisMa や PRAM など）では、リスクベースの戦略と要件ベースの戦略をあわせて使用することを推奨する。リスク分析への入力として要件またはその他の仕様を使用するが、ステークホルダの入力に基づくだけでも、機能することが可能である。入力として要件を使用することで、要件の適切なカバレッジを確保できるが、テストマネージャは要件によって示されていない重要なリスクを、特に非機能領域で見逃していないことを確認する必要がある。入力として、優先度付けした適切な要件が提供された場合、一般的にリスクレベルと要件の優先度との間に強い相関関係が見られる。

また、これらの技法の多くは、リスク識別プロセスとリスクアセスメントプロセスを、テストアプローチに関するステークホルダ間の合意を形成する手段として使用することを推奨する。これは強力な利点ではあるが、このためにステークホルダは、グループのブレインストーミングセッションまたは 1 対 1 のインタビューに参加するために時間を割く必要がある。ステークホルダの参加が不十分な場合、リスク分析でギャップが発生する。当然、ステークホルダはリスクのレベルに常に同意するわけではないので、品質リスク分析活動のリーダは、ステークホルダとともに創造的かつ積極的に作業を行い、最善の合意を達成する必要がある。レビューミーティングをリードする訓練を受けたモデレータの全スキルは、品質リスク分析をリードする担当者に適用できる。

軽量な技法は、より公式な技法と同様に可能性および影響の要因に対する重み付けを使用して（たとえば、テストの精緻さの度合いなどに応じて）、ビジネスリスクまたはテクニカルリスクを強調できる。ただし、より公式な技法とは異なり、軽量な技法には次の特徴がある。

- 可能性と影響の 2 つの要因のみを使用する。
- シンプルで定性的な判定と尺度を使用する。

これらのアプローチは、軽量という特性により、柔軟性と広い範囲のドメインへの適用性、あらゆる経験とスキルを持つチーム（非技術要員および若手要員を含む）へのアクセシビリティを持つ。

公式で重い技法では、テストマネージャは次のようなさまざまなオプションが利用可能である。

- ハザード分析。各リスクの根底にあるハザードの識別を試み、分析プロセスを上流に拡張する。
- エクスポージャーコスト。ここでは、リスクアセスメントプロセスで、各品質リスクアイテムに対して、次の 3 つの要因を決定する。
    1) リスクアイテムに関連する故障が発生する可能性（割合で提示）
    2) リスクアイテムに関連する一般的な故障が本番環境で発生した場合の損失コスト（金額で提示）
    3) このような故障をテストするコスト
- 故障モード影響解析（FMEA）およびそのバリエーション[Stamatis03]。品質リスク、その考えられる原因、および起こりうる影響を識別し、重要度、優先度、および検出率を割り当てる。
- 品質機能展開（QFD）。テストに関連した品質リスクマネジメント技法であり、特に、顧客またはユーザの要件に対する理解が誤っている、または不十分なことから生じる品質リスクに関連する。

- フォールトツリー解析（FTA）。実際に（テストまたは本番で）観察されたさまざまな故障、または潜在的な故障（品質リスク）が、根本原因分析の対象となる。最初に故障の原因となる欠陥を分析し、次にこれらの欠陥の原因となるエラーまたは欠陥を分析し、さまざまな根本原因を識別するまで分析を続ける。

リスクベースドテストで使用する必要がある具体的な技法、および各技法の公式度合いは、プロジェクト、プロセス、およびプロダクトの考慮事項に依存する。たとえば、Whittaker の探索的技法などの非公式なアプローチは、パッチまたは応急処置に適用する場合がある。アジャイルプロジェクトでは、品質リスク分析は各スプリント期間の初期に完全に取り込み、リスクの文書化はユーザストーリーの追跡に統合する。システムオブシステムズでは、それ全体および各システムで、リスク分析が必要になる。セーフティクリティカルまたはミッションクリティカルなプロジェクトでは、より高いレベルの公式度合いと文書化が必要になる。

リスクベースドテストの入力、プロセス、および出力は、選択した技法により決定する傾向がある。一般的な入力としては、ステークホルダの知識、仕様、および過去のデータがある。一般的なプロセスとしては、リスク識別、リスクアセスメント、およびリスクコントロールがある。また、一般的な出力には、品質リスク（関連するリスクレベルおよび推奨するテスト工数の割り当てを含む）、仕様などの入力ドキュメントで検出した欠陥、リスクアイテムに関連する疑問点や問題、およびテストまたはプロジェクト全体に影響するプロジェクトリスクのリストがある。

通常、リスクベースドテストのもっとも重要な成功要因は、ステークホルダの適切なチームがリスク識別とリスクアセスメントに関与することである。すべてのステークホルダは、プロダクトの品質の構成要素について独自に理解し、品質に関して独自の優先度と関心事を持っている。また、ステークホルダは、ビジネスステークホルダとテクニカルステークホルダの 2 つの大きなカテゴリに分類される傾向がある。

ビジネスステークホルダには、顧客、ユーザ、運用スタッフ、ヘルプデスクスタッフ、テクニカルサポートスタッフなどを含む。これらのステークホルダは顧客とユーザを理解しているので、リスクを識別し、その影響をビジネスの観点から評価できる。

一方、テクニカルステークホルダには、開発者、アーキテクト、データベース管理者、ネットワーク管理者などを含む。これらのステークホルダはソフトウェアが失敗に至る基本的な過程を理解しているので、技術的な観点からリスクを識別し、可能性を評価できる。

ステークホルダの中には、ビジネスとテクニカルな観点の両方を持っている人もいる。たとえば、テストまたはビジネス分析の役割を持つ特定分野の専門家は多くの場合、ビジネスおよびテクニカルな専門知識があるので、リスクについて幅広い見識を持っている。

リスクアイテムを識別するプロセスは、リスクの膨大なリストを生成するプロセスである。ステークホルダはリスクアイテムについて議論する必要はない。ステークホルダが、何かをシステムの品質に対するリスクと認識すれば、それがリスクアイテムになるからである。ただし、ステークホルダは、リスクのレベルに関する評価について、合意を形成することが重要である。たとえば、評価要因として可能性と影響を使用する軽量なアプローチでは、プロセスの一環として、可能性と影響に対する共通の評価方式を特定する必要がある。テストグループを含むすべてのステークホルダは、この同じ尺度を使用する必要があり、各品質リスクアイテムに対して単一の可能性と影響の評価が合意できるようにしなければならない。

リスクベースドテストを長期にわたって使用する場合、テストマネージャはステークホルダとともにリスクベースドテストの提唱および開始に成功する必要がある。クロスファンクショナルグループは、リスク分析の価値を認識し、その技法を継続的に使用しなければならない。そのためには、テストマネージャは、ステークホルダのニーズ、期待、およびこのプロセスに参加するために使用できる時間について理解する必要がある。

ステークホルダが品質リスク分析プロセスに適切に関与することにより、テストマネージャは重要な利点を得ることができる。この利点とは、要件が不明確または不足しているプロジェクトでも、適切なチェックリストを使用してガイドすると、ステークホルダがリスクを識別できるということである。この利点は、リスクベースドテストを実装して、テストチームの欠陥検出の有効性が改善した場合に確認できる。これが実現するのは、より完全なテストベース（この場合は品質リスクアイテムのリスト）を使用するからである。

リスクベースドテストのテスト終了作業時に、テストチームはその利点を実現できた範囲を測定する必要がある。多くの場合、この測定では、メトリクスおよびチームとのコンサルテーションを通じて、次の 4 つの質問の一部または全部に答えるという作業が行われる。

- テストチームは、重要度の低い欠陥よりも高い割合で重要度の高い欠陥を検出したか？
- テストチームは、テスト実行期間の早期に重要な欠陥のほとんどを検出したか？
- テストチームは、テスト結果をリスクの観点でステークホルダに説明できたか？
- テストチームによりスキップしたテストがあった場合、そのテストは実行したテストよりも関連するリスクのレベルが低かったか？

ほとんどの場合、成功したリスクベースドテストでは、4 つの質問すべてで肯定的な回答が得られる。テストマネージャは長期にわたって、品質リスク分析プロセスの効率性の改善に努めるとともに、これらのメトリクスに関してプロセスの改善目標を設定する必要がある。もちろん他のメトリクスと成功のための基準もリスクベースドテストに適用可能である。テストマネージャはこれらのメトリクス、テストチームが達成する戦略的目的、および特定のメトリクスと成功のための基準に基づくマネジメントによって生じる活動との関係を、慎重に考慮する必要がある。

2.3.3 テストを選択するためのその他の技法

多くのテストマネージャはリスクベースドテストをテスト戦略の 1 つの要素として採用しているが、一方、他の技術も採用しているテストマネージャも多数存在する。

テスト条件を作成し優先度付けするための代替技法としてもっとも優れたものの 1 つが、要件ベースドテストである。要件ベースドテストでは、曖昧性レビュー、テスト条件分析、原因結果グラフなどのいくつかの技法を活用できる。曖昧性レビューでは多くの場合、一般的な要件の欠陥チェックリストを使用することにより、（テストベースの役割を果たす）要件の曖昧性を識別し除去する（[Wiegers03]を参照）。

[Craig02]で説明したように、テスト条件分析では、要求仕様を詳細に読み、カバーするテスト条件を識別する作業を行う。これらの要件に優先度が割り当てられている場合は、この優先度を使用して工数を割り当て、テストケースに優先度付けすることができる。優先度が割り当てられていない場合は、要件ベースドテストとリスクベースドテストをあわせて使用しないと、テストの適切な工数と順序を決定するのは困難である。

原因結果グラフについては、テスト分析の一環としてテスト条件の組み合わせをカバーするという内容に関して、Advanced Test Analyst の資格種別で説明する。非常に大きなテストの問題をマネジメント可能な数のテストケースに減らし、テストベースの機能を 100%カバーできるという広範な用途がある。また、原因結果グラフは、テストケースの設計時にテストベースのギャップを識別する。これにより、ドラフトの要件に基づいてテスト設計を開始した際に、ソフトウェア開発ライフサイクルの早期に欠陥を識別できる。原因結果グラフの採用における主な障害の 1 つは、グラフ作成が複雑なことである。手動で作成すると複雑になるので、それを支援するツールが用意されている。

要件ベースドテストに対する一般的な障害は、多くの場合、要求仕様が曖昧、検証不能、不完全、または存在しないことである。すべての組織がこれらの問題の解決に意欲的であるわけではないので、このような状況に直面したテスト担当者は、別のテスト戦略を選択する必要がある。

別の方法として、要件を拡張するために、既存の要件を使用することに加え、利用方法または運用プロファイルの作成をすることがある。つまり、システムの実際の使用状況を正確に描けるように、ユースケース、ユーザ（ペルソナとも呼ばれる）、入力、および出力を組み合わせて活用するというモデルベースのアプローチである。これにより、機能だけではなく、使用性、相互運用性、信頼性、セキュリティ、および性能もテスト可能となる。

テスト分析およびテスト計画作業では、テストチームは利用方法プロファイルを識別し、テストケースを使用してそのプロファイルを網羅しようとする。利用方法プロファイルは、利用可能な情報に基づいた、ソフトウェアの実際の利用状況の見積りである。つまり、リスクベースドテストと同様に、利用方法プロファイルも最終的な実際の結果を完全にはモデル化できない可能性がある。ただし、十分な情報とステークホルダの入力が使用できる場合は、適切なモデルを作成できる（[Musa04]を参照）。

テストマネージャによっては、何をテストするか、どれくらいテストするか、およびどのような順序でテストするかを決定するために、チェックリストなどの系統的アプローチを適用する。非常に安定したプロダクトの場合、テストする主な機能領域と非機能領域のチェックリストと、既存のテストケースのリポジトリとを組み合わせることで十分な可能性がある。チェックリストは通常、発生した変更の種類と量に基づいて、工数の割り当てとテストの順序付けに関する経験則を提供する。このようなアプローチが、ごくわずかな変更以外のテストで使用される場合、有効性が低下する傾向がある。

最後に、一般的なもう 1 つの方法は、対処的アプローチを使用することである。対処的アプローチでは、テスト実行の前には、テストの分析作業、設計作業、または実装作業をほとんど行わない。テストチームは、実際に提供されたプロダクトに対する対応に焦点を当てる。バグの偏在を検出すると、そこでさらなるテストを実施する。優先度付けと割り当ては、完全に動的に行う。対処的アプローチは他のアプローチの補完として機能するが、単独で採用した場合は、重要ではあるが少数のバグしか発生していないアプリケーションの主要な領域を見逃す傾向がある。

2.3.4 テストプロセスにおけるテストの優先度付けと工数の割り当て

テストマネージャは、どのような技法または技法の組み合わせを使用する場合でも、それらをプロジェクトとテストプロセスに組み込む必要がある。たとえば、シーケンシャルライフサイクル（たとえば V モデルなど）では、テストチームは定期的な調整を行いつつ、要件フェーズでテストの選択とテスト工数の割り当てを行い、最初にテストの優先度付けを行う。一方、イテレーティブライフサイクルまたはアジャイルライフサイクルでは、イテレーションごとのアプローチが必要になる。テストの計画作業とコントロールでは、リスク、要件、および利用方法プロファイルが増大する程度を考慮し、それに対応する必要がある。

テストの分析、設計、および実装では、テスト計画作業で決定された割り当てと優先度付けが適用されるべきである。この情報はテストプロセスを進めていくガイドに使われるだけでなく、注意深い分析やモデリングを行う目的で、通常テストプロセスの中で詳細化する。この詳細化は通常、設計および実装で発生する。

テスト実行では、テスト計画作業で決定した優先度に従ってテストを実行すべきである。ただし、計画を作成した後で得られた情報に基づいて、定期的に優先度付けを更新することも重要である。テスト結果と終了基準ステータスを評価しレポートする場合、テストマネージャは、リスク、要件、利用方法プロファイル、およびチェックリストに関して、評価し、レポートする必要がある。さらに、テストを選択し優先度付けするために使用するそれ以外のガイドについても、評価し、レポートする必要がある。必要に応じて、テストの優先度付け方式に基づいて、テストのトリアージ（実行順序判定）を行う必要がある。

結果レポートおよび終了基準評価の一環として、テストマネージャはどの程度テストが完了しているかを測定できる。この測定では、テストケースと検出した欠陥が関連するテストベースにまで遡って追跡する。たとえば、リスクベースドテストでは、テストを実行して欠陥を検出すると、テスト担当者は残存する未解決のリスクのレベルを

調べることができる。これは、リスクベースドテストを使用して、正しいリリース時期を決定するのに役立つ。テストレポートでは、実現された利点と実現されていない利点の他、対応済みのリスクと未対応のリスクを掲載する必要がある。リスクカバレッジに基づいたテスト結果レポートの例については、[Black03]を参照されたい。

最後に、テストマネージャはテスト終了作業で、テストステークホルダのニーズと期待に関連するメトリクスと成功のための基準を評価する必要がある。これには品質に関する顧客およびユーザのニーズと期待を含む。テストがこれらのニーズと期待を満たした場合のみ、テストチームが本当に有効に機能したと言える。

2.4 テストドキュメントとその他の成果物

多くの場合、テストドキュメントはテストマネジメント活動の一環として作成される。テストマネジメントドキュメントの具体的な名前や各ドキュメントの対象範囲はさまざまであるが、一般的に組織やプロジェクトには、次の種類のテストマネジメントドキュメントが存在する。

- テストポリシー – 組織のテストに関する目的と目標を記述する。
- テスト戦略 –プロジェクトに依存しない組織の一般的なテスト方法を記述する。
- マスターテスト計画（またはプロジェクトテスト計画） – 特定のプロジェクトに関するテスト戦略の実装について記述する。
- レベルテスト計画（またはフェーズテスト計画） – 各テストレベルで実行する特定の活動を記述する。

これらの種類のドキュメントのうち実際にどれを準備するかは、状況によって異なる。ある組織やプロジェクトでは 1 つのドキュメントに統合し、別の組織では別々のドキュメントにする場合がある。また、別の組織では、内容の一部が直観的な文章であったり、そもそも記述されなかったり、あるいは伝統的なテスト方法として明記する場合がある。大規模でより正式な組織およびプロジェクトの場合、全ドキュメントを成果物とする傾向が強く、小規模で公式度合いが低い組織およびプロジェクトの場合、これらの中のいくつかだけを成果物とする傾向が強い。実際には組織およびプロジェクトの状況により各ドキュメントの正しい利用方法を決めるが、本シラバスでは、上記の分類で各ドキュメントを説明する。

2.4.1 テストポリシー

テストポリシーは、組織がテストを行う理由を記述する。つまり、組織が達成する必要があるテストの全体的な目的を定義する。このポリシーは、組織の上級テストマネジメントスタッフが、テストステークホルダグループの上級マネージャと協力して、作成する必要がある。

場合によっては、テストポリシーは、広い意味での品質ポリシーを補完する、もしくはその一部となる。この品質ポリシーは、品質に関連するマネジメントの全体的な価値と目標を記述する。

テストポリシーでは、上位のドキュメントとして、以下の内容を要約して記述する。

- 組織がテストから得られる価値を要約する。
- ソフトウェアの確信度合いの構築、ソフトウェアの欠陥の検出、品質リスクのレベルの軽減など、テストの目的を定義する（2.3.1 節を参照）。
- これらの目的を達成するためのテストの有効性と効率性を評価する方法について記述する。
- ベースとして一般的なテストプロセスの概要を記述する。たとえば ISTQB が示す基本的なテストプロセスを使用するなど。
- 組織がテストプロセスを改善する方法を指定する（第 5 章を参照）。

テストポリシーは、新規開発用に加えて、保守用のテスト活動にも対応する必要がある。また、組織全体で使用するテスト成果物および用語の内部標準または外部標準として参照することもできる。

2.4.2 テスト戦略

テスト戦略は、組織の一般的なテスト方法を記述している。プロダクトおよびプロジェクトのリスクマネジメント、テストレベルへの分割、およびテストに関する上位の活動を含む（組織によっては、ソフトウェア開発ライフサイクル、リスクのレベル、または規制上の要件が異なる場合、異なる戦略を採用することがある）。テスト戦略の他、テスト戦略に記載されるプロセスおよび活動も、テストポリシーと一貫していなければならない。また、組織または 1 つ以上のプログラムのために、一般的なテスト開始基準とテスト終了基準を提示する必要がある。

すでに述べたように、テスト戦略は次のような一般的なテスト方法を記述する。

- 分析的戦略（リスクベースドテストなど）。テストチームはテストベースを分析して、カバーするテスト条件を識別する。たとえば、要件ベースドテストでは、テスト分析により要件からテスト条件を導き出し、これらの条件をカバーするようにテストを設計し実装する。その後、各テストによりカバーする要件の優先度に基づいて、テストを実行する順序を決め、テストを実行する。テスト結果は、たとえば、要件をテストし合格した、要件をテストし不合格となった、要件をまだ完全にテストしていない、要件のテストがブロックされている、などの要件のステータスに関してレポートする。
- モデルベースド戦略（運用プロファイルなど）。テストチームは（実際の状況または予想される状況に基づいて）、システムが存在する環境、システムに対する入力と条件、および本来のシステムの動作方法の各モデルを作成する。たとえば、急成長のモバイルデバイスアプリケーションのモデルベースド性能テストでは、現在の使用状況と今後の想定される伸び具合に基づいて、受信と送信のネットワークトラフィック、アクティブユーザと非アクティブユーザ、および結果的に生じる処理負荷の各モデルを開発する。さらに、現在の本番環境のハードウェア、ソフトウェア、データ容量、ネットワーク、およびインフラストラクチャを考慮してモデルを開発する場合がある。また、スループット、応答時間、およびリソース割り当てに関して、理想的なモデル、予想されるモデル、および最小のモデルを作成することも可能である。
- 系統的戦略（品質特性ベースのものなど）。テストチームは品質標準（たとえば ISO 9126 [ISO9126] から改訂中の ISO 25000 [ISO25000]など）の事前に決定した一連のテスト条件、チェックリスト、あるいは特定のドメイン、アプリケーション、またはテストのタイプ（たとえばセキュリティテストなど）に関連する汎用的かつ論理的な一連のテスト条件を使用する。また、イテレーションから次のイテレーション、またはリリースから次のリリースまでの一連のテスト条件も使用する。たとえば、シンプルで安定した e-コマース Web サイトの保守テストでは、テスト担当者は主要な機能、属性、および各ページに対するリンクを識別するチェックリストを使用する場合がある。テスト担当者は、このサイトに対して変更を行うたびに、このチェックリストの関連する要素をカバーする。
- プロセス準拠または規格準拠戦略（米国食品医薬品局の規定の対象となる医療システムなど）。テストチームは規格委員会または他の専門家達が定義した一連のプロセスに従う。このプロセスでは、文書化、テストベースとテストオラクルの適切な識別と使用、およびテストチームの編成を行う。たとえば、スクラムアジャイルマネジメント技法に準拠するシステムでは、テスト担当者は各イテレーションにおいて、特定のフィーチャを説明するユーザストーリーを分析し、そのイテレーションの計画作業プロセスの一環として各フィーチャのテスト工数を見積り、各ユーザストーリーに対するテスト条件（通常、受け入れ基準と呼ばれる）を識別し、これらの条件をカバーするテストを実行して、テスト実行時の各ユーザストーリーのステータス（未テスト、不合格、または合格）を報告する。
- 対処的戦略（欠陥をベースにした攻撃の利用など）。テストチームはソフトウェアを受け取るまでテストの設計と実装を待ち、テスト対象の実際のシステムで対処する。たとえば、メニューベースアプリケーションで探索的テストを使用する際は、フィーチャ、メニューの選択、および画面に対応する一連のテストチャータを開発する。各テスト担当者には一連のテストチャータが割り当てられ、それを使用して探索的テストセッションを構築する。テスト担当者はテストセッションの結果を定期的にテストマネージャに報告し、その結果に基づいてテストマネージャはチャータを変更する場合がある。
- コンサルテーションベースの戦略（ユーザ主導のテストなど）。テストチームは一人以上の主なステークホルダの入力に依存して、カバーするテスト条件を決定する。たとえば、Web ベースアプリケーション

のアウトソース互換性テストでは、企業はアウトソーステストサービスプロバイダに、評価するアプリケーションの優先度付けしたリストを提供する。このリストには、ブラウザバージョン、マルウェア対策ソフトウェア、オペレーティングシステム、接続の種類、およびその他の構成オプションを含む。テストサービスプロバイダはペアワイズテスト（優先度が高い場合）、同値分割法（優先度が低い場合）などの技法を使用して、テストを生成する。

- 回帰的テスト戦略（広範囲の自動化など）。テストチームは回帰のリスクをマネジメントするさまざまな技法（特に 1 つ以上のレベルにおける機能／非機能回帰テストの自動化）を使用する。たとえば、Web ベースアプリケーションの回帰テストを行う場合、テスト担当者は GUI ベースのテスト自動化ツールを使用して、アプリケーションの一般的および例外的なユースケースを自動化する。これらのテストを、アプリケーションの変更時に常に実行する。

異なる戦略を組み合わせることもある。選択する特定の戦略は、組織のニーズや手段に適合していなければならず、個々の運用やプロジェクトに合わせて戦略を調整することもある。

また、テスト戦略では実行するテストレベルを記述することもある。この場合、各テストレベルの開始基準と終了基準および、各テストレベル間の開始基準と終了基準の関係について、ガイダンスを示すべきである（たとえば、テストレベル間でのテストカバレッジの目的の割り振りなど）。

テスト戦略では、以下の内容を示すこともある。

- 統合手順
- テスト仕様化技法
- テストの独立性（テストレベルによって異なることがある）
- 必須の標準および任意の標準
- テスト環境
- テスト自動化
- テストツール
- ソフトウェア成果物およびテスト成果物の再利用性
- 確認テスト（再テスト）および回帰テスト
- テストのコントロールおよびレポート
- テストの測定指標およびメトリクス
- 欠陥マネジメント
- テストウェアの構成管理アプローチ
- 役割と責任

テスト戦略では、短期向けおよび長期向けの戦略が必要となる場合がある。組織やプロジェクトによっても、適切なテスト戦略は異なる。たとえば、セキュリティクリティカルやセーフティクリティカルなアプリケーションを含む場合、他のシステムよりも徹底的な戦略が適切になることもある。さらに、開発モデルによってもテスト戦略は異なる。

2.4.3 マスターテスト計画

マスターテスト計画では、特定のプロジェクトで実行するすべてのテスト作業を対象とする。実行する特定のレベルと、それらのレベル間の関連、およびテストレベルと対応する開発活動との関係もこれに含む。マスターテスト計画では、特定プロジェクトへのテスト戦略の実装方法（テストアプローチ）を記述する。また、テストポリシーおよびテスト戦略との間で整合性がとれていなければならず、整合性がとれない箇所では、その逸脱や例外について記述する必要がある（逸脱により引き起こされる可能性がある影響を含む）。たとえば、組織のテスト戦略では、リリース直前に未変更のシステムでの回帰テストで完全に合格することが必要であるが、現在のプロジェクトでは回帰テストは行わない予定の場合、テスト計画では、このように計画した理由と、この戦略の相違による

リスクを軽減するために行う対応を記述する必要がある。また、この相違から予想されるその他の影響も記述する必要がある。たとえば、回帰テストをスキップすると、プロジェクトで最初にリリースしてから 1 カ月後に保守リリースを計画することが必要になる場合がある。マスターテスト計画では、より大きなプロジェクトまたは運用の一部としてテスト活動を記述し、プロジェクト計画や運用ガイドを補完する必要がある。

マスターテスト計画の個別の内容や構造は、組織、その文書化規格、およびプロジェクトの形式によって変動する場合があるが、マスターテスト計画の代表的な内容には次のようなものがある。

- テストするアイテムおよびテストしないアイテム
- テストする品質特性およびテストしない品質特性
- テストスケジュールおよび予算（プロジェクトまたは運用予算と整合しなければならない）
- テスト実行サイクルおよびソフトウェアリリース計画との関連
- テストを行う人々や部署と他の人々や部署との関連性と提出書類
- 記述している各レベルで、どのテストアイテムが範囲内でどれが範囲外かを示す定義
- それぞれのテストレベルに対する個別の開始基準、継続（中止／再開）基準、終了基準、およびテストレベル間の関連性
- テストプロジェクトリスク
- テスト活動の全体的な管理
- 各テストレベルを実行する責任の所在
- 各テストレベルにおける入力と出力

テストの 1 つのレベルのみを公式とするような小規模のプロジェクトまたは運用の場合、マスターテスト計画と公式なテストレベルのテスト計画は 1 つのドキュメントでまとめて記述することが多い。たとえば、開発者が実施するコンポーネントおよび統合テスト、顧客がベータテストプロセスの一環として実施する受け入れテストなどを非公式で行い、システムテストが唯一の公式なテストレベルである場合、システムテスト計画に、この節で記述している要素を入れることができる。

また、テストは一般的にプロジェクトの他の活動に依存する。これらの他の活動を十分に記述していない場合、特にテストとの影響や関連について記述していない場合、これらの活動に関連する内容をマスターテスト計画（または適切なレベルテスト計画）でカバーすることになる。たとえば、構成管理プロセスを記述していない場合、テスト対象をテストチームに渡す方法について、テスト計画で定義する必要がある。

2.4.4 レベルテスト計画

レベルテスト計画では、それぞれのテストレベルで実行する特定の活動を記述する。テストタイプごとで記述する場合もある。必要に応じて、記述対象である個別のテストレベルまたはテストタイプに関して、マスターテスト計画を詳細化したものもこれに含む。マスターテスト計画でカバーしていないスケジュール、タスク、マイルストンの詳細についての情報をレベルテスト計画で提供する。さらに、これらの詳細は、さまざまな規格やテンプレートを、状況に応じてテストに適用するため、レベルテスト計画でカバーすることになる。

公式度合いが低いプロジェクトや運用の場合、単一のテスト計画が、記述する唯一のテストマネジメントドキュメントになることも多い。このような状況では、これまでにこの節で記述している情報要素のいくつかを、このテスト計画ドキュメントでカバーすることもできる。

アジャイルプロジェクトでは、スプリントテスト計画またはイテレーションテスト計画が、レベルテスト計画の代わりになる場合がある。

2.4.5 プロジェクトリスクマネジメント

プロジェクトリスクに対処することは、適切な計画作業の重要な部分である。プロジェクトリスクは、2.3 節で説明したものと同様のプロセスで識別できる。プロジェクトリスクが識別されている場合、プロジェクトマネージャに通知し、その指示に従う必要がある。このようなリスクは、必ずしもテスト組織の力で軽減できるわけではない。しかし、次のような一部のプロジェクトリスクについては、テストマネージャによって軽減することが可能であり、対処すべきこととなる。

- テスト環境およびツールの準備
- テストスタッフの調達能力と資質
- テスト活動に対する標準類、ルールおよび技法の欠如

プロジェクトリスクマネジメントへのアプローチには、早期のテストウェアの準備、テスト環境の事前テスト、プロダクトの初期バージョンに対する事前テスト、テストの開始基準の厳格化、試験性要件の強化、早期のプロジェクト成果物のレビューへの参加、変更管理への参加、プロジェクトの進捗および品質のモニタリングなどを含む。

プロジェクトリスクを識別し分析した後は、次の主な 4 つのオプションでそのリスクをマネジメントする。

1. 可能性や影響を減らす予防対策でリスクを軽減する。
2. コンティンジェンシープランを作成して、リスクが現実化した場合の影響を軽減する。
3. リスクを別の部門に移して対処する。
4. リスクを無視する、または受け入れる。

オプションによって発生する利点と機会だけでなく、コストおよび場合によってはオプションによって発生する追加リスクに基づいて、最適なオプションを選択する。プロジェクトリスクに対するコンティンジェンシープランが特定された場合、ベストプラクティスはトリガー（コンティンジェンシープランを実行する条件と方法を決定する）と所有者（コンティンジェンシープランを実行する人）を識別することである。

2.4.6 その他のテスト成果物

テストでは、欠陥レポート、テストケース仕様、テストログなど、その他の成果物を多数作成する。これらの成果物のほとんどは、テストアナリストとテクニカルテストアナリストが作成する。Test Analyst シラバスでは、これらの成果物の作成および文書化に関する考慮事項について説明している。テストマネージャは、次の活動を通じて、これらの成果物の一貫性と品質を確保する必要がある。

- 拒否された欠陥レポートの割合など、これらの成果物の品質を監視するメトリクスを確立しモニタリングする。
- テストアナリストおよびテクニカルテストアナリストと協力して、これらの成果物の適切なテンプレートを選択しカスタマイズする。
- テストアナリストおよびテクニカルテストアナリストと協力して、テスト、ログ、およびレポートで必要となる詳細度合いなど、これらの成果物の標準を確立する。
- 適切な技法を使用して、適切な参加者およびステークホルダによるテスト成果物のレビューを行う。

テスト文書化の範囲、種類、および特性は、いくつかの考慮事項の中で特に、選択したソフトウェア開発ライフサイクル、適用する標準と規制、および開発される特定のシステムに関連するプロダクト品質とプロジェクトリスクの影響を受けることがある。

テスト成果物のテンプレートは、IEEE 829 [IEEE829]などのさまざまな資料に基づいて作成できる。テストマネージャは、IEEE 829 のドキュメントはどの業界でも使用できるように設計されていることを念頭に置く必要がある。そのため、テンプレートには詳細度の高い内容を含んでおり、特定の組織に適用できる場合とできない場合がある。IEEE 829 のドキュメントを改訂して、特定の組織で使用できる標準テンプレートを作成するのがベス

トプラクティスである。テンプレートを一貫して適用することで、トレーニングの必要性が減少し、組織全体でプロセスの統一化を達成する。

また、テストでは、通常テストマネージャが作成するテスト結果レポートも作成する。これについては、本章内の以降で説明する。

2.5 テストの見積り

マネジメント活動として考える場合、見積りは特定の運用またはプロジェクトに関わる活動に対応するコストおよび終了日の近似値を作成することになる。優れた見積りには次のような特徴がある。

- 経験を積んだ実務者の英知の結集となり、関連する参加者のサポートも得られる。
- プロジェクトのコスト、リソース、タスク、人について詳細な一覧を提供する。
- 見積り対象のそれぞれの活動に対し、もっとも可能性の高いコスト、工数、期間を算出する。

ソフトウェアエンジニアリングおよびシステムエンジニアリングの見積りは、技術的、政治的に困難を伴うものと長い間考えられてきたが、見積りに対するプロジェクトマネジメントのベストプラクティスが確立されてきている。テストの見積りは、これらのベストプラクティスを、プロジェクトまたは運用に対応するテスト活動に適用することである。

テストの見積りには、第 1 章で説明したテストプロセスに関係するすべての活動を含めなければならない。テスト実行が一般的にプロジェクトのクリティカルパスになるため、見積ったテスト実行のコスト、工数、特に期間はマネジメントにおいてもっとも重要な要素になることが多い。ただし、ソフトウェア全体の品質が低い場合や未知の場合は、テスト実行の見積りを作成することが難しくなり、しかも信頼性に欠けることが多くなる。さらに、システムに関する精通度合いと経験も、見積りの質に影響することがある。一般的には、テスト実行時に必要なテストケースの数も見積らなければならないが、この見積りは、テスト対象ソフトウェアの欠陥数が少ないと想定できる場合のみ機能する。見積りの際に設定した仮定は、見積りの一部として常に記述する必要がある。

テストの見積りでは、テスト活動のコスト、工数、期間に影響をおよぼすあらゆる要素について考慮する必要がある。これらの要素には次のものを含む（ただし、これらのみではない）。

- システムに求められる品質レベル
- テスト対象となるシステムのサイズ
- 過去のテストプロジェクトの履歴データ（他の組織の業界データまたはベンチマークデータを含む場合もある）
- プロセス要因。テスト戦略、開発または保守ライフサイクルおよびプロセスの成熟度、プロジェクト見積りの正確性など
- 物的要因。テスト自動化およびツール、テスト環境、テストデータ、開発環境、プロジェクトドキュメンテーション（たとえば要件、設計など）、再利用可能なテスト成果物など
- 人的要因。マネージャおよび技術リーダ、管理職および上級管理職の責務と期待、プロジェクトチームのスキル、経験、態度、プロジェクトチームの安定性、プロジェクトチームの関係、テストとデバッグ環境サポート、優れた請負業者やコンサルタントの利用可能性、ドメインの知識など
- プロセス、技術、組織の複雑さ、テストに関わるステークホルダの数、サブチームの構成と場所
- 著しい人員増によるトレーニングやオリエンテーションの必要性
- 新しいツール、技術、プロセス、技法、カスタムハードウェア、各種のテストウェアの適用や開発
- 詳細度合いが高いテスト仕様の作成要求（特に馴染みのない文書化標準の場合）
- コンポーネントの受け入れが時間的に前後する場合（特に統合テストやテスト開発の場合）
- 使える範囲の限られたテストデータ（たとえば時間の影響を受けるデータなど）

また、テストに提供されるソフトウェアの品質も、テストマネージャが見積りで考慮すべき大きな要因である。たとえば、開発者が自動化したユニットテストや継続的インテグレーションなどのベストプラクティスを採用していた場合、コードがテストチームに提供される前に 50%もの欠陥が除去される（これらの実践による欠陥除去の有効性については[Jones11]を参照）。テスト駆動開発などのアジャイル方式では、テストに対して高い品質レベルが提供されていることが、一部で報告されている。

見積りは、ボトムアップとトップダウンのどちらでも行うことができる。テストの見積りでは次の技法を、単独でまたは組み合わせて使用できる。

- 直感、推測、および過去の経験
- ワークブレークダウンストラクチャ（WBS）
- チーム見積りセッション（たとえばワイドバンドデルファイ）
- 企業の標準および基準
- プロジェクトの総工数またはスタッフ構成の割合（たとえばテスト担当者と開発者の割合）
- 組織のデータ蓄積およびメトリクス（欠陥の数、テストサイクルの数、テストケースの数、各テストの平均工数、関連する回帰テストサイクルの数を見積るメトリクス抽出モデルを含む）
- ファンクションポイント、コードの行数、見積った開発者の工数、他のプロジェクトパラメータなどの、業界平均および予測モデル（たとえば[Jones07]を参照）

ほとんどの場合、見積りを一度用意したら、根拠を添えて管理担当者に送らなければならない（2.7 節を参照）。多くの場合、協議を重ねることにより、見積りの再作成が必要になる。理想的には最終的なテストの見積りは、品質、スケジュール、予算、フィーチャの点で、組織とプロジェクトの目標において優れたバランスを表したものとなる。

すべての見積りが、その時点で利用可能だった情報に基づいていることに注意すべきである。プロジェクトの初期では、十分な情報が利用できないことがある。さらに、プロジェクトの初期に利用可能な情報は、時間が経つと変化する可能性がある。正確性を維持するには、新しい情報と変更された情報に基づいて、見積りを更新する必要がある。

2.6 テストメトリクスの定義および使用

マネジメントの世界には、「測定できるものは達成できる」という決まり文句がある。さらに、「測定されないものは実行されない」というのもあり、測定されないものが無視されやすいことを示している。このため、テストを含めてすべての活動で適切なメトリクスセットを確立する必要がある。

テストメトリクスは、次の 1 つ以上のカテゴリに分類される。

- プロジェクトメトリクス。実行済み、合格、不合格のそれぞれのテストケースの割合など、確立しているプロジェクト終了基準に対する進捗を測定する。
- プロダクトメトリクス。テスト範囲、欠陥密度など、プロダクトのいくつかの属性について測定する。
- プロセスメトリクス。テストで検出された欠陥の割合など、テストプロセスまたは開発プロセスの能力を測定する。
- 人的メトリクス。指定されたスケジュールでのテストケースの実施など、個人またはグループの能力を測定する。

それぞれのメトリックは、2 つ、3 つ、または 4 つのカテゴリに属することがある。たとえば、日々の欠陥検出率を示す傾向グラフは、終了基準（新しい欠陥が検出されない日が 1 週間継続する）、プロダクトの品質（テストでそれ以上の欠陥は見つけられない）、およびテストプロセスの能力（テスト実行期間の早期に大量の欠陥を見つける）と関連付けることができる。

人的メトリクスは、特に慎重に扱うべきである。マネージャが本来はプロセスメトリクスであるメトリクスを人的メトリクスとして誤り、担当者がメトリクスを自分に有利となるように解釈して、破滅的な結果となることがある。テスト担当者の適切な動機付けとアセスメントについては、本シラバスの第 7 章と Expert Test Management シラバス［ISTQB ETL SYL］で詳細に説明する。

Advanced Level では、ほとんどの場合、テストの進捗を測定するメトリクス、つまりプロジェクトメトリクスの使用に重点を置く。テスト進捗の測定に使用するプロジェクトメトリクスの一部は、プロダクトメトリクスおよびプロセスメトリクスにも関連する。プロダクトメトリクスおよびプロセスメトリクスのマネジメントを目的として使用するための詳細な情報については、Expert Test Management シラバスで述べている。プロセスメトリクスの使用に関する詳細な情報については、Expert Improving the Test Process シラバス[ISTQB ITP SYL]で述べている。

テスト担当者はメトリクスを利用することにより、継続的に結果を報告することができ、それらの状況を一貫して追跡できる。テストマネージャは、技術スタッフからマネジメント層に至る多くの層のステークホルダが参加するさまざまな会議で、メトリクスを提示することが頻繁に求められる。場合によっては、メトリクスはプロジェクトの全体的な成功を判定するために使用するので、追跡の対象とするもの、報告する頻度、およびその情報を提示するために使用する方法を、入念に考慮して決定する必要がある。特に、テストマネージャは、次のことについて考慮しなければならない。

- メトリクスの定義：メトリクスは有用なものに限定する必要がある。メトリクスは、プロジェクト、プロセス、プロダクトごとの目的に従って定義しなければならない。単一のメトリックではステータスや傾向に関して誤って解釈される可能性があるので、メトリクスはバランスをとって定義しなければならない。メトリクスを定義する上で、後でメトリクスの値だけが独り歩きして誤解が生じたりしないように、その解釈については、すべてのステークホルダからの合意を求める。定義するメトリクスは、多くの場合、適切な数よりも多くなりすぎる傾向がある。
- メトリクスの追跡：メトリクスのレポートやメトリクスの集計に要する時間を短縮するために、生データからメトリクスを算出するまでを極力自動化すると良い。ある特定のメトリックにおいて時間の経過によりデータが変化する場合、メトリックを定義した段階で合意した解釈以外の情報が影響していることもある。テストマネージャは、測定値が期待している値から逸脱する可能性と、その逸脱の理由を入念に分析しなければならない。
- メトリクスのレポート:レポートの目的は、マネジメントのために、速やかに理解できる情報を提供することである。特定の時期での「スナップショット」や、時間の経過に伴うメトリックの変化を提示することにより、傾向を評価できる。
- メトリクスの有効性：テストマネージャは、報告された情報を検証しなければならない。メトリック用に取得された測定値が、プロジェクトの本当のステータスを反映していなかったり、過度に良い傾向または悪い傾向を示したりすることがある。テストマネージャは、データを提示する前に、正確性、およびデータから読み取れるメッセージについて確認しなければならない。

テスト進捗のモニタリングの対象には、次の 5 つの主要な要素がある。

- プロダクト（品質）リスク
- 欠陥
- テスト
- カバレッジ
- 確信度合い

プロダクトリスク、欠陥、テスト、カバレッジは、開発プロジェクトまたは運用期間を通して、特定の方法で測定し、報告することが多い。このような測定は、テスト計画で定めた終了基準に関連する場合、テスト作業の完了を判断するための目標基準を提供する。確信度合いは調査を通して、またはカバレッジを代替メトリックとして使用することにより測定できるが、主観的な報告になることが一般的である。

プロダクトリスクに関連するメトリクスには、次のようなものがある。
- テストに合格することによって完全に対応されたリスクの割合
- 一部またはすべてのテストが不合格となるリスクの割合
- テストが完了していないリスクの割合
- リスクカテゴリで分類されたリスクのうち、対応されたリスクの割合
- 最初の品質リスク分析後に識別したリスクの割合

欠陥に関連するメトリクスには、次のようなものがある。
- レポート済みの（検出された）総数と解決済みの（修正された）総数の比
- 平均故障間隔（MTBF）または故障率
- 次のように分類した、欠陥の数または割合：
  - 特定のテストアイテムまたはコンポーネント
  - 根本原因
  - 欠陥の起源（たとえば、要求仕様、新しいフィーチャ、リグレッションなど）
  - テストリリース
  - 混入／検出／除去されたフェーズ
  - 優先度／重要度
  - 拒否されたレポートまたは重複レポート
- 欠陥のレポートから解決に至るまでのタイムラグの傾向
- 新たな欠陥（daughter bug とも呼ばれる）を発生させた欠陥修正の数

テストに関連するメトリクスには、次のようなものがある。
- 計画、仕様化（実装）、実行、合格、不合格、ブロック、スキップしたテストの総数
- 回帰テストおよび確認テストのステータス（不合格となった回帰テストおよび確認テストの傾向および総数を含む）
- 計画している 1 日当たりのテスト時間と実際のテスト時間との比
- テスト環境の可用性（計画されたテスト時間のうち、テストチームがテスト環境を使用できる時間の割合）

テストカバレッジに関連するメトリクスには、次のようなものがある。
- 要件および設計要素のカバレッジ
- リスクのカバレッジ
- 環境／構成のカバレッジ
- コードカバレッジ

テストマネージャは、テストステータスを理解し、レポートするために、カバレッジメトリクスを解釈し、使用する方法を理解しなければならない。システムテスト、システム統合テスト、受け入れテストなどの上位レベルのテストでは、主要なテストベースは、通常、要求仕様、設計仕様、ユースケース、ユーザストーリー、プロダクトリスク、サポートされる環境、サポートされる構成などの、成果物である。構造コードカバレッジメトリクスは、ユニットテスト（たとえばステートメントカバレッジおよびブランチカバレッジ）やコンポーネント統合テスト（たとえばインターフェースカバレッジ）などの下位レベルのテストに適用する。テストマネージャは、コードカバレッジメトリクスを使用して、テスト対象のシステムの構造まで測定できるが、上位レベルのテスト結果のレポートは、通常、コードカバレッジメトリクスに影響されてはならない。さらに、テストマネージャは、ユニットテストおよびコンポーネント統合テストが構造カバレッジ目標の 100%を達成する場合でも、欠陥と品質リスクは上位レベルのテストで対処しなければならない。

また、メトリクスは、Foundation シラバスと本シラバスで説明している基本的なテストプロセス活動にリンクすることができる。これにより、メトリクスはテストプロセス全体で、テストプロセス自体とプロジェクト目標への進捗をモニタリングするために使用できる。

テスト計画をモニタリングし、活動をコントロールするためのメトリクスには、次のようなものがある。
- リスク、要件、および他のテストベース要素のカバレッジ
- 欠陥検出
- テストウェアの開発とテストケースの実行に関する計画時間と実時間

テスト分析活動をモニタリングするためのメトリクスには、次のようなものがある。
- 識別されたテスト条件の数
- テスト分析時に検出した欠陥の数（たとえば、テストベースを使用してリスクまたは他のテスト条件を識別することにより検出した数など）

テスト設計活動をモニタリングするためのメトリクスには、次のようなものがある。
- テストケースによりテスト条件を網羅している割合
- テスト設計時に検出した欠陥の数（たとえば、テストベースに対するテストを開発する際に検出した数など）

テスト実装活動をモニタリングするためのメトリクスには、次のようなものがある。
- 構築済みのテスト環境の割合
- ロードしたテストデータレコードの割合
- 自動化したテストケースの割合

テスト実行活動をモニタリングするためのメトリクスには、次のようなものがある。
- 計画したテストケースで、実行、合格、および不合格となったテストケースの割合
- 実行、または合格したテストケースにより網羅したテスト条件の割合
- レポート、または解決した欠陥の計画数と実際の数
- 達成したカバレッジの計画と実際の比

テストの進捗および完了の活動をモニタリングするためのメトリクスには、マイルストン、開始基準、および終了基準（テスト計画内で定義および合意したもの）に対して割り当てられるものがある。これには、次のようなものを含む。
- 計画したテスト条件、テストケースもしくはテスト仕様の数、結果を合格・不合格ごとに分類した数
- 発生した欠陥の総数を、重要度、優先度、現在の状態、影響を受けたサブシステム、または他の方法で分類した数（第 4 章を参照）
- 変更の発生数、受け入れ数、ビルドした数、テスト済みの数
- 計画上のコストと実際のコストとの比
- 計画上の期間と実際の期間との比
- テストのマイルストンのために計画した日数と実際に要した日数
- テスト関連のプロジェクトマイルストン（たとえばコードフリーズなど）のために計画した日数と実際に要した日数
- プロダクト（品質）リスクのステータス、軽減したものと軽減していないものの比、主なリスク領域、テスト分析後に検出された新しいリスクなど
- 進捗を妨げたイベントまたは計画された変更により無駄になったテスト活動、コスト、または時間の割合
- 確認テストと回帰テストのステータス

テスト終了活動をモニタリングするためのメトリクスには、次のようなものがある。

- テスト実行中に実施したテストケース、合格したテストケース、不合格となったテストケース、ブロックしたテストケース、およびスキップしたテストケースの割合
- 再利用可能なテストケースとしてリポジトリにチェックインしたテストケースの割合
- 自動化したテストケースと自動化予定のテストケースの割合
- 回帰テストに統合したテストケースの割合
- 解決済みの欠陥レポートおよび未解決の欠陥レポートの割合
- テスト成果物として識別し、保管したものの割合

さらに、ワークブレークダウンストラクチャなど標準のプロジェクトマネジメント技法を、テストプロセスをモニタリングするために使用する。アジャイルチームでは、テストはバーンダウン・チャートに示されるユーザストーリーの進捗の一部を構成する。リーンマネジメント技法を使用する場合、ストーリーベースのテスト進捗は、カンバンの列を通してユーザストーリーカードを移動させることによってモニタリングする。

一連の定義したメトリクスを使用した測定結果は、口頭による説明、数値を使用した表形式、またはグラフの図で報告できる。測定結果は、次のような目的で使用できる。

- 分析。テスト結果に基づいて、認識される傾向と原因を見つける。
- レポート。テストで気づいたことを関係するプロジェクト参加者およびステークホルダに伝える。
- コントロール。一連のテストまたはプロジェクトを全体として見直し、その結果をモニタリングする。

これらのテスト測定を収集、分析、および報告するための適切な方法は、それらの測定指標を使用するユーザの特定の情報に対するニーズ、目標、および能力によって異なる。また、テストレポートの具体的な内容は、報告対象者によって異なる。

テストコントロールの目的のために、テストプロセス全体を通してのメトリクス（テスト計画が完了した場合）をテストマネージャに提供することが重要である。この情報は、テストの使命、戦略、および目的を成功裏に達成できるような方向へテストを導くために必要である。このため、計画時には必要な情報を考慮し、モニタリングには成果物に関するすべてのメトリクスを収集することを含める必要がある。必要な情報の量と、情報を収集するために費やす活動は、プロジェクトの規模、複雑さ、およびリスクなど、さまざまな要因によって決定する。

テストコントロールは、テストにより生成された情報と、プロジェクトまたはテストの状況の変化に対応する必要がある。たとえば、動的テストにより、多くの欠陥を含むとは予測しなかった領域で大量の欠陥を検出した場合、または、テスト開始の遅延によりテスト実行期間を短縮した場合、リスク分析と計画を見直す必要がある。これにより、テストの優先度の見直しや、残りのテスト実行活動の再割り当てが発生することがある。

テスト進捗レポートによりテスト計画からの逸脱を発見した場合、テストコントロールを実行する必要がある。テストコントロールの目的は、プロジェクトまたはテストが成功に向かうよう軌道修正することである。テスト結果を使用してプロジェクトのコントロール活動に影響を与えるか、活動を測定する際に、次のオプションを考慮する必要がある。

- 品質リスク分析、テストの優先度、およびテスト計画の見直し
- リソースの追加、またはプロジェクト活動やテスト活動の追加
- リリース日の延期
- テスト終了基準の緩和または厳格化
- プロジェクトの対象範囲（機能および非機能）の変更

このようなオプションの実装は、通常、プロジェクトや運用のステークホルダ間の合意と、プロジェクトマネージャや運用マネージャからの同意を必要とする。

テストレポートで提供される情報は、対象者（たとえば、プロジェクトマネージャやビジネスマネージャ）のニーズによって大きく異なる。プロジェクトマネージャにとっては、欠陥についての詳細情報が重要で、ビジネスマネージャにとっては、プロダクトリスクステータスが、報告されるべき重要になる。

2.7 テストのビジネスバリュー

テストマネージャは、優れたビジネスバリューを提供できるように、テストの最適化に努める必要がある。過剰なテストは、不合理な遅延や本来節約できるはずのコストがかかってしまう可能性があるので、優れたビジネスバリューを提供しない。過少なテストは、多くの欠陥をユーザに提供してしまう可能性があるので、優れたビジネスバリューを提供しない。最適なテスト量は、これら 2 つの間に存在する。テストマネージャは、テストのステークホルダが、この最適なテスト量とテストにより提供される価値を理解できるようにする必要がある。

ほとんどの組織では、何らかの意味で価値のあるテストについて検討しているが、この価値の定量化や、記述、または明確な表現を行えるマネージャ（テストマネージャを含む）はほとんどいない。また、多くのテストマネージャ、テストリーダ、テスト担当者は、テストの戦術的な詳細事項（タスクまたはテストレベルに固有の要素）に集中してしまい、他のプロジェクト参加者（特にマネージャ）が気にかけるような、テストに関連するもっと大きい戦略的な上位の問題は無視する。

テストは、組織、プロジェクト、運用に対し、定量的と定性的の両面において価値をもたらす。

- 定量的な価値としては、リリース前に予防／修正する欠陥の検出、リリース前に判明する欠陥の検出（修正していないが回避策と共に文書化している）、テスト実行によるリスクの軽減、プロジェクト／プロセス／プロダクトステータスについての情報提供などが挙げられる。
- 定性的な価値としては、品質による評判の向上、スムーズで予定を立てやすいリリース、確信度合いの向上、法律上の免責、システムが果たすべき使命が台無しになるリスクや、生命損失のリスクの軽減などが挙げられる。

テストマネージャは、組織、プロジェクト、運用において、このうちのどの価値が当てはまるかを理解することで、このような価値の点からテストに関する情報を伝えることができる。

テストの定量的価値および効率性を測定するための確立された方法として、品質コスト（または不良品質コスト）と呼ぶ方法がある。品質コストでは、プロジェクトまたは運用のコストを、プロダクト欠陥コストに関連する次の 4 つのカテゴリに分類する。

- 予防コスト。たとえば、保守性が良く、セキュリティを強化したコードを記述するような開発者へのトレーニング。
- 評価コスト。たとえば、テストケースの記述、テスト環境の構成、要件のレビュー。
- 内部失敗コスト。たとえば、提供前の、テストまたはレビュー期間中に検出した欠陥の修正。
- 外部失敗コスト。たとえば、顧客に提供した欠陥ソフトウェアに関連するサポートコスト。

テスト予算の一部は評価コスト（テスト担当者が欠陥を見つけなかった場合でもテストに要したコストや、テストを開発するために要したコスト）で、残りは内部失敗コスト（検出した欠陥に付随する実際のコスト）になる。評価コストと内部失敗コストの合計は通常、外部失敗コストを十分下回っている（これがテストに優れた価値をもたらしている）。テストマネージャは、この 4 つのカテゴリでコストを判定することにより、テストに対する説得力のあるビジネスケースを作り出すことができる。

品質コストを含む、テストのビジネスバリューの詳細については、[Black03]を参照されたい。

2.8 分散テスト、アウトソーステスト、およびインソーステスト

多くの場合、テスト活動の一部もしくはすべてが、他の会社の従業員やプロジェクトチームとは別の人々よって、異なった地域で行われる。テスト活動が複数の地域で行われる場合、そのテスト活動を「分散テスト」と呼ぶ。テスト活動を、同じ企業の従業員でなく、プロジェクトチームと同じ地域にいない人たちによって、1 つの地域または複数の地域で実行する場合、そのテスト活動を「アウトソーステスト」と呼ぶ。テスト活動を、プロジェクトチームと同じ地域に存在するが同僚の従業員でない人々によって実行する場合、そのテスト活動を「インソーステスト」と呼ぶ。

このようなテスト活動のすべてにおいて、共通して明確なコミュニケーションチャネルの他、使命、業務、提出書類に対する要望を十分にまとめておくことが必要になる。プロジェクトチームは、ホールでの会話やつきあいなどのような非公式のコミュニケーションチャネルにあまり依存してはならない。コミュニケーションを行う方法を定義することが重要である。この定義では、問題のエスカレーション、コミュニケーションされる情報の種類、使用されるコミュニケーション方法などのトピックに対応する必要がある。関係するすべてのチームに属するすべての人々が、自分の役割と責任だけでなく、他のグループの役割と責任も明確に理解して、誤解や非現実的な期待を避ける必要がある。地域、時間帯、文化、言語の違いが、コミュニケーションの問題と、期待との差異による問題を引き起こしやすくすることに注意する。

また、このようなテスト活動のすべてにおいて、方法論の調整が共通して必要になる。方法論の調整に失敗することは、どのプロジェクトでも起こりうることであるが、活動の分散や外部組織による実行の場合に、発生する可能性が高まる。2 つのテストグループが異なる方法を使用したり、テストグループが開発グループやプロジェクトマネジメントグループと異なる方法を使用したりすると、特にテスト実行において大きな問題になる。たとえば、クライアントがアジャイル開発を使用し、テストサービスプロバイダがシーケンシャルライフサイクルを想定する定義済みのテスト方法を所有する場合、テストサービスプロバイダにテストアイテムを提供するタイミングと提供する状態が、論点になる可能性がある。

分散テストの場合、テスト作業を複数の地域に分割することを明示し、これを合理的に決定しなければならない。このような指針がないと、もっとも能力のあるグループが、本来やるべきテスト作業を行わないということも出てくる。さらに、各チームが自分の責任を理解していない場合、人々が、本来やるべきことを行わなくなる可能性がある。各チームへの期待を、明確に伝える必要がある。マネジメントを適切に行わないと、ギャップ（出荷時の品質リスクが増す）や重複（作業効率が低下する）が発生し、テスト作業はうまくいかない可能性がある。

つまり、テスト活動のすべてにおいて、プロジェクトチーム全体の力で、それぞれのテストチームが、組織、文化、言語、地理的な境界を越え役割を正しく果たすという信用を勝ち取り維持することが、きわめて重要になる。このような信用を失うことは、活動の確認、問題に対する責任の分配、組織の政策などに関連する非効率や遅延の原因になる。

2.9 業界標準適用のマネジメント

Foundation シラバスと Advanced Level シラバスの両方で、多くの標準を参照している。参照しているこれらの標準は、ソフトウェア開発ライフサイクル、ソフトウェアテスト、ソフトウェア品質特性、レビュー、欠陥マネジメントを網羅している。テストマネージャは、標準、標準の使用に関する組織のポリシー、標準の使用が必須なのか、必要なのか、または役立つのかどうかを把握する必要がある。

標準の情報源には、以下のようなものがある。

- 国際標準または国際的な標準になることを目的としたもの
- 国内標準、たとえば国際標準を国レベルで適用したものなど

- ドメイン固有の標準、たとえば国際標準または国内標準を特定のドメインに適合したものや、特定のドメイン向けに作成したもの

国際標準には、ISO と IEEE の 2 つの主要な作成団体がある。ISO は、International Standards Organization（国際標準化機構）の略称で、IOS（International Organization for Standardization）とも呼ばれる。各国において、標準化にもっとも適した国家機関を代表するメンバで構成される。この国際機関は、ISO 9126（ISO 25000 に改訂中）、ISO 12207 [ISO12207]、ISO 15504 [ISO15504]など、ソフトウェアテスト担当者にとって有用な、多くの標準を推進してきた。

IEEE は、Institute of Electrical and Electronics Engineer（電気電子学会）の略称で、米国に拠点を置く専門家組織である。世界 100 カ国以上から国の代表が参加している。この組織は、IEEE 829 [IEEE829]や IEEE 1028 [IEEE1028]など、ソフトウェアテスト担当者にとって有用な、多くの標準を提案してきた。

多くの国には、独自の標準がある。これらの標準の一部はソフトウェアテストに適用可能であり、また有用である。英国の国内標準に BS 7925-2 [BS7925-2]がある。この標準では、Advanced Test Analyst シラバスと Advanced Technical Test Analyst シラバスでも説明している多くのテスト設計技法に関連する情報について記述している。

特定の技術ドメイン固有の標準も存在し、これらの標準の一部には、ソフトウェアテスト、ソフトウェア品質、ソフトウェア開発に該当する領域がある。たとえば、航空関連システムのドメインでは、米国連邦航空局の標準 DO-178B（および EU 版の ED 12B）が、民間航空機で使用するソフトウェアに適用される。この標準は、テスト対象のソフトウェアの致命度に基づいて、一定の構造カバレッジ基準を規定している。

他の技術ドメイン固有の標準として、医療システムドメインには、米国食品医薬品局（FDA）の Title 21 CFR Part 820 [FDA21]がある。この標準は、一定の構造的および機能的なテスト技法を推奨している。また、ISTQB シラバスと一貫性のあるテスト戦略と原則を推奨している。

特定の状況では、テストが、テストを主眼としているわけではないが、テストが発生するソフトウェアプロセスの状況に大きな影響をおよぼす標準や、広範に使用している方法の影響を受けることがある。1 つの例が、CMMI® ソフトウェアプロセス改善フレームワークである。このフレームワークには、検証と妥当性確認の 2 つのキープロセスエリアがあり、これらは、テストのレベル（それぞれシステムテストと受け入れテスト）として解釈されることがある。また、テスト戦略を記述した部分もあり、分析的な要件ベースドテストをテスト戦略の一部として包含することを要求している。

他の 3 つの重要な例として、PMI の PMBOK、PRINCE2®、および ITIL®が存在する。PMI と PRINCE2 は、それぞれ北米とヨーロッパで一般的に使用しているプロジェクトマネジメントフレームワークである。ITIL は、IT グループが、自分の所属する組織に価値あるサービスを確実に提供できるようにするフレームワークである。これらのフレームワークで指定している用語と活動は、ISTQB シラバスおよび用語集と大きく異なっている。PMI の PMBOK、PRINCE2、ITIL を使用する組織で活動する場合、テストマネージャは、その状況で効率よく作業するために、選択されているフレームワーク、実装、用語を十分に理解する必要がある。

適用している標準や方法に関係なく、それらは専門家グループが作成していることを認識することが重要である。標準は、ソースグループの経験と知恵の集合だけでなく、弱点も反映している。テストマネージャは、自分の環境と状況に適用される標準が、公式な標準（国際、国内、またはドメイン固有）、社内の標準、または推奨事項のいずれであるかを認識する必要がある。

複数の標準の使用を検討する場合、一部の標準は他の標準との間で一貫性がなかったり、矛盾した定義を提供したりすることに留意する必要がある。テストマネージャは、テストを行う特定の状況で異なる標準を使用する

ことの有用性を判断しなければならない。標準で記載されている情報が、プロジェクトに役立つことも、プロジェクトの妨げになることもある。ただし、標準は、実績のあるベストプラクティスと、テストプロセスを編成するための基本を提供する。

特定の状況では、標準への準拠が必須であり、標準に従ってテストを行う。テストマネージャは、そのような要件を認識して、標準に準拠し、準拠を適切に維持する。

3. レビュー – 180 分

用語

監査、非公式レビュー、インスペクション、マネジメントレビュー、モデレータ、レビュー、レビュー計画、レビューア、テクニカルレビュー、ウォークスルー

「レビュー」の学習の目的

3.2 マネジメントレビューと監査

TM-3.2.1　（K2）マネジメントレビューと監査の主な特徴を理解する。

3.3 レビューのマネジメント

TM-3.3.1　（K4）プロジェクトを分析して適切なレビューの種類を選択し、レビューの実行、フォローアップ、説明責任が確実に行われるように、レビュー実施計画を定義する。

TM-3.3.2　（K2）レビュー参画のために必要な要素、スキル、時間を理解する。

3.4 レビューのためのメトリクス

TM-3.4.1　（K3）レビューで使用するプロセスメトリクスとプロダクトメトリクスを定義する。

3.5 公式レビューのマネジメント

TM-3.5.1　（K2）公式レビューの特徴を、例を使用して説明する。

3.1 イントロダクション

ISTQB Foundation Level シラバスでは、レビューをプロダクトに対する静的テスト活動として紹介をした。監査とマネジメントレビューは、ソフトウェア成果物ではなくソフトウェアプロセスに、より重点を置いている。

レビューは静的テストの一種であるので、テストマネージャは全体的な成功、特にテストウェアプロダクトに関する成功に対して責任を持つ。ただし、ソフトウェアプロジェクトのさまざまな状況において、この責任は、組織的なポリシーと関係する。テストマネージャ、品質保証マネージャ、または訓練を受けたレビューコーディネータは、公式レビューを多くの領域にわたって広範に適用する可能性があるので、ソフトウェアプロジェクトの開始前および実行中の両方において責任を持つ。本シラバスでは、責任を持つ人は、それが誰であっても、レビューリーダと呼ぶ。

レビューリーダは、ISTQB Foundation Level シラバスで定義している成功要因の実現を推進する環境を確保する必要がある。さらに、レビューリーダは、レビューが効果的な価値をもたらすように、測定計画を作成する。

テスト担当者は運用での動作とソフトウェアシステムに求められる特徴を深く理解しているので、レビュープロセスに関与させることが重要である。

レビュー参加者は、レビューのトレーニングを受け、レビュープロセスにおけるそれぞれの役割についてよく理解していなければならない。すべてのレビュー参加者は、円滑なレビューとなるように寄与しなければならない。

レビューを正しく実施すると、全体の品質に最大限寄与する、もっともコスト効率の高い手段になる。このため、レビューリーダがプロジェクトの効果的なレビューを実施し、これらのレビューのメリットを示すことは、非常に重要である。

プロジェクト内のレビューとしては、次のようなものがある。

- 契約レビュー。プロジェクトの開始時および主要なプロジェクトマイルストンで開始する。
- 要件レビュー。要件がレビューのために準備できた際に開始する。理想的には機能要件と非機能要件を網羅する。
- 基本設計レビュー。全体的なアーキテクチャ設計がレビューのために準備できた際に開始する。
- 詳細設計レビュー。詳細設計がレビューのために準備できた際に開始する。
- コードレビュー。ソフトウェアの各モジュールが作成された際に実行する。ユニットテスト、その結果、およびコード自身を含むことがある。
- テスト成果物レビュー。テスト計画、テスト条件、品質リスク分析結果、テスト、テストデータ、テスト環境、およびテスト結果を含むことがある。
- 各テストレベルのテスト開始（テスト準備）レビューとテスト終了レビュー。前者は、テスト実行を開始する前にテスト開始基準をチェックし、後者は、テストを終了する前にテスト終了基準をチェックする。
- 受け入れレビュー。システムに対する顧客またはステークホルダの承認を得るために使用する。

レビューリーダは、1 つのプロダクトに複数の種類のレビューを適用できる。さらに、レビューは静的ドキュメントの欠陥を見つける手段として使用できるだけでなく、静的テストの他の方法（静的解析など）やコードの動的テストを補完するものになることを、認識する必要がある。これらの技法を組み合わせて使用することにより、テストカバレッジが向上し、より多くの欠陥を検出できるようになる。

技法ごとに重点は異なる。たとえば、レビューを行うことにより、コードに問題を実装する前に、要件レベルで問題を除去できる。静的解析を行うことにより、コーディング標準に準拠させることができ、成果物の精査では工

数がかかりすぎて見つけることのできない問題をチェックできる。インスペクションは、欠陥の検出と除去に役立つだけでなく、成果物に欠陥を作りこまないように開発者をトレーニングできる。

ISTQB Foundation Level シラバスでは、次のレビュータイプを規定している。
- 非公式レビュー
- ウォークスルー
- テクニカルレビュー
- インスペクション

これらに加えて、テストマネージャは、次のレビューにも関与することがある。
- マネジメントレビュー
- 監査

3.2 マネジメントレビューと監査

マネジメントレビューは、進捗状況のモニタリング、ステータスの評価、将来の対応の決定を行うために実施する。これらのレビューにより、適用するリソースの度合い、是正措置の実行、またはプロジェクトの対象範囲の変更など、プロジェクトの将来に関する決定が行えるようになる。

マネジメントレビューの主な特徴には、次のようなものがある。
- プロジェクトまたはシステムに直接責任を持つマネージャまたはその代理によって実施する。
- ステークホルダや意思決定者、またはその代理によって実施する。たとえば、上位マネージャまたは部門長など。
- 計画との整合性および逸脱をチェックする。
- マネジメント手順の妥当性をチェックする。
- プロジェクトリスクを評価する。
- アクションの影響と、これらの影響を測定する方法を評価する。
- アクションアイテム、解決すべき課題および行うべき意思決定のリストを作成する。

プロジェクトの振り返り（学習した教訓）などプロセスのマネジメントレビューは、プロセス改善活動の重要な構成要素である。

テストマネージャは、テスト進捗のマネジメントレビューに参加すべきであり、レビューを推進することもある。

監査は、通常、一定の基準（多くの場合、適用される標準、規制、または契約義務）に対する準拠を示すために実行する。このため、監査は、プロセス、規制、標準などへの準拠について独立した評価を提供することを目的としている。監査の主な特徴には、次のようなものがある。
- 監査リーダが管理、モデレートする。
- 準拠のエビデンスを、ヒアリング、観察、ドキュメントの検査などを通じて収集する。
- ドキュメントとしての成果には、観察事項、勧告、是正措置、合否アセスメントを含む。

3.3 レビューのマネジメント

レビューは、ソフトウェアプロジェクトの適切な区切り、またはマイルストンで実施する。通常、レビューは、要件と設計を定義した後に実施する。関連するレビューは、ビジネスの目的に対するところから始まり、詳細設計段階に対してまで行う。マネジメントレビューは、主要なプロジェクトマイルストンで、通常、テスト実行およびその他の重大なプロジェクトフェーズの前、途中、後の検証活動の一環として行う。レビュー戦略は、テストポリシーと全体的なテスト戦略とに適合させなければならない。

プロジェクトレベルでの全体的なレビュー計画を作成する前に、レビューリーダ（テストマネージャが兼ねることもある）は、次の点を考慮する。

- レビュー対象は何か（プロダクトおよびプロセス）
- 特定のレビューに誰が関与するか
- カバーすべき関連リスク要因はどれか

プロジェクト計画フェーズの初期において、レビューリーダはレビュー対象となるアイテムを特定し、適切なレビュータイプ（非公式レビュー、ウォークスルー、テクニカルレビュー、インスペクション、または 2 つか 3 つのタイプの組み合わせ）と公式度合いを決定する。この時点で、追加のレビューのトレーニングを推奨することがある。ここから、レビュープロセスの予算（時間とリソース）を割り当てることができる。予算を決定する際には、リスク評価と投資効果の計算を行う。

レビューの投資効果は、レビューを行うためにかかるコストと、レビューを行わずに以降の段階で検出した場合（または、まったく検出できなかった場合）に同じ欠陥に対処するためにかかるコストとの差である。2.7 節で説明する品質コストの計算が、この数値を決定するために使用できる。

レビューを実行するための最適なタイミングは、次の要素で決定される。

- レビュー対象のアイテムの、体裁面での完成度合い。
- レビューを行うのに適している担当者の参加可能性。
- アイテムの最終版が利用できるタイミング。
- その特定のアイテムのレビュープロセスにかかる時間。

レビュー評価を行うための適切なメトリクスを、テスト計画時にレビューリーダによって定義する。インスペクションを使用する場合には、ドキュメントの細分化（たとえば、個々の要件単位、または節単位など）が完了したときに、開発者の要求に応じて簡易的なインスペクションを実施するべきである。

レビュープロセスの目的を、テスト計画時に定義する。これには、効果的で効率的なレビューの実施、およびレビューのフィードバックに関する合意した意思決定を行うことを含む。

プロジェクトレビューは、システム全体に対して繰り返し行い、サブシステムおよび個々のソフトウェア要素に対して、必要に応じて行う。レビューの回数、レビューのタイプ、レビューの編成、および関与する人々といったものはすべて、プロジェクトの規模や複雑さ、およびプロダクトリスクに依存する。

レビューの参加者は、効率を高めるために、技術と手順の両方に関して、適切な知識を有している必要がある。レビューを効果的に行うために、他のスキルとしてレビューアに求められるものとしては、細部に対する完全さと注意力がある。良いレビューコメントには、明確かつ優先度付けが正しいという特徴がある。手順に関する知識が必要であり、レビューアが自分の役割と責任を確実に理解するために、レビューアに対して適切なトレーニングが必要になることがある。

レビュー計画では、技術的要因、組織的要因、およびレビュー実行時の人的問題に関するリスクに対処する。レビューが成功するためには、十分な技術的知識を持ち合わせているレビューアの参加がきわめて重要である。プロジェクトのすべてのチームをレビュー計画に参加させ、各チームにレビュープロセスの成功の責任を持たせる。計画では、各組織において、必要とされるレビューアがプロジェクトスケジュールの適切な時点でレビューの準備を行い、それらに参加するための時間を、十分に割り当てていることを保証しなければならない。必要とされるテクニカルトレーニングまたはプロセストレーニングをレビューアが受講する時間も計画する。人的計画またはビジネス計画の変更により主要なレビューアが参加できなくなった場合に備えて、バックアップのレビューアを決めておく。

公式レビューを実際に行う場合、レビューリーダは、次の点について確認する。
- 適切な測定指標をレビュー参加者が提供することにより、レビューの評価を効率よく行う。
- 将来のレビューに備えて、チェックリストを作成し、維持する。
- レビュー時に発見した問題の欠陥マネジメントを行うために、欠陥の重要度と優先度の付け方を定義する（第4章を参照）。

各レビューの実施後、レビューリーダは、次のことを行う必要がある。
- レビューメトリクスを収集し、識別された課題が、レビューの特定のテスト目的に十分に沿って解決されていることを保証する。
- レビューの投資効果（ROI）を決定する際の入力として、レビューメトリクスを使用する。
- フィードバック情報を関係するステークホルダに提供する。
- フィードバックをレビュー参加者に提供する。

レビューの効果を評価するのであれば、テストマネージャは、レビューレポートの結果と以降（レビュー後）のテストで発見した実際の結果を比較しても良い。成果物をレビューし、その結果に基づいて承認したが、以降で欠陥を発見した場合、レビューリーダは、欠陥を見逃したレビュープロセスの方法を再検討する必要がある。欠陥を見逃した原因としては、レビュープロセスの問題（たとえば、不適切な開始／終了基準など）、レビューチームの不適切な編成、不適切なレビューツール（チェックリストなど）、レビューアの不十分なトレーニングと経験、過少な準備期間およびレビューミーティングの時間などが、可能性として挙げられる。

欠陥（特に重大な欠陥）の見逃しが複数のプロジェクトで繰り返される場合、レビューの実施方法に重大な問題がある。このような状況では、レビューリーダはプロセスを十分確認し、適切な対応をとる必要がある。同様に、さまざまな理由により、時間の経過とともにレビューが効果を失う可能性がある。このような影響により、レビューでの欠陥の検出効率が落ちることは、プロジェクトの振り返りで明らかになる。さらに、レビューリーダは原因を調査して解決する必要がある。いずれの場合でも、レビューメトリクスを個々のレビューアまたは開発者に対する賞罰を行うために使用してはならない。レビューメトリクスは、レビュープロセス自体に焦点を置くべきである。

3.4 レビューのためのメトリクス

レビューリーダ（前の節で説明しているようにテストマネージャが兼務することもある）は、次のアクションを行うためにメトリクスを使用できるようにする。
- レビュー対象アイテムの品質を評価する。
- レビューを実施するためのコストを評価する。
- レビューを実施したことによる下流段階への利点を評価する。

レビューリーダは測定指標を使用して、投資効果とレビューの効率を確認できる。これらのメトリクスは、レポート活動やプロセス改善活動にも使用できる。

レビュー対象の各成果物に対して、プロダクト評価に向けて次のメトリクスを測定してレポートする。
- 成果物のサイズ（ページ数、コードの行数など）
- 準備時間（レビュー前に費やした時間）
- レビューを実施するための時間
- 欠陥を解決するための再作業時間
- レビュープロセスの期間
- 発見した欠陥の数とそれらの重要度
- 成果物内の欠陥の偏在（欠陥をより高い密度で発見する領域）の識別
- レビューのタイプ（非公式レビュー、ウォークスルー、テクニカルレビュー、またはインスペクション）
- 平均欠陥密度（たとえば、ページ当たり、またはコード1,000行当たりの欠陥数など）

- 推定残存欠陥数（または残存欠陥密度）

各レビューに対して、次のメトリクスをプロセス評価用に測定およびレポートできる。
- 欠陥検出効率（以降のライフサイクルで発見する欠陥を考慮）
- レビュープロセスの活動とタイミングの改善
- 計画した成果物を網羅している割合
- 発見した欠陥の種類とそれらの重要度
- レビュープロセスの効果と効率性に関する参加者調査
- レビューで発見した欠陥と、動的テストおよび運用時に発見した欠陥の比に関する品質メトリクスのコスト
- レビュー効率の相関関係（レビューのタイプと欠陥検出効率）
- レビューア数
- 費やした作業時間当たりの欠陥検出数
- プロジェクトで節約される推定時間
- 平均欠陥工数（総検出時間と総修正時間を足した時間を総欠陥数で割った値）

さらに、これまでに示したプロダクト評価用のメトリクスは、プロセス評価にも役立つ。

3.5 公式レビューのマネジメント

ISTQB Foundation Level シラバスでは、公式レビューの 6 つのフェーズ、つまり、計画、キックオフ、個々の準備、レビューミーティング、再作業、フォローアップについて記述している。公式レビューを的確に行うため、レビューリーダは、レビュープロセスの以下のすべてのステップを確実にする必要がある。

公式レビューには、次のような特性がある。
- 定義済みの開始基準と終了基準
- レビューアが使用すべきチェックリスト
- レポート、評価シート、または他のレビューサマリシートなどの提供物
- レビューの効果、効率、および進捗をレポートするためのメトリクス

公式レビューを開始する前に、レビューリーダは、手順または開始基準で定義しているレビューの前提条件を満たしていることを確認する。

公式レビューの前提条件を満たしていない場合、レビューリーダは、レビュー責任者に次のいずれかを提案して、最終判断を求めることができる。
- 目的が変更された場合のレビューの再定義
- レビューを進めるために必要な是正措置
- レビューの延期

公式レビューをコントロールする手順の一環として、これらのレビューは、全体的な（上位レベルの）プログラムの状況でモニタリングし、プロジェクト品質保証活動に関連付けられる。公式レビューのコントロールには、プロダクトメトリクスとプロセスメトリクスを使用したフィードバック情報を含む。

4. 欠陥マネジメント – 150 分

用語

不正、欠陥、欠陥選別委員会、故障、偽陰性結果、偽陽性結果、フェーズ内阻止、優先度、根本原因、重要度

「欠陥マネジメント」の学習の目的

4.2 欠陥ライフサイクルとソフトウェア開発ライフサイクル

TM-4.2.1　（K3）テストライフサイクルを通して、プロジェクトの欠陥をモニタリングしコントロールするために使用する欠陥レポートワークフローを含む、テスト組織のための欠陥マネジメントプロセスを開発する。

TM-4.2.2　（K2）効果的な欠陥マネジメントのために必要なプロセスと参加者について説明する。

4.3 欠陥レポート情報

TM-4.3.1　（K3）欠陥マネジメントプロセス中に収集すべきデータおよびクラシフィケーション情報を定義する。

4.4 欠陥レポート情報によるプロセス能力の評価

TM-4.4.1　（K2）テストプロセスとソフトウェア開発プロセスのプロセス能力を評価するために、欠陥レポートの統計情報をどのように使用するかを説明する。

4.1 イントロダクション

組織の欠陥マネジメントプロセスとこの作業のために使用するツールは、テストチームだけでなく、ソフトウェア開発に関与するすべてのチームにとって非常に重要である。欠陥の効率的なマネジメントによって収集する情報は、テストマネージャおよび他のプロジェクトのステークホルダが開発ライフサイクル全体でプロジェクトの状態を把握するのに役立つ。長期間にわたってデータを収集および分析することによって、テストプロセスおよび開発プロセスの改善の見込みのある領域を見つけるのに役立つ。

テストマネージャは、全体的な欠陥ライフサイクルと、これを使用してテストプロセスとソフトウェア開発プロセスの両方をモニタリングおよびコントロールする方法を理解するだけでなく、取得すべき重要なデータについて精通し、プロセスと選択した欠陥マネジメントツールの両方の適切な使用を提唱する必要がある。

4.2 欠陥ライフサイクルとソフトウェア開発ライフサイクル

Foundation Level シラバスで説明しているように、欠陥は、人が成果物を作成するときに誤りを犯すと混入する。欠陥が混入した成果物となりうるのは、要求仕様、ユーザストーリー、テクニカルドキュメント、テストケース、プログラムコード、またはソフトウェア開発プロセスやメンテナンスプロセスで作成した他の成果物である。

欠陥は、ソフトウェア開発ライフサイクルやソフトウェア関連の成果物のいずれの場所にも混入する可能性がある。このため、ソフトウェア開発ライフサイクルの各フェーズでは、潜在的な欠陥を検出および除去するための活動を含める必要がある。たとえば、静的テスト技法（レビューと静的解析）を要求仕様、設計仕様、コードに対して使用した後に、それらの成果物を以降の活動に提供する。各欠陥の検出および除去を早く行うほど、システムの全体的な品質コストは低くなる。各欠陥の除去を、混入したのと同じフェーズ内で行う（ソフトウェアプロセスで完全なフェーズ内阻止を達成する）と、そのフェーズの欠陥に対する品質コストが最小化される。さらに、Foundation Level シラバスで説明しているように、静的テストは、故障ではなく欠陥を直接発見する。このように、欠陥を特定するのにデバッグ活動を必要としないので、欠陥を除去するコストはより低くなる。

ユニットテスト、統合テスト、システムテストなどの動的テスト活動の実行時、欠陥の存在は故障の発生により明らかになる。これにより、実際の結果とテストの期待結果に相違が生じる（不正）。テスト担当者が不正を観察しない場合に、偽陰性結果が発生することがある。テスト担当者が不正を観察すると、さらなる調査が必要になる。この調査は、欠陥レポートを登録することから始まる。

テスト駆動開発では、自動化されたユニットテストを、実行可能な設計仕様という形で使用する。コードを開発すると、ただちにこれらを使用してテストを行う。ユニットの開発が完了するまで、一部またはすべてのテストは失敗する。このため、このようなテストで検出した故障は欠陥と見なさず、通常は追跡しない。

4.2.1 欠陥ワークフローと状態

ほとんどのテスト組織は、欠陥ライフサイクルを通して、欠陥レポートをマネジメントするためにツールを使用する。欠陥レポートは、通常、ワークフローを通して進み、欠陥ライフサイクルの進行に従って一連の状態を遷移する。これらの状態のほとんどにおいて、一人の欠陥ライフサイクル参加者がレポートの所有者となり、完了した場合に欠陥レポートを次の状態に遷移させる（および次の責任グループに割り当てる）タスクを実行する責任を持つ。欠陥レポートをクローズする（通常、基になる欠陥が解決し、確認テストにより解決を検証したことを意味する）、キャンセルする（通常、欠陥レポートが無効であることを意味する）、再現できない（通常、不正が再び観察されないことを意味する）、または延期する（通常、不正は欠陥に関連するが、その欠陥はプロジェクト内で修正しないことを意味する）などの終了状態では、そのレポートは、さらなる活動を必要としないので、所有者がいなくなる。

テスト時にテスト担当者が発見した欠陥に対して、テストチームが行うべき活動には、3 つの状態がある。

- 初期状態
  - この状態では、欠陥を解決する責任を持つ人が不正を再現できるように、一人または複数のテスト担当者が必要な情報を収集する（欠陥レポートに含む情報の詳細については、4.3 節を参照）。
  - この状態は、「オープン」状態または「新規」状態とも呼ぶ。
- 返却状態
  - この状態では、レポートの受取人がレポートを拒否したか、またはテスト担当者にさらなる情報を依頼している。この状態は、初期の情報収集プロセスまたはテスト自体が不十分であったことを意味する。テストマネージャは返却の割合が過剰でないことをモニタリングする必要がある。テスト担当者は追加情報を提供するか、レポートが実際に拒否されるべきものであることを確認する。
  - これは、「拒否」状態または「明確化」状態とも呼ぶ。
- 確認テスト状態
  - この状態では、テスト担当者は、確認テストを実行して（通常、欠陥レポート自体に記載されている故障を再現する手順に従って行う）、解決策により問題が実際に解決したかどうかを確認する。確認テストで欠陥が修正されたことが示唆された場合、テスト担当者はレポートをクローズする。確認テストで欠陥が修正されていないことが示唆された場合、テスト担当者はレポートを再度オープンし、以前の所有者にレポートを再び割り当てる。以前の所有者は、欠陥を修復するために必要な作業を完了する。
  - この状態は、「解決済み」状態または「検証」状態とも呼ぶ。

4.2.2 無効および重複した欠陥レポートのマネジメント

特定の状況では、不正が、欠陥の兆候としてではなく、テスト環境、テストデータ、テストウェアの他の要素、またはテスト担当者自身の誤解により発生することがある。テスト担当者が欠陥レポートをオープンし、その後、そのレポートはテストでの成果物の欠陥と関連しないことが判明した場合、偽陽性結果をもたらす。このようなレポートは、無効な欠陥レポートとして、キャンセルするか、クローズする。さらに、1 つの欠陥が異なる兆候を示し、テスト担当者にとってまったく関連していないように見える場合がある。2 つまたはそれ以上の欠陥レポートが登録され、その後、同じ根本原因に関連していることが判明した場合、通常、それらの欠陥レポートの 1 つのみを維持し、他の欠陥レポートは重複欠陥レポートとしてクローズする。

無効および重複した欠陥レポートは、非効率ではあるが、このようなレポートがある程度発生することは予測できることであり、そのようにテストマネージャによって受け入れられる。マネージャが無効および重複した欠陥レポートのすべてを排除しようとすると、通常、テスト担当者が欠陥レポートの登録をためらうので、偽陰性の数が増加する。これにより、ほとんどの場合はテスト組織の主要な目標に関連する欠陥検出効率が低下する。

4.2.3 クロスファンクショナルな欠陥マネジメント

テスト組織とテストマネージャは、通常、全体的な欠陥マネジメントプロセスと欠陥マネジメントツールを所有するが、クロスファンクショナルなチームは、一般的に、特定のプロジェクトでレポートされた欠陥のマネジメントを担当する。テストマネージャに加えて、欠陥マネジメント（または欠陥選別）委員会の参加者は、通常、開発中のソフトウェアに利害関係を持つ開発者、プロジェクトマネージャ、プロダクトマネージャ、およびその他のステークホルダを含む。

不正を発見し、欠陥マネジメントツールに登録すると、欠陥マネジメント委員会はミーティングを開き、各欠陥レポートが有効な欠陥を示しているかどうか、および解決するべきか延期するべきかどうかを決定する。この決定を行うために、欠陥マネジメント委員会は、欠陥を解決する場合または解決しない場合に生じるメリット、リスク、

およびコストを検討する。欠陥を解決することに決定した場合、欠陥マネジメント委員会は、欠陥を解決する活動の優先度を、他のプロジェクトタスクに相対して設定する。テストマネージャとテストチームは、欠陥の相対的な重要性について質問された場合に、利用可能な客観的情報を提供する。

優れたコミュニケーションを欠陥追跡ツールで置き換えることはできない。また、優れた欠陥追跡ツールを、効果的に使用している場合、欠陥マネジメント委員会のミーティングで置き換えることはできない。コミュニケーション、適切なツールサポート、適切に定義された欠陥ライフサイクル、および欠陥マネジメント委員会の関与のすべてが、効率的で効果的な欠陥マネジメントに必要である。

4.3 欠陥レポート情報

静的テストの一環として欠陥を検出するか、動的テストの一環として故障を観察すると、関与した担当者がデータを収集し、欠陥レポートに含める。この情報は、次の 3 つの目的を満たす必要がある。

- 欠陥ライフサイクル全体でのレポートのマネジメント
- プロジェクトステータスのアセスメント、特に、プロダクトの品質とテスト進捗に関するもの
- プロセス能力のアセスメント（以降の 4.4 節を参照）

欠陥レポートマネジメントとプロジェクトステータスで必要なデータは、欠陥をライフサイクルのどの時点で検出したかによって異なり、通常、早い時点ほど（たとえば、要件レビューとユニットテストなど）必要となる情報は少ない。ただし、収集される核となる情報は、ライフサイクル全体で、理想的にはすべてのプロジェクトで一貫しており、プロジェクト全体およびすべてのプロジェクトにわたって、プロセスに紐付く欠陥データの意味ある比較ができるようにしなければならない。

収集された欠陥データは、テスト進捗のモニタリング、コントロール、および終了基準の評価に使用できる。たとえば、欠陥情報は、欠陥密度分析、検出し解決した欠陥の傾向分析、欠陥の検出から解決までの平均時間、および故障強度（たとえば、MTBF 分析など）に使用できる。

収集される欠陥データは、次のアイテムを含む。

- 欠陥を発見した人の名前
- その人の役割（たとえば、エンドユーザ、ビジネスアナリスト、開発者、テクニカルサポート担当者など）
- 実行していたテストタイプ（たとえば、使用性テスト、性能テスト、回帰テストなど）
- 問題の概要
- 問題の詳細説明
- 欠陥に関する故障を再現する手順、実際の結果と期待結果（不正を強調する）、可能な場合はスクリーンショット、データベースダンプ、およびログ
- 欠陥の混入、検出、および除去が発生したライフサイクルフェーズ、可能な場合はテストレベル
- 欠陥が混入した成果物
- システムおよびプロダクトのステークホルダへのインパクトの重要度（通常、システムの技術的振る舞いにより決定する）
- 問題を解決する優先度（通常、故障のビジネスインパクトによって決定する）
- 欠陥が存在するサブシステムまたはコンポーネント（欠陥の偏在の分析に用いる）
- 問題を検出したときに実行していたプロジェクト活動
- 問題を明らかにした識別方法（たとえば、レビュー、静的解析、動的テスト、実運用など）
- 欠陥の種類（通常、使用する欠陥分類法に対応する）
- 欠陥により影響を受ける品質特性
- 欠陥を観察したテスト環境（動的テスト向け）
- 問題が存在するプロジェクトとプロダクト

- 現在の所有者、つまり、レポートが最終状態にない場合に、問題に関する作業が現在割り当てられている人
- レポートの現在の状態（通常、欠陥追跡ツールによりライフサイクルの一部としてマネジメントする）
- 問題を観察した特定の成果物（たとえば、テストアイテムとそのリリース番号など）、問題を最終的に解決した特定の成果物
- プロジェクトステークホルダおよびプロダクトステークホルダの利害へのインパクト
- 問題を解決するためのアクションの実行または不実行に関する結論、提案、承認
- 欠陥を解決する、または解決しない場合のリスク、コスト、機会、メリット
- 欠陥ライフサイクルのさまざまな遷移が発生した日付、各遷移でのレポートの所有者、欠陥の特定、修正および、解決を検証するためにプロジェクトチームメンバが行ったアクション
- 欠陥を最終的に解決した方法の説明と、解決策をテストするための推奨策（欠陥をソフトウェアの変更により解決した場合）
- 欠陥を明らかにしたテスト、欠陥に関連するリスク、要件、他のテストベース要素など、その他の情報（動的テストの場合）

ISO 9126 [ISO9126]（ISO 25000 に改訂中）、IEEE 829 [IEEE829]、IEEE 1044 [IEEE1044]、直交欠陥分類法など、テストマネージャが欠陥レポートのために収集する必要のある情報を決定するのに役立つ、さまざまな標準やドキュメントが利用できる。

欠陥レポートに必要であるとして決定した情報が何であろうと、テスト担当者が完全、簡潔、正確、客観的、適切、タイムリーのすべての特性を備えた情報を入力することがきわめて重要である。個々の欠陥を解決するという点において、欠陥レポートのデータに関する問題を、手動による操作および対面的なコミュニケーションによって克服できる場合でも、プロジェクトステータス、テスト進捗、プロセス能力を適切にアセスメントするという点では、欠陥レポートのデータに関する問題は克服できない障壁となりうる。

4.4 欠陥レポート情報によるプロセス能力の評価

第 2 章で説明しているように、欠陥レポートを、プロジェクトステータスのモニタリングとレポートに役立てることができる。メトリクスが示すプロセスの状態については、主に、Expert Test Management シラバス[ISTQB ETM SYL]で説明するが、Advanced Level では、テストマネージャは、テストプロセスとソフトウェア開発プロセスの能力を評価するという点で、欠陥レポートが何を意味するかを認識する必要がある。

第 2 章と 4.3 節で説明しているテスト進捗のモニタリング情報に加えて、欠陥情報は、プロセス改善の取り組みを支援する必要がある。次に例を示す。

- 混入、検出、除去の情報に含まれるフェーズ情報をフェーズごとに使用して、フェーズ内阻止を評価し、各フェーズでの欠陥検出効率を改善する方法を提案する。
- 混入フェーズの情報を、最多数の欠陥が混入したフェーズのパレート分析用に使用して、欠陥の総数を削減するための的を絞った改善を行う。
- 欠陥の根本原因情報を使用して欠陥混入の基になる理由を確認し、欠陥の総数を削減するためのプロセス改善を行う。
- 混入、検出、除去の情報に含まれるフェーズ情報を使用して品質コスト分析を実行し、欠陥により発生するコストを最小化する。
- 欠陥コンポーネントの情報を使用して、欠陥の偏在の分析を実行し、テクニカルリスク（リスクベースドテストの場合）の理解を深め、問題の多いコンポーネントのリエンジニアリングを可能にする。

テストプロセスの効果と効率を評価するためにメトリクスを使用する方法については、Expert Test Management シラバス[ISTQB ETM SYL]で説明する。

チームが、ソフトウェア開発ライフサイクルの一部またはすべてで見つけた欠陥を追跡しないことがある。この行為は、効率性の名目やプロセスオーバーヘッドを削減する目的で頻繁に行うが、実際には、テストとソフトウェア開発のプロセス能力に対する視認性を大幅に低下させる。これは、信頼できるデータの欠如により、上記で提案した改善の実行を困難にする。

5. テストプロセスの改善 – 135 分

用語

能力成熟度モデル統合（CMMI）、クリティカルテストプロセス（CTP）、体系的テストと評価プロセス（STEP）、テスト成熟度モデル統合（TMMi）、TPI Next

「テストプロセス改善」の学習の目的

5.2 テスト改善プロセス

TM-5.2.1　（K2）テストプロセスを改善することが、なぜ重要か、例を使用して説明する。

5.3 テストプロセスの改善

TM-5.3.1　（K3）IDEAL モデルを使用して、テストプロセスの改善計画を定義する。

5.4 TMMi によるテストプロセスの改善

TM-5.4.1　（K2）TMMi テストプロセス改善モデルの背景、対象範囲、目的をまとめる。

5.5 TPI Next によるテストプロセスの改善

TM-5.5.1　（K2）TPI Next テストプロセス改善モデルの背景、対象範囲、目的をまとめる。

5.6 CTP によるテストプロセスの改善

TM-5.6.1　（K2）CTP テストプロセス改善モデルの背景、対象範囲、目的をまとめる。

5.7 STEP によるテストプロセスの改善

TM-5.7.1　（K2）STEP テストプロセス改善モデルの背景、対象範囲、目的をまとめる。

5.1 イントロダクション

テストプロセスは、確立した後も、継続的に改善する必要がある。本章では、まず一般的な改善の問題について説明し、それに続いて、テストプロセス改善に使用できる具体的なモデルを紹介する。テストマネージャが、テストプロセスの変更と改善の推進役として機能することを想定する。テストマネージャは、本章で説明する業界で受け入れられている技法に精通する必要がある。テスト改善プロセスについての詳細な情報は、Expert Level Improving the Test Process シラバスで説明する。

5.2 テスト改善プロセス

テストがソフトウェアの改善に使用されるのと同じように、ソフトウェア開発プロセス（およびソフトウェア成果物）を改善するために、プロセス改善技法を選択し、使用する。プロセス改善は、テストプロセスにも当てはまる。ソフトウェアおよびソフトウェアを含むシステムのテストを改善する上で、さまざまな手段と方法を利用できる。これらの方法では、改善のためのガイドラインおよび領域を提供することにより、プロセス（および成果物）の改善を目指している。

テストは多くの場合、プロジェクトコスト全体において大きな部分を占める。ただし、CMMI®のようなさまざまなソフトウェアプロセス改善モデルでは、テストプロセスに対してわずかしか注意を払っていない（詳細については以降を参照）。

このような領域をカバーするため、テスト成熟度モデル統合（TMMi®）、体系的テストと評価プロセス（STEP）、クリティカルテストプロセス（CTP）、TPI Next®などのテスト改善モデルが開発されている。これらのモデルでは、一定の組織横断メトリクスを用意しており、「ベンチマーク」の比較に使用できるようになっている。

本シラバスで紹介するモデルは、使用を推奨するものではなく、モデルがどのように機能し、その中に何を含むかを示すために代表例として紹介している。

5.2.1 プロセス改善の紹介

プロセス改善は、テストプロセスの場合と同様、ソフトウェア開発プロセスに関連している。誤りから学習することで、組織がソフトウェアの開発とテストで使用しているプロセスを改善できる。数十年もの間、デミング改善サイクル（Plan、Do、Check、Act）を使用しており、テスト担当者が現在使用中のプロセスを改善する必要があるときは、現在でもこれを使用できる。

プロセス改善のための前提として、システムの品質が、ソフトウェアの開発に使用しているプロセスの品質によって大きく影響を受けるという考え方がある。ソフトウェア産業では、一般的に品質を改善すると、ソフトウェアを維持するためのリソースを削減できるため、将来優れたソリューションを作成するために時間を割くことができるようになる。プロセスモデルでは組織のプロセスの成熟度を測定することにより、改善を開始する。このモデルでは、アセスメントの結果に基づいて組織のプロセスを改善するためのフレームワークを提供する。

プロセスアセスメントに続いて、プロセス能力判定を行う。これがプロセス改善の基盤になる。この後、改善の効果を測定するため、プロセスアセスメントを行うことになる。

5.2.2 プロセス改善のタイプ

アセスメントモデルの使用は一般的に行われており、実証済みの実践例を使用してテストプロセスを確実に改善する標準的なアプローチとなっている。

プロセス改善モデルには、プロセス参照モデルとコンテンツ参照モデルの 2 つのタイプがある。

1. プロセス参照モデルは、アセスメントの一部として成熟度の測定指標を提供しており、モデルと比較して組織の能力を評価したり、フレームワーク内で組織を評価したりするために利用する。また、プロセスを改善するためのロードマップも提供する。
2. コンテンツ参照モデルは、プロセスを改善するための、組織的な機会におけるビジネス指向の評価を提供し、たとえば、目的思考の測定指標を使用して業界平均に対するベンチマークを行うために利用する。この評価は、プロセスを改善するためのロードマップを作成するために使用できる。

テストプロセスの改善は、たとえば、分析的アプローチおよび振り返りミーティングを行うことにより、モデルを使用しなくても達成できる。

5.3 テストプロセスの改善

IT 業界では、高いレベルの成熟度および専門性に達することを追求するため、テストプロセス改善モデルを使い始めた。業界標準モデルは、比較に使用できる組織横断メトリクスおよび評価基準を開発する上で役に立つ。テスト業界では、プロセス改善の必要性からいくつかの推奨プロセスをとりまとめた。これらの標準には、STEP、TMMi、TPI Next、CTP が含まれる。TMMi および CMMI などの段階的なモデルでは、異なる企業および組織を比較するための標準を提供する。CTP、STEP、TPI Next などの連続的なモデルでは、実行順序の自由度が高く、組織はもっとも優先度の高い問題に対応できる。それぞれについて本節で詳細に説明する。

これらのすべてのモデルでは、現在のテストプロセスの観点において組織がどのような立ち位置にあるかを判定できる。アセスメントを行ったら、TMMi および TPI Next により、テストプロセスを改善するためのロードマップを提示する。一方、STEP および CTP を使用した場合、プロセス改善による最大の投資効果をどの部分で得ているかを判定することができ、組織は適切なロードマップを選択できるようになる。

テストプロセスを見直し、改善するという点で合意できた場合、この活動では IDEAL[SM] モデル[IDEAL96]で次のように定義されているプロセス改善を推進するステップが採用できる。

- 改善プロセスの開始
- 現在の状況の診断
- テストプロセス改善計画の確立
- 改善を推進する活動
- 改善プログラムからの学習

改善プロセスの開始
プロセス改善活動を開始する前に、ステークホルダによって、プロセス改善の目的、目標、対象範囲や適用範囲について合意がなされる。プロセス改善モデルの選択も、この時点で行う。モデルは、公開されているモデル（CTP、STEP、TMMi、TPI Next など）から選択するか、内部で開発する。また、合格基準を定義しておき、この基準に基づき改善活動の際に測定する方法を決定しなければならない。

現在の状況の診断
合意したアセスメントのアプローチを用いて、テストアセスメントレポートを作成する。このレポートには、現在認められているテストにおける慣例と、見込めるプロセス改善項目のリストを含む。

テストプロセス改善計画の確立
プロセス改善項目リストは、優先度の順に並べる。この順序は、投資効果、リスク、組織戦略との整合性、測定可能な定量的または定性的な利点などに基づいて設定できる。優先順位を策定し、改善を実行するための計画を作成する。

改善を推進する活動
定義が終わったら、プロセス改善を展開する。このとき、必要なトレーニングやメンタリング、プロセスのガイドなどを導入することができ、最終的にはこれらをすべて展開する。

改善プログラムからの学習
プロセス改善をすべて展開したら、改善の前に合意した利点や予測していなかった利点などについて確認する。プロセス改善活動の合格基準に合致していることも重要になる。

適用するプロセスモデルによっては、この段階で次の成熟度レベルのモニタリングが始まり、改善プロセスを再開するかこの時点で活動を停止するかのいずれかを決定する。

5.4 TMMi によるテストプロセスの改善

テスト成熟度モデル統合（TMMi）は、CMMI を補間するもので、5 つの成熟度レベルで構成している。それぞれの成熟度レベルには、定義済みのプロセス領域があり、組織が次のレベルに進むためには、これらの特定または全般的な目標の 85%を満たしていなければならない。

TMMi の成熟度レベルは次のようになっている。

- レベル 1: 初期
  初期レベルは、公式な文書または体系的なテストプロセスがない状態を表す。テストケースは一般的に、コーディングの後、アドホックに開発され、テストはデバッグと同じ行為と見なされている。テストは、ソフトウェアの動作を証明するものと理解されている。
- レベル 2: 管理された
  レベル 2 は、テストプロセスはデバッグから明確に分離されている。テストポリシーおよび目標の設定、基本的なテストプロセス（たとえばテスト計画など）の導入、そして、基本的なテスト技法および手法の活用を行うことで達成される。
- レベル 3: 定義された
  レベル 3 は、テストプロセスをソフトウェア開発ライフサイクルに統合し、標準ドキュメント、進め方、手法を公式なものとして文書化したときに達成できる。この場合、レビューを実行し、ソフトウェアテストは明確にコントロールとモニタリングが可能な活動として機能しなければならない。
- レベル 4: 測定された
  レベル 4 は、定義したテストプロセスを組織レベルで効率的に測定およびマネジメントでき、特定のプロジェクトで効率的に適合できた場合に達成される。
- レベル 5: 最適化された
  テストプロセスの成熟度の最終レベルを表す。つまり、テストプロセスのアウトプットを使い欠陥を予防するのを手助けし、確立したプロセスをより最適化していくことに焦点を当てた状態である。

TMMi の詳細については、[vanVeenendaal11]と[www.tmmi.org]を参照されたい。

5.5 TPI Next によるテストプロセスの改善

TPI Next モデルは 16 のキーエリアを定義し、各キーエリアは、テスト戦略、メトリクス、テストツール、テスト環境などテストプロセスの特定の側面をカバーする。

このモデルでは、次に示す 4 つの成熟度レベルを定義している。

- 初期
- 制御された

- 効率的な
- 最適化している

各成熟度レベルの各キーエリアを評価するために、特定のチェックポイントを定義している。すべてのキーエリアをカバーする成熟度マトリクスを使用して、評価の結果を、要約および視覚化する。改善目的の定義とそれらの導入は、テスト組織のニーズと能力に応じて調整できる。

TPI Next は、一般的なアプローチを採用しており、すべてのソフトウェアプロセス改善モデルから独立し、テストエンジニアリングの側面と、管理者の意思決定支援の両方をカバーしている[deVries09]。

TPI Next の詳細については、[www.tpinext.com]を参照されたい。

5.6 CTP によるテストプロセスの改善

クリティカルテストプロセス（CTP）アセスメントモデルでは、モデルの中で特定しているテストプロセスが最重要（クリティカル）であることが基本的な前提となる。最重要なプロセスを正しく実行すれば、テストチームには成功がもたらされる。逆にこのような活動が貧弱であれば、能力の高いテスト担当者やテストマネージャであっても成功するのは困難になる。このモデルでは、12 のクリティカルテストプロセスを識別している。CTP は、どちらかと言えばコンテンツ参照モデルである。

CTP モデルは、状況に応じたアプローチであり、次のようなモデルのカスタマイズが可能になる。

- 特定の課題の識別
- 優れたプロセスの属性に対する認識
- プロセス改善を進める順序と重要度の選択

CTP モデルは、すべてのソフトウェア開発ライフサイクルモデルの状況に合わせて適合できる。

参加者のインタビューに加えて CTP モデルには、業界平均およびベストプラクティスに対する、ベンチマークとなる組織へのメトリクスの適用が含まれる。

CTP の詳細については、[Black03]を参照されたい。

5.7 STEP によるテストプロセスの改善

STEP（体系的テストと評価プロセス）は、TMMi や TPI Next と違って、CTP と同様に改善を特定の順序で行う必要がない。

STEP はどちらかと言えばコンテンツ参照モデルで、要件策定からシステムの廃止にわたりテストの活動が継続するライフサイクル活動であるという考え方に基づいている。STEP 方法論では、要件ベースのテスト戦略を適用することで「テストしてからコーディング」という方法を重視する。つまり、早期のテストケース作成により、設計とコーディングの前に要求仕様の妥当性を確認するようにする。

この方法論の基本的な前提には、次のようなものがある。

- 要件ベースのテスト戦略である。
- テストをライフサイクルの開始時に始める。
- テストは要件および利用方法のモデルとして使う。
- テストウェアの設計がソフトウェアの設計をリードする。
- 欠陥を早期に検出するか、またはまとめて予防する。

- 欠陥を体系的に分析する。
- テスト担当者および開発者が共同で作業する。

状況によっては、STEP アセスメントモデルは TPI Next 成熟度モデルと組み合わせて使用することもできる。

STEP の詳細については、[Craig02]を参照されたい。

6. テストツールおよび自動化 – 135 分

用語

オープンソースツール、カスタムツール

「テストツールおよび自動化」の学習の目的

6.2 ツール選択

TM-6.2.1　（K2）オープンソースツールを選択する場合のマネジメント上の問題点について説明する。
TM-6.2.2　（K2）カスタムツールを決定する場合のマネジメント上の問題点について説明する。
TM-6.2.3　（K4）ツール選択計画を策定するために、リスク、コスト、利点などを含む前提の状況を評価する。

6.3 ツールのライフサイクル

TM-6.3.1　（K2）ツールのライフサイクル内の各フェーズについて説明する。

6.4 ツールのメトリクス

TM-6.4.1　（K2）ツールを活用することでメトリックの収集および評価がどのように改善できるかを説明する。

6.1 イントロダクション

本節では、テストマネージャがツールおよび自動化に関して考慮すべき一般的な概念を紹介する。ここでの解説は、Foundation Level シラバスでの説明に加え、知っておかなければならないことである。

6.2 ツール選択

テストツールを選択する場合には、テストマネージャはさまざまな問題を考慮する必要がある。

歴史的に、もっとも一般的な選択は、商用ベンダーからツールを購入することである。場合によっては、これが唯一の選択肢となることもある。ただし、オープンソースツールやカスタムツールを、利用可能なオプションとして選択できることもある。

ツールの種類に関係なく、テストマネージャはコストメリットを分析して、ツールの想定されるライフサイクル全体での総所有コストを入念に調査する必要がある。このトピックは、以降の「投資効果（ROI）」節でカバーする。

6.2.1 オープンソースツール

オープンソースツールは、テストケースマネジメントから欠陥追跡、さらにはテストケースの自動生成に至る、テストプロセスのほとんどすべての局面で利用できるものがある。オープンソースツールで注意すべき重要な点は、一般的に、ツール自体の初期購入コストは高くないが、ツールに対する正式なサポートがほとんどないことである。ただし、多くのオープンソースツールでは、一般的でなかったり、非公式であったりするサポートをユーザに提供する熱心なサポータが存在する。

また、多くのオープンソースツールは元々、特定の問題を解決したり、単一の問題に対処したりするために作成されており、類似のベンダーツールの機能のすべては実行できないことがある。このため、オープンソースツールを選択する前に、テストグループの実際のニーズを入念に分析する必要がある。

オープンソースツールを使用する利点の 1 つは、ユーザが変更または拡張できることである。テスト組織に専門的な技術がある場合、他のツールと連携するように変更したり、テストチームのニーズに適合するように拡張したりできる。複数のツールを組み合わせて、ベンダーツールでは対処できない問題を解決できることもある。当然のことながら、使用するツールと適用する変更の数が増えるほど、複雑性とオーバーヘッドが増加する。テストマネージャは、他のツールと同様に、チームがオープンソースツールを使用すること自体を目的としないようにし、常に ROI がプラスとなるように努める必要がある。

テストマネージャは、選択したツールのライセンススキームを理解する必要がある。多くのオープンソースツールには、GNU GPL(GNU 一般公衆利用許諾契約書)が付属する。GNU GPL は、ソフトウェアの配布を、ソフトウェアを受け取ったのと同じ条件下で常に行わなければならないことを指定する。つまり、テストを行いやすくするためにテストチームがツールを変更した場合、それらの変更は、ツールライセンスの下で、ツールを使用するすべてのユーザが利用できるようにする必要がある。テストマネージャは、組織にソフトウェアを再配布することに伴う法的な問題点を確認する必要がある。

セーフティクリティカルなソフトウェアまたはミッションクリティカルなソフトウェアを開発する組織、または規制準拠の対象となる組織は、オープンソースツールを使用することにより問題が生じることがある。多くのオープンソースツールは非常に高い品質を保持するが、ほとんどのオープンソースツールで正確性は保証しない。ベンダーツールは、多くの場合、正確性と、特定のタスク（たとえば DO-178B など）への適合性を保証している。オー

プンソースツールは優れたものである可能性があるが、正確性を保証するのは、使用するグループの責任であり、余分なオーバーヘッドを引き起こす。

6.2.2 カスタムツール

テスト組織は、ベンダーツールやオープンソースツールでは対応できない特定のニーズを持つことがある。その理由としては、独自のハードウェアプラットフォーム、通常とは異なる環境、通常行うプロセスとは異なるプロセスなどが挙げられる。このような状況で、チームに専門的な技術がある場合、テストマネージャが、カスタムツールの開発を検討することがある。

カスタムツールを開発する利点は、ツールがチームのニーズと正確に適合し、チームが要求する状況で効率的に動作できることである。ツールには、使用中の他のツールとインターフェースをとるための機能や、チームで必要とされるのと同じ形式のデータを生成するための機能を実装できる。また、ツールを直近のプロジェクトだけでなく、組織の複数のアプリケーションで利用することがある。ただし、ツールを他のプロジェクトにリリースすることを計画する前に、まず、目的、目標、利点、予測されるマイナス面をレビューする必要がある。

カスタムツールの作成を検討しているテストマネージャは、予想されるマイナス面も考慮する必要がある。カスタムツールは、ほとんどの場合、ツールを作成した人に依存することになる。そのため、他者がメンテナンスできるように、適切に文書化しておかなければならない。これを怠ると、ツール作成者がプロジェクトから離れた場合に、ツールが放置され、使用されなくなる。時間経過と共に、カスタムツールは当初の利用目的を外れた新しい要件を加えられたり、拡張がなされたりする。これにより、ツールに品質上の問題が発生し、偽陽性の欠陥レポート、または不正確なデータの作成をもたらす可能性がある。テストマネージャは、カスタムツールがソフトウェアプロダクトの 1 つであること、このため、他のソフトウェアプロダクトと同じような開発上の問題が発生する可能性があることを認識する必要がある。カスタムツールは、期待どおりに動作するように、設計およびテストする必要がある。

6.2.3 投資効果（ROI）

テストマネージャは、テスト組織に導入するすべてのツールがチームの作業に価値を付加し、組織にプラスの ROI を示すようにする責任を持つ。ツールが実用的で、継続して利点をもたらすようにするために、ツールの入手または構築を行う前に、コストメリットを分析する。この分析では、ROI を固定費と初期コストの両方で検討する。これらの一部は金銭上のものであり、一部はリソースまたは時間のコスト、ツールの価値が減少するリスクによるコストである。

初期コストには、次のようなものがある。

- 目的と目標を満たすためのツール要件の定義
- 適切なツールとツールベンダーの評価および選択
- ツールの購入、適用、または開発
- ツールの初期トレーニングの実行
- 他のツールとの統合
- ツールをサポートするために必要なハードウェア／ソフトウェアの購入

固定費には、次のようなものがある。

- ツール所有コスト
  - ライセンス料金およびサポート料金
  - ツール自体のメンテナンスコスト
  - ツールにより作成された成果物のメンテナンス
  - 継続的なトレーニングおよびメンタリングのコスト
- 別の環境へのツールの移植

- 将来のニーズへのツールの適用
- 選択したツールを最適に使用できるようにするための品質とプロセスの改善

テストマネージャは、すべてのツールに伴う機会コストについても検討する必要がある。ツールの入手、管理、トレーニング、使用に費やす時間は、実際のテストタスクでも費やした可能性があるので、ツールを実運用で使用するようになるまでに、より多くのテストリソースが必要になる可能性がある。

ツールの使用には、多くのリスクが伴う。すべてのツールがリスクを上回る利点を提供するわけではない。ツールのリスクについては、Foundation Level シラバスで説明する。テストマネージャは、ROI を検討する場合に、以下のリスクも考慮する必要がある。

- 組織の未成熟性（ツールを使用する準備が整っていない）
- テスト対象のソフトウェアの変更により、ツールによる成果物に複数の版（リビジョン）が必要となるため、ツールによる成果物のメンテナンスが困難となる可能性
- テストアナリストによるテストタスクへの関与の減少がもたらす、テスト価値の低下（たとえば、自動化したスクリプトを実行しているときのみ、欠陥検出の効率が低下してしまうなど）

最終的に、テストマネージャは、ツールの使用により得られる利点を検討する必要がある。ツールの導入および使用がもたらす利点には、次のようなものがある。

- 反復作業の減少
- テストサイクル時間の削減（たとえば自動回帰テストの使用による削減など）
- テスト実行コストの削減
- 特定のテストタイプのテスト実行数の増加（たとえば回帰テストなど）
- テストのさまざまなフェーズでの人的エラーの減少。たとえば、以下のようなもの
  - データ生成ツールの使用による、テストデータの有効性向上
  - 比較ツールの使用による、テスト結果比較時の精度向上
  - スクリプトツールの使用による、テストデータ入力の正確性向上
- テストに関する情報にアクセスするために必要な工数の削減。たとえば、以下のようなもの
  - ツールにより生成されたレポートおよびメトリクス
  - テストケース、テストスクリプト、テストデータなどのテスト資産の再利用
- ツール使用により可能になるテストタイプの増加（たとえば性能テスト、ロードテストなど）
- 自動化を実現したテスト担当者の地位と、洗練されたツールを理解した上で使いこなすことを示すことでのテスト組織全体の地位の向上

一般的に、テストグループが単一のツールのみを使用することはほとんどない。テストチームが取得する総 ROI は、通常、使用する全ツールの相関関係により計算される。ツールは情報を共有し、連携して動作させる必要がある。長期にわたる包括的なテストツール戦略をとることを推奨する。

6.2.4 選択プロセス

テストツールは、単一プロジェクトが何度も反復する中で拡張されたり、多くのプロジェクトで適用されたりするため、長期にわたる投資となる場合がある。テストマネージャは、候補となるツールを、さまざまな観点から検討する必要がある。

- ビジネスにとっては、プラスの ROI が求められる。投資から高い価値を取得するために、ツール（テストツールと非テストツールの両方を含む）を連携して相互運用する。場合によっては、ツールを使用するためのプロセスおよび接続性を改善して、この相互運用性を達成する。ただし、これを達成するには、時間がかかることがある。

- プロジェクトにとっては、効率的なツールが求められる（たとえば、データ入力時のタイプエラーのような手動テスト実行時のエラーの防止など）。ツールの投資効果が現れるまでには、長い時間がかかることがある。多くの場合、投資効果は、自動化を実装した最初のプロジェクトではなく、2 回目のリリース時またはメンテナンス時に現れることがある。テストマネージャは、アプリケーションの全ライフサイクルを考慮する必要がある。
- ツールの利用者にとっては、プロジェクトメンバがタスクをより効率的および効果的に行えるようになることが求められる。学習曲線を考慮して、ユーザがすばやく、最小限のストレスでツールを学習できるようにする必要がある。ツールを初めて導入したときには、ユーザのトレーニングとメンタリングが必要である。

すべての観点を確実に考慮するために、テストツールを導入するためのロードマップを作成することが重要である。

テストツールの選択プロセスは、Foundation Level シラバスで次のように説明している。

- 組織の成熟度を評価する。
- ツールの要件を識別する。
- ツールを評価する。
- ベンダーまたはサービスサポート（オープンソースツール、カスタムツール）を評価する。
- ツールを使用するためのトレーニングおよびメンタリングの内部要件を識別する。
- 現在のテストチームのテスト自動化スキルに基づいてトレーニングニーズを評価する。
- コストメリットを見積る（「6.2.3 投資効果（ROI）」を参照）。

各種類のツールに対して、ツールを使用するテストフェーズに関係なく、テストマネージャは次の能力を検討する必要がある。

- 分析
  - このツールは、与えられる入力を「理解」できるか？
  - ツールは目的に合致しているか？
- 設計
  - このツールは、既存の情報に基づいてテストウェアを設計することを支援するか（たとえば、要件からテストケースを生成するテスト設計ツールなど）？
  - 自動的に設計が行えるか？
  - 実際のテストウェアコードは、メンテナンスおよび使用が可能な形式で、全部あるいは一部が生成されるか？
  - 必要なテストデータは、自動的に生成されるか（たとえば、コードを解析しデータを生成など）？
- データとテストの選択
  - ツールは必要なデータをどのように選択するか（たとえば、どのテストケースがどのデータセットを使用して実行するかなど）？
  - ツールは、手動または自動で入力した選択条件を受け付けるか？
  - ツールは、選択した入力に基づいて運用データを「選別」する方法を決定できるか？
  - ツールはカバレッジ基準に基づいて必要とするテストを決定できるか（たとえば、指定した要件セットに基づいて、ツールはトレーサビリティを検証し、どのテストケースを実行する必要があるかを決定できるかなど）？
- 実行
  - ツールは自動実行できるか？または手動による操作が必要か？
  - ツールはどのように停止し再起動するか？

- ツールは、関連するイベントを「把握」できるか（たとえば、テストケースに対してレポートされた欠陥をクローズした場合に、テストマネジメントツールは自動的にテストケースのステータスを更新できるかなど）？

- 評価
  - ツールは、適切な結果を受け取っているかどうかをどのように確認できるか（たとえば、ツールは、テストオラクルを使用して応答の正しさを確認できるかなど）？
  - ツールはどのような種類のエラー回復能力を提供するか？
  - ツールには適切なログ記録およびレポートの機能が用意されているか？

6.3 ツールのライフサイクル

ツールを有益に利用するためのライフサイクルには、テストマネージャがマネジメントする必要のある 4 つの段階がある。

1. 調達。ツールはここまで説明しているように、調達する必要がある（「6.2 ツール選択」を参照）。ツールを作成することを決定したら、テストマネージャはツールの管理者となる人を（テストアナリストまたはテクニカルテストアナリストから）割り当てる。管理者は、ツールを使用する方法と状況、作成した成果物を格納する場所、命名規則などを決定する。これらの決定を、場当たり的に行うのではなく、ツールの使用を開始する前に行うと、ツールの ROI を大幅に改善できる。また、ツールのユーザに適切なトレーニングを行う必要がある。
2. サポートとメンテナンス。ツールの継続したサポートとメンテナンスが必要である。ツールを維持するための責任は、ツールの管理者に持たせるか、専門のツールグループに割り当てる。ツールを他のツールと連携させて使用する場合、連携して動作するためのデータ変換とプロセスを検討する。また、ツールのバックアップとリストア、およびその出力を検討する。
3. 進化。改造を検討しなければならない。時間経過と共に、環境、ビジネスニーズ、ベンダーの問題によって、ツールおよびその使用方法が大きく変化することがある。たとえば、ツールベンダーがツールの更新を要求し、それによって、連携しているツールとの間で問題が発生することがある。ビジネス上の理由により環境が変化し、ツールに問題が発生することがある。ツールの運用環境が複雑であればあるほど、進化的変更がツールの使用におよぼす影響は大きい。この時点で、テストマネージャは、ツールがテストで果たす役割に応じて、サービスの継続性を確立する必要がある。
4. 廃棄。どのようなツールでも、利用を終了するときがくる。この時点で、ツールの使用をやめる必要がある。ツールによって提供されていた機能を何かに置き換え、データは保持およびアーカイブする必要がある。これが発生するのは、ツールがライフサイクルの最終段階にあるか、単純に、新しいツールへの移行による利点と機会がコストとリスクを上回るようになった場合である。

テストマネージャは、ツールのライフサイクル全体を通して、ツールによりテストチームに提供されるサービスが正常に機能し、故障なく連続して提供されるようにする必要がある。

6.4 ツールのメトリクス

テストマネージャは、テクニカルテストアナリストおよびテストアナリストによって使用されるツールに関して、目的のメトリクスを設計し収集できる。さまざまなツールで、価値あるリアルタイムデータを取得でき、データ収集の工数を削減できる。テストマネージャはこれらのデータを使用して、全体的なテスト活動をマネジメントできる。

ツールにより、重点的に収集するデータの種類は異なり、次のようなものがある。

- テストマネジメントツールは、さまざまな種類のメトリクスを提供できる。要件からテストケースおよび自動化スクリプトに至るトレーサビリティにより、カバレッジメトリクスを取得できる。現在利用可能なテスト、計画しているテスト、現在の実行ステータス（合格、不合格、スキップ、ブロック、待ち行列内）のスナップショットをいつでも取得できる。

- 欠陥マネジメントツールは、システム全体での欠陥に関する現在のステータス、重要度、優先度、分布など多くの情報を提供できる。欠陥が混入および発見したフェーズや、見逃し率などの他の重要なデータも提供する。これはテストマネージャがプロセスを改善するのに役立つ。
- 静的解析ツールは、保守性の問題を検出および報告するのに役立つ。
- 性能テストツールは、システムの拡張性に関する有益な情報を提供できる。
- カバレッジツールは、テストマネージャがテスト全体でテスト消化率を理解するのに役立つ。

ツールのレポート要件をツール選択プロセスで定義する。これらの要件をツール構成時に適切な方法で実装し、ツールにより追跡する情報をステークホルダに理解可能な方法でレポートする必要がある。

7. スタッフのスキル – チーム構成 – 210 分

用語

テストの独立性

「スタッフのスキル - チーム構成」の学習の目的

7.2 個人のスキル

TM-7.2.1　(K4)スキルアセスメントスプレッドシートを使用して、ソフトウェアシステムの使用、ドメインおよびビジネスに関する知識、システム開発領域、ソフトウェアテスト、対人関係スキルに関する、チームメンバの強みと弱みを分析する。

TM-7.2.2　(K4)チームのスキルに関するアセスメント結果を分析し、トレーニングとスキル開発計画を策定する。

7.3 テストチームの力学

TM-7.3.1　(K2)特定の状況でテストチームのリーダとなるために必要なハード面とソフト面のスキルについて説明する。

7.4 組織内におけるテストの適合

TM-7.4.1　(K2)独立テストのオプションを説明する。

7.5 モチベーション

TM-7.5.1　(K2)テスト担当者のモチベーションを上げる要因、および下げる要因について、例を挙げて説明する。

7.6 コミュニケーション

TM-7.6.1　(K2)テストチーム内、およびテストチームとステークホルダとの間のコミュニケーションを、効率的に行うための要因を説明する。

7.1 イントロダクション

有能なテストマネージャは、スキルが適切に混在するように、チームメンバの求人、採用、維持を行う。スキル要件は、時間経過と共に変化する可能性があるので、最初に適切な人を採用することに加えて、テストチームを保持し、最高の仕事の水準を維持するために、適切なトレーニングと成長機会を提供することが重要である。チームのスキルだけでなく、テストマネージャは、プレッシャーが厳しかったり、進みが速かったりする環境で効果的に機能するように自身のスキルセットを維持する必要がある。

本章では、スキルを評価する方法、内部的にまとまりがあり、組織内で効率的な相乗効果の高いチームを作成するためにギャップを埋める方法、チームのモチベーションを高める方法、効果的なコミュニケーションを行う方法について説明する。

7.2 個人のスキル

ソフトウェアテストのための個人の能力は、さまざまな作業領域における経験、教育、トレーニングを通じて獲得できる。以下のそれぞれが、テスト担当者の知識を身につける上で役に立つ。

- ソフトウェアシステムの利用
- ドメインまたはビジネスについての知識
- 分析、開発、テクニカルサポートを含む、ソフトウェア開発プロセスのさまざまなフェーズでの活動
- ソフトウェアテストの活動

ソフトウェアシステムのエンドユーザは、どのようにシステムを操作するか、どの部分の故障が重大な影響をおよぼすか、さまざまな状況でシステムにどのような反応を期待するかなどについて多くの知識がある。また、ドメインの知識が豊富なユーザであれば、ビジネスでもっとも重要な領域や、これらの領域が、要求に対応するためにビジネス面での能力にどのように影響するかを知っている。この知識は、テスト活動に優先順位を付けるときや、現実的なテストデータおよびテストケースを作成するとき、ユースケースを確認または提供するときに活用できる。

ソフトウェア開発プロセス（要求分析、アーキテクチャ、設計、コーディング）についての知識があれば、エラーがどのようにして欠陥の混入に結びつくかや、どこで欠陥を見つけるか、欠陥の混入をどのように防げば良いかなどについて推察することができる。また、テクニカルサポートの経験があれば、ユーザエクスペリエンス、期待するもの、使用性要件についての知識が身につく。ソフトウェア開発の経験は、プログラミングや設計に関する専門的知識が必要なテストツールの利用や、静的コード解析、コードレビュー、ユニットテスト、技術に重点を置く統合テストに参加する上で重要になる。

特定のソフトウェアテストスキルには、Foundation、Advanced Test Analyst、Advanced Technical Test Analyst の各シラバスで説明している仕様分析の能力、リスク分析に関与する能力、テストケース設計の能力、入念にテストを実行し結果を記録する能力を含む。

テストマネジメントには、たとえば計画の作成や進捗状況の追跡、ステークホルダへのレポートなどといった多くのプロジェクトマネジメント活動を含んでいるため、テストマネージャの場合は特にプロジェクトマネジメントの知識、スキル、経験を持つことが重要である。プロジェクトマネージャが不在の場合、特にプロジェクトの後半の段階では、テストマネージャがプロジェクトマネージャの役割を兼務することがある。これらのスキルは、Foundation シラバスと本シラバスで説明している能力を補完する。

テクニカルスキルに加えて、互いに建設的に批評を行う力、影響力、交渉力などの対人関係スキルは、テストの役割において非常に重要である。技術的に優れたテスト担当者でも、必要なソフト（対人関係）スキルを持っ

ていない限り、失敗しがちである。テストの専門家として成功するには、他者と効率的に仕事ができるだけでなく、組織をうまくまとめ、細かいことに配慮し、書面および口頭での優れたコミュニケーションスキルを持っていなければならない。

理想的なテストチームは、スキルと経験度合いが混在している。そして、各チームメンバが自発的に互いに教え、学び合う能力を持つべきである。環境によって、他のスキルより重要とされたり、尊重されたりするスキルが存在する。たとえば、API のテストとプログラミングのスキルが必要なテクニカルテスト環境では、テクニカルスキルがドメイン知識よりも重要と見なされる。ブラックボックステスト環境では、ドメインの専門知識がもっとも重要である。環境とプロジェクトは変化するということを、覚えておくことが重要である。

スキルアセスメントスプレッドシートを作成する場合、テストマネージャは仕事で重要なスキルだけでなく、職位に適切なスキルもすべてリストする必要がある。それらのスキルの一覧を作成したら、得点システム（たとえば、5 段階評価で 5 がその領域で期待するもっとも高い評価など）を使用してチームの各個人を評価できる。評価により、各個人の強みと弱みを確認し、その情報に基づいて個人またはグループのトレーニング計画を作成する。テストマネージャは、特定の領域のスキルを改善するために各人の能力目標を設定し、個人のスキルを評価するために使用する特定の基準を定義する。

単一のプロジェクトのためだけでなく、長期間を想定した人の雇用を行うべきである。テストマネージャがテスト担当者に投資し、継続して学習する環境を構築すると、チームメンバはモチベーションが高まって自分のスキルと知識を向上することに熱心になり、次の機会の準備を整えることができる。

チームマネージャが最適なチームメンバを採用できることはほとんどない。また、チームメンバが現在のプロジェクトに対して最適であっても、次のプロジェクトに対してはスキル構成が最適ではない可能性がある。テストマネージャは、知的で、好奇心が強く、適応性があり、仕事熱心で、チームのメンバとして効率的に作業し、学習意欲と学習能力が高い人を採用しなければならない。最適な個人の集合は構築できないが、個人の強みと弱みについてうまくバランスをとることによって、強いチームを構築できる。

テストマネージャは、スキルアセスメントスプレッドシートを使用して、チームの強みと弱みを識別できる。この情報は、トレーニングおよびスキル開発計画の基礎情報として活用できる。テストマネージャは、これらの計画をチームの効率と効果に最大の影響をおよぼす弱みから始めて、それらの領域に対処する方法を決定する必要がある。1 つの方法はトレーニングである。たとえば、人にトレーニングコースを受講させる、社内でトレーニングセッションを開催する、独自のトレーニングを開発する、または e ラーニングコースを利用するなどである。自己学習も 1 つの方法である。たとえば、本、オンラインセミナー、インターネットリソースを利用するなどである。さらには、クロストレーニングの方法もある。たとえば、あるスキルを学習する意欲のある人を、すでにそのスキルを持ち、それを必要とするタスクを実行している人と組ませる、または身近にいる専門家にその専門領域について簡単なプレゼンテーションを行ってもらう（メンタリングも同様の方法である。初めて役割を担う人をその役割の経験がある上級者と組み合わせて、上級者は、日々アドバイスや支援を提供する役割を務める）などである。弱みに対処するだけでなく、テストマネージャは、トレーニングとスキル開発計画の一環としてスキルアセスメントで識別した強みを強化する必要がある。テストチームの開発計画の詳細については、[McKay07]を参照されたい。

7.3 テストチームの力学

可能な限り最善のチームを構築するために、スタッフの選択は組織の管理者にとってもっとも重要な仕事の 1 つである。仕事に必要な個人のスキルだけでなく、さまざまな事項について考慮しなければならない。チームに参加する個人を選択するとき、チームの力学についても配慮しなければならない。たとえば、迎えようとしているテスト担当者がテストチームのスキルを補完できるか？また性格が合うかどうか？といったことである。テストチームに、技術的なスキルや、さまざまな性格のメンバが混在していることの利点について考えることも重要にな

る。強力なテストチームは、さまざまな複雑性を持つ複数のプロジェクトに対応できる上、他のプロジェクトチームのメンバとの個人同士の相互作用にもうまく対処できるからである。

テストは、多くの場合、プレッシャーの厳しい活動である。ソフトウェア開発スケジュールがひっ迫していたり、非現実的なことさえある。ステークホルダは、テストチームに高い期待を不合理に抱いている。テストマネージャは、プレッシャーの高い状況にうまく対処できる人を採用しなければならない。このような人は、フラストレーションを発散することができ、スケジュールが不可能に思える場合でも作業に集中できる。スケジュールや期待度の問題に対処するのはテストマネージャの責任であるが、テストマネージャは、これらのプレッシャーをチームメンバも同じように感じていることを理解しなければならない。テストマネージャは、チームメンバを採用する場合、作業環境と、性格が環境に適していることを考慮しなければならない。与えられた時間で完了できる量よりも多くの作業がある環境では、テストマネージャは、タスクを終了できる人を常に探す努力をし、次に何ができるかをメンバに確認しなければならない。

個人は、自分の価値と必要性が認識されていると感じると、より熱心に働き、周囲により気を配るようになる。個人は、それぞれがチームの重要なメンバで、自身の貢献がチーム全体の成功に必須であることを理解しなければならない。このような力学が醸成されると、クロストレーニングを非公式に行い、仕事量の平準化をチームメンバ自身で行い、テストマネージャはより多くの時間を対外的な問題の対処に振り向けることができるようになる。

新しいチームメンバはすぐにチームに組み入れて、適切に監督しサポートしなければならない。それぞれの個人には、そのチーム内における明確な役割を与えなければならない。これは、個人に対するアセスメントプロセスに基づいて行うことができる。最終目標は、それぞれのメンバが個人として成功し、同時にチーム全体の成功にも貢献できることである。これは、各個人に対して性格に応じたチーム内の役割を割り当て、その人の生来のスキルを活かすようにし、あわせてスキルの向上を図ることによって実現できる。

テストチームに採用または追加する人を決定する場合、スキルの目的指向のアセスメントが役立つ。これは、インタビュー、候補者のテスト、作業サンプルのレビュー、経歴の確認を行うことによって達成できる。評価するスキルには、次のようなものがある。

テクニカルスキル（ハードスキル）は、次の項目で示される。

- 要件ドキュメントからテストケースを抽出する。
- テクニカルドキュメント（要件、コードなど）をレビューする。
- レビューコメントを明確で分かりやすく、目的に沿った形式で記述する。
- 特定のシナリオでさまざまなテスト技法を適用する（たとえば、デシジョンテーブルを使用して一連のビジネスルールをテストするなど）
- 故障を評価し、正確に記録する。
- 欠陥分類情報の理解度をはっきりと示す。
- 欠陥の根本原因の理解度をはっきりと示す。
- ツールを使用して、正常系テストと異常系テストを含む、特定の API をテストする。
- SQL を使用してデータベース情報を検索および変更し、特定のシナリオをテストする。
- 複数のテストを実行して、テスト結果を収集するテスト自動化ハーネスを設計する。
- 自動化したテストを実行し、故障のトラブルシューティングをする。
- テスト計画、テスト仕様を記述する。
- テスト結果のアセスメントを含むテストサマリレポートを記述する。

対人関係スキル（ソフトスキル）は、次の項目で示される。

- スケジュールが遅れているテストプロジェクトに関する情報を提示する。
- 欠陥がないと思っている開発者に欠陥レポートを説明する。
- 同僚をトレーニングする。

- 効果的でないプロセスに関して、問題をマネジメント層に提示する。
- 同僚が作成したテストケースをレビューし、その同僚にコメントを提示する。
- 同僚をインタビューする。
- 開発者を称賛する。

これは完全なリストではなく、要望される特定のスキルは環境や組織によって異なるが、一般的に必要とするスキルをリストしている。効果的なインタビューの質問を作成し、スキルを示すチャンスを与えることで、テストマネージャは候補者のスキルを評価し、強みと弱みを判断できる。アセスメントに使用するスキルセットをうまくまとめたら、既存のすべてのスタッフに適用して、それら人の育成とトレーニングの領域を決定しなければならない。

テストマネージャには、個々の貢献者に必要とされるスキルに加えて、優れたコミュニケーションと渉外のスキルも必要である。テストマネージャは、論争の起こりそうな状況を和らげ、必要なコミュニケーションのために使用する正しい手段を把握した上で、組織内に適切な人間関係を構築し維持しなければならない。

7.4 組織内におけるテストの適合

組織と言っても多様であるため、テストを組織構造に適合する方法も幅広い。品質はソフトウェア開発ライフサイクル全体を通して全員の責任であるが、独立したテストチームは、質の高いプロダクトの作りこみに大きく貢献できる。テストの独立性は、次のリスト（独立性がもっとも低いものから順に並んでいる）で示すように、実際には大きく変動する。

- 独立したテスト担当者がいない。
  - この場合、独立性はなく、開発者が自身のコードをテストしている。
  - 開発者は、テストの時間があれば、コードが意図したとおりに動作するかどうかを判定する。そのため、実際の要件に合うこともあれば合わないこともある。
  - 開発者は、見つけた欠陥を迅速に修正できる。
- テストを、コードを書いた開発者と異なる開発者が実施する。
  - 開発者とテスト担当者との間にほとんど独立性がない。
  - 他の開発者のコードをテストする開発者は、欠陥のレポートを出すのに消極的になることがある。
  - テストに向かう開発者の気持ちは、通常、正常系のテストケースに集中する。
- テストを、開発チームに属するテスト担当者（またはテストチーム）が実施する。
  - テスト担当者（またはテストチーム）は、プロジェクトマネジメント担当または開発マネージャにレポートを出す。
  - テスト担当者の気持ちは、どちらかと言えば、要件への準拠の確認に集中する。
  - テスト担当者が開発チームのメンバであるため、テストだけでなく、開発の責任も負うことがある。
  - テスト担当者のマネージャが、品質目標を達成することよりも、スケジュールを維持することに重点を置くことがある。
  - チーム内で、テスト活動が開発活動より低いステータスを割り当てられることがある。
  - テスト担当者が、品質目標または目標の維持に影響をおよぼす権限を持たないことがある。
- ビジネス組織、ユーザコミュニティ、その他の開発に関わっていない技術組織からのテスト担当者がテストを実施する。
  - テスト結果情報を、ステークホルダに対して客観的に報告する。
  - 品質がこのチームの第一の関心事になる。
  - テストでは、スキルの開発とトレーニングに焦点を当てる。
  - テストはキャリアパスとして見なす。
  - 品質を第一の関心事にしているテストグループに専任のマネジメント担当がいる。
- 外部のテスト専門家が、それぞれのテストタイプについてテストを実施する。

  - 一般化したテストでは不十分である特定の領域に、専門知識を適用する。
  - テストタイプは、使用性、セキュリティ、性能、特殊性を求める他の領域などである。
  - 個人においては品質に焦点を当てるべきであるが、視点は特殊性の領域に限定する。性能が優れているプロダクトが機能要件を満たさない場合に、性能の専門家によって見逃されることがある。
- テストを企業の外部組織が実施する。
  - このモデルは、テスト担当者と開発者との間で、最大の独立性を実現する。
  - テスト担当者がテストを的確に行うための知識の移転が、不十分になる可能性がある。
  - 明確な要件およびしっかり定義されたコミュニケーション構造が必要になる。
  - 外部組織の品質を定期的に監査しなければならない。

このリストは、個人の一般的な関心事に基づいており、特定の組織には適さないことがある。職務や職位が個人の関心事を決定するとは限らない。開発マネージャは品質に重点を置くので、優れたテストマネージャにもなることができる。独立したテストマネージャは、スケジュールに重点を置いてレポートする傾向があり、品質よりもスケジュールに重点を置いてテストを実行する可能性がある。組織内のテストチームにとって重点を置く最善の配置を決定するために、テストマネージャは組織の目標を理解する必要がある。

開発組織とテスト組織との間の独立性の程度はさまざまである。ただし、独立性が高くなるほど、その組織は孤立し、移転する知識も少なくなることを理解しなければならない。独立性の度合いが低いほど、知識は増えるが、本来の目的からの乖離が生じる。独立性の度合いは、使用されるソフトウェア開発モデルによっても決まる。たとえばアジャイル開発においては、テスト担当者は開発チームの一員となる。

組織では、上記のテスト形態が混在することもある。独立したテスト組織だけでなく、開発組織内でもテストを実行し、外部組織が最終認定するという方式も考えられる。スケジュールや予算の制約を守りながら、最終プロダクトの品質を最大限向上させるためには、それぞれのテストフェーズに対する責任と期待を理解し、これらの要件を設定することが重要になる。

7.5 モチベーション

テスト担当個人にモチベーションを与える方法は多岐にわたるが、たとえば、次のようなものがある。

- 達成した仕事に対する評価
- マネジメント層から認められる
- プロジェクトチーム内および同僚からの敬意
- 責任と自己裁量を増やす
- 成し遂げた仕事に対する適切な報酬（給与、責任や評価の増加を含む）

このようなモチベーションを向上させる事柄を適用しにくくするようなプロジェクトからの影響もある。たとえば、テスト担当者が、不可能な納期のプロジェクトに懸命に取り組む場合が考えられる。テスト担当者は、チームの品質向上活動のためにできる限りのことを行い、努力を惜しまず残業までするが、外部の影響のために、プロダクトを本来より早く出荷してしまうなどということがある。その結果、テスト担当者が最大限に努力したにも関わらず、品質の劣るプロダクトができてしまう。テスト担当者の貢献を理解しない上に評価もしなければ、たとえ最終プロダクトが成功しても、モチベーションは簡単に下がることになる。

テストチームは、テストの実行、リスクの軽減、正確な結果の記録に向けて優れた仕事を行っていることを証明するために、適切なメトリクスを追跡しなければならない。チームのモチベーションは、このデータを収集も公開もしなかったり、仕事の達成に対して正当な評価が得られなかったりすれば、簡単に下がることになる。

正当な評価は、尊敬と認定のような無形物だけでなく、昇進の機会、給与の上昇、キャリアの向上などにもつながる。テストグループが尊敬されない場合、このような機会が与えられない可能性もある。

正当な評価と敬意は、テスト担当者がプロジェクトの価値向上に貢献することで得られる。個別のプロジェクトの場合、プロジェクトが始まったばかりのコンセプト固めの段階からテスト担当者を関与させ、ライフサイクルを通してこのような関与を継続させることで、もっともすばやく実現できる。その間、テスト担当者は、プロジェクトの前向きな発展に貢献することで、他のプロジェクトのステークホルダから正当な評価と敬意を勝ち取るのである。このような貢献は、品質コストの削減やリスク緩和という点から確認できるようになる。

テストマネージャは、より大きな組織のテストチームの第一人者として行動するだけでなく、テストチームの個人のモチベーションを向上させるのに重要な役割を果たす。テストマネージャは個々のテスト担当者の達成事項を認識し、過ちに対しても公平で誠実な評価を与えなければならない。公平で一貫性を備えたテストマネージャは、自ら範を示すことによってチームメンバのモチベーションを向上させる。

7.6 コミュニケーション

テストチームによるコミュニケーションは、主に次に示すようなもので行われる。

- テストプロダクトの文書化：テスト戦略、テスト計画、テストケース、テストサマリレポート、欠陥レポートなど。
- レビュー済みのドキュメントのフィードバック：要件、機能仕様、ユースケース、コンポーネントテストのドキュメントなど。
- 情報の収集と拡散：開発者、他のテストチームメンバ、マネジメント層などとのやりとり。

テストチームが敬意を持って扱われるようになり、その状態を維持するためには、テスト担当者と他のステークホルダとの間のコミュニケーションが専門的、客観的、効果的とならなければならない。他者の成果物についてフィードバック、特に建設的なフィードバックを行うときは、外交性や客観性も必要になる。付け加えると、コミュニケーションをとる上で、テスト目的を達成することと、プロダクトおよびソフトウェアシステム開発に使用するプロセスの両方の品質向上に注力しなければならない。

テストマネージャは、ユーザ、プロジェクトチームのメンバ、マネジメント層、外部テストグループ、顧客など、広範囲の人々とコミュニケーションをとる。コミュニケーションは、対象となる人々にとって効果的なものでなければならない。たとえば、重役のためのブリーフィング用途を考えた場合、開発チーム向けに作成した欠陥レポートだと、発見された欠陥の数や重要度の傾向や特性といった情報が詳細すぎるために不向きである。どのようなコミュニケーションでも、特にプレゼンテーションの場合、伝えたいメッセージ、メッセージを受け取ってもらう方法、メッセージが受け入れられるための適切な土壌を作るのに必要な説明は何であるかということを、理解することが重要である。テストマネージャはプロジェクトステータス情報を提示することが多いため、情報を適切な詳細度合いでまとめ（たとえば、マネジメント層は、通常、個々の欠陥よりも欠陥の傾向を知りたがるなど）、明確に提示し、簡単に理解できるような方法（たとえば、単純な図とカラフルなグラフの使用など）で提示できなければならない。効果的にコミュニケーションすることで、聴いている人の注意を引き付けることができ、正しいメッセージを伝えることができる。テストマネージャは、各プレゼンテーションを、品質と品質プロセスを促進するための機会として捉えなければならない。

テストマネージャは、部門の外部者とコミュニケーションするだけではない（外向きコミュニケーション）。テストマネージャの重要な仕事の 1 つは、テストグループ内で効果的にコミュニケーションし（内向きコミュニケーション）、ニュース、指示、優先度の変更、通常のテストプロセスで通知されるその他の標準情報を伝えることである。テストマネージャはまた、組織のマネジメント層の上（上向きコミュニケーション）と下（下向きコミュニケーション）の両方の特定の個人とコミュニケーションすることがある。コミュニケーションの方向に関係なく、同じルールを

適用する。つまり、相手にとって適切なコミュニケーションを行い、メッセージを効果的に発信し、理解度を確認しなければならない。

テストマネージャはさまざまなコミュニケーション手段を自由に使いこなせなければならない。多くの情報が、電子メール、口頭でのやりとり、公式または非公式のミーティング、公式または非公式のレポート、欠陥マネジメントツールなどのテストマネジメントツールを介して、コミュニケーションされる。すべてのコミュニケーションを専門的で客観的に行わなければならない。緊急を要するコミュニケーションの場合でも、品質と内容の両方を担保しなければならない。書面によるコミュニケーションは、多くの場合、実際のプロジェクトの終了後も、長く存在することがある。テストマネージャにとって、品質を推進する組織を表す専門的な品質のドキュメントを作成することが重要である。

8. 参考文献

8.1 標準

- [BS7925-2] BS 7925-2, Software Component Testing Standard.
  第 2 章
- [FDA21] FDA Title 21 CFR Part 820, Food and Drug Administration, USA
  第 2 章
- [IEEE829] IEEE Standard for Software and System Test Documentation
  第 2 章と第 4 章
- [IEEE1028] IEEE Standard for Software Reviews and Audits
  第 2 章
- [IEEE1044] IEEE Standard Classification for Software Anomalies
  第 4 章
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  第 2 章と第 4 章 JSTQB 訳注）日本では JIS X 0129-1 として発行されている。
- [ISO12207] ISO 12207, Systems and Software Engineering, Software Life Cycle Processes
  第 2 章 JSTQB 訳注）日本では JIS X 0160 として発行されている。
- [ISO15504] ISO/IEC 15504, Information Technology - Process Assessment
  第 2 章
- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation（SQuaRE）
  第 2 章と第 4 章
- [RTCA DO-178B/ED-12B]:Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  第 2 章

8.2 ISTQB ドキュメント

- [ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 1.0
  JSTQB 訳注）日本では「テスト技術者資格制度 Advanced Level シラバス 日本語版 概要 Version 2012」として発行されている。
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 1.0
  JSTQB 訳注）日本では「テスト技術者資格制度 Advanced Level シラバス 日本語版 テストマネージャ Version 2012」として発行されている。
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 1.0
- [ISTQB ETM SYL] ISTQB Expert Test Management Syllabus, Version 1.0
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011

JSTQB 訳注）日本では「テスト技術者資格制度 Foundation Level シラバス 日本語版 Version 2011」として発行されている。

- [ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, Version 2.2, 2012
- [ISTQB ITP SYL] ISTQB Expert Improving the Test Process Syllabus, Version 1.0

8.3 商標

以下の登録商標とサービスマークを、本ドキュメントで使用している。

- CMMI®、IDEAL℠、ISTQB®、ITIL®、PRINCE2®、TMMi®、TPI Next®
- CMMI は、カーネギーメロン大学によって米国特許商標庁にて登録されている。
- IDEAL は、カーネギーメロン大学ソフトウェア工学研究所（SEI）のサービスマークである。
- ISTQB は、International Software Testing Qualifications Board の登録商標である。
- ITIL は、米国商務省の登録商標および共同体登録商標で、米国特許商標庁にて登録されている。
- PRINCE2 は、英国内閣府の登録商標である。
- TMMi は TMMi Foundation の登録商標である。
- TPI Next は、Sogeti Nederland B.V.の登録商標である。

8.4 書籍

[Black03]: Rex Black, "Critical Testing Processes," Addison-Wesley, 2003, ISBN 0-201-74868-1

JSTQB 訳注）日本では「ソフトウェアテスト 12 の必勝プロセス」（日経 BP 社, 2005 年)として発行されている。

[Black09]: Rex Black, "Managing the Testing Process, third edition," John Wiley & Sons, 2009, ISBN 0-471-22398-0

JSTQB 訳注）日本では「基本から学ぶテストプロセス管理」（日経 BP 社, 2004 年)として発行されている。

[Black11]: Rex Black, Erik van Veenendaal, Dorothy Graham, "Foundations of Software Testing," Thomson Learning, 2011, ISBN 978-1-84480-355-2

JSTQB 訳注）日本では「ソフトウェアテストの基礎：ISTQB シラバス準拠」（ビー・エヌ・エヌ新社, 2008 年)として発行されている。

[Craig02]: Rick Craig, Stefan Jaskiel, "Systematic Software Testing," Artech House, 2002, ISBN 1-580-53508-9

JSTQB 訳注）日本では「体系的ソフトウェアテスト入門」（日経 BP 社, 2004 年）として発行されている。

[Crispin09]: Lisa Crispin, Janet Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams", Addison-Wesley, 2009, ISBN 0-321-53446-8

JSTQB 訳注）日本では「実践アジャイルテスト テスターとアジャイルチームのための実践ガイド」（翔泳社, 2009 年）として発行されている。

[de Vries09]: Gerrit de Vries, et al., "TPI Next", UTN Publishers, 2009, ISBN 90-72194-97-7

[Goucher09]: Adam Goucher, Tim Riley, "Beautiful Testing," O'Reilly, 2009, ISBN 978-0596159818

JSTQB 訳注）日本では「ビューティフルテスティング ―ソフトウェアテストの美しい実践」（オライリージャパン, 2010 年）として発行されている。

[IDEAL96]: Bob McFeeley, "IDEAL: A User's Guide for Software Process Improvement," Software Engineering Institute（SEI）, 1996, CMU/SEI-96-HB-001

[Jones07]: Capers Jones, "Estimating Software Costs, second edition," McGraw-Hill, 2007, ISBN 978-0071483001

JSTQB 訳注）日本では「ソフトウェア見積りのすべて 第 2 版 ―現実に即した規模・品質・工数・工期の予測―」（共立出版, 2009 年）として発行されている。

[Jones11]: Capers Jones and Olivier Bonsignour, "Economics of Software Quality," Pearson, 2011, ISBN 978-0132582209

[McKay07]: Judy McKay, "Managing the Test People," Rocky Nook, 2007, ISBN 978-1933952123

[Musa04]: John Musa, "Software Reliability Engineering, second edition," Author House, 2004, ISBN 978-1418493882

[Stamatis03]: D.H. Stamatis, "Failure Mode and Effect Analysis," ASQC Quality Press, 2003, ISBN 0-873-89300

[vanVeenendaal11] Erik van Veenendaal, "The Little TMMi," UTN Publishers, 2011, ISBN 9-490-986038

[vanVeenendaal12] Erik van Veenendaal, "Practical Risk-based Testing," UTN Publishers, 2012, ISBN 978-9490986070

[Whittaker09]: James Whittaker, "Exploratory Testing," Addison-Wesley, 2009, ISBN 978-0321636416

[Wiegers03]: Karl Wiegers, "Software Requirements 2," Microsoft Press, 2003, ISBN 978-0735618794

JSTQB 訳注）日本では「ソフトウェア要求」（日経 BP 社, 2003 年）として発行されている。

8.5 その他の参照元

以下は、インターネットで参照できる情報を示している。これらの参照については、本 Advanced Level シラバス発行時にチェックしているが、リファレンスがすでに参照できなくなっていても、ISTQB®はその責を負わない。

http://www.istqb.org
http://www.sei.cmu.edu/cmmi/
http://www.tmmi.org/
http://www.tpinext.com/

9. 索引

け

こ

さ

し

す

せ

そ

た

ち

つ

て

れ

わ