テスト技術者資格制度

Advanced Level シラバス日本語版
テストマネジメント

Version3.0.J04

International Software Testing Qualifications Board

著作権

改訂履歴

ISTQB®

バージョン	日付	備考
ISEB v1.1	2001/09/04	ISEB Practitione シラバス
ISTQB 1.2E	2003/09	EOQ-SG による ISTQB Advanced Level シラバス
V2007	2007/10/12	テスト技術者 Advanced Level シラバス、2007 版
D100626	2010/06/10	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	2010/12/10	書式変更と文字校正の反映
D2011	2011/10/31	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
アルファ 2012	2012/02/09	D2011 に対して受領したすべての NBs コメントの反映
Beta2012	2012/03/26	アルファ版に対して随時実施された監修結果の反映
Beta2012	2012/04/07	GA にベータ版をリリース
Beta2012	2012/06/08	NB への文書編集済みバージョンのリリース
Beta2012	2012/06/27	EWG および用語集コメントの反映
RC2012	2012/08/15	リリース前バージョン - 最終 NB 編集箇所の反映
Betav3.0	2023/10/31	アルファ版レビューで各国からの全コメントを反映
POST Beta v3.0	2024/01/31	ベータ版レビューで各国からの全コメントを反映
POST Beta v3.0	2024/02/29	校正の細かい修正
RC v3.0	2024/03/28	リリース候補版 - PWG が提案した最新の正式なテンプレートの変更
V.3.0	2024/05/03	誤字脱字や矛盾の修正

JSTQB®

バージョン	日付	備考
Ver1.0	2008/08/31	V2007 の日本語翻訳版
2012.J01	2013/08/26	ISTQB Advanced Level Syllabus Test Manager Version 2012 の日本語翻訳版
2012.J02	2013/10/19	「ソフトウェアテスト標準用語集（日本語版）Version2.2.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
2012.J03	2014/03/10	・1.5 テスト実装の節において、説明を一部変更。

2012.J04	2021/02/19	・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
Version3.0.J01	2025/07/08	ISTQB Certified Tester Advanced Level Test Management Syllabus Version 3.0 の日本語翻訳版
Version3.0.J02	2026/1/14	・「1 テスト活動のマネジメント」および「6 付録 B-学習の目的とビジネス成果のトレーサビリティマトリクス」において、「TM-1.3.1」の説明表現を変更 ・「1.5.2 モデルベースのテストプロセス改善」において、TPINEXT®の説明に関して「キーエリア」と「主要分野」で表現が揺れていたため「キーエリア」に統一
Version3.0.J03	2026/3/10	・「3.2.1 品質コスト」および「3.2.2 テストの費用対効果の関係」において、同じ意味を持つ「評定コスト」、「評価コスト」の表現が揺れていたため、「評定コスト」に統一
Version3.0.J04	2026/6/09	・2 章で使用している用語「フェーズ内阻止」を「フェーズ内封じ込め」に変更

目次

謝辞

本ドキュメントは、2024 年 5 月 3 日に開催された ISTQB® の総会において正式に発行された。

これは、ISTQB のチーム Horst Pohlmann (Product Owner, Vice Chair AELWG), Tauhida Parveen, Francis Fenner, Laura Albert, Matthias Hamburg, Maud Schlich, Tanja Tremmel, Ralf Bongard, Erik van Veenendaal, Jan Giessen, Bernd Freimut, Andreas Neumeister, Georg Sehl, Rabi Arabi, Therese Kuhfuß, Ecaterina Irina Manole, Veronica Belcher, Kenji Onishi, Pushparajan Balasubramanian, Meile Postuma and Miroslav Renda. が執筆した。

テクニカルレビュー（ベータ版）を行った Gary Mogyorodi、校正をしてくれた Julia Sabatine、提案や意見をくれたレビューチームと各国委員会に感謝をしたい。

次のメンバーが本シラバスのレビュー、コメント、投票に参加した。

アルファレビュー: Benjamin Timmermans, Mattijs Kemmink, Rik Marselis, Jean-Francois Riverin, Gary Mogyorodi, Ralf Bongard, Ingvar Nordström, Yaron Tsubery, Imre Mészáros, Mattijs Kemmink, Ádám Bíró, Ramit M Kaul, Chinthaka Indikadahena, Darvay Tamás Béla、Beata Karpinska, Young jae Choi, Stuart Reid, Tal Pe'er, Meile Posthuma, Daniel van der Zwan, Klaudia Dussa-Zieger, Jörn Münzel, Ralf Bongard, Petr Neugebauer, Derk-Jan de Grood, Rik Kochuyt, Andreas Hetz, Laura Albert, Eszter Sebestyeni、Tamás Szőke, Henriett Braunné Bokor, Ágota Horváth, Péter Sótér, Ferenc Hamori, Darvay Tamás Béla, Paul Weymouth, Lloyd Roden, Kevin Chen, Huang qin, Pushparajan Balasubramanian, Szilard Szell, Tamas Stöckert, Lucjan Stapp, Adam Roman、Anna Miazek, Márton Siska, Erhardt Wunderlich, László Kvintovics, Murian Song, Mette Bruhn-Pedersen, Petra Schneider, Michael Stahl, Ramit M Kaul, Imre Mészáros, Dilhan Jayakody, Francisca Cano Ortiz, Johan Klintin, Liang Ren, Ole Chr.Hansen, Zsolt Hargitai, Tamás Rakamazi, Kenji Onishi, Arnika Hryzszko, Rabih Arabi, Veronica Belcher, and Vignesh Balasubramanian.

ベータレビュー: Maria-Therese Teichmann, Dominik Weber, Thomas Puffler, Peter Kunit, Martin Klonk, Michaël Pilaeten, Wim Decoutere, Arda Ender Torçuk, Piet de Roo, Rik Marselis, Jakub Platek, Ding Guofu, Zheng Dandan, Liang Ren, Yifan Chen, Hallur Helmsdal, Ole Chr. Hansen, Klaus Skafte, Gitte Ottosen, Tanzeela Gulzar, Arne Becher, Klaudia Dussa-Zieger, Jan Giesen, Florian Fieber, Carsten Weise, Arnd Pehl, Matthias Hamburg, Stephanie Ulrich, Jürgen Beniermann, Márton Siska, Sterbinszky Ádám, Ágnes Srancsik, Marton Matyas, Tamas Stöckert, Csilla Varga, Zsolt Hargitai, Bíró Ádám, Horváth Ágota, Sebestyéni Eszter, Szilárd Széll, Péter Sótér, Giancarlo Tomasig, Nicola de Rosa, Kaiwalya Katyarmal, Pradeep Tiwari, Sreeja Padmakumari, Seunghee Choi, Stuart Reid, Dmitrij Nikolajev, Mantas Aniulis, Monika Stoecklein-Olsen, Adam Roman, Mahmoud Khalaili, Ingvar Nordström, Beata Karpinska, Armin Born, Ferdinand Gramsamer, Mergole Kuaté, Thomas Letzkus, Nishan Portoyan, Ainsley Rood, Lloyd Roden, Sarah Ireton.

Advanced テストマネージャーシラバス 2010-2012 は、ISTQB Advanced Level テストマネージャー作業部会のコアチームのメンバーである、Rex Black (Chair), Judy McKay (Vice Chair), Graham Bath, Debra Friedenberg, Bernard Homès, Paul Jorgensen, 大西建児, Mike Smith, Geoff Thompson, Erik van Veenendaal, 湯本剛が執筆した。

コアチームは、レビューチームと国内委員会の提案と意見に感謝をしたい。シラバスが完成した時点でワークグループのメンバーは以下の通りであった。（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, 大西建児, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Chair), Geoff Thompson, Erik van Veenendaal, 湯本剛.

次のメンバーが本シラバスのレビュー、コメント、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.

0 イントロダクション

0.1 本シラバスの目的

本シラバスは、テストマネジメント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、本シラバスを以下の主旨で提供する:

1. メンバー委員会に対し、各国語への翻訳およびトレーニング機関の認定の目的で提供する。メンバー委員会は、本シラバスを各言語の必要性に合わせて調整し、現地の出版物に合わせて参考文献を追加することができる。
2. 認定委員会に対し、本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. ISTQB® 認定のトレーニング機関に対し、コースウェアを作成し、適切な教育方法を決定する。
4. 受験志願者に対し、認定試験準備の目的で提供する(ISTQB® は、ISTQB® Advanced Level 試験に参加する前に、ISTQB® 認定トレーニングを受講することを推奨している)。
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て、ソフトウェアおよびシステムテストの専門性を向上させ、書籍や記事のベースとして提供する。

0.2 テスト技術者資格制度 Advanced Level Test Management

Advanced Level資格は、ソフトウェアテストのマネジメントに携わるあらゆる人を対象にする。Advanced Level資格の対象者には、テスト担当者、テストコンサルタント、テストマネージャー、ユーザー受け入れテスト担当者、スクラムマスター、プロジェクトマネージャー、プロダクトオーナーなどの役割の人が含まれる。Advanced Level Test Management資格は、プロジェクトマネージャー、品質マネージャー、ソフトウェア開発マネージャー、ビジネスアナリスト、ITディレクター、マネジメントコンサルタントなど、ソフトウェアテストをより深く理解したい人にも適切である。Advanced Level Test Management資格の保持者は、ISTQB® Expert Levelに進むことができる。ISTQB® Advanced Level - Test Manager もしくは Test Management 資格は生涯有効で、更新の必要はない。この資格は国際的に認知されており、テストマネジメントにおける受験者の専門的能力と信頼性を証明するものである。

0.3 テスト担当者のキャリアパス

ISTQB® スキームは、キャリアのあらゆる段階にあるテストプロフェッショナルを支援する。ISTQB® Advanced Level Test Management 認定資格取得者は、他のコア Advanced Level(例:テストアナリスト、テクニカルテストアナリスト)や、その後 ISTQB® Expert Level(例:テストマネジメント、テストプロセス改善)へと興味を広げられる。アジャイルソフトウェア開発におけるテストのスキルを身に付けたい人は、Agile Technical Tester または Agile Test Leadership at Scale の認定を検討することができる。Specialist コースでは、特定のテストアプローチやテスト活動(テスト自動化、AI テスト、モデルベースドテスト、モバイルアプリテストなど)、特定の業界ドメイン(自動車やゲームなど)のテストのノウハウを集約した分野を深く掘り下げることができる。ISTQB 認定テスト担当者制度に関する最新情報は、の認定テスト担当者スキームの最新情報については、詳細は www.istqb.org を参照のこと。

0.4 ビジネス成果

この節では Advanced Level Test Management の認定取得者に求められる 11 のビジネス成果を掲載している。

Advanced Level Test Management の認定テスト担当者は、次のビジネス成果を達成できる。

TM_01	プロジェクトチームまたはテスト組織のために確立されたテストマネジメントプロセスを適用し、さまざまなソフトウェア開発プロジェクトのテストをマネジメントする。
TM_02	特定のコンテキストで関連するテストのステークホルダーとソフトウェア開発ライフサイクルモデルを識別する。
TM_03	あらゆるソフトウェア開発ライフサイクルにおいてリスク識別とリスクアセスメントセッションを企画し、結果をテスト目的達成に向けてテストの指針として活用する。
TM_04	組織的テスト戦略およびプロジェクトのコンテキストと合致するプロジェクトテスト戦略を定義する。
TM_05	プロジェクトゴールを達成するためにテストを継続的にモニタリングおよびコントロールする。
TM_06	テストの進捗状況をアセスメントし、プロジェクトのステークホルダーに報告する。
TM_07	必要なスキルを特定し、チーム内でそのスキルを開発する。
TM_08	さまざまなコンテキストにおけるテストのコストと期待される利点の概要を示したビジネスケースを準備し、提示する。
TM_09	プロジェクトやソフトウェア開発のプロダクトストリームでテストプロセス改善活動をリードし、組織のテストプロセス改善のための施策に貢献する。
TM_10	必要なテストインフラを含むテスト活動を計画し、テストに必要な労力を見積る。
TM_11	ソフトウェア開発ライフサイクルに適した欠陥レポートおよび欠陥ワークフローを作成する。

0.5 試験対象の学習の目的と知識レベル

学習の目的はビジネス成果を支援し、テスト技術者資格制度 Advanced Level Test Management 試験問題作成を行うために使用する。

一般的に、本シラバスの内容は、「イントロダクション」「参考文献」「あとがき」「付録」を除き、すべて K1 レベルで受験可能である。つまり、本シラバスの 3 つの章のいずれかに記載されているキーワードや概念を認識、記憶、想起することが問われる可能性がある。具体的学習の目的と対応するレベルは、各章の冒頭に示されており、以下のように分類されている:

- K2: 理解
- K3: 適用
- K4: 分析

学習の目的の詳細と例は、付録Aに添付する。

各章の見出しの右側にキーワードとして記載されている用語は、学習の目的に明示されていなくても、すべて(K1)レベルとして、覚えておかなければならない。

0.6 Advanced Level Test Management の資格認定試験

Advanced Level Test Management 認定試験は、本シラバスに基づく。試験問題の解答には、本シラバスの複数の節に基づく資料の使用が要件となる場合がある。本シラバスすべての節は「イントロダクション」、「付録」、「参考文献」を除いて試験対象である。標準や書籍は参考文献として含まれているが（第 4 章）、それらに関して本シラバス自体で要約されている以上の内容は試験対象外である。

詳細については、「Examination Structures and Rules1.1」を参照のこと。

- Advanced Level Test Managementの受験資格：
    - 受験者は、Advanced Level Test Management試験を受ける前に、ISTQB® Foundation Level資格を取得している必要がある。受験者は、ソフトウェア開発またはソフトウェアテストの少なくとも最低限の実務経験、例えば、6ヶ月間のテスト担当者としての経験またはソフトウェア開発者としての経験があることが強く推奨される。

0.7 認定審査

ISTQB® のメンバー委員会にて、本シラバスにしたがったコースを提供するトレーニング機関を認定する。トレーニング機関は、メンバー委員会または認定を行う機関から認定ガイドラインを入手すること。教育コースがシラバスにしたがっていると認定されると、教育コースの一部として ISTQB® 試験を実施することができる。

本シラバスの認定ガイドラインは、ISTQB® のプロセスマネジメント・コンプライアンスワーキンググループが発行する一般的な認定ガイドラインに準ずる。

0.8 標準の取り扱い

Advanced Level Test Management シラバスには参照する標準がある（ISO、IEC、IEEE など）。これらの参考文献の目的は、読者が望む場合の枠組みの提供、追加情報の情報源を提供することである。シラバスではこれらの標準を参考文献として使用することに留意すること。これらの標準は試験を目的とするものではない。標準の詳細については、第 4 章参考文献を参照のこと。

0.9 詳細レベル

本シラバスの詳細なレベルは、国際的に一貫した教育と試験を可能にする。このゴールを達成するために、シラバスは以下のように構成されている：

- Advanced Level Test Managementシラバスの意図について説明する全般的な指導の目的
- 想起することができなければならない用語（キーワード）のリスト

- 達成すべき認知的な学習の成果について説明している、知識領域ごとの学習の目的
- 認められた情報源の参照を含む主要なコンセプトの説明

本シラバスの内容はソフトウェアテストの全知識領域の説明ではない。詳細レベルは、Advanced Level Test Managementのトレーニングコースでカバーされることを示している。本シラバスはすべてのソフトウェアプロジェクトに適用できるテストのコンセプトと技術に焦点を置いて説明している。

0.10 本シラバスの構成

試験可能な内容の章は3つある。各章の一番上の見出しには、認定トレーニングコースがその章の内容をカバーするために最低限必要な学習時間が明記されている。認定トレーニングコースの場合、本シラバスは、最低22.75時間の講義を必要とし、3つの章で以下のように配分する。

- 第1章 テスト活動のマネジメント(750分)
    - 受講者は、テストマネジメント活動(テスト計画、テストモニタリング、テストコントロール、テスト完了)を説明できるようになる。
    - 受講者は、プロジェクトテスト戦略を定義することを学ぶ。テスト対象、組織的テスト戦略やプロジェクトのコンテキストに沿った適切なテストアプローチの選択を含む。
    - さまざまなコンテキストにおけるプロジェクトのマネジメントについて学ぶ。
    - 受講者は、識別したリスクに集中してテストをするためのリスクベースドテストを適用する方法を学ぶ。
    - 受講者は、プロジェクトまたはイテレーションのふりかえりを実施することで、テストプロセス改善活動をリードすることを学ぶ。
    - 参加者は、ツールによる支援のリスク、コスト、および利点を考慮することにより、ツールによるテストの支援を改善する方法を学ぶ。
- 第2章:プロダクトのマネジメント(390分)
    - 受講者は、テスト目的への到達、およびテスト進捗をアセスメントし報告するために、テストメトリクスを使ってテストをモニタリングおよびコントロールする方法を学ぶ。
    - 受講者は、さまざまなソフトウェア開発モデルにしたがって、さまざまなチームで適切なテスト見積り技法を選択することを学ぶ。
    - 受講者は、シーケンシャル、アジャイル、ハイブリッドの各ソフトウェア開発モデルに適合する欠陥マネジメントにおける欠陥ワークフローを定義する方法を学ぶ。
- 第3章 チームのマネジメント(225分)
    - 受講者は、与えられたプロジェクトのコンテキストを分析し、テストチームに必要なスキルを識別する方法を学ぶ。
    - 受講者は、チーム全体アプローチにしたがってチームをマネジメントすることを学ぶ。

- 受講者は、プロジェクトのテスト活動のためにビジネスケースを定義する方法を学ぶ。

注:現在のシラバスには、各学習の目的に対応する節項があり、その内容が記載されている(例:LO-1.2.3には1.2.3項が存在する)。

0.11 シラバスの基本的な前提

本シラバスは、テストマネージャー、テストアナリスト、テストエンジニア、テストコンサルタント、テストコーディネーター、テストリーダー、プロジェクトマネージャーなど、テストマネジメントにおけるAdvanced Levelの能力に達したい人を対象としている。本シラバスは、ISTQB® Foundation Level Syllabus V.4 に準拠しており、ソフトウェアテストの基本的な知識と理解を提供する。

本シラバスでは、テストにおける2つの主な役割、すなわちテストマネジメントの役割とテストをする役割を取り上げる。テストマネジメントの役割は、シーケンシャル開発モデルのコンテキストではテストマネージャーとも呼ばれ、テストマネージャーは通常、プロジェクトマネージャーやプロダクトオーナーとは別の役割である。テストマネジメントの役割は、全体的なテストプロセス、テストチーム、テストマネジメントに責任を持つ。これには、テスト戦略の定義、テスト活動の計画、テスト進捗のモニタリングとコントロール、テスト結果の報告、テストリスクと課題のマネジメントが含まれる。テストマネジメントの役割はまた、テスト目的がビジネスやステークホルダーのニーズと整合していること、そしてテスト活動が他のプロジェクトのステークホルダーと調整されていることを確保する。

テストをする役割もまた、テスト評価、欠陥マネジメント、テスト終了作業を行う。さまざまなテストの技術を用いて、テスト成果物やテスト対象システムの品質と信頼性を確保する。また、テストツールと自動化を使用してテストプロセスを支援し、テストの効率性と有効性を改善する。これらの役割に割り当てられる活動やタスクは、プロジェクト、プロダクト、スキル、組織などのコンテキストによって異なる。(ISTQB® Foundation Level Syllabus V.4 参照)。

テストチームメンバーという用語は、本シラバスでは、組織のコンテキストや他の役割に関係なく、テストマネジメントやテストをする役割にあり、テストを実施するすべての人を指す。テストチームは、さまざまなスキルや能力を持つ個人で構成される。また、チームメンバーの経験や資格のレベルも、Foundation Level、Advanced Level、Expert Levelなど、さまざまである。また、テストチームのメンバーは、アジャイルテスト、モデルベースドテスト、リスクベースドテストなど、使用するテストアプローチやテストプロセスモデルによって、異なる役割と責任を持つこともある。

本シラバスが提供する視点について重要なポイントは、プロジェクトレベルのテストマネジメントに焦点を当てており、組織レベルのテストマネジメントには焦点を当てていないという事実である。したがって、本シラバスは、プロジェクトレベルのテストマネジメント活動には適合し、利用できる情報を含んでいるが、組織レベルのテストマネジメントにはあまり適合していない。

ハイブリッドソフトウェア開発とは、V字モデル、イテレーティブ開発モデル、インクリメンタル開発モデル、アジャイル開発モデルなど、さまざまなソフトウェアライフサイクルモデルの要素を組み合わせたソフトウェア開発アプローチを指す。ハイブリッドソフトウェア開発は、プロジェクトのコンテキストやニーズに応じて、各モデルの長所を活用し、短所を軽減することを目的としている。例えば、ハイブリッドソフトウェア開発アプローチでは、最初の計画と要件分析のフェーズではV字モデルを使用し、その後の設計、開発、テストのフェーズではアジャイルモデルを使用することができる。あるいは、ハイブリッドソフトウェア開発アプローチでは、プロジェクト全体のマネジメントにはイテレーティブ開発モデルを使用し、イテレーションごとにインクリメンタルモデルを適用し、インクリメントごとにアジャイルモデルを適用することもできる。ハイブリッドソフトウェア開発では、ステークホルダー間の柔軟性、コミュニケーシ

ョン、コラボレーションを高めることが求められるとともに、各フェーズとモデルのゴール、リスク、制約に対する明確な理解が求められる。

本シラバスと用語集によると、テスト戦略とは、与えられた状況下でテスト目的を達成するために、どのようにテストを実施するかを記述したものである。テスト戦略は、システムまたはプロダクトをテストするための全体的なスコープ、アプローチ、およびリソースを定義する。テスト戦略は、通常、テスト計画書の中に文書化されるか、あるいは、テストのコンテキストに応じて、他の文書の一部として文書化される。テスト戦略は、組織的テスト戦略に影響を受ける。組織的テスト戦略は、組織でどのようにテストが行われるかを記述した、高いレベルのテスト戦略である。テスト戦略は、異なる目的、ターゲット、基準に焦点を当てたテストの具体的な側面である1 つのテストレベルまたはテストタイプのために存在することもある。「テスト戦略」という総称は、どのようなコンテキスト（プロジェクト、組織、プロダクト）でも使用することができる。テストアプローチとは、テストタスクの実装方法であり、特に静的テストと動的テストのためのテストレベル、テストタイプ、テスト技法、およびテスト実践（スクリプトテスト、手動テスト、バックツーバックテストなど）の選択と組み合わせである。テストマネジメントの役割によって選択されるテストアプローチは、与えられたコンテキストに対して適切なテスト戦略を策定する際の重要な意思決定である。

1 テスト活動のマネジメント - 750 分

キーワード

経験ベースのテスト、機能テスト、ハイブリッドソフトウェア開発モデル、インクリメンタル開発モデル、イテレーティブ開発モデル、非機能テスト、プロダクトリスク、品質リスク、ふりかえり、リスク分析、リスクアセスメント、リスク識別、リスク影響、リスクレベル、リスク可能性、リスクマネジメント、リスク軽減、リスクモニタリング、リスクベースドテスト、シーケンシャル開発モデル、スマートの法則、ソフトウェア開発ライフサイクル、テスト完了、テストコントロール、テストレベル、テスト成熟度モデル統合、テストモニタリング、テスト目的、テスト計画書、テスト計画、テストプロセス改善、テスト戦略、テストタイプ、TPI NEXT

ドメイン固有のキーワード

ゴール クエスチョン メトリクス(GQM)、IDEAL、指標、尺度、メトリクス

第 1 章の学習の目的

1.1 テストプロセス

TM-1.1.1 (K2) テスト計画を要約する。

TM-1.1.2 (K2) テストモニタリングとテストコントロールを要約する。

TM-1.1.3 (K2) テスト完了を要約する。

1.2 テストのコンテキスト

TM-1.2.1 (K2) さまざまなステークホルダーがテストに関心を持つ理由を比較する。

TM-1.2.2 (K2) テストマネジメントにおいてステークホルダーの知識が重要である理由を説明する。

TM-1.2.3 (K2) ハイブリッドソフトウェア開発モデルで行うテストについて説明する。

TM-1.2.4 (K2) さまざまなソフトウェア開発ライフサイクルでのテストマネジメント活動を要約する。

TM-1.2.5 (K2) さまざまなテストレベルでのテストマネジメント活動を比較する。

TM-1.2.6 (K2) さまざまなテストタイプのテストマネジメント活動を比較する。

TM-1.2.7 (K4)テスト計画、テストモニタリング、テストコントロールに重点を置いたテストマネジメント活動を決めるために、特定のプロジェクトを分析する。

1.3 リスクベースドテスト

TM-1.3.1 (K2) リスクベースドテストにてリスクに対応するために取るさまざまな手段を説明する。

TM-1.3.2 (K2) テストマネージャーがプロダクト品質に関連するリスクを識別するために使用できるさまざまな技法の例を挙げる。

TM-1.3.3 (K2) プロダクト品質に関するリスクレベルを決定する要因について要約する。

TM-1.3.4 (K4) 特定のコンテキストにおけるリスクレベルに応じたリスク軽減するための適切なテスト活動を選択する。

TM-1.3.5 (K2) リスクベースドテスト技法について重量的と軽量的の例を区別する。

TM-1.3.6 (K2)リスクベースドテストに関連する成功メトリクスと困難さの例を挙げる。

1.4 プロジェクトテスト戦略

TM-1.4.1 (K2) テストアプローチの典型的な選択肢を説明する。

TM-1.4.2 (K4) 組織的テスト戦略とプロジェクトのコンテキストを分析し、適切なテストアプローチを選択する。

TM-1.4.3 (K3) 測定可能なテスト目的と終了基準を定義するために、スマートの法則を使用する。

1.5 テストプロセス改善

TM-1.5.1 (K2) 特定のプロジェクトにおけるテストプロセス改善のための IDEAL モデルの使用方法を説明する。

TM-1.5.2 (K2) テストプロセス改善のためのモデルに基づく改善アプローチを要約し、プロジェクトのコンテキストに合わせて適用する方法を理解する。

TM-1.5.3 (K2) テストプロセス改善のための分析に基づく改善アプローチを要約し、プロジェクトのコンテキストに合わせて適用する方法を理解する。

TM-1.5.4 (K3) テストプロセスを評価し、改善すべきテスト領域を発見するためにプロジェクトまたはイテレーションにふりかえりを実装する。

1.6 テストツール

TM-1.6.1 (K2) ツール導入の最良の実践例を要約する。

TM-1.6.2 (K2) ツールの種類を決定する際にさまざまな技術的側面とビジネス的側面へ与える影響について説明する。

TM-1.6.3 (K4)ツールの選択、カバーできるリスク、コスト、および利点を考慮した計画を立てるために特定の状況を分析する。

TM-1.6.4 (K2) ツールのライフサイクルの各段階を区別する。

TM-1.6.5 (K2) ツールを利用したメトリクス収集と評価の例を挙げる。

1.1 テストプロセス

イントロダクション

ISTQB® Foundation Level Syllabus V.4 シラバスでは、テスト計画、テストモニタリングとコントロール、テスト分析、テスト設計、テスト実装、テスト実行、テスト完了を含むテストプロセスを記している。

ISTQB® Foundation Level Syllabus V.4 では、テストプロセスにおけるこれらの活動は、ソフトウェア開発ライフサイクル (SDLC)モデルとプロジェクトのコンテキストに応じて、イテレーティブまたは並行して実装されることが多いと述べている。通常、プロダクトやプロジェクトのコンテキストに応じて、これらの活動を調整することが求められる。

本シラバスでは、以下の主要なテストマネジメント活動に焦点を当てる:

- **テスト計画:** テスト目的、テストアプローチ、テストスコープ、テストリソース、テストスケジュール、テスト成果物、テスト参加者(テストステークホルダー)を定義する。
- **テストモニタリングとコントロール:** テスト進捗、テスト結果、テスト逸脱を追跡し、必要に応じて是正措置を講じ、関連するステークホルダーにテストステータスと結果を報告する。
- **テスト完了:** テスト成果物の完成と保管、テストプロセスとテスト成果物の評価、テストプロセス改善アクションの識別、関連ステークホルダーへのテスト終了の伝達をする。

ISO/IEC/IEEE 29119-2 標準は、これらのテストマネジメント活動をカバーするテストマネジメントプロセスを定義している。これらのテストマネジメントプロセスは、プロジェクト、プログラム、ポートフォリオなどが異なるテストのレベルで適用することができる。各テストのレベルは、上位のテスト計画書と整合する各レベル固有のテスト計画書を持つことができる。

1.1.1 テスト計画の活動

本項では、プロジェクト全体、テストレベル、テストタイプ、アジャイルにおけるリリース/イテレーション/スプリントなど、さまざまなスコープに対するテストの計画の活動に焦点を当てる。テスト計画は、スコープによって、計画の開始と終了が開発プロセスの異なる時点になる場合がある。テスト計画は、テストポリシーで識別したテスト目的を達成するために必要な活動やリソースを識別する活動である。テスト計画は、開発プロセスのできるだけ早い段階、できれば要件を識別する前に開始すべきであり、プロジェクトが進行するにつれて更新していくべきである。テスト計画は、多くの場合、プロジェクト中に変更やフィードバックに対応するために再計画を必要とするイテレーティブなプロセスである。

以下のタスクは、テスト計画の一部である(ISO/IEC/IEEE 29119-2 にあるものと同様):

- **コンテキストを理解し、テスト計画の段取りを整える**
  組織のコンテキスト(例えば、テストポリシーや組織的テスト戦略)、テストのスコープ、テストアイテム(すなわち、テストされる作業成果物)を理解することは、テスト計画において非常に重要である。(1.2.節「テストのコンテキスト」参照)。これには、テスト計画書の開発を進めていくために必要なすべての活動や、それらの活動およびスケジュールをステークホルダー(例えば、プロダクトオーナー、プロジェクトマネージャー、開発チームマネージャー)から承認を得るための活動が含まれる。
- **プロダクトリスクの識別と分析**

リスク分析では、テスト計画書の一部として、プロダクトリスクの潜在的な影響と可能性を識別し、アセスメントする。プロダクトリスクの詳細については、本シラバスの1.3節「リスクベースドテスト」を参照のこと。

- **リスク対応アプローチを識別する。**
  リスク分析に基づき、適切なリスク対応アプローチを選択し、テスト計画書に文書化する。これらには、識別したリスクに対処するための予防措置、是正措置、軽減措置を含む。
- **テストアプローチを定義し、テストリソースを見積り、配置する。**
  組織的テスト戦略、規制標準、およびプロジェクトから与えられた制約、およびリスク対応アプローチに基づいて、現在のテストスコープに対するテストアプローチを定義する(1.4節「プロジェクトテスト戦略」を参照)。テストアプローチが定義されたら、テスト要員、テストツール、テスト環境、テストデータなど、必要なテストリソースを見積り、それらのリソースをテスト活動に割り当てることが重要である。
- **テスト計画書の作成**
  テスト計画書は、すべてのステークホルダーに受け入れられなければならないので、ステークホルダー間の意見の相違は解決すべきである。

1.1.2 テストモニタリングとコントロールの活動

テストマネジメントが効率的なテストコントロールを行うためには、テストスケジュールとモニタリングの枠組みを確立し、テストのステータスと進捗状況を追跡できるようにしなければならない。この枠組みには、テスト作業成果物やリソースの状況を、計画書や戦略の目的に関連付けるために必要となる詳細な尺度や目標値が含まれていなければならない。

小規模で複雑度の低いプロジェクトの場合、テスト作業成果物や活動を計画書や戦略の目的に関連付けることは比較的容易かもしれないが、一般的にこれを達成するためには、より詳細な目的を定義しなければならない。これには、テスト目的やテストベースのカバレッジを満たすために必要な尺度や目標値を定義することも含む。

特に重要なのは、プロジェクトやビジネスのステークホルダーにとって理解しやすく、関連性のある形でテスト作業成果物や活動のステータスを伝える必要があることである。

テストモニタリングとコントロールは継続的な活動である。

テストコントロールでは、テスト計画書に対して実際の進捗状況を比較し、必要に応じて是正措置を実装する。これにより、テスト戦略とテスト目的(1.4 節「プロジェクトテスト戦略」参照)を満たすようにテストを行う指針を示し、必要な場合にはテスト計画を検討し直す。コントロールで使うデータは、適切な対応をするために詳細なテスト計画の情報が必要となる。この活動には以下を含む:

- テスト計画書とコントロールの指示の実装
- 計画したテストからの逸脱のマネジメント
- 新たに識別したリスク、変化したリスクへの対処
- テスト開始の準備の確立
- 終了基準に基づくテスト完了の承認の実行と承認の獲得

テストモニタリングには、テスト結果の収集と記録、計画したテストからの逸脱の識別、テストを必要とする新たなリスクの識別と分析、識別したリスクに対する変更のモニタリングを含む。

1.1.3 テスト完了の活動

テスト完了は通常、プロジェクトのマイルストーン（リリース、イテレーションの終了、および/またはテストレベルの完了など）で発生する。未解決の欠陥については、変更要求やプロダクトバックログアイテムが作成される。ISTQB® Foundation Level Syllabus V.4 を参照のこと。終了基準が満たされたら、主要なアウトプットを取得し、保管し、関連するステークホルダーに提供する。テスト完了には、以下の作業が求められる：

- **テスト完了レポートを作成し、承認する**
  このタスクでは、すべてのテストを達成し、すべてのテスト目的を満たしていることを確認する。このタスクは、テスト計画書、テスト結果、テスト進捗レポート、テスト完了レポート、欠陥レポートなど、さまざまなテストウェアから関連情報を収集する。収集した情報を評価し、テスト完了レポートにまとめる。テスト完了レポートの承認の後、関連するステークホルダーへ通知する。
- **テストウェアを保管する**
  このタスクでは、典型的にはテストケースのような、将来役に立つ、あるいは再利用が期待されるテストウェアを識別する。これにより、将来の再利用のために、アクセスしやすく、理解しやすくなる。また、テスト結果、テスト記録、テストレポートなどのテストウェアは、構成管理システムに一時的に保管すべきである。
- **テストウェアを引き渡す**
  このタスクでは、必要とする人にとって価値ある作業成果物を引き渡す。例えば、延期された、あるいは受け入れられた既知の欠陥は、テストウェアを使う人々、あるいはテストウェアの使用をサポートする人々に伝達すべきである。
- **テスト環境をクリーンアップし、事前に設定した状態に戻すために必要なすべてのタスクを実行する**
  このタスクでは、テスト環境を次のテストサイクルやプロジェクトに対応できる状態にする。テスト環境からテストデータ、テストツール、テストドライバー、テストスタブ、テストスクリプトを削除する。また、テスト環境を元の状態または希望する状態に戻すことも含む。
- **学んだ教訓を実施／収集／文書化する**
  このタスクは、テストプロセス中に学んだ重要な教訓を議論し文書化するふりかえりで実施する。これには、ソフトウェア開発ライフサイクル(SDLC)全体に関する発見を含む。学んだ教訓は、本シラバスの 1.5. 節「テストプロセス改善」で示すように、テストプロセスの改善に利用できる。

1.2 テストのコンテキスト

イントロダクション

テストのコンテキストは、テストプロセスに影響を与える固有の条件と制約を包含し、テスト計画、テスト設計、テスト実行のための決定と戦略を形成する。テストマネージャーは、テストをソフトウェア開発プロジェクトの特定のニーズと目的に適合させるために、コンテキストを把握することが不可欠である。このコンテキストは、プロダクトのタイプ、業界、規制要件、そして重要なのは採用するソフトウェア開発ライフサイクル(SDLC)によって異なることがある。

テストマネージャーの仕事は、確立したテスト戦略を適用し、テスト技法を選択することであり、それらを開発することではない。テストマネージャーは、プロジェクトのコンテキストに合わせたテスト計画を策定する重要な役割を果たす。さまざまな要因を理解して考慮することによって、テストマネージャーは、テストが適切で、効果的で、効率的にテスト目的を達成することを確実にすることができる。

1.2.1 テストステークホルダー

テストステークホルダーとは、プロダクトの品質に直接的または間接的な関心を持つ個人またはグループである。以下は、潜在的なステークホルダーの典型的なリストであり、テストにおける多様な関心を反映するように調整している:

- **開発担当者、開発リーダー、開発マネージャー:** テスト対象システムの実装とテスト結果への対応に加え、これらのステークホルダーはユニットテストにも関与し、テストプロセスに貢献する。
- **テスト担当者、テストリーダー、テストマネージャー:** これらの個人は、テスト計画書を作成し、要件分析、テスト設計、テスト実行、欠陥追跡と報告、テスト自動化、テスト進捗報告などの活動を通じて、テストプロセスに貢献することを含むテストウェアを準備する。
- **プロジェクトマネージャー、プロダクトオーナー、ビジネスユーザー:** 要件を仕様化し、求められる品質レベルを定義し、認識したリスクに基づいて必要なカバレッジを推奨する。また、作業成果物をレビューし、ユーザー受け入れテスト(UAT)に参加し、テスト結果に基づいて判定を行う。
- **運用チーム:** 運用受け入れテストに従事し、システムの本稼働への準備を確実にし、非機能要件の定義に貢献する。
- **顧客とユーザー:** 顧客はプロダクトを購入し、ユーザーはプロダクトを直接利用する。両者とも要件定義の鍵を握っている。両者はプロダクトが自分たちのニーズを満たしていることを確認するためにユーザー受け入れテスト(UAT)に参加すべきである。

このリストには、すべての潜在的なステークホルダーが含まれているわけではない。テストマネージャーは、テスト戦略やテスト計画の作成の一環として、テストスコープを考慮しつつ、プロジェクトの特定のステークホルダーを識別するステークホルダー分析を行わなければならない。

1.2.2 テストマネジメントにおけるステークホルダーの持つ知見の重要性

テストマネジメントでは、さまざまなステークホルダーの視点と影響力を考慮することが極めて重要である。ステークホルダーマトリクスは、しばしば権力と関心度のマトリクスと呼ばれ、テストマネージャーがステークホルダーの関与

に優先順位を付けて効率的に期待値をマネジメントするための指針となる。ステークホルダーマトリクスは、以下のようにテストマネジメントにおける戦略的なツールである:

- ステークホルダーの専門知識を活用し、エンドユーザーや技術チームが性能やセキュリティに関するフィードバックや洞察を提供する。
- ステークホルダーの関心と影響力を明らかにし、積極的なリスク軽減の努力を促すことで、リスクマネジメントを支援する。
- 価値あるフィードバックによる多様な視点を重視する。

ステークホルダーマトリクスは 4 つの象限で構成されている:

- **推進者(影響力が高い、関心が高い):** 高い影響力と関心を持つ主要な協力者であり、テスト戦略やテスト計画の形成に不可欠である。
- **潜在的協力者(影響力が高い、関心が低い):** 日々のタスクには強い関心を持たないかもしれないが、リソース配置や高いレベルのプロジェクトの方向性に関する彼らの意思決定が重要である。
- **擁護者(影響力が低い、関心が高い):** 定性的なフィードバックを提供することが多く、定期的な報告や特定の議論への関与を通じて関心を維持できる。
- **無関心者(影響力が低い、関心が低い):** 密接には関与しないが、重要なマイルストーンで報告したり、特定の課題に関する意見を求めたりすることで、独自の洞察を得ることができる。

テストマネージャーの役割には、詳細なステークホルダーリストを作成し、各ステークホルダーとテスト活動との関連を理解することを含み、ステークホルダーマトリクスを使用してテストマネジメント実践の有効性を高めることが求められる。

図 1.さまざまなタイプのステークホルダー

1.2.3 ハイブリッドソフトウェア開発モデルにおけるテストマネジメント

ハイブリッドソフトウェア開発モデルは、特定のプロジェクトのニーズまたは組織の移行に合わせて、従来のシーケンシャルアプローチとアジャイルプラクティスの両方の要素を統合したものである。以下は、ハイブリッドソフトウェア開発モデルを使用する一般的な理由であるが、組織やプロジェクトによっては、他の理由もあり得る:

- **アジャイルへの移行としてのハイブリッド**：従来の手法からアジャイル手法への移行は、業務の流れ、文化、チームダイナミクスの根本的な変化により、困難な場合がある。ハイブリッドモデルは、従来の手法の構造とアジャイルプラクティスの柔軟性を組み合わせることで、この移行を容易にするバランスの取れたアプローチを提供する。
- **目的に合わせたハイブリッド**：組織やプロジェクトによっては、アジャイルに移行できない場合がある。リスクの高いプロジェクトでは、あるものにはシーケンシャルなタスクが要件となり、別のものにはアジャイルプラクティスが要件となることがある。そのようなプロジェクトでは、目的に合わせてハイブリッドモデルを使うことができる。

ハイブリッド環境では、テストマネジメント活動には以下を含む場合がある：

- 従来の方法論とアジャイル方法論の間を円滑に移行するためのチームの理解と能力を評価する。
- ハイブリッドアプローチへの適応における強みと弱みを識別する。
- チームが構造化されたプロセスとアジャイルな柔軟性を組み合わせることに習熟していることを担保する。
- テストチームとステークホルダーとの協調を強化し、スプリントや従来のテストフェーズにおけるテストをよりよくマネジメントする。
- テスト担当者向けのスクラムオブスクラムのような協調的な取り組みに参加し、全体的な開発目的に貢献しながら、テストへの集中を維持する。
- アジャイルプラクティスに沿っていることを確認するために、スプリント内で行うテスト活動とテストケースを追跡してレビューする。

さらに詳しい情報は、(Fowler, 2010)で確認できる。

1.2.4 さまざまなソフトウェア開発ライフサイクルモデルにおけるテストマネジメント活動

SDLC モデルの中にテストを適切に整合させるために、テストマネージャーは、組織で使用されているさまざまな SDLC モデルを理解し、テストを開発活動と適切に整合させるためにその知識を活用しなければならない。

下表は、さまざまな SDLC モデルに基づくさまざまなテストマネジメント活動の比較を示している：

活動	シーケンシャル開発モデル 例：V 字モデル	イテレーティブ開発モデル 例：スクラム
見積り	テストレベルごとの早期詳細見積り	イテレーションごとのストーリープランニングの一部としてのイテレーティブな見積り
テストウェア	戦略、計画書、ケース、スケジュール、レポートを含む	受け入れ基準と完了の定義に重点を置き、文書化は最小限にとどめる
役割	テストマネージャーは意思決定とチームマネジメントを統括する	役割は統合され、ファシリテーターやコーチが従来のテストマネージャーに取って代わる
ツール	主にフェーズベースのテストに適したテストマネジメントツール	CI/CD と自動化のためのツールが中心で、継続的なテストをサポートする

活動	シーケンシャル開発モデル 例:V 字モデル	イテレーティブ開発モデル 例:スクラム
テストアプローチ	プロジェクトのフェーズに対応し、事前にスケジュールする	適応性とフィードバックに重点を置き、イテレーションの中に組み込む
テスト自動化	戦略的に実装し、さまざまな段階で実施できる	CI/CD における自動リグレッションに重点を置き、初期段階から組み込まれている
モニタリングと報告	マイルストーンベースの報告、オプションの自動ダッシュボード	リアルタイムダッシュボードとデイリーステータスアップデートによる継続的な報告をする
メトリクス	従来のテストメトリクスと欠陥マネジメントに焦点を当てる。(テスト実行率、欠陥率など)	従来のメトリクスに加え、イテレーション追跡のためのアジャイルメトリクスを含む(例えば、チームのベロシティ、バーンダウンチャート)

表 1: さまざまな SDLC モデルにおけるテストマネージャーの活動

1.2.5 さまざまなテストレベルにおけるテストマネジメント活動

コンポーネントテスト:

- コンポーネントテスト(ユニットテスト)のスコープ、目的、完了基準を定義する。
- テスト担当者を、コードレビューのような従来のテストの役割を超えた活動に参加させ、彼らの分析スキルが付加価値を生むようにする。
- 課題事項の解決やユニットテストの貢献のために開発チームと調整する。

コンポーネント統合テスト:

- SDLC モデル、ツール、プロセスを考慮しながら、開発チームと協力して統合順序とテストの組み合わせを決定する。
- システムテスト戦略や受け入れテスト戦略と整合するように進捗を監視する。
- コンポーネント(ユニット)統合テストを考慮しながら、開発担当者とも協力してこのフェーズをマネジメントする。

システム統合テスト:

- システム統合テストのスコープと目的が明確で、リスクアセスメントと品質目標に適合していることを確認する。
- システム統合テスト中の進捗、結果、課題事項マネジメントを継続的に監視する。

システムテスト:

- リソースの割り当て、ツールの選択、スケジューリングを慎重に行い、SDLC モデルに合わせて計画を調整する。

- アジャイルプロジェクトでは、システムテストをイテレーティブなストーリーテストと統合し、独立したテストフェーズを避け、テストが継続的かつ統合的であるようにする。シーケンシャルモデルでは、テストは計画された段階にしたがうことがある。

受け入れテスト:

- ステークホルダーと協力し、受け入れ基準の充足のレビューと確認を行い、UAT におけるユーザーテストのマネジメントを含むテスト活動を計画する。
- 受け入れテストの段取りを調整し、顧客先でのテストを円滑に進めて、開発環境外でプロダクトがビジネスニーズと品質基準を満たしていることを確認する。
- UAT に関するあらゆる課題事項の解決を促進し、受け入れ基準を満たした後に、プロダクトのリリース承認プロセスへステークホルダーを導く。

1.2.6 テストタイプ別のテストマネジメント活動

効果的なテストマネジメントには、機能テスト、非機能テスト、ブラックボックステスト、ホワイトボックステストに特有の要求を考慮した統合的なアプローチが必要である。機能テストを実施するマネージャーは、すべての機能性が徹底的にテストされ、定義した要件を満たしていることを確認することに重点を置く。非機能テストのマネジメントは、性能やセキュリティといったシステム属性の検証を中心に展開する。ブラックボックステストのマネジメントは、テストがユーザーに焦点を当てたものであり、想定されるすべての外部との相互作用を確実にカバーしていく。ホワイトボックステストのマネジメントは、コード構造を理解し、テストが内部ロジックを徹底的にカバーすることを重視する。

機能テストのマネジメント:

- 戦略的な計画と進捗の追跡:機能要件とプロジェクト目的に沿った詳細なテスト戦略を策定し、進捗モニタリングを行う。
- リソースの調整:システムのすべての機能的側面をカバーするために、人的および技術的リソースを効率的に割り当てる。

非機能テストのマネジメント:

- 性能ベンチマーク:性能ベンチマークを設定し、これらの基準に照らしてシステムをアセスメントするテスト活動をマネジメントする。
- コンプライアンス検証:システムがセキュリティ、使用性、信頼性などの非機能標準を満たしていることを保証するテストを統括する。

ブラックボックステストのマネジメント:

- テストカバレッジ分析:ブラックボックステストがすべてのユーザーシナリオとビジネス要件をカバーしていることを確認する。
- フィードバックの取り込み:ブラックボックステストアプローチや欠陥修正を洗練するために、ステークホルダーからのフィードバックを収集するプロセスをマネジメントする。

ホワイトボックステストのマネジメント:

- コードカバレッジの最適化:ホワイトボックステストでのギャップを識別するためにコードカバレッジツールの使用を統括し、これらの領域にリソースを割り当てる。
- 技術的洞察の統合:テスト計画プロセスの中で、アプリケーションの内部動作を理解した上でテストを設計するなどの技術的な知見が反映されるようにする。

1.2.7 計画、モニタリング、コントロールのためのテストマネジメント活動

効果的なテストマネジメントは、成功するテスト活動の礎であり、慎重な計画、注意深いモニタリング、戦略的なコントロールを必要とする幅広い活動を包含している。テストマネージャーは、テストプロセスが効果的かつ効率的であるだけでなく、そのプロジェクト特有の要求に合わせて調整されるようにする上で、極めて重要な役割を果たす。

テスト計画:

- **包括的なスコープ定義:**テスト計画書は、包括的なスコープ定義を含むように慎重に作成されるべきである。これには、完全なテストカバレッジを確保するために、すべての機能要件と非機能要件を識別することを含む。また、ブラックボックステストとホワイトボックステストの両方の方法論を考慮し、開発したテストケースであらゆる角度からシステムの妥当性を確認できるようにすることも含む。
- **リスクアセスメントと軽減計画:**テスト計画の不可欠な要素として、強固なリスクマネジメントの枠組みがある。テストマネージャーは詳細なリスク分析を行い、プロジェクトの一連の進行や最終的なプロダクトに影響を与える潜在的な脆弱性や課題を突き止めなければならない。軽減戦略の策定は重要であり、リスクを効果的に回避または最小限に抑えるための予防的な計画を含む。
- **リソース配置戦略:**リソース計画も重要な要素である。これは単なる配置にとどまらず、チームの構造を定義し、役割を明確にし、コミュニケーションの手順を確立することを含む。オンサイト／オフサイトモデルのようにチームが分散している環境では、同期を保ち、円滑な協力を確保するために、この点を維持することが特に重要になる。

テストモニタリング:

- **実行の監視**:モニタリングは、テストマネジメントプロセスにおいて中心的な役割を果たす。これには、確立した計画に照らしてテスト実行を継続的にレビューし、テストケースの進捗を追跡し、表面化した欠陥をマネジメントすることを含む。リスクアセスメントとリアルタイムな開発状況に基づいてテストの優先度を調整することで、テストが最も重要な領域に集中し、一貫性を持つことを保証する。
- **ツールと環境の最適化**:テストツールとテスト環境の賢明な選択と使用は、テスト戦略をサポートするために非常に重要である。継続的モニタリングによって、それらが CI/CD パイプラインに効果的に統合され、アジャイル開発プロセスに不可欠な継続的テストと即時フィードバックループが促進される。
- **開発チームとのコラボレーション**:テストを成功させるためには、開発チームと緊密な協力関係を維持することが不可欠である。この協力関係では、テストへの包括的なアプローチをサポートし、ホワイトボックスとブラックボックスの両方の視点からの洞察を活用して、潜在的な課題事項に事前に対処すべきである。

テストコントロール:

- **適応的なプロセスマネジメント:** テストコントロールとは、新たな洞察や課題、進化するプロジェクトのダイナミクスに対応して、テストプロセスを動的に調整することである。テストマネージャーには、プロジェクトの現状を反映したテストアプローチの変更を実装できるような対応力と柔軟性が求められる。
- **品質ゲートマネジメント:** 品質ゲートマネジメントに対する構造的なアプローチは基本である。これには、テストライフサイクル内で何が品質ゲートかを定義し、テストフェーズの進行においてプロダクトの整合性を維持するために重要となる情報に基づいた意思決定を行うことを含む。

テスト計画、テストモニタリング、テストコントロールの具体的な活動に焦点を当てることで、テストマネージャーは、テストプロセスを明確に定義し、プロジェクトの変化に適応し、プロジェクトの要件とステークホルダーの期待の両方を満たすプロダクトの提供を確実にすることができる。

1.3 リスクベースドテスト

イントロダクション

リスクベースドテストには、テストを主導するためのリスクの識別、アセスメント、モニタリング、軽減を含む。これらのリスクは多様なステークホルダーが識別し、テストの選択と優先順位付けで利用する。リスクレベルが高いほど、テストは早期に開始して、労力をより集中させて長期間にわたって行うべきである。

1.3.1 リスク軽減活動としてのテスト

プロダクトリスクとは、プロダクトに品質問題が存在する可能性のある状況のことである。テストで欠陥が明らかになった場合、リリース前に欠陥に気づき、対処する機会を提供することで、テストはプロダクトリスクの軽減に役立っている。テストで欠陥が発見されなかった場合、テストはプロダクトリスクレベルが予想よりも低いことを示す。

中でもテストマネージャーは、プロダクト品質の正確で信頼性の高い評価を提供する責任がある。そのためには、品質保証に関連するプロジェクトリスク（例えば、終盤に行う妥当性確認で大きな問題を引き起こす曖昧な要件、テスト実行を阻害する不十分なテスト環境）に焦点を当てた、プロジェクトリスクマネジメントへの積極的な関与が必要である。

リスクベースドテストとは、品質リスクに焦点を当てたテストを行うことである。これは、一般的なリスクマネジメントプロセスにしたがった以下の活動が主なものとなる。:

- リスク分析は、リスク識別およびリスクアセスメントからなる。
- リスクコントロールは、リスクモニタリングとリスク軽減からなる。

これらの主な活動は、論理的に順序立てて構成されているが、重複することもある。

品質リスクにテストを集中させるためには、品質リスクを識別し、アセスメントしなければならない。最も効果的であるためには、リスク分析には多様なステークホルダーを含めるべきである。テストマネージャーは、品質リスク分析の主要なステークホルダーとして、これらの活動を理解し、モニタリングし、ステークホルダー間の調整を行うべきである。

テストモニタリングにはリスクモニタリングを含めるべきである。既知の品質リスクの進展をモニタリングすることに加え、新たな品質リスクを分析し、リスクレジスターを調整することも含めるべきである。

テストマネージャーは、品質リスクの軽減を推進する複数の人の中の 1 人である。リスク軽減は、いくつかのテスト活動に分散して行われる。例えば、品質リスク分析の結果は、正しい技法を使って正しい領域にテストを集中させるためにテスト計画に使用する。テスト分析では、リスクレベルがカバーすべきテスト条件の選択の指針となる。テスト実行では、リスクを基にした優先順位付けが、テスト実行の順序を決定する。

1.3.2 品質リスクの識別

テストマネージャーの仕事は、ステークホルダーからリスクを収集することである。ステークホルダーは、以下の技法の 1 つ以上を通じて、品質リスクを識別することができる:

- 専門家インタビュー

- 独自のアセスメント
- ふりかえり
- リスクワークショップ
- ブレーンストーミング
- チェックリスト
- 過去の経験の参照

リスク識別プロセスでは、多くの場合、可能な限り広範なステークホルダーが参加することにより、重大なプロダクトリスクのほとんどを識別できる。この段階で、どのステークホルダーを参加させるかの識別は非常に重要である。参加するステークホルダーのリストが包括的であり、プロジェクトマネージャーと合意していることを確認することが不可欠である。重要なステークホルダーが欠けているリスク識別は、大きな問題となる可能性がある。重要なのは、関係するステークホルダー全員に参加の機会を与えることである。参加できない場合は、少なくともタスクを委任する機会を与えるべきである。主要なステークホルダーの意見が反映されない可能性がある場合、キックオフミーティングを利用して、主要なステークホルダー意見が欠如しているかどうかを判断することができる。

リスクベースドテストでは、リスクはテスト対象内に一様に分布しないことを理解することが重要である。例えば、セルフサービスアプリケーションの顧客向けコンポーネントは、管理側コンポーネントとは全く異なる使用性リスクを持つことがある。さまざまなテストアイテムの個々のリスクを識別することは、テスト計画において重要なタスクである。

リスク識別は、しばしば副産物（プロダクトリスクではない課題事項の識別）を生み出す。例として、プロダクトやプロジェクトに関する一般的な疑問や課題事項、要件や設計仕様書などの参照ドキュメントにおける問題が含まれる。プロジェクトリスクは、品質リスク識別の副産物としてしばしば識別されるが、リスクベースドテストの焦点ではない。テストマネージャーは、これらの副産物を強調し、品質が全員の関心事であることを明確にする上で重要な役割を果たすことが多い。品質保証活動が SDLC 全体を通じて関与していると、欠陥のある要件や欠落した要件は、計画と準備におけるより根本的な問題を示していることが多い。

1.3.3 品質リスクアセスメント

リスクを識別すると、それをアセスメントすることができる。品質リスクアセスメントは、リスクタイプ（プロダクトリスクまたはプロジェクトリスク）および影響を受ける品質特性によるリスクの分類を含む。

リスクレベルの決定は通常、各リスクアイテムについて、リスク可能性とリスク影響のアセスメントを含む。品質リスクにおけるリスク可能性を左右する要因には、以下のようなものがある：

- 技術、ツール、またはシステムアーキテクチャの複雑度
- 組織の成熟度
- 使用している SDLC に関する知識を含む、スキル、可用性、モチベーション、または作業に関する自主性といった個人の課題事項
- チーム内の対立
- サプライヤーとの契約上の問題
- 地理的に分散したチーム

- 経営陣や技術陣のリーダーシップが弱い
- 時間、リソース、予算、マネジメントからのプレッシャー
- 早期の品質保証活動の欠如
- テストベース、プロダクト、人員の変更率が高い

リスク影響を左右する要因には以下を含む:

- 影響を受ける機能の使用頻度
- 影響を受ける機能の重要性
- 影響を受けるビジネスゴールの重要性
- 評判へのダメージ
- 事業収入の損失
- 財政的、環境的、または社会的な損失、もしくは賠償責任の可能性
- 民事または刑事上の法的制裁
- インターフェースや統合の課題事項
- 適切な回避策の欠如
- 安全性のニーズ

テストマネージャーは、リスク可能性とリスク影響を組み合わせて、リスクレベルを決定する。

リスク分析が、広範かつ統計的に有効なリスクデータに基づいている場合 は、定量的アセスメントが適切である。例えば、リスク可能性はパーセンテージで、リスク影響は金額で表すことができる。このような場合、リスクレベルはこれら 2 つの要因の積として計算できる。しかし、通常、リスク可能性とリスク影響は、例えば、非常に高い、高い、中程度、低い、または非常に低いというような順序尺度でのみ定性的にしか把握できない。リスク可能性とリスク影響の値は、リスクマトリクスで組み合わし、総合したリスクレベルが作成される。この総合したリスクレベルは、順序尺度による定性的な相対評価として解釈すべきである。

リスク分析が、広範かつ統計的に有効なリスクデータに基づいていない限り、リスク分析は、リスク可能性およびリスク影響に関するステークホルダーの主観的認識に基づく定性的なものとなる。

1.3.4 適切なテストによる品質リスク軽減

ソフトウェア開発において、テストは最も重要な品質リスク軽減活動であり、故障の可能性を軽減することができる。その他のリスク軽減の手段としては、コンティンジェンシープラン(回避策の提供など)、サードパーティ(コンポーネントのベンダーなど)へのリスク移転、リスク受容などが考えられる。

テスト計画において、テストの開発と実行に関連する時間と労力は、リスクレベルに比例すべきである。高いリスクレベルに対するテストは、早期に開始し、より厳密なテスト技法を用いるべきであるが、低いリスクレベルに対するテストは、後半に開始する場合があってよく、厳密ではないテスト技法を用いるべきである。テストを通じて全体的なリ

スクを最もよく軽減するために、テストマネージャーは、次のようなコンテキストに基づく要因を分析し、適切なテストアプローチを選択すべきである:

- **テストアイテム:** テスト対象内のテストアイテムが異なれば、同じリスクタイプでもレベルが異なる可能性があるため、テスト対象は一律な厳格さでテストする必要はない。
- **品質特性:** 特定の品質特性に影響するリスクは、特定のテスト労力、テスト環境、テストスキルを必要とする関連するテストタイプによって軽減されるべきである。
- **テストレベルとテストタイプ:** ある種のリスクは、特定のテストレベルで動的にテストするだけでよい。他のリスクは、静的テスト（例えば、保守性のための静的解析とコードレビュー）、またはその両方の組み合わせ（例えば、アーキテクチャのレビューとセキュリティの脆弱性のための統合したシステムの動的テスト）によってテストを行う。可能な限り早期にすべてのテストアイテムのテストを行うことで、ライフサイクルの後期に重大な欠陥が発見され、内部失敗コストや遅延が増大するリスクを軽減することができる。
- **SDLC:** テスト活動には、それぞれ固有の開始基準がある。異なる SDLC においては、開始基準を満たす時期もさまざまである。
- **テストチーム:** 最もリスクレベルの高いテストアイテムは、最も適任な担当者がテストすべきである。
- **規制要件:** 一部の安全関連規格（IEC 61508 規格など）では、インテグリティレベルに基づいてテスト技法と必須のカバレッジを規定している。テストマネージャーは、これらの規格が守られていることを保証しなければならない。

さらに、リスクレベルは、テストケースのような作業成果物のレビューの利用、テストの開発からの独立性レベル、リグレッションテストの実施範囲などの品質コントロールの判定に影響を与えるべきである。

テストモニタリングとテストコントロールにおいて、リスクベースドテストは任意の時点における残存リスクレベルに関するテスト進捗の報告を可能にする。これにより、開発チームとステークホルダーが、残存リスクレベルに基づいて、リリースの判定を含むソフトウェア開発のモニタリングとコントロールを行うことを支援する。そのためには、ステークホルダーが理解できる方法で、テスト結果をリスクの観点から報告することが必要である。

テスト実装において、テストの優先順位付けはリスクレベルに基づいて行われる。これにより、テスト実行の間に最も重要な領域の早期カバレッジと、最も高いレベルのリスクの軽減が確かなものになる。

- 場合によっては、テスト実行の優先順位を、カバーするリスクレベルの高いものから厳格に降順とする。このアプローチは縦型探索(depth-first)と呼び、最も高いレベルのリスクを可能な限り早期に軽減することが重要な場合に適している。
- あるいは、各リスクに対して少なくとも 1 つのテストが最優先として割り当てられる。その他のテストは、カバーするリスクのレベルに基づいて優先順位を付ける。このアプローチは横型探索(breadth-first)と呼び、ステークホルダーがプロダクト品質の全体像をできるだけ早い段階で把握したい場合に適している。実際には、テストは縦型探索アプローチから始めることが多いが、時間が限られてくると、残りのリスクアイテムをすべて少なくとも一度はテストする横型探索アプローチに切り替える。

リスクベースドテストが、縦型探索、横型探索、あるいはその組み合わせのいずれで進められるにせよ、テストに割り当てられた時間は、計画したテストがすべて実行できないまま消費してしまうことがある。リスクベースドテストは、この時点でテストを延長するか、残ったリスクを受け入れるかについて、正当な提案をマネジメントに提供することを可能にする。

1.3.5 リスクベースドテストの技法

リスクベースドテストを実装するための特定の技法には、さまざまな形式がある。技法の適否は、プロジェクト、プロセス、プロダクトの考慮事項に依存する。技法には、基本的に重量的と軽量的の 2 種類がある。セーフティクリティカルなシステムでは、重量的な技法を使うことが多い。セーフティクリティカルではないアプリケーションでは、通常、軽量的な技法を採用する。

重量的な技法は形式的であり、定義された手順と詳細な文書を用いる。これらの手法には、幅広いステークホルダーグループが関与する。重量的な技法におけるリスクアセスメントは、リスク可能性とリスク影響に関する詳細な要因を用いる。そして、それらの要因からリスク可能性とリスク影響を計算する数式を用いる。重量的な技法の例としては、以下のようなものがある:

- **ハザード分析:** 各リスクの根底にあるハザードを識別することにより、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:** 各品質リスクアイテムについて、故障の可能性、典型的な故障に関連する損失のコスト、そのような故障のために行うテストのコストを決定する。
- **故障モード影響解析(FMEA)とその派生:** 品質リスク、その潜在的な原因、および影響する度合いを識別し、重要度、優先度、および検出率を割り当てる。
- **フォールトツリー解析:** 潜在的な故障を、故障の原因となる欠陥に関連付ける。次にその欠陥の原因となるエラーに関連付け、さまざまな根本原因を識別するまで続ける。

対照的に、軽量的な技法は徹底度が低く、テストチームとステークホルダーの労力も少なくて済む。重量的な技法と同様、ステークホルダーの関与に基づいており、リスク分析の結果をテスト計画のベースとして使用する。しかし、ステークホルダーのグループはそれほど広くないかも知れず、リスク要因は通常、順序尺度に基づいてリスク影響とリスク可能性で整理する。体系的ソフトウェアテスト(SST)(Craig & Jaskiel, 2002)のように要件仕様が提供されている場合にだけ使うテスト技法もある。その他の技法としては、プラグマティックリスク分析マネジメント(PRAM)(Black, 2009)やプロダクトリスクマネジメント(PRISMA)(van Veenendaal, 2012)などがあり、これらはリスク分析の入力として要件やその他の仕様を使うか、もしくはステークホルダーからの入力に基づいて完全に機能することもできる。

1.3.6 リスクベースドテストに関連する成功メトリクスと困難さ

ふりかえりでは、テストチームは、リスクベースドテストの利点をどの程度実現できたかを測定すべきである。多くの場合、これには、メトリクスの利用やコンサルティングを通じて、以下の質問の一部またはすべてに答えることを含む:

- 関連するステークホルダーはリスク分析に関与したか、または意見を反映したか？
- リスク分析へのステークホルダーの関与は適切だったか？
- 深刻な欠陥の見逃しで深刻なインシデントが本番で発生した場合、それらは解決されたか？
- 優先度の高い欠陥のほとんどは、テスト実行の早い段階で発見されたか？
- テストチームはテスト結果をステークホルダーにリスクの観点から説明できたか？
- スキップしたテストは、実行したテストよりも関連するリスクレベルが低かったか？

ほとんどの場合、リスクベースドテストが成功すれば、これらの質問すべてに対して肯定的な回答が得られる。長期的には、成功メトリクスに対するプロセス改善ゴールを設定し、品質リスク分析プロセスの効率を向上させる努力を続けるべきである。

リスクマネジメントは、見過ごされがちな複雑さのために、しばしば予期せぬ困難に遭遇する。

- **リスクレベルのアセスメントが困難:** リスク影響とリスク可能性を見積ることは非常に困難である。解決策:過去のデータを使用し、主要なプロジェクトステークホルダーにアセスメントを求める。
- **適切な開始:** 適切なリスクベースのテストアプローチを設定し維持することは、短期的な成功への高いプレッシャーの中でしばしば軽視される。解決策:リスクの定期的なモニタリングとステークホルダーへの報告。
- **デジャブ:** プロジェクトごとに同じようなリスクが提起されるため、リスクに対する油断が生じている。解決策:リスク識別に適切な人材を参加させ、重要なリスクのみを軽減する。
- **重要なリスクの見逃し:** この課題の根本原因は、通常、経験の浅い人や不適切な人がプロセスに関与していることにある。解決策:適切な人材を関与させ、トレーニングする。
- **ステークホルダーの変化:** ステークホルダーは時間とともに変わる可能性があり、新たなリスクが発生することもある。そのため、リスク分析は継続的でイテレーティブな活動となる。初めに一度だけ実施するものではない。

1.4 プロジェクトテスト戦略

イントロダクション

本シラバスでは、組織的テスト戦略は与えられたものとして扱う。組織的テスト戦略の開発とメンテナンスは、ISO/IEC/IEEE 29119-3（「組織的テストプラクティス」と呼ばれる）、および ISTQB® Expert Level - Test Management、ISTQB® Certified Tester Agile Test Leadership at Scale のシラバスのコンテキストで検討する。

組織的テスト戦略が存在しない場合や、必要とされる側面をカバーしていない場合、テストマネジメントは、関連するステークホルダーとともに不足している詳細を明確にするよう努めなければならない。

本節のコンテキストでは、プロジェクトテスト戦略の定義は、プロジェクト、リリース、プロダクト、またはその他のタイプのシステム開発または導入を推進する取り組みのためのあらゆるタイプの詳細なテスト戦略の例である。プロジェクトテスト戦略（ISO/IEC/IEEE 29119-3 では「テスト戦略」と呼ばれる）は、組織の目的、特にプロダクト品質とテスト活動に関連する目的を達成できるように、特定のコンテキストにおけるテストへのアプローチを記述する。テスト戦略は、単一のテストレベルやテストタイプに対して存在することもある。

プロジェクトテスト戦略は、テスト計画の主要な成果物であり、通常、テスト計画書または他の文書の一部として文書化する。テスト戦略の文書化は推奨されるが、必ずしも形式的なテスト計画書の形である必要はない。文書化の必要性は、テストのコンテキストに依存する（1.2 節「テストのコンテキスト」参照）。プロジェクトがシーケンシャル開発モデルにしたがっている場合、プロジェクトテスト戦略は、通常文書化し、できればテスト計画書（ISO/IEC/IEEE 29119-3 参照）の中に文書化する。また、契約書、同意書、規制機関、または法律により、文書化が要求されることも多い。

1.4.1 テストアプローチの選択

プロジェクトテスト戦略は、プロジェクト内のすべてのテスト活動に対する指針であり、目的、リソース、スケジュール、および責任を詳細に記述する。この戦略は、プロジェクト特有の要件に合わせて調整しなければならない。重要な決定事項には、静的テスト、動的テスト、その他のテスト実践（例えば、手動テスト、バックツーバックテスト）、テストレベル、テストタイプ、テスト技法の選択を含む。

理論的には、すべてのテストタイプはどのテストレベルでも実施することができ、どのテスト技法もどのテストレベルのどのテストタイプにも適用することができる。実際には、これらの適切な選択と組み合わせが、テストの有効性と効率性に大きな影響を与える。例えば、コードの保守性は、静的コード解析やコードレビューを用いることで、より効果的かつ効率的に評価できることが多い。一方、性能効率性は、内部コンポーネントの相互作用により、スクリプト化したシステムテストで評価した方がよい場合もあるし、機能の有用性は、共同で開発した手動受け入れテストでユーザーと確認した方がよい場合もある。テスト戦略に最適なアプローチを選択することは、組織的テスト戦略、プロジェクトコンテキスト、その他の側面に影響を受ける複雑なプロセスになり得る。

テストレベル、テストタイプ、テスト技法を選択し、組み合わせることは、テストの効率性と有効性に大きく影響するため、効果的なプロジェクトテスト戦略にとって非常に重要である。

1.4.2 組織的テスト戦略、プロジェクトコンテキスト、その他の側面の分析

プロジェクトテスト戦略を開発するためには、組織的テスト戦略、プロジェクトコンテキスト、およびテストに関連する付加的な要因や制約を十分に理解しなければならない。

適切なテストアプローチを選択するためには、通常以下の要因を分析しなければならない:

- **ドメイン:** プロダクトが作成または修正されるドメイン。ドメイン固有の規制、標準、慣行が、テストの厳密さや求められる文書化、およびその詳細レベルを変える可能性がある。例えば、製薬や医療では、テストアプローチは、機能的なユーザー要求に基づいたテストケースを用いて、患者の健康に対するリスクに焦点を当てた集中的なユーザー受け入れテストを重視することが多いが、ウェブベースの保険アプリケーションのユーザー受け入れテストは、A/B テストを通じて、使用性と新規保険契約の可能性を高めることに焦点を当てるかもしれない。
- **組織のゴールと包括的な品質の特徴:** 組織のゴールには、テストの価値の説明の必要性、テスト自動化の度合いを高める必要性、テストの成熟度や欠陥検出の効率性などテストプロセスの品質の特徴を含む。これらによって、適応すべきテストレベルやテストタイプを決定することがある。
- **プロジェクトのゴールとプロジェクトのタイプ:** プロジェクトのゴール(例えば、予算、時間、品質など)とプロジェクトのタイプ(例えば、顧客志向のプロダクト開発か、市場志向のプロダクト開発か)には、通常テストに影響する制約とリスク、および機会を含む。例えば、予算と時間の制約が厳しい場合、テスト実行のためのテストケースの優先順位を決めるために、リスクベースドテストを厳格に使用する必要があるかもしれない。一方、顧客向けに特化したプロダクト開発では、事前に定義された契約による受け入れ基準をカバーするテストが必要になることがある。
- **テストリソース:** プロジェクトで使用するテストツール、テストインフラ、技術、開発環境、また利用可能なテスト要員とそのスキルなど、テストリソースの可用性に関するあらゆる制約を考慮しなければならない。(3.1 節「テストチーム」参照)。例えば、経験ベースのテストには、十分なドメイン知識を持つテスト担当者が必要であり、モバイルアプリケーションは、通常、限られた数の異なるデバイスでテストする必要がある。また、テストツールの使用は、利用可能なライセンス数によって制限されるかもしれない。
- **プロジェクトに使用するソフトウェア開発ライフサイクルモデル:** 適切なテストレベル、テストの労力、適切な開始基準および終了基準を決定するために、ISTQB® Foundation Level Syllabus V.4 の 2.2 節および 5.1 節を参照すること。継続的インテグレーションを伴うソフトウェアライフサイクルでは、ウォーターフォールモデルによる一度限りの開発よりも多くの自動テストが要求されるため、異なるテストタイプとテスト技法を使用する場合がある。
- **他のシステムとのインターフェース:** システムオブシステムズでは、テストを他のチームやプロジェクトと調整し、特にシステム統合テストのために適切なテストレベルを選択することが不可欠である。例えば、リスクベースドテストは、システム統合テストの優先順位付けと規模調整に役立つ。
- **テストデータの可用性:** テストデータの可用性に関する制約、例えば、本番環境から匿名化されたテストデータが必要である場合や、AI テスト用のデータのように、提供が困難で有効性が必要な特定のテストデータを作成する必要があることを考慮しなければならない。例えば、モデルベースドテストは、テストデータの作成とテストデータのマネジメントをサポートできる。

テストマネージャーは、組織的テスト戦略、プロジェクトコンテキスト、テストに関連するその他の要因や制約を満たす最適なアプローチとして、テスト技法、テストレベル、テストタイプの組み合わせを決定しなければならない。

1.4.3 テスト目的の定義

テスト計画書は、テストプロジェクトごとに定義され、特にテストスコープ、テスト目的、終了基準をその他の側面とともに含むべきである。テスト計画書は、リリースレベル、プロジェクトテスト計画書(「マスターテスト計画書」とも呼ばれる)、そして必要であれば、それぞれのテストレベルのレベルテスト計画書として作成することができる。さらに、セキュリティテスト計画書や性能テスト計画書のような、さまざまな品質特性のためのテスト計画書を定義することができる。アジャイルソフトウェア開発やハイブリッドソフトウェア開発プロジェクトでは、イテレーションテスト計画書を合意することができる。各リリースとイテレーションについて、提供すべき機能性のフィーチャーのスコープとその非機能特性の範囲は、テスト計画書で定義され、ステークホルダーによって合意される。

プロジェクトでテストへリリースするフィーチャーに関連して、プロジェクトのテスト目的と終了基準を定義しなければならない。これは、スマートの法則を使って行うことができる:

- S＝具体的な。プロジェクトのテスト目的と終了基準は、明確あるべきで曖昧ではないものであるべきである。
- M＝測定可能な。数値化可能で、達成状況を判断するための具体的な進捗基準を持っていなければならない。
- A＝達成可能な。リソースの可用性、時間枠、能力を考慮し、実現可能でなければならない。
- R＝妥当な。プロジェクト全体の目的に沿ったものでなければならない。
- T＝タイムリーな。具体的な時間枠と完了期限を定めるべきである。

プロジェクトのテスト目的は、測定可能または評価可能であれば、品質と量のすべての対象とする側面に対応すべきである。プロジェクトテスト目的の例を以下に示す:

- 定められた期間内に、指定された終了基準に達すること。
- 組織の品質ゴールを達成する(例えば、プロダクトに対する顧客からのクレーム件数を KPI として測定する)。
- 特定の業界のルールや規制を遵守する。
- 承認されたユーザーのみ、データの可用性を確保する(例:アクセス権による)。
- データ移行の機能完全性、機能正確性、性能、効率性、移植性、セキュリティを確認する。
- テスト自動化のレベルを向上させる(例えば、リグレッションテストや性能テストにおいて、定義されたパーセンテージで)。
- コードのリファクタリングに成功し、新たな欠陥が作りこまれていないことを示す(例えば、既存の機能性を維持しながら、構造の悪いソースコードや技術的負債を取り除き、リグレッションテストで証明する)。
- インターフェースのセキュリティを証明する(例えば、XML スキーマ定義に照らし合わせて XML(Extensible Markup Language)メッセージを確認し、悪意のあるデータの拒否を保証する)。

- ユーザーインターフェースの使用性を確認し、ある程度の副特性を達成する（例えば、オンラインショップで特定のタスクを完了するのにかかる時間を測定する）。

プロジェクトのテスト目的の集計や測定とは別に、業界の専門家やステークホルダーによる品質レベルのアセスメントも考慮すべきである。

プロジェクトコンテキストやテスト目的によっては、利用可能なリソースおよび/またはテストツールを備えた複数のテスト環境が必要となる場合がある。テスト環境はすべて同時に利用できるとは限らない。達成可能なテスト目的と終了基準を策定する際には、この点を考慮する必要がある。

プロジェクトコンテキストによっては、本シラバスの 1.2 節「テストのコンテキスト」に記載されているように、プロジェクトのテスト目的とテストスコープを定義する際に、さらなる要因を考慮する必要がある。

1.5 テストプロセス改善

イントロダクション

テストは、ソフトウェア開発において重要な部分であり、プロジェクト全体のコストの少なくとも 30～40%を占めることが多い。ソフトウェアプロジェクトが直面している多くの（技術的な）課題（複雑度や規模の増大、新しい技術、多種多様なデバイスやオペレーティングシステム、セキュリティの脆弱性など）に加えて、テストの有効性と効率性を最適化する必要性と、それに応じてテストプロセスを改善する必要性がある。既存の最良の実践例や自身のエラーから学ぶことで、テストプロセスを改善し、プロジェクトをより成功に導くことが可能になる。

組織レベルでの改善プロセスは、通常、プロジェクトやチームレベルでの改善プロセスよりも有用である。しかしながら、プロジェクトやチームレベルでプロセス改善を適用することも可能であり、有益である。テスト改善は、例えば、現在のテスト結果に対する不満、予期せぬ欠陥、状況の変化、ベンチマーク結果、コミュニケーション不足などによって開始される。テストを改善するために、さまざまな技法が利用可能である (Bath & van Veenendaal, 2014)。これらの技法のいくつかを以下に説明する。本シラバスで説明する技法は、シーケンシャル開発モデルとアジャイルソフトウェア開発/インクリメンタル開発モデルの両方に適用できる。ISTQB® Expert Level Improving the Test Process シラバスでは、より深い洞察が得られる。

1.5.1 テスト改善プロセス(IDEAL)

テストプロセスを改善することが合意されたら、この活動のために採用されるべきプロセス改善実装活動は、IDEAL モデルのように定義することができる。これは、よく知られた PDCA サイクルと同様の考えに基づいている。IDEAL とは、Initiating（開始）、Diagnosing（診断）、Establishing（確立）、Acting（行動）、Learning（学習）の頭文字を取ったものである。

IDEAL はもともと組織レベルでの改善活動を支援するために定義されたが、プロジェクトやアジャイルソフトウェア開発チームレベルでも適用できる。プロジェクトのコンテキストでは、活動の目的（以下参照）はまだ達成されていない。主な違いは、おそらく開始フェーズであり、プロジェクトまたはチームレベルでは、組織レベルよりもはるかに小さい。ふりかえりによる診断と計画の確立は、組織レベルよりもはるかに小さい可能性が高い。行動すること、そして/または学習することも、プロジェクトやチームレベルではすべて関連してくる。

改善プロセスの開始

改善プロセスの開始時に、プロセス改善の目的とスコープをステークホルダーと合意する。

現状の診断

現在のテストプロセスをアセスメントし、改善の可能性を特定する。このアセスメントは、モデルベースのテストプロセス改善の場合、通常、標準的なフレームワークに照らして行われる（1.5.2 項 モデルベースのテストプロセス改善を参照）、または特定のメトリクスの分析に基づくことができる。（1.5.3 項 分析ベースのテストプロセス改善を参照）。

テストプロセス改善計画の確立

テストプロセス改善計画書は、改善を達成するために実行しなければならない詳細なアクションをすべて列挙した、形式的な文書となる。コンテキストによっては、計画は非常に非形式的で非常に軽量的なものになる。可能な

プロセス改善のリストには、優先順位を付けるべきである。優先順位付けは、投資効果(ROI)、リスク、プロジェクトまたはチームの戦略との整合性、および/または達成すべき測定可能な定量的または定性的な利益に基づいて行うことができる。

テストプロセス改善を実装するための行動

改善を展開するためのテストプロセス改善計画書を実装する。これには通常、変更されたプロセスのトレーニングとパイロット実施、およびプロジェクトまたはチームへの完全な展開が含まれる。

改善プログラムからの学習

プロセス改善を完全に展開した後は、想定されていたものであれ、想定外のものであれ、どのような利益が得られたかを検証することが不可欠である。何がうまくいき、何がうまくいかなかったかを学んだ後、この情報に基づいて行動しなければならない。

1.5.2 モデルベースのテストプロセス改善

モデルベースのテストプロセス改善も、分析ベースのテストプロセス改善も、その前提の 1 つは、プロダクトの品質は使用され適用されるプロセスの品質に大きく影響を受けるという仮定である。モデルベースのテストプロセス改善を適用する場合、テスト改善モデルを使用する。テスト改善モデルは、テストの最良の実践例に基づいており、テスト改善を段階的に進める。

テストプロセス改善を支援する推奨プロセスモデルがいくつか登場している。これには、テスト成熟度モデル統合(TMMi ®)や TPI NEXT®を含む。

モデルベースの改善は、プロジェクトレベルで適用することもできる。このような場合、アセスメントと改善プロセスは、プロジェクトレベルの活動（テスト計画やテスト設計など）に関連するテストプロセスや、モデルで定義された重要な領域に特に焦点を当て、組織レベルの活動（テストポリシーやテスト組織など）は大きく除外されることが多い。あるいは、組織レベルに対応するプラクティスを、プロジェクトのコンテキストに合わせて適切に調整することもできる。

モデルベースのテストプロセス改善の詳細については、ISTQB® Expert Level Improving the Test Process Syllabus を参照のこと。

テスト成熟度モデル統合

TMMi®(van Veenendaal & Cannegieter, 2011)(van Veenendaal, 2020)は、5 つの成熟度レベルで構成されている。TMMi® レベル 1 を除く各成熟度レベルには、テストプロセス領域と改善ゴールが含まれている。さらに、TMMi® を実施、促進、支援するために、プラクティス、サブプラクティス、および事例が含まれている。TMMi® は当初、能力成熟度モデル統合(CMMI®)を補完するために開発されたが、今日では CMMI® とは独立して広く使用されている。

アジャイルソフトウェア開発における TMMi® の更新を容易にし、サポートするために、TMMi® がアジャイルソフトウェア開発においてどのように使用され、有益に適用されるかを説明する特定のガイドラインが開発された。

TMMi® についての詳細は www.tmmi.org を参照のこと。

TPINEXT®

TPINEXT®モデル (van Ewijk, 2013)は 16 のキーエリアを定義しており、それぞれがテストプロセスの特定の側面（テスト戦略、テストメトリクス、テストツール、テスト環境など）をカバーしている。このモデルでは、16 のキーエリアのそれぞれについて、4 つの成熟度レベルが定義されている。

各成熟度レベルにおける各キーエリアをアセスメントするために、特定のチェックポイントが定義されている。アセスメント結果は、すべてのキーエリアをカバーする成熟度マトリクスによってまとめられ、可視化される。

TPI NEXT の詳細については®www.tmap.net を参照のこと。

1.5.3 分析ベースのテストプロセス改善アプローチ

前節で説明したように、モデルベースの改善アプローチを用いると、プロジェクトやチームのテストアプローチを外部の最良の実践例と比較することで改善がもたらされる。分析的アプローチでは、プロジェクトやチーム自体のデータに基づいて問題を識別する。これらの問題の分析から、適切な改善策を導き出すことができる。分析的アプローチは、結果を検証し、多様性を提供するために、モデルベースのアプローチとともに使用することができる。

問題は、定量的データと定性的データを用いて識別することができる。本シラバスの 1.5.3 項では、主にテストプロセスの定量データと欠陥のデータを用いて、現在のアプローチをアセスメントする分析アプローチを紹介する。本シラバスの 1.5.4 項では、何がうまくいき、何がうまくいかないかに関する定性的データを、開発チームとテストチームのメンバーから収集するふりかえりを紹介する。

データ分析は、客観的なテストプロセス改善のために重要であり、純粋に定性的なアセスメントを行う際の貴重なサポートとなる。分析的アプローチを改善に適用する場合、多くの場合、テストプロセスを定量的に分析し、問題のある領域を特定し、プロジェクト固有のゴールを設定する。テストプロセスをアセスメントし、改善が成功したかどうかを評価するためには、主要なパラメーターの定義と測定が必要となる。

分析的アプローチの例としては、以下のようなものがある：

- 根本原因分析
- 尺度、メトリクス、指標を用いた分析
- GQM(Goal-Question-Metric)アプローチ

根本原因分析とは、問題の根本原因を特定するための調査である。これにより、単に目先の明らかな症状に対処するのではなく、問題の原因を取り除く解決策を識別することができる。考えられる分析手順としては、欠陥の適切なセットを選択し、このデータからクラスターを識別し、特性要因図（石川ダイアグラムまたは魚の骨ダイアグラムとも呼ばれる）を使用して、重要な欠陥クラスターの根本原因を識別する。そして、同様の欠陥の発生を防止するための改善策を導き出す。

尺度、メトリクス、指標は、プロジェクトやチームにおけるテストプロセスがどの程度うまく実行されているかをアセスメントするための定量的な方法として使用する。考慮すべきテストプロセスの主な属性は、有効性、効率性、予測性である。これらの属性ごとに、1 つまたは複数のメトリクスを選択することができる。対応するデータを収集、分析することで、改善が必要な主要領域を識別できる。

GQM アプローチ (Basili, et al., 2014)(van Solingen & Berghout, 1999)は、関連するプロジェクトステークホルダーの情報ニーズに合わせたメトリクスを定義し、分析するためのフレームワークを提供する。測定ゴールは、特定の目的、視点、コンテキストのために測定する必要がある対象物の品質側面を定義する。これらのゴールは、ステークホルダーの視点から品質の側面を定義する質問へ洗練する。そして、質問に答えるために必要な情報を提供

するメトリクスを選択する。メトリクスのために収集されたデータは、測定ゴールをアセスメントし、ステークホルダーの情報ニーズを満たすための質問の答えとなる。

これらの分析に基づくテストプロセス改善アプローチについてのさらなる情報は、ISTQB® Expert Level Improving the Test Process Syllabus で確認ができる。

1.5.4 ふりかえり

ふりかえりとは、チームが自分たちのやり方とコラボレーションをレビューし、学んだ教訓（よい点 と悪い点の両方）を把握し、（テストとテスト以外の課題の両方について）改善を達成するための変更と 行動を決定するためのミーティングである。ふりかえりでは、プロセス、人、組織、コラボレーション、ツールなどのトピックを取り上げる。

ふりかえりは、シーケンシャル開発モデルとアジャイルソフトウェア開発の両方で使用される。シーケンシャル開発モデルでは、テスト完了の一部である。このコンテキストでは、ふりかえりは、将来のプロジェクトをよりよくマネジメントするために、教訓を生み出すことを目的としている。アジャイルソフトウェア開発では、ふりかえりは一般的に各イテレーションの終わりに行われ、何が成功し、何を改善する必要があるか、そしてそれらの改善を次のイテレーションにどのように組み込むことができるかについて議論する。ふりかえりはチーム全体で行うため、チーム全体アプローチを支援し、継続的な改善を促進する。テストの課題に対処するために、専用のふりかえりが必要となることがある。

典型的なふりかえりは、以下のステップで構成される：

イントロダクション：ふりかえりのゴールと課題を確認する。そして非難や判断を下すことなく問題を話し合うことができるよう、相互信頼の雰囲気を作る。

データの収集：データは、イテレーションまたはプロジェクト中に起こったことに関して収集する。課題事項を特定し、各チームメンバーがそれらの課題事項についてどのように感じているかを列挙した、主要な出来事のタイムラインといった定性的なデータを収集することが可能である。さらに、メトリクスの定量的データを提示することもできる。例えば、テスト進捗、欠陥検出、テスト有効性、テスト効率、予測性といったデータは、プロジェクトやイテレーションのテストに対する客観的な洞察を提供することができる。

改善策の導出：収集したデータを分析して現状を把握し、改善のアイデアを生み出す。例えば、根本原因分析を適用して、特定された問題の根本原因を特定し、その根本原因を解決するためのアイデアを生み出すために、ブレーンストーミングセッションを開催することができる。

改善アクションの決定：改善案を実施するためのアクションを導き出し、優先順位を付ける。改善計画と責任者を定義する。識別した問題に対するアクションの影響を評価するために、ゴールと関連メトリクスを定義することができる。一度に多くの改善を実施することは、検証可能なステップでマネジメントすることが困難となる。

ふりかえりのクロージング：この最後のステップでは、ふりかえりプロセスの長所と改善点を識別するために、ふりかえり自体をレビューする。ふりかえりは、特にアジャイルソフトウェア開発では定期的に行われる。継続的な改善は、ふりかえり自体にも適用される。

ふりかえりの結果を適切に文書化することが重要である。シーケンシャル開発モデルでは、発見事項、結論、および推奨事項を、組織のメンバーに分かりやすい方法で配布し、伝達する必要がある。アジャイルソフトウェア開発では、次のイテレーションで問題とその潜在的な影響をレビューできるように、問題とアクションも文書化する必要がある。

テスト担当者は、(プロジェクト)チームの一員となり、独自の視点を持ち込むことができる。テストに関連する問題(およびその他の問題)を提起し、可能な改善策を考えるようチームを刺激することができる。

さらに詳しい情報については、(Derby & Larsen, 2006)で確認できる。

1.6 テストツール

イントロダクション

ビジネスツールには 3 つのタイプがある:

- 商用ツール
- オープンソースツール
- カスタムツール

ビジネスツールを選択する際には、組織やステークホルダーの要件や規制をすべて考慮しなければならない。

また、テスト自動化ツールやテストマネジメントツールなどの技術的なツールも存在する。

テストツールの例は、ISTQB® Foundation Level Syllabus V.4 にて確認できる。

1.6.1 ツール導入のためのよい実践

本項は、テストツールの評価と導入に必要なステップからなる。

テストマネージャーは、ツールの導入に関与することもあれば、導入プロセスを促進することもある。テストマネージャーは、通常、専用のテストツール、あるいは、要件マネジメント、欠陥マネジメント、モニタリングツールなどのテストに関連するツールを担当する。

テストツールを評価、選択する際には、一般的なよい実践例と考慮事項がある。これら実践例と考慮事項は以下を含む:

- 適切なツールのサポートを受けながら、プロセス改善の機会を識別する。
- 組織で使用されている技術を理解し、これらの技術と互換性のあるツールを選択する。
- ツールが技術的および組織的に SDLC にどのように統合されているかを理解する。
- 明確な要件と客観的な基準に照らしてツールを評価する。
- 商用ツールの使用を検討している場合は、ベンダーを評価する。非商用(オープンソースなど)ツールのサポートを評価する。
- ツールの使用に関するコーチング、指導、またはトレーニングの社内要件を識別する。
- さまざまなライセンスモデルの長所と短所を検討する
- 最終段階として、コンセプトの証明を評価する。

ツールの採用と展開における一般的なよい実践例には、以下のようなものがある:

- パイロットプロジェクトを実施し、選択基準と要件を妥当性確認し、ツールが既存のプロセスや実践にどのように適合するかを評価する。
- 必要であれば、既存のプロセスにもツールを適応させる。

- ツール使用のガイドラインを定める
- ツールユーザーのトレーニング、コーチング、指導を行う
- ツールを段階的に組織に展開する
- さらなる改善のために、ツールの実際の使用から情報を収集する方法を実装する。
- ツールのオーナーを定義する

1.6.2 ツール選定のための技術的側面とビジネス的側面

ツールの導入と使用に関する判定には、複数の要因が影響する。テストマネージャーにとって、それらを知り、対処することは重要である。

- **規制とセキュリティ**: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ソフトウェアを開発する組織、あるいは法令遵守の対象となる組織は、商用ツールの方が要件に適合していることが多く、適切な認定を取得していることが多いため、商用ツールを好むかもしれない。
- **経済的側面**:オープンソースツールは、コミュニティによるサポートや開発があるため、通常、初期費用が安い。商用ツールは、1 回限りの購入価格と定期的なライセンス費用がかかる場合がある。カスタムツールの初期費用は、要件とツールの開発段階に依存するため、決定するのは難しい。初期費用に加えて、ツールの耐用年数にわたるトレーニングとメンテナンスの費用も計算し、考慮しなければならない。どのツールも、メンテナンスとサポートに高いコストがかかる可能性がある。
- **ステークホルダーの要件**:最適なツールを評価し特定するためには、すべてのステークホルダーから要件を収集することが重要である。商用ツールやオープンソースツールは、必ずしもすべての要件を詳細に満たすとは限らない。カスタムツールは、個々の要件をすべて満たし、他のツールでは必要な機能が得られない場合に、最良の選択となり得る。
- **既存のソフトウェアランドスケープとツール戦略**:既存のツール構成(ソフトウェアランドスケープ)および関連するツール戦略は評価する必要がある。なぜなら、これらには、優先するベンダーがある場合、他のベンダーに変更しづらい場合がある他、他のプロダクトとの依存関係がある統合システム、または特定の規制があるソフトウェアランドスケープ全体に対する特別なフルサービスサポートモデルが存在する場合があるためである。

1.6.3 選定プロセスの考慮事項と ROI(投資効果)評価

テストツールは長期的な投資となる可能性があり、1 つのプロジェクトで何度も繰り返されることになるかもしれないし、そして/または多くのプロジェクトに適用できるかもしれない。したがって、将来性のあるツールは、さまざまな観点から検討する必要がある。

- 経営幹部にとっては、プラスの ROI が要件となる。
- サポートおよび運用チームにとっては、組織全体で使用されるツールの数は限られているが、必要な数であることが望ましい。より多くのツールを保守し、ライセンスをマネジメントし、ツールの数をマネジメントすることは、コストや時間のかかることであってはならない。

- プロジェクトリーダーにとって、そのツールはプロジェクトや組織に測定可能な価値を与えるものでなければならない。
- ツールを使う人にとって、使用性は非常に重要である。使用性とは、例えば、与えられた作業への対応、習得性、運用性などである。
- 運用スタッフにとって保守性は重要である。

ツールの機能は、ビジネスと技術タイプごとに分析しなければならない。テストマネジメント、(技術的な)テスト分析、テスト自動化、または開発など、さまざまな視点や関心がその分析に影響を与える。ツールの責任者(ツールオーナー)は、分析が達成され、上記の箇条書きが考慮されていることを確認しなければならない。

テストプロセスに導入されるすべてのツールは、組織にとってプラスの ROI を保証するものでなければならない。ROI の計算とさらなる評価を行うのはテストマネージャーの責任である。アジャイルソフトウェア開発では、それは開発チーム全体の責任かもしれない。

ツールの取得または構築の前に、それが組織にとって有益であることを確実にするために、費用対効果分析を行うべきである。この分析では、経常的なコストと非経常的なコストの両方を考慮に入れるべきである。

非経常的な活動およびコストには以下のものを含む:

- 目的を達成するためのツール要件の定義と決定
- 適切なツールとベンダーの評価と選択、コンセプトの証明
- 初期使用のためのツールの購入、適応、開発
- ツールの使用に関するガイドラインの定義
- ツールの初期トレーニング
- 既存のツールランドスケープへのツールの統合
- ツールをサポートするために必要なハードウェア/ソフトウェアの調達

経常的な活動やコストには以下のものを含む:

- 定期的なライセンス料とサポート料
- メンテナンスコスト
- 継続的なトレーニングコスト
- 異なる環境へのツールの移植

機会コストも考慮しなければならない。つまり、ツールの評価、管理、トレーニング、使用に費やされた時間は、代わりに実際のテストタスクに費やすことができた可能性がある。したがって、ツールを意図した活動に使用する前に、より多くのテストリソースが必要になる可能性がある。

ツールを選択する際には、ROI に関する以下のリスクを考慮すべきである:

- 組織の未熟さが、ツールの非効率的な使用につながる可能性がある。
- ベンダーのメンテナンス・ポリシーが変更されると、作業量が増える可能性がある。

- コストが予想を上回る
- 利益が予想を下回る

テストツールには次のような利点がある:

- 手作業による繰り返し作業の削減(リグレッションテストなど)
- 自動化によるテストサイクルのスピードアップ
- 手作業の減少によるテスト実行コストの削減
- ツールがサポートする特定のテストタイプのカバレッジ向上
- 手作業の減少によるヒューマンエラーの削減
- テストに関する情報への迅速なアクセス

特にテスト自動化ツールに関するその他の利点とリスクは、ISTQB® Foundation Level Syllabus version 4 と ISTQB® Test Automation Engineer Syllabus で確認できる。

一般的に、テスト組織が単一のツールを使用することはほとんどない。組織の総 ROI は通常、テストに使用されるすべてのツールの ROI の混合である。ツールは情報を共有し、協調して動作する必要がある。リスク、コスト、利益を含むテストツールの長期的かつ包括的な戦略が望ましい。

1.6.4 ツールのライフサイクル

ツールのライフサイクルには 4 つの異なるステージがある。ツールの管理者は、これらのステージの活動が定義され、実行され、マネジメントされることを保証するために任命されなければならない。

- **調達:**最初に、ツールを選定する判定が下される。次の段階では、ツールの所有者を割り当てる必要がある。この人物は、ツールの使用に関する決定を行う(例えば、作業成果物の命名規則や、これらの作業成果物の保管場所など)。前もってこれらの決定を行うことは、ツールの最終的な ROI に大きな違いをもたらす可能性がある。
- **サポートとメンテナンス:**ツールの所有者は、ツールのメンテナンスの責任を負う。メンテナンス活動の責任は、ツールの管理者または専用のツールグループにあるべきである。相互運用が行われるの場合、データ交換、および連携とコミュニケーションのプロセスを考慮しなければならない。また、ツールに関連する成果物のバックアップと復元に関する決定も要件となる。
- **進化:**時間の経過とともに、環境、ビジネスニーズ、またはベンダーによる決定により、ツールの変更が必要となることがある。ツールの運用環境が複雑化すればするほど、変更によってその利用が妨げられやすくなる。
- **廃棄:**ツールの有効期間が過ぎたら、そのツールは廃棄すべきである。ほとんどの場合、ツールによって提供される機能は置換され、データは保存および/または保管されなければならない。これはベンダーの決定に基づくか、または新しいツールに移行する利点と機会がコストとリスクを上回る段階に達したときである。

1.6.5 ツールメトリクス

ツールから得られる客観的なメトリクスは、テストチームやその他のステークホルダーのニーズに基づいて設計され、収集される。テストツールの多くは、貴重なリアルタイムデータを取得し、データ収集の労力を削減する。このデータは、テスト活動全体のマネジメントや最適化のための領域の特定に使用する。

異なるツールは、異なるタイプのデータを収集することに重点を置いている。例えば、以下のようなものがある:

- テストマネジメントツールは、利用できるテストアイテム、テストケース、実行を計画したテストケース、および現在と過去のテスト実行状況(合格、不合格、スキップ、ブロック、実行予定など)に関連するさまざまなメトリクスを提供する。
- 要件マネジメントツールは、合格および不合格のテストケースによる要件カバレッジに関するトレーサビリティを提供する。
- 欠陥マネジメントツールは、テストアイテムのステータス、重要度、優先度、欠陥密度などの欠陥情報を提供することができる。欠陥検出率、テストレベルで検出された欠陥の混入から検出までのリードタイム、その他の貴重なデータは、プロセス改善の推進に役立つが、欠陥マネジメントツールだけで提供できるとは限らない。
- 静的解析ツールは、コードの複雑度に関連するメトリクスを提供する。
- 性能テストツールは、負荷がピークの際の応答時間や故障率などの貴重な情報を提供する。
- コードカバレッジツールは、テスト対象のどの部分がテストによって実行されたかを理解するのに役立つ。
- テストツールは、メトリクスを収集するために使用できるが、テストツール自身をモニタリングすることも必要である。このコンテキストでは、テストプロセスの品質を測定することができる(例えば、ツールを使用した場合と使用しない場合の発見された欠陥の数や要件カバレッジ)。
- テスト効率(例えば、テスト実行時間や実行したテストの数など)

メトリクスの収集と使用に関する詳細は、本シラバスの 2.1 節「テストメトリクス」で確認できる。

2 プロダクトのマネジメント - 390 分

キーワード

不正、欠陥、欠陥レポート、欠陥ワークフロー、故障、メトリクス、テスト見積り、テスト目的、テスト進捗状況

ドメイン固有のキーワード

プランニングポーカー、三点見積り、ワイドバンドデルファイ

第 2 章の学習の目的

2.1 テストメトリクス

TM-2.1.1 (K2) テスト目的を達成するためのメトリクスの例を挙げる。

TM-2.1.2 (K2) テストメトリクスを用いてテストの進捗コントロールする方法を説明する。

TM-2.1.3 (K4)ステークホルダーの意思決定に役立つテストレポートを作成するためにテスト結果を分析する。

2.2 テスト見積り

TM-2.2.1 (K2) テスト見積りにおいて考慮すべき要因について説明する。

TM-2.2.2 (K2) テスト見積りに影響を与える要因の例を挙げる。

TM-2.2.3 (K4) 特定のコンテキストでテスト見積りのための適切な技法またはアプローチを選択する。

2.3 欠陥マネジメント

TM-2.3.1 (K3) 欠陥のモニタリングとコントロールに使用できる欠陥ワークフローを含め、欠陥マネジメントプロセスを実装する。

TM-2.3.2 (K2) 効果的な欠陥マネジメントに必要なプロセスと参加者を説明する。

TM-2.3.3 (K2) アジャイルソフトウェア開発における欠陥マネジメントについて詳細に説明する。

TM-2.3.4 (K2) ハイブリッドソフトウェア開発における欠陥マネジメントの課題について説明する。

TM-2.3.5 (K3) 欠陥マネジメントで収集すべきデータと分類情報を利用する。

TM-2.3.6 (K2) 欠陥レポートの統計情報がプロセス改善の考案にどのように利用できるか説明する。

2.1 テストメトリクス

イントロダクション - なぜテストメトリクスがあるのか？

マネジメントには「測定できるものは達成できる」という格言がある。同様に、測定されないものは、無視されやすいので、達成されることはないだろう。したがって、テストを含め、どのような取り組みに対しても、適切なメトリクスを確立することが重要である。

テスト目的は、なぜテストするのかの答えである（1.4 節「プロジェクトテスト戦略」参照）。テスト目的が達成されたかどうかを判断するには、それを測定する方法を定義しなければならない。テストメトリクスは、この問いに答えるための指標である。

テストメトリクスは以下のように分類できる：

- **プロジェクトメトリクス** テストの実行率、合格率、不合格率など、既存の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** プロダクトが想定ユーザーの期待する品質をどの程度満たしているかなどのプロダクト属性を測定する。
- **プロセスメトリクス** テストプロセスの能力とテストの有効性を測定する。したがって、プロセスメトリクスは、プロセスに関連する有効性と効率性を報告するために使用する。

プロダクトおよびプロセスのメトリクスマネジメントに関する詳細情報は、ISTQB® Expert Test Management Syllabus に記載がある。

プロセスメトリクスの使用に関する詳細は、ISTQB® Expert Improving the Test Process Syllabus に記載がある。

以下では、テスト計画、テストモニタリング、テストコントロール、テスト完了に関するメトリクスについて説明する。これらは、メトリクスに関連する 4 つの主要なマネジメント活動である。

2.1.1 テストマネジメント活動のためのメトリクス

Advanced Levelレベルのテストマネジメントシラバスは、以下の一般的なテストマネジメント活動に焦点を当てている：

- テスト計画
- テストモニタリングとテストコントロール
- テスト完了（1.1 節「テストプロセス」参照）。

テストマネジメントは、テスト計画の一部として、テストモニタリング、テストコントロールおよびテスト完了のための一連のテストメトリクスを定義できなければならない。各メトリクスを定義し、測定し、モニタリングして、報告する必要がある。

テスト計画では、プロジェクトテスト戦略からテスト目的に合う適切なテストメトリクスを定義する。

テストモニタリングとテストコントロールで使用されるメトリクスは、テスト完了時に使用されるものとは異なる場合がある。テストモニタリングとテストコントロールでは、メトリクスはテスト活動の進捗に関するものである。テスト完了時には、テスト目的を達成する必要がある。与えられたテスト目的に割り当てられたテスト終了基準を測定するために、1 つもしくは複数のメトリクスを組み合わせることができる。

以下の表は、テストマネジメント活動で使用するメトリクスの例である（他にもたくさんある）：

メトリクス（定義したマイルストーンに対する計画／モニタリング）	テストモニタリングとテストコントロール	テスト完了
要件カバレッジ	○	○
プロダクトリスクカバレッジ	○	○
コードカバレッジ	○	
テスト活動の実績 対 計画時の見積り（時間単位）	○	
テストケースのステータス（不合格、ブロックなど）ごとの実行割合 対 計画したテストケース	○	○
欠陥解決数累計 対 欠陥数累計	○	
実際の自動化テストケース 対 計画した自動化テストケース		○
実際のテストコスト 対 計画したテストコスト	○	

表 2：テストマネジメント活動で使用されるメトリクスの例

特定のテスト活動で使用されるメトリクスを表に示している。テストモニタリングとテストコントロールで ○ が付いているメトリクスは、主に進捗を測定するために使い、テスト進捗レポート(CTFL)で報告する。テスト完了に ○ が付いているメトリクスは、主にテスト目的の達成度を測定するために使い、テスト完了レポート(CTFL)で報告する。両方の欄に ○ が付いているメトリクスは、どちらにも使用できる。

また、テストの有効性をモニタリングするためのメトリクス（欠陥検出率(DDP)など）もある。

DDP は、ISTQB® Expert Level のシラバスのテストプロセス改善モジュール、特にテストプロセス改善の実装の節で扱われる。

2.1.2 モニタリング、コントロール、完了

テストメトリクスとは、テストがどこまで進んだか、終了基準または関連するテストタスクが達成されたかどうかを示す指標である。

テストモニタリングとは、テストとそれに関連する評価とアセスメントに関するデータを収集する活動である。テストモニタリングは、テストの進捗状況の評価、および終了基準または関連するテスト活動の達成を確認するために使用する（1.1.2 項「テストモニタリングとコントロールの活動」参照）。終了基準はテスト目的から導き出す。

テストコントロールは、効果的かつ効率的なテストを達成するためのガイダンスと是正措置を提供するために、テストモニタリングからの情報を利用する。テストコントロールに関する例には、識別したリスクが課題事項となった場合のテストの再優先順位付け、手戻りによりテストアイテムが開始基準または終了基準を満たすかどうかの再評価、

テスト環境の提供の遅れを考慮したテストスケジュールの調整、必要なとき必要な場所での新しいリソースの追加などが含まれる。

テスト完了は、完了したテスト活動のデータを収集し、学んだ教訓、テストウェア、その他の関連情報を統合する。テスト完了は、テストレベルの完了、イテレーションの完了、テストプロジェクトの完了（または中止）、またはメンテナンスリリースの完了などのプロジェクトのマイルストーンで発生する。

テストマネジメント活動で使用される一般的なテストメトリクスには、プロジェクト進捗メトリクスや、計画スケジュールや予算に対する進捗、現在のテストアイテムの品質、テスト目的やイテレーション目的に対するテストの有効性を示すメトリクスなどがある。

2.1.3 テスト報告

テストマネジメントは、テストステータスを理解して報告するために、どのようにメトリクスを解釈し、使用するかを理解するべきである。システムテスト、システム統合テスト、受け入れテスト、セキュリティテストのようなより高いレベルのテストでは、主要なテストベースは、通常、要件仕様書、ユースケース、ユーザーストーリー、プロダクトリスクのような作業成果物である。構造カバレッジのメトリクスは、コンポーネントテスト（ステートメントカバレッジなど）やコンポーネント統合テスト（インターフェースカバレッジなど）のような、より低いレベルのテストに適用できる。テストマネジメントは、テストがテスト対象システムの構造をどの程度実行しているかを測定するためにコードカバレッジのメトリクスを使用することができるが、より高いレベルのテスト結果の報告は、プロジェクトの特定のコンテキストとニーズに合わせるべきである。例えば、頻繁に変更される環境では、コードカバレッジのメトリクスは、テストスイートに対するコード変更の影響をモニタリングし、潜在的なギャップやリスクを識別するのに有用であろう。加えて、コンポーネントテストとコンポーネント統合テストにて構造カバレッジ 100%を達成したとしても、欠陥と品質リスクはより高いレベルのテストレベルで対処する必要が残る。

メトリクスを報告する目的は、マネジメントの目的のために情報に対する素早い理解を提供することである。メトリクスは、ある時点におけるメトリクスのスナップショットとして報告することも、傾向を評価するために時系列的なメトリクスの変遷として報告することもできる。

プロダクトリスク、欠陥、テスト進捗、カバレッジ、関連コスト、テストの労力は、プロジェクト終了時に特定の方法で測定され、報告される。

以下は、目的別に使用できるメトリクスの例である：

プロダクトリスクに関連するメトリクスは以下の通り：

- すべてのテストが合格したリスクの割合
- 一部またはすべてのテストが不合格になったリスクの割合
- まだ全部がテストできていないリスクの割合

これらのメトリクスは、テストベースの品質と、プロダクトリスクをカバーするテストケースの有効性をアセスメントするために使用できる。

欠陥に関するメトリクスは以下の通りである：

- 欠陥数累計に対する解決済み欠陥数累計
- 欠陥の数または割合の内訳：

- テストアイテムまたはコンポーネント
- 欠陥の発生源（要件仕様、新機能、リグレッションなど）
- テストリリース
- テストレベルまたはイテレーションでの混入件数、検出件数、修正件数
- 優先度／重要度
- 根本原因
- ステータス（却下、重複、オープン、クローズなど）

これらのメトリクスは、欠陥の検出と解決のプロセスをモニタリングし、欠陥密度や欠陥の重要度が高い領域を特定し、テストの効率性と有効性を評価するために使用することができる。

テスト進捗に関連するメトリクスは以下の通り：

- テスト実行状況：計画、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
- テスト工数：テストに費やしたリソース時間の計画に対する実績

カバレッジに関するメトリクスは以下の通り：

- 要件カバレッジ：テストケースによってカバーしている要件の割合。
- プロダクトリスクのカバレッジ：識別したプロダクトリスクのうち、テストケースによって軽減した割合。
- コードカバレッジ：テストケースによって実行されるコードのステートメント、ブランチ、パス、または条件の割合。

コストとテスト労力に関するメトリクスは以下の通り：

- テストしていないコンポーネントの残存リスク：テストしていないコンポーネントに欠陥がある場合の潜在的な影響と可能性。
- テストコスト：テストコストの計画に対する実績。

さらに、異なるカテゴリーのメトリクスを組み合わせることも有用である（例えば、未解決の欠陥の傾向と実行したテストの傾向の相関を示すメトリクスや、要件に見つかった欠陥の数に基づいてテストベースの品質を示すメトリクスなど）。テスト実行を継続し、識別される欠陥が少なくなってきたら、テストを終了する判断を下すことができる。この判断は、メトリクスの報告と合意された終了基準に基づくべきである。

2.2 テスト見積り

イントロダクション

システムやソフトウェアエンジニアリングの見積りに関するプロジェクトマネジメントのよい実践例が存在し、これには、あらゆる種類のリソース（例えば、コスト、人、時間）に関するものを含む。テスト見積りとは、これらのよい実践例を、プロジェクトや運用に関連するテストの活動に適用することである。

2.2.1 テストに関する活動の見積り

テスト見積りとは、テストマネジメントの活動の 1 つで、タスクの完了までにどれだけの時間、工数、コストがかかるかを見積ることである。テスト見積りは、テストマネジメントにおける主要かつ重要なタスクの 1 つである。

テストマネジメントにおける見積りの主な特徴は以下の通り：

- **工数**は通常、プロジェクトのテストタスクを完了するために必要な労力またはストーリーポイントで計算される。多くの場合、テスト工数とテスト期間（経過時間）は異なることがあり、テストマネジメントは活動の総期間を見積る必要があるかもしれない。完了するのに何人時がかかるのか？
- **時間**はプロジェクトを完了するのに必要な時間である。時間はプロジェクトにおいて重要なリソースである。テスト計画では、テスト工数をカレンダー日数と稼働日数で見積る必要がある。すべてのプロジェクトには、マイルストーンと納期がある。テストプロジェクトを完了させるのに、どれくらいの時間がかかるのか？
- **コスト**はプロジェクトの予算である。これには、テストリソース、ツール、インフラの費用を含む。テストプロジェクトにはどれくらいのコストがかかるのか？

テストは（大規模な）プロジェクト内のサブプロジェクトであることが多く、いくつかのテスト拠点（テストセンターなど）に分散していることもある。テスト見積りを行うには、最初のステップは、テストレベル、テスト活動、テストタスクを識別する。次に、テストプロジェクトをテストプロセスの中で、主要なテスト活動（例えば、テスト計画とテスト実行）に分割する(ISTQB® Foundation Level Syllabus V.4 を参照)。アジャイルプロジェクトでは、テスト活動は開発作業の中で見積ることが多く、個別の値として見積ることはない。次のステップは、タスクまたは作業成果物を完成させるために必要なテスト工数と、その予想コストを見積ることである。

テストはプロジェクトのサブプロジェクトであるため、常に見積りに影響を与えるプロジェクト制約が自然に存在し、妥協が必要となる。これらの値を恣意的に操作することはできない。これは、品質マネジメントにおける、時間-コスト-品質の三角形に見られるものである。プロジェクトマネジメントにおいて、この三角形は相互依存する 3 つの値で構成されており、それぞれが密接に関連し合い、互いに影響を与える。この関係は、プロジェクトのシナリオでよく見られる。

2.2.2 テスト工数に影響を与える可能性のある要因

テスト工数の見積りには、特定のプロジェクト、リリース、またはイテレーションのテスト目的を達成するために必要となるテスト活動関連の作業量を予測することが含まれる。テスト工数に影響を与える要因には、以下の特性を含む:

プロダクト:

- テストベースの品質
- テストするプロダクト(テスト対象)の大きさ
- プロダクトドメインの複雑度(環境、インフラ、歴史など)
- 品質特性(例えば、セキュリティや信頼性)をテストするための要件。

これらのプロダクトに関連する要因は、テスト対象システム特有のコンテキストを作り出すため、テスト見積りに影響を与える可能性がある。

開発プロセス:

- 組織の開発プロセスの安定性と成熟度
- 使用している開発モデル(例:アジャイルソフトウェア開発/イテレーティブ開発、またはハイブリッドソフトウェア開発モデル)
- 物理的な要因(テスト自動化、ツール、テスト環境の可用性など)。

テストは開発に直接関係するため、これらの開発プロセスに関連する要因は、テスト見積りに影響を与える可能性がある。

人:

- 人々の満足度(祝祭日、休暇、その他の期待する福利厚生など)
- 関与する人のスキルや経験、特に類似のプロジェクトやプロダクトに関する経験(ドメイン知識など)

人は最も必要なリソースであるため、いかなる不安定な状況も考慮する必要がある。したがって、テスト工数の見積りにおいて、人は重要な要素である。本シラバスの 3.1 節「テストチーム」も参照のこと。

テスト結果:

- テスト実行中に発見された欠陥の数と重要度
- 必要な再作業の量

過去の統計はテスト見積りをサポートする。したがって、これらの要因を知ることは、より正確な値での見積りをサポートすることになる。

テストコンテキスト:

- 複数の組織にわたるテストの分散、チームの構成と所在地、プロジェクトの複雑度(複数のサブシステムなど)。
- 働き方(例:バーチャル、またはオンサイト)

コンテキストに関連する要因とは、テスト見積り全体に影響する要因である。本シラバスの 1.2 節「テストのコンテキスト」も参照のこと。

2.2.3 テスト見積り技法の選択

テスト見積りは、テストプロセスに関わるすべての活動を対象とすべきである。テスト実行のコスト、工数、特に期間の見積りは、テストマネジメントにとって最も影響があることが多い。しかし、全体的なソフトウェアの品質が低い、あるいは未知である場合、テスト実行見積りの作成は困難である傾向がある。加えて、そのプロダクトに慣れ親しんだ経験も、見積りの質に影響する可能性が高い。一般的な実践例は、テストベース（要件やユーザーストーリーなど）から導き出されたテストケースの数 を見積ることである。テスト見積りにおける前提条件は、常に見積りの一部として文書化すべきである。

テスト見積り技法やアプローチは、メトリクスベースとエキスパートベースに分類できる。

テスト見積りの技法についての詳細については、ISTQB® Foundation Level Syllabus V.4 で説明されている。

ほとんどの場合、一度作成した見積りは、正当な理由とともにプロジェクトマネジメントに提出しなければならない。頻繁に、いくつかの入力パラメーター（例えば、テストスコープ）が変更され、その結果テスト見積りが調整される。理想的には、最終的なテスト見積りは、品質、スケジュール、予算、機能の領域において、組織のゴールとプロジェクトのゴールの可能な限り最良のバランスを表している。

どのような見積りも、作成時点での有効な情報に基づくものであることを肝に銘じておくことが重要である。プロジェクトの初期段階では、情報はかなり限られているかもしれない。さらに、この情報は時間の経過とともに変化することもある。正確さを保つために、見積りは新しい情報や変化する情報を反映して更新されるべきである。

見積り技法の選択は、以下のようなさまざまな要因に左右される：

- **見積り誤差：**いくつかの技法は、見積りの不確実性やばらつきを測定する尺度である標準偏差を計算する方法を提供している。例えば、三点見積り技法は、楽観的推定値、悲観的推定値、および最も可能性の高い推定値を使用して、期待値と推定値の標準偏差を計算する（詳細については、ISTQB® Foundation Level Syllabus V.4、5.1.4 項を参照）。
- **データの可用性：**手法によっては、過去のプロジェクトや類似プロジェクトの履歴データを要件とするものがあるが、これらのデータが入手できない、または信頼できない場合がある。例えば、比率や外挿による見積りは、現在のプロジェクトの比率や傾向を導き出すために過去のデータに依存する。
- **専門家の可用性：**手法によっては、正確で現実的な見積りを提供するための知識と経験を持つ専門家の関与が要件となるものもある。例えば、デルファイ法やプランニングポーカーは、専門家やチームメンバーの意見や判断に依存している。
- **モデリングの知識：**手法によっては，見積りを計算するために数学的モデルや数式を使用する要件があるため，モデリングに関するスキルや知識が必要になる場合がある．例えば、外挿や三点見積りは、見積りの期待値と標準偏差を導き出すために数式を使用する。
- **時間的制約：**技法によっては、他のものより多くの時間と労力を必要とするものがあり、それが技法の実現可能性や適合性に影響する場合がある。例えば、プランニングポーカーは簡単にできるが、外挿は難しいかもしれない。

このことは、適切なテスト見積り技法の選択基準が、テストのコンテキスト（例えば、SDLC、ステークホルダー、プロジェクトで使用されるテストレベルやテストタイプ）に大きく依存することを示している（本シラバスの 1.2 節「テストのコンテキスト」を参照）。テストマネージャーは、テスト見積り技法を調整し、適用できなければならない。（例えば、組織をまたいだ 1 つのプロジェクトで、異なる SDLC モデルを適用する場合など）。

例えば、適切な見積り技法を選択するには、まず対象の複雑度を判断する。複雑度が低い場合は、メトリクスベースの技法を使用することができる。複雑度が高い場合は、専門家ベースの技法を使用できる。シーケンシャル開発モデルを使用する場合、ワイドバンドデルファイ見積り技法が使用できる。アジャイルソフトウェア開発モデルを使用する場合は、プランニングポーカーを使用できる。

2.3 欠陥マネジメント

イントロダクション

ISTQB® Foundation Level Syllabus V.4 には、期待結果と異なる実際の結果を観察した後に開始する活動が記述されている。本シラバスでは、これらの活動を欠陥マネジメントと呼んでいる。他の標準では、「インシデントマネジメント (ISO/IEC/IEEE 29119-3 標準)」や「不正マネジメント(TMAP)」という用語を使って、このプロセスの初期段階では、その不一致が作業成果物の欠陥によるものなのか、それとも他の何か（例えば、自動テストの故障またはテスト担当者の要件に対する誤解など）によるものなのか分からないという事実を強調している。欠陥マネジメントとそこで使用する欠陥を取り扱うツールは、テスト担当者とソフトウェア開発に関わる他のチームメンバーにとって極めて重要である。効果的な欠陥マネジメントプロセスから得られる情報によって、テストチームと他のプロジェクトのステークホルダーは、SDLC 全体を通して、プロジェクトの状態を把握することができる。欠陥マネジメントは、どの欠陥を修正するかを決定するためにも重要である。これによって、欠陥修正に対する作業に適切に労力が費やされることが保証できる。欠陥に関連するデータを長期にわたって収集し、分析することは、テストと SDLC のうちの他のプロセス（例えば、アーキテクチャと技術的な設計の改善による、よりよい欠陥予防）の両方について、潜在的な改善領域を特定するのに役立つ。

欠陥ライフサイクル全体を理解し、それがソフトウェア開発とテストプロセスの両方をモニタリングしコントロールするためにどのように使用されるかを理解することに加えて、テストマネージャーとテスト担当者（またはアジャイルソフトウェア開発ではアジャイルチーム全体）は、どのデータを取得することが重要であるかを熟知していなければならない。テストマネージャーは、欠陥マネジメント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利用を推進する役割を果たさなければならない。

2.3.1 欠陥のライフサイクル

SDLC の各フェーズには、潜在的な欠陥を検出し、取り除く活動を含めるべきである。例えば、静的テスト技法（すなわち、レビューと静的解析）は、設計仕様書、要件仕様書、コードに対して、それらの作業成果物を後続の活動に提供する前に使用することができる。各欠陥を早期に検出して、取り除くことができることで、プロダクトの全体的な品質コストは低下する。各欠陥が混入したのと同じフェーズ内で取り除けるとき（すなわち、ソフトウェアプロセスがフェーズ内封じ込めを完全に達成するとき）、品質コストは最小化される。

静的テストでは欠陥を探す。動的テストでは、欠陥の存在は、欠陥が故障を引き起こしたときに明らかになる。その結果、実際のテスト結果と期待するテスト結果との間に違いが生じる（すなわち不正）。場合によっては、テスト担当者が不正を観測しなかったときに偽陰性結果が発生する。不正を観察した場合、さらなる調査を行うべきである。この調査は通常、定義されたテストおよび欠陥マネジメントプロセスにしたがって、欠陥レポートを作成することから始まる。テストが不合格だとしても、欠陥レポートが作成されるとは限らない。（例えば、テスト駆動開発では、通常自動化されたコンポーネントテストが、実行可能な設計仕様書の一形態として使用される）。コンポーネントの開発が完了するまでは、テストの一部または全部を、最初は不合格にしなければならない。したがって、このようなテスト結果は、必ずしも欠陥が原因とは限らず、通常、欠陥レポートによって追跡されることはない。

欠陥レポートは、ワークフローに沿って進行し（欠陥マネジメントツールの多くとの整合性を保つため、以下では「欠陥ワークフロー」という用語を使用する）、欠陥の一連の状態を通過する。これらの状態のほとんどにおいて、1人が欠陥レポートを所有し、タスク（例えば、分析、欠陥除去、確認テスト）の実行を担当する。以下の図は、単純な欠陥ワークフローを表している：

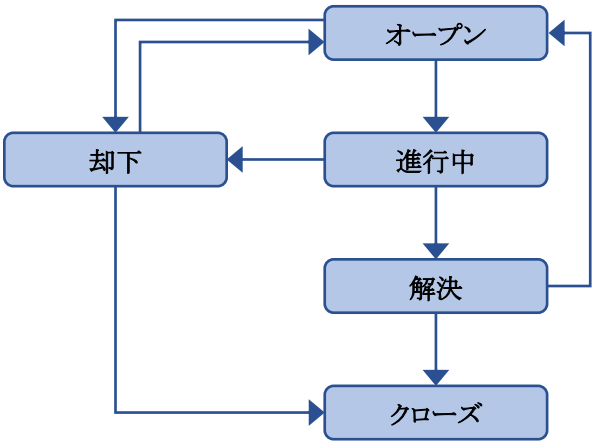

図 2: 単純な欠陥ワークフロー

単純な欠陥ワークフローは以下の欠陥状態をカバーする:

- オープン(新規と呼ぶこともある):欠陥レポートが作成されたときの初期状態。
- 進行中:チームは欠陥レポートの分析および/または修正に取り組んでいる。
- 却下:欠陥レポートは、それを処理した人 (通常は開発者やアナリスト) が却下している。却下の理由はさまざまであり(例えば、無効な情報、誤ったテスト、欠陥レポートの重複)、その情報は欠陥レポートへ追加する。
- 解決(修正済み、再テスト待ちと呼ぶこともある):テスト担当者は、修正によって欠陥が本当に解決されたかどうかを判断するために、欠陥レポートに記載された欠陥の再現手順にしたがって確認テストを実施する。
- クローズ:欠陥レポートは終了状態に達しており、これ以上の作業を行う予定はない。テスト担当者は、確認テストが成功した後、または欠陥レポートの拒絶を受け入れたことに対して、欠陥レポートをこの状態に遷移させる。

単純な欠陥ワークフローは、多くの組織で使用され、与えられたコンテキストに関連する他の欠陥状態(例えば、再オープン、受け入れ、明確化、見送り)を使用することによって拡張される。

欠陥ワークフローは、欠陥ステータスの名称、欠陥ステータス間の遷移のルール、特定の欠陥ステータスのタスクに責任を持つ役割という点で、組織によって異なる場合がある。アジャイルソフトウェア開発では、多くの場合、欠陥ワークフローはシーケンシャル開発モデルよりも単純である。欠陥ワークフローは、与えられたコンテキストに適応させるべきである。欠陥ワークフローを設計する際には、いくつかのよい実践例を尊重することが望ましい:

- 可能であれば、すべてのプロジェクトで統一された欠陥マネジメントを行うために、組織全体で定義すべきである。
- 重複欠陥や偽陽性欠陥は、別のステータス、または却下理由の選択と 却下ステータスを組み合わせて表すべきである。これらは、テストプロセスの改善を目的とした欠陥分析に役立つ可能性がある。

- 1 つの終端ステータス（例えば、クローズ）のみを使用することを推奨する。このステータスへの遷移は、多くの場合、プロセスアセスメントやプロセス改善活動に有用な、終結の理由を選択することを要件とする。
- 欠陥ワークフローにおけるステータスの名前は、作業を簡単にするために他のエンティティ（ユーザーストーリーやテストタスクなど）で使う類似のステータスと同じ名前にすべきである。
- 連続する欠陥ステータスは、異なる責任を持つ役割に割り当てるべきである。連続する 2 つ以上のステータスを同じ責任を持つ役割に割り当てる場合は、正当な理由があるべきである（例えば、欠陥ステータスに費やされた時間を測定するため）。
- 終端ステータスを除く各欠陥ステータスには、次のステップに関して責任を持つ役割が判断できるように、1 つ以上の遷移先を持つべきである。このルールの例外は正当化されるべきである（例えば、ある活動に費やされた時間をモニタリングするため）。
- ステータス遷移を実行する際に入力が要求される属性のセットは、欠陥マネジメントに実質的な価値を与えるものに限定すべきである。

2.3.2 機能横断的な欠陥マネジメント

テスト組織とテストマネージャーは、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していることが多いが、通常、特定のプロジェクトの欠陥マネジメントは、機能横断的なチームが担当する。このチームは、欠陥マネジメント委員会と呼ばれることもあり、テストマネージャー、開発の代表者、サプライヤー、プロジェクトマネジメント、プロダクトマネジメント、プロダクトオーナー、およびテスト対象のソフトウェアに関心のあるその他のステークホルダーの代表者を含むことがある。

不正が発見され、欠陥マネジメントツールに入力されると、欠陥マネジメント委員会は、各欠陥レポートが有効な欠陥であるかどうかを判断し、その欠陥を修正すべきか（そして、複数の開発チームが開発に参加している場合は、どのチームが修正すべきか）、却下すべきか、延期すべきかを決定する。この判定には、欠陥マネジメント委員会が、欠陥の修正に関連する利点、リスク、コストを検討する必要がある。この検討については、ミーティング（しばしばトリアージミーティングと呼ばれる）で議論することが有益である。欠陥を修正する場合、チームは、他のタスクに対する欠陥修正の優先度を設定すべきである。欠陥の相対的な重要性については、テストマネージャーとテストチームに相談し、有効な客観的情報を提供すべきである。

非常に大規模なプロジェクトでは、欠陥マネジメント委員会のミーティングでの意思決定の準備およびフォローに必要な労力を考慮すると、専任の欠陥マネージャーを任命することが適切な場合がある。少なくとも、テストが最も集中的に行われる SDLC フェーズではこのような対応が必要になる。他の状況では、いくつかの大規模なプロジェクトが欠陥マネージャーを共有することもある。

欠陥マネジメントツールを、優れたコミュニケーションの代用として使用すべきではなく、欠陥マネジメント委員会も、優れた欠陥マネジメントツールを効果的に使用する代用として使用すべきではない。効果的かつ効率的な欠陥マネジメントには、コミュニケーション、適切なツールサポート、明確に定義された欠陥ワークフロー（欠陥レポートの属性を含む。）そして、欠陥マネジメントチームの積極的な関与がすべて必要である。

2.3.3 アジャイルチームにおける欠陥マネジメントの特徴

アジャイルソフトウェア開発を使用している組織における欠陥マネジメントは、シーケンシャル開発モデルに比べて軽量的であることが多く、そして/または形式的でない場合が多い。アジャイルチームが同じ場所にいる場合、また

はコミュニケーション手段が十分に確立されている場合、形式的な欠陥レポートがなくても、テスト担当者、顧客担当者、開発者の間で欠陥や故障に関する情報交換を行うことが多い。しかし、次のような場合は欠陥レポートを作成すべきである:

- 他のスプリント活動(開発、テスト、またはその他)をブロックし、アジャイルチーム内ですぐに修正できない欠陥。
- 同じイテレーション内で解決できない欠陥。アジャイルチームによっては、故障が発見されたその日のうちに欠陥が解決できない場合、欠陥レポートを作成するというルールとしていることもある。
- 複合チーム組織において、他のチームによって、または他のチームと協力して解決しなければならない欠陥。
- サプライヤーが解決しなければならない欠陥。
- 欠陥レポートが明示的に求められている欠陥。(例えば、開発者がすぐに修正に取りかかれない場合など)。

一般的な実践例は、同じイテレーション内で解決できない欠陥をプロダクトバックログに追加し、後のイテレーションのために他の欠陥やユーザーストーリーの中で優先順位を付けられるようにすることである。

欠陥マネジメントの基本は組織的テスト戦略で定めるべきであるが、形式化のレベル、欠陥レポートを作成するトリガー、取り込むべき欠陥の属性など、多くの側面はアジャイルチームメンバーの合意に任されている。一般的に、欠陥マネジメントの形式化のレベルや欠陥レポートを作成するアプローチは、以下を反映すべきである:

- チームメンバーの共通の作業場所
- 時間帯を超えたチームメンバーの配置
- プロダクト開発に参画するチームの数
- チームの成熟性
- チームの規模
- プロダクトに関するリスク
- 規制、契約、またはその他の要件(該当する場合)

アジャイルチームによる欠陥マネジメントの詳細に関する最終決定は、常に文書化すべきである(例えば、ナレッジマネジメントツールのガイドラインに記載する)。

2.3.4 ハイブリッドソフトウェア開発における欠陥マネジメントの課題

実際には、複数のチームがシステムまたはシステムオブシステムの開発から提供までを協業して行うことが多い。例としては、顧客がアジャイルソフトウェア開発を使用し、サプライヤーの 1 つがシーケンシャル開発モデルを使用するハイブリッドソフトウェア開発、または、シーケンシャル開発モデルを使用している組織が、アジャイルソフトウェア開発を使用しているチームからサブシステムの提供を求める場合である。このような複合チーム環境では、さまざまな課題が発生する:

- **欠陥の属性と欠陥マネジメントに使用するツールに関する整合性:**理想的なシナリオでは、すべてのチームが 1 つの欠陥マネジメントツールを使用する。実際には、各チームが異なる欠陥マネジメントツールを使用するのが一般的である。特に、複数のサプライヤーチームがプロジェクトにて開発から提供まで貢献している場合はなおさらである。このような場合、欠陥マネジメントツール間の同期を確立することが望ましい(できれば自動的に)。
- **欠陥の優先順位付け:**プロダクトオーナーは欠陥マネジメントミーティングに関与し、欠陥に関連する結果とリスクに関する情報を積極的に求めるべきである。アジャイルソフトウェア開発では、アジャイルチームのプロダクトインクリメントの開発から提供までの速さに歩調を合わせるために、シーケンシャル開発モデルよりも欠陥マネジメントミーティングを頻繁に開催すべきである。しかし、アジャイルチームでは、これらのミーティングは短くてもよい。欠陥の優先順位付けについて、より少数の欠陥マネジメントのステークホルダーが最終的な判断を下すことが有益なことがある。
- **新規開発と欠陥修正のためのテスト計画書の整合性と透明性**:アジャイルソフトウェア開発モデルを使用しているか、シーケンシャル開発モデルを使用しているかに関係なく、すべてのチームの仕事は同じプロジェクト計画書に合わせるべきである。欠陥修正を含むすべての成果物は、このプロジェクト計画に沿ったものでなければならない。すべてのチームのメンバーが計画プロセスに積極的に参加することで、よりよい整合性を達成することができる(例えば、シーケンシャル開発モデルのチームが、アジャイルソフトウェア開発ミーティングに参加し、欠陥について議論し、優先順位を付ける。)開発計画書の透明性は、チーム間で共有することで向上する(例えば、ダッシュボードやプロダクトバックログを介して)。

2.3.5 欠陥レポート情報

欠陥レポートに記載される情報は、以下の目的で十分である。:

- 欠陥のライフサイクルを通じた欠陥レポートのマネジメント
- プロジェクト全体の状況、特にプロダクト品質やテストの進捗状況のアセスメント
- プロダクト品質の観点から、プロダクトインクリメントの状況のアセスメント
- プロセス能力のアセスメント

欠陥マネジメントとプロジェクトのステータスに必要な情報は、欠陥が SDLC の中でいつ検出されたかによって変わる。さらに、非機能品質特性に関連する欠陥レポートは、より多くの情報を必要とするかもしれない(例えば、性能課題の負荷条件)。しかしながら、収集される核となる情報は、SDLC 全体で、理想的には組織内のすべてのプロジェクトで一貫しているべきである。

欠陥レポートでは多くのデータ項目を収集することができる。テストマネージャーは、プロジェクトのコンテキストに応じて、どの情報が効果的な欠陥マネジメントに適しているかを判断する必要がある。属性を追加するたびに、欠陥レポート作成に費やす時間が長くなり、欠陥レポートを入力する担当者の混乱を招く可能性があるため、特定のコンテキストにおける欠陥マネジメントに必要なデータ、および/またはプロセス改善に使用されるデータのみを収集することが望ましい。

欠陥レポートをマネジメントするほとんどの環境では、以下のことが必須である:

- 欠陥のタイトルと不正の簡単な概要
- 故障を再現する手順を含む不正の詳細な説明

- テスト対象システムおよび/またはプロダクトのステークホルダーへの影響の重要度
- 不正を修正する優先度

その他の重要なデータ項目は、欠陥マネジメントツールによって作成されることが多い:

- 欠陥レポートの一意な識別子
- 欠陥レポート作成日時
- 不正を発見した人、そして/または報告した人の名前
- 不正が発見されたプロジェクトと SDLC フェーズ
- 欠陥レポートのステータス
- 現在の所有者(すなわち、その欠陥の作業を現在割り当てられている人)
- 欠陥の切り分け、修正、および修正済みであることの確認のためにプロジェクトチームメンバーが行った、日時情報を含む一連のアクションなどの変更履歴。
- 参照(テストケースや関連する欠陥など)。

コンテキストによっては、欠陥レポートでさらなる情報(トレーサビリティなど)を収集することもある(詳細は ISO/IEC/IEEE 29119-3 を参照)。以下の箇条書きは、意図する目的にしたがって情報をグループ化したものである:

- **欠陥解決に役立てる**:欠陥が存在するサブシステムまたはコンポーネント、不正が観察された特定のテストアイテムおよびそのリリース番号、または欠陥が観察されたテスト環境。
- **プロジェクト全体の状況をアセスメントする**:進捗状況をモニタリングするための情報(例えば、欠陥の修正するまたは修正しないことに関連するリスク、コスト、機会、利益、取り得る回避策の説明、欠陥によって影響を受ける要件など)。
- **プロダクトインクリメントの状態をプロダクト品質の観点からアセスメントする**:欠陥の種類(通常、欠陥分類法に対応する)、欠陥が混入した作業成果物、または欠陥によって影響を受けた品質特性/副特性。
- **プロセス能力をアセスメントする**:開発プロセスの有効性と効率性をモニタリングするための情報(例えば、欠陥または欠陥の根本原因に対する混入、検出、除去の SDLC フェーズ)。

2.3.6 欠陥レポート情報を用いたプロセス改善アクションの定義

本シラバスの 2.3.5 項「欠陥レポート情報」で述べたように、欠陥レポートは、プロジェクトのステータスのモニタリングや報告に有用である。テストプロセスにおけるメトリクスの意味合いは、主に Expert Test Management シラバスで扱われるが、Advanced Test Management Level では、テストマネージャーは、欠陥レポートがソフトウェア開発とテストプロセスの能力をアセスメントする上でどのような意味を持つかを認識しておくべきである。

本シラバスの 2.1.2 項「モニタリング、コントロール、完了」、および 2.1.3 項「テスト報告」で述べられている欠陥モニタリング情報に加えて、欠陥情報は、ふりかえりで議論されるようなプロセス改善のための施策をサポートすべきである。例えば、以下のようなものがある:

- 欠陥の混入、検出、除去のフェーズに関する情報を使用して、フェーズ内封じ込めをアセスメントし、そして/または品質コスト分析を行い、各フェーズにおける欠陥検出効果を向上させ、欠陥に関連するコストを最小化する方法を提案する。
- 欠陥が最も多く混入するフェーズの分析に混入フェーズの情報を使用して、欠陥予防のための的を絞った改善を可能にする。
- 欠陥の根本原因に関する情報を使用して、欠陥の総数を減らすプロセス改善を可能にする。
- 欠陥の発生箇所に関する情報を使用して、欠陥のクラスター分析を行い、(リスクベースドテストのための)技術的リスクをよりよく理解して、問題のあるコンポーネントのリファクタリングを可能にする。
- 再オープンした欠陥に関する情報を使用して、デバッグの際の実装の品質をアセスメントする。
- 重複した欠陥や却下した欠陥に関する情報を使用して、欠陥レポート作成の品質をアセスメントする。
- エラーを未然に防ぐための積極的な手段を導入することで、欠陥の総数を減らすプロセス改善を可能にする。

テストプロセスの有効性と効率性をアセスメントするためのメトリクスの使用については、Expert Test Management シラバスで説明している。

場合によっては、チームは SDLC の一部またはすべてのフェーズで発見された欠陥を追跡しないことを決定する。これは、効率性の名の下に、また、プロセスのオーバーヘッドを削減するために実施されることも多いが、ソフトウェア開発とテストのプロセス能力の可視性を大幅に低下させる。これは、信頼できるサポートデータが不足し、上記で提案した改善の実施を困難にする。

3 チームのマネジメント - 225 分

キーワード

評定、品質コスト、欠陥、欠陥予防、外部失敗、故障、内部失敗

第 3 章の学習の目的

3.1 テストチーム

TM-3.1.1 (K2)テストチームメンバーが必要とされる典型的なスキルについて、4 つの能力の領域から例を挙げる。

TM-3.1.2 (K4) テストチームメンバーに要求されるスキルを判断するため、与えられたプロジェクトのコンテキストを分析する。

TM-3.1.3 (K2) テストチームメンバーのスキルアセスメントの典型的な技法について説明する。

TM-3.1.4 (K2) テストチームメンバーのスキルを向上させるための典型的なアプローチを区別する。

TM-3.1.5 (K2) テストチームのマネジメントに求められるスキルを説明する。

TM-3.1.6 (K2) テストチームメンバーのモチベーションを上げる要因と衛生要因の例を挙げる。

3.2 ステークホルダーとの関係

TM-3.2.1 (K2) 品質コストを決定する 4 つのカテゴリーについて、それぞれの例を挙げる。

TM-3.2.2 (K3)ステークホルダーにとってのテストの付加価値を見積るために、費用対効果の計算を適用する。

3.1 テストチーム

イントロダクション

テストタスクを実行するすべてのチームは、異なる能力を持つ個人で構成される。チームが自己組織化される組織もあれば、テストマネージャーがメンバーを募集し、チームを育成する組織もある。すべてのチームがテストタスクを成功裏に完了するためには、スキルの適切な組み合わせ[1] が重要な要素である。

テストチームメンバーに求められるスキルは、時間の経過とともに変化する可能性がある。テストチームに適切な人材を選び、適切なトレーニングと成長の機会を提供することが重要である。加えて、テストチーム外の人々が、さらに特定のスキルを提供することもある。

この章では、テストチームメンバーに必要なスキルを分析し、開発するための基本的なプロセス、および、テストチームをリードしたり、コーチしたりするために必要なスキルについて見ていく。また、テストチームメンバーのモチベーションを上げる要因やモチベーションを下げる要因など、チームワークを成功させるための知識も含まれる。

各個人はすでにスキルを持っており、実務経験、教育、トレーニングなど、さまざまな方法でこれらのスキルをさらに伸ばすことができる。理想的なテストチームは、与えられたテストタスクに必要なスキルをすべて持っているか、または、必要なスキルを持つタスクのみを担当する。テストチームが成功するためには、さまざまなレベルのさまざまなスキルが必要である。プロジェクトのコンテキストによって、他のスキルよりも重要なスキルや必要なスキルがある。テストチームの能力を超えるような特定のテストタスクについては、外部の専門家を導入することが理にかなっているかもしれない。

3.1.1 4 つの能力領域における代表的なスキル

人のスキルは 4 つの能力領域に分類できる。(Sonntag & Schmidt-Rathjens, 2005) (Erpenbeck & von Rosenstiel, 2017) [2]:

- **専門的能力:** 専門的な業務を遂行するためのスキルで構成される。例えば、テスト技法のスキル、アプリケーション領域における技術的専門知識、ビジネス専門知識、プロジェクトマネジメントスキルなどがある。
- **方法論的能力:** ある領域で独自に使用でき、複雑度または新規性の高い課題を独自に遂行できる一般的な能力を含む。例えば、分析力、概念力、判断力などがある。
- **社会的能力:** 内部および異文化間のコンテキストにおけるコミュニケーション、協力、対立マネジメントに関するスキルを含む。ある状況において適切に行動し、個人および共通のゴールを達成するために、他者との関わりを可能にする。例えば、コミュニケーションスキル、対立解決スキル、チームワーク、適応性、自己主張などを含む。
- **個人の能力:** 自分自身を成長させ、才能、モチベーション、実行意欲の他、具体的な態度や個人の個性を伸ばす能力と意欲を含む。例えば、セルフマネジメント能力、個人的責任感、批判を受ける能力、信頼

[1]「スキル」という言葉は、スキルそのもの、何かの知識を持っていること、何かをする能力を持っていることの総称として使われる。

[2]ここで使われている 4 つの能力領域は、これらの文献に記載されているモデルに基づいており、広く使われているものである。文献には、異なるスキルをグループ化した他のモデルも記述されている。これらは本シラバスの一部ではない。

性、回復力、自信を持って行動する能力、規律正しさ、変化を受け入れる姿勢、助け合いや学ぶ意欲、任せる能力などが挙げられる。

テストチームの成功には、すべての領域の能力が重要である。方法論的、社会的、個人の能力はテストに特化したものではないため、ISTQB® は、専門的能力の開発に重点を置く。これには、テストタスクのマネジメント、テストベースの分析、テストケースの設計、リスクの識別と分析、テスト環境、テストスクリプト、テストデータの開発、設定と保守などのスキルが含む。

3.1.2 テストチームメンバーに求められるスキルの分析

人員配置はテスト計画の中の活動である。これには、テスト戦略でテストを実装するために必要な人員の役割とスキルを特定するタスクが含まれる。プロジェクトに必要なスキルを決定するには、詳細な状況分析が必要である。

専門的・方法論的能力

テストでは、テストタスクに必要なテストスキルに焦点を当てる。以下はその例である:

- テスト計画には、テスト戦略を策定するための概念的な知識が求められる。
- テストモニタリングとテストコントロールには、すべてのテストタスクをマネジメントするためのプロジェクトマネジメントスキルが求められる。
- テスト分析には、テストベースとプロダクトリスクを分析する分析スキルが求められる。
- テスト設計には、テストケースを設計するためのテスト技法のスキルと、テスト環境を設計するための概念的な知識が求められる。
- テスト実装には、テストの選択に関する判断スキル、テストスクリプトのプログラミングやテスト環境の構築などの技術的な専門知識が求められる。
- テスト実行には、自動化テストの実行、探索的テストの実施、テスト結果の評価などの技術的専門知識が求められる。
- テスト完了には、プロジェクトの成果を伝える能力と、下した決定に対して自分自身で責任を持つことが求められる。

テストタイプやテストレベルが異なれば、求められるスキルも異なる(例えば、システムの機能適合性をアセスメントするためには、アプリケーションドメインのビジネスに関する専門知識が必要であり、コードの保守性をアセスメントするためには、技術的な専門知識が必要である)。

加えて、プロジェクトコンテキストは、求められる専門的能力に関する価値ある情報を提供する:

- システムドメインには、情報技術、自動車産業、ギャンブリング産業など、アプリケーション分野のビジネス専門知識が求められる。
- プロジェクトで使用するソフトウェアやシステムのアーキテクチャや技術には、例えばプログラミング言語、インターフェース技術、セキュリティの脆弱性などの技術的専門知識が求められる。
- SDLC では、例えば、テストレベル、テストの役割、特定のテスト技法に関する知識が求められる。

社会的能力

テストのコンテキストにおいて、社会的能力は、テストチームメンバーが他のチームメンバーとの関係において適切に振る舞い、テスト目的を達成することを可能にする。特に、コミュニケーション、協調性、対立解決スキル（例えば、最適でないテスト条件への建設的な対処や、開発者への欠陥の報告）を含む。

ソフトウェア開発とソフトウェアテストは通常、（異なる）グループの異なるメンバーによって行われ、コミュニケーションを通じてタスクを調整する。プロジェクトの成功には、コミュニケーションスキル、チームワークや対立解決の能力が求められる。しかし、求められる社会的スキルのレベルは、プロジェクトのコンテキストによって異なる場合がある。例えば、アジャイルソフトウェア開発では、ドキュメント中心のシーケンシャル開発モデルや離れた場所で行うテストよりも高い社会的スキルが求められる場合がある。

個人の能力

テストチームメンバーの有効性と効率性は、彼らのスキルや態度、それらを向上する意欲にも左右される。例えば、自己組織化されたアジャイルチームで働く場合、チームメンバー全員に対して、より高いレベルで自身をマネジメントすることと規律を求めるかもしれない。一方、例えば階層的なテストチームのテストマネージャーは、仕事を委任する能力が必要である。特に時間厳守が求められるプロジェクトでは、高い信頼性と回復力が求められることが多い。加えて、すべての SDLC モデルにおけるプロセスの変化では、助け合いや学ぼうとする意欲、変化に対するオープンな姿勢が重要である。

3.1.3 テストチームメンバーのスキルアセスメント

多くの場合、テストチームは既存の人員で構成される。チームメンバーの能力と自己開発の必要性を理解するために、テストマネジメントは、既存のテストチームのスキルをアセスメントし、必要なスキルと比較する必要がある。

チームやチームメンバーがより効果的に機能するモデルがある（例えば、Meredith Belbin の「チームの役割」、DISG® または PCM® ）。Belbin によれば(Belbin, 2010)、チームは異なる性格や役割のタイプで構成されることで、効果的に機能する。これらのモデルは、チームにどのようなスキルがあり、どのようなスキルが不足しているかを識別するのに役立つ。

テストチームメンバーの専門的、方法論的能力は、典型的なテストタスクを実際にやってもらうことでアセスメントすることができる:

- テスト戦略を概説して、同僚とフィードバックを話し合う。
- テストベースをレビューし、発見事項を伝える中で、コミュニケーション能力も明らかになる場合がある。
- 特定のプロジェクトのコンテキストに応じて、特定のテスト目的を達成するためのテスト技法を決定する。
- さまざまなテスト技法を適切に適用する。
- テスト結果のアセスメントを含むテスト完了レポートを作成する。

さらに、スキルは外部の資格、認定、実務経験、学位によってアセスメントすることができる。

特に、アジャイルソフトウェア開発においては、定期的にふりかえりに参加し、フィードバックを受けることで、チームが求めるスキルを識別する。経験豊富なコーチまたはメンターが、スキルを開発し、知識のギャップを識別して解決することを支援する。

3.1.4 テストチームメンバーのスキル開発

テストチームは、プロジェクト開始時に、必要なスキルをすべて持っているとは限らない。完璧な人材がそろうとは限らないが、強力なチームであれば、個人の長所と短所のバランスを取ることができる。

テストマネジメントやテストチームは、スキルマトリクスで必要なスキルと現在のスキルを比較することで、必要な開発ニーズを識別できる。これに基づいて、以下のような能力開発のアプローチを決定できる:

- トレーニングと教育は、通常は(バーチャルな)教室で、あらかじめ定義された知識や実践例を教えることである(例えば、トレーニングコースに人を派遣する、社内でトレーニングセッションを行う、自分たち向けのトレーニングを開発する、ライブの e ラーニングコースを利用する)。
- 自己学習とは、(バーチャルな)教室で勉強するのではなく、1 人で勉強することである(例えば、本を読んだり、録画したビデオを見たり、インターネットのリソースを調べたりする)。
- ピアラーニングは、同僚同士が知識、アイデア、経験を共有し、互いに学び合うことである。
- メンタリングやコーチングは、ある役割に就いて間もないチームメンバーが、コーチから個別に指導を受けたり、経験豊富なメンターから知識やスキル、および/または経験を得たりするアプローチである。経験者は、アドバイスや支援を提供する継続的なリソースとして機能する。
- OJT は、自己学習、ピアラーニング、メンタリングまたはコーチングを組み合わせたもので、広く知られている。

能力開発に対するすべてのアプローチが、同じように効果的かつ効率的であるとは限らない。例えば、自己学習やトレーニングは、専門的・方法論的な能力開発に適している。このため、テストに関する基本的な知識は、ISTQB®のトレーニングに参加するか、ISTQB®のシラバスに基づいて自己学習することで身に付けることができる。しかし、社会的および個人の能力を開発するためには、トレーニングやコーチングのアプローチを利用することを推奨する。これらは自己学習よりも効果が期待できることが多いためである。社会的交流、フィードバック、内省は、社会的能力、個人の能力を開発するための重要な成功要因である。

3.1.5 テストチームがうまくいくために必要なマネジメントスキル

テストチームを成功に導こうとする人は誰でもマネジメントスキルがなければならない。これには、基本的なマネジメント活動における専門的かつ方法論的な能力を含む(例:計画、進捗のモニタリング、コントロール、および報告)。テストには、特定のテストマネジメントの知識とスキルが必要である(例:異なるテストアプローチの知識、テスト戦略の開発、テスト技法の使用、または適用する SDLC の理解)。

テストチームをリードする、またはコーチすることは、他のテストチームメンバーとの関係において適切に行動し、変化する状況に応じて成長する能力と意欲を持つことを意味する。したがって、テストチームを率いるには、社会的能力、個人の能力が不可欠な成功要因である。これには、回復力、委任する能力、テストチームに受け入れられる資質が含まれる。さらに、プロジェクトにおけるテストの利益を主張し、テストの利点を促進し、すべてのステークホルダーとコミュニケーションを取って対立を解決する能力も含まれる。

テストチームの人材を確保するには、社会やチームの状況、および労働条件を分析するスキルが求められる。これらのスキルは、テストチームが労働条件に適合すること、可能であれば労働条件をテストチームに適応させることの確認に役立つ。加えて、テストチームは動的な開発プロセスに影響を受けるため、以下のような状況に応じたスキ

ルを必要とする(例えば、小集団開発の Tuckman モデルのフェーズにしたがう。)(Tuckman, 1965) (Bonebright, 2010):

- テストチームメンバーのテストチームへの参加を支援する意欲(形成期)。
- テストチーム内の対立を解決する能力(混乱期)。
- 合意された価値観やルールを確実に守るための規律と目標指向のリーダーシップ(統一期)
- テストチームに自分自身の責任を持たせるための委任能力(機能期)。
- テストチームを離れるメンバーに対して感謝と信頼を持って接する能力(散会期)

3.1.6 特定の状況におけるテストチームのモチベーションを上げる要因とモチベーションを下げる要因

満足し、モチベーションが高いテストチームメンバーは、生産性とパフォーマンスを向上させるので、プロジェクトの成功に大きな影響を与える。これが達成されると、クロストレーニングが非公式に行われ、テストチームメンバーは自分の仕事量をマネジメントできるようになり、テストマネジメントは外部の課題に対処する時間を多く取ることができる。動機付け-衛生理論(Herzberg, et al., 1993)は、モチベーションを上げる要因と衛生要因を区別している。

モチベーションを上げる要因は意識的に認識され、成長と満足感につながる。これは以下を含む:

- 仕事の達成に対する承認と評価(例えば、インセンティブや、テストチームメンバーが評価されるような個別アプローチ)
- 責任と自律性の向上(例えば、テストチーム内でテストプロセスの定義すること)
- テストチームメンバーが、達成可能であり、同時に努力する価値があると認識する、興味深く、有意義で、やりがいのあるタスク(例:テスト自動化のための新しいツールの選択と導入)
- プロフェッショナルとしての昇進と成長(例えば、経験豊富なテスト担当者からテストマネージャーやテストプロセスオーナーになるための成長)

衛生要因は通常、当然のものと見なされる。それを満たすことは自動的により大きな満足感をもたらすわけではないが、欠けていれば、テストチームのメンバーのモチベーションが下がる可能性がある:

- 適切な報酬(例えば、市場に見合った給与、時間外手当、充実した社会保障)
- 評価される人事方針とマネジメントスタイル(例えば、無駄のないマネジメント、現実的な目標、外部からの干渉や過負荷からの保護)
- 快適な労働条件(例えば、曖昧ではない仕様、成熟したテスト対象、適切に修正された欠陥、適切な職場、安定したテスト環境)。
- 最低限必要な安全性(例えば、安全な職場と合意事項の遵守)
- 良好な対人関係(例:同僚や上司との関係)

したがって、テストマネジメントは、モチベーションを下げる要因を継続的に排除し、同時にモチベーションを上げる要因を作り出し、強化すべきである。

さらに詳しい情報は、(Belbin, 2010)、(Marston, 1999)、(Kahler, 2008)で見つけることができる。

3.2 ステークホルダーとの関係

イントロダクション

テストマネジメントでは、よいビジネス価値を提供するためにテストを最適化することが重要である。過剰なテストは、利益を上回る不合理な遅延とコストをもたらす可能性がある。一方、不十分なテストは、ユーザーに低品質のプロダクトを提供することにつながる可能性がある。最適なアプローチは、この 2 つの間にある。このバランスと、このバランスを達成するためのテストの付加価値について、ステークホルダーに理解してもらうと同時に、例えばプロジェクトの典型的な時間の制約を念頭に置くことが、テストマネージャーの責任である。

3.2.1 品質コスト

テストの利点は、品質コストによって相殺される。品質関連の労力と欠陥の総コストを定量化する手段を品質コストと呼ぶ。品質コストでは、プロジェクトコストと運用コストを、プロダクトの欠陥コストに関連する 4 つのカテゴリーに分類する:

- **欠陥予防コスト:** 低品質を防ぐために計画的かつ積極的に行われるすべての活動のコスト(例えば、保守性を高めるまたはセキュアなコードを作成するためのトレーニング、テストベースの可能な限り早期のレビュー、チーム内での適切なコミュニケーションなど、開発担当者が自分のタスクが適格であることを確認するための活動、)。
- **評定(アプレイザル)コスト:** 欠陥検出を目的としたすべての活動のコスト(例えば、静的テストおよび動的テストの実施、作業成果物のレビュー)
- **内部失敗コスト:** すべての対処的な活動にかかるコスト(例えば、テスト中に発見した欠陥の修正、回避策の提供)。
- **外部失敗コスト:** 付加価値のないすべての活動および対処的な活動のコスト(例えば、収益、資産、人の健康、人命、または環境のいずれかの損失、欠陥のあるプロダクトを顧客へ提供したために発生する(リリース後の)欠陥の修正、テスト、デプロイ、サポートに関連する法的コスト。(顧客から報告された)フィールド欠陥の修正。)

評定コストと内部失敗コストの合計は、通常、外部失敗コストよりも大幅に少ない。そのため、テストは非常に価値のあるものとなる。これら 4 つのカテゴリーにおけるコストを決定することで、テストマネージャーはテストのための説得力のあるビジネスケースを作成することができる。

品質コストを定義するために考慮できるアプローチは他にもある。ISTQB® シラバスは、そのうちの 2 つをサポートしている。本シラバスは Feigenbaum のアプローチに基づいており、ISTQB® Foundation Level Syllabus V.4 は Boehm のアプローチを提示している(ISTQB® Foundation Level Syllabus V.4 1.3 節「テストの原則」を参照)。これら 2 つのアプローチは、品質コストについてより広範な理解を得るために選択したものである。Feigenbaum のアプローチ(Feigenbaum, Nov/Dec 1956)は、品質を顧客志向の全社的プロセスとして捉えている。Boehm のアプローチ(Boehm, 1979)は、ソフトウェア開発における予防コストと失敗コストのトレードオフに焦点を当てる (Hadjicostas, 2004).

3.2.2 テストの費用対効果の関係

ほとんどの組織が、ある意味でテストに価値があると考えているが、テストマネージャーを含め、その価値を定量化し、説明し、明確にできるマネージャーはほとんどいない。さらに、多くのテストマネージャー、テストリード、テスト担当者は、テストの運用面の詳細（すなわち、テストタスクやテストレベルに特化した側面）に焦点を当てる一方で、他のステークホルダー、特に他のマネージャー陣が関心を持つ、テストに関するより大きな戦術的、戦略的な（より高いレベル）の課題を無視している。

テストは、定量的および定性的な方法で、組織、プロジェクト、そして/または運用に対して利点をもたらす：

- **定性的な**利点には、品質に対する評判の向上、よりスムーズで予測可能なリリース、信頼性の向上、法的責任からの保護、ミッションあるいは人命の損失リスクの低減を含む。
- **定量的**な利点には、リリース前に発見した欠陥、予防した欠陥、修正された欠陥、リリース前に既知であることが判明した欠陥（すなわち、修正しなかったが文書化し、おそらく回避策がある）、費用対効果(Bohm 1981, Böhler 2008)、テスト実行によるリスクレベルの低減、プロジェクト、プロセス、プロダクトのステータスに関する情報の提供などを含む。

テストのさらなる利点には、すべてのステークホルダーが、欠陥の有無に関わらず、プロダクトの品質が本番稼動に十分かどうか判断するための十分な情報を得られることがある。既知の欠陥を抱えたままで本番稼動する方が、欠陥が解決するまで本番稼動を待つよりもはるかによい場合もある。このような欠陥が許容されるケースでは、欠陥の発生確率と重要度が大きく関係する。

テストの欠陥あたりの品質コストは以下のように計算する：

欠陥あたりの平均節約額＝外部失敗コストの平均 −（評定コストの平均＋内部失敗コストの平均）

総品質コスト＝（欠陥予防コスト + （評定コスト * リリース前に発見した欠陥数））+ （（内部失敗コスト * リリース前に発見した欠陥数）+ （外部失敗コスト * リリース後に発見した欠陥数））

例として、あるプロダクトの欠陥あたりの品質コストを次のように計算したとする：

- 欠陥予防コスト：180 ドル
- 欠陥 1 件あたりの評定コストの平均：500 ドル
- 欠陥 1 件あたりの内部失敗コストの平均：200 ドル
- 欠陥 1 件あたりの外部失敗コストの平均：4,000 ドル

平均の欠陥予防コスト、評定コスト、内部失敗コストは、リリース前に発見した欠陥数を用いて計算し、平均の外部失敗コストは、リリース後に発見した欠陥数を用いて計算される。これらの値を用いて、欠陥 1 件あたりの平均節約額を以下のように計算することができる：

欠陥 1 件あたりの平均節約額＝4,000 ドル-（500 ドル＋200 ドル）＝3,300 ドル

Boehm 曲線は、SDLC における時間経過に伴う欠陥の修正コストのグラフ表現である。このことから、欠陥の修正コストを削減するために、テストは SDLC の早い段階で実施されるべきであると結論付けられる。Boehm の曲線は、内部失敗のコスト、すなわち欠陥を修正するコストは、SDLC において欠陥が発見されるのが遅くなるほど増加する。この情報を利用して、テストマネージャーは、欠陥予防コスト対内部コストと外部コストの最適な関係を見つけるように努力すべきである。

テストの労力は、プロジェクトやプロダクトの具体的なリスクと、ビジネスが取ることをいとわないリスクに基づかなければならない。テストが多すぎると、リスクレベルを軽減する利点よりもコストが高くなる可能性がある。テストが少なすぎれば、見逃した欠陥が高いリスクとなり、省略したテストのコストよりも高いコストが発生する可能性がある。リスクベースドテスト（本シラバスの 1.3 節「リスクベースドテスト」参照）は、リスクレベルに比例したテストの労力を投入し、リスクレベルに基づいてテストの優先順位付けを行うことで、テストの費用対効果の関係をサポートする。

テストマネージャーは、自分たちの組織、プロジェクト、そして/または運用において、これらの利点とコストのどれが当てはまるかを理解し、これらの利点と欠陥あたりの品質コストという観点からテストの付加価値を伝えるようにすべきである。

4 参考文献

標準

- IEC 61508 (2010) Functional safety of electrical/electronic/programmable electronic safety-related systems - Parts 1 to 7（日本では JIS C 0508）
- ISO/IEC/IEEE 29119-2 (2021) Software and systems engineering - Software testing - Part 2: Test processes
- **I**ISO/IEC/IEEE 29119-3 (2021) Software and systems engineering - Software testing - Part 3: Test documentation

ISTQB® ドキュメント

- ISTQB® Certified Tester Agile Test Leadership at Scale Syllabus v2.0 (2023)
- ISTQB® Certified Tester Foundation Level Syllabus V.4 (2023)
（日本では「テスト技術者資格制度 Foundation Level シラバス Version 2023V4.0.J02」）
- ISTQB® Certified Tester Expert Level Test Management Syllabus v1.0 (2011)
- ISTQB® Certified Tester Expert Level Improving the Test Process Syllabus v1.0.1 (2011)

書籍（日本語翻訳が出版されているものは追記している）

- Basili, V., Trendowicz, A., Kowalczyk, M., Heidrich, J., Seaman, C., Münch, J., & Rombach, D. (2014). *Aligning Organizations Through Measurement – The GQM+ Strategies Approach.* Springer International.
- Bath, G., & van Veenendaal, E. (2014). *Improving the Test Process - chapter 6: Process for Improvement.* Rocky Nook.
- Belbin, R. M. (2010). *Management Teams: Why They Succeed or Fail.* London: Routledge.
- Black, R. (2009). *Managing the Testing Process, 3rd Edition.* John Wiley & Sons.「基本から学ぶテストプロセス管理,日経 BP」（日本語訳は旧版であるが掲載）
- Boehm, B. (1979). *Software Engineering Economics.* Prentice-Hall.
- Bonebright, D. A. (2010). *40 years of storming: a historical review of Tuckman's model of small group development* (1 Ausg., Bd. 13). Human Resource Development International, 1, 2010, Vol. 13.
- Craig, R., & Jaskiel, S. P. (2002). *Systematic Software Testing.* Artech House.「体系的ソフトウェアテスト入門,日経 BP」

- Derby, E., & Larsen, D. (2006). *Agile Retrospectives – Making Good Teams Great,.* The Pragmatic Bookshelf.「アジャイルレトロスペクティブズ — 強いチームを育てる「ふりかえり」の手引き,オーム社」
- Erpenbeck, J., & von Rosenstiel, L. (2017). *Handbuch Kompetenzmessung.* Stuttgart: Schäffer-Poeschel.
- Fowler, M. (2010). *Hybrid development processes.* IEEE Software, 27(2), 57-63.
- Herzberg, F., Mausner, B., & Bloch Snyderman, B. (1993). *Motivation to Work.* London: Routledge.「仕事と人間性—動機づけ-衛生理論の新展開, 東洋経済新報社」(日本語訳は旧版であるが掲載)
- Kahler, T. (2008). *The Process Therapy Model: The Six Personality Types with Adaptations.* Taibi Kahler Associates, Inc.
- Marston, W. M. (1999). *Emotions Of Normal People.* London: Routledge.
- Sonntag, K., & Schmidt-Rathjens, C. (2005). *Anforderungsanalyse und Kompetenzmodelle.* Wiesbaden: VS Verlag für Sozialwissenschaften.
- Tuckman, B. W. (1965). *Developmental sequence in small groups* (Bd. 63(6)). Psychological Bulletin.
- van Ewijk, A. (2013). *TPI NEXT – Business Driven Test Process Improvement,.* Sogeti Nederland B.「TPI NEXT ビジネス主導のテストプロセス改善. トリフォリオ」
- van Solingen, R., & Berghout, E. (1999). *The Goal Question Metric Method – A Practical Guide for Quality Improvement of Software Development.* McGraw-Hill.
- van Veenendaal, E. (2012). *The PRISMA Approach: Practical Risk-Based Testing.* UTN Publishers.
- van Veenendaal, E. (2020). *TMMi in the Agile world, version 1.4.* TMMi Foundation.
- van Veenendaal, E., & Cannegieter, J. J. (2011). *The Little TMMi – Objective-Driven Test Process Improvement.* UTN Publishers.

記事

- Feigenbaum, Armand V. (Nov/Dec 1956) Harvard Business Review, Vol. 34 Issue 6, p93-101
- Hadjicostas, Evsevios (2004) Total Quality Management and Cost of Quality, Springer https://link.springer.com/chapter/10.1007/978-3-662-09621-5_7.

Web ページ

- www.tmmi.org テスト成熟度モデル統合 (TMMi®); 最終訪問日 2024 年 1 月 31 日 ,
- www.tmap.net テストプロセス改善(TPI); 最終訪問日 :2024 年 1 月 31 日
- www.wikipedia.org/wiki/PDCA Plan-Do-Check-Act; 最終訪問日 :2024 年 1 月 31 日

以前の参考文献は、インターネットやその他の場所で入手可能な情報を指している。これらの参考文献は、本シラバスの発行時に確認されたものであるが、ISTQB® は、その参考文献が利用できなくなった場合、責任を負うことはできない。

5 付録 A - 学習目的／知識の認知レベル

本シラバスに適用する具体的な学習目的は、各章の冒頭に示されている。シラバスの各トピックは、その学習目的にしたがって検討される。

学習目的は、以下のように、知識の認知レベルに対応する動作動詞で始まる。

レベル 1：記憶する(K1)

用用語または概念を認識し、記憶して、想起することができる。

動作動詞：想起する(recall)、認識する(recognize)

例
テストピラミッドの概念を想起する。
典型的なテストの目的を認識する。

レベル 2：理解する(K2)

課題に関連する記述について理由または説明を選択することができ、テストコンセプトについて要約、比較、分類、例の提示を行うことができる。

動作動詞：分類する(classify)、比較する(compare)、区別する(differentiate)、区別する(distinguish)、説明する(explain)、例を挙げる(give examples)、解釈する(interpret)、要約する(summarize)

例	備考
テストツールを、その目的とサポートするテスト活動にしたがって分類する。	
テストレベルを比較する。	類似点、相違点、またはその両方を探すために使用できる。
テストとデバッグを区別する。	コンセプトの違いを探す。
プロジェクトリスクとプロダクトリスクを区別する。	2 つ（またはそれ以上）の概念を別々に分類できる。
テストプロセスにおけるコンテキストの影響を説明する。	
テストが必要な理由の例を挙げる。	
特定の故障プロファイルから欠陥の根本原因を推測する。	
作業成果物レビュープロセスの活動を要約する。	

レベル 3: 適用する (K3)

身近な課題に直面したときに手順を実行できる、または正しい手順を選択し、与えられたコンテキストに適用できる。

行動動詞:適用する(apply)、実装する(implement)、準備する(prepare)、使用する(use)

例	備考
与えられた要件からテストケースを導き出すために、境界値分析を適用する。	プロシジャー、技法、プロセスなどを指す。
技術的、経営的要件をサポートするメトリクス収集方法を実装する。	
モバイルアプリの設置性テストを準備する。	
テスト進捗のモニタリングをして、テスト目的、テスト戦略、テスト計画との完全性と整合性を確認するためにトレーサビリティを使用する	受験者に技法やプロシジャーの使用を求める LO で使用できる。「適用する」に似ている。

レベル 4 : 分析する(K4)

プロシジャーや技法に関連する情報を、理解を深めるために構成要素に分離し、事実と推論を区別することができる。典型的な用途は、文書、ソフトウェア、プロジェクトの状況を分析し、問題解決やタスク完了のための適切な行動を提案することである。

行動動詞:分析する(analyze)、分解する(deconstruct)、概略を描く(outline)、優先順位を付ける(prioritize)、選択する(select)

例	備考
指定したゴールの達成に向けてどのブラックボックステスト技法(black-box)または経験ベースのテスト技法を適用すべきかを判断するため、特定のプロジェクトの状況を分析する	測定可能な分析のゴールと組み合わせてのみ審査可能である。 「xxxx から xxxx を分析する」(または類似の)形式でなければならない。
関連するプロダクトリスクに基づき、実行するテストスイート内のテストケースに優先順位を付ける。	
与えられた一連の要件を検証するために、適切なテストレベルとテストタイプを選択する。	選択が分析を必要とする場合に必要となる。

参考

(学習の目的の認知レベルについて)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing:

A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon.

本シラバスに適用される具体的な学習の目的は、各章の冒頭に示されている。

6 付録 B-学習の目的とビジネス成果のトレーサビリティマトリクス

この節では、**Advanced Level Test Management** のビジネス成果と学習の目的とのトレーサビリティを示す。

ビジネス成果:Advanced Level Test Management		BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM_01	プロジェクトチームまたはテスト組織のために確立されたテストマネジメントプロセスを適用し、さまざまなソフトウェア開発プロジェクトのテストをマネジメントする。	12										
TM_02	特定のコンテキストで関連するテストのステークホルダーとソフトウェア開発ライフサイクルモデルを識別する。		4									
TM_03	あらゆるソフトウェア開発ライフサイクルにおいてリスク識別とリスクアセスメントセッションを企画し、結果をテスト目的達成に向けてテストの指針として活用する。			6								
TM_04	組織的テスト戦略およびプロジェクトのコンテキストと合致するプロジェクトテスト戦略を定義する。				11							
TM_05	プロジェクトゴールを達成するためにテストを継続的にモニタリングおよびコントロールする。					4						
TM_06	テストの進捗状況をアセスメントし、プロジェクトのステークホルダーに報告する。						3					
TM_07	必要なスキルを特定し、チーム内でそのスキルを開発する。							6				
TM_08	さまざまなコンテキストにおけるテストのコストと期待される利点の概要を示したビジネスケースを準備し、提示する。								5			

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM_09	プロジェクトやソフトウェア開発のプロダクトストリームでテストプロセス改善活動をリードし、組織のテストプロセス改善のための施策に貢献する。										5		
TM_10	必要なテストインフラを含むテスト活動を計画し、テストに必要な労力を見積る。											9	
TM_11	ソフトウェア開発ライフサイクルに適した欠陥レポートおよび欠陥ワークフローを作成する。												6
LO 個別	学習目的	K レベル											
1	テスト活動のマネジメント												
1.1	テストプロセス												
TM-1.1.1	テスト計画を要約する。	K2	X			X							
TM-1.1.2	テストモニタリングとテストコントロールを要約する。	K2	X				X						
TM-1.1.3	テスト完了を要約する。	K2	X					X					
1.2	テストのコンテキスト												
TM-1.2.1	さまざまなステークホルダーがテストに関心を持つ理由を比較する。	K2		X		X							
TM-1.2.2	テストマネジメントにおいてステークホルダーの知識が重要である理由を説明する。	K2		X		X							
TM-1.2.3	ハイブリッドソフトウェア開発モデルで行うテストについて説明する。	K2		X		X							

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-1.2.4	さまざまなソフトウェア開発ライフサイクルでのテストマネジメント活動を要約する。	K2	X	X		X							
TM-1.2.5	さまざまなテストレベルでのテストマネジメント活動を比較する。	K2	X			X							
TM-1.2.6	さまざまなテストタイプのテストマネジメント活動を比較する。	K2	X			X							
TM-1.2.7	テスト計画、テストモニタリング、テストコントロールに重点を置いたテストマネジメント活動を決めるために、特定のプロジェクトを分析する。	K4	X			X							
1.3	リスクベースドテスト												
TM-1.3.1	リスクベースドテストにてリスクに対応するために取るさまざまな手段を説明する。	K2			X								
TM-1.3.2	テストマネージャーがプロダクト品質に関連するリスクを識別するために使用できるさまざまな技法の例を挙げる。	K2			X								
TM-1.3.3	プロダクト品質に関するリスクレベルを決定する要因について要約する。	K2			X								
TM-1.3.4	特定のコンテキストにおけるリスクレベルに応じたリスク軽減するための適切なテスト活動を選択する。	K4			X								
TM-1.3.5	リスクベースドテスト技法について重量的と軽量的の例を区別する。	K2			X								
TM-1.3.6	リスクベースドテストに関連する成功メトリクスと困難さの例を挙げる。	K2			X								

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
1.4	プロジェクトテスト戦略												
TM-1.4.1	テストアプローチの典型的な選択肢を説明する。	K2				X							
TM-1.4.2	組織的テスト戦略とプロジェクトのコンテキストを分析し、適切なテストアプローチを選択する。	K4				X							
TM-1.4.3	測定可能なテスト目的と終了基準を定義するために、スマートの法則を使用する。	K3				X							
1.5	テストプロセス改善												
TM-1.5.1	特定のプロジェクトにおけるテストプロセス改善のための IDEAL モデルの使用方法を説明する。	K2									X		
TM-1.5.2	テストプロセス改善のためのモデルに基づく改善アプローチを要約し、プロジェクトのコンテキストに合わせて適用する方法を理解する。	K2									X		
TM-1.5.3	テストプロセス改善のための分析に基づく改善アプローチを要約し、プロジェクトのコンテキストに合わせて適用する方法を理解する。	K2									X		
TM-1.5.4	テストプロセスを評価し、改善すべきテスト領域を発見するためにプロジェクトまたはイテレーションにふりかえりを実装する。	K3									X		
1.6	テストツール												
TM-1.6.1	ツール導入の最良の実践例を要約する。	K2										X	
TM-1.6.2	ツールの種類を決定する際にさまざまな技術的側面とビジネス的側面へ与える影響について説明する	K2										X	

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-1.6.3	ツールの選択、カバーできるリスク、コスト、および利点を考慮した計画を立てるために特定の状況を分析する。	K4										X	
TM-1.6.4	ツールのライフサイクルの各段階を区別する。	K2										X	
TM-1.6.5	ツールを利用したメトリクス収集と評価の例を挙げる。	K2									X	X	
2	プロダクトのマネジメント												
2.1	テストメトリクス												
TM-2.1.1	テスト目的を達成するためのメトリクスの例を挙げる。	K2					X						
TM-2.1.2	テストメトリクスを用いてテストの進捗コントロールする方法を説明する。	K2					X						
TM-2.1.3	ステークホルダーの意思決定に役立つテストレポートを作成するためにテスト結果を分析する。	K4					X	X					
2.2	テスト見積り												
TM-2.2.1	テスト見積りにおいて考慮すべき要因について説明する。	K2	X							X		X	
TM-2.2.2	テスト見積りに影響を与える要因の例を挙げる。	K2	X							X		X	
TM-2.2.3	特定のコンテキストでテスト見積りのための適切な技法またはアプローチを選択する。	K4	X							X		X	
2.3	欠陥マネジメント												

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-2.3.1	欠陥のモニタリングとコントロールに使用できる欠陥ワークフローを含め、欠陥マネジメントプロセスを実装する。	K3											X
TM-2.3.2	効果的な欠陥マネジメントに必要なプロセスと参加者を説明する。	K2											X
TM-2.3.3	アジャイルソフトウェア開発における欠陥マネジメントについて詳細に説明する。	K2	X										X
TM-2.3.4	ハイブリッドソフトウェア開発における欠陥マネジメントの課題について説明する。	K2	X										X
TM-2.3.5	欠陥マネジメントで収集すべきデータと分類情報を利用する。	K3											X
TM-2.3.6	欠陥レポートの統計情報がプロセス改善の考案にどのように利用できるか説明する。	K2										X	X
3	チームのマネジメント												
3.1	テストチーム												
TM-3.1.1	テストチームメンバーが必要とされる典型的なスキルについて、4 つの能力の領域から例を挙げる。	K2							X				
TM-3.1.2	テストチームメンバーに要求されるスキルを判断するため、与えられたプロジェクトのコンテキストを分析する。	K4							X				
TM-3.1.3	テストチームメンバーのスキルアセスメントの典型的な技法について説明する。	K2							X				

ビジネス成果:Advanced Level Test Management			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8	BO9	BO10	BO11
TM-3.1.4	テストチームメンバーのスキルを向上させるための典型的なアプローチを区別する。	K2							X				
TM-3.1.5	テストチームのマネジメントに求められるスキルを説明する。	K2							X				
TM-3.1.6	テストチームメンバーのモチベーションを上げる要因と衛生要因の例を挙げる。	K2							X				
3.2	ステークホルダーとの関係												
TM-3.2.1	品質コストを決定する 4 つのカテゴリーについて、それぞれの例を挙げる。	K2								X			
TM-3.2.2	ステークホルダーにとってのテストの付加価値を見積るために、費用対効果の計算を適用する。	K3						X		X			

7 付録 C - リリースノート

ISTQB® Advanced Level Test Management シラバス v3.0 は、Advanced Level Syllabus Test Manager 2012 に基づくメジャーアップデート版である。そのため、章や節ごとの詳細なリリースノートはないが、主要な変更点の要約を以下に示す。

本バージョンでは、すべての学習目的(LO)はそれだけで 1 つのまとまりとなるように編集されており、LO とシラバスの節の間に 1 対 1 のトレーサビリティを確立した。LO がなければコンテンツもないようにしている。また、より読みやすく、理解しやすく、学びやすく、翻訳しやすくすることをゴールとし、実践的な有用性を高め、知識と技能のバランスを重視している。

今回のメジャーリリースの変更は、以下の通り:

- シラバス全体のサイズダウンをした。シラバスは教科書ではなく、ソフトウェアテストのアドバンスコースにおいて、どのようなトピックをどのようなレベルで取り上げるべきかなど、基本的な要素をまとめたドキュメントである。したがって、特に次のように変更した。
  - トレーニング中に例題や練習問題を提供するのは、トレーニング機関の仕事であるため、ほとんどの例題をテキストから除外した。
  - K レベルごとに LO の最大ドキュメント量を提示している、「シラバス作成チェックリスト」にしたがった。(K2＝最大 1,500 文字(空白文字以外)、K3＝最大 2,500 文字(空白文字以外)、K4＝最大 3,000 文字(空白文字以外)、±20%)
- CTAL TM 2012 シラバスと比較して LO 数を削減した。
  - K2 の LO は CTAL TM 2012 の LO は 39 だったのに対して 36 となる。
  - K3 の LO は CTAL TM 2012 の LO は 12 だったのに対して 5 つとなる。
  - K4 の LO は CTAL TM 2012 の LO は 10 だったのに対して 7 つとなる。
- シラバスの構成を全面的に見直した
- ISTQB® Foundation Level Syllabus V.4 と全面的に整合している。
- 第 1 章(テストプロセス)の主な変更点
  - テスト活動のマネジメント(テスト計画、テストモニタリング、テストコントロール、テスト完了)に限定した。
  - 新設した章「テスト活動のマネジメント」の 1 つの節として統合した。
- 新設章 **テスト活動のマネジメント**
  - 1.1 節 テストプロセス: 上記参照
  - 1.2 節 テストのコンテキスト:シーケンシャルではないソフトウェア開発モデルをカバーするために拡張した。
  - 1.3 節 リスクベースドテスト:プロジェクトレベルでより適用できるように全面的に書き直した。

- 1.4 節 プロジェクトテスト戦略：テスト計画書は、ISTQB Foundation Level Syllabus V.4 ですでに定義されているため、適切なテストアプローチの選択と、測定可能なテスト目的をどのように定義するかに焦点を当てた。
- 1.5 節 テストプロセス改善：プロセス改善をテスト活動のマネジメントに統合し、プロジェクトのコンテキストへ適用する方法を示し、イテレーションまたはプロジェクト内でのふりかえりを使用して実装した。
- 1.6 節 テストツール：ISTQB® Foundation Level Syllabus V.3.1 からツールの紹介を移動した（ISTQB® Foundation Level Syllabus V.4 にはない）。

- 新設章 **プロダクトのマネジメント**
  - 2.1 節 テストメトリクス：テストメトリクスについて、以前の節で定義されたメトリクスとメトリクスの使用。
  - 2.2 節 テスト見積り：ISTQB® Foundation Level Syllabus V.4 は、テスト見積りの計算をすでにカバーしている。K4 レベルとして、適切なテスト見積り技法と SDLC モデル全体にわたったテスト見積りの使用を選択するように拡張した。
  - 2.3 節 欠陥マネジメント：標準の最新版に合わせ、アジャイルソフトウェア開発とハイブリッドソフトウェア開発で使用できるように拡張した。
- 新設章 **チームのマネジメント**
  - 3.1 節 テストチーム：主なトピックは TM2012 シラバスと同じである。個々のスキルを識別し、テストチームを構成した。
  - 3.2 節 ステークホルダーとの関係：旧 ATM 2012 シラバスの「テストのビジネス価値」と題した節である。
- CTAL TM シラバス 2012 の主な変更点および削除された節/章
  - 分散テスト、アウトソーステスト、インソーステストに関する節は削除した。
  - 業界標準の適用に関する章は削除した。
  - レビューの章は削除した。
  - テストプロセス改善の CTP と STEP に関する節は削除した。
  - テスト分析、テスト設計、テスト実装、テスト実行に関する節は削除した。

8 付録 D - ドメイン固有のキーワード

用語名	定義
ゴール クエスチョン メトリクス (GQM)	概念レベル(ゴール)、運用レベル(クエスチョン)、定量レベル(メトリクス)の 3 レベルモデルを用いたソフトウェア測定のアプローチである。
IDEAL	改善行動を開始し、計画し、実施するためのロードマップとなる組織改善モデル。
指標	定義された情報ニーズに関して、モデルから得られる特定の属性の推定または評価を提供する尺度。
尺度	測定によって、あるエンティティの属性に割り当てられる数値または分類。
メトリクス	測定尺度と測定方法。
プランニングポーカー	コンセンサスに基づく見積り手法の 1 つで、主にアジャイルソフトウェア開発においてユーザーストーリーの労力や相対的なサイズを見積るために使用する。ワイドバンドデルファイ法のバリエーションで、チームが見積る単位を表す値を持つカードの山を使用する。
三点見積り	専門家ベースの手法で、専門家によって 3 つの推定が行われる:最も楽観的な推定値(a)、最も可能性の高い推定値(m)、最も悲観的な推定値(b)。最終的な推定値 (E) は、それらの加重算術平均である。
ワイドバンドデルファイ	専門家ベースのテスト見積り技法で、チームメンバーの知恵を結集して正確な見積りを行うことを目的とする。

9 付録 E - 商標

CMMI® は、カーネギーメロン大学の米国特許商標庁における登録商標である。

ISTQB® は、International Software Testing Qualifications Board の登録商標である。

TMMi® は TMMi Foundation の登録商標である。

TPI-Next® はオランダ Sogeti 社の登録商標である。