

テスト技術者資格制度 テスト自動化戦略スペシャリスト シラバス 日本語版

Version 1.0 J01

International Software Testing Qualifications Board



著作権について

Copyright Notice © International Software Testing Qualifications Board (以降では ISTQB® と表記)

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

Copyright © 2024 Test Automation Strategy v1.0 シラバスの著者： Andrew Pollner (Chair) , Péter Földházi, Patrick Quilter, Gergely Ágnes, László Szikszai.

無断転載を禁じる。著者はここに著作権を ISTQB® に譲渡します。著者（現在の著作権者）と ISTQB®（将来の著作権者）は、本シラバスの利用条件として以下に合意している：

- 非営利目的で使用する本書の抜粋は、出典が明記される場合に限り、複製可能です。認定トレーニングプロバイダーは、著者と ISTQB® がシラバスの出典および著作権者として明示されることを条件として、本シラバスをトレーニングコースの基礎として使用できます。ただし、当該トレーニングコースの広告においては、ISTQB® 認定メンバーボードから正式な認定を受けた後のみ、本シラバスに言及することができる。
- 個人または団体は、著者と ISTQB® がシラバスの出典および著作権者として明示されることを条件として、本シラバスを記事や書籍の基礎として使用できる。
- ISTQB® による事前の書面による承諾を得ることなく、本シラバスをその他の目的で使用することを禁止する。
- すべての ISTQB® 認定メンバーボードは、翻訳版のシラバスに上記著作権表示を再掲することを条件として、本シラバスを翻訳できる。

Translation Copyright © 2026, Japan Software Testing Qualifications Board (JSTQB®), all rights reserved.

日本語翻訳版の著作権は JSTQB® が有する。本書の全部、または一部を無断で複製し利用することは、著作権法の例外を除き、禁じられている。

改訂履歴

ISTQB®

バージョン	日付	備考
Syllabus v1.0	2024/05/03	CT-TAS v1.0 GA リリース

JSTQB®

バージョン	日付	備考
v1.0 J01	2026/05/01	Version 1.0 の日本語翻訳版

目次

著作権について	2
改訂履歴	3
目次	4
謝辞	7
0 はじめに	8
0.1 本シラバスの目的	8
0.2 ソフトウェアテストにおけるテスト自動化戦略	8
0.3 テスト担当者およびテスト自動化エンジニア (TAE) のキャリアパス	9
0.4 ビジネス成果	10
0.5 試験対象の学習の目的および知識の認知レベル	11
0.6 テスト自動化戦略スペシャリスト認定試験	11
0.7 認定	12
0.8 標準の取り扱い	12
0.9 最新情報の維持	12
0.10 詳細レベル	12
0.11 本シラバスの構成	13
1 テスト自動化戦略の導入と目標 – 45 分 (K2)	15
1.1 テスト自動化プロジェクトの成功要因	16
1.1.1 テスト自動化戦略の目標と目的の定義	16
1.1.2 テスト自動化プロジェクトの技術的成功要因の特定	16
1.1.3 テスト自動化の候補プロジェクト選定における適切な投資基準の要約	17
2 テスト自動化リソース – 60 分 (K2)	18
2.1 テスト自動化ソリューション (TAS) 導入のコストとリスク	19
2.1.1 総保有コストの観点から代替技術ソリューションを比較する	19
2.1.2 テスト自動化ツールのライセンスモデルに関する考慮事項の説明	19
2.1.3 テスト自動化戦略定義時に考慮すべき要素の例示	20
2.2 テスト自動化における役割と責任	21
2.2.1 テスト自動化ソリューション (TAS) の成功に必要な役割とスキルの要約	21

3 テスト自動化の準備 – 225 分 (K3)	22
3.1 テストレベルをまたいだ統合	23
3.1.1 テスト自動化の配分の区別	23
3.1.2 テスト対象システム (SUT) のアーキテクチャに基づくテスト自動化アプローチの選択	24
3.1.3 シフトレフトおよびシフトライトアプローチを達成するためのテスト自動化配分の最適化方法の実証	24
3.2 異なるソフトウェア開発ライフサイクル (SDLC) モデルにおける戦略的考慮事項	25
3.2.1 レガシーソフトウェア開発ライフサイクル (SDLC) モデルへのテスト自動化プロジェクトの適合方法の説明	25
3.2.2 テスト自動化をサポートするアジャイルソフトウェア開発ベストプラクティスへのテスト自動化プロジェクトの適合方法の説明	26
3.2.3 継続的テストを達成するための DevOps ベストプラクティスへのテスト自動化プロジェクトの準備	26
3.3 テスト自動化の適用可能性と実現可能性	27
3.3.1 テスト自動化に適したテストの適合性判断基準の説明	27
3.3.2 テスト自動化のみが対処できる課題の特定	27
3.3.3 自動化が困難なテスト条件の特定	28
4 テスト自動化の組織的デプロイメントとリリース戦略 – 135 分 (K2)	29
4.1 テスト自動化ソリューション (TAS) の計画	30
4.1.1 テスト自動化が市場投入期間を短縮する方法の特定	30
4.1.2 テスト自動化が報告された欠陥の検証に役立つ方法の特定	30
4.1.3 テスト自動化のための運用関連シナリオの開発を可能にするアプローチの定義	31
4.2 テスト自動化デプロイメント戦略	32
4.2.1 テスト自動化デプロイメント戦略の定義	32
4.2.2 デプロイメントにおけるテスト自動化リスクの特定	33
4.2.3 デプロイメントリスクを軽減するアプローチの定義	34
4.3 テスト環境内の依存関係	35
4.3.1 テスト環境内のテスト自動化コンポーネントの定義	35
4.3.2 テスト自動化のインフラストラクチャコンポーネントと依存関係の特定	35
4.3.3 テスト自動化データとインターフェース要件の定義	36
5 テスト自動化影響分析 – 150 分 (K3)	37

5.1 テスト自動化の構築・維持への投資	38
5.1.1 テスト自動化ソリューション（TAS）構築の投資対効果（ROI）の算出	38
5.2 テスト自動化メトリクス	39
5.2.1 テスト自動化メトリクスの分類	39
5.3 プロジェクトおよび組織レベルにおけるテスト自動化の価値	41
5.3.1 テスト自動化利用に関する組織的考慮事項の特定	41
5.3.2 テスト自動化目標の最適な実装を判断するためのプロジェクト特性の分析	42
5.4 テスト自動化レポートからの意思決定	43
5.4.1 意思決定を支援するテストレポートデータの分析	43
6 テスト自動化の実装と改善戦略 – 150 分（K3）	45
6.1 手動テストから継続的テストへの移行活動	46
6.1.1 手動テストからテスト自動化への移行における要因と計画活動の説明	46
6.1.2 テスト自動化から継続的テストへの移行における要因と計画活動の説明	47
6.2 組織全体のテスト自動化戦略	48
6.2.1 テスト自動化資産とプラクティスの評価による改善領域の特定	48
参考文献	50
付録 A – 学習の目的／知識の認知レベル	53
付録 B – 学習の目的とのビジネス成果トレーサビリティマトリックス	55
付録 C – リリースノート	59
付録 D – ドメイン固有用語	60

謝辞

本文書は ISTQB®の総会により、2024 年 5 月 3 日に正式に発行された。

本文書は International Software Testing Qualifications Board の Specialist Working Group の Test Automation Task Force によって作成されました：Graham Bath（Specialist Working Group 委員長）、Andrew Pollner（Specialist Working Group 副委員長兼 Test Automation Task Force 委員長）、Péter Földházi、Patrick Quilter、Gergely Ágneecz、László Szikszai。レビューアー：Armin Beer、Armin Born、Geza Bujdosó、Renzo Cerquozzi、Jan Giesen、Arnika Hryszko、Kari Kakkonen、Gary Mogyorodi、Chris van Bael、Carsten Weise、Marc-Florian Wendland。

技術レビューアー：Gary Mogyorodi

以下の方々が本シラバスのレビュー、コメント、および投票に参加した：

Horváth Ágota、Laura Albert、Prasunkumar Banerjee、Jürgen Beniermann、Armin Born、Piet de Roo、Nicola De Rosa、Dingguofu Ding Guofu、Elizabeta Fournieret、Jan Giesen、Erik Haartmans、Matthias Hamburg、Tobias Horn、Mattijs Kemmink、Ilia Kulakov、Ashish Kumar、Vincenzo Marrazzo、Marton Matyas、Patricia McQuaid、Smitha Mohandas、Ingvar Nordström、Sreeja Padmakumari、Nishan Portoyan、Meile Posthuma、Swapnil Shah、Péter Sótér、Szilard Szell、Richard Taylor、Giancarlo Tomasig、Chris Van Bael、Daniel Van der Zwan、Carsten Weise、Marc-Florian Wendland、Claude Zhang。

日本語訳については、Japan Software Testing Qualifications Board メンバーおよび以下の日本語翻訳ワーキンググループメンバーにより行われた。

日本語翻訳ワーキンググループメンバー：

江村 禎昭（バルテス・ホールディングス）

山口 鉄平（LayerX）

末村 拓也（Ubie）

岡野 誠（Autify）

永田 哲（ASTER）

0 はじめに

0.1 本シラバスの目的

本シラバスはテスト自動化戦略スペシャリスト資格のベースとなる。ISTQB®は、本シラバスを以下の目的で提供する。

1. メンバー委員会に対しては、各国語への翻訳および教育機関の認定の目的で提供する。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、出版事情に合わせてリファレンスを修正することができる。
2. 認定委員会に対して、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. 教育機関に対して、コースウェアを作成し、適切な教育方法を確定できるようにする目的で提供する。
4. 受験志願者に対して、試験準備の目的で提供する（研修コースの一部として、または独立した形態で実施）。
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、ソフトウェアやシステムをテストする技能の向上を目的とする他、書籍や記事を執筆する際の参考として提供する。

0.2 ソフトウェアテストにおけるテスト自動化戦略

テスト自動化戦略スペシャリスト資格は、ソフトウェアテストおよびテスト自動化に携わるすべての人を対象としている。テスト担当者、テストアナリスト、テスト自動化エンジニア（TAE）、テストコンサルタント、テストアーキテクト、テストマネージャー、ソフトウェア開発者などの役割を担う人が含まれる。本スペシャリスト資格は、テスト自動化の基本的な理解を求める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にも適している。

テスト自動化戦略スペシャリスト（CT-TAS）シラバスは、組織内でテスト自動化を計画する際に考慮すべき複数の要因を示す。テスト自動化の手法とベストプラクティスの技術的な実装の側面は、別のISTQB CTAL-TAE シラバスで扱われているため、本シラバスの対象外である。

テスト自動化戦略は、技術的なツール実装と統合の課題を超えたテスト自動化のニーズに対応する。テスト自動化の戦略的な視点は、組織内の複数のプロジェクトにわたって体系的かつ一貫した方法で実装するビジョンを提供し、最終的に組織への価値を示す。

0.3 テスト担当者およびテスト自動化エンジニア (TAE) のキャリアパス

ISTQB®スキームは、広範な知識と深い知識の両方を提供することで、テスト専門家のキャリアのすべての段階を支援する。ISTQB® テスト自動化戦略スペシャリスト認定を取得した人は、テスト自動化エンジニアリング (CTAL-TAE) 資格へのステップアップも検討できる。

ISTQB® テスト自動化戦略スペシャリスト認定を取得した人は、**Core Advanced Levels** (テストアナリスト、テクニカルテストアナリスト、テストマネージャー)、その後 **Expert Levels** (テストマネジメントまたはテストプロセス改善) へのステップアップも検討できる。アジャイル環境でのテストプラクティスのスキルを開発したい人は、アジャイルテクニカルテスト担当者またはスケールされたアジャイルテストリーダーシップの認定を検討できる。スペシャリスト系は、テスト自動化エンジニアリング、性能テスト、セキュリティテスト、AI テスト、モバイルアプリケーションテストなど特有のテストアプローチと活動を持つ分野、または自動車ソフトウェアテスト、ゲームテストなどのドメイン固有の知識が必要な分野を深掘りできる。ISTQB®認定テスト担当者スキームの最新情報は www.istqb.org を確認のこと。

0.4 ビジネス成果

本セクションでは、テスト自動化戦略スペシャリスト認定を取得した受験者に期待されるビジネス成果を示す。

テスト自動化戦略スペシャリストは以下が可能になる。

TAS-B01	テスト自動化の成功に影響を与えるソフトウェアおよびシステムからの要因を理解する
TAS-B02	テスト自動化ソリューション（TAS）導入のコストとリスクを特定する
TAS-B03	テスト自動化に貢献する人々の役割と責任を理解する
TAS-B04	テストレベルをまたいだテスト自動化の統合を計画する
TAS-B05	異なるソフトウェア開発ライフサイクル（SDLC）モデルにおけるテスト自動化実装の戦略的考慮事項を特定する
TAS-B06	テスト自動化の適用可能性と実現可能性を理解する
TAS-B07	組織のニーズを満たすテスト自動化ソリューション（TAS）を計画する
TAS-B08	テスト自動化デプロイメント戦略を理解する
TAS-B09	テスト環境内のテスト自動化の依存関係を理解する
TAS-B10	テスト自動化の構築・維持コストを理解する
TAS-B11	意思決定を促進するテスト自動化メトリクスを習得する
TAS-B12	テスト自動化がプロジェクトおよび組織にもたらす価値を特定する
TAS-B13	ステークホルダーのニーズに対応するテスト自動化レポート要件を特定する
TAS-B14	手動テストからテスト自動化への移行活動を定義する
TAS-B15	組織全体で一貫した実装を確保するためにプロジェクトが資産と手法を共有するテスト自動化戦略を定義する

0.5 試験対象の学習の目的および知識の認知レベル

学習の目的はビジネス成果を支援するものであり、テスト自動化戦略スペシャリスト認定試験の作成に使用される。

本シラバスの内容は、はじめにと付録を除き、原則として **K2** および **K3** レベルで試験対象となる。受験者は **6** 章のいずれかで言及されているキーワードまたは概念を認識、記憶、または想起することが求められる場合がある。各章の冒頭に学習の目的レベルが示されており、以下のように分類される：

- **K2**：理解
- **K3**：適用

学習の目的の詳細と例は付録 **A** に記載されている。

章見出しの直下にキーワードとして記載されているすべての用語は、学習の目的に明示されていなくても記憶しなければならない。

0.6 テスト自動化戦略スペシャリスト認定試験

テスト自動化戦略スペシャリスト認定試験は本シラバスに基づいて実施される。試験問題の解答には、本シラバスの複数のセクションに基づく内容が必要な場合がある。本シラバスのすべてのセクションが試験対象であるが、はじめにと付録は除く。標準や書籍は参考文献として含まれているが、その内容はシラバス自体でまとめられている範囲を超えて試験対象とはならない。

テスト自動化戦略スペシャリスト認定試験の詳細については、「**Exam Structures and Rules v1.1 Compatible with Syllabus Foundation and Advanced Levels and Specialist Modules**」文書を参照すること。

テスト自動化戦略スペシャリスト認定試験の受験基準は、受験者がソフトウェアテストおよびテスト自動化に関心を持っていることである。ただし、受験者が以下を満たすことを強く推奨する：

- 企業へのテクノロジー実装のためのソフトウェアおよびシステム開発とリーダーシップに関する最低限の背景知識、およびシニアテストエンジニア、テストリード、またはソフトウェア開発者としての経験を持つこと
- **ISTQB** 認定メンバーボードのいずれかによって **ISTQB** 標準に準拠して認定されたコースを受講すること。

受験要件に関する注意：**ISTQB® Foundation** レベル認定を取得した後に、テスト自動化戦略スペシャリスト認定試験を取得していなければならない。

0.7 認定

ISTQB®メンバーボードは、本シラバスに準拠したコース教材を持つトレーニングプロバイダーを認定することができる。トレーニングプロバイダーは認定を実施するメンバーボードまたは団体から認定ガイドラインを取得すること。認定コースは本シラバスに準拠するものとして認められ、ISTQB®試験をコースの一部として含めることが許可される。

本シラバスの認定ガイドラインは、Processes Management and Compliance Working Group が発行する一般認定ガイドラインに準拠している。

0.8 標準の取り扱い

テスト自動化戦略スペシャリストシラバスには標準が参照されている（IEEE、ISO など）。これらの参照の目的は、フレームワークを提供すること（品質特性に関する ISO 25010 への参照など）または読者が必要とする場合に追加情報のソースを提供することである。シラバスは標準文書を参照として使用していることに注意されたい。標準文書自体は試験対象ではない。標準の詳細は第 7 章の参考文献を参照すること。

0.9 最新情報の維持

ソフトウェア業界は急速に変化している。これらの変化に対応し、関係者が関連する最新の情報にアクセスできるように、ISTQB ワーキンググループは、www.istqb.org ウェブサイトにリンクを設け、サポートドキュメントや標準の変更点などを紹介している。これらの情報は、テスト自動化戦略スペシャリストのシラバスでは試験対象外である。

0.10 詳細レベル

本シラバスの詳細レベルは、国際的に一貫したコースと試験を可能にする。この目標を達成するために、シラバスは以下で構成されている：

- テスト自動化戦略スペシャリストの意図を説明する一般的な教育目標
- 学習者が想起できなければならない用語のリスト
- 各知識領域の学習の目的（達成すべき認知的学習成果を説明するもの）
- 受け入れられた文献や標準などの出典への参照を含む主要概念の説明

シラバスの内容は、ソフトウェアテストの知識領域全体を説明するものではない。テスト自動化戦略スペシャリストのトレーニングコースでカバーされる詳細レベルを反映している。アジャイル手法に従うものを含む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概念と技術に焦点を当てている。本シラバスにはアジャイルテストに関する特定の学習の目的は含まれていないが、これらの概念がアジ

ャイルプロジェクトやその他の種類のプロジェクトにどのように適用されるかについては説明している。

0.11 本シラバスの構成

試験対象のコンテンツを含む **6** つの章がある。各章のトップレベル見出しに章の時間が指定されている。章レベル以下の時間は示されていない。認定トレーニングコースでは、シラバスは **6** つの章にわたって分配された最低 **12.75** 時間の指導を必要とする：

- **第 1 章：45 分** – テスト自動化戦略の導入と目標
 - テスト担当者はテスト自動化の概念を理解し、候補プロジェクトの選定基準を学ぶ
 - テスト担当者はテスト自動化の成功実装を定義する要因を理解する
- **第 2 章：60 分** – テスト自動化リソース
 - テスト担当者はテスト自動化に利用可能なさまざまなソリューションとそれぞれの相対的な投資を学ぶ
 - テスト自動化ツールのソフトウェアライセンスについて扱う
 - テスト担当者はテスト自動化に必要なスキルを理解する
- **第 3 章：225 分** – テスト自動化の準備
 - テスト担当者はテストレベルをまたいで、また各テストレベル内でテスト自動化がどのように使用されるかを学ぶ
 - テストを適切に配分し、シフトレフトおよびシフトライトを達成するためのテスト自動化戦略について扱う
 - テスト担当者はテスト自動化がレガシーおよびアジャイルプロジェクトをどのようにサポートするかを学ぶ
 - **DevOps** 内でのテスト自動化と継続的テストプラクティスについて扱う
 - テスト担当者はテスト自動化に最も適したテストを含む、テスト自動化使用基準の定義方法を理解する
- **第 4 章：135 分** – テスト自動化の組織的デプロイメントとリリース戦略
 - テスト担当者はテスト自動化が市場投入期間をどのように改善できるかを学ぶ
 - テスト担当者は運用に関連する自動テストの開発方法と欠陥のレポート方法を理解する
 - テスト自動化デプロイメント戦略とリスク軽減について扱う
 - テスト担当者はテスト自動化環境とその依存関係について学ぶ
 - テスト自動化とテストデータのテスト対象システム（SUT）への統合について扱う

- 第 5 章 : 150 分 – テスト自動化影響分析
 - 意思決定を支援するためのテスト自動化メトリクスとレポートについて扱う
 - テスト担当者はテスト自動化の投資対効果 (ROI) を計算する方法を学ぶ
 - テスト自動化を使用する組織とプロジェクトの目標について扱う
 - テスト担当者はテストレポートを分析し、明確でわかりやすい方法で意思決定者に情報を提供する方法を学ぶ
- 第 6 章 : 150 分 – テスト自動化の実装と改善戦略
 - テスト担当者は手動テストからテスト自動化、継続的テストへの移行方法を学ぶ
 - 継続的改善のためのテスト自動化の評価について扱う

1 テスト自動化戦略の導入と目標 – 45 分 (K2)

キーワード

テスト自動化アーキテクチャ、テスト自動化フレームワーク (TAF)、テスト自動化戦略

第 1 章の学習の目的：

1.1 テスト自動化プロジェクトの成功要因

CT-TAS-1.1.1 (K2) テスト自動化の目標と意義を説明する

CT-TAS-1.1.2 (K2) テスト自動化プロジェクトの技術的成功要因を特定する

CT-TAS-1.1.3 (K2) テスト自動化の候補プロジェクト選定における適切な投資基準を要約する

1.1 テスト自動化プロジェクトの成功要因

1.1.1 テスト自動化戦略の目標と目的の定義

テスト自動化プロジェクトの戦略を定義する際、以下の事項に対処する必要がある：

- 意図した目的の定義
- リスクの特定
- スコープの定義
- 関与するステークホルダーの特定
- 自動化のためのツールの選択
- テスト自動化アーキテクチャ（TAA）の設計
- 環境の特定

テスト自動化の目標には以下が含まれる場合がある：

- テスト効率の向上
- より広くより深いカバレッジの提供
- テスト対象システム（SUT）の総合品質の向上
- 総コストと市場投入期間の削減
- 手動テスト担当者には実施できないテストの実行
- テスト実行時間の短縮
- テスト頻度の増加

アジャイルソフトウェア開発はより速いデプロイメントサイクルを提供し、クラウドアプリケーションが普及しているため、開発は SUT の品質に関するより早期かつ迅速なフィードバックを必要とする。このシフトの結果として、テスト自動化はその特性とテストの目標を背景として、現代のプロジェクトにおいて注目度・重要性ともに高まっている。

1.1.2 テスト自動化プロジェクトの技術的成功要因の特定

以下の成功要因は、プロジェクトの長期的な成功に影響する要因に焦点を当てたテスト自動化プロジェクトに当てはまる。パイロットプロジェクトを行うことは、ツールの決定と、選択した技術の実現可能性の判断に役立つ。

テスト自動化の成功要因には以下が含まれる：

- テスト対象システム（SUT）のテスト容易性
 - テスト自動化が SUT のインターフェースにアクセスできるようにする
- テスト自動化戦略の定義
 - 戦略は適用可能でカスタマイズ可能である必要がある。その目標は時間とコストの制約内で達成可能である必要があり、最新の状態に維持される必要がある
- テスト自動化アーキテクチャ（TAA）
 - 何を実装するか、どのように実装するかの明確さ
 - 追加情報については、CTAL-TAE シラバスのセクション 3.1.1 を参照すること

- テスト自動化フレームワーク (TAF)
 - TAF は使いやすく、十分に文書化されており、保守可能で、テストを自動化する一貫したアプローチをサポートする。追加情報については、CTAL-TAE シラバスのセクション 3.1.3 を参照すること。
 - 定義・実装されたテストレポート
 - 容易なトラブルシューティング
 - 適切なテスト環境
 - ドキュメント化された自動テストケース
 - テストケース定義と要件への自動テストのトレーサビリティ
 - 容易な保守
 - 最新の自動テスト
 - 明確なデプロイメント計画
 - 明確なテスト実行計画
 - 必要に応じて廃止されるテスト
 - 効果的な例外処理

テスト自動化プロジェクトを開始する前に、既存の要因と欠けている要因を考慮し、選択したテストアプローチのリスクとプロジェクトのコンテキストを念頭に置きながら、プロジェクトの成功確率を分析することが重要である。すべての要因が必要というわけではなく、実際にはすべての要因が満たされることはほとんどない。

1.1.3 テスト自動化の候補プロジェクト選定における適切な投資基準の要約

プロジェクトにテスト自動化を設定することは投資を意味し、多くの場合、相当なコストを伴う。この投資は、プロジェクトの性質や、テスト自動化をそのプロジェクトで使用すべきかどうかと合わせて考慮する必要がある。

テスト自動化を導入する前に、以下の投資基準を検討されたい：

- 導入コスト：テスト自動化の設定に必要な作業に加えて、新しいテスト自動化エンジニア (TAE) の採用、新しいハードウェアの購入、またはトレーニングの設定が必要になる場合があるため、プロジェクトに追加のコストが必要になる
- プロジェクトのソフトウェア開発ライフサイクル (SDLC) における現在のフェーズ：テスト自動化がより早く価値をもたらせるよう、できるだけ早く開始することが望ましい
- プロジェクト/ソフトウェア開発の予想・計画期間：より短期間のプロジェクトでは、テスト自動化を開始するのに十分なリソースがないか、価値をもたらすのに十分な時間が残っていない場合がある
- 保守コスト：ゼロから開始する場合、テスト自動化ソリューション (TAS) の設定には時間がかかり、継続的な保守が必要である

プロジェクトへのテスト自動化導入の投資が許容できる場合、テスト自動化の準備を開始することができる。これには、適切なテストアプローチとテスト自動化ツールの選定が含まれる。このプロセスの主な責任は、関連する意思決定に必要なテスト自動化の知識を持つテストアーキテクトまたはテストマネージャーが戦略的役割として担う。

2 テスト自動化リソース – 60 分 (K2)

キーワード

テスト自動化エンジニア (TAE)、テスト自動化ソリューション (TAS)

第 2 章の学習の目的：

2.1 テスト自動化ソリューション (TAS) 導入のコストとリスク

CT-TAS-2.1.1 (K2) 総保有コストの観点から代替技術ソリューションを比較する

CT-TAS-2.1.2 (K2) テスト自動化ツールのライセンスモデルに関する考慮事項を説明する

CT-TAS-2.1.3 (K2) テスト自動化戦略定義時に考慮すべき要素の例を示す

2.2 テスト自動化における役割と責任

CT-TAS-2.2.1 (K2) テスト自動化ソリューション (TAS) の成功に必要な役割とスキルを要約する

2.1 テスト自動化ソリューション (TAS) 導入のコストとリスク

2.1.1 総保有コストの観点から代替技術ソリューションを比較する

オーナーシップの観点からは、オープンソースまたは商用ツールに基づくカスタマイズされた社内ソリューションが一般的なアプローチである。その他の代替アプローチには、カスタマイズ不可能な商用ソリューション、またはテスト自動化エンジニア (TAE) に作業を委託したアウトソース会社との契約締結がある。

すべての TAS を社内でも構築することで、コスト、リソース、リスク、ガバナンスがすべて組織内に収まることが保証される (例: チームメンバー/開発者、および実際のソリューションに必要なハードウェアとソフトウェアリソース)。このアプローチに従い、この活動に時間を割くことで、必要なすべての TAE とその知識が組織内で利用可能なため、ベンダーやアウトソース会社との契約なしにテスト自動化ソリューション (TAS) を開発できる。ただし、これを成功させるためには、組織は TAS の開発を推進できる適切な TAE を採用またはトレーニングする必要がある。

ベンダーベースのソリューションを使用する場合、オーナーシップは契約の詳細に応じてクライアントとベンダーの間で共有される。組織内に既にパイロットテストされ、すべての要件を満たすテストツールがあり、テストツールに追加機能が必要ない場合、このアプローチの採用は容易である。組織内のテスト担当者はベンダーから必要なトレーニングを受けることで生産的に作業できるようになり、通常は組織内にプロダクトオーナーとして指定された有識者 (SME) がいる。このアプローチのリスクは、テストツールに追加の作業 (欠陥の修正、追加機能の要求など) が必要になった場合、時間がかかる点である。適切な方法で期限内に完了させるためにベンダーとの交渉が必要になり、結果としてこのテストツールを使用しているテスト担当者の作業に影響を与える可能性がある。

組織がこのようなオーナーシップを持ちたくない場合は、アウトソーシングが推奨されるソリューションである。アウトソース会社と協力する場合、クライアントは実際のハードウェア、ソフトウェアを購入したり、必要なスキルセットを持つ従業員を採用したりする必要はない。すべての作業と追加コストはアウトソース会社が担い、ツールのオーナーシップも含まれる。進捗を透明で可視化するために、測定可能な期待値とメトリクスを契約で定義する必要がある。このアプローチは、プロジェクトの短いタイムライン、コスト、またはその他の考慮事項により、TAE の採用に投資する計画がない組織に推奨される。

2.1.2 テスト自動化ツールのライセンスモデルに関する考慮事項の説明

ライセンスはテスト自動化を確立する際に考慮すべき重要な側面である。以下に述べる各ライセンスモデルは、使用性 (ユーザビリティ) とコスト要因 (テスト実行のコスト、開発環境のセットアップの難しさなど) に異なる影響を与える。

ライセンスモデルには以下が含まれる:

- **オープンソース:** 多くの場合、組織はテスト自動化でのテスト目的を達成するためにオープンソースツールを使用している。主な理由は、テストツールの使用にライセンス費用やメンテナンス

費用がかからないことである。多くのユーザーと組織がオープンソースツールの開発に貢献しており、情報収集とサポートの受け取りが容易である。ほとんどのオープンソースライセンスでは、必要に応じてツールを変更したり、重大な制限なしに再公開したりすることも許可されている。

- **ユーザー/マシンごとのライセンス**：商用ツールには多くの場合、ユーザー単位またはマシン単位のライセンスがある。コスト面では、組織が長期的にプロジェクトで作業するテスト自動化エンジニア（TAE）の数を把握している場合、ツールを効率的に使用できる。多くの場合、組織が注文するライセンスが多いほど、より良い価格が得られる。
- **フローティングライセンス**：これは組織内の複数の人が異なる時間帯に使用できるライセンスである。異なるマシンで作業する多くのテスト自動化エンジニア（TAE）がいる場合に非常に便利である。ライセンス数は、TAE の総数または特定のマシン数ではなく、同時使用に基づいて計算される。
- **ランタイムライセンス**：多くの商用ツールには、テスト自動化の実行のためのランタイムライセンスがある。このタイプのライセンスは、テスト自動化ツール、複数の OS、ブラウザーバージョンをサービスとしてホストするクラウドベンダーで発生する。さまざまなモバイルデバイス、プラットフォーム、バージョン、携帯ネットワークを持つデバイスファームへのアクセスを提供するクラウドベンダーもある。これらのサービスを利用する場合は、テスト実行に使用した時間に対してのみ課金されるため、ランタイムライセンス方式で課金される。

2.1.3 テスト自動化戦略定義時に考慮すべき要素の例示

多くの要因がテスト自動化の実装とその戦略に関する意思決定に影響を与える可能性がある。

最も重要な要因は以下のとおりである：

- 時間的制約
- テスト自動化ソリューション（TAS）を開発するために必要な専門知識のレベルとテスト自動化エンジニア（TAE）の数
- テストハードウェア
- テストツールのライセンス
- 適応性
- 保守
- 異なるプラットフォームのサポート（Web アプリ、デスクトップ、モバイルなど）
- 継続的インテグレーション/継続的デリバリー（CI/CD）サポート
- テストマネジメントとの統合とテストレポート

最初かつ最も重要なコスト要因は、実施すべき作業とテスト自動化自体のためのタイムラインである。

タイミングの観点から、締め切りが短い場合によく使われるアプローチは、TAS を作成するために少数の熟練した TAE のみを採用することである。より長いロードマップとスケジュールの場合、組織はより多くの状況・情報を用いて、必要な TAE の数を決定できる。テスト対象システム（SUT）が成長し、改

善され、より複雑になるにつれて、**TAS** とテストを実装・保守するために、より多くの **TAE** を採用するという決定を行うことができる。

その他の重要なコスト要因は、ツールとそのライセンスである。予算には、必要なテストツール、ハードウェア、および **TAE** の追加トレーニングが含まれる必要がある。これらの要因は、テスト結果をテストマネジメントシステムにアップロードし、構成管理システムへのビルドをトリガーするなどの追加の非テスト機能を実行するための他のツールやシステムへの統合と強い関連がある。これらの追加機能のためには、無料または有料のツールとライブラリがあるが、テスト自動化戦略の作成時に考慮し、予算に含める必要がある。

開発チームが持つテスト環境とエージェントの数は常にコンテキストに依存する。**SUT** とリリーススケジュールが大きく複雑であるほど、組織が必要とするリソースが多くなり、コストも増加する。リソースコストを制限する最近のアプローチとして、クラウドプロバイダーを使用してオンデマンドのテストハードウェアリソースとランタイムライセンスの費用を払う方法がある。ただし、これは慎重に使用する必要がある。マシンが稼働したままになり、専用ハードウェアよりも高額になる場合がある。このアプローチは **TAE** チームの高い保守・運用コストを削減できる。

2.2 テスト自動化における役割と責任

2.2.1 テスト自動化ソリューション（TAS）の成功に必要な役割とスキルの要約

テスト自動化ソリューション（**TAS**）の成功のために、組織は強力なプログラミングと技術アーキテクチャ知識を持つ熟練したテスト自動化エンジニア（**TAE**）を必要とする。

プロジェクトの規模と成熟度によっては、実際のビジネスドメインとテスト目標を理解する少なくとも 1 人の強力な有識者（**SME**）（テストリード、アーキテクト、ビジネスアナリストなど）も必要である。この人物の役割は、テスト目標に基づいたコンセプトと実際の **TAS** のロードマップを推進・作成することである。実際の作業のためにチームをトレーニングし、動機付け、構築するためには、チームマネジメント力とソフトスキルが必要である。[CTEL-TM-MTT]

TAE は強力な技術スキルと異なる **SDLC**、テスト対象システム（**SUT**）のアーキテクチャおよびその開発環境に関する知識を持つ必要がある。技術的な側面以外に、**TAE** はテスト自動化の目標についてテストアナリストや他のステークホルダーと協力する能力を持つ必要がある。網羅的なテストが達成できないのと同様に、テスト自動化についても、**100%**のテスト自動化カバレッジは達成できない。時間と労力が限られているため、**TAE** はリスク評価に貢献することで、ビジネスと投資の観点から最も影響の大きいテスト条件を優先できる必要がある。

3 テスト自動化の準備 – 225 分 (K3)

キーワード

API テスト、コンポーネントテスト、契約テスト、シフトレフト、シフトライト、テスト対象システム (SUT)、テスト自動化アプローチ、テスト条件、テストダブル、テストレベル、テストピラミッド

第 3 章の学習の目的：

3.1 テストレベルをまたいだ統合

CT-TAS-3.1.1 (K2) テスト自動化の配分の違いを区別する

CT-TAS-3.1.2 (K2) テスト対象システム (SUT) のアーキテクチャに基づくテスト自動化アプローチを選択する

CT-TAS-3.1.3 (K3) シフトレフトとシフトライトアプローチを達成するためのテスト自動化配分の最適化方法を実証する

3.2 異なるソフトウェア開発ライフサイクル (SDLC) モデルにおける戦略的考慮事項

CT-TAS-3.2.1 (K2) テスト自動化プロジェクトがレガシーソフトウェア開発ライフサイクルモデルにどのように準拠するかを説明する

CT-TAS-3.2.2 (K2) テスト自動化プロジェクトがテスト自動化をサポートするアジャイルソフトウェア開発のベストプラクティスにどのように準拠するかを説明する

CT-TAS-3.2.3 (K3) 継続的テストにおけるテスト自動化をサポートする DevOps ベストプラクティスにテスト自動化プロジェクトが準拠するよう準備する

3.3 テスト自動化の適用可能性と実現可能性

CT-TAS-3.3.1 (K2) テスト自動化に適したテストの適合性判断基準を説明する

CT-TAS-3.3.2 (K2) テスト自動化のみが対処できる課題を特定する

CT-TAS-3.3.3 (K2) 自動化が困難なテスト条件を特定する

3.1 テストレベルをまたいだ統合

3.1.1 テスト自動化の配分の区別

Mike Cohn はテスト自動化ピラミッドの元々の概念を考案し、ユニット（ISTQB ではこれをコンポーネントと呼ぶ）、サービス、UI の 3 つのレベルを説明した。それ以来、多くのバリエーションが登場し、それらはすべてテストレベルの役割とこれらのレベルをまたいだテストケースの配分を説明しており、同じ基本原則を共有している（ISTQB CTFL シラバス、セクション 5.1.6 テストピラミッドを参照）。

Mike Cohn のオリジナルのテスト自動化ピラミッドは、テスト自動化ソリューション（TAS）によって実行されるテストタイプに焦点を当てている。サービスレベルは 3 つのテストタイプに分解することができる：コンポーネント統合テスト、契約テスト、API テスト。

ユニットテスト（ISTQB ではこれをコンポーネントテストと呼ぶ）は、コードの品質に焦点を当てた個別コンポーネントの検証のためである。コンポーネント統合テストでは、モックやスタブなどのテストダブルを活用して UI と API の検証が可能である。契約テストはサービス間の契約の検証を可能にしている。API テストは、実際のサービス接続を通じて実データによる特定サービスの機能的検証に焦点を当てている。UI テストは、GUI を操作して行うシステムのエンドツーエンドテストである。

現在のテスト状態をベースラインとして、目標状態を描くことは有益である。これにより、不足しているものや許容できるテストのレベルを明確に理解できる。これにより、各テストレベルで行われるテストの量とそれが手動テストか自動テストかが示される。目標状態は、タイムラインやその期間内に達成可能なものに依存する。達成可能であれば、ピラミッド形が目標形状となるべきである。

テスト配分の例：

- **ピラミッド**：これは安定した迅速なテストで、上位テストレベルでのテストが少なく、下位テストレベルでより多くのテストを行うバランスのとれたテスト配分である。利用可能なリソースと時間枠が許す場合、これはしばしば目標状態のピラミッドである。
- **アイスクリームコーン**：これはピラミッドの逆バージョンである。サービステストの量はバランスが取れているが、テストは UI テストレベルでの欠陥の大部分を見つけることに大きく依存しており、複雑さのため自動化のコストが高くなりがちである。コンポーネントテストの欠如により、ソフトウェア開発ライフサイクル（SDLC）の後半で欠陥が発見される。
- **砂時計**：テストは最高・最低テストレベルが多く、サービスレベルのテストがほとんど欠けており、統合欠陥が生じる。ビジネスロジックが API（つまりサービス）によって提供されている場合、多くの UI テストを下位テストレベルに移行できる。
- **傘**：テストはコストの高い UI テストに完全に依存している。これにより、欠陥解決が遅く、テストケースと TAS の保守コストが高くなる。下位レベルのテストケースを実装することが技術的に不可能な場合、傘形からの脱却は達成できないかもしれない。その場合は、UI テスト自動化スイートの最適化、テスト実行時間の短縮、安定性の向上に焦点を当てる必要がある。

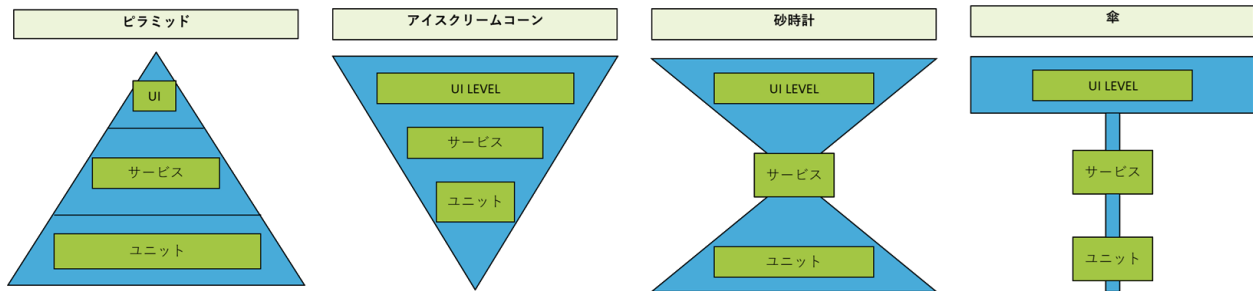


図 1：テスト配分の例

3.1.2 テスト対象システム（SUT）のアーキテクチャに基づくテスト自動化アプローチの選択

テスト戦略でテストレベルを定義する方法は多数ある。どれを選択するかは、組織の文化、ソフトウェアエンジニアリングの全体的な成熟度、従うソフトウェア開発ライフサイクル（SDLC）、SUT がメインフレームかマイクロサービスアーキテクチャかなどの複数の要因に大きく依存する。

ピラミッド形は理想的なテスト配分と考えられているが、常に達成可能なわけでもなく、最終状態への移行に長時間かかることもある。傘形から砂時計形への変換など、目標状態として現実的な形状を持つことは良い実践である。一度達成すれば、新しい目標状態を決めることができる。

メインフレームシステムに適したテスト自動化配分の選択は、現代のソフトウェア開発の場合とは異なる。一部のテストレベルはまったく自動化できない可能性があるなど、これも多くの要因に依存する。メインフレームへのアクセスは従来、端末エミュレーション（グリーンスクリーン）を介して行われている。バッチプロセスの検証は可能だが制限される。コンポーネントテストの実施はより困難である。マイクロサービスがほとんど存在しないため、API テストによるインターフェース間のデータ通信の検証は常に選択肢とはならない。バッチジョブ、データベース、GUI を通じたテストがより一般的である。

レガシーソリューションをモダナイズし、マイクロサービスアーキテクチャへの移行を徐々に行っている組織は、システムの近代化した部分に API テストと契約テストを導入できる。

3.1.3 シフトレフトおよびシフトライトアプローチを達成するためのテスト自動化配分の最適化方法の実証

現在の状態の配分を特定し目標状態の配分を選択したら、改善のロードマップを計画できる。これにより、自動でテストするもの（テスト自動化の範囲）とテスト方法（テスト自動化戦略）が決まる。実装すべきアイテムの優先順位付けされたバックログとテスト自動化の開始基準を決定する必要がある。

テスト自動化の配分が不均衡な場合は、ボトムアップアプローチで進めることが推奨される。コードカバレッジが低い場合は、コンポーネントテストケースが不足していることの兆候であり、追加のコンポーネントテストケースを作成する必要がある。次のステップとして、コンポーネントテストの品質を見

直し、必要に応じて改善する必要がある。ベストプラクティスの提案やテスト駆動開発技法の採用、テスト技法ワークショップの開催によりテスト自動化の品質を向上させる。

UI や API テストのためのコンポーネントテストを導入し、テストダブル（モック、スタブなど）を活用することで、コストの高い、遅い、信頼性の低いテストからのシフトレフトを達成する。実際のサービスとデータへの依存を取り除くことで、テスト実行の一貫性が向上し、CI/CD パイプラインへの統合が容易な早期フィードバックが得られる。

近年、契約テストがより顕著になっている。プロバイダーとコンシューマー間の契約を検証することで、欠陥の根本原因を早期に特定しやすくなる。チームは API テストや UI テストに欠陥の検出を依存することなく、そのようなテストケースの数を減らすことができる。API のコールグラフを構築することは、どの API が互いに接続されているか、また選択した API のコンシューマーまたはプロデューサーはどれかを追跡するための賢い方法である。これにより、システムの潜在的な問題点をより良く特定できる。

シフトレフトがソフトウェア開発ライフサイクル（SDLC）でテストをより早く移動させるアプローチであるのに対し、シフトライトは、プレプロダクションテスト環境や本番環境でのパフォーマンス評価の手段として、テスト対象システム（SUT）がリリースされた後にテストを後ろに移動させる。シフトライトを適用することで、テスト担当者は実際のユーザーからのフィードバックを得ながらアプリケーションと API のパフォーマンスを監視できる。シフトレフトアプローチではテスト自動化がより顕著であるが、可観測性と組み合わせることで、テスト自動化はフルリリースを進めるか、リリース候補をロールバックするかについてより迅速な意思決定を支援する。

シフトライトテストの目標：

- ユーザーの好みの把握
- カナリアリリースとダークローンチをサポートし、ユーザー機能への影響を最小限に抑える
- 本番環境での欠陥の早期発見
- テスト自動化のスコープと使用の拡大を支援する
- カバレッジの向上

3.2 異なるソフトウェア開発ライフサイクル（SDLC）モデルにおける戦略的考慮事項

3.2.1 レガシーソフトウェア開発ライフサイクル（SDLC）モデルへのテスト自動化プロジェクトの適合方法の説明

ウォーターフォールモデルでは、テストフェーズは要件分析、システム設計、実装フェーズの後に来る。その厳格な順序のため、テスト自動化活動は後になる。テストとテストの間隔が長いと、テスト自動化を活用する機会が減り、テスト自動化からのフィードバックも通常は遅すぎるため、投資対効果（ROI）が低くなる。

V字モデルでは、すべてのテスト計画とドキュメント準備が早期フェーズで行われる。例として、アーキテクチャ設計フェーズは統合テスト設計を生成し、ウォーターフォールよりも早くテストを設定する。残念ながら、テスト自動化ソリューション（TAS）の実際の実装はソフトウェア開発ライフサイクル（SDLC）の後半になる。ウォーターフォールモデルと比較してテスト自動化のROIは高いものの、現代のアジャイルソフトウェア開発プラクティスには依然として及ばない。

3.2.2 テスト自動化をサポートするアジャイルソフトウェア開発ベストプラクティスへのテスト自動化プロジェクトの適合方法の説明

アジャイルソフトウェア開発の理想の一つは、スプリント内テスト自動化を達成することである。これは、各ユーザーストーリーの終了基準の一部として必要なすべてのテスト自動化活動を決定することを意味する。テストケースの定義、自動化されたテストケースの実装、テスト自動化フレームワーク

（TAF）の更新、および場合によってはCI/CDパイプラインへの統合が含まれる。アジャイルプラクティスを正しく適用していない組織は、テスト工数の見積もりを含めず、別のチケットで管理するか、まったく監視しないことがある。スプリント内テスト自動化を達成することで、アジャイルチームは各スプリント内またはスプリント終了時に合意したスコープをデプロイメントする準備が整っていることを確認できる。チームがアジャイルの観点から成熟していない場合、最初の目標はスプリント内テストを達成することで、自動化はスプリント一つ分遅れることになる。そこから、チームはスプリント内テスト自動化の達成に向けて取り組むことができる。

3.2.3 継続的テストを達成するための DevOps ベストプラクティスへのテスト自動化プロジェクトの準備

アジャイルソフトウェア開発は作業の編成方法に焦点を当てているのに対し、DevOpsはソフトウェアのエンドツーエンドデリバリーを担当する。これはビルド、統合、テスト、デプロイメント、および本番活動を自動化することで達成される。これにより継続的改善を確保するフィードバックループを通じた継続的テストが促進される。

重点はコンポーネント、コンポーネント統合、契約テストなどの下位レベルの自動テストケースの実装にある。実際のデータとサービスへの依存を減らすことで、テストはより短時間で実行される。実現可能であれば、テスト自動化はテスト対象システム（SUT）がビルドされる同じパイプラインで実行されるべきである。各開発・ビルドフェーズ（ローカル、プルリクエスト、マージ、デプロイメントなど）の後に異なるテストスイートがトリガーされる。

テスト担当者が大規模なUIおよびAPIテストスイートを構築する能力がある場合、それらは個別に実行されて、追加の価値を提供できる。手動テストの工数は探索的テストとエンドユーザーフィードバックに焦点を当て、テスト自動化の補完的な活動として推奨される。

3.3 テスト自動化の適用可能性と実現可能性

3.3.1 テスト自動化に適したテストの適合性判断基準の説明

テスト自動化のためのテストケースの選択は、自動化できるテストケースを理解するテストアナリスト、またはそのような意思決定に必要なドメイン知識を持つテスト自動化エンジニア（TAE）によって通常行われる。

テスト自動化のためのテストケースの選択と優先順位付けにおいて以下が考慮される：

- テストケースを自動化された形で実装することは技術的に可能か？
- 自動化されたテストケースのデリバリーに影響する技術的課題はあるか？チームは実装作業を行うための準備とトレーニングができていますか？
- コーディング工数は適切な投資対効果（ROI）を提供するか？（セクション 5.1.1 を参照）
- テストケースを頻繁に実行することに価値があるか？
- 機能テストか非機能テストか？スモークテストスイート、リグレッションテストスイート、または確認テストスイートの一部か？
- テストケースは繰り返し実行可能か？
- テスト対象システム（SUT）が更新により変更された場合、テストケースは容易に保守できるか？
- テストケースは頻繁に使用されるビジネスワークフローをカバーしているか？
- テストステップとテストデータの再利用を可能にするテストに機能的な重複があるか？

3.3.2 テスト自動化のみが対処できる課題の特定

テスト自動化でのみ実施できる特定のテストがある。これらには以下のカテゴリが含まれる：

1. 手動テスト実行時間が適切な時間より長い場合
2. テストケースの実行を同期する必要がある場合
3. テスト結果をパイプラインで利用可能にする必要がある場合
4. 欠陥を見つけるために大きなログファイルを解析する必要がある場合
5. テストでのタイミングの精度が必要な場合
6. 複数の OS、ブラウザー、デバイス、場所、または設定にまたがるテストの順列が必要な場合
7. カバレッジを最大化するために大量のテスト実行やデータ入力が必要な場合
8. ストレステストや信頼性テストなど、自動化された監視・分析や大量のユーザーからの入力を必要とするすべての非機能テスト

3.3.3 自動化が困難なテスト条件の特定

テストケースの自動化を困難または不可能にする特定のテスト条件がある。実際の現場での例は多数ある。その一部を以下に示す：

- 異なるプラットフォーム間での UI の一貫性を含む設計要件の検証。ソフトウェアの全体的な外観と使用感は主観的であり、人間によるレビューが必要である。
- 人間の介入が多すぎるテストケース。金融セクターでは、ローン申請が良い例である。このプロセスでは、特定の規制や条件（収入不足、個人情報の誤りなど）のケースが存在する場合がある。そのような状況では、担当者は実際のローン申請を再確認し、手動での意思決定または顧客への連絡が必要になる。
- OS レベルの制限などのテスト自動化活動をブロックする技術的な困難。一例として、ユーザーにテキストメッセージを送信するネイティブ OS ソフトウェアがある。これらのテキストメッセージのインタラクションや検証は、OS がテスト対象システム（SUT）の外部でのやり取りを許可しないため、UI テスト自動化ツールでは不可能である。
- 長時間の依存関係を持つテスト条件は、テスト自動化を非効率にし、手動テスト実行と比較して適切に設定することが難しいことが多い。例として、テスト担当者が SUT にログインし、ホームページのコンテンツを更新して、セッションタイムアウトにより SUT が自動的にログアウトするまで 2 時間待つ必要があるケースがある。経過時間が何らかの手動の方法（モック、サーバーサイドレスポンス、OS レベルの時間変更など）で変更できない場合、そのようなテストケースを自動化された方法で実装することは推奨されない。

4 テスト自動化の組織的デプロイメントとリリース戦略 – 135 分 (K2)

キーワード

コンポーネント、確認テスト、品質ゲート (quality gate)

第 4 章の学習の目的：

4.1 テスト自動化ソリューション (TAS) の計画

CT-TAS-4.1.1 (K2) テスト自動化が市場投入期間を短縮する方法を特定する

CT-TAS-4.1.2 (K2) テスト自動化が、報告された欠陥を要件に照らして検証するのに役立つ方法を特定する

CT-TAS-4.1.3 (K2) テスト自動化のための運用関連シナリオの開発を可能にするアプローチを定義する

4.2 デプロイメント戦略

CT-TAS-4.2.1 (K2) テスト自動化デプロイメント戦略を定義する

CT-TAS-4.2.2 (K2) デプロイメントにおけるテスト自動化リスクを特定する

CT-TAS-4.2.3 (K2) テストデプロイメントリスクを軽減するアプローチを定義する

4.3 テスト環境内の依存関係

CT-TAS-4.3.1 (K2) テスト環境内のテスト自動化コンポーネントを定義する

CT-TAS-4.3.2 (K2) テスト自動化のインフラストラクチャコンポーネントと依存関係を特定する

CT-TAS-4.3.3 (K2) テスト対象システム (SUT) 内での統合のためのテスト自動化データとインターフェース要件を定義する

4.1 テスト自動化ソリューション (TAS) の計画

4.1.1 テスト自動化が市場投入期間を短縮する方法の特定

テスト自動化はテストサイクルの時間を短縮するため、ソフトウェアを市場により早く投入する助けになる。ソフトウェアリリース前のテストは、十分な量の検証と妥当性確認が必要である。これはリグレッションテスト中に特に重要で、このタイプのテストは再利用を促進し、コストを削減し、繰り返しテスト実行の一貫性を提供する。

テスト自動化は手動テスト工数を減らし、同じテストスコープをカバーしながら開発者に迅速なフィードバックを提供する。コンポーネントテストとコンポーネント統合テストは、通常手動では実施されないが、自動化される場合、ソフトウェア開発ライフサイクルの早期にテストを行うことも可能になる。

品質ゲート (quality gate) は、ソフトウェアが次のフェーズに進む前に満たすべき、プロセスに組み込まれた強制的な手段である。テスト自動化に基づいた品質ゲートの設定により、プレプロダクションまたは本番環境への加速されたデプロイメントプロセスが可能になる。これらの品質ゲートを活用することで、欠陥がより早く発見され、市場投入期間が短縮される。テスト実行時間は並列テスト実行を活用し、シフトレフトアプローチに従うことで削減できる。シフトレフトアプローチは品質意識の文化を促進し、ソフトウェア開発ライフサイクルの早い段階でのテストを奨励する。これには複数の独立したテストケースやクロスブラウザーテストが含まれる場合がある。

追加情報については、セクション 3.1.3、6.1.2、および 6.2.1 を参照のこと。

4.1.2 テスト自動化が報告された欠陥の検証に役立つ方法の特定

コード修正後に実施される確認テスト (confirmation testing) は、報告された欠陥に対処できる。テスト担当者は通常、欠陥が再現する手順に従ってテストを実施し、欠陥が存在しなくなったことを確認する。

欠陥は後続のリリースに再び現れることがある（例えば、これは構成管理やコードリポジトリ管理の問題を示す場合がある）。そのため、確認テストはテスト自動化の適切な候補であり、既存のリグレッションテストスイートに追加できる。

自動化された確認テストは通常、機能の範囲が限定的である。実装は欠陥が報告され、再現に必要なテストステップが理解された後はいつでも行える。

自動化された確認テストの追跡により、欠陥の解決に費やされた時間とテストサイクル数のレポートが可能になる。

テスト自動化を使用して複数のプラットフォーム、デバイス、ブラウザー、OS バージョン間で修正を検証することで、テストに費やされる時間が大幅に削減される。

4.1.3 テスト自動化のための運用関連シナリオの開発を可能にするアプローチの定義

運用受け入れテストは本番システムのソフトウェアの準備を保証し、通常リリース直前に実施される。この取り組みは、テスト対象システム（SUT）のシステム、コンポーネント、その他のインフラストラクチャの最終的な検証を行い、本番への準備状態をテストするためのものである。これが必要なのは、テスト自動化エンジニア（TAE）の最善の努力にもかかわらず、SUT がテスト環境の外部と本番環境で同じように動作するという保証がないためである。

良い運用テスト計画には、信頼性のテスト、フォールトトレランスのテスト、インテグリティ、保守性のテストが含まれる。以下に使用できるテスト自動化アプローチを示す：

- 静的コード解析：コードのセキュリティと脆弱性を分析する自動化ツール。テスト結果は時間の経過による傾向分析のために保存される。
- エンドツーエンドテスト：エンドツーエンドのユーザーシナリオを実施し、テスト環境のあらゆる側面を検証して欠陥を発見する自動テストスクリプト。
- フェイルオーバーテスト：アプリケーションのハードウェアがオフラインになったときに何が起これるかを具体的にテストする自動テストスクリプト。例として、物理サーバー、クラウドサーバー、ネットワーク、コンピューターディスク、その他のハードウェアコンポーネントが誤動作したときにアプリケーションがどのように回復するかが含まれる。テストスクリプトは SUT の自動回復能力の成否を測定するよう設計されており、カオスエンジニアリングを実装している組織にとって特に重要である。
- バックアップと復元テスト：現在のバージョンのバックアップ作成、その後以前のポイント（つまり、ソフトウェアの古いバージョン）へのロールバックの成否をテストする自動テストスクリプト
- 性能効率テスト（負荷テストなど）：多数のユーザーが同時に SUT を使用することをシミュレートするために使用できる自動テストスクリプト。テスト結果は比較のために保存され、時間の経過とともにトレンド化される。
- 運用ドキュメントレビュー：自動テストスクリプトは、SUT のドキュメントのバージョンを比較し、特定のリリースに追加された新機能に基づいて更新が必要かどうかを開発チームに通知するために使用できる。
- セキュリティテスト：SUT を静的および動的に評価するために、業界標準のセキュリティテストツールと組み合わせて自動テストスクリプトを作成できる。テスト結果は比較と傾向分析のために時間の経過とともに保存される。
- 組織のサービスレベル合意書（SLA）に基づく監視：ソフトウェア開発ライフサイクル（SDLC）中に使用される自動テストスクリプトは本番運用の監視に再利用でき、自動テストが失敗した場合にアラートを送信する。これは実際のユーザーが経験する前に本番障害を検知するための予防的な手段である。

運用ソフトウェア検証のためのテスト自動化は、特に自動テストが繰り返し実行でき、複数の環境で実行できる場合、大きなメリットをもたらせる。専用の自動テストスイートを作成することで、テスト結果を生成し、以前のテスト実行と比較できる。これにより、SUT の新しいバージョンがリリースされた

際の一貫性が確保される。運用固有のテスト条件の信頼性の高い自動テストスイートは、時間の経過とともにコストを削減できる。

4.2 テスト自動化デプロイメント戦略

4.2.1 テスト自動化デプロイメント戦略の定義

良いテスト自動化デプロイメント戦略は、全体的なテスト環境、利用可能なテストツール、テスト対象システム (SUT) へのアクセス、テストスクリプトのストレージとその他の依存関係、テストデータのプロビジョニングを考慮するが、これらに限定されない。これらのハイレベルな考慮事項を念頭に置いて、テスト自動化エンジニア (TAE) はテスト自動化ソリューション (TAS) を開発・デプロイメントするための戦略の検討を開始できる。

テスト環境：TAE は、異なるテスト環境で SUT にどのようにアクセスするかを検討する必要がある。キーワード駆動アプローチや他のソリューションを使用してテスト自動化のテストウェアを開発し始める際、このソリューションが複数のテスト環境でどのように動作するかを考慮する必要がある。例えば、テストスクリプトは、Web ベースのアプリケーションの場合は URL (Uniform Resource Locator) を変更するだけで、どのテスト環境にあるかにかかわらず同じように実行できるよう開発されるべきである。

ツール：TAE は TAS を構築するために使用するツールも検討する必要がある。商用ツールの場合、そのライセンス方式を理解する必要がある。TAE はテスト環境がツールのライセンスサーバーにアクセスできる必要があることを知る場合がある。テストスクリプトが複数のテスト環境で使用される予定の場合、これを考慮する必要がある。テスト環境でライセンスサーバーが利用可能だからといって、プレプロダクション環境でも利用可能とは限らない。

ソフトウェアアクセス：SUT へのアクセス方法について重要な考慮事項を理解する必要がある。テストスクリプトはパラメーターを受け入れるよう設計できるため、SUT のエンドポイントが変更された場合に特定のユーザーや特殊なアクセス権を持つ認証情報をすぐに更新できる。これは異なるテスト環境に移行する際および URL を変更する必要がある場合に再び非常に重要になる。さらに、プレプロダクション環境によっては、異なる認証情報 (ユーザーアカウント、テストアカウント、生体認証、スマートカードなど) を使用する必要がある場合がある。良く設計された自動テストスクリプトには、シンプルなパラメーターを設定でき、テスト環境に関わらずテストスクリプトが期待通りに実行できるだけの十分な柔軟性が含まれる。

テストスクリプトストレージ：TAE は、自動テストスクリプトを保存・管理する中心的な場所を決定する。良い戦略は、構成管理を利用したソースコードリポジトリを組み込むことである。これにより、テストスクリプトはソースコードリポジトリにアクセスできれば複数のテスト環境からアクセス可能になり、SUT の特定バージョンに対してバージョンを作成できる。すべての設定と依存関係はテストスクリプトと同じリポジトリに保存でき、最適なポータビリティを提供する。つまり、TAS、TAF とすべてのテストケースはリポジトリ内に保存・管理できる。

データプロビジョニング：自動テストスクリプトが、SUT がテストされているテスト環境にすでに存在する特定のテストデータに依存しているかどうかを理解することが重要である。良いテストスクリプト設計はできる限り静的データ（固定データセット）を避ける。しかし、実現不可能な状況もある。そのような場合、TAE はテストデータをプリロードするか、テスト自動化を使用して実際のテストスクリプトが実行されて SUT をテストする前に必要なテストデータを生成するセットアップテストスクリプトを作成するソリューションを決定する必要がある。どちらのアプローチにも長所と短所がある。管理者にテストデータをプリロードしてもらうのは便利である。しかし、TAE はサポートの利用可能性について別のリソースに依存することになる。セットアップテストスクリプトで自己完結するアプローチは、別のリソースへの依存を排除する。ただし、これらのテストスクリプトをまとめるのに時間がかかり、保守が必要なソリューションの別の部分になる。

上記のすべての項目を考慮することで、テスト自動化戦略はテスト自動化活動の適切な計画と制御を提供できる。

4.2.2 デプロイメントにおけるテスト自動化リスクの特定

技術的な問題は製品リスクとプロジェクトリスクにつながる可能性がある（追加情報については、CTFL シラバス、セクション 5.2.2 を参照）。典型的な技術的問題には以下が含まれる：

- 過度な抽象化により、テスト自動化コードが実際に何をしているかの理解が困難になる場合がある（例：キーワード駆動アプローチのキーワードなど）
- テストデータテーブルが過大、複雑、あるいは扱いにくくなって他のテスト環境へ移行できなくなり、一貫性のないステータス結果となる
- テスト自動化ソリューション（TAS）が特定の OS ライブラリや、テスト対象システム（SUT）のすべてのテスト環境で利用可能でない可能性がある他のコンポーネントを使用する依存関係

典型的なデプロイメントプロジェクトリスクには以下が含まれる：

- 要員の問題：テスト自動化を維持するための適切な人員の確保が困難な場合がある
- SUT の更新に起因する計画外の TAS 保守（TAS が誤動作する原因となる）
- テスト自動化の導入の遅延
- SUT に加えられた変更に基づく TAS 更新の遅延
- TAS が非標準の UI オブジェクトをキャプチャできない
- 古くなったテストケースをテストスイートに残し、テスト実行時間を無駄にする

TAS プロジェクトの潜在的な失敗ポイントには以下が含まれる：

- 異なるテスト環境への移行
- 本番環境へのデプロイメント
- 自動化もテストすべきソフトウェアであることを忘れること

さまざまな技術的問題、プロジェクトリスク、潜在的な失敗ポイントがテスト自動化プロジェクトの成功を脅かす可能性があることを認識することが重要である。一般的な技術的問題には、過度な抽象化から生じる複雑さ、テストデータ管理の課題、特定コンポーネントへの依存関係がある。プロジェクトリスクは、要員の困難、システム更新による保守問題、デプロイメントの遅延、古くなったテストケース

の存在も考慮する必要がある。さらに、異なる環境への移行や自動化ソフトウェア自体のテストの必要性を見落とすなどの潜在的な失敗ポイントも、適切な計画で対処すべきである。全体的に、監視と積極的な対策がテスト自動化プロジェクトの成功を確保する。

4.2.3 デプロイメントリスクを軽減するアプローチの定義

これらのリスク領域に対処するために採用できる多くのリスク軽減戦略がある。その一部を以下に説明する。

テスト自動化ソリューション (TAS) は、社内開発であれ取得したソリューションであれ、独自のソフトウェア開発ライフサイクル (SDLC) を持っている。覚えておくべきことは、TAS も他のソフトウェアと同様に、構成管理下に置かれ、その機能が文書化される必要があるということである。そうでなければ、その異なる部分をデプロイメントして連携させたり、特定のテスト環境で動作させたりすることが非常に困難になる。

また、文書化された、明確で、従いやすいデプロイメント手順が必要である。この手順はバージョン依存であるため、構成管理下にも含める必要がある。

TAS をデプロイメントする際には、2つの明確なケースがある：

1. 初回デプロイメント
2. 保守デプロイメント：TAS はすでに存在しており、更新をデプロイメントする必要がある

初回デプロイメントに関連するリスクには以下が含まれる：

- テストスイートの総テスト実行時間は、テストサイクルの計画されたテスト実行時間より長くなる場合がある。この場合、次のスケジュールされたテストサイクルが始まる前に、テストスイートが完全に実行されるのに十分な時間を計画することが重要である。
- テスト環境でのインストールと設定の問題が存在する（データベースのセットアップと初期ロード、サービスの起動/停止など）。TAS は、テスト環境内で自動テストケースが実行するための必要な事前条件を作成するためのテストフィクスチャ（事前定義されたデータセット）を必要とする。

保守デプロイメントについては、追加の考慮事項がある。TAS 自体が進化する必要があり、その更新は本番環境にデプロイメントする必要がある。TAS の更新版を本番環境にデプロイメントする前に、テストする必要がある。したがって、新機能を確認し、テストスイートが更新された TAS で実行できること、テストレポートが送信できること、パフォーマンスの欠陥や他の品質問題がないことを検証する必要がある。場合によっては、TAS の最新バージョンに合わせてテストスイート全体を変更する必要があるかもしれない。

4.3 テスト環境内の依存関係

4.3.1 テスト環境内のテスト自動化コンポーネントの定義

テスト自動化コンポーネントは通常、ツール、仮想マシン、自動テストスクリプト、コンテナ、設定で構成される。テスト環境にはテスト対象システム（SUT）も含まれる。テスト環境内のテスト自動化コンポーネントは以下のように定義できる：

SUT：これはテスト環境の明白な部分である。SUT は全体としてテストされるか、サブコンポーネント（API、Web ベースのインターフェース、データベースなど）に分割される。

プラットフォーム：プラットフォームはテスト自動化コンポーネントがホストされている場所を説明する。これには、テスト自動化を効率的でポータブルにするために使用できるクラウドインフラストラクチャ、ネットワーク、仮想マシン、コンテナが含まれる。

テストケースとテストスイート：自動化と手動の両方のアクションを指示するテストステップで構成される個別のテストケースを説明する。テストスイートはテストケースの論理グループを説明し、効率的に一緒に実行できる。

ツール：さまざまなテスト自動化ツールがいくつかの理由で使用される。ツールのコレクションには、UI テスト自動化、API エンドポイントのテスト、データ生成、監視、要件管理、欠陥管理、ログと報告ツール、およびメトリクスに基づくトレンドを生成するためのツールが含まれる。

TAF：これはテスト自動化フレームワーク（TAF）設計を構成するすべての項目を含む。TAF には、ドライバースクリプト、共通ライブラリ、自動テストケースのテンプレート、データロード/プロビジョニングスクリプト、TAF コンポーネントの使用法に関するドキュメント、およびテスト自動化エンジニア（TAE）が TAF を活用するのに役立つチュートリアルが含まれる。追加情報については、CTAL-TAE シラバスのセクション 3.1 を参照のこと。

テストコンポーネントの保守は重要な考慮事項である。テスト環境を過度に複雑にすると、ソリューションの恩恵を受ける代わりに、TAF の欠陥修正に何時間も費やすことになる場合がある。コンポーネントをできる限り再利用可能にするために、ツール、設定、プラットフォームのポータビリティの適切なバランスを見つけることが重要である。

4.3.2 テスト自動化のインフラストラクチャコンポーネントと依存関係の特定

テスト自動化を組み立てる際に認識すべきインフラストラクチャコンポーネントと依存関係が多数ある。これらはテスト自動化ソリューション（TAS）を実行するために必要なすべての前提条件をカバーする。主要なコンポーネントと依存関係には以下が含まれる：

ホストマシン：仮想マシン、物理サーバー、ラップトップ、デバイス（タブレット、モバイルなど）を指す。テスト自動化ソフトウェアがインストールされており、テストスクリプトが作成・実行される場所である。

ネットワーク：テスト自動化ソリューション（TAS）にテスト対象システム（SUT）へのアクセスを提供するものである。自動テストの並列実行を提供するために複数のホストマシンをネットワーク接続することも含む場合がある。通常、ホストマシンは同じネットワーク上にあり、互いに通信できるよう適切に設定される必要がある。

プラットフォーム：テスト自動化は、他のソフトウェアと同様に、クラウドプラットフォーム上やコンテナ環境上で実行できるように設計できる。プラットフォームが基盤となる OS への権限とアクセスを提供する限り、必要なすべてのツールと依存関係をインストールできる。

ソフトウェア依存関係：テスト自動化ツールを正しく動作させるためには、テスト自動化が持つすべての依存関係を理解することが必要である。例えば、特定のテスト自動化ツールは、最初にホストマシンにプログラミング言語の最新バージョンをインストールする必要がある場合がある。TAE はツールを選択する前後にこれらの依存関係を考慮する必要がある。

SUT：コンポーネント、依存関係、全体のインフラストラクチャを考慮し適切に設定したら、最後のステップは SUT へのアクセスを確保することである。ネットワークに加えて、SUT とのインターフェース方法も考慮する必要がある。例えば、Web ベースのアプリケーションでは、ホストマシンにブラウザーをインストールする必要がある。ブラウザーの種類とバージョンを考慮する必要がある。

4.3.3 テスト自動化データとインターフェース要件の定義

テスト自動化エンジニア（TAE）が自動テストスクリプトを開発する前に考慮すべき 2 つの事項は、テスト対象システム（SUT）とのインターフェース方法とテストケースのデータ依存関係である。以下に例を示す：

API：SUT が API を使用する場合、アプリケーションレベルで UI を必要とせず、エンドポイントレベルでテストできる。これは、UI によって隠されたエンドポイントとデータ通信のより深い理解が必要な場合がある。テスト自動化エンジニア（TAE）はビジネスプロセスを作成するために API を呼び出す順序と、テストケースをそのまま維持するためにデータを相関させる方法を理解する必要がある。インターネットに公開されている API は WebAPI や Web サービスとも呼ばれる。

データベースインターフェース：一部のテスト自動化ツールは、SUT の基盤となるデータベースと直接インターフェースできる。テストスクリプトは、格納されたプロシージャやその他のデータベースルールが正しく設定されていることを確認するために、列と行のデータを検証するよう作成できる。これは信頼性テストのために、特定のデータ値がデータベースにすでに存在する必要がある場合がある。

インターフェース互換性：契約テストの利用は、2 つの独立したシステム（例：2 つのマイクロサービス）が互換性があり、互いに通信できることを確保する。契約テストはコンシューマー駆動、プロデューサー駆動、双方向で行える。追加の詳細は CTAL-TAE シラバスのセクション 5.1.3 で確認できる。

さらに、TAE は SUT とのインターフェース方法を慎重に考慮し、テストケース内のデータ依存関係を理解する必要がある。API を通じてテストするか、データベースインターフェースを使用するか、またはシステム間のインターフェース互換性を確保するかにかかわらず、これらの考慮事項への注意が堅牢で効果的なテスト自動化の不可欠な要素である。これらの要因を思慮深く対処することで、TAE は自動化の信頼性と効率性を高め、最終的には SUT の品質と成功に貢献できる。

5 テスト自動化影響分析 – 150 分 (K3)

キーワード

カバレッジ、テストレポート

第 5 章の学習の目的：

5.1 テスト自動化の構築・維持への投資

CT-TAS-5.1.1 (K3) テスト自動化ソリューション (TAS) 構築の投資対効果 (ROI) を示す

5.2 テスト自動化メトリクス

CT-TAS-5.2.1 (K2) テスト自動化メトリクスを分類する

5.3 プロジェクトおよび組織レベルにおけるテスト自動化の価値

CT-TAS-5.3.1 (K3) テスト自動化利用に関する組織的考慮事項を特定する

CT-TAS-5.3.2 (K3) テスト自動化目標の最適な実装を判断するためのプロジェクト特性を分析する

5.4 テスト自動化レポートからの意思決定

CT-TAS-5.4.1 (K2) テストレポート (test report) データを分析して意思決定を支援する

5.1 テスト自動化の構築・維持への投資

5.1.1 テスト自動化ソリューション (TAS) 構築の投資対効果 (ROI) の算出

プロジェクトの実装を開始する前に、テスト自動化のセットアップと保守コストを見積もることが重要である。自動化がプロジェクトにもたらす価値に加えて、プロジェクトの ROI 計算を理解しておくことは有益である。

ROI 計算は、投資した工数に対する活動の見返りを示すことで、プロジェクトライフサイクルのどの時点でも意味のあるフィードバックを提供できる。ROI 計算は、費やした時間にそれぞれの人件費を掛けることで、手動テスト担当者コストとテスト自動化エンジニア (TAE) コストの両方を含めることでさらに調整できる。

ROI を計算するには、テスト自動化の投資（時間とコスト）とそれによって達成された削減効果を決定する必要がある：

ROI = 削減効果 / 投資

削減効果と投資は、異なるメトリクスとデータを考慮し、異なる指標と単位で計算できることに注意することが重要である。本シラバスの範囲内では、アプローチを示すために、コストではなく時間単位を使用した単純なモデルを使用する。特定の活動がコストのみで測定される場合、その金額はプロジェクト固有のレートを使用して時間に変換できる。

一般に、テスト自動化による削減効果は、同じテストを手動で実行するよりも大幅に短い時間で実行できることに起因する。これはまた、より頻繁に実行できることも意味する。したがって、実行されるテストの数を増やすことができる。

削減効果を計算するには、以下のメトリクスを考慮する必要がある：

- テストケースを手動で実行する時間
- 自動テストケースを実行する時間
- テストケースの数
- テスト実行の数

投資を計算するには、以下のメトリクスを考慮する必要がある：

- テスト自動化の設定時間
- 自動テストスクリプト開発の平均時間
- 実装された自動テストスクリプトの数
- 自動テストスクリプトの平均保守時間
- 自動テストスクリプトの実行時間
- 失敗した自動テストスクリプトの割合
- 定義されたテストケースの数
- テスト実行の数

概説されたモデルをアジャイルソフトウェア開発に適用することで、以下のグラフに示すようにプロジェクトのスプリントを予測できる。グラフを使用して、テスト自動化が投資を回収したスプリントを判断できる。

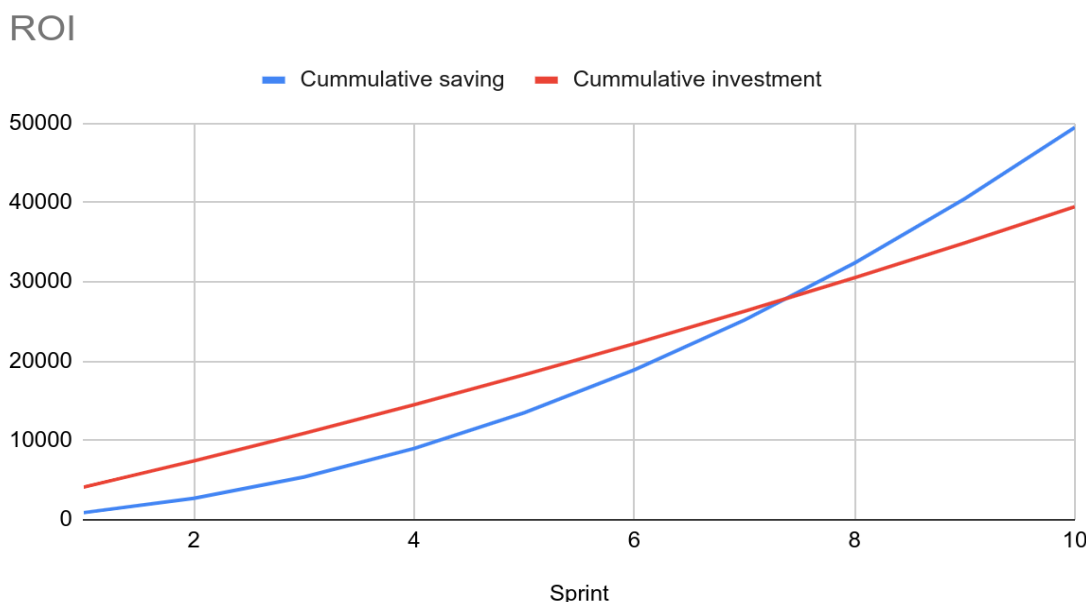


図 2 : ROI 計算の例（投資対効果の回収点を示す）

ROI を計算するために測定されたメトリクス間の特定の関連性に留意すること：

- 計画されたプロジェクト期間が ROI の転換点より短い場合、テスト自動化を導入する価値はない。この場合、テストを手動で実行することで時間と工数を節約できる。
- 投資は自動テストの実行時間に大きく依存するため、テストピラミッドを適用し、適切なテストレベルでテストケースを実装することで、この実行時間を短縮し ROI を向上できる。

5.2 テスト自動化メトリクス

5.2.1 テスト自動化メトリクスの分類

事実データに基づいたトレンド分析が意思決定を助ける。テスト自動化ソリューション（TAS）のメトリクスを収集することで、テスト担当者は TAS を評価し、以下について意思決定を行うことができる：

- TAS のプロジェクトへの適合性
- 新しいテスト条件に対応するための、機能拡張に対する TAS の適応性
 - テスト対象システム（SUT）におけるユーザーフローの変更
 - テストの実施方法の変更

- **TAS で発見された欠陥によるテスト自動化の保守性**

測定コストはできる限り低くする必要があり、メトリクスの収集と報告を自動化することで実現できる場合が多い。以下にいくつかの例を示す。

合格・失敗比率

これは一般的なメトリクスであり、期待される結果を達成できた（合格した）自動テストと達成できなかった（不合格になった）自動テストの比率を追跡する。 $\text{合格・不合格比率} = \text{合格したテストの数} / \text{不合格になったテストの数}$ 。

故障と欠陥の比率

自動テストの一般的な問題は、多くのテストが同じ理由で失敗することである。つまり、SUT の単一の欠陥である。特定の欠陥に対して失敗する自動テストの数を測定することで、これが問題になる可能性がある場所を示すのに役立つ。また、欠陥あたりの故障数を収集し、サブシステムの欠陥、システム全体の欠陥、テストデータの問題、またはインフラストラクチャの問題といった失敗の理由を追跡することもできる。

テスト自動化実行時間

より簡単に判断できるメトリクスの一つは、自動テストの実行にかかる時間である。TAS の開始時点ではあまり重要でないかもしれないが、自動テストケースの数が増えるにつれて、このメトリクスはかなり重要になる場合がある。このメトリクスには TAS のビルド時間も含まれる。

自動テストケースの数

このメトリクスはテスト自動化プロジェクトの進捗を示すために使用できる。しかし、自動テストケースの数はそれほど多くの情報を明らかにしないことを考慮する必要がある。例えば、カバレッジのレベルを示さない。

テスト自動化の機能カバレッジ

このカバレッジは自動テストケースでカバーされている機能要件の割合を示す。

コードカバレッジ

コードカバレッジは下位レベル（コンポーネント）テストによってコードの行数がどの程度テストされているかを追跡する。

適切なカバレッジを示す絶対的な割合はなく、最もシンプルなソフトウェア以外では 100% のコードカバレッジは達成が難しいことが多い。しかし、一般にカバレッジが多いほど SUT への信頼性が高まるため、より多くのカバレッジが良いとされる。下位レベルの自動テストが追加されるにつれて、コードカバレッジは増加することが期待される。

5.3 プロジェクトおよび組織レベルにおけるテスト自動化の価値

5.3.1 テスト自動化利用に関する組織的考慮事項の特定

任意のプロジェクトでテスト自動化を開始する前に、組織内で以下を特定する必要がある：

ソフトウェア開発のポリシーとプラクティス

開発チームがどのように作業しているか、どのような種類のドキュメントが存在するかを確認することは有益である。利用可能なドキュメントはすべて、開発チームのプロセスにテスト自動化をどのように接続できるかを特定するのに役立つ。ドキュメントには、SUT の技術仕様、使用されているソフトウェアと開発ツール、またはコードレビューガイドライン、定義されたコーディング標準、マージプロセスなどの開発に関する利用可能なポリシーが含まれる場合がある。

既存のアクティブなテスト自動化プロジェクトとそのステータス

継続中の開発プロジェクトの場合、異なるチームによる 1 つ以上の継続中の TAS 開発プロジェクトが存在する場合がある。意思決定者への一般的な推奨事項は、これらの既存プロジェクトとそのステータスを確認し、そのいずれかが新しい TAS の定義されたテスト目標に適合するかどうかを確認することである。ソリューションを分析することで、既存の TAS を再利用するか、現在のニーズに基づいて新しいものを作成するかを判断できる。

テスト自動化に関する組織の有識者

組織内で新しい TAS の展開を支援できる有識者（SME）を探すことが推奨される。SME は自分たちの TAS とその展開に関するストーリーと教訓を共有できる。これにより、TAS の展開を成功させるために考慮または回避すべきさまざまなリスクを特定できる。

テスト環境の利用可能性

大規模な組織の場合、既存のテスト環境、その使用状況、利用可能性に関する情報を収集することが一般的な推奨事項である。この情報は、通知や合意なしに新しい TAS を展開してシステムを使用することで他のチームの作業を妨げないために不可欠である。プロジェクトのニーズは新しいテスト環境を必要とすることが多いため、既存のものを新しいものを作成する際のベースラインとして使用でき、責任あるチームと担当者が特定され利用可能であれば、作業をより容易に行える。

テストツールとライセンス

組織が現在持っているツールとライセンスについて情報を得ることは常に価値がある。これらを特定することで、コストとタイムラインを削減し、TAS の計画を容易にできる。新しいプロジェクト向けのツールがすでに利用可能な場合もある。例えば、クラウドテストが定義されたプロバイダーを通じて設定されており、新しいプロジェクトにライセンスが利用可能な場合、異なるクラウドプロバイダーを使用することは意味がない。プロジェクトコストを削減するために、同じツールとライセンスを使用することが推奨される。

5.3.2 テスト自動化目標の最適な実装を判断するためのプロジェクト特性の分析

最適な作業方法を定義し、成功につながるテスト自動化目標の定義を助けるいくつかの主要なプロジェクト特性がある。

ドメイン

各ドメインが規制または標準において異なることを理解することが重要である。例えば、観光ドメインと金融や医療では、異なる規制とリスクがある。計画されたテスト自動化目標が準拠するように、さまざまな標準とドメインの制限を確認することを常に推奨する。

プラットフォーム

テスト自動化目標の観点から、プロジェクトがカバーするプラットフォームとどこにテスト自動化を適用するのが有益であるかを評価することも強調される。複数のプラットフォームのテスト自動化計画は、異なるプラットフォームをカバーするために複数のソリューションが必要になる場合があるため、より困難な場合がある。多くの場合、モバイルと Web など、同じツールを使用できる。しかし、計画によって TAS の再利用性が決まる。

プログラミング言語と技術スタック

プロジェクトのプログラミング言語や技術スタックなどの実装詳細も、組織が定義すべきテスト自動化目標の種類を決定する。開発者がプロジェクトで使用しているのと同じプログラミング言語を使用することが推奨される。そうすることで、コラボレーションがはるかに容易になり、コードレビューを共同で実施することによるクロスラーニングで SUT の品質と TAE の知識がさらに向上する。

プロジェクトの成熟度

プロジェクトの成熟度を分析することで、理想的なテスト自動化目標を設計するための情報を得ることができる。テストケースの数、既存の CI/CD パイプライン、テスト担当者と TAE の数などのさまざまな要因を特定することで、異なるテスト目標とロードマップを作成できる。例えば、高優先度の手動テストケースが多く、手動テスト実行時間が長い場合は、時間と工数を節約するためにまずこれらを自動化することを目指すことが重要である。上記の要因以外に、プロジェクトのタイムラインも非常に重要である。プロジェクトが短いまたは間もなく終了し、作業を完了する時間が少ない場合、大規模で堅牢な TAS を計画する価値はない。しかし、プロジェクトが新規開発プロジェクトであれば、段階的な最小実行可能製品 (MVP) ベースの段階的なロードマップを構築する価値がある。

ステークホルダーの賛同

多くの場合、主要なステークホルダーが賛同しないためにテスト自動化が活用されない。これは、目標ベロシティを下回る懸念、厳しい締め切り、予算配分、またはその他の懸念によるものである。プロジェクトの成熟度分析後、TAS を導入する余地がある場合、戦略的なステークホルダーは製品リスクを特定し、テスト自動化を導入するメリットをリストアップし、適切な計画を立てる必要がある。この計画では、テスト自動化で達成すべきマイルストーンを強調し、特定されたリスクに対処する TAS を成功裏

に導入するために SUT のテスト容易性をどのように改善する必要があるかを示す必要がある。最後に、この計画をステークホルダーに提示する必要がある。

チームの知識と関連経験

テスト自動化を成功させるための最も重要かつ関連性の高い要因はテスト担当者のスキルと経験である。テスト担当者と協力して、その知識を活用し、テスト担当者とビジネスアナリストが快適に感じるテスト目標を定義することは有益である。多くの場合、テストマネージャーやテストリーダーは、チーム内の利用可能な知識を特定し、同時にギャップを特定するために、コンピテンシーとスキルマトリックスを作成する。これらのギャップはトレーニングや割り当てられた実装タスクなど、さまざまな目標で特定できる。

テストマネジメントサポートと予算配分

プロジェクトの規模と成熟度に応じて、テスト自動化のテスト目標を定義する際に、利用可能な予算とテストマネジメントサポートを考慮する必要がある。テストマネジメントからのサポートを可能にするために、達成できるテスト目標を定義することが重要である。

テスト自動化戦略の提案を管理層に行う際、ドキュメントは簡潔で、現在特定されているギャップや、テスト自動化が正しく実装された場合の将来のコストを明確に定義する必要がある。ビジネス価値とコスト削減を指摘した推奨事項とそのメリットのリストは、テストマネジメント層が TAS の開発や改善を承認する動機づけとなる。

品質特性

ISO/IEC 25010:2011 は、ISTQB CTFL シラバスのセクション 2.2.2 「テストタイプ」に記載されている品質特性を定義している。これらの品質特性は、現在の TAS を評価したり、TAS が収集するメトリクスを決定したりするために使用できる。

5.4 テスト自動化レポートからの意思決定

5.4.1 意思決定を支援するテストレポートデータの分析

テスト自動化レポートのフォーマットと内容は、受け取るステークホルダーによって異なる場合がある。マネジメント、運用、または技術的なステークホルダー向けに作成できる。追加情報は CTAL-TAE シラバスのセクション 6.1.3 で確認できる。

テスト自動化レポートを受け取ると、より広いテストレポートに組み込まれるか、統合されて組織構造内でエスカレーションされる。

異なるステークホルダーはそのようなテスト自動化レポートに異なる価値を見出す。戦略的なアプローチは、関与するステークホルダーにとって重要な主要メトリクスを特定し、これらの重要なメトリクスを強調することである。

自動化で収集されたデータは以下を支援できる：

- 傾向の特定と根本原因分析の実施
- テスト自動化の工数を保守にシフトする
- テスト自動化フレームワーク（TAF）の改善と更なる開発にシフトする
- 全体的なテスト自動化ソリューション（TAS）に機能を追加する
- シフトレフトとシフトライトのテストへのアプローチを拡大する
- 将来のスプリントでテスト自動化の機能カバレッジを拡大する
- 欠陥クラスターへの重点化
- コード改善領域について開発者に助言する
- 全体的なソフトウェア開発ライフサイクル（SDLC）プロセスについて助言する
- 将来のテスト自動化レポートのコンテンツとフォーマットを変更する

上記で説明した情報に基づき、テスト自動化エンジニア（TAE）は他のステークホルダーと協力して、既存のテスト自動化カバレッジとテスト結果のギャップと特定の改善ポイントを特定できる。

6 テスト自動化の実装と改善戦略 – 150 分 (K3)

キーワード

カバレッジ、事前条件、テストスイート

第 6 章の学習の目的：

6.1 手動テストから継続的テストへの移行活動

CT-TAS-6.1.1 (K2) 手動テストからテスト自動化への移行における要因と計画活動を説明する

CT-TAS-6.1.2 (K2) テスト自動化から継続的テストへの移行における要因と計画活動を説明する

6.2 テスト自動化から継続的テストへの移行における要因と計画活動の理解

CT-TAS-6.2.1 (K3) テスト自動化資産とプラクティスの評価を実施して改善領域を特定する

6.1 手動テストから継続的テストへの移行活動

6.1.1 手動テストからテスト自動化への移行における要因と計画活動の説明

手動テストからテスト自動化への移行で最も容易な機会は、リグレッションテストをターゲットにすることである。今日の機能テストや非機能テストが明日のリグレッションテストとなることで、リグレッションテストスイートは成長していく。リグレッションテストの数が従来の手動テストチームが持つ時間とリソースより大きくなるのは時間の問題である。

移行コスト

手動テストから自動テストへの移行中、手動テストと自動テストが同時に行われるため、プロジェクトはコストの増加を見込む必要がある。自動テストが手動テスト実行活動を適切に代替または上回ると見なされたら、より多くの工数を探索的テストと自動化の追加テストケース作成にシフトすることができ、それは手動リグレッションテストと比較して異なるコスト構造を持つ。

機能的重複

機能的重複は、テストスクリプト開発者が異なるテストケースに同じテスト自動化ステップを含める場合に発生する。例えば、ほとんどのテストケースはログインシーケンスのテストステップで始まる。これにはユーザー名、パスワードの入力、ログインボタンの選択が含まれる。各テストケースにそれを追加すると保守活動が増加する。ログインプロセスにテストステップが追加されると、同じ変更がすべてのテストケースで更新される必要がある。より良いアプローチは、ログインプロセスから再利用可能なテスト自動化コンポーネントを作成し、すべてのテストケースがその機能を参照するようにすることである。CTAL-TAE シラバスのセクション 3.1.5 のフローモデルパターンの詳細を参照のこと。

データ共有

テストはしばしばテストデータを共有する。これは、テストが同じテストデータのレコードを使用して異なるテスト対象システム (SUT) の機能を実行する場合に発生する。この例として、従業員の利用可能な休暇時間を検証するテストケース「A」と、従業員がキャリア開発目標の一環として受講したコースを検証するテストケース「B」があるかもしれない。各テストケースは同じ従業員を使用するが、異なるパラメーターを検証する。手動テストでは、同一従業員を用いるテストケースごとに、従業員のテストデータが繰り返し利用されることが通常である。しかし、自動テストでは、共有されるテストデータは可能な限り重複や誤りの混入を避けるために、単一のソースから保存・アクセスされるべきである。

テストの相互依存性

複雑なリグレッションテストを実行する場合、一つのテストが一つ以上の他のテストに依存している場合がある。この発生はかなり一般的である。例えば、テストステップの結果として新しい「注文 ID」が作成される。後続のテストでは以下を検証したい場合がある：a) 新しい注文がシステムに正しく表示されている、b) 他の注文への変更が可能、c) 注文の削除が成功している。それぞれの場合、最初のテストで動的に作成された「注文 ID」の値は、後続のテストによる再利用のために取得されなければならない。テスト自動化ソリューション (TAS) の設計によっては、これに対処できる。「注文 ID」が後続のテストケースで見つからない場合、それらは失敗する。

テスト実行の事前条件

テスト自動化エンジニア（TAE）は、テストケースを確実に実行するための事前条件を最初に理解せず、テストスクリプトの開発をすぐに始めることがよくある。これは複数のテスト環境で同じ SUT に対してテストケースを実行しようとする場合に非常に困難になることがある。事前条件の例には、ユーザー名、アカウントの役割、データテーブルの値、テストケースを繰り返し実行可能にするその他の特定のデータエントリが含まれる。良い戦略には、特定のテストケースを自動化する前に SUT に存在する必要がある情報を理解するための事前分析が含まれる。さらに、実際のテストケースが実行される前に事前条件の作成を自動化することで時間を節約し、戦略を強化できる。これには、UI から SUT にデータを投入する事前条件テストスクリプトや、データベースにテストデータをロードする自動化バッチプロセスが含まれる場合がある。

機能カバレッジ

第 3 章で説明されているように、テスト自動化の候補となるテストの機能的な抜け漏れを特定する。手動テストケースの 100% を自動化したとしても、それは、テストツールで自動化できるすべてのテストケースの 100% を表すわけではない。より多くのテストが自動化されるにつれて、テスト担当者はテスト実行に費やしていた時間を取り戻し、カバレッジを向上させるための追加の SUT のテストを特定できるようになる。

実行可能テスト

手動リグレーションテストを自動リグレーションテストへ変換する前に、手動リグレーションテストが正しく動作することを確認することが重要である。これにより、自動リグレーションテストへの変換を確実に成功させる正しい出発点が提供される。手動リグレーションテストが正しく実行されない場合、それは書き方が悪い、無効なテストデータを使用している、現在の SUT と最新でないか同期していない、または SUT の欠陥によるものかもしれない。失敗の根本原因を理解・解決する前に自動化することは、機能しない自動リグレーションテストを作成してしまい、無駄で生産性がない。従来の手動テストを置き換える自動テストに対する信頼性を高めるためには、新たに導入された自動テストが、同等の機能を持っていることを示すことが重要である。

6.1.2 テスト自動化から継続的テストへの移行における要因と計画活動の説明

継続的テストは、従来のソフトウェア開発ライフサイクル（SDLC）よりもはるかに頻繁にテスト資産を活用する。これは、コード変更が完了してテスト環境で利用可能になった直後から、テストスイートを継続的に実行し始めることで行われる。これにより即時フィードバックが得られ、プロセスの早い段階で欠陥を検知することで欠陥コストが削減される。これは、高度な開発ツールと、ビルドプロセスの早い段階へテストを移す意欲があれば実現できる。

継続的テストのためのテスト自動化ソリューション（TAS）の適応には以下が必要である：

- テストスイートは更新された TAS で実行できるよう変更する必要がある：テストスイートに必要な変更を加え、TAS にデプロイメントする前にテストする
- テストで使用するスタブ、ドライバー、インターフェースは更新された TAS に合わせて変更する必要がある：テストハーネスに必要な変更を加え、TAS にデプロイメントする前にテストする
- インフラストラクチャは更新された TAS に対応するよう変更する必要がある：変更が必要なインフラストラクチャコンポーネントのアセスメントを行う
- 更新された TAS には追加の欠陥またはパフォーマンス欠陥がある：リスクとメリットの分析を実施する。発見された欠陥により TAS の更新が不可能な場合、更新を進めないか、次のバージョンの TAS を待つ方が良い場合がある。欠陥がその修正メリットと比較して軽微であれば、TAS はそれでも更新できる。
- テスト自動化エンジニア（TAE）や他のステークホルダーに通知し、欠陥がいつ修正されるかの見積もりを得るために、既知の欠陥のリリースノートを必ず作成する

このセクションで述べたすべての点は、CI/CD を活用する場合に特に重要になる。パイプラインをまとめてビルドプロセスを自動化することは、開発中の TAS と自然に適合する。ビルドオーケストレーションツールが正しいテスト環境にある場合、パイプラインを拡張してデプロイメント後すぐに SUT を検証する自動テストを含めることができる。これにより、テスト自動化ツールが適切に設定されて SUT にアクセスでき、コードリポジトリにアクセスしてデータプロビジョニングスクリプトにアクセスできるようにする必要がある。

6.2 組織全体のテスト自動化戦略

6.2.1 テスト自動化資産とプラクティスの評価による改善領域の特定

他の開発活動と同様に、テスト自動化ソリューション（TAS）の開発を一時停止し、ソリューションをリファクタリングする機会を探すための戦略を持つことが重要である。監視すべき領域は初期実装、保守、および再現性の観点からソリューションを評価する能力である。追跡するための良いカテゴリとして、TAS の開発に費やす時間数、TAS の修正に費やす時間数、手動テストと比較してテスト担当者が実現する時間節約がある。TAS の保守時間が手動テスト時間を上回っている場合は、何らかの問題がある兆候である。

TAS の最初のデプロイメントを始める前に、それが独自の環境で実行でき、ランダムな変更から隔離されており、テストケースが更新・管理できることを確認することが重要である。TAS とそのインフラストラクチャの両方を保守する必要がある。初回デプロイメントの場合、以下の基本的な手順が必要である：

- コードカバレッジツールはコンポーネントテストスイートによってコードの何%が実行されるか、また追加のコンポーネントテストでカバーできるギャップがどこにあるかを示す
- SUT の機能カバレッジは、どの機能がまだテストケースでカバーされていないかを明らかにする要件トレサビリティマトリックスを作成することで決定できる
- プロジェクト全体または組織全体で TAS の一貫したインフラストラクチャ使用を定義する
- テストスイートのための一貫した構成管理戦略を開発する
- 共通の TAS 開発ガイドラインを CTAL-TAE シラバスのセクション 3.1.4 で説明されているベストプラクティスを活用して作成する
- 事前条件の実装。テストは多くの場合、事前条件を設定する前には実行できない。これらの事前条件には、テストする正しいデータベースまたはテストデータの選択、初期値またはパラメーターの設定が含まれる場合がある。テストの事前条件を確立するために必要なこれらの初期化ステップの多くは自動化できる。これらの手順をテスト実行前に必ず実行することで、より信頼性が高く確実なソリューションを実現できる。リグレッションテストがテスト自動化に変換されるにつれて、これらの事前条件はテスト自動化プロセスの一部に含める必要がある。

新機能や保守目的のための段階的な TAS の更新が発生した場合、以下を考慮すべきである：

- TAS に拡張機能を提供できるテストツールやその他の新しいテストツールの更新を評価する
- TAS の機能とパフォーマンスをさらに最適化する方法を評価する
- 再利用性を強化するためにテストスクリプトをさらに分解・モジュール化する機会を特定する
- 再利用可能なコンポーネントの知識・認識と一貫した利用を確保する
- 潜在的な改善領域に関する証拠を収集し、推奨事項とそのメリットを提供する
- 機能的重複の領域を評価・修正する。既存のリグレッションテストを自動化する際、テストケース間の機能的重複を特定し、可能な限り以前に開発されたテスト自動化コンポーネントを再利用することは良い実践である。
- 自動化の機会を求めて追加の手動テストケースを評価し、実装のためのバックログアイテムを作成する
- テスト設計とテストデータ管理の改善機会を強調する
- すべての既存のテストスイートが TAS の最新バージョンに適応していることを確認する
- テスト自動化がパイプラインの滞留の原因となっている場合、統合テストの範囲を最も重要なものに絞る（つまり、スモークテストスイートを作成する）。より大きなリグレッションテストスイートは別々にトリガーするか、必要に応じて実行できる。

参考文献

標準

テスト自動化のための標準には以下が含まれるが、これに限定されない：

IEEE (Institute of Electrical and Electronics Engineers) による Automatic Test Markup Language (ATML) は以下で構成される：

- IEEE Std 1671.1 : テスト説明
- IEEE Std 1671.2 : 機器説明
- IEEE Std 1671.3 : UUT 説明
- IEEE Std 1671.4 : テスト設定説明
- IEEE Std 1671.5 : テストアダプタ説明
- IEEE Std 1671.6 : テストステーション説明
- IEEE Std 1641 : 信号とテスト定義
- IEEE Std 1636.1 : テスト結果

ISO/IEC 30130:2016 (E) ソフトウェアエンジニアリング — ソフトウェアテストツールの能力

ETSI (European Telecommunication Standards Institute) および ITU (International Telecommunication Union) による Testing and Test Control Notation (TTCN-3) は以下で構成される：

- ES 201 873-1: TTCN-3 コア言語
- ES 201 873-2: TTCN-3 表形式プレゼンテーション形式 (TFT)
- ES 201 873-3: TTCN-3 グラフィカルプレゼンテーション形式 (GFT)
- ES 201 873-4: TTCN-3 オペレーショナルセマンティクス
- ES 201 873-5: TTCN-3 ランタイムインターフェース (TRI)
- ES 201 873-6: TTCN-3 コントロールインターフェース (TCI)
- ES 201 873-7: TTCN-3 での ASN.1 の使用
- ES 201 873-8: TTCN-3 での IDL の使用
- ES 201 873-9: TTCN-3 での XML の使用
- ES 201 873-10: TTCN-3 ドキュメント
- ES 202 781: 拡張機能：設定とデプロイメントサポート
- ES 202 782: 拡張機能：TTCN-3 パフォーマンスとリアルタイムテスト
- ES 202 784: 拡張機能：高度なパラメーター化
- ES 202 785: 拡張機能：動作タイプ
- ES 202 786: 拡張機能：連続信号を持つインターフェースのサポート
- ES 202 789: 拡張機能：拡張 TRI

OMG (Object Management Group) による UML Testing Profile (UTP) は以下のテスト仕様概念を規定する：

- テストアーキテクチャ
- テストデータ
- テスト動作
- テストログ
- テストマネジメント

ISTQB®ドキュメント

識別子	参照
ISTQB-AL-TM	ISTQB Certified Tester, Advanced Level Syllabus, Test Manager, Version 2.0, October 2012, available from [ISTQB-Web]
ISTQB-EL-TM-MTT	ISTQB Certified Tester, Expert Level Test Management Managing the Test Team, Version 2.0, November 2011, available from [ISTQB-Web]
ISTQB-FL	ISTQB Certified Tester, Foundation Level Syllabus, Version 4.0, April 2023, available from [ISTQB-Web]
ISTQB-PT	ISTQB Certified Tester, Performance Testing Syllabus, December 2018, available from [ISTQB-Web]
ISTQB-TAE	ISTQB Certified Tester, Test Automation Engineering Syllabus, February 2024, available from [ISTQB-Web]
ISTQB-Glossary	ISTQB Glossary of terms, available online from [ISTQB-Web]

書籍

Paul Baker、Zhen Ru Dai、Jens Grabowski、Ina Schieferdecker 著、「Model-Driven Testing: Using the UML Testing Profile」、Springer 2008 年版、ISBN-10: 3540725628、ISBN-13: 978-3540725626
Efriede Dustin、Thom Garrett、Bernie Gauf 著、「Implementing Automated Software Testing: how to save time and lower costs while raising quality」、Addison-Wesley、2009 年、ISBN 0-321-58051-6
Efriede Dustin、Jeff Rashka、John Paul 著、「Automated Software Testing: introduction, management, and performance」、Addison-Wesley、1999 年、ISBN-10: 0201432870、ISBN-13: 9780201432879
Mark Fewster、Dorothy Graham 著、「Experiences of Test Automation: Case Studies of Software Test Automation」、Addison-Wesley、2012 年
Mark Fewster、Dorothy Graham 著、「Software Test Automation: Effective use of test execution tools」、ACM Press Books、1999 年、ISBN-10: 0201331403、ISBN-13: 9780201331400
Boby Jose 著、「Test Automation, a manager's guide」、2021 年 9 月、ISBN: 9781780175478
James D. McCaffrey 著、「.NET Test Automation Recipes: A Problem-Solution Approach」、APRESS、2006 年、ISBN-13:978-1-59059-663-3、ISBN-10:1-59059-663-3
Daniel J. Mosley、Bruce A. Posey 著、「Just Enough Software Test Automation」、Prentice Hall、2002 年、ISBN-10: 0130084689、ISBN-13: 9780130084682
Casey Rosenthal 著、「Chaos Engineering: System Resiliency in Practice by Casey Rosenthal」、2020 年 4 月、ISBN-13: 1492043869
Colin Willcock、Thomas Deiß、Stephan Tobies、Stefan Keil 著、「An Introduction to TTCN-3」、Wiley、第 2 版、2011 年、ISBN-10: 0470663065、ISBN-13: 978-0470663066

記事

Robert V. Binder、Suzanne Miller 著、「Five Keys to Effective Agile Test Automation for Government Programs」、2017 年 8 月 24 日、Software Engineering Institute、Carnegie Mellon University、 https://resources.sei.cmu.edu/asset_files/Webinar/2017_018_101_503516.pdf
DoD CIO、Modern Software Practices、「DevSecOps Fundamentals Guidebook: Activities & Tools」、Version 2.2、2023 年 5 月、 https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsActivitesToolsGuidebookTables.pdf?ver=_Sylg1WJB9K0Jxb2XTvzDQ%3d%3d
Naveen Jayachandran 著、「Understanding roi metrics for software test automation」、2005 年、 https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=3937&context=etd
Thomas Pestak、William Rowell、PhD 著、「Automated Software Testing Practices and Pitfalls」、2018 年 9 月、 https://www.afit.edu/stat/statcoe_files/Automated%20Software%20Testing%20Practices%20and%20Pitfalls%20Rev%201.pdf
Andrew Pollner、Jim Simpson、Jim Wisnowski 著、「Automated Software Testing Implementation Guide for Managers and Practitioners」、2018 年 10 月、 https://www.afit.edu/stat/statcoe_files/0214simp%20%20AST%20IG%20for%20Managers%20and%20Practitioners.pdf

付録 A – 学習の目的／知識の認知レベル

以下の学習の目的は本シラバスに適用されるものとして定義されている。シラバスの各トピックはそれに対する学習の目的に従って試験される。

学習の目的は以下に記載されている知識の認知レベルに対応する行動動詞から始まる。

レベル 2：理解（K2）

受験者はトピックに関連した記述の理由または説明を選択でき、テストの概念について要約、比較、分類、例示ができる。

行動動詞：分類する、比較する、区別する、識別する、説明する、例示する、解釈する、要約する

例	注記
テストツールをその目的とサポートするテスト活動に従って分類する。	
異な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。
テストとデバッグの違いを識別する。	概念間の違いを探す。
プロジェクトリスクと製品リスクを区別する。	2つ（またはそれ以上）の概念を個別に分類できる。
コンテキストがテストプロセスに与える影響を説明する。	
テストが必要な理由の例を挙げる。	
与えられた故障のプロファイルから欠陥の根本原因を推測する。	
ワークプロダクトレビュープロセスの活動を要約する。	

レベル 3 : 適用 (K3)

受験者は慣れ親しんだタスクに直面したときに手順を実施できるか、正しい手順を選択して特定のコンテキストに適用できる。

行動動詞 : 適用する、実装する、準備する、使用する

例	注記
境界値分析を適用して、与えられた要件からテストケースを導出する。	手順/技法/プロセスなどを参照すべきである。
技術要件と管理要件をサポートするためにメトリクス収集方法を実装する。	
テスト自動化環境を準備する。	
トレーサビリティを使用して、テスト目標、テスト戦略、テスト計画との完全性と一貫性についてテストの進捗を監視する。	受験者がある技法や手順を使用できることを望む LO に使用できる。「適用する」に類似。

参照

(学習の目的の認知レベルの場合)

Anderson, L. W. および Krathwohl, D. R. (編) (2001 年) 「A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives」、Allyn & Bacon

付録 B – 学習の目的とのビジネス成果トレーサビリティマトリックス

本セクションは、テスト自動化戦略スペシャリストのビジネス成果とテスト自動化戦略スペシャリストの学習の目的間のトレーサビリティを示す。

ビジネス成果 テスト自動化戦略スペシャリスト			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
第 1 章	テスト自動化戦略の導入と目標																
1.1	テスト自動化プロジェクトの成功要因																
1.1.1	テスト自動化の目標と関連性の説明	2	X														
1.1.2	テスト自動化プロジェクトの技術的成功要因の特定	2	X														
1.1.3	テスト自動化の候補プロジェクト選定における適切な投資基準の要約	2	X														
第 2 章	テスト自動化リソース																
2.1	テスト自動化ソリューション（TAS）導入のコストとリスク																
2.1.1	総保有コストの観点から代替技術ソリューションを比較する	2		X													
2.1.2	テスト自動化ツールのライセンスモデルに関する考慮事項を説明する	2		X													
2.1.3	テスト自動化戦略定義時に考慮すべき要素の例を提供する	2		X													
2.2	テスト自動化における役割と責任																
2.2.1	テスト自動化ソリューション（TAS）の成功に必要な役割とスキルを要約する	2			X												

ビジネス成果 テスト自動化戦略スペシャリスト		B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
第3章	テスト自動化の準備															
3.1	テストレベルをまたいだ統合															
3.1.1	テスト自動化の配分を区別する	2			X											
3.1.2	テスト対象システム（SUT）のアーキテクチャに基づくテスト自動化アプローチの選択	3			X											
3.1.3	シフトレフトおよびシフトライトアプローチを達成するためのテスト自動化配分の最適化方法の実証	2			X											
3.2	異なるソフトウェア開発ライフサイクル（SDLC）モデルにおける戦略的考慮事項															
3.2.1	テスト自動化プロジェクトがレガシーソフトウェア開発ライフサイクル（SDLC）モデルにどのように準拠するかを説明する	2				X										
3.2.2	テスト自動化プロジェクトがテスト自動化をサポートするアジャイルソフトウェア開発ベストプラクティスにどのように準拠するかを説明する	2				X										
3.2.3	継続的テストにおけるテスト自動化をサポートするDevOps ベストプラクティスにテスト自動化プロジェクトが準拠するよう準備する	3				X										
3.3	テスト自動化の適用可能性と実現可能性															
3.3.1	テスト自動化に適したテストの適合性判断基準を説明する	2					X									
3.3.2	テスト自動化のみが対処できる課題を特定する	2					X									
3.3.3	自動化が困難なテスト条件を特定する	2					X									

ビジネス成果 テスト自動化戦略スペシャリスト			B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
第4章	テスト自動化の組織的デプロイメントとリリース戦略																
4.1	テスト自動化ソリューション（TAS）の計画																
4.1.1	テスト自動化が市場投入期間を短縮する方法を特定する	2							X								
4.1.2	要件に基づいて報告された欠陥を検証するテスト自動化の支援方法を特定する	2							X								
4.1.3	テスト自動化のための運用関連シナリオの開発を可能にするアプローチを定義する	2							X								
4.2	デプロイメント戦略																
4.2.1	テスト自動化デプロイメント戦略を定義する	2								X							
4.2.2	デプロイメントにおけるテスト自動化リスクを特定する	2								X							
4.2.3	デプロイメントリスクを軽減するアプローチを定義する	2								X							
4.3	テスト環境内の依存関係																
4.3.1	テスト環境内のテスト自動化コンポーネントを定義する	2									X						
4.3.2	テスト自動化のインフラストラクチャコンポーネントと依存関係を特定する	2									X						
4.3.3	テスト対象システム（SUT）内での統合のためのテスト自動化データとインターフェース要件を定義する	2									X						
第5章	テスト自動化影響分析																

ビジネス成果 テスト自動化戦略スペシャリスト		B 0 1	B 0 2	B 0 3	B 0 4	B 0 5	B 0 6	B 0 7	B 0 8	B 0 9	B 1 0	B 1 1	B 1 2	B 1 3	B 1 4	B 1 5
5.1	テスト自動化の構築・維持への投資															
5.1.1	テスト自動化ソリューション（TAS）構築の投資対効果（ROI）を示す	3									X					
5.2	テスト自動化メトリクス															
5.2.1	テスト自動化メトリクスを分類する	2										X				
5.3	プロジェクトおよび組織レベルにおけるテスト自動化の価値															
5.3.1	テスト自動化利用に関する組織的考慮事項を特定する	3											X			
5.3.2	テスト自動化目標の最適な実装を判断するためのプロジェクト特性を分析する	3											X			
5.4	テスト自動化レポートからの意思決定															
5.4.1	意思決定を支援するテストレポートデータを分析する	2												X		
第 6 章	テスト自動化の実装と改善戦略															
6.1	手動テストから継続的テストへの移行活動															
6.1.1	手動テストからテスト自動化への移行における要因と計画活動を説明する	2													X	
6.1.2	テスト自動化から継続的テストへの移行における要因と計画活動を説明する	2													X	
6.2	組織全体のテスト自動化戦略															
6.2.1	テスト自動化資産とプラクティスを評価し改善領域を特定するための評価を実施する	3														X

付録 C – リリースノート

ISTQB® Test Automation Strategy シラバス
2024 は、2016 年に公開された従来の ISTQB
Test Automation Engineer シラバスにおける戦
略的側面を取り込みつつ、テスト自動化の導入
および成功の測定に関する最新のベストプラク
ティスや更新内容を加えて構成された、新しい
ISTQB シラバスである。そのため、章やセクシ
ョンごとの詳細なリリースノートは提供されて
いない。

付録 D – ドメイン固有用語

用語名	定義
カナリアリリース	リスクの軽減と新機能の検証を目的として、一部の限定されたユーザーにのみ先行リリースするデプロイメントおよびテスト戦略
コンテナ	コードとその依存関係をパッケージ化したソフトウェアの単位で、ソフトウェアが環境全体で迅速かつ確実に実行されるようにする。
DevOps	ソフトウェア開発と IT 運用の作業を統合・自動化することで、ソフトウェア開発ライフサイクル（SDLC）を改善・短縮する方法論。
フローモデルパターン	業務ドメイン（対象領域）の全体像、それを構成する要素、および要素間の相互接続関係