

テストを学び成長する

～テストの全体観を持ち、テストに強みをもった技術者になるための学びと育成について～

2021年1月22日(金)
於 JaSST ' 21 Hokuriku

自己紹介

JaSST東京実行委員/ASTERテストプロセス改善WG
テスト設計コンテストU30審査員/29119勉強会
JSTQB FL研修講師/テスト自動化研究会コミッター
探索的テスト研究会/WACATE二代目実行委員長などなど



山崎 崇

@yamasaki696
株式会社ベリサーブ

A man with a blonde curly wig and glasses is speaking into a black microphone. He is wearing a dark blue hoodie with a purple graphic that says "milk tub" and a green lanyard with a badge. The background is a blurred indoor setting with a ceiling light.


特技ソフトウェアテスト

趣味ソフトウェアテスト

Yes, I'm loving Software Testing!

この講演のゴール

1. ソフトウェアテストの本質とはなんであるかを理解している
2. ソフトウェアテストについての全体像を俯瞰し、整理できる軸を識別している
3. テストに強みを持ったエンジニアになる・なってもらうための1事例を知っている



テストがなぜ必要なの？
テストって何だろう？
テストの目的とは？

ソフトウェアは人が作り、人は間違える生物



誘因① 納期のプレッシャー

誘因② 人間の誤りを犯しやすい性質

誘因③ プロジェクト参加者の経験不足または技術不足

誘因④ プロジェクト参加者間の誤ったコミュニケーション

※要件や設計の誤ったコミュニケーションを含む

誘因⑤ 複雑さ

※ コード、設計、アーキテクチャ、解決すべき根本的な問題、そして/または使用する技術の複雑さ

誘因⑥ システム内またはシステム間のインターフェースに対する誤解

※ 特に、関連するシステム数が多い場合

誘因⑦ 新しく不慣れな技術 など

ハードウェア

物質的実体
目に見える
触れられる

物理化学法則に基づ
くので自由度は低い

ソフトウェア

論理的集合体

目に見えない

触れられない

物理化学法則に基づ
かないので自由度が
高すぎる

```
default="v",
)
global_scale_setting = FloatProperty(
    name="Scale",
    min=0.01, max=100.0,
    default=1.0
)

def execute(self, context):

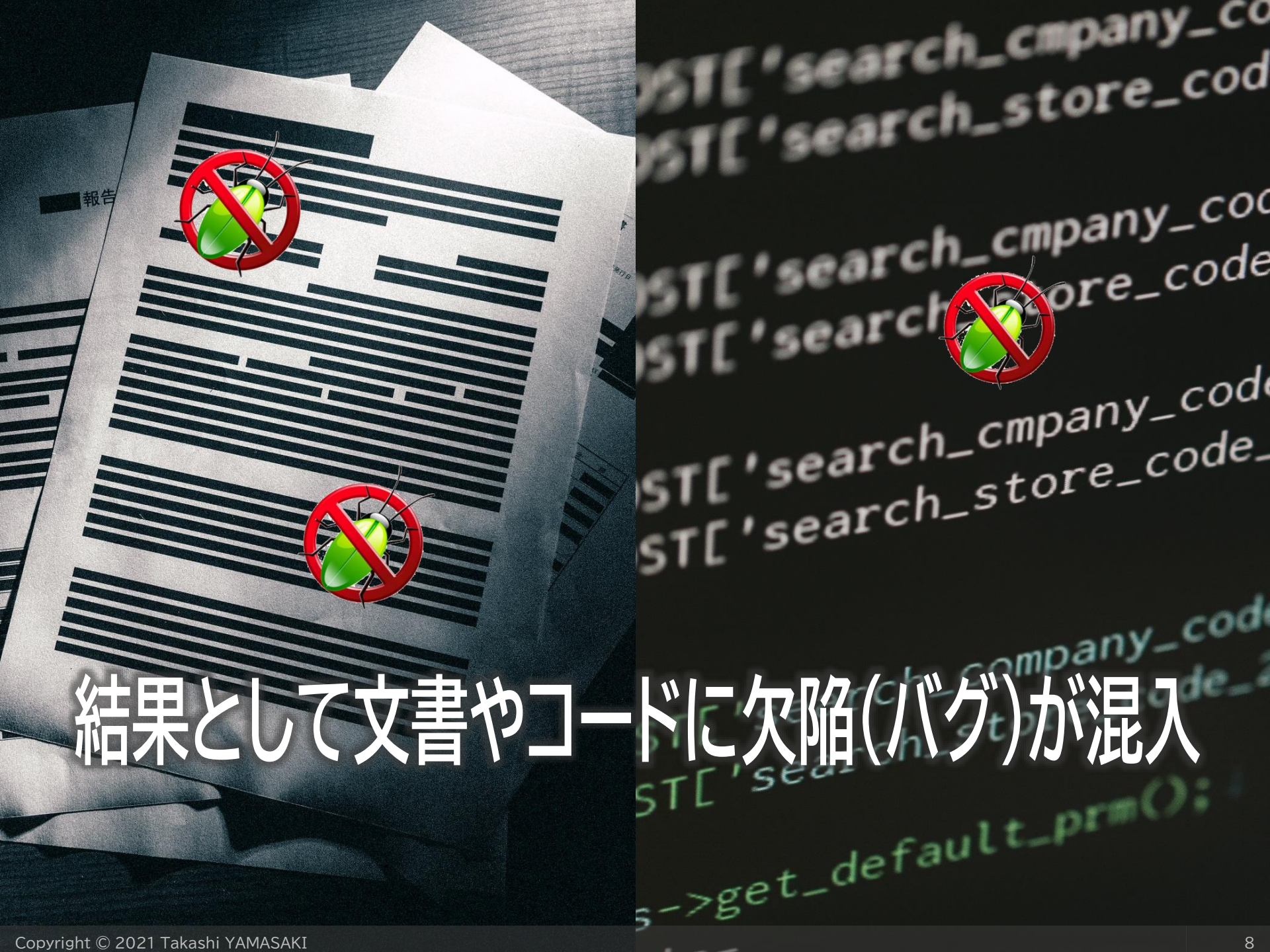
    # get the folder
    folder_path = (os.path.dirname(self.filepath))

    # get objects selected in the viewport
    viewports = bpy.context.viewports
    selected_objects = bpy.context.selected_objects

    # get export objects
    obj_export_list = []
    for obj in bpy.context.selected_objects:
        if obj.type == 'MESH':
            obj_export_list.append(obj)

    # deselect all objects
    bpy.ops.object.select_all(action='DESELECT')

    for item in obj_export_list:
        item.select = True
        if item.type == 'MESH':
            file_path = os.path.join(folder_path, "{}.obj".format(item.name))
            bpy.ops.export_scene.obj(
                filepath=file_path,
                axis_up=self.axis_up,
                use_edges=self.use_edges,
                use_smooth_groups=self.use_smooth_groups,
                use_normals=self.use_normals,
                use_uv=self.use_uv,
            )
```

結果として文書やコードに欠陥(バグ)が混入

JSTQBの用語集での定義(testing)

”全てのライフサイクルを通じて実施する静的、動的なプロセスにおいて、成果物が特定の要件を満足するかを判定し、目的に合致することを実証し、欠陥を見つけるため、ソフトウェアプロダクトや関連成果物に対し、計画、準備、評価をすること。”

テストの目的も一義的ではない

欠陥を防ぐため、要件、ユーザーストーリー、設計、およびコードなどの作業成果物を評価する

明確にしたすべての要件を満たしていることを検証する

テスト対象が完成したことを確認し、ユーザーやその他ステークホルダーの期待通りの動作内容であることの妥当性確認をする

テスト対象の品質に対する信頼を積み重ねて、所定のレベルにあることを確証する

欠陥や故障を発見し、ソフトウェアの品質が不適切になるリスクレベルを軽減する

ステークホルダーが意志決定できる、特にテスト対象の品質レベルについての十分な情報を提供する

契約上、法律上、または規制上の要件や標準を遵守する、そして/またはテスト対象がそのような要件や標準に準拠していることを検証する

テストとは？

「品質に関わる新たな 情報を得るための諸活動」

※以下のテストの目的をすべて満たします

欠陥を摘出する

対象ソフトウェアの
品質レベルが十分で
あることを確認する

意志決定のための
情報を示す

欠陥の作り
こみを防ぐ

上記4つのテストの目的は、テスト技術者資格制度 Foundation Levelシラバス日本語版 Version 2011.J02より引用 (JSTQB FL)

本質的に**テスト**の価値は**情報収集・提供**

早い

テストから得られる情報の
タイミングが早いほど価値が高い

安い

テストから得られる情報の
取得コストが安いほど価値が高い

旨い

テストから得られる情報がニーズに合致
していて正確で分かり易いほど価値が高い

テストを通じて得られた情報によって製品やサービスの品質に対する確信度合いを積み上げていく



そのための手段がテストだが、テストは本質的に完全ではない

原則 1

テストは欠陥があることは示せるが、**欠陥がないことは示せない**

原則 2

全数テストは不可能

原則 3

早期テストで時間とコストを節約

原則 4

欠陥の偏在

原則 5

殺虫剤のパラドックスにご用心

原則 6

テストは状況次第

原則 7

「バグゼロ」の落とし穴

A black and white photograph of a person walking on a track. The track is marked with a grid of white lines on a dark surface. The person is in the center, walking away from the viewer. The lines create a sense of depth and perspective.

情報はそれを使うことで意味を持つ

テストはテストのみでは存在しえない活動
(テストのことだけ考えても意味がない)

テストとは、テストやテストに
関係するテスト以外のことに
についても考えに考え抜いて
それらの本質を知ること



テストを学んで成長しよう





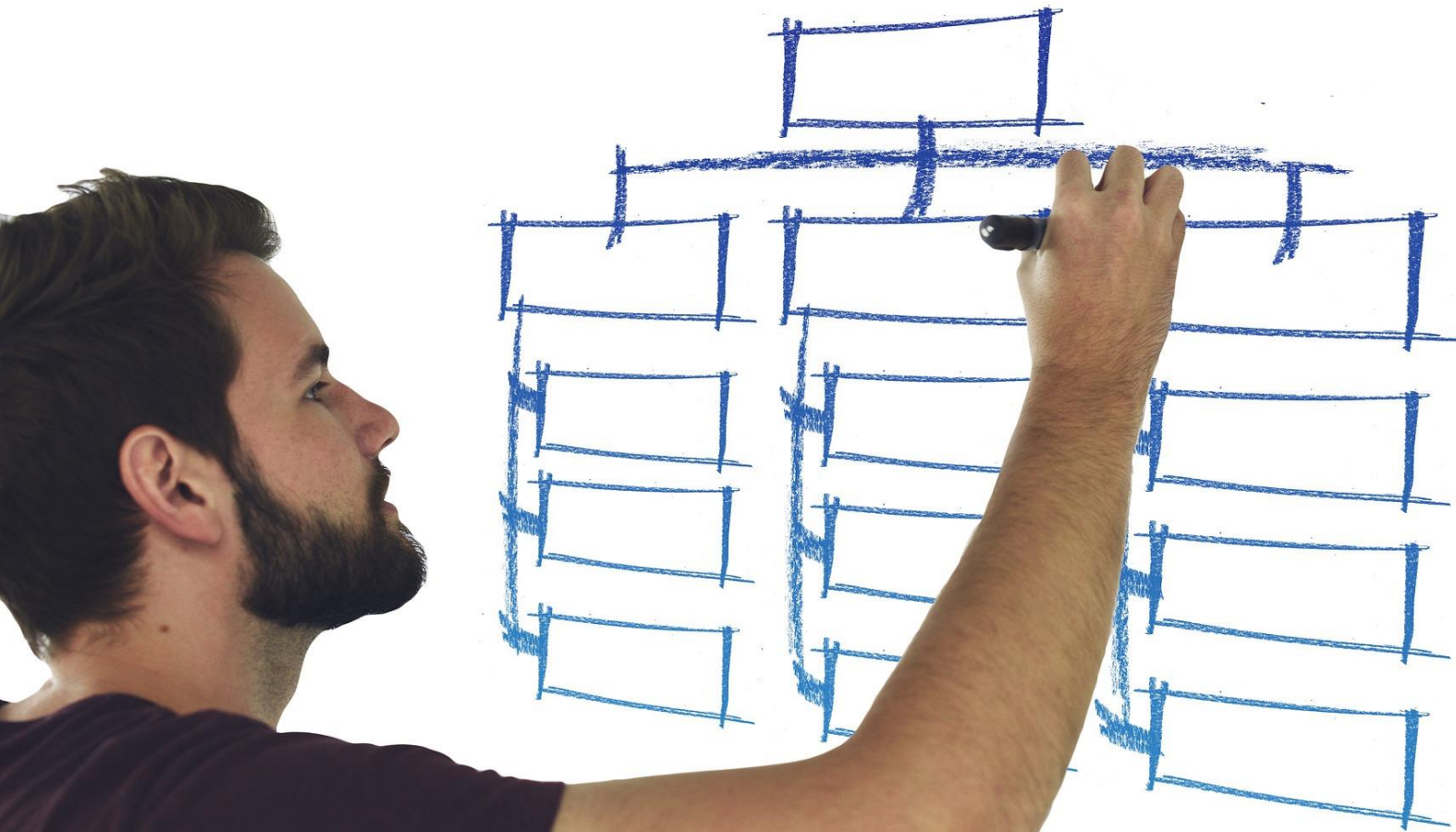
理解への**第一歩**は
言葉を知ることから

でもテストの領域は「●●テスト」や「テスト●●」だらけ…

デシジョンテーブルテスト
統合テスト テスト駆動開発 ブラックボックステスト
性能テスト 負荷テスト
システムテスト スモークテスト テストデータ
ベータテスト テストウェア ユニットテスト
非機能テスト テストベース
機能テスト テストツール テスト戦略 テスト環境
モンキーテスト テスト技法 グレーボックステスト
受け入れテスト ホワイトボックステスト 探索的テスト
アルファテスト ステートメントテスト
セキュリティテスト テスト設計 ロードテスト 結合テスト
パステスト テスト条件 テスト実装
テスト分析 テスト実行 テストフレームワーク
テスト管理 テストアプローチ テスト手順 テストケース
テスト報告
コンパチビリティテスト テスト計画 ペネトレーションテスト

「分かることは分けられること」

「思考・論理・分析 ―「正しく考え、正しく分かること」の論理と実践―」 波頭 亮(著)



言葉を理解して**区別**できるようになりましょう

同じ語を使っているにも異なる意味を想像していることも多いため
コミュニケーションを取る時には注意が必要です
常に相手が異なる意味で使っていないかを疑って適宜認識を合わせて
いかないと合意形成が上手くいかずに苦しむ事になりがちです



共通言語としてJSTQB FL 取得のススメ



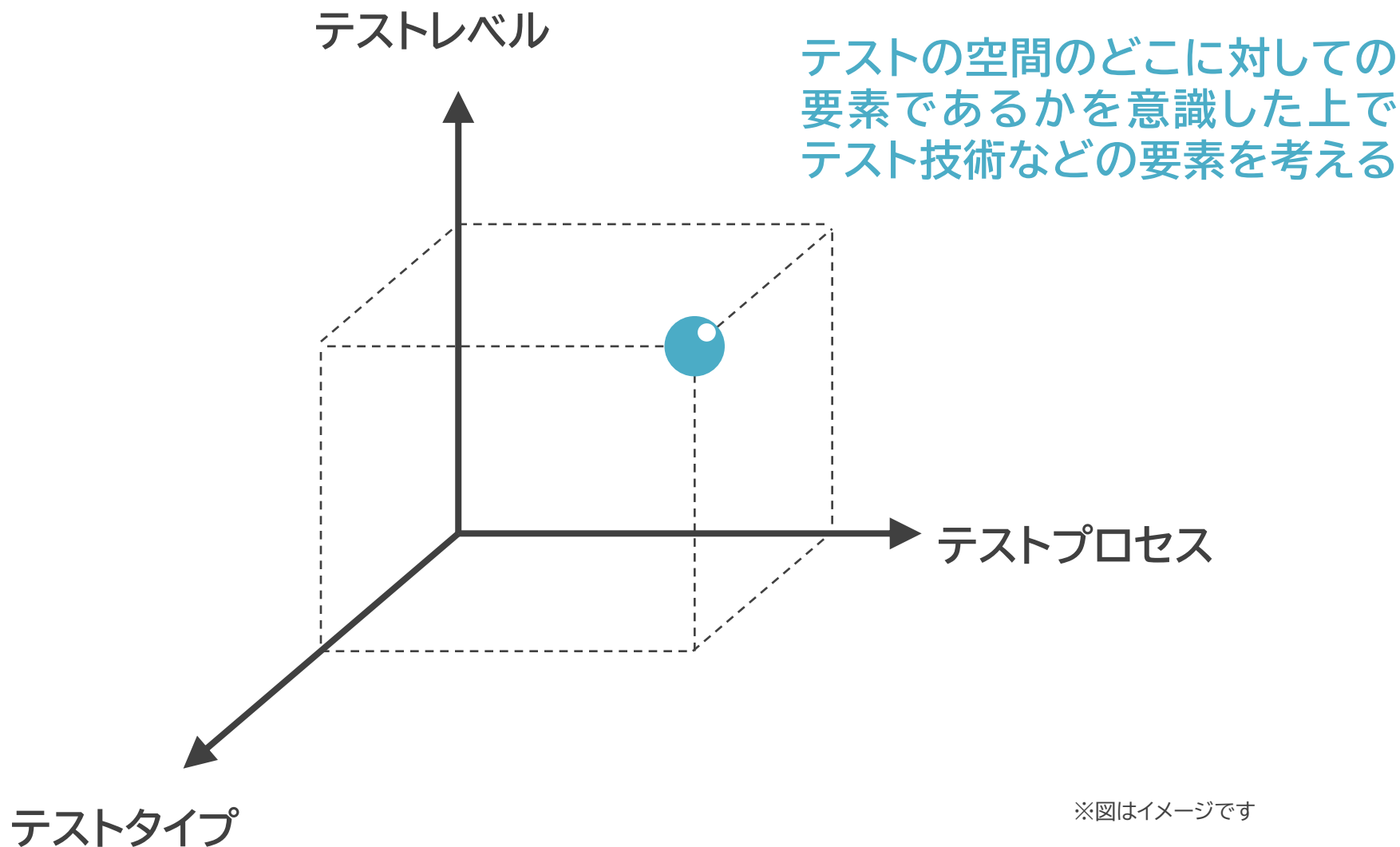
ソフトウェアテスト技術者の認定資格で、Foundation Levelは基礎レベルを意味します

ISTQBを通じて国際的にも有効な技術資格で、日本でも多くの資格所有者がいます

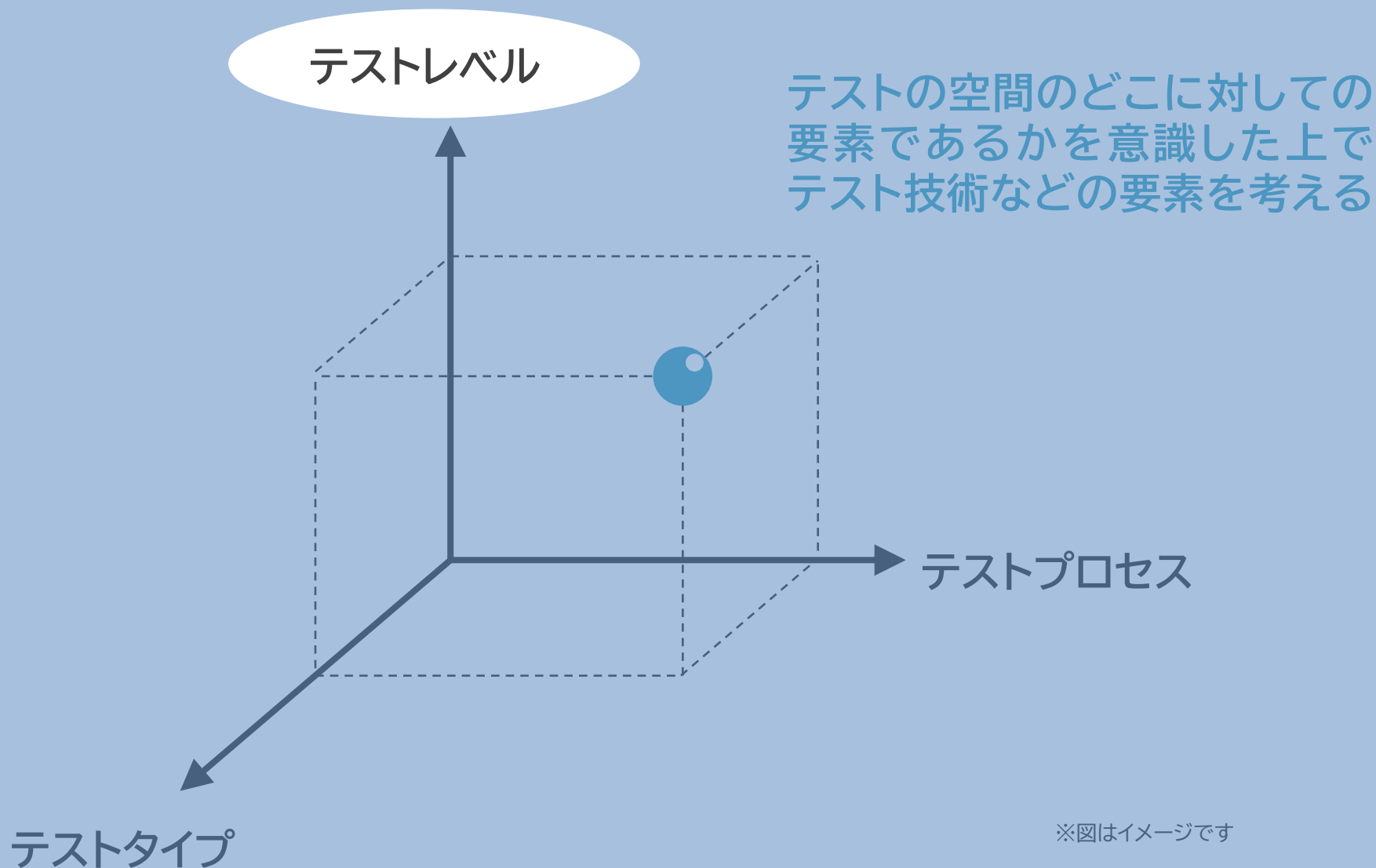
JSTQBの言葉を借りてコミュニケーションをとると、合意形成が捗ります

また、軸を持った上で学習すると、学習効率も高まります

分けるためにも概観できるテストの軸が重要



分けるためにも概観できるテストの軸が重要



テストレベルとは？

“

テストレベルは、系統的にまとめ、マネジメントしていくテストの活動のグループである。

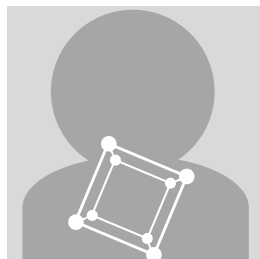
テスト技術者資格制度 Foundation Levelシラバス日本語版Version 2018V3.1.J02より

“

具体的にインスタンス化したテストプロセス。

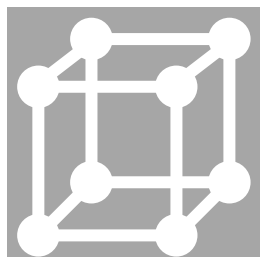
ISTQB Glossary V3.5 <https://glossary.istqb.org> より

小さなことからコツコツと品質に関する 確信度合いを積み上げていく



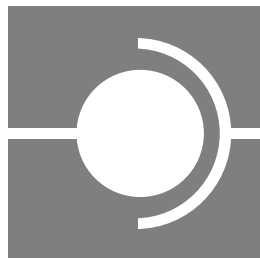
受入れテスト

開発したシステムがユーザの求めているものに合致しているか確認する(受け入れ可能かどうか判断する)テスト



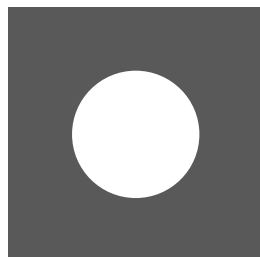
システムテスト

最終的に統合されたシステム全体を、実環境(またはそれに準ずる環境)で実施するテスト



統合テスト

テスト済みのコンポーネントやシステムを統合して行うテスト



コンポーネントテスト

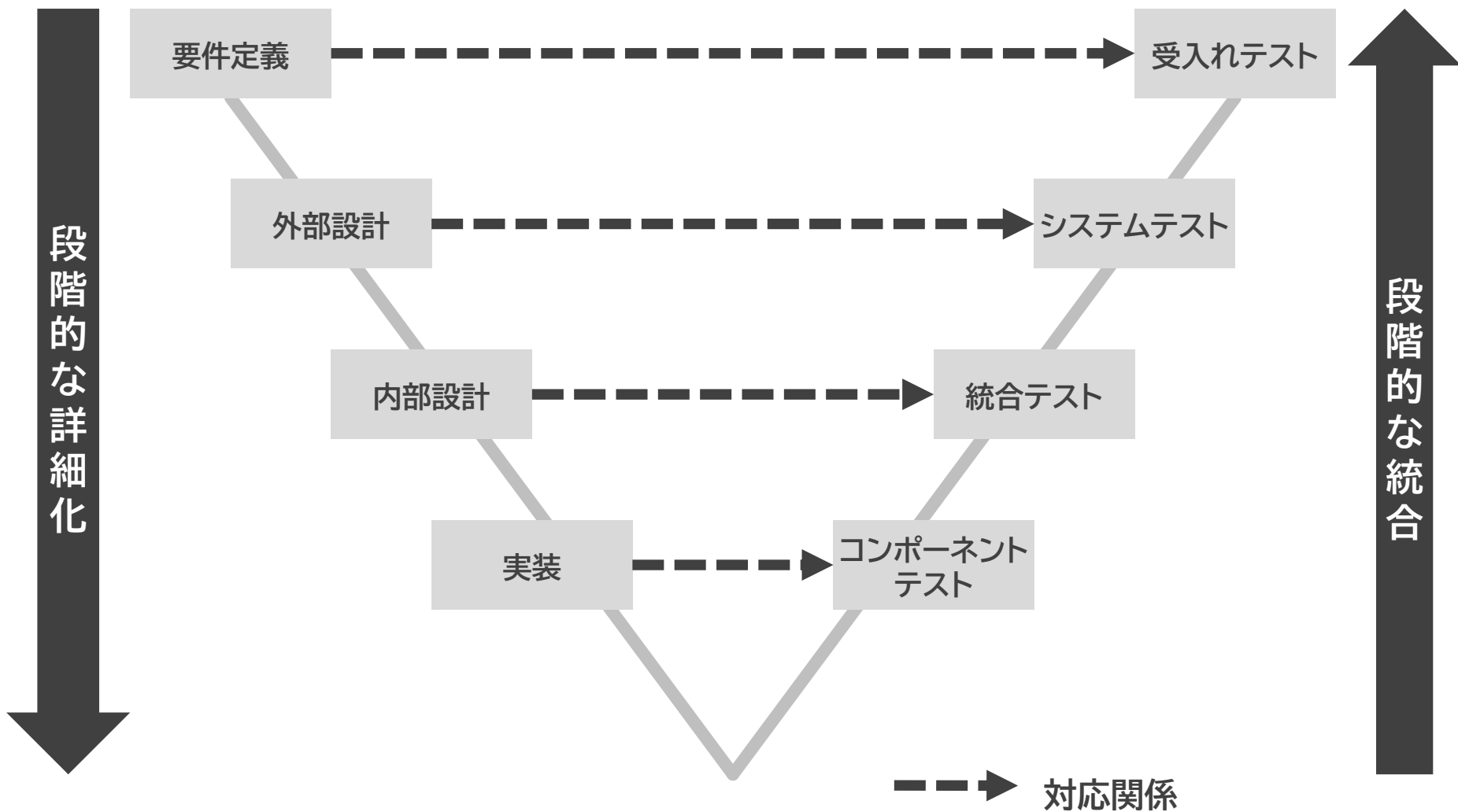
分離してテストが可能なソフトウェアコンポーネント(部品)ごとに単体で行うテスト

大

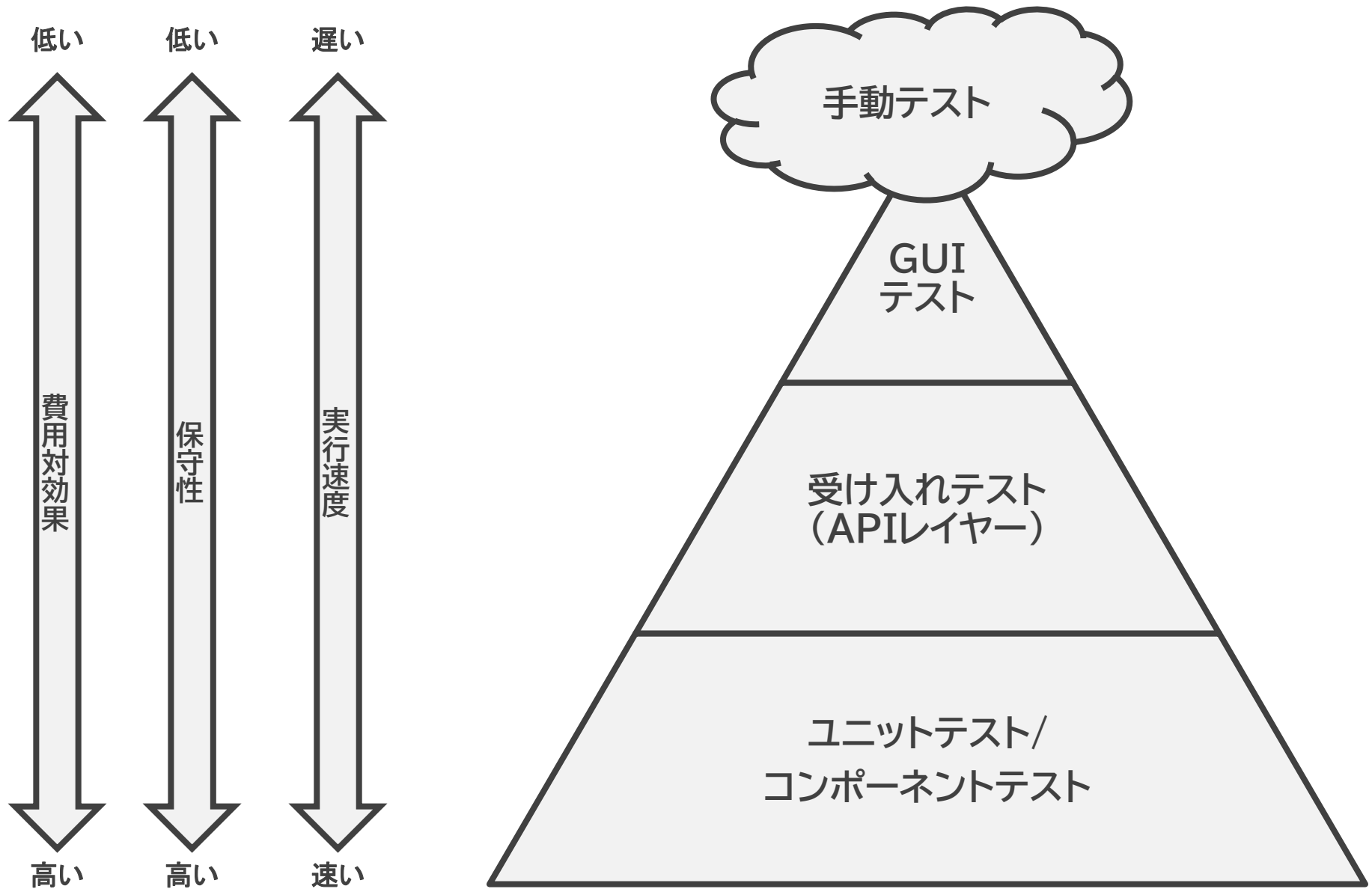
テスト対象として着目する詳細度

小

上流工程の開発と下流工程のテストにおける責務との対応関係

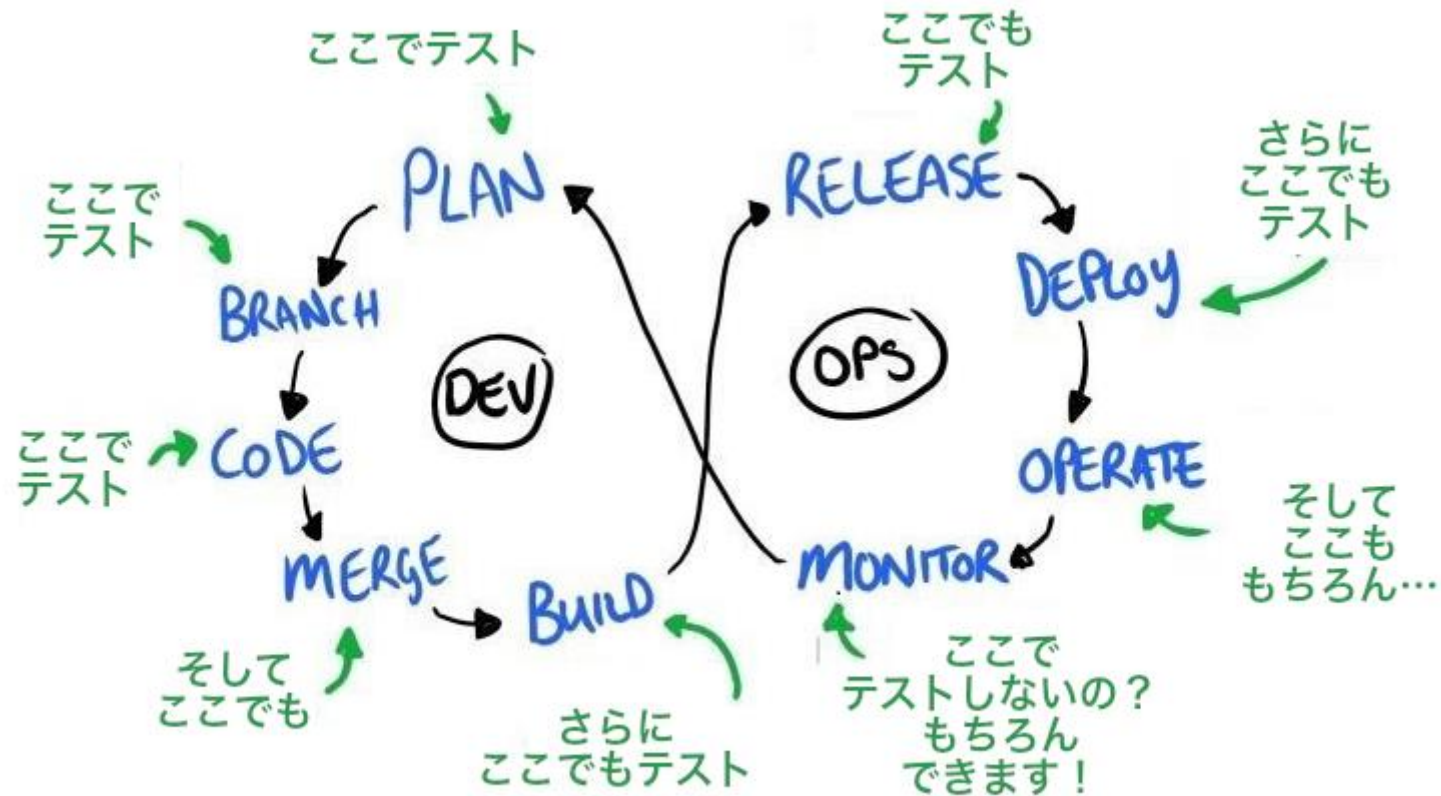


テスト自動化のピラミッド

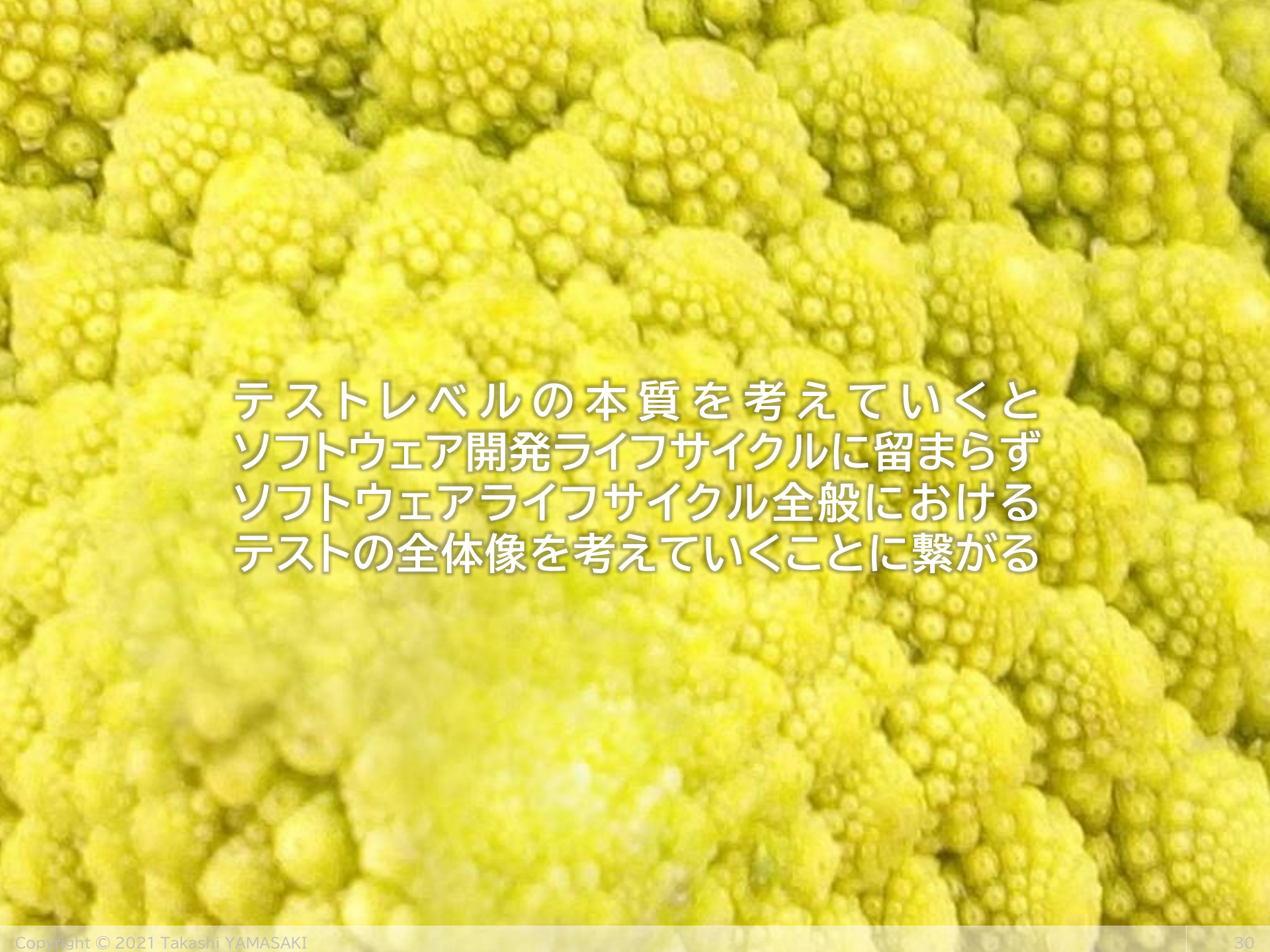


「実践アジャイルテスト P.274 図14-2 テスト自動化のピラミッド」に軸を付け加えた図(図の左側を独自に追加している。

DevOpsにおける継続的テスト (シフトレフトとシフトライト)



出典: 「Agile Testing Condensed Japanese Edition」 Janet Gregory, Lisa Crispin 著



テストレベルの本質を考えていくと
ソフトウェア開発ライフサイクルに留まらず
ソフトウェアライフサイクル全般における
テストの全体像を考えていくことに繋がる

分けるためにも概観できるテストの軸が重要

テストレベル

テストの空間のどこに対しての
要素であるかを意識した上で
テスト技術などの要素を考える

テストプロセス

テストタイプ

※図はイメージです

テストタイプとは？

“

テストタイプは、以下に列挙する特定のテストの目的から見たソフトウェアシステム(あるいはシステムの一部)の特性をテストするための活動を束ねたものである。

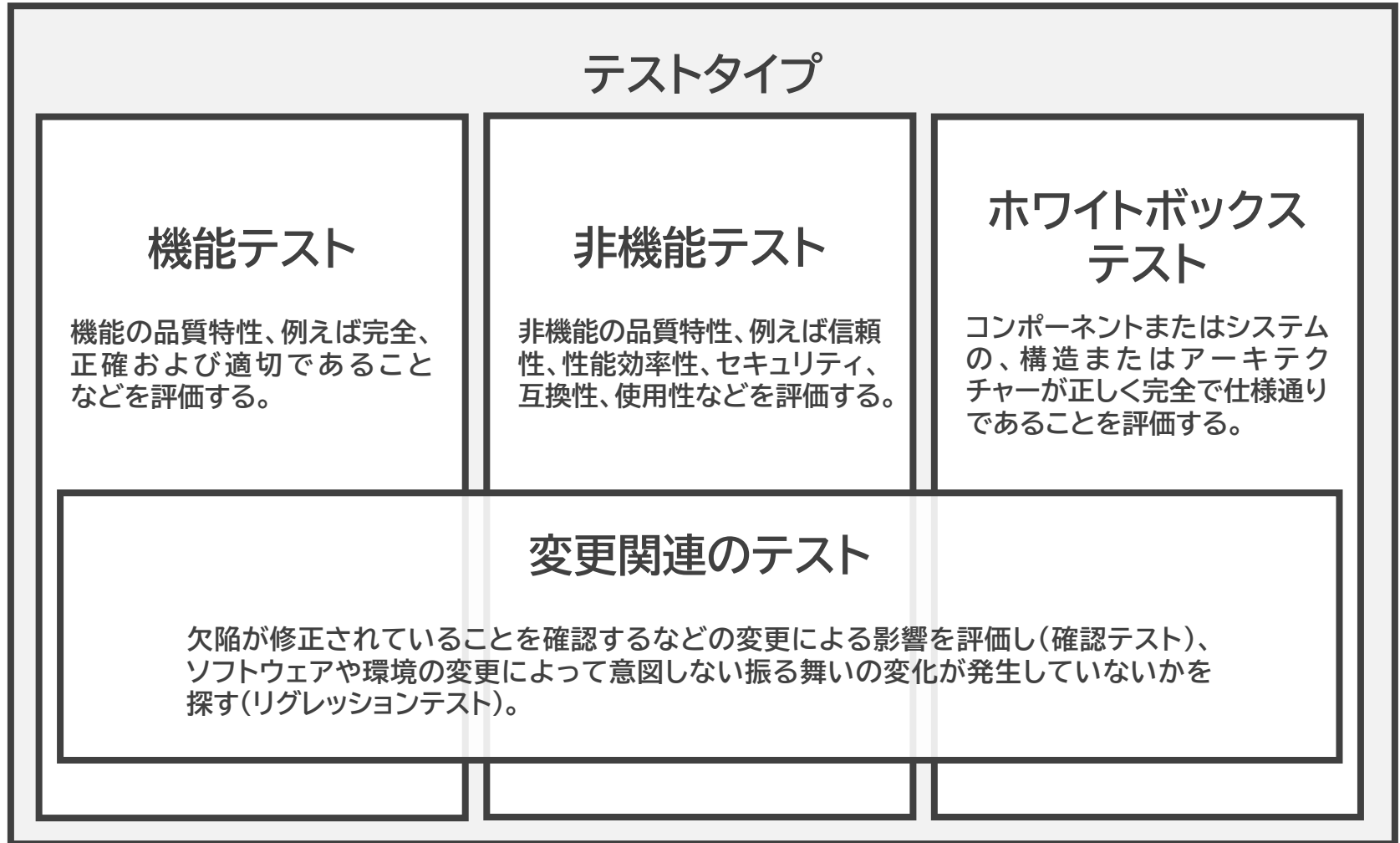
テスト技術者資格制度 Foundation Levelシラバス日本語版Version 2018V3.1.J02より

“

コンポーネントやシステムのある特性に対応したテストの目的を基にテスト活動をまとめたもの。

ISTQB Glossary V3.5 <https://glossary.istqb.org> より

JSTQBによるテストタイプの分類



ISO/IEC 25010の製品品質モデル



“製品品質モデル” from “JIS X 25010:2013” 図4

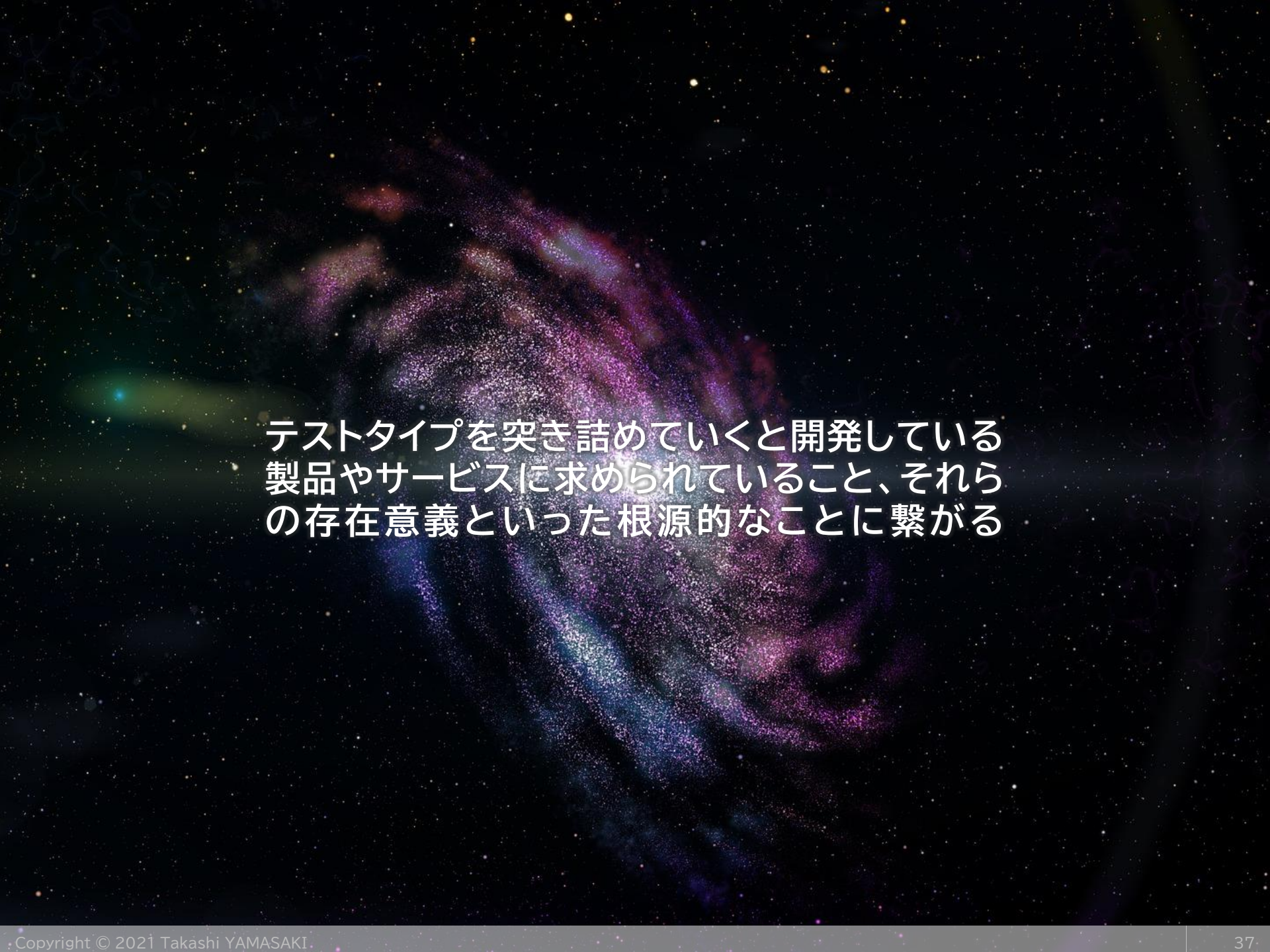
ソフトウェアテストでは機能以外の様々な側面について考慮することも重要です

品質モデルをテストの十分性や網羅性の文脈でつかうのは難しい

- 様々なテストを考慮するにあたり品質モデルをガイドとして利用するのはよいですが、テストの中に押し込むのはちょっと無理筋
- 前提として品質モデルは品質をMECEに定義していません
- また、各品質の特性は相互作用(相乗、相克、相反)があり、画一的なものではありません
- 各特性が「どうあるべきか」は、品質要求として定める必要があり、テストではその要求を満たしているのかを検証することになります
 - 例えば、時間効率性(パフォーマンス)として、何msの応答であれば妥当であるかは品質要求次第

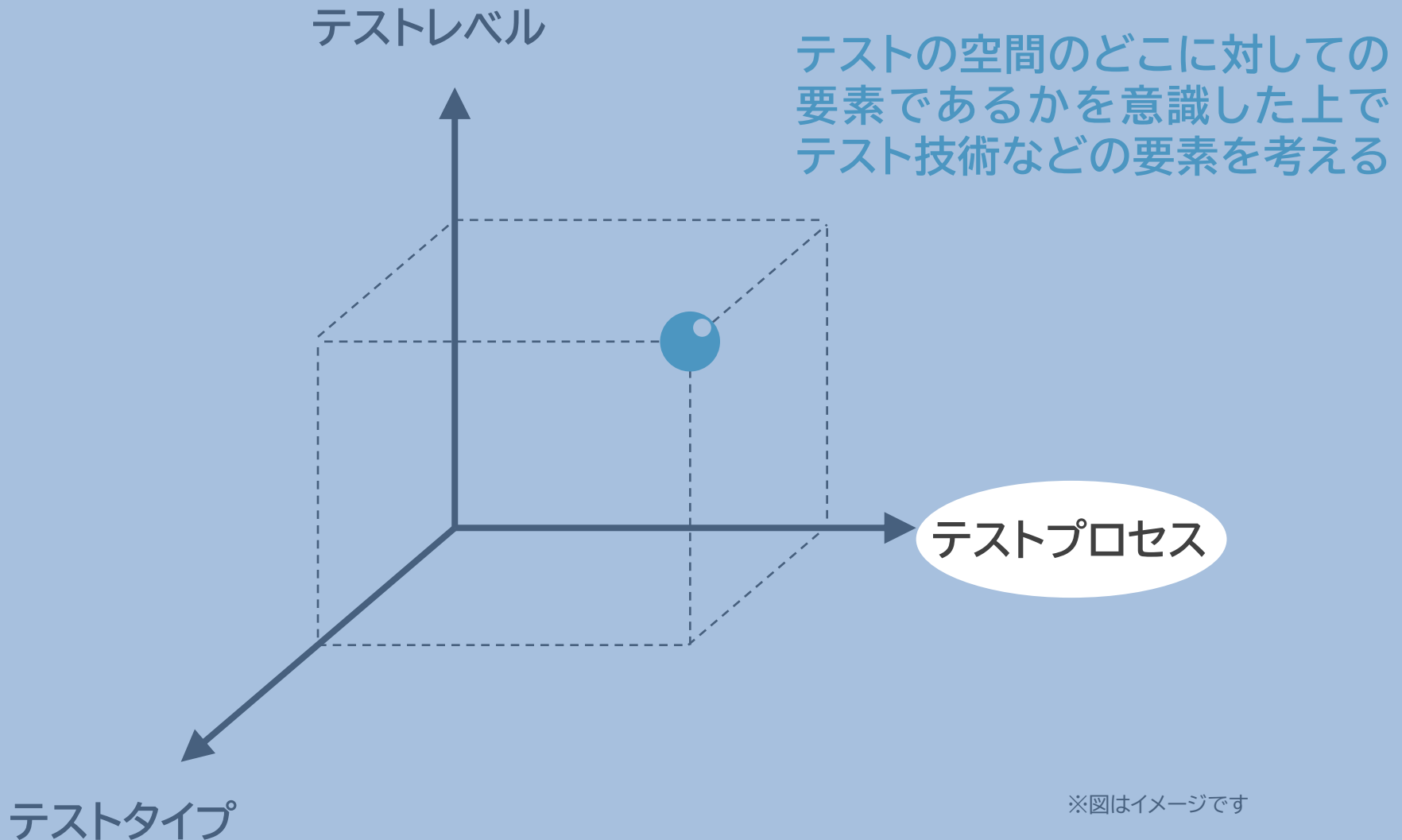
| | 可用性 | 効率性 | 柔軟性 | 完全性 | 相互接続性 | 保守性 | 移植性 | 信頼性 | 再利用性 | 堅牢性 | 試験性 | 使用性 |
|-------|-----|-----|-----|-----|-------|-----|-----|-----|------|-----|-----|-----|
| 可用性 | | | | | | | | + | | + | | |
| 効率性 | | | - | | - | - | - | - | | - | - | - |
| 柔軟性 | | - | | - | | + | + | + | | | + | |
| 完全性 | | - | + | | - | | | | - | | - | - |
| 相互接続性 | | - | + | - | | | + | | | | | |
| 保守性 | + | - | + | | | | | + | | | + | |
| 移植性 | | - | + | | + | - | | | + | | + | - |
| 信頼性 | + | - | + | | | + | | | | + | + | + |
| 再利用性 | | - | + | - | + | + | + | - | | | + | |
| 堅牢性 | + | - | | | | | | | | | | + |
| 試験性 | + | - | + | | | + | | + | | | | + |
| 使用性 | | - | | | | | | | | + | - | |

図：品質属性間のプラスとマイナスの相関関係 / Karl E. Wiegers 著「ソフトウェア要求 - 顧客が望むシステムとは」P-228より



テストタイプを突き詰めていくと開発している
製品やサービスに求められていること、それら
の存在意義といった根源的なことに繋がる

分けるためにも概観できるテストの軸が重要



※図はイメージです

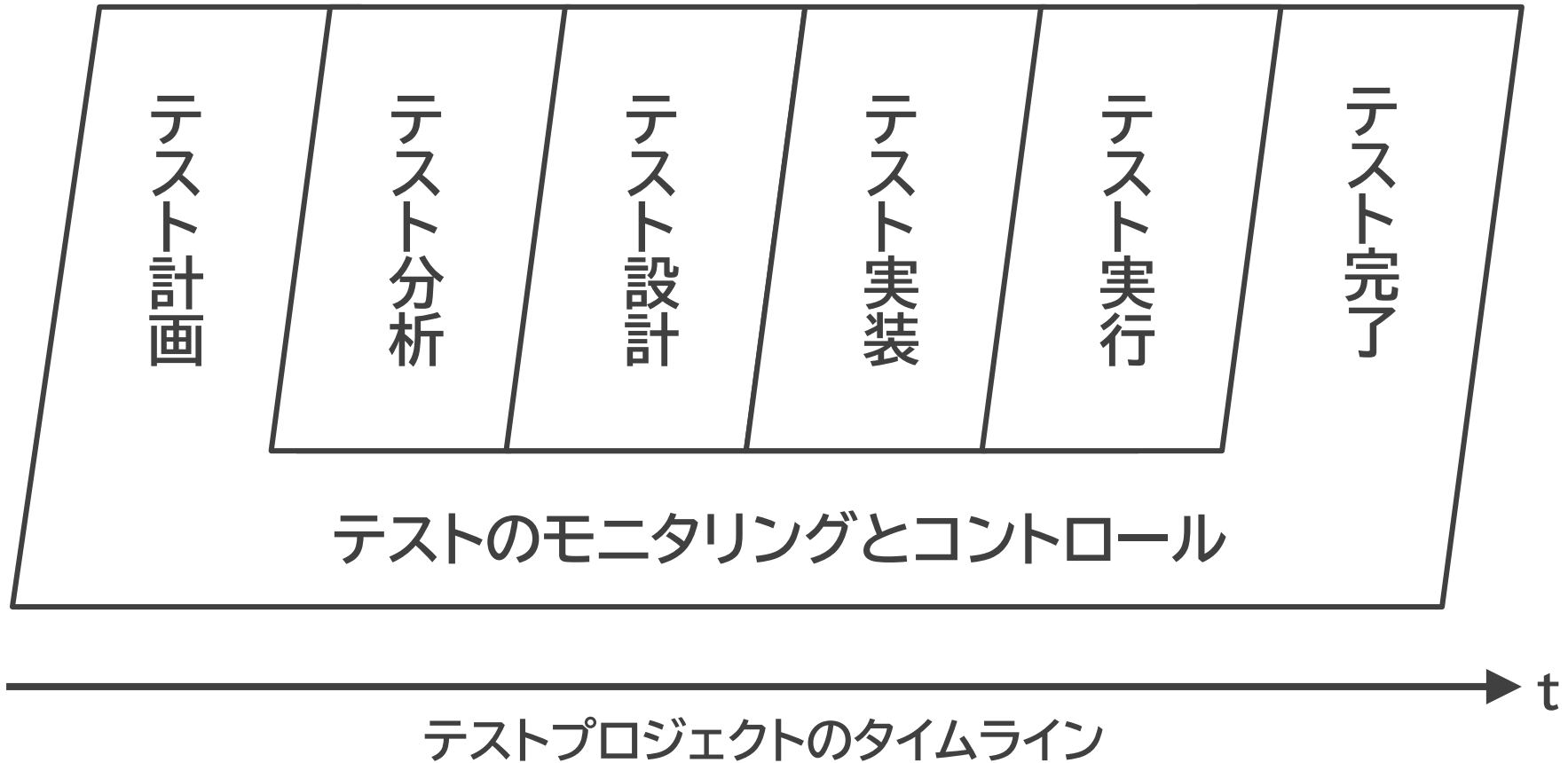
テストプロセスとは？

“

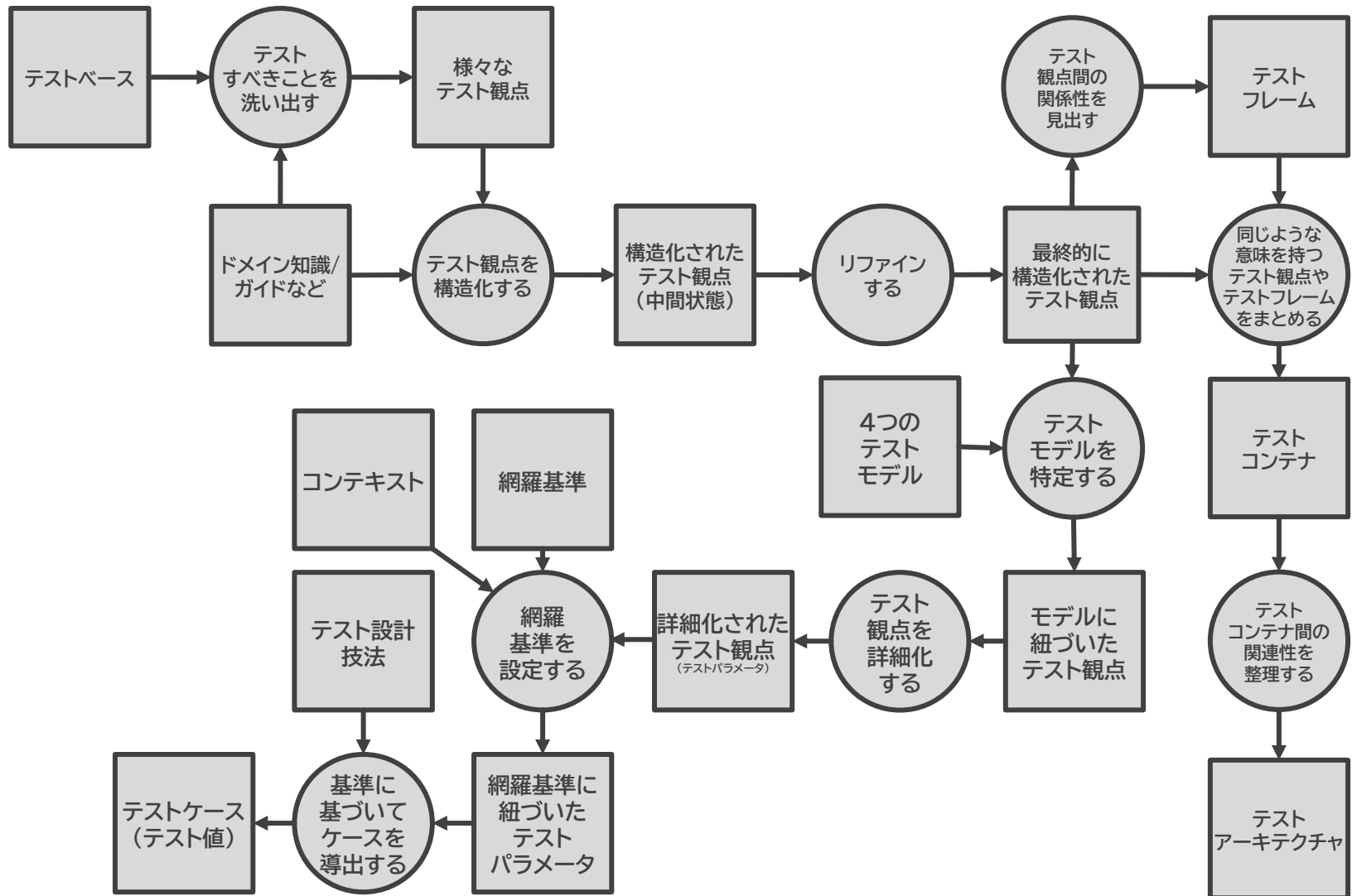
相互に関連する活動のセット。テスト計画作業、テストモニタリングとコントロール、テスト分析、テスト設計、テスト実装、テスト実行、テスト完了といった活動から構成される。

ISTQB Glossary V3.5 <https://glossary.istqb.org> より

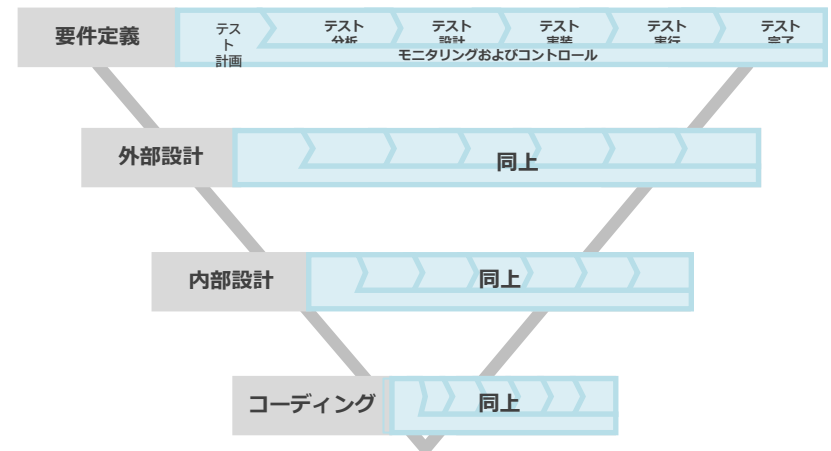
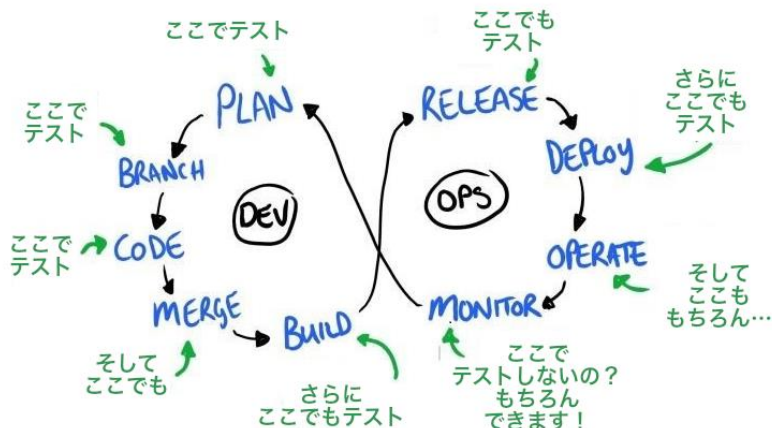
テストプロセスの例(JSTQB)



さまざまなテストレベルやテストタイプを実現するためのテストプロセスを設計しよう



ソフトウェアライフサイクルの中で 継続的に行っていく各テストの 特徴を見据えたプロセスを設計する



パイプラインも構築していく

継続的インテグレーション(CI)

継続的デリバリー(CD)

継続的デプロイメント(CD)

Commit

Build

Test

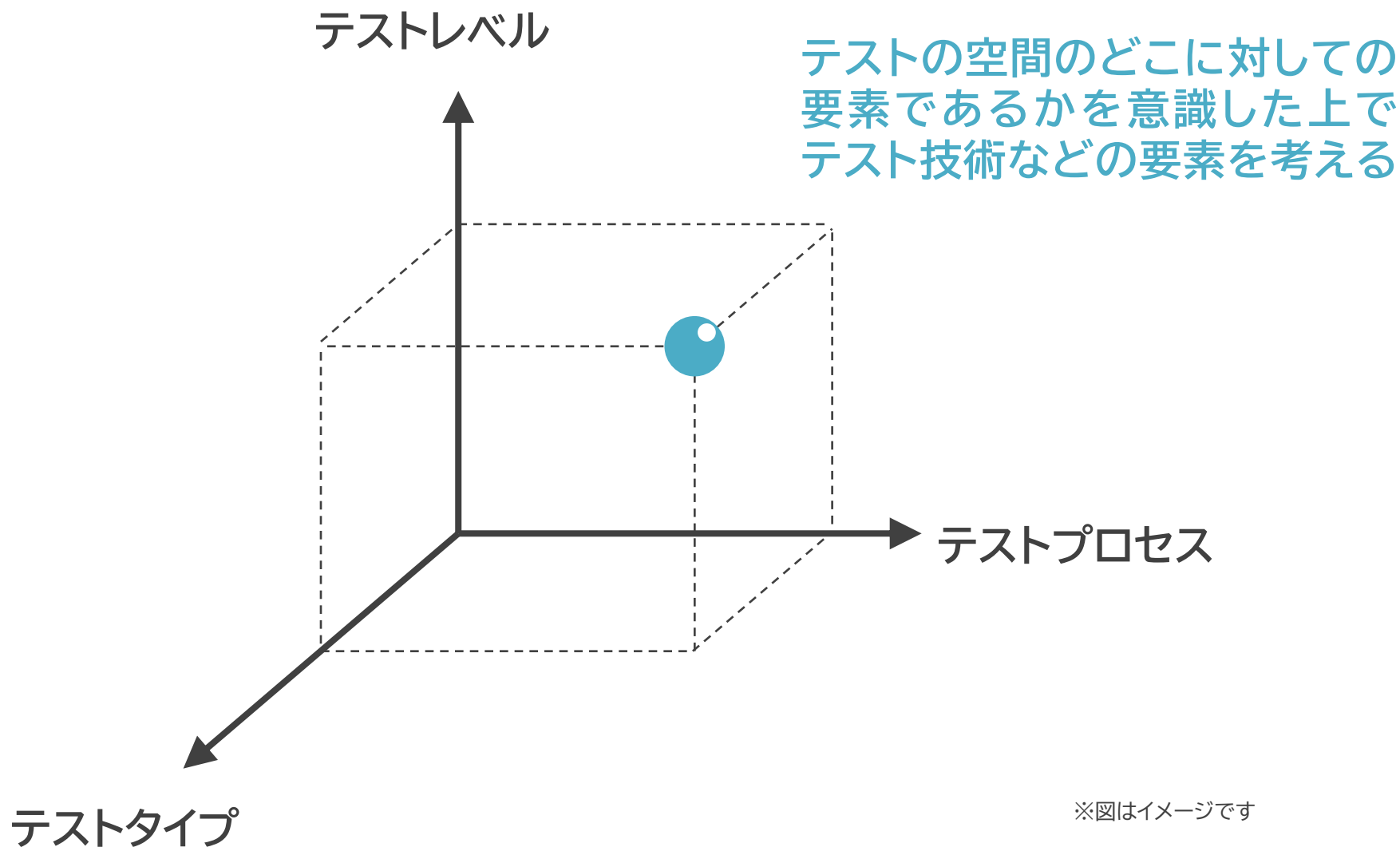
Stage

Deploy



テストプロセスを突き詰めて考えていくと
テストを実現するためのプロセスだけではなく
ソフトウェア開発ライフサイクルやソフトウェアラ
イフサイクルとの融合やパイプライン化に繋がる

分けるためにも概観できるテストの軸が重要



テストの空間の第1面(テストタイプ×テストレベル)

| | | | | | | | |
|------------|-------|-----------|----------|--------|-----|-------------|----------|
| 受け入れテスト | | | | | | | |
| システムテスト | | | | | | | |
| 統合テスト | | | | | | | |
| コンポーネントテスト | | | | | | | |
| | 機能テスト | セキュリティテスト | 相互運用性テスト | 使用性テスト | ... | ホワイトボックステスト | 変更関連のテスト |
| | | 非機能テスト | | | | | |

テストの空間の第2面(テストプロセス×テストレベル)

| 受け入れテスト | | | | | | | |
|------------|-------|-------|-------|-------|-------|-------|---------|
| システムテスト | | | | | | | |
| 統合テスト | | | | | | | |
| コンポーネントテスト | | | | | | | |
| | テスト計画 | 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 | テストのM&C |

テストの空間の第3面(テストプロセス×テストタイプ)

| 機能テスト | | | | | | | | |
|-------------|-----------|-------|-------|-------|-------|-------|-------|---------|
| 非機能テスト | セキュリティテスト | | | | | | | |
| | 相互運用性テスト | | | | | | | |
| | 使用性テスト | | | | | | | |
| | ⋮ | | | | | | | |
| ホワイトボックステスト | | | | | | | | |
| 変更関連のテスト | | | | | | | | |
| | | テスト計画 | 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 | テストのM&C |

テスト以外のスキルや知識が必要なだけでなく、
テストとテスト以外を如何にして繋げるかが重要

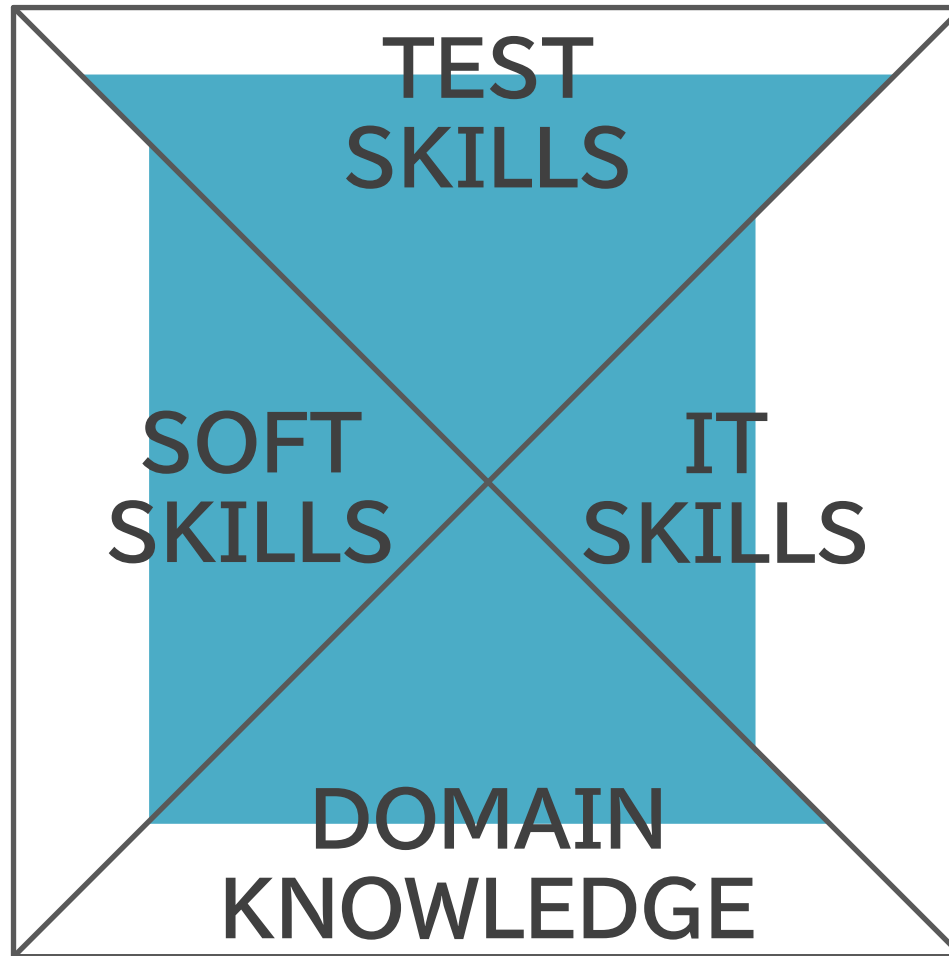
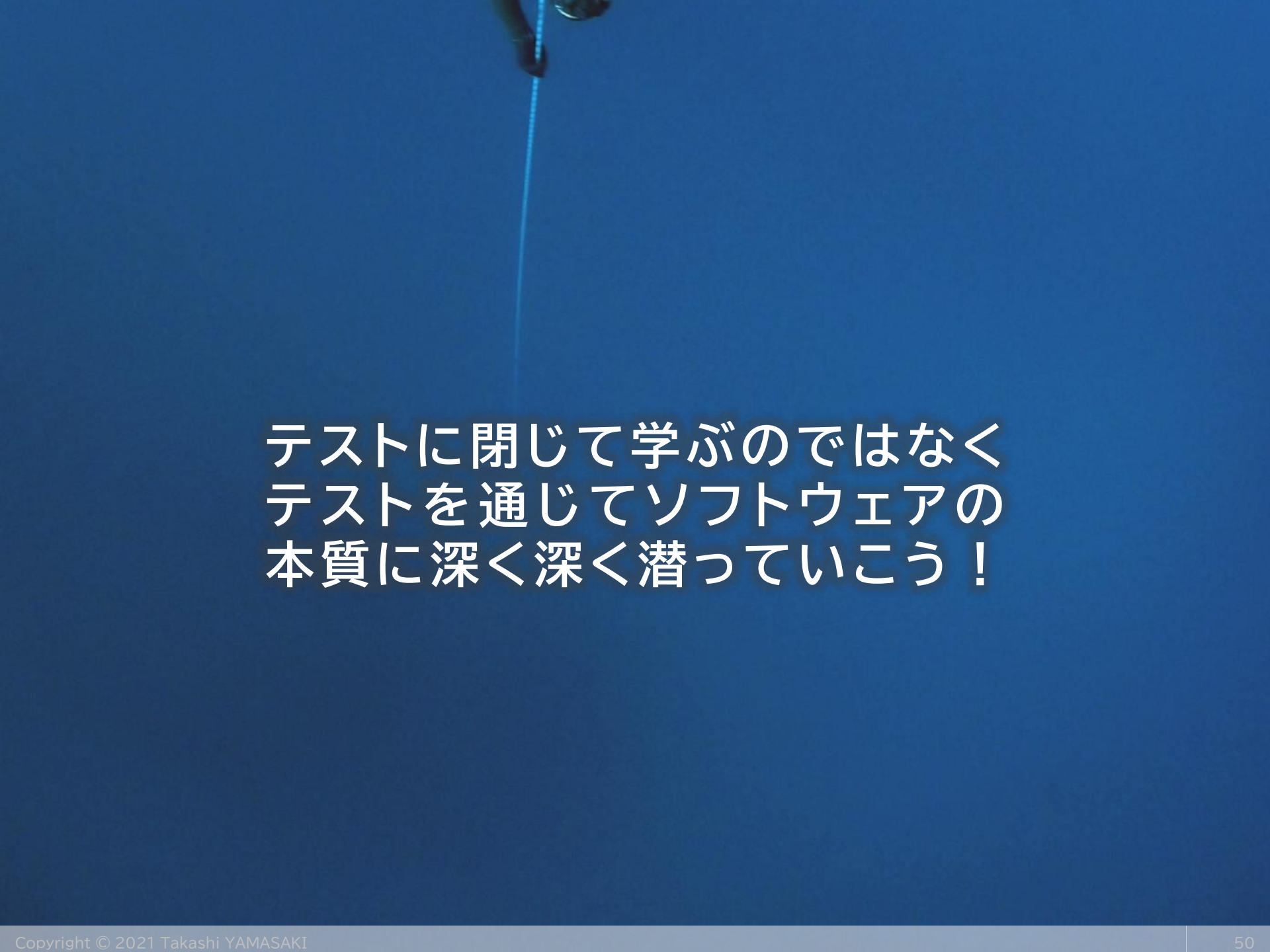


Fig. “Tester Skillspace” by Stuart Reid from JaSST Tokyo’ 14 Session Material (H8-2-2.pdf)



テストに閉じて学ぶのではなく
テストを通じてソフトウェアの
本質に深く深く潜っていこう！



学びや育成にも銀の弾丸はない

興味をもっていない人に働きかけても難しい





必要に迫られたり関心を持った際に、テストに近づいてきた時がチャンス！



そのためにも普段からテストの楽しさや
面白さなどを醸し出しておこう

社外に飛び出の機会を提供しよう！



さまざまなフィールドを通じて 知見を広めるとともに刺激を得る



テストを楽しんだり好きになれば
あとは勝手に転がり始めて成長してくれる

テストの全体観を持ち、
テストに強みをもった技術者になろう！



この講演のゴール

1. ソフトウェアテストの本質とはなんであるかを理解している
2. ソフトウェアテストについての全体像を俯瞰し、整理できる軸を識別している
3. テストに強みを持ったエンジニアになる・なってもらうための1事例を知っている

本講を切っ掛けにソフトウェアの深淵を
覗く旅への第1歩を踏み出してください

ご静聴ありがとうございました！

写真素材

pixabay

 FREEIMAGES

 使って楽しい、見て楽しい
PAKUTASO