説明できるテストをつくるために できることを考える

（公開用資料）

Kumiko Iseri

JaSST‘24 Tohoku　基調講演

自己紹介と注意事項

コミュニティ活動	JSTQB技術委員、 テスト設計コンテスト U-30クラス審査委員、等
過去の資料	https://speakerdeck.com/omn https://www.slideshare.net/omn

「日本におけるソフトウェアテスト技術者資格認定の運営組織」。**無料で参照できるシラバス**を公開。本講演でも何度か参照します。
https://jstqb.jp/index.html

テスト設計の出来を競うイベント。**来月6月OPEN／U-30のオンラインチュートリアル有、コンテストに応募せずとも参加可能。**学びたい方はぜひ！
https://aster.connpass.com/event/315775/
https://aster.connpass.com/event/315776/

この講演はSNSへの感想投稿は歓迎ですが資料のキャプチャの投稿はしないでいただければ。

本資料は、一部（ワーク周辺・事前質問への回答等を予定）を除き後日公開予定です。

https://speakerdeck.com/omn/jasst24tohoku-keynote

すみません、予稿集ではtohokuの後が_ですが-(ハイフン)が正しいです

「」内の引用元：https://jstqb.jp/committee.html

主な対象者

テストで仕様を分析する役割の方、テストケースを考える役割の方を想定していますが、職種や経験問わずテストの改善に興味のある方は歓迎です

- 開発プロセスは問いません
- 統合テストとシステムテストが主な対象ですが他のテストでも使える考え方を含みます

「テストをなんとなくで考えている」
「技法は使えるが、もっと工夫したい」方

見直しや改善のきっかけとして
いただければ幸い

一人で取り組める話も取り上げます

「既にテストの基本はできている」方

ご自身のやり方との比較、
ふりかえりをしてみて
いただければ幸い

説明できるテストがテーマ　なぜテストの説明が大事？

自分たちが納得できるプロダクトづくり

テストという切り口で支えるには？

テスト担当者だけでなく
ステークホルダーも納得できる
テストができるとより良い、
またテスト内容によっては
周辺チームの協力が必要なことも

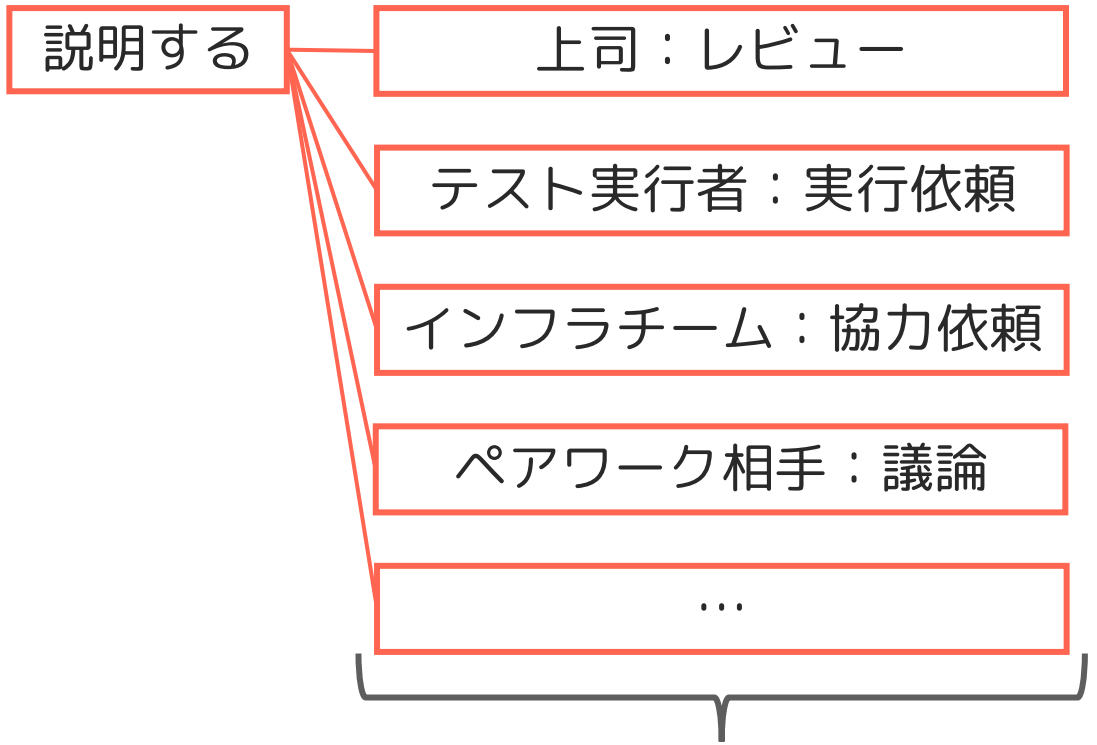
自分たちが納得できるテスト

ステークホルダーの協力

ここをうまく対応したい

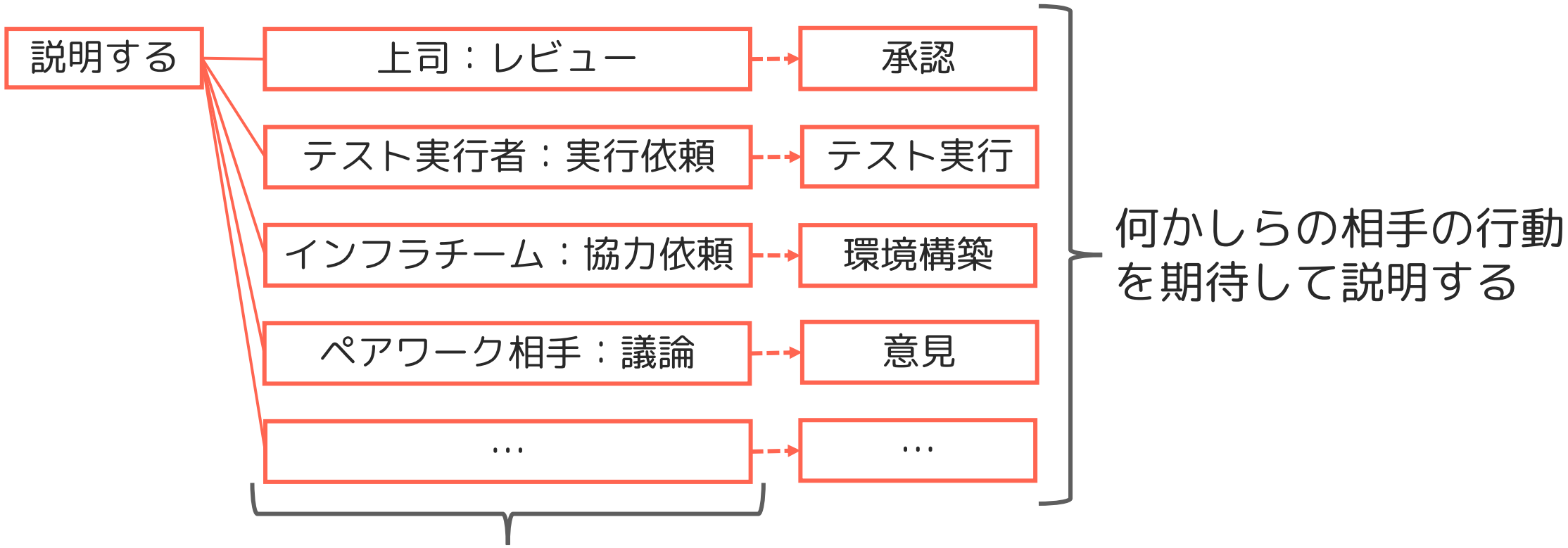
テストについて説明して理解してもらうこと

どんな場面で説明？



説明が必要な場面は意外と多い

なぜ説明？


説明する
上司：レビュー
承認
テスト実行者：実行依頼
テスト実行
インフラチーム：協力依頼
環境構築
ペアワーク相手：議論
意見
…
…
何かしらの相手の行動
を期待して説明する
説明が必要な場面は意外と多い

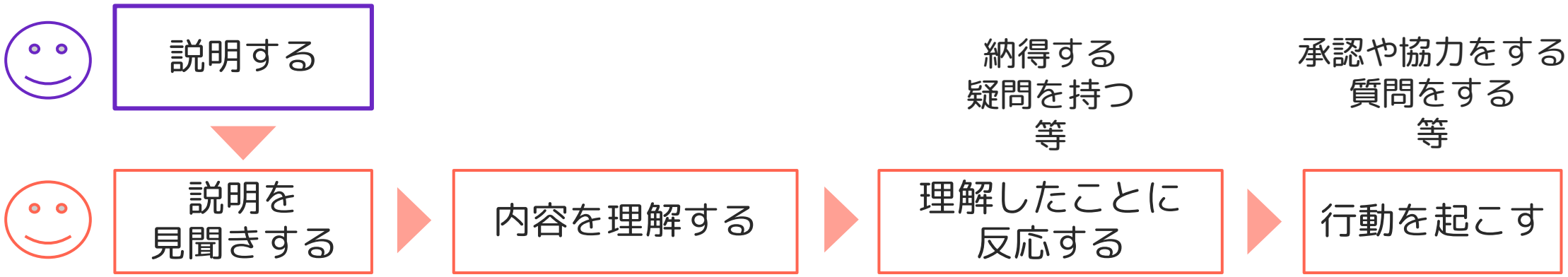
しかしテストケースや手順だけを見せても…

具体的なテストケースや手順を見せても、良い反応をもらえない

- 「で、結局網羅できているの？」
- 「こんなにテストが多いのはなぜ？」
- 「このテスト、どこから出てきたの？」

内容のほか、意図・拠り所の説明も必要になることが多そう

説明して相手に行動を起こしてもらうまでを分解してみる



どこで問題が起きる？

説明する

説明を
見聞きする

内容を理解する

理解しやすさが
今一歩だと？

納得する
疑問を持つ
等

理解したことに
反応する

内容自体の質が
今一歩だと？

承認や協力をする
質問をする
等

行動を起こす

できそうなこと？

内容自体の質が今一歩

思いつき（だけ）では、良くなさそう

→ **テストを多角的に、体系的に考える** ために

できることを知ると改善できそう

理解しやすさが今一歩

同じ主旨の説明をしても、伝え方が分かりやすさを左右する

→ **考えたテストについて分かりやすく伝える** コツを

知ると改善できそう

※一般的なコミュニケーションのコツはたくさんあります。

本講演では、その中から特にテストの説明に関係が深いと思われることを一部お話しします。

参考　他人に説明する機会がなくても説明を想定すると良いことがある

現状整理しやすくなる	意図や拠り所を説明しようと思うと、目の前のテストケースだけではなく、前提や途中で考えたことを思い出すことになる
将来の自分が理解しやすくなる(特に記録しておけば)	数カ月前に書いたテストケースがなぜ必要だったか思い出せないことがある… 説明するために意図や拠り所を明確にして記録しておけば、将来、思い出すときに役に立つ
改善点や改善方法が見えやすくなる	クマのぬいぐるみの話

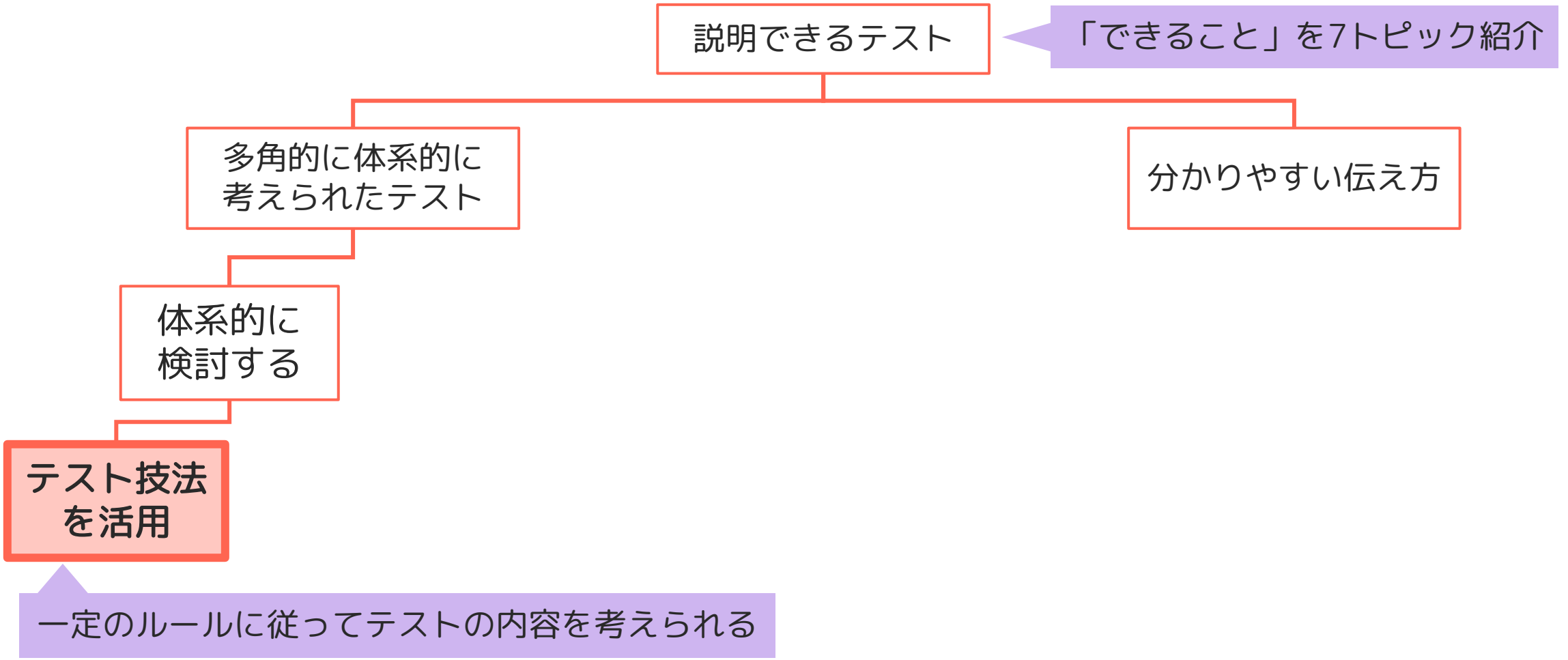
参考　クマのぬいぐるみでも

コードのバグ修正の話であるが、テストの改善にも同じことが言えそう

『**自分のコードを他人に説明してみよう。**自分のコードを誰かほかの人に説明して聞かせるのも効果的なテクニックだ。**こうすると自分自身にバグが見えてくることが多い。**（略）このテクニックは意外なほど有効だし、聞き手は別にプログラマでなくてもかまわない。**ある大学の計算機センターのヘルプデスクのそばにはテディベアのぬいぐるみが常備されており、魔訶不思議なバグに悩む学生は、人間のスタッフに相談する前にそのぬいぐるみに向かって説明しなければならないことになっていた。**』

引用元：Brian W. Kernighan, Rob Pike(著), 福崎俊博(翻訳).プログラミング作法. アスキードワンゴ. 2017年. 173ページ　（太字青字の装飾は本資料作成者による）

説明できるテストをつくるためにできること



テスト技法とは

テスト技法は、

「テスト条件の定義、テストケースの設計、およびテストデータの明確化をするための手続き。」

引用元：https://glossary.istqb.org/ja_JP/term/test-technique

以下の架空の仕様に対して、金額に応じて種類が正しく表示されることを確認したい場合、どんな金額でテストをすればよいだろうか？

金額が3万円以上ならば種類は “高額” 、それより低ければ種類は “通常” と表示する。
（3万円ちょうどなら “高額” ）

各自、頭の中で少し考えてみましょう（あてませんので、お気軽に）

テスト技法を活用する　例1

金額が3万円以上ならば種類は “高額” 、それより低ければ種類は “通常” と表示する。
（3万円ちょうどなら “高額” ）

境界値を狙って（**境界値分析**を適用して、）3万円より少し低い場合と3万円をテストする

テスト技法はあまり知らないよという方も、普段も同じような考え方、使ってませんか？

テスト技法を活用する　例2

条件同士の組合せによる結果の変化を確認するために、デシジョンテーブルテストを行う

[架空の備品貸出システムの仕様]

貸出備品の受け取り日は「●●以降にカウンターに受け取りにきてください」と画面表示される。条件により●●（いつから受け取りが可能か？）は変化する。緊急の場合は、対象の備品や貸出個数を問わず翌営業日。
緊急でない場合は、通常は2営業日後。ただし、

- 高額な備品を含む場合
  …準備にかかる日数が+1日。
- 合計6個以上の備品が対象の場合
  …準備にかかる日数が+1日。
- 高額な備品を含み、合計6個以上の場合
  …準備にかかる日数が+2日。

			ケース1	ケース2	ケース3	ケース4	ケース5
条件	緊急の貸出か	緊急	T	F	F	F	F
		通常	F	T	T	T	T
	貸出対象の金額	高額	N/A	T	T	F	F
		通常	N/A	F	F	T	T
	貸出対象の個数	5個以内	N/A	T	F	T	F
		6個以上	N/A	F	T	F	T
アクション（●●部分の表示）	翌営業日		X				
	2営業日					X	
	3営業日			X			X
	4営業日				X		

技

テスト技法を活用する　テスト技法？

JSTQBのシラバス(※)で紹介されている技法は…

同値分割法
境界値分析
デシジョンテーブルテスト
状態遷移テスト
クラシフィケーションツリー技法
ペアワイズテスト
ユースケーステスト
ステートメントテスト
ブランチテスト
など

様々なテスト技法があるうち、本講演では
ブラックボックステスト技法とホワイトボックステスト技法を想定してお話しします

※参考：https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf 4 テスト分析と設計、
https://jstqb.jp/dl/jstqb.jpdlJSTQB-SyllabusALTA_V311.J03.pdf　3.テスト技法

テスト技法を使うメリット

抜け漏れや重複に気づきやすくなる

一定のルールに従って考えることになり、ランダムに考えるよりも気づきやすくなる

共通言語として使える

「デシジョンテーブルテストを使っている」と言えば、最近チームに入った人にもイメージが容易に伝わる

カバレッジを説明しやすくなる

適用するテスト技法によっては、網羅対象（例：デシジョンテーブルテストの条件の組合せ）や、どの程度網羅するか、が伝わりやすくなる

テスト技法さえ使えば良いテストケースが書ける…？

適用するテスト技法を
決めるのは人間

テスト技法を使わない場合も同じ

何をテストの対象にするか
決めるのは人間

不適切なテスト技法では期待した効果が出ない

「技法は知ってしまうと使いたくなる」

要注意

- 知っているから使う、ではなくテスト技法の特徴や制約を学び、状況にあっているかを考え適切に選択する

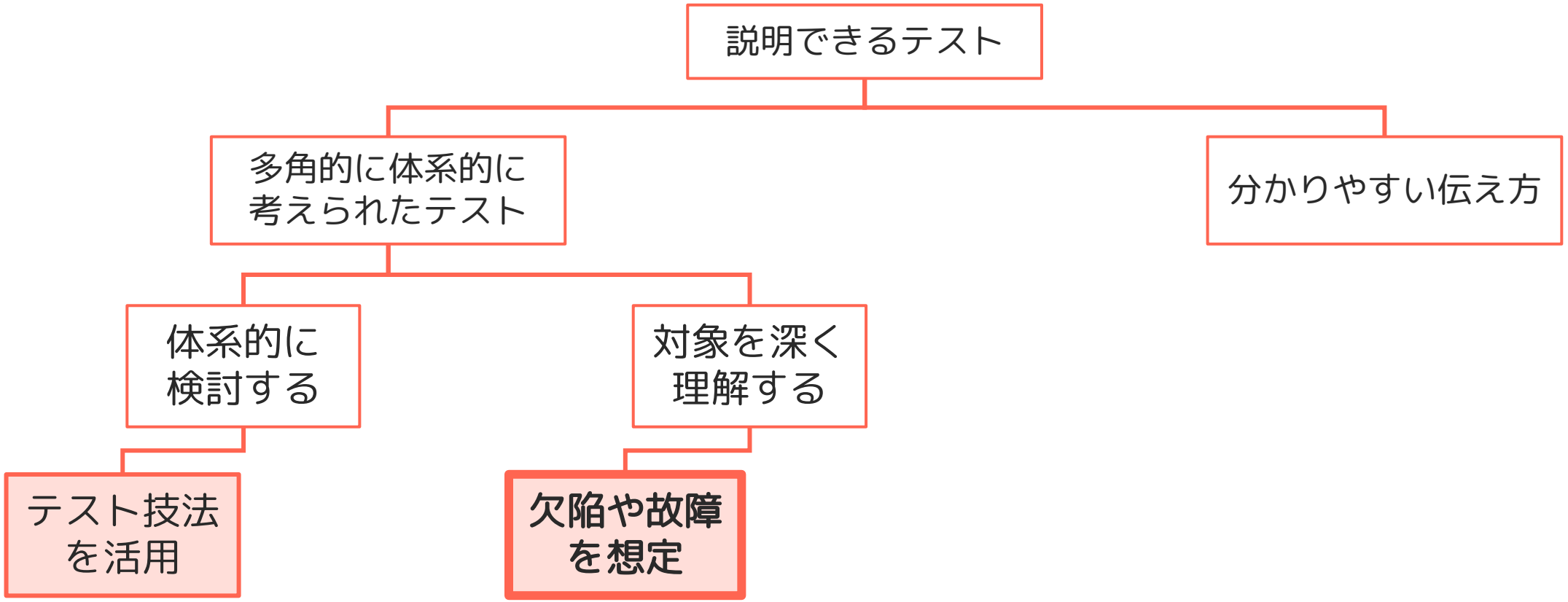
例えば、先ほどの備品貸出システムの例で緊急度・金額・個数を組合せることに気づかなかったら？

- 欠陥を見逃すかもしれない
- テストの内容は見つけたい欠陥・故障に左右される

テスト技法を活用するだけでなく起こりそうな欠陥・故障を考える必要がありそう

緑枠内の「」内の引用元：JaSST‘19 Hokkaido「テスト設計技法、その前に～フェイスアップ、次にビルドアップ、その先にマインドアップ～」 https://www.jasst.jp/symposium/jasst19hokkaido/pdf/S1.pdf 38ページ

起こりそうな欠陥や故障を検討



参考　欠陥？故障？

「**人間はエラー（誤り）を起こす。そのエラーが、欠陥（フォールト、バグ）を生み出し、その欠陥は、 故障につながることもある。**人間は、時間的なプレッシャー、作業成果物、プロセス、インフラ、相互作用の複雑度、あるいは単に疲れていたり、十分なトレーニングを受けていなかったりなど、さまざまな理由でエラーを起こす。
欠陥は、要件仕様書やテストスクリプトのようなドキュメント、ソースコード、またはビルドファイルのような補助的なアーティファクトの中から発見できる。（略）コードの欠陥が実行されると、システムがすべきことをしない、またはすべきでないことをし てしまうことがあり、故障の原因となる。欠陥の中には、実行すれば必ず故障になるものもあれば、特定の状況下でしか故障にならないもの、絶対に故障にならないものもある。
故障の原因は、エラーや欠陥だけではない。例えば、放射線や電磁波によってファームウェアに欠陥が生じる場合など、環境条件によっても故障が発生することがある。 」

引用元：https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf
1.2.3 エラー、欠陥、故障、および根本原因（太字青字の装飾は本資料作成者による）

本講演を聞くうえでは区別しなくても支障はありませんが、使い分ける考え方もあることを知っておくとよいでしょう

故障や欠陥の特徴？

「テストの原則」のひとつに「欠陥の偏在」がある

> 「通常、見つかる欠陥や運用時の故障の大部分は、少数のシステムコンポーネントに集中する」
>
> 引用元：テスト技術者資格制度 Foundation Level シラバス Version 2023V4.0.J01
> https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf 1.3 テストの原則 19ページ
> （枠外「」内も同様）

偏在していると想定すると、すべてをまんべんなくテストするのではなく、
欠陥やその結果として起こる故障が潜んでいそうなところを考えることに意味がある

どうするか？

起こりうる欠陥や故障を想定する

まずは、出し切ることに集中する

次ページからもう少し考えます

想定した欠陥や故障に優先度をつける

考えた欠陥や故障それぞれに対して、
想定する可能性や頻度、影響度をもとに
優先度をつけ、優先度の高いものに確実に
リソースを割けるようにする

優先度はビジネスの状況など様々なことに
左右されるためステークホルダー間で合意
することがベスト

どうやって想定するか？

ステークホルダーとも認識を合わせられると先々のレビューや欠陥の修正判断にも役立つ

気になるところを聞く

例えば、ソフトウェア設計者に不安なところを聞き出す

「3H（変化、はじめて、久しぶり）の確認」

取捨選択の理由の
説明も必要な点に注意

一人でも手掛かりをもとに考えることはできる

提供したい価値、シナリオ	組織体制	一般的なキーワード 例 ISO/IEC25010
価値提供を阻害する故障、等	複数組織が関係する部分、等	通常、取捨選択が必要
過去・類似ソフトウェアの 欠陥や故障	組織のチェックリスト	成果物の状態 例 複雑度、修正頻度

技
欠

緑枠内の「」内の引用元：JaSST'21 Kyushu 「１番役に立った定番の ソフトウェアテスト」
https://www.jasst.jp/symposium/jasst21kyushu/pdf/S3.pdf 13ページ

緑枠内の「」内の参考：「27号：欠陥の偏在」 https://note.com/akiyama924/n/n79b902f69ddf

考えるときのお勧め

抽象と具体を意識的に行き来する

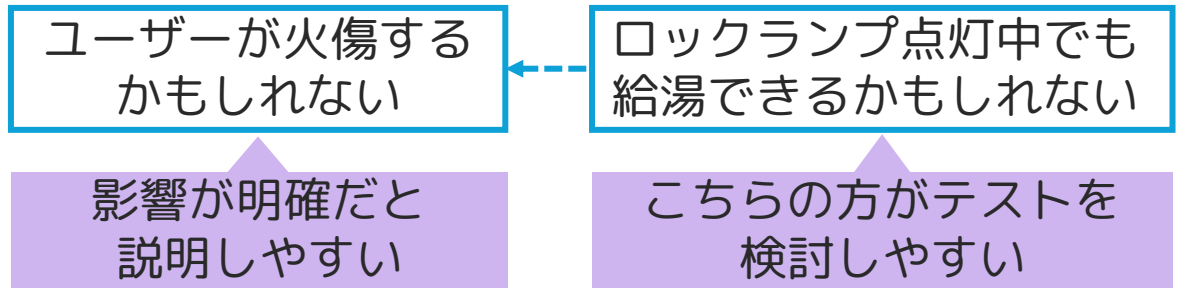
- 考えついた欠陥や故障を共通点をもとに抽象化し、抽象化した結果から更に他の具体的なことを考える

自分たちでは検出や解決が難しくても一度は検討する

- 専門家のレビューを追加して検出可能性を上げたり、マニュアルの充実化で影響を小さくできるかもしれない
- ソフトウェア要件定義や設計の段階で気づいたならば、機能の仕様やつくりを見直す方法が最善なことも

ステークホルダーやビジネスへの影響も考える

- 拠り所の説明がしやすくなる、優先度の議論に役立つポットの例



うまく欠陥や故障が考えられないこともある

- アイディアが出てこない…
- 過去の欠陥を見ているが、どれを取り上げるべきかよく分からない…

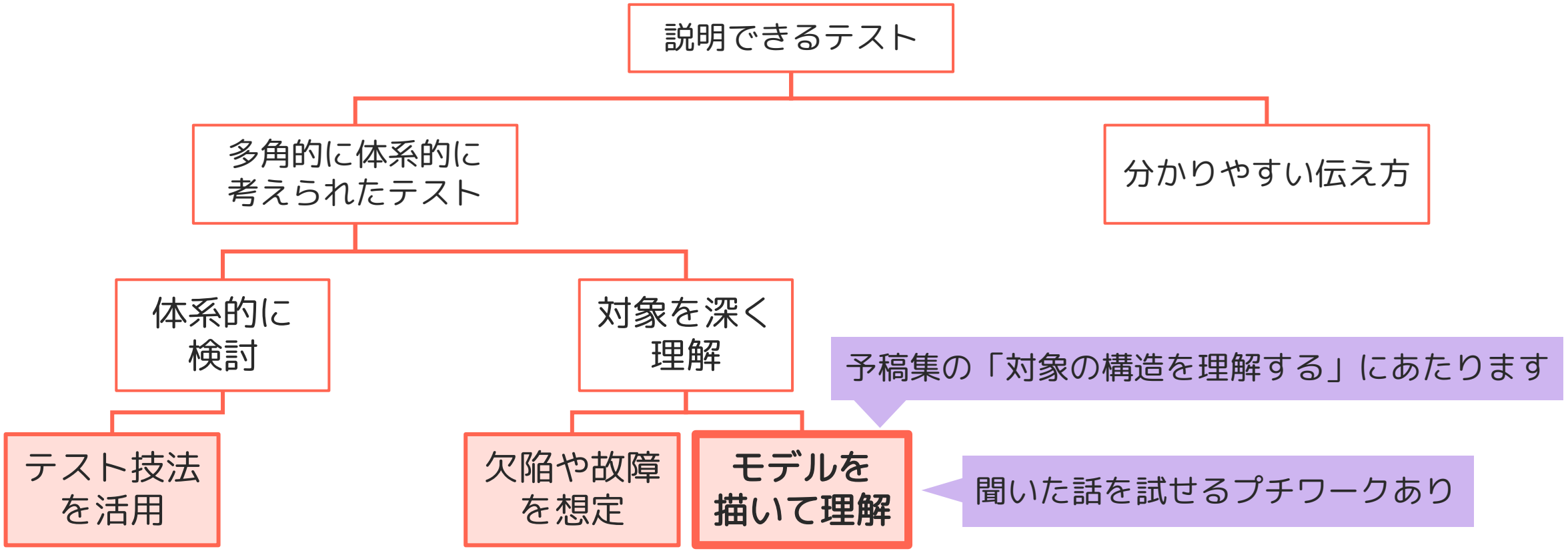
→ひょっとすると、テスト対象の理解が不十分なのかもしれない

しかし、仕様書、設計書、…を読んだだけでは、なかなか理解しきれない
通常、仕様書等は目の前のテストに最適化された内容や構成になっていないことが一因

- まとめて考えたい情報同士が離れたところに記されていたり、書かれていなかったりする
- 不要な情報が混ざっている

情報収集して終わり、ではなく、得た情報を整理する必要がありそう

モデルを描いて理解する



モデルを描いて理解する

得た情報を整理する方法のひとつとして、本講演ではモデルを描くことをお勧めしたい

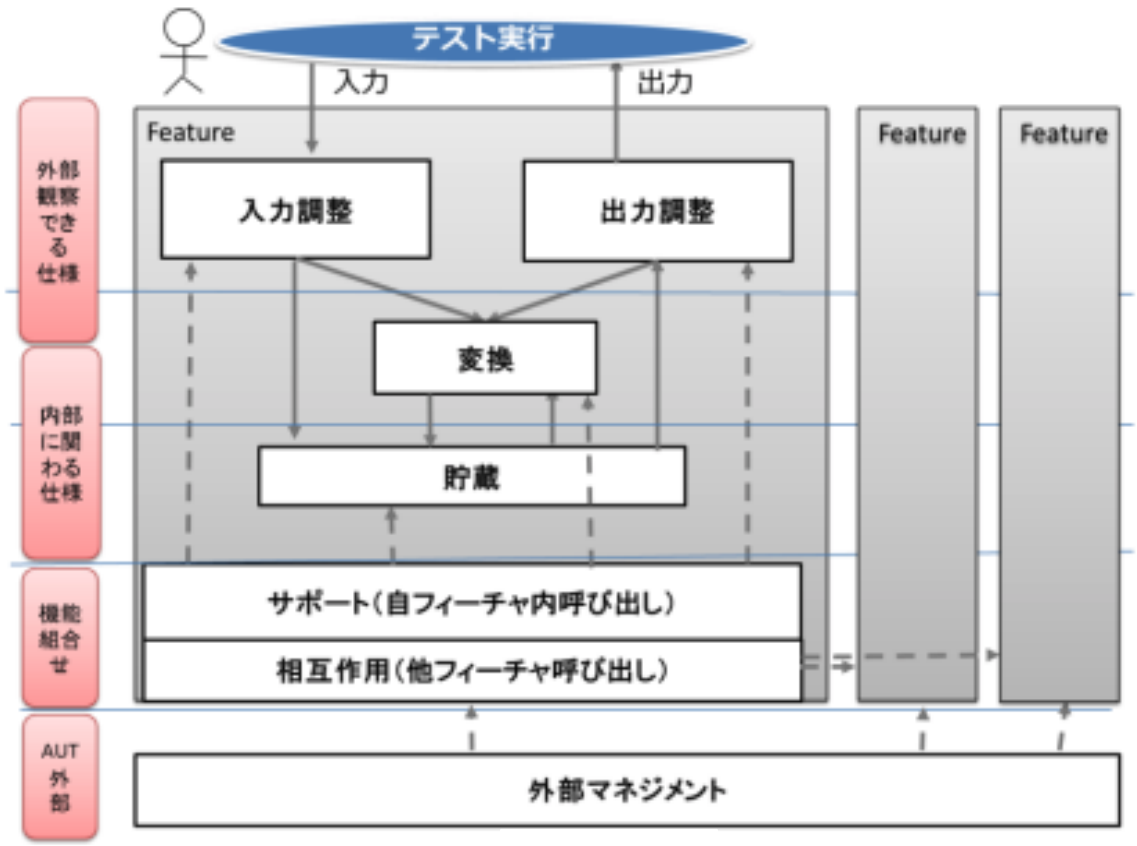
次の3点を同時に実現することを、本講演では「モデルを描く」と表現する

全体像を見せる	構成要素の要点を適切な抽象度で見せる	構成要素の関係性を見せる
着目している範囲、規模、複雑さが分かりやすくなる	余分な情報が減り着眼点に沿って理解や検討がしやすくなる	要素が互いにどう影響し合うか分かりやすくなる

テスト対象のIPO(入力、処理、出力)を簡単な図に描いただけでも、モデルを描いた例といえる

どんなモデルがある？

- IPO(入力、処理、出力)
- テストで使われる考え方を参考に
  - テスト対象を入力調整、貯蔵といった要素で分解した、ゆもつよメソッドの論理的機能構造
  - 『大村朔平さんの書籍「一般システムの現象学」に書かれていたものを、湯本さんがテスト分析のモデルとして改変したものである。』
- ソフトウェア要件定義、設計で使われる考え方を参考に
  - ユースケース図、シーケンス図、など



『』内の引用元：JaSST'21 Tohoku　ゆもつよメソッドワークショップ「仕様書読みとモデル図作成」
https://jasst.jp/symposium/jasst21tohoku/pdf/S3-1-3.pdf　8ページ
図の引用元：JaSST'15 Tokyo「解決！テストアーキテクチャ設計」講演資料：「湯本からのコメント」
https://jasst.jp/symposium/jasst15tokyo/pdf/B2-3.pdf　4ページ

モデルを描く手順

次ページ以降で例を使って、モデルを新規に描く流れをもう少し詳しく見ていく

1. 着眼点を決める

2. インプット情報を収集し必要な要素を抜き出す

3. 要素の要点を抽出したり抽象度を調整したりする

4. 要素同士の関係性が分かるようまとめる

例題：架空の備品管理システムの購入機能の仕様

次の仕様に対して、どのような着眼点でモデルを描くか？

申請者が申請作成を開始すると、作成中という状態になる。申請者は作成を終えたら、申請ボタンを押す。申請ボタンの代わりに一時保存ボタンを押すかEscキーを押すと一時保存処理が実行されるが、状態は変わらない。

購入額が税抜3万円以上の備品は高額備品である。対象が高額備品の場合は所属部部長承認待ちとなる。対象が通常備品の場合は所属部の課長承認待ちとなる。所属部の課長または所属部の部長が所属部承認ボタンを押すと、経理部課長であるAさんまたはBさんによる承認待ちとなる。

AさんまたはBさんが経理部承認ボタンを押すと申請が完了状態となる。

所属部の課長または所属部の部長が申請に不備を見つけた場合、差し戻しボタンを押す。この場合、1つ前の状態に戻る。

モデルを描く手順　前半

1. 着眼点を決める

着眼点は、仕様を確認する中で気になることや、持っている知識をもとに決める

特になれないうちは、1まとまりのモデル（最終的には主に図表）の着眼点は少数に絞る

入力値の範囲だけ、画面遷移だけ、システム構成だけ、など

2. インプット情報を収集し必要な要素を抜き出す

ここでのインプット情報は文書、チャット、ヒアリング結果、記憶等、様々なものを指す

例題　架空の備品管理システムの購入機能の仕様

1段落目に状態が複数登場していることから、状態とその遷移を着眼点とした

必要な要素としては、状態／状態遷移を引き起こすイベント／状態遷移するうえでの条件のピックアップが必要と考え、それぞれを黄色マーカー／下線／囲みで印をつけた

申請者が申請作成を開始すると、作成中という状態になる。申請者は作成を終えたら、申請ボタンを押す。申請ボタンの代わりに一時保存ボタンを押すかEscキーを押すと一時保存処理が実行されるが、状態は変わらない。

購入額が税抜3万円以上の備品は高額備品である。対象が高額備品の場合は所属部部長承認待ちとなる。対象が通常備品の場合は所属部の課長承認待ちとなる。所属部の課長または所属部の部長が所属部承認ボタンを押すと、経理部課長であるAさんまたはBさんによる承認待ちとなる。

AさんまたはBさんが経理部承認ボタンを押すと申請が完了状態となる。

所属部の課長または所属部の部長が申請に不備を見つけた場合、差し戻しボタンを押す。この場合、1つ前の状態に戻る。

技
欠
モ

モデルを描く手順　後半

3. 要素の要点を抽出したり抽象度を調整したりする

インプット情報はそのまま使えるとは限らない
不要な情報は除外／不足情報は補う、共通する性質でまとめて抽象化／分けて具体化

最初に決めた着眼点に基づいて考える

4. 要素同士の関係性が分かるようまとめる

包含や順序等の関係性を分かりやすくするため、基本的には図表で可視化する
着眼点が見た人に伝わるようにまとめることが大事

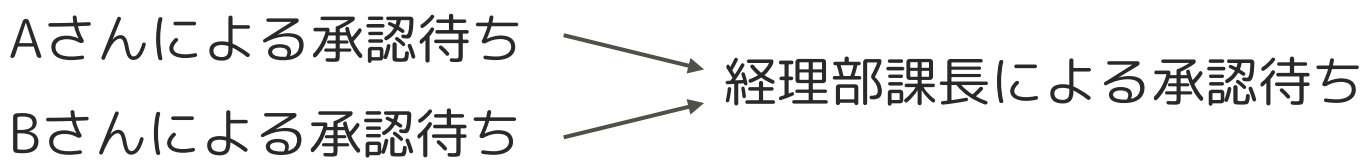
例題　架空の備品管理システムの購入機能の仕様

着眼点を意識する

今回の例では、伝えたいのは状態とその遷移

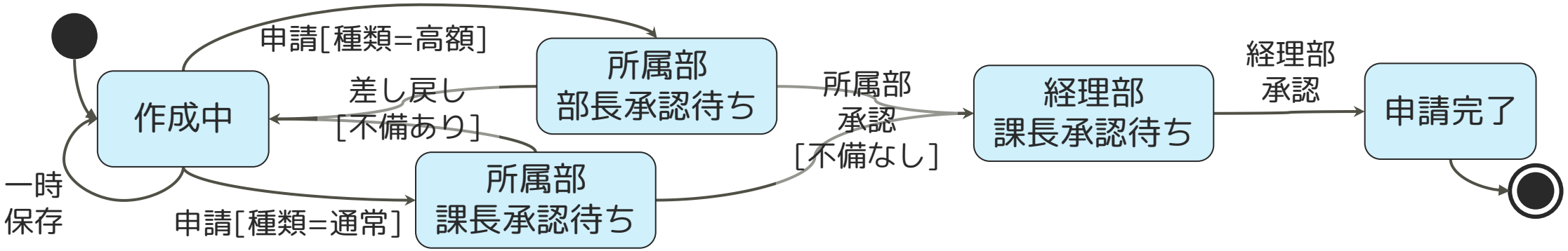
具体的な個人名の情報などは不要なので、除外する

「経理部課長であるAさんまたはBさんによる承認待ち」



例題　架空の備品管理システムの購入機能の仕様

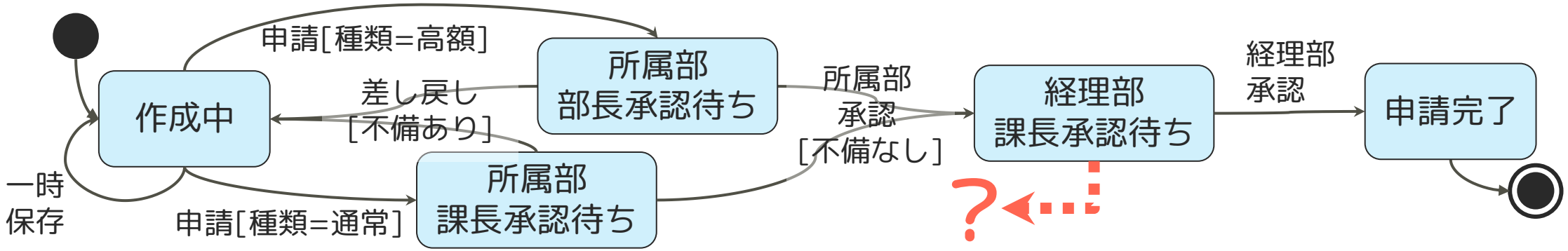
状態遷移図、状態遷移表で表現すると、複雑さや、状態とイベントの関係が俯瞰しやすくなった



イベント＼状態	作成中	所属部課長承認待ち	所属部部長承認待ち	経理部課長承認待ち	申請完了
申請[種類=通常]	所属部課長承認待ち	N/A	N/A	N/A	N/A
申請[種類=高額]	所属部部長承認待ち	N/A	N/A	N/A	N/A
一時保存	作成中	N/A	N/A	N/A	N/A
所属部承認	N/A	経理部課長承認待ち	経理部課長承認待ち	N/A	N/A
経理部承認	N/A	N/A	N/A	申請完了	N/A
差し戻し	N/A	作成中	作成中	N/A	N/A

例題　架空の備品管理システムの購入機能の仕様

モデルを描いただけでも理解は進むが、できた図表を見ながら更に考えてみると？



		所属部課長承認待ち	所属部部長承認待ち	経理部課長承認待ち	申請完了
		N/A			N/A
					N/A
		N/A			N/A
		経理部[illegible]		N/A	N/A
		N/A	N/A	申請完了	N/A
差し戻し	N/A	作成中	作成中	N/A	N/A

不足や重複や不揃いがないか、割込みや分岐などがありえないか、等を確認すると、仕様が曖昧な部分や発生しそうな欠陥を見つけやすくなることも

他の承認待ちと違い、
経理部課長承認待ちからの
差し戻しはない？

技
欠
モ

モデルを描く手順

1. 着眼点を決める

2. インプット情報を収集し必要な要素を抜き出す

3. 要素の要点を抽出したり抽象度を調整したりする

4. 要素同士の関係性が分かるようまとめる

- まとめる際に要素の不足に気づいて情報収集するなど、行きつ戻りつすることも多い
- 既にあるモデルを利用する場合でも、着眼点や要素に過不足ないか確認は必要
- この例ではテスト対象の内部をテーマにしたが、ユーザーとシステムとのやりとりといったものについても、モデルを描き整理することは役に立つ

モデルを描く・欠陥や故障を考えるお試しワーク

題材として、架空の社内備品管理システムの貸出申請機能の仕様の印刷物を配布します。
唯一の正解を求めるワークではなく、聞いた話を試すためのワークです。

休憩したい方は
この間にどうぞ

約15分間　個人ワーク：モデル作成、曖昧な部分と想定される欠陥・故障の検討を試す

- 次の2つの作業を、できるところまでして、結果を紙またはPCにメモしてください
  - テスト対象について**モデル（図表）を描き仕様を整理**してください。複数描いてもOKです
  - 余裕があれば、**仕様が曖昧な部分や起こりそうな欠陥・故障**も考えてください
- **グループワークにて考えたことを1分程で共有していただく心の準備を**お願いします

約5分間　グループワーク（テーブルごと）：他の人の着眼点や考えたことを知る

- 前側かつドア側の席の方から時計周りで、1人につき約1分で以下を共有してください
  - **お名前 (ニックネームOK)**
  - **個人ワークで考えたこと (どんなモデルを描いた・描こうとしたか、曖昧な部分、欠陥・故障)**
- 1分ごとに合図しますので次の方に交代してください

次はグループワーク開始ページ

モデルを描く・欠陥や故障を考えるお試しワーク

題材として、架空の社内備品管理システムの貸出申請機能の仕様の印刷物を配布します。
唯一の正解を求めるワークではなく、聞いた話を試すためのワークです。

約15分間　個人ワーク：モデル作成、曖昧な部分と想定される欠陥・故障の検討を試す

- 次の2つの作業を、できるところまでして、結果を紙またはPCにメモしてください
  - **描き仕様を整理**してください。複数描いてもOKです
  - **こりそうな欠陥・故障**も考えてください
- **分程で共有していただく心の準備を**お願いします

オープニングセッションでのワークと同じグループでお願いします

約5分間　グループワーク（テーブルごと）：他の人の着眼点や考えたことを知る

- 前側かつドア側の席の方から時計周りで、1人につき約1分で以下を共有してください
  - **お名前 (ニックネームOK)**
  - **個人ワークで考えたこと (どんなモデルを描いた・描こうとしたか、曖昧な部分、欠陥・故障)**
- 1分ごとに合図しますので次の方に交代してください

合図が聞こえたら途中でも話を止め交代しましょう

次はグループワーク開始ページ

ワーク　回答例（正解ではなく一例）

モデルを1つ思いついたら、他にもないか探してみることをお勧めします

ワークの回答例は非公開です

意外と難しいのが、着眼点の決定

テストの範囲が限定的な場合等は自然に見えてくることもある

が、そうでない場合はどうすれば？

着眼点の候補を知っておく

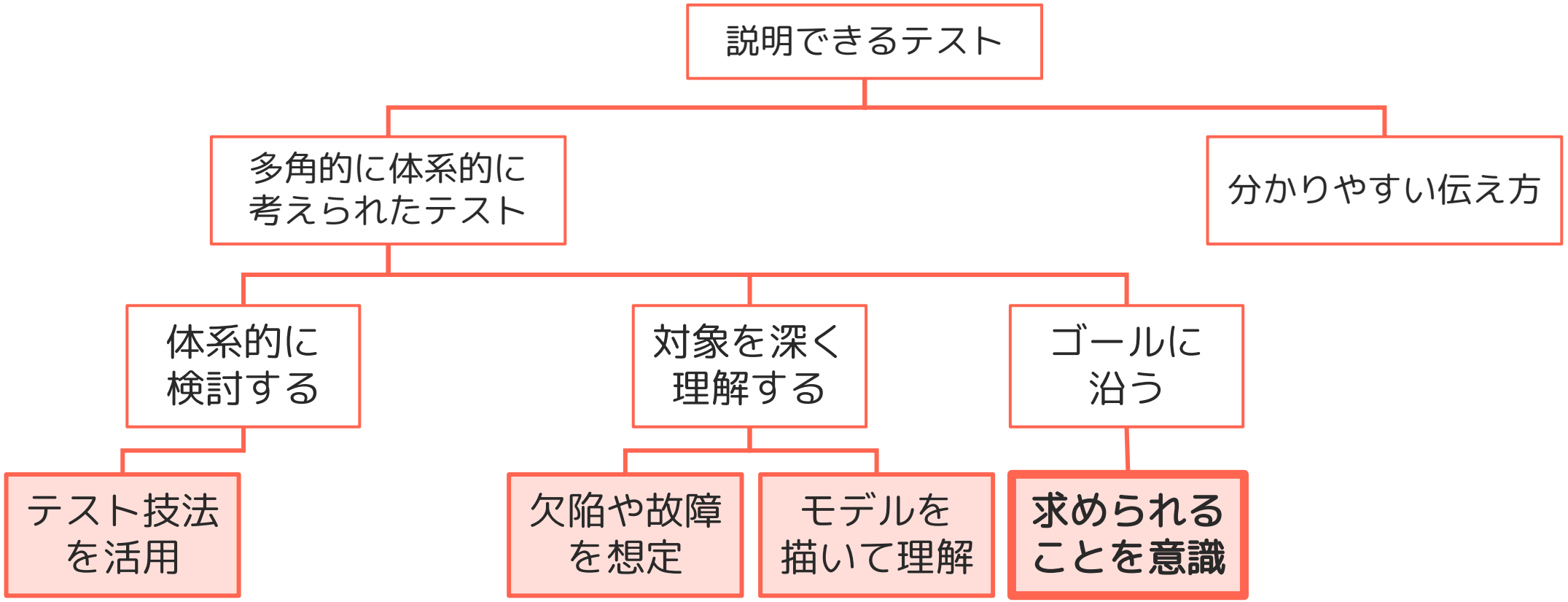
テストやソフトウェア設計について学び、関わるモデルを把握しておく

着眼点を思いつきやすくなるだけでなく、より早く思いつくためにも有効

目指すべきゴールに沿う

テストに求められていること（目的や要求）に沿ったモデルを選ぶ

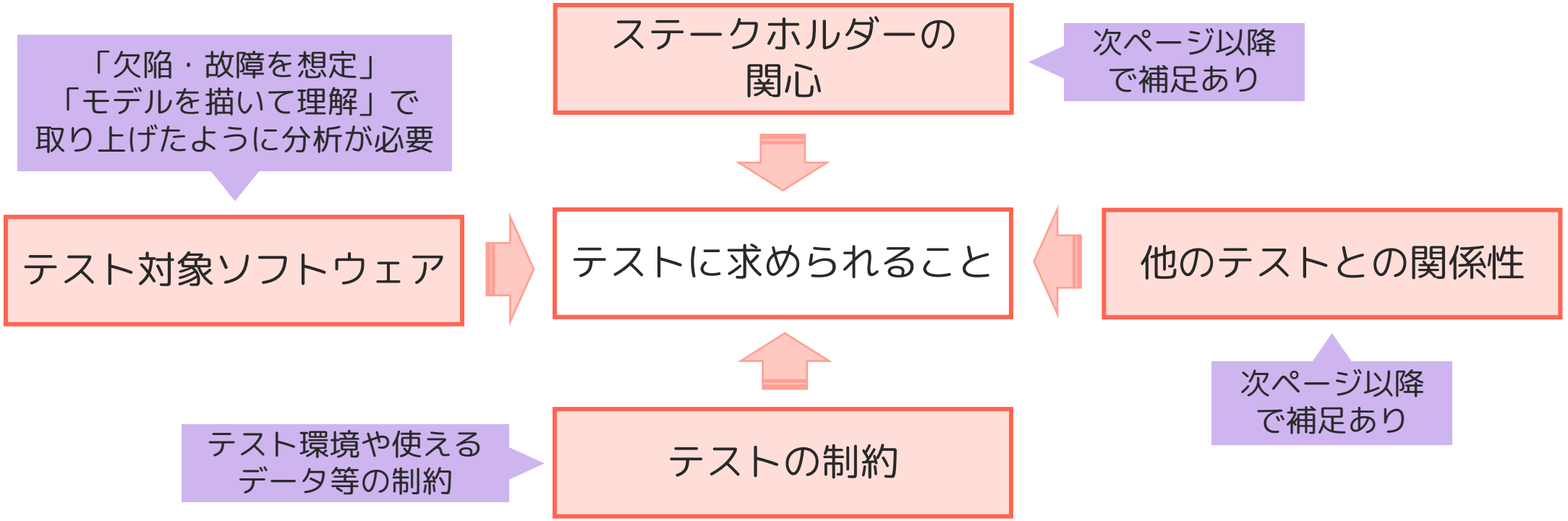
テストに求められることを意識する



テストに求められることは様々な背景より変化する

計画書等にテスト目的が書かれることもあるが、その背景にまで目を向けるとより良い

テスト対象ソフトウェア自体だけでなく、その周辺やテストの周辺も気にする必要がある



ステークホルダーの関心に注意をはらう

同じソフトウェアと関わっていても、こんな違いがあるかもしれない

マネージャー リリース日までに十分な機能提供ができるか	保守運用チーム リリース後の保守運用はしやすいか	監査チーム 監査に必要なログは出力されるか
営業担当者 担当のお客様が要望した機能は盛り込まれるか	顧客 新規顧客：使い方は分かりやすいか 既存顧客：遭遇した故障は直ったか	

テスト対象やステークホルダーの状態、社会の動きにも左右される

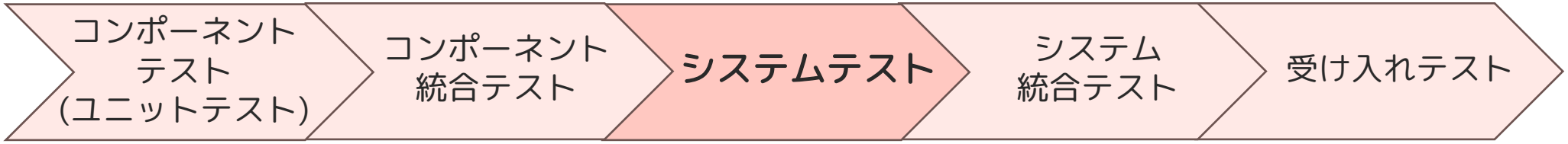
どのようなステークホルダーがいるか整理し、ヒアリングや観察等を通して
それぞれのテスト対象との関わり方等を推測、求められていることを考えてみる

技
欠
モ
求

他のテストとの関係性に注意をはらう

目の前のテストの位置づけや他のアクティビティとの関係性もテストの手掛かりになる

以下のうちシステムテストを担当する場合…



コンポーネント統合テスト以前の検証の内容やスコープは？

システム統合テスト以降の検証の内容やスコープは？

プロジェクトや開発全体として、必要十分な検証ができそうか？

本講演のスコープ外とはなるが、テスト実行開始以降も観察を続ける必要がある

例えば、前のテストで見つかっていると想定していた欠陥が多数検出されたら…？

すべて同時に実施するのは難しい

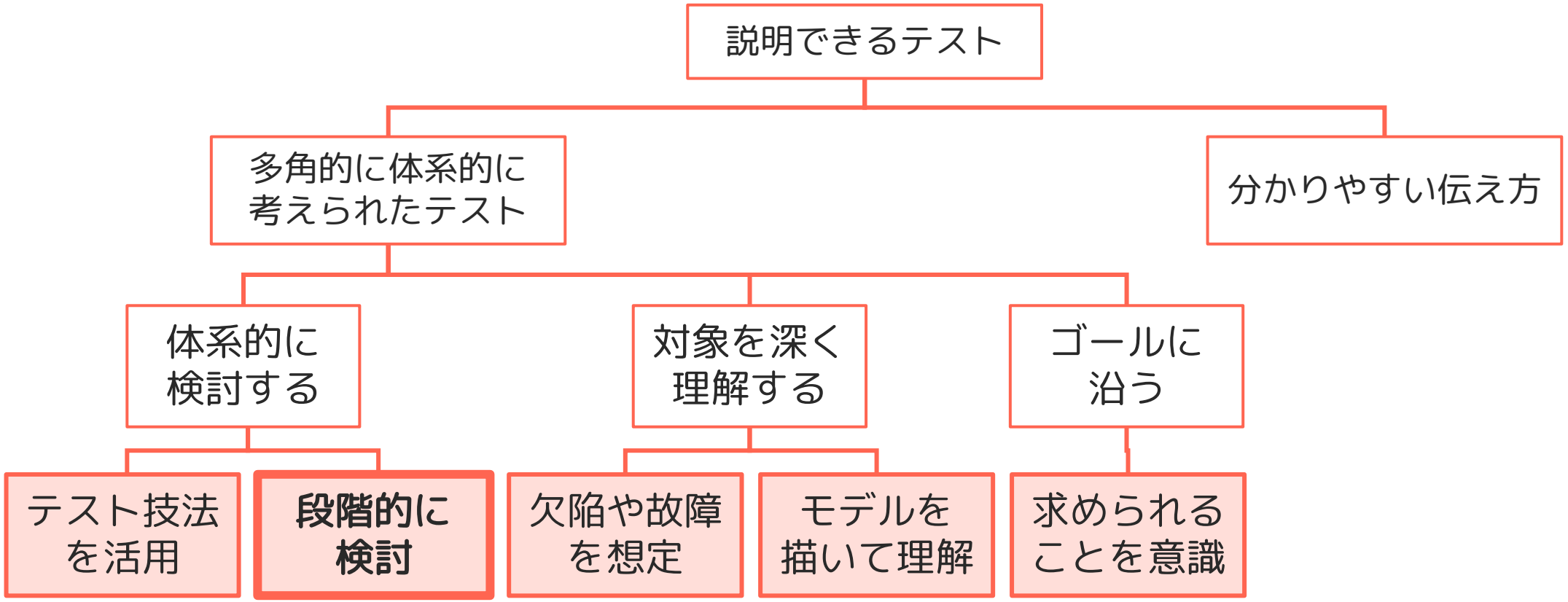
説明できるテストをつくるにはたくさんのことを行う必要がある

- テスト技法を活用
- 欠陥・故障を想定
- モデルを描いて理解
- 求められることを意識

一度に行おうとすると、抜け漏れたり十分考えないうちに次のことをやりたくなりがち

→体系的に進めやすくするために、段階を追ってテストを考える

テストプロセスを意識し段階的に考える



テストプロセスを意識し段階的に考える

普段どのようなプロセスを経てテストケースや手順を作成しているだろうか？　例えば…

テスト計画をつくる ▶ テストケースや手順をつくる

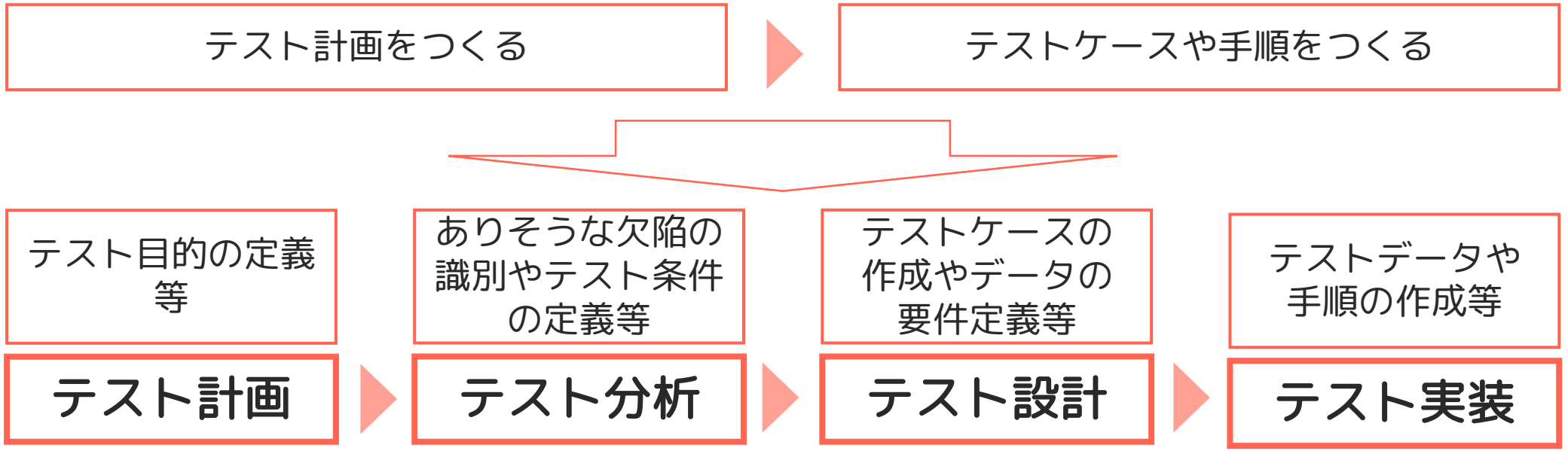
テスト対象のモデルを描いたり欠陥や故障を考えたり

テスト技法を使ったりしてテストを考えている場合は、もっと細分化できるはず

もう少し細分化すると、嬉しいことがある

- 一度に扱うことが絞られ考えやすくなる
- 途中経過を残しやすくなり、ふりかえりや変更への対応をしやすくなる

テストプロセスを意識し段階的に考える



複数を同時に進める場合でも、分けられることを意識して抜けないように進めるとよい

なお、テストプロセス全体としては他にも「テスト実行」等のプロセスがあることに注意

ここで「多角的に体系的に考えられたテスト」のための話は終わり、残るは「分かりやすい伝え方」

参考：https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf　1.4.1 テスト活動とタスク

参考　「分かりやすい伝え方」の話をする前に少し寄り道　その1
お申込時にいただいたご質問への回答(1/2)

ご質問への回答は非公開です

参考　「分かりやすい伝え方」の話をする前に少し寄り道　その1
お申込時にいただいたご質問への回答(2/2)

ご質問への回答は非公開です

参考　「分かりやすい伝え方」の話をする前に少し寄り道　その2
他の人のテストの進め方を見たいときは

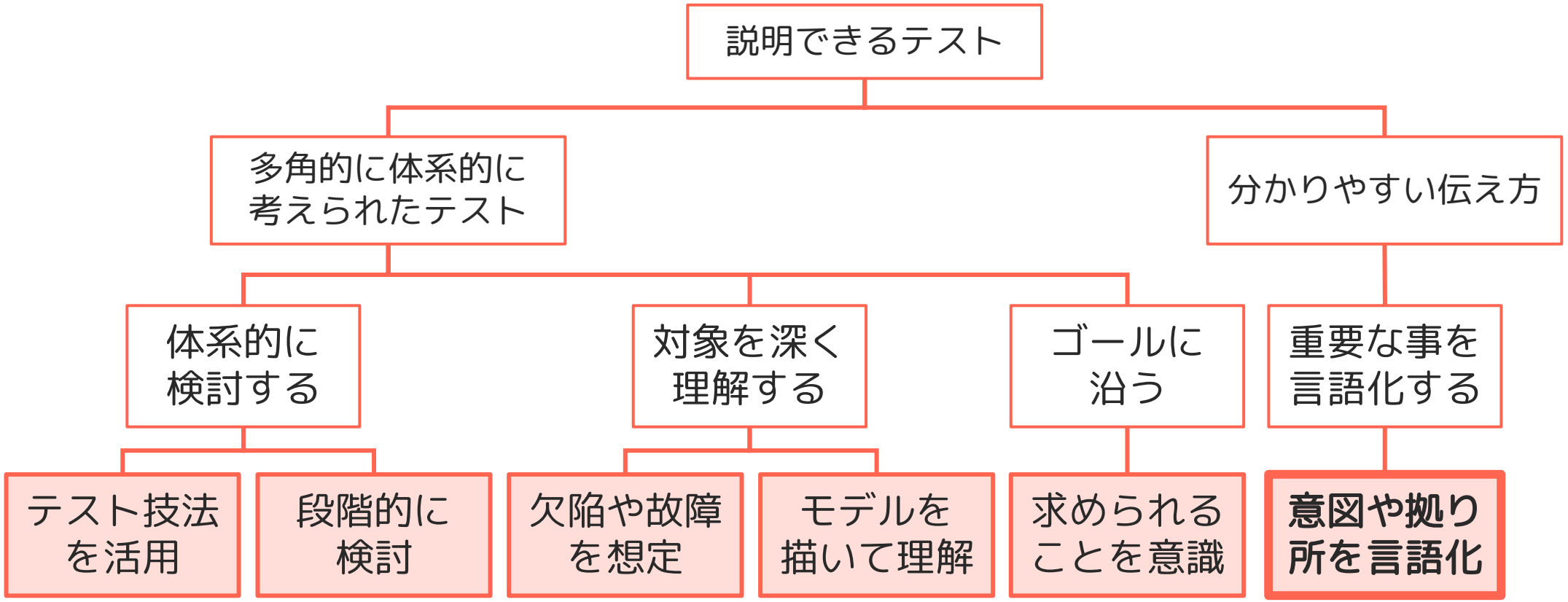
テスト設計コンテスト'23年(昨年度)の決勝戦出場チームの資料が公開されています。
一部資料ではありますが、どのようなプロセスや考え方を採用したのかが垣間見えます。

- https://www.aster.or.jp/testcontest/finalgameopen2023.html
- https://www.aster.or.jp/testcontest/finalgameu302023.html

他の人のテストの進め方が気になる方はご覧になってはいかがでしょうか？

画像の引用元：テスト設計コンテスト'23 U-30クラス 決勝戦レポート https://www.aster.or.jp/testcontest/finalgameu302023.html

更にうまく説明できるように、伝え方にも気を配る



意図や拠り所を言語化しよう

「なぜ、ここまでテストが必要なの？（これで十分なの？）」「そのテスト技法を選んだ理由は？」
「このテストはなぜ優先度高なの？」　「この品質特性だけテストするのはなぜ？」

→「仕様書通りです」、「前回と同じやり方です」…では、納得してもらえないかも？

- 仕様書に書かれていない項目間の組み合わせはテストしていない？
- 前回リリース後に起こった障害は考慮されていない？

実は、ただの「仕様書通り」「前回と同じ」ではないかもしれない

本当は意図や拠り所があるのに、うまく言語化できていないだけ、ではないだろうか？

意図や拠り所を言語化するには？

意図や拠り所を 自問自答する	比較して考える。どこが仕様書や前回と同じでどこが違う？ 一部変えてみる。そのテストをしないとどんな欠陥を見逃すか？
考えた過程を示す	途中の成果物を見せる。検討した故障、描いたモデル、等

意図や拠り所を言語化するには？

意図や拠り所を自問自答する

比較して考える。どこが仕様書や前回と同じでどこが違う？

一部変えてみる。そのテストをしないとどんな欠陥を見逃すか？

考えた過程を示す

途中の成果物を見せる。検討した故障、描いたモデル、等

成果物内・間の整合性をとる

例　意図は「多様なユーザーにとって使いやすいことの検証」、
　　でもユーザビリティのテストケースはない…？？？
要所要所で意図や拠り所を再確認する

「あえて提示しないと根拠が伝わらないことを意識しよう」

引用元：JaSST'22 Tokyo「テストの設計意図を届けよう」
https://www.jasst.jp/symposium/jasst22tokyo/pdf/C5.pdf 39ページ

本講演において意図、拠り所と呼んでいるものも同じ、
あえて提示しないと伝わらない

技
欠
モ
求
段
言

（時間があれば）序盤の話に補足　思いつきが悪いわけではない

内容自体の質が今一歩

思いつき（だけ）では、良くなさそう
→多角的に、体系的に考える

思いつき＝ダメ、ではない

思いつきの背景にある言語化できていない意図や拠り所を言語化した方がよい

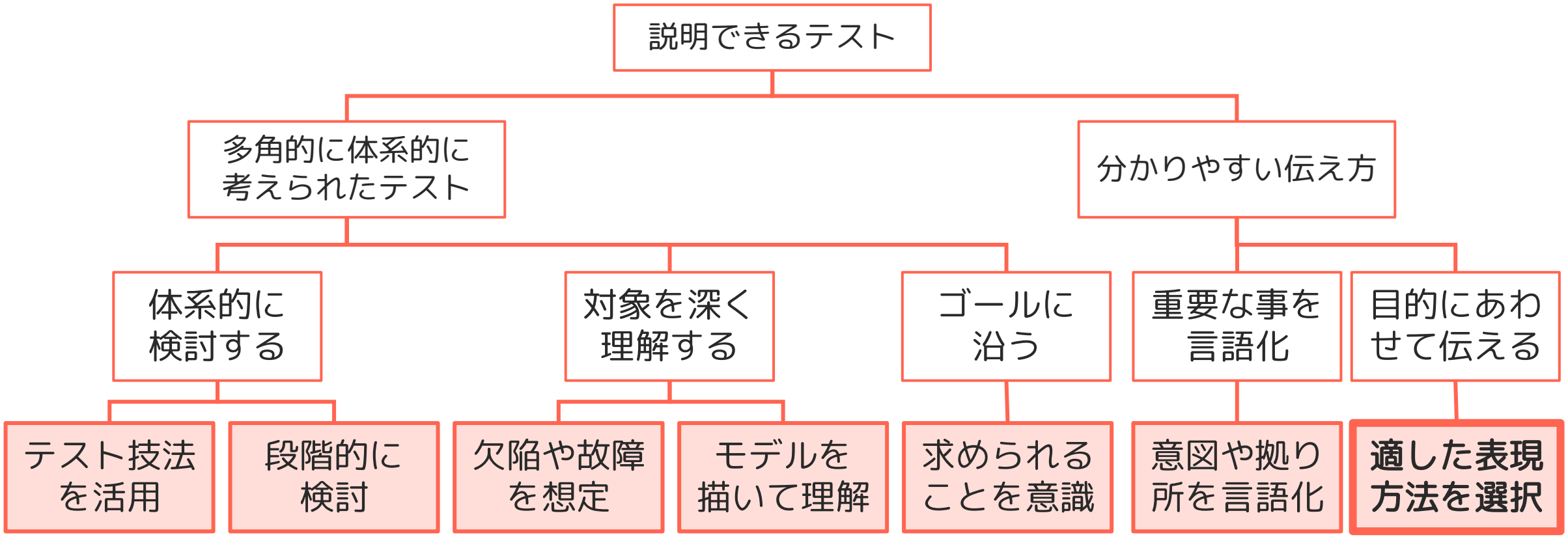
説明が容易になる

似た欠陥や関連するモデルを考える
等して更に思いつく

更に多角的に
考えられる

技
欠
モ
求
段
言

更にうまく説明できるように、表現方法にも気を配る



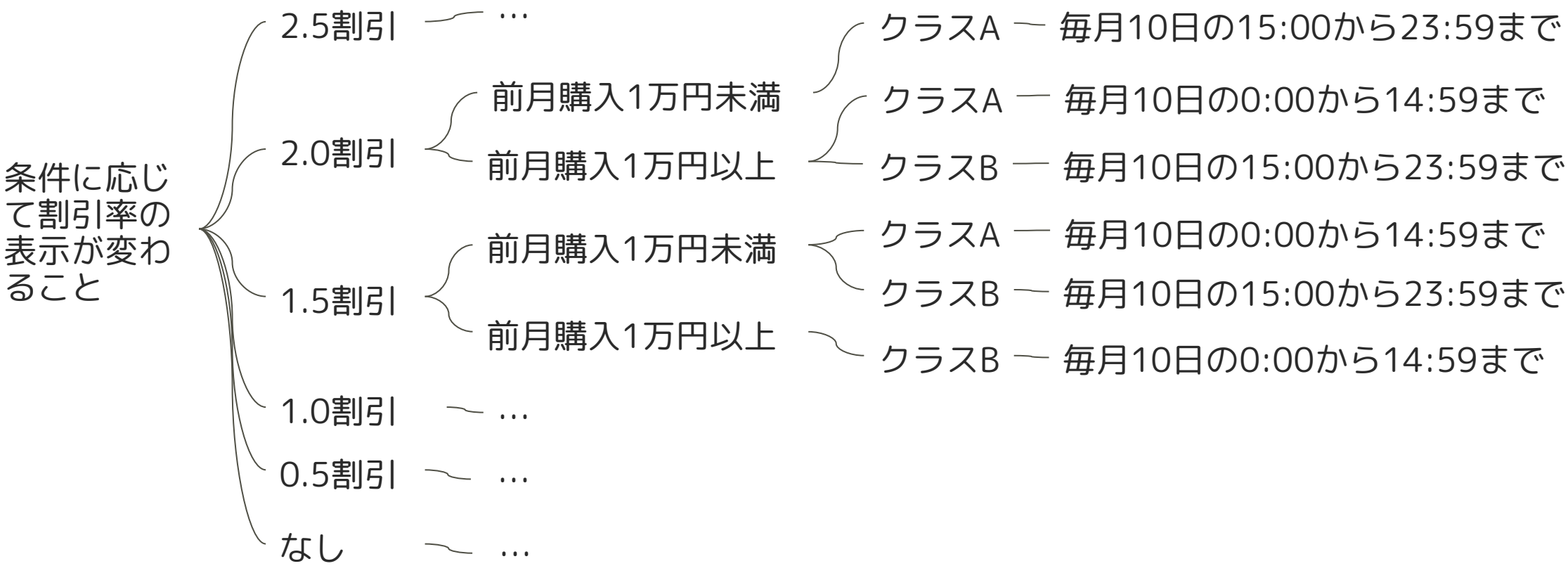
例　過不足ないか議論したい。テストの内容、理解できますか？

架空のECサイトの割引の仕様のテスト。過不足の議論をしたいが、すぐ理解できるか?

以下であることを確認する
- 前月購入1万円以上のクラスAの会員が毎月10日の0:00から14:59までに購入した場合は2.0割引と表示
- 前月購入1万円未満のクラスAの会員が毎月10日の0:00から14:59までに購入した場合は1.5割引と表示
- 前月購入1万円以上のクラスBの会員が毎月10日の0:00から14:59までに購入した場合は1.5割引と表示
- 前月購入1万円未満のクラスBの会員が毎月10日の0:00から14:59までに購入した場合は1.0割引と表示
- 前月購入1万円以上のクラスAの会員が毎月10日の15:00から23:59までに購入した場合は2.5割引と表示
- 前月購入1万円未満のクラスAの会員が毎月10日の15:00から23:59までに購入した場合は2.0割引と表示
- 前月購入1万円以上のクラスBの会員が毎月10日の15:00から23:59までに購入した場合は2.0割引と表示
- 前月購入1万円未満のクラスBの会員が毎月10日の15:00から23:59までに購入した場合は1.5割引と表示
- 前月購入1万円以上のクラスAの会員が上記以外の日時に購入した場合は1.0割引と表示
- 前月購入1万円未満のクラスAの会員が上記以外の日時に購入した場合は0.5割引と表示
- 前月購入1万円以上のクラスBの会員が上記以外の日時に購入した場合は0.5割引と表示
- 前月購入1万円未満のクラスBの会員が上記以外の日時に購入した場合は割引なしと表示

例　過不足ないか議論したい。テストの内容、理解できますか？

先ほどよりは見やすいかもしれないが、短時間での理解や議論開始は難しそう



例　過不足ないか議論したい。テストの内容、理解できますか？

これなら、条件や規模が一目で分かる

				1	2	3	4	5	6	7	8	9	10	11	12
条件	前月購入金額	1万円以上		Y	Y	Y	Y	Y	Y	N	N	N	N	N	N
		1万円未満		N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
	会員クラス	A		Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
		B		N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
	購入日時	毎月10日	0:00-14:59	Y	N	N	Y	N	N	Y	N	N	Y	N	N
			15:00-23:59	N	Y	N	N	Y	N	N	Y	N	N	Y	N
		上記以外の日	終日	N	N	Y	N	N	Y	N	N	Y	N	N	Y
動作	割引率	2.5割			X										
		2.0割		X				X			X				
		1.5割					X			X				X	
		1.0割				X							X		
		0.5割							X			X			
		なし													X

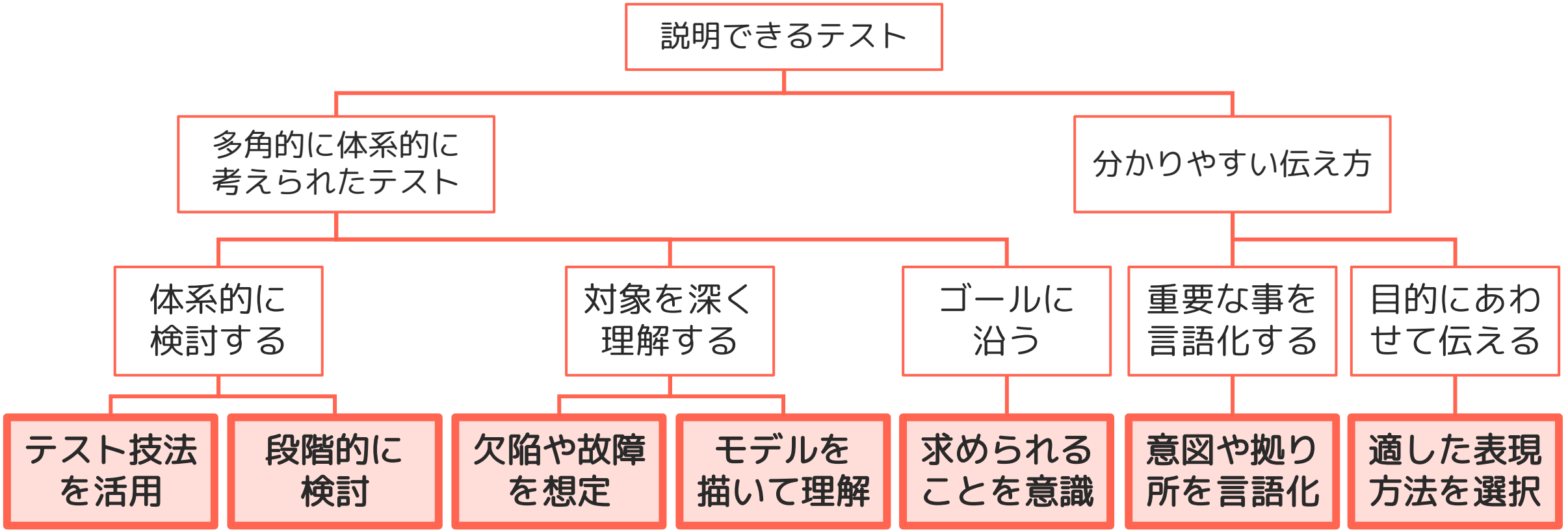
目的にあった表現方法を選ぶには？

説明の目的やどうなったら目的達成かを明らかにする	なぜ説明をするのか、どのような反応を得たら目的達成か、そのために何を理解してもらうべきか、を明確にする
考え方や表現方法を幅広く学び選択肢を増やす	ソフトウェアの要件定義や設計の考え方、一般的な図表等にヒントがあることも
一部だけでも表現してみて伝わりやすいか確認する	（セルフ）レビューを行い、目的を踏まえて伝わりやすさを確認する

「ドキュメントの字面通りに解釈して問題がないかどうか、用語やコード体系などの一貫性が保たれているかどうか、誤りがないかどうか――などは、**書いた直後にセルフチェックをしても見つけるのは難しいものです。ドキュメントを書いたときの記憶が鮮明に残っていて、自分の頭のなかで勝手に内容を補足してしまうからです。**最低でも1日は空けてチェックしてください。」

引用元：森崎修司(著). なぜ重大な問題を見逃すのか？間違いだらけの設計レビュー第3版. 日経BP. 2023年. 「2-1 レビューの準備（リーダー、作成者）」（太字青字の装飾は本資料作成者による）

説明できるテストをつくるためにできること



故障を想定すると拠り所を言語化しやすくなるなど、互いに促進し合う関係がある

やることが多い分、工夫の入口もたくさん

- 特にとっつきやすいのはテスト技法の活用
  - 書籍等の資料も入手しやすい、一人でも比較的学びやすい
  - まずは午後のセッションをお楽しみに！
- **「どれか1つだけ」実践して満足しても説明できるテストをつくることは難しい**
- **とっつきやすいところから少しずつ幅を広げながら、実践してみるのがお勧め**
  - 無意識にやっていた方は、繋がりも考えながら意識的に実践すると、
    更に説明できるテストをつくりやすくなるはず

次のページで終わりです！

テストは工夫のしがいがある活動

「**ソフトウェア開発は、要求を把握し設計原則を理解・適用して、 段階的に詳細化しながら順序立てて進めていく必要がある**
» **テスト開発も、テスト要求を把握しテスト設計原則を理解・適用して、 段階的に詳細化しながら順序立てて進めていく必要がある**
» これまで皆の経験とセンスを結集して規模の大きなソフトウェアを開発することである
» 経験とセンスを結集するために、 様々なソフトウェア工学上の概念や技術、方法論が必要となる」

引用元：JaSST'16 Tohoku「VSTePによるソフトウェアテストの開発」
https://www.jasst.jp/symposium/jasst16tohoku/pdf/S1.pdf 11ページ（太字青字の装飾は本資料作成者による）

テストは、決められた正解と照合する検算作業ではなく、
学びや創造が必要で、段階的な、順序だてて進めるべき活動と捉えると、
工夫のしがいがあるのは当然と言えるのではないでしょうか

この講演が少しでも、より良いものづくりのヒントになれば嬉しいです

参考文献(1/2)

本資料に掲載の引用文献・参考文献の情報は、すべて本資料作成時点の情報である。
本文中に記載していない参考文献について以下に記載する。順不同。

- テスト技術者資格制度　Foundation Level シラバス　Version 2023V4.0.J01
  （JSTQBのWEBサイトで公開されているシラバスのうちのひとつ）
  - https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf
  - 上記リンクが記載されているページ　https://jstqb.jp/syllabus.html
- ジョン・ソンメズ (著), まつもとゆきひろ (解説), 長尾 高弘 (翻訳). SOFT SKILLS ソフトウェア開発者の人生マニュアル 第2版. 日経BP. 2022年.　「第4章 社交スキルを鍛える」
- David Thomas (著), Andrew Hunt (著), 村上雅章 (翻訳). 達人プログラマー ―熟達に向けたあなたの旅― 第2版. オーム社. 2020年.
  「第1章　達人の哲学」
- Publickey『グーグルはコードの品質向上のため「バグ予測アルゴリズム」を採用している』
  - https://www.publickey1.jp/blog/11/post_193.html
- IT用語辞典　e-Words「モデリング 【modeling】 モデル化」
  - https://e-words.jp/w/%E3%83%A2%E3%83%87%E3%83%AA%E3%83%B3%E3%82%B0.html
- @IT　情報マネジメント【連載：モデリング修行 入門編】　「第1回 モデリングなしで開発はできない」
  - https://atmarkit.itmedia.co.jp/fjava/devs/modeling01/modeling01.html
- 児玉 公信 (著). UMLモデリング入門. 日経BP. 2008年.　「第1章 モデルが表現するもの」

参考文献(2/2)

- 開米瑞浩 (著). エンジニアのための図解思考 再入門講座. 翔泳社. 2010年.
- JaSST'23 Kyushu「わからない？をわかる！に変えよう！- QAエンジニアが実践している基本的な考え方と方法」（特に、24ページ「それぞれの役割になりきって、どんなサイトにしたいか書いてみよう」）
  - https://speakerdeck.com/____rina____/wakaranai-wowakaru-nibian-eyou-qaenziniagashi-jian-siteiruji-ben-de-nakao-efang-tofang-fa
  - 上記リンクが記載されているページ　https://www.jasst.jp/symposium/jasst23kyushu/report.html
- JaSST'13 Kyushu「ちょっと明日のテストの話をしよう」
  - https://www.jasst.jp/symposium/jasst13kyushu/pdf/S5.pdf
  - 上記リンクが記載されているページ　https://www.jasst.jp/symposium/jasst13kyushu/report.html
- JaSST'22 Tokyo「テストの設計意図を届けよう　～テストケース、テストスクリプトだけ渡していませんか？　- テスト設計コンテストU-30セッション -」
  - https://www.jasst.jp/symposium/jasst22tokyo/pdf/C5.pdf
  - 上記リンクが記載されているページ　https://www.jasst.jp/symposium/jasst22tokyo/report.html
- JaSST'23 Tokyo「テストの設計意図を届けよう2023　～テストしたいことを、よりスマートに伝えるための第一歩　- テスト設計コンテストU-30セッション -」
  - https://www.jasst.jp/symposium/jasst23tokyo/pdf/C6.pdf
  - 上記リンクが記載されているページ　https://www.jasst.jp/symposium/jasst23tokyo/report.html
- WACATE2020冬　招待講演資料「良いテストは良いフィードバック」
  - https://speakerdeck.com/omn/wacate2020w