“レビュー体系化”の経過報告
レビュー体系とレビューアーキテクチャ

JaSST Review2023実行委員会

Web告知

https://www.jasst.jp/symposium/jasstreview23/timetable.html

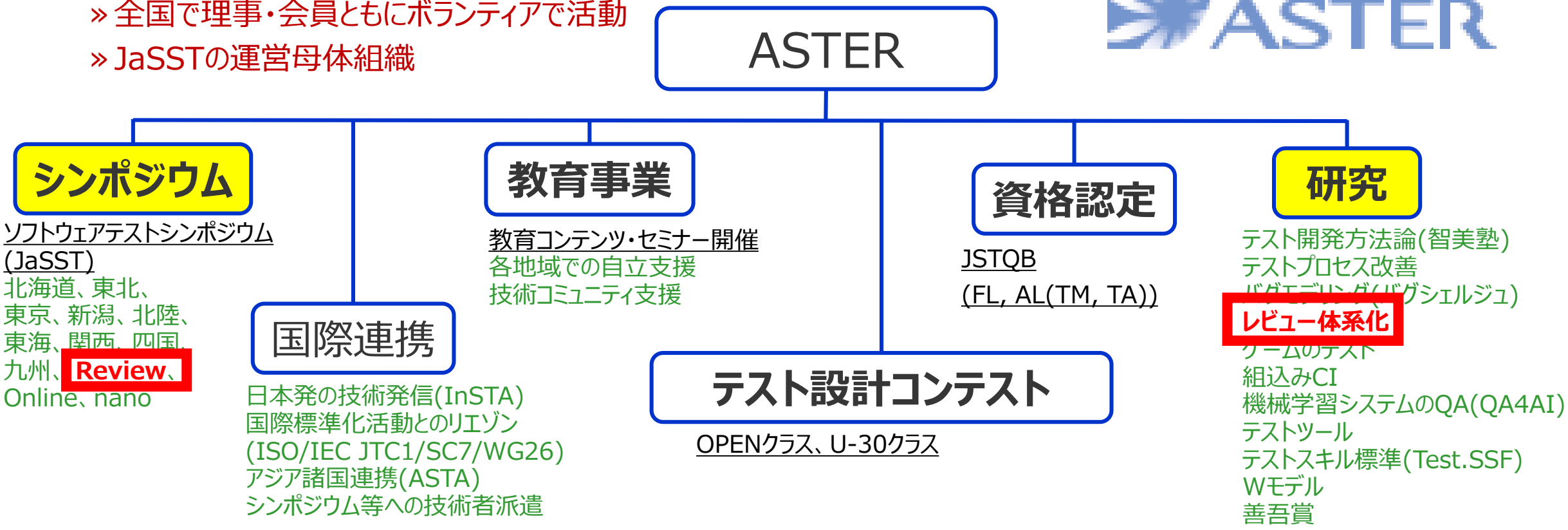
- **JaSST Review実行委員会では、「レビューの体系化」について普段から議論しています。**
  具体的には、テストの体系をレビューに置き換えて考えたり、実際にレビューしてそれぞれの観点の違いを体感したり、出てきた観点の言葉を直して分類してみたりといったことを進めています。
- 昨年のJaSST Reviewで実施したワークではその成果の一端をみなさんにお知らせしましたが、今回はその内容を含めた「レビュー体系の全体像とその構成要素」として、特にレビューアーキテクチャ設計、実際に実行委員でレビューした事例における観点や分類についてご紹介します。
- 現場のガチガチなレビューから開発とQAのちょっとした相談のようなレビューまで、みなさんのモヤモヤが少し晴れたり、レビューアとしての不安がちょっと減ったり、普段のレビューをより良くするヒントにできるようにお話ししたいと思います。

ASTER: ソフトウェアテスト技術振興協会

Association of Software Test EngineeRing

ソフトウェアテストを軸にして、ソフトウェア品質向上に関する研究開発、普及振興、教育、国際連携、資格認定などの事業を行う特定非営利活動法人(NPO法人)

» 2006年4月に設立
» 全国で理事・会員ともにボランティアで活動
» JaSSTの運営母体組織

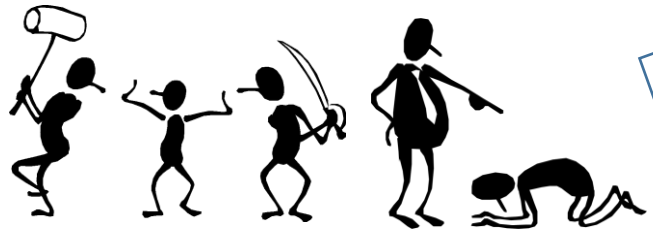


よくあるレビューの風景

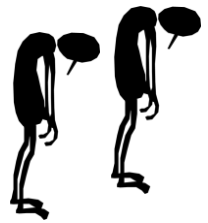
典型的なレビューの状態

レビュー会議で
初めてレビュー対象
を配付

ジャイアン独演会・あら捜し・
横道逸脱・人格否定・いざこざ



モチベダウン

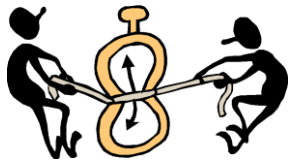


レビューの効果が
実感できない

書かれていることに反応し、
気が付いたことを指摘

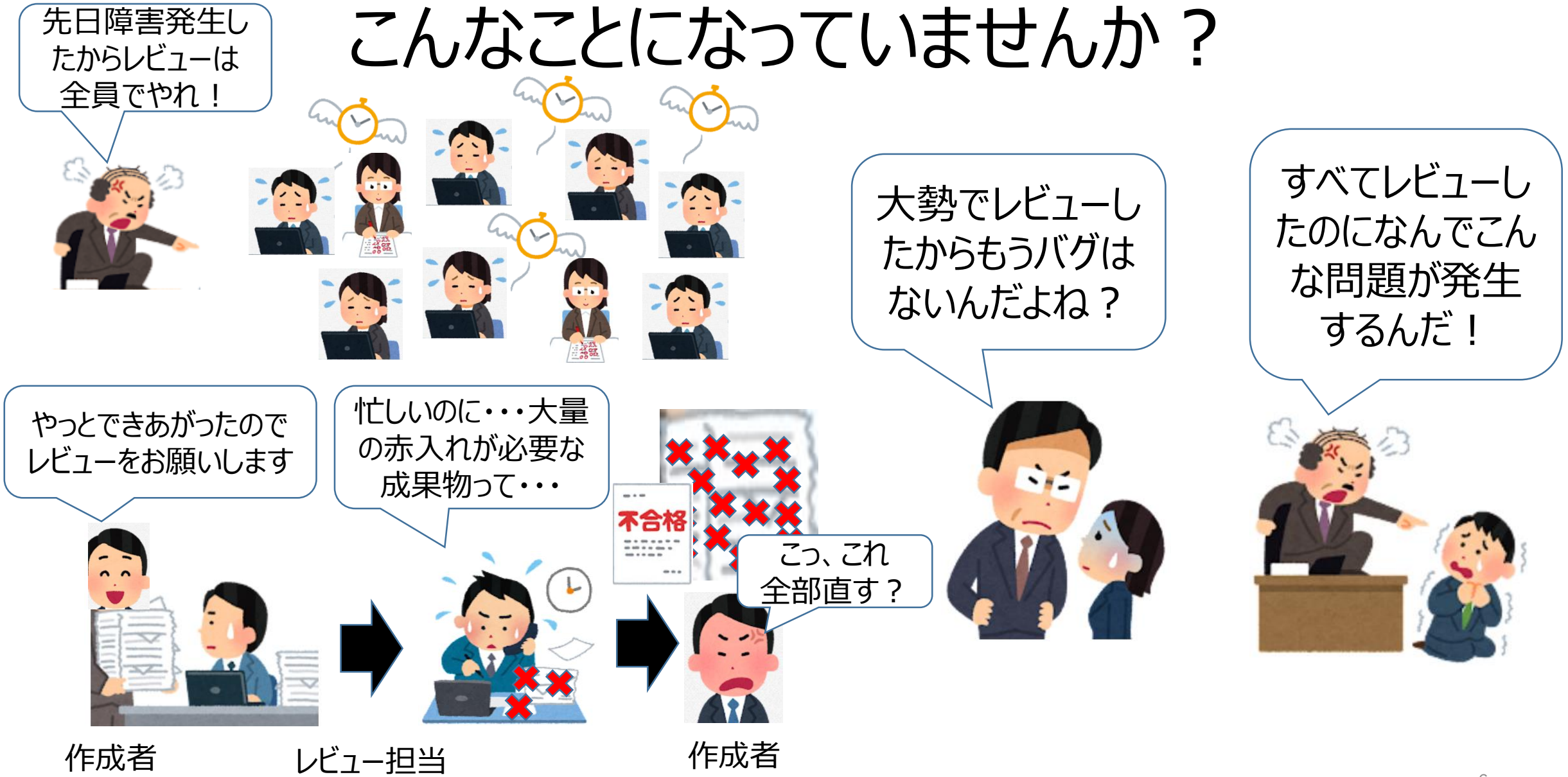
有識者がいないと軽微
で、表面的な指摘ばかり

多忙・前向きになれ
ない等の理由から
レビューを省略



見逃した欠陥が
あとになって爆発

こんなことになっていませんか？



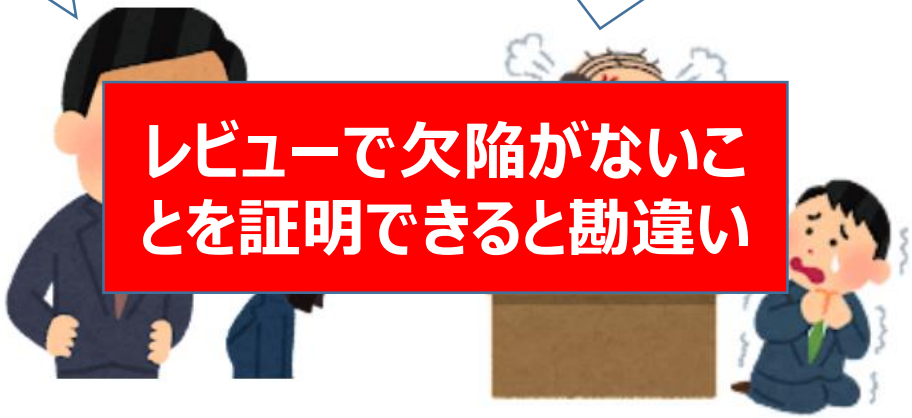
こんなことになっていませんか？

先日障害発生したからレビューは全員でやれ！

多重度を上げると漏れがなくなるんじゃないかレビュー

大勢でレビューしたからもうバグはないんだよね？

すべてレビューしたのになんでこんな問題が発生するんだ！



レビューで欠陥がないことを証明できると勘違い

やっとできあがったのでレビューをお願いします

忙しいのに・・・大量の赤入れが必要な成果物って・・・

対象がすべてできあがらないと実施しないレビュー

こっ、これ全部直す？

作成者　　レビュー担当　　作成者

理解不足

レビューへの誤認識や不適切な行動と結果

□レビューとは、レビュー会議でレビュー対象をチェックして指摘すること。
□レビューでは、対象を知るための情報源（例：仕様書やコード）に書かれていることを確認すればよい。
□レビューで十分な確認を行うには、レビューアの人数を多くすればよい。
□レビューチェックリストに記載されている項目を確認すれば終わりだ。
□レビューの成果は業務知識豊富な有識者が出席出来るかどうかで決まる。

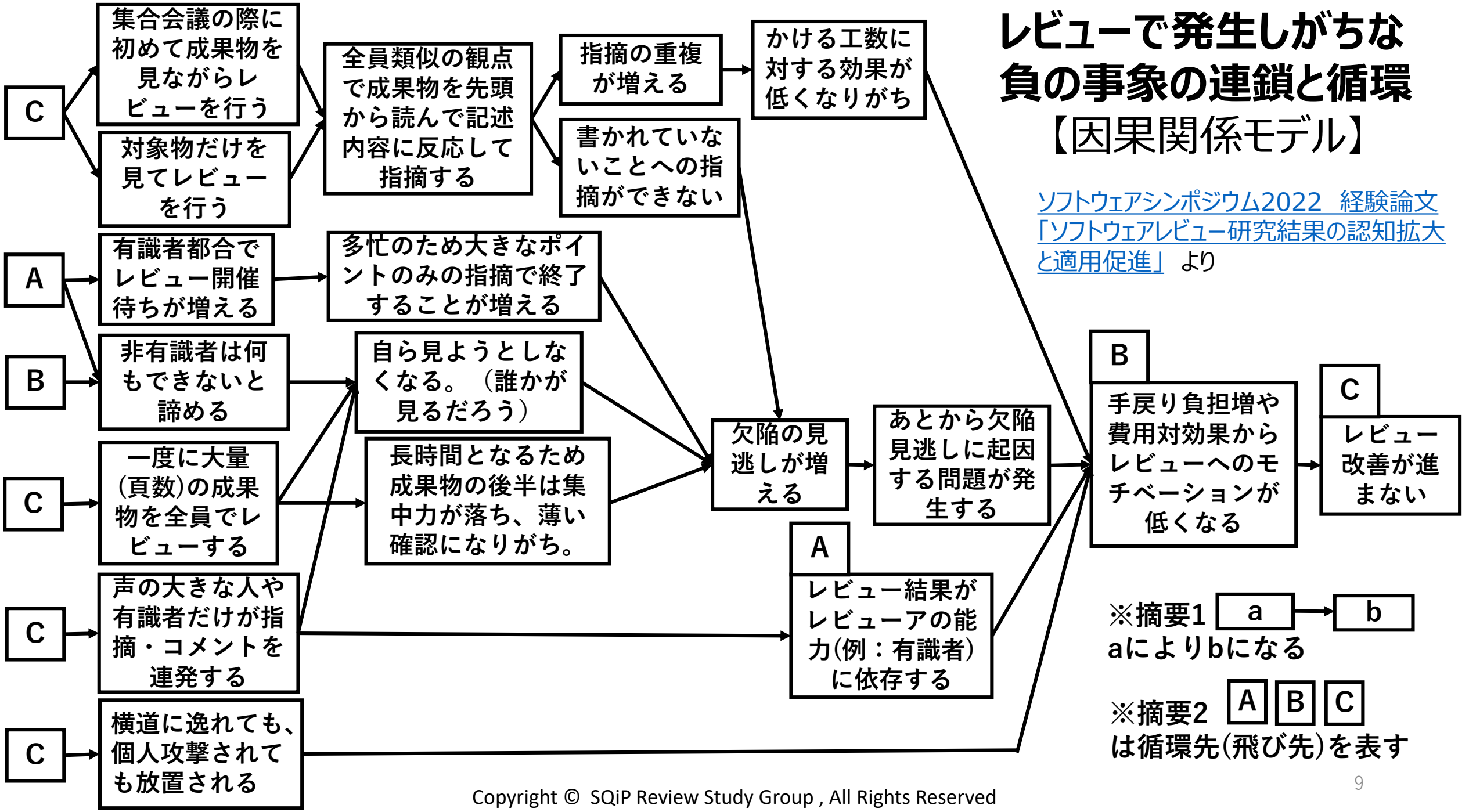
その結果・・・

□レビューでの見逃しが多く、後工程で問題が発生する。
□もうあの人の独演会レビューには参加したくない。
□レビュー時間がムダに長く、ダメ出しが多くて気が滅入る。
□業務知識が重要→レビュートレーニングなんて必要ない。
□レビューの効果が実感できない。

理解不足や誤認識は
不適切な行動を生み
イヤな結果を引き寄せる

レビューで発生しがちな負の事象の連鎖と循環【因果関係モデル】

[ソフトウェアシンポジウム2022　経験論文「ソフトウェアレビュー研究結果の認知拡大と適用促進」](#) より



JaSST2022東京ワークショップ参加者が感じていたレビューの問題点

レビューでの**見逃しが多い**(時支出来ていないことも？)

レビュー**指摘の出来が不安定**

工数がかかる
レビューイ理解度(レビューアの伝え方)

人によってレビューの観点と粒度がバラバラ／ほぼ機能していないレビューもある

レビュー**成果物の質**
発言・指摘が毎回同じ人
レビューMTGの進め方
レビューの効果が可視化できてない

レビュアーの勘と経験に頼ったレビューになってしまっている。

途中から参加したプロダクトについて**突っ込んでいいのか悪いのかわからない**

必要以上に時間がかかっている
レビューイの心構え（受け身）

経験則に頼りすぎている

レビューイとレビューア間の関係性が薄くて**協力関係になれなさそうなとき困る**

レビュー以前に、何がわからないのか探す習慣がない
非形式な**レビューをカジュアルにやってみるなど、取り組みができていない**
うまくいってない感を解消したいとおもえる、前向きな気持ちを育てたい
レビューを身近なものにしたい

ソフトウェアレビューシンポジウム2022東京　ワークショップ「そゆことね！よくわかるレビューテクニック～明日から使える技術をSQiPレビュー研究会からあなたに～」より

JaSST2022東京ワークショップ参加者が感じていたレビューの問題点

誤認識や**不適切な行動**で
欲しい結果が出ない・・・
この“もやもやした状態”を
レビュー体系化で変えたい！

レビュー
多い
ことも？

の関係
係にな
困る

レビュー
不安定

からな
ジュアル
組みが

工数
レビュ
ビュー

消したい
持ちを

験に頼ったレビュー
になってしまっている。

レビューを身近なものにしたい

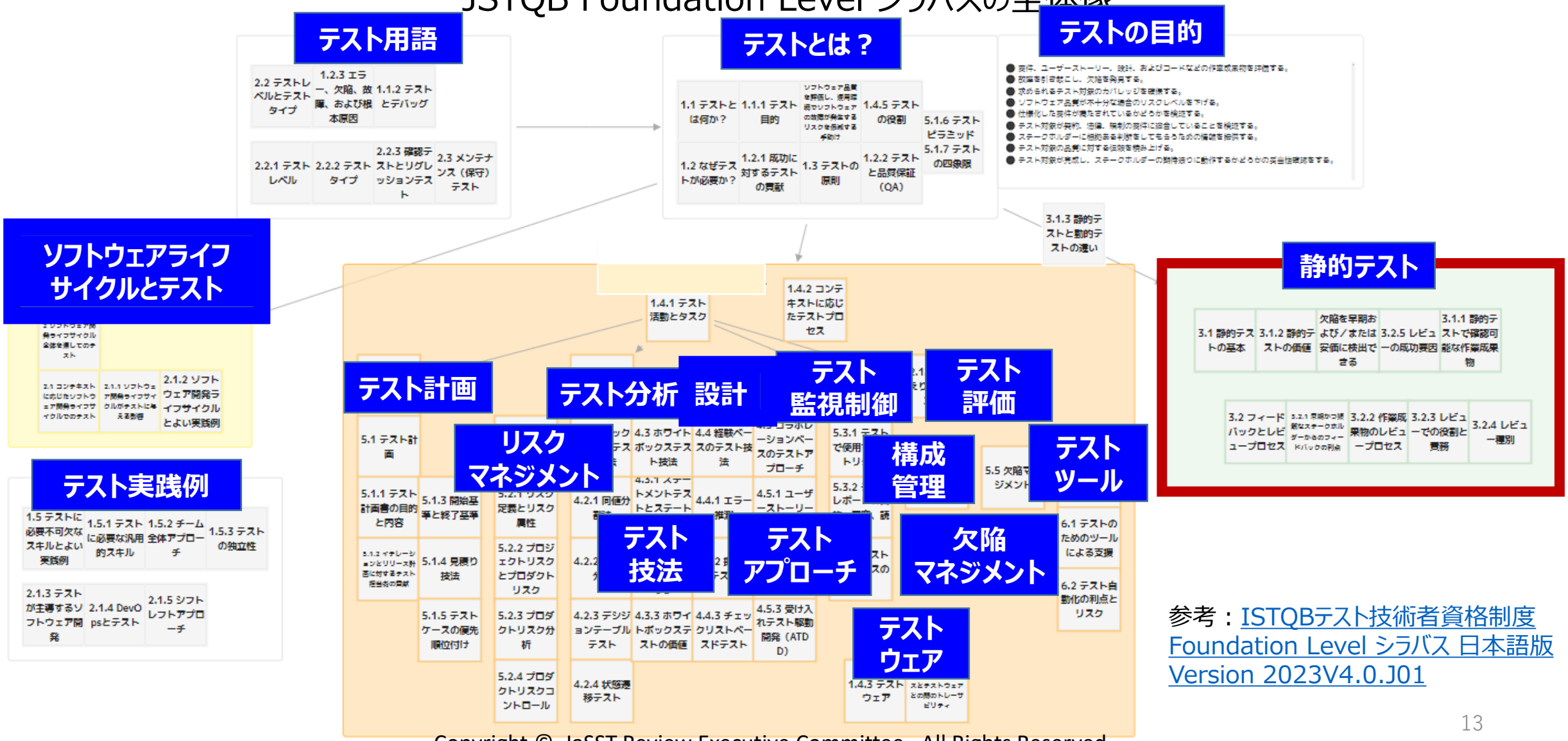
ソフトウェアレビューシンポジウム2022東京　ワークショップ「そゆことね！よくわかるレビューテクニック～明日から使える技術をSQiPレビュー研究会からあなたに～」より

われわれがやっていること
“レビューの体系化”

“体系化”とは、バラバラな個々の物事を一つにまとめ、わかりやすくすること

レビュー体系化のイメージ

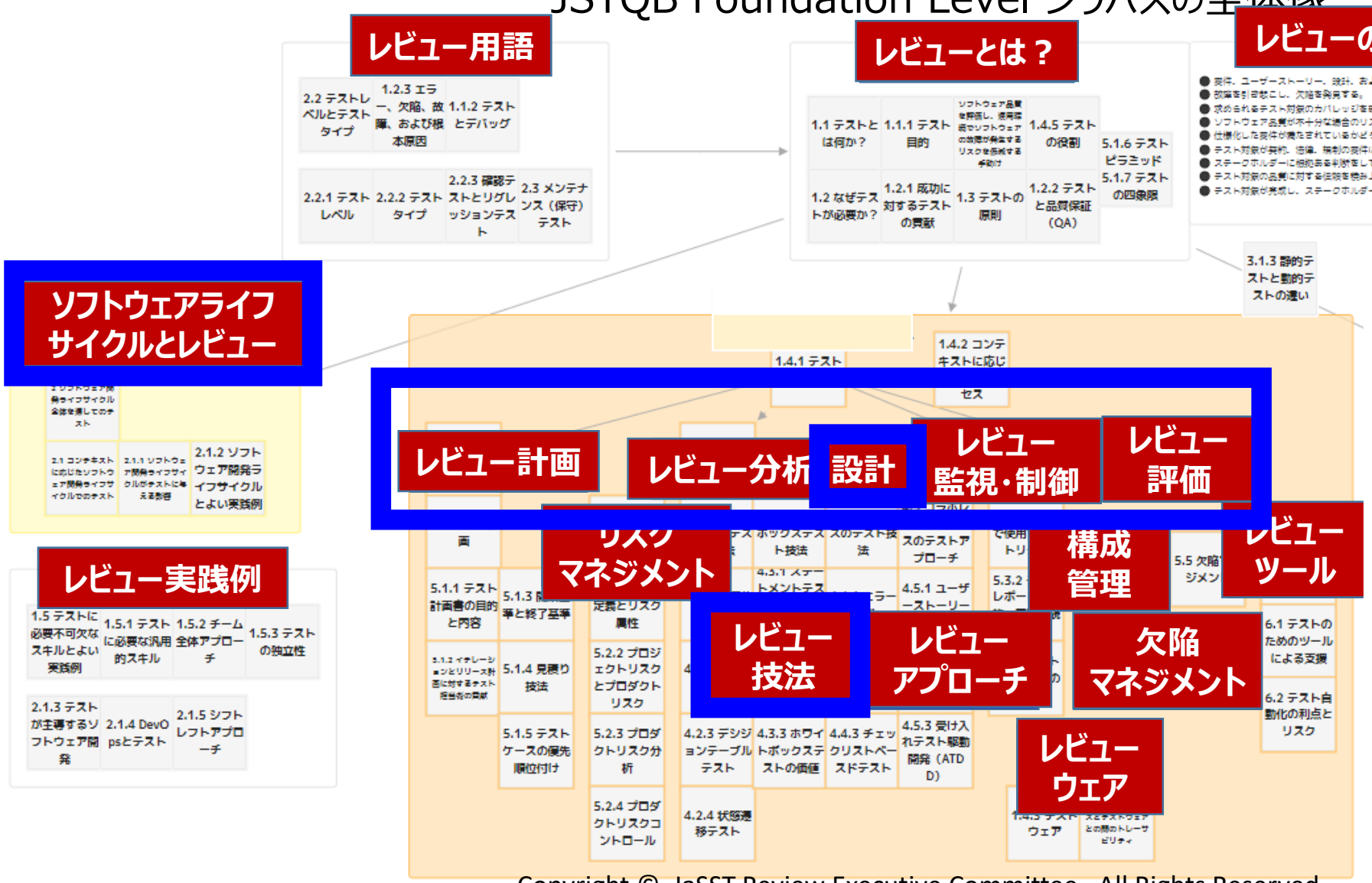
JSTQB Foundation Level シラバスの全体像



参考：[ISTQBテスト技術者資格制度 Foundation Level シラバス 日本語版 Version 2023V4.0.J01]

レビュー体系化のイメージ

JSTQB Foundation Level シラバスの全体像



今日はこれらのうち
(1)レビュープロセスと実践
(2)ソフトウェアライフサイクルとレビュー
(3)レビュー観点
(4)レビューアーキテクチャ
についてお伝えします。

参考：[ISTQBテスト技術者資格制度 Foundation Level シラバス 日本語版 Version 2023V4.0.J01]

レビュープロセスとその実践

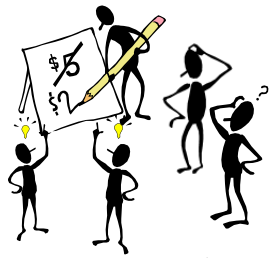
現状の「レビュープロセス」

ISTQBテスト技術者資格制度　Foundation Levelシラバス　Version 2023V4.0.J01

レビュー計画

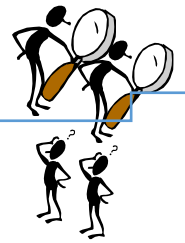
下記の項目を定義する
・目的、レビュー対象となる作業成果物
・評価対象の品質特性
・重点を置く領域
・終了基準
・標準などの裏付け情報
・レビューの工数
・時間枠からなるレビューの範囲

レビューの立ち上げ



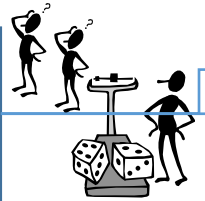
レビュー開始までに関係者や物事が準備できたことの明確化がゴール
・すべての参加者がレビュー対象の作業成果物にアクセスできる
・自分の役割と責務を理解している
・レビューの実行に必要なすべてのものを受け取っていること
を確認することが含まれる。

個々人のレビュー



・すべてのレビューアがレビュー対象の作業成果物の品質を精査
・1つ以上のレビュー技法（チェックリストベースドレビュー、シナリオベースドレビューなど）を適用
不正、推奨、および質問を識別する
・レビューアは、特定されたすべての不正、推奨、および質問を記録する

共有と分析



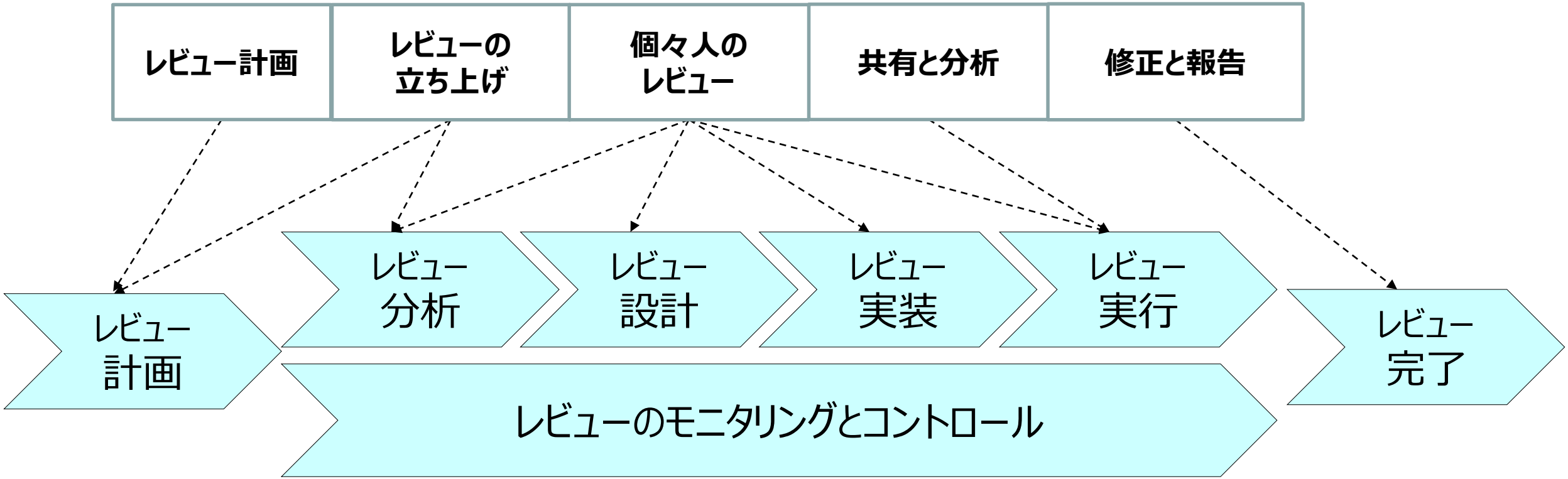
・通常、レビューミーティングを実施する
・すべての不正を分析して議論する
・すべての不正について、ステータス、オーナー、必要なアクションを決める
・レビュー済作業成果物の品質レベルと必要なフォローアップのアクションを決める
・完了するには、フォローアップのレビューが必要になる場合がある

修正と報告

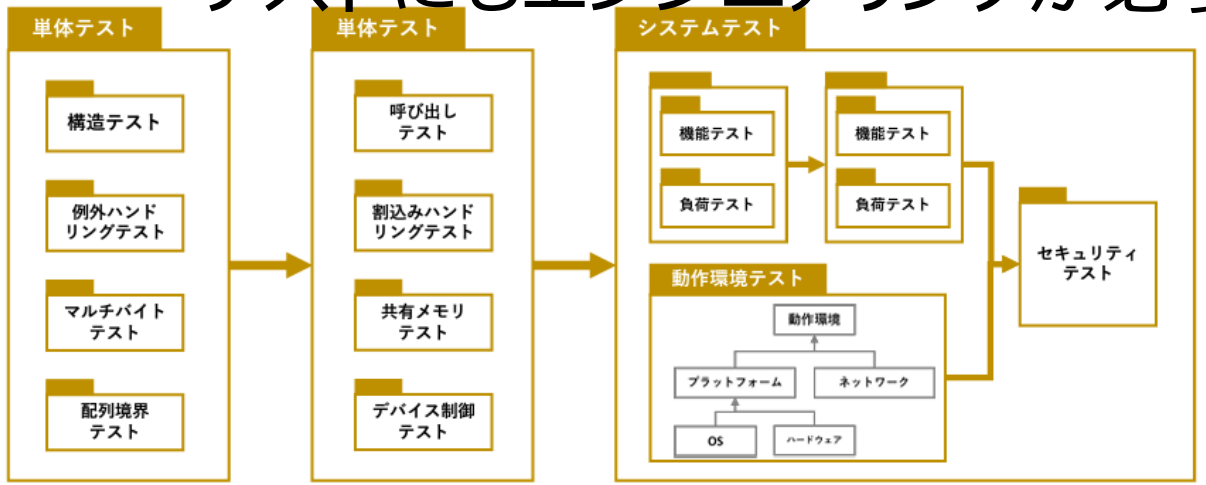
・是正処置をフォローアップできるようにすべての欠陥に対して欠陥レポートを作成する
・終了基準を満たす場合、作業成果物を受け入れる
・レビュー結果をレポートする

END

検討中の「レビュープロセス」
テストプロセスを参考にしたレビュープロセスの分解



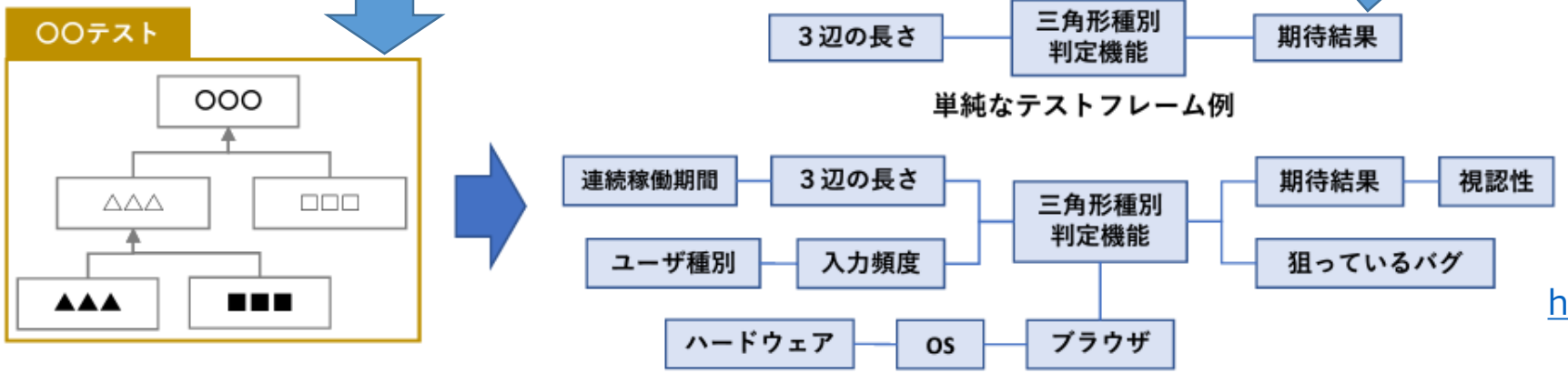
テスト体系に沿って実践すると
テストにもエンジニアリングが必要／テストケースの意図が重要



テストアーキテクチャ(設計)／コンテナモデリング

ID	3辺の長さ	ブラウザ	期待結果
1	(3, 4, 5)	Safari	不等辺三角形
2	(3, 3, 4)	Safari	二等辺三角形
3	(3, 3, 3)	Safari	正三角形
4	(3, 3, 6)	Safari	潰れて三角形にならない

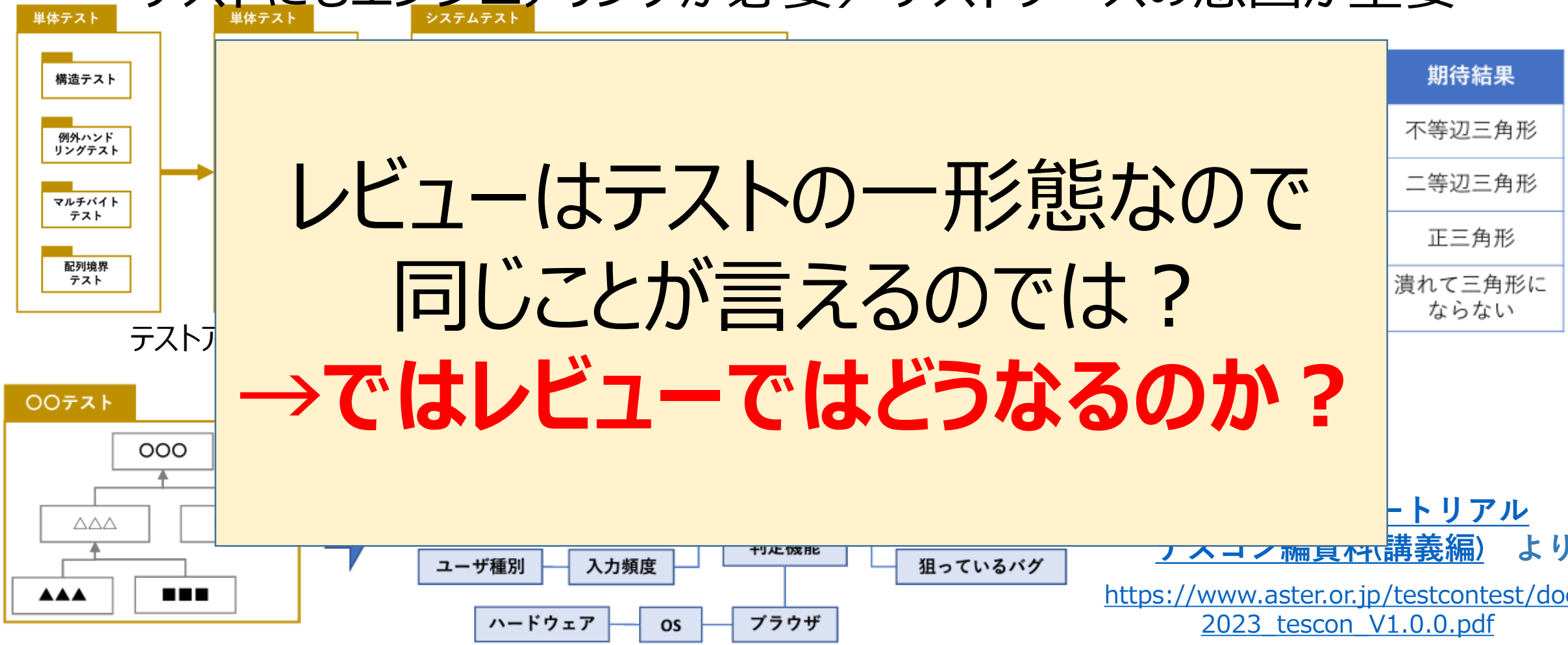
テストケース表



テストケースの構造化／テストフレーム

[テスト設計チュートリアル テスコン編資料(講義編)　より](https://www.aster.or.jp/testcontest/doc/2023_tescon_V1.0.0.pdf)

テスト体系に沿って実践すると
テストにもエンジニアリングが必要／テストケースの意図が重要

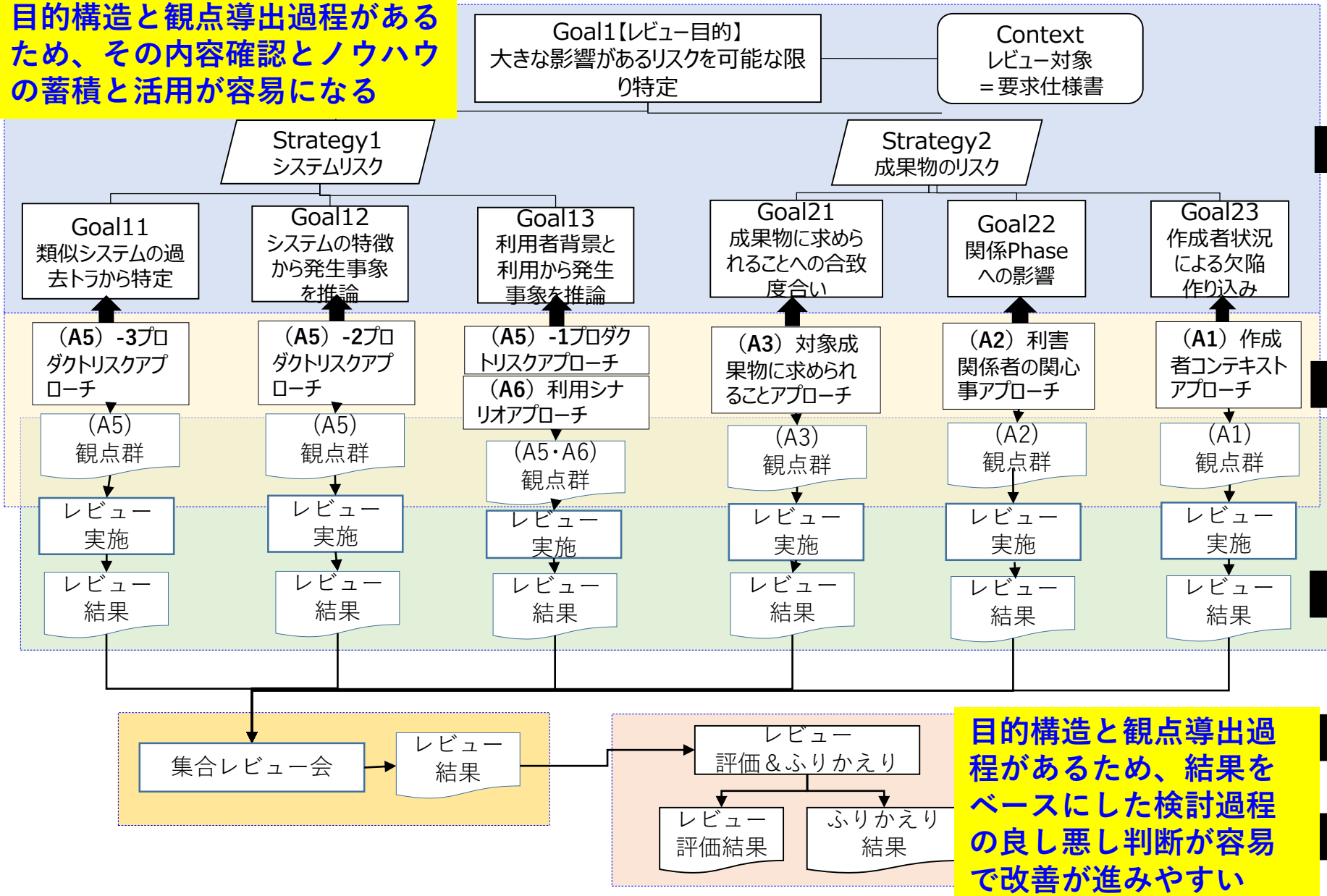


テストケースの構造化／テストフレーム

ートリアル
テスコン編資料(講義編) より
https://www.aster.or.jp/testcontest/doc/2023_tescon_V1.0.0.pdf

レビュー体系に沿った実践事例

目的構造と観点導出過程があるため、その内容確認とノウハウの蓄積と活用が容易になる



目的構造と観点導出過程があるため、結果をベースにした検討過程の良し悪し判断が容易で改善が進みやすい

レビュー目的を基軸とした目的構造化表記の採用[拡張]

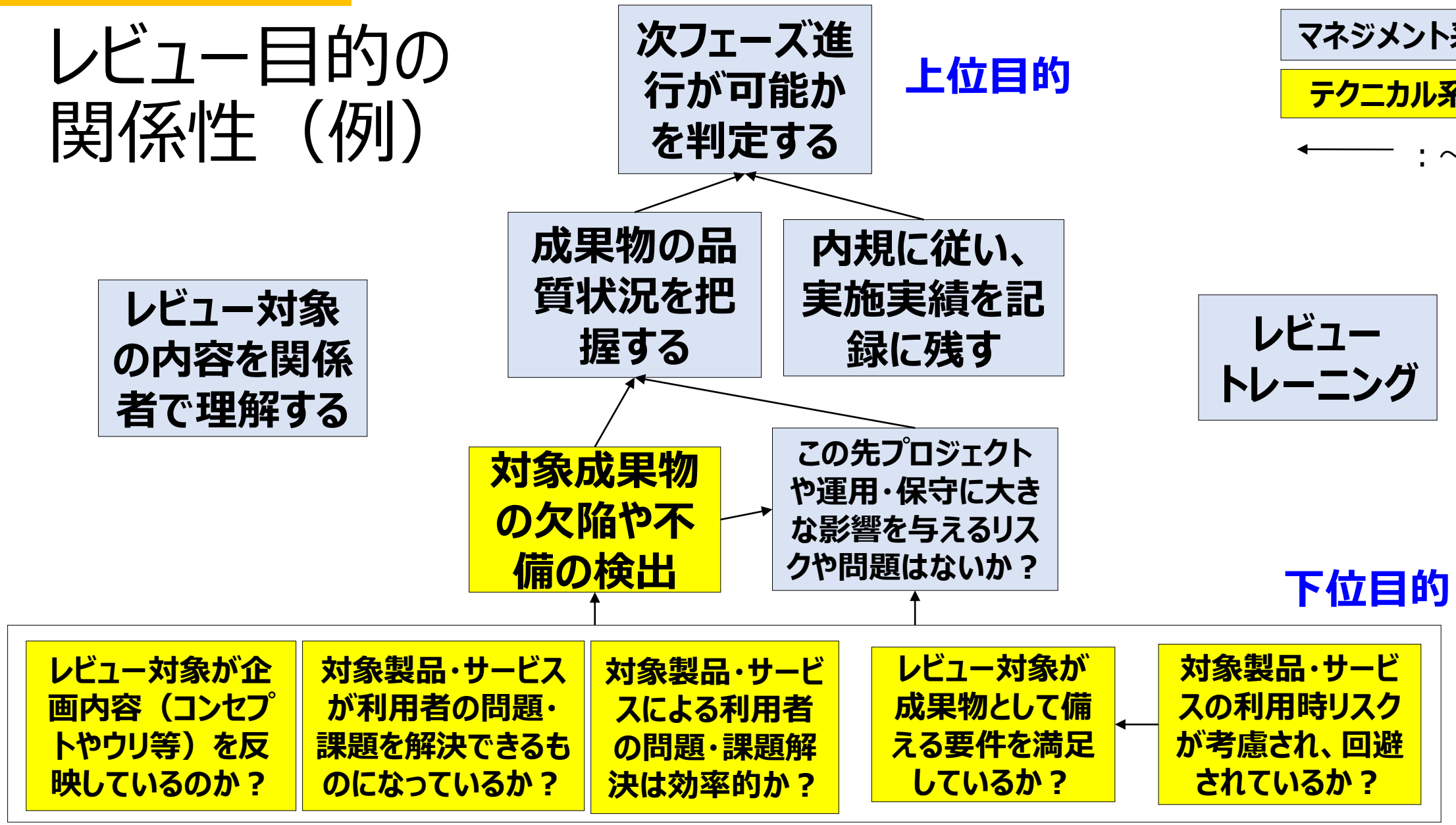
レビュー観点導出技法としての部品化と目的構造化表記への割り付けによるレビュー観点導出[技法バリエーション充実]

レビュー観点に基づくレビュー実施

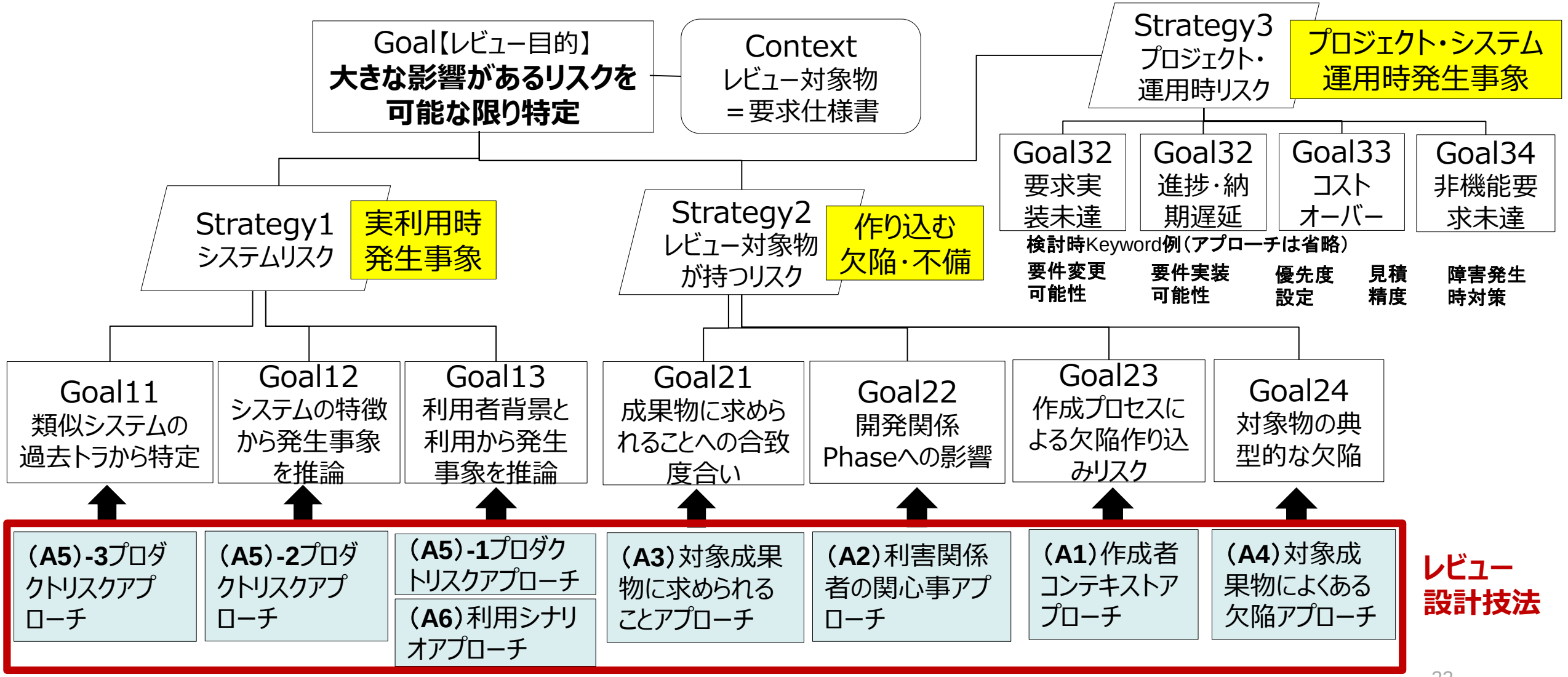
集合レビュー会での建設的議論による結果共有

レビュー結果評価[拡張]とふりかえりによるレビュー改善検討

レビュー目的の関係性（例）

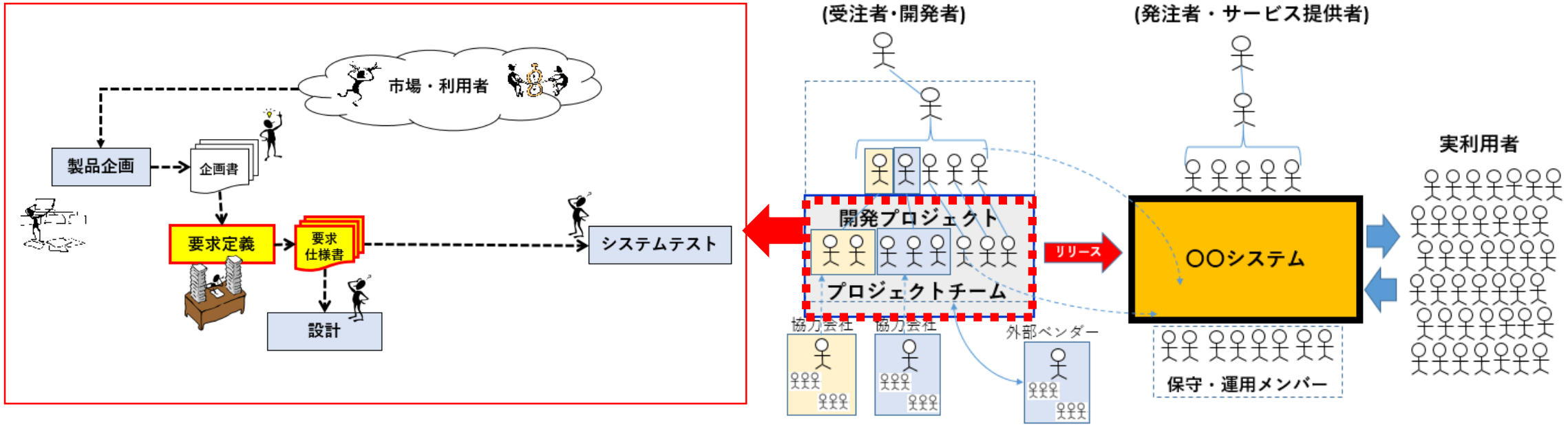


レビュー目的達成のためのアプローチ概要設計と観点導出アプローチの割り当て[ワーク結果例]



レビュー設計技法（例1）
利害関係者の関心事アプローチ

腕っこきレビューアの思考方法を見える化したもの



利害関係者	対象システムにおける主な活動	システムへの関心事（期待・疑問・懸念等）	必要なレビュー観点・対象
例．システム設計担当者	当システムの基本設計を担当	システム設計に必要な情報が漏れなく入手できるか？	システムに求められる要件が理由や目的、制約事項と共に明示されているか？

レビュー設計技法（例2）
対象成果物に求められることアプローチ

腕っこきレビューアの思考方法を見える化したもの

ソフトウェア要求仕様の目次例

観点	確認方法
1.はじめに ドキュメント目的 記述範囲 用語定義 参考文献 全体構成	例.目次に該当する事項が存在しているかを目視で確認する。
2.製品の背景と概要 製品の背景 製品機能 ユーザー特性 制約 要求項目の仮定と依存関係	
3.具体的な要求事項 外部インタフェース 機能 性能要求 論理データベース要求 設計の制約 非機能要求等ソフトウェア特性 要求仕様の段落構成	例.解決したい利用者の課題に対して必要な機能が漏れなく明記されているかを目視で存在を確認する

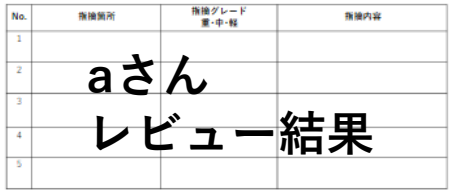
ソフトウェア要求仕様が持つべき特性

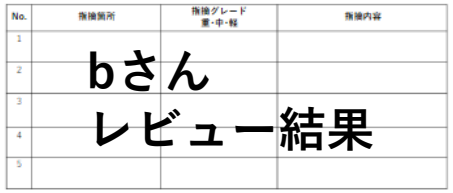
観点	確認方法
正当性（Correct） システムに対するすべての要求が含まれ、以外の要求を含まないこと	
無曖昧性（Unambiguous） 全ての要求の意味が一意に識別されること	
完全性（Complete） 次をすべて含んでいること＝(1)すべての必要な要求、(2)すべての入力データと状況に関する応答の定義、(3)用語および図表の説明	
一貫性（Consistent） 要求間で矛盾がないこと	
順位付け（Ranked for importance and/or stability） 要求が重要性や安定性に関して順位付けられていること	
検証容易性（Verifiable） すべての要求に対して有限のコストで評価可能な手続きが存在し、検証できること	
修正容易性（Modifiable） 要求の変更に対して、容易かつ完全で一貫性を保って修正できるような構造を持つこと	
追跡性（Traceable） **要求の根拠が明確で、開発工程全体で参照できること**	**例：要求の背景や理由→要求＋制約条件が漏れなく追跡できるかを企画書と要求仕様で目視確認する**

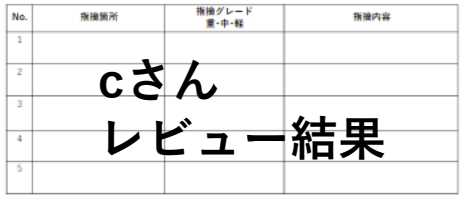
引用：要求工学：第3回要求仕様
https://www.bcm.co.jp/site/2004/2004Dec/04-youkyuu-kougaku-12/04-youkyuu-kougaku-12.htm

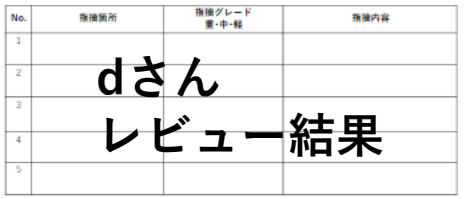
観点に基づくレビューの評価（例）

レビュー結果の見える化
→ふりかえりに活用


aさん
レビュー結果


bさん
レビュー結果


cさん
レビュー結果

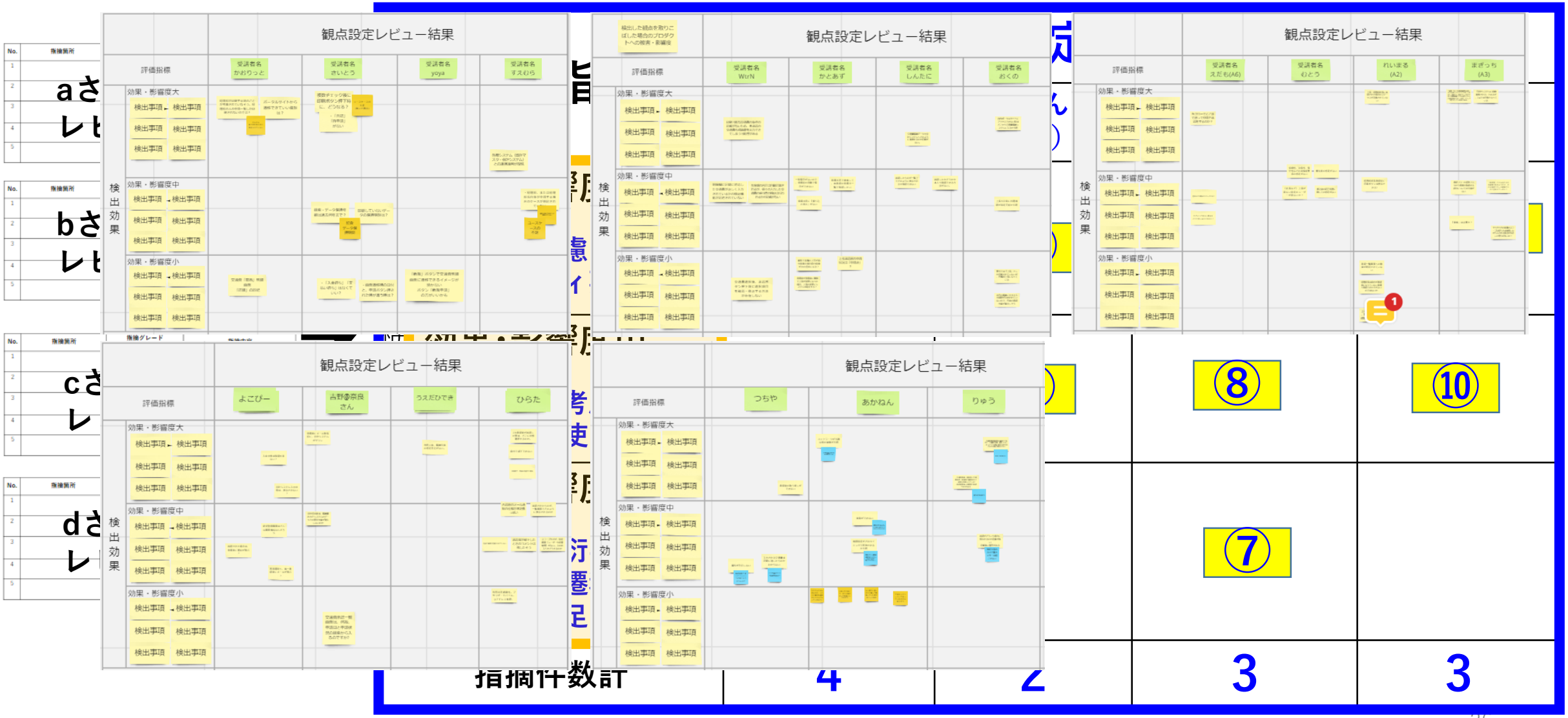

dさん
レビュー結果

評価指標	観点設定レビュー結果			
	aさん（A5）	bさん（A1）	c さん（A3）	dさん（A4）
効果・影響度大 主対象： 安全性未考慮 キャパシティ要件欠落	②　③	⑥	⑨	⑪　⑫
効果・影響度中 主対象： 性能要件未考慮 機能不足→使いにくい	①	⑤	⑧	⑩
効果・影響度小 主対象： 誤字・脱字・衍字 部分的画面遷移なし 画面項目不足	④		⑦	
指摘件数計	4	2	3	3

※記述内容は事例

観点に基づくレビューの評価（例）

レビュー結果の見える化
→ふりかえりに活用



※記述内容は事例

各チームのレビュー導出技法選択と指摘数

チーム	人数	レビュー観点導出技法選択と指摘数						指摘数計
		M1	**M2**	**M3**	**M4**	**M5**	**M6**	
A	4	3-2	6-2	1-0		2-1		12-5
B	4			1-1	9-3	3-1	5-2	18-6
C	4	4-1			3-1	4-1	6-0	17-3
D	3	6-5	3-1				3-1	12-7
E	4		5-1	6-2	4-2		3-1	18-6

※全指摘数(m件)とそのうちの効果大の指摘数(n件)を**"m-n形式"**で示した　　計77-27

某組織でのワーク結果

各チームのレビュー導出技法選択と指摘数

チーム	人数	レビュー観点導出技法選択と指摘数						指摘数計
		M1	**M2**	**M3**	**M4**	**M5**	**M6**	
A	4		3-1	3-0		2-0	3-2	11-3
B	4		3-0	2-1		3-1	4-0	12-2
C	4		1-0	1-1		1-1	8-2	11-4
D	4			1-0	1-1	2-0	7-2	11-3
E	4	1-0	0-0	4-2			2-1	7-3
F	3	4-1	3-1				8-3	15-5

※全指摘数(m件)とそのうちの効果大指摘数(n件)を**“m-n形式”**で示した　　計67-20

ソフトウェアシンポジウム2022　経験論文 「ソフトウェアレビュー研究結果の認知拡大と適用促進」 より

ワークふりかえり結果

レビュー依頼

レビューイーのセルフチェックでも使える

レビュー対象を確認する事前準備の時間は大事

深い知識なくてもある程度のレビューできそう
見てほしい所を伝えておくとかで

レビュー依頼する時に、「XXの観点からレビューお願いします」みたいなコミュニケーションするとレビューの負担も下がるし精度も部分的にあがるかもしれない

今回のような仕様レビューに限らず、すべての「レビュー」と言える活動すべてで事前に観点を定めたり宣言したりするのは有用そう。少なくとも宣言することは明日から取り組みたい。

レビュー分析・設計・計画

レビュー導出アプローチをテーラリングして各プロジェクトに適するものにして使用するとよいかと思いました。

誰がどのスキル・レビュー観点にふさわしいか
把握する必要がある

抽象、具体と多角的にアプローチできた
組み合わせるとパワフル

チェックリスト作成がゴールでないため、
目的に応じて選択できるようになりたい

A2
自分と違う意見や背景からアイデアが出せた

レビューの設計はよさそう。使いまわしていきたい

色んな人とレビュー観点話し合えるって新鮮

レビュー観点導出のレビューをしたくなる

レビュー導出アプローチに慣れが必要でここにもスキルレベルの差が観点に出てしまうのではないかと思いました。

レビュー実施

過去の知見からのレビューになっていたが、レビュー観点導出で行うとまた違う観点で洗い出せるので良いと思った

A2
はユーザーの使い勝手などの観点からレビューできた

自分の担当箇所にフォーカスする。これにより楽になる一方、自分の担当範囲に対する緊張感と、担当範囲外への不安が生まれる

観点によって発見できる問題がぜんぜん違う

レビュー評価・ふりかえり

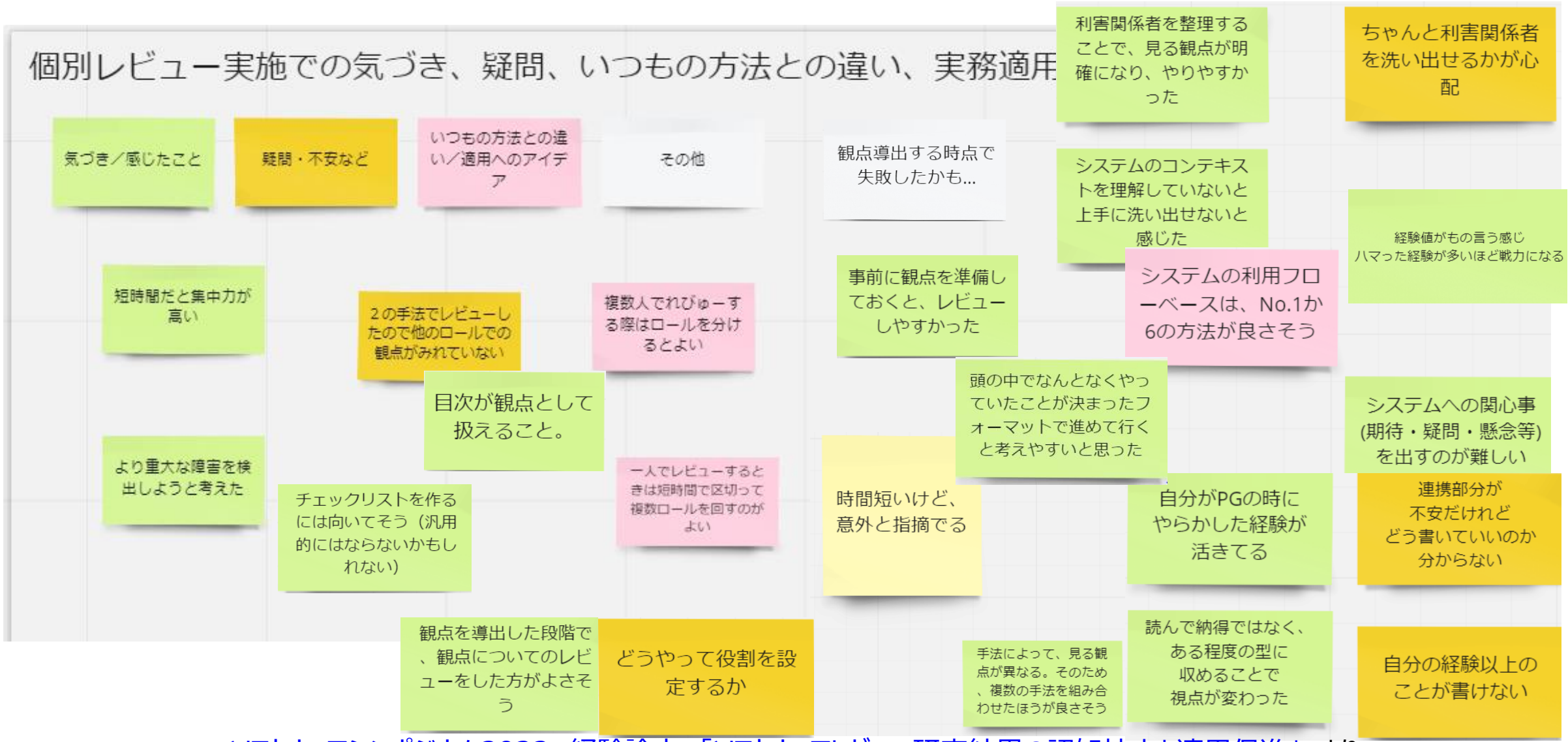
「良い観点」についての共通認識を取るのは難しそう。日々のレビューを定期的に振り返って、良いレビュー観点を育てていきたい。

レビューを振り返るプロセス、考えは今までなかった

派生開発で観点の使い回しできそうなので、観点を資産化する

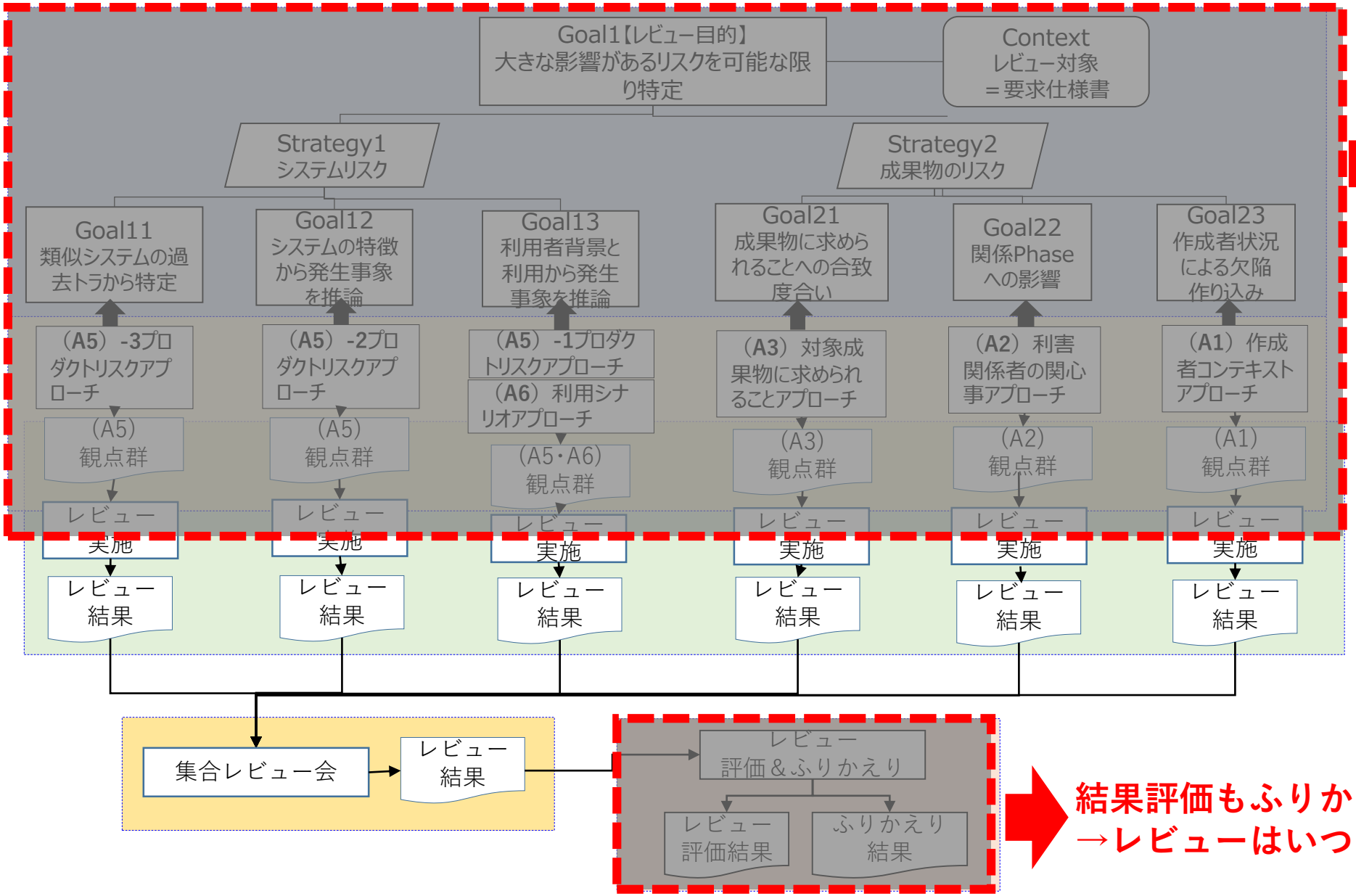
体系的なレビュースキルの蓄積、向上につなげられそう

ワークふりかえり結果



[ソフトウェアシンポジウム2022　経験論文「ソフトウェアレビュー研究結果の認知拡大と適用促進」](#) より

よくあるレビューアプローチ



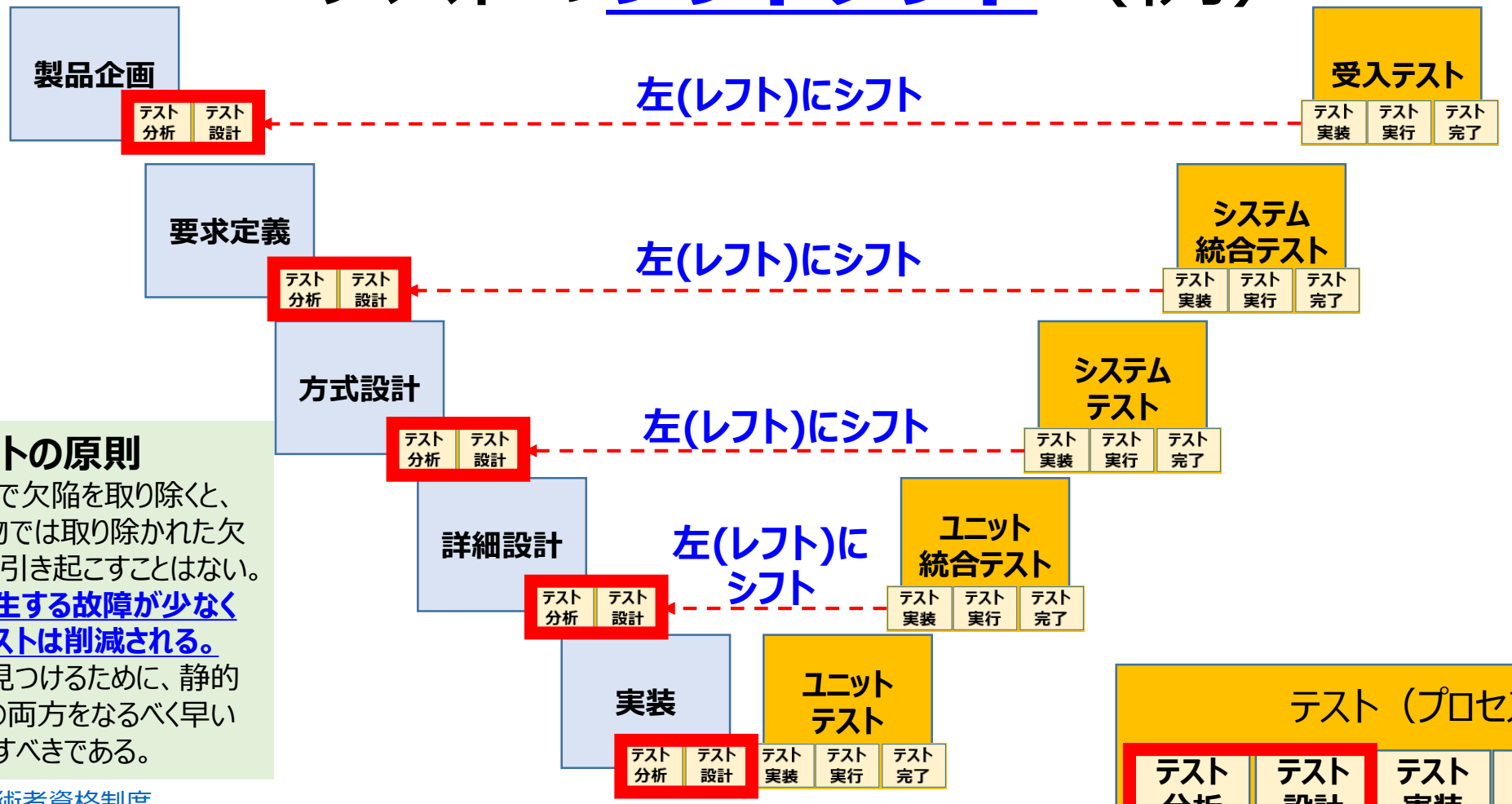
ここをすべてレビューアの頭の中だけで暗黙的に実施する／あるいは形式的・固定的なチェックリスト等で済ませている

結果的に
レビューの成果が
有識者の参加有無に
頼る状態に

結果評価もふりかえりも行わない
→レビューはいつも同じやり方に

ソフトウェアライフサイクルとレビュー

テストプロセスの前段部分を先出しする ～テストの**シフトレフト**（例）



早期テストの原則

プロセスの早い段階で欠陥を取り除くと、その後の作業成果物では取り除かれた欠陥に起因する欠陥を引き起こすことはない。

SDLCの後半に発生する故障が少なくなるため、品質コストは削減される。

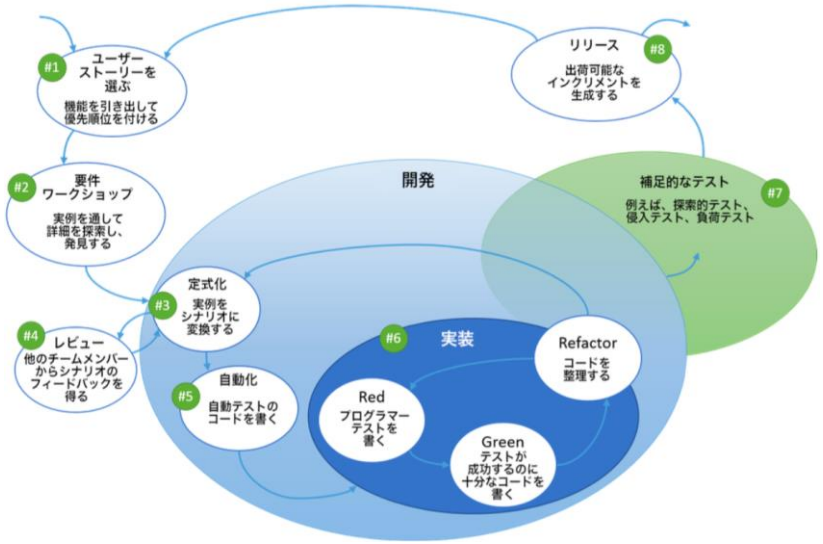
早い段階で欠陥を見つけるために、静的テストと動的テストの両方をなるべく早い時期に開始すべきである。

引用：ISTQBテスト技術者資格制度 Foundation Level シラバス 日本語版 Version 2023V4.0.J01

シフトレフトの原理を活用した テスト主導ソフトウェア開発

BDD

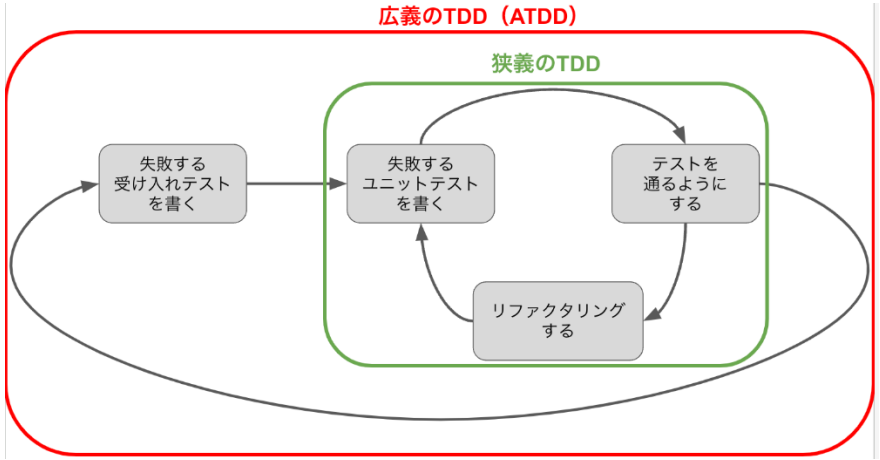
Behavior Driven Development
振る舞い駆動開発



[TDDとBDD/ATDD(4) ツールとしてのBDDとプロセスに組み込まれたBDD](#) より

ATDD

Acceptance test–driven development
受け入れテスト駆動開発



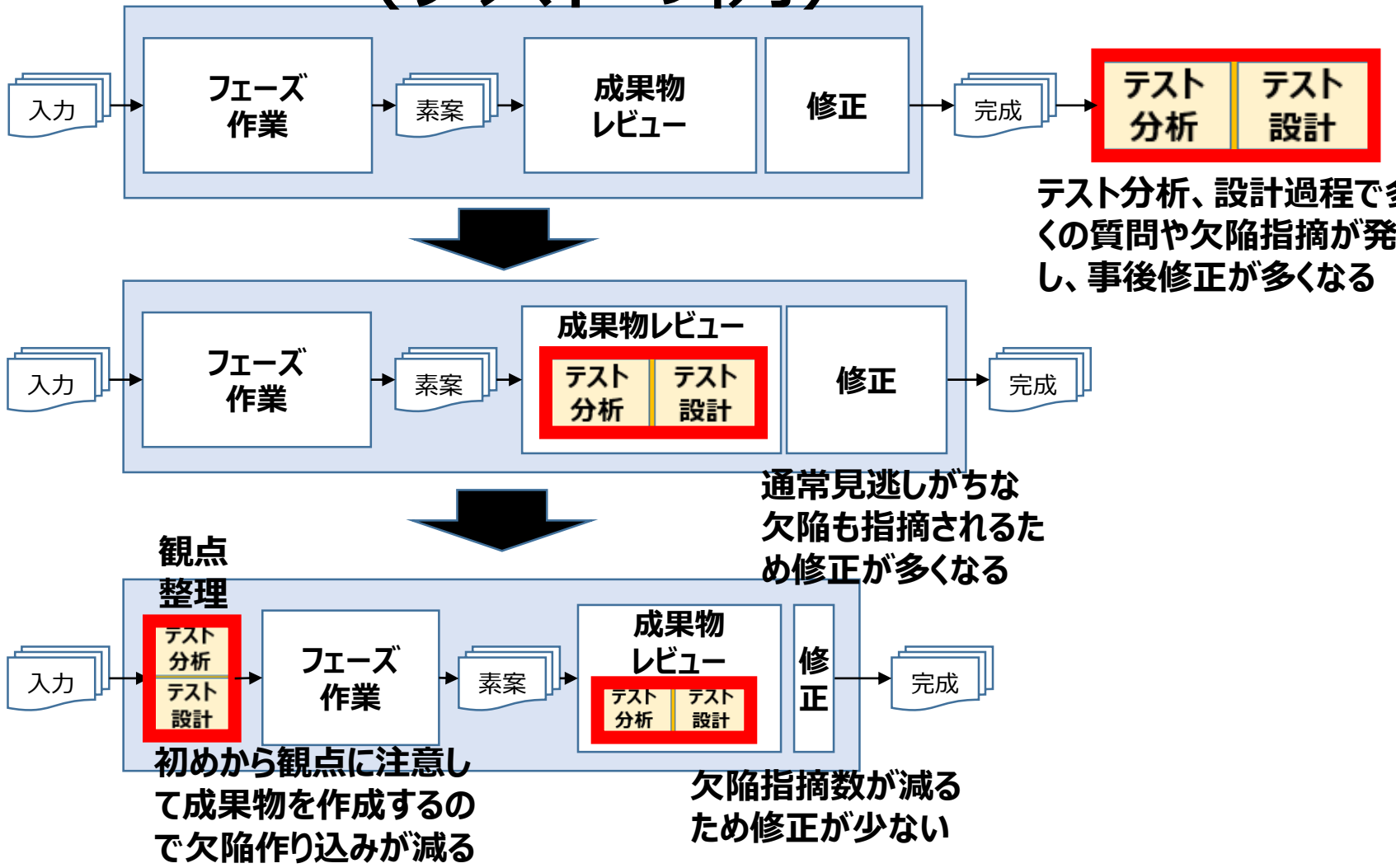
[TDDとBDD/ATDD(3) BDDとATDDとSbE](#) より

TDD

Test-Driven Development
テスト駆動開発

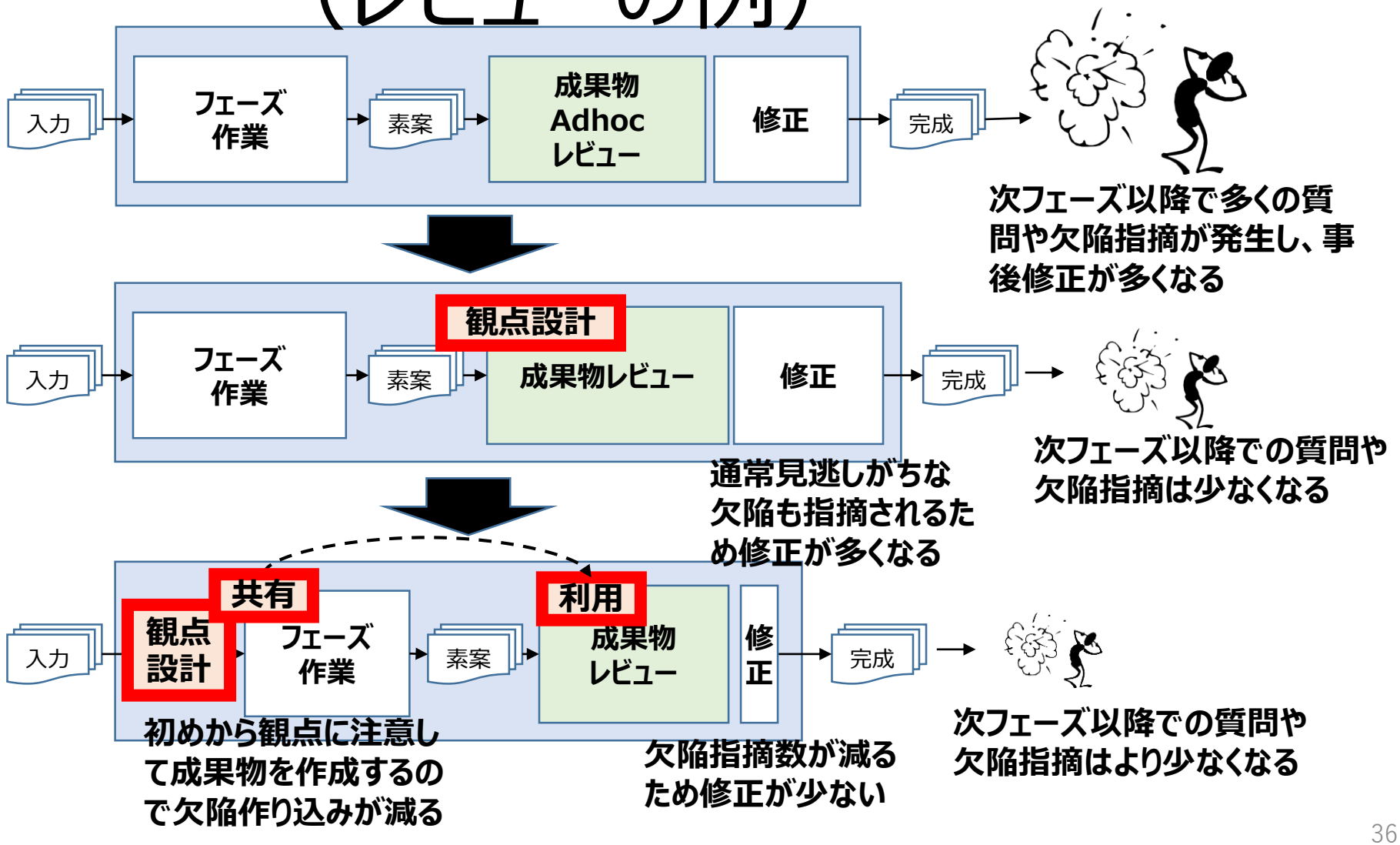
[50分でわかるテスト駆動開発 / TDD Live in 50 minutes](#) より

同一フェーズ内でのシフトレフトライクなこと（テストの例）



最初から品質を作り込む

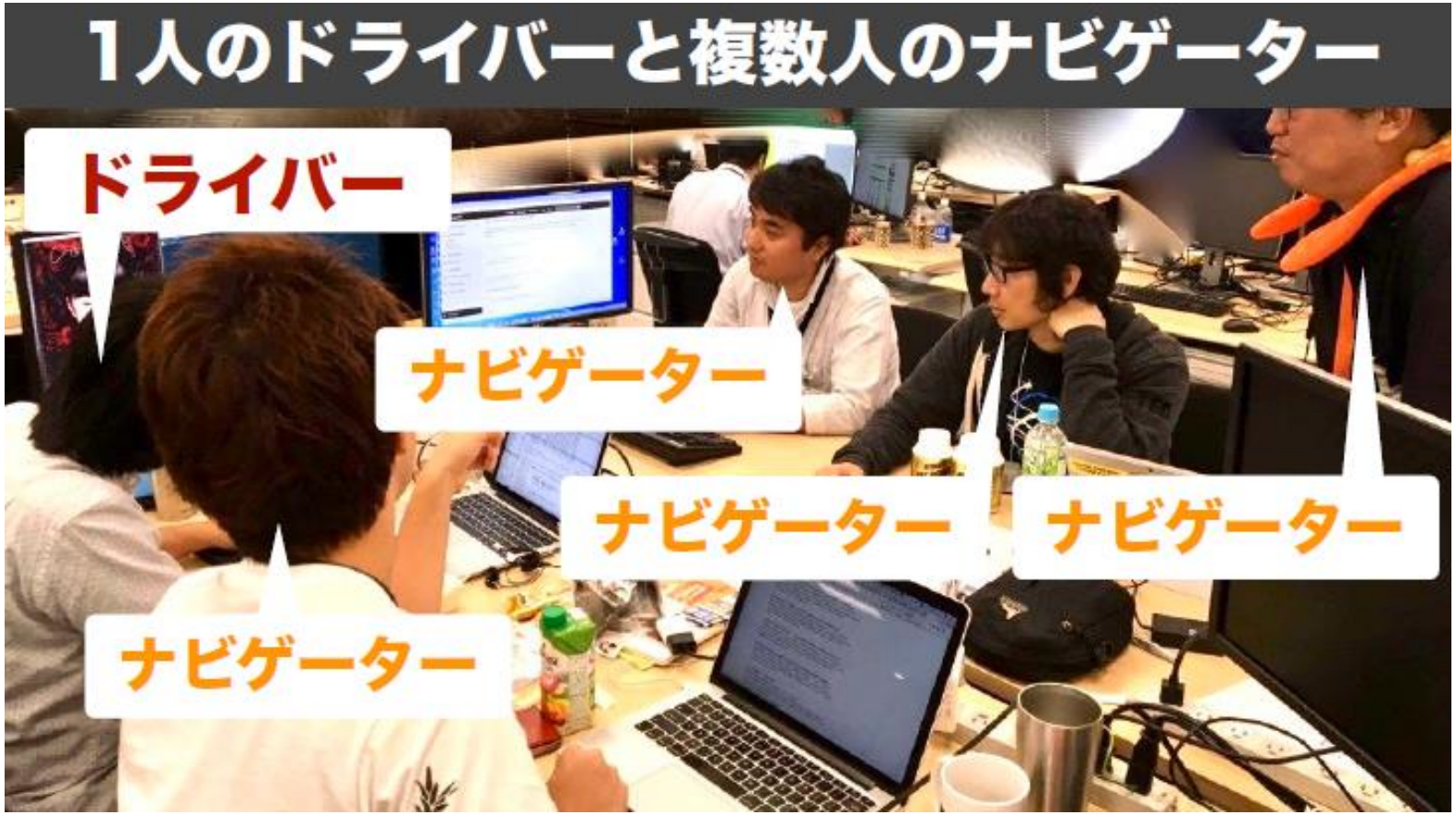
レビューにおけるシフトレフトライクなこと（レビューの例）



最初から品質を作り込む

モブプログラミング

JaSST Review2018　@TAKAKING22さん「レビュー再定義」講演資料より

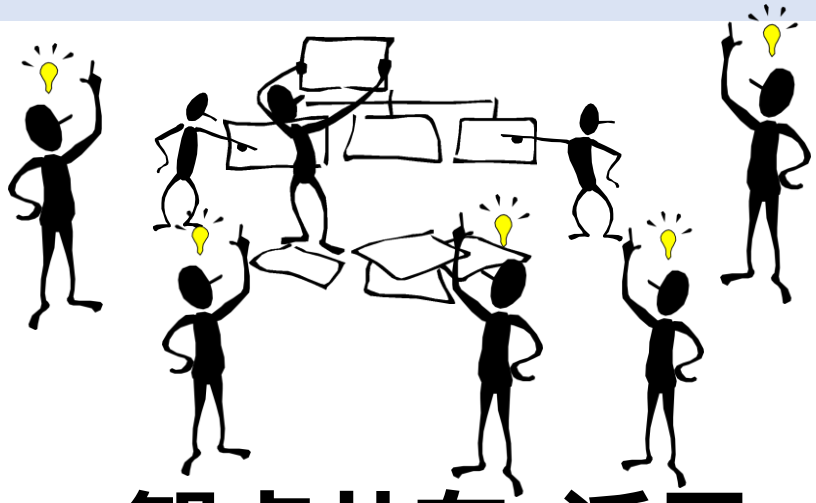


レビュー観点
レビューアーキテクチャの構成要素

われわれが目指していること

レビュー観点を活用して
「**誰が確認したか？**」から「**何を確認したか？**」
へシフトしよう！

有識者依存



観点共有・活用

【実験】レビュー観点を共有せずに Adhocにレビューすると何が起きるのか？

1	3	3-8	ビジネス目的、システム化目標など、上位の概念や目標、計画などの確認	上位の根拠と矛盾がないか、足りないことはないか、そもそも何を達成したいのか、何を参考にこの要集仕様書を作ったか、何か似たようなパクれるものはないのか	目的・価値	企画書案 要求仕様書	うれしの	
2	4	5	想定ユーザー、特にアウトプット利用者が困っていること、無駄だと思っていることの確認	経理や承認者、申請者が最近困っていることや非効率なことが解決されるのか	目的・価値	企画書案 要求仕様書	うれしの	3
3	1	5	効率的にとは具体的に何を意図しているか、どのくらいの効率を狙いたいのか	不明点の明確化	よしざわ			
4	1	5	システム化の目的には「社内の従業員が交通費を効率的に精算できる環境を提供する。」とあるが、すべて手入力であるなど効率化への配慮を欠いたシステムになっている	システム目的の実現性	あだち			
5	5	6	交通費の精算プロセスを自動化し、ヒューマンエラーを削減する。	どのようなヒューマンエラーを削減したいのか？	目的・価値	企画書案 要求仕様書	うれしの	3
6	2	6	ヒューマンエラーとはどのようなことを意図しているのか？	不明点の明確化	よしざわ			
7	2	6	システム化の目的には「交通費の精算プロセスを自動化し、ヒューマンエラーを削減する。」とあるが、エラーチェック機能もなくヒューマンエラーへの配慮を欠いたシステムになっている	システム目的の実現性	あだち			
8	6	7	紙の使用を最小限に抑え、経費節減とSDGs対策を推進する。	・経費削減はどれくらい？ ・SDGsを社内目標などに落としたものがあるのか？何を推進したいのか？	目的・価値	企画書案 要求仕様書	うれしの	3
9	7	7	紙の使用を最小限に抑え、経費節減とSDGs対策を推進する。	SDGs対策を→「SDGsを念頭に置き、○○を」「SDGs達成に向けた社内目標に従い、○○を」（SDGsは目標なり基準なりなので、続く言葉は「対策」ではおかしい今までの何を次からどうしたいのか？	誤用	企画書案 要求仕様書	うれしの	
10	3	7	システム化の目的には「紙の使用を最小限に抑え、経費節減と SDGs 対策を推進する。」とあるが、結果を印刷することが前提となっている。	システム目的の実現性	あだち			
11	8	8	精算プロセスの透明性	今までの何を次からどうしたいのか？	目的・価値	企画書案 要求仕様書	うれしの	3
12	9	8	監査可能性	今までの何を次からどうしたいのか？	目的・価値	企画書案 要求仕様書	うれしの	
13	4	8	システム化の目的には「精算プロセスの透明性と監査可能性を向上させる。」とあるが、監査ログ取得などの運用関連要件が不明のままになっている。	システム目的の実現性	あだち			

実験：実行委員4名で「交通費精算システム要求仕様書」をそれぞれ突然レビューし、記録した。

↓分析

- 指摘総数: のべ92件 (52種類)
- 重複なし指摘数：25種
- **重複した指摘：27種類**

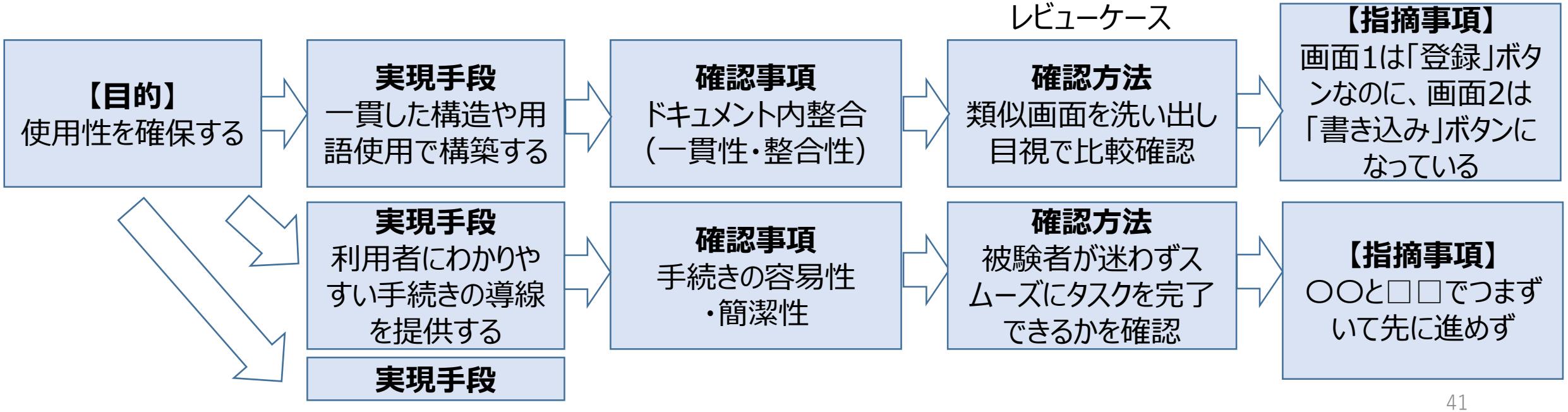
→複数人が指摘した件数：計67件

同一指摘: 67 － 27＝40件分

レビュー観点はどのようなものか？

人によって"観点"の捉え方には幅がある

レビューの意図や目的を段階的に詳細化したもの。
レビュー目的を達成するための、レビューアによるレビュー対象の見方、レビューで検出したい欠陥を見つけるためにレビューアが集中して着目する対象成果物の側面。さらに何を、どのように確認するのかを表したもの。



レビュー観点は階層構造

抽象度の高いレビュー観点（抽象的）

人によってはその意味や内訳が異なる可能性がある

【ハイレベルレビューケース：HRC】

粒度が大きい

【ミディアムレベルレビューケース：MRC】

具体的なレビュー観点

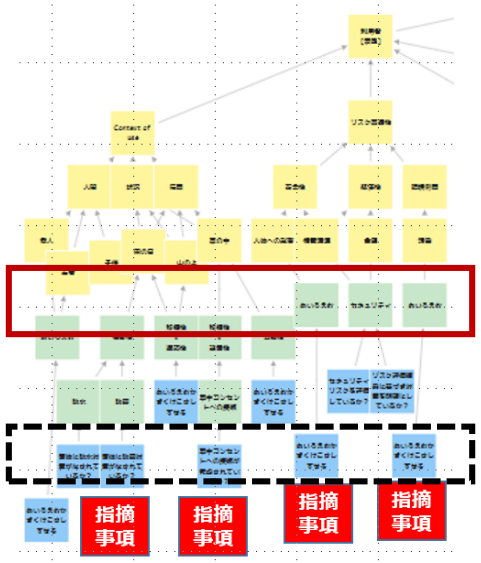
誰がでも同じ結果が導ける可能性が高い

【ローレベルレビューケース：LRC】

粒度が小さい

人によって適切に実践できる観点粒度が異なる
レビューアの所有スキルにより観点粒度を調整する

利害関係者	対象システムにおける主な活動	関心事	必要なレビュー観点（対象）	観点の具体化・詳細化
実利用者	システムに必要情報を入力して、ほしい結果を得る	誤入力情報がそのまま処理されないか？	入力ミス検知機能があるか？（入力画面）	□必須項目記入漏れの検出 □入力不可文字の検出 □日付、時間範囲外の検出



Aさん

Bさん

Aさん

OK!　この場合だと
□必須項目記入漏れの検出
□入力不可文字の検出
□日付、時間範囲外の検出
ができればよいね！

Bさん

ん～この場合だと
□必須項目記入漏れの検出
は思いつくけど他には何ができればよいのかな？？？

Bさん

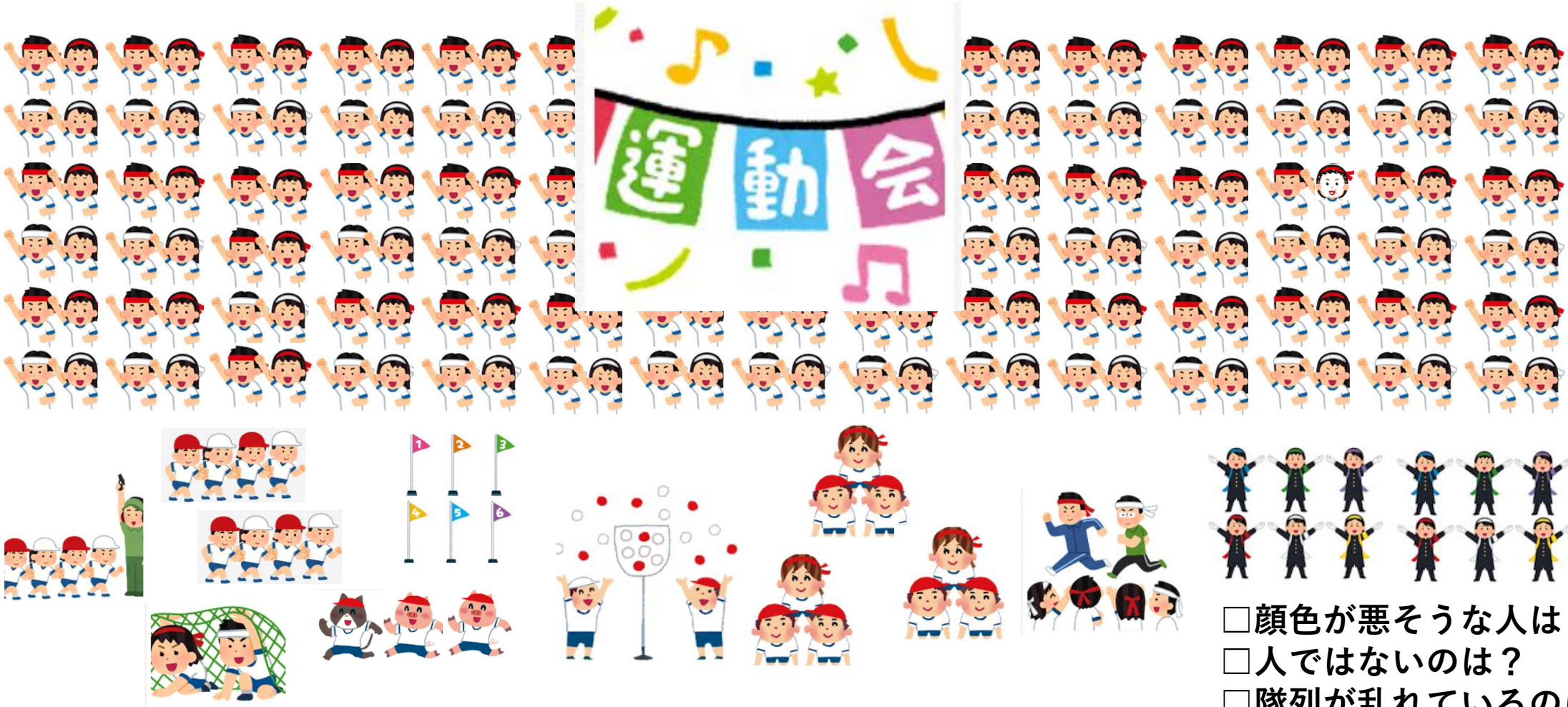
これをやればいいんだね！

なぜレビュー観点が必要なのか？

No.	内容
1	”観点”があれば雑多な情報群の中から**集中して該当を直接探し当てられやすい**
2	”観点”がない場合、**書かれていることだけに反応した指摘に終始してしまう**
3	有効な指摘事項など**優先度の高い欠陥・不備を見つけやすい**
4	”観点”がない場合、メンバー間での**重複確認**や**誰も見ていない抜け・漏れ**が発生しやすい
5	”観点”があれば、**メンバー間で確認する観点の分担が可能になる**
6	**有効指摘のノウハウを再利用しやすくなる**（**有識者依存度を低減できる**）

なぜレビュー観点が必要なのか？(1)

"観点"があれば雑多な情報群の中から**集中して該当を直接探し当てられやすい**

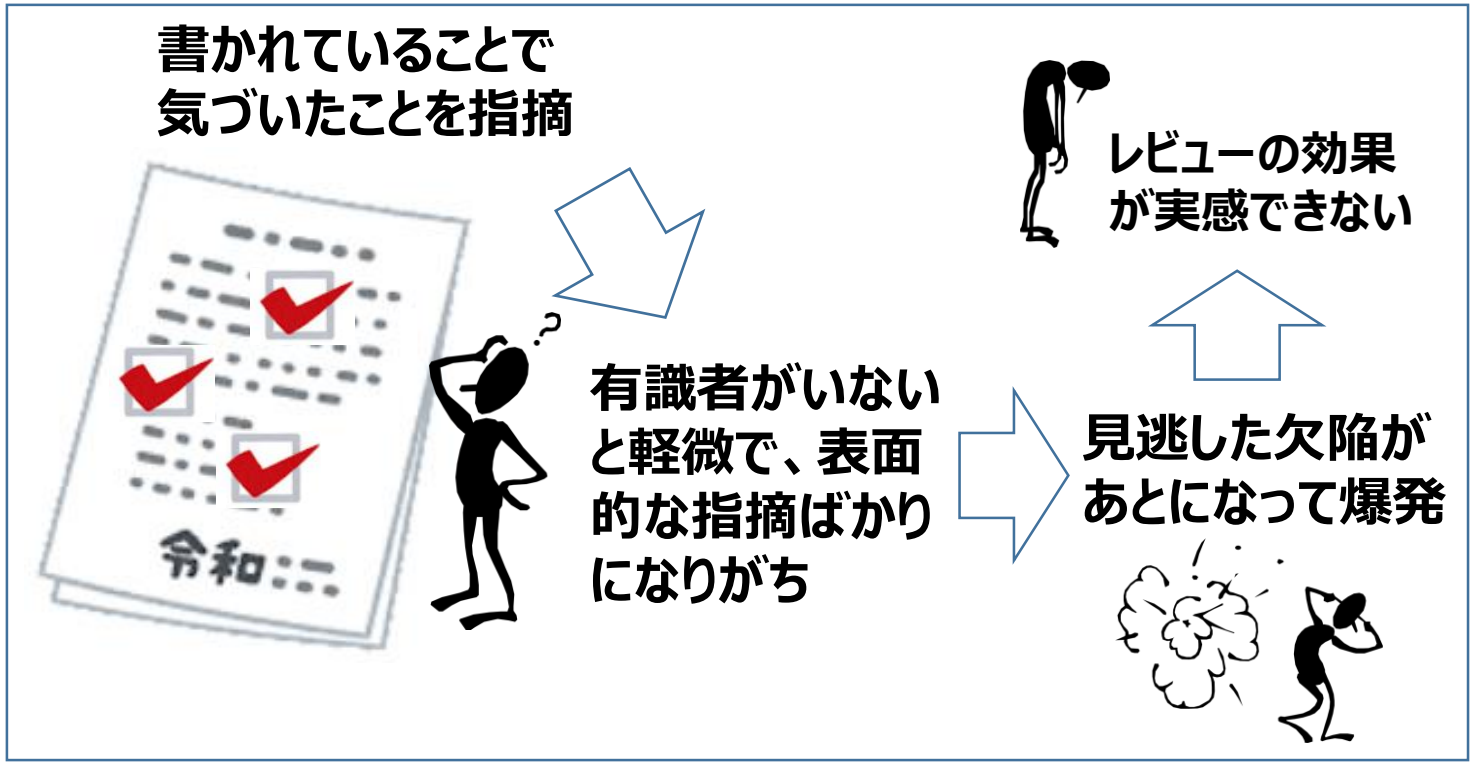


□顔色が悪そうな人は？
□人ではないのは？
□隊列が乱れているのは？

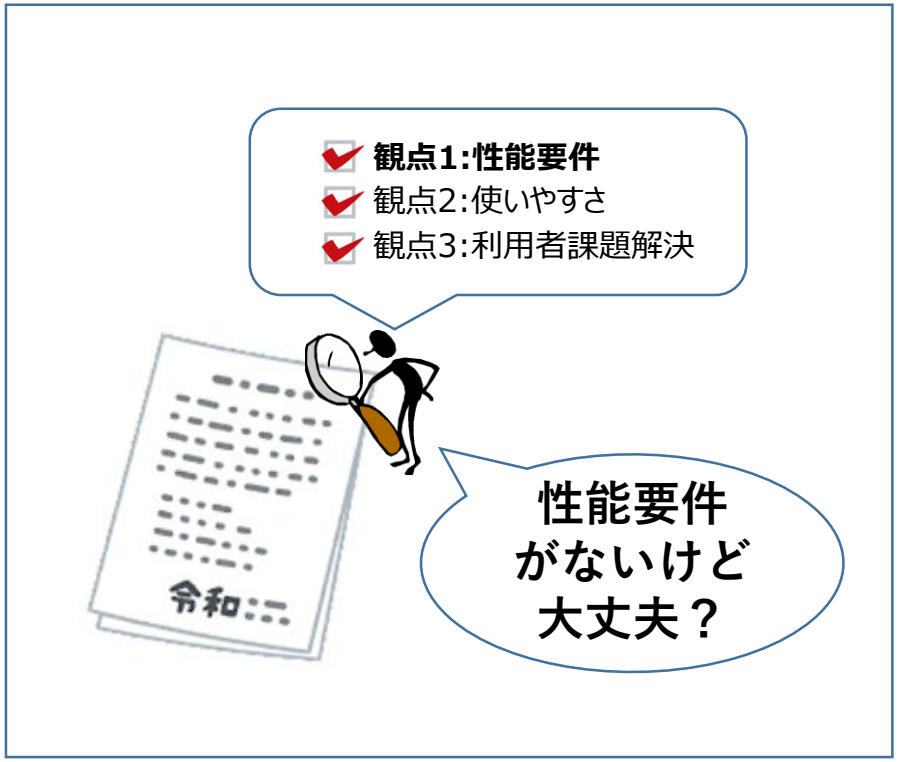
なぜレビュー観点が必要なのか？(2)

"観点"がない場合、**書かれていることだけに反応した指摘に終始してしまう**

対象を読みながらぼんやり行うレビュー



観点に基づくレビュー

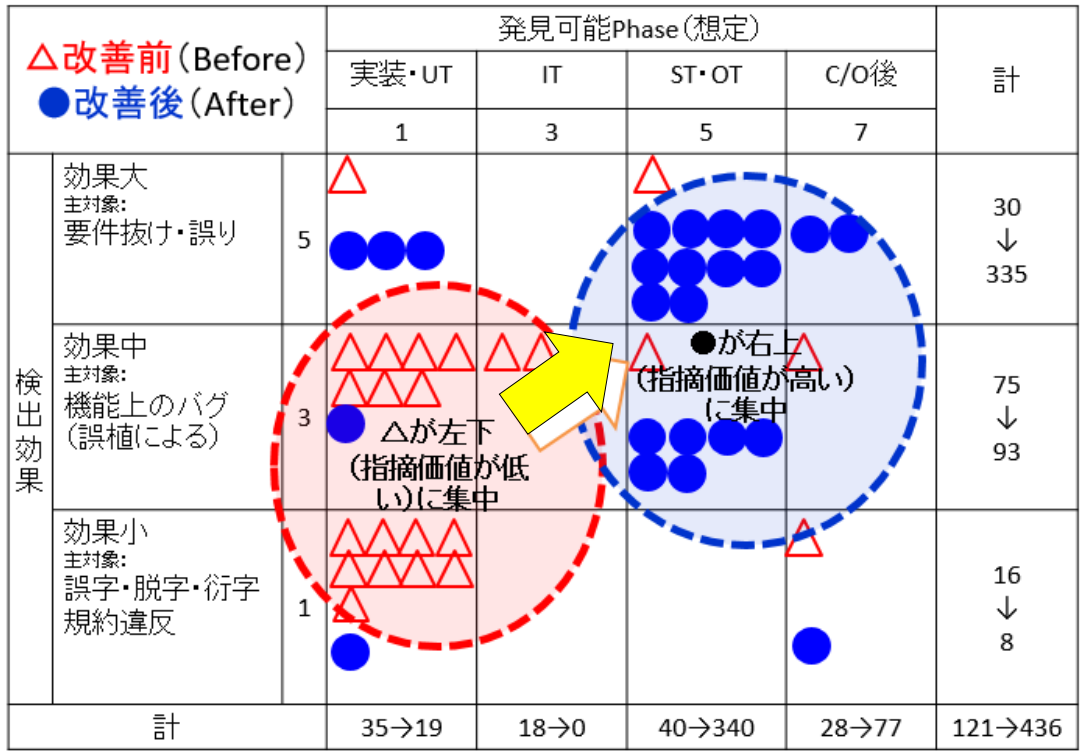


レビューに期待されていること（テストでは実現が困難なこと）

書かれるべきなのに書かれていない／書いてあるけど必要ないことを見つける

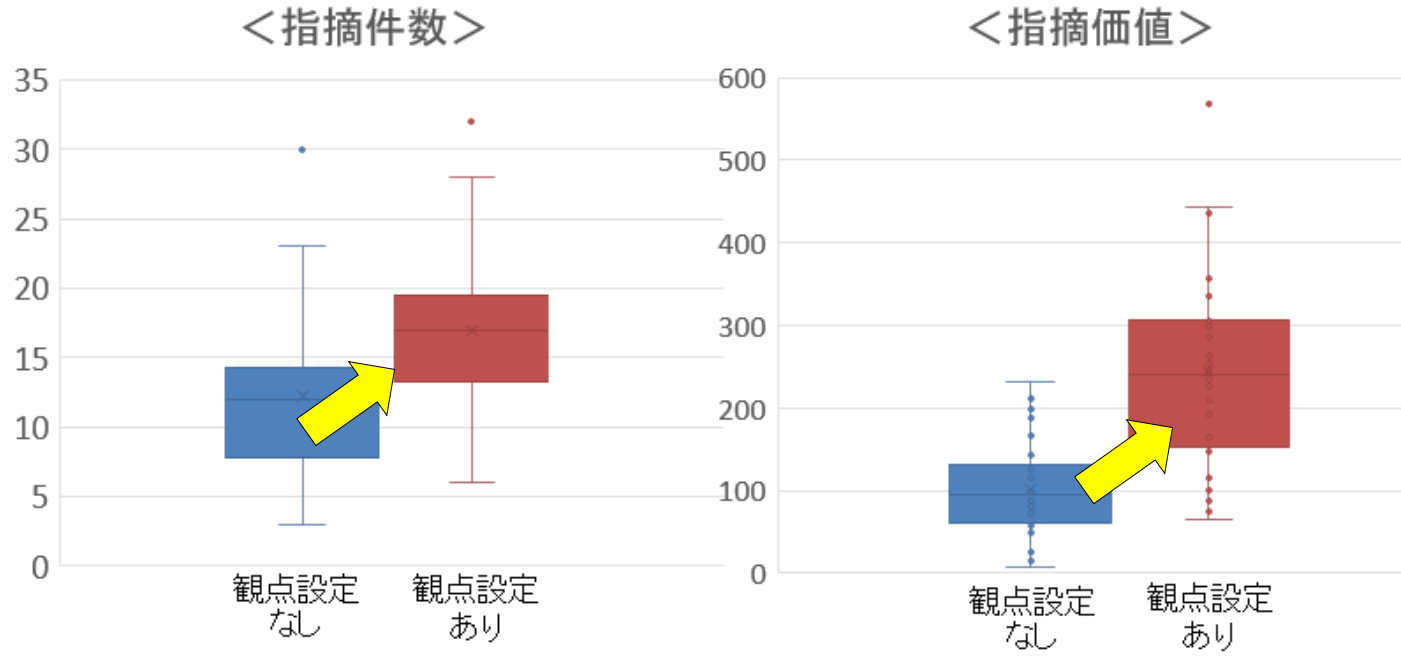
なぜレビュー観点が必要なのか？(3)

有効な指摘事項など**優先度の高い欠陥・不備を見つけやすい**



レビュー観点設計による典型的な指摘変化の例
△＝アドホックレビュー指摘
●＝観点設計レビュー指摘

2012～2020年に実施した全ワークショップ（11社30チーム）実績から算出

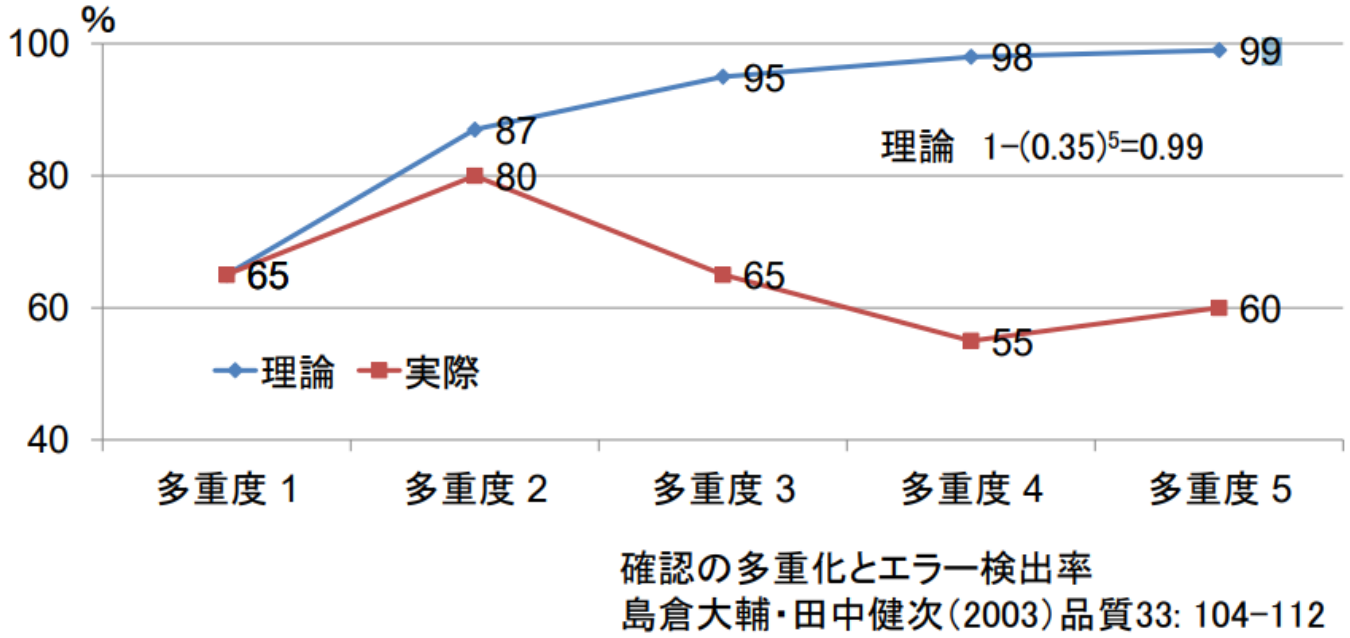


JaSST2016東京　事例発表「レビュー目的・観点設定の効果と課題」より

なぜレビュー観点が必要なのか？(4)

"観点"がない場合、メンバー間での**重複確認**や**誰も見ていない抜け・漏れ**が発生しやすい

ぼんやりレビューすると、みな上から順に読んで書かれていることに反応して指摘する
→同じ指摘が重複する（重複チェック）

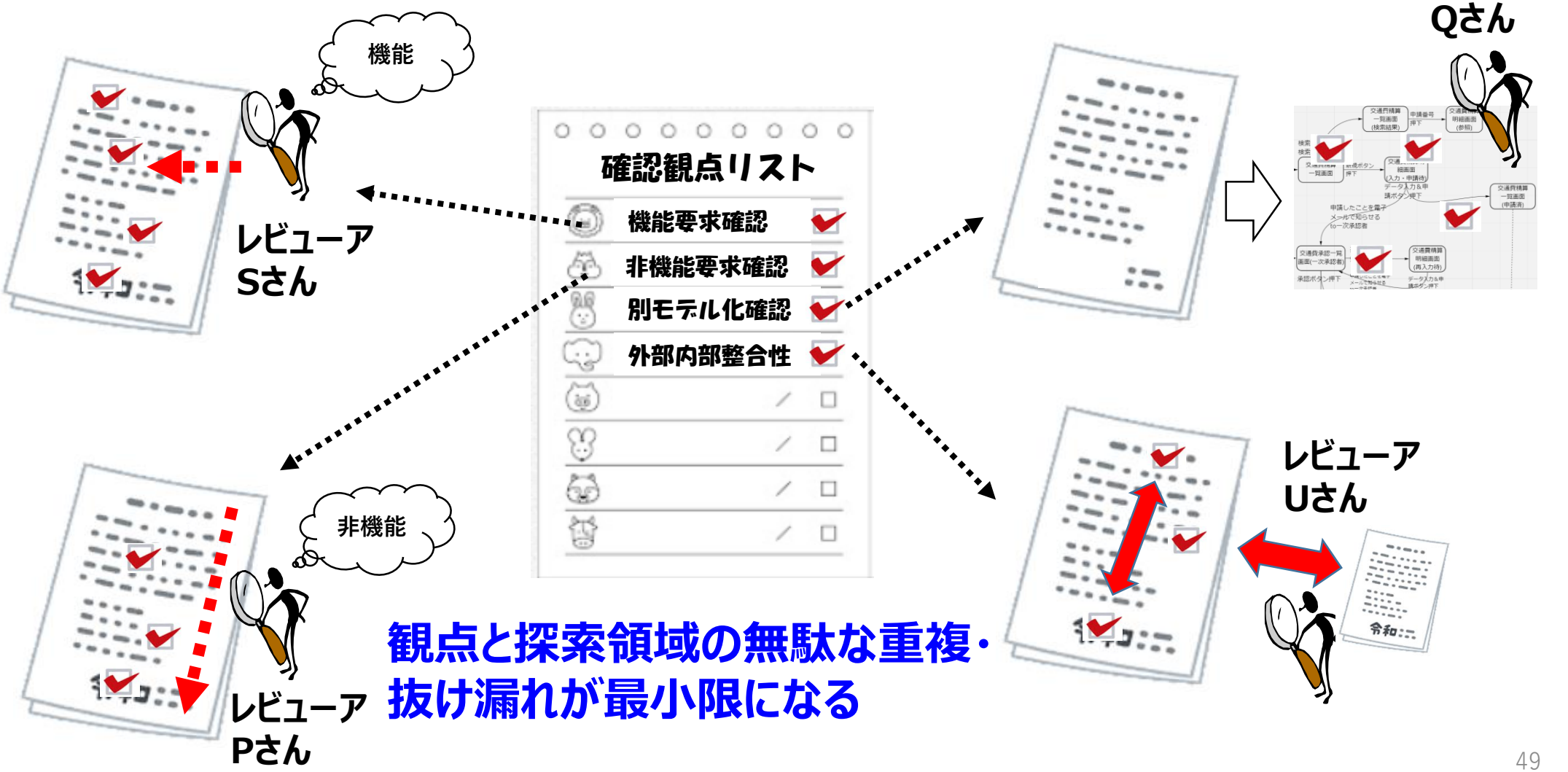


確認の多重化とエラー検出率
島倉大輔・田中健次(2003)品質33: 104−112

三重チェック以降は効果が少ない
(誰かが見るから自分はいいや　等)

なぜレビュー観点が必要なのか？(5)

"観点"があれば、**メンバー間で確認する観点の分担が可能になる**



観点と探索領域の無駄な重複・抜け漏れが最小限になる

なぜレビュー観点が必要なのか？(6)

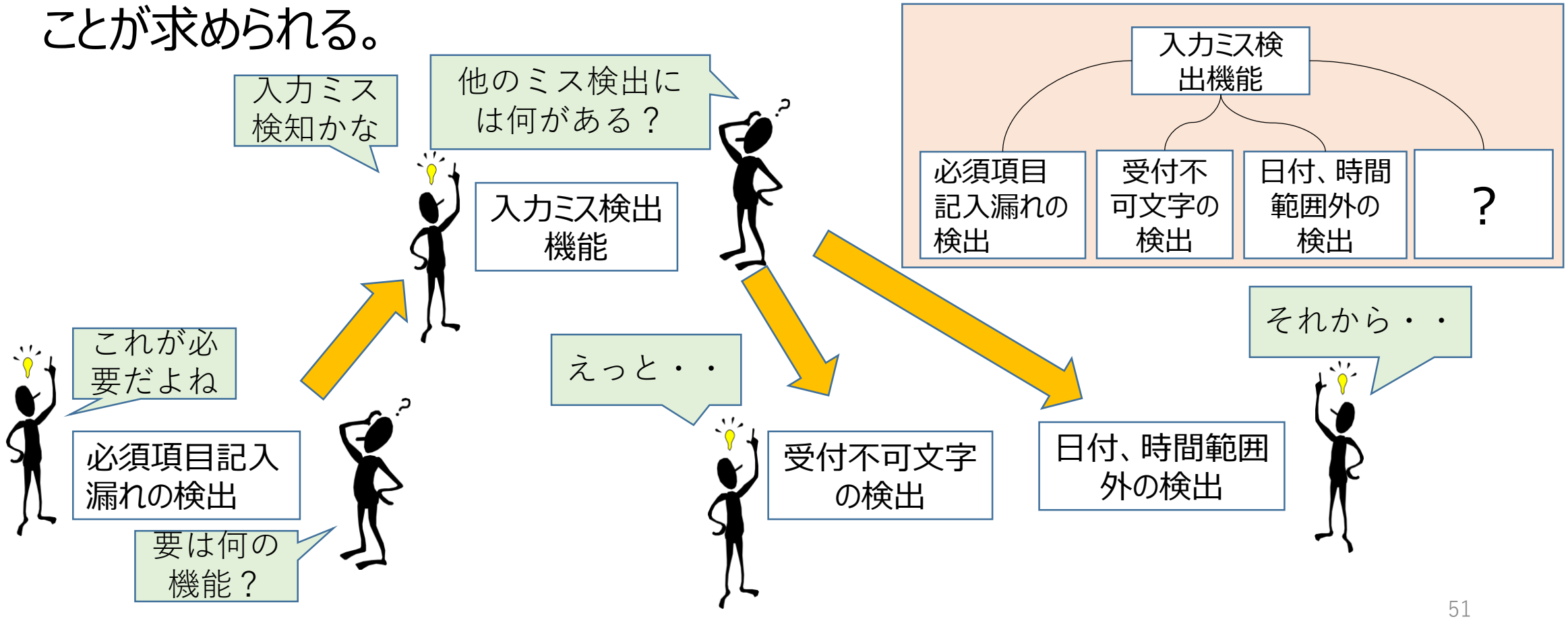
有効指摘のノウハウを再利用しやすくなる（有識者依存度を低減できる）

有効なレビュー観点をまとめて再利用することでレビューの効果を高める

目的～観点構造化は重要だが簡単ではない

構造化・モデル化にはノウハウがある

- 観点設定ではトップダウン／ボトムアップの両面アプローチとMECEであることが求められる。



適切な思考・実践方法を身につけるには継続実践が不可欠

- 観点設定→観点に基づくレビューを、**思考を伴わず形式的に行っても効果は期待できない**。
- 重要なのはレビュー対象や観点設計への理解に基づくに実践。そこには**状況に応じた「適切な思考」**と**「モデル化」**が求められる。
- その**実践力は、継続して取り組むことで身につくもの**。

  当初は時間がかかる割に効果は？？？～失敗も多い。

  [実践→ふりかえり]の継続～時短＆効果向上

レビューアーキテクチャ

レビュー観点によるレビューアーキテクチャの設計

にしさんの予告 レビューアーキテクチャ

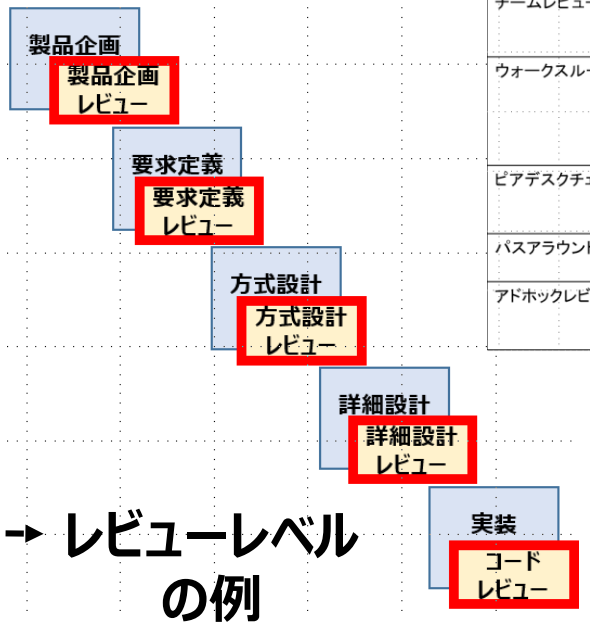
レビュー観点によるレビューアーキテクチャの設計

- どのようなタイプやレベルのレビューを行うことによって、
  どんな品質をどう保証するのか／どんなバグを検出するのか、
  の全体像を描く活動をレビューアーキテクチャと呼ぶ
  - マスターレビュープランやレビュー戦略と呼んでもよい
  - 個々のレビューのアクティビティ(≒レビュー会議)をレビューコンテナと呼ぶ
    - 例) ○○サブシステムに対する性能レビュー、○○サブシステムレビュー、性能レビュー
    - それぞれのレビューアクティビティでどのレビュー観点を分担するのか、を明示する
  - レビューコンテナ間の順序関係や依存関係を明示して、レビューの全体像を描き把握する

レビュー観点によるレビューアーキテクチャの設計

- どのようなタイプやレベルのレビューを行うことによって、
  どんな品質をどう保証するのか／どんなバグを検出するのか、
  の全体像を描く活動をレビューアーキテクチャと呼ぶ
  - レビューコンテナ内のレビュー観点、レビューコンテナで用いる技術や必要な技術レベル、
    レビューコンテナの(副次的)役割などをはっきりさせる
    - レビューレベル：レビュー対象物の種類や粒度、順序関係などによる分類
      ー、仕様レビュー、アーキテクチャレビュー、詳細設計レビュー、コードレビュー、
      ムレビュー、モジュールレビュー、派生部位レビュー、母体レビューなど
    - レビュータイプ：レビューの意図や観点、技術などによる分類
      レビュー、ウォークスルー、インスペクション、性能見積もり、セキュリティ設定確認、形式検証など
    - レビューサイクル：類似のレビューコンテナを繰り返す際の分類
      - ○○レビュー第2回、やり直しレビューなど
    - レビューの(副次的)役割は組織ごとに色々多岐に渡る
  - レビューアーキテクチャとテストアーキテクチャを合わせたV&Vアーキテクチャを用いると、
    開発工程の進捗に合わせてどのように品質が保証されていくのかが可視化できる
    - V&Vアーキテクチャが無いと、メトリクスやプロセスで「間接保証」せざるを得ず、大味になる

SQiPシンポジウム2019　併設チュートリアル3
「ソフトウェア開発における品質の作り込み ～フロントローディングの基礎～」より



レビューレベルの例
(テストレベルの対比)

レビュータイプ	概要	期待効果(欠陥除去)
インスペクション	最も厳格で、体系的、公式に実施されるレビュー。	1000行当たり16～17件という報告有り
チームレビュー	軽量化されたインスペクション。 構造化されているがインスペクションほど公式、厳格でない。	1時間当たりの欠陥検出数はインスペクションの2/3という報告有り
ウォークスルー	作成者が主導して実施されるレビュー。 作成者自らが対象成果物内容を説明することで	1000行当たり8件程度(インスペクションの半分)という報
ピアデスクチェック		陥検出する
パスアラウンド	作成者が作業成果物のコピーを複数人に配付し、複数のコメント・フィードバックを獲得する方法。	(比較的時間がある場合や分散開発で有効)
アドホックレビュー	場当たり的、思いつきで実施するレビュー。ある特定の切り口に対する意見・コメントを即興で求める場合にのみ有効。	欠陥除去にはあまり寄与しない

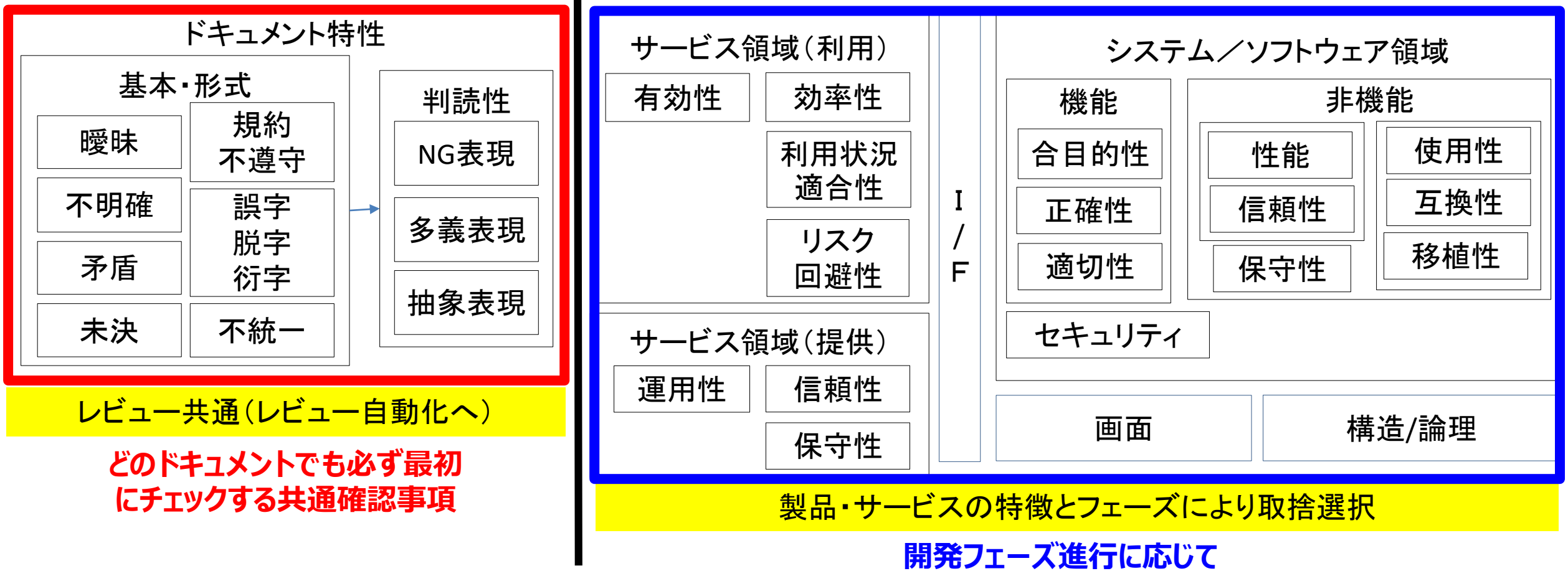
レビュー(運営)タイプの例

レビュータイプの例
(テストタイプの対比)

□誤字・脱字・衍字チェック
□曖昧・不明確・矛盾チェック
□規約・様式遵守チェック
□利用時シナリオ分析
□インターフェース分析
□合目的性確認
□リスクマネジメントレビュー
□性能見積レビュー
□セキュリティ要件実装レビュー
□内部・外部整合性確認　など

レビューアーキテクチャ検討例

「要求仕様書のレビュー技法と実践のポイント」安達2020　より



レビュー共通（レビュー自動化へ）

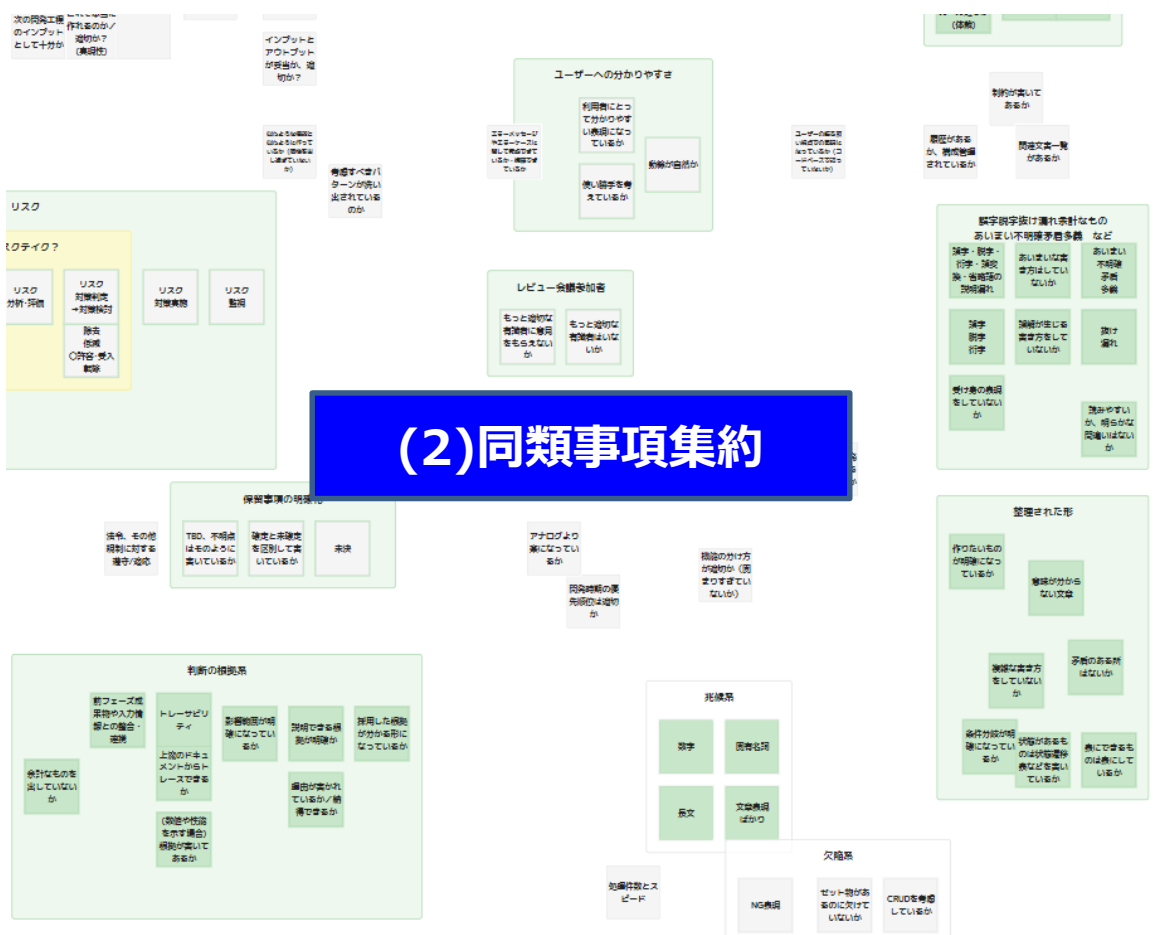
どのドキュメントでも必ず最初にチェックする共通確認事項

製品・サービスの特徴とフェーズにより取捨選択

開発フェーズ進行に応じて選択してチェックする確認事項

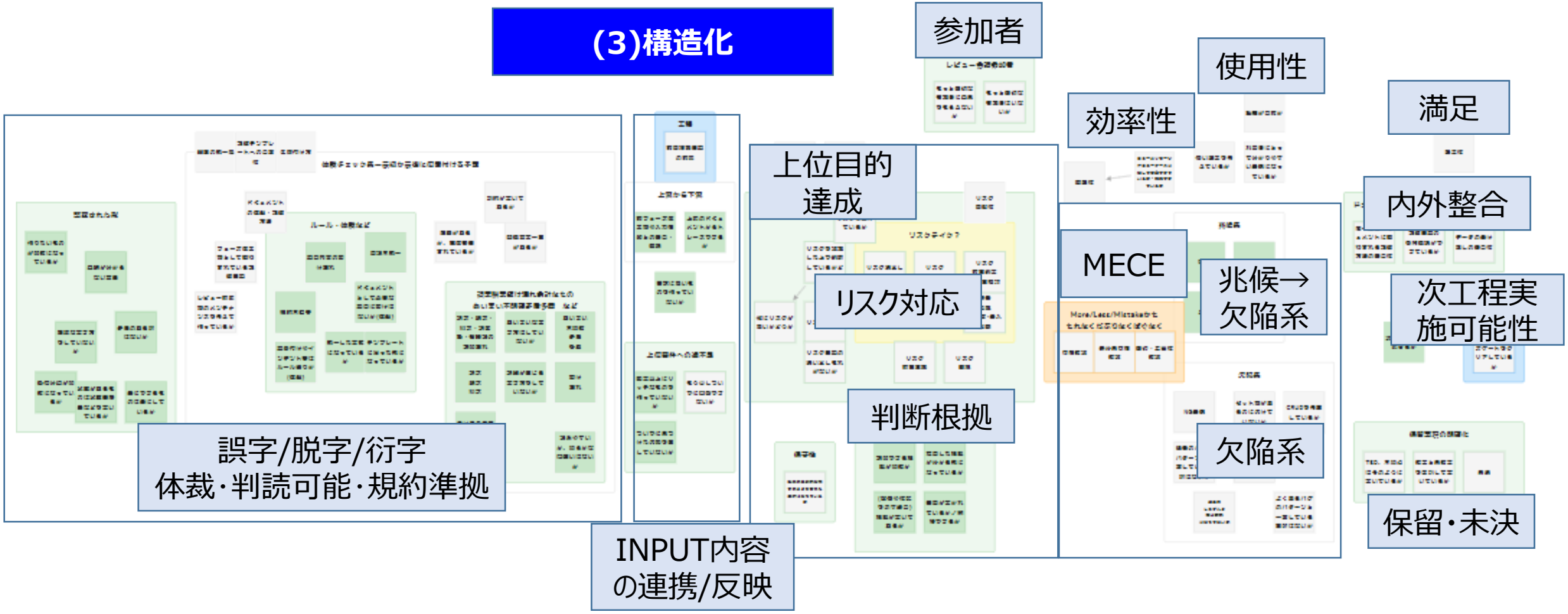
実行委員によるレビューアーキ構築試行結果

(1)観点洗い出し←→(2)同類事項集約←→(3)構造化

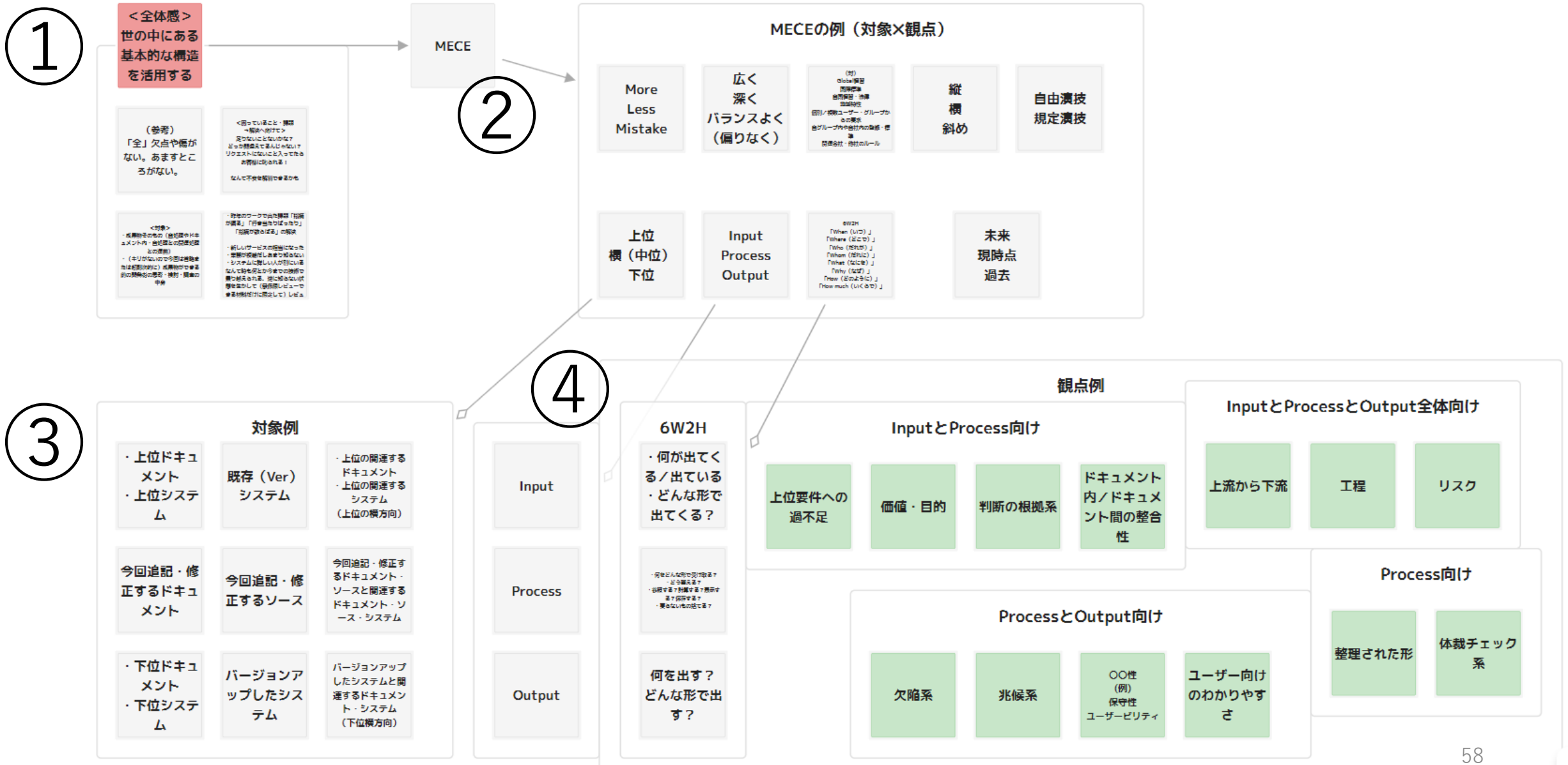


実行委員によるレビューアーキ構築試行結果
(1)観点洗い出し←→(2)同類事項集約←→**(3)構造化**

(3)構造化



レビューアーキテクチャ検討（嬉野さんの例）



レビューアーキテクチャ検討（嬉野さんの例）

①

＜全体感＞
世の中にある基本的な構造を活用する

→

（参考）
「全」欠点や傷がない。あますところがない。

→

MECE

＜対象＞
・成果物そのもの（自処理やドキュメント内・自処理との関連処理との連携）
・（キリがないので今回は省略または超副次的に）成果物ができる前の開発者の思考・検討・調査の中身

＜困っていること・課題
→解決へ向けて＞
足りないことないかな？
どっか間違えてるんじゃない？
リクエストにないこと入ってたらお客様に叱られる！

なんて不安を解消できるかも

＜メリット＞
・昨年のワークで出た課題「指摘が偏る」「行き当たりばったり」「指摘が散らばる」の解決

・新しいサービスの担当になった
・業務が複雑だしあまり知らない
・システムに詳しい人が別にいる
なんて時も何とか今までの技術で乗り越えられる、逆に知らない状態を生かして（最低限レビューできる材料だけに限定して）レビューを重ねつつ徐々に知るのもあり

レビューアーキテクチャ検討（嬉野さんの例）

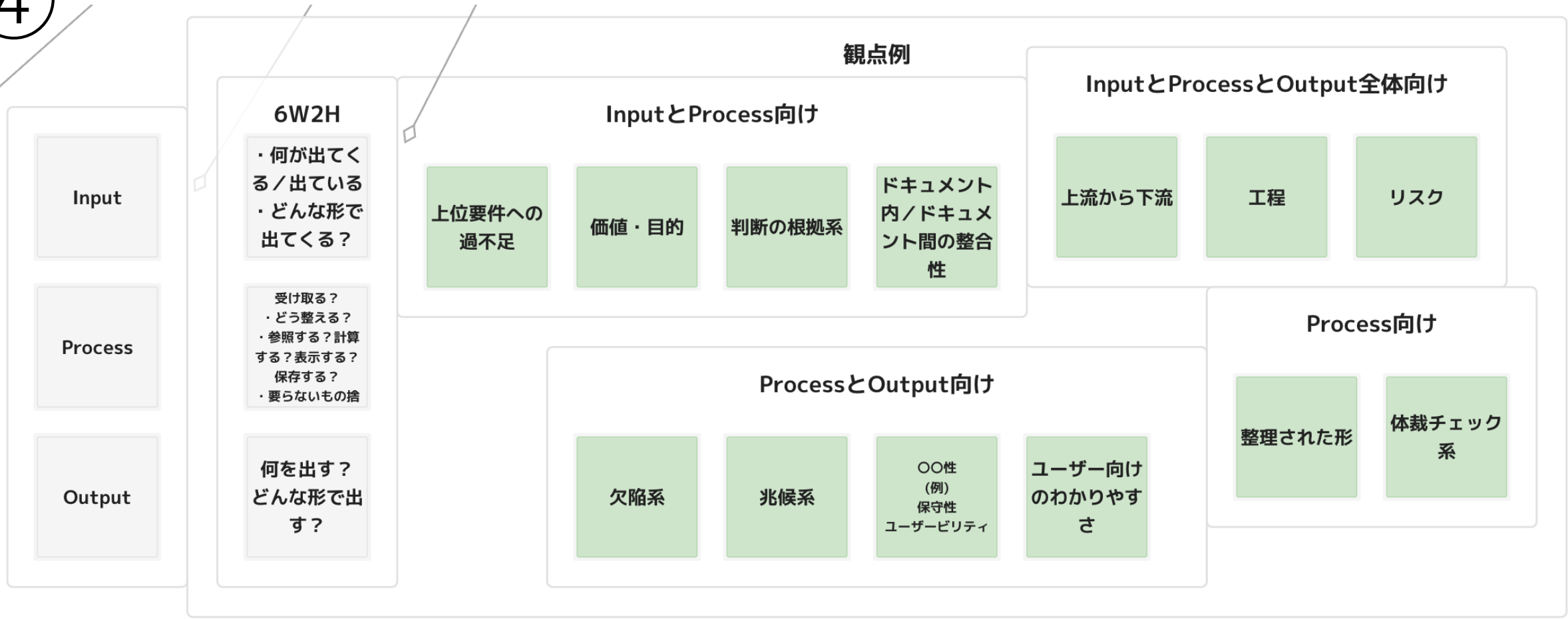
② MECE

レビューアーキテクチャ検討（嬉野さんの例）

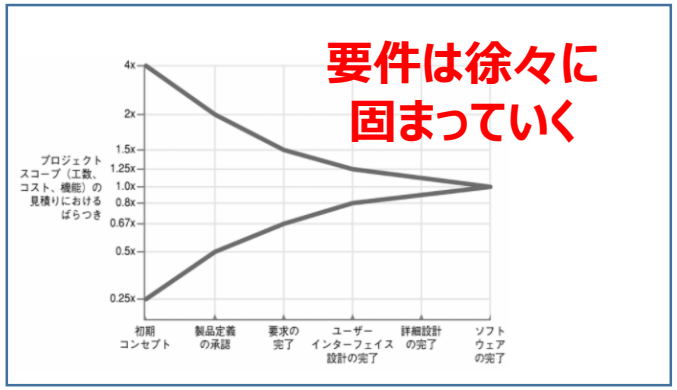
③

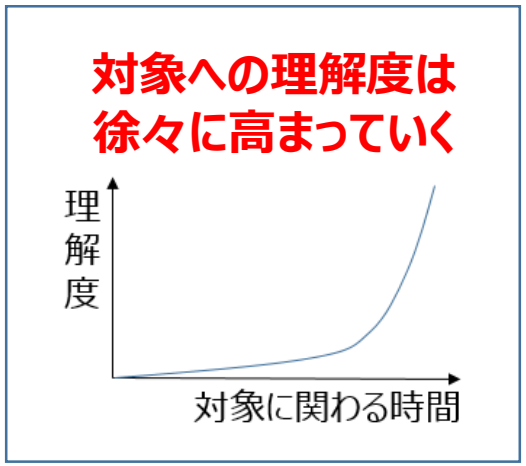
レビューアーキテクチャ検討（嬉野さんの例）

④



レビューアーキテクチャはプロダクト開発サイクルを回しながら段階的に洗練していく





次のプロジェクトへ

プロジェクト内で回しながらRAを洗練していく

次のフェーズへ

企画書
プロジェクト
コンテキスト
など

レビュー観点
導出＆アーキテクチャ構築

構想段階
粗々なレビューアーキテクチャ(RA)

レビュー
(RA利用)

レビューアーキテクチャ
利用結果

レビューアーキテクチャの
更新(洗練)

更新(洗練)済
レビューアーキテクチャ

ふりかえり

レビュー
対象成果物

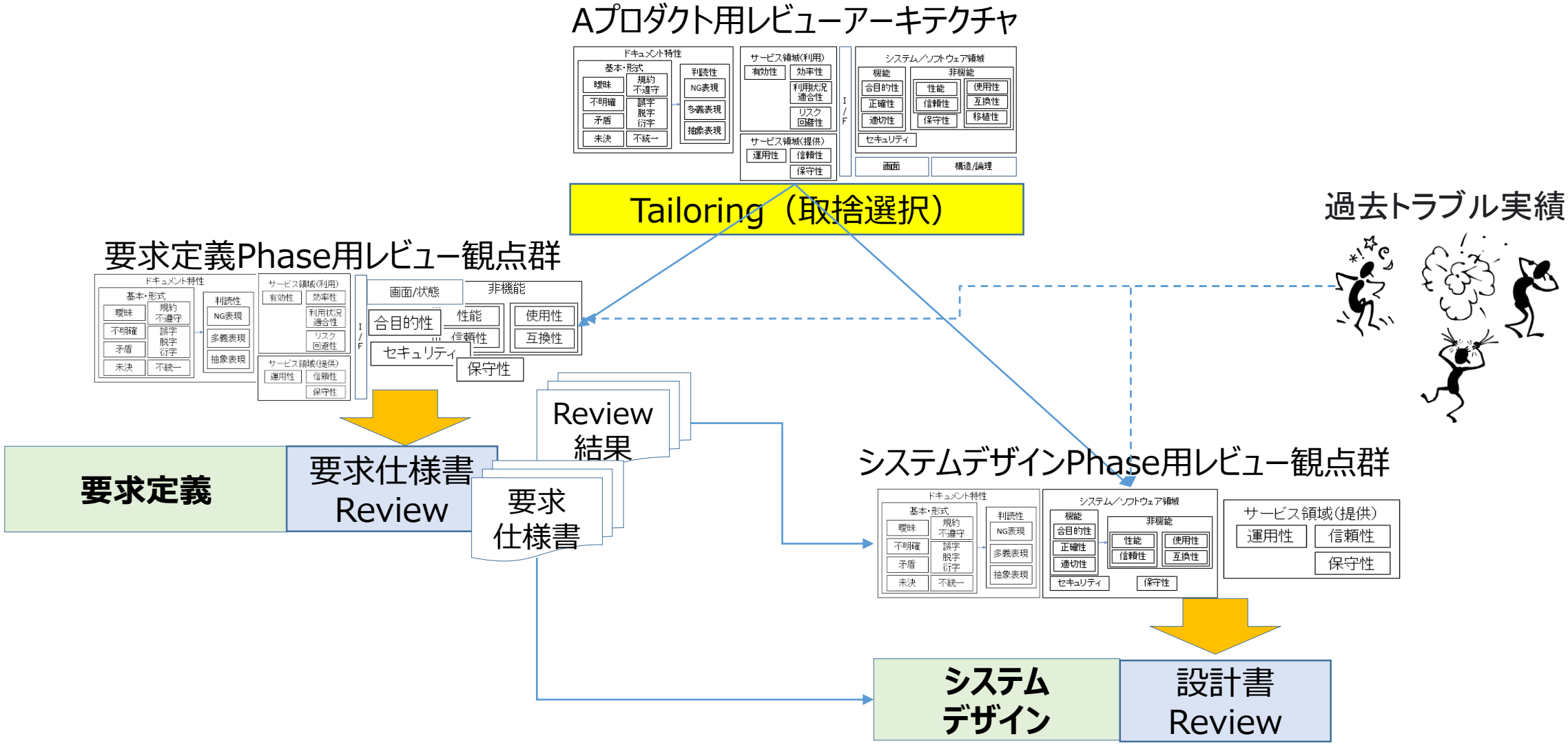
1stPhase：要求仕様書
2ndPhese：基本設計書
3rdPhase：詳細仕様書
4thPhese：コード

レビュー
結果

レビュー
対象成果物の
修正

更新済
レビュー
対象成果物

レビューアークテクチャの利用方法（例）



レビュー体系化～このあと

レビュー体系化のこの先（予定・想定・妄想等）

□**レビューとは何か？どのようなものか？の探索（われわれの原点～継続検討）**

□体系化SCOPEの明確化
例：モブワーク、プロジェクト計画レビュー、テスト仕様レビュー、テスト結果レビュー、フェーズ/プロジェクトふりかえり等は対象？

□用語定義の作成
例：レビューレベル、レビュータイプ、レビュー設計、レビュー設計技法、レビューベース、レビューウェア、レビューケース（ハイレベル・ミディアムレベル・ローレベル）、レビュータイプ、レビュー運営タイプ　など

□レビュー設計技法の構築／整理
これまでに提案されてきたレビュー観点導出アプローチの抽出・技法化、リーディング技法の整理／すみ分け／マッピング

□レビューマネジメントの構成要素分解と実践事項の明確化

□レビューファシリテーションの構成要素分解と実践力習得方法の明確化

□レビューアーキテクチャの深掘り

レビュー体系化

□各種実践事例構築／収集／共有

□体系化結果～成果物構築・公開（レビューシラバス化？）

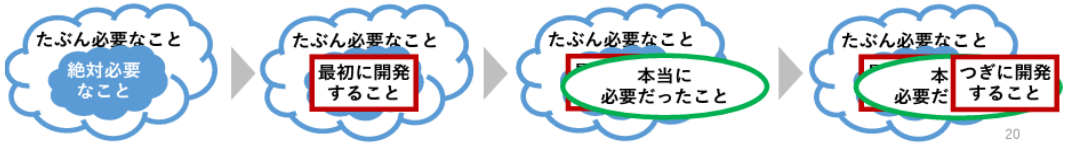
□レビュー実践力向上の場の提供
レビュー設計トレーニング／各種ワークショップ／コンテスト　等

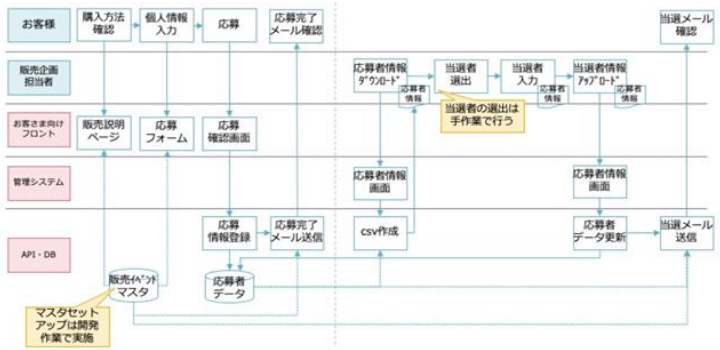
レビュー体系が使えるモノにするための活動

レビューとは何か？どのようなものか？の探索

JaSST Review2021 スクラム開発におけるPOの振る舞い

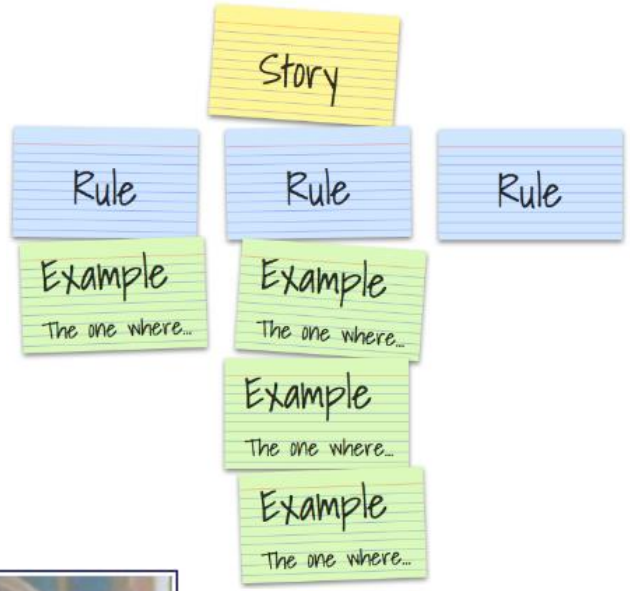
鈴木 祥子氏





河村明彦氏

「三越伊勢丹におけるデジタルサービスのつくりかた」



Matt Wynne氏

JaSST Review2022 実例マッピング

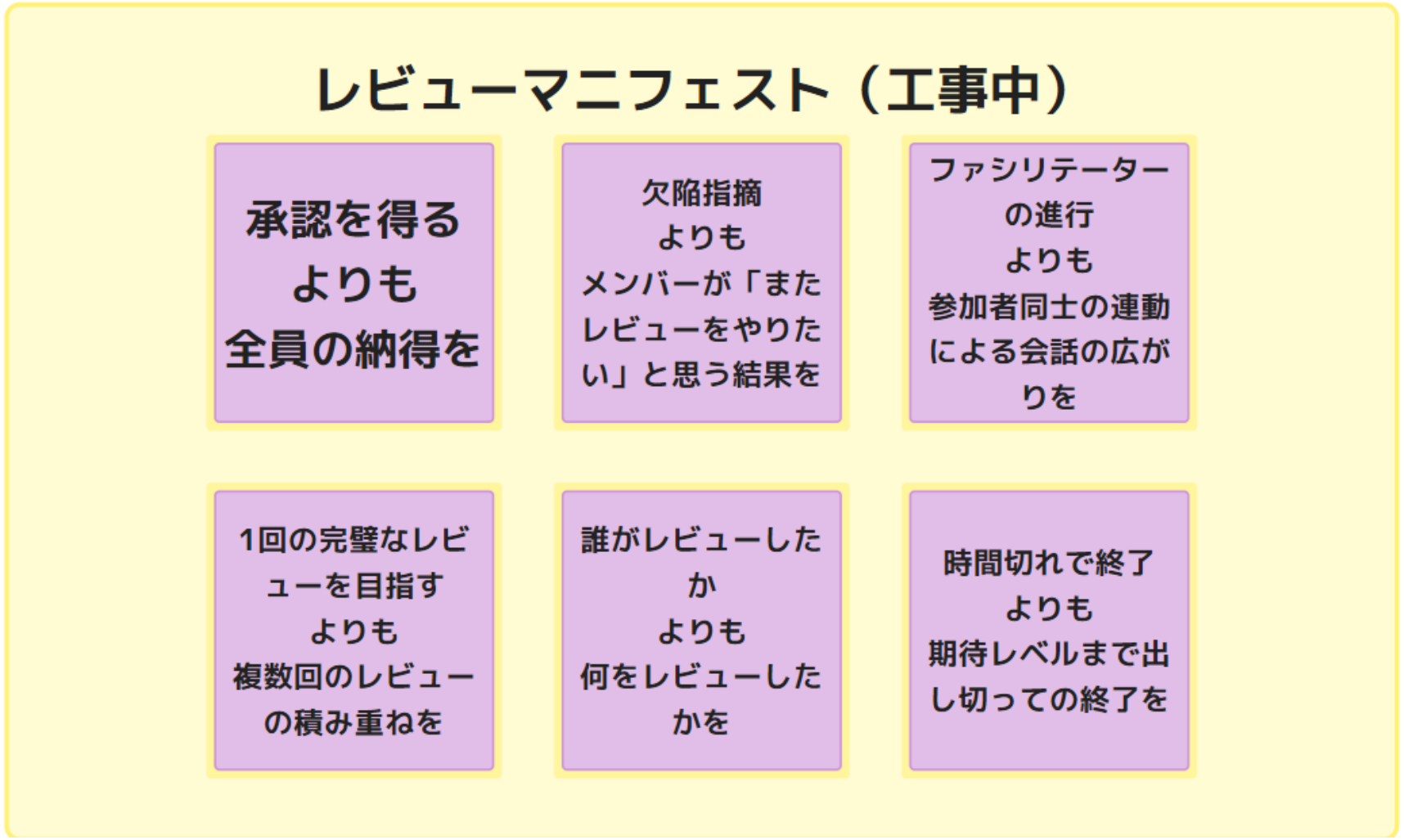
「うまくコラボレーションするためのヒミツ」

JaSST Review2023 チーム全体でテスト･品質に向かうアプローチ

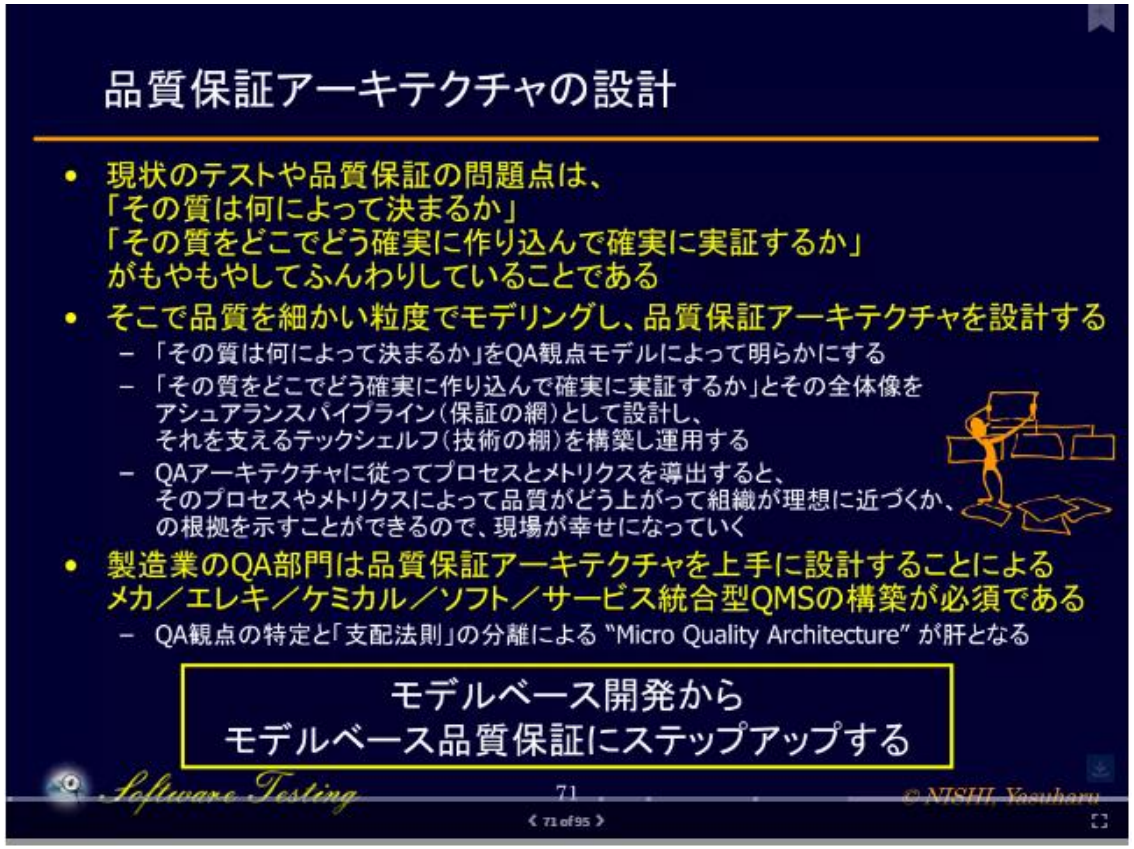
Lisi Hocke 氏

「チーム一丸となったテストと品質のためのチーム変革戦術」

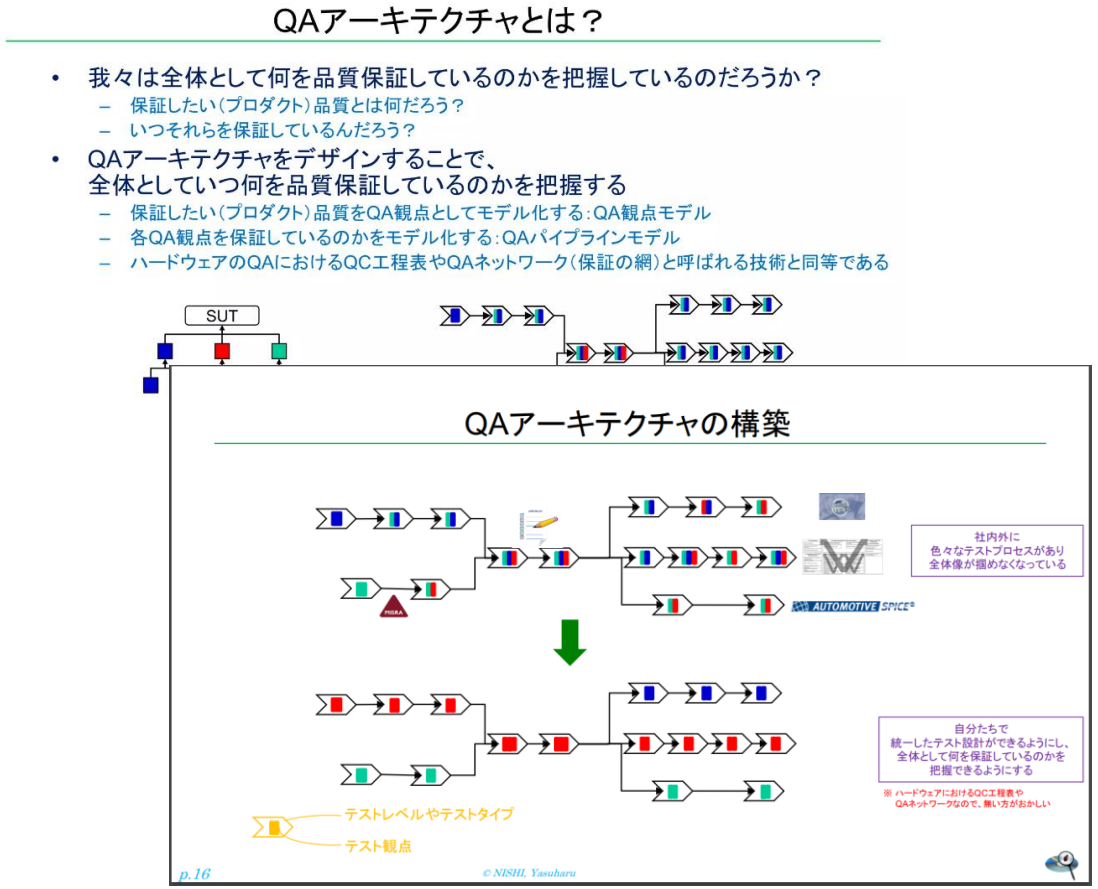
レビューマニフェスト（工事中）



にしさんの予告　QAアーキテクチャ
レビューアーキテクチャ＋テストアーキテクチャの統合？



[QAアーキテクチャの設計による説明責任の高いテスト・品質保証（2016）]



[車載ソフトウェアの品質保証のこれから(2020/8/28)]

参考文献

- ソフトウェアシンポジウム2022　経験論文
  「ソフトウェアレビュー研究結果の認知拡大と適用促進」
  https://www.sea.jp/ss2022/download/7-SS2022.pdf
- ソフトウェアレビューシンポジウム2022東京　ワークショップ 「そゆことね！よくわかるレビューテクニック～明日から使える技術をSQiPレビュー研究会からあなたに～」
  https://www.jasst.jp/symposium/jasst22tokyo/pdf/F6.pdf
- ISTQBテスト技術者資格制度　Foundation Level シラバス 日本語版 Version 2023V4.0.J01
  https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J01.pdf
- テスト設計チュートリアル テスコン編資料(講義編)
  https://www.aster.or.jp/testcontest/doc/2023_tescon_V1.0.0.pdf
- ソフトウェアレビューシンポジウム2022　ワークショップ 「レビューでは何を確認するといいのかな？さまざまなレビューの意図を整理して確認事項を明らかにしてみよう！」
  https://www.jasst.jp/symposium/jasstreview22/pdf/S4-2.pdf
- 要求工学：第3回要求仕様
  https://www.bcm.co.jp/site/2004/2004Dec/04-youkyuu-kougaku-12/04-youkyuu-kougaku-12.htm

参考文献

- 50分でわかるテスト駆動開発 / TDD Live in 50 minutes
  https://speakerdeck.com/twada/tdd-live-in-50-minutes?slide=9
- TDDとBDD/ATDD(3)　BDDとATDDとSbE
  https://sqripts.com/2023/08/07/61460/
- TDDとBDD/ATDD(4) ツールとしてのBDDとプロセスに組み込まれたBDD
  https://sqripts.com/2023/08/28/61494/
- ソフトウェアレビューシンポジウム2018　講演 「レビュー再定義」
  https://www.jasst.jp/symposium/jasstreview18/pdf/S2.pdf
- JaSST2016東京　事例発表 「レビュー目的・観点設定の効果と課題」
  https://www.jasst.jp/symposium/jasst16tokyo/pdf/A2-1.pdf
- 島倉大輔・田中健次（2003）：人間による防護の多重化の有効性、「品質」、33、〔3〕、104-112。
- ソフトウェア開発における品質の作り込み ～フロントローディングの基礎～
  https://www.slideshare.net/YasuharuNishi/software-frontloading-and-qa

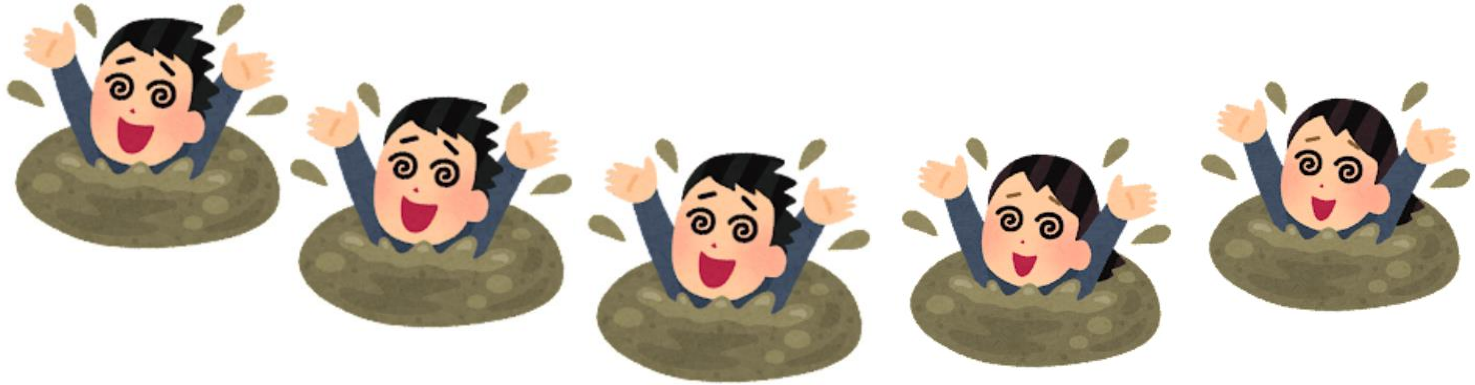
参考文献

- ソフトウェア品質知識体系ガイド （第3版） －SQuBOK Guide V3－
- ソフトウェアレビューシンポジウム2021　講演 「三越伊勢丹におけるデジタルサービスのつくりかた」

  https://www.jasst.jp/symposium/jasstreview21/pdf/S1.pdf
- ソフトウェアレビューシンポジウム2022　講演 「The secrets of effective collaboration 「うまくコラボレーションするためのヒミツ」」
- QAアーキテクチャの設計による説明責任の高いテスト・品質保証

  https://www.slideshare.net/YasuharuNishi/qa-67559281
- 車載ソフトウェアの品質保証のこれから

  https://www.slideshare.net/YasuharuNishi/ambidexterity-of-automotive-software-qa



このあともレビューという**深い沼**を探索し続けます
役立つ成果物が出せるのか・・・乞うご期待