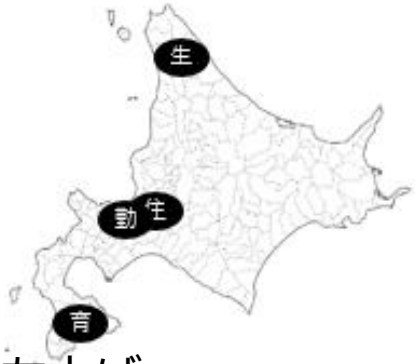
レビュー/テストからのフィードバックにどう立ち向かうのがよいか？

～レビュー観点活用で開発が変わるかも

Software Review Engineering Explorers

(SReEE)安達 賢二

安達 賢二（あだち けんじ） adachi@hba.co.jp
株式会社HBA　経営企画本部　Exective　Expert（理事）
　　イノベーション推進室　アドバイザー
http://www.softwarequasol.com/
株式会社Levii　共創ファシリテーター
https://levii.co.jp/about/



【経歴】
2012年社内イントレプレナー第一号事業者として品質向上支援事業を立ち上げ。
自律運営チーム構築・変革メソッドSaPIDをベースに、
関係者と一緒に価値あるコトを創る共創ファシリテーター／
自律組織・人財育成コーチとして活動中。

【社外活動】
NPO法人 ソフトウェアテスト技術振興協会（ASTER）理事
JSTQB（テスト技術者資格認定）技術委員
JaSST（ソフトウェアテストシンポジウム）北海道
　2006-2009実行委員長　2010-2018実行委員　2019～2022サポーター
JaSST-Review（ソフトウェアレビューシンポジウム）実行委員
JaSST-nanoお世話係

ソフトウェアレビューをエンジニアリングっぽく捉える会
**　：Software Review Engineering Explorers／SReEE(スリー)メンバー**

テスト設計コンテスト本部審査委員(2015-2017)
SEA（ソフトウェア技術者協会）幹事・北海道支部メンバー
SS（ソフトウェア・シンポジウム）プログラム委員
第33-40期SQiP研究会レビュー分科会アドバイザー
SQuBOK_Ver3プロセス改善研究Grリーダ（with プロセス改善の黒歴史研究）
TEF北海道お世話係／TOCfE北海道幽霊メンバー　など

発表概要

- 多忙で限られた時間の中で**一生懸命に作り上げた開発成果物**なのに**レビュー、テストで容赦なく突き返される**・・・どう立ち向かうのが良いですか？
- **単に「突き返される→手直し」を繰り返しても解決しません**よね。
- 積み重ねられたレビュー結果やバグ票をすべて読み解いて開発に活かす・・・よく聞く話ですが実際に成功した事例はあまり聞きませんし、本当に多忙な中でできることなのでしょうか？
- その解決策として**今回は「レビュー観点を活用した開発」を提案**し、そのカラクリや想定される効果（Quality of Engineer Lifeがどう変化するかを含む）を共有します。

コンテンツ

- よくある開発の状態
- Quality・Speed・Costをマルっと変えるために
- 戦略1：欠陥混入予防
- 戦略2：問題早期発見・解決
- [戦略1×戦略2]の実践スタイルと位置づけ
- この発表の意味と価値
- この提案の実践に必要なこと
- おわりに

よくある開発の状態

プロジェクトチームの実態

☑**ベストメンバーは揃わない**

失敗を許容し、メンバーの特徴を活かして、協調・成長しながら進める

☑**厳しい制約条件**

いろんなワガママに付き合う～**良いモノを、早く、安く、安全に**

☑**チームはどこかで壁にぶつかる**

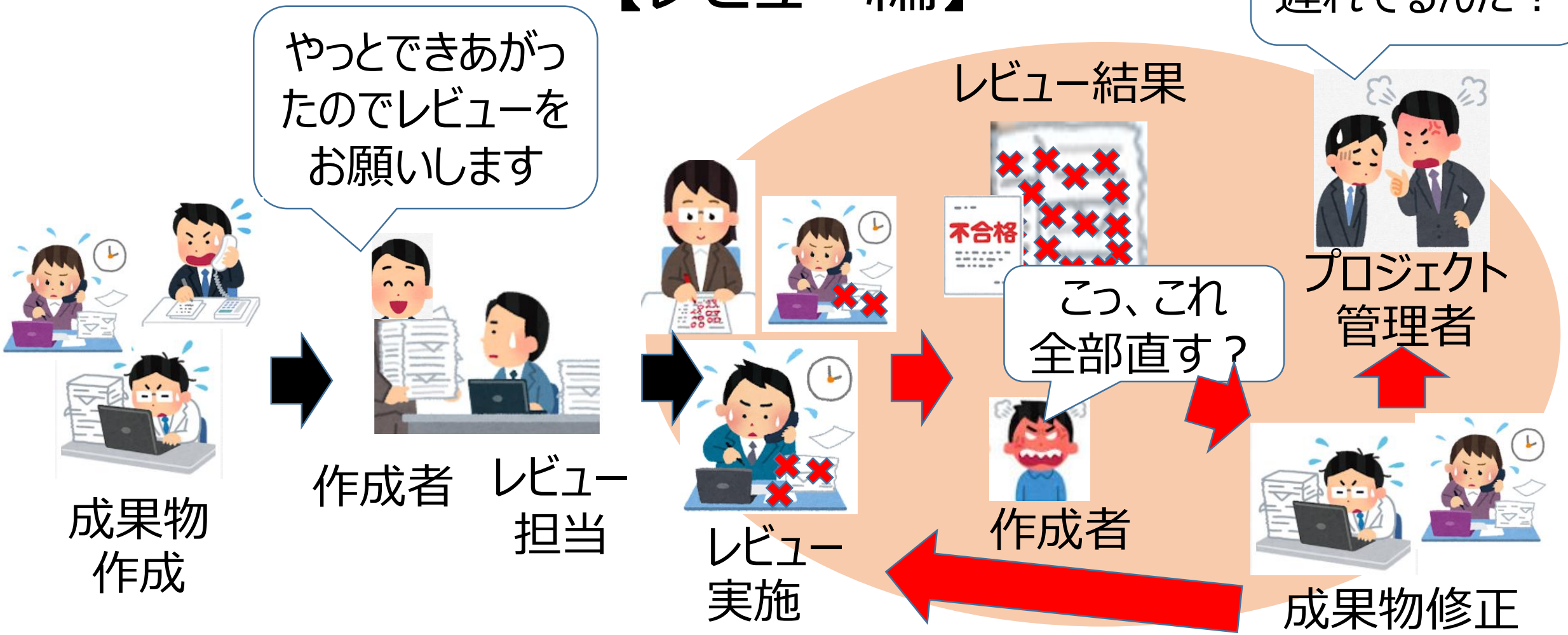
新しいチームは壁を乗り越えられれば成長し、跳ね返されると崩壊する

失敗から学ぶ文化と**相互理解**に基づく心理的安全性の確保がカギ

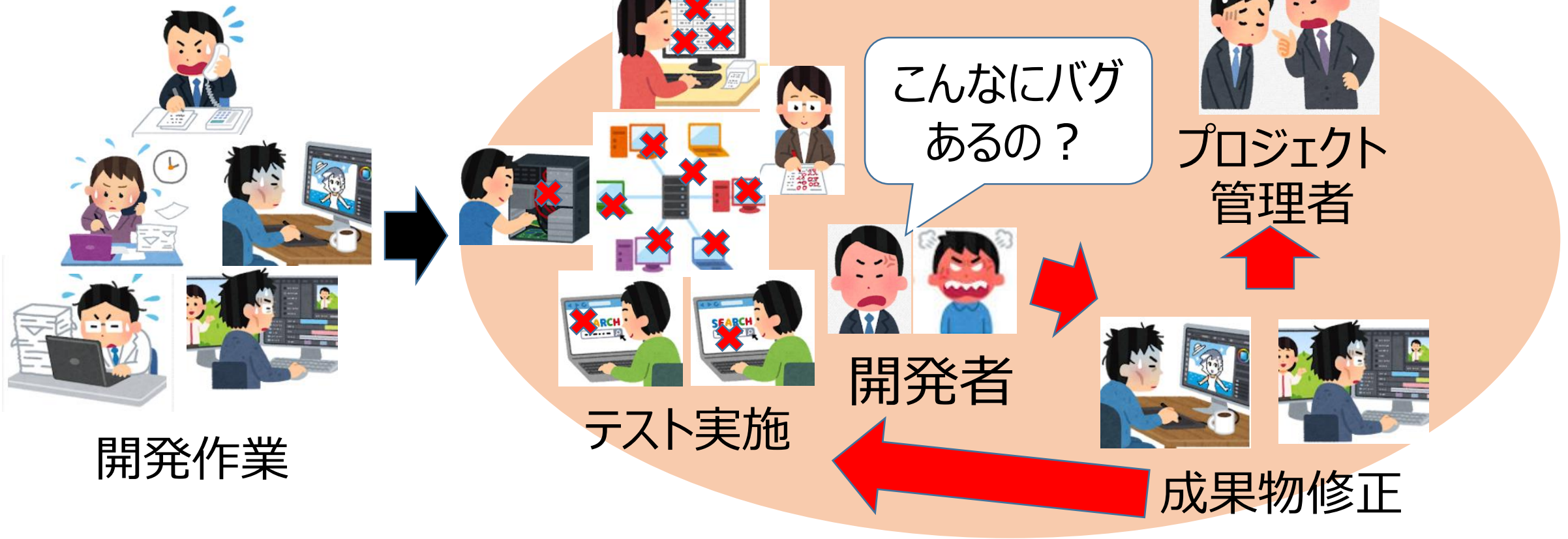
☑**大事なことは目に見えない**

本当のポリシー・場を支配するルール・信頼・気持ち・ノウハウ・共感など～
適切な方針を言動と行動で示し、共有する（知行合一）

こんなことになっていませんか？
【レビュー編】



こんなことになっていませんか？【テスト編】



それは**レビュー工数**ではなく**開発工数**です

手戻り

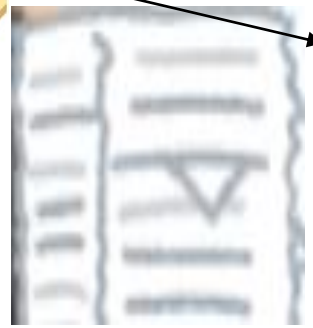
再レビュー依頼

レビュー分析・設計・実装

レビュー実行

再レビュー

レビュー終了処理



成果物確認・修正(デバッグ)

それは**テスト工数**ではなく**開発工数**です

手戻り

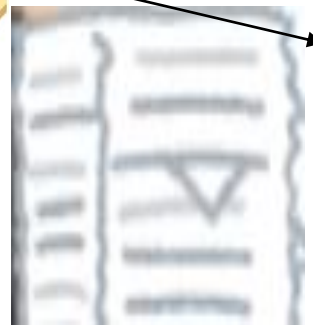
再テスト依頼

テスト分析・設計・実装

テスト実行

再テスト

テスト終了処理



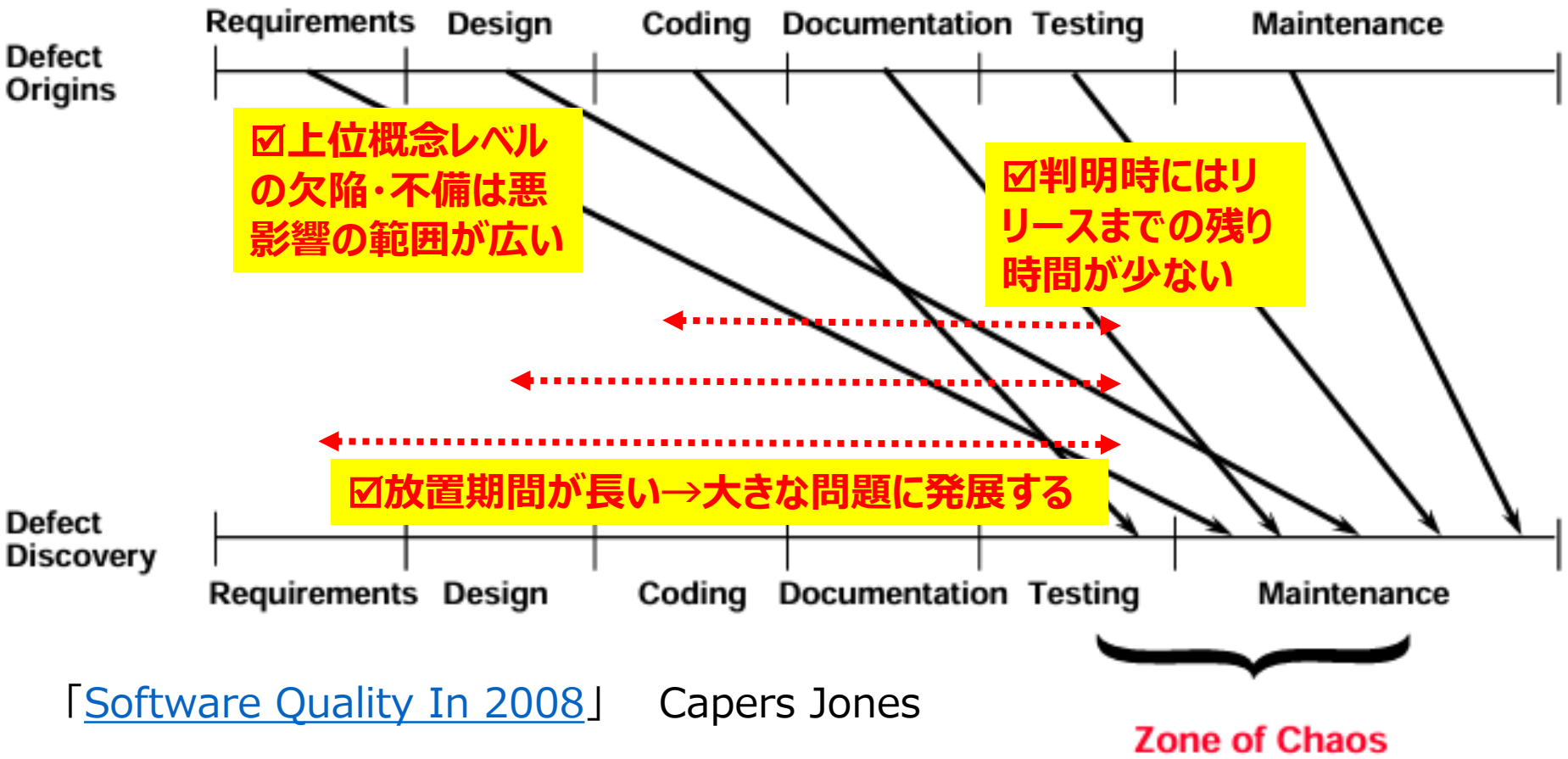
成果物調査・修正(デバッグ)

プロジェクトの混乱/スピード鈍化の原因 [**手戻り＝やり直し・修正作業**]

- **手戻り**
  手順や成果物の内容を間違えて作業をやり直すこと。**追加工数が必要**で**時間もかかる**ため**開発スピードが鈍化**する。
- **手戻り発生の主な要因**
  - 顧客等からの変更要求
  - 関係者間の齟齬／認識違い
  - レビュー・テストによる欠陥・不備の露呈

今日の話はここに着目

上流フェーズ成果物を対象としたレビューでの欠陥・不備の見逃しがソフトウェアプロジェクトに与える悪影響は大きい



「Software Quality In 2008」　Capers Jones

上流フェーズレビューによる欠陥・不備の見逃し防止・緩和は プロジェクトリスクを大幅に低減する

今日は

【レビュー観点の活用】

を中心にお伝えします

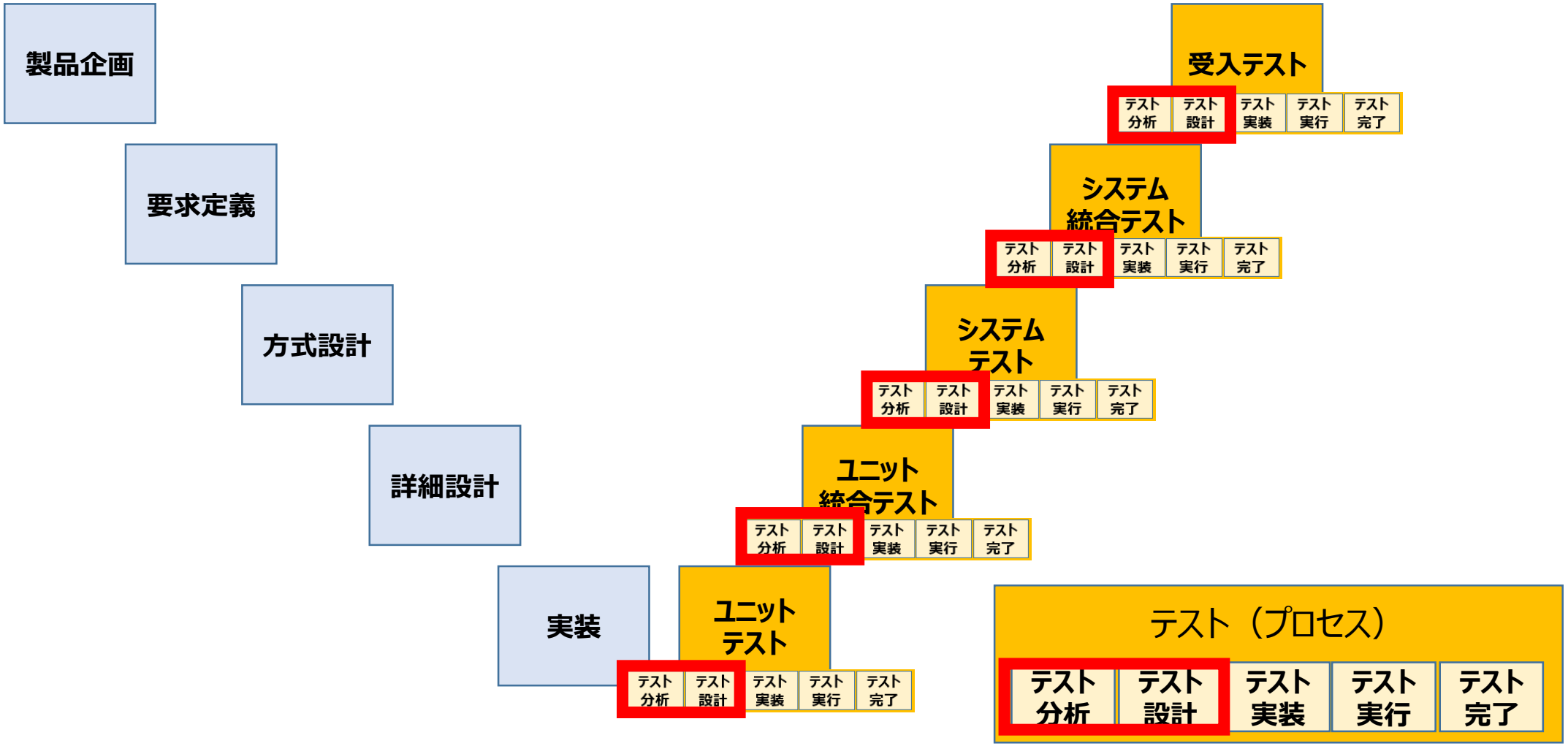
※テスト観点も同様の考え方で対応可能です！

Quality・Speed・Costをマルっと変えるために

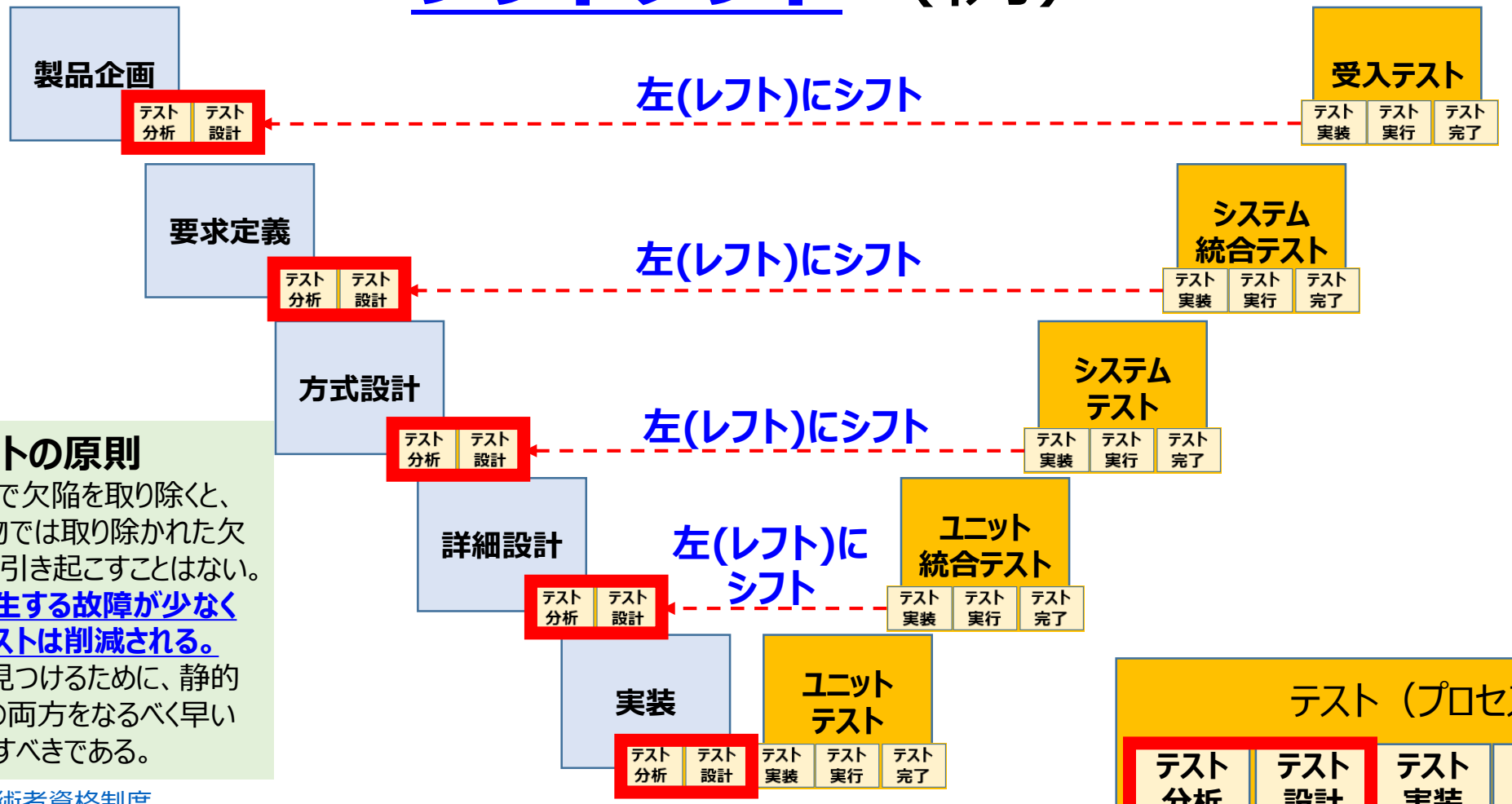
シフトレフト

- 「シフトレフト」はソフトウェア開発プロセス～開発ライフサイクルの右側にある**テストフェーズ（の一部）を開発フェーズから実施し、不具合を早期に検出・修正する手法**
- **不具合を早期に検出し修正することで、手戻りを削減し、レビュー、テストと修正にかかる時間とコストを節約**し、**最終的にプロジェクト全体期間を短縮する**

開発プロセス（Vモデル）



テストプロセスの前段部分を先出しする **シフトレフト**（例）

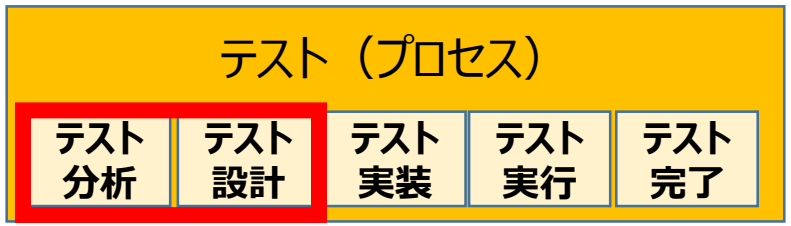


早期テストの原則

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

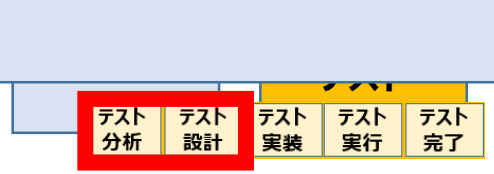
テストプロセスの前段部分を先出しする **シフトレフト**（例）

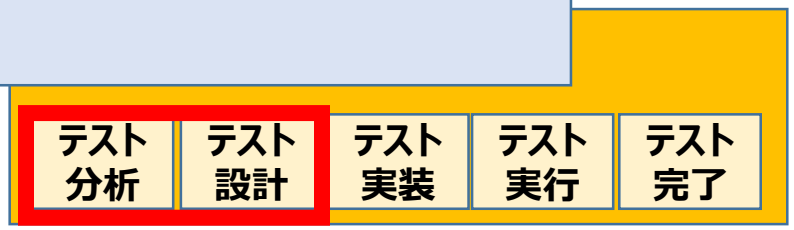
欠陥や不備を作り込んでから検出・除去するまでのリードタイムを短くする

【問題の早期発見・解決】

早期
プロセスの早い
その後の作業
陥に起因する
SDLCの後半
なるため、品
早い段階で欠
テストと動的テ
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

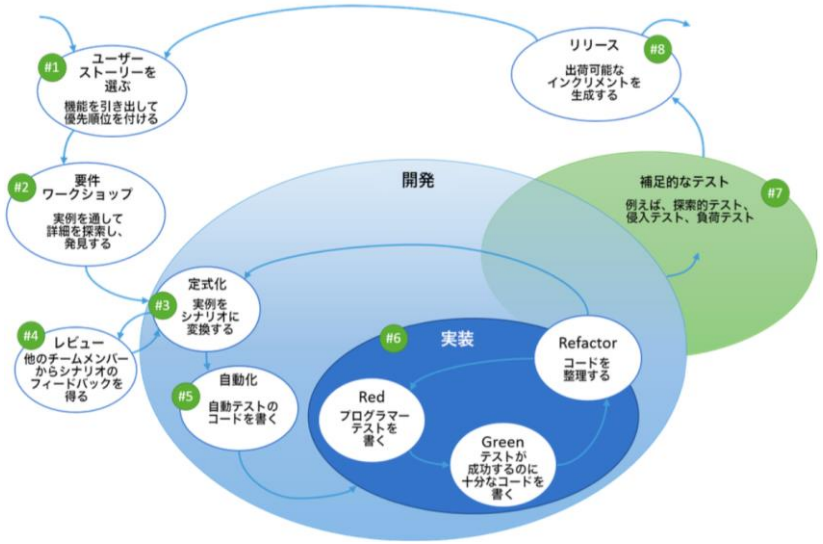




シフトレフトの原理を活用した テスト主導ソフトウェア開発

BDD

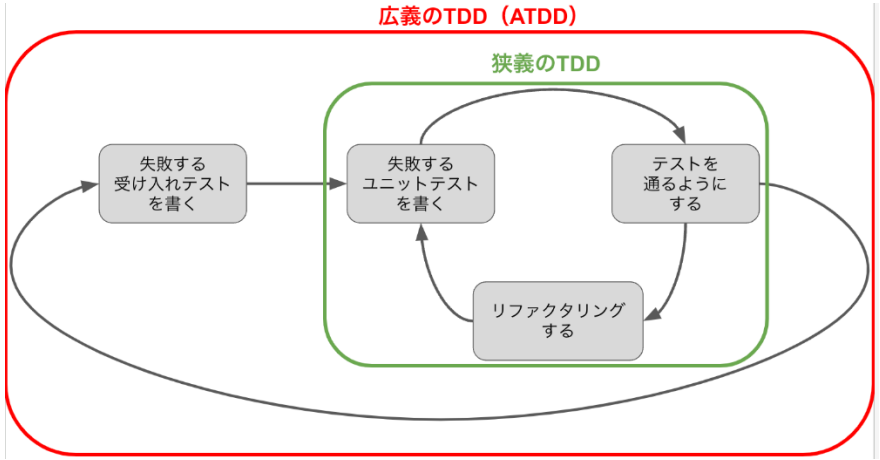
Behavior Driven Development
振る舞い駆動開発



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



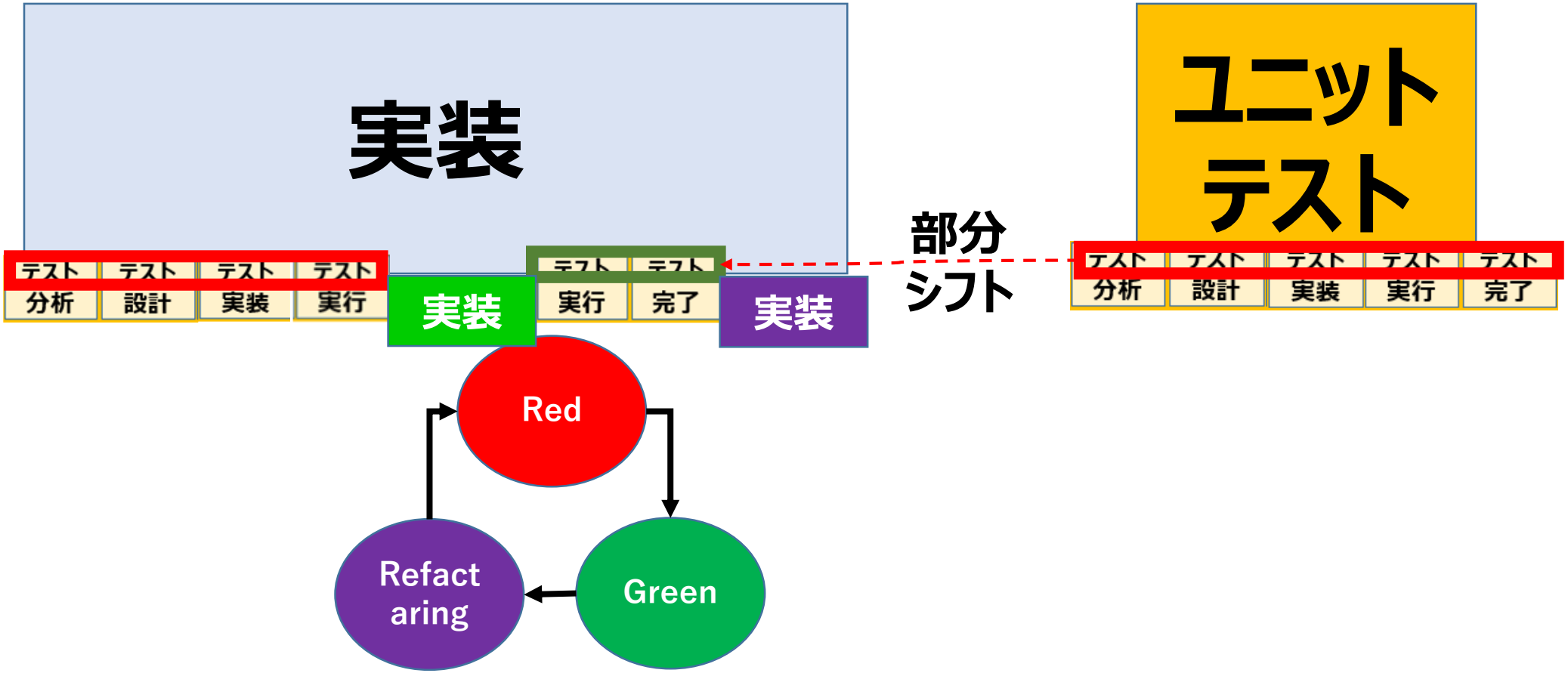
TDDとBDD/ATDD(3)
BDDとATDDとSbE より

TDD

Test-Driven Development
テスト駆動開発

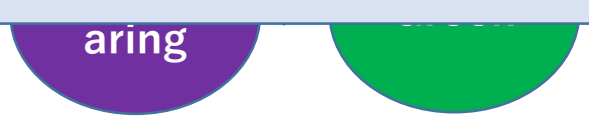
50分でわかるテスト駆動開発 /
TDD Live in 50 minutes より

TDD:テスト駆動開発は**欠陥・不備の早期発見**＋**欠陥・不備の作り込み防止**を志向するアプローチ

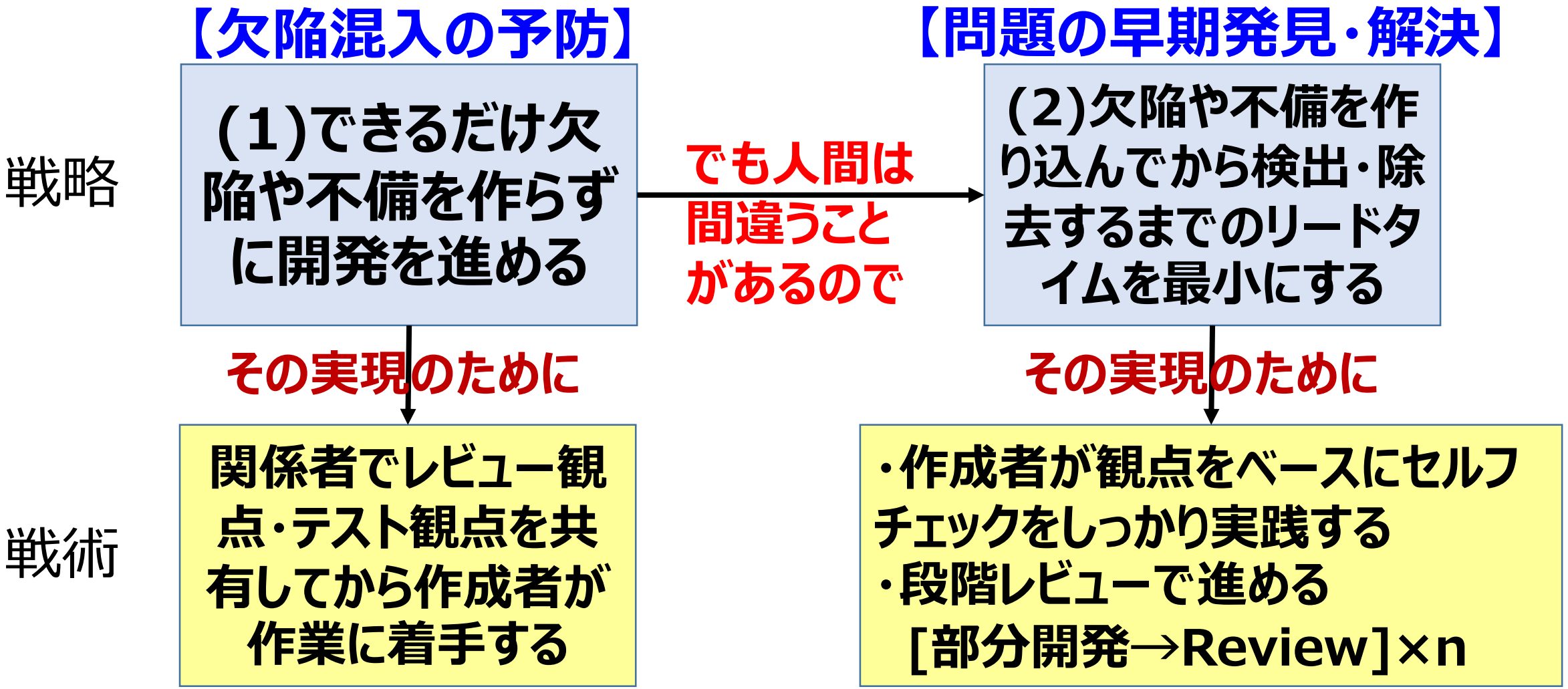


TDD:テスト駆動開発は**欠陥・不備の早期発見**＋**欠陥・不備の作り込み防止**を志向するアプローチ

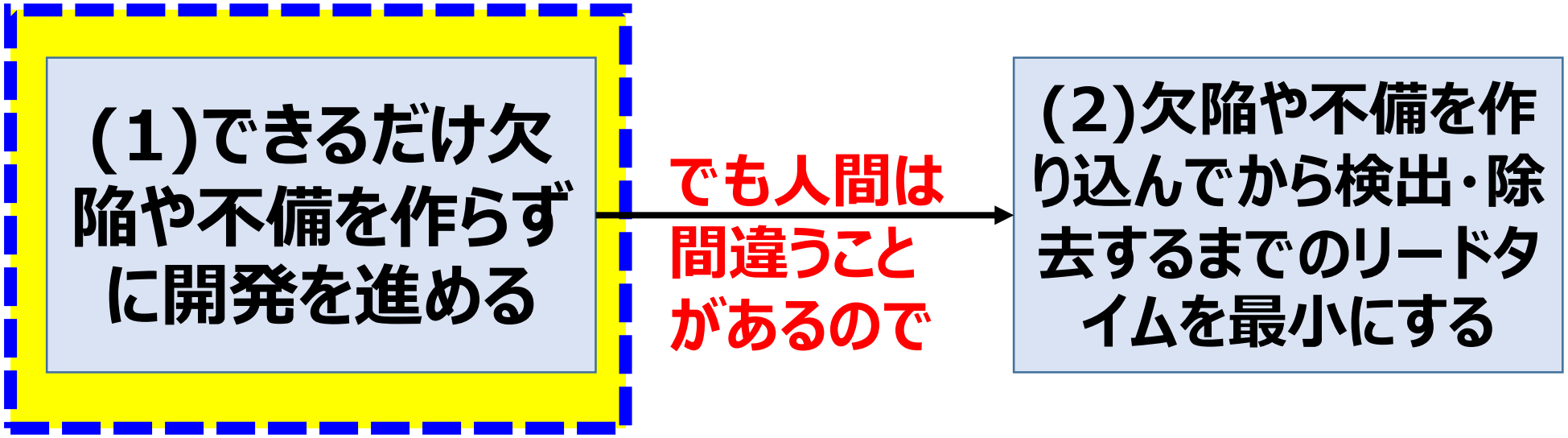
欠陥や不備を
できるだけ作り込まない
【欠陥混入の予防】



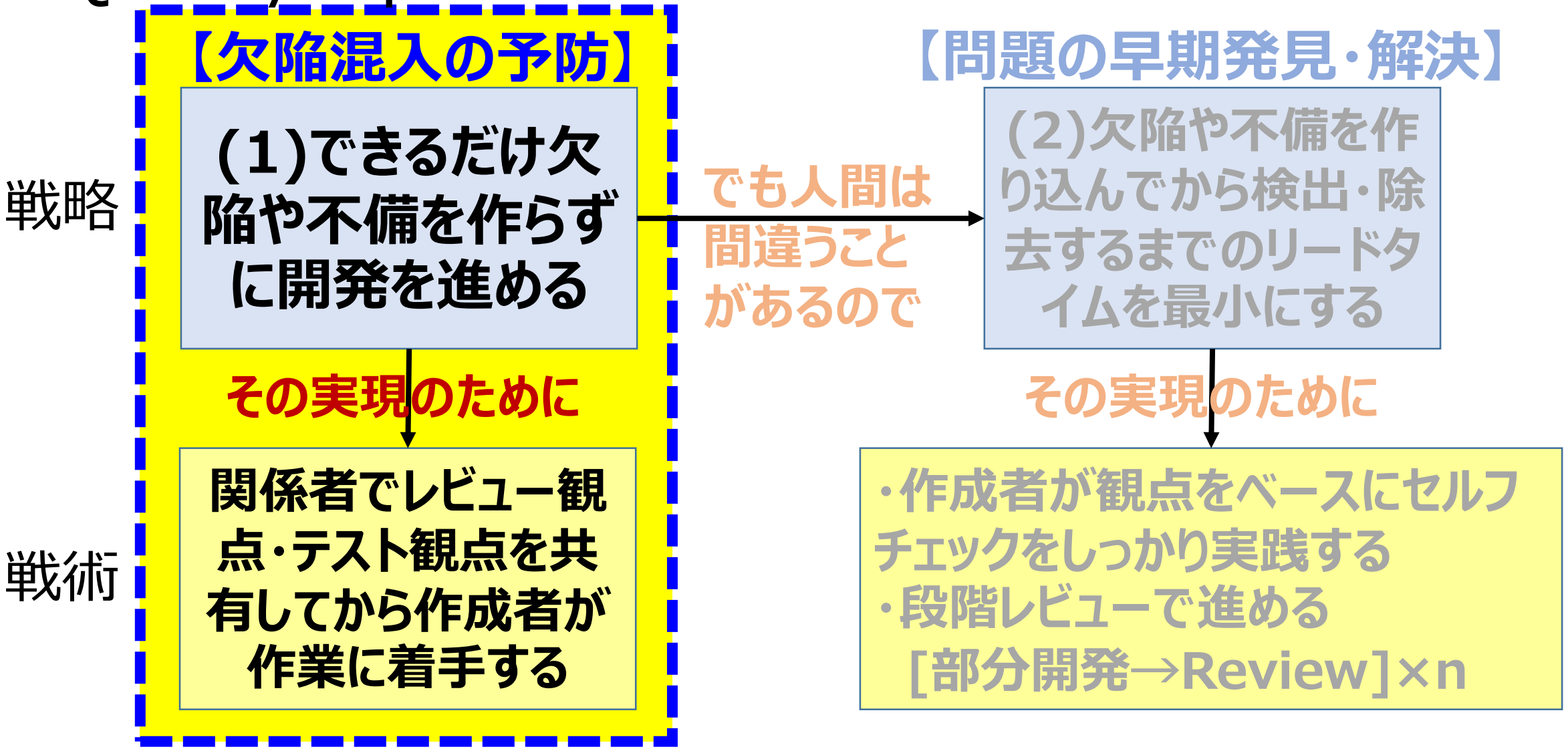
Quality・Speed・Costをマルっと変える戦略と戦術



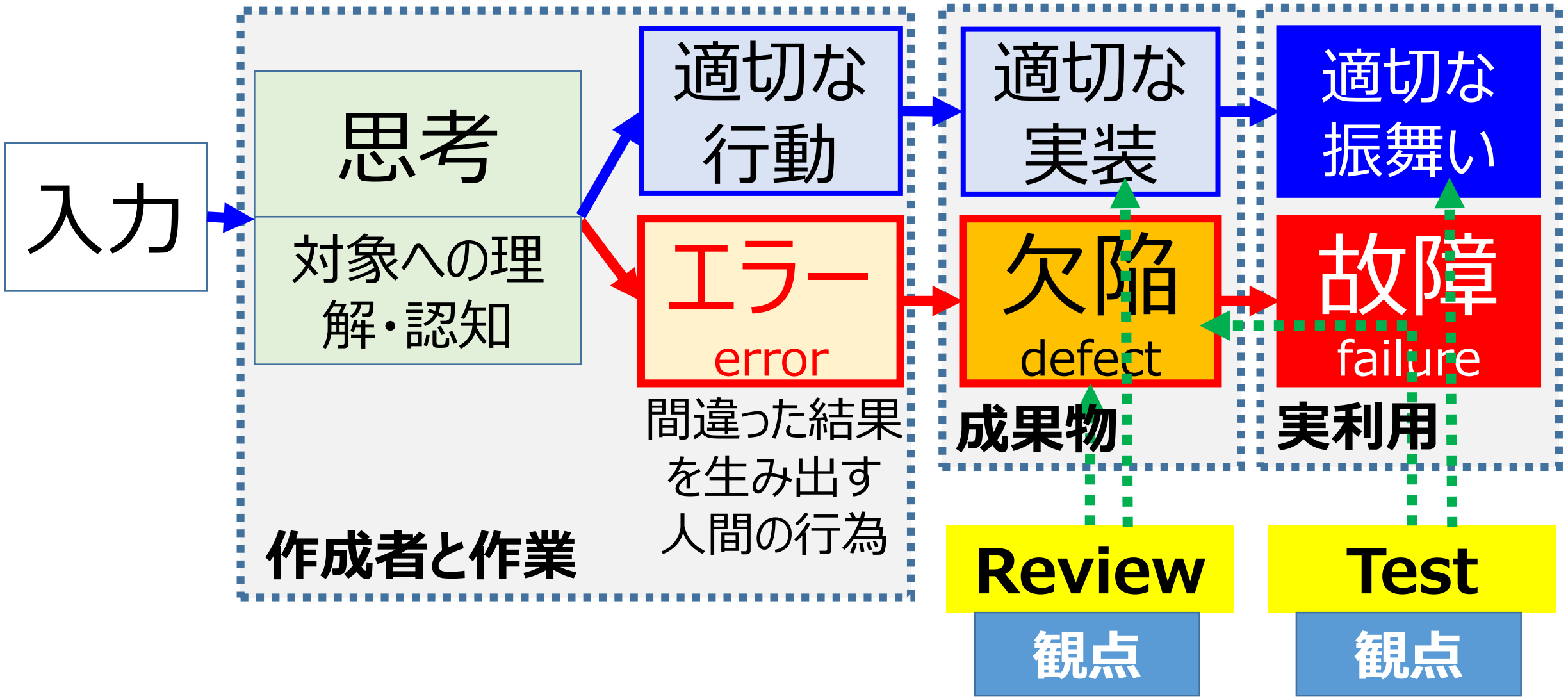
戦略1：欠陥混入予防



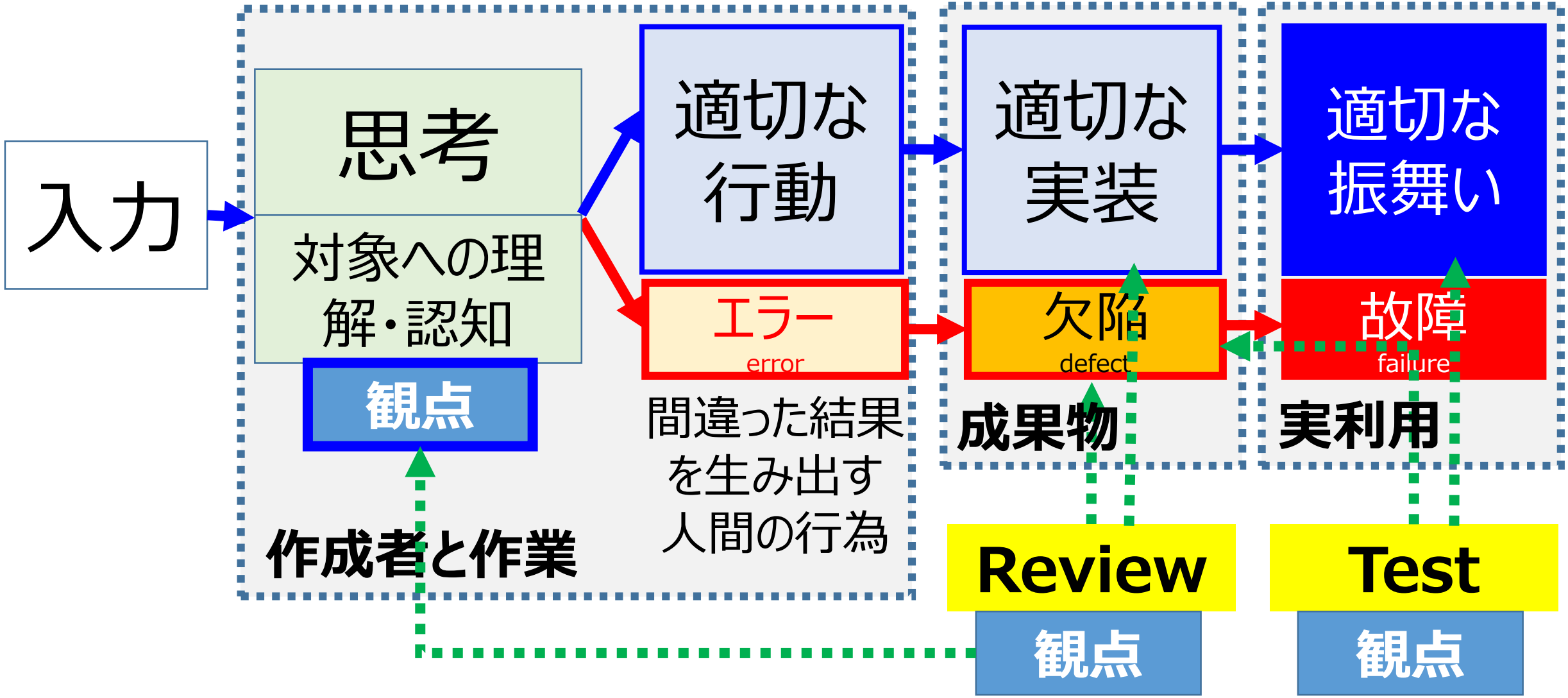
Quality・Speed・Costをマルっと変える戦略と戦術



欠陥混入の現状　[現状：Before]



今回の提案：**欠陥混入の予防**のカラクリ [After]



レビュー指摘には観点がある！

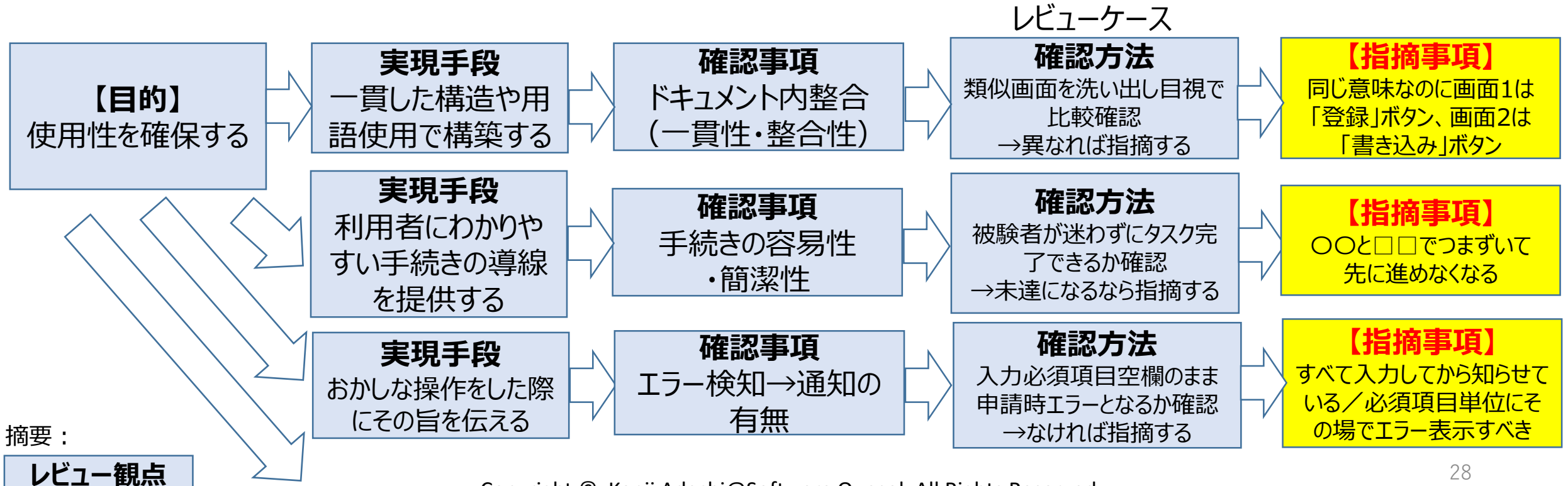
(4)どんな目的を達成するためのもの？	(3)どのような確認をしたことになる？	(2)どのように調べた結果？	**(1)レビューで検出した欠陥・不備の内容**
使用性・保守性が確保されていることを確実にする	**ドキュメント内、および画面・機能間整合確認（一貫性・整合性）** **システムを一貫した構造や用語使用で構築するため**	**類似画面を洗い出し、同じ意味のオブジェクトや説明を目視で比較確認→表現や形状が不一致の場合は指摘する**	**P1では「登録」ボタンなのに、P3では「書き込み」ボタンになっている**

左に一つずつシフトしながら回答する

検出した欠陥・不備を転記

レビュー観点とは？

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、レビューで欠陥を見つけるために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を含む。



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】

粒度が小さい

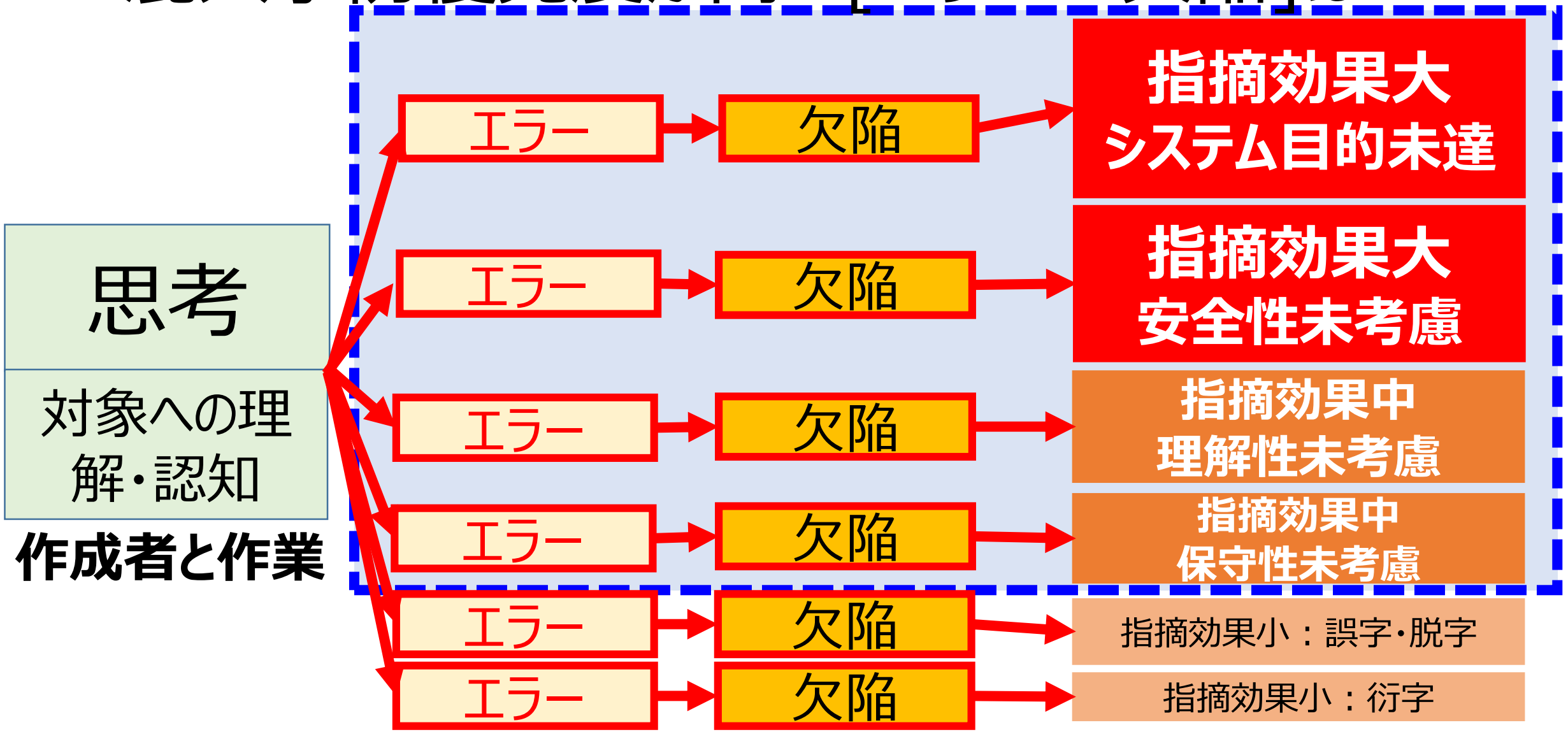
抽象度が低いレビュー観点（具体的）

誰がチェックして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】

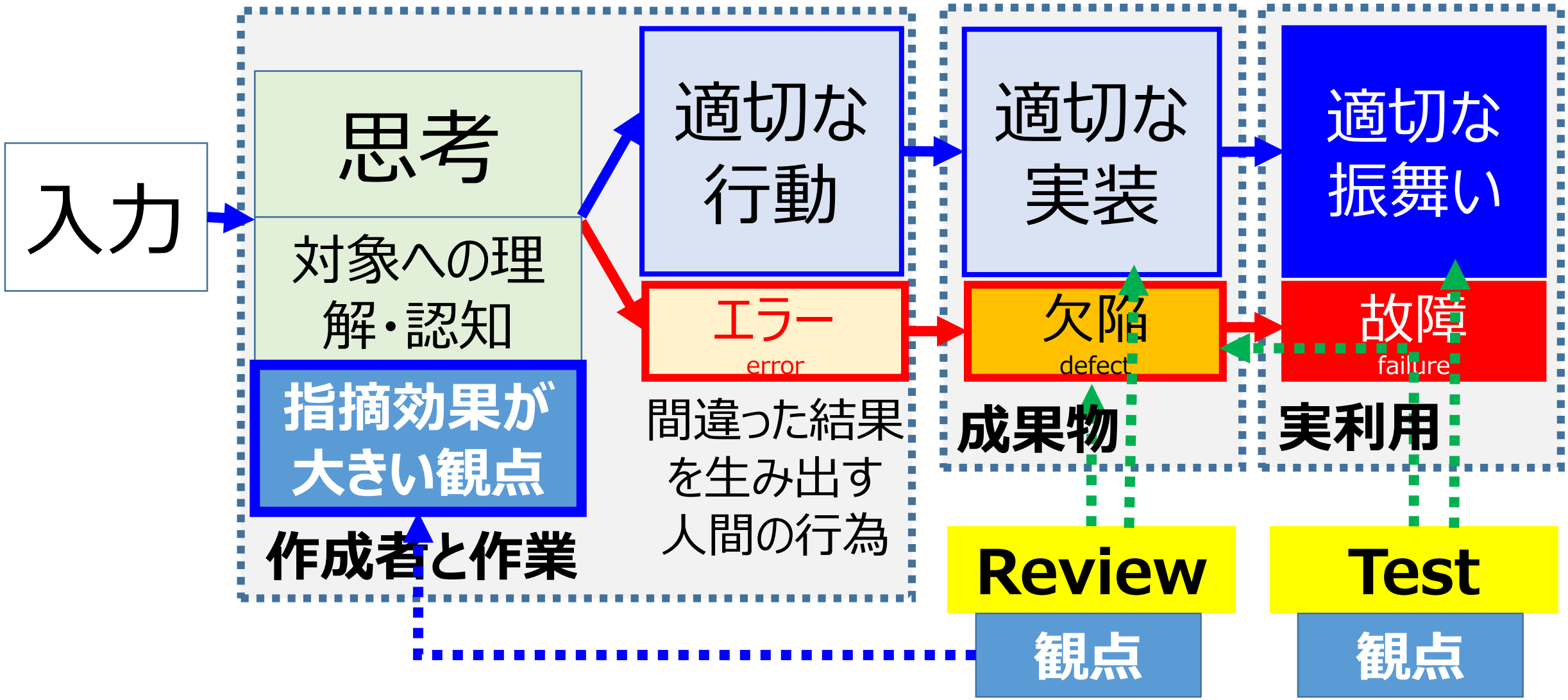
レビューで指摘した欠陥・不備はそれぞれ**効果が異なる**
指摘効果＝それを見逃した場合の悪影響度≒手戻り規模

指摘事項(欠陥・不備例)	指摘事項の意味(観点)	指摘効果
この機能構成だけでは利用者課題を解決できない	**利用者課題不解決＝システム目的未達**	**大**
子供がいたずらして使うとケガをする可能性が高い	**安全性未考慮**	**大**
抽象用語と文章説明ばかりで内容がわかりにくい	**理解性・保守性未考慮**	**中**
回収→改修 **格納る→格納する** **書き換込む→書き込む**	**誤字・脱字・衍字**	**小**

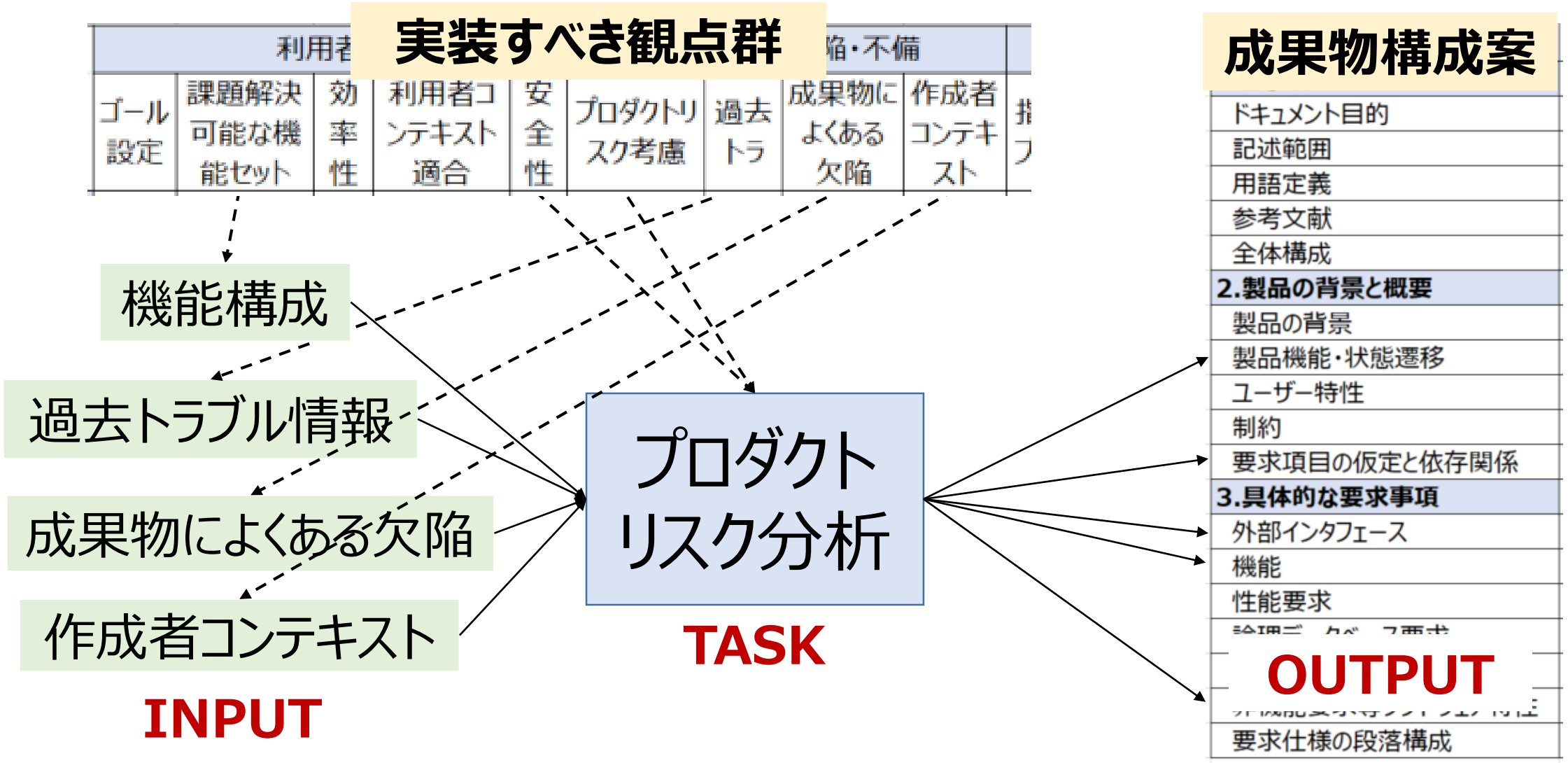
混入予防優先度が高い[エラー→欠陥]は？



欠陥混入予防は「指摘効果が大きい」観点優先で



レビュー観点活用例1：フェーズのタスク設計



レビュー観点活用例2：**成果物観点実装マッピング**

成果物構成案に観点割り付け←→成果物実装→セルフチェック

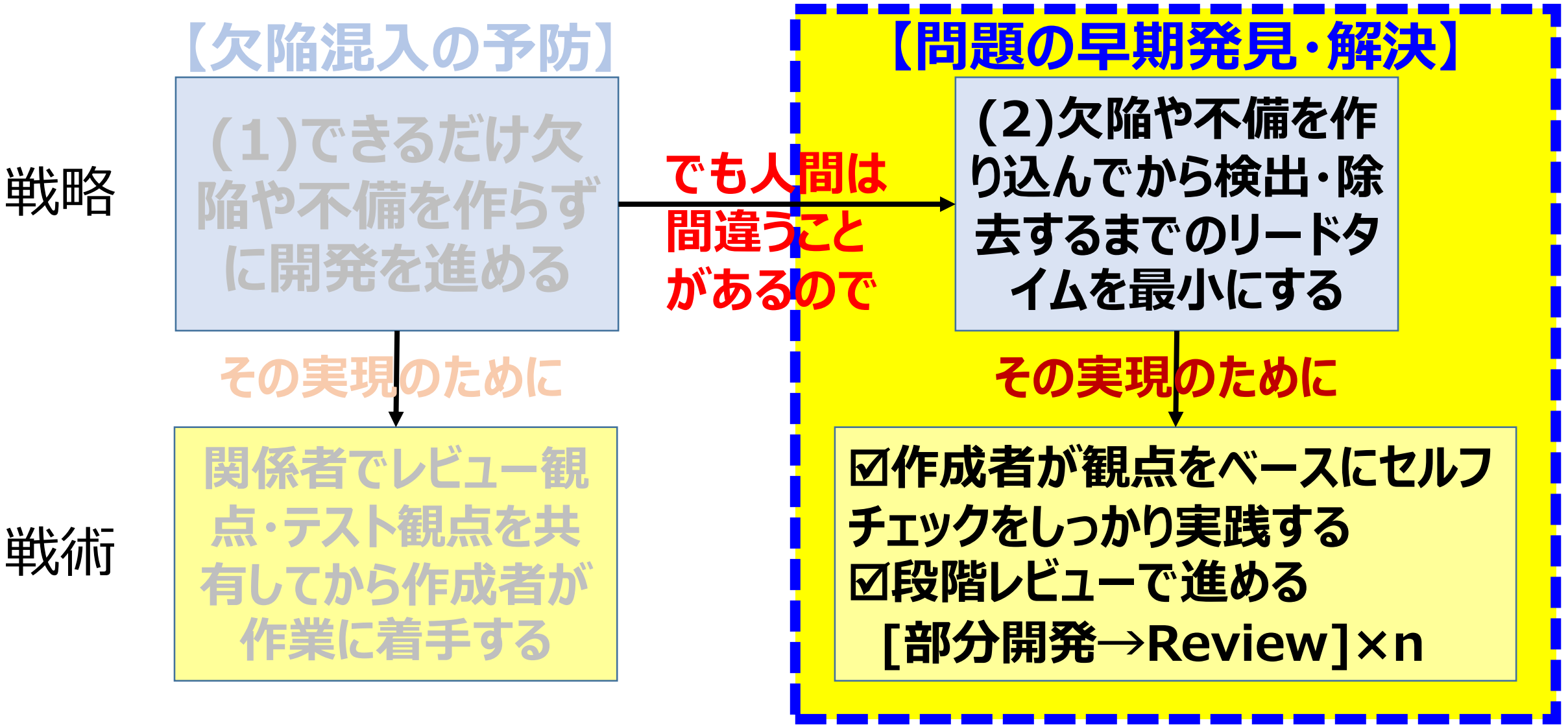
実装すべき観点群[優先度考慮]

成果物構成案

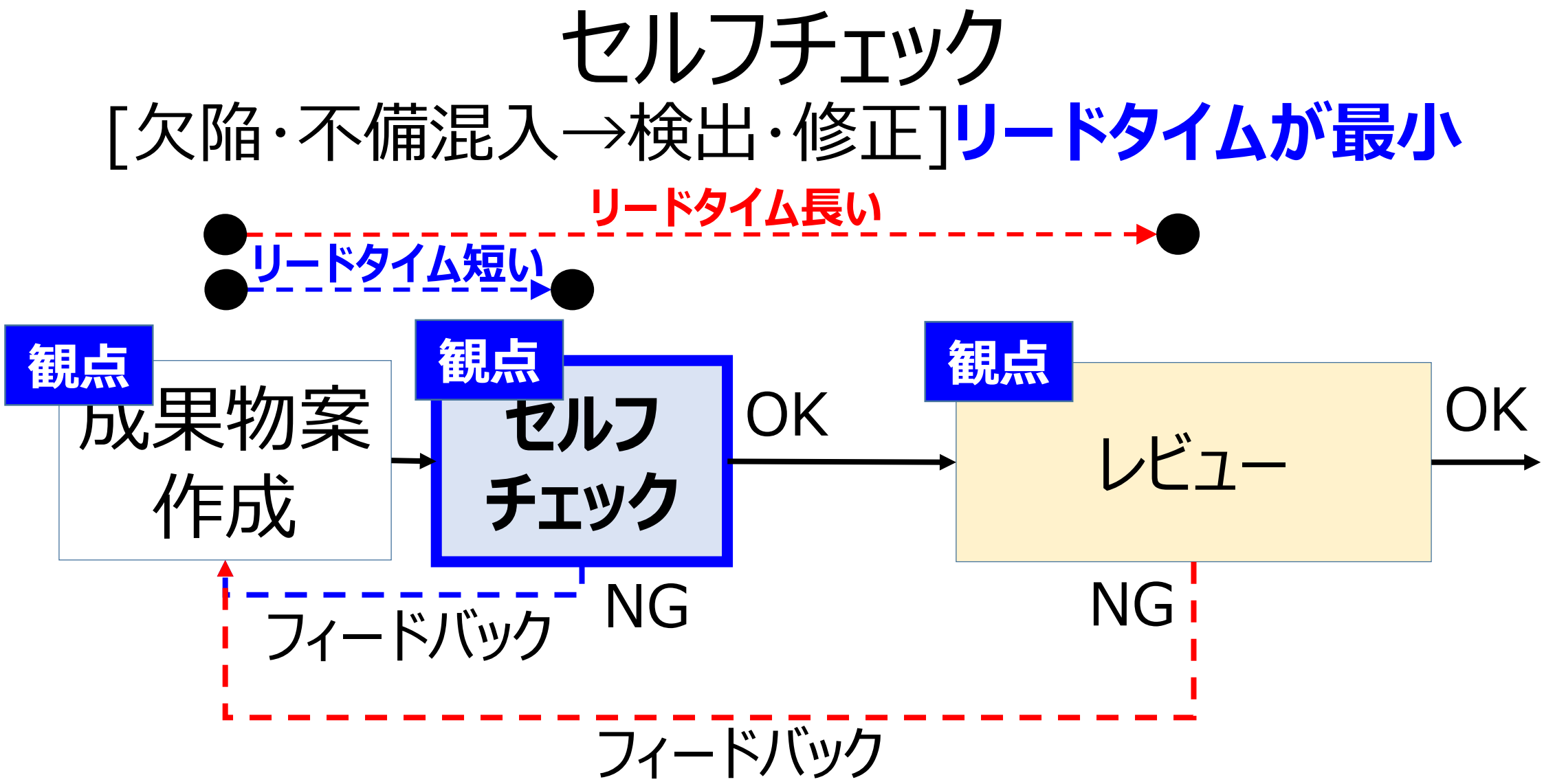
実装すべき観点→ ↓成果物構成案	利用者・利用時観点						欠陥・不備			保守性				
	ゴール設定	課題解決可能な機能セット	効率性	利用者コンテキスト適合	安全性	プロダクトリスク考慮	過去トラ	成果物によくある欠陥	作成者コンテキスト	指定テンプレ使用	図表で示す	用語統一	課題-要件-仕様は追跡可能	
1.はじめに														
ドキュメント目的														
記述範囲	□								□					
用語定義									□			□		
参考文献														
全体構成	□													
2.製品の背景と概要														
製品の背景	□			□					□		□			
製品機能・状態遷移		□			□			□			□			
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			□					□	□				□	

戦略2：問題早期発見･解決

Quality・Speed・Costをマルっと変える戦略と戦術



☑作成者が観点をベースにセルフチェックをしっかり実践する



☑作成者が観点をベースにセルフチェックをしっかり実践する

セルフチェック徹底～レビュー開始基準励行

同じ欠陥・不備を見つけるためのレビュー工数最小化

成果物案

- 指摘効果大 システム目的未達 — 欠陥
- 指摘効果大 安全性未考慮 — 欠陥
- 指摘効果中 理解性未考慮 — 欠陥
- 指摘効果中 保守性未考慮 — 欠陥
- 指摘効果小：誤字・脱字 — 欠陥
- 指摘効果小：衍字 — 欠陥

セルフチェック

☑修正
☑修正
☑修正
☑修正



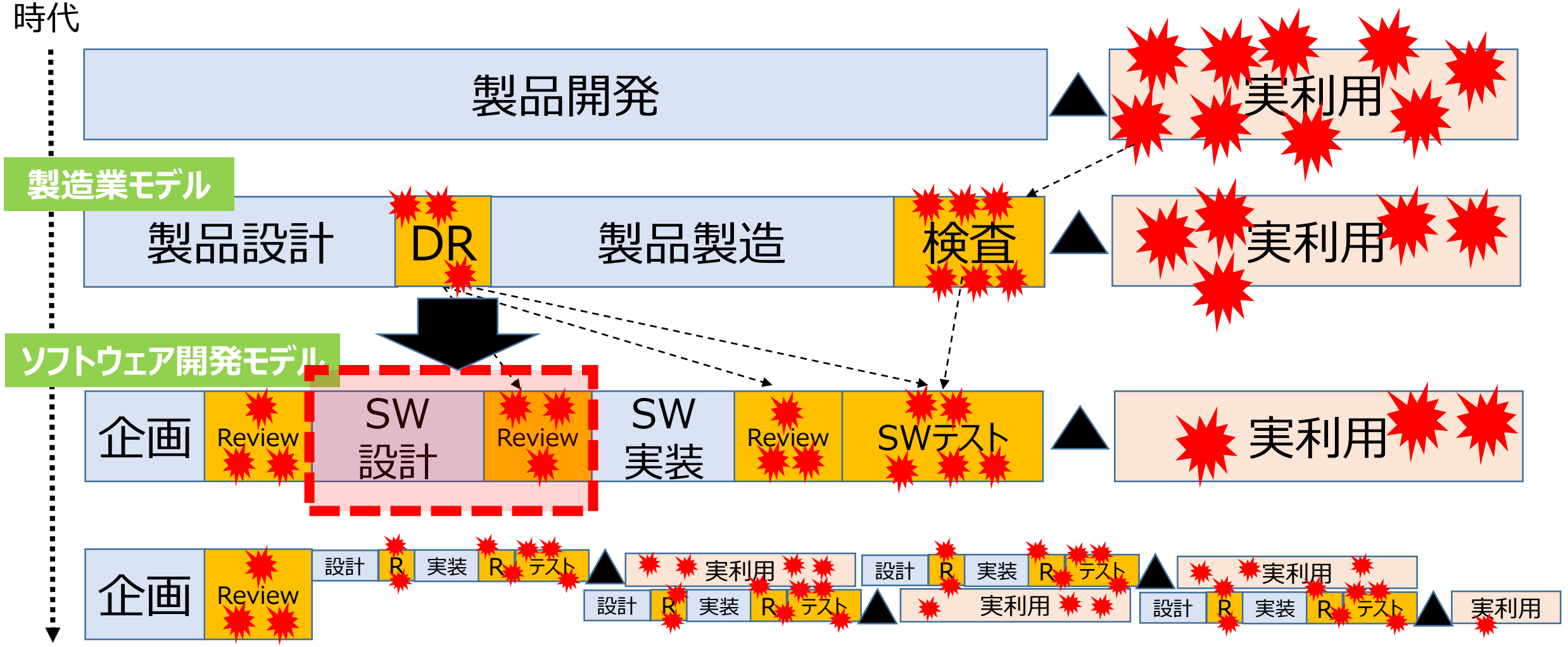
レビュー

☑ ☑ ☑
☑ ☑ ☑

セルフチェックによりレビューアの
質問や欠陥・不備を記録する
質問や欠陥・不備を伝える
工数が削減できる

☑段階レビューで進める　[部分開発→Review]×n

製品・サービス開発と品質保証の変遷



☑段階レビューで進める　[部分開発→Review]×n

成果物案が完成してからレビュー→手戻りを促進

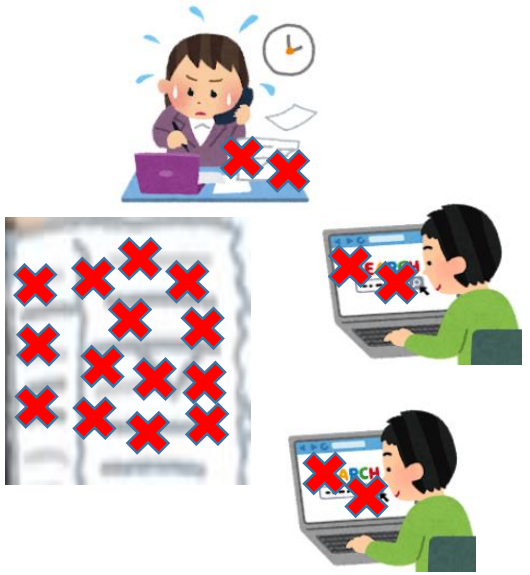
スピードが鈍化＋余計にコストがかかる+**全員が不幸**に

一人の知見で作る

やっとできあがったのでレビューをお願いします

作成者の**誤認識や不認識、癖等が成果物全体に反映**される

他者の知見が入る



混入した多くの欠陥を見つける＋記録して伝えるために**工数と時間がかかる**

欠陥の見逃しも増える

指摘された欠陥を漏れなく直すために**工数と時間がかかる**

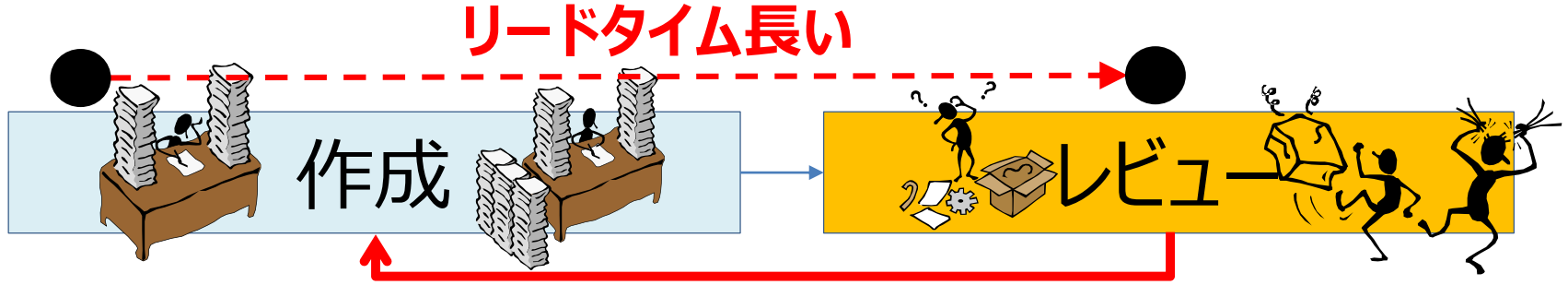
あとになってから発覚して余計な時間と工数がかかる

☑段階レビューで進める　[部分開発→Review]×n

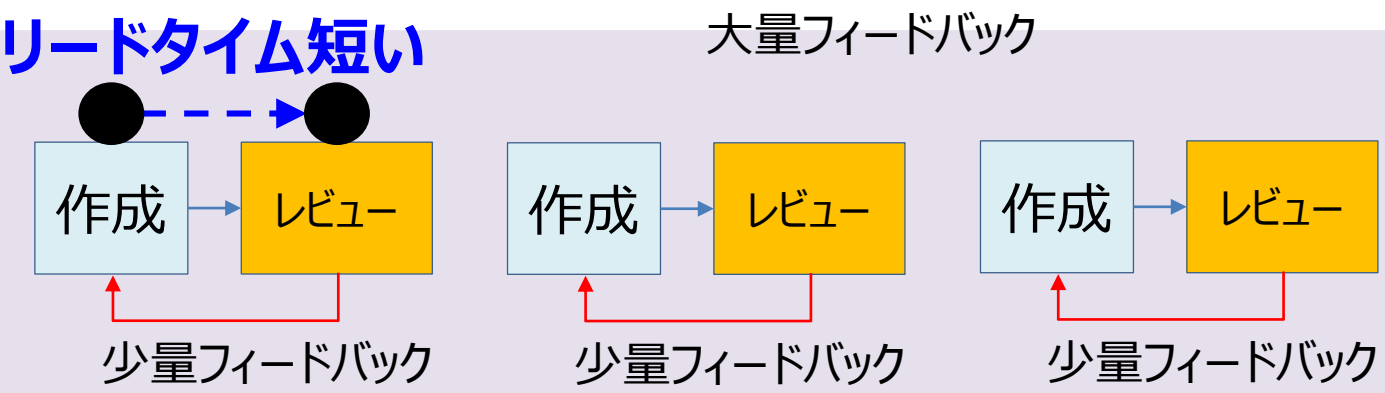
一括レビュー → 段階レビュー → モブ○○ へ

[作成－フィードバック]のリードタイム最小化

一括レビュー



段階レビュー



モブ○○

作成：Driver

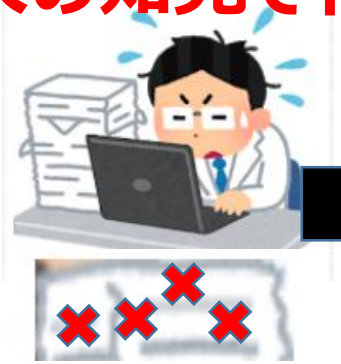
フィードバック：Navigator

☑段階レビューで進める　[部分開発→Review]×n

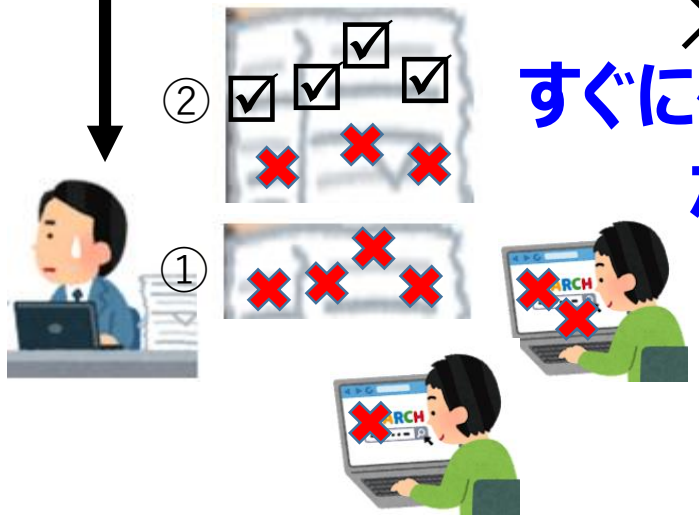
[部分成果物作成⇒レビュー]×n回で進める

総混入欠陥数が減少し、レビュー工数も減少する

一部分を
一人の知見で作る



作成者の**誤認識や認識不足、癖等が**部分成果物に**反映**される

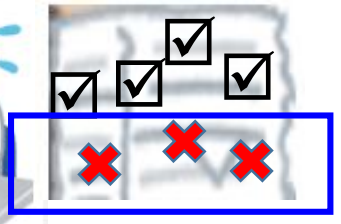


×n回
すぐに他者の知見が入る

部分成果物なので、**少ないレビュー工数で欠陥を見つける＋記録して伝えられる**

修正

次の部分成果物作成

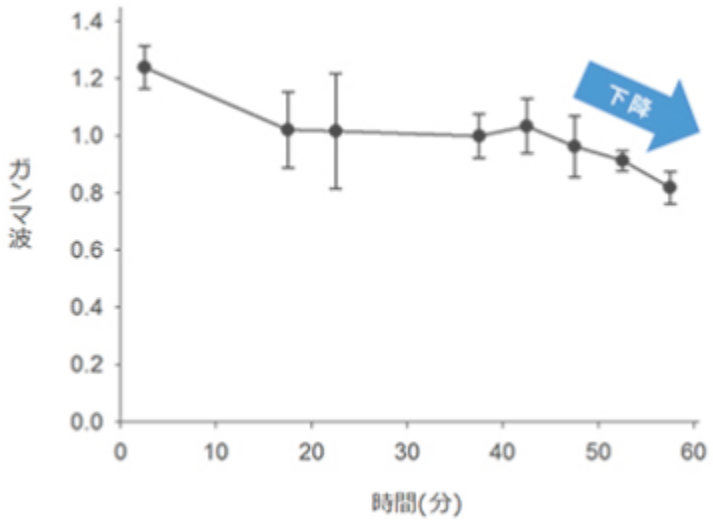


レビュー結果により**自分の誤認識、認識不足、癖等を把握して以降の作業を注意しながら進められる(欠陥の作りこみが減少)**

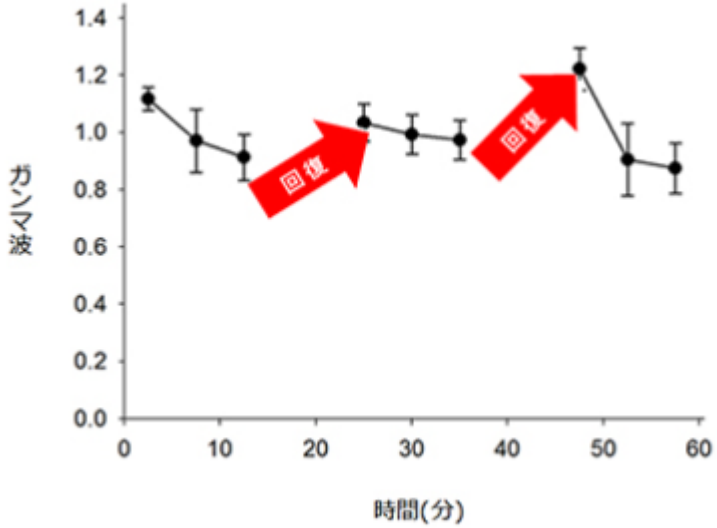
☑段階レビューで進める　[部分開発→Review]×n

段階レビューにすると欠陥検出率が向上する

1回のレビュー規模を小さく→レビュー時間短縮→欠陥検出力向上



６０分学習グループの
ガンマ波波形



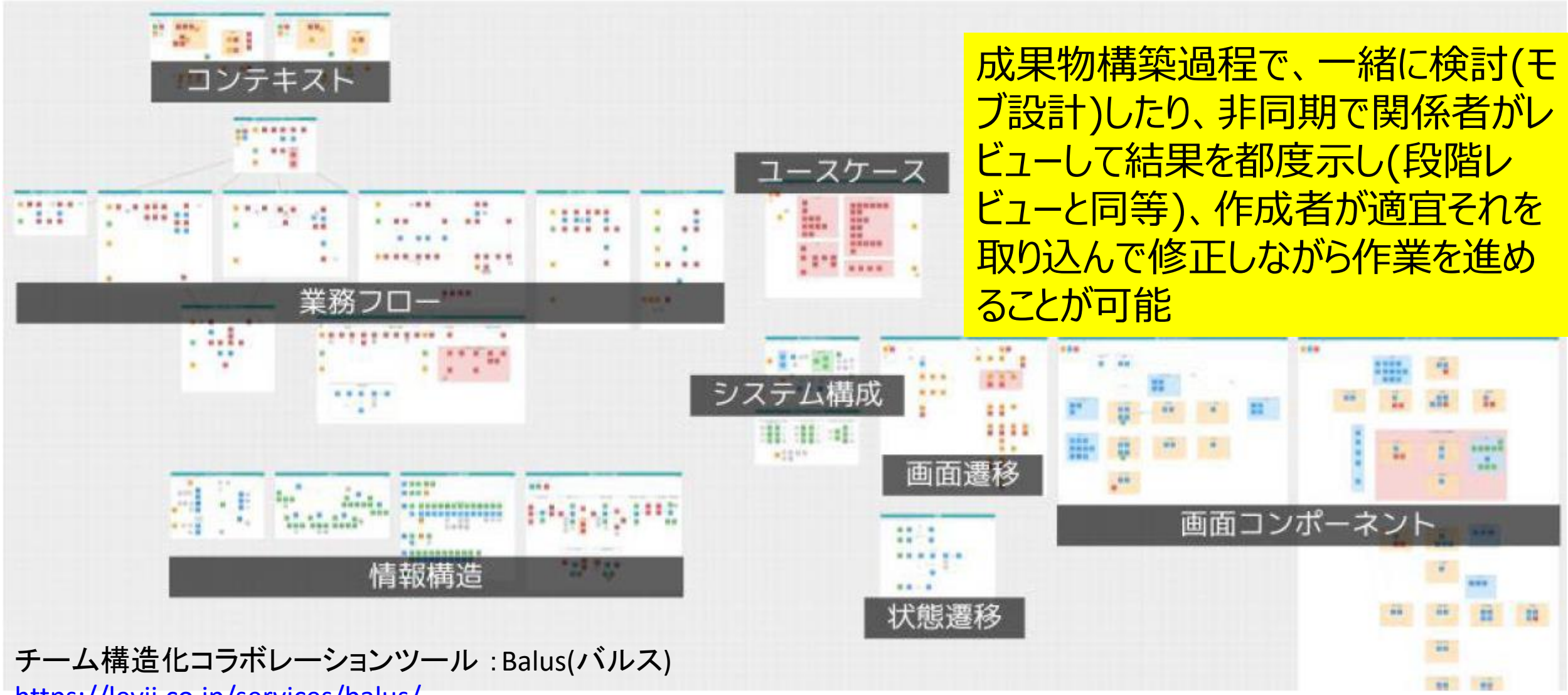
１５分×３（計４５分）
学習グループのガンマ波波形

※グラフ１と２のガンマ波の絶対値の大小は関係なし

集中力の維持と長期的な学習効果につながる方法
（東京大学・池谷裕二教授の見解）より
http://www.asahi.com/ad/15minutes/article_02.html

上流設計モデリング環境例

上流フェーズ担当者＋関係者全員がアクセス



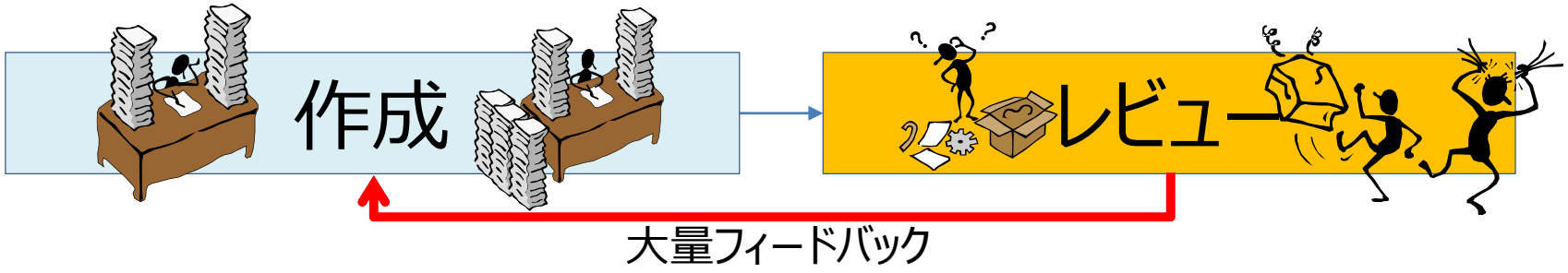
成果物構築過程で、一緒に検討(モブ設計)したり、非同期で関係者がレビューして結果を都度示し(段階レビューと同等)、作成者が適宜それを取り込んで修正しながら作業を進めることが可能

チーム構造化コラボレーションツール ：Balus(バルス)
https://levii.co.jp/services/balus/

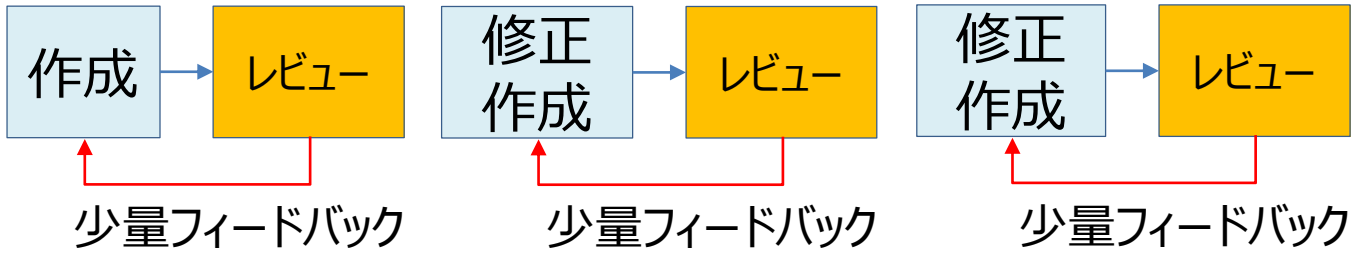
[戦略1×戦略2]の実践スタイルと位置づけ

まとめ：[戦略1×戦略2]の実践スタイルと位置づけ

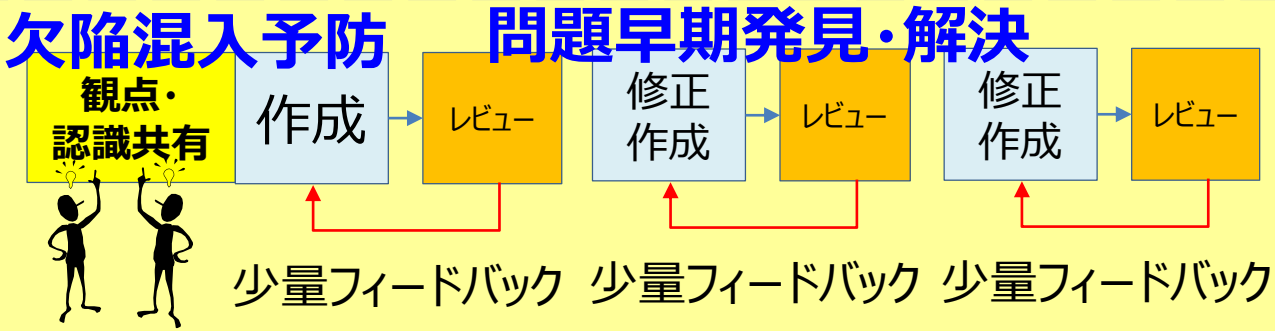
一括レビュー



段階レビュー



事前観点・認識共有
＋段階レビュー



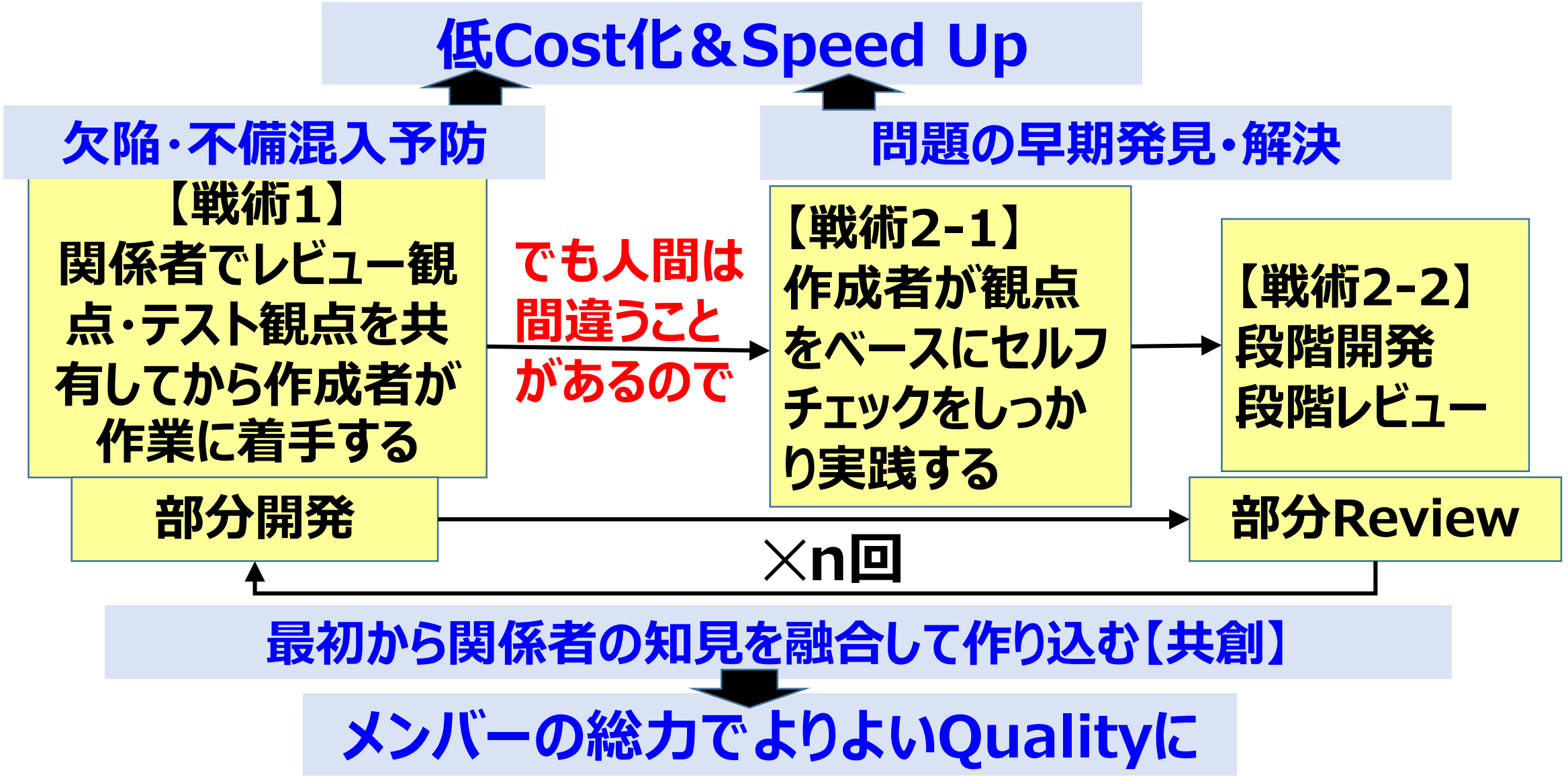
モブ〇〇

作成：Driver

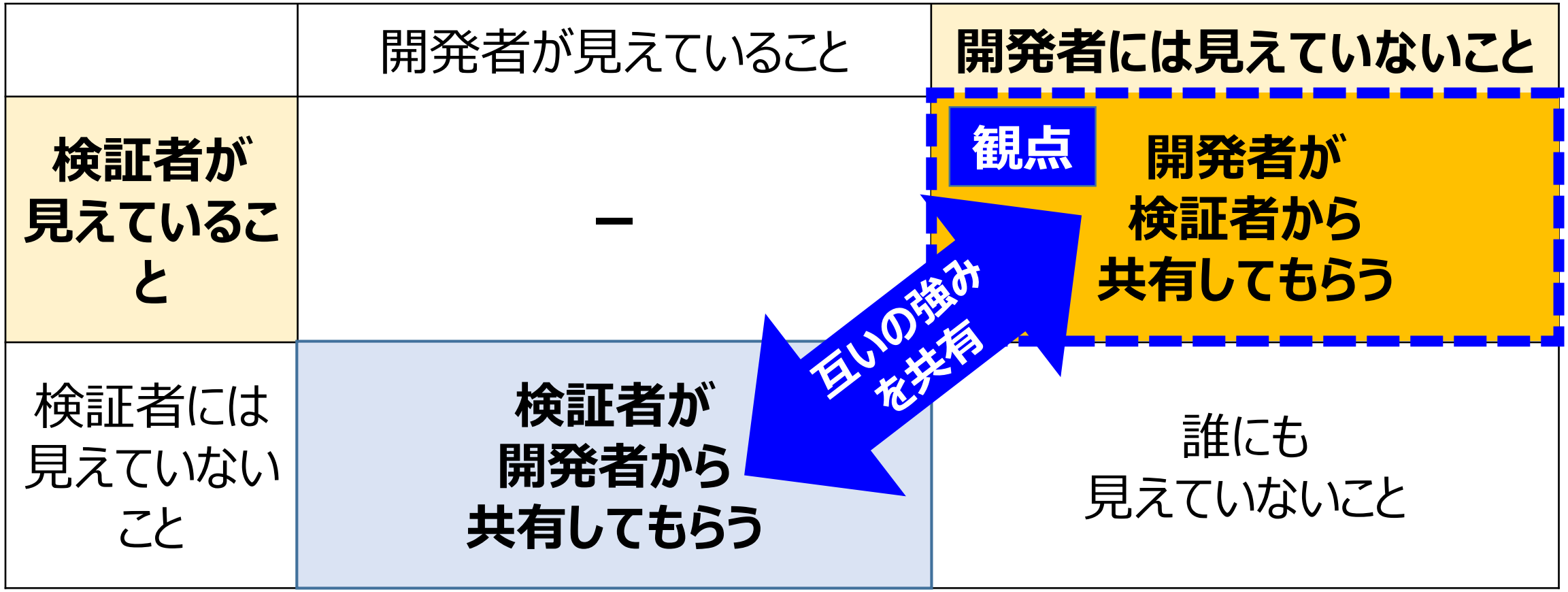
フィードバック：Navigator

この発表の意味と価値

Quality・Speed・Costをマルっと変える



メンバーの総力で良いモノを作る＝**共創**



最初から品質を作り込む→関係者がみな幸せに

	Before	After
開発者 	**一生懸命やっている**のに**後出しじゃんけん**的な**ダメ出しが多い**	不得意・気づかないことに他者の知見や**先出支援**が入るため**欠陥・不備が少なくなる**
	ダメ出しが多いので**修正に時間がかかる**	**修正件数が少ない**ため**短時間で済む**
	あとフェーズで**見逃した欠陥への対応が多い**ため**手戻り工数**と**余計な苦労増**	**見逃す欠陥も少ない**ため対応が最小限で済む
検証者 	**欠陥・不備が多い**ので**チェックと記録に時間がかかる**	欠陥・不備が少ないため**チェックと記録が短い時間で済む**
	欠陥・不備の見逃しも多くなる →**レビュー能力への疑念を持たれる**	短時間レビューのため集中でき、**欠陥・不備の見逃しが減る**
管理者 	**進捗遅延**と**突発問題にあくせく**する 最終的に**システム品質と生産性が悪い結果**に	**進捗遅延や予期せぬ問題発生が最小化**し、**システム品質と生産性が高まる**

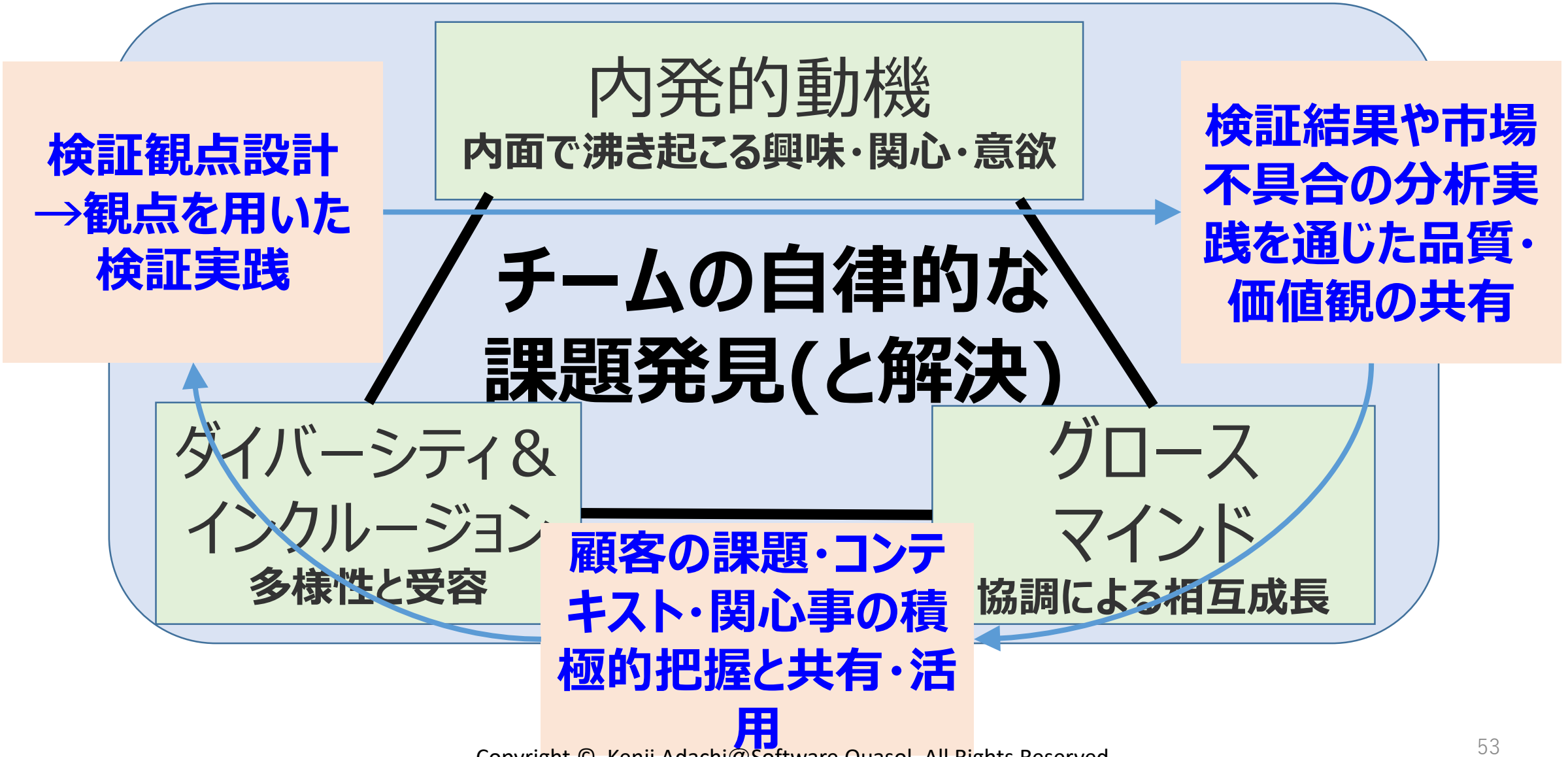
最初から品質を作り込む→関係者がみな幸せに

	Before	After
開発者 	**一生懸命やっている**のに**後出しじゃんけん**的な**ダメ出しが多い**	**手戻り工数が減る** ↓ **時間に余裕が生まれる** ↓ **マネージャ・リーダー・エンジニアとしてより価値のあるタスクやトレーニングに有効活用できる（QoLが高まる）**
	ダメ出しが多いので**修正に時間がかかる**	
	あとフェーズで**見逃した欠陥への対応が多い**ため**手戻り工数**と**余計な苦労増**	
検証者 	**欠陥・不備が多い**ので**チェックと記録に時間がかかる**	
	欠陥・不備の見逃しも多くなる →**レビュー能力への疑念を持たれる**	
管理者 	**進捗遅延**と**突発問題にあくせく**する 最終的に**システム品質と生産性が悪い結果**に	

この提案の実践に必要なこと

この提案の実践に必要なこと

組織的要因 | **技術的要因**



おわりに

シャープ伝説のエンジニア
佐々木正氏（シャープ元副社長）の座右の銘

「いいかい、君たち。分からなければ聞けばいい。持っていないなら借りればいい。逆に聞かれたら教えるべきだし、持っているものは与えるべきだ。人間、一人でできることなど高が知れている。

技術の世界はみんなで共に創る『共創』が肝心だ。」

https://kitto-cea.com/column/detail/17

参考文献

- Software Quality In 2008(JaSST'08東京) Capers Jones
  https://www.jasst.jp/archives/jasst08e/pdf/A1.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.J02.pdf
- 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/
- 観点活用レビューワークでわかったこと 一意な観点設定から観点設計への壁(SQiP2024)
- 集中力の維持と長期的な学習効果につながる方法（東京大学・池谷裕二教授の見解）
  http://www.asahi.com/ad/15minutes/article_02.html
- システムモデルを用いた対話型上流設計によるサービス開発 - モデルで納品・モデルで開発・モデルで検証 -（三浦政司）
- 【ロケット・ササキ】 https://kitto-cea.com/column/detail/17