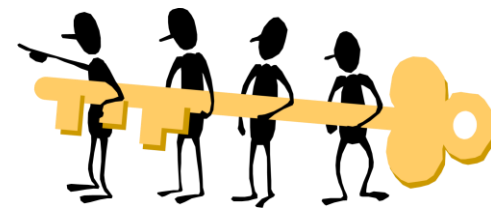


SaPID club vol.01のワークを実施した部分に必要なポイント解説を入れています。  
vol.02以降の実施による解説も順次追記していきます(つまりこの資料は今後も改訂される)のでご注意ください。



# サピッド SaPIDの概要

株式会社 HBA Software Quasol 安達 賢二

[adachi@hba.co.jp](mailto:adachi@hba.co.jp)

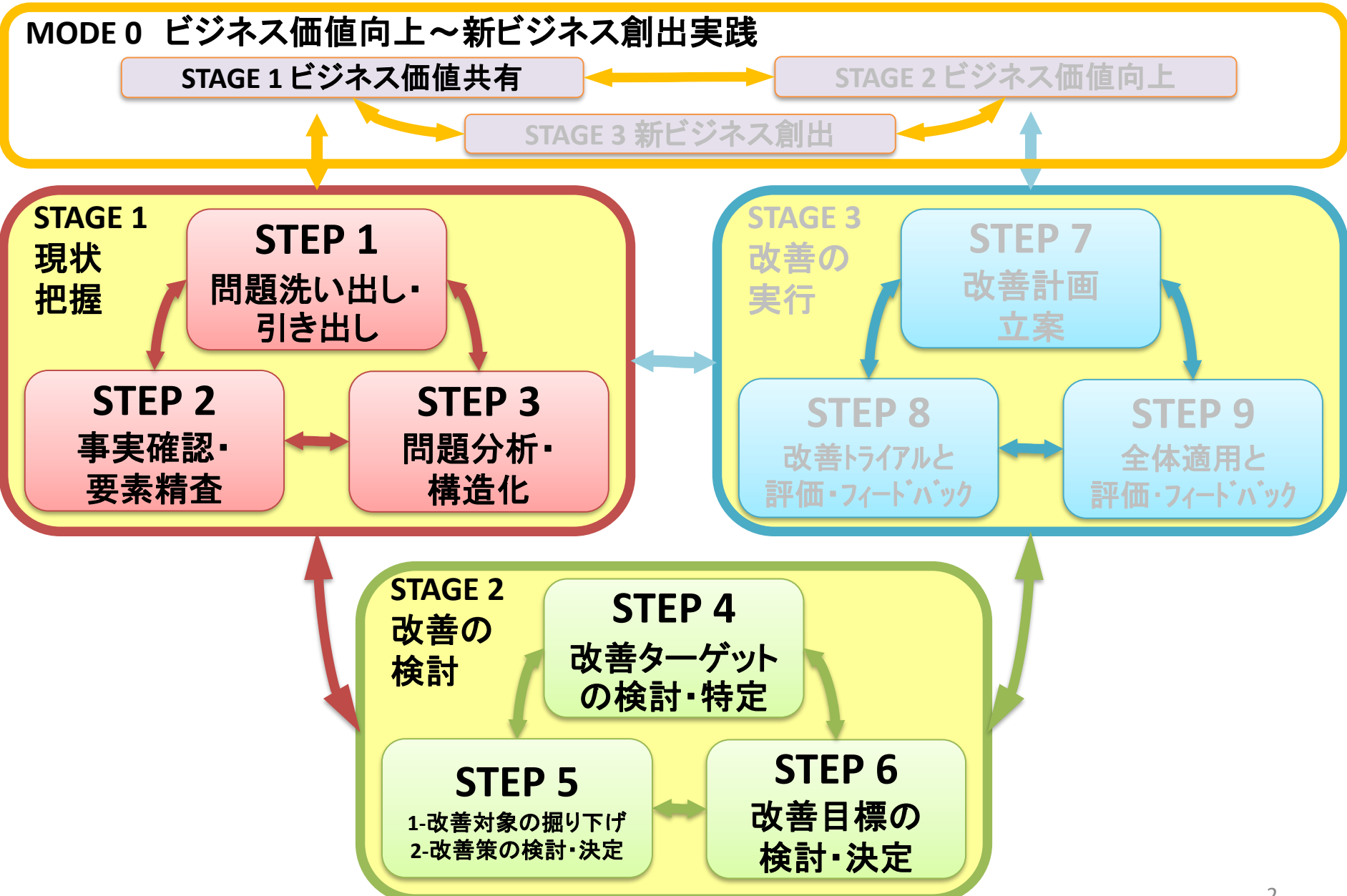
<http://www.software-quasol.com/>

P.000 ←書籍「ソフトウェアプロセス改善手法SaPID入門」の記載頁数を指します。

※ “SaPID”は、株式会社HBAの日本における登録商標です。当スライドではTM,<sup>®</sup>などの表記を省略します。

Copyright © Kenji Adachi@Software Quasol, All Rights Reserved

# 主なワーク対象STEP



# 事前準備：テーマ設定＝SCOPE設定

- テーマ設定により、明らかにする、検討する対象領域をやりわり決めます。
  - 対象領域を決めることで、対象領域の事項に検討や議論を集中し、対象外事項を取り扱わないようにする
- テーマ設定の例
  - ○○開発プロジェクトの問題点
  - テストチームのパフォーマンス向上に向けて

## ① 自らの仕事(業務)と役割

〇〇〇〇〇〇サービス

## ② その顧客

直接顧客～間接顧客

## ③ サービスで顧客に提供している価値

顧客にとっての価値～自分にとっての価値

## ④ 自らの仕事・役割の評価方法

どうなれば成功？ 何で評価する

この対応の過程で自らのミッションやポリシーは何か？を明らかにするのが理想です。

# ボードに配置してワークSTART!!

①○○○○○  
○サービス

②顧客:  
□□□□□  
▽▽▽▽

③提供している  
価値  
□□□

④評価方法:  
□□□□□  
△△△△△

①～④の内容についても随時見直し(見直し前の内容も残し)ながら進めます。

# STEP1: 問題点洗い出し・引き出し

必要に応じてチェックリストを併用すると効果的

納品後に多くの障害が発生

毎日残業 & 休日返上対応が多い

プロジェクトの収支が赤字

要求事項の決定が遅延する

顧客クレーム多発

特にプロジェクト後半に進捗遅延が常態化した

無責任な奴がいる

実装・テストは担当者の経験則に任せている

協力会社Aは危ないのではないか

システムテスト時に大量バグ検出

開発標準や各種判定基準がない

要求事項が何度も変更された

要求事項は記録されないことが多い

役割が長い間固定されている

統合テスト・システムテストで想定していた以上の(はるかに超える)バグが検出された

設計レビューでは有効な指摘が少ない(誤字・脱字・衍字が多い)

実装内容がバラバラ

顧客からシステムへの要求事項がなかなか提示されなかった

設計書はあったり、なかったりする

設計書レビューでは有効な欠陥指摘が少ない

良い・悪いは抜きにして、何が起きているのか、どう感じているのか等をありのまま収集する

# STEP1のポイント

- テーマに対してそれぞれのメンバーが自分で感じている事項を素直に書いて出せるようにする。
  - 感情的に受けとめたモノゴトの背後には、インパクトがある事象や出来事が存在している、という仮説に基づいている。
  - 「テーマに対して」以外の制約を排除し、自由に書いてもらう
    - 問題点、困り事、あったらいいなと思うこと、これ嫌だなーってこと等
    - 変な制約があると間違いたくない意識が働いて書けなくなってしまう
    - 鬼軍曹や高圧的な上司などがいない場を作る等の場づくりが大事
- 付箋がなかなか出ない場合の対処例
  - 実務のライフサイクル(例:開発プロジェクトであれば、着手からリリース、安定稼働確認まで)の当初から終了タイミングまでの実務と想定される発生事象を段階的に思い浮かべてみる。
  - フェーズ単位に発生しそうな事象が列挙されたチェックリストで存在を確認してみる。

# STEP2: 事実確認・要素精査

不適切な感覚・感情論なども手掛かりにして実在する問題・課題を具体的に捉える

納品後に多くの  
障害が発生

15件  
218人時

毎日残業 &  
休日返上対  
応が多い

IT以降  
平均残業315h/m

プロジェクトの  
収支が赤字  
対計画150%

要求事項  
の決定が  
遅延する

対期日45日遅延

実装・テスト  
は担当者の  
経験則に任  
せている

コード・テストケ  
ースのレビューなし

特にプロジェクト後半  
に進捗遅延が常態化

IT以降毎週10日以上  
の遅延が継続

顧客クレーム  
多発

クレーム8件  
4回客先説明

~~無責任な  
奴がいる~~

システムテスト時に  
大量バグ検出

計画132件→実績208件

~~開発標準や各  
種判定基準が  
ない~~

要求事項が何度も  
変更・追加された

記録分のみ  
変更35・追加42

要求事項は  
記録されな  
いことも多い

存在15／対象28

~~協力会社A  
は危ないの  
ではないか~~

実装内容  
がバラバラ

コーディング規約  
違反有が57%

設計書は  
あったり、な  
かったりする

存在13／対象42

役割が長い  
間固定されて  
いる

5年間同一担当

設計書レビュー  
では有効な欠陥  
指摘が少ない

誤字脱字衍字  
系指摘が73%



# 文章表現の原則・禁則

文章表現の原則	内容
事実準拠の原則	事実に即して記載する。
断定・推測区分の原則	推測を含める場合は事実と分けてはっきりと記載する。
個性化の原則	決まり文句や流行語、一般論的な表現を避け、実際に存在する個別事項を記載する。
共通理解の原則	訴えたいことがそのまま関係者に理解されるように表現する。
具体化の原則	抽象的な表現を避け、どのような状態や結果なのかを具体的に把握できるように記載する。
一文一義の原則	一つの文に一つの内容を記載する。
簡潔性の原則	余計な修飾語や冗長な説明を削り落として簡潔に記載する。

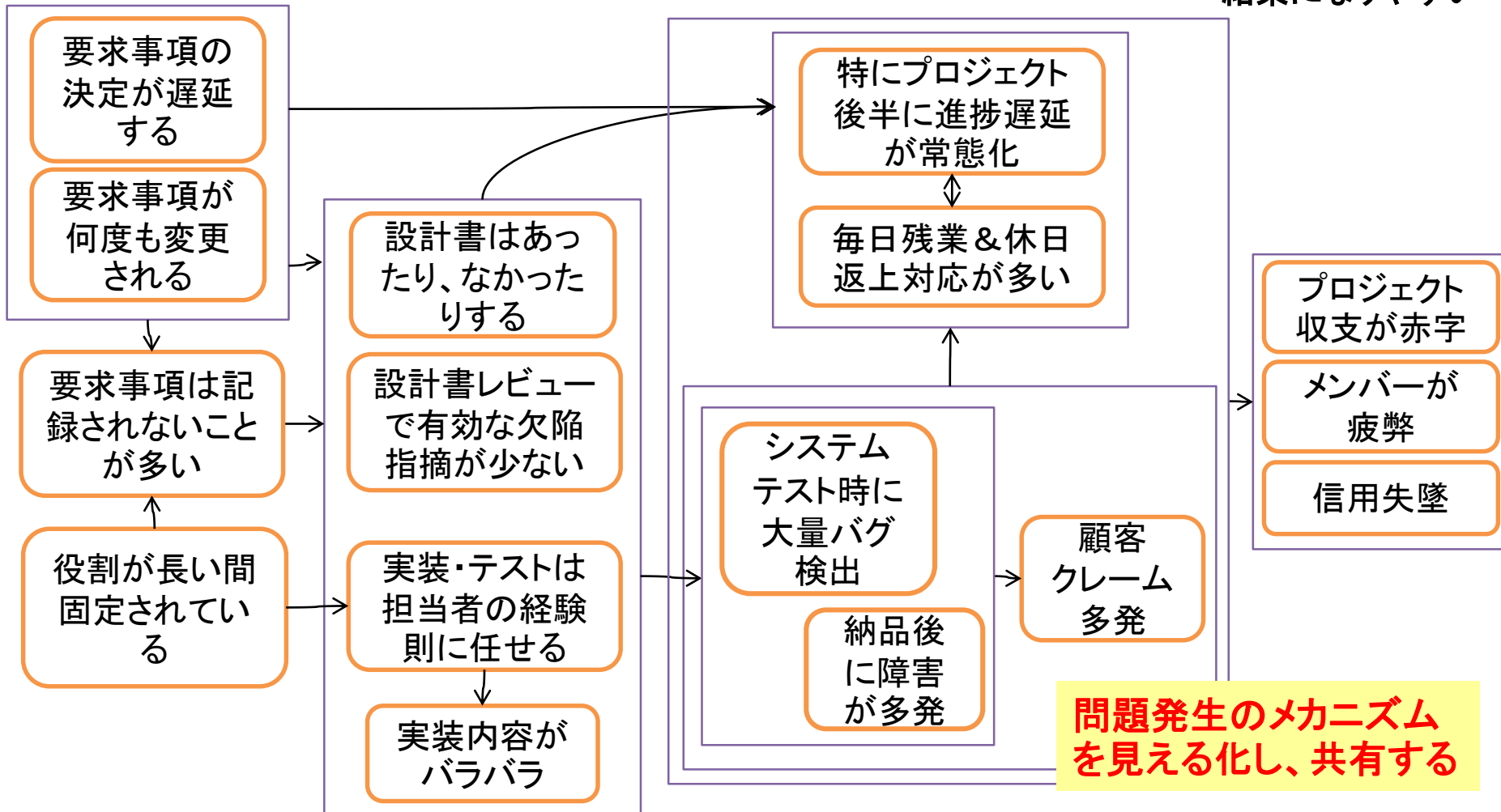
禁則	悪い例	適切な見直し方法
体言止め・紋切型	モラル 計画	内容を具体化して生々しい出来事を記載する。
不足型・不十分型	レビュー不足 テストが不十分	不足・不十分となっている内容を具体的に示す／不足していることで発生している出来事を明確にする。
対策型	□□基準が存在しない	それがなかったために発生している困った出来事や状態を記載する。
疑問型	〇〇スキルに問題あり？	疑問に思った経緯や背景・出来事を具体的に記載する。
断定型・推論型	Aさんはやる気がない 最初からムリな計画なのではないか？	そう考えた経緯や背景・出来事を具体的に記載する。

# STEP3: 問題分析・構造化

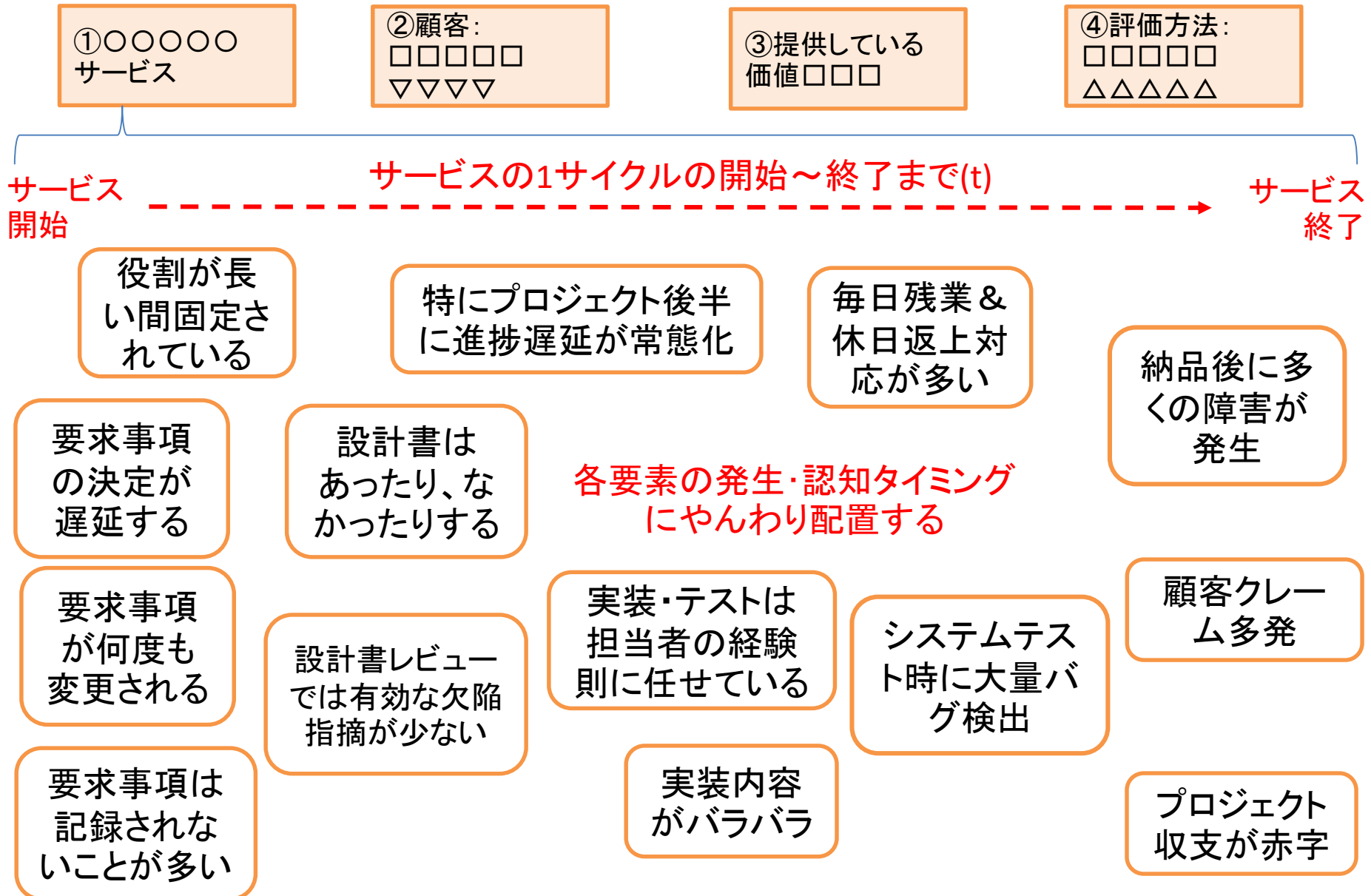
摘要

**【要約】**顧客の言いなりで要求事項が不明確なままプロジェクトを進めているため、途中から仕様変更や進捗遅延が多発し、レビューが追い付かず、テストで大量のバグが検出され、納品後クレームが多発している。

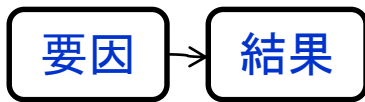
原因 → 結果  
原因が存在すると結果になりやすい



# STEP3-1: 各要素を1サイクル上に配置する



# STEP3-2: 要素間の因果関係を分析する

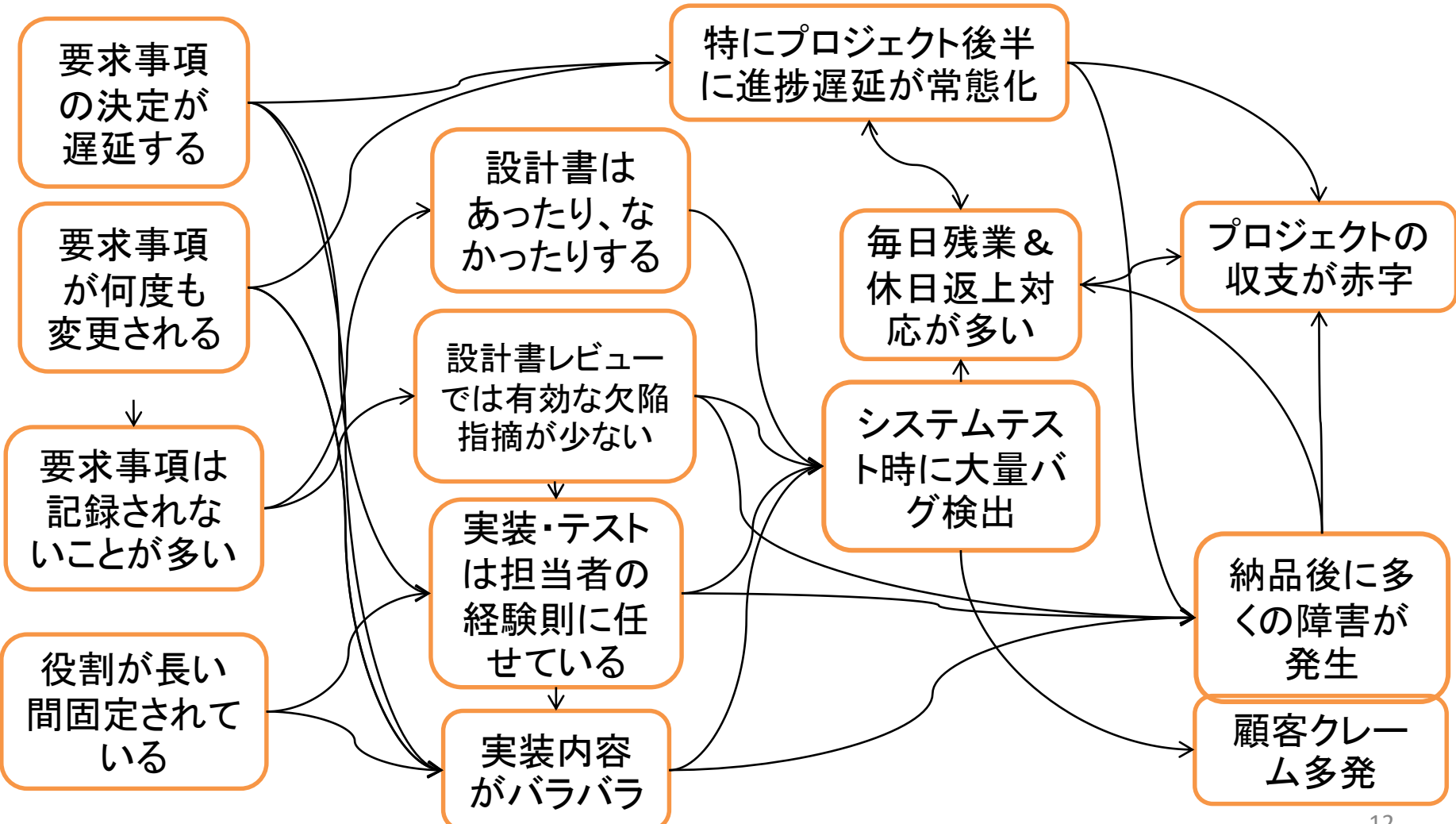


① ○○○○○○  
サービス

② 顧客:  
□□□□□  
▽▽▽▽

③ 提供している  
価値□□□

④ 評価方法:  
□□□□□  
△△△△△



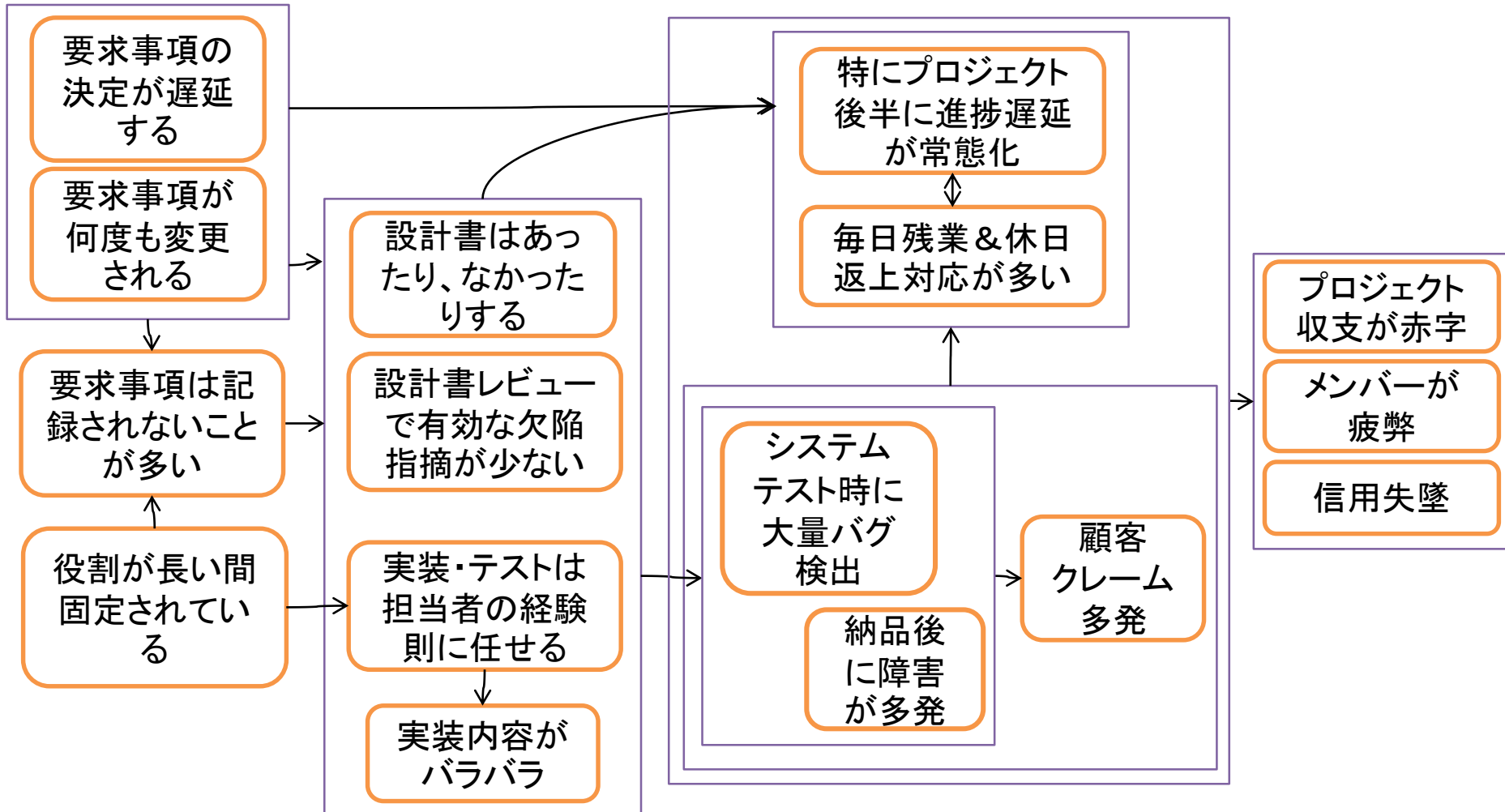
# STEP3-3: 要素間の関係や内容を簡潔に整理する

①○○○○○  
サービス

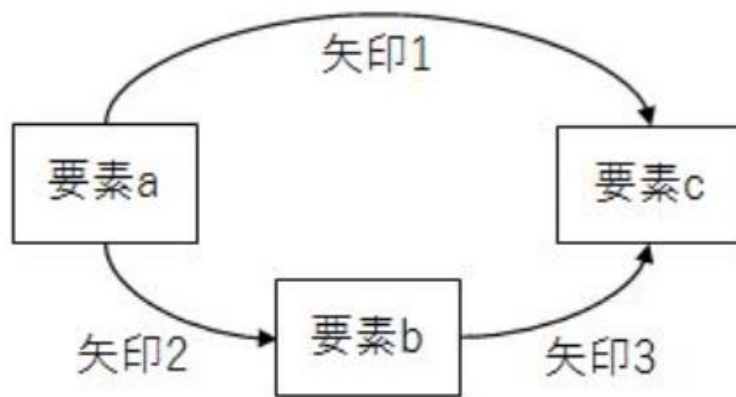
②顧客:  
□□□□□  
▽▽▽▽

③提供している  
価値□□□

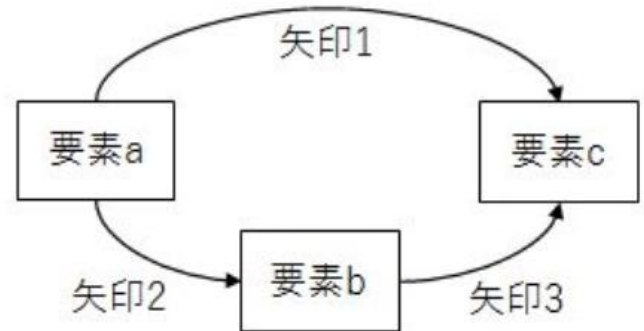
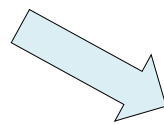
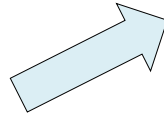
④評価方法:  
□□□□□  
△△△△△



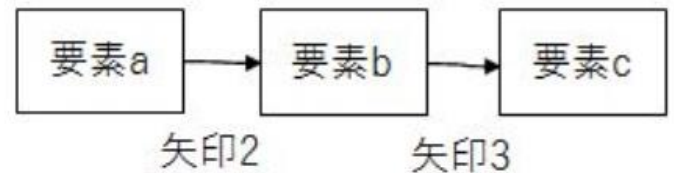
# 関係線(の意味)を正しくつなぐ



ある方が要素間の因果関係を分析して→を付与した結果



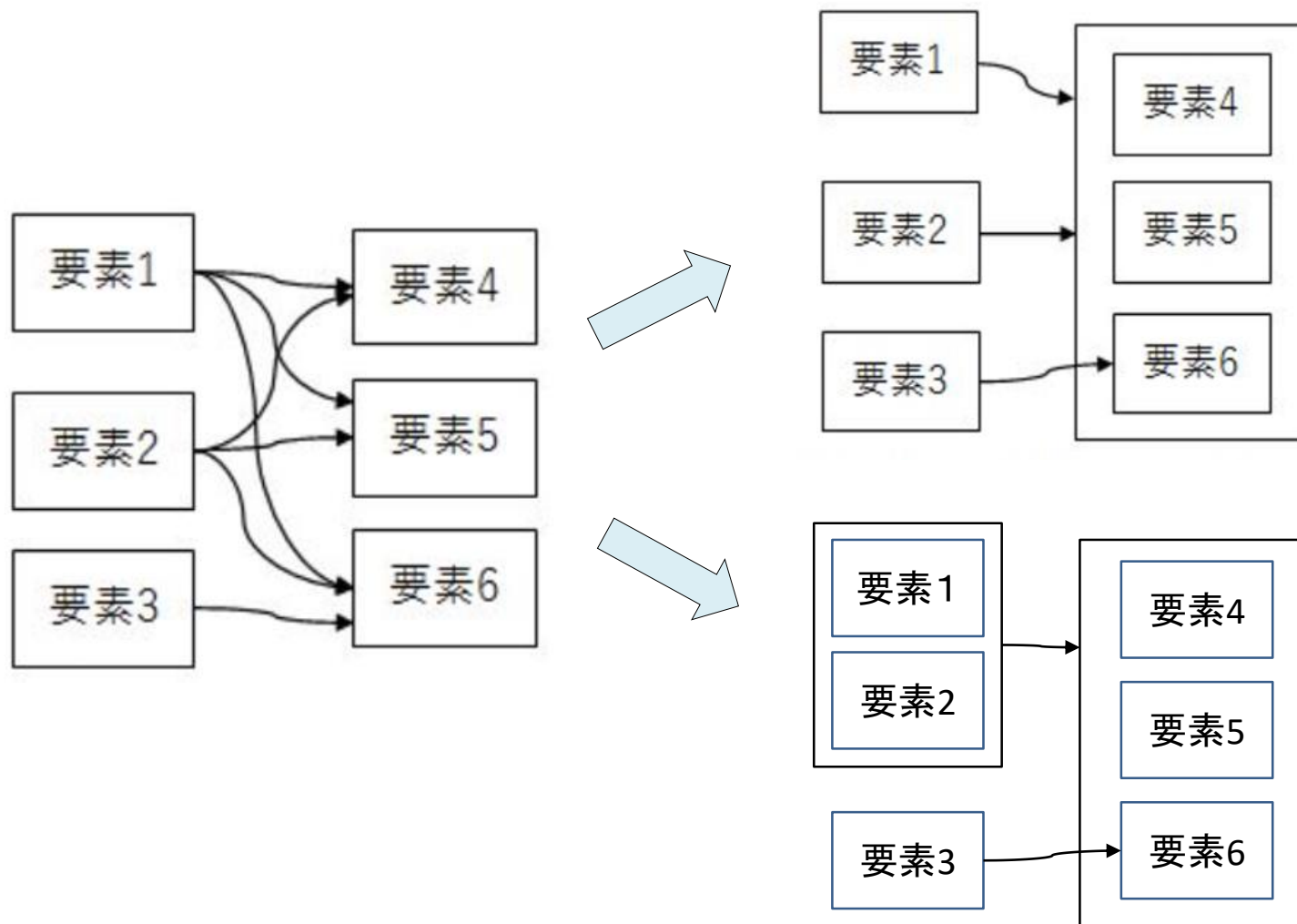
要素aから矢印2・要素b・矢印3を経由して要素cに至る因果関係とは別に、要素aから直接要素cにつながる関係性は存在するとわかった場合(構造図はそのまま)



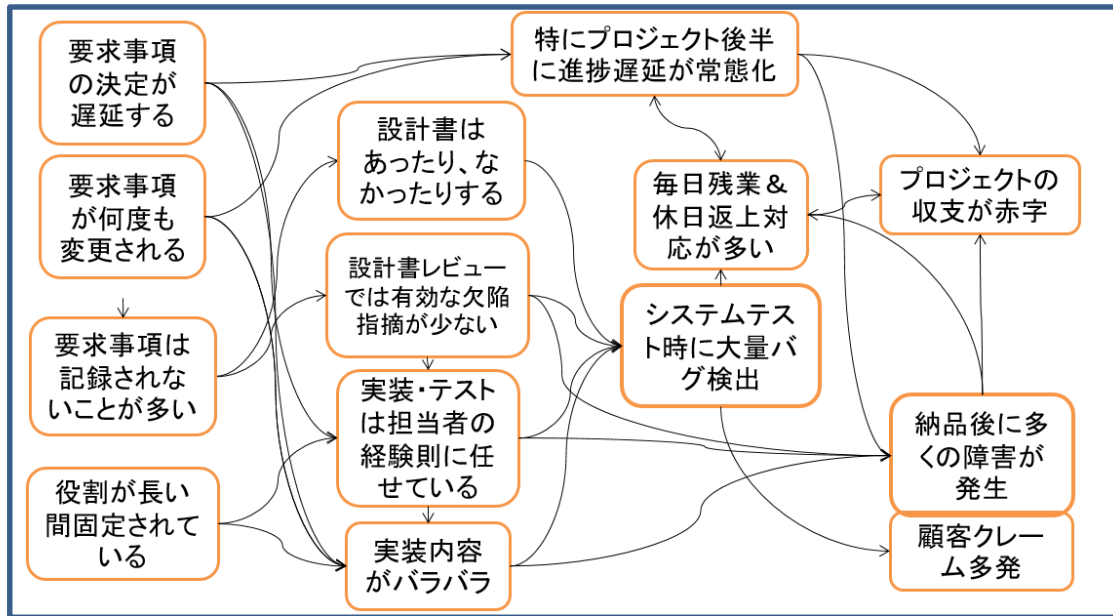
要素aから矢印2・要素b・矢印3を経由して要素cに至る因果関係のみが正しいという結果になった場合(矢印1を削除する)

# 関係線は可能な限り少なく/クロスしないように

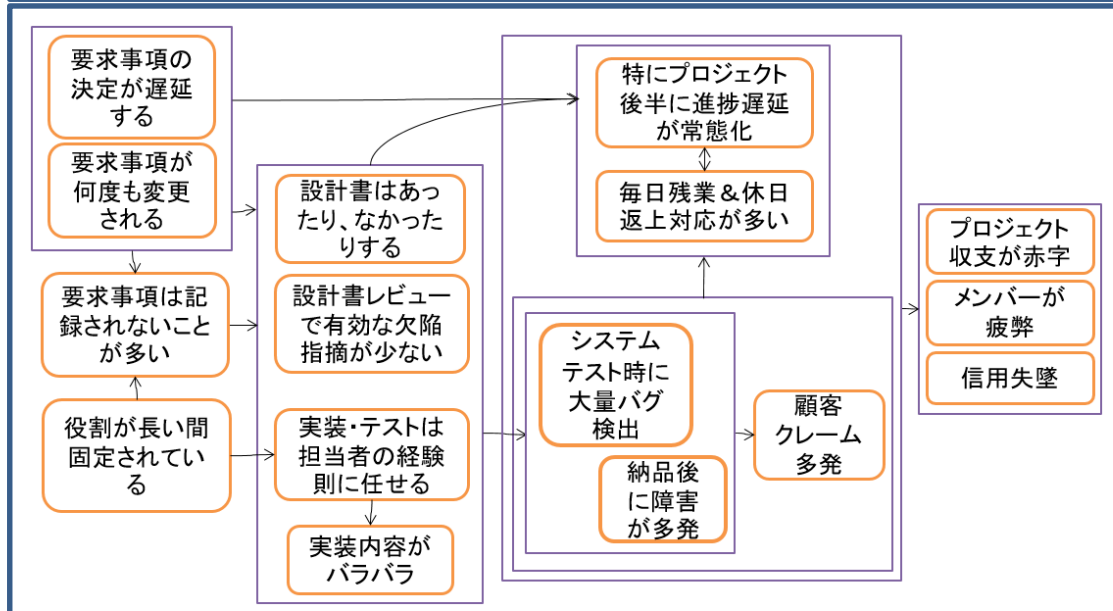
関係線が最小・クロスしない＝わかりやすい＝解きやすい



# どちらがわかりやすいか？



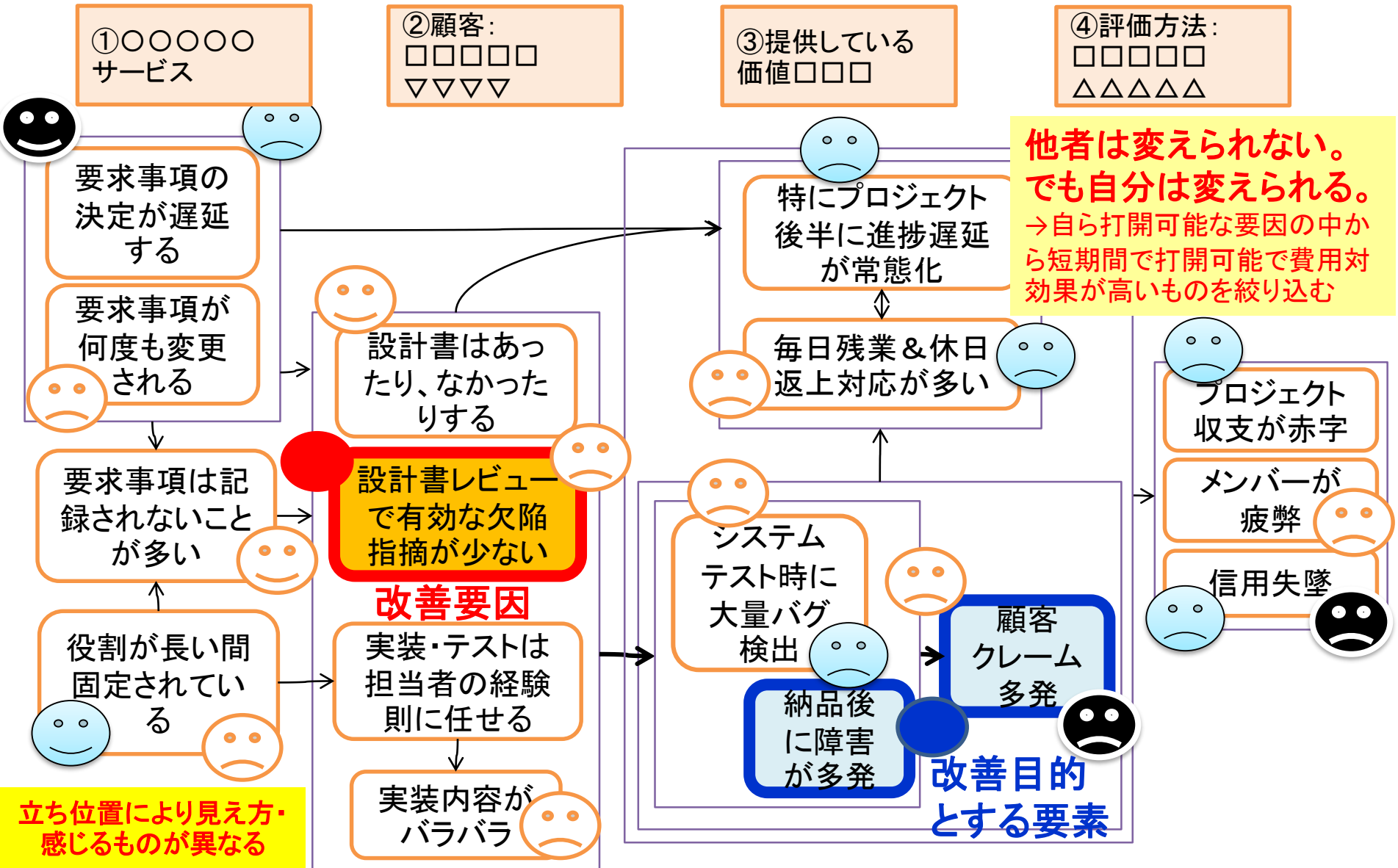
- ・関係線が多い
- ・クロスする線が多い
- ・発生時の重なりが多い



- ・関係線が少ない
- ・クロスする線がない
- ・発生時の重なりが少ない



# STEP4: 改善ターゲット検討・特定



# STEP5-1: 改善対象の掘り下げ

設計書レビュー  
では有効な欠陥  
指摘が少ない

上位問題構造図の  
改善要因

改善目的要素の現状

レビュー  
チェックリスト  
が抽象的

レビュー観点  
はレビューア  
の解釈で実施

要求事項が記  
述されていない  
場合がある

誤字・脱字・  
衍字が多い

仕様書の記述  
内容は担当者  
ごとにまちまち

有識者が多忙  
なためほとんど  
参加できない

有効な欠陥指  
摘が少ない

後工程でレビュー  
の見逃し／修正  
漏れ・ミスを原因  
とする手戻りが発  
生している

- ②結果内訳  
の掘り下げ
- テストや導入後に検出される上位3事項
- ①条件ELSE側不備
  - ②エラー処理不備
  - ③文字数Max、指定文字以外未考慮

## ①レビュープロセスの 掘り下げ

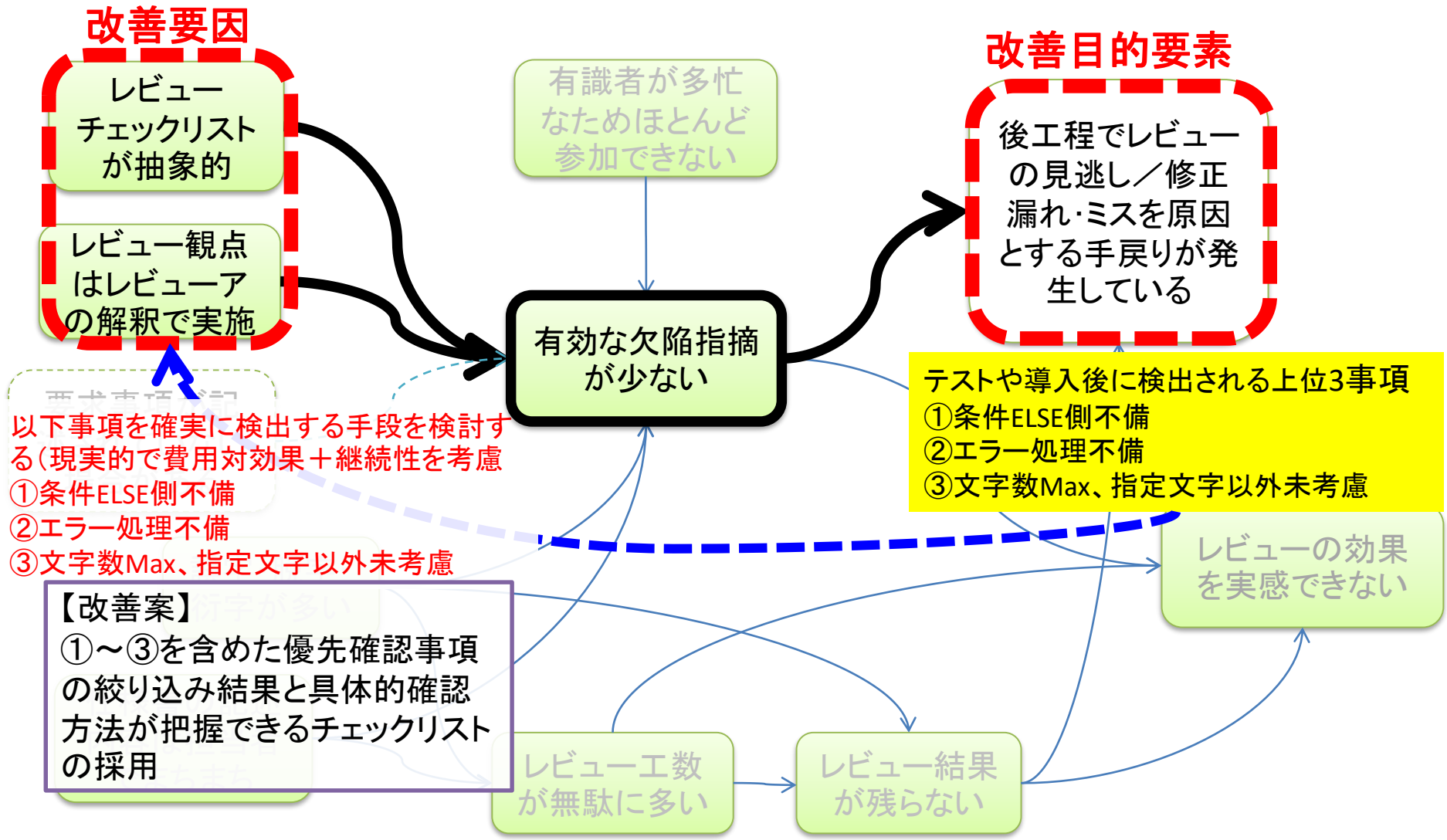
レビュー工数  
が無駄に多い

レビュー結果  
が残らない

レビューの効果  
を実感できない

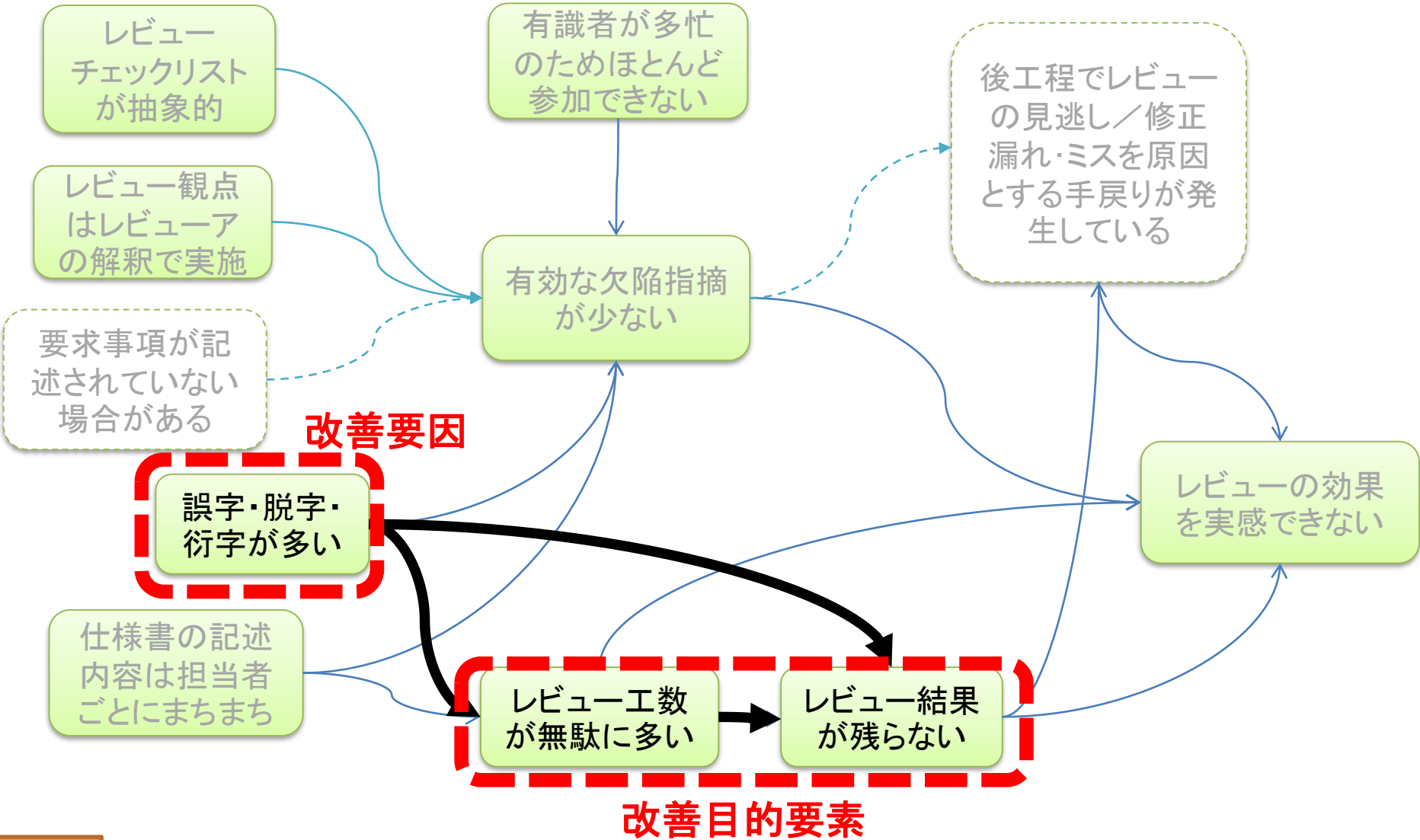
# STEP5-2: 改善手段検討・決定

レビューの本質的な改善に着手する例



# STEP5-2: 改善手段検討・決定 (別解)

先行して無駄な工数を削減し、  
余裕を確保してから本質的な  
改善に着手する例



# STEP6: 改善目標の検討・決定

改善目標も個人・チーム・組織の状況に応じて段階的に高度化することが重要

## 改善要因

設計書レビューは実施していない場合が多い

テスト時に大量バグが検出され、想定以上の工数と期間がかかる

### 施策系改善目標

- 例1: レビュー実施
- 例2: レビュー実施率

## 改善要因

設計書レビューは実施していない場合が多い

## 改善目的要素

テスト時に大量バグが検出され、想定以上の工数と期間がかかる

### 施策系改善目標

- 例1: レビュー実施
- 例2: レビュー実施率

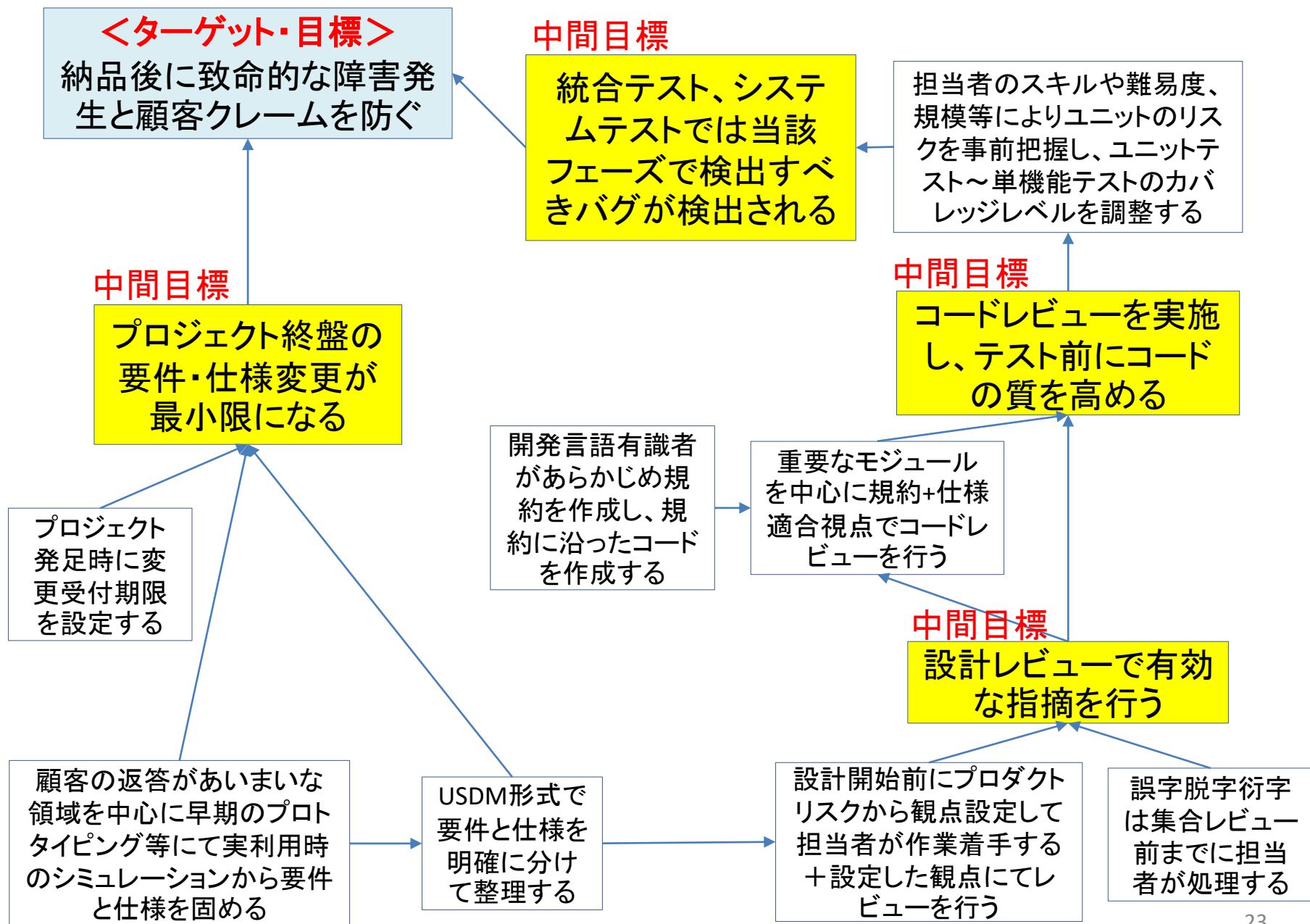
### 成果系改善目標

- 例1: 規模あたりのテスト時バグ検出量減少
- 例2: 規模あたりのテスト工数減少
- 例3: 規模あたりのテスト期間の短縮

# アンビシャスターゲットツリー 1/2

①ターゲット: 納品後に致命的な障害発生と顧客クレームを防ぐ		
②障害	③中間目標	④行動
統合テスト、システムテストで大量にバグが検出される	統合テスト、システムテストでは当該フェーズで検出すべきバグが検出される	<ul style="list-style-type: none"> <li>・担当者のスキルや難易度、規模等によりユニットのリスクを事前把握し、ユニットテスト～単機能テストのカバレッジレベルを調整する</li> </ul>
コードレビューが実施されない	コードレビューを実施し、テスト前にコードの質を高める	<ul style="list-style-type: none"> <li>・開発言語有識者があらかじめ規約を作成し、規約に沿ったコードを作成する</li> <li>・重要なモジュールを中心に規約+仕様適合視点でコードレビューを行う</li> </ul>
設計レビューで有効な指摘ができていない	設計レビューで有効な指摘を行う	<ul style="list-style-type: none"> <li>・誤字脱字衍字は集合レビュー前までに担当者が処理する</li> <li>・設計開始前にプロダクトリスクから観点設定して担当者が作業着手する+設定した観点にてレビューを行う</li> </ul>
プロジェクト終盤になって要件や仕様がゴロゴロ変わる	プロジェクト終盤の要件・仕様変更が最小限になる	<ul style="list-style-type: none"> <li>・USDM形式で要件と仕様を明確に分けて整理する</li> <li>・プロジェクト発足時に変更受付期限を設定する</li> <li>・顧客の返答があいまいな領域を中心に早期のプロトタイピング等にて実利用時のシミュレーションから要件と仕様を固める</li> </ul>

# アンビシャスターゲットツリー 2/2

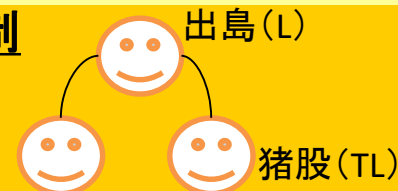


# STEP7: 改善計画立案

改善計画立案段階で初めて様式に書き始めるのは形式対応になりやすいので注意

チーム名: やってみなはれGr

体制



テーマ: ○●運営の効果向上 & 効率化を目指す  
□■情報作成～共有方法の改善

1. 業務概要	△▲部の統括運営・管理を行っています。 部署運営の効果・効率に直結する運営の最適化を目指しています。
2. 着目した問題	・内部、外部関係者間で共有、活用する□■情報の作成・修正・共有に手間と時間がかかっている。共有情報が有効活用されていない。部署内で共有情報活用が進まない理由に情報提供のタイムリーさの欠如が存在している。
3. 要因と改善方法 ※参照: 5. 改善前後の運営イメージ	<p>要因1: 共有情報の作成を元ネタ情報FIX後に手作業で行い、その後再確認、共有、連絡発信の流れになっている。</p> <p>→改善方法1: 元ネタ情報がFIXするまでの過程で同時並行で作成・関係者確認を行う。</p> <p>要因2: 共有情報が紙媒体のため、ロケーションが離れるとアクセスできない。</p> <p>→改善方法2: 情報を格納する共有フォルダを構築し、そこにテキストコピー可能なPDFで格納し、各関係者がアクセスすることにする。</p>

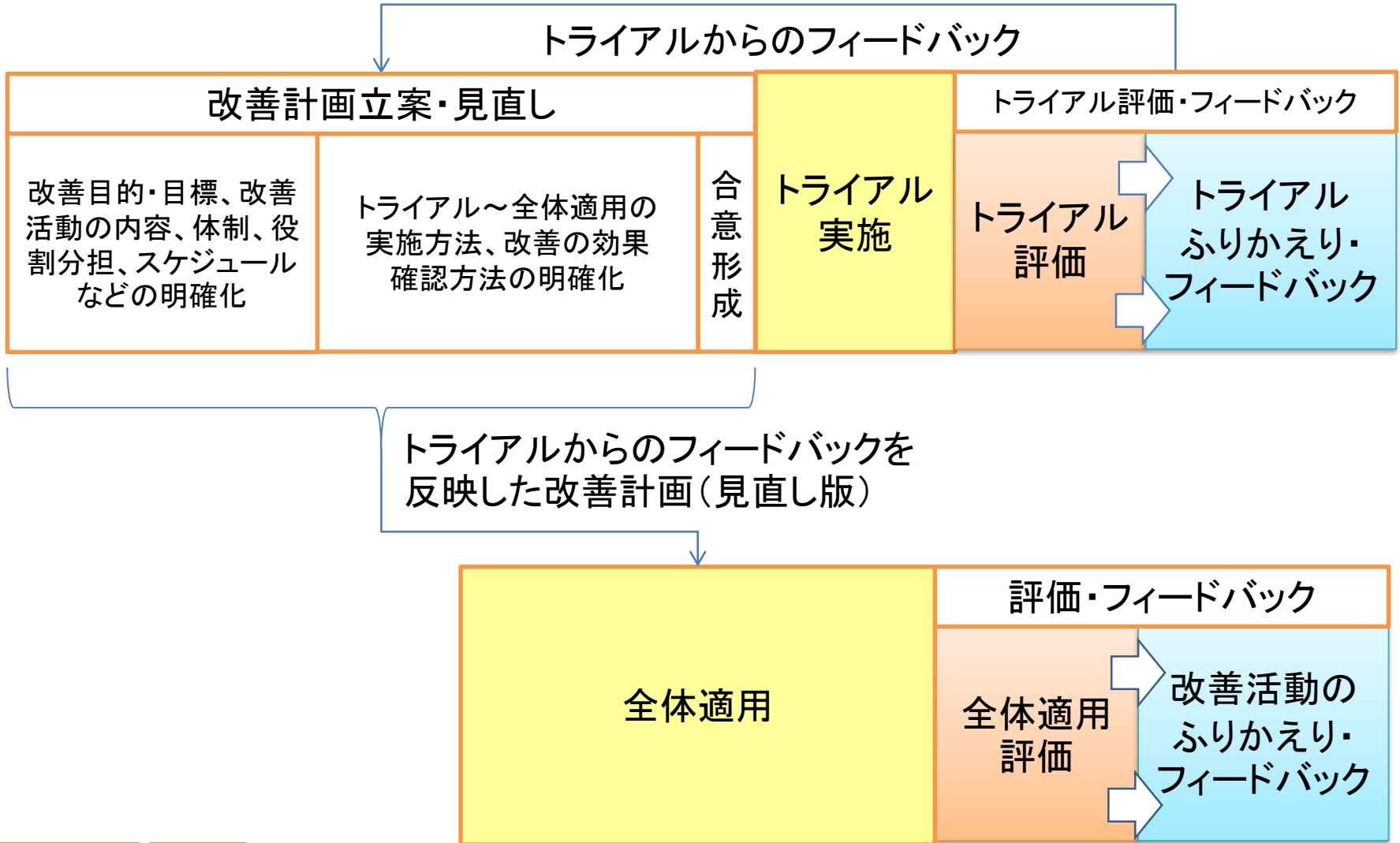
4. 改善期待効果 と改善目標	評価指標	現状 直近2カ月間、対象9回の平均	改善目標	実績 (計画時空欄)
	①情報作成～共有・関係者連絡までの期間(日数)	5～6日間	0日間(当日完了)	0日間 1.5時間程度
②手戻り工数(人時)	・問合せ14件 ・対応工数7.5人時	0件、0人時に近づける	2件 3分程度	
③共有情報活用率	39%	80%以上	82%	

様式記載例



# STEP8: 改善トライアルと評価・フィードバック

# STEP9: 全体適用と評価・フィードバック



# STEP9: 全体適用→評価・ふりかえり&フィードバック

全体適用

評価・ふりかえり  
によるフィードバック

フィードバック事項  
の反映→改善継続

ふりかえり結果例

EP: 良い点・まずはよかったことからスタート

(俺、ナイスプレー！)

<成果>  
A.O.部員化に伴って発注書の手が早いという評判が上がるのは悪い、(B)部員化に伴って、ロコナイスプレー！  
C.部員化に伴って、発注書の手が早いという評判が上がるのは悪い、(B)部員化に伴って、ロコナイスプレー！  
D.部員化に伴って、発注書の手が早いという評判が上がるのは悪い、(B)部員化に伴って、ロコナイスプレー！

ふりかえり結果例

Z-PROBLEM: 向題事項・改善必要事項→重要事項はTRYで対策

言いにくいことでもスバット

大事なことは強調

分類してまとめる

感じたことも忘れずに

取り組み内容

活動内容のレベルアップを求め!

納得して感じること

ふりかえり結果(例)

**Keep (K)**  
K1: 皆が能動的に動いたので管理能力向上目標ノバーあたがリスク識別事項は全員達成! すこいぞ!(あ)

**Problem (P)**  
P1: 作成者側の言葉で記載されていたため、受け手側で理解できず質問が相次いだ。(い)

**Try (T)**  
T1: 受け手でも分かる言葉で表現する  
T2: 成果物レビューチェック欄点Keywordに“作成者側のローカル言葉”を追加し、レビュー時に確認する

その場ですぐに次の活動準備

業務プロセスKnow-Howリスト

Process	(T) リスク対策	(K) リスク (P)
管理	H24管理能力向上目標ノバー	H23能力向上目標全員達成!
設計・実装	受け手が分かる言葉で表現する	作成者側の言葉で記載されていたため、受け手側で理解できず質問が相次いだ。
Review	Keyword12 “作成者側のローカル言葉”	

次の活動時にそのまま使う