2024/2/13　日科技連オープンセミナー
ODC分析x SaPIDにより的確で実効性のある開発改善策策定・実行を目指す

ODC分析×SaPID

チームパフォーマンス向上のカギ

Software Quasol
安達 賢二
https://www.softwarequasol.com/
adachi@hba.co.jp

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
http://www.softwarequasol.com/
株式会社Levii　共創ファシリテーター
https://levii.co.jp/about/

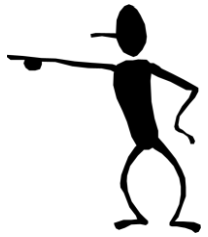
Twitter（X）

【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
ASTER研究事業「レビュー体系化」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33-39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

コンテンツ

- SaPIDとは？
- ODC分析×SaPIDとは？
- ODC分析を活用したチームパフォーマンス改善
  【ODC→SaPID連携】
- ODC分析が実践できるようにする
  【SaPID→ODC連携】
- チームパフォーマンス向上のポイント



今日はここを中心に
お伝えします

SaPIDとは？

自律・自己組織化を促進する価値共創プログラム：SaPID

Systems **a**nalysis / **S**ystems **a**pproach based **P**rocess **I**mprovement metho**D**

- 当事者自らが対象のメカニズムをシステムとして構造的に捉え、最も効果的で現実的な箇所、あるいは新しい価値創出に有効な施策・対策をうち、成果をあげていくための手法。＝**システム思考**
- チームや関係者、組織のメンバーが参画し、**デザイン思考**を併用して一緒に成長する、そして新しい価値を共創する方法を併せ持つ。

共創プログラム

チームや組織に存在する関係者全員が参画して、相互に理解しあい、それぞれの特徴や得手・不得手を融合して同じ目的を一緒によりよく達成することを促進するアプローチ群。

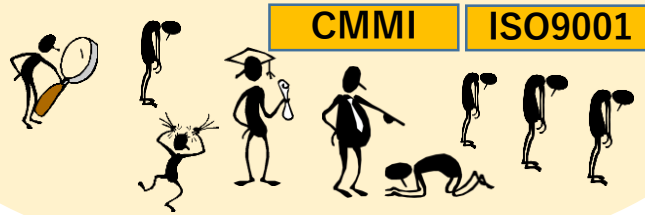
シャープ伝説のエンジニア　故佐々木正氏（シャープ元副社長）の言葉

「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」

SaPIDが解決したい世界（主な用途）

～システム思考×デザイン思考実践により依存から能動・自律へ

プロセスモデルベース　トップダウン偏重
やらせる／やらされるプロセス改善の問題

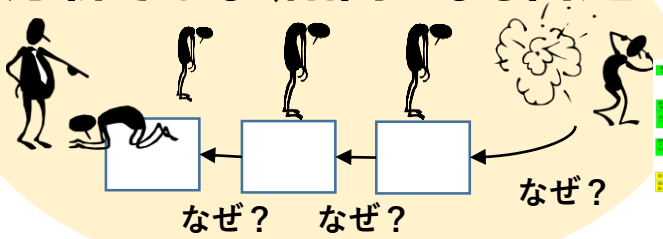


管理統制ばかりでぶら下がりメンバーによる問題
駆動/強制・指示型プロジェクト運営

問題構造図
（因果関係モデル）

システム思考 ✖ デザイン思考

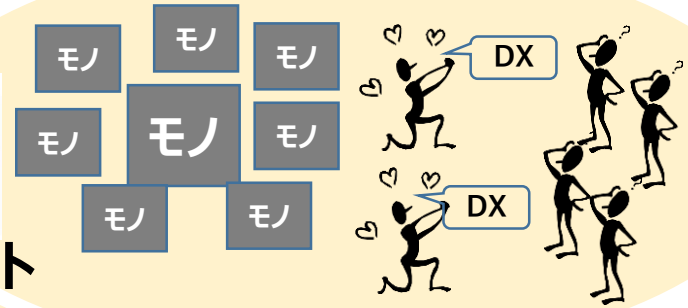
大問題しか分析しない
分析ではなく詰問になる問題



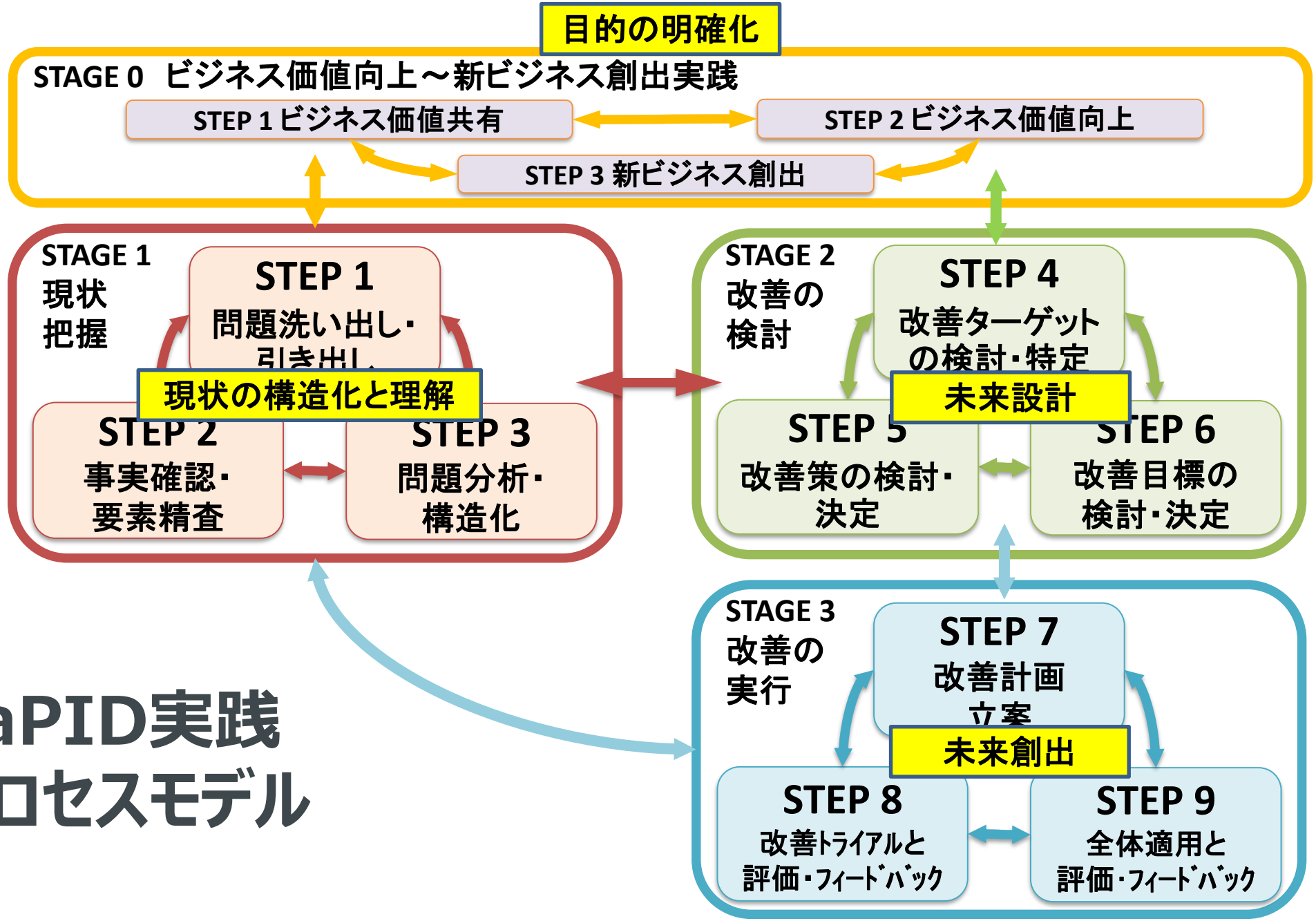
未来予想図

価値連鎖モデル

モノありき／コトづくりなしのエセDX実践

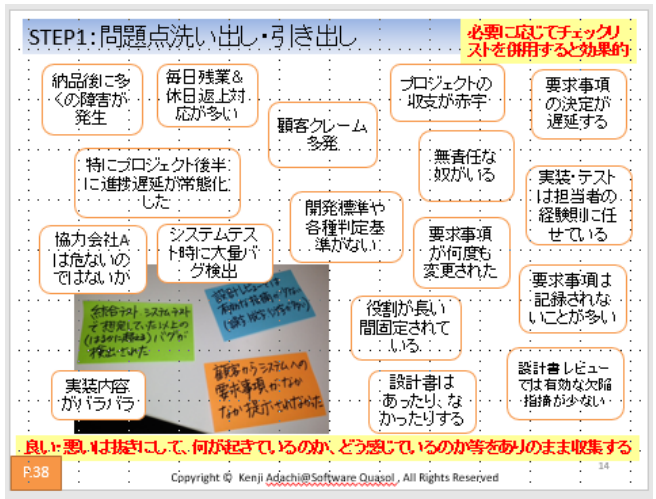


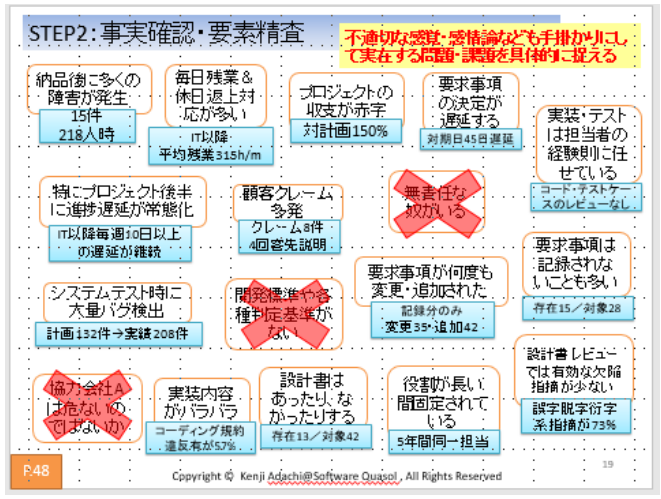
突然要件や仕様から始まる受け身の開発プロジェクト

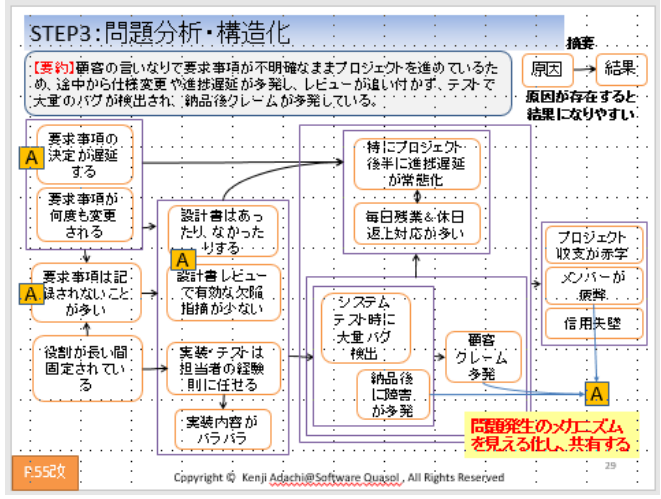


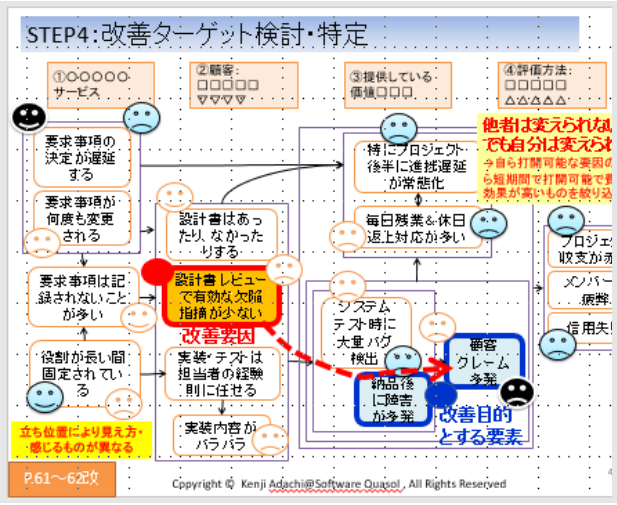
SaPID実践プロセスモデル

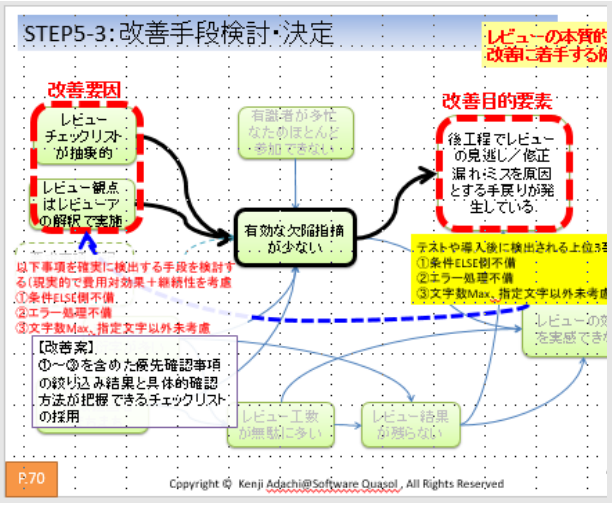
SaPID標準アプローチ

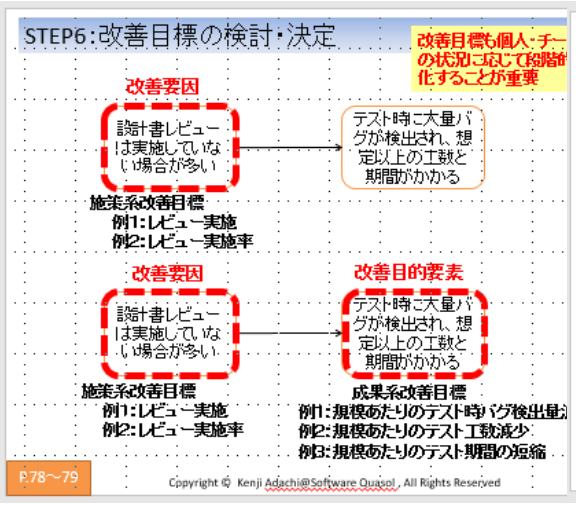


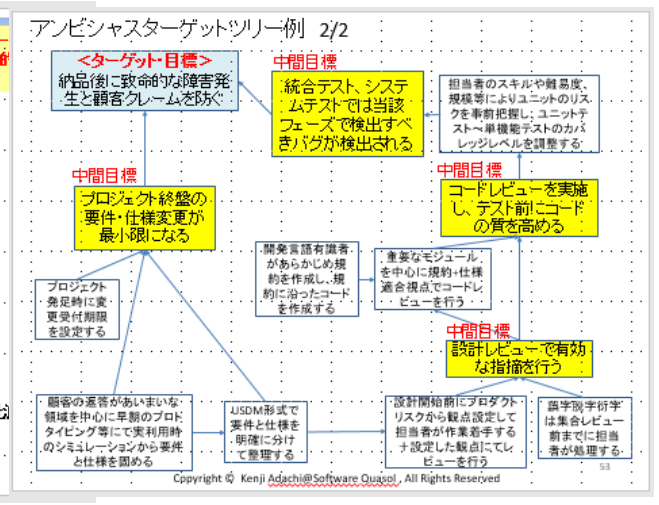












これまでのSaPID関連事例発表

SPI Japan 2007：現場の様々な事実情報分析に基づく現実的な改善アプローチのご紹介－システムズアプローチを活用した改善実践事例－ http://www.jaspic.org/event/2007/SpiJapan/2A3.pdf

SPI Japan 2011：ふりかえり実践方法の変遷による業務運営プロセスと成果の改善

http://www.jaspic.org/event/2011/SPIJapan/session3B/3B4_ID008.pdf

SPI Japan 2012：システムズアプローチによる問題発生構造分析とPFD（Process Flow Diagram）を用いたプロセス改善

http://www.jaspic.org/event/2012/SPIJapan/session3A/3A3_ID009.pdf

SPI Japan 2012 【最優秀賞受賞】：システムズアプローチに基づくプロセス改善メソッド：SaPIDが意図するコト～プロセスモデルをより有効活用するために／そして現場の自律改善運営を促進するために～

http://www.jaspic.org/event/2012/SPIJapan/session3A/3A4_ID023.pdf

SPI Japan 2013 【実行委員長賞受賞】：SaPID実践事例より～改善推進役がやるべきこと／やってはいけないこと 現場が自らの一歩を踏み出すために http://www.jaspic.org/event/2013/SPIJapan/session2B/2B3_ID011.pdf

SS2013 【最優秀発表賞受賞】：「プロセスアセスメント結果の現実的・効果的活用方法の提案」

http://sea.jp/ss2013/accepted_papers.html#s3

派生開発カンファレンス2013：問題構造分析とPFDの併用による現実的・段階的な改善実践方法の提案～PFDを使いこなす能力を確実に身に着けるために https://affordd.jp/libraries/xddp2013-p8/

SQiP2014：「プロジェクト運営と改善実践の連携・一体化」～プロジェクトマネジメントにおけるSaPIDシステムズアプローチ活用事例

https://www.juse.jp/sqip/symposium/archive/2014/day1/files/happyou_B1-3.pdf

SPI Japan2015 【わくわく賞受賞】：「自律型プロジェクトチームへの変革アプローチ事例」～チームの価値観変容を重視し、問題モデリングを活用したSaPID流プロセス改善アプローチ～ http://www.jaspic.org/event/2015/SPIJapan/session3C/3C-3_ID012.pdf

JaSST2016東京 【ベストスピーカー賞受賞】：「レビューの目的・観点設定の効果と課題」

Slideshare：http://www.slideshare.net/AdachiKenji/ss-59510938

SaPIDの派生先例

IPA/SEC プロセス改善研究部会

[SEC BOOKS
プロセス改善ナビゲーションガイド
～自律改善編～]

SPINA^{3}CH自律改善メソッド

WG24 小規模企業向けソフトウェアライフサイクル ISO/IEC 29110 小規模企業向けソフトウェアライフサイクルのプロファイル

ISO/IEC TR 29110-3-4：2015
Systems and software engineering -- Lifecycle profiles for Very Small Entities (VSEs)
-- Part 3-4: Autonomy-based improvement method （自律に基づく手法）


TECHNICAL REPORT
ISO/IEC TR 29110-3-1
First edition 2015-10-15
Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs) —
Part 3-1: Assessment guide
Ingénierie des systèmes et du logiciel — Profils de cycle de vie pour très petits organismes (TPO) —
Partie 3-1: Guide d'évaluation

SPI Japan2017 【特別賞受賞】:「自分事化影響要因に着目した中期経営計画立案・展開への共創アプローチ[現状分析～計画立案編]」
http://www.jaspic.org/event/2017/SPIJapan/session3B/3B1_ID003.pdf
JaSST2018東京 【ベストスピーカー賞受賞】:「TPI Nextを活用したチームメンバーの問題意識から始めるテストプロセス改善」
http://bit.ly/2FLtpHU
SS2018札幌 「リスク構造化を用いたリスクマネジメント手法の提案と効果分析」～「未来予想図」を用いたリスクマネジメントPDCAサイクル～
https://www.slideshare.net/AdachiKenji/ss2018-sapidtocpresentation
JaSST2019北陸 招待講演 「そのレビュー、大丈夫ですか？ ～現状レビューの問題発見・解決」
http://www.jasst.jp/symposium/jasst19hokuriku/pdf/S2-1.pdf
http://www.jasst.jp/symposium/jasst19hokuriku/pdf/S2-2.pdf
SPI Japan2019 【特別賞受賞】:静的×動的プロセス改善の実践と課題 共通性×相違性～見つけ方とつなぎ方
http://www.jaspic.org/event/2019/SPIJapan/session2A/2A2_ID008.pdf
SS2020 「リスク構造を読み解いてアプローチするFRI(Factor-Risk-Influence)モデルによるリスク構造の見える化」
https://www.sea.jp/ss2020/download/SS2020-10(slide).pdf
SPI Japan2020 「FRI(Factor-Risk-Influence)モデルによるリスク構造の見える化」
http://www.jaspic.org/event/2020/SPIJapan/session6/6-3_ID008.pdf
JaSST2021東京 チュートリアル 「価値につながる要件・仕様からテストを考える」
http://jasst.jp/symposium/jasst21tokyo/details.html
JaSST2021四国 招待講演「レビューのキキメ～Part2「レビューの成功要因」はどの程度のキキメがあるの？」
Slideshare：https://speakerdeck.com/kitanosirokuma/rebiyufalsecheng-gong-yao-yin-hadofalsecheng-du-falsekikimegaarufalse
JaSST 2023東京 ワークショップ 「現状モデリングによるテスト対象の背景分析ワーク～未来を創る価値のタネを「現状」から見出す～」
https://www.jasst.jp/symposium/jasst23tokyo/details.html#E3

SaPIDの用途別活用例

SaPID標準アプローチ：〇
SaPID応用アプローチ：★

組織・チーム・個人による活用

〇：個人パフォーマンス改善：SaPID Bootcampにて参加者がそれぞれ実践
★：チームパフォーマンス改善：SPI Japan2013＋SPI Japan2015事例発表 →スライド60-64
〇：組織的生産性向上施策展開：SPI Japan2012事例発表
〇：組織中期事業計画立案：SPI Japan2017事例発表

プロジェクトマネジメントへの活用

〇：プロジェクトリスクのモデル化(FRIモデル)：SS2020事例発表　SPI Japan2020事例発表
〇：未来予想図を活用したリスクマネジメント実践：SS2018事例発表

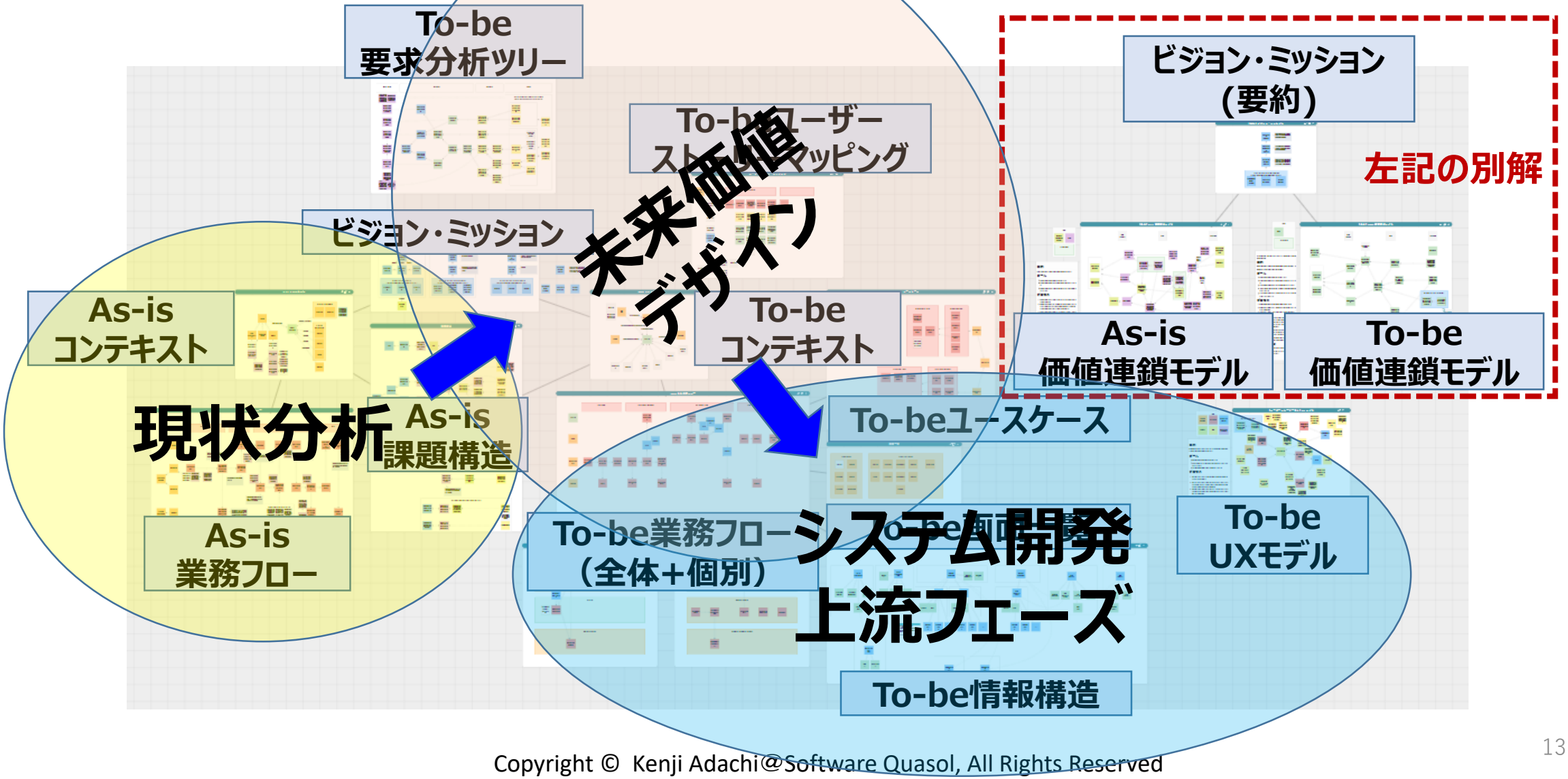
システム開発への活用

〇：SaPID→RDRA連携：JaSST2021東京チュートリアル（資料）
〇：ソフトウェアレビュー改善：JaSST2016東京事例発表　他
〇：ソフトウェアテスト改善：JaSST2018東京事例発表
〇：ロスコンプロジェクト分析：SPI Japan2019事例発表中の一部項目として紹介
★：ODC→SaPID連携：2024/2/13 ODC分析×SaPIDオープンセミナー**（今回）**

DXへの活用

〇：SaPID→RDRA連携：JaSST2021東京チュートリアル（資料）
〇：現状分析→未来価値設計～コトづくり：JaSST2023東京ワークショップ

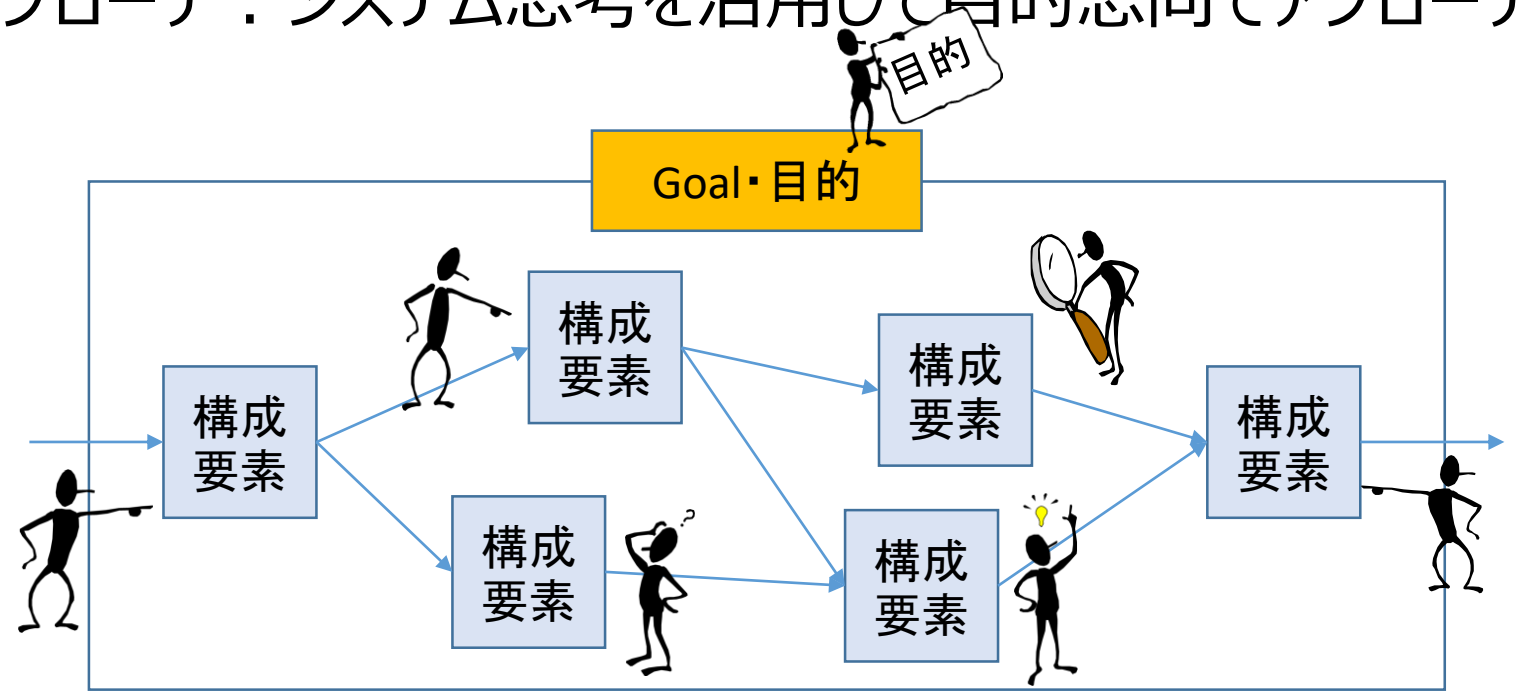
SaPID-DXアプローチ成果物の全体像



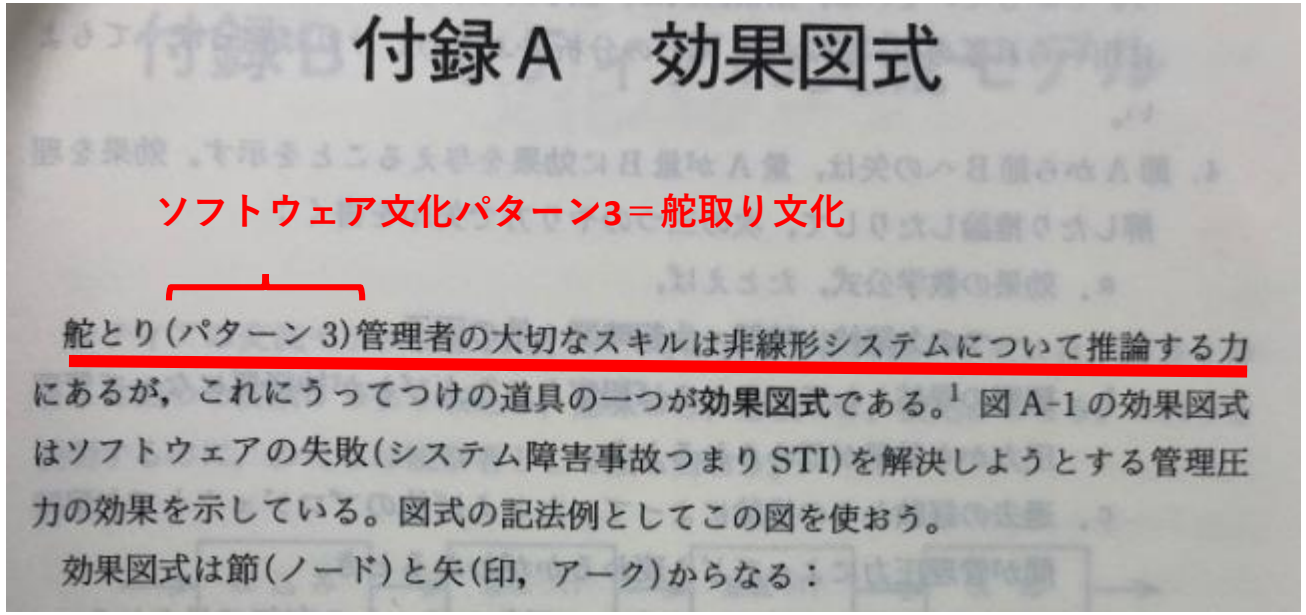
システム思考＝目的志向・指向

システムとは？
目的を達成するために、 1つまはた複数の構成要素が相互に連携・作用するしくみ全体のこと

- システム思考：対象をシステムとみなして明確化する＝Systemic（俯瞰的）×Systematic（系統的）に思考する（ロジカルシンキングを含む）
- システムズアプローチ：システム思考を活用して目的志向でアプローチする



舵取り文化を支える
非線形システムを推論する力＝”システム思考”



付録A　効果図式

ソフトウェア文化パターン3＝舵取り文化

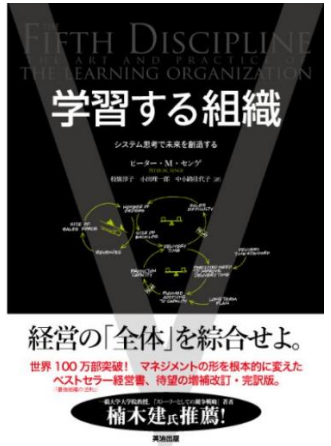
舵とり(パターン3)管理者の大切なスキルは非線形システムについて推論する力にあるが，これにうってつけの道具の一つが効果図式である。[1] 図A-1の効果図式はソフトウェアの失敗(システム障害事故つまりSTI)を解決しようとする管理圧力の効果を示している。図式の記法例としてこの図を使おう。

効果図式は節(ノード)と矢(印，アーク)からなる：

[ソフトウェア文化パターン]
パターン5：適合文化
パターン4：予知文化
パターン3：舵取り文化
パターン2：慣習的文化
パターン1：可変の文化
パターン0：無意識の文化

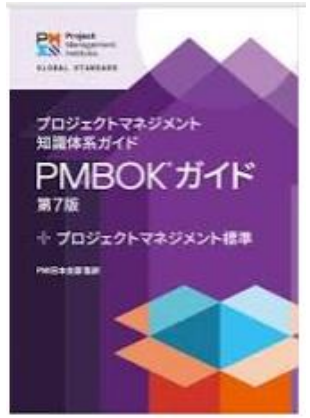
ソフトウェア文化パターン：G.M.ワインバーグ氏が工学的プロセスをどのレベルで実践する組織なのかを「管理の心的態度」で分類したもの
類似の分類には、Software CMM（プロセスの型による分類）、People CMM（組織内の人々の処遇による分類）などがある。

学習する組織を構築する３つの柱
ピーターMセンゲ



- 志を立てる力
  ～組織と個人のビジョン
  「現実」と「なりたい姿」のギャップの間に生まれる力を推進力に
- 複雑性の理解力
  ～システム思考
- 共創的な対話力
  ～メンタル・モデルとダイアログ

PMBOK 第7版
プロジェクトマネジメントの12原則



- スチュワードシップ
- 協力的なプロジェクトチームの環境を作る
- ステークホルダーを効果的に連携する
- 価値に集中する
- システムの相互作用を認識し、評価し、対応する
- リーダーシップを行動で示す
- 文脈に基づいたテーラリング（カスタマイズ）
- 品質をプロセスと成果物に組み込む
- 複雑性に適応する
- リスクへの対応を最適化する
- 適応力とレジリエンスを高める
- 未来の状態を達成するために変化できる

システム思考

デザイン思考＝みんなで探索的に創る

【5つのMode】

・Empathize：**共感**

・Define：**問題定義**

問題のリフレーミング

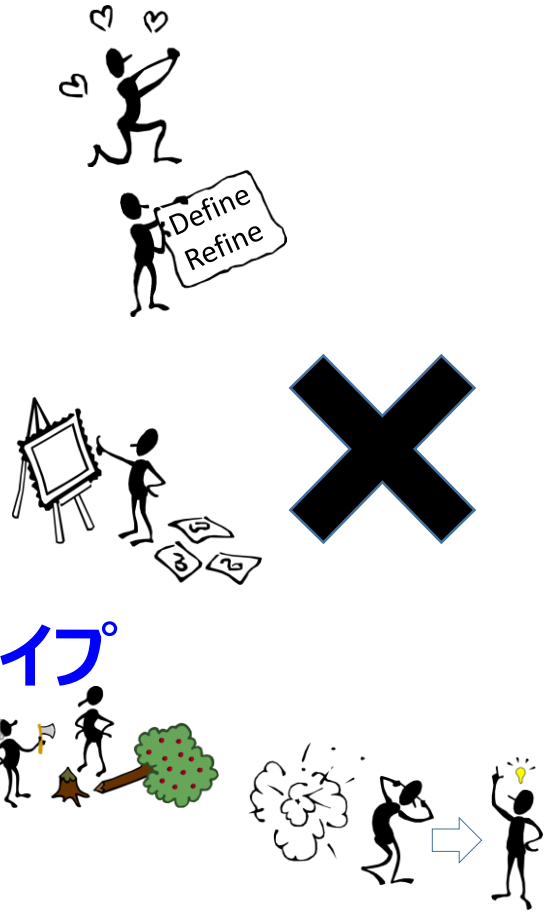
・Ideate：**創造**

発散⇔収束

・Prototype：**プロトタイプ**

・Test：**テスト**

やってみて学ぼう！



引用元：-the d.school bootcamp bootleg

【4つのMindset】

・Human-Centered：**人間中心**

人間を意識して考える

・Collaborative：**協力的**

多様性を活かす

・Optimistic：**楽観的**

われわれにもできる！信念

・Experimental：**実験的**

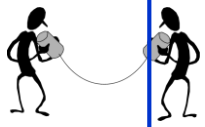
たくさんの試行経験から学ぶ

引用元：Design Thinking for Educators Toolkit,IDEO,2011

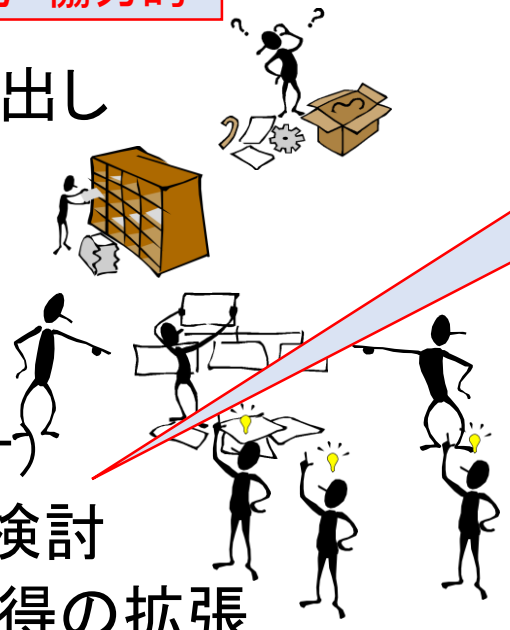
SaPID＝システム思考×デザイン思考

システム思考

- テーマ設定＋Vision Mission Concept把握
  - 例1：〇〇開発プロジェクトの成功要因・失敗リスク
  - 例2：レビューにおける困り事・問題点
- 現状分析
  - 関係者が持つ関連情報（要素）の洗い出し
  - 構造（関係性）分析：物理モデル構築
  - 関係性構造の論理モデル化
- 未来設計・実装・実行
  - 新たな価値連鎖モデル構築（価値設計）
  - 成長に必要な改善要素の特定と施策検討
  - [トライアル＆ふりかえり]による成果獲得の拡張



共感・問題定義・人間中心・協力的



人間中心・楽観的・協力的・創造的

デザイン思考

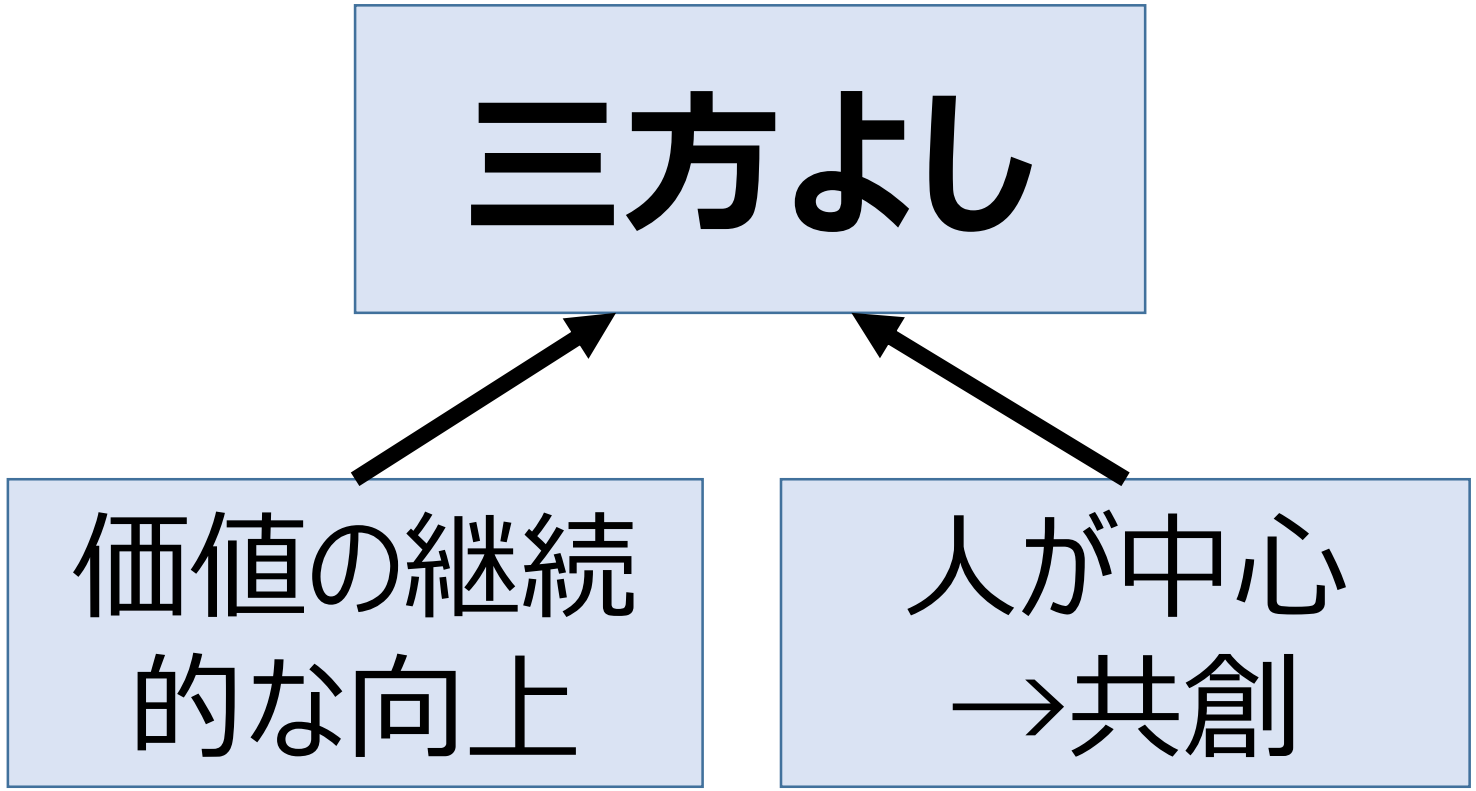
実験的／プロトタイプ・テスト

SaPIDのコンセプト：対象全体につらぬかれた骨格となる発想や観点

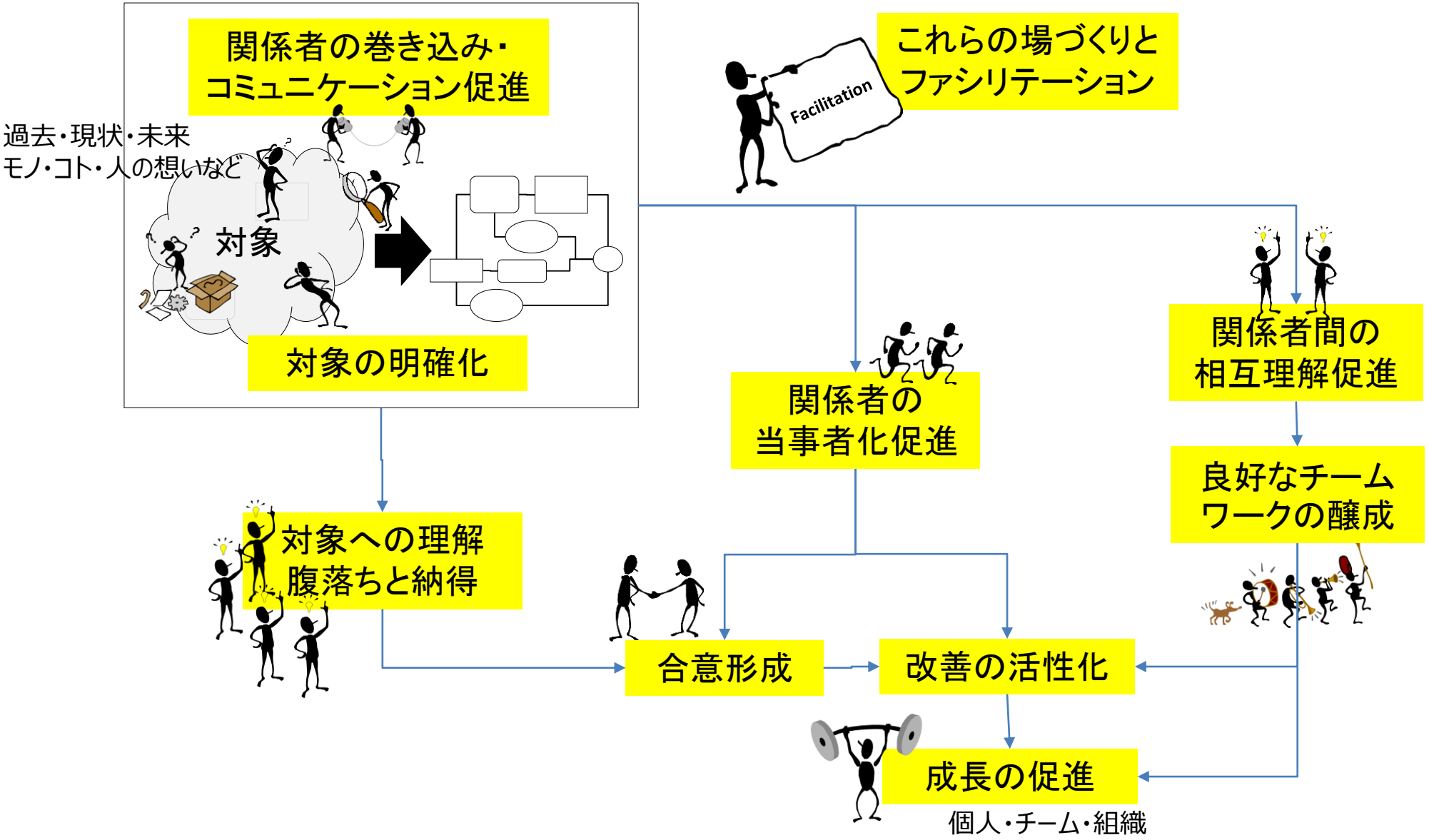
自律～自らが変化の起点になる

- 自らを律する、変えることによって結果を出す。成果を上げる。
  そして（その結果）周囲が変わる。

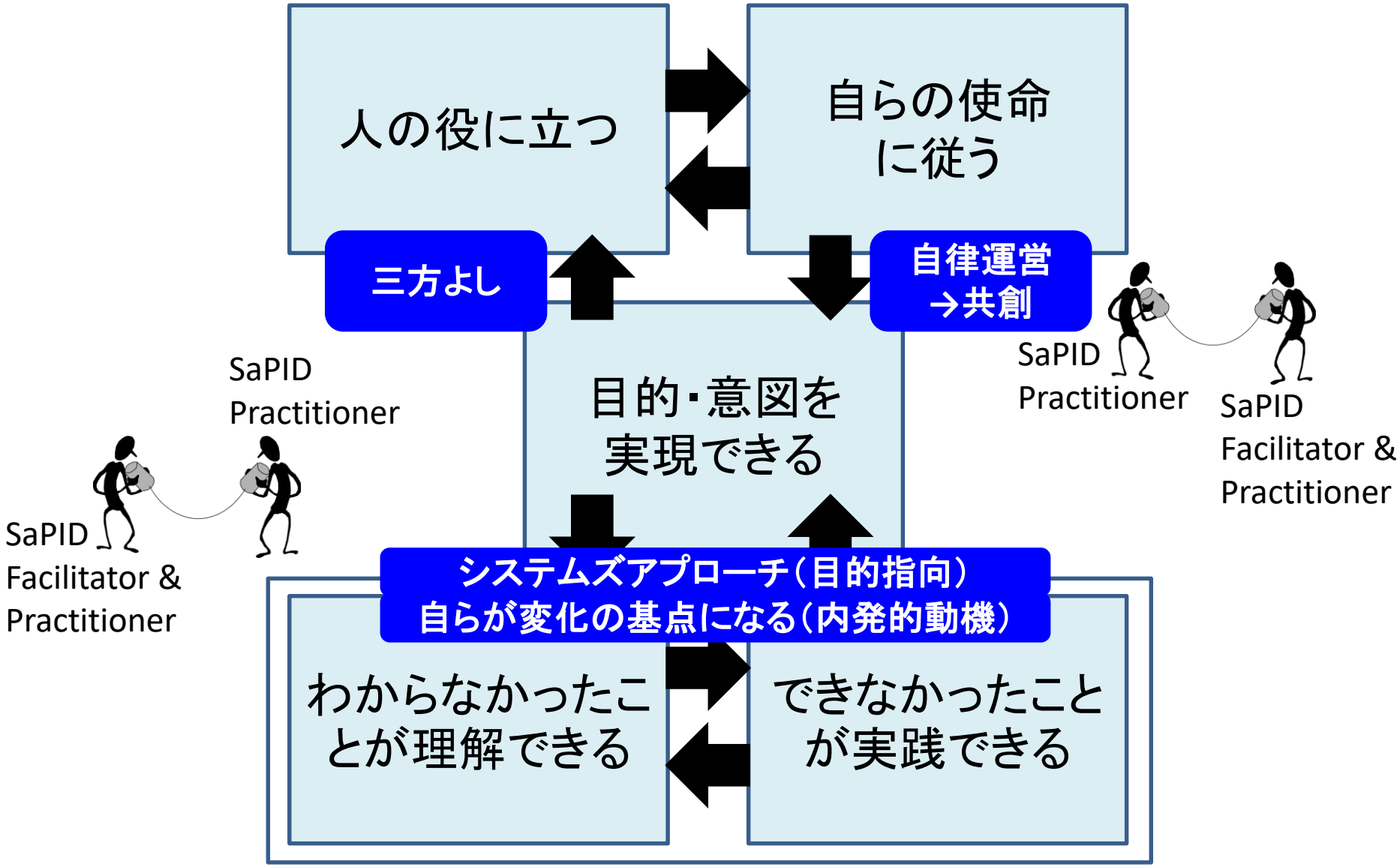
SaPIDのポリシー：物事を行うときの方針や原則



SaPIDでできること・やっていること

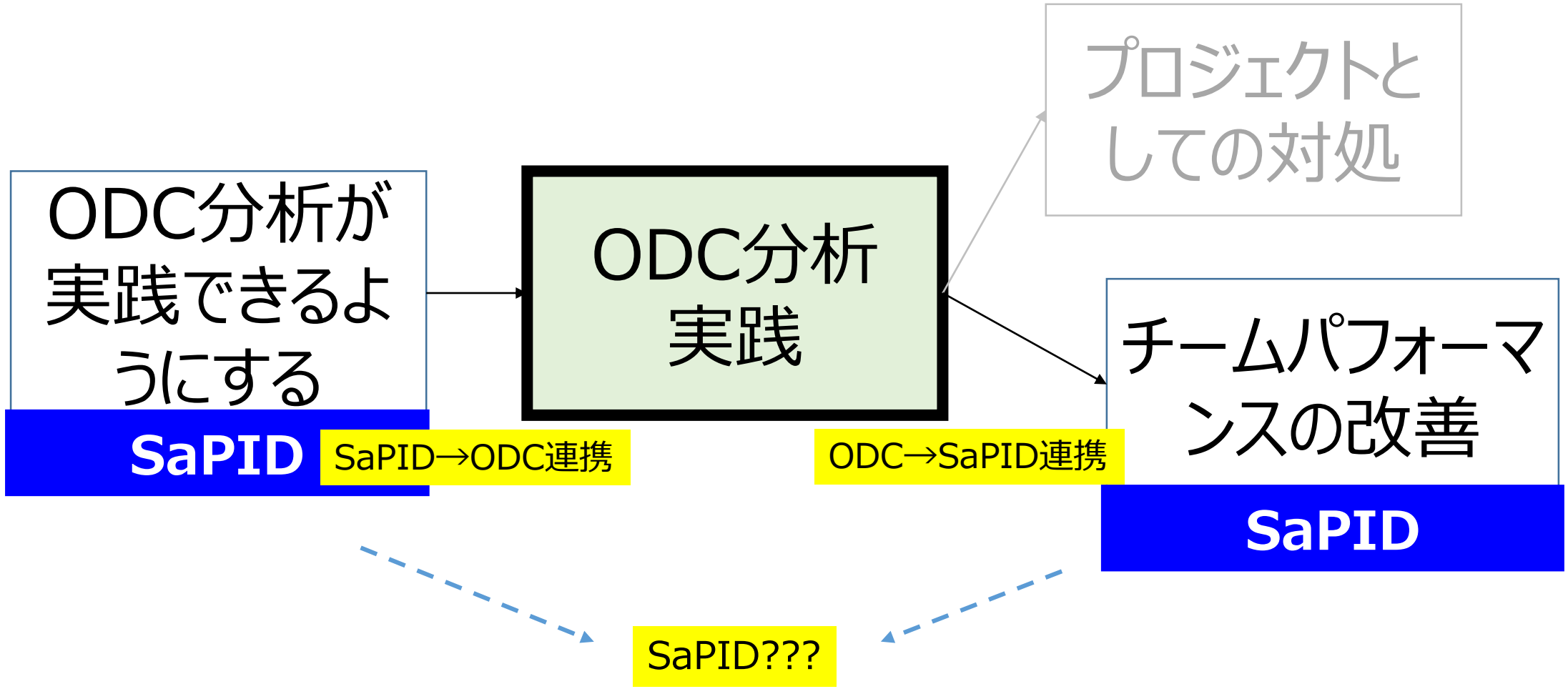


（依存）→自律→共創（相互依存）による幸せ実現モデル



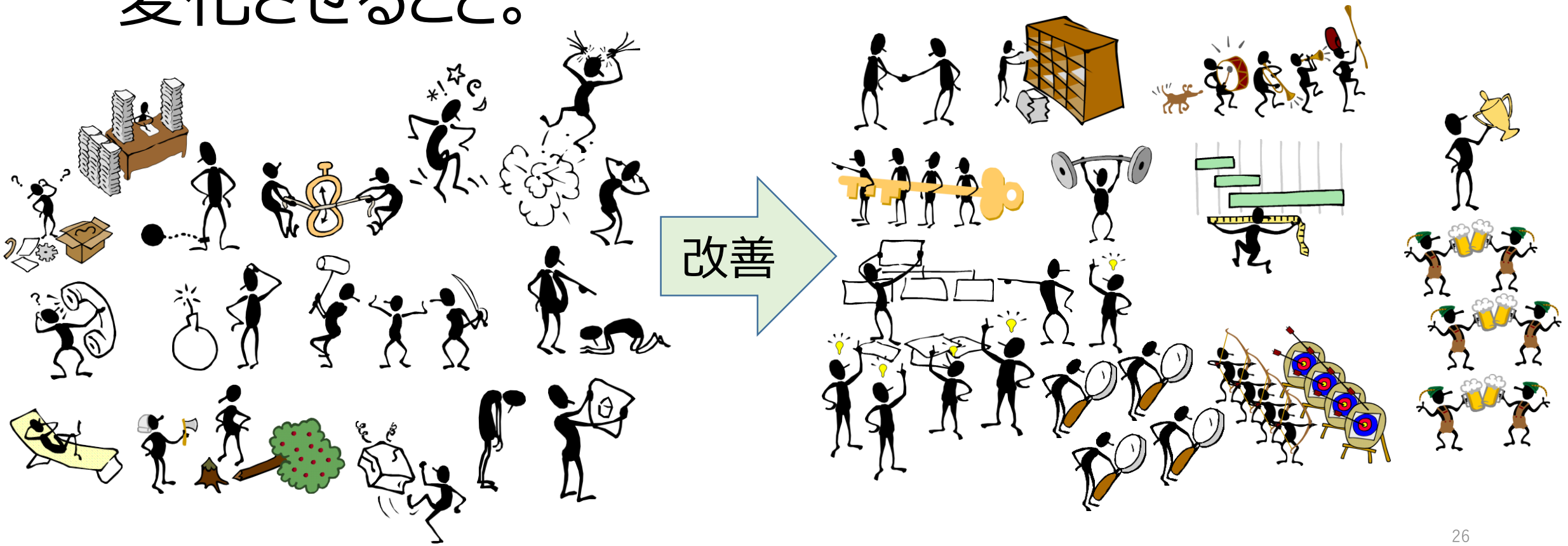
ODC分析×SaPIDとは？

今日のテーマ～ODC分析とSaPIDの関係

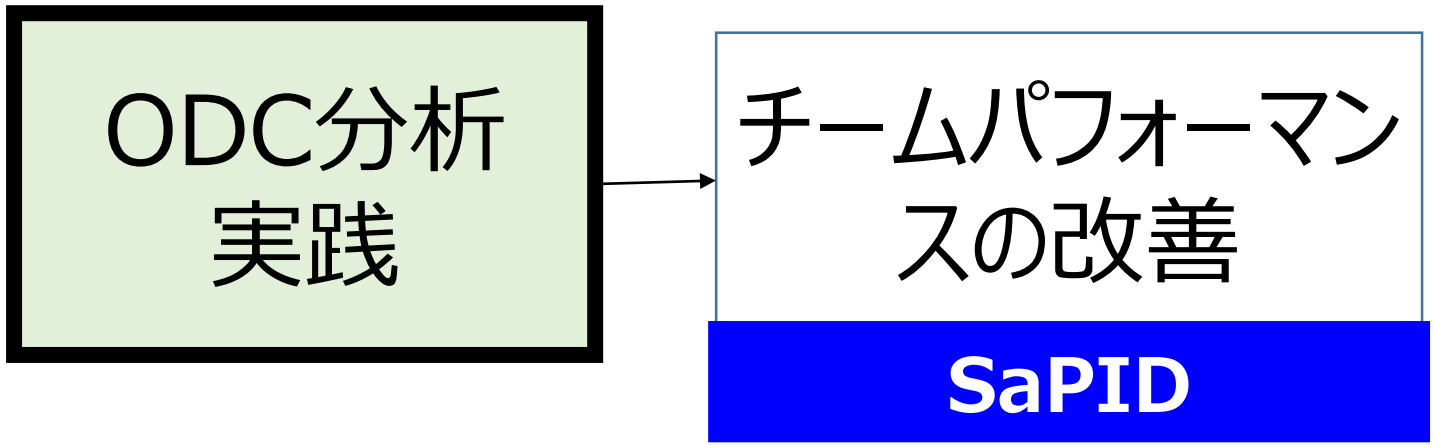


“チームパフォーマンス”の改善とは？

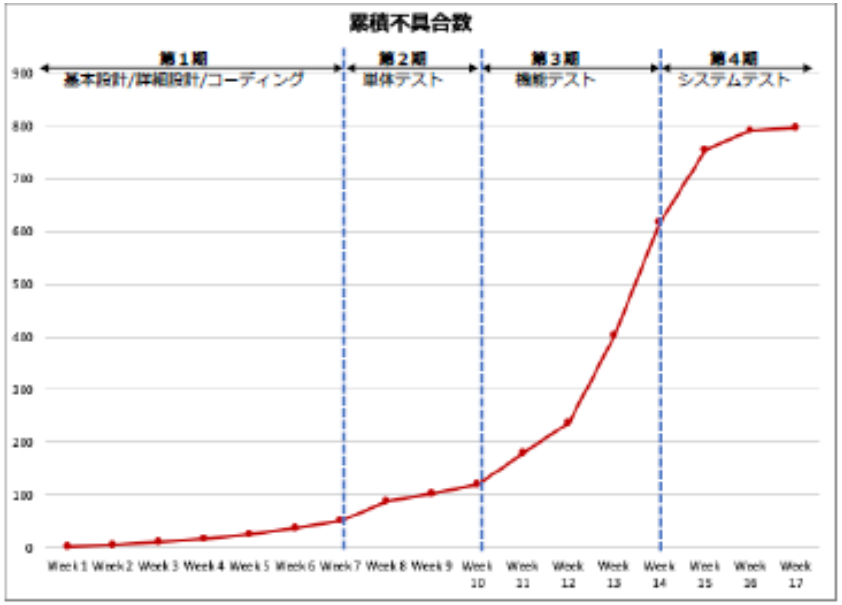
- チームとしての活動や成果を“より相応しい”状態に変化させること。



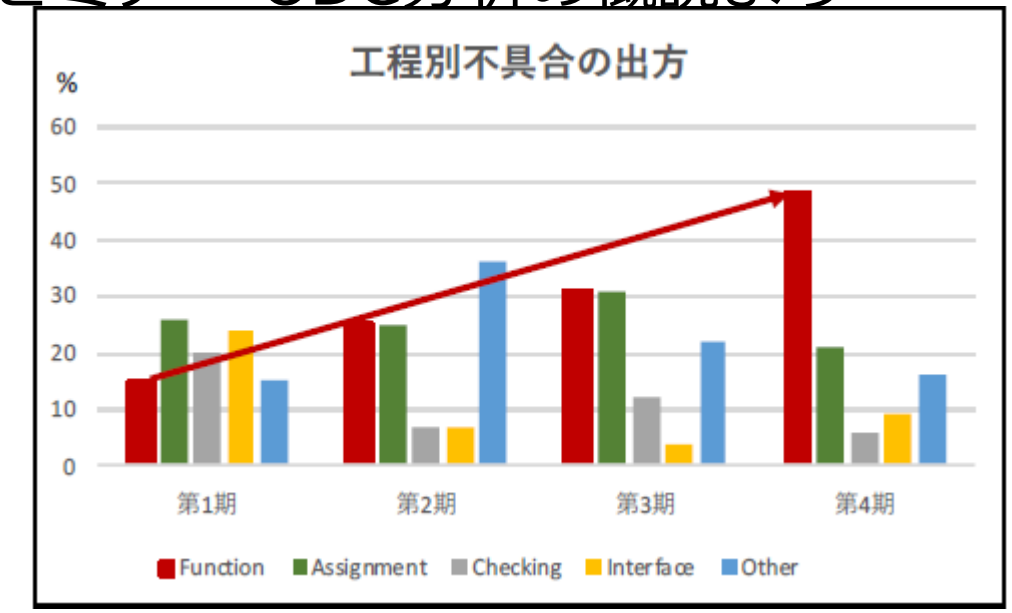
ODC分析を活用したチームパフォーマンス改善【ODC→SaPID連携】



ODC分析適用事例1
ODC分析研究会オープンセミナー ODC分析の概説より



適用された開発プロセスに沿って計画どおり開発・テストを進めて、以下のような判定で「出荷可能」と判断した。
1) 各工程で定義された開発・テスト活動は、すべて完了した。
2) 摘出された不具合は、すべて修正した。
3) 摘出された不具合検出数は、おおむね品質計画どおりの傾向を示し、
4) システム終盤に来て、新しく摘出される不具合数は収束している。



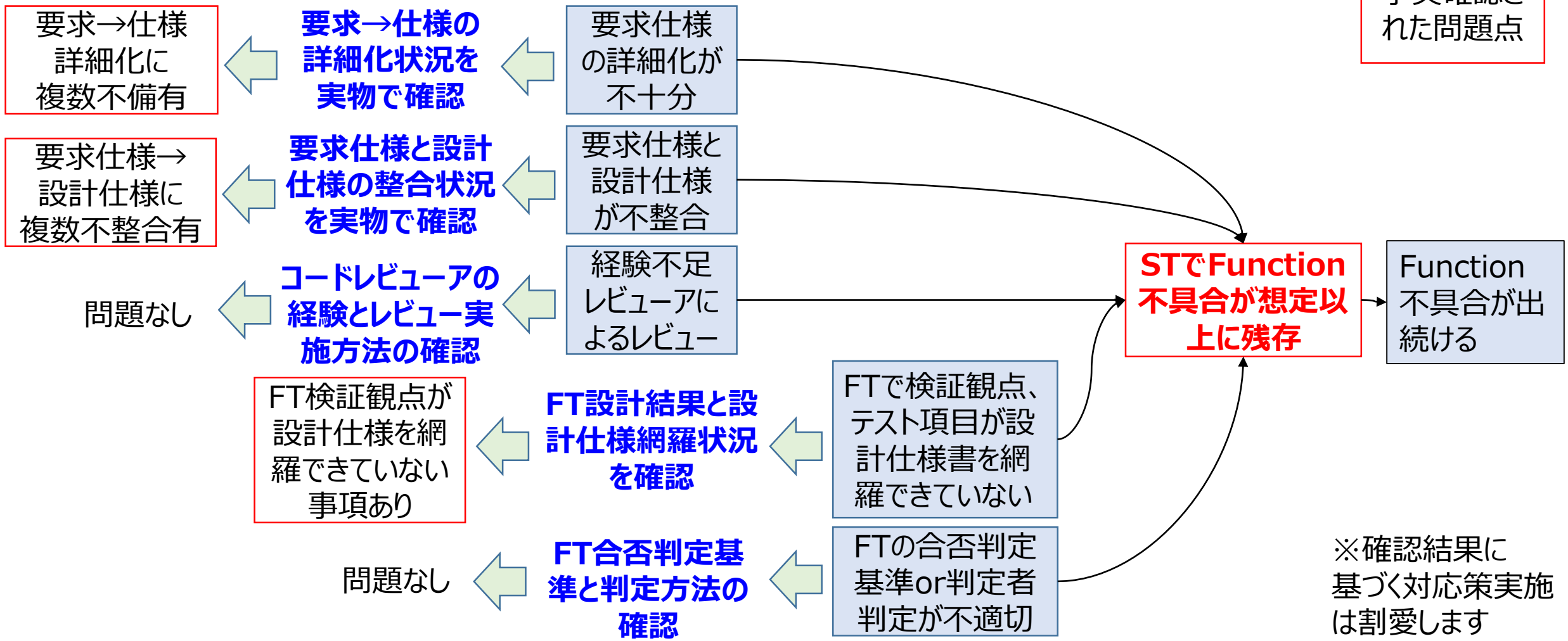
指摘1：Function(機能性)に関わる不具合の前工程からの漏れ
指摘2：残存不具合(Function) 対応によるシステムテストへの影響

示唆される対応策1：設計工程(設計レビューやコード検証)での機能性に対する次のレビュー観点の見直しが必要である。
- 要求仕様の詳細化の十分性
- 要求仕様と設計仕様の整合性
- コード・インスペクションのレビューアの経験値

示唆される対応策2：機能テストのカバレージの網羅性と詳細性および実施体制の検証が必要である。以下の点を検証する。
- 機能テスト(統合テスト)での、検証観点、テスト項目が設計仕様書を網羅しているか。（ここで食い止めるべき）
- テスターの選定は適正か、テストの合否判定基準は正しく文書化され教育されているか。

SaPID適用事例

STEP1
ODC分析で立てた仮説を事実で確認する



SaPID適用事例

STEP2-1
フェーズ内作り込み要因の状態を確認する

事実確認済
問題点

新たに事実
確認された
問題点

要求→仕様
詳細化に
複数不備有

要求→仕様の詳細化で不備が出た要因は？

要求なのか仕様なのかが不確かな記述形態になっている

変更要求の仕様化に一部失敗していた

現状分析結果がどこにも存在しない

要求仕様→
設計仕様に
複数不整合有

要求仕様→設計仕様の複数不整合が出た要因は？

設計仕様のみ記述する形式となっている(どの要求に対する設計仕様なのかは要求仕様書と突合しないとわからない)

複数の要求をそのまま設計仕様化し、詳細化不足になっている

FT検証観点が
設計仕様を網
羅できていない
事項あり

FT検証観点が設計仕様を網羅できていない要因は？

基本的に*CPM法*でテスト設計を行っていたが、テスト設計後の仕様変更発生箇所については未対応であった

テスト設計後の仕様変更への対応がコード修正のみとなっていた

※確認結果に
基づく対応策実施
は割愛します

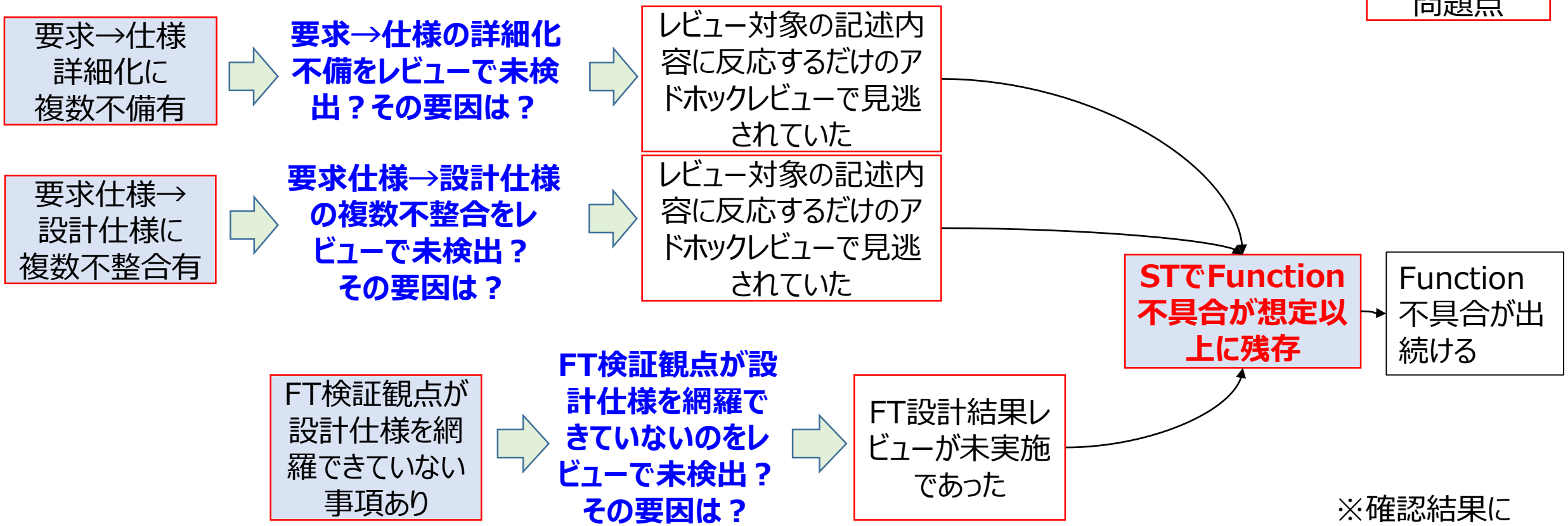
※CPM法：Copy and Paste and Modify法（電気通信大にしさんが命名）

SaPID適用事例

STEP2-2
フェーズ外流出要因を確認する

事実確認済
問題点

新たに事実
確認された
問題点



※確認結果に
基づく対応策実施
は割愛します

SaPID適用事例

STEP3
確認した事実情報を構造化する

作り込み要因

流出要因

要求定義

- 現状分析結果がどこにも存在しない
- 要求なのか仕様なのかが不確かな記述形態
- 変更要求を処理する際に仕様への展開に一部失敗していた
- レビュー対象の記述内容に反応するだけのアドホックレビューで見逃されていた

基本設計

- 複数の要求をそのまま設計仕様化し、詳細化不足になっている
- 設計仕様のみ記述する形式となっている
  (どの要求に対する設計仕様なのかは要求仕様書と突合しないとわからない)
- レビュー対象の記述内容に反応するだけのアドホックレビューで見逃されていた

機能テスト

- テスト設計後の仕様変更への対応がコード修正のみとなっていた
- 基本的に*CPM法*でテスト設計を行っていたが、テスト設計後の仕様変更発生箇所については未対応であった
- FT検証観点が設計仕様を網羅できていない事項あり
- FT設計結果レビューが未実施であった

STでFunction不具合が想定以上に残存 → Function不具合が出続ける

※確認結果に基づく対応策実施は割愛します

※CPM法：Copy and Paste and Modify法（電気通信大にしさんが命名）

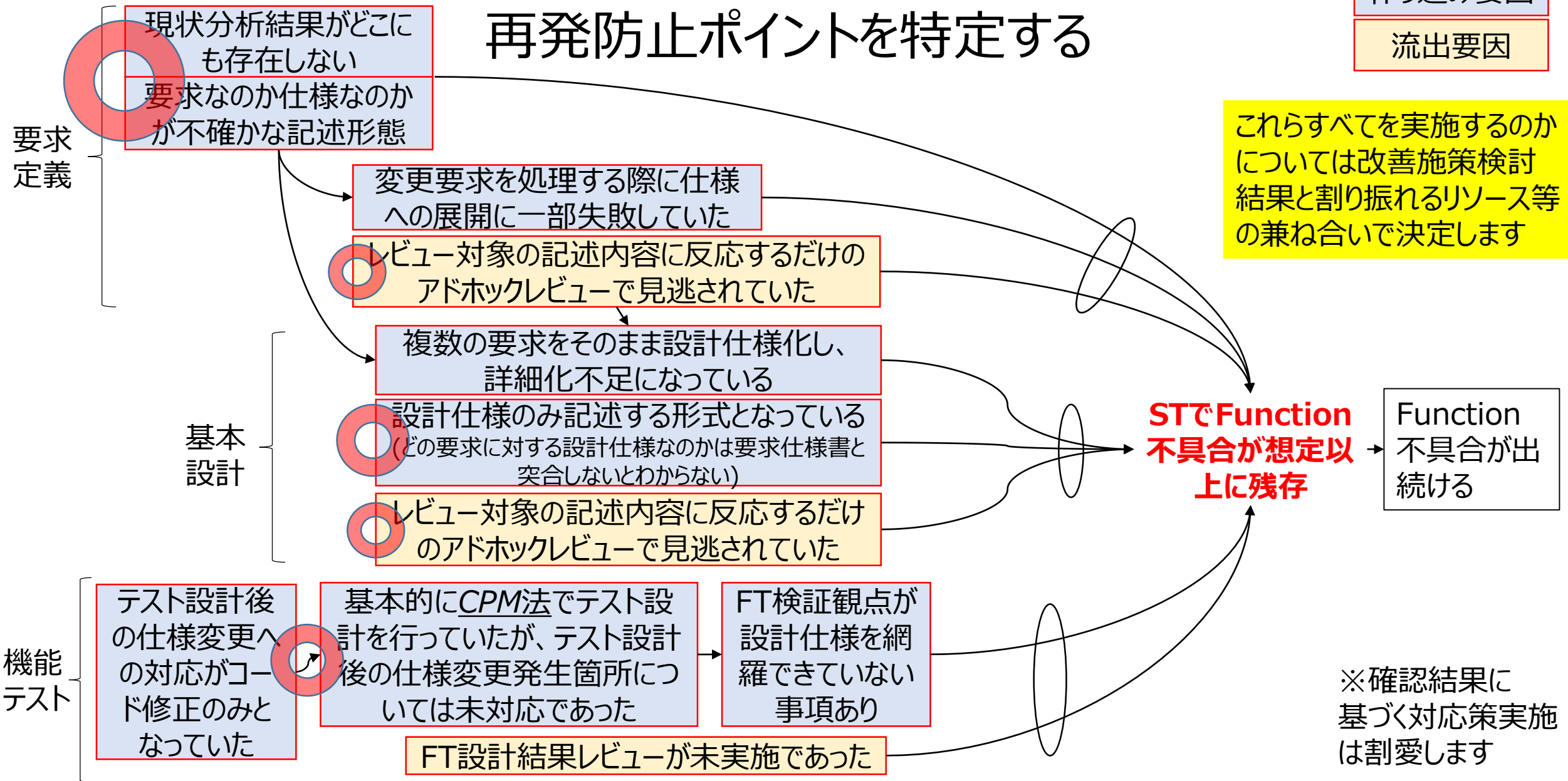
SaPID適用事例

STEP4
再発防止ポイントを特定する

作り込み要因

流出要因

これらすべてを実施するのかについては改善施策検討結果と割り振れるリソース等の兼ね合いで決定します



※確認結果に基づく対応策実施は割愛します

※CPM法：Copy and Paste and Modify法（電気通信大にしさんが命名）

SaPID適用事例

STEP5
再発防止施策を検討する

作り込み要因

流出要因

再発防止ポイント

要求定義

現状分析結果がどこにも存在しない

要求なのか仕様なのかが不確かな記述形態

レビュー対象の記述内容に反応するだけのアドホックレビューで見逃されていた

※基本設計・機能テストの再発防止施策検討は割愛します

① 施策案1

現状分析結果を成果物として共有する

既存記述に要求、仕様を示す識別子を付与する

実践容易性：低1～中2

有効性：中2

② 施策案2

理由-要求-仕様をセットで記述するテンプレートに沿って表現する(USDM)

実践容易性：低1～中2

有効性：高3

評価結果からこちらを選択

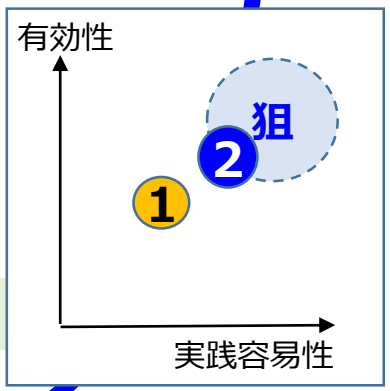
要求仕様書レビュー時の要求-仕様の十分性確認観点を必須とする

実践容易性：低1～中2

有効性：高3

実践容易性：中2～高3

有効性：高3



自由形式の要求仕様記述例

■前提条件
電源ON/OFFはコンセントの抜き刺しで行われる。
電源ON時は00:00を設定し、タイマ動作停止モードとする。
00:00の前2桁は分、後ろ2桁は秒を示す。
■機能
設定した時間から1秒ずつカウントを減らし、00:00になった時点でブザーを鳴らす。
＜時間設定＞
・タイマ時間設定は“分のみ”対象とする。（秒はそのままとする）
・初期値は00分、最大値は60分
・タイマ動作停止モードでタイマボタンを3秒以上長押すると、時間表示をすべてリセット（00:00）して点滅し、時間設定モードに遷移する。
・時間設定モードでタイマボタンを1回押下（ 3秒未満）すると1分カウントアップし、ブザー音“ピ”を鳴らす。（60分表示でタイマボタンを押すと00分とする）
・時間設定モードでタイマボタンを3秒以上長押すると、時間表示点滅を終了（点灯）し、タイマ動作停止モードに遷移する。
＜タイマ動作開始・停止＞
時間設定が00:00を超える場合、タイマーボタンを1回押下（ 3秒未満）すると計測を開始する。設定時間から1秒ずつカウントを減らし、その経過をリアルタイムで時間表示する。また計測中にタイマボタンを1回押下（ 3秒未満）するとタイマ動
＜タイムアッ
カウントダウ 00になった時点でブザーを鳴らす。ブザー音は“ピピピピ、ピピピピ、 イマボタンを1回押下（ 3秒未満）するとブザー音を止める。

参考：[入門+実践]要求を仕様化する技術・表現する技術 -仕様が書けていますか?

USDM形式の要求仕様記述例

要求	A1	タイマの電源をON／OFFする	
	理由	タイマを使用するとき電源を入れたい。使わないときはOFFにしたい。	
仕様	□	A1-1	コンセントに差すと電源ON。時刻に00:00を設定・表示し、タイマ動作停止モードとなる。
	□	A1-2	コンセントを抜くと電源OFF。
要求	A2	タイマに計測する時間を設定する	
	理由	計測する時間が用途で異なるため自由に設定したい。	
仕様	□	A2-1	タイマ動作停止モードでタイマボタンを長押すると、時間表示をすべてリセット（00:00）して点滅し、時間設定モードに遷移する。
	□	A2-2	タイマボタン押下すると1分カウントアップ、ブザー音“ピ”を鳴らす。(60分表示でタイマボタン押下→00分とする/3000～4000Hz、40dB前後の音)
	□	A2-3	時間設定モードでタイマボタンを長押すると、時間表示点滅を終了（点灯）し、タイマ動作停止モードに遷移する。
要求	A3	タイマ計測を開始する／一時停止する	
	理由	利用者の都合やタイミングで計測を開始したり、一時止めたり（再開したり）したい。	
仕様	□	A3-1	時間設定が00:00を超える場合、タイマボタンを押下すると計測を開始する。1秒ずつカウントダウンし、その経過をリアルタイムで時間表示する。
	□	A3-2	計測中にタイマボタンを押下するとタイマ動作を停止する。
要求	A4	設定時間が経過したら音で知らせる／音を止める	
	理由	多少離れた場所でも設定した時間が経過したことを知りたい／把握できたら音を止めたい。	
	□	A4-1	カウントダウンの結果、00:00になった時点でブザーを鳴らす。ブザー音は“ピピピピ、ピピピピ、・・・”とし、3000～4000Hz、40dB前後の音とする。

SaPID適用事例

STEP6-1
再発防止施策の展開方法を検討する

要求仕様書は理由-要求-仕様をセットで記述するテンプレートに沿って表現する(USDM)

①基本タスク
USDMテンプレートを準備する。
典型的な記載事例、Q&Aを作成して共有する。
最初に着手するメンバーにUSDM説明会＆トレーニングを行う／単元ごとにビデオ化していつでも参照できるようにする。→以降は着手前にビデオを視聴してから着手する。
問合せ窓口担当者を割当て、問合せ実績をQ＆Aに反映する。
レビュー結果の指摘内容の推移をモニタリングして、定着度を把握し、施策実践期間をコントロールする。

※以外の施策については割愛します

②リスク	③中間目標	④リスク対策
ゼロからUSDM記述を作成すると時間がかかりすぎる。	想定内時間でUSDM記述が完成する。	□既存の典型的な要求仕様書をUSDM記述に転写し、過不足を把握する。過不足に対する対策を明確にして記述事例やガイド記述に反映する。 □まずは規模、難易度が小さいプロジェクトでトライアル実践し、結果をふりかえ、必要な対策を揃えていく。徐々に難易度の高いプロジェクトに適用し、最終的にすべての案件に適用する。 □記載事例をコピペして作業を進める。 □定着するまではUSDM特有のチェック項目をレビューで確認し、的確にフィードバックする。 □完成した要求仕様書は関係者が利用可能な事例として活用する。 □実践済みメンバーが以降のトレーニングを実施するように運営する。 □テンプレート、記述事例などに記述ガイドを付与する。 □典型的な非機能要求記述例を充実させ、それらを利活用しながら作業を進められるようにする。 □規模あたりの要求→仕様展開失敗件数をBefore-Afterで確認・共有する
これまで記述していなかった項目を記述するため時間がかかる。		
理由欄が空欄のまま／自社都合の理由が記述される。	適切な理由記述になっている。	
非機能要求が記述されない。	非機能要求が適切に記述される。	
改善効果が実感できない	改善効果を実感している	

SaPID適用事例

STEP6-2
再発防止施策の展開方法を具体化する

基本タスク
中間目標
リスク対策
施策管理タスク

改善効果を実感している

規模あたりの要求→仕様展開失敗件数をBefore-Afterで確認・共有する

実践済みメンバーが以降のトレーニングを実施するように運営する

完成した要求仕様書は関係者が利用可能な事例として活用する

レビュー結果の指摘内容の推移をモニタリングして、定着度を把握し、施策実践期間をコントロールする

問合せ実績をQ&Aに反映する

問合せ窓口担当者を割当てる

想定内時間でUSDM記述が完成する

非機能要求が適切に記述される

適切な理由記述になっている

改善失敗リスクをMin化

徐々に難易度の高いプロジェクトに適用し、最終的にすべての案件に適用する

まずは規模、難易度が小さいプロジェクトでトライアル実践し、結果をふりかえ、必要な対策を揃えていく

定着するまではUSDM特有のチェック項目をレビューで確認し、的確にフィードバックする

着手するメンバーにUSDM説明会＆トレーニングを行う

記載事例をコピペして作業を進める

典型的なQ&Aを作成して共有する

USDMテンプレートを準備する

典型的な記載事例を作成して共有する

テンプレート、記述事例などに記述ガイドを付与する

過不足に対する対策を記述事例やガイド記述に反映する

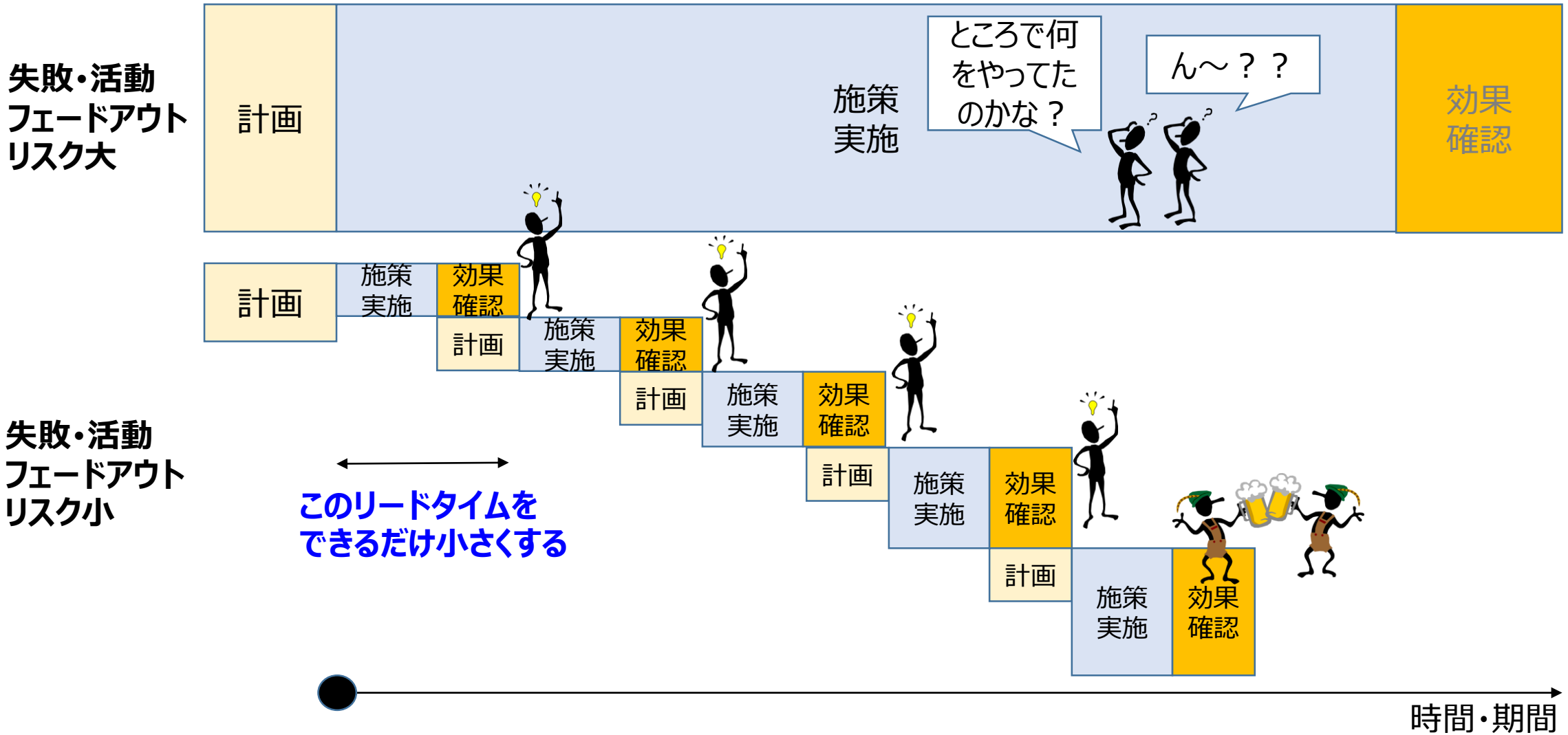
既存の典型的な要求仕様書をUSDM記述に転写し、過不足を把握する

典型的な非機能要求記述例を充実させ、それらを利活用しながら作業を進められるようにする

施策を具体化した結果と割り振れるリソース等の兼ね合いで内容を調整します

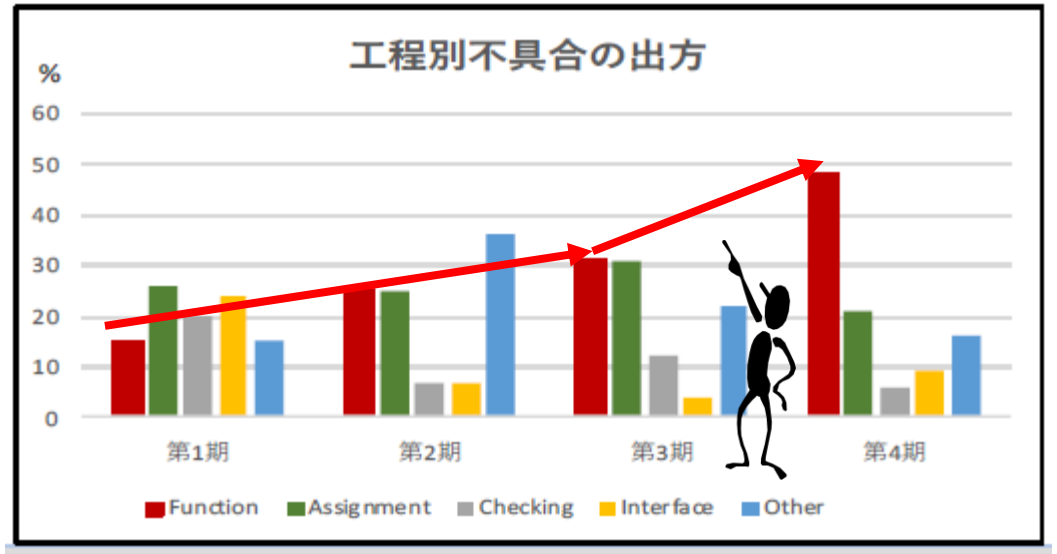
※レビューに対する施策2については割愛します

パフォーマンス改善では効果実感や手ごたえを早期に



参考

事例1、お前ならどう見る？と言われたら・・・



どうして急に立ち
上がったのか？

□仮説
直前にいくつかの機能に対する変更要求が発生し、その対処を行った。あるいは概要レベルだった機能要求や仕様が直前でようやく具体的に固まった可能性。
→対処への確認テストを第4期に実施した結果かもしれない。
→対処によりデグレードを起こしているかもしれない。

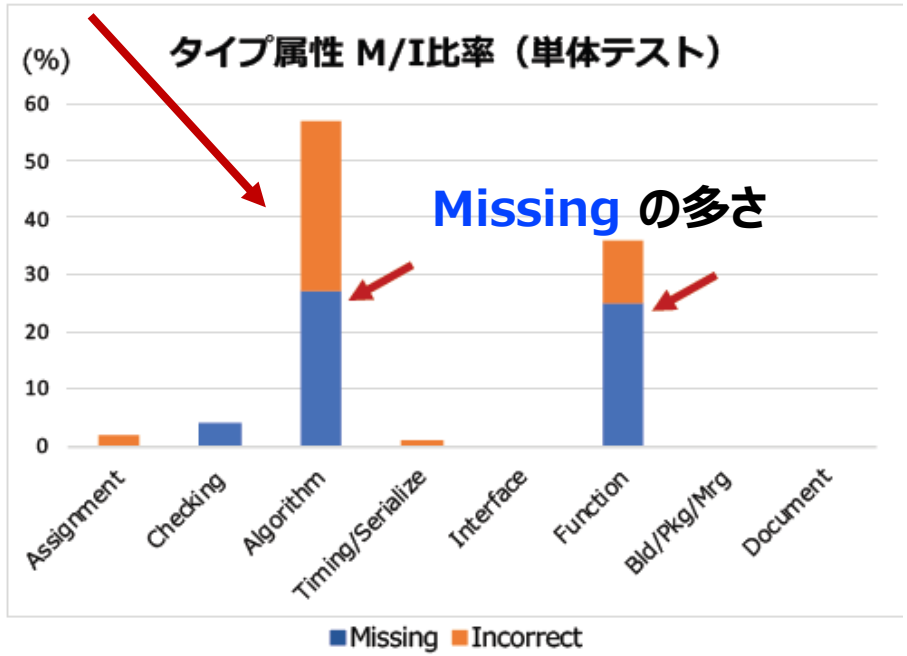
□検証
・Functionバグの内容、対象機能、テスト項目と意図（観点）、バグ発生の背景と要因把握
・機能に対する変更要求とその対処実績（時系列で）
・機能テスト対象の状態とテスト項目、観点の把握
・システムテスト項目と観点の設計方法と具体的内容把握

システムテスト結果の積算、サマリ分析をする以前に、個々の故障検出結果から確認して処置する方がより適切
さらに、システムテストに入る前に変更要求対処の有無やテスト項目設定について関係者で認識共有することが大事

ODC分析適用事例2

ODC分析研究会オープンセミナー ODC分析の概説より

「アルゴリズム(Algorithm)」
の不具合の割合が突出



かなりタイトなスケジュールで開発を進めているプロジェクトである。設計を終えて、単体テストを開始したが、あまりの不具合の多さに、一旦テストを中断してタイプ属性のM/I比率を分析すると、下図の様な分布を示した。

指摘1：「アルゴリズム」の不具合の多さ
指摘2：機能性のMissing (忘れてました！)の多さ
「Function(機能性)」にかかわる不具合のうちMissing の不具合がこの工程での不具合総数の1/4 (25%)を、「Function」の不具合数の70%近くを占めている。

仮説
・「アルゴリズム」の不具合の多さは、詳細設計が存在していないか、不十分であることを示している。
・「Missing (忘れてました！)の多さ」は、設計工程またはそれ以前の要求分析工程でのレビューが不完全だったため、要求事項あるいは設計事項の見落としや考慮不足が多発していることを示している。

示唆される対応策
・要求から基本設計、詳細設計への紐付け再検討
単体テストを中断したのは正しい。このまま続行してもテストの意味をなさない。
正しく要求が仕様化され、それが基本設計書、詳細設計書へと抜け漏れなく紐づけられることを検証すべきである。
もし抜け漏れがあれば、工程を遡って再設計、再レビューの実施と仕様書の更新が必要である。

SaPID適用事例

STEP1
状況を整理し、確認事項を明確にする

かなりタイトな
スケジュール

要求定義		基本設計		詳細設計		実装	
要求定義	レビュー	基本設計	レビュー	詳細設計	レビュー	コード化	レビュー

課題から要求、要求から仕様への展開結果は？

レビューは実施された？

レビュー観点と結果は？

要求仕様からの設計結果は？

レビューは実施された？

レビュー観点と結果は？

詳細設計は実施された？

レビューは実施された？

レビュー観点と結果は？

コード化は何に基づいて実施？

レビューは実施された？

レビュー観点と結果は？

要求変更・仕様変更の発生実績とその対処はどのように？(不備やデグレードの可能性)

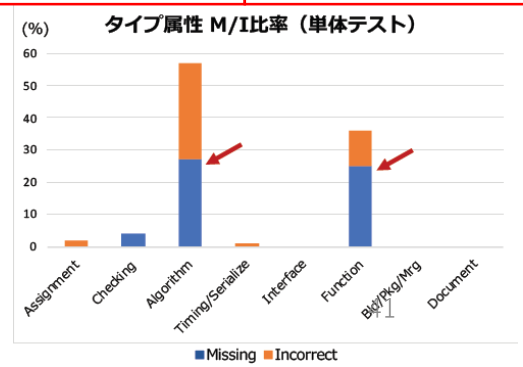
事実確認された問題点

単体テストの途中段階

アルゴリズムの不具合が全体の55%

機能性の不具合が全体の35%

Missing の不具合が不具合総数の25%、Function不具合数の70%



SaPID適用事例

STEP2
フェーズ内作り込み・流出要因を確認する

既存システムを要求・仕様で説明できる顧客は存在しない

タイトなスケジュール

新たに事実確認された問題点

既存システムを運用している要員がいるから安く焼き直せると受注

要求定義		基本設計		詳細設計		実装	
要求定義	レビュー	基本設計	レビュー	詳細設計	レビュー	コード化	レビュー

単体テストの途中段階

既存システムを大枠理解している要員は2名/詳細はわからない

「既存通り」の箇所は詳細設計なし

既存システムを理解できていない要員が実装したところが多い

アルゴリズムの不具合が全体の55%

Missingの不具合が不具合総数の25%、Function不具合数の70%

既存システムの振る舞いは動作させないと＋顧客実務担当者しかわからない

既存システムで使いにくい箇所への改修依頼が実装中にも多数発生

新機能を中心にレビューを実施

機能性の不具合が全体の35%

設計結果の多くが「既存通り」で済ませていた

要求・仕様ともに「既存通り」で済ませていた

改修依頼対応時に仕損じとデグレードが複数発生

レビューで「既存通り」部分はすべて問題なし扱い

レビューで「既存通り」部分はすべて問題なし扱い

改修レビューは参加できる要員だけで実施

顧客の各機能の実務担当が単体Tに参画

SaPID適用事例

STEP3
確認した事実情報を構造化する

作り込み要因

流出要因

背景・事実

タイトな
スケジュール

既存システムを運用している要員がいるから安く焼き直せると受注

既存システムで使いにくい箇所への改修依頼が実装中にも多数発生

要求・仕様ともに「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし

改修依頼対応時に仕損じとデグレードが複数発生

改修レビューは参加できる要員だけで実施

既存システムを要求・仕様で説明できる顧客は存在しない

既存システムを大枠理解している要員は2名/詳細はわからない

既存システムの振る舞いは動作させないと＋顧客の実務担当しかわからない

設計結果の多くが「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし扱い

「既存通り」の箇所は詳細設計なし

既存システムを理解していない要員が実装

新機能を中心にレビューを実施

顧客の各機能の実務担当が単体Tに参画

アルゴリズムの不具合が全体の55%

機能性の不具合が全体の35%

Missing 不具合が不具合総数の25%、Function不具合数の70%

SaPID適用事例

STEP4
再発防止ポイントを特定する

作り込み要因
流出要因
背景・事実

タイトな
スケジュール

既存システムを運用している要員がいるから安く焼き直せると受注

既存システムを要求・仕様で説明できる顧客は存在しない

既存システムを大枠理解している要員は2名/詳細はわからない

既存システムの振る舞いは動作させないと＋顧客の実務担当しかわからない

要求・仕様ともに「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし

設計結果の多くが「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし扱い

「既存通り」の箇所は詳細設計なし

既存システムで使いにくい箇所への改修依頼が実装中にも多数発生

改修依頼対応時に仕損じとデグレードが複数発生

改修レビューは参加できる要員だけで実施

既存システムを理解していない要員が実装

新機能を中心にレビューを実施

顧客の各機能の実務担当が単体Tに参画

アルゴリズムの不具合が全体の55%

機能性の不具合が全体の35%

Missing 不具合が不具合総数の25%、Function不具合数の70%

SaPID適用事例

STEP5
再発防止施策を検討する

作り込み要因

流出要因

背景・事実

再発防止ポイント

既存システムを要求・仕様で説明できる顧客は存在しない

既存システムを大枠理解している要員は2名/詳細はわからない

既存システムの振る舞いは動作させないと＋顧客の実務担当しかわからない

既存システムを運用している要員がいるから安く焼き直せると受注

施策案

事実に基づき状況を把握し、実見積に基づき受注判断を行う。別途必要なら顧客に提示する見積を別ものとして作成する。

顧客提示見積ではなく、実見積結果に基づきプロジェクトを計画・運営する

プロジェクトリスクを特定し、その対処を計画に含める

受注判定者や契約承認者、営業担当、プロジェクト管理層が対応

要求・仕様ともに「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし

設計結果の多くが「既存通り」で済ませていた

レビューで「既存通り」部分すべて問題なし扱い

「既存通り」の箇所は詳細設計なし

開発成果物上は「既存通り」であっても必ず要求・仕様・業務フローなど具体的成果物に落とし込み開発を進める

プロジェクト要求定義、設計責任者・担当者が対応

※他の再発防止施策検討は割愛します

SaPID適用事例

STEP6-1
再発防止施策の展開方法を検討する

事実に基づき状況を把握し、実見積に基づき受注判断を行う。
別途必要なら顧客に提示する見積を別ものとして作成する。

①基本タスク
当改善実施体制の構築（責任者・実施運営者・問い合わせ窓口・対象部門等）
現状調査・要求事項明確化・見積（実見積→顧客提供見積）・受注判定プロセスの構築と社内承認
構築したプロセスの関係者説明会の実施
一定期間、新プロセス実践状況のモニタリングとフィードバックを実施→責任者へ報告
施策が定着し、効果が確認できたこと＋通常運営状態にシフトする判定と連絡を行う

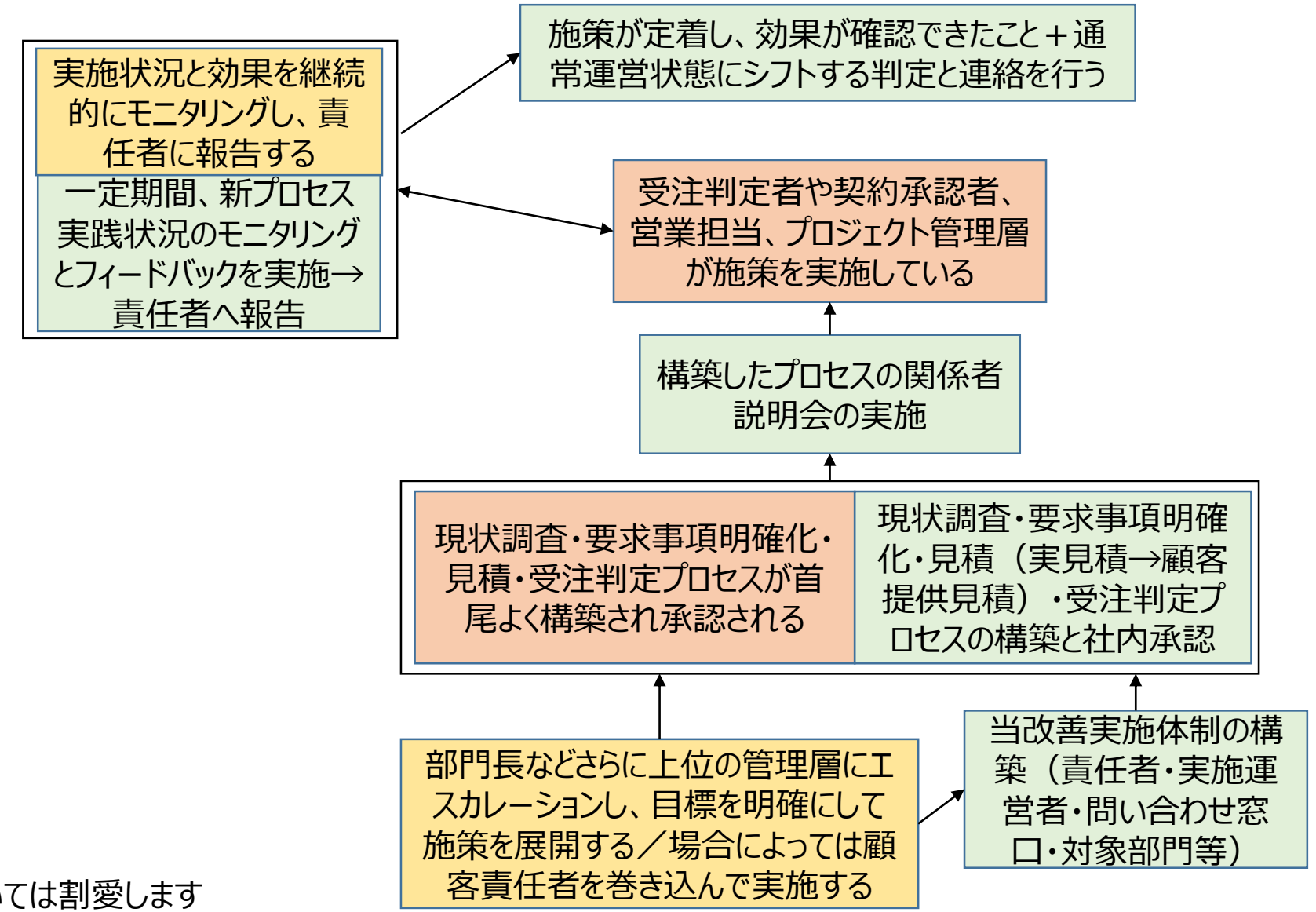
②リスク	③中間目標	④リスク対策
現状調査・要求事項明確化・見積（実見積→顧客提供見積）・受注判定プロセスの構築が難航して決まらない	現状調査・要求事項明確化・見積・受注判定プロセスが首尾よく構築され承認される	部門長などさらに上位の管理層にエスカレーションし、目標を明確にして施策を展開する
受注判定者や契約承認者、営業担当、プロジェクト管理層が施策実施に対して反対する／消極的対応に終始する	受注判定者や契約承認者、営業担当、プロジェクト管理層が施策を実施している	部門長などさらに上位の管理層にエスカレーションし、施策を展開する 場合によっては顧客責任者を巻き込んで実施する 実施状況と効果を継続的にモニタリングし、責任者に報告する

※以外施策については割愛します

SaPID適用事例

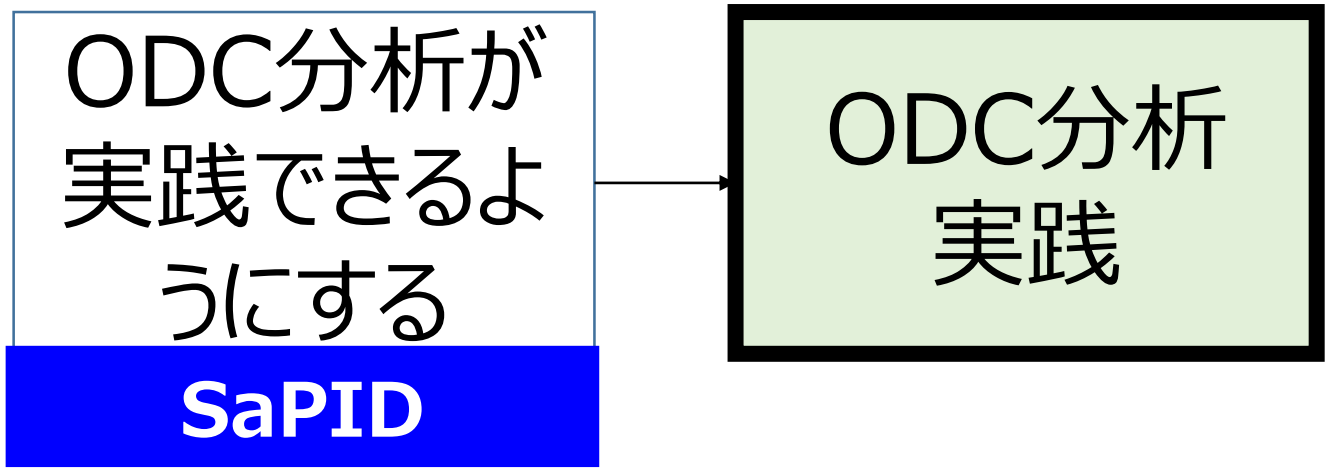
STEP6-2
再発防止施策の展開方法を具体化する

基本タスク
中間目標
リスク対策



※以外施策については割愛します

ODC分析が実践できるようにする【SaPID→ODC連携】



直交欠陥分類

何の準備もなくこれらを分類、分析して活動に活かせる組織はそう多くない

パフォーマンス改善は簡単ではない
プロセス・プラクティス不備の背景をも読み解く

- **改善に活用できるリソースはほとんどないのが実情**

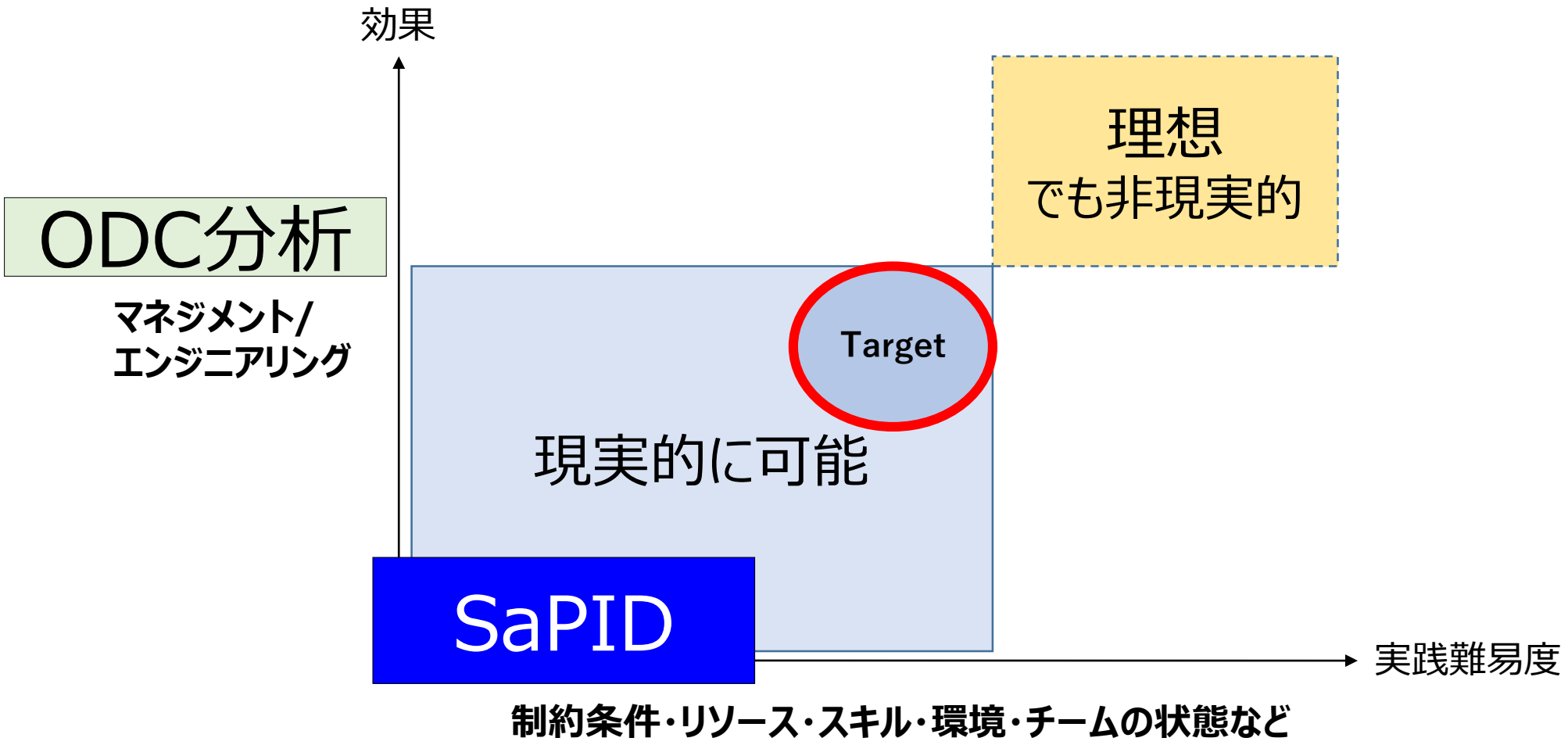
  プロジェクト計画で割り振られた役割、タスクを実践するための工数≒自らが持つ工数になっていることがほとんど。 少ない工数を最も効果的な対象に投下することが必要

  改善は「余裕があればやる」状態→工数に余裕があるプロジェクトはほとんどない。

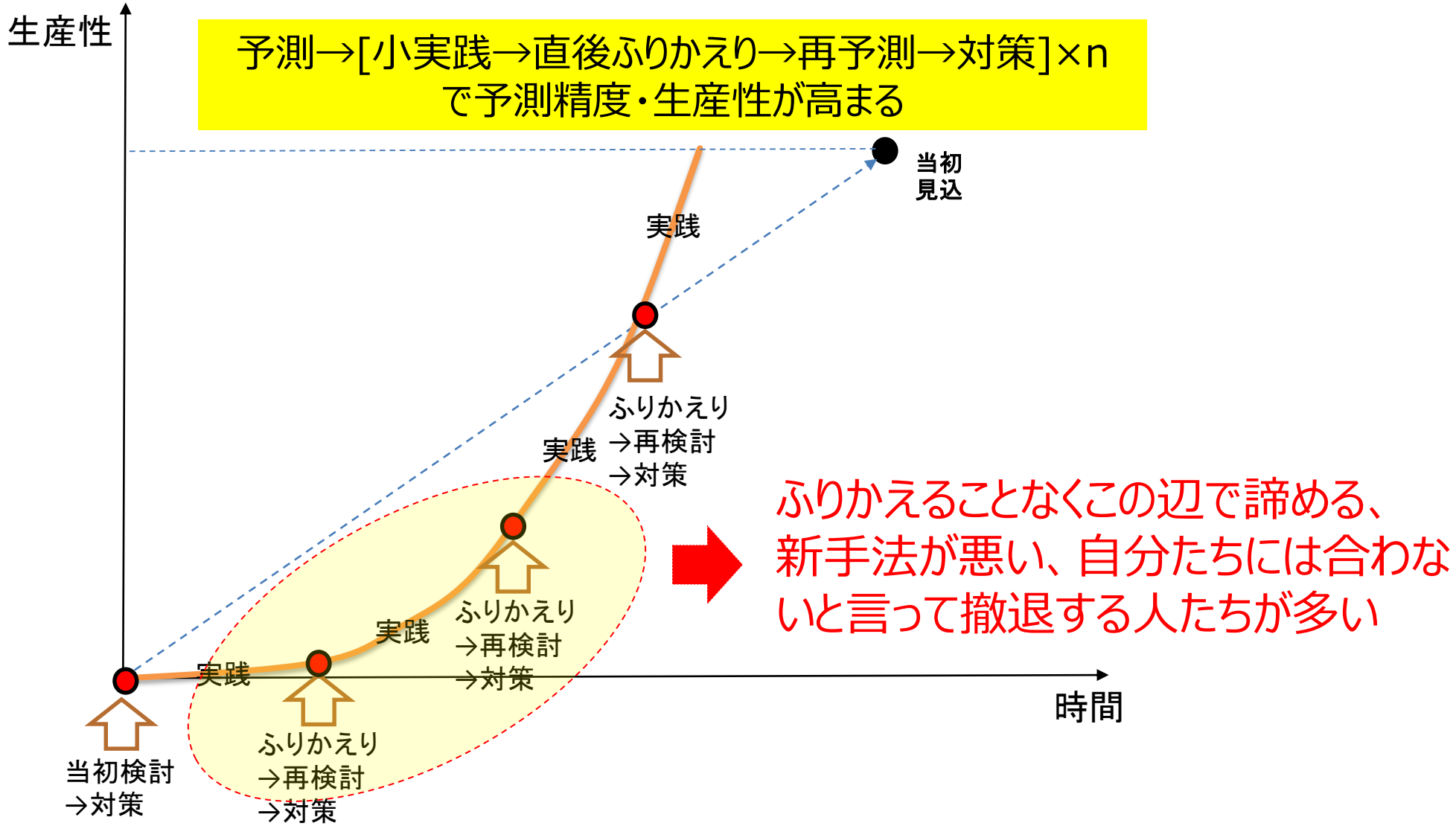
- **マネジメントやエンジニアリング実践の不備だけを改善しようとしても進まないことも多い** ～不備はなぜ発生しているか？が鍵

  例：「要件明確化段階で顧客に言われるがまま受身の対応」をしているチームに、要件開発プラクティス適用を提案してもうまく進まない・・・よくよく確認してみるとその実践を阻んでいた要因が「エンジニア評価指標が顧客クレームが少ないこと」であり、要件確認時に顧客にあれこれ質問すると「俺の仕事の邪魔をするな。そのくらい自分たちで考えろ。」と言われ、顧客によるエンジニア評価が下げられる環境であった。

新手法で「効果」だけ目指してもうまくいかないことが多い

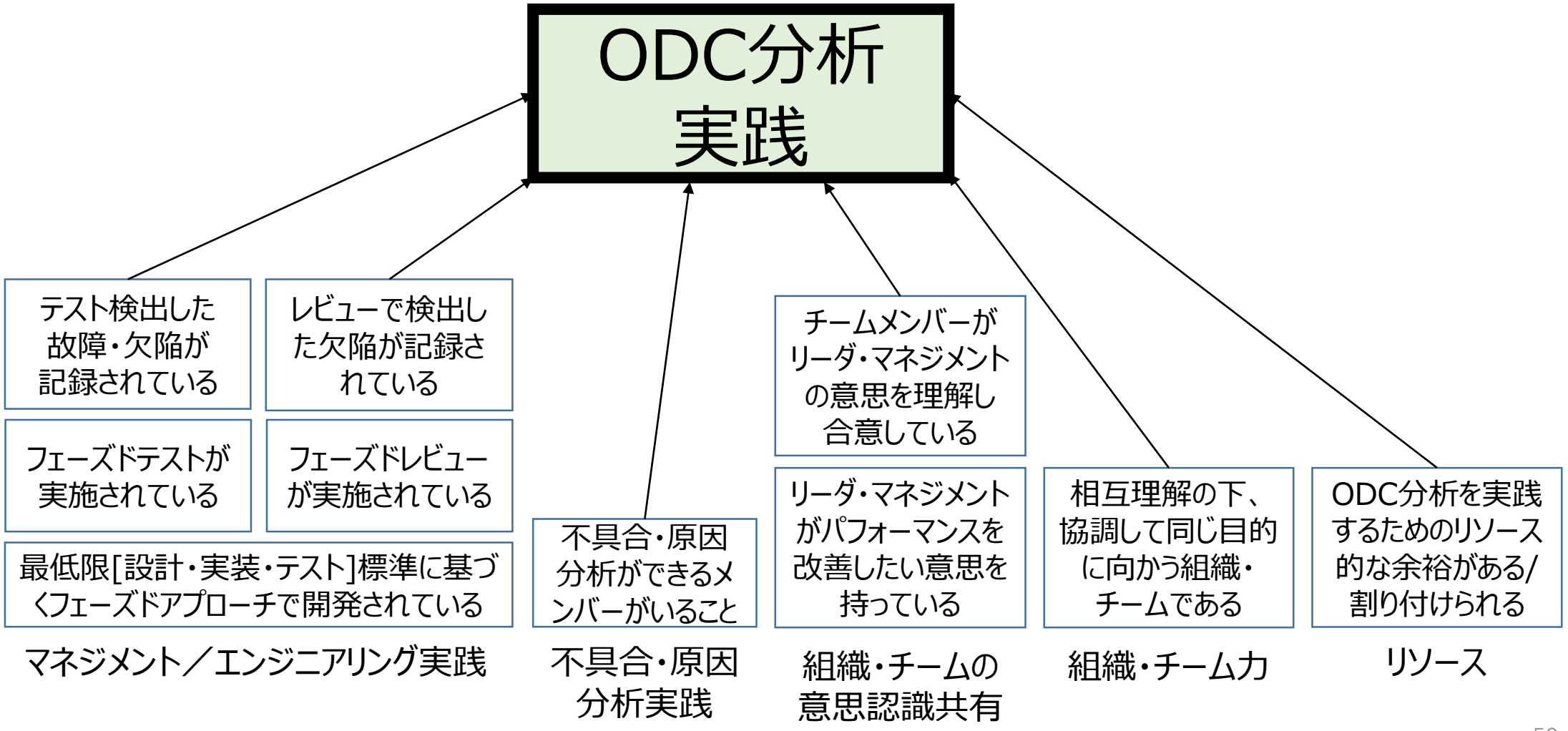


“新手法”を適用してから効果が出るまで



ODC分析を実践するためのMin条件例

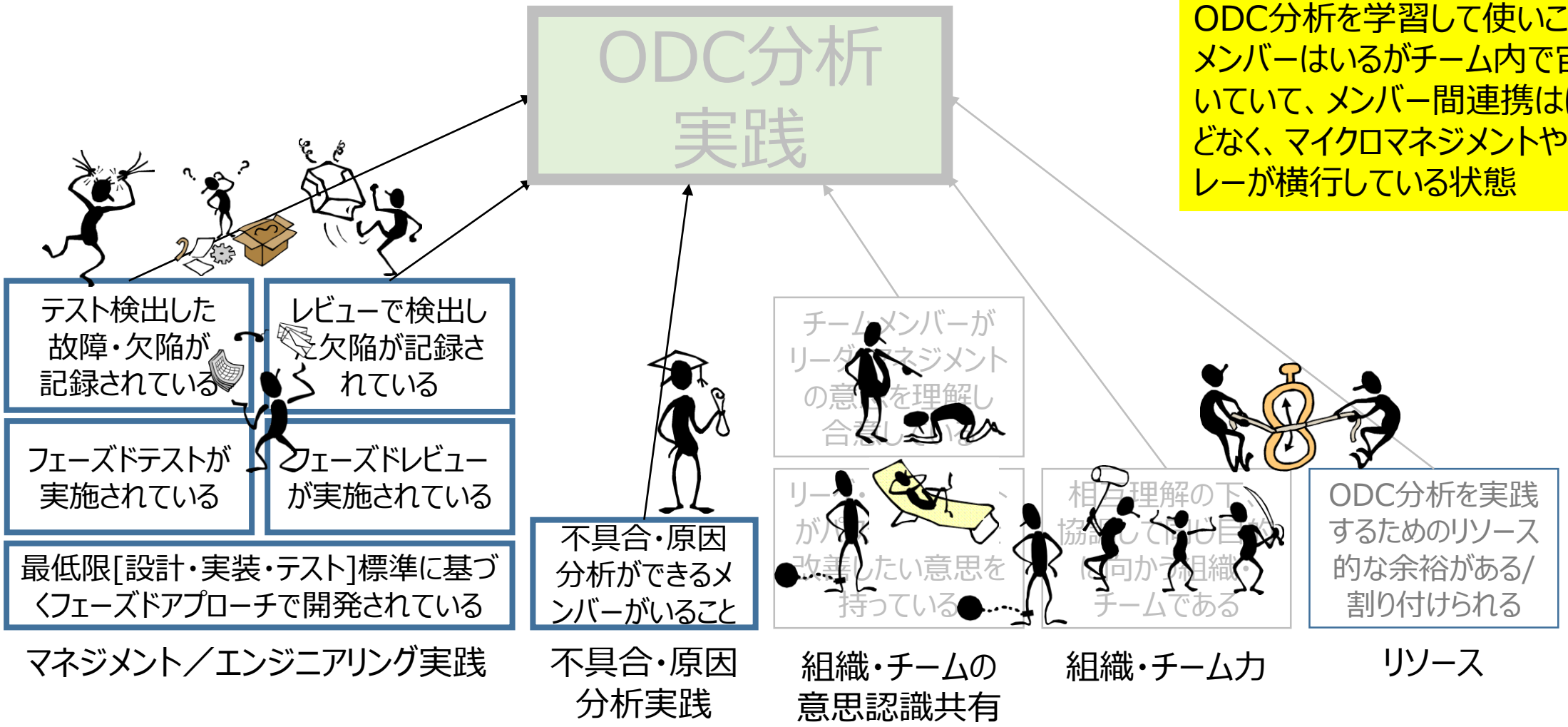
注意：Minであるためリリース後障害発生に起因するアクションは含んでいません



ODC分析がうまくいかないケース例1

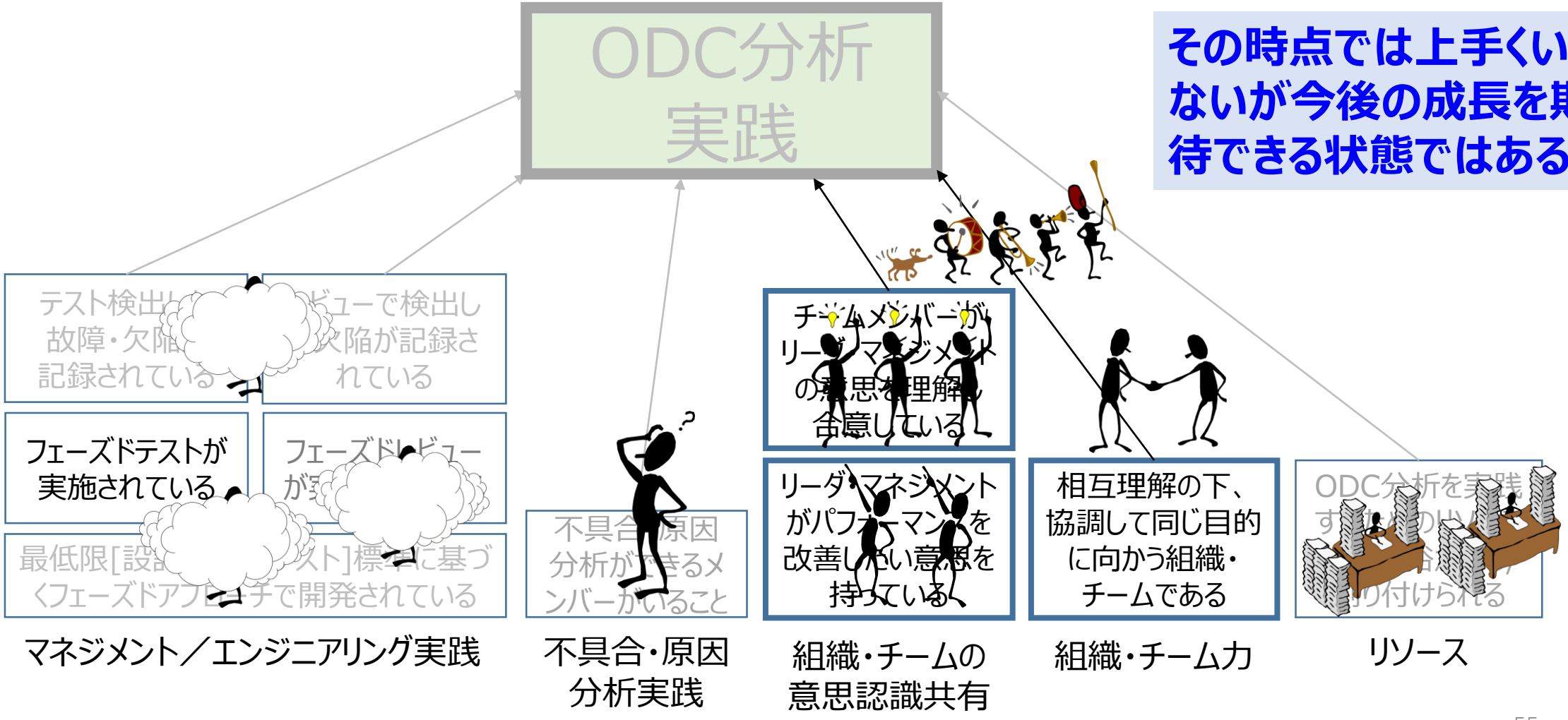
形や情報があってもチーム連携がないと上手くいかない

ODC分析を学習して使いこなせるメンバーはいるがチーム内で宙に浮いていて、メンバー間連携はほとんどなく、マイクロマネジメントや個人プレーが横行している状態

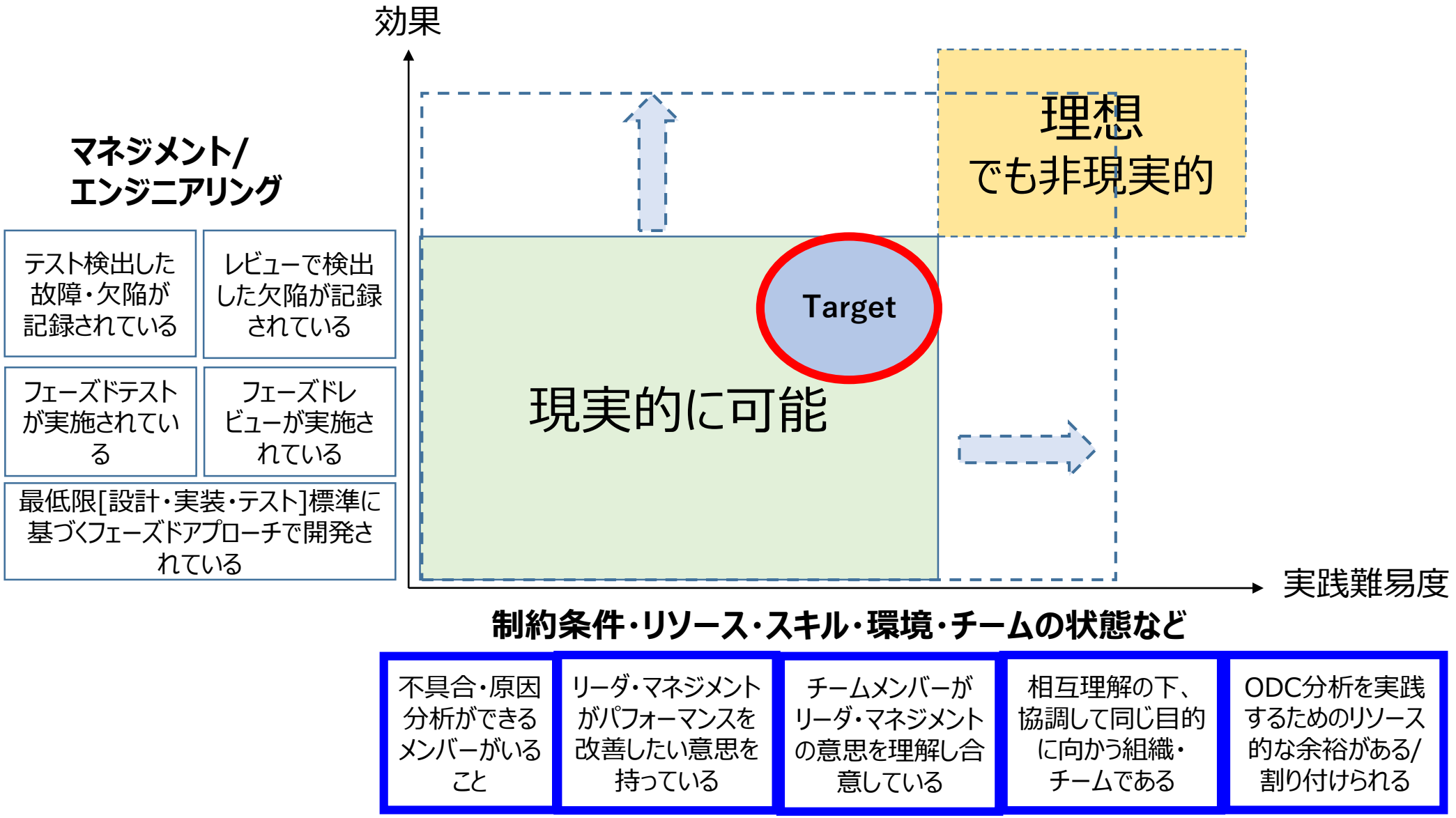


ODC分析がうまくいかないケース例2

協調的なメンバーで構成されていても形や情報がないと上手くいかない

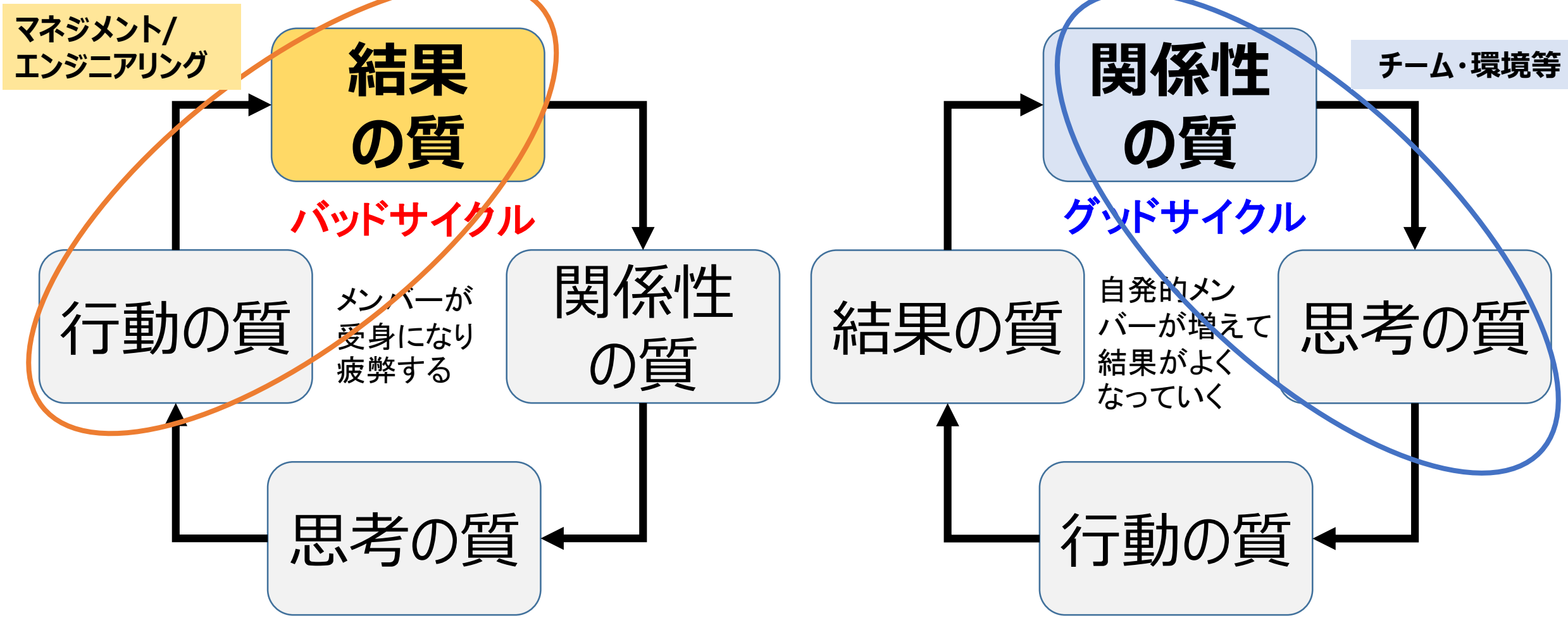


組織・チームによる実践が必要なら横軸の打開が鍵

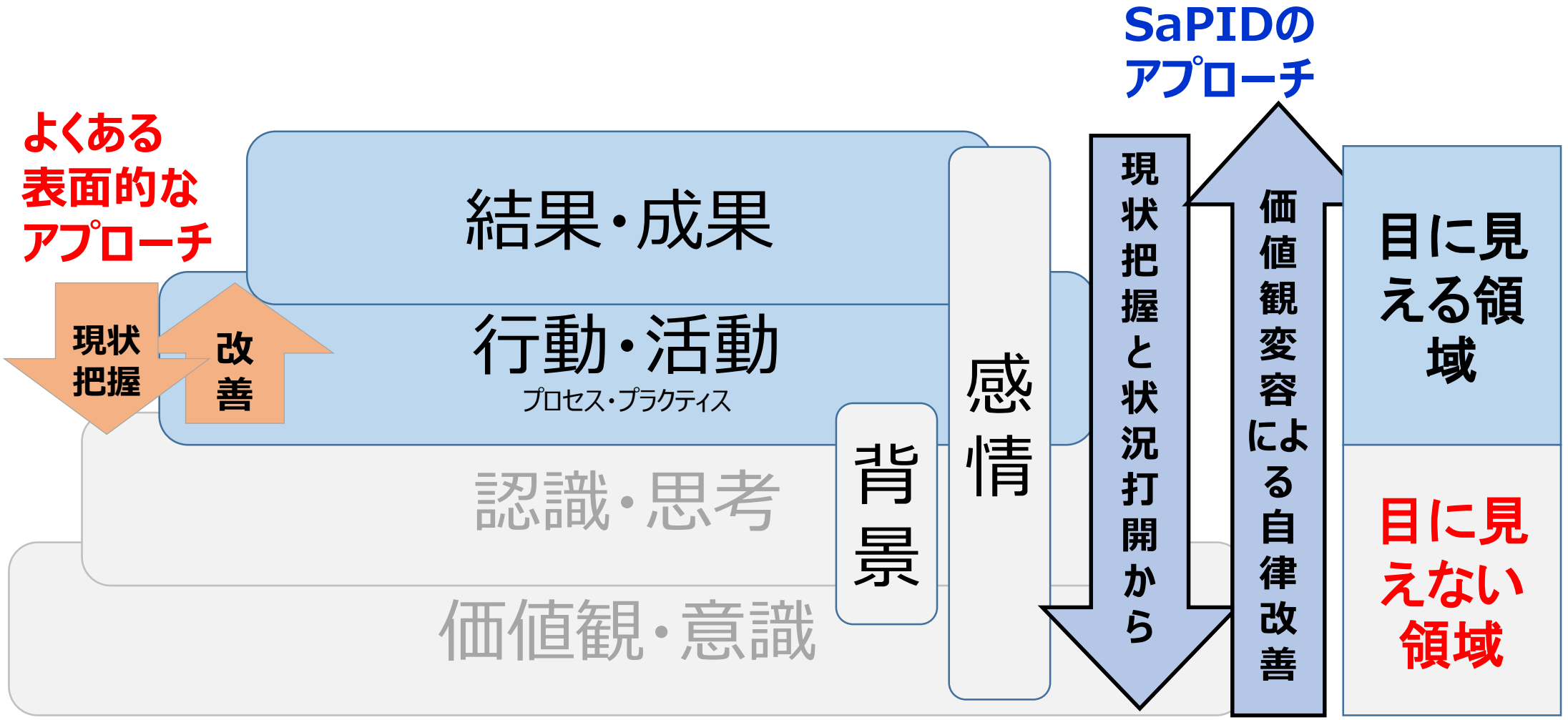


何から着手すべきか～成功循環モデル

マサチューセッツ工科大学ダニエル・キム教授が提唱



関係性の質は組織・チームの価値観から



横軸実践～SaPID流階段の上がり方

組織・チームの認識共有／組織力・チーム力向上のために

【段階5】総合的なリスクマネジメント実践による自律運営の実現

【段階4】予防処置をも実践し、最終目標を定め、徐々に規模・難易度を高めながら取り組む

【段階3】問題モデリングを高度化しながらパフォーマンス改善にも取り組む

【段階2】チームによる個別改善をあたり前化する

【段階1】毎日＋週次のふりかえりによりチームで問題発見・解決を実践する

チームパフォーマンス改善事例

問題の早期発見・解決＋リスク対応→手戻り減少＆リードタイム短縮

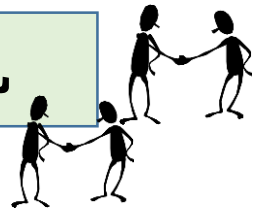
個別/チーム改善・リスク対策実践

- チーム内で発生する問題数の減少、および問題の高度化
  - → 問題発生＆解決報告＆協力メンバーへのお礼＋自己タスク分析結果と自己改善宣言／チーム改善提案＋リスク要因・手戻りなどのムダ事項と対策提案　など

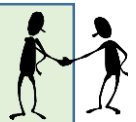
- 問題発生＆解決報告＆協力メンバーへのお礼＋自己タスク分析結果／チームタスク分析

問題発見・解決中心

- チーム内問題発見～解決リードタイム短縮
  - → 問題発生＆解決報告＆協力メンバーへのお礼

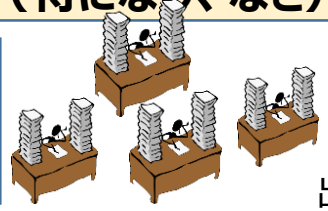


- 問題記述・問題解決報告＋協力してくれたメンバーへのお礼



その日の問題その日のうちに！ふりかえり導入

- ぼんやりした境遇説明や感想（ある意味小学生的なw）／他人事のように感じる外的責任への言及／未記入（特になし、など）
- 個別対応者の集まりふりかえりなしパフォーマンス↓



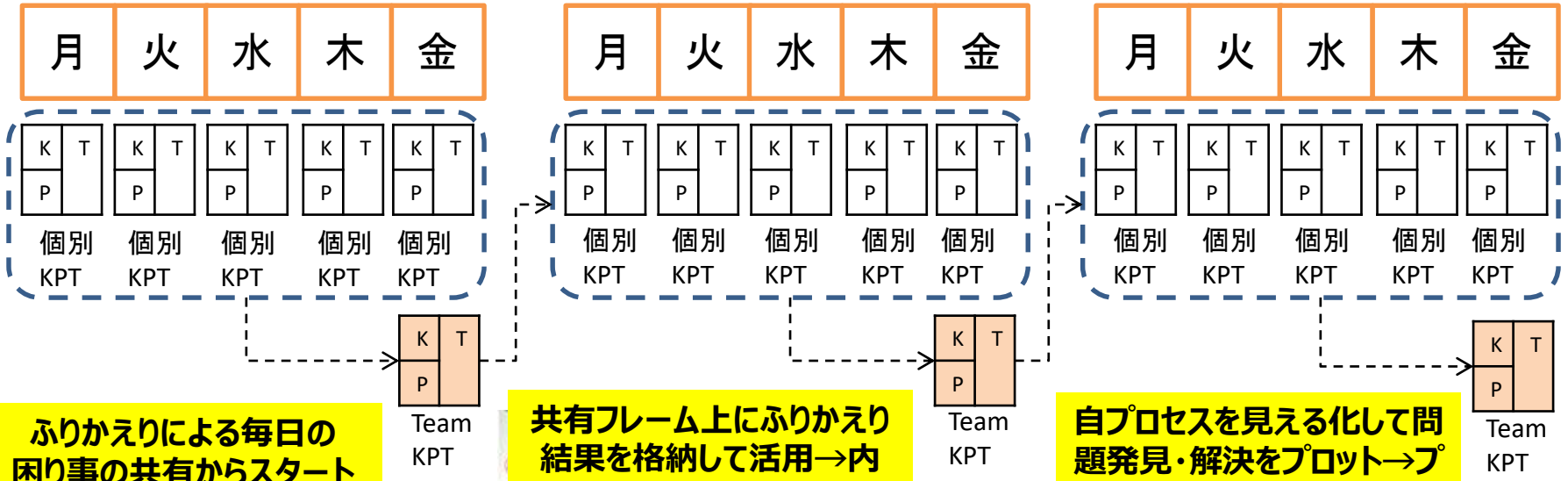
記載内容の段階的高度化＝モノゴトの見方・思考の変化

- 問題解決型→リスク・先読み型へ
- 解決手段提示→問題事象中心へ
- 個人タスク型→チーム問題発見型
- 感情面偏重→エンジニアリング的内容へ
- ぼんやりした概要記述→具体的詳細記述

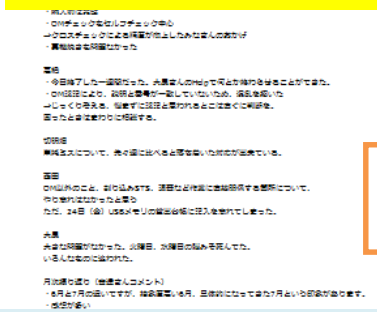
出典：SPI Japan2015　自律型プロジェクトチームへの変革アプローチ事例

出典：SPI Japan2015　自律型プロジェクトチームへの変革アプローチ事例

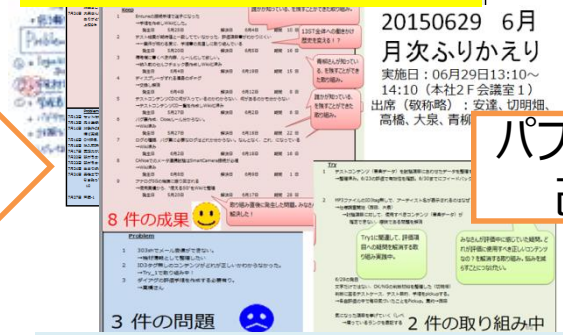
日次・週次・月次ふりかえり実践→問題発見・解決を徐々に高度化→パフォーマンス改善実践へ



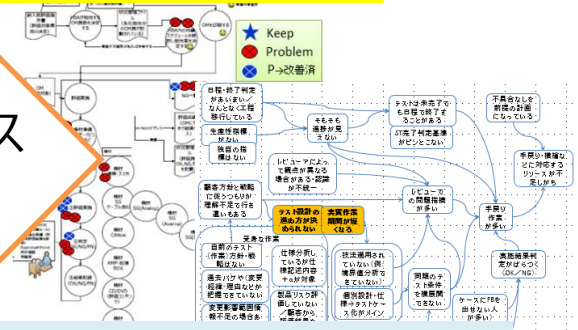
ふりかえりによる毎日の困り事の共有からスタート



共有フレーム上にふりかえり結果を格納して活用→内容をメトリクスで管理



自プロセスを見える化して問題発見・解決をプロット→プロセス見直し実践



パフォーマンス改善へ

事前観察→暗黙の価値観を推定

チームの暗黙の価値観を見抜き問題提起→目標共有
チーム問題発見・解決の実践

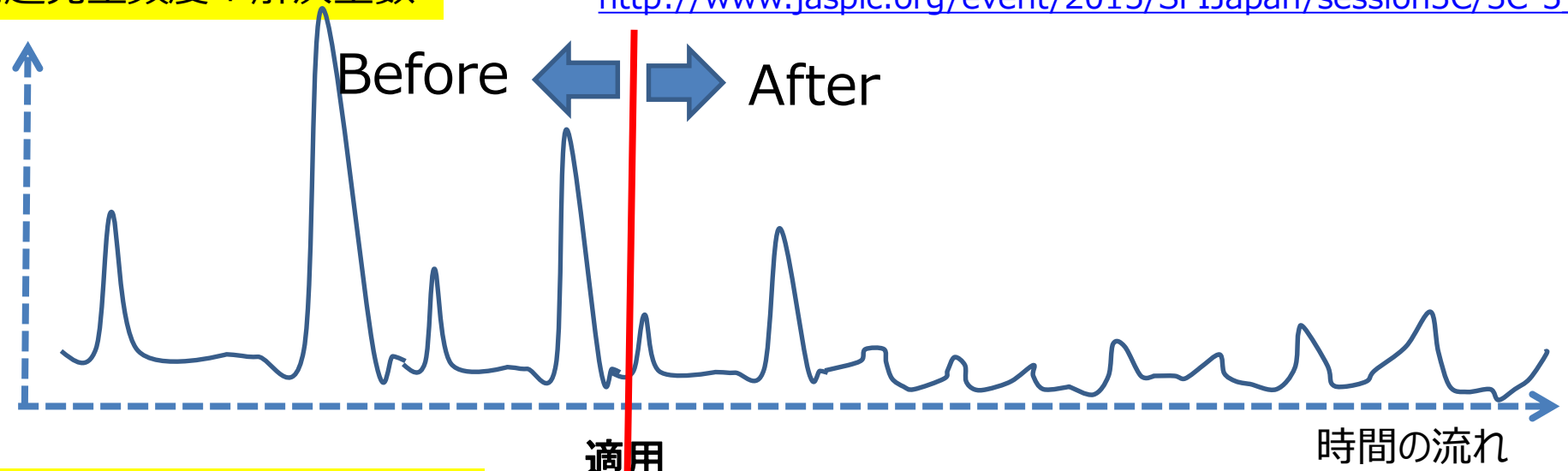
達成目標を段階的に上げ、チームパフォーマンスを上げる

必要なツールを使いこなせる自律運営チームに仕上げる

適用後のパフォーマンスの変化例

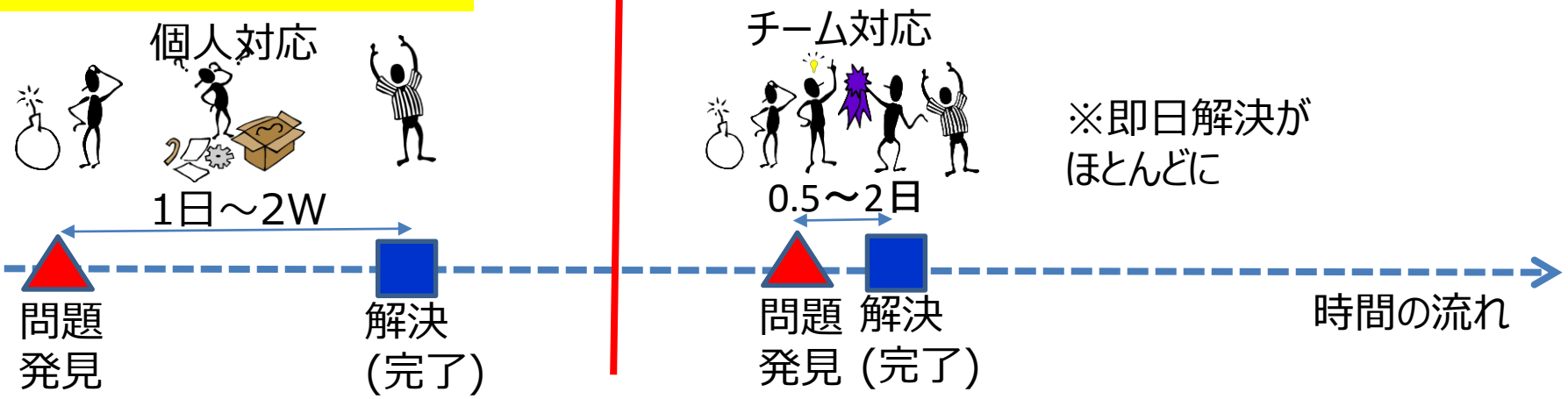
出典：SPI Japan2015　自律型プロジェクトチームへの変革アプローチ事例
http://www.jaspic.org/event/2015/SPIJapan/session3C/3C-3_ID012.pdf

■問題発生頻度＋解決工数



適用開始

■問題発見－解決リードタイム



チームパフォーマンス改善試行の観察結果

比較事項	Before	After
ミーティング	ほとんどなし。自作業に専念できる。	ミーティングに時間が割かれるようになった。緊急時や作業立て込み時は時間のやりくりが難しくなった。
大事にすること	各自が役割を果たせばよい。	個人ではなく“チームとしての効率の良さ”として捉える。
ノウハウ	属人的。個人にノウハウが集中。 初めて実施する作業の立ち上がりが遅く、実務・管理両面で苦労していた。	チームとして共有。
問題発見・解決	個人対応で場当たり的。 他者の状況が見えず対応が後手後手。	どのような問題が起こっているのかが、全体で把握できる。 問題発生したらメンバーで解決する。 早めに解決策を取れることが増えた。
改善	指示がなければ特段実施しない。	これまで課題だった再発防止ができた。 日々必要な改善を実践している。 問題発見・解決と分けて実践していない。同じ困りごとが発生しそう→改善を実践
生産性など	新規参入者や初着手作業でノウハウが不明でミスが多く、生産性も悪い。	スピードが上がった。最終チェックNGが減少。新規参入者が短時間でミス低減ができ生産性の向上が計れる。

出典：SPI Japan2015　自律型プロジェクトチームへの変革アプローチ事例
http://www.jaspic.org/event/2015/SPIJapan/session3C/3C-3_ID012.pdf

過去の体験・経験から学ぶ
個別改善→パフォーマンス改善への発展

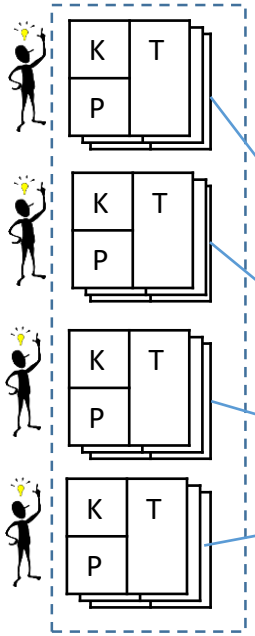
日次個別ふりかえり
とチーム共有・活用

個人で毎日3分
ふりかえり

個人で毎日3分
ふりかえり

個人で毎日3分
ふりかえり

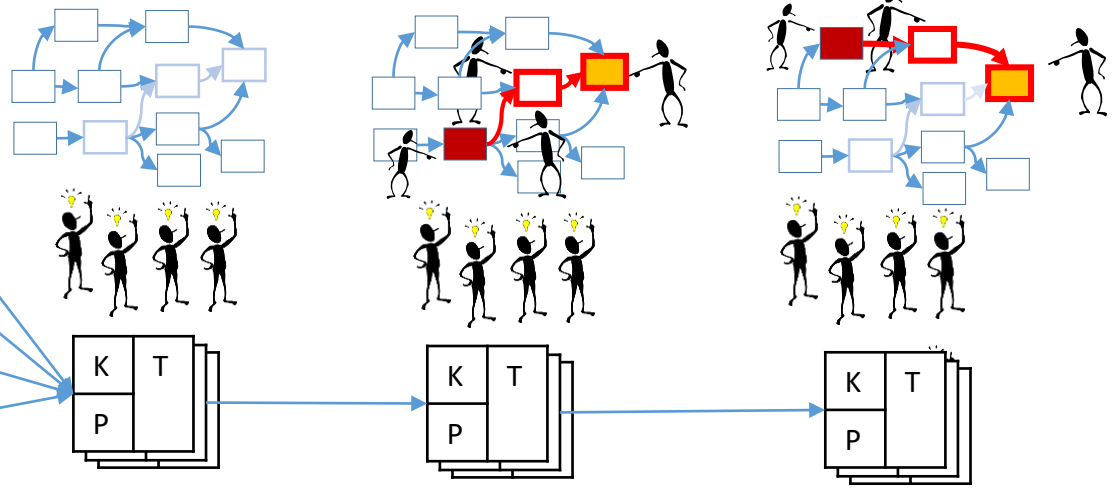
個人で毎日3分
ふりかえり



週次Team
集合ふりかえり

月次・Phase
集合ふりかえり

四半期・Final
集合ふりかえり



Teamとして毎日共有し、
主に日々の問題解決・
個別改善に活用

Teamとしての共通性導出・活用（プロセス改善）
ふりかえり結果情報のサマリ→構造化分析による全体俯瞰
ボトルネックの解消／最も効果的な改善ポイントの特定による
パフォーマンス改善の実践

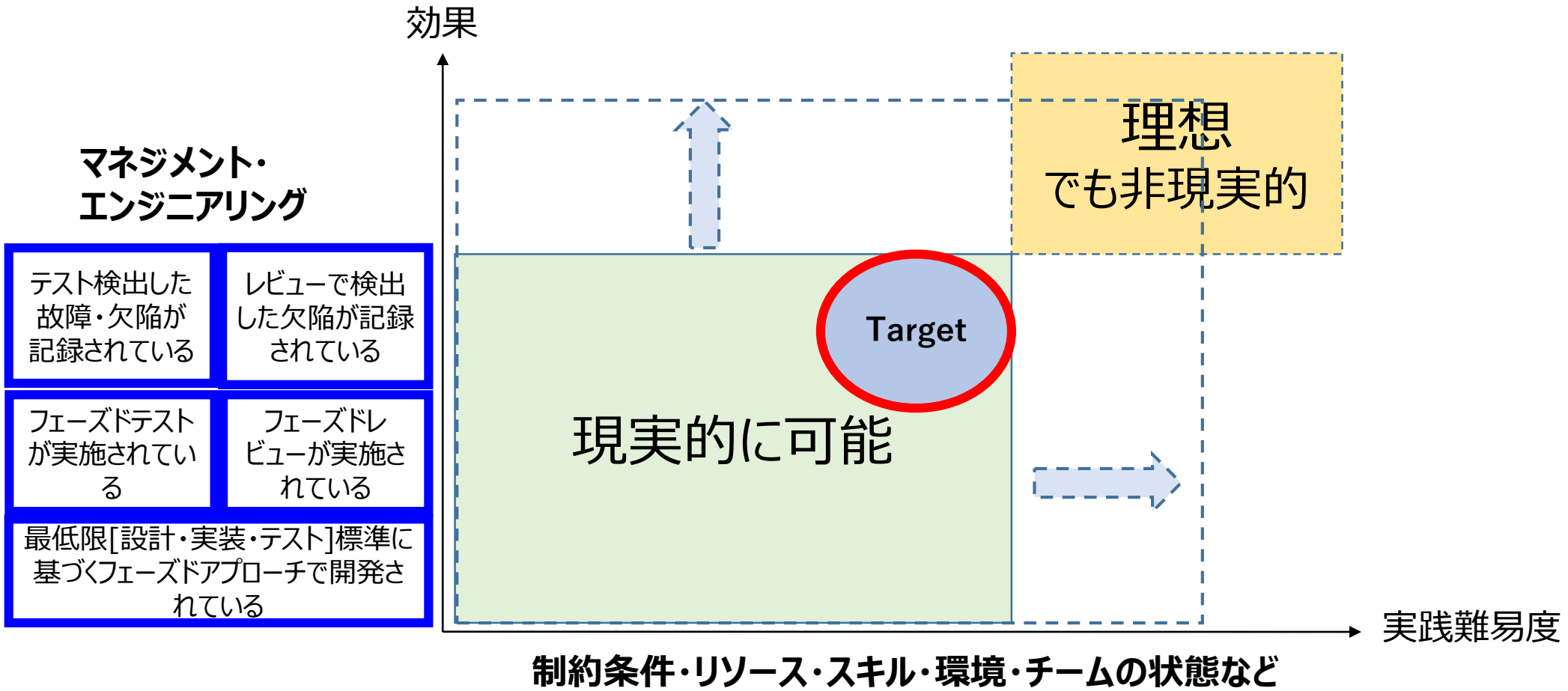
横軸実践上の注意事項

- [計画→実践→ふりかえり]のサイクルタイム
  短い方がよい
- 当手法を無効化する要因
  - 問題を共有できない文化
  - 圧の強い人、失敗への辛辣な批判が得意な人
  - 自分のことしか考えない自己中心的メンバー
  - マイクロマネジメント実践者
  - 不適切な介入や指示・命令（上級マネジメント）　などなど

Waterfall的に長い期間で
進めるとうまくいかないことが多い

メンバーのやる気を削ぐ、チームワークを無効化するのは意外と簡単

横軸実践中＝“問題発見・解決”により縦軸が動き出す



マネジメント・エンジニアリング

テスト検出した故障・欠陥が記録されている	レビューで検出した欠陥が記録されている
フェーズドテストが実施されている	フェーズドレビューが実施されている
最低限[設計・実装・テスト]標準に基づくフェーズドアプローチで開発されている	

制約条件・リソース・スキル・環境・チームの状態など

不具合・原因分析ができるメンバーがいること	リーダ・マネジメントがパフォーマンスを改善したい意思を持っている	チームメンバーがリーダ・マネジメントの意思を理解し合意している	相互理解の下、協調して同じ目的に向かう組織・チームである	ODC分析を実践するためのリソース的な余裕がある/割り付けられる

効果と効率の両面を兼ね備えたソフトウェア開発の原則

☑関係者でゴールイメージ(実現しようとしている価値)を明確にし、共有する。

☑欠陥をできるだけ作り込まない。

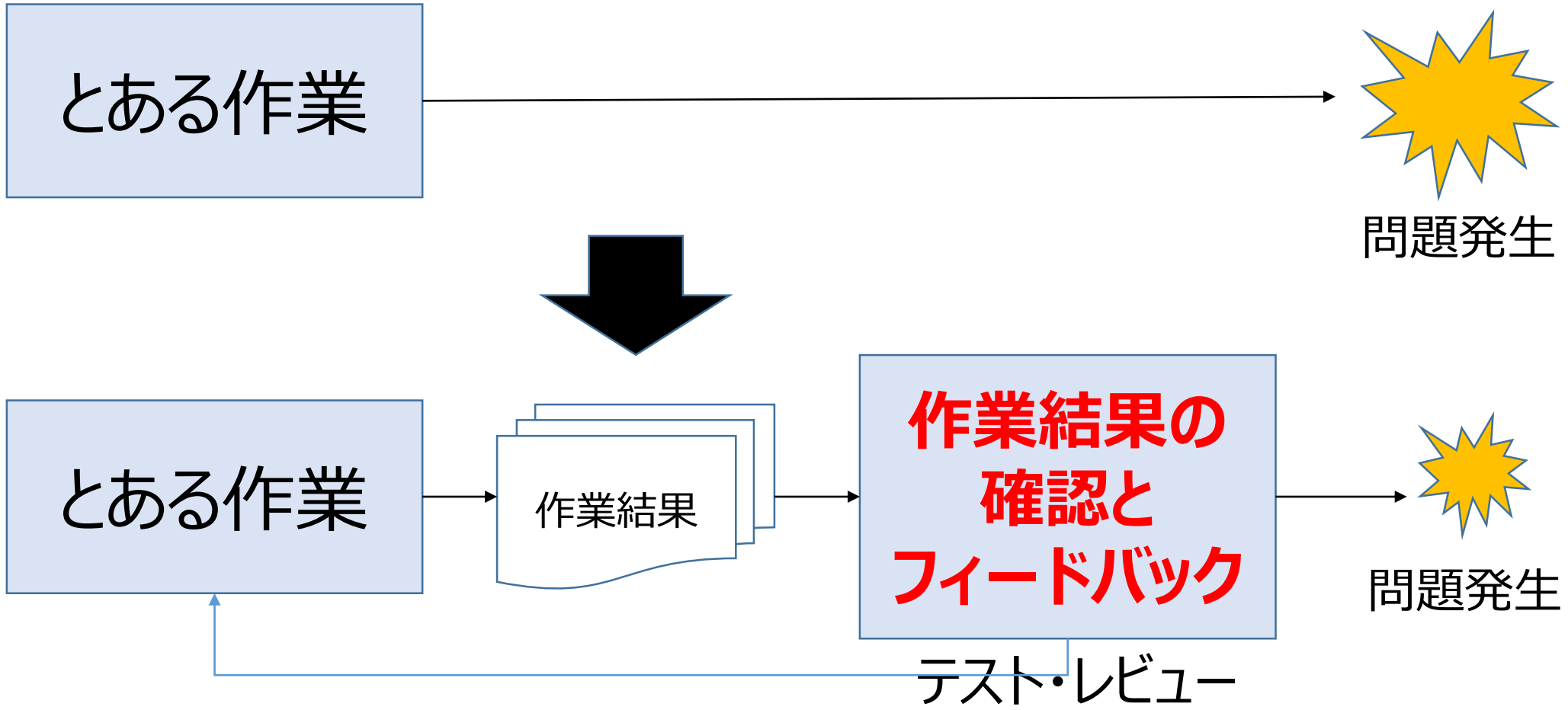
でも完璧な人間はいない（人間は間違える）

☑欠陥作り込み～検出・修正完了までのリードタイムを最小にする（早期問題発見・解決）

自分たちの仕事をよりよく進められるように、そして顧客にもっと喜んでもらえるように変えていこうとすると、この原則に沿って進めることになる。

人間は間違う

仕損じたことを把握・認識するところから



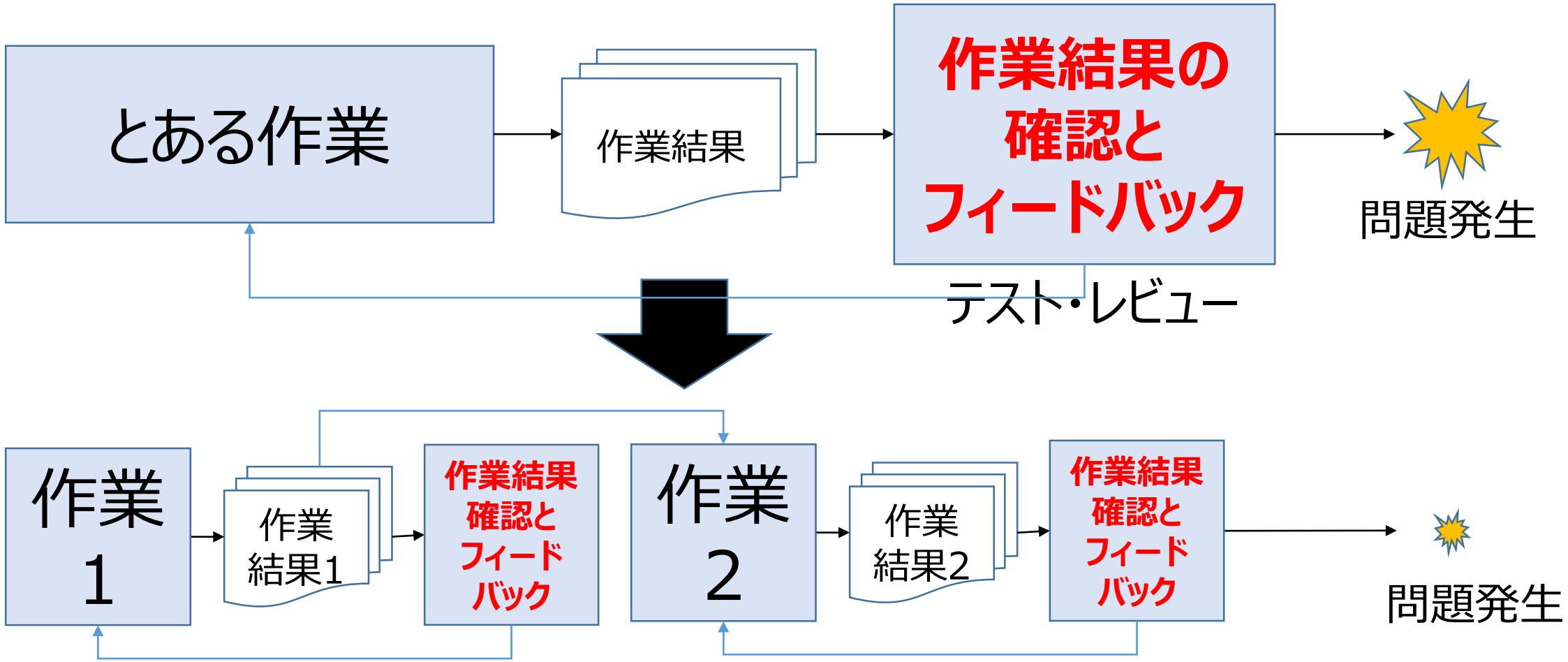
ソフトウェアのVUCASへの対処

Volatality/変わりやすさ、Uncertainty/不確実さ、Complexity/複雑さ、
Ambiguity/曖昧さ、Scalability（大規模さ）

- 難しいことは分割して考えよう。
- 大きなものは階層化して分割しよう。
- 不確実な時は考えすぎず取りあえずやってみて、素早くどんどん改善しよう。
- 同じようなことはまとめて済まそう。
- 端っこは普通と違うから気をつけよう。
- 手順だけ暗記してもいいことないよ。

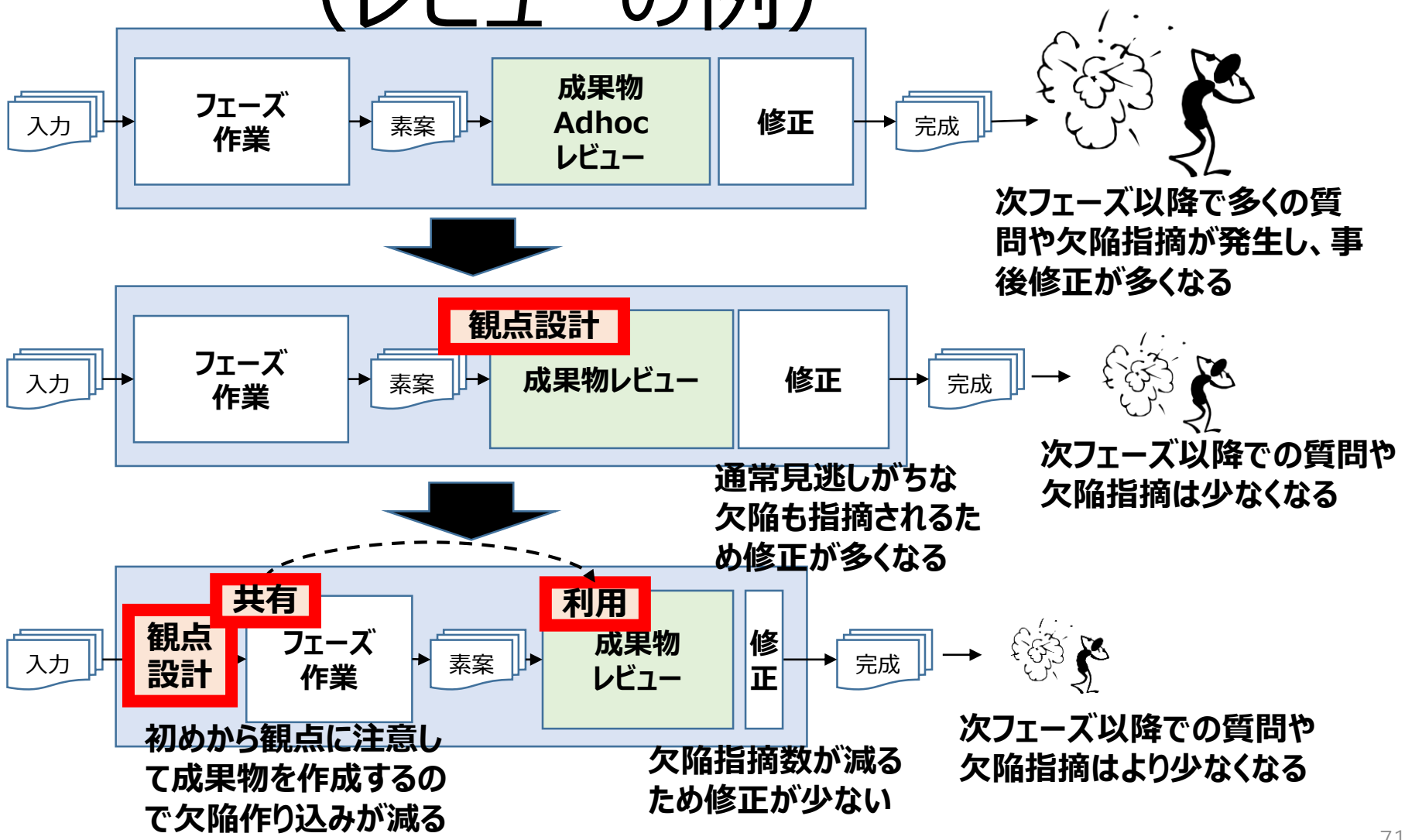
テストの未来　テストエンジニアの「これまで」と「これから」を考える　より
https://www.veriserve.co.jp/asset/approach/column/test-technique/post-1.html

規模が大きく複雑な作業＝ソフトウェア開発を うまく取り扱うように整理する



フェーズ内のシフトレフトライクなこと（レビューの例）

JaSST Review2023「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ー」より



最初から品質を作り込む

ソフトウェアテスト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。

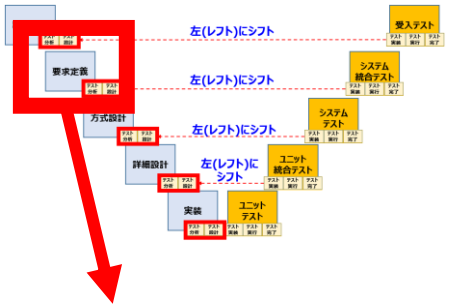
SDLCの後半に発生する故障が少なくなるため、品質コストは削減される。

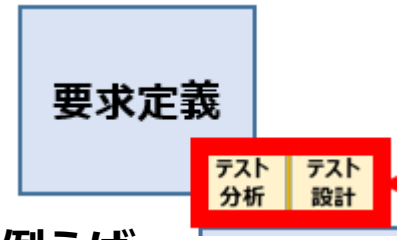
早い段階で欠陥を見つけるために、静的テストと動的テストの両方をなるべく早い時期に開始すべきである。

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

JaSST Review2023「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ー」より

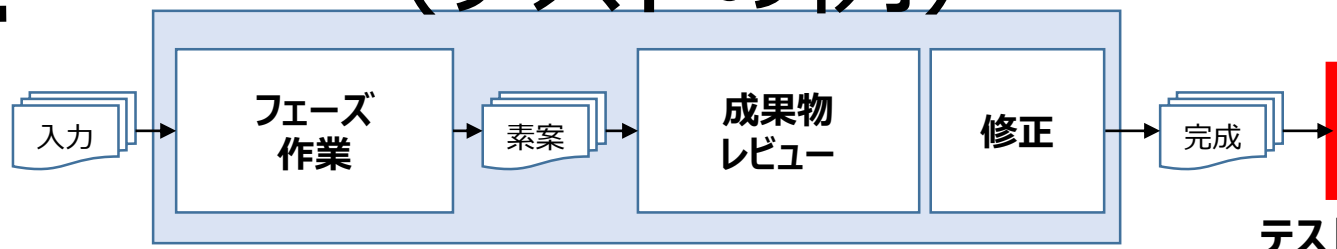
同一フェーズ内でのシフトレフトライクなこと（テストの例）



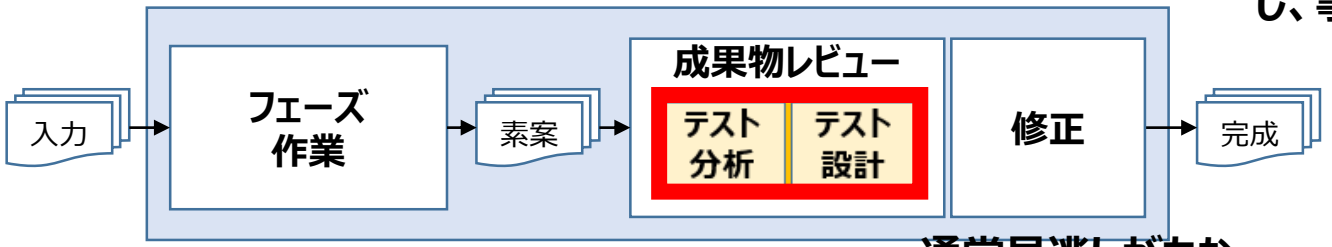


例えば
「要求定義フェーズ」

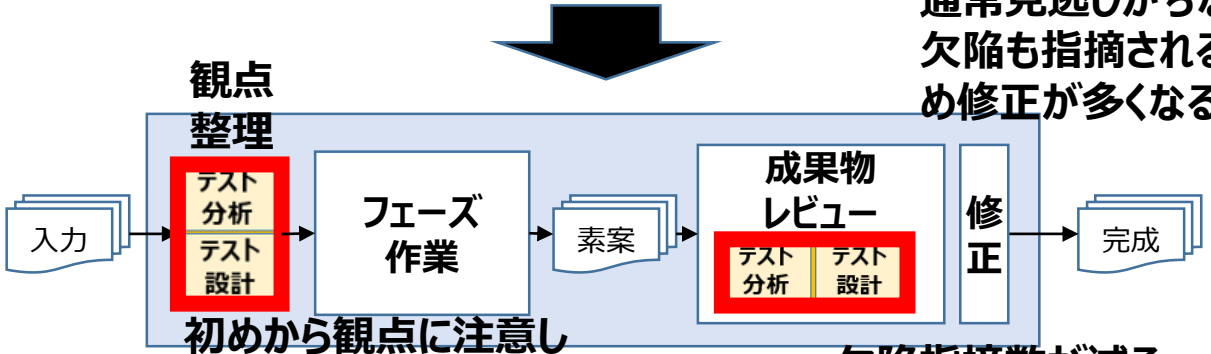
最初から品質を作り込む



テスト分析、設計過程で多くの質問や欠陥指摘が発生し、事後修正が多くなる



通常見逃しがちな欠陥も指摘されるため修正が多くなる

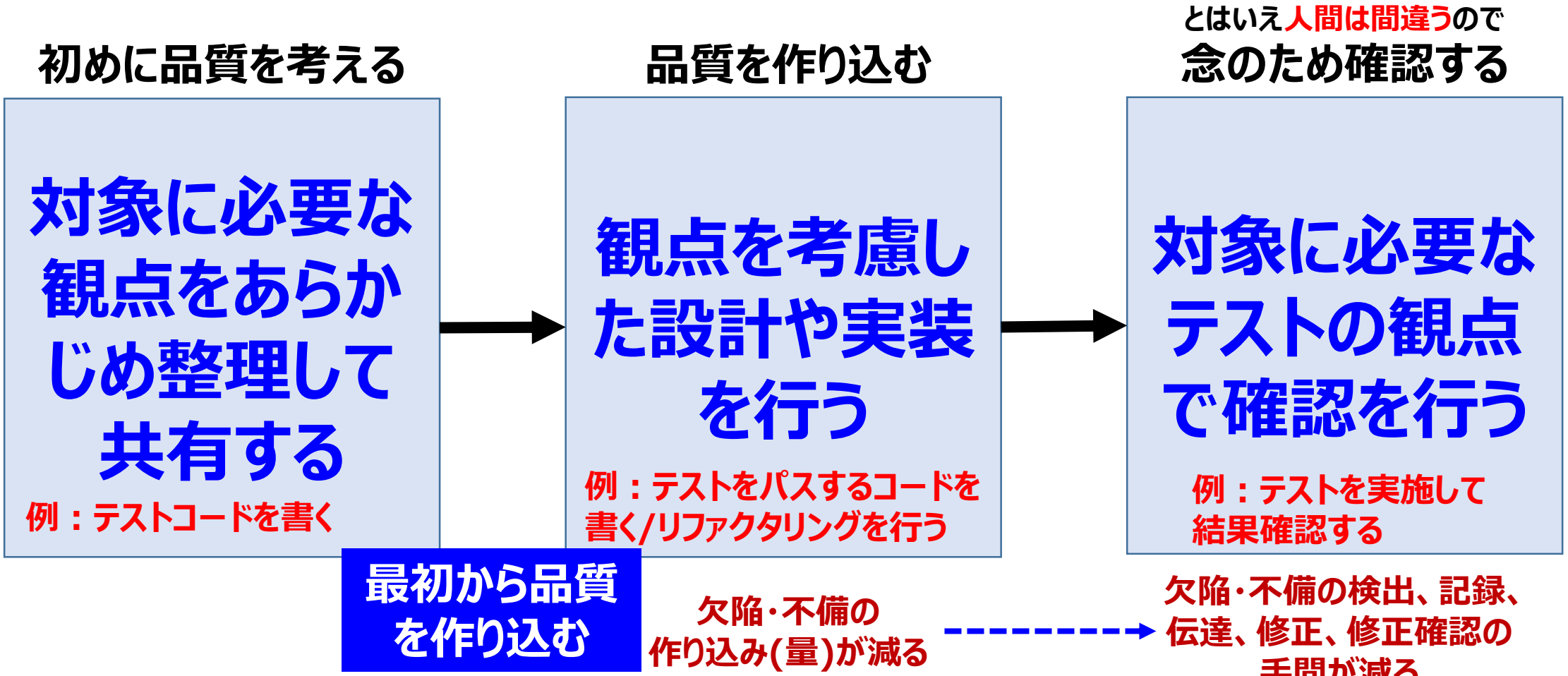


初めから観点に注意して成果物を作成するので欠陥作り込みが減る

欠陥指摘数が減るため修正が少ない

JaSST Review2023
「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ー」より

シフトレフトでやっていること(原理)



結局これらは“ソフトウェアエンジニアリング”のより適切な実践

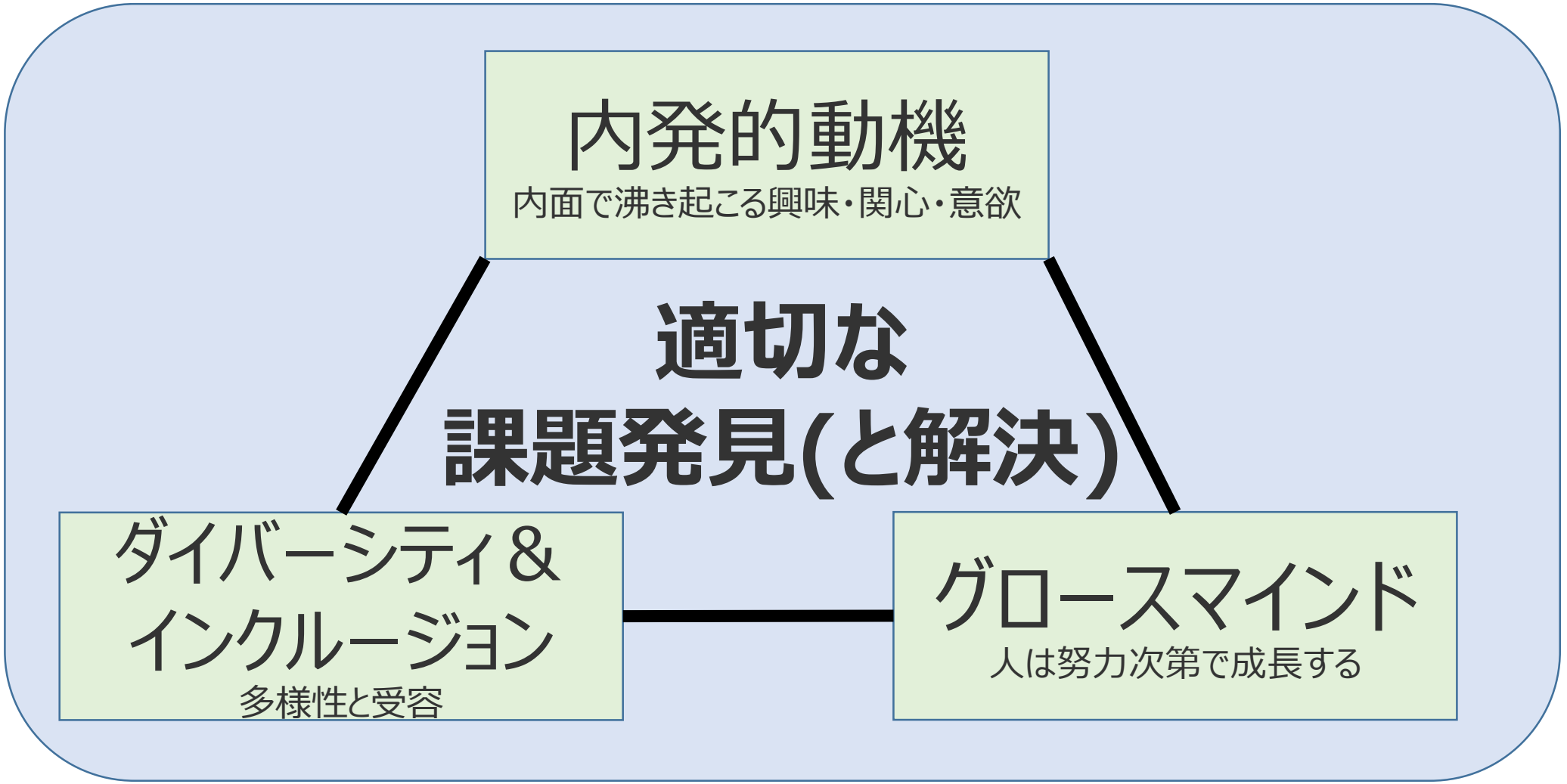
- “ソフトウェアエンジニアリング”に対する私の理解

ソフトウェアの開発・保守・運用を確実に、合理的に、可能な限り容易に進めるためのノウハウをまとめたもの。

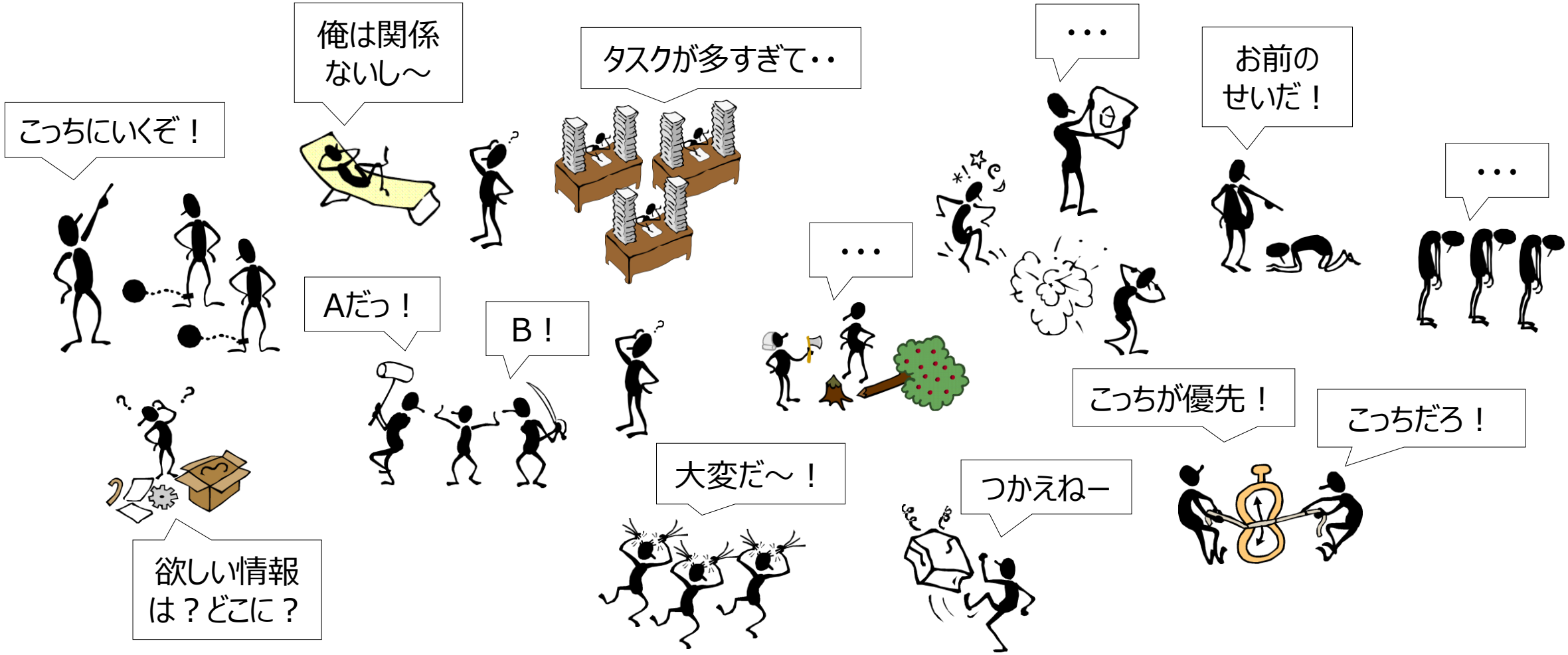
制約事項の中で、自分たちの仕事をよりよく進められるように、そして顧客にもっと喜んでもらえるように変えていこうとするとソフトウェアエンジニアリング実践の適切化に向かう。

チームパフォーマンス向上のポイント

チームパフォーマンス向上のためのポイント



組織・チームには多くの問題が・・・・・・
しかし、その解決に使えるリソースには限りがある



適切な課題発見

現状を構造的に理解して もっとも適切な課題を特定する

作り込み要因

流出要因

すべての問題に個別バラバラに対処するようなことはしない！

要求定義

- 現状分析結果がどこにも存在しない
- 要求なのか仕様なのかが不確かな記述形態
- 変更要求を処理する際に仕様への展開に一部失敗していた
- レビュー対象の記述内容に反応するだけのアドホックレビューで見逃されていた

基本設計

- 複数の要求をそのまま設計仕様化し、詳細化不足になっている
- 設計仕様のみ記述する形式となっている (どの要求に対する設計仕様なのかは要求仕様書と突合しないとわからない)
- レビュー対象の記述内容に反応するだけのアドホックレビューで見逃されていた

機能テスト

- テスト設計後の仕様変更への対応がコード修正のみとなっていた
- 基本的に*CPM法*でテスト設計を行っていたが、テスト設計後の仕様変更発生箇所については未対応であった
- FT検証観点が設計仕様を網羅できていない事項あり
- FT設計結果レビューが未実施であった

STでFunction不具合が想定以上に残存 → Function不具合が出続ける

適切な“課題”の発見

“課題”＝多くの問題の中に存在する“解決すべき問題”

- The right answer to the wrong problem is very difficult to fix.

間違った問題への正しい答えほど、始末に負えないものはない

Peter.F. Drucker

- 問題解決力とは、優れた解決策を出す能力ではない。

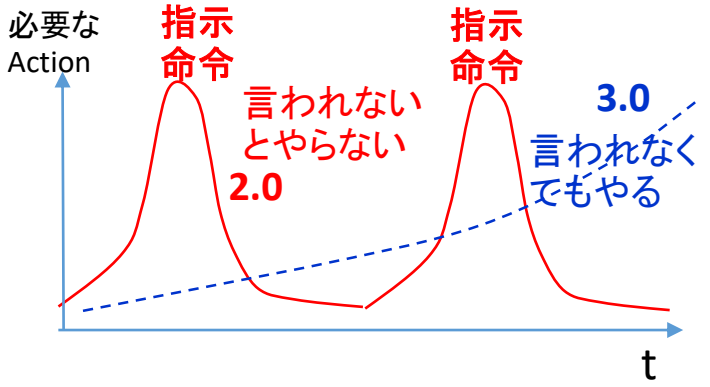
本当の問題に気づく能力のことである。

「モチベーション3.0」ダニエル・ピンク

- **モチベーション*2.0***(アメとムチなど外発的動機付け)

から**モチベーション*3.0*(内発的動機付け)**という2つの動機付け）へ

- モチベーション2.0の管理で報酬を用意すると、管理される側はその報酬のために短期的にやるだけになり、「自律性(オートノミー)」を失う。
- アメとムチの致命的な７つの欠陥
  1. 内発的動機づけを失わせる。
  2. かえって成果が上がらなくなる。
  3. 創造性をむしばむ。
  4. 好ましい言動への意欲を失わせる。
  5. ごまかしや近道、倫理に反する行為を助長する。
  6. 依存性がある。
  7. 短絡的思考を助長する。



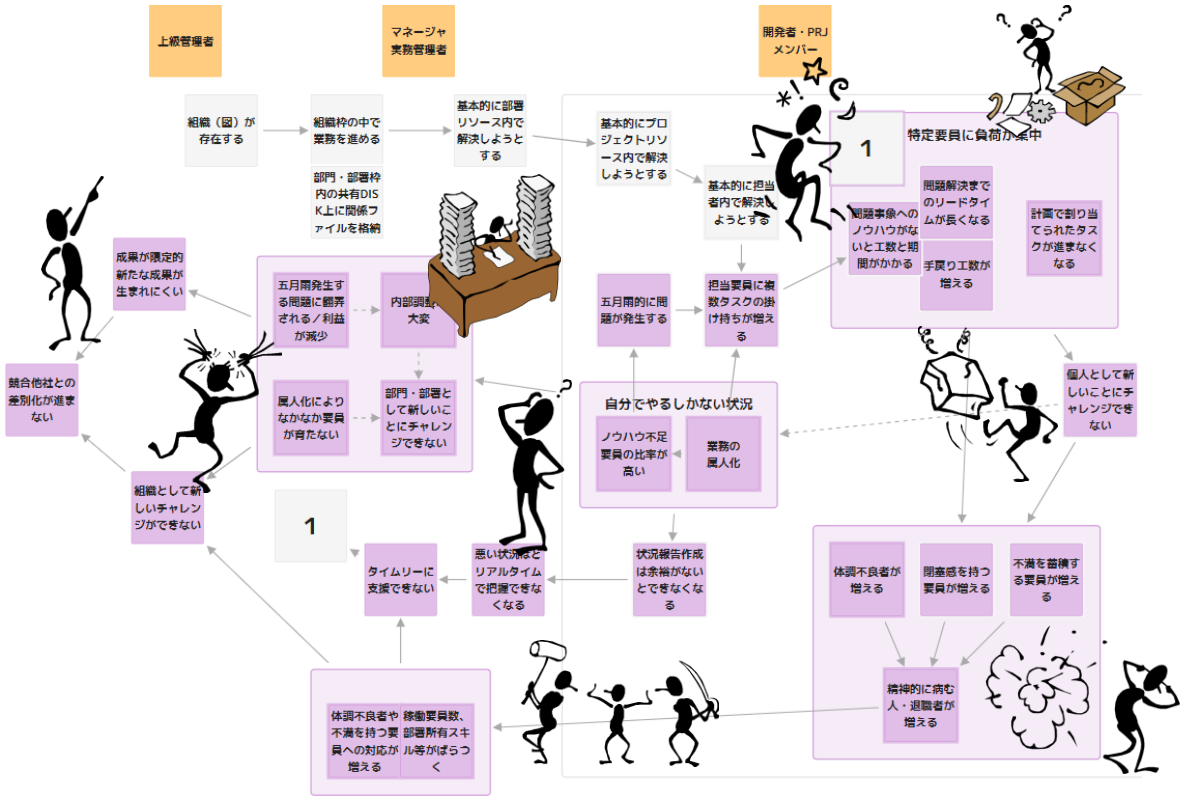
「ODC分析を導入しろ」「ODC分析を導入することになりました」はモチベーション2.0？3.0？

SaPID～身近な困り事や問題を自ら見える化しつつ目指すゴールと連携させて解決していく

- モチベーションの源泉

自分の身近な困り事や問題が解決するのはうれしい。

チームワークを高めながら自分事の範囲を徐々に広げていく



取り組み過程で発見する／あらためて認識すること

□普段の仕事で目指していることは何かそれはどのような価値があるのか
□自らの仕事の過程や結果が周囲に思わぬ影響を与えている
□他のメンバーが何の仕事をどのように実践しているか／どんな状態か
□自分の知らないところでいろいろなことが発生している／それに自分も加担している場合がある

Microsoft社の経営方針
ダイバーシティ＆インクルージョンとグロースマインド

ダイバーシティ＆インクルージョン
多様性と受容

人がみな「違う」ことを大事に

□「自分だけよければいい」ではなく、その人がその人らしい能力を存分に発揮できる場を、みんなで助け合ってつくっていこうというマインド

□「相手のことを理解して認める」エンパシー力（自分とは異なる相手の意見を知的に理解する力）で物事を進める

グロースマインド
人は努力次第で成長する

人は変わり、成長するものだ

□いまはできなくても、勉強して変わろうという人を応援するとともに、他人を応援しサポートする人を評価する

□みんなで助け合いながらお互いに成長できる働きやすい場に

「他人を応援する人が評価された結果…」44歳で転職したエンジニアが実感したマイクロソフトが“女性に働きやすい”理由 より

チームを成功へと導く5つの鍵

Google re:Work掲載 GoogleとAP通信社との共同研究成果より

信頼性 (Dependability)	構造と明瞭さ (Structure & clarity)	仕事の意味 (Meaning of work)	インパクト (Impact of work)
任せると着実にプロセスを進めるメンバーがいる	仕事のゴールと役割、プランを明らかにする	仕事がどのような個人的な意味を持つのかを知る	仕事がどのような影響を社会に与えるのかを知る

心理的安全性(Psychological safety)

4つの不安＝無知だ・無能だ・邪魔してる・ネガティブだ

【個人ができる取り組み】
(1)仕事を実行の機会ではなく学習の機会と捉える。
(2)自分が間違うということを認める。
(3)好奇心を形にし、積極的に質問する。
(以上はエドモント氏の提言)

【管理者ができるアプローチ】
1.発言機会を平等に与える
2.競争よりも協力を
3.ポジティブ思考を意識する
4.上司が部下を尊重する
5.付加価値の高い1on1を実施する
6.チームで新入社員を支援する
7.評価方法を見直す
8.風通しの良い組織を作る

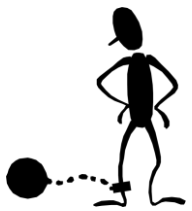
政治でもスポーツでも『熱狂』は危険なものだ。**その人が信じることだけを正しいとする盲目**につながり、あらゆる疑問を覆い隠す。

それは賢さとは対極にあるものだ。

ウルグアイ　ホセ・ムヒカ元大統領

専門家バイアス

出典：https://this.kiji.is/916396302473134 11　琉球新報



専門家バイアスから脱却するには？
＝自分を成長させるには？

☑まずは自分は専門家バイアスにかかっていると自覚する。

☑異なる見方、意見から学び、自らの既存知識と併せて新しいモノゴトの捉え方、考え方に更新し、より適切な思考・行動へと統合する。

ふりかえりが有効

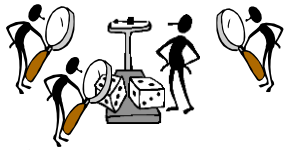
私が感じている“ふりかえり”のよい効能

ふりかえりカンファレンス2021～ふりかえりの傾向と対策「ふりかえりのふりかえり」から作法を学ぶ　より

☑相互理解

自分が、それぞれのメンバーがどのような人なのか、どんな状況なのかを共有できる。

☑チーム学習



他者の気づきを自分の気づきにできる。自分の実務(経験)だけではわからないことを他者から学べる。チームとして学ぶことができる。また、対話やファシリテーションを通じてモノゴトの適切な捉え方や伝達方法を浸透させることができる。

☑合意形成と実践促進

チームとしての合意形成が容易になる。チームで決めたことの実践が促進される。

☑価値観共有・共創～そして浸透

チームの価値観を共有したり、新しい価値観やより適切な価値観を一緒に作り上げる、それを浸透させることができる。

自分自身への気づきを高める法

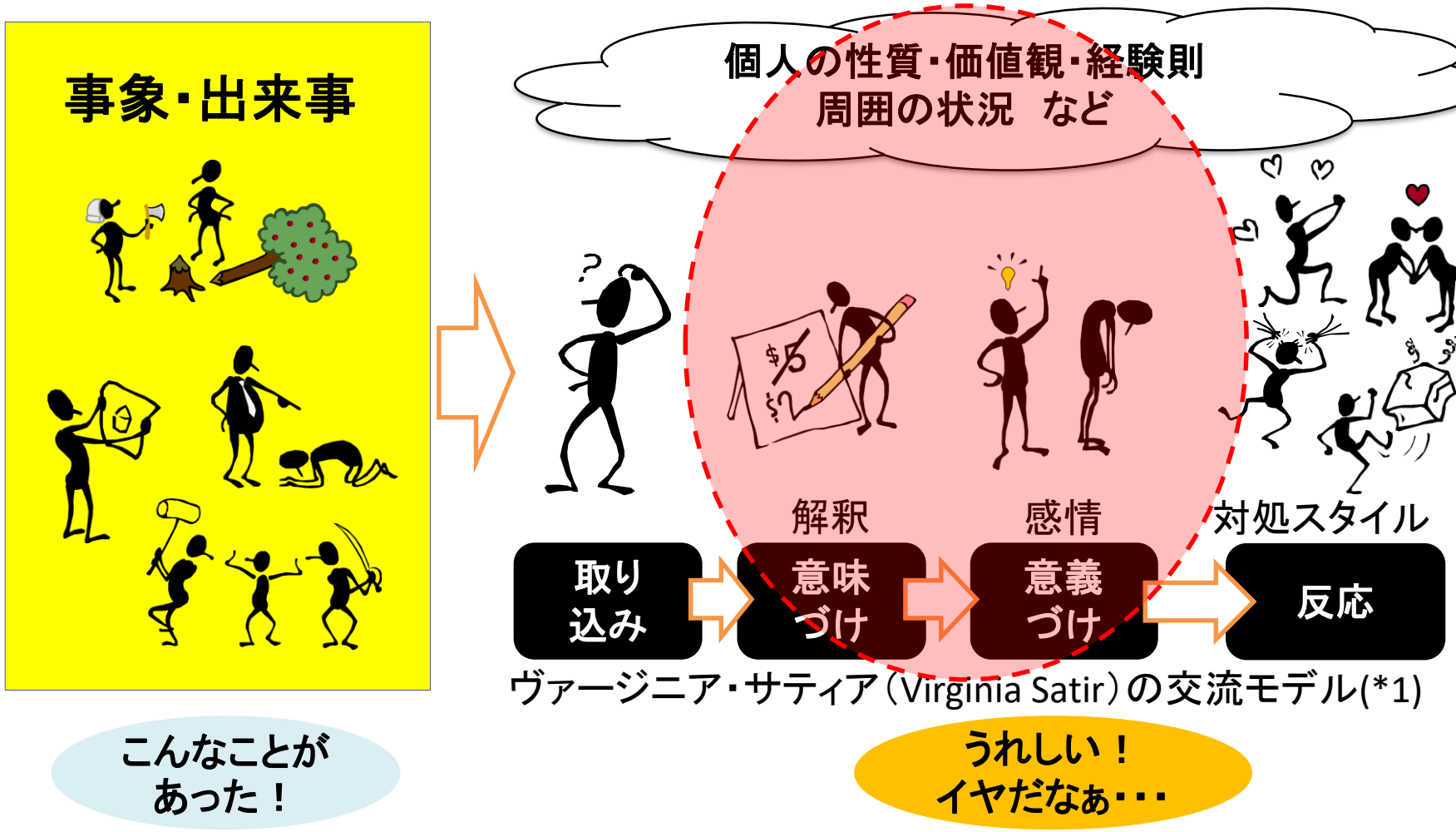
「スーパーエンジニアへの道　技術リーダシップの人間学」第七章

- 自己変革に成功するのに十分なだけの動機づけを持っているかを調べるテスト～たったいまから三箇月間、個人的な日誌をつけるために毎日五分使うこと
- 日誌には何を書くべきか？

「自分について書く」
- **事実**：自分に何が起こったか？（客観的に）
- **感情**：自分はそれにどう**反応**したか？
- **発見**：それにより何を学んだか？

- 日誌は**自分自身について学ぶため**につける。たいていの場合、**学ぶのはずっとあとになってその項目を読み直したとき**です。

人間のモノゴトの取り込み～反応まで



ヴァージニア・サティア（Virginia Satir）の交流モデル(*1)

*1:参考　ソフトウェア文化を創る2 「ワインバーグのシステム洞察法」 共立出版　G.M.Weinberg

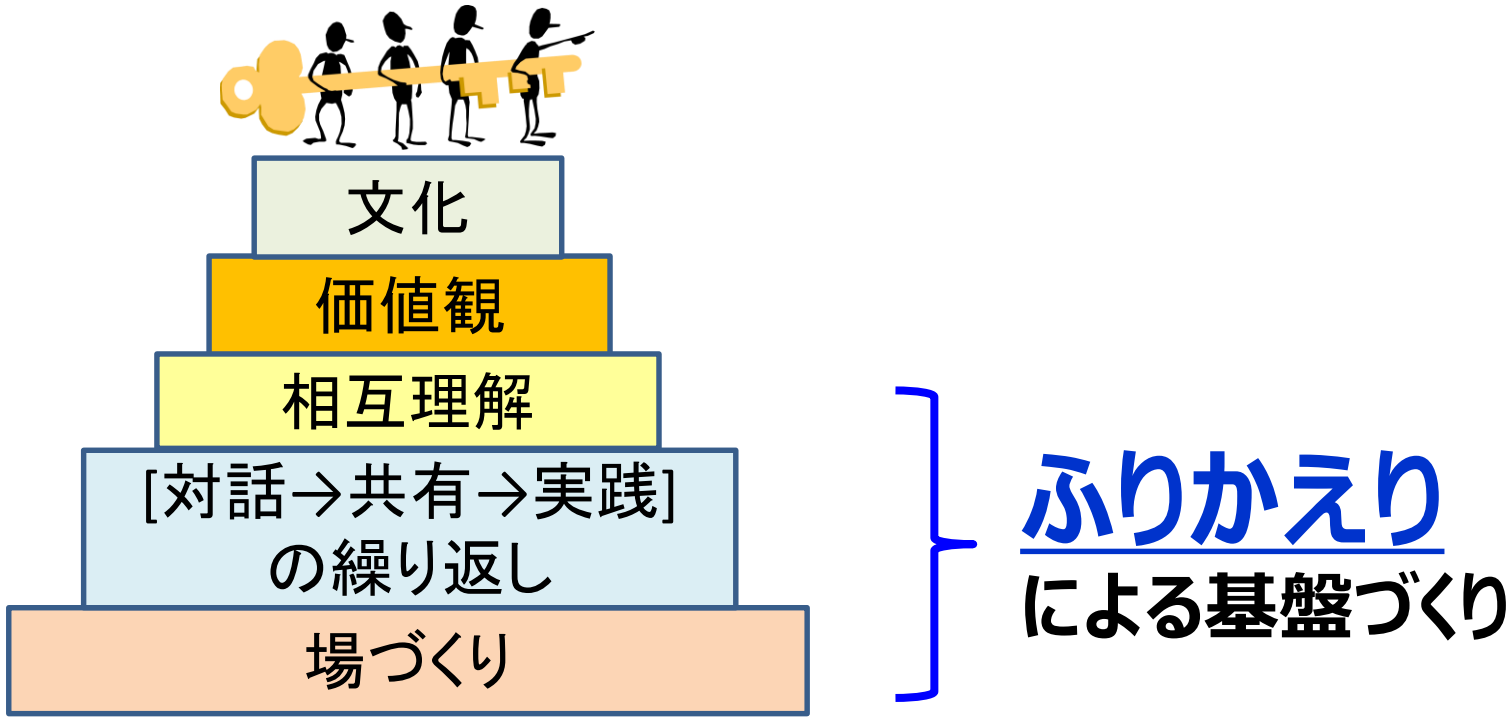
大切なのはできごとではない。できごとに対するわれわれの反応なのだ。

Gerald.M.Weinberg

技術リーダになるのは、彼らが失敗に反応するそのしかたによる。逆境を克服するだけでなくプラスに変える。
（技術リーダとは）敗北を成功の跳躍台に使う能力を持った人々である。

相応しい価値観→文化は大事なチームの礎

- ふりかえりを**適切に、繰り返し実践**することで、相応しい価値観に基づく運営が浸透すればそれが“**チームの文化**”になる。
- **チームの文化がチームのパフォーマンスを左右する。**



カーリングチーム：LocoSolare（ロコソラーレ）

- 本当に強いチームとは？

https://locosolare.jp/

「ピンチが来ても復活できるチーム」

ロコソラーレ代表理事　本橋 麻里さん

https://www3.nhk.or.jp/news/html/20220108/k10013416951000.html

「勝ち続けるチームではなく、負けてから這い上がれるチーム」

ロコソラーレ　サード　吉田 知那美さん

https://real-sports.jp/page/articles/62965027053974405 9

本音のコミュニケーション

「カーリング 本橋麻里 追い求めた本音のコミュニケーション」他　をベースに編集

- カッコつけず素直に言葉にする＝そのままの自分で
  - メンバーの素養を**そのまま受けとめられるメンバー、チームは強い**
  - 腹を割って話せるチームメンバーがいる（信頼感／お互いを尊重する）
  - 意見が言える・本音でぶつかり合える～エネルギーをもらえる、人生が豊かになる
- メンバーに興味を持つ＝**仲間を知る**～そのための“**傾聴**”と“**対話**”
  - どのようなコトやモノが好きか？等、メンバーの心が動く瞬間を知る
  - 相手の気持ちを引き出す質問等会話のキャッチボール（対話）で掘り下げる
  - 一見くだらない、他愛のない会話に大事なヒントがある
  - **互いに弱さを見せあえる→ピンチの時こそ支え合う**

心理的安全性の前提”受容”

本当の自律＝相互依存

カーリングの理念・文化

- 相手を見下さない。
- プレーを妨害しない。相手の負けを喜ばない。
- 審判はいない／卑怯なことはしない。
- 不当に勝つなら負けを認める。
- ギブアップではなくコンシード（相手に“譲る”）。
- 勝ったチームがシートの掃除を行う。
- 勝負が終わったら対戦相手と一緒にお互いの戦いをふりかえる。

まとめ

いかがでしたでしょうか？

今回は、

☑SaPIDとは？
☑ODC分析×SaPIDとは？
☑ODC分析を活用したパフォーマンス改善【ODC→SaPID連携】
☑ODC分析が実践できるようにする【SaPID→ODC連携】
☑チームパフォーマンス向上のポイント

についてお伝えしました。

SaPIDで
チームパフォーマンス沼に飛び込もう！

ODC分析を適用・実践する際の参考になれば幸いです

お疲れさまでした！

参考文献

- 「ソフトウェアプロセス改善手法SaPID入門」 安達賢二
- 「ワインバーグのシステム洞察法」 G.M.ワインバーグ
- 「学習する組織――システム思考で未来を創造する」 ピーターMセンゲ
- 「プロジェクトマネジメント知識体系ガイド（PMBOKガイド）第7版＋プロジェクトマネジメント標準」 PMI日本支部
- -the d.school bootcamp bootleg
  https://dschool.stanford.edu/resources
- Design Thinking for Educators Toolkit,IDEO,2011
  https://page.ideo.com/design-thinking-edu-toolkit
- ODC分析研究会オープンセミナー「ODC分析の概説」
- 「[入門+実践]要求を仕様化する技術・表現する技術 -仕様が書けていますか?」 清水吉男
- 成功循環モデル　マサチューセッツ工科大学ダニエル・キム
- SPI Japan2015「自律型プロジェクトチームへの変革アプローチ事例」　安達賢二
  http://www.jaspic.org/event/2015/SPIJapan/session3C/3C-3_ID012.pdf

参考文献

- テストの未来　テストエンジニアの「これまで」と「これから」を考える　より
  https://www.veriserve.co.jp/asset/approach/column/test-technique/post-1.html
- JaSST Review2023「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ー」　　安達賢二
  https://www.jasst.jp/symposium/jasstreview23/pdf/S1.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
- 「モチベーション3.0」 ダニエル・ピンク
- 「他人を応援する人が評価された結果…」44歳で転職したエンジニアが実感したマイクロソフトが“女性に働きやすい”理由 https://crea.bunshun.jp/articles/-/46227?page=2&utm_source=crea&utm_medium=direct&utm_campaign=nextpage_link
- グーグルが発見！成功するチームに共通する「心理的安全性」のつくり方
  https://gendai.media/articles/-/66539?page=1&imp=0

参考文献

- ＜社説＞ムヒカ氏初来日　戦争なき世界実現考えたい　琉球日報
  https://this.kiji.is/916396302473134 11
- ふりかえりカンファレンス2021「ふりかえりの傾向と対策「ふりかえりのふりかえり」から作法を学ぶ」 安達賢二
  https://speakerdeck.com/kitanosirokuma/hurikaerifalseqing-xiang-todui-ce-hurikaerifalsehurikaeri-karazuo-fa-woxue-bu
- 「スーパーエンジニアへの道　技術リーダシップの人間学」 Gerald.M.Weinberg
- カーリング 本橋麻里 追い求めた本音のコミュニケーション
- ロコ・ソラーレ、銀メダル最大の要因は「勝つか学ぶか」。負けるたび強くなる“成長の本質”　[カーリング]
  https://real-sports.jp/page/articles/62965027053974405 9/