システム開発保守QCD研究プロジェクト

2016年4月21日

一般社団法人　日本情報システム・ユーザー協会

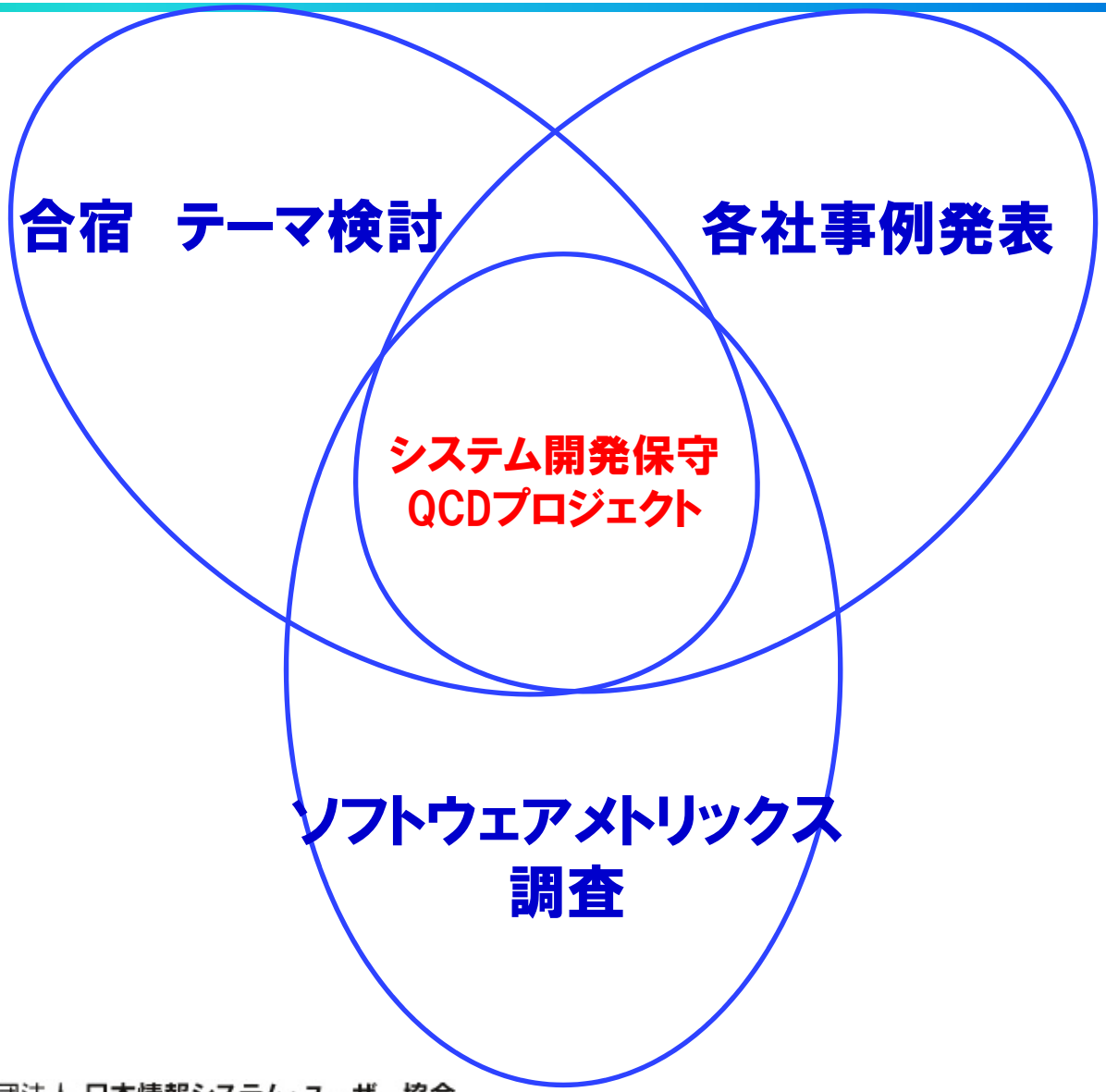
システム開発・保守QCD研究プロジェクト

JUAS 一般社団法人 日本情報システム・ユーザー協会
Japan Users Association of Information Systems

目次

1. プロジェクトの取り組み
2. 2015年度のテーマと進め方
3. 部会参加企業・参加者（非公開）
4. 各社事例発表　2015年テーマ
5. 合宿の取り組み（2015年テーマ）

1．プロジェクトの取り組み



2. 2015年度のテーマと進め方

（1）2015年度のテーマ

①システム開発・保守のQCD向上
②システム再構築（パッケージ活用等）
③保守の生産性向上（保守の評価等）

（2）部会活動の進め方

①毎月第1火曜日　15時～18時（3時間）。4～5テーマの紹介。

②部会メンバー各社からの事例発表
　年に1回、「システム開発・保守のQCD向上」に関する自社の事例を紹介
　各社の最近の取り組み事例が紹介され、大変参考になるとともに
　部会メンバー各社のレベルアップに繋がる。

③合宿テーマ
　2015年度は
　「レビューの品質指標」
　「システム再構築」
　「要求・要件定義の迅速化、効率化」

2015年度の取り組み

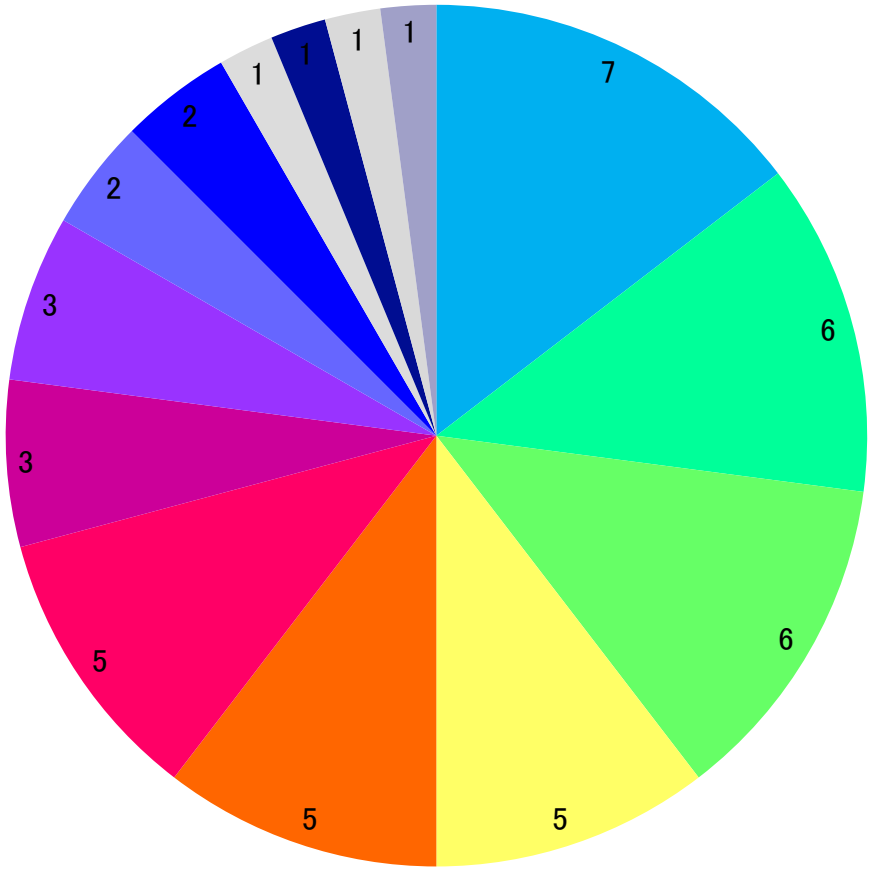
4. 各社事例発表　2015年テーマ

システム開発・保守のQCD向上

システム再構築（パッケージ活用等）

保守の生産性向上（保守の評価等）

4. 各社事例発表　48事例



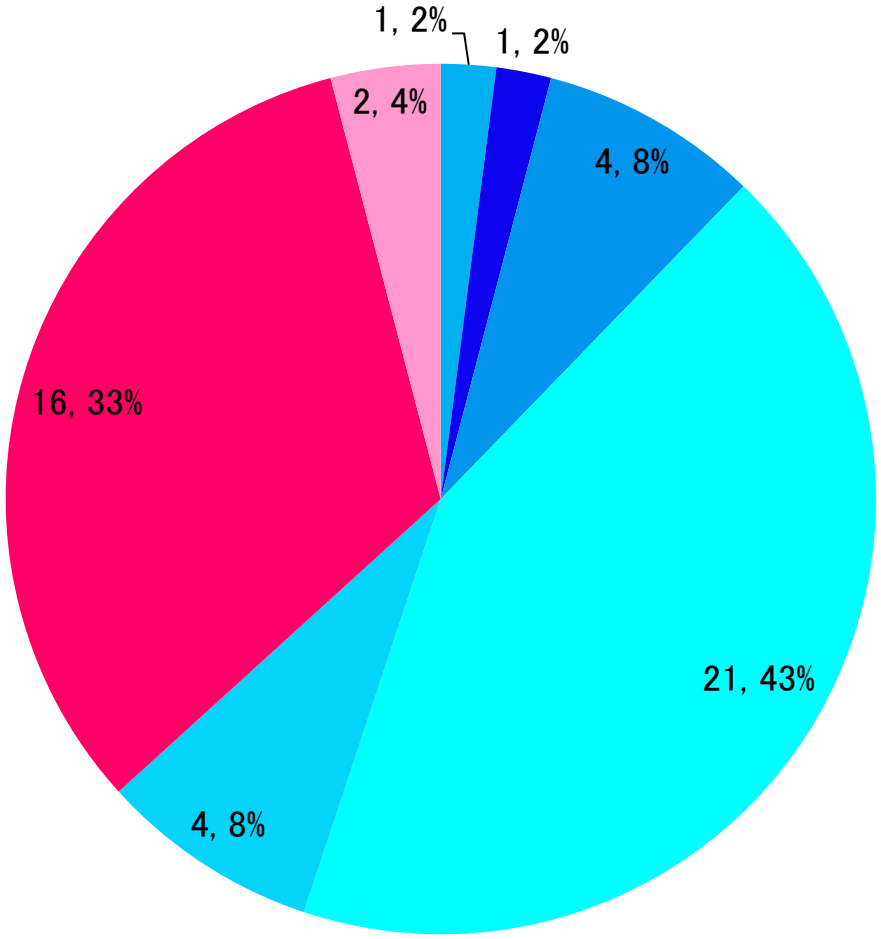
- 品質向上
- SWM
- 要求・要件定義の迅速化
- オフショア開発
- システム再構築
- 人材育成
- 自動化
- 保守の生産性向上
- プロジェクトマネージメント
- システム再構築（パッケージ活用等）
- システム開発・保守のQCD向上
- 開発生産性向上
- アジャイル
- レビューの品質指標

システム再構築の定義

詳細	リホスト	リライト		リビルド	
		ロジックを流用し 新しい言語で作り直す方法		ビジネスプロセスから見直し システムを構築する方法	
		単純 マイグレーション	システムリフォーム	スクラッチ	パッケージ＋Addon
	ソースにできるだけ手を 加えず 移行する手法	変換ツールによる 単純コンバージョン	リバースエンジニアリング 人が対応するので 対応言語が多く制約が 少ない	現行調査＋要件定義	現行調査＋Fit&Gap
OS	置き換える	置き換える	置き換える	置き換える	置き換える
DB	置き換える	置き換える	置き換える	置き換える	置き換える
言語	無し	ソース to ソース 変換ツールで書き換え	ソース to ソース 変換ツール ＋ 人手で書き換え	新仕様 to ソース	パッケージ＋新仕様 to ソース
仕様	変更なし	変更なし	変更なし	変更あり（新仕様）	パッケージ ＋ 新仕様

★DB　同じタイプのDBへの置換・拡張と異なるタイプへのDB置換の方法がある。

アンケート結果(49件)

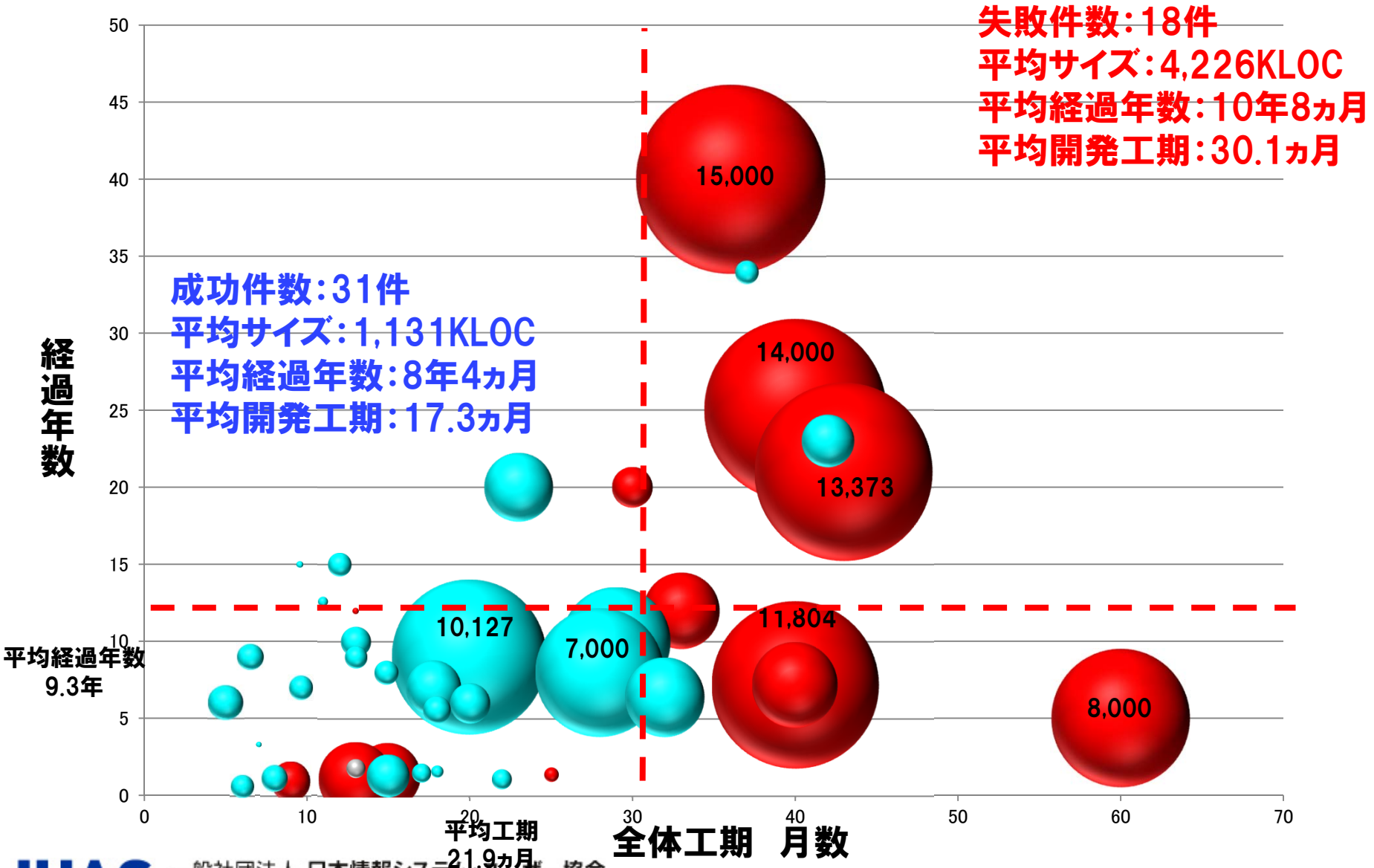


- リライト単純マイグレーション成功
- リライトリフォーム成功
- リホスト成功
- リビルドスクラッチ成功
- リビルドパッケージ成功
- リビルドスクラッチ失敗
- リビルドパッケージ失敗

成功		失敗	
リビルド	25	リビルド	18
リホスト	4		0
リライト	2		0
	31		18

アンケート結果（49件）

バブルサイズ：KLOC



アンケート結果プロフィール

全体工期　＞　30か月　失敗比率　8割以上

経過年数　＞　10年　　失敗比率　6割以上

KLOC　＞　3000　失敗比率　5割5分以上

KLOC　＜　1000　成功比率　7割以上

リライト・リホスト　　6件　全て成功

まとめ

Step1

旧システム分析調査　再開発の前に単独実施

Step2

開発方式の決定（工期　30か月未満　3000KLOC以下にまとめる）

目的合意：老朽化や保守切れ対応か、新規ビジネス対応なのか明確にする
旧システム分析：経過年数、規模、品質、複雑さ、システム錬度を可視化する
システム分割の有無：システムをどの程度分割できるか
プロジェクト分割の有無：ビックバンでなければならないか、何分割できるか
工期、コストの制約：リホストからリビルドではダメか
新規要件がどの程度あるか：リライト、システムリフォームではダメか

Step3

体制構築・要件定義

会社の意思決定ができる体制構築（リビルド）
業務メンバー・システムメンバーのシステム錬度確保
「現行通り」の要件可視化・旧システム仕様と新規要件の整合性確認

合宿でのテーマ検討

5. 合宿の取り組み(2015年テーマ)

レビューの品質指標

**パッケージを適用した
システム再構築プロジェクトを成功に導くために**

**要求・要件定義の迅速化、効率化
GQM＋Strategies**

合宿　レビュー品質向上

事前アンケート
レビューの種類
レビュー指標
レビューの工夫（方法・効果）
終了判断基準
問題点と対策
レビュー観点表

合宿　レビュー品質向上

＜人（レビュー態勢）＞

・ユーザーのレビュー参画義務化
　→　ユーザー山積み要員表

・多面的なレビューができるよう専門家をアサイン
　→　業務、アプリケーション、インフラ‥

＜もの（基準、ツールなど）＞

・要件の“つまり度”の判断
（レビューでどう判断すればよい？）
　→　レビュー指摘密度の評価＋指摘内容の深堀で判断。
　→　指摘箇所の分類（経営／事業／業務）により要求と要件定義のずれを可視化。
　→　検討状況の見える化（マインドマップ等）

合宿　レビュー品質向上

■レビュー内容

《設計》

- 設計担当者セルフチェック
- 委託先内部レビュー
- 社内レビュー(ピアレビュー)
- ユーザレビュー
- 工程完了レビュー

《指標》

レビュー欠陥抽出率（抽出欠陥数÷設計量　ページ　KS）

レビュー密度　　（レビュー時間÷設計量　ページ　KS ）

レビュー回数　　（同一設計書のレビュー回数　観点・メンバを変える）

見逃し率

（見逃し率　＝　e　÷（e　＋　E）

e:後工程で明らかになったエラー数、E:自工程で明らかになったエラー数）

レビュー指摘効率　（指摘件数÷レビュー時間）→改造の場合

合宿　レビュー品質向上

1　設計の作りこみ
　要件定義フェーズ：Wモデル利用（ベンダ要件確認まで）

2　成果物の品質
　見逃し率
　残存しているエラーがないか分析
　（混入形態や混入原因、摘出状況、影響度、担当者別、難易度別、
　　目標値との差異、根本分析に応じた対策）
3　プロセス
　開発計画書にレビュー計画を記載する
　（レビュー目的、日程、どういう種類レビューするのか、品質目標（システム　機能）
　　レビューの終了条件、有識者がアサインできなかった場合の対応策（時間を増やす、
　　事前にソースを解読））
　レビュー体系を定義、基準を設けワークシート（レビュー要領書）を用意
　レビューを1時間程度で行える単位に小分け
　代表機能（重要度・複雑性）を選定して対面レビューを実施、
　類似機能は対面レビュー時の指摘を横展開した上で査読レビューを実施

4　教育
　設計書作成開始段階で関係者をできる限り集めてレビュー（設計書のイメージを共有）
　レビュー教育研修（レビュー種類・目的、用語、心構え、レビューのやり方、ガイドライン、手順、実施記録）

合宿　パッケージを適用したシステム再構築

●過去に実施したパッケージリビルド適用の経験、知見の
　乏しさを補うことが必要
　（そもそもPKGリビルド案件を複数経験したユーザー企業の
　　IT部門担当者は多くないはず）
　-PKG選定段階での試用
　　→準備に限界あり（本番データを使えればよいのだが）
　　→費用はどうする？

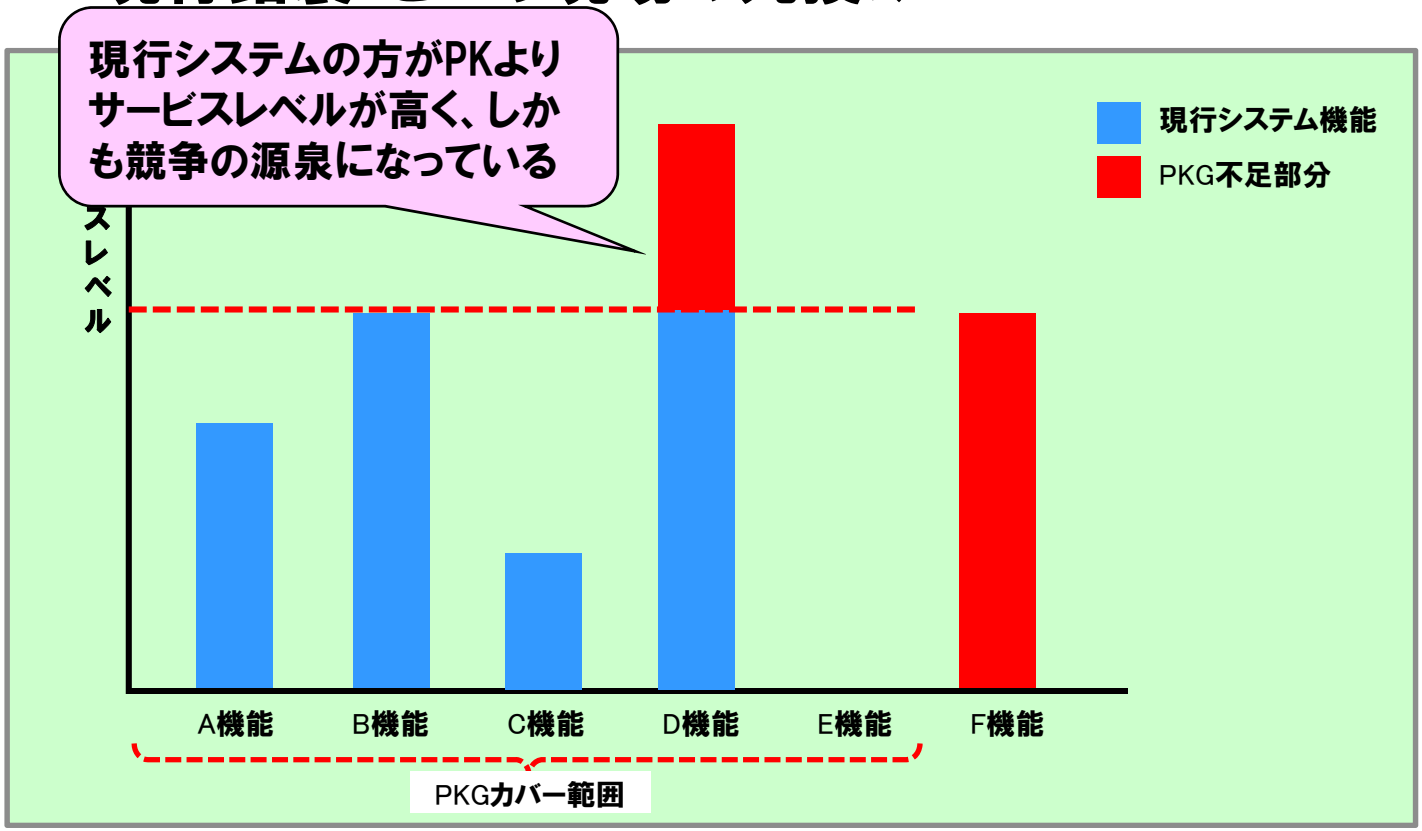
**PKG適用の場合、昨年の「システム再構築」の対策を実施
したただけでは解決できない課題がある！**

合宿　パッケージを適用したシステム再構築

●それって「Fit & Gap」分析不足では？

→「Fit & Gap」はちゃんとやってる！…けどなぜ失敗する？

→“現行踏襲”という現場の丸投げ



合宿　パッケージを適用したシステム再構築

●安易にPKGの適用をしてはいけないケース

－赤い部分（PKG機能範囲外）が競争の源泉

－お客様へのサービスレベルに直結する部分

→機能レベルのF&Gはできるが、サービスレベル（深さ）

まではわからない

●PKGが向く、向かないの判断

●カネと時間をかければ解決する問題　バランス

●ライフサイクルを通してかかるトータルコスト（運用、保守）

合宿　要求・要件定義の迅速化、効率化 GQM＋Strategies

良い点

- 目的（達成すべきこと）と方針（どんな切り口なのか）を構造化できる
- 利用部門のやりたいことを可視化できる、施策実施の前提が見える化される
- （きれいに書ければ）わかりやすい

→　文章より図はわかりやすい。

- 事実・仮説の確認で施策の妥当性検証に役立った
- 変な施策を見つけられる
- 事実なのか仮定なのか確認する中で必要な証跡（足りない情報）が共有できた
- 要求・要件の脱線を抑制できそう

→　妥当性が確認しやすい

- GOALの設定フォロー点が階層化、細分化されている
- 責任範囲が明確
- 目的の中身の作成がラク

→　書きやすい

合宿　要求・要件定義の迅速化、効率化 GQM＋Strategies

悪い点（難しかった点）

- 構造化するときに上位の施策を失念してしまうことがある。
- 構造が縦割りで、他の部門との重複が発生
- どうしても手段（下位のStrategies）から考えてしまう。作るのが難しい。
- 下位の目標が上位の施策以外の要因によるものが多いのではないか。

→　経営目標と整合のある施策を考えるのは難しい（利用部門もシステム部門も）

- きれいに構造化するのは難しいんじゃないか。
- 施策を検討する際に、上位の別の施策とどっちに紐づくのか分かりにくい。

→　経験やセンスがいる

- 情報システム部門がどうやって書くのか。（自分で書くのか、書かせるのか）
- 利用部門がついてきてくれるかな？

→　書ける人が必要

- 正しい結論を導けたのか？の検証は？

→　識者が必要。　　×属人性　　×一般性

合宿　要求・要件定義の迅速化、効率化 GQM＋Strategies

今後の活用について

●事業部門の要求を受けてから、IT部門がGQM+Sグリッドの作成を始めると、数ヵ月の期間を要すると思われる。
（つまり、検討～開発～C/Oに時間がかかる）

●要件定義の開始前に、事業部門側でGQM+Sグリッドの作成が終わっているとよい。

●事業部門に、GQM+Sの良さを理解してもらう必要がある