# SPI Japan 2014　発表概要集

・セッション順番＞発表順番に、掲載されています。
・目次から、各発表の概要にジャンプすることができます。

## 目次

1A1「Ruby を使った開発プロジェクトでの取り組み」細美彰宏(日立ソリューションズ)

〈タイトル〉: Ruby を使った開発プロジェクトでの取り組み

〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 細美彰宏（さいみあきひろ）
所属: 日立ソリューションズ
〈共同執筆者〉
氏名（ふりがな）:
所属:

〈要旨〉
Ruby on Rails の登場でプログラミング言語 Ruby が注目されて 10 年経ち、ようやく業務システムの構築に Ruby を適用した事例が増えてきた。当社では Ruby 専門組織を設置して、Ruby の活用推進に取り組んでいる。中小規模の業務システムや周辺システムの開発を中心に Ruby 適用を開始し、現在、大規模システムやサービスにも適用範囲を拡げている。更なる適用拡大に備えるために、大規模開発での実践内容やノウハウを開発手順にまとめ、比較的新しい技術やツールを導入した開発環境を整備した。本発表では、当社における Ruby 推進と適用事例をベースにシステム構築の改善活動を紹介する。

〈キーワード〉
Ruby on Rails、開発手順、開発環境、アジャイル、タスク管理、テスト自動化

〈想定する聴衆〉
ソフトウェア開発者

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可: 具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

当社では Ruby 専門組織を設置し、Ruby の活用推進に取り組んでいる。当初は中小規模の業務システムや周辺システムの開発を中心に Ruby 適用を推進してきた。当社案件も含めて、約 50 プロジェクトもの Ruby の適用実績ができ、業務システムの構築にも Ruby を使った事例が増えてきた。開発の実践やノウハウが蓄積できたことで、大規模システム開発やアジャイル型開発への適用にも挑戦した。

(2)改善前の状態

Java や C 言語のように確立された開発方法がなく、プロジェクトが Ruby 活用を検討する際、どのような開発方法、環境を使えば良いか、整備が進んでいなかった。また、Ruby はアジャイル開発と親和性が高いと言われており、サービスをアジャイル型開発に進めるにあたって、課題の確認や有効性検証が求められていた。システム開発やサービス開発に Ruby を選択した際に、円滑にプロジェクトを立ち上げ、効率的に開発が進められることを期待する。

(3)改善前の状態をもたらした原因（因果関係）

当初は中小規模を対象に、少人数チームで Ruby 開発を行っており、Ruby 専門組織の技術者の経験やノウハウに依存した開発を行っていた。数少ない Ruby 技術者がプロジェクトに入り、開発プロセスや開発環境を整備して、サポートしながら開発を行っていた。

(4)計画した変更内容

比較的新しい技術やツールの導入を方針として、開発手法と開発環境の整備を行った。また、アジャイル型開発での利用も意識し、アジャイル型開発のプラクティスを取り入れた技術整備とした。

(5)変更の実現方法

大規模開発での実践内容やノウハウを開発手順にまとめ、比較的新しい技術やツールを導入した開発環境を整備した。開発手順と開発環境の適用を前提として、Ruby 専門組織がプロジェクトの立ち上げと開発準備を行う。開発が完了した時点で、適用結果や改善要望をフィードバックとして受け、手順や環境を改善するサイクルとした。

1. 開発手順

Ruby プロジェクトの立ち上げから設計、プログラミング、テストまでの開発手順や、Ruby/Ruby on Rails コーディング規約、設計留意事項、品質管理、性能対策等をガイドにまとめた。Ruby にはテストを行うための標準的な環境があり、テストフレームワーク RSpec を用いたテスト自動化および継続的インテグレーションの適用を推奨している。

2. 開発環境

Ruby/Ruby on Rails のためのプログラミングからテストを行うための環境を整備した。タスク管理(Trac/Redmine)、ソースコード管理(Git)、継続的インテグレーション(Jenkins)を行うための環境を準備した。また、日本の企業システムでよく使われる機能を持つ部品群を提供している。

(6)変更後の状態や改善効果

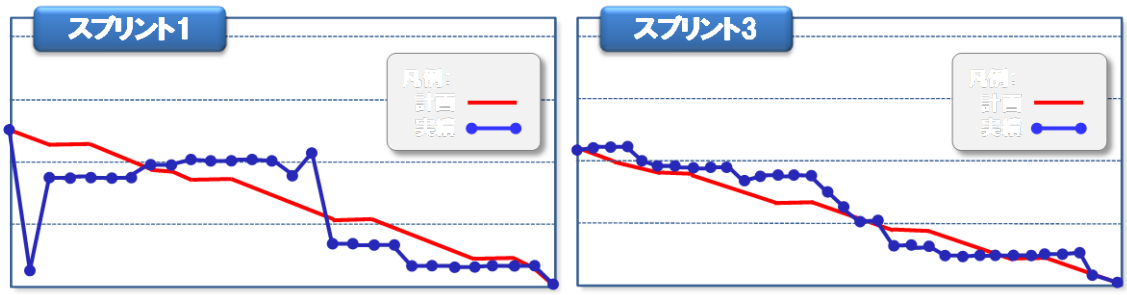
大規模システム開発とアジャイル型開発へ適用した事例を紹介する。

1. 大規模システム開発

300 画面の大規模 Web システムを Ruby で構築した。Java/C 言語で構築されていた現行システムは度重なる改造と機能追加で複雑化していた。制度改正への対応の迅速化を目的に、高い開発効率と開発規模の削減による保守性向上を狙い、Ruby を採用した。Java/C 言語では約 500K ステップを 1/5 の約 100K ステップに削減でき、保守性の向上が期待できる。品質安定化のために、テスト自動化を導入して、デグレードの早期検出と対策が可能になった。更に、変更の影響範囲が特定しやすくなり、リファクタリング実施が容易になった。

2. アジャイル開発

教育分野の Web システムを Ruby で構築するにあたり、アジャイル型開発を導入して、課題確認や有効性検証を行った。16 ストーリー、106 タスクに対して、1 ヶ月のスプリントを 3 回実施した。スモールスタートで早期に本番運用を開始して、ユーザーニーズを吸い上げながら、徐々に機能拡張を行った。スプリント毎の進捗をバーンダウンチャートで管理した。下図のように、スプリント 1 ではタスク状態の逐次更新が不徹底であったことと、ツールがうまく使いこなせなかったことで、正確な進捗管理ができなかった。スプリント 3 ではプロジェクトの成熟度が上がり、開発作業が円滑に進行した。Ruby を使ったことで、変更や追加要望に対して柔軟な対応が期待できる点で、Ruby とアジャイルの親和性が高いことを確認した。




(7)改善活動の妥当性確認

Ruby の開発手順と開発環境を整備して、大規模システム開発やアジャイル型開発へ適用した。改造や追加機能の開発スピードや保守性の向上につながると評価している。Ruby とアジャイル型開発の両方を理解している技術者はまだ少なく、大規模システム開発で、Ruby とアジャイル型開発の両方をどう適用するかが課題である。

以上

1A2「レビューの有効性測定と改善へのトライアル」伊藤浩子(キヤノンソフトウェア)

〈タイトル〉:　　レビューの有効性測定と改善へのトライアル
〈サブタイトル〉:

〈発表者〉
　　氏名（ふりがな）:伊藤　浩子（いとう　ひろこ）
　　所属:　　キヤノンソフトウェア株式会社　企画本部　プロジェクト推進部
〈共同執筆者〉
　　氏名（ふりがな）:
　　所属:

〈要旨〉今回、改善対象となった開発現場では、顧客からの厳しいコスト・納期の要求があり、品質を確保するためのレビュー時間、コストが十分ではない、という問題が常態化していた。そのため、上流工程の品質に起因する手戻りが多発していた。そこで「フェーズ欠陥阻止比率」というメトリクスを用いて、レビューの有効性を測定、分析した。その結果、レビューの実態を定量的に把握できるようになり、測定のプロセスも根付いた。現在は、その結果をもとにレビューの有効性を高めるための改善を行なっている。

〈キーワード〉
　　レビュー、レビュー有効性、上流工程、測定と分析、プロセス改善、メトリクス

〈想定する聴衆〉
　　プロセス改善初心者、品質保証部門の方、SEPG、ソフトウェアエンジニア（リーダー以上）

〈適用状況〉
　　□多用されている段階、■適用できる段階あるいは初めて適用する段階
　　□適用するにはさらに検討を必要とする、□着想の段階
　　□その他（　　）

〈適用可能性に関する制限〉
　　■汎用性がある、
　　□類似プロジェクトにも適用可：具体的な類似点（　　）
　　□自プロジェクトのみ

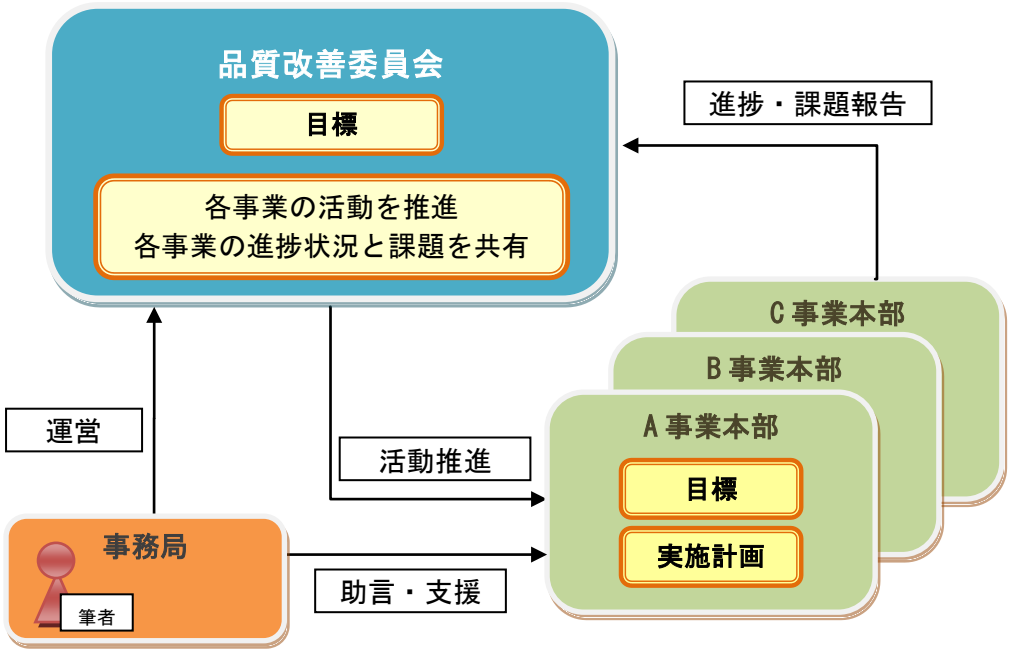
〈発表内容〉

(1) 背景

＜品質改善委員会＞

全社的な品質改善を推進していく組織として「品質改善委員会」を設置した。委員長は社長が務め､委員は､本部長/事業部長クラス､品質スペシャリストなどで構成されている。活動の主体は各事業本部であり、それぞれに目標と実施計画を設定する。委員会は、各事業本部の進捗状況や課題を共有し、活動を推進するために、四半期に１回のペースで開かれる。

今回、発表する事例は、ある部門が行なった、上流工程における品質向上の取組みである。この取組みにおける私の役割は、品質改善委員会事務局の一員として、取組みに対する助言、データ分析の支援をすることであった。




＜改善対象部門のプロフィール＞

今回、改善対象となったのは、中堅企業向けに、主に基幹系システムを受託開発している部門である。顧客からは厳しいコスト・納期要求があり、品質を確保するためのレビュー時間、コストを十分に確保できないという問題が常態化していた。

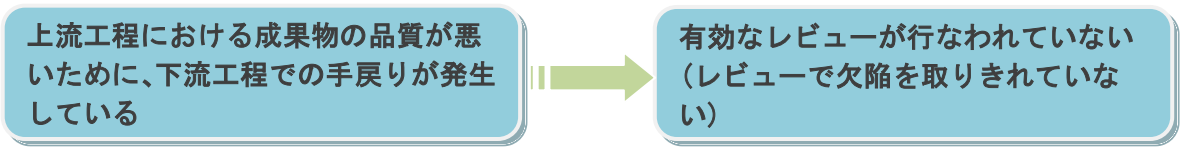
そのため、上流工程での品質に起因する手戻りが多発しており、プロジェクトの利益を圧迫するという状況が度々発生していた。

(2) 改善前の状態

この部門では、上流工程での成果物の品質低下が下流工程に影響を及ぼし、手戻りを多く発生させていた。レビューは行なわれていたが、重大な欠陥除去につながる有効なレビューが行なわれていない状態だった。

また、計画時点から工程が重なり合っている部分があり、移行判定は、定性的な判定基準で行なっていた。そして、現状を把握するためのデータが収集されていなかった。

まずは、測定により、レビューの実態を定量的に把握できるようにすること、そしてレビューの有効性を上げることが、この部門の急務であった。また、データを収集することで、今後、品質基準に基づいた移行判定が行なわれる状態に改善する必要があった。




(3) 改善前の状態をもたらした原因（因果関係）

原因分析は過去のプロジェクト完了報告書の調査やヒアリングで行なった。有効的なレビューが行なわれていない原因として、以下のことが考えられた。

① レビューの詳細なプロセスがない
② レビューのメトリクスを収集していない
③ レビューに関する品質基準がない
④ レビューチェックリストなどのツールを使っていない
⑤ レビューのノウハウが蓄積されておらず属人的である
⑥ レビュー時間が不足している

このうち、すぐに着手できることとして①～③を先行して行ない、④、⑤はその後で検討することとした。また、⑥については、この部門においては、顧客からの厳しいコスト・納期要求があり、多くの制約があることが分かった。そこで、限られた時間の中で、レビューの有効性を上げるための改善施策を検討した。

(4) 計画した変更内容

改善は、段階的に行なう必要があったため、下記の順番で実行した（一部実行中）。

今回の発表は④までの内容である。また、①、②は部門 QMS 担当者が行なった。

① 収集メトリクスの定義、集計シート作成、トレーニング
② メトリクス収集
③ 結果分析
④ 改善施策Ａ（メトリクス再検討、プロセス定義）
⑤ 結果分析
⑥ 改善施策Ｂ（レビュー有効性改善）
⑦ 結果分析（改善の妥当性確認）

(5)変更の実現方法

① 収集メトリクスの定義、集計シート作成、トレーニング

＜収集データと導出メトリクス＞

収集メトリクスとして以下を定義した。

- ページ数
- レビュー工数
- レビュー人数
- レビュー参加者の単価
- 指摘事項件数（指摘分類別）
- 指摘密度（件/ページ）
- 工数密度（人時間/ページ）
- フェーズ欠陥阻止比率※1 巻末参照のこと
- 軽微欠陥率{（誤字脱字件数/全指摘数）*100}

＜集計シート作成＞

- 集計シートはエクセルで作成
- レビューだけでなく、全工程を通しての障害件数もそれぞれ記入できるようにした
- 欠陥は、原因工程別に分類して記入するようにした
- レビューの指摘事項は（誤字脱字、記載漏れ、記載曖昧、記載間違い、その他）に分類し、導出メトリクスを算出する際は、誤字脱字を除いた指摘事項の件数で行なった

＜トレーニング＞

- プロセスは明確に定義しなかった（文書化しなかった）
- 収集データと集計シートの使用方法の説明を中心に行なった

② メトリクス収集

プロジェクトマネージャーがデータ集計シートに記入し、QMS 共有フォルダに格納する。

③　結果分析

各種メトリクスと、フェーズ欠陥阻止比率を用いて、レビューの有効性を測定し、各プロジェクトのレビューの現状を定量化した後、傾向を分析した。

図 1 は、工数密度と指摘密度の散布図である。縦軸にレビュー指摘密度、横軸に工数密度をとっている。
A および B プロジェクトは、改善対象部門のプロジェクトであり、C プロジェクトは別部門の商品開発系プロジェクト（CMMI：レベル 2 達成）である。

A および B プロジェクトが含まれる、左上の黄色いゾーンは、工数密度が低く、指摘密度が高い。これは、レビュー対象物の品質が悪い可能性と、短時間で効率的にレビューしている可能性が考えられる。改善対象である A および B プロジェクトは、C プロジェクトと比べて工数密度が低く、レビューに十分な時間を確保できていない、という事実がデータからも裏付けられた。

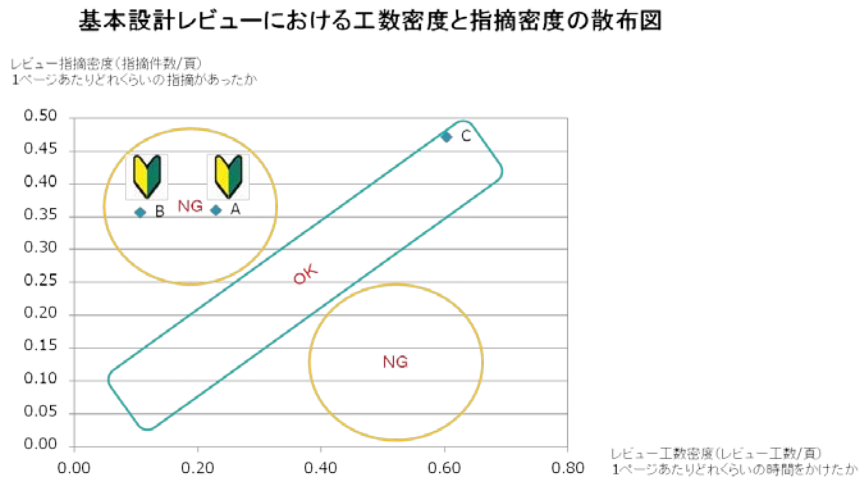



図 1. 基本設計レビューにおけるレビュー工数密度と指摘密度の散布図

A プロジェクトは、レビュー対象物の品質が悪い可能性と、短時間で効率的にレビューしている可能性が考えられることが分かった。
そこで、今度は、フェーズ欠陥阻止比率を用いて、レビューの有効性を測定した。フェーズ欠陥阻止比率の詳細については、巻末を参照していただきたい。

A プロジェクトのフェーズ欠陥阻止比率は 64.4%　※(67/104)＊100 であった。つまり、残りの 35.6%は基本設計レビューでは発見されずに後ろの工程に流れてしまっていることが分かる。一般的に、フェーズ欠陥阻止比率は 70～80%以上が妥当であるとされており、A プロジェクトでは、レビューで十分に欠陥を取りきれていない、ということ分かった。

また、基本設計レビューにおける指摘密度は 0.36(件／ページ)であった。これは参考部門品質基準(0.5～1.0)を下回った。この場合、レビュー対象物の品質がよかった、という可能性も考えられるが、フェーズ欠陥阻止比率と併せて分析すると、レビューで欠陥を取りきれていない、ということが推察できる。

| | | 作りこみ工程 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 発見工程 | | 要件定義 | 基本設計 | 詳細設計 | 実装 | 合計 | PDCE | 軽微欠陥率 |
| | 要件定義レビュー | | | | | | | |
| | 基本設計レビュー | 6 | 67 | | | 73 | 64.4% | 6.4% |
| | 詳細設計レビュー | | | | | | | |
| | コードレビュー | | | | | | | |
| | 単体テスト | 0 | 32 | 0 | 71 | 103 | 93.4% | |
| | 結合テスト | | 5 | 0 | 5 | 10 | | |
| | 合計 | 6 | 104 | 0 | 76 | 186 | | |

| | 指摘件数（件） | 規模（頁） | 工数（人時） | 指摘密度（件/頁） | 工数密度（人時/頁） | 指摘効率（件/人時） |
|---|---|---|---|---|---|---|
| 基本設計 | 73 | 203 | 47 | 0.36 ※0.5~1.0 | 0.23 | 1.55 |

図 2.　A プロジェクトにおけるフェーズ欠陥阻止比率と各種メトリクス

以上の結果を総合すると、下記のような分析結果が得られた。

- レビュー時間不足
  ＡおよびＢプロジェクトはレビュー工数密度が低く、レビュー時間を十分に確保できていない、という現状が数値からも裏付けられた。

- レビュー有効性が低い
  Ａプロジェクトでは指摘密度が部門参考基準より低い、0.36(件/ページ)であり、フェーズ欠陥阻止比率が 64.4%であることから、レビューによって欠陥を取りきれておらず、そのことが、下流工程における手戻りの原因になっていることが推察できた。

④　改善施策Ａ（メトリクス再検討、プロセス定義）
③の分析により、レビューを改善する上での課題が３つ見つかった。

- 出荷後（ユーザーテスト時）障害件数の収集
  フェーズ欠陥阻止比率を正確に算出するには、出荷後の障害件数が必要であったが、この部門では、顧客に納品した後のデータは収集していなかった。

- 指摘分類の細分化、あるいは指摘事項の分析
  指摘分類が「記載漏れ」「記載曖昧」「記載間違い」の三種類であるため、具体的にどのようなことを改善すればよいのかが数値データからは分析できなかった。

- レビュープロセスの定義が未着手
  レビュープロセスは定着したが、移行判定を含めたプロセスの定義と実施には至らなかった。

(6) 変更後の状態や改善効果

現時点で改善できたことと、改善途上であるものは以下の通りである（◎改善できた、△まだできていない）。

◎定量的なレビュープロセスの評価
◎レビュープロセス、測定のプロセスの定着
△レビューの有効性向上（フェーズ欠陥阻止比率向上）
◎品質基準策定
△品質基準による移行判定

(7) 改善活動の妥当性確認

第一段階の改善施策である、レビューの実態を定量的に測ること、品質基準を策定することは実現でき、改善対象部門で定着した。レビューの有効性を向上させるための改善は引き続き取組み中である。さらには、レビューの改善が、上流工程での成果物品質の向上につながり、最終的には上流工程への手戻りが減るかどうかを監視していく必要がある。

※1 フェーズ欠陥阻止比率とは

フェーズ欠陥阻止比率とは、各々の欠陥除去工程で除去すべき欠陥をどれだけ除去できたか、を表すメトリクスである。この値が高いほど、レビューの有効性が高いと言える。
一般的に、フェーズ欠陥阻止比率の目安は 70～80%と言われている。
フェーズ欠陥阻止比率は下記の計算式で算出される。

フェーズ欠陥阻止比率＝
（ある工程で発見された欠陥数／ある工程で作りこまれた欠陥数）＊100

1A3「開発工程の無駄を見える化」浦田有佳里(ＨＳ情報システムズ)

〈タイトル〉:　　開発工程の無駄を見える化

〈サブタイトル〉:　　指摘対応コストの見える化：コストモデルの定義

〈発表者〉
氏名（ふりがな）:　浦田有佳里（うらたゆかり）
所属：　株式会社ＨＳ情報システムズ
〈共同執筆者〉
氏名（ふりがな）:　佐野貴史（さのたかし）
所属：　株式会社ＨＳ情報システムズ
氏名（ふりがな）:　井村優太（いむらゆうた）
所属：　株式会社ＨＳ情報システムズ
氏名（ふりがな）:　服部一宏（はっとりかずひろ）
所属：　株式会社ＨＳ情報システムズ

〈要旨〉

品質に関する指標はある、生産性に関する指標は規模あたりとして表現できる。
そんな中で、品質・生産性の両面から見れる「ものさし」が欲しいと考えた。
開発時のレビュー指摘・バグにかかるコストを「ものさし」とし、
プロジェクトの業務種類や開発特性、規模により指摘・バグにかかる対応コストの
推移をグラフにし、理想とされるコスト、無駄コストを明確にしたコストモデルを作り、
生産性向上や品質向上の「ものさし」として利用を進めている。

〈キーワード〉

・生産性向上
・工程別の指摘・バグ・故障・障害
・前工程で抽出すべき指摘・バグ
・指摘・バグの対応コスト

〈想定する聴衆〉

品質保証の方、ソフトウェアエンジニア、プロジェクト・マネージャ

〈適用状況〉

□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

生産性の向上が当社の目標になっており、生産性をはかる「ものさし」を検討する必要があった。品質向上を進めながら、生産性を測る共通の「ものさし」を明確にし、品質・生産性向上を目指すために、「ものさし」づくりを推進した。

(2)改善前の状態

品質向上や生産性の向上が経営課題であったが、生産性をはかる「ものさし」がなく、概ね原価低減といった活動が進められていた。その原価低減が正しい姿なのか、もっと明確な指標はないかを模索していた。

(3)改善前の状態をもたらした原因（因果関係）

弊社はプライムベンダーであり、多くの開発は委託先に発注している状態である。実際の委託先のマネジメントは詳細まで出来ておらず、品質向上や生産性向上の対応や「ものさし」について、検討をする状況に踏み込めていなかった。

(4)計画した変更内容

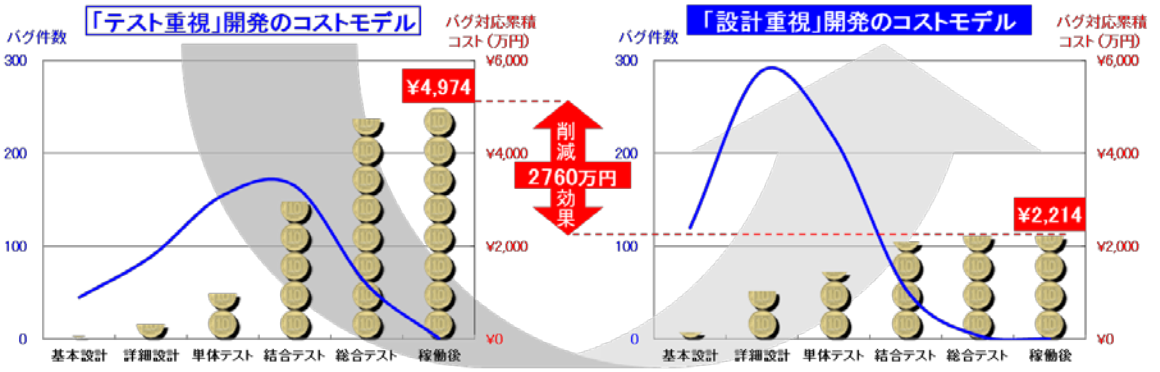
各工程でのレビューやテストによる指摘やバグの数、分類については、既に実行しており、工程別の指摘・バグ・故障・障害の分類として、「本工程で抽出すべき指摘・バグ」と「前工程で抽出すべき指摘・バグ」の分類もできている。

また、「前工程で抽出すべき指摘・バグ」については本工程での品質についてのレビューも行われている。今回、その状況をコストとして見える化する取組みを実施した。

１件あたりの工程別指摘・バグ対応コストを設定し、下流工程重視の形から、上流工程重視の形にコストを見える化することにより、コストモデルという「ものさし」を定義し、生産性向上を目指す試みをすることとした。

| 工程 / ｺｽﾄﾓﾃﾞﾙ定義 | 基本設計 | 詳細設計 | 単体テスト | 結合テスト | 総合テスト | 稼動後 |
|---|---|---|---|---|---|---|
| 相対コスト ※1 | 基準 | 3倍 | 5倍 | 10倍 | 25倍 | **200倍** |
| 1件あたりのバグ対応コスト ※2 | **¥12,000** | ¥36,000 | ¥60,000 | ¥120,000 | ¥300,000 | ¥2,400,000 |

※1：相対コストの考え方は、"JASPIC SPIJapan2009 奈良隆正「ソフトウェア品質保証の方法論、技法、その変遷」"を参考に設定。お客様に合わせチューニング。
※2：1人月100万円とすると、時間6000円(100万円/20日/8時間)。基本設計でのバグ1件に係る対応時間を2時間とし、コスト単価を12,000円と設定。お客様に合わせチューニング。



⑸変更の実現方法

１．各工程での指摘・バグを集め、分類
各工程での指摘・バグを収集し、当該工程で摘出する指摘・バグであったか、または、別途の工程で摘出する指摘・バグであったか、を分類し、数値（件数）を明確にした。

２．各工程での指摘・バグ対応コストの算出
各工程での１件あたりの指摘・バグ対応コストを設定し、
１の分類にあてはめて、抽出された工程ごとの指摘・バグ対応コストを算出した。
※バグ対応コスト：現状は１件あたりの工程別指摘対応コストの算出ができていないため、別途の論文より採用した。

３．各工程での無駄コストを明確にする
各工程で、当該工程で摘出すべき指摘・バグを「必要な対応コスト」とし、
上流工程で摘出すべき指摘・バグを「無駄な対応コスト」とし、見える化する。
※無駄コスト：工程ごとに指摘・バグ対応コストを算出し、
さらに上流工程で摘出すべき指摘・バグを明確にし、上流工程で摘出していればかからなかったコストを算出し、無駄なコストとした。

４．同様なプロジェクト、同様な規模のプロジェクトでコストモデル作成
同様な種類のプロジェクト、同様な規模のプロジェクトのサンプルを複数集め、
全工程を俯瞰したコストのモデルづくりを行った。
※コストモデル：業務種類や開発特性、規模により開発コストの推移をグラフにし、
それぞれのモデルとして、理想とされるコスト、無駄コストを明確にしたもの。

５．コストモデルから目指す生産性を明確にする
コストモデルが出来たところで、無駄と思われるコストを３０%削減する、
といった活動の指標をつくることができる。

６．本番リリース後の品質に関しての考察
コストモデルが出来上がってくると、同様のプロジェクトや同様の規模の
プロジェクトでの予測ができるようになる。同様のプロジェクトでの工程別指摘・
バグの抽出が少なかったり、多かったりした場合に、それぞれその後の工程の
注意点などをプロジェクトにフィードバックすることができる

７．自社の「ものさし」作成
将来的には、コストモデルのパターンを多く集め、
自社の工程別指摘・バグ対応コストを実に近いものとして作成することができる

⑹変更後の状態や改善効果

１．コストモデルから目指す生産性を具体的にすることができる
自社のコストモデルを特性や規模が同様のプロジェクトに当てはめ、
無駄と思われるコストを明確にし、無駄コストの３０%削減する、
といった活動の具体的指標である「ものさし」をつくることができた。

２．次工程、本番リリース後の品質に関しての想定ができる
自社のコストモデルが出来上がってくると、同様の業務種類や開発特性、
規模のプロジェクトでの予測ができるようになる。工程別指摘・バグ対応コストが
少なかったり、多かったりした場合に、現在の工程、またその後の工程に対する
注意点などをプロジェクトにフィードバックすることができた。
また、最終工程での指摘・バグコストが低くなる傾向がある場合には、
指摘・バグが収束してきており、本番リリース時の障害も少ないといった考察も
可能である。本番リリース時のリスクの目安ともなる。

(7)改善活動の妥当性確認

自社の「ものさし」が出来、目指す指標や改善する指標が精緻になる。
将来的には、自社のコストを出来るだけ精緻にとり、コストモデルのパターンを多く集め、
工程別指摘・バグ対応コストを実に近いものとして作成することができる。

1B1「開発現場を救うプロセス改善の進め方」井谷健一(パナソニックアドバンストテクノロジー)

〈タイトル〉: 開発現場を救うプロセス改善の進め方

〈サブタイトル〉: Redmine/Subversion を利用した具体的な実践方法

〈発表者〉

氏名（ふりがな）: 井谷健一（いたにけんいち）
所属：パナソニックアドバンストテクノロジー株式会社

〈要旨〉

自社で実施していたプロセス改善の経験を活かして、他社のプロセス改善の支援を２０１２年から実施しており、その取り組みを通じて、支援を効率的に実施する方法を確立することができ、その効果を確認することができた。但し、ただ単に他社で効果があったツールを導入や組織標準を構築するだけでは本質的な改善ができない。重要なのは、開発者の主体性である。その主体性を引き出すために、初期の段階で支援者が開発者
に対して、現状の課題を解決する方法を提供し、改善効果を開発者が実感してもらうことが必要である。

〈キーワード〉

プロジェクト管理、Redmine、Subversion、プロジェクト見える化、RestAPI、
管理支援ツール、プロセス改善、開発現場主導、意識統一、チケット

〈想定する聴衆〉

組み込み開発のソフトウェアエンジニア、プロセス改善担当者

〈適用状況〉

■多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

２０１２年までエンジニアリングセンターという社内の専門組織として、社内のプロセス改善、組織標準の構築などの作業を担っていた。しかし、会社の方針が変更となり、他部門と同様に、他社から収入を得ることを求められたため、これまで社内で実施していたプロセス改善、組織標準の構築などの作業を他社に対して実施することになった。

但し、他社に対してプロセス改善活動に対して対価を払うだけの価値を認めてもらう必要があるが、当初は、その方法がわからなかった。そのため、まず、大きな課題が発生している支援先に対して個別に対応する活動から開始した。

これらの支援活動を通じて、以下の４つのことがわかった。

① 現実的な対策案の提示と早期の活動開始
② 課題の早期特定と導入効果確認のための支援先の活動の見える化
③ 課題に対して迅速に対応するための武器（道具）の準備／利用
④ 支援先の支援はやりすぎない（開発者主体）

２０１２年、２０１３年の組み込み開発に対する支援活動を通じて、Redmine/Subversion を利用した支援方法を確立した。この支援方法を利用して各開発現場に合ったプロセス改善を実行することで開発現場主導の自発的なプロセス改善のサイクルを定着させるための取り組みを紹介する。

(2) 改善前の状態

プロセス改善が進まない（定着しない）組織の典型的な状況として以下のような内容があげられる。改善活動を全くしていないわけではなく、むしろ熱心に実施している組織もあった。

① 壮大な改善計画だけが存在（改善計画立案）
② 膨大な標準帳票が存在（組織標準構築）
③ 職人気質な個人的に優秀なメンバーに頼ったプロジェクト運営（個人能力依存）

(3) 改善前の状態をもたらした原因（因果関係）

以下に各課題に対する原因を記載する。

① 壮大な改善計画だけが存在
   - 開発者または組織の実力に応じた現実解の提示がない
   - 改善活動推進が開発者任せになっている
② 膨大な標準帳票が存在
   - 組織標準は各プロジェクトでそのまま使用できない
   - 開発者は組織標準を上手に利用できていない
③ 職人気質な個人的に優秀なメンバーに頼ったプロジェクト運営
   - 個人の能力だけでも対応可能な規模である（あった）
   - 他者に仕事を渡すことが不得手である

開発規模が個人レベルで管理できる（5 名以内）の場合、(3)-③であっても問題なくプロジェクト運営ができる。しかし、開発規模が増大し、関係者が増加した場合、(3)-③では問題が発生する。その結果、(3)-①、(3)-②の取り組みを行ったものの開発現場に定着することなく、プロジェクトが破綻する事例を多く見てきた。

特に、異なる組織の文化の異なる人と一緒にプロジェクトを実施すると顕著に現れる。

原因として、以下の点にあると考える。

１）「当たり前」、「常識」の違い
２）「言葉」の違い
３）「プロセス」、「手順」の違い

(4) 計画した変更内容

解決方法として、以下の点が重要と考え、計画した。

① 開発者のプロセス改善の時間の確保と意欲の向上

[実施内容]

開発者の困りごとの解決

開発者と支援者が協力して改善計画の策定

[理由]

開発者は自分たちの利益のあることに対してしか動かない

まず、現状の困りごとの解決により工数を確保するとともに開発者と支援者の信頼関係を構築する

② 開発実施内容／意識の統一

[実施内容]

開発者の現場に即した各種ワークフローの構築

開発者と構築した内容を共有

[理由]

「当たり前」、「常識」の内容を明確化し、共有する

「言葉」の統一を図り、意識の齟齬をなくす

③ プロジェクトの見える化

[実施内容]

プロセス、Redmine による見える化

開発者とプロジェクトの実態を共有

[理由]

正しい状況をプロジェクト全体で共有し、早期に課題解決を図る

④ 導入支援の効率化

[実施内容]

プロセスの再利用

RestAPI による Redmine 支援ツールの開発

[理由]

早期に、現状の開発現場での課題解決を実施する

開発者のプロセス改善に費やす工数を最小化する

(5) 変更の実現方法

〇活動実施済（定量的効果確認）

△活動実施済（定量的効果未確認）

×今後実施予定

実現方法のポイントは、初期の段階では、現状の開発者が実施している内容自体は大幅に変更することなく、各種ツールや仕組みを利用して具体化するための支援を行うことである。

① 開発者のプロセス改善の時間の確保と意欲の向上

・ 支援者が開発者が困っていることを早期に改善し、プロセス改善工数の捻出する

（△）
- 支援者が開発者と協力して最終ゴールを明確化する(△)
- 支援者が開発者と協力して最終ゴールと現状のギャップ明確化した上で、現実的な改善計画を策定する(△)
- 開発者以外のメンバーがプロセス改善の支援を実施する(△)

② 開発実施内容／意識の統一
- STEP1
  - 支援者がプロジェクト開始時にプロジェクト管理面で確実に実施できる仕組みを構築する(○)
  - 開発者が Excel 管理または管理未実施から Redmine 管理とし、支援者は Redmine 管理の導入支援を実施する(○)
- STEP2
  - 開発者が共有ファイル管理から Subversion 管理とし、支援者は Subversion 管理の導入支援を実施する(○)
- STEP3
  - 支援者が開発者と協力してエンジニアリングプロセスを定義する(△)
  - 支援者が開発者と協力してエンジニアリングプロセス定義にしたがって構成管理フォルダ構成を確定する(△)
  - 支援者が開発者と協力して進捗管理方法を定義する(△)

③ プロジェクトの見える化
- 支援者が開発者と協力して集約されたデータを見える化する(○)
- 開発者が見える化されたデータ基づき、活動を実施する(○)

④ 導入支援の効率化
- Redmine の問題(欠点)を補完するツールを開発することで各種作業の効率化を図る(○)
- 作成したツールを利用して実施済 Redmine チケットの雛形化を実施する(○)

(6) 変更後の状態や改善効果

① 開発者のプロセス改善の時間の確保と意欲の向上
- Excel 帳票で管理していたプロジェクトが Redmine 導入/支援により、以下の効果が現れ、プロセス改善の工数を確保することができた事例
  - 平均解決所要日数は、69.9 日から 18.8 日に削減
  - 解決所要日数は、全体的に前へシフト
  - メールのやり取りが 15%削減
  - リアルタイムに情報を参照することが可能
  - チケット番号をベースにコミュニケーションを実施(議論、会話、議事録への記載など)

② 開発実施内容／意識の統一とプロジェクトの見える化
- Redmine チケットデータを利用したプロジェクトの見える化の事例
  - 電子メールでチケット状況を送付
  - 客観的なデータ（担当チケット数）を利用することで本質的な責任部署の見える化

③ 導入支援の効率化

・ 各ツールの利用により、Redmine の操作が削減

| ツール | 操作 | 時間 | 手動 | 差分 |
|---|---|---|---|---|
| チケット一括登録 | ①チケット発行(300 件) | 7 時間 | 16 時間 | 9 時間 |
| | ②ｽｹｼﾞｭｰﾙ設定(300 件) | 1 時間 | 8 時間 | 7 時間 |
| | ①＋② | 8 時間 | 24 時間 | 16 時間 |
| チケットデータ登録 | 帳票⇒Redmine(500 件) | 8 時間 | 80 時間 | 72 時間 |
| チケットデータ出力 | Redmine⇒帳票(500 件) | 4 時間 | 100 時間 | 96 時間 |

・ 各ツールの利用／作業プロセスの再利用により、導入時のリードタイムが削減

| 実施作業 | ﾌﾟﾛｼﾞｪｸﾄ 1 | ﾌﾟﾛｼﾞｪｸﾄ 2 | ﾌﾟﾛｼﾞｪｸﾄ 3 | 次回 |
|---|---|---|---|---|
| STEP1(主に Redmine) | 12 ヶ月 | 3 ヶ月 | 1 ヶ月 | 0.5 ヶ月 |
| STEP2(主に Subversion) | 未実施 | 3 ヶ月 | 1 ヶ月 | 0.5 ヶ月 |
| STEP3(ﾌﾟﾛｾｽ定義,進捗管理) | 未実施 | 未実施 | 1 ヶ月 | 0.5 ヶ月 |

(7) 改善活動の妥当性確認

プロセス改善活動を実施するにあたって重要な点は、改善開始時に開発者が実施している活動内容を極力変更しないで、迅速に改善することであった。このようなことを取り組んだ結果、改善のための工数の確保と取り組みに対する信頼感を獲得することできた。

さらに、迅速に改善提案するためには、改善するための武器（道具）が必要であったため、支援実施結果を再利用するための雛形の作成やツール開発を実施した。

以上の結果、Redmine/Subversion を利用したプロジェクト管理方法の確立と効率的なチケット発行、更新、管理が実現でき、支援先のプロセス改善を効率的に、実施することができるようになった。

今後は、エンジニアリングプロセスの実施内容および支援方法を確立する必要がある。

但し、Redmine/Subversion や開発したツールはあくまでも効率化する手段であり、本質的なプロセス改善を実施するためには、以下のことが重要である。

① プロセス改善は、開発者の主体性が絶対必要

② 開発者の主体性は、「理解→納得」の結果、「行動」として出現

・ 開発者の理解は、運用ルールを明確化した上で心理的負担および工数的負担の軽減策を検討し、計画的に実施することから得られる

・ 開発者の納得は、感覚でなく、客観的な事実（データ）に基づく判断／対応や繰り返しの周知徹底を実施することから得られる

1B2「プロセス改善活動をスムーズに立ち上げるための取り組み～改善活動 Startup の作成と展開～」大川瑠里子(NTT データ)

〈タイトル〉: プロセス改善活動をスムーズに立ち上げるための取り組み
～改善活動 Startup の作成と展開～

〈サブタイトル〉:

〈発表者〉

氏名（ふりがな）: 大川　瑠里子（おおかわ　るりこ）
所属：NTT データ

〈要旨〉

NTT データのコーポレート SEPG は、NTT データグループ内のさまざまな組織におけるプロセス改善活動を 2001 年より支援してきた。プロセス改善活動において、立ち上げフェーズは重要であり、立ち上げをスムースに質よく実施することが、その後の改善活動の質や効果に大きく寄与する。昨年までシステム開発プロジェクトに従事しながらプロジェクトの改善に問題意識を持っていた発表者が、コーポレート SEPG として、支援先組織の改善活動の立ち上げをよりよく行うための取り組みの必要性を認識して「改善活動 Startup」の作成を着想した。
本発表では、「改善活動 Startup」の作成に至った背景、その構成と内容、展開について紹介し、本取り組みが支援先組織の立ち上げにどのような効果を及ぼしたのか、今後の展開に向けた課題について考察する。

〈キーワード〉

プロセス改善活動、立ち上げ、ノウハウ有形化、ベストプラクティス、CMMI

〈想定する聴衆〉

組織の SEPG

〈適用状況〉

□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

発表者自身の背景：
以前はシステム開発プロジェクトに 5 年間在籍していた。当時は目の前の作業や課題の対処に追われ、自身や自チーム・組織の改善の必要性を感じながらも、改善の進め方に対する知識や上位層の協力が不十分で、思うような改善を進められなかった。2013 年 10 月以降、コーポレート SEPG として、NTT データグループ内でプロセス改善活動を希望する組織に対する支援を行っている。

取組の動機：
コーポレート SEPG として、改善活動の立ち上げから支援した、改善活動の経験が浅い組織からのフィードバックがきっかけである。当該組織は改善活動に割り当てられるリソースが限られており、組織の SEPG に許されたリソース・活動への動機付けが不十分な状態である。
当該組織のスポンサーと合意のうえで改善活動に着手して、改善計画を作成するための初回アセスメント（2014 年 2 月）を実施することになった。これまでの組織と同じように、コーポレート SEPG が必要と考える説明をその都度詳細に提供した。改善活動対象組織の決め方や、インタビューの際の注意点、CMMI のモデル解釈等である。
しかし、初回アセスメント終了時の振り返り議論の中で、「改善活動の立ち上げやギャップ分析の中で、何を誰がどのくらい実施するのか、どれが重要なタスクかよくわからなかったので、稼働調整が難しかった。また、作業や成果物の関連が見えず、自律的に動けなかった」というフィードバックを受けた。この意見は改善活動の根本に迫る問題提起である。何のために何をするのかわからなくてもギャップ分析が終わってしまうのである。
これを受けて、これまでコーポレート SEPG が提供してきた CMMI ベースのギャップ分析に関する先述のような説明の再構成が必要であることに気付いた。この気付きを受けて、プロジェクトから離れて間もない発表者が、コーポレート SEPG として、改善活動の経験が浅いメンバーでも改善活動を始める際のタスク、成果物、確認観点、等を理解できるコンテンツを「改善活動 Startup」と銘打ち、活動立ち上げ時に提供するべく作成に着手した。

(2)改善前の状態

プロセス改善活動の経験が浅い SEPG で構成される組織において、限られたリソースで改善活動を効果的に進める必要がある。
組織の SEPG は改善活動の知識が少なく、活動の全体像や有効事例を把握せず、スポンサーや管理者層から言われるままにアクションに対応している。

期待する効果：
組織の SEPG が、今後の改善活動に効果的に取り組めるようになること。
また、活動の全体像や有効事例を理解することで、自律的かつ能動的に動けるようになること。

(3)改善前の状態をもたらした原因（因果関係）

事象と原因の洗い出しのため、コーポレート SEPG と組織の SEPG で振り返りの議論を実施した。
(2)で述べた、知識が乏しいという事象の一因は、(1)で述べたコーポレート SEPG からの活動の詳細説明が、プロセス改善活動の経験が浅い SEPG には難易度が高いためである。個別の成果物の内容や CMMI モデルの知識は身についても、成果物の関連性や活動の相対的な重要度までは理解できないのである。

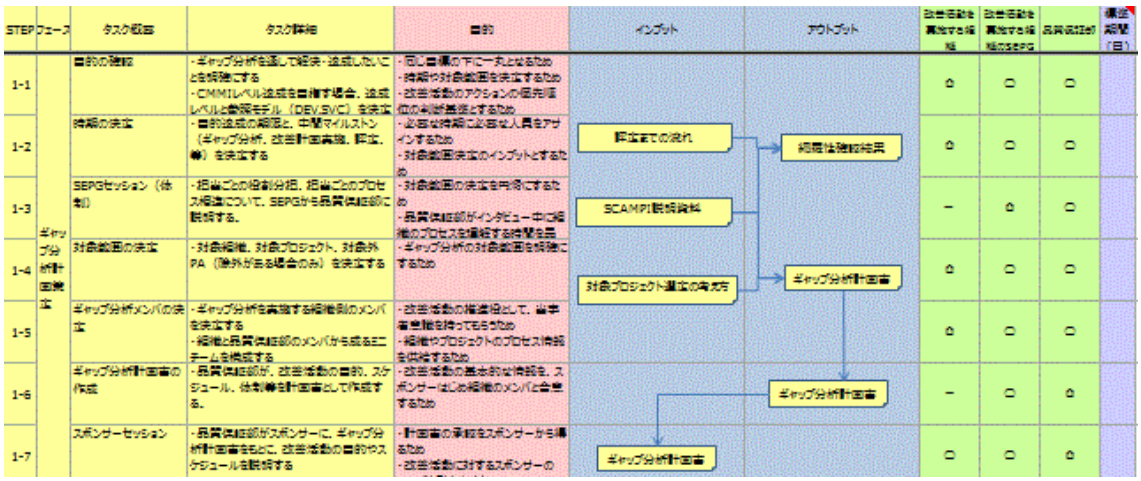
(4)計画した変更内容

１．プロセス改善活動の立ち上げ時に支援先組織に提供していた各種資料を再構成して、改善活動初心者でも改善活動の全体像や有効事例を容易に把握できるパッケージ(「改善活動 Startup」) を作成する。作成にあたって、現在支援中の組織の SEPG から、要望やフィードバックを収集する。
２．作成後の検証にも協力してもらい、次組織への展開に向けた改善にも取り組む。
３．改善活動に新規着手する組織の SEPG に適用して、フィードバックを獲得する。

(5)変更の実現方法

Startup の構成を以下の通り設計した。

０．導入説明
１．フロー
　１－１．タスクと成果物
　１－２．確認観点
　１－３．成果物様式・事例
　１－４．用語集




図：「１－１．タスクと成果物」イメージ

メインの構成要素は「１－１．タスクと成果物」である。
改善活動の全体フローと各フェーズの目的、詳細タスク、役割分担、入出力などを詳細に定義している。

さらに、今回の支援先組織からの質問、コーポレート SEPG が把握している過去の典型的な質問・陥りやすいトラップをもとに、各詳細タスクに対応する「１－２．確認観点」(観点数は１タスクあたり最大１０程度) を設けた点も特徴である。これは組織の SEPG が自らの作業をチェックし、改善活動のタスクの目的や完了条件をよりよく理解できるようにするためである。
また、「１－３．成果物様式・事例」では、NTT データグループ内の改善活動の有効事例と、それを様式化したドキュメントを提供する。この様式は「１－１．タスクと成果物」の各成果物に紐づいている。例えば、プロセス改善活動計画書や、ギャップ分析のスケジュールフォーマット等である。「１－２．確認観点」は、本様式を使用して成果物を作成する際のチェックリストとしても活用できる。

工夫した点、特筆したい点：
改善活動の経験が浅いメンバーを想定利用者として、以下の点に留意した。
・活動の全体像と各フェーズの目的を示すことで、利用者が改善活動の到達点と現在のタスクの位置づけを容易に理解できるようにしたこと。
・各フェーズを詳細タスクに分割して役割、入出力、確認観点を明確化することで、利用者が改善活動の流れと各タスクの位置づけを正確に把握できるようにしたこと。
・タスクの目的や成果物間の関連を示すことで、タスクを行う必要性や、実施しないとどんな問題が発生するのかを意識しながら活動を進めるように促していること
・専門用語の解説や詳細な説明をつけることで、利用者の理解を助けたこと。
・本取組の契機となった振り返りの議論と同様に、Startup の説明を対面で実施することで、利用者の生の声を収集すること。
・コーポレート SEPG が蓄積してきたノウハウやベストプラクティスを有形化し、有効活用できるように構成したこと

(6)変更後の状態や改善効果

冒頭で紹介した支援先と、新規に改善活動に着手する社内の別組織の 2 組織への適用を計画している。適用を 2014 年 7 月に開始し、効果測定を 2014 年 10 月に完了する予定である。

効果の測定および検証方法として、適用先へのアンケート、適用先からの質問内容の確認、および適用先 SEPG との対面での振り返りを実施する。
これにより、冒頭に述べたような「改善活動の全体像や知識が不足していることに伴う質問や活動の停滞」が防止できていることの確認、「冒頭の組織において以前より能動的に活動できているかどうか」のヒアリングを行う。

(7)改善活動の妥当性確認

(6)で述べた改善効果が見られるかを確認することで妥当性確認を行う予定である。
効果が十分でない場合は、振り返りであがった意見を今後の改善事項として取り組む。

1B3「効果をもたらす“現場支援”の仕組みの改善」柏原一雄(デンソークリエイト)

〈タイトル〉: 効果をもたらす“現場支援”の仕組みの改善
〈サブタイトル〉: ～改善支援組織が形式的・表面的な活動に陥らせていた？～
〈発表者〉
氏名（ふりがな）: 柏原一雄（かしわばらかずお）
所属：株式会社デンソークリエイト 事業推進センター 品質推進室
〈共同執筆者〉
氏名（ふりがな）: 竹下千晶（たけしたちあき）
所属：株式会社デンソークリエイト 事業推進センター 品質推進室

〈要旨〉
現場をプロセス改善でありがちな表面的・形式的な活動に陥らせず、効果的に標準プロセス等の仕組みを活用できるよう、“現場密着型・支援型”の改善支援組織で支援活動を実施してきた。
当初は、現場に対し、表面的・形式的な活動を助長しかねない等、現場にとって有益な活動ができているとは言い難かった。その原因を分析したところ、「軸足を現場に置かず、表面的な事実だけを見て、型にはめようとしていた」ことがわかった。この問題を解決するための仕組みを構築し、導入した。

〈キーワード〉
SQA、SEPG、プロセス、改善、定着、支援、形式的、表面的、主体性

〈想定する聴衆〉
SEPG、SQA、PMO、PM の方

〈適用状況〉
■多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

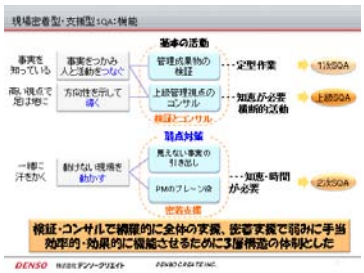
〈適用可能性に関する制限〉
□汎用性がある、
■類似プロジェクトにも適用可：具体的な類似点（現場のプロセス改善を支援する“間接部門”）
□自プロジェクトのみ

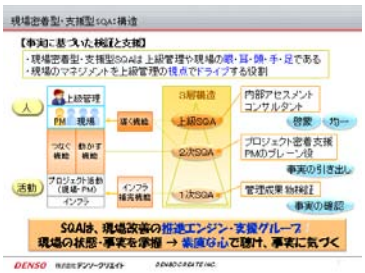
〈発表内容〉

(1)背景

組込みソフトウェア開発では、特徴である終盤での仕様変更の多発や大規模化により、プロジェクトの状況を把握し制御することは難しくなっている。その結果、現場は混乱状態に陥り疲弊してしまう。このような状況を脱するため、弊社ではトップダウンによるプロセス改善に取り組んできたが、厳格なプロセス遵守を現場に強いることで、現場は余分な仕事が増え、「やらされ感」のもと形式的・表面的な活動に陥り、より疲弊させてしまっているように見えた。

形式的・表面的な活動ではなく、現場がプロセスを理解し、有効に活用できるようにするためには、改善推進組織がプロセス遵守を強いるのではなく、現場に降り立ち、上位の視点も持ちながら現場と一緒に汗をかき支援をする機能を持つことが有効だと考え、「現場密着型・支援型 SQA」という三層構造の枠組みを構築した。(SPI Japan2011 で発表)。





この三層構造のうち、現場の活動に特に密接に関係し、影響を与えるのは「2 次 SQA」である。他の二層は横断的に支援するのに対し、2 次 SQA は、初心者の PM や、混乱状態、形式的・表面的な活動に陥っているプロジェクト等、弱みのあるプロジェクト・PM を対象に、PM のブレーンとなって現場でのプロセスの活用や改善を支援する。「現場密着型・支援型 SQA」が狙い通りの機能を果たすには、2 次 SQA の活動の質を高く保つ必要があった。

(2)改善前の状態

2 次 SQA による『支援活動』を開始した当初は、以下のような状況も散見し、質が高い活動ができるとは言い難かった。

- 理想形の押しつけ
  混乱している現場に対して、べき論でやることを増やしたり、一度に多くの問題の是正を促すことで、現場が更に混乱・疲弊する。
- タイミングの悪さ
  プロジェクトの改善点を掴むのが遅く、改善点を明らかしたときには既に手遅れで、現場が混乱状態に陥り、手が打てない。

『支援活動』は、「理想形の押しつけ」や「タイミングの悪さ」によって、2 次 SQA が頑張れば頑張るほど、形式的・表面的な活動に陥ることを防ぐどころか、そちらに導くような活動になっていた。

(3)改善前の状態をもたらした原因 (因果関係)

現場で、形式的・表面的な活動に陥ったり、必要性や効果を感じられず活動が続かない原因の多くは、以下のように現場の個別の状況・状態に関係なく、一律同じプロセスを同じやり方で適用しようとしていたことにある。

- “現場の目標”や“現場の抱えている問題(リスク)”と活動が繋がらない
- “現場で実際に実施されている活動”に同じような活動が既にある

・　活動が“現場の状態（習慣、体制、負荷状況）”に合わず、実行できない

2 次 SQA の『支援活動』は、一律同じやり方にはめる支援ではなく、現場に降り立ち、現場の目線も持ちながら、PM のブレーンとして、現場の状況・状態に合わせた支援でなくてはならない。

しかし、2 次 SQA は“間接部門・業務”という特徴から、現場から離れた立場・目線になりやすく、支援という行為をすることが目的にもなりやすい。その結果、自分たちでも気付かぬうちに(2)で示した状態となることが多く、現場に合わせた現場のための活動ができていなかった。

“間接部門・業務”でありながら、現場の状況・状態に合わせた支援を実施するためには以下が必要と考えた。

・　プロジェクトの特徴や起きている事象等、あらゆる事実を掴む
・　掴んだ事実をもとに、優先度を踏まえて手を打つ問題・リスクを特定し、実現可能性を考慮し改善に導く

これらは、自然にできてしまう人にとっては簡単に思えることだが、間接的な立場で関わる 2 次 SQA にとっては難しい。難しい原因は以下と考えた。

(a)　道具がない
何を使って、何を見て様々な事実を掴めばいいか分からない。

(b)　経験・スキルがない
優先度を踏まえて手を打つ問題・リスクを特定したり、改善策の実行可能性の判断をするには、経験や事実を元に分析をするスキル等が必要だが、それらが不足している。

(c)　目的のずれ
分かっているつもりでも、どうしても“支援をすること”が目的になりやすい。目的がずれてしまうと、道具があり、経験・スキルがあっても、適切な判断や支援ができない。

(d)　現場からの信頼の獲得
いくら適切な判断や支援をしようとしても、現場が受け取ってくれなければ効果がでない。間接的な立場で、指揮命令権なく現場を動かすためには、信頼を得る存在でなくてはならない。信頼を得るためには、現場が 2 次 SQA の言葉に納得でき、効果を実感できることが重要である。

(4) 計画した変更内容

(3) で述べた、2 次 SQA の『支援活動』の課題((a)～(d))を解決するために、以下の 3 つの仕組みを構築した。

①　プロジェクトの状態把握を支援するツール
プロジェクトが実行可能な改善策かを検証するため、その入力となるプロジェクトの状態を漏れなく把握するための道具を用意。･･･課題(a)

②　支援計画を基盤としたプロセス
現場（プロジェクト）の狙いやゴールに即した『支援活動』の目的を、支援計画にて明確に示す。・・・課題(c)
目的を明確にした上で、具体的な『支援活動』の内容も支援計画に落とし込む。支援計画をベースに、SQA のチームとしてレビューする。計画立案時だけでなく、以降の活動中にも計画と照らしながら、実際の『支援活動』についてチームでレビュー、議論する。また現場の上級管理者とも事前に認識合わせをすることで、現場の視点や上位者の視点を加えることで、2 次 SQA 担当者個人の経験やスキルを補い、質を高める。･･･課題(b)
支援計画は、PM とも認識合わせを行い、現場が納得した状態で活動を進める。･･･課題(d)

③　現場とのコミュニケーション

『支援活動』の実施内容や、それによるプロジェクトの変化・効果を、定期的に、PM および現場の上級管理者と共有する場を設け、納得感や効果の実感を高める。その機会に、現場の意見や期待を吸い上げると共に、『支援活動』がどう受け止められているかを把握する。・・・課題(d)

(5)変更の実現方法

(4)で挙げた 3 つの仕組みの実現方式を述べる。

① プロジェクトの状態把握を支援するツール

プロジェクトの特徴や起きている事象等、あらゆる事実を漏れなく掴むため、プロジェクトから引き出したり、観察すべき情報を確認項目一覧という形で整理した。確認項目は、下記のような観点で大別し、詳細な項目を抽出した。

・プロジェクトの概略、特徴

そもそもどういう業務なのか、どんな計画なのか等、プロジェクトの基本情報を掴むための項目。

業務範囲、目標、スケジュール、体制・役割分担等。

・人（プロジェクトメンバ、関係者）

人の意識や考えが行動に影響するため、プロジェクトメンバや関係者を理解するための項目。

プロジェクトメンバの経歴や特徴、思い・考え等。

・仕事のやり方

標準のプロセス等と実際の活動には乖離があることはよくあるため、定義されたプロセスではなく、実際にどんな仕事の仕方をしているのかを掴むための項目。

・プロジェクトの健全性

プロジェクトが不健康な状態（高負荷、疲弊、無秩序等の状態）では、改善点がわかっても、それに対する対処をするための力がなく、改善が進まないことが多い。改善活動への対応力を見極めるための項目。

基本的なルールが守られる風土、疲弊度合い等。

| 項目 | | データ |
|---|---|---|
| QCD | 品質 | 手戻り工数比率 |
| | 計画／進捗 | 計画工数と実績工数の差（平均） |
| | | スケジュール変更の発生件数/工数推移 |
| | | 遅れ工数推移 |
| | | リソース負荷推移 |
| | 忙しさ | 残業工数（1人平均） |
| | | 休出日数（1人平均） |
| | | 出勤時刻、帰宅時刻 |
| 計画の適切性 | コミュニケーションの取りやすさ | PMの出張割合（ざっくり週あたり何H） |
| | | PMが席を外している時間の傾向（ざっくり日あたり何H） |
| | コミュニケーションの取り方の適切性 | 部下指導のような時間の傾向（ざっくり週あたり何H） |
| | | 席で話（電話含む）をしている時間の傾向（ざっくり日あたり何H） |
| | 状況把握の可能性・容易性 | 重要成果物の有無 |
| | | 各会議の議事録作成状況 |
| | | 帳票のメンテ状況 |
| | | PMの受信メール数（平均） |
| | | PMの未読メール数（平均） |
| | 役割分担の適切性 | 担当者毎の実績工数のばらつき |
| | 会議計画の適切性 | 進捗会議の平均欠席人数 |
| | 負担・余裕 | PMの工数比率の推移 |
| | | プロジェクト全体の工数比率（何に時間を使っているのか？） |
| 生活習慣 | 整理整頓する | PMの受信メール数（平均） |
| | | PMの未読メール数（平均） |
| | | 各会議の議事録作成状況 |
| | | 帳票のメンテ状況 |
| | | プロジェクトフォルダ内のゴミファイル数の傾向 |
| | 仕事を溜めない | 工数入力定着率 |
| | | アクションアイテム残件数 |
| | | レビュー承認までの平均所要日数 |
| | 時間を守る | 期限切れアクションアイテムの件数 |
| | | 週報の提出遅れ率 |
| | | 進捗会議の平均遅刻時間 |
| | 約束をする | レビュー案内送付タイミング |

② 支援計画を基盤としたプロセス

『支援活動』を下記のように支援計画を活動の基盤とするプロセスとした。

（ｉ）支援計画の立案

「確認項目一覧」に基づいて収集した情報を元に、『支援活動』でプロジェクトと共に目指す目標・ゴールとその評価方法、重点的な支援項目、現場とのコミュニケーション計画を設計する。

（ii）チーム内レビュー

収集した情報、支援計画の内容を SQA チームでレビューし、有効な『支援活動』となる確度を高める。

（iii）PM、現場の上級管理者との認識合わせ

支援計画の内容を現場と認識合わせし、目指す方向を一致させる。その過程で挙がった現場からの意見や要望も踏まえ、必要に応じて支援計画を見直す。

（iv）支援活動の実施と結果・効果の確認

支援計画に基づいて活動を開始した以降は、定期的に活動結果とその効果を、チーム内でレビューする。現場で効果が得られていない場合は、適宜、計画の見直しをしながら、現場の状況にあった『支援活動』を維持する。

③ 現場とのコミュニケーション

『支援活動』の実施内容や、それによるプロジェクトの変化・効果を共有し、現場と双方向のコミュニケーションを取る機会として以下を設定した。

（ｉ）支援報告書

定期的に（基本的には週 1 回）PM および現場の上級管理者に報告書としてまとめて展開。一方的にならないよう、現場が記入するコメント欄を設けた。

（ii）現場の公式レビューへの参加

現場が定期的に開催する公式レビューに参加し、2 次 SQA からも状況報告を行う。支援報告書より期間も範囲も広い視点で、変化・効果を示す。

（iii）SQA 連絡会

SQA チームで、SQA の各種活動の状況確認、活動で掴んだ情報、全社を見渡した現場の状況等を共有し、それを元に活動の方向性等を議論・決定する定期的な場が SQA 連絡会である。そこに、月に 1 回、現場の上級管理者にも参加してもらい、個別のプロジェクトだけではなく全社的な情報の共有した。また、現場の上級管理者から状況の変化を吸い上げ、協力要請を受けることで、2 次 SQA 担当者の配置の調整や、2 次 SQA の支援の方向性見直す場とした。

(6)変更後の状態や改善効果

構築した仕組み導入後の 2011 年度～2013 年度に支援した 14 プロジェクトのうち 11 プロジェクトで、計画立案時に設定した目標を達成し、狙った効果が得られた。

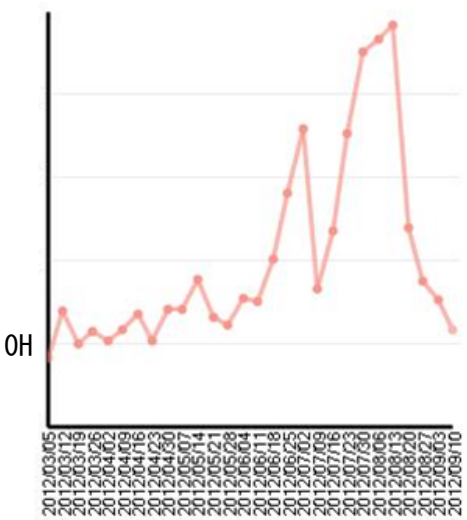
効果的に支援ができた例として、あるプロジェクトの改善効果を示す。

このプロジェクトでは、リリース直前に高負荷状態にならないということを目標に、現場の計画立案・進捗管理に関する活動の改善を支援した。

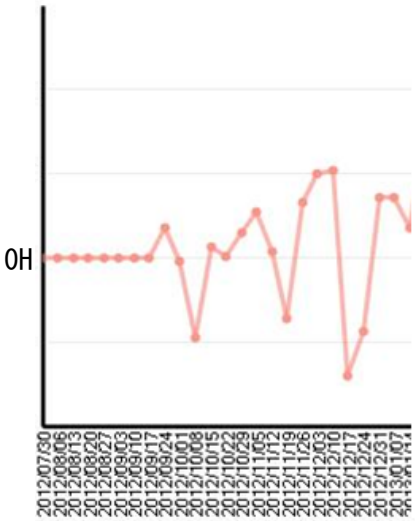
リリース直前に高負荷状態にならないためには、プロジェクト期間を通して遅れが制御できている必要がある。「遅れ工数」という指標を使い、その収束状況により現場の改善効果を確認した。以前は遅れ工数が開発終盤まで広がり続けていたが、改善後は早い時期から遅れ工数が制御できるようになった。

【遅れ工数推移】

改善前




改善後




(7)改善活動の妥当性確認

(6)で示したように、現場と認識合わせした目標・ゴールを達成し、現場も納得できる『支援活動』ができるようになったと判断している。

また、定性的ではあるが、活動中や終了時に「助かった」「ありがとう」という言葉を PM や現場の上級管理者からかけてもらえるようになり、また「2 次 SQA で支援をしてほしい」という現場からの要請も増え、『支援活動』が効果的であり、2 次 SQA が信頼される存在として認められたと評価している。

よって、構築した仕組みは、『支援活動』の質を向上させ、維持できる仕組みであると判断している。

1C1「ソフトウェア構造を見れば品質がわかる！　構造メトリクスとバグの関係」綾井環(テクマトリックス)

〈タイトル〉:
ソフトウェア構造を見れば品質がわかる！　構造メトリクスとバグの関係

〈サブタイトル〉:
最新研究事例紹介と追試結果：構造メトリクスとソフトウェア品質との相関関係

〈発表者〉
氏名（ふりがな）：綾井 環(あやい かん)
所属：テクマトリックス株式会社

〈共同執筆者〉
氏名（ふりがな）：小向 順(こむかい じゅん)、福永 一寛(ふくなが かずひろ)
所属：テクマトリックス株式会社

〈要旨〉
ソフトウェア構造の複雑さとバグの関係について報告を行う。構造の複雑さがソフトウェアの品質や開発活動に与える影響を分析した研究事例が、Sturtevant (2013, Ph.D. Thesis)”System design and the cost of architectural complexity”によって公開された。この研究では「ソフトウェアの中で構造的に複雑な箇所はバグ密度が上昇する」ことを定量的にモデル化し実証している。筆者らはOSSプロジェクトを対象として、同様の結果が得られることを追試にて確認した。構造の複雑さを示す構造メトリクスをソフトウェア開発時のモニタリング対象に含めることで、ソフトウェア品質改善の効率化が期待できると考える。

〈キーワード〉
ソフトウェアメトリクス、バグ密度、リポジトリマイニング、DSM、ソフトウェアアーキテクチャ、バグ予測、ソフトウェア品質改善

〈想定する聴衆〉
品質保証の方、ソフトウェアエンジニア、アーキテクト

〈適用状況〉
□多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、■着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

- 今回の活動の背景・動機

「一生懸命テストをしているのに、一向に品質が安定しない」
「想定外の箇所に影響が及んでいることが多く、たびたび工数超過に陥る」
「リファクタリングを行いたいが、定量的な効果がわからず踏み出せない」

特に既存コードをベースとした流用開発において､多くのお客様からこうした悩みを聞く。ソフトウェアの規模拡大に伴いソフトウェア構造が複雑になった結果、上記のような課題がより顕在化していると考えられる。

近年のソフトウェアサイエンスの研究では、ファイル間の依存度やコードの複雑度などの構造メトリクスが、バグ密度や開発者の生産性と相関関係にあることが分かってきた。

本発表では、構造メトリクスとバグ密度の関係に関する最近の研究成果に対して、筆者らが行った追試の結果と応用例の報告を行う。

(2)改善前の状態

- (一般論として)複雑度が品質等に与える影響、現状の理解
- より深く理解することで得られる期待効果・想定用途

**【Sturtevant[1]の研究背景】**

(a) ソフトウェアシステムの巨大化とシステム設計＆保守の困難化：
大量の要件変更・予測外のビジネス環境変化・技術的変化の急速化・システム構造上のロックイン等の要因

(b) 巨大なシステムは本質的に複雑：

開発面：
ソフトウェアシステムの開発作業では､チーム間を跨ったうえで設計プロセスが分割されていたり、全ての開発作業を理解している人が存在することは稀

ソフトウェア面：
個々の構成要素の独立機能だけでは動作しない。ソフトウェアの自由さ(構造、表現)を起因として、ソフトウェアの実体はサイズに対して、人間が作ったほかのモノに比べるとより複雑

(c) ソフトウェアアーキテクト達の取組：

- 管理可能な単位まで設計を分解する
- 合理的なコストで開発でき、必要な機能を提供する
- 非機能要件的な、保守性、柔軟性、展開可能性、スケーラビリティ、安全性をシステムに附与していく

上記等の取組結果、ソフトウェアは構造化され、図１に示すような構造パターンが構築されてきた。

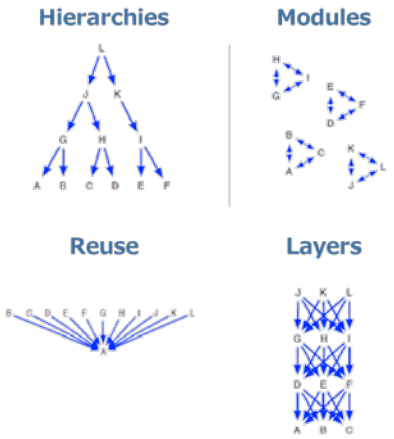



図 1　構造化パターン

(d) ソフトウェアの構造化を行っても複雑さは発生する：
図 2 に示すように、たった 2 か所の設計ルールに違反したソフトウェア実装を行うと、すべての要素間に直接的または間接的な循環依存関係が発生し、ソフトウェアの保守性を損ねると共に、バグの発生危険度も上昇する。

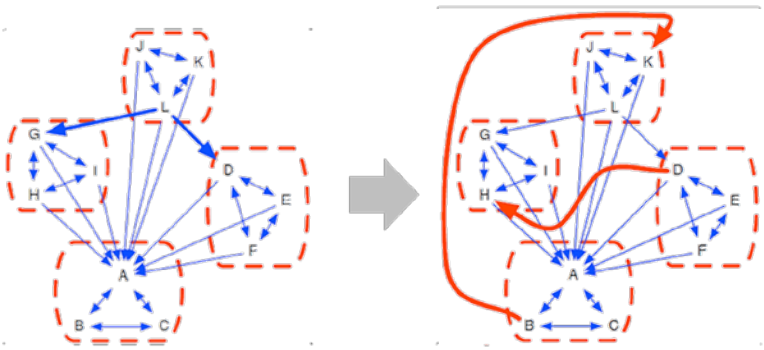


図 2　構造化パターン組合せと設計ルール違反発生時

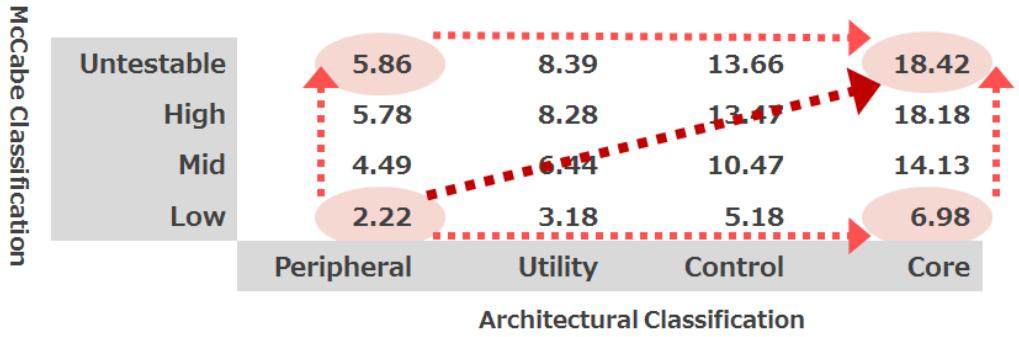
[1] Sturtevant, Daniel Joseph, “System design and the cost of architectural complexity”, Thesis (Ph. D.) Massachusetts Institute of Technology, Engineering Systems Division, 2013

(3)改善前の状態をもたらした原因（因果関係）
- ベースとなる理論（複雑度とバグの関係のモデル化）

**【Sturtevant による分析結果】**

Sturtevant の取組[1]では、図 2 に示したようなソフトウェアモジュール、コンポーネント間の依存関係に注目し、ソフトウェア構造複雑度[2]という概念を用いて構造複雑度がバグ密度を増加させるかについて、大規模商用ソフトウェアを対象にリポジトリマイニングを用いて分析を行った。分析結果は表 1 に示すとおりで、構造複雑度(Architectural Classification)が高い箇所(Core)は低い箇所(Peripheral)に比べて 3.1 倍もバグ密度が高くなることが示された。

表 1　構造複雑度による分類毎のバグ密度分布表




[2] MacCormack, Alan. et al. "The Architecture of Complex Systems: Do Core-Periphery Structures Dominate? ." Academy of Management Best Paper Proceedings, 2010

- OSS での分析・実証結果

**【筆者らの追試結果】**

Apache Ant™[3]を対象にバグ発生密度を調査した。Ant の対象バージョンは Version1.7、Version 1.8 とした。他の Version はバグトラッキング不可能であったため対象外とした。結果は表 2 と表 3 に示す通りで、構造複雑度(Architectural Classification)が高い箇所(Core)は低い箇所(Peripheral)に比べてバグ密度が高くなり、Sturtevant の結果と同様の傾向になることを確認した。

表 2　追試：Ant Ver.1.7 のバグ密度分布

| | Peripheral | Utility | Control | Core |
|---|---|---|---|---|
| Untestable | – | – | – | – |
| High | – | – | 36% | 67% |
| Mid | 13% | – | 17% | 38% |
| Low | 4% | 8% | 4% | 8% |

表 3　追試：Ant Ver.1.8 のバグ密度分布

| | Peripheral | Utility | Control | Core |
|---|---|---|---|---|
| Untestable | – | – | – | – |
| High | – | – | 26% | 20% |
| Mid | 13% | – | 22% | 13% |
| Low | 10% | 13% | 12% | 11% |

[3] Apache Software Foundation, Open source project. http://ant.apache.org/

また、Apache Ant™[3]を対象に構造複雑度と潜在的コードエラーの関係についても調査した。ソースコード静的解析(フロー解析)ツールを用いて潜在的コードエラーをソースコード中から検出し、以下の表 4 と表 5 に示す結果のとおり、構造複雑度が増すと潜在的コードエ

ラーも増加していることが確認できた。

表 4　追試：Ant Ver.1.7 のコードエラー検出密度分布

| | Peripheral | Utility | Control | Core |
|---|---|---|---|---|
| Untestable | – | – | – | – |
| High | – | – | 5.64 | **7.00** |
| Mid | 1.25 | – | 2.16 | 5.76 |
| Low | 0.50 | 0.12 | 0.64 | 0.90 |

表 5　追試：Ant Ver.1.8 のコードエラー検出密度分布

| | Peripheral | Utility | Control | Core |
|---|---|---|---|---|
| Untestable | – | – | – | – |
| High | – | – | **8.35** | **12.20** |
| Mid | 1.25 | – | 3.00 | 5.65 |
| Low | 0.72 | 0.38 | 0.64 | 1.20 |

(4)計画した変更内容

- 想定される用途

（1）リファクタリング

（2）コードレビュー効率化

設計・実装プロセスにおける欠陥除去の効率化を目的として、ソフトウェア開発プロセスにおける構造複雑度の応用として上記 2 施策の提案を行う。

(5)変更の実現方法

施策提案 1.リファクタリング

一般に用いられているソフトウェアリファクタリングのルールに、構造複雑度による監視ルールを追加することを提案する。ソフトウェア開発に進むにつれ、構造的な複雑さを起因とする欠陥が発生しやすい状況になるのを回避する。

施策提案 2.レビュー効率化

下記 2 点を行うことで、従来では残留バグとして残ってしまっていたようなバグの発見や、不足しがちな人的リソース(レビューア)の確保に役立てることを提案する。

・ソースコードレビュー前の静的解析時に構造複雑度メトリクスも監視し、構造的に複雑な箇所をリスクの高いコードと特定する。

・リスクの低いコードは開発者自身+静的解析でチェックし、リスクの高いコードは静的解析に加えて人手(開発者以外の目視)によるチェックを行う。

(6)変更後の状態や改善効果

着想段階のため該当せず。

(7)改善活動の妥当性確認

着想段階のため該当せず。

以上

1C2「全社の品質戦略に基づく SEPG の新たな役割と取組み」和良品文之丞(キヤノンソフトウェア)

〈タイトル〉: 全社の品質戦略に基づく SEPG の新たな役割と取組み

〈サブタイトル〉: 上流工程の品質向上を目的としたツール活用の推進

〈発表者〉
氏名（ふりがな）: 和良品文之丞（わらしな　ぶんのじょう）
所属：キヤノンソフトウェア（株）企画本部プロジェクト推進部

〈共同執筆者〉　　*★いれば記載する。複数指定可。*
氏名（ふりがな）: 石橋秀之（いしばし　ひでゆき）
所属：キヤノンソフトウェア（株）企画本部プロジェクト推進部

〈要旨〉
全社の品質改善活動の一環として、上流工程の品質向上を目的としたツール利用の促進を掲げ、現状分析、POC（概念検証）、インフラ構築、解析実施等を行った。
POC によるツールの有効性確認と現場のフィードバックを踏まえ、多様な利用形態に応じたサービスを SEPG の役割として提供することとした。
その結果、普及が進み開発現場が自ら解析する事例が増え、利用状況に応じて効率良くライセンスに投資ができるようになった。また、外部設計の検証にツールを活用して品質の良いソフトウェアを納品した事例も得られた。
全社の品質戦略に基づく SEPG の新たな役割と取組みとして当社の事例を紹介する。

〈キーワード〉
静的解析ツール、品質リスク、規模、複雑さ、
MTM（Metrics Tree Map）、DSM（Dependency Structure Matrix）
全社 SEPG、品質改善活動、概念検証（POC）

〈想定する聴衆〉
全社の品質改善推進者、全社のプロセス改善推進者、SEPG、SQA

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

2013 年、高品質のソフトウェア開発力を競争力と位置づけ、特に上流工程における品質の向上を目的とした全社的な品質改善活動を開始した。活動は大きく２つあり、品質メトリクスの導入と、ツール活用である。

品質メトリクスの導入は、事業課題に直結したメトリクスの収集と改善活動を、各事業本部が継続すると共に、メトリクスをプロセスに組み入れていくことを意図した。

一方で、ツール活用に関しては、事業本部横断の取組みとして、品質改善活動の事務局である全社 SEPG が推進することとした。

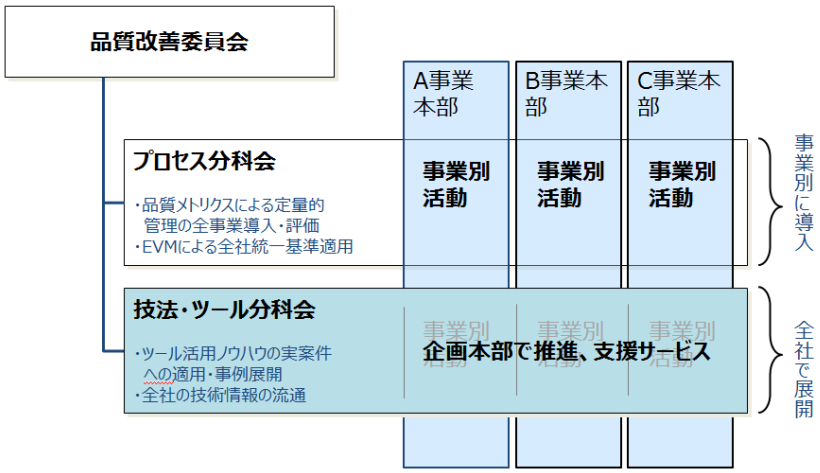
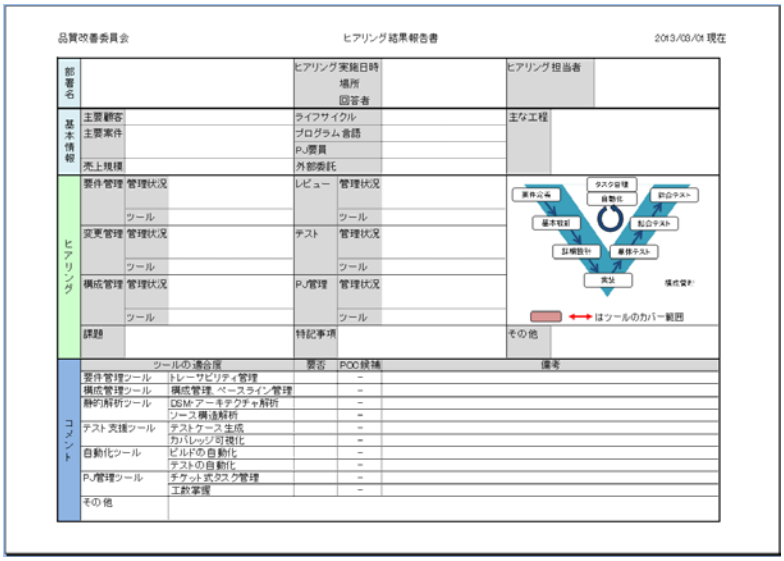



図１．品質改善委員会活動のカバー範囲

(2) 改善前の状態

品質向上に寄与するツールは多岐にわたり、そのカバー範囲も様々である。ツール導入に向けて、全事業本部にヒアリングを行い、ツール導入の状況を調査した。


品質改善委員会　ヒアリング結果報告書　2013/08/01 現在

| | | | | | |
|---|---|---|---|---|---|
| 部署名 | | | ヒアリング実施日時<br>場所<br>回答者 | | ヒアリング担当者 |
| 基本情報 | 主要顧客 | | ライフサイクル | | 主な工程 |
| | 主要案件 | | プログラム言語 | | |
| | | | PJ要員 | | |
| | 売上規模 | | 外部委託 | | |
| ヒアリング | 要件管理 | 管理状況 | レビュー | 管理状況 | |
| | | ツール | | ツール | |
| | 変更管理 | 管理状況 | テスト | 管理状況 | |
| | | ツール | | ツール | |
| | 構成管理 | 管理状況 | PJ管理 | 管理状況 | |
| | | ツール | | ツール | |
| | 課題 | | 特記事項 | | その他 |

| | ツールの適合度 | | 要否 | POC候補 | 備考 |
|---|---|---|---|---|---|
| コメント | 要件管理ツール | トレーサビリティ管理 | | － | |
| | 構成管理ツール | 構成管理、ベースライン管理 | | － | |
| | 静的解析ツール | DSM・アーキテクチャ解析 | | － | |
| | | ソース構造解析 | | － | |
| | テスト支援ツール | テストケース生成 | | － | |
| | | カバレッジ可視化 | | － | |
| | 自動化ツール | ビルドの自動化 | | － | |
| | | テストの自動化 | | － | |
| | PJ管理ツール | チケット式タスク管理 | | － | |
| | | 工数管理 | | － | |
| | その他 | | | | |

図２．調査票

その結果、下流工程ではテストの自動化や構成管理等でツールが活用されていたが、要件管理・影響分析、アーキテクチャ検証といった上流工程側の導入はあまり進んでいなかった。

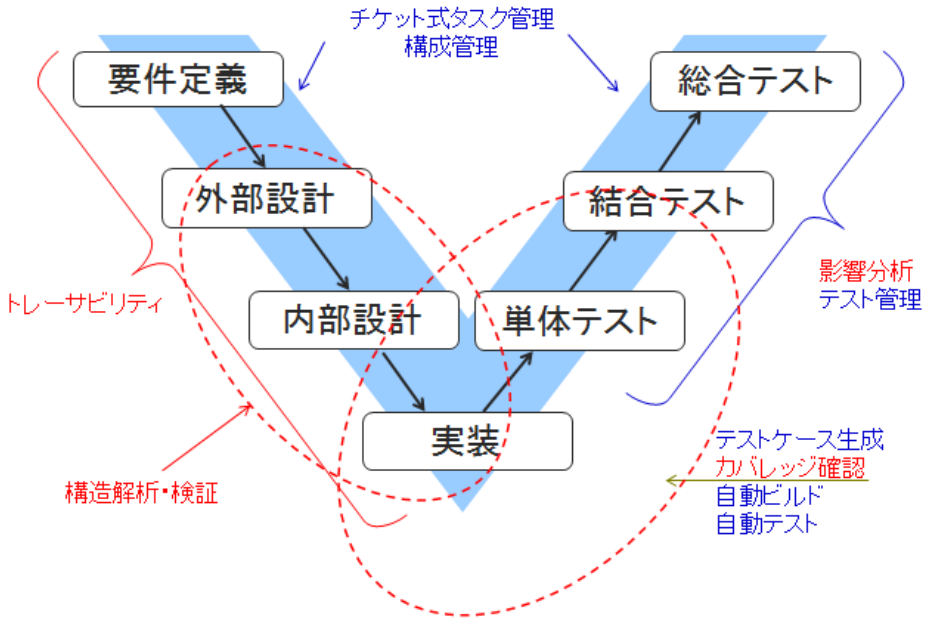



図３．ライフサイクルにおけるツールのカバー範囲

また、ツールの導入の状況も事業による差異が大きく、導入・活用のノウハウも個々の組織に閉じており、ライセンスの偏りや休眠といった状況も見受けられた。

(3)改善前の状態をもたらした原因（因果関係）

ツールの利用が下流工程側に偏りがちであった現状を分析するため、上流工程と下流工程の観点で、ツールの状況を考察した。

- 上流工程に比べて下流工程の方が、プロジェクトの作業成果物が揃っており、機械的に解析できる範囲が広い。
- 下流工程で用いる開発環境に付属する、もしくは依存するツールが用意されている。
- テストの自動化や処理の自動実行、チケットによるプロジェクト管理などは、有償・無償のツールが豊富である。
- トレーサビリティ管理、構造解析ツールは、比較的高価である。
- 有償ツールの試用期間が短く、期間内に使いこなして効果を検証することが難しい。
- 解析結果を分析し、解釈するための解析内容に対する知識が必要である。

ツールの利用が下流工程側に偏りがちであることの理由としては、開発者が日常的に使う開発環境に組み入れやすいことや、無償ツールが豊富で開発者の責任範囲と工夫によって導入しやすいためと考えた。

その一方で、上流工程のツールは、無償ツールが限られる上、有償ツールは高価で導入の負担や効果を事前検証することが難しく、導入の障壁が高いと考えた。現場での上流工程に対する問題意識は高く、ツールの共有や教育のニーズがあった。

(4)計画した変更内容

前述の状況を踏まえて、上流工程の品質向上に対する取組みとして、以下を計画した。

- 上流工程の品質向上に効果的なツールの調査。
- 開発現場の協力による POC（概念検証）の実施。
- ツールの活用を促進するための施策検討。

a. 上流工程の品質向上に効果的なツールの調査
上流工程の品質に直接影響を与える活動として、要件管理、変更管理、アーキテクチャ検証を選定した。その中でツールによる効果が期待できるものを、双方向のトレーサビリティ管理、影響分析、構造解析に絞り込んでツールの調査を行うこととした。

b. 開発現場の協力による POC（概念検証）の実施
開発現場のニーズに合うことだけでなく、現場が継続して使えるものか、という観点を POC の目的に組入れた。

c. ツールの活用を促進するための施策検討
ツールの特性や現場での利用形態に応じて、全社 SEPG の観点で、どのような施策によって活用が促進できるか、現在の全社 SEPG の役割にとらわれず、考案し、POC と合わせて試行することとした。

(5)変更の実現方法

ツールの調査は、すでに実施済みの現場へのヒアリング時の情報を参考にした。ヒアリング前に全社 SEPG が調査したツールのリストと、現場から利用のニーズから候補を絞り込んだ。結果として以下の 4 つのツールの POC を実施した。

- トレーサビリティ管理ツール
- ソースコードの静的解析ツール
- 構造解析ツール
- テスト管理ツール

上記の中では、テスト管理ツールが上流工程から外れているが、テスト工程から上流工程を含むライフサイクル全体に管理を広げられるか検証したいという現場からの強い要望を受けて候補とした。

開発現場の協力による POC（概念検証）の実施は、ツールの特性、導入コスト、現場のニーズを踏まえて、３つの異なるアプローチで実施した。

- 全社 SEPG が、ライセンスを導入した PC をセットアップし、PC ごと現場へ貸し出し、現場主体で実施（トレーサビリティ管理ツール）
- 全社 SEPG が、ライセンスを導入した PC をセットアップし、現場から情報提供を受けて全社 SEPG が主体で解析を実施（ソースコードの静的解析ツール、構造解析ツール）
- 全社 SEPG の予算でツールメーカーにコンサルを委託、現場主体で実施（テスト管理ツール）

POC の結果は、以下のとおりである。

a. トレーサビリティ管理ツール
ツールの特徴として、独自の DB を持たずタグ付けによってトレーサビリティを視覚的に表現できるため、広く適用可能と判断し、社内の貸出しサービスを開始することとした。

b. ソースコードの静的解析ツール、構造解析ツール
規模、複雑さ、依存関係を効率よく分析できる。解析結果は現場に事実として受け入れられたものの、その結果をどのように活用したら良いかわからない、というフィードバックがあり、課題が残った。

c. テスト管理ツール
ツールが目的としているとおりテスト工程の管理が中心であり、開発のライフサイクル全般に適用を広げることに無理があった。また類似の OSS に対する優位性も自社の開発規模では発揮できず、導入は見送りとなった。

POC の結果を踏まえて、トレーサビリティ管理ツール、ソースコードの静的解析ツール、構造解析ツールの導入を決定した。要件や設計と成果物のつながりの確保、作成したプログラムを解析することを通じた設計の妥当性確認、変更要求に対する影響分析、これらにツールを活用することを意図して、上流工程の改善に役立つと考えた。

ツールの活用を促進するための施策検討として、特に意識したのは、上流工程向けツールの導入障壁となった点の解消である。それを踏まえて以下の形態を考案した。

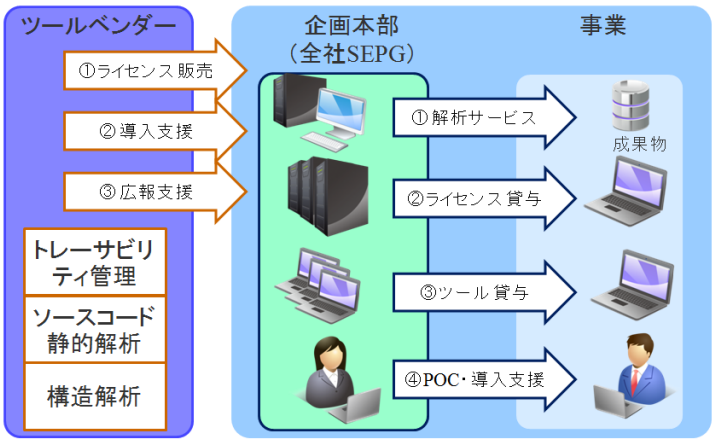


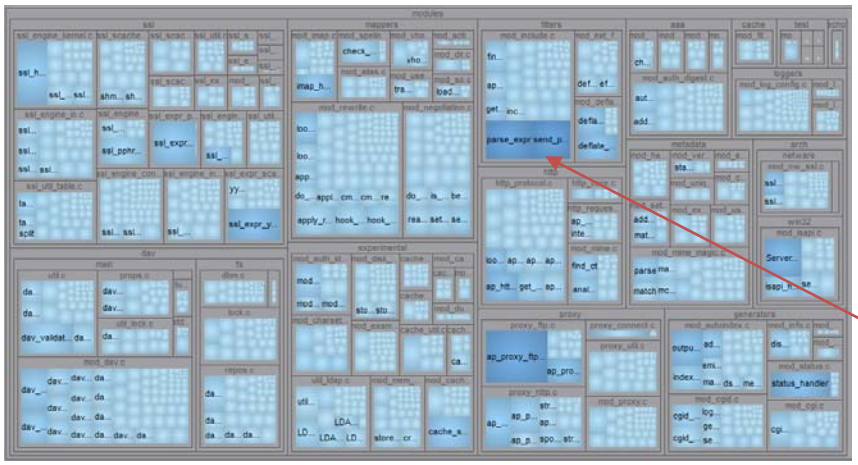

図４． 全社 SEPG によるツール関連サービス例

前述のアプローチの違いを反映して、①全社 SEPG による解析サービス提供、②ネットワーク経由によるライセンス共有、③ツールを導入した PC ごとの貸出し、④現場による POC やツール導入時の支援の４つの形態を検討した。

ソースコードの静的解析ツール、構造解析ツールに対しては、解析結果の特性を再確認し、より視覚に訴える報告項目を選定した上で、「品質リスク」の観点から、どのようなリスクがあり、どのようなアクションを起こすかを示すこととした。

報告する上で視覚的な表現として特に重視したものは、MTM（Metrics Tree Map）と、

DSM（Dependency Structure Matrix）の２つである。



箱の大きさ：規模
色の濃さ：複雑さ

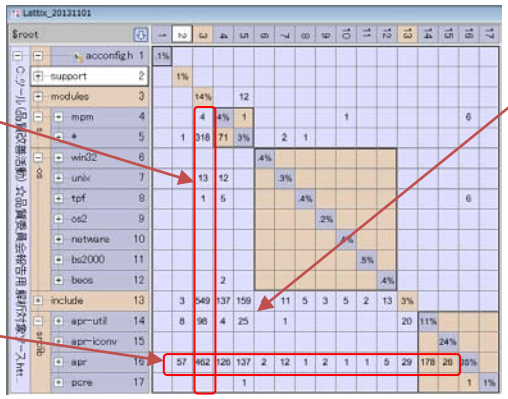
箱が大きくて色が濃いものが、規模が大きくて複雑

図５．MTM（Metrix Tree Map）*【Understand】*

Understand による MTM 画面例：ツールベンダに掲載可を確認済みです

縦に数値が並ぶ：
多くを参照する
→多くのモジュール
から影響を受ける

横に数値が並ぶ：
多くから参照される
→多くのモジュール
に影響を与える



交点の数字が相互の参照関係を示す

図６．DSM（Dependency Structure Matrix）*【Lattix】*

Lattix による MTM 画面例：ツールベンダに掲載可を確認済みです

規模と複雑さ、参照関係に着目し、着目した理由とアクションを以下に示す。

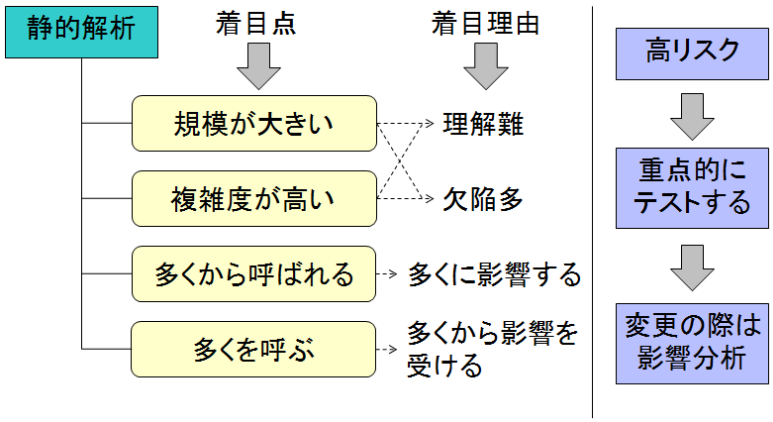


図７．品質リスクとアクション

(6)変更後の状態や改善効果

品質改善活動は、全社横断の委員会として 1 年間活動したが、2014 年も活動の継続が認められた。

トレーサビリティ管理ツールは、現場で実プロジェクトに適用され、厳しい品質要求に応えている。活用事例は委員会の場や技術的なイントラマガジンを通じてノウハウが紹介され、さらなる水平展開に取り組んでいる。IT 業界の展示会にも出展し、当社の開発力向上に寄与した。

ソースコードの静的解析ツール、構造解析ツールに関しては、プロジェクトの課題に基づいて、リファクタリングや外部委託の派生開発を行うプロジェクトに対して構造解析を提案し実施した。その結果、改造の前後を比較することでリファクタリングの妥当性確認や、品質リスクの変化を読み取る等の具体例を追加することができた。

ツール説明会の実施、インストールガイドの作成、解析サービスの提案と実施、現場に出向いての導入支援、グループウェアによる貸出し管理等の施策を進めたことがポイントで、現場での利用も増えている。

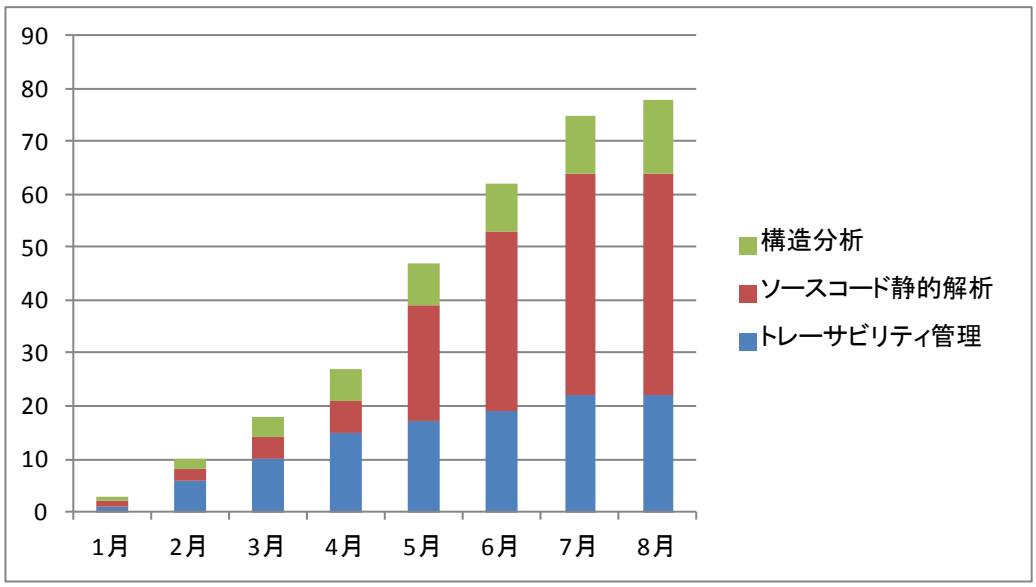



図８．ライセンス利用実績（累計、人）

(7)改善活動の妥当性確認

今回の取り組みで、ニーズはあるが高価であったり、導入の負荷が高かったり、利用頻度や使い勝手が不明だったりするツールに対する全社 SEPG としての新たな役割と取り組みを検証することができた。

現場での普及を進めるには、ツール活用による具体的な効果が実感できることと、利用に対するインフラやサポートが整っていることが必要である。現場のツールに対する

理解度とニーズ、利用頻度を踏まえて、様々なサービスを用意したことも利用を促進できた一因である。
　またライセンスを一元管理して共有の仕組みを作ることにより、利用状況を踏まえた無駄のない投資が可能となる。グループウェアによる予約システムの仕組みもあり、効率良く他部門で利用できている。

表１．ライセンス導入実績

| ツール | オンライン利用 | PC利用 |
|---|---|---|
| トレーサビリティ管理 | 1 | 2 |
| ソースコード静的解析 | 2 | 2 |
| 構造分析 | 0 | 2 |

　ここで、全社 SEPG がこれらのサービスを提供する意義は何か、いわゆる集中購買と比較を以下の表に示す。

表２．サービス提供者の差異

| 比較項目 | 全社SEPGによる管理 | 集中購買 |
|---|---|---|
| 目的 | 上流工程の品質向上 | コストダウン |
| 選定の最終責任 | 全社SEPG | 現場（事業部門） |
| ライセンス管理 | 資産管理＋共有管理 | （資産管理） |
| 費用負担 | 全社SEPG | 現場（事業部門） |
| 占有期間 | 短（貸出し予約制） | 長（減価償却） |

　目的の違いから、選定、ライセンス管理、費用負担などに違いがでている。上流工程の品質向上という目的に沿ったツールを全社 SEPG の責任で導入し、品質向上施策の一つとして活用を推進することが特徴であり、また推進上の重要な点である。

　さらに、現場のプロジェクトでトレーサビリティを活用したことによる高品質ソフトウェアの納品事例を報告する。

- 適用プロジェクト
  - ・中規模・6 ヶ月の開発案件
- ツール適用の範囲
  - ・外部設計と結合テスト（テストケース）のトレーサビリティに適用
  - ・カバレッジ機能を使い、テストの網羅性を確認
  - ・トレーサビリティ欠落部分の確認、テストケース・確認の追加

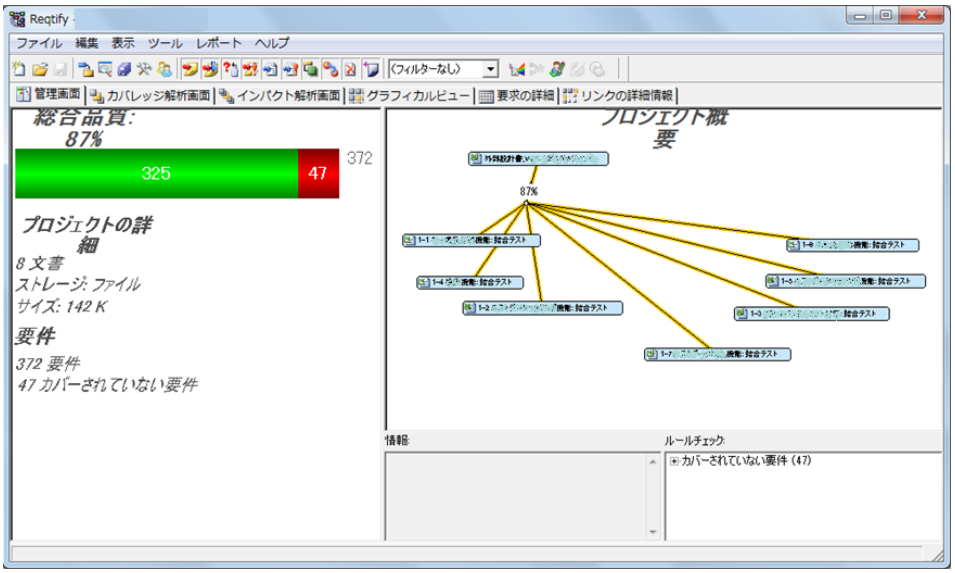



図９．トレーサビリティツール適用例

- 品質向上結果
  - ・後工程のオンサイトテスト及びシステムテストの障害 0 件
  - ・稼動後も不具合の問合せなし
  - ・高品質アプリケーションとして顧客の信頼向上

今後の活動としては、ツールの活用によって数値化されたパラメータをメトリクスとしてプロセスに組み入れ、工程移行の判定や妥当性確認の判断に用いることを計画している。組織的に推進することと、その成果をプロセスに反映することこそが、全社 SEPG がツール活用促進を担う意義である。

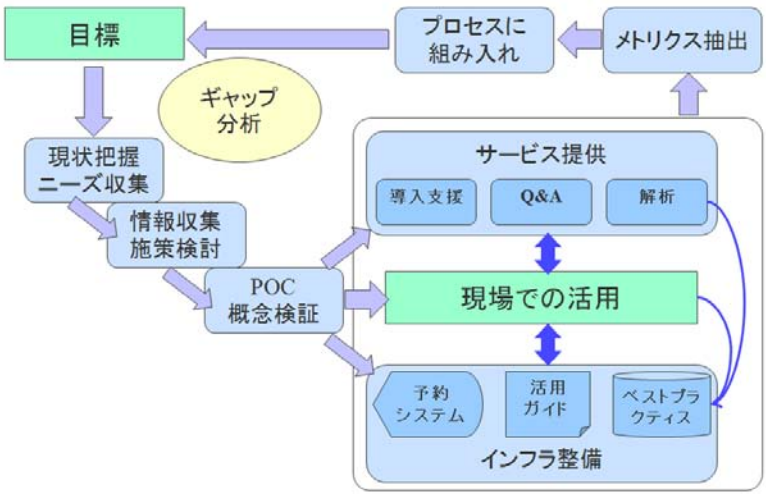



図１１．全社 SEPG が推進するツール活用のサイクル

以上

1C3「イテレーティブなプロセスと欠陥モデルによる要因分析法(Root Cause Analysis)の改善」永田敦(ソニー)

〈タイトル〉: イテレーティブなプロセスと欠陥モデルによる要因分析法(Root Cause Analysis)の改善

〈サブタイトル〉: アジャイル RCA の提案

〈発表者〉

氏名（ふりがな）: 永田　敦
所属: ソニー株式会社　PSG　品質保証部門

〈要旨〉

RCA(Root Cause Analysis)は、品質問題を解決していくための最も重要なプラクティスの一つである。弊社でも、なぜなぜ分析として行われてきて、ソフトウェアの障害の要因分析にも使われている。しかし、時間がかかり、数がこなせず、時間をかけた割に真の要因に届かず、効果的な対策が打てていない例も多い。本提案は、軽量にかつイテレーティブに行う RCA の手法で、これにより、より短い時間で、現場に負担をできるだけかけずに、効果的かつ実行性のある対策を打つことができた。また、従来は重たい単発の活動であったが、この方法により継続する改善活動になっていった。

〈キーワード〉

Root Cause Analysis, なぜなぜ分析、欠陥、欠陥エンジニアリング、失敗学、アジャイル、イテレーション、モデリング

〈想定する聴衆〉

ソフトウェア開発者(特にアジャイル開発をしているチームメンバー)、品質保証の方、SEPG 担当者

〈適用状況〉

■多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

今回の改善に取り組んだ主体はソフトウェア開発チームである。それらのチームはアジャイル開発手法であるScrumを取っている。ウォータフォール開発であろうが、アジャイル開発手法であろうが、開発における品質の問題を同様に持っている。ソフトウェア開発チーム品質の改善活動のためには、障害における欠陥やその真の原因が突き止められ、表現できていなければならない。真の要因を認識できず、取り除くべき欠陥が除かれていないままに、ただ、障害を起こさないように対処した場合、ピンポイントでは障害は収まるかもしれないが、品質は改善してはない。その欠陥は違う形で障害となって現れるかもしれない。設計者がその欠陥を埋め込む原因に気が付かなければ、同じようなミスを犯して再び欠陥を埋め込んでしまう。一方で、入りやすい欠陥が捉えられていないため、適切なテストが設計できない。テスト漏れを防ぐためテストケースは多くなり、テスト実行に時間をかけてしまう。しかし、テストの狙いが外れていれば市場に漏れてしまい、ある条件で顧客の目の前で障害を起こしてしまう。

根本分析法：RCA(Root Cause Analysis)は、このような品質問題を解決していくための最も重要なプラクティスの一つである。

(2)改善前の状態

RCAの課題は、その活動が時間がかかり根気のいる仕事であるということである。[1]要因分析のプロセスを考え、なぜなぜ分析の手法を取り入れてきたが、真の原因を出していくには時間がかかっていた。

一方、分析技術にはスキルが必要だが、その機会が少ないので技量が上がらない。

なぜなぜ分析では、複数回、被分析者に対し質問をしていくが、最初の質問が悪いと、分析が違った方向に行ってしまうことになる。さらに、障害を起こした担当者(被分析者)の行為を責める質問になってしまうことも多い。このようにRCAは現場にとって時間的にも精神的にも負担が大きい割に、効果を出すことが難しい。そのため、次第にやらなくなってしまう。

(3)改善前の状態をもたらした原因（因果関係）

活動の時間がかかる原因は、そのプロセスにある。

まず、準備段階で、被分析者が、障害の原因やメカニズムの説明と対策の提案のために資料を作る。障害の分析から、対策の妥当性までの調査検討も含まれる。開発行為とは違う性質の報告書のために、作成には時間がかかり、場合によって数日間、十数時間かかる場合もある。また、自分の間違いを曝け出し、説明をしなければならないという痛みも伴う。

次に、その調査報告をベースに、RCAのレビューを行う。数人のレビューアにより質問を繰り返して、真の要因に向けて分析を行っていく。1回のレビューに2時間を使っていた。

しかし、報告書も足りない部分が見つかり再調査が必要になり、2時間でも真の原因に至らない場合が多いので、複数回行うことが多い。

これにより、被分析者に対しては、8時間から十数時間と多くの負荷がかかる。

もともと、RCAもなぜなぜ分析も対象は“モノ”であった。欠陥自身がモノにあるので、それに対して客観的に分析することができる。たとえ、原因が人間の間違えであっても、その間違えを起こしても問題を起こさないように“モノ”に対して工夫をしていく、いわゆる”ポカよけ“で対処されている。一方で、ソフトウェアの欠陥の場合、分析の対象は

人間になる。人間による間違いそのものに対して分析をしていかなければならない。なぜなぜ分析の”なぜ”という質問は、モノに対しては非常に強力である。一方で、そのなぜを人に使ってしまうと、その人の判断、つまり価値観にまで触れてしまうので、場合によっては木津つけてしまうことがある。人間はその恐れを感じると防御しようとするので、事実として起きたことなども含めて情報を出さなくなってしまう。いわゆる、責められ攻撃されていると思ってしまう状態である。そのような状態で、なぜを繰り返しても、答えることができなくなる。そのため、本当のメカニズムを認知することができなくなり、真の要因にたどり着かなくなってしまう。また、何とかそこから逃れるために、他へ問題の転嫁を行う。つまり言い訳をするようになる。そうなると、真の原因とはかけ離れた違う方向に分析の議論が飛んでしまうことになる。それに“なぜ”と問うても時間の無駄になる。そのように、真の原因に到達しなければ、効果的な対策を打つことができない。レビューが十分に行う、などのｘｘを十分にやるという漠然とした対策になり、結局効果が上がらないことが多い。また、一度このような責められる思いを経験すると、ひどいときにはトラウマになり、やりたがらなくなる。なぜなぜ分析の解説書では、人を責めないようにすることが書かれているものがあるが、言葉自体に力があり、責める意思がなくても受け取るほうは責められるように感じることがある。

真の原因に到達したいもう一つの理由は、分析のスキルが上がらないことである。分析技術は、レビュー技術の一つで、ファシリテーションスキルが非常に重要になる。このスキルを上げるには訓練と経験が必要である。そのためには、場数が必要である。しかし、時間がかかり、つらい思いや傷つけられるリスクがあるものは、あまりやらなくなる。ほとんどの場合、単発でおわり、チームが継続して RCA をするということはない。そのため、ファシリテーションのスキルは上がらず、また再びやる機会があってもあまり効果を上げられないという悪循環に入ってしまう。これにより、RCA のモチベーションが落ちてしまうどころか、RCA という手法自身が忘れ去られてしまう。事実、RCA という言葉自身を知っている人は多くない。

しかし、バグが見つかり、トラブルが発生されれば、なぜそれが起きたのか、それを防ぐことができないのか、改善していきたいという思いが繰り返しされてきた。何とか現場で、RCA を定着して行うことができないかというのが本提案を生む課題だった。

(4)計画した変更内容

まず、1 回の分析時間を短時間に規定する（タイムボックス）そして、それを毎日行って習慣化していくことで、負担を軽くする。短くすることは、負荷を軽くするだけではなく、短時間ならば集中するという効果もある。

分析行為は、二つのフェーズで考えられる。最初のフェーズは、インシデント（Incident）分析である。なぜ起こったかの前に、何が起こったかが正確に網羅的に分析され、表されて共有されなければならない。

次のフェーズは、質問のフェーズで、分析の主体である。質問の質が分析の質を決める。（３）でも表したように、なぜなぜ分析のなぜ、Why は大変強力な質問である。しかし、強力だからこそ注意して取り扱わなければコントロールができない。強力な質問をするのではなく、適切な質問をすることが肝要なのである。つまり、適材適所でいろいろな質問（What, When, Where,How）をして行くことにした。

それぞれのフェーズに対して 15 分ないし 30 分程度のタイムボックスを設け、それを毎日行うようにした。タイムボックス内で出た答えに対し、分析者は、RCA の目的やそれが真の原因に近づいているのかを調べて次の質問を考えて行く。それを図１のようなプロセスで行うようにした。これをイテレーティブ RCA と呼んだ。

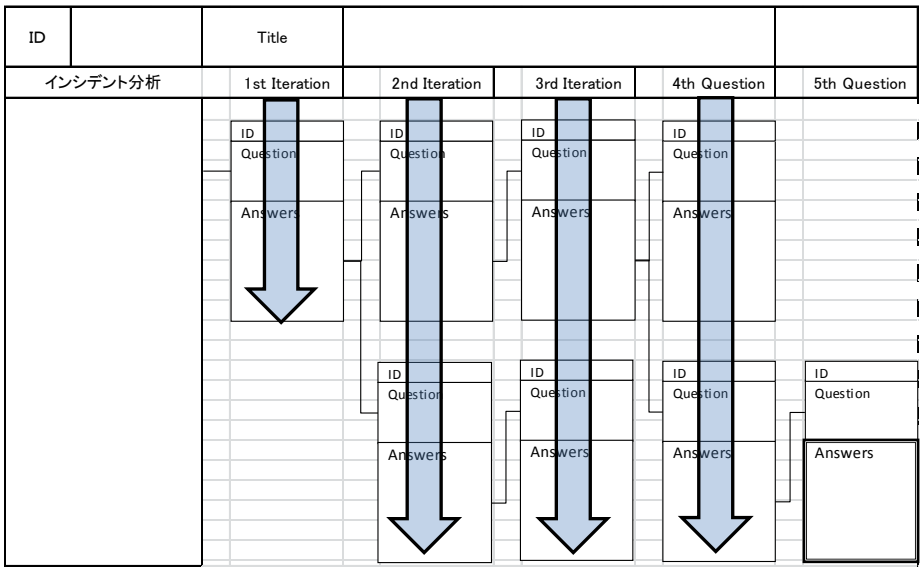



図１　イテレーティブ RCA テンプレート

図１では、イテレーションごとに矢印の方向で分析を進めていった。イテレーションのたびに、質問を考えて表現する。

(5)変更の実現方法

(4)の方法では、準備もイテレーションの範囲内にスコープを限定したので、被分析者の負担も３０分から長くて１時間以内と負担が減っていった。また、質問も明確に表現していくので、質問の妥当性もレビューすることもできた。

しかし、適切な質問が出せない場合もしばしば起こった。質問で明らかになる事柄は、すべて欠陥ではなく、欠陥に関連する要素が多く出てくる。たとえば、それ自身は欠陥ではないが､その欠陥を引き起こす要素だったり､欠陥の影響を加速する要素だったりする。そのような欠陥と関連する要素が関係を持ってメカニズムができていると仮定すると、それはモデルと考えられる。そのモデルが整理されて表現できれば、それを確かめたり、そこから考えられる推定により、適切な質問を考えることができる。違っていることがわかれば、そのモデルを変更していけばよい。このモデルとして、細川 宣啓氏らが作っている、Project Fabre[1]が提案している欠陥モデルを使うことにした。図 2 参照

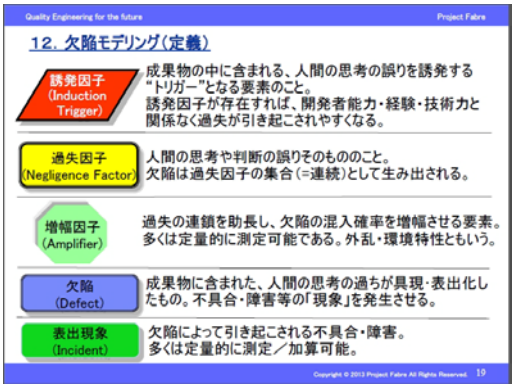



図 2 欠陥モデリング定義 [1]

欠陥モデルをイレテーティブ RCA に取り入れるために、図 3 のようの様なプロセスを考えた。

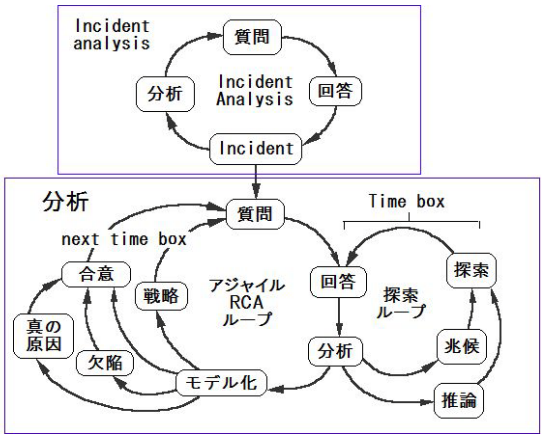



図 3　アジャイル RCA プロセス

これをアジャイル RCA プロセスと呼ぶ。アジャイル RCA は、次の 3 つのループを持っている。

1. Incident analysis
2. 探索ループ
3. アジャイル RCA ループ

まず、最初のタイムボックスで Incident Analysis を行い、障害で何が起こったかを分析し共有する。参加者は、被分析者、レビューアおよびモデレータである。ループ内で、不明確な点や事実関係を確認するために時間内で質問を繰り返す。

次に、アジャイル RCA ループに入る、アジャイル RCA ループは、タイムボックスで回る探索ループを持つ 2 重ループ構造になっている。

次のタイムボックスまでに、モデレータは、Incident Analysis の結果をもとに適切な質問を用意する。

次のタイムボックスでは、探索ループに入り、時間まで被分析者の回答を基に探索ループで質問を繰り返し､回答の理解のための確認や､背景や裏付ける事実を引き出していく。また、その繰り返しの中から想起される推論や見えてくる兆候に基づいてより深堀を行う質問を行っていく。

タイムボックスが終わった後、モデレータはアジャイル RCA のメインループに戻り、探索ループで得られた回答を分析して欠陥をモデル化し、戦略をもって次の質問を策定していく。

モデルの作成により、欠陥や、その欠陥に影響を及ぼす要素を抽象化して関係づけることにより、欠陥が問題を引き起こすメカニズムや、欠陥を作り出すメカニズムを表現していく。それにより、欠陥の特定や、真の原因を表出していく。

次のイテレーションが始まるとき、まず、モデレータは作られた欠陥モデルを被分析者およびレビューアに説明し、そのモデルの合意をとる。モデレータの解釈が違えば、直ちに指摘されるので､速やかに修正する｡モデルが正しければ数分で共有することができる。

欠陥モデルの合意の後､用意された質問を使って再び探索ループに入っていく｡この時、分析の場は欠陥モデルによって、より深いレベルの分析に入っていくことができる。

アジャイル RCA ループから出る基準は、参加者が真の欠陥と真の原因となる過失要因を

認識したときであるが、大体 3 回か 4 回で終わることが多い。

事例：一つの事例として、ある機能がハングして動かなくなった障害を分析した。そこで最終的に出来上がったモデルの事例を図 4 にしめす。


図 4 事例：ある障害に対する欠陥モデルによる分析事例

この例をみると、二つの大きな塊が見える。この障害は、if の条件式を誤って変えてしまったことと、データベースのアクセスにレーシングが起きて Null アクセスをした欠陥がもとになっている。しかしそれぞれの欠陥の背景には、思い込みや慣れた言語の書式で書いてしまう心理があったり、モジュールを流用したときの影響度のレビューができていなかったり、動いているものだから大丈夫だろうという思い込みと納期のプレッシャーから安易に他のモジュールを流用するなどの複雑な要素の関係がわかる。ここまで見える化できれば、チーム自らどの部分が直すべき真の原因に気づくことができる。どこを真の原因にするかは、その改善を実際に行う被分析者に考えさせる。もちろんモデレータはその妥当性を検討はするが、たとえわずかな改善でも、自分で気づき、自ら治す意思をコミットさせて、小さな成功に結びつかせたほうが良いと考えている。

(6)変更後の状態や改善効果

この手法を実施したことによる一番の変化は、チームの RCA の対応と考え方である。対応としては、以前は単発でしかできなかった RCA を継続的に行うようにできたことである。

今まで 3 チームの Scrum を採用しているアジャイル開発のチームがアジャイル RCA を行った。組織変更の都合で 2 チームは今では行っていないが、それぞれ 6 か月間、１か月間継続した。もう一つのチームも 6 か月継続しており現在も行っている。

RCA は自分たちの活動に対して有効なものとして受け入れている。そして、開発活動の一つとして、その時間になると集まるように習慣化していった。

期待効果は、このように継続していくことだが、達成された。もう一つのスキル向上であるが、最初モデレータのスキル向上だけを考えていたが、RCA に参加するチームメンバーの分析スキルも上がっていった。よって、最初のうちは欠陥は分析フェーズでモデレータがガイドして出していたが、最近ではインシデント分析で欠陥を出せるようになってきている。

3 チームのうち長く続いている 2 チームについてのデータを表１に示す。

表１　アジャイル RCA の結果

| | 扱った課題数 | イテレーションの平均値 | 欠陥数の平均値 | 抽出したファクタ数の平均値 | 課題ごとにかかった時間の平均値（分） |
|---|---|---|---|---|---|
| チーム A | 6 | 4 | 2 | 15 | 103 |
| チーム B | 7 | 4 | 4 | 21 | 133 |

これを合わすと、6 か月のうちで 13 の課題を継続的にこなして、欠陥を抽出している。

これを見ても、アジャイル RCA の効果には再現性があることがわかる。真の原因は、抽出したファクタ、特に過失因子に含まれる。このように、欠陥の周りに多くのファクタを出して問題を分析して表現していることがわかる。このように、分析の質でも確実に分析結果を出している。また、かかっている時間も平均 4 回のイテレーション（インシデント分析を除く）で終わっており、以前のやり方では最低で 8 時間かかることに比べれば、少なくとも 4 分の 1 以下は改善している。なお、従来のデータは、今まで RCA を行ってきた別のチームの事例をつかっている。

(7)改善活動の妥当性確認

適応したチームは、どれもアジャイル開発のチームで、無駄や余計な負荷、ましてや押しつけの改善や人を責めたりすることは極端に嫌う。そのチームに、1 日 1 回、朝会の後や夕食後などに 15 分から 30 分をもらい実施した。結果として長期間継続して行えることができたのは、この方法が彼らの開発に役に立つ情報や気づき、学びを与えるものと認知されているからである。ソフトウェアの分析では人をせめてしまいがちであるが、この手法では、その分析対象として欠陥モデルが人の代わりになっている。これにより、レビューアも被分析者も客観的に欠陥のメカニズムを分析することができる。つまり、この方法では、無意識に人を責めるようなことにはなりにくい。

欠陥モデルを作ることにより、対象となる障害の真の要因の方向が戦略的に推測することができる。それに対して探索的に質問をしていって裏付けをとっていくことで、欠陥モデルを確定することができると同時に、参加者もそのモデルを共有する。この欠陥モデルでは、積極的に事実も紐づけて表現している。これは本来のモデル化とは反するかもしれないが、事実と紐づいていないと抽象化した考えの空中戦となりがちになり、事実と乖離してしまい、さらに深い真実を引き出すことができない。

この活動で皆が思うのは、もし自分が同じ条件に放り込まれたら、おなじ過ちを犯してしまうだろうという感覚である。誤りはある意味で起こるべくして起きたのであり、だから、それを未然に防ぐことは難しいことに気づく。ここを出発点として皆で知恵を出し合って対策を考えていかなければ、効果的な手は打てない。だから、誰もが被分析者に対し

ては傾聴を行い、つまり respect(尊敬)の態度で質問していくのである。いつも、RCA を行うと、その原因の奥深さにうならざるを得ない。ここに、何とかしてそれを直していかなければならないという改善の心が芽生えてくる。その改善の心が、継続した RCA を実現していると考える。

費用対効果：2 時間程度で一つの問題を解決できるとすれば費用対効果は高い。だから継続している。

残存課題：欠陥モデルの作成はモデレータ（事実上筆者）が行っている。まだ、欠陥モデルの表現方法は開発途上であり、研究しながら行っているので作成には時間がかかっている。1 回あたり 30 分から 1 時間かかっているが、それを早くしなければならない。

また、アジャイル RCA を行っていると欠陥やそれに伴う因子の関係や意味が同じようなパターンがあることがわかっている。まだ、パターンの表現方法が開発できていないが、それができれば、より分析とモデルの作成が早くなることが期待できる。

以上

[1] 細川 宣啓, 西 康晴, 嬉野 綾, 野中 誠, 原 佑貴子, Project Fabre, 過失に着目した欠陥のモデリング, バグ分析はなぜうまくいかないのか？, ソフトウェアテストシンポジウム 2013 東京、2013
[2] 小倉仁志、「なぜなぜ分析徹底活用術-「なぜ？」から始まる職場
の改善」、JIPM ソリューション (1997)
[3] 早川勲 他、ソフトウェア品質シンポジウム 2008、「ソフトウェア開
発へのなぜなぜ 5 回の適用」、pp185-194
[4] 小倉仁志、「なぜなぜ分析を使いこなそう！なぜなぜ分析徹底攻略
ドリル」、JIPM ソリューション (2002)
[5] Ram Chillarege., Orthogonal defect classification-a concept for in-process measurements, Software Engineering, IEEE Transactions on (Volume:18 , Issue: 11 ), http://www.chillarege.com/articles/odc-concept, 1992.
[6] Eric E.Vogt, Juanita Brown, and David Isaacs, THE ART OF POWERFUL QUESTIONS: Catalyzing Insight, Innovation, and Action, Whole Systems Associates, CA, USA, 2003
[7] Tom Gilb, Fundamental Principles of Evolutionary Project Management, INCOSE, 2005
[8] Andersen, B., Fagerhaug, T.: Root Cause Analysis: Simplified Tools and Techniques. ASQ Quality Press (2006)
[9] Duke Okes, Root Cause Analysis, The Core of Problem Solving and Corrective Action, ASQ Quality Press, Wisconsin, 2009
[10] Tom Gilb, Agile Specification Quality Control: Shifting emphasis from cleanup to sampling defects, INCOSE, 2005

2A1「ソフトウェアプロダクトラインにおけるコア資産評価の仕組み確立」原田真太郎(オムロン ソフトウェア)

〈タイトル〉: ソフトウェアプロダクトラインにおけるコア資産評価の仕組み確立

〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 原田 真太郎（はらだ しんたろう）
所属: オムロン ソフトウェア株式会社
〈共同執筆者〉
氏名（ふりがな）: 筒井 賢 （つつい けん）
所属: オムロン ソフトウェア株式会社

氏名（ふりがな）: 赤松 康至（あかまつ やすゆき）
所属：オムロン株式会社

〈要旨〉
ソフトウェアプロダクトライン開発における、目的に応じた３つの評価手法を組み合わせた、コア資産の確立と維持フレーム構築の事例

〈キーワード〉
ソフトウェアプロダクトライン､アーキテクチャ評価､ＣＩ（継続的インテグレーション）、スクラム、反復型開発

〈想定する聴衆〉
ソフトウェアプロダクトラインを導入、または、導入を検討しているエンジニア、品質保証、ＳＥＰＧの方

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（ ）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（ ）
□自プロジェクトのみ

〈発表内容〉

(1)背景

本件が対象とする主軸製品のライフサイクルの多くは、１０年を超える。

現行製品群に搭載されているソフトウェア基本構造（以降、ソフトＰＦ）は、作成されてから長期にわたる改修により、構造が複雑化・規模が肥大化しており、新たな機能追加時の影響範囲調査、実装、テストに多くの工数が必要となっていた。結果、顧客ニーズに応えられる競争力のあるＱＣＤを確保することが困難な状況であった。

そこで、対象製品群のおける開発効率を３０%向上することをゴールに設定し、その実現のためにソフトウェアプロダクトライン（以降、ＳＰＬ）開発を導入した。ＳＰＬの考えを参考にし、“良いソフトウェア構造を考え（構想）”、“計画通りにコア資産を構築し（実現）”、そして“そのコア資産を維持していく（維持）”という取り組みを具体的に検討・計画し、このビジネスゴールの達成に向けて改善活動を推進した。

本件は、この３つのステップ（構想、実現、維持）すべての活動で必要不可欠である、“ソフトウェア資産を診断し、ビジネスゴールの達成の確からしさ、構想と実態との乖離を分析・是正する”のための「コア資産評価」の仕組み構築についての事例である。

(2)改善前の状態

複数製品にて長期にわたって利用可能なソフトＰＦを構築するためには、個別製品としての最適では無く、製品群全体および将来を見越した全体最適を考える必要がある。

しかし、弊社の標準プロセスでは、個別製品としての機能性・信頼性・使用性については、主にテスト工程にて検証しているが、製品群としての中長期にわたる改修のしやすさ（保守性・移植性）に関しては、主に工程毎のレビューで検証しており、属人的な要素が多く、十分に検証されていないケースが多かった。

また、大半の開発者は、個別製品開発の成功に主眼があり、製品群として、ライフサイクル全体を通しての品質（ソフトＰＦの良し悪し）には関心がまわっていない状態であった。

結果として、ソフトＰＦ再構築を行っても、数年後には現行ソフトＰＦと同様の状態になるリスクが高かった。

(3)改善前の状態をもたらした原因（因果関係）

現行のソフトＰＦを分析したところ、複雑度やコード規模、依存関係など、設計当初の方針に従っていない箇所が多数存在した。開発者にヒアリングしたところ、この複雑なソフト構造は、ソフト改修時の調査・修正・検証に多大な時間と費用を要する大きな要因となっていることが判明した。

この状態は（２）で述べたとおり、以下が主な原因となっていると考えられる

① 個別製品開発に主眼が当たりすぎている

② 資産を維持する観点での良し悪しの見える化の仕組みの欠如

(4)計画した変更内容

ソフトＰＦを構想通りに実現し維持していくためには、ソフトＰＦの良し悪しを見える化することで開発者に気づきを与えて是正を促す仕組みが必須と考え、以下を仕組みを開発プロセスに導入した。

・ 変更シナリオを用いたアーキテクチャ評価によるリスクの洗い出しと対策検討

・ 分析者による詳細なソースコード分析によるリスク・課題とその修正方法の提示
・ 継続的インテグレーション（以降、ＣＩ）を用いた、ソフトウェアメトリクスの計測

(5)変更の実現方法

開発に関わるメンバ全員が製品群、中長期の視点を常に持ち続けることが、ソフトＰＦの実現と維持のために必要不可欠である。よって、目的・対象メンバに応じて３つのサイクルでソフトウェアを診断し、当初狙い通りの構造が作られているか(維持されているか)を診断し、気づきを与え改善を促した。

| 目的 | 頻度 | 対象 | 方法 | 確認項目 |
|---|---|---|---|---|
| 将来の変更への対応力の確認<br>ビジネスゴール達成の確からしさ確認 | 工程毎 | アーキテクト<br>プロダクトオーナー | レビュー（変更シナリオ分析） | ・ 変更シナリオにより影響を受けるコンポーネント<br>・ 対応工数の概算 |
| ソフト構造の良し悪しの確認 | 3週間 | 設計者 | 分析者による解析 | ・ 複雑なロジックの詳細<br>・ 依存関係 |
| コーディングレベルの良し悪しの確認 | 毎日 | 実装者 | ＣＩ | ・ ＬＯＣ<br>・ 複雑度<br>・ コーディング規約違反<br>・ 静的解析指摘<br>・ コード内のコメント |

【レビュー（変更シナリオ分析)】

大きなマイルストーンのタイミング（アーキテクチャ設計完了時、ソフトＰＦの実装完了時など）にて、シナリオベースのアーキテクチャ評価（ＳＥＩ作成のＳＡＡＭをベースに、弊社でカスタマイズしたもの）を実施した。

関係者により抽出した将来における変更可能性を変更シナリオとして抽出し、アーキテクトを中心としたメンバにて、変更シナリオ適用時のコア資産のリスクを分析し、その対策を事前に検討することで、高い変更容易性を実現した。

【コード分析】

本プロジェクトは、反復型開発手法の一つであるスクラムをベースにプロジェクト運営をしており、反復（スプリント）期間は３週間に設定している。

３週間毎のスプリントにおける振り返りミーティングにて、ソースコード分析結果およびリスク・課題への対応案を開発者にフィードバックし、改善を促した。具体的には、ツールを用いて、コンポーネント間の依存関係分析および複雑なコードの詳細な分析を行った。

【ＣＩ】

ＣＩツール（Ｊｅｎｋｉｎｓ）を導入し、静的解析、単体テスト、コーディング規約チェックなどを毎日（統合ビルドなどはコミット毎）に自動で実施するようにした。

違反が発生した場合は、自動で開発者にメールが送信されるように設定されており、開発者への迅速なフィードバックによりロスコストを削減した。

現在、対象製品群においては、上記仕組みにより、変更が容易で競争力のあるソフトＰＦを実現し、維持フェーズに入っている。
また、上記仕組みは、他プロジェクトでの活用を見越して評価フレームとして手順化している。

(6)変更後の状態や改善効果

現行ソフトウェアＰＦと比較して、コア資産の維持に関するメトリクス値が大幅に改善した。(詳細な分析は現在実施中)
また、変更シナリオ評価を用いた検証の結果、ＱＣＤに関しても大幅な改善の見込みが得られ、ビジネスゴール達成の確からしさを確認することができた。

(7)改善活動の妥当性確認

（６）で述べたとおり、ゴール設定した開発効率向上は本取組により実現の見込みである。また、これまで見える化できていなかった“製品群・中長期視点”でのソフトＰＦの良し悪し見える化し、開発者全員が意識できる状態になっている。

さらに、副次的な効果として開発者の育成があげられる。
製品群としての視点を持った変更しやすいソフトウェアを常に意識させることで、開発者の意識レベルが向上し、設計・実装スキルが向上している。(プロジェクトの序盤と終盤では、コーディング規約などの違反頻度が減少)

今後は、本仕組みをＳＰＬにおける構想・実現・維持の標準プロセスとして、組織に定着させることが課題である。

2A2「“カイゼン”の輪を効果的に広げるマフィアオファー」八木将計(日立製作所)

〈タイトル〉: “カイゼン”の輪を効果的に広げるマフィアオファー

〈サブタイトル〉: XDDP 提案における事例

〈発表者〉
氏名（ふりがな）: 八木 将計(やぎ まさかず)
所属：株式会社 日立製作所
〈共同執筆者〉
氏名（ふりがな）: 小川 秀人(おがわ ひでと)
所属：株式会社 日立製作所

〈要旨〉
ソフトウェアプロセス改善の手法や活動を開発現場へ広げていくには，改善推進者のみによる推進では限界があるため，開発現場での協力者を増やしていくことが重要である．本報告では，そのための手法として，「やらされ感がなく」「他者へ簡潔に説明できる」マフィアオファーシートを用いたマフィアオファー型プロセス改善を提案する．また，その事例として，派生開発プロセスである XDDP の提案にマフィアオファーを用いた結果を示す．

〈キーワード〉
マフィアオファー，プロセス改善，ソリューション提案，質問，キーワード，やらされ感，XDDP，TOC 思考プロセス

〈想定する聴衆〉
SEPG 関係者，改善活動が広まらずに困っている方

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（ ）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（ ）
□自プロジェクトのみ

〈発表内容〉

(1)背景

ソフトウェアプロセス改善を推進する立場の個人・組織(SEPG など)が開発現場に改善手法や活動を「広める」場合.

(2)改善前の状態

改善手法や活動を広めるには，改善推進側のみが推進するだけでは限界があるため，開発現場側に協力者を作っていくことが重要になるが，そのような協力者を増していくことは難しいことが多い.

(3)改善前の状態をもたらした原因（因果関係）

開発現場における協力者を増す場合，以下のような課題が考えられる.

- 改善手法や活動に「やらされ感」があるとそもそも協力者にならない
- 協力者が改善手法や活動を他者に説明できないと広がっていかない

一方，一般的にソフトウェア開発のプロセス改善を行う場合，以下の 2 つのパターンが考えられる.

- CMMI(Capability Maturity Model Integration)などに代表される「モデルベース型のプロセス改善」
- SaPID(Systems analysis／Systems approach based Process Improvement methoD)[1]の問題構造図や TOC(Theory of Constraints)思考プロセス[2][3]などに代表される「問題解決型のプロセス改善」

これらは，それぞれ以下のような特徴がある.

- モデルベース型プロセス改善
  - ✓ 利点:多方面で実績のある効果的な手法が纏められており，プロセスモデルに基づき，体系的に改善を進めることができる.
  - ✓ 課題:適合を目的化して形式的な対応をとりがちになる．また，客観性を重視すると，現場にとって「やらされ感」が出てしまう.
- 問題解決型プロセス改善
  - ✓ 利点:当事者が自己の問題分析から入るため,「やらされ感」が生じにくい.
  - ✓ 課題:問題分析のスキルが必要であり，時間もかかる．また，問題分析に参加した当事者の満足度は高くなるが，問題構造が複雑になり，他者への説明が難しくなりやすい.

また，問題解決型とモデルベース型プロセス改善の両者の特徴を活かした手法として SPINA3CH(Software Process Improvement with Navigation, Awareness, Analysis and Autonomy for CHallenge)[4]がある．ただし，問題分析にスキルと時間がかかる点，他者への説明の困難さについては，問題解決型と同様の課題を内包している.

よって，従来のプロセス改善のアプローチは，開発現場での協力者を増す観点では，必ずしも効果的とは言えない.

(4)計画した変更内容

改善手法や活動を広めるためには,開発現場での協力者を増していくことが重要であり,そのためには「やらされ感がなく」「他者へ簡潔に説明できる」必要がある．本報告では,その課題に対するアプローチとして，改善推進者(および協力者)が開発現場に対してソリューションを導入する「マフィアオファー型プロセス改善」を提案する．

「マフィアオファー」とは，TOC における営業・提案手法であり，別名:URO(Un Refusable Offer:断わることのできない魅力的な提案)と呼ばれる[5]．人は，ソリューション導入などの変化への抵抗は 6 階層(表 1)になるといわれている．マフィアオファーは，この 6 階層の順番に抵抗を解消していく方法であり，このマフィアオファーを体系的に纏めたツールが「マフィアオファーシート」である(図 1，図 2)[6][7]．マフィアオファーシートは，営業(主に法人営業)において，抵抗の 6 階層に基づく合意形成プロセスの順番に合意していくための営業戦略の話法を纏めた A3 一枚のシートである．

このマフィアオファーシートでは，提案したいモノ(製品・技術)の特徴から，それらが解決できる顧客の問題構造を導き，その問題構造を「質問」という形にする．つまり，直接，顧客の問題を分析するのではなく，(提案したいモノが解決できる)問題の存在を質問し，顧客自身に問題構造について考えさせることで，抵抗の第 1 階層「問題の存在を認めない」を解消，やらされ感をなくす．本当に問題が存在するかの検証は必要になるが，既にソリューションが存在している状況では，問題分析から入るよりも時間がかからない．また，問題構造を提案者側から提示するため，顧客側には問題分析のスキルを必要としない．

また，マフィアオファーシートは，営業のためのツールであるため，A3 一枚に纏められており．短い時間で簡潔に説明することができる．

表 1　変化に対する抵抗の 6 階層

| 階層 | 抵抗の 6 階層 |
|---|---|
| 1 | 問題の存在を認めない |
| 2 | 解決策の方向性に合意しない |
| 3 | 解決策が問題を解決できると思わない |
| 4 | 解決策を実行すると副作用が生じる |
| 5 | 解決策の実行を妨げる障害がある |
| 6 | 未知のことへの恐怖感がある |

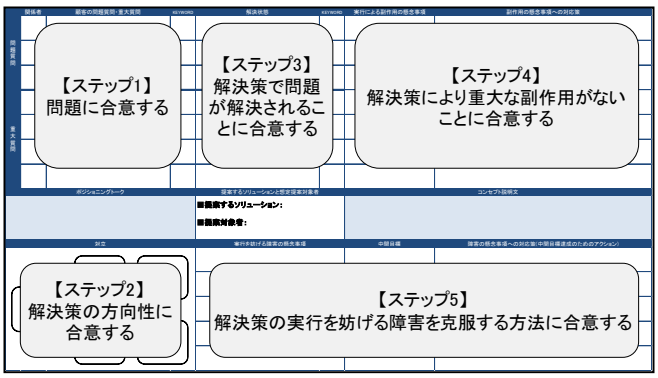



図 1　マフィアオファーシートと合意形成プロセス

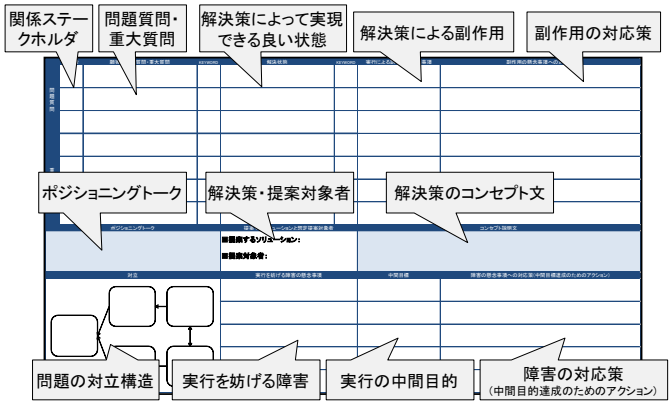



図 2　マフィアオファーシートの各要素

(5)変更の実現方法

派生開発推進協議会(AFFORDD)[8]の T1 研究会では，XDDP(eXtreme Derivative Development Process)[9][10]の導入障壁の克服方法を研究テーマに活動している．T1 研究会は，2011 年，XDDP が解決したい問題構造を明確にするため，XDDP の考案者である清水氏の書籍“「派生開発」を成功させるプロセス改善の技術と極意”について，TOC 思考プロセスを用いた問題構造分析を行なった．結果，図 3 に示すような問題構造を抽出することができた．本分析には，以下に示す工数がかかった．

- 人員:6 名(報告者含め)
- 期間:2 日間(合宿にて実施)
- 備考:報告者は TOC 思考プロセスの専門資格(TOC-ICO 登録 Jonah)を有する

こうして得られた問題構造については，分析を実際に行なった研究会のメンバー内での満足度は高かったが，図 3 に示す通り構造が複雑(これでも問題構造としてはシンプルな部類)であるため，他者へ提案が困難である．また，TOC 思考プロセスの知識を持っている報告者がサポートしていたが，時間としては 2 日間という長い時間がかかった．このように(3)に示した問題解決型プロセス改善では，時間がかかる上，問題分析の結果を共有しづらい．

そこで，T1 研究会では XDDP についてマフィアオファーシート(図 4)を作成した[11]．

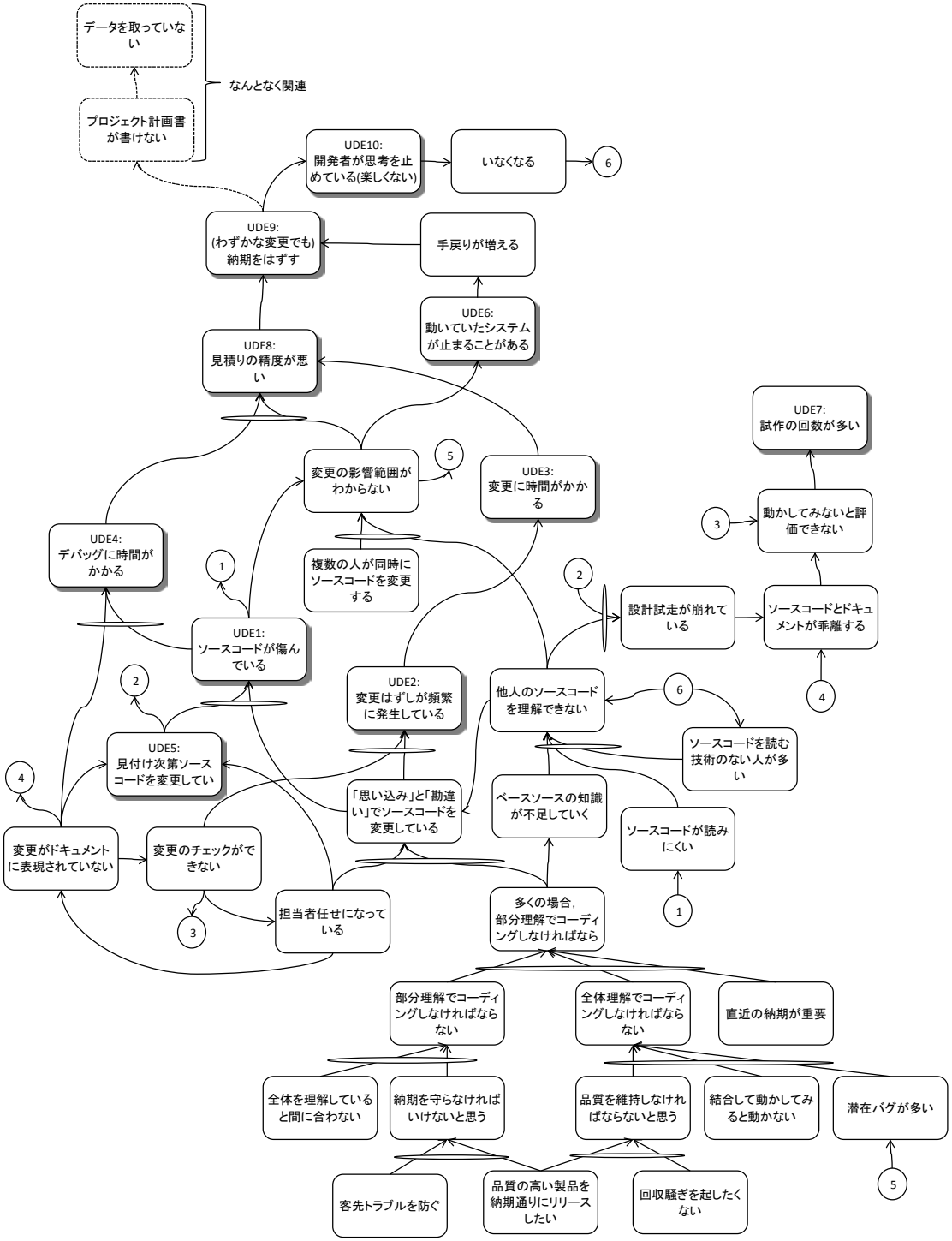



図 3　TOC 思考プロセスによる XDDP が対象とする派生開発の問題構造

| | | | |
|---|---|---|---|
| 解決策 | XDDP | 確認質問 | ソフトウェアの派生開発を行っていますか？ |
| 対象 | 派生開発において納期遵守に困っているソフト開発関係者 | | 品質低下や納期遅延で困っていますか？ |

| | 関係者 | I: 問題に合意する | KEYWORD | III: 解決策で問題が解決されることに合意する | KEYWORD | IV: 解決策により重大な副作用がないことに合意する |
|---|---|---|---|---|---|---|
| 問題質問 | 開発者，マネージャー，経営者，顧客 | 後工程やリリース後にデグレードや変更間違いによる手戻りが多いですか？ | デグレードと変更間違いによる手戻り | コーディング後の手戻りが少ないと良いですか？ | コーディングの留保 | 工数が増える／追加の作業が増える<br>・レビューによって手戻りが減るので，工数増加はしないというロジック／事例を説明する<br>・PFDによりプロセス・ドキュメント体系をプロジェクト毎に適切に設計するので，作業の無駄が減る<br>・スモールスタートで検証してみる |
| | 開発者，品質保証部 | 開発プロセスが実情にあっておらず，無駄だと感じる作業がありますか？ | 開発プロセスの無駄 | 派生開発に適したプロセスで無理・無駄がないと良いですか？ | 変更用プロセス | コーディングを留保しすぎて，納期を守れなくなる<br>・サイズ見積りに基づく，工程見積りにより，コーディング開始時期を明確に定義する．これにより納期を守れないという状況は発生しづらいことを説明する |
| | 開発者，マネージャー | (時間がないや納期が怖いなどの理由で)ソースコード変更の精査は担当者任せになっていませんか？ | 担当者任せになる変更精査 | 変更方法が十分かつ効率的に設計・レビューされていると良いですか？ | 変更用ドキュメント(変更3点セット) | 自組織の開発に適さない可能性がある／本当に自組織で効果があるかわからない<br>・スモールスタートで検証してみる<br>・既存開発のデータを用いて，効果をシミュレーションしてみる |
| 重大質問 | 開発者，マネージャー，経営者，顧客 | ささいな変更だと思われたものでも納期に間に合わないことが多いのではないですか？ | 納期遅延 | 見積り通りに開発が終了すると良いですか？ | 見積り通りの開発 | 失敗のリスクがある<br>・スモールスタートで検証してみる<br>・既存開発のデータを用いて，シミュレーションしてみる |
| | 開発者，マネージャー，経営者，顧客，品質保証部 | ソフトの品質がどんどん劣化していませんか？ | ソフト品質の低下 | ソフト品質が維持／改善していると良いですか？ | ソフト品質の維持／改善 | 過大な効果を期待してしまう<br>・過大な期待をさせないように，トップ，マネージャーに正確な情報を入力する<br>・スモールスタートで早期に適用効果を見積る |
| | 開発者，マネージャー | 開発者のモチベーションが低下していませんか？ | モチベーションの低下 | 開発者が開発の意義を感じていると良いですか？ | モチベーションの向上 | 定着しない<br>・エバンジェリストを育成する<br>・組織的に定着化を図る<br>・トップダウンで適用を宣言する |
| | | | | | | 成功後に他プロジェクトに巻き込まれる<br>・トップ，マネージャーに他プロジェクトに巻き込まないという確約をもらう |

| ポジショニングトーク | 提案する解決策と想定提案対象者 | コンセプト説明文 |
|---|---|---|
| XDDPとは，<br>派生開発において，品質が低下し，納期も守れなくなるという問題に対処する手法．従来の変更箇所を見付け次第変更するといった開発とは異なり，コーディングを留保し，その間で徹底的にレビューを行うことで，手戻りがなくなるため，納期も守りながら品質も維持することが可能になる．そのための効率的なドキュメント(変更3点セット)や変更プロセスを含んでいる． | **■提案するソリューション：**<br>**XDDP**<br>**■提案対象者：**<br>**派生開発において納期遵守に困っているソフト開発関係者** | デグレードや変更間違いによる手戻り，開発プロセスの無駄，担当任せのソースコード変更精査といった問題があるとのことですが，XDDPは，コーディングの留保，変更専用のプロセス，変更専用のドキュメント(変更3点セット)により，解決することが可能です．これらの問題が解決できれば，ソフト品質低下，納期遅延，モチベーションの低下といった重大な問題を解決できます．さらにそのことで，来たるべき新規開発への備えも可能になります． |

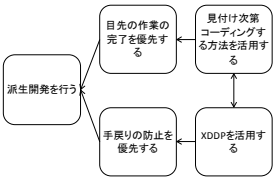
| II: 解決策の方向性に合意する | V: 解決策の実行を妨げる障害を克服する方法に合意する |
|---|---|
|  | (1)社内関係者の合意を得る<br>・社内関係者に対して，本マフィアオファーシートを用いて合意を取る<br>・対象者に合せてマフィアオファーシートをカスタマイズする |
| | (2)社外関係者の合意をとる<br>・社外関係者に対して，本マフィアオファーシートを用いて合意を取る<br>・Win-Winになるような方法の検討のために，対象者に合せてマフィアオファーシートをカスタマイズする |
| | (3)組織標準や従来のやり方との対応をとる<br>・組織標準のドキュメントやプロセスとの対応関係を取る(USDMは〇〇仕様書に対応する，など)<br>・XDDPを組織にテーラリングした事例を参考にする |
| | (4)導入工数を確保する<br>・工数／予算の決定権のある人物にXDDPをプレゼンして，工数／予算を貰う<br>・スモールスタートで検証して，必要工数／コストを見積る<br>・既存開発のデータを用いて，擬似的に検証し，必要工数／コストを見積る |
| | (5)スキルを習得する<br>・AFFORDD主催の勉強会に参加する<br>・独自の勉強会を開催する<br>・エバンジェリストを置いて，展開を推進する<br>・XDDPのスキルは，基本的には「書く」だけのことであることを認識してもらう |

図 4　マフィアオファーシートの例(XDDP)

(6)変更後の状態や改善効果

T1 研究会のメンバーが各人の組織や勉強会にて，前述のマフィアオファーシートを基に XDDP 導入提案を試行した．その際，提案対象者に対して，XDDP に関する質問とどのステップまで合意できたかを確認できる質問で構成したアンケートを実施した．提案対象者は開発者を中心に合計 23 名となった．結果，図 5 に示すように，提案対象者の 87%に「XDDP やってみよう」という導入の動機付けを与えることができた(Q3)．

このようにマフィアオファーを用いると，提案対象者自身が問題分析を行う必要はなく，改善推進者が提案する手法に対して，導入の動機付けを行うことができる．

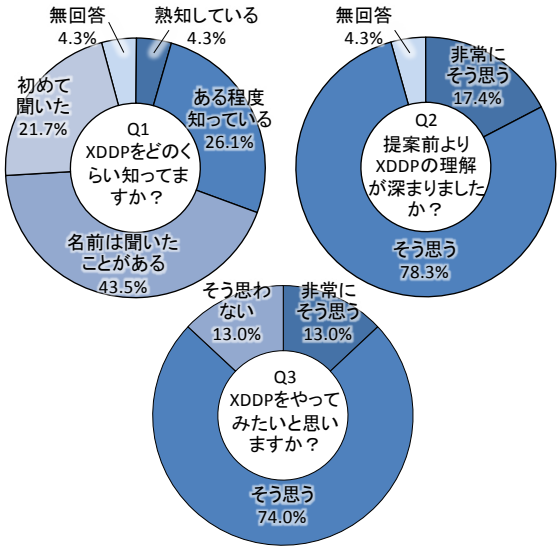



図 5　マフィアオファーによる提案に対するアンケート結果

(7) 改善活動の妥当性確認

推進者や開発現場の協力者が既存のソリューションを提案することでプロセス改善を推進する場合において，モベルベース型や問題解決型ではない，第三のアプローチとして，マフィアオファー型プロセス改善を提案した．マフィアオファーシートを用いることで，スキルを要する問題分析を提案対象者に課することなく，問題構造とソリューションの関係性を明確にし，ソリューションの動機付けを行うことができることを示した．また，XDDP 提案を行なった研究会メンバーより「説明しやすい」という意見を得ている．よって，マフィアオファーシートを用いることで，ソリューションを「やらされ感なく」「他者へ簡潔に説明できる」と考えられる．

ただし，本報告では新たな協力者が他者へ XDDP を提案するところまでは追跡できていない．その点については今後の課題であると考えている．


[1] SaPID (Systems analysis／Systems approach based Process Improvement methoD), http://hba-web.sharepoint.com/Pages/SaPID000.aspx

[2] Eliyahu M. Goldratt, ザ・ゴール 2 ―思考プロセス, ダイアモンド社, 東京, 2002.

[3] 村上悟, 問題解決を「見える化」する本, 中経出版, 東京, 2008.

[4] SPINA3CH (Software Process Improvement with Navigation, Awareness, Analysis and Autonomy for CHallenge), http://www.ipa.go.jp/sec/softwareengineering/reports/20110707.html

[5] 村上悟, 高橋淳, 小林昇太郎, 儲かる会社のモノづくり マーケティング 売るしくみ, 中経出版, 東京, 2008.

[6] 西原隆, “リーン TOC によるヒット商品開発,” リーンカンファレンス 2013, (2013).

[7] 西原隆, “『マフィアオファー』TOC 流で断れないほど魅力的な提案を開発する!!, ” 第 5 回アフォードフォーラム, (2014).

[8] 派生開発推進協議会 AFFORDD, http://www.xddp.jp/

[9] 清水吉男, 失敗しない派生開発(Software People Vol.8), 技術評論社, 東京, 2006.

[10] 清水吉男,「派生開発」を成功させるプロセス改善の技術と極意, 技術評論社, 東京, 2007.

[11] 八木将計, 奥山麻美, 佐津川勝彦, 須田晃, XDDP 導入を断わることのできない提案, 派生開発カンファレンス 2013, (2013).

2A3「XDDP と SPLE の連携・移行・使い分けガイドの紹介」派生開発推進協議会 T-14 研究会

〈タイトル〉:　**XDDP と SPLE の連携・移行・使い分けガイドの紹介**

〈サブタイトル〉:　どちらを適用するかで迷ったときに

〈発表者〉
氏名（ふりがな）:　派生開発推進協議会 T-14 研究会（SPL と XDDP の連携）（はせいかいはつすいしんきょうぎかい てぃーじゅうよんけんきゅうかい えすぴーえるとえっくすでぃーでぃーぴーのれんけい）
所属：
〈共同執筆者〉
氏名（ふりがな）:　安倍 信孝（あべ のぶたか）
所属：　パナソニック ファクトリーソリューションズ株式会社

氏名（ふりがな）:　梶本 和博（かじもと かずひろ）
所属：　株式会社エクスモーション

氏名（ふりがな）:　北崎 敦（きたざき あつし）
所属：　株式会社リコー

氏名（ふりがな）:　桜庭 恒一郎（さくらば こういちろう）
所属：　株式会社日立産業制御ソリューションズ

氏名（ふりがな）:　林 好一（はやし よしかず）
所属：　株式会社ＳＲＡ

氏名（ふりがな）:　藤田 将志（ふじた まさし）
所属：　東京エレクトロン株式会社

〈要旨〉

XDDP は派生開発を適切に行なうための方法論である。一方、SPLE は、類似した複数のシステムを効率的に開発するためのパラダイムである。これらのどちらを採用すべきであろうか。SPLE が必ずしも XDDP よりも「良い」やり方という訳ではなく、それぞれ適用の適した場面があり、また両方を同時に使う開発もある。典型的には、XDDP から SPLE に開発の形を移行させることを検討するであろうが、その逆方向の移行もありうる。また、移行ではなく、同じ製品/システムの開発で、一部を XDDP で、一部を SPLE で開発する形もある。さらに、XDDP での開発に SPLE の技術の一部を取り入れて、より良い結果を導くことも考えられる。発表者は、このことをわかりやすく説明するガイドを執筆中であり、それを 2014 年 11 月に公開する予定である。
本発表では、前述のガイドの構成を紹介する。併せて、どちらも類似システムの開発のために適用検討される XDDP と SPLE を、排他的な選択肢ではなく状況に合わせて「いいとこ取り」のできる対象として、その形態を例示し、どちらを採用すべきかという上掲の課題を持つ者の判断に供する。同時に、さらに詳しい分析が読めるガイドの閲覧へと導く。

〈キーワード〉

XDDP, SPL, SPLE, PLE, eXtreme Derivative Development Process, Software Product Line, Software Product Line Engineering, Product Line Engineering, 派生開発, 改造開発, プロダクトライン開発, プロダクトライン, ガイド, いいとこ取り

〈想定する聴衆〉

XDDP は導入済みまたは導入検討中だが､SPLE は導入障壁が高いと感じているソフトウェア開発従事者（技術者、マネージャ、SEPG）

〈適用状況〉

□多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
■その他（発表翌月には「適用できる段階あるいは初めて適用する段階」となる予定）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

- 主体は発表者に同じ。複数の企業に所属する者が、それぞれの知見を持ち寄ってさらに知識を深める活動を行なっている
- XDDP は、単発の派生開発に有効な手法であり、改造時の品質向上が期待できる
- 一方、ソフトウェアプロダクトライン開発（SPLE）は、製品系列で長期間に渡って派生を繰り返す場合の効率向上を図る目的で導入する組織が多い
- 当研究会では、XDDP と SPLE のそれぞれが適している条件の検討や XDDP と SPLE の間を移行する際の留意点などを議論してきた
- 議論の結果は、整理してガイドの形にし、公開すべきだと考えた
- 派生開発の状況を調べるためにアンケートの実施も行なっており、そこから見えた現状課題に対応することも必要だと考えた
- 発表応募時点では、当ガイドはまだ完成していない

(2) 改善前の状態

- XDDP も SPLE も、アンケート調査の回答者の周囲では、必ずしも正しく理解されていない
- SPLE は XDDP よりも導入障壁が高いと感じられている
- XDDP と SPLE は互いに関連のない別のものだと捉えられている
- 正しい理解があれば、状況に合わせて両者の技術や考え方を選択して開発に役立てることがより容易になる

※アンケート結果の分析より判断した

(3) 改善前の状態をもたらした原因（因果関係）

- XDDP も SPLE も、アンケート調査の回答者の周囲では、必ずしも正しく理解されていない
- そのため、導入の方法、時期、適用対象などが適切に判断されていないように思われる

※アンケート結果の分析より判断した

(4) 計画した変更内容

- 上記(3)を解消するためには、正しい知識を、読み易い形で提供することが必要と判断した
- これをガイドという、XDDP または SPLE に興味のある読者が手に取ってくれるであろう形に編集することにした
    - 想定読者は、本発表の想定聴衆にほぼ同じだが、XDDP は一応実行できている組織の人で、かつ SPLE の基本的なことは知っている人という条件が加わる
- ガイドの中心は、XDDP と SPLE の間には移行だけではなく、使い分けや融合という形態があるということを説明する
    - それぞれの形態を以下に説明する

【移行】

- 同一システム/製品のシリーズに対して、XDDP（SPLE）のプロセスでの開発していたのを、SPLE（XDDP）のプロセスでの開発に切替える
    - XDDP → SPLE　または　SPLE → XDDP
- 形態には
    - 全面的な切替
    - 部分的な切替
- があり、後者では次の「使い分け」が生じる

【使い分け】

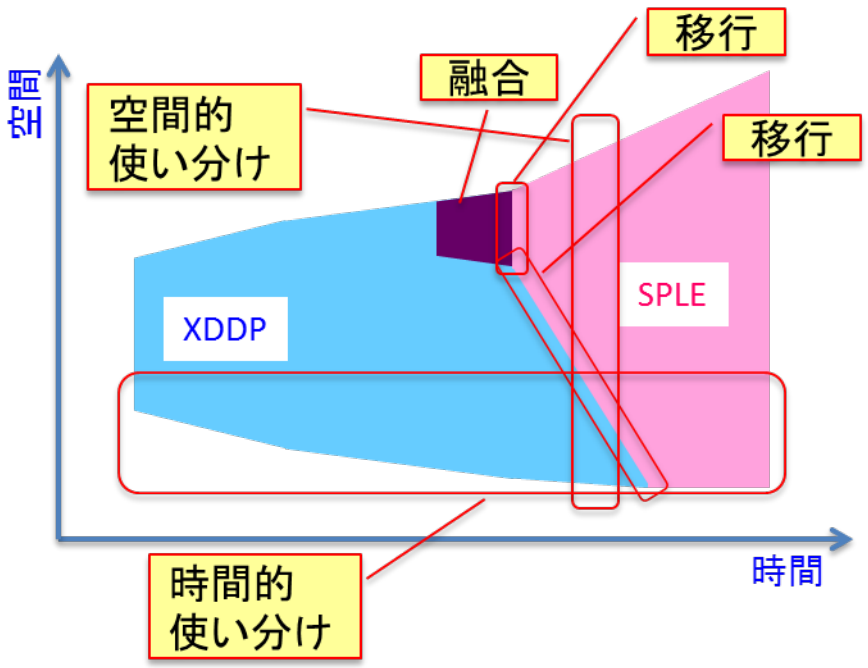
- 同じシステム/製品の中で、XDDP と SPLE を別々の部分に適用する
    - 「移行」においてこのパターンが現われることがある
- 極端な例として、どちらか一方しか適用しない場合もある

        - 途中で条件が変わり、「移行」を実施することはありうる

【融合】

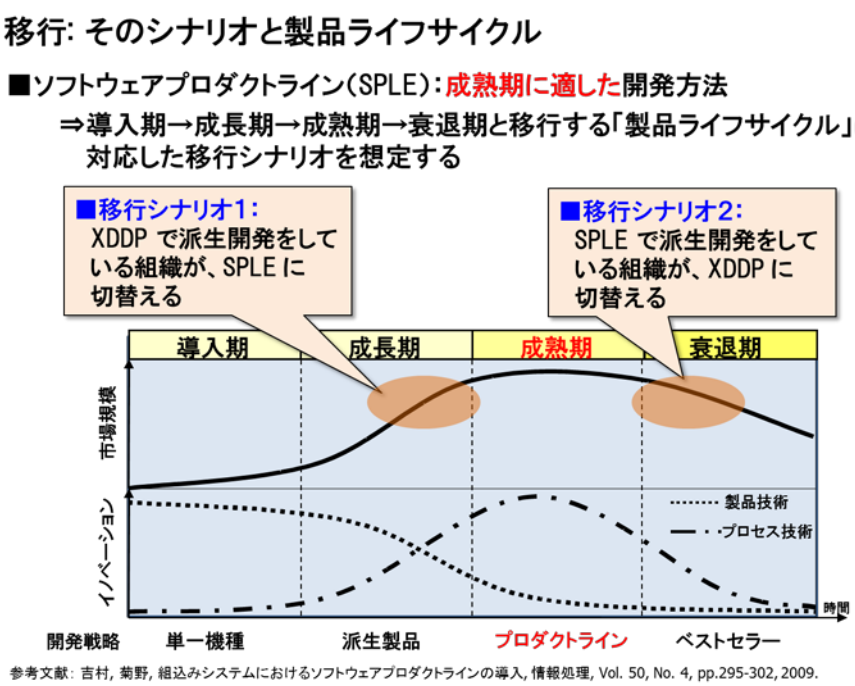
- 基本的には XDDP を実施するが、SPLE に使う技術を応用してより良い開発を行う
- SPLE に使う技術 ＝ 可変性を扱う技術
    - 可変性モデリング（フィーチャモデリング）
    - 可変性モデルと資産の関連付け
    - 可変性の実現（サブクラス化、設定ファイル、モジュールの入れ換え、等）

- これらの形態を、横軸に時間（タイミング）、縦軸に空間（システムのそれぞれの部分、それぞれの種類の成果物等）をとって、下に模式的に例示する



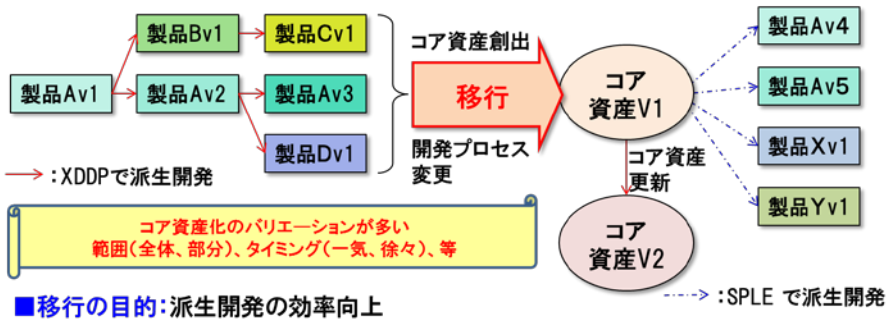

- ガイドの構成は以下の通り
    - 当ガイドの背景
        - このガイドを作成した動機
        - 発端となったアンケート集計結果の概説
    - 当ガイドの使い方
    - 想定読者
        - 技術者、プロセス支援者（SEPG, SQA, etc）、技術のわかるマネジャ
        - XDDP はできていること
        - SPLE に関する基本的な知識があること
    - XDDP と SPLE
        - それぞれの概要
        - それらの違い
        - 導入/移行のシナリオ
    - プラクティス（仮題）
        - フィーチャ分析とその成果の扱い

        - 過去製品の分析
        - 共通性・可変性の扱い
        - リファクタリング
        - （他のプラクティスも検討中）
    - プラクティスの適用
    - 移行、使い分け、融合
        - それぞれの違い
            - 移行、融合の際に必須のこととそうでないこと
        - プラクティスの適用
            - 移行
            - 使い分け
            - 融合
        - プラクティスの組合せパターン（か、それが難しければ組合せ例（できれば））
    - 結言
    - 付録
        - （アンケート集計結果、これまで作成した資料、等）
- ガイドの内容は概ね一般論になるが、読者それぞれの状況と関連付けて理解できるように、いくつか比較的具体的な適用シナリオを用意する
- シナリオの例を以下に示す



## 移行シナリオ1：XDDPからSPLEへ

■移行シナリオ1：XDDP で派生開発をしている組織が、開発方法を SPLE に切替える



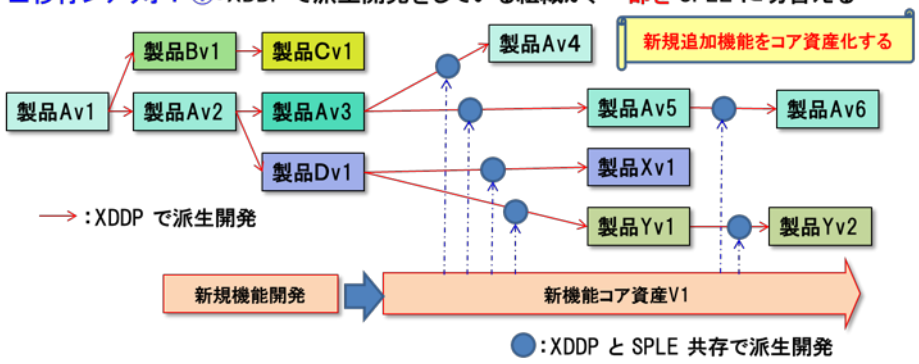

■移行の目的：派生開発の効率向上

■背景：製品系列のバリエーション、派生開発件数の増加に伴う開発コスト増大

■課題：①既存資産を基に SPLE コア資産の開発方法検討
⇒現状資産の状態や組織等の状況に適した SPLE の開発方法を採用する
②コア資産の可変性を実現するためのアーキテクチャ検討
③移行の際に XDDP の成果物を活用する方法の確立
⇒過去の XDDP 成果物（変更要求トレーサビリティマトリクス）をコア資産化の際の検討材料として使用する、等

## 移行シナリオ1 の変種1

■移行シナリオ1-①：XDDP で派生開発をしている組織が、一部を SPLE に切替える



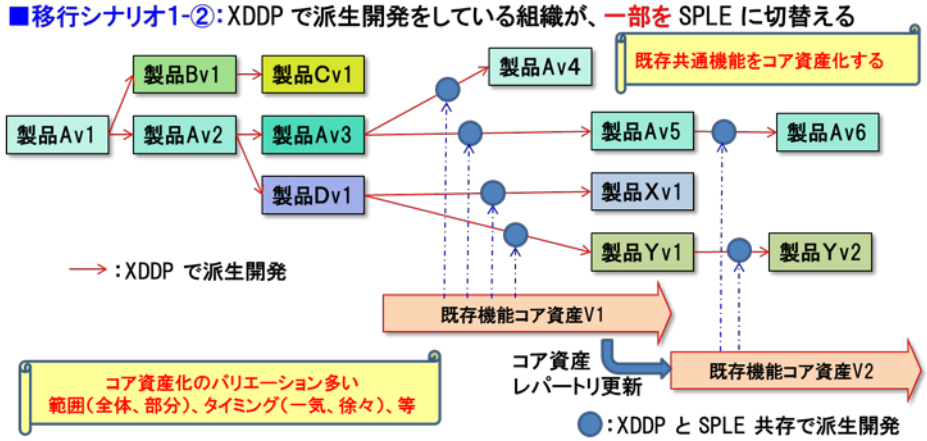

■移行の目的：既存製品への新規追加フィーチャのコア資産化による効率向上

■背景：既存製品群共通の新規機能追加

■課題：①新規機能をコア資産化（フィーチャ分析実施）
⇒既存製品群へのコア資産適用
②コア資産の可変性を実現するためのアーキテクチャ検討

## 移行シナリオ1 の変種2

■移行シナリオ1-②:XDDP で派生開発をしている組織が、一部を SPLE に切替える



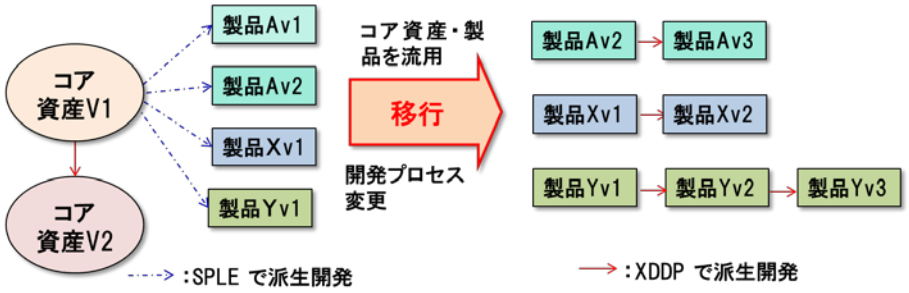

■移行の目的:既存製品の共通フィーチャのコア資産化による効率向上

■背景:既存製品群共通機能の増加および維持・保守効率悪化

■課題:①既存機能のコア資産化(フィーチャ分析実施)
⇒既存製品群へのコア資産適用
②コア資産の可変性を実現するためのアーキテクチャ検討

## 移行シナリオ2: SPLE から XDDP へ

■移行シナリオ2:SPLE で派生開発をしている組織が、開発方法を XDDP に切替える



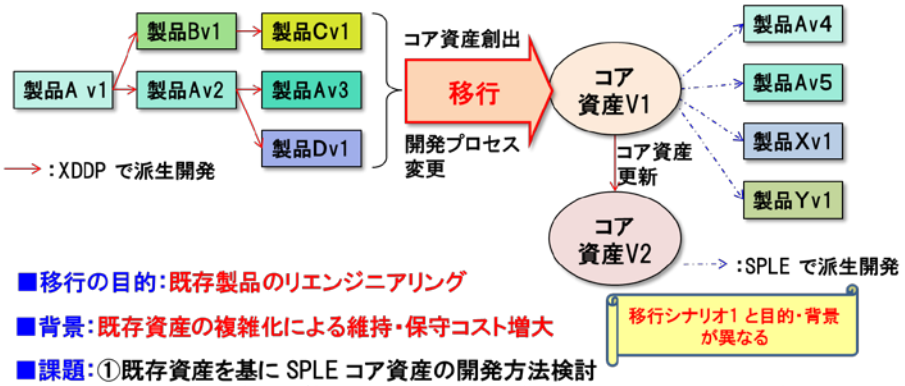

■移行の目的:開発効率の維持 ←コア資産維持工数の削減

■背景:製品系列寿命の衰退期への遷移に伴う派生開発件数の減少

■課題:①SPLE コア資産の運用方法改訂(例:コア資産管理をコア資産管理チームから
製品開発チームに移管してしまう)
②SPLE 開発で納入した製品の保守方法確立(担当組織、改訂プロセス、等)

## その他の移行シナリオ

■移行シナリオ3：XDDP で派生開発をしている組織が、開発方法を SPLE に切替える




■移行の目的：既存製品のリエンジニアリング

■背景：既存資産の複雑化による維持・保守コスト増大

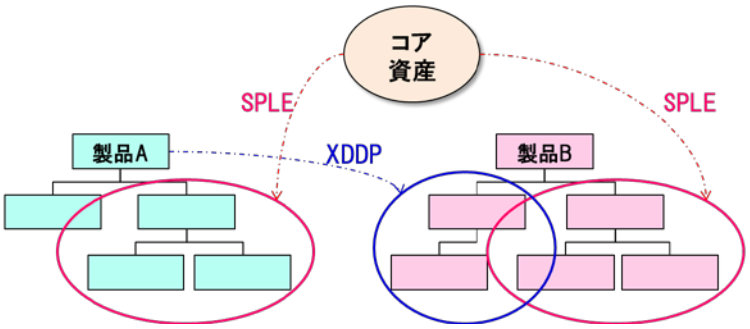
■課題：①既存資産を基に SPLE コア資産の開発方法検討
⇒現状資産の状態や組織等の状況に適した SPLE の開発方法を採用する
②コア資産の可変性を実現するためのアーキテクチャ検討
③移行の際に XDDP の成果物を活用する方法の確立
⇒過去の XDDP 成果物（変更要求トレーサビリティマトリクス）をコア資産化の際の検討材料として使用する、等

## 使い分けの例1

■コア資産による恩恵（効果）が得やすい部分には SPLE を、そうでない部分には XDDP を適用

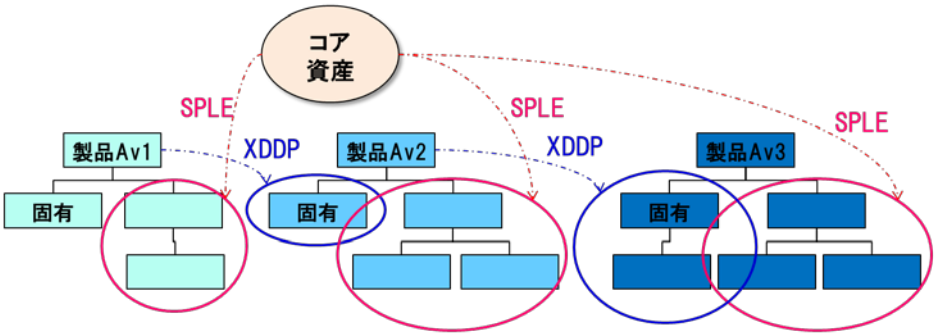
■コア資産の効果が大きい例：その部分は製品間での共通性が高い、今後開発する内容が比較的明らか、等

■例えば、GUI は組織全体で Look & Feel を統一するためにコア資産に基づいて構築する



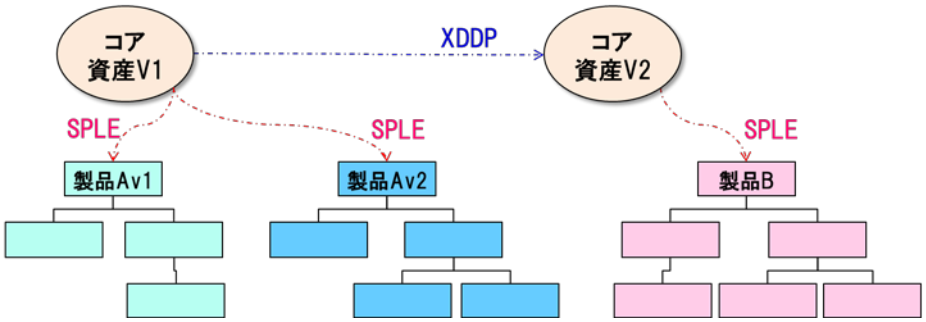
使い分けの例2

■そのシステム/機種に固有の(他のシステム/機種と共有しない)部分には XDDP を適用し、他の部分に SPLE を適用する




使い分けの例3

■各製品の開発には SPLE を、コア資産の進化には XDDP を適用する

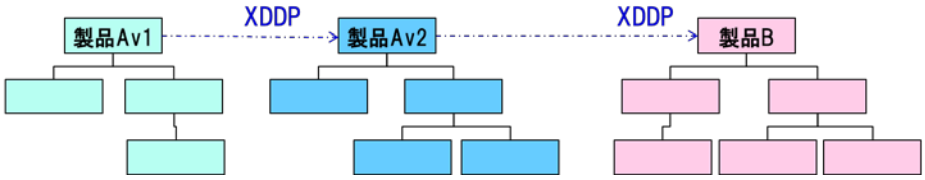


使い分けの例4

■シリーズ化の予定のない開発、またはシリーズ開発であっても開発のインターバルが長いものでは、SPLE を適用せずに XDDP のみで開発する

■コア資産による効果があまり見込めない場合の「使い分け」

■使い分けの例1 の特別な場合




融合

■「融合」では、SPLE で用いられる技術を XDDP に取り入れて効率・品質のさらなる向上を図る

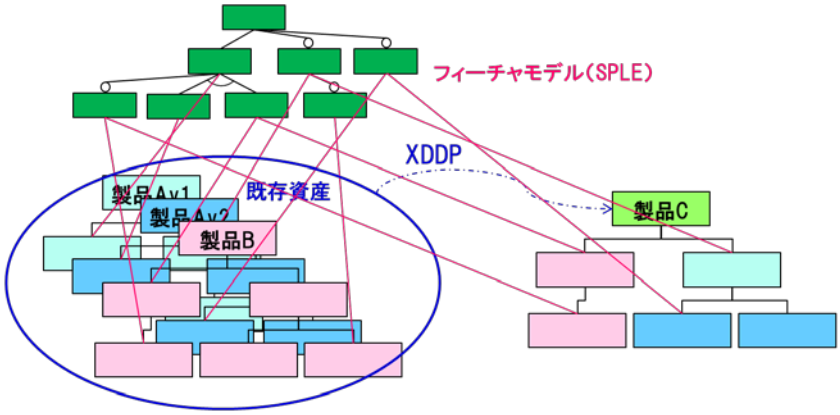
■SPLE に使う可変性モデルと各種成果物の可変点との間の関連付けは、可変点(どこが変わりうるのか)に関する間接的なトレーサビリティを示すことになる

■成果物: 要求仕様、アーキクチャ、詳細設計、実装、テスト仕様・設計、開発プロセス、等

■XDDP では可変点は陽に扱わないが、それぞれの開発で扱ったフィーチャをフィーチャモデルに組み上げ、開発間で共通であった点、異なった点を可変性モデルとして整理することによって、次回以降の開発で最も適切な過去資産(要求仕様の一部、アーキテクチャの一部、他)を見つけるのに役立てることができる

融合の例

■既存資産をフィーチャモデルを用いて整理し、次回以降の XDDP 開発のベースを識別する際に役立てる




(5) 変更の実現方法

- ガイドの構成および内容は、当研究会の会合で検討している
- 検討は、前述のアンケートの分析結果と、研究会メンバの知識および経験を基にしており、ガイドの各項目は起稿とレビューを重ねて蓄積していっている
- 経験はメンバの所属組織内または所属組織の顧客内の情報を含むため、一部抽象化してガイドに取り入れている

(6) 変更後の状態や改善効果

- ガイドはまだ未公開であり、効果の確認等のフィードバックは得られていない

(7) 改善活動の妥当性確認

- ガイドはまだ未公開であり、その作成プロセスの妥当性は確認できていない

2B1「SECI モデルによる改善活動基盤の評価」中村伸裕(住友電工情報システム)

〈タイトル〉:
SECI モデルによる改善活動基盤の評価
〈サブタイトル〉:
改善活動の第３ステージへ
〈発表者〉
氏名（ふりがな）:中村　伸裕（なかむら　のぶひろ）
所属：住友電工情報システム株式会社　ＱＣＤ改善推進部
〈共同執筆者〉
氏名（ふりがな）:
所属：

〈要旨〉
当組織では 2007 年 から公式アプレイザルを３回実施するなど、モデルベース改善活動を継続してきたが、評定で改善効果が期待できる弱みが少なくなってきている。今後、課題ベースの改善活動に軸足が移ることが予想され、これに伴い改善活動基盤も変更していく予定であるが､これまで構築してきた改善活動基盤の強みと弱みを理解していないため、改善活動基盤の変更が悪影響をあたえるリスクがある。そこで、野中郁次郎先生の『知識創造企業』[1]をモデルとして当組織の改善活動基盤に関する強みと弱みを評価した。

〈キーワード〉
改善活動基盤、SECI モデル、プロセス改善、知識創造、技能、暗黙知、形式知、改善推進部門

〈想定する聴衆〉
改善活動を推進する部門の方、SEPG

〈適用状況〉
■多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

本稿ではプロセス改善のステージを３段階で考える。第１ステージは CMM 登場以前に行われていた、経験知による手順の定義や改善活動である。第２ステージはモデルを使って弱みを特定し、改善する状態である。第１ステージは自分達の経験がベースになっているが、第２ステージでは世界中の経験が集約されたのもがベースになっている。従来なかったプロセスを新規追加することも多い。第３ステージは CMMI のプラクティスが自然に行われている状態で、自分達の経験知（暗黙知）をベースにプロセス改善を進める状態である。第１、第２ステージで新規されたプラクティスに経験から得た暗黙知をフィードバックし、既存の手順が更新される。このステージは世界の知識と組織内の経験知（多くは暗黙知）を統合する形でプロセス改善を行っているのが特徴である。

当組織では 2011 年に特別に編成されたチームが CMMI レベル５を達成し、2014/6 には対象を組織全体に広げ CMMI レベル５の公式アプレイザルを実施している。第２ステージから第３ステージに移行しようとしている状態であり、今後、改善の主体はモデルベースから課題ベースに移行すると予想されている。変化する環境のなかで改善活動が引き続きうまく進められるように、これまでの改善活動の経験を評価し、継続するもの、改善するものを把握する必要があると考えた。第３ステージでは経験によって開発者個人が保有する暗黙知に焦点を当てる必要があると考え、野中郁次郎先生の知識創造[1]をモデルとして改善活動基盤の評価を行うことにした。

以降、当組織の評価結果を報告するが、応募用紙のフォーマットに合わせ、プロセス改善がうまく進まなかった状態と実施した改善策、最終的な状況もあわせて記載する。

(2)改善前の状態

2006～2007 年は組織横断型の WG を組織したり、課単位での QC サークルを組織したりしたが時間の経過と共に活動が停止し、アウトプットを出せたチームは少数であった。
また、2007 年に実施した CMMI 公式アプレイザルでは、開発者が何をすれば合格できるのかといった意識が強く、納得感のないプロセス変更を行ってしまった。

(3)改善前の状態をもたらした原因（因果関係）

改善活動がうまく進まなかった要因は以下の通りである。

(3a) ソフトウェア開発に関する基礎知識が共有されておらず、それぞれの開発者が自身の経験のみに基づいて適切ではない問題設定や解決策を考えている。経験が異なれば関心事も異なり、改善活動に参加しているメンバー全員が納得できる問題認識やあるべき姿（改善ゴール）が設定できなかった。

(3b) ソフトウェア開発の知識が不足しており、あるべき姿がイメージできない

(3c) 組織目標との関係性や目的が明確でない、等の理由で改善活動の重要性が共有されていなかった。

(4)計画した変更内容

以下の内容を計画し、実行した。

(4a) ソフトウェア・エンジニアリングの評価軸の共有

ソフトウェア開発に対する価値観の基準軸を共有するために、組織内の全グループ長

（課長）およびすべての改善実施担当者に”CMMI 入門”を受講してもらい、各グループで進行中のプロジェクトの PIIs を作成してもらった。その PIIs をコンサルタントに評価してもらいモデルの理解を深めてきた。

部長、グループ長に昇進した人はソフトウェア開発の成功体験をいくつも持っており、自分のやり方に自信があるため、自分流のやり方を押し付けてしまうリスクがある。ソフトウェア・エンジニアリングの知識を全員が身に付けることで共通の評価軸を持ち、若手でも改善提案しやすく、また、改善提案を正しく評価できる体制の構築を図った。

(4b) 改善活動の位置付けの明確化

毎年、部長、グループ長が参加する会議体で組織目標およびアプレイザル結果に照らして実施する改善活動を決定することで、改善活動の位置付けを明確にする。

(4c) 外部知識の取り込み

改善活動にコンサルタントに参加していただき、継続的に外部の情報を改善活動参加メンバーに提供することで脳を活性化し、創造力が発揮できる状態を作り出す。

(5)変更の実現方法

*■改善活動基盤は(4)で計画した施策以外に、改善活動の中で暗黙的に確立されたものも多い。本稿では、意識的および無意識に構築した改善活動基盤を知識創造の観点で評価した結果を示す。*

(5-1) 改善活動の課題

当組織の改善推進部門が改善実施者のモチベーションを高めるために利用している主な手段は次の２つである。１つは Learning Organization（センゲ）[2]のクリエイティブ・テンションで、もう１つは一般的な知識欲（知りたいという要求）である。クリエイティブ・テンションは、高いビジョンを共有することで現実をビジョンに近づけようとする力である。(逆向きの力はビジョンを現実に近づけようとするエモーショナルテンションである。）ここではビジョンをあるべき姿という意味で To Be と表記する。To Be は改善活動のゴールを示しているため、To Be の設定が活動の成果や改善活動の活発さに大きな影響を与える。図１は、”テストデータ作成に時間がかかるので、テストデータを自動生成し、効率化する”という課題に対する各改善担当者の To Be をイメージとして示したものである。B さんはあまり効果が期待できない手軽な方法でやりたいと考えている。他の改善者はもっと効果的な施策を実施したいと考えているが、具体的な実現方法がイメージできていない。長年、組織の中で課題と認識され、解決されていない課題はこのようなものが多い。この様な状況で To Be を導くプロセスを明文化することが改善活動の推進部門の１つの課題であり、明文化できれば改善活動を推進する有益なツールとなる。

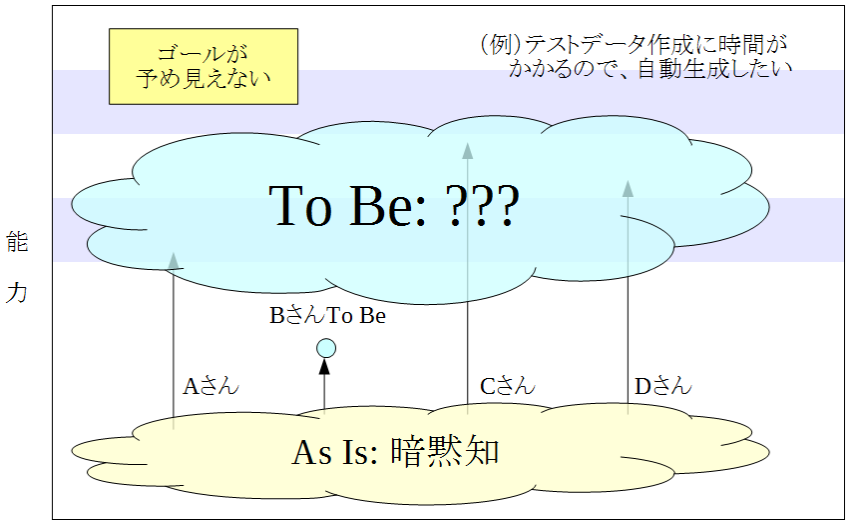



図１．改善活動におけるあるべき姿(To Be)設定の課題

(5-2) SECI モデルと改善活動の関係

当組織ではテストデータの作成方法は標準化されておらず、プログラマがそれぞれの方法で実施している。プログラマにテストデータの作成方法をヒアリングすると「適当にやっている」という回答が多く、データ作成はできるが説明できないという暗黙知の状態になっていた。まず、To Be を設定するためにブレインストーミングを行ったが、有効な解決策は見つからなかった。そこで、説明できないテストデータの作成方法を形式知に変換できないか考えた。各プログラマはそれぞれの方法でテストデータを作成し、顧客が満足する品質のソフトウェアを開発しているため、解決策を持っているはずである。テストデータの作成以外にも暗黙知の状態で実施されている以下のようなプラクティスが多く存在している。

・プロジェクト計画における各プログラムの規模 or 工数見積もり
・リスク管理におけるリスクの検出および発生確率変化の把握
・外部仕様書のレビューにおける機能不足の指摘
・要件定義における業務提案
・設計作業における要件から外部仕様への変換
・複雑な機能のコーディング

重要なプラクティスであるにもかかわらず、作成した成果物を説明することはできるが、経験の少ない初心者に言葉で手順を伝えることができないという問題を抱えている。

図２に野中郁次郎氏が提唱する SECI モデル[2]を示す。ものづくりに関わる技能の多くは暗黙知として体で覚え、実践されている。共同化の状態は、多くの技術者が OJT 等により同じ作業を行なっている状態である。表層化はその暗黙知を形式知に変換する作業である。主に対話によって行われる。連結化は形式知を組み合わせて新しい形式知を作る作業である。この作業は改善活動の中でよく行われる。内面化は個人が新しい知識を実践して体得している状態である。このサイクルを知識創造プロセスと呼ぶ。

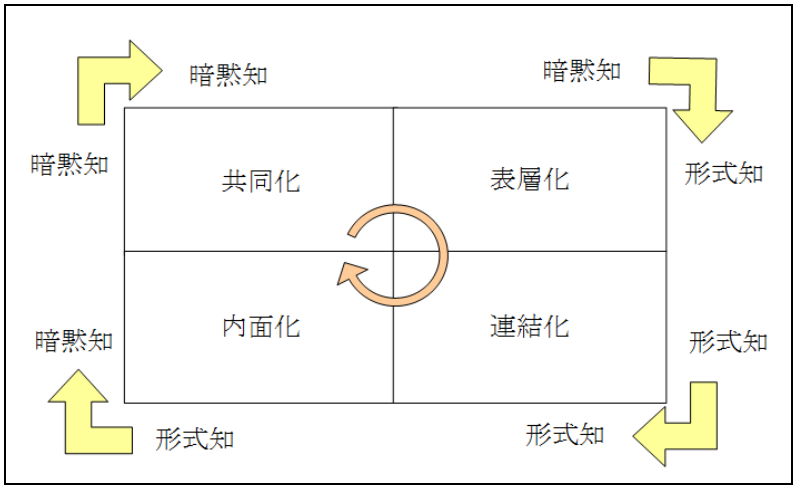



図２．SECI モデル

初歩的なプロジェクト管理は形式知化されているものが多く、このようなサイクルを経ることなく実施できるものも多い。しかし、ソフトウェア設計・実装では重要なプラクティスが暗黙知になっていることが多く、このような知識創造プロセスが必要となる。このサイクルがうまく回せるようになることが改善活動をうまく進めることであると考えている。(5-1)の To Be の設定の課題は、共同化から連結化のプロセスがうまく進められることに相当する。

(5-3) 知識創造プロセスが促進される条件

『知識創造企業』[1]には知識創造プロセスが促進される５つの条件が記載されている。ここでは、これらの５つの条件と当組織でのプラクティスを対比させることで、当組織のプロセス改善の暗黙知を形式知に変換した。

(a) 組織の意図

*（条件１）組織の構成員が、組織全体の目標、ビジョン、戦略を共に達成しようとする意図を持っていること。*

取締役、部長、グループ長、改善担当者で構成させる SWAT と呼ばれる会議体を月１回のペースで開催している。活動するＷＧは毎年、(i) 組織目標を達成するために必要な活動、(ii) CMMI モデルで抽出した弱みの克服、という２つの観点から参加者全員の意見を反映して決めている。組織の意図がプロセス改善に関わるメンバー全員で共有できるようになっており、知識創造が促進される体制になっている。また、品質や生産性の目標値は毎年、昨年の実績をベースに見直しており、プロセス改善を進めなければ目標は達成できない。常に改善ニーズが存在している状態となっている。

(b) 個人とグループの自律性

*（条件２）組織の構成員が、自由に発言し行動する機会を与えられていること。*

日本の改善活動はデミング博士の指導により活性化された。ＱＣサークルは業務が終わった後、従業員が無償で自発的に行っていた。当組織では業務時間中に改善活動を行って

いるが、当時、改善活動の成功要因を考え以下の様な考え方で進めている。

・内発的モチベーションを引き上げるために本人が希望するＷＧに参加してもらう
・改善活動の納期や成果物は参加者が決定する。
　（開発標準の改定、教育資料の作成、教育の実施等が推奨されている）
・改善活動の計画は外部の管理層がトレースするためのものではなく、改善活動の参加者がチーム力を最大化するために作成する。
・改善推進部門からの提言は参考意見で、実施事項の決定は参会者に委ねる。
・一歩前に進めば成果である。

　また、『最強組織の法則』[2]では意見交換の成立条件として、参加者がお互いをより深い洞察と明快さを追求する仲間だと見なすことが示されている。当組織では参加者が新人であってもソフトウェア開発を実践していれば暗黙知を保有しており、改善活動に貢献できると考えている。こうした考え方が自由な発言ができる土台になっていると考えられる。

　ただし、ＷＧの中には開発プロジェクトが多忙で、なかなかＷＧ活動に時間が割けないという時期もあり、時間的な自由度の制限は残っている。

(c) ゆらぎ／創造的なカオス

*（条件３）ゆらぎ：組織が前提としている習慣や行動スタイルに対して、周期的に疑問を投げかけること。*
*創造的なカオス：組織が危機に直面した時に、問題の発見とその解決に対して、構成員の注意が振り向けられること。*

改善推進部門の担当者は意識的に素人モードでの「なぜ、ＸＸＸしているのですか？」といった質問で開発者から本質を導きだしている。例えば以下のような会話が行われている。

開発者Ａ：「コスト削減のためには共通部品設計が重要。」
推進部門：「部品化とコストはどういう関係にあるのですか？」(*1)
開発者Ａ：「部品化することで作成するソースコードの量を減らすことができる。」
推進部門：「なぜ、ソースコード量が減ると開発工数が減るのですか？」(*2)
開発者Ａ：「？？？」
開発者Ｂ：「部品の利用方法を勉強する時間は考慮しなくてよいのですか？」

(*1)、(*2)が素人モードの質問である。この会話で”規模と工数は比例する”という思い込みが常に成立せず、新規と再利用でプロセスが異なる場合、単純には比較できないという結論にたどり着くことになる。
　開発を実践していない改善推進部門の担当者の方が、こういった質問が自然にできる。このような質問は思い込みに気づかせ、従来考えなかった領域に思考を誘導することになり、知識創造に有効な手段である。

また、SEC データ白書や JUAS ソフトウェアメトリックス調査の結果と組織のデータを比較して、「なぜ、当組織の実績が低いのか」というテーマでセッションを実施することで、現状に対する問題意識を持たせている。

(d) 情報の冗長性

*（条件４）直接業務にかかわる知識情報以外にも多くの知識を構成員が持っていること。また構成員同士の間で知識情報の重複や共有があること。*

改善推進部門の担当者は複数のＷＧに参加しており、ＷＧ間の情報伝達の役割を果たしている。各ＷＧの参加者は他ＷＧ活動で関係しそうな情報をリアルタイムで入手できるようになっている。

また、開発標準の他、全システムのドキュメントが閲覧可能になっており、他のプロジェクトがどのような成果物を作成しているのか、現状が把握しやすい状態になっている。

さらに組織としてプログラム開発の経験を重視しており、ＳＥは実装のことも知っているべきという文化も情報の冗長性に寄与していると考えられる。

(e) 最小有効多様性

*（条件５）複雑で多様な環境の変化に対応するのに、必要最低限の多様性を組織が保持していること。*

アイデアの多くは既存知識の組み合わせで生ませると言われており、普段得られない知識を提供することは知識創造の点で重要であると考えている。多様性は組み合わせる元の知識を増やす役割を持つ。多様性を確保するため各ＷＧは複数の部署からの参加者で構成するようにしている。コンサルタントに参加してもらって情報を提供してもらうこともある。また、改善推進部門の担当者は社外のコミュニティに参加したり、書籍や論文で社外の情報を入手したりして継続的に社外の知識を収集している。これらの手段により改善活動の場に知識フローができている状況を作り出している（図３）。改善担当者は継続的に新しい知識刺激を受け取ることで、アイデアが出やすい状況になっている。

また、SPI Japan は改善推進部門だけではなく、開発部門からも参加してもらうことで多様性の確保に寄与している。

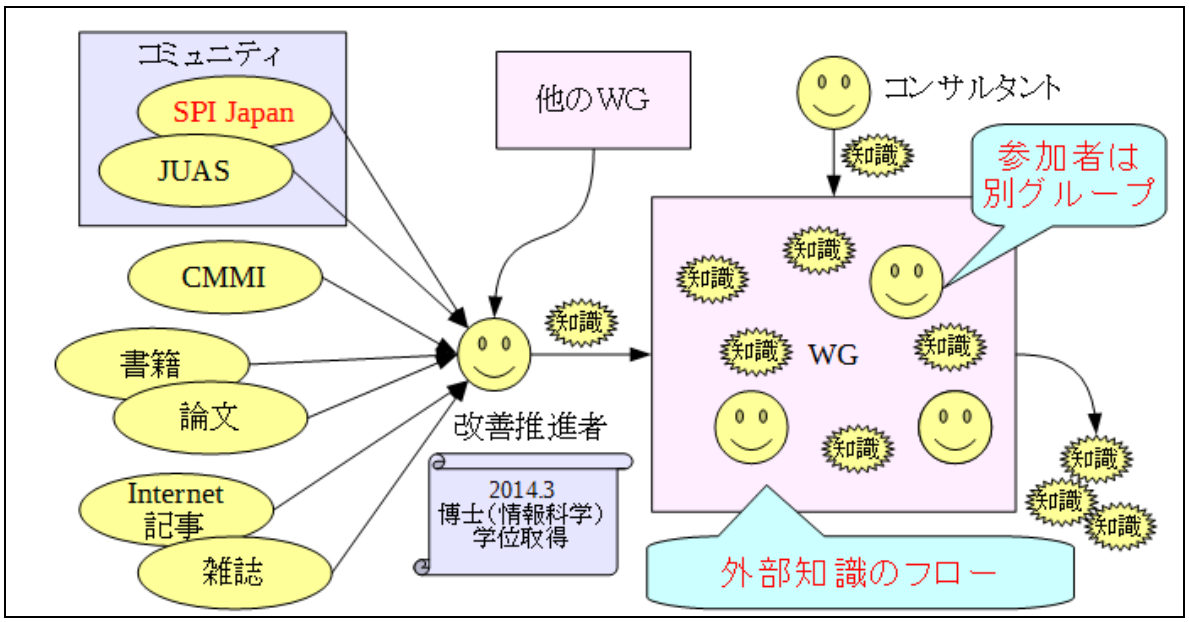



図３．知識フロー

(5-4) 組織構造

『知識創造企業』[1]では、トップダウンやボトムアップではなく、ミドル・アップ・ダウンが推奨されている。トップダウンはトップのみが知識を創造するので現場の暗黙知が反映されない。ボトムアップでは組織内の横展開が難しい。当組織では全グループ長が参加する会議体が改善活動の司令塔になっており、容易にボトムが持つ暗黙知を吸い上げたり、組織内に展開したりできる体制になっている。

(5-5) 暗黙知から形式知への変換

『知識創造企業』[1]には暗黙知を形式知に変換する方法として比喩やアナロジーを多用されることが書かれているが、より具体的な方法は書かれていない。以下、当組織の経験を記載する。表１に知識創造に関係する脳の活動を示す（なお、この表は専門家のレビューを受けてないので修正される可能性が高い)。当組織のＷＧ活動では、ある人が宿題をやってきて、その結果をレビューする。その際、表１に示すような活動が脳内で引き起こされる。暗黙知を形式知に変換する最初のステップは”思う”という活動であると考えている。違和感やうまく説明できないネガティブな印象がわき起こることがあるが、この感覚がトリガーとなる。この違和感を解消するためには質問や感想を述べることで議論を進めていく必要がある。その仮定で徐々に暗黙知が形式知として表現されていく。暗黙知は表現が難しいものであるため、メタファ（比喩）を使ったり、共通点と相違点に注目（アナロジー）したりして理解が進んでいく。共通の理解が得られれば、暗黙知は形式知に変換されており、ここから先はエンジニアが得意なロジカルに考えるという作業を経て To Be が設定されることになる。

なお、”悩む”という行為は成果物を生まないため回避したいが、現実には多くの工数が浪費される。我々は効率的に改善活動を進める方法をまだ発見していない。逆に時間の制約があると良い成果を得ることが難しいと考えており、非効率が高付加価値に繋がることを経験している。ＷＧの会議は通常 1.5～2 時間に設定しているが、議論が活性化している場合はアジェンダに縛られることなく、とことん探求する雰囲気になっている。

表１．知識創造に関係する脳の活動

| | 活動 | 説　明 | 動作 | 対象 |
|---|---|---|---|---|
| 1 | **ひらめく** | 異なる知識を組み合わせ、<br>**アイデア**を出す | 能動的 | 形式知<br>暗黙知 |
| 2 | 考える | 帰納法または演繹法を使って新しい<br>知識を作成。紙と鉛筆で**手を動かす**。 | 能動的 | 形式知 |
| 3 | 悩む | 労力と時間の無駄遣い。<br>手が動いていない。アウトプットなし。 | 能動的 | 形式知 |
| 4 | **理解する** | 納得する質問、メタファ（隠喩）、<br>アナロジー | 能動的 | 形式知<br>暗黙知 |
| 5 | **思う** | 入力に対して自然に脳が働く。<br>**違和感**、自身との違い、合意、・・・ | 受動的 | 暗黙知 |

(5-6) 改善推進部門の改善活動

改善推進部門では毎年 SPI Japan の応募を契機として過去１年間の改善活動の振り返りを行っている。活動の本質はどこにあるのか、といった観点で分析することで、改善活動に含まれる暗黙知を形式知化している。SPI Japan 2010 に最優秀賞をいただいた”ワーキンググループ(WG)活動を成功させる秘訣”の発表は、この活動の１つの成果である。

(5-7) まとめ

『知識創造企業』[1]をモデルとして当組織の改善活動を分析した結果、(5-1)から(5-6)の要素が当組織で改善活動が比較的活発に行われている要因であるという結論が得られた。これらの活動が今後も継続すべきも重要な要素であることを確認することができた。

(6)変更後の状態や改善効果

(6-1) CMMI モデルによる評価軸の共有

(4a),(4b) の活動の結果、CMMI のモデルがプロセス改善の共通の評価軸となり、CMMI を共通語としてプロセス改善の議論が行えるようになった。若手も上下関係に萎縮することなく、改善提案ができるようになっている。(3a)の問題が解消されている。

図４は 2014/3 に実施したアンケートの結果で、”ＣＭＭＩ（モデルおよびそれに基づく改善活動）について興味はありますか？”という質問の回答を示したものである。興味がある、どちらかというとあるの回答が半数を超えており、ネガティブな印象を持つ開発者よりも多い状態となっている。CMMI が単なる評価用ツールとして考えられているのではなく、ソフトウェア開発の基本知識として浸透した結果であると考えている。

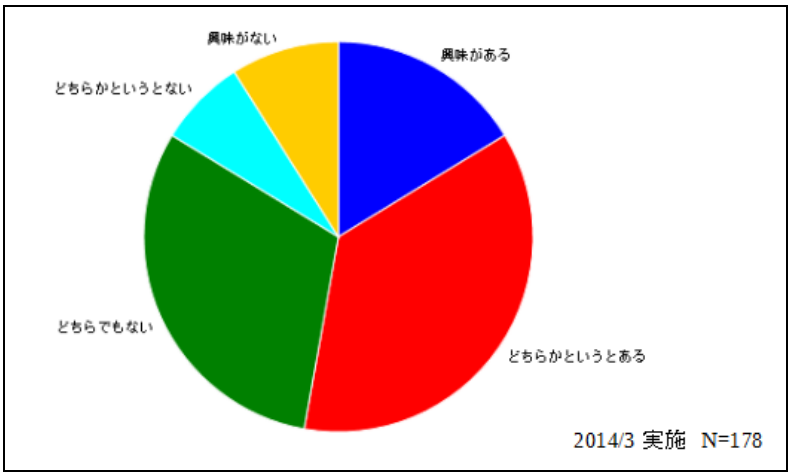



図４．ＣＭＭＩについて興味はありますか？

(6-2) 改善活動に必要な知識の供給

図５は“ＷＧ活動があなたの成長に寄与しましたか？”というアンケートの結果である。約半数が ”ＷＧ活動が成長にかなり or ある程度寄与した”と回答しており、改善活動の中でクリエイティブ・テンションが働いた、または、知識欲が満たされた結果であると考

えられる。(3b)の問題解決に必要な知識の習得ができていると考えられる。外部からの知識供給とＷＧの知識創造プロセスによるものであると考えている。
　ただし、ネガティブな評価も半数残っており、今後の課題である。

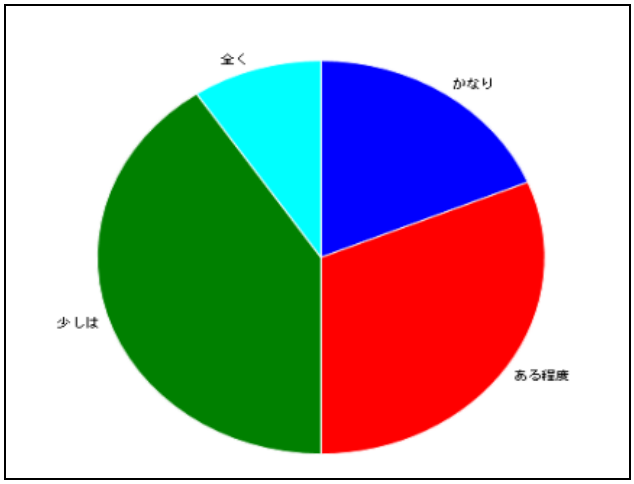



図５．ＷＧ活動があなたの成長に寄与しましたか

(6-3) 改善活動の状況
　(4)、(5)で示した施策の結果、以下に示すように改善活動が活性化している。

・2007 年, 2011 年, 2014 年に公式アプレイザルを実施
・開発部門からの公式アプレイザルのアプレイザル希望者が 8 名になり、改善部門からアプレイザが１名しか参加できない状況となった。開発部門の CMMI に対する意欲が高い。
・約 250 名の組織で“CMMI 入門”の受講者が 50 名を超える。
・役員、部長クラスが参加する改善活動の進捗報告の場が１ヶ月に１度開催され、改善活動が制度化され、定着している。
・開発標準は当初、改善推進部門が作成していたが、現在では開発部門の開発者が作成している。

(6-4) 改善活動の成果
　図６に出荷後に発見される欠陥密度の推移を示す。縦軸は 2005 年の中央値を 100 とした相対値である。2009 年以降大幅に改善されており、改善活動が組織目標達成に貢献していることがわかる。

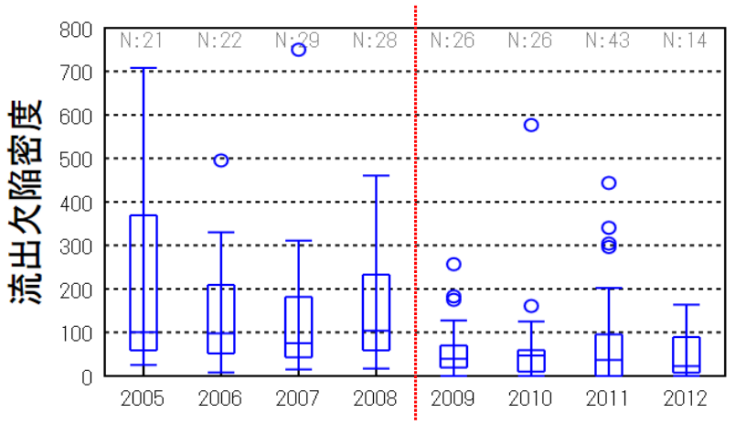



図６．出荷後の欠陥密度の推移

(6-5) 改善活動基盤の強みの理解

(5) で示したような改善活動基盤の強みを理解できた。

今後、改善活動推進における失敗リスクを低減できると考えられる。

(7)改善活動の妥当性確認

本稿では、『知識創造』をモデルとして利用し、当組織の改善活動の評価を行った。その結果、改善活動が活性化できている要因を理解し、今後の改善活動に活用できるようになった。また、プロセス改善は知識創造プロセスそのものであることがわかった。ただし、今回の評価の妥当性確認は実施できていない。今後、社外のコミュニティで議論することで、改善活動に対する形式知を修正・強化していきたい。

[参考文献]

[1] 野中郁次郎, 竹内弘高, "知識創造企業", 東洋経済新報社, 1996

[2] ピーター・センゲ, “最強組織の法則”, 徳間文庫, 1995

以　上

2B2「パッケージ製品の継続的開発における PDCA サイクル定着への取り組み」松田行正(住友電工情報システム)

〈タイトル〉: パッケージ製品の継続的開発における PDCA サイクル定着への取り組み

〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 松田行正（まつだゆきまさ）
所属：住友電工情報システム株式会社
〈共同執筆者〉
氏名（ふりがな）:
所属：

〈要旨〉

開発効率の改善や品質向上に必要を感じながらも、日々の開発に追われてプロセス改善に取り組む余裕がないと悩んでいるプロジェクトマネージャの方に是非お聞きいただきたい内容です。
現状分析して、少しずつ無理のないペースで改善を積み重ね、１年たったら大きな改善成果を実感できるようにマネージャが舞台裏で取り組んだ記録です。また、改善に対して反対する抵抗勢力をかわしながら実績を積み上げる実践的な手法もご紹介します。

〈キーワード〉

PDCA
仮説と検証
プロジェクトマネージャ
プロセス改善
反対
抵抗勢力

〈想定する聴衆〉

プロセス改善の進め方に悩みを持つプロジェクトマネージャ

〈適用状況〉

■多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（ ）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（ ）
□自プロジェクトのみ

〈発表内容〉

(1)背景

外販向けパッケージ製品( 製品 X と呼びます )は販売開始から 5 年を経過し､販売顧客数も毎年増加し、継続的に新機能開発や不具合修正を続けています。

製品 X の不具合修正は、社外から発見報告を受けたものや、社内でコードのリファクタリング中に見つけたものを定期的なバージョンアップの都度修正します。しかしながら、「 発見数 > 解消数 」の状態が続き、未解消の不具合数( 不具合総数 と呼びます )が増加傾向となっていることが社内で問題となっていました。


(2)改善前の状態

発見する不具合は、大きく分けて 2 つあります。

① 開発当初から潜在していた不具合で、購入顧客が増えるほど、開発当初に想定していなかった使い方によって顕在化する不具合( エラー時のメッセージ誤りなど )。

② パッケージ製品をより高機能にするために機能追加したが、既存機能に対してデグレーションとして混入する不具合。

この 2 種類の不具合は、毎月ほぼ一定の件数で報告が上がる傾向がありました。

本発表では、同一予算で作業効率を改善し､「解消数」の向上を図った事例ついて説明します。
なお、②のデグレを新たに埋め込まないアプローチ（「発見数」の削減)については本発表とは別に取り組みを行っています。

(3)改善前の状態をもたらした原因（因果関係）

プロジェクトメンバは意図してサボっているわけではなく、やれることはやっているという意識を持っているようでした。現状把握のために現場ヒアリングを行ったところ、「無駄な会議時間」「設計レビューの決定事項が曖昧で、再レビューしても前回決定したはずの内容が議論になり堂々巡りが起こる」といった問題が判明しました。

「無駄な会議時間」

不具合修正の設計及びコードレビューの会議を設定しているが、レビュワーが多忙という理由で、会議の開催時刻が 15 分、30 分と遅れることが頻発し、レビュー時間が十分に確保できていませんでした。
その結果、修正作業が完了しているにもかかわらず、すべてのレビューが完了できないため、リリースできない不具合修正が残る状態となっていました。」

「修正レビューの決定事項が曖昧」

レビューの指摘内容が「〇〇の方がいいんじゃないか?」「××だと思う」といった曖昧な文言で議事録もそのまま転記しているため、指摘を受けた設計者は何をしたらいいのか分からず、直すべき箇所が漏れたり、直さなくて良いコードを直して結局やり直すといった無駄が生じていました。

他にも問題と思うところに気が付くこともありましたが、製品 X の開発チームには現状を改善する活動は積極的には行われていないようでした。

(4)計画した変更内容

私は不具合件数が顕著に右上がりになり始めた頃にプロジェクトマネージャとして製品 X の開発チームに参画しました。ヒアリングで判明した 2 つの問題を解消することは容易で、単位期間あたりの「不具合解消数」の向上に効果があることも予想できました。

しかし、この 2 つを改善して終わりでは、製品 X 開発チームにプロセス改善の文化が根付かないと危惧しました。私は開発チームに「少しの成功体験」を実感してもらい、次の「成功体験」にチャレンジする意欲を持たせ、「少しの成功体験」を積み重ね、結果として「大きな改善」を身に着ける計画を立てることにしました。

(5)変更の実現方法

最初に 2 つの改善施策をプロジェクトマネージャ権限で導入し、改善効果を確認する会議を設けて下記のように同意を連鎖する方法で改善を継続することにしました。

① 改善後の開発効率を検証し、効果があったこと同意してもらいます。
② 今後もその改善内容を継続して開発効率を維持する同意を取り付けます。
③ 次に、他にも取組をする価値がある改善ポイントがあるのではないかという仮説を募集します。
④ 仮説を実行する合意を得て次の保守開発サイクルで仮説を試験導入します。

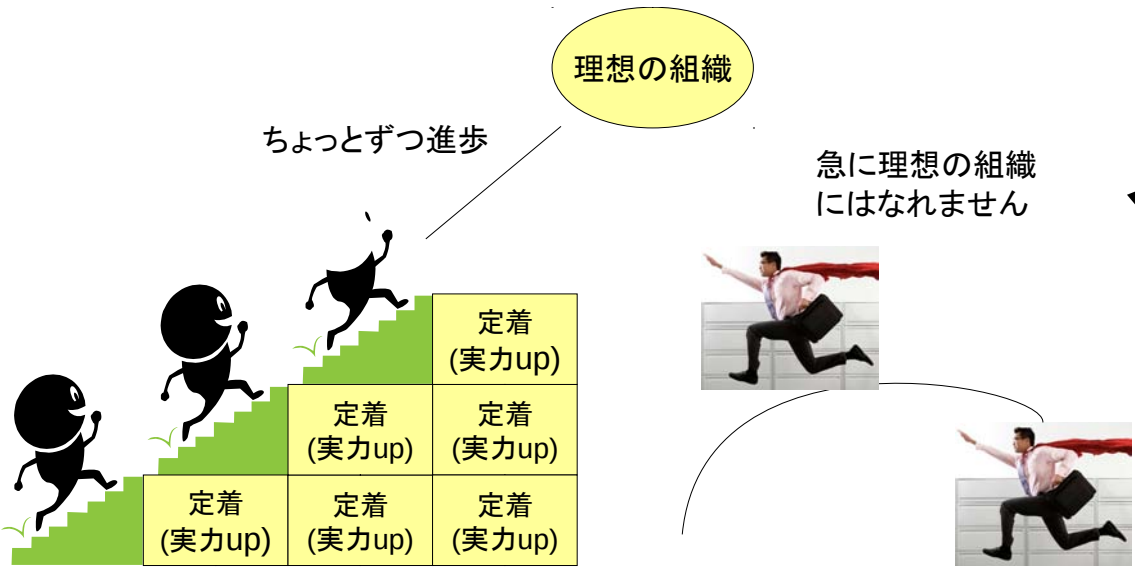
さらに、次回の改善効果を確認する会議(「振り返り会」と呼びます )で①②③④を繰り返し行いました。ちなみに、よくプロジェクトの終わりにこの種の会議を開くときに「反省会」と呼んでいる組織がありますが、これだと悪いことをしたので反省するようなイメージとなります。前向きな気持ちにするため「振り返り会」という呼び方の方が好ま

しいと思います。

保守開発は 2 ヶ月に 1 度、製品のバージョンアップ版をリリースする開発サイクルとなっており、このサイクルで①②③④を実行することにしました。なお、①②③④のサイクルを回し続けることをここでは PDCA サイクルと呼ぶことにします。

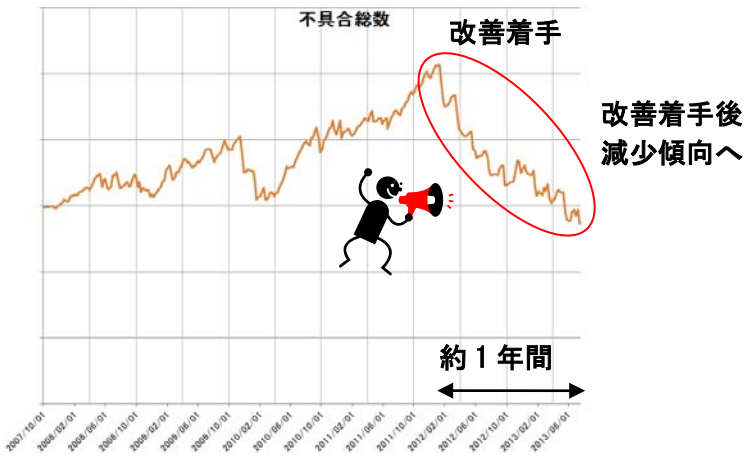
ただし、開発チームに「PDCA サイクルによるプロセス改善を実施する」と宣言すると、変化を嫌うメンバから活動への反対意見がでることが予測されました。現状を変えたくないという意見は一般的にどの組織にも存在します。私の経験では「本に書いてある PDCA は〇〇とか△△をやれって書いているけど、そんなことまでできません。」と言って、本に書いてある状態は負担が大きいから取り組みはやめましょうという反論をしてきます。
しかしながら、今が 0 点とすれば、30 点でも 50 点でも十分に改善効果はあるので、まずはそこを目指します。そのため、PDCA という単語はプロジェクトマネージャの頭の中だけにとどめて口外せず、効果を実感できることを優先させました。

本取組のイメージを下図に示します。プロジェクトマネージャは図の左の階段の段差を、開発チームのその時点の実力に応じて適切に設定して、チームの育成を主導する必要があります。
図の右側のように、急に「本で読んだ理想の組織」がやっていることを全部やれと言っても、挫折する可能性が高いです。



(6)変更後の状態や改善効果

①～④のサイクルを 1 年以上繰り返して改善に取り組み、不具合総数は、同じ予算でありながら減少傾向に転じました。




改善活動を 1 年間以上継続して、増やしてきた取組事項の一覧です。状態が継続のものは今でも実施しています。効果が十分でないと判断したものは一旦取組を外しました。

| No. | 取り組み内容 | 状態 |
|---|---|---|
| 1 | 会議開催時刻、納品時刻を守る | 継続 |
| 2 | 議事録を発言録ではなく ToDo リストにする | 継続 |
| 3 | 設計前に該当する機能の仕様書を確認し、仕様書の整備状況を確認する | 停止 |
| 4 | レビュー前に仕様を 2 名以上でチェックを行う | 継続 |
| 5 | リリース 1 週間前に、出荷 CD を確定させる | 継続 |
| 6 | チーム内で CI をしてからレビューに臨む<br>→（変更）レビューで規模が大きいと判断したものは追加で、<br>相互 CI を行い、ソースコードに 2 名以上で責任を持つ。 | 変更 |
| 7 | ソース以外のファイルが CD 出荷から漏れないように機械的に出荷物の差分チェックを行う | 継続 |
| 8 | テスト NG 修正時に設計書の修正も同期する | 継続 |
| 9 | 不具合修正の設計書には、不具合の画面ダンプを掲載し、修正の before/after の説明を記載する | 継続 |
| 10 | リリース日は お昼 12:00 に出荷を完了させる | 継続 |

(7)改善活動の妥当性確認

一連の取り組みにより、下記の効果が得られました。

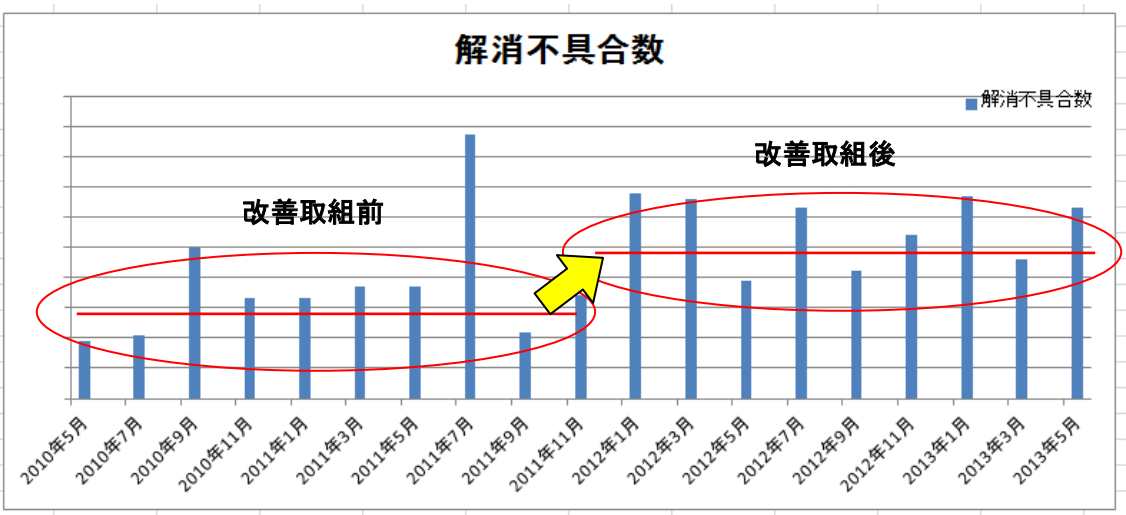
① 1年後にチームが当たり前のように身に着けた改善内容を列挙してみると、開発チームが同一コスト、同一期間に解消する不具合数が向上したことが分かりました。

② 改善のPDCAサイクルを回すことが定着し､チームメンバも振り返り会で積極的に次の改善案を発言する風土が出来ました。

③ 改善効果を評価する際に開発効率を計測する必要があったため、開発効率の定量化を行いました。その結果、次回の保守サイクルで解消可能な不具合数の個数を予測できるようになり、サイクル内でチームが達成すべきゴールが明確になりました。
これは開発チームメンバのメンタル面でも良い効果がありました。

これらの効果から､開発チームにはプロセス改善のPDCAサイクルが身につき､実際に開発効率の向上を得られていると評価することができます。

最後に2ヶ月サイクルの保守サイクルについて、ほぼ同じ投入コストに対しての不具合解消数の遷移を示します。改善取組後の方が開発効率が向上していることが分かります。




以上。

2B3「SPI 活動の活性化と組織定着への取り組み」寺野下昌秀(東芝テック)

〈タイトル〉:　　SPI 活動の活性化と組織定着への取り組み

〈サブタイトル〉:　　成果を分かち合い仲間意識を得る改善事例

〈発表者〉
氏名（ふりがな）: 寺野下 昌秀（てらのした　まさひで）
所属：東芝テック株式会社 グローバルソリューション事業本部 ソフトウェア技術部
〈共同執筆者〉
氏名（ふりがな）: 松田 陽二（まつだ　ようじ）
所属：東芝テック株式会社 グローバルソリューション事業本部 ソフトウェア技術部

〈要旨〉
長年 SPI 活動をしてきましたが、現場や組織への浸透が難しく、継続的な活動がしづらい状況でした。また、2006 年当時の現場は、SPI 活動に懐疑的であり、仲間意識も薄い状況でもありました。そこで、SPI 活動の取り組み方を見直し、徹底的な現場への貢献と成果を求める事、更に、活動の目的について、都度話すようにしました。
その結果、SPI 活動の共通認識と必要性についての理解を得ると共に、組織活動としての定着にまで至った、SPI 活動の改善事例を紹介します。

〈キーワード〉
SPI 活動の組織定着
開発現場の改善意識
SPI 活動の失敗
SPI 活動の考え方や活動範囲
開発現場が理解しやすいプロセス領域の定義
SPI 活動の目的

〈想定する聴衆〉
SEPG 初心者、SPI 活動中の SEPG の方、組織長の方

〈適用状況〉
☑多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
☑汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

弊社では、2003 年から SPI 活動を本格化してきましたが、開発現場への改善意識が浸透せず、積極的に活動ができていない状況でした。『現場の改善が無く向上しない企業は、衰退する』。そのような危機感から、SPI 活動を組織的な活動にする為に、2008 年に SPI 活動の取り組み方を改めました。その結果、SPI 活動が組織全体に認識され、考えが浸透し、2013 年には組織活動にまで至りました。今回、その経緯について発表します。

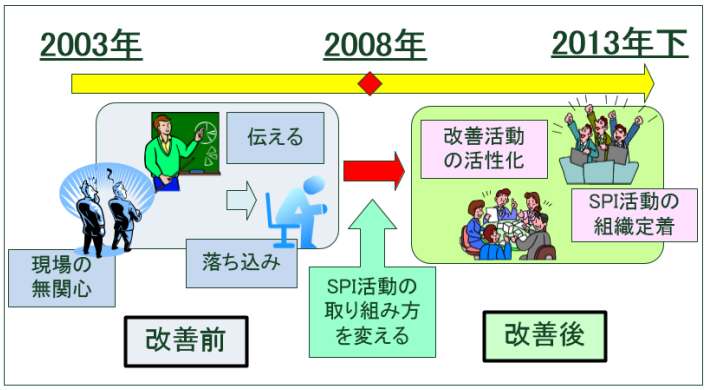



図 1 SPI 活動の取り組みの改善前と改善後

(2)改善前の状態

改善前の SPI 活動は、CMM/CMMI 等、新しく定義された外部団体の情報や、他社事例の収集と教育、規定作りが主な活動でした。しかし、組織や開発現場の問題解決結びついておらず、それらの規定や情報が、活用されるまでには至っていませんでした。

また、現場との整合を取らずに新しい規定を導入しようとした為、現場の負担が増え、SPI 活動に批判的な意見が生じている状況でした。

そこで、SPI 活動が批判され、改善活動が後ろ向きに感じられている状況を変える為にも、担当者と問題意識を共有し、改善効果を証明し、改善意識を浸透させていきたいと考えました。

(3)改善前の状態をもたらした原因（因果関係）

改善前の SPI 活動は、上段に構えて活動しており、積極的に現場を見ていなかった事が主な原因でした。

例えば、現場の声から課題を集めるための次のような行動です。

「ヒアリングをするので、各リーダ参加して意見を述べて下さい」

もし、現場の課題を集めるのであれば、プロジェクト進捗会議や課題対策会議、リリース認定会議に参加し、挙がっている課題を収集し現場の状況を把握する事が、現場の声（状況）となります。

「集まってもらって聞く」のと「現場に入って状況を見て・聞いて・知る」では、行動一つとっても主体性が違います。

また、挙がった課題に対しても、一部の項目に対して、他社事例の紹介、教育や用語説明

といった対応のみでした。その結果、開発者から、共に良い方向に向けて積極的に汗をかいていないと思われたのも、SPI 活動が浸透できなかった原因でした。

そして、SPI 活動メンバは、開発を成功させようとする志を持った仲間と思われていない状況を生み出していました。

(4) 計画した変更内容

SPI 活動の考え方や活動範囲を、開発現場の課題への迅速な貢献に切り替え、以下を柱として実行する事にしました。

- CMMI や PMBOOK の本質を理解し､開発現場で使われている用語に置き換えて利用する。事例を紹介する場合、自部門の用語や風土を考慮し、何が違うかを明確にした上で説明する。
  (例： 図 2 社内で利用する SPI プロセス領域)
- 半期毎に、重点的に改善するプロセス領域を定め、数年で一巡する。
- プロジェクトに飛び込み、積極的に会議へ参加し担当者と話をする事で、状況や課題を把握する。その後、組織長・プロジェクトリーダ・チームリーダ・担当者の役割を意識し、課題となっている原因を見つける。
- 課題に対し、“ムダ・ムラ・ムリ”を無くす為に、必ず『プロセス』･『スキル』･『手段(ツール)』の三つを考慮した改善策を提案し実行する。
- 課題が大きく複雑な場合でも、周りの人を巻き込んで一緒になって改善策を考え実行する。
- 「できない、難しい」を、「できる、やりやすくなった」と、現場が変わる事を追求する。

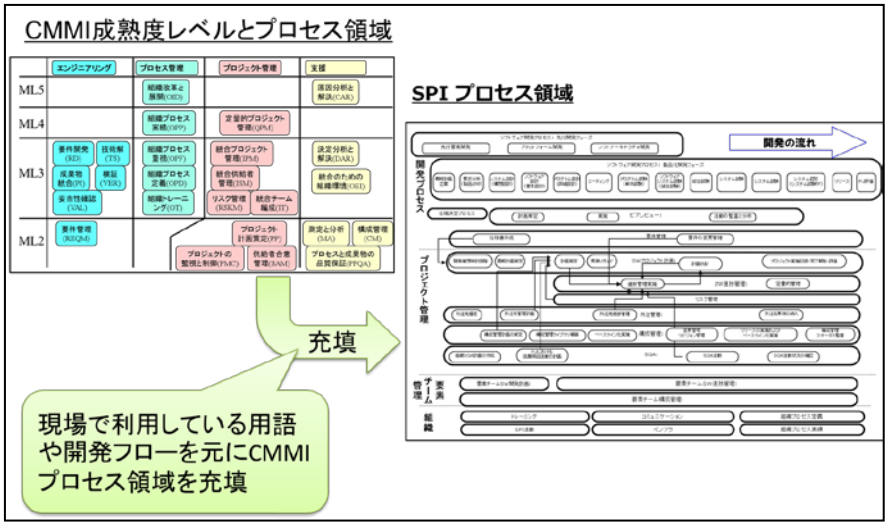



図 2 社内で利用する SPI プロセス領域

(5)変更の実現方法

6 年間で プロセス領域を一巡し、約 40 件の改善を実施してきました。
図 3 組織・PJ レベルの改善事例(’08-’13)

今回、3 件の事例を紹介します。

・ **<u>海外拠点との成果物品質管理</u>**

**背景**：

海外拠点のリーダに、設計書とソースコードの承認まで委譲したプロジェクト体制が発足。

**課題**：

海外拠点の開発者が、国内と同等の品質を確保する、成果物管理を実施する必要が発生。しかし現状、成果物のバージョン管理のみ実施しており、変更管理や承認ルーチンは暗黙知で不明確である。

**解決**：

**プロセス**：承認までに必要なプロセスとプロシージャを CM Plan として定義。
**ツール**：CM Plan のプロセスを管理する為、プロセス管理までを含む構成管理ツールを構築。
**スキル**：全開発メンバへ CM Plan とツールの説明会を実施し、品質意識の向上を図る。

**結果**：

品質確保に必要なプロセスを明確にし、ツールでも管理できるようにした。
これにより、海外拠点間での成果物が何時・誰が登録し、誰が承認したかまでのトレースができるようになり管理されている状態となった。また、今後、品質問題が発生し、開発工程や承認工程における対策が生じた場合、このプロセスを改善し品質向上図る共通のプロセス・ベースラインを作成した。

**工夫・苦労した点**：

**苦労した点**：プロセスを明確化し定義する為に、現場の声を聞いて 個々の変更管理と成果物のプロセスを定義するのが苦労した。変更管理プロセス：８カテゴリ、成果物の管理プロセス：７カテゴリを定義。
**工夫点**：やるべき点とやりたい点を分類して記載。今はしてないが、やりたい事は、説明やコメントにしてメリハリをつけた。また、現状でも目的が不明な箇所を見つけムダが生じないようにした。そして、自分が担当者としてプロセスをシミュレーションし、納得するまで検討を進めた。

・ **<u>評価結果の報告と修正による品質見込み計算のリアルタイム化</u>**

**背景**：

開発終盤、日本と海外拠点の評価サイトで、システム試験を分担して実施。

**課題**：

複数拠点の評価完了から集計・レポートにかかる時間が、データの整合やマージ作業により、1 拠点より時間がかかる状況であった。その為、夕方の設計へのフィードバック会議に間に合わず翌日の報告となり、修正検討が遅延する状況

でもあった。
また、効率的に品質を上げる為の優先項目を判断する際、修正項目と品質改善効果に対するシミュレーション、確認に手間取り、判断に時間がかかっていた。

**解決：**

**プロセス**：どの役割の人が、何時、何の目的で、何を見るべきかを明確にした。また、それにはどのような情報が必要かを定義した。
**ツール**： 評価結果等の集計を全自動化。リアルタイムに評価結果と修正結果の見込み品質が出力できるようにDBと,BI環境を構築。
**スキル**：関係者へレポート情報の活用方法とツールの利用方法について説明会を実施。

**結果：**

リアルタイムに評価結果を集計できるようにした結果、評価途中や評価完了直後に設計へのフィードバック会議が開催可能となり、時間的ロスが無くなった。
また、試験結果と予定修正項目から、すぐに見込み品質をシミュレーションする事で、迅速な投入判断ができるようになった。
担当者も同じ情報を取得し見てもらう事から、品質意識の向上ができた事と、試験領域と評価シートの情報から不具合の有効な再現方法の取得ができるようになった為、調査時間の短縮につながった。

**工夫・苦労した点**：

**苦労した点**： 誰が何の責任でデータを見ているかが不明確だった為、これを明確にしていった点。また、DB設計では、評価シートや評価結果、不具合レポート、投入情報等、データの結合を考慮していなかったデータモデルに対して、結合KEYを決定する作業が苦労した。
**工夫点**：限られた時間の中で確認と予測、判断のプロセスやツールを構築する必要があった。その為、会議中に分析やシミュレーションができる様、使い勝手の良いBIツールを活用する事で、担当者がすぐに利用できる様工夫した。
また、「必要な情報は開発状況に応じて変化する」という事を念頭に、定常の情報だけでなく、プロジェクトの状況変化で必要となる情報について担当者と話し合い、情報の収集と分析を進めていった。

・ **<u>拠点間の情報伝達向上</u>**

**背景：**

海外拠点へ依頼する工程が広がり、情報量の増加や、設計情報など的確に伝える必要もあり、情報伝達の質に対する要求も高くなってきた。

**課題：**

電話会議やテレビ会議が主な通信手段の為、説明者がどこを説明しているかが分からなくなり、時間内に正しく設計情報が伝わらない場合がある。Face to Faceも、頻繁に行うのは難しい状況であった。
また、担当者がツールを提案しても、説明不足やセキュリティー上の理由などにより、ツール導入が認められていない状態であった。

**解決：**

**プロセス**：どのような情報をやり取りする場合に、どのインフラを用いるべきかの運用を定義。
**ツール**：IT部門と共に、会社が認めるツールを調査し導入。
**スキル**：情報伝達の向上が作業効率を上げる事を理解して貰う為、事例やツー

ルの利用方法、運用プロセスに関して説明会を実施。

**結果**：

会社が認める NetMeeting のツールを導入し、会議拠点と画面共有した会議が可能となった。その結果、情報伝達の量と質が向上した。

例えば、設計の修正や質問時、複雑な情報や不明瞭な情報をやり取りするが、画面共有する事により、タイムリーに且つ的確に情報伝達する事が可能となり、情報欠落や認識違いを防止できるようになった。

**工夫・苦労した点**：

**苦労した点**： 運用形態を検討する際、会議の完了時間が決めづらい、不具合調査や質問など、運用への落とし込みに苦労した。

運用の展開において、全担当者が対象であった為、説明会の回数も多くなり体力的にきつかった。

**工夫点**：導入には部門を跨いだ調整が必要な為、具体的な課題や解決手段を丁寧に説明し理解して頂けるようにした。

また、積極的に行動して頂く為にも、一緒に製品を開発しているという仲間意識を持って頂ける様に心がけた。その為、通常、上長からの依頼やメールで対応する案件だが、直接、各部門の担当者と会って説明や検討を進めた。

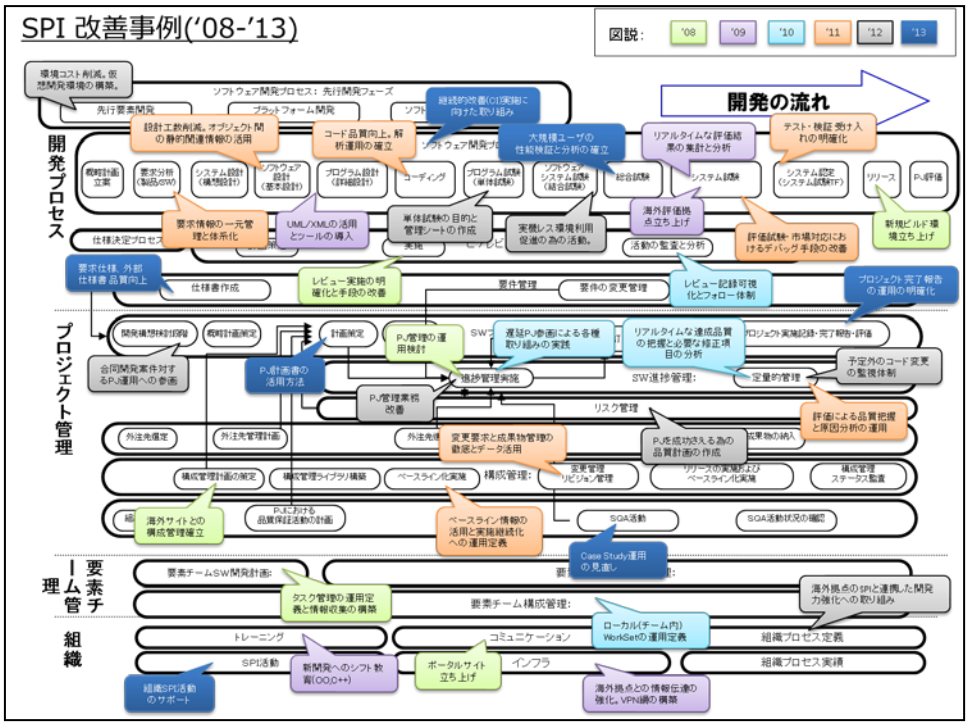



図 3 組織・PJ レベルの改善事例(’08-’13)

(6)変更後の状態や改善効果

現場の成果を徹底し、製品リリースやそれに伴う変革を実現してきた結果、全プロセス領域に対し改善効果が蓄積され、開発現場や組織は、SPI 活動を意識するに至りました。

実際、改善前の現場は、プロセスでソフト品質が改善されると思っていなかった様です。

それが、効率的に成果物を生産し、品質を維持する為には「プロセス」が重要であるという考え方が一般的になりました。ルーチンになっている業務でも、作業の「目的」を

意識し、状況の変化に応じて「あるべき姿(プロセス)」を考えるようになりました。

製品開発の QCD を確保する為に、開発者と共に様々な改善を実施し、共に成果を分かち合う事ができた結果と思いますが、開発現場から SPI 活動がムダなモノであるという批判的な意見は徐々に聞こえなくなり、2012 年には、むしろもっとしっかりすべきだという意見に変わりました。

(7)改善活動の妥当性確認

改善前、SPI 活動の言う『改善』を進める理由について、開発現場ではあまり納得されていませんでした。
しかし、今では、『事業計画を確実に達成する為に、新しい取り決めや既存のやり方を向上させ、目標に向けて担当者が実行できる状態にする』ということが必要である旨、開発現場でも納得しています。
その結果、改善前では SEPG という集められた担当者が改善実施する体制でしたが、改善の結果、組織長がプロセス領域の責任者となり、事業計画から、何を『改善』すべきかを検討し実施責任を担うまでに定着してきました。
各責任者は、事業計画を SPI 活動の BSC へ展開し、目標の KGI/KPI を達成するため改善を推進しています。

今後、さらになる開発案件の増加や国際企業組織への展開、Agile 開発, IPD(Integrated Product Development :IBM) 等 新定義の開発工程による見直しや改革が発生します。
しかし、『事業目標を達成する為にやるべき事』を常に考え行動する事で、明確な『目的』と強い『改善』意思を持った組織を持続させ成熟していきたいと思います。

2C1「経験者採用で、それぞれのメンバが独自のやり方や経験・文化的背景を持つベンチャー企業での、作業標準の策定と適用について」泉友弘(ＮＴＴデータ)

Ａ社は金融サービスを提供するベンチャー企業である。サービス提供先の拡大に伴い、経験者採用で開発メンバの増員を行なっているが、それぞれのメンバが独自のやり方や経験・文化的背景を持ち、作業の進め方が属人化しており、共有されていない。また、Ａ社の作業標準は開発実態に合わず利用されてない。
そのためＡ社では、要件定義工程・設計工程での作業内容の不統一による品質不良や進捗遅れ、商用システムでの故障が大きな経営課題となっていたが、ベンチャー企業として成長途上でサービス範囲の拡大中にあるため、開発メンバは恒常的に複数のプロジェクトを抱え、また、1 つのプロジェクトが終了するとすぐに次のプロジェクトが始まるなど、日常の開発だけでも多忙であることに加え、商用システム故障等のお客様クレーム対応に忙殺され、プロセス改善を行なう意識や余力が持てない状況であった。
Ａ社社長は、Ａ社の成長の次のステップとして、システム品質やサービス品質の確保など組織的な成熟度を上げることに注力するべき時期であると判断し、会社方針として取り組んできた。
私たちは、Ａ社の要請を受け、サンプルのプロジェクトに対してアセスメントを行ない、経営層・スタッフ・開発メンバ・営業メンバを巻き込み、改善対象を“要件定義書作成プロセス”として、ワークショップを実施した。

〈タイトル〉:
経験者採用で、それぞれのメンバが独自のやり方や経験・文化的背景を持つベンチャー企業での、作業標準の策定と適用について
〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 泉　友弘（いずみ　ともひろ）
所属：ＮＴＴデータ　品質保証部　品質マネジメント担当
〈共同執筆者〉
氏名（ふりがな）: 渡辺　清孝（わたなべ　きよたか）
所属：ＮＴＴデータ　監査部　監査担当
氏名（ふりがな）: 大鶴　英佑（おおつる　えいすけ）
所属：ＮＴＴデータ　品質保証部　品質マネジメント担当

〈要旨〉開発メンバが独自のやり方や経験・文化的背景を持ち、作業の進め方が属人化しており、共有されておらず、また、作業標準がない、あるいはあっても利用されない組織において、開発メンバが主体となり、作業標準を策定・展開するための、組織トップによる動機づけ、策定・展開の体制構築、メンバの意識合わせなどを含めた、ワークショップの進め方の実績を報告し、情報共有を図り、さらに良い進め方などの討論を行なう。

〈キーワード〉
作業標準、標準化、標準化プロセス、属人化、経験者採用、ベンチャー、動機づけ

〈想定する聴衆〉
標準化に取り組みたい組織の方、標準化プロセスを適用したい組織の方、ベンチャー企業でプロセス定義したい方、メンバに共通認識がない組織で標準化に取り組みたい方

〈適用状況〉

□多用されている段階、■適用できる段階あるいは初めて適用する段階

□適用するにはさらに検討を必要とする、□着想の段階

□その他（　　）

〈適用可能性に関する制限〉

□汎用性がある、

■類似プロジェクトにも適用可：具体的な類似点（作業標準がない、あるいはあっても利用されない組織）

□自プロジェクトのみ

〈発表内容〉

(1)背景

Ａ社はベンチャー企業で、開発メンバの多くは経験者採用であり、それぞれのメンバが独自のやり方や経験・文化的背景を持ち、作業の進め方が属人化している。また、Ａ社の作業標準は開発実態に合わず利用されてない。そのため、要件定義・設計工程での作業内容の意識不統一による品質不良や進捗遅れ、商用システムでの故障が大きな経営課題となっていた。

Ａ社社長も、サービス提供先が一定程度に拡大したので、Ａ社の成長の次のステップとして、システム品質やサービス品質の確保など組織的な成熟度を上げることに注力するべき時期であると判断し、これまでも様々な対策を講じてきたが、十分な成果は出ていなかった。

当社がＡ社に出資している関係があり、品質保証部・品質マネジメント担当が、品質向上施策等の支援要請を受け、対応を行なった。

(2)改善前の状態

開発のためのＣＭＭＩの成熟度レベル 3 に含まれるプラクティスを基にしたアセスメントチェックリスト（計 108 項目）を作成し、サンプルプロジェクトを選定し実施した。その結果、要件定義で実施するべき内容・詳細度等が不明確である、仕様の合意・変更管理の段取りが不明確であるなどが、特に弱みであることが明らかになった。要件定義工程などの上流工程での品質向上は、後工程への品質・進捗への改善影響が大きいので、“要件定義書作成プロセス”の改善をテーマとした。

(3)改善前の状態をもたらした原因（因果関係）

作業標準がないまま要件定義を行なうため、決めることが属人化し、共有されない。そのため、意識不統一による品質不良や進捗遅れ、商用システムの故障等が発生し、その対応に忙殺されていた。また、1 人の開発者が複数のプロジェクトを並行して担当するなど、長時間労働が常態化した余裕のない状況で開発を行なっている。改善が必要であるという認識はあるものの、現在の開発を止める・遅らせることはできないため、多忙であることを理由にアクションを打たないまま、品質不良・進捗遅れ、システム故障等が繰り返し発生するなどの悪循環に陥っていた。この悪循環を打開するためには、経営層から担当者まで全社で意識を合わせて改善に取り組む必要があるが、成長途上の会社であり、また、致命的なインシデントが発生していなかったために、改善に踏み出せずにいた。

(4)計画した変更内容

ワークショップで、“要件定義書作成プロセス”を改善するためのツールとして、①要件定義の調整事項フロー　②要件定義書のひな型・サンプル、および③チェックリストを作成し、実際のプロジェクトに適用し、さらにブラッシュアップを図ることを計画した。

前述のように、Ａ社開発メンバは非常に多忙であるため、やらされ感を持たず自律的に検討のできる問題意識の高いメンバを選定した。また、Ａ社社長による動機づけも十分な時間を取り、メンバとディスカッションするなど、納得感を持って進めてもらえるよう計画した。

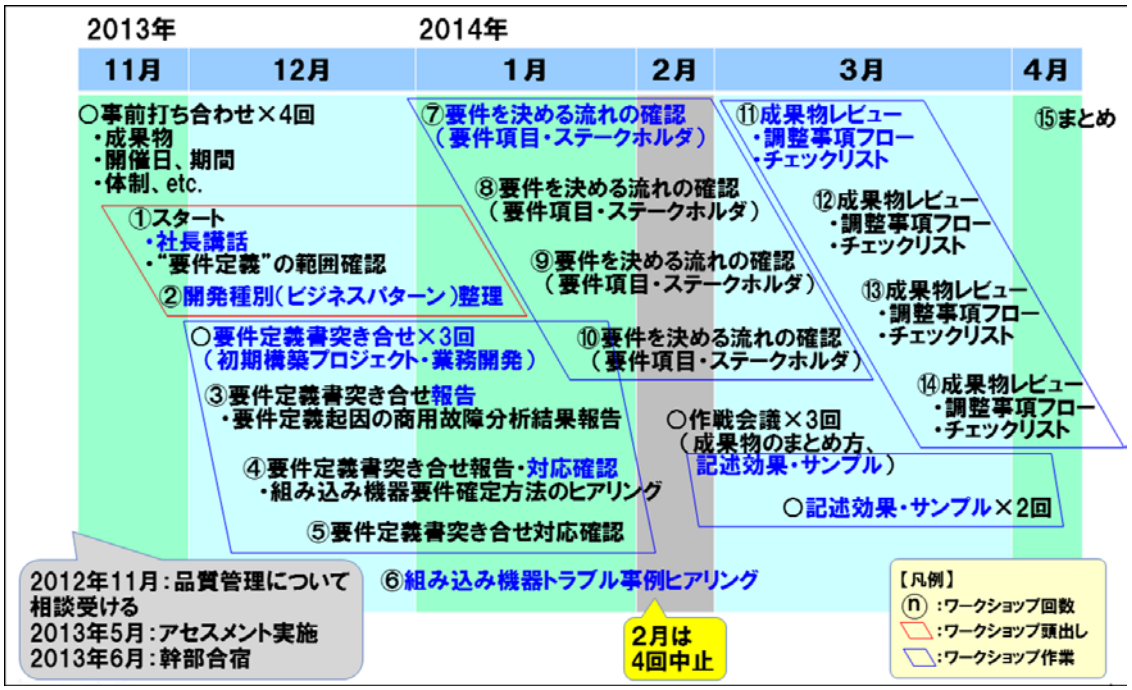
(5)変更の実現方法

【Ａ社に特徴的な難しさ】

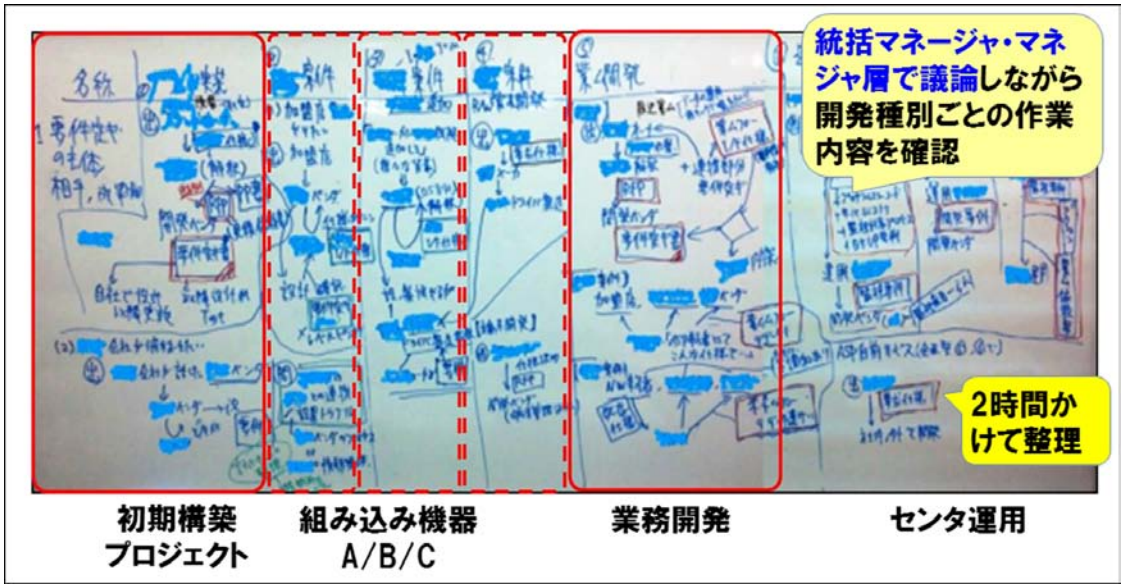
・開発メンバが経験者採用のため、それぞれのやり方を持っている。
・それらを取り持つ作業標準は存在しない。
・要件定義と言っても、人により考える範囲が異なる。
・開発メンバが多忙のため、情報やノウハウの共有、記録ができず、属人化している。
・まずプログラムを作ることを優先し、品質について考える時間がない。
・全体の進め方をきちんと理解しているのは幹部のキーマン 1 名のみである。
・そのキーマンも作業をプロセスとして考え実施することの慣れていないため、プロセスの可視化が容易ではない。
・1 つのサービスを提供するために、6 本程度のサブプロジェクトが相互に調整しながら進める必要があり、また、ステークホルダが約 20 と非常に多いなど、要件定義項目・流れの構造が複雑である。

【Ａ社メンバへの動機付け】
・Ａ社幹部会議（Ａ社のビジョンを実現するための活動について議論）に参加し、Ａ社課題を共有し、幹部への意識付けを行なった。
・ワークショップメンバには、Ａ社社長と品質についての議論を行ない、改善の取り組みの意識合わせ・動機付けを行ない、また、Ａ社の今後の進むべき方向について会社や所長の方針についてレクチャを行なった。

【スケジュール】

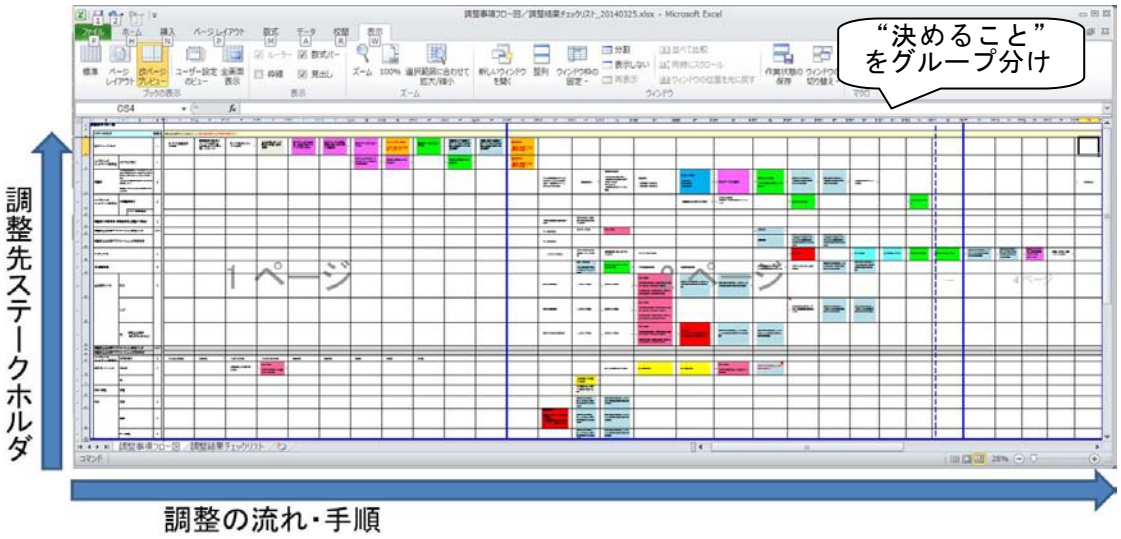



【サブプロジェクトの関連性の可視化】
・キーマンも作業をプロセスとして考え実施することの慣れていないため、プロセスの可視化が容易ではなかった。成果物の事例（ゴールのイメージ）を示したにもかかわらず、進め方を理解し、絵に描くに至るまでに 2 週間を要した。




【要件項目とそれを調整するステークホルダとの対応付けと流れの一覧化】

・1 つのサービスを提供するために、6 本程度のサブプロジェクトが相互に調整しながら進める必要があり。また、ステークホルダが約 20 と非常に多いことから、要件項目とそれを調整するステークホルダとの対応付けと流れの一覧を作成した。作成のポイントは、キーマンを集め、その頭の中にある作業の流れを丹念に洗い出し、可視化し、全員で確認しながら進めることにあった。いつ・誰と・何を調整するべきか 1 つ 1 つ確認しながらヒアリングを行ない、4 週間を要した。




・標準的な要件定義書の調査を行ない、それを元に要件定義書のひな型・記述サンプルの作成

・チェックリストの作成

を行なった。

【今後の展開】
今回ワークショップでの当社支援が終了後も、Ａ社が自律的に改善活動を実施できるよう、振り返りを行ない、また、今回の改善活動の進め方を記録に残し共有する取り組みを行なった。

(6)変更後の状態や改善効果
①2014 年 6 月からの開発プロジェクトの要件定義に成果物（調整事項フロー・調整結果チェックリスト）を適用し、項目に漏れがなく有効であることを確認した。調整事項フロー図を、より初心者でも使えるよう確認内容を具体的にするし、成果物相互の項目の対応付けと、その粒度を合わせるよう、改善を図っている。
②経験者採用で属人化した開発を行なってきたメンバが、プロセスの考えを取り入れ、進め方を共有して考えるようになった。幹部キーマンからも、「混沌としていた要件定義の進め方を、一度立ち止まり、プロセスとして体系化できたのは大きな前進である」旨のご意見を頂いている。また、ワークショップの最終回で実施したＫＰＴでは、開発のキーマンから「調整事項が明確になり、早目に調整することで品質向上にも寄与すると思われる」とのご意見を頂いている。
Ａ社では、今回のワークショップの進め方を適用し、新たなプロセス改善の取り組みに着手している。

(7)改善活動の妥当性確認
現在、“変更後の状態や改善効果”①で確認中であるので、発表時に報告する。
【妥当性】
・日常の開発だけでも非常に多忙であることに加え、様々な問題の対応に忙殺され、プロセス改善を行なう意識や余力がない状況のＡ社メンバに、一度立ち止まって作業内容の洗い出し・共有・ドキュメント化を行なうことの重要性を納得・理解してもらうことができた。経営層・スタッフ・開発メンバ・営業メンバを巻き込むことの必要性も理解してもらった。
・属人化していた作業手順を、プロセスとして可視化することで、知識・ノウハウが共有でき、品質や進捗の改善に寄与することを理解してもらった。

【反省点】
・アセスメントを実施したプロジェクトと、ワークショップの参加メンバが異なっていた。ワークショップの参加メンバは、要件定義に課題があることは認識しつつも、テーマや進め方について完全に納得感をもって参加しているわけではないことが途中でわかり、あらためて意識合わせや方向性の確認を行なった。ワークショップ開始時にしっかりと目線を合わせておくべきであった。
・Ａ社のリリース対応のための稼働逼迫・商用故障対応のため 2 月のワークショップ（4 回）が中止になり、後半に作業が集中したが、事務局との作業を中心に進めることで乗り切った（トラブルのため改善活動ができない“悪循環”の一例である）。
・“要件定義書作成プロセス”を作成することはできたが、2 月に 4 回ワークショップが中止になったため、完成度が十分でない成果物もあった。Ａ社メンバ主体で精度を上げていく。

(8)教訓
プロセスの可視化や標準化の必要性を認識しながらも、多忙であることを理由に取り組む

ことができずにいる組織・プロジェクトは多い。
しかし、そこで、一度立ち止まって、作業内容の洗い出し・共有・ドキュメント化に踏み出せるかどうかが大きな違いであり、そのために、経営層・スタッフ・開発メンバ・営業メンバを巻き込み、メンバを動機付けしながら進める仕組みを作ることが、重要なポイントである。
標準化に取り組みたい、標準化プロセスを適用したい組織・プロジェクトであれば、取り組み方次第で、プロセス改善を実現することができる。

2C2「ＧＱＭを用いたメトリクス定義と測定・分析システムの構築」伊沢武史(住友電工情報システム)

〈タイトル〉：ＧＱＭを用いたメトリクス定義と測定・分析システムの構築

〈サブタイトル〉：システムサポート業務の測定・分析システムの構築事例紹介

〈発表者〉
氏名（ふりがな）： 伊沢　武史（いざわ　たけし）
所属：住友電工情報システム株式会社 システムソリューション事業本部 第三システム部
〈共同執筆者〉
氏名（ふりがな）：
所属：

〈要旨〉
我流でのメトリクス定義失敗の経験をもとにしたＧＱＭを用いたメトリクス定義と測定・分析システム構築、それによる改善事例を紹介する。ＧＱＭというフレームワークをどのように使ってメトリクス定義したか、どのくらい改善に寄与したかという実践に役立てやすい事例を紹介する。

〈キーワード〉
ＧＱＭ メトリクス ＭＡ 測定と分析 プロセス改善

〈想定する聴衆〉
ＱＣＤ改善担当者など改善活動に興味のある方

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

私が所属する第三システム部でシステムサポート業務のサービスレベル向上を目的としたメトリクス検討ＷＧが発足した。第三システム部は案件の６０％をシステムサポート業務が占めている。だが、システムサポート業務でメトリクス測定は行われておらず、改善活動に結びついていなかった。全社で決められたメトリクスは新規システム構築をターゲットとしたものでシステムサポート業務に適用が難しかったためである。そこで、部門でシステムサポート業務のメトリクスを定義し改善につなげていく活動を開始した。

(2)改善前の状態

ＷＧで我流でのメトリクス検討を始めた。まずは改善要望対応ＬＴ短縮(*1)というゴールを仮置きできた。そこで既存のデータを用いて業務グループ単位で平均ＬＴをグラフ化、分析を行った。しかし、そこから改善に向けてどう取り組めばよいか糸口が見つけられなかった。ＬＴはどうあるべきか、なにがゴールなのか、何を改善すればサービスレベルが向上すると言えるのか、あいまいな議論しかできず発散ばかりで収束できなかった。
各メンバーともどうすればよいのか思い悩み、モチベーションが低下し、ＷＧ活動頻度も下がっていった。
(*1)ＬＴ＝リードタイム(対応完了までにかかる日数)

(3)改善前の状態をもたらした原因（因果関係）

原因は「ゴールを達成した状態」を定量的に定義せず、それに紐づくメトリクス検討をしなかったためである。もし、定義していれば「ゴールを達成した状態」に向けて改善すればよいと分かったはずである。また、仮置きしたゴールももっとサービスレベル向上に紐づく具体化されたゴールとなったはずである。

(4)計画した変更内容

ＧＱＭを使って測定ゴールおよびメトリクスを再検討することとした。

ＧＱＭ(Goal Question Metrics)とはメトリクス計測の枠組みを定義する手法である。以下の３つの層で定義する。

| | |
|---|---|
| Goal 層 | :測定するゴール(目的)を定義する。 |
| Question 層 | :目的を達成した状態を定量的に評価する質問を定義する。 |
| Metrics 層 | :質問に回答できる測定可能な指標を定義する。 |

ＧＱＭには一般的に以下のような効果が挙げられる

効果１．ゴール達成に近づくメトリクス定義可能
効果２．ゴール・メトリクスを双方向に辿れる

効果１が失敗した原因の対策となると考え採用を決定した。

(5)変更の実現方法

ＧＱＭを採用したメトリクス定義を行い、更に測定・分析システムを構築した。

手順１．測定ゴールの決定

測定ゴール(G)をブレストして洗出し、その後５つに集約した。

注意点は以下の２つ

・Gが最終ゴールを詳細化したものになるようブレストテーマで工夫した。

「システムサポート業務のサービスレベルが高い状態とは？」

・担当者に納得できるGとなるよう集約時に担当者にアンケートを実施した。

・レベル感で分類整理することで現在の実力に合わないゴールを除外するとともに、ゴール達成後の次のゴールが分かるようにした。

| | 納期管理している | 約束を守れる(回答納期) | 希望に答えられる(希望納 |
|---|---|---|---|
| 問合せ | 納期管理している | 回答納期が守れる | 希望納期に応えられる |
| 改善要望 | 納期管理している | 回答納期が守れる | 希望納期に応えられる |
| 作業依頼 | 納期管理している | 回答納期が守れる | 希望納期に応えられる |

1

2

図１：測定ゴールマトリックス サンプル

手順２．ＧＱＭツリーを作成

Google で検索すると測定ゴールを達成した状態を定量的に問う質問(Q)とその回答となるメトリクス(M)を定義するとあるが、どのように挙げればよいかわからない。

そこで、以下の手順でＱ、Ｍの検討を行った。

手順２－１．最も端的なＱとその回答となるＭを定義

Ｇ達成状態をもっとも端的に問うＱを定義。次にインディケータ（＝Ｉ ※目に見えるものグラフ、一覧表など)、Ｍの順に定義した。ＩとＩに必要な情報を定義することでＧＱが明確になり、改善に結びつくＭの定義を助ける。

「Ｇ＝改善要望が希望納期に応えられる」としてＱを検討

Ｑ：希望納期に応えられなかった改善要望の件数は？

実際にＩ作成に必要な情報を考えると以下の疑問点が出てくる。

希望納期とはどんな日付か、応えられたとはどんな状態、測定のタイミング、期間

これらを議論しＱを明確化していく

Ｑ：「希望納期＝当月」の改善要望のうち「希望納期＜対応完了日」の件数は？

Ｉが作成できそうな状態まで詳細化すれば、関係者間でＧに対する認識のずれはなくなる。

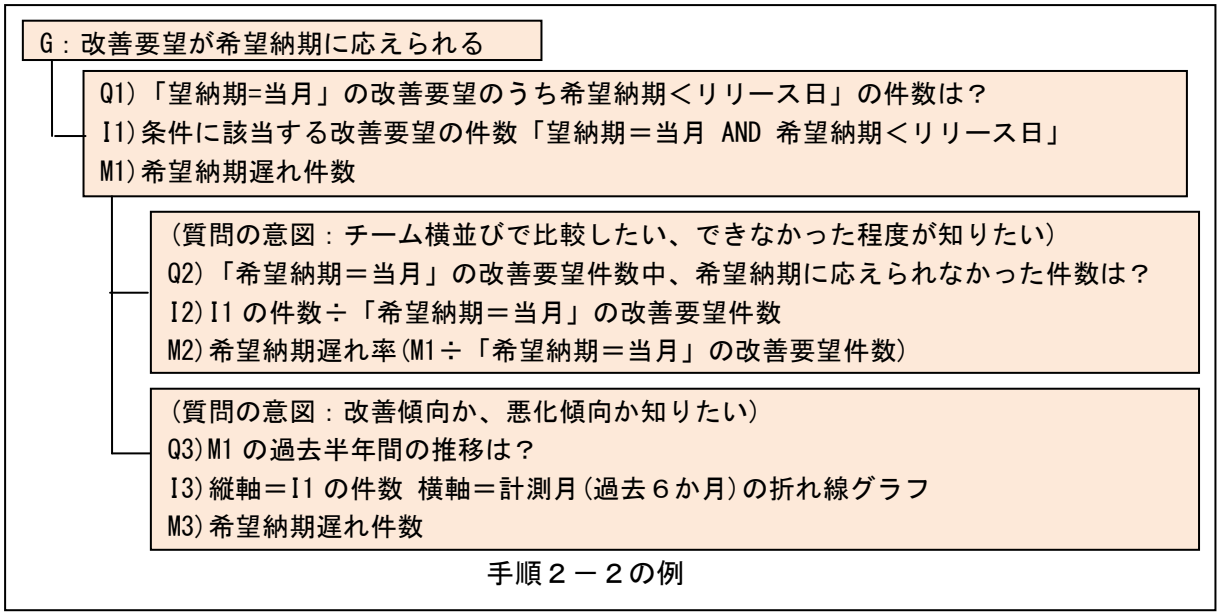
手順２－１の例

手順２－２．Ｇが達成できなかった場合に受けるＱを想定、Ｍを検討

Ｇ未達成の場合、程度、傾向、原因の質問が予想される。

これらをＱとして、Ｉ、Ｍを検討した。

測定結果を分析、利害関係者問共有可能なＩ・Ｍを定義する助けとなる。




手順２－２の例

手順３．定期的に測定するＭを選択

ＭＩを定期報告するものと問題があった場合に見るものに分類した。

定期報告するＭＩは２つのグラフ、表などで表現できる程度にした。

手順２で作成したＩ１、Ｉ２、Ｉ３を１つのグラフでまとめる。

Ｉ１＋Ｉ２＋Ｉ３＝　縦軸：Ｍ２　横軸：計測月の折れ線グラフ

監視対象とするメトリクスは「Ｍ１÷「希望納期＝当月」の改善要望件数」となる

手順３の例

手順４．測定・分析から改善活動につなげる方法の決定

以下事項を決め、キックオフで周知し運用を開始した。

・Ｉを報告書にまとめる仕組みを構築(月１回)
・GQM、測定方法を担当と共有
・担当、管理者層と結果を共有する会議体を設定
・分析し改善活動につなげる仕組みを構築

改善活動につなげる仕組みとして、会議体の場で改善点の議論と改善一覧の共有、状況のトレースを行うこととした。

自チームのプロセスしか知らないため、結果の悪かったチームだけではプロセス改善点を見つけ出すことができず、良いチームに秘訣を聞いてもわからない。

会議の場で悪かったチームのプロセスをヒアリングし、良かったチームとのプロセス差異より改善点を見つけ出すようにした。

(6)変更後の状態や改善効果

2014 年 4 月度から計測開始、以降毎月会議体を開催している。
会議後のＷＧで振り返りを行い、以下の改善効果を確認した。

・ＧＱＭ手法の導入により、参加者全員で合意したメトリクスが確立できた。
・確立したメトリクスによる測定が定着化している。
・収集したデータをグラフ化して開発者にフィードバックできており、メトリクスの運用可能性が確認できた。
・ヒアリングの結果、集計結果が有用であることが確認できた。
・ＧＱＭ手法の導入により、ＷＧ活動の議論が停滞なく進められた。

改善活動による成果としては、今までに２点の改善を実施しメトリクス自体も改善傾向にある。また、すでに達成に近づいているゴールについては上位レベルのメトリクス設定をするため新たなＧＱＭを作成している。

(7)改善活動の妥当性確認

ＧＱＭを用いたことにより、改善活動に結びつくメトリクス定義でき当初問題を解決することができた。よって、ＧＱＭを用いたメトリクス定義手法は有用であると考える。

## 2C3「「サービスのための CMMI」活用事例～最初の一歩～」舟山正憲(NEC ソリューションイノベータ)

<タイトル>：　「サービスのための CMMI」活用事例
～最初の一歩～

<サブタイトル>：　―なし―

<発表者>


氏名（ふりがな）：　舟山正憲（ふなやままさのり）

所属：　NEC ソリューションイノベータ株式会社 技術統括部


<共同執筆者>


氏名（ふりがな）：　伊井真由美（いいまゆみ）

所属：　NEC ソリューションイノベータ株式会社 技術統括部

氏名（ふりがな）：　後藤徳彦（ごとうのりひこ）

所属：　NEC ソリューションイノベータ株式会社 技術統括部


<要旨>


ITIL(V3)をベースに改訂した QMS が現場に浸透できず、早急な改善が必要な状況となった。その際、チェックリストとしても利用可能な「サービスのための CMMI」を採用することを決めた。「サービスのための CMMI」は馴染みがなく、社内で先行組織もないため、独自な仕組み・やり方などを検討し、セルフチェック手順を確立した。チェック結果については、数値などで見える化したことにより、強み・弱みが明確になり、改善のための“道しるべ”として活用することができ、プロセス改善として大きな効果が期待できる状態となった。


<キーワード>


ITIL(V3) SLA　QMS　サービスのための CMMI

セルフチェック　プロセス改善　サービスプロジェクト


<想定する聴衆>

サービス PJ 管理プロセス導入初期組織 SEPG

<適用状況>

□多用されている段階、■適用できる段階あるいは初めて適用する段階

□適用するにはさらに検討を必要とする、□着想の段階

□その他（　　）

<適用可能性に関する制限>

■汎用性がある、

□類似プロジェクトにも適用可：具体的な類似点（　　）

□自プロジェクトのみ

<発表内容>

(1) 背景

昨今のサービス事業の高まりもあり、弊社におけるサービス PJ 数比率も 50%を超える状況となってきた。しかし、開発 PJ と比較すると、「サービス PJ は多種多様であり、監理することは困難である」という先入観から、サービス PJ 管理プロセスとしては現場主体（任せ）の管理プロセスであった。しかし、これからのサービス事業強化の方針は変わらず、サービス PJ 数比率向上傾向は続くことが予想され、サービス PJ 管理プロセス強化の必要性があった。

そこで、サービス PJ を管理・監理するためには何が必要か？ 何を強化しなければならないのか？ などを検討する過程で、様々なフレームワークやモデル さらには NEC グループ標準などを比較し、SLA も含めたサービス PJ 管理プロセスのベストプラクティスが記載されている ITIL(V3)をベースとした強化を QMS に施すことに決定し、2013 年 8 月に QMS（主にサービス PJ 管理プロセス）の大幅改訂を実施した。

(2) 改善前の状態 及び (3) 改善前の状態をもたらした原因（因果関係）

2013 年 8 月 QMS 改訂後、現場からは、「言っていることは理解できるが、具体的に何をやるべきかわからない・わかりにくい」「今のやり方で問題はない」など、監理されることへの拒否反応が発生してしまった。その結果として、改訂後半年後のサービス PJ 管理プロセスの実践度は低く、0%～約 7%という惨憺たる数値であった。

このことから、「サービス PJ 管理プロセスの改善が急務」となった。

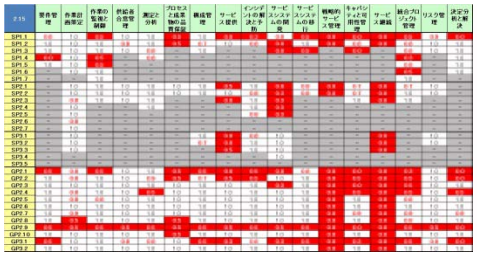
(4) 計画した変更内容

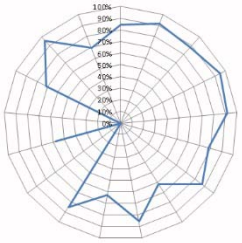
しかし、「何がいけないか？」「どこをどう直せば良いのか？」など、“改善するための道しるべ”がない状態であり、チェックリストとしても利用可能なことから「サービスのための CMMI」を採用することとした。

次に「サービスのための CMMI」を採用すると決めたが、このモデルを先行して実践した組織が社内には存在しなかった。そこで、まずは自分たちで理解した範囲・自分たちの組織あったモデル解釈・実践度合が客観的な数値で理解できるなどを骨子とした**「セルフチェック手順」**を確立した。

(5) 変更の実現方法

その結果、弊社の QMS は「サービスのための CMMI」と比較すると、2.15（組織成熟度 3 に対して、セルフチェック手順に従い独自計算により算出した換算値）という検証結果であることが判明した。さらに、PA 単に確認すると、サービス PJ ライフサイクル部分やサービス設計・構築に関する PA（以下、“サービスエンジニアリング”と称す）に弱みがあり、その詳細としては「記載してある意図が読み取れない」ということも判明した。





表：QMS とプラクティス　　グラフ：同左チャート図

そこで、「サービス PJ 管理プロセス改善」の急先鋒として、サービス PJ ライフサイクル部分の改訂を先行実施することを決めた。さらに、その改善案作成には現場からメンバを募り、「管理標準検討 WG」を発足させ、現場からの意見を取り入れるようにした。これは、今回のサービス PJ ライフサイクル部分が、先の改訂後に現場から寄せられた意見が集中した部分と一致したこともあり、現場からの改善要求とも一致した。そして、2014 年 6 月 「サービス PJ 管理プロセス改善」をメインとした QMS 改訂を実施した。

(6) 変更後の状態や改善効果

今後の予定としては、2014 年 8 月に再度セルフチェック手順に従い「サービスのための CMMI」との再検証を実施する予定であるが、今回の QMS 改訂では、2.22 程度まで向上することを想定しており、SJ14 時には数値が確定する予定である。さらに、現場の実践度も再測定する予定であり、大幅な改善（目標 60%）を期待している。こちらの数値も SJ14 時点では確定する予定である。

その後は、今回の改訂項目として見送った“サービスエンジニアリング”系の強化・確立が必要であると考える。開発には様々な手法やツールなどが存在するが、サービス品質を高める手法やアプローチなどは皆無であり、その中核となる“サービスエンジニアリング”を強化・確立する必要があると認識している。

(7) 改善活動の妥当性確認

今回の一連の活動により、「サービスのための CMMI」を活用した改善プロセス（プロセス改善の PDCA・道しるべ）が確立することができ、サービス事業の高まりへの対応時にも期待できる結果となった。「サービスのための CMMI」に対しては一言で言えば“食わず嫌い”であった。使ってみれば、長年「開発のための CMMI」を使っている我々 SEPG としては馴染み深いモデルであり、継続したプロセス改善に活用可能なアプローチを確立できたことは大きな成果であると考える。

＝以上＝

3A1「基盤方式人財の集約組織における組織活性化事例のご紹介」平井賢仁(ＮＴＴデータ)

〈タイトル〉：基盤方式人財の集約組織における組織活性化事例のご紹介

〈サブタイトル〉：～組織を元気にするための、人財育成施策の取組み～

〈発表者〉
氏名（ふりがな）： 平井 賢仁（ひらい たかひと） 所属：(株)NTTデータ 第一法人事業本部
〈共同執筆者〉
氏名（ふりがな）： 川井 隆志（かわい たかし） 所属：(株)NTTデータ 第二法人事業本部
氏名（ふりがな）： 川手 元（かわて はじめ） 所属：(株)NTTデータ 第二法人事業本部
氏名（ふりがな）： 中村 佳子（なかむら よしこ） 所属：(株)NTTデータ 第三法人事業本部
氏名（ふりがな）： 和田 尚子（わだ なおこ） 所属：(株)NTTデータ 第三法人事業本部

〈要旨〉

- ✓ 組織再編にて、筆者の所属組織は、E-IT カンパニーの基盤方式人財を集約した約 180 名体制となったが、中長期的な将来を見据えると、基盤方式リーダが不足しており、また、組織メンバの繋がりが弱く、集約組織としてのメリットを十分に活かせる状態ではなかった。
- ✓ そこで、組織活性化および組織全体の底上げを図るため、”元気で風通しのよい組織を‼”をスローガンに、組織長や管理職層を巻き込みながら、若手社員～中堅社員をメインターゲットとして実施した、さまざまな施策（制度、トレーニング等）ついて紹介する。

〈キーワード〉

- ✓ 組織活性化、組織力向上、人材育成、コミュニケーション、モチベーション

〈想定する聴衆〉

- ✓ 人材育成に携わる方、組織的なプロセス改善に携わる方

〈適用状況〉

□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（ ）

〈適用可能性に関する制限〉

■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（ ）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

- ✓ 当社には、“カンパニー”と呼ばれる 3 つの大きな事業形態がある。著者らは法人分野のお客様を対象とした“E-IT カンパニー（以降、E-IT と略記）”に属している。
- ✓ 組織再編にて、筆者の組織は、E-IT の基盤方式人財を集約した約 180 名となったが、中長期的な将来(2～5 年後)を見据えると、基盤方式リーダが不足しており、リーダ育成が必要であった。
- ✓ また、組織メンバ間のつながりが弱く、成果物やノウハウの共有・展開が不十分であった。

- ✓ そこで、組織活性化および組織全体の底上げを図るため、基盤方式人財育成を中心とした、“風通しのよい組織風土醸成”の仕組みを導入した。
- ✓ なお、今回のテーマは、基盤方式人財の集約組織における活性化事例であるが、SEPG 人材育成やCMMIモデルに代表されるような組織的なプロセス改善にも適用できると考え、応募した。

(2)改善前の状態

(3)改善前の状態をもたらした原因（因果関係）

- ✓ 基盤方式リーダが不足している
  - ➢ 基盤方式リーダに必要な基盤方式技術力及びマネジメント力をバランス良く強化するための仕組みがない
- ✓ 集約組織としてのメリット(成果物やノウハウの共有、要員流動、人的交流等)が活かせていない
  - ➢ 現場プロジェクト支援が中心であり、自組織への帰属意識やモチベーションが低く、メンバ間の交流・コミュニケーション・やる気を活性化させる、組織横断的な仕組みがない。

(4)計画した変更内容

- ✓ 組織活性化および組織全体の底上げを図るため、”元気で風通しのよい組織を!!”をスローガンに、組織長や管理職層を巻き込みながら、若手社員～中堅社員をメインターゲットとした、さまざまな施策（制度、トレーニング等）を企画・導入した。

(5)変更の実現方法

- ✓ 本取組は、「制度系」に関して４つ、「トレーニング系」に関して３つの合計７つの施策があり、更にこれらを企画検討・実施する際の共通手順を以下のように定義した。
- ✓ [共通手順]（以下①～⑦の共通プロセス）
  - ➢ 企画検討プロセス
    組織の育成方針を基に、必要に応じ関連部署と連携を図りながら、施策の企画検討・計画等を行う。特に E-IT やコーポーレートと重複感がないかが重要。
  - ➢ コミットメントプロセス
    部議(全部長出席の会議)にて、施策内容についてコミットメントを得る。
  - ➢ 振返り＆フィードバック＆情報共有プロセス
    対象者/受講者にアンケートを実施し、集計・分析結果等を関係者にフィードバックするとともに以降の改善のインプットとする。各種成果物やトレーニング教材・開催模様等は、可能な限り部内ポータルやコーポレート DB 等に登録し、横展開を図る。
- ✓ [制度系]

① スキルマップシート【対象：全社員（管理職除く）】
  - ➢ 社内の業務経歴システムでは表現できない基盤方式領域に特化した業務経験や保有スキルを記録するシート。本シートを利用し基盤方式リーダに必要な経験をバランス良く積めるようなロードマップを上司と部下で策定する。

② ローテーション【対象：全社員（部長除く】
  - ➢ 基盤方式リーダとしてのキャリアパスを実現するため、同一ポスト（同じ技術領域、工程等）の従事年数を制限し、目標期間内の技術取得を意識させる。

③ iPro 活動【対象：全社員】
  - ➢ 特に若手・中堅メンバが中心となって、活動テーマ毎に組織横断的なチームを作り、新たな価値や武器を創造したり、支援先のプロジェクト外メンバと交流し視野を広げる等、各メンバのスキルアップを図る。

④ 部内 Award およびベンチャ【対象：全社員】

- 組織メンバへの貢献に感謝の気持ちを込めた、部内表彰制度。表彰の対象は、本業の枠にとらわれず、組織力向上に繋がる全ての活動とする。また日常業務において、横展開に向けたノウハウやアイデアの抽出を意識し、ソリューションやサービス化のスキームを提案するコンテストを開催し､優秀な提案には1人月相当の稼働を与える等、各メンバのスキルアップとモチベーション向上を図る。

✓ ［トレーニング系］

⑤ 基盤方式人財養成講座【対象：入社 6 年目以上社員】

- 基盤方式リーダ育成のため、基盤・方式分野に特化した実践的なトレーニング。主に初級者を対象とした[講義編]と、中級者及び初級者を対象とした[実践編]がある。

⑥ メンタリング【対象：入社 3～5 年目社員】

“リーダになりきれず困っている人を救う”を主眼とし、メンティが希望するメンタ（部課長等）および同世代の参加メンバとの意見交換や Q&A 等を通じて、何かしらの”気づき”を得る。

⑦ 育成社員部内成果発表会【対象：入社 2 年目社員】

E-IT 主催の「成果発表会」において、自己の取組み成果や課題認識、考え方等を、的確にアピールできるよう、発表内容とプレゼンテーションの部内最終的なレビューやリハーサルを行う。

(6)変更後の状態や改善効果

✓ 約 2 年間(2012 年 6 月から 2014 年 3 月)の活動実績と成果は､「表 1 活動実績と成果」の通りで、各施策とも概ね目標は達成し､以下の改善効果を得ることができた。

**表 2 活動実績と成果**

【評価凡例】◎:達成、○:ほぼ達成、－:未評価

| 項番 | 施策名 | 目標[処置限界] | 実績 | 評価 |
|---|---|---|---|---|
| ① | スキルマップシート | 収集率=100% [80%] | 収集率=91%(=135/149) | ○ |
| ② | ローテーション | ローテ率=100% [80%] | FY2013 より本格実施<br>（現在、実績収集中） | － |
| ③ | iPro 活動 | 参加率=50% [40%] | 活動テーマ数=15 テーマ<br>参加率=50.3%(=75/149) | ◎ |
| ④ | 部内 Award & ベンチャ | ベンチャ賞=2 件［1 件］ | 優秀賞=22 活動<br>ベンチャ賞=2 活動 | ◎ |
| ⑤ | 基盤方式人財養成講座 | 参加率=80%[70%] | 開催数=6 回、受講者数=50 名<br>参加率=81.9%(=50/61) | ◎ |
| ⑥ | メンタリング | 参加率=80%[70%] | 開催数=3 回、参加者数=20 名<br>参加率=71.4%(=20/28) | ○ |
| ⑦ | 育成社員部内成果発表会 | 参加率=100%[80%] | 開催数=2 回、参加者数=19 名<br>参加率=100%(=19/19) | ◎ |

✓ 基盤方式リーダ育成の土壌形成

- 主に②ローテーション、⑤基盤方式人財養成講座、⑥メンタリング、⑦育成社員部内成果発表会を通じ、基盤方式リーダに向けてのスキルアップを図ることができた。
- 特に⑤基盤方式人財養成講座では、社内基盤方式人財認定試験において、本講座受講

者が、全社平均の合格率[中級者:58.0%、初級者:71.3%]を上回る、中級者:75%(17 ポイント差)、初級者:73.5%(2.2 ポイント差)の実績を得た。また⑦育成社員部内成果発表会では、E-IT の育成社員成果発表会において、最優秀賞=1 名、優秀賞=2 名、受賞した。

- ✓ 組織活性化の土壌形成
  - ➢ ③iPro 活動、⑥メンタリング、⑦育成社員部内成果発表会への参加等を通じ、組織メンバ間の繋がり・ネットワークができ、また①スキルマップシート、④部内 Award&ベンチャ制度の導入により、自組織への帰属意識やモチベーションが向上した。その結果、組織メンバによる自発的な支援活動報告会が開催され、また情報流通のためのメーリングリスト等も運用されるようになり、ノウハウや成果物等の共有・流通が活発になった。
- ✓ E-IT やコーポレート展開
  - ➢ 副次効果として、施策の幾つかは部外から好評を得ており、特に①スキルマップシート⑤基盤方式人財養成講座⑥メンタリングは E-IT へ、③iPro 活動は他カンパニーのメンバが参加する等コーポレートへも展開されている。

(7)改善活動の妥当性確認

- ✓ 本取組に対し、“組織メンバ自身がどのように感じたか”、がより重要と考え、各施策につき、以下の 3 段階評価でアンケートを実施し、妥当性確認を行った。【回答率 48%=82/171】
  - (A)成果があったと思うか? (成果) [(思う)3 ～1(思わない)]
  - (B)今後も継続した方がよいと思うか?(継続) [(思う)3 ～1(思わない)]

**表 3 アンケート結果**

| 項番 | 施策 | (A)成果 | | (B)継続 | |
|---|---|---|---|---|---|
| | | 部下 | 上司 | 部下 | 上司 |
| ① | スキルマップシート | 1.8 | 1.8 | 2.2 | 2.4 |
| ② | ローテーション | 2.0 | 2.5 | 2.7 | 2.8 |
| ③ | iPro 活動 | 2.4 | 2.1 | 2.5 | 2.3 |
| ④ | 部内 Award & ベンチャ | 1.7 | 1.9 | 2.0 | 2.1 |
| ⑤ | 基盤方式人財養成講座 | 2.6 | 2.8 | 2.6 | 2.9 |
| ⑥ | メンタリング | 2.3 | 2.1 | 2.3 | 2.5 |
| ⑦ | 育成社員部内成果発表会 | 2.3 | 2.4 | 2.3 | 2.6 |

- ✓ 全体傾向
  - ➢ 上司は、全社的な方針や取組みと関連する施策(②ローテーション、⑤基盤方式人財養成講座、⑦育成社員部内成果発表会)について、成果があったとし、今後も継続した方がよいと考えている傾向が、部下よりも高い。
  - ➢ 部下は、主体的に取り組む施策(③iPro 活動、 ⑤基盤方式人財養成講座、⑥メンタリング、⑦育成社員部内成果発表会)に成果があったとし、今後も継続した方がよいと考えている。
- ✓ 今後の課題
  - ➢ 施策に対する期待値は高いが、期待値ほどの成果が出ていないと考えている。
    ⇒施策の成果をフィードバックする仕組みが必要
  - ➢ 施策に参加したいが、業務が繁忙で施策に費やす時間がない、または、支援先や上司の理解が得られにくいと感じていて、参加できない。
    ⇒部下が施策に参加しやすくなるような環境が必要

3A2「高度ソフトウェア専門技術者の育成」上杉卓司(デンソー技研センター)

〈タイトル〉:
高度ソフトウェア専門技術者の育成
〈サブタイトル〉:
P2E コンセプトに基づくカリキュラム開発とその評価
〈発表者〉
氏名（ふりがな）: 上杉 卓司（うえすぎ たくじ）
所属：(株)デンソー技研センター　技術研修部

〈共同執筆者〉
氏名（ふりがな）: 古畑 慶次（こばた けいじ）
所属：(株)デンソー技研センター　技術研修部

氏名（ふりがな）: 足立 久美（あだち ひさよし）
所属：(株)デンソー　電子基盤システム開発部

〈要旨〉
高度ソフトウェア専門技術者の育成は，大規模システム開発を成功に導くためにも非常に重要な課題である．弊社では，2004 年より研修を開講したが，研修生側，研修提供側ともに問題点を抱えていた．これらを P2E（実践，哲学，技術）の視点から分析し，研修を段階的に再構築し，施策の有効性を確認した．
さらに，ここで得られた知見から，今度はゴール指向分析を行い，育成ゴールからトップダウンに展開する P2E コンセプトを確立した．この P2E コンセプトに基づいてカリキュラムを再度検証，改善し，有効性を確認した．

〈キーワード〉
人材育成　ソフトウェア工学　課題解決　リーダーシップ　P2E コンセプト
ゴール分析　Lightning Talk　能動型実践学習

〈想定する聴衆〉
人材育成担当者

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
□汎用性がある、
■類似プロジェクトにも適用可：具体的な類似点（　ソフトウェア人材育成　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

デンソーの技術者育成の仕組みは，基礎技術の向上を目的とするスキルアップ研修と，デンソーを支える次世代の技術開発リーダー（プロフェッショナル）育成を担うハイタレント研修で構成されている．ハイタレント研修には，技術分野ごとに研修コースが設けられている．

その中で，ソフトウェア工学コースでは，「ソフトウェア工学を実践して課題解決できる技術者」の育成を目指して，2004 年より入社 5 年程度の中堅技術者に対して研修を実施してきた．

(2)改善前の状態

開講当時のカリキュラムは，ソフトウェア開発に関する従来技術の講義と，その技術を職場に適用した課題発表で構成されていた．しかし，研修生の課題発表を分析すると，講義内容をうまく適用できず，また有効性に乏しい発表内容であった．

(3)改善前の状態をもたらした原因（因果関係）

これは，研修生の課題解決力が，中堅技術者が本来保有すべきレベルを満たしていないことが原因であった．また，従来技術の講義だけでは，高機能・複雑化する車載製品のソフトウェア開発の課題に，十分に対応することは困難であることが判明した．

我々が開発する車載製品は，品質と安全性の両立が常に求められている．高品質で安心・安全な製品を提供し続けるためには，高い技術力と同時に，技術課題を的確に抽出し，課題解決できる品質意識の高い人材の育成が必要不可欠である．

また，自動車におけるソフトウェアは製品開発の中核を成すばかりか，開発対象も自動車だけでは完結せず，社会インフラとの連携など扱う範囲も急速に広がっている．今後，新製品，新サービスを顧客にタイムリーに提供していくためには，最新技術と従来技術を融合し，新たな課題に果敢に挑戦していく技術者を育成していく必要がある．

(4)計画した変更内容

①P2E 視点に基づく研修の再構築

カリキュラムを再構築するために，これまでの研修の問題，製品開発・ソフトウェア開発への要求，弊社の企業理念に着目して，目指す人材像を「ソフトウェア開発を牽引できる技術者」と再定義した．さらに目指す人材の必要要件を，P2E（実践：Practice，哲学：Philosophy，技術：Engineering）の視点からカリキュラムを設計，推進することを計画した．

②P2E コンセプトの確立とカリキュラム改善

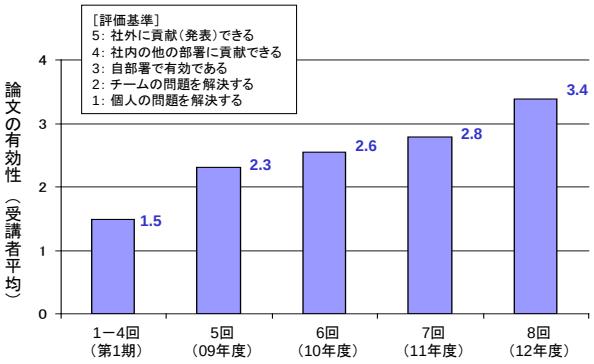
①で得られる知見をフィードバックすることで，研修コンセプトを確立（P2E コンセプト）し，それに基づく改善のフレームワーク構築につなげるように計画した．

(5)変更の実現方法①

「実践」の視点からは，課題解決の場を与えるために，研修期間かけて論文にまとめることを取り入れた．「技術」や「哲学」の視点からは，各分野の一線の講師による特論を取り入れ，技術者としての高い技術力やリーダーシップの考え方を学び取るようにした．段階的に効果測定できるよう，これらを順次導入した．

(6)変更後の状態や改善効果①

2009 年より P2E の視点に基づいて段階的に研修カリキュラムを改善してきた．図 1 に，2004 年から 2012 年の研修を評価した結果として，研修生が最後に提出する課題解決に関する論文の有効性推移を示す．着実な上昇が認められ，各施策の有効性が確認できた．




[図 1] 最終課題(論文)の有効性推移

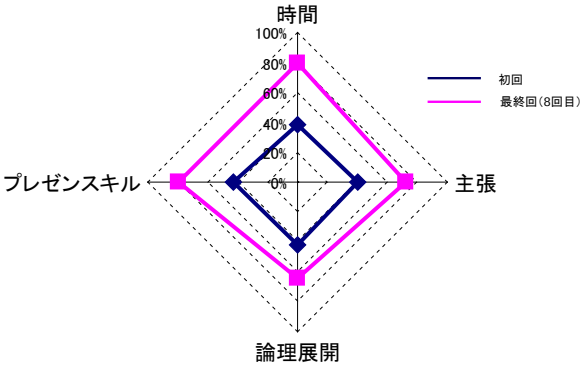
(7)変更の実現方法②

(6)で得られた知見から，今度は要求工学のゴール分析を適用して P2E コンセプトとして確立した．技術者の育成ゴールからゴール達成のために評価可能な P2E のサブゴールを設定した．さらにそこからタスクとして知識，スキル，教授法，評価指標を導出した．

これにより，新たな改善ポイントが浮かび上がり，13 年度は，教授法の改善として，LT（ライトニングトーク），反転授業をベースとした能動型実践学習に取り組んだ．

(8)変更後の状態や改善効果②

結果の一例として，図 2 に LT の導入効果を示す．4 つの観点で評価しており，全 8 回実施した．図には初回と最終回の結果を示してあるが，大幅なコミュニケーション力の向上が確認できた．




[図 2]LT 実施結果

(9)改善活動の妥当性確認

高度ソフトウェア専門技術者の育成には P2E の視点から育成にあたることが有効であると，経験的にも論理的にも確認でき，それぞれの相乗効果を促進するカリキュラムとするよう，今後も改善を進めていく．

一方で相互作用するが故に，カリキュラム評価と育成ゴールとの関連性をどう評価するかが難しい．この評価のしくみを確立することが重要で，有効なメトリクスとあわせて研究していく．

なお，研修生が自職場でリーダーとして活躍するには，研修後も自ら学び続け，経験を積むことが重要である．従って，職場とうまくタイアップした人材育成基盤の確立を進めていく．

3A3「現場メンバーの、現場メンバーによる、現場メンバーためのプロセス改善」奥村貴士(住友電工情報システム)

〈タイトル〉: 現場メンバーの、現場メンバーによる、現場メンバーためのプロセス改善

〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 奥村　貴士（おくむら　たかし）
所属：住友電工情報システム株式会社
〈共同執筆者〉
氏名（ふりがな）:
所属：

〈要旨〉
トップダウンのソフトウェア・プロセス改善は、現場のメンバーに改善内容や効果が充分に伝わらず、結果的に組織の長のための改善となっている。本発表では、開発現場のメンバーが自分たちの置かれた状況を改善するために、自らプロセス改善を行い成功の実感を得るまでの事例を、計画、監視と制御、決定分析の内容を中心に紹介する。

〈キーワード〉
現場、満足、改善、計画、会議、決定分析、CMMI、DAR、QPM

〈想定する聴衆〉
現場を改善したいプロジェクトマネージャー

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

弊社では、品質目標の指標として本番稼働後 3 か月間に発見される欠陥数を使用している。2013 年度の目標は、2012 年度と比べて 25%引き下げられた値に決まった。2013 年 4 月に親会社である住友電工の役員日程管理システムの再構築を請け負うことになった。QCD すべての目標が必達であるこのプロジェクトを成功させるため、CMMI プロセス・モデルを活用してプロセス改善を行うことにした。

(2)改善前の状態

1. 品質目標達成の見通しが立たない

当部門は 2012 年度の品質目標を達成していたが、同結果を 2013 年度の目標と比較すると下回っていた。期初ということもあり、目標達成に向けた具体的な計画はなかった。

2. プロジェクトの状況が悪い

プロジェクト管理は場当たり的で、報告を求められる都度、必要な数値を集めていた。プロジェクトの終盤に進捗の遅れが明るみになり、挽回のためにメンバーの負荷を大幅に上げたり、他プロジェクトから多くの要員を投入したりしていた。

プロジェクト序盤に実現が難しい仕様を選定していた。作成のフェーズで問題が明るみになるが、顧客と合意済みのため変更が難しく、時間を消費して何とか作成していた。コストを犠牲に何とか品質と納期を担保したものの、現場のメンバーは疲弊していた。

3. 改善することで期待できること

これらを改善できれば、計画的に作業できる、高品質のシステムを構築できる、現場のメンバーと顧客双方の満足が得られるといった効果が期待できると考えた。

(3)改善前の状態をもたらした原因（因果関係）

完了したプロジェクトが進めたプロセスを CMMI のプロセス・モデルと比較することで、プロセスの弱点を特定した。監視ができていない原因は監視プロセスにあるのではなく、計画プロセスにあることが判明した。また、プロジェクト序盤に進め方、体制、実装方法において誤った選択をしており、プロジェクトの後半のしわ寄せになっていることが分かった。代表的な原因は以下の内容であった。

1. 計画の不備

計画を立てるための計画や監視の計画がなく、プロジェクト管理が場当たり的であった。

2. 監視の成果物不備

不定期な監視のうえ何を残すかを決めていないので、データや記録が残っていなかった。この状況が、必要な数値を集めることを困難にしていた。

3. 仕様決定の不備

担当の割り振りが不適切で、知識がないメンバーが決断をしていた。

(4)計画した変更内容

すべての原因の根底が計画フェーズにあることがわかったので、まずは計画プロセスの改善を行うことにした。そのうえで、決定分析と解決(DAR)、プロジェクトの監視と制御(PMC)のモデルを活用し、不足しているプロセスを補うことにした。

開発メンバーに新人1名と、2年目の社員1名が加わることが分かっていたので、定量的プロジェクト管理(QPM)を活用し、問題が早期に検知できるよう計画した。

(5)変更の実現方法

プロジェクトを進めるうえで一体感や達成感は不可欠なので、現場のメンバーには「CMMIアプレイザル対策のためにやらされている」と思われずに実行することが課題であった。そこで、(a)CMMIの用語は一切使わない(b)原則、新しい成果物は作らず過去の資産を活用するようにして「現場メンバーの、現場メンバーによる、現場メンバーためのプロセス改善」を進めた。具体的には、以下の活動を行った。

1. WBSテンプレート改善

計画と監視が確実に実施されるよう、計画のための計画と監視のための計画をWBSに追加した。監視の成果物を定義し結果を確認できるようにした。

2. 決定分析のプロセス改善

過去に「必要な都度、決定分析を行う」と決めたことがあるが「都度」が訪れず実施できなかった。決定分析すべき内容の多くが課題に挙がっていたことに目をつけ、課題管理表に決定分析要否欄を追加した。また、決定分析すべきものがないか毎週のトレース会議で確認できるようにした。

目標と照らし合わせて重みづけや評価が実施されるように、テンプレートの作成も実施した。テンプレートにはKT法の決定分析手法を参考にMUSTとWANTが区別されるように工夫した。図1にサンプルを示す。

課題　Excelダウンロード機能の開発方法

| | | | 選択肢1<br>FWのみ<br>・FW提供機能のみを使用 | | 選択肢2<br>FW＋独自実装<br>・FW提供機能を基本的に使用<br>・実現できない箇所は独自実装(POIを直接使用)する。 | | 選択肢3<br>独自実装のみ<br>・FW提供機能を使用せず、独自実装(直接POIを使用)する。 | | 選択肢4<br>FW技術Gに機能追加を依頼し、選択肢1のMUSTを合格にする | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 条件 | 重み | 採点 | 得点 | 採点 | 得点 | 採点 | 得点 | 採点 | 得点 |
| MUST | 行の高さ調整 ができる | 5 | 5 | 25 | 5 | 25 | 5 | 25 | 5 | 25 |
| | 行の非表示 ができる | 5 | 5 | 25 | 5 | 25 | 5 | 25 | 5 | 25 |
| | 枠線の追加 ができる | 5 | 0 | 0 | 3:FWとPOIの連携調査要 | 15 | 5 | 25 | 5 | 25 |
| | 折り返し ができる | 5 | 5 | 25 | 5 | 25 | 5 | 25 | 5 | 25 |
| | セル結合 ができる | 5 | 5 | 25 | 5 | 25 | 5 | 25 | 5 | 25 |
| | フォント色の指定 ができる | 5 | 0 | 0 | 3:FWとPOIの連携調査要 | 15 | 5 | 25 | 5 | 25 |
| | 背景色の指定 ができる | 5 | 0 | 0 | 3:FWとPOIの連携調査要 | 15 | 5 | 25 | 5 | 25 |
| | 期日内に開発できる | 5 | 0 | 0 | 3:連携の調査工数に依存 | 15 | 5 | 25 | 0:不可 | 0 |
| | | 合計得点 | | 100 | | 160 | | 200 | | 175 |
| | | 評価 | 不合格 | | 合格 | | 合格 | | 不合格 | |
| WANT | コーディング量が少ない | 3 | | | 3:連携部分のコーディング量多い | 9 | 3:FWがラップしている箇所の実装がいる | 9 | | |
| | 実装が容易 | 3 | | | 3:FWとPOIの連携部分× | 9 | 5 | 15 | | |
| | FW技術Gのサポートが得られる | 1 | | | 3:どこまでうけれるかは不明 | 3 | 0 | 0 | | |
| | 改善の対応がしやすい | 3 | | | 3:FWとPOIの連携部分× | 9 | 5:POIの全機能利用可 | 15 | | |
| | | 合計得点 | | | | 30 | | 39 | | |
| | | 順位 | － | | 2 | | 1 | | － | |

重み　1～5(数字が大きいほど重い)
採点　5：可能
　　3：条件つきで可能

図１．決定分析のシート

3.　監視のプロセス改善

弊社で標準利用されているｕ管理図を使い品質の監視を実施するにあたって、1 年目と 2 年目のメンバーは常に異常値になる可能性があったので、経験者と分けて管理図を作成した。作成した管理図は(a)1 年目と 2 年目のメンバーの欠陥密度が時間の経過とともに減少傾向にあるかをチェックするため(b)経験者の異常値をチェックするために利用した。

会議のアジェンダに監視事項の報告を追加し、測定した結果が確認されるようにした。さらにゴールと現在地の報告を追加し、全員がゴールまでの道筋をイメージできるようにした。議事録として残す内容を定義し、監視の記録も残るようにした。

(6)変更後の状態や改善効果

今回の役員日程管理システム再構築プロジェクトは、他のシステムと比較して作成難易度が高く、1 年目と 2 年目の開発メンバーがいるという難しい状況であったが、次の効果があった。

1.　WBS テンプレート改善の効果

・　目標達成に向けた監視の計画を立てることができるようになった
・　毎週、プロジェクトの状況が確認されるようになった

2.　決定分析のプロセス改善効果

・　必要なときに、決定分析が実施されるようになった
・　選定ミスによる手戻りや、追加の調査を行う必要がなくなった

3.　監視と制御のプロセス改善効果

・ 単体テスト以降の欠陥が激減した
・ 組織の品質目標を上回るSTの欠陥2件､稼働後3か月間の欠陥0件という結果を得た（システムの規模は1KFP）
・ スケジュールには常にバッファがあり、追加の要望18件にも対応できた

4. その他の変化

・ 開発者満足度のポイントが3から4に向上（5段階）
・ 顧客満足度のポイントが3.5から4に向上（5段階）

以下に、開発者および顧客アンケートの自由記入欄から抜粋した内容を示す。改善で取り組んだ内容に関する意見が多く、改善の取り組みが上述のポイントの向上に影響しているといえる。

・ 作業にスピード感があった
・ トレースにより進捗のコントロールが適切になされた
・ 不要な手戻りが少なかった
・ 監視が適切になされた
・ 計画を明確にして進められた

(7)改善活動の妥当性確認

プロジェクトは品質目標、予算目標、納期目標すべてを達成した。課題であった満足度も、取り組み内容がよい結果につながったうえ、CMMI成熟度レベル３も達成することができた。モデルを参考にプロセス改善ができたといえる。

今回のプロセス改善は１つのプロジェクトへの適用であったので､次のプロジェクトに適用することでブラッシュアップを図りたい。決定分析のプロセスについては、全社展開を目指しており標準を策定中である。現場の知見を全社へ展開していきたい。

3B1「要件定義の変更による、パッケージ製品の魅力品質向上」松浦豪一(富士通マーケティング)

〈タイトル〉: 要件定義の変更による、パッケージ製品の魅力品質向上

〈サブタイトル〉: アジャイルで変わるパッケージビジネス

〈発表者〉
氏名（ふりがな）: 松浦　豪一（まつうら　ひでかず）
所属: 株式会社　富士通マーケティング
ＧＬＯＶＩＡ事業本部 業務ソリューション事業部 ソリューションビジネス部
〈共同執筆者〉
氏名（ふりがな）:
所属:

〈要旨〉
パッケージ品質向上のためにアジャイルプラクティスを適用することにより、障害の作り込みは減少した。しかし、お客様の満足度向上には至らなかった。そして、私はパッケージ開発リーダの５年間を通じて、開発項目の選定方法により、パッケージの品質が大きく変わることに着目した。品質機能展開を利用して問題分析を行った結果問題が、「要求品質の分析がされていない」、「品質特性が分析されていない」ことに問題が判明した。更に、２つの問題がパッケージプロジェクトのコミュニケーションを阻害することも分かった。この対策としてパッケージ品質を改善施策として、エクストリームプログラミングの計画ゲームを実施し、狩野モデルを利用した品質特性分析を実施した。上記施策により、パッケージ製品の品質が向上するとともに、パッケージ製品の売上が向上した。本論文では、問題分析からプロセス改善までのアプローチについて説明する。

〈キーワード〉
**アジャイル、品質機能展開、狩野モデル、品質、計画ゲーム**

〈想定する聴衆〉

*※高成熟度組織の方、SEPG 初心者、品質保証の方、ソフトウェアエンジニア、など*

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

私のプロジェクトにおいても、アジャイルプラクティスを適用し、品質向上に取り組んできた。具体的には、タイムボックスに合わせて開発項目を分割し、テスト自動化を組み合わせて、繰返型で開発する方式である。アジャイルプラクティスの適用の結果は以下のとおりである。

適用前
- V01L20:開発規模 64.6KS　障害累積件数 0.44 件/Ks

適用後
- V01L30:開発規模 59.0KS 障害累積件数 0.30 件/Ks
- V01L40:開発規模 89.7KS 障害累積件数 0.16 件/Ks
- V01L50:開発規模 72.6KS 障害累積件数 0.11 件/Ks

※障害累積件数は、1000Step あたりの累積障害件数

出荷後の発生障害は減少したが、パッケージの魅力的品質は向上するまでには至らなかった。

私は５年間のパッケージ開発リーダの経験を通じて、開発項目の決定過程により、パッケージ製品の魅力が向上しないことに着目した。その結果、製品の魅力が向上しないため、売上が向上しないことが分かった。

(2)改善前の状態

パッケージ製品は、特定のお客様の要望を聞きながら対応するＳＩ開発と異なり、開発元が開発項目決定する特性をもつ。開発項目の決定過程にて、お客様の要望を忠実に実現しようとすることで、結果的にパッケージ製品の魅力が向上しないことがあった。

パッケージ製品が魅力的品質になるか、否かは、以下がポイントとなる。

- 他社社製品に比べて機能的に劣らない、もしくは優位になる機能が存在する。
- 導入を簡易化する機能が存在するか。いかに導入コストを低減するか。
- トラブル発生した場合のエラー情報が分かりやすく、即時に解決できるか。

パッケージ製品の魅力的品質向上を阻害しているのは、パッケージ開発項目の決定過程にある。

(3)改善前の状態をもたらした原因（因果関係）

魅力的品質を開発するための手法である品質機能展開と開発項目の決定過程を比較した。その結果、以下の２つの問題が存在するため、開発項目の選定方法を不信に思うようになった。

- 開発項目の選定をするにあたって、お客様の重要度を考慮していない。
  （要求品質分析不足）
- 開発項目がお客様にとって価値があるのか分析がされていない。
  （品質特性の分析不足）

(a) 要求品質の分析不足

要求品質の分析が不足しいることが、パッケージ開発に対して以下の問題を引き起こしている。

- 開発項目が選定されない。
- 開発項目を実現する価値が明確にならない。
- 開発項目が肥大化し、ムダが増加する。

上記の問題について説明する。

（ア）　開発項目として選定されない

開発項目は具体的な手段になっているため、実現性が困難な場合に開発項目から除外されてしまう。パッケージ製品の魅力向上させるための項目が開発されない場合がある。開発項目に実現手段が記載されているため、内容を評価する前に難易度やコストで実現性を判断する。要求品質を分析する前にコストなどの問題で開発を断念してしまう。

（イ）　開発項目が与える価値が明確ならない

開発項目は要求品質展開の原始情報のような情報になっており、潜在的な要求品質が明確になっていない。お客様にとっての価値が明確にならないことにより、開発項目を正しく理解することができない。特に開発チームにとって価値のない開発項目になる。

（ウ）　要件の肥大化

新機能を開発する場合、具体的な機能レベルではなく大きなくくりで設計されるため、要件が肥大化する。肥大化した要件の中にある開発項目には優先度が設定されない。お客様にとって価値のある項目と価値のない項目が区別されないで開発されてしまう。

(b) 品質特性の分析不足

品質特性の分析が不足していることにより、開発項目の優先順位が正しく設定されないため、魅力的品質向上しない項目を開発してしまう場合がある。逆に魅力的品質向上する項目を開発しない場合がある。開発項目ごとの顧客評価を理解しないで開発することにより、顧客満足度につながらないため、パッケージ製品の売上向上に結びつかない。

狩野モデルを利用した分析とは、品質特性を見極めるための手法である。狩野モデルの考え方について説明する。品質には二つの側面があると考えられており、満足・不満足という主観的側面と物理的充足・不充足という客観的側面である。この対応関係は図１の通りであり、魅力的品質、一元的品質、当たり前品質という３つの品質要素を導き出している。これに加え、実現することによって変化しない無関心品質や実現することが弊害になる逆品質がある。狩野モデルの分析により、５つの品質要素に分類することができる。品質特性があきらかになるため、開発によってお客様に与える効果を事前に把握でき、目標とする品質にあわせて開発項目を選択できる。

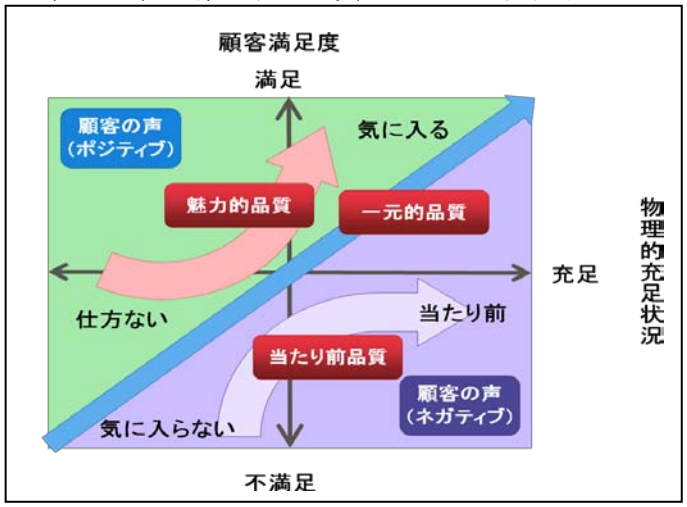



図１　品質の二側面性の関連図

製品の売上を左右する品質特性である、５つの品質要素について説明する。

- 魅力的品質要素
  「充足であれば満足を与えるが、不充足であってもしかたないと受け取られる品質要素」
- 一元的品質要素
  「充足であれば満足を与えるが、不充足であれば不満を引き起こす品質要素」
- 当たり前品質要素
  「充足であれば当たり前と受け取られるが、不充足であれば不満を引き起こす要素」
- 無関心品質要素
  「充足でも不充足でも、満足も不満お引き起こさない品質要素」
- 逆品質要素
  「充足されているのに不満を引き起こしたり、不充足であるのに満足を与えたりする品質要素」

**(4)計画した変更内容**

上記の問題を解決するには、開発項目の決定過程に対して以下のアジャイルプラクティスを応用して適用するプロセス改善に取り組んだ。

- エクストリームプログラミングの計画ゲーム
- 狩野モデルを利用した品質特性分析
- アンケート結果をベースとした計画会議

**(5)変更の実現方法**

(a) 計画ゲームを応用した開発項目選定

パッケージ開発の開発項目選定にエクストリームプログラミングの計画ゲームを応用することにより、パッケージエンハンス項目の要求品質を向上させる。当プロジェクトでは一般的な計画ゲームに比べ以下の手法を追加した。

- 開発項目選定時の洗い出し条件
- ユーザストーリ作成時の工夫

計画ゲームを応用した開発項目の選定方法について説明する。

（1） 開発項目の選定

当プロジェクトでは、開発項目の選定に問題があったため、解決するために独自の手法で開発項目を選定した。お客様の要求に従って開発項目を選定するために、以下の視点で洗い出しを行う。

- 同じ開発要求が複数社で起票されている。
- 他社と比較して機能的に劣っている。
- 実現することで他社を凌駕できる。

上記の視点で開発項目を抽出することによって、実現性やコストを優先した開発項目選択を防ぐことができる。

（2） ユーザストーリの作成

選定した開発項目をユーザストーリに分割し、お客様の価値を明確にする。エクストリームプログラミングの計画ゲームでは、開発項目の優先順を決定するためにユーザストーリを作成する。想定した利用者をもとに「誰が」「何を」「どのように」を明確にしたユーザストーリにする。あわせてその機能を利用するための理由を明記する。開発項目設定にくらべて実現する内容は小さく、想定利用者が明確になり、得られる利益も明確にできる。当プロジェクトでは価値を明確にするために、具体的な手段ではなく、価値を明確にするように取り組んだ。

（3） まとめ

上記の施策により、お客様視点で開発項目の選定分析を行うことが可能になった。ユ

ーザストーリ作成は、開発項目の要求品質を展開と同じ効果があることが判明した。このため、計画ゲーム応用した開発項目の選定により、パッケージ製品の要求品質向上を実現できる。

(b) 狩野モデル利用した品質特性分析

狩野モデルを利用した分析により、パッケージ開発項目の品質特性を明確にする。一般的な狩野モデルの分析に加えて、利用者を変えて複数のユーザストーリにする。狩野モデルを利用した品質特性分析について説明する。

（1） 充足・不充足のアンケート

パッケージ関係者に対して、開発項目の充足・不充足のアンケートを実施し、評価の二元表で分類し、開発項目ごとの魅力的品質を明確にする。ユーザストーリごとに以下の形式でアンケートを実施する。

「もし、○○○は、○○○できるようになったら」どう感じますか。

1. 気に入る
2. 当然である
3. 何とも感じない
4. しかたない
5. 気に入らない

「もし、○○○は、○○○できなかったら」どう感じますか。

1. 気に入る
2. 当然である
3. 何とも感じない
4. しかたない
5. 気に入らない

表２　評価の二元表

| | | 不充足質問 | | | | |
|---|---|---|---|---|---|---|
| | | 気に入る | 当然である | 何も感じない | しかたない | 気に入らない |
| 充足質問 | 気に入る | Q | E | E | E | L |
| | 当然である | R | I | I | I | M |
| | 何も感じない | R | I | I | I | M |
| | しかたない | R | I | I | I | M |
| | 気に入らない | R | R | R | R | Q |

M　当たり前品質（Must-have：必須）
L　一元的品質（Linear：線形）
E　魅力的品質（Exciter：魅力的）
R　逆品質（Reverse：逆効果）
I　 無関心品質（Indifferent：無関心）
Q　懐疑的回答（Questionable：懐疑的回答）

（2） 品質要素の分類

ユーザストーリに対して、各自が思っている品質要素が明確になる。ユーザストーリごとに５つの要素に対して何件選択されたかを数値にして、開発項目ごとの割合を明確にする。

（３） まとめ

開発項目を品質要素が分析することにより、お客様に提供する価値を明確にすることができる。開発項目の価値が明確になることによって、開発の優先順位や開発範囲を容易に決めることができる。当プロジェクトでは、アンケートする項目を工夫して実施した。具体的には、同じ機能に関しても想定する利用者を変えてアンケートを実施した。

(c) アンケート結果をベースとした計画会議

ユーザストーリごとに、概算工数・品質特性を明確にし、開発項目の優先順位を決める計画会議を実施する。開発項目の優先順位は、顧客からの要望や優先度を合わせて加味する。それらのすべての情報を持ち寄り開発の優先順位を決定する。品質特性は立場により異なった値がでる場合もあり、乖離している部分については計画会議で話し合いを実施する。

パッケージ製品に関わる拡販・導入・サポートメンバに参加してもらい計画会議をすることにより、製品のエンハンスを実感することができる。パッケージプロジェクト全体での意思疎通が円滑になる。

**(6) 変更後の状態や改善効果**

（１）選定方法改善による要求品質の向上

パッケージエンハンス項目の選定方法を改善したことにより、パッケージの要求品質が向上した。とくに開発項目をユーザストーリにした上で価値に分割できた。製品の魅力を向上させる開発項目を選択することができた。その結果、以下の効果が得ることができた。

- 他社提案に比べ遜色のない提案ができるようになり、なおかつ、自社の必勝パターンを作ることができ、商談化した案件の提案確率が向上した。
- パッケージ導入費用を低下させることができるため、提案した案件の内示率が向上した。
- トラブル発生から解決までの時間を短縮することができるようになり、導入 SE の負荷が減少し、お客様への提案活動などに労力をさくことができるようになった。

（２）品質特性分析による品質特性の改善

品質特性を分析した開発を実施することにより、品質特性が改善した結果、未実施の開発項目の品質特性が変化した。魅力的品質が一元的品質に変化し、一元的品質が当たり前品質に変化する。パッケージエンハンス前は、不充足であっても不満にならなかった案件が不満の案件に変化する。

（３）計画会議によるモチベーションの向上

開発項目の優先度を決めるにあたって、項目の選定及び分析方法が改善された結果、開発項目決定過程の満足度が向上した。開発項目が実現されリリースすることにより、拡販・導入・サポートメンバのモチベーションも向上した。開発項目に対する共通理解をはかることができ、開発チームのモチベーションも向上した。

**(7) 改善活動の妥当性確認**

パッケージ開発項目決定過程の改善により、製品の魅力的品質の向上をはかることができ、パッケージ製品の売上が向上した。

結果的に、商談の提案率・内示率が向上し、製品売上高が向上した。具体的には商談化（S3）：提案（S5）：内示（S7)の比率（商談の勝率）が改善している。具体的には、V01L20当時は、24社商談化して、４社提案し、１社内示を貰う割合であった。V01L50については、５社商談し、３社提案し、１社内示をもらえるようになっている。

適用前

- V01L20　24:4:1
- V01L30　18:4:1

適用後

- V01L40　10:3:1
- V01L50　 5:3:1

パッケージ拡販部隊が１社獲得するのに５社まわれば内示をもらえるようになった。同じ時間をかけることができるようになれば、約５倍の成果を得ることができる。

パッケージ製品の組合せ提案をするケースが増えており、商談規模は３倍になっている。

3B2「残念スクラムに立ち向かえ！スクラムマスター奮闘記」内藤優介(富士通エフ・アイ・ピー)

〈タイトル〉: 残念スクラムに立ち向かえ！スクラムマスター奮闘記

〈サブタイトル〉: ～waterfall な若手 SE チームのスクラムチームビルディング～

〈発表者〉

氏名（ふりがな）: 内藤　優介（ないとう　ゆうすけ）
所属 : 富士通エフ・アイ・ピー株式会社

〈要旨〉

エンタープライズシステム開発の主流は waterfall による開発である。だが、昨今ではアジャイル開発（特にスクラム）がエンタープライズの世界でもブームになりつつある。弊社でも、それにもれずスクラム開発への挑戦をすることとなった。waterfall な世界の人材育成により形成されたコマンド＆コントロール型若手ＳＥにより形成されたスクラムチームは次々とスクラムがうまく行っていない状態、「残念スクラム」に陥っていく。本論はそんなチームに対してスクラムマスターが施した改善策についての経験則の発表である。

〈キーワード〉

アジャイル開発、スクラム開発、スクラムマスター、waterfall 開発、若手ＳＥ、
サーバントリーダーシップ、ティーチングリーダーシップ、スクラム開発の品質

〈想定する聴衆〉

スクラム開発を実践している方（特にスクラムマスター）、若手ＳＥを率いているリーダーの方、
スクラムの品質について考えている方

〈適用状況〉

□多用されている段階、□適用できる段階あるいは初めて適用する段階
■適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉

□汎用性がある、
■類似プロジェクトにも適用可：具体的な類似点
（数値に現れない部分に目を向けて改善を図るという点）
□自プロジェクトのみ

〈発表内容〉

(1)背景

スクラム開発というものは非常に実体がとらえにくく、また各プラクティスについてもそれぞれの「やる意味」をチームが把握し、自分たちにとって効果的なものと判断した上で実施しないと、その効果は発揮されない。それはスクラム開発のみならず、アジャイル開発全般にいえる事である。本論の課題もこの点から来るものである。本論のスクラム開発は waterfall の要件定義と設計段階で何度も手戻りを発生させていたプロジェクトに対して解決策として、若手ＳＥチームが初めてスクラムに挑んだプロジェクトについての話である。

(2)改善前の状態

コマンド&コントロールが体に染みついている waterfall 開発一択な世界のＳＥ、更に、コントロールされる側である若手ＳＥのチームがいきなりスクラム開発に挑んだ結果その受け身、指示待ち、リーダー頼みの意識がスクラム開発の大きな阻害要素となった。前工程が完了していることが前提であったり、工程ごとに品質管理部門の監査が前提の waterfall な会社の中でスクラムを実施したりする事で生まれる誤解も、開発チームを大きく混乱させていた。１回目のスプリントではリリース可能なものは何もなく、何一つ「価値」を生み出すことが出来なかった。

(3)改善前の状態をもたらした原因（因果関係）

スクラムマスターとは本来、空気のような存在であることが健全とされる。サーバントリーダーシップという単語が示す奉仕的なリーダーシップが要求されるのがスクラムマスターである。本論はスクラムチームを「壊れた状態から遠ざける」ことにすべてを捧げるスクラムマスターのあり方に着目した。

スクラムにおいて、waterfall が体に染みつき、しかも若手 SE のチームのチームビルディング期に本当に必要なリーダーシップはサーバントリーダーシップではなく、スクラムマスターもまた違ったリーダーシップをチームに発揮すべきなのではないかという部分に問題解決の糸口を見つけた。

(4)計画した変更内容

スクラムマスターの振る舞いを変更することで、問題の解決を試みた。
具体的にはスクラムの伝道目的以外では開発チームに教育を行わない、計画や成果物に対して意見を言ったり、作業をしたりしないというスクラムマスターの振る舞いを一旦諦め、「スクラムマスターとして開発作業に介入」することを試みた。

(5)変更の実現方法

ミーティング中、実作業を行わないスクラムマスターの決まりを破り「スクラムマスターとして」設計作業に参加し、喋りすぎたり強くメンバーに意見したりする開発者を抑えつつ、意見の少ないメンバーに意見を求め、出てきた問題点はリスト化してリーダー風メンバーに報告するのではなく、付箋に記載し自分たちで管理することで、コマンド＆コントロールの雰囲気の破壊を試みた。

また、コマンド＆コントロールが得意なメンバーの性質を逆に使用して「スクラムの基本的なやり方」についてはコマンド＆コントロールとまでは行かないまでも、「スクラムマスターから自然と教えてもらうような」環境や状況をつくりだす教育的なリーダーシップを振舞う存在としてスクラムマスターの立ち位置を少し変えた。ミーティング

をスタンディング形式に変更し長くなると疲れるようにし、メンバーがより早く無駄なく進行する工夫を始めるようにした。ＰＣの持ち込みも禁止し、下を向いたり、メモ中で意見できないそぶりをしたり出来ないようした。また司会も持ち回りにし、自分たちのスクラムのうまくいっている様子、うまくいっていない様子を客観的に見せた。

最後に、スクラムを半強制的に成功するような状況を用意し、メンバーに小さな成功をもたらすよう仕向けた。具体的に、非常に簡単なものを、非常に短い時間で作るスプリントを用意した。メンバーは期間の短さに焦りながら、自律と工夫と無駄を省くことに集中させた。

(6)変更後の状態や改善効果

スクラムマスターの３つの問題の解決策はそれぞれ以下のように若手ＳＥチームに効果を発揮した。

スクラムマスターとして設計作業などに参加し、指示だしばかり行うメンバーの勢いを抑えながら、指示を受けるばかりのメンバーに少しずつ発言や考えさせる時間を生み出すことに成功し、チーム全員が良くも悪くも同じ立場に立って議論、開発できるチームを作ることに成功した。

教育的なリーダーシップを発揮し、「スクラムを自然と教えてもらう」環境や状況をつくることで、自然と waterfall な環境から来ていた受け身な姿勢が改善され、メンバーが正しくスクラムを実行できるようになりスクラムに対する不安を取り除くことに成功した。

半強制的に成功するような状況を設ける（簡単なものを短い期間で作らなければならない）ことで、メンバーは技術的問題による失敗から解き放たれ、スクラムを実行することに多くの集中を割くことができた。また短い開発期間が waterfall のやり方である設計を全て終わらせてから次の工程に移るといった「癖」から開発チームを解放し、工夫、議論、早期の問題発見、優先度を意識したアジャイル開発に導き実際にそのスプリントは無事にリリースした。この小さな成功が失敗を続けていたチームの雰囲気を一遍させ、チームに安心と自信を与えた。

結果、若手チームは waterfall をやっていた時から見違えて成長し、更にお客様からも好評をうけるような価値あるプロダクトを提供することに成功した。

(7)改善活動の妥当性確認

スクラムをはじめとするアジャイル開発において、書籍や論文にしるされている「やり方」を忠実に実施する事では開発チームの熟成やよりよい価値の創造には直結しない。スクラムに対するある程度の目的、効果の理解とスクラムを健全に遂行するためのチームをビルドすることが必ず必要であることが分かった。そしてスクラムマスターが必ず取り組まなければいけないことは、そのチームに本当に必要で効果的なチームビルドの「やり方」を確立する事である。本論においてスクラムマスターは、アジャイルが初体験でかつ、コマンド＆コントロール体質な若手ＳＥチームに対してサーバントリーダーシップではなく教育的リーダーシップを強く要求された。

プロジェクトマネジメントの知見からこの件を考えると waterfall 開発によく見られる数値的な監視、マネージメント重視のアプローチだけでは、「スクラムがうまく行っていない状態」を検知するためには意味をなさないに等しいレベルで難しい。プロダクトの品質は測れるが、開発チームが継続的にお客様にとって本当に必要な価値を高いアジリティをもって創造し続けることができる状態になっているかどうかを「測る」事ができない。その部分の数値には表れない監視責任はスクラムマスターにゆだねられおり、

マネージメントを行う先は開発チームではなく、スクラムマスターであるいう事を全員が理解しなければならない。

今後取り組むべき課題については上記知見にあった「数値化できないスクラムチームの健康状態」をスクラムチームの「外」にも見える化できるように取り組みたい。社内の品質管理部門による定期的な検証が主流といえる日本のＳＩ企業内でスクラム開発を行う際にプロダクトのいわゆる品質だけでなく、スクラムの最重要部分である「チームが常に改善され続けているのか、お客様にとって重要な価値を創造できているか」についての品質も検証できるような企業全体としてのアジャイル開発プロセスの確立に挑戦していく。

3B3「大規模組込み開発のアジャイル型プロセス改善」陸野礼子(AVC テクノロジー)

〈タイトル〉: 大規模組込み開発のアジャイル型プロセス改善

〈サブタイトル〉: ～アジャイルマインドによる継続的アプローチ～

〈発表者〉
氏名（ふりがな）: 陸野　礼子（りくの　れいこ）
所属： AVC テクノロジー株式会社　システム技術グループ
〈共同執筆者〉
氏名（ふりがな）: 前川　直也（まえかわ　なおや）
所属： パナソニック株式会社

〈要旨〉

組込みでのアジャイルの活用事例も増えてきつつある。多くの組込み開発が抱える「大規模・短期間・低コスト開発での消耗戦」を打破し、変化に強く、価値（魅力ある製品）をお客様にお届けできる開発現場を目指すために、アジャイルも強力な武器になるが、ノウハウがなければ失敗することも多い。本発表では、具体的な製品開発の取組みを通して、アジャイルを導入するためのポイントや、現場で活用するためのプラクティス、さらにアジャイル型のアプローチ方法をご紹介する。

〈キーワード〉
アジャイル　組込み系大規模ソフト開発　スクラム　スプリント　チームワーク　コミュニケーション

〈想定する聴衆〉
ソフトウェアエンジニア、アジャイルを導入検討されている SEPG や組織
*※高成熟度組織の方、SEPG 初心者、品質保証の方、ソフトウェアエンジニア、など*

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

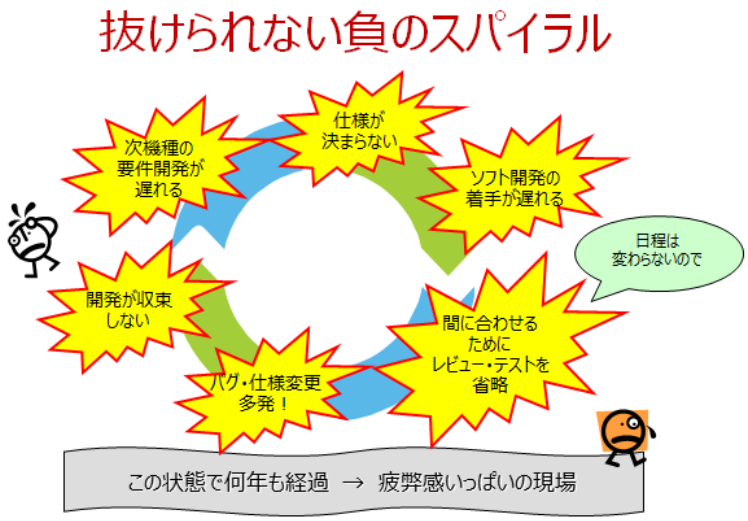
- ■ 組込みを取り巻く現状：
  - ➢ 商品仕様の複雑化／多様化
  - ➢ ソフトウェアの規模増大→開発体制の巨大化／複雑化
  - ➢ 激しい価格競争→「トガリ」がなければ即座に転落
  - ➢ ビジネスモデルの変化→単品商品からコラボ商品へ

  ★★『要求される価値』から『創り出される価値』へ変化。
- ■ ソフト業界の現状：
  - ➢ 以前は狙っていけば的中する確率が高かったが、今の組込み業界では、環境の変化、ユーザニーズの多様化、競合他社との競争激化などで先が読めない状況。
- ■ アジャイルへの期待：
  - ➢ 『価値を創りだす』『変化に強い』といったアジャイルのキーワードに期待して、組込みでもアジャイル導入の要求が増えてきているが、理解不足や誤解があるまま活用しようとすると失敗することも多い。そこで実際の失敗事例から見えてきた組込み開発へのアジャイル導入のコツ（ヒント）を紹介する。

(2) 改善前の状態

- ■ 大規模開発体制（約 100 名）：フラットな体制になっており責任と権限が曖昧
- ■ 多機種展開、五月雨式の発売ラッシュによる開発期間の短縮と負荷増大
- ■ ソフト構造はムダに複雑化されスパゲティ状態

★★　この状態を切り抜ける術もなく、抜けられない負のスパイラルに疲弊感が募るばかりの開発現場。




＜目標＞

この疲弊感を拭い去り、元気な開発現場にしたい！

そして、魅力ある商品を開発できる開発現場にしたい！

(3)改善前の状態をもたらした原因（因果関係）

根本的な課題を３つ抽出した。

- 『だらだら遅れる』
  - 計画が曖昧、場当たり的な開発。
  - 日程感が希薄（出荷に間に合えばいい？！）。
  - 遅れないための組織の工夫がない。
- 『バグが多い』
  - 毎回終盤になると火の車。バグ対策型開発が当たり前。
  - システム全体を把握しているメンバーが少ないので影響範囲が不明。
  - 設計テスト不足。→システムテスト依存
- 『とことん受け身な開発』
  - 要求が曖昧なまま機能を実装。
  - 仕様変更多発、仕様がらみのバグが多発しても工夫のないまま開発。
  - そもそも商品への愛着が希薄？！

(4)計画した変更内容

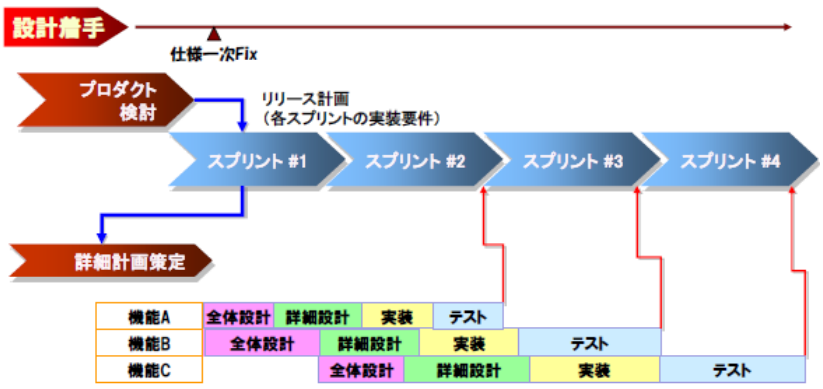
- ゴールとリズムで開発の流れを進化
  - 「だらだら」から脱して、メリハリをつける！
- スムーズに流すための体制と役割を強化
  - ソフトウェア開発の基本の強化、かつコミュニケーションを密にし協調することで、わけのわからない部分をフォローしあいバグ削減を目指す！
- 自分たちで考え成長を促すために風土を進化
  - もっと価値のある商品にするための「トガリ」を自分たちで創り出す！

(5)変更の実現方法

- 最初は基盤作りから。アジャイルを知らなくてもできることから始める。
- マネジメント層や他部門(ハード、工場など)に迷惑をかけないように、運用期間を限定（マネジメント層を説得しなくても運用できる範囲、つまり仕様決定から全機能実装までの開発期間に限定)。かつ商品開発基準のマイルストーンにリンクすることが重要。

＜スプリント開発（スクラム）の導入＞

- ゴールとリズムを作る
  - 仕様決定（ソフト要件の FIX）から全機能実装（システムテスト開始）までの開発期間を１週間のタイムボックス（スプリント）で分割。ソフト要件に対し設計者の概要見積りをベースにリリース計画を作成し各スプリントの実装要件を決定。



- スプリントの最初に計画を確認し、最後に成果レビューの振り返りを実施し、進捗を確認しながら、遅れや課題を是正する。
- ゴール設定の継続：仕様が変更されたことを明確に意識し、ゴールの見直しをスプリント毎に行う。計画は変わるものという意識と、機能の優先度を見直す機会をスプリント開発に取り込む。
- 進捗管理としては、「かんばん」の代わりに Redmine（ツール）を活用。シンプルなワークフローによる実績更新の簡易化を図った。
-

■ コミュニケーション強化のプラクティスを導入
- チーム単位で KPT 実施、昼会（チームリーダー＋ソフトプロジェクトリーダ）、課題管理表、結合テストシート、設計内システムテスト期間

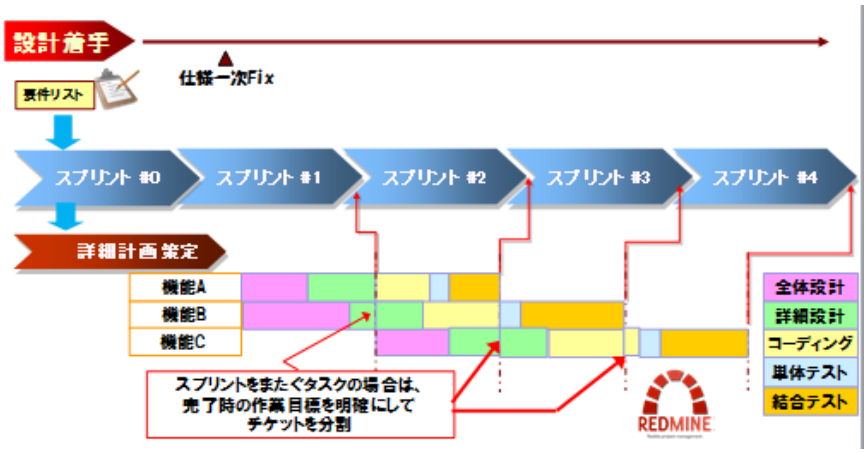
＜最初の結果＞

■ 日程面でのゴールは守ったが、品質面でのゴールが曖昧で結果的に遅延してしまった。
- 予定通りリリース、でも、ゴールを守るために結合テストを省略してしまい、結果、完成度が低く、仕様変更とバグが多発し、遅延！！
- スピード感は出た。ただ、リズムっぽいけどメリハリが足りない？！

＜さらなる改善＞

■ 完成度を上げるために、エンジニアリングの基本である V 字モデルを意識してもらい、実施すべきプロセスを計画に反映する方法に変更。
- スプリント#0 を追加。見積り時に UI 設計（動作仕様書作成）、全体設計、詳細設計など何の設計が必要か検討し、その検討結果から詳細スケジュール（Redmine のチケット）を作成する。（必要なプロセスの抜け漏れの防止と設計者自身への意識づけとなる）

■ メリハリをつけるために、チケットの分割方法を見直す。スプリントの期間である１週間以内にチケットを分割し、途中成果物であってもスプリント毎にゴール設定をする方法に変更




■ 仕様変更多発を防ぐために、設計者が基本動作仕様書を作成。企画と設計部門とのレビューを通じて、仕様の漏れ・誤解等の欠陥を上流で見つけて対策するプロセスに変更。

- スプリントの完了条件の明確化。結合テストを完了し残欠陥が無いことを完了条件とし、リリース時のチェック項目とする。

(6)変更後の状態や改善効果

- リズムとゴールにのせていくことで、今まで見えなかったことが明確になった
- 一人ひとりの意識は変わってきた。
  - シンプルな「見える化」と「場」の設定によりコミュニケーションが活性化。遅れの是正、仕様変更対応、作業の優先度や負荷調整など積極的に関与できるようになった。
  - 受け身な姿勢からの変化：基本動作仕様書の作成やレビューを通じて、開発者も自分が製品仕様に直接関わっていることを認識するようになった。
- バグも減ってきた
  - 基本動作仕様書を作成した要件は、結合テスト以降の欠陥が減り、その効果を全員で共有。

(7)改善活動の妥当性確認

- アジャイルは万能薬ではなく、基本スキルが十分でない場合、改善に時間がかかる。
- 要求分析～出荷までのモノづくり全体のプロセスに適用させることも可能だとわかった。
- プロセス改善活動もリズムを持つことで、変化に対応しながら舵を取ることが重要
  - 半年ごとに大反省会を開催。各チームからの報告は、最初は不具合の反省中心だったが、回を重ねるうちに、定量的なメトリクスや改善取組の KPT が報告されるようになった。
  - SEPG は 3 つの改善ポイントを軸に、現場の変化を組み入れ、具体的なアプローチを方針として発信し、ゴールを全員で共有する機会として活用。
- 今回のプロセス改善を通じて、アジャイルを導入することで、より強みと弱みが明確になり、的確な対策を打ち出すことができた。プロセス改善自体もタイムボックスで回していくことでよりスピーディーに効果が期待できるとわかった。

3C1「テストデータ自動生成による品質・コストの改善」服部悦子(住友電工情報システム)

〈タイトル〉:
テストデータ自動生成による品質・コストの改善
〈サブタイトル〉:
テスト設計システム構築とアイデアを出す工夫
〈発表者〉
氏名（ふりがな）:服部　悦子（はっとり　えつこ）
所属：住友電工情報システム株式会社　ＱＣＤ改善推進部
〈共同執筆者〉
氏名（ふりがな）:
所属：

〈要旨〉
昨年、単体テストの品質向上を目的に自動テストを実現したが、データベーステストに関しては欠陥削減には繋がらなかった。一方でテストデータ作成時間を効率化したいという声もあり、両者を解決すべくテストデータを自動生成できる仕組みを構築した。本発表では、これを実現するアイデアを開発部門メンバーから引き出す工夫と、テストデータ自動生成の仕組みについて述べる。

〈キーワード〉
改善のアイデア、ワーキンググループ、単体テスト、テストデータ、自動生成

〈想定する聴衆〉
改善活動を推進する部門の方
プログラム開発者、特に単体テスト担当者

〈適用状況〉
□多用されている段階、■適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1) 背景

当組織は毎年、品質と生産性に関する事業目標を設定し、目標達成に必要なワーキンググループ（ＷＧ）を新規に立ち上げたり、既存のＷＧに役割を割り当てたりしている。私はその中の複数のＷＧに参加しているが、今回は単体テスト工程における品質改善とコスト削減の改善を目的としているＷＧの活動について報告する。昨年、品質向上の目的から単体テストの自動テスト化を実現した。(SPIJapan2013 発表）その結果、欠陥の作り込みは組織基準値に比べ約半減した。一方、開発者から単体テスト時にテストデータ作成に時間がかかるという声があがっており、生産性の面で何らかの対策を考える必要があると感じていた。

(2) 改善前の状態

課題１．さらなる品質の改善

自動テストの実現により作込欠陥実績は以下のようになった。

表１．自動テスト実施 PJ の作込欠陥（組織基準を 100 とした相対値）

| | 組織基準値 | 自動テスト実現ＰＪ実績 |
|---|---|---|
| ＣＩ＋ＵＴ検収 検出 | 100 | 47 |
| (欠陥区分別内訳 TOP3) | | |
| ロジックの実装ミス | 23 | 2 |
| 実装漏れ | 19 | 3 |
| DB 処理 | 10 | 9 |

「ロジックの実装ミス」「実装漏れ」の作り込みが大きく削減できている。自動テストの効果かどうかを確認するため作り込みを防げなかった欠陥の内容を調べたところ、欠陥が残った（作り込んだ）ものはデータベースの値をもとに動作が決まるようなロジックであった。逆にデータベースの値を使わないロジック（画面の入力値やセッション変数の値をもとに動作が決まるもの）は０件で作り込みを抑制できていた。
「DB 処理」は上位２つと比べ削減率が低い。

課題２．単体テスト時のテストデータ作成工数の削減

単体テストで使用するテストデータはプロジェクトによって本番データをマスキングしてテストに使ったり、例えば出荷機能のテストでは先行業務である受注機能を使って出荷機能のテストデータが作成できるように開発順を工夫したりすることで準備工数を削減していた。
しかし、新規のシステムは本番データが無いし、現行システムと新システムのＥＲモデルが大きく変わる場合は現状のデータは使えない。(移行プログラムは完成していないと利用

できない)。又、開発を部分的に請負に出す場合は開発順を工夫することもできない。
このように全てのプロジェクトで準備工数を削減できる状態では無く、開発者が時間をかけてテストデータを作成していた。

(3) 改善前の状態をもたらした原因（因果関係）

・課題１の原因

(a)テストケースを網羅するテストデータの不足

画面の入力値やセッション変数は自動テストから動作するときにパラメータとして与えられるようにしており、パラメータが取り得る値のケースを自動テストとしてコーディングすることができたので作り込みが抑制できている。一方でデータベースの値を使うロジックは、事前にテストに必要な種類のテストデータをデータベースに登録しておく必要があるが、コスト面での負担が大きく自動テストが実現できていなかった。

(b)繰り返しテスト不可

データベースの値はテスト（プログラムの実行）によって更新され値が変わってしまうので繰り返しテストができず繰り返し自動テストができないことも原因である。

・課題２の原因

テストデータ作成に時間がかかる理由を確認するため、どのよう作成しているか開発者数名にインタビューした。その結果、以下の点で時間がかかることがわかった。

(c)レコードの多さ

テストに必要なレコードは出荷実績であっても、その先行業務となる出荷指示や出来高、さらには受注データなどの登録が必要である。加えてそれらのデータが先行業務のプログラムから発生するデータの仕様を理解して作成する必要がある。

(d)項目の多さ

区分値や数量・金額はテスト箇所の仕様によってテスト値が決まるが、テストには関係しない項目も型・桁の定義（制約）に則った何らかの値を決めて登録する必要がある。(c)のケースでレコード数が多いとさらに増える。

(4) 計画した変更内容

これらの問題を解消するため、テストデータを自動生成することを考えた。自動生成できれば、面倒なレコード作成や値設定が不要になりテストデータ作成工数を削減できる。具体的には開発対象システムのメタデータ（項目の型・桁 等）を活用することにより機械的に整合性のとれた値を生成できる。又、そのデータを繰り返しテスト環境（データベース）に反映できれば、自動テストが可能となる。

このことからテストデータ自動生成を提案し、その実現方法について単体テスト改善ワーキンググループ（以降、ＷＧ）で取り組むことになった。

(5) 変更の実現方法

(5-1) アイデアを出す工夫

どうすればテストデータを自動生成できるのか、初めはどう取り組んでいいかわからず、ブレーンストーミングでメンバー各自が思いつくアイデアをあげてみた。

「ＥＲモデルから区分値の組合せで自動生成しておき、それを利用できないか」
「プログラム中のＳＱＬを解析して、必要なデータを生成できないか」

前者はシステムに存在する区分値から無限の組合せのデータを発生させたところで、どうやって個々のプログラムのテストデータに選択するのか、というところで断念した。後者はそもそも SQL が間違っている場合に誤りを検出できないし、抽出後のロジックのケースを網羅できないというところでうまくいかなかった。

そこでやり方を少し変えた。
ＷＧメンバーに、テストデータを作成するのが面倒だったプログラムをあげてもらい面倒だった箇所（機能）に着目して、その箇所に必要なテストデータをどのようにインプットすれば自動生成できるか考えてみた。するとこのやり方で議論は活発になり、メンバーから沢山のアイデアが次々に出てきた。

何故、議論が活発になり結果としてアイデアが出やすくなったのか分析したところ、３点の重要な事がわかった。以下、具体例で説明する。

（新規）検索→一覧

1．対象エンティティ

| 対象エンティティ | JOIN | R/E | テストデータ作成対象 |
|---|---|---|---|
| 【出来高】 | INNER | E | ○ |
| 【品目】 | LEFT | R | |
| 【受注DTL】 | LEFT | E | ○ |

2.テストケース

| | |
|---|---|
| T1-1 | Rのデータがあるケース |
| T1-2 | Rのデータがないケース |
| T1-3 | 受注DTLのデータがあるケース |
| T1-4 | 受注DTLのデータがないケース |
| T1-5 | 【品目】品目区分 = 9:試作<br>AND【受注DTL】完了状態 NOT IN ( 3:欠完了, 7:過完了, 9:定量完了) |
| T1-6 | 【品目】品目区分 <> 9:試作 |

図１．テストデータ自動生成に必要なインプットの検討イメージ

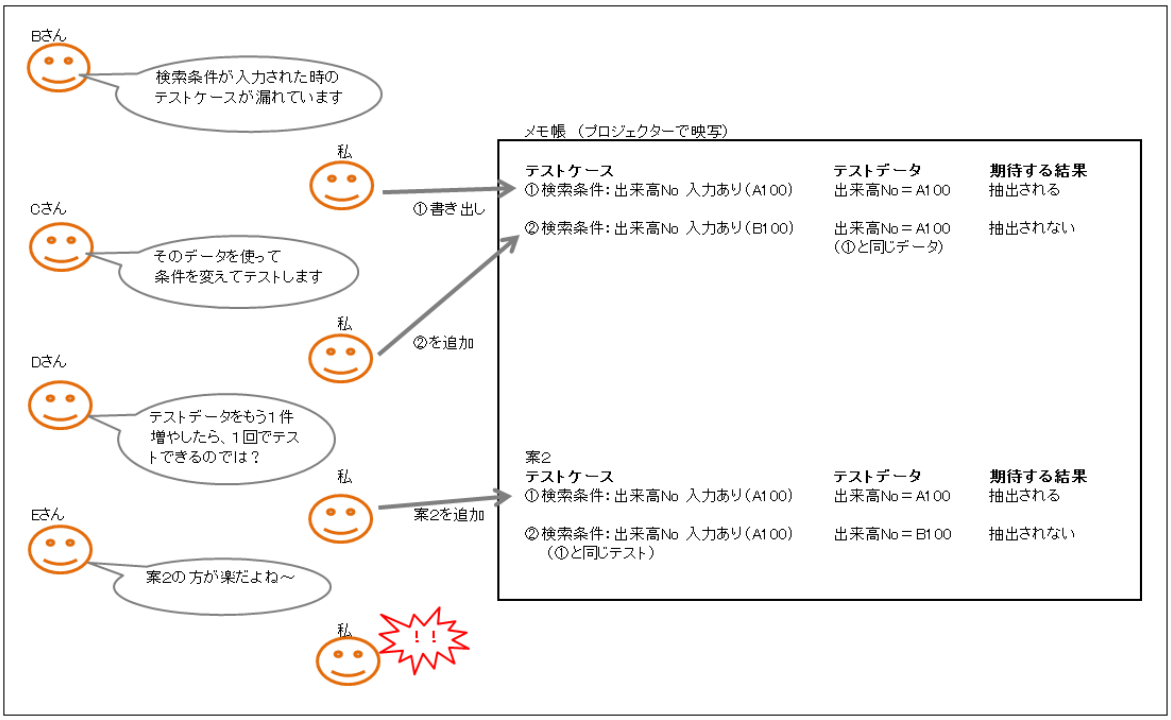



図２．議論中の様子

① 実物で検討していた

抽象的なモデルで検討していると一般的な例は出せるがプロジェクト固有の具体的な情報は少ない。例えば受注データをイメージしたときに受注データのステータス（状態）として完了/未完了は思いつくが、欠完了（予定数量より少ない状態で終わること）/過完了（予定数量より多い状態で終わること）は出てこない。実物で検討していると、受注データのステータスで判断する仕様があった場合に「欠完了/過完了の場合もそれでいいのか？」という疑問が生まれる。
実物で検討することは他の人の知識を疑似体験でき、情報が多く与えられることになる。

●図１を見た結果の反応
Ａさん：（疑似体験の結果、その後のテストをすることを考え）
「どうやってテストする？」
Ｂさん：（情報量が多かった結果、テスト設計そのものを考え）
「テストケースがもれている？」

実物で検討すると他の人の知識を疑似体験し、情報量が多いことにより、脳がより多くの刺激を受け疑問・意見が発生し、結果として議論が活発になる。

② 可視化していた

会議中はメモ帳（Windows 標準テキストエディタ）をプロジェクターで映写しながら、メンバーの意見をテキストにして共有しながら進めていた。誰かの考えを文字で見ることにより、各自が直感的に感じる違和感や自分の考えとの相違を出せる状況にあった。

●その後の議論（図２参照）
前述のＢさんの指摘を受け、私はＢさんが思うテストケースを聞き出し、

メモ帳に記述した。
その後、以下の発言が出た。

Ｃさん「そのデータを使って条件を変えてテストする。」
Ｄさん「データを２件作れば１回の実行で２つのテスト項目の結果を得られるのでは？」
Ｅさん「１回でできた方が楽だよね」

ここまでの発言を聞いて、今まではテストデータ作成に時間がかかるのでＣさんのやり方をしていたが、自動生成できるなら時間がかからなくなるのでテスト回数を減らすＤさんのやり方がよいというアイデアに気付いた。
参加していた全員が同じテキストを見ていたからこそ､このアイデアが得られた。

誰かの意見・疑問を書き出すことにより、それを見た人の頭の中で自分の知識と混ざり、その結果が意見となって発せられる。さらにそれを書き出すとまた次の人の頭の中に入り意見が生まれる。これが繰り返し起こり、新たな知見やアイデアが生まれる。

可視化せず会話だけの議論だと、自分が考えている最中に議論が進んで追いつけなくなったり、最初に議論した内容が残っていないので同じ議論を繰り返してしまったりと漏れ・ムダが多くなる。

③ 検討中は時間が超過しても止めない

ＷＧ活動は毎週２時間程度の会議を行う。２時間の使い方は事前にアジェンダを用意しているが、予定通りに進まなくても、その時、盛り上がっている議論を優先することにしていた。
上述の①、②が幾度も繰り返されることにより複数のアイデアが生まれ、よりよい案が採用される。そのための時間を確保することは必要不可欠である。

今回の例では議論の目的はテストデータを自動生成するための入力を検討することだった。しかし議論の結果テスト回数を減らすアイデアが出た。最初からテスト回数を減らすアイデアを出そうとしていたわけではない。

このことから、３つの取り組みがＷＧでアイデアを出すには非常に効果的であることがわかった（改善のアイデア）。この取り組みを続け、ＷＧメンバーで週１回、２時間の会議を２ヶ月ほど実施し、テストデータ自動生成の仕組みをまとめた。

(5-2) テストデータ自動生成システムの仕組み

検討の結果、テストデータの自動生成は使用する区分値等、テストケースに紐付くデータが必要になることがわかり、単なるテストデータ自動生成システムではなく、テスト設計システムとしてテストケースと利用するテストデータを一元管理できる仕組みとした。

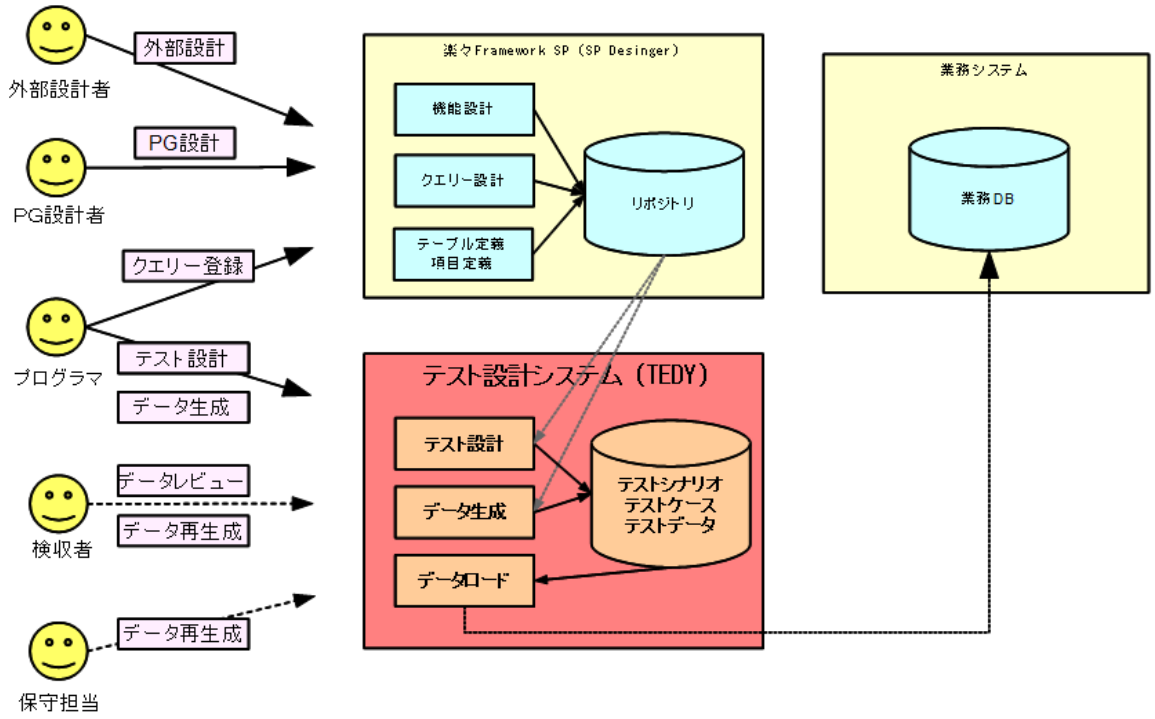



図３．テスト設計システム 概要図

＜主な仕様＞

・テストケースを設計しケース毎に必要なテストデータを生成する

⇒課題１．(a)解消

・テストデータが必要なテーブルを選択するだけで整合性のとれたレコードが生成できる

⇒課題２．(c)解消

・テストケースに必要な項目は値を指定し、テストに使わない項目は型・桁の制約に則った値を生成する

⇒課題２．(d)解消

・繰り返しテストの際、必要なテストケースを選択して再生成やテスト環境への反映が簡単にできる

⇒課題１．(b)解消

(6) 変更後の状態や改善効果

(6-1)改善推進部門として、ワーキンググループ活動の成果を出すためのノウハウを確立し、他のワーキンググループでも活用し成果を生み出している。

(6-2)テスト設計システムを実現することにより、以下の効果が見込まれる。

・テストデータ作成作業の効率化によるプログラム開発生産性向上

・自動テスト対象の拡大によるプログラム開発品質向上

(7)改善活動の妥当性確認

現在、テスト設計システムの開発が完了し、システムテストを開始している。

本発表の時には本システムを活用した効果を確認し、経過とあわせて報告する。

以　上

## 3C2「テスト自動化を現場に普及するためのオフショア活用」小林道央(インテック)

〈タイトル〉: テスト自動化を現場に普及するためのオフショア活用

〈サブタイトル〉:

〈発表者〉
氏名（ふりがな）: 小林 道央（こばやし みちお）
所属：株式会社インテック　技術本部　技術部
〈共同執筆者〉
氏名（ふりがな）: 池田 浩明（いけだ ひろあき）
所属：株式会社インテック　技術本部
氏名（ふりがな）: 相澤 武（あいざわ たけし）
所属：株式会社インテック　技術本部　技術部
氏名（ふりがな）: 小林 麻美（こばやし あさみ）
所属：株式会社インテック　技術本部　技術部
氏名（ふりがな）: 加藤 康記（かとう やすのり）
所属：株式会社インテック　先端技術研究所　研究開発部
氏名（ふりがな）: 西川 美紀（にしかわ みき）
所属：株式会社インテック　先端技術研究所　研究開発部
氏名（ふりがな）: 高木 慎也（たかぎ しんや）
所属：株式会社インテック　先端技術研究所　研究開発部

〈要旨〉
テスト自動化を多くの現場に広く普及し、効果的に利用するためには多くの課題がある。テスト自動化のハードルを下げるため、当社では 2012 年下半期にテスト自動化ツール(TaaS)を自社開発して、テスト自動化に比較的簡単に取り組むことを可能にした。その結果、2012 年下半期から 2013 年上半期までの間に 10 数件の適用実績があった。さらに多くの現場に普及していくために、2013 年 8 月にテスト自動化作業を委託できるチームをオフショア子会社に設立した。オフショアを有効に活用するために、テスト対象のシステムに詳しくないオフショアメンバーでも、必要最低限の情報でテスト自動化作業を行うことができるプロジェクト・オフショア間の分業方式を構築した。本発表では、構築した分業方式の効果と今後の課題について、先行案件の事例に基づいて紹介する。

〈キーワード〉
ソフトウェアテスト、テスト自動化、テスト設計、テストツール、分業方式

〈想定する聴衆〉
SEPG、プロジェクトマネージャ、ソフトウェアエンジニア

〈適用状況〉
□多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
■その他（多用までいかないが適用され始めている段階　　）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（　　）
□自プロジェクトのみ

〈発表内容〉

(1)背景

- ビジネス変化のスピードは年々加速し､それを支援する IT システムも迅速な対応が必要であり、開発・改修期間の短期化と生産性向上が迫られている。
- 一方、スマートデバイスや多種のブラウザなどの出現により、ユーザインタフェースのマルチデバイス化が進んでいる。それに伴い、アプリケーションが動作保証すべき環境も多種多様となり、また、それらの製品ライフサイクルも短くなっている。
- 従来の手動テストに頼っていては､このような IT の多様化やスピードに対応するためのテスト工数が爆発的に増え、いずれ対応が困難になることは明白である。早期にテスト自動化技術を確立し、現場に普及・展開し、テストの生産性向上を実現する必要がある。

(2)改善前の状態

- 2011 年から社内システムの開発・保守プロジェクトにおいて、テストの標準化と自動化の試行に取り組んでいる。2013 年下半期にはクラウド型テスト自動化環境 TaaS（Test as a Service）の構築と利用を行ってきた。
- TaaS は開発したアプリケーションの機能テスト､回帰テストを効率的に行うことを目的としたツールである。TaaS は、(アプリケーションに自動的にテスト実行を行わせる）テストスクリプトを自然言語によるキーワードを入力した専用の Excel シートから自動生成しているため、次の 3 つの利点がある。
  1. テストスクリプト言語の専門知識が不要なため、容易な習得が可能
  2. TaaS は可読性が高いため、レビューや修正を容易に可能
  3. テストスクリプトに比べ、TaaS シナリオの標準化が容易なため、分散開発が可能
- TaaS は、テスト自動化のハードルを下げ、テスト自動化が初めての場合でも比較的簡単に取り組むことを可能にする｡その結果これまで中小規模プロジェクトを中心に 10 数件の適用実績があり、効率面でも相応の効果が表れている。
- ただし、TaaS によりテスト自動化のハードルが下がったとはいえ、テスト自動化技術を現場により広く、速やかに展開するためには次の 2 つの課題がある。
  1. 手動テストに比べると、初期工数が大きい
  2. プロジェクト外のメンバーにテストスクリプト作成を依頼する場合、テスト設計者の意図通りに作成できていないことがある

(3)改善前の状態をもたらした原因（因果関係）

- TaaS 開発のコアメンバーが TaaS スクリプト作成を社内システムの開発・保守プロジェクトで担当した経験や TaaS 適用案件の事例から上記 2 つの課題の原因を分析した。
  1. 初期工数の大きさの原因
     - テストスクリプトは誰でも比較的容易に作成できるものの、効率的な作成やデバッグにはノウハウが必要である。しかし、ノウハウを集約する仕組みがなかったため、ノウハウが経験者に閉じてしまいがちで皆が同じような壁にぶつかり、工数が増大していた。
       ※ なおテスト自動化は繰り返しテスト実行を行うことで初めて工数削減効果がでるため、初回実行のみを比べた場合、手動テストに比べて工数が大きくなることはある程度しかたがないといえる。
  2. テスト設計者の意図と異なるテストスクリプトが作成される原因
     - テスト仕様書に曖昧さや属人的な部分が多い。テスト仕様書のテンプレートや項目はプロジェクトごとに決められ、取り決めが曖昧なため、その内容は担当者に

よってばらつき（抜け・漏れ・粒度）がある。自動テストの場合、テストの内容や手順を詳細にスクリプトとして記述する必要がある。曖昧で属人性のあるテスト仕様書に基づいてプロジェクト外のメンバーがテストスクリプトを作成する場合、作成担当者によって解釈が異なるケースがある。

### (4)計画した変更内容

- 上記２つの課題を解決し、テスト自動化技術を現場に広く、速やかに展開するために以下の活動に組織的に取り組むことを計画し、順次実施している。
  1. TaaS によるテスト自動化の専門チームをオフショア子会社内に組織した。TaaS に特化することでノウハウを蓄積し、低コストでテスト自動化作業を請け負うことで、初期工数が大きいという課題の解決を目指す。
  2. オフショアを有効活用するためには、プロジェクト外のメンバーでもテスト設計者の意図通りのテストスクリプトを作成する仕組みが必要である。そのために、テスト仕様書の標準化（テスト観点や設計項目など）を行った。また、テスト設計者の意図を反映できるようなプロジェクト・オフショア間の分業方式を構築して、テストケースの曖昧さや属人性を排除した。

### (5)変更の実現方法

- 上記の２つの取り組みについて、詳述する。特に２つ目の取り組みについては、先行案件に適用した事例に基づいて具体的に述べる。

1. オフショアの活用
   - TaaS 開発のコアメンバーがオフショア子会社の専門チームメンバーに対して OJT を実施した。
   - オフショアメンバーと TaaS 開発コアメンバー間の連携を密にとり、オフショアメンバーだけでは解決が難しい問題に対してコアメンバーが迅速に支援・問合せ対応できる体制を作った。
2. プロジェクト・オフショア間の分業方式の構築
   - 検証対象それぞれに対して検証方法を標準化した。
     - 画面遷移の検証であれば、スナップショットのみ。データ登録の検証であれば、スナップショットで登録データの値の検証を行う、などの標準を決める。
   - テスト仕様書からテストスクリプトを作成する前に、テスト仕様書を「OK ボタンをクリックする」などの１つ１つの操作単位に詳細化した中間成果物（テストシナリオ）を作成するプロセスを導入し、全体として次のような分業方式でテスト自動化作業を進めた。
     - テスト仕様書作成（プロジェクト）⇒ テストシナリオ作成（オフショア）⇒ テストシナリオレビュー（プロジェクト） ⇒ テストスクリプト作成（オフショア） ⇒ テストスクリプトレビュー（プロジェクト）
       ※()内は担当組織。ただし、先行案件ではプロジェクト担当のプロセスは、コアメンバーが担当し、分業方式の効果の検証を行った。
     - テストシナリオの作成とレビューのプロセスは、例えば、テスト仕様書で「ログインする」という手順があり、そのログインがベーシック認証の場合、ログインダイアログに ID とパスワードを入力してログインしているか、URL に ID とパスワードを組み込むことでログインしているか確認するため。また、データ登録、変更、削除を行う手順があった場合、登録、変更、削除全てで同一のデータを使用して検証しているか、それぞれ異なるデータを使用して

検証しているか確認するために行う。確認の上、テスト設計者の意図と異なる手順だった場合に、レビューバックを行う。

3. テストスクリプトのレビューバックの頻発と対策の実施
    - 上記の分業方式を先行案件に適用したところ、案件の初期段階からテストスクリプトのレビューバックが頻発し、進捗遅延が発生した。
    - 原因究明のために、オフショアメンバーの作業方法の確認を行ったところオフショアメンバー内での内部レビューを行っていないことが判明した。またレビューバックの内容を分析したところ、比較的単純なルール違反が大半をしめていた（画面上の 10 個のボタンが無効化状態であることを検証すべきところで、9 個しか検証していないなどのルール違反）。
    - 対策として、オフショアのリーダを内部レビュー担当者とし、また内部レビューでのチェック観点としてオフショアメンバーが違反しやすい点を整理したチェックリストを作成した。
    - しかしながら、この対策は十分な効果を得られず、案件の最後までレビューバックが一定数存在した。

### (6)変更後の状態や改善効果

- 下記の効果を得られた。
    1. 初期工数の大きさという課題への効果
        - テストシナリオの段階で、テスト設計者とテストスクリプト作成者の意図のずれに気が付くことができることで、手戻り工数を削減できた。ただし、テストスクリプトでのレビューバックが頻発したことで、プロジェクトのレビュー工数、オフショアのテストスクリプト作成工数が大きくなり、オフショアの活用による低コストのテスト自動化作業という効果は得られなかった。
        - 今後の重要な課題としては、比較的単純なルール違反によるレビューバックを減少させる仕組み作りの検討と実施がある。
    2. テスト設計者の意図と異なるテストスクリプトが作成される課題
        - 上記の施策によりテストスクリプトはテスト設計者の意図通りのものとなった。テスト仕様書の情報からテストシナリオレビュー、テストスクリプトレビューと 2 通りのチェックを行うことでプロジェクト外のメンバーでもテスト自動化作業を担当できるという効果が得られた。

### (7)改善活動の妥当性確認

- 本施策は、テスト自動化を社内の多くの現場に広く普及し、効果を上げることが目的であり、現在はその途上にある。目的達成のための課題がある程度明確になり、それらの課題への対策を取りながら組織的に推進・運営する体制はできつつある。
- 当面は、テスト自動化の効果を見える化し、それをもって各現場の理解を得ながら順次展開を図っていくことになるが、普及がある程度進んだ段階ではテスト自動化を梃子に現状のテストプロセスや体制などを大きく変革していくような判断も必要となるだろう。
- また、将来的にはテストの自動化だけでなく、その技術的な成果をもとに順次他の工程の自動化へと発展させていく計画である。自動化の追求により生産性を飛躍的に向上させることが目的の一つであるが、一方で単純・繰り返し作業をできるだけ IT に任せ、技術者が本来行うべき知的な業務に専念できるようにすることも大きな目的と考えている。

3C3「開発時に構成管理 Tool のブランチを有効活用して、品質を高める。」長橋敦(シナジーテック)

開発環境に存在する「構成管理 Tool」を、製造工程プロセスに組み込み「有効利用」し、品質を高め、後工程を安定させる

〈タイトル〉:
開発時に構成管理 Tool のブランチを有効活用して、品質を高める。
〈サブタイトル〉:
なし

〈発表者〉
氏名（ふりがな）: 長橋敦
所属: 株式会社シナジーテック ※ソフトハウスです
〈共同執筆者〉
なし

〈要旨〉
「単納期」「垂直立ち上げ」「多機能」といったプロジェクトが多くなって来た昨今、急遽開発エンジニアを増やす為「協力会社」に発注することが多いかと思います。また、最終工程になるにつれ開発者は減り、総合試験などでは担当者のみが残され大変な思いをするといったプロジェクトも多く見かけます。本発表は「後工程を安定させる為に、中工程を仕組みでカバーする」といった内容になります。また、「依頼先の協力会社さんに行ってもらえるプロセス」にもなっておりますので、協力会社と直接やりとりする担当者に是非聞いていただきいと思います。

〈キーワード〉
構成管理、マージ、コーディング、コードレビュー、ペアプログラミング

〈想定する聴衆〉
協力会社さんとやりとりするメーカー側の担当者、プロジェクトマネージャー、ソフトウェアエンジニア

〈適用状況〉
■多用されている段階、□適用できる段階あるいは初めて適用する段階
□適用するにはさらに検討を必要とする、□着想の段階
□その他（ ）

〈適用可能性に関する制限〉
■汎用性がある、
□類似プロジェクトにも適用可：具体的な類似点（ ）
□自プロジェクトのみ

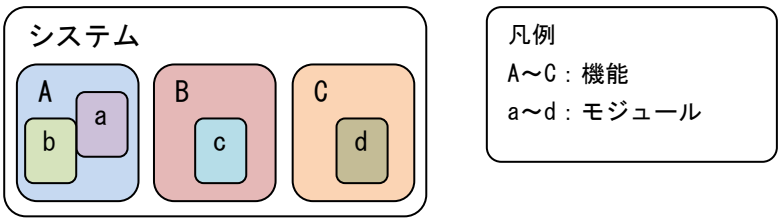
〈発表内容〉

(1)背景

私は５名～１０名前後で行うプロジェクトのプロジェクトリーダーを多く行ってきました。
１つのプロジェクトには複数の機能があり、それを複数の担当が作成して、最後にひとまとめにします。この「まとめる」時がわりかし曲者で、いつも注意深く行ってきました。
「この作業をもう少しシステム化出来ないだろうか」というのが今回の改善作業のキッカケでした。

(2)改善前の状態

たとえば、以下のようなシステムがあったとしましょう。




A、B、C をそれぞれ別の担当に依頼し、動作確認後、マスターファイルにマージしてもらう　というのが今までの方式でした。
しかしながら、マージ時に「古いものをマージしてしまった」「マージミス」「マージ漏れ」など、マージ前の各機能単体は問題ないのですが、マージ時にミスにより、気付かずにバグを埋め込んでしまい、そのバグの発見が遅れたなど苦労をしていました。
そこで、マージ時に「マージの内容をチェックする」といった仕組みも入れたのですが、「マージミス」「マージ漏れ」に関してはあまり減りませんでした。

(3)改善前の状態をもたらした原因（因果関係）

「なぜマージ時のミスは減らないのか？」「私が担当者から預かってマージする時にはマージミスは起こらないのになぜ？」と思い、色々考えてみたところ、「ちょっとした意識の差がマージ漏れを真似ているのでは？」と思うようになりました。
＃「私がマージ上手である」とか、「いい加減に仕事をしている人がいる」
＃という話ではありません。

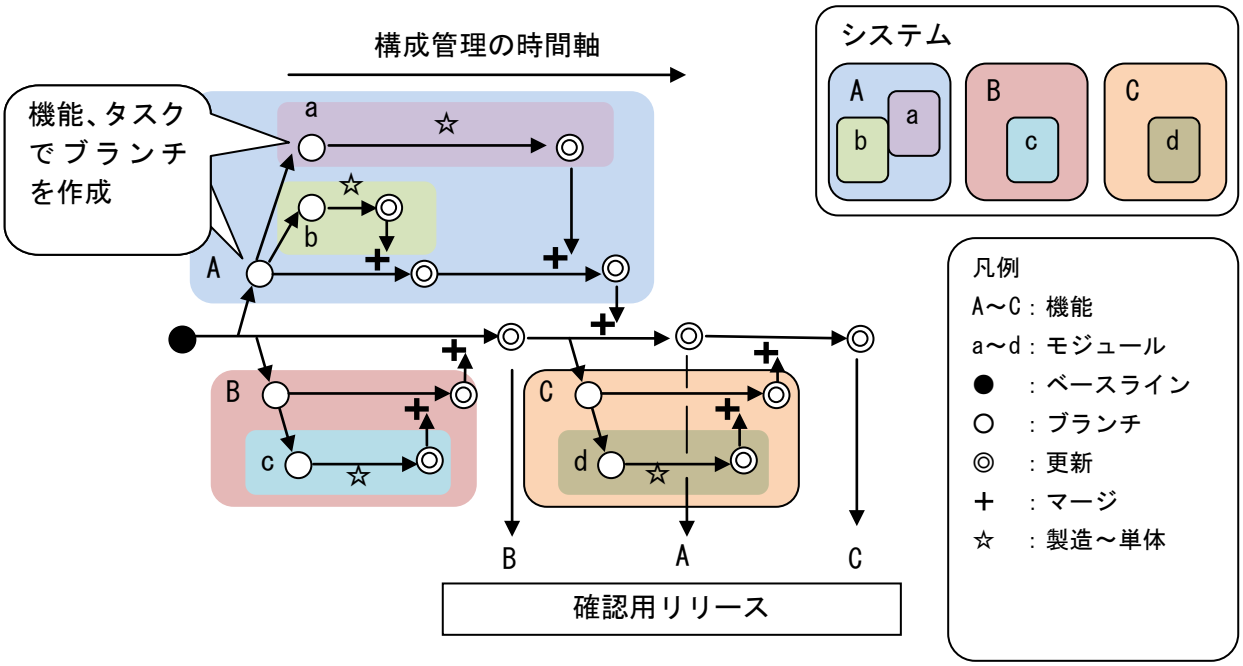
私がマージを行う際には、
・自分が作った機能でないので注意深くなれる（あれは不要？これは修正しない？など）
・マスターファイルがぐちゃぐちゃになったら大変
といった事が頭の中で考えていたことが判りました。

(4)計画した変更内容

それであれば、構成管理 Tool の「ブランチ機能」を使用して、各ブランチに「ブランチ責任者」を決め、「ブランチ責任者がマージを行う」というルールを決めてしまおう　と考えました。

(5)変更の実現方法

具体的に構成管理 Tool を使った今回の「ブランチ構造」を書くと以下のようになります。




各色のブロックが「ブランチ責任者」の範囲です（○がブランチであり、責任者を決めます）。各ブランチ責任者は、そこから派生して作成しているパーツが戻ってくる際、ブランチ責任者自身がマージを行います。

たとえば、マスターから派生した「B 機能のブランチ責任者」は、c モジュールの担当がデバグ完了した時点で、c モジュールの担当に B ブランチへのマージをさせるのではなく、B ブランチの担当者が c モジュールの担当者から依頼を受けてマージすることになります。

これにより、「マージ時にコードレビュー」を行う事も出来るようになり、しかも、ブランチ責任者の判断で、「ミスが多いからマージしない」と判断することができるのです。

自分は常々「コードレビューでしか検出出来ないバグがある」と思っています。例として「swich case」の「case 漏れ」です。これは、コンパイラでも静的解析 Tool でも検出しません。なぜなら、「不要なのか」「漏れているのか」が Tool では判断できないからです。

コンパイラ　　→　文法チェックのみ
机上デバグ、コードレビュー　→　論理チェックが可能

(6)変更後の状態や改善効果

今回はブランチ責任者を配置し、意識を高めるのを目的としていましたが、「マージ時のコードレビュー」といった思いもよらぬ「副産物」が付いて来たことで、「マージミス」「マージ漏れ」以外にも、バグを検出する事が出来ました。

(7)改善活動の妥当性確認

また、マージ作業は必ず「２人」で行います。ブランチ責任者のみで行うと、効果が半減します。ならなら、マージする内容に不明点があっても「面倒だからマージしてしまえ」

ということになり、意味がなくなってしまうからです。一緒に行う事で「その場で聞ける」という「気軽さ」「手軽さ」が良いように感じます。

このマージ時のコードレビューが功を奏し、デバグ時、評価事といった「Ｖ字の後半」がとても安定し「発見しづらい不具合」が減ったのも実感しています。

(8)今後の課題

現在は、この仕組みを発展させ、ドキュメント、テスト結果（エビデンス）、レビュー議事録などの「ＯＵＴＰＵＴ」をマージ時にチェックし、「プロセスの抜け漏れを仕組みでカバーする」といった事が進められればと考えています。