Empowered by Innovation NEC

派生開発で USDMとDRBFMをミックスして 一気通貫で品質確保する

～ おめさん、なじらね？ ～

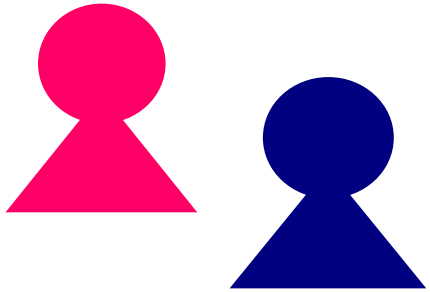
2011年2月18日

NECソフト株式会社　新潟支社

酒井 賢

おめさん、なじらね？

- 「おめさん、なじらね？」は新潟弁で次のような意味があります。
- おめさん [omesan]
  - 【意】あなた、お前さん
    「おめ」より丁寧ないい方です。



- なじ [naji]
  - 【意】調子・様子・具合
    新潟弁の基本で相手を思いやる・相手を気づかうこと
    を代表する言葉です。

目次

1. 自己紹介
2. プロジェクト概要
3. プロジェクトのおかれた状況（2006年当時）
4. プロジェクトの問題点
5. 抽出した課題
6. 課題を解決するための施策
7. スケジュールと施策のタイミング
8. 施策1　変更要求仕様書(2006年～)
9. 施策2　心配点シート(2008～)
10. まとめ
11. 今後の課題
12. 最後に

1. 自己紹介

- 1993年入社で、今年18年目です。
  - テスターを2年経験して、プログラマー5年を経験しました。
  - そのあとサブリーダーを経て2000年からプロジェクトリーダーをやっています。
- 初めてさわったパソコンはPC－6001でした。
- 主にプリンターのユーティリティソフトの開発に携わってきました。
  - 入社して最初に与えられた仕事は、Windows NT3.1のプリンタードライバーのテストでした。

- この10年間で40案件程度を担当しました。
  - 大規模なシステムの一機能を担当するというものではなく、多いときでも5名程度のメンバーで、設計からテストまでを通して行うような形態の開発だけをやってきました。

2－1．プロジェクト概要（１）

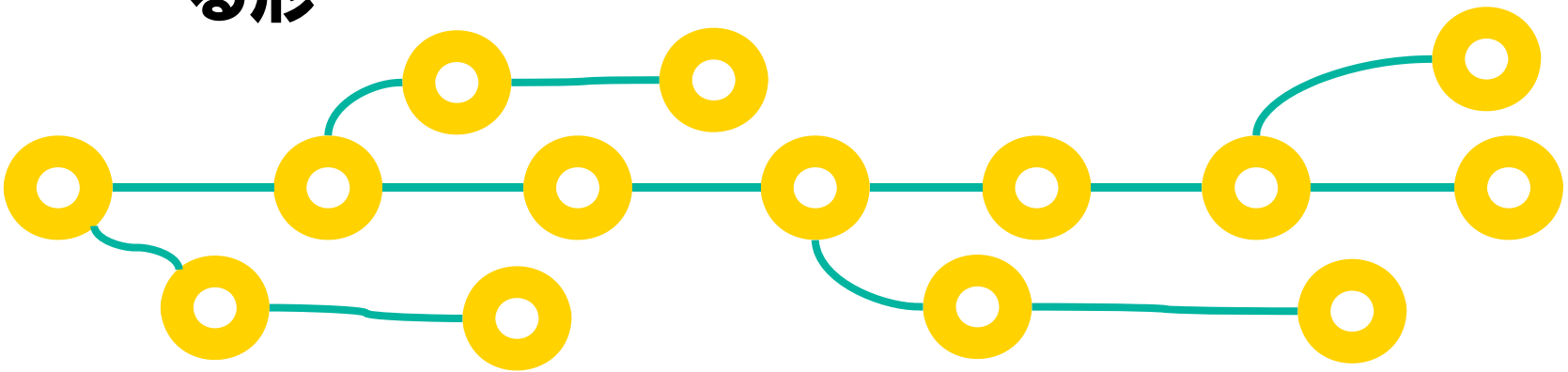
- オフィス機器（ハードウェア）に添付されるWindows上で動作するユーティリティソフトウェアの開発
- ソフトウェア単体では用をなさず、ハードウェアと通信を行うことで利用することができる
- 要件の仕様化から製造、テストまでの工程を担当
- お客様が新規開発を行い、当方が保守という形で2003年に引き継いだ
- 機種対応開発の他に、メンテナンスリリース、特定用途向けの開発がある

いわゆる派生開発

2−2. プロジェクト概要(２)

派生開発とは

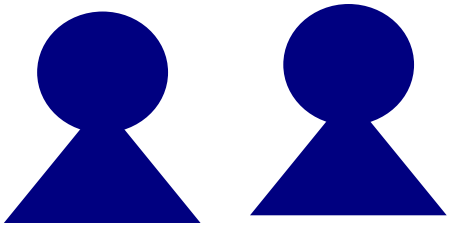
- **既存**製品(システム)のソースコードがあって
- 新しい機能を**追加**したり
- 既存機能の一部を**変更**したりして
- **新しいバージョン**のソフトウェア製品として開発する形

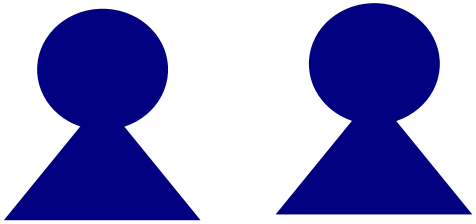


※ソフトウェア・ピープルVol.8「失敗しない派生開発」(株)システムクリエイツ清水吉男氏の定義を引用しています

3. プロジェクトのおかれた状況（2006年当時）

- 引き継ぎ段階で提供された以下の成果物からリバースエンジニアリングした仕様書
  - ソースコード
  - 実行モジュール
  - ヘルプ
- サポートOS
  - 日本語版／英語版あわせて12エディション
- 規模：120KL
- 開発言語：VB6／VC++6
- モジュール数：21モジュール
- メンバー ：4名（モジュール別に担当を割り当て）
- サポート機種種別：日英で12機種
- 全OS、全機種対応のシングルソースコード





4−1. プロジェクトの問題点 1

- デザインレビュー後に仕様の不明点/矛盾が判明
  - 後戻りにつながる
- どうして今の仕様になったのか経緯がわからない
  - 互換性を維持するための譲歩した設計が必要
- ソースコードのどこを変更するとどこに影響するかがわからない
  - デグレードが発生

要求を仕様化する技術、**変更点の抜け漏れ防止**
変更の追跡が必要

4−2. プロジェクトの問題点 2

- **OSに依存する問題が発生する**
  - 設計時にすべてのOSで動作可能かの判断は難しい
- **ハードウェアに依存する問題が発生する**
  - 機種によって微妙に仕様が異なる
- **非機能要件（操作性、性能）が実現できていない**
  - 総合テスト時に問題発覚

問題発生の未然防止の必要性

5. 抽出した課題

- どうやったら要求を正しく仕様化できるのか？
- どうやったら変更点を追跡できるのか？
- どうやったら変更の影響範囲を特定できるのか？
- どうやったら問題発生を未然に防止できるのか？

解決できるのか？？？

6. 課題を解決するための施策

施策1

要求の仕様化技術　変更点の追跡　抜け漏れ防止

- USDM(Universal Specification Description Manner)の表記法を取り入れる
- メンバーが持ってきた本がきっかけ
- 私たちは「**変更要求仕様書**」と呼んでいます。

「コンサルタントが教える戦慄の仕事術 失敗しない派生開発」(Software People Vol.8)

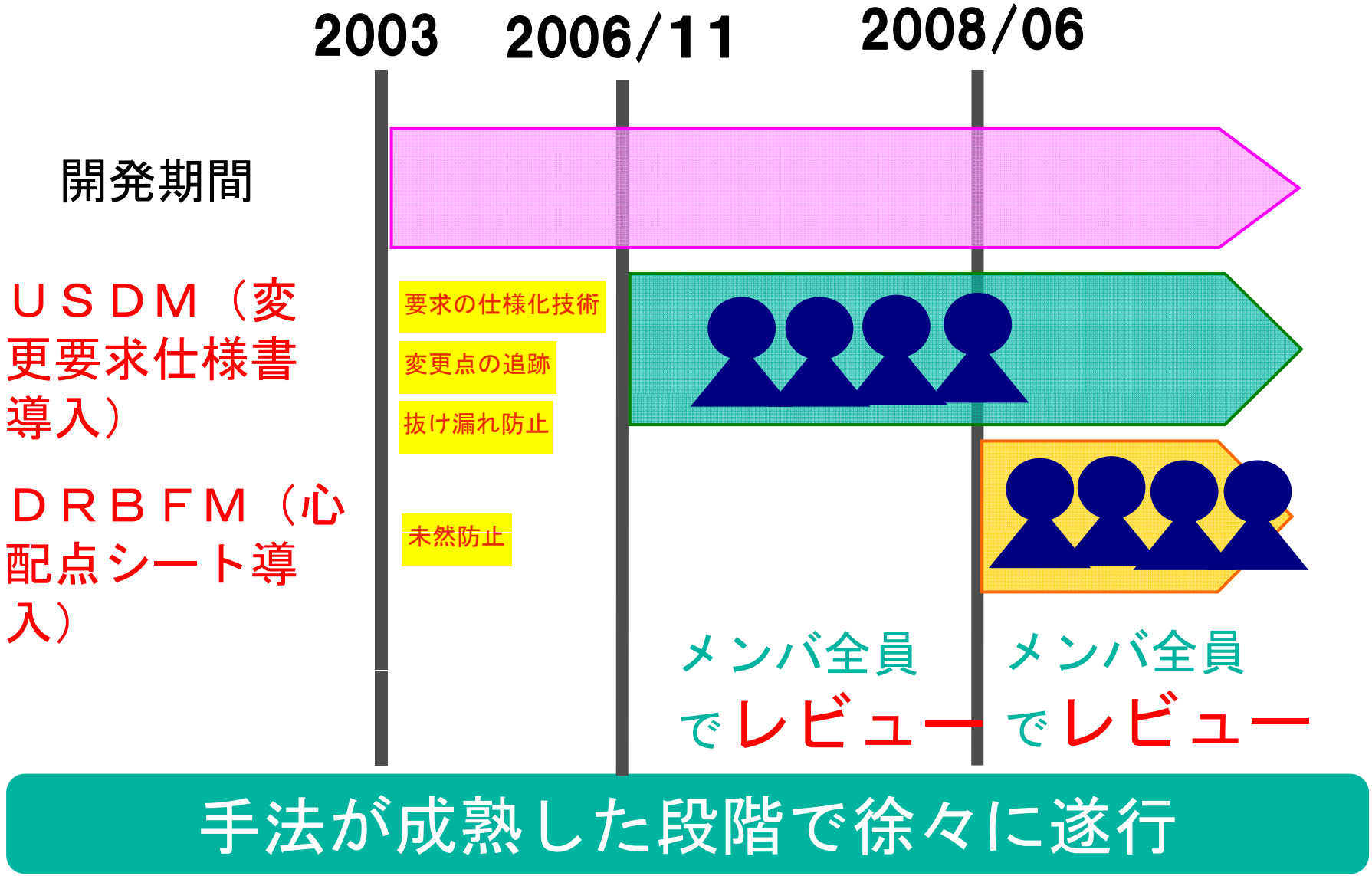
施策2

未然防止

- DRBFM (Design Review Based on Failure Mode) を取り入れる
- 私たちは「**心配点シート**」と呼んでいます。

どちらも根底には徹底的なレビュー

7. スケジュールと施策のタイミング



手法が成熟した段階で徐々に遂行

8－1．施策1　変更要求仕様書（2006年～）

要求の仕様化技術　　変更点の追跡　　抜け漏れ防止

USDM：Universal Specification Description Manner

- （株）システムクリエイツ 清水吉男氏が開発。
- 日本情報システム・ユーザー協会（JUAS）が表記法として「要求仕様定義ガイドライン」で取り上げている

要求	MAL01	受信および送信した電子メールを
	理由	メールが多くて、関連するメールを探せない
	要求	MAL01-01　検索対象のメールボックスを指
		理由　自動で振り分けられることがあ
		MAL01-01-1　複数のメールボックスを同
		MAL01-01-2　グループ化したものに任意の名
		MAL01-01-3　グループ化したものに対してキ
		MAL01-01-4　……

システムに対する要求の階層構造を表形式で記述し、仕様漏れを防ぐ記法

- 1つの表に全ての要求・仕様を記述する
- 要求を2～3段の階層構造で定義する
- 各要求に対して、その要求が必要な理由を併記する
- 最下位の要求に対して仕様（その要求を実現するためのシステムの振る舞いHow）を記述する
- 要求・仕様にはユニークなIDを付与する

もうひと工夫

モジュール単位に列を作成してトレーサビリティマトリクス化

8－2. 施策1 変更要求仕様書 導入

- 4プロダクト同時開発
- 早速、本を数冊買ってきてメンバーに配布
- **勉強会**を実施
- 見よう見まねでシートを作成して、やってみた
  - ●変更要求列は**本の通り**に作成
  - ●トレーサビリティマトリクスは**ソースファイルを列**にした

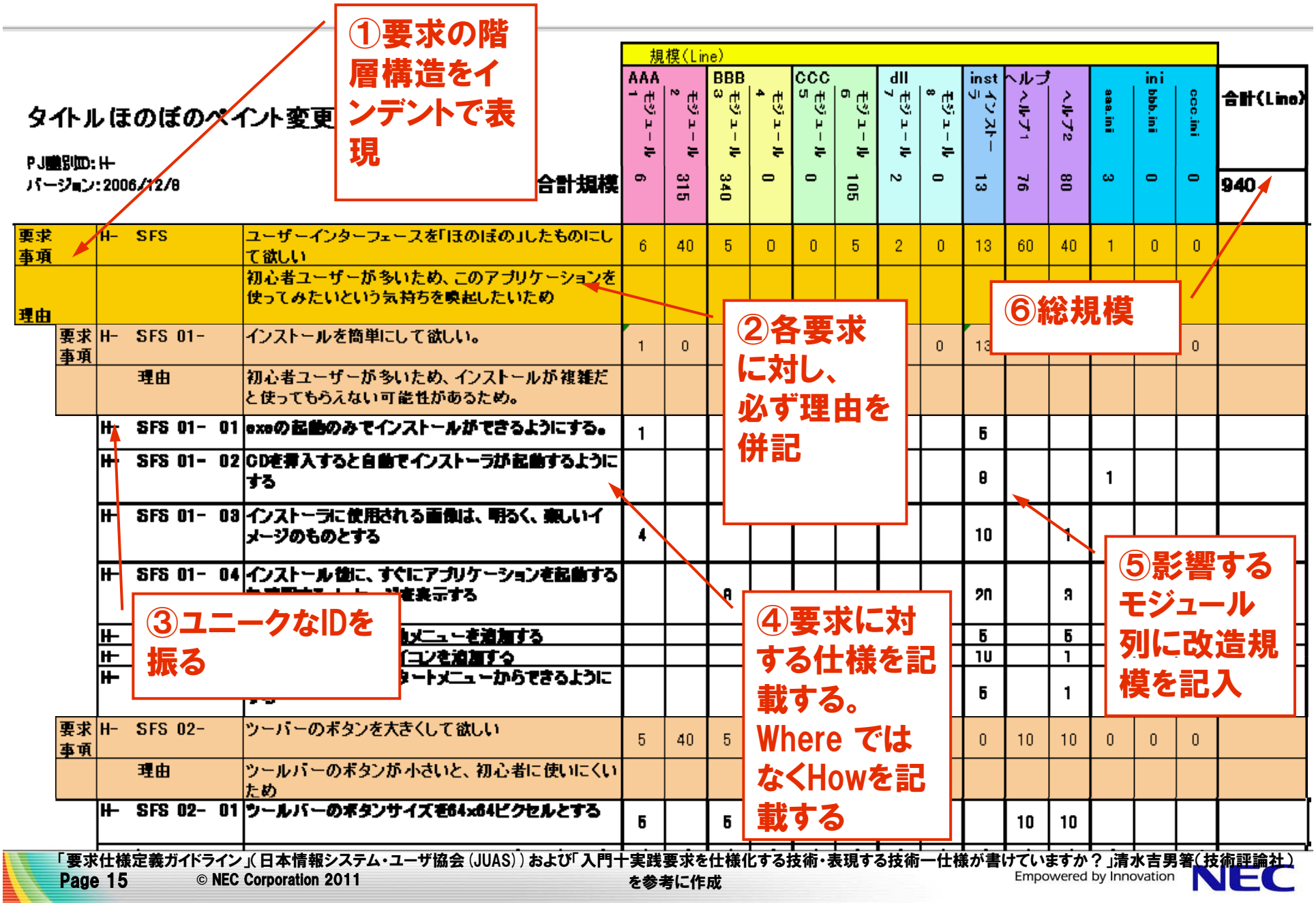
- 効果はあった。同時開発自体は遂行。

ただし、

- なかなか**理由**が書けない。要求から理由が導き出せない
  - ●**お客様を巻き込む**必要性を痛感
- ソースファイル単位では**列が多すぎ**て結局、影響が追い切れない

カイゼンの余地あり

8－3. 施策1　現在の変更要求仕様書の例



タイトルほのぼのペイント変更

PJ識別ID: H-
バージョン: 2006/12/8

			合計規模	規模(Line) AAA モジュール1 6	AAA モジュール2 315	BBB モジュール3 340	BBB モジュール4 0	CCC モジュール5 0	CCC モジュール6 105	dll モジュール7 2	dll モジュール8 0	instラ インストーラ 13	ヘルプ ヘルプ1 76	ヘルプ ヘルプ2 80	.ini aaa.ini 3	.ini bbb.ini 0	.ini ccc.ini 0	合計(Line) 940
要求事項	H- SFS	ユーザーインターフェースを「ほのぼの」したものにして欲しい		6	40	5	0	0	5	2	0	13	60	40	1	0	0	
理由		初心者ユーザーが多いため、このアプリケーションを使ってみたいという気持ちを喚起したいため																
要求事項	H- SFS 01-	インストールを簡単にして欲しい。		1	0						0	13					0	
理由		初心者ユーザーが多いため、インストールが複雑だと使ってもらえない可能性があるため。																
H- SFS 01- 01		exeの起動のみでインストールができるようにする。		1								5						
H- SFS 01- 02		CDを挿入すると自動でインストーラが起動するようにする										8			1			
H- SFS 01- 03		インストーラに使用される画像は、明るく、楽しいイメージのものとする		4								10		1				
H- SFS 01- 04		インストール後に、すぐにアプリケーションを起動する[illegible]を表示する				8						20		3				
H-		[illegible]メニューを追加する										5		5				
H-		[illegible]アイコンを追加する										10		1				
H-		[illegible]スタートメニューからできるようにする										5		1				
要求事項	H- SFS 02-	ツーバーのボタンを大きくして欲しい		5	40	5						0	10	10	0	0	0	
理由		ツールバーのボタンが小さいと、初心者に使いにくいため																
H- SFS 02- 01		ツールバーのボタンサイズを64x64ピクセルとする		5		5							10	10				

「要求仕様定義ガイドライン」(日本情報システム・ユーザ協会(JUAS))および「入門十実践要求を仕様化する技術・表現する技術一仕様が書けていますか？」清水吉男著(技術評論社)を参考に作成

８－４．施策１ 変更要求仕様書 導入（２）

- Excelを使用する。１つのファイル（シート）にすべての要求と仕様（振る舞い）を記載する
- 変更要求仕様書の要求事項、理由、仕様は、必ずプロジェクトリーダーが書く
- 影響モジュール、規模の見積もりは各モジュール担当が記入する
  - Line単位で記入する
- お客様も交えてメンバー全員でレビューを行う
  - お客様の理由や思いを正確に把握するため
  - 正確に仕様化されているか確認するため
  - 変更点・影響範囲に漏れがないか確認するため
  - 仮に、変更要求仕様書完成後、変更要求があったらその都度追記し、お客様も交えてレビューを行う
- 変更要求仕様書が完成するまでは設計に手をつけない
- 各フェーズの成果物のレビュー毎にその機能が取り込まれているか照らし合わせる

8－5. 施策1　変更要求仕様書の効果

要求の仕様化技術　変更点の追跡　抜け漏れ防止

- 要求の真の理由を把握することで、適切な仕様化ができた
  - 後戻りがなくなった
- 一覧化されていることで、どの派生開発でどの変更が入ったかわかるようになった
- モジュールのトレーサビリティマトリクスで、変更に対する各モジュールへの影響がわかるようになった
  - デグレードが発生しなくなった



思わぬ副産物

メンバー全員がレビューに参加することで、**自分の作業範囲だけでなく、その周辺**も意識するようになった

変更要求が怖くなくなった

9−1．施策2　心配点シート（2008～）

未然防止

DRBFM（Design Review Based on Failure Mode）とは

- トヨタのGD³（ジー・ディー・キューブ）の一翼をなすもの
- 吉村達彦氏の造語

未然防止の手法

- **問題解決**－現実に起こってしまった問題の原因を究明し、解決すること
- **再発防止**－その問題が再度起きないように、防止すること
- **未然防止**－まだ起きていない問題の発生を予測し、それが起きないように**未然に防止**すること

9－2. 施策2　DRBFM：未然防止手法とは

変更要求仕様書と
親和性が高い

9－3. 施策2　心配点シート導入(１)

部品名/変更点	機能	変更に関わる心配点		心配点はどんな場合に生じるか		お客様への影響	心配点を除くためにどんな設計をしたか	推奨する対応(DRBFMの結果)						対応の結果 実施した活動
		変更がもたらす機能の喪失、商品性の欠如	他に心配点はないか(DRBFM)	原因・要因	他に考えるべき要因はないか(DRBFM)			DRBFMで示された設計へ反映すべき項目	担当期限	DRBFMで示された評価へ反映すべき項目	担当期限	DRBFMで示された製造へ反映すべき項目	担当期限	

- **本に書かれた通りのフォーマットを使用して導入**
  - Excelで１行につき１つの心配点を記入

- **記入やレビューに、とても時間がかかる**
- **穴を埋めて行を増やすだけの形骸化が見受けられた**

プロジェクトのサイズを意識する必要あり

表は「トヨタ式未然防止手法GD3 いかに問題を未然に防ぐか」吉村達彦著（日科技連）を参考に作成

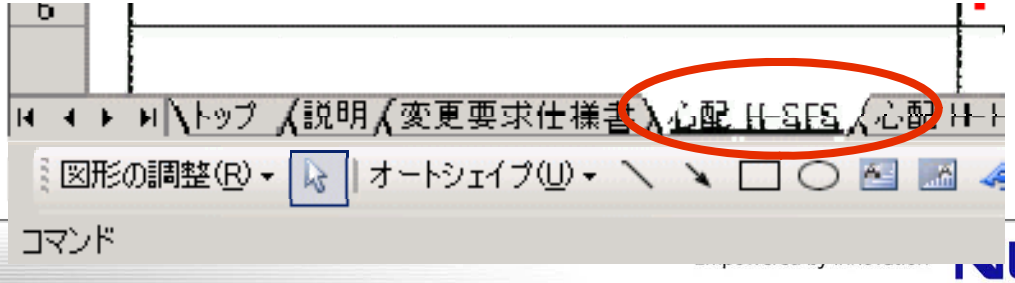
9−4. 施策2　心配点シート導入（2）

- **いろいろ試行錯誤し、DRBFMシートより簡素にし、必要十分な情報のみを記入するようにフォーマットを変更ー「心配点シート」**

- **変更要求仕様書に大要求単位でシートを追加する**
  - 「要求事項」
  - 「理由」
  - 「変更するに当たっての心配点」
  - 「心配点を取り除くためにどんな検討/設計をしたか」
- **変更要求仕様書のレビュー時、各フェーズの成果物のレビュー毎に確認する。**
  - 心配点の追加があるか
  - どんな設計対処をしたか

要求事項	理由
メールの検索機能を追加して欲しい	多くのメールの中から、求めているメールを探し出したいため

変更するに当たっての懸案、心配点	心配点を取り除くためにどんな設計をしたか
メールが多くなると、検索に時間がかかるのではないか	【変更要求レビュー】 ・検索スピードを高めるロジックを調査する ・何件のメールを基準とするか検討する ・推奨するPCのスペックを販売推進部に確認する（お客様） 【設計レビュー】 ・xxx方法にて検索するのが最適と判断 ・1万件を基準値とする ・PCの推奨スペックはxxxとする ・10秒以内に検索が完了することとする 設上記設計書および変更要求仕様書に記載した。 【テスト項目レビュー】 テスト項目に記載があることを確認した



9−5. 施策2　心配点シートの効果

未然防止

設計段階で検出できる問題が増えた

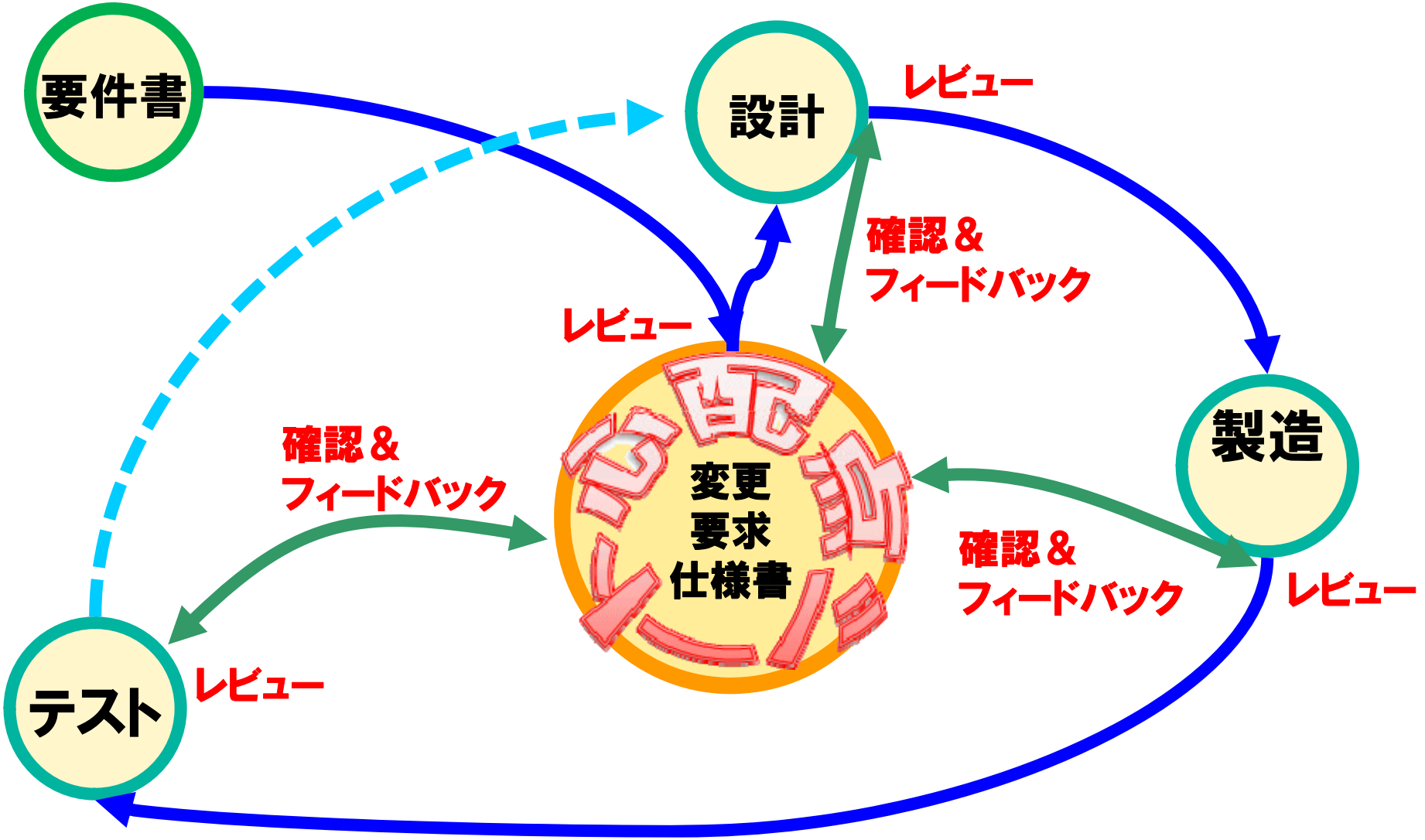
- 後戻りがさらに減った

思わぬ副産物

- 今までのレビューにプラスアルファされるだけなのでメンバーの負担が少ない
- 問題発見の**探索的な作業**なので、メンバーが楽しんでできた
- メンバーの**コミュニケーション**が活発になった
- ベテランの**経験**が若手に伝わる
- **ナレッジ**がどんどんたまっていった

未然防止に効果あり

10−1．まとめ：変更要求仕様書と心配点シートの位置づけ



10−2. まとめ：問題は解決したのか

No.	課題	テーマ	参考にした手法	呼び名	成果
1	どうやったら要求を正しく仕様化できるのか？	要求の仕様化技術	・USDM ・トレーサビリティマトリクス	変更要求仕様書	
2	どうやったら変更点を追跡できるのか？	変更点の追跡			
3	どうやったら変更の影響範囲を特定できるのか？	抜け漏れ防止			
4	どうやったら問題発生を未然に防止できるのか？	未然防止	・DRBFM	心配点シート	

解決できた！

11. 今後の課題

メンバー全員レビューが不可能な場合の方法を考える

- メンバーが多い場合
- 拠点分散開発の場合
- 規模が大きくて機能が詳細に分割されており担当者が専任になっているような場合

使うにはプロジェクトの
サイズの考慮が必要

12. 最後に

- システムやツールや手法はたくさんありますが、ソフトウェア開発は最後は「人」が主役だと思います。
- 開発完了時に完成した最終版の「変更要求仕様書」と「心配点シート」は、メンバーの叡智の固まりであり誇りです。
- そこには、気づき、コミュニケーション、チームワーク、経験、助け合い、なじらね など、人にしかできことがつまっています。
- 品質の向上は最終的にはそこにあると思います。

おまけ

どちらも2011年1月27日

おまけ 1

おまけ 2

おまけ 3

おまけ ４

おまけ ５

おまけ ６

状況に合わせたツール

おまけ

- 品質向上は雪かきに似ている
- ほっておくと家がつぶれる
- 玄関からも出られなくなる
- やっても、やっても切りがない気がする

- でも、春は必ずやってくる。
- 明るい、生き生きとした春を迎えたい。

参考文献

- 「入門十実践要求を仕様化する技術・表現する技術ー仕様が書けていますか？」
  - 清水吉男著（技術評論社）
- 「要求の仕様化入門」(Software People Vol.4)
  - (株) システムクリエイツ 清水吉男著
- 「コンサルタントが教える戦慄の仕事術 失敗しない派生開発」(Software People Vol.8)
  - (株) システムクリエイツ 清水吉男著
- 「トヨタ式未然防止手法GD^3 いかに問題を未然に防ぐか」
  - 吉村達彦著（日科技連）

人と地球にやさしい情報社会を
イノベーションで実現する
グローバルリーディングカンパニー

NECグループビジョン2017

Empowered by Innovation