AVASYS

SPI Japan 2015

設計プロセスの課題を ODC分析で改善してみた

小島　義也

エプソンアヴァシス株式会社

e-mail:yoshiya.kojima@avasys.jp

EPSON AVASYS CORPORATION

目次

背景

【現況】
今回対象とした開発プロジェクトは、組込みの派生開発をここ数年に渡って行っており、リリースを数ヶ月 単位で繰り返している。

【課題】
納期に間に合わず、再リリース日を設定することが多いことにある。
当然、顧客との信頼関係や当社の利益にも大きな影響を与えてしまう。

【方針】
この問題に対し、組織としてはスケジュール厳守と利益確保、つまりはプロジェクトの正常化を求めている。

対策

【目標】
この要求を実現するために、現場ではいくつかの開発プロセスの改善を計画し、そのうちのひとつとして、上流工程での品質の作り込みの実現を目標に掲げた。

【実施】
この目標に対し、品質管理部ではODCを用いて設計欠陥の真の原因を突きとめ、リリースを遅らせている品質問題を解決するため、改善活動を開始した。

改善前の状態

① 開発者は欠陥修正に追われてしまう
② 同様な欠陥をまた作りこんでしまう
③ テスト期間が長くなってしまう

疲弊する開発が続くだけ・・・

テスト期間が長くなってしまう要因は？

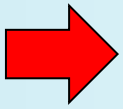
① 欠陥検出がテスト後半になっても多く検出され、欠陥修正に計画以上の工数を取られてしまう。

② 修正期日に追われ、欠陥修正に漏れが出てしまう。

③ 障害の影響に対して品質保証をコミットメントするには、更なるテスト期間の延長が必要とされてしまう。

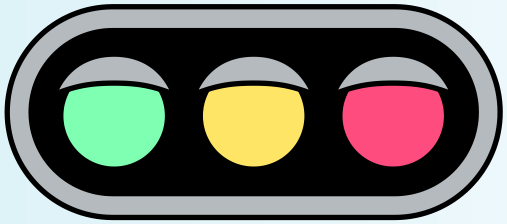
欠陥の要因を突き止めるためには？

- テスト工程での欠陥を少なくするためにはどうすればよいか？
  - 上流工程での欠陥混入を防止する対策を立てることが抜本的である。
  - レビューを厚くする対策も講じたが、効果がいまひとつ結果として結びつかなかった。
- 設計欠陥の真の原因を突き止め、しかるべき対処を取れば、テスト工程の欠陥数が減るはずだと判断した。



ODCによる分析作業を始めた

ODC（直交欠陥分類）分析とは？



統計的手法

ODC

原因解析

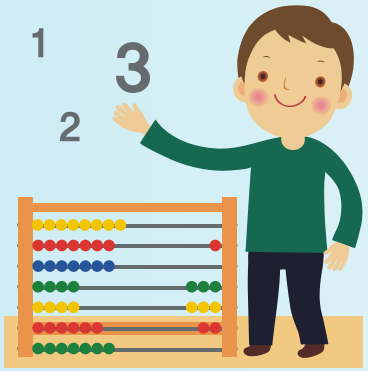
対策に直結しない

多くの労力が必要

効率よく原因・傾向が分析できる

ODC（直交欠陥分類）分析とは？

統計的手法

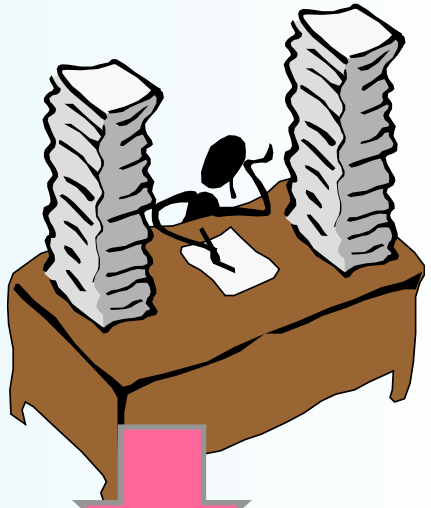


全体量を眺める

ODC

タグだけを見て、定量・定性的に分析する

原因解析



すべてを調べる

ODC（直交欠陥分類）分析とは？

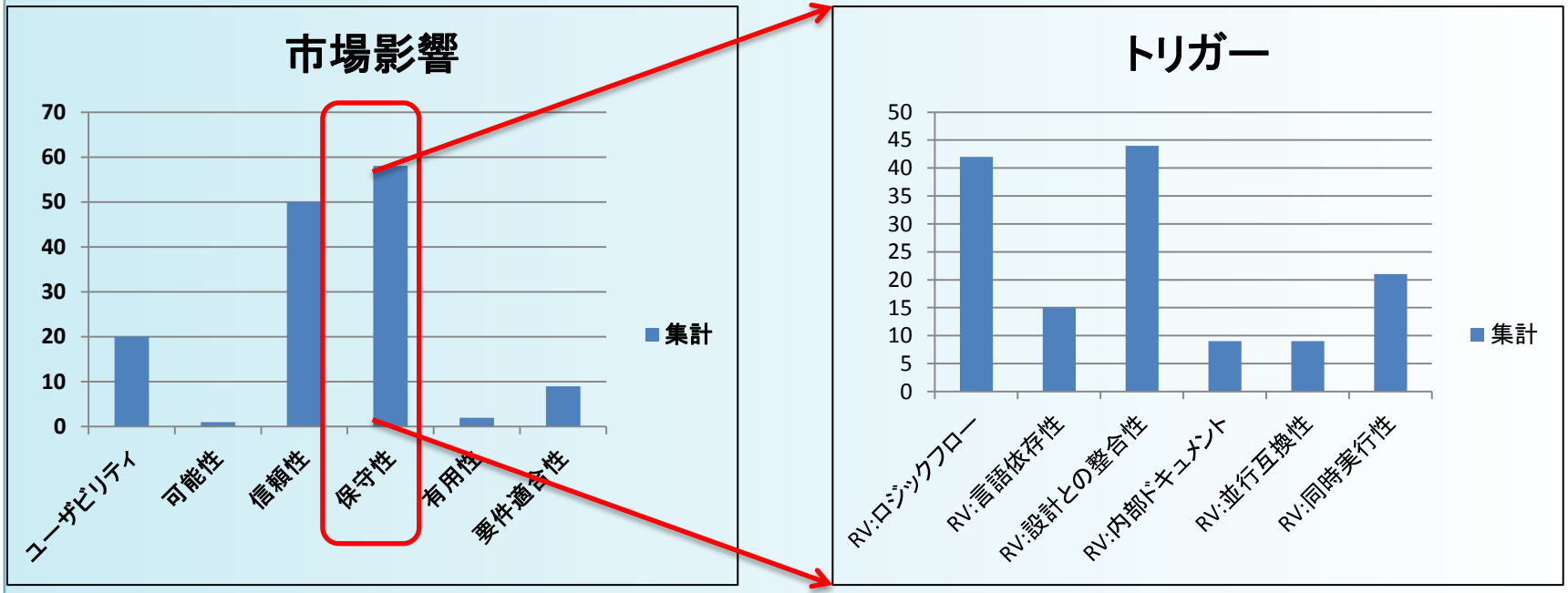
- 米国IBM研究所で開発。現在V5.2が最新。
- ODC 分析(Orthogonal Defect Classification：直交欠陥分析)とは、欠陥を多角的な属性（意味）に従って分類し、分析すること。
- 8つの属性があり、それぞれが重ならないように定義されている。
- 不具合数や信頼度曲線などでは露見しない問題が発見でき、多角的な評価が可能。

ODC分析の8つの属性

属性	概要
検出工程	欠陥を発見したフェーズ
トリガー	欠陥を発見する観点
市場影響	お客様に与える影響
修正対象	修正対象の成果物
欠陥タイプ	修正した欠陥の種類
起因	欠陥実装した種類
世代	いつ作りこまれたか
発生源	成果物の出所

交差する分析－Deep Dive による分析手法とは？

- ODCの属性間の関係性を探索していくことで、設計や実装の問題点を洗い出す手法。
- ODCの定量、定性の両方の特性を効果的に使う分析手法。



▲顧客からみた保守性の欠陥は何に起因しているのか？

DDT（Deep Dive & Tactics）

① バグ票へODC属性を追加設定
② ODC属性の接点分析
③ 欠陥の検証【Deep Dive】
④ 真の原因を追求
⑤ 改善策の立案と実施

（以下事例ベースですすめます）

①バグ票へODC属性を追加設定

グ管理一覧

結合テスト] テスト期 2014/7/17 ～]
]

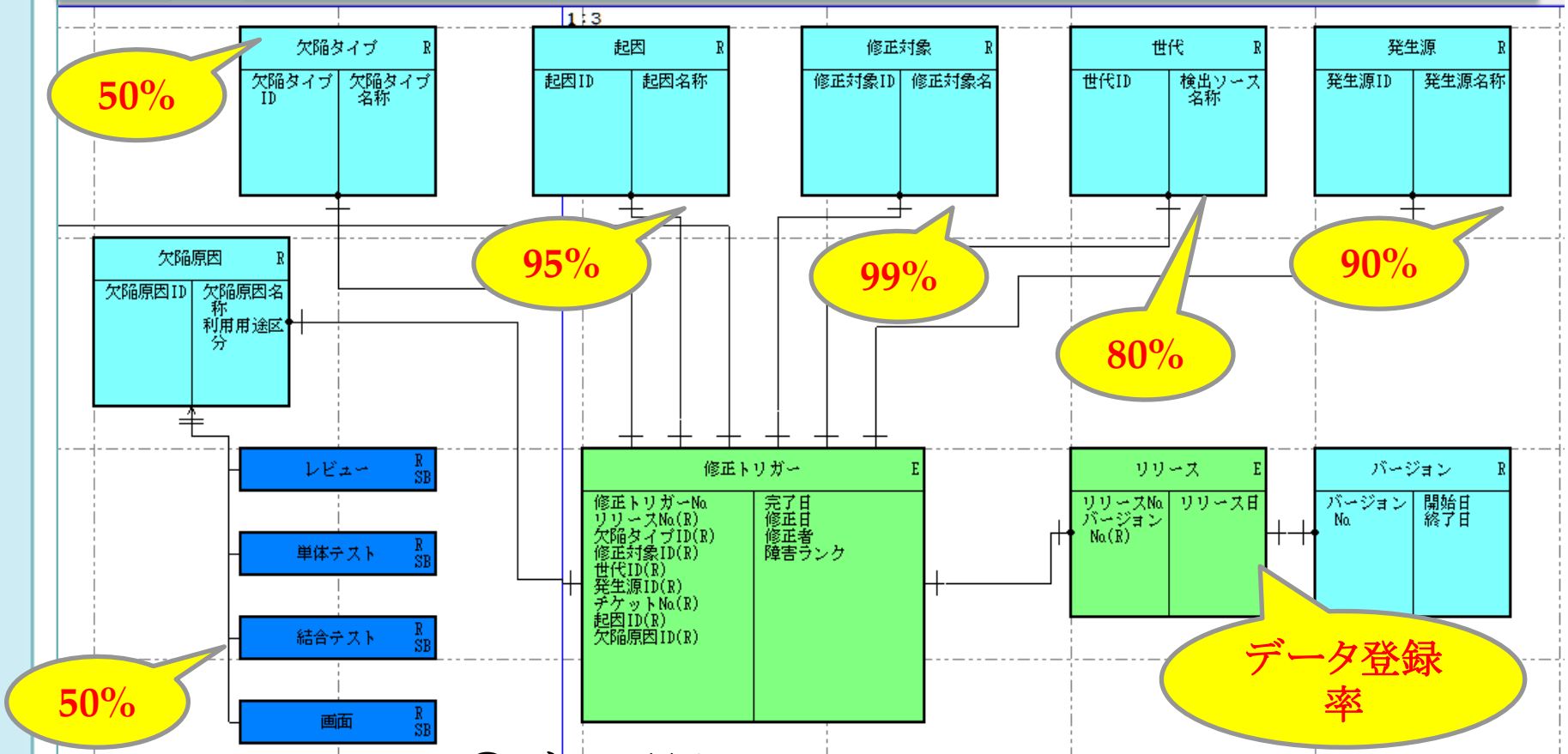
障害件数	0	件
未完了数	0	件

							ODC分析用追加入力項目													
			*	*		*	*	**	**	*	**	*	*	*		**	*			
発生日	発見者	緊急度	機能名	障害概要	調査者	障害原因	検出工程	発生要因	市場影響	修正対象	欠陥タイプ	起因	世代	発生源	処理者	対策区分	処置完了日	検証者	完了日	備　考
2014/7/17	発見者	通常	画面1	障害件名	調査担当	プログラムミス	要件定義レビュー	RV:設計との整合性	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	プログラム修正	2008/10/2	検証者	2008/10/2	備考
2014/7/17	発見者	通常	画面2	障害件名	調査担当	デグレード	設計レビュー	RV:ロジックフロー	完全性/セキュリティ	コード	検証	誤り	新規	再利用	処置担当	プログラム修正	2008/10/3	検証者	2008/10/3	備考
2014/7/17	発見者	通常	画面3	障害件名	調査担当	仕様変更	製造	RV:並行互換性	パフォーマンス	ビルド/パッケージ	アルゴリズム/メソッド	無関係	新規	アウトソース	処置担当	プログラム修正	2008/10/4	検証者	2008/10/4	備考
2014/7/17	発見者	通常	画面4	障害件名	調査担当	設計バグ	コードレビュー	RV:並行互換性	有用性	コード	関数/クラス/オブジェクト	誤り	本体	内製	処置担当	プログラム修正	2008/10/2	検証者	2008/10/2	備考
2014/7/17	発見者	通常	画面5	障害件名	調査担当	仕様通り	機能テスト	RV:設計との整合性	保守性	設計書	関数/クラス/オブジェクト	漏れ	本体	内製	処置担当	プログラム修正	2008/10/5	検証者	2008/10/5	備考
2014/7/17	発見者	通常	画面名6	障害件名	調査担当	パッケージバグ	評価	RV:内部ドキュメント	設置性	設計書	タイミング/直列化	漏れ	本体	内製	処置担当	プログラム修正	2008/10/6	検証者	2008/10/6	備考
2014/7/17	発見者	通常	画面名1	障害件名	調査担当	パッケージバグ	顧客テスト	RV:副作用	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	パッチ適用	2008/10/7	検証者	2008/10/7	備考
2014/7/17	発見者	通常	帳票3	障害件名	調査担当	ユーザ起因	設計レビュー	RV:言語依存性	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	その他	2008/10/8	検証者	2008/10/8	備考
2014/7/17	発見者	通常	帳票2	障害件名	調査担当	プログラムミス	設計レビュー	RV:設計との整合性	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	その他	2008/10/9	検証者	2008/10/9	備考
2014/7/17	発見者	通常	帳票6	障害件名	調査担当	H/W障害	設計レビュー	RV:設計との整合性	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	その他	#########	検証者	#########	備考
2014/7/17	発見者	通常	帳票1	障害件名	調査担当	[illegible]	設計レビュー	RV:設計との整合性	設置性	要件定義書	代入/初期化	漏れ	本体	内製	処置担当	その他	#########	検証者	#########	備考

※ODC分析を行うためには、バグ個票に、上記の「ODC分析用追加入力項目」が必要となります。
※検出工程、検出トリガ、発生源はプロジェクトに依りますが、固定化することが多く、入力項目から省いても問題ない

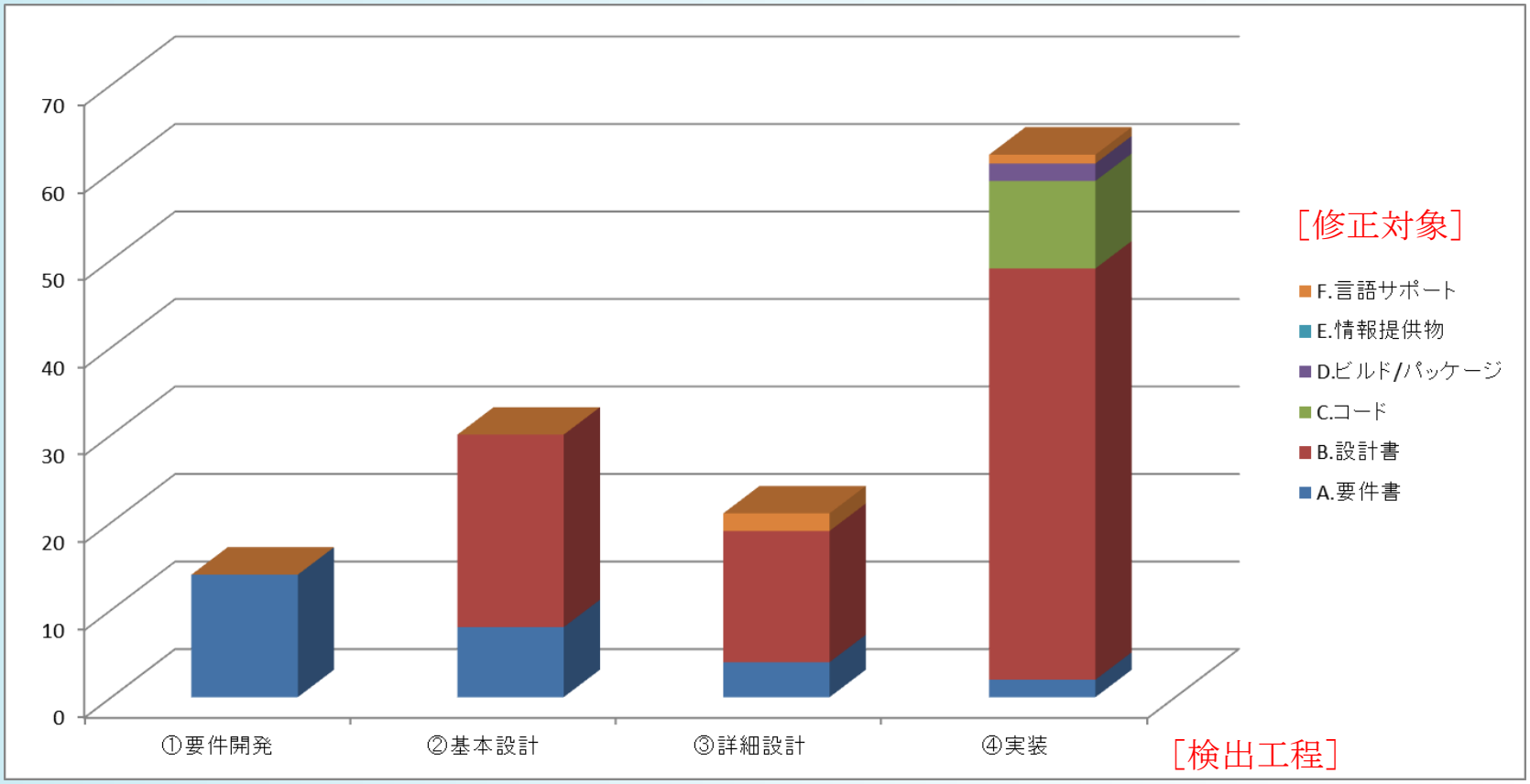
【バグ票の整理】
①ODC分析に適したバグ票の設計を実施
②設計と評価者が分類設定できる属性項目を検討追加
↓
※人によって異なってくるデータの分類作業の誤差をなるべく小さくする工夫を考える

②ODC属性の接点分析



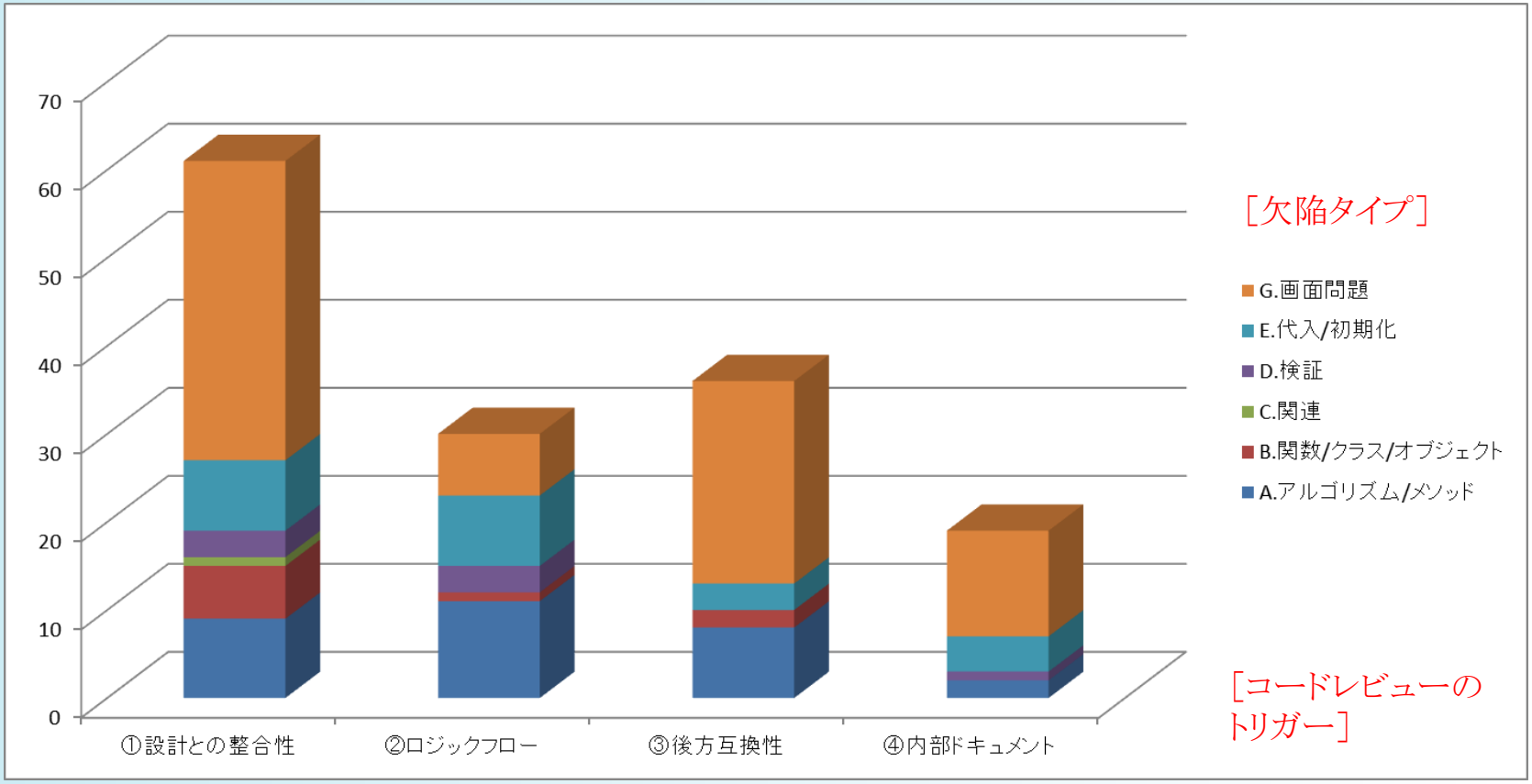
①データ量を見る
②データ量の多い関係を優先して分析をする
③データ量を分析の判断基準に加える

③欠陥の検証 【工程×修正対象】



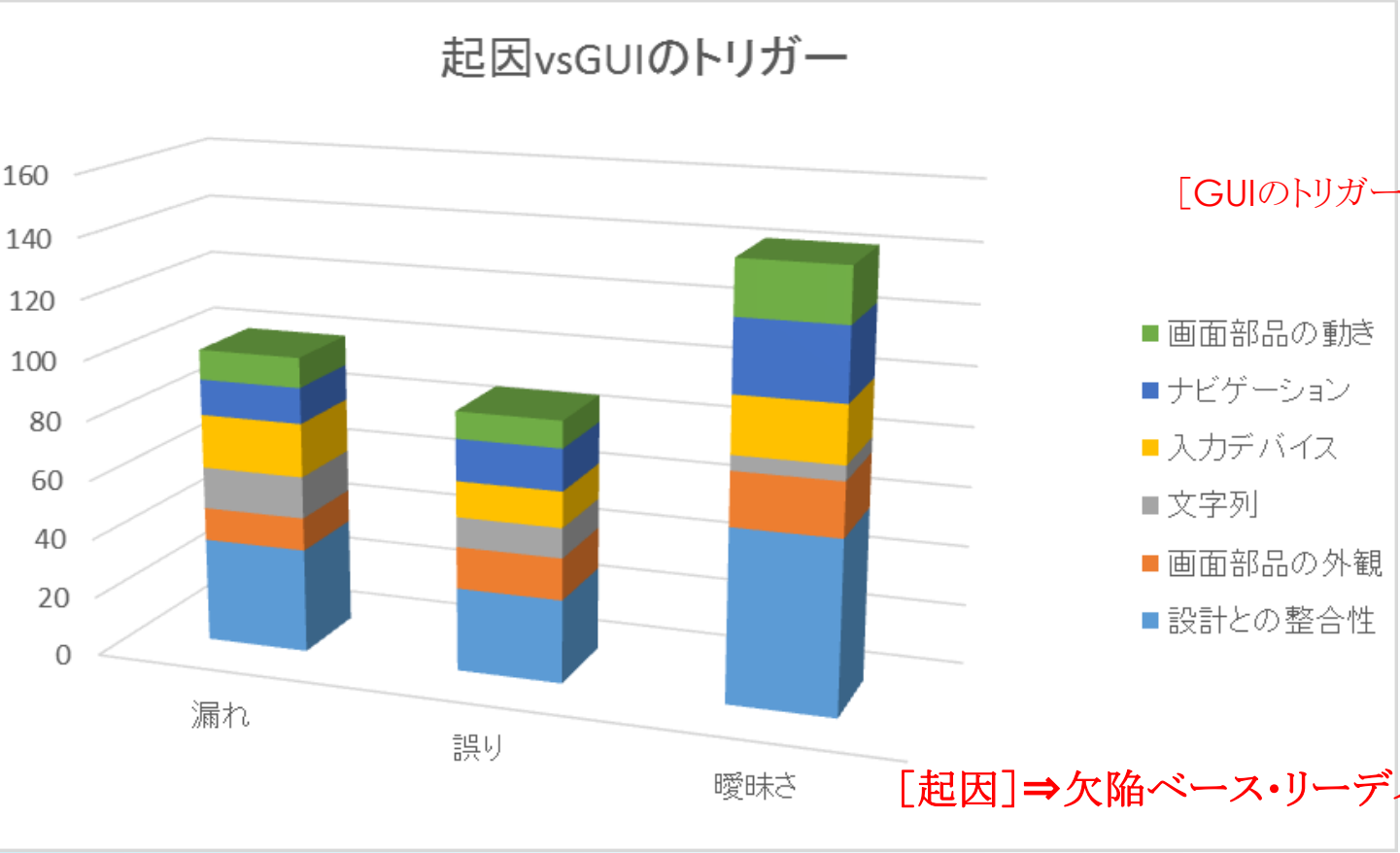
実装ミスは少なく、ほとんどが設計型欠陥であることがわかる。

③欠陥の検証 【欠陥タイプ×コードレビューのトリガー】



欠陥要因の多くが、画面問題であることがわかる。

③欠陥の検証 【起因×GUIのトリガー】



画面問題の多くが、解釈する人によって解釈の仕方が異なるような問題であることがわかる。

④真の原因を突きとめる

- DeepDiveの分析結果から、プロジェクトの弱点や品質の作りこみ状況を確認できた。
- この分析結果を用いて、設計側と品質管理側で内容の深堀を実施した。
- そこから間違った設計判断をしたり、ある制約条件を忘れていたり、要求を誤解していたり、といった原因を挙げていった。
- このなかで真の原因として多く結びついたのが、設計表現の不備であると判断した。

⑤改善策をたてる

- 設計者が要件解釈のうえで妥協したり、充分な情報なしに設計判断したりすると、すべてのテスト工程にはより多くの欠陥が混入し、残存してしまうことがわかった。
- 設計表現は、論理的な文章を書く、図的表現を使用する、擬似コードを記載するなどといった理解しやすく、勘違いの幅を狭くできる表現を磨いていくことが必要であり、そう簡単に改善できるものではない、ということがわかった。
- そこで長期的な改善活動を計画し、設計改善の指導を開発部門でリードしていくことを決めていった。

設計表現の曖昧さを改善する

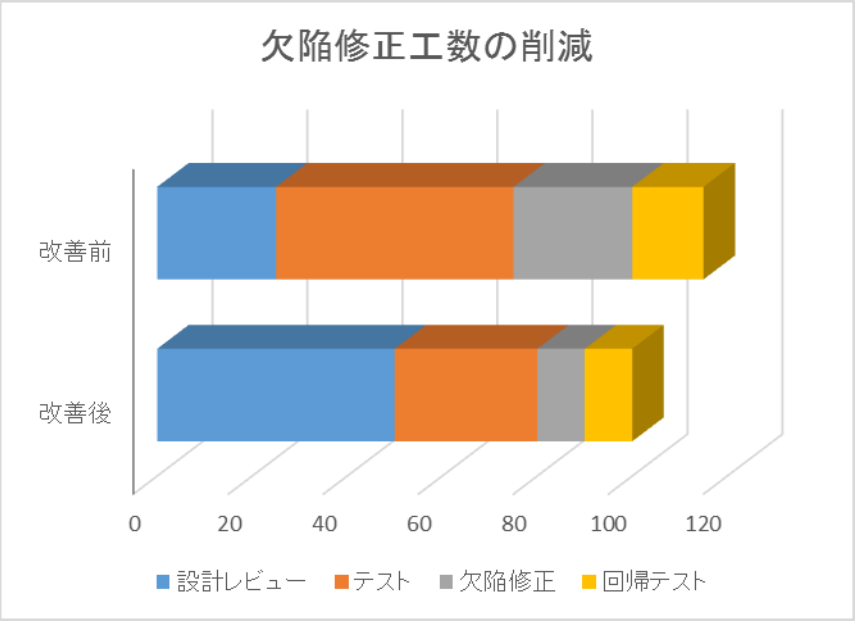
	問題点	改善策
1	状態遷移図から処理のイメージが優先され、状態の粒度や階層にばらつきがでてしまい、設計が曖昧になっている。	状態遷移図を使用する場合は、必ず状態遷移表も作成する。
2	追加要件が多い場合、多数の状態のなかに多数の遷移が交錯し、設計の理解が難しくなる。	状態遷移設計を適度なタイミングで見直す。重複する状態や不要な状態になっていないか検証する。
3	優先度逆転などの予期しなかった問題が起きてしまう。	割込みレベルの増加や変更は、PMが管理し、アドホックな変更は許さない。
4	将来的に機能追加など発生しそうにない箇所が汎用的な設計になっていて、機能効率が悪い。	必要以上の汎用性、拡張性を持たせた設計をやめる。必要であれば見直す。

対策の実施と効果確認

- 画面の問題に起因する変化、相違を曖昧なまま残さず、明確に表現できるよう努力する。
- 画面の問題を誘発する、漏れや曖昧さを残さないように、状態遷移図や状態遷移表を必ず用いて説明しているかをチェックし、レビュー時に確認指示するやり方を実施するようになった。
- 当初の目的である、リリース日の変更が必要なプロジェクトはまだ無くなったわけではないが、改善傾向に向かっていることがプロジェクトサマリから見ることができるようになってきた。

欠陥修正工数の削減

- 欠陥除去コストは、欠陥修正（ドキュメント修正＋モジュール改修）＋回帰テストの工数を計測した。



- テストでは重度の欠陥検出は少なくなり、欠陥数そのものが減少したことにより、欠陥の平均除去コストが改善されているプロジェクトも出てきた。
- 加えて評価者からの修正依頼・返却期間が短縮され、テスト期間の正常化につながったと思われるケースも見られている。

ODCおよびのDBRについては、以下のWEBを参照ください

- 「ODC(直交欠陥分類)概説」森 龍二　氏　(エクサ)
- <http://www.jasst.jp/symposium/jasst15tokyo/pdf/A5-1.pdf>
- 「ODC分析による欠陥除去と品質の成熟度可視化～医療機器の安全性・品質を担保するために～」山崎 隆 氏　(オリンパスソフトウェアテクノロジー)
- <http://www.jasa.or.jp/etwest/2014/conf/conf_ipa4.html>
- 「漏れ、誤り、曖昧さを制約とする defect-based reading」森崎 修司 氏(名古屋大学)
- <http://blogs.itmedia.co.jp/morisaki/2011/09/post-638d.html>
- 「評価者によるODCを使用した不具合分析の現場展開～属人化を排除していく試み」小島 義也　(アヴァシス)
- <http://www.jasst.jp/symposium/jasst15tokyo/pdf/A5-2.pdf>