システム開発･保守プロジェクトのQCDに目標値を持つ意味

システム開発･保守プロジェクトのQCD向上に向けて

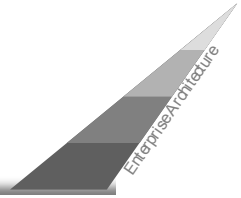
～プロジェクト担当者は、この方法で自社データを比較してほしいコース～

Enterprise Architecture

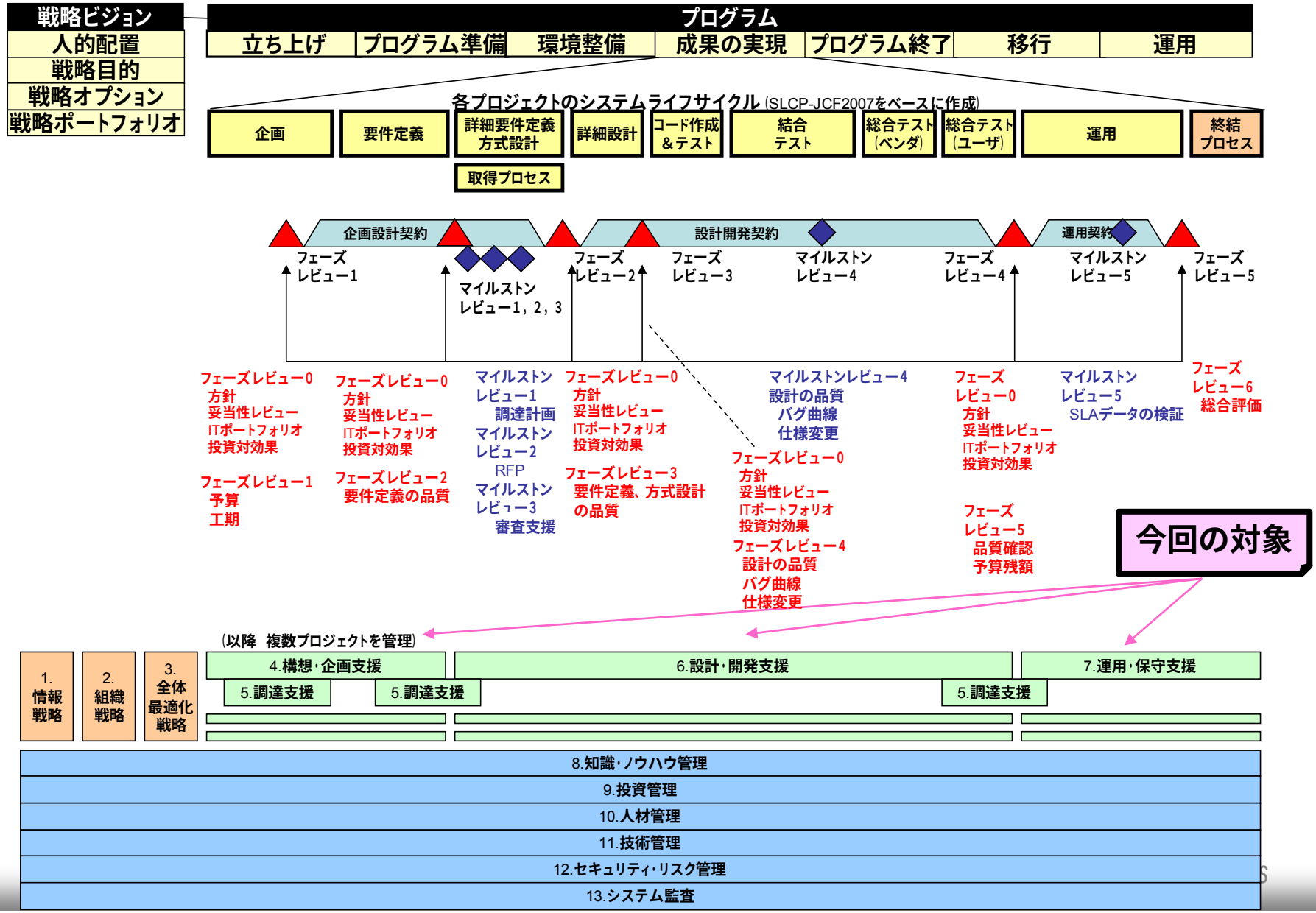
2008.7.22　13:00-17:00

株式会社フューリッジ
平本健二
hiramoto@fuledge.co.jp

はじめに



ITガバナンス全体での今回の位置付け



QCDを向上するためのベンダ選定

- システム開発･保守の近代化は進んでおり、客観的なデータに基づくベンダ選定ができるようになってきている。

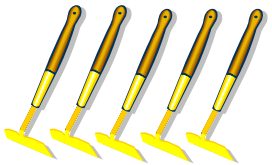
ブランドによる信頼の担保 → 認証による信頼の担保 → 実績による信頼の担保とモニター

ISO9000は品質を担保するのか？

大手は認証を取得

データはうそをつかない

本当に使いこなしているか見極めが必要



多くのブランドはOEM供給

数回使うとメッキがはがれ質実剛健な下請け企業が登場

ブランドベンダは安心か

- 安心料として単金の高いベンダもしくは要員に発注することも多いが、人月単価と品質の関係には相関が見られない。
  - ただし難易度による評価が入っていないので留意が必要である

	Aランク	Bランク	Cランク	Dランク	Eランク	Fランク	総計
欠陥率	0	0.25未満	0.5未満	1未満	3未満	3以上	
件数	8	76	38	18	16	6	162
単価（平均）[万円]	90.2	106.2	104.3	100.6	116.7	107.8	
単価（最大）[万円]	117.5	272.7	236.9	162.8	285.7	250.0	
単価（最小）[万円]	71.5	46.2	43.2	41.4	70.8	45.7	

簡単なwebシステムかもしれない　　高いのは難易度が高かったかもしれない

- ただし、カルチャーとしてしっかりとしたものを持っているし、もしかしたら他部門が手伝ってくれるかもしれない。

平均より上であるが、最高が保証されるわけではない。
保険のようなものである。

「ソフトウェアメトリックス２００７」JUAS

認証は信用できるのか

- ISO9000を取得している企業のプロダクト品質は高いのか？
- CMMを持っている企業の品質は高いのか？

- No！！

- 見せ掛けだけの認証取得をしている企業が多い
  - ISO14000を持っているのに真夏にスーツで名刺交換してくる企業とか
  - つまりは、理念が浸透していない

- CMMが出てきたときのベンダの反応
  - ISO9000を持っているのになんで必要なんだ？
    - CMMはマイルストンではなくプロセスで品質を確保することを理解していない

- CMM認証企業への質問
  - あなたは何にかかわりましたか？
  - CMM取得部門からの支援とは具体的になんですか？
- ISO9000取得企業への質問
  - ドキュメント以外での品質管理は何をしていますか

参考：

- 日本科学技術連盟　第20年度（2004年度）第1分科会Aグループ報告書によると、ISO9000の認証取得している企業でも、品質データによる管理はされていないことがわかる。CMM適用企業は、明確に未適用企業に対して実施率が高い。
  - ただし、コストや開発後期などを踏まえ、品質データの取得を取捨選択することも多く、一概に実施率が高いからよいというわけではない点に留意が必要である

品質データ取得の実施率

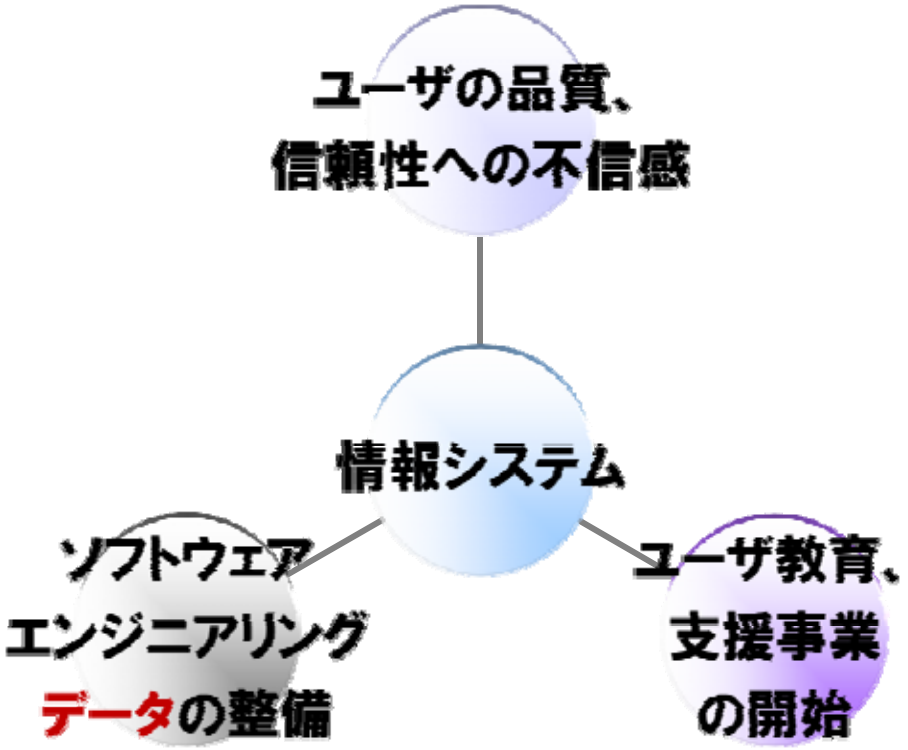
	ISO9000認証取得		CMM/CMMIの適用	
	認証取得	未認証	適用	未適用
計画・要求	16.6%	21.1%	26.6%	13.8%
設計	18.9%	22.0%	31.4%	14.2%
製造	29.6%	31.2%	40.6%	25.2%
検査	24.0%	33.1%	30.6%	24.0%

こっちのほうがしっかりしている。

ここと較べても変わらない。

システム開発のエンジニアリングが急速に進展

- 情報システムの可視化への機運が高まり、最後に未整備だったデータの整備も行われたことにより、定量管理が可能になった。



ＥＡによる図面の整備
ＰＪ管理の近代化
ＩＴＳＳ、ＵＩＳＳによる人材の明確化
（ＳＬＣＰによるドキュメントの明確化）

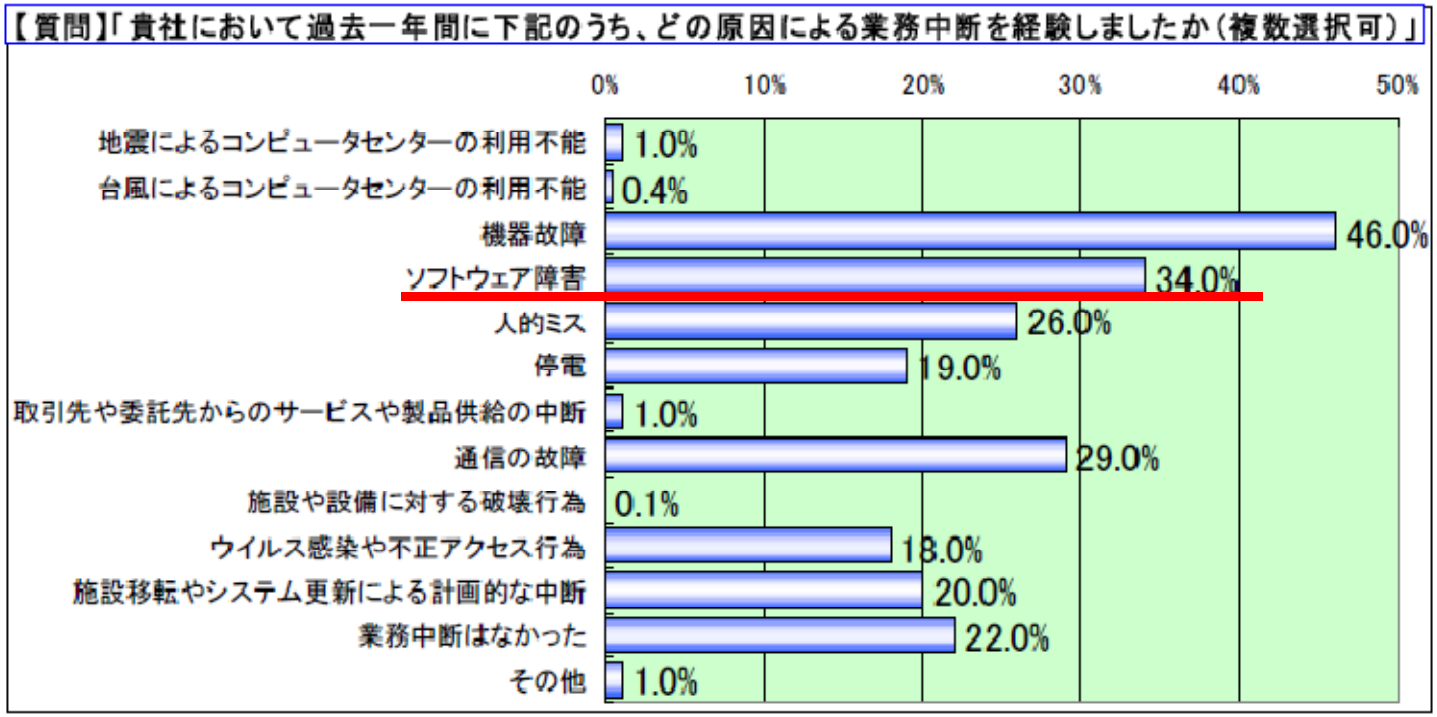
生産データがそろって初めてエンジニアリングの原点に立てた

特に進展するソフトウェアメトリックス

- ソフトウェアメトリックスデータの充実
  - ソフトウェア開発の標準的なデータの蓄積はこれまでほとんど行われてこなかった
  - 海外のデータはＣＡＳＥなどを使っていることや契約習慣も違うことから国内にはそのまま適用できなかった
  - 3年前から情報処理推進機構、日本情報システムユーザ協会が本格的な収集を開始
  - ソフトウェアメトリックスに関する書籍が徐々に増え始めている
  - 現在はベンダが読んでいるが、ユーザへの浸透を模索しているところ
- 現状のソフトウェアメトリックスデータの信頼性
  - 本格的な取り組みが始まってから3年しかたっていないため、データの信頼性には揺らぎがある。
    - 取捨選択する必要がある。
    - あくまでも参考値として使う必要がある

ところで品質や信頼性にかかわる経験は？

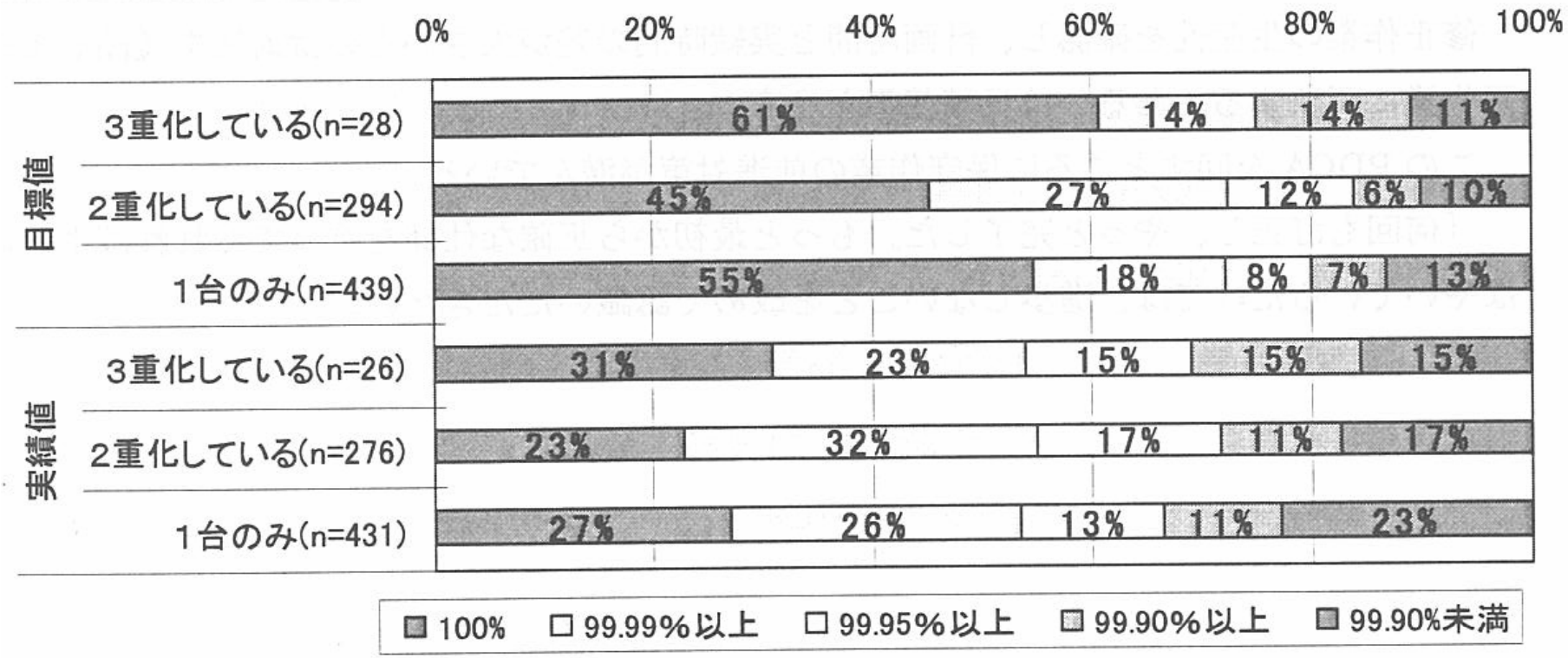
- BCPの重要性も指摘されるが、通常運用においても業務停止になることが多い。３４%の人がソフトウェア障害を経験している。業務継続性の観点からも、ソフトウェアの品質管理は避けて通れない課題である。



事業継続計画策定ガイドライン、METI

ハード的にどこまで信頼性を確保できるのか

◆ 実は目的ほどに効果を発揮していない。



IT動向調査2004,JUAS

品質や信頼性を向上させるためのポイント

- システムの品質や信頼性を向上させるには以下の取り組みが重要である。
  - さまざまな障害を想像できる力を身につける
    - これが不足していると設計書を見たときにチェックポイントがつかめない
      - 日経コンピュータの動かないコンピュータなどを見て、さまざまなケースをイメージできるようにトレーニングする
  - データに基づき沈没しつつあるプロジェクトを予見する
    - 沈没するプロジェクトはさまざまな危険信号を発している
  - 楽観的に考えない
    - プロジェクトマネージャは、特に前半において、希望的観測により問題の対策を遅らす場合が多い。ユーザ、ベンダの双方に同じ傾向があるので注意が必要である
  - 品質の悪いシステムを防ぐために
    - 入りを制す
      - 正しい予算や計画にすることでコントロールを行う
        - » 予算査定や調達などで、失敗プロジェクトの「入り」をコントロール
    - 流れを制す
      - 開発工程をモニターすること、傾向把握からコントロールを行う
        - » 走り出したプロジェクトが外れた方向に進んでいないか「流れ」をモニター
      - 内容的にはレビューで潰していく
    - 出を制す
      - 納品の検査で納品前後のコントロールを行う
        - » 運用側で受け入れられるのか「出」をコントロール

これまでの品質や生産性に関する取り組み

- これまでの取り組みはプロセスやマイルストンチェックが中心であり、定量的な指標管理ができていない。
  - QC活動
    - ツールが用意されただけで所詮は現場の改善活動
  - ISO9000
    - これをもっているから品質が高いなんて評価は聞いたことがない
  - CMM
    - 品質と生産性を確保するためにプロセス改善を行おうとしたが、十分に普及していない
      - ただし、現在でも本命の取り組みなのは間違いない。
      - 基幹システムをする企業はレベル3程度が必要
  - PMBOK
    - 準拠してプロジェクトをやっていると各社は言っているが、ではなぜ失敗するの？

品質管理を行うための前提

- **基礎データをそろえる**
  - **システム規模はわかっているか**
    - **何がしかのデータを元に、共通尺度であるFPを算出**
      - **価格→FP、画面数→FP、KLOC→FP**
  - WBS**はかけているか**
    - **進捗と品質をあわせて管理するのであれば、基礎となるWBSがしっかりかけている必要がある**
      - WBS**は網羅性があるのか？WPは10日前後か？**
  - **日常の各種数値を記録しているか**
    - **バグ数、テスト項目数・・・・**
  - **ドキュメンテーションの充実**
    - **最低整備すべきドキュメントの整備**

品質は人で担保する

- 品質管理に知見のある人をPMにすることで品質に関する意識を高めることができる。
- チームをきちんと見極める。
- メトリックスを導入しただけで、倒れたプロジェクトは数多くある。

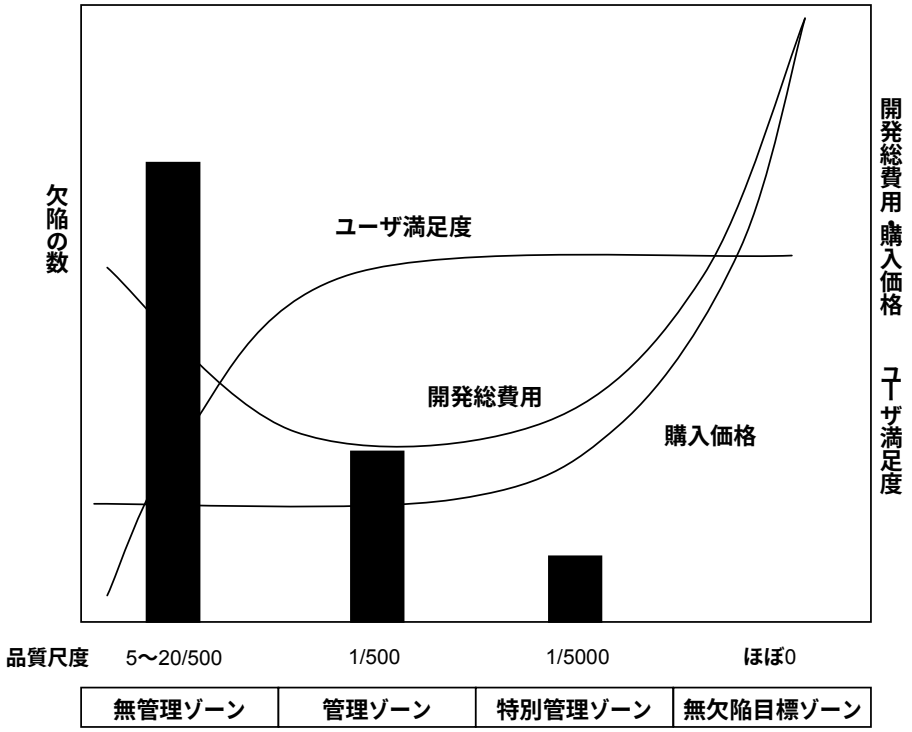
- 重要な質問
  - 品質担当者は誰ですか
    - 品質管理の実績と手法は？
  - 品質管理はどのように行いますか
  - 品質指標値はありますか

ちなみに大手ベンダでは

- 基本的にソフトウェアメトリックスデータは完備している
  - 社内管理用に使ってきており、外部の問い合わせに対しては社外秘としている会社が多い
  - 契約条項などにすることで出すことは可能
- ベンダ現場はソフトウェアエンジニアリングデータの収集を嫌っている
  - 何のためにやっているかわからない
  - そのプロジェクトだけではあまりメリットがない
  - そんなもの管理している暇がない

品質はむやみに要求すればよいと言う話ではない

- まず始めに、品質と価格のバランスを理解する必要がある。高度な品質を求めると、それに対応して指数関数的に開発費用がかかるようになる。また、逆に品質が低いと対応費用が必要となり、安く開発を行ったとしても結局は高いコストが必要になることも多い。ユーザ満足度を意識して品質レベルを決めていく必要がある。
  - セキュリティについても同様のことが言えるので、コストと信頼性のバランスを熟考の上、セキュリティレベルを考えていく必要がある。



注1 品質尺度: (納入時から安定稼働期迄の欠陥個数)／欠陥費用(万円)
注2 開発総費用と購入価格のギャップはテスト結果の確認、修正結果の確認のために要するユーザ側の付加増加費用をイメージ化したもの

Good Enough Quality
を目指す

出典:「「システム・リファレンス・マニュアル(SRM)」」JUAS 2005

参考

	レベル1	レベル2	レベル3	レベル4	レベル5
稼働率	98%以下	99%	99.9%	99.99%	99.999%以上
バックアップ機	なし	あり (部分的)	あり (2／N＋1台)	あり (Hot stand by)	あり (Hot stand by)
サービス停止時間 (　)時間／年	172時間	86時間	8.6時間	50分	5分
到着時間	1-6時間 (昼) 12時間 (夜間)	1-6時間	1-3時間 (昼) 6時間 (夜間)	常駐 ケースによっては2時間	常駐
修復時間 ・故障修復 ・再立ち上げ	6時間-12時間 10分-1時間	6時間-12時間 10分-1時間	3時間-6時間 10分-1時間	3時間-6時間 0分-10分	3時間-6時間 即時
費用 ・構築費用 ・運用費用	1.0倍 1.0倍	1.2～1.8倍 1.1～1.3倍 (マニュアル)	1.2～3倍 1.3～2.0倍	1.5～4倍 2.0～3倍 (保守も)	4～6倍 3～4倍
システム構成 (例) 必要な機能		NAS	SAN NAS クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ	SAN クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ 三重化	SAN クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ 三重化、四重化
ペナルティ			対象	対象	対象

参考：症例分類表（一般マップ）

対象（何が）		症状（どうした）	偏り					不十分				変化			
			依存	過信	集中・偏在	疲弊	超過	不足	不在	不備（不良）	無関心・無自覚	交代・交換	変更	離脱	低下
人	個人	プロジェクトマネージャ		○	○	○		○	○	○	○			○	
		キーパーソン	○	○	○	○	○	○	○	○	○			○	
		担当者	○	○	○	○		○	○	○	○	○		○	
	グループ	要員			○	○	○	○	○	○	○	○		○	○
		顧客	○					○	○	○	○	○			
		協力・関係会社	○				○	○	○	○	○	○		○	
	組織	組織						○	○	○					
もの	成果物	ドキュメント		○			○	○	○	○					
		プログラム		○	○		○	○	○	○			○		
		仕様		○				○	○	○			○		
	材料	ハードウェア	○	○				○		○					
		パッケージ	○	○				○	○	○	○				
		ツール						○	○	○					
金		予算					○	○	○	○	○				
		売り上げ								○					
		費用					○	○	○	○			○		
スキル	テクニカルスキル	技術		○				○	○	○	○				
		設計						○		○					
		テスト			○		○	○	○	○					
		見積もり	○				○	○	○	○					
		実測						○	○	○					
		知識・経験			○			○	○	○					
		レビュー・検証						○	○	○					
		調査						○	○	○					
	ヒューマンスキル	基本動作・文化						○	○	○	○				
		ルール・手順						○	○	○					
		管理					○	○	○	○	○				
		検討						○	○	○					
		コミュニケーション	○					○	○	○	○				
		理解・合意						○	○	○					
		注意・考慮						○	○	○					
		判断						○	○	○					
環境		プロジェクト内部環境						○	○	○					
		プロジェクト外部環境					○	○	○	○					
その他		契約	○	○	○	○		○	○	○		○			
		計画						○	○	○					
		スケジュール		○			○	○	○	○		○			
		範囲・役割					○	○	○	○		○			
		ブランド	○	○						○					
		教育・育成						○	○	○					

参考：SQuBOK


squbok-beta[1].pdf (保護) - Adobe Reader
ファイル(F) 編集(E) 表示(V) 文書(D) ツール(T) ウィンドウ(W) ヘルプ(H)
9 / 171
133%
検索
SQuBOK® 樹形図（全体）
Guide to the Software Quality Body of Knowledge
1. ソフトウェア品質の基本概念
1.1 品質の概念
1.1.1 品質の定義（品質の考え方の変遷）
1.1.2 ソフトウェア品質モデル
1.1.3 ディペンダビリティ
1.1.4 セキュリティ
1.1.5 ユーザビリティ
1.1.6 セーフティ
1.2 品質のマネジメント
1.2.1 品質コントロール
1.2.2 品質保証の考え方
1.2.3 改善の考え方
1.2.4 ソフトウェアの品質マネジメントの特徴
1.2.5 ソフトウェア測定の考え方
1.2.6 ソフトウェア評価の考え方
1.2.7 V&V（Verification & Validation）
1.2.8 検査
2. ソフトウェア品質マネジメント
組織レベルのソフトウェア品質マネジメント
2.1 ソフトウェア品質マネジメントシステムの構築と運用
2.1.1 品質マネジメントシステム
2.1.2 ソフトウェア品質推進活動
2.1.3 品質マネジメント組織
2.2 ライフサイクルプロセスのマネジメント
2.2.1 ライフサイクルモデル
2.2.2 プロセスモデル
2.3 プロセスアセスメント・プロセス改善のマネジメント
2.4 検査のマネジメント
2.5 監査のマネジメント
2.6 教育・育成のマネジメント
2.7 法的権利・法的責任のマネジメント
プロジェクトレベル（共通）のソフトウェア品質マネジメント
2.8 意思決定のマネジメント
2.9 調達マネジメント
2.9.1 パートナーとのコミュニケーション
2.10 構成管理
2.10.1 変更管理
2.10.2 バージョン管理
3. ソフトウェア品質技術
カテゴリ
副カテゴリ
3.1 メトリクス
知識領域
副知識領域
3.1.1 測定理論
3.1.2 プロダクトメトリクス
3.1.3 プロセスメトリクス
3.2 品質計画の技法
3.3 要求分析の技法
要求獲得(*1)
3.3.1 品質要求定義
（アーキテクチャ）設計の技法(*1)
実装の技法(*1)
3.4 レビューの技法
3.4.1 レビュー方法
3.4.2 仕様・コードに基づいた技法
利用に基づいた技法(*1)
3.5 テストの技法
3.5.1 経験および直感に基づいた技法
3.5.2 仕様に基づいた技法
3.5.3 コードに基づいた技法
3.5.4 フォールトに基づいた技法
3.5.5 利用に基づいた技法

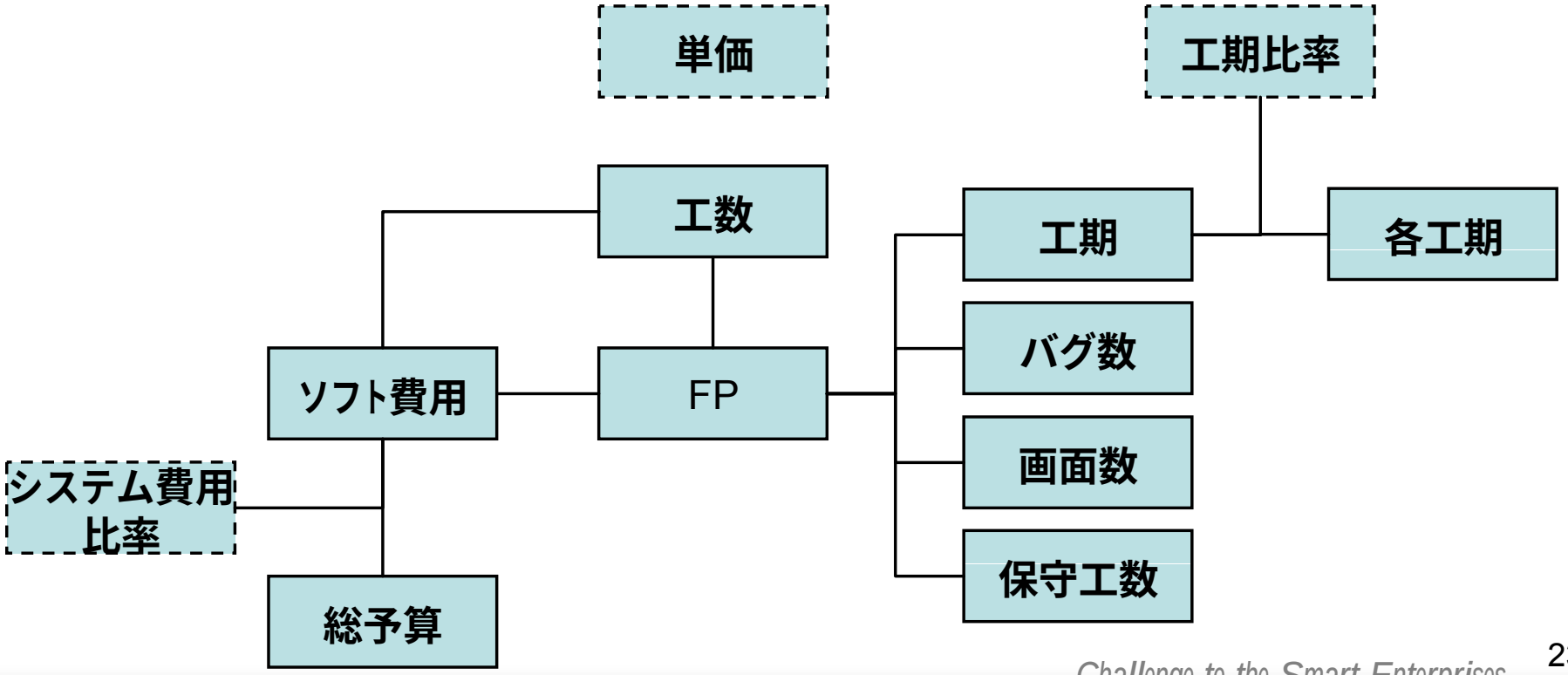
注視すべき動向

- 要求定義についても合理的な管理方法が求められており、要求定義管理ツールや形式言語などが注目されている。
  - 形式言語は理論的であり普及は難しい
  - 要求管理ツールは海外製品があるが、国内展開ではソフトではなく自動車産業など洗練された産業をターゲットにしている。

ソフトウェアメトリックス調査
分析結果紹介

各種データの関係

- ソフトウェアの開発はこれまで経験に依存するところが多かったが、ベンダ内では生産データを蓄積し、異常プロジェクトの検出に努力している。これらを共通化しようという取り組みも始まっており、情報処理推進機構ではソフトウェア開発白書を刊行している。また、情報システムユーザ協会では経済産業省と連携して、ユーザから見た生産性データの蓄積をソフトウェアメトリックス２００７として作成している。



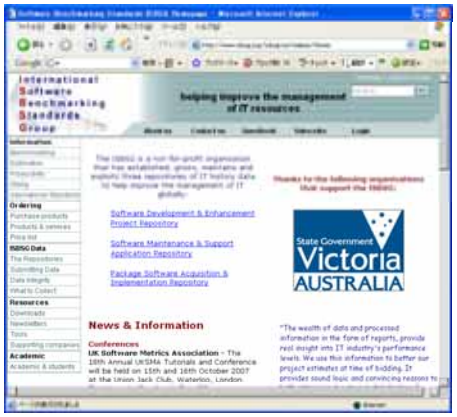
各種ソフトウェアメトリックス調査

- ソフトウェアに関しては多くのデータが公表されている。
  - ユーザ視点での評価
    - JUAS (日本情報システムユーザ協会)
  - ベンダ視点での評価
    - IPA (情報処理推進機構)
    - 日科技連
  - 国際調査
    - ISBSG (International Software Benchmarking Standards Group)

JUAS
http://www.juas.or.jp/

IPA
http://www.ipa.go.jp/

日本科学技術連盟
http://www.juse.or.jp/



ISBSG
http://www.isbsg.org/

各種学会
・情報処理学会
・ソフトウェア科学会

メトリックス調査に補正は必要か？

- ソフトウェアのメトリックスデータは統計的に処理されているので、もっと詳細なデータが知りたいことがある。
- たとえば、言語により生産性が大きく異なることが知られている。

言語	レベル	LOC／FP
アセンブラ	1.00	320.00
C	2.50	128.00
C＋＋	5.00	64.00
COBOL	3.00	106.67
FORTRAN	3.00	106.67
PowerBuilder	20.00	16.00
SQL	25.00	12.80
VB	10.00	32.00
平均	7.63	92.35

「ソフトウェア見積もりの全て」共立出版
平均は、adaなどの言語を含む15言語の平均
レベルは、アセンブラを1としたときの生産性

Source Code Language	Conversion Ratio (LOC Per Function Point)
Basic Assembly	575
JCL	400
Macro Assembly	400
C	225
Cobol74(Cobol I)	220
FORTRAN	210
Cobol85(Cobol II)	175
Pascal	160
PL/1	126
RPG I	120
RPG II/III	110
Natural	100
C++	80
Java	80
dBaseIII	60
Focus	60
Clipper	60
oracle	60
Sybase	60
dBaseIV	55
Perl	50
JavaScript	50
VBScript	50
Shell Script	50
SAS	50
APL	50

資料
DCG

KLOC/コンバージョンレシオ＝FP　でコンバージョン可能

様々な補正が可能

- アーキテクチャー別の補正値
- 新規開発、再開発別の補正値
- 言語別の品質

SLOCあたりの不具合数

言語	平均	標準偏差
COBOL	0.075	0.160
C	0.051	0.100
VB	0.103	0.206
Java	0.122	0.330

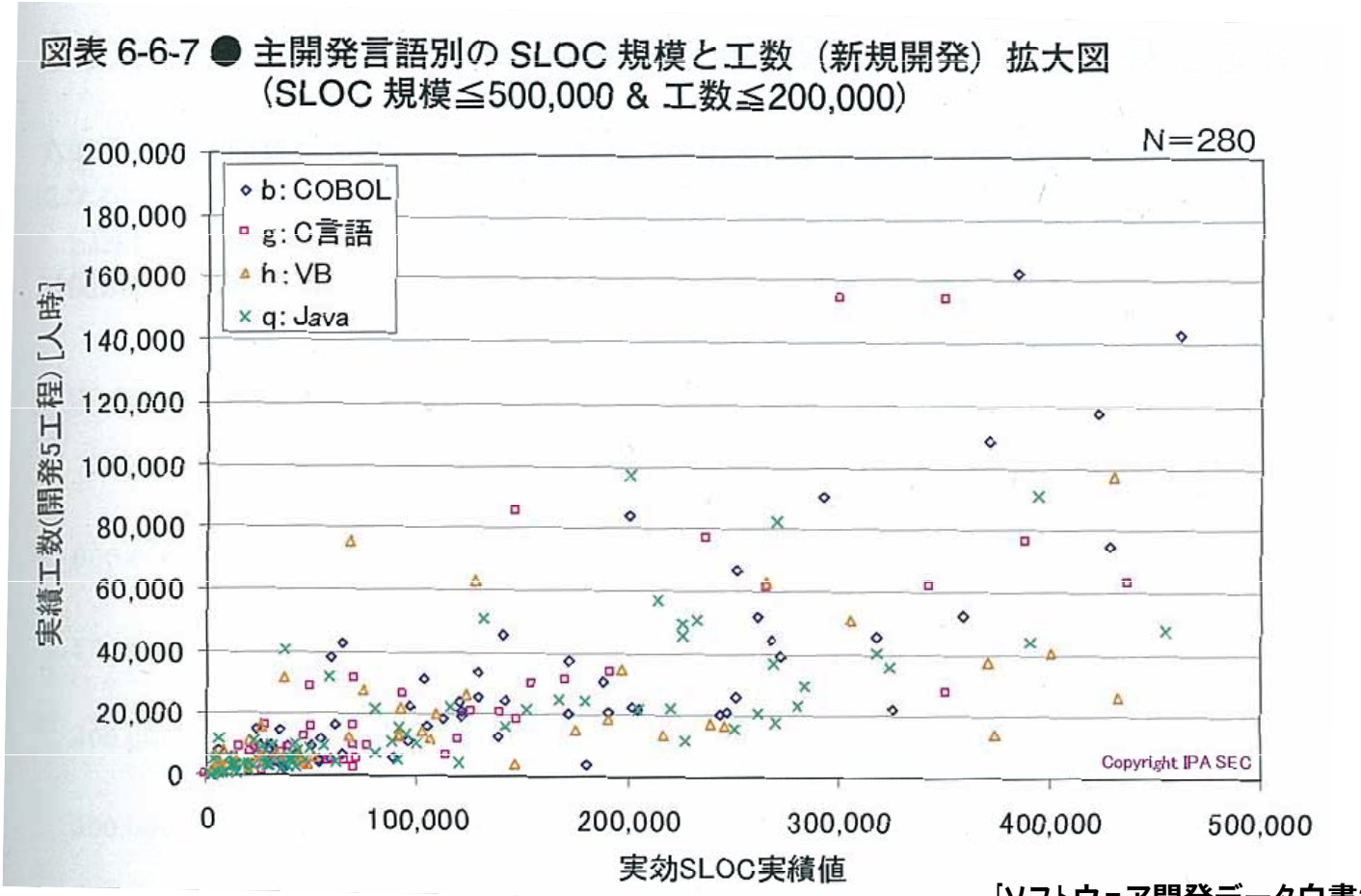
SLOCあたりの不具合数

言語	平均	標準偏差
新規開発	0.119	0.375
改修･保守	0.175	0.694
再開発	0.181	0.417
拡張	0.077	0.192

「ソフトウェア開発データ白書2007」IPA

BUT! 現段階では補正をしなくてもよい

- エンジニアリングはまだ夜が明けたばかり。



「ソフトウェア開発データ白書2007」IPA

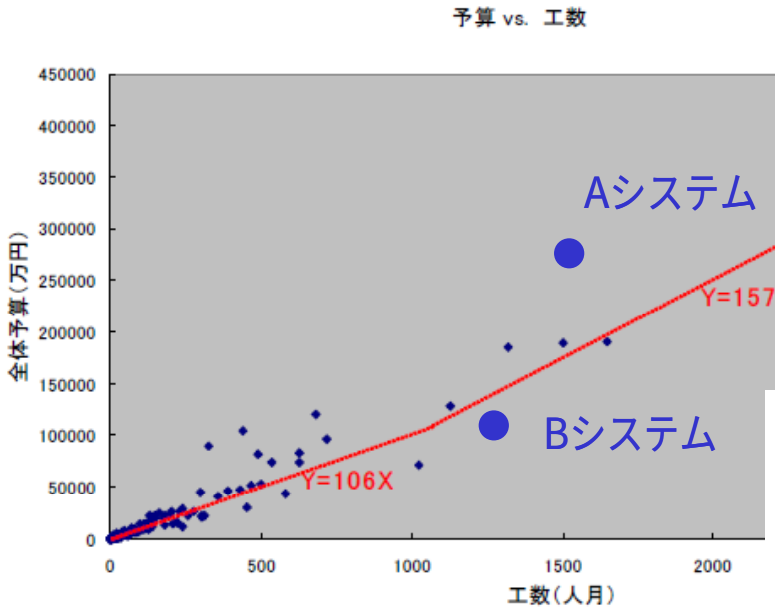
過去の自社データの活用方法

過去のデータを分析する意味

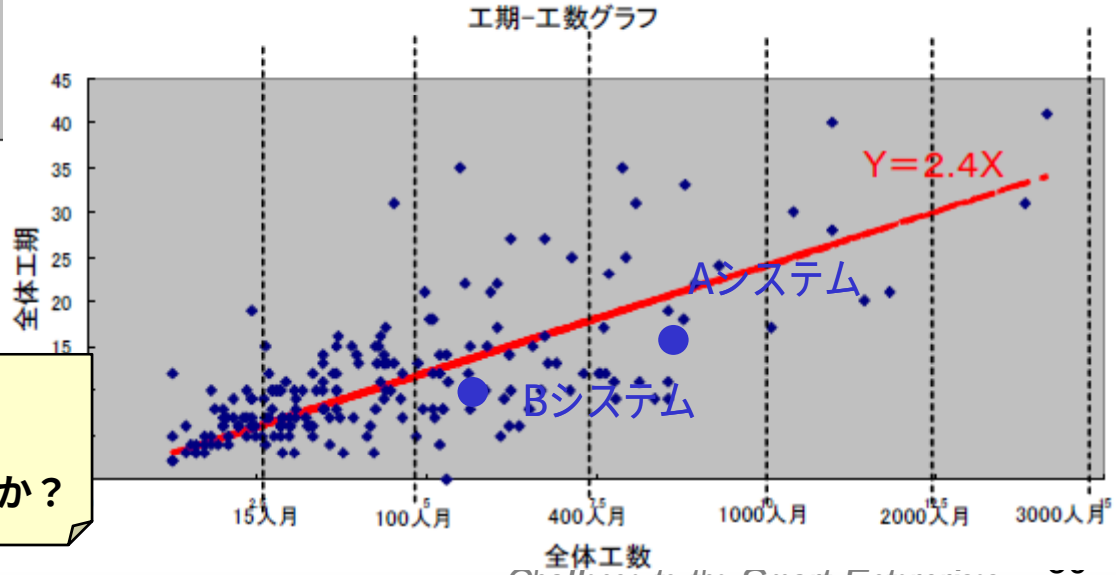
- 今後、ソフトウェアメトリックスを活用していくためには、過去のデータを分析することが重要である。
  - 自組織の弱点を把握
    - システムが適正価格で購入されてきたのか？
    - これまで失敗したプロジェクトの原因はなんだったのか
  - 自組織の傾向を把握
    - 統計とは違った自社、業界特有の傾向を知る
    - 今後の対策用に整理する
  - ソフトウェアメトリックス活用の練習
    - 開発中のシステムにいきなり導入すると、現場の理解も得られないし、実施している本人も混乱する。

過去データ分析の例

- 高い買い物か安い買い物かわかる！
- 適正工期かわかる！



契約によって契約単価がバラバラ
・難易度が違ったのか？
・既存か新規か？
・ぼったくられたか？
・買い叩いたか？
・品質との関係はどうなっているのだろうか

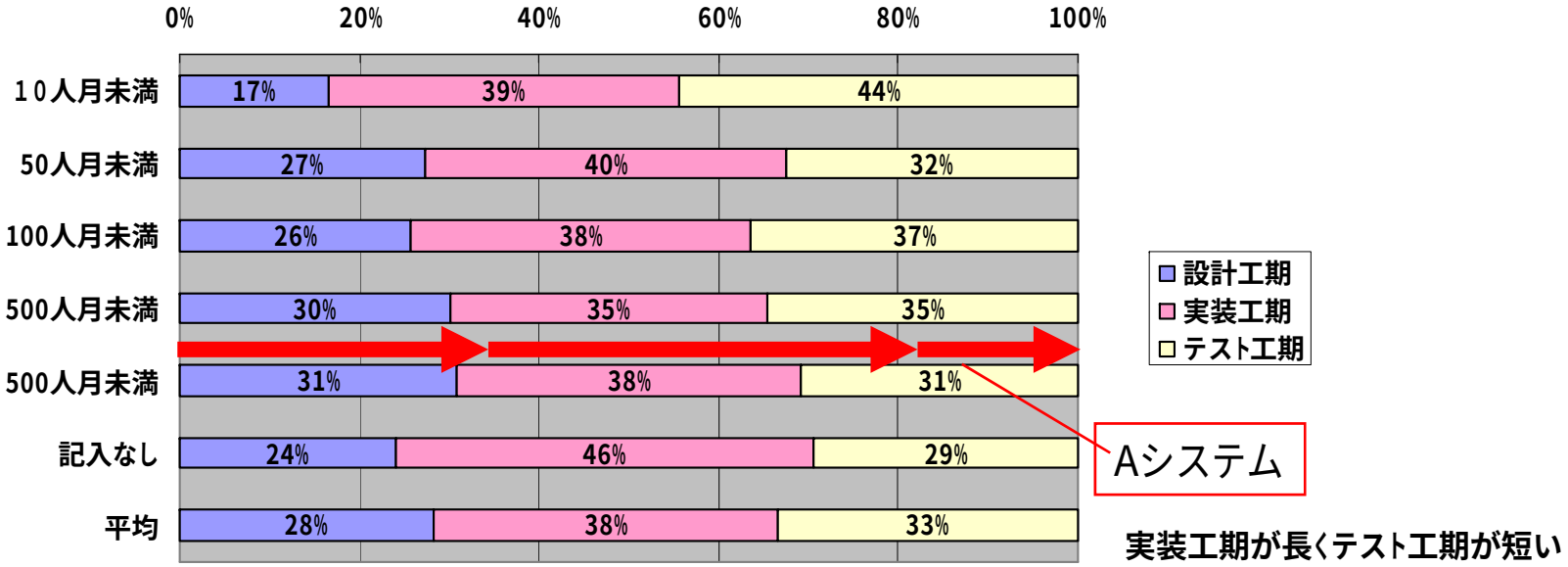


全般的に短工期開発
・品質は大丈夫か？
・人を突っ込みすぎているのではないか？

「ソフトウェアメトリックス２００７」JUAS

過去データ分析の例

◆ 工期配分が適正かわかる！



実装工期が長くテスト工期が短い

テスト工程が短すぎる
・品質は大丈夫か？
・計画時のスケジュールはどうなっていたのか？
・何か遅延の原因があったのか？

「ソフトウェアメトリックス2007」JUAS

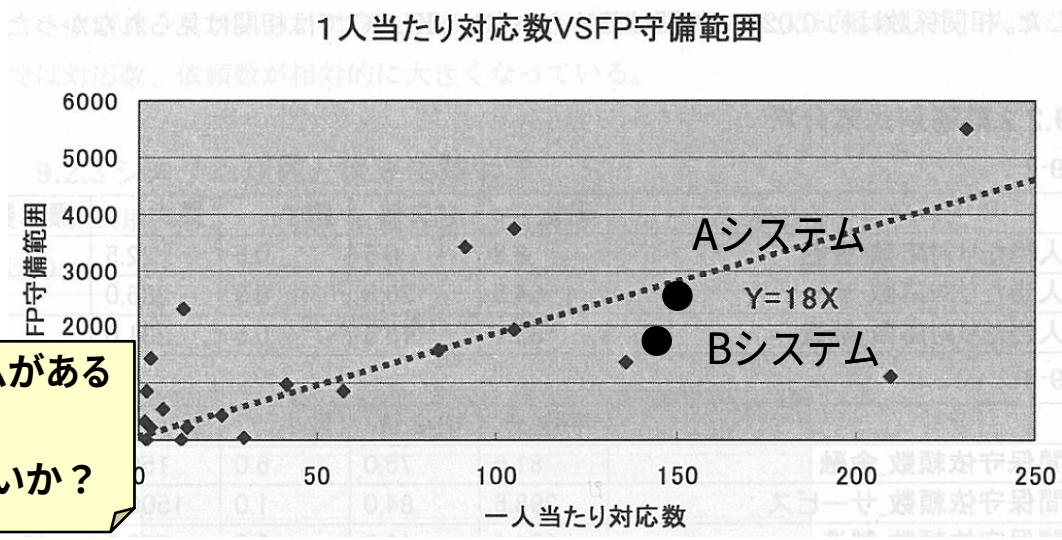
過去データ分析の例

- 適正な品質レベルかわかる！
- 保守工数が適正かわかる！

	Aランク	Bランク	Cランク	Dランク	Eランク	Fランク
欠陥率	0	0.25未満	0.5未満	1未満	3未満	3以上
割合	9.7%	33.1%	18.8%	17.5%	14.9%	5.8%
件数	15	51	29	27	23	9

Aシステム　　Bシステム

導入時の品質が悪いものがある
・目標品質は？
・難易度は？
・欠陥レベルは？



FP保守範囲が狭いのに対応数が多いシステムがある
・納品品質が低かったのではないか？
・運用マニュアルが整備されていないのではないか？

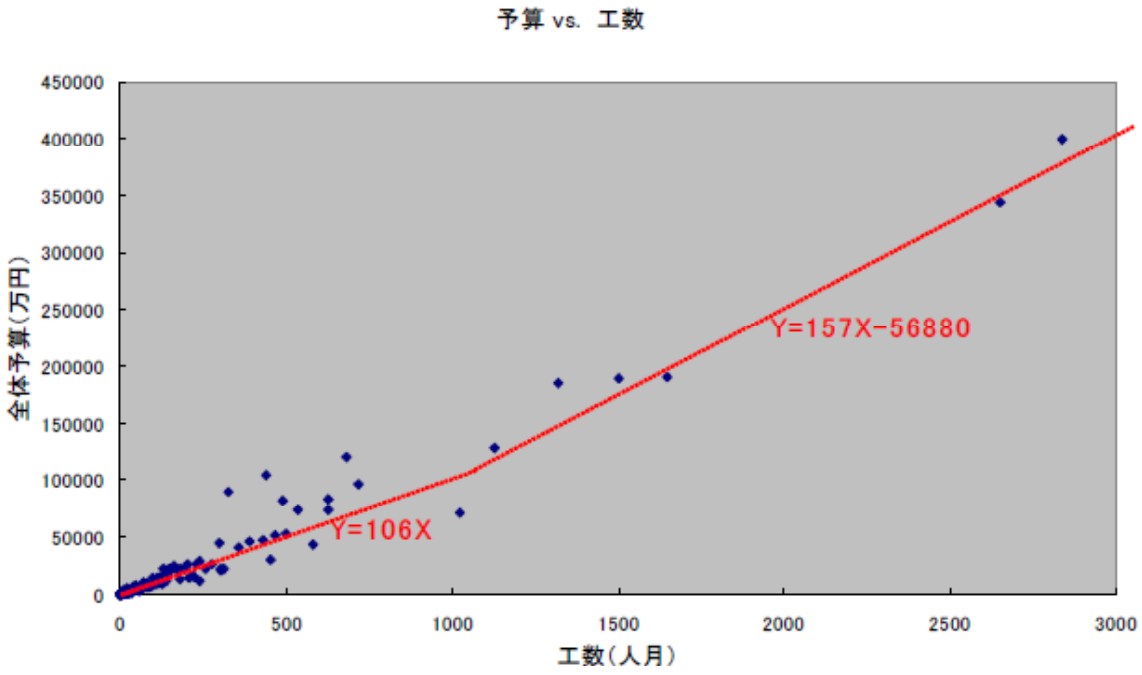
「ソフトウェアメトリックス２００７」JUAS

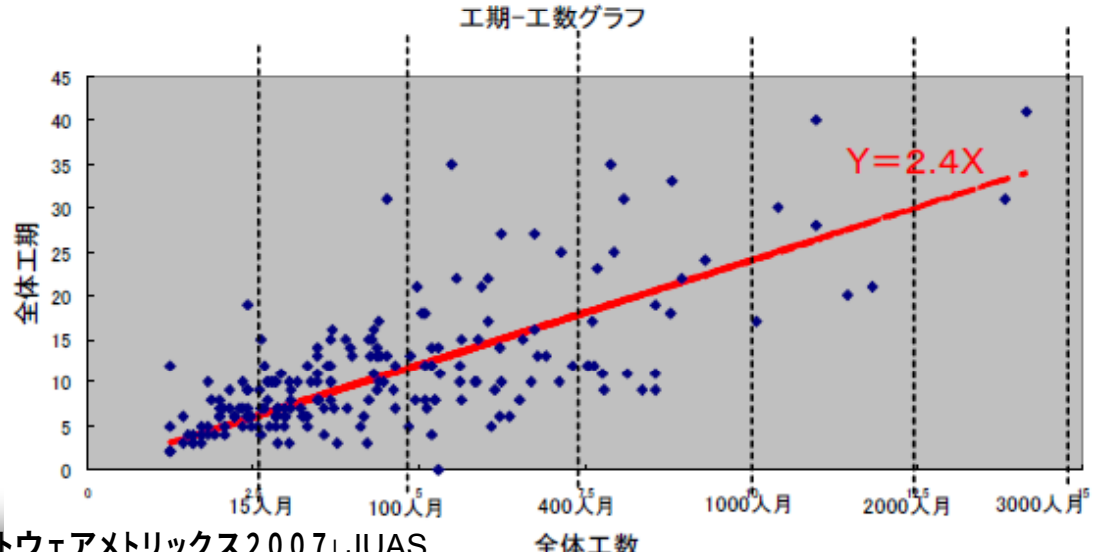
演習

- システム概要
  - 3部門で持っていた販売システムを統合し、顧客情報を一元管理するシステムであり、先進の技術を導入しリアルタイムレスポンスを実現するシステムである。
- データ
  - 予算　25000万円 (設計から導入まで)
  - 工数　350人月
  - 計画工期　10ヶ月 (設計:実装:テスト＝2ヶ月:4ヶ月:4ヶ月)
  - 実績工期　10ヶ月 (設計:実装:テスト＝3ヶ月:5ヶ月:2ヶ月)
  - 欠陥率 (フォロー不具合数:総合テスト2での不具合数)　0.63 (62:32)
  - FP保守範囲　750FP/人
  - 一人当たり対応数　82件/人
- 利用部門の感想
  - 時々止まるが、前のシステムもこんなものだったのであきらめている。
- システム部門の感想
  - 他のシステムに較べて、手間がかかる気がする

- 問題
  - 各データをグラフ上にプロットしてください
  - そのシステムの現状に関する評価をしてください
  - どこでプロジェクトマネージャは手を打つべきだったか考えてください

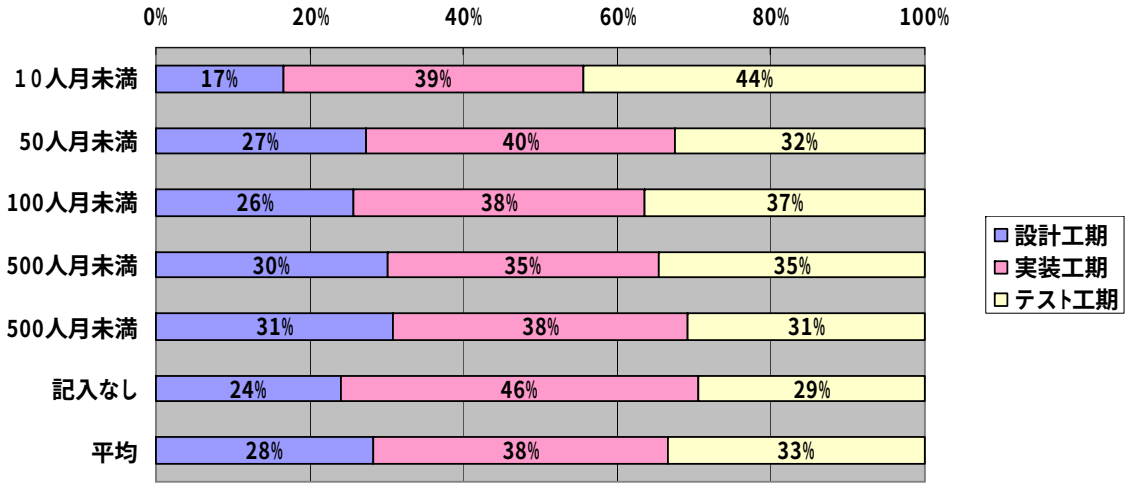
演習



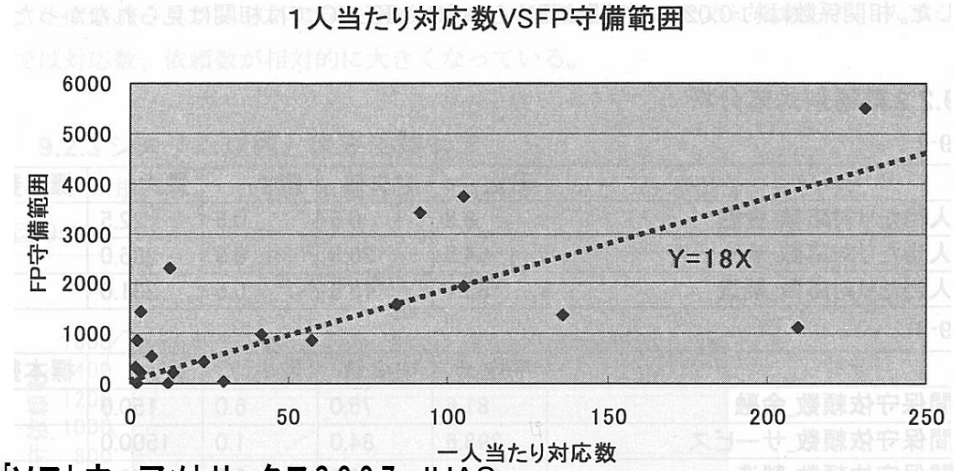


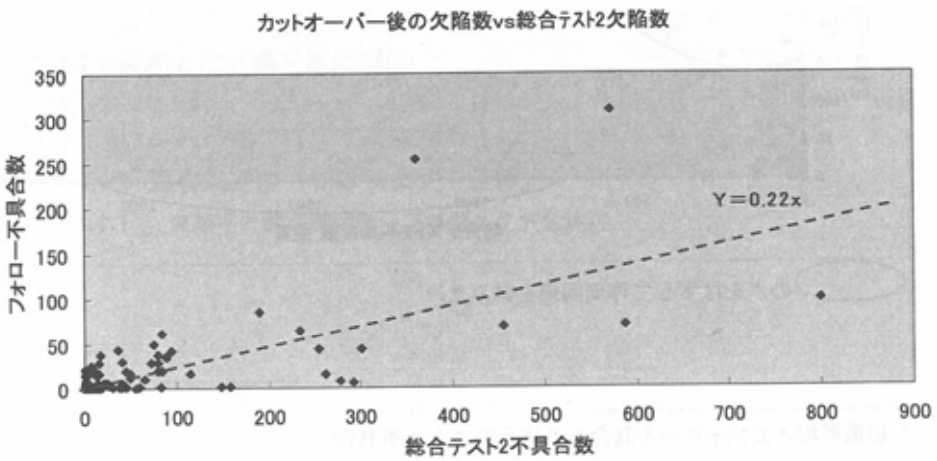
「ソフトウェアメトリックス2007」JUAS

演習



	Aランク	Bランク	Cランク	Dランク	Eランク	Fランク
欠陥率	0	0.25未満	0.5未満	1未満	3未満	3以上
割合	9.7%	33.1%	18.8%	17.5%	14.9%	5.8%
件数	15	51	29	27	23	9





「ソフトウェアメトリックス２００７」JUAS

これだけはデータとしてとるべき項目の洗い出し

欠陥除去には何が有効か

- 全ての工程できちんと検査をしていくことが重要なのはもちろん、前工程で問題点を潰していくことが重要。
- 後工程でバグが見つかると、上流工程で発見されたバグの数倍の労力（コスト、時間）がかかる。
  - 要求段階での修正コストを1とすると、アーキテクチャ設計時で3、システムテスト段階で１０、出荷後で１０～１００倍と言われている

		欠陥除去率の最悪の場合の結果（%）		
		最小	中央値	最大値
設計インスペクション	×	30	40	50
コードインスペクション	×			
品質保証	×			
正規のテスト	×			

		欠陥除去率の最悪の場合の結果（%）		
		最小	中央値	最大値
設計インスペクション	×	32	45	55
コードインスペクション	×			
品質保証	○			
正規のテスト	×			
設計インスペクション	×	37	53	60
コードインスペクション	×			
品質保証	×			
正規のテスト	○			
設計インスペクション	×	43	57	66
コードインスペクション	○			
品質保証	×			
正規のテスト	×			
設計インスペクション	○	45	60	68
コードインスペクション	×			
品質保証	×			
正規のテスト	×			

		欠陥除去率の最悪の場合の結果（%）		
		最小	中央値	最大値
設計インスペクション	×	50	65	75
コードインスペクション	×			
品質保証	○			
正規のテスト	○			
設計インスペクション	×	53	68	78
コードインスペクション	○			
品質保証	○			
正規のテスト	×			
設計インスペクション	×	55	70	80
コードインスペクション	○			
品質保証	×			
正規のテスト	○			
設計インスペクション	○	60	75	85
コードインスペクション	×			
品質保証	○			
正規のテスト	×			
設計インスペクション	○	65	80	87
コードインスペクション	×			
品質保証	×			
正規のテスト	○			
設計インスペクション	○	70	85	90
コードインスペクション	○			
品質保証	×			
正規のテスト	×			

		欠陥除去率の最悪の場合の結果（%）		
		最小	中央値	最大値
設計インスペクション	×	75	87	93
コードインスペクション	○			
品質保証	○			
正規のテスト	○			
設計インスペクション	○	77	90	95
コードインスペクション	×			
品質保証	○			
正規のテスト	○			
設計インスペクション	○	83	95	97
コードインスペクション	○			
品質保証	○			
正規のテスト	×			
設計インスペクション	○	85	97	99
コードインスペクション	○			
品質保証	×			
正規のテスト	○			

		欠陥除去率の最悪の場合の結果（%）		
		最小	中央値	最大値
設計インスペクション	○	95	99	99.99
コードインスペクション	○			
品質保証	○			
正規のテスト	○			

ソフトウェア開発の定量化手法第二版、共立出版

どのタイミングを重視するか

- 入りを制す
  - 正しい予算や計画にすることでコントロールを行う
    - 予算査定や調達などで、失敗プロジェクトの「入り」をコントロール
- 流れを制す
  - 開発工程をモニターすること、傾向把握からコントロールを行う
    - 走り出したプロジェクトが外れた方向に進んでいないか「流れ」をモニター
  - 内容的にはレビューで潰していく
- 出を制す
  - 納品の検査で納品前後のコントロールを行う
    - 運用側で受け入れられるのか「出」をコントロール

入りで確保すべきデータ

- **予算確保･事業計画時**
  - **予算額**
    - システム規模の基本データとして
  - **想定FP (FPが無理なときは工数)**
    - システム規模の基本データとして
  - **工程別期間**
    - 適正工期で開発するかの検証をするため
  - **目標品質**
    - 概要でどのレベルを目指すのか

品質を確保するにはきちんと計画がされている必要があるので、それを検証するためのデータを集める

PJリーダーは、計画策定時にメトリックスを使って検証を行う

PMOは、チェック項目を示すとともに、PJリーダ検証済みのデータを基にその乖離理由などを確認する

流れで確保すべきデータ

- プロジェクト開始時
  - WBS
  - 工数比率 (WBSから作成)
- プロジェクト中
  - 進捗
  - バグ数
  - 仕様変更数
  - 目標品質 (見直し版)
  - レビュー比率

実際に開発工程に入ると品質データが計測されるので、そのデータを分析していく。

出で確保すべきデータ

- テスト計画時
  - テストケース数
  - テストケース密度
- テスト時
  - テスト消化率
  - 障害数
  - テスト工数

検証するときに最終的なテスト報告書だけではなく、そのデータの傾向などを詳細に見ることで、導入後の品質をコントロールすることが可能である

運用後のサービス品質の確保

- サービス品質を左右する非機能要件を確保する方策もプロジェクト管理者には求められる。

機能要件

・情報システムの要件（機能、画面、帳票、情報・データ、外部インタフェース）

非機能要件

・規模・性能要件（規模、性能）
・信頼性等要件（信頼性、拡張性、上位互換性、システム中立性、事業継続性）
・情報セキュリティ要件（権限、情報セキュリティ対策）
・システム稼働環境（全体構成、ハードウェア構成、ソフトウェア構成、ネットワーク環境、アクセシビリティ）
・テスト要件
・移行要件（移行、教育）
・運用要件（情報システムの操作・監視等、データ管理、運用施設・設備）
・保守要件（ソフトウェア保守、ハードウェア保守）
・作業の体制及び方法（作業体制、開発方法、導入、瑕疵担保責任）
・特記事項

- 非機能要件の品質を確保するためには、システム企画段階からの計画が必要である
  - 要件定義でしっかり定義しておき、SLAでコントロールする。

品質属性のトレードオフ

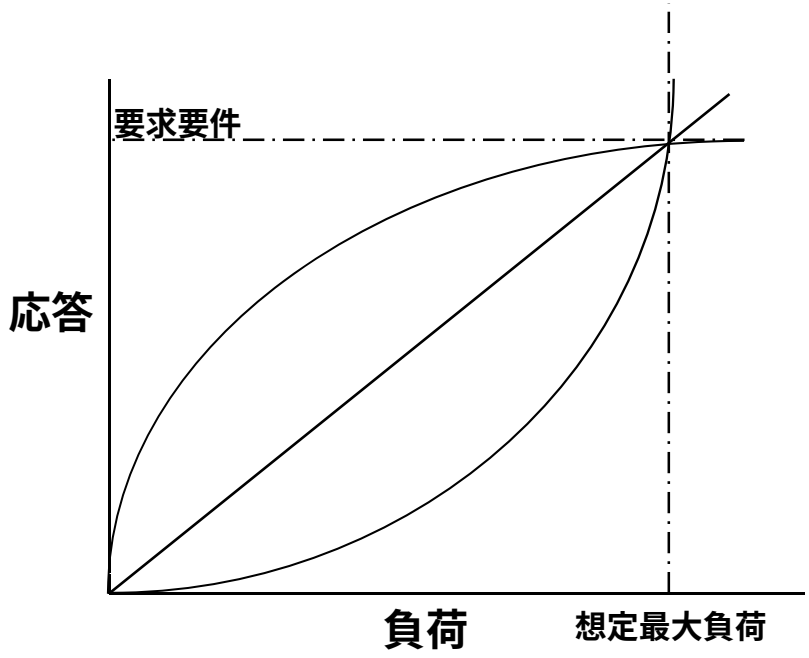
◆ 非機能品質にはトレードオフの関係があり、全てを満足させることは難しい。以下は一例であるが、開発システムにあったポイントで性能を精査していく必要がある

out \ in	正確性	ユーザビリティ	効率性	信頼性	完全性	適応性	正当性	堅牢性
正確性	↑	↑	↓		↑		↑	↑
ユーザビリティ		↑				↑	↑	
効率性	↓		↑	↓	↓	↓	↓	↓
信頼性	↑	↑		↑	↑		↑	↑
完全性	↑	↑	↓	↑	↑		↑	
適応性	ここを高めるには何をするのか？等				↓	↑		↓
正当性	↑	↑	↓	↑		↓	↑	↓
堅牢性			↓	↑	↑	↓	↑	↑

「ソフトウェアテスト手法」高橋、湯本を参考に作成

パフォーマンステスト

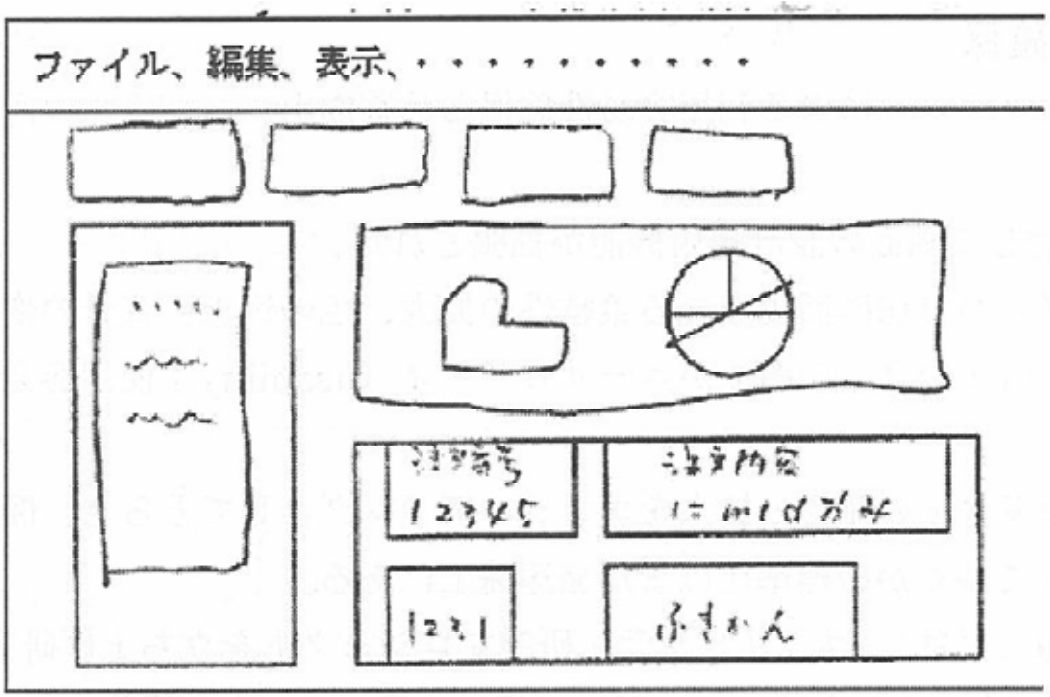
- システムにおける性能は非常に重要な要素である。
  - 最大負荷時に要件を満たすのか
  - 想定負荷までの応答の状況はどうなっているのかなど検証していく必要がある
  - また、同じシステム上で並行稼動しているプロセスがある場合は、その条件もテスト時に確認されているか検証する必要がある。



サービス品質としてのユーザビリティ

- 一般に性能の次に問題になるのがユーザビリティである。ユーザビリティは、単にユーザの満足度を高めるだけではなく正確な作業を支える大切な役割を担っている。
- この確認には、画面設計時にユーザビリティの使用を明確にしておくとともに、プロトタイピングとウォークスルーによって検証を行うことが有効である。
  - ペーパープロトタイピング
    - 設計初期に行い画面のイメージを手書きの画面で検証する
  - スクリーンタイプアプローチ
    - ペーパープロトタイピングをPPTなどで清書したもので検証する
  - オンラインタイプアプローチ
    - じっすぁいの操作イメージで検証する

ペーパープロトタイピングの例



＜チーム編成＞

①ユーザー

②コンピュータ役

③進行役

④観察者

＜作業手順＞

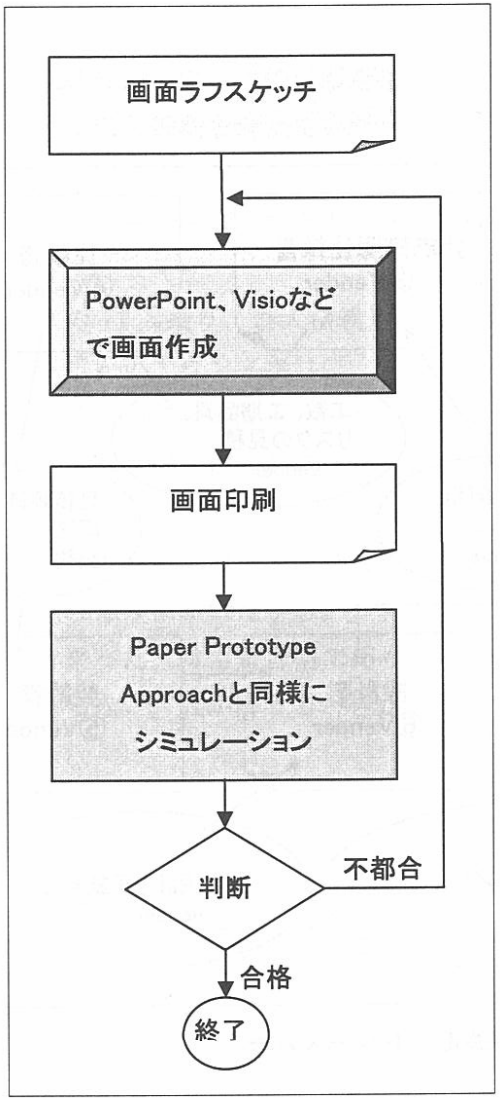
①手書きの画面レイアウトに対してユーザーはデータを書く

②コンピュタ役はその結果に黙って紙の画面上に反応を返す

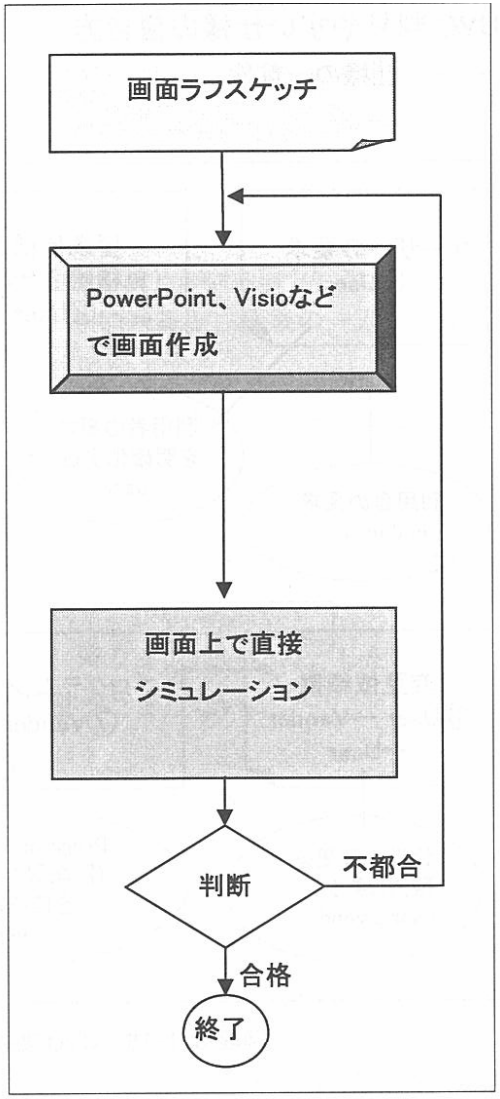
③その結果を観察役がチェックし状況、課題を記録する。

④①～③を繰り返し課題解消を図る

スクリーンタイプ・オンラインタイプの流れ



図表 3-1-13　Screen Type Approach



図表 3-1-14　Online Type Approach

サービス品質を担保するSLA

- サービス導入後には、SLA指標を設定し管理を行う。また開発データとともに管理を行い、開発の検証もあわせて行う。
  - 導入後の不具合数
  - 導入後の保守コスト
  - 導入後の保守工数
  - 稼働率
  - 平均故障間隔
  - 平均修復時間
  - 満足度　　等
- SLAは罰則を与えるためではなく改善のために使うことが重要である。
  - 具体的な指標はJEITAのSLAガイドが有効である。

演習

- 要件定義におけるサービス品質の記述について、以下の文書を修正してください。

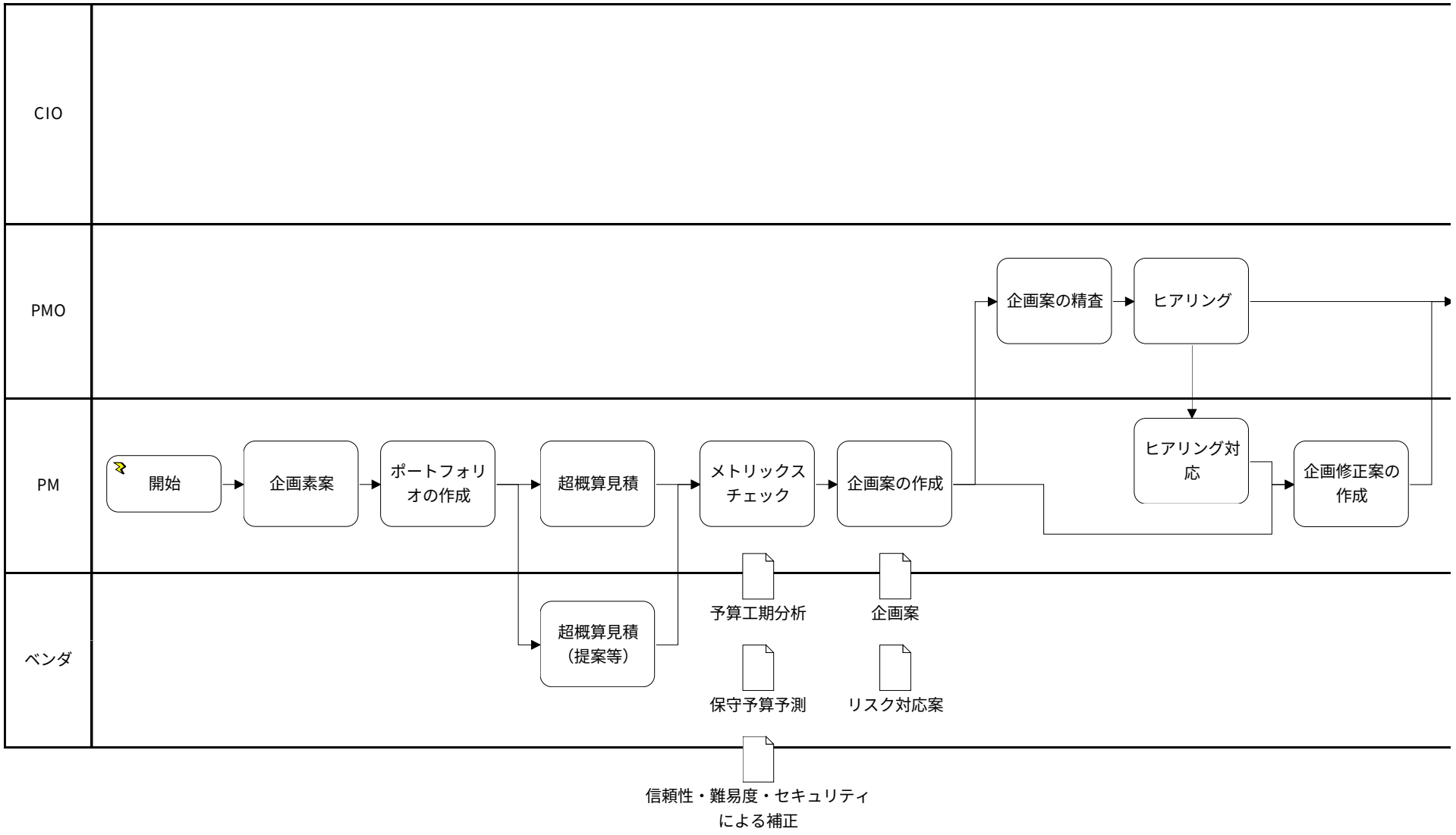
  - システムの稼働時間は工場の操業1時間前から停止1時間後までとする。
  - 従来のシステムと同じ使い勝手を実現すること。
  - データのバックアップは日次で行い。復旧が容易であること
  - 高齢職員の利用に配慮すること

企画開発など各工程で比較活用する方法

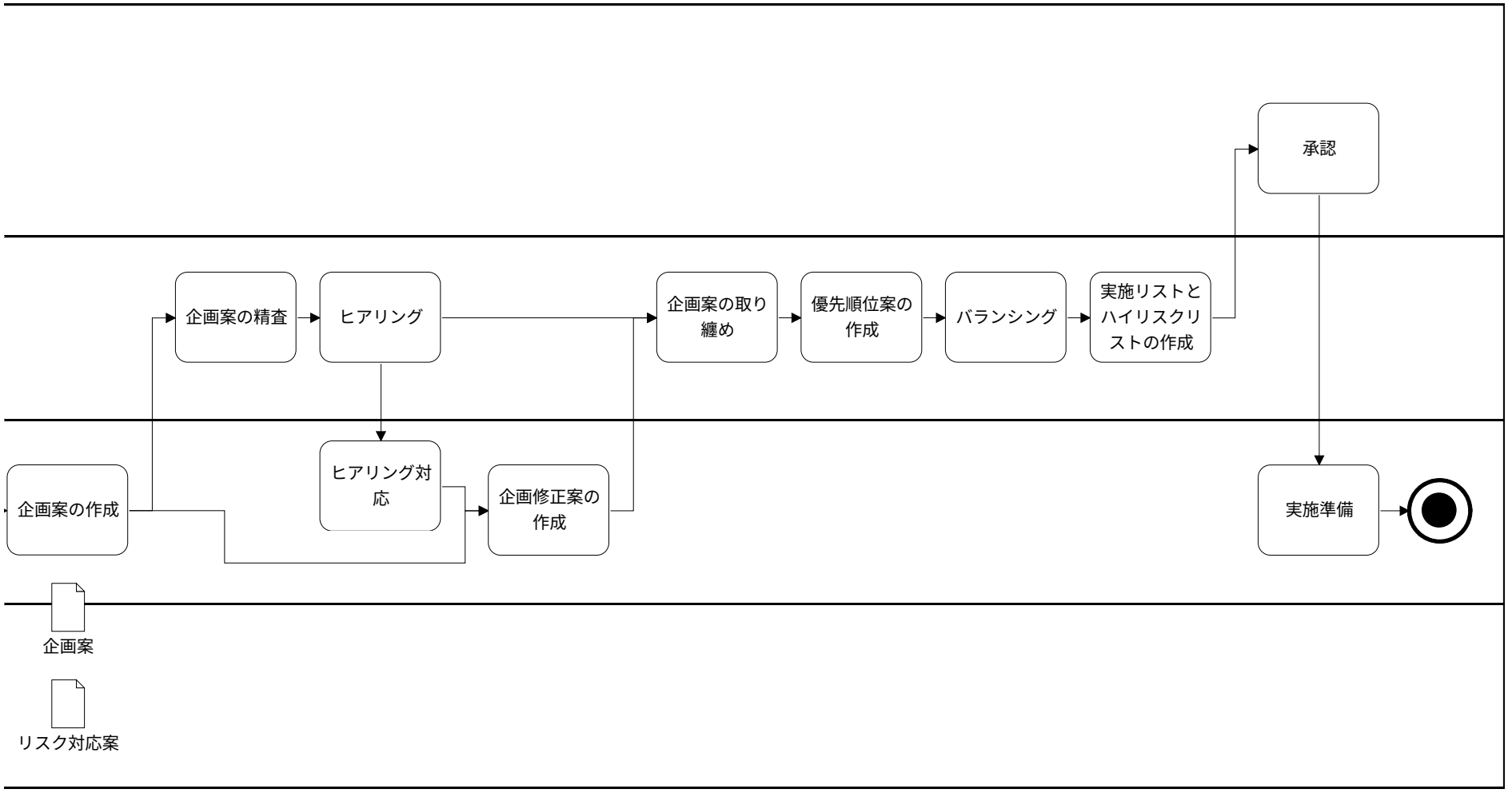
入りを制す

正しく計画されたプロジェクトは失敗が少ない

入り (企画) のプロセス1


CIO
PMO
PM
ベンダ
開始
企画素案
ポートフォリオの作成
超概算見積
メトリックスチェック
企画案の作成
企画案の精査
ヒアリング
ヒアリング対応
企画修正案の作成
超概算見積（提案等）
予算工期分析
企画案
保守予算予測
リスク対応案
信頼性・難易度・セキュリティによる補正

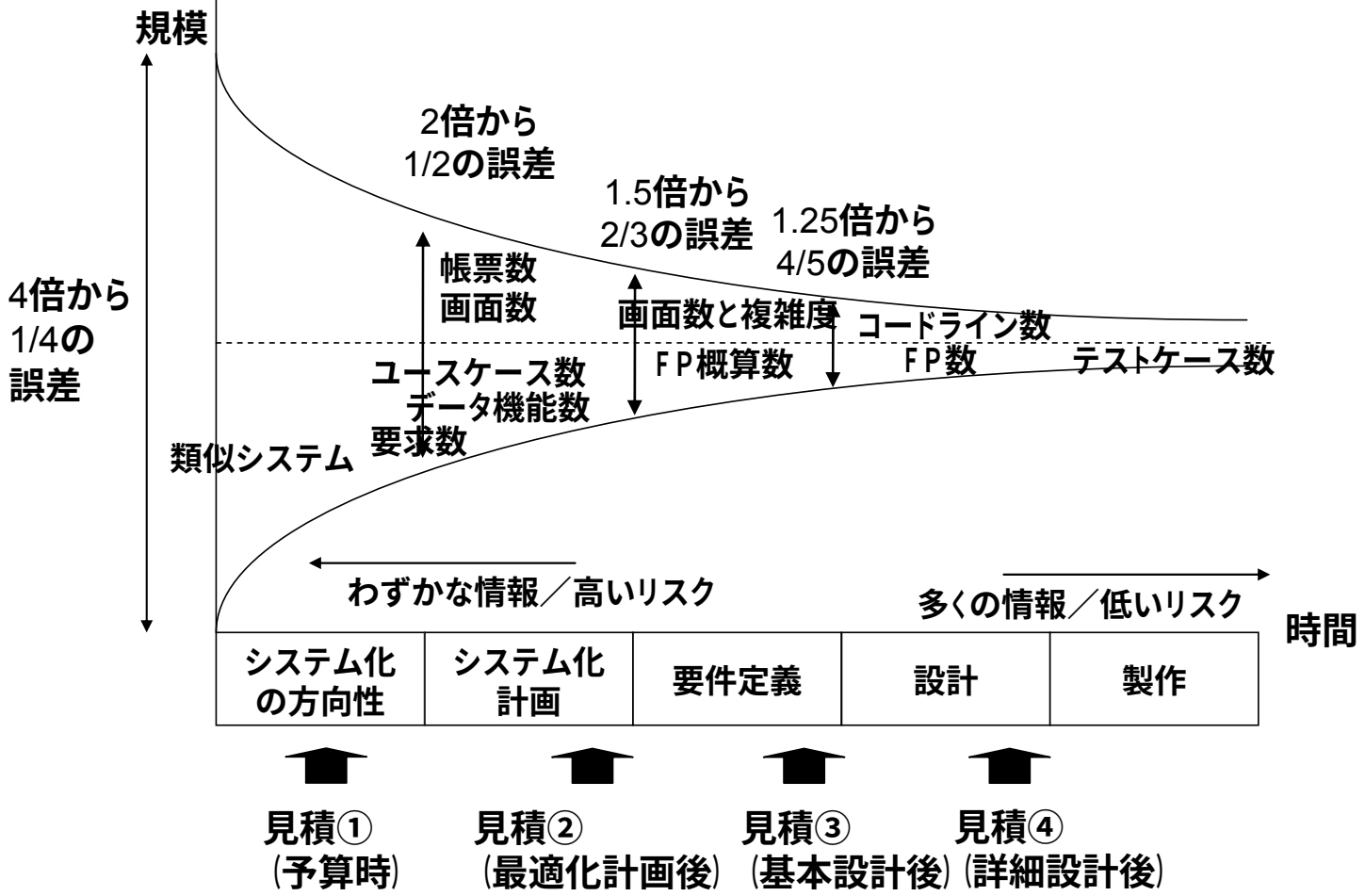
入り (企画) のプロセス2



ティ

予算がゆがむと全てがゆがむ

- 予算は工程が進むにつれて正確になっていくが、あまりに安値の見積もりを突き通すと品質にも大きく影響を与えることがある。



出典:「ITユーザとベンダのための定量的見積りの勧め」,SEC,2006
「ソフトウェアの規模決定、見積もり、リスク管理」富野他2008

参考：FPは正確か？

- More Betterという程度
- 本来はファンクションを基にしているので正確

- BUT!!
- ベンダは経験を基に補正することが多い
  - パターン1
    - このシステムならこのくらいの金額かな
    - FPに換算するとこのくらいかな
    - では、こんな割り振りで
  - パターン2
    - FPで積み上げ
    - もう少し高くならないとおかしいな
    - FPを増やして調整しよう

- 何でそんなことが起こるのか？
  - FPに関する経験不足

- でも、何も指標がないより重要だし、最近では統計も増えてきているので、よりよい方向に向かっている。

様々なせめぎあいで調整が難しい

- 予算だけではなく要件でもせめぎ合いがあるが、ここをきちんとやらないと後工程の品質管理に影響が及んでくる。

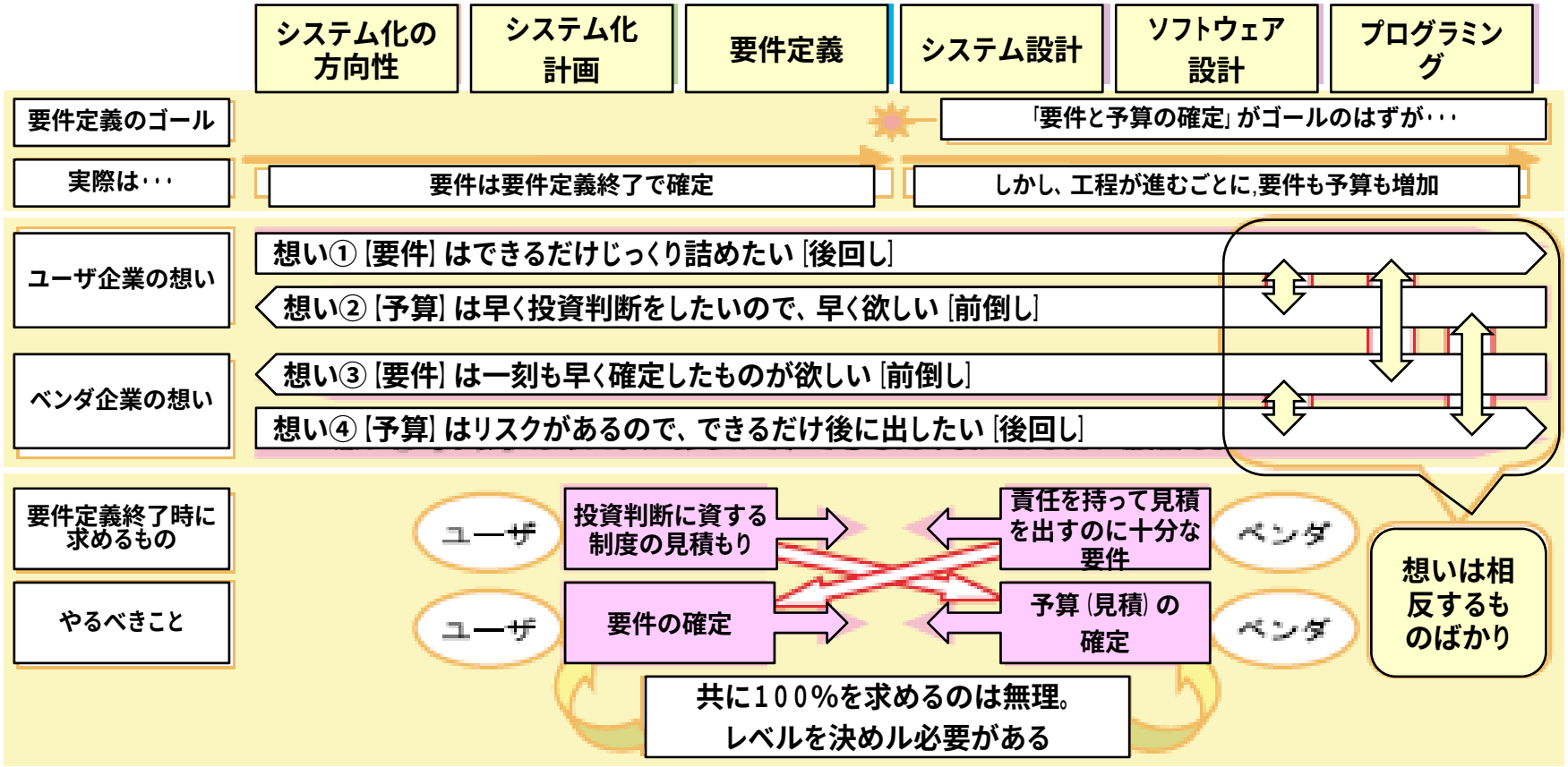


図 4.6　おのおの異なる想い

経営者が参画する要求品質の確保、情報処理推進機構

参考：生産性環境変数による見積もり精度の向上

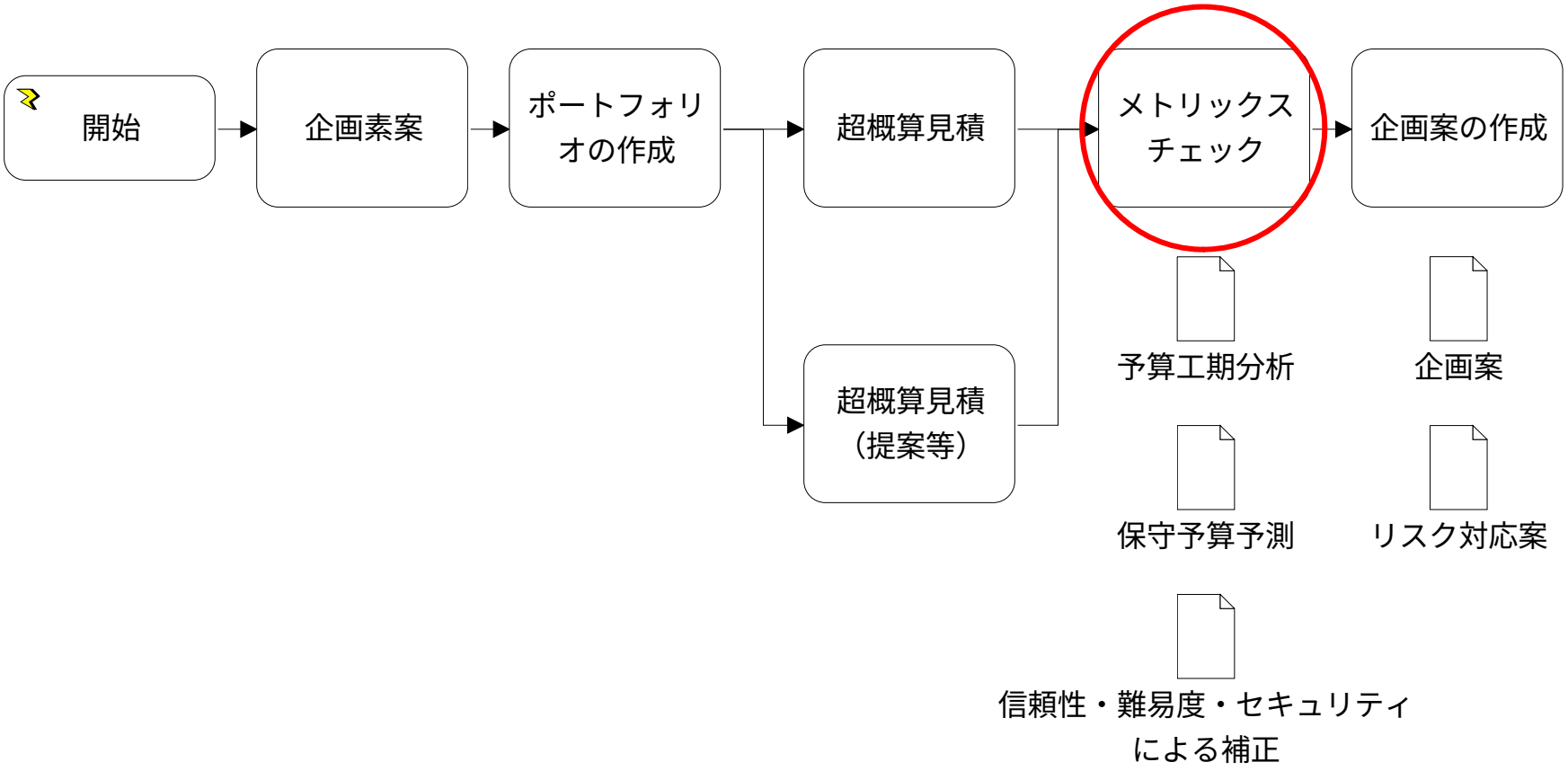
- システムリファレンスマニュアルで紹介されるJASTECのモデル（左図）
- JISA 品質ベースの価格設定の可能性に関する調査（右図）

生産性見積り方式　Pi ＝ PBi × (1+Σ βij)　βij:生産性環境変数

生産性特性	副生産性特性	評価の観点 内容	特記、特例事項	外貨評価基準（見積プレゼン用） 影響度(%) 要件定義	設計	製作	テスト		内貨評価基準（参考） 影響度 設計	製作	テスト
業務特性	業務ナレッジ(P01)	・顧客／当社の開発対象業務に対する業務ナレッジが生産性に及ぼす影響を、それぞれ（外貨／内貨）個別に評価する ・外貨評価はユーザー部門を含む顧客プロジェクト組織全体の能力として評価する ・内貨評価は当該プロジェクトへのアサインベースで評価する	→当の中にない業務モデル等の創造、システム化の要素も含む。 → プロジェクト全体としての方針・仕様意思決定能力（旧組織複合度の要素を取り込む） → 実／予定アサインメンバーの業務ナレッジで評価	-10	-10	—	-10	中核メンバー全員が当該業務経験多数				
				-5	-5	—	-5	中核メンバーの一部の当該業務経験少				
				0	0	—	0	中核メンバーの一部に未経験者あり				
				40	8	—	8	中核メンバーに当該業務経験者不在				
				50	10	—	10	メンバー全員が当該業務経験なし				
ハードウェア特性	安定度／信頼度／使用実績（P02）	・システムもしくは製品となるハードウェアの安定度・信頼度を外貨として評価し、当社使用実績を内貨として評価する （注）「使用実績がある」ということは、ハードウェアの特性を把握し、その対策は既知であるが故に、生産性が高いということ。	→ 実／予定アサインメンバーの使用実績で評価	—	-5	—	-5	当該HWでの効果的開発手法を保有				
				—	-3	—	-3	当該HWでの開発実績多数（安定度高）				
				—	0	—	0	当該HWでの開発実績が当社基準での平均				
				—	3	—	3	顧客に使用実績のないHW				
				—	5	—	5	世間で使用実績のないHW				
ソフトウェア特性	安定度／信頼度／使用実績(P03)	・システムもしくは製品となる他社作成ソフトウェアもしくはCotsの安定度・信頼度を外貨として評価し、当社使用実績を内貨として	→ 実／予定アサインメンバーの使用実績で評価 →要件定義の「－」は、ERPを利	—	-5	—	-5	当該SWでの効果的開発手法を保有				
				—	-3	—	-3	当該SWでの開発実績多数（安定度高）				
				—	0	—	0	当該SWでの開発実				

図表４　プロジェクト要件の整理

分類	プロジェクト要因
ベンダー側に起因するプロジェクト要件	プロジェクトリーダ（プロジェクトマネージャ）のマネジメント業務に対する熟練度
	プロジェクト要員の業務・システムに対する熟練度
	プロジェクト要員の充足度
	プロジェクトの外部委託先の業務・システムに対する熟
	プロジェクト推進体制の適切性
	プロジェクトのレビュー体制の適切性
	プロジェクトのコミュニケーションの充実度
	プロジェクトのコミュニケーション基盤の効率性
	プロジェクトのドキュメント管理の充実度
	ベンダー側の役割分担・責任所在の明確性
	ベンダー側のPMO、品質管理部門等の品質保証体制
	ベンダー側の定量的な品質管理基準の有無
	ベンダー側の顧客窓口の適切性
顧客側、及びベンダー・顧客間に起因するプロジェクト要件	顧客側のプロジェクトリーダ（プロジェクトマネージャ）のマネジメント業務に対する熟練度
	顧客側の業務部門の担当者（利用部門の担当者）のシステム化対象業務に対する熟練度
	顧客側のシステム部門の担当者の開発システムに対する熟練度
	顧客・ベンダー間の役割分担・責任所在の明確性
	顧客側のプロジェクト推進体制の適切性
	顧客側のレビュー体制の適切性
	顧客側のPMO、品質管理部門等の参画の有無
	顧客側のベンダー窓口の適切性
	顧客側の他プロジェクトとの依存関係の度合い
	顧客側と合意した開発期間の適切性
	顧客側から提供される業務・システム関連資料の充実度・理解性
	顧客・ベンダー間のコミュニケーションの充実度
	顧客・ベンダー間のコミュニケーション基盤の効率性
	開発するソフトウェアに対するプラットフォームの安定度
	当該システム重要度


開始
企画素案
ポートフォリオの作成
超概算見積
メトリックスチェック
企画案の作成
超概算見積（提案等）
予算工期分析
企画案
保守予算予測
リスク対応案
信頼性・難易度・セキュリティによる補正

情報システム予算の分析

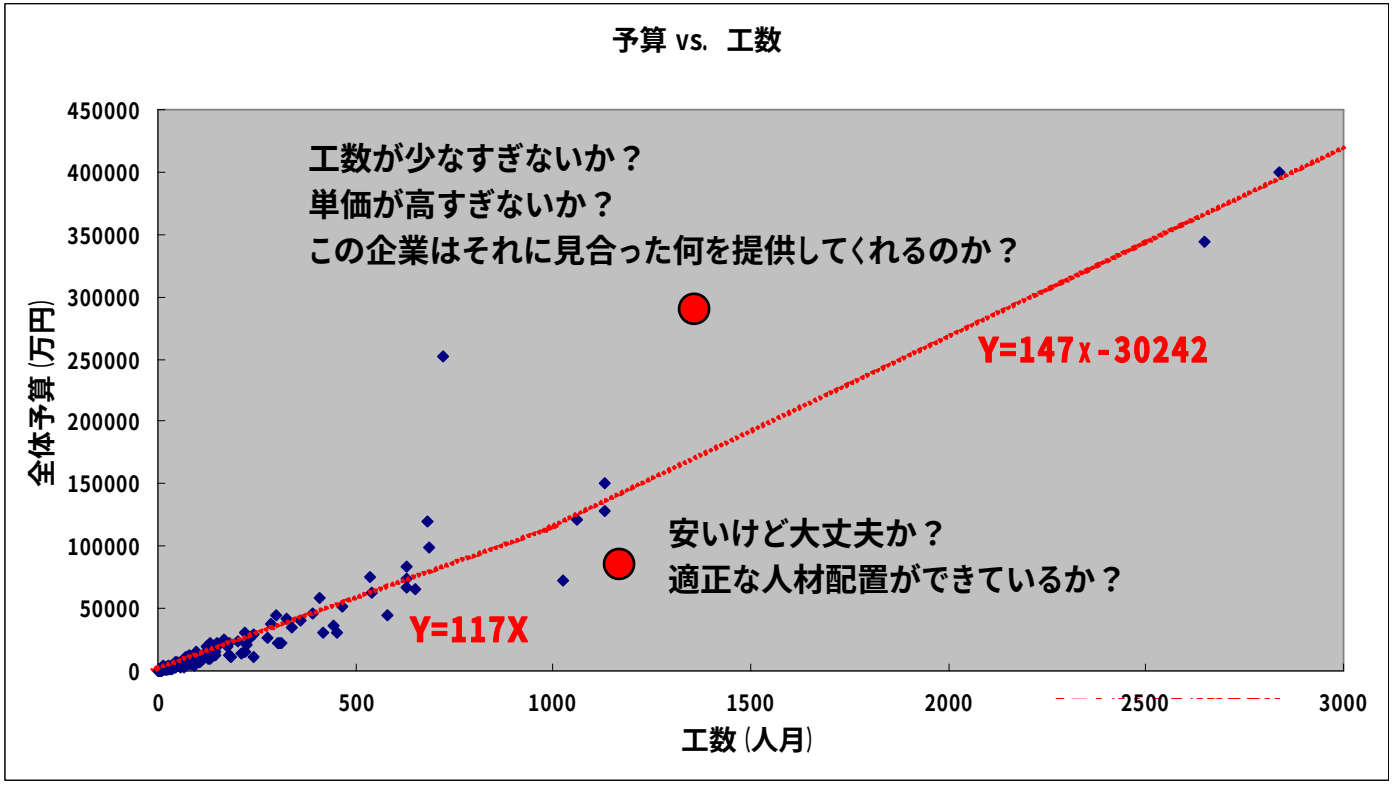
- 予算におけるハードウェア、ソフトウェア、パッケージソフト、通信、運用などサービス、省内担当職員人件費、その他は、平成18年度情報処理実態調査から、企業においては以下の比率になる。
  - この数値をもとに、要求されている予算の妥当性を検証していく、但し、あくまでも企業における平均値であり、システムの特性に寄って費用比率は変わってくるのであくまで上記の数値は参考値である。

分類	比率	備考
ハードウェア	１７．７％ 12.4%	買い取り、減価償却、レンタル、リース等
ソフトウェア	２７．５％ 36.4%	買い取り、減価償却、レンタル、リース、開発・カスタマイズ関連費用等
通信関連	４．５％ 6.2%	
運用などサービス	２８．０％ 31.9%	データ入力、運用･保守委託料、教育、外部派遣要員等
職員人件費	１３．５％ 1.7%	
その他	８．９％ 11.4%	

概算や企画案は、この比率に大きく外れていないのか？

予算と工数の関係

- ユーザ企業ソフトウェアメトリックス調査２００７によると、予算と工数の関係は以下の関係で関係付けられる。
- このグラフに予算要求データをプロットすることにより工数の妥当性を判断することができる。標準的なデータから５０％以上乖離がある場合には、計画を提出した原課に理由を確認することが望ましい。



「ソフトウェアメトリックス２００８」JUAS

工程別の契約単価

- 工程別の平均的な契約単価は以下のとおりである。ただし、あくまでも各プロジェクトの平均値であり、各プロジェクト内では、さらに金額の分布があることに留意が必要である。

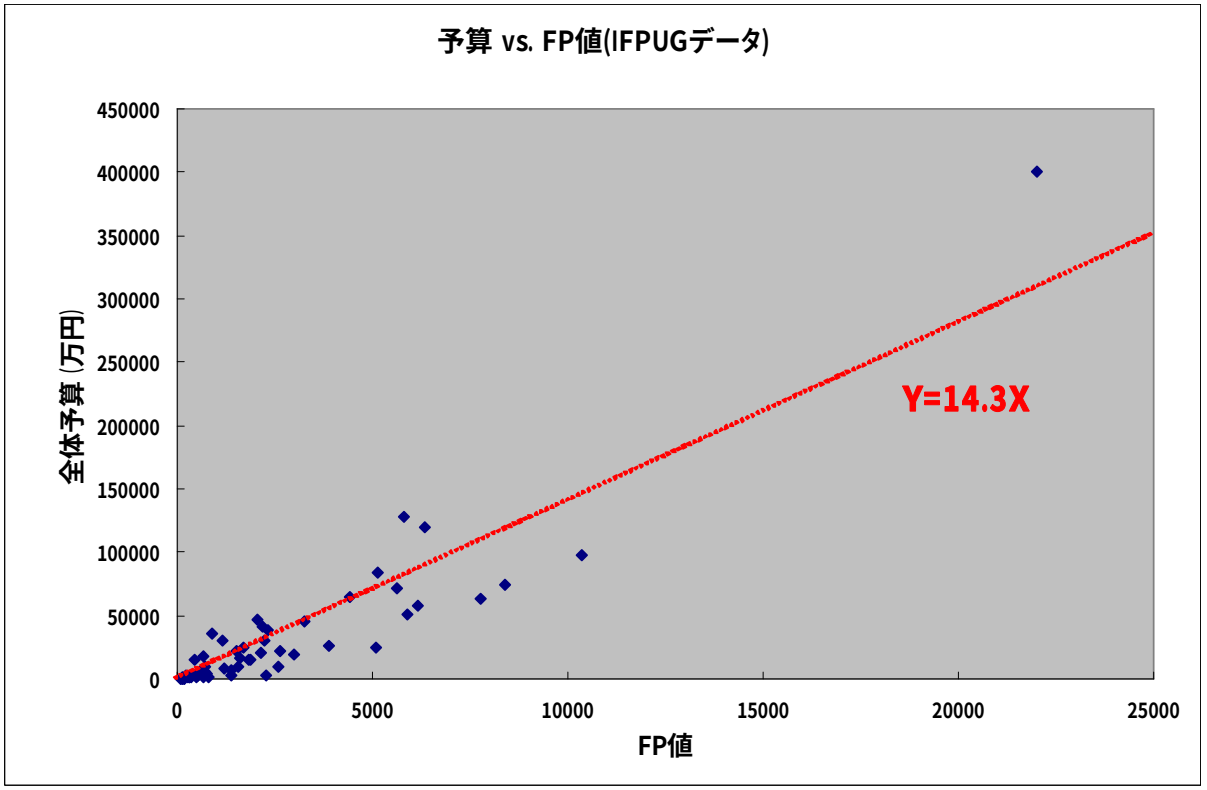
		工程別単価(万円/月)				
		要件定義単価	設計単価	実装単価	テスト単価	トータル単価
パッケージ開発	件数	6	7	7	7	10
	最大値	300.0	250.0	200.0	250.0	250.0
	平均値	165.7	140.1	120.4	134.3	127.3
	最小値	100.0	97.0	73.0	90.0	83.0
スクラッチ開発	件数	55	69	70	69	49
	最大値	170.0	170.0	170.0	168.0	157.0
	平均値	110.7	105.7	86.3	96.5	96.3
	最小値	60.0	60.0	50.0	55.0	60.0
合計	件数	61	76	77	76	59
	最大値	300.0	250.0	200.0	250.0	250.0
	平均値	116.1	108.9	89.4	100.0	101.6
	最小値	60.0	60.0	50.0	50.0	60.0

低価格での提案に対するチェック項目

- 本当にできるんですねと念を押す
- 仕様変更に対する変更条件を確認する
- ソフトウェアエンジニアリングデータによる品質要求
- サービス品質の要求

基礎データとしてのFPの算出

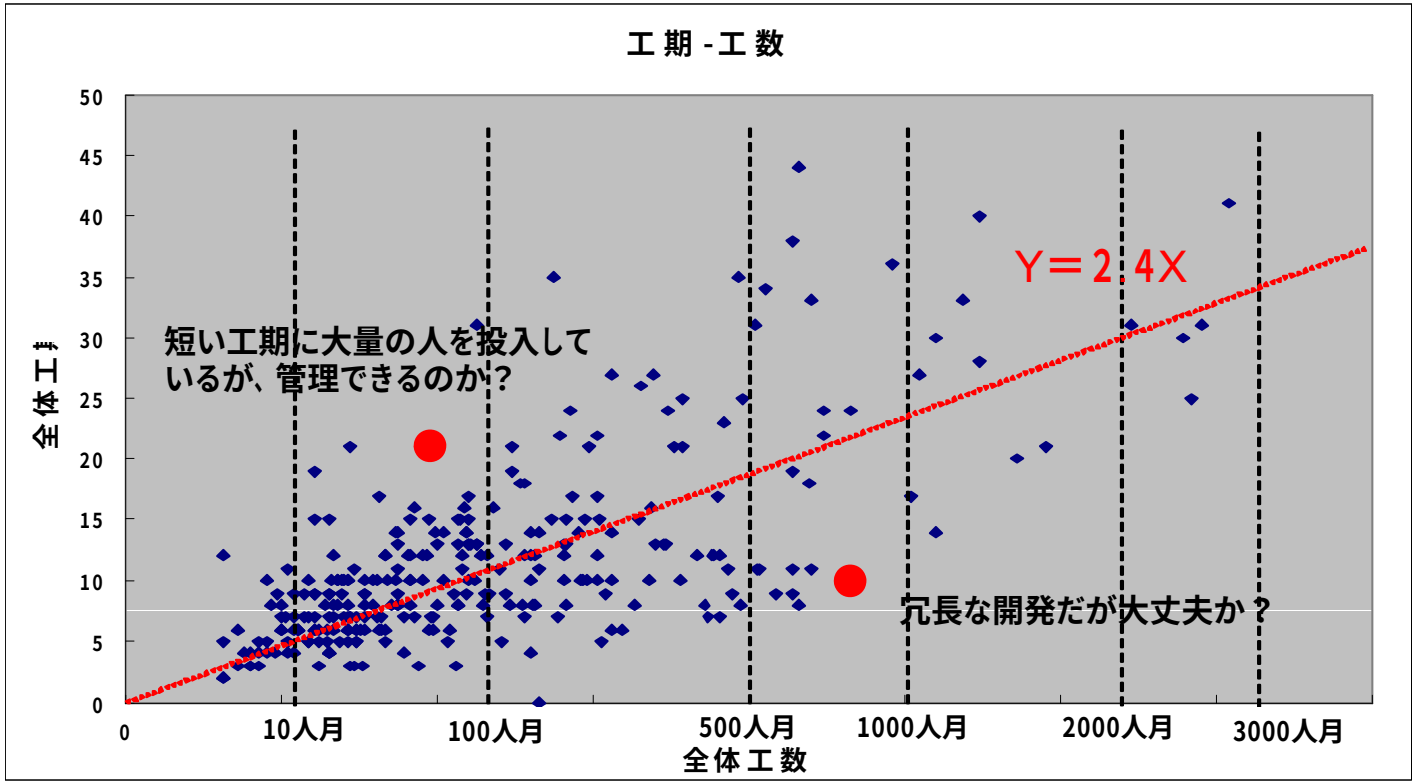
- FPでないシステムは、予算やステップ数を元に概算のファンクションポイントを推定する。ファンクションポイントを基軸にすることで、開発中の各種情報を横串の比較、推定することが可能になる。



「ソフトウェアメトリックス２００８」JUAS

工数と工期の関係

- 予算を元に算出した工数を元に標準的な工期を推定することが可能になる。
- 開発全体工期（月）＝2．38×（工数（人月）の3乗根）



「ソフトウェアメトリックス２００８」JUAS

PMOは、次ページ以降の工期と結果の関係をPJリーダーに示し、その対応策があるのか、覚悟ができているのか確認しなければならない

工期と満足度の関係

- 標準的な工期に対して短工期、長工期の分布が25%になるようにして分析を行うと以下の結果がある。標準的な工期に対して短工期で行う場合にはリスクが高まることから、短工期開発にする理由を確認するとともに、短工期開発で行う場合のリスク回避対策（優秀なPMを配置する等）を提示させる必要がある。

工期乖離度		顧客満足度(プロジェクト全体)					
		満足	やや不満	不満	未回答	総計(割合)	
長工期	件数	47	20	0	3	70	24.2%
	割合	67.1%	28.6%	0.0%	4.3%	100.0%	
適正工期	件数	93	40	9	7	149	51.6%
	割合	62.4%	26.8%	6.0%	4.7%	100.0%	
短工期	件数	43	22	2	3	70	24.2%
	割合	60.4%	31.3%	4.2%	4.2%	100.0%	
総計	件数	183	82	11	13	289	100.0%
	割合	63.3%	28.4%	3.8%	4.5%	100.0%	

「ソフトウェアメトリックス2008」JUAS

工期乖離と遅延、品質との関係

- 長工期プロジェクトは余裕があるのが油断につながるのか遅延や品質問題の割合が大きい。

工期乖離度		工期遅延度							遅延度20%以上の割合
		予定より早い	予定通り	10%未満	20%未満	50%未満	それ以上	総計(割合)	
長工期	件数	3	40	8	5	6	5	67	16.4%
	割合	4.5%	59.7%	11.9%	7.5%	9.0%	7.5%	100.0%	
適正工期	件数	3	103	9	10	13	6	144	13.2%
	割合	2.1%	71.5%	6.3%	6.9%	9.0%	4.2%	100.0%	
短工期	件数	13	45	1	4	5		68	7.4%
	割合	19.1%	66.2%	1.5%	5.9%	7.4%	0.0%	100.0%	
総計	件数	19	188	18	19	24	11	279	12.5%
	割合	6.8%	67.4%	6.5%	6.8%	8.6%	3.9%	100.0%	

工期乖離度		換算欠陥率							平均欠陥率
		0	0.25未満	0.5未満	1未満	3未満	3以上	総計(割合)	
長工期	件数	3	14	13	5	8	6	49	1.29
	割合	6.1%	28.6%	26.5%	10.2%	16.3%	12.2%	100.0%	
適正工期	件数	7	51	27	12	10	0	107	0.35
	割合	6.5%	47.7%	25.2%	11.2%	9.3%	0.0%	100.0%	
短工期	件数	4	32	6	3	5	0	50	0.27
	割合	8.0%	64.0%	12.0%	6.0%	10.0%	0.0%	100.0%	
総計	件数	14	97	46	20	23	6	206	0.55
	割合	6.8%	47.1%	22.3%	9.7%	11.2%	2.9%	100.0%	

「ソフトウェアメトリックス２００８」JUAS

工期短縮に対する対策

◆ 工期短縮に対しては適切な対応を行う必要がある。

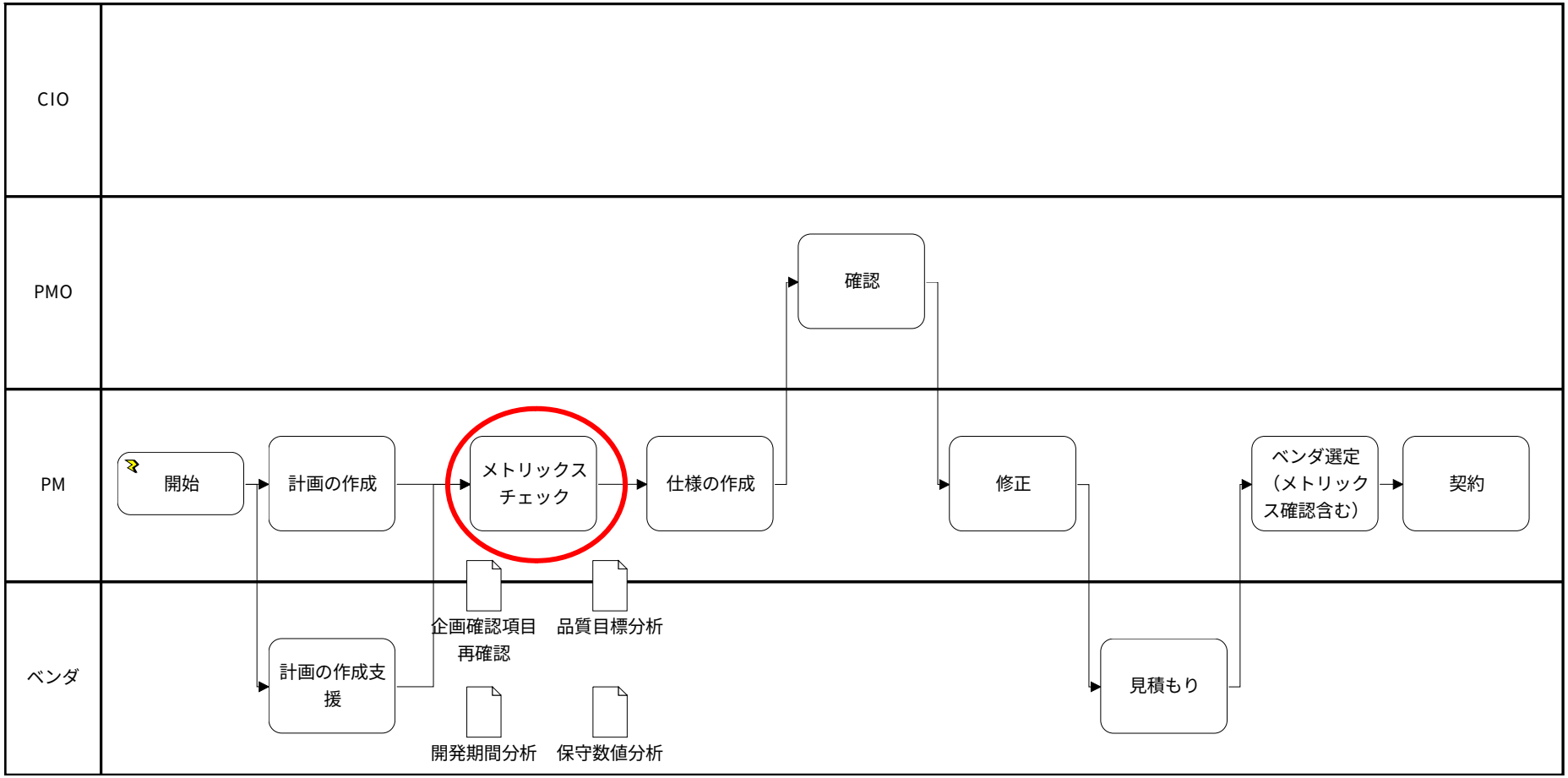
	標準より長い工期	標準	25%工期短縮	25%以上工期短縮
工期の標準の考え方	金融等欠陥の発生を無くしたい品質重視のプロジェクトの場合	工数の立方根の2.4倍（例：1000人月のプロジェクトは24ヶ月）	・ユーザの要望 ・流通業のシステム化などに多い。	ユーザのやむを得ない外的事情で実施する場合（対コンペ戦略、新商品の販売、株式の上場、企業の統合など）
スケジューリングの対応	充分なシステムテスト期間の確保	中日程計画の充実（役割分担別WBS管理）	中日程計画の充実 （週間別管理）	小日程計画の充実（日別管理）
その他の対応策	・品質重視のテスト計画書及びテストケースの緻密化 ・安定稼動のための分割立ち上げ等	・WBSによる総合計画と局面化開発 ・レビューの徹底 ・テストケース充実 ・コンバージョンデータのフル活用 ・確実な変更管理	同左　＋ ・PGの選抜 ＊標準化の徹底と実力のある一括外注の採用。 ・システム範囲、対象の部分稼動 ・RAD＋DOA ・性能事前検証 ・変更管理の強化	同左　＋ ・ベテランＰＭによる采配と会社あげての協力及び監視 ・パート図での計画 ・ベストメンバー選出 ・クリーンルーム手法 ・二交代制の配置 ・顧客主体のテストチーム設置 ・パッケージの活用 ・部分の再利用 ・オープンな進捗情報管理

出典：「「システム・リファレンス・マニュアル（ＳＲＭ）」JUAS,2005

ハイリスクリストへの登録

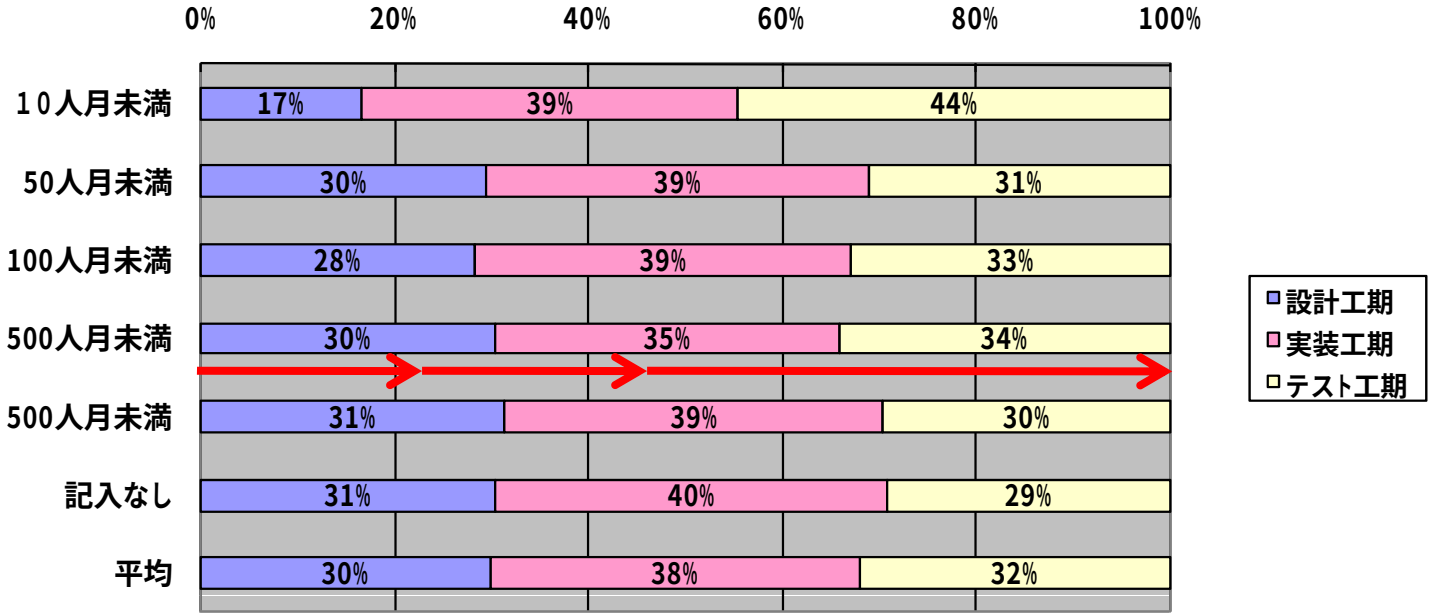
- ここまでの確認で、支店間システムなので通信料が一般より高いなどのPMOが納得できる理由があるもの以外の一般値より外れたプロジェクトは、ハイリスクプロジェクトとして管理する。
  - ハイリスクリストに関する管理
    - 管理頻度を高める
    - 管理項目を多くする
    - リスク報告を求める
  - ハイリスクプロジェクトも、プロジェクトが進むうちにハイリスクでなくなる場合もある。その場合にはリスト指定を解除する。
  - ハイリスクプロジェクトは、メトリックスだけで作られるものではない。ポートフォリオ評価でリスク指定されるものもある。

入り（予算時）のプロセス


CIO
PMO
PM
ベンダ
確認
開始
計画の作成
メトリックス
チェック
仕様の作成
修正
ベンダ選定
（メトリック
ス確認含む）
契約
企画確認項目
再確認
品質目標分析
計画の作成支
援
見積もり
開発期間分析
保守数値分析

予算に対する開発期間配分

- 予算に対する標準的な開発期間の配分がある。この配分から大きく外れている企画に対しては、妥当性の確認を行う必要がある。



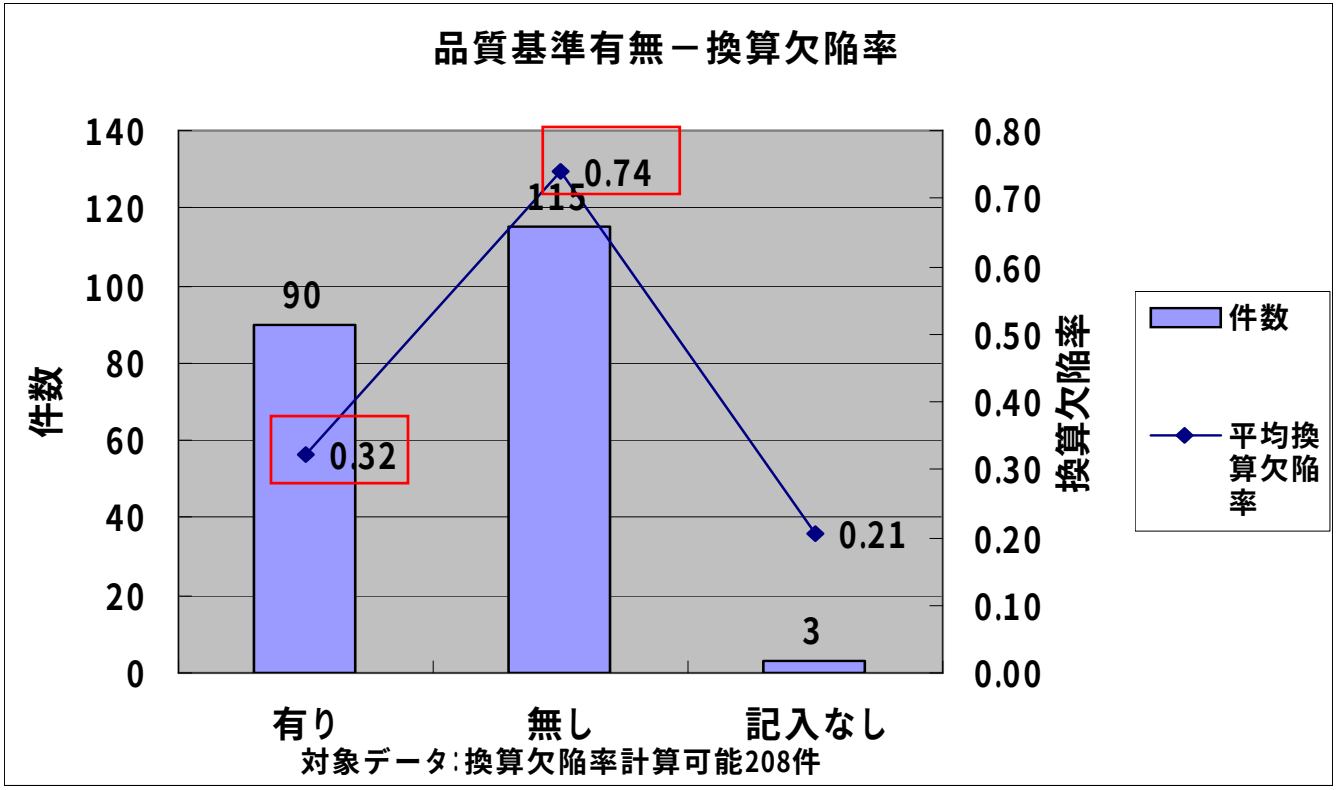
「ソフトウェアメトリックス２００８」JUAS

高信頼性システムではテスト期間を通常よりも多くするなど配慮する必要がある。

通常の1.5倍のテストを実施するならば相応の期間が必要等の配分が必要であるが、現在どのくらいが適正化の指標踏破できていない

品質基準の有無と満足度

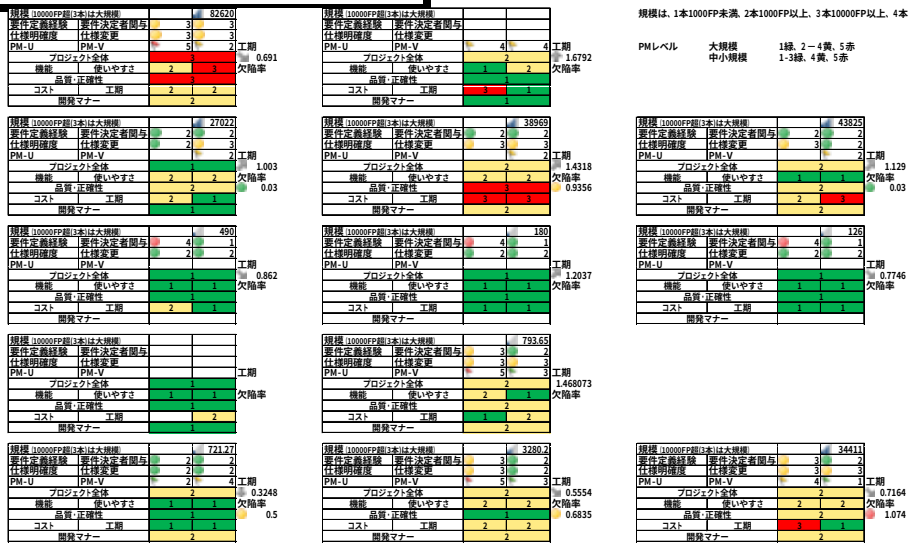
◆ 品質基準があるものは、やはり品質がよい。



参考：満足度を意識する

- 品質と正確性はまだ満足度向上の余地があり、それが、プロジェクト全体の満足度向上にも貢献できるものと考えられる。
- 意識して取り組んでいく必要がある

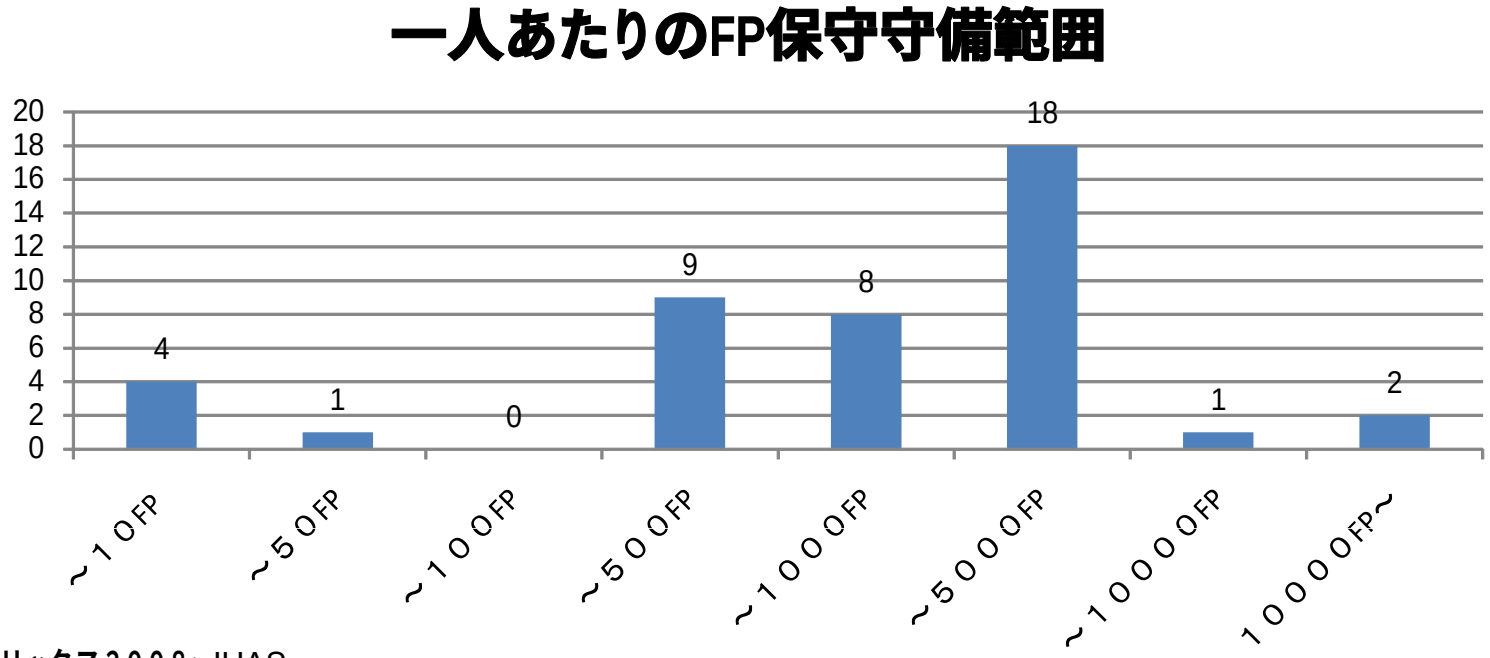
プロジェクト全体		63%	
機能	使いやすさ	75%	69%
品質・正確性		57%	
コスト	工期	51%	64%
開発マナー		62%	



立ち上げの段階で適正な保守計画を行う

- データは少ないが数百から数千FPに一人の割合で要員を配置している。

平均　　3652.4FP



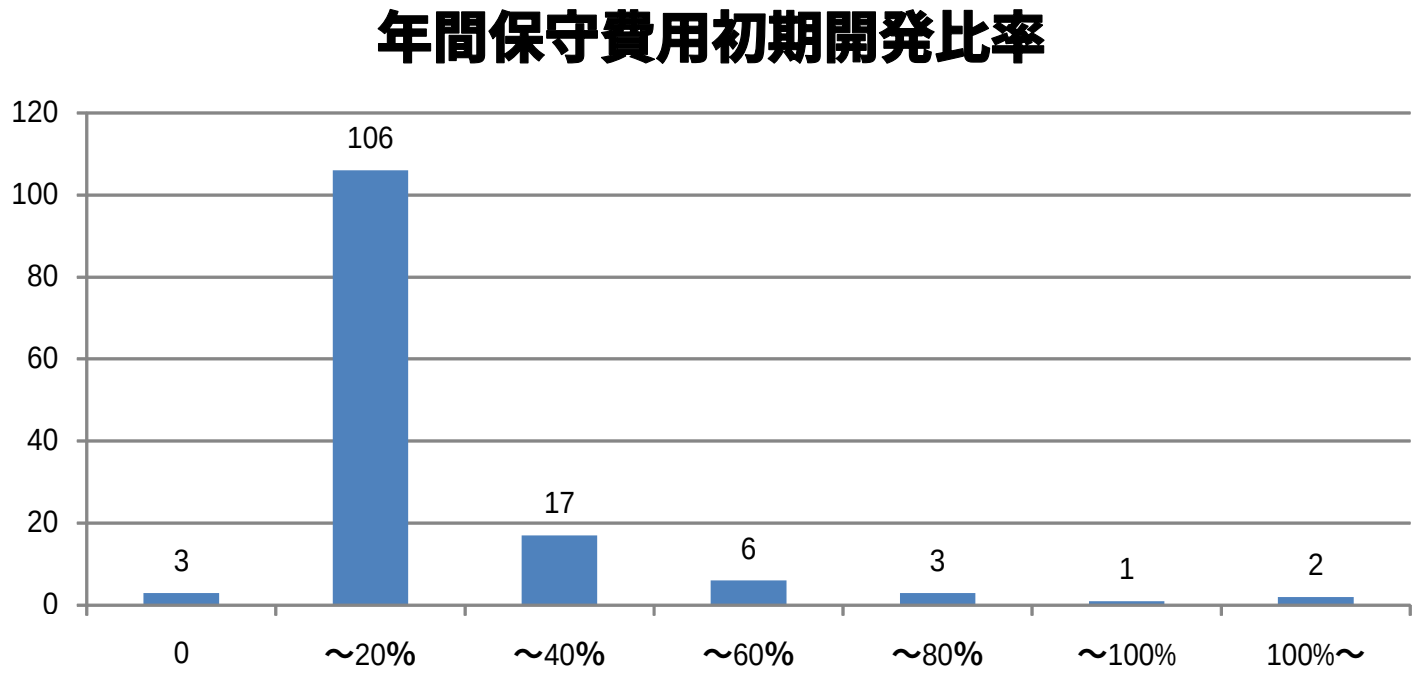
「ソフトウェアメトリックス2008」JUAS

ライフサイクルコストとして適正化検証をしていく

初期費用に対する保守費用

◆ 年間保守費用は、初期開発費用の１６．６%程度である。

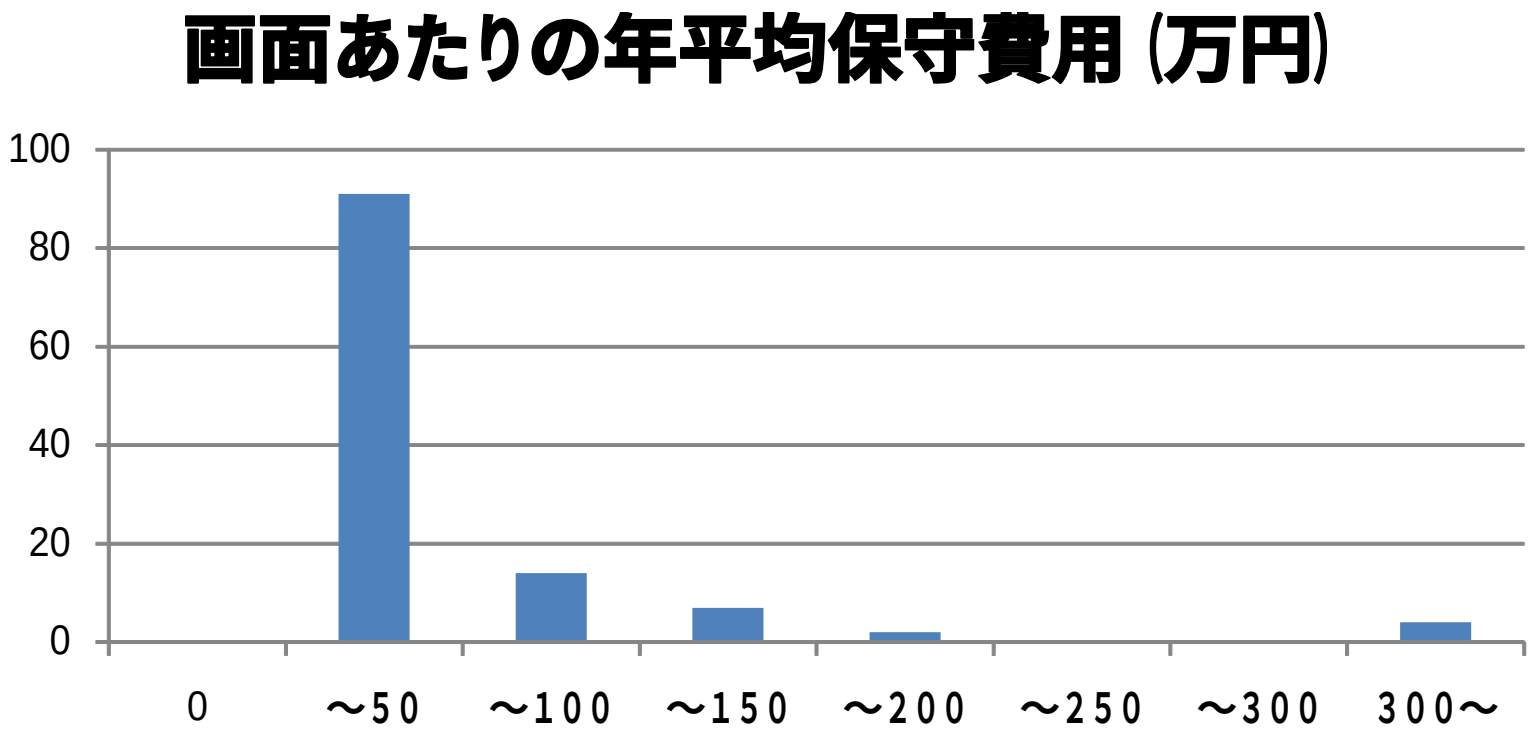
平均　　16.60%



「ソフトウェアメトリックス２００８」JUAS

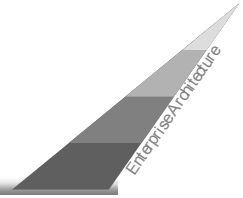
画面あたりの保守費用

平均　102.1万円

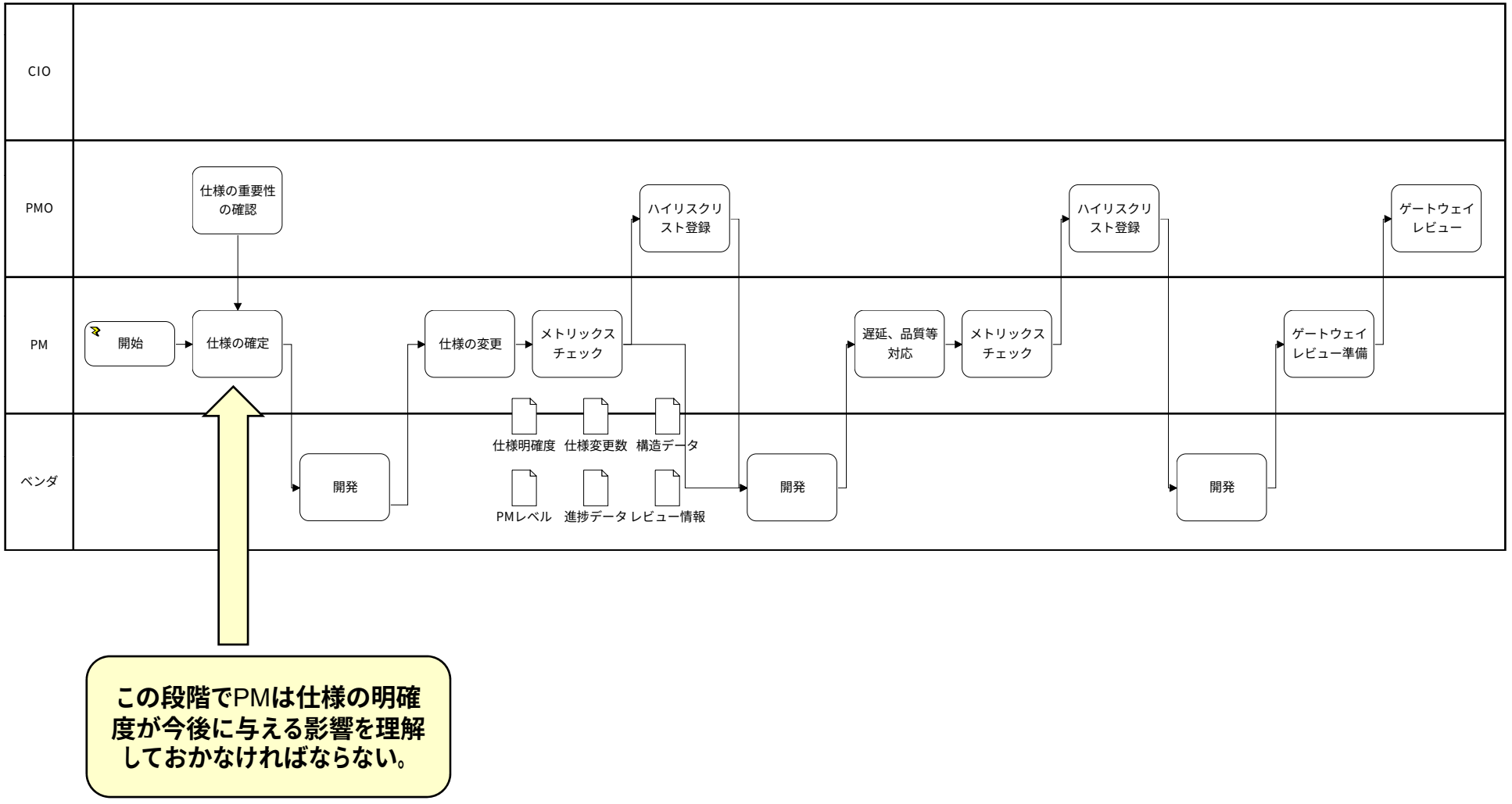


「ソフトウェアメトリックス２００８」JUAS

流れを制す



流れのプロセス


CIO
PMO
PM
ベンダ
仕様の重要性の確認
開始
仕様の確定
開発
仕様の変更
メトリックスチェック
ハイリスクリスト登録
仕様明確度
仕様変更数
構造データ
PMレベル
進捗データ
レビュー情報
開発
遅延、品質等対応
メトリックスチェック
ハイリスクリスト登録
開発
ゲートウェイレビュー準備
ゲートウェイレビュー
この段階でPMは仕様の明確度が今後に与える影響を理解しておかなければならない。

ベンダPMの重要性

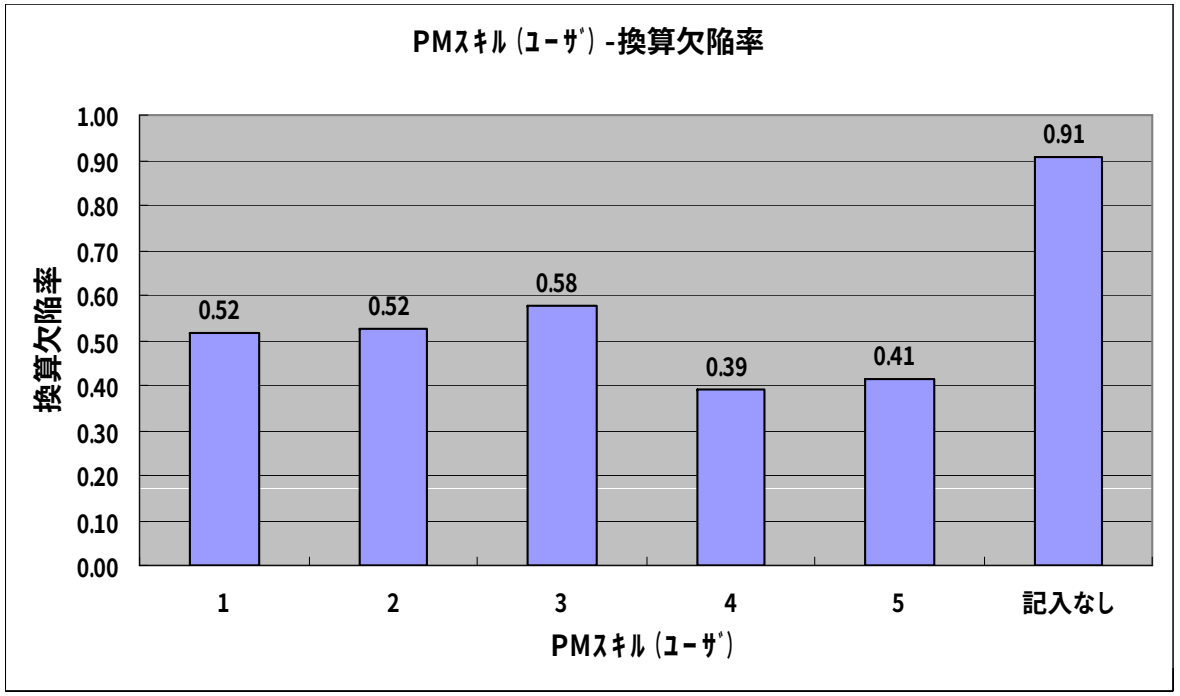
- また、PMスキルが高いほど欠陥率が低いとの傾向がある。特に未経験者がPMの場合には、途中の確認を行うなど注意が必要である。

		PMスキル(ベンダ)						
		1	2	3	4	5	記入なし	計
換算欠陥率	件数	54	35	54	30	5	30	208
	平均	0.27	0.49	0.7	0.42	1.54	0.82	0.55
	最大	1.69	2.95	9.06	1.83	4.38	11.89	11.89
	最小	0.00	0.00	0.00	0.00	0.02	0.00	0.00

PMスキル
1.多数の中･大規模プロジェクトの管理を経験
2.少数の中･大規模プロジェクトの管理を経験
3.多数の小･中規模プロジェクトの管理を経験
4.少数の小･中規模プロジェクトの管理を経験
5.プロジェクト管理の経験なし

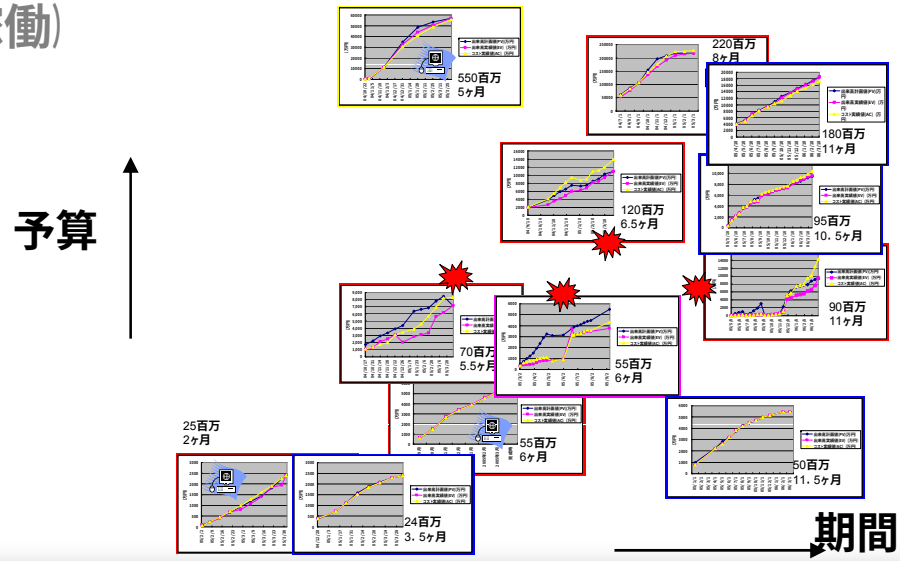
ユーザPMの重要性

◆ ユーザPMのスキルと品質には相関は見られない。

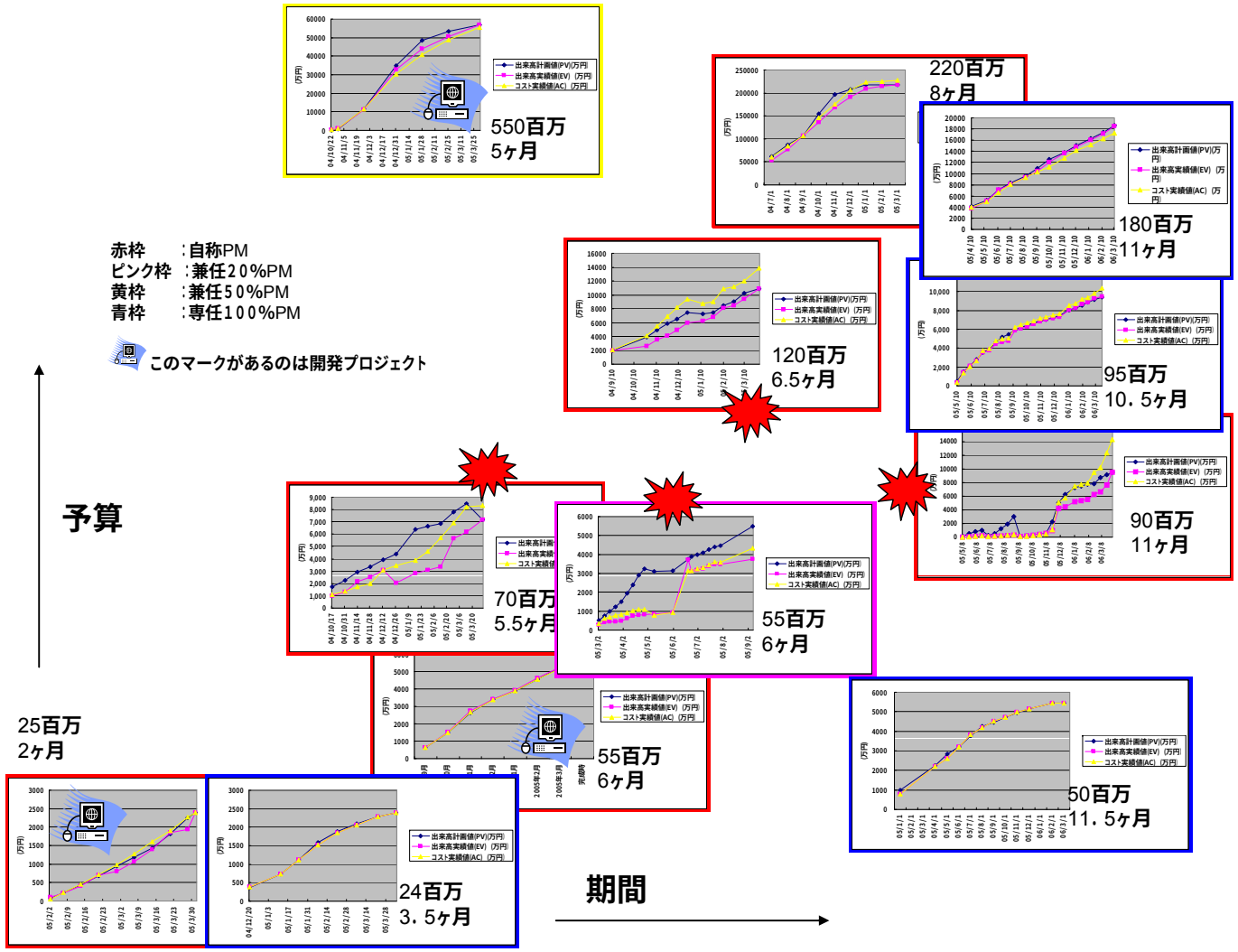


PMの資格をどう考えたらよいか

- 情報処理技術者やＰＭＰ（Project Management Professional）などのプロジェクト・マネジメント専門技術者の有無とプロジェクト管理手法であるＥＶＭの結果を比較することにより、認定された技術者を導入した場合と導入しない場合の品質の差異などを分析したものがある。下図は、右に行くほど実施期間が長く、上に行くほど開発金額が大きくなるようにEVMのデータをプロットしたものである。（正常なプロジェクトは右肩上がりにグラフが上がっていき、計画と実績があまりずれない）このグラフには業務分析プロジェクトと開発プロジェクトが含まれるが、開発プロジェクトはグラフ中にコンピュータマークを付けているが失敗が少ない傾向がある。それに対し赤で囲まれたプロジェクトは、プロジェクトマネージャの資格は持っていないが自称プロジェクトマネージャが実施しているプロジェクトであるが、多くのプロジェクトが失敗に終わっている。逆に資格者が配置されている青いプロジェクトでは失敗プロジェクトは存在していない。（黄色は資格を持ったPMが50%以上稼働、ピンク色は資格を持ったPMが20%以上稼働）

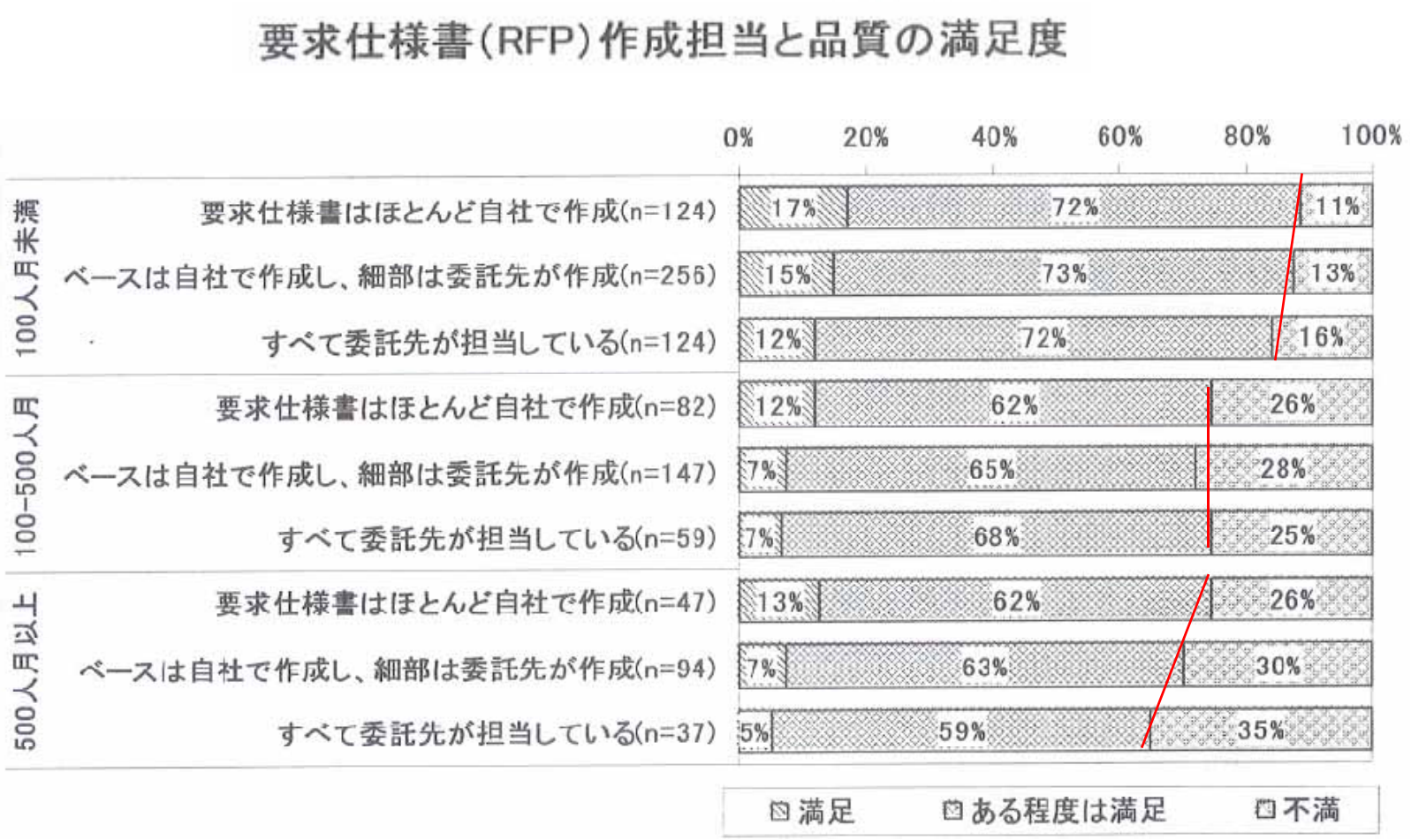


参考：拡大


赤枠　：自称PM
ピンク枠　：兼任２０％PM
黄枠　：兼任５０％PM
青枠　：専任１００％PM
このマークがあるのは開発プロジェクト
予算
期間
550百万
5ヶ月
220百万
8ヶ月
180百万
11ヶ月
120百万
6.5ヶ月
95百万
10．5ヶ月
90百万
11ヶ月
70百万
5.5ヶ月
55百万
6ヶ月
55百万
6ヶ月
50百万
11．5ヶ月
25百万
2ヶ月
24百万
3．5ヶ月

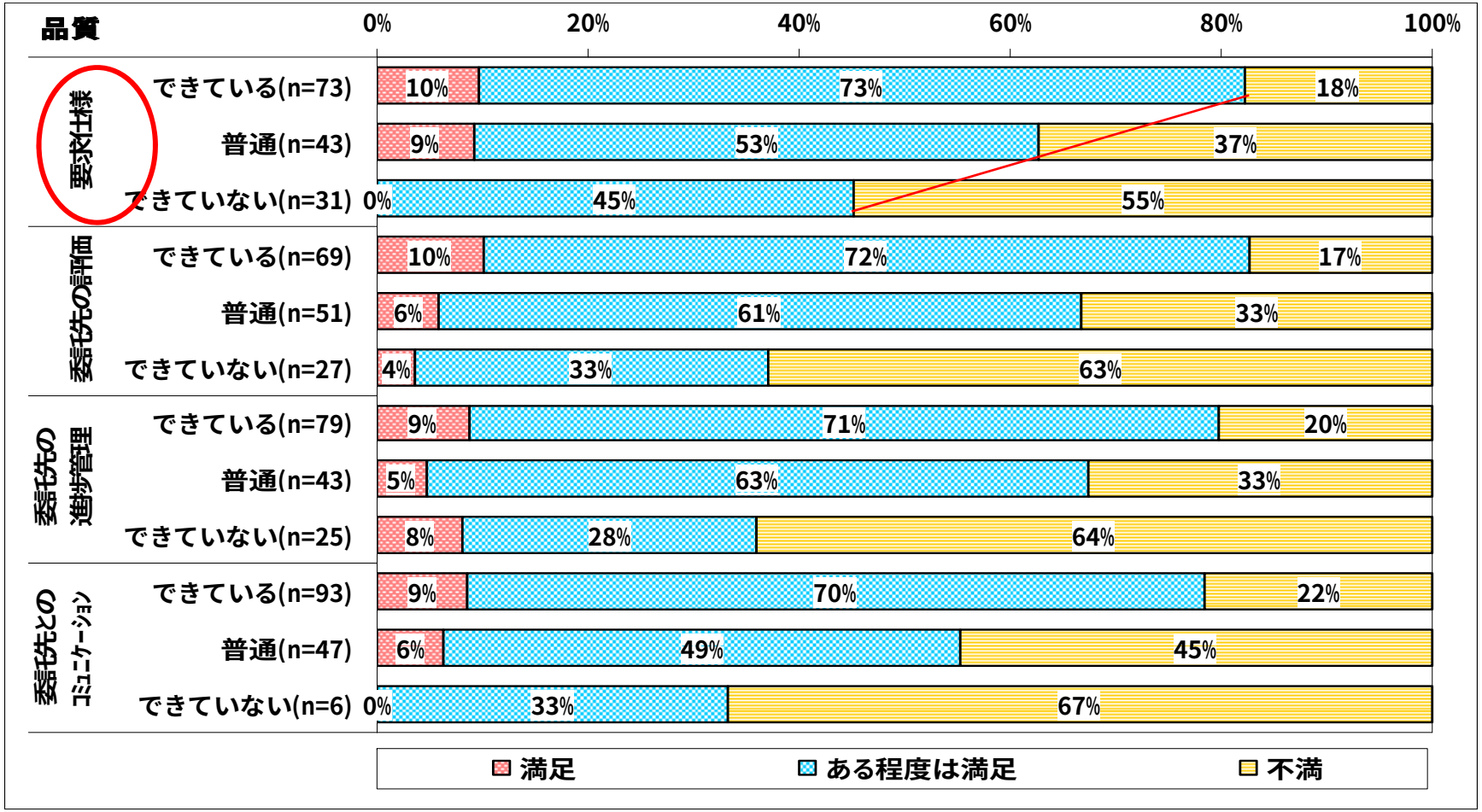
要求仕様書は誰が作る？

◆ 自分が作ったほうが品質に対する満足度が高い



プロジェクトの失敗の原因はＲＦＰと仕様書にある

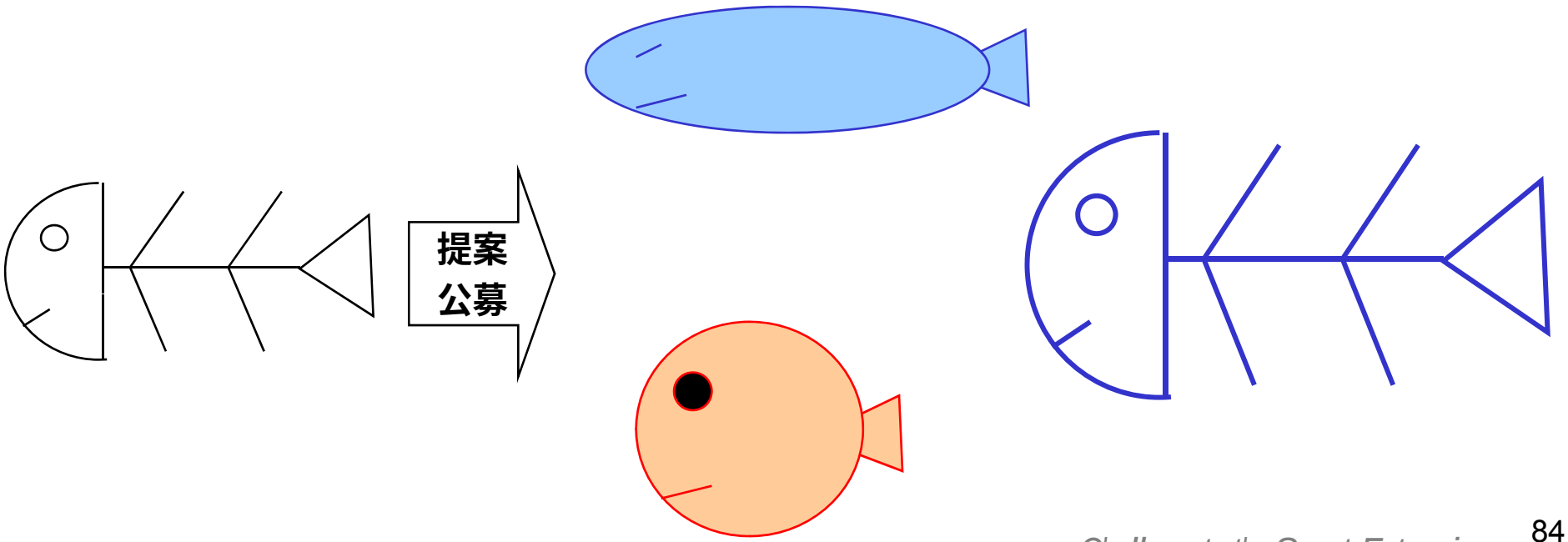
◆ 発注者としての意識と品質の関係



JUAS　企業IT動向調査２００８

要件定義におけるトラブルパターン

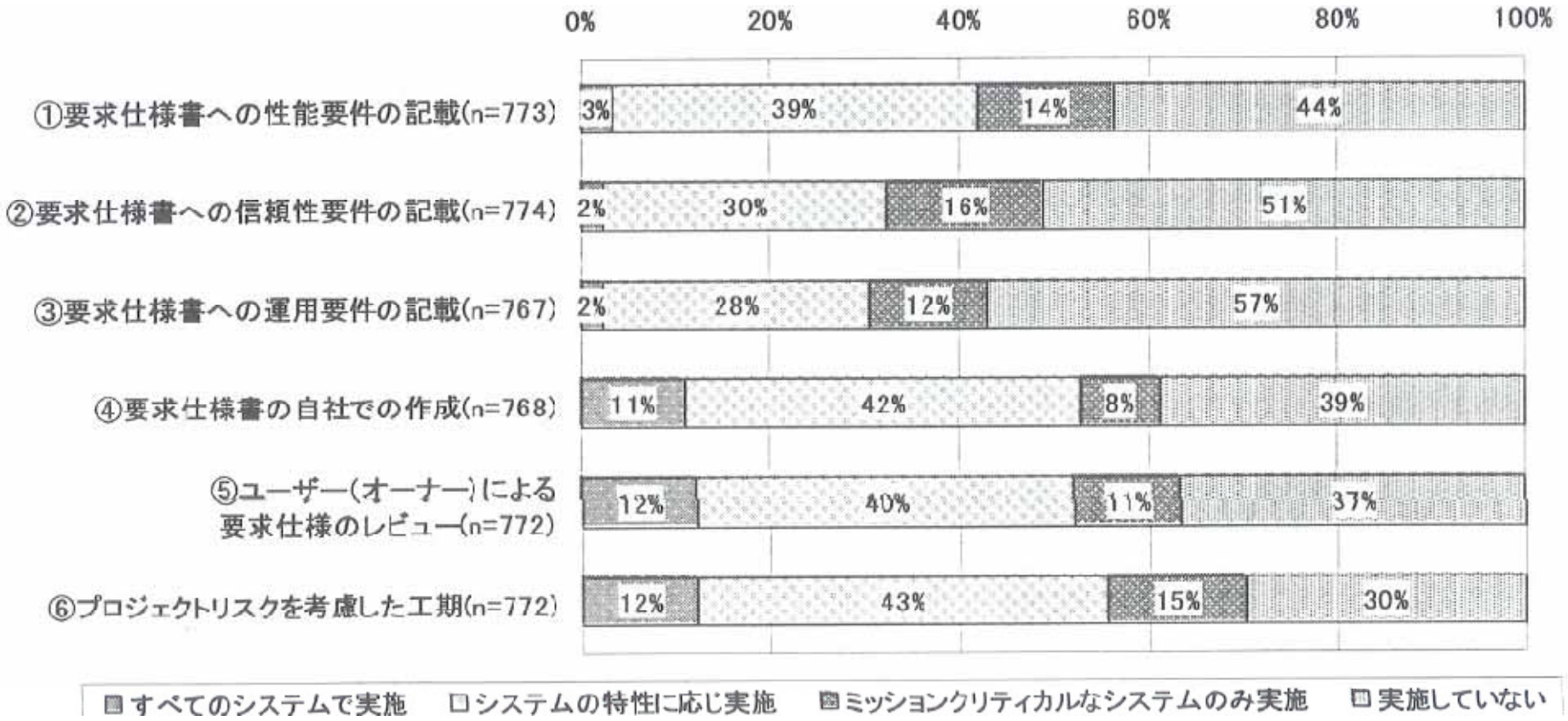
- 不十分なＲＦＰ（要求仕様書）
- 仕様外の要求
- 検収時点での要求不一致、サービスレベル不足
- ミニマム仕様の提供



いったいどこまでやっているのか

◆ 意外にやっていない

信頼性向上のためのシステム企画・要求仕様作成段階での施策



「ソフトウェアメトリックス2007」JUAS

遅延の要因

- システムの開発プロジェクトでは、納期遅れなどのスケジュールの遅延に関する報告をよく聞く。実際に、日本情報システムユーザ協会の「ソフトウェアメトリックス2007」では、７０％以上のシステムが工期内に開発を終わらせているが、１３％のシステムが２０％以上の工期遅延を起こしている。このような工期遅延の可能性について留意が必要である。
- このような工期遅延の原因は、主に以下のものに起因している。
  - 要件仕様の決定遅れ　　２０．９％
  - 要件分析作業不十分　　１３．３％
  - 開発規模の増大　　１３．７％
- このような工期遅延により、標準工期で開発していたシステムが短工期システムになってしまう恐れもある
  - 結果として品質に与えることとなる
- リリース時期に厳密な制約のあるシステムでは、上記の遅延原因が起こらないようなプロジェクト管理が求められる。

仕様の明確度と満足度と遅延

- 情報システムの調達において、仕様が曖昧なことがよく指摘されるが、仕様は明確に定義することが重要である。下表のように、仕様が曖昧であると工期遅延を発生することが多くなり、満足度も低くなることが多い。このような傾向を念頭に置き、仕様作成時には仕様の明確さを、プロジェクトメンバ、リーダ、プログラムマネジメントオフィスの各段階で十分に確認する必要がある。

仕様明確度		顧客満足度(プロジェクト全体)				
		満足	やや不満	不満	未回答	総計
非常に明確	件数	23	1		1	25
	割合	92.0%	4.0%	0.0%	4.0%	100.0%
かなり明確	件数	120	40	6	8	174
	割合	69.0%	23.0%	3.4%	4.6%	100.0%
やや曖昧	件数	59	47	8	4	118
	割合	50.0%	39.8%	6.5%	3.4%	100.0%
非常に曖昧	件数	6	5		1	12
	割合	50.0%	41.7%	0.0%	8.8%	100.0%
合計	件数	208	93	14	14	329
	割合	63.2%	28.3%	4.3%	4.3%	100.0%

仕様明確度		工期遅延度							遅延度20%以上の割合
		予定より早い	予定通り	10%未満	20%未満	50%未満	それ以上	総計	
非常に明確	件数		20	2	1	2		25	8.0%
	割合	0.0%	80.0%	8.0%	4.0%	8.0%	0.0%	100.0%	
	平均工期遅延率		0.0%	6.9%	11.1%	33.9%		3.7%	
かなり明確	件数	12	109	10	13	10	2	156	7.7%
	割合	7.7%	69.9%	6.4%	8.3%	6.4%	1.3%	100.0%	
	平均工期遅延率	-30.86%	0.00%	6.45%	14.71%	29.01%	61.11%	1.91%	
やや曖昧	件数	8	67	8	8	14	8	113	19.5%
	割合	7.1%	59.3%	7.1%	7.1%	12.4%	7.1%	100.0%	
	平均工期遅延率	-29.9%	0.00%	6.65%	14.79%	28.60%	64.06%	7.48%	
非常に曖昧	件数		7			4	1	12	41.7%
	割合	0.0%	58.3%	0.0%	0.0%	33.3%	8.3%	100.0%	
	平均工期遅延率		0.00%			36.21%	50%	16.24%	
合計	件数	20	203	20	22	30	11	306	13.4%
	割合	6.5%	66.3%	6.5%	7.2%	9.8%	3.6%	100.0%	
	平均工期遅延率	-30.46%	0.00%	6.58%	14.58%	30.10%	62.25%	4.68%	

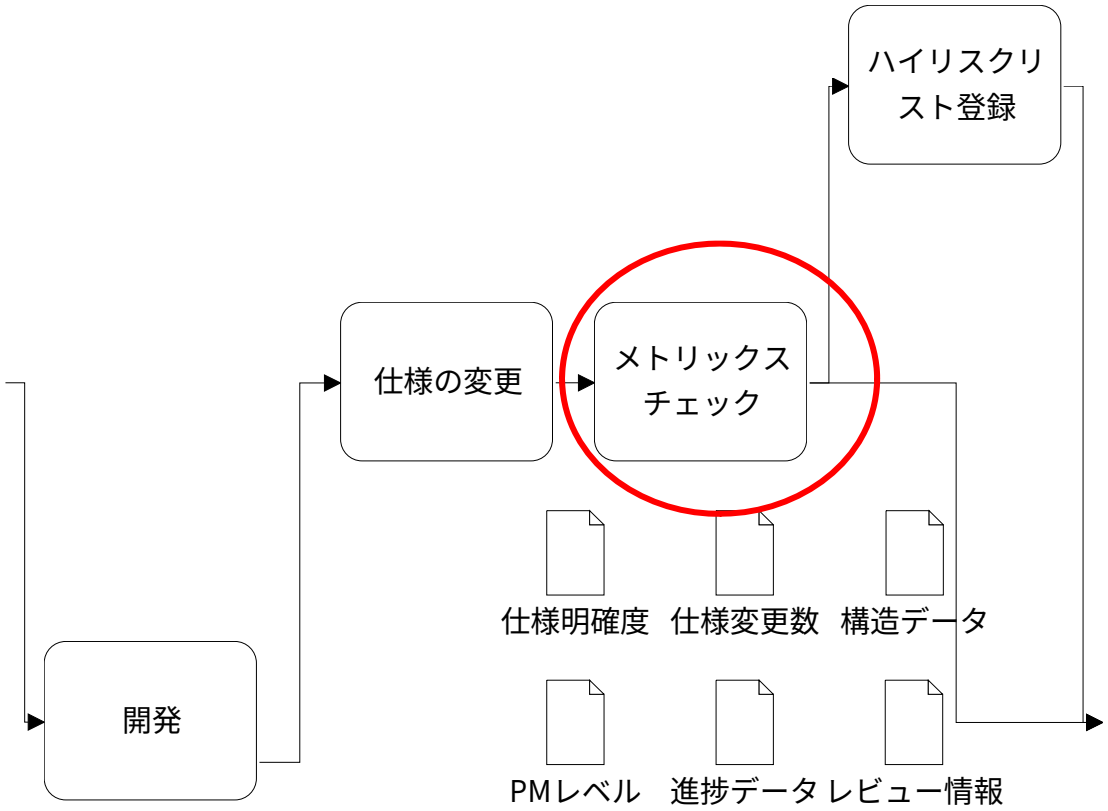
「ソフトウェアメトリックス２００８」JUAS

仕様の明確度と欠陥率

- 当然のことながら仕様が明確でないものは，欠陥率も大きくなる傾向がある。

仕様明確度	件数	平均換算欠陥率	最大欠陥率
非常に明確	17	0.27	1.66
かなり明確	103	0.44	5.37
ややあいまい	80	0.75	11.89
非常にあいまい	6	0.63	2.15
合計	206	0.55	11.89

仕様変更による影響



仕様変更が起こると、開発工数、機能数などに変化が起こる。

変更影響日数が多いと工期不足によるハイリスクプロジェクトになっている可能性もある。メトリックスによる検証と必要に応じてハイリスクリストへの登録をしていく必要がある

仕様変更と満足度と遅延

- 更に仕様変更であるが、仕様変更が大きなものは遅延も生じやすく満足度にも問題が生じがちであるので、このような傾向を念頭に置き、仕様作成時に仕様内容を、プロジェクトメンバ、リーダ、プログラムマネジメントオフィスの各段階で十分に確認する必要がある。

仕様変更発生度		ユーザ満足度（プロジェクト全体）				
		満足	やや不満	不満	未回答	総計
変更なし	件数	11	5			16
	割合	68.8%	31.3%	0.0%	0.0%	100.0%
軽微な変更が発生	件数	154	54	4	11	223
	割合	69.1%	24.2%	1.8%	4.9%	100.0%
大きな変更が発生	件数	42	30	10	3	85
	割合	49.4%	35.3%	11.8%	3.5%	100.0%
重大な変更が発生	件数	1	2		1	4
	割合	25.0%	50.0%	0.0%	25.0%	100.0%
合計	件数	208	91	14	15	328
	割合	63.4%	27.7%	4.3%	4.6%	100.0%

仕様変更発生度		遅延度							遅延度20%以上の割合
		予定より早い	予定通り	10%未満	20%未満	50%未満	それ以上	総計	
変更なし	件数	3	8		1	2		14	14.3%
	割合	21.43%	57.1%	0.0%	7.1%	14.3%	0.0%	100.0%	
	平均工期遅延率	-34.1%	0.0%		18.4%	32.2%		-1.4%	
軽微な変更が発生	件数	12	148	11	14	16	6	207	10.5%
	割合	5.80%	71.5%	5.3%	6.8%	7.7%	2.9%	100.0%	
	平均工期遅延率	-34.18%	0.00%	7.09%	13.64%	30.81%	68.75%	3.69%	
大きな変更が発生	件数	5	45	9	6	10	5	80	18.8%
	割合	6.25%	56.3%	11.3%	7.5%	12.5%	6.3%	100.0%	
	平均工期遅延率	-19.3%	0.00%	5.94%	16.71%	27.77%	54.44%	7.59%	
重大な変更が発生	件数		1			3		4	75.0%
	割合	0.00%	25.0%	0.0%	0.0%	75.0%	0.0%	100.0%	
	平均工期遅延率		0.00%			30.69%		23.01%	
合計	件数	20	202	20	21	31	11	305	13.8%
	割合	8.2%	63.9%	7.4%	5.7%	11.5%	3.3%	100.0%	
	平均工期遅延率	-30.46%	0.00%	6.58%	14.74%	29.91%	62.25%	4.73%	

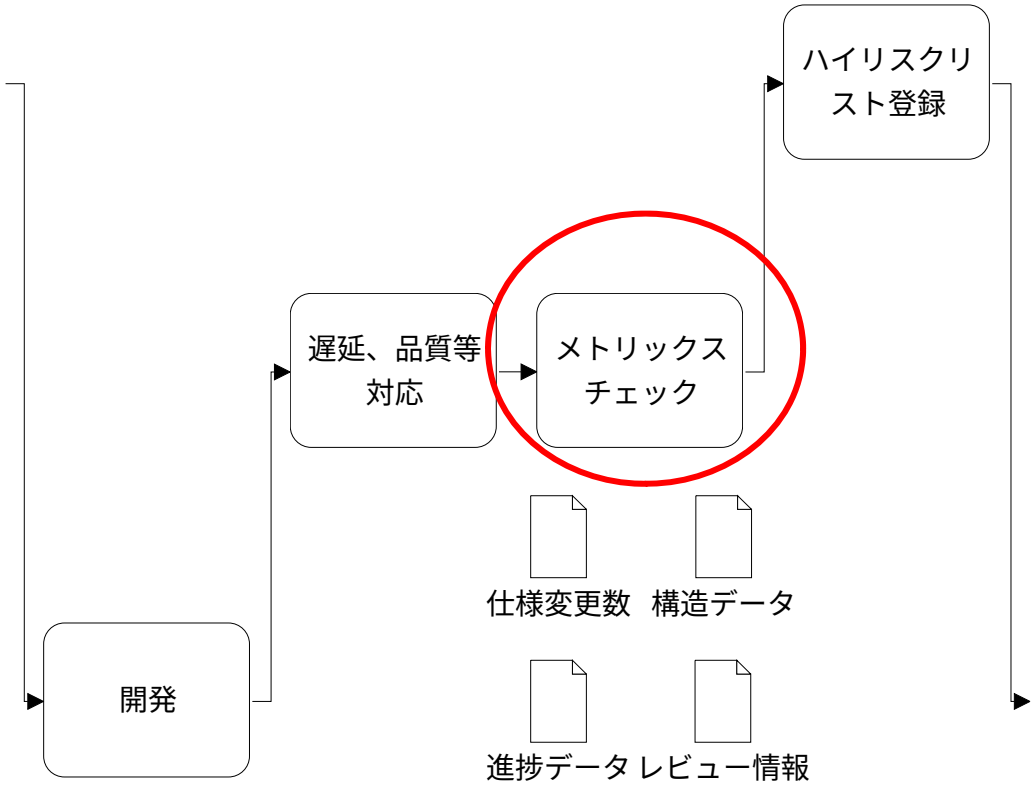
「ソフトウェアメトリックス２００８」JUAS

仕様変更と欠陥率

- 当然のことながら仕様変更が発生したものは変更しないものに対して作業にゆがみが生じることから欠陥率も大きくなる傾向がある。

仕様変更発生度	件数	平均換算欠陥率	最大欠陥率
変更なし	8	0.36	0.73
軽微な変更が発生	140	0.57	11.89
大きな変更が発生	56	0.54	4.38
重大な変更が発生	1	0.03	0.03
合計	205	0.81	11.89

進捗による影響

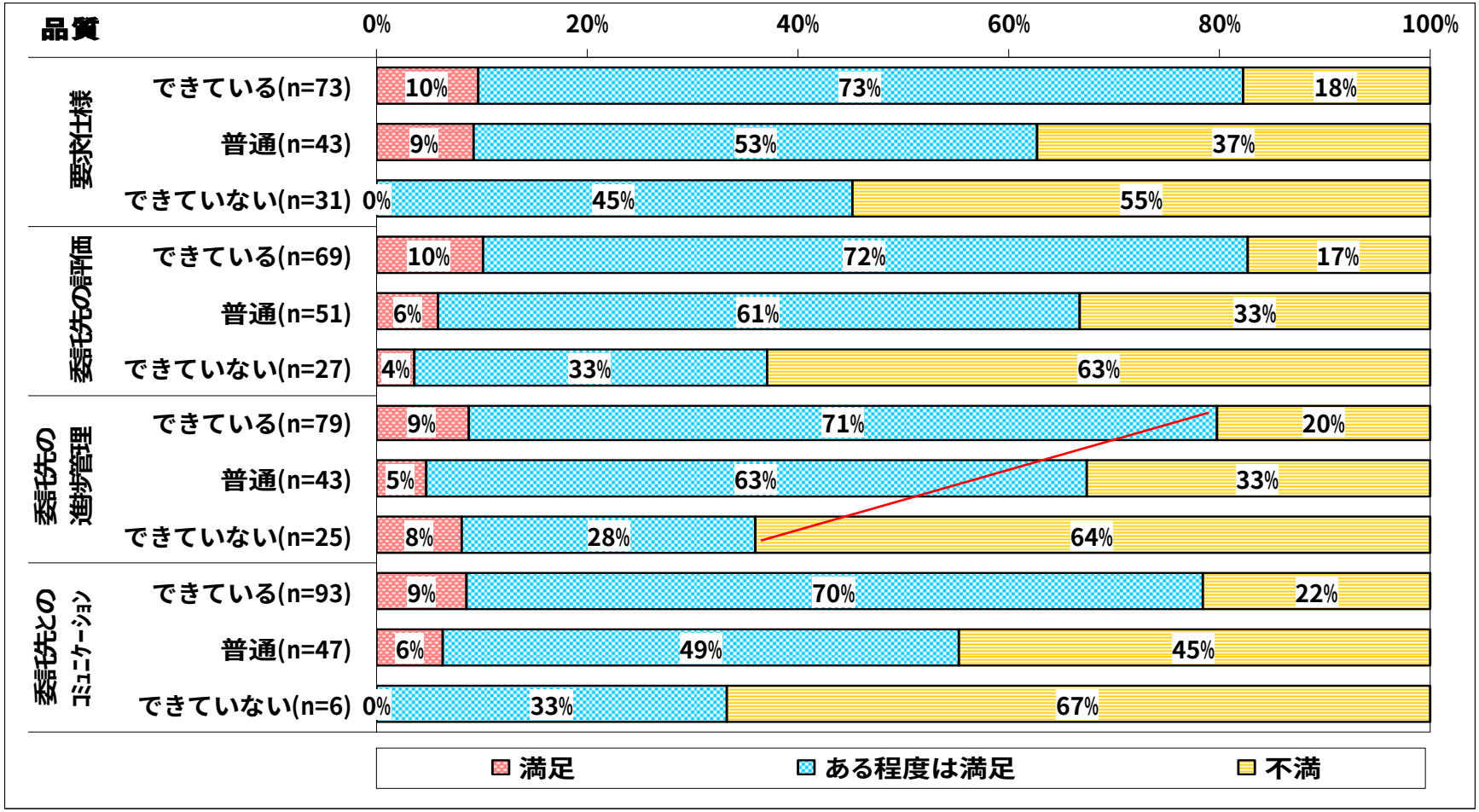


進捗情報では、特に遅延に関する注意が必要である。計画では適正配分されていたものが短工期になっている可能性がある

納期に間に合わないからと人材を投入しても管理上問題があることは明確であり、リリース延期なども考えなければならない

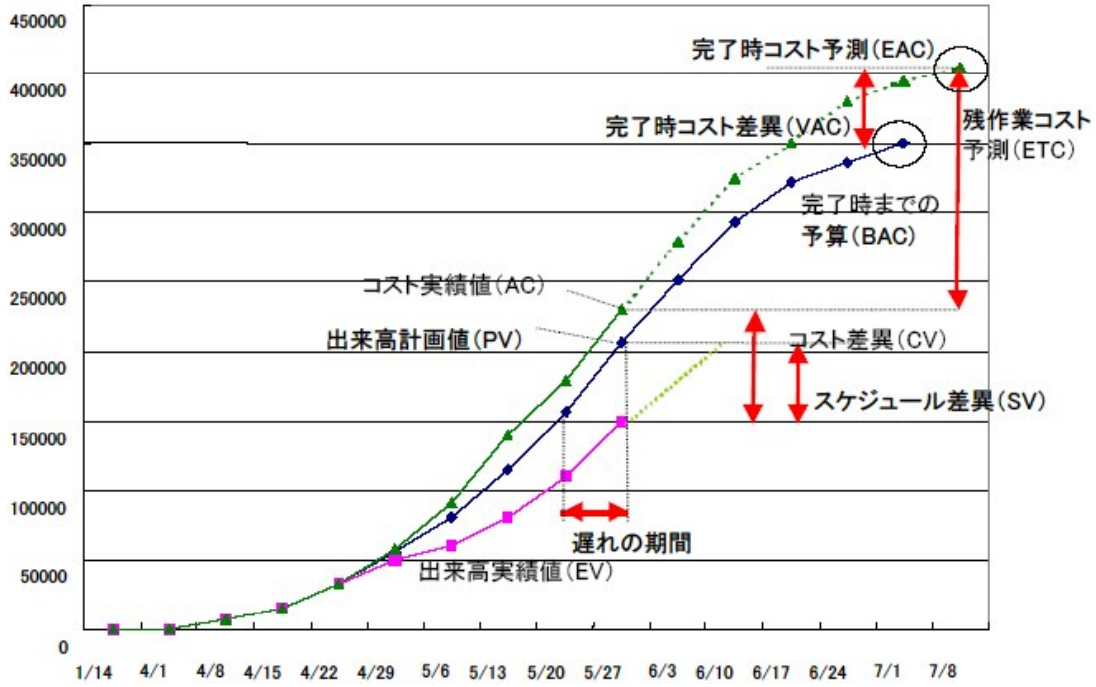
進捗の管理は品質に影響するのか

◆ 進捗の管理ができていないシステムは半分以上が不満な結果になっている



進捗管理の手法

- ＥＶＭでは、プロジェクトの進捗を価値に換算して表す出来高計画値（ＰＶ）を軸に計画策定が行われる。プロジェクトのＷＢＳ（Work Breakdown Structure）を作成し、ＷＢＳ項目毎にスケジュール予定と投入予定工数を基に作成した出来高計画値を設定する。
- プロジェクト開始後は、計画値と実績値をプロットしたグラフによって、計画値に対する進捗状況と工数投入状況を一元的に把握することが可能である。



EVMの管理シートと複数プロジェクトの見え方

◆ EVMは思ったよりも難しくない

事業名：　　※別様式で実績値を管理している場合には、表計算形式などのファイルで提出いただければ転記する必要は

進捗実績表（EVM）

	4月1日										
出来高計画値(PV)(万円)											
出来高実績値(EV)（万円）											
コスト実績値(AC)（万円）											
スケジュール差異(SV)（万円）	0	0	0	0	0	0	0	0	0		
コスト差異（CV)（万円）	0	0	0	0	0	0	0	0	0		
スケジュール効率指数（SPI）	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!		
コスト効率指数（CPI）	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!		
完了時コスト予測（EAC）（万円）	#DIV/0!	#DIV/0!									

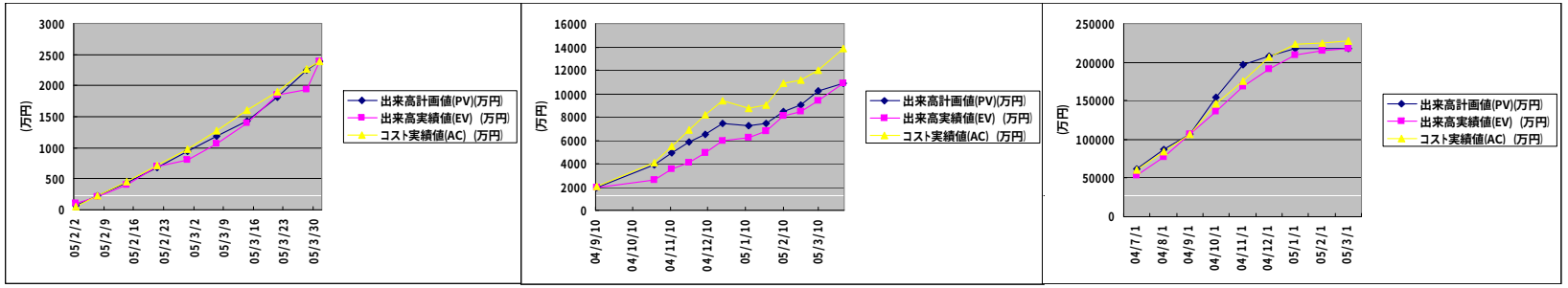
EAC=AC+(BAC-EV)/CPI　　左の式の使っていない方の式を消去
EAC=AC+(BAC-EV)/（CPI*SPI)

品質・仕様変更実績表

	4月1日										
累積不具合件数											
未解決不具合数											
平均障害滞留時間（時間）											
仕様変更数											
仕様変更延日数（日）											

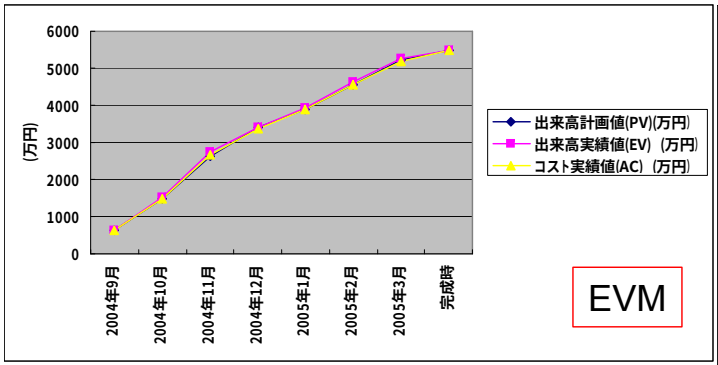
仕様変更延日数とは、仕様変更で追加される作業量（見込みでも可）の総和から、仕様変更で計画より削減される作業量の総和とする

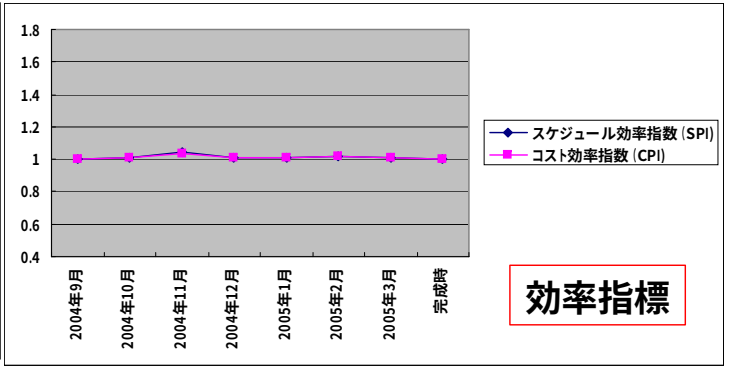
◆ 複数プロジェクトを同じ視点で管理することができる

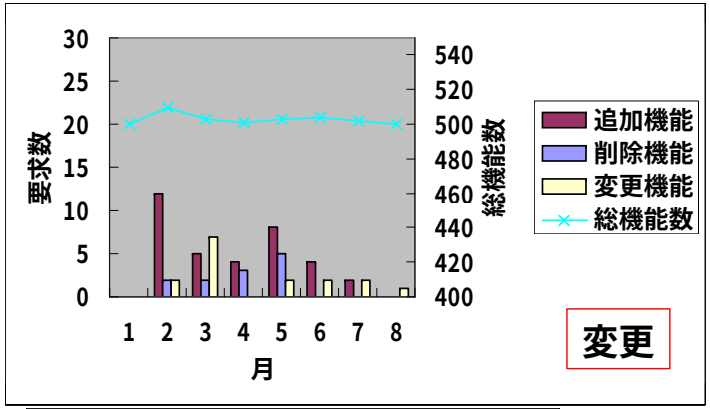


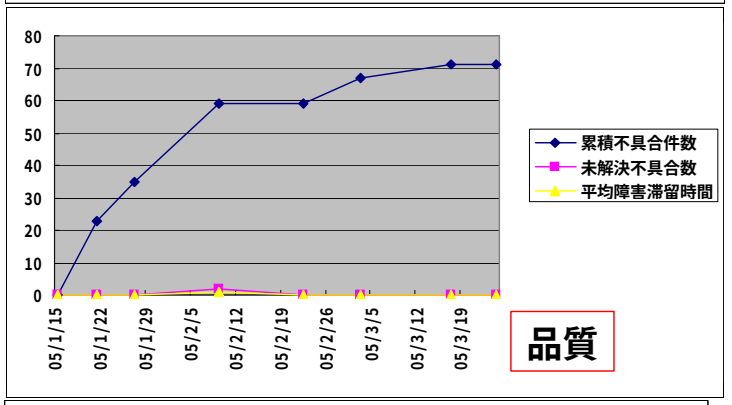
プロジェクトモニタデータの統合管理

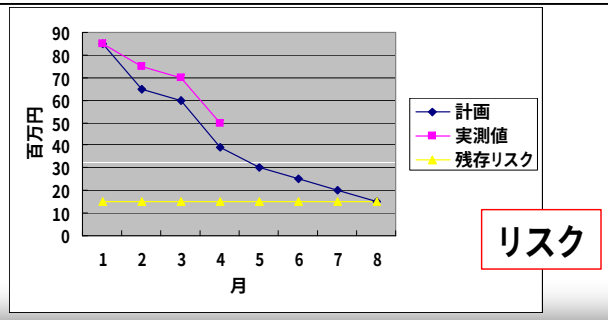
◆ 詳細データを聞かなくても、状況把握が簡単になる

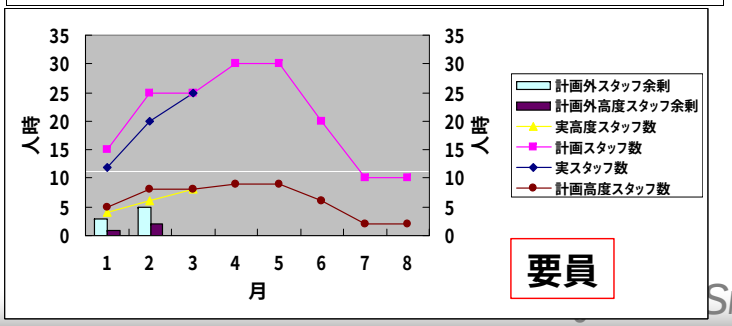






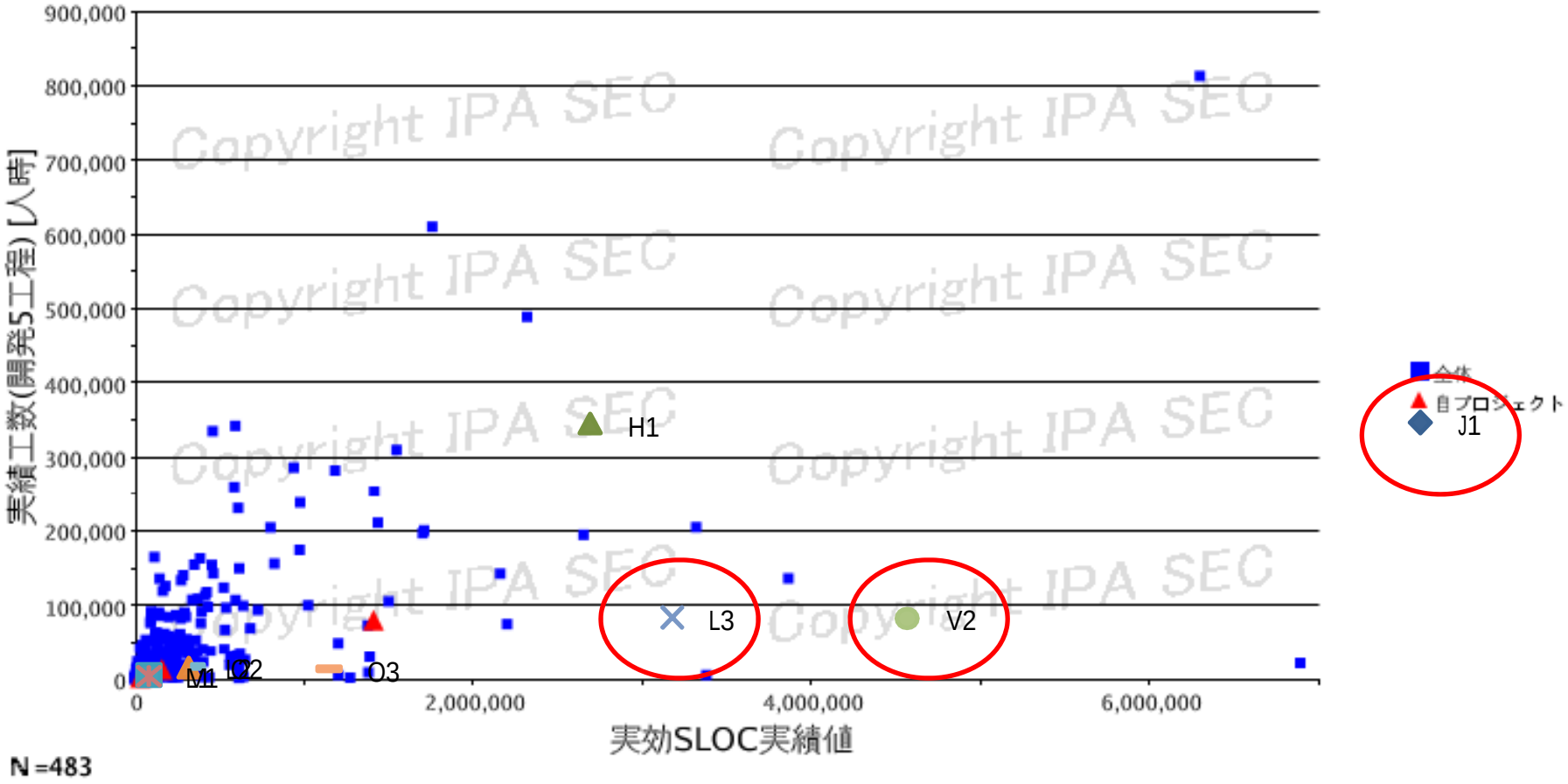






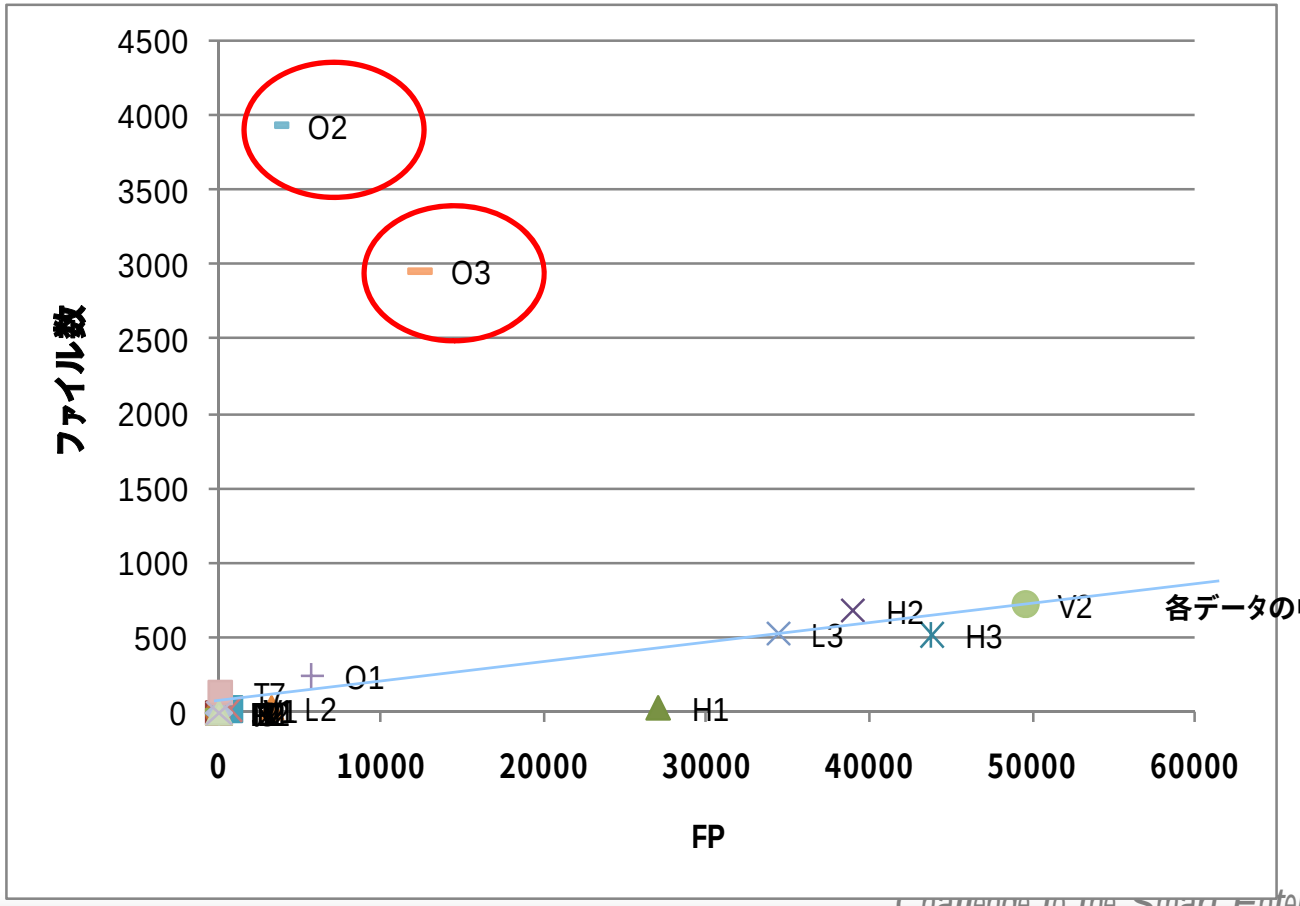
SLOC規模と工数

- 工数を使わずに大量のコードを生産しているシステムが発見できる



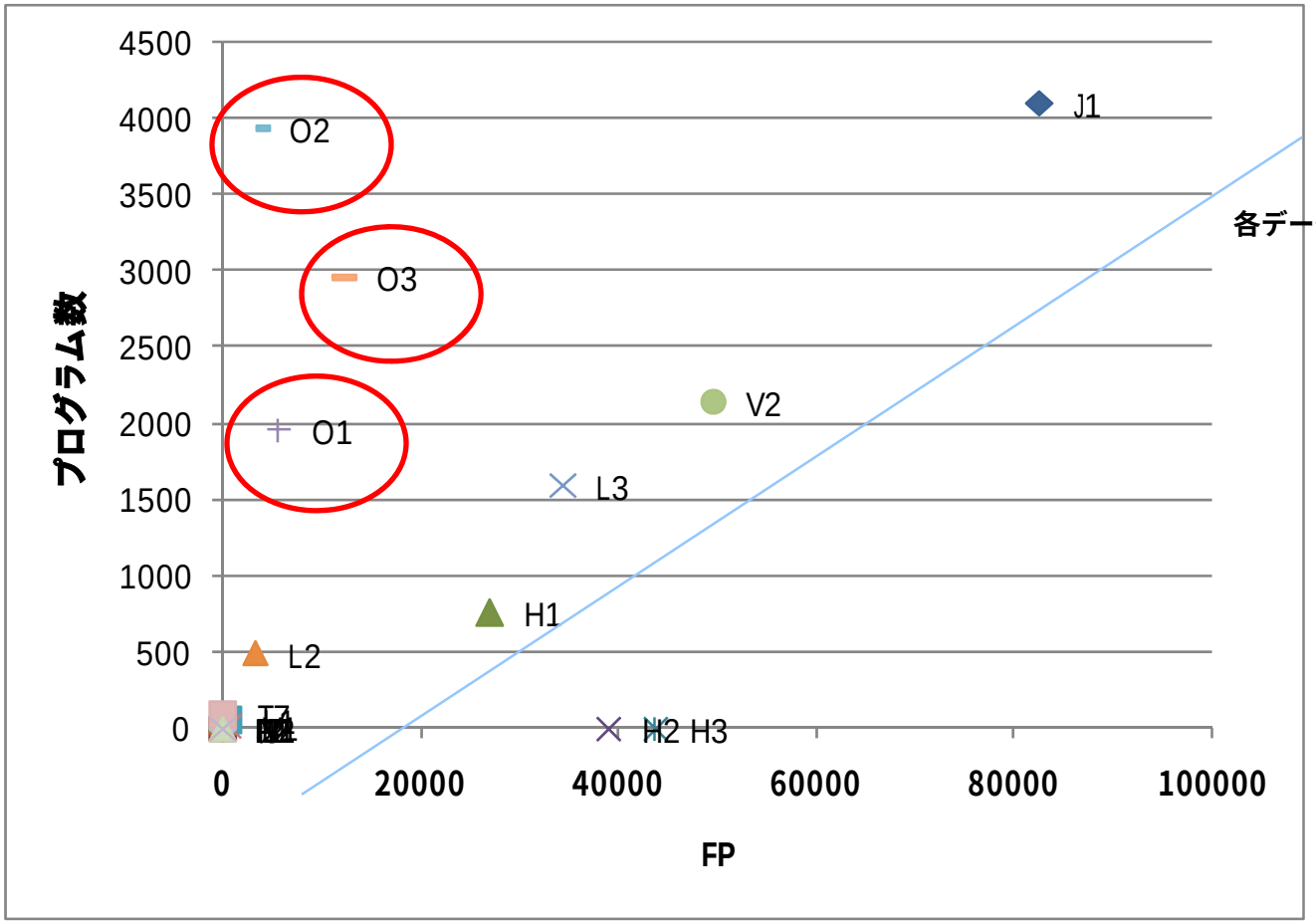
規模あたりのファイル数

- ファイル数が異常に多いシステムが発見できる。
- このようなＰＪは、作りに問題があるのではないか。
  - 保守性への影響も想定される。



規模あたりのプログラム数

- ファイル同様に、プログラムの分割が細かいレベルで行われているものも発見できる。

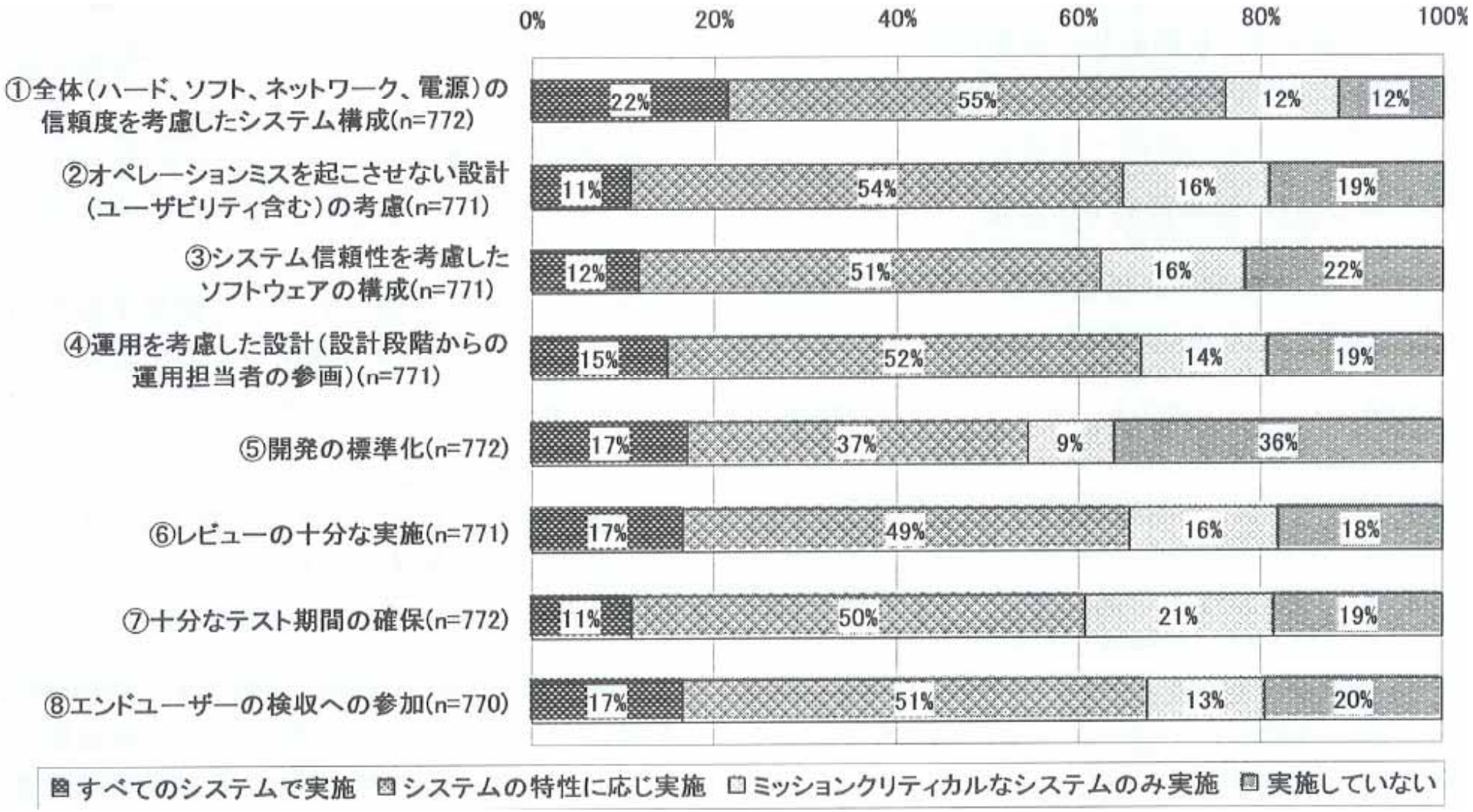


ブランドベンダは安心か

- ちなみに、安心料として単金の高いベンダもしくは要員に発注することも多いが、人月単価と品質の関係には相関が見られない。つまりは、単価よりも上記PMスキルなど、個人の能力をベースに要員を配置したほうがより有効と考えられる。
  - ただし難易度による評価が入っていないので留意が必要である

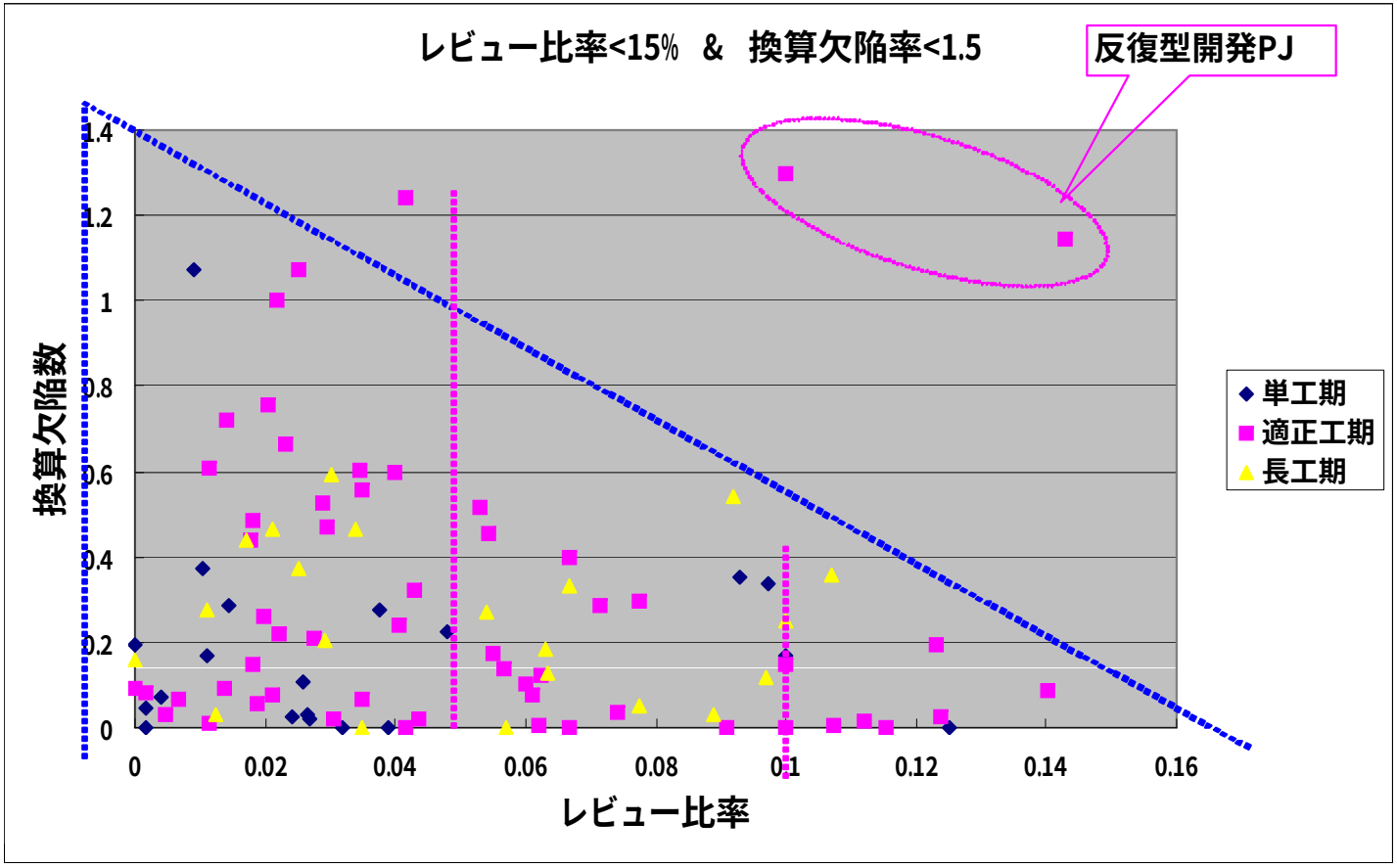
	Aランク	Bランク	Cランク	Dランク	Eランク	Fランク	総計
欠陥率	0	0.25未満	0.5未満	1未満	3未満	3以上	
件数	8	76	38	18	16	6	162
単価（平均）[万円]	90.2	106.2	104.3	100.6	116.7	107.8	
単価（最大）[万円]	117.5	272.7	236.9	162.8	285.7	250.0	
単価（最小）[万円]	71.5	46.2	43.2	41.4	70.8	45.7	

信頼性向上のためのシステム設計・開発段階での施策



レビュー

◆ 当然のことながら、レビューをすれば欠陥は少なくなる。



正しいレビューの実施

- レビューは役割分担やチェックリストなどを用意した上で計画的に実施することが重要である。ただし、現場に過剰な負荷をかけてはいけない。
  - アドホックレビュー
    - 作業途中で必要に応じて実施するレビュー
  - パスアラウンドレビュー
    - メール配布などによるレビュー依頼
  - ペアレビュー
    - 作成者とレビューアが成果を確認
  - ウォークスルー
    - 業務の流れに沿って作成者が説明を行うレビュー
  - チームレビュー
    - 計画された成果物レビューなど
  - インスペクション
    - モデレータ、記録者など役割を指定して行う査察的、厳格なレビュー

重点レビュー項目

- ISO9126にも規定されるソフトウェア品質特性に基づきレビューを行っていく
  - 機能性
    - 合目的性、正確性、接続性、整合性、セキュリティ
  - 信頼性
    - 成熟性、障害許容性、回復性
  - 使用性
    - 理解性、習得性、操作性
  - 効率性
    - 実行効率性、資源効率性
  - 保守性
    - 解析性、変更作業性、安定性、試験性
  - 移植性
    - 環境適応性、移植作業性、規格準拠性、置換性

レビューにおけるカバレッジ

- CMMレベル5を持つInfosysでは以下の基準でレビューを行っている。

Infosysのレビュー能力ベースライン

レビュー項目	準備のカバレッジ率	グループレビューのカバレッジ率	表面的/軽微な欠陥の欠陥密度	致命的/重大な欠陥密度
要件		5-7page/時	0.5-1.5欠陥/page	0.1-0.3欠陥/page
ハイレベル設計		4-5page/時（または200-250仕様文/時）	0.5-1.5欠陥/page	0.1-0.3欠陥/page
詳細設計		3-4page/時（または70-100仕様文/時）	0.5-1.5欠陥/page	0.2-0.6欠陥/page
コード	160-200LOC/時	4-6page/時	0.01-0.06欠陥／LOC	0.01-0.06欠陥/page
統合テスト計画		5-7page/時	0.5-1.5欠陥/page	0.1-0.3欠陥/page
統合テストケース		3-4page/時		
システムテスト計画		5-7page/時	0.5-1.5欠陥/page	0.1-0.3欠陥/page
システムテストケース		3-4page/時		
プロジェクト管理計画および構成管理計画	4-6page/時	2-4page/時	0.6-1.8欠陥/page	0.1-0.3欠陥/page

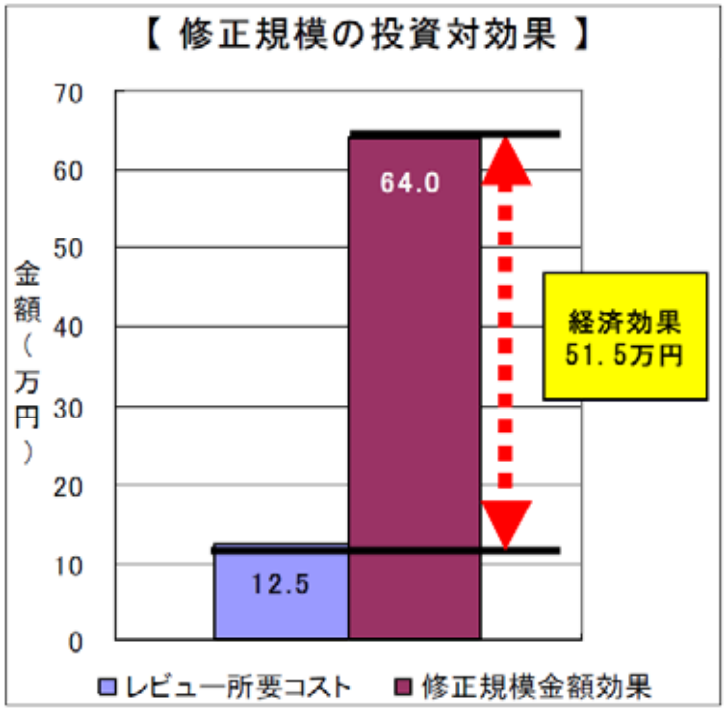
ソフトウェア開発のためのプロジェクトマネジメント入門、ソフトバンクパブリッシング

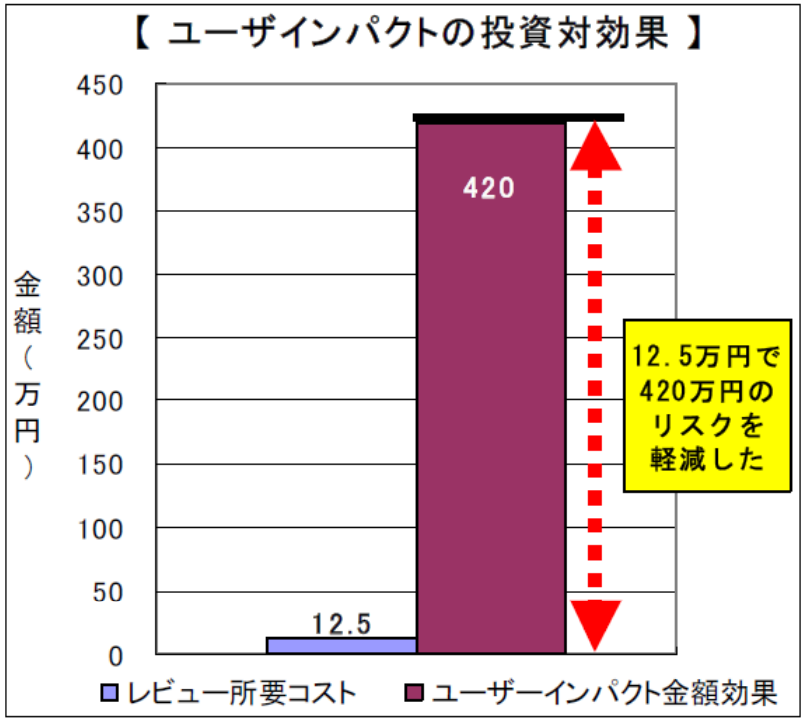
レビュー状況の評価

レビュー／規模 バグ／規模	予定＞実績	予定＝実績	予定＜実績
予定＞実績	品質を判断する時期ではない →レビューをさらに行う	作りこみ品質が予想よりも良い →バグ／レビュー工数で潜在バグ予想値を下方修正	作りこみ品質が予想よりも良い →バグ／規模で潜在バグ予想値を下方修正
予定＝実績	作りこみ品質が予定よりも悪い →バグ／レビュー工数で潜在バグ予想値を上方修正	見直しの必要なし	見直しの必要なし
予定＜実績	作りこみ品質が予定よりも悪い →バグ／レビュー工数で潜在バグ予想値を上方修正	作りこみ品質が予定よりも悪い →バグ／レビュー工数で潜在バグ予想値を上方修正	作りこみ品質が予定よりも悪い →バグ／規模で潜在バグ予想値を上方修正

レビューの効果

- レビューを行いバグを防ぐことで、未然に無駄な支出を防止することが可能である。





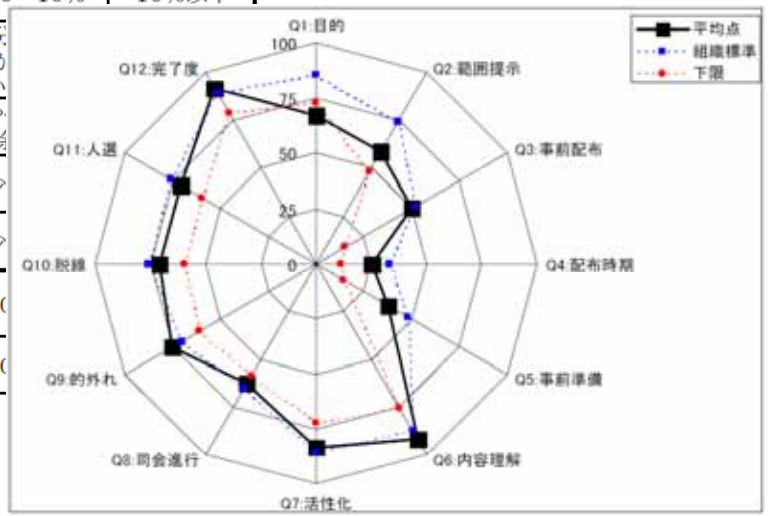
日本科学技術連盟第23年度(2007年度) 分科会成果報告 第1分科会「ソフトウェアプロセス評価･改善」 グループA (論文)レビューの質と価値の定量化の提案

レビューの質を高める取り組み

- レビューが正しく行われていたかは、以下のようにチェックリストで管理できる

表 2. レビューの質チェック票

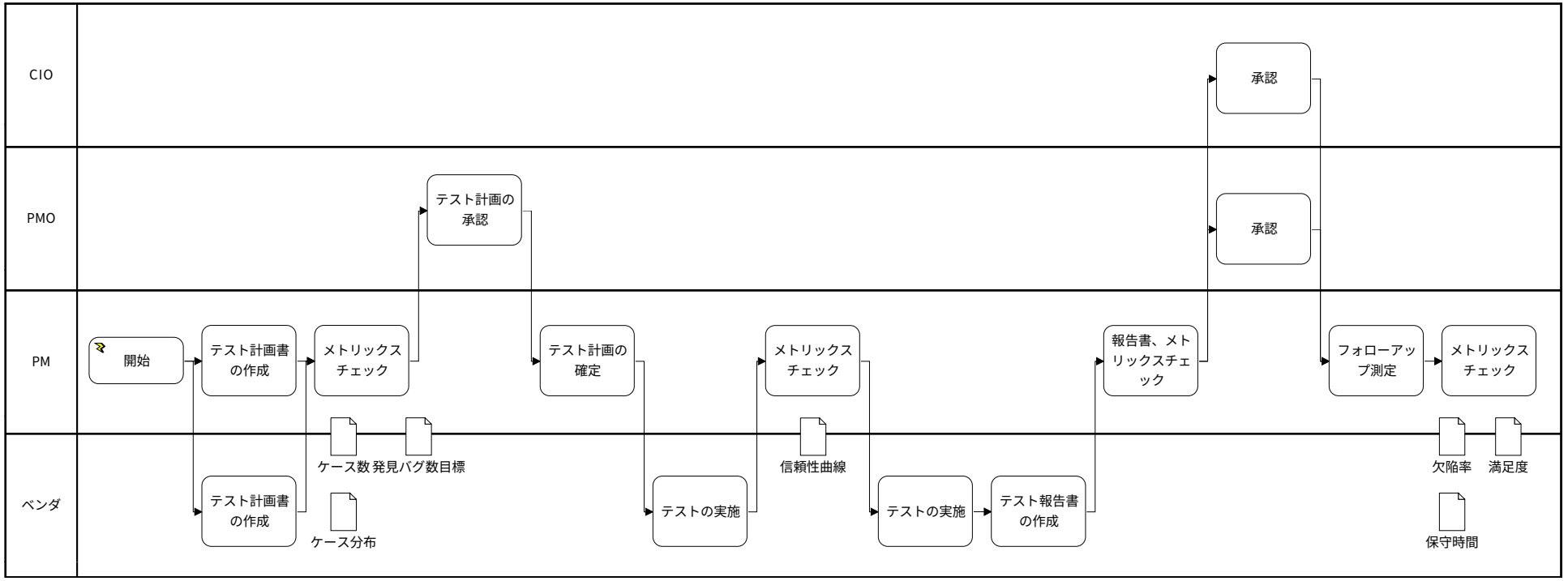
1	レビューの目的／狙いが明確だったか	良い	まあ良い	やや改善の余地有り	改善の余地有り
2	レビュー物件の対象範囲は適切に示されていたか（前回との差分などが明確だったか）	良い	まあ良い	やや改善の余地有り	改善の余地有り
3	必要な資料は全て事前配布／提示されていたか	80%以上	80～50%	50～10%	10%以下
4	資料はいつ頃事前配布されていたか（資料に目を通す期間は確保されていたか）	十分余裕が有った	期間が有った	期間が十分では無かった	事前配布無しまたは直前
5	レビュー会議前に資料に目を通したか	80%以上	80～50%	50～10%	10%以下
6	レビュー物件の内容は十分伝わったか	80%以上	80～50%	50～10%	10%以下
7	全員が満遍なく発言していたか	全員発言していた	半分以上は発言していた	半分	
8	レビューリーダのファシリテートは適切だったか	良い	まあ良い	や	
9	的外れな指摘/コメントが少なかったか	全く無し	殆ど無し	少	
10	脱線することが少なかったか	全く無し	殆ど無し	少	
11	参加者で問題領域をカバーできていたか（レビュー参加者の人選は適切だったか）	100%	80%以上	80	
12	予定のレビュー対象範囲が完了したか	100%	80%以上	80	



日本科学技術連盟第23年度(2007年度) 分科会成果報告 第1分科会「ソフトウェアプロセス評価･改善」 グループA (論文)レビューの質と価値の定量化の提案

出を制す

出のプロセス


CIO
PMO
PM
ベンダ
開始
テスト計画書の作成
メトリックスチェック
テスト計画の承認
テスト計画の確定
メトリックスチェック
報告書、メトリックスチェック
承認
承認
フォローアップ測定
メトリックスチェック
テスト計画書の作成
ケース数
発見バグ数目標
ケース分布
テストの実施
信頼性曲線
テストの実施
テスト報告書の作成
欠陥率
満足度
保守時間

欠陥とは何か

- バグ管理のツールであるBugzillaの定義では以下の通りである。
  - Blocker
    - 開発やテストができ名くるくらい重大なもの
  - Critical
    - クラッシュやデータ損出につながるもの
  - Major
    - 機能が動かないもの
  - Minor
    - 機能は動くが、手続きが必要など完全ではないもの
  - Trivial
    - 機能に影響を与えない記述ミスなど
  - Enhancement
    - 機能強化要望

欠陥はどこまで許されるのか

- システムの品質において、欠陥の含まれる割合は重要であるが、一般的に以下のように分布しており、通常のシステムとしてはBランク以上の品質が望まれ、少なくともCランクは満たしていることが望まれる
- ここで欠陥率とは、「ユーザが発見した欠陥数（顧客側総合テスト以降、フォローまで出発見された欠陥）」÷プロジェクト全体工数である。つまり欠陥率0．25とは4人月あたり1個のバグということである。

	Aランク	Bランク	Cランク	Dランク	Eランク	Fランク	合計
欠陥率	0	0.25未満	0.5未満	1未満	3未満	3以上	
件数	20	77	47	35	28	11	218
割合	9.2%	35.3%	21.6%	16.1%	12.8%	5.0%	100%

欠陥の影響はどのくらいか

- 欠陥はMTTFとも関連があり、それを基に要求品質を設定していく必要がある。

KLOCあたりの欠陥レベル	平均故障間隔（MTTF）	品質レベル（参考）
30以上	2分以下	
20～30	4～15分	
10～20	5～60分	
5～10	1～4時間	
2～5	4～24時間	E～Fクラス相当
1～2	24～160時間	Eクラス相当
1以下	ほとんど故障しない	A～Dクラス相当

プロトタイプレベル

商用ソフトレベル
10-2000件/1000時間

高信頼性ソフトレベル
0.1-10件/1000時間

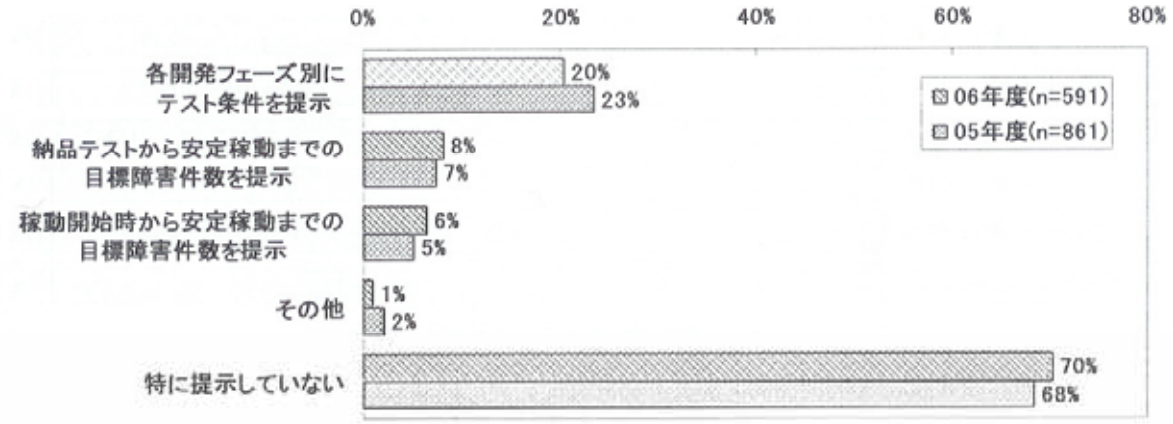
ソフトウェア開発の定量化手法第二版、共立出版

電球が切れる確率1000時間に一回
バイクに乗って怪我をする確率1000時間に0.143回

「ソフトウェアテスト手法」高橋、湯本

品質基準を示すことは有効

- 品質基準を持っているプロジェクトは全体の３７．２％であり、品質基準を持っているプロジェクトの平均欠陥率が０．６０に対して、品質基準を持っていないプロジェクトの平均欠陥率が０．９９となっている。品質に関する高い意識を持っていることと、ユーザ、ベンダにとって超えるべき目標が明示されるので品質向上活動の意識が上がるためと考えられる。
- 日本情報システムユーザ協会「IT動向調査2007」における品質目標の付け方は以下のとおりである。



「IT動向調査2007」JUAS

欠陥率と満足度の関係

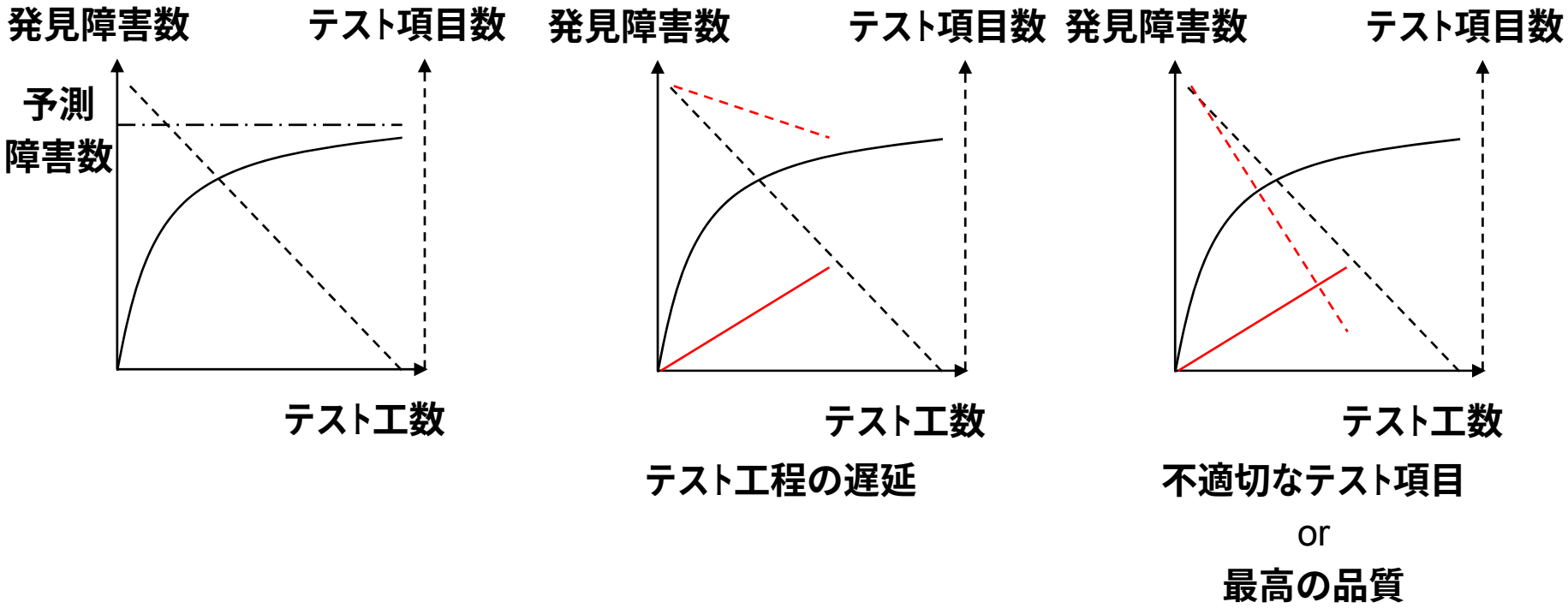
- 欠陥率のみでユーザの満足度が決まるわけではないが、やはり、Bランク（0．25未満）以上のシステムが満足という顧客評価を得られる割合が高くなっている。

換算欠陥率		顧客満足度(品質)					
		満足	やや不満	不満	未回答	計	満足率
0	件数	12	0	0	2	14	100.0%
	割合	85.7%	0.0%	0.0%	14.3%	100.0%	
0.25未満	件数	64	26	3	4	97	68.8%
	割合	66.0%	26.8%	3.1%	4.1%	100.0%	
0.5未満	件数	21	16	4	7	48	51.2%
	割合	43.8%	33.3%	8.3%	14.6%	100.0%	
1未満	件数	9	8	3		20	45.0%
	割合	45.0%	40.0%	15.0%	0.0%	100.0%	
3未満	件数	14	7	2		23	60.9%
	割合	60.9%	30.4%	8.7%	0.0%	100.0%	
3以上	件数	3	2		1	6	60.0%
	割合	50.0%	33.3%	0.0%	16.7%	100.0%	
計	件数	123	59	12	14	208	63.4%
	割合	59.1%	28.4%	5.8%	6.7%	100.0%	

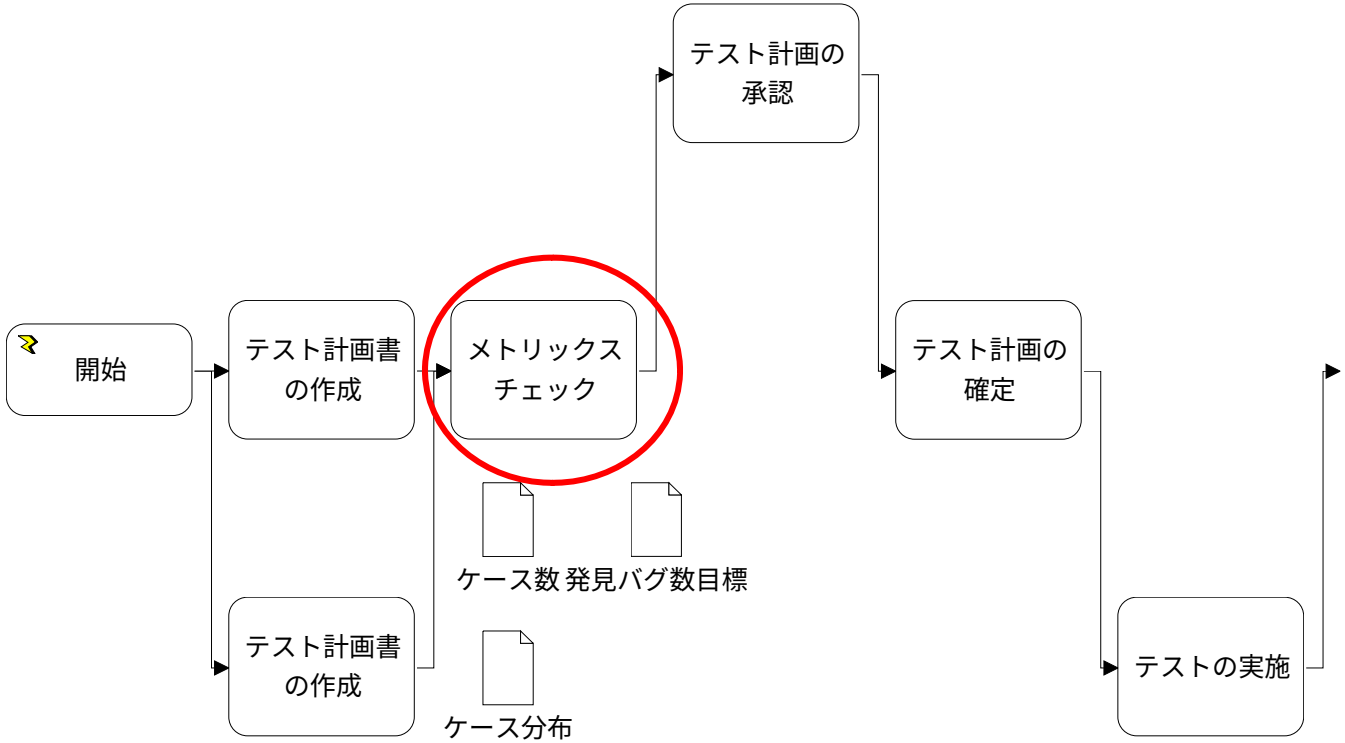
「ソフトウェアメトリックス２００８」JUAS

テストには科学的データがあるが活用されていないことが多い

- テストではバグがあることは証明できるが、バグが無いことを証明することはできない。
  - ただし、コストとバランスの取れたレベルを実現することはできる
- FPあたりのテスト項目数



テスト計画の段階で確認を行う



テスト項目数と項目密度

- 規模あたりのテスト項目数と項目比率は以下のとおりである。
- 特に、項目は多ければよいというものではなく、比率よく配分しなければならない。

	検査項目総数
制御系	30-50件/KLOC
業務系	20-25件/KLOC

	項目比率
基本/正常	50-60%
異常/障害	10-20%
限界/境界	5-10%
周囲条件/インタフェース	20%

第48回SEA-SPIN Meeting　ソフトウェアテストと品質

工程	ケース数
単体テスト	7.5/FP (5-15/FP)
結合テスト	2.5/FP (0.5-8/FP)
総合テスト(1)	1.2/FP (0.6-10/FP)
総合テスト(2)	ー

2007JUAS　QCDワーキング

参考10/KLOC=1/FP

FPあたりのケース密度

- まだ、データ数が少なく、今後のデータの充実が望まれる。

		～10人月	～50人月	～100人月	～500人月	500人月超	記入なし	総計
件数		2	5	5	6	3	3	24
ベンダ内テスト	平均（CASE/FP）	0.7	2.8	3.0	12.7	1.8	0.8	4.8
	最大（CASE/FP）	1.2	5.9	9.9	31.4	3.2	1.4	31.4
	最小（CASE/FP）	0.3	0.1	0.1	2.8	0.5	0.1	0.1
顧客側テスト	平均（CASE/FP）	0.7	0.1	0.6	2.2	0.2	0.2	0.8
	最大（CASE/FP）	1.3	0.3	1.3	6.9	0.4	0.4	6.9
	最小（CASE/FP）	0.0	0.0	0.0	0.0	0.1	0.0	0.0

テスト計画書のメトリックスチェックの例

システムテスト計画書の確認項目

開発区別	新規	改良
言語	Java	
規模（Kstep）	1200	

目標設定値	下限	上限	実績	IPA参考	JUAS参考
テストケース密度（ケース数/Kstep）	2			3.948	22.7374
テストケース数	2400				
不具合検出率（不具合数／Kstep）	0.02	0.08		0.0615	
不具合数	24	96			

評価

移植であるためテストケース密度は低いが、品質目標は適正レベルといえる。

改良 java 中央値 ベンダ内テストのため統計量を1/2	500人以上 平均 言語コンバージョン5/7.63

ソフトウェア開発データ白書2008　P240

主要開発言語別SLOCあたりの総合テストケース数の基本統計量（改良開発）

（ケース数/Kstep）	N	最小	中央	最大	平均
全体	124	0.019	10.188	796.881	45.464
COBOL	43	0.019	6.392	82.875	13.589
C言語	32	1.250	17.114	796.881	84.459
VB	19	0.723	18.784	215.700	47.879
Java	30	0.115	7.896	604.000	48.027

上記総合テストには、ユーザ受け入れ試験を含む

主要開発言語別SLOCあたりの総合テストケース数の基本統計量（改良開発）

ケース数/Kstep	N	最小	中央	最大	平均
全体	122	0.000	0.219	13.333	1.068
COBOL	46	0.000	0.097	12.222	0.905
C言語	30	0.000	0.338	13.333	1.339
VB	18	0.036	0.576	4.917	1.160
Java	28	0.000	0.123	10.000	0.988

上記総合テストには、ユーザ受け入れ試験を含む

ソフトウェアメトリックス調査2008　P142

		～10人月	～50人月	～100人月	～500人月	500人月超	記入なし	総計
件数	件数	2	37	15	26	5	4	89
ベンダ内テスト（ケース数/Kstep）	平均	38.8	92.2	23.5	88	14.9	120.5	75.1
	最大	41.1	963.4	125.1	916	30.5	456.1	963.4
	最小	36.5	0	0.1	0	3.7	0.3	0
顧客側テスト（ケース数/Kstep）	平均	5.2	37.9	26.9	13.6	1.4	2.6	24.5
	最大	10.4	588.5	347.4	80	4.1	5.2	588.5
	最小	0.0	0.0	0.1	0.0	0.1	0.1	0.0

結合テスト計画書の確認項目

開発区別	新規	改良
言語	Java	
規模（Kstep）	1200	

目標設定値	下限	上限	実績	IPA参考
テストケース密度（ケース数/Kstep）			3.36	19.478
テストケース数			4032	
不具合検出率（不具合数／Kstep）			0.051	0.7005
不具合数			61.2	

テストケース項目比率

	比率	参考
基本/正常		50-60%
異常/障害		10-20%
限界/境界		5-10%
周囲条件/インタフェース		20%

第48回SEA-SPINソフトウェアテストと品質

評価

移植であるためテストケース密度は低い。この時点で品質は非常によいが、ケースが少ないため不具合の検証に漏れがある可能性がある。今後もモニターしていく必要がある。

改良 java 中央値 ベンダ内テストのため統計量を1/2

ソフトウェア開発データ白書2008　P240

主要開発言語別SLOCあたりの結合テストケース数の基本統計量（改良開発）

（ケース数/Kstep）	N	最小	中央	最大	平均
全体	104	0.321	24.889	1964.000	84.837
COBOL	32	0.321	15.285	82.438	22.695
C言語	25	3.632	43.894	674.000	111.403
VB	15	1.445	32.061	631.064	115.358
Java	32	3.774	38.956	1964.000	111.955

上記総合テストには、ユーザ受け入れ試験を含む

主要開発言語別SLOCあたりの結合テストケース数の基本統計量（改良開発）

ケース数/Kstep	N	最小	中央	最大	平均
全体	102	0.000	1.126	42.357	2.770
COBOL	34	0.000	0.457	5.226	1.181
C言語	25	0.067	1.333	42.358	5.230
VB	15	0.181	0.896	10.946	2.599
Java	28	0.000	1.401	24.000	2.596

上記総合テストには、ユーザ受け入れ試験を含む

カットオーバー時の品質の保守への影響

- 保守作業との関係はあるが、欠陥率のデータはまだ十分ではない。

平均保守作業期間

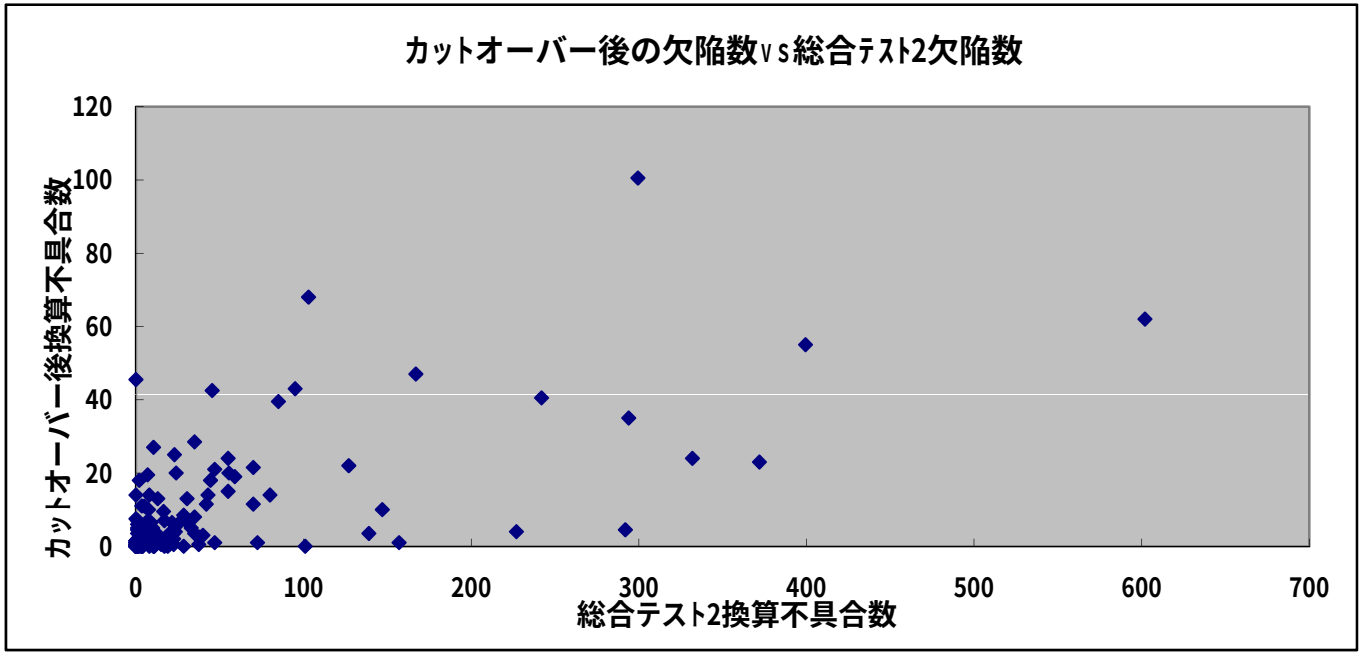
	半日以下	1日以内	3日以内	1週間以内	1月以内	1月以上
CO時の品質はよい	31.7%	19.4%	20.0%	12.1%	11.1%	5.7%
CO時の品質はよくない	29.9%	17.8%	14.3%	14.1%	14.3%	9.5%

CO時品質	初年度保守欠陥率	2年目以降保守欠陥率	受入確認即時合格率
非常によい	8.5%	6.3%	50.3%
よい	20.1%	11.7%	64.2%
普通	19.7%	13.4%	61.2%
やや悪い	22.1%	7.1%	81.2%
非常に悪い	9.0%	5.0%	95.0%

保守欠陥率＝欠陥発生件数／保守作業実施件数

サービス開始後の品質は適正レベルか

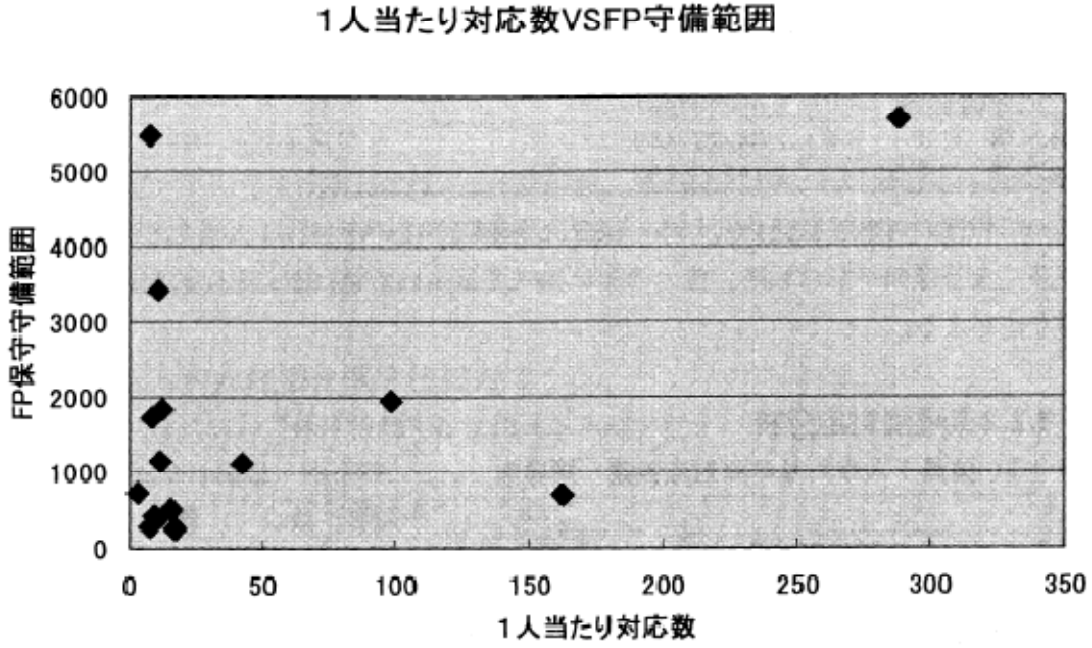
- またテスト中の欠陥数から稼働後の欠陥数を推定することができ、これと比較することにより品質の安定度を判断することが可能になる。テスト時の発生不具合数から、導入初期の不具合数を想定し、それを念頭に置いた運用を行い、異常な不具合の多さなどが見られるときは、テストデータの確認など必要な措置を講じる必要がある。



「ソフトウェアメトリックス２００８」JUAS

保守による品質の検証

- 規模あたりの欠陥数、フォロー時の欠陥の多さで品質の概要はつかめるが、長期にわたっても検証をすることが重要である。保守要員がどのくらいの範囲を見ているのか、その中でどのくらいの対応をしているかという点からも対策が考えられる。
  - FP守備範囲が広く、一人当たり対応数が多いものは、保守要員の人で不足による悪循環などが想定される。



「ソフトウェアメトリックス２００７」JUAS

稼働率の目標

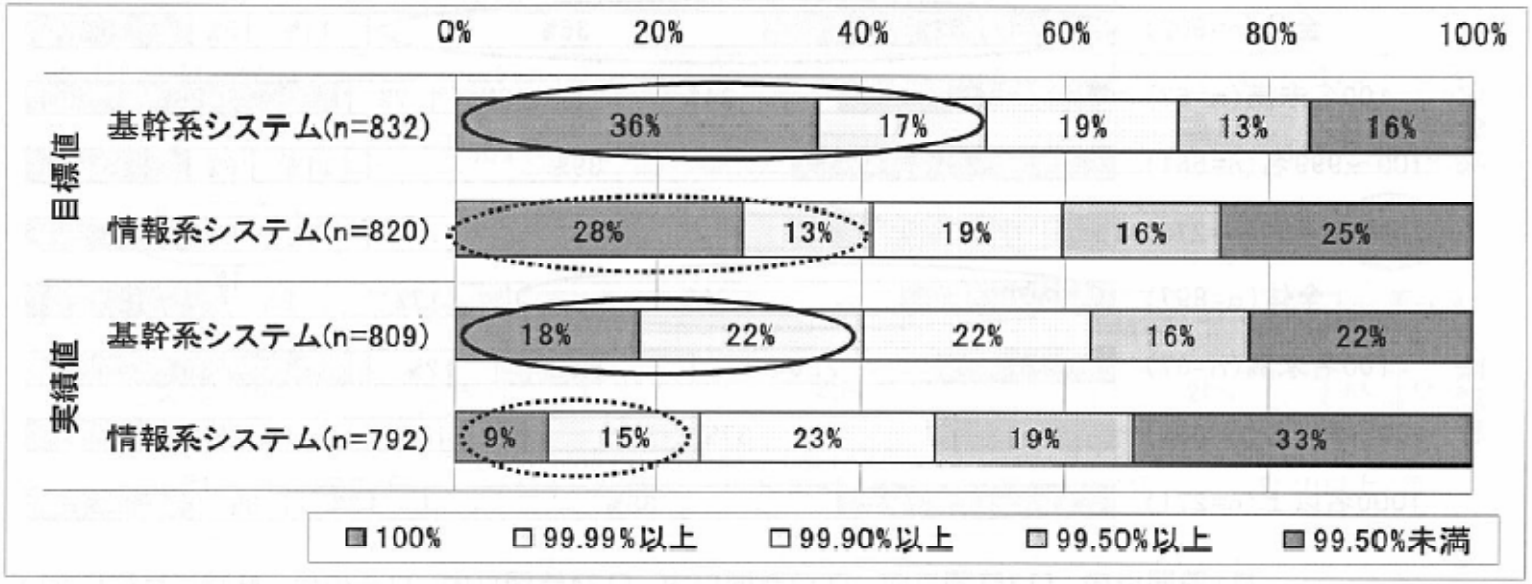
	レベル1	レベル2	レベル3	レベル4	レベル5
稼働率	98%以下	99%	99.9%	99.99%	99.999%以上
バックアップ機	なし	あり (部分的)	あり (2／N＋1台)	あり (Hot stand by)	あり (Hot stand by)
サービス停止時間 (　)時間／年	172時間	86時間	8.6時間	50分	5分
到着時間	1-6時間 (昼) 12時間 (夜間)	1-6時間	1-3時間 (昼) 6時間 (夜間)	常駐 ケースによっては2時間	常駐
修復時間 ・故障修復 ・再立ち上げ	6時間-12時間 10分-1時間	6時間-12時間 10分-1時間	3時間-6時間 10分-1時間	3時間-6時間 0分-10分	3時間-6時間 即時
費用 ・構築費用 ・運用費用	1.0倍 1.0倍	1.2～1.8倍 1.1～1.3倍 (マニュアル)	1.2～3倍 1.3～2.0倍	1.5～4倍 2.0～3倍 (保守も)	4～6倍 3～4倍
システム構成 (例) 必要な機能		NAS	SAN NAS クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ	SAN クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ 三重化	SAN クラスタリング ﾛｰﾄﾞﾊﾞﾗﾝｼﾝｸﾞ 三重化、四重化
ペナルティ			対象	対象	対象

実際の稼働率

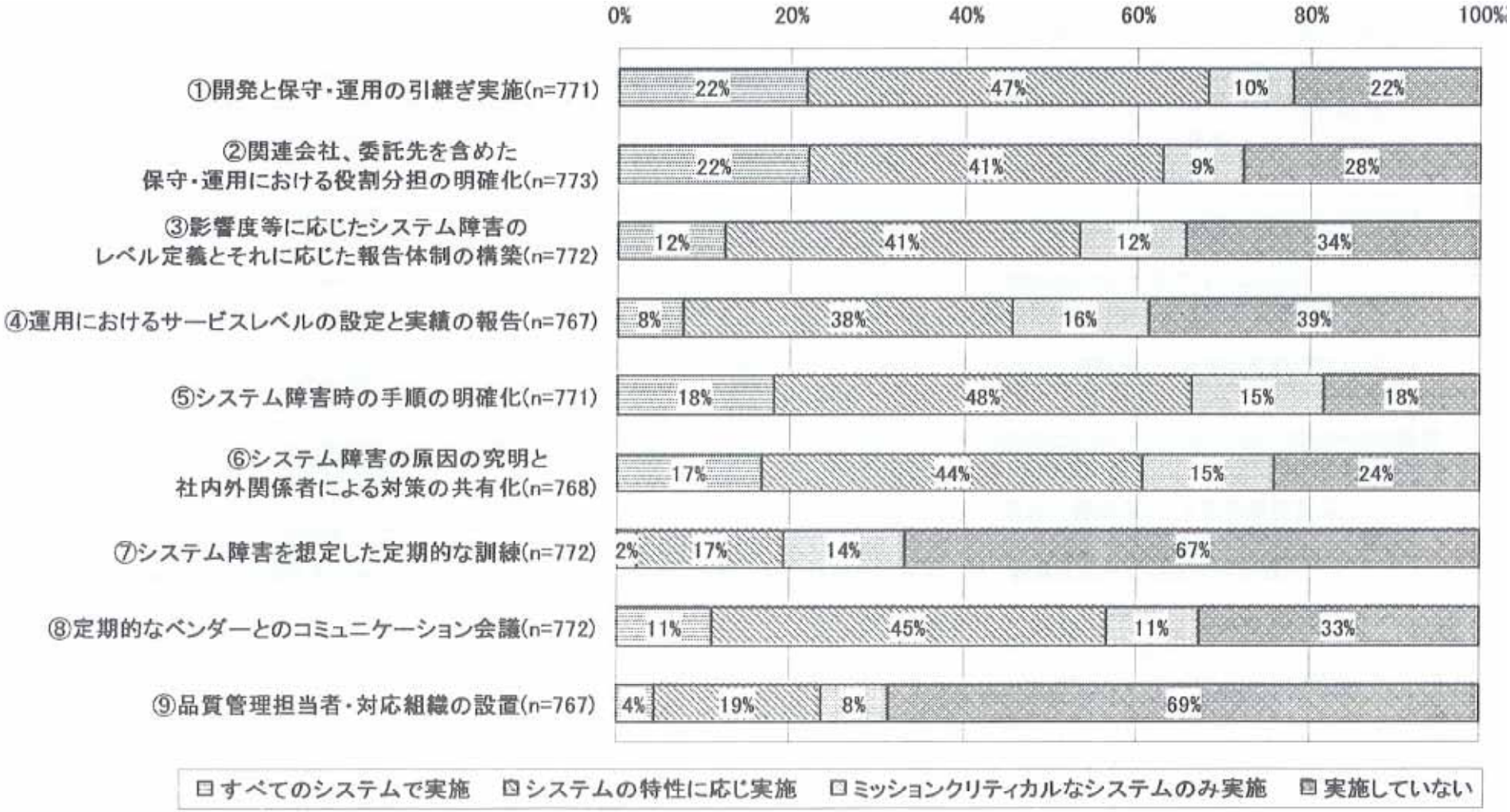
基幹系システムでの稼働率の目標値は99.99%以上が53%だが、実績値は40%と乖離が大きい

•基幹系システムの稼働率目標は、「99.99%（1年間で50分の停止）以上」としている企業が53%、情報系でも41%。

•それに対し実績値は、「99.99%以上」と回答した企業が、基幹系システムで40%、情報系システムで24%となっており、目標稼働率と実績稼働率がかなり大きく離れている。

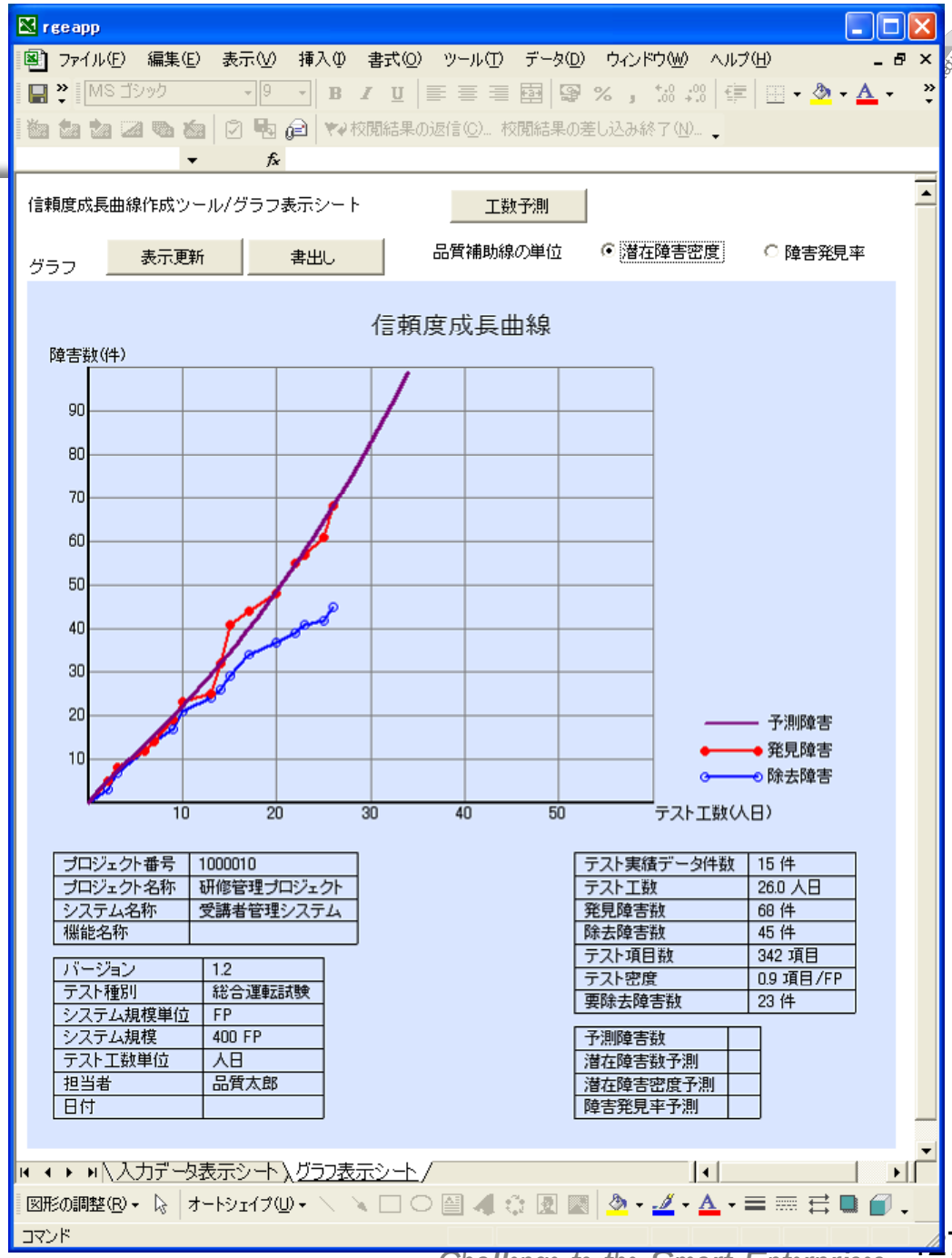


信頼性向上のためのシステム保守・運用段階での施策



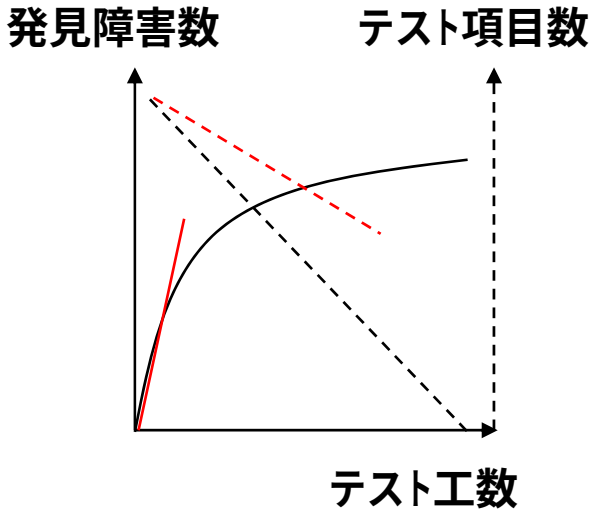
演習

- システム検収段階になり信頼性曲線の提示を求めたところ、以下の曲線になっていた。どのような確認を実施するべきか



演習

- テスト工程にテスト消化状況の提示を求めたところ下図のようなグラフが提示された。
- 問題の原因と対策を考えてください。



まとめ

システム開発･保守プロジェクトのQCD向上

- エンジニアリングのデータを使うことによって、中長期的には間違いなく品質は上がる
- 自社だけで取り組むのは限界があり、業界全体で協力していくことが重要
- 品質、テストに知見を持った専門家を企業内で育成していくことが重要
- 現場に過度の負担をかけてはいけない。Good Enough Qualityを目指す
- メトリックスが提示する数字は、あくまでも参考値であり、それを基に考える必要がある
  - ディスカッションのスタートラインに過ぎない

参考：投資対効果などと総合的に管理

- 以下のようにシステムのライフサイクルを通じて、投資対効果とあわせて継続的に把握していくことも重要である。

	従来	予算要求時	最適化計画	基本設計	詳細設計	導入時	導入1年後	導入2年後	導入○年後
利用者満足度	30%	80%	80%	80%	80%	23%	26%	48%	
職員満足度				70%	70%	70%	35%	32%	
コスト削減（予測）額（円／年）		100,000,000	120,000,000	80,000,000	70,000,000	45,000,000	40,000,000	40,000,000	
経済効果									
利用者数（人）・・紙利用者含む	10,000	10,000	10,000	12,000	12,000	12,000	12,000	13,000	
利用率	0%	98%	98%	95%	95%	15%	14%	16%	
プロセス満足度						24%	26%	26%	
総入力時間（分）	0	5	5	5	5	5	5	5	
総手続き時間(分)	60	20	15	15	15	18	18	18	
総事務時間（分）	120	20	30	25	20	22	20	19	
手続き当たりのコスト(円)						3,000	29,000	28,000	
職員・組織に対する満足度						26%	32%	35%	
組織成熟度									
職員充実度						80%	100%	100%	
機器やシステム等の技術に対する満足度						39%	42%	42%	
技術基盤成熟度									
全費用(円)		500,000,000	500,000,000	500,000,000	500,000,000	500,000,000	600,000,000	700,000,000	
既投資		0	20,000,000	100,000,000	200,000,000	400,000,000	450,000,000	500,000,000	
効果予測（コスト削減）		600,000,000	720,000,000	480,000,000	420,000,000	270,000,000	240,000,000	240,000,000	
効果予測（総合効果）		700,000,000	820,000,000	580,000,000	520,000,000	300,000,000	270,000,000	270,000,000	
投資対効果（コスト削減）		1.20	1.44	0.96	0.84	0.54	0.40	0.34	
投資対効果（総合効果）		1.40	1.64	1.16	1.04	0.60	0.45	0.39	
稼働率				99%	99%	99%	98%	99%	
平均故障間隔（時間）						1000	800	2000	
平均回復時間（分）						0	30	30	