高信頼化ソフトウェアのための開発手法ガイドブック

- 予防と検証の事例を中心に -

(Ver.1.0)

独立行政法人　情報処理推進機構
ソフトウェア・エンジニアリング・センター
エンタプライズ系ソフトウェア開発力強化推進委員会

高信頼ソフトウェア領域
高信頼化のための手法ＷＧ

2010 年 9 月

< 空欄 >

高信頼化ソフトウェアのための開発手法ガイドブック
- 予防と検証の事例を中心に -

はじめに

既発刊の「ソフトウェアテスト見積りガイドブック」に続き、本ガイドブックではソフトウェアの高信頼化を目指し、ソフトウェアの品質保証活動にかかわる予防活動および検知活動での各種手法や技法を中心に、高信頼化ソフトウェアのための開発手法にかかわる解説を行います。構成は大きく 2 つに分かれており、第 1 部が「総論」、第 2 部が「高信頼ソフトウェアに向けた各社の取り組み事例集」となっています。

第 1 部では「高信頼」を「高品質」ととらえ、ソフトウェアの品質保証活動にかかわる予防活動および検知活動にかかわる手法や技法の一般的な事項および心得的な内容を、実際の活動につなげるための方法について示しています。予防活動にかかわる手法は、ソフトウェアでの追跡性（トレーサビリティ）の管理に関する手法、および障害事例から代用特性展開表を利用した再発防止策を紹介し、検知活動にかかわる手法では、テストの網羅性と効率性を客観的に表現する直交表などを使用したテスト技法を紹介しています。

対象読者は、次表に示す A)～D)のすべての方を想定しています。ソフトウェア開発プロジェクトのマネジメント・運営を行う観点から示されている事項も多く、ユーザ企業にとっては、ビジネス要求の導出に基づく業務要件、品質要件およびシステム要件の明文化、ならびにそれら要件を担保する妥当性確認方法など、自社の仕組みを見直すための参考にしていただきたいと考えます。

想 定 読 者
A) ベンダ企業のプロジェクトマネージャ
B) ベンダ企業の社内改善メンバ（企画メンバ）
品質保証部門の担当者
C) ユーザ企業の契約担当者
ベンダ企業の営業担当者
D) ユーザ企業のトップマネジメント
プロジェクトマネージャ
システム部門メンバ
社内改善メンバ（企画メンバ）

第 2 部では、高信頼ソフトウェアに向けて先進的な取り組みを行っている各社の事例、当該方法の導入に当たっての留意点を示します。具体的な例として、第 1 部から必要に応じて参照されるとともに、個別の方法に興味のある方が 1 つ 1 つの事例を独立して参照できるように構成しております。

本ガイドブックの前提および対象範囲は次の通りです。

- 一般的に、開発プロセスマネジメントモデルにおける定量的なマネジメントは、ソフトウェア開発の効果を上げるために、最適な各種手法や技法を使用することが求められており、その証跡が重要になります。
  本ガイドブックで紹介するソフトウェア品質保証活動にかかわる予防活動および検知活動での各種手法や技法は、そのことを前提に取り組んでいます。

- ソフトウェアの品質にかかわるJIS規格（JIS X 0129-1:2003）によれば、ソフトウェアの品質「外部品質、内部品質」は、6つの品質特性と27の品質副特性から構成され定義しています。その中の「信頼性」は6つの品質特性の1つであり、4つの品質副特性（成熟性、障害許容性、回復性、信頼性標準適合性）から構成されています。
  本ガイドブックでの「品質」は、要求品質の網羅性および水準を的確にとらえるため、「信頼性」を含む6つの品質特性と27の品質副特性を対象としています。

- 一般的に高信頼ソフトウェアの対象は、運輸、電力・ガス、通信、金融および医療などの公共性を有する事業領域の業務システムが対象となります。しかしながら、本ガイドブックでは、さらに個々の一般企業等の重要度評価（障害時の損害金額やお客様迷惑度指数の多寡など）において、重要インフラ情報システムとして判定された情報システムのソフトウェアも高信頼ソフトウェアの対象としています。

- ビジネスアプリケーションを中心としたソフトウェア開発を対象としています。ただし、本ガイドブックで示されたガイドラインは、組込みソフトウェアなど、他の分野でも十分に適用可能です。

なお本ガイドブックでは、あくまでも手法および技法を中心に紹介しますが、定量的なマネジメントと連携しプロセスを安定化させ、さらなる改善を通してプロセス成熟度を高水準へと向上させることにより、高信頼なソフトウェアが実現できることを期待しています。

2010 年 9 月

高信頼ソフトウェア領域　高信頼化のための手法WG　一同

本ガイドブックの読み方

☆　ベンダ企業のプロジェクトマネージャ

第 1 部の第 1 章～第 7 章を通読してください。また第 2 部の各事例を参照し、品質保証活動と結びついた予防活動および検知活動での具体的な手法事例として示されている内容を活用してください。

☆　ベンダ企業の社内改善メンバ(企画メンバ)、品質保証部門の担当者

第 1 部の第 1 章～第 7 章を通読してください。第 1 部の第 4 章を参考にして、予防活動に結びついた障害対応プロセスの適用検討など、組織的な品質保証活動の成熟度向上に取り組んでください。また設計、開発およびテストでの一貫性を識別したトレーサビリティ管理方法、ならびにそれら一貫性を担保する検証方法など、自社における仕組みの見直しに役立ててください。第 2 部の各事例を参照し、自組織との共通点や差異を把握して、実務上での予防活動および検知活動に役立ててください。

☆　ユーザ企業の契約担当者、ベンダ企業の営業担当者

第 1 部の第 1 章～第 3 章を通読してください。続いて、第 1 部の第 6 章を参考にして、検収条件（テスト網羅性、残存欠陥率）などを検討してください。第 2 部の各事例を参照し、ソフトウェアの品質と価格に関する考え方および品質保証活動事例として示されている考え方を把握してください。

☆　ユーザ企業のトップマネジメント、プロジェクトマネージャ、システム部門のメンバ、社内改善メンバ(企画メンバ)

第 1 部の第 1 章～第 6 章を読み、第 1 部の第 4 章を参考にして、予防活動に結びついた障害対応プロセスの自社への適用検討など、組織的な品質保証活動の成熟度向上に取り組んでください。また業務要件、品質要件およびシステム要件の明文化、ならびにそれら要件を担保する妥当性確認方法など、自社における仕組みの見直しに役立ててください。また第 2 部の各事例を参照し、自組織の品質の向上に役立ててください。

(注) 本ガイドブックの事例紹介は、各社で使用している情報システム用語で記述しています。

目　　次

第 1 部　　総　　論

はじめに i
本ガイドブックの読み方 iii

第 1 章　高信頼ソフトウェアにおける問題意識 3

1.1　情報システムを取り巻く環境の変化と想定リスク 3
1.1.1　環境の変化と情報システム 3
1.1.2　想定される新たなリスク 3

1.2　情報システム全体を鳥瞰した取り組み 4

1.3　高信頼ソフトウェアへの取り組み課題 4
1.3.1　ディペンダビリティとしてとらえたソフトウェア品質 4
1.3.2　定量的なマネジメントを目指したプロセスマネジメントモデルの適用 5
1.3.3　定量的なマネジメントに結びついた手法の導入 6

第 2 章　高信頼ソフトウェアとは 7

2.1　高信頼ソフトウェアと品質特性 7

2.2　システムプロファイルに基づく重要インフラ情報システム 8

2.3　予防活動と検知活動を対象にした手法 11

第 3 章　予防活動および検知活動にかかわる手法 13

3.1　予防活動にかかわる手法の一般的な事項 14

3.2　検知活動にかかわる手法の一般的な事項 17
3.2.1　レビューおよびテストでの欠陥検出戦略の統合 18
3.2.2　レビュー手法の概要 18
3.2.3　テスト技法の概要 28

第 4 章　障害事例から学ぶ予防活動 47

4.1　予防活動としての障害事例の扱い方 47

4.2　障害発生時の障害分析と対処方法 49
4.2.1　再発防止策の立案方法 49
4.2.2　予防活動として整理・蓄積された情報の活用方法 50

4.3　品質特性ごとの再発防止事例 51
4.3.1　品質特性ごとの代用特性事例 51
4.3.2　代用特性に関連する「障害・再発防止事例」 55

第 5 章　トレーサビリティ管理の手法 101

5.1　トレーサビリティの重要性 101

5.2　トレーサビリティとは 101

5.3　品質展開の概要 103
5.3.1　品質機能展開の原理 103
5.3.2　品質展開の手順概要 104
5.3.3　品質展開の規則 105
5.3.4　品質展開の手順 106

5.4　品質展開の要点および期待効果 107

5.5　ソフトウェア開発におけるトレーサビリティ管理の具体例 108
5.5.1　顧客要件一覧表の作成事例 108
5.5.2　要求品質の優先順位付け事例 111
5.5.3　要求品質に基づくトレーサビリティ管理事例 112

第 6 章　テスト網羅性の高度化技法 117

6.1　テスト要求分析 118

6.2　テスト観点とテストアーキテクチャ設計 119
6.2.1　テスト観点 119
6.2.2　テストアーキテクチャ設計 121

6.3　高度化技法 125
6.3.1　直交表を活用した網羅的な組み合わせテスト 125
6.3.2　シナリオを用いた効果的なピンポイントテスト 131

6.4　エンタプライズ系代表企業十社のテスト実態分析 132

6.5　テスト実態分析結果の使い方 137
6.5.1　各社テスト一覧表の使い方 137
6.5.2　テスト観点表の使い方 138

第 7 章　高信頼ソフトウェア開発に関する海外事例および研究動向 141

7.1　はじめに 141

7.2　欧米における情報システムの信頼性確保の取り組み 141
7.2.1　調査内容 141
7.2.2　調査結果 142

7.3　ソフトウェア工学に関連する国際会議に見る研究動向の分析 146

第 1 部 参考文献 150

第 2 部　事　例　編

第 1 章　事例編の概要 153

第 2 章　東京海上日動システムズ株式会社の事例 155

2.1　取り組みの背景 155

2.2　システム重要度に基づいたリスク管理 155
2.2.1　概要 155
2.2.2　システムプロファイル 156
2.2.3　システムリスク・アセスメントシート 156
2.2.4　効果 157

2.3　設計・製造工程の品質向上 157
2.3.1　取り組みの背景 157
2.3.2　要件の精度向上の取り組み 157
2.3.3　設計・製造品質向上の取り組み 158
2.3.4　取り組みの効果 158

参考文献 160

第 3 章　富士ゼロックス株式会社の事例 161

3.1　ソフトウェアの品質保証の課題と対応 161

3.2　ソフトウェアテストの概略 161

3.3　HAYST 法の詳細 162
3.3.1　テスト戦略とテスト分析 163
3.3.2　テスト設計 164
3.3.3　テスト実装 165
3.3.4　テスト実施 166

3.4　活動の効果 166

参考文献 166

第 4 章　日本ユニシス株式会社の事例 167

4.1　取り組みの背景 167
4.1.1　プロセスの標準化 167
4.1.2　テスト技術への取り組み 168
4.1.3　フロントローディングと W モデル型開発 169

4.2　検査体系の構築 169
4.2.1　体系構築のための基本的考え方 169

4.2.2 検査体系 170

4.3 ソフトウェア検査部の役割 170
4.3.1 基準の設定 170
4.3.2 支援 171
4.3.3 検査 173

4.4 アセスメントの実際 174
4.4.1 設計書アセスメント 174
4.4.2 テストアセスメント 175

4.5 実績 175
4.5.1 共通課題 175
4.5.2 費用対効果 175

4.6 今後の取り組み課題 176

参考文献 177

第 5 章 株式会社日立製作所の事例 179

5.1 取り組みの背景 179

5.2 高信頼性重点管理システム 179
5.2.1 概要 179
5.2.2 対象システム 179
5.2.3 推進体制 179
5.2.4 管理プロセス 180
5.2.5 効果 181
5.2.6 当該手法の優位点と課題 182

5.3 保守フェーズのリスク管理強化 182
5.3.1 概要 182
5.3.2 対象システム 182
5.3.3 推進体制 182
5.3.4 管理プロセス 183
5.3.5 効果 183
5.3.6 当該手法の優位点と課題 184

参考文献 184

第 6 章 富士通株式会社の事例 185

6.1 取り組みの背景 185

6.2 高信頼ソフトウェアにかかわる手法 185
6.2.1 要件定義監査 185
6.2.2 外部設計ドキュメント診断 186

6.3 手法導入による費用対効果 188

6.4 当該手法の優位点と課題 188

参考文献 189

第 7 章 三菱電機株式会社の事例 191

7.1 取り組みの背景と経緯 191

7.2 高信頼情報システム開発の進め方 191
7.2.1 高信頼情報システムの構築プロセス 192
7.2.2 実行基盤環境の評価・選定のプロセス 194

7.3 非機能要求トレーサビリティ管理の方法 197
7.3.1 非機能要求に対するトレーサビリティマトリクスの様式 197
7.3.2 トレーサビリティ管理の実施手順 198

7.4 試行結果 199

7.5 非機能要求のトレーサビリティ管理プロセスの導入効果 199

第 8 章 株式会社ジャステックの事例 201

8.1 取り組みの背景と経緯 201

8.2 ソフトウェアの信頼性にかかわる問題意識 201

8.3 ソフトウェアの信頼性と開発コストとの関係 202
8.3.1 見積りモデルの基本アルゴリズム 202
8.3.2 システムプロファイルごとの要求品質と開発コストとの関係 202
8.3.3 出荷後の残存欠陥とソフトウェア開発コストとの関係 206

8.4 高信頼ソフトウェアを目指した見積りモデルの有効利用 207

参考文献 209

INDEX 211

第1部

総　論

< 空欄 >

第1章　高信頼ソフトウェアにおける問題意識

1.1 情報システムを取り巻く環境の変化と想定リスク

1.1.1 環境の変化と情報システム

近年、情報システムはIT（情報技術）の進展などにより利用範囲が広がっているとともに、企業活動の重要なインフラとしての機能のみならず、一般国民の生活に直結する重要な社会インフラとしての役割も担っています。

このような状況下、情報システムの構成要素の1つであるソフトウェアは、開発量の増大と機能の複雑化、さらにコスト削減と開発期間の短縮などの要求とともに、信頼性・安全性への対応課題が顕在化してきています。

(1) 質の高度化とともに情報システムの規模の増大

業務効率の改善に伴う自動化の推進および利便性向上を目指した商品ならびにサービスの高度化は、情報システムの複雑化および大規模化を加速させ、それに伴いソフトウェアへの依存度が高まり、ソフトウェアの開発量を増大させています。

(2) 時空に拡がる情報システム

情報システムはインターネット普及などによる個人の利用、規制緩和などによる異業種間の業務連携およびグローバル化の進展に伴う海外との業務連携など、ますます利用範囲が拡大しています。IT、とりわけネットワーク技術の進展により、情報が広範囲に、かつリアルタイムに授受されてきています。さらには記憶密度の飛躍的な向上により、情報の蓄積と利用範囲は過去および未来にかけて拡がってきています。

その結果、情報システム間にかかわる相互接続性の多様化および障害時などによる情報の遡及対応への考慮など、ソフトウェア機能は単機能から複合機能へと複雑化してきています。

(3) 情報システムにかかわる QCD* 要求の高まり

100 年に一度と言われている昨今の景況悪化により、情報システムの構築、保守および運用においても、例外なくコスト削減への期待を課せられています。また、大規模化に伴う社会への影響から、情報システムにおける品質の担保責任、および企業間競争によるソフトウェア開発期間の短縮要求などが激しくなってきています。

1.1.2 想定される新たなリスク

重要な社会インフラとして利用される情報システムが、何らかの事情により機能しなくなった場合、企業経営リスクだけでなく、多数の顧客およびシステムの利用者に影響を及ぼし、多くの国民生活に支障をきたすような社会問題に発展するリスクがあることは言うまでもありません。特に、ここでは、1.1.1 項で記載した情報システムを取り巻く環境の変化から、新たに留意すべきリスクに関して提起します。

(1) 複合障害による負の連鎖

情報システムの規模増大および時空への拡がりは、情報システム間の相互接続性が多様化し、ソフトウェア機能が単機能から複合機能になることにより個々の欠陥による障害が、

---

* 品質（Quality）、コスト（Cost）、納期（Delivery）

その対応ミスなどによる新たな障害を誘発させるなど、負の連鎖を増加させています。
あるいは、いくつかの条件が組み合わさり、時間の経過とともに重大障害を発生させたりするような、いわゆる交互作用‡の可能性が高まっています。
つまり個々の欠陥による障害が、単独の場合に与える被害の質および量の総和を超える相乗的な障害として表れ、複合障害へと発展するリスクが増えることになります。

(2) リアルタイムに伝播する負の連鎖

異業種連携やグローバル化は、情報システムの利用範囲の広がりをみせるとともに、ネットワーク技術の進展により広範囲にかつリアルタイムでの情報共有が可能となる一方で、一度、障害が発生すると、その影響は広範囲にかつリアルタイムに負の連鎖が広がってしまうリスクへと変遷します。
特に、昨今のように外部システムの利用、異業種間結合およびグローバルネットワークの利用など外部との相互接続の増加は、当該システムにかかわる障害の影響を、瞬時に他の相互接続するシステムに伝播させてしまいます。例えば、打ち上げ花火を打ち上げてから、さく裂するのを止めるような事態が発生する可能性が高くなるわけです。

1.2 情報システム全体を鳥瞰した取り組み

特に重要な社会インフラとして利用される情報システムは、1.1.2項で述べたようなリスクに対応するために、情報システムの企画、開発、保守、運用にかかわるプロセスと実装技術、教育制度、組織の構造と文化、さらには商慣行など、様々の問題や課題を広範囲に考慮する必要があります。
一方で、リスクはいかなる対応をしても完全になくすことはできません。リスクの極小化およびリスク発生時での影響範囲の局所化など、リスクをコントロールすることが、大切になってきます。例えば、障害影響度などを考慮したコンティンジェンシープラン、障害事例の蓄積による再発防止対策、さらには事業継続に配慮したBCP（Business Continuity Plan）、BCM（Business Continuity Management）などを考慮する必要があります。

1.3 高信頼ソフトウェアへの取り組み課題

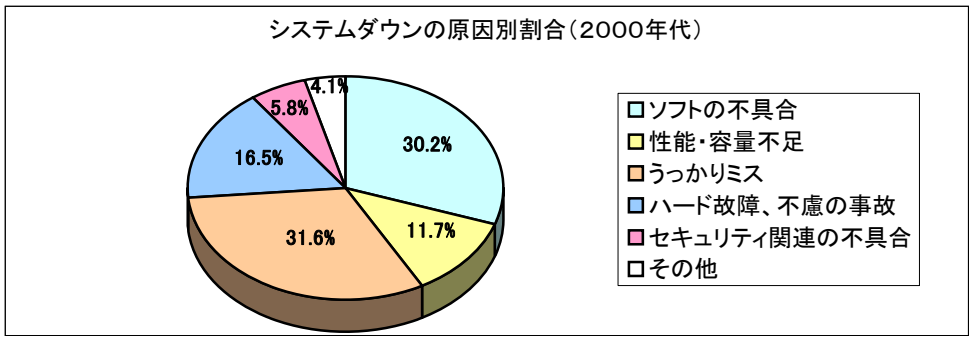
1.3.1 ディペンダビリティ§としてとらえたソフトウェア品質

ソフトウェアの障害は、システムのリリース直後に多く発生し、時間とともに減少するのが一般的な傾向です。システム障害を起した原因分類については、日経コンピュータ誌および日本情報システム・ユーザー協会（JUAS）などで調査報告が発表されています。
その調査結果によりますと、両調査報告ともソフトウェアの障害は全体の約30%として報告されています。この報告は業務が停止するようなシステムダウンを対象にしていますが、業務停止に至らないシステム障害まで数えると、ソフトウェアの欠陥による障害は格段に多くなっているのが実態です。

---

‡ 因子同士が作用することによる相乗効果、因子の効果が、他の因子の水準によって影響を受ける場合に発生する（Interaction）。

§ JIS C5750-1:2010 では、ディペンダビリティとは（Dependability）は、「アベイラビリティ性能およびこれに影響を与える要因、すなわち信頼性性能、保全性性能、保全支援能力などを包括的に記述するための総称（注記：ディペンダビリティは、非定量的用語として一般的記述に限り用いられる）」と定義されている。



（出典）　日経コンピュータ誌　2009.8.19

図 1-1　システムダウンの原因別割合

ソフトウェアの障害を限定し分析することは、ソフトウェア欠陥（プログラムバグ等）の傾向をとらえ、対策をする上で大切になります。

しかし、ディペンダビリティとしてのソフトウェア品質をとらえる場合は、ソフトウェアの障害のみならず、品質を網羅的にとらえる必要があります。例えば、日経コンピュータ誌の障害分類調査から情報システムの運用上において「うっかりミス」による障害が31.6%と第1位になっており、その中には「うっかり操作ミス」も含まれていると思われます。

このような「うっかり操作ミス」の運用にかかわる改善として、運用操作マニュアルの見直し、運用オペレータの再訓練および監視体制の強化など様々な対策が考えられますが、人間はミスを犯すものとの認識をもち、「うっかり操作ミス」に対するソフトウェア開発での対策を講じておくことも重要です。この場合、「操作ミスを誘発しない操作のしやすさ」という要求品質が明確に提示され、ソフトウェア機能として「画面操作時のナビゲート機能」、「エラー項目のカラー表示機能」、「日本語の表示機能」などの実装を確実に行うことが必要になります。

このように、ディペンダビリティとしてソフトウェア品質をとらえる場合は、品質特性（JIS X 0129-1:2003「ISO規格ではISO/IEC9126-1:2001」）を使用し、広くソフトウェアの要求品質をとらえる必要があります。「うっかり操作ミス」の事例では、品質特性（使用性）、品質副特性（運用性）としてとらえています。

1.3.2　定量的なマネジメントを目指したプロセスマネジメントモデルの適用

ソフトウェアは頭のなかでの思考が製品に直結するため、開発プロセスおよび製品それ自体の柔軟性や自由度が高くなる反面、プロセスおよび製品品質を可視化し、コントロールする上で不確定要素が多くなり標準化がし難い傾向にあります。

そのため、一般的にソフトウェアの信頼性は客観的な把握や制御が難しいと言われており、ソフトウェアのディペンダビリティを企画、開発、保守および運用のライフサイクル全般を通して確保および向上させるために、各ライフサイクルフェーズでのプロセスマネジメントモデルの導入が必要になります。とりわけ開発フェーズではシステム開発プロセスのマネジメントモデル（例：CMMI**など）の適用が重要になります。

一般的に開発プロセスのマネジメントモデルは、品質基準に準拠した実施状況を測るフレームワークを提供しているにすぎません。モデル適用の組織は、フレームワークに沿って、組織としての具体的な品質基準、各種手順、各種手法および技法を用意し、実践

** CMMI:(Capability Maturity Model Integration;by Carnegie Mellon University SEI)
連続表現モデル（能力度レベル「CL0~CL5」）と段階表現モデル（成熟度レベル「ML1~ML5」）が存在し、一般的には成熟度レベルでの達成度を表現する場合が多い。連続表現モデルはすべてのプロセスエリア(22PA)に関して高水準レベル（CL5）を目指すことができる。

（PDCA）する必要があります。
現状では、プロセスマネジメントモデルの適用が一部の組織に偏っていたり、QCD を安定させるために各種手法や管理指標を十分に用いていないなど、定量的なマネジメントとして定着していないのが実態です。

1.3.3 定量的なマネジメントに結びついた手法の導入

一般的に、ソフトウェア開発にかかわる手法や技法は、開発プロセスマネジメントとの関係を希薄にしたまま取り扱う傾向があります。しかし、開発プロセスマネジメントモデルにおける定量的なマネジメントは、ソフトウェア開発の効果（QCD）を上げるために、最適な各種手法や技法を使用することが求められており、その証跡が重要になります。
本ガイドブックで紹介するソフトウェア品質にかかわる予防活動および検知活動での各種手法や技法は、これを前提に取り組んでいます。

第2章　高信頼ソフトウェアとは

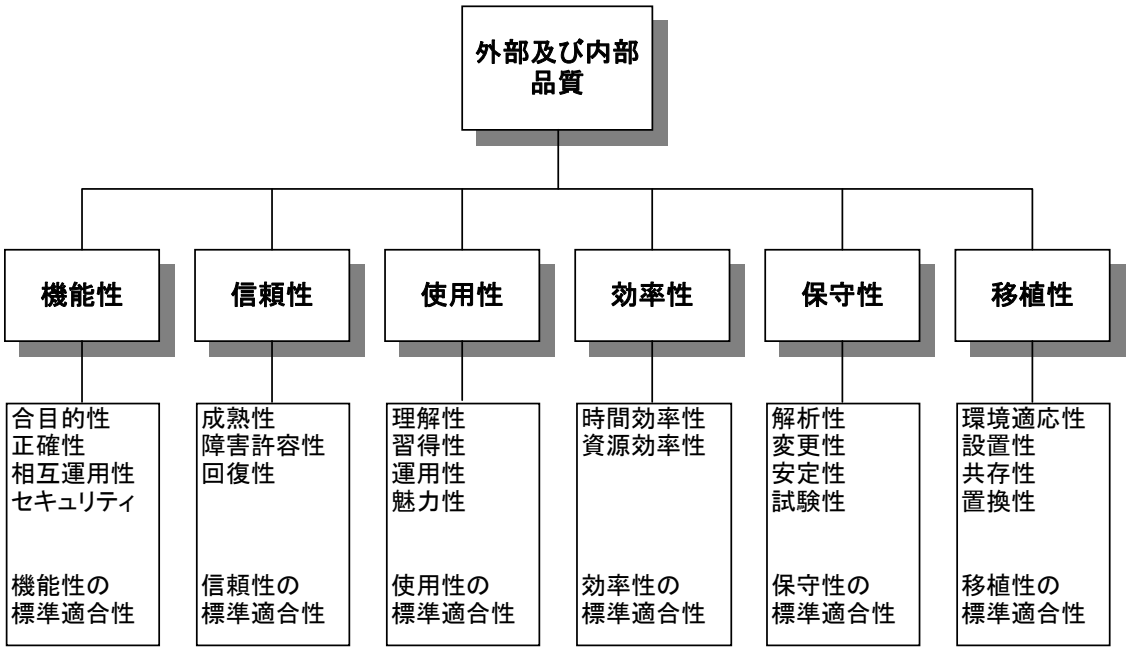
2.1 高信頼ソフトウェアと品質特性

「高信頼」そのものは概念であって、具体的な数値などで表現されているものではありません。従って、ソフトウェアの「高信頼」を明確に定義するためには、定性的な定義の他に、品質にかかわる代用特性などを使用して定量的な尺度の定義と計測を実施することにより把握する必要があります。

ソフトウェアの品質にかかわるJIS規格（JIS X 0129-1:2003）によれば、ソフトウェアの品質「内部品質、外部品質」は、6つの品質特性と27の品質副特性から構成され定義しています。その中の「信頼性」は6つの品質特性（内部品質）の1つであり、4つの品質副特性（成熟性、障害許容性、回復性、信頼性標準適合性）から構成されています。

本ガイドブックでは「高信頼」を「高品質」ととらえ、要求品質の網羅性および水準を的確にとらえるため、「信頼性」を含む6つの品質特性と27の品質副特性を対象としています。なお、ソフトウェア品質の品質特性を

図 2-1に示します。



（出典）JIS X 0129:2003(ISO/IEC 9126:2001)

図 2-1　内部および外部品質のための品質モデル

2.2　システムプロファイルに基づく重要インフラ情報システム

(1) 信頼性要求水準に応じた品質担保

情報システムおよびその構成要素の1つであるソフトウェアは、運輸、電力・ガス、通信、金融および医療などの公共性を有する事業領域の業務から一般的な企業の社内業務まで、様々な範囲に利用されています。

業務にとって情報システムがどのような役割を担い、それに応じてどの程度の信頼性を求められるかは、対象とする業務の特性と情報システムの依存度によって異なると考えられます。つまり、信頼性の側面から考察するとシステム全体を一律の重要度でとらえるのでなく、情報システムごとに重要度を評価して、重要度に応じた信頼性要求水準を決定しソフトウェア品質を担保する必要があります。

しかしながら、一般的には情報システムごとの特性、各システムが有する信頼性側面に関するリスクおよびコストとのトレードオフなどのとらえ方に関して、かならずしも客観的な指針として提示されていないのが実情です。

一部の先進的な企業では、情報システムごとに重要度（可用性「障害影響リスク」、機密性「機密漏えいリスク」、完全性「財務歪曲リスク」）の判定と評価などを勘案して信頼性要求水準に応じた品質を担保しています。なお、本ガイドブックの第2部（事例編）の第2章「東京海上日動システムズ株式会社」、第5章「株式会社日立製作所」などで紹介しています。

(2) システムプロファイルの定義

既に2.2節の(1)で示したように、情報システムの信頼性はシステムおよびソフトウェアコンポーネントごとに重要度を評価して、重要度に応じた信頼性要求水準を決定しソフトウェア品質を担保する必要があります。その際、当該情報システムがどの程度の信頼性要求水準なのかを、「システムプロファイル」を取り入れて客観的に評価するのも、1つの方法ととらえています。

この「システムプロファイル」の考え方は、ソフトウェアの信頼性を議論するうえでの出発点となる考え方ですので、以下に「システムプロファイル」に関する事項を紹介します。

2006年に経済産業省が公表した「情報システムの信頼性向上についてのガイドライン」において、3段階のプロファイルが示されました。

- 重要インフラ等システム
  他に代替することが著しく困難なサービスを提供する事業が形成する国民生活・社会経済活動の基盤であり、その機能が低下または利用不可能な状態に陥った場合に、我が国の国民生活・社会経済活動に多大の影響を及ぼすおそれが生じるもの、人命に影響を及ぼすものおよびそれに準ずるもの。

- 企業基幹システム
  企業活動の基盤であり、その機能が低下または利用不可能な状態に陥った場合に、当該企業活動に多大の影響を及ぼすおそれが生じるとともに、相当程度の外部利用者にも影響を及ぼすもの。

- その他のシステム
  重要インフラ等システムおよび企業基幹システム未満の水準のもの。

さらに、2008年に経済産業省、情報処理推進機構（IPA）および日本情報システム・ユーザー協会（JUAS）共同の「重要インフラ情報システム信頼性研究会‘平成20年度’」

で、詳細な定義がなされました。ここでの「システムプロファイル」は、「信頼性」への「要求水準」を同定し、その要求水準に沿って情報システム・ソフトウェアのライフサイクルへの信頼性水準となるべきものという意味でとらえています。

以下にその「システムプロファイル」に関する定義を紹介します。

この「信頼性」への「要求水準」としては、人命への影響度（システム故障により何らかの形態で人命への損害を与える影響度）、経済的な影響度（各事業者の機会損失を含めたシステム故障が引き起こす経済的損失の影響度）、および公共性への影響度（システム故障が引き起こした場合の国民への公共サービス提供の面からの影響度）の側面から、4段階のレベルを設けています。

図 2-は、分類分析の要因（人命への影響度、経済的な影響度、公共性の影響度など）を明確にして、情報システム・ソフトウェアに対するシステムへの信頼性要求水準（TypeⅠからTypeⅣの4段階）と、その要求水準が情報システム・ソフトウェアに適用される場合の「システム」への信頼性要求水準（システムプロファイル）を示したものです。

図 2-は、「システムプロファイル」にかかわる様々なTypeの判断とその情報システム・ソフトウェアの信頼性要求水準実現のための手法を示したものです。信頼性要求水準を達成するための情報システム・ソフトウェアの実現においては、重要インフラ事業者における、リスク受容・回避・予防の判断を実施後、最終的な「システムプロファイル」を決定する必要があることを示しています。

(3) 高信頼ソフトウェアを対象とする重要インフラ情報システム

高信頼ソフトウェアの対象は運輸、電力・ガス、通信、金融および医療などの公共性を有する事業領域の業務であり、「システムプロファイル」を適用すればTypeⅢおよびⅣになります。しかしながら、本ガイドブックでは、個々の一般企業での重要度評価（障害時の損害金額やお客様迷惑度指数の多寡など）において、重要インフラ情報システムとして判定された情報システムのソフトウェアも高信頼ソフトウェアの対象としています。

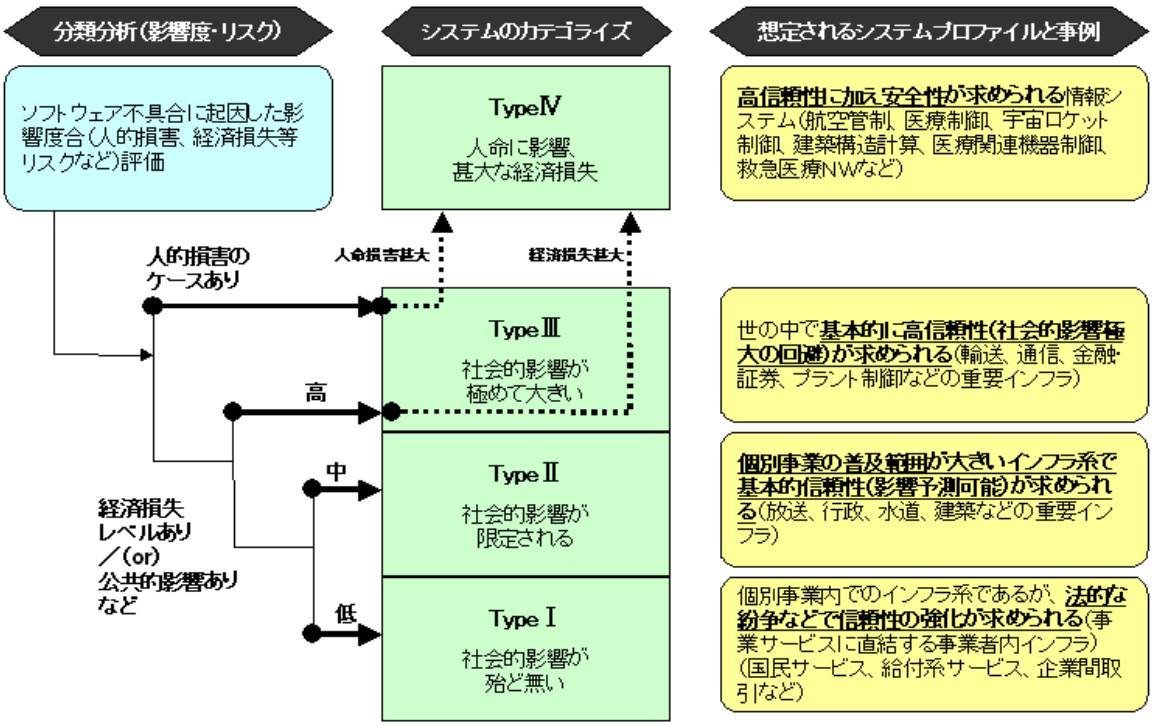


図 2-2　システムプロファイルについて

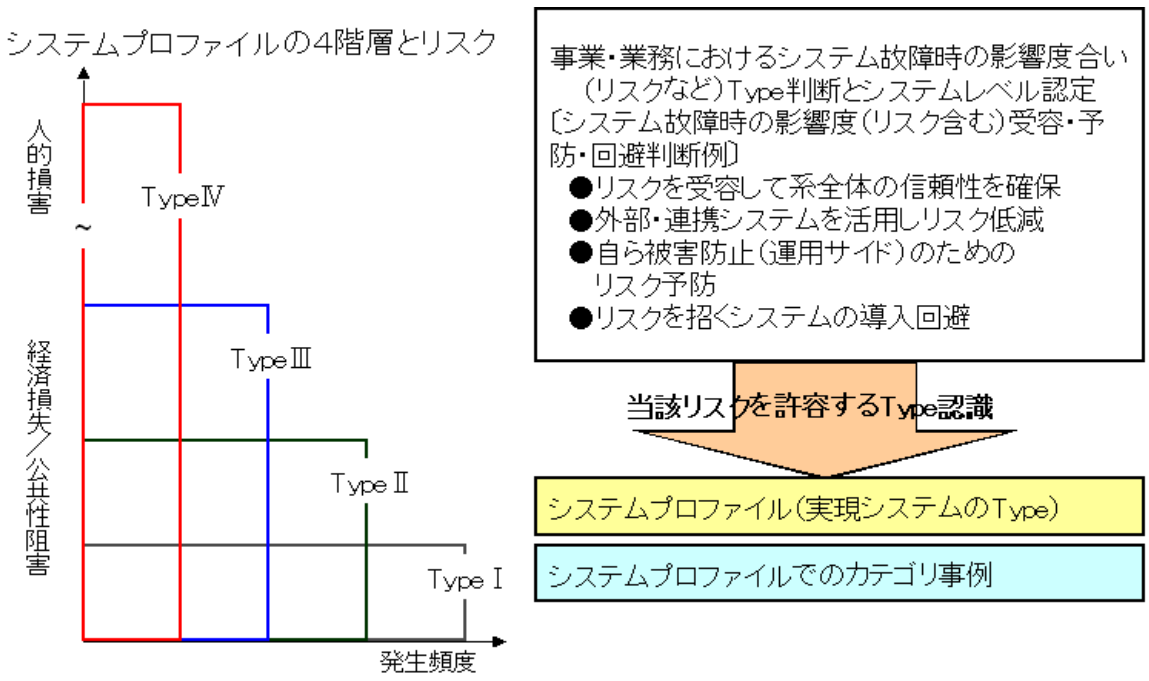


図 2-3　システムプロファイルの考え方

2.3 予防活動と検知活動を対象にした手法

本ガイドブックではソフトウェア品質にかかわる予防活動と検知活動での各種手法および技法を対象にしています。予防活動と検知活動は、一般的な開発プロセスモデルのプロセス領域として、「要件開発」、「要件管理」、「妥当性確認」、「検証」にあたるものです。

本ガイドブックでは、あくまでも手法および技法を中心に紹介しますが、定量的なマネジメントと連携しプロセスを安定化させ、さらなる改善を通してプロセス成熟度を高水準へと向上させることにより、高信頼のソフトウェアを実現することを期待しています。

定量的なマネジメントとソフトウェア開発での各種手法や技法との関係、および必要性は 1.3.3項で述べた通りです。

なお、ソフトウェア開発でのプロセスマネジメントモデルのフレームワーク、および定量的なマネジメントに必要になる管理指標（信頼性要求水準に関係する「ものさし」）に関しては、一部のソフトウェア・エンジニアリング分野で考え方や実践内容が提案されていますので、その報告書および専門書を参照してください。

< 空欄 >

第3章　予防活動および検知活動にかかわる手法

品質保証の活動*は、予防、検知および修正に大別され、品質保証活動を構成する個々の手段（予防活動、検知活動および修正活動）相互の関係が重要になります。

図 3-1にその関係を示します。

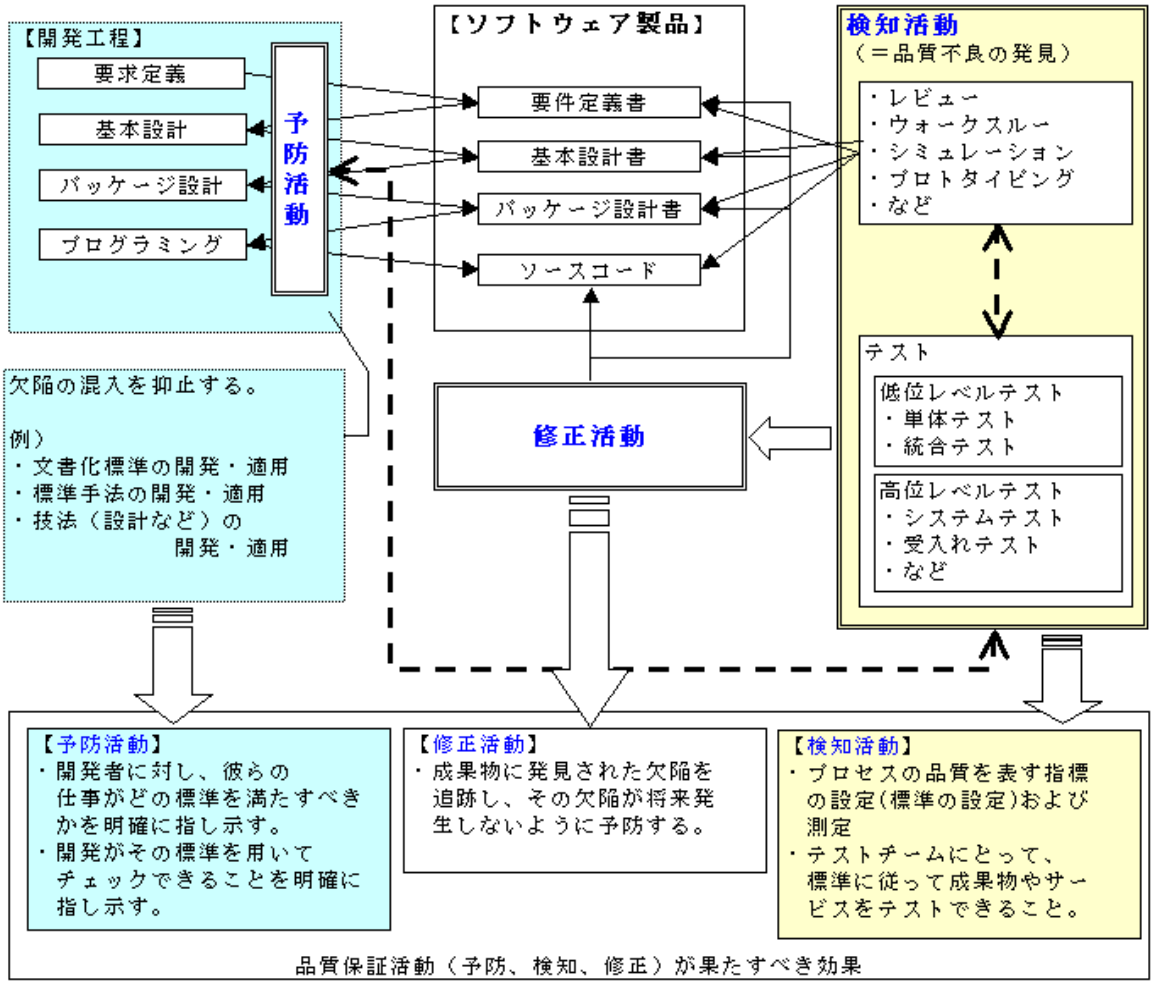


図 3-1　品質保証活動における予防活動、検知活動および修正活動の位置づけ

特に予防活動と検知活動は、ソフトウェアの信頼性を担保する上で、極めて重要な活動です。医療の世界では、痛みがなければ健康で、症状が出てから対処すればよいといった対処療法でなく、普段からの規律正しい生活による予防活動や過去の病理学症例からの仮説検証に基づく予防活動などが重要になります。同じようにソフトウェア品質面での予防活動は、予防活動そのものを目的とした手法や技法は稀少ではありますが、開発標準の構築と地道な遵守活動、さらには過去の障害事例からの仮説検証に基づく予防活動などが重要になります。

---

* ISO 8402 では、品質保証を「あるものが品質要求事項を満たすことについての十分な信頼感を供するために、品質システムの中で実施され必要に応じて実証される、すべての計画的かつ体系的な活動」と定義されています。

品質と健康は、どちらも「完全に問題のない状態」を明確に証明することは困難であるといわれています。従って、検知活動は検知の網羅性を高めつつ、現実的にはコストおよび開発期間などの制約を勘案し、合理的なレビューおよびテストにかかわる手法を選択することが必要になります。

3.1 予防活動にかかわる手法の一般的な事項

予防活動は、要件定義書、設計書およびソースコードなどソフトウェア開発の各種成果物の作成段階において、そもそも欠陥を混入させない技術や手法を指します。従って、要求工学で取扱われている「要求の導出」、「要求の記述」および「要求の検証」ならびに「要求の管理」に関する技術や手法、さらにはソフトウェアの設計技術などソフトウェア・エンジニアリングの多くの技術や手法が広く該当します。本書では、これらの中からソフトウェアのトレーサビリティの管理に関する手法を取り上げます。

ソフトウェアのトレーサビリティの管理は、ソフトウェアの要件定義、設計・開発およびテストの局面を通して一貫して行うものです。まずは合意された要求を識別し、かつ要求の内容を把握して、以降指定された要求に対する変更を管理するとともに、要求間の関係や要求とソフトウェア構成品との関係をトレーサビリティとして管理します。これは、ソフトウェアの品質保証活動と密接に関連しています。

これらを統合した取り組みは「JIS Q 9025:2003 マネジメントシステムのパフォーマンス改善－品質機能展開の指針」として規格化されています。品質機能展開（Quality Function Deployment：QFD）とは、設計・開発の源流から始まるすべてのプロセスで品質を確保する「製品開発の品質保証（Quality Assurance：QA）」のための具体的方法を提供するものです。まず顧客†が満足する設計品質を設定し、次にその設計の意図と品質保証上の重点を製品の生産段階まで展開します。これは製品の品質を確保する方法として、内外を問わず各企業の製品開発や品質保証を行う上で広く活用され、成果が認められてきています。

品質機能展開の構想図の一例を、図 3-2 に示します。規格で定める品質機能展開は、品質展開（Quality Deployment：QD）と業務機能展開の総称です。品質展開とは「顧客の要求を代用特性(品質特性)に変換し製品の設計品質を定め、これを各種機能部品の品質、さらに、個々の部品の品質や工程要素に至るまで、これらの関係を系統的に展開していくこと」であり、業務機能展開とは「品質を形成する機能ないし業務を系統的にステップ別に細部に展開していくこと」です。「品質展開」と「業務機能展開」とは「保有技術展開」を媒介して結合されています。

ここでいう保有技術とは実際に企業が保有もしくは必要としている技術であり、技術表現‡を用いて技術を言語情報として展開したものが、技術展開表です。技術展開表を媒介として、品質展開と結合して、現状の技術で要求品質を満足できるのか、設計品質を確保できるのか、機能を実現することができるのか、目標原価で製造することができるのか、信頼性を確保することができるのか、ということが確認できるようになります。

本節では、図 3-2 に示す品質機能展開の全体構想図のうち、ソフトウェアのトレーサビリティの管理とかかわりが深い品質展開を取り上げます。

---

† 本節では、エンドユーザに限定しない様々なユーザの総称として「顧客」を用いる。

‡ 「市場を調査する技術」、「市場情報を分析する技術」など。

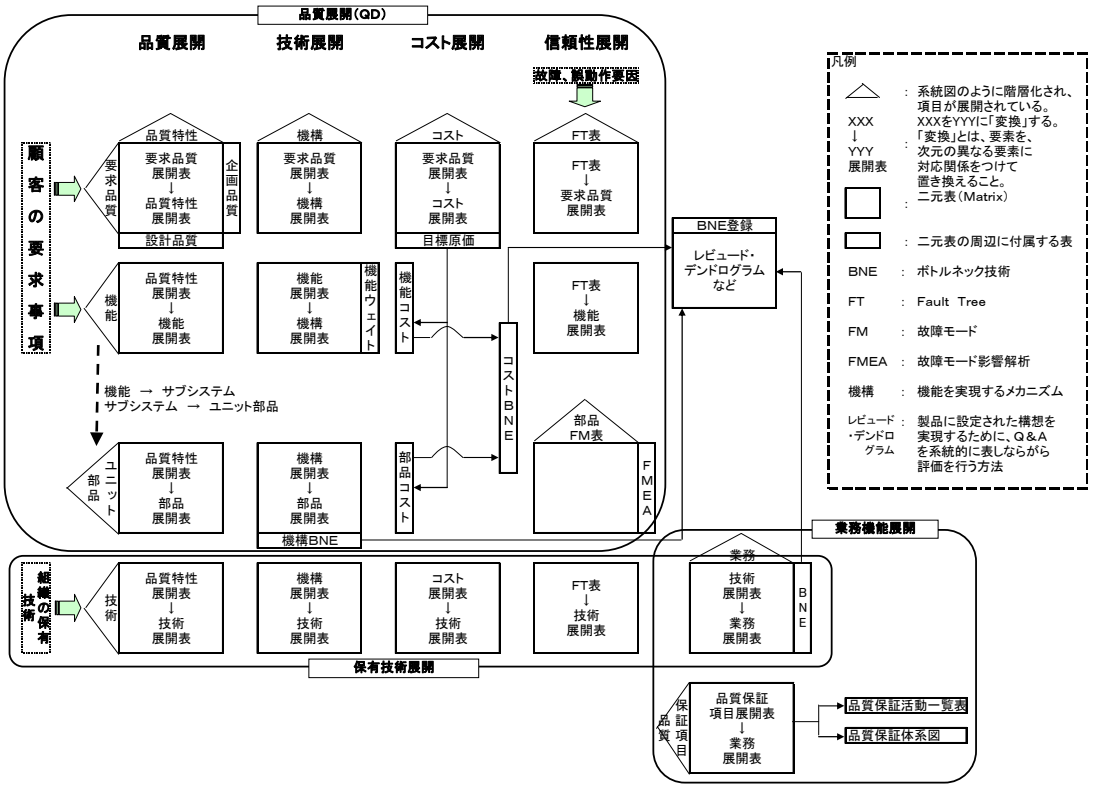


図 3-2　品質機能展開の全体構想図（例）

品質展開の主要なツールには顧客の要求を展開した要求品質展開表と製品を提供する側の技術的な品質特性を展開した品質特性展開表およびこれらの二元表である品質表とがあります。

二元表とは2要素間のトレーサビリティマトリクスであり、表 3-1に要求品質展開表と品質特性展開表との二元表の例を示します。2つの要素（ここでは要求品質要素と品質特性要素）間の関係を対応関係と呼び、対応関係の強さから表に示すように◎、○、△の記号をつけて対応関係を示します（空欄は対応関係がないことを示します）。◎、○、△の各記号には図に示すように重みを割り当てます。

ソフトウェア開発では機能と成果物、もしくは成果物と成果物間の定められた粒度での対応付けとしても利用されます。これがあると、プログラム理解、ソフトウェアの変更時の影響範囲追跡、障害発生時にお客様へのビジネスインパクト分析などができるようになり、例えばCMMIのプロセスエリアの1つであるREQM（要件管理）では絶対に必要となる機能になります。

表 3-1 二元表の例

品質特性展開表 / 要求品質展開表		1次	操作性				ソフト充実度				形状寸法				質量			話題性			
		2次	接続時間	メモリ容量	CPU速度	携帯性	ソフト互換度	ソフト拡張性	キャラクタ充実度	ソフト多様性	本体の厚さ	外形寸法	操作部寸法	開口部寸法	本体質量	操作部質量	附属品質量	意匠性	安全性	注目度	リアル度
1次	2次																				
使いたくなる	面白い							○	◎									○		○	○
	会話できる							○													
	体感できる								○				○			△					◎
	デザインが良い											◎						◎		○	
ソフトが良い	どのソフトも使える						◎			◎	○										
	ソフトが多く入る			○				○													
	ソフトが作れる						○	◎													
長く楽しめる	多人数で楽しめる							○												◎	
	若者が好む					○												◎		◎	○
	長く使える			○	△					◎											
頑丈である	水に強い													○					○		
	ほこり(埃)に強い													◎							
	熱に強い											○							◎		
使いやすい	接続しやすい		◎			○								○							
	コードがない		○			○															
	持ち運べる					◎					○	○			◎		◎				
高性能である	音質が良い																				◎
	ロードが早い				◎					○											
	画像がきれい			○				○													○
操作しやすい	簡単にセーブできる			○	○					○											
	ボタンが押しやすい													◎		○					
	片手で操作できる													◎		○	○				

◎(重み＝5):強い対応がある
○(重み＝3):対応がある
△(重み＝1):対応が予想される

(出典) JISハンドブック2008　品質管理

品質展開は、新製品の開発を効率よく進めるために適用する場合もあれば、確実な品質保証を行うためにも利用されます。品質展開をどのように適用するかということは、これを実施する人たちが決めることであり、目的によっては様々な使い方が可能な方法論とされています。

品質展開は図3-2に示す通り、品質展開、技術展開、コスト展開および信頼性展開から構成されています。

品質展開は、製品の企画・開発段階から製品に顧客が要求する品質を作りこむことを目的としています。技術展開は、品質特性展開表で展開した代用特性ごとに戦略的に設定する設計目標値に対して､最も経済的に実現するための固有技術を具体的に割り付け、不足している固有技術を「ボトルネック技術（BNE）」として抽出し、段取りよく解決していくことを目的としています。また、コスト展開は経済性（価格競争力）を企画・開発段階から保全することを目的としたものです。

品質展開では、顧客の要求を源流として顧客が要求する品質、つまりポジティブな品質を製品に作りこむことに主眼を置いていますので、故障などのネガティブな要素を完全に押さえ込むことが困難となります。

最後の信頼性展開は、FTA（Fault Tree Analysis）、FMEA（Failure Mode and Effect Analysis）を活用して、故障に着目し機能が損なわれることを防ぐことを目的としています。

信頼性展開の具体的な取り組みは「第4章 障害事例から学ぶ予防活動」で取り上げています。

このように、品質展開では目的によって様々な展開表を作成します。例えば、要求品質展開表、品質特性展開表以外に、機能展開表、機構展開表、ユニット・部品展開表、技術展開表、コスト展開表、FT展開表、FM展開表、業務機能展開表などの展開表を作成して活用します。

これらの展開表のなかで、目的を達成するのに必要な表は何かを明確にし、二元表を作成する目的、二元表によって何が得られるかを事前に検討する必要があります。また展開表は一種の一覧表ですが、目的によって作成すべき表は細部にわたっての展開表が必要な場合と、単なる一覧表でも十分な場合があります。

第 5 章では、製品の企画・開発段階から製品に要求される品質を作り込むことを目的とした品質展開に焦点を当てて、ソフトウェアの信頼性を担保するいくつかの具体例を紹介します。主な内容は 3 つあり、1 つ目は二元表をソフトウェアのトレーサビリティに活用して、ユーザの要求を代用特性（品質特性）に変換し、完成品の設計品質を定めることです。2 つ目は完成品の設計品質を各種機能部品の品質、さらに個々の部品の品質や工程要素に至るまで、これらの関係を系統的に展開することです。3 つ目は、これらのトレーサビリティの充足性をソフトウェア開発の様々な段階の設計検証において確認することです。

3.2 検知活動にかかわる手法の一般的な事項

ソフトウェア開発にかかわる検知活動は、上流工程での品質レビュー（デザインレビュー等）と下流工程におけるテストが代表されます。さらに検知内容から分類すると、検知活動には「妥当性確認」と「検証」が存在します。ANSI/IEEEでの「検証」を要約しますと、「ソフトウェア開発ライフサイクルの各開発工程での成果が、その前工程で確立された要求事項について、満たしているかどうかを決定する過程」と示されています。また、「妥当性確認」とは「ソフトウェアの要求事項に従属しているかどうかを決定する過程」と示され、「検証」と「妥当性確認」を区別してとらえています。

品質レビューは上流工程の要件定義・基本設計になるほど、およびテストは下流工程のシステムテストになるほど「妥当性確認」（品質特性での合目的性など）の要素が強くなり、プログラム製造の品質レビューおよび単体テストは「検証」（品質特性での正確性など）の要素が強くなります。

例えば、単体テストでは不要機能の存在や構造化の失敗などを検出しようとすると、検出不可能ではないですがコスト高になります。

品質レビューとテストの相違について以下に示します。

- 検知活動の範囲から考察しますと、品質レビューは成果物（文章、各種構造図、入出力形式、コードなど）の記載内容を静的に評価し、問題点に対する改善を即対応できる点にあります。一方、テストは実装環境下で成果物の動作を実際に評価できる点にあります。
- コスト面での評価は、検知コスト（実装環境具備コスト、人件コストなど）および欠陥除去コスト（正味棄却コスト§）を考慮する必要があります。特に、下流工程のテストは、正味棄却コストが高くなります。

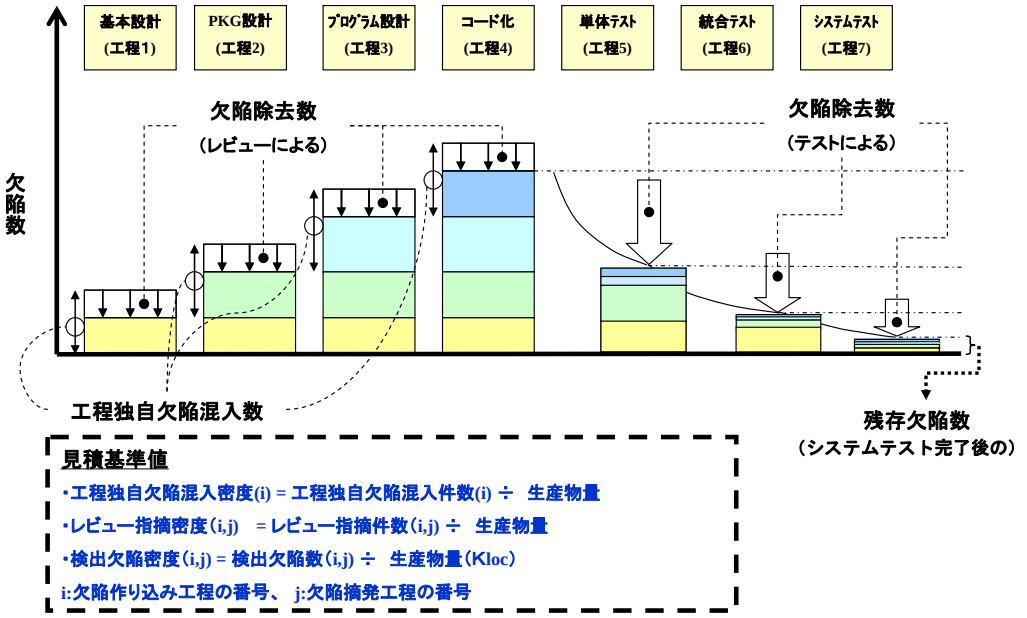
§ 正味棄却コストとは、仕様変更（欠陥修正の場合は錬度が高い改造型）などの修正行為により、棄却対象となる作成済み工程生産物量に対応するコスト。IPA/SEC「ソフトウェアテスト見積もりガイドブック」より

3.2.1　レビューおよびテストでの欠陥検出戦略の統合

ソフトウェアに混入する欠陥の総量を見積り、レビューを含めて各開発工程で欠陥を検出する数を想定することが重要です。

設計・実装工程から品質を作り込んでいれば、テスト工程で検出すべき欠陥は少なく、テスト工程でのコストも少なくなる傾向があります。従って、設計・実装工程のレビューとテスト工程の品質確認テストの両面で欠陥数を測定して管理すべきで、テスト実施工程のみを対象にした過去の統計値（信頼度成長曲線などの品質目標値）を基にテスト見積りを行う場合は、そのことを留意する必要があります。

なお、設計･実装工程とテスト工程での欠陥除去数と残存欠陥数との関係を図 3-3 に示します。



レビューでは当該工程で作り込んだ欠陥の除去だけでなく、前工程で作り込んだ欠陥を除去することもある。

（出典）ジャステック社「ソフトウェアテスト見積り－欠陥の混入および除去モデル－」

図 3-3　設計・実装工程とテスト工程での欠陥除去数と残存欠陥数との関係

ソフトウェアの欠陥は、それぞれの設計・実装工程で作り込まれ（レビューにより一部が除去されますが）、作り込まれた欠陥は次の工程に引き継がれます。単体テスト前のプログラムには、基本設計工程～コード化の各工程で作り込まれた欠陥が集積されています。

テストでは、設計・実装工程でプログラムに作り込まれた欠陥を除去していきますが、すべての欠陥を除去することはできないので欠陥が残ります（図 3-3 では、これを残存欠陥数と呼んでいます）。

それぞれの設計･実装工程で作り込む欠陥の密度およびレビューなどで除去できる欠陥の割合、ならびに各テスト工程で除去できる欠陥の割合は、出荷後に検出される欠陥も含めて測定して蓄積し、基準値として精錬しておくことが重要です。

3.2.2　レビュー手法の概要

一般的に、ソフトウェア開発にかかわるレビューは大きく2つに分類されます。1つ目はプロジェクト管理を推進するためのマネジメントレビューやプロジェクト完了レビューなどです。2つ目はソフトウェア開発成果物そのものの品質に主眼を置いたレビューがあ

ります。本項では、2つ目の直接的にソフトウェア開発成果物の品質に主眼を置いたレビューを対象にします。

レビューの効果に関して、前項での欠陥除去コスト（正味棄却コスト）の低減以外に、定性的な効果として開発に必要となる知識の共有や技術移転促進などが上げられます。

レビュー技法には公式および非公式を含め多くの種類が存在しますが、特に高信頼ソフトウェアを考慮する場合、インスペクションは重要な技法になります。

実用面ではソフトウェアライフサイクルの各工程で、プロジェクト特性との整合およびレビュー手法の特徴をとらえたレビュー技法の選択、ならびに自組織にあったやり方をカスタマイズし構築・導入して行くことが必要になります。

図3-4に品質レビュー技法のポジショニングマップを示します。さらに、各レビュー技法に関する内容を (1)から(8)で紹介します。なお、図 3-4についてはソフトウェアテストPRESS VOL1.2「細川宣啓：さまざまな上流品質レビュー技法」を参考に作成しています。

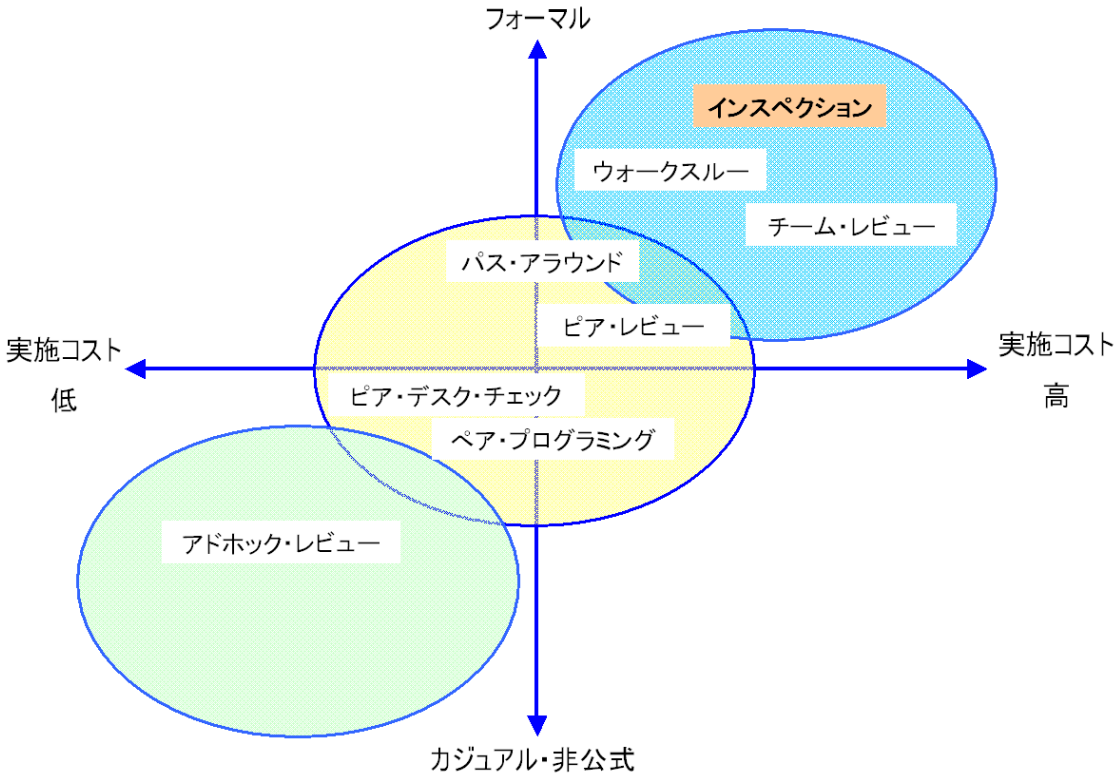


図 3-4　品質レビュー手法のポジショニングマップ

(1) アドホック・レビュー

非公式なレビュー手法で、文字通りアドホック（即席に、その場かぎりに、あるいは場当たり的に）に行うものです。我々は作業を行う場合に、ひとりで黙々と行う場合も無いとはいえませんが、普通は隣の席など近隣の同僚に「これってどうするのだっけ？」と問いかけ、「それならこうするのだよ。」というような、ちょっとした会話を行います。

この会話は単なる雑談の場合もありますが、作業に関するものも含まれています。このちょっとした会話は作業時間のうち、作業に関するものがかなりの部分を占めているという報告もあります。アドホック・レビューとはこのような、作業に関するちょっとした会話を指します。アドホック・レビューはいつ実施するかという明確な取り決めはありません。レビュー内容を記録するための形式も定められていることはほとんどあり

ません。
アドホック・レビューではその性質上、担当者の目前にある問題を解決することしかできず、ソフトウェアシステム全体から見た問題のようなものを解決することには不向きです。また非公式で記録に残すこともないため、メンバ間における問題の共有、技術の共有/移転などもそれほど期待できません。

(2) ペア・プログラミング

もともとはエクストリーム・プログラミングというアジャイル型開発手法のプラクティスの1つで、厳密に言うならばレビュー手法というよりは開発手法に属するものです。
ペア・プログラミングでは 2 人が 1 台の PC を共有し、1 つのプログラムを開発します。このとき 2 人のうちどちらか一方がプログラムを入力し、これをドライバと呼びます。もう一方はナビゲータと呼び、入力されているプログラムを確認しながら指示を与えます。(ナビゲータという呼び方のほかに、オブザーバやパートナーという呼び方もあります。役割の微妙な違いにより呼び分けているようです。) ドライバとナビゲータの役割は固定ではなく適宜交代します。交代する間隔は 5 分から長くても 30 分といわれています。またペアも固定ではなく、1 日程度で交代するほうが良いとも言われています。
ペア・プログラムの利点として以下のようなものがあるといわれています。

- 複数人の技能や知識を統合することが期待でき、結果としてプログラムの品質が高くなる。
- 1 つのプログラムに対し、必ず最低 2 人の担当者が存在することになり、要員配置変更や事故により、どちらか一方がいなくなっても作業が続けられる。
- プログラミングが共同作業になるためサボりにくくなり、作業が進む可能性が高い。
- ペア間のコミュニケーションを促進し、知識や技術の共有化が計れる。
- プロジェクトの全員がペア・プログラミングに参加し、頻繁にペアを代える場合、プログラム全体に対する知識を全メンバで共有することができる。

一方ペア・プログラムの欠点として以下のようなものがあるといわれています。

- プログラム作成時には1人で集中して行う方が能率的だと考える人が一般的であり、ペアでの作業を面倒だと考える傾向がある。
- 能力差のあるペアの場合、能力の高い方から低い方への一方的な指示になりかねない。
- 能力差のないペアの場合、それぞれの能力以上の良いプログラムを作ることが出来ない可能性が高い。
- 頻繁に役割やペアを代える事が、かえってプロジェクト全体やプログラム全体への関心を薄くさせる。(目の前にあるプログラムだけを考えてしまうことになりかねない。)
- ペアがいることに甘えて他人任せになり、結果として責任感が希薄になる。(ペアがいるから自分がやらなくても良いと考えるメンバが出てくる。)

冒頭でも述べたように、ペア・プログラミングは厳密に言うと開発手法の一部であり、開発チームの運営方式に関する文化的な変化ととらえるべきものです。いわゆる旧来のレビュー手法を置き換えるものではない点には注意が必要です。

(3) ピア・デスク・チェック

ピア・デスク・チェックはバディ・チェックまたはペア・レビューといわれることもあります。プログラム作成者とレビュー担当者の 2 名だけで成果物を調べる方法です。

ここではピア・デスク・チェックと後述のピア・レビューを分けて説明していますが、同一と見なす場合も多くあります。

レビュー内容や結果を成果物として残すか残さないかは運用によります。成果物を残す場合は比較的公式な手法になり、逆に成果物を残さない場合には非公式な手法になります。ピア・デスク・チェックはアドホック・レビューを除き、レビューの最も基本的な手法です。一般的にピア・デスク・チェックは、作成者がレビュー担当者にレビュー依頼をし、レビュー担当者がレビューを行った後、結果を作成者に伝える形をとります。

ピア・デスク・チェックの最も特徴的な利点は、追加コストがレビュー時に使われるレビュー担当者の時間だけであり、比較的公式なレビュー手法のなかでは最も少なくて済む点です。このため欠陥を発見する能力が十分に高いメンバがいる場合、そして時間とリソースの制限が非常にきついプロジェクトである場合、最も有効な選択肢になります。

問題点としては、レビュー担当者が作成者に一方的に指示を与えることになりかねず、作成者の欠陥を見つける能力を伸ばすという教育的側面が小さいという点が上げられます。これを避けるためレビュー時にレビュー担当者と作成者を同席させ、質疑応答などを交えながらレビューをすることも良く行われます。2 人が同席する場合を、特に「2 人インスペクション」と呼ぶ場合があります。この場合、後述のピア・レビューとの違いはほとんど無いといえます。

もう 1 つの問題点として、レビュー担当者の欠陥発見能力に強く依存することが上げられます。レビュー担当者の能力が十分で無い場合、いわゆる「ざる」の状態となり、レビューの効果は甚だしく低くなります。

(4) ピア･レビュー

ピア･レビューはプログラム作成者と（場合によっては複数の）レビューアがレビューする手法の総称です。一般的に言えばピア・レビューは学術誌に論文を発表する際に行われる査読を指します。ここでいうピアとは「仲間」や「同僚」といった意味であり、同じ研究分野の専門家、研究仲間のことです。すなわちピア・レビューは同一分野の専門家である同僚による吟味や調査、検証を指すことになります。この定義に従うなら、ここで取り上げたレビュー手法はすべてピア・レビューということになります。事実ピア・レビューの定義や説明は文献により若干の差があるものの、前述のアドホック・レビュー、ピア･デスク･レビューを含み、また後述のウォークスルー、チーム・レビュー、インスペクションも含みます。これを広義のピア・レビューと呼ぶことにします。広義のピア・レビューの特徴は、前述および後述の各手法を参照してください。

広義のピア・レビューに対して 2 つの視点から狭義のピア・レビューが説明される場合があります。1 つめの視点は知識や技術の共有を目的にするかしないかという視点です。前述のアドホック・レビュー、ピア・デスク・レビュー、そして後述のパス・アラウンド、ウォークスルーの各手法は目前の問題点に対する解答を得ること、あるいは目前にある成果物の欠陥を除去することが主たる目的であり、メンバ間における問題の共有、技術の共有/移転、コミュニケーションの促進は目的としていません。一方で狭義のピア・レビューではこれらを目的の一部とします。

もう 1 つの視点は公式か非公式かという視点です。後述のチーム・レビュー、インスペクションは公式な手法に属し、通常は成果物を残します。これに対し狭義のピア・レビューは必ずしも成果物を残すとは限りません。(成果物を残さないとも言い切れません。) また、公式なレビュー手法はほとんどの場合、作業が終了した後に行うのに対し、狭義のピア・レビューは作業中にも行います。

これ以降は狭義のピア・レビューについて説明します。ピア・レビューの参加者は役割を分けることが推奨されます。役割は以下のようになります。

- 担当者
  レビュー対象の作成者でレビューを受ける人です。
- まとめ役
  レビューの司会・進行を行う人です。最小構成では担当者とまとめ役の 2 人でレビューを行うことになります。この場合、前述のピア・デスク・レビューで説明した「2 人インスペクション」とほぼ同じになります。
- レビューア
  レビューしてくれる人です。多くの場合は同僚になりますが、他のグループの有識者かもしれません。
- 記録係
  レビューで取り上げられた課題や結論を記録する人です。これは担当者、まとめ役、レビューアの誰かが兼任してもかまいません。

レビュー参加者は最小構成で 2 人ですが、可能であれば担当者、まとめ役、レビューアの 3 人いることが望まれます。レビューアによる第三者の視点からの指摘は、担当者が持ちやすい思い込みによる欠陥の混入を発見するために非常に効果的です。また、まとめ役を置くことにより、些細な問題に対してムダに時間を掛けることのないように、そしてより重要な問題に集中するよう、レビューの進行を整理することが出来ます。レビューアの数を増やせばより客観的で網羅的なレビューを行うことが出来ます。記録係はピア・レビューの目的により置くか置かないかを決めます。

ピア・レビューの長所短所は、目的によって変わります。メンバ間における問題の共有、技術の共有/移転、コミュニケーションの促進を主たる目的とするのであれば、比較的非公式な進行を行います。ただし、レビューで指摘された問題の修正や変更の正しさの検証があいまいになる可能性があります。

目的の変更または欠陥修正の正しさの確認や、原因の分析などを重視する場合、手順を明確にすること、記録を残すことは必須になります。このように公式的に行う場合はそうでない場合と比べ、コストは高くなります。

(5) パス・アラウンド

パス・アラウンドは多重かつ同時進行のピア・デスク・チェック（あるいは狭義のピア・レビュー）だといえます。ピア・デスク・チェックではレビュー者の欠陥発見能力に強く依存しますが、パス・アラウンドでは複数人、場合によっては大多数からのフィードバックを得ることができるため、欠陥を見落とすリスクが減少します。またパス・アラウンドでは、レビュー関係者が一同に介して行う必要が無いため、メンバの招集コストを軽減することも出来ます。またレビュー対象の文書が、例えば変更履歴をもつ文書編集ソフトの場合、変更管理機能やメモ機能を使うこともできるほか、E メールを使用するなどして、遠隔地にいる関係者もレビューに参加することが出来ます。

パス・アラウンドの欠点としては、期間内にレビューを行わない関係者の存在や申し訳程度にしかレビューを行わない関係者の存在があります。どちらも同席しないことや、遠隔地にいるため、精神面での刺激が少ないことが原因といえます。

また複数人が同時期に同席せずにレビューを行うことから、同一の問題点が複数人から指摘されることもあるなど、冗長なコストが発生することもありますし、成果物を変更する前に、複数のレビュー結果を最終的にまとめるというコストが必要になります。これらのコストは多人数になればなるほど大きくなりますが、欠陥の発見数もまた、多人数になればなるほど多くなるというトレードオフの関係にあります。一般的に、どの程度の人数でどの程度の欠陥を発見し、それらを修正できるかの見極めは非常に難しい問題です。

(6) チーム・レビュー

一般的には作業の節目ごとに計画され、手順や成果物が規定されているレビューで、公式なレビュー手法に属します。公式さがインスペクションほど厳密ではないので「軽量インスペクション」と呼ばれることもあります。また構造化されている（手順が定められている）という点から構造化ウォークスルーと呼ばれることもあります。チーム・レビューは他にも様々な名前で呼ばれており、単にレビューといった場合がチーム・レビューを指していることもあります。

チーム・レビューは様々な目的のために利用できます。一般的にチームは同じ目的を共有している集団です。このためメンバそれぞれが密に関係していることが重要になります。このような集団の特性を考慮し、チームのメンバが持つ技術/知識を共有する機会として利用することができます。新しく参加したメンバや熟練していないメンバへの教育の場としても利用できます。チーム・レビューを行うことでメンバ間でのコンセンサスを形成することもできます。当然のことですが成果物に含まれる欠陥を発見するためにも利用できます。

上述のようにチーム・レビュー目的は様々です。例えば新メンバや熟練していないメンバの教育を目的としている場合と、チーム内の合意形成を目的としている場合と、欠陥の発見を目的としている場合では、レビューの事前準備や進行、成果物が異なります。このためレビューの内容と結果が混乱しないよう計画時に目的を決めておくことが重要になります。レビューする成果物の内容は目的に応じて調整することになります。

公式なレビューの性格として、問題や対象成果物の作成者と、進行・まとめ役と、レビューを行う役、そして記録役を分けることが推奨されます。後述するインスペクションほど厳密ではなく、時に作成者と進行・まとめ役が兼任することはありえます。しかし、特に欠陥の発見を目的とする場合、作成者と進行・まとめ役とレビューを行う役が異なる点は重要です。第三者の視点から見たコメントや質問に対して作成者が答え、それをまとめる形式を取ることで、作成者の思い込みによる欠陥や、作成者自身が考えていなかった状況を発見しやすくなるからです。レビューの記録をとることも重要な点となります。レビュー内容を記録した成果物から、修正や変更が正しく行われているかを検証します。

チーム・レビューの長所は先に述べたように、様々な用途に利用できることです。欠点は、レビュー参加者を招集するコスト、レビューに参加する間、メンバを拘束するコストがかかることです。

チーム・レビューは後述のインスペクションのような厳格さ、厳密さを必要としないグループや成果物に対して有効です。

(7) ウォークスルー

ウォークスルーは作成者が他のメンバに対して問題点を説明しコメントを得る形式をとります。ウォークスルーは準公式から全くの非公式まで様々な運用が可能です。単にウォークスルーという場合は、公式に定められた手順は存在せず、レビューを終了する公式な基準や定義もありません。レポートなど成果物も特に定められていません。この意味ではウォークスルーは非公式な手法に属します。しかしウォークスルーを行うにあたり、メンバ間で合意された基準を作ることが禁止されているわけではありません。特にE.ヨードンが提唱する「構造化ウォークスルー」は実施基準や成果物を定めており後述のインスペクションに近い公式なレビュー方法になります。

ウォークスルーが他のレビュー方式と決定的に異なっている点は、作成者が支配的な役割を果たすことです。作成者がまとめ役、司会役を兼ね、自分にとって問題となる点、あるいは悩んでいる点を説明し、他のメンバからの意見を募るので、問題点ははっきりしており、その問題に対する解答は得られやすい傾向があります。

反面、作成者の問題意識に縛られることにより、本来、発見されるべき欠陥が見逃される可能性があります。また異なる問題点としては作成者が説明し司会をするという性格上、

一方的な説明会になってしまう可能性があります。特に作成者が自分の能力に自信を持っている場合『欠陥などあり得ない』という気持ちになることは容易に予測できる展開であり、潜在および顕在する欠陥の発見を阻む障害となりがちです。欠陥を発見するという観点からみると、ウォークスルーの効果は後述のインスペクションに比べ明らかに劣ります。

以上のようなリスクはレビュー基準や視点、レビュー終了条件など、運用基準を設けることで軽減できますが、ブレーン･ストーミング的な自由で解放的な発想を得にくくなるというトレードオフ関係が存在します。

(8) インスペクション

インスペクションは、ANSI/IEEE 標準 1028-1998 において体系が定義されている最も厳格で公式なレビュー手法です。インスペクションは他の手法に比べ、欠陥の発見に最も効果の高い手法といわれており、その目的を次のように定めています。

- 成果物（ソフトウェア要素）が基本仕様、指定された品質属性、顧客要求を満たしているか否かの検証。
- 成果物（ソフトウェア要素）が関連する基準、規制、規則、計画、手順に適合しているか否かの検証。
- 発見された欠陥とインスペクションに使用した時間に関する評価指標の供給。
- 表現方法の良し悪しや文体に関する問題の検査は行わない。

インスペクションの特徴の 1 つに役割をはっきり分けることがあげられます。役割は作成者、進行・まとめ役、記録役、説明役、レビューを行う役があります。特に注意しなければいけないこととして、作成者は兼任できないという点があります。インスペクションでは第三者の目からみた指摘やコメントを受け、これに対して回答することで、作成者の思い込みを廃し、可能な限り多くの欠陥を取り除くことを目指しています。指摘やコメントは記録役により迅速に収集されます。インスペクションの対象にもよりますが、インスペクションでは記録係、進行・まとめ役の負担もかなり高いため、これら 2 つの役も兼任を避けるべきです。

一般的に言えば、進行・まとめ役とレビューを行う役にあたる人物はインスペクションに対する高い教育を受け、十分に経験を積んだ人である必要があります。

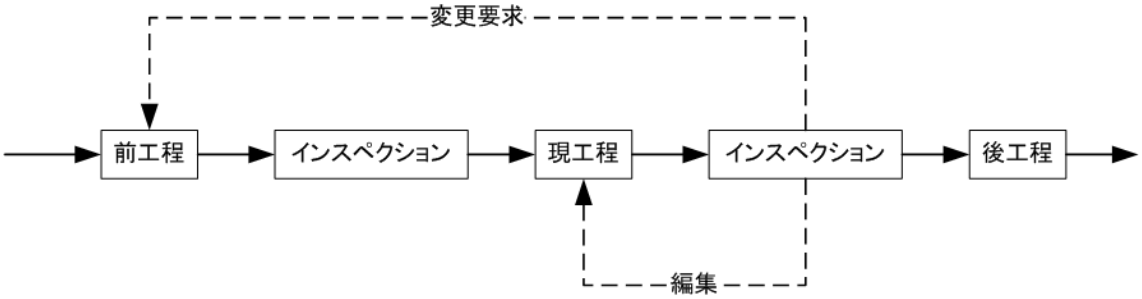


図 3-5　工程とインスペクション

表 3-2 インスペクションでの役割

役割	作業内容等
進行・まとめ役 Moderator	インスペクションの主催者です。 インスペクションを熟知している必要があります。 インスペクション・メンバーの選抜やスケジュール調整、成果物の受領と配布、作業進捗の管理を行います。 最終的には当該インスペクションの結果報告責任者になります。
レビューを行う役 Inspector	欠陥検出の担当者です。当該チームの作業内容について既知であるか否かを問わず、高い欠陥発見能力を持つ人が望まれます。 インスペクションに関する正式な教育を受けていることが必要です。
説明役 Reader	進行に合わせて対象成果物の該当部分を他に説明します。一般的には作成者が担当します。
記録役 Recorder	指摘された欠陥と問題を分類、記録します。
作成者 Author	インスペクション対象の成果物の作成者または保守者です。

インスペクションでは手順も定められており、一般的に次の７つの段階を踏みます。

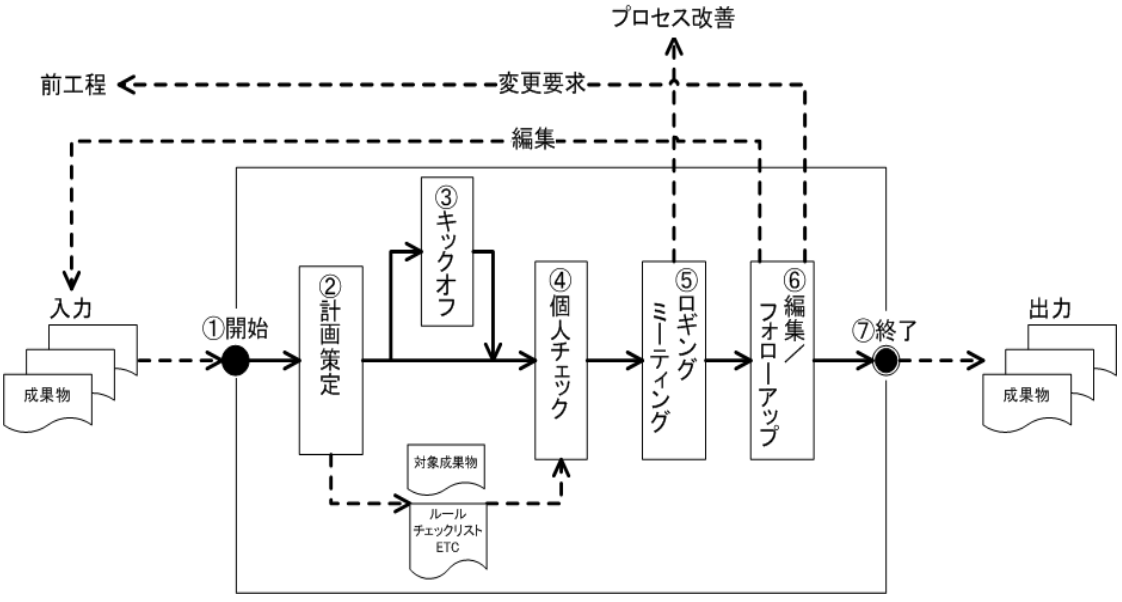


図 3-6　　インスペクションのプロセス

① 開始

インスペクションの開始プロセスでは入力となる成果物を入手します。対象となる成果物は、インスペクション前に作成されたものです。いくつか例をあげるならば「マーケティング文書」「要求仕様」「ユースケース」「ビジネスルール」「アーキテクチャ定

義」「ユーザインタフェース設計/モデル」「データベース設計/モデル」「RFP」「予算見積」「前作業の議事録」などです。一見してわかるように、前作業で作成されたすべての成果物が入力になりえます。この入力となる成果物を入手することが意外に難しいことがあります。

次に、入手した成果物の中からインスペクションの対象になるものを選び出します。無制限に時間があるのであれば、すべての成果物をインスペクションすればよいのですが、限られたリソースのなかでより高い効果を得るためには、効率の良い（欠陥を含んでいると影響の大きい）成果物を選ぶ必要があります。この作業は進行・まとめ役が行いますが、これには非常に高い技能や能力が必要になります。

② 計画

インスペクションを行うメンバを選抜し、メンバが参加できるようにスケジュールの調整を行います。そして各メンバの役割分担を決めます。またインスペクション作業の全体像を定義するとともに、インスペクションの終了条件や中断条件を定義します。計画を行うのは通常、進行役・まとめ役ですが、インスペクションに対する深い理解が必要になります。

③ キックオフ

インスペクションを行うメンバに、全体の期間、準備期間、各自が行う作業指示を行い、インスペクション対象の概要を通知します。また対象成果物の配布を行います。

④ 個人チェック

事前準備とも呼ばれます。配布された資料をあらかじめ調査し、指摘箇所をまとめておきます。個人チェックは次に行うロギングミーティングを効率的に進めるために非常に重要です。

⑤ ロギングミーティング

インスペクションの中核となります。ここでは個人チェックであらかじめ発見された欠陥を指摘・報告するとともに、他の指摘を考慮しながら新たな欠陥をできる限り発見することが目的になります。

⑥ 編集およびフォローアップ

ロギングミーティングで洗い出された問題点や欠陥の修正を依頼します。正しく修正された否かを追跡・確認し、必要であれば再修正の指示を出します。特に修正による新たな欠陥の混入には注意を払う必要があります。

⑦ 終了

インスペクション対象の成果物に含まれていた欠陥がなくなっており、整合性に問題が無いことを確認し公開します。そして今後の改善に向けて欠陥と作業時間に関する統計情報を記録して終了します。

ここで作成された成果物は、次の作業の入力となります。

字句の誤りや形式の不備の指摘は誰にでもできるかもしれません。しかしインスペクションにおいて最優先する事項である「後フェーズで除去しにくい欠陥（不整合、不統一といった形であることが多い）」と「ユーザ要求へ適合しない欠陥」を発見・指摘するためには優れた洞察力・集中力・自由な発想・想像力が必要になります。そのため優れたインスペクションの専門家になるためには経験と訓練が不可欠です。

最後に、各レビュー手法のプロセスと効果について、表 3-3、表 3-4 にまとめています。

表 3-3　レビューの基本プロセス対応表

種別/プロセス	計画	キックオフ	個人チェック	ロギングミーティング	編集/フォローアップ		
					修正	追跡	分析
アドホック・レビュー				●	●		
ペア・プログラミング				●	●	●	
ピア・デスク・チェック			●	継続的	●		
ピア・レビュー	△	△	△	●	●	△	
パス・アラウンド		△	●	可能	●		
チーム・レビュー	●	△	●	●	●		
ウォークスルー	●	△		●	●		
インスペクション	●	△	●	●	●	●	●

●：行う　　△：必ずしも行うとは限らない　　空欄：通常行わない

表 3-4　レビュー手法に期待される主な効果

	欠陥製品の発見	仕様との適合性のチェック	標準との適合性チェック	製品の完全性と正しさの検証	理解性と保守性の評価	クリティカルまたは高リスクのコンポーネントの品質の実証	プロセス改善のためのデータ収集	文書品質の測定	製品に関する他のチームメンバの教育	対処法についてのコンセンサスの決定	変更または欠陥修正の正しさの確認	別の対処法の検討	プログラム実行のシミュレーション	レビューコストの最小化
アドホック・レビュー	●													●
ペア・プログラミング	●				●				●	●		●		
ピア・デスク・チェック	●	●	●									●		●
ピア・レビュー	●	●	●	●					●	●	●			
パス・アラウンド	●	●	●		●				●					
チーム・レビュー	●	●	●		●		●		●	●	●			
ウォークスルー	●			●					●	●	●	●	●	
インスペクション	●	●	●	●	●	●	●	●						

3.2.3　テスト技法の概要

ここでは、テスト技法に関しての概要を紹介します。

図3-7にテスト技法のポジショニングマップを示します。さらに、各テスト技法に関する概要を(1)から(10)に紹介します。

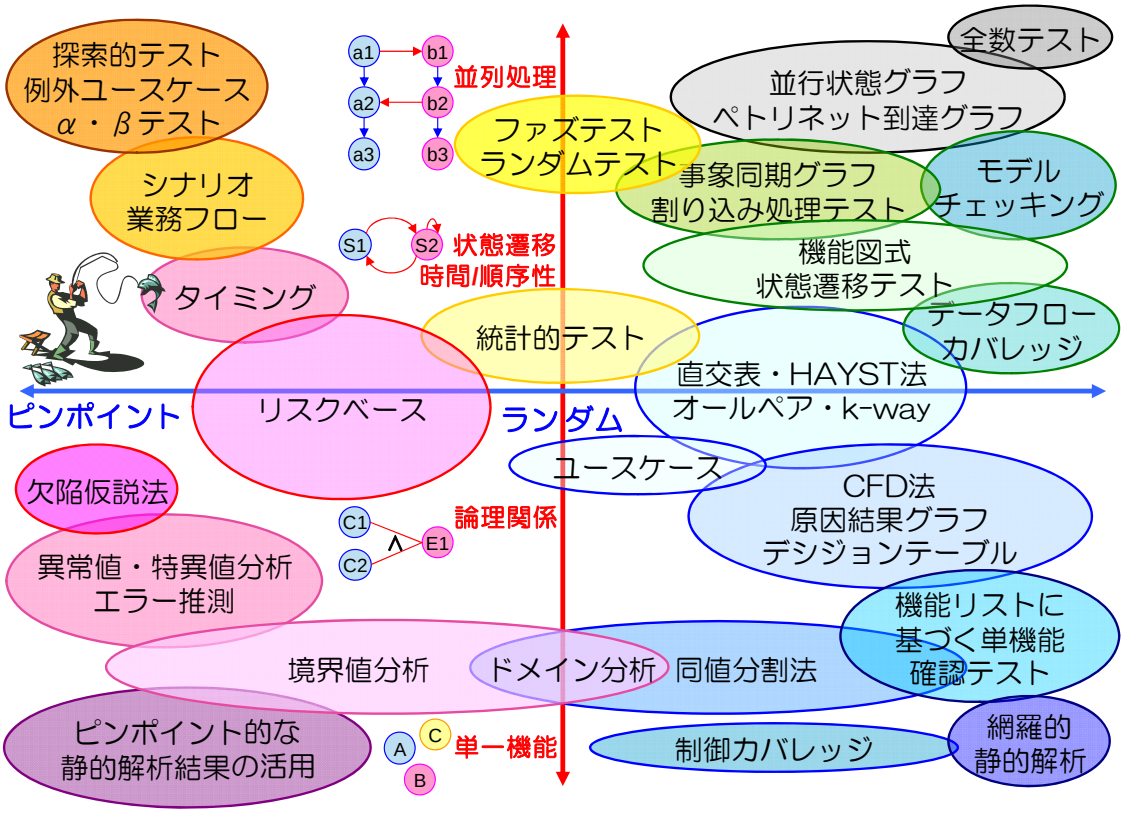


図 3-7 テスト技法のポジショニングマップ

(1)　基本の技法

基本の技法とは、すべてのテスト技法のなかで意識的あるいは無意識で使用されるものを指します。

本項では、「同値分割法」「境界値分析法」「異常値・特異値分析法」の3つを示します。いずれも、テストケース数に対する欠陥検出率（＝検出される欠陥数÷テストケース数）すなわちテストの効果・効率をアップするために用いられる基本的な技法です。

（1-1）同値分割法

仕様ベースのブラックボックステスト技法の1つで、すべてのテスト技法において適用できます。この技法は、ソフトウェアの仕様から判断し同一の処理がされて同様の結果をもたらすことを期待できる入力セットや出力を想定し、テストケースを設計する方法です。

同値分割法では、どちらかというと処理や出力結果に着目して入力を選択します。このとき、同じとみなせる入力領域（入力セット）や出力領域のことを同値クラスと呼びます。同値クラスは、正常系の同値クラスである有効同値クラスと、異常系の同値クラスである無効同値クラスに分かれます。有効・無効とはあくまでもソフトウェアにとっ

ての話ですので、入力ができるかどうかとは無関係です（無効同値クラスの値は入力時にエラーとして処理する防御的アプローチをとるとよい）。

＜使用例＞

JR の料金体系は、
おとな 12 歳以上（12 歳でも小学生は「こども」です）
こども 6 歳～12 歳未満（6 歳でも小学校入学前は「幼児」です）
幼児 1 歳～6 歳未満
乳児 1 歳未満
となっています。

年齢を符号付き4 バイト整数（int）とし、幼児クラスの年齢について同値分割を行うと、

・無効同値クラス： MININT(-2,147,483,648)以上、1 歳未満
・有効同値クラス： 1 歳以上、6 歳未満
・無効同値クラス： 6 歳以上、MAXINT(2,147,483,647)以下

となります。 同値分割法によってテストを実施する場合には、それぞれのクラスから代表値を選びますので、例えば、{－10 歳、3 歳、15 歳}が入力値となります。
この場合の期待結果は、{幼児ではない、幼児、幼児ではない}です。

※ 本例のように同値クラスが連続する場合は、乳児以下の無効同値クラス、乳児、幼児、こども、おとなの有効同値クラス、おとなの設計定義における最高年齢を超える無効同値クラスに分けてそれぞれから代表値を取ります。例えば、{－10 歳、0 歳、3 歳、9歳、15 歳、200歳}が同値分割をした結果としての入力値となります。

（1-2）境界値分析法

入力同値クラスと出力同値クラスの端(境界値)や、その上下の隣接値に着目して効果的に欠陥を検出する技法です。同値クラスの境界付近で誤りが生じやすいという経験則（欠陥のパターン分析：欠陥を起こしやすい要求仕様・設計・実装の分析）が背景にあります。

＜使用例＞

JR の料金体系は、
おとな 12 歳以上（12 歳でも小学生は「こども」です）
こども 6 歳～12 歳未満（6 歳でも小学校入学前は「幼児」です）
幼児 1 歳～6 歳未満
乳児 1 歳未満
となっています。
年齢を入力値とした「幼児かどうかを判定する」関数では、
・有効同値クラス： 1 歳以上6 歳未満
となっています。

* Beizer の方法

境界は値にあるのではなく0 と1 の間と、5 と6 の間にあるとして、{0 歳, 1 歳, 5 歳, 6 歳} を境界値テストの入力値集合として扱う方法です。この考え方は、国際的なテスト技術者資格認定制度であるISTQB（日本ではJSTQB）で採用されています。Beizer の方法を採用すると不等号の書き間違え、例えば、「＜」を「＞」と間違えたことは見つかります。ところが、「＜」を「＝＝」と書き間違えると見

つかりません。この間違いの頻度は低いものですが、その間違いも検出するために、有効同値クラスの代表値（本ケースなら3 歳など）を加えてテストすることをお勧めします。

* Jorgensen の方法
仕様に書かれている端の値と端±１の値、すなわち{0 歳, 1 歳, 2 歳, 5 歳, 6 歳, 7 歳}を境界値テストの入力値集合として扱う方法です。英国標準(BS 7925-2)で採用されています。

どちらの方法を採用するかは、考え方次第です。効率が良いのはBeizer の方法です。しかし、Beizer の方法はテスト設計時に間違いが紛れ込む危険があります。上の例でいうと、{0 歳, 1 歳, 6歳, 7 歳}を境界値テストの入力値集合として採用してしまう危険があるということです。
この間違えを防ぐために、

――――――――0○●1――――――――5●○6――――――――

という絵を描いて、「○：無効同値クラスの境界値」、「●：有効同値クラスの境界値」を視覚的に明らかにすると良いでしょう。

(1-3) 異常値・特異値分析法

本技法の適用目的は、境界値分析と同様でプログラミング上、特別な処理を実施している変数を明らかにし、そこをピンポイントでテストすることで欠陥を検出することです。
この技法は、仕様書や設計ドキュメント、場合によっては開発者へのヒアリング、ソースコードから異常値や特異値をリストアップし、それを入力として与えます。境界値は良く知られた特異値の一種と考えることもできます。

＜使用例＞

4 年に一度開催されるオリンピックの年は、たいていうるう年です。北京オリンピックが開かれた2008 年もうるう年でした。ところがパリ大会が開かれた1900 年は、うるう年ではなく2 月28 日の次の日は3 月1 日でした。
実はうるう年のルールは、次のようになっています。

1. 西暦年が4 で割り切れる年はうるう年
2. ただし、西暦年が100 で割り切れる年は平年
3. ただし、西暦年が400 で割り切れる年はうるう年

従って、うるう年のテストにおいては少なくとも、

* 1 のケース：2008 年2 月28 日とその翌日
* 2 のケース：2100 年2 月28 日とその翌日
* 3 のケース：2000 年2 月28 日とその翌日

のテストが必要です。

(2) ホワイトボックステスト

ホワイトボックステストとは、プログラムの内部設計情報を元にしたテスト設計方法で「構造ベースのテスト」とも呼ばれます。従って、コンポーネントなど比較的小さなプログラム単位に対して、プログラムコードや設計書をもとに内部構造を分析し、テスト設計を実施します。ここでは、「制御フローカバレッジ」、「データフローカバレッジ」、「モデルチェッキング」の3つについて説明します。

(2-1) 制御フローカバレッジ

本技法は、プログラムコードの制御構造に対して、その網羅性を高めるテスト方法です。
網羅性の基準には様々なものがあります。代表的なものは下表の通りです。

表 3-5 制御フローカバレッジの種類

制御フローカバレッジの種類	説明
ステートメントカバレッジ	テストケースによって実行された行の割合。 C0 カバレッジとも呼ぶ
ブランチカバレッジ	テストケースによって実行された分岐の割合。C1 カバレッジとも呼ぶ

通常は、ツールを使用してカバレッジを計測しながら漏れているパスを通るようなテストデータを入力として与えることで、網羅基準になるまでテストを追加します。
ブランチカバレッジが100%の場合、ステートメントカバレッジは100%となります。

(2-2) データフローカバレッジ

本技法は、プログラムコード（変数、データオブジェクト）に対して、その状態の変化をチェックし問題が無いかテストする方法です。データの状態には定義・消滅・使用の3 つがあります。

表 3-6 データの状態の種類

略号	データの状態	説明
d	defined	定義済み。生成済み(created)、初期化済み(initialized)を含む
k	killed	消滅。未定義(undefined)、解放済み(released)を含む
u	usage	使用。演算で使用(computation)と、判定述語で使用(processing)

従ってデータの状態の組み合わせには、順序を考えて次の9 つが考えられます。

表 3-7 データの状態の組み合わせと判定

状態	説明	判定
dd	2重定義	実害は無いが要注意
dk	定義され使われること無く消滅	欠陥の可能性大
du	定義して使用	通常のケース
kd	消滅させて再定義	通常のケース
kk	2回消滅	欠陥の可能性大
ku	消滅したデータへアクセス	明確な欠陥
ud	消滅させずに再定義	実害が無いケースが多いが要注意
uk	使用して消滅	通常のケース
uu	繰り返しの使用	通常のケース

この表の判定結果を元に怪しいところをコードレビューすることで欠陥を検出します。

(2-3) モデルチェッキング

本技法は、システムの振る舞い仕様（状態遷移などの外部に見える挙動）が要件を満たすかどうかを検証します。つまり、コードからではなく設計の段階で振る舞い仕様が正しいかどうか検証する方法です。

特にSPIN というモデルチェッキングツールを用いた検証が有名で、SPIN はPromelaという仕様記述言語で振る舞い仕様を記述し、SPIN を用いて動かすことによって状態空間の網羅的な探索を行い設計の不備を検証します。

富士ゼロックス株式会社の山本氏らは、Executable UML(Bridge Point)を用いて書いたステートチャートをPromela 言語へ自動変換することで、大規模なモデルチェッキングを可能としています（C 言語換算で3 万行程度のプログラムに対して300 万のモデルチェックを自動で実施）。

(3)　論理組み合わせ検証テスト

論理組み合わせ検証テストとは、入力の組み合わせに論理関係（＝規則）があり、様々な入力条件によって出力が変わることの検証を行うテストです。ここでは、「デシジョンテーブル」「原因結果グラフ」「CFD技法」の3つを示しますが、どのテスト技法も最終的にはデシジョンテーブル（決定表）を作成することで、入出力の論理組み合わせ仕様を明らかにします。

(3-1) デシジョンテーブル

本技法は、仕様ベースのブラックボックステスト技法の1つです。JIS では「決定表」として規定されています。この技法は、仕様の論理関係すなわち考えうる論理的条件の組み合わせを表にまとめ、その論理検証網羅性を高めるテスト方法です。つまり、入力する条件の組み合わせによって実行結果が変わる場合に有効な手法です。

表 3-8 デシジョンテーブルの構成

条件項目	ルール（条件の組み合わせ）
結果項目	ルールに基づいたアクション

＜使用例＞

JR の料金体系は、

おとな　12 歳以上（12 歳でも小学生は「こども」です）
こども　6 歳～12 歳未満（6 歳でも小学校入学前は「幼児」です）
幼児　1 歳～6 歳未満
乳児　1 歳未満

となっています。

こちらをデシジョンテーブルで表現すると、表 3-9 となります。

表 3-9 デシジョンテーブルの 1

JR 料金体系 1	ルール 1	ルール 2	ルール 3	ルール 4	ルール 5	ルール 6
13 歳以上	T	F	F	F	F	F
12 歳	–	T	F	F	F	F
7 歳～11 歳	–	–	T	F	F	F
6 歳	–	–	–	T	F	F
1 歳～5 歳	–	–	–	–	T	F
0 歳	–	–	–	–	–	T
おとな	×					
こども			×			
幼児					×	
乳児						×
次表へ		×		×		

表 3-10 デシジョンテーブルの 2

JR 料金体系 2	ルール 7	ルール 8	ルール 9	ルール 10
12 歳	T	T	F	F
6 歳	–	–	T	T
小学生	F	T	T	F
おとな	×			
こども		×	×	
幼児				×

このケースで、論理条件は、「13 歳以上、12 歳、7 歳～11 歳、6 歳、1 歳～5 歳、0 歳、小学生（か否か）」の7つあります。従って、総組み合わせで考えると2 の7 乗の128 通りあるのですが、上に示した通り10のルールで検証できるわけです。

※ 実際のテストの場合は､1つのルールに対して境界値を複数テストする場合があります。例えば、単機能テストが不十分な場合は、ルール3で7 歳と11 歳の2つのテストケースを実施したほうが良いでしょう。

(3-2) 原因結果グラフ

本技法は、1970年にElmendorf が発表したテスト技法で、仕様ベースのブラックボックステスト技法の1つです。入力やイベント（＝原因）の組み合わせと、出力（＝結果）との関係をブールグラフ化し、そこからデシジョンテーブルを作成します。

この技法は、仕様の論理関係、すなわち考えうる論理的条件の原因結果グラフにまとめ、その論理検証網羅性を高めるテスト方法です。つまり、入力する条件の組み合わせによって実行結果が変わる場合に有効な手法です。

＜使用例＞

JR の料金体系は、

おとな　12 歳以上（12 歳でも小学生は「こども」です）

こども　6 歳～12 歳未満（6 歳でも小学校入学前は「幼児」です）
幼児　1 歳～6 歳未満
乳児　1 歳未満
となっています。

こちらをデシジョンテーブルで表現すると、前項の表 3-9、表 3-10 のようになりますが、表 3-9 が非常に単純なものであるのに対して、2 つ目の表 3-10、すなわち小学生判定の方は少々論理関係が複雑です。このケースで、原因結果グラフを作成すると図 3-8 のようになります。

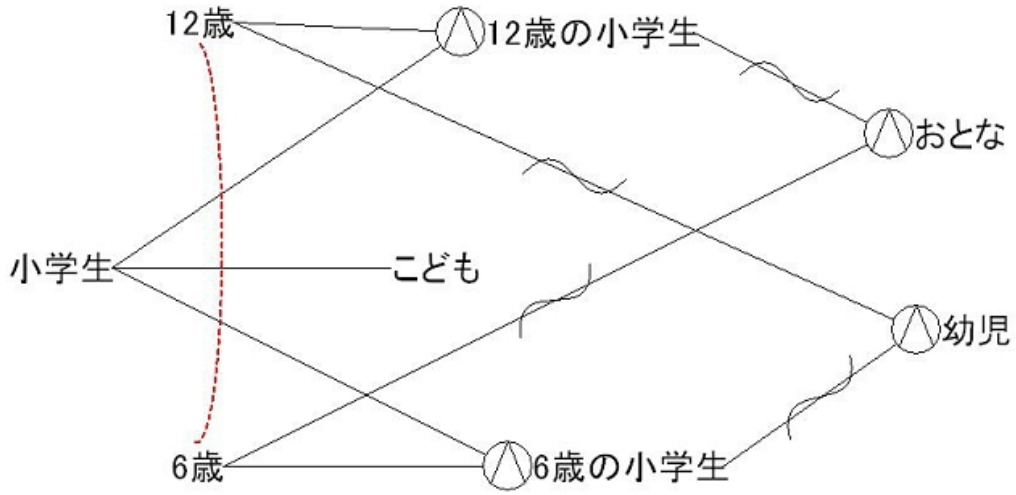


図 3-8 原因結果グラフ

図 3-8で、左側にある「12歳」「小学生」「6歳」を原因ノードと呼びます。ここでは、特に論理的に判定に絡んでくる「12歳」と「6歳」をピックアップしています。「12歳」と「6歳」のノードの間が点線でつながれているのは同時に成立することがありえないというONE制約を示しています。

右端の「おとな」「こども」「幼児」を結果ノードと呼びます。間にある「12歳の小学生」、「6歳の小学生」を中間ノードと呼びます。

これらのノードを線で結び、論理積（AND）関係にあれば「∧」、論理和（OR）関係にあれば「∨」のマークをつけます。また、線上の波マークは否定（NOT）を表しています。

そして、図 3-8の原因結果グラフに表れるすべてのノードのTrueとFalseがテストされるように、デシジョンテーブルを作成します。作成手順は割愛しますが、デシジョンテーブルを作るまでもなく、この図を作成するだけで問題が発見できる場合も多いものです。

(3-3) CFD技法

本技法は、仕様ベースのブラックボックステスト技法の1つです。「処理の流れ」をもとにテスト設計するため、設計情報が若干必要となります（グレーボックステストといいます）。

この技法は、まずソフトウェアに与えられる入力を集合で用いるベン図と同様の記法で図式化し、次に、それらを処理の順番に並べて取り得るテストケースをデシジョンテーブルにまとめます。この時に、補集合（年齢であれば不明という条件）を絵にしておき、検討することがCFD技法の特長となっています。補集合の検討は、仕様検討やレビュー

で漏れやすいものです。

＜使用例＞

JR の料金体系は、

おとな　　12 歳以上（12 歳でも小学生は「こども」です）
こども　　6 歳～12 歳未満（6 歳でも小学校入学前は「幼児」です）
幼児　　1 歳～6 歳未満
乳児　　1 歳未満

となっています。

こちらをデシジョンテーブルで表現すると、前々項の表 3-9 のようになりますが、表 3-9 が非常に単純なものであるのに対して、2 つ目の、すなわち小学生判定の方は少々論理関係が複雑です。このケースで CFD を作成すると、図 3-9 のようになります。

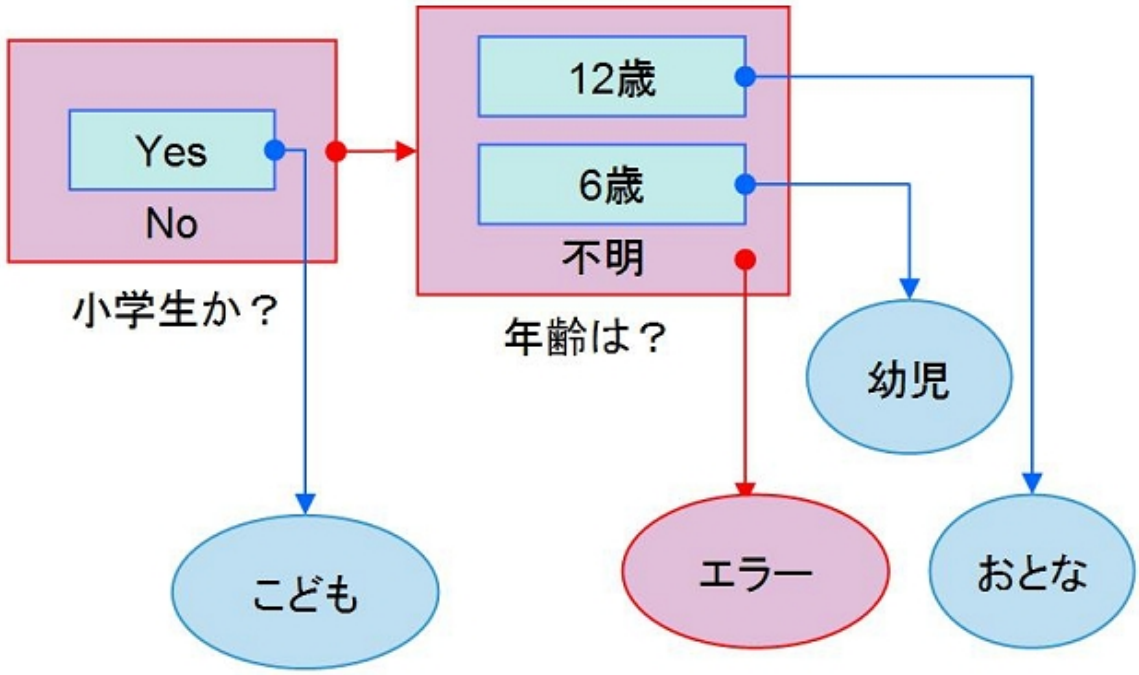


図 3-9 CFD

さて、この図から線をたどってデシジョンテーブルを作成すると、表 3-11となります。

表 3-11 CFD から作成したデシジョンテーブル

JR 料金体系 2	ルール 1	ルール 2	ルール 3	ルール 4
小学生	T	F	F	F
12 歳	–	T	F	F
6 歳	–	–	T	F
不明	–	–	–	T
こども	×			
おとな		×		
幼児			×	
エラー				×

デシジョンテーブルが若干異なっていることに注意してください。もし2番目の年齢処理のときに、12 歳と6 歳しかデータがこないことが保証されていたら（契約的プログラミング）、ルール4 のテストケースは不要となり3 テストケースになります。

これは、処理の流れという情報を追加することによってテストケースを合理的に減らすことができるということです。

(4) 組み合わせテスト

組み合わせテストとは、入力が独立していて、組み合わせに論理関係（＝規則）がほとんどない場合に全体を組み合わせて想定外の組み合わせ欠陥を検出することを目的としたテストです。

大きく分類すると「直交表」、「HAYST法」、「All-Pair法 / Pairwise」の3種類になりますが、ここでは「直交表」と「All-Pair法 / Pairwise」について説明します。

「HAYST法」については、第6章および第2部3章の『富士ゼロックス株式会社の事例』を参照ください。どのテスト技法も最終的には組み合わせ表を作成することで、テストの入力条件を明らかにします。ただし、入力数が多いため、期待結果を事前に用意することが困難です。強いて言えば、入力が独立しているのでそれぞれの入力値が示す単独の出力結果が期待結果となります。

(4-1) 直交表

本技法は、ブラックボックステスト技法の1つです。本技法の適用目的は、デシジョンテーブル（原因結果グラフ、CFD技法を含む）と異なり、入力条件の組み合わせに仕様上は論理関係が特にないようなケースで全体の組み合わせを網羅的に確認することです。

この技法については、第6章で詳細に説明します。

(4-2) All-Pair法/Pairwise

本技法は、ブラックボックステスト技法の1つです。本技法の適用目的は、直交表と同様で、デシジョンテーブル（原因結果グラフ、CFD技法を含む）と異なり、入力条件の組み合わせに仕様上は論理関係が特にないようなケースで全体を網羅的に確認することです。

直交表と異なる点は、直交表が2因子間の水準組み合わせが「同数回」出現するのに対

して、All-Pair法では「同数回」という条件を外すことでよりテストケース数を削減できる点にあります。PICTというフリーで使用できるツールが有名です。

(5) 状態遷移系のテスト

状態遷移系のテストとは、メニュー選択時に画面が選択したボタンなどによって変わるといった状態遷移時に発生する欠陥を検出することを目的としたテストです。ここでは、「状態遷移テスト」「タイミングテスト」「並行処理テスト」「ペトリネット」の4つを示しますが、状態遷移テストは簡単なプログラムであってもすべてのテストパスを網羅させようとするとテストケース数が膨大になるためテスト方針（テストモデルの網羅方針）を立てることが重要です。

(5-1) 状態遷移テスト

本技法は、仕様ベースのブラックボックステスト技法の１つです。この技法は、状態遷移図から、状態遷移表を作成し、

有効な状態遷移が正しく行われること
無効な状態遷移が行われないこと

をテストします。

状態遷移テストには、いくつかの網羅基準が定義されています。

表 3-12 状態遷移テストの網羅基準

名称	説明	方法
状態の網羅	各状態（○）をすべてテストする	状態遷移図の状態を確認
0 スイッチカバレッジ	各遷移（→）をすべてテストする	状態遷移表のセルを確認
1 スイッチカバレッジ	2 回の遷移で行けるパスをすべてテストする	グラフ行列からパスを求める

＜使用例＞

うたた寝モード付きの目覚まし時計を考えてみます。

うたた寝モードとは、目覚ましが鳴っているときに、Snoozeボタンを押すと5分鳴り止んで、その後、また、目覚ましが鳴るという機能です。

図 3-10は目覚まし時計の仕様をもとに作成した状態遷移図です。

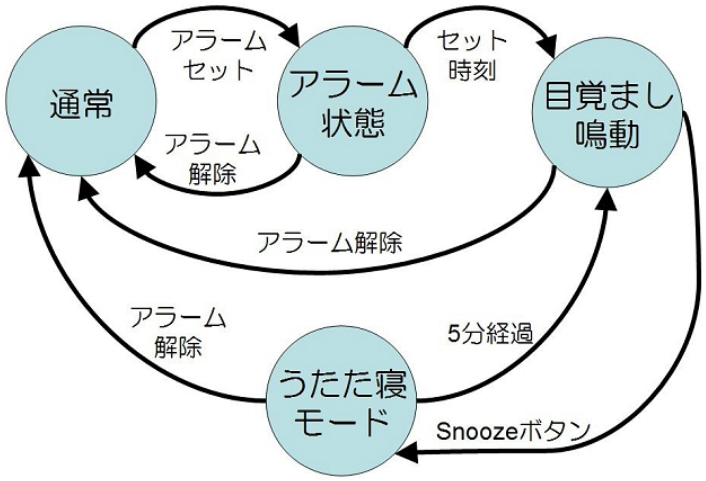


図 3-10 目覚まし時計の状態遷移図

この状態遷移図を見ながら、丸（○：状態）をすべてテストするというのが「状態の網羅テスト」です。

次に、状態遷移図から状態遷移表を作成します。

状態遷移表は、表の見出し行にすべての遷移イベントを列挙し、表の左端の列にすべての状態を列挙したものです。表中のセルの内容は、イベントにより遷移する先の状態を示しています。

表 3-13 状態遷移表

	アラームセット	セット時刻	Snooze ボタン	5 分経過	アラーム解除
通常	アラーム状態	N/A	N/A	N/A	N/A
アラーム状態	N/A	目覚まし鳴動	N/A	N/A	通常
目覚まし鳴動	N/A	N/A	うたた寝モード	N/A	通常
うたた寝モード	N/A	N/A	N/A	目覚まし鳴動	通常

※ N/A は、"Not Applicable" の略で、仕様上は遷移しないという意味です。

この状態遷移表のセルを1つ1つ、N/A の部分が遷移しないことを含めて確認することを「0スイッチカバレッジ」と呼びます。状態遷移図から状態遷移表を作成する理由は、状態遷移図に直接表現されないN/A部分をきちんと認識し、テストするためです。イベントのガードが甘いと予期せぬ遷移を起こしてしまうからです。

次に、グラフ行列から「1スイッチカバレッジ」を実現する方法を説明します。まず、表 3-14のように状態遷移表を状態×状態の2元表に書き直します。

表 3-14 関係行列

	通常	アラーム状態	目覚まし鳴動	うたた寝モード
通常		アラームセット(a)		
アラーム状態	アラーム解除(r)		セット時刻(s)	
目覚まし鳴動	アラーム解除(r)			Snooze ボタン(b)
うたた寝モード	アラーム解除(r)		5 分経過(5)	

となります。これを行列式とみなして2乗すると表 3-15が得られます。

表 3-15　1 スイッチカバレッジを満たす表

	通常	アラーム状態	目覚まし鳴動	うたた寝モード
通常	アラームセット(a) アラーム解除(r)		アラームセット(a) セット時刻(s)	
アラーム状態	セット時刻(s) アラーム解除(r)	アラーム解除(r) アラームセット(a)		セット時刻(s) Snooze ボタン(b)
目覚まし鳴動	Snooze ボタン(b) アラーム解除(r)	アラーム解除(r) アラームセット(a)	Snooze ボタン(b) 5 分経過(5)	
うたた寝モード	5 分経過(5) アラーム解除(r)	アラーム解除(r) アラームセット(a)		5 分経過(5) Snooze ボタン(b)

表 3-15に示すように、各セルには2つずつイベントが書き込まれています。つまり通常から通常へ2 回の状態遷移で到達するには、「アラームセットを実行してからアラーム解除を実行する」というように読みます。

このセルをすべてテストすることにより「1スイッチカバレッジ」が100%となります。

(5-2) タイミングテスト

本技法は、欠陥が発生しそうなきわどいタイミングを狙ったテストのことです。

特に、組込み系では、

電源が不安定な状態（自動車のエンジン起動中など）
割り込みが重なる状況
システム停止中の割り込み
エラー処理中の割り込み

といった、欠陥が発生しやすい（設計エラーを起こしやすい）状況が存在します。

過去の障害解析を実施し、失敗しやすい場所についての情報を蓄積し、そのタイミングを狙ったテストをすることが効果的です。また、タイミングを狙う代わりに高負荷状態にしておき、発生確率を上げる方法をとる場合もあります。

＜使用例＞

自動車のエンジンをかけた直後にCD のイジェクトを試す
エラー処理中に、入力イベントを重ねて送る
高負荷状態を作り出しておいて、機能テストを何度か実施する

(5-3) 並行処理テスト

本技法は、並行処理（例えばクライアントサーバ）に対して網羅的に実施するテストのことです。並列処理テストについては、同値とみなしてよい処理パスを明らかにし、それらについての組み合わせを網羅する方法が取られています。ただし、テストケース数が膨大となるためきわどい箇所に限定して並列パス網羅性を確認するケースがほとんどです。

(5-4) ペトリネットテスト

本技法は、並行処理（例えばクライアントサーバ）に対してペトリネットを使用してソフトウェアを開発し、併せてテストを実施する方法です。カラーペトリネットを使用したオブジェクト指向ソフトウェアへの並列テストが提案されています。

(6) ユーザ視点のテスト

ユーザ視点のテストとは、ユーザから見たシステムを意識したテストです。ここでは、「ドメイン分析法」、「ユースケース/例外」の2つを示します。

(6-1) ドメイン分析法

本技法は、各入力要因に対して、on、off、in、out の分析を実施し、それらの要因をデシジョンテーブルで組み合わせます。

on、off、in、out をまとめると表 3-16のようになります。

表 3-16 ドメインテストのポイント

名前	説明	x >= 10 の場合	x > 10 の場合
on	境界上の値	10	10
off	境界に隣接する値（※）	9	11
in	ドメインの内側の値(on/off ポイントは除く)	15	15
out	ドメインの外側の値(on/off ポイントは除く)	5	5

※ off ポイントは、「境界に関する条件を満足しない点」です。

境界の開閉をまとめると表 3-17のようになります。

表 3-17 ドメインテストの開閉

境界	on ポイント	off ポイント
閉(<=、>=、=)	ドメインに含まれている	ドメインの外側近傍
開(<、>)	ドメインに含まれていない	ドメインの内側近傍

同値分割、境界値分析法と近い関係にあるため、ドメイン分析を「マルチ変数同値分割」、「マルチ要素同値分割」と呼ぶことがあります。

＜使用例＞

印刷部数として、1 部から99 部まで指定することができるプリンタドライバがあったとします。このときに、ドメイン分析を行ってみます。まず、on ポイントに1 部と99 部を選択した場合のことを考えてみましょう。取りうる値は、整数値のみですから、表 3-18となります。

表 3-18　on ポイントに 1 と 99 を選んだ場合

ドメイン	式	on	off	in	out
小さすぎて無効	x < 1	1	0	−5	30
有効値(下側の有効)	x >= 1	1	0	30	−5
有効値(上側の有効)	x <= 99	99	100	30	150
大きすぎて無効	x > 99	99	100	150	30

次に試しに、同じ仕様に対してonポイントに0部と100部を考えてドメイン分析を実施すると表 3-19となります。

表 3-19　on ポイントに 0 と 100 を選んだ場合

ドメイン	式	on	off	in	out
小さすぎて無効	x <= 0	0	1	−5	30
有効値(下側の有効)	x > 0	0	1	30	−5
有効値(上側の有効)	x < 100	100	99	30	150
大きすぎて無効	x >= 100	100	99	150	30

テストデータは、コーディングをどちらで実施したとしても、同じで良いことが分かります。

※ ドメイン分析の場合、境界で分析するので有効同値クラスであっても、その両端で分析します。2つの隣接ドメインは同じoff ポイントを共有します。

(6-2) ユースケース/例外

本技法は、個々の機能の検証ではなく、システム全体に対して一連のトランザクションを流すことで、コンポーネント間の不整合を検出することを目的としています。

ユースケーステストの設計方法は、

1. ユーザの業務を調査し、代表的な使用方法をリストアップする
2. 上記使用方法に対してテストシナリオを作成する
3. 最もありそうなシナリオ（メインストリーム）
4. 例外的なシナリオ（拡張ストリーム）

になります(開発工程で作成したユースケースやアクティビティ図が使用できるかもしれません）。これをシステムテストまたは受け入れテストフェーズで実行します。

＜使用例＞

インターネットショッピングでの購入ユースケース。

1. 品物を検索
2. ショッピングカートに入れる
3. レジに進む
4. サインインする
5. 配送条件を確定する
6. 注文を確定する

※ 例外シナリオテストの場合は、2 の後に「注文を削除する」といったアクティビティを実行します。

補足

ユースケースには、次のレベルがあります。

空のレベル

「書籍を注文する」といった大まかなシステムの目標を明らかにする段階

凧のレベル

「新規の注文」、「品目の追加」など詳細な段階

海のレベル

複数のアクタの相互作用を考慮する段階

泥のレベル

実装を意識した詳細な階層構造を決める段階

ユースケーステストとして最も適切な素材となる情報は「海のレベル」です。

(7) 非機能要件のテスト

非機能要件のテストとは、システムが何をするのか（What の確認）ではなく、システムがどのように動作するのか（How の確認）です。「どのように」ということで個別の要求色が強いため、あまり技法らしい技法はありません。

非機能要件のテストタイプには、「パフォーマンステスト」「操作性テスト」「保守性テスト」「セキュリティテスト」などがあります。

(8) 品質保証のためのテスト

品質保証のためのテストとは、欠陥出しを目的としたテストではなく、システムがユーザに受け入れられる品質レベルに達したかどうかを判定するテストです。サンプリング結果を統計処理し、品質レベルを判定します。ここでは、「サンプリングテスト」「探針テスト」「統計的テスト」の3つを示します。

(8-1) サンプリングテスト

本技法は、たくさんのテストケースからランダムに選択したテストケースを実行することで、「テストをすべて実施したときに発見されるはずの欠陥数を予測する」テストです。

欠陥数の予測方法は、比例関係から推定します。

＜使用例＞

全テストケース数(Tt)： 1,000 件

ランダムサンプリングして選んだテストケース数(Ts)： 100 件

上記100 件のテスト結果(Es)： 7 件の欠陥を発見

もし全部テストしたら(Ep = ES×(Tt/Ts))： 70 件位見つかっただろう

補足

サンプリングテストを成功させるためには、「サンプル」が良質である必要があります。

すなわち、

- サンプルとして選んだテストケースがすべての機能(含エラー処理)を網羅していること
- サンプルとして選んだテストケースの粒度がそろっていること

が満たされる必要があります。また、開発の力量も高いことが前提です。

安易なサンプリングテストによる品質評価はリスクが高いことを忘れてはなりません。

(8-2) 探針テスト

本技法は、製品検査に先立って、デバッグ・テスト段階における品質を検査部門が実際に測定・評価するものです。

製品検査と同様に実際の計算機を使用して実施し、この時の摘出欠陥件数から製品検査時の欠陥件数を統計的手法で推定します。それにより、欠陥内容の分析・評価による弱点の具体的指摘でデバッグ・テスト段階での欠陥の先取りを加速し、品質向上施策への指針を与えます。つまり、「統計的手法で区間推定」できるようになる点がポイントです。

区間推定には、ベータ分布を使用するとよいでしょう。また「見つかった欠陥件数とその傾向から、開発部門が行ってきたテストについてその弱点を指摘し、品質向上施策への指針を与える」ことが重要です。テスト内容を探針によってコントロールします。

＜使用例＞

全テストケース数(N)： 1,000 件
ランダムサンプリングして選んだテストケース数(n)： 100 件
上記100 件のテスト結果(r)： 7 件の欠陥を発見
信頼率(1 - α)： 95%

Pu = β(1- α/2, r+1, n-r) = 0.1389
です。
表計算ソフト*では、「＝BETAINV(1-(1-0.95)/2;7+1; 100-7)」となります。

Pl = β(α/2, r, n-r+1) = 0.0286
です。
表計算ソフトでは、「＝BETAINV((1-0.95)/2;7; 100-7+1)」となります。

PuとPlを使用して次の式で上限の欠陥数と下限の欠陥数を推定します。

上限の欠陥数 = Pu × N = 0.1389 × 1000 = 139
下限の欠陥数 = Pl × N = 0.0286 × 1000 = 29

もし全部テストしたら、95%の確率で29 件～139 件の間に欠陥数が収まるだろうと推定できます。

補足

探針テストは、サンプリングテストの一種ですから、サンプリングテストと同様に、探針テストを成功させるためには、「サンプル」が良質である必要があります。すなわち、
- サンプルとして選んだテストケースが、すべての機能(含エラー処理)を網羅していること
- サンプルとして選んだテストケースの粒度がそろっていること

が満たされる必要があります。また、開発の力量も高いことが前提です。
安易な探針テストによる品質評価はリスクが高いことを忘れてはなりません。

(8-3) 統計的テスト

本技法は、ユースケースを状態遷移グラフに変換し、さらにユーザの使用頻度をきちんと統計的に処理した遷移確率をもとにテストケースを生成する方法です。統計的テスト技法を用いるとソフトウェアの信頼性を予測することができるようになります。

---

* OpenOficce.org 3.2

(9) 熟練テスターのノウハウ

熟練テスターのノウハウによるテストとは、形式知化されていない技術を用いてテストを実施する方法です。ここでは「エラー推測」「探索的テスト」の2つを示します。

(9-1) エラー推測

本技法は、テスト担当者の経験・知識・直感をベースとした非公式なテスト設計技法の1つです。開発者が起こしやすいミス（エラー）に着目し、その結果どのような欠陥が作りこまれるか仮説を立てて、その欠陥があるものとしてテストケースを設計する方法です。

エラー推測の方法は、次の通りです。

1. ありうるエラーとその結果作りこまれる可能性がある欠陥をリスト化する
2. 上記エラーリストに対するテストケースを作成する

基本的に、事前にテストケースを作成しておきます。

＜使用例＞

図面管理システムが新しくなったので、過去のシステムから図面を移行することになったとします。この時、図面番号（7 桁の数字以外は再番する）としてどのようなテストデータを考えるかをエラー推定すると表 3-20のようになります。

※ 表 3-20はエラー推測の例です。他にも必要なテストデータを考えることができます。

表 3-20 エラー推測の例

データ	理由
0	どのようなときにも数値データの 0 はエラーを起こしやすい
-1	マイナスのデータ
007	頭ゼロの取扱い
123-2	枝番の処理
123-a	枝番かつ文字
_123	先頭にスペース
123_	末尾にスペース
12_3	間にスペース
12345678	桁あふれ
12.3	小数点
1E3	浮動小数点形式
0x10	16 進形式
同一番号	同一番号が来た場合の警告処理
上記の組み合わせ	エラーが組み合わさった場合の処理

補足

エラー推測は、思いつくままにテスト設計無しにテストするアドホックテストでも多くの欠陥を見つけることができる人がいることの分析から生まれたものです。
エラー推測のコツは、開発者が見逃しそうな問題を想定することです。

(9-2) 探索的テスト

本技法は、テスト担当者の経験・知識・直感をベースとした非公式なテスト設計技法の1つです。「テストケースを事前に作成しない」点が特徴で、アドホックテストと似ていますが、アドホックテストが過去にエラーを起こした事柄やユーザにとってのリスクに焦点を当てて直感に頼って操作を行い、欠陥を検出しようとするのに対し、探索的テストでは、テスト結果をフィードバックしながら経験に基づいて重要と判断した領域を対象として、さらに深く掘り下げたテストを実施します。

補足

探索的テストを成功させるためには、高水準の知識とテストの経験が必要です。もし、探索的テストで欠陥が見つからなかったら、テストのカバーが不足していたのかテスターの知識・経験が不足していたのか判断がつきません。従ってテストチームのメンバの状況に合わせ、探索的テストの割合を調整することが大切です。
なお、探索的テストの網羅性を計測するというアプローチをしている人もいます。

(10) ユーザ受け入れテストユーザ受け入れテストとは、ユーザが業務で使用する方法です。ここでは、「α・βテスト」「受け入れテスト」の2つを示します。

(10-1) α・βテスト

αテストは、社内ユーザが業務でテスト版を使用することです。βテストは社外ユーザが業務でテスト版を使用することです。

ユーザに試して欲しいことの提示と、ユーザからのフィードバックのプロセスが大切です。

(10-2) 受け入れテスト

ユーザ自身が行うテストです。実際のデータを流すことが中心となります。

第4章　障害事例から学ぶ予防活動

病理学では、症例からの仮説検証に基づく病気の予防活動などが研究されていますが、同じようにソフトウェアに関しても、障害事例から再発防止策に基づく予防活動が重要になります。この章では障害事象から、いかにして再発を防止するかの方法を解説します。

また、過去に発生した社会的に重要なシステムの障害事例を取り上げ、根本原因および影響度合い評価（コラム：「障害影響度指標」の表 4-2を参照）により品質要求に関係する課題を仮説検証し、品質特性（JIS X0129-1：2003）ごとに代用特性*を定め、再発防止策を具体的に例示しています。

なお、紹介している障害事例は、ソフトウェアのライフサイクルの観点から、ソフトウェア開発、保守†および運用における障害事例を対象としていますが、直接の障害原因が保守および運用に起因していても、ソフトウェア開発での予防活動および検知活動として再発防止策をとらえています。

4.1 予防活動としての障害事例の扱い方

障害事例を予防活動での手法として取り扱うためには、突発的に発生する障害を「他山の石」として、再発防止のために障害の根本原因と適切な対処方法の事例を分類・整理して蓄積するというプロセスを確立させることが必要になります。

以下の表 4-1には、障害事例が予防活動に結びつかいないプロセス（負のスパイラル）と予防活動に結びついたプロセス（正のスパイラル）に関する事例を対比して示しています。

表 4-1　障害対応プロセスの違い

		予防活動に結びつかない障害対応プロセス（負のスパイラル）	予防活動に結びついた障害対応プロセス（正のスパイラル）
情報公開・共有		**■非オープン（個人、組織、社会）** 一部の組織や利害関係者のみが情報を保有し、秘密裏に対応しようとする。	**■オープン（個人、組織、社会）** 全社レベルで情報を共有し、障害の発生防止を最優先タスクとして対応方針を定め、必要なリソース、期間を準備する。
障害分析		**■直接原因のみの追究** 早期復旧のため直接原因の究明に注力してしまい、根本原因の究明まで至らない。	**■根本原因の追究** 発生障害の直接原因究明にとどまらず、障害を誘発させた根本原因を追究し課題を明らかにする。
障害対処	縦展開（二次障害）	**■緊急療法のみの対処** 根本原因が解決されないまま障害復旧作業（データパッチ等の直接原因の排除）を実施し、後続処理の影響を検討せずに対処完了としてしまうため二次災害が防止できない。	**■二次災害防止を考慮した対処** 障害復旧のための対処は当然のこと、後続処理への影響を考慮した障害対処案を立案し、これを実施する。
	横展開（類似障害）	**■類似障害を点検しない対処** 障害の根本原因を組織的に共有しないため、同時期に開発した他の部分の類似する作り込みを点検せず、同様な障害発生を防止できない。	**■類似障害を撲滅するための対処** 障害の根本原因から関連する開発全体の点検項目を洗い出し、組織的な協力により点検を行い、検知されたリスクを排除する。
情報整理・蓄積		**■人的記憶に依存** 「障害事例から得られるノウハウを保有システムの開発・保守局面で展開しよう」とする組織的な取り組みは期待できず。ノウハウの展開は担当者個人に依存してしまう。	**■新たな開発又は保守作業への利用** 障害分析と対処内容を体系的に整理しノウハウとして蓄積しておき、新規開発や次回の改修時に利用することで、システムの品質向上・障害発生防止を図る。

予防活動

---

* 「要求される品質特性を直接測定することが困難なため、その代用として用いる品質特性」（JIS ハンドブック 14 品質管理 1995 日本規格協会）、例えば、「操作しやすい」は品質特性で あるが、「キータッチ回数」 は代用特性の 1 つとなる。

† JIS X 0161:2001「ソフトウェア保守」での「適応保守」および「完全化保守」に関しては、ソフトウェア開発に含めている。

コラム： 障害影響度指標一覧表

システム障害が発生した場合、障害による影響（被害の度合い）を客観的に把握する必要があります。障害の程度によりその被害の度合いは大きく変わりますし、対処方法や再発防止への取り組み内容も変わるはずです。

システム障害による影響度合いを指標化し発生障害をレベル付けすることで、障害発生時での対処方法および新たに構築するシステムの再発防止策などに関する水準を判定することが可能となります。

以下の表 4-2 は、4.3 節以降に紹介している障害事例での障害影響度を定めるための障害影響度指標一覧表です。影響度合いを判定する項目は「サービス停止時間」、「業務への影響」、「ユーザへの影響」、「監督官庁からの要求」、「経済的損失」の 5 項目とし、それぞれ影響度合いを 5 階級に設定しています。

当該指標を適用される場合、判定項目や階級は自社のシステムに合わせて変更していただく必要があります。

表 4-2 障害影響度指標一覧表

階級	サービス停止時間	業務への影響	ユーザへの影響	監督官庁からの要求	経済的損失
1	数分（10分）未満でサービス停止となる	サービス停止業務の担当者は、遂行業務と併行して障害対応を行わなければならない			
2	10分以上1時間未満の範囲でサービス停止となる	サービス停止業務の担当者は、遂行業務を一時中断し、障害の復旧ならびにユーザ対応を行わなければならない	システム利用ユーザの5%未満に影響を与え、クレームが発生する		
3	1時間以上4時間（半日）未満の範囲でサービス停止となる	サービス停止業務の関連部門担当者は、遂行業務を中断し、障害の復旧ならびにユーザ対応を行わなければならない	システム利用ユーザの5%以上20%未満に影響を与え、賠償を伴うクレームが発生する	監督官庁より注意をうける	自社経常利益の1%未満の損失が発生する
4	半日以上サービス停止となる	サービス停止業務の関連部門の全担当者は、全ての遂行業務を中断し、障害復旧ならびにユーザ対応を行わなければならない	システム利用ユーザの20%以上40%未満に影響を与え、賠償を伴うクレームが発生する	監督官庁より勧告をうける	自社経常利益の1%以上3%未満の損失が発生する
5		全社レベルで遂行業務を全て中断し、障害復旧ならびにユーザ対応を行わなければならない	システム利用ユーザの40%以上に影響を与え、賠償を伴うクレームが発生する	監督官庁より業務改善命令以上の処罰をうける	自社経常利益の3%以上の損失が発生する

4.2 障害発生時の障害分析と対処方法

4.2.1 再発防止策の立案方法

根本原因を特定するには、ソフトウェア開発、保守および運用にかかわるプロセス、教育制度および内容、組織の構造および体質、さらには商慣行など様々の問題や課題を広範囲に考慮する必要があります。

本ガイドブックでは、次の2つの考え方に基づき根本原因分析から再発防止策の立案方法を紹介しています。

1つ目は、第4章の冒頭で述べましたように、ソフトウェア開発、保守および運用における障害事例を対象としていますが、直接の障害原因が保守および運用に起因していても、ソフトウェア開発での予防活動および検知活動として再発防止策をとらえています。

2つ目は、根本原因の網羅性を担保し、かつ特定するため品質特性（JIS X0129-1：2003）を利用しています。さらに根本原因分析を通して、本来実現すべき要求品質を想定（想定要求品質）し、その想定要求品質から代用特性を変換することで、再発防止策を導いています。

再発防止策の立案方法の詳細を、以下の(1)と(2)で説明します。

(1) 根本原因分析から想定要求品質を設定

障害の根本原因分析から、本来あるべき要求品質を想定（想定要求品質）します。

一般的に、要求品質は「操作がしやすい」のような上位レベルから、「画面項目は日本語表示でわかりやすい」の下位レベルまで階層構造をもっています。再発防止策を導き出すには、要求品質が達成されなかった項目とその理由を詳しく分析することが必要になりますので、想定要求品質は「画面項目は日本語表示でわかりやすい」までの下位レベルまで展開する必要があります。

なお、次項(2)では最下位レベルの想定要求品質を使用し、再発防止策の立案方法を紹介します。

(2) 代用特性展開表を利用した再発防止策の立案

要求品質と同じように代用特性も階層構造をもっています。

例えば、発生障害の根本原因分析から「操作のしやすさ」という想定要求品質に対して、是正が必要と結論されたとします。「操作のしやすさ」という想定要求品質から品質特性「使用性」の品質副特性「運用性」（1次）が選択肢として考えられます。「運用性」から「利用時のメッセージをわかりやすくする」、「画面操作時のナビゲートを工夫する」などの代用特性（2次）が展開できます。しかし、「利用時のメッセージ理解性を向上させる」や「画面操作時のナビゲートを工夫する」だけでは、具体的な再発防止策になり得ません。さらに、代用特性（2次）「利用時のメッセージ理解性を向上させる」から、「画面項目の日本語表示機能の実装」および「画面操作時のナビゲートを工夫する」から「エラー時の項目カラー表示機能の実装」といった代用特性（最下位）まで深堀する必要があります。

このように代用特性を最下位レベルまで展開すると、具体的な再発防止策を得ることが可能になります。つまり具体的な再発防止策が得られた段階が、最下位レベルの代用特性となるわけです。また、この例ではソフトウェア機能まで展開しましたが、ものによっては再発防止予防策として、開発標準化規約など多様な具体案が立案されてきます。

表 4-3に代用特性展開表の事例を示します。

表 4-3　代用特性展開表

発生障害の根本原因	想定要求品質	代用特性展開			
		代用特性(1次)		代用特性(2次)	代用特性(最終)
		品質特性	品質副特性		
不正(誤入力)データの摘出遅れ	操作のしやすさ	使用性	運用性	利用時のメッセージをわかりやすくする	画面項目の日本語表示機能の実装
		使用性	運用性	画面操作時のナビゲートを工夫する	エラー時の項目カラー表示機能の実装
		使用性	習得性	ユーザレベル(初・中・上級)などを配慮した習得容易性を向上させる	本番稼働前でのオペレーション教育の実施
	不正データの早期検知	保守性	安定性	システム保有データの整合性を維持する	不整合データ摘出機能の実装

※ 以下に代用特性展開表を展開する工夫ポイントを列記します。

- 開発（要件定義、設計、実装、テスト）、保守および運用フェーズごと、または成果物（例：データベース設計書での品質に関係する記述内容など）ごとに代用特性を分類します。
- 重要度の高い要求品質に的を絞っての展開、さらにボトルネックになりそうなサブシステムやプログラムに対して重点的に展開します。
- 関係する所属メンバでチームを作り、KJ法やブレーンストーミングを利用して要求品質の収集、整理を行います。
- 「想定品質要求の達成度合いを評価するための手段」、「評価をするためにはどんな手立てが必要か」などを自問してヒントを得ることも有効です。

なお、再発防止の施策への参考として、代用特性を利用したソフトウェア信頼性への工夫事例」を、IPAのホームページ(URL　http://sec.ipa.go.jp/index.html）へ掲載しておりますので、参照してください。

4.2.2　予防活動として整理・蓄積された情報の活用方法

(1) 新たなソフトウェア開発または保守作業への利用

要求品質を作り込む際に要求品質から代用特性に変換する方法は、高信頼ソフトウェア開発または保守作業においても有効な手法の1つとなります。

この「要求品質から代用特性に変換」する作業において、障害事例から得た「想定要求品質からの代用特性」を利用し、新たなソフトウェア開発における要求品質そのものの過不足および代用特性としてのソフトウェア機能の有無などを検証することが期待できます。また、第5章の「トレーサビリティ分析による設計検証」と併用することで、さらに高信頼ソフトウェアとしての効果が高まります。

(2) 定期障害予防診断の実施

医療の世界では血圧、BMI、血糖値などの閾値を定期健診により診断し、病気の兆候を知り、事前の病気予防に役立てています。同じように代用特性を定期障害予防診断項目として閾値を設けることにより、障害の兆候を知り事前の障害予防に役立てることが期待できます。

つまり代用特性を定量化（閾値を設定）し、定期障害予防診断のチェックリストとして利用することは、高信頼ソフトウェアを維持するための1つの方法です。

4.3 品質特性ごとの再発防止事例

情報システムの場合の障害発生要因は、コンピュータシステム、とりわけその構成要素であるソフトウェアの障害が起因となる場合が多く存在します。しかしソフトウェアの障害は、障害事象の可視化や原因分析が難しいといった側面があり、結果として実フィールドで発生する様々なシステム障害に基づく、再発防止策の立案など対策へのフィードバックが十分に行われないといったことに、つながっています。

予防活動においては、過去の障害事例から品質特性ごとにノウハウを分類・整理して活用していくことが有効な手段です。以下に代用特性とその予防活動における主な考慮点を整理しています。

なお、4.3.1項に品質特性ごとの代用特性事例を示し、その代用特性に関連する「障害・再発防止事例」(39事例)を4.3.2項で紹介します。

4.3.1 品質特性ごとの代用特性事例

品質特性ごとの代用特性事例は、表4-4～表4-9に示しています。なお、品質特性とそれに該当する「障害・再発防止事例」の対応は、表4-4～表4-9の「代用特性（最終）」に「障害・再発防止事例」の番号と、複数の再発防止策があるものはそれを枝番で記載しております。

(1) 機能性に関する予防活動

機能性には、品質副特性として合目的性、正確性、相互運用性、セキュリティ、機能性標準適合性があります。

実装機能の漏れ、誤りおよび他システムインタフェース不整合などの障害の要因は、機能性の問題から発生しており、その多くは個人の思い込みや、有識者への確認不足が根本原因のため、最大の予防策としては、暗黙知を作らず情報を漏れなく利害関係者に周知することになります。

しかしながら、大規模なシステム開発では、全参加メンバがすべての情報を共有するのは現実には不可能なため、情報の分類整理と伝達手段を工夫していく必要があります。

実際の事例では、レビュー等の開発プロセスの改善やマニュアル等の標準類文書の充実が効果を発揮しています。また、要件がプログラムの製造工程で実装されることを確認するためのトレーサビリティの工夫事例もあります。

相互運用性では、パイロット開発でシステム間の連携が確実に動作することを確認する手法も多用されています。

セキュリティにおいては、ネットワーク技術の進歩に対して、一般的に普及しているものや先進的な事例をいち早く察知して取り入れていくような活動が望まれます。

以降の表 4-4 に品質特性の機能性における代用特性事例対応表を示します。

表 4-4　品質特性（機能性）と代用特性事例対応表

代用特性					
代用特性(1次)		代用特性(2次)	代用特性(最終)		
品質特性	品質副特性		事例		
機能性	合目的性	妥当性(利用者側の要件)の確認(レビュー、テスト)に対するユーザーとベンダとの役割分担	1－③	2－①	3－①
			28－④	33－①	
		暗黙要求(要件記述なし)の確認手段に関する取り決め	1－①		
		仕様変更に対するユーザーとベンダーとの合意	4－①		
		ユーザ要件に対する設計書の必要十分性(トレーサビリティ)の検証	2－②	1－②	
	正確性	計算精度やデータ精度要求の実装確認	5－①		
		システム利用マニュアルの記述の正確性向上施策	6－①	7－③	
		検証(工程整合)確認に対するユーザーとベンダとの合意	8－②		
		上位設計書に対する下位設計書(またはプログラム、テスト仕様)の必要十分性(トレーサビリティ)検証	9－①		
	相互運用性	外部システムとの接続要求および仕様の確認(レビュー、テスト)に対するユーザーとベンダとの合意	5－②	10－①	11－①
		外部接続に対するデータ数、変換、編集(データ形式、コードなど)に関するコミュニケーション	11－②		
		相互接続する他のソフトウェアのバージョンアップによる影響に対する考慮	12－①		
		接続先外部システムとのコンテンジェンシープラン共有化	13－⑤		
	セキュリティ	データ暗号化対策	14－①		
		開発時でのデータ漏洩防止対策	14－②		
		セキュリティ事故発生に備えた、追跡可能性および監査容易性などの向上施策	14－③	15－①	
		セキュリティパッチなど脆弱性の予防対策			
		不正アクセス、不正ログインに備えたアクセス制御対策	16－①		
		なりすましのモニタリング、警告機能に関する工夫	16－②		
		データ損傷などのデータ保全対策	16－③		
		セキュリティ要件(実装検証含む)のユーザとベンダーとの合意	15－②		
	機能性標準適合性	機能に対応する規定(業務、内部統制、ISMS、国際会計など)および開発標準の適合監査対策			

(2) 信頼性に関する予防活動

信頼性の品質副特性は、成熟性、障害許容性（発生防止策）、障害許容性（拡大防止策）、回復性、信頼性標準適合性があります。

機能性に次いで多くの障害要因があり、予見的に障害を未然に防ぐことが必要となります。

予防事例の多くは、自社の過去の障害事例から学んだことや、世間一般で普及してきた対策等を、新たな開発に取り入れていくことが多く見られます。また、資源の使用状況等を定量的に監視し、危険な状況に至る前に障害発生を防止したり、影響が拡大化することを防ぐことも必要になります。

定量的管理をするにあたり、閾値を超えた場合のコンティンジェンシープランを設計段階で検討し、テストフェーズでそのプランに基づいて訓練を行うことも有効な手段となります。

以降の表 4-5 に、品質特性の信頼性における代用特性事例対応表を示します。

表 4-5　品質特性（信頼性）と代用特性事例対応表

代用特性					
代用特性（1次）		代用特性（2次）	代用特性（最終）		
品質特性	品質副特性		事例		
信頼性	成熟性	潜在的障害予測（又は潜在的障害低減）および障害解決状況の分析	17－①		
		仕様変更管理（傾向、内容分析）に基づく、仕様変更の制御	4－②		
		MTBF向上施策	18－①	30－③	
		「テストコスト」と「本番稼動後の欠陥による社会的影響や復旧にかかるコスト」とのトレードオフを考慮したテスト手法（テスト網羅率）の取り込み	2－③	8－①	35－③
	障害許容性（発生防止策）	イレギュラー処理の実装に関する工夫	18－②		
		障害を事前に検知するためのサブシステム間のトレース機能の工夫	17－②		
		障害に対応するためのテスト環境準備に関する工夫	19－①	20－①	37－③
		障害管理（傾向、内容分析）に基づく、障害の予防・是正処置の実施	21－③		
		運用時点での障害発生状況（傾向、内容分析）に基づく、障害の予防・是正処置の実施	21－②		
		ミスオペレーションの防止に関する工夫	13－①		
		システム資源（CPU、ディスクなど）の使用状況監視、警告に関する工夫	18－③	22－①	
	障害許容性（拡大防止策）	停止防止対策	20－②		
		稼働初期に起きる故障対策・分析に関する施策	23－②	24－①	
		取り扱い可能なデータ件数等システムのキャパシティオーバー時の誤動作防止に関する工夫	25－①	26－①	
		既存システム修正時における障害（リグレッションテスト漏れ、本番リリース不備による二次障害など）回避策	21－④		
		ハードウェアのアラーム対応に関する工夫	18－④		
		障害対応マニュアルに関する工夫			
		異常を検知し他システムなどへの影響を遮断する機能に関する工夫	27－①	28－①	
		予防訓練に関する施策	20－③		
	回復性	可用性（システムの稼働率、サービス提供時間、運転時間等の割合）向上施策	29－①	30－①	
		障害分類を配慮した障害回復時間短縮施策			
		サービス提供再開施策（復旧予測時間の配慮など）	11－③	21－①	33－③
		自動リカバリー機能など故障耐久能力向上施策	20－④		
		バックアップ機への切り替えに関する施策	18－⑤	31－①	
		広域・局所災害対策	30－②		
		要件定義時点におけるユーザとベンダー間のコンテンジェンシープラン共有化	19－②		
	信頼性標準適合性	信頼性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策			

(3) 使用性に関する予防活動

使用性の品質副特性は、理解性、習得性、運用性、使用性標準適合性があります。

携帯電話やインターネット予約のように、一般市民が使用するシステムが増加あるいは機能の多様化が進んでいる現状では、システムを操作する人のスキルは一様ではありません。

使用性の予防事例は、エラーメッセージの出し方などに新しい技術を導入し、使用者のスキルレベルに合わせて操作性を提供するような仕組みを導入しているケースもあります。

なお事務処理システムの統括部門では、キータッチの回数や操作時間等の定量的な測定項目から、さらなる工夫をしていくことも念頭に置く必要があります。

以降の表 4-6 に、品質特性の使用性における代用特性事例対応表を示します。

表 4-6　品質特性（使用性）と代用特性事例対応表

代用特性					
代用特性（1次）		代用特性（2次）	代用特性（最終）		
品質特性	品質副特性		事例		
使用性	理解性	ユーザインターフェイス（メニュー、アイコンなど）の工夫	13－②		
		利用者側の教育効果向上施策	6－②	23－③	
	習得性	ユーザレベル（初級、中級、上級）などを配慮した習得容易性向上施策	23－①		
		システム利用マニュアルおよびシステム運用マニュアルを判り易くするための工夫			
		ヘルプ機能の容易性などに関する工夫	13－③		
	運用性	利用者に対する操作のナビゲート、操作結果の確認のし易さに関する工夫	6－③	7－④	13－④
		システム運用の容易性向上、誤操作防止施策	28－②		
		インストールやバージョンアップの容易性（バージョンアップ後の確認の容易性、配布容易性を含む）向上施策	32－①		
		誤り修正（取り消し作業を含む）の容易性向上施策			
		誤操作からの回復性向上施策	7－①		
		利用時におけるメッセージのわかりやすさの向上施策	7－②		
		オペレータの介入操作に関する工夫	28－③		
	魅力性	少ないオペレーションで多くの処理を実現する工夫			
	使用性標準適合性	使用性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策			

(4) 効率性に関する予防活動

効率性の品質副特性は、時間効率性、資源効率性、効率性標準適合性があります。

予防事例としては、ハードウェアと技術の進歩をタイムリーに取り込んでいるケースがあります。また定量的に資源を監視し、予見的に対策を講じることが必要です。

以降の表 4-7に、品質特性の効率性における代用特性事例対応表を示します。

表 4-7　品質特性（効率性）と代用特性事例対応表

代用特性					
代用特性（1次）		代用特性（2次）	代用特性（最終）		
品質特性	品質副特性		事例		
効率性	時間効率性	非平常時（ピーク時、障害および災害時など）に対応するレスポンスタイム、スループット、ターンアウンドタイムの考慮点などに関する施策	25－②		
		非平常時（ピーク時、障害および災害時など）に対応する業務のスループット、デリバリータイムの考慮点などに関する施策			
	資源効率性	非平常時（ピーク時、障害および災害時など）に対応する資源（CPU、メモリー、伝送、入出力、設置条件（スペース、電源容量など））、体制などの考慮点に関する施策	26－②		
	効率性標準適合性	効率性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策			

(5) 保守性に関する予防活動

保守性の品質副特性は、解析性、変更性、安定性、試験性、保守性標準適合性があります。

以前のようにスクラップ＆ビルドで、再構築を重ねていくシステムの作り方から、最近では完成されたシステムを有効利用する保守形態が増え、それに伴い保守性に起因する障害も増加傾向にあります。また有識者の退職により、システムがブラックボックス化してしまうという悩みも増えてきました。

保守の場合は、現行システムを知ることが一番であり、予防事例としては保守用のドキュメントを整備することと、ログやトレース機能の充実等、機械的に解析できる仕組みの構築が目立ちます。

試験フェーズについても、前回でのテストデータ再利用等のテスト環境整備の工夫があります。本番移行前にリハーサルを十分行うことも有効な手段です。

以降の表 4-8 に、品質特性の保守性における代用特性事例対応表を示します。

表 4-8　品質特性（保守性）と代用特性事例対応表

代用特性					
代用特性（1次）		代用特性（2次）	代用特性（最終）		
品質特性	品質副特性		事例		
保守性	解析性	データログ実装、状況監視データ取得など活動記録保有能力に関する施策	33－②		
		診断機能実装および故障原因解析に関する工夫			
		ソフトウェアのトレース機能など解析容易性向上施策	17－③	29－②	
	変更性	変更履歴、構成管理などの変更制御に関する工夫	34－②		
		母体システムの構造化度、変更生産性など変更容易性向上に関する施策	35－②		
		保守ドキュメント、コメント率など変更容易性向上に関する施策	17－④		
	安定性	保守作業での欠陥混入是正に関する工夫	24－②		
		母体品質向上施策	17－⑤		
		システム保有データの整合性維持に関する施策	36－①		
		バージョンアップによるデグレード防止に関する施策	34－①	37－①	
		障害混入確率の低減に関する施策	35－①		
	試験性	本番リリースを保証するためのテスト範囲特定方法などに関する取り組み	9－②	21－⑤	22－②
		保守テストでのテストツール（自動再帰テストツールなど）、本番環境具備、標準テストセットの具備などのテスト環境整備に関する工夫	38－①		
		試験性に配慮したソフトウェアおよびデータの構造化、ならびに他システム接続方式に関する施策	5－③		
	保守性標準適合性	保守性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策	39－①		

(6) 移植性に関する予防活動

移植性の品質副特性は、環境適応性、設置性、共存性、置換性、再利用性、移植性標準適合性があります。

システムの移植は特殊性が多いため、汎用的な予防活動はあまり存在しません。前述の５つの品質特性と同様に、以下のような点を注意して個別に予防活動を検討する必要

があります。

・一般的に普及している技術や先進的な事例をいち早く察知する
・暗黙知を作らず利害関係者に情報を提供する
・前例の少ない取り組みはパイロット作業で事前に十分検証する
・定量的な管理により予見的に行動する
・コンティンジェンシープランを早めに設定し十分訓練する
・ログやトレース機能を組り込み、解析性を高める

以降の表 4-9 に、品質特性の移植性における代用特性事例対応表を示します。

表 4-9　品質特性（移植性）と代用特性事例対応表

代用特性					
代用特性（1次）		代用特性（2次）	代用特性（最終）		
品質特性	品質副特性		事例		
移植性	環境適応性	ソフトウェア（アプリケーション）の環境適用性向上施策（OS、ミドルウェア、DBMSなど）	37-②		
	設置性	移植時の設置作業の確実性向上施策（移植作業支援ツール(移植箇所特定)など）			
	共存性	同一環境下（同一サーバ、同一DBMSインスタンスなど）で、複数アプリケーションを共存させるための施策			
	置換性	制度変更などシステム機能を置換しやすくするための施策（部品化など）			
	移植性標準適合性	移植性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策			

4.3.2　代用特性に関連する「障害・再発防止事例」

ここでは表 4-4～表 4-9 で紹介した「代用特性」ごとに、39 の「障害・再発防止事例」を紹介します。以下に、「障害・再発防止事例」のなかで記載している項目の説明をします。

＜障害事例＞

過去、社会的に問題があったとしてマスコミが取り上げたビジネス・システムの障害事例から、再発防止策としてできる限りすべての品質特性（代用特性）を網羅するように、39 の障害事例を取り上げています。

＜障害発生元＞

ソフトウェアライフサイクルをソフトウェア開発、保守および運用のフェーズに分けた場合、取り上げた障害事例がどのフェーズにおいて起因した障害かを想定し、記載しています。

＜障害による影響度合い＞

障害発生時の対処方法および新たに構築するシステムの再発防止策などに関する水準を判定するために、システム障害による影響度合いを指標化し表示しています。詳細はコラム：「障害影響度指標」に説明しています。

＜根本原因＞

実際の根本原因は別にあることも考えられますが、ここでは予防活動としての再発防止策を機軸にしていますので、障害事例から再発防止策を考慮し根本原因を想定しています。

＜再発防止策＞

再発防止策は、高信頼化のための手法 WG 各委員からの起案を収集した資料編（代用特性を利用したソフトウェア信頼性への工夫事例）※ を参考に記載しています。

なお再発防止策は、根本原因との相関があるゆえ想定した根本原因によっては、新たな再発防止策が考えられますので、参考事例としてとらえてください。

※資料掲載 URL　http://sec.ipa.go.jp/index.html

障害・再発防止事例 (1)

障害事例(1)	業種	鉄道					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		

路線進入ルート切替プログラムに不備があり、列車が誤ったルートに進入する事象が発生した。
プログラム仕様誤り(要件との不整合)が原因である。

障害による影響度合い	障害影響評価指標値	5	

幸いにも人身事故は発生しなかったが、列車の運休、遅れが50本以上発生。影響利用者数数万人。

根本原因

要件自体が業界常識と受け取れることもあり、要件定義書上に記載されていなかった。また、テストケース設定は開発担当者任せとなっており、ユーザ視点でのテスト仕様(テストケース)が漏れ、テスト工程での欠陥抽出ができていなかった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	合目的性
代用特性(2次)	暗黙要求(要件記述なし)の確認手段に関する取り決め		

(1)要件に記載が漏れやすい下記の内容'b)'について、要件定義工程および設計工程の早い段階で要求を収集し明文化(記述)する。さらに、要件定義工程や設計工程のインプット情報として利用する。
a)実業務等の観察、インタビューやアンケートを利用した聞き取り、観察や聞き取り結果の網羅性チェックなどから暗黙知を形式知化すべき情報を収集する。
b)業界常識、顧客常識および顧客ビジネス標準となっている業務手順・規約など

(2)上記(1)にて形式知に変換された暗黙知[*1]については、第三者診断体制を組織化(第三者診断は、診断を行う第三者のスキル、知識に負うところが大きいことに留意する)し、妥当性を検証する。
a)妥当性検証は要件網羅、要件要素間矛盾および背景・スコープの不明確さなどの観点で行う。

[*1]:「個人的な経験により得られる、言語化しえない(し難い)知識」「言葉で学ばなくても視覚的、体感的に覚えることで伝達されるノウハウ、職人技」などの解釈があるが、ここでは「特定の集団内で共有されている、かつ明示されていない知識、ノウハウ、常識」を示す。

【再発防止策②】

特性	機能性	副特性	合目的性
代用特性(2次)	ユーザ要件に対する設計書の必要十分性(トレーサビリティ)の検証		

要件定義書からの要件一覧化、各要件ごとにIDの付与および以降の設計書ならびに試験仕様書において、このIDをベースに詳細化(IDの枝番付与等)しながらトレーサビリティを確保し、矛盾の発見を行う。

【再発防止策③】

特性	機能性	副特性	合目的性
代用特性(2次)	妥当性(利用者側の要件)の確認(レビュー、テスト)に対するユーザとベンダとの役割分担		

システムテスト仕様書の作成、システムテストはユーザ中心で行うことにより、業務面での確認を行う。
a)開発者が想定できない実業務に沿ったテストが可能となり、残存欠陥摘出に効果がある。

障害・再発防止事例 (2)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		

特定銀行の受取口座に振り込まれないトラブルが発生した。プログラム改修時の漏れ（上限桁数の対応漏れ）が原因である。

障害による影響度合い	障害影響評価指標値	4

障害の対応は即時に行われたが、振込み不可となった金額も大きく（1千億以上）、監督省庁からの指導も発生した。

根本原因

要件定義書に桁拡張の必要を記載すべきであったが、エンドユーザ、システム部門のどちらが定義すべきか決まっていなかったため、要件定義書レビュー時点で不備を摘出できなかった。後続の設計レビューはエンドユーザ不在で行われており、要件漏れを摘出できないまま後続の開発を進めてしまった。また、本番を想定したテストケースの設定が不十分なままテストを実施していた。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	合目的性
代用特性（2次）	妥当性（利用者側の要件）の確認（レビュー、テスト）に対するユーザとベンダとの役割分担		

ユーザ要件定義工程において、業務主管部門とIT部門のそれぞれの部署が責任をもって作成しなければならないドキュメント類を定め、要件定義における確認・検討ポイントや記載要領等を解説したチェックリストを準備し運用する。
a）プロジェクト参加者の役割分担、責任範囲が明確となるため、作業分担の曖昧性を無くし、また重複を防止するとともに上流（当該）工程を効果的に進めることができる。

【再発防止策②】

特性	機能性	副特性	合目的性
代用特性（2次）	ユーザ要件に対する設計書の必要十分性（トレーサビリティ）の検証		

要件定義書・基本設計書にユーザ要件がすべて盛り込まれているか、誤解している内容がないかを観点とした内部レビューを実施した後、最終的にユーザ・ベンダ両者参加の設計書レビューにて確認する。
a）内部レビュー観点に加えたことにより、設計書作成者も要件の漏れを意識することになり、品質向上に役立つ。

【再発防止策③】

特性	信頼性	副特性	成熟性
代用特性（2次）	「テストコスト」と「本番稼動後の欠陥による社会的影響や復旧にかかるコスト」とのトレードオフを考慮したテスト手法（テスト網羅率）の取り込み		

本番稼動後のリスクを勘案した試験項目を抽出し、テストを実施する。リスクに応じテスト実施要否を判定する。

障害・再発防止事例 (3)

障害事例	業種	行政					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		

交付金算出システムの欠陥により、全国の自治体（市町村）に交付する金額を誤って算定。交付金額を過少に支払う障害が発生。省令に基づいたプログラム仕様書に仕様漏れがあり、金額を算定するロジックが誤っていた。

障害による影響度合い	障害影響評価指標値	5	

支払い不足金額は数千億円にのぼる。自治体の信用失墜は免れない。

根本原因	

要件定義書の記載に不備があった。エンドユーザ、システム部門のどちらが定義すべきかが決まっておらず、レビュー時点で不備を検出できないまま設計を進めていた。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	機能性	副特性	合目的性
	代用特性（2次）	妥当性（利用者側の要件）の確認（レビュー、テスト）に対するユーザとベンダとの役割分担		

（1）成果物に対するユーザレビューを実施し、成果物の記述水準・記述密度を事前にユーザと合意することにより、レビュー品質の向上を図る。
a）成果物の記述水準や記述密度を標準化することで、レビュー担当者によるレビュー結果のバラつきが抑えられる。また、レビュー観点や指摘レベルも明確になるため、効果的なユーザレビューの実施に寄与する。

（2）ユーザ要件定義工程において、業務主管部門とIT部門のそれぞれの部署が責任をもって作成しなければならないドキュメント類を定め、要件定義における確認・検討ポイントや記載要領等を解説したチェックリストを準備し運用する。
a）プロジェクト参加者の役割分担、責任範囲が明確となるため、作業分担の曖昧性を無くし、また重複を防止するとともに上流（当該）工程を効果的に進めることができる。

障害・再発防止事例 (4)

障害事例	業種	金融(カード)					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○	○	

制度対応後に本番リリースしたクレジットカード・キャッシングのシステムに不具合があり、キャッシングが利用不能状態になった。制度対応と同時に取り込んだ機能変更にて、特定トランザクションデータで異常終了となる障害であった。納期最優先の改良開発であるが、仕様変更の取り込みに関して開発ベンダーともめてしまい、短期間での対応を余儀なくされていた。テスト検証が不十分であったことが判明している。

障害による影響度合い	障害影響評価指標値	5	

キャッシング利用者の影響数は公表されていないが、すべて対処するまでに数日間を要す。

根本原因

仕様変更に関する取り決めや調整、対応範囲決定のプロセスに問題があった。
①仕様変更に関する関係者間の合意遅延が、短期間対応を誘発していた。
②仕様変更の制御ができていなかった。(対処可能な量を超える仕様変更を取り込もうとしていた。)

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	合目的性
代用特性(2次)	仕様変更に対するユーザーとベンダーとの合意		

大規模プロジェクトでは、要件変更は直接の関連部門だけでなく全体とりまとめ事務局の職位者の承認を必要とし、要件変更によるコスト、工数およびスケジュールへの影響を全体として管理する仕組みをとる。
a)仕様変更の要否について検討する場として事務局を設置し、開発者の工数負担を軽減させる。
b)費用対効果を明確にすることで、予算内での開発を推進することができる。
c)仕様変更(変更判定、変更見積方法、対応時期、精算方法など)については、事前に文書による合意を前提とすることで、後の揉め事を回避する。
・事前に文書による合意を行ったうえで対応するため、後になって揉めることがなくなる。
・仕様変更の進捗についても逐一報告しているため、本体への影響についても合意を得ることができる。

【再発防止策②】

特性	信頼性	副特性	成熟性
代用特性(2次)	仕様変更管理(傾向、内容分析)に基づく、仕様変更の制御		

(1)仕様変更に関する発生、回答の推移をモニタリングするとともに、仕様変更内容を分析し、内容・観点別に分類しチェックリスト化する。
a)仕様変更を分析し横展開を図ることにより、類似機能に関し要件の変更が必要か判断し後工程での手戻り工数の発生を抑止する。

(2)仕様変更の発生状況に特定機能への偏りが見られる場合、対応策を検討する。対応策が解決するまで、開発を一時中断することも併せて検討する。
a)仕様変更を分析し、偏った機能あるいはサブシステムに仕様変更が頻発している場合、当該機能やサブシステムについて再度ユーザと機能調整、合意する仕組みを持つことで、後工程での仕様変更の発生を抑止する。
b)プロジェクト管理者は、仕様変更要望の取り込みによって想定するコスト、品質および納期面での懸案事項を整理し、対処可能な範囲での取り込みとなるよう調整する。

障害・再発防止事例 (5)

障害事例	業種	通信					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		

携帯電話の利用料金請求にて、実際の請求額よりも1桁もしくは2桁多い金額を印字し請求書を発行してしまうといった障害が発生。料金管理を行うシステムと料金結果を受領するシステム（他社の請求書印刷システム）とのインタフェースで、特定の利用者にて考慮すべき小数点が無視された。

障害による影響度合い	障害影響評価指標値	5	

発生件数は数万件にのぼり、全国各地の利用者へ影響を与える。利用者からのクレームに対する謝罪も必要となり、信用失墜となった。

根本原因	

インタフェース確認やテスト実施の必要性は認識していたが、相互での仕様確認やテスト実施負荷が重いこと、これまで接続実績があることを理由に、不十分な仕様確認やテスト密度で開発をすすめていた。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	機能性	副特性	正確性
	代用特性（2次）	計算精度やデータ精度要求の実装確認		

計算途中の丸め誤差、集計時最終桁扱い誤りなどの微妙な誤差を防止するため、以下の手段を講じる。
a）起点日の考え方(片端計算、両端計算等）も含め、計算精度については、設計書に順序、計算途中での端数の取り扱いや桁落とし方法を明記して、ユーザと合意をとる。

【再発防止策②】	特性	機能性	副特性	相互運用性
	代用特性（2次）	外部システムとの接続要求および仕様の確認（レビュー、テスト）に対するユーザとベンダとの合意		

外部接続時は、開発の早い段階でプロトタイプを作成し、疎通確認を行う。
a）開発に先立ち疎通確認を行うことにより、開発の早期段階で、接続先とのインタフェースミス、環境設定ミスの洗い出しが可能となる。

【再発防止策③】	特性	保守性	副特性	試験性
	代用特性（2次）	試験性に配慮したソフトウェアおよびデータの構造化、ならびに他システム接続方式に関する施策		

アプリ基盤として、データベースマネジメントシステムなどに依存しないソフトウェアの構造化を行う。
a）PCなど簡易で軽量な環境にて下流テストが実施可能となり、試験負荷の軽減、テスト密度向上に寄与する。ただし本番環境とかけ離れている場合は、環境差異により障害が摘出できない可能性もあり、本番環境での確認が必要である。

障害・再発防止事例 (6)

障害事例	業種	保険					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○	○	

新商品の追加に伴いコールセンターシステムのユーザインタフェースを一部変更したところ、一部オペレータによる誤操作（入力）が多発。後日、顧客からの請求額問い合わせで入力誤りによる過剰請求が判明するといった障害が発生した。誤入力直後では処理が正常終了（手数料控除サービスが効いていない状態で終了）しており、発見が遅れた。

障害による影響度合い	障害影響評価指標値	5	

1ヶ月間での利用者数万人に対し、過剰請求がなされており、後日返済を実施。返済額は未公表。

根本原因

ユーザインタフェースの変更に対しマニュアルの改訂が行われておらず、オペレータの特性（錬度、経験等）に考慮した教育も行われていなかった。また、ユーザの誤入力を抑制する対策がとられていなかった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	正確性
代用特性（2次）	システム利用マニュアルの記述の正確性向上施策		

（1）改良開発後のユーザ誤操作や誤入力が散見された事実を認識し、改良開発にて「マニュアル改訂作業」を必須とする。改訂作業により、ユーザの誤操作や誤入力が抑制できる。

（2）マニュアル作成後、実環境を使用した試験によりマニュアルの精度を検証する。

【再発防止策②】

特性	使用性	副特性	理解性
代用特性（2次）	利用者側の教育効果向上施策		

（1）利用者の教育のため、操作訓練等のトレーニングを実施する。
a）重要データの入力等、通常はなかなか実践できない業務に絞って入力練習のメニューを構築する。
b）一日の流れにそって全てのノーマル業務を実行できる試験モードを構築する。
c）本番環境に研修用DBを構築。本番機よりデータを抽出し小規模のデータベースを構築し、本番機同等の業務処理のトレーニングを可能とする。
d）「研修」をシステムのサービスメニューとしてエンドユーザに提供する（基本的な操作パターンを準備し、本番データにはアクセスせず、ダミーの帳票出力などを行う）。

（2）利用者の運用補助のため、下記対応（工夫）を行う。
a）ヘルプデスクの設置を実施する。
b）運用手順の省略（自動化製品の利用推進）とマニュアルの充実を図る。
c）共通ユーザインタフェース機能（画面内の共通レイアウト部分のデザイン、利用を許可する画面コンポーネント、項目配置ルール、など）を利用者が直感的に理解しやすい一貫性のある規約・基準として確立する。

【再発防止策③】

特性	使用性	副特性	運用性
代用特性（2次）	利用者に対する操作のナビゲート、操作結果の確認のし易さに関する工夫		

メニュー体系について、そのとき使えるメニューだけをアクティブ状態にする工夫を行う。
（例 'A' 本番・試験選択→'B' 業務開始→'C' 情報確認→'D' 入力→'E' 業務終了という流れが必要であった場合、'A' を実施すると'B' がアクティブになり、'B' を実施すると'C' がアクティブになり. . . という仕組みを構築する）
a）利用手順が明確になるため、操作ミスが低減する。

障害・再発防止事例 (7)

障害事例	業種	保険					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○	○	

一般市民の保険料徴収について、既存システムの点検により将来誤徴収につながる障害が判明したとの発表があった。誤徴収となる原因は担当者の「データ変更時登録誤り」である。

障害による影響度合い	障害影響評価指標値	4	

事前検知となったため、一般市民への影響は少ない。本障害に伴う謝罪（お知らせ発行）に限らず、その他システムの一斉点検を行うなど突発的な業務が発生しているため、他システム案件の凍結（新種品の導入遅れ）が起こり、コストや売上の面で企業業績を圧迫している。

根本原因	

誤操作を誘発するようなインタフェースであること、誤操作に対し警告や注意喚起を行う仕組みが不十分であったことは否めない。また、ユーザ向けマニュアルについても、システムの新規作成時に整備された以降、機能追加や制度変更等の改修内容が正しく反映しきれていないといった課題も判明している。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	使用性	副特性	運用性
	代用特性（2次）	誤操作からの回復性向上施策		

（1）誤操作からの回復を目的に、削除要求時には実際に削除せず、削除要求の取消し機能を提供する（オンライン終了後、取消し要求のない削除要求のみ削除する）。
a）入力データのトレーサビリティが明らかになり、誤操作による障害発生の防止に役立つ。

（2）一般に入力エラー項目は画面内に複数個発生するため、下記のような画面設計基準を設けることにより入力ミスが明確になり、操作性が向上する。
a）複数個のエラー項目をすべて赤表示する⇒エラー項目の視認性が向上する。
b）赤表示されたエラー項目を操作（カーソル位置付け）するタイミングでエラー説明メッセージを動的表示する⇒修正入力が容易となる。

【再発防止策②】	特性	使用性	副特性	運用性
	代用特性（2次）	利用時におけるメッセージのわかりやすさの向上施策		

登録時のメッセージ表示に関して、以下の基準を設ける（登録時に発行されるメッセージによりエラー箇所に理由が明記されるため、誤操作によるダーティデータの発生件数が減少する。また、問い合わせ自体も減少し、運用部門の負荷も低減する）。
a）利用者の理解しやすい表記を用いる。（システム管理者的表現は避ける。）

【再発防止策③】	特性	機能性	副特性	正確性
	代用特性（2次）	システム利用マニュアルの記述の正確性向上施策		

（1）改良開発後のユーザ誤操作や誤入力が散見された事実を認識し、改良開発にて「マニュアル改訂作業」を必須とする。改訂作業により、ユーザの誤操作や誤入力が抑制できる。

（2）マニュアル作成後、実環境を使用した試験によりマニュアルの精度を検証する。

【再発防止策④】	特性	使用性	副特性	運用性
	代用特性（2次）	利用者に対する操作のナビゲート、操作結果の確認のし易さに関する工夫		

お客様が使用する機能の改定時には、オーナーだけでなくヘルプデスクやコールセンターにも要件を確認し、意見を取り入れ、わかり易い機能とする。

障害・再発防止事例 (8)

障害事例	業種	行政					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
				○	○		

一般家庭に送付する納税通知書の作成に関してプログラム改修不備があり、当該帳票が顧客に届かないという障害が発生。ユーザニーズを取り込んだ結果、当初計画したテスト期間では不足が生じていたが、予算や納期の制約で体制強化や期間延長は実現できなかった。また、テスト密度の実績は、予定を下回るものとなっていた。

障害による影響度合い	障害影響評価指標値	4	

一般家庭への謝罪のほか、改修システムの品質担保作業を実施。結果システムの最終リリースが数ヶ月遅延するとともに予算の2.5倍のコストがかかる。

根本原因	

テスト負荷と品質（障害発生時のリスク）とのトレードオフについて、予算制約のみ先行し、議論されていなかった。

品質特性に準拠した再発防止策	

【再発防止策①】

特性	信頼性	副特性	成熟性
代用特性（2次）	「テストコスト」と「本番稼動後の欠陥による社会的影響や復旧にかかるコスト」とのトレードオフを考慮したテスト手法（テスト網羅率）の取り込み		

システムの各機能（画面、帳票、ファイル・インタフェース）について、障害が発生したときに発生するリスク（機会損失なども含む）とテスト項目網羅率に基づくテストコストの比較表を作成し、機能ごとにテストの軽重を決定しテスト戦略に反映する。

a）機能ごとに本番障害による社会的影響度合い（金額）を設定し、目標のテスト密度を設定する。
開発コスト削減が必要な場合も、当該影響度が高い機能については十分なテストを実施する
（＝テストコストは落とさない）。

【再発防止策②】

特性	機能性	副特性	正確性
代用特性（2次）	検証（工程整合）確認に対するユーザとベンダとの合意		

（1）ユーザにフェーズごとのプロセス品質（計画・実績）、残存欠陥等含めたカットオーバ基準を公開し事前に合意する。

a）次工程に着手する前に品質の計画値と実績値の差異分析を行い、ユーザと合意形成を行うことができる。
差異が大きい場合、レビュー精度の見直し等早期対策を講じられるため、結果として高品質の製品の提供に繋がる。

b）残存欠陥率「残存欠陥件数/KLOC等」（テスト網羅率「テスト項目数/KLOC等」）についてユーザと合意を得ることにより、品質目標とテストコストとの整合を図る。

（2）工程完了報告会を実施し、開発規模、試験密度、レビュー指摘件数、障害件数をユーザに報告。
ユーザより承認を得ることで次工程の作業に着手する。当該工程にて保留となった事項については申し送り事項として、別途管理し解決の都度ユーザに報告する。

a）次工程に着手する前に開発規模の増減等についてユーザと事前合意できるため、開発費用の増減もしくは機能の削減を調整することができる。

b）必要以上に指摘件数・障害件数が多い場合、次工程に入る前に製品に対するレビュー強化を実施することで、品質向上を図ることができる。

c）未決事項を共有することができ、解決に向けた動きがスムーズに行うことができる。

障害・再発防止事例 (9)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		

株式注文システムの改修にて、注文を証券取引所に取り次ぐ機能が不正動作となる障害が発生した。プログラム改修内容の誤りが原因であった。要件定義書と詳細設計書の記載内容がそもそも不整合となっていた。

障害による影響度合い	障害影響評価指標値	3	

障害復旧が短時間であったため、大きな影響は与えていない（実害は公表されず）とのことであるが、株式注文ゆえに大口取引などへの影響も想定される。

根本原因

本番リリースに対する機能検証が不十分であることが原因。本改修は小規模であるため開発期間が短く、結合テスト実施（ベンダ主体）後に本番移行しており、ユーザ確認テスト（システムテストや運用テスト）が省略されていた。また、ユーザは要件定義書レビューにより安心してしまっており、その後の詳細設計書や結合テスト仕様書のレビューには参加していない。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	正確性
代用特性（2次）	上位設計書に対する下位設計書（またはプログラム、テスト仕様）の必要十分性（トレーサビリティ）検証		

仕様書と試験仕様書、概要設計書と詳細設計書にトレーサビリティマトリクスを活用。さらに網羅性検証のためにチェックシートに基づく要件定義の具体化時に確認・検証するツールを導入し確認する。
a）ツールを使用することで実現工数はあまり要していないが、トレーサビリティが確保されるため、機能の漏れがなくなり高品質の製品提供が実現できる。

【再発防止策②】

特性	保守性	副特性	試験性
代用特性（2次）	本番リリースを保証するためのテスト範囲特定方法などに関する取り組み		

（1）障害対応時を含め、本番（同等）環境でテストする項目を用意して、予想結果と一致することを確認してからリリースする。

（2）本番環境とテスト環境の差異を明確化した上でテストを実施する。
a）本番環境とテスト環境との差異を明確にすることにより、本番環境でしか確認できない項目の扱いについて、ユーザと事前に調整することができ、対象項目に関わる箇所を重点的にレビューするなど、欠陥除去に向け早期の対応を行うことができる。

障害・再発防止事例 (10)

障害事例	業種	メール・サービス					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		

グローバルなメール・サービスシステムが利用不可能になる障害が発生。個人向けおよび企業向けを問わず世界規模でメールが送受信できなくなる障害であった。導入を進めていたデータセンター用新基盤ソフトの障害対応機能と自社システムの双方に不備があり、データセンターで発生した障害をきっかけに自社システムの欠陥が顕在化した。

障害による影響度合い	障害影響評価指標値	5	

世界レベルでの使用停止となり、影響は数千万人に上ると想定される。自社信用失墜は否めない。

根本原因	

接続先システム（データセンター側）の新基盤ソフトウェアの障害対応機能について、事前検証が不十分であったと考えられる。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	機能性	副特性	相互運用性
	代用特性（2次）	外部システムとの接続要求および仕様の確認（レビュー、テスト）に対するユーザとベンダとの合意		

外部接続時は、プロトタイプを作成し早い段階で疎通確認およびインタフェース確認を行う。
a）外部接続での課題を早期に発見することが出来るため、他開発に影響を与えることが少なく、
コストダウンにも寄与する。

障害・再発防止事例 (11)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		○

ネットバンクで受付けた一部の予約振り込みが処理できない障害が発生。他の銀行や支店向けの振込み要求時に一部のデータが欠落してしまっていた。障害は直ちに検知されたが、データ欠落時のリカバリ作業に手間取り復旧が大幅に遅延し、ネットバンク側のシステムを1営業日停止することになった。

障害による影響度合い	障害影響評価指標値	5	

ネットバンクで受け付けた約2万件の振込みが即時処理できず、対応の完了に1営業日を費やした。ネットバンク使用停止に伴う影響は公表されなかった。

根本原因

外部接続システム間の確認が不十分なため、データ編集時の不備を摘出できていなかったと考えられる。また、リカバリ作業の遅延については、不測事態への備えが不十分であったと考えられる。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	相互運用性
代用特性（2次）	外部システムとの接続要求および仕様の確認（レビュー、テスト）に対するユーザとベンダとの合意		

外部システムとの接続仕様確認では、相手側のシステムの理解度不足や組織間での文化、標準化の違いにより仕様漏れを引き起こす可能性が高い。以下の対策を実施することで、レビューおよびテスト品質を向上させる。

a）レビュー時には双方の有識者を参加させ、双方のシステムで仕様の妥当性を検証する。
b）接続双方で洗い出したテストケースは統合せず、個別ケースとして実施する（ケースの統合により双方のシステム固有の確認項目が欠落してしまうことを回避する）。

【再発防止策②】

特性	機能性	副特性	相互運用性
代用特性（2次）	外部接続に対するデータ数、変換、編集（データ形式、コードなど）に関するコミュニケーション		

外部システムのコミュニケーションミス（確認漏れ）の防止、および障害発生時に迅速なリカバリ処理を実現すべく、以下の対策を実施する。

a）外部インタフェース確認内容（データ仕様、受渡時期、順序、編集方法等）をチェックリスト化し、設計工程時に接続双方で突合せチェックを実施、認識齟齬の早期発見に努める。
b）障害発生時の対応について役割および手順を整理（コンティンジェンシープランを作成）し、利害関係者の合意を得る。
c）迅速なリカバリ処理を実現するため、コンティンジェンシープランの各ケースに合わせ接続双方の回復手順を詳細化し、テスト時に訓練を実施する。

【再発防止策③】

特性	信頼性	副特性	回復性
代用特性（2次）	サービス提供再開施策（復旧予測時間の配慮など）		

障害発生時の対応については、業務回復優先とした体制、手順を準備する。障害解析チームと業務回復チームを分けて対応することにより、重複した作業を行わず効率的な回復を行う。

障害・再発防止事例 (12)

障害事例	業種	鉄道					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
					○	○	

自動改札機が動かない障害が発生。 特定ケースの利用者に関してのみ発生するが、電源を切ってしまう処理につながっており、機械自体が動かず一般利用者に影響を及ぼした。

障害による影響度合い	障害影響評価指標値	5	

数百万人に影響（自社売上影響額は不明）

根本原因	

自動改札機組み込みソフトウェアの潜在バグで、接続先システムのバージョンアップに伴い発覚した。接続先システムのバージョンアップに対し、本番リリース前の動作確認テストは実施していなかった。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	機能性	副特性	相互運用性
	代用特性（2次）	相互接続する他のソフトウェアのバージョンアップによる影響に対する考慮		

（1）ミドルウェア製品のバージョン変動時（アップ／マイナーアップ）には、そのミドルウェアを前提とした製品の非互換を確認する。
a）同一環境にあるソフトウェアのバージョンアップ情報を関連箇所に通知する仕組みを構築し、
独断でのリリースは行わない。

（2）相互接続するソフトウェアのバージョンアップ時は必ず相互に情報連携し、接続インタフェースの見直しを行い同期を取ってバージョンアップする。またバージョンアップ後の動作確認は自社システム側の改修有無にかかわらず必須とする。
a）バージョンアップする内容により影響を受ける範囲を特定できるため、ユーザとの調整、
合意形成が容易となり、早期に対策を取ることができる。

障害・再発防止事例 (13)

障害事例	業種	小売					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○		○

Webシステムに移行した販売店向け発注システムにて、システム移行直後にシステム間のデータ転送部に不具合が発生し、発注データの受信が行われなくなる事象が発生した。不正データの登録に起因し、転送データ編集時に不正処理が行われたためである。接続会社間で障害対応時の各種調整に手間取り、最終的なシステム復旧は1週間程度かかっている。

障害による影響度合い	障害影響評価指標値	5	

システムダウンにともなう影響は一千億円以上。消費者へのお詫びに値引きキャンペーンも実施している。

根本原因

不正データ登録を抑止する仕組みが考慮されていなかったこと、利用者へのユーザビリティへの配慮（ユーザインタフェース変更時）がなかったことが、障害を誘発した原因と考えられる。なお、本システムの接続先との間でも、有事に関する取り決めや有事での対応範囲などが整理できておらず、コンティンジェンシープランが共有されていない課題も発覚している。

品質特性に準拠した再発防止策

【再発防止策①】	特性	信頼性	副特性	障害許容性（発生防止策）
	代用特性（2次）	ミスオペレーションの防止に関する工夫		

（1）取引の重要度に応じて、取引成立に必要なプロセスを3段階に区分して、画面操作を実装する。
　'A' 低ランク：操作者のみで取引成立
　'B' 中ランク：操作者＋上位権限者の確認入力が必要
　'C' 高ランク：異例取引（赤残になる取引など）はさらに上位者権限の承認入力が必要

多数の取引（アプリケーション）で対応が必要なので、全取引に共通機能として提供し、これを業務共通機能に組み込んで、個別のアプリケーションでは意識しないようにする。また取引毎にAorBorCを外部テーブルに登録するように、チェック条件の変更はプログラム修正を不要とする。
あわせて、誤操作防止対策として、ペアオペレーションも以下のように対処する。
　'A' 担当者申請⇒上席による承認登録を必須とする
　　重要取引は、入力者と確認者による2重オペレーションを必須とする
　'B' 更新ファイル、設定ファイル等重要作業時にペアオペレーションを必須とする
責任範囲が明確となり、複数人での相互チェック機能も働くため、操作ミスによる障害の削減に寄与する。

障害・再発防止事例 (13)（続き）

【再発防止策②】

特性	使用性	副特性	理解性
代用特性（2次）	ユーザインタフェース（メニュー、アイコンなど）の工夫		

（1）ユーザインタフェースの視点に立ち、下記対策（工夫）を実施する。
a）開発のベースとなるフレームワークは、画面制御内容に制限を設ける。当初から、
その制限を踏まえた画面設計を行い、業務に影響がないことを確認しつつ設計を進める。
b）イレギュラーな操作／入力に対する適切なメッセージの表示（利用者がメッセージから誤った
入力／操作を把握可能であること）を実施する。
c）各業務画面の設計に先立って、共通ユーザインタフェース機能（画面内の共通レイアウト部分の
デザイン、利用を許可する画面コンポーネント、項目配置ルールなど）について、当該システムに
相応しいユーザビリティを実現するための設計規約および設計基準を確立させる。

（2）ユーザインタフェースに関連するドキュメントやツールなど情報発信する体制を確立し、Webシステムのユーザビリティ向上に向け以下の対策を講じる。
a）ユーザビリティの評価手法として、「ヒューリスティック評価」を採用する、
b）さらに、特殊な技術を持つユーザビリティ評価の専門家でなくても簡単に画面を見ながらチェック
できるような、”ユーザビリティチェックシート”を策定する。
c）ユーザビリティを高めるためのポイントを体系的に整理し、基礎知識さえあれば誰でもが
ユーザビリティを評価できるようにする。

【再発防止策③】

特性	使用性	副特性	習得性
代用特性（2次）	ヘルプ機能の容易性などに関する工夫		

監視照会画面での表示方法について、ヘルプ機能を選択しなくともカーソルがあたるとその項目名称を表示させるような工夫を行う。
a）画面表示エリアを効果的に使用しているため、表示データが多量なときに有効であり、説明文も
あわせて表示されるため、コードのみのときに比べ、操作ミスが低減する。

【再発防止策④】

特性	使用性	副特性	運用性
代用特性（2次）	利用者に対する操作のナビゲート、操作結果の確認のし易さに関する工夫		

メニュー体系について、そのとき使えるメニューだけをアクティブ状態にする工夫を行う。
（例 ’A’本番・試験選択→’B’業務開始→’C’情報確認→’D’入力→’E’業務終了という流れが必要であった場合、’A’を実施すると’B’がアクティブになり、’B’を実施すると’C’がアクティブになり．．．という仕組みを構築する）
a）利用手順が明確になるため、操作ミスが低減する。

【再発防止策⑤】

特性	機能性	副特性	相互運用性
代用特性（2次）	接続先外部システムとのコンティンジェンシープラン共有化		

（1）開発を同時期に行う接続外部システムが存在する場合、コンティンジェンシープラン作成時はすべての利害関係者が参加し作成することにより、対応策、担当範囲を明確にする。
a）すべての利害関係者が対応策について合意することにより、早急な対応を取ることができる。

（2）多数の利害関係者が存在する場合、他者の進捗状況により影響を受けることが予想されるため、可能であれば、要件定義の段階より他者を巻き込んだ形でプロジェクトを進めていく。必要により、全体を統括するプロジェクト管理チーム（ユーザ主体）を設けることも検討し、提言する。

＜狭義での事例＞
汎用機／オープン機が連携している場合にオープン機がダウンしていると、取引に関する全てのサービス提供が停止するような構成になっていることがある。取引の全面停止を避けるために、オープン機がダウンしていても、取引の予約だけは出来るようにした。
a）取引に関する全てのサービス提供が停止すると、ビジネス上の損害は多大になる。当該対策を
実装することで、当該リスクを低減できる。

障害・再発防止事例 (14)

障害事例	業種	流通					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		

開発中にシステムテストデータ紛失事故が発生。テスト利用目的でメディアに格納したのちメディアを紛失。対処で混乱し、事態把握に長時間を要してしまった。メディア内には取引先の部署名、電話番号、担当者氏名情報がおよそ3000件格納されていたが、個人情報をマスキングしていない過去データも含まれていた。パスワード設定は行われていた。

障害による影響度合い	障害影響評価指標値	4	

幸い顧客からの被害の情報は寄せられなかったが、紛失対象者へのお詫びが発生し、信用失墜となった。

根本原因

本番データの参照および流用については、アクセス権限の設定等、セキュリティ対策を進めている最中であったため、メディアに格納していたシステムテストデータの状態（マスク範囲や状況）が正確に把握できていなかった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	セキュリティ
代用特性（2次）	データ暗号化対策		

特に高い機密性が要求されるシステムにおいて、パスワードの暗号化を2段階とする（基盤（DBI’データベースインタフェースソフト’）で暗号化した上に、更に各業務で暗号化しテスト環境に展開する）。
a）両方の暗号化ルールを知る人間は、情報資産管理責任者等に制限しているため、
パスワードの漏えいリスクが大幅に低減できる。

【再発防止策②】

特性	機能性	副特性	セキュリティ
代用特性（2次）	開発時でのデータ漏洩防止対策		

情報セキュリティ管理策（本番機データの取り扱いルール、個人情報等機密データのマスキング、持ち出し時のデータ所管部署の許可、証跡の確保等）を構築する。

【再発防止策③】

特性	機能性	副特性	セキュリティ
代用特性（2次）	セキュリティ事故発生に備えた、追跡可能性および監査容易性などの向上施策		

各システム毎に異なっていたセキュリティログ（ID、資源アクセス記録etc）を定義、ログフォーマットを標準化し運用等社内ルールを規程する。セキュリティログ取得保管のための運用管理支援システムを構築しセキュリティログの一元的な保管・管理を行う（各部門サーバのセキュリティログを定期的に取得し保管する）。
a）バラバラであったログ情報を統一的に管理することで、進入経路や手段等の特定作業が向上する。

障害・再発防止事例 (15)

障害事例	業種	小売					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○				

Webサイトから顧客情報（2万件弱クレジット情報を含む）への不正アクセスが発生。アクセス検知が遅れ、アクセス範囲が拡大していた。結果、顧客向けに電子メール、電話および書簡による対処を余儀なくされる。

障害による影響度合い	障害影響評価指標値	5	

顧客へのお詫びおよび対処（損害額不明）。企業信用の失墜。

根本原因

技術的な対策（不正侵入検知システム（IDS）や不正侵入予防システム（IPS）の導入検討）が不足していた。また、ISMSの運用においてリスクアセスメントが不完全で、保有システムの脆弱性に対し客観的な評価（有識者（第三者）の意見・評価）を受けるべきであった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	セキュリティ
代用特性（2次）	セキュリティ事故発生に備えた、追跡可能性および監査容易性などの向上施策		

（1）特に機密性の高いシステムの場合、以下のような対策を実施する。
a）不正アクセスを検知する仕組み：不正な通過パケットを自動的に発見するIDS／IPS等の措置を講じる。
b）重要な情報設備については、他の区画との明確な区別を実施する。
c）入退室記録の保存を行う。
d）監視カメラの稼働および監視映像保存を実施する。
e）警備員を常駐させる。（情報漏えいリスクの低減、不正アクセスの防止）

（2）各システム毎に異なっていたセキュリティログ（ID、資源アクセス記録etc）を定義、ログフォーマットを標準化し運用等社内ルールを規程する。セキュリティログ取得保管のための運用管理支援システムを構築しセキュリティログの一元的な保管・管理を行う（各部門サーバのセキュリティログを定期的に取得し保管する）。
a）バラバラであったログ情報を統一的に管理することで、進入経路や手段等の特定作業が向上する。

【再発防止策②】

特性	機能性	副特性	セキュリティ
代用特性（2次）	セキュリティ要件（実装検証含む）のユーザとベンダとの合意		

ISO/IEC15408)[CC（Common Criteria）/ST（Security Target）]認証レベルで合意し、第三者機関によるセキュリティ設計仕様書（ST）のST評価・ST確認を実施する。
a）セキュリティチェックを第三者が行うことにより、システムの脆弱性を客観的に評価することができ、攻撃対象と思える箇所に対し、事前にリスク軽減の対策を施すことが可能となる。

障害・再発防止事例 (16)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○	○	○		○

退職者によるNW不正アクセスにより社内システムのファイル数千件を削除・改ざん。
社内システムが十数時間停止し、顧客向け業務も中断するという事象が発生した。

障害による影響度合い	障害影響評価指標値	3	

顧客や個人情報の流出はなかった。

根本原因

以下観点での対処漏れが原因である。
①ID,パスワードの管理不徹底
②なりすましに関する警告機能の準備不足（不正アクセスを一定時間放置）
③データ損傷対策が不十分（可用性・正確性要求の高い情報に対するデータ保全策がなされていない）

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	セキュリティ
代用特性（2次）	不正アクセス、不正ログインに備えたアクセス制御対策		

（1）情報漏えいリスクを低減するため、統一的な認証基盤を構築。IDパスワード変更管理、誤入力とアカウントロック等の社内セキュリティポリシーを実装し運用する。
a）社外からのアクセスユーザはICセキュリティカードを発行。
b）カード申請、発行、更新、失効等の管理業務システムを構築し運用管理を実施。

（2）情報漏えいリスクを低減させるため、セキュリティプログラミングのガイドラインを活用。お客様から預かった業務資産（アプリ、データ）について、業務ごとにアクセスできる人間を制限し、開発環境とアカウントを分離することによって担保する。
a）本番環境（本番稼働サーバなど）にアクセスできる端末の利用は申請制。かつ監視室内設置の専用端末とする。アカウントも申請毎に変更する。

【再発防止策②】

特性	機能性	副特性	セキュリティ
代用特性（2次）	なりすましのモニタリング、警告機能に関する工夫		

情報漏えいリスクの低減、不正アクセスの防止を目的に、不正侵入検知／予防システム（IPS／IDS）を導入する。

【再発防止策③】

特性	機能性	副特性	セキュリティ
代用特性（2次）	データ損傷などのデータ保全対策		

システム構成上、可用性や機密性、完全性を考慮し物理配置上の工夫として下記を行う。
a）ファイルの二重化（隔離保管）
b）バックアップ（正・副）の取得
c）ストレージの冗長化
d）データのバックアップと保管場所分散
・ディスクの破損に対する復旧時間の短縮が図れる。
・局所的な自然災害の場合、早期の復旧が可能となる。

障害・再発防止事例 (17)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○	○	○

老朽化したホストシステムからオープン系システムに移植した際に、取引先還元情報に誤データを混入させてしまった。本番確認で発見したが、原因特定に時間を要し、取引先へのデータ提供が遅れた。移植時の改修によるプログラム欠陥だけでなく、潜在欠陥も関連した障害であった。

障害による影響度合い	障害影響評価指標値	4	

本障害では幸いにも取引先企業のみの影響であるが、銀行の立場ゆえ監督省庁からの厳しい指導あり。

根本原因

以下の原因が考えられる。
①潜在欠陥に対する対策（母体システムの品質向上施策）が取られていない。
②障害発生に備え、原因究明を容易にする対策（解析性向上施策）が取られていない。
③保守をし易くすることへの配慮（保守性向上）が不足している。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	成熟性
代用特性（2次）	潜在的障害予測（又は潜在的障害低減）および障害解決状況の分析		

（1）残存不良を推定し製品品質、信頼性を客観的に評価する。
a）サンプリングテスト：QA部門により検査項目の10%を目安に試験を行い、摘出された不良件数から残存不良を推定する手法
b）信頼度成長曲線モデルによる推定：テスト工程の80%時点を目安に、摘出された不良件数からモデルをあてはめ残存不良件数や信頼度、MTBFを推定する手法
c）ゴンベルツ曲線による推定：変曲点より総バグ数を求めることにより、バグ収束状況を判定する。

（2）システムテスト、運用等にて発生した障害については、事象、原因、対策および類似機能の有無を障害連絡報告に記載するとともに、再発防止策を立案する。類似機能が存在する場合、調査範囲を特定し、ユーザと合意を得た上で調査、場合により改修を実施する。

（3）障害を迷惑度別に切り分け、迷惑度の高い障害については、関連する各社の役員、本部長クラスが出席する会議で報告、検討することにより、障害原因、対策の周知を図る。
a）障害発生時に関連機能を調査することで、更なる障害の発生を未然に防止する。
b）障害の原因となった事項をチェックリストに追加し、類似する障害の発生を未然に防止する。

【再発防止策②】

特性	信頼性	副特性	障害許容性（発生防止策）
代用特性（2次）	障害を事前に検知するためのサブシステム間のトレース機能の工夫		

サブシステム間のシステム整合性チェック機能を準備し、不整合発生時は注意喚起する仕組みをとる。本機能は開発環境（システムテスト）にて他サブシステムも検証用として利用し、サブシステム間の不整合の早期発見に役立てる。

障害・再発防止事例 (17)（続き）

【再発防止策③】

特性	保守性	副特性	解析性
代用特性（2次）	ソフトウェアのトレース機能など解析容易性向上施策		

（1）アプリケーション実行ログフォーマットを取り決め出力ログの集計・解析ツールを作成する。
a）多量に出力されたログファイルのデータ集計作業を効率化し、エラーやボトルネック等の原因究明のためのトレーサビリティ（トランザクションの識別や問題箇所の特定）が向上する。
b）ツールを使用することにより解析精度の向上も図れる。
c）ソフトウェア資産の貸出/返却/更新状況を管理する。
いつ、誰が更新したのかの履歴を残し、その履歴を参照することで、ソフトウェア資産へのアクセス状況がトレース可能となる。

【再発防止策④】

特性	保守性	副特性	変更性
代用特性（2次）	保守ドキュメント、コメント率など変更容易性向上に関する施策		

プログラムソースへの修正履歴等のコメント挿入を標準化とする。
a）設計書を保守ドキュメントとしても使用可能となるよう考慮しながら、補足事項等記載内容を充実させる。

【再発防止策⑤】

特性	保守性	副特性	安定性
代用特性（2次）	母体品質向上施策		

保守開発でのテストフェーズにおいて、テストケースに過去本番障害事例を追加する。過去障害時に解析、検証で利用したデータも保管しておき、デグレード確認テストケースとして利用する。

障害・再発防止事例 (18)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○	○	○		

情報系システムにてデータベースアクセスが多発しシステムがダウン。イレギュラーデータの発生に伴い、データベースアクセス時のバッファ不足が発生。Webサイトが半日利用できなくなる。

障害による影響度合い	障害影響評価指標値	4	

システム停止時間中の影響取引は公表せず。監督省庁からの厳しい指導あり。

根本原因

以下の原因が考えられる。
①システムダウンを抑止する(＝MTBFを向上させる)対策がとれていない。
②システム資源の監視ができていない。
③イレギュラー処理実装方式の検討が不足している。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	成熟性
代用特性(2次)	MTBF向上施策		

サーバ障害に対し、冗長度を高めることにより信頼性を高める。下記対策(サーバ障害時の代替機用意)により、システム全体の信頼性を向上させる。
a)DBを持たないWeb／アプリケーションサーバなどはスケールアウト構成、DBサーバはクラスタ構成
b)待機系サーバ構成によるサーバ障害時の切替による継続運用
c)負荷分散装置の適用による複数サーバ構成の運用

【再発防止策②】

特性	信頼性	副特性	障害許容性(発生防止策)
代用特性(2次)	イレギュラー処理の実装に関する工夫		

データベースへの大量アクセスによるバッファ不足に対し、一定件数ごとにバッファを開放するようなDB検索方式を採用する。

【再発防止策③】

特性	信頼性	副特性	障害許容性(発生防止策)
代用特性(2次)	システム資源(CPU、ディスクなど)の使用状況監視、警告に関する工夫		

(1)一定間隔でのパフォーマンス監視とアラーム(警告)を実現する。
a)資源使用状況により、顕著な兆候が見られた際に当該システム開発部署に連絡を取り、状況を確認する。また、警告メッセージを使用することによりシステムの異常を早期に発見し、資源不足による障害の発生を未然に防止する。

(2)DBの使用容量を監視し、設定率(70%等)超過時にアラームメッセージを発生させる(プログラム内部テーブルについても同様)。
a)警告メッセージを使用することにより資源の不足を早期に発見し、資源不足による障害の発生を未然に防止する。

(3)特定業務のトランザクションの一時的な増加による影響を排除するため、あるデータベースの単位あたりの更新回数をモニタリングし、設定した上限閾値以上となった場合、該当するトランザクションを規制する仕組みを構築。一度規制がかかった場合は、更新回数が下限閾値以下になった場合自動的に規制を解除する(この規制の仕組みを手動でも発動可能)。
a)定期的な監視、評価により資源の不足を早期に発見し、資源不足による障害の発生を未然に防止する。

障害・再発防止事例 (18) （続き）

【再発防止策④】	特性	信頼性	副特性	障害許容性（拡大防止策）
	代用特性（2次）	ハードウェアのアラーム対応に関する工夫		

各種運用管理ソフト（SNMPによる機器監視ソフト）を利用してサーバ・端末に加え、ネットワーク機器についても監視を行う。
a）障害を検知した際にはパトランプを鳴らすということが効果的で、運用者が即時障害を知ることができる。

【再発防止策⑤】	特性	信頼性	副特性	回復性
	代用特性（2次）	バックアップ機への切替に関する施策		

サーバの冗長化設計をする場合には、縮退運転時の保証機能および性能劣化について、運用ルールを定めユーザと合意する。
a）縮退運転に関しユーザと事前に合意を行っておくことで、障害時には即時に実作業に着手でき、
システム使用停止時間の短縮に寄与する。

障害・再発防止事例 (19)

障害事例	業種	鉄道					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
					○		○

改修直後の座席予約システムで障害が発生。一旦システムを旧バージョンに戻しテスト環境下にて障害対応を実施。対応完了後、再度新システムに移行したが、直後に本来は空席の指定席が「発売済み」と判断され販売されない障害が発生した。

障害による影響度合い	障害影響評価指標値	5	

直接の影響は自社売上の減少であるが、一般利用者からのクレームが多発、信用失墜させてしまった。

根本原因

障害のリトライ作業をすべきところを、初回用の移行作業を行ってしまったため。(テスト環境で使用したデータの一部を誤って本番環境に反映してしまったというミスである。)

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	障害許容性(発生防止策)
代用特性(2次)	障害に対応するためのテスト環境準備に関する工夫		

運用開始に向け、移行計画(リハーサルを含む)について事前準備を行う。
a)計画範囲: データ移行、関連サーバ切替;N／W環境切替、端末切替、業務切替
b)計画内容:
【手順整備】移行手順(DB展開時間etc)・検証手順の取り決め
【役割整備】移行作業時の要員配置・役割分担の取り決めと作業環境の把握
【不測事態への対策】発生時の役割分担の明確化、情報連絡ルートの確立、戻し手順
【教育】移行作業における要員教育
【リハーサル実施】実施日程および回数

【再発防止策②】

特性	信頼性	副特性	回復性
代用特性(2次)	要件定義時点におけるユーザとベンダ間のコンティンジェンシープラン共有化		

コンティンジェンシープランとして、移行失敗時に旧システムに戻す時期や判断基準を設ける。
a)利用者数の多いシステムを対象とする。
b)事前取り決め範囲: 旧システムに戻すか対策前進するかを決断する有識者、
判断条件(対策前進時のタイムリミット、旧システムに戻す場合の切替時期やそれに伴う
付加費用に付随する問題)

障害・再発防止事例 (20)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○			○	○	○

インターネット取引サービスのシステムにて機能追加後の本番にて障害が発生、サーバダウンとなった。障害発生後の対応に手間取り、復旧まで1．5営業日を費やしてしまった。

障害による影響度合い	障害影響評価指標値	5	

長時間の取引停止となり、自社の信用失墜となる。影響件数（利用者数）等は公表されていない。

根本原因

障害発生防止の観点で、テスト密度不足（多様なデータパターンのテスト実施不足）があげられる。また、障害発生後の拡大防止の観点で、障害作業に対する準備不足があげられる。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	障害許容性（発生防止策）
代用特性（2次）	障害に対応するためのテスト環境準備に関する工夫		

特に重要インフラ等システムでは、本番環境でのテストケースを実施（テスト密度不足への対応）すべく、本番と同等性能の障害テスト確認環境（24時間利用可能）を構築し、障害対応時に利用する。
a）データは、個人情報に配慮したマスク済の本番データを利用する。
b）本番環境での確認用に、テスト用口座やテスト用ユーザIDを設ける。

※以下はホスト機での例。
本番使用JCLをテスト機用に自動変換する仕組みを実装し、かつ、個人情報をマスクした本番データの使用を可能とすることで、テスト環境で本番環境とほぼ同一のテスト（再確認）を実施可能にする。
a）抽出パターンを指定することによりテストに必要な最低限のデータのみ抜き出すことを可能とする。
b）本番JCLをテスト環境向けに自動変換するため、テスト環境の構築が容易に行え、
障害の解析およびテスト確認コストも削減できる。
c）本番JCL作成後にテスト環境向けにJCLを作成するため、本番JCLの確認が行え、
品質向上に繋がる。

【再発防止策②】

特性	信頼性	副特性	障害許容性（拡大防止策）
代用特性（2次）	停止防止対策		

障害発生時には、館内コールで関係者を集め、広く情報共有して影響の拡大を防いでいる。障害連絡第一報までの時間、暫定対応決定までの時間を定め、SLAとする。
a）障害状況を関係者が認識することにより、迅速な対応が実施できる。
b）障害発生時のSLAを定めることにより、障害発生時の連絡を迅速に行うことができる。

【再発防止策③】

特性	信頼性	副特性	障害許容性（拡大防止策）
代用特性（2次）	予防訓練に関する施策		

定期的に復旧訓練を実施する。シナリオを作成しバックアップ機で対象業務システムの復旧作業と情報伝達の訓練を実施する。
a）復旧処理迅速化のため作業手順や復旧コマンドの簡素化を図る。また、要員育成も兼ねて、悪意のある攻撃を想定した演習なども行う。

障害・再発防止事例 (20) （続き）

【再発防止策④】	特性	信頼性	副特性	回復性
	代用特性（2次）	自動リカバリ機能など故障耐久能力向上施策		

自動リカバリ機能として下記対応（工夫）を行う。
a）ディスクはRAID構成（ミラー構成あるいはホットスペア構成）として、障害発生時の復旧の自動化を行う。
b）ネットワークはルータ、スイッチ、経路等を二重化して障害時は迂回する。
c）処理単位にコミットポイントを設け、データを一部更新した状態で障害が発生した場合、所定の処理単位までリカバリを可能にする。
d）登録・更新系SQLをログ出力しておき、障害発生時は朝時点のDBから復旧させる。
e）アプリケーションプログラム起因による異常終了の場合、トランザクションは自動リカバリする（ミドルウェアの機能を活用するが、アプリケーションプログラム側での自動リカバリ可否を指定可能にする）。
f）オンライン稼働中にオンラインアプリケーションプログラムの入替を可能とする（ミドルウェアの機能を活用するが、アプリケーションプログラムの入替指示を可能にする）。

障害・再発防止事例 (21)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○		○

システム障害により、ATM、インターネット・バンキング、他行への振込みなど、同銀行のカードを使用したサービスが停止。復旧に1営業日を費やしたが、復旧作業に一部誤りがあり、翌営業日に再度システムがダウンする二次障害が発生した。初回障害原因はATM関連のパラメータの設定ミスであるが、二次障害は運用後障害の影響範囲特定ミスによる対応漏れ(トランザクション除去漏れによるデータ重複発生)が原因である。過去においても障害対応誤りによる二次障害を発生させた事実がある。

障害による影響度合い	障害影響評価指標値	5	

約3千件の取引に影響。振込みもあり、謝罪および賠償等も発生している。
また、安易な対応により二次障害が発生したことを受け、監督庁からの指摘を受ける。

根本原因

障害発生時の対応として、幾つか障害影響を拡大させた原因があると考える。
①必要な復旧作業が整理できていない(システム障害と業務との関連が整理できていない)。
②障害対応時の本番リリース保証活動が不十分である(二次障害を防止できていない)。
③過去失敗を教訓とした準備ができていない(類似障害発生に基づく横展開等が実施できていない)。
④本番環境とテスト環境との差異確認(環境設定漏れ防止)ができていない。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	回復性
代用特性(2次)	サービス提供再開施策(復旧予測時間の配慮など)		

発生した障害により、以降どのような重要処理・イベントが影響を受けるか、検索できる仕組みを構築する。
a)復旧作業のタイムリミット、作業優先度が的確に判断できる。

【再発防止策②】

特性	信頼性	副特性	障害許容性(発生防止策)
代用特性(2次)	運用時点での障害発生状況(傾向、内容分析)に基づく、障害の予防・是正処置の実施		

本番にて発生した障害、および保守案件対応時に発見した潜在障害については、事象と原因を特定するとともに、障害連絡報告を作成し、他の類似事象に関する調査範囲を特定するとともに再発防止策を作成する。
a)発生障害にランクをつけて、ランク毎の内容や傾向分析を実施。特に重大障害は特別なルールを定めて、障害分析～対策、水平展開を行い、再発を防止する。
b)再発防止策を是正処置・予防処置に展開する。

障害・再発防止事例 (21) （続き）

【再発防止策③】

特性	信頼性	副特性	障害許容性（発生防止策）
代用特性（2次）	障害管理（傾向・内容分析）に基づく、障害の予防・是正処置の実施		

（1）過去プロジェクトでの欠陥混入パターン分析に基づく「チェックリスト運用」「欠陥除去目標設定」などを実施する。
a）障害原因となった項目、コーディングをチェックリストに追加する（これにより、同様障害の発生を未然に防止することができる）。
b）発生した障害の事象および原因から、調査範囲・抽出する観点を導きだし、他の同事象もしくは同一原因に起因する不良がないか調査を行う。
c）障害の要因については、詳細分析し再発防止策を実施（開発用ドキュメントや各種チェックポイントへの反映も実施）する。

（2）重要なシステムについては、機能面、データ面およびバリエーションなどをキーに過去障害を分類し、必要に応じ強化テストを実施する。
a）品質強化を図る部分に着目し強化テストを実施することで、予定コスト内での品質向上につながる。

【再発防止策④】

特性	信頼性	副特性	障害許容性（拡大防止策）
代用特性（2次）	既存システム修正時における障害（リグレッションテスト漏れ、本番リリース不備による二次障害など）回避策		

既存システム改修時での故障抑制として、以下の手段を講じる。
a）システム資源の構成管理を充実する（開発中資産の誤った取り込みを抑止する）。
b）環境設定項目の列挙と、本番環境とテスト環境での差異を確認する（環境設定漏れ、設定不正を抑止する）。
c）デグレード確認の実施および確認テスト環境準備を必須とする。

【再発防止策⑤】

特性	保守性	副特性	試験性
代用特性（2次）	本番リリースを保証するためのテスト範囲特定方法などに関する取り組み		

（1）障害対応時を含め、本番（同等）環境でテストする項目を用意して、前述のテストツールにより予想結果と一致することを確認してからリリースする。
a）常に実施すべき確認項目を予め提供することで、一定品質を保つことができ、障害の発生を未然に防止する。

（2）本番環境とテスト環境の差異を明確化した上でテストを実施する。
a）本番環境とテスト環境との差異を明確にすることにより、本番環境でしか確認できない項目の扱いについて、ユーザと事前に調整することができ、対象項目に関わる箇所を重点的にレビューする等、障害除去に向けた早期の対応を行うことができる。

障害・再発防止事例 (22)

障害事例	業種	鉄道					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		○

インターネットからの指定席や乗車券が予約できる会員制サービスシステムにおいて、障害が発生。
ホスト機で稼働する、取り扱い履歴の並び替え処理を実施するプログラムに不具合。 このプログラムがメモリを過剰に占有し、本番系、待機系のホスト機がともにダウンし、周辺サーバと情報をやり取りできなくなった。待機系システムまでがダウンしたため復旧に手間取り、サービス提供に8時間程度の時間を要した。

障害による影響度合い	障害影響評価指標値	5	

インターネット利用による数万件の予約申し込み・変更が提供できず。終日エンドユーザは窓口業務(利用者対応)に追われる。

根本原因	

システム資源の監視がなされておらずメモリ占有に気づかなかったこと、本番リリース前での高負荷テスト実施が不十分であったことが原因である。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	信頼性	副特性	障害許容性(発生防止策)
	代用特性(2次)	システム資源(CPU、ディスクなど)の使用状況監視、警告に関する工夫		

本番稼働システムの異常を早期検知するためにシステム内に下記機能を盛り込む。
a)一定間隔でのパフォーマンス監視とアラーム発信機能。
b)ディスク(データベース領域)の使用容量が閾値を超過した場合に、統合監視システムへ自動で通知する機能。
c)ログファイルなど単純増加となるファイル使用率の閾値監視機能。
d)特定トランザクションの一時的な件数増加による影響を排除するため、特定トランザクションに関連するデータベースの更新回数をモニタリングし、閾値超過時に該当トランザクションを自動的に規制する機能。

【再発防止策②】	特性	保守性	副特性	試験性
	代用特性(2次)	本番リリースを保証するためのテスト範囲特定方法などに関する取り組み		

本番リリース前に、本番と同等のハードウェア環境下での負荷テストが実施できるよう、テスト用データベースやトランザクションデータを準備する。
a)高負荷テストでのパフォーマンス測定結果(目標達成度合い)をリリース条件とする。

障害・再発防止事例 (23)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○		

本番移行直後に、新規登録した顧客データの登録ミスが多発した。これにより誤った請求データを発送してしまう障害が発生。

障害による影響度合い	障害影響評価指標値	5	

発送した誤データは約2万件に及ぶ。

根本原因	

顧客データの登録画面操作が複雑で、誤登録を誘発しやすく、かつ誤りと判断しずらいものであった。また、納期必須の案件で本番移行期限までに間がなく、本番移行前でのユーザ試行期間は用意することができなかった。

品質特性に準拠した再発防止策	

【再発防止策①】

特性	使用性	副特性	習得性
代用特性（2次）	ユーザレベル（初級、中級、上級）などを配慮した習得容易性向上施策		

使用性（ユーザビリティ）の向上施策として、標準ルールを制定する。（以下はキーボード操作ルール）
a）高年齢の方向けにはキーボード操作のみとする。
b）キーボードは利用するキー以外を無効にする。
c）基幹業務系システムにおいては、最終的にキーボードによる連続入力の操作性が要求されることを考慮し、マウスによる操作以外にキーボードによる操作に対応する。
d）処理までのキータッチ数に対し、閾値を設定（多い場合はタッチ数を削減するよう変更）する。

【再発防止策②】

特性	信頼性	副特性	障害許容性（拡大防止策）
代用特性（2次）	稼働初期に起きる故障対策・分析に関する施策		

カットオーバー直後の本番での実運用形態を想定し、試験環境で移行～カットオーバー後数日間の本番リハーサルテストを実施する。
a）実際の運用形態に従った確認テストを行うことにより、環境による障害、インタフェースの誤認識による障害の発生を未然に防止する。
b）実際のタイムチャートを使用したテストを行うため、本番稼働時の実行時間の予測が行えることにより、想定以上の時間を要している処理ロジックを見直すことができるため、本番移行時のトラブルの削減が見込める。

【再発防止策③】

特性	使用性	副特性	理解性
代用特性（2次）	利用者側の教育効果向上施策		

利用者の教育のため、操作訓練等のトレーニングを実施する。
a）一日の流れにそって全てのノーマル業務を実行できる試験モードを構築する。
b）重要データの入力等、通常はなかなか実践できない業務に絞って入力練習のメニューを構築する。
c）本番環境に研修用DBを構築。本番機よりデータを抽出し小規模のデータベースを構築し、本番機同等の業務処理のトレーニングを可能とする。
d）「研修」をシステムのサービスメニューとしてエンドユーザに提供する（基本的な操作パターンを準備し、本番データにはアクセスせず、ダミーの帳票出力などを行う）。

障害・再発防止事例 (24)

障害事例	業種	保険					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
				○	○	○	

数百万件のお客様向け証明書の発送が遅延。原因はデータの抽出日判定誤り(プログラムミス)。

障害による影響度合い	障害影響評価指標値	5	

対応に時間がかかってしまったため、証明書の再送に数日間を要してしまった。

根本原因	

テストバリエーション(抽出日のバリエーション)不足が障害を摘出できなかった原因となっている。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	信頼性	副特性	障害許容性(拡大防止策)
	代用特性(2次)	稼働初期に起きる故障対策・分析に関する施策		

(1)一定期間、既存システム処理を残しておいて、本番処理の中で既存処理結果と新処理結果を比較し、差異があればアラームを発する仕組みを実装した。
a)本対応によって発見された障害は多数あり、早期対応に役立つ。

(2)既に運用に供しているシステムの機能追加・改修では、不具合発生時に直ちに直前のバージョンに戻せるツールを用意する(コンティンジェンシープランに則った対応を行う)。
a)不具合発生時早期な対応を行うことができるため、復旧時間の短縮と、障害拡大の未然防止に寄与する。

【再発防止策②】	特性	保守性	副特性	安定性
	代用特性(2次)	保守作業での欠陥混入是正に関する工夫		

システムの改良等で発生したテストケース(データ)を累積しておき、次回のリグレッションテスト実施時に追加する。
a)改修に伴う箇所だけでなく、少ない工数で、プログラム全体に関してテストを実施することができるため、改修箇所が及ぼす影響を早期に把握することができる。

障害・再発防止事例 (25)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		○

顧客情報照会システムで、レスポンスの遅延が発生し、大量の照会トランザクションが「処理待機状態」となる障害が発生した。(アクセス集中によるキャパシティオーバー)

障害による影響度合い	障害影響評価指標値	5	

利用者影響は数万(トランザクション件数は10万弱)と予測されている。

根本原因

あらかじめ予想していたピーク時想定値を過小評価していたことが原因で、「想定件数をオーバーすることはない」という判断のもと、オーバー時を考慮した仕組みも検討していなかった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	障害許容性(拡大防止策)
代用特性(2次)	取り扱い可能なデータ件数等システムのキャパシティオーバー時の誤動作防止に関する工夫		

(1)プロセス間通信用のキューの処理に対し、取り扱うデータサイズによって利用するキューの種類を動的に変更する仕組みを構築する(極力無駄の無いサイズのキューを使い、最適なサイズのキューに空きが無い場合は別のサイズのキューを使う)。
a)ひとつのキューが使いきられていた場合にも別のサイズのキューにて処理を継続することができ、障害を発生することなく高負荷状態を乗り切ることができる。これ以外として「入力規制」、「十分なキャッシュの確保」「ログのサイクリック」といった領域利用に関する工夫も行う。

(2)ホストから拠点サーバに情報を送信する際に、下記対応を行うことにより、取り扱い可能なデータ件数に配慮してシステム誤動作を防止する。
a)各アプリケーションプログラムで作成した送信データに対して、送信先拠点サーバ毎にデータを集約する。
b)拠点サーバ側に1回の受信可能データ量の制約があったため、分割送信する仕組みを構築する。送信タイミングを制御することのできるデータは、タイミングを遅らせる(翌日へ繰り延べる)仕組みも構築する。

【再発防止策②】

特性	効率性	副特性	資源効率性
代用特性(2次)	非平常時(ピーク時、障害および災害時など)に対応するレスポンスタイム、スループット、ターンアウンドタイムの考慮点などに関する施策		

(1)業務アプリ設計者自身が設計中の機能についてピーク時も考慮したオンライン応答時間／バッチ処理時間を容易に見積れるツールを提供する。提供ツールは、資源使用率等も把握するため資源のサイジング情報も同時にアウトプットする形式とする。
a)設計の早い段階で性能要求を満たせるか否かを把握できるため、顧客との共通認識のもと手戻りリスクも減らすことができる。
b)負荷分散構造システムによる異常時の考慮を事前に行うことにより、性能低下を未然に予測することができる。
(2)システムテストにおいて必ず高負荷テストを実施し、ピーク時における性能評価を行う。

障害・再発防止事例 (26)

障害事例	業種	輸送（空運）					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		

運航業務システムにてトラブルが発生。拠点サーバに大量のデータが送信された際、一部のデータロストに起因して拠点サーバが停止する障害が発生した。

障害による影響度合い	障害影響評価指標値	5	

運航遅延による影響者数は1万人程度。影響便は数十便。

根本原因	

データベースの容量不足(設計考慮不足)が原因であるが、想定外のデータ量が発生する場合の考慮も不十分であった。

品質特性に準拠した再発防止策	

【再発防止策①】

特性	信頼性	副特性	障害許容性（拡大防止策）
代用特性（2次）	取り扱い可能なデータ件数等システムのキャパシティオーバー時の誤動作防止に関する工夫		

（1）ホストから拠点サーバに情報を送信する際に、下記対応を行うことにより、取り扱い可能なデータ件数に配慮してシステム誤動作を防止する。
a）各アプリケーションプログラムで作成した送信データに対して、送信先拠点サーバごとにデータを集約する。
b）拠点サーバ側に1回の受信可能データ量の制約があったため、分割送信する仕組みを構築する。
送信タイミングを制御することのできるデータは、タイミングを遅らせる（翌日へ繰り延べる）仕組みも構築する。

【再発防止策②】

特性	効率性	副特性	資源効率性
代用特性（2次）	非平常時（ピーク時、障害および災害時など）に対応する資源（CPU、メモリー、伝送、入出力、設置条件（スペース、電源容量など））、体制などの考慮点に関する施策		

仮想化技術やパーティショニング技術によって、負荷量が大きく変動した時に動的にCPUリソース等を増強・配分変更できる仕組み、および伝送データを配分する仕組みを構築する。
a）高性能のシステム構成を最小限のコストにて実現できる。
b）データベースを可変長レコード定義し、DBI（データベースインタフェースソフト）で圧縮展開機能を組み込んで格納量を削減したことにより、ディスク使用量が減少するとともに、性能も向上する。

障害・再発防止事例 (27)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○			

株式の取引成立を知らせる通知作成に不具合があり、一部の証券会社に取引完了情報が送信されない障害が起こり、一部の株式取引を終日停止させる事態が起こった。プログラム改修時のバグが原因である。

障害による影響度合い	障害影響評価指標値	5	

取引停止となった会社および投資家への謝罪に限らず、接続先社にも影響を与える。自社の信用も失墜。

根本原因	

システムの異常を早期に検知する仕組みができていなかった。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	信頼性	副特性	障害許容性（拡大防止策）
	代用特性（2次）	異常を検知し他システムなどへの影響を遮断する機能に関する工夫		

（1）プログラム間やプログラム－データベース間などのインタフェースとしてやり取りするデータについて、異常検知時にアラームを発行し、誤データ送信を防止する仕組みを組み込む。これにより、ダーティデータ拡散を防止する。

（2）特定処理（他サブシステム連携等）での異常終了やデータ不整合を検知し、自動的に後続処理を規制する仕組みを構築し、影響を最小限に留めることができる。

障害・再発防止事例 (28)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○			○	○

保守案件実施後のシステムにおいて夜間バッチ処理が異常終了となった。システム復旧が遅れ翌日午前6時のサービス開始までに対応が間に合わず、再開が翌日午後となってしまった。障害原因はプログラム欠陥による。

障害による影響度合い	障害影響評価指標値	5	

取引利用者数は公表されていないが、翌日午前中の利用は一切できず。数千億円の取引に影響があるとされている。

根本原因	

プログラム欠陥を作りこんだ原因は設計書誤りであった。当該案件では特定機能のみユーザレビュー対象としており、誤りの設計書はユーザレビュー対象外となっていた。また障害影響を拡大させた原因としては、障害影響が他システムのデータベースまで広がっていたこと、回復オペレーション処理が煩雑であったことがと考えられる。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	信頼性	副特性	障害許容性（拡大防止策）
	代用特性（2次）	異常を検知し他システムなどへの影響を遮断する機能に関する工夫		

（1）ネットワーク、サーバおよび処理プロセスなどのシステムの構成要素単位に分けて、単位ごとに障害発生、影響範囲をシュミレートする。機能単位の工夫により、回復方法も手順化できる。
a）障害を局所化することにより、障害の拡大を防止し、影響を最小限に留めることができる。

（2）プログラム間、プログラム−画面間、プログラム−データベース間等のインタフェースとしてやり取りされるデータについて、含まれる項目ごとの型チェック機能等を装備し、誤ったデータが伝搬することを防ぐ。
a）無チェック（ダーティ）データが他システムに拡散することを防止できる。

【再発防止策②】	特性	使用性	副特性	運用性
	代用特性（2次）	システム運用の容易性向上、誤操作防止施策		

平常時および障害時の運用を記載した「運用設計チェックシート」を開発段階で作成し、運用部門のレビューを受ける。
a）標準外の運用を極力削減し、運用部門と開発段階で調整を行うことで運用トラブルの削減を図る。

【再発防止策③】	特性	使用性	副特性	運用性
	代用特性（2次）	オペレータの介入操作に関する工夫		

運用部門担当者が操作を実施またはシミュレーションし、操作のし易さ、妥当性を検証する。

【再発防止策④】	特性	機能性	副特性	合目的性
	代用特性（2次）	妥当性（利用者側の要件）の確認（レビュー、テストに対するユーザとベンダとの役割分担		

ベンダが要件定義書をもとに設計書が作成する場合、ユーザが設計書レビューに参加し要件が漏れなく展開されていることを確認する。業務部門のユーザが参加することで、要件のトレーサビリティを保証する。

障害・再発防止事例 (29)

障害事例	業種	金融					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○	○	○

他金融機関とのデータ通信システムにてデータが滞留し始め、同日決済予定であった100万件弱のトランザクションデータが滞留する障害が発生した。原因はプログラム欠陥である。障害特定およびシステム再起動に手間取り復旧に1営業日を費やした。

障害による影響度合い	障害影響評価指標値	5	

トランザクション量の多さ、回復まで時間を要したことから、監督省庁からの指導、改善要求をうけることとなった。

根本原因

システムの基盤(監視)部分の設計漏れが原因だが、更に復旧に手間取った原因としては、システム再起動時のオペレーションが煩雑であったこと、障害内容を解析する仕組み(工夫)がなされていなかったことが考えられる。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	回復性
代用特性(2次)	可用性(システムの稼働率、サービス提供時間、運転時間等の割合)向上施策		

(1)過去障害(DBミドルウェアプロセスの異常終了時に原因が判明せず)を教訓に、ミドルウェアを監視する仕組みを構築し、当該プロセスが停止していた場合には直ちに再起動する。
a)異常終了が発生しても自動的に30秒程度の間に復旧することが可能となる。

(2)アプリケーションの基盤機能として、初期起動か再起動かを判断できる仕組みを構築する。再起動の場合、バックアップファイルから共有メモリ上のデータを復元してからアプリケーションを起動する仕組みを持ち、システム再起動後、オペレーション無しで業務を再開できる仕組みとする。同様にアプリケーションの基盤機能として、アプリケーションの動作状態を監視する仕組みを構築する。例えば、アプリケーションが特定のファイルを一定時間ロックし続けた場合、監視機能が当該アプリケーションを強制終了し、且つ、その後自動的に再起動する等の仕組みとする。
a)再起動に要する時間の削減となり、障害復旧時間が削減できる。

(3)DB更新処理では一般に複数の利用者間(プロセス間)でデッドロック状態が発生する可能性があるが、アプリ基盤で自動的にトランザクションのロールバックとリトライの処理を実行する機能を提供する。
a)各アプリに負担をかけずに、稼働率向上を図ることができる。

【再発防止策②】

特性	保守性	副特性	解析性
代用特性(2次)	ソフトウェアのトレース機能など解析容易性向上施策		

(1)アプリケーション基盤機能にて、プロセス間のキュー通信情報を全てログに残す仕組みをとる。(基盤機能が自動的にログを残すようにしており、送信元プロセス名、宛先プロセス名等も記録する。アプリケーションの実行ログはプログラミング後に自動的にログ出力を挿入する仕組みにしており、入力トランザクション毎に一意の番号づけ、発生時刻を刻印して、トランザクションがシステムから消滅するまでデータ中に保持させる。)
a)的確なログを採取することにより、障害原因が特定できるため、解析時間の短縮が図れる。

(2)アプリケーション実行ログフォーマットを取り決め出力ログの集計・解析ツールを作成する。
a)多量に出力されたログファイルのデータ集計作業を効率化し、エラーやボトルネック等の原因究明のためのトレーサビリティ(トランザクションの識別や問題箇所の特定)が向上する。
b)ツールを使用することにより解析精度の向上も図れる。

障害・再発防止事例 (30)

障害事例	業種	気象					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
		○	○		○		○

気象データを報道機関などに配信する「電文形式データ配信システム」が17時間にわたって停止し、地震・津波の注意報・警報・予報などのデータ配信が不可能になった。

障害による影響度合い	障害影響評価指標値	4	

万が一、停止中に大規模な災害があったことを想定すると、人命を含め大きな影響が出たものと推測。

根本原因

①UNIXサーバのハード異常、②引継ぎ情報を格納したファイルの異常、③回線切り替え機の故障が原因となっているが、可用性を向上させる施策がとれていない。システム構成の問題で障害影響が広範囲となり、復旧時間を遅らせた。

品質特性に準拠した再発防止策

【再発防止策①】

特性	信頼性	副特性	回復性
代用特性（2次）	可用性（システムの稼働率、サービス提供時間、運転時間等の割合）向上施策		

アプリケーションの基盤機能として、初期起動か再起動かを判断できる仕組みを構築する。
a）再起動の場合、バックアップファイルから共有メモリ上のデータを復元してからアプリケーションを起動する仕組みを持ち、システム再起動後、オペレーション無しで業務を再開することができる。
b）同様にアプリケーションの基盤機能として、アプリケーションの動作状態を監視する仕組みを構築する。例えば、アプリケーションが特定のファイルを一定時間ロックし続けた場合、監視機能が当該アプリケーションを強制終了し、且つ、その後、自動的に再起動する等の仕組みとする。再起動に要する時間の削減となり、障害復旧時間が短縮する。

【再発防止策②】

特性	信頼性	副特性	回復性
代用特性（2次）	広域・局所災害対策		

※以下は、天災等の災害発生時における効果事例

災害発生時に影響を局所化すべく、以下の工夫を行う。
a）大規模データベースを拠点別に区画分割構成することにより、障害の全体への影響波及および相互干渉を防止する。
b）現場業務継続続行に必要な必要最低限のデータを事業所に退避保管。定期的に更新し災害時の情報検索業務が可能な仕組みを整備する。

【再発防止策③】

特性	信頼性	副特性	成熟性
代用特性（2次）	MTBF向上施策		

（1）ネットワーク障害に対し、冗長度を高めることにより通信関係の信頼性を高める。
a）異経路の複数回線（異なる通信会社を使用した回線の2重化）を使用する。
さらにバックアップとして携帯電話、FAX、近隣事業所へのデータ持込を可能とする。

（2）サーバ障害に対し、冗長度を高めることにより信頼性を高める。下記対策（サーバ障害時の代替機用意）により、システム全体の信頼性を向上させる。
a）DBを持たないWeb／アプリケーションサーバなどはスケールアウト構成、DBサーバはクラスタ構成とする。
b）待機系サーバ構成によるサーバ障害時の切替による継続運用を実現する。
c）負荷分散装置の適用による複数サーバ構成の運用とする。

障害・再発防止事例 (31)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
						○	○

夜間取引システムで故障ハードディスクがバックアップに切り替わらない障害が発生し、株式売買取引、個人向け国債の予約注文、取引コース変更、ATMの入出金が不可能になった。ハードウェア故障は即時に検知できたが、バックアップ機に切り替えた際に一部ソフトウェアのバージョンが古く、立ち上げ直後にシステム停止となり、復旧に半日ほど時間を要してしまった。

障害による影響度合い	障害影響評価指標値	4	

取引停止による投資家影響は公表されず。信用失墜は否めない。

根本原因	

バックアップ機への切替に際し、留意すべき事項が整理されていないことが原因である。通常の保守運用業務において管理徹底ができていれば立上後再停止は発生しなかったはずである。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	信頼性	副特性	回復性
	代用特性（2次）	バックアップ機への切替に関する施策		

バックアップ機について「定義ファイルの変更（テスト機として利用時の変更）」や「プログラムバージョンの更新漏れ」などにより、切替時にシステムが立ち上がらないといった弊害を招くことがあるため、通常保守および運用において、定義ファイル確認やプログラムバージョン管理を徹底する（プログラムライブラリはディスクミラーリングで同期をとる）。

障害・再発防止事例 (32)

障害事例	業種	通信					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
						○	○

各営業店に配置されているシステムから収集する取引情報に誤りがあり、翌朝に統合サーバがダウンしてしまい、利用者へのサービス提供が半日不可能となる障害が発生した。本番稼働後に判明した潜在欠陥対応の際、一部営業店のシステム切替を手作業で行った際にインストール方法を誤ってしまい、既存データも一部破損させてしまっていた。

障害による影響度合い	障害影響評価指標値	4	

システム停止時間内での影響量は未公表である。

根本原因	

システム切替について、人的ミスによるリスクを過小評価していたことが原因である。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	使用性	副特性	運用性
	代用特性（2次）	インストールやバージョンアップの容易性（バージョンアップ後の確認の容易性、配布容易性を含む）向上施策		

ソフトウェアの自動配信機能を構築し、人手によるインストール時のミスを防止する。

拠点サーバのプログラム入れ替えは、通常は自動配信機能を使用し、大量のプログラム入れ替えではCD－R等のメディアを現場へ郵送して行う。どちらの手段でも、新機能リリースの1週間前から更新を開始し、済/未了の監視をメインフレーム側で監視できる仕組みを構築し、2日前には未了の現場へ督促する態勢にある。
なお、サーバのアプリプログラムは、リリース日により処理ロジックが制御される仕組みであり、リリース日の前に新機能が動くことは無い（テストで事前に確認済み）。

障害・再発防止事例 (33)

障害事例	業種	証券					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○			○

株価指数先物の取引にて複数銘柄の取引を1営業日午後いっぱい停止させる障害が発生した。DBに不整合が発生し、約定処理が停止した。原因はプログラムバグであった。また、原因特定に時間がかかり、システム復旧が遅れてしまった。

障害による影響度合い	障害影響評価指標値	4

取引停止による影響件数等は未公表。自社の信用失墜はまぬがれない。

根本原因

プログラム欠陥を作りこんだ根本原因はシステムの基盤（監視）部分の設計誤り（要件誤読）である。本来ユーザのIT部門担当がレビューすべきところを、業務部門担当のユーザがレビューを実施し、レビュー完了としていた。また、アプリケーションログによる解析を試みたが、ログ内容から有効な情報が得られなかったことが、復旧時間を遅らせる原因となっている。

品質特性に準拠した再発防止策

【再発防止策①】

特性	機能性	副特性	合目的性
代用特性（2次）	妥当性（利用者側の要件）の確認（レビュー、テストに対するユーザとベンダとの役割分担		

生産物レビューについて、ベンダとユーザ（IT部門、業務部門、運用部門）のレビュー担当範囲を定義する。また、各担当がレビュー結果に対する責任を宣言する意味で、レビュー報告書への検収捺印を行う。

【再発防止策②】

特性	保守性	副特性	解析性
代用特性（2次）	データログ実装、状況監視データ取得など活動記録保有能力に関する施策		

（1）以下の観点でログ方式の型決め（標準化）を行う。
a）ログデータの目的の整理と分類の方法
b）各アプリ実装時のログデータ取得方法
c）システム運用時のログデータ参照方法
d）業務アプリケーション実行時のログ収集（アプリログ）

ログ収集の標準化を図ることにより、システム開発時に新たな仕組みを検討する必要がなく、開発効率の向上が図れる。ログ内容については過去発生インシデントの分析より、効果的な項目、吐き出しタイミングを評価する。

【再発防止策③】

特性	信頼性	副特性	回復性
代用特性（2次）	サービス提供再開施策（復旧予測時間の配慮など）		

障害発生時の対応については、業務回復優先とした体制、手順を準備する。障害解析チームと業務回復チームを分けて対応することにより、重複した作業を行わず効率的な回復を行う。

障害・再発防止事例 (34)

障害事例	業種	省庁					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
						○	○

プログラム欠陥により、地震規模計算の際に別の地震データを取り込んでしまい、結果誤った地震発生情報を鉄道、電力、ガス等社会インフラシステムを保有する企業へと連携してしまった。このため社会インフラシステムが一部停止する（交通機関の運転見合わせ等）といった事象を引き起こした。

障害による影響度合い	障害影響評価指標値	4	

情報連携後、社会インフラは直ちに復旧し大きな混乱はおこらなかった。ただし、今回は小規模の天災を「大規模」と誤判断したが、逆の場合は人命につながる障害になる可能性がある。

根本原因	

本来本番リリースするべきでない、改修途中のプログラムを本番移行してしまったため。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	保守性	副特性	安定性
	代用特性（2次）	バージョンアップによるデグレード防止に関する施策		

ソフトウェアのバージョンアップ（本番移行）について、厳格な運用規準を設ける。
a）本番障害を除き、事前の移行申請－承認を経て試行環境からのみ移行可能とする。
試行環境での稼働実績（品質「デグレード防止」に関する結果承認）がない限り、本番移行することができない仕組みとする。

【再発防止策②】	特性	保守性	副特性	変更性
	代用特性（2次）	変更履歴、構成管理などの変更制御に関する工夫		

機能検証（結合テストなど）以降のシステム資産について、構成管理用パッケージ製品等を用いて資産を一元管理するとともに、管理運用ルールを徹底する。また、改修案件については、システム資産（プログラムや設計書等）に案件と資産の貸出・変更履歴、リリース履歴など蓄積することで、資産の変更制御を容易にする。

障害・再発防止事例 (35)

障害事例	業種	通信					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
				○	○	○	

請求額を計算するプログラムの欠陥が原因で、一部の契約者から受信料を余分に徴収してしまう障害が発生した。数年前に組み込んでいた残存欠陥（特定ケースのみ発生するため顕在化せず）が原因で、特殊なトランザクションデータが発生した際にエリア初期化タイミングが悪く後続データ全件に影響してしまう障害であった。

障害による影響度合い	障害影響評価指標値	4	

余分に徴収した金額は数百万円であるが、類似障害を過去にも発生させており、監督省庁からの指導が発生。

根本原因

過去のシステム改修タイミングや同様の本番障害発生時に対象システムの点検を行っていれば、未然にプログラム欠陥を検知できたものと考えられる。システムが古くスパゲティ化していることからも、「プログラム欠陥を作りこみやすいシステム」ととらえ、あらかじめ対策を講じるべきであった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	保守性	副特性	安定性
代用特性（2次）	障害混入確率の低減に関する施策		

過去の障害混入より、再発防止観点をリストアップしたチェックリストを励行する。
a）改良開発時において、過去事例と類似する障害の発生を抑止することができる。
b）類似テストケースを実施することで、母体品質向上（残存欠陥摘出）が図れる。

【再発防止策②】

特性	保守性	副特性	変更性
代用特性（2次）	母体システムの構造化度、変更生産性など変更容易性向上に関する施策		

アプリ基盤のアーキテクチャとして、システム（資産全般）を階層的に構造化し（サービス、データモデル、データソース）、かつ階層間のインタフェースも標準化する。
a）システムの経年劣化を防止し、変更に強い資産とすることに寄与する。

【再発防止策③】

特性	信頼性	副特性	成熟性
代用特性（2次）	「テストコスト」と「本番稼働後の欠陥による社会的影響や復旧にかかるコスト」とのトレードオフを考慮したテスト手法（テスト網羅率）の取り込み		

本番稼働後のリスクを勘案した試験項目を抽出し、テストを実施する。リスクに応じテスト実施要否を判定する。

障害・再発防止事例 (36)

障害事例	業種	保険					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○	○	○		

一般加入者向けWebシステムがDB不整合によりダウンする障害が発生。ホスト処理の日中バッチ処理で不正データが混入してしまい、DB間の不整合が発生、結果情報系システムへの連携データが重複してしまった。

障害による影響度合い	障害影響評価指標値	5	

一般加入者で利用不可能となった件数は公表されず。ほぼ1営業日が利用不可能となり、監督省庁からの指導が発生。

根本原因	

システムに保有するデータの整合性が崩れた際の警告発信や不整合データの遮断など、システム内のデータを安定させる工夫がなされていなかったことが原因である。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	保守性	副特性	安定性
	代用特性（2次）	システム保有データの整合性維持に関する施策		

DBの相関関係やデータの整合性を検証するツールを開発し、通常運用や障害対応のなかで走行させる。システム内やシステム間でのDB整合性をチェックし、チェック結果を出力する。

a）当該ツールを実行することで、表面化しにくいDB間の不整合を早期に摘出することが把握できる。

b）障害の早期解決（原因の早期特定）に寄与する。

c）テスト工程に適用することで、テスト品質向上（摘出漏れ防止）および生産性向上も期待できる。

障害・再発防止事例 (37)

障害事例	業種	行政					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
			○		○		

全国の事務所にて一般市民の年金記録の照会ができなくなった。

障害による影響度合い	障害影響評価指標値	4	

全国の都道府県合計で、数百の事務所にて影響発生。利用者数は公表されていない。

根本原因

オンライン・システムを制御するミドルウェアのバージョンアップにあたり、業務アプリケーションへの改修が必要であったが、非互換項目の存在を発見できず、業務アプリケーションの改修は行っていなかった。

品質特性に準拠した再発防止策

【再発防止策①】

特性	保守性	副特性	安定性
代用特性（2次）	バージョンアップによるデグレード防止に関する施策		

OSやミドルウエア等のアップグレードにおいて、明示的な非互換項目が無い場合でも、稼働確認テストに加え開発環境で十分な期間（2～3ヶ月程度）稼働させスタビリティを確認してから本番機へ適用する。
a）試行環境にて稼働することで、動作結果を事前に把握でき、障害の発生を未然に防止する。

【再発防止策②】

特性	移植性	副特性	環境適応性
代用特性（2次）	ソフトウェア（アプリケーション）の環境適応性向上施策（OS、ミドルウェア、DBMSなど）		

プラットフォームに依存しない開発言語を推進するとともに、アプリ基盤としてEJBコンテナやDBマネジメントシステムなどに依存しない構造化を工夫する。
a）OSやミドルウェア等のアップグレードに対し、業務アプリケーションへの影響を出来る限り抑制することができる。

【再発防止策③】

特性	信頼性	副特性	障害許容性（発生防止策）
代用特性（2次）	障害に対応するためのテスト環境準備に関する工夫		

業務の準本番環境（保守環境）を構築し、最終検証をこの環境で実施する。本番同等の環境でテストを実施することができ、機能考慮漏れを摘出することができる。

障害・再発防止事例 (38)

障害事例	業種	銀行					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
					○	○	

利子計算を行うプログラムに欠陥があり、顧客に送付した取引残高報告書に誤情報を記載してしまう障害が発生した。

障害による影響度合い	障害影響評価指標値	5	

送付済みの報告書約1万件。顧客へのお詫びを実施。開発プロセス自身に問題があることから、組織横断で見直しを実施。行員30人体制で、ルール策定から開始し、3ヶ月を要し整備し改定している。

根本原因	

利子計算自体が複雑なため、計算結果検証用のツールを別途用意していたが、保守案件対応時、当該ツールに用いる一部条件式（パラメータ）の変更が必要なところを漏らしたままシステム改修を実施していたため（検証ツールの修正は複雑なため、ユーザ側IT担当者の作業となっていた）。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	保守性	副特性	試験性
	代用特性（2次）	保守テストでのテストツール（自動再帰テストツールなど）、本番環境具備、標準テストセットの具備などのテスト環境整備に関する工夫		

テスト工程に備え、「DB生成ツール」や「同整合性確認ツール」「計算結果検証ツール」を作成し、保守案件にて変更が発生する場合はツールもメンテナンスを必須作業とする。

a）本ツールにより、人手によるデータ準備や検証工数が大幅に削減され、かつ保守のテストにおいてデータ整備不備による手戻りの発生を防止できる。
テスト検証ではツール結果に依存するところが大きいため、ツールの整備（使用データも含む）には必要十分な工数を投入する。

障害・再発防止事例 (39)

障害事例	業種	保険					
	障害発生元	開発				保守	運用
		要件定義	設計	実装	テスト		
						○	

保守案件対応時に作りこんでしまったプログラム欠陥が原因で、一部契約の還付金の支払い不足が発生。

障害による影響度合い	障害影響評価指標値	5	

数千人の加入者に還付金不足が発生し、お詫びおよび還付金の追加支払いを実施した。また、他保険商品に関しても点検を実施したことにより、システム部門および業務部門（ユーザ）が数週間点検作業に追われるという事態を招く。

根本原因	

実際の対応は、担当者の休職により代理のシステム担当者が行っていた。代理担当者は対象システムの状況、特徴などを把握できておらず、影響範囲の特定を誤っていたため、レビューやテストにて事前検知することができなかった。もともと保守案件の対応は、すべて専任のシステム担当者に依存しており、担当者毎に保守に向けた準備、取り組み方法がまちまちとなっていた。組織的な保守作業の取り組み（標準化、規定類の整備や運用徹底を含む）ができていないことが根本原因である。

品質特性に準拠した再発防止策				
【再発防止策①】	特性	保守性	副特性	保守性標準適合性
	代用特性（2次）	保守性に対応する規定（業務、内部統制、ISMS、国際会計など）および開発標準の適合に関する監査対策		

システム毎に当該システムの概要を記載したプロファイルを作成し、システム構成や関連システム、非機能要求レベルおよび障害時の影響レベルなどを規定することを標準化する。当該プロファイルは保守ドキュメントの位置づけであるが、リスク分析の元ネタとして活用し、保守コスト評価にも利用する。

a）一元化されたプロファイルを参照することにより、システムの現況が容易に理解でき、保守工数の低減にも寄与する。

＜　空欄　＞

第5章　トレーサビリティ管理の手法

5.1 トレーサビリティの重要性

ソフトウェア品質保証を行う上で必ず話題にあがるのが、ソフトウェアトレーサビリティです。

ソフトウェアの品質を保証する上で、まずソフトウェアの提供者側が考えなければならないことは、ビジネス・業務プロセスにおいて、魅力的で有益なソフトウェアを提供することです。魅力的で有益なソフトウェアとは、機能的で信頼性が高く、効率的である必要があります。ソフトウェアの提供者は、機能性を追求するだけではなく、高性能で安全性（健全性）が高い等、高品質を保証した製品の開発を目指すべきです。信頼性の高いソフトウェアを作るためには、十分なテストを行い、そのソフトウェアの動作状況をしっかりと確認しなければなりません。しかしながらテストでは、ソフトウェアが魅力的で有益であることを確認するのですから、当然テストに先立って魅力的で有益であることを、ソフトウェアへの要求として的確に定義し、これを製品の中に確実に作りこむことが、より重要になります。つまりテストに至る前に、欠陥（ここでは定義された、あるいは暗黙のソフトウェアへの要求からの逸脱という意味で用いています）を作りこむことを予防することがより重要であり、これにソフトウェアのトレーサビリティが深くかかわります。

以上からソフトウェアの品質保証は、次の要素を保証するものでなければなりません。

(1) 有益であり安全性が高いこと
- (1-1) 要求仕様が現行業務に対して適切であること
- (1-2) システム分析に使用した基礎データが適切であること
- (1-3) システムは想定外の状況に対応できること

(2) システム設計は要求仕様を満足していること
- (2-1) システム設計がソフトウェア要件に合致していること
- (2-2) システム設計が想定外の状況に対応できること

(3) 実装がシステム設計に合致していること
- (3-1) 実装が詳細設計に合致していること
- (3-2) 実装が想定外の状況に対応できること

(4) 要求仕様がシステム試験で確認されること
- (4-1) システム試験工程が要求仕様に合致していること
- (4-2) すべての試験工程が再現可能であること

「(1) 有益であり、安全性が高いこと」は、顧客の要求と要求仕様とのトレーサビリティ、「(2) システム設計は要求仕様を満足していること」は、要求仕様とシステム設計とのトレーサビリティ、「(3) 実装がシステム設計に合致していること」は、システム設計と実装とのトレーサビリティ、「(4) 要求仕様がシステム試験で確認されること」は、要求仕様とシステム試験とのトレーサビリティにそれぞれ対応しており、ソフトウェアの品質保証の根幹を成すものであることが分かります。

5.2 トレーサビリティとは

一般的には、ソフトウェアのトレーサビリティとして、以下の 2 つが挙げられます。

(1) 製品に関するトレーサビリティ
(2) 文書およびデータに関するトレーサビリティ

(1) は、ソフトウェアアイテム（ソフトウェアの部品の最小構成要素）が、製品にどのように組み込まれているかが追跡できることであり、ハードウェア製品などの部品が、どのように製品に組み込まれているかを管理できることと同じです。(2) は、ソフトウェア要件が、製品のどの部分で実現されているかを追跡できることを意味し、逆にできあがっているソフトウェアの構成部品（例えば、ある実行モジュール）が、どのソフトウェア要件を実現するためのものかを追跡できることも意味します。ISO9001 で要求されているトレーサビリティは (1) だけですが、TickIT*では(2) のトレーサビリティも要求しています。本章では、「(2) 文書およびデータに関するトレーサビリティ」を扱います。

ソフトウェア開発の途中で仕様の一部に変更があった場合、変更に関連している他の部分も正しく修正しなければなりません。逆にソースコードの中に設計上の問題があった場合は、その基になっている設計まで立ち返って仕様を修正することとなり、変更の影響は次々に連鎖します。

このようにソフトウェア開発では、個々の成果物をあいまいなく記述するだけでなく、それらの成果物が互いにどのように影響し合っているかを把握することが重要であり、そのような特性がトレーサビリティ（traceability、追跡可能性）です。トレーサビリティを実現することによって、ソフトウェアの一部分を修正したときに、その影響を受けるソフトウェアの構成部品および関連する箇所が特定可能となるため、必要なソフトウェアの修正漏れがなくなり、矛盾を防ぐことができます。

トレーサビリティを実現するには、ソフトウェアの様々な構成品に対して、

- 自分の変更が影響を与える構成品（自分→他）
- その変更が自分に影響を与える構成品（他→自分）

というソフトウェア構成品間の関係（以下「トレース情報」と呼ぶ）を記録しておく必要があり、各企業では様々な構成管理ツールを利用して、ソフトウェア構成品間の関係を管理しておられます。

ここで注意しなければならないことは、ソフトウェア構成品間の矛盾を防いでも、「ソフトウェアが有益で安全性が高い」ということを、具体的に定めたソフトウェア要件との矛盾を防ぐことにはならないということです。

要件をトップダウンに展開してソフトウェアに実装していく過程は、ソフトウェアの企画、設計および開発行為そのもので、要件はこの過程で変わり得ます。従って、要件とソフトウェア構成品とのトレーサビリティを管理するプロセスを確立して、トレース情報を記録していくことが望まれます。

要件とソフトウェア構成品とのトレース情報を記録する方法は、いくつか考えられます。例えばソフトウェア設計検証の目的は、要求されている事項すべてが適切に実現されていること（達成されるであろうこと）を担保することですから、トレース情報を設計検証のインプットとし、ソフトウェア設計検証によってこれを確認して更新する方法も考えられます。

本章では、主にソフトウェア要件とソフトウェア構成品とのトレーサビリティの管理に焦点をあてます。

---

* TickIT は、英国コンピュータ学会（BCS）を中心に英国の IT 業界が協力して構築した、ソフトウェアの開発・保守、情報サービス事業者を対象とした第三者認証制度です。ISO9000 シリーズをベースにしながら、独自の組織・規則・手順を持ち、手引書と付属書を発行しています。

5.3　品質展開の概要

本節では、3.1 節で紹介した「品質展開」を参照して、要件とソフトウェア構成品とのトレーサビリティの管理において、考慮すべき事項を整理します。品質展開は、製品品質の要素をできるだけ細分化し品質要素の相互関連を整理して、品質要素を製品の目的に適合するように集合として管理する方法です。

実際には、品質展開は膨大になりがちで、顧客の関心度合いや当該システム化対象業務の他社との競争力（実現水準との比較による）などにより、戦略的に重点項目を選定して利用されているようです。現実には、既に保有しているシステムの基盤（実証されているハードウェア、OS、ソフトウェア）や技術（実証されている処理方式、設計標準や実装標準など）によって担保できる品質要素も多々あります。重点事項の漏れを防ぐために要求品質展開表は網羅的に作りますが、その後は要求品質に対して重要度を設定し、重点項目を絞ります。

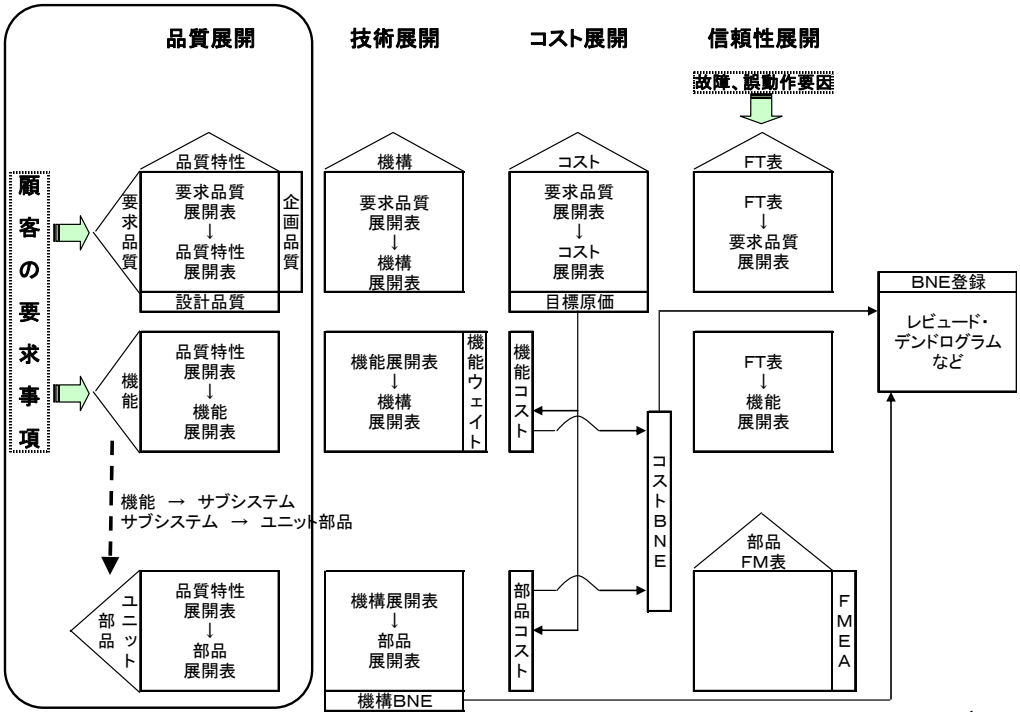


図 5-1　品質展開構想図（例）

5.3.1　品質機能展開の原理

(1) 展開の原理

次の 2 つの側面があります。1 つは品質機能展開表、品質特性展開表や機能展開表などのように、抽出した個々の要素を 1 次、2 次、3 次・・・と系統的に表示して、構成要素の構造を把握することです。もう 1 つは顧客の要求事項を把握して、これをソフトウェアアイテムや生産現場で行う品質保証活動に割り当てていくことです。

(2) 細分化・統合化の原理

細分化とは、製品に対する顧客の要求事項と、これを実現するための設計要素(例えばソフトウェア構成品)とをそれぞれ分離して検討し、品質要素の強弱や新たな品質要素や

機能の発見を促すことです。一方統合化とは、例えば製品に対する顧客の要求事項と設計要素とを二元表の形式で結合して、両者の因果関係を明らかにすることです。

その結果､それぞれの品質要素に対し要求水準と要求の重要度を定めてこれをまとめ、製品全体の品質の水準を定めることになります。

(3) 多元化・可視化の原理

品質展開は、図 5-1 に記載の通り各種展開表間の二元表から構成されているので、多元的にとらえることができます｡またそれぞれの二元表によって､品質要素や技術要素、故障・誤動作要因などが系統的に可視化されていること、さらにそれらの相互関係が二元表で示されていることから、情報共有が促されます。

(4) 全体化・部分化の原理

部分最適と全体最適のバランスをとることです。作成された二元表全体から重要な要素を抽出し、また部分的にとらえて具体化する一方で、その結果について全体とのバランスを考慮します。

(5) 変換の原理

図 5-1 に記載の通り各種展開表を組み合わせることで、顧客の要求に基づいて作成する品質表から、これを実現するための品質特性に変換し、さらにサブシステム、部品へと伝達していきます。

5.3.2 品質展開の手順概要

品質展開では展開と変換を繰り返していきますが、展開は構成要素を細分化して特定すること、変換は要求を構成要素に漏れなく割り付けることです。

(1) 品質表の作成

図5-1に示す通り品質展開は、顧客*の要求事項をインプットします。顧客の要求事項は､顧客の満足を高めるために提供する製品に対する顧客のニーズおよび期待を把握し、製品設計に反映させるための情報です。エンタプライズ系のシステムでは、不特定ユーザ向けの製品とは異なり、企業の業務をシステム化の対象としていますから、機能の多くはシステム化の対象とする業務から直接導き出されますので、図5-1では顧客の要求事項から要求品質展開表および機能展開表を導出することとしています。

品質表の作成は、まず、顧客の要求事項を要求品質に展開し顧客分野の言葉で要求品質展開表を作成し、一方で技術分野で表現した代用特性である品質特性を展開して品質特性展開表を作成します。次に、要求品質を品質特性に変換して、二元表を作成することにより、顧客分野から技術分野に変換します。これを品質表と呼び、要求品質に対応してその要求水準を企画品質として定め、これを技術分野で表現した品質特性に変換してその要求水準を設計品質として定めます。

このステップは、ソフトウェア開発でいえば主に要件定義に該当します。ソフトウェア開発においても、非機能要求を引き出して、優先順位付けして背反する要求を解決して定義することが重要な課題となっており、これを、品質機能展開では要求品質から非機能要求に相当する代用特性に変換し、代用特性（非機能要求）を機能に対応付ける方法を提供しています。

また背反する要求を解決するときに優先順位を利用できるように、個々の要求品質に対して重要度を設定して、AHP 等の手法を用いて分析する方法を用いて相対的な重要度を求めます。

---

* 本節では、エンドユーザに限定しない様々なユーザの総称として「顧客」を用いる。

(2) 技術展開、コスト展開および信頼性展開

品質表で定めた企画品質や設計品質を実現するために、技術展開で実現性を、コスト展開で経済性を、信頼性展開では安定性を検討して、構成要素の設計品質を定めるとともに、ボトルネックを解決していきます。

技術展開では、代用特性を機構（仕組み、構造）とも対応付けて分析します。コスト展開では、設計品質の重要度などに着目し目標コストを機能、部品などに配分したコストと、設計の結果によって推定するコストとを比較して、そのギャップをボトルネックとして抽出し、これを解決します。従って技術展開、コスト展開および信頼性展開は、ソフトウェア開発でいえばフィージビリティスタディに該当します。

(3) 下流工程への展開

品質特性を、機能、ユニット部品へと展開していき、要求品質を実現するためのシステム構成を検討します。これはソフトウェアの設計から実装の過程に対応しており、それぞれの段階でフィージビリティスタディ†を行います。

(4) トレーサビリティの記録

(1)から(3)の一連の流れを品質展開といいます。品質展開において繰り返し行われる変換の過程で作成する各種の二元表が、双方向（上流から下流、下流から上流）のトレーサビリティを表現します。

トレーサビリティを確保することは、初期の開発においてはシステム品質の担保にも深くかかわります。すなわち、開発の各段階で作成する各種設計書において、それぞれの段階の構成品に対して設計品質を示して次の段階の設計にインプットするとともに、設計検証においてトレーサビリティの完全性を担保することによって、品質を保全します。

また、開発段階における変更や修正および稼動後のシステム改良時に、要求の変更に関係する構成品を確実に把握してシステム全体を最適化すること、システム全体で最適化され作成された構成品の修正において、システム品質に立ち返って検証するなど、システム品質を維持するために特に重要な情報となります。

従ってトレーサビリティを管理するとは、ソフトウェアの設計および設計検証においては、次の 4 つの観点から設計の正確性および妥当性を確認し、これをトレーサビリティの記録(各種の二元表をはじめとするトレーサビリティを表現した情報)として残し、維持していくことになります。

- 要求（機能要求および非機能要求）を網羅していること
- 実現性が担保されていること
- 経済性が担保されていること（利益が担保されていること）
- 安定性（故障、誤動作や誤操作などの外乱に対するリスクの顕在化を防止する）が担保されていること

5.3.3 品質展開の規則

(1) 品質展開における要求品質

品質展開でいう要求品質は、機能に機能の達成水準である品質を修飾語として付加して表現します。ソフトウェア開発の一般的な用語でいえば、要求品質は要件に対応するもので、要求品質展開表は、顧客の要件を 1 次、2 次、3 次・・・と階層的に表した要件一覧表に位置づけられます。

ソフトウェアの設計は、上位の設計段階の設計書に定めている命題の実現手段を検討

† 技術展開、コスト展開および信頼性展開

して、その実現手段として当該設計段階における構成品を定め、それぞれの構成品に対して命題を定めていきます。上位の設計段階の命題に定めている品質（機能の達成水準）は、当該設計段階の構成品の命題では機能に変わる場合もありますが、要求品質の上流から下流への展開は、設計段階におけるこのような変化を加えることなく展開して、要求品質とサブシステムの対応付け、要求品質と部品との対応付けを行うことによって、サブシステムと要求品質および部品と要求品質とのトレーサビリティを確保します。

(2) 機能と品質とを厳密に区別

品質展開では、機能と品質とを厳密に区別して扱います。顧客から収集して確認した機能を、品質特性などと対応させて具体化していく過程で、要求を満たす実現手段として機能を付加する場合がありますが、付加された機能は機能として扱い、品質とは区別します。品質は機能を果たすための速度、効率、安定性、利用者の心理特性などで、機能に対して識別します。

(3) 要求品質の表現規則

要求品質は、次の５項目を守って記述することとされています。

① 表現が具体的である
② ふたつ以上の意味を含めない
③ 否定的表現を避ける
④ 方策や対策を含まない
⑤ 体言止めとしない

表 5-1 に規則に反する事例を挙げています。

表 5-1 要求品質の悪い事例

悪い事例	事由
分類してラベルを付ける	意味が２つ
大量印刷を避ける	否定的
検索機能を強化する	対策
グラフィックに表示	体言止め

5.3.4 品質展開の手順

品質展開の手順を図 5-2 に示します。

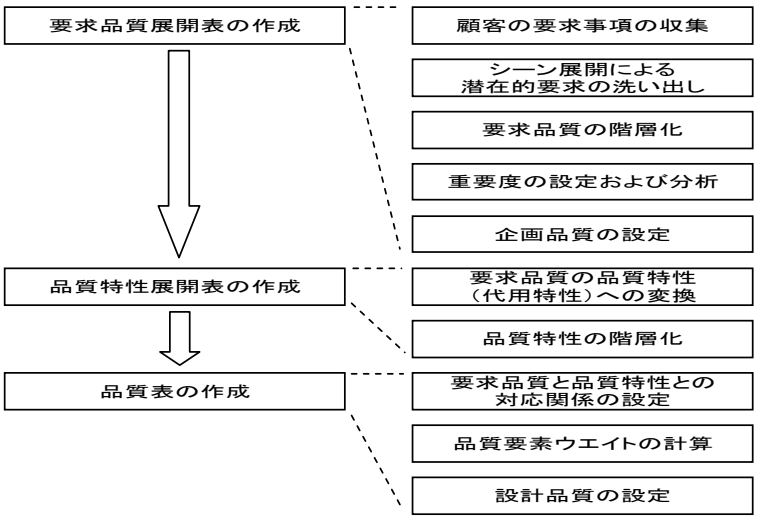


図 5-2 品質展開の手順

市場調査などで得た要求品質の全体を把握し、階層的に構造化して要求品質展開表を作ります。ボトムアップで洗い出した要求品質を、親和図法によってグルーピングし系統表示して要求品質展開表を作成します。

集めた情報が製品に使われるシーン（エンタプライズ系のシステムではユースケース）を想定して、また 5W1H を確認して潜在的要求（付加すべき機能、満たすべき品質など）を洗い出します。

もちろん、この方法に変えてゴール指向要件分析手法（AGORA、NFR、KAOS、GQM、I＊“アイアスター”）などの要求分析手法を用いて洗い出すこともできます。

5.4 品質展開の要点および期待効果

品質機能展開をソフトウェア開発に適用するには、いくつかの問題点が指摘されております。

- QFD だけでは、状態遷移図や DFD のように時間の流れが表現できない。
- ソフトウェア開発は、顧客にとっても要求があいまいな状態から出発するため、顧客要求があいまいな状態から適用できなければならない。
- 選択や判断が必要な場合の記録を QFD では定めていない。オーダメイドのソフトウェア開発は、」長期にわたり顧客要求が時間とともに変化するので、最初に特定した顧客要求を前提とした品質管理はなじまない。
- 品質表の作成などの作成負荷が高い。

本節では、ソフトウェア製品のトレーサビリティを管理するために有効と考えられる事項を取り出し、期待効果および期待効果を獲得する上での要点を整理します。品質機能展開をそのままソフトウェア開発に適用するのではなく（すなわち、トレーサビリティを二元表で表すことにとらわれずに）、それぞれの組織が抱えているトレーサビリティに関連する課題に応じて、これらのソフトウェアの開発およびその品質管理の仕組みの中に、要点を組み込んで活用いただくことを想定しています。

(1) ソフトウェア製品に対する顧客の要求を品質機能表現で簡潔に表現して一覧する

① 期待効果

- 要求品質の表現規則（2 つ以上の意味を含めない、方策や対策を含まない）をもとに顧客の要求を表現することで、要求項目の粒度が均一になり、要件管理システムなどにより要件がハンドリング可能となり、トレーサビリティ管理にシステムを利用できる可能性が高まる。
- 顧客の要求を品質機能表現（機能に機能の達成水準を付加して表現）すること、および要求品質の表現規則（表現が具体的である）により表現することにより、非機能要求を具体的に特定できる。

② トレーサビリティを管理する上での要点

- システム化の対象とする業務自体は、業務用語および業務処理用語などにより、要求品質の記述とは別に定める。要求品質における機能の表現においては、業務用語および業務処理用語など定義された用語を用いることによって、簡潔に表現する。
- 要求品質と JIS 規格で定める品質特性や、それぞれの組織でソフトウェアの品質として定義している特性との関連付けを考えることにより、「補助的品質」を補強し、あいまいなものを具体化する。
- ユースケースから 5W1H、すなわち「誰が」、「何を」、「なぜ」、「いつ」、「どこで」、「どのような」潜在的要求を確認することにより、品質（機能の達成レベル）を具体化する。

・安定性（故障、誤動作や誤操作などの外乱に対するリスクの顕在化を防止する）要求を組み込む。

(2) 要求品質の相対的な優先順位を設定する

① 期待効果

・対立する要求項目のトレードオフにあたり、客観的根拠を提供できる。
・コスト分析を通して、開発対象の要求の取捨選択に客観的根拠を提供できる。
・レベルアップおよび重点ポイント（セールスポイント）を可視化して、優先順位に組みいれることにより、重点指向で要求項目を絞り込むことができる。

② 期待効果を獲得する上での要点

・ソフトウェアを開発する組織が持ちえている技術レベルと、顧客の要求に基づく技術レベルとのギャップをレベルアップ率として認識し、優先順位に組み込む。
・他社比較はビジネスニーズにある場合のみに限定して実施する。
・経営ニーズに基づいてセールスポイントを識別する。
・優先順位に基づいてトレーサビリティ管理の対象を絞り込み、品質管理の効率化を図る。

(3) 要求品質から機能、サブシステム、部品への一貫したトレーサビリティを管理する

① 期待効果

・設計および設計検証における検討、確認ポイントに品質機能展開の観点を利用することによって、有用性（要求品質を網羅していること）、実現性、経済性および安定性（故障、誤動作や誤操作などの外乱に対するリスクの顕在化を防止する）の 4 つの観点でトレーサビリティを管理することにより、信頼性を高めることが期待できる。
・要求品質から機能に展開する段階で、顧客分野で表現した要求品質を技術分野で表現した代用特性（品質特性）に変換することにより、要求される品質特性および設計品質を具体的に定義できるので、開発の過程でトレーサビリティが確保されていることを容易に確認できる。

② 期待効果を獲得する上での要点

・要求品質を機能、サブシステム、部品へと展開し、ソフトウェア開発工程のそれぞれの局面（設計、実装、テストなど）でトレーサビリティを確認し記録する。
・要件の変更時には、要求品質までさかのぼって再度トレーサビリティを確認し記録する。
・要求品質を洗い出すときに用いたユースケースと機能とを関連付けて参照しつつ、機能に割り当てる要求品質から機能に対する設計品質を定義して、以降サブシステム、部品へと引き継いで、妥当性を確認していく。

5.5 ソフトウェア開発におけるトレーサビリティ管理の具体例

5.5.1 顧客要件一覧表の作成事例

サンプルシステム（教育受講管理システム）を用いて、実施例を紹介します。

ユーザ部門がユースケースから抜き出して提示された要求品質の原始情報を表 5-2 に示します。

表 5-2　サンプルシステムの要求品質の原始情報

1次	アクタ	2次
使いやすい	受講者	いつでも使える
		何処でも使える
		誰でも使える
		過去の履歴が見える
		お奨めが分かる
		全社のトレンドが分かる
		好きなときに勉強できる
満足できる	受講者	やる気を削がない
安心して使える	受講者	他人には見えない
		受講中に教材を没収されない
使いやすい	上長	部下が何を受講しているか分かる
		部下の過去の履歴が見える
		全社および配下のトレンドが分かる
		受講目標と対比できる
		受講効果を確認できる
使いやすい	教材管理者	いつでも使える
		在庫数が分かる
		在庫の変化が読める
		競合者が分かる
		誰が使っているか分かる
		全教材のトレンドが分かる
		教材の在処を追跡できる
		教材への満足度を確認できる
安心して使える	教材管理者	実在庫と一致している

原始情報に対して、次の事項を実施して要求品質を4階層に整理した結果を、表 5-3および表 5-4に示します。
これを顧客要件一覧表として、トレーサビリティの管理に利用します。

・要求と JIS 規格で定める品質特性や、それぞれの組織でソフトウェアの品質として定義している特性との関連付けを考えることによって「補助的品質」を補強する。
品質機能表現によって要求品質を記載しなおす。その結果を、表 5-4の4次レベルの要求品質に示す。
・4 次レベルの要求品質を親和図法をもとにグルーピングして統合する。

表 5-3　サンプルシステムの要求品質展開表（3 次レベルまで）

1 次		2次		3 次	
1	運用しやすい	1.1	受講しやすい	1.1.1	好きなときに勉強ができる
				1.1.2	受講期限を管理しやすい
				1.1.3	受講講座を選びやすい
				1.1.4	過去の受講履歴が分かる
				1.1.5	目標管理ができる
		1.2	部下の受講を管理しやすい	1.2.1	部下の受講を管理しやすい
				1.2.2	部下のスキルアップを指導しやすい
		1.3	教育の提供管理をしやすい	1.3.1	教材在庫を管理しやすい
				1.3.2	教材の競合調整をしやすい
				1.3.3	教材の回収／発送管理をしやすい
				1.3.4	修了試験提供管理をしやすい
				1.3.5	教材への満足度を確認できる
2	使いやすい	2.1	作業効率がいい	2.1.1	手間が少ない
				2.1.2	必要な情報を入手しやすい
				2.1.3	作業効率がいい
		2.2	業務実態と適合している	2.2.1	業務実態と適合している
				2.2.2	代理操作ができる
		2.3	操作が分かりやすい	2.3.1	操作が分かりやすい
		2.4	リアルタイムに処理できる	2.4.1	リアルタイムに処理できる
		2.5	何時でも使える	2.5.1	何時でも使える
3	安心して使える	3.1	個人情報が保護されている	3.1.1	個人情報が保護されている
		3.2	故障しない	3.2.1	故障しない
		3.3	何処でも使える	3.3.1	何処でも使える
		3.4	環境に依存しない	3.4.1	環境に依存しない
		3.5	正確である	3.5.1	正確である

表 5-4　サンプルシステムの要求品質展開表（４次レベル）

３次		４次	
1.1.1	好きなときに勉強ができる	1.1.1.1	社員は月の途中でも思い立ったときに受講の申し込みができる
		1.1.1.2	社員は受講期間（受講開始時期、受講終了時期）を自分が設定できる
		1.1.1.3	社員は受講申し込みの際に講座の学習標準時間がわかる
		1.1.1.4	社員は月の途中からでも思い立った時に修了判定試験の申し込みができる
		1.1.1.5	社員は修了判定試験の受験期限を自分で設定できる
		1.1.1.6	社員は受講希望の講座の受講に必要なPC環境等が分かる
		1.1.1.7	教材は受講を終了するまで、受講者は教材を占有使用できる
1.1.2	受講期限を管理しやすい	1.1.2.1	社員は自分が現在受講している講座の受講期限を照会できる
		1.1.2.2	社員は自分が現在受講している講座のステータス（受講申し込み～教材返却）を照会できる
		1.1.2.3	社員が教材をメールなどで発送した時点で受講者の教材返却責任は終了す
1.1.3	受講講座を選びやすい	1.1.3.1	カリキュラム別に次のプロフィールが分かる a)目指すキャリアとの関連 b)受講の前提となる経験、事前に修了しているべき講座 c)受講推奨層（経験年数、資格別など）など受講推奨事項 d)学習標準時間および受講期間 e)学習に必要な環境 f)修了認定試験
		1.1.3.2	社員は任意の講座について自分が受講したか否かを照会できる
1.1.4	過去の受講履歴が分かる	1.1.4.1	社員は自分の受講履歴および成績を照会できる
		1.1.4.2	社員は自己の受講済みの講座の一覧を照会できる
1.1.5	目標管理ができる	1.1.5.1	社員は自分が目指すキャリアに対して、講座の修了度合いをグラフィカルに照会できる
		1.1.5.2	社員は自分が目指すキャリアに対して、受講優先順位別の講座の一覧をグラフィカルに照会できる
		1.1.5.3	社員は自分と同レベル（経験年数、資格など）の社員が、受講している講座の一覧を照会できる
		1.1.5.4	社員は自分が受講した講座について自分の位置（偏差値）をグラフィカルに照会できる

5.5.2 要求品質の優先順位付け事例

サンプルシステム（教育受講管理システム）を用いて、実施例を紹介します。
表 5-3に記載する2次レベルの要求品質要素間を一対比較して、AHPによって分析した結果を表 5-5に示します。

- 一対比較による相対価値は小数点以下の表示を省略しています。「0」と表記されている箇所は、対角に表記されている整数値の逆数になります。
- 品質企画では他社比較の必要はないので、自社の現状品質のみを評価して記載しています。この例では、従来手作業で実施していた業務をシステム化することを前提として、現状品質を評価しました。企画品質には、システム化することで目標とする品質を設定し、レベルアップ率を求めています。
- 教育を提供するユーザ部門は、システム化に当たって第 1 に「社員の自主的な受講を推進すること」を、第 2 に「上司が社員のキャリアアップを指導して推進すること」を目標としているとのことなので、これをセールスポイントとして挙げています。

表 5-5 サンプルシステムの AHP による重要度分析結果

	対比較による相対価値													幾何平均	重要度	品質企画					
																自社現状品質	企画			ウェイト	
	受講しやすい	部下の受講を管理しやすい	教育の提供管理をしやすい	作業効率がいい	業務実態と適合している	操作が分かりやすい	リアルタイムに処理できる	何時でも使える	個人情報が保護されている	故障しない	何処でも使える	環境に依存しない	正確である				企画品質	レベルアップ率	セールスポイント	絶対ウェイト	要求品質ウェイト
受講しやすい	1	3	0	0	0	0	0	0	0	3	7	5	0	0.618	0.040	2	4	2.0	◎	0.120	0.06
部下の受講を管理しやすい	0	1	0	0	0	0	0	3	0	3	5	3	0	0.571	0.037	2	4	2.0	○	0.089	0.04
教育の提供管理をしやすい	3	5	1	5	3	3	0	3	5	5	7	7	0	2.423	0.157	2	5	2.5		0.392	0.18
作業効率がいい	5	5	0	1	1	1	0	5	1	3	7	5	1	1.619	0.105	2	4	2.0		0.209	0.10
業務実態と適合している	3	3	0	1	1	0	0	3	0	3	5	3	0	1.046	0.068	4	4	1.0		0.068	0.03
操作が分かりやすい	5	5	0	1	3	1	0	3	0	0	5	3	0	1.132	0.073	3	3	1.0		0.073	0.03
リアルタイムに処理できる	7	7	5	5	5	5	1	7	1	5	9	9	1	4.080	0.264	2	5	2.5		0.660	0.31
何時でも使える	3	0	0	0	0	0	0	1	0	0	5	5	0	0.522	0.034	4	4	1.0		0.034	0.02
個人情報が保護されている	5	5	0	1	3	3	1	5	1	5	7	5	1	2.257	0.146	4	4	1.0		0.146	0.07
故障しない	0	0	0	0	0	3	0	5	0	1	5	5	0	0.686	0.044	3	3	1.0		0.044	0.02
何処でも使える	0	0	0	0	0	0	0	0	0	0	1	0	0	0.190	0.012	1	1	1.0		0.012	0.01
環境に依存しない	0	0	0	0	0	0	0	0	0	0	5	1	0	0.316	0.020	3	3	1.0		0.020	0.01
正確である	7	5	5	1	5	5	1	5	1	5	7	3	1	3.085	0.200	3	4	1.3		0.266	0.12
																				2.133	1.00

(1) 優先順位に基づく背反要求の調整

背反する要求は、「業務実態が適合している－ネットワークに接続できない環境にある受講者のために、受講申込みは代理人が申込みできる」という要求と、「個人情報が保護されている－受講の申込みは本人のみができる」という要求とが背反していました。要求品質ウェイトは、「個人情報が保護されている」が 0.07 であり、「業務実態と適合している」が 0.03 ですので、「個人情報が保護されている－受講の申込みは本人のみができる」という要求を採用します。

(2) 優先順位に基づく要求の絞込み

「どこでも使える」および「環境に依存しない」の要求品質ウェイトは、0.01 と低

いので不採用とします。さらに詳細な絞込みを行う必要がある場合は、ある 2 次レベルの要求品質配下の 3 次レベルの要求品質要素間、4 次レベルの要求品質要素間を一対比較して行ないます。

5.5.3 要求品質に基づくトレーサビリティ管理事例

サンプルシステム（教育受講管理システム）を用いて、実施例を紹介します。現実には、トレーサビリティ管理を支援するツールの整備と併せて取り組むことが必要です。

(1) 要求品質の機能への割当て

表 5-6に示す機能に要求品質を割り当てた結果を、表 5-7に示します。

機能項目に要求品質を割り当てた後に、要求品質を代用特性（特性 1～特性 n）に変換して設計品質を定めます。

代用特性は、JIS 規格で定める品質特性などを参考に定めます。

- 操作性　画面操作性（画面サイズ、キータッチ数等）、デザイン操作性（一覧性、直感性等）
- 信頼性　安全性、故障率、誤操作防止性　等

表 5-6　サンプルシステムの機能一覧

分類	アクタ	機能項目
照会	受講者	受講結果を照会する
	上長	部下の受講状況を照会する
	受講者	全社の受講トレンドを照会する
	上長	全社の受講トレンドを照会する
	教材管理者	教材の在庫を照会する
申し込み	受講者	受講を申し込む
	上長	受講希望を承認する
	教材管理者	優先度を調整する
	教材管理者	教材を発送する
終了	受講者	修了判定試験を申し込む
	教材管理者	修了認定試験を提供する
	受講者	受講結果を報告する
	上長	受講結果を承認する
	受講者	教材を返却する
	教材管理者	教材を回収する
		受講結果を照会する
年度更新	教材管理者	カリキュラムを設定する
	教材管理者	教材の棚卸しをする

表 5-7　機能への要求品質割当表の例

分類	照会			
アクタ	受講者			
機能項目	受講結果を照会する			
要求品質		設計品質		
ＩＤ	要求品質	特性１	・・・	特性ｎ
1.1.2.1	社員は自分が現在受講している講座の受講期限を照会できる			
1.1.2.2	社員は自分が現在受講している講座のステータス（受講申し込み～教材返却）を照会できる			
1.1.3.2	社員は任意の講座について自分が受講したか否かを照会できる			
1.1.4.1	社員は自分の受講履歴および成績を照会できる			
1.1.4.2	社員は自己の受講済みの講座の一覧を照会できる			
1.1.5.1	社員は自分が目指すキャリアに対して、講座の修了度合いをグラフィカルに照会できる			

(2) 要求品質のサブシステム、モジュールへの割当て

サブシステムの設計段階で、機能をサブシステムに割当て、モジュール別に要求品質および品質特性を割り当て、その結果はユーザを含めて確認します。その結果を表 5-8 に示します。要求品質を割当てた結果は「要求品質トレーサビリティ」欄に、品質特性を割り当てた結果は「品質特性トレーサビリティ」欄に記載しています。

表 5-8　要求品質のサブシステム、モジュールへの割当て例

分類	照会
アクタ	受講者
機能項目	受講結果を照会する
サブシステム	受講状況を照会する

要求品質		要求品質トレーサビリティ	
ＩＤ	要求品質	実現モジュール	実現手段
1.1.2.1	社員は自分が現在受講している講座の受講期限を照会できる	受講中講座照会	①講座別に受講期限を表示する ②修了認定試験申し込み前の場合は、修了認定試験申し込み期限を表示する
1.1.2.2	社員は自分が現在受講している講座のステータス（受講申し込み～教材返却）を照会できる	受講中講座照会	講座別に次のステータスを表示する ①受講希望承認待ち ②受講申し込み受理待ち ③教材到着待ち ④修了認定試験申し込み待ち ⑤修了認定試験提供待ち ⑦修了認定試験受験中 ⑧受講結果報告待ち ⑨受講結果承認待ち ⑩教材返却待ち ⑪教材返却済 ⑫受講期限切れ
1.1.3.2	社員は任意の講座について自分が受講したか否かを照会できる	講座照会	分野別、キャリア別に講座を一覧し、修了済／未修了/未受講のいずれかを表示する
1.1.4.1	社員は自分の受講履歴および成績を照会できる	受講済講座照会	受講済講座一覧に、次の項目を表示する ①受講終了日（受講結果報告日） ②修了ステータス（修了／不合格／受講断念（受講期限切れ）） ③修了認定試験点数
1.1.4.2	社員は自己の受講済みの講座の一覧を照会できる	受講済講座照会	受講済の講座を一覧する
1.1.5.1	社員は自分が目指すキャリアに対して、講座の修了度合いをグラフィカルに照会できる	キャリア充足照会	①目指すキャリアに必要な講座を一覧する ②講座には受講順位および受講の前提となる必要経験を示し、受講中に順に並べる ③修了状態は、色で直感的に表現する
品質特性		品質特性トレーサビリティ	
特性	設計品質	関連モジュール	実現手段および／または設計品質
画面操作性	キータッチ数	全画面	全ての操作にショートカットキーを割り当てる
	選択簡易性	講座照会 キャリア充足照会	①受講者のキャリア志向から選択する分野をナビゲートできる ②受講者の経験から選択するキャリアをナビゲートできる
デザイン操作性	一覧性 （14.インチ全画面で横スクロール不要とする）	全画面	①画面デザインで、１４.インチ画面で、横スクロール不要とする ②物理的に収まらない場合は、ポップアップまたはドリルダウン
	直感性（知りたい情報を直感できる）	キャリア充足照会	①修了状態は、色で直感的に表現する ②受講優先順位別に講座を並べ一覧する ③受講者の経験によって、受講可能な講座を色で直感的に表現する ④受講を希望する講座を選択することによって、受講に要する
性能	LAN接続において、レスポンスタイム1秒以内	全モジュール	データアクセスに要する時間を0.5秒以内とする

コラム：第 5 章の用語説明

第 5 章では、様々な品質機能展開にかかわる用語を活用して解説を行っていますので、説明が必要な用語については本コラムにまとめて掲載します。

要求品質

機能とは、ある「もの」が持つ働き、役目であり、「名詞」+「動詞」で表します。品質とは、その働きや役目が持つ特徴（性質）なので、機能の達成水準を意味します。機能に対して、その達成度を表す修飾語を付加した簡潔な表現を「品質機能表現」といいます。要求品質とは、顧客の求める真の品質を機能中心に体系化して「品質機能表現」で記載したものです。

トレーサビリティ

トレーサビリティ（traceability）は、日本語で「追跡可能性」「履歴追跡」とも呼ばれ、幅広い分野で使われる用語です。原義は“トレース（追跡）ができること”であり、そこから発展して、あるものの来歴や行方、所在、構成・内容、変化・変更の履歴などを後から確認できることを意味しています。ISO9000：2000 おいては「考慮の対象となっているものの履歴、適用または所在を適用できること」と定義されており、具体的には「処理の履歴」「材料および部品の源」などが挙げられています。

ソフトウェア開発では、ソフトウェアの要件定義書、仕様書、変更履歴、テストや障害の記録、ソースコード、バージョン、実装などを相互に参照・確認できるような仕組み、紐付けることができるシステムを指します。

品質展開

品質展開（QD: Quality Deployment）とは、顧客の声を製品やサービスの開発につなげるための手法で、新製品開発の現場など多くの「ものづくり」の現場で、国内・海外問わずに活用されているフレームワークです。近年は、米国を中心にシックスシグマの主要ツールとして活用されているほか、製造の現場のみならずサービスやソフトウェア開発の現場、およびナレッジマネジメントとも連携できる手法として注目を浴びています。品質展開は品質機能展開として統合され、既に JIS Q 9025：2003 として JIS 化されています。また、プロジェクトマネジメント知識体系ガイド（Project Management Body Of Knowledge, 略称 PMBOK）でも、スコープ定義に役立つテクニックであると紹介され、近年ではソフトウェア開発分野においての適用事例も増えてきました。

機能

機能とはある「もの」が持つ働き、役目であり「名詞」+「動詞」で表します。品質とは、その働きや役目が持つ特徴（性質）なので機能の達成水準を意味します。機能に対してその達成度を表す修飾語を付加した簡潔な表現を「品質機能表現」といいます。要求品質とは、顧客の求める真の品質を機能中心に体系化して、「品質機能表現」で記載したものです。

展開

要素を順次「変換」の繰り返しによって、必要とする特性を求める操作のことです。

変換

要素を次元の異なる要素に対応関係をつけて置き換える操作のことです。

二元表

2 つの展開表を組み合わせて、それぞれの展開表に含まれる要素の対応関係を表示した表のことです。

FTA/FMEA

FTA は、故障の発生メカニズムを論理記号により解析するものです。故障をいくつかのパターンの現象として分類して、その発生メカニズムを論理記号により故障モードを系統図で示したものが FT 図で、FT 図を展開表に書き直したものが FT 展開表です。故障モードは故障の状態が観測でき、過去の製品または類似製品の故障モードが情報として蓄積されていれば、これらを論理的に検討して発生のメカニズムを推定できます。FMEA は故障モードとその影響を解析するもので、事前に故障を予測して予防するために実施するものです。

< 空欄 >

第6章　テスト網羅性の高度化技法

第3章では、ソフトウェアテスト技法の基本について説明しました。ところがソフトウェアテストの現場においてテスト技法を知っていても、その適用方法が分からないという声を多く聞きます。またやみくもにテスト技法を適用したために、テスト技法の適用箇所のみ詳細にテストがされ、全体としては大きなテスト漏れが発生しリリース後に大問題が発生したという話もよく聞きます。つまりテスト技法を知っているだけでは、高信頼ソフトウェアを開発することはできないのです。

それではどうしたらよいのでしょうか？ 実は、従来のテストには3つの問題が隠れています。1つ目の問題は、テストだけでソフトウェアの欠陥をすべて見つけようと考えがちであることです。ソフトウェアテストは、欠陥の検出に非常に有効な方法です。しかし前章で述べた通り「要求－要件－仕様－設計－コード－テスト」について、トレーサビリティを取ってソフトウェア開発活動全体のなかで、欠陥発生を予防していくということを同時に考えていく必要があります。

2つ目の問題は、活動成果の把握の問題です。1つのソフトウェア開発には、非常に多くの組織が関与します。それぞれの部署による活動成果の見える化が不十分な場合、その皺寄せは最終工程であるテストに集中します。そのようなケースでは、人海戦術を使ってカバーするといったことが多く行われています。さらに、テスト工程に対するインプットとなる開発成果物と、テスト工程の成果としてアウトプットされる製品との品質の違いが明確に示されない場合は、組織全体の改善のきっかけを失います。このような組織においては振り返りが行われないため、次もまた同様の失敗を繰り返すことになります。

3つ目の問題は、ソフトウェアテスト自体の問題です。テストに対する要求の分析が不十分なまま、いきなり仕様書からテストケースを作成しているようではテスト全体の網羅性を確保することはできません。テストに対する要求の分析をきちんと実施するとともに、テスト設計における網羅性とピンポイント性についてよく理解し、それぞれ適切なテスト技法を使用することが大切です。

1つ目の問題については前章に、2つ目については『ソフトウェアテスト見積りガイドブック』において考え方や対策が示されています。そこで本章では、3つ目の問題に焦点をあてた解説を行います。

6.1節では、テストプロセスのなかで従来省略されることが多かったテストに対する要求の分析、すなわち「テスト要求分析」について説明し、テストの全体をカバーすることの意味について考えます。

6.2節では、テスト対象をどのようにすれば網羅性が高くテスト設計できるのかについて、エンタプライズ系代表企業十社のテスト実態分析から得られた「テスト観点」を組み合わせることによる文法構造の設計、すなわち「テストアーキテクチャ設計」の方法を説明します。

6.3節では、テスト詳細設計における具体的な「高度化技法」として、直交表を活用した網羅的なテスト方法と、シナリオを用いた効果的なピンポイントテストについて説明します。

6.4節では、エンタプライズ系代表企業十社のテスト実態分析結果を示すとともに、高信頼化に向けたテスト観点を導き出すための「テスト観点表」を示します。

そして6.5節で、6.4節で示したテスト実態分析結果の使い方と、テスト観点表の使い方について解説します。これによって、より十分性の高いテストを計画することができるようになります。

6.1　テスト要求分析

これまで多くのソフトウェアテストの現場では、仕様書から直接テストケースを作成し、続いてそのテストケースに与えるテストデータを用意し、テストを実施するという方法が採られてきました。ところがソフトウェアの大規模・複雑化にともない、この方法ではシステム全体のテスト網羅性が十分に確保できなくなってきました。

そこで、ソフトウェア開発において構造化分析やオブジェクト指向分析が必要となったのと同様に、ソフトウェアテストにおいてもテスト設計前にテスト要求分析*が行われるようになってきました。テスト要求分析には、NGTというテスト観点をツリー型に列挙・整理・検討・詳細化・体系化する方法や、HAYST法のFV表（第2部3章『富士ゼロックス株式会社の事例』を参照）、ゆもつよメソッド、マインドマップ（MM）を使用するTiramisやTAMEという方法が発表されています。これらをまとめると表 6-1のようになります。

表 6-1　テスト要求分析手法

手法名	表現方法	特徴	プロセス
NGT	ツリー	テスト全体をテスト観点で網羅	VSTeP
FV 表	表形式	目的機能の切り口で V&V を網羅	HAYST 法
ゆもつよメソッド	表形式	機能×テストタイプで網羅をチェック	ゆも豆
Tiramis	ツリー（MM）	テストカテゴリ分析を実施。機能はMM	－
TAME	ツリー（MM）	テスト設計思考の可視化とレビューによるテスト観点の洗い出しおよび整理	－

テスト要求分析では、始めにテスト対象の情報収集を実施します。情報にはお客様に属するものと、開発プロジェクト自体に属するものがあります。

お客様に属する情報を収集する目的は、真の要求の把握を行うためです。ソフトウェアテストは、仕様通りに「正しく」製品を作っていることを検証することが第一ですが、お客様の要求と合致しているか、つまり「正しい」商品を作っているか、お客様の問題や課題はこのソフトウェアで本当に解けるのかといった妥当性確認をするという目的もあります。これらお客様に属する情報には、品質、価格、納期、解決すべき問題や課題、要求などがあります。

次に、開発プロジェクトの情報を収集します。こちらは開発プロジェクトの進捗状況、ソフトウェア開発力（個々人のスキルとチームスキル）、投入リソース（人・物・金）、完成度、ソフトウェアの流用元、選択技術（開発言語、OS、COTS、FLOSS 等々）、ここに障害があるとシステム全体が止まってしまうといったシステムの急所(Single Point of Failure）等々の情報です。進捗状況には、静的解析ツールによるメトリクスの収集結果やレビューの指摘件数といった品質に関連が高いものも含まれます。また開発力の結果系情報として、これまでに同じ開発メンバが出した欠陥の分析を実施しスキルが不足している技術項目を把握します。

このようにしてテストのために収集したデータは、サーバに格納して全員が参照可能とします。このデータのことをテストベースと呼び、それらはソフトウェアテストの構成管理の対象となります。つまり、最新バージョンを常に参照可能な状況に維持管理するとともに、変更履歴の参照ができるようにします。

テスト要求分析では、こうして集まったテストベースを参照しながら、前述のテスト要求分析手法を使用してテスト対象を扱いやすい単位まで分解・整理しながら、テストの全体を明らかにします。

---

* 「テスト要求分析」のことを「テスト分析」と呼ぶ場合もありますが、「テスト分析」は「欠陥分析」や「テストの進捗分析」と混同しやすいため、本書ではテストに対する要求の分析という意味で「テスト要求分析」と呼んでいます。

6.2　テスト観点とテストアーキテクチャ設計

テストエキスパートとテスト初心者の差は、気配りや目配りができるか否かの違いといわれています。テスト設計時に注目するポイントの良し悪しがテストの成否を分けるのです。そこで、テスト時に注目するポイントを「テスト観点」として抽象的にまとめ、その情報を横展開することによって、テストエキスパートの考え方を共有することができるようになるという考え方が生まれました。

実際のテストにおいては、様々なテスト観点を上手に組み合わせてテスト条件を作成する必要があります。思い付きでテスト観点を組み合わせるだけでは、網羅的なテストはできません。テスト観点をタイプ分けして、組み合わせるパターン（文法構造）を構造化しておくと、文章を作るようにテスト条件を作成することができます。

本節では、まずテスト観点の4つのタイプについて説明し、その後、文法構造を用いた組み合わせ方すなわちテストアーキテクチャ設計について説明します。

6.2.1　テスト観点

テストには、様々な「観点」が必要といわれています。例えば、Ostrandの4つのビュー（ユーザ、仕様、設計、欠陥）や、ISO/IEC 9126の6つの品質特性（機能性、信頼性、使用性、効率性、保守性、移植性）などが観点の一例です。しかし、これまで用いられてきたこれらの観点はテストとして使用するにはあまりに粒度が粗く、またテストという切り口ではバランスが悪いものでした。例えば、ISO/IEC 9126を使用してテスト設計を実施すると機能性に偏ったテストになりがちでした。

テストとして重要なことは、テストの全体像をテスト要求分析で把握しながら、すべてのテスト観点を漏れなくダブりなく挙げることです。そのためには、テスト用の観点の切り口が必要となります。

エンタプライズ系代表企業十社のテストの現状を調査し、その内容を分析しテスト観点を整理することで典型的なテストの観点をリストアップするとともに、テスト観点の構造を明らかにしました。具体的には次の手順で分析しました。

① 各社のテストのやり方を単語レベルまで分解し、テスト観点の要素を抽出する
② 発見されたテスト観点の要素を、特定の性質に着目し分類しラベルを付ける
③ それらをテスト観点ツリーにまとめる
④ テスト観点ツリーの親ノードと子ノードの関係を精査する
⑤ テスト観点ツリーの兄弟ノードの抽象化の度合い（粒度）について精査する

その結果、テストにおける基本的な観点には、テスト観点1からテスト観点4まで大きく4つのタイプがあることが判明しました。

テスト観点1は基本構造を組み立てるもの。テスト観点2は基本構造から派生構造を作り出すもの。テスト観点3は組み合わせ。テスト観点4は期待結果の網羅性の観点です。

今回テスト観点を、図 6-1のような階層構造で表現することによって、テスト全体を表すことができることが分かりました。またテスト観点3を用いることで、離れた枝に位置する関連性のあるテスト観点を結びつけることが可能であることから、テスト観点のネットワーク構造も表現することができます。それでは4つのテスト観点について、順に説明していきます。

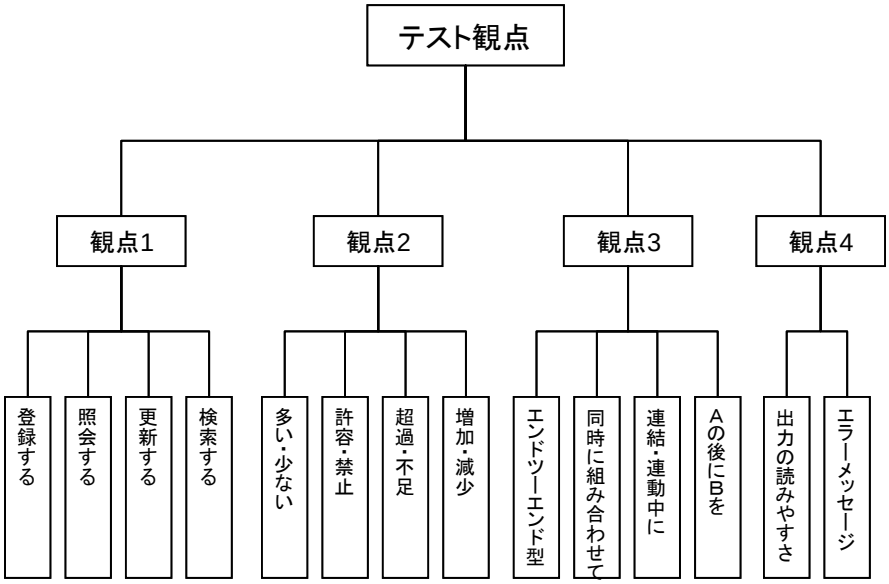


図 6-1 テスト観点ツリー（一部）

テスト観点1は、「～する」という動詞で表現されるものでソフトウェアに対する「入力トリガー」から見つけ出します。具体的には「登録する」、「照会する」、「検索する」などを探します。テスト観点1は、機能そのものにあたります。

テスト観点2は、テストデータや機能のバリエーションを増やすために、それらを修飾する形容詞や副詞で表現されるものです。ソフトウェアの「異常を誘発するための要因」を挙げます。例えば「大量の・少量の」、「連続して・飛び飛びに」、「素早く・ゆっくりと」、「超過して・不足して」といったものがテスト観点2にあたります。

テスト観点3は、「集合、関係、組み合わせ」を示すものです。このテスト観点は、ソフトウェアというよりもシステム全体としてのテスト観点となります。例としては、「エンド・ツー・エンド型で」、「同時に組み合わせて」、「連結・連動中に」などです。

テスト観点4は、「どうなる」という期待結果の属性を表すものです。テスト観点1、2、3は、見つけ出したテスト観点自体をさらに整理・分解してテストを詳細化することが可能です。例えば「登録する」という観点に対して、様々な登録方法を見つけることで分解することができます。ところがテスト観点4は結果系の観点なので、例えば「出力の読みやすさ」という観点を見つけたときに、それを「文字の大きさ」、「文字の種類」、「文字の配置」と分解してもそこから直接にテスト設計することはできません。

テスト設計は､テスト対象ソフトウェアについての入力と期待結果を見つける作業です。そのため入力から期待結果を探しても､また期待結果からそれを導く入力を探してもよいのですが、一般に後者の方が困難になります。従ってテスト観点4については、それを単独で使うケースもありますが、主にはテスト観点1、2、3を使用しテストケースを作成した後に、期待結果の観点で漏れがないか網羅するために使用するとよいでしょう。

以上をまとめると、表 6-2のようになります。

表 6-2 テスト観点の 4 つのタイプ

テスト観点	分類	例
テスト観点 1	動詞	「登録する」、「照会する」、「検索する」
テスト観点 2	形容詞・副詞	「大量の」、「連続して」、「素早く」
テスト観点 3	関係・組み合わせ	「エンド・ツー・エンド型で」､「同時に組み合わせて」
テスト観点 4	期待結果	「出力の読みやすさ」

6.2.2 テストアーキテクチャ設計

これらのテスト観点を使用して、網羅的なテストを設計する方法について説明します。まず、テストは何をテストするのか、つまりテスト対象を発見し決定するところから始まります。

テスト対象は、図 6-2に示されるように3つの軸でとらえることが多いものです。例えば、単体テストのときに規模が小さく、テストアーキテクチャがバッチ処理のテスト対象を考えると、「モジュール」とか「関数」といったものがテスト対象として見つかります。また同じ単体テストであっても、規模が大きくオンラインといったテストアーキテクチャのテスト対象を考えると、「画面」といったテスト対象が見つかります。このように当該テストフェーズにおいて、規模とテストアーキテクチャの組み合わせを考えることでテスト対象を発見することができます。

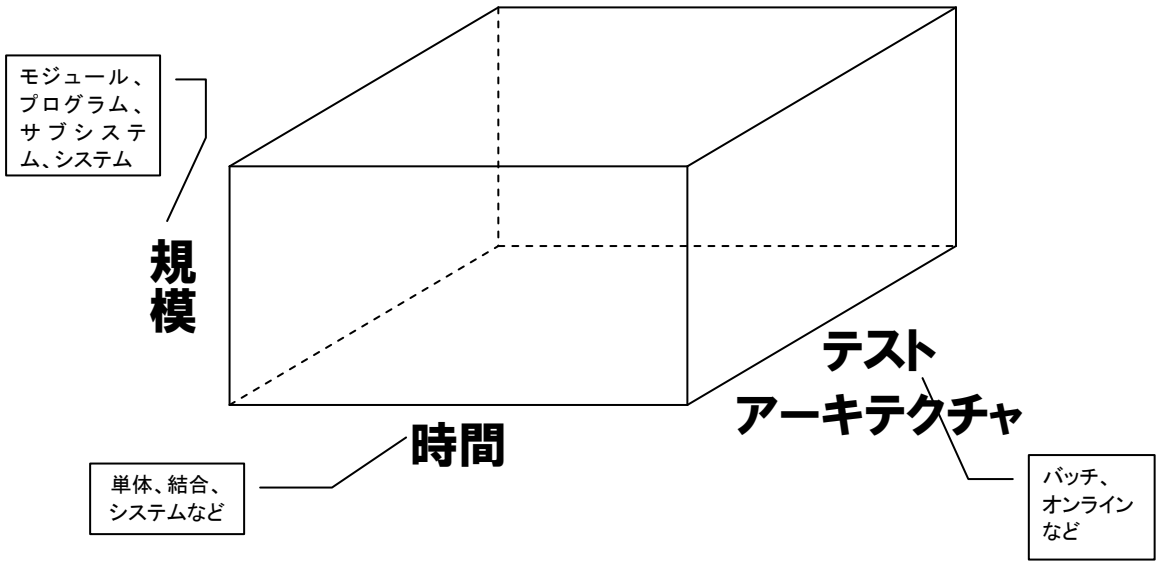


図 6-2 テスト対象とは

次に、発見したテスト対象に対して、テスト観点1、2、3、4を用いてテストを作成していきます。このとき、テスト対象とテスト観点を用いてどのようなテストができるか、先に構造を組み立てておきます。つまり、テスト観点を用いて【テストタイプ】を文法構造で整理しておきます。これは、テストモデルを作成することと同義です。

(1) 基本構造

テスト観点1（動詞）を使用することで、テストの基本構造を構築することができます。例えば【機能網羅テスト】というテストタイプを、｛テスト対象｝と｛テスト観点1｝を使用してその文法構造を書くと、

【機能網羅テスト】
　　｛テスト対象｝に｛テスト観点1｝させる

となります。同様に、
【入力網羅テスト】
　　｛テスト対象｝に（目的語）を｛テスト観点1｝させる

【状態網羅テスト】
　　（XX状態）にある｛テスト対象｝に（目的語）を｛テスト観点1｝させる

【操作網羅テスト】
（XX状態）にある｛テスト対象｝に（操作）することで
（目的語）を｛テスト観点1｝させる

といったように、基本構造を追加していくことが可能です。Tiramisという方法論では、これらの基本構造を構成する具体的な要素を図に表しています。

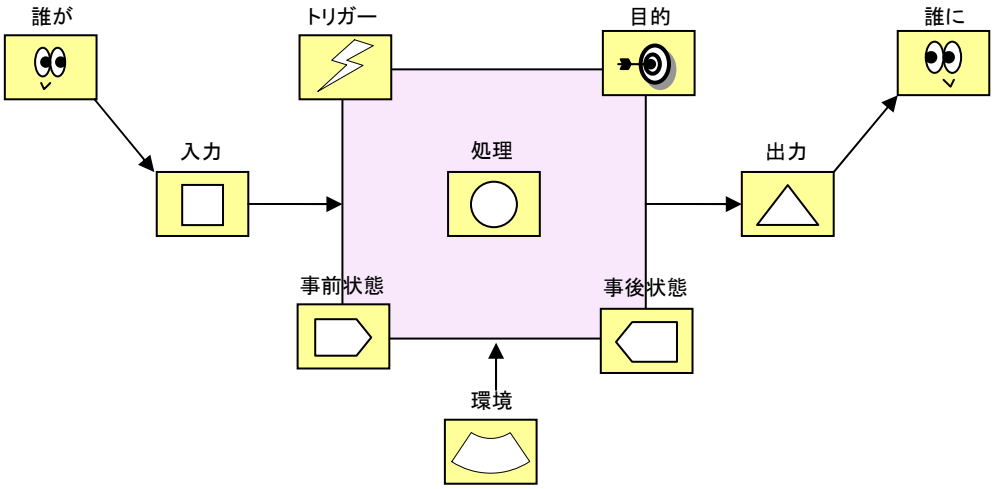


図 6-3　Tiramis で使用している基本構造構成要素

図6-3の要素を用いて基本構造を追加していくことによって、「クラス機能検証」「プログラム機能検証」「バッチ単体テスト」「バッチ処理確認」「バッチ出力項目確認」、「ジョブ機能確認」「画面機能テスト」「画面入力チェック」「画面属性確認」「画面操作試験」「画面レイアウト確認」「帳票表示仕様確認」「システム動作確認」「システム機能テスト」「業務手順確認」「実環境テスト」「本番環境テスト」「疑似本番環境稼動確認」「照会機能確認」「オペレータ操作テスト」「ユーザ操作テスト」「チェック制約動作確認テスト」「オペレーション手続きテスト」「可用性確認」「セキュリティ機能テスト」といった具体的なテストセットを見つけ、その網羅性を高めることができます。

(2)　派生構造

基本構造にテスト観点2（形容詞・副詞）を追加することで、システムの使用条件を考慮したテスト構造を構築することができます。あるものはいじわる条件であり、別のものは顧客のきわどい操作をシミュレートするテストの構造となります。例えば、基本構造の【入力網羅テスト】に対してテスト観点2（形容詞・副詞）を追加することで、

｛テスト対象｝に
（目的語に対する）｛テスト観点2｝の（目的語）を｛テスト観点1｝させる

といったテスト構造を考えることができます。

表 6-3にテスト観点2の例を示します。

表 6-3　テスト観点 2 の例

強度	強い	弱い		**時期**	早く	遅く	
数量	多い	少ない		**時間**	長く	短く	
	増加	減少		**間隔**	広く	狭く	
	余分	不足		**期間**	長く	短く	
	ある	ない	ゼロ	**順序**	先に	後で	
種類	多種	小種			同時に	並行に	
規模	大きい	小さい			早く	遅く	
距離	遠い	近い			逆転	反復	挿入
速度	速い	遅い		**タイミング**	早く	遅く	
	高速	低速		**拡張**	広く	狭く	
	急	緩			多く	少なく	
設定	許容	禁止		**程度**	いつも	たまに	
	必須	任意		**回数**	多く	少なく	
範囲	長い	短い		**接続**	つなぐ	切り離し	切り替え
	上限	下限		**方向**	上へ	下へ	
	広い	狭い		**負荷**	高い	低い	
	最大限	最小限			超過	不足	
頻度	高い	低い		**位置**	高く	低く	
	多い	少ない			遠く	近く	
金額	大きい	小さい					

これらの要素を基本構造や派生構造に追加することによって、「大容量テスト」「ゼロ件テスト」「最大データ性能検証」「実環境負荷確認」「高負荷状態稼動確認」「過負荷状態確認」「最大同時利用者テスト」「高頻度テスト」「バッチ性能テスト」「画面レスポンス測定」「ターンアラウンドタイム要件検証」「長時間運用テスト」「システム機能異常テスト」「業務フロー通常日回しテスト」「通常処理フロー確認」「特殊処理フロー確認」「定常業務運用サイクル確認」「正常系業務フローテスト」といった具体的なテストタイプを見つけ、その網羅性を高めることができます。

なお、今回の分析でテスト観点2（形容詞・副詞）には、表 6-3よりも抽象度の高い、主に反対語を中心としたガイドワードというものがあることが分かりました。表 6-4のようなガイドワードを、標準的なテスト観点2（形容詞・副詞）と組み合わせて使用することで、さらに詳細なテスト観点2（形容詞・副詞）を作成し派生構造をテストすることが可能となります。

表 6-4　テスト観点 2 のガイドワード

	原則	例外			正常	異常	準正常
	全体	部分			通常	非通常	
	全部	一部			通常	例外	代替
	主	従			定期	臨時	
	正	誤			通例	異例	常例
	無事	有事			平時	非常時	緊急時
	公正	不正			定常	可変	
	任意	強制			尋常	非常	
	基本	詳細			一般	特別	
	完了	未完			普通	特殊	
	有効	無効			並	特異	
	起動	停止					

(3)　組み合わせ構造

テスト観点3（関係・組み合わせ）を用いることで、複数の基本構造や派生構造を組み合わせることができます。例えば、基本構造の【入力網羅テスト】に対してテスト観点3（関係・組み合わせ）を追加することで、

｛テスト対象｝に（目的語）を｛テスト観点1｝させることと
｛テスト対象｝に（目的語）を｛テスト観点1'｝させることを
｛テスト観点3｝の組み合わせ方法で実施する

といったテスト構造を考えることができます。テスト観点3（関係・組み合わせ）を用いることで、「モジュール連動テスト」、「モジュール連携テスト」、「ジョブネット連携確認」、「多重実行時排他制御検証」、「運用機能連結テスト」といったテストタイプの網羅性を高めることができます。

(4)　期待結果構造

基本構造、派生構造、そして基本構造・派生構造の組み合わせ構造のそれぞれに対して、テスト観点4（期待結果）の期待結果を考えることができます。

テスト設計の場合、期待結果を考えるのは期待結果を網羅するためであり、想定する期待結果を導出できるように基本構造や派生構造、組み合わせ構造を考えるためです。

例えば基本構造の【入力網羅テスト】に対して、テスト観点4（期待結果）を追加することで、次のテスト構造を考えることができます。

｛テスト対象｝に（目的語）を｛テスト観点1｝させると｛テスト観点4｝となる

このときに、テスト観点4（期待結果）を網羅させるようにテストを設計していくようにするのです。代表的なテストセットには、「エラーコードのテスト」といったものが挙げられます。

以上の4つのテスト構造をまとめると、表 6-5のようになります。

表 6-5　テスト観点とテストアーキテクチャパターン

テスト観点	構造名	テストアーキテクチャパターン(テスト構成方法)
テスト観点 1	基本構造	「～する」という動詞で機能を構成
テスト観点 2	派生構造	基本構造に形容詞、副詞を付けて派生構造を構成
テスト観点 3	組み合わせ構造	複数の基本構造や派生構造を組み合わせて構成
テスト観点 4	期待結果構造	テストの期待結果を網羅させる

6.3 高度化技法

6.2節では、テスト観点とテストアーキテクチャ設計について説明しました。これらを使用することで､テスト全体を網羅的に実施する枠組みを設計することが可能となります。

6.2節の方法を用いて作成したテスト構造には、単純でそのままテストケースに変換できるものもあれば、テスト技法を使用してテスト条件を明らかにした上で、テストケースを実装する必要があるものもあります。

第3章のテスト技法ポジショニングマップで示したように、テスト技法には網羅的な技法、ピンポイントな技法、そしてランダムな技法があります。網羅的な技法とは、何らかのテストモデルを仮定し、そのモデルに従って網羅性をチェックする技法です。例えば、「ソフトウェアはステートメントの集まりである」といったテストモデルを採用し､ステートメントの実施率をテスト網羅性と見る方法です。ピンポイントな技法とは、リスクが高い部分に対してピンポイントにテストを実施するための方法です。

本節では、まず網羅的な技法の代表として、「直交表を活用した網羅的な組み合わせテスト」を説明します。続いてピンポイント的な技法の代表である「シナリオを用いた効果的なピンポイントテスト」について説明します。

6.3.1　直交表を活用した網羅的な組み合わせテスト

(1)　直交表による組み合わせテストの概要

「直交表を活用した網羅的な組み合わせテスト」は、ブラックボックステスト技法の一つです。本技法の適用目的は、デシジョンテーブル（原因結果グラフ、CFD技法を含む）と異なり､入力条件の組み合わせに仕様上は論理関係が特にないケースで全体の組み合わせを網羅的に確認することです。

この技法は､始めにソフトウェアに与えられる入力の種類を因子としてリストアップします（例：用紙サイズ、用紙方向）。次に、各々の因子に対してその選択肢を同値分割・境界値分析などを使用して決定します（例：用紙サイズという因子に対する水準は、A3、A4、B4、葉書など）。そして、選定した因子・水準に合わせた直交表を選択し割り付けを実施します。

例えば、電車の切符を発券するソフトウェアを考えてみましょう。乗車駅、下車駅、大人・子供、枚数、表示言語などが因子にあたります。特急指定券の発券システムを考えるとさらに、座席の位置、禁煙席の希望、席のグレード、往復券とするかといった因子が考えられます。それぞれの因子に対して水準を決定すると、表 6-6の通りになります。

表 6-6　因子・水準

因子	水準
乗車駅	A､B､C､D
下車駅	近県､遠方
種別	大人､子供
枚数	1 枚､3 枚
表示言語	日本語､英語

次にこれらの因子・水準を直交表に割り付けると、表 6-7となります。

表 6-7　直交表に割り付けた結果

No.	乗車駅	下車駅	種別	枚数	表示言語
1	A	近県	大人	1 枚	日本語
2	A	遠方	子供	3 枚	英語
3	B	近県	大人	3 枚	英語
4	B	遠方	子供	1 枚	日本語
5	C	近県	子供	1 枚	英語
6	C	遠方	大人	3 枚	日本語
7	D	近県	子供	3 枚	日本語
8	D	遠方	大人	1 枚	英語

この場合、テストケースは8つとなります。乗車駅のそれぞれに対して、他の因子・水準の組み合わせがすべて表れていることに注目してください。また、下車駅、種別、枚数、表示言語については、任意に取り出した3つの因子の水準組み合わせがすべて出現しています。

今回の例では、任意の2因子間の総組み合わせ数が56個、3因子間の総組み合わせ数が128個でした。そして直交表を使用した組み合わせテストでは、8回のテストで2因子間の組み合わせ網羅率は100%(56/56)、3因子間の組み合わせ網羅率は62.5%(80/128)となっています。

(2)　直交表による組み合わせテストの意義

それではここで、直交表を用いてソフトウェアテストを実施することの意義について考えてみましょう。

直交表を用いた組み合わせテストでは、因子・水準を徹底的に考えることになります。因子・水準を徹底的に考えるためには、局所的な機能に着目するのではなくテスト対象となっているシステムを利用するユーザを含めて全体的に考える必要があります。

ある機能をテストするときに、従来は仕様書に書いてある内容をテストケースに書き写し、その動作結果を判断するだけでした。ところが直交表を用いたテストでは、因子・水準を発見するために、まずその機能の入力にはどのようなものがあるかをリストアップし、次にその機能が動作する環境や使用場面についての要因を考え、最後にそれら機能を乱す可能性のある要因について、FMEA的な分析を行う必要があります。

このとき入力にあたるものを信号因子、環境や使用場面等の機能を乱す要因を分析して抽出したものを、誤差因子（またはノイズ）と呼びます。先の電車の切符を発券するソフトウェアの例においては、「乗車駅、下車駅、大人・子供、枚数」が信号因子、「表示言語」が誤差因子となります。実際のテストにおいては、「ボタンを押す順番」や「一度押した内容の変更」といった要因も、誤差因子として挙げていきます。

また直交表による組み合わせテストでは、組み合わせた後に「任意の2つの因子を取り出したときの水準の組み合わせ」についての網羅性を計算することができます。これを「強度2の組み合わせ網羅率」と呼びます。

表 6-8　欠陥を顕在化させるために必要となる因子数と網羅率（%）の関係

FTFI	NASAエキスパートシステム RAX(convergence, correctness, interf, engine)				OS POSIX modules	組込み 医療機器 ﾘｺｰﾙﾃﾞｰﾀ	Browser Mozilla	Server Apache	DB NASA GSFC
1	61	72	48	39	82	66	29	42	68
2	97	82	54	47	*	97	76	70	93
3	*	*	*	*	*	99	95	89	98
4	*	*	*	*	*	100	97	96	100
5	*	*	*	*	*		99	96	
6	*	*	*	*	*		100	100	

表 6-8は、「IEEE Transactions on software engineering, Vol.30, 2004.」のデータを加工したものです。FTFIとは”failure-triggering fault interaction”の略で、欠陥を顕在化させるために必要となる因子の数です。通常のソフトウェアにおいては、2因子までの組み合わせ問題がほとんどで、3因子以上の組み合わせ問題はわずかであることが分かります。オープンソースのブラウザやサーバのように、世界中の多くの人が関わっているものについては4因子以上の組み合わせ問題も発生していますが、そちらは設計の改善で対応すべき問題でありソフトウェアテストの課題ではないと考えます。

直交表を使用して組み合わせテストを作った場合、特に禁則などがなければ強度2の組み合わせ網羅率は100%となります。強度については、2に限らずいくつでも定義することはできますが、上記IEEEのデータから分かる通りソフトウェアテストにおいては強度2を使用することで、2因子間の欠陥を取りきることが効果的と考えます。特に強度4を超えると、組み合わせテスト数が膨大なものになるため通常は使用されません。

これら直交表によるテストの特徴を合わせると、因子・水準を分析することによって得られたテスト全体に対して、強度の基準でどこまで網羅しているかを提示することができるようになります。従来のテストにおいては、まず何をテスト全体と定義したかが不明確でした。従って、どのくらいテストがプログラムコードをカバーしたかについては、C0, C1等のカバレッジ指標で確認できたとしても、どのくらいの機能をカバーしたのかについては不明でした。直交表はそれに対する解の1つとなります。

もちろん、因子・水準を分析してもその分析方法が不足していればテスト全体も不十分な定義になります。しかし仮にそうであったとしても、市場導入後に出た欠陥を因子・水準にフィードバックすることで、他の因子との組み合わせを含めてテストの改善を図ることができます。従来も市場で発生した欠陥をテストケースに追加し、再発防止を実施してきている組織は多いのですが、それは単機能における再発防止ができただけで、組み合わせについての再発防止にはなっていません。

(3)　直交表とトレーサビリティ

前項で述べた通り、直交表を用いた組み合わせテストでは、因子・水準の選択がテストの品質を決定する鍵となります。一般的にいうと、「組み合わせアイテム分析」を実施する必要があります。このときに大切となるのは、要求、要件、仕様、設計、テスト、欠陥の関連性を明らかにすることです。第5章で述べたトレーサビリティの確保が重要となります。

品質機能展開表をトレーサビリティの確保に使用し、本当に組み合わせる必要がある因子を要求や設計に踏み込んで吟味することで組み合わせの爆発を防ぐことができ、効率的なテストを実現することができるようになるでしょう。

(4)　エンタプライズ系企業における直交表の活用

これまで直交表は、どちらかというと組込み系企業において使用されてきました。それはハードウェアの世界では、実験計画法を使用することが当たり前であり、ハードウェアと密接な関係を持つ組込みソフトウェアにおいて、その延長線上で直交表が使用されてきたからです。ところがエンタプライズ企業においては、トライアルで使用されることはあってもテストタイプとして確立しているものは、今回の調査ではありませんでした。

エンタプライズ系ソフトウェアに対して直交表を適用するためのボトルネックは、扱う機能の多さとその組み合わせの複雑さです。従来存在する直交表で最大のL_{64}を使用しても、63個の因子（すべて2水準の因子）の組み合わせしかテストできません。また複雑なビジネスルールがあるため、単純に直交表に因子を割り付けただけでは、あり得ない組み合わせが多発しテストとして成立しなくなってしまうからです。

2005年に、上記問題点を「多因子・多水準の生成」と「禁則の回避」で解決したHAYST法（第2部3章『富士ゼロックス株式会社の事例』を参照）が生まれました。HAYST法も直交表と同様に、主に組込み系企業のソフトウェアテスト技法として広まっていますが、エンタプライズ系の給与システム、保険システム、ドキュメントマネジメントシステムにおいて、部分的に適用が始まり効果が出始めています。これらのシステムでは、因子数が200を超え複雑な禁則を持つものもあります。

(5)　直交表の適用例

あるシンポジウムのウェブによる参加登録画面から因子・水準を抜き出すと、表 6-9の通りになりました。これらの因子には、次に述べる禁則を除き論理的な関係は仕様上ないとされています。直交表によるテストは、このような仕様上は論理的な関係がないとされている因子を組み合わせて、仕様書や設計から検知できない組み合わせの欠陥を効率良く検出するための技法です。

表 6-9　FL 表（因子・水準表）

因子	水準1	水準2	水準3	水準4	水準5	水準6	水準7	水準8
名前	山田太郎	佐藤次郎						
所属	株式会社ABC	ABC大学						
役職	一般	管理者	役員	その他				
チケット種別	一日券	一日券＋チュートリアル						
情報交換会への参加	しない	する						
お弁当の手配	しない	する						
セッションAの選択	しない	する						
セッションBの選択	しない	する						
セッションCの選択	しない	する						
セッションDの選択	しない	する						
チュートリアルの選択	しない	する						
支払い方法	クレジットカード	銀行振込						
開催情報取得先	ML	ウェブ	メールニュース	雑誌	学会	上司	社内の案内	その他
業種	ソフトハウス	メーカー	教育機関	その他				
職種	開発者	品質関係	コンサルタント	その他				
社員数	1万人以上	1千人～1万人未満	100人～1千人未満	100人未満				

ここでチケット種別が一日券の場合は、チュートリアルが選択できないという禁則があるとします。表 6-10は、「一日券×する」の組み合わせができないことを表しています。

表 6-10　禁則表

		チュートリアルの選択	
		しない	する
チケット種別	一日券		×
	一日券＋チュートリアル		

上記、因子・水準について禁則を回避しながら直交表に割り付けを実施すると、後述の表 6-11となります。

このときテスト項目数は32項目で、網羅率は2因子間網羅率100%(971/971)、3因子間網羅率76.45%(9,564/12,510)となっています。

表 6-11　直交表による組み合わせテストマトリクス

No.	名前	所属	役職	チケット種別	情報交換会への参加	お弁当の手配	セッションAの選択	セッションBの選択	セッションCの選択	セッションDの選択	チュートリアルの選択	支払い方法	開催情報取得先	業種	職種	社員数
1	山田太郎	株式会社ABC	一般	一日券	しない	しない	しない	しない	しない	しない	しない	クレジットカード	ML	ソフトハウス	開発者	1万人以上
2	山田太郎	ABC大学	管理者	一日券＋チュートリアル	する	しない	する	する	しない	する	しない	銀行振込	ML	メーカー	品質関係	1千人～1万人未満
3	佐藤次郎	株式会社ABC	役員	一日券＋チュートリアル	する	する	しない	する	する	しない	する	銀行振込	ML	教育機関	コンサルタント	100人～1千人未満
4	佐藤次郎	ABC大学	その他	一日券	しない	する	する	しない	する	する	しない	クレジットカード	ML	その他	その他	100人未満
5	佐藤次郎	ABC大学	一般	一日券	しない	する	する	しない	する	しない	しない	銀行振込	ウェブ	メーカー	品質関係	100人～1千人未満
6	佐藤次郎	株式会社ABC	管理者	一日券＋チュートリアル	する	する	しない	する	する	する	しない	クレジットカード	ウェブ	ソフトハウス	開発者	100人未満
7	山田太郎	ABC大学	役員	一日券＋チュートリアル	する	しない	する	する	しない	しない	する	クレジットカード	ウェブ	その他	その他	1万人以上
8	山田太郎	株式会社ABC	その他	一日券	しない	しない	しない	しない	しない	する	しない	銀行振込	ウェブ	教育機関	コンサルタント	1千人～1万人未満
9	佐藤次郎	ABC大学	一般	一日券＋チュートリアル	しない	する	しない	する	しない	する	しない	銀行振込	メールニュース	教育機関	その他	1万人以上
10	佐藤次郎	株式会社ABC	管理者	一日券	する	する	する	しない	しない	しない	しない	クレジットカード	メールニュース	その他	コンサルタント	1千人～1万人未満
11	山田太郎	ABC大学	役員	一日券	する	しない	しない	しない	する	する	しない	クレジットカード	メールニュース	ソフトハウス	品質関係	100人～1千人未満
12	山田太郎	株式会社ABC	その他	一日券＋チュートリアル	しない	しない	する	する	する	しない	する	銀行振込	メールニュース	メーカー	開発者	100人未満
13	山田太郎	株式会社ABC	一般	一日券＋チュートリアル	しない	しない	する	する	する	する	しない	クレジットカード	雑誌	その他	コンサルタント	100人～1千人未満
14	山田太郎	ABC大学	管理者	一日券	する	しない	しない	しない	する	しない	しない	銀行振込	雑誌	教育機関	その他	100人未満
15	佐藤次郎	株式会社ABC	役員	一日券	する	する	する	しない	しない	する	しない	銀行振込	雑誌	メーカー	開発者	1万人以上
16	佐藤次郎	ABC大学	その他	一日券＋チュートリアル	しない	する	しない	する	しない	しない	する	クレジットカード	雑誌	ソフトハウス	品質関係	1千人～1万人未満
17	佐藤次郎	株式会社ABC	一般	一日券	する	しない	する	する	する	しない	しない	銀行振込	学会	ソフトハウス	その他	1千人～1万人未満
18	佐藤次郎	ABC大学	管理者	一日券＋チュートリアル	しない	しない	しない	しない	する	する	する	クレジットカード	学会	メーカー	コンサルタント	1万人以上
19	山田太郎	株式会社ABC	役員	一日券＋チュートリアル	しない	する	する	しない	しない	しない	しない	クレジットカード	学会	教育機関	品質関係	100人未満
20	山田太郎	ABC大学	その他	一日券	する	する	しない	する	しない	する	しない	銀行振込	学会	その他	開発者	100人～1千人未満
21	山田太郎	ABC大学	一般	一日券	する	する	しない	する	しない	しない	しない	クレジットカード	上司	メーカー	コンサルタント	100人未満
22	山田太郎	株式会社ABC	管理者	一日券＋チュートリアル	しない	する	する	しない	しない	する	する	銀行振込	上司	ソフトハウス	その他	100人～1千人未満
23	佐藤次郎	ABC大学	役員	一日券＋チュートリアル	しない	しない	しない	しない	する	しない	しない	銀行振込	上司	その他	開発者	1千人～1万人未満
24	佐藤次郎	株式会社ABC	その他	一日券	する	しない	する	する	する	する	しない	クレジットカード	上司	教育機関	品質関係	1万人以上
25	山田太郎	ABC大学	一般	一日券＋チュートリアル	する	する	する	しない	する	する	する	クレジットカード	社内の案内	教育機関	開発者	1千人～1万人未満
26	山田太郎	株式会社ABC	管理者	一日券	しない	する	しない	する	する	しない	しない	銀行振込	社内の案内	その他	品質関係	1万人以上
27	佐藤次郎	ABC大学	役員	一日券	しない	しない	する	する	しない	する	しない	銀行振込	社内の案内	ソフトハウス	コンサルタント	100人未満
28	佐藤次郎	株式会社ABC	その他	一日券＋チュートリアル	する	しない	しない	しない	しない	しない	しない	クレジットカード	社内の案内	メーカー	その他	100人～1千人未満
29	佐藤次郎	株式会社ABC	一般	一日券＋チュートリアル	する	しない	しない	しない	しない	する	する	銀行振込	その他	その他	品質関係	100人未満
30	佐藤次郎	ABC大学	管理者	一日券	しない	しない	する	する	しない	しない	しない	クレジットカード	その他	教育機関	開発者	100人～1千人未満
31	山田太郎	株式会社ABC	役員	一日券	しない	する	しない	する	する	する	しない	クレジットカード	その他	メーカー	その他	1千人～1万人未満
32	山田太郎	ABC大学	その他	一日券＋チュートリアル	する	する	する	しない	する	しない	しない	銀行振込	その他	ソフトハウス	コンサルタント	1万人以上

6.3.2 シナリオを用いた効果的なピンポイントテスト

「シナリオを用いた効果的なピンポイントテスト」は、ユーザ視点のブラックボックステスト技法の1つです。

本技法は、ユースケーステストの一種です。ユースケーステストとシナリオテストの違いは、ユースケーステストがある程度網羅性を重視するのに対し、シナリオテストはよりピンポイントに「重要な業務シナリオが流れること」を検証する点です。

例えば、ウェブベースの商品購買システムのシナリオテストにおける1シナリオは、

1「夜半(When)」に「Aさん(Who)」が「自宅(Where)」から「品物(What)」を検索
2. ショッピングカートに入れる
3. レジに進む
4. サインインする
5. ★ クレジットカードの有効期限が切れている場合の処理
6. 配送条件を確定する
7. 注文を確定する

といったものになります。シナリオ作成時には、いつ(When)、誰が(Who)、どこで(Where)というコンテキスト（使用の文脈）を定義することが大切です。

また、ステップ5にあるような例外シナリオを発見することが大切です。例外シナリオは、基本シナリオに｛テスト観点3｝を当てはめることで見つけることが可能です。

システムが大きくなると、上記のようなシナリオを業務の観点で連結していくことになります。具体的には、業務フロー、業務周期、業務データの観点で複雑なテストシナリオを構築していきます。以下、それらについて説明します。

(1) 業務フローの観点によるシナリオテスト

テスト工程以前に顧客から業務フローをヒアリングし、業務フロー定義書を作成します。しかし、一般に業務フローは業務の代表的な流れを表現したものであり、細かい例外フローを盛り込んでいないことが多いため、ヒアリングから作成した業務フロー定義書はその網羅性が低く、そのままでは漏れが多いためテストには使用できません。

そこで業務フロー定義書を元に、テスト設計でテスト技術者が不足分を補い、それを顧客と合意することで業務フローの観点によるシナリオテストを作成します。

例えば、正常業務フローのテスト設計では順次処理の観点でテストを追加します。また、異常業務フローのテスト設計では、ガイドワードすなわち「進む、戻る、繰り返す、止まる、超える、途切れる、分ける、纏める、無くす、変える」等々を使用して、データの重複や消えるなどの欠陥を検出します。

(2) 業務周期の観点によるシナリオテスト

業務の種類によっては、業務フローで表現するのではなく「毎週の処理」とか「月末の処理」といった業務周期で表現する方が、業務が明確になるケースがあります。従って、業務周期の観点によるシナリオテストの設計を考えることができます。

「週次、曜日、月次、期末、年末」等々、個々の人がサイクルとしてとらえている業務をサイクルごとにリストアップし、それを元に業務フローを運用サイクルに落としていくことでテストシナリオを作成します。

ただしこのとき、特定の年次処理をさせるためのテストデータの作成は、結果系から逆演算を繰り返す必要があるためロジックの整理が必要となります。

なおサイクルによるテストシナリオは、それぞれの周期において人が作業を開始する場合が多いものですが、定期的に内部トリガーが発生し、それによって処理が動くケースがありますので、それらを見落とさないようにテストシナリオに含めることが大切です。

(3) 業務データの観点によるシナリオテスト

一般に、処理フローよりもデータ構造やデータの変化の方が情報量が少ないため、テスト設計しやすいものです。

従って、データの変化に着目したシナリオテストを考えることができます。これをエンティティ・ライフ・ヒストリに基づく、シナリオテストと呼びます。

つまり、顧客の状態や口座の状態に着目したシナリオを考えます。このときデータだけでなく、人・物・金・承認の流れを併せてシナリオに盛り込むことで、より現実に近いシナリオをテスト設計することが可能です。

6.4 エンタプライズ系代表企業十社のテスト実態分析

各社は、様々なテストに名称を付けてプロジェクトを運営しています。「パフォーマンステスト」という大きなテストタイプから、「ゼロ件テスト」のようにボリュームテストの一部を表すテストまで、多くの名称が付いています。テスト技法の名称に大きな違いはありませんが、テストタイプの名称はテスト技法と異なり各社によって異なるケースもあります。

「高信頼化のための手法WG」では、高信頼システムを構築しているエンタプライズ系代表企業十社で、どのようなテストを実施しているかを調査いたしました。各社で用いられている名称のまま載せたのが、各社テスト一覧表（表 6-12）です。テストレベルと開発工程は異なる概念であり、詳細なテストレベルを定義している企業もありますが、表が煩雑になるため共通フレームで定める開発工程のくくりでまとめています。

「セキュリティテスト」のように同じ名称を用いているケースもあれば、「負荷テスト/ストレステスト/ラッシュテスト」のように同じテストでも異なる名称を用いていることが分かります。

このようにテストの名称は様々ですが、各社がテストで狙っているところ、つまりテスト観点は共通するものがありそうです。各社が用いているテスト観点を抜き出して整理することで、高信頼化のためのシステムをテストする際に必要な観点が浮き彫りになります。テスト観点を導き出す思考過程も踏まえてまとめたものがテスト観点表（表 6-13）です。縦軸に品質特性、横軸にハイレベルなテスト観点を配置しています。縦軸と横軸が交差する箇所に、各社が名付けている各種テストが入ります。なお、表のすべてのセルは埋まりません。

表 6-12　各社テスト一覧表

		ユーザ系		
SLCPで定めるテスト工程		A社	B社	C社
1.6.7	ソフトウェアコード作成およびテス	単体テスト	コードカバレッジ確認 画面構成確認 画面操作確認 画面処理内容確認 DB更新確認 出力項目確認 実行時間確認 大量データ性能確認	プログラム機能テスト 機能テスト 異常値テスト ゼロ件テスト 少量データテスト 大量データテスト
1.6.8 1.6.9	ソフトウェア結合 ソフトウェア適合性確認テスト	サブシステム内結合テスト サブシステム間結合テスト	画面遷移処理バリエーション確認 バッチ・プロセス間連携動作確認 業務・サブ業務機能間テスト 統合運用管理システム連携テスト 機能テスト 運用機能連結テスト バッチ・リラン・リカバリ方法確認 性能テスト 特殊な性能テスト 外部システム連携テスト	オンライン画面遷移テスト バッチ入出力確認テスト Webアプリ脆弱性診断
1.6.10 1.6.11	システム結合 システム適格性確認テスト	サイクルテスト 導入性テスト 操作文書テスト 業務処理文書テスト 運用文書テスト 運用テスト プロシジャーテスト リカバリーテスト 性能テスト ストレステスト 大容量テスト ストレージテスト スケーラビリティテスト セキュリティテスト ファシリティテスト	災害対策テスト サービスレベルテスト 性能テスト 負荷テスト 機能テスト セキュリティテスト	JCL稼動確認テスト シナリオテスト プレ本番テスト オーナーテスト 有事全体テスト 有事月例テスト 性能テスト 負荷テスト 機能テスト（条件組合せ） セキュリティテスト 疎通テスト リグレッションテスト
1.6.13	ソフトウェア受入れ支援	導入性テスト 受入テスト 運用テスト		運用テスト

表 6-12　各社テスト一覧表（続き）

		独立系		ベンダー系	
SLCPで定めるテスト工程		D社	E社	F社	G社
1.6.7	ソフトウェアコード作成およびテスト	カバレッジテスト DB単体テスト DB単体テスト 入力データバリエーションテスト 出力データバリエーションテスト エラーテスト 操作確認テスト 入力チェックテスト 入力バリエーションテスト 更新バリエーションテスト 照会バリエーションテスト 検索バリエーションテスト	カバレッジテスト フラグ確認テスト データマトリクステスト データバリエーションテスト DBアクセステスト エラーテスト レイアウト確認テスト 入力チェックテスト 画面機能テスト 単画面性能テスト	機能単体ホワイトボックステスト 機能単体ブラックボックステスト	命令網羅試験 分岐網羅試験 機能試験 画面操作試験
1.6.8 1.6.9	ソフトウェア結合 ソフトウェア適合性確認テスト	画面遷移テスト 排他制御確認テスト 操作組合せテスト JOB正常系テスト JOB異常系テスト 外部システム連携テスト セキュリティテスト リグレッションテスト	画面遷移機能テスト 画面遷移網羅テスト 画面遷移性能テスト JOB正常系テスト JOB異常系テスト サブシステム内結合テスト サブシステム間結合テスト	機能連携テスト サブシステム連携テスト	並行／同時処理試験 誤操作試験 プログラム間I／F試験 機能間機能試験 機能ブロック間機能試験 機能試験 障害回復試験 障害発生時稼動確認 エラーメッセージ動作確認 平均性能試験 限界性能試験 容量試験 インストール試験
1.6.10 1.6.11	システム結合 システム適格性確認テスト	通常業務シナリオテスト 例外業務シナリオテスト 本番データテスト データ移行リハーサル リカバリーテスト パフォーマンス測定 資源測定 セキュリティテスト	運用サイクルテスト 主要業務シナリオテスト 詳細業務シナリオテスト 例外業務シナリオテスト 本番データテスト 移行データ確認テスト 移行シナリオテスト 操作マニュアルテスト フィールドテスト 運用手順テスト 運用監視テスト 障害発生テスト 障害対応テスト バックアップテスト リカバリーテスト 性能要件テスト 高負荷／ラッシュテスト スループット検証テスト 高頻度テスト ロングラインテスト サイジングテスト スケーラビリティテスト ボリュームテスト リソース不足テスト ユーザビリティテスト セキュリティテスト 疎通確認テスト 外部接続テスト デグレード確認テスト 保守マニュアルテスト 構成テスト データ互換性テスト	業務運用テスト 導入テスト インフラ運用テスト リカバリーテスト パフォーマンステスト 負荷テスト ストレステスト ロングランテスト スケーラビリティテスト ボリュームテスト 機能性テスト ユーザビリティテスト セキュリティテスト 構成テスト	運用繰り返し試験 業務フロー機能試験 システム起動停止試験 休日運転試験 障害発生時稼動確認試験 縮退運転試験 エラーメッセージ対応 動作確認試験 障害回復試験 耐久試験 連続運転安定性試験 他システムI／F試験 並行／同時処理試験
1.6.13	ソフトウェア受入れ支援	運用テスト			運用試験

表 6-12　各社テスト一覧表（続き）

		ベンダー系		
SLCPで定めるテスト工程		H社	I社	J社
1.6.7	ソフトウェアコード作成およびテスト	部品テスト モジュールテスト 表示仕様確認 入力チェック確認 基本機能確認 出力項目確認	データアクセス層単体テスト ビジネスロジック層単体テスト プレゼンテーション層単体テスト バッチ単体テスト	JOBSTEPテスト JOB機能テスト 画面機能テスト 性能テスト 標準準拠確認テスト
1.6.8 1.6.9	ソフトウェア結合 ソフトウェア適合性確認テスト	サブシステム内結合テスト サブシステム間結合テスト 業務機能テスト システム機能異常テスト 性能テスト 限界テスト	モジュール連動テスト JOBフローテスト ユースケースシナリオテスト 性能テスト 接続テスト セキュリティテスト 業務フローテスト 原新確認テスト	チーム内結合テスト チーム間結合テスト 排他テスト 方式設計確認テスト 画面負荷テスト ボリュームデータテスト 外部インタフェーステスト
1.6.10 1.6.11	システム結合 システム適格性確認テスト	業務サイクルテスト 業務機能テスト 特殊処理フロー確認 可用性テスト システムリカバリテスト 性能測定 システム負荷テスト 使用性テスト セキュリティテスト 保守性テスト	業務系ジョブネット連動テスト 運用系ジョブネット連動テスト 業務－運用ジョブネット連動テスト 日回しテスト 移行プログラム連動テスト エンドユーザ参加テスト コンティンジェンシー観点テスト 運用確認テスト 運用確認（異例運用） 運用確認（監視端末） 業務系ジョブネット障害テスト 障害時リラン 信頼性テスト 性能テスト 性能テスト（高負荷） セキュリティテスト 外部センタ接続テスト システム切替 並行運用	運用サイクルテスト 正常系業務テスト 異常系業務テスト 非通常運用テスト バッチ性能テスト 画面負荷テスト 長時間運用テスト ボリュームデータテスト 保守テスト
1.6.13	ソフトウェア受入れ支援	運用テスト	運用テスト	

表 6-13　テスト観点表

		開発					移行		業務		
			機能				移行・導入	受け入れ・本番	業務	保守	運用
			データ	状態	操作	コード					
機能性	機能性										
	相互運用性										
	セキュリティ										
信頼性	障害許容性										
	回復性										
使用性	使用性										
効率性	時間効率性										
	資源効率性										
保守性	変更性										
移植性	共存性										

6.5 テスト実態分析結果の使い方

6.5.1 各社テスト一覧表の使い方

(1) 自社テスト標準への反映

テスト標準を作成する担当の方の場合、各社が実施しているテストと比較し自社の標準へ反映することができます。過去に実施してきたテストと似た傾向を持つ企業のスタイルを取り込むこともできますし、複数社のテスト標準から抜き出しして（よいところ取りをして）標準に反映することもできます。

自社のテスト標準へ反映する際の注意点は2つあります。1つは開発工程とテストタイプの関係で、もう1つはテストタイプの詳細度です。

どの工程にどんなテストタイプのテストを実施するかは、各社によって異なっています。例えばF社は、1.6.9. ソフトウェア適合性確認テストにおいて「機能連携テスト」「サブシステム連携テスト」を行い、1.6.11 システム適格性確認テストにおいて「パフォーマンステスト」「リカバリーテスト」などを行っています。一方、G社はソフトウェア適合性確認テストにおいて「機能間機能試験」だけでなく、「障害回復試験」「平均性能試験」などを行っています。回復性や時間効率性などの非機能に関するテストをどの工程で行うかは各社によって判断が異なっています。テスト標準に反映するにあたり、テストタイプをどの工程で実施するかは、自社の開発スタイルに合わせて決める必要があります。

テストタイプの詳細度も異なります。C社が「シナリオテスト」としているところを、D社は「通常業務シナリオテスト」「例外業務シナリオテスト」と分けています。さらに、E社は「通常業務シナリオテスト」を2つに分け、「主要業務シナリオテスト」と「詳細業務シナリオテスト」に分けています。このように、テストタイプの詳細度は異なります。テスト標準に反映するとき、詳細度が低すぎるとシステムプロファイルのレベルに関係無く事細かなテストを実施しなくてはいけなくなり使い勝手の悪い標準になります。詳細度が高すぎると何をテストすればよいのか判断に迷うため、標準で期待するテストをプロジェクトで実施されない可能性もあります。ほどよい粒度に設定する必要があります。

(2) プロジェクトへの反映

プロジェクトにかかわる方の場合、この表を使うことで発注者/受注者がどのスタイルのテストを望んでいるのかが分かるようになります。各社によって、工程ごとに実施するテストが異なっていますし、名称も異なっています。一覧表を見ることにより工程や用語の摺り合わせができるようになり、合意形成が早くできるようになります。

例えば、1.6.11.システム適格性確認テストにおいて、A社には「サイクルテスト」、E社には「運用サイクルテスト」、H社には「業務サイクルテスト」というテストタイプがあります。同じ「サイクルテスト」という名前が付いているので、同じテストのように思われますが、その内容を掘り下げてみると違う内容のテストを指していることもあります。

また、G社の「耐久試験」とJ社の「長時間運用テスト」は異なる名称を用いていますが、テスト目的は同じだと思われます。しかし、確認しなければ本当のところは分かりません。

これはユーザ系、独立系、ベンダ系の違いによるものではありません。そのため、発注者/受注者がこの表を用いてそれぞれのテストタイプを摺り合わせることが合意形成の早道となるでしょう。

6.5.2 テスト観点表の使い方

(1) 自社テスト標準への反映

テスト標準を作成する担当の方の場合、自社のテスト標準を根本的に見直すときに使うことができます。自社の標準をテスト観点表に割り付けて、実施しているところと実施していないところを明らかにします。環境の変化に対応して削るテスト、残すテスト、加えるテストを判断し、テスト標準を見直します。

各社テスト一覧表は、各社のテストタイプを開発工程に割り付けたものです。そのため、高信頼化を実現するために十分なテストが実施されているかどうか分かり難くなっています。また、既存のテスト標準が企業環境の変化やテクノロジーの変化に順応しているかどうかも判断しづらくなっています。

そこで、自社のテスト標準をテスト観点表に割り付け、現時点で自社標準とされているテスト観点がどの領域を中心にテストをしているのかを明らかにします。

例えば、A社のテストタイプを分類すると、表 6-14のようになります。

表 6-14 A社のテストタイプ分類

		開発					移行		業務		
			機能				移行・導入	受け入れ・本番	業務	保守	運用
			データ	状態	操作	コード					
機能性	機能性	2			2	1	1	1	1		2
	相互運用性										
	セキュリティ	1									
信頼性	障害許容性										
	回復性										1
使用性	使用性										
効率性	時間効率性	2	1								
	資源効率性		1								1
保守性	変更性										
移植性	共存性										

空白になっている箇所がありますが、これはテストを実施していないということではありません。テストタイプ、つまり、実施しているテストに名前がないために表に挙がっていないのです。

さて、システムを取り巻く環境の変化に対応すべく、このテスト標準を変更すると仮定します。このとき大切なことはすべての枡目を埋めることではありません。必要なテスト観点だけを取り込むことです。すべてを網羅しようとすると、一番重点を置かなければならない肝心なテストをおざなりにしてしまうからです。

A社の例で言えば、今後、性能拡張性を重視してスケーラビリティテストなどを含めるのか、障害許容性を重視して縮退運転テストなどを含めるのかなどを検討するかもしれませんし、使用性を重視するかもしれません。

このように枡目を埋めるという考えではなく今後必要になるテストは何だろうと考えなくてはなりません。テスト標準を作ろうとすると、網羅性を重視したものを作りかねませんので注意が必要です。

(2) プロジェクトへの反映

プロジェクトにかかわる方の場合、テスト戦略を立てるときに使うことができます。テスト観点表のすべてのセルを網羅することはせずに、システムプロファイルに合わせて使用します。すべての品質特性をテストで検証するのかどうかの判断をし、プロジェクト特性やシステム特性に合わせて品質特性を絞り込みます。ハイレベルテスト観点も同様に絞り込みます。

例えば、システムによっては、使用性や共存性のテストは重視しないということもあり得ます。ハイレベルテスト観点では、機能、データ、コードだけの観点を重視し、状態と操作の観点を除くということもあり得ます。このような判断を行うことで表のサイズを小さくし、テスト観点を減らします（表6-15）。

表 6-15 プロジェクトごとのテスト観点表の例

		開発			移行		業務		
		機能			移行・導入	受け入れ・本番	業務	保守	運用
			データ	コード					
機能性	機能性								
	相互運用性								
	セキュリティ								
信頼性	障害許容性								
	回復性								
効率性	時間効率性								
	資源効率性								
保守性	変更性								

次にテストタイプを決めていきます。ここで注意して欲しいことは、すべての枡目を埋めることを重視するのではないということです。必要なテストタイプだけを選びます。

この後は、6.2節で述べているテストアーキテクチャ設計を行います。

第7章　高信頼ソフトウェア開発に関する海外事例および研究動向

7.1　はじめに

本章では高信頼ソフトウェア高信頼にかかわる海外動向を紹介します。

（独）情報処理推進機構は、2008 年度に「情報システムの信頼性評価手法の調査」として米国および欧州の動向の調査を実施しました。この調査では、海外の調査機関や国内外の有識者の支援を得て、文献および Web 調査により信頼性確保および信頼性評価に関する公知の情報を収集・整理し、そのうちいくつかについてヒアリング調査を実施しました。ここではまず、この調査報告において取り上げられている欧米における情報システムの信頼性確保の取り組み状況を紹介します。

また、アカデミアからの視点による補足として、IEEE の著名な国際会議の１つであり、2009 年において第 31 回の開催を迎えた ICSE（International Conference on Software Engineering）を取り上げ、厳しい審査を通過した理論ベースの研究論文、ならびにソフトウェア開発の実務従事者、もしくはそれに近い著者が主体となり執筆した実証研究論文が取り扱った主題について、過去 3 回分（2007～2009）を対象としてこの章の最後に示します。

7.2　欧米における情報システムの信頼性確保の取り組み

7.2.1　調査内容

ここでは、欧州および米国の各 20 機関の情報システムの信頼性確保の取り組みについて文献等で調査を実施し、その結果から表 7-1 に示す通り、各 5 機関に絞り込んだ上でヒアリング調査を実施しました。調査は、開発ライフサイクル、技術、人・組織、商慣行・契約・法的要素に関する各観点について次の項目に絞って実施しました。

表 7-1　ヒアリング調査対象機関

米国		欧州	
分類	機関名	分類	機関名
官	サンディア研究所	官	エネルギー技術研究所（IEF、ノルウェー）
産	AT&T 研究所	官	NTNU（ノルウェー）
官	食品医薬品局	学	ニューキャッスル大学 ソフトウェア信頼性センタ
産	メタベラ	産	Fraunhofer Institute
産	アンベリオン	産	INRIA

(1) 先進的な研究・技術への取り組み状況
- 形式手法(Formal Method)
- ソフトウェア部品再利用：ソフトウェアプロダクトライン(SPL)
- モデルベース開発：モデル駆動型アーキテクチャ（MDA）
- テスト駆動型開発（Test Driven Development（TDD））
- 自動コード生成

(2) 信頼性を確保するための開発/調達要件
- CMMI や PMP などの開発体制・人材に関する認定制度を利用しているか
- ISO/IEC などの開発プロセスに関する標準を導入して（義務付けて）いるか
- 信頼性に関するどのような SLA を設定しているか

・ 開発過程において欠陥を管理し、これを減らすための手法、ツールが整備されているか
・ 高い品質を保つための試験工程における体制・手法が整備されているか

(3) 上記要件の社内ルールへの組込み状況
・ 開発手法・プロセスに関する社内標準を有しているか
・ 信頼性をチェックするための PMO や品質保証部門を設定しているか

(4) 条件としている国際基準等の有無とその内容
・ 民間や業界の標準
・ 政府機関の標準

7.2.2 調査結果

欧米事例を基に分析した結果を以下にまとめます。

(1) 情報システムの信頼性向上に向けた取り組み

(1-1) 米国

◇政府・政府系機関

食品医薬品局（FDA）の取り組みについてヒアリングを実施しました。医療分野で利用される機器、デバイスの種類は多種多様ですが、人の生命に直接影響を与えることから、ソフトウェア等に対して一定の信頼性を保つことは極めて重要です。FDA 自体はソフトウェアの開発は行わないものの､所管官庁の立場から医療分野のシステム、機器、デバイス等の承認を行っています。具体的には、1996 年の品質システム規則によって、上記のような機器等に対してはソフトウェアの承認を求めており、FDA の研究施設において、医療機器のソフトウェアのテストを行っています。

上記のソフトウェアの承認に関しては、「ソフトウェア検証に関する一般原則」において詳細に規定されており、このガイドラインは FDA の Web サイト上に公開されています。このガイドラインでは具体的な開発方法論を規定することはなく、プランニング、要件、設計、テスト、構築といった開発ライフサイクルのフェーズにおけるドキュメントを提出することを義務化する、といったアプローチを採っています。

また、政府向けのシステム構築などを行う民間のコントラクタは、個々の省庁が規定する独自の標準に従うケースが見られます。例えばソフトウェア開発標準として、国防総省（DoD）の J-STD-016、エネルギー省（DoE）の DOE-SSTD-1150 や DOE-STD-1172 などです。

◇民間

情報システムの信頼性向上には、ソフトウェア言語や開発手法等の技術的な要素に加え、ソフトウェア開発におけるプロジェクトマネジメント、さらには開発や運用を行うエンジニア等のスキルなど様々な要素が影響を及ぼします。また、情報システム自体も多種多様であることから、すべてのシステムに適用できるような信頼性向上の理想的な解決法はなく、むしろ従来からの方法論をベースに各社の業務や顧客のニーズにあったアプローチを採っているようです。

情報システムの信頼性向上にあたっては、その要素であるソフトウェアの品質が重要と考えられますが、一方でシステムの開発者や運用者にソフトウェアとハードウェアの両方の知識や経験が必要とされる場合もあります。従来、ハードウェアの機能であった部分がソフトウェアで構築される傾向が近年見られ、その点ではソフトウェアがより複雑化し、信頼性向上の点でも重要になりつつあるとも言えますが、一方でシステム障害の要因は必ずしも欠陥等のソフトウェアの要素だけでなく、ハードウェア的な障害であるケースも多いです。このような場合には、ハードウェア自体の信頼性向上が必要であるとともに、障害が発生した際にハードウェアとソフトウェアのどちらに原因があるかを迅速に特定することも重要です。

すなわち、「ダウンしないシステム」を構築するだけでなく、「ダウンした場合にいかに早く復旧するか」といった点も、情報システムの信頼性の観点で従来から重要とされており、これは後述の通り、可用性や MTTR、MTBF などが信頼性の評価指標として多く利用されていることからも明らかです。

先進的な手法の適用については、テスト駆動型開発を採用し、要件定義と同時にテストプランを策定するといった取り組みの例が見られました。

テスト工程は、システムの品質向上において重要であり、革新的な取り組みは特に見られないものの、ユニットテスト/システムテスト/実環境あるいはエンドユーザによるテストのような段階的なテストや、欠陥修正による影響をチェックするリグレッションテスト、ソースコードのレビュー、要件定義時のテスト項目策定など、いわゆる正攻法的な取り組みが実践されています。

また国際標準の導入の観点では、CMMI の適用、ISO の各種規格採用などの例が、いくつか見られました。

(1-2) 欧州

◇政府・政府系機関

ここでは大学等の取り組みも含めた非民間企業の取り組みや EU 全体の動向について説明します。

ディペンダビリティ/信頼性に関する研究活動として、IEEE における「ディペンダブルなシステムとネットワーク国際会議（IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)）」が例として挙げられます。この分野では、米国が最先端で、欧州、日本がこれに続く状況である、との声も聞かれています。全体としては、ディペンダビリティ、信頼性、安全性、セキュリティといったテーマで行われている各取り組みで重複があります。

なお信頼性やディペンダビリティの確保・向上のためには、単なるソフトウェアの信頼性だけでなくシステムレベルの信頼性が必要であり、そのためには人的要因やマネジメントの要素も視野に入れるべきである、といった考え方が政府にも認識されています。また、信頼性に対する要件定義についても洗練されたアプローチが必要となっていますが、政府機関自身が IT 調達の最大の顧客であることから、政府の調達プロセスにも影響を与えることになります。

◇民間

エネルギー分野においては高い信頼性が求められることから、形式手法などの研究が行われており、信頼性向上に効果をもたらしています。

また交通管制などの信頼性が求められる分野では、リスク分析やソフトウェア開発に対して ESARR4 と ESARR6 と呼ばれる規定が適用されています。

その他の分野においては、モデル駆動型アーキテクチャ（MDA）や自動コード生成などの手法も研究レベルでは取り組まれています。なお上述の形式手法については、一般的には優先的に使う手法ではなく、サポート的に利用されるものです。

(2) 情報システムの信頼性確保に向けた国家の研究開発戦略

(2-1) 米国

NSF（全米科学財団）等が推進する、米国の研究開発の取り組みである「ネットワーキングおよび情報技術研究開発（NITRD）プログラム」は、毎年約 10 億ドル（1,000 億円弱）の予算で取り組まれていますが、そのなかでもコンピュータおよび情報科学とエンジニアリングの分野には、5 割以上が費やされています。2009 年度予算において、同分野に含まれるテーマの 1 つに「高信頼性のあるソフトウェアおよびシステム（High Confidence Software and Systems）」が挙げられており、このなかでディペンダビリティに関する研究開発が行われるこ

とが想定できます。

具体的には、コンピュータベースの安全性、ディペンダビリティ、正確性を保証するシステムと検証技術のための高信頼ソフトウェアとシステムをテーマとしています。これらの研究の対象とするアプリケーションとしては、スマート自動車、センサーネットワーク、組込み系医療デバイス、ポータブル・パーベイシブコンピューティングなどの、サイバーと実世界における将来的な複雑なシステムが挙げられています。

なお、NSF では、従来からディペンダビリティに関する研究開発に取り組んでおり、上記のほか、セキュリティ、プライバシ、信頼性、ユーザビリティを含むサイバートラストなどのテーマも手掛けています。

また、政府各省庁が所管する産業分野においては、ソフトウェア開発やリスクマネジメント等の標準等が省庁で定められています。

(2-2) 欧州

EU レベルでは第 7 次プログラム（FP7）において、ソフトウェア工学が重要なテーマとなっているとともに、レジリエンス（Resilience、回復性。「Resiliency、回復力」とも言われる）が近年注目されています。これは、故障のない、人に害を及ぼさない、正しく機能するシステムの構築を目指すものであり、その要素の多くはディペンダビリティにおける従来の取り組みがベースとなっています。

政府の方針としては、この分野で成果を得たいという考えがあるものの技術的にも難しいテーマであることや、上記のように類似するテーマにおいて重複する分野がある点は、大きな課題かつ問題となっています。特に、従来の信頼性のテーマに対する予算支援が中止され、ディペンダビリティへの支援に移っているのが現状であり、さらに近年はレジリエンスに注目されていることから、1 つのテーマを中長期的に実施することが難しい状況と言えるようです。

(3) 情報システムの信頼性向上に関する研究テーマ

(3-1) 米国

国家戦略に関する項目で述べたように、サイバー世界と実世界をまたぐような複雑なシステムにおける信頼性、ディペンダビリティの維持・向上が、NSF の研究テーマとなっています。そのなかでは、スマート自動車、センサーネットワークに例示されるようなアプリケーションが対象システムとして想定されていますが、これは仮想世界と実世界をまたぐシステムであると同時に、その実現のためにはソフトウェアとハードウェアを組み合わせた信頼性向上技術も、1 つの要素であると言えます。

さらに、従来 NSF で取り組まれているセキュリティ、プライバシ、信頼性、ユーザビリティを含むサイバートラストの観点でのディペンダビリティも研究テーマとなっています。

また、ヒアリング結果を踏まえた研究テーマとしては、ハードウェア上の制約とソフトウェアコードの複雑さが及ぼすレスポンスタイムへの影響や、追加されたコードが不具合を生じていないかをテストするための自動チェックのしくみなどが挙げられます。

(3-2) 欧州

前述の通り EU レベルの取り組みでは、従来の信頼性からディペンダビリティへ、さらには、レジリエンス（回復性）が研究開発テーマとなっています。具体的には、故障がない、人に害を及ぼさない、正しく機能するシステムの構築を目指すものでありますが、本質的な研究要素はディペンダビリティに関する考え方を応用するものと考えられます。

また、セキュリティ上の重要なシステムに対するリスク分析も EU のテーマとなっており、CORAS プロジェクトをはじめとして取り組まれています。

ソフトウェア開発手法としては、形式手法やソフトウェアプロダクトライン、モデル駆動型アーキテクチャ（MDA）、テスト行動型開発、自動コード生成などの様々な手法が研究プロジ

ェクトのニーズに合わせて利用されています。

学会レベルでは、フォールトトレラント・システムの研究や障害発生に対応する自動システムの研究などがテーマとなっています。

一般に信頼性（reliability）と比較して、ディペンダビリティはミッションや取り組み、障害の様子や種類などの要素を含むと同時に、アーキテクチャ的な考察も含まれています。

また、」明確な研究テーマというわけではありませんが興味深い課題として、開発フェーズにおける品質保証だけではなく、運用・メンテナンスフェーズでの品質保証のしくみがより重要となっています。これは、開発より運用・メンテナンスの方が 10 倍前後長いこともありますが、システムがよりオープンになり、従来と比べ運用段階において正常に動作し続け、信頼性を維持することが一層難しくなってきているためといえます。

さらに信頼性あるいはディペンダビリティには、技術的な要素だけが存在するわけではないことは既に述べた通りですが、法的あるいは社会的つながりを重要視するトラスト（trustworthiness）が一部で注目されており、これは現場でのシステムの研究が必要とされています。

(4) 情報システムの信頼性確保に向けた産学官の方針・戦略の比較

(4-1) 米国

研究開発においては、IEEE など国際的な学会においてディペンダビリティなどに関する取り組みが行われていますが、一般に米国が最も先進的であり、欧州、日本がこれに続くとされています。

また米国内の研究開発においては、前述の通りサイバー世界と実世界をまたぐような複雑なシステムにおける信頼性、ディペンダビリティの維持・向上が NSF の研究テーマとなっています。NSF は IT やコンピュータシステムなど様々な分野における基礎研究に対して予算措置を行っており、大学や企業などの研究機関が NSF の支援を受けるとともに、大学間あるいは大学と企業のコラボレーションなども行われています。

一方、実ビジネスにおける情報システムの信頼性については、各産業分野を所管する省庁が独自の標準を規定したり、あるいは国際標準や業界のデファクト・スタンダードの利用を推奨したりしています。

(4-2) 欧州

欧州では、IST(Information Society Technologies)において様々な IT 分野の研究開発が取り組まれています。そのなかで情報システムの信頼性確保については、セキュリティ、信頼性、ディペンダビリティ、レジリエンスといったキーワードによる研究が大学や企業によってなされています。

特に、前述の通り EU の研究開発の第 7 次枠プログラム（FP7）においては、ソフトウェアシステム工学が焦点を当てられているとともに、回復性（レジリエンス）といったテーマも注目されつつあります。

なお政府機関としての EU では、このような枠組みにおける研究開発に予算措置を行っているものの、短期間での成果を期待しすぎるとの声も聞かれます。また信頼性、ディペンダビリティ、レジリエンスなどは、本質的な技術要素は共通する場合が多いものの、ディペンダビリティ、レジリエンスといった新しいキーワードが好まれる傾向にあり、逆に従来取り組まれていた信頼性のプロジェクトが中止されてしまうといった弊害も見られます。

上記のような問題点はあるが、政府は研究開発に対して投資を行い、また実用化を支援することが重要であるといった認識がなされています。

7.3　ソフトウェア工学に関連する国際会議に見る研究動向の分析

ソフトウェア品質・信頼性に関する著名な国際会議はこれまでにも数多く開催されています。例えばDSN　(Annual IEEE/IFIP International Conference on Dependable Systems and Networks)、ICSE　(International Conference on Software Engineering)、ISSRE　(International Symposium on Software Reliability Engineering)があります。

DSNは前身となったFTCS (International Symposium on Fault-Tolerant Computing)から数えると、ゆうに35年を越えて開催されて来ており、またこれら3つのなかで最も歴史の浅いISSREでも初回は1990年代初頭から始まっているようです。

これらの国際会議のうち、ここではソフトウェア工学全般に渡ってテーマをカバーするICSEを取り上げ、Web上に公表されている情報に基づいて、ここ3年間の動向について見てみることにします。

(1) 論文テーマとセッションのカテゴリおよび件数

論文を大きく分けると、research papersいわゆる理論に重きを置いた論文と、実務者（practitioners）が主体となり、実際の経験に基づく知見に重きを置いて執筆した論文（2007年の呼び方はexperience reports）とがあります。それぞれにおいて採録された論文は、下記に述べるいくつかのカテゴリに分けられた上で、国際会議でのセッション（あるいはトラックと呼ばれます）を構成します。近年のICSEでは、おおむねresearch papersならびにexperience reportsともに、1つのセッションあたり3編となっているようです。セッションによってはそのテーマの採録論文が多い場合、複数のセッションが同一テーマに設定されます。

例として、表 7-2に2007年のカテゴリと採録論文数をそれぞれ示します。

表 7-2　2007年の論文カテゴリと採録数

2007	
Research papers	
Program analysis	9
Testing	6
Debugging and fault correction	6
Design	6
Models	3
Clone detection and removal	3
Aspect-oriented software engineering	3
Maintenance	3
Software architecture	3
Human aspects in software development	3
Refactoring and reuse	3
Security	3
Software defects	2
Experience reports	
Agile methods and software design	3
Performance and metrics	3
Modeling	3
Testing	3
Software development processes	3

表 7-2よりresearch papersは53編、experience reportsは15編となっており、特に前者については、ソフトウェア工学の広い範囲の分野から投稿されていることが伺えます。

またこれらのカテゴリ分けは、各回のプログラム委員会の考え方などで変化し、表 7-3 に示す 2008 年の例では、表 7-2 と細かい点では異なっていることがわかります。

表 7-3　2008 年の論文カテゴリと採録件数

2008	
Research papers	6
Specification	6
Testing	3
Software tools	3
Components and reuse	3
Empirical software engineering	3
Empirical software process	3
Empirical testing and analysis	3
Program analysis	3
Format analysis	3
Software process	3
Program comprehension	3
Architecture	3
Evolution	3
Refactoring	3
Frameworjds	3
Software engineering economics	3
Models	2
Experience papers	
SE in health care	8
SE on automotive systems	9
Telecommunications	9
ー Modeling and architecture*	3
ー Quality and processes*	3
ー Invited industry speakers*	3

(* Telecommunications の小分類、件数は内訳)

この 2008 年の開催においては、experience track のカテゴリは医療、自動車、通信に絞られています。また、これらのうち通信カテゴリの invited industry speakers での講演者の所属は、Opera software および Deutsche Telekom となっており、それぞれの講演タイトルは "Architecture and process: the anatomy of a web browser", by Rikard Gillemyr (Opera Software) および "Industrial software engineering in applied R&D: an operator's view", by Heinrich Arnold (Deutsche Telekom)でした。

さらに 2009 年に目を転じると表 7-4 の通りですが、世界的な不況により産業界からの参加はどの国際会議においても減少傾向にあるなかで、Experience papers において次に示す株式会社日立製作所（Software Division）、株式会社富士通研究所からの 2 論文が審査を通過して採録となっています。

・"Using a Validation Model to Measure the Agility of Software Development in a Large Software Development Organization", by Mikio Ikoma, Masayuki Ooshima, Takahiro Tanida, Michiko Oba (Software Division, Hitachi, Ltd.), Sanshiro Sakai (Shizuoka University).
・"WEAVE: Web Applications Validation Environment", by Sreeranga P. Rajan, Oksana Tkachuk, Mukul Prasad, Indradeep Ghosh, Nitin Goel, (Fujitsu Laboratories of America), Tadahiro Uehara (Fujitsu Laboratories Limited).

これら 2 論文は、いずれも Validation のカテゴリとなっています。

表 7-4　2009 年の論文カテゴリと採録件数

2009	
Research papers	
Program analysis	6
Testing	4
Software quality and metrics	3
Program comprehension	3
Modeling	3
Model synthesis	3
Maintenance	3
Dynamic adaptation	3
Development tools	3
Development paradigms and software process	3
Debugging	3
Concurrency	3
Components	3
Collaborative development	3
Code generation and transformation	3
Web applications	2
Experience papers	
Architecture	4
Testing	4
Model-driven	2
Validation	2
Defect prediction	1
Process improvement	1
Risk management	1
Security	1

(2) 研究動向と国際会議への参加の勧め

各回で 50 編程度に絞られた research papers（論文の応募・投稿は開催年にもよりますがおおむね 400 編を超えます）においては、特定の企業における開発現場で得られた知見を取りまとめたものは少ないです。誤解を恐れず総じて言えば、ソフトウェア品質と生産性の維持・向上について理論に裏うちされた、例えば上流工程では要求理解のための社会工学的アプローチや下流工程での新しい数理的分析手法を論じる、といったものが多いようです。一方、experience reports/papers はその反対と言え、例えば既存の理論＋αのものをベースとしたソフトウェアエンジニアリングツールの開発による品質向上策への取り組み、SAP の Java コードを実際の分析対象とした defect-prone モジュール（欠陥を含んでいる可能性が高いソフトウェアモジュール）の予測の手法についての

報告など、実務に携わる方々にとって、より関心の高い論文が採されています。これら全体の割合としては、おおむね理論研究：実証研究＝3：1 です（2009 年度の実績では 51：16）。この国際会議の性格から、理論研究の方が投稿件数も多いことは否めませんが、開発現場での成果と理論研究による結果との間のギャップを狭めて行けるかどうかというところに、こうした計測・評価技術の開発と実務における実用化のもつ本来の困難さがあるとも考えられ、これを克服していくためにも我が国の実務家の方々にこうした場で発表される件数を今後増やしていって頂けることを期待するものです。ソフトウェア工学においても日本的品質管理なるものが成立し得るとするならば、それは理論に裏付けられた実務の側から湧き出すものではないでしょうか。

この節では、いわゆるソフトウェア工学のトップカンファレンスの 1 つである ICSE を題材にとりましたが、近年、より実務家の成果を発表する場としての性格が強い国際会議である ESEM（Empirical Software Engineering and Measurement）も始まっていることを付記します。

第 1 部 参考文献

第 1 章 高信頼ソフトウェアにおける問題意識

[1] 日経コンピュータ 2009.8.19, 日経 BP 社 (2009).

[2] JIS C 5750-1:2010 ディペンダビリティマネジメント第1部: ディペンダビリティ マネジメントシステム, 日本規格協会 (2003).

第 2 章 高信頼ソフトウェアとは

[3] 重要インフラ情報システム信頼性研究会報告書, IPA, 2009.

第 3 章 予防活動および検知活動にかかわる手法

[4] 細川: 仕様書の欠陥を検出する上流インスペクション, ソフトウェアテスト PRESS, Vol.2, 技術評論社 (2006).

[5] 織田: ソフトウェア・レビュー技術, ソフトウェア・リサーチ・センター, .

[6] Wiegers (大久保監訳): ピアレビュー, 日経 BP ソフトプレス, 日経 BP 出版センター, 2004.

[7] グラハム, ファン・フェーネンダール, エバンス, ブラック: ISTQB シラバス準拠 ソフトウェアテストの基礎, センゲージラーニング, 2008.

[8] 高橋: 知識ゼロから学ぶソフトウェアテスト, 翔泳社, 2005.

[9] マイヤーズ, トーマス, バジェット, サンドラー: ソフトウェアテストの技法, 第 2 版, 近代科学社, 2006.

[10] バイザー: ソフトウェアテスト技法, 日経 BP 出版センター, 1994.

[11] 保田, 奈良: ソフトウェア品質保証入門, 日科技連出版社, 2008.

[12] IPA SEC： SEC BOOKS ソフトウェア開発見積りガイドブック，2006 年 6 月.

[13] IPA SEC： SEC BOOKS 続・定量的品質予測のススメ

第 5 章 トレーサビリティ管理の手法

[14] JIS Q 9025:2003 マネジメントシステムのパフォーマンス改善 -品質機能展開の指針, 日本規格協会 (2003).

[15] 新藤編（赤尾, 吉沢監修）：実践的 QFD の活用 － 新しい価値の創造, 日科技連, 1998.

[16] 田井: 要求品質と開発目標 ～”感動”具現化ツールとしてのQFD～, 連載開発工学(その 3）, YAMAHA MOTRO TECHNICAL REVIEW 2003-3 No.35 (2003).

[17] 新倉: 実践ソフトウェアの ISO9000, 日本能率協会マネジメントセンター, 1995.

第 6 章 テスト網羅性の高度化技法

[18] SQuBOK 策定部会: ソフトウェア品質知識体系ガイド, オーム社, 2007.

[19] 湯本, 秋山, 西, 池田, 鈴木: JaSST '09 Tokyo クロージングパネル資料, NPO 法人 ASTER (2009,http://jasst.jp/archives/jasst09e.html#closing).

[20] 吉澤, 秋山, 仙石: ソフトウェアテスト HAYST 法入門, 日科技連出版社, 2007.

[21] Kuhn, Wallace, Gallo Jr.: Software Fault Interactions and Implications for Software Testing, IEEE Transactions on Software Engineering, vol. 30, No. 6 (2004).

第 7 章 高信頼ソフトウェア開発に関する海外事例および研究動向

[22] 情報システムの信頼性評価手法の調査報告書, IPA （予定）.

[23] DSN (2009), http://2009.dsn.org/.

[24] ICSE, http://www.icse-conferences.org/.

[25] SSRE (2009), http://www.issre2009.org/.

[26] ESEM (2009), http://www.csc2.ncsu.edu/conferences/esem/.

第 2 部

事　例　編

< 空欄 >

第1章　事例編の概要

第1章では、高信頼ソフトウェアに向けた先導的な取り組みを行っている各社の事例概要を紹介します。

各事例は、高信頼ソフトウェアのための取り組みとして、要件開発、要件管理、妥当性確認、検証、リスク管理など、プロセス領域として多方面にわたっています。そのため、各事例の構成は統一せずに、事例ごと固有の構成にしています。また事例編においては、欠陥や障害などの用語は企業の固有用語により記載しており、第1部とは意味合いが異なる場合があります。

第2章から第8章で紹介している各社事例の概要は、次の内容となっています。

● 東京海上日動システムズ株式会社

高品質なシステム開発のためには、システムの重要度にあわせたリスク評価・対応と、上流工程での要件の精度の向上や設計書・プログラムミスの削減が必要である。

開発するシステムをシステム運用の観点から評価する、「システム重要度に基づいたリスク管理」（システムプロファイル）の取り組みを紹介する。

また、設計・製造工程の品質向上の取り組みとして、上流工程での要件精度向上の取り組みと、設計・プログラミングの開発局面の品質向上を目的とした取り組みについて紹介する。

● 富士ゼロックス株式会社

ソフトウェアの品質保証の課題と対応について、下流工程であるソフトウェアテストの改善から上流に向かっていった事例である。一般に下流工程での問題点は、お客様に直結するため明白であり、また下流工程の改善は、その対策効果を把握しやすいという特徴がある。

富士ゼロックスでは、HAYST法と呼ばれる直交表の技術を利用したテスト技法を核として改善を進めている。組込み系の事例ではあるが、エンタプライズ系においても参考になるだろう。

● 日本ユニシス株式会社

日本ユニシスでは、かねてより品質保証を支える重要技術として、プロセス管理とともにテスト技術を重視し、数年前からインドのテスト専門会社であるSTAG Software Private Limited社（以下STAG社）の技術を導入し、重要プロジェクトに適用してきた。

その結果を踏まえ、品質保証部門内にテスト技術を用いたソフトウェア検査を専門とする部隊を設置し、プロセスとプロダクトの両面からシステムの開発行為を組織として保証する体制が整った。

本稿では、今回強化したプロダクト面での第三者品質保証体制での活動について紹介する。

● 株式会社日立製作所

要求されるシステムの信頼度に応じて、システムの品質ランクを5段階に設定している。

特に高信頼性を要求される社会インフラシステムについては、通常の開発プロセスで実施するレビューに加え、要件定義から運用・保守の各工程で厳格な公式レビューを実施することにした。

上記の公式レビューをプロモートするために、開発プロジェクトとは独立した専任部

署を設置した。
　専任部署を設置したことにより、レビュー精度の向上、レビュー指摘事項のトレーサビリティ確保、レビューノウハウの蓄積と共有が促進され、重大障害軽減に寄与することができた。

● 富士通株式会社

　富士通では、システム開発の失敗原因の多くを占める上流工程で発生する課題・問題に着目した。要件定義の妥当性を確認するための監査と、外部設計ドキュメントの完成度を診断する施策の内容と実施状況を紹介する。
　これらを実施した結果、プロジェクトのQCDに寄与する成果が得られており、機能性の品質確保を通して、使用性、効率性にも良い影響を与え、高信頼システムの実現に役立っている。

● 三菱電機株式会社

　三菱電機では、最近の重大障害の発生傾向から外部調達のISV/IHV製品を起因とする障害が増加傾向にあることに着目し、その発生を予防するために非機能要件のトレーサビリティ管理手法への取り組みを試行している。
　非機能要求を具体的に開発し、確認するための作業手順をプロセスとして示すとともに、非機能要求管理の成果物として作成する非機能要求トレーサビリティマトリクスについて、その形式と利用方法を紹介する。
　結果として、役割の明確化、作業項目や試験項目の具体化、見える化に寄与していることが確認できており、高信頼システム構築に役立っている。

● 株式会社ジャステック

　ジャステックでは、独自のソフトウェア生産管理方式「ACTUM」およびCMMIレベル5（V1.1およびV1.2ともレベル5達成）に基づき、高信頼ソフトウェアへの取り組みを行っている。
　ここでの取り組み事例では、「ACTUM（環境変数）」を使用し、システムプロファイルごとの要求品質と開発コストとの関係、および出荷後の残存欠陥と開発コストとの関係について、弊社の統計分析および経験則に基づいて紹介する。
　さらには「品質機能展開」を使い、要求機能および品質特性「代用特性」ごとの重要度、ならびに要求機能ごとの開発コスト（環境変数）をとらえ、開発予算に見合った機能および品質を調整する方法を紹介する。

第2章　東京海上日動システムズ株式会社の事例
～　システム重要度に基づいたリスク管理と、設計・プログラム工程での品質向上施策　～

2.1　取り組みの背景

弊社で開発・運用しているシステムは、保険会社の業務全般を担うだけでなく、全国の数万店の代理店さんにも様々な機能をオンラインでご利用いただいている。そのため、システム障害が発生した場合は、社内だけでなく広く代理店さんやお客様まで影響を及ぼす可能性がある。また、システムの停止がビジネスの停止につながる経営リスクも有している。品質確保と適正な開発力の確保は、経営サイドも重要課題と認識しており、従来から品質確保のために以下の取り組みを行ってきている。

過去の主な取り組み：

① アプリケーションオーナー制度とレビュー制度の導入
② 非機能要件と運用要件設計のレビューの導入

上記の結果、障害件数は大幅に減少したが、開発規模の増大や利用者の拡大により、さらに品質の向上を図る必要性が増してきた。

発生した障害の原因を分析した結果、以下の傾向が判明した。

① 開発に起因する障害で最も多いのは要件齟齬・要件誤りに起因するバグであった。
② 次に多い原因は、設計書不備・プログラムミスに起因するバグであった。

また、同じ原因のミスであっても、対象となるシステムによって影響は大きく異なるため、全社一律の基準でリスクを管理するのでは不十分であることも明確になった。

高品質なシステム開発のためには、システムの重要度にあわせたリスク評価・対応と、上流工程での要件の精度の向上や設計書・プログラムミスの削減が必要であることが、判明した。

本章では、開発するシステムをシステム運用の観点から評価する「システム重要度に基づいたリスク管理」の取り組みと、上流工程での要件精度向上の取り組み、設計・プログラミングの開発局面の品質向上を目的とした取り組みについて紹介する。

2.2　システム重要度に基づいたリスク管理

2.2.1　概要

システムリスク管理の実効性をより確実にするため、システム全体を一律の重要度でとらえるのではなく、システム個々に重要度を評価し、それぞれの重要度に応じた対策を行うことが必要である。

そこでシステムの概要と特徴を明確に定義し、システムの重要度を評価判定する「システムプロファイル」を作成することとした。

さらに個別システムのリスクを明らかにするために、「システムリスク・アセスメントシート」を作成し、個々のシステムのどこにリスクがあるかを明確にした。

この「システムプロファイル」と「システムリスク・アセスメントシート」を定期的に見直すことで、システムリスクを適正に把握し必要な対策が行われているかを管理できる仕組みとした。

2.2.2 システムプロファイル

システムプロファイルとは、システム単位ごとに、システムの概要と特徴（主な業務機能、システム構成、主なIN・OUT、稼働時間、利用基盤）を明確に定義するとともに、システムの重要度判定要素をもとに重要度を判定するシートである。

開発プロジェクトの要件確定時とサービスイン前に見直しを行う。さらに、定期的な再評価を実施する。

【主な記載事項】

- システム管理者、アプリケーションオーナー部門
- システムの概要
- 主な業務機能
- システム構成
  （システム構成、処理形態、基盤、開発・運用形態、非機能要件の状況）
- システム情報（重要なデータ、主要な関連システム、利用者、サービス時間）
- システム重要度（可用性、機密性、完全性）の評価結果

【重要度の評価】

可用性、機密性、完全性の3つのカテゴリの評価を行い、その結果をもとに重要度のランクを判定する。

ａ．**評価基準**

①可用性

お客様や代理店さんに影響を与えるシステムを重要度の高いシステムとして認識し、業務を停止させない等の確実な対応を講じる必要がある。

お客様・代理店さんへのサービス提供の有無と、当該システムの停止による影響度を勘案して評価する。

②機密性

個人情報や機密情報を取り扱うシステムは、情報管理の観点から重要性の高いシステムとして認識する。個人情報・機密情報の取り扱い有無を評価する。

③完全性

財務報告を歪めるリスクがある業務（重要な勘定科目への影響がある業務）を対象とするシステムは、会社経営への影響度合いが高いシステムとして認識する。財務報告を歪めるリスクの有無を評価する。

b.システム重要度の判定

可用性・機密性・完全性の３つのカテゴリの内、高重要度の評価が１つ以上あるシステムの重要度ランクを「A」とする。

また、共通基盤の重要度は標準値を「A」とする。

ランク「A」判定のシステムについては、リスク管理面からリスク最小化に向けた対策が十分に考慮されているかを「システムリスク･アセスメントシート」を用いて評価する。

2.2.3 システムリスク・アセスメントシート

システムリスク・アセスメントシートとは、個別システムのリスクを明らかにするためのチェックシートであり､システムリスク評価にかかわる設問で構成されアプリ編と基盤編に分かれている。システム重要度ランクが「A」ランク評価である場合に作成する方針としているが、初回は全システム単位を作成対象として作成した。

【主な内容】

- リスクカテゴリ（情報の不正使用、情報システムへの不正侵入、情報の改ざん、情報システムの破壊・停止、利用上の過失、自然災害）別に、リスクを評価する

視点をあらかじめ明記。
・ 視点にそって、システム担当が評価し、運用部門がさらに評価を行う。

2.2.4 効果

システムの重要度を明らかにすることで、可用性確保の基準が明確になり、適切なコスト配分が可能となった。同重要度は運用SLAともリンクしており、よりビジネスインパクトに則した運用管理ができるようになった。

2.3 設計・製造工程の品質向上

2.3.1 取り組みの背景

弊社で実際に開発した案件について、テスト工程の作業量と品質（障害発生状況）を分析した結果、プロジェクトごとに状況は異なるものの、テスト工程での品質とリリース後の品質にはある程度の相関関係があることが明らかになった。

（詳細は、既刊「ソフトウェアテスト見積りガイドブック：第2部事例編」を参照）

・結合テスト・システムテストともにバグ発生率が低いと、全体の生産性が高く、サービスイン後の障害は少ない。
・一方、結合テストは平均以下だが、システムテストで平均以上のバグが発見されたグループは、全体の生産性が最も低く、サービスイン後の障害も多い。

この結果から、結合テスト・システムテストのバグ発生率を低下させることが、本番障害率の削減につながることが見えてきた。

テスト工程でのバグを削減させるために、設計段階でバグを作りこまないことと、単体テストまでの前段階でバグを早く検出することの2つのアプローチから取り組み施策を検討した。

① 要件齟齬・要件誤りに起因するバグは、外部設計での作りこみを削減する
② 設計書バグやコーディングバグは、当該工程で見つけ、次工程への持ち越しを削減する。

上記の2つの取り組みについて、具体的な内容を以下に述べる。

2.3.2 要件の精度向上の取り組み

a. 要件定義に関する障害の状況

ある期間のリリース後の障害原因の約55%は、要件定義にかかわる問題であった。これはアプリケーションオーナーと開発部門、または開発部門とパートナー会社間の要件認識の齟齬、あるいは思い込み・考慮漏れに起因するものであった。

要件の誤りの発見には業務知識が必要であるため、詳細設計やプログラミング工程では検出しづらく、システムテスト工程に持ち越されることが多いため手戻りのロードが大きくなっていた。

b. 取り組みの概要

取り組み内容を一言でいうと「要件定義工程で、システムテストケースを作成する」ということである。一般的に“Wモデル”とか“V&Vモデル”と言われているものに近いが、特に“要件定義”と“システムテストケース”に焦点を当てたことがポイントである。

弊社ではシステムテストのテストケースを、サプライヤとアプリケーションオーナー双方が作成している。要件定義時に、アプリケーションオーナー・サプライヤ双方で、要件に対する“システムの動き”（＝テストケース）と“答え”（＝テスト結果）をあ

らかじめ確認することで、要件の精度を高めるものである。
利用者を想定したテストケースを検討することによって、要件がより明確になり、また要件の齟齬や漏れが見つけやすいという効果が期待できる。

2.3.3 設計・製造品質向上の取り組み

a. 設計・製造品質に関する状況

アプリケーションオーナー制度や公式レビュー制度により要件の確認やプロジェクト管理面でのレビューは充実してきたが、一方、個々の設計書やプログラムの品質については、プロジェクトリーダやパートナー会社に任せられており、どのような品質状況になっているかを把握できていなかった。

b. 取り組みの概要

レビュー時の指摘漏れを防止するため、設計書のレビューポイントを統一し指摘結果を記録して、設計書の品質向上を図った。
プログラムの単体テストの結果を記録し、単体テストの精度向上を図った。
プロジェクトリーダが記録した結果を確認することを義務付け、担当者や会社によるバラつきを削減していった。

c．具体的な対応内容

① 設計品質の向上

レビューポイントを記載した「設計書レビューシート」を作成し、レビューの都度、レビューポイントごとの指摘事項数、レビュー時間、ページ数などを記録する。
レビューポイントは “機能・記述面”と“標準化”の2つに大きくわけ、それぞれに指摘件数を記述する。
「設計書レビューシート」は設計者からレビューア、プロジェクトリーダへと回付され、設計書1つずつの品質を確認できるようにした。

② 製造品質の向上

単体テスト工程で実施すべきことを記載した「単体テスト確認シート」を作成し、テストケースと障害の状況、プログラムソースコンペアの実施状況や網羅率の結果を記録する。
「単体テスト確認シート」は、プログラマから設計者、プロジェクトリーダへと回付され、プログラム1つずつの品質を確認できるようにした。

2.3.4 取り組みの効果

定量的な効果は現在分析中であるが、実施した案件では、それぞれ以下の声が挙がっている。

a. 要件精度向上の取り組みの効果

- 従来あいまいだった内容をより具体的に定義した上で開発に入るため、その後の工程がスムーズになった
- アプリケーションオーナーも要件提示時にテストケースを考えるため、要件がより具体的になった
- アプリケーションオーナーサイドで要件の再確認をする機会ができた
- 仕様についてアプリケーションオーナーと認識すり合わせができた
- テスト計画の準備が前倒しにでき、工数が削減できた
- 検証すべきテストケースを明確化してパートナー会社と打ち合わせを行う事で、エビデンスを見てから漏れに気づきテストケースの追加をすることによる二度手間を防ぐことができ、委託工数の削減となる

b. 設計・プログラム品質向上の取り組みの効果

レビューシートの使用について、以下の意見が寄せられた。

- 設計書毎の品質のバラつきが明確になり、重点的に再レビューすべき設計書・担当者などが明確になった
- レビューを行う際のポイントが明確になった
- 誤字・脱字や記述不足などの標準化・形式的な指摘だけでなく、要件の明確性や要件認識の同一性など、機能・記述面での指摘が予想以上に多かった
- 部門や会社別のレビューの状況がわかるようになった
- 単体テストでのバグの摘出率が上がった
- 言語別のバグ摘出率がわかるようになった
- プログラム改定時のコンペアテストの実施状況が確認できるようになった
- 部門や会社別の単体テストの実施状況がわかるようになった

c. 2つの取り組みの相乗効果

取り組みを始めてから8ヶ月間のデータを分析したところ､以下の結果が判明した。

要件定義工程ではレビューの指摘件数が増加した。これは「要件の精度向上」と「設計・製造品質の見える化」の2つの取り組みの相乗効果により、自工程での設計書バグの検出率が高まったものと評価できる。

取り組み当初と最近の結果では、開発規模あたりの機能・記述面での指摘件数が3.5倍と増加した。これは「見える化」により、レビューの実施状況が明らかになりレビュー回数が増加し、精度が高まった結果と考えられる。

また、指摘内容としては、以下の3点の増加が目立つ。（図 2-1参照）

①「対象範囲の明確性の観点」
②「要件認識の同一性」
③「要件とテストケースの対応関係」

特に②、③の指摘事項は、テストケースを同時に作成することでの効果が大きいものと考えられる。

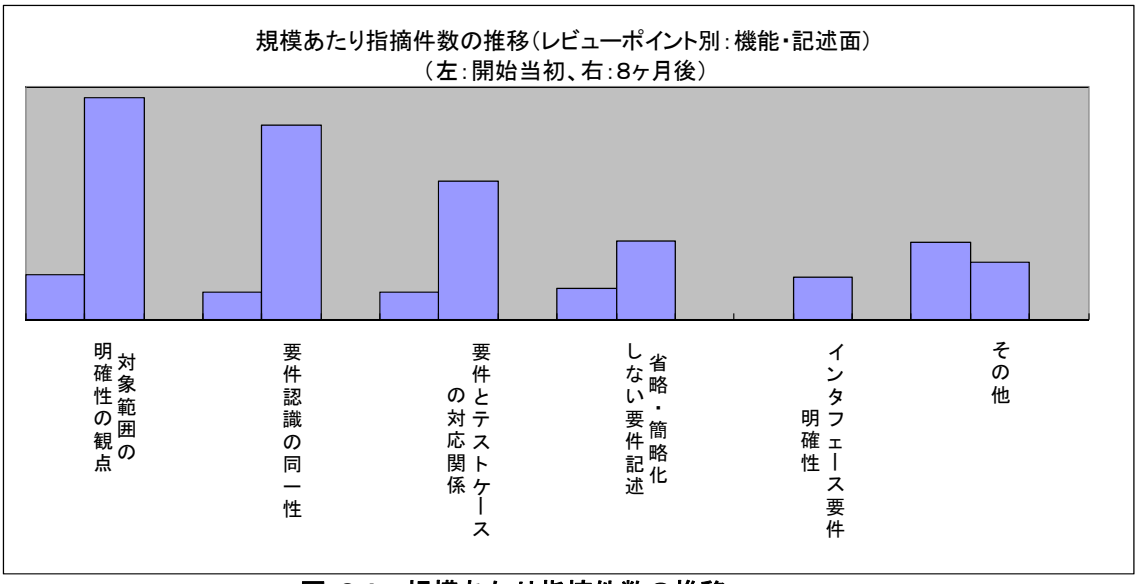


図 2-1　規模あたり指摘件数の推移

参考文献

[1] IPA SEC： SEC BOOKS ソフトウェア開発見積りガイドブック，2006 年 6 月.

< 空欄 >

第3章　富士ゼロックス株式会社の事例
～ HAYST 法　～

3.1　ソフトウェアの品質保証の課題と対応

当社におけるソフトウェア開発プロセスは、IPA/SEC の『ESPR Ver.2.0 SEC-TN07-005』[1] の V 字モデルをベースとしている。そのなかでソフトウェア評価部門は、「SYP4：システムテスト」以降を担当する第三者検証組織として位置付けられている。

V 字モデルの開発プロセスを実践するにあたり、上流工程のアウトプットである仕様書のレビューや、それに対応するテスト計画の早期作成の重要性が認識されたことから、要件決定時に関連部門の合意を形成するための仕組みと「要件合意システム」を構築するとともに、テスト設計者がテストの視点をもって仕様レビューに参加するといった対応が取られてきた。

しかし、単に新しいプロセスモデルを取り入れるだけでは相変わらず仕様変更が多く、かなり遅い工程まで仕様が固まらず、十分な効果は得られなかった。

大規模でかつ多人数の開発プロセスにおいて本質的な改善を進めるためには、上流工程においてアーキテクチャレベルで再設計を行い、それをプラットホームとして共通活用していくことによる品質の向上と、下流工程で成功したプラクティスを上流工程へ展開することが実践的でかつ効果的であった。本事例では後者、すなわち、テスト工程の改善と上流への働きかけについて述べる。

3.2　ソフトウェアテストの概略

ソフトウェアの品質保証を行うためにはテストによる品質確認が欠かせない。ここでは、当社におけるソフトウェアテストの実際についてフェーズを追いながら述べる。

まず、「テスト戦略・計画」のフェーズにおいては、ソフトウェア開発プロセス全体として解決すべき課題を明確にして開発部門と品質評価部門の合名で「品質獲得戦略」を作成する。次に、その戦略に基づき ISO/IEC 9126 をベースに独自の品質特性を追加したものを「品質点検表」として商品開発開始時に提示している。

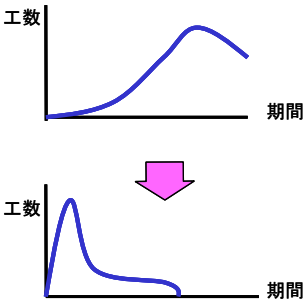


図 3-1 手戻りの削減

ただし、この段階では未定の項目も残っているため商品出荷判定まで適宜改定が実施され、常に最新の情報に保たれるように注意している。

また上流工程である仕様検討時に、HAYST 法[2] の FV 表を用いたテスト計画を並行実施することによって手戻りを防ぎ、ソフトウェア開発全体に掛かる工数の削減を図っている。

次に「テスト設計」フェーズにおいては、直交表を用いた組み合わせテストの体系を HAYST 法という手法にまとめて実施しているところに特徴がある。直交表の割り付けには独自のテクニックが必要となるため、それをサポートするテスト設計ツールを開発し誰でも簡単に禁則処理を含めた割り付けができるようにしている。

また近年、複合機とサービスとの連携評価のニーズが高まるに従って、リスクベーストテストによるお客様視点での要求の重要度を加味した重点志向テストや、統計的テストを用いた信頼性の確認のためのテストも実施している。

「テスト実施」フェーズにおいては、上記「テスト設計」のアウトプットである「テストスクリプト」を低コストでかつ正確に実施することがポイントとなる。そのために、テスターの操作およびその結果確認を自動化する「評価自動化ツール」を使用している。ツール活用のポイントは、テストスクリプトの標準化とテストスタブの役割を果たすシミュレータの活用によるデバイスの問題の排除、そしてテスト技術者のスキルアップである。

ここでデバイスの問題とは、例えば「夜間大量自動印刷による紙切れ、またはトレイあふれによるテスト中断」であったり、「ハードウェアでは作り出しにくい微妙なタイミングの生成」のことである。

また、それと並行し市販の負荷ツールを活用した複合機への「大規模アクセスシミュレーション」テストを実施している。なお、自動化できない部分についてオフショアする際には、オフショア先の文化の理解とスキルの担保がポイントとなっている。

複合機のテストはテスト規模が大きいことから、複数の拠点にまたがったテストが実施されている。そのため、ネットワークを活用した「テスト管理」の一元化が必須であり、当社では SourceForge と PostgreSQL を利用した Web ベースの「テスト進捗管理システム」と ORACLE 社のデータベースを用いた「バグ管理システム」を開発し運用している。

ところでソフトウェアテストの向上には、「顧客情報のフィードバック」を欠かすことが出来ない。トラブル情報と VOC のシェアと活用に加え、中堅のテスト技術者を営業最前線の SE 部門や、場合によってはお客様先に訪問させ、現場でどのように複合機（特に新しいサービス）が使用されているか、その目で業務の理解をさせ、それをテストへ反映することがますます重要になってきている。また、これらフィールド訪問によって自身が SE や顧客へ提供すべき情報を理解し、積極的に有益な情報をまとめ展開を実施している。これらの情報の中には機密情報も含まれているが、図 3-2 のように当社の「コンテキストセキュリティ」機能を使用することによって、イントラネット内でしか閲覧できない仕組みを実現し、情報を必要とする者に対してスピーディに、かつ安全（セキュア）に情報展開することを可能としている。

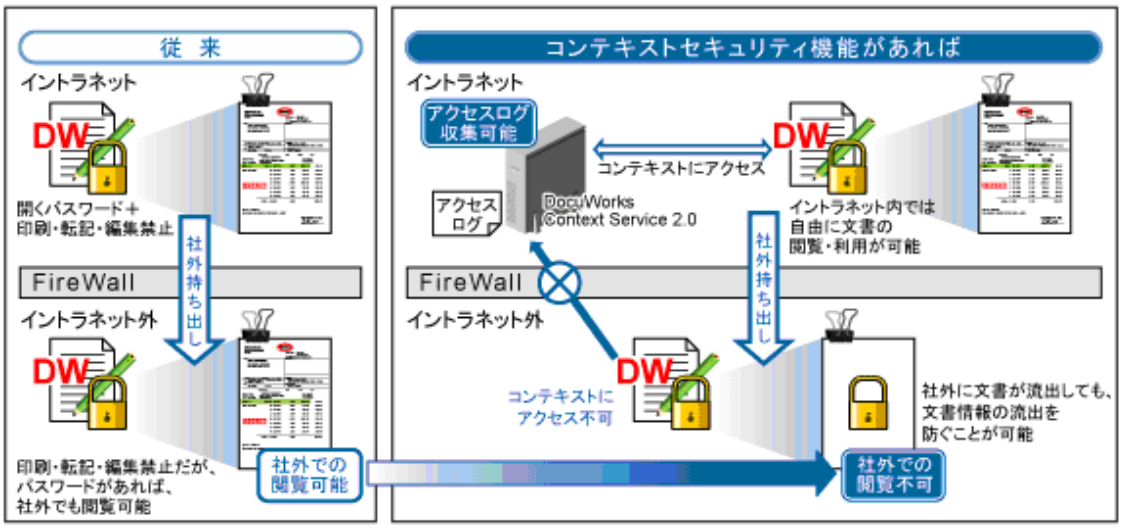


図 3-2 コンテキストセキュリティ

3.3 HAYST 法の詳細

上記活動において、特にユニークな HAYST 法について説明する。HAYST 法とは、”Highly Accelerated and Yield Software Testing” の略で、直交表を活用したソフトウェア組み合わせテストの技法の 1 つである。大規模ソフトウェア（入力として同時に与える可能性のある機能数が 300 程度の組み合わせテスト）へ適用できることと、禁則回避処理を持つ点に特徴がある。

HAYST 法の適用ステップと、そこで使用されている技術について以下に示す。

3.3.1 テスト戦略とテスト分析

HAYST 法のテスト戦略は、大きく 2 つの戦略から成り立っている。1 つは開発工程へのテスト戦略である。これは、HAYST 法の一部である FV 表(Function Verification table)、FL 表(Factor Level table)を作成することで、開発の初期段階で市場における使われ方を徹底的に分析し、そこで得た知見を適用することで要件のあいまいさの排除や、仕様の明確さと矛盾の排除を実施するものである。特に市場における使用条件を決める要因として、システムへの入力である信号因子と、入出力関係を乱す要因であるノイズ（誤差因子とも呼ぶ）の組み合わせを徹底的に分析する方法であり、品質工学の戦略を参考に構築を行った。

FV 表とはテスト分析工程で使用する表であり、テスト対象物を識別しテストの網羅性を確保することを目的としている。これまでテスト分析では、テスト対象物のインベントリと呼ぶテスト対象物階層化一覧表の活用が提案されている。これは要件や機能を、表の縦軸に優先度が高い順番で列挙し、横軸には既存のテストケースを並べて、機能と既存のテストケースの対応と漏れを認識しテスト網羅率を上げる方法である。

表を用いてテスト分析を行うやり方は、多くの企業で採用されている標準的な方法であるが、一般に巨大な表となり扱いにくいという問題点がある。そこでテスト対象物の認識と分解のために、列数が多くならない表形式とし、それを FV 表と名付けた。FV 表の形式は、以下の通りである。

表 3-1　FV 表

No.	目的機能 (F)	検証内容 (V)	テスト技法 (T)

まずテスト対象のソフトウェアの機能仕様書から、「機能」を取り出す。次に、その機能が持っている目的について考え、目的を含んだ文章に書き直して「目的機能」の欄に記入する。

ソフトウェアにはなんらかの製作意図、すなわち果たすことが期待される目的が存在する。それを FV 表では「目的機能」と呼んでいる。機能仕様書には、ソフトウェアというクラス図やプログラムコードを開発するための情報が書かれているが、それがどのような環境や背景、そして使用上の文脈で使われるかといった顧客の使い方については、記載されていないことが多い。従って、FV 表の目的機能欄に機能そのものの情報だけではなく、要求の背景や使用目的の情報を記載することで、正しい動作が行われていることのテストを実施することが可能となる。

次の欄は、機能仕様書から発見した目的機能に対する検証内容を記述する。検証内容に書くものは、「どのようなテストをしたらユーザの目的に叶う機能が作りこまれたことが確認できるか」である。「ユーザの目的に叶うか？」の確認方法を抜け漏れなく記述するためには、検証内容の欄に 5W2H の要素が入っていることを確認する必要がある。すなわち、

What? ： 機能仕様（正常時と異常時の機能仕様）
When? ： いつ使われるか
Where? ：どこで使われるか
Who? ： 誰が使うのか
Whom? ：誰のために使うのか
How? ： どのように使うのか
How much? ： 価格はどれくらいのソフトウェアなのか

を分析した上で、記載を行う。

最後は、テスト技法の欄である。この欄には、目的機能をテストするために使用するテスト技法を記述する。つまり、検証内容で発見したテスト対象物において確認すべきことに対して、その方法を記述する。

組み合わせを考慮する必要がない機能はほとんどないため、HAYST 法と書く場合が多いが、論理関係が複雑な場合は原因結果グラフ法や、機能図式、CFD 技法と書くこともある。また状態遷移が絡む場合は、n-スイッチカバレッジテスト法と記述する。通常は、1 つの目的機能に対して機能要件、非機能要件等、様々な観点でテストを実施するため、この欄には、複数のテスト技法を記載することになる。

このように、FV 表の 1 行は目的機能の粒度で、テストのための目的 Why と、検証内容である What、そしてテスト技法で手段 How を決定するという関係にある。言い換えると、ある機能の本来の目的を把握し、それが実現できたかの確認項目を記述し、最後に確認方法を記述する。

FV 表で、目的機能という「視点」でテスト対象物を分解することの利点は、テスト対象物をセキュリティ要件や、非機能要件も含めてカプセル化できることにある。ユーザは何かの目的を実現するためにソフトウェアを使用する。例えば、1 つの目的を実現する途中でセキュリティ要件が変わることは通常はありえない。また目的には、それを実現する時の振る舞いである性能や操作性といった非機能要件がついてまわる。つまり、目的機能単位に動作を確認する際に、非機能要件の確認・計測を実施すると非常に効率が良い。

HAYST 法によるもう 1 つのテスト戦略は、テスト工程に合わせて組み合わせるべき要因（因子）を段階的に変化させていくことにある。テストの初期段階、いわゆる単体テストにおいては、コンポーネント単位にテストを行う。その場合、関数の引数を因子とし、その取り得る値を水準とする。

次に統合テストにおいては、関数ではなく機能に着目し、1 つ 1 つの機能を因子とする。ここでは開発者の視点を中心に、特別な意味を持つ値（水準）を漏れなく選び出し FL 表を作成して、それらを直交表に割り付け組み合わせテストを実施する。また、コードの弱いところ（共有リソースなど）をシステムの機能を阻害する要因として選出し、それを誤差因子として組み合わせてテストを実施する。

そしてシステムテストにおいては、お客様が得ようとしている価値（目的）を因子・水準とする。またシステムテストにおいては、特に市場条件となる要因を誤差因子として積極的に取り込んだテストを実施する。つまり、製品に誤差因子という負荷を掛けたテストを実施することで、さまざまな市場条件下において安定して動作する頑健性・堅牢性（ロバストネス）を確保するためである。

3.3.2 テスト設計

HAYST 法のテスト設計では、図 3-3 のテストモデル（ラルフチャートと呼ぶ）を用いて、テスト対象システムのモデル化を実施している。テスト対象システムは、コンポーネントテストにおいてはコンポーネントであるし、システムテストであれば目的機能となる。

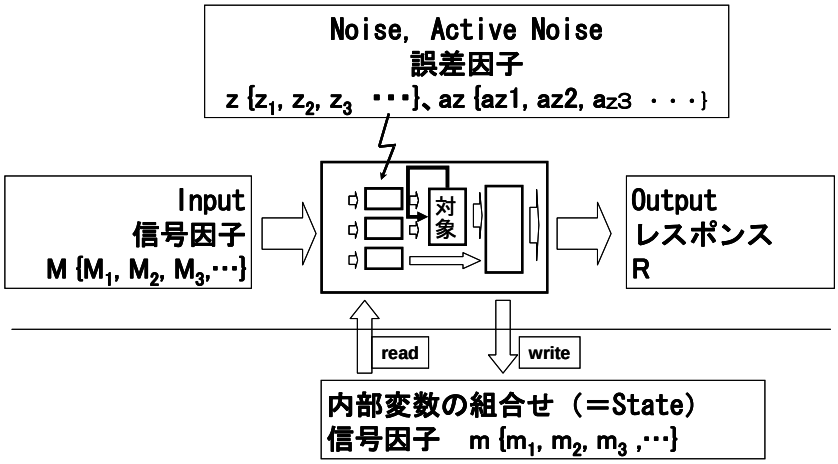


図 3-3 ラルフチャート

定義：ソフトウェアの操作 f は、以下の通りに定義される。

ソフトウェアの動作を、出力 R への関数として考えると、

R ＝ f (Input, State, Noise, Active Noise)

であらわされる。Input はテスト対象システムへの入力、State はテスト対象システムの状態、Noise は入出力の関係を妨げる要因、Active Noise は悪意を持ったユーザによるシステム破壊要因（セキュリティの弱点を突いてシステムを破壊する行為など）を示す。

HAYST 法のテストモデルと Mills の「ボックス構造化手法」との相違点は、市場条件である Noise と Active Noise を明示的に書き込んでいる点にある。

従来からソフトウェアテストにおいて「意地悪テスト」と称し、負荷をかける、あるいは少ないリソースでテストをすることが行われている。ところがテスト対象のソフトウェア（部分）に対して、どのような「意地悪条件」が効くのかについて十分な検討は行われてこなかった。本来、ソフトウェアにはそのパーツごとに考慮しなければならない「意地悪条件」は異なるはずである。HAYST 法では、テスト設計段階で当該ソフトウェアが持っている弱点を狙ったノイズ、使用されるユーザ環境の違い、悪意を持ったユーザによる破壊行為を徹底的に洗い出してテストを実施する。

3.3.3 テスト実装

HAYST 法のテスト実装では、まずテスト設計で用いたラルフチャートを元に、因子・水準・禁則情報を洗い出し FL 表(Factor Level table)にまとめる。

次に選出した情報を用いて、直交表に割り付けを行う。なお、この割り付け作業は Visual C++で開発した Matrix Tester というツールを用いている。

Matrix Tester の特徴は、300 程度の因子数、そして 1 因子に最大 64 水準まで対応していることと、PICT や jenny といった pair-wise 系のツールが処理できない複雑な禁則へ対応していることにより、大規模システムへの割り付けが可能となっている点にある。直交表は、2 水準系強度 2 の L4～L256、3/5/7 水準系強度 3 の L81～L343 まで対応している。大きな直交表は、有限射影幾何を用いて独自に作成したものである。

これまでは一度作成した組み合わせテスト表は、その作成に多大な工数がかかるため、テスト設計のレビューはしても大きな作り直しはほとんど不可能であった。HAYST 法では、Matrix Tester というツールにより因子・水準レベルの変更はもちろんのこと、表の分割・統合といったテスト設計の再構築が簡単にできるようになっているため、レビュー結果をその場で反映させることができテスト実装の効率が上がっている。

3.3.4　テスト実施

HAYST 法のテスト実施においては、可能な限り生成したテストマトリクスを元に、データ駆動型のテストの自動化を実施する。例えば因子数が 50 で、L64 にテストが割り付いた場合、テストオペレータが入力する水準数は、50×64＝3,200 個にも達する。これに、一般に入力ミスの発生確率である 1.5%をかけると 48 件ものテストミスが生じる計算になる。自動化を行うことによって、これら入力によるテストミスを防止することができる。また、表形式であるためデータ駆動型の自動テストを行いやすい。

実行中のテストに欠陥が生じた場合、原因の調査が必要であるが、この時に直交表の性質を活用して欠陥の絞込みを行うことができる。直交表では組み合わせが同数回表れるので、任意の2因子間について同じ組み合わせが他の行に含まれているケースがほとんどである。

従って、欠陥が発生したテスト項目の組み合わせに着目し、それを他のテスト項目を実施することで欠陥を再現させ原因にたどり着くことができる。

3.4　活動の効果

HAYST 法を上流工程から適用することにより、お客様の観点でテスト対象を漏れなくダブりなく抽出することが可能となり、仕様の不備の確認が効率的に行えるようになった。またテスト分析結果が、直交表に割り付ける因子と直結するため組み合わせテストを効率的に実施可能となった。具体的には、テスト工数を増やすことなく 2 因子間組み合わせ網羅率を 30%から 80%～90%に向上させることにより、市場導入後の不具合を適用領域においてほとんど発生させないことを実現した。

現在は、組込み系だけではなく、受注開発においても HAYST 法を適用することで、お客様のデータ固有の問題を網羅的に作り出しテストを実施している。これにより従来発生していた異常データによるシステム不具合を防止することができている。

今後は、さらに適用領域を広げていくことが課題である。

参考文献

[1] IPA/SEC：『組込みソフトウェア開発プロセスガイド』, 2007, pp18-20.
[2] 吉澤正孝, 秋山浩一, 仙石太郎：『ソフトウェアテスト HAYST 法入門』, 2007

第4章　日本ユニシス株式会社の事例

高品質ソフトウェア開発の現状　－ソフトウェア検査への取り組みについて－

4.1　取り組みの背景

システムの品質を確保するためには、案件の提案・受注段階から開発そして運用・保守に至る、すべてのライフサイクルを通して中間成果物の品質を保証し、それを各工程で積み重ねていく必要がある。日本ユニシスグループ（以後、当社）では、“Quality First”という品質ビジョンのもと品質保証部門を設置し、これら全工程において品質の向上を追求してきた。案件を獲得する際には、プロジェクトとは独立した専門部隊が提案の内容、見積もりの妥当性をレビューし、プロジェクト開始に際しては計画の妥当性や体制の確認を、また開発途上では、EV（Earned Value）分析による進捗の確認や品質メトリクスによる品質の確認を行ってきた。現時点では、当社で大規模と呼ばれる全案件に対し、これらのレビューは実施されており、プロジェクトの失敗を早期に発見し、その対策を立てる上で大きく貢献してきた。これらの取り組みはどちらかといえば、プロセスの品質に重点を置いた活動であった。

これに対し、プロダクトの品質という観点から見た時、状況は少し異なる。プロダクトの品質とは、各工程での成果物である仕様書やプログラムの内容そのものの品質のことである。

従前プロダクトの品質に関しては、技術的に困難であると予想される特定プロジェクトに対して、都度それぞれの領域の専門家を集めたレビューチームを組織して品質確保にあたってきたが、大半のプロジェクトについてはプロジェクトの内部レビューなどに任され、品質保証部門のような第三者が、組織的に成果物の品質を検証し保証するまでには至っていなかった。

一方、当社ではかねてより品質保証を支える重要技術として、プロセス管理とともにテスト技術を重視し、数年前からインドのテスト専門会社であるSTAG Software Private Limited社（以下 STAG 社）の技術を導入し、各種プロジェクトに適用してきた。重要プロジェクトに対するテスト技術の適用が、プロダクトの品質改善に大きな効果のあることが実証されてきたため、今年度は要員を増強し品質保証部門内にテスト技術を用いたソフトウェア検査を専門とする部隊を立ち上げた。結果として、プロセスとプロダクトの両面からシステムの開発行為を組織として保証する体制を構築した。

本稿では、この“ソフトウェア検査部”の活動を紹介する。

4.1.1　プロセスの標準化

システムインテグレータにとっては、プロジェクトの破滅をいかに早期に発見するかが最重要課題である。品質問題は、案件の提案初期から始まっている。

当社では、作業の全工程を “受注獲得” “契約” “立ち上げ” “実行” “終了”の各フェーズに分けている（ビジネスプロセス）。さらに“実行”フェーズを、“要件定義” “論理設計” “物理設計” “プログラム開発（単体テストを含む）” “結合テスト” “システムテスト” “導入” “保守”工程に分割している(エンジニアリングプロセス)。各フェーズ、各工程の決められたタイミングで、品質保証部門による各種審査、検査、レビューが実施される。提案内容は適切か、見積りは妥当か、この体制/スケジュールで開発可能か、スケジュール通りに作業が進んでいるか、などが都度確認される。これらレビューの運営要領が社内標準プロセスとして規定され、運用されてきた。

各工程の成果物に記述すべき内容は、開発標準で詳細に定義されている。例えば、オープン系システムの開発では AtlasBase®（アトラスベース）*として体系化されており、各プロ

---

* 2009 年 6 月に当社が発表したシステム基盤。システム共通機能領域からインフラストラクチャ領域までを網羅している。

ジェクトはここで規定された内容に従い、各種成果物を作成する。
各プロジェクトが当たり前のこと（決められた手順や内容）を当たり前に実施していれば問題は少ないが守られないこともあるので、品質保証部門内には、それを是正する役割を持つ部署が存在する。必要な対策が講じられるまで、プロジェクトが次工程に進むことを差し止めることもある[1]。
これらの活動の結果、当社では大きな赤字を生むプロジェクトの数は従前に比べて大幅に減少した。
とはいうものの一部のプロジェクトでは、結合テストやシステムテストに入ってからも不具合が収束せず、計画工数を大幅に上回ることもあった。失敗プロジェクトの多くは、要件定義や論理設計が正しく行われていないことに起因しているが、その不具合をレビューで指摘できないとか、指摘できたとしても、その後の対応を怠ったことによるものであった。

4.1.2 テスト技術への取り組み

テスト技術そのものは古い技術であるが、日本で注目を浴び始めたのは 10 年ほど前からである。
当社では、2006 年からテスト専門部隊を設置しテスト技術を研究するとともに、その技術を実業務に適用してきた[2]。従前我々は、テストの必要性（このテストケースは必要である）を語ることはできたが、十分性（これらのテストケースで十分である）を語ることはできなかった。そのような状況で、当社は STAG 社を知った。STAG 社の STEM™[3] は、テスト技術を方法論として体系化したものであり、そこで述べられている各種原理原則やポテンシャルデフェクト（潜在的不具合†）の考え方は、テストの十分性を検証する手法として 1 つの指針を与えてくれた。
当社では、テスト技術をまずは単体テストに応用し、単体テストアセスメントを実施してきた[4][5]。
単体テストアセスメントでも多くの成果を上げることはできたが、

- 単体テストケース作成に関する不具合のかなりの部分は、上位仕様書の悪さに起因する
- プログラム開発工程の最終段階で単体テストアセスメントを実施し、多くの不具合を検出したとしても、その成果をプロジェクトに反映させられない場合もある（上流工程に問題のあるプロジェクトでは、単体テストケースを改善しても無意味な場合が多い）

などの問題点が明らかになり、上流工程での成果物品質をいかに高めるかに焦点が移ってきた。上流成果物の品質が良くなければ、下流工程のテストで品質向上を図っても限界があり設計書変更にさかのぼることが必要である。当然のことではあるが、「全仕様書の正しさと相互の整合性維持が品質保証の要」であることを改めて認識した。
そこで、上流工程の成果物（要件定義書や論理設計書など）を含め、すべての成果物を検査するとともに、テスト技術の高度化と普及活動を推進するため、新しくソフトウェア検査部を立ち上げ、一層の品質向上を図ることにした。品質を向上させることは、結果として顧客満足度向上や開発コスト削減、赤字撲滅につながる。
これにより、プロセスの品質とプロダクトの品質を組織として保証する体制ができ上がった。

† “潜在的不具合”の概念は STAG 社に限らず、各機関/企業でいろいろな名前で呼ばれている。

4.1.3　フロントローディングと W モデル型開発

フロントローディングとは、製造業において 30 年以上前に提唱されたビジネスプロセス改革の考え方である。製造工程やテスト工程で不具合が発見されると、その対策のためには設計工程にまでさかのぼることが必要な場合があり、多大な工数・コストがかかる。そのため後工程における不具合を事前に予測・検討し、その検討結果を事前に設計工程で織り込んでおき、後工程での不具合の発生を押さえるという考え方である。このような考え方は、ソフトウェア開発の世界でも同様の時期に Duke[6] や Beizer[7] 等によって提唱されており、近年、この考え方をシステム開発に取り込もうとする動きが各社で試行されている。W モデル型開発はこの流れを汲むものであり、システム開発にWモデル型開発を適用しようと試みる開発会社が増えている。当社のシステム開発に、フロントローディングの考え方を適用すると図 4-1 のようになる。

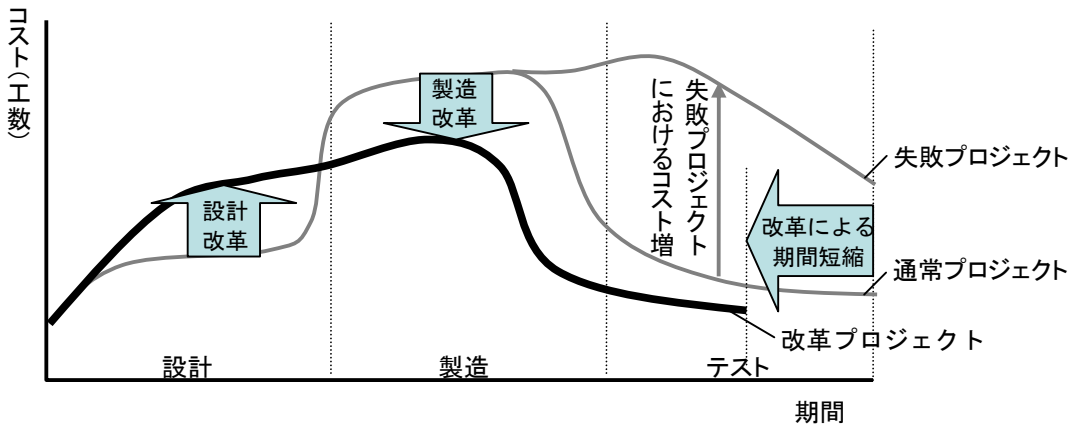


図 4-1　日本ユニシスにおけるフロントローディング

ここで“設計改革”とは、Wモデル型開発への取り組みや成果物検査により、上流工程での品質を確保することを意味する。当然設計工程でのコストは増えるが、後工程でのコストが減るため全体コストは減少し、また開発期間の短縮が見込める。

“製造改革”とは、製造の標準化を推進することによるコスト削減であるが、当社ではそれに加え、設計書の品質を上げることにより当社の 100%オフショア子会社である USOL ベトナムを活用できるのでコストメリットは大きい。正確でない仕様書は、言語の壁が高いベトナムオフショア開発にとっては致命的である。

4.2　検査体系の構築

この章では、当社におけるプロダクト検査の体系と、その体系を構築するに至った考え方を述べる。

4.2.1 体系構築のための基本的考え方

4.1.1 項で述べたように、当社の開発工程は要件定義→論理設計→・・・と進み、それぞれの工程における成果物は開発標準により詳細に定義されている。この開発標準にはWモデル型開発を取り込んでおり、それを推進中であるが、現時点では完全なWモデル型開発には至っていない。従って、検査体系構築にあたっては W モデル型開発の適用推進・支援を考慮するとともに、成果物検査においてはWモデル型開発の思想である予防指向を取り入れている。すなわち「その成果物（要件定義書や論理設計書）からテストシナリオが描けるか」や、「どこに不具合があるかを予測し、潜在的欠陥を早期に発見する」ことを念頭において

検査体系を構築した。

4.2.2　検査体系

当社における成果物の作成支援と検査体系を図 4-2 に示す。

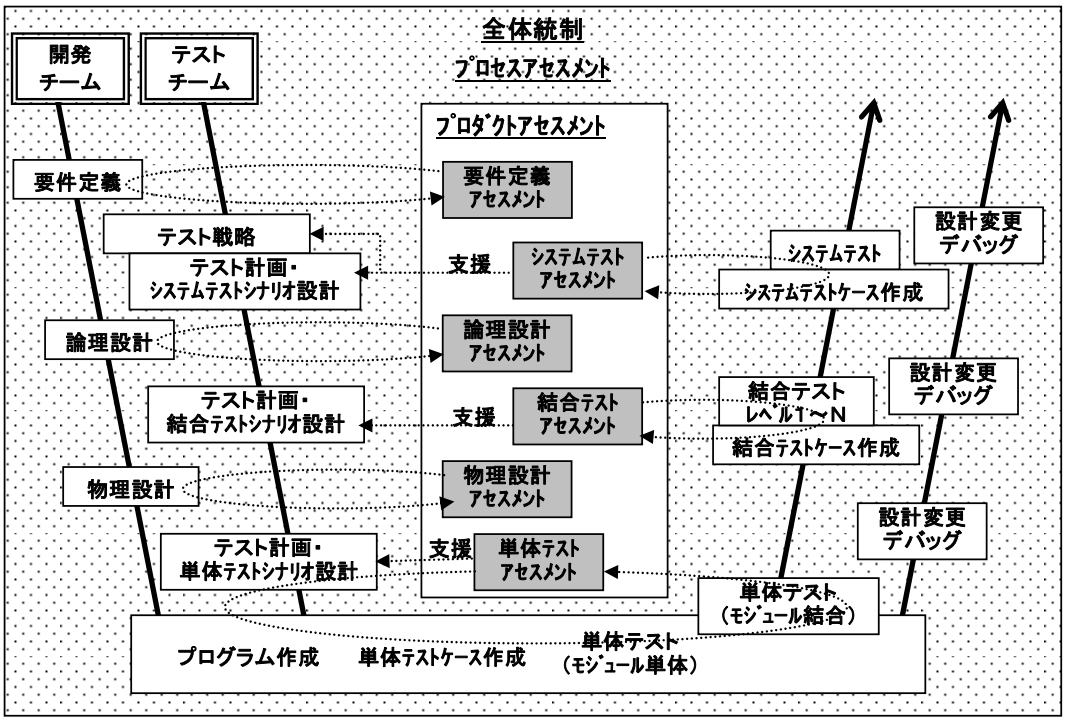


図 4-2　検査体系

図 4-2 において“××アセスメント”と網掛表記しているところで、それぞれの成果物を検査するとともに、その成果物の完成度を向上させるべくプロジェクトを支援する。

大規模開発では、上流工程でテスト戦略・計画を立案することは定着しつつあるが、上流工程でテストシナリオを設計するなどテストチームを分離するのは、試行が始まったばかりである。現時点では数プロジェクトで実績が出ている。

4.3　ソフトウェア検査部の役割

ソフトウェア検査部の役割は、文字通り“検査”であるが、検査するためには“基準”が必要であり、その基準を周知することが肝要である。また、検査で不具合を発見することは無駄が多い。フロントローディングの考え方によれば、検査で見つける前に“支援”することにより、未然に不具合の作りこみを防止する必要がある。

その結果、ソフトウェア検査部には“基準作成”と“支援”と“検査”の機能を持たせている。

4.3.1　基準の設定

検査をするためには基準が必要である。ここでは基準の設定について述べる。

4.1.1 項で述べたように、当社ではビジネスプロセスとエンジニアリングプロセスが存在し、各工程の成果物に記述すべき内容が詳細に定義されている。それをベースに以前から各種チェックリストが作成されてきたが、従前のものは経験をベースにした必要性に基づくチェックリストであった。そのチェック項目に加え、今回はテスト技術の観点としてステーク

ホルダの視点や、ISO/IEC9126 に準拠したソフトウェアの品質特性から考えられる潜在的不具合の項目などを追加することにより、チェックの網羅性を確保し検査の十分性を説明できるようにした。こうして全工程における基準を策定し、これをベースに支援と検査を実施している。チェックリストの策定にあたっては、極力属人性を排除することを念頭に、数値化、可視化、パターン化している。チェックリストの各項目に○△×を付けることにより、そのサマリ結果が品質特性のレーダチャートとして表示されるようなスコアシートを用意した。図 4-3 に論理設計工程におけるチェックリストとレーダチャートの一部を示す。

チェック項目	ﾁｪｯｸID	チェック内容	チェック観点（ガイドライン）	確認対象資料	必須	評価	指摘事項
1.目的と範囲						91.7	
1.1.目的	1.1.1	論理設計の目的が明確か	・プロジェクト管理計画書に論理設計の目的が明記されていること ・目的が「外部仕様の確定」となっていること 例：要件定義書で定義されたシステム化・・・	プロジェクト管理計画書		△	NUL社内標準の論理設計とは異なる。概要設計（客先工程名）を論理設計と位置づけており、次工程に外部設計工程がある。
1.2.対象範囲	1.2.1	システム化境界が明確か	・システム化境界（範囲）が明確であること	システム全体図 システム方式設計書		○	改造開発であり、現行と大きな変更なし。
	1.2.2						
1.3.成果物	・						
	・						
2.機能定義						92.6	
2.1.業務シナリオ	2.1.1	業務全体が分析され 整理されているか	・システム全体の概要図があり、サブシステム構成、および外部システムとの境界が明確に識別できること ・・・	システム全体図 システム化業務一覧 ユースケース一覧 要件定義書		○	
	2.1.2					○	
2.2.画面機能定義	2.2.1			画面一覧		×	ID管理されていない。

	チェック項目	評価対象項目数	○	△	×	－	達成率
1	目的と範囲						91.7
2	機能定義						92.6
3	データ定義						91.7
4	ｱｰｷﾃｸﾁｬ定義						87.0
5	運用／移行						71.4
6	要件網羅性						62.5
7	設計書品質						85.7
	計						86.6

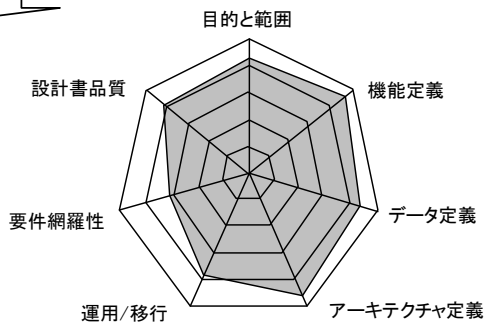


図 4-3　チェックリストとレーダチャート

またプログラム開発部分を協力企業に発注する際には、開発環境ごとに取り決めたツールを使い、静的検査や複雑度、網羅率を受け入れ基準として設定し、通知することを一部のプロジェクトで実施し始めた。

4.3.2　支援

品質は検査により作り込まれるものではない。前述したように、できあがった物を検査し、是正を促すのは無駄が多い。当社では支援と検査を対でとらえている。各工程の初期段階で、その工程の成果物作成方法を助言し、成果物が少し出来上がった段階でその成果物を評価することにより、そのまずさを指摘し、それ以降の作業でそのまずさを改善してもらうようにしている（図 4-4）。

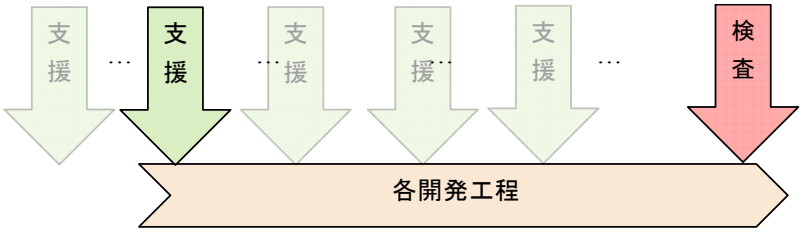


図 4-4　支援のタイミング

要件定義書や論理設計書など従前から存在する仕様書については、書き始めた初期のものを取り寄せ、評価し、改善点を指摘する。テスト工程の支援としては、協力会社要員を含めたプロジェクト要員を対象に、テスト技術をもとにテストケースの作成方法を事前に指導しており、その効果は大きい。代表的な支援であるテスト戦略書作成支援、単体テスト仕様書作成支援等について次項で述べる。

(1) テスト戦略書作成支援

W モデル型開発を推進するためには、テスト戦略やテスト計画を早い段階で策定することが不可欠であり、要件定義工程の途中からこれら成果物の作成を支援する。システムの成立ちをベースに（why)、テストの全体戦略（どの工程で何をテストするか/where、what)、戦術（何を使ってどのように/how)、実行計画（誰がいつ/who、when）を作る。こうして、テストの目的、対象者、範囲、方針、リスク、テストレベル、テストタイプ、テスト技法、メトリクス、テスト自動化、テスト合否基準、スケジュールなどが確定する。開発の初期段階で、品質特性に対する検査工程と管理方法を決めるのである。これにより各種要件（特にシステム境界、非機能要件、移行方針など）の確定や、以降の工程で必要となるテストのアクティビティ、リソースが明確になる。

既に、いくつかのプロジェクトで実施され大きな効果を上げている。

(2) 単体テスト仕様書作成支援

当社では、物理設計やプログラム開発（単体テストを含む）については、協力企業に依頼することも多い。単体テストアセスメントについては、参考文献[5](を参照)で詳しく述べているが、この内容を事前に協力企業に伝え、その後協力企業が作成してきた初期の成果物（物理設計書と単体テスト仕様書）を評価し、テストの網羅性が不足していることを指摘することにより、その後の成果物品質向上につなぐことを単体テスト段階で実施している。このようにテストの網羅性を高めて協力企業から納品してもらう効果も大きい。

また、いわゆるテストファースト（実装コードを書く前にテストコードを書く）も推進している。

(3) 教育

上記テスト技術に関する教育を、プロジェクト開始時に社内外プロジェクト要員を対象に実施するとともに、一般教育用として各種コースを用意している

・e-Learning としてテスト基礎コースを用意
・Off-JT（集合教育）としてテスト技術各種コースを用意

(4) 知財化

知財化も支援の 1 つである。成功モデルを社内の知財データベースに登録するとともに、テスト技法やテストに関する話題をホームページ上に掲載し、社内展開を図っている。

4.3.3 検査

ここでは、検査の内容と検査のプロセスについて述べる。

(1) 検査項目

仕様書の検査で重要なことは、Verification（検証）＆ Validation（妥当性確認）、さらに記述漏れや矛盾をなくすこと、あいまい性を排除することである。

V&V を検証するために、要求追跡検証マトリクス（RTVM： Requirement Traceability Verification Matrix）を用いている。仕様書から設計要素を抽出し、それが各工程に正しく引き継がれているかマトリクスを使って検証する。

プロジェクトの当事者は、背景や暗黙値を理解しているため往々にして書き漏らすことが多い。また同様の理由で、漏れやあいまいさを内部レビューでチェックしきれないことも多い。これを第三者の目でチェックすることは意義深い。要件定義書や論理設計書を検査するとき、「これでテストシナリオが描けるか？」と考えながら仕様書を読み進めると、たくさんの漏れを見つけることができる。確かに、その段階でそこまで書く必要があるかと議論になることもあるが、それが書かれていないということをその段階で認識することが重要である。少なくとも認識して課題管理表に書きとめ、早期にその課題を解決するように努めることで問題の先送りを防ぐことができる。

テストアセスメントに関しては、テスト密度よりも網羅性の確保に重点を置き、テスト設計技法を適用してテストの網羅性を検証している。

(2) 検査プロセス

4.1.1項で述べたように、当社では各工程の終了時に次工程に進んでよいかを判定するための第三者品質保証レビュー（QAR）を実施する。そのタイミングに合わせて、プロダクトの品質を検査する。要件定義書検査例を図 4-5に示す。検査期間はおおむね約2週間である。最初の１週間で検査し、報告書を作成する。その内容をプロジェクト側に伝えて認識を合わせ、各種やりとりの後、最終的な報告書に仕上げるのにまた１週間かける。この期間は目安であり、超短期で実施することもあるし、もっと長くかける場合もある。QARのタイミングに合わせられない場合もしばしば起こり得る。

報告書には品質判定とともに、指摘事項、改善方法を提示する。検査をスムーズに進めるためには現場の協力が不可欠である。

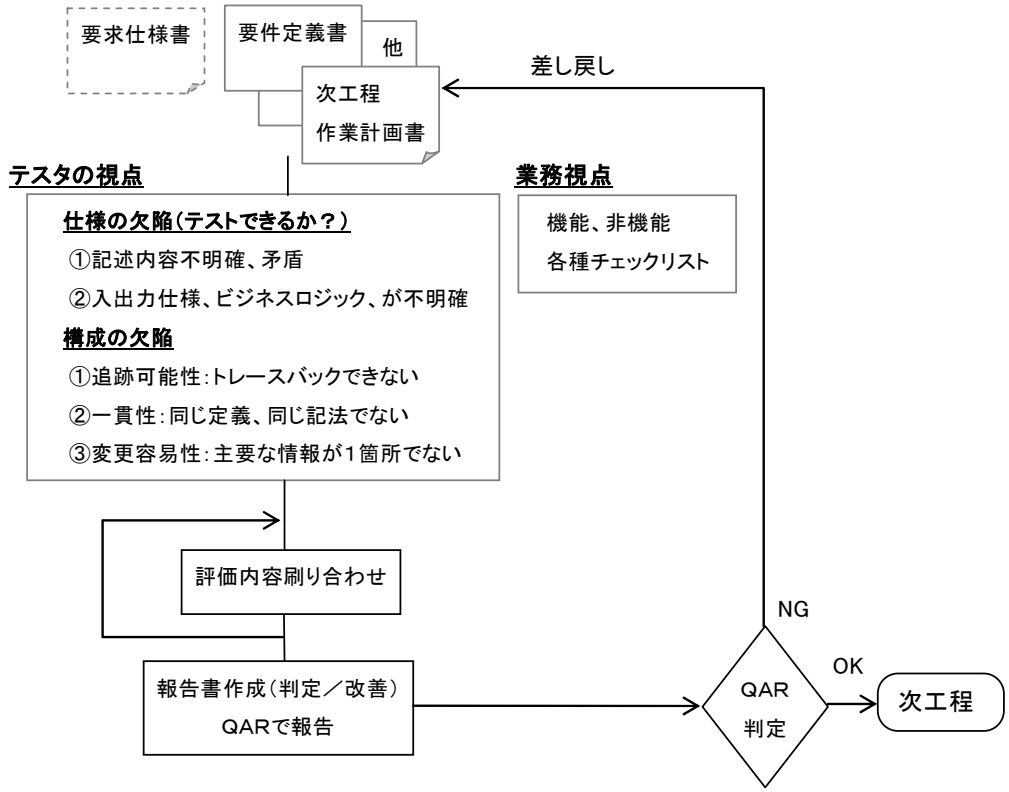


図 4-5　要件定義検査

指摘事項の対応を条件に次工程への進行を許可する場合、その対応については “課題管理システム‡”で追跡フォローする。

図 4-5 における“QAR 判定”では、プロダクトの品質だけではなく各種レビューの結果を用いて総合的に判定される。

4.4 アセスメントの実際

この章では実際にどのような評価を下し、どのような指摘をしているのかを紹介する。

現在、アセスメントの対象としているのは大規模システム（の一部）である。当然のことながら、大規模システムではその仕様書は膨大であり、全数検査するのは不可能である。設計書アセスメントであれ、テストアセスメントであれ、プロジェクト側と相談して重要機能部分をサンプリングすることになる。

4.4.1 設計書アセスメント

アセスメントを実施したら最終的には評価を下す。評価基準としては、
5: 優、4: 良、4: 可/一部見直し必要、2: 不可/大幅な見直し必要、1: 不可/全体見直し必要、の5段階を用意している。1、2の場合は、その工程の成果物基準を満たしていないことを意味している。指摘事項に対する追加・修正作業を追跡フォローする。3、4の場合は、ある程度基準を満たしているとはいうものの、1、2の場合と同様、指摘事項については追跡フォローする。

報告書本紙には評価結果の他に、全体状況、特筆すべき個々の状況、課題、提言が含まれ

‡ 品質保証部門が運営するシステムで、課題、担当者、対応期日などが管理され、時期が近づくと自動的に関係者にメールが配信される。対応が済むまで追跡される。

る。本紙の他に詳細な指摘事項などを添付する。わずかなサンプリング量であっても、指摘項目が数百に達する場合もある。もちろん、その中には重要なものから些細なものまで含まれる。プロジェクト側にその対応方法や時期を回答してもらう。これを2、3度繰り返し、品質レベルを2→3→4と上げて行く場合もある。

表 4-1に指摘事項一覧のサンプルを示す。

表 4-1　指摘事項一覧

指摘No	指摘箇所（章番号）	指摘内容	改善案	重要度
1		・・・	・・・	
2	1.5画面一覧 1.6帳票一覧 1.7外部接続一覧 1.8バッチ一覧	①機能一覧がない。（要件定義書の章番号で代替） ②画面一覧は、機能（要件定義書の章番号）との関連を書いてあるが、帳票、外部接続、バッチは機能との関連が分からない。 ③各一覧内では、項番として付番されているが、画面、帳票、・・・の識別ができない。（論理設計へのトレーサビリティが困難）	①機能一覧を作り付番 ②帳票、外部接続、バッチと機能との関連の記述 ③論理設計で混乱しないように、画面、帳票、・・・一覧の付番見直し。	B
3		・・・	・・・	

現在アセスメントの対象になっているのは大規模案件なので、この報告書は担当プロジェクトマネージャ（PM）、PM の上司、担当営業、営業上司に報告され、システムおよび営業の本部長クラスにまで写しが届けられる。

4.4.2 テストアセスメント

テストアセスメントでは、テストケースが設計書の内容を網羅できているかを検査する。同値分割や境界値分析を行い過不足を指摘する。システムテストなどでは業務フローからデシジョンテーブル等を作成し、その網羅性を確認する。これもスコア化され、設計書アセスメント同様、プロジェクト側関係者に報告される。

4.5 実績

約 1 年間、各種検査を実施してきたが、それぞれの工程におけるドキュメントの正確度について、スコアシートから最高値、平均、最低値などを%で表示することができるようになった。今後この精度を高めていく必要がある。

また、品質確保に関し、色々な課題が見えてきた。

4.5.1　共通課題

要件定義工程では、顧客要求との整合性不明、業務要件定義不足、システム化範囲不明、概念 DB 定義不足などが見られる。これらがあると、後工程でコストオーバーが発生する。上流工程の検査により、これを明確化し極力排除している。

プロジェクト要員のみが知る暗黙知が多く、開発標準化が不足している。これではオフショア開発を活用できないし、緊急の場合には要員追加も困難である。これに対しては、用語集やビジネスルールの明文化、成果物標準化を推進している。

その他、ドキュメント間の不整合や冗長性の問題、テスト設計技法の未活用など、改善しなければならない問題は多い。

4.5.2 費用対効果

4 .1 節で述べたように、当社では大規模と言われるプロジェクトについては、全件その案

件が発生した時点から、その状況は追跡されている。しかしながら、成果物検査については、まだ全件対応には至らず、特に注意を要する案件に対してのみ実施している状況である。活動を始めて現時点で約 1 年経過したが、大規模プロジェクトが検査の対象なので、我々が上流工程から支援・検査したプロジェクトで完結したものはまだない。1 案件・1 工程の検査で指摘する数は当然プロジェクトごとに異なるが、100 項目以上に及ぶことも多い。この効果を具体的数値として把握するため、世に知られたコストモデル[8][9] 等を参考に、表 4-2 に示す効果モデルを作成した。

表 4-2 検査による効果（総利益を人月で表示）

工程	要件定義			論理設計			物理設計			単体テスト			結合テスト			システムテスト		
指摘項目重要度	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
重要度別不具合指摘数	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15	n16	n17	n18
1 個発見あたり利益（人時）	100	50	10	50	25	5	20	10	2	20	10	2	40	20	4	60	30	6
活動による総利益（人月）	Y=(100n1+50n2+10n3+50n4+25n5+5n6+20n7+10n8+2n9+20n10+10n11+2n12+40n13+20n14+4n15+60n16+30n17+6n18)/140																	

検査により要件定義工程で重要度 A に相当する不具合を指摘できれば、1 件につきプロジェクトは 100 人時の利益を得ると仮定する。同工程で重要度 B の指摘なら 50 人時の利益、同様に単体テスト工程で重要度 C の指摘なら 2 人時の利益を得ることを示している。その指摘事項総数から、本活動の総利益 Y を表 4-2 のように算出した。

この利益とそれにかけたコスト（人件費）を比較することにより費用対効果を算出したところ、検査以外の支援工数を差し引いても余りある効果を得たことが確認できた。本活動が軌道にのれば、効果はさらに大きくなると考えられる。

4.6 今後の取り組み課題

本活動における課題を以下に述べる。

・ 検査の量

現時点では、成果物検査の対象は大規模プロジェクトの一部であり、かつ対象プロジェクトでも検査する部分は重要機能とはいえ、ごく一部なので、検査できない部分の方が圧倒的に多い。いかに少ないサンプリングで、より正確な評価を下せるかを見極める必要がある。また、我々の指摘した事項をプロジェクト側で横展開し、自律的に品質向上を図ることが技術力の底上げにつながる。さらに、ここで得た成果を中小のプロジェクトにも展開する必要がある。その手段はやはり標準化の推進と教育であり、これを充実させたい。

・ 検査の質

各種成果物を検査するが、当然のことながら、その正確性を完璧に保証することなどできない。また、例えば要件定義書の場合、そこに書かれた要件が正しいということと、それで顧客の業務が回るかどうかということとは別の問題である。検査に従事する要員はその業務の専門家ではないので、そこまで踏み込めない。完璧に保証出来ないことを承知で、できるだけ属人性を排した検査手段を模索しなければいけない。

・ 費用対効果

4.5 節で述べたコストモデルの各原単位は、まだ精度が不十分である。さらに検査数を増やし、精度を上げていく必要がある。

・ 全体的な問題
　基準の設定と検査、支援のバランスを組織としてどうとるか、さらに検査の基礎技術をどのように蓄積していくかを今後も考えて行きたい。

当たり前のことを当たり前にやれば品質はついてくる。納期に追われると、当たり前のことが当たり前にできなくなる。もちろん見積りが困難なこともあるし、顧客との関係で初めから厳しいプロジェクトがあるのは事実だが、納期に追われるのは問題に気づかなかったか、その問題を先送りしたことによることも多い。そのために、結合テストやシステムテスト時に無駄なコストが発生する。検査に頼ることなく、各プロジェクトが自律的にフロントローディングできるような全社の技術力底上げとプロセス構築（CMMI レベル 4 や 5）を目指したい。

参考文献

[1] 服部克己：「ソフトウェア品質保証の考え方と実践」，ユニシス技報 99 号 2009 年 2 月.
[2] 大塚俊章、荻野富二夫：「ソフトウェアテスト技術」，ユニシス技報 93 号 2007 年 8 月.
[3] T.Ashok:「STEM2.0 －クリーンなソフトウェアを生み出す科学的で規律のある方法－」，ユニシス技報 99 号 2009 年 2 月.
[4] 飯田志津夫：「テストアセスメントによる品質向上への取り組み」，SEC journal No.13 (2008).
[5] 沖汐大志：「テストアセスメントの効果と課題」，SEC journal No.15 (2008).
[6] M.O.Duke：「Testing in a complex systems environment」，IBM Systems Journal Vol.14 No.4 (1975).
[7] B.Beizer:「Software Testing Techniques」，Van Nostrand Reinhold, New York, 1983.
[8] Boehm and V. Basili:「Software Defect Reduction Top 10 List」，IEEE Computer, vol.34 January 2001.
[9] 大杉直樹、他：「ソフトウェア開発の「見える化」を支援するデータ分析力 ～エンピリカルアプローチによる既存データの有効活用～」，JISA 会報 No.80 2006 年 1 月.

第5章　株式会社日立製作所の事例
～ 信頼性レビューの高度化および保守フェーズのリスク管理強化による重要インフラシステムの事故撲滅 ～

5.1 取り組みの背景

重要インフラシステムで長時間ダウンなどの重要障害を発生させ、社会的に大きな影響を及ぼした経験を踏まえ、「社会問題になりうる大規模で高信頼性を要求されるシステム」については、従来から実施しているデザインレビューに加え「高信頼性重点管理システム」を導入することで、信頼性レビューの精度向上と信頼性要件のトレーサビリティを確保した。

一方、システム障害の原因は開発時に作り込まれたものだけでなく、システムが稼動した後システムに修正変更を加える保守作業時に作り込まれるものや、トラフィックの変動などにより顕在化するものも多い。このため保守フェーズでのリスク管理を強化し、システム稼動以降のシステム障害ポテンシャルの排除に努めてきた。

5.2 高信頼性重点管理システム

5.2.1 概要

新規開発および改造プロジェクトを対象に、開発の各フェーズにて開発プロジェクトとは独立した「高信頼性システム管理本部」を主査とした公式レビューを実施する。システム品質を阻害する要因を明確にし、その対策および是正状況を設計フェーズから保守フェーズまで継続してトレースすることにより、システム品質が実装され維持されることを確実にする。

5.2.2 対象システム

要求されるシステムの信頼度に応じて、システムの品質ランクを 5 段階で設定している。このうち管理対象システムは

- AAA ランクのシステム
- AA ランクの中から、経営幹部が指定したシステム

である。

表 5-1　品質ランクの定義

品質ランク	システムのプロフィール
AAA	社会インフラシステムのなかで特に指定するもの （障害の影響が社会問題化するもの）
AA	社会インフラシステム、企業の基幹システム （障害の影響が経済、国民の生活に影響する）
A	企業の基幹システム （障害の影響が企業内にとどまらない）
B	企業の基幹システム （障害の影響が企業内に限定される）
C	企業内部門システム

5.2.3 推進体制

高信頼性重点管理システムを確実に実行するために、事業部/開発プロジェクト（設計、品証）とは独立した専任部署（高信頼性システム管理本部）を設置した。

高信頼性システム管理本部の要員は専任者のほかに、各事業部の品質保証部門のシニアエンジニアを兼務者とした。また、各事業の設計部門からはシニアマネージャを認定レビュー

アとして選出した。

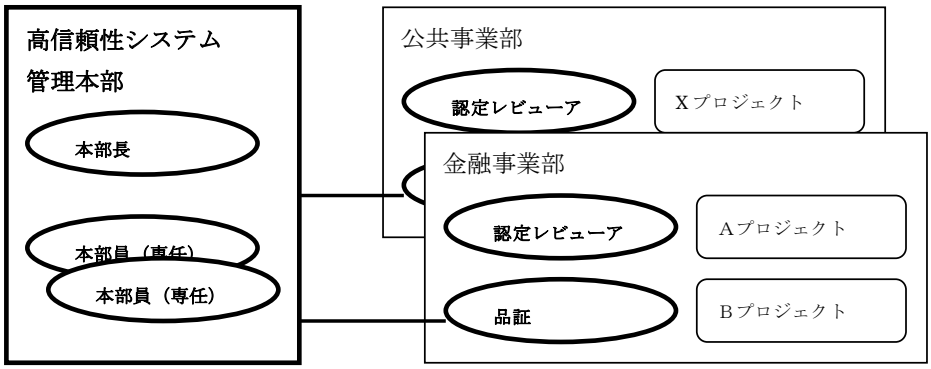


図 5-1　推進体制

5.2.4　管理プロセス

高信頼性システムの管理は、具体的には下記のメンバが出席する推進会議の形でフォローする。

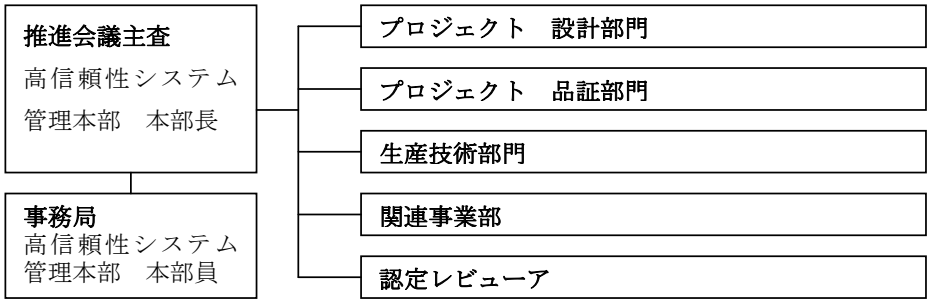


図 5-2　推進会議の参加者

(1) 推進会議の開催タイミング

推進会議は以下のタイミングで実施する。

システム分析	基本設計	詳細設計	製造	システムテスト	運用テスト	保守
	△ ①		△ ②	△ ③	△ ④	△ ⑤

① 要件実現のための問題点と対策レビュー
② システム品質達成状況の把握と対策レビュー
③ システムテスト実施状況の把握と対策レビュー
④ 残課題の把握と対策レビュー
⑤ 経年変化による見直しと対策レビュー

図 5-3　推進会議の実施時期と内容

以下、各推進会議でのレビュー観点について説明する。

(2) レビュー観点

① 要件実現のための問題点と対策レビュー

下記の観点からシステム要件を実現するための阻害要因、リスクを洗い出し、その対策を整理する。

性能（新製品のリスク、特異日、特異処理、負荷分散、排他処理・・・）
信頼性（新製品のリスク、MTTR、単一障害点の対応・・・）
拡張性（データ量/端末数、アプリケーション、外部接続・・・）
セキュリティ（セキュリティポリシー、アクセス制御、インターネット、暗号・・）
保守性（機能分割、処理の難易度、制御と業務処理の分離・・・）
DB 設計（DB 構造、DB 回復時間の評価、排他制御・・・）
標準化/ツール（開発言語、開発ツール、テストツール・・）
システムリソース（監視タイマ、リトライ、バッファ、プロセス数・・）
システム移行（移行時間、移行失敗時の戻し方式・・）
システム運用（自動化範囲と異常時運用、運用設計での顧客との分担・・）
プロジェクト体制

② システム品質達成状況の把握と対策レビュー

ソフトウェアの品質目標の達成度合い（テスト密度、不良密度など）を確認し、問題点と対策を明確にする

③ システムテスト実施状況の把握と対策レビュー

要件のシステムテストでの評価・検証状況を確認し、問題点と対策を明確にする

④ 残課題の把握と対策レビュー

稼動 1～2 ヶ月前に、残課題を整理し問題点と対策を明確にする

⑤ 経年変化による見直しと対策レビュー

稼動後に積み残した課題のフォローアップ、経年変化によるシステムの動作環境の変化や要件の変化を把握し、必要な対策を明確化する

5.2.5 効果

図 5-4 に、推進会議の実施状況の推移を示す。

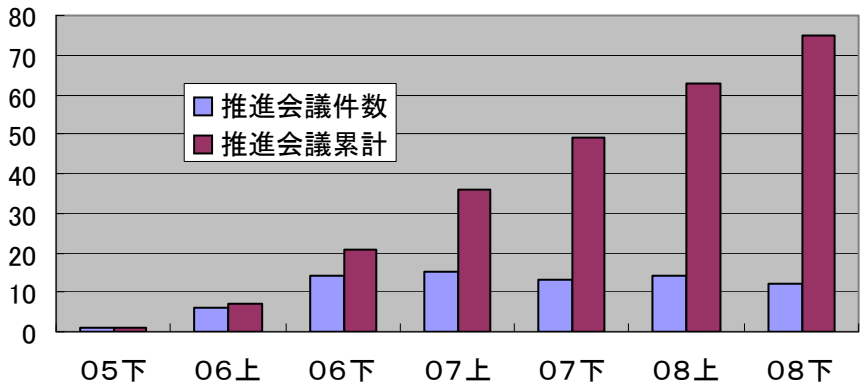


図 5-4 推進会議の実施件数（05 下を 1 とした比）

図 5-5 に、本手法導入前後での重要事故の発生状況を示す。

本手法を適用したシステムでは、本手法導入以降重要事故は発生しておらず、本手法の有効性が確認された。

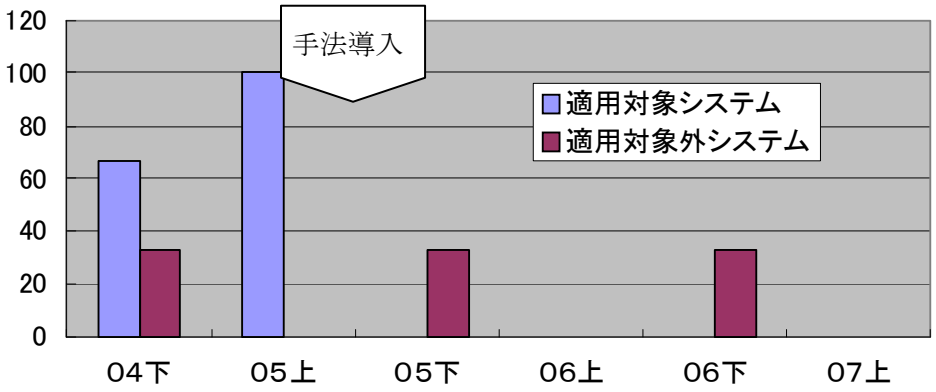


図 5-5 重要障害の発生状況（05 上を 100 とした比）

5.2.6 当該手法の優位点と課題

(1) 信頼性に特化し、過去の経験を集大成したレビュー観点に基づくレビューを実施することでレビュー精度を向上した。
(2) 事業部/開発プロジェクトとは、独立した専任推進部署（高信頼性システム管理本部）を設置することにより問題点の継続的なフォローが可能となり、トレーサビリティが確保できた。
(3) 固定した専任部署を置くことにより、プロジェクト解散後も継続して保守フェーズでの経年変化への対応がフォロー可能である。
(4) 同じく、専任部署設置により活動が継続的であり、ノウハウの集約、形式知化ができた。
(5) 専任部署の要員が限られており、管理対象システムが増加した場合にスケジュール調整が困難となった。
(6) 管理対象システムは、規模も大きく難易度も高いことが多い。また関連組織も多岐にわたる。このためプロジェクトが混乱するケースが多い。そのような状況下でも、確実に推進会議を遂行することが成否の鍵である。
この点でも、事業部/開発プロジェクトとは別に専任部署を設置することが有効と思われる。

5.3 保守フェーズのリスク管理強化

5.3.1 概要

システム障害の原因を分析すると、システム開発時に作り込まれたものだけでなく、システムが稼動した以降の保守作業で作り込まれたものも多い。開発時の問題が29%、保守作業を間違えたものが31%、運用時の問題が40%とした報告もある[1]。このため、稼動中のシステムを対象に「修正変更プロセス点検＆システムリスク点検」を実施し、保守フェーズでの修正変更および経年変化に伴うリスクを排除する。

5.3.2 対象システム

表 5-1で示したAランク以上の稼動中システム。効果が明らかとなったため、順次Bランクに適用拡大中。

5.3.3 推進体制

立上げ（現状分析、点検項目/点検プロセスの設計）については有期のタスクフォースに

て推進した。以降は専任部署は設置せず、各事業部の品質保証部門がプロモータとして推進中である。点検結果や得られたノウハウについて定期的な連絡会やイントラにて情報共有を図っている。

5.3.4 管理プロセス

弊社にて、628システムを対象に保守フェーズに起因した障害のポテンシャルを分析すると、次のようなリスク要因が洗い出された。

(1) システム修正変更にかかわるリスク
- 開発に比較し保守体制が脆弱（有識者の不在）
- 保守に関するプロセスがあいまい（変更の承認、検証、リリースの承認に関する顧客とベンダの役割分担）
- 開発に比較し検証環境が脆弱

(2) 経年変化に伴うリスク
- 保守体制（ベンダ側、顧客側）変更時にノウハウが継承されない
- 対象システムの保守性の低下（複雑さの増加、文書化が不十分）
- 外部要因（トラフィック、システム利用者/利用形態、要求される信頼度）の変化
- 予防保守（OSなどの既知不良のパッチなど）のポリシーが不明確

これらのリスク要因をベースに、保守フェーズの点検項目を設定し、以下の活動を推進している。

表 5-2 保守フェーズのリスク管理プロセス

No	アクティビティ	内容
1	修正変更プロセス点検＆システムリスク点検	修正変更プロセス点検項目、システムリスク点検項目にもとづき、保守部門により自己点検しプロセス成熟度をスコアリング
2	修正変更プロセス監査	上記点検結果を参考に、監査の必要ありと判断した場合に品質保証部門による監査を実施
3	障害発生システムの第三者監査	修正変更に起因した障害が発生した場合に監査を実施し、原因分析と再発防止策の水平展開を実施
4	ナレッジの蓄積と活用	得られたノウハウを点検項目や、常識集にフィードバック

5.3.5 効果

図 5-6 に、本手法導入以前（05 年度）を 100 とした修正変更件数と修正変更に起因して発生した障害件数の推移を示す。併せて、保守プロセスのプロセス成熟度（100 点満点）の推移を示す。

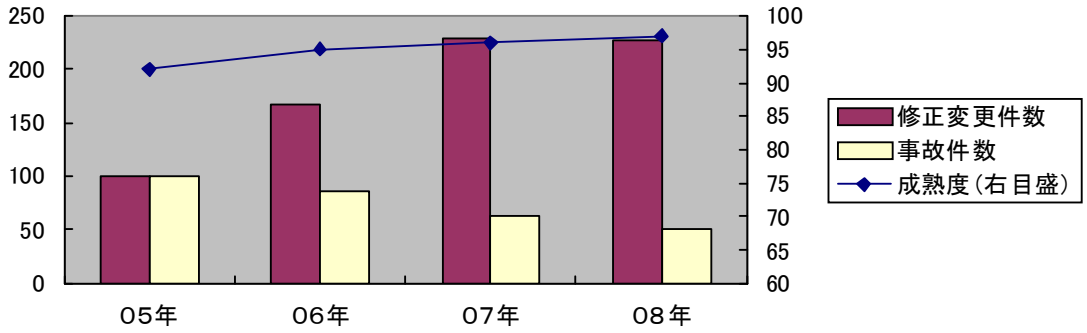


図 5-6　修正変更件数と障害件数の推移

5.3.6　当該手法の優位点と課題

(1) これまで開発フェーズでの信頼性向上施策が中心であったが、障害原因の分析結果に基づき保守フェーズの信頼性向上施策を導入することが効果的と思われた。
保守フェーズは、開発フェーズに比較し体制や開発プロセス、開発環境が脆弱であり、障害原因作り込みのポテンシャルは高い。このため保守フェーズでの改善効果は高いことが期待されるが、実績からも明らかとなった。

(2) しかしながら、保守フェーズの重要性は必ずしも（ユーザ側、ベンダ側ともに）認識されておらず、十分なリソース（保守体制や検証環境など）が割り当てられていないのが実態であり、ここで紹介した施策を推進する最大の妨げとなった。
また、保守フェーズでは顧客とベンダの共同作業となる場合も多く、顧客の協力が得られなければ改善が進まないことも多い。
保守プロセスの改善を推進するためには双方の理解と協力が不可欠である。

参考文献

[1] 中尾政之ほか：「重要インフラ情報システム信頼性研究会報告書」，（独）情報処理推進機構，2009 年 4 月.

第6章　富士通株式会社の事例
～　高信頼ソフトウェアの開発に関する上流工程での手法の紹介　～

6.1 取り組みの背景

IT ベンダとしてお客様の業務システム開発に携わっていると、要求品質に関するトラブルに遭遇することも多い。大きなトラブルに発展する典型的な例として、システム開発工程の上流工程に起因したものが挙げられる。例えば、業務の見直しがなされないままシステムの構築を開始し、無駄な開発工数を掛けて不適切なシステムを開発してしまうとか、要求される稼働条件を明確にしなかったために、運用テスト以降のフェーズでエンドユーザの了解を得られず、大幅な作り直しが必要になるなどのトラブルはよく経験し、また見聞きするところである。

プロジェクトの失敗原因という観点からは、いくつかの調査結果[1] が報告されており、おおむね 5 割以上が要件定義工程までの問題に起因しているといわれている。

このように、トラブルの原因として① 要件定義が十分出来ていない、② システムの外部仕様の検討が不十分なまま、それ以降のアプリケーション開発を進めてしまうことがあげられる。このような上流工程での問題発生は、プロジェクト失敗原因の多くを占める。しかもその問題に対する対策をとる工程が後ろになればなるほど、膨大な改修コスト増や納期遅延を引き起こすことになる。

弊社では、システム開発におけるトラブル発生を防ぐため各種の取り組みを進めている。本稿では、このような状況を踏まえて要件定義監査と外部仕様ドキュメント診断に絞って紹介する。

6.2 高信頼ソフトウェアにかかわる手法

6.2.1 要件定義監査

要件定義の作業において、何をどうすべきかの標準化を行った事例を報告する。

背景に述べたように、要件を適切に定義することは適切なシステム実現のキーポイントである。しかしながら、今まではコンサルタントや上流を実施する技術者の知見によらざるを得なかったというのが実態である。そのため成果にばらつきが生じているといえよう。

“ソフトウェア”の信頼性は、ソフトウェア開発に関する対策だけでは担保できない。ソフトウェアは“システム”の構成要素であり、システムは“業務”を構成する一部である。業務は“事業(ビジネス)”を遂行するための構成要素であり[2] 、この構造のなかで信頼性の定義、実現策等を議論する必要がある。すなわち「経営」における要求を実現するために「システム」に課せられた要求を把握し、そこで要求される信頼性を踏まえたシステムの実現に結びつけることが必要である。

要件定義においては、システムを求める人(経営者、エンドユーザ)とモノづくりの人(開発担当者)の間で、要求から要件への『変換』が行われる。この変換(橋渡し)には大きなギャップが存在し、溝を埋める為のプロセスが必要となる。例えば、利害関係者による要件定義、業務分解やシステム分解、業務やシステムの範囲と深さの設定、要求仕様書としてドキュメント化する、などである。

この構造を十分に踏まえて要件定義を正しく行っていないと、運用テストでのトラブル発生(一般に膨大な対応コストがかかる)や、稼働後のシステム評価での不十分な投資効果という結果になってしまい、信頼性の確保の点からも大きな問題となる。

要件定義には、仕様化技術、獲得分析技術、検証技術が必要である。本項では、仕様化技術の例として「要件定義書き方ガイドライン」を、検証技術の例として「第三者要件定義書診断」を紹介する。なお、ここでの説明範囲はいわゆる形式からみた品質担保の観点であり、

内容そのものに踏み込んだ品質については別の機会としたい[3]。

① 要件定義書き方ガイドライン
・このガイドラインでは、以下を記載・説明している
- 要件定義開始の前提条件(全体企画状況、プロジェクト企画状況、要件定義計画状況)
- 要件定義で作成するドキュメント一覧および要件要素間の関連
- 個々のドキュメントの記載項目の説明と記述サンプル

・弊社では、グループ会社を含む社内 SE 向けにこのガイドラインについて研修会を実施し、ドキュメント種類・項目とも充足率が向上するという成果を得ている。

② 第三者要件定義書診断
・この診断では、集合・整合・適合の観点からの確認を行っている
- 要件定義において、検討し決定すべき事項が必要十分に記載されているかを確認する(集合)
- 要件定義として記載された事項において相互に矛盾はないかを確認する(整合)
- システム化が企画された段階でのビジネス要件とシステムへの要求事項として整理された要件定義内容に矛盾がないか、実現性はあるかを確認する(適合)

・これらの観点からドキュメント要素ごとに評価し、レーダチャートで可視化して示している
・チェック観点のなかで、集合は比較的容易にかつ十分に確認できる。整合は手間がかかるが、それなりの確認を行うことができる。もっとも困難なのは適合であり、ビジネス要件からの連続性(トレーサビリティ)を踏まえた確認、モデル表現としての無矛盾性の確認などを行う必要がある[3]。

要件定義については、試行を踏まえてリファインし、現時点では以下のような状況が達成できている：

・大規模プロジェクトでの監査義務化
・要件をどのレベルまで書くか（決めるか）のレベル感の標準化
・何を先送りしているか(何がリスクか)の見える化
・お客様とのレビュー時の情報共有（必要事項が網羅されているかの判断など）

これらの結果、信頼性の向上への寄与とともに、プロジェクトの遅延率、コスト遵守率が大幅に改善されるという効果も得られている(QCD* が大幅改善されている)と判断している。

6.2.2 外部設計ドキュメント診断

外部設計において何をすべきか整理し、それを設計書類に記載する際の標準化を行った[4] [5]。

システムインテグレーションの現場では、要件定義・仕様定義のフェーズにおいて本来解消すべき課題を後工程へ先送りしてしまうという問題が発生しがちである(理由はいくつかあるが、形だけ取り繕って次工程に進まざるを得ないということもある)。

ここでの目的は、設計をきちんと行い、その後のソフト開発プロセスを工業化†して高い品質で効率よく開発を進めることである。今までのような属人的開発スタイルから、製造プロセス重視の開発スタイルへの変革が必要である。工業化の前提は、外部設計品質の確保であり、今までのようなアプローチでは(業務アプリの設計にはあいまい性が残るため)限界があるということである。そのため、システムの QCD 確保に必要な外部設計工程での設計内

* 本章では「QCD: スコープ･機能・品質、費用・資源、納期・時間」と定義
† 再現性のあるプロセスでの進捗を定量的に把握し、効率化し高品質化すること

容・レベルを担保した記述が要請される。

(1) 実プロジェクト状況はいかに

工業化実現に向けた現状調査のため、弊社のソフトウェア開発工場での開発実態を調査し、外部仕様の記述状況がプログラム作成以降にどのような影響を及ぼしているかを分析した。その結果、記述すべきと想定される項目数の 4 割強の項目しか充足されていないことが分かった。ひどいものでは 3 割以下、よく書けた外部設計書でも 6 割程度であった。継続調査の結果、外部設計の項目充足度が低いと詳細設計以降の開発工程において上流設計者への質疑が多発し、進捗に悪影響を及ぼしていることが分かった。また、設計時の考慮に不足があると、特にオフショア開発などにおいて稼働後、品質*の悪化に直結するような問題が埋め込まれてしまうことも明確になった。

後ろの工程（詳細設計工程以降）を工業化するためには、外部設計工程で、顧客・SE・開発者・運用保守の観点での設計が必要なことが整理された(257 設計項目)。その後の改版で、この観点は細分化されて、顧客・SE(外部設計)・開発者・SE(システムテスト)・運用・保守・プロジェクトマネジメントの 7 観点になり、設計項目も 297 項目に増えている。

(2) 評価ポイント、評価観点(テスト観点の導入、アプリ基盤観点の導入)の変遷

具体的なプロジェクトの状況を分析した結果、成功したプロジェクトでは外部設計書として記述できた項目の充足度が高く、失敗と見なされたプロジェクトではこの充足度が低いことが分かった。とはいえ、外部設計工程での検討項目をすべて満たすことを要求することは現実的ではない。粗い言い方であるが、おおむね 75%程度実現できていればその後の対応でもプロジェクトを成功裏に実現できるようである(設計にかかる負荷と開発にかかる負荷の合計が最小に近くなるのではないかと思われる)。これは、設計内容の骨格が定まっているので、上流工程設計者への質疑が発生しても時間的にも作業量的にもカバーできることによるものと思われる(逆に、カバーできないほどタイトなプロジェクトでは、充足度はもっと要求されるものと思われる)。

(3) 「先送り」への対応

充足度が低いプロジェクトでは、診断の結果を踏まえて次の工程に進む前に充足度を上げる手立てが必要である。これによって、問題を先送りし後工程でトラブルに対処する(それによって、費用も時間も人的資源も無駄遣いする)といった愚を避けることが可能になる。

外部設計におけるドキュメント診断は、具体的には以下の切り口で実施される。

①外部設計を開始できる条件が満たされているか、②外部設計ドキュメントの体系が適切か、③外部設計の設計項目が次工程（内部設計工程）を開始できる程度に充足されているか。いずれも 7 観点を軸とした充足度レーダチャートと、[不足点－想定される影響－推奨する対応策]からなるリストを診断結果として提供している。

プロジェクト規模によっては当診断を義務付けている例があり、利用の定着が図られている。また、外部設計工程でのレビューに際して、必要事項が網羅されているかの判断に用いることもできる。なお、要件定義ドキュメントの項目と外部設計ドキュメントの項目の関連も定義している。

これらにより、ソフトウェア開発の工業化に向けた第一歩が踏み出せたといえる。工業化には設計と製造の分離が必要であるが、ソフトウェア開発担当者から上流の設計者に質問を投げる頻度を減らすことで、この分離を実現できる。また単に機械的な分離を行うと、

* 開発が完了し本番稼動になってから、不都合が顕在化する品質

不明確な仕様を確認せずに実装してしまうことになり、不都合を埋め込んでしまうがそれを減らすことにも寄与している。

6.3 手法導入による費用対効果

(1) 手法導入にかかる負荷

- 診断側の負荷
  要件定義監査にかかる期間は対象資料の量によるが、典型的には 1 週間程度である。外部設計書診断は、要件定義監査より時間がかかる(10 日弱)。
- 受診側の負荷
  診断を受ける側にも負荷はかかる。例えば、事前ヒアリングを行うので打合せ負荷がかかる。また、資産(ライブラリ)がまとまっておらず、その整理に負荷がかかることが多い。さらに、資料を見るだけでなく聞かないと分からないことがあるため、質疑応答の負荷もかかる。

(2) 現場での負荷と効果の比較

受診側にも上記のような負荷がかかるが、以下のような効果が得られることから、大きな効果が得られていると判断している。

- 手戻りの減少
- 解決できていない箇所(リスク)が把握でき、次工程でリカバリできる(予定できる)
- 本来やっておかねばならない作業(資産管理、分かりやすい設計書記述など)を着実に実施する

(3) まとめ

要件定義監査においては、要件検討が不十分なまま次の工程に進んでしまうことがチェックできることが最大のメリットであろう。定義内容が十分であるかの判断には別の手法[3] が必要であることを述べたが、形式的な品質評価でも検討漏れを防ぐ観点からは大きな成果が得られる。

外部設計におけるドキュメント診断においても、検討漏れや決めるべき事項の先送りによる問題の多発を防止することができる。何よりも未解決事項(リスク)が明確になっているので、自工程および次工程での対応を予定できるメリットは大きい。

品質指標のうち、本項で述べた対策は、「機能性」に対する直接的な解となる。さらに、ユーザ要求・要件が適切に設定されているかも含めて確認することを始めており、「合目的性」をより広範囲に確認することを目指している。また「使用性」「効率性」に対しては、これらを意識した標準化を実現することで、副次的な解となろう。

6.4 当該手法の優位点と課題

(1) 改善された点、他の手法との違い

今までは要件定義においては、有識者レビューなどを行う例もあるが、ばらつきや漏れが発生しがちであった。また本項で説明した範囲では、まだ内容の適正さまで踏み込んで監査できているわけではないが、統一したレベルでの監査、漏れの少ない監査ができるようになったと言える。

外部設計ドキュメント診断は、設計と製造の分離(ソフト開発工場でのソフト製造)を目指した取り組みから必要性が認識され始めており、今までの外部設計の内容ではうまく機能できなかったものが改善されてきている。

(2) 課題と対策

要件定義の監査では、何といってもビジネス要件と整合性のあるシステム要件になっているかの監査が必要であり、これを実現する技術開発に取り組んでいる[3]。

外部設計診断においては、非機能要件に係る設計、総合テストに結び付けた設計などの観点で拡大していくことが必要と考えている。

参考文献

[1] 例えば、日経 BP 社編： 特集「成功率は 31.17%」、日経コンピュータ 2008/12/1 号.

[2] IPA SEC 編：SEC-BOOKS「経営者が参画する要求品質の確保 ～超上流から攻める IT 化の勘どころ～第 2 版」,オーム社, 2006 年 5 月.

[3] 日経 BP 社編： 「要件定義の新手法 5 階層に分けて漏れを防止 内容の品質を高める」、日経コンピュータ 2009/11/25 号、p.94.

[4] 中村 一仁、廣瀬 守克：「設計と製造に向けた外部設計書の定量的評価」、情報処理学会創立 50 周年記念（第 72 回）全国大会 5B-4.

[5] 日経 BP 社編：設計書の作り方、日経 SYSTEMS 2007 年 11 月号、pp21-41.

< 空欄 >

第7章　三菱電機株式会社の事例

～ ISV/IHV 起因の重大障害を予防する 非機能要件トレーサビリティ管理手法への取り組み ～

7.1 取り組みの背景と経緯

近年、特に社会の様々な場面で情報化が進展するなか、情報システムに対する要求はますます高度なものとなっている。特に、高可用性やリアルタイム性あるいは運用性の向上などの非機能要求（Non-Functional Requirements）の実現が強く求められるようになってきている。こうした社会要請を受け、2006 年 6 月には経済産業省が中心となって”情報システムの信頼性向上に関するガイドライン”を発行（2009 年 3 月に第二版発行）し、また国内主要ベンダが参加する非機能要求グレード検討会が中心となって非機能要求グレードの標準化を進める等、非機能要求に対する注目が高まりつつある。

我々情報システム開発を担う側もこうした社会要請に応え、我々の情報システム開発のなかでも顧客の抱える非機能要求に確実に応えていくことが強く求められてきている。高可用性などの顧客要求をいかに具体化し、この要求を確実に情報システム構築のなかで実現し、さらに顧客要求を満たすことを具体的に評価・確認して、当社の情報システムとして顧客の元に送り届けていくことが求められる状況になっている。

当社はこれまで、重大障害の再発防止活動から、重大障害につながる可能性のあるヒヤリハット障害情報の活用による予防活動に重点を移し、フィールドでの重大障害発生削減を実施し、一定の効果を上げて来た。しかし今日、情報システム構築にあたって、その実行基盤環境は高度化され、さらに ISV/IHV 製品を組み合わせて実行基盤環境が構築される方式が主流となってきた。それに伴い、重大障害発生傾向も変化しつつある。最近の重大障害の事例を分析したところ、導入した ISV/IHV 製品に関連する障害が多く発生し、全体に占める割合が他の障害と比較して増加傾向にあることがわかった。そこで、ISV/IHV 起因の重大障害を予防する非機能要件トレーサビリティ管理手法への取り組みを試行中である。

本章では、その実施方法ならびに整備し試行中のガイドの内容について説明する。

非機能要求を実現する際の考え方は、機能要求と同様に、上流の顧客要求で具体的に同定し情報システムの構成を選択しながら設計し、そして顧客要求を満たす製品が開発されたかを評価・試験する。この一連の活動のなかで非機能要求トレーサビリティ管理は、

- 非機能要求がどのように実現に向けて具体化され
- 仕様書のどこでどのように定義され
- 開発ライフサイクルを通じてどのように非機能要求の実現を確認し

ているかを追跡し、将来の非機能要求の変更に対し、既存のソフトウェア資産に対する影響範囲を見積もれるようにすることまでを目的としている。

本ガイドでは、非機能要求を具体的に開発し確認するための作業手順をプロセスとして示すとともに、非機能要求管理の成果物として作成する非機能要求トレーサビリティマトリクスについて、その形式と利用方法を示している。

7.2 高信頼情報システム開発の進め方

信頼性要求や性能要求などの非機能要求について、高度な要求を実現する必要がある高信頼な情報システムを構築する場合、情報システムの構築プロセスも、一般的な開発標準（当社では SPRINGAM という名称で標準化、手順化している）に加え、より非機能関連部分を具体化、詳細化した活動を加えていく必要があると判断した。同時にこの開発プロセスは、要求トレーサビリティ管理の際の成果物を定義するという観点からも重要な要素となる。

ここでは、高信頼情報システムの構築プロセスおよび実行基盤環境の評価・選定のプロセスについて、特にこれまでの開発標準からの差分を中心に記述する。

7.2.1　高信頼情報システムの構築プロセス

高信頼情報システムを構築する際のプロセスについて、概観を示す。

今日、情報システム構築にあたって、その実行基盤環境は高度化され、さらに ISV/IHV 製品を組み合わせて実行基盤環境が構築される方式が主流となっている。同時に、情報システムの高信頼化にあたって、可用性要件に応えうる機能を持った基盤環境（ISV/IHV 製品）を選定することも重要となってくる。さらに、これらの高信頼性実行基盤環境は、情報システム構築のコストにも大きなインパクトを与える。

このため、高信頼情報システムなど、高度な非機能要求に対応した情報システム構築では、

- 契約に先立って、外部から調達した ISV/IHV 製品の組み合わせで、どの程度の非機能要求が達成できるのかを明らかにすること
- 実現の見通しを持って、情報システムの実現機能、実現する非機能要求の具体化を顧客と実施し、受注金額を具体化すること

が重要となる。こうした要請に応えるため、情報システムの開発チームを大きく基盤チームと業務アプリケーション開発チームの2つのチームに分割して運用する方法を取るのは、1 つの方法である。基盤チームは、ISV/IHV 製品の組み合わせや必要な共通プラットフォーム SW の開発を担当する。業務アプリケーション開発チームは、顧客・エンドユーザが実際に利用する業務アプリケーションを開発する役割を持つ。

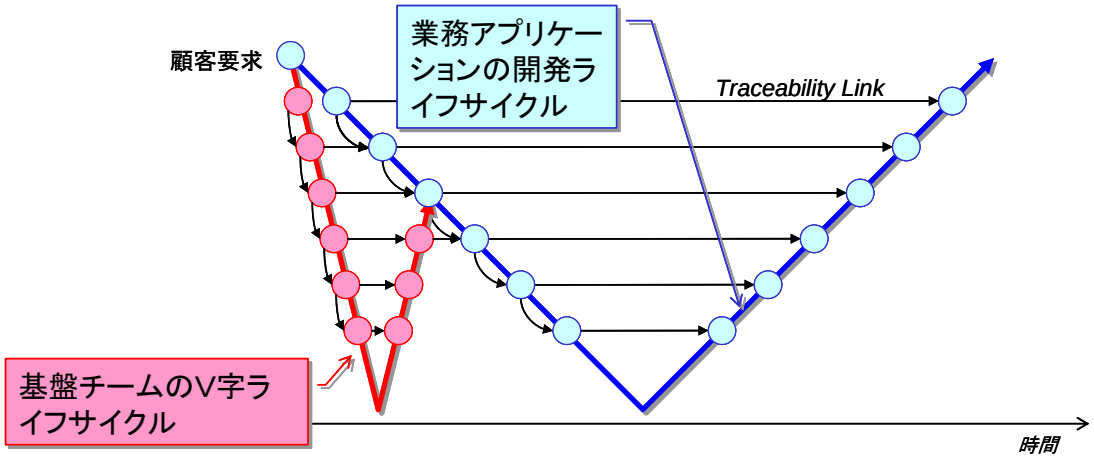


図 7-1　基盤チームと業務アプリケーション開発チーム

両者の開発の流れを、一般的なV字モデルにマッピングすると、図 7-1 のようになる。提案～契約～要求定義のフェーズでは、基盤チームが先行してシステムの実装方式を検討、非機能要求の達成に対する Feasibility Study を行う。これを受けて、業務アプリケーション開発チームは要求仕様の確定と契約に向けた見積りを行い、業務アプリケーション群の開発を進める。一方試験では、システム試験から後のフェーズは、業務アプリケーション開発チームと基盤チームは合流して、共同して作業を進める。

この作業の流れを、SPRINGAMの開発プロセスにマッピングすると、図7-2のようになる。

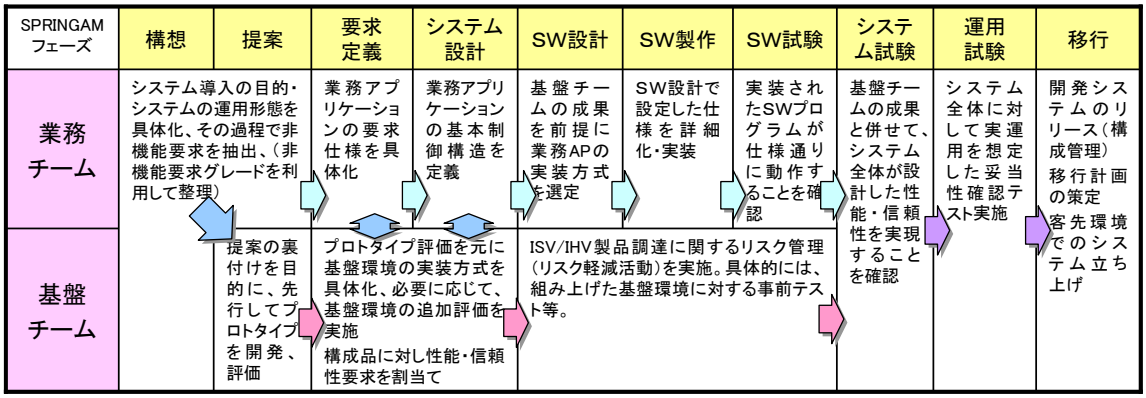


図 7-2　業務チームと基盤チームの役割分担

① 構想フェーズ

業務アプリケーション開発チーム、基盤チームが協力して情報システムの構想を具体化する。この時点で、顧客の業務遂行に必要と判断される非機能要求を、非機能要求項目リストを利用して可能な範囲で具体化する。

② 提案フェーズ

業務アプリケーションチームが機能要求、非機能要求を具体化するのと並行して、基盤チームが提案する情報システムのシステム構成を具体化し、必要に応じて実行基盤環境の非機能要求への対応度を評価する。この評価は、例えば既存の情報システムの拡張開発など、既に運用経験のある基盤環境を利用する範囲では、机上での検討で一定の効果を得ることができる。一方、新しい実行基盤環境を利用する必要がある場合は、事前にプロトタイプを構築し、特に業務面から重要な機能をサンプルに性能評価、信頼性評価を実施することになる。

提案フェーズが終わった段階で、顧客との契約が行われる。この段階で、プロジェクトが負うことのできるリスク対策費なども具体化されるため、

- 特に顧客の非機能要求のうち実現性に疑問があるもの、リスクが大きいものを識別し、提案フェーズのなかで先行して性能評価、信頼性評価を実施し、顧客予算に応じた非機能要求の実現程度（メトリクスの目標値）を設定、提案していくこと
- 将来の情報システムの拡張計画を念頭において、非機能要求の実現程度を提案していくこと
- 特に実行基盤に関しては、複数の実現方法を選択肢として用意し、そのなかで妥当性の高いものを選択すること

が重要となる。同時に、この段階での実行基盤環境の実現性評価は、短期間の間に非機能要求の実現性を確認することが求められる。従って、特に実現上のリスクが高い部分を限定して行評価を行うことが重要である。

③ 要求定義フェーズ

このフェーズでは、業務アプリケーション開発チームは、機能要求や非機能要求を、顧客、エンドユーザとともに具体化する。この段階では、顧客との情報システム開発にかかわる契約が終了しているため、プロジェクトの開発費がほぼ具体化していると考えてよい。従って、情報システムとして開発する機能の具体化と同時に、非機能要求については、②での評価結果を念頭に、過大な要求を受け付けないようにコントロールすることが求められる。

④ システム設計フェーズ

このフェーズでは、開発する情報システムの実現方式、システム構成の具体化が行われ

る。ここで採用する実行基盤環境が最終的に選択され、非機能要求に対する実現性も担保されることになる。もし、業務アプリケーション開発チームが定義した要求に変更が加わり、非機能要求の実現目標値が変更された場合、これは基盤環境上で実現可能性を評価する活動を改めて行う必要がある。

また、業務アプリケーション開発チームが開発するアプリケーションの基本アーキテクチャが具体化するため、この段階で目標の機能、性能を、情報システムの構成要素に割り振る活動が具体化する。各システム構成要素(例えば DBMS など)にどの程度の性能、信頼性が要求されるかを具体的に割り振っていくと良い。このように非機能要求の実現方法を具体化し、情報システムの構成要素に割り振っておくのは、後の情報システム統合に際してシステム構成品毎に性能や信頼性要求を満たすか、あるいはセキュリティ要件などを満たすかを単独でテスト可能とするためである。

⑤ SW 設計～SW 試験フェーズ

このフェーズでは、④の段階で識別された（定義された）情報システムを構成する個々のソフトウェアについて、設計、製造、試験が行われる。大規模なシステムになれば、外部委託先を利用して並行に作業が進むことになる。開発の主体は、業務アプリケーション開発チームとなる。開発された個々のソフトウェアが、ソフトウェアに割り当てられた機能要求、非機能要求を満たしているかを個別に試験し、ソフトウェア設計の妥当性を確認する。

一方このフェーズでは、基盤チーム側は実行基盤環境が期待通りの機能、性能、信頼性などの品質特性を満たすことを確認すると良い。特に業務アプリケーション開発チームに対して、実行基盤環境がどのように動作するか、また業務アプリケーションを構成する個々のソフトウェアからどのように利用すると良いか（実行基盤環境が提供する API をどのように利用すると良いか）等の情報を、業務アプリケーション開発チームに提供する。また、ソフトウェア試験のための試験仕様書レビューに参加し、テスト方法やテスト基準の妥当性を確認するという役割を負う。

⑥ システム試験フェーズ～移行フェーズ

業務アプリケーション開発チームが開発し、ソフトウェア試験で要求された機能や非機能要求を満たしていることを確認したらシステム試験が始まる。この段階で、実行基盤開発チームと業務アプリケーション開発チームは協力して、システム試験、運用試験などの作業を進める。

特にこのフェーズでは、顧客側から改めて非機能要求に対する新規の要求が提出される可能性がある。また、設計フェーズでは想定していなかった実装方法の採用や、実運用環境に持ち込んではじめて明らかになる情報システムの運用項目などもある（例えば、ネットワークのトラフィックなど）。実行基盤環境の開発チームは、これらの不測の事態が発生したときに、ISV/IHV 製品の設定方法等を調整し対応策を検討する役割を担う。また、すべての業務アプリケーションが実装された状況で、実行基盤環境と併せたテストを行い、当初設定した非機能要求の目標値が達成されているかどうかを確認する。

7.2.2 実行基盤環境の評価・選定のプロセス

7.2.1 項で示したように、基盤環境チームと業務アプリケーション開発チームが協調して作業を進める体制をひくことで、特に高信頼性や高性能を求められる情報システムの構築がより確実に行えるようになる。一方、基盤チーム側が担当する実行基盤環境の評価は、SPRINGAM の開発プロセスのなかでは明示的に言及されていない。ここでは、この基盤チーム側が担当する実行基盤環境の評価プロセスを具体的に示す。ここで示す作業プロセスは、全体のなかでは、図 7-3 に示す活動に位置づけられるものである。

この部分の作業の全体的な流れは、図 7-4 に示すとおりとなる。

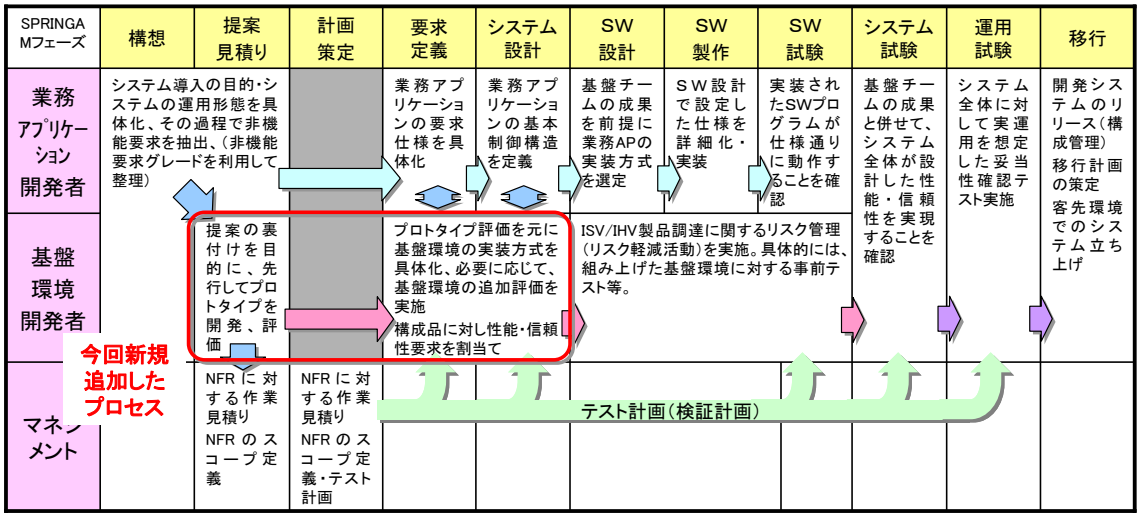


図 7-3　実行基盤環境の評価・選定プロセス

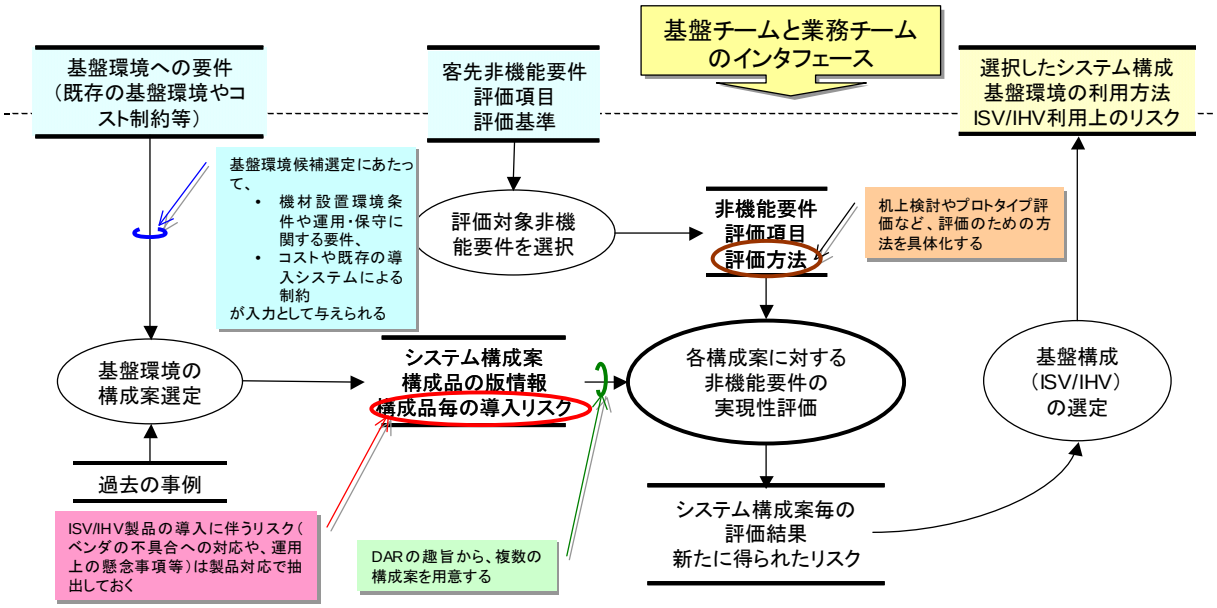


図 7-4　基盤環境の選定と評価の作業フロー

各作業は、おおむね次のステップを踏んで実施する。

① 基盤環境の構成案選定

まず、実行基盤環境を構成する ISV/IHV 製品の組み合わせを選定する。作業に対する入力は、1 つは基盤環境に対する要件として、顧客のコスト面での制約や現在運用している基盤環境の特性、課題を取る。また結果として、システム構成案、構成品の導入リスクを記載したリスク管理シートを作成する。

この段階では、実行基盤環境のシステム構成案は複数用意すること。また、各構成品のバージョン情報も明確にしておく。特に ISV/IHV 製品のバージョン情報は、機能内容や API が変わる可能性もあるため、明確にしておく。

② 評価対象非機能要求の選択

ここでは、初期の実行基盤環境のプロトタイピングを実施する、あるいは机上評価を

実施するための評価基準として、非機能要求として何が求められているのか、またどの程度の目標値を達成すればよいのかを具体化したリストを業務アプリケーション開発チームから収集する。これを入力にして、実行基盤環境の評価手順、評価方法を具体化するのが、この活動の目的である。従って、この活動の出力は、非機能要求の評価項目と評価方法になる。評価方法には、例えば、

・ プロトタイピングを実施し、特定のポイントでの入出力の反応時間を計測する、
・ 机上検討により、どのような条件を想定して試算を行う

等の方法を具体的に定義する。

この段階で、顧客の業務要件から具体化された当該情報システムの構築にあたって考慮すべき非機能要求の範囲が具体化される。これは、必要な非機能要求項目を選択する形で実施する。

③ 各構成案に対する非機能要求の実現性評価

本作業は、①で得られた複数の基盤環境構成案から、いずれかの基盤環境構成案を選択する作業となる。基盤環境構成案を具体的に評価するために、代表的な業務を選択し、これをサンプルに、各基盤環境構成案がどの程度非機能要求を達成するか、また採用した基盤環境構成案がどのような運用リスク、開発リスクを抱えるかを具体化する。評価にあたっての基準、評価方法は、②で定めた評価方法、評価項目を入力として得る。

この③の活動としては最初に、これから開発する情報システムの代表的な機能をサンプルとして選択し、用意したシステム構成案上でこの機能のシナリオを具体化する。ここで、シナリオの実施フローに従ってシステム構成品に非機能要求を展開、割当て、顧客の非機能要求が情報システムの上でどのように実装されるかを具体化する。その上で、システム構成品毎の評価基準をより詳細に設定し、評価用プロトタイプを構築して実行評価を行う。これを、用意した基盤環境構成案すべてについて実施する。

重要な点は、これらの評価活動で得られた知見や技術的なリスクを、記録として残し、組織のノウハウとして蓄積することにある。評価した結果は、後述するシステム実装検討書の形でまとめ、どのような実行基盤環境構成を、どのような条件のもとで評価し、どのようなリスクが明らかになったか、が後から参照できるようにすることにある。これにより、将来システム構成を変更する必要があった場合にも、すべての評価活動を再度実施するのではなく、過去の経験に対して継続的にノウハウを蓄積することが可能となる。

④ 基盤環境の構成案選定

最後のステップは、③での評価結果を受けてもっとも技術的なリスク、顧客の導入コストや運用上のリスクなどに照らしてもっとも適切な実行基盤環境の構成案（ISV/IHVの組み合わせ案）を選定する。この作業の入力は、③で得られたシステム実装検討書の記載項目になる。

このような作業手順を踏むのは、これによってどのような根拠で実行基盤環境を選定したか、また非機能要求の実現にあたってどのような制限が存在するかを、業務アプリケーション開発チームの開発活動が本格化する前に具体化することが、基盤チームの開発として重要になってくる。

7.3　非機能要求トレーサビリティ管理の方法

7.2 節に示した開発の流れを前提にして、ここでは非機能要求トレーサビリティの管理方法を概説する。

7.3.1　非機能要求に対するトレーサビリティマトリクスの様式

非機能要求トレーサビリティ管理を実施するための主要文書が、非機能要求に対するトレーサビリティマトリクスである。

この非機能要求トレーサビリティマトリクスの様式を、図 7-5 に示す。以下の考え方でこの様式を設定している。

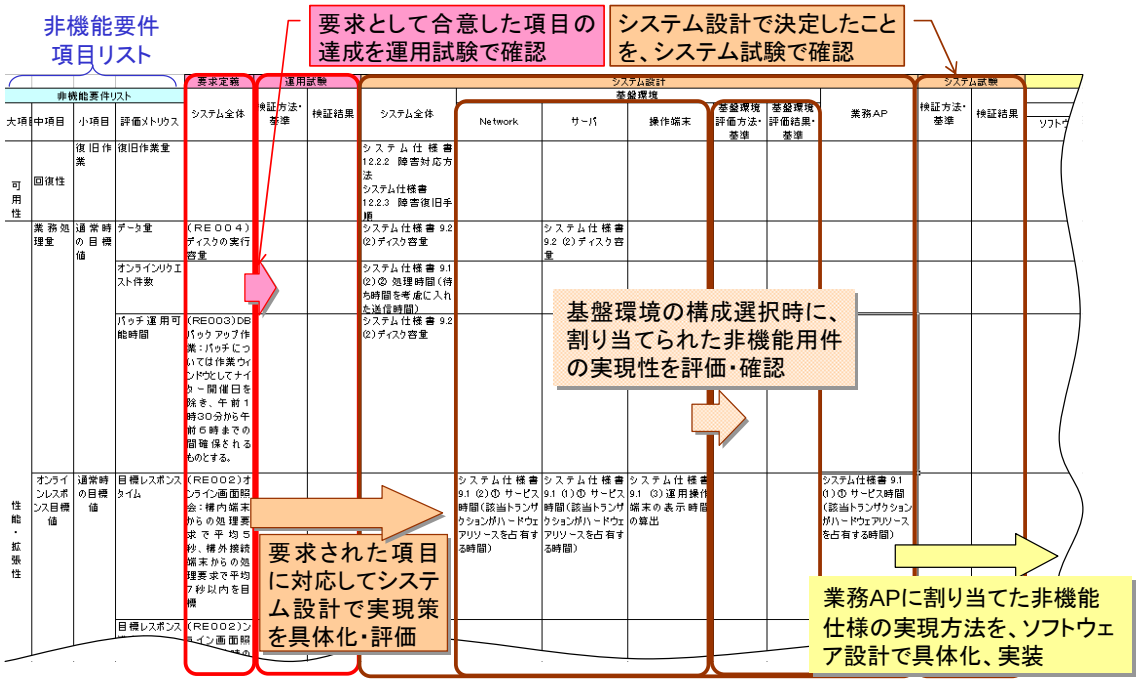


図 7-5　非機能要求管理のためのトレーサビリティマトリクス

①　縦軸に非機能要求項目のリストを、横軸に開発フェーズを示す

縦軸に示してある非機能要求項目リストは、非機能要求項目リストに記載されている非機能要求である。開発対象のシステムによって、いずれの非機能要求の達成が求められるかは異なることに注意する。開発する情報システムに要求されない非機能要求は、記載する必要は無い。一方で、情報システムの構築では ISV/IHV 製品などがもたらす実装上の制約事項から、新たな非機能要求が具体化する場合がある。このような場合は、開発フェーズが進んだところで、新たな非機能要求としてトレーサビリティマトリクス上に項目を追加することを想定している。

横軸は、開発ライフサイクルプロセスを示している。ここでは、特に非機能要求に対して

- どのような非機能要求の実現が要請されているか（要求されている項目）
- その非機能要求の達成を評価するために、どのような手段を取れるのか（確認の手段）
- 実際に非機能要求を達成した結果どうだったのか（確認の結果）

の対応付けが取れるように、トレーサビリティマトリクスを構成している。

② 顧客要求として定義された非機能要求は、どの開発フェーズで、どのシステム構成品に割り当てられ、どのようにその実現が確認されたかを管理することを狙いとしている。

情報システムの構築にあたって、顧客の要求は顧客の情報システム導入の目的にあわせて設定される。これに対して我々SIer は、顧客の目的を達成する手段として情報システムを構築する。この過程で、顧客要求を満たすためにどのようなシステム構成をとるのかを決定することになる。この作業の流れは、図 7-3 に示した通りである。また、この過程で図 7-4 に示したように情報システムのシステム構成品が決まった時点で、情報システム全体に対する非機能要求はシステム構成品に割り当てられ、より具体的な非機能要求として展開されていく。ここで示したトレーサビリティマトリクスは、このような非機能要求の展開を、システム構成品にどのように割り振ったか、また割り振られた非機能要件をどのように試験したかをトレースすることを狙っている。このため、横軸ではシステム構成品への機能割当ての項目を入れている。

③ 割り当てられた非機能要求に対し、優先順位付けを与える

個々のシステム構成品に割り当てた非機能要求は、そのすべてを完全に達成することが難しい場合がある。例えば、高いセキュリティ要件がある場合、しばしば計算機リソースを余計に消費する可能性がある等である。このように相反する非機能を調整する必要がある場合、抽出された非機能要求のいずれを優先するかを明示的に定義することが重要となる。

この優先順位付けを明示的に記載する道具として、非機能要求トレーサビリティマトリクスを利用すると良い。どのシステム構成品に割り当てた非機能要求を、どの優先順位で実現するかを記録することで、システム設計時の実現方式の選択基準が明らかになるためである。なお、これらの優先順位付けを利用して方式設計を実施する手法としてはいくつか提唱されている。Architecture Tradeoff Analysis Method もその一例であり、これら手法を用いて非機能要求間のコンフリクトを調整する。

7.3.2 トレーサビリティ管理の実施手順

非機能要求に対するトレーサビリティ管理の実施手順は、以下の通り。

なお、以下に示す一連の作業を実施する際、基準となる文書（提案書や要求仕様書）のベースラインを確立し文書番号を保持すること。これは要求仕様が変更されたとき、これに伴って非機能要求も変更される可能性があるためである。

① 非機能要件グレード表を参考に、要求仕様に記載されている非機能要求の記載箇所を抽出する

非機能要求は、要求仕様書や提案書の様々な場所に記載されている。例えば、利用性にかかわる非機能要求はユーザインタフェースを記載した部分にあるかもしれないし、運用要件の中に記載されている可能性もある。また、性能にかかわる非機能要求も、開発対象システムの提供するサービス・機能ごとに記載されている可能性もある。従って、非機能要求を要求仕様からピックアップする際には、これらのことを勘案して何が求められているのかを明らかにする。

② 抽出した各非機能要求に対し、非機能要求グレード表から、どの非機能要求に相当するかを割り当てる。

この際、非機能要求グレード表を基準に大項目・中項目・小項目までを識別する。非機能要求グレード表に記載されたメトリクスは、必要に応じて代替メトリクスを設定することもあるので、メトリクスを単位として非機能要求グレード表とマッピングする必要は無い。ただし各項目に対して、その達成を評価する指標と、その指標の目標値が設定されていることを確認する。

なお、非機能要求グレードに記載される全非機能要求に、なんらかの顧客要求を割

り当てる必要は無い。開発対象となる情報システムに必要な項目だけに絞り込んで記載すればよい。

③ 非機能要求が記載されている箇所に、非機能要求を示すタグを埋め込む

いずれの非機能要求に相当するかがわかるように、要求仕様の非機能要求記載の箇所に非機能要求識別のタグを埋めこむ。

ここで埋め込むタグの形式は、例えば以下のような形式で記載する。

<[ソース文書 ID]-[構成品 ID]-[非機能要求 ID]>

ここで非機能要求 ID 番号は、②で割り当てた「非機能要求の種別」を識別する番号を想定している。

また、非機能要求識別タグを要求仕様書などの文書中に埋め込むことで、将来のツールによる非機能要求トレーサビリティマトリクスの更新を行うことを想定している。

④ 埋め込んだタグの記載箇所を、非機能要求トレーサビリティマトリクスに転記する

③で埋め込んだ「非機能要求タグ」を埋め込んでいる文書の章・節番号を、非機能要求トレーサビリティマトリクスに転記する。

以降は、トレーサビリティマトリクスを有効活用し、設計/実装/試験の各ステップで評価、検証を繰り返していく。

7.4 試行結果

特定部門にて、新たに追加したプロセスのもと、ISV/IHV から構成される基盤環境に対する要求と検証項目を「システム基盤実装検討書」として文書化し、システムのテスト項目と顧客要求との対応付けを取る「トレーサビリティ管理シート」を試行運用し、結果として、その有効性の確認はできた。

効果： 作業項目が具体化され、どれだけの試験項目が必要か、より明らかになった。
従来意識して確認していなかった試験項目が、具体的にチームのみんなに見えるようになった。

課題： 確認作業量が当初の計画よりも膨らむ傾向にある。
開発者は気がついた確認項目をすべて書き出そうとするため、非機能要求に対する優先順位付けを、より明確に設定する必要がある。
また経験した不具合を念頭に、試験方法、確認項目を抽出する傾向がいまだ強いため、確認項目の網羅性を担保するためのチェックリストは別途必要である。

7.5 非機能要求のトレーサビリティ管理プロセスの導入効果

ISV/IHV 起因の重大障害のうち、非機能要求トレーサビリティ管理の枠組みを導入することで、過去の ISV/IHV 起因の重大障害の約半数の障害に対し、将来の障害発生の可能性を把握できた可能性があると判断している。

ただし、ISV/IHV 製品起因の障害発生を 100%防止するのは現実的ではなく、また H/W 製品の経年変化に伴う故障は避けられない。さらに、ISV/IHV 製品に対する網羅的な試験を、現実的なコスト・期間で実施することは困難であるため、ISV/IHV 製品起因の障害発生をリスクとしてとらえ、障害発生時の影響を軽減する方策を準備する（リスク軽減策を計画し、運用する）ことも並行して重要となる。

しかし ISV/IHV 製品導入にあたって、どのようなリスクがあるのか（試験しきれない仕様がどの程度あるのか）を事前に把握すること、そしてどの品質特性を事前に確認する必要があるか、どれだけのテストを必要とするのかをプロジェクト計画時から把握すること、すなわちリスク管理と一体となった運用こそが重要である。

< 空欄 >

第8章　株式会社ジャステックの事例
～　独自の生産管理方式「ACTUM」に基づく高信頼ソフトウェア開発の取り組み　～

8.1　取り組みの背景と経緯

省力化や利便性向上といったニーズの高まりと情報技術の急速な進展を背景に、情報システムは社会的な基盤を担うようになり、ソフトウェアの信頼性や安全性への要求が高まりつつある。このような状況下、弊社では創業以来、役務提供を前提にした人月単価による契約を排除し、一括請負契約拡大を標榜している。そのために、ソフトウェアの生産管理および品質管理に立脚した価格設定に基づくソフトウェア開発の生産管理方式「ACTUM」を独自に構築し、運用している。

弊社の生産管理方式「ACTUM」は、1980 年代にプロジェクト管理の基本を確立し、その後、1993 年には ISO9000 を取り入れ、組織レベルで規程を整備し運用基盤を確立した。さらに、客観的管理指標によるプロセス改善点を掘り起こすなど、品質システムを高度化し、2003 年 10 月に CMMI バージョン 1.1 で、2009 年 1 月に CMMI バージョン 1.2 で、いずれも全社レベルにおいて成熟度レベル 5 の公式評価を得ている。

8.2　ソフトウェアの信頼性にかかわる問題意識

弊社ではソフトウェアの信頼性に関して、以下①から③の 3 つの課題を「見える化」から、「計る化」に基づく「見せる化」を実践している。

① ソフトウェアの信頼性は、客観的な把握や制御が難しいとされている。ソフトウェアの信頼性を開発、保守、運用のライフサイクル全般を通して確保および向上させるために、定量的なマネジメントの確立が必要となる。

弊社では、お客様と約束した品質、費用および納期の目標を達成することを目的とし、「信頼性を向上させる管理項目」として開発および保守にかかわる 192 項目の管理指標を定めている。なお、その内容「計る化」は 2008 年 4 月発刊の社団法人情報サービス産業協会（JISA）「信頼性向上のベストプラクティスを実現する管理指標調査報告書」で紹介した。

② 業務アプリケーションソフトウェアには航空、鉄道、電力、ガス、水道、金融、情報通信、政府・行政サービスおよび医療など、高信頼性を要求されるシステムと通常の信頼性を要求される社内業務システムなど、当該システムのシステムプロファイルごとに要求されるソフトウェアの要求品質は一様ではない。よって、ソフトウェアの品質を高める手段を明確にし、ソフトウェア品質と価格の関係を整理した上で、適正コストの枠組みをユーザとベンダとで合意する必要がある。

弊社では、「ACTUM」に基づくソフトウェア開発の見積りモデルを使用し、要求品質を定量的にとらえ「計る化」、価格面の根拠「見せる化」としてお客様へ提示している。8.3 節では、弊社の見積りモデルを適用してソフトウェアの信頼性と開発コストとの関係を紹介する。

③ システム化ニーズの高揚などによる要求機能の増加、およびソフトウェアの信頼性への担保要求が高まる反面、システムにかかわる予算や開発期間が抑えられるといった二律背反的な要求が強まっている。問題は、この二律背反課題をあいまいにしたまま開発に着手するが、コスト増加や信頼性の低下を招くことである。

弊社では、品質機能展開を応用し要求機能（非機能含む）ごとの重要度を確認「見せる化」し、弊社の見積りモデルに連携させる試みを始めている。8.4 節で、その事例を紹介している。

8.3 ソフトウェアの信頼性と開発コストとの関係

8.3.1　見積りモデルの基本アルゴリズム

本項では、ソフトウェアの信頼性と開発コストとの関係を明示するために、弊社での見積もりモデルの基本アルゴリズムについて紹介する。新規開発、改造型開発、仕様変更およびテストにかかわる見積り方式の詳細は、2006 年 4 月発刊の IPA/SEC　BOOKS「ソフトウェア開発見積りガイドブック」、2007 年 10 月発刊の「ソフトウェア改造開発見積りガイドブック」および 2008 年 9 月発刊の「ソフトウェアテスト見積りガイドブック」で紹介しているので、参照を願う。

弊社が考案した、新規開発におけるコスト見積りモデルの基本アルゴリズムを示す。本アルゴリズムは、ある開発工程 i の基準生産物量を V_i^B、基準生産性 P_i^B で表現すると、基準開発コスト C_i^B は $C_i^B = V_i^B \times P_i^B$ となる。通常、ソフトウェア開発には基準値に対して様々な変動要因が生産物量および生産性に影響を与える。そこで、変動要因を考慮した生産物量 V_i、生産性 Pi および開発コスト Ci を求める式を以下に示す。

$V_i = Vi^B \times (1 + a_i)$　　$(a_i = \Sigma\ \alpha_{ij})$

$Pi = Pi^B \times (1 + b_i)$　　$(b_i = \Sigma\ \beta_{ij})$

$C_i = V_i \times P_i$

a_i は生産物量環境変数と呼ぶパラメータであり、Vi^B に対して品質要求の多寡による変動を吸収する変数である。また b_i は、生産性環境変数と呼ぶパラメータであり、Pi^B に対して開発環境の違いや品質要求の多寡による変動を吸収する変数である。a_i、b_i は品質特性と環境特性から影響される独立した変動要素（α_{ij}, β_{ij}）から構成されている。

品質特性は JIS X0129-1：2003 の品質特性を使用しており、ソフトウェア欠陥の多寡のみならず、障害許容性（障害予防機能、拡大防止機能の実装度合い等）および回復性（回復機能の実装度合い等）など、15 の品質副特性を対象にしている。具体的には、成果物の生産物量に影響を及ぼす生産物環境変数 a_i は、JIS X 0129-1：2003（元の ISO 規格は ISO/IEC 9126-1：2001）の外部品質/内部品質に挙げられている特性/副特性別に、影響度を 5 段階に分けて数値化している。要求品質が最も高い場合を 5 つ目の段階（レベル 5）として区分したものである。

同じように生産性に影響する生産性環境変数 b_i については、前述の品質特性に加えて、環境特性が影響を及ぼすとしている。環境特性には当該開発チームの編成、開発環境の具備度合いおよび顧客とのコミュニケーション特性などがある。生産性環境変数 bi は生産物量環境変数 a_i と同様の 5 つの段階に分けた数値化を定めている。

8.3.2 システムプロファイルごとの要求品質と開発コストとの関係

(1) ソフトウェアの信頼性に関係する環境変数

表 8-1に示すように、環境変数は全部で45個の因子を持っている。しかし、ソフトウェアの信頼性に関係する特性は、品質特性が対象［関連(有)］となるため、45個の環境変数因子の中から19個を選択している。さらに19個の因子のうち、信頼性の要求水準に強く影響する因子［関連(強)］と普通に影響する因子［関連(普)］とに分類している。

表 8-1　ソフトウェアの信頼性に関連する環境変数

全環境変数因子数			関連(有)環境変数の因子数									関連(無)環境変数因子数		
						関連(強)			関連(普)					
a_i	b_i	計	a_i	b_i	計	a_i	b_i	計	a_i	b_i	計	a_i	b_i	計
14	31	45	13	6	19	13	3	16	0	3	3	1	25	26

a_i：生産物量環境変数　b_i：生産性環境変数

(2) システムプロファイルと環境変数の因子レベルとの対応

ここで使用するシステムプロファイルは、経済産業省「情報システムの信頼性向上に関するガイドライン」が定義している情報システム分類のプロファイルに基づいている。

高信頼ソフトウェアとしての要求は、「重要インフラ等システム」が最も高いとしており、環境変数「関連（強）」の 5 段階区分ではレベル 4～レベル 5 を設定した。なお、システムプロファイルごとのレベル設定は、表 8-2 に示す。

表 8-2　情報システムの分類と環境変数の因子レベルとの対応

情報システム分類	関連（強）環境変数	関連（普）環境変数
重要インフラ等システム	レベル4～レベル5	レベル3～レベル5
企業基幹システム	レベル3～レベル4	レベル2～レベル4
その他システム	レベル1～レベル2	レベル1～レベル3

システムプロファイルごとの開発コスト変動値を求めるため、システムプロファイルごとに 19 個の環境変数の取り得る値を、表 8-3 から表 8-6 に示す。

表 8-3 から表 8-5 がシステムプロファイルごとの生産物量に影響する生産物量環境変数 a_i で、表 8-6 がシステムプロファイルごとの生産性に影響する生産性環境変数 b_i である。

表 8-3　システムプロファイルごとの生産物量に影響する生産物量環境変数（1/3）

品質特性	副品質特性	評価の観点	外責評価基準				品質水準毎による影響評価												関連（強）
							重要インフラ等システム				企業基幹システム				その他システム				
		内　容	影響度（%）				影響度（%）				影響度（%）				影響度（%）				
			要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	◎
機能性	合目的性	利用者／利害関係者の広がり，コンテンジェンシー対応，不正移行データ対応などの該当事象数	0～50	0～50	0～50	0～50	30～50	30～50	30～50	30～50	10～30	10～30	10～30	10～30	0～0	0～0	0～0	0～0	◎
			影響度小計					39	～	65		13	～	39		0	～	0	
	正確性	正確性（検証）にかかわる標準テスト密度を基準にしたテスト項目量への要求水準	－	－	0～20	0～50	0～0	0～0	15～20	30～50	0～0	0～0	10～15	20～30	0～0	0～0	0～0	0～0	◎
			影響度小計					15	～	23		10	～	15		0	～	0	
	セキュリティ	対応が必要なセキュリティ実現機能数，ただし機能要件に定義されている部分は除く	0～20	0～20	0～20	0～20	15～20	15～20	15～20	15～20	10～15	10～15	10～15	10～15	0～0	0～0	0～0	0～0	◎
			影響度小計					20	～	26		13	～	20		0	～	0	
信頼性	成熟性	故障低減に必要な実現機能数	0～5	0～10	0～10	0～10	3～5	6～10	6～10	6～10	2～3	3～6	3～6	3～6	0～0	0～0	0～0	0～0	◎
			影響度小計					7	～	12		4	～	7		0	～	0	
	障害許容性	異常検知に必要な機能数	0～5	0～10	0～10	0～10	3～5	6～10	6～10	6～10	2～3	3～6	3～6	3～6	0～0	0～0	0～0	0～0	◎
			影響度小計					7	～	12		4	～	7		0	～	0	
	回復性	再開処理に必要な実現機能数	0～5	0～10	0～10	0～10	3～5	6～10	6～10	6～10	2～3	3～6	3～6	3～6	0～0	0～0	0～0	0～0	◎
			影響度小計					7	～	12		4	～	7		0	～	0	

（注）影響度の工程統合（要件定義＋設計）：製作：テスト＝3：4：3）

表 8-4　システムプロファイルごとの生産物量に影響する生産物量環境変数（2/3）

品質特性	副品質特性	評価の観点	外責評価基準				品質水準毎による影響評価												関連（強）◎
							重要インフラ等システム				企業基幹システム				その他システム				
		内　容	影響度(%)				影響度(%)				影響度(%)				影響度(%)				
			要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	
使用性	理解性	理解性向上（機能など）のためのプレゼンツール等の作成対象数	－	0～10	－	－	0～0	6～10	0～0	0～0	0～0	3～6	0～0	0～0	0～0	0～0	0～0	0～0	◎
			影響度小計					2	～	3		1	～	2		0	～	0	
	習得性	習得性向上（使い方など）のためのマニュアルなどの作成対象数	－	0～10	－	－	0～0	6～10	0～0	0～0	0～0	3～6	0～0	0～0	0～0	0～0	0～0	0～0	◎
			影響度小計					2	～	3		1	～	2		0	～	0	
	操作性	操作性向上（心理的／肉体的配慮，運用やインストール容易性など）のための実現機能数	0～10	0～20	0～20	0～20	6～10	15～20	15～20	15～20	3～6	10～15	10～15	10～15	0～0	0～0	0～0	0～0	◎
			影響度小計					17	～	23		11	～	17		0	～	0	
保守性	解析性	解析に必要な実現機能数	－	0～10	0～10	0～10	0～0	6～10	6～10	6～10	2～0	3～6	3～6	3～6	0～0	0～0	0～0	0～0	◎
			影響度小計					6	～	10		4	～	6		0	～	0	
	変更作業性	作成する保守用ドキュメントの数	－	0～10	－	－	0～0	6～10	0～0	0～0	0～0	3～6	0～0	0～0	0～0	0～0	0～0	0～0	◎
			影響度小計					2	～	3		1	～	2		0	～	0	
	試験性	試験に必要な機能数	－	0～5	0～15	0～20	0～0	3～5	7～15	10～20	0～0	1～3	3～7	5～10	0～0	0～0	0～0	0～0	◎
			影響度小計					7	～	14		3	～	7		0	～	0	

（注）影響度の工程統合（要件定義＋設計）：製作：テスト＝3：4：3）

表 8-5　システムプロファイルごとの生産物量に影響する生産物量環境変数（3/3）

品質特性	副品質特性	評価の観点	外責評価基準				品質水準毎による影響評価												関連（強）◎
							重要インフラ等システム				企業基幹システム				その他システム				
		内　容	影響度(%)				影響度(%)				影響度(%)				影響度(%)				
			要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	要件定義	設計	製作	テスト	
機能性	正確性（既存母体）	改造／流用母体が正しく動作しない場合のテスト量（現行保証）に及ぼす影響	－	0～5	0～15	0～20	0～0	3～5	10～15	15～20	0～0	2～3	3～10	5～15	0～0	0～0	0～0	0～0	◎
			影響度小計					9	～	14		3	～	9		0	～	0	
	生産物量影響度総計							140	～	220		72	～	140		0	～	0	

表 8-6　システムプロファイル-ごとの生産性に影響する生産性環境変数（1/1）

品質特性	副品質特性	評価の観点 内容	外責評価基準 影響度(%) 要件定義	設計	製作	テスト	品質水準毎による影響評価 重要インフラ等システム 影響度(%) 要件定義	設計	製作	テスト	企業基幹システム 影響度(%) 要件定義	設計	製作	テスト	その他システム 影響度(%) 要件定義	設計	製作	テスト	関連（強）◎
機能性	合目的性（要求仕様の網羅性）	要求の記述水準および網羅性。要件定義については新規性，方針明確性，ステークホルダーの多様性などを考慮	0～100	0～30	–	0～10	20～100	5～30	～	3～10	0～50	0～15	～	0～6	0～20	0～5	～	0～3	
			影響度小計				8	～	42		0	～	21		0	～	8		
	正確性	正確性（検証）にかかわる標準レビュー工数（各工程8%）を基準にした要求水準	0～5	0～5	0～3	0～5	3～5	3～5	2～3	3～5	2～3	2～3	1～2	2～3	0～0	0～0	0～0	0～0	◎
			影響度小計				4	～	6		2	～	4		0	～	0		
	整合性	整合を取る社内／社外の規格・基準の数，全体適合性やグローバル化対応を含む	-10～10	-5～10	-3～5	-3～5	0～10	3～10	1～5	1～5	-5～5	0～5	0～3	0～3	-10～0	-5～3	-3～1	-3～1	
			影響度小計				2	～	10		-2	～	5		-7	～	2		
効率性	実行効率性	実行効率に対する一般的要求水準（既知）の標準事例を基準にした要求水準	0～5	0～10	0～5	0～10	2～5	3～10	2～5	3～10	0～3	0～6	0～3	0～6	0～2	0～3	0～2	0～3	
			影響度小計				3	～	10		0	～	6		0	～	3		
保守性	解析性	ソースコードの解析性を、コード化規約に定めるコメント率に対する要求水準により評価	–	–	0～5	–	0～0	0～0	3～5	0～0	0～0	0～0	2～3	0～0	0～0	0～0	0～0	0～0	◎
			影響度小計				1	～	2		1	～	1		0	～	0		
	安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価	0～5	-3～7	-3～5	–	3～5	4～7	3～5	0～0	2～3	2～4	2～3	～	0～0	-3～0	-3～0	0～0	◎
			影響度小計				3	～	6		2	～	3		-2	～	0		
	生産性影響度総計						21	～	76		3	～	40		-9	～	13		

(3) システムプロファイルごとの基準開発コストからの変動率

システムプロファイルごとの生産物量環境変数と生産性環境変数は、表 8-3 から表 8-6 に示す 19 個の環境変数（生産物量環境変数と生産性環境変数）の取り得る値を開発工程ごとに統合し求めている。その結果を表 8-7 に示す。

表 8-7　システムプロファイルごとの生産物量環境変数と生産性環境変数

環境変数（合計）	システムプロファイルごとの環境変数（影響度%）		
	重要インフラ等システム	企業基幹システム	その他システム
生産物量環境変数（a）×100	140 ～ 220	72 ～ 140	0 ～ 0
生産性環境変数（b）×100	21 ～ 76	3 ～ 40	-9 ～ 130

表 8-7 で求めたシステムプロファイルごとの生産物量環境変数 a および生産性環境変数 b を以下の式に挿入することにより、システムプロファイルごとの基準開発コスト C^B「$=V^B \times P^B$」からの変動率 ε が求まる。

$$C = C^B(1+a+b+[a \times b])$$

$$\varepsilon = (a+b+[a \times b]) \times 100$$

その結果を表 8-8 に示す。

表 8-8　システムプロファイルごとの基準開発コストからの変動率

	システムプロファイルごとの基準コストからの変動率(%)		
	重要インフラ等システム	企業基幹システム	その他システム
基準コストからの変動率(ε)	190 ～ 463	77 ～ 236	-9 ～ 13

(4) 高信頼ソフトウェアの開発コスト

高信頼ソフトウェアとしての要求は、「重要インフラ等システム」が最も高いとした。

つまり、高信頼ソフトウェアを要求する情報システムのソフトウェア開発は、基準開発コストと比較して、開発コストが「1.9 倍から 4.6 倍」の追加コストが発生するという試算結果となった。この数字は、あくまで弊社の統計分析からの基準値および経験則からの因子レベル選定に基づいているため、参考値として利用されることを望む。

なお、すべての環境変数（生産物量環境変数、生産性環境変数）に対応する 1 段階から 5 段階の数値は、社団法人日本情報システム・ユーザー協会（JUAS）「システム開発生産性プロジェクト」で検証・評価された数値に基づいており、詳細内容は JUAS の「システム・リファレンス・マニュアル‘SRM’」および「UVC 研究プロジェクト報告書‘要求仕様定義書ガイドライン’」に記載している

8.3.3 出荷後の残存欠陥とソフトウェア開発コストとの関係

(1) システムプロファイルごとの残存欠陥密度

図 8-1は、弊社の品質/環境管理室が全プロジェクトを対象として出荷後の残存欠陥密度を数年間（図 8-1では4年間）にわたり実績収集したものを、システムプロファイルごとにサンプリングし表示した。出荷後の残存欠陥密度（件/KS）は、「その他システム(2005年)での平均値」を基準‘1’として、企業基幹システムおよび重要インフラ等システムの残存欠陥密度を年度ごとに相対比較したものである。なお、平均値の比較は統計的に有意差があることを検証した値である。

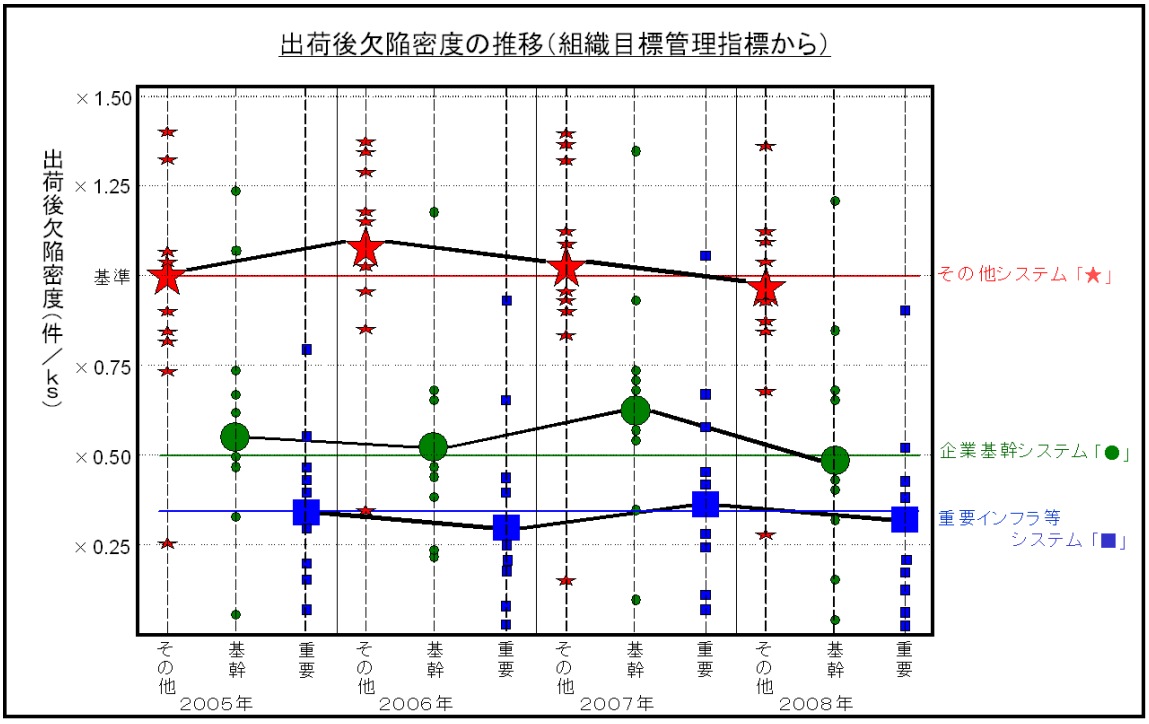


図 8-1　システムプロファイルごとの残存欠陥密度の推移

(2) 残存欠陥密度に影響する品質特性

ここでの残存欠陥密度に影響する品質特性は、正確性と試験性を選択した。

正確性： レビュー充実度およびテスト密度（テスト網羅率）

試験性： 試験環境（機器、支援ツール等）の充実度

(3) 残存欠陥密度に影響する環境変数と基準開発コストからの変動率

品質特性（正確性、試験性）に対するシステムプロファイルごとの環境変数（生産物量環境変数 a、生産性環境変数 b）およびシステムプロファイルごとの基準開発コスト C^B からの変動率 ε を、表 8-9 に示す。

表 8-9 システムプロファイルごとの環境変数と基準開発コストからの変動率

	重要インフラ等システム	企業基幹システム	その他システム
生産物量環境変数(a)×100	31 ～ 51 （%）	16 ～ 31 （%）	0 ～ 0 （%）
生産性環境変数(b)×100	4 ～ 6 （%）	2 ～ 4 （%）	0 ～ 0 （%）
基準コストからの変動率(ε)	36 ～ 60 （%）	18 ～ 36 （%）	0 ～ 0 （%）

(4) システムプロファイルごとの残存欠陥密度と開発コストとの比較

システムプロファイルごとの残存欠陥密度と開発コストとの比較を、以下の表 8-10 に示す。図 8-1（システムプロファイルごとの残存欠陥密度の推移）と表 8-9（システムプロファイルごとの基準開発コストからの変動率）から求めている。なお、システムプロファイルごとの残存欠陥密度は、図 8-1 での 2008 年の平均値を適用しており、「その他システム」を基準‘1’としている。

表 8-10 システムプロファイルごとの残存欠陥密度と開発コストとの比較

システムプロファイル	残存欠陥密度	開発コスト
その他システム	1(基準)	1(基準)
企業基幹システム	×0．5	+18% ～ +36%
重要インフラ等システム	×0．33	+36% ～ +60%

8.4 高信頼ソフトウェアを目指した見積りモデルの有効利用

弊社の見積りモデルに使用している環境変数は、システム全体をとらえて評価し、お客様に「見せる化」している。高信頼ソフトウェアを目指し、さらなる見積りモデルの有効利用を促進するために、要求機能ごとの環境変数（生産物量環境変数 a_i、生産性環境変数 b_i）を求める試みを始めている。

その関係を図 8-2に示す。ここに出てくる「品質機能展開」は、JIS Q 9025;2003（マネジメントシステムのパフォーマンス改善 ―品質機能展開の指針―）をソフトウェア開発向けに応用したものである。

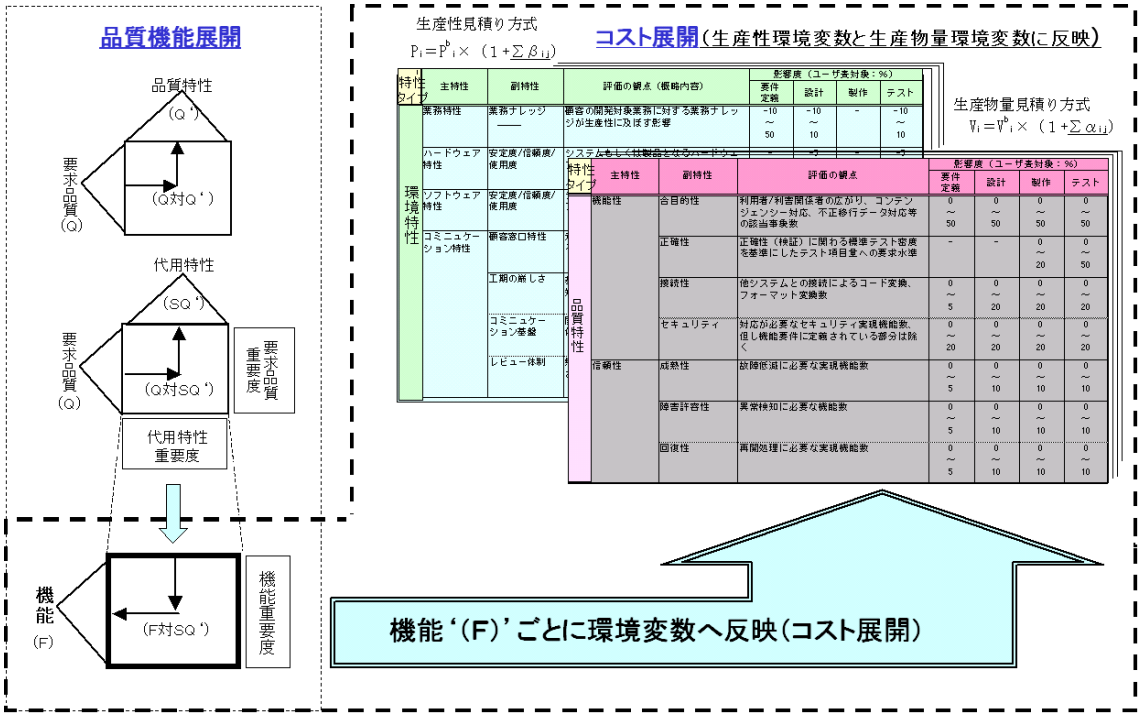


図 8-2　品質機能展開からコスト展開（環境変数への反映）

図 8-2（品質機能展開からコスト展開）は、品質機能展開から得られる機能から、環境変数を算出し開発コストを求めること、さらに、機能および品質特性「代用特性」ごとの重要度をとらえ、開発予算に見合った機能および高信頼ソフトウェアの担保範囲を、上流工程の段階で調整することを目的にしている。これは、8.2 節 ③で述べたような、ソフトウェアに対する要求の増加と開発予算の抑制といった課題を「計る化」に基づく「見せる化」の方法として紹介した。

なお、表 8-11は、教育管理システムの一部機能「照会系」を品質機能展開の応用事例として紹介している。

表 8-11 品質機能展開を応用した事例

機能と品質特性「代用特性」との展開事例（教育管理システム）

	要求機能		機能性				効率性		使用性								信頼性	
	（NO1） 照会系		合目的性		セキュリティ		時間効率性		操作性		運用性		魅力性				回復性	
①	■知識取得機会を増やす為に、全社で企画する全ての受講講座内容が把握できるようにしたい。 【受講者、上司】	5	①人材開発部＆各部の教育講座一覧表示 ②受講講座内容の把握	○15 ◎25	①閲覧権限（権限クラスC）	○15	①レスポンス（ランクC）	○15			①ログインの自動認識	○15					①障害回復30分以内	○15
②	■仕事の状況に対応し、柔軟に対応する為など、いつでも受講履歴が閲覧できるようにしたい。 【受講者、上司、教育部】	4	①受講予定と実績の把握 ②受講予定変更への連携 ③受講成績の把握	◎20 ○12 ◎20	①閲覧権限（権限クラスC）	○12	①レスポンス（ランクC）	○12	①自動連携（受講履歴照会と申込み’変更’）	○12	①ログインの自動認識	○12					①障害回復30分以内	○12
③	■申し込んだ講座および内容が間違っていないか確認したい。 【受講者】	5	①講座一覧の表示 ②講座内容の把握	○15 ◎25	①閲覧権限（権限クラスC）	○15	①レスポンス（ランクC）	○15			①ログインの自動認識	○15					①障害回復30分以内	○15
④	■今後の知識取得の参考にするなど、受講成績を知っておきたい。 【受講者、上司】	5	①受講成績の把握	◎25	①閲覧権限（権限クラスB）	◎25	①レスポンス（ランクC）	○15			①ログインの自動認識	○15	①グラフィカル表示	○15			①障害回復30分以内	○15
⑤	■受講予定者の変動に対応すべく講座で使用する教材の在庫量を把握しておきたい。 【教育部】	4	①教材在庫の把握 ②未回収教材の督促連携	◎20 ○12	①閲覧権限（権限クラスC）	○12	①レスポンス（ランクC）	○12			①ログインの自動認識 ②連携（督促メール）	○12 ○12					①障害回復30分以内	○12
⑥	■教育計画改善への参考として受講者側からの声を収集したい。 【教育部】	5	①受講成績の把握 ②アンケートの収集・分析	◎25 ◎25	①閲覧権限（権限クラスB）	◎25	①レスポンス（ランクC）	○15			①ログインの自動認識	○15	①グラフィカル表示	○15			①障害回復30分以内	○15
⑧	■教育部門でのISMS対応として閲覧権限管理を行う必要がある。 【教育部】	3	①閲覧権限者の一覧表示	○9	①閲覧権限（権限クラスC）	○9	①レスポンス（ランクC）	○9	①自動連携（閲覧権限情報を人事sysから取得）	○9	①ログインの自動認識	○9					①障害回復30分以内	○9

（留意）表中は代用特性を示す

要求機能の重要度

5；絶対に実現してほしい
4；実現してほしい
3；他の条件を考慮し、実現してほしい
2；あったほうがよい
1；不要、無意味である

代用特性の重要度（下記点数×要求機能の重要度）

◎’5’；品質特性として必須である
○’3’；品質特性として必須ではないが、ないと満足感が低下する
△’1’；品質特性としてあったほうがよい

参考文献

[1] IPA SEC： SEC BOOKS ソフトウェア開発見積りガイドブック, 2006 年 6 月.
[2] IPA SEC： SEC BOOKS ソフトウェア改造開発見積りガイドブック, 2007 年 10 月.
[3] IPA SEC： ソフトウェアテスト見積りガイドブック, 2008 年 9 月.
[4] 社団法人日本情報システム・ユーザー協会（JUAS）： システム・リファレンス・マニュアル（SRM）第 2 巻.
[5] 経済産業省　情報システムの信頼性向上に関するガイドライン第 2 版, 2009.
[6] プロジェクトマネジメント学会　2005 年度第 2 回研究委員会フォーラム：プロジェクト計画における QFD 応用研究会活動報告 1(2005).

INDEX

欧文

ACTUM....201
All-Pair 法/Pairwise....36
ANSI/IEEE....17, 24
CFD....34
FTA/FMEA....115
FTFI....127
IEEE....127, 141
ISV/IHV....191
JIS X0129....47, 49, 202
JUAS....4, 206
QCD....3, 186
SPRINGAM....191
V&V....118, 157, 173

あ行

アジャイル....20
アドホック・レビュー....19
α・βテスト....45
異常値・特異値分析法....30
インスペクション....24
ウォークスルー....23
受け入れテスト....45
運用テスト....185
エラー推測....44
オフショア開発....175, 187

か行

開発ライフサイクル....142, 191
管理プロセス....180, 183
期待結果構造....124
基本設計工程....18
境界値分析法....29
共通フレーム....132
禁則処理....161
組み合わせ構造....124
組み合わせテスト....36, 125
グレーボックステスト....34
結合テスト....157, 167
原因結果グラフ....33
構成管理ツール....102
コンティンジェンシープラン....4

さ行

残存欠陥....206
残存欠陥数....18
残存欠陥密度....206
サンプリングテスト....42
システム運用....155, 181
システム適格性確認テスト....137
システムテスト....41, 157, 161, 167
システムプロファイル....8, 156, 202
システム要件....181
システムリスク・アセスメントシート....156
実運用環境....194
シナリオテスト....131
障害影響度指標....48
状態遷移テスト....37
正味棄却コスト....17
信頼性要求水準....8, 11
信頼性レビュー....179
信頼度成長曲線....18
制御フローカバレッジ....31
生産性環境変数....202
生産物量環境変数....202
製造工程....157
セキュリティ要件....164, 194
設計工程....169
設計品質....14, 104, 158
総合テスト....189

た行

タイミングテスト....39
代用特性展開表....49
妥当性確認....17, 173
Wモデル型開発....169
探索的テスト....45
探針テスト....43
単体テスト....17, 164, 168
チーム・レビュー....23
直交表....36, 125
ディペンダビリティ....4, 143
データフローカバレッジ....31
適応保守....47
デシジョンテーブル....32
テストアーキテクチャ設計....119
テスト観点....119
テスト観点表....117, 138

INDEX

テスト技法 28, 125
テスト計画 158, 161
テスト項目数 129
テスト実態分析 132
テストシナリオ 41, 131, 170
テスト設計 164
テスト戦略 163
テストタイプ 121, 137
テストツール 181
テストファースト 172
テスト密度 207
テスト網羅性 117
テスト網羅率 163
テスト要求分析 118
テストレベル 132, 172
統計的テスト 43
統合テスト 164
同値分割 29, 40
同値分割法 28
特異値 30
ドメインテスト 40
ドメイン分析法 40
トレーサビリティ 101, 114, 127

は行

パス・アラウンド 22
派生構造 122
ピア・デスク・チェック 20
ピア・レビュー 21
非機能要求 191, 197, 199
非機能要件 42
品質指標 188
品質展開 14, 103, 107, 114
品質特性 7, 51
品質評価 188
品質目標値 18
品質要求 47
フィージビリティスタディ 105
負荷テスト 132
複合障害 3
ブラックボックステスト 28, 131
プログラムコード 30, 127
プログラムソースコンペア 158
プロジェクトマネジメント 142, 187
フロントローディング 169
ペア・プログラミング 20
並行処理テスト 39
ペトリネットテスト 40
保守プロセス 183
ホワイトボックステスト 30

ま行

モデルチェッキング 32

や行

ユーザ受け入れテスト 45
ユースケーステスト 41
要求品質 105, 111, 114, 202
要件定義監査 185
要件定義工程 157, 175, 185
予防活動 13, 47

ら行

リグレッションテスト 143
レビュー 14, 18
ロギングミーティング 26
論理組み合わせ検証テスト 32

執筆・監修（敬称略）

主査：太田　忠雄　株式会社ジャステック
秋山　浩一　富士ゼロックスアドバンストテクノロジー株式会社
育野　准治　日本ユニシス株式会社
石井　信也　株式会社テプコシステムズ
岩切　　博　三菱電機株式会社
小野　直子　東京海上日動システムズ株式会社
大原　道雄　社団法人情報サービス産業協会（JISA）
加藤　和彦　NTT データシステム技術株式会社
木村　光宏　法政大学
黒澤　義次　株式会社 JAL インフォテック
鈴木三紀夫　TIS 株式会社
徳武　康雄　富士通株式会社
中田　雅弘　株式会社日立製作所
西　　康晴　電気通信大学
庭野　幸夫　株式会社ジャステック
水野　浩三　日本電気株式会社
藤瀬　哲朗　IPA/SEC
三毛　功子　IPA/SEC