システム構造に基づく効果的なテストシナリオの作り方
ーちょっとシステム構造に眼を向けてみませんか？ー

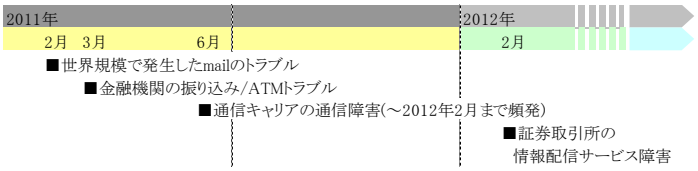
アブストラクト

1. 研究の背景／問題意識

情報システムのトラブルは社会基盤や我々の生活そのものを脅かす要因となる事もある。そのため情報システムを提供する者は、多面的な取り組みを恒常的に実施する事が求められている。しかし多大なコストをかけて開発、検証をしているはずであるのに本番環境でのトラブルは後を絶たない。

当分科会ではその原因の一つとして、テストシナリオ作成時に業務とシステム構造の関連性についての考慮が不足しているためであると考えた。

図表１　近年発生した社会的問題となった事象



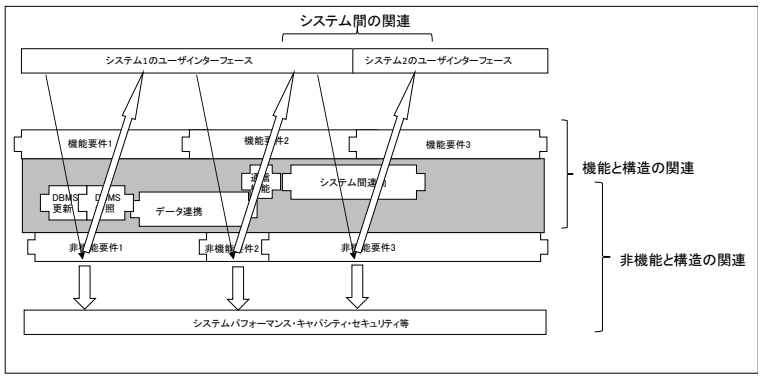
2. 研究の進め方

現状テストシナリオの作成方法は確立されていない、機能単位での個別のテストや業務の流れに沿った一連の機能確認に留まるなど、テスト品質にばらつきがある。当分科会ではシステム構造に着目して効果のあるテストシナリオ作成方法を探った。テストシナリオが抱える問題の背景には現在の統合テストがシステム構造部分をブラックボックス化していることが挙げられる。ユーザインターフェースを介した入力と出力の確認が中心で、システム間の連携や、システムで達成すべき非機能要件が見えていないからである。

解決にあたって、テストシナリオ作成時に問題となる点と、システム構造という切り口で考えられる視点を洗い出し、どの視点でとらえればどの問題の解決につながるかを検討した。その結果、三つの視点でアプローチすることで多くの課題を解決できると考えた。

(1) システム間の関連
(2) 機能と構造の関連
(3) 非機能と構造の関連

図表２　統合テスト解決策イメージ



3. 研究内容

3.1 システム間の関連

派生開発において、機能拡張を繰り返すことによるサブシステム間連携の複雑化と、短納期によるドキュメント作成・整理の遅れから、システム全体を意識したテストシナリオが作成できず局所的なテストに終始してしまうことが起こる。システム間連携の可視化を行えれば、こうしたことから起こるテストシナリオの漏れを防止することができると仮定した。

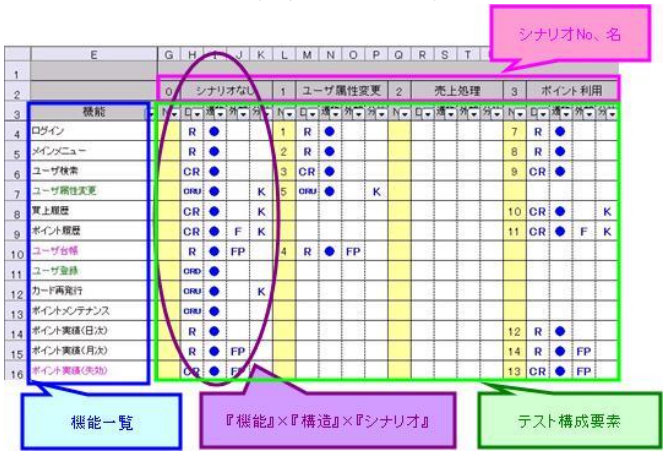
可視化を実現する方法として、「システム俯瞰図」の作成を提案する。これは「システム関連図」、「データインターフェース一覧」、「業務機能一覧」といった既存システムの開発時にはバラバラに作成される基本設計書の一部を保守、派生開発用ドキュメントとして融合させたものである。この資料を使用して分科会メンバーの機能拡張プロジェクトに適用した結果、既存システムに習熟していない担当者でも、より短時間で網羅性の高いテストシナリオが作成できる効果が確認できた。

3.2 機能と構造の関連

テストシナリオを作成する上で重要な情報は「業務要件（機能)」と「テスト構成要素」であり、その関連性に注目することでテストシナリオの品質向上を目指した。その実現手法として FOSS 図(Function Operation System Structure) の作成・活用を提案する。これは「業務要件」と「テスト構成要素」をマトリクスにしたものに、DB や画面遷移といったシステムの構造を加えたものである。

テスト　範囲の明確化、修正の影響調査漏れや有識者不在によるシナリオの漏れを防ぐ効果を、分科会メンバーのパッケージカスタマイズ案件の結合テストで検証を実施した。

図表３　FOSS 図



その結果、上流工程時に FOSS 図を活用することで、作業負荷軽減、工数削減、システム概要のナレッジ化等の効果が期待できることに加え、テストの規模や影響調査、注意点等が事前に把握できることで、要件定義時の見積りや影響調査に対して効果を発揮できることが明らかになった。

3.3 非機能と構造の関連

業務要件分析～設計時に「ハードウェア/処理構成」を検討する段階で、非機能要件に関するテストシナリオ抽出方法を構築することで、非機能要件に起因する障害を未然に防止することを目指した。そのために非機能要件テストシナリオを「ハードウェア/処理構成」ごとに分類した DB を作成し、必要なシナリオを抽出するためのガイドラインを用意した。これを実際のプロジェクトの過去の障害事例に当てはめて検証した結果、半数を超える事例について障害を防止できたことが実証された。また、副次的効果として、現在進行中のプロジェクトにも適用したところ、非機能要件部分のテストの工数把握を可能とすることができ、これらの重要性についての合意形成に役立てることができた。次案件において非機能要件部分のテスト工数の予算化に成功し、有意性のある結果を得た。

図表４　非機能要件テストシナリオ DB

NO	トリガー	テストシナリオ	テスト留意点
1-1	インプットの	整合性の確認	全UIの類似機能が一貫性がとれているかを確認する。対象ユーザにとって直感的に理解が可能かの視点で確認する
1-2	変更・新規・追加	エラー時の確認	予期せぬイベントに対処可能なUIとなっている(ヘルプの使用有無)。機能要件以外のエラー時のアクション。
1-3		データ入力確認	ディスプレイがサポートする最小の解像度に対してフォームやダイアログボックスのサイズが妥当か？マウスとキーボードのアクション(機能要件以外の入力で予期せぬ動きをしないか)。ショートカット、組み合わせ。
1-4			標準的な機能の使用(ドラッグ&ドロップ等)可否が明確か？
1-5		初期設定(状態)の確認	初期状態でのカーソル等の動きを確認
1-6		互換性の問題	想定される全ての環境下で確認を実施
1-7		パフォーマンスの確認	ピーク/オフピークでの処理時間の確認
1-8			最小/平均/MAXでのトランザクションでのレスポンス確認
1-9		セキュリティの確認	要注意な項目等を扱う場合、必要なアクセス権限が適切に設定されている必要がある
2-1	バッチ処理への	パフォーマンスの確認	処理時間の確認(日中/夜間での許容終了時間内で終わる裏づけがあるか)
2-2	変更・新規・追加	キャパシティ確認	バッチ処理中間バックアップやリラン対応の一時的なソートのために利用される資源は適切に確保できているかの確認
2-3		環境違いの確認	テスト環境と本番環境での違いはないかを確認できているか
2-4		他部分への影響	今回修正した部分が、(修正の必要がない)他の部分に全く影響を及ぼさないか確認は取れているか
2-5		本番並みのデータ確認	テストデータのバリエーションは十分か、またはピーク時のみ発生する特殊なデータの考慮はできているか

4 評価／提言

システム構造に着目した三つの視点で解決手段を考案し、検証を通じてそれぞれで一定の効果を上げることが確認できた。この結果が示すように、システム構造に着目することは、システム障害減少への一つの答えといえる。

業務要件だけでなくシステム構造との関連性に着目しそこに踏み込んだテストにより「障害抑止」を目指すべきである。