ソフトウェアテストシンポジウム 2004
JaSST'04: Japan Symposium on Software Testing 2004

事例に見る「非トランザクション型システム」の品質低下原因と効果的な改善対策

石川 和則†‡

†株式会社エス・キュー・シー 〒215-0021 神奈川県川崎市麻生区上麻生 1-7-14

E-mail: †kishikawa@sqc.co.jp

あらまし 非トランザクション型システムの検証事例を通じて得られた品質低下原因のまとめと考察.
キーワード 検証事例，アプリケーションシステム

The quality fall cause of a "non-transaction type system", and the example of the effective measure against an improvement

Kazunori ISHIKAWA†‡

†SQC Inc. 1-7-14 Kamiasao, Asao-KU, Kawasaki-Shi, KANAGAWA, 215-0021 Japan

E-mail: †kishikawa@sqc.co.jp

Abstract The conclusion of a quality fall cause and consideration which were obtained through the verification example of a non-transaction type system
Keyword Example of the Testing, Application System,

1. はじめに

近年の開発環境の進歩は、システム構築の低価格化と効率化を進展せしめ、特殊な技術を有するエンジニアや専門の業者に委託せずとも、利用者自身の手でシステムを開発できるというメリットをもたらしている。

しかし、このような環境が整備されたことのデメリットとして、十分に要件を検討しないままシステムを構築してしまう危険性があることは周知の事実であろう。

特に企業の業務システムの場合、上記の開発方法で構築されたシステムは、本来トランザクションに基づいて処理や承認が行なわれるべきところについても利用者の操作優先で設計されるなど、運用的な観点での脆弱性が懸念されるものが多い。

このような方法で構築されたシステムは、システムへの要求が大きく変化した場合には再構築することが前提だが、運用中のシステムを再構築することは様々な理由で難しいため、重大な欠陥を生じる度に何とか修正を施し、根本的な解決をはかれないまま運用を続けることが少なくない。

本論文は、このようなシステムを実際に検証し品質を向上させた事例から、問題を抱えたシステムを再構築するのではなく、維持したままで品質を向上する場合の方法や考え方について考察する。

2. 事例システムの紹介

本論文では、前述したシステムの事例として下記の２つの業務システムを対象としている。

2.1. 工場における在庫・出荷管理システム

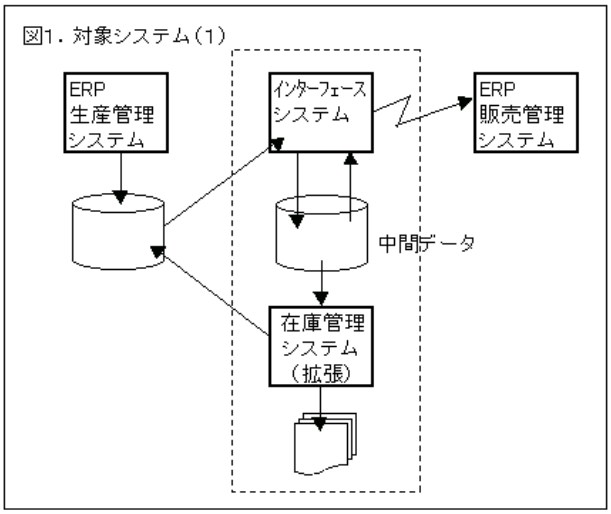
工場で生産された製品在庫に関して、工場倉庫内への入庫、倉庫間在庫移動、出庫、及び客先への出荷について管理し、日次、月次の在庫状況、出荷実績などをレポートする業務システム。

システムの前段となる生産管理システム、後段となる販売管理システムは ERP が利用され、当初はこれらの間を結ぶインタフェース・システムとして構築されたが、最終的には在庫管理システムとしての要求が大きくなり、様々な障害が発生した。（図１参照）

2.2. サービスに関する請求情報作成システム

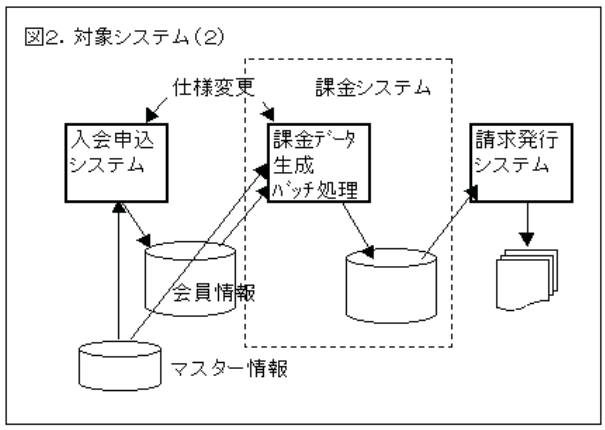
利用者に提供するサービスの内容や種別、申込情報、退会情報、その他営業が企画するキャンペーンなどの情報に基づいて、各利用者への当月分の請求額を算出・生成するシステム。

前段となる利用申込システムが先行して開発され、



当初はこのシステムのサブ･システムとなるバッチ処理として開発されたが、会員数の増大やサービス種類の追加などでシステムの規模が大きくなり、同じチーム内で運用が困難となり、システム的にではなく、運用的に２つのシステムに分割管理されることとなった経緯を持つシステムである。

当然、元となるデータは入会申込システムに依存しており、システム的な切り分けが難しく、様々な障害を生じる原因となっていた。（図２参照）



3. 発生していた問題の概要

上記の２つのシステムは、共に検証計画／実施時には稼動中で、既に様々な障害を抱えながら運用されていた。

このため検証の計画にあって、既に発生している既知の障害を調査することとした。

3.1. 各システムの障害の類別

各システムで発生していた障害を運用、システム管理および利用者の各部門からヒアリングし、これを類別したものが下表である。

表1．各システムで発生していた障害の類別

問題の類別	問題数(%) システム(1)	問題数(%) システム(2)
前段の操作ミスが影響して発生する障害	38	46
前段／後段システム間の仕様不一致による障害	22	5
データやマスター情報の認識不一致による障害	19	26
仕様変更時の相互理解の不足による障害	8	16
データ更新と取得タイミングの差による障害	8	0
その他の障害	5	7

この結果、対象とした２つのシステムは、どちらも前段システムの操作ミスやデータベース上のフラグなどに関する認識の誤りから、多くの障害が発生するという、類似した障害発生傾向を持つ事が判明した。

3.2. システムの障害対応の実際

次に、障害が発生した場合の対応方法について、各システム毎に調査したものが下表である。

表2．各障害の対処方法の実際

各問題への対処方法	回答数(%) システム(1)	回答数(%) システム(2)
ミスのあったデータを修正する(手作業)	75	88
出力されたデータを直接加工(手作業)	20	45
問題の発生したプログラムの対策／修正	15	5
ミスを起こした部門や利用者への注意や教育	8	13
仕様書やドキュメントの改善	2	10
システム間の連絡会やコミュニケーションツールの充実	5	4
根本的なシステムの見直し、再構築への取り組み	10	0

この結果、驚くべきことに、発生している問題のほとんどが、プログラム修正などによる障害発生の根本原因を対策するのではなく、あくまで対処療法として手作業でのデータ修正や出力帳票の加工が行なわれている実態も明らかになった。

また、これらの対処療法に頼らざるを得ない原因としては、システム運用の大半の工数が障害の対処に費やされ、結果としてシステムの根本的な見直しやプログラムの修正を行なう工数が捻出できないことが挙げられ、システム運用のコスト削減が叫ばれる中、上記のような対応を選択せざるを得ない苦しい運用の実態も合わせて明らかになった。

4. 品質向上アプローチの策定

上記で述べたように、システムの根本的な改善や再構築が望めない状況で、いかにしてシステム品質を向

上させるかを各システム現状の調査結果に基づき検討し、システム検証の方針として策定したものが下記の３つの改善アプローチである。

これらの方法は、改善の効果が高く、且つコストが低いアプローチを目指したものではあるが、抜本的な問題解決の方法ではない。

あくまでも、将来に抜本的な改善やシステムの見直しを行なう意味での「システムの延命策」であることを踏まえて提案を行なった。

4.1. 入力データリアルタイム検証スクリプトの配置

前段となるシステムで操作のミスが行なわれると、対象システムでは予期しない入力が発生して障害となってしまう。

これらは、本来「入力画面」や「入力チェック」などをシステムに実装すること、また入力データを後段システムが直接参照するのではなく、システム間に適切なトランザクションアプローチを挿入し、前段システムがトランザクション内容を保証するように改善すべきポイントである。

しかし、前述したシステムの実態を鑑み、ここでは入力データを逐次監視するテストスクリプトをシステムの外部に構築し、これを随時実行して入力ミスを検出するアプローチを提案した。

スクリプトの作成は、前段システムの外部設計書、要件書、及びインタフェース仕様書を用いて、ブラックボックステストのアプローチにより実施し、仕様変更があった場合はスクリプトの見直しを行なうことを前提とした。

また、テスト結果（発生した問題）はシステム運用部門の特定の管理者のみに通知され、そこで選別された上で関連部門や操作担当者に通知することとした。

4.2. データ加工、帳票加工履歴の管理

これまでは、各システムで個別に行なわれていた障害の対応を、各システムの責任者が承認した上で実施するルールを作成し適用した。

これも、本来はシステム間でインタフェースを保証することで完結するはずであるが、インタフェースが不明確な上、マスターデータなど共通で参照する情報が多いため、このようなルールを策定して提案することとしたものである。

この提案には、当初、責任者の負荷が増加する等の理由で反対意見も多く出されたが、インタフェースの保証ができない実態を鑑みて実施することとした。

4.3. システム仕様の整理統合と仕様検討会の開催

通常、新規システム開発では、上流から仕様の要求が出され、これに基づいて下流システムへ変更の要件が伝えられる。

しかし、既に運用中のシステムの場合には上流／下流含め、仕様変更の詳細を検討して影響範囲や実施の可否を判断する必要がある。

特に、本論文の対象とする２つのシステムは、前段システムの持つデータを後段のシステムが直接参照する構造が見られ、前段での変更が後段に大きな影響を及ぼす構造である。

そこで、前段、後段などのシステムに閉じて整理されているシステム仕様やデータベース設計などを再整理して、システム共通部仕様として整理統合することを提案した。

また、この仕様に影響する変更が発生した場合は、各システムの責任者出席のもとで「仕様検討会議」を開催し、各担当箇所への影響の有無や対応方法などを十分に検討して変更を承認するルールを定めることとした。

5. 検証結果と品質改善効果

これらの改善方法を実施する前後で、障害の発生状況を調査した結果を次ページの図３に示す。

図中から分かるように、システム（１）では対策実施後、特に入力操作のミスによる問題数の低減が著しく絶大な効果が上がっている。

また、仕様変更時の障害発生もある程度改善が見られ、認識不一致などに基づく障害数も減り、一定の改善効果が見られた。

このことから、提案した改善方法が類似システムに対して十分に品質向上効果があることを確認できたと考えている。

しかし、システム（２）に対する入力ミスによる障害が計画したほど改善されないなど、想定した効果が上がっていない部分もあり、品質向上の進め方や内容などには改善の余地がある。これらについては、今後の分析と見直しが必要であろう。

6. 考察

本論文で対象とした２つのシステムの内、システム（１）については、2003 年 10 月を持ってシステムの寿命を全うし、新たな設計に基づいたシステムへと移行が完了している。

また、システム（２）は、現時点でも苦しい運用が続いており、より一層の改善対策を求められている状況である。

これらの２つのシステムは、どちらも本来あるべきトランザクションによるインタフェース保証や、前段や後段システムとの依存関係の見直しをなど行なって

図3. 対策実施による障害発生数の低減効果

	各システムで発生した障害の類別	障害発生件数 対策実施前々月	対策実施前月	対策実施当月	対策実施翌月	品質改善効果（実施前／後比）
システム(1)	前段の操作ミスが影響して発生する障害	8	12	3	3	−70%
	前段／後段システム間の仕様不一致による障害	4	4	2	2	−50%
	データやマスター情報の認識不一致による障害	3	5	3	3	−25%
	仕様変更時の相互理解の不足による障害	1	1	1	1	0%
	データ更新と取得タイミングの差による障害	1	3	2	1	−25%
	その他の障害	0	1	0	1	0%
	合計	17	26	11	11	−49%
システム(2)	前段の操作ミスが影響して発生する障害	12	18	12	13	−17%
	前段／後段システム間の仕様不一致による障害	2	3	2	2	−20%
	データやマスター情報の認識不一致による障害	5	6	4	4	−27%
	仕様変更時の相互理解の不足による障害	3	3	2	3	−17%
	データ更新と取得タイミングの差による障害	0	0	0	0	なし
	その他の障害	5	1	3	3	0%
	合計	27	31	23	25	−17%

図4. システム(1)の対策前後での障害発生数の変化

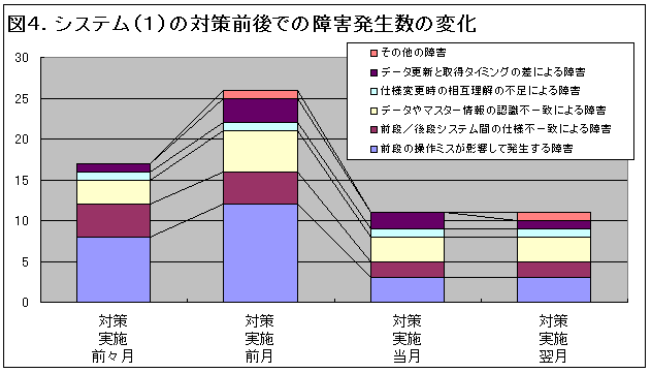
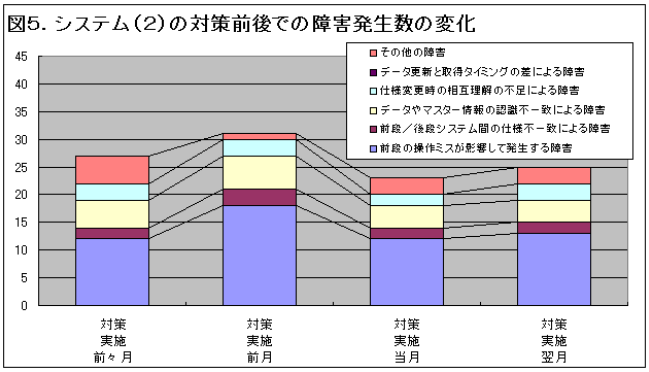


図5. システム(2)の対策前後での障害発生数の変化



将来的には再構築すべきシステムであったが、このようなシステムを利用する企業では様々な事情によって対処療法が求められることが少なくない。

また、効果があったとはいえ、今回提案したアプローチは特別な方法ではなく、旧来からシステム運用や管理の上で云われてきた一般的な方法をアレンジして使用したに過ぎないことも事実である。

しかし、一方では、このような例は前述の２つのシステムに限らず各所で散見され、多くは人海戦術などで暫定対策を講じているという実態もある。

これは、「抜本的な対策」としてシステムの再構築が必要性なことは云うまでもないのだが、現実問題として適切な対処療法や延命策すら講じられないシステムが多く存在していることを示している。

本論文は、品質向上の「現場の事例」を紹介したが、システム品質向上の実際を考える場合、理論的なアプローチとは別に対処療法的な方法も並行して検討して行く必要があるのではないかと感じるところである。

尚、現在も稼動を続けているシステム（２）については、今後も状況の監視を続け、より効果の高い提案を行なって行く予定である。

この結果については、別の機会に是非発表をしたいと考えている。

文　献

[1] Cem Kaner, James Bach, Bret Pettichord, ソフトウェアテスト 293 の鉄則, テスト技術者交流会（訳）, pp.1-380, （株）日経 BP, 東京, 2003.

[2] Cem Kaner, Hung Quoc Nguyen, Jack Falk, 基本から学ぶソフトウェアテスト, テスト技術者交流会（訳）, pp.1-471, （株）日経 BP, 東京, 2001.