IPA Better Life with IT

情報処理システム 高信頼化 教訓集

ITサービス編

独立行政法人 情報処理推進機構
社会基盤センター

情報処理システム高信頼化 教訓集

IT サービス編

独立行政法人情報処理推進機構（IPA）
社会基盤センター
https://www.ipa.go.jp/ikc/

目 次

3 技術領域の教訓 91

序 言

政府は 2013 年（平成 25 年） 6月 14 日の閣議で「世界最先端 IT 国家創造宣言について」(以下、宣言）をとりまとめた（本宣言は毎年改訂版が閣議決定されており、2016 年（平成 28 年）度版では宣言に基づく取り組みが着実に成果をあげているところとされている）。宣言は、景気長期低迷・経済成長率の鈍化による国際的地位の後退と、「少子高齢化、社会保障給付費増大、大規模災害対策等」、課題先進国と言われる日本の現状に鑑み、「成長戦略」の柱として、IT を成長エンジンとして活用し、日本の閉塞の打破、持続的な成長と発展を意図したものである。以降、５年程度の期間（2020 年までを目途）での実現を目指しており、具体的には、IT 総合戦略本部の下、政府 CIO により省庁の縦割りを打破し、政府全体を横串で通して、IT 施策の前進、政策課題への取り組みを行うとしている。その中で、IT 利活用の裾野拡大に向けた組織の壁・制度、ルールの打破、成功モデルの実証・提示・国際展開を図ることを意図としている。

ひと口で言うなら、政府 CIO の下、IT 施策の前進と政策課題への取り組みの障害になる縦割り行政など様々な行政上の課題を克服し、強力に IT 化を推進して世界最先端の IT 国家になることを目指すというものである。今後この宣言が実効をあげれば、政府主導による官民一体となった IT 化が推進されるものと期待される。

そして、その戦略が実を結び、輝かしい IT 基盤社会を到来させるためには、IT 産業を取り巻く課題にも注視する必要がある。例えば、我が国の IT 社会には、宣言で述べている「各種施策の推進力を妨げる障壁」以外にも、IT 化を推進する上での根本的な開発マネジメント能力、プロジェクト推進能力上の問題が指摘されている。その結果、ソフトウェアやシステム上の不具合に起因した障害がしばしば報道され、社会問題ともなってきたことは周知の事実である。したがって、真に世界最先端 IT 国家創造を進めるには、情報システムや社会システム構築における脆弱さの克服にも注力しなければならない。

しかも、社会的に大きな問題となった情報システムの障害は、社会に大きな影響を与えてもその実態が社会的に共有できる状況にない。ましてや、障害から学んで類似事故の再発防止に活かすというメカニズムもできていない。回転ドアやエレベータ等で発生した事故は、大きな社会問題とされ、その後の第三者を交えた事故調査を経て再発防止策が構造規格として法律的に整備されたのと比べると、社会的影響が甚大な情報インフラにおける障害に対する克服の体制や再発防止のためのシステムは十分とは言えない。

先端 IT 国家たるには、「IT システムがもたらす障害の科学的分析」や、「障害なしの IT 構築に向けての科学的方法論の研究開発」という地味ではあるが本質的な分野に対する取り組みを世界に先駆けて行う必要がある。

このような状況を背景に独立行政法人情報処理推進機構 (以下、IPA) では、2013 年 4 月より「重要インフラ IT サービス高信頼化部会」（2018 年度からは「情報処理システム高信頼化部会」。以降、単に「部会」と総称）を組織し、重要インフラシステム等の障害情報の収集・分析、再発防止策の導入促進、障害事例に対する対策支援を進める取り組みを開始した。情報システムにかかわる障害情報を幅広く収集・分析する仕組みが整備されるなら、その実態を解明し、そこから多くの教訓を引き出して類似障害の再発防止に貢献することができる。情報システムの障害について、業種や分野を越えて横断的に対策や教訓を分析・共有する取り組みに関して、公的な解明の機関や体制を備えている事例は、国際的にもほとんどない。この面で、IPA の当部会の取り組みは、世界に先駆けた事業としての意義もある。

実は、発生した障害の徹底した分析や、再発防止への真剣な努力については、その企業内ではもち

ろんのこと、関係機関と共同での真剣な取り組みも行われている。もし、これらの検討成果が広く水平展開されるなら、産業界に大きく貢献すると考えられる。

このような状況認識の下、当部会では、障害事例情報の収集と分析による『教訓』の発掘に加え、各企業で個々に分析解明され、『教訓』として企業内共有されている事例の収集と、それら『教訓』の社会的共有という方向性を定め具体的な作業を開始した。当部会には、社会の重要インフラにかかわる企業の CIO クラスの方々に参画していただき、教訓的事例のサーベイと内容の吟味・検討を行ってきた。ここで審議された豊富な事例の中から、社会に普及し学ぶべき価値のあるものを教訓として体系化してまとめた「情報処理システム高信頼化教訓集（IT サービス編）」を IPA の Web ページにて公開し、その普及に努めてきた。その中で、2015 年 12 月より「教訓リンク集」として個々の教訓を個別にダウンロードできるようにした。また、2017 年度には、これまでまとめた教訓および別途収集している「情報システムの障害状況一覧」の障害事例について、『注意すべき観点』に基づいて分類した「障害事例の一覧」を作成し Web で公開した。

今回、 6年間に及ぶこれら活動の総まとめとして、教訓や障害事例などの解説も交え、「情報処理システム高信頼化教訓集（教訓集）」として出版することになった。これは、独立行政法人情報処理推進機構および関係する方々のご尽力の賜物であり、深く感謝したい。

あわせて、当部会の取り組みを、関連する業界団体や地域団体を中心に展開する試みも行われ、活動の広がりとともに本事業の意義についての理解も広がったものと思われる。今後もこのような活動がより広く浸透し、教訓が社会的に蓄積され活用される仕組みの構築へとつながればと考えているが、その中で、本教訓集が大きな役割を果たすものと期待している。引き続き皆様方のご支援とご協力をお願いしたい。

2019 年2月

情報処理システム高信頼化部会
主査　　中村　英夫
日本大学　名誉教授

1

はじめに

1.1 背景と課題

情報処理システムは、銀行や証券などの金融サービス、各種手続きのための行政サービス、ソーシャルネットワーク等の情報通信サービス、交通機関の運行制御など、私たちの生活や社会・経済基盤を支える重要インフラ分野等の IT サービスに深く浸透し、ひとたび障害が発生するとその影響は非常に大きい。私たちが安全で安心な生活や社会・経済活動を続けるためには、重要インフラ等を支える IT サービスにおける一層の信頼性向上が求められる。

社会インフラに影響を与えマスコミ等で報道された IT サービス障害情報を IPA は継続的に収集している。2017 年の IT サービス障害の発生状況（文献 1－1）は高い水準となっており、調査を開始した 2009 年から引き続き増加傾向にある（図 1.1－1）。

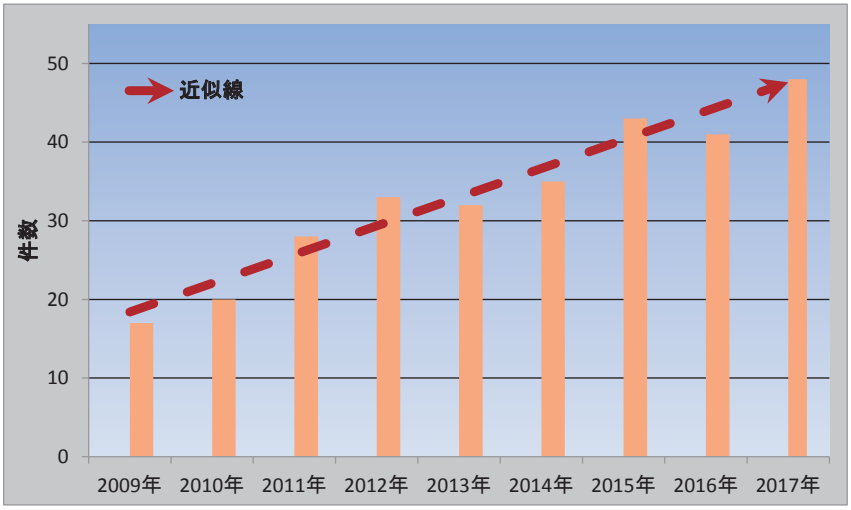


図 1.1－1　報道された IT サービス障害発生件数の推移

従来、情報処理システムの障害に対する原因分析と再発防止対策の実施は、多くの場合、当事者においてのみ行われ、その情報は公開されて来なかった。そのため、他事業者のシステムにおいて、あるいは他業界・分野のシステムにおいて、類似の障害が発生することがあった。

情報処理システムの構築・運用やその管理は、社会や技術の進展につれて複雑化・多様化しており、一人や一企業のカバーできる範囲には限界がある。そして、その範囲は今後ますます狭くなっていくことは明らかである。したがって、情報処理システムの構築・運用及びその管理にかかわる課題を解決するために、より多くの人たち・企業の経験を社会全体で共有・伝承することが求められている。近年、システム障害の発生に際して、発生直後の速報から完了まで各事業者がホームページ等で逐次公開するケースが増えており、システム障害に対する説明責任の重要性への認識が広まってきている。この動きがさらに進展することを期待したい。

1.2 IPA の取り組み

IT サービスの一層の信頼性向上に向けて、各企業等の経験と情報を社会全体で共有することにより、類似障害の再発を防ぐことが大切である。そのためには、各企業等が個別に保持する『経験知』／『暗黙知』を『形式知』化する必要がある。そこで、重要インフラ等を担う事業者等の協力を得ながら、IT サービスの障害事例情報の分析と対策の検討、既に実施済の対策を含めた整理・体系化等を行ってきた。この取り組みは、その結果得られる「教訓」を業界・分野を越えて幅広く共有し、類似障害の再発防止・影響範囲の縮小につなげる仕組みの構築を目指すものである（図 1.2－1）。

また、このようにして導かれる教訓の有用性を示すとともに、障害情報の分析等により導かれる教訓を業界・分野横断で共有するための、障害情報や教訓の共通様式、それらの収集・公開に際しての機密保持等のルールについても、あわせて取りまとめた。さらに、得られた教訓に基づき、類似障害の再発防止に向けたシステム構築や運用・管理の継続的なプロセス評価・改善手法等を取りまとめるとともに、広くその導入を促進する活動にも取り組んできた。

なお、IPA の取り組み内容については、Web サイトをご覧いただきたい。[1]

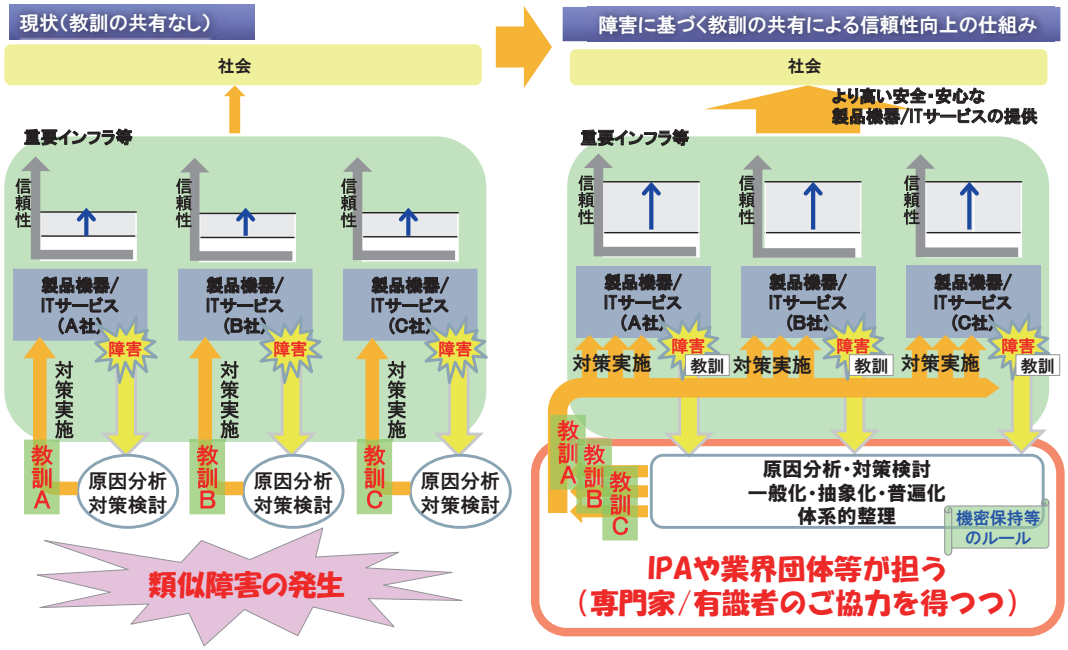


図 1.2－1　障害事例に基づく教訓の共有による信頼性向上の仕組み

[1] 重要インフラ分野のシステム障害への対策　https://www.ipa.go.jp/sec/system/index.html

1.3 具体的活動

IPA は、公的機関として、IT サービスを担う情報処理システムの主としてソフトウェアに起因する障害事例関連情報を収集し、それらの分析や対策手法の整理・体系化を通して得られる「教訓」を業界・分野を越えて幅広く共有し、類似障害の再発防止や影響範囲縮小につなげる仕組みの構築を目指した取り組みを、2013 年度より開始した。

具体的には、重要インフラ分野等の企業からの情報提供や有識者からのヒアリング等により、過去の障害事例情報を収集した。並行して、複数の重要インフラ分野等の有識者・専門家の委員を中心とする委員会である「重要インフラ IT サービス高信頼化部会」（2018 年度からは「情報処理システム高信頼化部会」。以降、単に「部会」と総称）を設置し、収集した障害事例情報の分析と対策の検討を行い、それらを「教訓」として一般化・抽象化した。それを取りまとめたものが、本書である。（図 1.3－1）

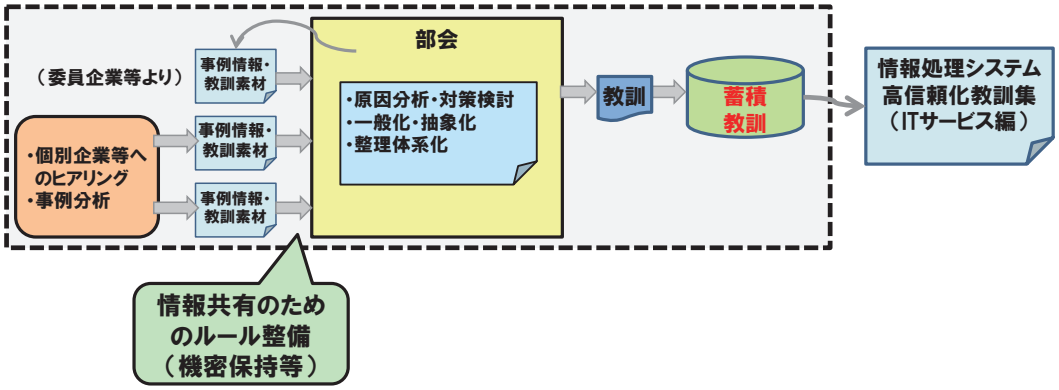


図 1.3－1　今回の障害事例情報の収集と教訓化活動

1.4 本活動及び本書の特徴

(1) 詳細な障害事例情報の分析

2008 年から 2009 年にかけ、経済産業省、IPA、一般社団法人 日本システム・ユーザー協会（JUAS）及び協力企業により、重要インフラシステムの信頼性に関する研究活動を実施した。（文献 1－2、文献 1－3）この活動では、主に報道された障害事例情報 100 件近くを整理し、障害の再発防止策等を取りまとめた。

部会では、所定の機密保持ルールの下で、未公開情報を含む障害事例及びその再発防止策等の提供を受け、その分析等のための議論を行った。

(2) IT ガバナンス／マネジメント領域にも踏み込んだ教訓化

組込みシステム系の取り組みとしては、これまで、個々の障害事例に関する原因と対策の分析として、主に技術的側面に焦点を当てた取り組みが行われる傾向が見られた。

IT サービス系では、各障害事例についての技術的視点から導かれる原因・対策よりも、障害の背後に存在する要因（組織が絡むガバナンス／マネジメント要因）がより本質的な場合も多い。今回、マネジメント層の委員も参加した部会での議論結果を取りまとめた「教訓」には、主に技術者層向けの技術領域のものに加え、主にマネジメント層向けのガバナンス／マネジメント領域のものを含む。

(3) 業界・分野を越えて活用可能な教訓

IT サービスでは、環境や背景等、その置かれた周辺状況（以下、コンテキスト）は多様であり、障害の物理的現象は同じであっても、コンテキストが異なると対策も異なってくる。従来の類似活動では、コンテキストの観点での検討が十分でなかったことから、“どんな IT サービスも一括り”に捉えた対策として示されることが多かった。そのため、対策の活用者の立場では、次のように、十分に有効活用されないことが多かったと考えられる。

- 対策内容が抽象的過ぎていたり、当たり前なものであったりしがちである。
- 自分のシステムでは関係ないと考えられることが多い。

今回、多くの分野（通信、銀行、証券、保険、鉄道、電力、政府・行政など）の専門家により、障害事例の原因や対策について検討した。各部会委員は、紹介された各障害事例について、自分野のコンテキストに照らして、どのような事象が考えられるか、また、どのような対策が参考となるか、といった観点で議論した。このように、障害事例の背景や環境にまで深く踏み込んで分析し一般化・抽象化したため、IT サービスの事業者にとり、業界・分野を越えて、類似障害の発生防止対策として役立つと考える。

1.5 本書の使い方

普段の活動においては、通常，どこかのシステムで不幸にも何らかの障害が発生し、その情報を入手した時点で、それに基づいて自システムの（緊急）点検を行う。そのときに得られる情報は十分に詳細なものではないことが多く、また、時間の経過とともに徐々に詳細化されていくこともある。その時々に得られる情報の詳細度の範囲で点検を行い、その結果、必要に応じてしかるべき対応を行うことになる。したがって、一つの障害発生について時間を置いて複数回にわたって点検を行うこともある。

このような点検や対応の範囲や観点は、自システムのライフサイクルにおける段階により異なる。例えば、開発前であれば、類似障害が発生しないような対策を施すよう、プロセスを含む開発計画に盛り込むことになる。また、運用中であれば、類似障害の発生するリスク要因がないか確認したり、あるいは発生時の影響を見積もったりする。

もちろん、障害の原因によっても、点検や対応の重点、実施部門が異なるのは、言うまでもない。あるいは、障害の発生したシステムの応用分野によっても、自システムと同じ分野であればより慎重になる等、点検や対応への“心構え”が異なるかも知れない。

さて、本書は、過去のシステム障害事例に基づく教訓を集めたものであり、教訓により異なるものの、いずれも対象とする障害の発生からかなり時間が経過している。ただし、教訓に含まれる情報としては、障害の原因や再発防止の対策が整理されたものとなっている。特に、多くの応用分野に適用可能なように、一般化・抽象化された普遍的な内容となっている。

したがって、本書の使い方としては、自システムのライフサイクルにおける特定の段階で、可能ならば、あらかじめ社内の開発・運用標準に規定された段階で、自システムのリスク評価に用いることが想定される。例えば、チェックリストのように、一つ一つの教訓について確認していくのである。個々の教訓に関する確認の観点は、図 1.5－1 に示すように、

- 自システムで類似の障害は起きないか?
- 類似障害の発生防止のため、あらかじめ実施しておくべき対策はないか?
- 万一類似障害が発生した場合の影響は、どの程度か?
- 類似障害発生時に影響範囲を小さくする対策はあるか?

といったようなこととなる。

図 1.5－1　障害発生時の点検例

ところで、上述のように、本書に掲載されている各教訓は、様々な応用分野・業界に適用可能なように一般化・抽象化されている。すなわち、障害事例に含まれる本質（エッセンス）を保持しつつ、そのコンテキスト（背景や前提となる環境等）は普遍化されて説明されている。したがって、特定の分野・業界に属する各企業や組織において本書中の教訓を活用し上記のような点検や対応を行う場合には、教訓の説明に含まれている共通コンテキストと自システム・組織のコンテキストとを比較・照合し、適用可能と判断した教訓について、教訓中の対策を参考に、自身のコンテキストに合った具体的対策を検

討・実施することになる。（図 1.5－2）

もちろん、対策の適用にあたっては、そのために必要なコストと、対象システムの重要度あるいはその障害時の影響度を勘案し、一定のリスクを認識した上で、敢えて対策を実施しないという選択肢はあり得る。すなわち、対策実施に際してのリスク評価が重要である。

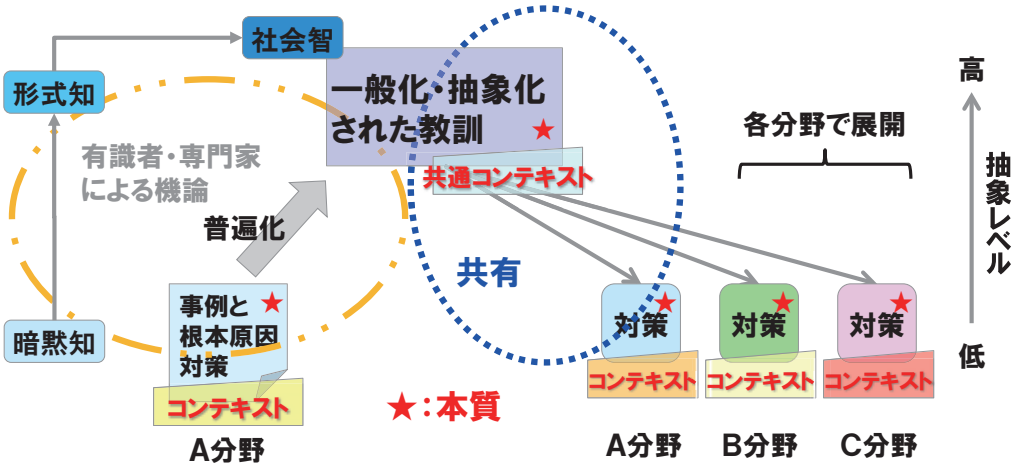


図 1.5－2　教訓の作成と活用の流れ

このような作業を適切に行うためには、“想像力”が要求される。教訓に含まれる本質を正しく見極め、それが自システム・組織のどこに対応するかをじっくりと考える力が必要である。そして、教訓ごとに着実にこの作業を実施するというシステマティックな取り組みが、その企業や組織のシステムの信頼性向上のために極めて重要である。

以上のような取り組みを進める中で、適当な時期に、各社の組織体制や開発・運用標準等に、教訓の内容のうちから従来不十分であった事項を反映する。これにより、教訓を活用した半永久的な対策が施されることになる。

実際に実施される対策のカテゴリとしては、図 1.5－3 に示すように、システム構築・運用の役割に応じ、いくつか考えられる。すなわち、ガバナンス／マネジメント系の教訓に関しては、経営層から業務部門を中心とする「組織・体制の整備」、業務部門を中心とする「運用手順の整備」があげられる。技術系の教訓に関しては、業務部門の IT 担当からシステム部門、ベンダに至るまでを対象とする「開発手順の整備」や、システム部門からベンダを中心とする「レビュー・テスト項目」の充実があげられる。また、調達に関係する教訓については、業務部門を中心とする「調達時の指示事項」及び「調達時の確認事項」の明確化があげられる。また、異なった種類の活用方法としては、トラブル発生時に、その事象内容と教訓に記載されている問題の内容とを比較することにより、その原因の推定に利用することも可能である。

さらに、これら全般について、社内教育のための教材として利用することも有効である。IT 人材は、

一般に、システム構築など大型プロジェクトを経験することにより育つが、近年はシステムの改修・維持保守案件が多くシステム構築から学ぶことが少なくなっている。このような状況において、各企業・組織では、幅広い経験に基づいて得られた本書の教訓を人材育成に活用することにより、若手担当者に技術を伝承することが可能となる。

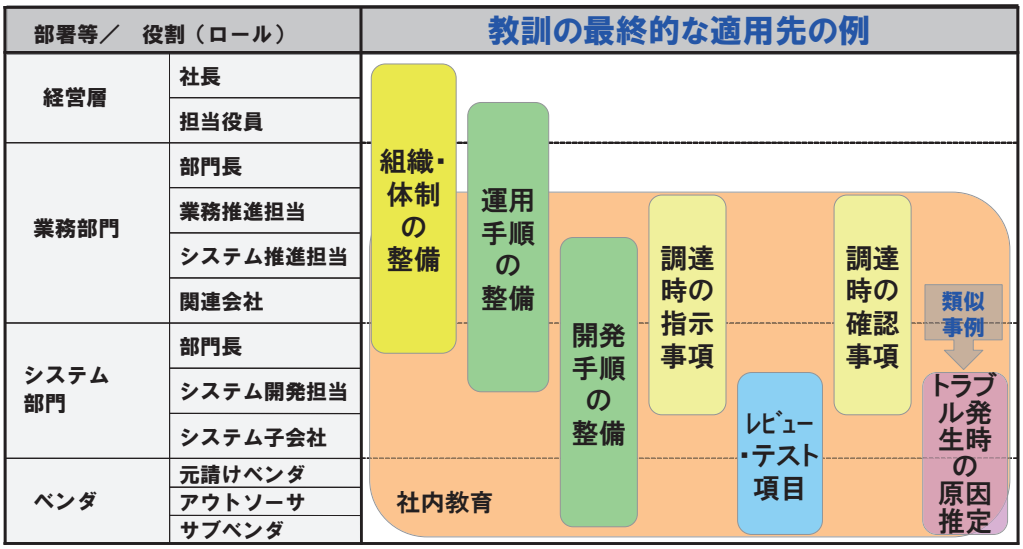


図 1.5－3　教訓の最終的な適用先の例

なお、同様な取り組みは、各組織内で発生する障害あるいはヒヤリハットの事例を対象にして行うことができる。

以上の内容に関しては、より具体化した上で、「情報処理システム高信頼化教訓活用ガイドブック（IT サービス編）」として取りまとめている。（文献 1－4）

最後に、本書が各組織における IT サービスの高信頼化に役立てられ、社会の人々がより安全・安心な IT サービスを享受できるようになることを期待したい。

1.6 本書の構成

本書では、以降、ガバナンス／マネジメント領域の教訓と技術領域の教訓を、それぞれ、第 2 章と第 3 章で説明している。なお、技術領域の教訓でも、一部にマネジメント的要素を含むものがある。また、収集した障害事例を分析しその傾向について第 4 章で説明する。第 5 章ではこのような教訓の共有などについて述べる。

本書に収録されている教訓の一覧を、表 1－1 に示す（以降、説明のため、各教訓には、“Gnn”あるいは“Tnn”の教訓 ID を付与する）。各教訓は、そのタイトル（『教訓概要』）に続き、障害事例の内容（『問題』）、その問題を引き起こした原因の分析結果（『原因』）、問題の原因を取り除き再発を防止するための対策（『対策』）、対策の実施により見られた／期待される効果（『効果』）、得られた教訓の補足説明（『教訓』）の 5 項目に整理した。

表 1－1　本書に掲載されている教訓一覧

教訓 ID	教　訓　概　要
ガバナンス / マネジメント領域	
G1	システム開発を情報システム部門だけの仕事にせず、各事業部門が自分のこととして捉える「態勢」を作ることが大切
G2	発注者は要件定義に責任を持ってシステム構築にかかわるべし
G3	運用部門は上流工程（企画・要件定義）から開発部門と連携して進めるべし
G4	運用者は少しでも気になった事象は放置せず共有し、とことん追求すべし
G5	サービスの拡大期には業務の処理量について特に入念な予測を実施すべし
G6	作業ミスとルール逸脱は、個人の問題でなく、組織の問題！
G7	クラウド事業者と利用者が連携した統制がとれたトラブル対応体制を整備すべし
G8	共同利用システムでは、非常時対応を含めて利用者間の情報共有を図ること
G9	システム利用不可時の手作業による代替業務マニュアルを作成し定期的な訓練を行うべし
G10	関係者からの疑義問い合わせは自社システムに問題が発生していることを前提に対処すべし！
G11	システムの重要度に応じて運用・保守の体制・作業に濃淡をつけるべし
G12	キャパシティ管理は、業務部門とシステム部門のパートナーシップを強化するとともに、管理項目としきい値を設定して PDCA サイクルをまわすべし
G13	キャパシティ管理は関連システムとの整合性の確保が大切
G14	設計時に定めたキャパシティ管理項目は、環境の変化にあわせて見直すべし
G15	保守作業は「予期せぬ事態の発生」を想定し、サービス継続を最優先として保守作業前への戻しを常に考慮すること
G16	本番環境へのリリースは、保守担当が無断でできないような仕組みを作るべし！
G17	サービスの重要度を識別し、それに応じた連絡体制や障害検知のしくみを作れ
G18	障害対策とは許容時間内の回復や停止中の業務継続まで具体化すること
G19	みんなで唱和！障害減らす教訓共有
G20	「システム運用環境変更時の品質向上」は正攻法の成功事例に学べ！
G21	サーバ証明書等の有効期限の確認方法を工夫せよ

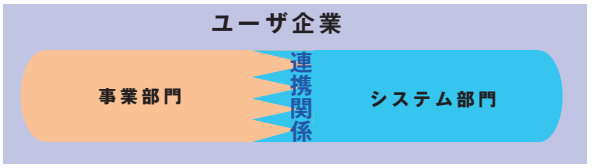
教訓 ID	教 訓 概 要
技術領域	
T1	サービスの継続を優先するシステムにおいては、疑わしき構成要素を積極的にシステムから切り離せ（“フェールソフト”の考え方）
T2	蟻の目だけでなく、システム全体を俯瞰する鳥の目で総合的な対策を行うべし
T3	現場をよく知り、現場の知識を集約し、現場の動きをシミュレートできるようにすべし
T4	システムに影響する変化点を明確にし、その管理ルールを策定せよ
T5	サービスの視点で、「変更管理」の仕組み作りと「品質管理責任」の明確化を！
T6	テスト環境と本番環境の差異を体系的に整理し、障害のリスク対策を練る
T7	バックアップ切替えが失敗する場合を考慮すべし
T8	仮想サーバになってもリソース管理、性能監視は運用の要である
T9	検証は万全？それでもシステム障害は起こる。回避策を準備しておくこと
T10	メッシュ構成の範囲は、可用性の確保と、障害の波及リスクのバランスを勘案して決定する
T11	サイレント障害を検知するには、適切なサービス監視が重要
T12	新製品は、旧製品と同一仕様と言われても、必ず差異を確認！
T13	利用者の観点に立った、業務シナリオに即したレビュー、テストが重要
T14	Web ページ更新時には、応答速度の変化等、性能面のチェックも忘れずに
T15	緊急時こそ、データの一貫性を確保するよう注意すべし
T16	システム構成機器の修正パッチ情報の収集は頻繁に行い、緊急性に応じて計画的に対応すべし
T17	長時間連続運転による不安定動作発生の回避には定期的な再起動も有効！
T18	新たなサブシステムと老朽化した既存システムとを連携する場合は両者の仕様整合性を十分確認すべし
T19	リレーショナルデータベース（RDBMS）のクエリ自動最適化機能の適用は慎重に！
T20	パッケージ製品の機能カスタマイズはリスクを認識し 特に必要十分なチェック体制やチェック手順を整備して進めること
T21	作業ミスを減らすためには、作業指示者と作業者の連携で漏れのない対策を！
T22	隠れたバッファの存在を把握し、 目的別のしきい値設定と超過アラート監視でオーバフローを未然に防止すること
T23	障害監視は、複数の観点から実装し、障害の見逃しを防げ！
T24	サービス縮退時の対策を考慮せよ
T25	障害原因が不明でも再発予防と発生時対策はできる
T26	既存システムの流用開発はその前提条件を十分把握し、そのまま利用可能な部分と変更する部分を調査して実施する
T27	パッケージはサポートを買え
T28	パッケージを更新する時は、変更内容の詳細確認と回帰テストで二重に安全を確保せよ
T29	単位などの定義が異なる制限値、連携するシステム間で使っていませんか？
T30	意味がない、一緒に束ねた 2 重化配線！
T31	復旧手順は、システムとその環境の変化に対応させ常に最新に！
T32	周期処理、「時間」と「変化」を監視せよ！
T33	入念な方式設計と多段階の確認は当たり前、個人情報を扱う場合には特に排他制御に気をつけて

2

ガバナンス / マネジメント領域の教訓

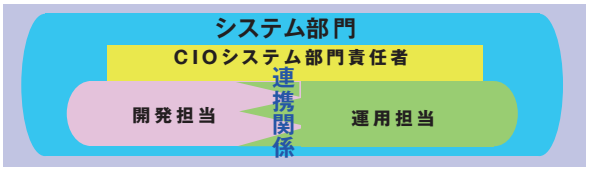
今回、得られたガバナンス／マネジメント領域の教訓は図２－１のように大別できる。いずれも組織間の連携関係にかかわるものである。

1 ユーザ企業内組織（事業部門－システム部門）間連携関係



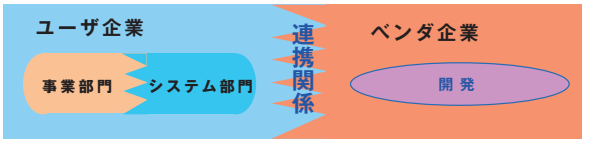
教訓 ID
教訓 G1　教訓 G12
教訓 G6　教訓 G13
教訓 G9　教訓 G14
教訓 G18

2 システム部門内の連携関係



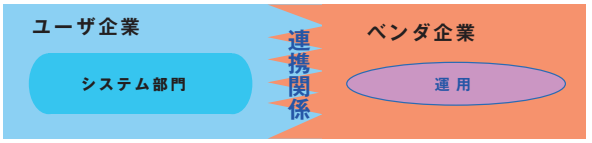
教訓 ID
教訓 G3
教訓 G4
教訓 G15
教訓 G19、教訓 G20

3 ユーザ企業とベンダ企業との連携関係



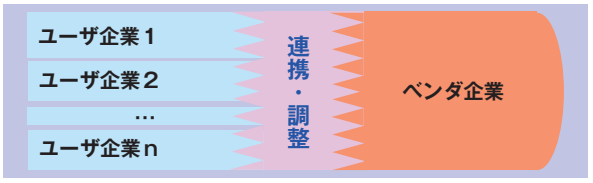
教訓 ID
教訓 G2
教訓 G16
教訓 G17

4 システム部門（運用）とベンダ企業の連携関係



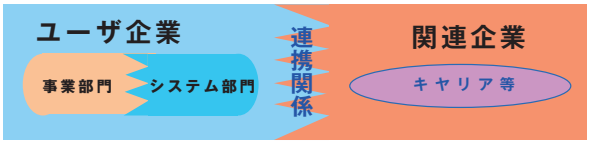
教訓 ID
教訓 G7
教訓 G11
教訓 G21

5 共同利用時のユーザ企業間の連携関係



教訓 ID
教訓 G5
教訓 G8

6 ユーザ企業と関連企業の連携関係



教訓 ID
教訓 G10

図２－１　組織内の連携関係とガバナンス／マネジメント領域の教訓との対応

以降の各節では、上記の各教訓について解説する。

2.1 事業部門と情報システム部門の役割分担に関する教訓（G1）

教訓 G1　システム開発を情報システム部門だけの仕事にせず、各事業部門が自分のこととして捉える「態勢」を作ることが大切

問題

Windows95 の登場、PC の廉価化、インターネットの普及等をきっかけに、90 年代後半から企業の活動に IT が不可欠となるだけでなく、競争の激化にともない、商品の複雑化が進み、すべての商品がシステム開発をともなう形で出されることとなり、従来とはレベルの違う量のシステム開発が行われるようになってきた。

しかしながら、急激に増大したシステム開発を成功させるためのポイントも十分掴めないまま実施したことにより、システムトラブルが多発し、企業の本来活動に支障をきたし、経営問題となった。

原因

コンサルタントも入れた分析の結果、システムトラブルの 8 割は、上流の要件定義局面でのコミュニケーション・ギャップから問題が生じていることが判明した。コンサルタントからは、以下のような問題事象も指摘された。

① ビジネス側の要件の確定が遅い（期日までに決めなくてはいけないというマインドが乏しい）。
② 要件の変更が多い。
③ 要件を最終的に文書で確認していない。
④ その要件が他システムにどのような影響を与えるかの分析が甘い。
⑤ 要件が設計に正しく反映されたかを複数の眼でチェックしていない。

④、⑤については、情報システム部門内での仕事の進め方に関する問題であり、これらについても抜本的な改善が必要であるが、①～③については、事業部門が「システム開発を情報システム部門の仕事」として任せきりにするのではなく、事業部門も情報システム開発において一定の役割と責任を果たすようにしないと、本質的に解決できない問題であると考えた。

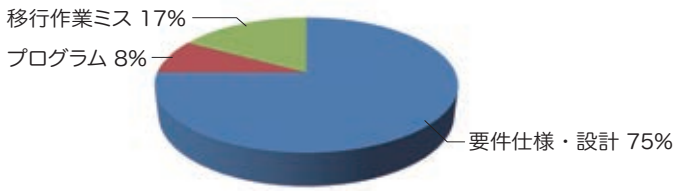


図 2.1－1　システムトラブルの原因 [2]

[2] 東京海上日動火災保険株式会社の例、1999 年

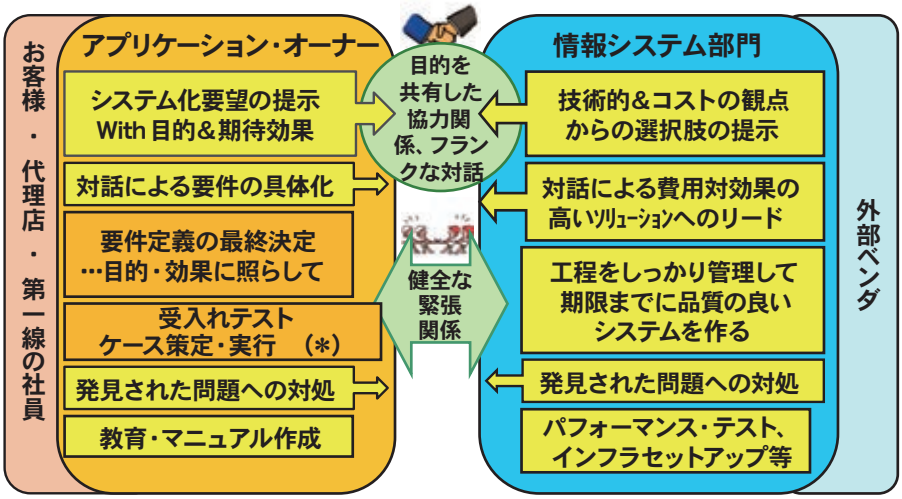
対策

この問題を解決するために、システム開発におけるビジネスサイドの役割と責任を明確化し、コミュニケーションの質を高めることとした。

そのためには、お互いの理解を補正し合い、「言いたいこと」と「聞いて理解したこと」が同じになるまで、対等な立場でオープンなダイアローグを繰り返すことができる「態勢」づくりが必要である。この態勢を「アプリケーション・オーナー制度」と呼ぶ。アプリケーション・オーナー制度のポイントは以下のとおり。

① システム開発は、情報システム部門に任せきりにすべき仕事ではなく、自分の考えた商品や施策を具体化するために行う自分自身の仕事であるという「オーナーシップ」の考え方を持たせる。
② 事業部門に、要件の詳細が固まるまで、情報システム部門と対話を繰り返す責任を持たせ、要件定義の最終責任を負わせる。
③ 事業部門に、要件定義どおりにシステムが出来たかどうか受入れテストを実施する責任を負わせる。

→「事業部門は要件定義に責任を持つこと」とするだけでは、表面的な責任に留まり、要件の揺り戻し等の問題は解消しないので、要件を定義した以上、その要件どおりシステムが出来たかどうかを確認する UAT（User Acceptance Test、ユーザ受入れテスト）のテストケースを作ってテストすることが重要である。実際に手を動かす（ハンズオンの）責任にしない限り、本当に責任を取ることにならない。

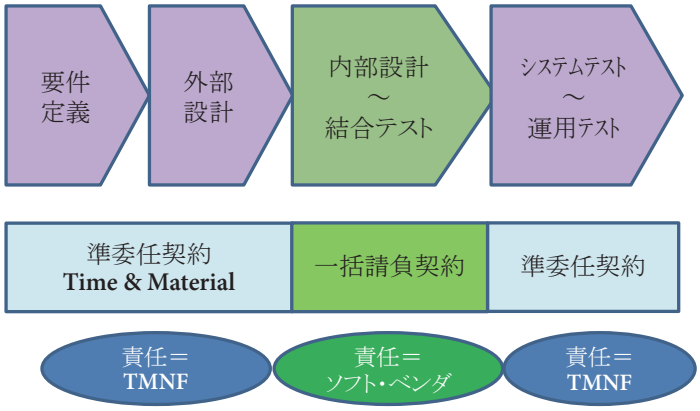


＊　要件定義に責任を持つ以上、要件通りできたかの受入れテストも実施することが重要。
このように手を動かす責任にしない限り、表面的なものになる。

図 2.1－2　アプリケーション・オーナー制度：責任と役割分担 [3]

[3] 東京海上日動火災保険株式会社の例

因みに、この制度に基づく、東京海上日動火災保険株式会社の情報システム部門とソフトベンダとの契約関係は以下のとおりである。この責任態勢は、アプリケーション・オーナーが要件定義～外部設計の工程、及びシステムテスト～運用テストの工程に責任を持つことで裏打ちされている。



※図中の TMNF（Tokio Marine Nichido Fire）とは、東京海上日動火災保険株式会社のこと

図 2.1－3　情報システム部門とソフトベンダの契約関係 [4]

効果

2000 年からアプリケーション・オーナー制度を導入したことにより、東京海上日動火災保険株式会社では 2000 年度から 2001 年度にかけてシステムトラブルが 8 割削減され、以降 10 数年、同じ水準を保っている。

教訓

システム開発を情報システム部門だけの仕事にせず、各事業部門が自分のこととして捉える「態勢」を作ることが大切。因みに、「態勢」とは以下のとおり。

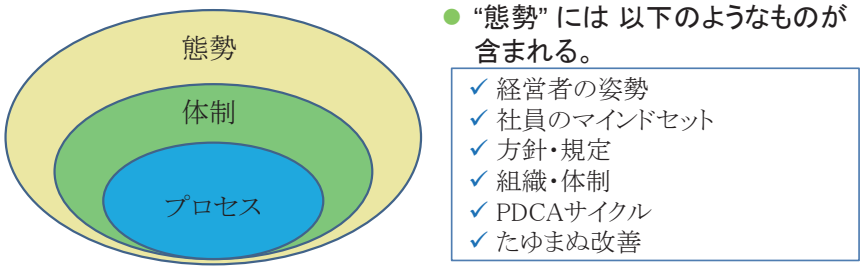


図 2.1－4　態勢・体制・プロセスの関係 [5]

[4] 東京海上日動火災保険株式会社の例
[5] 東京海上日動火災保険株式会社から提供

2.2 発注者の要件定義責任に関する教訓（G2）

教訓 G2　発注者は要件定義に責任を持ってシステム構築にかかわるべし

問題

A 社のシステムが本稼働後に、注文した処理の取り消しが不可など基本的な仕様に大きな漏れが見つかった。さらに、多数の要件の抜けや漏れが見つかった。

原因

上記の問題の直接の原因は要件の定義漏れである。

また、根本原因は、以下である。

A 社はシステム開発及び運用を IT ベンダに外部委託している。開発案件の増加にともなって委託先に任せる業務が徐々に拡大し、要件定義や受入れテストなど、発注者としての役割を果たし切れていなかった。

1. システム開発の発注形態や役割分担の問題
   システム要件定義を含めた IT ベンダへの請負契約で、要件定義を IT ベンダ任せとしていた。
2. 開発プロセスの問題
   要件定義書をもとにした、上流工程重視のシステム開発のプロセスになっていなかった。

対策

IT 組織改革を実施し、発注者として要件定義に責任を持ってシステム構築にかかわるようにした。主なポイントは以下である。

1）要件定義書の中身と受入れテストについての責任は発注者とする。（図 2.2－1）
（作成は IT ベンダに手伝ってもらうが、発注者の責任とし、契約で明確にする）
① 要件定義と設計以降で契約を分け、要件定義の契約は発注者が最終的な責任を持つ。
② 要件や設計内容に変更が発生した場合は必ず仕様変更書を作り、新たな契約を結ぶ。

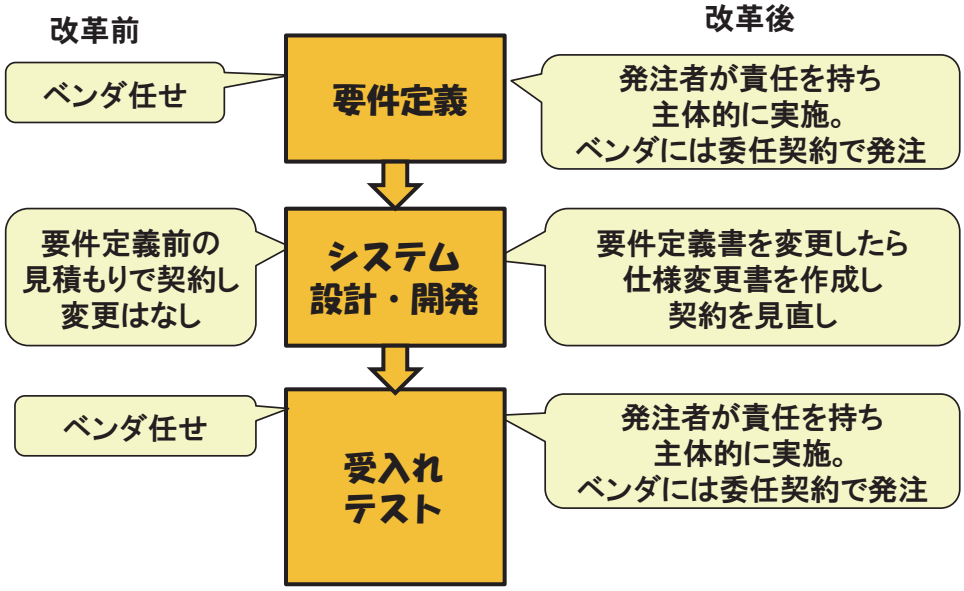


図 2.2－1　発注者の責任の明確化 [6]

2）開発プロセス標準を見直し、上流の要件定義を押さえる（上流工程完璧主義）。（図 2.2－2）
① 要件定義書の記述レベルの詳細化を図り、記載内容の網羅性チェックを徹底する。
② 受入れテストのテストケースを上流工程で作成し、要件の充足性 / 要件品質を確保する。
③ 発注者自ら要件定義が設計書に反映されたことを確認し、要件定義書と設計書のズレをなくす。

[6] 共通フレーム 2013（IPA/SEC 発行）（文献 2.2－1）を参考に作成

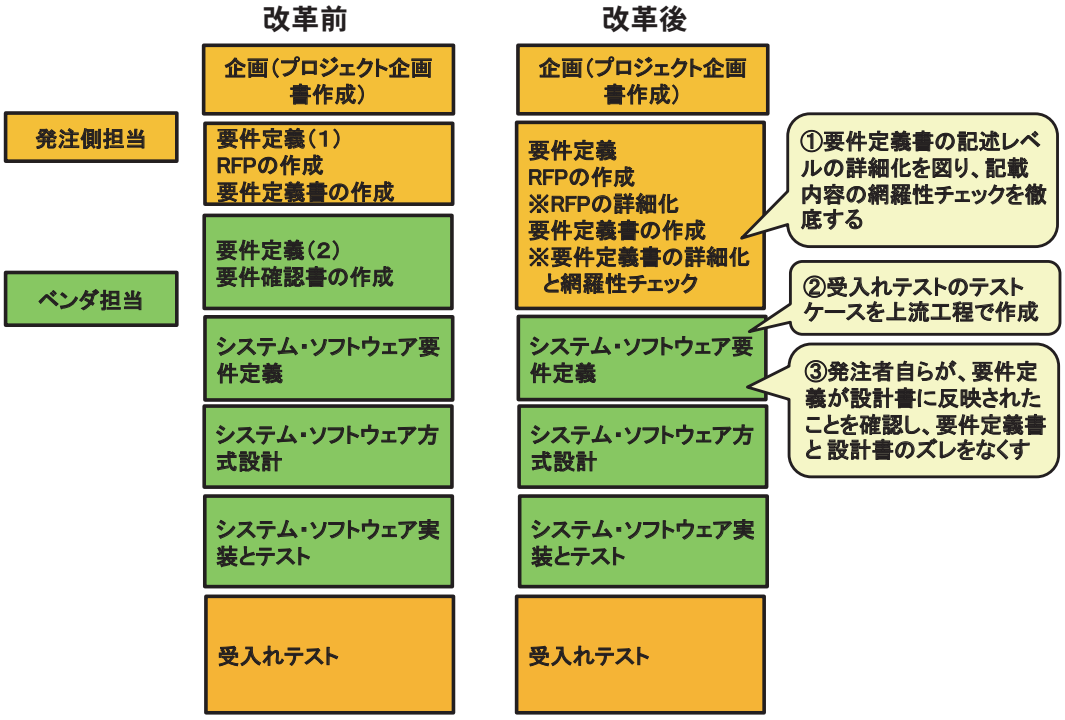


図 2.2－2　開発標準プロセスの比較（改革前・後）[7]

効果

上記の改革の実施により、稼働後 6 カ月の時点のバグ密度は、0.0074 件 /Kstep であり、金融業の稼働後 6 カ月バグ密度（0.030 件 /Kstep）[8] と比較しても、1/40 という高い品質であった。稼働後 4 年間も安定した状態である。

また、請負契約においてシステム開発を丸投げにせず、発注者であるユーザ企業のシステムへの関わりを増やすことで、以前は当たり前だった開発プロセスが変わった。プロジェクトの進捗などの透明性が要求されてくるため、受注者のベンダ企業の視点も変わり、組織同士の継続的な信頼関係の向上につながっている。

教訓

発注者は要件定義に責任を持ってシステム構築にかかわるべし

[7] 共通フレーム 2013（IPA/SEC 発行）（文献 2.2－1）を参考に作成
[8] ソフトウェア開発データ白書 2012－2013（IPA/SEC 発行）（文献 2.2－2）

2.3 上流工程での運用部門の関与に関する教訓（G3）

教訓 G3　運用部門は上流工程（企画・要件定義）から開発部門と連携して進めるべし

問題

最近、システム運用の現場で以下のような問題が多発している。

① A 社では、新システムの運用を開始してからオペレータの操作ミスが多発した。

② B 社では、販売店向け発注システムを Web システムに移行したところ、システム移行直後にシステム間の電文データに不具合が発生し、発注処理が行われなくなる事象が発生した。入力データのミスに起因し、電文データが誤って編集されたためである。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 週間程度かかった。

原因

直接の原因は以下である。

① A 社では企画や要件定義段階において、オペレータ操作に関する運用の要件検討が十分されていなかった。運用テストの段階から初めて運用者が参加してテスト及び引き継ぎを行ったがオペレータ操作関連のバグが多発して収束しないまま本稼働を開始した。要件定義段階でのオペレーション要件の検討漏れが原因だった。

② B 社ではシステムへのデータ入力ミスを抑止する工夫や仕組みが考慮されていなかった。
また、本システムの接続先との間で、有事に関する取り決めや対応範囲などが整理できておらず、コンティンジェンシプランが共有できていなかった。

上記①、②の根本原因は、運用要件の検討漏れあるいは軽視である。これは、運用者が要件定義作業へ参加していないことや、参加していても運用要件が取り込まれなかったことに起因している。

対策

上記の対策として、企画・要件定義作業において、運用者の視点からシステム要件を確認するようにする。運用者が確認する項目の例を表 2.3－1 に示す。なお、これは一例であり、これを参考にして、対象となるシステムに合わせて役割分担表などに表現する。プロジェクト開始前に関係者でレビューして合意し、プロジェクトに適用する。

表 2.3－1 「運用者が企画・要件定義工程で確認する項目」（1/2）

No	工程	分類	項目	全体で確認する項目	運用部門の関与度合い ◎：責任 ○：担当 △：支援	運用部門が確認する項目
1	企画	起案	■経営戦略を見据えたシステム構築及びシステム構築の目的・目標の明確化	①経営戦略の具現化 ②情報（システム）戦略の具現化 ③システム化の目的・方針 ④納期（ｽｹｼﾞｭｰﾙ） ⑤システム利用期間、ライフサイクル概要計画	△	①要件の把握 ②納期（スケジュール）の確認 ③システムのライフサイクル（更新間隔）確認 ④経営戦略の理解 ⑤情報（システム）戦略の理解
2		現状分析	■システム（業務）の現状分析、問題点・課題の抽出と分析	①運用状況 ②問題点・課題 ③最新のシステム動向、技術動向の分析	△	①構築ノウハウの提供 ②現状課題の提供 ③最新技術動向・トレンド等の情報分析提供 ④運用状況の報告
2-2		企画立案	■新システムの企画立案	①全社目標の体系化と施策の定義 ②情報システム要件のまとめ ③システム化の企画立案	○	①運用改善から見た新システム要件の提案
3		投資対効果	■システム構築における投資対効果の明確化	①投資対効果 ②コスト計画の立案	○	①運用要件者側に必要なコストの見積作成
4		承認	■システム構築の承認	①システム構築の承認を得る		
5	要件定義	経営、業務、システム要件の確認	■システム化要件の確認 ■サービス内容・レベルの確認 ■体制の確認 ■計画・スケジュールの確認 ■コストの確認 ■移行要件の確認	①システム化の背景、意図確認 ②現行業務調査と現行システム調査より現状の業務フロー概要、システムフロー概要把握 ③システム化の範囲確認 ④新業務フロー、新システムフロー確認 ⑤システム化要件・機能確認 ⑥問題点・課題確認 ⑦サービス内容、サービスレベル確認 ⑧開発計画・スケジュール確認 ⑨開発（プロジェクト）管理方法確認 ⑩開発 / 運用 / 保守体制と規模確認 ⑪概算見積もりを算出し、拘束期間、予算、コスト確認 ⑫企画段階で確立した開発者・運用者の双方の要件を確認し運用要件として確定 ⑬移行時の制限事項などを確認	○	① RFP の理解 ②システム化の狙い・概要・範囲の理解 ③マスタスケジュールの認識 ④サービスレベルの認識 ⑤コストの確認 ⑥ライセンスや守秘義務などの規約や条件の認識
6		非機能要件システム基盤	■システム構成の確認 / 提案 ■サーバのハード構成・環境要件の確認 / 提案 ■性能・拡張性要件の確認 / 提案	①機能要件を把握し、システム構成が運用に耐えられるか判断 ②システムを構成する機器の性能・容量は処理ピークに耐え拡張性も備えている事 ③システム予算と照らし合わせた最適な構成	○	①システムのハード構成の確認 ・ピーク性能をカバーしているか ・拡張性はどうか ・信頼性はどうか ・サービスレベルを満たしているか ・保守性はどうか ・標準化に準拠しているか ・ルールに準拠しているか ・コストは範囲内か ②システムのハード構成の提案
7			■キャパシティ管理要件の確認	①取得する項目 ②各リソースのしきい値 ③ピーク日のデータ処理量 ④トランザクション応答時間のしきい値 ⑤処理データの増加率を定期的に取得 ⑥バッチジョブの所要時間のしきい値 ⑦バッチジョブの終了時刻の許容範囲、異常を検出できる仕組 ⑧1ジョブで使用する最大テープユニット台数 ⑨管理手順の整備と資源枯渇防止対策	○	①必要データ量の確認 ②必要トラフィック量の確認 ③必要レスポンス値の確認 ④必要キャパシティの把握（ピーク、平常） ⑤現状のキャパシティ状況の把握 ⑥しきい値の確認 ⑦将来値の予測 ⑧制限項目の把握 ⑨ルール・標準化に準拠してるか ⑩システム構成がキャパシティ的に適正かどうかの検討と判断 ⑪システム構成の提案
8			■システム構成の確認 ■サーバのソフトウエア要件（運用系）の確認	①ＯＳはルールや標準化に準拠している運用可能なＯＳである事 ②待機系システムがある場合は待機系システムにも各種ソフトウエアを導入する ③各種運用ツールは既存標準ツールに合わせる事	○	①システムのソフト構成の確認 ・OS は運用可能か、 ・運用系ソフト、既存標準ソフト ②システムのソフト構成の提案
9			■システム構成の確認 ■パソコン・周辺機の要件の確認 "	①パソコンの構成 ②プリンタ ③その他周辺機 ④稼働可能ＯＳ、ソフトウエア、開発資産、サービスパック、累積パッチのレベル ⑤ＩＤ、パスワードの設定、ログイン権限設定 ⑥資産配布	○	①クライアントのソフト構成の確認 ・OS は運用可能か、 ・運用系ソフト、既存標準ソフト ②クライアントのソフト構成の提案 ・ソフト構成を確認し NG の場合は提案する
10			■信頼性要件の確認 （冗長化・二重化など）	①最重要システム業務再開 ②サーバやネットワーク機器の冗長化／分散化構成 ③コールドスタンバイ、ホットスタンバイ構成 ④システムの拠点切替え時の他システムの影響 ⑤拠点切替え時、連携システムなどでの切替え操作 ⑥単独部品故障では機能停止に陥らない冗長化構成、ホットスペア実装、部品活性化交換 ⑦ 24 時間 365 日のサービス提供の場合は冗長化構成とし、保守作業等は、待機系に切り替えて作業を行う。 ⑧クラスタ切替え手順自動化、手順書作成	○	①信頼性対応方針の確認 ②信頼性対応システムの確認 ③信頼性対応範囲の確認 ④信頼性対応方法の確認 ⑤対応時間の確認 ⑥信頼性構成の確認 ⑦ルール・標準化に準拠してるか ⑧信頼性構成の提案

表 2.3－1 「運用者が企画・要件定義工程で確認する項目」(2/2)

No	工程	分類	項目	全体で確認する項目	運用部門の関与度合い ◎：責任 ○：担当 △：支援	運用部門が確認する項目
11	要件定義	非機能要件システム基盤	■セキュリティ要件の確認	①セキュリティ方針及び規定に対し、運用部門の観点で問題のないことの確認 ②アクセス権限の設定 ③ＩＤ管理 ④個人情報ファイルの取扱い ⑤センターポリシーの通知 / セキュリティパッチ適用 ⑥ウィルス対策	○	①セキュリティ対応方針の確認 ②セキュリティ対応システムの確認 ③セキュリティ対応範囲の確認 ④セキュリティ対応方法の確認 ⑥セキュリティ構成の確認 ⑦ルール・標準化に準拠してるか ⑧セキュリティ構成の提案
12			■システム構成の確認 ■ネットワークの要件の確認	①システム全体のネットワーク構成 ②既存増強（伝送容量アップ）、新規増設（拠点追加）、帯域制御 ③ネット機器、メーカ ④設置拠点の大小によるネット構成と機器 ⑤ＩＰアドレス、コンピュータ名 ⑥冗長化のルール ⑦監視方法（死活監視など） ⑧ファームウエアの更新 ⑨事故停電や計画停電の対応 ⑩帯域制御 ⑪Ｆ／Ｗ	○	①ネットワーク構成の確認 ・トラフィック量は構成に耐えうるのか ・帯域制御の確認 ・冗長化の確認 ・監視方法の確認 ・保守方法の確認 ・停電対策の確認 ・ルール・標準化に準拠してるか ②ネットワーク構成の提案
13			■施設要件	①機器の設置場所 ②設置場所の耐震、電源、空調、セキュリティの仕様 ③設置機器の電源仕様 ④設置機器の物理仕様 ⑤設置機器のネットワーク使用 ⑥機器の設置期間	○	①設置場所は適正か確認 ②耐震仕様の確認 ③電源仕様の確認 ④空調仕様の確認 ⑤入室ルール規則の確認 ⑥施設要件の提案
14		運用・保守	■オペレーション要件	①稼働日及び、業務運用時間 ②ジョブ実行マニュアルの整備 ③コンソール要求とマニュアルの整備 ④入力運用 ⑤出力運用 ⑥バックアップの復旧要件 ⑦変更管理 / リリース / 資産管理運用 ⑧システムの監視範囲 ⑨障害検知及び通知のルール ⑩運行保全用ドキュメントと引き継ぎ期間 ⑪ログの表記、保存期間、削除タイミング	○	システム運用方針の確認 ①新規システムの業務運行時間の確認 ②類似するシステムの保守内容の情報提供 ③保守契約の妥当性の確認 ④類似するシステムの入出力・バックアップ ⑤新規システムの入出力・バックアップ ⑥現行システムと共用できる設備の提案 ⑦運用時の全オペレーションの確認 ⑧運用作業効率化のための提案 ⑨運行保全用ドキュメントの確認 ⑩ログの表記、保存期間、削除タイミング ⑪運用者要員計画の立案 ⑫変更管理 / リリース / 資産管理運用 ⑬標準化に準拠しているか ⑭ルールに準拠しているか ⑮運用要件の提案
15			■異常時、障害時の要件	①業務停止から復旧までの許容時間 ②異常発生時は既存の通報システムと連動 ③ホットスタンバイ、コールドスタンバイなどシステム構成 ④異常発生時の影響範囲、有事の際の関連部署と連携マニュアル、連絡網、通報 ⑤システム(サーバ、ホスト、ネット機器)の自動監視 ⑥リモートログイン遠隔地から障害対応 ⑦縮退運用の優先順位とその影響内容 ⑧障害ＤＢの整備 ⑨ジョブネットワーク構成	○	①類似システムの情報提供 ②障害対応方針の確認 ③障害検知方針の確認 ④障害時連絡体制の整備の確認 ⑤障害証跡の保持方針の確認 ⑥安定的な業務運営のための提案 ⑦標準化に準拠しているか ⑧ルールに準拠しているか ⑨障害対応要件の提案
16			■保守要件	①定期保守 ②保守の範囲（回数）/ 稼働報告の範囲 ③サービス時間が 24 時間 365 日の場合の保守時間の確保されている事 ④システムの稼働状況を報告する定例会 ⑤拠点の計画停電対応等の計画停止用の臨時システム	◎	①保守方針の確認 ②保守サイクルの確認 ③保守要件の提案
17		移行	■移行要件	①本番移行方法・切替日が明確になっている事 ・一括切替本稼働 ・順次切替本稼働 ・平行稼働	○	①影響範囲の明確化のための現行システムの情報提供 ②周知方法、周知先の妥当性の確認 ③移行・切替日の確認 ④移行方法の確認 ⑤影響範囲の確認 ⑥移行要件の提案
18		監査	■内部統制、システム監査要件	①システム証跡を出力する	○	①類似システムのシステム証跡取得方法 ②各種認証運用の確認 ③必要資料の確認 ④パッケージの選定、各種認証資料 ⑤内部統制要件の提案
19		合意	■要件定義の合意	①要件定義の全項目を合意する	○	①教育 / 引継計画の確認と修正 ②教育 / 引継スケジュールの確認と修正 ③教育 / 引継資料の確認と修正 ④教育 / 引継環境の構築と確認

さらに、運用者が企画・要件定義に参加するときの心構えを以下にまとめた。

1：運用者は経営戦略、情報システム戦略を十分理解すること（企画）

システム要件を確定し開発・運用を進めるにあたり「オーナーが真にシステムに求めているものは何か?」「なぜそのようなシステムが必要なのか?」を理解することが重要である。その根底・背景には必ず企業の経営戦略や情報戦略が存在する。

2：運用者はプロジェクト企画書の作成段階から企画・要件定義者へのサポートを行うこと（企画）

プロジェクト企画書は企画・要件定義者のみで作成するものと決めつけてはならない。

運用者はプロジェクト企画書を企画・要件定義者や経営から直接説明を受け、適宜質問を行い、意見具申し、その内容を十分理解しながら適切なサポートを行わなければならない。

3：運用者は運用方針／運用要件を定義すること（要件定義）

運用者は、運用方針／運用要件を「自分の言葉で」提案しなければならない。要件定義に運用要件が盛り込まれていないケースは多く見受けられる。開発者は、業務の要件定義には慣れているが、運用にかかわる要件定義は不得意である。企画・要件定義者も要求が曖昧になりやすい。

また、運用者が社外中心となっている場合は社外の運用者の意見を求めることも必要である。

効果

運用者が要件定義に参加することで、運用要件の漏れに起因する障害の多くが解決できる。

教訓

システムは運用されてこそ、その価値を生み出すものであり、システムの企画、要件定義段階において、無駄のない運用ができるような運用設計が大変重要である。このためには、企画・要件定義作業へ運用者が参加して確認する項目や作業を「見える化」することが重要である。

また、運用者は、要件を十分に説明できていない場合も多いので、開発するものは自分のシステムであることを自覚して、積極的に要件をだす必要がある。

2.4 障害発生時連絡の情報共有に関する教訓（G4）

教訓 G4　運用者は少しでも気になった事象は放置せず共有し、とことん追求すべし

問題

A社のシステムが数時間停止し、また対応が遅れたため、公表も遅れた。

同社のオンラインサービスを提供するシステムは現用・待機ノードによる二重化冗長構成を採っており、現用ノードが故障すると、自動的に待機ノードに切り替わる仕組みとなっていた。また、同システムの運用は子会社に委託され、運用子会社の担当者はシステムの異常を検出すると診断メッセージを出力して保守ベンダに解析を依頼し、その解析結果を本社の運用部門に伝え、本社運用部門では重大な状況の場合にのみ経営層（CIO）に報告する手順となっており、このような運用手順がマニュアルに記載されていた。

ある日の夜間バッチ処理中に1台の現用ノードでメモリコントローラの障害が発生し、監視端末にエラーを示すメッセージが表示された。

運用子会社の担当者は「障害診断ツール」を使い、診断レポートを出力し、保守ベンダのSEに対し、電話と電子メールで診断レポートの内容を報告した（図2.4－1①）。

保守ベンダのSEは診断レポートの内容を見て、待機ノードが正常に稼働していると判断し、運用子会社の責任者に切替え処理が成功しているとの見解を伝えた（図2.4－1②）。

運用部門の統括責任者は「当日の業務への影響はない」と判断し、処理が切り替わっていると誤認したまま障害対応を完了し、経営陣への報告を行わなかった（図2.4－1③）。

運用部門が経営陣（CIO）へ報告すべきだと判断したのは、翌日未明の業務で一部の情報が配信できないことが判明してからだった。CIOに連絡が取れたのは翌日朝で、午前中のオンラインサービスは停止となった（図2.4－1④）。

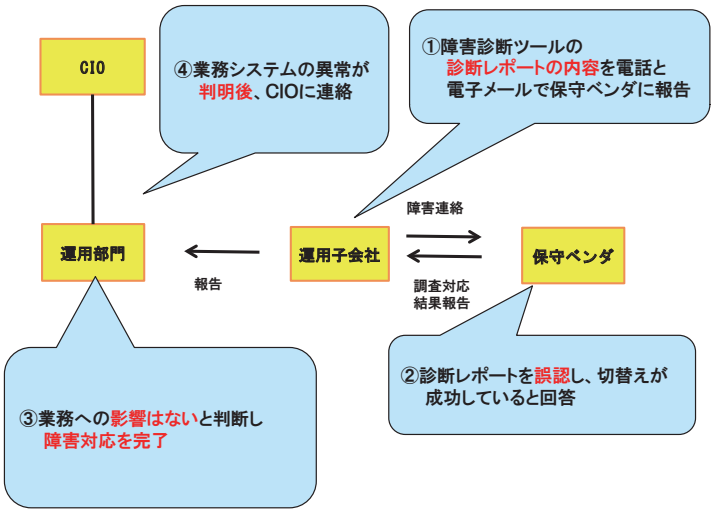


図2.4－1　障害の経緯

原因

今回の問題の直接の原因はメモリコントローラの障害が発生したことである。

これに続いて、以下のヒューマンエラーによるミスが発生した。

① 保守ベンダの SE が診断レポートを誤認した。

実際には、待機ノードが処理を引き取って継続するには、現用ノードが処理不能だと表明し、“バトン”が渡される必要があった。運用子会社の担当者は診断レポートの内容から「現用ノードは死にかけているが、待機ノードにバトンが渡っていない」らしき状況に気づいていたが、保守ベンダの SE にそのことを連絡せず、保守ベンダの SE は現用ノードからバトンが渡っていると誤認し運用子会社の責任者に報告した。

② 運用部門が主体的にシステムの状態を確認せず、運用子会社からの報告で問題なしと判断した。

③ 運用部門が経営陣に適切な報告を怠り、業務の対応の遅れにつながった。

さらに調べると、ミスが発生した根本原因は以下であることがわかった。

④ 運用子会社の担当者と保守ベンダの SE は、マニュアル通りに作業したが、それに加えて運用子会社の担当者は、現場での異常を察知したときには、その情報を保守ベンダの SE や運用子会社の責任者に伝達した上で協議すべきであった。

運用子会社には、オペレーションしている人も疑問に思ったら相談するという「顧客視点」でシステムを捉える考え方が欠けていた。

対策

A 社は障害発生時の緊急態勢・運用手順を以下のように整備し再発防止策とした。

最も重視した再発防止策は、

①「運用担当者が現場での異常を察知したときには、状況判断できる運用部門の社員にその情報を連絡して協議する態勢を作る」ことをオペレーションマニュアルに明記する。

ということである。

あわせて以下のような再発防止策も実施した。

② 障害対応を体制面で改善して強化する。

- 事象として障害と断定できない場合でも障害の可能性がある場合は、早期に上位役職者へ報告するルールを追加作成する。
- 状況判断できる運用部門の社員（プロパ）がセンターへ 24 時間常駐する。

③ 確認手順及び確認項目を明確に定義する。

- 速やかな復旧に向けた取り組みに変える。
- 装置の停止有無（コンソールメッセージの伝達方法の改善）、業務影響、障害リスクを明確に定義する。
- 障害対応の確認項目を明確に定義する。
- 障害対応時アクションリストに項目を追加する。
- エスカレーション管理台帳を整備する。

- 保守ベンダの連絡ルールを改善する。

④ ハードウェアのハングアップなどで二重化の切替えが不確定な場合には、疑わしき要素を切り離す仕組みを順次取り込む。（フェールソフト）

⑤ 必要な教育及び訓練を実施する。

効果

今回の対応後にも障害が発生しているが、報告ルールの整備などの再発防止策により、利用者に影響を及ぼす重大障害につながらずに対応できている。

教訓

運用現場ではあらかじめ想定できないようなことが起こるので、運用担当者が現場での異常を察知したときには、状況判断できる運用部門の社員にその情報を連絡して協議するような仕組みを考えておくべきである。

2.5 共同利用システムの業務処理量予測に関する教訓（G5）

教訓 G5 サービスの拡大期には業務の処理量について特に入念な予測を実施すべし

問題

X システムは業界共同システムで、稼働後 1 年が経過し、業界普及率がシステム稼働当初から 2 倍に増加する時期に差し掛かっていた。業務の特性上、X システムは年度末に処理が集中する傾向があるが、キャパシティ予測が不十分であったため、想定を大幅に上回る負荷がサーバにかかったことによりミドルウェアの潜在バグが発現し、システムダウンとなった。潜在バグの修正はすぐには対応できず、またキャパシティの増強も時間がかかるため、年度末、数日間にわたってシステムがダウン、あるいはレスポンスが極度に悪化する事態となり、運用上、大幅な利用制限を行わざるを得ず、業務が著しく停滞した。重要な業務が停止したことにより、本システム障害による社会的影響は大きいものとなった。

原因

本システムは、端末からのオンラインリクエストでバッチ処理を起動し、処理結果をオンラインで端末に返す処理形態である。その概要と今回の障害の流れについて、図 2.5－1 に示す。

トランザクション量の予測が難しいケースに対しては、トランザクションの流量制御を行い、キャパシティ不足によるダウンを回避する等の工夫が必要である。しかし、このケースでは、アプリケーションサーバで流量制御を行っていたものの、負荷が重いバッチ型オンライン業務に含まれる 1 トランザクション当たりのデータ数が、予測よりはるかに多かった。本障害の直接原因は、DB サーバ内で SQL 処理のタイムアウトが多発し、それがミドルウェアのバグによるメモリリークを引き起こし、アプリケーションサーバがダウンしたことにある。これにより、処理の負荷は比較的軽いが、重要な業務である通常オンライン業務も停止することとなった。なお、メモリリークを引き起こしたミドルウェアの潜在バグについては、後日修正された。

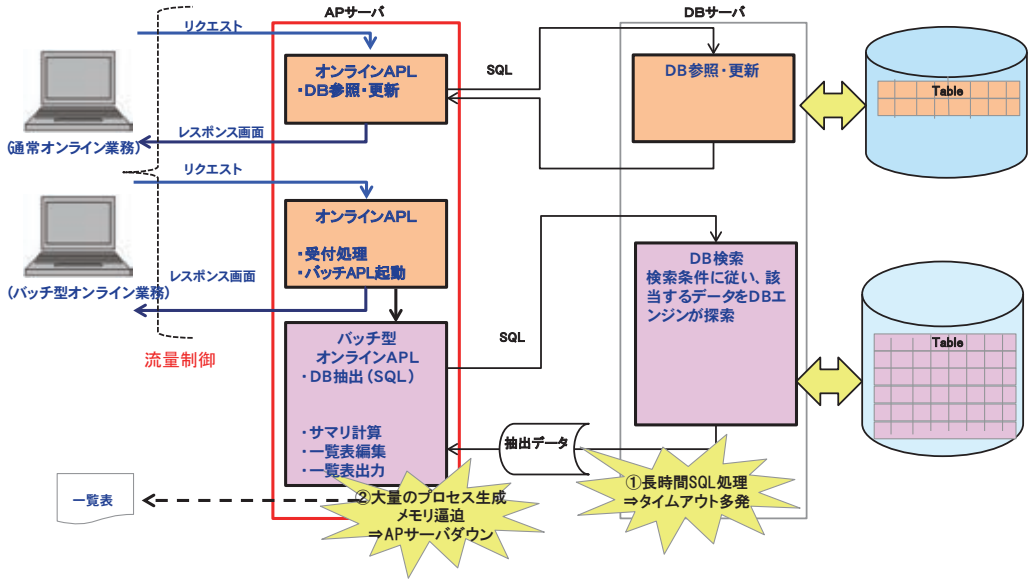


図 2.5－1　X システムの概要と障害の流れ

原因を突き詰めていくと、障害の根本原因は、前もって行われた年度末のピーク時における処理件数予測の誤りであるといえる。当該時期における業務別の予測と実績には、表 2.5－1 のような乖離があった。

表 2.5－1　業務別の予測と実績

○通常オンライン業務

	処理件数	ピーク時トランザクション数
予測	100 万件	20.0TRX/ 秒 →この予測に従い、25.0TRX/ 秒までの負荷テストを実施した。
実績	120 万件	35.0TRX/ 秒（秒間の集中度が予測より高かった）

○バッチ型オンライン業務

	1 つのバッチに含まれるデータ数
予測	「通常は 20 ～ 30 件で、まれに 100 件程度のものもあるだろう」との推測のみ。 →この予測に従い、10 分間のテスト時間の中で、100 件のデータが含まれるテストを 1 回実施。
実績	平均で 100 件以上、最大 600 件のデータが含まれていた。

このような不十分な予測となった背景には、これが業界共同システムであり、利用各社が運営をベンダ任せにして、自社のシステムの場合に行うような責任をもった予測を行っていなかったという背景がある。通常オンライン業務についても、バッチ型オンライン業務についても、過去の利用実績を、

業務の実態をよく知る者が深く分析した上で予測すれば、これほど乖離した予測となることはなかった。

対策

このような事態を招いたのは、X システムをベンダに委託している利用各社が、運営をベンダ任せにして、システムの運営に十分コミットメントした「かまえ=態勢」を構築していなかったためである。（X システムは開発から運用までをベンダに委託している）

その点を反省し、利用各社は「X システム運営協議会」（以下、運営協議会と呼ぶ）を作り、キャパシティ・プランニングを始め、重要なシステム変更やリリース等、委託する側が責任を持って決めるべき点を明確にし、運営協議会で決める態勢とした。また、利用各社とベンダとの個別契約にも、運営協議会で決めるべき項目を明記した。

効果

利用各社の責任態勢を明確にし、利用各社とベンダがそれぞれの責任にもとづいて健全な緊張感を持って対峙する態勢としたことにより、以降、業界への普及率が 9 割を超えた現在に至るまで、システム障害は発生していない。

また、このシステム障害を契機に、利用各社がバックアップシステムの必要性を再認識することとなった。そこで、運営協議会で議論の上、バックアップシステムを構築したことにより、更なる信頼性の向上を実現した。さらに、「通常運用しないバックアップシステムはいざというときに稼働しない」という経験則を踏まえ、一年に一回、訓練も兼ねて、バックアップサイトで本システムを運用している。

教訓

一部のトランザクション量の変動が激しい大規模なシステムを除けば、コスト等の観点から、コンピュータキャパシティの余裕を十分確保することは難しい。しかしながら、サービスの拡大期においては、利用率が急増する場合がある。

このような場合には、業務の処理量についての入念な予測が重要である。業務をよく知る者を中心とした態勢により、責任をもって予測することが必要である。

2.6 作業ミス、ルール逸脱の問題に関する教訓（G6）

教訓 G6　作業ミスとルール逸脱は、個人の問題でなく、組織の問題！

問題

A社システム部門は、多数のグループ会社及び関連会社が利用するグループウェア・サービスを運用している。同サービスは、統合アカウント管理ツールとグループウェアとから構成されている。統合アカウント管理ツールは、運用作業者が通常、ユーザの登録・削除等の運用操作を行うためのシステムである。グループウェアは、利用者データベースを持ち、利用者データに対する操作が可能となっている。通常運用作業者は、直接グループウェアのアカウント情報を直接操作してはならないルールである。

そのグループウェア・サービスの障害により、多数の利用者のデータ（送受信メール、スケジュール、アドレス帳等）が消失してしまい、復旧するのに2日間を費やした。

経緯は、以下の通りである。

- 運用作業者が、統合アカウント管理ツールを使って、新規ユーザ（50名分）のユーザ登録作業を実施したところ、ユーザの設定が誤っていることに気づいた（図2.6－1①）。
- 再登録するために統合アカウント管理ツールを使い先に登録したユーザを削除しようとしたが、削除できなかったため（図2.6－1②）、直接グループウェアサーバ上で登録したユーザ（50名分）を削除しようとした（図2.6－1③）。
- ところが、誤って“全ユーザ削除”を実行してしまった（図2.6－1④）。
- すぐに気づき、実行処理を停止しようとしたが、できなかったため、グループウェアサーバを強制再起動した（図2.6－1⑤）。

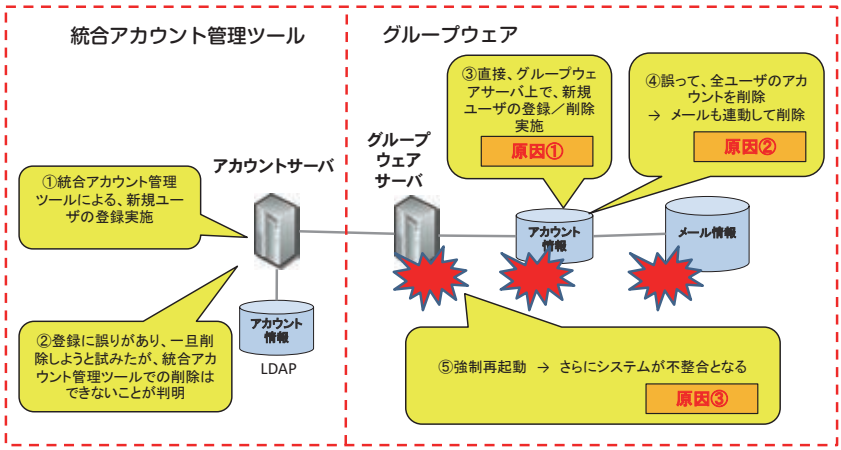


図2.6－1　障害状況

原因

直接原因は、運用作業者が運用ルールにない作業を行ったことによる。

作業者は、自分の判断でマニュアルに記載がない作業を不用意に行った結果、統合アカウント管理ツールでアカウントの更新を行わなければならないのを、直接グループウェアサーバで作業をしたり（図 2.6－1－原因①）、全ユーザを削除するボタンを押下して障害を発生させてしまったり（図 2.6－1－原因②）、グループウェアサーバを強制再起動させたり（図 2.6－1－原因③）、障害を拡大させてしまった。

根本原因は、システム部門作業者が作業に入れる状況にないにも関わらず、作業を行わせた組織（システム部門）のマネジメントが働かなかったことである。

当時、グループウェア・サービスの移行スケジュールが立て込んでおり、今回のアカウント登録作業もすぐに行わなければならない状況であった。そのため、組織（システム部門）の作業依頼者から作業の緊急性が運用チームに伝えられていた。運用チームでは、教育、作業手順マニュアルの作成が追いつかず、チーム内のスキルの共有化ができていなかった（作業の属人化）。そのため、作業は、不慣れな作業者一人で行うことになった。作業中、チーム内ではお互い自分の作業に追われていたため、作業者は、問題が出ても熟練者に聞くことができず、また、時間の制約による焦りがあった。そのため、早く作業を終わらせないと業務に影響が生じ、自分がその原因を作ってはいけないというプレッシャを感じていた作業者は、不具合が生じても、独断で作業を進めてしまった（図 2.6－2）。

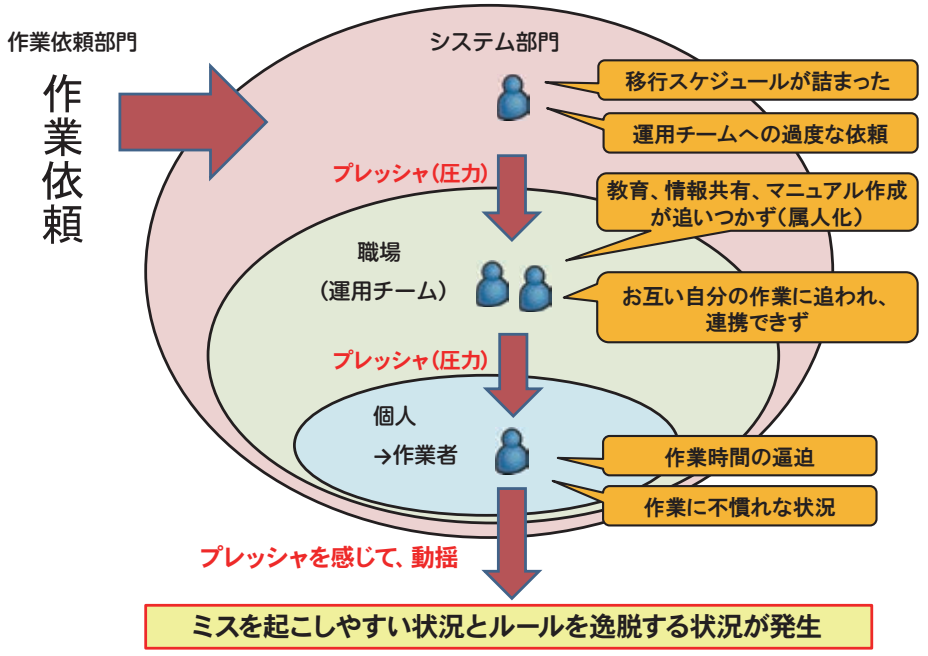


図 2.6－2　マネジメントの問題から引き起こされた障害

対策

再起動後、バックアップを使ってデータ復旧を行った。バックアップで復旧できないユーザについては、新規にユーザカウントを再作成し、受信メールのみ 3 カ月前のメールアーカイブから復旧作業を行なった。

システム部門は、マネジメントの観点から、作業ミスを減らし、ルール逸脱を無くする対策を行った。作業ミス、ルール逸脱は、個人の問題とするのではなく、組織の問題と捉え、総合的な対策を立てることとした。以下、今回の事例を基に検討した対策案を述べる。このような行動を PDCA サイクルで廻していく。

① 作業を受ける場合は、作業を実施した場合のリスクを分析して判断基準を作成し、その基準をクリアした上で作業実施を決定する。

システム部門は、作業を受ける前に、自部門の作業者のスキルや稼働状況、及び作業の目的、重要度、影響度を基に、作業を引き受けた場合のリスク分析を行い、作業実施の判断基準を設定する。その場合、過去のヒヤリハット、障害記録なども参考にする。

例えば今回の事例では、以下のような判断基準を設定し確認していれば、障害を未然に防げた。

- 作業マニュアルは、万全か。
- 作業内容は、チーム内で共有化されているか（作業の属人化に陥っていないか）。
- 作業者は、作業中に疑問が生じたときの対応が取れるようになっているか。

当然、このような判断基準がクリアできない場合は、対策を立てる。また、作業依頼部門に作業実施の再スケジュールを依頼することも必要になるので、作業依頼部門とも十分話し合っておく必要がある。

② 作業実施時にルールを逸脱しない作業規定を作る。

システム部門は、作業実施時にルールを逸脱しないような方策を立て、それをチーム内作業者と常に改善していくプロセスを実施する。今回の事例では、以下のような対策を実施していれば、障害の拡大を防ぐことができた。

- 作業のオペミスを防ぐため、作業は 2 名体制（実施者、確認者）で行い、手順書に書かれた作業以外は、作業者の判断では、絶対に行わない。また、作業記録をとり、作業実施ログを保存する。
- 手順書と違う状態が発生した時点で作業を止め、上位者へ報告し、組織として判断、対応する。インシデント管理（障害記録の作成）を実施する。
- 作業ミスを減らすためのシステムの改善点があれば実施する。今回の場合は、重要コマンドは、簡単に投入できない様に権限を設定する改善を行う。

効果

組織としての対策を総合的に行うことで、作業ミスを減らし、ルール逸脱を無くすことができ、システム障害を減らすことができる。

教訓

作業ミスとルール逸脱は、個人の問題にするのではなく、組織として取組むことが重要である。

組織、職場（チーム）、個人が受けるプレッシャによって誘発される作業ミス、ルール逸脱は、組織内、職場（チーム）内の問題として取組むことによって減らすことができる。それには、組織内、職場（チーム）内の体制作り、ルール策定、システムによる防御対策を常にPDCAサイクルで回しながら改善を行う。

また、日ごろからの組織内、職場（チーム）内のコミュニケーションが重要な事は言うまでもない。

2.7 クラウドサービス利用時の障害対応体制に関する教訓（G7）

教訓 G7　クラウド事業者と利用者が連携した統制がとれたトラブル対応体制を整備すべし

問題

A 社はオンラインによる情報登録及び情報照会の基幹業務システムを当初はオンプレミスで運用していたが、運用コストの削減を目的に複数企業間の共同利用を進める方針となり、B 社が提供するクラウドサービスに移行した。同時期に共同利用に移行するのは他に D 社があり、類似のビジネスを行っていた。B 社が提供するシステムは、業務システム用のサーバと負荷分散装置に分かれている。業務システムのサーバだけでなく、負荷分散装置も仮想化されており、その一つの論理区画をA社は利用していた。（図 2.7－1）

ある日、オンライン開始時からこのシステムに障害が発生してまる 1 日業務が停止した。基幹オンラインシステムが端末から起動できず、すべての窓口でデータベースの更新をともなう処理の受付けができなかった（図 2.7－1 ①）。

なお、A 社があらかじめ用意していたクラウド外の「障害時バックアップシステム」に切り替わり、データ照会処理はできたので、データの更新をともなわないサービスのみを実施した（図 2.7－1 ②）。

B 社は障害個所の特定に時間を要し、また A 社は各方面への説明対応に追われたこともあり、障害個所が判明したのは 16:00 であった。既に業務終了時間が近づいていたためオンラインは終日停止、障害復旧作業はその後実施となった。

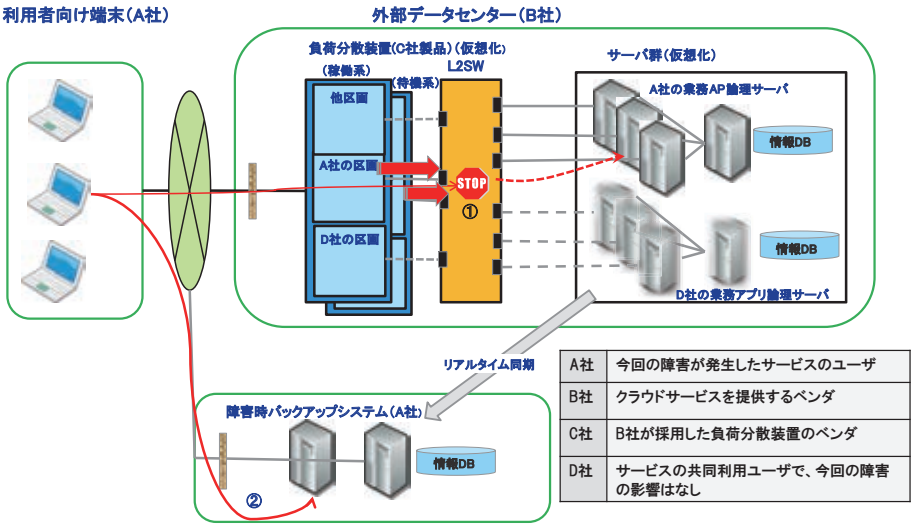


図 2.7－1　システム概要

原因

直接の原因は、C 社製負荷分散装置の sod プロセスのメモリ資源が時間とともに増加するという既知の不具合であった。

単なる負荷分散装置の障害にも関わらず、その解決と業務の再開に多大の時間を要し丸一日間オンラインサービスが停止することとなった原因は、以下のとおりである。

- 通信路の疎通状況を確認する ping が通ったことから、B 社はシステムの運用に問題ないと誤認した。
- 初めのうちはアプリケーションサーバや専用線の問題と誤認し調査を行っていた。B 社の自社製品ではない C 社製負荷分散装置の障害調査は後回しにした。
- 負荷分散装置は D 社と共用しており、再起動等の対処による D 社サービスへの影響が不明のため、A 社の了解のもと対処を業務終了後まで先送りすることとした。

根本原因は以下のとおりである。

1. 運用時のトラブル対応体制が決まっていなかった。
   - 障害調査を進め、一体となって協力して進めていく体制ができていなかった。B 社の SE は A 社に常駐していたが、トラブル管理体制が明確化されていないので、報告、連絡、相談がうまく回らなかった。
2. A 社は B 社と役割分担やサービスレベルが不明確な運用委託契約のままサービスを開始していた。
3. B 社に C 社製負荷分散装置の専門家が不在で、社外の製品のため障害情報の入手もしづらく、障害やパッチの情報をタイムリーに入手していなかった。

対策

再発防止策は以下のとおりである。

1. トラブル対応の体制の強化
   トラブル対応体制を明確化し障害発生時の報告、連絡、相談を行う。
   （考慮すべきこととして、ユーザは、対応をベンダ任せにせず積極的に働きかける。
   ベンダは、ユーザに対する状況報告を密に行う）
   トラブル発生時はユーザとベンダを TV 会議で結び、ユーザとベンダが密接に協力して対応するなどを検討する。
2. 適切な契約でサービスのレベルの定義を行い、責任分界点を明確にする。
3. ベンダは関係する各サードパーティ業者とトラブル対応体制を確立し、障害時の連携を適確に行う

効果

クラウドサービスにおいても役割や責任が明確となり、障害発生時のエスカレーションや対応を迅速に行うことができる。

教訓

ユーザはクラウド型システムの障害発生に備えて、クラウド事業者と連携した統制がとれたトラブル対応体制の整備が必要である。特にユーザはベンダに対して、役割分担や契約などのやるべきことをはっきりと要求し、厳しく緊張感を持って対応すべきである。これによりシステムの信頼性が向上するだけでなく、両者がお互いに成長することができる。

参考資料）
クラウド適用のガイドラインについて *
特定非営利活動法人 ASP・SaaS・IoT クラウド コンソーシアム（ASPIC）
ASPIC が取組んできたガイドライン・指針等

* http://www.aspicjapan.org/information/guideline/index.html

2.8 共同利用システムの利用者間情報共有に関する教訓（G8）

教訓 G8　共同利用システムでは、非常時対応を含めて利用者間の情報共有を図ること

問題

教訓 G7 と同じ事例である。本教訓で特筆すべきこととしては以下があげられる。

A 社と D 社は、あらかじめ協議を行い、運用コスト削減という目的が一致したことからシステムを共同利用することとした。

B 社は負荷分散装置のあるファームウェア処理に原因があることをほぼ特定し、再起動により回復することが見込まれた。しかし、この負荷分散装置は仮想化され、複数の利用者用にそれぞれ論理区画が設定されていた。そして、A 社の論理区画を再起動した場合の共同利用している他企業 D 社のオンライン業務に影響する可能性を明確に排除できなかった。よって、A 社の判断により、D 社のオンライン業務が終了する時間まで障害対処を見送ることとなった。また、これらの状況は D 社には全く伝えられていなかった。

原因

直接の原因は、C 社製負荷分散装置の sod プロセスのメモリ資源が時間とともに増加するという既知の不具合であった。

単なる負荷分散装置の障害にも関わらず、その解決と業務の再開に多大の時間を要し丸一日間オンラインサービスが停止することとなった原因は、以下のとおりである。

- 負荷分散装置は D 社と共用しており、再起動等の対処による D 社サービスへの影響が不明のため、A 社の了解のもと対処を業務終了後まで先送りすることとした。

根本原因は以下のとおりである。

B 社においては、共同利用サービスで使用する負荷分散装置等の共用機器に障害が発生した場合の復旧について、利用者への影響範囲の明確化を含めた手順が整備されていなかった。また、利用者との間でも、障害復旧に関する合意ができていなかった。したがって、D 社に連絡するとともに A 社の論理区画を再起動するという、迅速な復旧のための対処が出来なかった。

一方、A 社は、障害発生時には、システムを D 社と共同利用していることは認識していたが、トラブル発生時等の情報共有のための利用者間の連絡体制は確立されていなかった。

対策

対策は以下である。

再発防止策）

- 障害復旧時の関係者（サービスの共同利用者であるA社とD社、事業者であるB社）の役割分担や、システムにおけるコンポーネント（各種機器（ハード・ソフト）、データ）ごとの回復措置の利用者業務への影響範囲を明確化する（表2.8－1）。特に、障害発生時に停止／再起動する単位と、その停止により影響を受ける利用者の範囲を事前に整理・明確化する。
- B社は、システムを停止／再起動させる場合についての条件や手順、責任等に関する取決めをSLAで定義し、共同利用各社と合意する。
- 共同利用者間の情報共有の強化を行う。基本的にこれはサービス事業者の役割であるが、共同利用サービスでは、共同利用者間の日常の情報共有を行うとともに、非常時の緊急連絡体制等を定めておくことが望ましい。

表2.8－1　コンポーネント（今回の事例）

ハードウェア	負荷分散装置、ネットワーク機器、サーバ（アプリケーション, DB, Web）
ミドルウェア	仮想化OS, ゲストOS, OLTP（オンライン基盤ソフトウェア）など
アプリケーション	業務アプリケーション
データ	データベース, 一般ファイル

効果

障害発生時に、障害対応と再起動のための他社のオンライン業務の一時停止の協力等が得られるので、早急なシステムの復旧と業務継続が可能となる。

教訓

共同利用システムでは、利用者は通常はシステムの内容は理解していないので、障害が発生してもサービス事業者に頼るしかない。しかし、障害により困るのは業務サービスを利用するサービス利用者、エンドユーザなので、もしもの場合に備え、影響範囲等の制約条件を含む障害復旧手順を明確にするとともに、共同利用者間での情報共有を図ることが必要である。

2.9 非常時代替事務マニュアルに関する教訓（G9）

教訓 G9 システム利用不可時の手作業による代替業務マニュアルを作成し定期的な訓練を行うべし

問題

教訓 G7 と同じ事例である。本教訓で特筆すべきこととしては以下があげられる。

- あらかじめ用意された障害時バックアップシステムがサポートする参照系業務については、基幹業務システムと同じオンラインマニュアルが整備されており、それを利用して業務を遂行できた。
- 業務窓口は、登録系業務の処理の対応手順がわからなかったため　顧客の登録・変更申請に対応できなかった。結果として、窓口に来た顧客に帰って頂く対応となった。

原因

直接の原因は、C 社製負荷分散装置の sod プロセスのメモリ資源が時間とともに増加するという既知の不具合であった。

しかし、単なる負荷分散装置の障害にも関わらず、顧客の登録・変更申請に対応できず、結果として、窓口に来た顧客に帰っていただくことになった。

根本原因は以下のとおりである。

基幹業務システムを利用する各業務では、今まで障害が発生したことがほとんどなかったこともあり、システムが使えないと業務遂行はお手上げの状態であった。基幹業務システムが利用できない場合の事務マニュアルはなく、業務部門とシステム部門の対策検討もされていなかった。

対策

A 社では、今回の障害を契機に、従来の災害対応中心の IT-BCP をシステム障害対応にも拡張し、再発防止策を検討中である。内容としては、業務部門とシステム部門が協力してシステムの障害の規模に合わせた手作業による事務処理マニュアルの作成を行うことと、そのマニュアルに沿った訓練を定期的に実施することである。

効果

IT システムが使えない状態になった場合でも、手作業での対応など、顧客に多大な迷惑をかけないような業務継続が可能となる（登録・変更申請などを、窓口で一旦受付・手書き作成して、後日オンライン登録し郵送するなど）。

教訓

最近は業務処理をほとんどシステムに依存しているため、業務部門・システム部門ともにシステムが動かないと業務処理が全くできないと考えてしまう傾向がある。顧客視点で考え、様々な場面を想定した上で、システム利用不可時の手作業による代替業務マニュアルを作成するとともに、定期的な訓練を行う必要がある。

2.10 システム動作の疑義問い合わせがあった場合の対応に関する教訓（G10）

教訓 G10　関係者からの疑義問い合わせは自社システムに問題が発生していることを前提に対処すべし！

問題

A社は24時間365日コールセンター受付システムを運用している。コールは会員からの作業員派遣要請が主でその内容により最寄りのサービス拠点から現場に駆け付けて対処するものであり、1日のコール数は平均約3,000件／日である。本システムは全面的なシステム更改実施後1カ月あまりを経過していた。

ある日の午後、電話コールの一部が着信後に即切断されてしまう事象が発生していた。当初オペレータはいわゆるワンギリ（着信後、発信側から通話せずに切断するイタズラ電話）が通常時でもたまにあることから気にしていなかったが、コールを受けて現場に駆け付けた作業員がコールセンターに連絡をとろうとして、電話したところ通話がすぐに途切れる現象に気づき申告したことで異常に気づいた。また、ある通信回線事業者からコールの接続異常が時々発生しているが問題はないかと問い合わせがあったが、他の通信回線事業者に確認したところ異常は見受けられないとの回答だったため、問い合わせのあった通信回線事業者側の問題ではないかと回答していた。この事象はシステムを交代系装置に切り替えて復旧するまで約4時間近く続いており、この時間帯でのコールは約500件でそのうち50件程度が正常に受信できていなかったことが判明した。

A社のコールセンター受付システムの概要構成は以下のようになっている。（図2.10－1）

- 一般固定電話、携帯電話等の各回線種別との発着信通話用の回線収容基板を複数個装備
- 通話は電話機からの発信により、通信回線事業者からコールセンターの複数の端末装置に呼出信号が着信し、1台が受話器を取ることで応答信号が発信側に届くことで通話が確立
- 回線状態の正常確認のために端末装置側からすべての回線に向けてコールする回線テストを1日2回実施

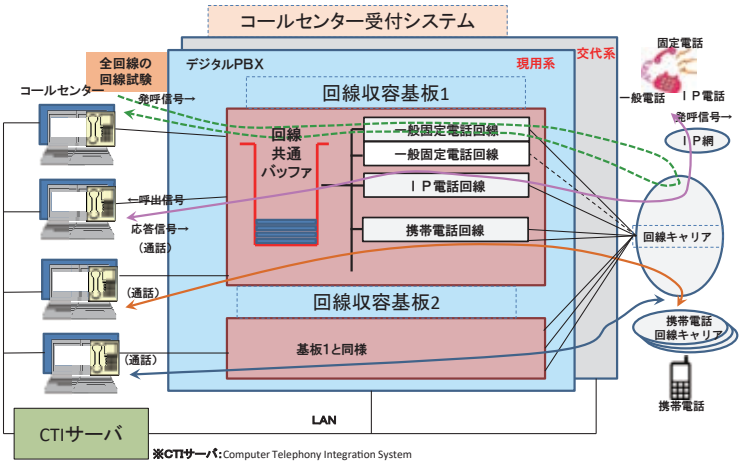


図2.10－1　コールセンター受付システムの構成概要

原因

A 社のコールセンターへのアクセスは携帯電話からの利用割合が増えていたため、コールセンター受付システムの回線収容基板内の一般固定電話回線用モジュールを削減し、携帯電話回線用モジュールを増やすこととし、通信回線事業者と連携して変更作業を実施した。しかし、回線テスト用の回線管理テーブルの変更も合わせて実施する必要があったが、作業が漏れたため、廃止した一般固定電話回線に対して継続して回線テストのコールが端末装置から発信されていた。

廃止した一般固定電話回線向け回線テストの呼出信号がシステム内で送出できず、送信待ちとなって回線収容基板の回線共通バッファ内に滞留していった。この回線共通バッファは回線収容基板を経由するすべての発信通話データの一時待機用に使われるものであり、バッファの空きがなくなると通話データの送出が出来なくなる。今回、回線テスト用の通話データが滞留し続けて、構成変更実施後約 3 週間で回線共通バッファがオーバフローしたため利用者からのコールの呼出信号に対する応答信号も送出不可となり、この回線収容基板を経由した通話は確立できずにキャリア側から切断される状況となった。

障害状況を図 2.10 – 2 に示す。

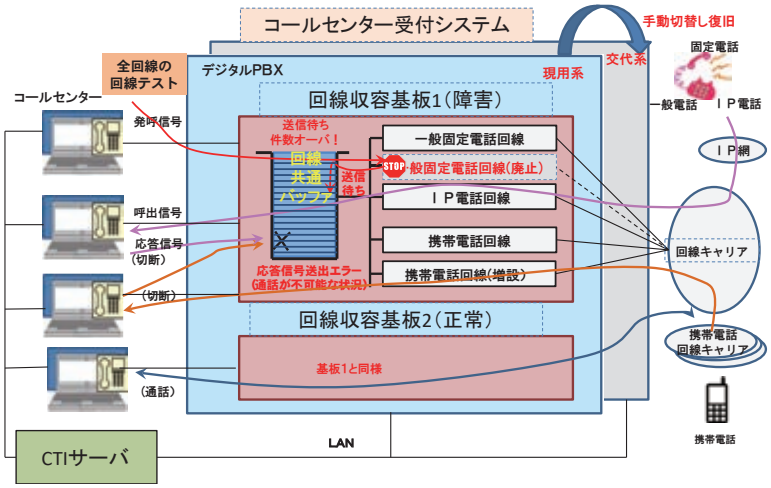


図 2.10 – 2　コールセンター受付システムの障害状況

異常となった回線収容基板 1 を経由しない場合は正常に通話ができていたため、システムの障害と認識するまでに時間がかかった。システム障害と認識した後は、緊急対策として交代系システムに切り替えて復旧させた。なお、通話中に交代系に切り替えても通話状態は利用者にとって違和感なく継続されることは仕様に盛り込まれており、テストで動作確認済みであった。

問題はシステム障害状態となってから正常に復旧するまでに約 4 時間が経過していたことであり、派遣要請した利用者は何回かのコール即切断を繰り返し、暫くしないと接続できない状況が続いたことにある。

対策

本システム障害事例でとられた対策を以下に記す。

① 直接原因と復旧措置

コールセンターに着信後即切断される直接の原因はデジタル PBX の回線収容基板の回線共通バッファがオーバフローしたためであり、そもそもこのバッファは発信電文の送出待ち待機用であることから想定外の事象であった。A 社はこのデジタル PBX を交代系に切り替えることを決定し手動で切替え運用が実施され、オーバフロー状況がなくなり、正常状態に復帰した。また、回線テスト用の回線管理テーブルの変更を実施するとともに、当面の再発防止のために、回線テスト運用を停止する運用措置もとった。

② 作業漏れへの対策

オーバフローとなった原因は一般固定電話回線の一部廃止保守作業時に、回線テストの設定も合わせて変更する必要があったが漏れたことである。ベンダは設定変更作業等で使用する保守運用マニュアルの全面的な点検を行い、作業漏れがないよう改善を実施した。

③ 障害状態検知への対策

障害状況を検知できなかったシステムの問題に対しては、ベンダは未送出電文がバッファに蓄積された場合、蓄積件数にしきい値を設定しオーバしたタイミングで監視コンソールにアラートを表示するようシステムの改善を検討している。

④ 通信回線事業者からの異常申告への対策

今回、ある通信回線事業者から接続異常が発生しているという申告があり、他の通信回線事業者に同様の事象が発生しているか確認を行っているが、発生していないとの回答であったため自システムの問題と気づかなかった。実際には他の通信回線事業者でも発生していたと思われる。A 社は複数の通信回線事業者との緊急連絡体制と問い合わせ手順を整備しコミュニケーションの精度の向上を図ることとした。

⑤ 交代系への切替判断

自システムの障害と切り分けたが、ベンダは原因調査に手間取っていた。A 社にとってサービス継続が最優先であり、交代系への切替による復旧の可能性があることから強制切替を決断した。これにより、サービスは復旧することが出来た。このことから、障害対策運用マニュアルを原因調査より復旧作業を最優先するよう改訂した。

効果

対策による効果を以下に記す。

①の対策により、同様のシステム障害は再発していない。②から⑤の対策では万が一発生した場合の速やかな検知とサービス継続を維持する効果が期待できる。

教訓

本事例からの教訓はいくつかあるが、障害復旧まで長時間かかった要因のひとつは通信回線事業者からの疑義の問い合わせがあったにもかかわらず、システム障害と認識できなかったことである。

システム障害と認識できていれば、早期の交代系への切替判断がなされシステムは復旧していたと思われる。

連携のある他のシステム事業者からの問い合わせに対し自システムの問題を優先して調査確認し、障害事象を確認したら速やかに復旧運用を優先して実施しサービスを継続することが肝要である。

2.11 システムの運用・保守に関する教訓（G11）

教訓 G11 システムの重要度に応じて運用・保守の体制・作業に濃淡をつけるべし

問題

A 社の社内ワークフローシステムに障害が発生し、連携している顧客向けサービスが終日全面停止した。障害発生後、システム関係者が集まったものの、状況を把握できず、障害個所の特定に多くの時間が掛かった。途中、様々なリカバリ手順を実施したが、なかなか復旧せず、システム停止時間は長時間に及んだ。

本ワークフローシステムの構成図（概要）を以下の図 2.11－1 に示す。また、図中には障害発生時の状況を順番に番号で示している。

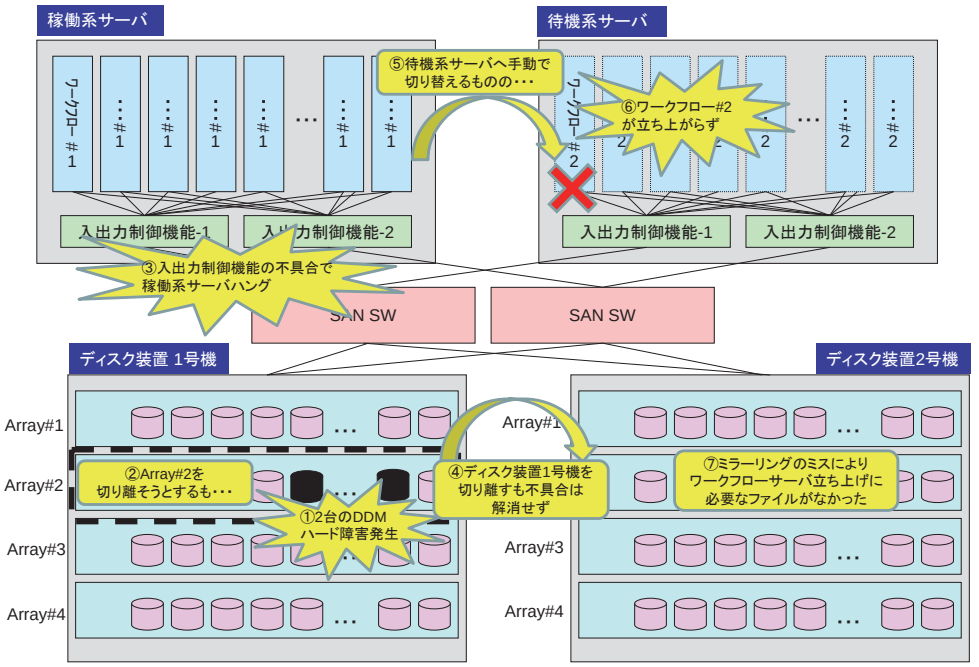


図 2.11－1　システム構成図（概要）

物理的なサーバ装置は、稼働系と待機系の 2 台にて構成されている。なお、この 2 台の物理的なサーバ装置内を入出力制御機能にて論理区画分割することで、複数の論理サーバとして機能させている。

また、ディスク装置は 2 台あり、こちらも二重化されている。各ディスク装置は 4 つのディスクグループ（以下、Array）によって構成されており、それぞれ RAID5[9] で構成されている。1 つの Array に障害が発生した場合には、当該 Array を切り離して稼働を継続する。また、一方のディスク装置全体に障害が発生した際には、当該ディスク装置を切り離して稼働を継続する構成としている。

原因

（直接原因）

本障害の直接の原因は、以下の 3 点であった。ここでは、システムが停止した原因と、復旧に時間が掛かった原因に分けて記載する。

○システムが停止した原因

- DDM（Disk Drive Module）のハード故障

  ディスク装置 1 号機の Array#2 において、ほぼ同時刻に 2 台の DDM が故障した（図 2.11－1①）。RAID5 の仕様により、Array#2 は稼働を停止した。

  なお、今回障害が発生した 2 つの DDM のうち 1 つは、DDM が自己診断機能で異常を検知し、物理的には機能を継続できる状態であるにも関わらず、製品仕様により自ら機能を停止していた。

- 入出力制御機能の不具合

  停止した Array#2 を切り離す過程で、入出力制御機能の製品不具合により、稼働系サーバがハングアップした（図 2.11－1②、③）。稼働系サーバがハングアップした場合、待機系サーバに自動的に切り替わる仕様となっているが、切り替わらなかった。

○復旧に時間が掛かった原因

- ミラーリング[10] の誤設定

  ディスク装置 1 号機をシステム全体から切り離すことで、稼働系サーバのハングアップを解消しようと試みたが、成功しなかった（ディスク装置の切離し自体は成功した）（図 2.11－1④）。次に、稼働系サーバの強制シャットダウンにより、待機系サーバへの切替えを実施したところ、他のサーバは立ち上がったものの、ワークフローサーバだけが立ち上がらなかった（図 2.11－1⑤、⑥）。

  これは、結果的にはディスク装置 1 号機と 2 号機間のミラーリングが誤って設定されていたことにより、ワークフローサーバの稼働に必要なファイルの一部が、ディスク装置 2 号機上に存在しなかったからである。しかし、この原因究明に辿り着くまでに多くの時間を要したため、復旧に時間が掛かってしまった。

（根本原因）

○システムが停止した原因

[9] データ及びパリティを複数のハードディスクに分散することで、信頼性を向上させる技術。
[10] 複数台のハードディスクに、同時に同じ内容を書き込むこと。

- 入出力制御機能については、当該製品の不具合について販売元のベンダも認識しており、他社には修正プログラムが提供されていた（他社でも障害が発生していたため）。A 社は、重大な修正プログラムについては、ベンダから報告を受けることとしていたが、他社で発生していた障害は影響が大きいものではないと判断されたため、ベンダから A 社には修正プログラムの情報が伝わっていなかった。

○復旧に時間が掛かった原因

- ミラーリングの誤設定については、システムの構築時は正しくミラーリングがなされていたものの、その後の運用保守会社（以下、B 社とする）による保守作業において、誤ったミラーリングの設定がなされていた。当該保守作業では当日の作業指示書を作成していたが、記載に誤りがあった。作業指示書は、上位役職者（現場管理者レベル）のチェックを受けるルールとなっており、ルールどおりチェックを受けていたが、上位役職者も誤りを検知することはできなかった。
- 障害発生時に関係者が集まったとき、それぞれのチームが独自の資料を持ち寄ってきて、共通した資料（システム構成図など）を持ち合わせていなかった。それに加え、障害発生時の対応マニュアル等が十分整備されていなかったため、全員で意思疎通を図るのに多くの時間を要した。

対策

本障害に対する再発防止策として、以下の対策を実施した。

○ DDM 保守運用の改善

- DDM の停止を極力低減させる観点から、DDM が自己診断機能で異常を検知した後、自ら機能を停止する条件が見直された DDM の制御プログラムの適用を実施した。
- DDM のハード障害率そのものを低減させていく観点からも、DDM についてはベンダとともに品質評価を毎月実施し、相対的に障害率の高い DDM 製造ベンダや、製造ロットのものについては継続的に予防交換を行うこととした。

○システムの重要度に応じた修正プログラム適用ルールの見直し

- ベンダが A 社へ報告・連携する修正プログラムを選定する際に、これまでは他社での障害における影響の大きさが要否の重要な判断基準となっていた。今後はこうした基準に加え、システムの重要度に応じて、修正プログラムの適用範囲を拡大することとした。システムの冗長化に関する修正プログラムについては、より幅広く抽出するよう見直しを実施した。

○システムの重要度に応じた保守作業におけるチェック体制の見直し

- B 社における上位役職者（現場管理者レベル）によるチェックだけでは誤りを検知できなかったことを踏まえ、システムの重要度に応じて、（B 社の）上長レベル等による二重のチェックを行う体制に変更した。また、これをルールとして定め、作業マニュアルに明記した。

A 社では特に、復旧に多くの時間が掛かり、顧客への影響が大きくなってしまったことに注目し、以下の対策を実施した。

○システムの重要度に応じたシステム保守における運用面、体制面の見直し

- 障害対応メンバ間の意思疎通がしやすいよう、システム共通の資料を準備し、障害対応の迅速化を図ることとした。なお、この資料には、全面障害が発生した場合にまず確認すべきポイント等も記載されている。
- 本番システム稼働後、障害発生時に待機系システムに問題なく切り替わるかどうかを確認するための、実機を使ったシステム障害訓練・テストを行うこととした。今回のミラーリング設定の不備も、実際に実機でテストを行っていれば検知することができたと考えられる。
- 実機を使った訓練・テストに加え、システムのベンダと協業して作成した障害発生のシナリオに基づき、机上での障害対策演習を行うこととした。
- システム障害発生時に、組織内で適切に情報の共有を行うため、システム障害の対策本部を新規に立ち上げた。この組織には、システム企画部門、リスク管理部門、広報部門等の責任者がメンバとして参画している。

効果

上記対策を行ったことにより、当該ワークフローシステムを含め、A 社においてはその後、重大トラブルは発生していない。

教訓

ここでは、復旧に時間が掛かってしまったことを重大な問題と考え、その主たる原因について、改めて考える。

前述の根本原因では、保守作業の内容について、ルールどおりチェックを受けていたにもかかわらず、誤りを発見できなかったことをあげたが、A 社ではシステムによらず、一律にこのチェックのルールを適用していた。また、修正プログラムの適用範囲もシステムによらず一律であったことを考慮すると、一番のポイントは、「システムの重要度に優先順位を付けていなかった」ことにあると言える。

A 社ではその後、基幹システムを中心に、顧客への影響の有無、影響の範囲、推定される損害額など、システムの重要度に応じてランク付けを行い、そのランクに応じてシステム保守対応を行うことにした。このランク付けに基づき、重要なシステムに対して実機を使った障害訓練を定期的に実施したり、修正プログラムを優先して適用したりすることとした。

以上から、今回の障害に横串を通して得られる教訓は、「システムの重要度に応じて運用・保守の体制・作業に濃淡をつけるべし」となる。

2.12 キャパシティ管理のマネジメントに関する教訓（その 1）（G12）

教訓 G12 キャパシティ管理は、業務部門とシステム部門のパートナーシップを強化するとともに、管理項目としきい値を設定して PDCA サイクルをまわすべし

問題

A 社のシステムはサービスの継続を優先する、データ非同期送受信（メッセージ交換型）のオンラインシステムである。このシステムの処理データ量には、図 2.12－1 のように以前から全体的な取引量の増加にともなう緩やかな増加の他に、突発的な事象による急増がみられた。

このA社のシステムにある日、処理能力（キャパシティ）を越えた注文が殺到し、サービスの時間が短縮となった。

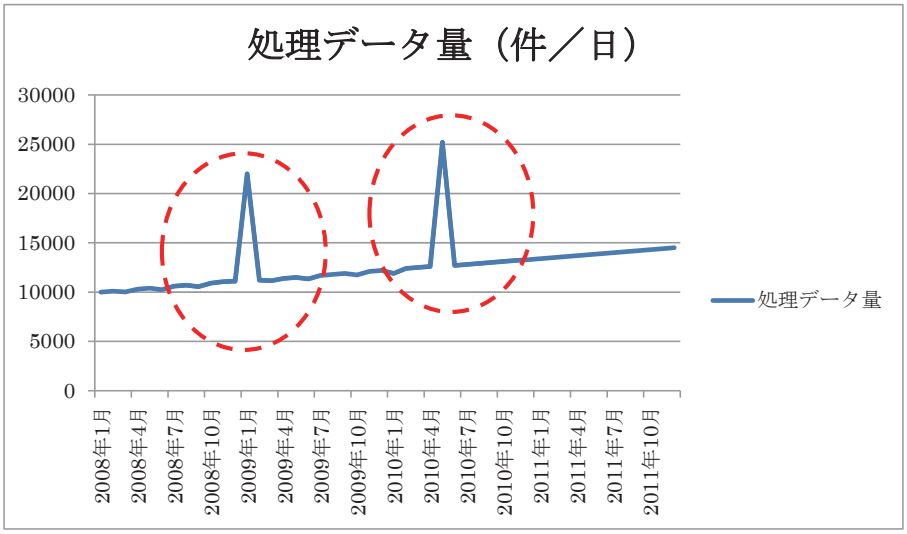


図 2.12－1　日単位の処理データ量

原因

近年はコンピュータでの取引が進み、処理件数の変化が激しい。全体的なデータ量の増加に加えて、想定を超える突発的な事象によるデータ量の急増が起こり、それに対応できずにサービスが停止してしまうこともある。データ量の急増に対応できない主な原因としては、業務部門とシステム部門とのキャパシティの合意形成やキャパシティの管理方法が明確でないことがあげられる。ここでは、この問題に対するある企業での対策（ベストプラクティス）を紹介する。

対策

実施した対策は以下の通りである。

① キャパシティ管理については、システムごとに責任を持つ業務部門を決め、適材適所で役割分担し、コミュニケーションをとる協力体制を作る。
（ア）業務部門がオーナーとなり、業務量とキャパシティの管理に責任を持つ。
（イ）業務部門とシステム部門が連携して図 2.12－2 のように、キャパシティ管理を行う。

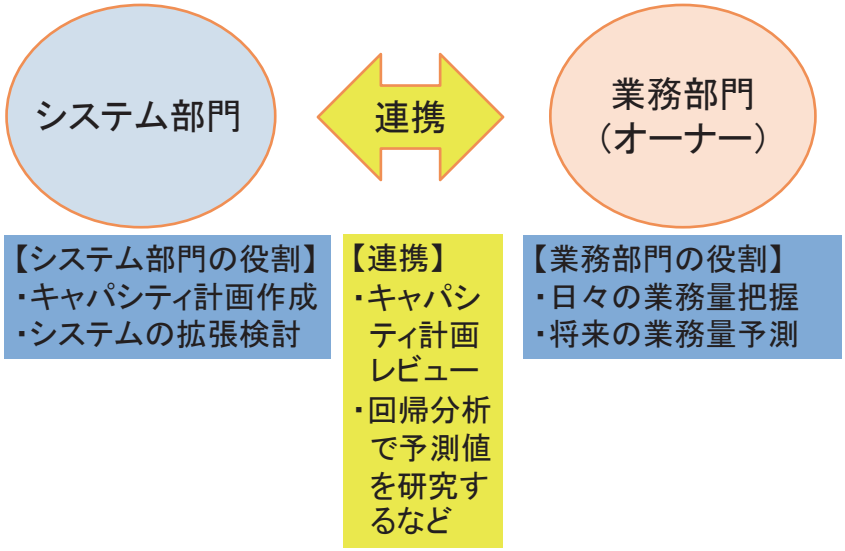


図 2.12－2　システム部門と業務部門のパートナーシップ

② 過去の実績をもとに算出したルール（例えば、「突発的な増加に対応可能な過去最高実績の 2 倍のキャパシティを確保する」といったルール）に基づいてシステムの性能を拡張する。
上にあげたルールに従う場合、単位時間当たり 100 件となる注文急増が発生したら、単位時間当たり 200 件処理できるようにキャパシティを増強し、次に同レベルの急増が発生したときにしきい値に抵触しないようにするといった対策が考えられる。

③ システムごとに管理項目とそれぞれの管理項目のしきい値を設定し、キャパシティの拡張方法や拡張限界等を明確化する。（図 2.12－3）。さらに、月次の PDCA サイクルをまわし、システム部門及び業務部門によってキャパシティの月次報告を確認する。

管理項目	閾値設定の考え方
業務量	業務量予測とキャパシティ増強期間を考慮して設定する
システムリソース	急激にサービスが悪化するポイントを基準として、それを超えそうな場合アラームが鳴るようにする
サービスレベル	サービスレベルに関わる要件を基本として、開発期間中のテスト実績や稼働後の実績を踏まえて設定する

閾値のレベル	Level1 危険域	Level2 重要警戒域	Level3 警戒域
対処分類（キャパシティの拡張方針）	投資を伴い即時対応が必要なもの	運用で対処可能なもの	情報を収集してから検討するもの
報告するレベル	経営トップに報告	関連部門の担当者レベルで会議	運用部門内で会議

管理項目	項目例	Level1 危険域	Level2 重要警戒域	Level3 警戒域
業務量に関する管理項目	1日の取引件数	7,000件	6,000件	5,000件
システムリソース	取引サーバの空きメモリ	10%	20%	30%
サービスレベル	取引応答時間	10秒	5秒	1秒

図 2.12－3　管理項目としきい値設定の考え方とレベルごとの設定例

④ 業務部門が日々の業務量やビジネス環境などから各管理項目について将来予測を行い、業務部門の予測に基づいてシステム部門がシステムの拡張を検討する。

効果

キャパシティに起因する障害を未然に防ぐことができるようになった。

教訓

もともとの想定を超えるようなデータ量の急激な変化に対応するには、業務部門とシステム部門の連携する体制の構築が重要である。（システムをシステム屋の目だけで見て、SLA だけをベースにして運用してはいけない。なぜなら、システムは実ビジネスを支えるためにあるのだから）

2.13 キャパシティ管理のマネジメントに関する教訓（その 2）（G13）

教訓 G13　キャパシティ管理は関連システムとの整合性の確保が大切

問題

A 社のシステムはサービスの継続を優先する、データ非同期送受信（メッセージ交換型）のオンラインシステムである。このシステムの処理データ量には、以前から全体的な取引量の増加にともなう緩やかな増加の他に、突発的な事象による急増がみられた。

このA社のシステムにある日、処理能力（キャパシティ）を越えた注文が殺到し、サービスの時間が短縮となった。

原因

近年はコンピュータでの取引が進み、処理件数の変化が激しい。全体的なデータ量の増加に加えて、想定を超える突発的な事象によるデータ量の急増が起こり、それに対応できずにサービスが停止してしまうこともある。データ量の急増に対応できない主な原因としては、システムが一連の系としてコントロールされておらず、相互の連携データ、連携時間帯等について統一した管理がされていないことがあげられる（図 2.13－1）。

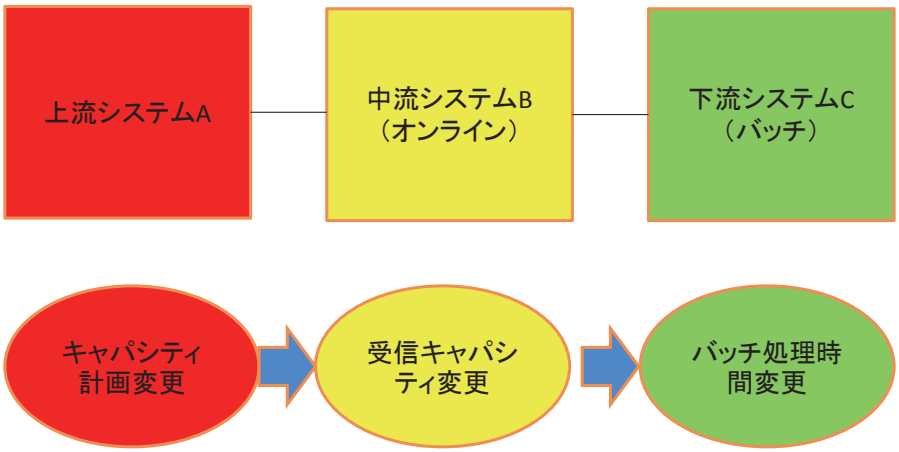


図 2.13－1　キャパシティ計画の変更によるシステムへの影響

対策

実施した対策は以下の通りである。

キャパシティ管理に関する課題を解決するために、全社的なキャパシティ管理業務を担う会議体を作り、システム連携情報を管理する。個別システムのキャパシティ計画の変更に対しては、図 2.13－2 のように、全システムの担当者が参加するキャパシティ管理会議でレビューし、システム連携情報の変更と影響の有無を確認して連携して対処する。

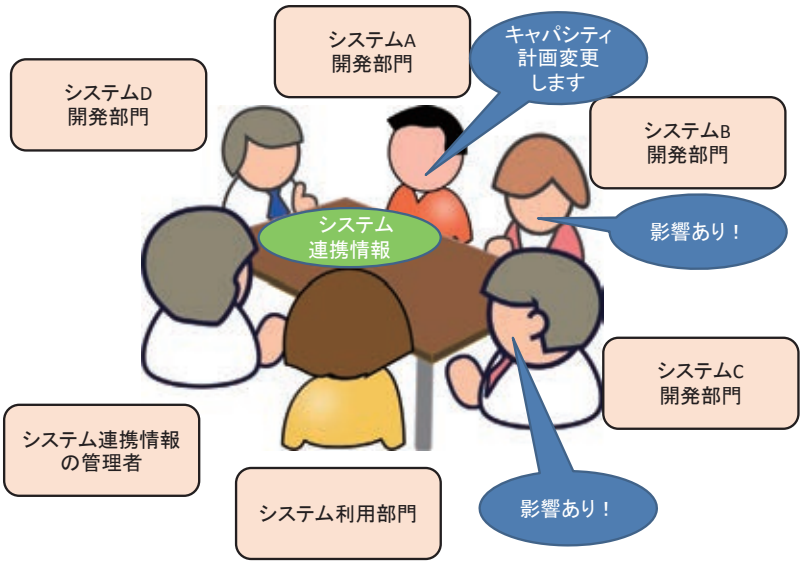


図 2.13－2　キャパシティ管理会議

効果

キャパシティに起因する障害の影響範囲の拡大を防ぐことができるようになった。また、このスキームは情報システム全般にわたって有効なので参考にされたい。

教訓

キャパシティ管理は関連システムとの整合性の確保が大切である。

参考）以下に、データディクショナリを活用して、システム連携情報を管理し、キャパシティ計画の変更によるシステム間の影響をチェックしている企業の事例（ベストプラクティス）を紹介する。

データディクショナリに①のドメイン一覧表や②システム間データ連携表の情報を登録して管理する。

① 全システムで言葉の定義を統一する。

図 2.13－3 のように、システムで扱うデータや処理等で、共通するものは同じ呼び名・表記となるように定める。

例）ドメイン一覧表

分類	項目名	定義	データ型	桁数
xxx	商品コード	商品の種類毎に定めた文字列	Char	8

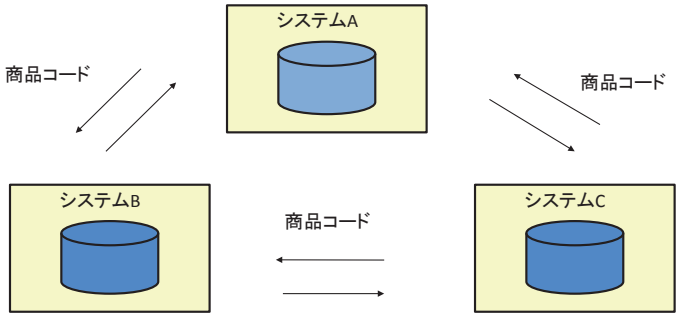


図 2.13－3　全システムで言葉の定義を統一

② システム間のデータ連携を可視化する。

図 2.13－4 のように、「どのデータが、いつ、どのくらいの頻度でどこからどこへ送られ、どれくらいの時間応答がないとエラーになるのか」が見えるようにする。

例）システム間データ連携表

連携時刻	データ名	サイクル	制限時間	送信元→送信先
9:00	照会データ	日	9:15	問い合わせシステム → 検索システム

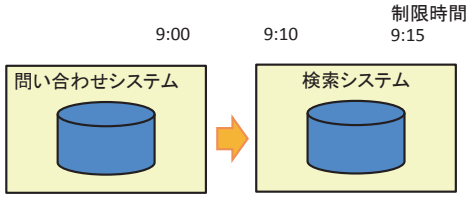


図 2.13－4　システム間のデータ連携を可視化

③ 以下のようなデータ管理方針を定め、新システム開発時、データ管理方針に準拠しているか、図 2.13－5 のように、フェーズごとの会議で確認する。

データ管理方針）

- データディクショナリを標準のデータ項目（アトリビュート）と各項目の定義（名称・意味・型桁・値）とし、これを管理する。複数システムで共通に使用されるデータを管理対象とする。
- システムの新規開発と更改は、データディクショナリのデータ項目定義と齟齬のないようにする。
- システム間データ連携及びデータベースのデータ項目について、図 2.13－3 や図 2.13－4 の表のような規定の書式を使用する。
- システム間データ連携部分については、データディクショナリのデータ項目定義と齟齬がないか確認するために事務局へ提出する。

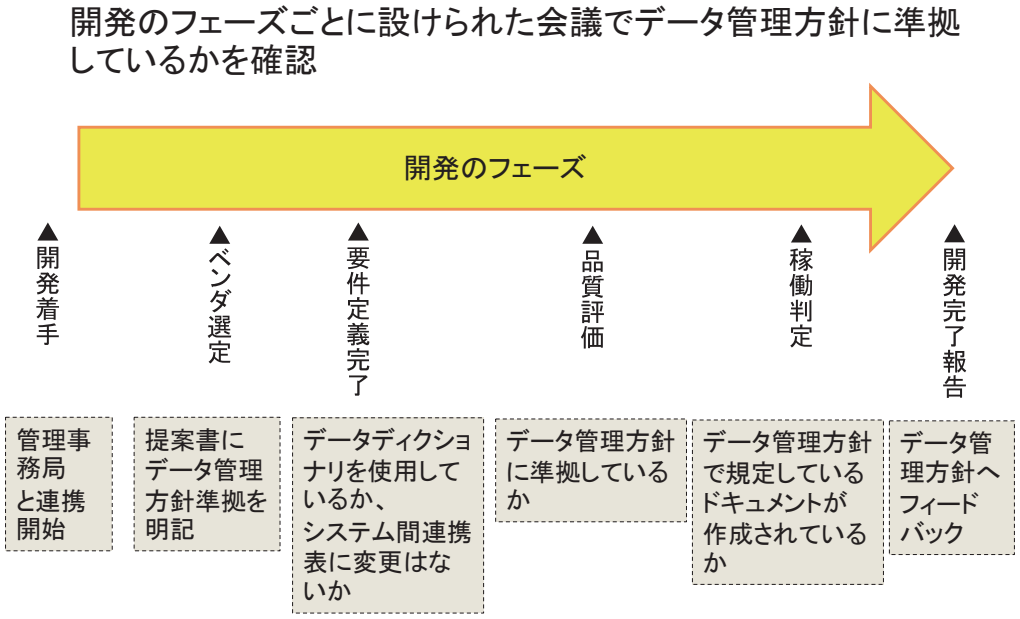


図 2.13－5　開発のフェーズごとに設けられた会議でデータ管理方針に準拠しているかを確認

2.14 キャパシティ管理のマネジメントに関する教訓（その 3）（G14）

教訓 G14　設計時に定めたキャパシティ管理項目は、環境の変化にあわせて見直すべし

問題

A 社のシステムはサービスの継続を優先する、データ非同期送受信（メッセージ交換型）のオンラインシステムである。このシステムの処理データ量には、以前から全体的な取引量の増加にともなう緩やかな増加の他に、突発的な事象による急増がみられた。

この A 社のシステムにある日、処理能力（キャパシティ）を越えた注文が殺到し、サービスの時間が短縮となった。

原因

近年はコンピュータでの取引が進み、処理件数の変化が激しい。全体的なデータ量の増加に加えて、想定を超える突発的な事象によるデータ量の急増が起こり、それに対応できずにサービスが停止してしまうこともある。データ量の急増に対応できない主な原因としては、設計時に定めた監視の時間間隔（キャパシティ管理項目の一つ）では十分にシステムを監視できていないことがあげられる。

図 2.14－1 に示すように、設計時に定めた監視間隔の 1 分単位では処理件数は安定しているように見えたが、1 秒単位でみると、処理件数が急増していて、処理遅延が発生していた。

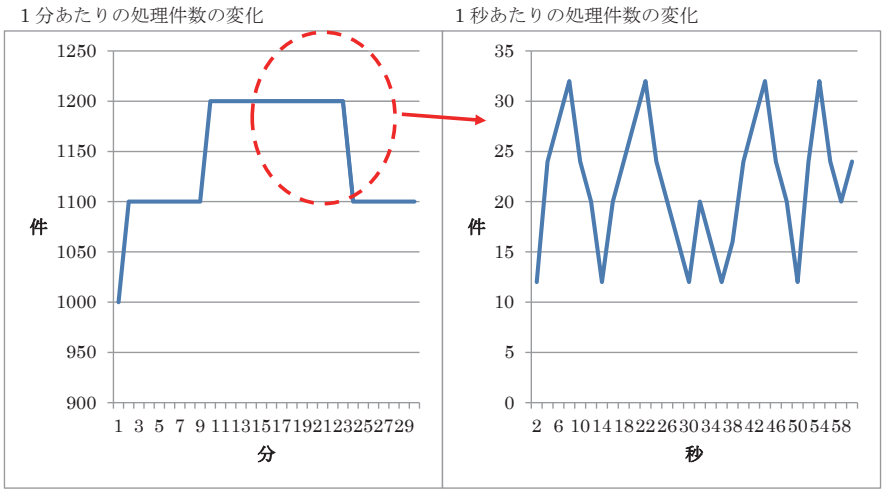


図 2.14－1　1 分当たりと 1 秒当たりの処理件数の変化

対策

実施した対策は以下の通りである。

① 現行システムに対しては、監視の時間間隔を含むキャパシティ計画の修正と、設定したしきい値（アラームを鳴らす処理件数のレベル等）の見直しを行う。

② ①で見直した内容は、次世代システムのキャパシティ管理要件として、開発時の要件定義に組み込む。

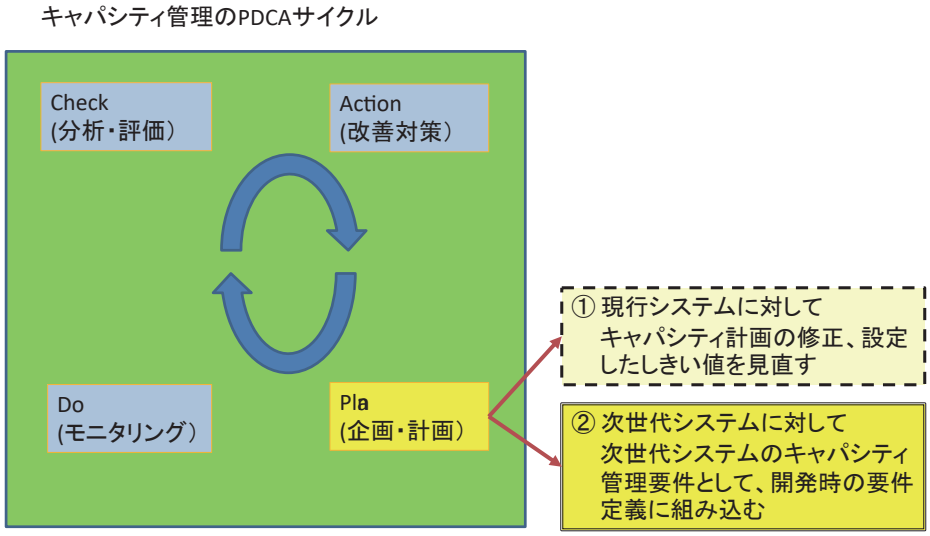


図 2.14－2　キャパシティ管理の PDCA サイクルと他のシステムへのインプット

効果

次世代システムの開発においてキャパシティに関する要件が適切に盛り込まれるようになった。

教訓

キャパシティ管理の仕組み（PDCA サイクル）をまわしていくと、設計時に考えていたしきい値に当たるものが実際は適切でなかった、あるいは環境の変化により適切でなくなっていたことが判明する場合がある。このようにシステムを運用する中で判明した問題は、次世代システムのキャパシティ管理計画へのインプットの課題として、開発時の要件定義に組み込むことが必要である。

2.15 保守作業時のリスク管理に関する教訓（G15）

教訓 G15　保守作業は「予期せぬ事態の発生」を想定し、サービス継続を最優先として保守作業前への戻しを常に考慮すること

問題

（システムの概要）

X 社のシステムは 24 時間稼働のオンラインシステムであり、業務の特性から高度な耐障害性を要求されている。システムは制御サーバ 1 号機と 2 号機で稼働系と交代系に二重化され、さらに各制御サーバ内で A 系システム／ B 系システムによるホットスタンバイ構成となっており、全体で 4 重化構成をとって運用している。

運用中の A 系システムと待機中の B 系システムはリアルタイムでトランザクションデータの系間同期が行われ、さらに 1 号機と 2 号機の間でもサーバ間同期が行われている。

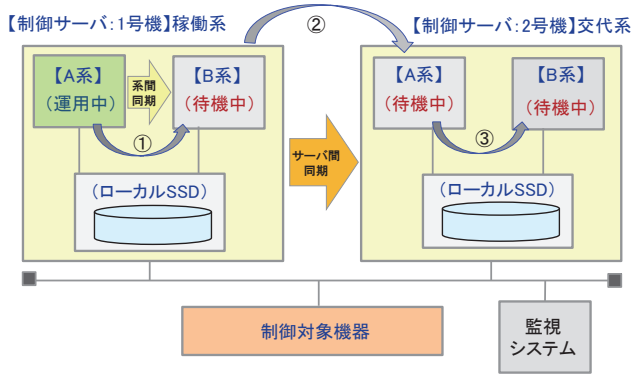


図 2.15－1　4 重化構成の概要

監視システムが A 系システムの障害状態を検知すると、以下のように自動的に切り替わり、トランザクション状態を引き継いでオンライン処理が継続される。

① 運用中の A 系システムが障害状態となると、同一制御サーバ内で待機している B 系システムに瞬時に自動で切り替わりサービスを継続する。この後、A 系システムはトランザクションデータを一旦リセットし、新たに運用中となった B 系システムからトランザクションデータの系間同期が開始され、数分後にホットスタンバイ待機状態となる。

② 稼働系制御サーバ 1 号機が障害状態（待機中のシステムがない場合も含む）となると、交代系制御サーバ 2 号機に瞬時に自動で切り替わり待機中の A 系システムが運用中となりサービスを継続する。この際もサーバ間同期が一旦リセットされ、新たに稼働系となった制御サーバ 2 号機から交代系となった制御サーバ 1 号機にサーバ間同期が開始され数分後にホットスタンバイ待機状態となる。

③ 交代系制御サーバが稼働系となり、サーバ内のA系システムが運用中となり、系間同期・サーバ間同期が再確立されホットスタンバイ待機構成となる。

④ 系間切替えやサーバ間切替えが実施された直後から同期処理が再確立するまでの数分間は瞬時の切替えが出来ない状態になっている。

（保守作業の実施）

ソフトウェア保守を行う場合は、自動切替え制御を解除した上で稼働中の制御サーバ1号機のB系システムに対してソフトウェア更新を実施し、作業完了後A系システムから保守作業後のB系システムに手動切替えをすることにより24時間オンラインシステムを中断することなく新たなサービスが開始される。この際、新たなソフトウェアによる不具合が発見された場合は、保守作業前の状態であるA系システムに手動で切り戻すこととしている。正常にサービスを開始できた場合は、残りのA系システム及び交代系の制御サーバ2号機に対し保守作業を行い、完了後に自動切替え制御を開始する。

（障害発生の状況）

障害発生の状況は以下の通りであった。（図 2.15－2）

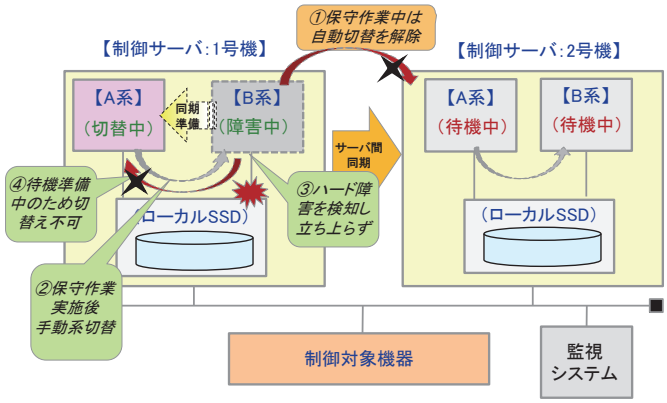


図 2.15－2　障害状況

自動切替え制御を解除（図 2.15－2 ①）しB系システムに保守作業を実施後、手動切替えを実施（図 2.15－2 ②）したところB系システムが立ち上がった直後にB系システムのハードウェアに故障が発生し停止（図 2.15－2 ③）してしまった。A系システムは新たな同期処理の開始中でありまだホットスタンバイ待機状態となっておらず（図 2.15－2 ④）切り戻すことができなかった。

通常運用であれば、ホットスタンバイ待機状態である制御サーバ2号機に瞬時に自動切替えが行われ保守作業前の状態でサービスを継続できるが、自動切替え制御を解除していたため、自動切替えが行われず、手動で制御サーバ2号機に切り替えて運用を開始するまで10分程度オンラインサービスが停止する状況となった。

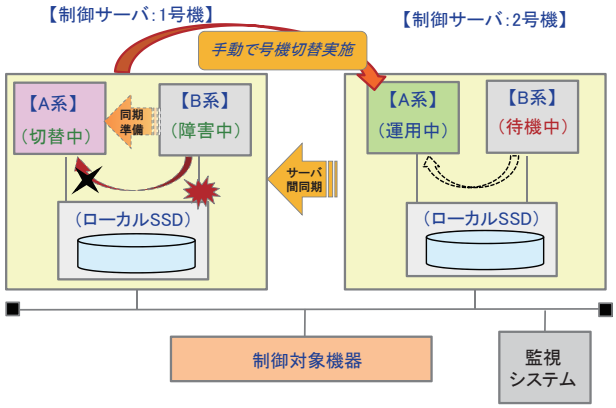


図 2.15－3　障害状況

原因

障害の直接の原因は制御サーバ１号機内のハードウェア故障であるが、ハードウェア故障への高度な耐障害性を確保したシステムであったにもかかわらず、サービスが停止となったのは保守作業時の自動切替え制御を解除していたことが原因である。保守作業時に自動切替えが発生すると予期しない事態となり保守作業自体が混乱することを危惧し、保守作業マニュアルで自動切替え制御の解除が手順化されていたものである。

また、一般的には保守作業は稼働中の制御サーバ１号機ではなく、交代系の制御サーバ２号機で実施し完了後切り替えるという手順がとられる。しかし、本システムでは保守作業完了後に切替えを行うとサーバ間同期が一旦リセットされ制御サーバ１号機がホットスタンバイ待機状態になるまでの数分間の間に障害が発生した場合に、保守作業前の状態である制御サーバ１号機への瞬時の切り戻しができないため採用していなかった。

対策

ハードウェア故障等は予期せず発生するものであり、保守作業時にも発生確率はゼロではない。再発防止対策として、サービス継続を最優先とし、自動切替えを解除しない状態で保守作業を実施することとし、万が一ハードウェア故障が発生した場合は、制御サーバの自動切替えが行われることを前提に保守マニュアルに対応手順を追加した。なお、稼働中の制御サーバでの保守作業にはリスクがともなうが、上記の理由から、保守マニュアル改訂後も従来通り稼働中サーバで保守作業を行うこととした。

効果

保守作業中に予期しない事態の発生があってもサービスが停止するリスクは解消され、改訂された保守マニュアルに基づいて継続的に実施されている。

教訓

保守作業は時間との戦いである。システムの保守作業を実施する場合、サービス継続を最優先とし、「予期せぬ事態の発生」を想定してシステム切替え時間・保守作業前の状態への戻し時間を考慮した保守計画を策定し実施することが重要である。

保守計画はサービス継続の業務特性レベルに応じて例えば以下のように策定することが考えられる。なお、実務では保守作業の内容等によりサービス継続と比較した上で策定する場合もあるであろう。

●サービス停止が許されない重要なシステムの場合：（ホットスタンバイ）

自動切替え制御を有効にしたままホットスタンバイ環境で保守を実施し作業完了後に切り替える。切り替え後障害発生時は自動切替えにより保守作業前の状態でサービス継続、保守作業は中止判断をする。

●数分間のサービス停止が許容できるシステムの場合：（コールドスタンバイ）

交代系で保守作業を実施し、作業完了後切り替える。切替え後、障害発生時は手動で保守作業前の状態を保持しているシステムに切り戻しを行うとともに、交代系も保守作業前の状態に戻してコールドスタンバイ環境を確保し、保守作業は中止判断をする。

●夜間・休日などサービスを停止可能な時間帯枠が確保可能（通常はこのケースが多い）な場合：

その場合でも翌日のオンラインサービス開始を最優先し予期せぬ事態が発生する場合を想定して、保守作業前の状態への戻し時間を確保しチェックポイントと戻し判断のタイミング等を必ず盛り込んだ保守計画を策定し作業を実施する。

また、保守作業計画書は上記業務特性レベルに関わらず例えば以下のような内容を明確にして作業をマネジメントする。

（保守作業計画書の内容）

- 各作業の所要時間見積もり、タイムチャートを作成
- 作業人員の確保と役割分担
- バックアップの取得と戻し手順の準備
- チェックポイントを複数設定
- チェックポイントでの計画と実施状況の差異確認と今後の作業予測
- 作業続行／中止の判断、緊急時の意思決定体制

など

2.16 本番環境における作業ルールに関する教訓（G16）

教訓 G16 本番環境へのリリースは、保守担当が無断でできないような仕組みを作るべし！

問題

世界中に顧客を持つグローバル企業 A 社は、いつでもどこからでもサービス要求を受付可能な 24 時間運転の Web システムを運用している。ある日、システム障害が発生したことにより、数十分間、サービス要求を受け付けられない状態となった。

本システムの構成（概要）を、図 2.16－1 に示す。

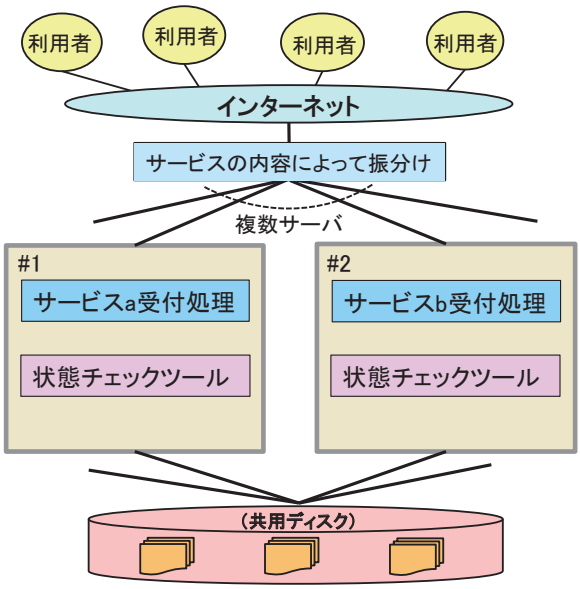


図 2.16－1　本システムの構成（概要）

① システムの構成について

サービスの種類ごとに、「サービス a 受付処理」、「サービス b 受付処理」等、それぞれサーバが別々に処理を行っている構成である（図中には 2 号機までしか示されていないが、実際はさらに複数のサーバで構成されている）。共用ディスクは、ユーザが送信したデータを一時的に保存しておく領域で、夜間バッチで顧客 DB に登録される運用となっている。（システム構成図に顧客 DB は書かれていない）

② 状態チェックツールについて

図 2.16－1 に桃色で示した状態チェックツールは、サーバの稼働状態（リソース使用状況、ディスク転送速度等）をチェックするツールである。本ツールは、使用頻度がそれほど高くないことから、現状、各サーバ上で手動にて起動する運用となっている。

③ 運用・保守の体制について

本システムは、ベンダに保守と運用を一括委託しており、24 時間体制で運用を実施している。アプリケーション資産の変更、システムへの機能追加など、システムの変更をともなう作業については、A 社主体のシステム変更会議による審議・承認を得る規則となっている。しかし現状、審議対象案件の明確な規定はなく、どの程度の変更ならば会議に諮るかは属人的な判断となっている。

原因

システム障害が発生した経緯について、図 2.16－2 に示す。

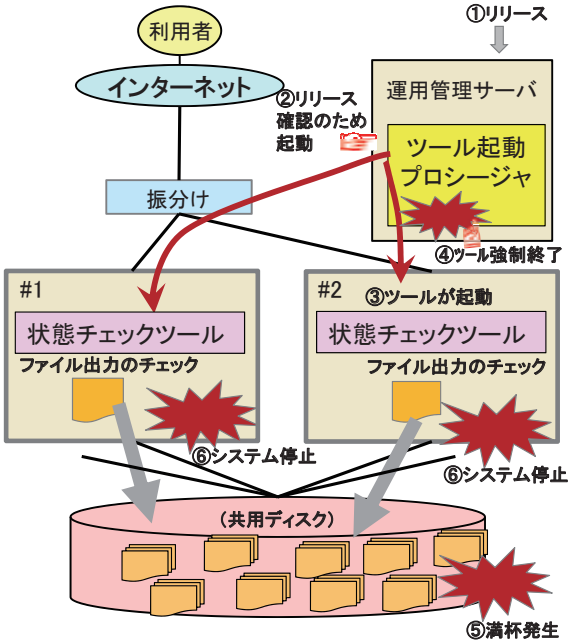


図 2.16－2　システム障害が発生した経緯

① 作業の負荷軽減を目的として、各サーバ上で手動にて起動している状態チェックツールを、一括で起動可能とするツール起動プロシージャ（バッチファイル）を作成し、運用管理サーバにリリースした。
② プロシージャのリリース確認のため、システムのオンライン中にこのプロシージャを起動した。
③ 各サーバで状態チェックツールが起動されたことを確認した。
④ 確認は済んだと認識し、各ツールの完了を待たずにプロシージャを強制終了した。

⑤ ツールが行うチェック項目の一つとして、共有ディスクへの書き込み出力機能のチェックがあるが、ツールが正常に終了しなかったことにより、この機能が繰り返し起動される状況となってしまった。程なくして、共有ディスクはツールの同機能が出力するテストファイルにより一杯となった。
⑥ 当該ディスクを共有するすべてのサーバがシステム停止に至った。

なお、作業担当者は、今回の作業はシステム変更会議への付議が不要な運用改善作業の位置付けと考え、会議に付議していなかったため、A 社側では当該作業が行われていることを認識していなかった。

本事象に対して、ベンダから以下 3 点の原因及び対策が示された。

【原因 1】当該作業は本来、システム変更会議に諮るべき内容の案件だったにもかかわらず、判断を誤ってしまった。具体的には、作成したものは運用改善のための簡単なバッチファイルであり、確認作業も問題なくすぐに終わるはずなので、会議に諮るほどのことはないと安易に考えてしまった。
【対策 1】システム変更会議に諮るべき作業対象を明確化する。

【原因 2】当該プロシージャを作成した担当者と、リリース作業を実施した担当者が異なっており、プロシージャを強制終了することによってどのような影響が出るかについて、情報の共有ができていなかった。
【対策 2】リリース作業担当者は、自分が扱うものについて、その動作条件を十分に確認した上で実施することを徹底した。

【原因 3】「ツールの強制終了」という、作業手順書に書かれていないことを実施してしまった。
【対策 3】作業手順書に書かれていない操作を禁止する。また、作業ルールに関する継続的教育を行う。

A 社はベンダからの報告を聞き、再発防止への対策はこれで本当に十分かを検討した。検討の結果、A 社は、保守作業担当者が承認なしに本番環境で作業を行っていたことがそもそもの問題ではないかと考えた。たとえ会議に諮る作業の対象を見直したとしても、個人の判断に委ねられる部分が残る限り、再発のリスクが残るからである。

対策

A 社は、再発防止策を以下のとおりとし、実施のためのルールを策定した。

○再発防止策

本番環境へのリリースは、保守担当が無断でできないような仕組みを作る。

○策定したルール

- 保守担当は、作業実施にあたってログイン ID の払い出しを受けなければならない。（作業ごとにログイン ID は異なる）
- ログイン ID の払い出しを受けるために、保守担当は作業内容について、A 社主体のシステム変更会議に必ず諮らなければならない。
- システム変更会議には、以下の内容を含む保守作業計画書の提出を必須とする。記載内容について、関係者と情報を共有し、当該変更部分の有識者を交えたレビューを行う。

  ※保守作業計画書に記載すべきもの：作業の目的、テスト環境における作業実施結果、作業タイムチャート、作業実施体制、作業手順（作業結果の確認方法を含む）、万が一の際の戻しの判断基準、連絡体制、ログイン ID 払い出し申請
- システム変更会議で承認を受けた作業手順以外のことを実施してはならない。（作業手順書に書かれていないことを実施してはならない）

効果

この対策を実施してから、A 社が認識していない作業によるシステム障害は発生していない。

教訓

本番環境へのリリースは、保守担当が無断でできないような仕組みを作るべし！

なお、今回の問題を引き起こした保守担当者は、作業効率の向上という現場改善意識から自ら工夫を行った。その手順に不適切な点があったが、作業改善の意欲自体は悪いことではない。もしかしたら、この担当者は、これまでに数々の改善を実践してきたかもしれない。この問題だけを契機に管理を厳しくするあまり、現場の改善意欲を委縮させることは好ましくない。再発防止施策のバランスについての配慮も重要であることを付記しておく。

2.17 重要サービスの運用に関する教訓（G17）

教訓 G17 サービスの重要度を識別し、それに応じた連絡体制や障害検知のしくみを作れ

問題

（システムの概要）

A 社の B サービスは、本社で作成・編集された情報を、通信回線を通じて全国のグループ企業の関連部署に配信するグループ企業内のシステムサービスである。配信される情報は音声や動画コンテンツなども含め多岐にわたっており、また重要性・緊急性が異なる情報が混在している。そのため、通信回線が混雑している場合でも優先的に重要情報を配信できるように帯域制御装置を使用している。配信される情報は重要度に応じて多段階に分けられ、最重要レベルの情報は、リアルタイムに伝達・処理されるべき情報として、関連部署に送られる。

また、システムは二重化されており、片方の帯域制御装置が故障した場合には、もう一方の側に寄せてサービスが継続できる構成になっている。ただし、最重要レベルの情報は図 2.17－1 のシステム構成概要図に示す帯域制御装置 #1 にだけ流れる運用になっていた。

（障害内容）

この帯域制御装置を駆動するソフトウェアは定期的なライセンス更新作業が必要であり、過去数度にわたり更新を実施しているが、特段の問題は発生していなかった。今回の故障が発生した際も、これまでと同じ手順で作業を実施し、更新そのものは正常に終了した。ところが、作業実施後すぐに B サービスの利用者から通信ができないという連絡があった。確認したところ、一部のグループ企業の拠点に対し、最重要レベルの情報が不通になっていることが判明した。帯域制御装置からはアラートは発出されておらず、また最重要レベル以外の情報配信は正常に行われていた。このため運用管理担当者は故障に気づくのが遅れ、約 30 分にわたり該当の拠点への最重要レベル情報の通信断絶が続いた。

ライセンス更新作業は、最重要レベルの情報配信に影響を与えないように、先に帯域制御装置 #2 において実施し、問題が発生しなかったことを確認したのちに、帯域制御装置 #1 において実施したが、その際に #1 で上記の故障が発生した。既に両装置とも同じ更新を実施済であったことから、帯域制御装置 #2 側に切り替えても同様の故障が再発する可能性があると考え、通信を帯域制御装置 #2 に片寄せすることはせず、他の策による回復を選択することにした。最終的に帯域制御機能を停止することで最重要レベルの情報配信を回復した。

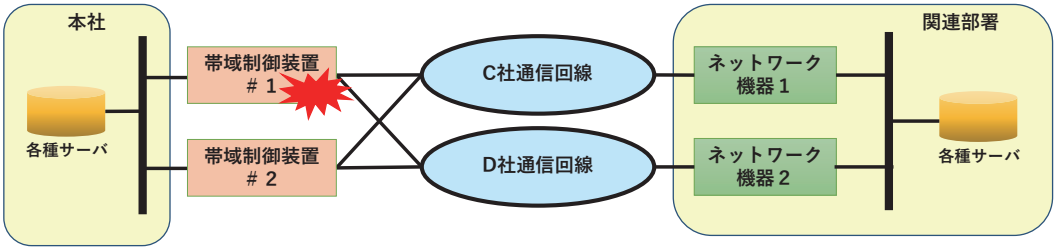


図 2.17－1　システム構成概要

原因

（故障の原因）

障害の状況から帯域制御装置の故障が疑われたが、原因の解明に結びつく特段のログが出力されていなかったことから、発生原因は特定できなかった。当該装置は海外製品であり、原因分析の対応に時間がかかるなど、故障発生時に十分なサポートが得られなかった。また、後日再現テストを実施したが現象は再現せず、原因は不明なままとなっている。

（対応が遅れた原因）

過去の同様の保守では問題が起きておらず、今回の作業においてもアラートは発生していなかったため、運用管理担当者は故障発生に気づかなかった（原因①）。

また、最初に障害に気づいたのはシステムを利用する業務担当者であったが、この時点では業務担当者から運用管理担当者への連絡ルートは確立しておらず、初動の遅れの一因となった（原因②）。これは、少しの遅れも許されない重要サービスの運用管理体制としては、不十分な状態であった（原因③）。

さらに、過去に実績のある作業であったため、関連部署への事前の作業実施連絡は不要として実施された（原因④）。事前に作業連絡があれば、もっと早く故障を発見する、あるいは現場での運用対処による影響回避策がとれた可能性がある。

対策

重要サービスに対する保守作業においては、特段の考慮をした作業環境やルールを用意する必要があると考え、以下の追加対策を実施した。

対策①　代替アラートを発出させる仕組みの追加

装置本体が持つアラート機能だけでは検知できない場合があるため、データ流量の下限しきい値監視など、他の機能や周辺機材を活用して代替アラートを発出させる仕組みを構築し、俯瞰的に故障発生を検知できるようにした。

対策② 連絡系統の見直し

重要サービスに関しては、迅速に運用管理担当者に連絡が入るよう、通常とは別の連絡系統を追加した。

対策③ 重要作業の分類

すべての保守作業項目（約 100 件）を抽出し、その作業が影響を与える各サービスの重要度に応じて、連絡要・不要などの分類を行い、関連部署と共有した。

対策④ 事前の作業連絡

たとえ過去に実績のある作業であっても、重要サービスに対する作業を実施する場合には、関連部署への作業連絡を必ず実施するようルール化した。

効果

重要サービスに障害が発生した場合、障害発生を早期に検知し、迅速に復旧対応が開始できるようになった。仮に今回の事例と同様の障害が発生した場合には、10 分以内に回復が可能となった。

教訓

本件の原因や対策を総括すると、「重要サービスに対しては特別な環境やルールを設定し、遵守を徹底する」という教訓事例になる。この事例の重要サービスは、重要と位置づけられるものの中でも、止まると社会的にも影響を与えかねないレベルのサービスであった。そのようなシステム / サービスに対しては、極めて厚い障害対応への備えが求められる。

より詳細にこの事例を見ていくと、以下のような点が見受けられた。

- 何度も実績がある作業だから今回も大丈夫だろう、という判断があった。また製品の輸入代理店から「ライセンス更新作業は動作に影響を与えない」という情報も得ていた。しかし影響を与える結果となった。重要サービスに関係する作業に対しては、「実績のある作業であっても、想定外の事象は起こり得る」と考え方を変えて、二重三重の備えをしておくべきであった。
- システムのアラートが発生した場合は、システム設置拠点から運用管理担当者にメールで通知が届くフローになっていた。運用管理担当者への通知が一刻を争う場合、より迅速に伝えるため、また確実に伝わったことを確認するという意味でも、電話を使った緊急連絡という手段・体制を加えておくべきであった。
- この業種・業務に特化して実績のある製品は、海外製品しか選択肢がなかった。しかしそうであればサポート面の不足は避けられないため、国内製品の場合とは異なる特別な運用ルールや体制の整備を行うことにより、サポートが不足する部分を補っておくべきであった。「場合によっては自組織だけではなく、海外製品製造元と国内代理店ベンダとの間の連携体制にまで踏み込み、改善に取り組む必要がある」という意識にまでは至っていなかった。

これらは、従来から当たり前に実施していたやり方や考え方には不備があり、障害からの回復の長時間化のリスクが内在していた、ということを表している。重要サービスの提供にあたっては、通常サービスでは必要十分とみなされ、見落としがちな仔細な部分まで深く考慮し、徹底的にリスクを排除した運用環境を整備していく必要がある。

その際には、極力思い込みを排除し、「大丈夫」と判断する根拠に曖昧な点がないか、をチェックすることが重要となる。例えば上記に「輸入代理店から影響を与えないと聞いた」とあるが、なぜ影響を与えないと言えるのか、根拠が曖昧な点が残る。こうした曖昧な点がある場合、「そうではないかもしれない」「仮に影響を与えたらどういうことが起こるのか」と想定を変えて現状のルールの検証及び事前対策を実施することにより、影響を軽減できる可能性がある。重要サービスに対しては、そのような特段深い考慮に基づいたルールを整備することが求められる。

本事例から得られたことを中心に、教訓は「サービスの重要度を識別し、それに応じた連絡体制や障害検知の仕組みを作れ」としたが、リスクについてさらに深く考えることで、この 2 点以外の課題や改善策にも気づくことができるであろう。

2.18 障害対策を立案する際に利用部門と取り決めるべき事項に関する教訓（G18）

教訓 G18　障害対策とは許容時間内の回復や停止中の業務継続まで具体化すること

問題

A 社で保有する社内特定業務向け基幹業務システムは、現場部門に対して 24 時間のオンライン業務サポートを実施している。業務は日中だけでなく夜間も継続しており、かつシステム停止による現場業務への影響が大きいことから、運用サポート部門も 24 時間体制でシステムの運用をサポートしている。業務システムは当該用途向けのパッケージを購入して構築されており、アプリケーションサーバ、DB サーバ、他システムとの連携サーバなど各層のサーバはそれぞれ冗長構成になっている。システムは本稼働後 3 年の間にサービスの一時停止に至る障害を 3 回経験していた。

A 社では、おのおのの障害の発生時点で原因分析と再発防止策を適切に実施し、それぞれ効果を上げてきた。しかしながら、このシステムが A 社業務の中核を担うサービスを提供するものであり、サービスが一時的にでも停止すると A 社の業務に甚大な影響をもたらすことから、A 社は障害発生時に早期にシステムを回復させる施策と、システムが停止中でも最低限の業務を継続できる施策の検討を開始した。

原因

A社が直面したサーバダウンと、その原因、主な再発防止策の概略は、以下のとおりである。

表 2.18－1　A 社システムにおけるサーバ停止の概要

No.	内容	停止時間	発生時期	直接の原因	影響	個別の再発防止策
①	アプリケーションサーバダウン	8 時間	本稼働 2 カ月後	パッケージの潜在的な不具合	大	開発者による 24 時間直接サポート
②	アプリケーションサーバ応答なし	3 時間	メジャーバージョンアップ 2 週間後	バージョンアップ時に実施されたリリースノート未記載のアプリケーション修正の不具合	中	すべての修正に対するレビュー会の実施 回帰テストシナリオの作成と自動実行環境の構築
③	他システムとのデータ連携サーバの二重起動（運用系と待機系）	1 時間	②の 1 カ月後	通信制御ミドルウェアの不具合による待機系切替えの誤起動	小	異常検知時のメッセージの整理と、障害発生時の運用要員の対応範囲変更

障害を経験し、対策を早期に実施することにより、発生から回復までの時間は確実に短縮できていることが、この表から読み取れる。

なお、上記の障害のうち①と②については、それぞれ教訓「T27 パッケージはサポートを買え」、「T28 パッケージを更新するときは、変更内容の詳細確認と回帰テストで二重に安全を確保せよ」としてまとめたので、詳細はそちらを参照していただきたい。

対策

上記の各障害に対する個別の再発防止策は確実に効果を上げているが、A 社では、このシステムが一時的にせよ停止した場合の自社業務への影響が看過できないことから、何らかの障害で一時的に停止したときに早期に回復する手段の構築と、システム停止期間中でも業務の停止を最低限に止める方法を検討した。

【システムの早期回復策】業務システムの二重化と常時並行稼働によるシステム切替えの迅速化

A 社の本番系システムはもともと各層のサーバが冗長化されており、運用系に何らかの障害があった際には待機系に切り替わるようになっていた。ところが、③の障害のように、運用系と待機系を巻き込んだ障害が発生した際には、通常の運用系と待機系の組み合わせだけでは、短時間での回復には不十分である事態が想定される。そのため、これを重視した A 社では、テスト用に保有していた別環境を増強して第二本番系として整備し、常時、本番系と同じ処理を並列で実行させておく運用を開始した。

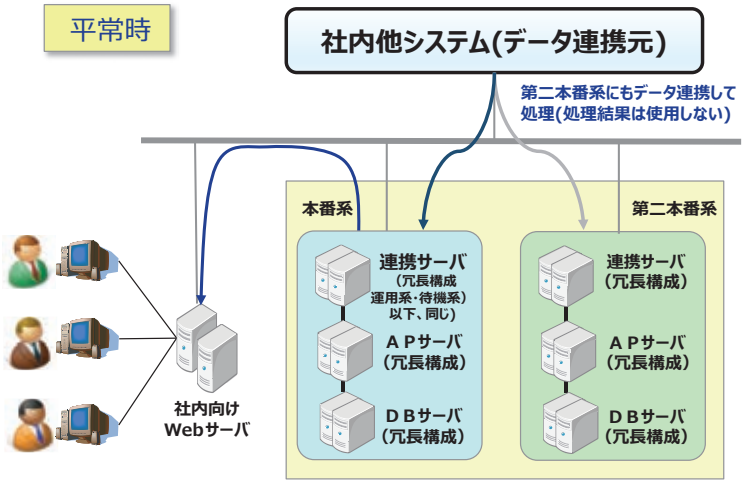


図 2.18－1　本番系と第二本番系での運用イメージ

図 2.18－1 の構成で第二本番系を構築し、他システムから連携されてくるデータはそれぞれで受信して、第二本番系でも同一の処理をさせ続ける運用をとることにより、本番系でトラブルが発生した際に、並行して稼働している第二本番系に短時間で切り替えられるようにした。これによりシステム運用者は、障害発生時の対応が長期化しそうな場合には、調査を打ち切って直ちに第二本番系に切り替えることによりサービスを再開し、その間に本番系での調査を継続できるようになった。

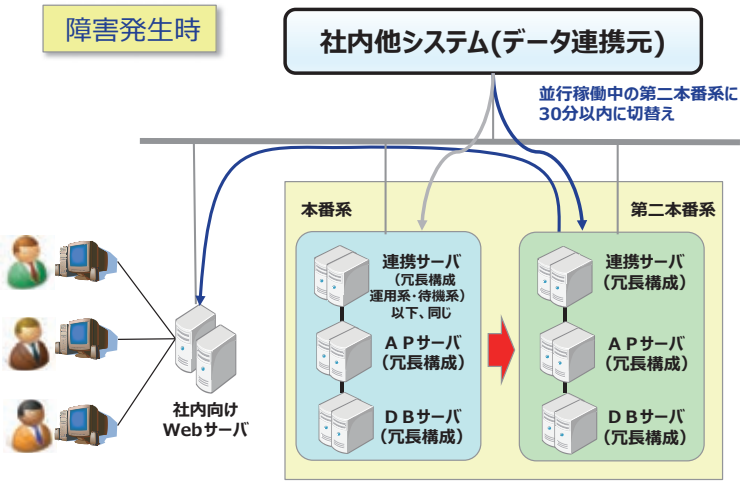


図 2.18－2　障害発生時の切替えイメージ

【業務を停止させないための対策】障害時対応をサービス利用者と明確化

また A 社では、このシステムで提供されるサービスの利用部門に対してこのサービスが使用できない場合の影響について以下の点を確認した。

① サービス停止から業務に影響を与えない（またはリカバリ可能な）復旧までの許容時間

② サービスが上記時間内に回復しない場合の業務側での代替策の有無と実現性

その結果、A 社ではこのサービスの停止後からサービス回復までの許容時間（業務の現場に対する SLA）を 30 分とした。また、サービスが回復しない場合の現場での最低限の業務継続施策を決定してもらい、それらを手順化した。作成した手順は、定期的に訓練を行い、継続的に現場への浸透を図ることとした。

効果

上記の施策により、この事例では以下の効果を得た。

(1) トラブル発生時の回復までの時間短縮

第二本番系システムでの並行稼働の開始と、サービスの利用者と合意したサービス再開までの許容時間にあわせた回復手順の構築により、これまでに発生したことがない障害に遭遇した場合も含めて、トラブルが発生しても利用者（さらにこのサービスにより提供されるエンドユーザへのサービス）への影響を極小化することができた。今後、障害が発生した際に、本番系での障害調査・対応に手間取り、本番系ではサービス利用部門と決定した 30 分以内にサービスを再開できない見通しになった場合には、そこで調査を中断して第二本番系に切り替えてサービスを再開することにより、現場への影響を最小限に止めることができるようになった。

(2) サービスがどうしても再開できない場合の代替手段の構築

上記 (1) のような施策を講じていてもサービスが再開できない万が一の事態が生じた際、その時点でこのサービスを使用しないでできる現場の業務を検討するのではなく、事前に現場で実施できることを想定して準備することにより、現場において、サービスを使用しないで維持する業務の優先度、実施範囲、実施手順をあらかじめ用意する契機になった。

教訓

基幹業務の一時停止は、多かれ少なかれ、自社の業務に何らかの影響を与える。そのため、IT システム／サービスの運用者は、システムが停止しないようにサービスの品質を安定化させることが最優先であることは言うまでもない。しかし、どのように対策してもサービスの停止は発生し得る。

そのため、サービス運用においては、その際の備えとして、以下の事項について準備をしておくことが、障害の影響を縮小させる施策として有効に機能すると思われる。

(1) サービス復旧までの許容時間は影響度に応じて異なる。サービスごとに停止不可、1 時間以内など許容時間を明確化し、それらを目標としてシステム／サービス復旧策を具体化する。

(2) サービスの復旧までの間、現場で実施できる施策を整理しておく。その上で、用意した施策を定期的に現場で訓練し、周知徹底させる。

昨今、IT システムおよびそれにより提供されるサービスは、業務の支援レベルにとどまらず、それ自身が業務そのものになりつつある。システム／サービスが停止した際には、それによって実施するビジネスそのものが停止することがほとんどである。ところが、IT システムは加速度的に複雑性を増しており、システム停止のリスクは増加傾向にある。そのため、IT システム／サービス運用においては、停止させないことの追求だけでなく、もし停止した場合に、早期にサービスを復旧させることがこれまでにも増して重要視されてきている。

この事例では、パッケージを導入して構築したシステムに対して、障害発生を機に運用品質を向上させると同時に、新たなパターンの障害に直面したときの対策として、回復目標時間をサービス利用部門と合意し、それを達成できる早期復旧策と、施策を実施しても万一復旧しなかったときにサービス利用の現場で採り得る対策の両方を構築して、万全の備えとしている。高度の運用品質を確保するために、運用系と待機系の組み合わせをさらに二重化するという、一般にはそう多くは実施されない方法を採った事例であるが、そのサービスレベルの維持に対する姿勢は、すべてのシステム運用に共通するものであると考える。障害発生を糧として、システム運用の品質をあらゆる方面から向上させる、それを継続する姿勢を教訓として発信したい。

教訓タイトルは、「障害対策とは許容時間内の回復や停止中の業務継続まで具体化すること」とした。

2.19 システム開発現場のコミュニケーションとモチベーション向上に関する教訓（G19）

教訓 G19　みんなで唱和！障害減らす教訓共有

問題

A社では、A社の開発部門が構築した制御装置を監視制御する物流拠点内の配送自動化システムを運用している。A社の開発部門は、同システムを、長期にわたり物流ニーズの変化に合わせた保守開発を行っている。

ある日の早朝にこのシステムの制御装置の1台にシステム障害が発生した（図2.19－1）。

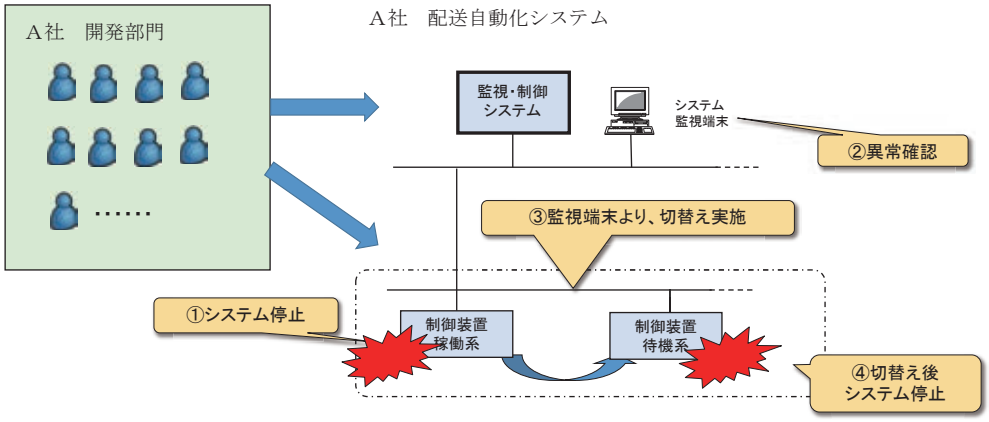


図2.19－1　障害状況

制御装置の稼働系が障害になり（図2.19－1①）、システム監視端末が、異常を検知した（図2.19－1②）。システム運用者は、端末の表示メッセージからハードウェアの問題と思い、すぐに、制御装置を待機系に切り替えた（図2.19－1③）。しかし、切替え後も障害となった（図2.19－1④）。

運用管理者は、すぐに原因が分からなかったが、システムを一旦リセットすることにした。そこで、システム全体を再起動したところ、稼働系の制御装置は正常に戻った。

原因

直接原因は、制御装置プログラムが持っている制限値オーバによる制御装置の停止であった。そのため、制御装置を待機系に切り替えても、待機系の制御装置も制限値オーバで停止してしまった。

根本原因は、今回の障害となった制御装置が持つ制限値は、システム構築当初から存在していた

ものだったが、このような制限値があることを知っていたのは、この個所の設計、製造を受け持った一部のメンバだけであった。そのため、この制限値が開発チーム全体で共有されることがなかったため、障害復旧が遅れてしまった。

対策

今回のような「担当者しか知らない」仕様があったために起きた障害を 2 度と起こさないために、A 社の開発チームは、「どこに問題があったのか」、「対策は何か」等、何回も議論を重ね、アイデアを出し合って、原因、対策をまとめた。その過程で課題がいくつか浮かび上がった（図 2.19－2）。

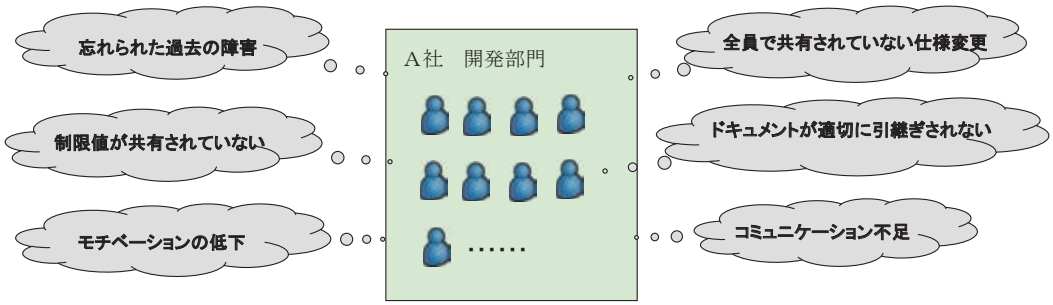


図 2.19－2　浮かび上がった課題

これらの課題にどのように対応するかを検討したときに、教訓として残し、それを常にメンバが意識できるようにしておくことが、新しいメンバが参加しても風化させないことにもなると考えた。そこで、開発チームの代表メンバで「教訓」を作成し、開発チーム内全員で集まったときに、その「教訓」をみんなで声を出して読み合うこととした。

具体的には、以下の「教訓」をまとめ、毎日の朝会（開発チーム全員への伝達事項等の共有を図るミーティング）の開始前に、全員で唱和している。

なお、「教訓」は、大きな障害が起こるごとに、チーム内で議論し合い、その度に追加・更新していった。

みんなが困る！ 知らない仕変（仕様変更）

システムやプログラムの制御において，テーブルのサイズ等には上限を定めることが一般的である。そして、この制限値は、システム間やプログラム間でも相互に影響をもたらすことがある。例えば、あるプログラムの制限値を拡張した場合、そのプログラムと連携する次のプログラムにも影響することが多々ある。したがって、この制限値がどのように決められているのか、チームメンバが知っているのか、などの点を管理することがシステム障害を減らすポイントである。そこで、この開発チームでは、「みん

なが困る！　知らない仕変」と各人が肝に銘じておくことに決めた。これによって、今後制限値変更があった場合、その変更内容を全員が共有できることになった。

みんなが実施！ 着実なホウレンソウ

システム障害対策においては、チーム全員の連係が重要である。特に多くの開発メンバで仕事をしている場合は、誰の担当個所の障害かを判断することや、各人の担当パートをつなぎ合わせてどこが障害かを探すことなども必要となる。そこで、全員の間で情報伝達が、早く正確にできることを目標に、着実な、「報告、連絡、相談」を行おうとの決意を込め、このようなメッセージとした。

みんなで共有！ 常に最新、設計書

仕様変更などが起きた場合の情報共有のポイントは、「何をもとに共有するのか」になる。そこで、開発チームの共有すべきものを設計書とした。一旦、設計書を情報共有の基盤としたことによって、システム更改に合わせて設計書の改版を行うことは、開発チーム全員にとって重要な作業となった。このような教訓メッセージを定めることにより、ソフトウェアの変更は行ったが、設計書の改版は後回しにしてしまったり、改版を忘れてしまったり、などにともなう情報の不整合をなくすことに努力した。

みんなで確認！ その疑問

システム障害が起こる前には、なんらかの予兆が存在していることが多い。そのひとつが、開発チームひとりひとりの「この点はどうなのだろうか?」、などの疑問から生まれる「気づき」ではないかと考え、「みんなで確認！ その疑問」とした教訓メッセージを定めた。この教訓メッセージを定めることにより、チーム内で確認し合うことに遠慮がなくなり、コミュニケーションが積極的に取られることを目指した。

みんなで目指せ！ 変化に敏感、気づく人

上記の「みんなで確認！その疑問」を実践するためには、ひとりひとりの「気づき」が発揮されないと実践できない。やはり、基本はチームひとりひとりのモチベーションの向上が欠かせない。そのためにも、ベテランメンバから新規参画メンバへの教育や、意見交換などの日ごろの活動が重要になってくる。モチベーション向上の意味でも、この教訓メッセージの全員での唱和は、重要な活動のひとつとして、定着している。

効果

このような活動を通して、開発チーム内では、システム障害が起きて対応が一段落するごとに、反省とこれからどう対策を行うかといった、メンバひとりひとりのシステム障害への積極的な取り組みが生まれ、自然と、みんなで話し合う風土が醸成されることになった。

朝会など、チームメンバが集まった場で、チーム全員で「教訓」を唱和することにより、チーム内のコミュニケーション向上と、各人のモチベーションの向上が図られる。

また、システム障害時の経験で学んだことで得た「教訓」をみんなで共有することで、その経験を風化させないことができる。

教訓

毎日の大半を、ＰＣに向かって行う仕事が多いシステム開発の現場や運用・保守の現場においては、このような「教訓」の唱和をチーム全員で毎日行うことに違和感を持ったり、なかなかそのような時間が取れなかったり、などの状況もあるであろう。

しかしながら、システム障害を未然に防ぐのは、そのシステムに参画しているひとりひとりにかかっている。この事例は、チーム内のコミュニケーションの円滑化とモチベーション向上に役立つ取り組みを行って、過去の障害事例で得た「教訓」を忘れないという大きな成果を生んでいる。

そこで、この事例を紹介することは非常に重要であると考え、この事例のチーム内の個々の「教訓」を取り上げるのではなく、「教訓」共有活動の重要性を取り上げた。

このような活動に取り組む多くの組織が現れることを期待したい。

2.20 システム運用環境変更の品質に関する教訓（G20）

教訓 G20 「システム運用環境変更時の品質向上」は正攻法の成功事例に学べ！

問題

A社は自社開発した基幹業務システムを運用して、販売店経由での商品販売を実施している。A社ではこれまで、運用するシステムの維持・管理は自社要員およびシステム開発・運用を支援する会社の要員により実施してきており、長年の経験を蓄積して運用の安全性を維持してきた。ところが、ある年、オープン系のインフラ上で構築した基幹業務システムを変更したのちそれを運用環境に反映（以下、この教訓ではシステムを改変した結果の運用環境への反映をシステム変更と記す）した際のミスが原因の重大トラブルにより提供するサービスを一時的に停止させ、自社および販売店の業務に大きな影響を与えたことにより、結果として、お客様に迷惑をおかけする事態を複数回発生させた。

A社ではこれまでも、トラブル発生の都度、原因分析を行い、適切な再発防止策を実施してきたが、トラブル発生により自社の基幹業務システム／サービスの提供を一時的にせよ停止させる事態を複数回発生させたことそのものが、ITシステムを維持・管理する組織の運営上の重大な問題であると認識し、オープン系システムの運用について組織レベルでの抜本的な対策を実施することにした。

原因

A社では、今回の件を含む、これまでの基幹業務システムを停止させるに至ったシステム維持・管理の失敗原因を、ハードウェア面とソフトウェア面に分けて分析し、その結果を表2.20－1に示すように分類した。

表2.20－1　A社システムの維持・管理において発生したトラブルの原因分類

分野	原因分類
1. ハードウェア	① システムリソース不足
	② 機器障害
2. ソフトウェア	③ システム変更が複雑化し誤りを誘発
	④ 人為的な単純ミスによる作業誤り
	⑤ 変更内容に対する組織的なチェック不足
	⑥ 熟練した基盤要員の不足による実施内容の誤り
	⑦ 万が一システムが停止した時の対策準備不足
	⑧ 過去と同じ原因によるトラブルの再発

例えば⑤に関しては、A社では従来から新規開発したシステムの本稼働の際には「リリースレビュー」

や「リリース判断会議」などリスクに応じた審査プロセスが整備されていたが、一方で、サービス開始後のシステムの変更についてはシステム変更を実際に行うチームによる「変更管理レビュー」が中心であった。そのため、その審査には表 2.20－2 に示すような問題が生じていた。

表 2.20－2　従来のシステム変更審査の問題点

No.	従来のシステム変更審査の問題点
1	作業手順など技術的な観点でのレビューが中心
2	チーム間でレビュー品質にバラツキがある
3	実施タイミング、作業により影響を受けるシステムの範囲、トラブル発生時のリカバリ計画などが網羅的にチェックできない
4	レビュー関係者の判断によっては、作業を実施することが部門内で止まり、報告する必要があるレベルの作業が全体を管理する部門に伝達されないことがある

表 2.20 －1 に示すような原因のトラブル発生を抑止するためには、従来実施してきたシステム変更の計画承認プロセスやトラブル原因分析・再発防止活動について、組織全体での見直しを検討する必要があり、またそれらの施策を実効性のあるものにするためには、システム変更を実際に現場で実施する個々の要員にシステム変更の重要性を改めて意識付けることが不可欠であると分析した。

対策

A 社は上記の分析結果にもとづき、組織全体で実施すべきプロセス改善施策を以下のように定めた。

施策 1. システム変更の品質を確保できるよう作業計画の承認プロセスを再構築する

施策 2. 過去の失敗を教訓として再発を防止するシステム部門全体での品質向上サイクルを確立する

施策 3. 障害の原因になるような行動を起こさないよう作業者の意識醸成のための活動を実施する

A 社は、上記施策を推進するために、これらの施策を専任で担当する組織（品質保証室：Quality Assurance Office）を、開発・運用部門から独立した組織として創設した。（図 2.20－1）

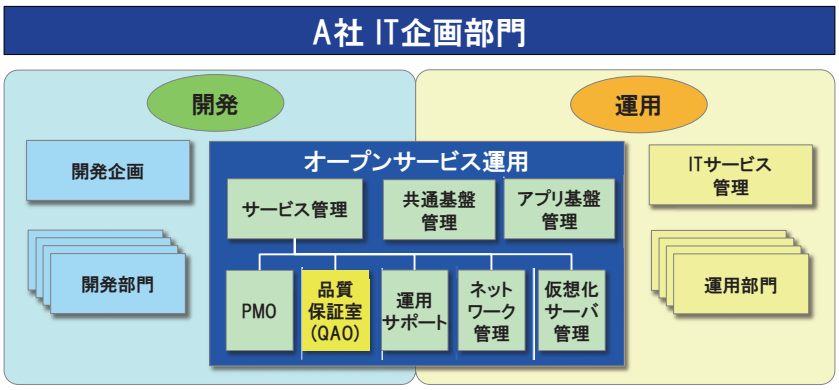


図 2.20－1　A 社の IT システム開発・運用部門における品質保証室の位置付け

(1) 施策1：システム変更計画の承認プロセスの再構築

品質保証室はまず、表 2.20－3 に示す「運用サービス指針」を整備し、今後のシステム運用全般に対する品質保証活動の柱とした。次いで、保有する各システムが要求する運用品質のレベルを、システムの停止が業務に与える影響度や変更実施の難易度を元に「運用品質確保ランク」（表 2.20－4）として分類し、どのような要求度のシステムがどれだけ分布するかを見える化し、今後の運用改善のベースとした。

表 2.20－3 「運用サービス指針」整備の実施内容

No	優先度	取り組み	内容
1	高	品質保証室の業務の設計	• 他社の品質管理部門からの事例収集、ヒアリングを実施し、作業品質を保証するためのプロセスの見直し、拡充を行い、品質保証室としての業務を確立
2	高	運用品質確保ランクの定義と展開	• システム停止時に被る損害の大きさに基づく「事業への影響度合い」とシステムの複雑性や規模に基づく「運用難易度」の組み合わせにより、運用品質確保ランクを5段階に分類して定義し、各システムをランク付け • このランクに基づいた運用品質目標の設定、リソース配分を行うための運用設計を実施

表 2.20－4　運用品質確保ランクの定義

運用品質確保ランク		事業への影響度合い			
		大	中	小	軽微
運用難易度	高	特A	A	B	C
	中	A	A	B	C
	低	A	B	C	D

保有する各システムの運用品質確保ランクの定義に続いて A 社では、システム変更の管理プロセスの改革のために、システム変更の実施を審査する会議体を「システムリリース評議会」として再定義した。システムリリース評議会を開催するかどうかは、新たに設定した運用品質確保ランク（表 2.20－4）と審査対象のシステム変更案件の作業難易度をもとに、規約（表 2.20－5）として定めた。これにより、どのシステム変更案件の審査も統一規準にしたがって確実に実施されるようにした。

表 2.20－5　システムリリース評議会の開催規約

システムリリース評議会開催要否		審査対象システムの運用品質確保ランク				
		特A	A	B	C	D
変更作業難易度 *2	高	必須	必須	必須	部門判断 *1	部門判断
	中	必須	部門判断	部門判断	不要	不要
	低	部門判断	部門判断	不要	不要	不要

*1 部門判断：評議会に付議するかどうかをシステム変更実施部門で判断
*2 変更作業難易度は事前検証有無、過去実績、作業時間、作業体制、作業内容から評価

さらに、これまで各部門で実施してきたシステムリリース判定の実効性を評価した上で、審査内容についても表 2.20－6 に示す見直しを実施した。

表 2.20－6　システムリリース評議会での審査内容

No.	従来の実施内容の問題点	改善点
1	作業手順など技術的な観点でのレビューが中心	システム開発・運用責任者と品質保証室をレビュアとして、システム変更実施タイミングの妥当性、連絡体制、トラブル時のリカバリ計画など、サービス運用環境への影響をチェック
2	チーム間でレビュー品質にバラツキがある	実施規準にしたがって漏れなく実施し、関係者がそろって統一されたチェック項目を審査 A）案件の概要 B）目的・経緯 C）リリース日時 D）実施スケジュールと体制 E）実施内容（過去実績、システム変更内容、対応期限、災害対策有無、検証方法、初日の監視） F）影響範囲（性能、キャパシティ、業務への影響、業務停止） G）トラブル発生時のリカバリ計画（作業時間、影響、トラブルであるかどうかの判断規準、判断者、サービス開始後にリカバリする場合の内容）
3	実施タイミング、作業により影響を受けるシステムの範囲、トラブル発生時のリカバリ計画などが網羅的にチェックできない	
4	レビュー関係者の判断によっては、作業を実施することが部門内で止まり、報告する必要があるレベルの作業が全体を管理する部門に伝達されないことがある	

また、上記のチェック内容は案件事例データとして蓄積し、以降、類似案件の計画時に計画や審議内容を参照できるようにした。また、システム変更管理プロセスそのものの見える化施策として、実施件数と成功率、トラブル原因別件数割合、お客様への迷惑度などを数値化し、KPI として取得するプロセスにした。

(2) 施策 2：システム変更の品質向上サイクルの確立

上記 (1) のような施策を実施しても、システム変更に起因するトラブルの発生がすぐにゼロになるわけではない。A 社では、発生するトラブルを謙虚に受け止めるとともに、トラブル事例をもとにした原因分析と再発防止策の立案を「なぜなぜ分析」を使用して実施し、それをシステム部門全体で周知する活動を開始した。再発防止策を立案する際には、品質保証室が検討をサポートし、「誰の責任か」ではなく「誰がどのタイミングで防止できたのか」が明確になるように進行を支援している。

また、インフラ系作業の常識集の作成や、考慮漏れチェックリストの公開、メルマガ等での最新情報の発信、ヒューマンエラー研修の実施などにより、システム運用で得た貴重なノウハウを蓄積し伝授する活動を上記と並行して実施することにより、品質向上のサイクルが回るようにした。

(3) 施策 3：システム変更担当者のマインド醸成

システム変更の品質を向上させるためには、各種チェックの強化や再発防止の徹底だけでは足りない。始めは効果的な施策も時間の経過とともに陳腐化する、運用担当者の「慣れ」が怖い、業務プ

ロセスだけでなく担当者の意識も変える必要がある、と考えたA社は、品質向上に対する担当者のマインド醸成のために表2.20－7に示す7つの施策を実施した。

表2.20－7　システム変更担当者のマインド醸成施策

No.	マインド醸成のための施策
1	運用基礎研修（新入社員、他部門からの転入者、転職者）
2	ヒューマンエラー研修（基礎編は社外から適した教育を導入、実践編は過去トラブルをベースに自社で企画）
3	なぜなぜ分析研修
4	システム運用チームと品質保証室との対話会
5	最新の情報をメルマガ発信（週次）
6	システム変更の品質優秀者表彰（社内だけでなく社内の委託先要員も対象）
7	半期ごとに活動内容を振り返りシステム運用部門内全体に情報提供

上記の施策は、システム変更においては、各作業者が以下の基本動作を遵守することが失敗を防ぎ、システム変更の品質を向上させることに直結するという意識を各担当者に定着させることを狙いとしている。

- 作業をする際には実施内容を計画し、ルールに沿った承認を受ける
- 承認された手順以外の操作はしない
- 入力したコマンドに誤りがないか実行前に再度確認
- 想定外のエラーが出たときにはすぐに報告し、勝手に行動しない

「よかれ」と思った行動がトラブルを生むこともある。このような誰もが当たり前だと思うことを、基本動作として担当者に繰り返し浸透させ、それを守らせないと、現場での作業品質の向上にはつながらない。

効果

多方面での施策の組み合わせが奏功して、A社のシステム変更品質には以下の効果が得られた。

(1) 数値に直接現れる成果

① 活動初年度

- 前年度比でトラブル発生が半減
- 作業ミスは4割減
- トラブル影響度指数（トラブルが発生した場合に業務に与える影響度を運用品質確保ランクと発生したトラブルの重症度ランクをもとに数値化した指標）も3割減

→ トラブルが発生した場合でも、これまでよりも影響を小さくできている

② 二年目

トラブルの重症度が軽微（業務に何らかの影響が出たもの）以上のトラブルの発生件数は前年度とほぼ同じだが、トラブル影響度指数は 4 割減

効果を数値で端的に評価できるようになったことそのものも、この施策の大きな効果である。

(2) 結果に対する評価と見直し

単純な作業ミスがかなり削減できている一方で、実施内容・手順に対する考慮漏れによるミスが残り、相対的に目立ってきている。これに対しては、発生した事例に基づくインフラ系常識集、考慮漏れチェックリストの追加を行い、順次水平展開を実施予定である。考慮漏れ対策についてはノウハウの体系化が難しく、これは今後の課題になっている。

教訓

本件は、システム運用環境でのシステム変更を確実に実施し、サービスをお客様に安定的に提供するために、システム変更の承認、トラブル再発防止、人材育成の各プロセスの改善を専任する組織作りから始めた事例である。

システム変更は、利用の現場から「無事に実施できて当たり前」、を期待される、システム開発・運用の中でも品質に対する要求レベルの高い作業であるが、それを確実に達成するためには、以下のようなプロセスが必要であることをこの事例は示している。

(1) 保有するシステムそれぞれが要求する運用品質レベルを明確化し、システム変更の際には、レベルに応じた検査を、実施内容に関する知識を有する第三者の目で行う
(2) 発生したトラブルを貴重な経験と思い、個人の責任で終わらせず前向きに原因分析と再発防止策の立案を実施して組織全体で再発を防止する
(3) 各要員がシステム変更に取り組む際に基本動作として身に付けておくべき意識（何をすべきであり、何をすべきでないか）を、各種施策により繰り返し醸成する

これらのことを順次実施し、組織全体でシステム変更の品質を向上させることが、システム運用全体の品質を継続的に向上させ、組織を成長させる最も確実な施策である。

A 社ではシステム変更時の品質の向上施策は「幕の内弁当のようなもの」と例えている。『これをやれば必ず改善する』という特効薬のようなものはなく、一件一件確実に改善するための地道な取り組みを組織化して 1 つのパッケージにすることにより施策の全体で成果を生み出すことができる、との考えで取り組んでいる。

2.21 システムに利用期限のある機器／ソフトを組み込む際の教訓（G21）

教訓 G21　サーバ証明書等の有効期限の確認方法を工夫せよ

問題

A社では全社向け基幹業務システムを構築し、同システムを使用して来訪する自社顧客向けのサービスを全事業所で提供している。サービスの提供は平日日中から夜間までだけでなく、一部の事業所では休日も実施している。業務システムは2年前に構築したものであり、システム保守はシステム構築を委託した先の事業者が継続して担当し、システム運用はA社が自ら実施している。A社の基幹業務システムは端末側アプリを仮想化し、サーバと仮想端末が通信する構成にしていた。

A社基幹業務システムの構成を図2.21－1に示す。

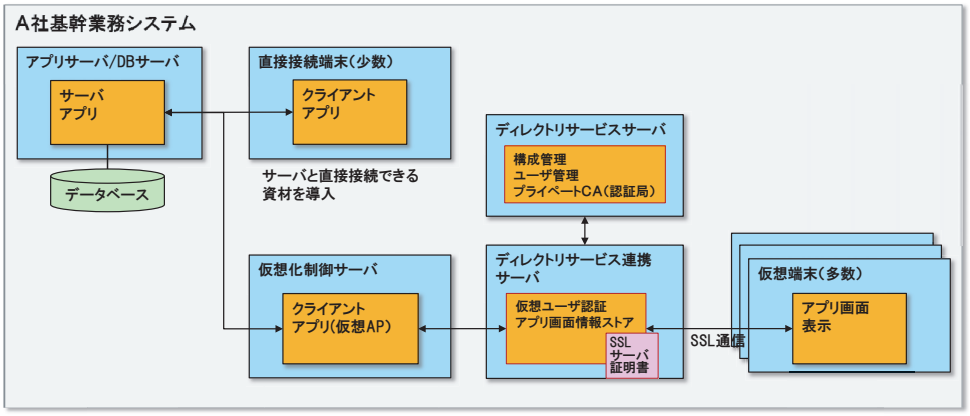


図2.21－1　A社基幹業務システムの構成

ある休日の朝、サービス利用の現場で、仮想端末上で基幹業務が起動しないという現象が発生した。仮想端末を起動すると「ディレクトリサービス連携サーバとの接続ができない」とのメッセージが表示されるだけであり、仮想端末上で動作するアプリケーションが全く使用できない状態になっていた。この現象は特定の端末によらず、すべての仮想端末で発生していた。表示されるメッセージから原因が端末の仮想化技術に関係するとの報告を保守委託先から受けたA社のシステム運用責任者は、休日にサービスを提供する事業所内に少数ながらも直接アプリサーバと接続する端末（以下、直接接続端末）が用意されていたことから、トラブルが収束するまでの間、直接接続端末だけを利用することを利用の現場に連絡して、当日の業務を開始した。

しかしながら、直接接続端末は数量が限定されており処理の順番待ちが多数発生したことから、一部の顧客が自分の順番を待ちきれずに帰るという事態に陥った。

原因

仮想端末に表示されるメッセージから、基幹業務システムが仮想端末で起動しなかった原因はディレクトリサービス連携サーバと仮想端末の通信途絶であることは明白であった。通信途絶が発生した直接の原因は、サーバのハードウェア故障等によるものではなく、ディレクトリサービス連携サーバの SSL サーバ証明書（以下、サーバ証明書と記す）の有効期限がトラブル発生前日までで切れ、当日以降は有効でなくなったことにより、サーバと仮想端末が SSL 通信できなくなったことであった。

トラブル発生時の状況を図 2.21－2 に示す。

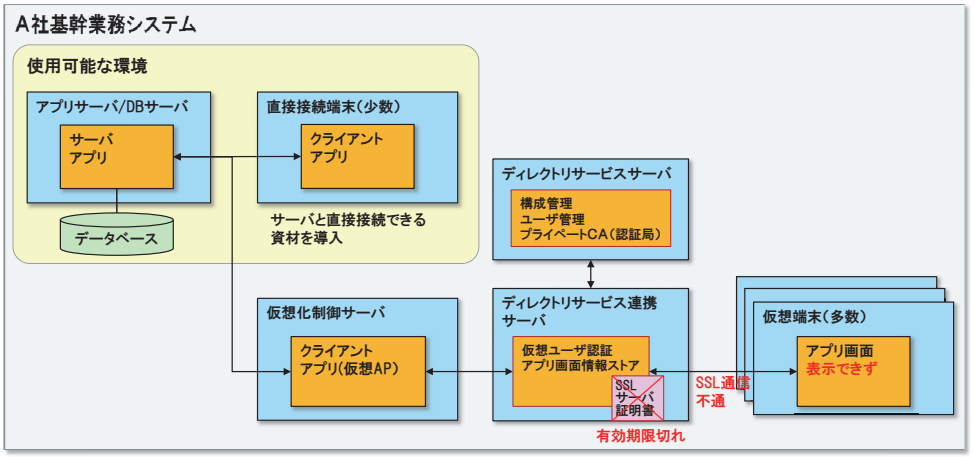


図 2.21－2　トラブル発生時の状況

そして、業務運用中にサーバ証明書の有効期限切れを発生させたシステム運用管理上の原因は、ディレクトリサービス連携サーバにサーバ証明書が組み込まれており、2 年ごとに更新する必要があったこと、さらにその期限がトラブル発生前日までであったことを A 社もシステム開発保守委託先の保守担当者も認識していなかったことであった。

幸い発生したのが休日であったため平日に比べればサービス利用者への影響は小規模で済んだが、少数ながらもサービスの提供が滞ったことにより一部の顧客がサービスを受けられなかったことを重く受け止めた A 社は、再発防止に向け、根本原因の分析を開始した。

対策

トラブル発生の原因が上記であることが判明し、A 社は直接の対応として以下の処置を実施した。

- ディレクトリサービス連携サーバのサーバ証明書を作成し、ディレクトリサービスサーバに登録
- 上記対応が完了し、システムを復旧するまで直接接続端末を増設

次いで A 社ではこのトラブルの再発防止のため、自社および開発・保守委託先会社の両当事者を

集めて根本原因の分析と採り得る再発防止策の選択を実施した。

再発防止策を検討する際には、単にサーバ証明書発行の期限切れの再発防止だけでなく、システム運用において定期的に実施する必要のある管理上の作業全般を対象とした再発防止（他個所の点検への水平展開）の観点で、実施すべきこと及び優先度の洗い出しを行った。検討の結果を以下に示す。

表 2.21－1　根本原因の分析および再発防止策の検討結果

原因			対策案（*斜体は水平展開*）	対応要否
仮想アプリが動作するサーバの SSL 認証が運用中に期限切れになった				
第一階層の原因	誰も期限を監視していなかった		サーバ証明書を毎年定期的に更新して期限切れを防止するよう運用変更	要 すぐ
			他のサーバ証明書やパッケージライセンスの監視に漏れはないか確認 *すべてのサーバ証明書の有効期限、管理担当者等を台帳管理し、定期的に確認する*	*要* *すぐ*
	第二階層の原因	仮想環境上で SSL 通信を使用すること、それには構築時のサーバ証明書発行と運用開始後の定期的な更新が必要になることをA社では認識していなかった	構築会社と同レベルの仮想化環境構築技術を内部留保	不要 現実的でない
		サーバ証明書の期限の管理者が曖昧 （開発委託先の意識は、証明は認証サーバが行うもの、認証サーバの管理はA社の担当）	サーバ証明書の定期更新を保守委託先の作業として契約時の仕様書に明示	要 次年度
			他に管理者が曖昧になっているハード、ソフトはないか確認	*要* *すぐ*
			管理すべき対象がA社側でもすべて認識できているか確認	*要* *時期調整*
		サーバ証明書の定期的な更新が必要であることが環境構築時に開発委託先からA社に伝達されていない	*他に同様の対処を必要とする案件がないか確認する*	*要* *すぐ*
		第三階層の原因：サーバ構築時の設定手順書にはサーバ認証に関する記載がなかったが、構築担当者は仮想化に詳しくサーバ証明書を登録できた それがA社にも開発保守委託先の保守担当者にも引き継がれていない	*サーバ証明書を組み込んでシステムを構築する際には、開発委託先に対してA社側からも引継ぎ事項の有無を確認する、* *引継ぎ事項チェックリストに確認事項として提示し、調査結果を相互確認することにより漏れを抑止する。* *（開発保守委託先における引継ぎ漏れ防止への自律改善策は別途実施する）*	*要* *次のシステム構築契約時*

効果

A 社では以前からオフィス等のソフトウェアのライセンスの保有状況や有効期限の管理を実施していたが、それに加えて表 2.21－1 に記載された各対策を実施することにより、今回トラブルが発生したサーバの証明書の有効期限にとどまらず、電子的な証明書を利用する他のシステムを把握し、それらの証明書の有効期限と更新作業者、保守委託契約書の記載を調査し、定期的な更新が漏れないよう管理監督することにより、運用中に不測の期限切れが再発することを防止できている。

また、保守委託先とは契約の不備を見直し、委託した業務においても同様に再発を防止する取り組みを行っている。今後のシステム開発や運用環境構築時の構築会社とのサーバ証明書などの組込み有無の確認については、次期システム構築までの見直し検討課題としている。

教訓

この事例では、運用中のサーバの証明書の期限切れを事例として、そこから根本原因を分析して広範囲の再発防止を検討した過程を説明した。サーバ証明書の期限の管理は、それが原因で大規模なシステム運用障害を引き起こした事例もあり、単純なようで意外に見落としやすい監視対象である。

ここでは、サーバ証明書の管理ができなかった原因をさ様々に考察し、それぞれに適した対策を検討した。その中で、証明書の期限が近いことをA社がトラブルになる前に認識できていなかった根本原因として特記すべき事項は、構築や保守を担当した委託先も、システムの保有者であり運用の最終責任者である A 社も、それがシステムに組み込まれているかどうかを運用開始時に確認できていなかったことである。

一般に、システム開発や運用環境の構築を委託する側に委託先と同レベルの技術が留保されていることは、大規模システムを自社で開発運用する組織以外には期待できない。今回の A 社の事例に限らず、大多数のシステム構築においては、システムを開発した側から運用を担当する側への情報伝達の漏れをなくすことが、このようなトラブルの発生を防止する最も有効な対策になる。しかしながら、構築時と保守時で委託先や再委託先が異なることや同じ委託先であっても担当 SE の変更が生ずることがままあり、その引継ぎの際に技術やノウハウの連携漏れが生ずるケースは少なくない。情報連携の漏れ防止は、システム構築や運用を受託する側の技術レベルの向上や内部チェックの強化が最も効果的な施策であるが、委託する側の自己防衛策は、以下の二点である。

- サーバ証明書の更新など運用継続に必要な作業が漏れなく報告されているかどうかを、運用引継ぎチェックリストに確認結果の記載を依頼する等により依頼元から積極的に確認することにより、システム構築委託元が知らないうちに証明書が組み込まれているというような事態の発生を抑止する
- サーバ証明書の有効期限のチェックや更新を漏れなく実施する（年間運用スケジュールに明示する）

この事例から教訓として伝えることは、「システム開発や環境構築を委託した際には、SSL サーバ証明書の有効期限の更新などのシステム運用開始後に定期的に実施すべき運用管理上のイベントの有無やその内容が漏れなく報告されているかどうかを、運用開始前に委託元から自主的に委託先に確認することを怠らない工夫をすること」とした。

3

技術領域の教訓

重要インフラ企業には、同じ企業内であっても、全く性格が異なるITサービスがある。

例えば、鉄道会社では、以下の2種類のITサービスは“一括り”で捉えるのは妥当ではなく、今回のコンテキスト調査の結果から2通りに分類し、教訓を整理・流通させることが妥当と考えられる。

- 列車の安全運用を司るもの（以降、「制御系」と分類）
  「変わらぬ使命である究極の安全」を実現するもの（JR東日本のATOSなど）
- 鉄道にかかわる付帯事業を司るもの（以降、「サービス系」と分類）
  座席予約、鉄道会社ICカード（JR東日本のSuica）など

今回の電力やガスのコンテキスト調査結果からは、「制御系」と「サービス系」による同様な分類が、下記例のように電力やガスでも有効であることが確認できた。

- 制御系：発送配電システムやガス供給システムなどが該当
- サービス系：料金システムなどが該当

また、サービス系は企業の基幹業務を司る「基幹業務系」とインターネットや携帯端末の普及にともない顧客が直接利用する「顧客サービス系」は障害発生時の影響度合いが異なるため分けて整理する。

なお、制御系とサービス系のコンテキストに依存しない共通的なシステムは共通系と整理する。

以上のコンテキスト分類に従い、今回得られた教訓は、各コンテキスト分類（教訓活用者）と関連付けて下図のように整理する。

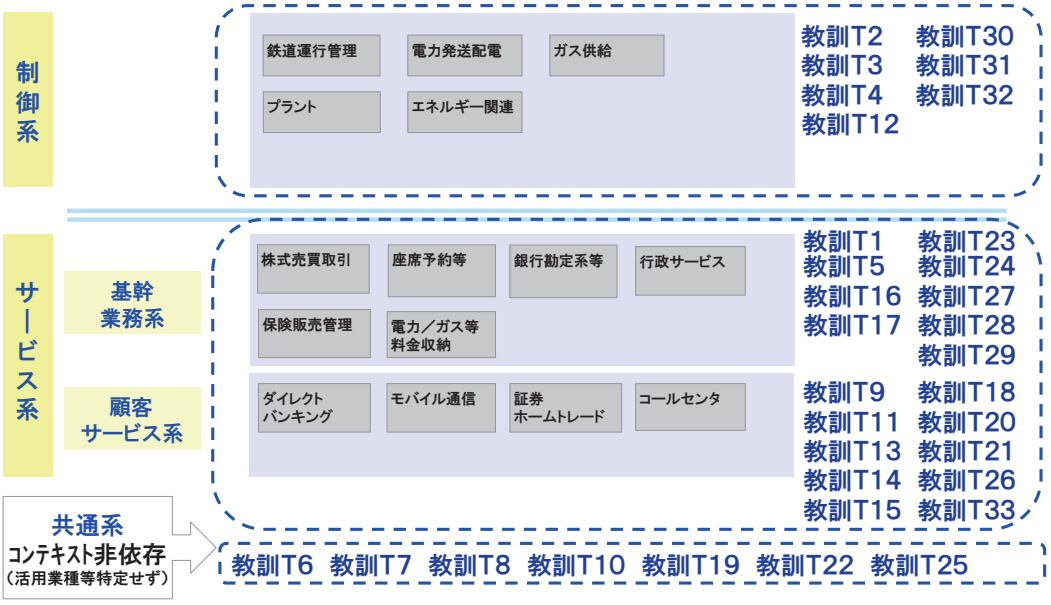


図3－1　技術領域の教訓と各コンテキスト分類（教訓活用者）との対応

以降の各節では、上図のコンテキストMAPに記された各コンテキストにおける教訓について解説する。

3.1　フェールソフトに関する教訓（T1）

教訓 T1　サービスの継続を優先するシステムにおいては、疑わしき構成要素を積極的にシステムから切り離せ（“フェールソフト”の考え方）

問題

サービスの継続を優先するデータの非同期送受信（メッセージ交換型）のオンラインシステムで、サーバの自動切替えが失敗し二つのノードが稼働する事象が発生した。

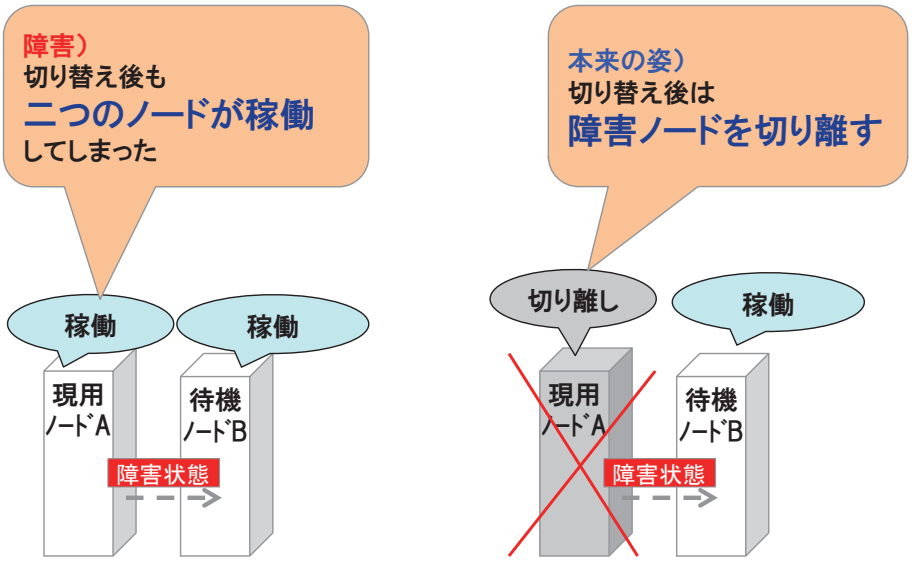


図 3.1－1　障害時のイメージ

原因

直接の原因はミドルウェア・OS の潜在バグである。根本原因は、それに対応してノード A 自身が自動停止しなかったことによるものである。

本システムはキューを使用したシステムであり、システム障害時にトランザクションが不完全に完了してもデータの整合性を保つように設計されている。このシステムにおいてノード A がノード B に『障害状態との誤ったメッセージ』を送り、ノード B も A の代わりに『稼働可能ステータス』に向けた遷移を開始した。そのため、ノード A も B も稼働状態となってシステム全体のステータスに不整合が発生した。

対策

応急対策としては、手動により障害となったノードＡを強制的に切り離した。

恒久対策としては、以下のようにフェールソフトの考え方を適用する。フェールソフトとは、システムの障害時の際にも、正常な部分だけで稼働を継続させることを重視した考え方である。

- **具体的な考え方**

  業務内容に基づいて、システムごとにポリシーを作成した上で、フェールソフトを適用する。ハードウェア機器の故障、ソフトウェアの処理プロセスの異常等があった場合には、その部位を積極的に停止させることでシステムから切り離す、場合によってはその系全体を放棄するといった考え方のもとに処理・対応する。

  一方、そのような状況下で一部の部位や系をシステムから切り離しても、システム全体としてのサービスは継続できるように、フェールソフトの考え方に基づいて設計・運用する。

- **この考え方に立つ理由**

  機器やソフトウェアそれぞれの動作継続を優先し過ぎてしまうと、予期せぬ障害の場合にサービスの影響がかえって大きくなってしまう場合があり、サービスの継続を優先させるためには、むしろ積極的に関連する部分をシステムから切り離す方が良い場合が多い。

  具体的には、ハードウェア、ミドルウェア、ソフトウェアにおいて、それぞれの機器やプロセスが自己診断等を基に個々に切替え・停止を判断する場合が多いが、予期せぬ障害においては、その動作が不安定となりコントロールできない状態になってしまうことがある。そのような場合には、周囲の機器やプロセスが、コントロールできない状態となっているおそれのある部位を強制的に切替え・停止させシステムから切り離すことにより、円滑に対応を行える仕組みを構築することで、サービス継続がより実現しやすくなる。

効果

フェールソフトを実装しておくことで、重大障害が発生しても業務が継続可能となり、取引の停止や報道発表につながるような重大トラブルが未然に防止できるようになった。

また、未知の障害が発生しても、システムを停止することなく業務の継続が可能となった。

教訓

サービスの継続を優先するシステムにおいては、疑わしき構成要素を積極的にシステムから切り離せ

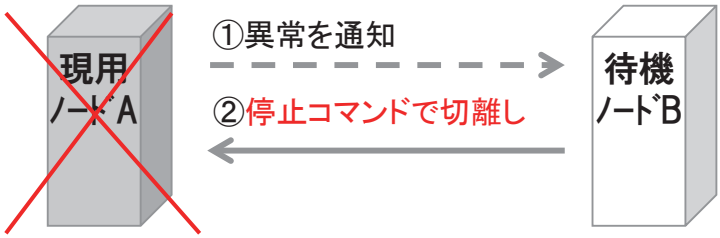
（サービスの継続と原因究明とのどちらを優先するかは、サービスの特徴に基づいてマネジメント層が決定すべきことである。上記対策は、比較的短いメッセージの１ラウンドの送受信によって完結するような、単純な処理において適用可能であり、適用に際しては十分な技術検討が必要である）。

【補足説明】対策の詳細

図 3.1－2 のようにフェールソフトの考えを取り入れ実装する。

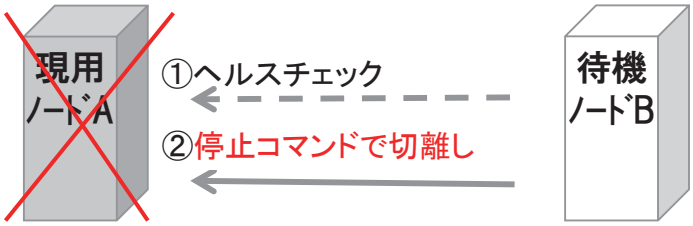
(1) 自身のヘルスチェックの場合

構成要素自身がヘルスチェックを行う場合、異常を検知して他系の構成要素にその旨の通知を行ったとき、通知を受けた系の構成要素は、念のため、通知を行った系の構成要素に対して停止コマンドを送る。



(2) 他系のヘルスチェックの場合

相互に他系の構成要素のヘルスチェックを行う場合、他系の構成要素の異常を検知して自身が現用系として動作するとき、検知した系の構成要素は、念のため、異常と判断した他系の構成要素に対して停止コマンドを送る。



(3) 自動停止できない場合は手動による停止で切り離し

自動停止できない場合のために、手動による停止を容易にするための仕組みを組み込んでおく。

図 3.1－2　フェールソフトによる切替え時のイメージ

3.2　システム全体を俯瞰した対策に関する教訓（T2）

教訓 T2　蟻の目だけでなく、システム全体を俯瞰する鳥の目で総合的な対策を行うべし！

問題

A社の制御系システムの下位システム（以下、下位）にある制御装置の稼働系に故障が発生した。自動的に待機系に切り替わるところが、切り替わらなかった。さらに、上位システム（以下、上位）の監視端末からの指示による系切替えを実施したが、切り替わらなかった（図3.2－1）。

一般的に制御系システムは、制御システムの監視・制御を行う上位と、それぞれが独立した制御装置で構成された下位にグループ分けされた構成になっている。

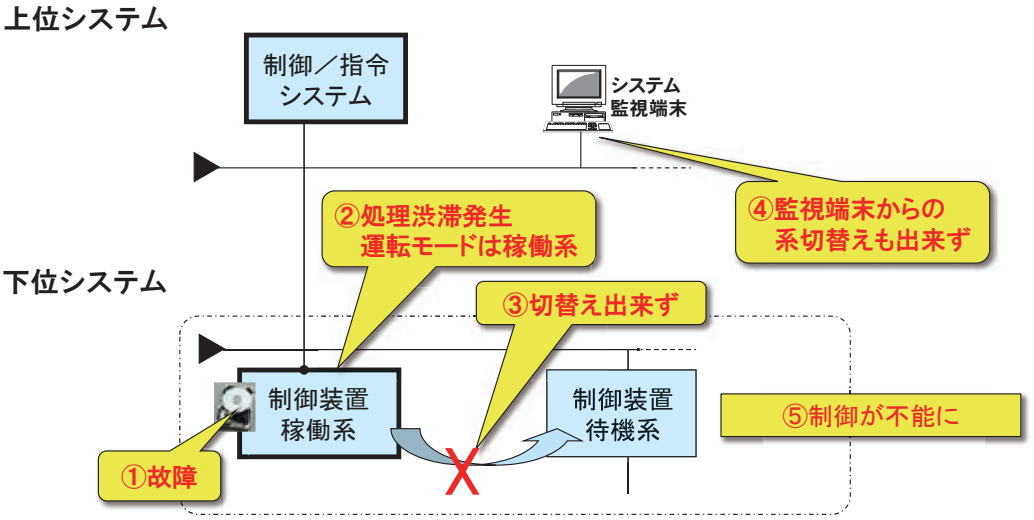


図3.2－1　システム全体図と障害発生状況

原因

直接の原因は、下位の制御装置の稼働系のハードディスクの機器故障である。通常初めに1回だけ行う「リセット通知」が故障のため出続けた。そのため処理が渋滞した。ところが稼働系は自分が処理中であると認識していたため、処理継続状態を続け、待機系への切替えは行われなかった。また、処理の渋滞のため上位との通信が途絶えたため、上位の制御監視端末からの系切替えもできなかった。

今回の事象を分析すると、以下のような根本原因があることがわかる。

【原因 1】稼働系が 1 回だけ行う「リセット通知」が出続けていることを、待機系は「稼働系の障害」と判断する機能が無かった。そのため、系の切替えが下位だけで完結せず、下位が、十分な自律機能（下位の中で障害対策を行う）を有していなかった。

【原因2】上位が、下位からの出続けている「リセット通知」を障害と判断する機能が考慮されていなかった。そのため、系の切替えなど、下位を制御することが十分できなかった。

制御系システムを正常に稼働させるためには、下位の中で自律した動きをまず実施することが重要だが、下位での障害が生じた場合は、上位が下位の監視・制御を行う必要があった。今回の障害は、そのいずれにも不具合があった。

対策

制御系システムの系切替えの対策として、以下の 3 つの対策を行った。

【対策 A-1】待機系装置からの停止制御機能を追加し確実に系切替えができるようにした（図 3.2－2 ①）。

【対策A-2】上位の制御装置から下位の制御装置の停止を確実にできるようにした（図 3.2－2 ②）。

【対策 B】　さらに、上位の制御装置の監視機能の強化を図った（図 3.2－2 ③）。

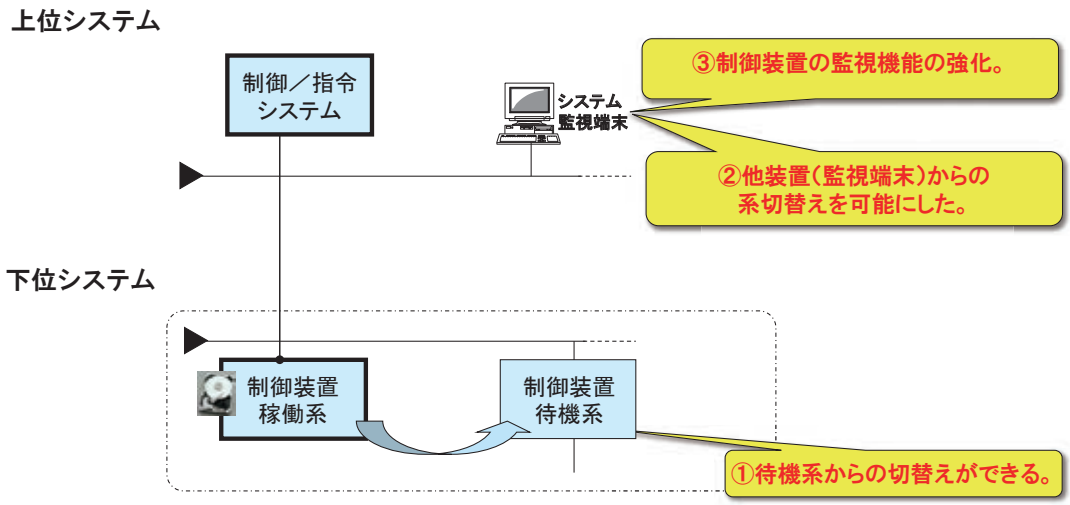


図 3.2－2　対策

この系切替えの基本的な考え方は以下の 3 項目に整理できる。

【対策A】システムで検知（上位、下位の制御装置）

【対策B】ソフトで検知（上位）

【対策C】手動による系切替え

今回の対策を当てはめると図 3.2－3 のようになる。

下位だけでの対策（対策 A－1）では、「蟻の目」対策であり、それだけでは制御系システムの対策として不十分である。そこで、「システム全体を俯瞰した鳥の目」対策として、上位からの対策（対策 A－2、対策 B）を行うことにより、システム全体の信頼性が向上する。また、対策の順序としては、対策 A から対策 C に向かっていくほど障害復旧の時間が長くなるので、対策 A から順番に行った。

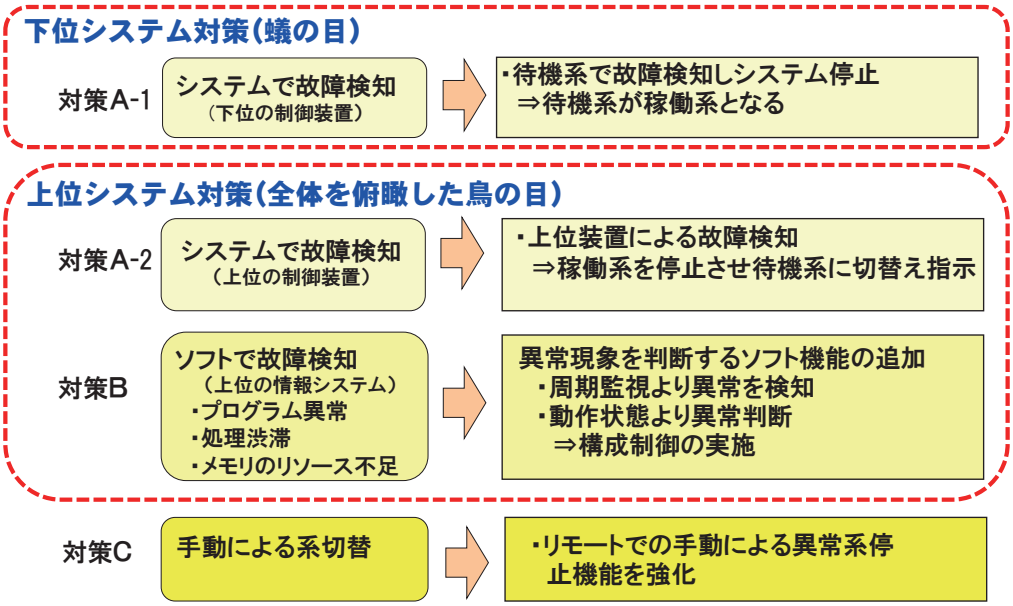


図 3.2－3　信頼性向上のための系切替えの対策

効果

障害対策は、障害を起こした下位の制御装置だけの対策を考えがちであるが、蟻の目の対策だけでなく、システム全体を俯瞰する鳥の目対策を活用することで障害発生時の復旧時間の短縮が可能であり、システムの安定稼働（障害発生頻度の減少）が保たれる。

例えば、停電、列車運行停止などが起こらなくなり、復旧が早まる。今回の事例では、2 ～ 3 時間かかっていたのが、瞬時に復旧することができた。

教訓

上位システムと下位システムとで構成されている制御系システムの障害対策は、障害の発生した制御機器の対策（蟻の目）だけでなく、システム全体を俯瞰する鳥の目で総合的な対策を行うことが重要である。

3.3 テストパターンの整備に関する教訓（T3）

教訓 T3

現場をよく知り、現場の知識を集約し、現場の動きをシミュレートできるようにすべし！

問題

ある鉄道会社の制御系システムの事例である。駅に出入りする列車の運転は、駅に設置してある列車制御システムにより制御が行われている。折返し運転のある駅で、A 列車が折返しホームに入り、乗客の乗降後、出発時間となったので、元来た方向に出発していった。同じホームを使用する B 列車が A 列車の出発後、反対方向から近づいて来た。しかし、A 列車が出発していったにも関わらず、ホーム手前の信号機が B 列車の「進入」を表示しなかったため、B 列車は、ブレーキが自動で動作し駅の手前で停止してしまった（図 3.3－1）。

原因

直接の原因は、列車制御システムのソフトウェアのバグである。先行の A 列車の出発完了にも関わらず、A 列車に対する制御信号が出続けてしまったため、続行の B 列車がホームに接近したにも関わらず、B 列車の制御信号の表示ができなかったためである。

制御系システムは、一般に、制御対象の様々な「処理の動きとタイミング」をすべて捉え、その動きすべてに明確な制御・指示（コントロール）ができなければならない。

このような列車の動きを想定した本番と同じようなテスト環境を準備することは、困難であるため、システムの本稼働が開始する前に実列車を用いてテストを行ってはいたが、今回の事象のようなケースのテストは行っていなかった。

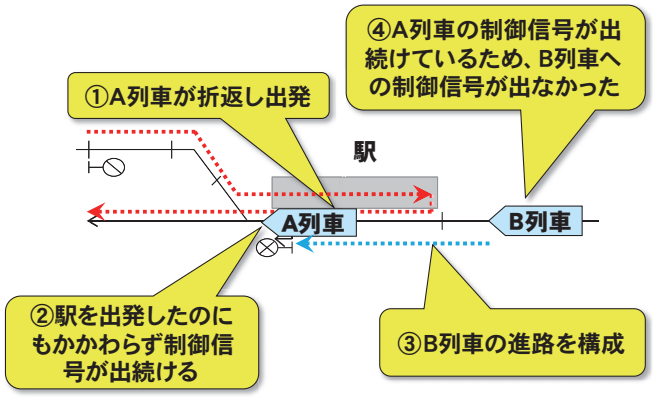


図 3.3－1　駅での障害発生状況

今回の事象を分析すると、根本原因は以下であることが分かった。

【原因 1】有識者（ベテラン社員を含む）により、以下のとおり制御信号の機能確認を行っても、まだ洗い出せていない機能が存在する。（機能要件漏れ）

- 有識者（ベテラン社員を含む）による要件確認。
  機能要件作成時にあらゆるパターンの洗い出しを有識者に実施。
- 有識者による要件&テスト仕様レビュー。
- 有識者による思いもよらない入力テストを実施。（意地悪テストの実施）。

【原因2】列車の動き、信号システムの動作などを総合的にテストできる環境、つまり、組込みソフトウェアを持った制御システムと列車などの動作のすべてのテストが行えるテスト環境ができていない。

対策

直接の対策として、駅の列車制御システムの制御信号送信プログラムを改修した。

今回のように、列車の運転を考慮したテストを行わなかったことにより、重大な列車運転阻害を招く可能性がある。まさしく、「現場」を熟知することが、「安全、信頼」を堅牢なものとする条件である。

【原因 1】については、従来から行ってきていた作業に加えて、一度設計された「機器の動き（列車の運転）」のパターンを知識データベースとして蓄積し、そこに、さらに新しい動き、機能漏れの動きを追加登録していく。これにより、すべての「機器の動き」のパターンを知識データベース上で把握することが可能になり、暗黙知が形式知化される。

【原因 2】については、本番環境と全く同じテスト環境を用意し、すべてのタイミングの問題点を実機や操作員で確認することは不可能である。そこで【原因 1】の制御装置の動きのパターンの知識データベース化が進めば、シミュレーション・システムの強化が可能になる。

制御系システムのシミュレーション・システムの開発を行うためには、現実の制御装置のプロセスを分かりやすく可視化し、プロセスの骨子を見極めてモデリングする。特に、微妙なタイミングを問題にするテストは、実機で再現することは難しいが、シミュレーションでは容易に再現することが可能である。

【補足説明】知識データベース

知識データベースとは、ここでは過去の制御装置、移動装置、操作員の「動き」のパターンを記録することを指す。この制御系システムに新しい要件を追加する場合、この知識データベースの記録や、有識者からの新たな知識を参照すると、設計時の考慮漏れやリスクの洗い出しができることを目指したものである。

効果

以下の効果を得ることができる。

- 知識データベースにより、機能漏れを防ぐことができる。
- シミュレーションにより、特殊ケースでの障害を防ぐことができる。

これにより、堅牢な「安全、信頼」の制御系システムを構築でき、事例で起きた社会インフラの混乱回避が期待される。

今回の事例は鉄道についてであるが、このような制御系システムは、工場における生産ライン制御、電力の供給ライン管理制御、通信の交換機制御など、世の中に数多く存在している。多くの制御システムにおいても、今回の列車の運転にあたる被制御機器の動作について熟知し、その動作に対するテスト項目をたてテストするだけでなく、シミュレータを使用したテストも行うことは、制御システムの品質を高める上で欠くべからざるものである。まさしく、「現場」を熟知することが、「安全、信頼」を堅牢なものとする制御システムの条件である。

教訓

現場をよく知り、現場の知識を集約し、現場の動きをシミュレートできるようにする、そのためには、機能要件の蓄積と実地テストができないためのシミュレータテストを考慮することによって、障害を予防することが重要である。

3.4 システム環境の変化への対応に関する教訓（その 1）（T4）

教訓 T4

システムに影響する変化点を明確にし、その管理ルールを策定せよ！

問題

制御系システムは、監視・制御を行う上位システム（以下、上位）とそれぞれが独立した制御装置・移動装置を持つ下位システム（以下、下位）とで役割分担をしている。

列車制御システムも同様な構成になっており、列車の運転は上位の指令センターと呼ばれる所で集中監視を行っている。さらに、事故などがあって列車が乱れた場合に対処するために、数時間先までの運転がどのようになるかの予測をシステムで行い、その結果を予測ダイヤとして画面に表示している。

ある日、雪によるポイント不具合が早朝に複数の駅で発生した。駅間停車防止のため複数列車に列車抑止（列車に対して駅にとどまるよう指示）を入力したところ、画面表示がすべて消えてしまい、予測ダイヤを見ることができなくなった（図 3.4－1）。

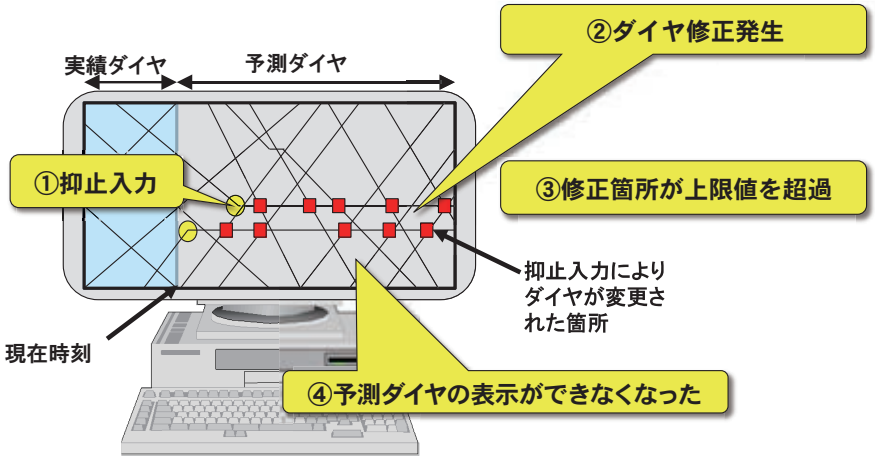


図 3.4－1　障害発生状況とモニター画面

原因

直接原因は、駅間に列車が停車するのを防止するため、複数列車に対し「列車抑止」入力を行ったことである。「抑止」入力を行うと、対象となる列車ごとに予測ダイヤの修正個所がシステムの画面に表示されることになっている。この入力が早朝であったため、変更入力に基づく予測ダイヤ上の「修正

個所」がほぼ 1 日分発生した結果、システムの上限値（修正個所：600 件）を超えてしまい、予測ダイヤを表示できなくなった。（上限値を超えた場合には画面を消すという仕様になっていた）

この上限値はシステム構築当初から決まっていたものであった。

システムは、長期間の使用にともない変化が生ずる。その変化は、新サービス開始や法改正等のようにある一点で起こるものと、利用者の増減や装置の劣化のように徐々に緩やかに変わっていくものある。そのような変化によってシステムの要件（ここでは上限値）変更が必要となる要因を変化点という。また、変化点管理とは、変化点を監視することによって未然に対策を取る管理のことである。

根本原因は、システムに大きな変化点があったにも関わらず、それを見逃していたことである。具体的には、以下のように、変化点の見逃しが 3 つ存在していた。

【原因 1】予測時間の延長

列車ダイヤの予測時間を 4 時間先から 24 時間先までに変更した際、「修正個所数」の上限値の増加などシステム全体の機能要件変更を行わなかった。

【原因 2】列車本数の増加

列車の本数が年々増加しており、本来ならば（運転本数の増加の都度、）上限値を超えた際のシステムの動作を見直す必要があったにも関わらず、行わなかった。

【原因 3】「抑止」機能の使い方の変化

システム導入当初、「抑止」機能は、指定駅区間に指示を行う「駅抑止」を使っていた。その後、今回の障害のトリガである「列車抑止」が、年月とともに徐々に、かつ優先的に使われ出した。「列車抑止」は「駅抑止」に比べ、予測ダイヤの「修正個所数」が多い。このような傾向にも関わらず、「抑止」機能の使い方の変化を監視し、それにともなう「修正個所数」への影響を確認することを怠っていた。

まとめると、根本原因は、全体に影響する変化点（この場合、予測時間、列車運転本数、使い方の変化）が管理されていなかったことである。

このように、予測時間、列車運転本数といったハード的（物理面）な観点だけでなく、使い方の変化（利用形態の変更）といったソフト的（運用面）な観点も含めた変化点を監視し、そこから、システム障害に結びつく上限値超えを事前に察知し、対策を打つことが必要であった（図 3.4－2）。

事例では、上限値超えを取り上げているが、システム的には、リソース不足、性能不足などの非機能要件に影響する変化点を監視することといえる。

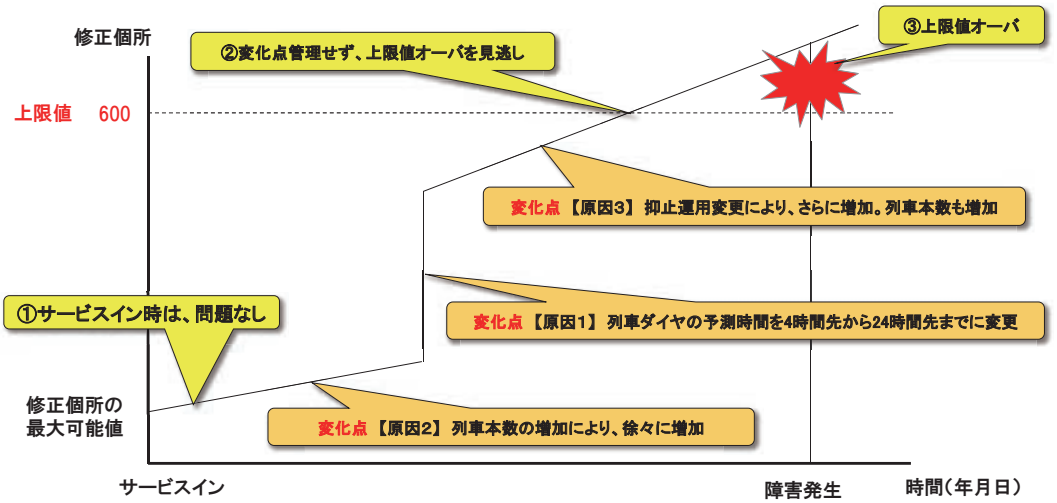


図 3.4－2　上限値超えに至る「修正個所の最大可能値」の変化

対策

予測ダイヤの処理（1 分おき）のたびに修正個所のクリアを行うとともに、修正個所 600 件までは予測ダイヤの演算処理を継続し、予測ダイヤを描画するようにプログラムを改修した。

また、システムの変化点を管理するルールを設け、要件の変更につながるような案件に対しては、システムの見直しを行うようにした。

過去においては、制御系システムは、上位と下位とで役割分担をしているため、上位が変更されても、下位（それぞれ独立した制御装置・移動装置）には影響がなかった。しかし、上位によって下位が管理されるようなソフトウェア構成になってきている現在ではシステム全体での変化点の管理が必要になる。

制御系システムでの変化点は、一般的なシステムの仕様変更の他に、運用形態の変化、対象時間、対象機器の動き、機器数の変化なども含む。障害を事前に防止するためには、この変化点を見逃さない仕組みを構築することがポイントになる。

特に、機能の拡張（制限値に影響する場合）時は、変化点を見逃してはならない。この事例のようにアプリケーションプログラムの中でテーブル内のデータ件数の制限を持っているパターンは非常に多い。制限値に対する上記のような変化点が管理指標のひとつである。

今回の障害では、ある開発時に決定した上限値が、ユーザの取扱い方や輸送形態等の変化、システムの改良などでシステムにかかわる諸元の変動により上限値を超えることがないよう変化点を適切に管理することになる。

今回の問題については、以下の 3 つの変化点管理を行った。

- 上位における仕様変更（予測時間の変更等）　→　【原因 1】の対応
- 下位における列車制御装置の変更（列車本数の増加等）　→　【原因 2】の対応
- （オペレーション、運用）項目表を作成し、使い方の変化を常時チェック　→　【原因 3】の対応

制御系システムの変化点管理ルールを明確にし、そのルールを守る仕組みを構築するために、次の 4 点を行う。

- システムが監視・制御する対象と仕様の変化点を洗い出し、管理項目とする。
- 変化点管理のルールとそれを守る仕組みを構築する。
- 上限／下限値の設定された管理項目の物理的な変化、特に重要データの「見える化」を検討し、モニタリング機能の強化を行う。
- 変化点管理で使用する管理指標を関係部門で共有し、また定期的な会議でその管理指標を確認しあい、「変化点の見落とし」を防ぐ。

これらの対策により、現場環境の変化、システム要件に変化があった時点を可視化することができる。

効果

制御系システムの変化点管理を行うことにより、システム要件の変更の時期が明確になる。その結果、システムの更新を行うことにより、システム障害を防ぐことができる。

制御系システムは、列車運転に限らず、工場における生産ライン制御、電力の供給ライン管理制御、通信の交換機制御など、世の中に数多く存在している。これらのシステムにおいても変化点管理を応用し、社会インフラの混乱を防ぐことができる。

教訓

制御システムのような長期的に使うシステムは、様々な内的（運用の変化）、外的要因（機器などの物理的増加）から、当初設定した上限値や機能の使い方に大きな変化が起きていることがある。また、新サービス開始や法改正等のような一点の変化点だけでなく、利用者の増減等長期的な変化でとらえなければならない変化点もある。

それらの変化点を見逃すと、後日重大な障害を起こすことになる。システム全体に影響する変化点を明確にし、その管理ルールに基づいた変化点管理を行うことが、システムの改修を促す要件変更のときを逃さず、システム障害を未然に防ぐ対策である。

3.5 サービス視点での変更管理に関する教訓（T5）

教訓 T5

サービスの視点で、「変更管理」の仕組み作りと「品質管理責任」の明確化を！

問題

サービス系システムの中には、本店ホスト、事業所サーバ、営業端末などの機器を接続したシステムが多い。

A社の使用料請求システムは、本店ホスト／サーバから請求データを営業員が持ち歩くHT（ハンディ・ターミナル）に転送するシステムである（図3.5－1）。

そのシステムから出される請求書の金額が誤ってお客様に渡ってしまい、A社は個別謝罪・請求書の再発行に追われることになった。

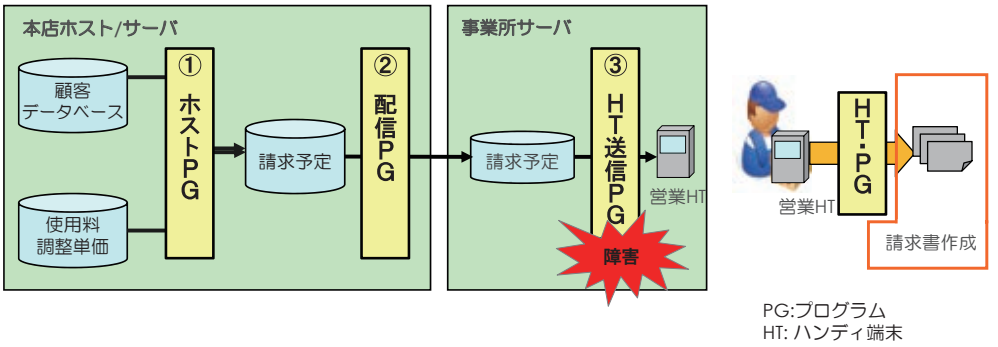


図3.5－1　使用料請求システム

原因

直接の原因は、事業所サーバの③「HT送信プログラム」において、使用料請求データの符号判定処理で、調整額を減算すべきところを加算してしまったことによる。本店ホスト／サーバ上では、正しく減算ができていた。

さらに、障害の原因を分析したところ、以下の2点が判明した。

【原因1】「使用料請求データの符号判定処理で、調整額を減算する」機能は、初期は本店ホスト／サーバ上の①、②では考慮されていたが、当時の状況では、減算となる状況が発生していなかったため、減算を考慮したテストが不十分であった。今回、初めて調整額の減算が発生した。

【原因 2】HT はシステム稼働後、数年が経過した後に新規要件として追加された機能であった。また、インフラ機能であったため、アプリケーション開発部門は参加せず、システム部門だけで更改作業をおこなった。そのため、機能変更に対する影響の認識が甘かった。

この事例のようなサービス系システムのライフサイクルは長い。その間に、途中で新たな要件が追加されたり【原因 2】、使用方法が変わったり【原因 1】する。そのために今まで正常に稼働していたシステムに、突如障害が発生する場合がある。

それを防ぐには、常にエンドユーザへのサービスの視点で「何が変わったのか」をチェックすることである。この事例では、障害を未然に防ぐポイントが 2 カ所あった。先ずは、【原因 2】の HT を導入することによりサービスが変更になったことと、次に、【原因 1】の今まで調整額は、プラスしか発生していなかったが、今回初めてマイナスが発生し、やはりサービスが変更されたことである。つまり、根本原因としては、システムをサービスの視点で見渡した変更管理ができていなかったことである。サービスの視点で見渡した変更管理ができていれば、障害を未然に防ぐことができたと考える。

さらに今回の事例で、「本店ホスト／サーバ→事業所サーバ→ HT」と経由していく途中で、以下の問題が生ずる可能性が考えられる。

① データ整合が取れない可能性がある。
② プログラム仕様、実装、それぞれ整合が取れない可能性がある。
③ テスト仕様・実施整合が取れない可能性がある。

対策

このシステム障害を抑えるためには、サービスの視点での変更点を見落とさない仕組みを作る。つまり、変更があったときに、システム全体のプログラムの整合性、データの整合性、テスト仕様の整合性を保つための変更管理を以下の手順で行う（図 3.5－2）。

① 機能要件は当初からあったとしても、長い間に変わっている場合がある。エンドユーザにとって、初めて使用する機能であれば、新規の要件とみなして対応する。
② 現状分析をおこない現行システムの仕様を確認し、機能変更に対する「要件定義書・設計書・テスト仕様書、プログラム」の影響調査を実施する。
③ 上記①、②の変更を含めたシステム全体のテストを実施する。
④ 修正漏れがないように、「要件定義書・設計書・テスト仕様書、プログラム」の整合性を管理する。
⑤ システム全体のデータ整合性、プログラム整合性、テスト仕様整合性を確認する人を決め、品質管理責任を明確にし、開発フェーズごとの検証（レビュー等）を行う。

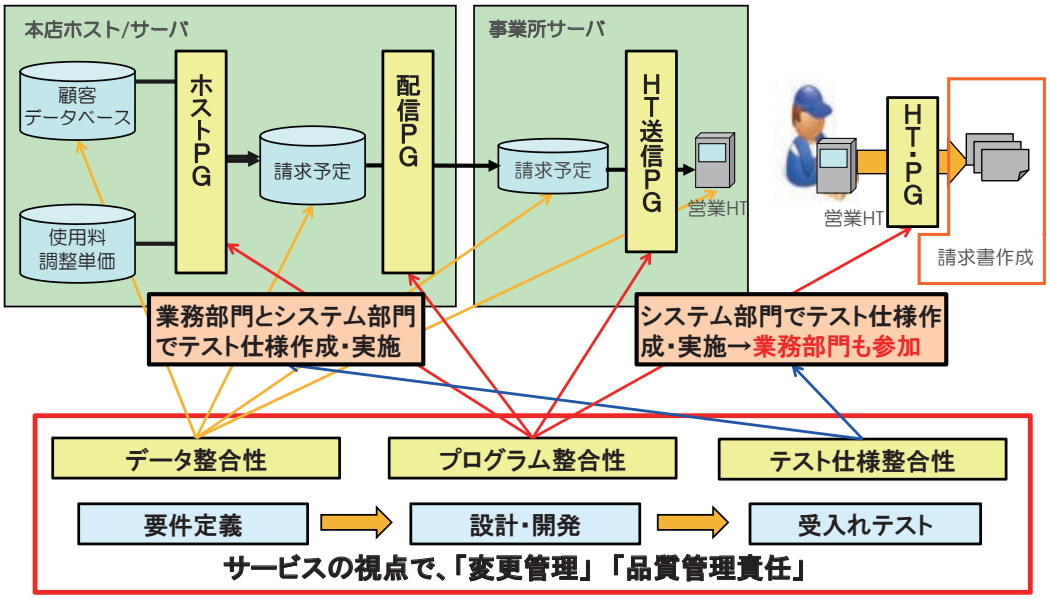


図 3.5－2　システムの概要と障害への対策

効果

変更管理の仕組み作りと開発工程ごとの確認体制を作ることにより、システムの品質を保つことができる。

教訓

サービスの視点で、「変更管理」の仕組み作りと「品質管理責任」の明確化を行うことが重要である。サービス開始時に入れた機能でも、その機能を使用していない状態が長く続くと、いざ使うときに、思わぬ障害を引き起こすことに注意すべきである。

3.6 本番環境とテスト環境の差異に関する教訓（T6）

教訓 T6

テスト環境と本番環境の差異を体系的に整理し、障害のリスク対策を練る

問題

サービス系システムで本番保守作業（商用作業）を行うとき、テスト環境での事前確認のときは問題が生じなかったが、本番環境では障害となることがある。

近年の傾向として、サービス系システムの保守作業が以下のように難しくなっている。

- サービス時間の拡大にともない、オンライン稼働中に保守作業、システム変更を行わざるを得なくなっている。また、そのような作業も、限られた時間内で行わざるを得ない。
- 事前に検証を行うテスト環境には様々な制約（リソース、コスト等）があるため、本番環境と同一の環境を構築することが困難である。

A社では、サービス系システムのリプレースにともなうデータ移行準備作業として、データベースのパラメータ変更作業を計画した。サービス停止時間を極力短くするため、その変更作業の事前準備をサービス提供中（オンライン稼働中）に実施することとした。本番環境の保守作業実施前にテスト環境で作業を実施し確認したところ問題は発生しなかった。そこで、本番環境で実施したが、システムダウンが発生し、サービスが停止した。

原因

テスト環境と本番環境とに相違があり、テスト環境でうまくいったソフトウェアのリリースが、本番環境で障害となった。

本番環境には、テスト環境にないデータベース・オプションを導入（コスト抑制のため、テスト環境では未導入）していた。

今回の障害の直接原因は、データベース・オプション使用環境において、オンライン実施時にのみ発生するデータベース・パッケージのバグであった。しかし、製品マニュアルには、注意事項の記載がなかった。また、作業実施日の1カ月前に「既知のバグ」として製品ベンダの技術情報データベースに掲載されていたが、見逃していた。

根本原因は、テスト環境と本番環境の差異が明確になっていなかったため、事前テストにおける環境差異の影響が十分に把握できていなかったことにある。そのため、十分な対策が取られていなかった。

対策

テスト環境と本番環境の構成（ハードウェア、ソフトウェアや個々のソフトウェアのバージョン、パラメータなど）を極力同一にすべきであるが、合わせられない場合には、以下の対策を実施する（図 3.6－1）。

① テスト環境と本番環境の差異を明確にする。特に、「ソフトウェア製品管理」、「ハードウェア製品管理」、「アプリケーション管理」など重要項目をすべて洗い出した「差異分析」を行う。

② テスト仕様において、事前にテスト環境で確認できない項目、機能が存在する場合、事業部門／システム部門開発担当／システム部門運用担当の 3 者で、リスク分析を行う。

③ そのリスク分析結果を基にそのリスクの度合いに応じて、リスク対策やコンティンジェンシプランを立案して、事業部門、または経営トップへリスクを伝える。

④ 本番環境の保守作業のリスクをステークホルダ（経営トップ、製品ベンダを含め）で共有する体制を作る。

⑤ 大きいリスクは、経営トップが判断する。

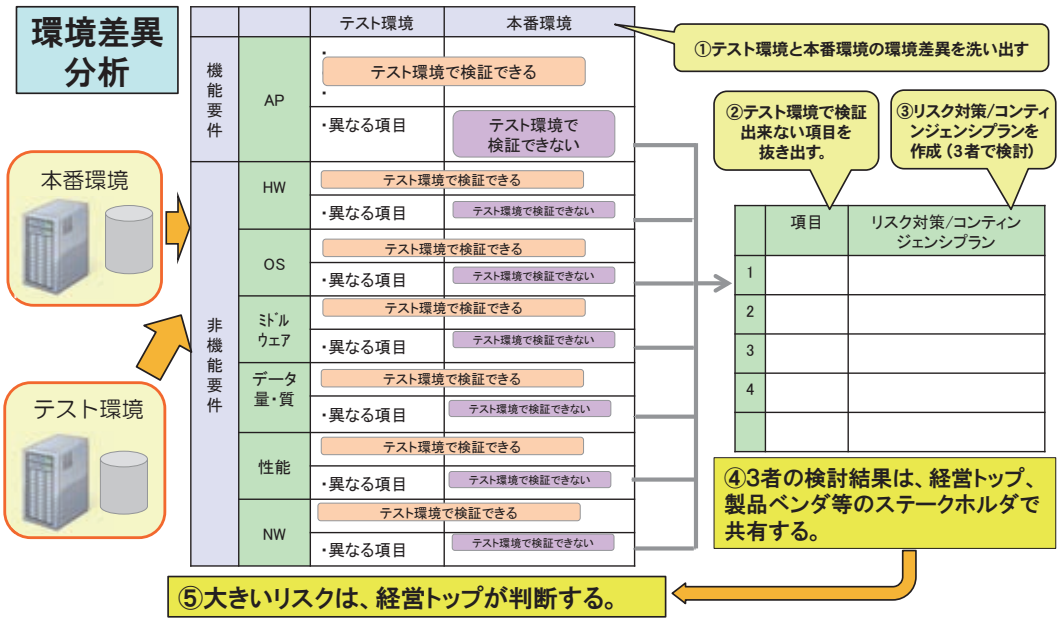


図 3.6－1　環境差異分析

また、今回のような事例の場合、製品ベンダからの定常的なパッチ情報の入手とパッチ適用サポート支援を受けることも重要である。製品ベンダとの支援体制を作ることが対策として有効である。

効果

以下の効果により、本番保守作業でのシステム障害を減らすことができる。

- 環境差異分析により、本番保守作業時のリスクの存在個所が明確になり、障害発生時の原因の絞り込みが行いやすくなる。また、環境差異分析の一覧をチェック表として活用し、環境変更時、システムリリース時等の確認に使用することができる。
- 経営トップも含めたリスク対策を講ずることにより、全社でのリスク体制が取れる。

教訓

テスト環境と本番環境の差異を明確にし、事前テストにおける環境差異の影響を十分に把握できるようにする。さらに、障害のリスク対策が立てられることが、障害の予防になる。

3.7 バックアップ切替え失敗に関する教訓（T7）

教訓 T7　バックアップ切替えが失敗する場合を考慮すべし

問題

冗長化構成を取っていても、障害時、バックアップ切替えが正常に機能しなかったり、障害機器の切離しによる縮退運転が正常に機能しなかったりと、システム稼働の継続ができない事例が後を絶たない。

A社の基幹システムは、デュプレックスシステムでのホットスタンバイ構成をとっている。稼働系システムのハードウェア障害が発生し稼働系がダウンした（図3.7－1①）。障害検知にともない自動的にバックアップ切替え処理が駆動され（図3.7－1②）、待機系システムを稼働させようとしたが、待機系が立ち上がった後のオンライン処理が障害となり、待機系もダウンした（図3.7－1③）。このとき、現場が混乱し、後の対処方法の決定まで多くの時間を要した。最終的に、稼働系のハードウェア故障の部品を交換し、再度、稼働系を立ち上げて処理を再開させた。

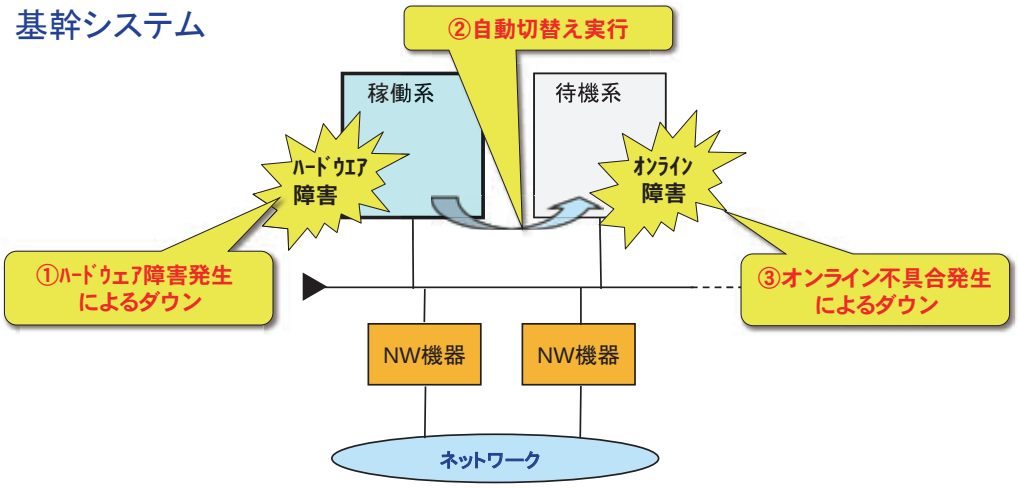


図3.7－1　障害状況

原因

バックアップ切替え時障害の直接原因は、以下の 2 点であることが分かった。

【原因 1】切替え失敗の直接原因

以前、障害が起きた際の緊急対応時に、稼働系へのソフトウェアのパラメータ設定変更を行った後、同様に待機系にもこの対応をする必要があったにも関わらず、これを怠った。さらに、稼働系と待機系の同期を取るべきソフトウェアパラメータと、それぞれの系で独自に設定するソフトウェアパラメータがあるが、それらの管理を怠った。

【原因2】復旧時失敗の直接原因

稼働系のパラメータを基に、待機系のパラメータを最新化しようとしたが、稼働系と待機系で独自に設定するパラメータが分からなかったため、修正を素早く実施することができず、待機系からのオンライン再立ち上げをあきらめざるを得なかった。稼働系のハードウェア復旧を行ってから、稼働系で再立ち上げを行ったため、復旧に大幅な遅れが出てしまった。

稼働系システムは変化するため、それに合わせて待機系システムも同期を取る必要があるが、日常の運用で検証していく仕組み（切替え実施、同期チェック、手動切替え手順書の更新等）を作らないと待機系システムは取り残されていく。

根本原因は、待機系システムを本番運用の重要な機能であるとの観点が不足しているため、日常の運用で待機系システムの検証が十分行われていないために起きている。

対策

バックアップ切替えは、本番運用であることを認識し、稼働系と同じ運用を待機系でも行うことを考えた運用計画、リスク対策を立てる。

この事例を通して、以下の対策を立てた。

- バックアップ切替え対策　（原因 1 →対策 1、2）
- 切替え失敗時の復旧対策　（原因 2 →対策 3、4）

＜対策 1＞通常保守運用において、稼働系、待機系のソフトウェアパラメータの確認、プログラムバージョン管理を徹底する。

通常運用のプロセスの中で、冗長構成を定義したソフトウェアパラメータに矛盾がないことを確認する。チェックプログラムを作成し、日常バッチ処理で、稼働系、待機系の構成定義、各サーバの構成定義のチェックを行う。さらに、システムが正しく動作するかどうか、実機のテストを行う。切替えが成功したことを確認するだけでなく、待機系でも業務が正常に稼働することまで確認する。そのため、確認事項／チェック項目（サービスはすべて稼働したか、すべての接続端末は稼働するか、等の動作確認）を明確にする。

＜対策 2＞定期的にサービス停止時間帯を設け、障害訓練を行う。

バックアップ切替えの運用を理解するために、待機系への切替えの障害訓練を行う。

なお、障害訓練で、待機系に切り替えたために本番処理が稼働してしまい、システム障害を引き起こす事例が過去にあった。この事例では、業務処理を稼働系、待機系でそれぞれ実行してしまい、二重処理になった。本番環境で実施するので、事前準備（本番環境のデータ保存、手順書の作成、訓練終了後の戻し手順、確認手順等）をしっかり行うことが必要である。

＜対策 3＞切替え失敗を想定し、復旧のための手順を明確にする。

待機系への切替えができなかったときを考え、手動で障害から復旧する場合（復旧は、バックアップ切替え方法も含めた処理継続の確立を言う）も考慮し、様々なシナリオ（目標所要時間を含む）を想定した手順書を作成しておく。また、障害復旧テストを行い、各シナリオについての所要時間を計測し、手順書の確認を行う。

＜対策 4＞障害復旧訓練を行い、実際に使える手順書を作成しておく。

障害復旧訓練を行い、シナリオに定めた時間内に復旧できるかどうかを確認する。また、実際の人の動きや判断基準等を考慮して、手順書に反映する。

効果

「バックアップ切替え失敗」になる事例を理解し、対策を実施することにより、バックアップ切替えが失敗し、障害の復旧に多大な時間を要するリスクを減らすことが期待できる。

教訓

バックアップ切替えが失敗する場合を考慮し、設計時、運用保守時、予防対策を行うことが重要である。

ここで示した障害事例に加えて、過去に起きたバックアップ切替えにかかわる障害事例を調査し、分類したところ、11 パターンあることが分かった。

“4.2　バックアップ切替え失敗の問題と対策（詳細説明）”では、それらの事例を発生原因の主な要因である「切替え失敗」、「性能不足」、「切替え無効」、「ネットワークの切替え失敗」、「設備の切替え失敗」を問題と対策として説明している。この資料を利用して、発生した障害と同じ問題に対応する対策を実施することで、再発を防止することができる。また、切替処理に関する対策を網羅的に確認・実施することができる。

3.8 仮想化時の運用管理に関する教訓（T8）

教訓 T8　仮想サーバになっても リソース管理、性能監視は運用の要である

問題

A 社システム部門は、プライベートクラウド（仮想サーバホスティング）の運用を行っている。クラウド環境の共有ストレージは 2 つの論理ボリュームから成り、それぞれ仮想サーバグループを構成している（図 3.8－1）。

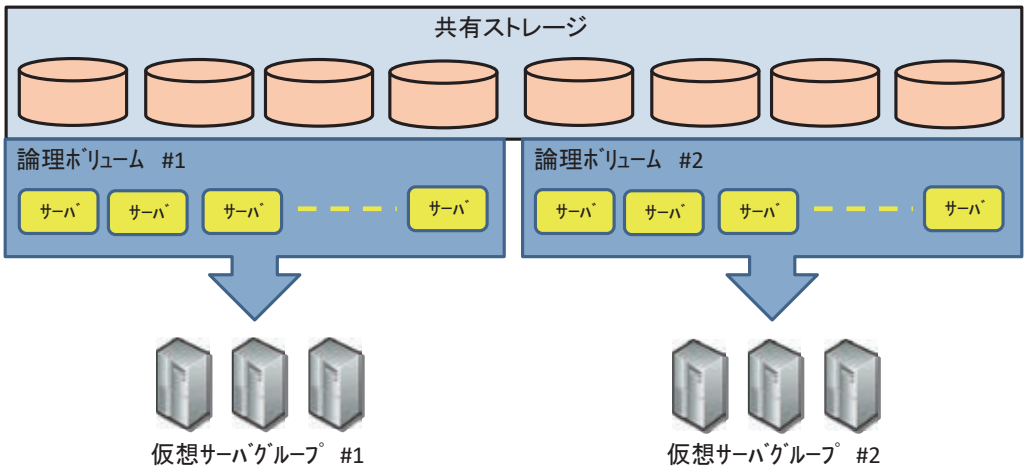


図 3.8－1　構成図

突然、論理ボリューム #1 の容量の空きが無くなり、その仮想サーバグループ #1 の全サーバの動作が不安定となった（図 3.8－2 ①）。そこで、マウントしているサーバで不要なサーバを削除したところ（図 3.8－2 ②）、一時的に回復した。しかし、削除したサーバのスナップショットが大量に出力されたため、また空き容量が不足してしまい、再度不安定になった（図 3.8－2 ③）。そこで、さらに不要サーバの削除とスナップショットの世代数を削減し論理ボリュームの空きを確保したところ、障害発生から 6 時間後、正常に向かった（図 3.8－2 ④）。

その後、仮想サーバグループ #1 で削除したサーバの運用要件の確認を行い、仮想サーバグループ #2 に削除したサーバを移したことで最終的に全業務正常となった。その間、削除されたサーバ上の 3 業務が、24 時間に亘り正常に稼働することができなかった（図 3.8－2 ⑤）。

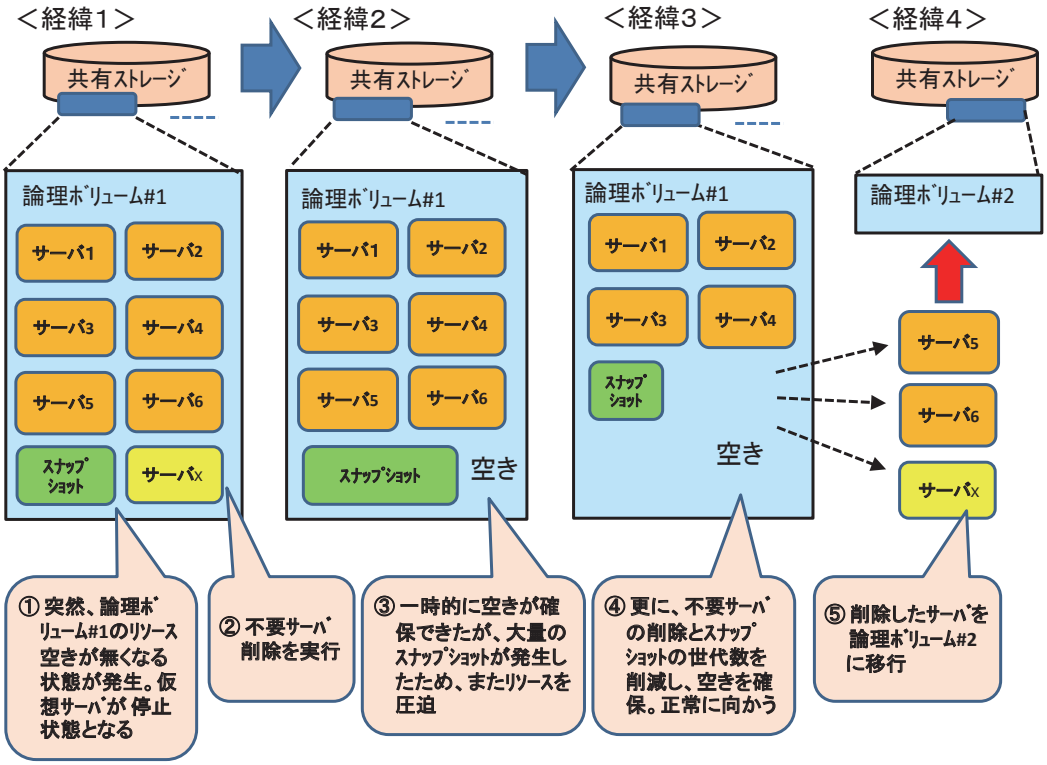


図 3.8－2　障害発生時の論理ボリューム状況

原因

直接の原因は、システム部門の運用担当者が、物理サーバを仮想サーバに移行する過程で、物理サーバ X を論理ボリューム #2 に割り当てなければならないのを、誤って論理ボリューム #1 に割り当ててしまったことにあった。またサーバの停止状態を復旧するため、割り当てたサーバを削除したが、そのサーバのスナップショットを共有ストレージ上に定義しており、また大量に出力されることを認識していなかったため、再度急激に論理ボリュームの容量が不足し仮想サーバグループ #1 が不安定になってしまった。

さらに、運用担当者は、リソース監視をきちんと行っていなかった。クラウドにサーバを集約したことにより急激にサーバ数が増え、ストレージリソース（論理ボリューム #1）を圧迫していたことに気づかずにいた。そのために、十分な容量を確保しない状況でサーバ移行を行い、作業ミスにより容量不足を引き起こしてしまった。

また、運用担当者は、従来の IT 管理業務の経験は長いが、新技術となる仮想サーバの管理は初めてであったため、経験不足、教育不足により、障害を素早く復旧させることができなかった。

根本原因は、システム部門が、仮想化に移行しても、運用の要であるリソース管理や性能監視が重要であることを理解していなかったことによる。

特に障害事例（図 3.8－2 ⑤）で示したように、障害が発生したとき、仮想化サーバグループ間での物理サーバの入替えが生じたため、復旧作業に大幅な時間が必要となってしまった。これは、システム部門が、業務部門から出される、機能要件、非機能要件を運用要件として設計せずに、仮想サーバへの移行を進めたことにより生じたリソース不足が原因である。

このように、各サーバの運用要件を整理せず仮想サーバに移行すると、リソース不足や、性能不足を起こし、仮想化の効果が上がらない事態が生ずる。

対策

システム部門は、緊急対策として、以下の対策を行った。

- 共有ストレージの論理ボリューム割当ての見直しにより、ストレージ容量を確保する。
- スナップショットの世代管理を見直し、世代数を 7 → 3 世代に減らした。
- 物理的な機器の追加により十分なデータ領域が確保されるまで、新規サーバの構築案件の受付を停止する。
- 実施予定しているストレージ増設を前倒しで実施するよう検討する。
- システム化を含めて論理ボリュームのしきい値としきい値監視方法の見直しを実施する。

再発防止対策として、システム部門は、物理サーバを仮想サーバグループに移行する際の、リソース管理、性能監視を行うプロセスを策定した。

物理サーバを移行する場合、運用設計者は、業務部門（アプリケーション担当）から要件をヒアリングし、例えば、ファイル単位のバックアップ方法、ストレージ設計、専用デバイスの有無　等を整理し、同じ要件のサーバ同士をグループ化し、その仮想サーバグループ単位でのリソース見積、性能見積を行う。その場合、仮想化 SW（ソフトウェア）が追加されることにより、オーバヘッドが増大することを見逃してはならない。仮想サーバグループの性能は、「サーバ台数分＋仮想化 SW のオーバヘッド」として見積もる必要がある。リソースについても、「サーバ台数分＋仮想化 SW」として見積もる（図 3.8－3）。

このような運用設計を実施する中で、システム部門は、サービス開始までに要員のスキルを十分に高めるために、障害対策の検討、障害対応マニュアルなどの作成などを行い、運用要員教育、障害訓練（移行時、稼働時）を実施する。

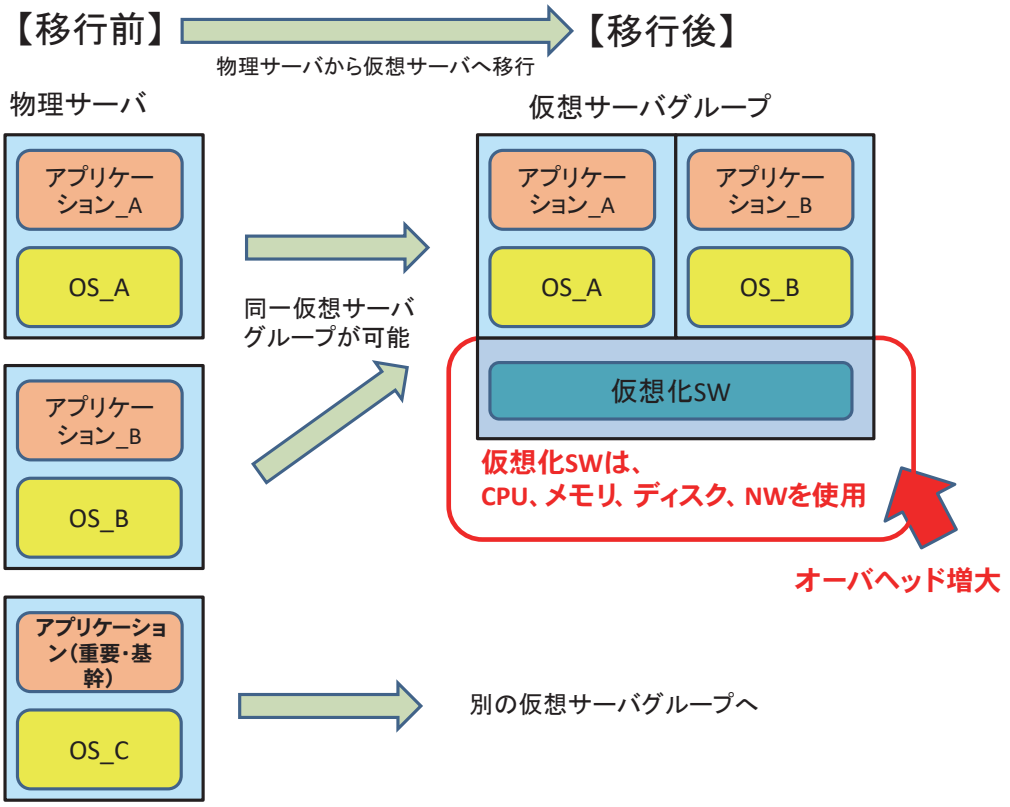


図 3.8－3　仮想サーバのグループ化とオーバヘッドの増大

効果

仮想化サーバへの移行について、システム部門は、設計時から非機能要件、運用面、障害対策を運用部門が業務部門の要求を考慮することによって、移行時の障害発生を減らすことができ、サービス稼働後も安定稼働を得ることができる。

教訓

仮想サーバになってもリソース管理、性能監視は運用の要である。

仮想サーバへの移行計画を行う場合、仮想サーバ環境としてのリソース管理、性能監視を設計時に考慮すべきである。その場合、仮想化 SW のオーバヘッドに注意することが必要である。

また、仮想サーバへの移行は、非機能要件定義を明確にしないで実施すると、障害時の復旧が素早く行われず、サービス後も運用がより複雑になったり、性能の劣化を引き起こしたりして、信頼性向上に結びつかない事態を引き起こすことになる。

3.9 不測事態発生への備えに関する教訓（T9）

教訓 T9

検証は万全？それでもシステム障害は起こる。回避策を準備しておくこと

問題

A 社の、あるアプリーションサービスを提供する 2 つの設備間において、通信障害時にタイムアウト検出機能の潜在不良が顕在化し、装置のバッファオーバフロー、再起動につながった。サービス再開までの間、利用者からのリクエスト処理が停滞した。

システムの概要は次の通り（図 3.9－1）。

- アプリケーションサービス用のサーバを冗長化。設備 1 でリクエストを受付け、設備 2 で処理を行う。
- 設備 1 には外部からのリクエストの振分制御、処理待ちリクエストのバッファ、サーバ #1、#2 それぞれのクライアントプロセス #1、#2 が実装される。
- 設備 2 には 2 つのリクエスト処理用サーバ（サーバ #1、#2）が実装される。
- 通常、設備 1 で受付けたリクエストは振分制御を介してプロセス #1 へ受渡され、設備 2 のサーバ #1 により処理され、終了するとプロセス #1 へ通知される。

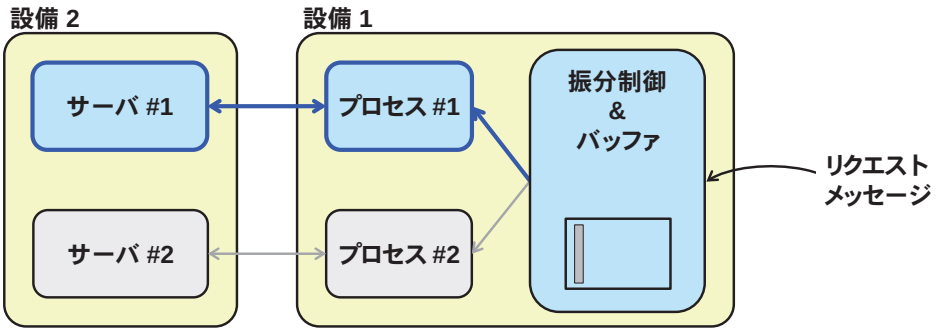


図 3.9－1　システムの構成イメージ。通常はサーバ #1 の系統が稼働

- サーバ #1 の障害をプロセス #1 が検知すると、リクエストは振分制御によりプロセス #2 へ受渡され、設備 2 のサーバ #2 により処理され、終了するとプロセス #2 へ通知される（図 3.9－2）。
- サーバの処理待ち、障害時切替え等の間に受付けたメッセージは、設備 1 の処理待ちバッファに蓄積される。
- 処理待ちバッファがオーバフローした場合、設備 1 を再起動、バッファはクリアされる。

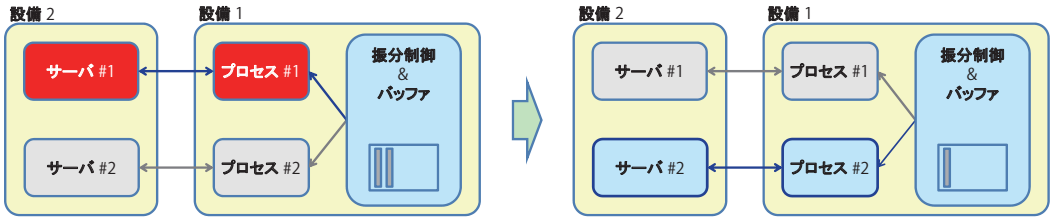
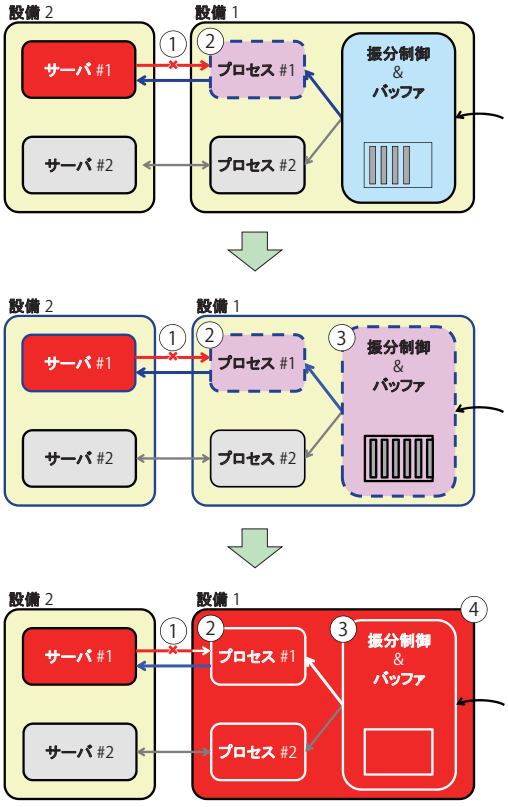


図 3.9－2　障害時の自動振分

障害発生の経緯を図 3.9－3 に示す。



① 設備 2 のサーバ #1 に障害が発生し、TCP レイヤの一方が途絶した。

② 設備 1 のプロセス #1 はセッション断を検知できず、サーバ #1 からの応答待ち状態のままとなった。

③ このため振分制御側で異常を検知してプロセス #2 へ切り替えることができず、リクエストはバッファに蓄積され続けた。

④ バッファが満杯となり、設備 1 がリブートされた。

図 3.9－3　障害発生の経緯

原因

直接的な原因は次の通り。

- 設備1のプロセス #1 における、アプリケーションレイヤのタイマ制御処理ソフトウェアのバグにより、TCP レイヤでの切断時にタイムアウトを検知できなかった。

根本的な原因は、本事例の契機となった通信障害のケースについて事前に検証できていなかったことにあるといえるが、その責を当該システムの調達側であるA社の不備に帰することは現実的ではないと考える。

- ネットワーク機器の不具合に起因する障害は、検証のため再現することさえ難しい場合も多く、取りうるすべての状況を事前に確認することは事実上不可能である。
- 当該システムは全体を外部から調達したもので、タイムアウト検出機能の動作についても、本来製造元で担保すべきものである。さらに、本事例の原因となった通信途絶は希少なケースであり、検収時点においては調達側、供給側の双方とも当該ケースの存在を認識していなかった。

対策

本事例において実際に行われた対策は次の通り。

- 暫定処置
  - 手動によりプロセス #2 へ強制的に切り替え、業務を継続した。
- 直接的な原因への対策
  - 設備1クライアントプロセスのタイマ制御不具合を修正した。
- 類似障害の再発防止策
  - タイムアウト前にオーバフローによるリブートが発生しないよう、バッファサイズを拡大した。
  - サービス復旧のための作業優先順位を定め、障害時の強制切替え手順を明記した。

当該システムがとり得るすべての状態の組み合わせを検証できていなかったことについては、対策として、設計・検証を徹底的に厳密化すべしと謳うことは、些か実現性に乏しいと考える。

現場においては、対象システムの重要性に応じて、検証に費やすことのできる時間と労力は制約される。発生確率の低い、希少なケースについて、完全に洗出し検証することは難しい。

当該システムの供給者の立場においては、システムの価値に応じて要求される品質水準を満たすよう、必要十分な検証を行ったことが確認できれば、責任は果たしていると考えるのが一般的であろう。

本事例で見ると、次のようにいえる。

- プロセス #1 とサーバ #1 の間にクローズドループ（指示に対して必ずフィードバックが得られる関係）が成立していることを確認（保証）するために、各レイヤごとの応答有無のすべての組み合わせについて動作を想定し、検証しておけば、本事例の不具合が事前に検出された可能性はある。
- テストスタブを用いて、各レイヤの応答が得られなかった場合の動作を検証することは理論的には可能だが、本事例のような不具合を検出するためには、少なくとも各レイヤの応答有無の組み合わせを網羅する必要がある。
- 上記検証方法が、当該システムの品質要求に照らして必要、または所要コストからみて許容できるものであった場合には、実施されるべきであった。

したがって、当該システムを用いて業務機能を提供するサービス提供者の立場においては、常に不具合が潜在しているとの前提に立ち、検知と障害個所の回避により業務の継続性を確保することが、障害発生の予防策に劣らず重要である。

効果

- 不具合修正により、同様の通信途絶時にタイムアウト検知が可能となった。
- バッファサイズ拡大により、設備１がリブートされる前に人手による強制切替えを実行できる時間的余裕が確保された。
- 今後製造側では設計、検証段階において、調達側では検収、統合テスト段階において、より注意深いチェックが行われるようになると期待される。

教訓

要件定義、設計、構築等の各段階（工程）において、十分な検証が行われたことが関係者間で確認・合意されていても、運用開始後の障害につながる不測の事態（要因）が完全に排除されたと保証できるわけではない。

サービス提供者は、システムには常に不具合が潜在するとの前提に立ち、業務継続性を担保できるよう、障害回避策等を準備しておく必要がある。

- 調達（購買、内製の双方を含む）局面において、当該システムが要求する品質水準を確保できるよう、必要十分な範囲で障害発生条件を網羅検証し、想定し得る障害の発生を予防する。
- 運用局面において、システム障害に備えて、設備を冗長化し、障害を検知したら発生個所を切離して運用を継続する等、当該システムが要求するサービス継続性を確保できる体制、手続きを整える。

3.10 共有ディスクのメッシュ接続に関する教訓（T10）

教訓 T10　メッシュ構成の範囲は、可用性の確保と、障害の波及リスクのバランスを勘案して決定する

問題

A 社の、サーバと NAS をフルメッシュ接続したシステムにおいて、局所的なディスク障害がシステム全体に波及し、すべてのサーバがダウンした。サービス再開までの間、トランザクション処理が中断した。

システムの概要は次の通り（図 3.10－1）。

- あるサービス用のシステムにおいて、性能及び可用性確保のため複数のサーバと 8 基の NAS をフルメッシュで接続していた。

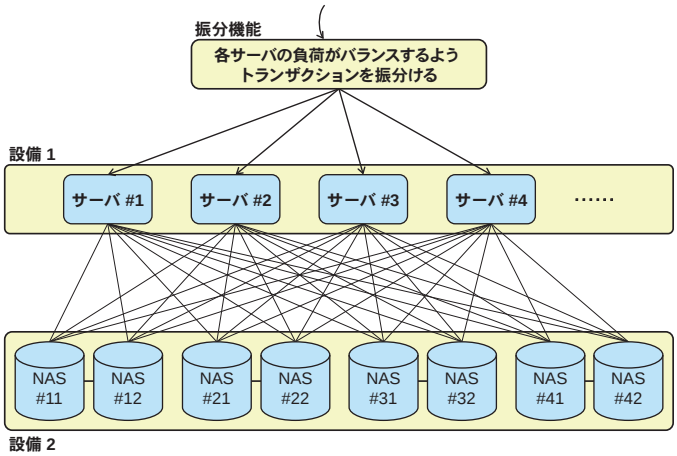


図 3.10－1　フルメッシュ構成のイメージ

- NAS #11 と #12、～ #41 と #42 はそれぞれペア（4 ペア）を組み、同時並列稼働（アクティブ / アクティブ）している。各ペアには、処理対象クラスのインスタンスが分散して保管されている。
- ペアの一方に障害が発生した際には他方で処理を引継ぐ（図 3.10－2）。

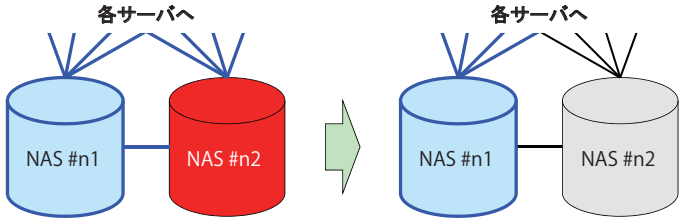
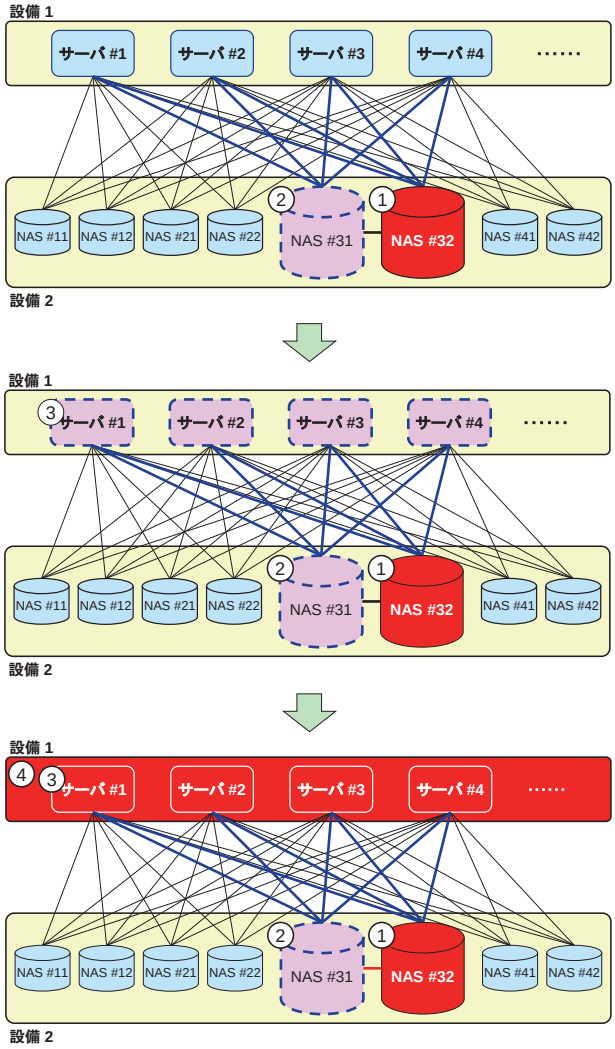


図 3.10－2　ペア構成。障害時にはペアの一方が処理を引継ぐ

障害発生の経緯を図 3.10－3 に示す。



NAS#31, #32 のペアにおいて、

① NAS #32 にハードウェア障害が発生。

② ファームウェアの不具合（バグ）により、#32 を切り離すことができず、NAS #31 への処理引継ぎ、切替えが完了できない状態となった。

③ フルメッシュ構成であったため、すべてのサーバにおいて NAS #31 のマウントポイントが「外れている」と認識された。

④ そのまま時間が経過し、全サーバがダウンした。

図 3.10－3　障害発生の経緯

原因

直接的な原因はフルメッシュ構成そのものにある。すべての NAS がすべてのサーバに接続されていたため、NAS コントローラファームウェアの不具合により障害ディスクの切離しに失敗したことがシステム全体に影響を及ぼし、全サーバのダウンにつながった。

根本的な原因は、フルメッシュ構成のリスクについて充分に考慮しないまま採用したことにある。メッシュ構成により理論上の信頼性（可用性）は向上するが、ネットワークを構成する機器・装置の一

部に発生した障害が、メッシュ化された全領域に波及する危険性も増大させる。サービス提供者として、許容可能なリスクの程度についての事前検討が必ずしも十分でなかった。

対策

本事例において、実際に行われた対策は次の通り。

- 暫定処置として、トラフィックを BCP 設備へ迂回させることによりサービスを継続した。
- NAS コントローラに対してファームウェアの機能修正パッチを適用した。
- 類似障害の再発防止のため、局所的な障害により全サーバがダウンするような事態を回避するため、フルメッシュ構成を見直した。
  - 1 基のサーバと 2 ペアの NAS を接続した「グループ」から成る構成に改めた。
    一部のデータにアクセスできなくなっても、他のデータに対するトランザクション処理は継続できるようにした（図 3.10－4）。

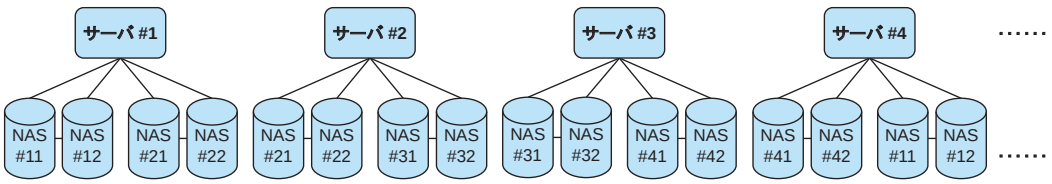


図 3.10－4　サーバと NAS のグルーピング例

  - グルーピングに対応して、振分機能を、単なる負荷分散から、各サーバが接続する NAS に応じてトランザクションを振分けるように改修した。

グルーピング後のシステム構成のイメージを図 3.10－5 に示す。

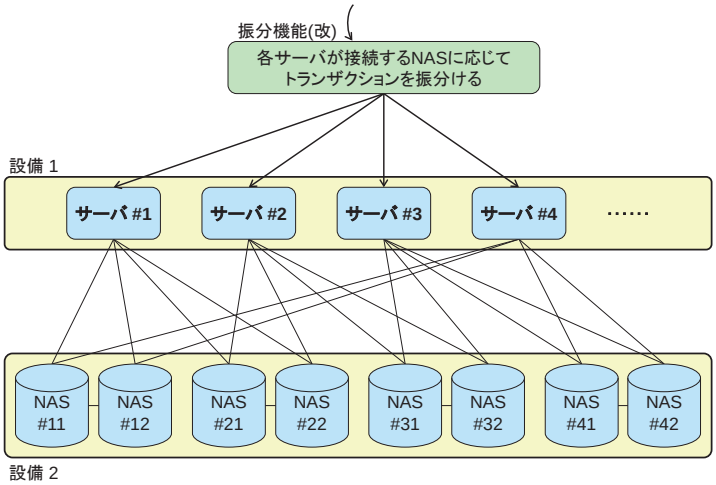


図 3.10－5　メッシュ化見直し後のシステム構成

- さらに、再発防止策ではないが可用性を担保するための施策として、迂回時の性能劣化を防ぐため、BCP 設備を増強した。

効果

NAS の局所的な障害が波及する範囲が限定されたことにより、同様の障害によりすべてのサーバがダウンするリスクは低減された。

教訓

- メッシュ構成により可用性の向上が期待できる。一方で局所的な障害が全体に波及するリスクも増大する。安易にフルメッシュ化するのではなく、目的に応じた最適な構成を検討することが望ましい。
- 本稿に揚げた構成は対策の一例であり、グルーピングの単位、メッシュ化対象範囲の考え方等については個別に検討されたい。

3.11 サイレント障害に関する教訓（T11）

教訓 T11

サイレント障害を検知するには、適切なサービス監視が重要

問題

A社の、あるWebサービスにおいて、明示的に障害が検出されていないにも拘らず性能が劣化する、いわゆる「サイレント障害」が発生した。障害が解消されるまでの間、利用者は応答速度低下の影響を受けた。

ネットワークシステムにおいて、通常の運用監視の仕組みからは検出されないまま、性能劣化等の現象が発生することを「サイレント障害」と呼ぶ。放置すれば最終的にネットワークに接続できなくなるなど、システム全体に影響が及んで大規模な障害につながることもある。

サイレント障害では、発生個所、原因の特定に多くの時間と労力を要することが多い。そのため利用者は不利益を、サービス提供者は機会損失や風評リスクを長時間にわたり蒙ることになりがちである。

システムの概要を図3.11－1に示す。

- 負荷分散のため、1台の負荷分散装置下に3～6台のサーバを配したセットを、複数配置している。

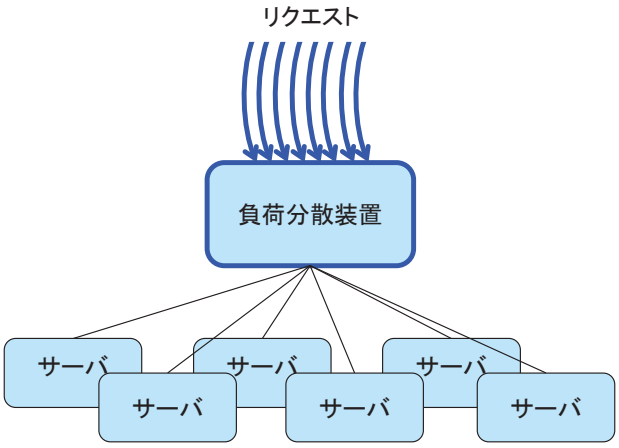
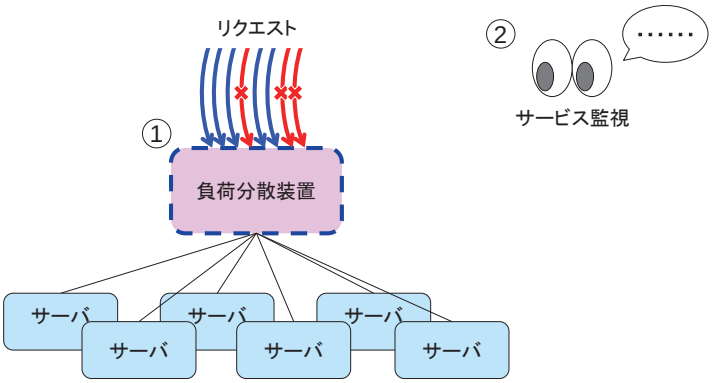


図3.11－1　システム構成のイメージ

障害発生の経緯を図3.11－2に示す。



① 負荷分散装置において、セッション数オーバによるリクエスト廃棄・再送が発生。応答速度が低下した。
② サービス監視からの通知はなく、性能が低下した状態が続いた。
③ 外部（社員）から指摘を受けるまで発見できなかった。

図 3.11－2　障害発生の経緯

原因

直接的な原因は次の通り。

- 負荷分散装置のファームウェアの不具合があった。障害発生時に使用中のバージョンにおいては、受付けるセッション数の上限が、設定値の 1/4 迄しか許容されない「仕様」とされていた。これを認識しないまま上限値を設定したため、想定を遥かに下回るトラフィック数であったにも拘わらずリクエスト廃棄が発生した。
  後のバージョンで改善（修正）されているので現在は問題ないとメーカは説明している。
- サービス監視はリクエストが一定回数連続して廃棄された場合にアラームを発するよう設定されていたが、しきい値を超えるまでには至らなかったため、サービス監視からの通知はなかった。

根本的な原因は、サービス監視条件が必ずしも最適ではなかったことにあるともいえるが、ファームウェアの不具合がなければ、特段問題視されることはなかった可能性もある。

対策

本事例において、実際に採られた対策は次の通り。

- 直接的な原因への対策
  - 負荷分散装置のファームウェア更新
  - サービス監視条件を変更。リクエストが一定時間内に一定回数以上廃棄されたら通知するよう改めた。
- 類似障害の再発防止策は次の通り。
  - 取扱ベンダを一本化し、製品の情報を直接入手できるようにした
  - 本事例の発生以前から、定期的なサービスへのアクセス、目視による SNS 上の「つぶやき」監視等を行っている。つぶやきから監視機能よりも早く障害を検知できる場合もあり、有効であると認識している。

－ さらに、「インバリアント分析」を用いた早期検知の試みを開始した。

本事例においては、サービス監視条件の設定が妥当であれば、より早い段階で障害の兆候を把握できた可能性はある。

運用中にサイレント障害の発生を速やかに検知し、分析・対処につなげるための基本的な取り組みは、サービス監視機能を用いる、或いは実際にサービスにアクセスしてみる等の方法により、性能劣化が見られないかをチェックすることである。

基本的な取り組みを実践した上で、更なる早期発見、対処を望むサービス提供者においては、監視機能に加えて障害検知、分析のための技術・製品が用いられるようになっている。

参考に、最近登場している、サイレント障害の検知や分析を行うための技術の例をいくつか紹介する。

(1) サイレント障害切分けシステム

IP ルータ網監視システムのサブシステムとして開発された。従来の障害検出技術によってサイレント障害を発見するには、高度な技能を持つ保守担当者による長時間の探索が必要であったが、当該システムが提供する「サイレント障害検出機能」と「サイレント障害発生区間特定機能」により、サイレント障害の検出及び障害個所の特定が容易になった。

「大規模な IP ネットワークにおける高精度な障害切り分けシステムの開発」（NTT DOCOMO テクニカル・ジャーナル Vol18 No.1）

(2) インバリアント分析技術の応用

性能情報間の相関関係のうち、平常時に変化しない関係（インバリアント）を自動的に学習してモデル化し、そのモデルと一致しない「いつもと違う」動作をサイレント障害として検知する。自動的に性能情報を収集し、リアルタイムで分析を行い、障害を検知した時点で警告メッセージを通知する機能もある。

「WebSAM の分析技術と応用例～インバリアント分析の特長と適用領域～」(NEC 技報 Vol65 No.2/2012)

(3) ビッグデータ分析技術の応用

ネットワーク装置のログ（Syslog）や SNS 上の「つぶやき」情報といったビッグデータをリアルタイムに分析し、障害の早期発見、ひいては予兆検知に利用する技術が研究されている。

「Syslog と SNS 分析によるネットワーク故障検知・原因分析技術」（NTT 技術ジャーナル 2013.07）

効果

- 直接的な原因への対策により、システム障害に至る前の段階で、性能劣化を検知できる確率が高まった。
- さらに、SNS 監視や、各種技法を用いることにより、サイレント障害をより早期に検知できるようになると期待できる。

教訓

サービス中断、サーバ停止といった「分かりやすい」障害と異なり、サイレント障害はその発生を検知することが難しく、解決にも時間を要しがちである。

サイレント障害検知のためには適切なサービス監視が最も重要であり、基本的な取り組みである。更なる早期検知、分析のための技術、製品等の利用も始まっている。

3.12 互換部品の入れ替えに関する教訓（T12）

教訓 T12

新製品は、旧製品と同一仕様と言われても、必ず差異を確認！

問題

ユーザ X 社の制御系システム（24 時間稼働）には、2 重化されている制御装置（CPU とディスク装置を含む）が組み込まれている。

その制御装置の A 系の LAN ボードに不具合が生じた。そこで、制御装置の A 系のみを停止させて LAN ボードを交換することとした。

先ず A 系を停止させる自動シーケンスを実行した。A 系 CPU が切り離された（図 3.12－1 ①）後、B 系 CPU から両系のシステムディスク装置へのリセットが行われた（図 3.12－1 ②）。この処理において、両系のシステムディスク装置ともタイムアウトが発生し、制御装置全体が動作を停止した（図 3.12－1 ③④）。

A 系、B 系ともに再立上げを続けたが、復旧できなかった。原因究明中に、開発環境の制御装置が本番機として使用できることが分かり、急遽代用機として使用した（図 3.12－1 ⑤）。

このサービスの利用者は、1 日中影響を受けた。

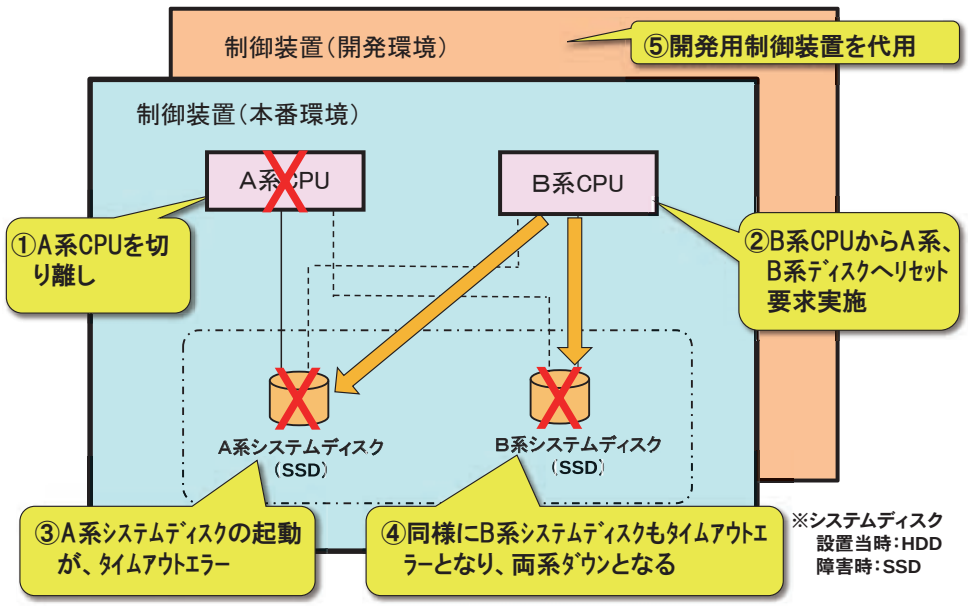


図 3.12－1　障害状況

原因

システム導入当初、制御装置のシステムディスク装置は，HDD を使用していた。

保守運用フェーズに入り、数年前に HDD から SSD に交換した。

しかし、今回の障害では、制御装置の片系 CPU 停止時の OS のタイマ監視時間と SSD のリセット完了時間とに不整合があった。直接の原因は、ベンダが、SSD は、標準として規定されているコマンド・インターフェースで HDD と互換性があると認識していたため、HDD と SSD の起動時性能（標準の規定外の事項）での差異を見逃してしまったことによる（図 3.12－2）。

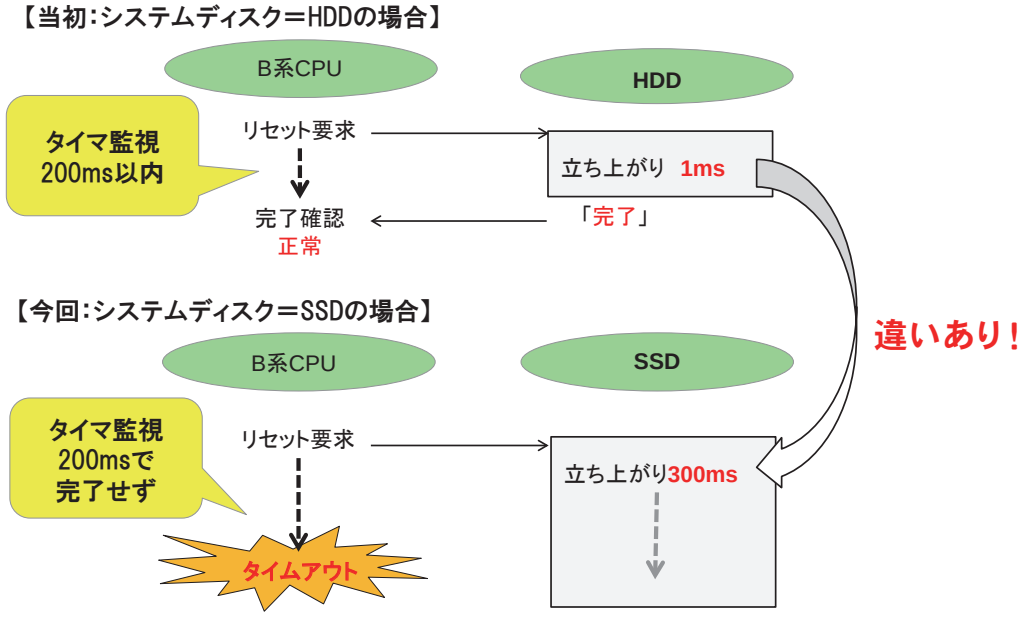


図 3.12－2　HDD と SSD の立ち上がり時間の相違

また、根本原因は、今回の障害を防ぐことができた（事前に不具合を発見できた）時点が 2 カ所あったが、以下のようにそれを見逃してしまったことである。

① 数年前、ベンダが、システムディスクを HDD から SSD に交換したとき

ベンダは、交換する新部品がシステムと整合しているかについて、SSD は、HDD と互換性がある仕様になっていると判断してしまい、以下のような視点からの確認を怠ってしまった。

- 機能やインターフェースが一部異なる。また、新機能が追加されている。
- 異なる部品、製品の場合には，設計思想そのものが異なっている。
- 機能仕様は同じであっても、性能等の非機能仕様が異なる。

また、テストの中でも、先の観点から、2 重化されている制御装置の片系を交互に停止させる動作確認を行うべきであったが、テストの必要なしと判断してしまったため、不具合を発見することができなかった。

② 今回の本番作業前に行った事前確認のとき

ベンダは、自工場での手順書確認で制御装置の片系を止めて LAN ボードの交換を実施していたが、HDD で動作確認しただけで「問題なし」と報告し、本番環境と同じ SSD での動作確認を行っていなかった。そのため、ここでも不具合を発見することができなかった。

対策

直接の対策として、ベンダは、プログラムの更改（制御装置 OS のタイマー監視時間を SSD のリセット処理完了までに延長）を行った。さらに、部品の更新時に確認不足を起こさないために社内ルールを見直し、作業手順を自工場内で確認するときは、工場内の装置は現地と同一構成で行うようにした。

また、ユーザは、制御装置に関する作業準備について、影響範囲を関係部署と共有し、事前に開発環境の制御装置にて動作確認を行った上で、本番環境の部品交換を行うようにした。

制御系システムは、保守運用が非常に長期になる場合が多い。そのため、途中で部品の供給が中止となったり、ソフトウェアの保守契約が切れてしまったりなどの理由により、新製品（HW、SW）に交換せざるを得ない事態が起こる。

予防対策では、今回の事例のように、新製品と従来製品との差異を見逃さないためのルールを作ることが重要となる。その場合、従来と異なる新装置、新ソフトウェアがシステムに組み込まれるときは、どんなに互換性があると言われているものであっても、変更部分と非変更部分との整合性を確認することが重要である。

そのためには、ベンダ、ユーザ双方が、相手の役割分担を支援し合うことが重要である。特にユーザがベンダの障害予防対策の実施状況を確認する、つまり、ベンダの確認の見落としを補完することがより有効である。ユーザは、ベンダからの報告を鵜呑みにせず、リスク、ハザード分析などを行い、システムの重要度に応じてテスト範囲を明確にし、受け入れテスト（デグレードテスト）を行うなど、ベンダ側に一歩踏み込んだ対応が重要である。

ベンダとユーザ双方が新製品をシステムに組み込む場合のそれぞれの対策例を一覧で示す。さらに、本編の他の参照すべき教訓（T6, T7）を加えた（表 3.12－1）。

一覧の作業項目は、対象となるシステムや役割分担の決め方によって異なるので、ベンダとユーザで十分話し合って決める必要がある。

表 3.12－1　対策例一覧

	ベンダ	ユーザ
予防対策	①部品の機能仕様、非機能仕様の相違点の洗出し ②部品の更新時に確認不足を起こさないための社内ルールの見直し ③作業手順を工場内で確認する時、工場内の装置は現地と同一構成とし、実施 ④機能停止を伴う部品交換作業は、制御装置の機能に影響しないよう開発用の装置にてオフラインで動作確認を行い、その結果を確認した上で部品交換を実施	①ハザード分析の実施　→　リスク対策 ②関係者ベンダを交え、新製品と従来製品との差異をチェック ③ベンダの工場内でのテスト内容、結果を確認した上で、ユーザの受入テスト内容の確認 ④開発環境での事前確認 ・相違点に関する動作確認の実施 ・デグレードテストの実施 （活用できる他の教訓） 教訓T6：テスト環境と本番環境の差異を体系的に整理し、障害のリスク対策を練る。
実行時対策	①本番作業実施前にバックアップを準備する。 （活用できる他の教訓） 教訓T7：バックアップ切替が失敗する場合を考慮すべし	①ハザード分析によるリスク対策の策定と実施 ②上記リスク対策として、作業に因るシステム停止の影響範囲を関係部署と共有し、これを想定した作業計画を立案し、実施

効果

制御系システムを長期間保守運用する場合、今までと異なる新装置、新ソフトウェアをシステムに組み込まざるを得ない事態が生ずる。その場合、ユーザとベンダが役割分担の上、連携し、補完しあいながら、変更部分と非変更部分との整合性を確認することにより、障害を減らすことができる。

教訓

新旧製品の差異に因る障害は、互換性があると言われても、変更部分と非変更部分との整合性を確認することが重要である。

特に、ユーザは、ベンダの予防対策に踏み込みことにより、ベンダの予防対策をより効果のあるものにする。

つまり、ユーザとベンダは、既存製品から新製品に交換するときは、同じ物と言われても別物に交換するものとして対策を考え、連携し補完し合いながら、従来製品と新製品との整合性を確認することになる。

3.13 業務シナリオテストに関する教訓（T13）

教訓 T13　利用者の観点に立った、業務シナリオに即したレビュー、テストが重要

問題

A 社が提供する Web サイトにおいて、サービス申込みができなくなる事態が生じた。システムの概要は次の通り（図 3.13－1）。

- 業務系システムが既にサポートしているサービス J について、Web 経由で申込・利用を可能とする窓口を、Web サイトに追加した。
- 申込時には、当該サービス利用のための顧客の適格要件確認が必要である。Web サイトでは①適格確認 → ②サービス申込 と続けて入力した後、適格確認済と判断されたデータを業務系システムに連携する。業務系システムはこれを用いてサービス登録処理を行う。
- 申込の受付時間は 07：00 ～ 19：00 である。

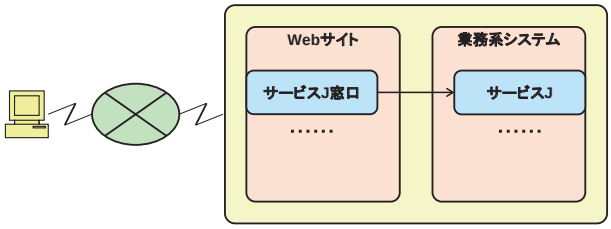


図 3.13－1　システムの概要

障害発生の経緯は次の通り。

- Web 窓口リリース後、18：45 ～ 19：00 の 15 分間に限り、Web サイトからのサービス J 申込みがすべて「適格確認が未済」との理由でエラーとなった。
- 顧客からの連絡で判明した。

原因

直接的な原因は次の通り。

- Web サイト上のサービス J 窓口と業務系システムの間で、適格確認情報、サービス申込情報の連携について、次のような制約があった。
  - 適格確認情報は 18：45 以降業務系システムへ連携されない。
  - サービス申込情報は 19：00 まで業務系システムへ即時連携される。
- 当初の全体設計では、業務系システムへ連携されるデータは適格確認済のものに限定されるので、

業務系システムではWebサイトからの申込については適格確認済チェックを行わないものとしていた。

- 実際には、Web側で適格確認済のデータに対して、業務系においても再度同様の適格確認済チェックを行っていた（図3.13－2）。

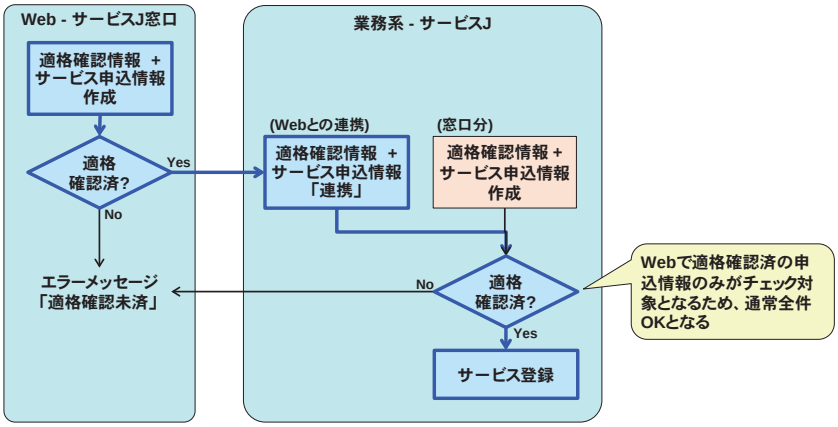


図3.13－2　処理の概要

- 問題の15分間にWebから受付けたデータは適格確認情報を持たないため、このチェックによりすべてエラーとされた（図3.13－3）。

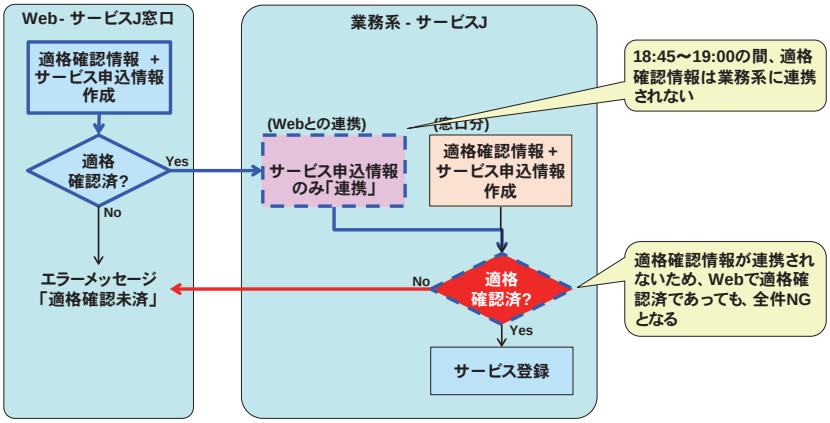


図3.13－3　障害時の状況

根本的な原因は、Webサイトと業務系システムの間で、適格確認データの連携に関する仕様が、全体設計から個別システム設計に正しく引継がれず、そのまま実装されたことにある。

- 当初の全体設計では、①店頭取扱分は業務系システム側で適格確認済チェック、②Webサイトからの申込分はWebサイト側で適格確認済チェック、と定めていた。この処理内容であれば、非同

期時間帯（18：45 ～ 19：00）の間も、Web サイトからの申込データを用いてサービス登録を行うことができる（図 3.13－4）。ところが、業務系システムの内部設計段階で、業務系システム側で全件チェックする仕様に変わってしまった。

- 業務系システム開発担当者に、Web サイトからの申込情報に対して再度チェックを行うことについて、冗長ではあっても不当なものではなく、実害はないとの誤認があった。

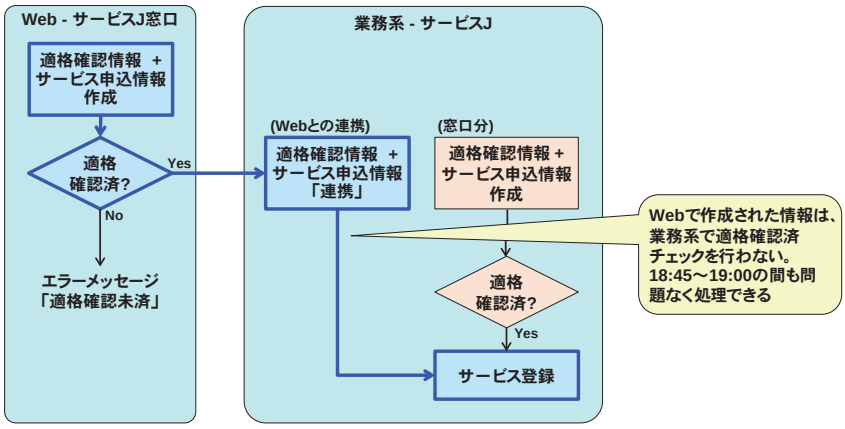


図 3.13－4　当初の設計

- Web サイト開発担当からも、非同期時間帯（18：45 ～ 19：00）における業務系システムの処理設計について、積極的に確認することはしなかった。
- その後のシステムごと及び全体レビューにおいても、この設計変更の影響、問題点等について指摘されることはなかった。

リリースまでに発見できなかった原因は次の通り。

- 適格確認、サービス申込それぞれのデータが、それぞれが連携可能な時間帯において Web サイトと業務系システム間で正常に連携できることは確認した。
- Web サイトからのサービス J 申込にかかわる一連の処理が完遂できるかという、利用者の観点に立った、業務シナリオに即したテストを実施しなかった。
  本事例においては、サービス開始前、開始時（直後）、通常運用時間帯、終了時（非同期時間帯）、終了後の各タイミングで、Web サイトからのサービス J 申込が実行・完了できるか否かを検証していれば、問題は発見できた筈である。

対策

本事例において実際に行われた対策は次の通り。

- 直接的な対策
  － 業務系システムのプログラム修正。Web サイトからのサービス申込に対しては、業務系で適格確

認済チェックを行わないようにした。

- 類似障害の再発防止策
  - 要件定義、設計段階において、関連システム全体で、対象業務の実現性が担保されているか、インターフェース変更個所において曖昧な記載はないか等について、ウォークスルーによるレビュー等を通じて、関係者相互で確認するよう周知徹底した。
  - 複数システムにより提供されるサービスについては特に、テスト設計・実施段階において、利用者の観点に立った、業務シナリオに即した検証を計画・実施するよう周知徹底した。確認のためのチェックリストも整備した。

効果

- 直接的な対策により、Web サイトからのサービス J 申込業務は支障なく提供されるようになった。
- 再発防止策により、今後同様の案件において、本来リリース迄に除去可能であった潜在バグに由来するような障害は少なくなると期待できる。

教訓

複数システムの連携により実現される業務サービスにおいては特に、既存部分と新規追加部分のインターフェースや、両者に跨る処理に着目し、利用者の観点に立った、業務シナリオに即したレビュー、テストが確実に行われるよう、注意する必要がある。

3.14 Web ページ更新時の性能に関する教訓（T14）

教訓 T14

Web ページ更新時には、応答速度の変化等、性能面のチェックも忘れずに

問題

A 社が提供する Web サイトにおいて、特定のサービス B にアクセスしにくい事態が生じた。応答遅延により、多くの顧客がサービス B を利用できなかった。

システムの概要を図 3.14－1 に示す。障害発生の経緯は次の通り。

- サービス B のトップページのコンテンツを更新した。
- リリース直後より、利用者が A 社企業サイト上のサービス B トップページをクリックすると、応答に長時間を要し、サービス B 本体に接続できないケースが多発した。
- DDoS 検知装置での検知、及び利用者からの照会により、障害発生が判明した。

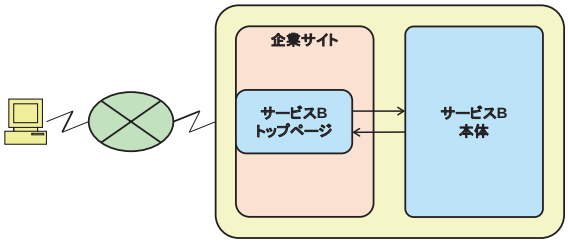


図 3.14－1　システムの概要

原因

直接的な原因は次の通り。

- サービス B の業務部門担当が、当該トップページコンテンツの大規模リニューアルを行った。その結果、1 顧客当たりのダウンロードサイズが従来の約 4 倍に増大した。
- データ量増大による応答速度等への影響を確認しないまま、当該トップページを A 社企業サイトに組込みリリースした。その結果、利用者が当該ページにアクセスすると、ネットワーク帯域の逼迫、応答遅延によるスレッド枯渇が発生した。
- 大部分のページは高速ネットワークサービスを経由してアクセスされるようになっているが、当該トップページへのアクセスは同サービスを経由していなかった。

根本的な原因は次の通り。

- 各サービス用サイトのコンテンツは、それぞれの担当業務部門の管理に任されており、内容の追加・更新に際して、システム部門等の専門家が技術的な観点から評価することはルールとして定められ

ていなかった。

- 業務部門は、Web ページコンテンツのサイズ変更（増大）が、サービスの応答速度等に大きく影響を与える危険性について、それほど懸念していなかった。
- Web サイトの運用を担当するシステム部門では、コンテンツ更新にともなうデータ量の増減について特に自発的に注視することはなく、自部門で所管するシステムとの連携を確認する程度であった。少なくとも高速ネットワークサービスを経由せずにアクセスされる Web ページのコンテンツに対しては、運用担当の観点から注意しておくべきであった。

対策

本事例において実際に行われた対策は次の通り。

- 暫定的な対策
  - 当該トップページコンテンツを更新前のものに戻し、サービスを再開した。合わせて、ネットワーク帯域の拡張、受付スレッド数の拡張を行った。
- 直接的な対策
  - 当該トップページへのアクセスを高速ネットワークサービス経由に変更し、再リリースした。
  - コンテンツ変更量の自動チェック機能を導入し、最新のコンテンツ量とアクセス量を可視化した。
- 類似障害の再発防止策
  - 業務部門が Web ページコンテンツを更新する際には、システム部門が技術的な観点で確認を行うことを、コンテンツ追加・更新手順書に明記した。
  - システム部門は Web ページ更新時のコンテンツ量の増減を確認し、必要に応じて打鍵確認や机上計算を行い、サービス提供に悪影響があると判断した場合には、業務部門に対してリリース中止を指示できるようルールを改めた。

効果

- 直接的な対策の結果、当該トップページを経由したサービス B へのアクセスに支障がなくなった。
- 再発防止策により、Web ページの更新にともない性能が著しく低下するような事態は少なくなるものと期待できる。

教訓

業務内容をよく知る担当部門が、当該業務サービスにかかわる Web サイトのコンテンツ管理に責任を持つことは妥当である。ただし、単に内容を追加・更新するだけでなく、リリース前にシステム部門等の専門家により、技術的な問題の有無について評価・助言を受けることも、業務部門の責任の一端と認識すべきである。

Web コンテンツの管理における各部門の役割と責任、実施すべき作業項目等は、規定・手順書等に明記し、周知を図ることが望ましい。

3.15 データの一貫性確保に関する教訓（T15）

教訓 T15

緊急時こそ、データの一貫性を確保するよう注意すべし

問題

A社が提供するWebサービスにおいて、顧客ごとのサービス利用状況と人口統計データを組み合わせて分析し、顧客を分類するためのカテゴリ判定に不正が生じた。修正対応までの約1日間、一部の顧客に対して不適切なカテゴリが適用された。

カテゴリ判定処理の概要は次の通り（図3.15－1）。

- カテゴリ判定・適用は毎月末に夜間バッチ処理で行う。
- ①事前に顧客データ（オンライン）から作成した顧客データ（バッチ処理用）、②顧客別の月次取引実績、③外部から取得する人口統計データを用いて、各顧客のカテゴリを算出する。
- 結果を顧客データ（オンライン）に反映する。

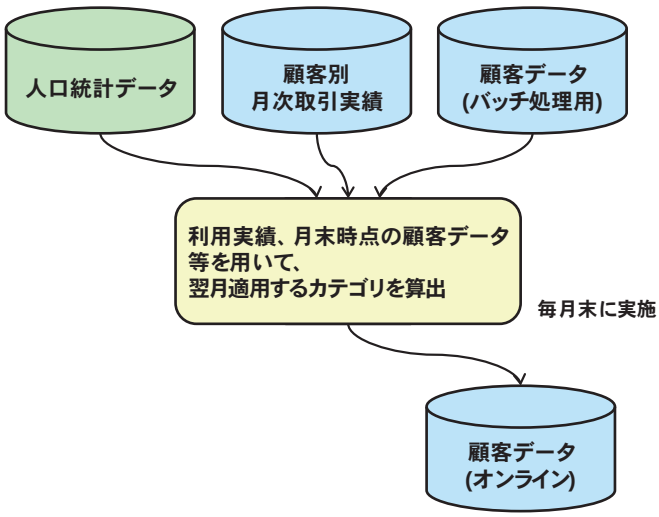
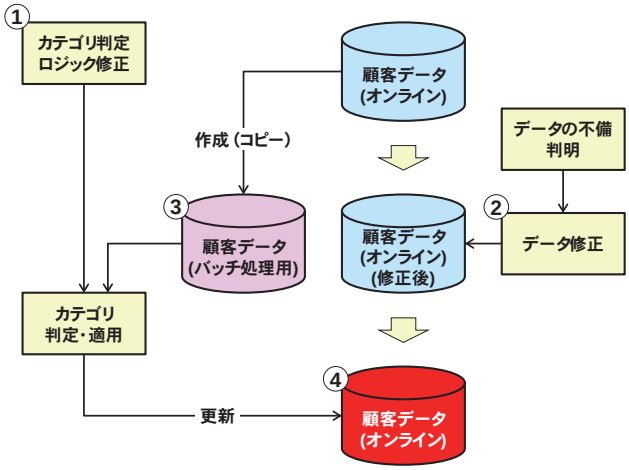


図3.15－1　処理の概要

障害発生の経緯を図3.15－2に示す。



① カテゴリ判定ロジックを修正した。
② 新ロジックに対応するためのデータ移行の際に顧客データの不備が判明。業務部門の依頼により、緊急作業として顧客データ（オンライン）に対し修正を行った。
③ このとき、既に作成済みであったカテゴリ判定バッチ処理用顧客データは修正または再作成されなかった。
④ 月末にそのままカテゴリ判定処理を行い、顧客データ（オンライン）を更新したため、一部の顧客について誤ったカテゴリが適用された。

図 3.15－2　障害発生の経緯

原因

直接的な原因は次の通り。

- カテゴリ判定に用いる顧客データ（バッチ処理用）の内容に不備があった。当該データの源である顧客データ（オンライン）に対する修正が顧客データ（バッチ処理用）に反映されていなかった。

根本的な原因は次の通り。

- 顧客データ（オンライン）の修正が異例かつ緊急を要するものであったため、影響範囲の事前調査が不十分であった。
- 修正作業及び関係各所への報告に意識が集中し、後続する通常業務への引継ぎが疎かになった。
- データ更新時の影響確認を効率化するための、原本（マスタ）と複製（コピー）の対応表等の情報が整備されていなかった。

対策

本事例において実際に行われた対策は次の通り。

- 直接的な対策
  - 修正済み顧客データに基づくバッチ処理用データを作成。カテゴリ判定を再実行した。
- 類似障害の再発防止策
  - 緊急時対応に際して、通常業務とは異なる手順、内容の作業を行うことの影響範囲を見極め、対応結果が後続する平常時の業務及びシステム運用の流れに確実に繰り込まれるよう、確認手順、テストケースの洗出し等を特に意識的に行う旨周知した。

効果

- 直接的な対策により、一部顧客に対する不正なカテゴリの適用は約１日で解消された。
- 再発防止策により、緊急時対応の影響により後続の通常業務に支障を来すような事態は少なくなると期待できる。

教訓

システム内には、ある原本（マスタ）データに対して、業務プロセスやアプリケーションの必要に応じて、複数の複製（コピー）データが存在する。これらの原本及び複製は、各業務における追加・更新・削除等を通じて、一貫性（インテグリティ）が保たれなければならない。

一貫性を確保するためには、原本とそこから派生するすべての複製の対応関係を把握し、各業務プロセスを通じてそれぞれのデータがどのように同期、更新される必要があるかについて整理し、作業時に参照する、あるいは必要に応じてツール等を利用して自動的にデータが同期、更新されるよう、作業ルール・手順等を明確化し、環境を整備する必要がある。

通常業務について、上記のような作業手順・環境を整備済みの組織であればなおさら、緊急時対応の影響により一貫性が破綻するような事態は避けたい。自動化の要否については管理対象の重要性やコストを勘案して個別に決定すれば良いが、少なくとも緊急時対応において、以下の点に十分な注意を払い、作業手順等に定めておくことが望ましい。

- 作業開始時に、一貫性の観点から影響を受けるデータを把握しておくこと
- 作業終了時に、それらのデータを同期させ、後続の通常業務に支障を来さないようにすること

3.16 修正パッチの適用に関する教訓（T16）

教訓 T16　システム構成機器の修正パッチ情報の収集は頻繁に行い、緊急性に応じて計画的に対応すべし

問題

A 社はオンラインによる情報登録及び情報照会の基幹業務システムを当初はオンプレミスで運用していたが、運用コストの削減を目的に複数企業間の共同利用を進める方針となり、B 社が提供するデータセンターに移行した。B 社が提供するシステムは、業務システム用のサーバと負荷分散装置に分かれている。業務システムのサーバだけでなく、負荷分散装置も仮想化されており、その一つの論理区画を A 社は利用していた。ある日、オンライン開始時からこのシステムに障害が発生してまる1日業務が停止した。基幹オンラインシステムが端末から起動できず、すべての窓口でデータベースの更新をともなう処理の受付けができなかった。

システムが障害となった経緯は以下の通りである。（図 3.16－1）

① 負荷分散装置のファームウェアで行っているある処理（sod プロセス）にてメモリ不足エラー（out of memory）が発生した。

② 待機系がスタンバイからアクティブへ切り替わり、この段階で未だ稼働系がアクティブであったため、両系間で多数の電文が繰り返し転送される現象（系間ループ形成によるマルチキャストストーム）が発生した。

③ L2 スイッチのポートが閉塞した。

④ sod プロセスが再起動された。

⑤ 稼働系がスタンバイへ切り替わった。

⑥ 系切替え（フェールオーバ）動作が完了し、待機系に切り替わったが、待機系自体が out of memory に近い状態であったため、極端なレスポンス悪化が発生した。

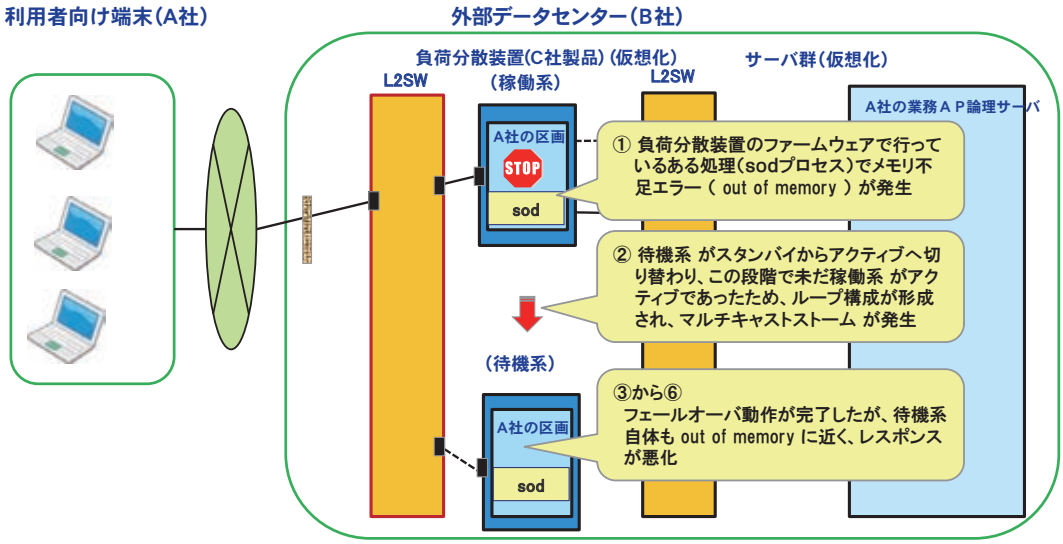


図 3.16－1　障害の経緯

原因

直接の原因は、C 社製負荷分散装置の sod プロセスのメモリ資源が時間とともに増加するという既知の不具合であった。本不具合に関する技術情報とファームウェアの修正パッチは、A 社のシステム障害が発生する約 1 カ月前に、C 社より公表されていた。

根本原因は以下である。

B 社は、システム構成機器の技術情報の確認サイクルを、3 カ月に 1 回程度と非常に粗く設定していた。そのため、パッチの公表に気づかず、本障害発生前に出ていた負荷分散装置の重大障害の修正パッチを適用できなかった。また、A 社は、同社のシステムを構成する負荷分散装置に関する技術情報が、メーカより時々公表されていることを認識していなかった。

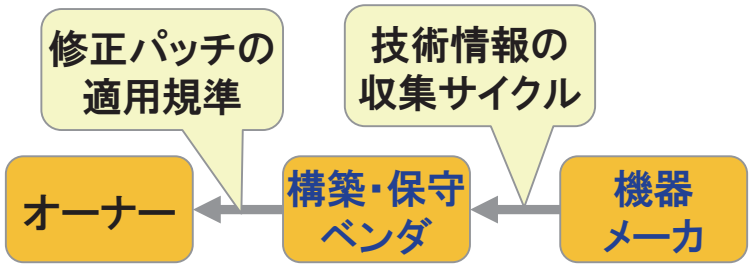


図 3.16－2　パッチ適用の流れ

対策

原因となった不具合は、装置の再起動により障害発生を抑止できるものであったため、直接対策としては、まず、負荷分散装置を再起動した。

その後、期間をおいて負荷分散装置における既知の不具合に関する技術情報を確認の上、業務への影響が少ないタイミングでその修正パッチを適用した。

再発防止策としては、A 社と B 社とで協議の上、技術情報の確認サイクルを見直して 3 カ月に 1 回から 2 週間に 1 回に変更した。特に重大な不具合では、修正を反映させる手立てや修正の優先度付けを行うこととした。

効果

構成機器の技術情報の収集に漏れがなくなり、障害発生が予防できる。

教訓

ネットワーク機器についても、修正パッチ情報等の技術情報を定期的に検索し、その緊急性に応じてパッチを当てるか否か判断する。また、特に重大な不具合では、修正を反映させる手立てや修正の優先度付けが重要である 。

特に、クラウドサービスなどでは、パッチの適用については、オンプレミスと異なり、利用者には、自らパッチの情報を得て適用する方策がないため、パッチ情報の提供者（サービスやハード・ソフトを提供する側）とパッチの適用判断者（主にサービスやハード・ソフトを所有する側）がお互いに協力しあうことが信頼性の向上につながると考える。

3.17 定期的な再起動に関する教訓（T17）

教訓 T17

長時間連続運転による不安定動作発生の回避には定期的な再起動も有効！

問題

教訓 T16 と同じ事例である。特記事項としては、A 社のシステムでは、本稼働以来、負荷分散装置は 8 カ月以上連続運転状態であり、一般的なネットワーク機器と同様に再起動をしたことがなかった。

原因

直接の原因は、C 社製負荷分散装置の sod プロセスのメモリ資源が時間とともに増加するという既知の不具合であった。本不具合に関する技術情報とファームウェアの修正パッチは、A 社のシステム障害が発生する約 1 カ月前に、C 社より公表されていた。

根本原因は以下である。

負荷分散装置の再起動によりこの不具合による障害発生を回避することが可能であった。

しかし、A 社のシステムでは、本稼働以来、負荷分散装置は 8 カ月以上連続運転状態であり、一般的なネットワーク機器と同様に再起動をしたことがなかった。

今回の障害は定期的に再起動をしていれば顕在化しなかった。

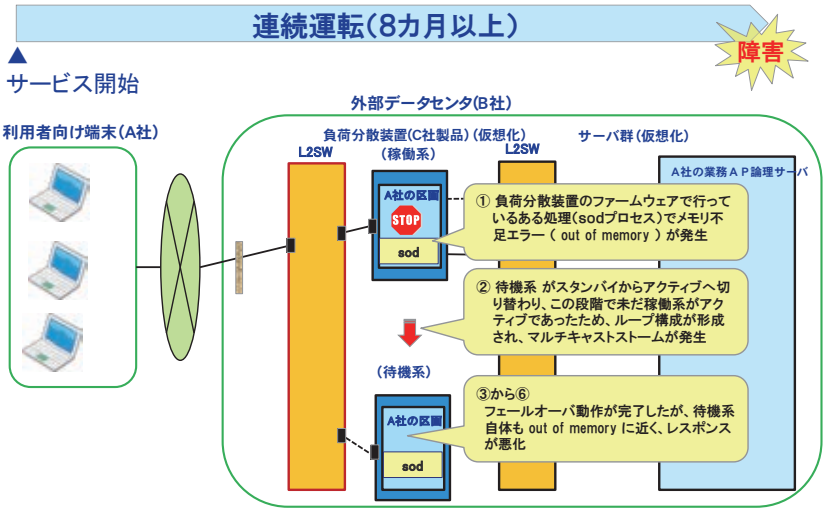


図 3.17－1　障害の経緯

対策

原因となった不具合は、装置の再起動により障害発生を抑止できるものであったため、直接対策としては、まず、負荷分散装置を再起動した。

その後、期間をおいて負荷分散装置における既知の不具合に関する技術情報を確認の上、業務への影響が少ないタイミングでその修正パッチを適用した。

再発防止策としては、本来の対応として、装置に関する技術情報をこまめに確認するルールを設けた。

さらに次善の対応として、システムの再起動のサイクルを検討し、定期的な再起動も行うこととした。具体的には、保守ベンダ（B 社）と協議して、毎月の定期保守日に状況を見て再起動を行うこととした。

効果

ネットワーク機器を定期的に再起動することにより、長時間連続運転による不具合の顕在化が回避でき、障害発生のリスクを低減できる。

教訓

ネットワーク機器については、定期的に再起動することにより、長時間連続運転による不具合の顕在化が回避できることが経験的に知られている。

かつては単純な機能がハードウェア制御により実現されていたため、可用性向上等のために連続稼働されていたが、近年では複雑な機能がファームウェア／ソフトウェアにより実現されているものが出現してきている。そのため、潜在的な不具合によるシステム障害発生のリスク低減のために、サービスに影響のないタイミングで定期的に再起動することも有効である。

3.18 既存システムとのデータ連携に関する教訓（T18）

教訓 T18　新たなサブシステムと老朽化した既存システムとを連携する場合は両者の仕様整合性を十分確認すべし

問題

A社のシステムは日中のオンラインと夜間バッチで構成されている。日中のオンラインにより受け付けたサービス要求を夜間バッチに連携し、そのサービス実行処理を完結する流れとなっている。

ある日、特別な事象が契機となり、オンラインで大量の入力処理が集中した（図3.18－1①）。

しかし、オンラインではデータ制限がなされておらず、そのまま夜間バッチに連携されたが、夜間バッチの処理能力の制限値を超えたため異常終了した（図3.18－1②）。

その制限値を拡大して夜間バッチを再実行したが、異常終了時に欠落したデータの復元作業が難航し、夜間バッチ処理に予定以上の時間を要した（図3.18－1③）。

ぎりぎりの時点での判断により、翌朝の通常時刻でオンライン処理を開始するため、夜間バッチを中断しバッチからオンラインへの切替えの実行に着手した（図3.18－1④）。

強制中断の結果、以降の（あるいは，残存した）バッチの自動運行ができなくなり、手動で夜間バッチを実行せざるを得なくなった（図3.18－1⑤）。

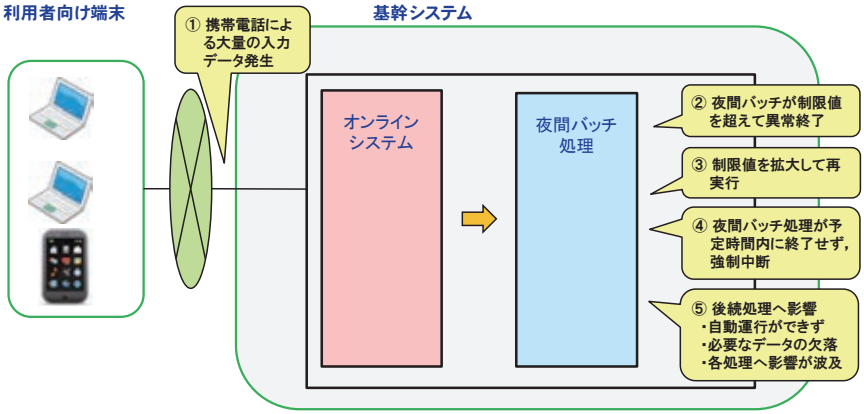


図3.18－1　障害の経緯

この結果、膨大な作業が発生し、処理失念や誤処理による多数の副次的障害も発生した。これにより、翌日のオンライン開始処理とこれにともなう各種の処理が大幅に遅延した。この状況は1日では回復せず、影響は日を重ねるごとに広がり、収束までに10日間を要した。顧客への影響としては、オンライン業務のサービス開始が遅延、及びサービス停止となり、一部のサービス明細が欠落することとなった。

原因

直接の原因は、夜間バッチの処理能力の制限値を超えたため異常終了したことである。

携帯電話からのバースト的なサービス要求（あるいは，サービス利用）データが無制限に受け付けられて、後続のバッチシステムに連携された。

根本原因は以下である。

バッチシステムの１日の処理量には上限が存在すること、それを超過した場合は異常とみなし処理停止することが確認できていなかった。また、バッチ処理を一旦強制中断させると以降の処理の自動運行はできず、手動によらざるを得ないことも把握できていなかった。

対策

本件に対する再発防止策を以下に示す。

(1) 既存システムの要件定義の有無と要件定義の内容を再度チェックして見える化し、これをもとに現在の環境との整合性や、新たなサブシステムを構築して既存システムと連携する場合は両者の整合性を確認する。これらの事項を盛り込んだルールを作成しマニュアルに取りまとめる。

さらに、以下についても実施しておく。

(2) システムが異常終了しても途中から再開始可能な仕組みと対応手順をあらかじめ考慮しておき、異常終了してもシステムが誤動作せずに制御できるようにする。

効果

全体の視点で既存システムの要件を見える化したため、新システムの追加による既存システムへの影響が明確化され、障害の連鎖回避、影響の極小化などが実現できた。

教訓

老朽化した既存システムは、仕様・制限等が不明確になっていることがある。既存のバッチ処理システムに対しフロントにオンラインシステムを追加する場合や、フロントのオンラインシステムを機能拡張する場合は、制限値を超えたときのチェック方法や、既存のバッチ処理システムが異常終了しないかどうかについて、両者の整合性を十分確認する必要がある。

3.19 RDBMS のクエリ最適化機能に関する教訓（T19）

教訓 T19

リレーショナルデータベース（RDBMS）のクエリ自動最適化機能の適用は慎重に！

問題

A 社の営業拠点は全国に約 1,000 カ所存在する。この営業支援システムは全国地域を数個のブロックに分けて順次リリースを行いシステムリソースの増強を行いながら全国規模で順調に稼働していた。

営業拠点の活動は、当日訪問する既得意あるいは新規の顧客に対して提案資料を営業拠点で朝一番にオンライン作成・印刷しパンフレット等と同梱して営業に出発する形態が多い。オンラインシステムは朝 7 時に起動され、8 時から 9 時 30 分が利用のピークとなる。ピーク時には約 50 件／秒のトランザクション処理が発生していた。ある朝、オンラインシステムのタイムアウトが多発し運用監視コンソール端末にトランザクション異常終了のメッセージが連続的に出力された。利用者はシステムからの印刷物が得られず営業活動に出られない状況となった。図 3.19－1 にシステム構成の概要を示す。

障害の発生状況は次のようであった。

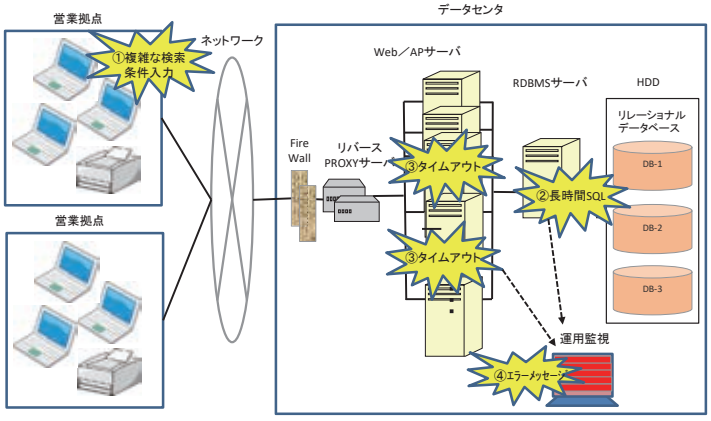


図 3.19－1　A 社営業支援システムの概要

当日 1 件目のトランザクションにおいて利用者から広範囲にわたる検索条件入力が投入された（図 3.19－1①）ため、DB サーバで稼働するリレーショナルデータベース（RDBMS）の自動最適化機能により全件フルスキャンを行う SQL 実行計画が適用され RDBMS サーバ内で SQL 処理が長時間実行されたものである（図 3.19－1②）。これにより、アプリケーションサーバの SQL 監視タイマのタイムアウト（3 分）となりアプリケーションが異常終了した（図 3.19－1③）。また、以降のトランザクションも同じ SQL 実行計画が適用され続け、タイムアウトが連続的に発生し運用監視コンソール端末に大量のエラーメッセージが出力され続けた（図 3.19－1④）。

原因

RDBMS は DB のデータ分布状態の統計情報と起動後最初に実行される SQL の内容により最適と判断した SQL 実行計画を選択する（クエリ最適化機能）。統計情報取得は日次夜間バッチによるデータベース再構築後に毎日実施している。

図 3.19－2 はデータの変動と RDBMS 性能（レスポンスに要する時間）の関係を表したものであり、一般的にデータベースのデータ件数増大にともなって徐々に性能（レスポンスに要する時間）は劣化する。

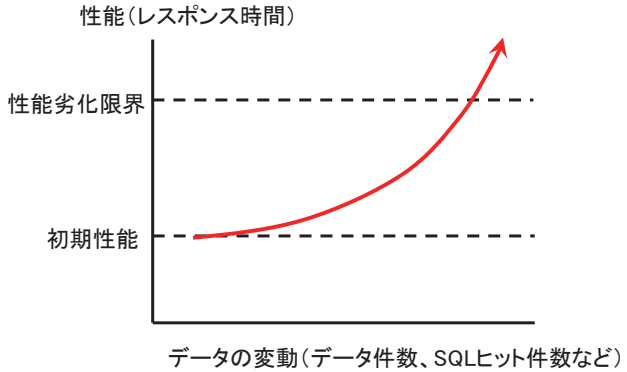
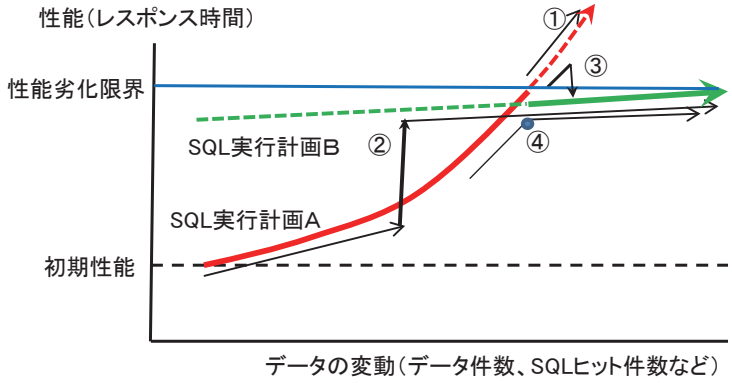


図 3.19－2　データ変動にともなうレスポンスの劣化

クエリ最適化機能による SQL 実行計画の切替えの概要を図 3.19－3 に示す。



①SQL実行計画Aが切り替わらない場合（徐々に性能劣化限界へ）
②性能劣化限界以前にSQL実行計画AからBに切り替わったケース（突然性能劣化発生）
③性能劣化限界を過ぎた後でSQL実行計画AからBに切り替わったケース（徐々に性能劣化限界へ）
④性能劣化限界でSQL実行計画AがBに切り替わったケース（理想）

図 3.19－3　データの変動にともなう SQL 実行計画の自動切替

RDBMS の SQL 実行計画は最適化機能によりデータの変動に応じて選択されるが、図 3.19－3 に示すように性能劣化限界点で瞬時に切り替わるものではなく、統計情報と SQL のヒット件数で危険域を予測し事前に切り替わることが多い。今回は、最初のトランザクションの SQL ヒット件数が大量となることにより、フルスキャン型の SQL 実行計画 B が選択されたと推察できる。つまり図 3.19－3 の②のケースが最初の SQL 実行計画選択時に発生したと考えられる。

その後、2 件目以降のトランザクションにおいても SQL 実行計画Aに戻らず続行されたため、通常のトランザクションもフルスキャン型となり長時間処理、タイムアウト多発に至った。

根本的には RDBMS のクエリ最適化機能の特性と利用者の使い方に関する分析が不足していたことが原因としてあげられるが、十分な分析を事前に行うことは難しく、同様の障害が他のサイトでも発生することが多い。

対策

データセンターでは緊急措置としてオンラインサービスを停止し、RDBMS を再起動して回復したがこの間はサービスが中断となった。

翌日以降の当面の運用対処として早朝にアプリケーション担当者がオンライン起動後の最初のトランザクションを通常のインデックスを使用する検索条件で投入し SQL 実行計画Aが選択されるようにした。

抜本的な対策として、表 3.19－1 に示すようにインデックスを追加しフルスキャンが発生しないようにする、あるいは SQL 実行計画を固定する（統計情報を取得しない）ということが考えられるが、A 社は更新の負荷増大とクエリ最適化機能を無効化することのメリット・デメリットを比較しインデックスを追加する方策を 1 カ月後にリリースした。

表 3.19－1　SQL 実行計画の変更による性能劣化発生への対策

案	対策案	メリット	デメリット	備考
1	インデックスを追加	クエリ最適化機能は活用できる	更新負荷の増大による性能劣化のおそれあり	データ量の変動が予測される場合
2	SQL 実行計画の固定化（利用比率の高い SQL のみ）	案 3 に比べて効率的	利用比率が均等だと効果は小さい	システム全体に影響させたくない場合
3	すべての SQL 実行計画を固定化	SQL 実行計画の自動切替は発生しなくなり極端な劣化は発生しない。	データ量の変化に追随しないため高速処理は望めない。	極端に遅くなるのを防止したい場合

効果

抜本的な対策を実施後は、SQL 実行計画 A が常に選択され極端な性能劣化は発生しなくなった。

教訓

RDBMS の最適化機能はデータ変動と SQL のヒット件数を予測しながら SQL 実行計画を選択するため、必ずしも臨機応変に最適な選択が行われるわけではない。ユーザの利用要件を十分に分析し、データベース設計時にインデックスの付け方、SQL 実行計画の選択方法（SQL 実行計画を RDBMS の最適化機能に任せる、あるいは固定化し最適化機能を使用しない方式にするか）、また統計情報取得のタイミングと頻度（リリース時のみの取得にするか、あるいは定常運用で行う場合は頻度をどうするか）について検討しリスクヘッジすることが肝要である。このためにはアプリケーション担当とインフラ担当（データベース担当）の協力が欠かせない。

3.20 パッケージ製品の機能カスタマイズに関する教訓（T20）

教訓 T20　パッケージ製品の機能カスタマイズはリスクを認識し特に必要十分なチェック体制やチェック手順を整備して進めること

問題

A 社は 24 時間 365 日コールセンター受付システムを運用している。コールは会員からの作業員派遣要請が主でその内容により最寄りのサービス拠点から作業員が現場に駆け付けて対処するものであり、1 日のコール数は平均約 3,000 件であり連休の際には特に集中する。

連休中のある日の 12 時頃、オペレータが受付端末のタッチパネルが反応しなくなっていることに気づいた。また利用者からコールがつながらないとのクレームも数件寄せられていた。本事象は同じ構成を持つ交代系システムへの手動切替えを実施し回復するまで約 10 分間継続していた。この間のコール数は不明だが、直前の 10 分間では 20 件のコールが発生していた。

このコールセンター受付システムは新システムに更改してから 1 週間連続稼働していた。新システムは手作りで構築された旧システムと異なり、ベンダのパッケージ製品ソフトを導入し構築されたが、A 社の要求仕様に対応するためベンダは機能追加[*1]のカスタマイズを実施していた。

システムの構成の概要は以下のとおり（図 3.20－1）。

- 電話系の呼処理を制御するデジタル PBX サーバ
- 電話コールに連動して受付端末の画面を制御する CTI サーバ
- 各サーバは現用系と交代系の二重化構成

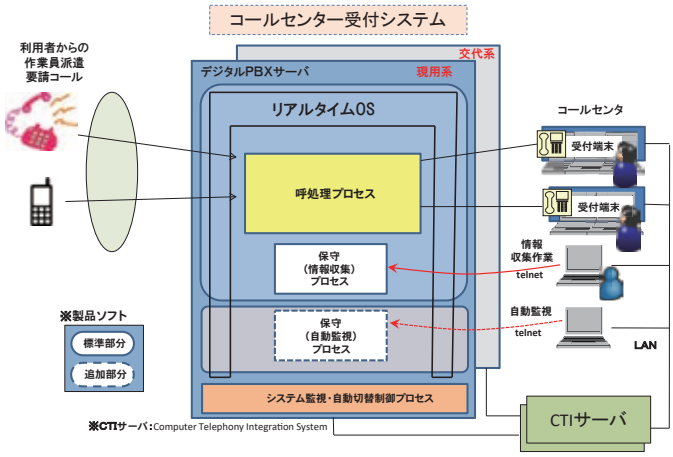


図 3.20－1　コールセンター受付システムのプロセス構成概要

[*1] telnet による呼処理プロセスの稼働状況を自動監視する機能、デジタル PBX サーバ内に実装（図中の「保守（自動監視）プロセス」）

原因

デジタル PBX サーバを交代系に手動切り替えし回復させた後で、ハングアップ状態であった現用系のシステムダンプを取得し解析した結果、保守員が行った障害情報収集のための telnet 接続と呼処理の稼働状態を自動監視するための telnet 接続が競合し無限ループ状態に陥ったことが原因であると判明した。また、システムの異常を検知して交代系システムに自動的に切り替える機能もソフトウェアの不具合から無限ループを検知できなかったため作動しなかった。障害の発生状況を図 3.20－2 に示す。

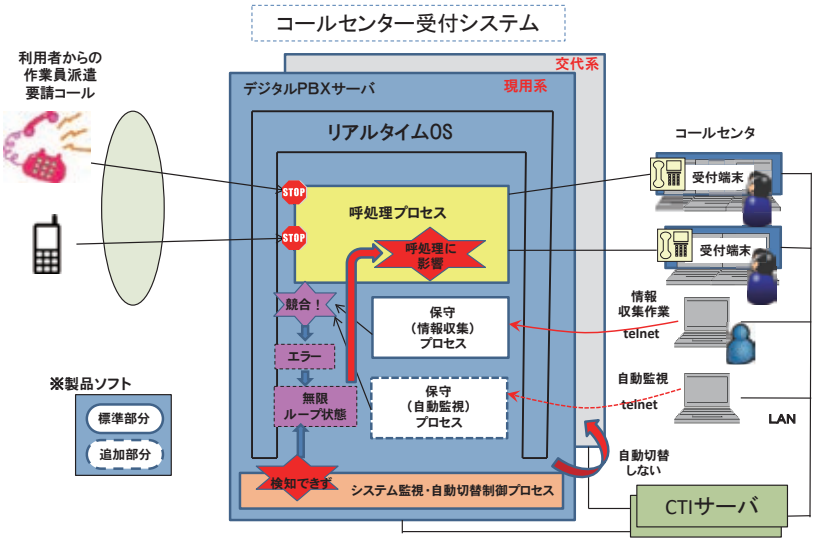


図 3.20－2　コールセンター受付システムの障害状況

調査の結果、telnet の競合は希少なタイミングで発生するコールセンター受付システムのファームウェアのプログラム不具合と判明した。また、交代系への自動切替えが出来なかった事象は自動切替え制御機能の不具合が原因であるとベンダから報告された。

なお、本システムの構築にあたりユーザである A 社から提示された要求仕様を満たすために、ベンダは同業他社でも広く利用されているパッケージ製品ソフトをベースにカスタマイズを実施したが A 社に十分な説明がされていなかった。

対策

本システム障害事例でとられた対策を以下に記す。

① 直接原因と復旧措置

コールセンター受付システムのファームウェアの不具合によるシステムハングアップがサービス停止の直接原因であった。交代系への手動切替え実施により復旧し、受付サービスを再開した。

② プログラム不具合への対応

問題を起こしたファームウェアのプログラム修正を行い、再現テストを繰り返して検証し 4 日後に本番環境に適用した。

③ 2つの telnet 接続の競合への対策

機能追加した呼処理自動監視プロセスへの接続は telnet プロトコル方式をやめて、RS232C 回線による接続方式に変更し競合が発生しないように対処した。

④ FTA 分析[11] を活用した総点検

同様の競合発生個所の有無について、FTA 分析手法を活用した総点検を実施し、同様の不具合を数件検出、対策計画を策定し、緊急性に応じて順次対策を実施している。

効果

前述の対策により、障害は再発せず順調に稼働している。また、telnet 接続を本来のパッケージ製品ソフトの仕様どおり 1 本にしたことから、同種の製品を使用している同業他社と同様の信頼性を確保したと考えられる。

教訓

今回の障害はシステムを刷新し、稼働を開始した直後の初期トラブルであるが、非常にまれなタイミングで発生する競合が引き金になったものであり、事前のテストでは検出が困難であった。同業他社でも使用されている信頼性の高いパッケージ製品ソフトを採用していたが、呼処理の自動監視をする要求仕様に対応するため、このパッケージ製品ソフトに機能追加を実施しており、そのプログラム不具合が競合発生の原因となった。ベンダは発注者に対してこの機能追加にともなうリスクを共有し、改造個所の処理方式を説明し、必要十分なチェック体制とチェック手順を整備して進めることが重要である。

[11] FTA 分析（Fault Tree Analysis: 故障の木解析）

3.21 運用保守で起こる作業ミスに関する教訓（T21）

教訓 T21　作業ミスを減らすためには、作業指示者と作業者の連携で漏れのない対策を！

問題

A 社は、顧客の集荷依頼の電話をその顧客の所属するエリアの該当集荷本部へ転送するサービスを行っている。A 社は、B 社への集荷依頼を関係外の C 社集荷本部に転送していたことを、C 社からの連絡で知った。詳しく調べたところ、4 回に 1 回の割合で B 社への集荷依頼を C 社集荷本部に転送しており、C 社が B 社へ電話転送することで対処していたものの、現場は混乱し、集荷作業漏れが多発し、顧客からの苦情が殺到していたことが判明した。

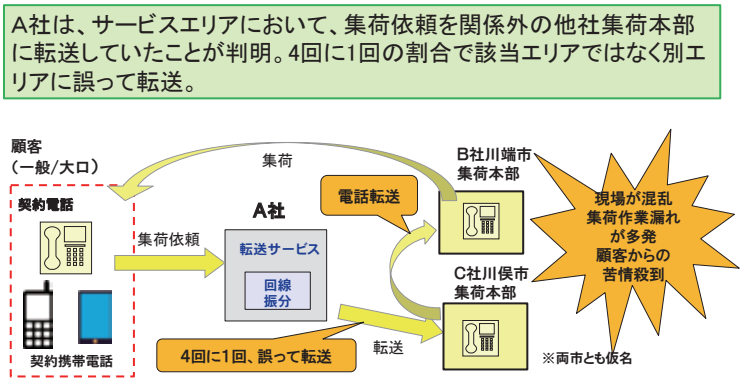


図 3.21－1　障害状況

原因

システム運用保守は、様々なソフトウェア、ハードウェアの変更作業をサービス中に行わなければならない場合があり、そのような状態では作業ミスが起きやすくなる。

障害の直接原因は、作業者の些細な誤りであったが、根本的には、誤りを見逃しやすい作業環境と最後の砦となるべき作業指示者の確認不足によるものであった。

【直接原因】

A 社は、コール数の増加に対応するため、ゲートウェイ（GW）の GW#3, #4 の増設作業を行った（図 3.21－2 ①）。作業指示者から依頼された作業者（図 3.21－2 ②）は、エリア情報管理サーバの GUI 画面から転送先データの登録作業を実施した。しかし、GW#3、GW#4 とも転送先名「KAWAHATA（カワハタ）」を「KAWAMATA（カワマタ）」と誤って設定してしまった（図 3.21－2 ③）。

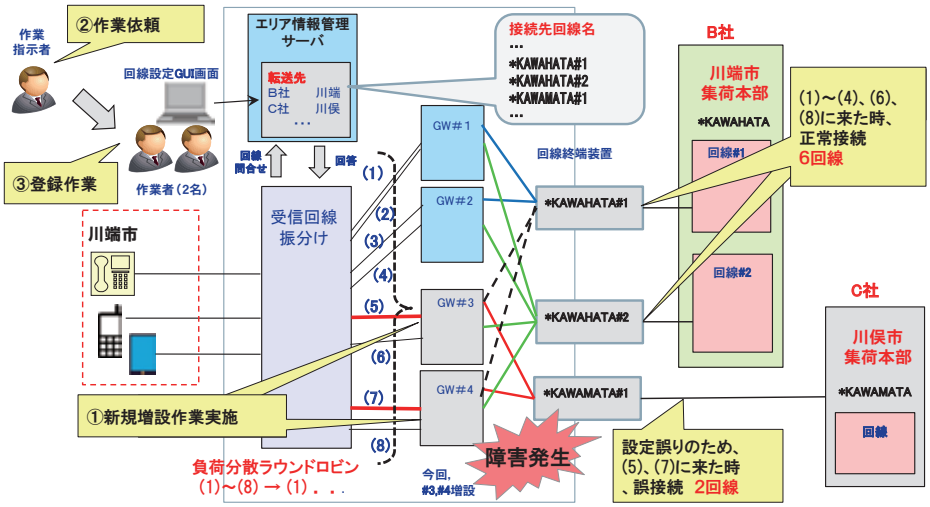


図 3.21－2　障害原因

【根本原因】

根本原因は、「作業者が作業ミスを起すような状況、環境に置かれた」ことと、「作業指示者が、作業者の実行結果をきちんと確認していなかった」ことである。以下、根本原因を、作業者と作業指示者に分けて示す。

◆作業者の観点からの根本原因

① パラメータがローマ字で見誤りやすかった。

作業は 2 人体制で行っていたが、操作担当者が「KAWAMATA（カワマタ）」を誤って「カワハタ」と読み上げたため、チェック担当者は「OK」としてしまい、操作担当者は、そのまま「KAWAMATA（カワマタ）」と誤って登録してしまった。

海外ベンダ製の管理ツールの GUI 表示がアルファベット順のローマ字表記のため、読み間違いが起こりやすい状況であった。

② 設定に関係しないパラメータが表示されていた。

管理ツールの GUI 表示に、B 社の設定と関係のない C 社のエリアまで表示されていたため、設定すべきエリアを見落としやすい状況にあった。

③ 設定結果を自ら再確認する手順が抜けていた。

作業者は、サービス中に直接本番環境で変更作業を行い、集荷依頼コールを送っていた。

このとき、作業者は、テスト環境がないため本番環境で変更作業を行わざるを得ず、さらに、作業者が設定作業を終えた後自分自身でその内容を確認する手順が抜けていた。

◆作業指示者の観点からの根本原因

① 設定結果を作業指示者が確認する手順が抜けていた。

作業者が本番環境で設定作業を終えた後、作業指示者は、作業実施チェックリストや、作業手順書、作業ログなどを使って、その設定作業の結果を確認していなかった。

② 相手との関係で、全ルートの確認テストができなかった。

作業者は、設定後に 2 回のコールで集荷本部への 2 回線が正常に繋がることを確認したのみであり、全振分けパターン（8 回線）をテストしていなかった。

これは、作業指示者が、「集荷本部側は、本番の集荷依頼コールが優先であり、テストコールのために集荷依頼コールを受けられない時間が増えることに抵抗がある」ことを気にして、2 回のコールでも良いことにしてしまったことによる。

対策

それぞれの根本原因から、対策を立てた。

◆作業者の観点からの対策

① パラメータがローマ字で見誤りやすかった。

【対策 1】 このゲートウェイ (GW) は海外機器メーカ製のため、漢字などの表示はできない。そのため、登録時の入力確認のための読上げは、アルファベットを「K（ケー）A（エー）W（ダブル）A（エー）...」と 1 文字ごとに区切って読み上げることにした。

② 設定に関係しないパラメータが表示されていた。

【対策 2】 エリア単位の設定画面に当該エリアのみ表示するようツールの改良を機器メーカに依頼した。

③ 設定結果を自ら再確認する手順が抜けていた。

【対策 3】 作業者が、設定作業後、設定誤りを未然に発見できるように、「設定変更結果を CSV 出力し、変更前／後の差分チェックを行う」ルールを運用手順書に明文化した。

【対策 4】 テスト環境と本番環境との 2 面環境とし、テスト環境での確認後、その設定をそのまま本番移行できるような仕組みの実装を機器メーカに依頼した。

【対策 5】 テスト時の接続ルートの可視化として、実回線を接続せずにテストが可能な「疑似確認ツール」の強化を機器メーカに依頼した。

◆作業指示者の観点からの対策

① 設定結果を作業指示者が確認する手順が抜けていた。

【対策 6】作業指示者は、作業者が作業終了した後の作業結果を、作業実施チェックリストや、作業手順書、作業ログなどによって確認する手順を運用手順書に追加した。

② 相手との関係で、全ルートの確認テストができない。

【対策 7】設定作業を行った場合、全ルートの確認を行うプロセスを相手と合意して実施することに決め、その旨を各社に申し入れた。

効果

作業指示者が、運用保守作業に責任を持った取り組みを行うことで、作業指示者と作業者の連携が生まれ、作業ミスを減らすことができる。このような連携は、関係者の合意形成を築きやすい。さらに、このような取り組みの蓄積が、ノウハウの蓄積に繋がる。

教訓

作業ミスは、ともすれば作業者の自覚の問題とされる場合が多い。しかし、作業者の観点から、個人、環境、ハードウェア、ソフトウェアの視点で、作業ミスの原因、対策を考えることが重要である。

さらに、作業には、必ず作業指示者が存在する。その作業指示者が、作業結果に責任を持った行動を取ることが作業ミスを減らす上で重要な役割を持つ。作業指示者は、障害の責任を作業を間違えた作業者、ソフトウェアのバグ、ハードウェアの故障などに持っていかず、システムの問題を仕組みや組織として改善することに主眼を置くことが重要である。

3.22 バッファプールの管理に関する教訓（T22）

教訓 T22　隠れたバッファの存在を把握し、目的別のしきい値設定と超過アラート監視でオーバフローを未然に防止すること

問題

A 社は 24 時間 365 日予約受付システムを運用している。複数種類の業務の予約を同時に受け付けており 1 日の受付数は平均約 10,000 件／日、ピーク日は 30,000 件／日である。

ある日の夜間、業務予約処理 A のバッファが処理待ちで滞留し、予約受付システムからの転送が停止していた。暫くすると、通常処理できていた予約処理 B と C も受付ができなくなり、すべての予約処理の受付が停止する状況となった。A 社の予約受付システムの構成概要と問題の発生状況は図 3.22－1 のようになっている。

処理概要を以下に示す。

- 稼働系と交代系のホットスタンバイ構成で交代系に切り替えてもバッファの内容は引き継がれる
- インターネットからの受付処理により、チェック・登録された入力データは共通バッファに入れられる
- 共通バッファから振分処理で随時入力データを仕分けし、予約処理ごとの転送バッファに格納する
- 各転送処理は転送バッファに登録された入力データを予約処理サーバに転送する
- 予約処理サーバは転送された入力データで予約データベース（以下、予約 DB）を更新し、インターネットを通して回答を端末に返却する。

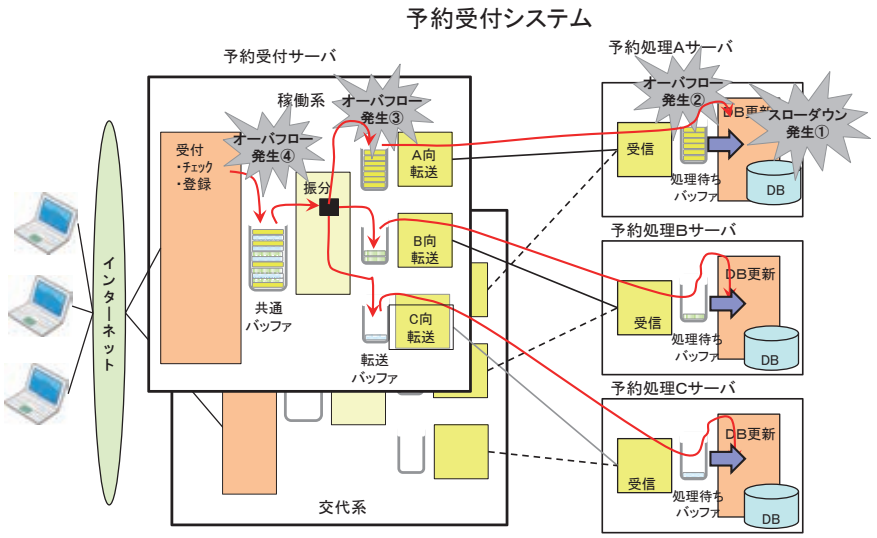


図 3.22－1　システム概要と障害の発生状況

原因

予約処理 A の受付はこの日から開始されたものであった。予約処理 A サーバの処理がメモリ不足により、スローダウン状態となった（図 3.22－1①）ため、処理待ちバッファに徐々に入力データが滞留し、上限の 1,000 件を越えた（図 3.22－1②）。これにより、予約受付サーバ側の転送バッファに未送信データが滞留して満杯となり（図 3.22－1③）、共通バッファから予約処理 A の入力データが取り出されなくなり、共通バッファも満杯となり（図 3.22－1④）全予約処理の受付が停止した。スローダウンが発生してから、受付が停止するまで 30 分程度の時間差があった。

この障害発生過程において、予約受付システムの共通バッファが満杯になった段階で、すべての予約受付業務を停止するメッセージが表示されたため障害が発生していることが判明した。

予約受付システムは、予約処理の種類ごとに受付中／受付停止を設定することは可能であったが、問題が起こった予約処理 A のみを受付停止する前に全予約業務停止となってしまった。さらに、交代系に切り替えてもバッファの内容は引き継がれるため、このケースでは復旧できなかった。

対策

① 復旧措置

予約業務 A サーバのスローダウン状態を設定変更により回復させ、滞留している入力データを順次処理していくことにより、2 時間程度かけて共通バッファを空にした。この後、予約受付システムを再開し、通常運用に戻った。

② バッファオーバフロー状態の検知

障害状況を早期に検知できなかったシステムの問題に対しては、各バッファの蓄積状況を監視し、警戒レベルに達したら監視コンソールにアラートを表示するようシステムの改善を実施した。

効果

バッファへの入力データの滞留状況を監視し、異常な状態を早期に検知することでシステム障害の発生を未然に防止することが可能になった。

一般的に情報システムはメッセージをプロセス間でバッファを用いて受け渡すバケツリレー方式となっていることが多い。（図 3.22－2）

よって受け取り側のプロセス処理が遅れたり、停止したりするとバッファにメッセージが滞留していくことになる。システムが良好なパフォーマンスを維持しているかどうかはこのバッファの滞留状況を監視し適切な状態であることを評価することで判断できる。

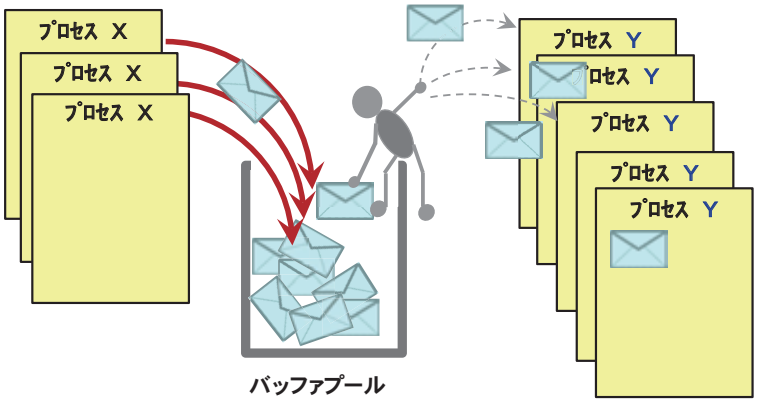


図 3.22－2　バッファを経由したプロセス間通信

このバッファがオーバフローすると、プロセスＸからのメッセージ登録ができなくなりシステム障害となることが多い。よって、バッファの蓄積状況を監視し、一定のしきい値を超えたら監視コンソールにアラート情報を表示することが有効であり、しきい値は注意レベル、警告レベル、危険レベルのようにレベルを分けアラート情報の緊急度も合わせて表示することが望ましい。

なお、しきい値の設定にあたっては、プロセスＸとプロセスＹの連携度合いにより適切な値を検討する。

リアルタイムに近いメッセージ連携形態の場合は数件蓄積されただけでも異常が発生している可能性があるが、メッセージの登録が時間帯によってピーク性がある場合には 70% ～ 80% 程度まで許容範囲とみなすことができるので、それぞれの適切なタイミングでアラートを表示するよう設計を行う。

また、各バッファのサイズ（最大収納件数）は構成情報として管理し、稼働状況や業務要件の変化等に応じて見直しを行う。

通常の１日単位で運用されるシステムでは、システム開始時にバッファは空であり、稼働中のバッファは増減を繰り返すが、システム終了時に空に戻る、あるいは終了時点で蓄積データが存在した場合には後処理で対応を行う。しかし、24 時間連続稼働のシステムではこのバッファは常に増減をしているため、管理には特に注意し、オーバフローを未然に防止しなければならない。

また、パッケージ製品を利用する場合はバッファが明示されないことがあるため、どこに何のバッファが存在しているかを把握することも必要である。

教訓

本システムのような共通バッファのオーバフローに起因するシステム障害から得られる教訓は以下のとおりである。

○各種バッファの存在を認識し、構成情報としてその最大収納件数を認識・管理する。
　特にパッケージ製品を利用する場合は明示されていない内部バッファの存在を把握しておく必要がある。

○用途別にしきい値を設定し、しきい値を超えた場合にアラートを表示することでオーバフローによるシステム障害を未然に防止する。

3.23 障害監視機能のあり方に関する教訓（T23）

教訓 T23

障害監視は、複数の観点から実装し、障害の見逃しを防げ！

問題

A 社の基幹システムは 24 時間 365 日稼働のオンラインシステムであり、業務の特性から瞬時の停止も許されない。DB サーバは 4 重化されており、それぞれ障害監視機能を持っている。

ある日、DB サーバ 4 台すべてが順番に停止した。

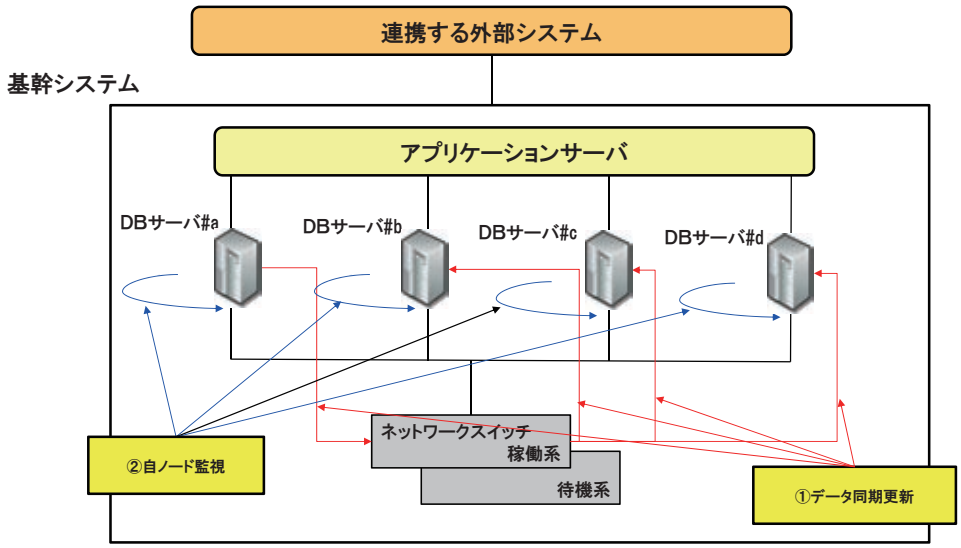


図 3.23－1　DB サーバの障害監視

DB サーバの障害監視には 4 つの機能が用意されていたが、今回の障害に関係した機能は以下の 2 つであった（図 3.23－1）。

（1）データ同期更新

DB サーバは 4 台の物理サーバでデータ同期機能を構成しており、1 台の DB サーバでデータ更新があれば、DB のバッファ・キャッシュと呼ばれるメモリ領域にあるデータをコピーし、ネットワークスイッチを介在して、他の DB サーバにデータ同期更新を実行する。このデータ同期更新処理がタイムアウトした場合、DB ミドルウェアに付随する監視プロセスから監視サーバに通知を行う。監視サーバはこれを受けて、データ同期更新の発信側の DB サーバを停止させる（図 3.23－1 ①）。

(2) 自ノード監視

これは、各 DB サーバが自ら稼働しているかどうかを監視する機能である。各 DB サーバが、サーバ OS の障害保護機能を使って自分自身に SQL を投入し、返信の有り無しで死活を調べる監視を 45 秒ごとに行っている（図 3.23－1 ②）。

原因

サーバ停止の直接の原因は、ネットワークスイッチのキャッシュメモリ故障であった。このスイッチは、本件に関してエラーメッセージを出力しなかったので障害検知ができなかった。以下、全 DB サーバが停止した経緯を示す（図 3.23－2）。

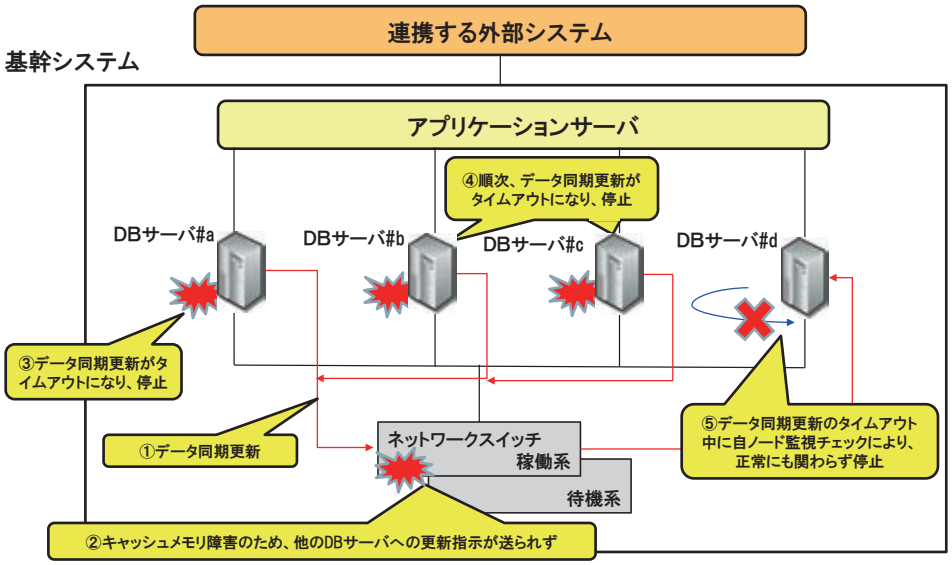


図 3.23－2　障害経緯

DB サーバ #a にデータ更新が行われた際、DB サーバ #a はデータ同期更新機能によって他の DB サーバ 3 台との間でセッションを張り、DB サーバ #a から他の DB サーバ 3 台へデータ更新の送信を行った（図 3.23－2 ①）。この時点でネットワークスイッチが正常動作しておらず他の DB への同期更新が正常に行われなかったため（図 3.23－2 ②）、DB サーバ #a と受信側の DB サーバがタイムアウトになった。監視機能は DB サーバ #a が DB 更新エラーになった通知を受け取り、DB サーバ #a を止めた（図 3.23－2 ③）。DB サーバ #b 、#c でも DB サーバ #a と同様の DB 更新エラーが発生し、順次停止していった（図 3.23－2 ④）。

ネットワークスイッチの障害だけであれば、DB サーバ #a、#b 、#c が停止した時点でデータ同期更新機能は稼働しないため、最後の 1 台の DB サーバ #d で運用が継続できるはずであった。しかし、タイミング悪く DB サーバ #d は、停止する直前の DB サーバ #c のデータ同期更新中に自ノード

監視機能の SQL を投入したため、この SQL がデータ同期更新のタイムアウトに巻き込まれてタイムアウトになってしまい、DB サーバ #d も停止してしまった（図 3.23－2 ⑤）。

このように、2 つの監視機能（データ同期更新、自ノード監視）の実行タイミングの重複によって全 DB サーバが停止してしまった。

このような障害が発生した根本原因は、ネットワークスイッチを含んだ DB サーバの障害監視が適切でなかったことである。以下にその原因を示す。

【原因 1】 ネットワークスイッチの死活監視でつかめないエラーの発生

本来であれば、稼働系スイッチが障害を起こせば、待機系スイッチに切り替えるのだが、今回は、稼働中のネットワークスイッチが完全に動作を停止しておらず、障害メッセージを出力しなかったため、監視機能でエラーを摘出できなかった。

ネットワークスイッチは、「動くのが当然」と思っていたため、設計段階でスイッチの観点でのリスク抽出が十分網羅されておらず、十分な対応策が練られていなかった。そのため、せっかく冗長構成における予備系への切替え機能を用意しながら、実行することができなかった。

【原因 2】 DB サーバの監視機能の不整合

データ同期更新と自ノード監視が同時に起動したときのエラー判断に考慮漏れがあった。このパターンに対応できていれば、ネットワークスイッチの障害（データ同期更新ができない）によって 4 台の DB サーバの内 3 台が停止したとしても、1 台で運用を継続できた。

さらに障害原因を調査すると、A 社のシステム部門では、業務部門から「瞬時の業務停止も許されない」との過酷な要件を課せられていた。点検内容については製造ベンダも含めて設計を行っており、システム部門が必要と判断した保守作業は業務要望よりプライオリティを上げて対応を行っていた。しかし、今回の障害を踏まえ、敢えて以下の点を指摘したい。

【原因 3】 保守（点検作業）不足

ネットワーク機器の点検の実施範囲が最小限に抑え込まれており、十分ではなかった。障害を起こしたネットワークスイッチは、メーカ推奨に基づき、保守点検を行っていたが、電源 off/on、稼働系 / 待機系の入替えなどは実施していなかった。

対策

直接の原因であるネットワークスイッチは、交換することにより対応した。

根本原因としてあげた「ネットワークスイッチを含んだ DB サーバの障害監視が適切でなかった」ことに対する対策をまとめると以下のようになる。

【対策 1】個々の機器の監視を複数のツールで行う

今回のように監視メッセージが出力されないため障害を見逃すことがあることを想定し、複数の観点から監視機能を検討し、実装する。予備機への切替えは、いずれの監視機能でも検知し実行できるように設計する。

【対策 2】複数の監視機能の組み合わせで動作に問題がないか確認する

この障害の対策としては、データ同期監視が完了してから自ノード監視の判定を行うように、自ノード監視機能に、「SQL 投入からの死活監視のデータのタイムアウト待ち時間を延ばす」、「リトライを数回行う」などの対策を行う。これにより、データ同期監視機能と自ノード監視機能の不整合を解消する。さらに、他に同様な問題がないか監視機能の組み合わせ確認を行い、テストを実施する。

【対策 3】十分な保守時間の確保

システム部門は、業務部門との調整を行った上で、システムの安定稼働のための保守時間を確保する。その保守時間を使って、ネットワークスイッチの電源 off/on、パッチの適用などの定期保守点検を十分に行う。さらに、バックアップ切替え確認、正副機器の切替えによる入替えなど、日ごろ待機系になっている機器を動作確認する、あるいは稼働系として使うなど行う。

効果

ネットワークスイッチなどの機器の監視は見逃される傾向があるため、そのスイッチが障害になると重大な影響を及ぼす。システム部門は、ネットワークスイッチを含めた複数の監視方法を実装し、監視漏れがないように、またそれぞれの監視機能に矛盾が生じないようにすることで、障害を減らすことができる。

また、業務部門は「瞬時の業務停止も許さない」運用を行うことを求める傾向があるが、今回のような事例を考えれば、システム部門は、十分な保守時間の確保を業務部門に提案することができる。これにより、システム障害が減り、トータルのサービス提供時間が増えることになる。

教訓

ネットワーク機器が高度になり、接続される機器も増えている中、システム障害対策も複雑になっている。そのため、監視技術が重要度を増している一方で、監視の不具合による障害も増えている。

この事例は、監視ミス（自ノード監視がエラーでないものをエラーとした）と監視漏れ（ネットワークスイッチのエラーを見過ごした）が同時発生した事例である。

教訓は、「障害監視は、複数の観点から実装し、障害の見逃しを防げ！」とした。

3.24 障害中の運用に関する教訓（T24）

教訓 T24

サービス縮退時の対策を考慮せよ

問題

A社の統合システムは、基幹業務と情報提供業務をメインとして様々な対外システムやインターネットと連携したシステムである。

ある日、統合システムの3台でクラスタリング構成を組んだDBサーバが順次停止し、全DBサーバが停止した。原因が判明しなかったが、1台だけ稼働させたところ（図3.24－1①）、性能的に問題があるもののサービスを続けることができた。そこで、情報提供業務を停止させて基幹業務だけに専念させてサービスを継続させた（図3.24－1②）。

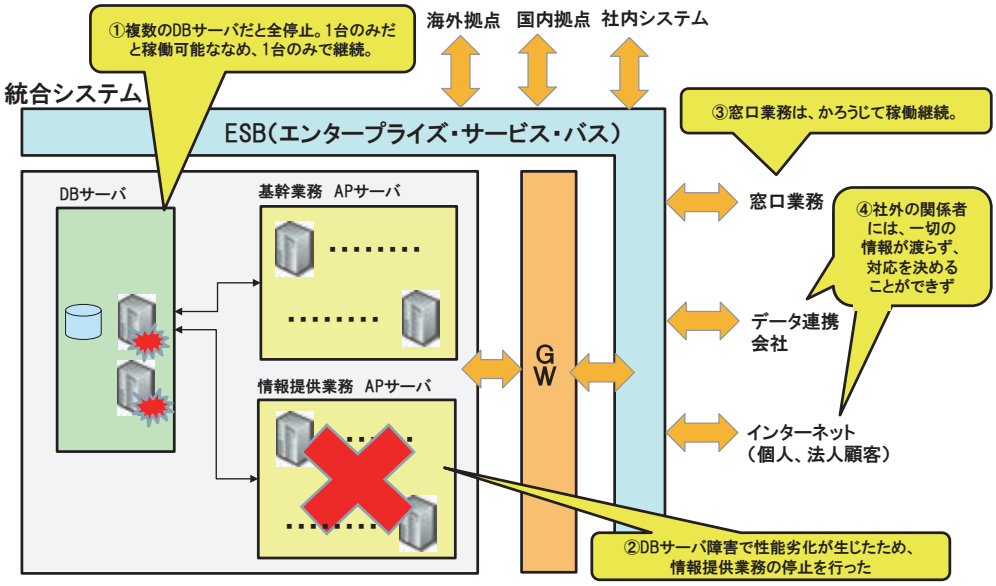


図3.24－1　障害状況

A社の窓口では、システム部門と状況を確認しながら業務を実施することができた（図3.24－1③）。しかし、データ連携会社は、情報提供を受けられなかったため、自社の業務を進めることができなかった。さらに、インターネットからの法人顧客や個人顧客も情報提供を受けられなかったため、直接電話でA社の窓口に状況を確認するしか方法がなかった（図3.24－1④）。そのため、A社の窓口も外部からの対応に追われてしまい、業務の遅れが徐々に大きくなり、混乱は終日続いた。

原因

統合システムの停止に至った直接原因は、DB サーバのソフトウェアのバグであった。

障害の影響が拡大したのは、情報提供業務を稼働させることができなかったためであった。全 DB サーバが停止したが、その後 1 台だけ稼働させ基幹業務を稼働させることができた。しかし、情報提供業務は、性能上の問題から停止せざるを得なかった。これは稼働当初からの懸案で、当時 1 台で性能要件を満たせる高性能のサーバが製品化されていなかった。

当初統合システムは、基幹業務が中心であった。数年後に情報提供業務を追加することになったが、あくまで付属的な意味合いで考えていた。そのため、大幅な業務要件の見直しを考えていなかった。

そのため、アプリケーションサーバは基幹業務アプリケーションサーバと情報提供業務アプリケーションサーバに分割していたが、DB サーバについてはひとつのインスタンス（1 論理 DB）構成としたままであった。

しかし、インターネットやスマートフォンを活用した業務へのニーズが増すにつれて、データ連携会社やインターネット（個人、法人顧客）に対する情報提供業務の全業務に占める割合は大きく拡大し、多くの外部関係者に重要なサービスを提供することが業務の中心になっていた。

今回のシステム障害では、統合システムが障害になりサービス縮退となった場合、対外システムやエンドユーザに「障害情報と復旧見込み」などの情報を発信することができなかったことが、システム障害の影響を大きくしてしまった。

根本原因は、サービス縮退が長期化することによって、問題が大きくなることを見逃していたことである。

対策

直接原因の DB サーバのソフトウェアのバグを改修した。また、業務制限を行わなくても 1 台で十分性能を満たせる DB サーバに入れ替えた。

さらに、根本原因で提示した「サービス縮退が長期化することによって、問題が大きくなること」を未然防止する対策を考えたい。

【対策 1】 エンドユーザの要求の変化を検証する

今回の事例における情報提供業務の追加のように、新システムを検討する場合は、お客様の要求変化を考慮し、基幹系業務とそれ以外の周辺システムを分けるなど、サービスの特徴、役割を考慮した設計を行う。そのために新システムを検討する場合は、要求定義の中で「お客様の要求の変化」を分析する。

その上で、設計時にリスク分析を行い、システム障害発生時のリスク低減策を講ずる。

【対策 2】ディペンダビリティ[13]を追求したシステム構成

【対策 1】の検証に基づき、今回のような事例においては、お客様の観点でシステム障害発生時の情報を提供できるように、業務の疎結合を考慮し DB サーバを基幹業務と情報提供業務に分離した DB サーバ構成とする。

その際、基幹業務サーバが障害のときに情報提供業務サーバで障害情報を提供するのはもちろんのこと、情報提供業務サーバが障害の場合でも、外部へ情報提供できる代替策を検討する（図 3.24－2）。

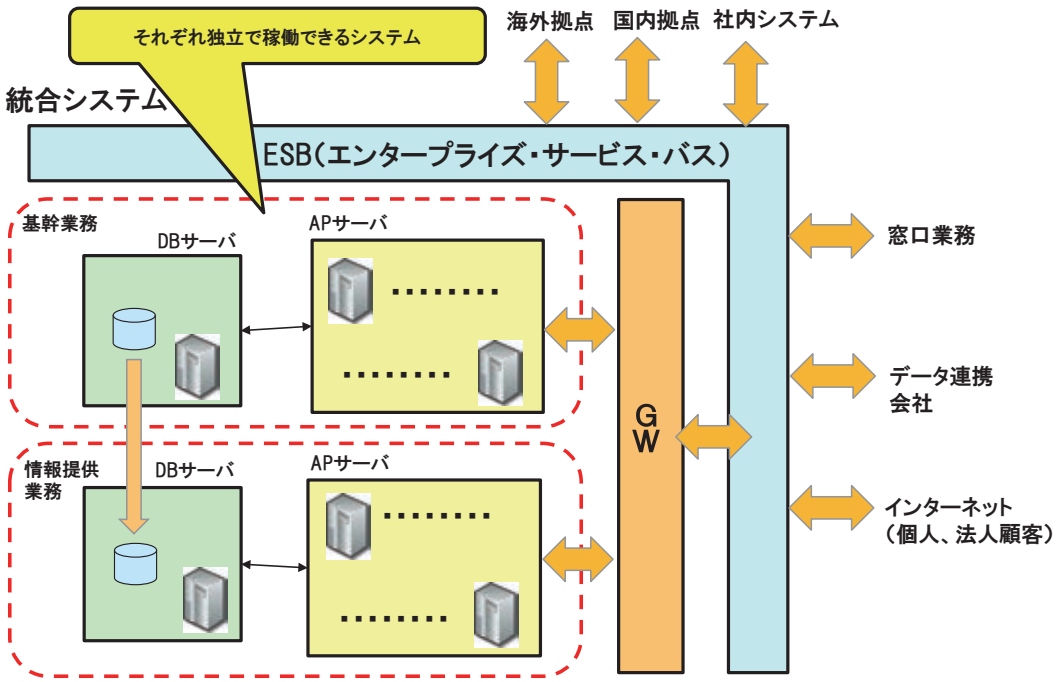


図 3.24－2　【対策 2】の一例

この事例のように途中で新たな要件が加わり、長年使用し続けているシステムのエンドユーザのニーズが変化することは、往々にして起こり得る。「システムを利用する顧客のどの要件に照準を当ててシステムを見直すか」は、重要なポイントである。

新たなシステムの要件を検討する場合は、例えば変化したステークホルダとの要件確認（あるいは合意）プロセスで、要件定義を再検討することも必要であろう。

[13] ディペンダビリティ（dependability）とは、例えば、システムの一部が壊れても残りの部分で補いながら、または機能を縮小させながら、稼働状態を保つといった自立的、自己修復的な動作を示す概念である。

【対策 1】と【対策 2】の対策をまとめると、「【対策 1】エンドユーザの要求の変化を検証する」ことにより、「【対策 2】ディペンダビリティを追求したシステム構成」のあり方が、方向づけられることになる。

効果

サービスの拡大にともない今まで付属的なシステムと思われていたものが、重要度を増すような変化がしばしば見られる中において、この事例のような「エンドユーザが要求する機能」の優先度も変化していくことがある。そのため、今までサービス縮退で停止しても問題にならなかった機能が、長時間停止することで問題になったりする。

前述した対策は、そのようなサービス縮退時の観点を見失わず、対策を検討するのに役立つ。

教訓

この事例では、設計時にリスク分析を行いディペンダビリティの確保に対応する最適なサービス縮退を考慮することにより、最適対策を取ることができる。

サービス縮退のあり方から、「システムの集中」が良いか、「システムの分散」が良いかなど、システム障害のリスクを軽減する構成を決めることも検討できる。

この教訓は、「サービス縮退時の対策を考慮せよ」とした。

3.25 原因不明障害への対応に関する教訓（T25）

教訓 T25

障害原因が不明でも再発予防と発生時対策はできる

問題

A 社の社内共通基盤システムは Web での社内向け／顧客向けの様々な情報公開や各従業員の仮想化 PC のドメイン管理などの機能を保有している。システムの各サーバは仮想化されており、外部連携用の基幹スイッチ、自社内連携用の集約スイッチ、仮想ネットワークスイッチにより機器間の通信を行っている。また、業務の特性上、システム障害発生時に銀行オンラインや航空券予約システムのような高レベルの即時回復は要求されない。

ある朝、業務開始後に物理 PC や社内共通基盤上に構築された仮想化 PC から使用していた業務画面からのレスポンスがなくなる現象が発生し始めた。Web で提供される各種サービスは動作せず、仮想化 PC はすべて動作しなくなった。物理 PC は単独では動作したが、ネットワークに接続する業務は動作せず、外部とのメールも送受信できない状態になった。

社内システム管理者は個別の業務ではなく、ネットワークを経由するシステム全体が動作しない状況からスイッチやストレージなどの基盤系処理装置の障害を疑い、各層のスイッチの調査およびコマンドによる再起動を実施したが業務時間終了までの回復が期待できず、17 時になって当日の業務再開を断念し全社に通知した上で、ハード／ソフト両面からの再調査に着手した。

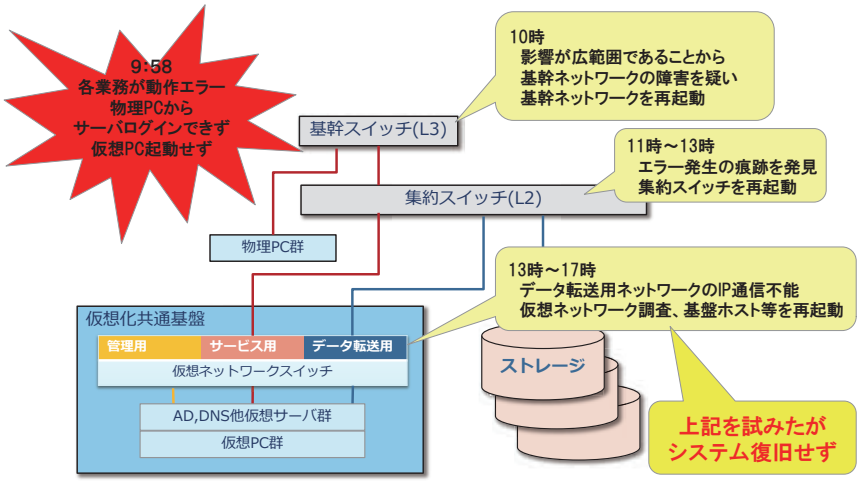


図 3.25－1　障害状況

ハード／ソフト両面からの全面再調査の過程で、この会社に設置していたサーバやスイッチ機器を物理層からすべて再点検した結果、午前中に再起動したはずの集約スイッチの状態表示 LED の点灯状況が通常と異なることが判明した。コマンドによる再起動では状態が改善していないことから、電源コードオフによる物理的な再起動を試みたところ、今回は通信が回復し、当日の 22 時頃にシステムの動作は正常に戻った。

この結果、いずれかのスイッチ機器の動作異常がトラブルの原因であるとの、トラブル発生後の初期調査時の判断が正しかったことが証明されたが、一方でこの障害はコマンドによる再起動では解消しないという別の問題があることも判明した。

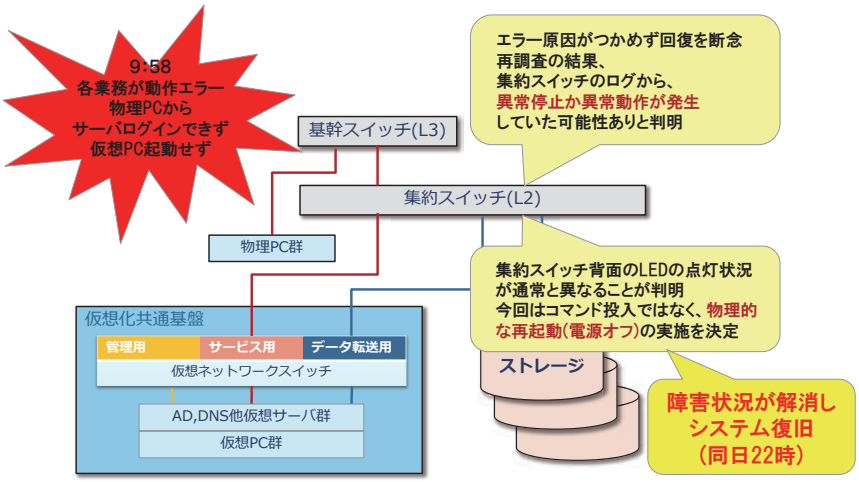


図 3.25－2　障害対応状況

原因

トラブル発生時にスイッチ機器から取得したログを機器の製造元に調査依頼した結果、ファームウェアの障害が原因である可能性が高いとの回答が寄せられたが、以下の理由から発生原因についての確証が得られず、本件の発生原因はこの教訓の作成時点で判明していない。

原因を特定できない理由

(1) 障害発生時のトラフィック（通信内容）の情報を取得できていない

(2) 障害発生の前後の時間帯に、異常が発生したスイッチに障害を示すログが出力されていない

(3) 調査用の情報が限られるため、スイッチ機器開発元でも自社に寄せられた過去の障害事例から原因を類推するまでが限度であり、その後、根本原因が判明したとの報告はない。

対策

上記の理由により発生原因を特定できず、この事例では再発防止に向けた恒久的対応を実施することができなかった。そのため、この会社では次善の策として障害再発リスク軽減のための予防保守【対策 1】と、それでも事象が再発したときの発生時対策の準備【対策 2】をそれぞれ実施した。

【対策 1】 障害再発リスクの軽減

この事例では、確定ではないながらも以下の障害原因が考えられたことから、以下の再発防止策を実施した。

(1) 集約スイッチ機器のハードウェア故障

未知の故障個所が内在している可能性もあるため、予防のために機器の交換を実施。ただし、機器を交換するためにはこのスイッチで通信を制御している社内共通基盤システムを一時的に停止する必要があったため、後日、休日にシステム停止日を設けて交換を実施した。

(2) 集約スイッチのファームウェア障害

当該のスイッチ機器は 1 回 / 年の法定点検日にその時点の最新のファームウェアを適用する運用にしており、障害発生時には最新のファームウェアを適用していなかった。ファームウェアを最新のものに更新していれば今回の障害の発生を予防できたかどうかは不明であるが、今回の障害の調査の過程で集約スイッチの開発元からはファームウェア障害が原因である可能性も報告されたことから、集約スイッチ機器の交換時にあわせてファームウェアを最新のものに更新した。

【対策 2】 障害再発時の対策の準備

このシステムは 365 日 24 時間の稼働を要求されるようなものではないが、トラブル発生時には早期に回復ができないと、回復までの間は社内の業務が停止する影響度がある。【対策 1】の実施により同一原因による障害の発生リスクを軽減できたと考えられるが、原因が判明していない以上、再発の危険性は解消されていないものとして、以下に示す対策を実施した。

(1) 障害原因切り分け基準の決定

今回と同様の事象が発生した際に、スイッチ装置の LED 点灯に異常が発生しているかどうかの判断が対応した要員によって異なることがないように、正常動作時の LED 点灯状況を文書化した。

(2) 障害対応手順の作成

スイッチ装置の動作異常が発生していると判断できた場合に、スイッチの物理的な電源切断から再起動、正常に再起動したかどうかの確認、正常動作しなかった場合の機器の切り離しまでの一連の動作を間違えずに実施できるよう、作業手順を文書化した。

(3) 根本原因調査のための施策の組込み

このシステムで提供する業務も障害発生時には回復最優先であるが、今後もこの障害が多発するリスクをなくすため、障害が今後再発したときに原因調査用に最低限取得するログを選択し、回復処置を実施する前に短時間で確実に取得できる手順を作成した。

(4) 障害対応マニュアルへの追加

上記で作成した (1) から (3) までの各文書を既存の障害対応マニュアルに追加し、印刷して設

備近くに常備することにより、障害発生時にすぐに参照できるようにした。その際には、今回の対策で追加したスイッチ点灯状況の切り分け基準に当てはまらなかった場合や、対応手順どおり実施してもスイッチが正常動作に復旧しなかった場合の対応、スイッチ機器製造元への障害発生の連絡などの対応ルールを追加した。

効果

上記の施策により、この事例では以下の効果を得た。

(1) 同一事象再発の抑止。

予防保守の効果があったかどうかは不明であるが、同一事象は再発していない。

(2) 障害発生時の早期回復

仮に同一の障害が発生した場合には、作成した手順に沿って調査から回復処置までを実施し、早期に業務を回復させられるプロセスを構築できた。

(3) 今後新たなパターンの障害があったときの基本動作の醸成

今後、今回と異なるパターンの未知の障害に直面した場合にも、今回と同様に予防保守の実施、発生時対策の構築により、サービス運用の改善を図る組織風土を醸成できた。

教訓

この事例からは、以下の事項について教訓として活用ができると考えられる。

(1) スイッチ機器がコマンドによる再起動で回復しないときは電源切断を試みる

コマンドによる再起動で正常動作に復帰しない場合には、機器の内部で論理エラーが発生していてコマンドを正常に処理できない状態になったと判断して、電源切断と再投入による機器の回復操作を行うことが必要になる。スイッチ機器のように電源投入用のハードウェアスイッチがない機器の場合には、電源プラグの抜線により上記を代替することが必要になる。スイッチ機器の物理的な電源断が可能な環境では、一見乱暴な手段に見えてもこれらの操作を回復手順に組み込む事が、障害からの早期回復に役立つ。ただし、判断基準と作業手順を文書で周知徹底し、作業を標準化することとペアで実施することが必要である。

(2) システムの運用要件に沿ったリスク軽減策と発生時対策を用意して障害に備える

発生原因不明の障害や原因解明済でも根本対策が実施できない障害の再発リスクがあるときには、とり得る対策を実施した上で、いざ発生したときにどう対応するかを決定して、再発に備える事が必要になる。稼働に対する要求はシステムごとに異なる。再発のリスクが残っても直ちに現場の業務を復旧させる必要があるシステムもあれば、原因調査を綿密に実施して障害を再発させないことが最優先のシステムもある。障害対策の立案においては、対象とするシステムの特性に合わせて実施することが必要になる。

(3) 根本解決のために必要最低限の調査資材を取得する

障害発生時における対応については、即時復旧と再発防止のバランスの考慮が必要である。即時復旧最優先のシステムであっても、原因究明用に取得が必要な必要最小限の調査資材を選択し、

それらを短時間で確実に取得する手段を用意しておくことが、障害の根本原因解明への糸口となり、最終的にシステム稼働品質の向上に結びつく。

この事例では、障害とその対応から得られた経験は相応の対価を払って得た貴重な財産であり、たとえ根本原因がわからず恒久的な解決策が実施できない状況にあったとしても、次回発生への備えとして必ず活用すべきものであることをまとめた。

昨今、システム構築においては自己の開発するアプリケーション以外に多種多様な機器を組み合わせることが多く、それらの機器には未摘出の障害が潜んでいるリスクも大きくなってきた。あらゆるリスクを事前に察知し、リスクの除去や軽減を行うことは事実上不可能である以上、システム運用においては障害再発の予防策と障害発生時の対応策を、そのシステムの要求レベルに応じて実施することが必要である。この教訓では、各システムにおいて上記を実施して障害に備えることの重要性を発信したい。

教訓タイトルは、「障害原因が不明でも再発予防と発生時対策はできる」とした。

なお、ネットワーク機器における未知の障害への予防策として、教訓 T17「長時間連続運転による不安定動作発生の回避には、定期的な再起動も有効！」もあわせて参照を願う。

3.26 既存システムの流用開発に関する教訓（T26）

教訓 T26　既存システムの流用開発はその前提条件を十分把握し、そのまま利用可能な部分と変更する部分を調査して実施する

問題

A社のシステムは営業店に対してサービスを提供するオンラインシステムであり、夜間の業務終了から業務日付変更のバッチ処理が行われ翌日定刻にオンラインサービスが開始される（図3.26－1）。

ある日、定刻のオンライン業務が朝から開始されず取引が停止した。ログ調査の結果、通常は朝方に実施される業務日付変更処理がこの日は正常終了していないことが判明した。オンラインシステムが復旧し取引可能となるまで数時間を要した。

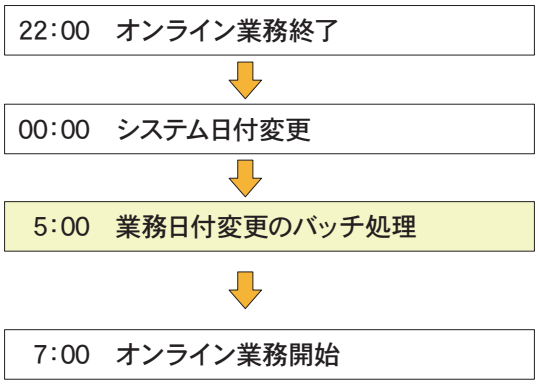


図3.26－1　オンラインシステムの日次起動スケジュール（自動運用）

原因

【直接原因】

原因は、業務日付変更のバッチ処理の不具合である。

当時、バッチの一つとして夜間に起動される業務日付変更処理は、通常時、次のように動作していた（図3.26－2）。

JOB管理機能から2つの運用ジョブXとYが順次起動され、並列に動作を開始する。各運用ジョブ内の先行サブ処理が完了するとJOB管理機能宛に完了メッセージを送る。JOB管理機能は、サブ処理X1からの完了メッセージを受領すると後続のサブ処理X2を起動する。同様に、サブ処理Y1からの完了メッセージ受領により、後続のサブ処理Y2を起動する。サブ処理X2、Y2がともに終了

すると、取引業務日付変更処理が完了する。先行する各サブ処理と後続の各サブ処理は完了メッセージを介して同期をとる仕組みであった。

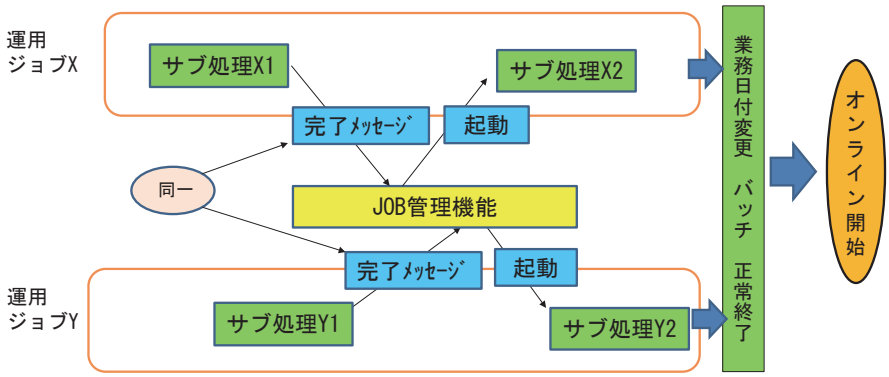


図 3.26－2　業務日付変更のバッチ処理（通常時）

ところが、当日は、サブ処理 X1 の処理が遅延し、サブ処理 Y1 がサブ処理 X1 よりかなり先に終了した。このとき、サブ処理 X1 からの JOB 管理機能宛の完了メッセージと，サブ処理 Y1 からの JOB 管理機能宛の完了メッセージとが同一内容であったため、JOB 管理機能が先にサブ処理 Y1 から受領した完了メッセージをサブ処理 X1 からの完了メッセージと誤認し、後続のサブ処理 X2 を起動した。サブ処理 X2 はサブ処理 X1 の終了が前提となっているが、この時点でサブ処理 X1 は終了していなかったため、処理に論理矛盾が発生しサブ処理 X2 が未処理となり業務日付変更のバッチ処理が正常に終了しなかった（図 3.26－3）。

過去のログを調査したところ、これまでにも同様の終了時刻の逆転はときどき発生していたが、極小時間の差でありサブ処理 X2 の開始までにはサブ処理 X1 が終了していたことから、JOB 管理機能による完了メッセージの誤認があっても問題は発生していなかったことが判明した。

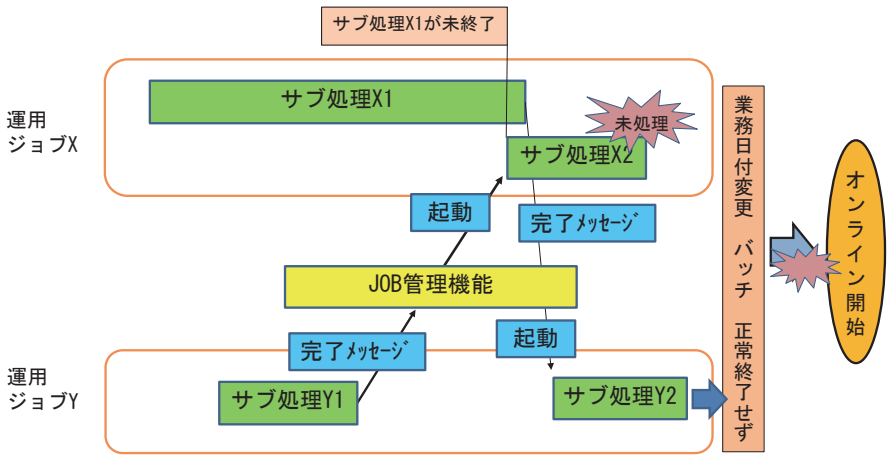


図 3.26－3　業務日付変更のバッチ処理（トラブル発生時）

【根本原因】

A 社のシステムは当初システムの開発から機能拡張－1、機能拡張－2 と追加改修が実施された。

表 3.26－1　システム開発の時期と運用ジョブの追加

システム開発の時期	当初システム	機能拡張－1	機能拡張－2
運用ジョブ	すべて単独ジョブ	運用ジョブ X 追加	運用ジョブ Y 追加
開発の形態	初期構築	流用開発	流用開発
完了メッセージ制御	なし	**あり**	**あり**
並行ジョブの有無	なし	なし	**あり**

機能拡張－1 から完了メッセージによる制御機能を入れたが、このときには運用ジョブ X は単独のジョブであったので問題にならなかった。機能拡張－2 から運用ジョブ Y が追加され、運用ジョブ X と並行して動作するようになった。追加した運用ジョブ Y は運用ジョブ X を流用して開発され、完了メッセージの内容がそのままコピーされていたため、2 つの完了メッセージが運用ジョブ X、Y どちらのものか識別できない（メッセージが一意でない）不具合が作りこまれた。

運用ジョブ X の処理詳細は仕様書に記載されていたが、運用ジョブ Y の設計者は運用ジョブ X の設計者とは異なっており、既存仕様書の調査分析が不十分であった。

また、サブ処理 X2 が未処理となっていることに気づかなかったため、原因究明に時間がかかったことも問題であった。

対策

問題の直接原因を取り除く措置として、実施した対策は以下の通りである。

- 運用ジョブ X と運用ジョブ Y で使用する完了メッセージをそれぞれ区別できるメッセージに修正し、メッセージの一意性を担保した。
- 念のため、運用ジョブ X と運用ジョブ Y を並行処理から逐次処理に変更した。

なお、全システムを調査して、同様のケースがあれば同じ対処も実施し、完了メッセージ一覧をドキュメントに追記した。

また、オペレータがトラブル発生を早期に検知するために、未処理となったジョブを監視するようにした。

再発防止策は以下の通りである。

- 設計でのレビューの強化

  流用開発における既存システムの前提条件の確認、追加部分と既存部分とが不整合や競合を起こすことがないか、システム全体を俯瞰するチェックを行う。

  特に、一部を流用している場合には、システム内に類似処理が存在することになり、それらが

コンフリクト（衝突）を起こすリスクがあることを認識する。（今回は、並行稼働する２つの運用ジョブの完了メッセージが同じものとなった。）

　このための手段としては、例えば、システム全体の運用スケジュールとプロセスやジョブの起動するシーケンス図を作り、ウォークスルーを実施する。特にメッセージやファイル名等の一意性が担保されているかチェックを行う。

- 既存仕様書への処理の前提事項の記述追加

　完了メッセージによる順序性の制御は単独処理が前提であり、並行処理する場合の留意事項の追加を記述する。

効果

　その後のシステム改修では再発防止策の観点で稼働中システムに対しても設計レビューや追加テストを実施するようになったため、以降は特に問題なく運用している。

教訓

　システムを流用開発する場合は、稼働済みの既存システムを信頼してまるごとコピーしがちであるが、メッセージやファイル名等が一意でなくなるなどの落とし穴があることが多い。既存システムでの前提条件を十分把握し、そのまま流用可能な部分と変更が必要な部分を調査分析し、可能であれば既存システムの開発者も交えてレビューすることが大切である。

3.27 基幹系システムにパッケージソフトを適用する際の教訓（その 1）（T27）

教訓 T27

パッケージはサポートを買え

問題

A 社で保有する社内特定業務向け基幹業務システムは、現場部門に対して 24 時間のオンライン業務サポートを実施している。業務は日中だけでなく夜間も継続しており、かつシステム停止による現場業務への影響が大きいことから、運用サポート部門も 24 時間体制でシステムの運用をサポートしている。業務システムは当該用途向けの海外製パッケージを購入して構築されており、アプリケーションサーバ、DB サーバ、他システムとの連携サーバなど各層のサーバは冗長構成になっている。構築したシステムは本稼働後、2 カ月目を迎えようとしていた。

ある日の午前、アプリケーションサーバが突然システムダウンした。そこから、システム回復までの経緯はおおよそ以下のとおりである（カッコ内の時刻はパッケージ開発元の現地時間を表す）。

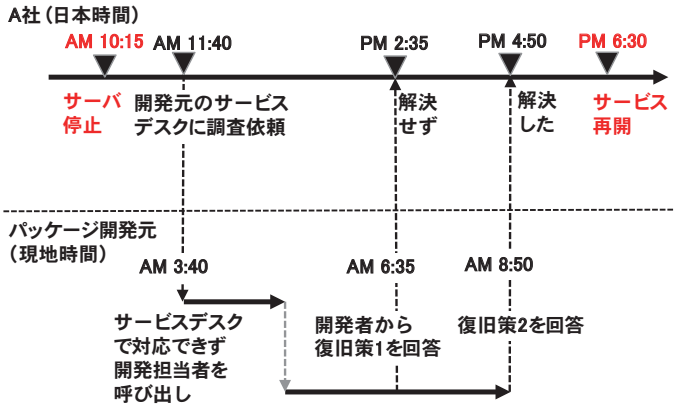


図 3.27－1　障害発生後の対応状況

10:15（2:15）　運用センターでアプリケーションサーバのシステムダウンを検知
すぐにサーバ再起動、スタンバイへの切替えを行ったが状況は変わらず

11:40（3:40）　導入したパッケージの開発元である B 社のサービスデスクに調査を依頼
その際にサーバのシステムリソースの枯渇状況も連絡
サービスデスクでは問題を解決できず、パッケージの開発チームを深夜に呼び出す

14:35（6:35）　B 社から復旧策 1 を受領
指示された作業を実施したがサーバダウンは解消せず

16:50（8:50）　B 社の継続調査により、原因と思われるデータの削除を提案される（復旧策 2）
　　提案の作業を実施した結果、サーバダウンは解消した
17:30（9:30）　他システムとの連携を再開し当該システムへのデータ入力を再開
18:30（10:30）　すべてのデータ入力が完了し正常稼働を確認した上で、サービスを再開

問題発生後、サービス再開まで 8 時間を要した結果、その間、当該特定業務の業務運用は手作業に頼らざるを得なくなり、当該業務を通してエンドユーザに提供しているサービスにも大きな影響が出た結果、その復旧を含めてA社のビジネスに大きな損害をもたらした。

原因

サーバがダウンした直接の原因は、導入したパッケージの潜在的な不具合によるサーバのシステムリソースの枯渇（アプリケーションが使用する内部テーブル上に古い情報が滞留したこと）であり、A 社で初めて発生した事象であった。このパッケージは A 社以前に複数社に導入された実績のあるものであるが、今回のトラブルが先行する他の導入先で発生しなかった理由は、先行する他の導入先では取り扱うデータ量が A 社ほど大量ではなく、ディスク、メモリ等のシステムリソースを枯渇するまで使い切ることがなかったためであった。A 社がこのパッケージで取り扱うデータ量が他の先行する導入先に比べて圧倒的に多かったことが、この障害を初めて顕在化させた。

また、このトラブルの発生後、復旧までに長時間を要した理由は、A 社から B 社に調査を依頼する先が B 社のサービスデスクであり、開発チームがすぐに調査を開始することになっていなかったため、高度の技術、製品に対する豊富な知識を必要とする調査であることの判断、深夜に開発チームを呼び出して調査を開始するまでの初動に時間を空費したことがあげられる。

対策

上記の原因を特定した A 社では、直接の原因であるパッケージの不具合の修正だけでなく、今後、同様にトラブルが発生した場合に、対応時間を短縮するために、以下の対応を B 社と協議の上で実施した。

【対策 1】 B 社開発チームによる 24 時間サポート体制の構築

このシステムで提供する業務は障害発生時には回復最優先であることから、パッケージ開発者自身による調査を迅速に開始することが最も効果がある施策であると判断した A 社は、障害発生時の B 社連絡先をこれまでのサポートデスクから B 社開発チームのメンバへの直接連絡に変更し、日本国内製の他の重要パッケージのサポートと同様に 24 時間いつでも直接開発担当者によるサポートを受けられるよう B 社に対応強化を依頼した。その上で、B 社の開発チームのメンバの氏名を公開してもらい、当番制で直接連絡できるようにした。

【対策 2】トラブル対応のサービスレベルを明確化

トラブル発生時のファーストコールから回答までの目標時間を設定し、この時間内に正しい回答が得られない場合にはペナルティを課すことを SLA（Service Level Agreement）に定め、サポート契約を締結し直した。

これを今回のトラブルに当てはめると、下記のようになる。

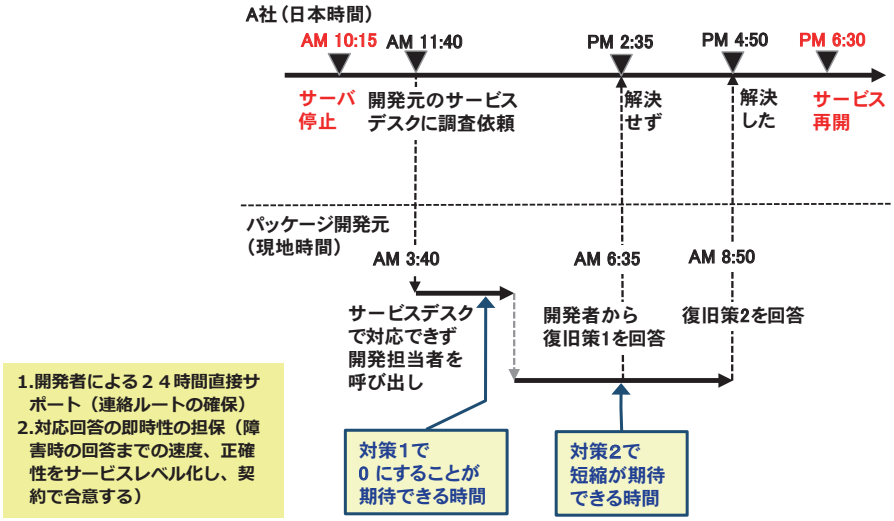


図 3.27－2　施策と期待効果

効果

上記の施策により、この事例では以下の効果を得た。

（1）トラブル発生時の回復までの時間短縮

この施策で期待する効果そのものであり、A 社ではその後発生した別内容のトラブル対応において大幅な時間短縮（再開まで 1 時間）を達成し、施策に効果があることを立証した。

（2）SLA 締結による調査精度向上

SLA 締結は B 社内での緊張感の持続をもたらし、結果として後続の障害における調査、対応方法提案の精度を向上させることができた。その効果は上記（1）の時間短縮に表れている。

教訓

この事例からは、一時的なシステムの停止が業務に大きな影響をもたらすシステムにパッケージを組み込む際の以下の注意点が教訓として活用できると考えられる。

（1）パッケージ提供元の開発チームによる直接の協力体制の構築

業務時間中にシステムが停止すると業務に多大な影響を与えるようなシステム（IT サービス）にパッケージを組み込む場合には、そのパッケージが運用中にトラブルを発生させた場合に即時対応でき

る体制を開発元の協力により構築することが有効である。自社開発したシステムにおいて、開発に関わったチームを維持メンバとして保持することに相当する行為として、パッケージの内容を最も知悉した開発元の要員に素早く調査を委ねることが最速のトラブル解決方法になる。

(2) トラブル対応の目標設定による協力と緊張の維持

トラブルに即時対応する体制を構築しても、ゴールを定めないと、調査の開始時間が早まるだけで、調査時間の短縮には直接は結びつかない。

そのため、今回の事例のように、トラブル対応の委託先とは以下のような事項について目標を設定して SLA として文書化し、契約することが有効に作用する。

（SLA の項目の候補）

- トラブル発生の連絡が開発元の技術者に受け付けられるまでの時間
- 調査・回答の提案までの時間
- 回答の正確性（回答によって事象が解決した率）
- パッケージ障害発生率

これらの事項を決定して契約することにより、開発元にも緊張感が生じ、製品やサポートの品質に対する意識の向上も期待できる。

(3) 重要なシステムリソースの枯渇監視を組み込む

今回の事例でもサーバリソースの枯渇監視を実施しており、本件の原因になったリソースの枯渇が報告されていたが、枯渇の原因まではわからず、解決には開発元の調査を待つしかなかった。だからといって、リソースの枯渇監視には意味がないというわけではなく、リソースの枯渇状況は開発元に情報として伝えられ、調査の取り掛かりに有力な情報になった。直ちにトラブル解消に結びつかなくても、予兆の察知や調査情報の提示に有効に働くことが期待できることから、重要なサーバのシステムリソースの枯渇監視も実施を検討すべき施策であると考える。

昨今、システム構築においてパッケージ製品を組み込むことが多くなってきている。パッケージを適用することは、システム構築のスピードを向上させることに多大なメリットをもたらすが、同時に内容がブラックボックス化して導入した会社だけではトラブルを解決できないというデメリットをもたらす。システム構築、特に高度の安定稼働を必要とする業務システムの構築にパッケージを適用する際には、パッケージ本体の製品品質だけでなく、何かあったときに必要なサポート体制を高い品質で提供してくれる開発元であるかどうかも、パッケージ選択の重要な要素になる。この教訓では、パッケージ選択の際には、適用するシステムの特性に応じた必要十分なサポート体制を供給してくれる提供元であるかどうかを確認することが重要であること、国内はもちろんであるが海外製パッケージを導入する際には、現地時間の休日・夜間においても上記のサポート体制を維持できるかどうかが、導入の重要な決め手になることを発信したい。

教訓タイトルは、上記を強調することを狙いとして「パッケージはサポートを買え」とした。

3.28 基幹系システムにパッケージソフトを適用する際の教訓（その 2）（T28）

教訓 T28　パッケージを更新するときは、変更内容の詳細確認と回帰テストで二重に安全を確保せよ

問題

A 社で保有する社内特定業務向け基幹業務システムは、現場部門に対して 24 時間のオンライン業務サポートを実施している。業務システムは当該用途向けのパッケージを購入して構築されており、アプリケーションサーバ、DB サーバ、他システムとの連携サーバなど各層のサーバは冗長構成になっている。構築したシステムは本稼働後、2 年間安定稼働したのち、2 週間前にメジャーバージョンアップを実施したばかりであった。

ある日、アプリケーションサーバからの反応が突然なくなった。そこから、システム回復までの経緯はおおよそ以下のとおりである。

7:48　サービスが使用できない（応答がない）と現場から多数の連絡が入るようになった
すぐにアプリケーションサーバやデータ連携用のサーバを再起動したが、状況は全く改善せず

8:51　未処理のまま滞留している他システムからの連携データを一度すべて削除し、他システムとのデータ連携も一時的に停止した上でアプリケーションサーバを再起動した結果、レスポンスが回復

10:30　正常の構成に戻してサービスを再開

問題発生後、サービス再開まで 3 時間を要した間、当該特定業務の運用が完全に停止し、当該業務を通してエンドユーザに提供しているサービスにも大きな影響が出た結果、その復旧を含めて A 社のビジネスに大きな損害をもたらした。ただし、当サービスの不測の停止に備えて、このサービスを使用する運用現場とサービスが稼働しない状態でも実施できる範囲の業務対応を準備していたことから、最も影響のある業務の停止だけは回避できた。

原因

アプリケーションサーバがダウンした直接の原因は、2 週間前にパッケージをバージョンアップした際に置き替えられたアプリケーションに混入していたバグであった。パッケージの開発元ではメジャーバージョンアップの際に、一部のアプリケーションについて性能改善のための改造（アプリサーバのキャッシュで更新制御）を実施していたが、その際、誤って複数の処理間でのリソース競合が発生す

るような内容での修正を実施したことにより互いに処理待ちになっていたことが，サーバが反応を返さなくなった原因であった。

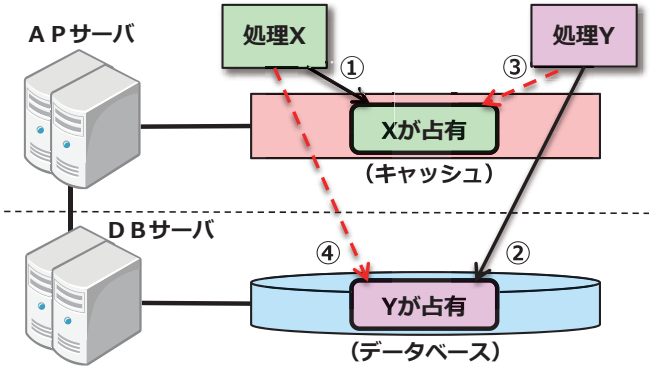


図 3.28－1　サーバ資源占有競合の概要

上図において処理 X はキャッシュ、ディスクの順で資源を確保し、処理 Y はその逆の順で資源を確保する不統一な作りになっていた。その上、それぞれが自分の占有した資源を開放せずに他方が占有した資源の解放を待つ作りになっていたため、両者ともに相手の資源開放待ちで停止した。

また、パッケージ開発元から提供されるリリースノートには、機能レベルの大きな変更までしか記載されておらず、今回のような個々の処理に対する性能改善レベルの修正が実施されていることは A 社には伝わっていなかった。そのため、A 社ではその修正内容の是非やリスクの検討、テスト環境での事前の動作確認はできていなかった。

対策

パッケージ開発元から発生原因が上記である旨の報告を受けた A 社では、パッケージの当該個所の修正だけでなく、今後もし同様のトラブルが再発したときの備えとして、パッケージ開発元と協議の上で、以下の対策を実施した。

【対策 1】パッケージアップデートに含まれるすべての修正に対するレビューの設定

A 社では、これ以降、パッケージ開発元からアップデートが提供される際に、アップデートに含まれるすべての修正に対して内容の詳細な報告を求め、両者による修正内容のレビュー会を実施することをパッケージ開発元に承諾させ、提示されたすべての修正内容に応じた受入テストを計画することとした。また、A 社内ではレビュー会を実施していないアップデートの運用環境への適用を一切禁止するルールを敷いた。これにより、パッケージ開発元から報告された修正に対しては運用環境に適用する前に A 社でも内容を認識し、確認することができるようになった。

また、レビュー会においては、今回の障害の直接原因になった資源占有（排他）順序の不統一のような誤りを、パッケージ開発元で出荷前にちゃんと摘出できるようなプロセス改善が適切に計画され、実施されているかどうかをあわせてチェックすることにより、不具合混入の再発防止を行った。

【対策 2】回帰テストシナリオの作成とテストの実施

A 社では、さらにパッケージ開発元の管理者でも認識していないような修正が、開発元内での修正報告漏れなどにより実施されているような場合に備え、回帰（リグレッション）テスト用のテストシナリオを作成して、パッケージが入替え前後で同一の動作をすることを確認することを手順化した。さらに、このテストシナリオを自動で動作させられるテスト環境を構築し、システムを更新するときに回帰テストを実施する際の負荷軽減とテスト手順の誤りによる障害摘出漏れの防止を実現した。

【対策 3】アプリケーションレベルでの死活監視の組込み

A 社では、今回のアプリケーションサーバと DB サーバの間での資源競合のような、サーバ単体でのデッドロック監視では検出しきれないエラーへの対策として、アプリケーションレベルでの死活監視を新たにシステムに加えた。さらに、実行中のプログラムがエラー発生を検知したら、すぐにシステム運用担当者に直接メールが発信されるようにアプリケーションに機能を追加し、システムに組み込んだ。これにより、場合によってはエラー発生時の初動を、サービスの利用者からの連絡よりも早く開始することができるようになった。

効果

上記の施策により、この事例では以下の効果を得た。

(1) アップデート適用前の修正内容の全件把握とテストによる確認の確実な実施

提供されたアップデートを運用環境に適用しても問題ないかどうかの確認を、これまでよりも正確かつ緻密に実施できるようになった。また、アップデート受入れ時の動作確認においても、修正個所に直接関係するテスト項目の設定を高精度で実施できるようになり、確認の精度が向上した。

(2) パッケージアップデート前後での同一動作の担保

パッケージをアップデートしても、アップデートに関係しない部分の動作がこれまでと同様であることを維持できているかどうかを、より低負荷かつ確実に確認できるようになった。

教訓

この事例からは、パッケージのように他社から実行形式モジュールだけを提供され、その内容についてまでは利用者側では詳細を知り得ないソフトウェアを業務システムに組み込んでいるシステムを更新する際に考慮すべき、以下の注意点が教訓として活用できると考えられる。

(1) アップデートを適用する際には、修正内容を詳細まで把握すること

通常、パッケージソフトウェアの更新モジュールにはリリースノートが添付されているが、そこに記載される内容は、業務上の機能レベルでの追加や変更が大まかに記載されていることがほとんどである。パッケージの個々のモジュールのどこにどのような修正をしたかについてまでは、リリースノートに記載されていないどころか、開発元でも詳細レベルまでは管理しきれていないのが実態ではないかと思われる。パッケージの利用者はそれらの不十分な情報をもとに、自己の運用するシステムに適用する前に受入確認を行い、その後に運用環境に適用する。今回の事例のような高度の運用品質を求められる

システムにパッケージを適用する必要がある場合には、特別のサポートにより修正内容を詳細に報告させ、それに基づいた厳密な受入れテストを実施できるような体制を採ることが必要である。

(2) 回帰テストの実施により、予期しない変更に備えること

すべて報告すると契約していても、漏れが生ずるリスクは当然残る。そのリスクの顕在化に備えて、パッケージがアップデート前後で同じ動作をすることを、上記 (1) の受入れテストとは別に、回帰テストによって確認することが望ましい。できればすべての機能について、あるいは、それが現実的には困難な場合には、パッケージの主要な機能についてテストシナリオを用意して、それをシステム更新前に必ず実行するように準備しておけば、アプリケーションの入替えだけでなく、OS やミドルウェアのアップデートや、機器の入替えの際にも動作確認に使用できることから、確認の範囲が広がり、システム保守の信頼性をより向上させることができる。

システム構築において、OS、ミドルウェア、アプリケーションの各層にパッケージ製品を組み込むことは最早当たり前になってきている。パッケージを適用してシステムを構築した場合、そのアップデートが出ても自己の運用するシステムに影響がある修正でない限り適用をしないユーザも多い。ただし、自己のシステムに影響があるアップデートや、あるアップデートを適用する際に、これまで適用を保留してきた大量のアップデートをまとめて適用することが必要になった場合、今回の事例のようなアプローチによって、少しでも安全にアップデートを行うことが必要になる。また、もしパッケージに選択の余地があるなら、アップデートの提供時により詳細な内容の更新情報を提供するパッケージ開発元を選択することも必要になってくると考えられる。この教訓では、システムに組み込まれているパッケージにアップデートを実施する際には、適用するシステムに求められる運用品質に応じた更新情報の取得努力と、それが得られない場合のリスクヘッジを用意することが重要であることを発信したい。

教訓タイトルは、「パッケージを更新するときは、変更内容の詳細確認と回帰テストで二重に安全を確保せよ」とした。

なお、ソフトウェアパッケージと同等にブラックボックス化しやすいハードウェアコンポーネントを交換した際に発生したシステム障害の事例として、教訓 T12「新製品は、旧製品と同一仕様と言われても、必ず差異を確認！」もあわせて参照を願う。

3.29 システム環境の変化への対応に関する教訓（その 2）（T29）

教訓 T29　単位などの定義が異なる制限値、連携するシステム間で使っていませんか？

問題

A 社の物流センタでは、全国に拠点ターミナルを設置し、拠点ターミナルでは、配送担当の運転手にハンディ端末を持たせ、配送先への荷物の優先度に応じた配送を管理している。この管理は、連携する 3 つのシステムを用いて行われている。中央管理センタの全社配送管理システムは、全社の配送計画を作成し、拠点配送計画システムへ全体配送計画を送る。拠点配送計画システムは、全体配送計画を受け取り、拠点ターミナルから入力される「配送ルート」データと「配送車両」データを受け取り、配送車両に応じた個別の配送ルート計画を策定し、その情報を個別配送監視システムに送る。個別配送監視システムは、個々の配送車両と配送ルートの配送状況をリアルタイムで管理する（図 3.29－1）。

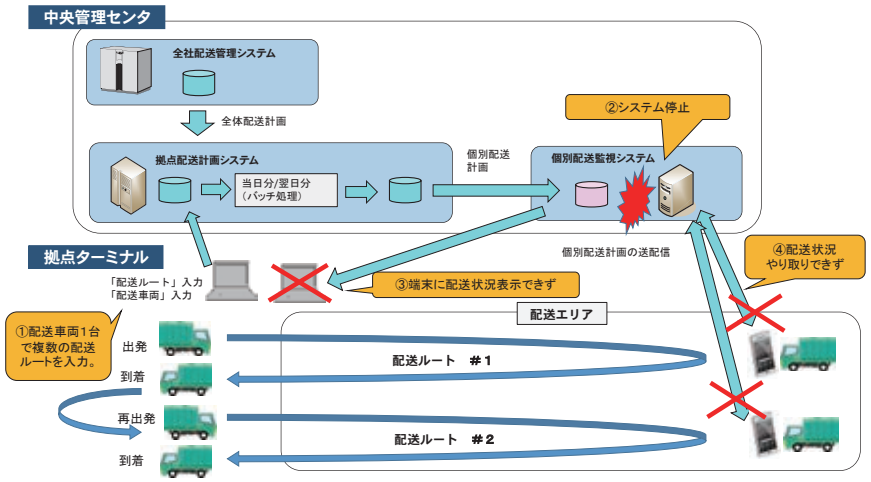


図 3.29－1　システム構成と障害状況

障害のあった日は、大型連休明けの初日でもあり、大量の荷物を扱うことになったため、前日に登録する配送車両1台当たりの配送ルート数が増えていた（図 3.29－1①）。障害のあった日の早朝、個別配送監視システムが停止してしまい（図 3.29－1②）、拠点ターミナルでは配送状況が分からなくなり（図 3.29－1③）、運転手も配送状況のやり取りができなくなった（図 3.29－1④）。原因はすぐにつかめなかったが、配送そのものについては順調に行われていたので、拠点ターミナルと運転手の間で、電話でのやり取りで対応し、その日の業務を終えることができた。

業務終了後に原因が判明し、緊急システム改修（暫定対応）を実施し、翌日早朝にシステムは復旧した。

原因

A 社は、過去にシステムの配送ルート数の制限値の管理をしていなかったために、システムが停止する障害を発生させたことがあった。その障害を踏まえ、配送ルート数の制限値を拠点配送管理システムと個別配送監視システムを通し、一律管理する運用を行っており、1 日当たりの配送ルート数の制限値を、「1,000 ルート」としていた。ここでいう配送ルート数は、例えば配送車両 1 台が拠点ターミナルから出発して戻ってくるまでを 1 ルートと数える。繁忙時に、1 台の車両で拠点ターミナルを 2 回出入りすれば、2 ルートと数えるので、物理的な車両台数ではない。

A 社のシステム部門は、拠点配送管理システムから個別配送監視システムに転送されるデータは、拠点配送管理システムで上限を超えないことについて担保されるため、下流側の個別配送監視システムでは制限値を超える可能性についての検討は重要視していなかった。そのため、システム部門は、拠点配送計画システムの開発ベンダとは別のベンダである個別配送監視システムの開発ベンダに、制限値「1,000 ルート」と伝えただけであった。

しかし、このシステムでは、配送ルート数として持つ制限値としては、当日分の配送ルート数と、翌日分の配送ルート数が存在していた。それは、拠点ターミナルで当日に翌日分の配送計画を事前に立案できるようにし、拠点ターミナルでの負担を減らすためであった。その制限値が 2 つあるにも関わらず、個別配送監視システムの開発ベンダは、当日「1,000 ルート」、翌日「1,000 ルート」とあるべきところ、当日、翌日あわせて「1,000 ルート」と理解して、システムの設計、製造を行ってしまった（図 3.29－2）。

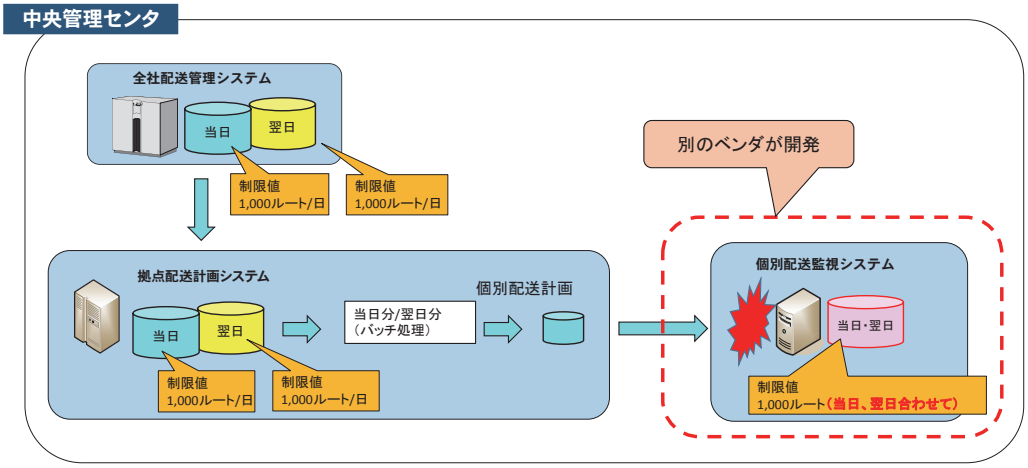


図 3.29－2　制限値の意味の違い

今回の障害は、当日「900 ルート」、翌日「400 ルート」の 2 日分のデータが、個別配送監視システムの制限値「1,000 ルート」を超えた「1,300 ルート」であったために起きた。直接原因は、上位システムと下位システムとで制限値の定義が異なっていることに気づかなかったことであった（図 3.29－3）。

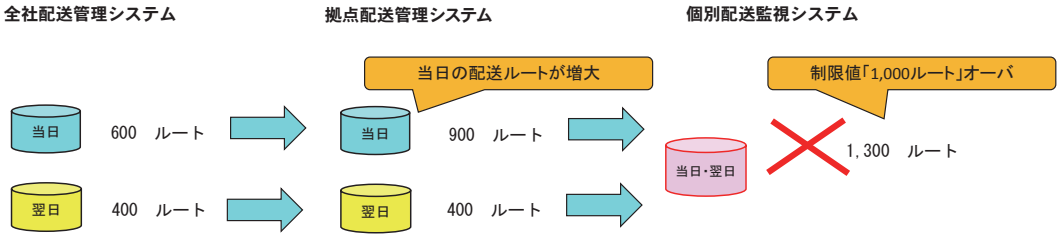


図 3.29－3　制限値オーバになるケース

根本原因は、機能仕様書を作成するとき、制限値の単位（本事例では、1 日ごと）や、システム連携範囲（本事例では、システム全体で一元管理）を明確にすることまで考慮が及ばず、記載内容を曖昧なままにしていたことであった。そのため、設計レビューでもその不整合を確認することができなかった。また、このような制限値を確認する総合テストも行っておらず、そのため、システム部門、開発ベンダともに十分な確認が行われていなかった。

対策

緊急対策として、個別配送監視システムの制限値を、「1,000 ルート」→「2,000 ルート」に拡張した。

再発防止策としては、以下の手順を実施した。

⑴ 個別配送監視システムの機能仕様書の制限値の定義見直し

- 機能仕様書とプログラム実装の制限値の定義に差異がないよう比較調査し、記載内容の見直しを行った。
- 文章だけではなく、シーケンス図等を使い処理の流れを明確化にした。

⑵ 個別配送監視システムの制限値超過時の振る舞い調査

- 個別配送監視システムの全制限値において、制限値超過時の振る舞いを設計書から調査した。誤りがあれば修正し、また説明が足りない部分は、それを設計書に明文化した。

⑶ 各システム間の制限値再点検と一元管理

連携するシステム間で持つ制限値は、その定義を明確にし、複数のシステムの制限値が同じ意味を持つことを確認する。さらに、そのような制限値を一元管理する。

- システム間の制限値の整合性を確認した。
- 運用時に、実際のデータが制限値にどこまで迫っているのかのデータ使用率調査を行うこととした。
- 上記の調査結果を常に監視できるよう、データ管理方法の見える化（システムごとの制限値一覧の作成、その制限値の変更履歴管理、性能調査状況履歴管理、等）を行い、システム間での制限値がどのように使われているかを明らかにし、すべての制限値を共通パラメータ化して定義する一元管理の運用を開始した。将来的には、共通パラメータ値の変更により自動的にすべての制限値定義の変更ができるよう、システム化することを検討する。

制限値には、機能要件と言われる業務要件から決めるものと、非機能要件と言われるシステムリソースの制限から決められるものがある。

また、システム間で管理すべきものもあるが、あるシステム内の閉じた中で管理するものもある。さらに個々のプログラムのロジックの関係から制限値を決めている場合もあり、その管理は複雑であり、すべてを管理することは至難である。

したがって、制限値の確認が必要な事象をとらえ、実際の運用に入る前に、本番環境と同様な環境で、事前テストを行うことが有効である。

個別配送監視システムの恒久的な対策として、制限値再調査の結果を受け、システム間で整合性を保つよう制限値を拡張した上で、制限値、およびピーク時の実データでの確認テストを実施することにした。

事例では制限値の中の上限値オーバが障害となったが、下限値についても同様に管理する必要があり、その他の注意点も、当然のことではあるが、以下になる。

- 制限値には、上限値、下限値がある。また、等号（=）が含まれるかも確認する。
- 上限値、下限値は、プラス、ゼロ、マイナス　なのかを確認する。
- 上限値、下限値は、一般に、「△△当たり○○」（または、○○ / △△）といった形で表され、△△と○○は、単位を明確にする。例えば、「1 日当たり 1,000 個」（または、1000 個 / 日）、「1 時間当たり 100 円」（または、100 円 / 時間）などのように確定する。

  「△△当たり」は、「時間当たり」、「面積当たり」、「個別当たり」などが当てはまる。「時間当たり」の単位は、1 秒、1 分、1 時、1 日、1 月、1 年、・・・などであり、「面積当たり」の単位は、1㎢、1㎡、・・・など、「個別当たり」は、1 プログラム当たり、1 人当たりなど、様々な単位が考えられる。○○は、「数量」の単位（ルート、台、回、個、トン、・・・など）であるが、「%」などの率もある。

図にまとめると、以下のようになる（図 3.29 – 4）。

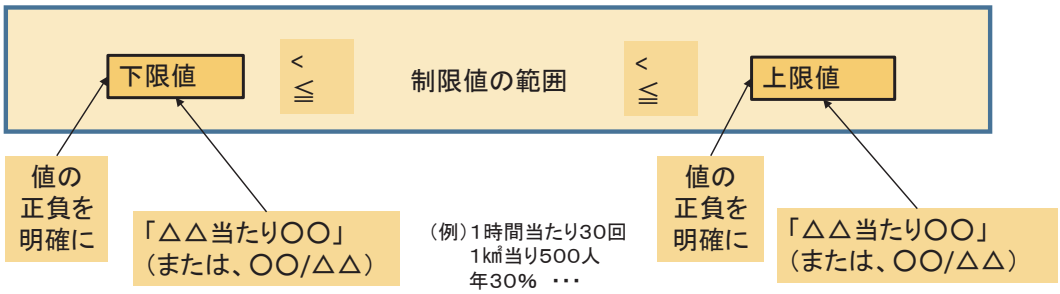


図 3.29 – 4　制限値の明確化ポイント

効果

制限値の管理を厳密に定義する、つまり、システムで持つ制限値は、数値だけでなく、「単位」、「単位当たり」、サイン（符号）、上限値、下限値など、正確な定義をした管理を行うことにより、連携システム間の不整合によって起こるシステム障害の発生を防ぐことができる。具体的には、以下の対策を行うことにより実現される。

- 設計時のレビュー時に制限値の定義を明確にする。
- 制限値付近の境界テストや、制限値の定義に合った動作ができるかのテストを行う。

また、これらの対策を実施するにあたり、マルチベンダでの開発時におけるコミュニケーションギャップによる不具合にも対応することができる。

教訓

連携システム間で持つ制限値は、その定義を明確にすることにより、複数のシステムの制限値が同じ意味を持つことが確認でき、さらに、そのような制限値は、一元管理することが重要である。

本事例では、1 日の制限値「1,000 ルート／日」であったが、システムによっては、1 日の制限値に加えて、1 時間の制限値、あるいは 1 秒の制限値など、意味の異なる複数の制限値を個別に管理する場合もあり得る。また、本事例では上限値オーバであったが、下限値の制限も忘れてはならない。特に「－（マイナス）」がチェックできなかったため、誤った結果をもたらすこともある。そのような点も踏まえ、この教訓を活用していただきたい。

この教訓では、制限値の定義を主題にしたのだが、他に、制限値（上限値）オーバに関する教訓があるので、参考にしていただきたい。制限値の管理に関しては、その定義を明確化することと、それが変わる変化点を見逃さないこととを合わせて検討していただきたい。

【教訓 T4】 システム全体に影響する変化点を明確にし、その管理ルールを策定せよ！

また、システム間での連携の在り方を述べた教訓がある。こちらも参考になると考える。

【教訓 T5】 サービスの視点で、「変更管理」の仕組み作りと「品質管理責任」の明確化を！

さらに、このような制限値を一元管理する教訓がある。こちらも参考になると考える。

【教訓 G13】 キャパシティ管理は関連システムとの整合性の確保が大切

【教訓 T18】 新たなサブシステムと老朽化した既存システムとを連携する場合は両者の仕様整合性を十分確認すべし

3.30 ネットワーク2重化の敷設に関する教訓（T30）

教訓 T30　意味がない、一緒に束ねた2重化配線！

問題

A社では、自然再生可能エネルギーによる電力供給を管理するシステムを持ち、個々の風車をつないだウィンドファーム（風力発電所）を運用している。個々の風車の制御装置をつなぐネットワークは、デュアルリング型の構成を取っており、A系ループ（プライマリ）とB系ループ（セカンダリ）がある。A系ループで切断があれば、瞬時にB系ループに切り替わり、また、A系、B系が同時に切断する事態が発生しても、ノードからの折り返し機能により復旧することが可能なネットワークであった。

以下、障害のあったC風車の回線を中心に、A系ループ、B系ループのそれぞれのノード（結線）を持つ回線を往路回線、ノード（結線）を持たない方を復路回線と呼ぶことにする。

ある日、A社のウィンドファーム内の風車の制御装置をつなぐネットワークで、障害が発生した（図3.30－1）。上述の通り、A社のネットワークは、デュアルリング型なので、A系ループの障害が起きた時点で、B系ループに切り替わろうとしたが、A系、B系ともに接続断が発生した（図3.30－1①）。さらに同時にノードがない復路回線も接続断となり（図3.30－1②）、折り返し機能によって、C風車とD風車間のすべての風車がネットワークから切り離された（図3.30－1③赤点線枠）。

C風車の現場に担当者が駆けつけ、C風車でケーブルの断線を発見して再接続した。その結果、ネットワークは復旧し、A社は、運転を再開することができた。

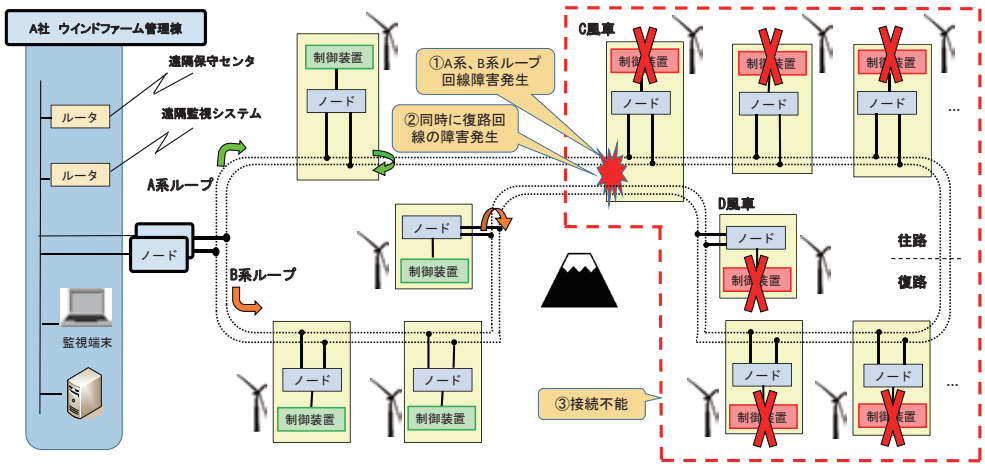


図3.30－1　障害状況

原因

直接原因は、工事作業員がC風車の建屋の中で「風車保守作業」を行った際に、ケーブルを誤って切断したことであった。2重化されたケーブルの両系が断線され（図3.30－2①）、同時にリングの復路にあたるケーブルが切断されてしまった（図3.30－2②）。切断個所は、工事のために一時的に両系のケーブルの往路（4回線すべて）と復路が建屋の中で束になって仮配線されていた個所であった。

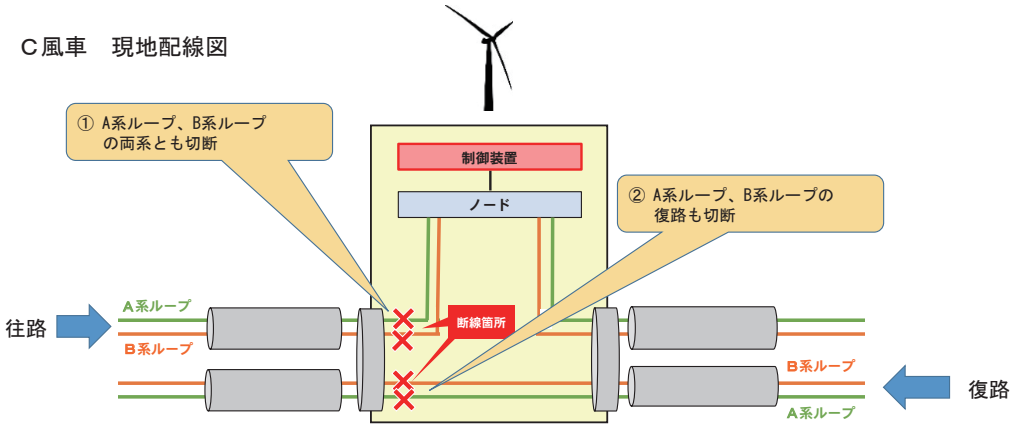


図3.30－2　障害原因

背後要因を探ると、工事発注者が施工者に工事の注意点を説明しておらず、さらに施工者の作業状況を確認していなかった。そのため、現場でどのような状況でケーブルが敷設されているか知らなかった。つまり、折り返しケーブルまでも建屋の壁に空けた1個の穴を通る同一の配管内に束ねた工事を行っていたという「見えないところでシングルポイントとなっていた」ことが根本原因である。

本来なら、発注者は、施工者に工事を発注する場合に、それぞれのケーブルがどのような役割を持っているかを丁寧に説明すべきであった。つまり、各ケーブルを束ねて敷設すると、複数ケーブルの同時切断が起こる障害発生の可能性が高くなるため、複数ケーブルを束ねるようなことがないように敷設条件を明確に提示する必要があった。それによって、均質な品質を保つことができたと考えられた。

対策

せっかくネットワークを2重化にしていても、敷設工事のときにケーブルを一緒に束ねておいては、今回のように意味をなさないことが起きてしまう。そこで、以下のような対策を行った。

(1) 緊急対策の実施

全風車の建屋内のケーブルの設置状況、敷設状況の点検を行い、今回の切断個所と同様に集線されている個所では、配線を分離した。

⑵ 2 重化構成になっているが両系切断になりうる個所の点検・変更

ウィンドファーム構内のケーブル敷設で、2 重系となっているが両系切断となりうる個所の点検・変更を行い、その分離を 2 つの方法で行った（図 3.30 – 3）。

- ケーブルの敷設を A 系ループのシールドと、B 系ループのシールドに分離し、同時に断線にならないようにした（図 3.30 – 3 ①）。
- 建屋内の基幹ループの往路と復路の両方が引き込まれている建屋では、のケーブルのシールドを往路（A 系ループ、B 系ループ）と、復路（A 系ループ、B 系ループ））とに分離した（図 3.30 – 3 ②）。

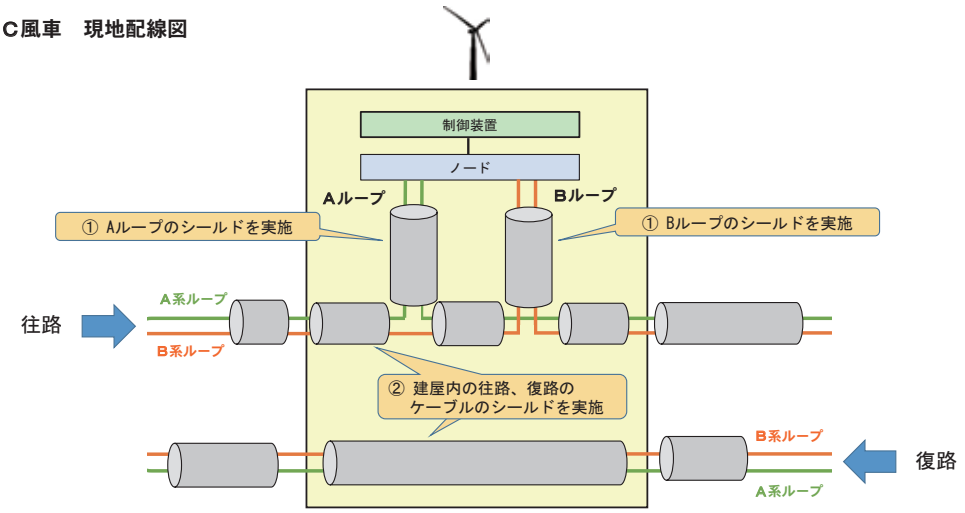


図 3.30 – 3　対策

⑶ 施工標準の見直し

ネットワーク構成の強化および故障時の信頼性確保のため、②の対策を盛り込んだ「施工基準」を設定し、社内担当者はもちろんのこと、施工業者にも説明をし、理解してもらった。

⑷ 将来のネットワークの見直し

今回の対策を行っても、今後両系ともに障害が発生しないとは限らないため、以下の 4 点を将来の設計時に見直すことにした。

a) 大規模災害を想定した対策

分離したとはいえ、大規模な災害などを考慮すると、ループの往路と復路を身近なルートで施設するのは、未だ危険が残るので、さらに敷設ルートの距離を空けること（別ルート）を検討する。

b) ネットワーク障害マニュアルの作成と研修、訓練

今回の障害は、日常の運用保守で障害訓練を行っていれば、事前に気づくこともあったと考え

られた。そこで、ネットワーク障害時の対策マニュアルを作成し、そのマニュアルを使った研修、障害訓練を行うことを検討する。

c）ネットワークのシングルポイントの点検、検討

システムのネットワーク全体を見直したとき、シングルポイントの個所を点検し、装置の重要度に合わせて、2重化にすべきかどうかを検討する。特に、確認できない（見えない）個所をなくすことが必要である。

d）監視機能の強化

ウィンドファーム管理棟で風車の制御装置の状況が監視できない事態が発生したときの対策として、別回線を使った監視用ネットワークを検討する。

これらの将来の設計段階でのネットワーク構成の見直しを考えると、図 3.30－4 のようになる。

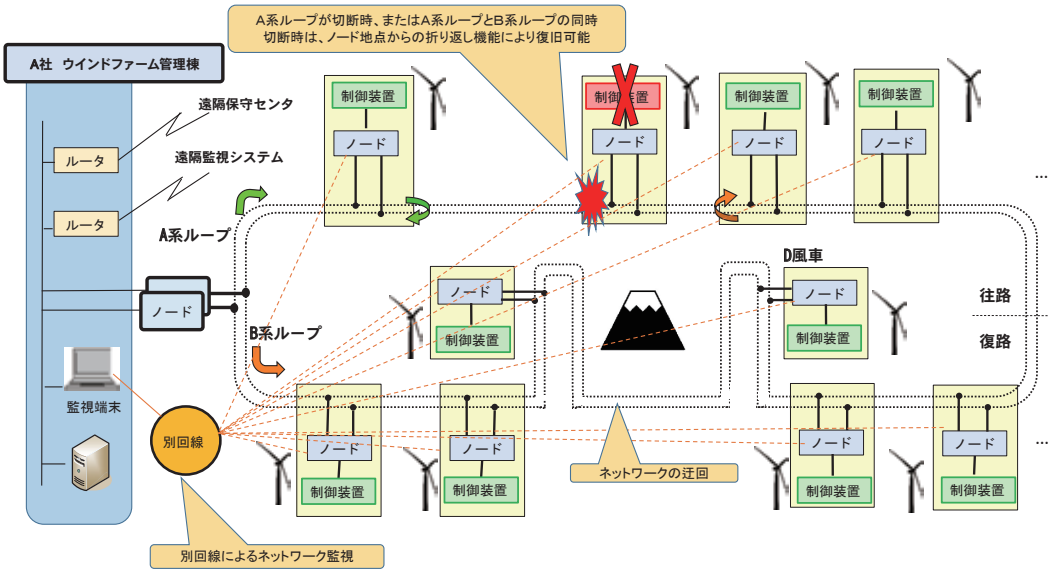


図 3.30－4　将来に向けたネットワーク

効果

ネットワークが現場でどのような物理的形状で敷設されているのか、物理的な2重化が機能するような形状になっているのかを現地確認することにより、信頼性は担保される。

また、ケーブルを2重化している目的を、「施工基準」として明確にすることにより、「2重化しても意味のない」工事を防ぎ、どの敷設業者が敷設を行っても、均質な品質を保った工事を行うことができる。

教訓

ネットワークは、現場でどのような物理的形状で敷設されているのかを確認しないと、「2 重化しているから大丈夫。」とは言えず、シングルポイントを作っていることがある。この教訓は、「ネットワークは現場での敷設状況を確認することが、設計時の品質を担保するために必要である」ことを教えてくれる。

3.31 障害対策マニュアルに関する教訓（T31）

教訓 T31　復旧手順は、システムとその環境の変化に対応させ常に最新に！

問題

A社では、各工場に置かれた制御装置を専用回線でつないだネットワークを構築していた。

ある日の早朝、A社の制御装置の稼働状況が集中監視センターから確認できなくなった。

監視センターの監視員が原因を調査したところ、ネットワーク障害が発生していることが判明した。その後センターのルータが故障していることが判明し、そのルータの交換を行い、昼頃ネットワークを再度立ち上げた。

その後、監視員は、障害対策マニュアルに従い、稼働再開のため「全リセット」機能（各制御装置の障害復旧後に障害時の仕掛中の稼働情報をリセット）を実行した（図 3.31－1①）。しかし、正常に作動せず、さらに制御装置のいくつかがハングアップ状態となり、復旧が大幅に遅れた。

制御装置には新型機種と旧型機種が混在していたが、障害となったのは旧型の制御装置であった（図 3.31－1②）。

そこで監視員は現地に赴き、ハングアップ状態の旧機種の制御装置に対して1台ごとに再立ち上げ（リセット）を実施した（図 3.31－1③④）。

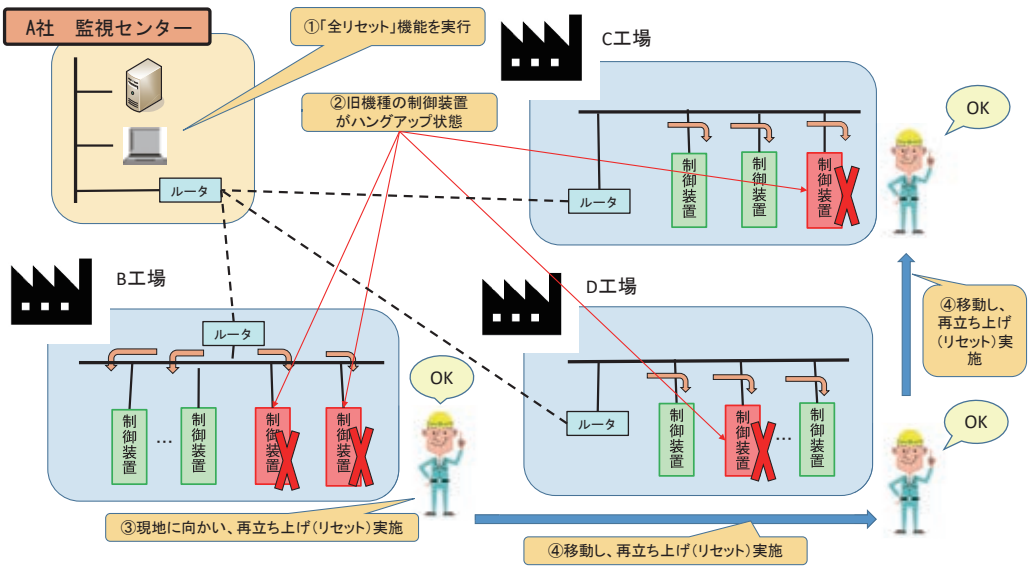


図 3.31－1　障害状況

原因

直接原因は、障害対策マニュアルの記述が曖昧であり、本来使用する必要がなかった機能を実行したことであった。

監視員は、障害対策マニュアルにしたがって一連の操作を行った。マニュアルの手順では、「全リセット」機能を使用するようになっていたが、この機能は、今回のようなすべての制御装置が正常に停止した場合には使う必要がなく、ネットワーク復旧後には、通常の立ち上げ時に行っている制御装置への「稼働開始」機能を実行すればよかった。また、「全リセット」機能は、システムの初期リリース時に保守用として作られて以来、全く使われておらず、その後の制御データの増大にともなっても、システム拡張や機能追加時にもテストされていない機能であった。当然、データ量増大時の旧型制御装置の動作性能への影響についても十分な確認テストが行われていなかった。

根本原因は、本来使用すべきでない機能が障害復旧手順として障害対策マニュアルに記載されており、そのような誤った記述のマニュアルが更新されずに放っておかれたことによるものであった。

システム障害の原因がすぐに突き止められ、その原因から復旧の道筋が明確になった場合の復旧作業は、ある意味簡単であろう。しかし、今回のように当初原因が分からず、その原因をつかむのに時間がかった場合や、想定していなかった障害の場合、その復旧手順は複雑になり、従来のマニュアルの手順ではうまくいかない事態が発生する。

まとめると、以下のような課題が存在した。

⑴ 今までにない（想定されない）事象が起きた場合のマニュアルやオペレーションの在り方

- 二次障害を回避するオペレーションはあったのか
- オペレータは常にマニュアルに従うべきか

⑵「滅多に使わない機能」の管理、運用方法

- その機能は，オペレーションマニュアルでどのように扱うべきか
- その機能は、事前に，どこまでテストされるべきであるか

このような課題に対して、関係者が集まって議論することが必要であった。

対策

システム障害対策マニュアルは、障害発生時に原因をどう突き止めるか、そこからどのように復旧するかをまとめている。

今回のシステム障害は、その頼るべきマニュアルが、障害時に使える記述になっていなく、また内容も管理されていなかったので、A 社では、「障害復旧改善に向けた障害対策の最新化する運用」を設定することにした（図 3.31－2）。

具体的には、障害対策マニュアルを常に最新版の状態に保てるように、障害対策マニュアルを維持

管理する運用ルールを定め、実践していく体制を作った。

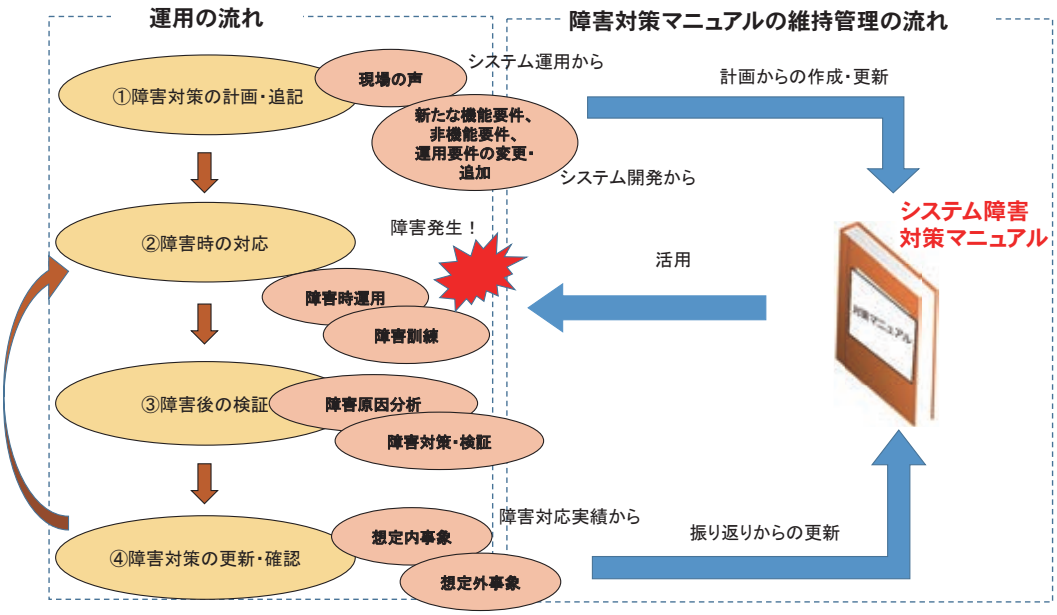


図 3.31－2　運用と障害対策マニュアルの維持管理の流れ

⑴ 障害対策の計画・追記

- 障害対策は、障害時に障害対策マニュアルのどこを見れば良いか、どこから作業を行えば良いか、どの手順（フローチャート）で行えば良いか、が担当者にすぐにわかるものとする。
- そのためにも、現場の声を聴き、現場の声を反映させる障害対策マニュアルとする。
- 障害対策マニュアルが正確であることを確認する、検証、障害訓練を計画する。

⑵ 障害時の対応

- 障害対策マニュアル通りに実行する。復旧しなかった場合は、エスカレーションを実行する。
- その対応が，障害対策マニュアルの手順通りにできたかどうか記録する。
- 障害対策マニュアルにない想定外の事象にどう対処したか記録する。

⑶ 障害後の検証

以下の観点などを検証する。

- その対応は、正しい行動だったのか
- 障害対策マニュアルにない行動だった場合、なぜそのように行動したのか
- その対応がマニュアル通りとしても、正しい行動だったのか
- その機能が事前にどこまでテストされていたのか

⑷ 障害対策の更新・確認

- 新たな障害に対応した後に、その際に実施した対策を追加する場合は、手順が明確になっている「想定する事象への対応」と、万が一、手順通りにいかなかった場合の「想定外の事象への対応」の両方についてのフローを障害対策マニュアルに記述し、システム障害時の対応を検証する。

  「想定外の事象対応」を記述することは、例えば「想定外の事象」が起きた場合は、緊急対策本部を立て、関係者にすぐ連絡する、お客様対応部門への連絡、などの危機管理対応を障害対策マニュアルに明記することを指す。
- 制御データ量の増大、新規機能追加、機能変更が発生した場合についても、同様に、「想定する事象対応」と「想定外の事象対応」の両方について、障害対策マニュアルに記述し、そのマニュアル通りにシステムが復旧することを検証する。

効果

A社は、今回のネットワーク障害時における稼働再開までの手順とマニュアルの見直しを行い、常にマニュアルを中心にした、システム運用を開始した。それにより、以下の効果が生まれた。

a) システム運用と、障害対策マニュアルを関連付けながら行うので、システム障害対策が運用の中心として位置づけられるようになり、障害対策マニュアルが置き去りになる事態を防ぐことができた。

b)「機能の解説」から「障害対策」を中心とした障害対策マニュアルへ修正したことにより、障害対応の手順（フローチャート）を明確にすることができた。具体的には、以下のような効果が表れた。

- 障害時は、マニュアルのフローにしたがって実施できるようになり、手順も確認しやすくなった。
- 新たな障害パターンについての追加記述も関係者の合意を取りながら行えるようになった。
- 監視員の対応が明確になった。
- 障害対策の訓練や教育が、マニュアルを中心に行うことができるようになった。
- 障害発生時に連絡すべき関係者が明確になった。

教訓

システム障害対策マニュアルは、どのシステム部門も管理しているが、システムを取り巻く環境の変化やシステム更改に合わせたマニュアルの更新が追いつかず、後回しにされる場合もよくある。そのため、いざ障害が発生しても、マニュアル通りに行うことを躊躇して対応が遅れたり、いざ行うと当該事例のような二次的障害を引き起こしてしまったりと、システム障害の影響範囲を拡大してしまう。

この教訓は、「システム障害対策マニュアルは常にシステム運用の中心として管理すべきもの」であり、そのためには、「システム障害対策マニュアルは、常に最新にしていくべきこと」を教えてくれる。

3.32 周期起動を持つシステムに関する教訓（T32）

教訓 T32

周期処理、「時間」と「変化」を監視せよ！

問題

A 社の物流センターの制御システムは、同期して連係動作する多数の PLC（Programmable Logic Controller）[14] により構成され、新型 PLC と旧型 PLC が混在していた。

ある日の早朝、A 社工場内の制御ネットワークで障害が発生した。

ネットワーク復旧後、運転再開のため、「再稼働初期設定」機能を実施した（図 3.32－1 ①）。「再稼働初期設定」機能とは、監視制御サーバから各 PLC の制御装置に対して、その保持する最終稼働状況（稼働が仕掛状態となっていた個所）の検索を行い、その中断個所を初期化し、各制御装置の開始時点の同期を合わせる機能である。この機能を実施したところ、旧型の PLC の制御装置が正常に作動せず、ハングアップ状態となり（図 3.32－1 ②）、その原因究明と対処のために制御システムの復旧が大幅に遅れた。

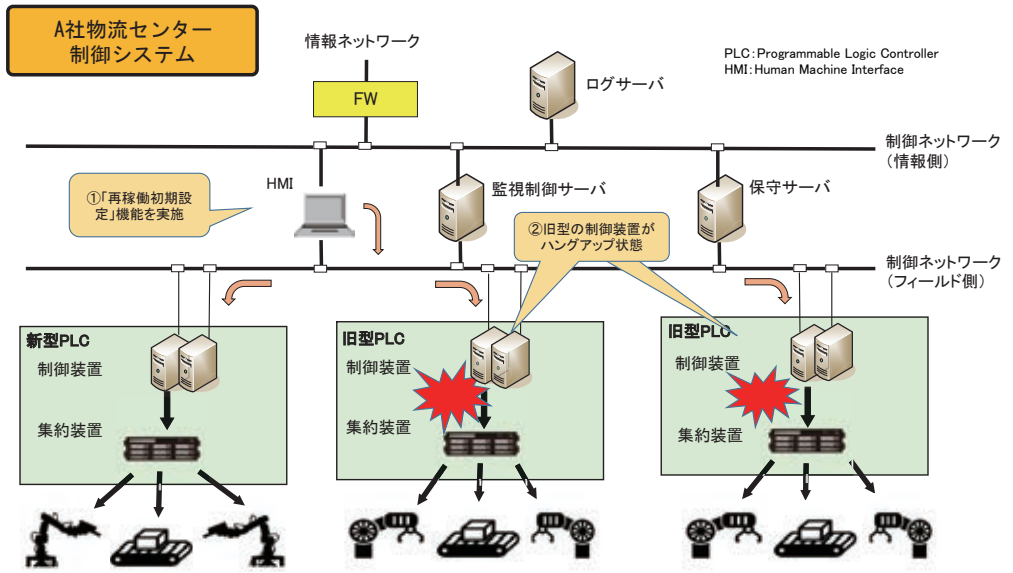


図 3.32－1　障害状況

[14] リレー回路の代替装置として開発された制御装置。自動機械の制御に使われる。

原因

A 社の制御装置のソフトウェアは、制御ネットワークからコマンドや制御情報をイベント発生時に受信・登録する処理「イベント処理」と、周期起動でコマンドや制御情報を受け取り、集約装置を稼働させたり、制御情報を更新させたりする処理「周期処理」の 2 つで構成されている（図 3.32－2）。監視制御サーバから発せられた「再稼働初期設定」コマンドは、イベント処理により登録され、周期処理により取り出されて実行される。

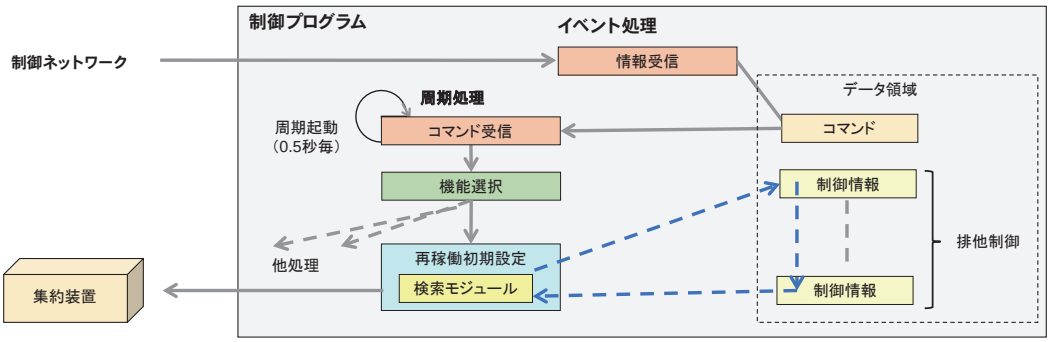


図 3.32－2　制御装置　ソフトウェア構造

制御システムの復旧が大幅に遅れた直接原因は、今回の「再稼働初期設定」機能の実施によって、旧型（20 年以上稼働）の PLC の制御装置上での周期起動で開始する「コマンド受信」から「再稼働初期設定」の検索までの処理が、周期時間内（1 サイクル当り 0.5 秒）で処理が完了しなかったため、以降の周期処理のコマンドが次々と滞留し、その制御装置に高負荷状態が発生し、ハングアップ（無応答）状態となったことであった。

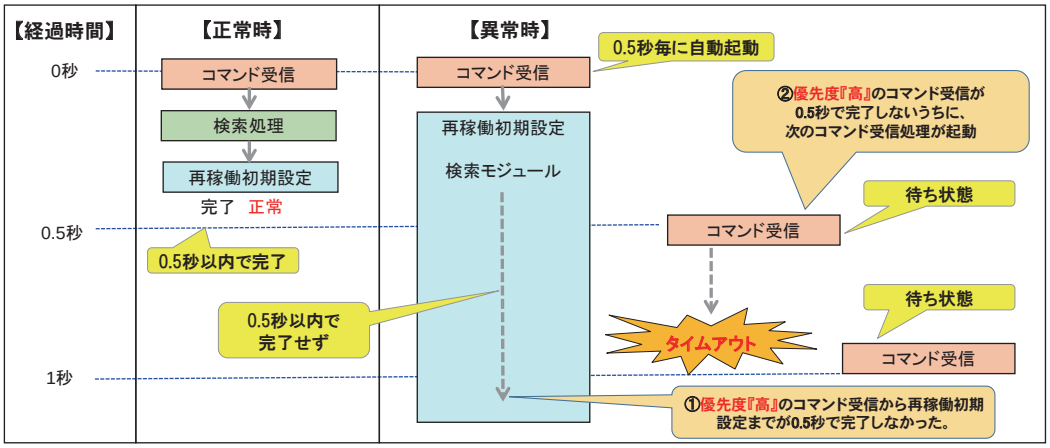


図 3.32－3　障害原因

監視センターから実行した「再稼働初期設定」機能が、各制御装置に命令を出した。旧型 PLC の制御装置は、「コマンド受信」から「再稼働初期設定」を実行した。それは、本来 0.5 秒以内で行える処理であったが、今回は、「作業開始時点から現時刻（昼過ぎ）までの、対象となる制御情報」を検索する際に制御情報が想定数以上存在したため、0.5 秒以内で完了することができなかった（図 3.32－3 ①）。そのため、0.5 秒ごとに「コマンド受信処理」が起動され、これらが連鎖的に遅れたため、旧型 PLC の制御装置は、ハングアップ状態となった（図 3.32－3 ②）。

周期処理は、ひとつの完結する処理群が周期時間内に完了することが条件であり、新たな機能や、制御すべき機器が増えた場合でも、その周期内で処理が完了することは必須である。すべてのコマンド処理が正常に稼働するかを事前に実機で確認すべきであった。

「再稼働初期設定」機能は制御システム導入時に作られたものだが、障害時にだけ使う機能であったため、長年このコマンドを実行していなかった。その間、集約装置につながれた機器の増加や、流れる制御データの増大、ネットワークの長時間の停止による検索処理時間の増大を考慮する、などの実行環境の変化に気づかなかったにも関わらず、障害マニュアルでは、使われるツールとして記載されていた。

したがって、根本原因は、周期処理を持つ制御システムについて、その周期時間を管理しておらず、新しい環境の変化があった場合の周期時間の見直しや、すべての PLC 機器について動作確認を行っていなかったことである。

また、A 社の物流センターの制御システムで使用している PLC は、コンピュータ部分のようなライフサイクルが比較的短い制御装置と、メカ部分のようなライフサイクルが非常に長い集約装置が一体となっている機器であった。そのため、機器全体の交換時期が設定しづらい装置であった。今回のような集約装置につながる現場機器の増設が行われ、また検索に時間がかかるような制御装置の処理速度上の問題がリスクとして潜在化し、最新の制御装置に交換しなくてはならない事態になっても、コスト面の関係からすべての PLC を旧型から新型に入れ替えることがなかなかできない状態であったことも根本原因の一つとしてあげられる。

対策

周期処理が含まれる制御システムの機能追加を確実に実施する対策として、以下の 3 点を実施することにした。

⑴ 周期時間管理

周期処理が組み込まれている制御プログラムで設定されている周期時間を管理するとともに、その周期時間を超える場合がないかを適宜監視する管理ルールを定める。このような周期時間管理は、制限値管理と同様な管理方法となる。

さらに、周期処理の周期時間を超え得ることを考慮したプログラムロジックも検討することとした。

（例）立ち上がった周期処理は、前の周期処理が完了していない状態で、かつ 0.5 秒を経過した場合、実際の処理を行わずに終了し、結果を監視に通知（または監視が検知）する仕組みとする。

⑵ 新機能追加時の運用ルールの策定

- 機種ごとの仕様をもとに周期時間を評価した上で、すべての機種について動作確認を実施する。
- 障害対策を設計時に検討し、マニュアルに追加する。そして、検証時は、どこまでの検証を行うか、関係者と議論し、「テスト一覧」を作成する。それをもとに、できない検証についてのリスク評価を行い、その対策の手順を明確にし、対策マニュアルとして作成する。

以上、周期処理を持つ制御システムの機能追加の手順をまとめると、図 3.32－4 のようになる。

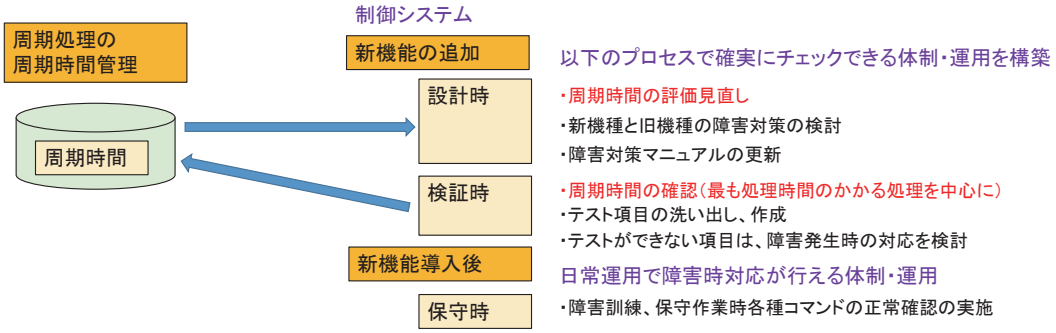


図 3.32－4　周期時間管理と制御システムの機能追加との関係

⑶「旧機種の装置を新機種の装置に更新しやすい」構成を作る

今回の場合、遅い PLC の制御装置を最新の高速な PLC の制御装置に置き換える対策もある。しかし、A 社の PLC は、制御装置と集約装置とが一体となっているため、製品寿命の短い制御装置だけではなく PLC 全体を入れ替えなくてはならず、高コストとなっていた。

近年、この制御装置と集約装置の内部 I/F（インターフェース）が、独自プロトコルでなく、汎用的なイーサネット（Ethernet）になった。

そこで、制御装置だけを入れ替えられるように、両装置間の I/F がイーサネットで、制御装置 - 集約装置分離型の PLC をベンダに製造してもらうことを要請した。この分離型の PLC を導入することで、ベンダの制約が緩和されるとともに、新制御装置だけの入替えもできるようになった（図 3.32－5）。

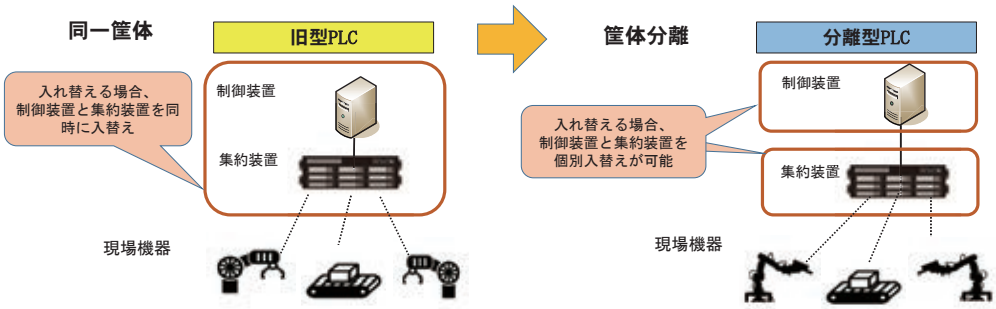


図 3.32－5　分離型 PLC

効果

制御システムの周期時間は、設計時にすべての処理時間を考慮して決定される。当然、周期時間が短ければ処理効率がよくなるが、短すぎると今回のような障害を引き起こしたり、空振りや CPU 負荷増となったりする。そのようなバランスの上で決定された周期時間は、制御対象機器が増えたり、制御データが増大したり、新規機能追加や新型機器が導入されたりすることにより、見直す必要性が生ずる。この教訓は、制限値管理の「制限値」と同様に、「周期時間」の変化を見逃さない周期時間管理の重要性を気づかせてくれる。

制御システムの周期処理に新旧の制御装置が混在する場合は、性能差が生ずることから、システム全体の構成に注意が必要である。その際には、製品の入れ替えを容易にする製品構成についても検討する機会を持つことができる。

教訓

一般的に、制御システムは、20 年以上使うこともよくあり、使用中に部品の供給が中止になったり、ソフトウェアやハードウェアの保守契約が切れたりすることが多い。そのような状況において、コストの面から一度にシステム内のすべての装置を更新できないことから、同一システムの中に、徐々に新型の高性能な機器が導入され、新旧の機器間で処理時間に違いが生ずることが多々起こる。また、長期の間に運用方法も変わり、今まで使われてこなかった機能が使われるようなことも起こる。その中で、周期時間は、制限値と同様に変化を見逃さない管理方法が必要である。具体的には【教訓 T4】で述べている変化点管理を対策とすることができる。

【教訓 T4】システムに影響する変化点を明確にし、その管理ルールを策定せよ！

制御システムは、比較的短期のうちに性能が上がっていくコンピュータ部分と、長期にわたって使用していくメカニック部分が混在する機器がある。このような状況も踏まえ、短期間で交代可能なコンピュータ部分と長期間使用するメカニカル部分を分けた機器構成も検討する必要がある。

この教訓は、そのような制御システムに対する運用管理とシステム構成の在り方を教えてくれる。

3.33 排他制御に関する教訓（T33）

教訓 T33

入念な方式設計と多段階の確認は当たり前、個人情報を扱う場合には特に排他制御に気をつけて

問題

イベント運営会社Ｉは、顧客からのイベント参加予約を複数の代理店を介してネットで受け付け、イベント管理システムによりその情報を一元的に処理している。各代理店は、それぞれの登録された顧客からのイベント予約を仲介している。（図 3.33－1 参照）

ある日、代理店Ａの担当者が、予約が締め切られたばかりのイベントの予約者一覧リストをＩ社のイベント管理システムからダウンロードしたところ、リストに同代理店の顧客以外の情報が含まれていた。同じ頃、代理店Ｂの担当者も同様の作業を行ったところ、同代理店の顧客以外の情報がダウンロードしたリストに含まれていた。それらの情報には、個人情報として扱われるものも含まれていた。

不審に思った両代理店の担当者は、前後してイベント管理会社Ｉのコールセンターにその状況を伝えた。同時期に２つの代理店からダウンロード情報の異常を伝えられたＩ社のシステム担当は、情報漏えい（セキュリティ・インシデント）が発生したと認識し、同社の情報セキュリティ規定に基づく対応を開始した。

まず、イベント管理システムのダウンロード機能を即座に停止した後、発生原因の調査を開始した。また、同様の事象が過去に発生していた可能性がないか、システムログの解析を行った。並行して、同社幹部が両代理店に出向き、今回の件について謝罪するとともに、ダウンロードしたリスト情報の廃棄を依頼し、その実施を確認した。その後、Ｉ社は、規定に基づき、今回のインシデントの発生をホームページで公表した。

幸い、過去に同様の事象が発生していた形跡は確認できなかった。また、発生原因が突き止められ、改修の後、発生から３日後にダウンロード機能の提供を再開した。

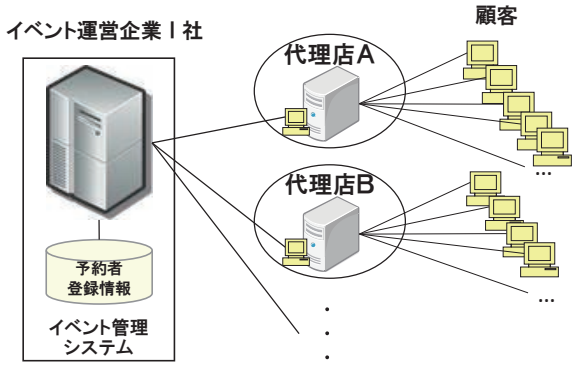


図 3.33－1　システムの全体構成概要

原因

(1) 直接原因

本セキュリティ・インシデントの直接原因は、I社のイベント管理システムのソフトウェアの不具合であった。イベント予約者一覧リストのダウンロード処理において、リスト情報転送用の一時ファイルの競合対策が不十分であった。具体的には、次のように処理が行われていた。（図 3.33－2 参照）

代理店からのリストのダウンロード要求に対し、一旦、ダウンロードデータを一時ファイルに書き込んだ後、所定の契機で一時ファイルの内容を要求元に転送する処理になっていた。ところが、複数の代理店から同タイミングで要求されるケースへの対策が不十分であり、要求の競合を考慮せずに一つの一時ファイルを排他制御無しで使用するようになっていた。そのため、ほぼ同時にダウンロード要求のあった 2 つの代理店の予約者情報が、同じ一時ファイルに混在して書き込まれてしまった。その後、その内容がそのまま両代理店に転送された。

なお、I社のイベント管理システムは、5 年以上運用していたが、今回のような問題が発生したのは初めてであった。

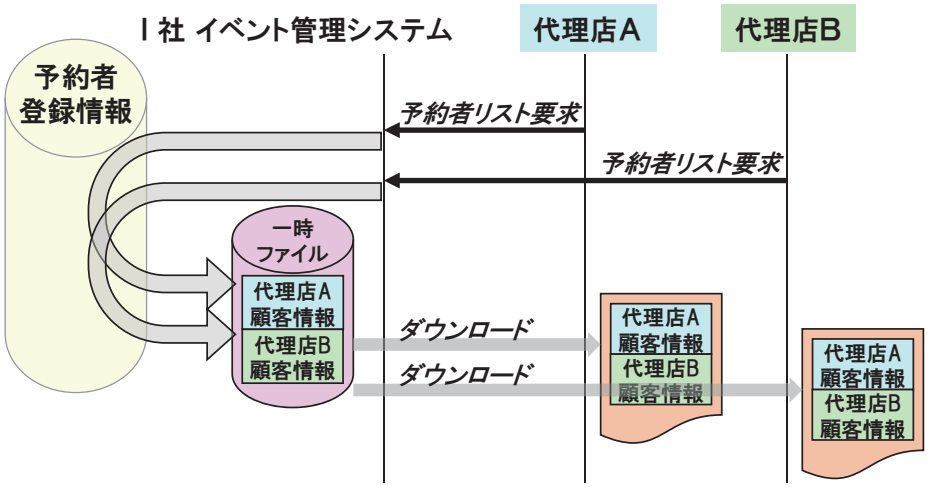


図 3.33－2　問題となった処理の流れ概要

(2) 不具合の混入原因

一時ファイルの競合対策が不十分であった原因を分析するため、関係者にヒアリングを行った。その結果、システム開発当時の当該処理の設計担当者には、自身のそれまでのシステム開発経験に基づく思い込みがあり、それが今回の問題を引き起こす要因となったことが判明した。

その担当者のそれまでの設計経験では、オペレーティングシステム（OS）あるいはミドルウェアがリソース競合対策の機能を有していたので、排他制御の機能をアプリケーションレベルで実装する必要はなかった。今回の対象システムの動作環境では彼の経験とは異なり、排他制御のための処理を担当プログラムに組み込む必要があったが、上記の理由により、それを自ら作り込むということには思い

が至らなかった。

また、当時の開発チーム編成時には、開発対象システムの内容や開発プロセス・環境に照らした、各担当者のスキルや経験についての細かな確認が行われていなかったことも分かった。

(3) 不具合が開発時に摘出できなかった理由

システム設計時には、第一に、リソース競合対策（排他制御）を始めとする処理方式等が慎重に考慮されるべきである。個人情報を取り扱う場合には特に、細心の注意が必要である。次に、たとえ設計時にリソース競合対策（排他制御）を適切に実装していなくとも、その後のレビューやテストにおいて、そのことは発見されるべきである。それが今回の問題発生まで発見されなかった理由を追求するため、I社では、システム開発時の膨大な記録を分析した。その結果、以下の（a）～（d）に示すような問題点が判明した。

（a）I社のシステム開発標準における設計指針には、リソース競合対策に関する記載がなかった。

過去の経験から、一時ファイルの競合対策（排他制御）を担当プログラムに実装することに思いが及ばなかった設計担当者は、設計指針によっても、その必要性に気づかされることはなかった。

リソース競合対策は、情報処理システムにおける常識ではあるものの、念のための注意喚起の意味から、少なくとも項目だけでも設計指針の中に記載されるべきである。また、リソース競合対策を含む方式設計を入念に行った上で、アプリケーション設計に進むべきである。

（b）I社のシステム開発標準プロセスでは、設計指針／コーディング指針と設計レビュー用チェック指針／コードレビュー用チェック指針とは、それぞれ全く同じものを兼用し、指針以外の観点でのレビューはなされなかった。

競合対策が不十分な設計がなされたが、設計指針にはしたがっていたため、設計レビューで不備が指摘されることはなかった。コーディングについても、指針にしたがって設計書の内容をそのままコード化したため、コードレビューで不備が指摘されることはなかった。

設計者／プログラマは、設計指針／コーディング指針に基づいて作業する。その作業結果に対し、設計レビュー用チェック指針／コードレビュー用チェック指針に基づいてレビューされるわけであるが、設計指針と設計レビュー用チェック指針とが全く同じものであると、設計指針に基づいて忠実に設計する限り、レビューにおいて指摘されることはあり得ない。コーディングについても同様である。レビューでは、設計指針／コーディング指針にしたがっているかという観点に加え、より多様な観点からチェックされるべきである。

（c）I社のテスト項目抽出基準は、設計書の内容のみに基づいてテスト項目を導くものであった。

設計書には共有リソース競合対策（排他制御）に関する記載が一切なかったので、その処理に対応するテスト項目が抽出されることはなかった。今回問題となった事象が発生するのは、複数の代理店がほぼ同時にダウンロード要求を行った場合である。この場合には、システム実装上の共有リソースの競合が発生し得る。通常、設計書の記載とは関係なく、このようなシナリオや

ユースケースを想定して共有リソース競合対策に問題がないかを確認するテストを行う。すなわち、「複数代理店からランダムに要求が来るということは、リソースの競合が起き得るな」ということを想定するはずであり、そう想定したならば、リソース競合対策の妥当性を確認するためのテスト項目を設定するはずである。今回は、そのテストが実施されなかった。また、テスト項目を含むテスト仕様書に対するレビューは行われなかった。

なお、このような微妙なタイミングに絡むケースのテストを実地で行うことは一般に困難であるが、疑似的な環境で負荷テストを長時間実施することにより、発生確率を高めることはできる。

(d) 対象システムの開発においては、第三者によるレビューが実施されていなかった。

レビューは開発チーム内で行われ、その主な観点は、「上位設計書の通りに実装されているか?」てあった。そのため、今回は上位設計書に誤りがあったが、それが摘出されなかった。レビュー者のうち一人でも、複数のダウンロード要求を同時に扱う処理のレビューを行っているということを意識していれば、その処理における共有リソースの有無やその競合対策について思いが及んだものと思われる。

一般に、知見や経験の豊富な第三者やシステム運用担当者などが、それらに基づいてチェックすることにより、誤りの摘出率は高くなる。レビュー対象がどのようなシナリオ（ユースケース）に対応する処理かということを意識しさえすれば，それに対応した注意事項や関連した過去のトラブル事例に思い当たることは間違いない。

(4) 保守時での不具合発見の機会逸失（セキュリティの観点）

I社のイベント管理システムでは、その初期開発時には、代理店によるイベント予約者一覧リストのダウンロード機能において個人情報を扱ってはいなかった。ところが初期リリースから数年後に、当該機能を更新していた。このときの更新内容は、ダウンロード対象項目の拡張であり、ここで初めて、ダウンロード情報にいわゆる個人情報が含まれることとなり、セキュリティに対する要求レベルが上がった。にもかかわらず、I社では十分な影響確認が行われなかった。もし、この時点で影響確認がセキュリティ要求レベルにふさわしい内容で実施され、それによってソフトウェアの不具合が発見されていれば、今回のインシデントは発生しなかった。

実際には、単なる情報項目の追加に対して、一般に、それを扱う処理のロジックは影響されないことから、当該システムにおいて、この機能更新時には十分な確認が行われなかった。もし、「セキュリティ（個人情報の取扱い）」という点を重視し、更新対象の情報項目にかかわる処理パスを方式設計にまで遡ってすべて徹底的にチェックしていれば、予約者一覧リスト情報を一時ファイルに書き込む処理の妥当性確認も確認対象となったはずであり、あるいは、競合対策漏れの不具合を発見できたかもしれない。

なお、I社のシステム開発標準には、セキュリティにかかわる処理に関し、このようなチェックを行う具体的な規定はなかった。

対策

直接原因への対応として、I社は早急に、次の不具合修正を行った。

(1)　イベント管理システムのソフトウェアの不具合である、イベント予約者一覧リストのダウンロード処理における一時ファイルの競合対策処理（排他制御）を正しく修正

また、根本原因への対策として、I社は、全般的に、セキュリティ（特に、個人情報）を扱う場合には入念な設計・確認が必要な旨をシステム開発標準に明記した。その上で、各原因に対応し、システム開発標準を次のように改訂した。

(2)　開発対象システムの内容や開発プロセス・環境に照らして、開発チームメンバ候補のスキルや経験について確認し、全体として必要条件をカバーできるようチームを編成

(3a)　設計指針、及びコーディング指針に、リソース競合対策（排他制御）に関する事項を追記

(3b)　設計／コーディングレビュー用チェック指針に、過去のトラブルに基づく確認項目を追加

(3c)　テスト項目リストもレビュー対象に追加

(3d)　一定規模以上のシステムや重要度の高いシステムにおいては、知見・経験の豊かな第三者によるレビューを義務化

(4)　セキュリティ（特に、個人情報）にかかわる更新時には、当該システムの重要性が一定以上の場合に、更新対象のセキュリティにかかわる部分に関連する全処理の再トレースに基づく確認を義務化

以上の措置について、今回のインシデントの概要とともに、I社内の関係者に周知した。

効果

I社のイベント管理システムでは、対策実施後、セキュリティ・インシデントは発生していない。

また、システム開発標準の改訂後、I社の開発するシステムに同種のインシデントは発生していない。

なお、I社では、改訂したシステム開発標準に基づき、システム開発チームの編成時に、メンバの経験やスキルにより注意深く配慮するようになった。また、個人情報を取り扱う場合には、リソースの競合対策（排他制御）等について開発の各段階で確実にチェックする習慣ができていった。例えばレビューに関しては、どのようなシナリオやユースケースに対応する処理を確認しているのかを考慮するなど、レビュー対象範囲のみに集中するのではなく、より広い視点でレビューを行う傾向が高まっていった。

教訓

システムの開発段階で混入した不具合は、システム障害を引き起こしたり、出力の誤りという形で現れたりすることが多い。本教訓における問題は、個人情報の漏えいというセキュリティ・インシデントを招いたものである。最近では、個人情報の漏えいは社会的影響が非常に大きくなっており、その

十分な対策が求められている。したがって、個人情報を扱うシステムでは、その処理で重要な排他制御等について、開発の初期段階から保守段階に至るまで、その入念な設計・確認が必要である。

今回のケースでは、不具合の混入は担当者の思い込みによるものであるが、それがずっと摘出されず、システムのサービス開始以降何年も経過した後で初めてセキュリティ・インシデントという形で発現した。その理由としては、開発チーム編成や方式設計、レビュー方法、テスト項目等、様々な段階での要因が考えられる。

まず、開発の早期で入念な方式設計を行うことが最も重要である。特に、排他制御はセキュリティ対策の要の一つであり、動作環境を踏まえた慎重な設計が求められる。

不幸にしてこのような検討をすり抜けて混入した不具合は、開発チームあるいは第三者による多様な視点からのレビューにより、システムのリリース前に摘出することができる。特に、微妙なタイミングに絡むような実地テストが困難な処理については、レビューが欠かせない。第三者によるレビューが効果的であるが、コスト面の制約から小規模なシステムでは開発チーム内でレビューすることもある。多様な視点でのレビューを行うには、レビュアの知見や経験、関心や専門等が多様であることが重要である。すなわち、チーム編成やレビュアの選定がポイントの一つとなる。

特に、個人情報を取り扱うなどセキュリティがクリティカルなシステムにおいては、より慎重な確認が求められる。そのためには、方式設計時やレビュー時にセキュリティに着目したシナリオやユースケースを設定する必要がある。また、テスト項目に関する適切な抽出、レビューも重要かつ効果的である。

そして、これらを踏まえ、設計やレビュー等の関係者の拠り所となるシステム開発標準を適切に整備・保守することが重要である。

4

事例から見えてくる傾向

4.1　IT サービスマネジメント（ITSM）プロセス観点での分類と傾向
4.2　バックアップ切替え失敗の問題と対策（詳細説明）
4.3　ヒューマンエラーの問題と対策（詳細説明）
4.4　システムの高負荷／過負荷に関する問題と対策（詳細説明）
4.5　「注意すべき観点」に基づく障害の分類

事例収集・原因分析から教訓化の作業を通していくつか傾向が見えてくる。本章では、その内容について解説を行う。

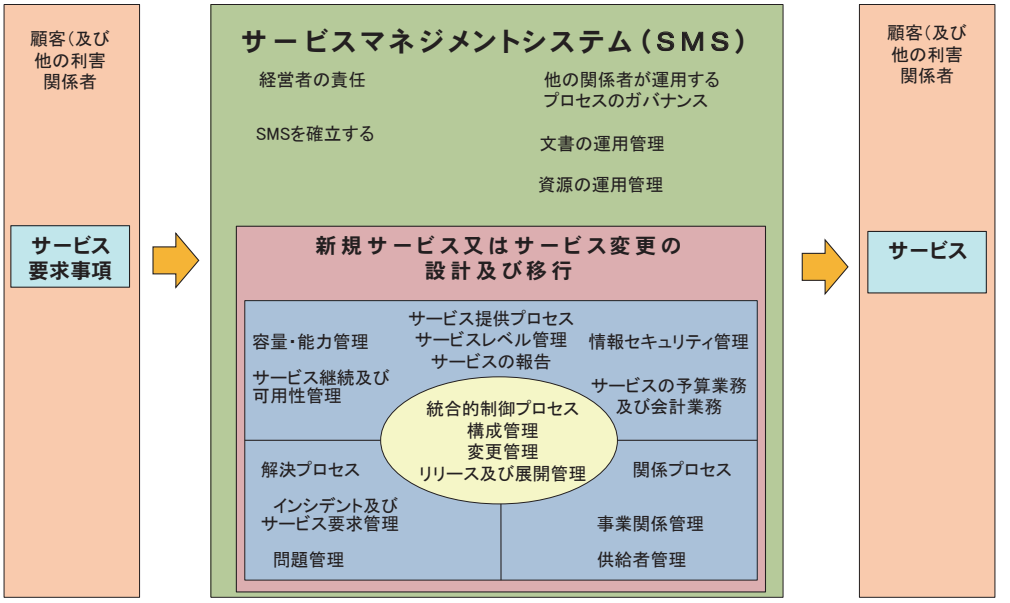
4.1 IT サービスマネジメント（ITSM）プロセス観点での分類と傾向

IT サービス提供者は提供する IT サービスのマネジメント（ITSM）を効率的、効果的に運営管理し、安定的なサービスを提供することが求められる。IT サービスマネジメントの仕組み（IT サービスマネジメントシステム：ITSMS）の仕様及び実践のための規範が JIS20000（ISO/IEC20000）として規格化されている。（文献 4.1－1）

JIS　Q20000 の規格群には、次に示す部編成がある。

JIS　Q20000-1　第 1 部：サービスマネジメントシステム要求事項

JIS　Q20000-2　第 2 部：実践のための規範



（JIS Q 20000-1:2012　より引用）

図 4.1－1　IT サービスマネジメントシステムのプロセス

ここでは、教訓集に登録されている教訓事例を対象として原因の分析を行った。改善が必要と思われる主要なサービスプロセスに●印、関連して改善を検討すべきサービスプロセスに△印をつけて整理している。（図 4.1－2 及び図 4.1－3 参照）

No.	JIS Q20000-1：2012より（●主な問題個所、△関連する問題個所）												
	5.	6. サービス提供プロセス					7. 関係プロセス		8. 解決プロセス		9. 統合的制御プロセス		
	新規またはサービス変更の設計及び移行	サービスレベル管理	サービス継続・可用性管理	サービス報告	容量・能力管理	情報セキュリティ管理	事業関係管理	供給者管理	インシデント管理	問題管理	構成管理	変更管理	リリース管理
G1	△						●						
G2	●							△					
G3	●	△											
G4				△				△	●				
G5					●		△						
G6											△	●	△
G7				△			△	△	●				
G8							●				△		
G9			△				●						
G10							△		●				
G11			△								●		
G12			△		●								
G13			△		●							△	
G14			△		●								
G15	△		△									●	
G16	△							△				●	△
G17	△	△	●					△					
G18		△	●				△			△			
G19							●					△	
G20	△							△				●	△
G21	△							△			●		△

図 4.1－2　ガバナンス・マネジメント領域の教訓と ITSM プロセス

No.	JIS Q20000-1：2012より（●主な問題個所、△関連する問題個所）												
	5.	6. サービス提供プロセス					7. 関係プロセス		8. 解決プロセス		9. 統合的制御プロセス		
	新規またはサービス変更の設計及び移行	サービスレベル管理	サービス継続・可用性管理	サービス報告	容量・能力管理	情報セキュリティ管理	事業関係管理	供給者管理	インシデント管理	問題管理	構成管理	変更管理	リリース管理
T1			●								△		
T2					△						●		
T3	●	△										△	
T4		△	●		△						△	△	
T5	△											●	
T6											●	△	
T7			●		△								
T8					●						△		
T9			●								△		
T10			△								●		
T11		●			△								
T12								△			△	●	△
T13	●	△										△	
T14					△							●	△
T15											△	●	
T16						△		△			●		
T17			△								●		
T18	●										△		
T19	●				△								
T20	△							●					
T21	△										△	●	
T22			●		△						△		
T23			●		△				△				
T24	△		●				△		△				
T25			△							●			
T26	△										●		
T27		△						△		●			
T28				△				△				●	△
T29	△				●						△	△	
T30			●							△	△		△
T31		△					△		●		△		
T32					●				△		△		△
T33	△					●		△					△

図 4.1－3　技術領域の教訓と ITSM プロセス

本教訓集に掲載した各教訓の障害発生に繋がったプロセス上の問題個所（●と△）の ITSM 観点での分類を試みた。

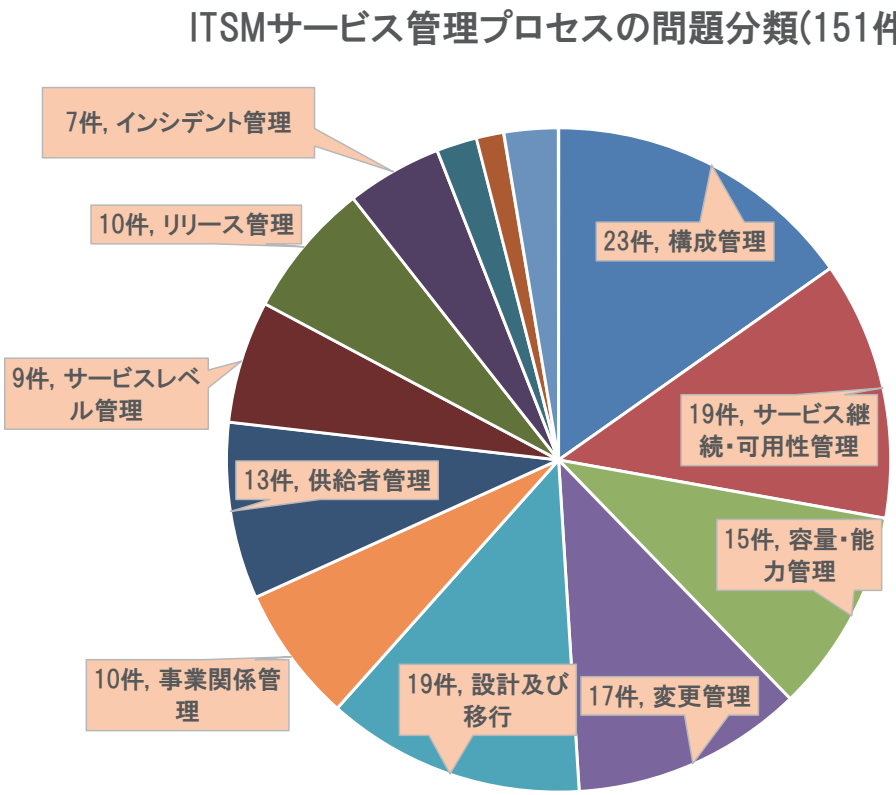


図 4.1－4　ITSM サービス管理プロセス別の問題の分類

教訓集に取り上げた障害事例を ITSM サービス管理プロセスのプロセス別に分類すると「構成管理」、「サービス継続可用性管理」、「変更管理」、「容量・能力管理」、および「サービスの設計・移行」のプロセスにおいて問題が多いことがわかる。

IT サービスの安定的な提供に向けて、特にこれらの管理プロセスを組織としてマネジメントする体制・運用管理ルールの整備がポイントとなる。

4.2　バックアップ切替え失敗の問題と対策（詳細説明）

バックアップ切替えが成功しない様々な状況を考慮して、対策を立てよ

冗長化構成を取っていても、障害時、いざバックアップ切替えや障害機器切離しによる縮退運転を行ってみると、システム稼働の継続ができない事例が後を絶たない。

そこで、バックアップ切替えの失敗事例を報道事例と教訓集から調べ、それらを分類、分析し、問題と対策としてまとめた。

本対策の特徴は、バックアップ切替え失敗事例を「問題」として分類、体系立てて整理している点である。過去にまとめられたもの（文献 4.2－1）に対し、部会で議論された障害事例情報等を含めて再整理したものである。

（失敗事例の問題分類）

この資料では、失敗事例を 11 パターンの問題に整理し、発生個所に合わせてマッピングした（図 4.2－1）。

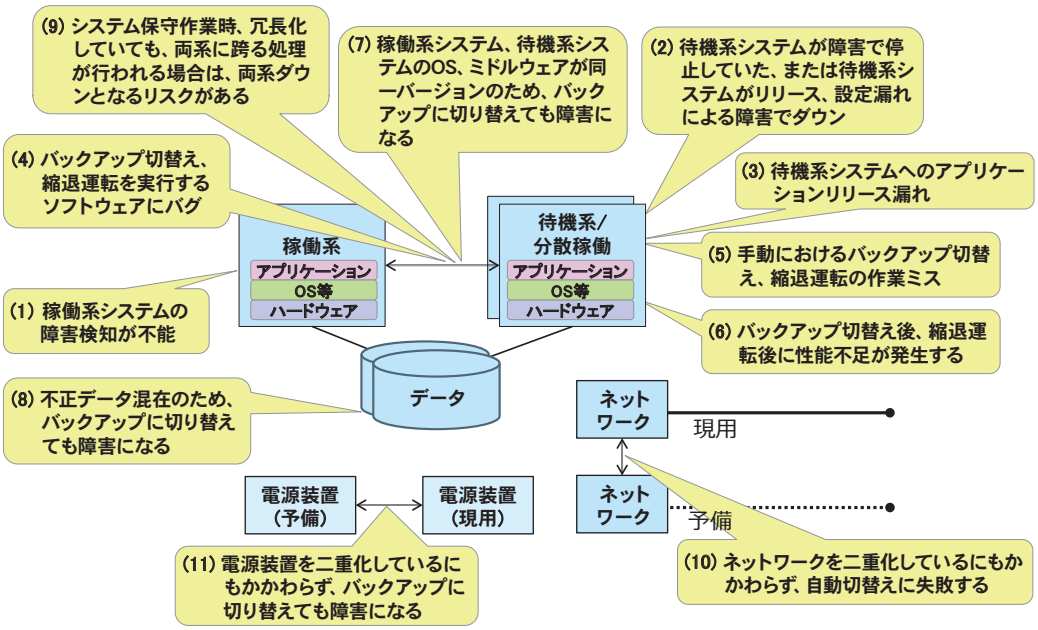


図 4.2－1　問題の種類と発生個所

※注　吹き出しの中の括弧内数字は、問題番号を示す。

（原因の分類）

この 11 パターンの問題から原因を以下の 5 つに整理した。

4.2.1. 切替え失敗
4.2.2. 性能不足
4.2.3. 切替え無効
4.2.4. ネットワークの切替え失敗
4.2.5. 設備の切替え失敗

「切替え失敗」、は、バックアップ切替え機能を管理できていないことにより起こる。

「性能不足」は、要件設計時の考慮不足、あるいは急激な情報量急増に対する対策の遅れにより起こる。

さらに、注意すべき点は、バックアップ切替えが正常に実行されても、処理の継続ができない「切替え無効」が起こることである。

また、「ネットワークの切替え失敗」は、ネットワークに関する切替え失敗や性能不足が起こることである。

「設備の切替え失敗」は、システム機器に重大な被害をもたらすため、決して疎かにできない。

（問題と対策）

このような点を踏まえ、対策を整理し、「問題と対策一覧」としてまとめた（表 4.2－1）。

- 設計時に、チェックリストとして活用することにより、バックアップ切替えの機能要件漏れや対策漏れを減らすことができる。
- 運用保守時に、発生した障害と同じ問題に対応する対策を実施することで、障害の再発を防止することができる。
- 切替え処理に関する対策を網羅的に確認、実施することができる。

表 4.2－1 「問題と対策」一覧

分類	NO	問題	対策
切り替え失敗	1	稼働系システムの障害検知が不能	＜対策1＞通常保守運用において、稼働系、待機系のソフトウェアパラメータの確認、プログラムバージョン管理を徹底する。
	2	待機系システムが障害で停止していた、または待機系システムがリリース、設定漏れでダウンした	＜対策2＞定期的にサービス停止時間帯を設け、障害訓練を行う。
	3	待機系システムへのアプリケーションリリース漏れ	＜対策3＞計画的に待機系システムを本番システムとして使用する。
	4	バックアップ切替え、縮退運転を実行するソフトウェアにバグ	＜対策4＞切替え失敗を想定し、復旧のための手順を明確にする。
	5	手動におけるバックアップ切替え、縮退運転の作業ミス	＜対策5＞障害復旧訓練を行い、実際に使える手順書を作成しておく。 ＜対策6＞切替え、縮退運転の運用要件（訓練、障害時対応）は、設計時に明確にする。
性能不足	6	バックアップ切替え後、縮退運転後に性能不足が発生する	＜対策7＞バックアップ切替え後、縮退運転の場合でのピーク時性能は、日常の監視の中で想定し、必要に応じてシステムを見直す。 ＜対策8＞本番稼働に近いテスト環境を用意し、性能（負荷）確認が行えるようなツールを作成する。 ＜対策9＞切替え、縮退運転の性能要件（ピーク時）は、設計時に明確にする。
切り替え無効	7	稼働系システム、待機系システムのOS、ミドルウェアが同一バージョンのため、バックアップに切り替えても障害になる	＜対策10＞OS，ミドルウェア，アプリケーションのリリース時は、旧バージョンでの切戻し手順を明確にし、障害訓練 を実施する。
	8	不正データ混在のため、バックアップに切り替えても障害になる	＜対策11＞本番稼働に近いテスト環境を用意し、OS、ミドルウェアのバージョンアップ時の動作確認と性能（負荷）確認が行えるようなツールを作成する。
	9	システム保守作業時、冗長化していても、両系に跨る処理が行われる場合は、両系ダウンとなるリスクがある	＜対策12＞冗長化を実施する場合、すべての機器が冗長化されており、単体機器、または密結合の機器のような冗長化になっていない機器、機能がないことを常に点検する。 ＜対策13＞切替え失敗のリスクを考慮し、失敗の影響を局所化する対策を立てる。 ＜対策14＞ 計画的に待機系システムを本番システムとして使用する。
ネットワーク	10	ネットワークを二重化しているにもかかわらず、自動切替えに失敗する	切替え失敗、性能不足の対策が有効 ＜対策1＞～＜対策8＞ 〈対策15＞2重化回線の同時切断を避ける敷設を行う。
設備	11	電源装置を二重化しているにもかかわらず、バックアップに切り替えても障害になる	＜対策16＞電源等の設備については、 定期点検と、予備設備の稼働確認を行う。また障害訓練を定期的に行う。 ＜対策17＞電源等の設備の冗長化の運用要件は、設計時に明確にする。

4.2.1 切替え失敗

稼働系・待機系運用では、稼働系から待機系への「切替え失敗」の障害事例が多い。また、多重化運用でも障害機器の切離しが失敗する事例もある。最近は、切替え専用ソフトウェアの性能・品質向上、仮想化などの新技術などにより、このような障害は防げる傾向にあるが、依然として大きな課題である。

問題

(1) 稼働系システムの障害検知が不能

これは、稼働系システムの障害を検知できず、バックアップ切替えや障害機器の切離しが実行されないままで、障害回復ができない問題である。

事例では、稼働系システムが停止していると判断ができない状態（サーバ間の状態監視不具合、制御信号の不具合など）であった。

また、前処理用サーバが後処理用サーバの障害を検知できずにデータのやり取りを続行したため、折角後処理用サーバの待機系があったにも関わらず、障害となった事例もあった。

他に何らかの理由（急激なデータ量増加による輻輳状態の発生、ある部分に集中したトランザクションのデッドロック多発のためロールバック多発など）で処理が大幅に遅延しながらも稼働状態のままで完全に停止しない事例もある。

※技術の教訓 T1、T2,T9、T20 は、このパターンである。

(2) 待機系システムが障害で停止していた、または待機系システムがリリース、設定漏れによる障害でダウンした

待機系システムに障害が発生していたが切替え時まで気づかない。稼働系システムへの基本ソフトのパッチ適用、パラメータ等の設定変更を行った後、同様に待機系システムにもこの対応をする必要があったにも関わらずこれを怠った。このような状況で、バックアップ切替えを実行したため、待機系システムがダウンしてしまった。

また、待機系システムの DISK ミラーリング定義が誤っており、切替え後に使用できない DISK が生じてしまい、業務が停止してしまったなどの事例もある。

※ガバナンス／マネジメントの教訓 G11、技術の教訓 T7 は，このパターンである。

(3) 待機系システムへのアプリケーションプログラムリリース漏れ

稼働系システムへのアプリケーションプログラムリリースを行なった後、待機系システムに対しても稼働系システムと同じアプリケーションプログラムリリース作業を実施する必要があるが、これを怠った。そのため、稼働系と待機系でアプリケーションプログラムに相違が発生した。これを放置したまま、バックアップ切替えを実行したため、待機系システムで、旧バージョンのアプリケーションプログラムが動き、データベースのデータ不具合が発生した。そのため、待機系システムを止めて、データ修復を行う甚大な障害に至った。

(4) バックアップ切替え、縮退運転を実行するソフトウェアにバグ

バックアップ切替え、障害機器の切離しの縮退運転を自動的に実行するソフトウェア自体にバグがあった。そのため、障害が発生しても、バックアップ切替え、障害機器の切離しを実行できず、障害復旧ができなかった。

(5) 手動におけるバックアップ切替え、縮退運転の作業ミス

上記、問題 (1) から問題 (4) の自動切替えが失敗した場合には、手動で切替えを実行する。また、デュプレックス構成のコールドスタンバイやウォームスタンバイ、クラスタ構成の一部でも手動切替えを行う場合がある。そのような状態の場合、マニュアルの不備や障害訓練不足、オペレーションミスなどにより、手動での切替えに失敗することがあった。

原因

稼働系システムは変化するため、それに合わせて待機系システムも同期を取る必要があるが、日常の運用で検証していく仕組み（切替え実施、同期チェック、手動切替え手順書の更新など）を作らないと待機系システムは取り残されていく。多重化構成についても、同様に検証する仕組みが必要である。

根本的な原因は、「待機系システムも本番運用の重要な機能である」との観点が不足していることである。日常の運用で、待機系システムを検証していく仕組みが本番運用として十分に行われていないために起きている。

対策

＜対策 1＞ 通常保守運用において、稼働系、待機系のソフトウェアパラメータの確認、プログラムバージョン管理を徹底する。

通常運用のプロセスの中で、冗長構成を定義したソフトウェアパラメータに矛盾がないことを確認する。チェックプログラムを作成し、日常バッチ処理で、稼働系、待機系の構成定義、各サーバの構成定義のチェックを行う。

また、プログラムライブラリは、DISK ミラーリングで同期取りをする。それができない場合は、各サーバ上のバージョン管理をチェックする。（文献 4.2－2）

さらに、システムが正しく動作するかどうか、実機のテストを行う。切替えが成功したことを確認するだけでなく、待機系でも業務が正常に稼働することまで確認する。そのため、確認事項／チェック事項（サービスはすべて稼働したか、すべての接続端末は稼働するか、等の動作確認）を明確にする。

＜対策 2 ＞定期的にサービス停止時間帯を設け、障害訓練を行う。

バックアップ切替えの運用を理解するために、待機系への切替えの障害訓練を行う。

さらに、ネットワーク機器の切替え訓練も行うようにする。

なお、障害訓練で、待機系に切り替えたために本番処理が稼働してしまい、システム障害を引き起こす事例が過去にあった。この事例では、業務処理を稼働系、待機系でそれぞれ実行してしまい、二重処理になった。本番環境で実施するので、事前準備（本番環境のデータ保存、手順書の作成、訓練終了後の戻し手順、確認手順等）をしっかり行うことが必要である。

＜対策 3 ＞計画的に待機系システムを本番システムとして使用する。

定期的に本番稼働するシステムを交互に使うことにより、「稼働系と待機系システムの同期が取れている」確証が得られる。さらに、障害で切り替わっても日常運用の一環であり、運用部門に負担とならない。また、急な保守作業が入って、片系を止めるような運用にも対応することが容易となる。

さらに、ネットワーク機器も交互に切り替えて使用する。

＜対策 4 ＞切替え失敗を想定し、復旧のための手順を明確にする。

待機系への切替えができなかったときを考え、手動で障害から復旧する場合（復旧は、バックアップ切替え方法も含めた処理継続の確立を言う）も考慮し、様々なシナリオ（目標所要時間を含む）を想定した手順書を作成しておく。また、障害復旧テストを行い、各シナリオについての所要時間を計測し、手順書の確認を行う。

＜対策 5 ＞障害復旧訓練を行い、実際に使える手順書を作成しておく。

障害復旧訓練を行い、シナリオに定めた時間内に復旧できるかどうかを確認する。また、実際の人の動きや判断基準等を考慮して、手順書に反映する。

＜対策 6 ＞切替え、縮退運転の運用要件（訓練、障害時対応）は、設計時に明確にする。

バックアップ切替えの成功のためには、その対策を設計時に明確にしておく。

運用に関する「原理原則」を活用し、冗長化対策として設計時に活用する。（文献 4.2－3）

- システム化の方針として、バックアップ切替え、縮退運転の稼働運用を定義する。
- システム要件として、バックアップ切替え、縮退運転の稼働運用を定義する。そして、受入テストとして記述する。また、障害訓練も日常運用として定義する。
- バックアップ切替え時間を定義するとともに、障害時の復旧時間、災害時の復旧時間を定義する。

4.2.2 性能不足

稼働系システムから待機系システムへの切替え時、障害機器の切離し（縮退運転）時に起きた性能不足の問題と対策を述べる。

問題

(6) バックアップ切替え後、縮退運転後に性能不足が発生する

バックアップ切替え時は、切替えが完了するまでに新規のトランザクションを滞留させることになる。そのため、切替え後、滞留したトランザクションが待機系システムに一気に入るため、高負荷がかかりシステムが処理できなくなる。

また、多重化運転の場合、縮退運転時には通常より性能が落ちた状態で処理することになる。縮退運転時のトランザクション処理性能を十分に確認していなかったため、縮退運転に切り替えた後、たまたまピークのトランザクションが発生した結果、処理しきれずにシステムが停止する。この縮退運転切替え時にトランザクションを滞留させる事態が発生すると、さらにピーク時処理性能を高く考慮する必要がある。

また、縮退運転は、障害時以外にも保守作業時に実施することもある。そのときに性能不足が発生した事例もある。

原因

多重化運用では、縮退運転直後の「性能不足」事例が多く起きている。複数サーバで負荷分散を兼ねた多重系システムで、年々処理件数が増え、処理負荷の増加が進んでいるのを見逃したため、障害時に縮退運転を行ったときに性能不足が起きてしまう。

また、稼働系・待機系運用でも切替え時間が長くなってしまった場合は、その間処理の滞留が生ずるため、切替え直後に性能不足が起こる。

根本的な原因は、日常の運用で、縮退時運転、バックアップ切替え時の負荷を想定した対策を怠ったためである。

対策

<対策７> バックアップ切替え後、縮退運転の場合でのピーク時性能は、日常の監視の中で想定し、必要に応じてシステムを見直す。

バックアップに切り替える場合、切替え時間中は仕掛中のトランザクションの復旧作業を優先するため、新規のトランザクションはキューに溜められる。このため、バックアップ切替え後に、キューに滞留したトランザクションを一気に処理するため高負荷がかかることが多い。性能監視を日常の運用で監視することにより、切替え後のピーク稼働時での必要性能要件を明確にし、システムの見直し（性能向上）を実施する。

多重化運用（負荷分散クラスタ構成など）で稼働しているシステムの障害時は、障害機器を切り離

す縮退運転になる。性能監視を日常の運用で監視することにより、縮退運転時での必要性能要件を明確にし、システムの見直し（性能向上）を実施する。

また、保守作業時でも縮退運転が行われることも考慮する。特に緊急対応を行う保守作業は、システム障害発生後の縮退運転時に行うことになる。よって同様に縮退運転時での必要性能要件を明確にする。

これらの対策を、常に日常運用で監視、分析し、必要に応じて性能要件の見直しを行う。

＜対策 8 ＞本番稼働に近いテスト環境を用意し、性能（負荷）確認が行えるようなツールを作成する。

性能（負荷）確認は、本番稼働に近いテスト環境で、自動的に大量のトランザクションを発生させることができるツールを用意し、性能（負荷）確認が行えるようにする。

＜対策 9 ＞切替え、縮退運転の性能要件（ピーク時）は、設計時に明確にする。

切替え時、縮退運転時の性能不足を起こさないためには、その対策を設計時に明確にしておく。

運用に関する「原理原則」を活用し、冗長化対策として設計時に活用する。（文献 3.19 – 3）

- システム要件として、バックアップ切替え、縮退運転の稼働運用を定義する。
- バックアップ切替え時、縮退運転時の性能要件にピーク時の負荷要件を明記する。

4.2.3 切替え無効

バックアップ切替えが正常に機能しても、処理の継続ができない事例がある。そのような事例を、「切替え無効」と名付けた。ここでは、その問題と対策を述べる。

問題

(7) 稼働系システム、待機系システムの OS、ミドルウェアが同一バージョンのため、バックアップに切り替えても障害になる

バージョンアップした OS、ミドルウェアそのものにバグがあり、バックアップ切替えを実行しても、待機系もそのバグで障害となってしまった。さらに、切戻し手順が明確になっていなかったため、復旧に大幅な遅れが出てしまった事例もある。

また、稼働系が、ファームウェアのバグでメモリ不足エラーを起こして待機系に切り替わったが、待機系も同様にメモリ不足エラーを起こしたといった事例もある。

※技術の教訓 T10、T16 は、このパターンである。

(8) 不正データ混在のため、バックアップに切り替えても障害になる

アプリケーションプログラムにバグがあり、不正なデータがデータベースに混在したため、バックアップ切替えを実行しても、待機系もその不正データの処理で障害となってしまった。

(9) システム保守作業時、冗長化していても、両系に跨る処理が行われる場合は、両系ダウンとなるリスクがある

冗長化しているにも関わらず、一方の系のシステムの処理が他方の系のシステムにも影響が及ぶような仕様になっている場合、その処理が両系に跨り両系ともシステム障害となってしまった。

※技術の教訓 T12 は、このパターンである。

原因

ここで提示した問題は、結果として「切替え失敗」の事例であるが、本質は、切替えがうまく行けば問題が無くなるわけではなく、冗長化の前に対策を行うべき問題である。このようなバックアップ切替えが正常に機能しても処理の継続ができない「切替え無効」の事例では、冗長化していても防げないことがある。

一般に、「冗長化は、ハードウェア障害のためであり、ソフトウェア障害のためではない」と言われる。しかし、ソフトウェア障害でも、稼働系から待機系に切り替えたことにより、システムの再起動が行われ障害がなくなった事例もよくあり、稼働系の動きがおかしいと思った場合に待機系に切り替えてみるなどの運用が実際に行われる。

そのような前提で、問題 (7) から (9) のような事例があることを理解しておくことが、障害時にあわてないためにも重要となる。

対策

＜対策 10 ＞ OS、ミドルウェア、アプリケーションのリリース時は、旧バージョンでの切戻し手順を明確にし、障害訓練を実施する。

OS、ミドルウェア、アプリケーションのリリース時は、そのソフトウェアにバグがあった場合、待機系に切り替えても、稼働系と同様に待機系も障害になる。

よって、旧バージョンの手動での切り戻し手順を考え、その手順書を作成しておく。当然、障害訓練をおこない、手順書の確認を行う。

＜対策 11 ＞本番稼働に近いテスト環境を用意し、OS、ミドルウェアのバージョンアップ時の動作確認と性能（負荷）確認が行えるようなツールを作成する。

本番稼働に近いテスト環境で、新バージョンでの動作確認と性能（負荷）確認が行えるツールがあれば、本番稼働が正常に行えるかどうかの検証ができる。それにより、システム障害を未然に防ぐ手立てとなる。

＜対策 12 ＞冗長化を実施する場合、すべての機器が冗長化されており、単体機器、または密結合の機器のような冗長化になっていない機器、機能がないことを常に点検する。

冗長化をしていると思っていたところが、厳密には冗長化と言えない構成が存在する。製品特性をよく理解し、障害発生時に影響の生じない構成になっていることを常に点検する。冗長化された一つ一つの機器を確認し、一方が障害になっても問題がないことを確認する。

※技術の教訓 T12 は、この対策も有効である事例である。

＜対策 13 ＞切替え失敗のリスクを考慮し、失敗の影響を局所化する対策を立てる。

例えば、サーバが複数台あり冗長化構成のグループを複数作ることができるならば、グルーピングすることによって、障害のあったグループだけが停止し、他グループは正常に稼働を続けるようなシステム構成を取ることもできる。

これは、サーバだけでなく、ネットワーク、ストレージ（NAS）でも同様にグルーピングを考慮することによってシステム全体が止まるようなリスクを軽減できる場合がある。一般には、フルメッシュ構成を取る場合が多いが、このような対策も検討に値すると考える。

※技術の教訓 T10 は、この対策をとった事例である。

＜対策 14 ＞計画的に待機系システムを本番システムとして使用する。

定期的に本番稼働するシステムを交互に使う。そのことで、一方を止める（電源断）ことになり、長期間連続運転によって引き起こされる OS，ミドルウェアなどの潜在バグを未然に防止することができる。あわせて、「稼働系と待機系システムの同期が取れている」確証を得ることができる。

また、ネットワーク機器にもこのような対策は、効果があると考える。

※技術の教訓 T16 は、この対策をとった事例である。

4.2.4 ネットワークの切替え失敗

ネットワークでの待機系への切替え時、障害機器の切離し（縮退運転）時の失敗について、問題と対策を述べる。

問題

(10) ネットワークを二重化しているにも関わらず、自動切替えに失敗する

ネットワークの機能向上により、ネットワーク機器のソフトウェアも複雑になってきている。そのため、ネットワークの切替えや、機器切離しの設定ミス、オペレーションミス、切替え後の性能問題など、ホスト／サーバで起きている障害が、ネットワーク機器でも発生している。

- ネットワークで相互監視をおこなっている装置（ルータ等）が相手の障害を検知できずに、自動切替えが実施されずに障害が発生した。
- ネットワーク障害時、ネットワーク機器の設定パラメータが誤っていたため、正しく切替えができなかった。
- ネットワークの切替え時にも端末側の再接続要求が集中したりして輻輳状態が発生し、性能問題が生じた。
- サーバ間のメモリ上のデータ同期をとるため、ネットワークスイッチを介在した構成で、稼働系スイッチが自身の故障を認識できず、スイッチが待機系に切り替わらなかった。
  ※技術の教訓 T23 は、このパターンである。
- 2 重化された回線が、集約された敷設のため、物理的な同時切断を引き起こす場合がある。
  ※技術の教訓 T30 は、このパターンである。

原因

ネットワークの切替え失敗、縮退運用での性能問題の原因は、ホスト／サーバで起きている原因と同じものがある。近年、ネットワーク機器の機能が向上しているため、ホスト／サーバと同様に管理しなければならないが、現場の体制、技術が追い付いていないことも考えられる。

対策

対策についても、ホスト／サーバで立てた対策　＜対策 1 ＞から＜対策 9 ＞までを活用できる。

＜対策 15 ＞ 2 重化回線の同時切断を避ける敷設を行う

せっかくネットワークを 2 重化にしていても、敷設工事の時にケーブルを一緒に束ねておいては、物理的切断が同時に起こる可能性が高くなる。そこで、ケーブルの設置状況、敷設状況の点検を行い、同時切断が起こるような回線の集線個所では、配線を分離する。

4.2.5 設備の切替え失敗

設備の冗長化失敗について、問題と対策を述べる。設備の障害は、結果として甚大な被害になることがあるので、対策は重要である。

問題

(11) 電源装置を二重化しているにも関わらず、バックアップに切り替えても障害になる

無停電電源装置（UPS）や、自家発電機を準備しているにも関わらず、日常の点検漏れで動作確認が行われず、いざ商用電源の供給が止まったとき、それらの装置が予定通り稼働しないために、システム障害となった。また、自家発電機を複数台用意していたが、センサーが冗長化されておらず、なおかつセンサーの障害に気づかなかったため、燃料切れを認識できなかったなどの事例もあった。

原因

運用管理者にとって設備については、やや専門外でもあり、注意が回らない場合がある。しかし、一旦電源などの障害が起きた場合、すべてのシステム機器が故障する可能性を持つため被害は甚大である。さらに、近年クラウドの普及により、データセンターの設立が相次いでおり、設備での不具合が多数のシステム障害を一気に発生させる原因になっている。

対策

＜対策 16＞電源等の設備については、定期点検と、予備設備の稼働確認を行う。また障害訓練を定期的に行う。

電源等の設備は、定期点検を行っている中で、予備設備に切り替えたりして稼働確認を行う。また、設備担当の点検内容をヒアリングし、システムからの以下の観点も考慮する。

- 冗長化されていない設備や周辺機器があるか。
- 故障している設備や周辺機器があるか。
- 無停電電源装置（UPS）などの充電は十分か、自家発電の燃料は充分あるか。　等

また、電源等についても、障害訓練で予備設備の本番稼働での使用を確認する。障害訓練中に障害になる可能性もあるので、実施計画は慎重に立てる。例えば、一旦本体機器の電源を落とした後、切替え後に順次本体機器の電源を入れるなどの工夫が必要である。

＜対策 17＞電源等の設備の冗長化の運用要件は、設計時に明確にする。

電源等の設備においても、切替え時の不具合を起こさないように、多重化になっている個所、多重化されていない個所の洗い出しを明確にし、対策を設計時に立てておく。

4.3 ヒューマンエラーの問題と対策（詳細説明）

ヒューマンエラーは、多層防御で防げ！

システム開発手法にのっとった、ソフトウェアのバグが皆無のシステムを安定稼働させていても、運用時に些細なヒューマンエラーで、重大なシステム障害になる場合がある。これほど残念なことはない。

このヒューマンエラーは、どのくらい発生しているのであろうか。IPA の「情報システムの障害状況一覧」の 2010 年から 2018 年前半に報道されたシステム障害件数 315 件のうち、76 件（全体の 24%）が「作業ミス、設定ミス」などのヒューマンエラーであった。また別の調査[15]では、ヒューマンエラーは、第 1 位の「ハードウェア障害（40%）」に次いで、「ソフトウェア障害（13%）」を抜いた、第 2 位の 29% であった。

このヒューマンエラーは、現象は単純だが、完璧に無くすことは、人間が介在する以上、不可能である。しかし、過去の報道事例から、ヒューマンエラーの失敗事例を分析し、その問題と対策をまとめることによって、何らかの体系立った対策が得られるのではないかと考え、本節を設けた。

対策の特徴は、報道されたヒューマンエラー事例を「問題」として分類、体系立てて整理し、ヒューマンエラーに対して、多層防御で対策を考えることを提案している点である。さらに、部会で議論された障害事例情報等を含めて、ヒューマンエラーの観点での対策の立て方を整理した。

4.3.1 原因分析方法と原因の分類

ヒューマンエラーが原因で起きたシステム障害は、原因分析方法を「人間特性」[16]に基づいて考える必要がある。ここでは、ヒューマンエラーの原因分析方法と実際の原因分類を述べる。

（ヒューマンエラーの原因分類）

一般に報じられるシステム障害で起こるヒューマンエラーの原因は、「作業ミス」や「設定誤り」として、片付けられることが多く、本当の原因をつかむことが困難である。それは、システム障害の原因が、障害が起きた個所がどこ（稼働時、保守作業時など）なのかを問題にする「システム観点」で捉えられるため、人がどのような作業で間違えたのかを問題にする「人間特性の観点」での視点が詳しく分析されず、そのために「作業ミス」や「設定誤り」と簡単に片付けられてしまうからである。

そこで、ヒューマンエラーの原因分類には、従来の「システム観点」からの原因分析だけでなく、「人間特性の観点」からの原因分析が必要である（図 4.3－1）。

---

[15] 日常でも、いざという時にも役立つバックアップとは？ ITmedia，2013.3.28
http://www.itmedia.co.jp/enterprise/articles/1303/28/news003.html

[16] 人間が本来持っている性質

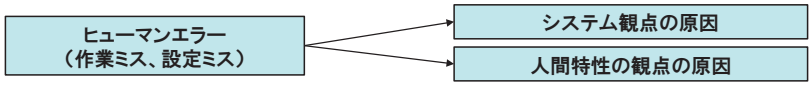


図 4.3－1　ヒューマンエラーの原因分類

（システム観点での原因分類）

システム障害の報道事例から直接原因をシステムの観点で括って見ると、5つのフェーズで障害が発生していることが判明した。さらに、その5つのフェーズを起きた障害原因で分類したところ、12の原因に分けられた（図 4.3－2）。

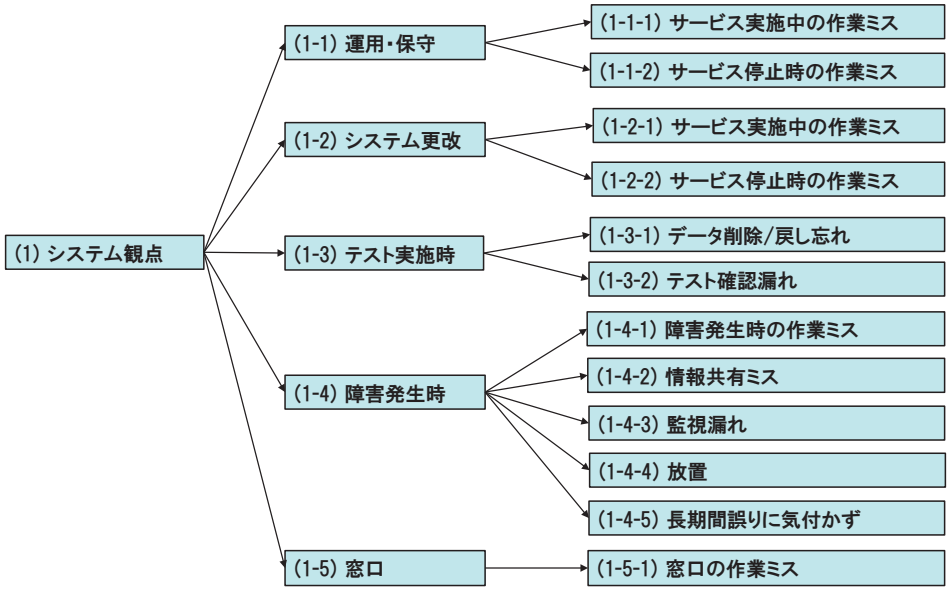


図 4.3－2　原因の種類と分類

「(1－1) 運用・保守」は、運用保守作業時に起きた事例である。日常的な運用とは言え、対応する人の経験状況、作業時間の制限など、いくつかの条件によって引き起こされている。

「(1－2) システム更改」は、システム更改時の、移行作業、運用設定時に起きた事例である。このような更改作業時は、運用手順書を作成し、限られた時間内で、しかも1回で完了させなければならないため、作業ミスが目につく。

さらに注意すべき点は、システム更改時に事前に行う「(1－3) テスト実施時」がシステム障害を引き起こすことがある。せっかくシステム更改を障害なく行うために行った事前テストが仇となってしまっては、せっかくの苦労が水の泡となったことであろう。

また、「(1－4) 障害発生時」は、システム障害発生時にヒューマンエラーを起こした事例である。障害対策の訓練等が事前に十分に行われていたならば、発生を減らせるものである。

「(1－5) 窓口」は、システムを利用する人が起こしたシステム障害事例である。今後システムが社会にますます浸透していく中で、システムの利用者は広がっていき、システム開発時に想定しなかった使い方や操作が多くなっていくことによって起きているエラーである。今後、決して疎かにできないエラーであろう。

さらに、「(1－1) 運用・保守」時に起きたヒューマンエラーと「(1－2) システム更改」時に起きたヒューマンエラーには、「サービス実施中の作業ミス」と「サービス停止時の作業ミス」がある。当然「サービス停止時の作業ミス」よりも「サービス実施中の作業ミス」の方が作業の難易度が高い。

(人間特性の観点での原因分類)

報道事例では、障害原因を作業ミス、設定ミスの一言で終わっているため、システム障害の現場でも人間特性からのヒューマンエラーの原因は掴めていないのではないだろうか。

人間特性についての分類は、医療におけるヒューマンエラー[17]などの先行研究があるが、これらの文献を参考にして、システム障害の原因分析に活用していきたい。

人間特性は、大きく 3 種類に分類できる（図 4.3－3）。[18]

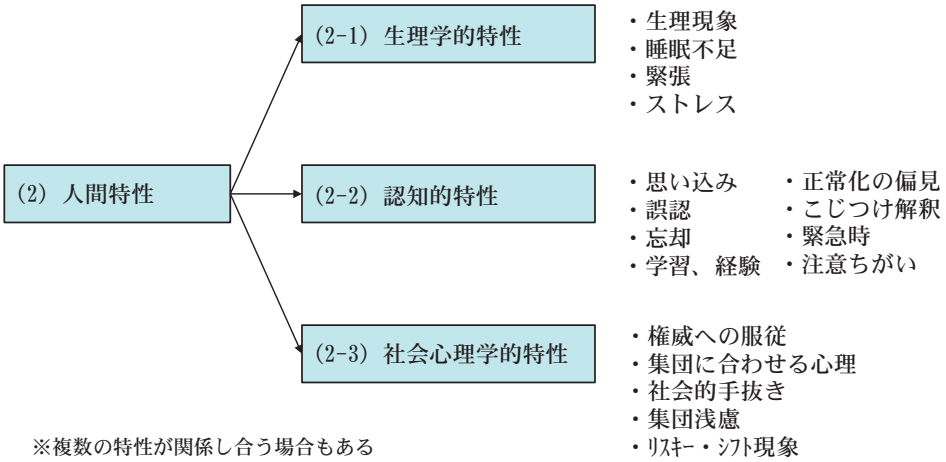


図 4.3－3　3 つの人間特性（文献をもとに編集）

「(2－1) 生理学的特性」は、人間の持っている生理的特性であり、避けることができないものである。この特性を無視し、作業者が、睡眠不足な状態であるにも関わらず、無理な作業スケジュールを立てて、実行時エラーを起こすことなどは、この特性に起因するヒューマンエラーと言えよう。

「(2－2) 認知的特性」は、外部からの情報に対する人間の認知能力が、誤りを起こす判断となる特性である。「思い込み」、「誤認」、「忘却」、「学習・経験」に基づいた認識などは、日常的にもしばしば見られるものである。「正常化の偏見」は、保守的で異常を認めない、明確な証拠がないと動かない傾向を指す。「こじつけ解釈」は、情報の中から不都合な情報を都合の良いようにこじつける行動を指す。「緊急時」は、異常事態では、簡単なことでも思い出せない状態を指す。「注意ちがい」は、あるものに集中すると他の注意はおろそかになることや、関心があるものには注意がいくが、ないとおろそかにな

[17] 河野龍太郎「医療におけるヒューマンエラー第 2 版」医学書院　2014 年
[18] 同上

ることや、注意は同じ水準で持続させることができないことを指す。

「(2－3) 社会心理学的特性」は、複数の人間（集団）になると、その集団に属している人間が表す、ある特性を持った行動を指す。「権威への服従」は、思っていても言えない、権威を持った人に逆らえない状態を指す。「集団に合わせる心理」は、みんなが言うからいいや、などの安易な追従行為を指す。「社会的手抜き」は、誰かがやるだろうと考え、1人当たりの作業量が、単独の場合と比べて集団では低下する現象を指す。「集団浅慮」は、連帯感が強い集団は、集団の外部からの意見を聞かなくなる状態を指す。「リスキー・シフト現象」は、赤信号みんなで渡れば怖くない、など集団の決定が、個人の決定よりも危険な方に向かう状態を指す。

システムを開発、運用するのは、人間集団である。このような、人間特性が原因となるシステム障害があると思われるため、この人間特性を考慮したシステム障害対策を考えることが、ヒューマンエラーを減らす方向につながると考える。

4.3.2 システム観点での原因分類事例

問題（事例）については、「(1) システム観点」での分類にしたがって紹介する。

日常の運用・保守作業において、ヒューマンエラーを起こす確率は高い。発生した事例を見ていくと、運用・保守作業時における作業ミスを防ぐチェック機能が十分でなく、システム開発時におけるようなシステムの不具合を取り除くチェックフェーズ、防御壁が薄いためと考えられる。

(1－1) 運用・保守時のヒューマンエラー

システム運用保守作業時に作業ミスを起こした事例をまとめた。作業ミスの主体者は、「運用担当者」、「運用オペレータ」となる場合が多い。

このシステム運用時の作業ミスは、「サービス実施中の作業ミス」と「サービス停止時の作業ミス」に分類できる。

(1－1－1) サービス実施中の作業ミス

これは、サービス実施中に運用・保守作業を行ったときに、システム障害となった事例である。サービス向上が進んでいる中、情報処理システムも「24時間365日」稼働させることになり、運用保守作業時にサービスを停止することができないため、サービス実施中に、運用・保守作業を行う場合が増えている。

ここでは、「サービス実施中の作業ミス」としてまとめた。

【事例1】証券取引所・取引システム

証券取引所において、日経平均先物やオプションなどのデリバティブ取引のシステムが障害となり、11時5分から同30分までの25分間、取引が中断した。先物取引が止まったことでヘッジ手段が限られ、現物株に売りが出た。影響は日経平均で数十円程度とみられる。

原因は、基準値段（誤発注等による価格急変の防止の観点で導入された即時約定可能値幅）入力後に人手によって実施すべきオプション取引のステータス切替えの作業にミスがあったことである。

【事例 2】消防・司令システム

通話コールが 4 時間にわたって着信してもすぐに切れてしまう状態となった。この時間帯の通報のうち、着信後すぐに電話が切れる事態が発生。担当者が記録されている番号に折り返し電話することで対応した。

通報を受ける装置の設定ミスが原因。過去に固定電話用の回線を一部廃止したにもかかわらず、システムの設定を変えず、廃止した回線に接続確認のためのデータを送り続けたため、バッファオーバフローにより通報がつながりにくくなった。

【事例 3】報道機関・ニュース配信システム

記事編集や配信に使う基幹システムに障害が発生し、加盟新聞社 56 社やテレビ・ラジオ局 106 社への記事配信ができなくなった。

電源設備の定期点検中に、誤ってシステムへの電力供給の不具合が発生、システムがダウンした。

（1－1－2）サービス停止時の作業ミス

これは、サービス停止時に運用・保守作業を行ったときに作業ミスを起こし、サービス稼働に影響を及ぼし、システム障害となった事例である。

これらの作業ミスを見ていくと、以下のような疑問が生ずる事例が多いように思われる。

- 作業前の準備は万全だったのか
- 作業後の動作確認は、万全だったのか

※ガバナンス／マネジメントの教訓 G6 は，このパターンである。

【事例 1】レンタルサーバサービス会社

利用していた約 5,700 件の顧客のメールデータやホームページデータなどを消失。大半のデータについて復旧も不可能となる。

サーバの保守作業に用いたプログラムに誤って「データ消去」のコマンドが混入していたのを気づかず実行したため、不具合が発生。作業手順にも不備があり、バックアップデータも同時に消去してしまった。

【事例 2】銀行・ATM

早朝から全国 73 拠点、429 台の ATM について障害が発生し、取引不能となった。（ATM は全国に約 6,000 台）停止時間は約 6 時間半。

業務終了後の定期保守で ATM のセキュリティ対策（OS のバージョンアップ）を実施したときに一部作業ミスがあり、翌日の業務開始時点に ATM が動作しなかった。

【事例 3】バス会社・運賃システム

バス会社では、消費増税にともなう運賃システム変更にミスがあり、消費税改定前の 3 月 31 日から増税後の運賃を乗客から取っていた。ミスがあったのは路線バス 1 台で、乗客 68 人から 10 円ずつ余分に取っていた。

3月26日の運行終了後にシステム切替えの設定をしたが、1台だけ日付を誤って1日早くセット。31日始発から増税後の運賃を取り、約2時間後に乗客の指摘で発覚した。

【事例4】自治体・住民基本台帳システム

証明書発行システムで障害が発生。発行窓口に設置された約3,200端末のすべてで遅延が発生した。約3万5千件の書類発行が最大で2時間遅れた。

年明けから「基幹系システム統合基盤」が稼働したが、印刷サーバの設定ミス（同時に実行できる印刷命令数の設定ミス）により、処理遅延が発生した。設定ミスの原因は、帳票作成用パッケージソフトウェアの仕様を把握していなかったため。

(1－2) システム更改時のヒューマンエラー

システム更改は、一定期間内での一発作業であり、また多くの人の作業をスケジュール通りに進行させていく工程であるため、作業ミスは起こりやすい状態である。

具体的には、以下の困難さをともなう。

① 作業時間の制限

新しい機能、システムの開始予定時間までには、作業を完了させなければならない。

② 十分な確認時間がとれない

①と同様ではあるが、作業の手戻りや、作業進捗が予定通り進まないことにより、十分確保した確認時間が無くなってしまうことがある。

③ ベンダによる独自のパラメータ/SG仕様等がある

ベンダ独自のパラメータ/SG仕様があるため、作業者は一般に不慣れな状態で作業を行う。新製品の扱いであればなおさらのことであろう。

作業ミスの主体者は、「運用担当者」、「運用オペレータ」の他に、データ移行システム開発者なども加わることがある。このシステム更改時の作業ミスも、「サービス実施中の作業ミス」と「サービス停止時の作業ミス」に分類できる。

(1－2－1) サービス実施中の作業ミス

システム更改時の作業は、難易度が高い上に、サービス実施中の作業であれば、さらに難易度が高い運用作業になる。

※ガバナンス／マネジメントの教訓G10は，このパターンである。

【事例1】通信会社・携帯電話サービス

一部ユーザの各種設定情報が、他ユーザにより閲覧・変更可能となる。メールアドレス設定、モードパスワードなどが変更された人数は約780人、迷惑メール設定などが変更された人数は約4,600人。各種設定サイトを停止した。

ソフトウェア更改時の適用先サーバの誤りにより、本来は許されない参照・更新が可能となった。このシステムは、利用者を3群に分けて同じ機能を持つサーバA群とB群に分散して収容していたが、

今回ソフトウェアの更改にあたって、B 群サーバに誤って A 群サーバ用のソフトウェアを適用してしまったため、B 群側から A 群側のユーザの情報を参照・更新可能となってしまった。

【事例 2】通信会社・携帯電話サービス

携帯電話（スマートフォン）の E メールリアルタイム受信設定を行っていた端末について E メール送受信サービスが利用できなくなった。

新機能の追加のために実施したバージョンアップ作業において、現行設備のユーザ情報の新設備へのコピー作業中のコマンドに誤りがあり、現行と新設備の間でユーザ情報に不一致が生じた。

【事例 3】ケーブルテレビ会社・IP 電話サービス

一部地域の利用者が 110 番通報すると、4 回に 1 回の割合で地域外の警察署につながっていた。

IP 電話サービスのサーバ内の設定を変更した際、誤って別データを登録したことにより障害となった。

（1－2－2）サービス停止時の作業ミス

これは、サービス停止時にシステム更改作業を行ったときに作業ミスを起こし、サービス稼働に影響を及ぼし、システム障害となった事例である。

※技術の教訓 T21 は，このパターンである。

【事例 1】銀行・勘定系システム

オンラインシステムに障害発生。全国の本支店の店頭及び ATM での入出金、為替、照会などの取引 2,884 件が不能に。インターネットバンキング利用の 988 名が取引出来ず。他行やコンビニ店などでの ATM における 2,805 件の取引が出来ず。

正月 1 日から 3 日にかけて実施したシステムの更新作業で、更新すべきファイルを取違えたため。

【事例 2】銀行・勘定系システム

ATM システムの障害により、ATM356 台で通帳やキャッシュカードが ATM から戻らないトラブルなどが発生し取引出来なくなった。全国のコンビニの ATM でもトラブルが発生したため入金と振込みを停止した。

サーバのシステム改修中に、関係のない ATM のデータを初期化したのが原因。

【事例 3】鉄道会社・座席予約システム

鉄道会社のインターネット列車予約サービスでシステム障害が発生し、携帯電話・パソコンからの座席予約・変更操作が利用出来なかった。

鉄道会社は、座席予約システムの管理をベンダに委託しており、ベンダの「旅客販売総合システム」に接続している。この「旅客販売総合システム」サーバがダウンしたためサービスが停止した。サーバのダウンの原因はプログラムの設定ミスであった。

（1－3）テスト実施時のヒューマンエラー

テスト時にヒューマンエラーを起こしてしまい、本番環境に影響を及ぼしたために、システム障害となった事例が存在する。これらの事例を、「（1－3－1）データ削除 / 戻し忘れ」と「（1－3－2）テスト確認漏れ」に分けて整理した。

（1－3－1）データ削除 / 戻し忘れ

本番環境でテスト実行後、テストデータを削除し忘れたり、元の本番環境に戻し忘れたりした事例である。

【事例 1】銀行・為替システム

残高証明書の発行手数料を 2 重に引き落としてしまった。

原因は、月初めに実施したテストにおいて用いたテストデータの削除を忘れ、そのままバッチ処理を実行したため、二重引き落としが発生してしまった。

【事例 2】鉄道会社・改札機システム

消費増税に対するシステム更改前のある日、始発から午前 8 時 40 分ごろまで、鉄道会社 A 駅の自動改札機を IC カードで出た利用者から過剰に運賃を収受した。

改札機の周辺機器を前日夜に更新した際、メーカが誤って消費税が 8% に増えた場合の運賃を登録していた。改札機の周辺機器の更新を受託したメーカが、社内で消費増税に対応した動作をするか確認するテストをした後、正しいデータを登録し直さないまま、現場に機器を設置してしまった。

【事例 3】金融機関・為替システム

外貨の為替レートとして誤った値を配信してしまった。別の日にも誤配信が発生した。

テスト用に使った仮のデータを誤って本番に反映してしまったために起きた。

（1－3－2）テスト確認漏れ

テストを本番環境で確認できなかったため、いざ本番で使うときに稼働しなかった事例である。このような本番環境で確認できない事例は、本番で障害を起こしやすい。

※技術の教訓 T12 は，このパターンである。

【事例 1】自治体・緊急速報メール

システム導入時から約 1 年半にわたって、緊急速報メールを送れない状態だった。24 日の市総合防災訓練までメールを使用する機会がなかったため、発覚しなかった。

職員が市内の携帯電話やスマートフォンにメールを送信しようとシステムを操作したが、携帯電話 3 社との通信が始まっても、メールを送れないままシステムが途中で終了した。システムを納入したベンダの担当者の初期設定にミスがあったほか、自治体がテスト送信をせず、気づかなかったことが原因。

（1－4）障害発生時のヒューマンエラー

システム障害発生時に、ヒューマンエラーを起こした事例を整理した。原因を、「（1－4－1）障害発生時の作業ミス」、「（1－4－2）情報共有ミス」、「（1－4－3）監視漏れ」、「（1－4－4）放置」「（1－4－5）長期間誤りに気づかず」に分類した。

（1－4－1）障害発生時の作業ミス

障害発生時に、作業ミスが起きた事例である。
※ガバナンス／マネジメントの教訓 G11、技術の教訓 T15 は，このパターンである。

【事例 1】銀行・勘定系システム

東日本地域の約 1,000 台の ATM で、取引出来なかった。
前日夜に発生した地震による停電から、復旧作業を行っていたが、そのときに人為的ミスが生じた。

【事例 2】通信事業者・通信システム

新機能対応端末でパケット通信障害が発生、Web 閲覧やメール送受信が全国でできなくなった。
信号制御装置の呼処理をする部分で、呼処理のログを現用系システムから予備系システムにリアルタイムでコピーする機能に遅延が生じた。このとき、通信異常が生じたという誤ったアラームが出た。運用者は、稼働系と待機系を収用する装置全体の復旧措置を実施した。そのため、LTE 端末のセッションがすべて開放され、LTE 端末から一斉に再接続要求が発生し、過度にアクセスが集中し、新規接続ができない状況となった。
本来であれば、このアラームに対する対処としては、稼働系システムから待機系システムに切り替えれば済むはずであったが、対処法が復旧手順書に記述されていなかったため、誤った手順で復旧作業を行ってしまった。

（1－4－2）情報共有ミス

障害発生時に、情報共有ができていなかった事例である。
※ガバナンス／マネジメントの教訓 G4 は，このパターンである。

【事例 1】鉄道会社 3 社・予約券売システム

予約券売システムが計約 4 時間半にわたって停止した。切符の受け取りができなかったり、エラーとなって販売ができなかったり、混乱が続いた。
原因は、システムを管理する関係会社間で、保守を実施する情報が共有されていなかったことである。利用者へ通告がないままサービスが停止した。

【事例 2】自治体・納税サービス

ある自治体は、クレジットカードで納税ができるサービスを新規リリースした。市民に「スマートフォンからでも納税サービスが利用できる。」とアナウンスしていたが、実はその機能がなかった。
自治体の担当者が、ベンダの提案資料を誤解し、別契約が必要だったスマートフォンからの納税サー

ビスがそのまま利用できると勘違いした。クレジットカードの利用については事前にテストしていたものの、スマートフォンからの納税サービスについては運用テストを行わなかったため、市民から通報があるまで気がつかなかった。

(1－4－3) 監視漏れ

障害が発生しているにも関わらず、障害と認識できなかった事例である。

【事例 1】警察本部・遺失物管理

県内遺失物が警視庁のシステムに登録できず、また警視庁のシステムから他県からの拾得物情報を受け取れない事態が発生した。その結果、ある期間の遺失物が持ち主に返還出来ないなどの影響が出た。

障害発生後、サーバのメモリ増強、プログラム修正に加えて、警視庁システムとの間の情報転送を自動から手動に切り替えた。

【事例 2】共同利用センター

スパコンを共同利用するために結んだネットワークの共同利用センターにて障害があり、利用者の保存データの破損、消失が起きた。

共同利用センターで、データの書き込みを行う部分に障害が起き、保存データが破損した。中にはコピーも破損し、復旧ができなかったものも生じた。障害はその後システムを更新する過程で修復されたとみられ、利用者が指摘するまでトラブルに気がつかなかった。

(1－4－4) 放置

放置の事例は、問題があると分かっていたにも関わらず、そのままにした事例である。

【事例 1】自治体・土砂災害防災情報システム

対応している Web ブラウザが IE のみで、さらに IE7 以降のバージョンでは、互換表示機能を使わないと表示ができなかった。

自治体は不具合を解消する必要性を認識していたものの、検討のスピードが遅かった。（IE と互換表示機能の利用を呼びかけるに留まっていた）次年度予算にシステム改修費を計上したものの、契約方法の検討に手間取ったため、次年度になっても着手ができなかった。

(1－4－5) 長期間誤りに気づかず

長期間誤りに気づかず、後になってシステムの誤りに気づく頃には被害が大きくなっている事態がたびたび発生している。原因は、作業者（窓口担当者、エンドユーザ）の誤りとされたり、ソフトウェアのバグとされたり、そのようなシステムの不具合に長期間気づかなかった事例を取り上げた。

【事例 1】損害保険会社・保険金支払いシステム

システム上の不備で長期間にわたり、契約者から保険料を取り過ぎていたと発表した。

担当者が保険金支払いシステムに事故の種類を入力し忘れ、翌年以降、多く保険料を支払わなくては

ならない等級にしてしまった。車の盗難や火災、落書きなどで保険金の支払いを受けた契約者が該当。

【事例 2】通販会社・キャッシングサービス

キャッシングサービスにおいて、過去 10 年分の利息を誤って請求していた。

誤請求が発生したのは、キャッシングをして、その一部を約定日前に返済したケース。このケースでは、利息を日割りで計算するが、システムには日割りの計算機能がなく、手作業で計算していた。手作業での事務オペレーションにミスがあり、過剰請求につながった。

顧客対応や事務処理は、現場の判断で個別に対応していたので、システム部門や経営陣は、業務実態を把握していなかったとみられる。

(1－5) 窓口のヒューマンエラー

システム利用者（エンドユーザ）が起こしたシステム障害を「窓口のヒューマンエラー」としてまとめた。

(1－5－1) 窓口の作業ミス

【事例 1】航空会社・予約システム

予約システムの誤設定により、国内線の座席予約情報が消失。約 10 万 6,000 席の座席指定情報が取り消され（航空券の予約は無効になっていない）、各顧客に座席予約のやり直しを依頼。

原因は、営業担当者が時刻表情報を予約システムへ更新する際に、誤って座席指定の予約情報を消去したことである。営業担当者 2 人による二重チェックを行ったが防げなかった。

【事例 2】自治体・住民票システム

住民から求められていないのに、誤って個人番号（マイナンバー）を記載した住民票を交付した障害が、5 自治体で発生した。

原因は、窓口の職員の誤り。住民票コードを記載した住民票の申請に対して、個人番号（マイナンバー）を記載した住民票を交付したケースが多い。

4.3.3 ヒューマンエラー対策

ヒューマンエラーの対策は、「(1) プロセスの観点の対策」、「(2) 関係性の観点の対策」、「(3) 人間特性の観点の対策」の 3 つの観点を組み合わせた対策として考える（図 4.3－4）。

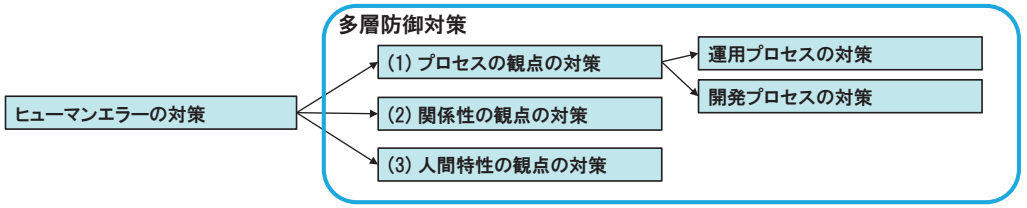


図 4.3－4　ヒューマンエラー対策（多層防御対策）

ヒューマンエラー対策は、「システム観点」からの原因と「人間特性の観点」からの原因との組み合わ

せで、様々なパターンが生ずるため、「このシステム観点で起きたこの事例グループには、これが対策である」といった限定した対策の整理が難しい。

そこで、ヒューマンエラー対策は、対策の立て方、考え方についての手順を提示する。実際の対策は、ここで示した手順で考えることにより、漏れのない対策を多層防御的に立てることができる。以下に、具体的に説明する。

(1) プロセスの観点の対策

プロセスの観点の対策は、「(1－1) 運用プロセスの対策」と「(1－2) 開発プロセスの対策」とに分けて立てる。

運用時の些細な障害であれば、運用プロセスの対策で考えれば良いが、システム改修時の障害やシステム開発時にまでさかのぼって対策を考えなければならない障害の場合は、開発プロセスの対策も検討する必要がある。

(1－1) 運用プロセスの観点の対策

通常の、運用プロセスは、「準備」、「作業」、「確認」の3段階のプロセスが存在し[19]、スイスチーズモデルで表すと（図 4.3－5）のように、3つの防御壁ができることになる。システム障害を防ぐには、様々な防御壁を立てて、その壁の穴を塞ぐことが有効である。

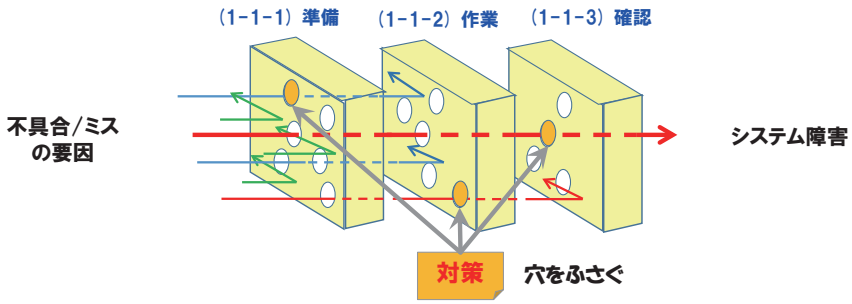


図 4.3－5　運用プロセス

「(1－1－1) 準備」プロセスは、運用・保守作業を行う場合、いきなり作業をするのではなく、その準備作業が存在する。準備作業は、作業計画の作成（目的、実施時期、対象システム、HW、SW、アプリケーション、・・・等）、作業の変更内容、手順書作成、関係者のレビュー、事前テスト、確認などがある。これらの準備が十分行われれば、次の「作業」に進む。

「(1－1－2) 作業」プロセスは、作業計画、作業手順書にしたがって、実際に本番環境で作業を行う。ここでのポイントは、作業中に想定外の事態が生じた場合の対策をどのように考えておくかである。

最後に、作業終了後の「(1－1－3) 確認」プロセスである。この確認が漏れ無く、誤り無く行われていることが確認できれば、本番稼働時の障害を防ぐことができる。

[19] 共通フレーム 2013 では、保守プロセスとしてさらに細かく分類しているがここでは簡潔に 3 段階とした

（1－2）開発プロセスの観点の対策

ここでは簡潔に、開発プロセス[20]を、「設計」、「製造」、「テスト」の 3 段階のプロセスとした。運用プロセスと同様に 、スイスチーズモデルで表すと（図 4.3－6）のように、3 つの防御壁ができることになる。運用プロセスと考えは同じなので、説明は、省略する。

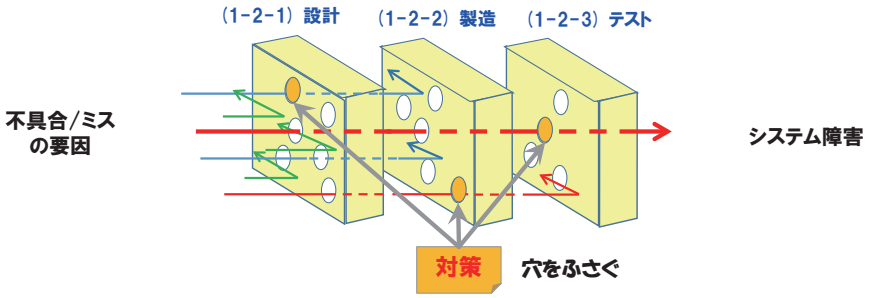


図 4.3－6　開発プロセス

(2) 関係性の観点の対策

ヒューマンエラーの原因と対策は、ヒューマンエラーの起こした本人とその周りの要素を「m-SHEL」モデルとして考えることができる（図 4.3－7）。[21]

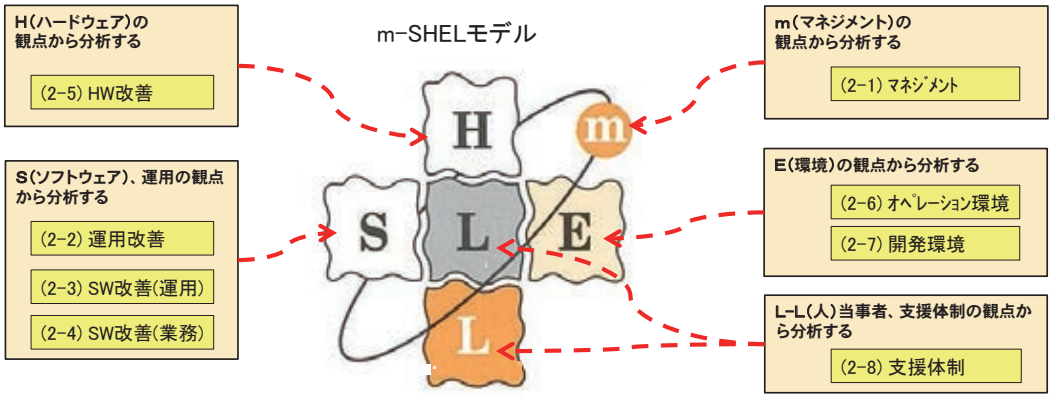


図 4.3－7　m-SHEL モデルと IT 関係性

ここでは、m-SHEL モデルの詳細は省略するが、m-SHEL モデルを活用して、IT における対策の観点を抽出すると、「マネジメント」、「運用改善」、「SW 改善（運用）」、「SW 改善（業務）」、「HW 改善」、「オペレーション環境」、「開発環境」、「支援体制」に分類される。

「(2－1) マネジメント」は、m（マネジメント）の観点から対策を検討する。

[20] 運用プロセスと同様に、正確なプロセスは、共通フレーム 2013 を参照
[21] 河野龍太郎「医療におけるヒューマンエラー第 2 版」医学書院　2014 年

「(2－2) 運用改善」、「(2－3) SW 改善（運用）」、「(2－4) SW 改善（業務）」の 3 つは、S（ソフトウェア）、運用の観点から対策を検討する。m-SHEL モデルでの観点をシステムの観点から 3 分割することにより、運用面での対策、運用ソフトウェアの面での対策、業務ソフトウェアの面での対策、それぞれに対策の漏れが生じないようにした。

「(2－5) HW 改善」は、H（ハードウェア）の観点から対策を検討する。

「(2－6) オペレーション環境」、「(2－7) 開発環境」は、E（環境）の観点から対策を検討する。「(2－6) オペレーション環境」は、本番環境、運用プロセスでの対策であり、「(2－7) 開発環境」は、テスト環境、開発プロセスでの対策になる。

「(2－8) 支援体制」は、L-L（人）当事者、支援体制の観点から対策を検討する。

(3) 人間特性の観点の対策

ヒューマンエラー対策では、戦略的エラー対策の考え方として、エラー対策を 4 ステップに段階的に考えていく方法がある。[22]

◆ステップ1（機会最小）：危険をともなう作業遭遇数の低減

ここでは、危険がともなう作業を減らすことを考える。具体的には、「やめる（なくす）ことはできないか」を考える。しかし、多くの場合、行わないだけでは、本質的な解決にならないであろう。

◆ステップ2（最小確率）：各作業におけるエラー確率の低減

レヴィンの行動モデルでは、エラー発生確率＝ f（人的要因、環境要因）で表される。

環境要因での具体策では、「できないようにする」、「わかりやすくする」、「やりやすくする」があり、人的要因の具体策では、「知覚能力を持たせる」、「認知・予測させる」、「安全を優先させる」、「できる能力を持たせる」がある。

◆ステップ3（多重検出）：多重のエラー検出策

一般に多層防御（多重防御）などと呼ばれ、エラー検出を多数の手段、段階で行う対策を立てることである。具体的には、「自分で気づかせる」、「検出する環境を作る」がある。

システム障害は、日常の運用中に起こるヒューマンエラーであることを考えれば、作業の様々な個所で、エラーを検出することが重要になる。

◆ステップ4（被害局限）：被害を最小とするための備え

これは、万が一、システム障害になった場合の備えを考えることである。システム障害の拡大は、大きなリスクになる。逆に、障害が発生しても、すぐ復旧できたり、障害を局所化できたりすれば、大きな問題になることを防げる。

この 4 ステップの対策をシステム障害の発想手順として、「(3－1) 作業軽減」、「(3－2) チェック機能」、「(3－3) 未然防止」、「(3－4) 能力向上」の 4 つにまとめた（図 4.3－8）。

[22] 河野龍太郎「医療におけるヒューマンエラー第 2 版」医学書院、2014 年　P.65

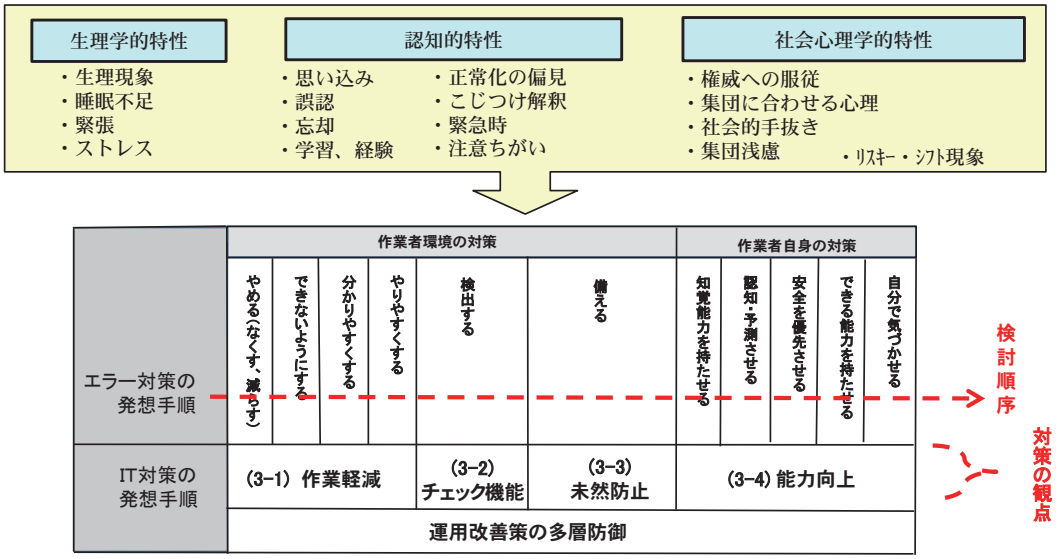


図 4.3－8　人間特性の観点の対策

「(3－1) 作業軽減」は、やめる、できないようにする、分かりやすくする、やりやすくする、をまとめた作業者環境の対策である。

「(3－2) チェック機能」は、障害を検出する対策を考えることである。

「(3－3) 未然防止」は、障害を未然に防止する対策を考えることである。

この「(3－2) チェック機能」と「(3－3) 未然防止」の対策は、システム障害の発生をソフトウェアで防ぐ対策が有効である。よって、システム部門としては、対策を実行にうつしやすい。

「(3－4) 能力向上」は、知覚能力を持たせる、認知・予測させる、安全を優先させる、できる能力を持たせる、自分で気づかせる、などのヒューマンエラーを起こした作業者自身の対策であるが、システム障害を減らすための作業者への過度な期待は対策として限られるため、検討順序は下位になる。

(4) 多層防御対策

今まで述べてきた 3 つの観点から対策をまとめると、多層防御対策になる。

(1) プロセスの観点の対策　　6 段階
(2) 関係性の観点の対策　　8 段階
(3) 人間特性の観点の対策　　4 段階

を掛け合わせることで、192 段階の対策を検討することができ、192 の防御壁を構築することができる（図 4.3－9）。

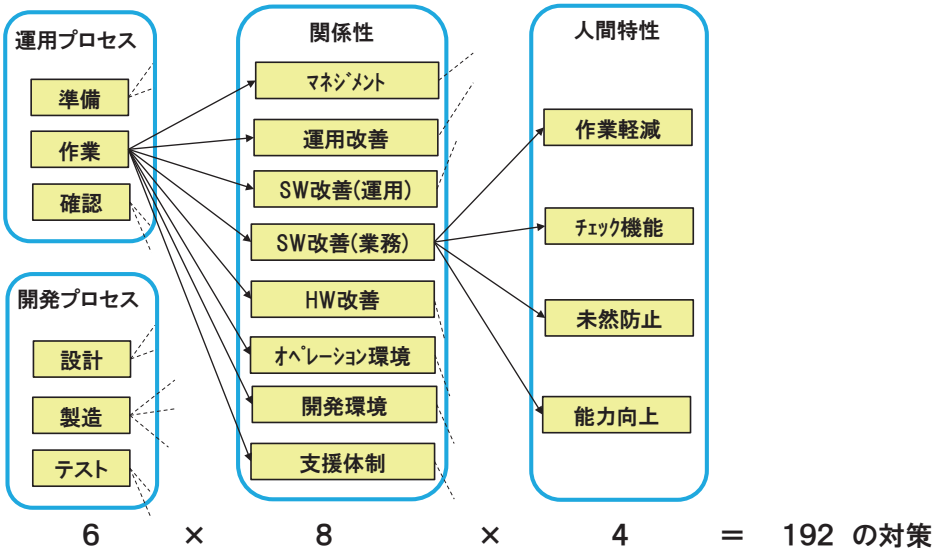


図 4.3－9　多層防御対策

では、具体的に対策の検討方法を述べる。ここでは、運用プロセスで考えてみたい。

対策の検討は、「(1) 運用プロセスの観点」の段階で、ヒューマンエラー当事者の「(2) 関係性の観点」面での、「(3) 人間特性の観点」対策を考えることによって、見つけ出すことができる（図 4.3－10）。

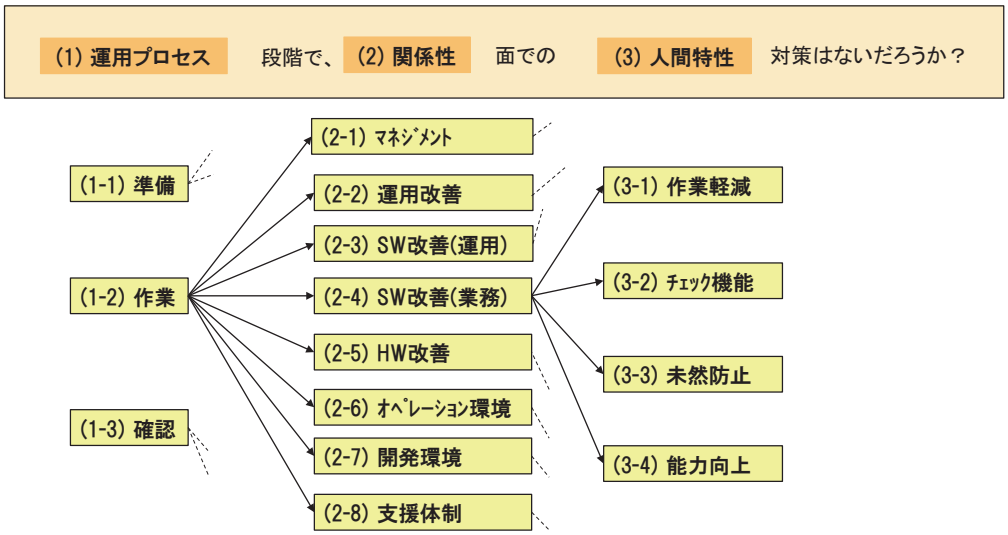


図 4.3－10　対策の検討方法

例えば、以下の問いかけを発することにより、対策を探し、検討することができる。

- 準備段階で、支援体制面での　未然防止策はないだろうか？
- 準備段階で、運用改善面での　作業軽減はできないだろうか？
- 作業段階で、SW 改善（運用）での　チェック機能はないだろうか？
- 作業段階で、オペレーション環境での　能力向上策は？
- 確認段階で、支援体制での　チェック機能はなにがあるか？
- 確認段階で、開発環境での　未然防止策はないだろうか？

こうして考えられる 192 個の対策案をひとつひとつ検討することによって、漏れのない、そして多層防御的な対策になる。なお、考えられた対策は、すべて実行する必要はなく、優先度（効果、コストなど）を検討して実行することになる。

4.3.4 ヒューマンエラー対策事例

このような対策事例は、報道事例からでは導き出すことは難しいので、「教訓集」の事例から考えてみたい。教訓集の中にヒューマンエラーの事例がいくつかあるが、そのうちの 1 件をもとにヒューマンエラーの原因分析と対策を導き出してみたい。

【事例】［教訓 G6］　作業ミスとルール逸脱は、個人の問題でなく、組織の問題！

事例の経緯については、「2.6　作業ミス、ルール逸脱の問題に関する教訓　［教訓 G6］」を参照していただきたい。

【人間特性からの原因】

当時、グループウェア・サービスの移行スケジュールが立て込んでおり、今回のアカウント登録作業もすぐに行わなければならない状況であった。そのため、組織（システム部門）の作業依頼者から作業の緊急性が運用チームに伝えられていた。

運用チームでは、教育、作業手順マニュアルの作成が追いつかず、チーム内のスキルの共有化ができていなかった（作業の属人化）。そのため、作業は、不慣れな作業者一人で行うことになった。

作業中、チーム内ではお互い自分の作業に追われていたため、作業者は、問題が出ても熟練者に聞くことができず、また、時間の制約による焦りがあった。そのため、早く作業を終わらせないと業務に影響が生じ、自分がその原因を作ってはいけないというプレッシャを感じていた作業者は、不具合が生じても、独断で作業を進めてしまった。

この経緯をまとめると、（図 4.3－11）のようになる。教訓に示した図と基本は同じだが、右側にシステム観点の原因として「通常運用　サービス停止時の作業ミス」があり、人間特性観点の原因分析として、それぞれ、システム部門長、運用チームリーダー、作業者の原因を表示した。

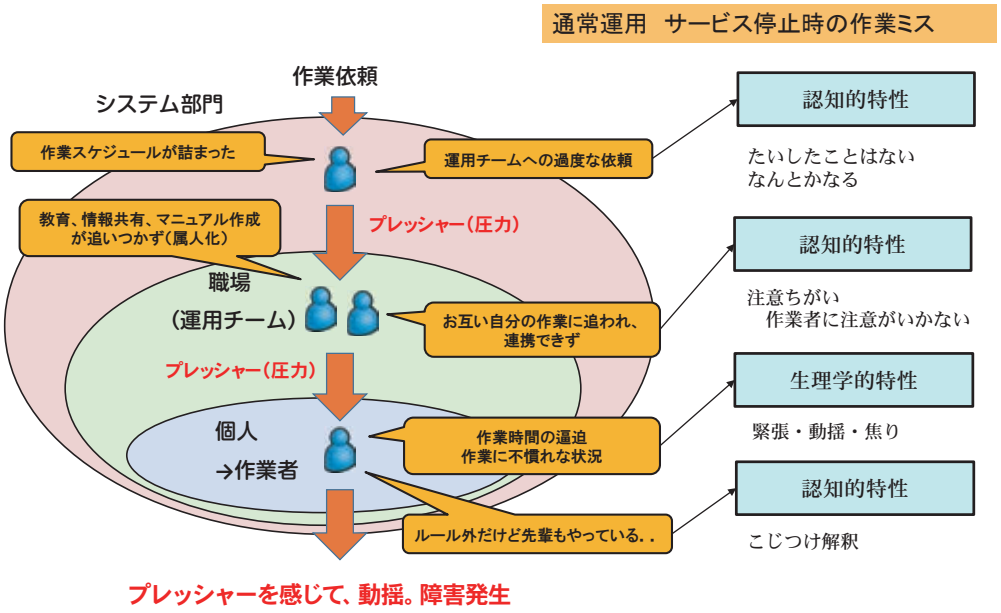


図 4.3－11　人間特性による原因

組織（システム部門）の作業依頼者は、作業スケジュールが詰まっているにも関わらず、運用チームへの過度な期待から、「大したことではない。いつもやりきっている。なんとかなる。」と思い、作業の指示を運用チームに伝えた。これは、人間特性の「認知的特性」の思い込みや誤認に相当する。

運用チームは、お互い自分の作業に追われていたため、不慣れな作業者に対する注意が行かない状態であった。これは、人間特性の「認知的特性」の注意ちがいのひとつで、あるものに集中すると他の注意はおろそかになる状態に相当する。

作業者は、新人のため作業に不慣れな上、短い時間で終わらせなければならないといった状態であった。これは、人間特性の「生理学的特性」の、緊張、動揺、焦りなどの状態である。さらに、作業者は、不具合が生じたとき、本来行ってはいけない作業手順を先輩もやっていたので自分もやっても構わないだろうと考え、独断で作業を進めてしまった。これは、人間特性の「認知的特性」の情報の中から不都合な情報を都合の良いようにこじつけることに相当する。

【ヒューマンエラー対策】

ヒューマンエラー対策は、「(1) 運用プロセス」段階で、「(2) 関係性」面での「(3) 人間特性」対策を考えることになる。

（図 4.3－12）に、事例で登場するシステム部門長、運用チームリーダー、作業者について、それぞれの原因から対策の一部を示した。

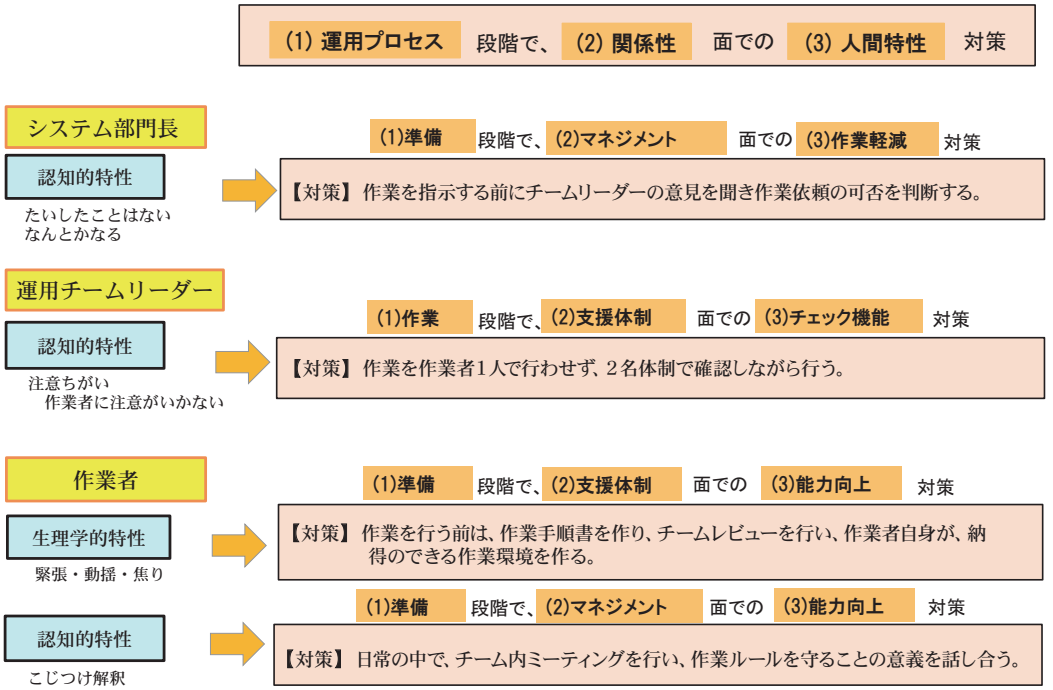


図 4.3－12　ヒューマンエラー対策例

システム部門長は、「たいしたことはない。何とかなる。」といった点を改善する対策として「作業を指示する前にチームリーダーの意見を聞き作業依頼の可否を判断する」ことにより、作業スケジュールの適切化が図られ、ヒューマンエラー防止ができる。

運用チームリーダーは、「作業者に注意がいかない」点について、「2 名体制」をとることにより、誤りのない手順で作業を行わせることができる。

作業者は、「緊張、動揺、焦り」が起きないように、事前の対策を十分検討する。また、「先輩がやっているからいいや」と言った、ルール違反をチームで行わない対策を立てることができる。

対策は、これ以外にも色々考えられるであろう。ここは、一例としてとらえていただきたい。この事例のような手順で、様々なアイデアを関係者同士で出し合い、問題と対策の共有を図ることが、今後のヒューマンエラーを防ぐことになると考える。

4.3.5 問題と対策のまとめ

ここまでの問題から対策への流れを整理し、「問題と対策一覧」としてまとめた（図 4.3－13）。
ヒューマンエラーの対策は、以下の手順で行うことになる。

① 問題（作業ミス・設定ミス）

障害発生時の状況を整理する。図 4.3－13 の問題の分類を見て、該当するグループがあるかどうかを調べる。このグループに当てはまれば、類似事例を見つけやすくなるが、このグループに当てはまらない場合であれば、新たなグループとして整理すると、別の観点での対策が立てやすくなる。

② システム観点の原因分析

システム観点の原因分析は、システムのどのフェーズで、どの場所で起きたかを分類することにより、システム観点でのヒューマンエラーの類似性を見つけることができる。

今後、システム障害事例の収集が進むにつれて、新たなカテゴリが発生することも考えられる。

③ 人間特性観点の原因分析

ヒューマンエラーの原因分析は、人間特性を考慮した、「なぜなぜ分析」を分析ツールとして用いるのが一般的である。

人間特性については、他分野（航空、鉄道、原子力、医療など）での色々な先行研究があり、幅広くそれらの文献を当たってみることも新たな気づきを得る上で、有効であろう。

④ ヒューマンエラー対策

「4.3.3　ヒューマンエラー対策」で述べた、「(1) プロセスの観点」の段階で、ヒューマンエラー当事者の「(2) 関係性の観点」面での、「(3) 人間特性の観点」の対策を考える。

プロセスの観点は、運用プロセスと開発プロセスがあることを述べたが、開発プロセスの観点が必要となる対策は、その障害原因が、システム開発プロセスで対策を講ずる必要があるかどうかを検討すべき場合に、検討する。

開発プロセスの観点では、サービス実施中の運用対策、システム更改時の運用対策、障害発生時の窓口運用対策などが、考えられる。

ヒューマンエラーは、システム障害の単独の原因である場合のほかに、他の障害（ハードウェア、ソフトウェア）が起きたとき、そのときの人間の対応に問題があり、システム障害の影響が拡大する場合がある。

そのような場合でも、ヒューマンエラーの原因を中心に考えてみると、そこから新たな対策が見いだされると考えられる。

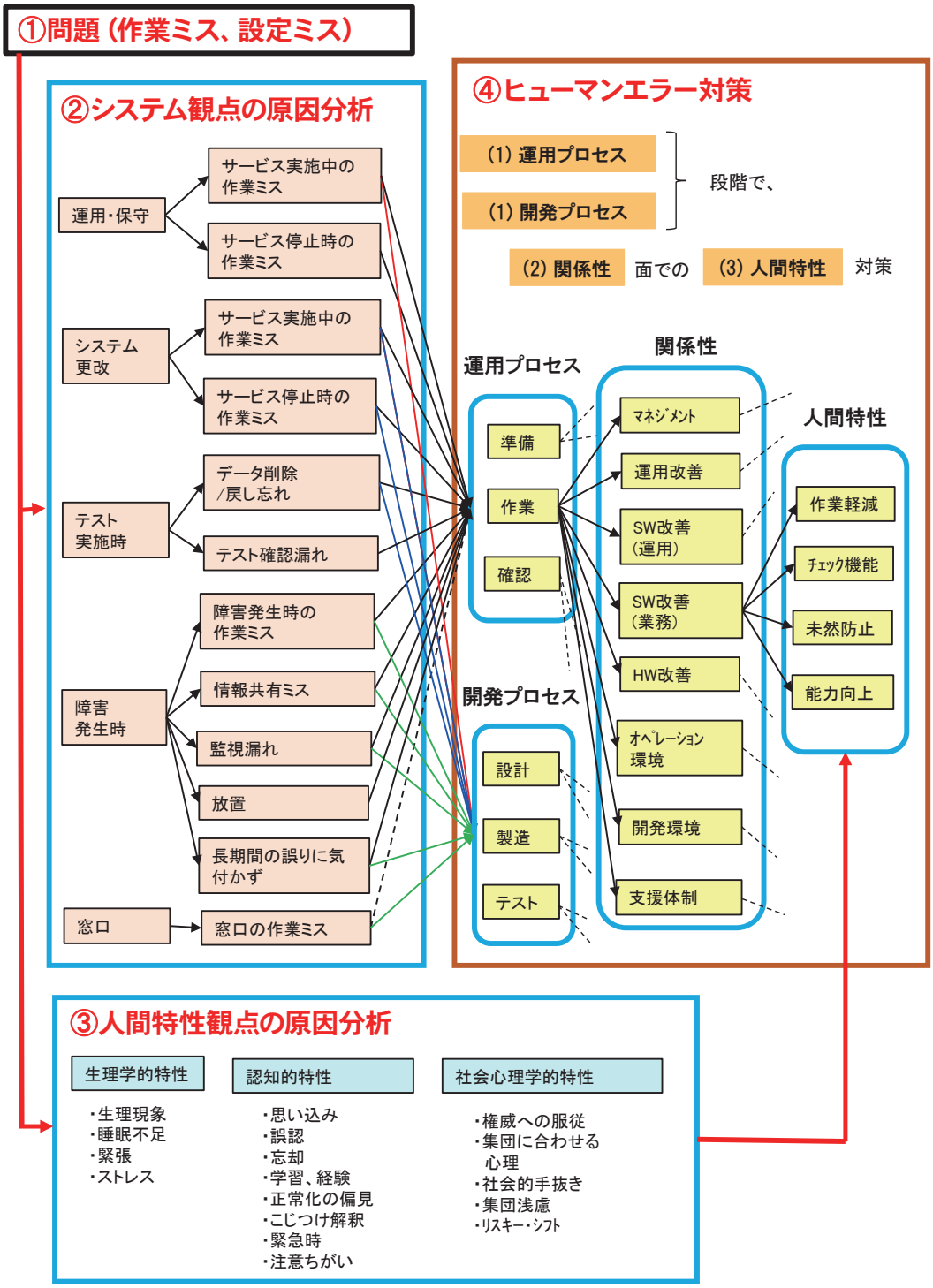


図 4.3－13　ヒューマンエラー　原因と対策一覧

4.4 システムの高負荷／過負荷に関する問題と対策（詳細説明）

システムの性能条件は、開発の最終段階での確認、調整に委ねるのではなく、運用を含むすべての工程で必要な手を打つ。

4.4.1 高負荷／過負荷に関する問題の発生状況

本教訓集にもいくつか事例が収録されているように、システムが何らかの原因によって高負荷／過負荷状態に陥ったこと（直接原因）をきっかけに、システムの停止などの障害を引き起こした事例がしばしば見られる。IPA の調査によると、2010 年から 2018 年前半の間に発生した情報システムの障害総数 315 件（新聞等で報道されたもの）の内、この種の事例が 37 件、約 12%に上っている。

これらを現象面から見ると、システムが高負荷の状態になったために、潜在的に抱えていたソフトウェアのバグが顕在化したもの、ハード障害や保守作業によって高負荷となり障害となったケース、障害から復旧してサービスを再開した直後に待機していたユーザからのアクセスが集中して再びサービスが中断した事例など、原因は様々である。しかし、いずれの障害もシステムの高負荷／過負荷が契機（直接原因）となっており、この共通の観点からの障害分析や対策は、事故の防止のために有効であると考える。

4.4.2 直接原因の分類

システムが高負荷／過負荷状態に陥った原因は様々であるが、システム障害の報道事例や本教訓集の教訓事例から同じような原因と考えられる関係する事例を括って見ると次の 5 つに大別できた。さらに、その中でも、まとめられる事例を中分類として括ってみたところ、15 の原因に分けられた（図 4.4－1）。

システムが高負荷／過負荷状態になる原因は、その原因がシステム開発時に正しく想定できていたか、いなかったに分けられた。

システム開発時に正しく想定できていなかった理由は、もともとの想定が正しくなかった、「(1) もともとの処理能力不足」と、運用の中での一時的に生じた特殊事態を想定できなかった、「(2) 一時的なリソース不足」と、想定していた以上に処理能力が必要な事態が起きてしまった「(3) 想定を超える負荷」の 3 つに分類できた。

残りの原因は、システム開発時には正しく高負荷／過負荷状態になる事態を想定していたが、その後の運用や環境の変化にともない、当初の想定以上の高負荷／過負荷状態になった、「(4) 環境の変化への対応遅れ」によるものに分類することができた。

さらに、2018 年前半の事例から、外部要因による高負荷・過負荷が起こることが新たに加わった。このような外部要因としては、サイバー攻撃などによって起こる事例も過去にあったことから、それらも含め、「(5) 外部要因による高負荷・過負荷」とした。

次に、各大分類に含まれる事例を整理しながら、さらにどのような原因があったのかを、述べていきたい。

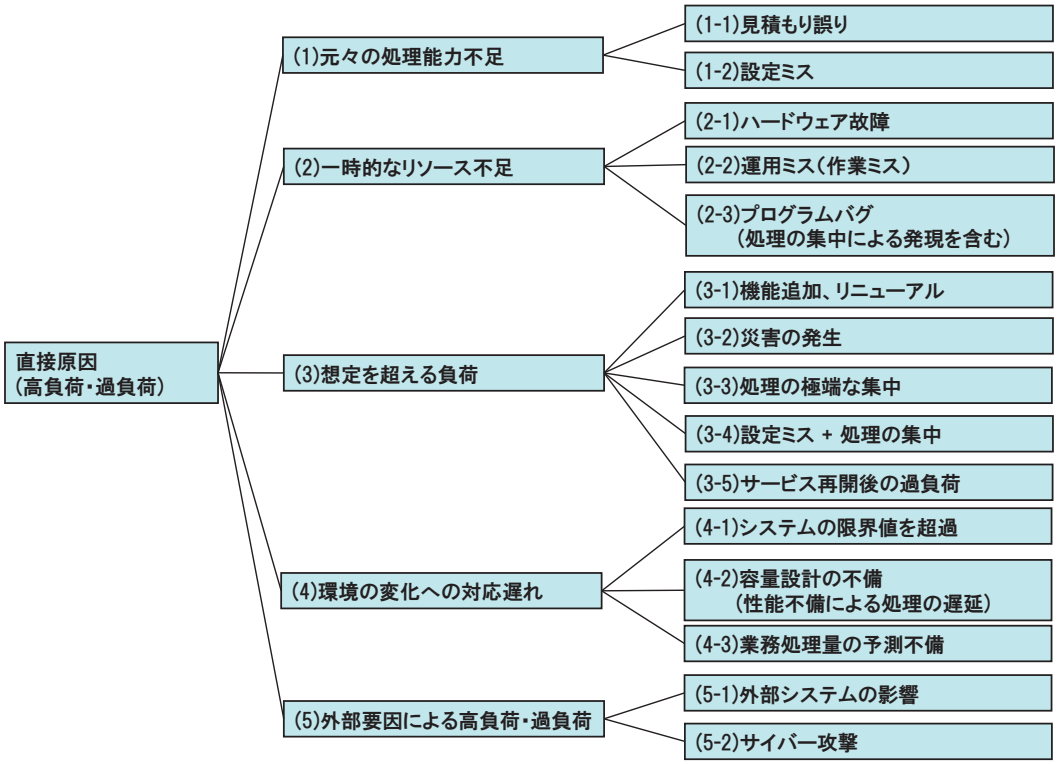


図 4.4－1　高負荷／過負荷　原因の分類

(1) もともとの処理能力不足

システムの高負荷／過負荷の第一に考えられる原因は、システム構築時の設計段階における、処理能力の見積もり誤りであろう。この原因は、さらに分析すると以下の 2 つに分類できる。

(1－1) 見積もり誤り

もともとの処理能力不足の原因を探ると、そもそもシステムへの負荷の見積りが誤っていたことが判明した事例があった。

【事例 1】映画祭・電子チケット販売システム

ある日 12:00 からのチケット発売開始後、アクセスが集中し、1 時間半あまりでシステムが停止した。発売開始直後から、Web サーバに 1 分間で 3 万 7000 件を超えるアクセスが集中した（想定の 6 倍以上）。新たにクラウド上に構築した本システムは、利用状況に応じて処理能力を増やせる仕組みだったが、最大限に拡張しても対応できなかった。チケット購入希望者のアクセスの仕方が、通常の映画ファンとは異なり、短い時間で何度もアクセスを繰り返すことを認識できていなかった。

教訓集の中では、この分類に入る事例は、【教訓 G6】が該当する。

（1－2）設定ミス

さらに、もともとの処理能力不足の原因を探ると、そもそもシステムへの負荷の見積りは正しかったが、設計ミス、容量設計ミス、確認ミスなどによって、システムの能力がそれを満足できていなかった事例があった。

【事例 2】自治体・住民基本台帳システム

証明書発行システムで障害が発生し、発行窓口に設置された約 3,200 端末のすべてで遅延が発生した。約 3 万 5 千件の書類発行が最大で 2 時間遅れた。年明けから「基幹系システム統合基盤」が稼働したが、印刷サーバの設定ミス（同時に実行できる印刷命令数の設定ミス）により、処理遅延が発生した。昼休み時間帯に、設定を暫定的に変更したところ、遅延は解消した。設定ミスの原因は、帳票作成用パッケージソフトウェアの仕様を把握していなかったため。

教訓集の中では、この分類に入る事例は、【教訓 T28】が該当する。

（2）一時的なリソース不足

一時的なリソース不足とは、システムの本番稼働中に、システムのリソースが一時的に不足し、相対的に負荷が高くなった（主に内的要因によるもの）事例をまとめてみたものである。

この原因は、さらに分析すると以下の 3 つに分類できる。

（2－1）ハードウェア故障

ハードウェア故障とは、ハードウェアの故障により、一時的にリソースが不足した事例である。

【事例 3】通信事業者・携帯電話システム

スマートフォンのサービスで、ホームページへの閲覧やメールの送受信がしにくい状態が 7 時間続いた。通信の中継設備が故障し、それにともない通信が集中したために発生した。

【事例 4】通信事業者・携帯電話システム

国際ローミングサービスが利用しづらい状況となり、220 の国と地域の最大 7 万人に影響を与えた可能性があった。国際電話用交換機の故障がきっかけとなって、国際共通線の輻輳および接続・切断の繰り返しが発生し、利用がしづらくなった。

【事例 5】自治体・防災情報システム

当該期間の約 1 カ月、システムを利用できない状態が続いた。処理が遅延し、4 ～ 5 時間前の情報が表示される状態に陥ったため、システムの稼働を止めた。原因はハードウェア故障。冷却ファンとハードディスクを交換すると、正常に稼働するようになった。ハードウェア故障は経年劣化によるもの。

(2－2) 運用ミス (作業ミス)

運用ミス（作業ミス）とは、運用作業で、作業ミスを犯したために、本番稼働時にシステムが一時的にリソース不足を起こした事例をまとめたものである。

【事例 6】通信事業者・LTE サービス

4G LTE 対応端末でパケット通信障害が発生、Web 閲覧やメール送受信が全国でできなくなった。通信異常のアラームに対処するためにシステム全体の復旧措置を実施したため、LTE 端末のセッションがすべて開放された。その結果、LTE 端末から一斉に再接続要求が発生し、過度にアクセスが集中し、新規接続ができない状況となった。 このアラームの対処としては、現用系システムから予備系へ切り替えれば済むはずであったが、手順書に記述されていなかった。

【事例 7】官庁・指令管制システム

電話が、4 時間にわたって着信してもすぐに切れてしまう状態となった。この時間帯の通報 712 件のうち、59 件で着信後すぐに電話が切れ、担当者が記録されている番号に折り返した。59 件のうち 8 件では再度通報があり、救急隊が出動した。原因は、通報を受ける装置の設定ミスだった。先月 24 日に固定電話用の回線を一部廃止したにもかかわらず、システムの設定を変えず、廃止した回線に接続確認のためのデータを送り続けたため、バッファオーバフローにより通報がつながりにくくなった。

【事例 8】法人団体　血液事業情報システム

全国各地の 150 カ所の献血ルームの約 7 割や採血車で献血を受け入れられない状態が発生。数千人が献血できない状態になった。献血者のデータなどを管理するシステムに障害システム内にある文書データのフォルダー名を変更したことで、サーバに負荷がかかり、障害が発生した。

教訓集の中では、この分類に入る事例は、【教訓 T8】が該当する。

(2－3) プログラムバグ (処理の集中による発現を含む)

プログラムバグ（処理の集中による発現を含む）は、一時的にリソースが不足した原因を探ったら、プログラムバグにより生じていたことが判明した事例である。

【事例 9】通信事業者・通信サービス

高速データ通信サービス LTE で障害が発生し、データ通信ができなかったり、利用しづらい状況となっていたりした。関東の一部利用者で最大 56 万台に影響した。さらに、この影響で、音声通信についても同日 9 時 30 分から 12 時 22 分の間、繋がりにくい状況となった。LTE 基地局制御装置に修正ファイルを投入中にハード障害が発生。切戻しを実施したが、その最中に輻輳が発生、リカバリ処理のバグが顕在化し、MME（移動性管理機器）の両系がダウンとなった。LTE の障害により、端末が LTE から 3G 網へハンドダウン。加入者情報管理システムが過負荷状態となり、接続が一部正常に行われず音声通信が困難または不可の状態となった。

【事例 10】損害保険会社・基幹情報システム

5月 12 日から断続的にシステム障害が発生し、保険契約の確認や事故対応などに影響が出た。5/15 にシステムを全面停止し、18 日に復旧させた。この間は代替システムや手作業で業務を実施したが、保険証券の送付など一部の業務が遅延した。11 日に実施したプログラムの変更が原因。このプログラムに不具合があったため、夜間バッチが中断し、処理を積み残した。翌日以降、バッチ処理と並行してプログラムの修正を試みたもののうまくいかず、処理能力に余裕がなくなり、保険申込書の新規登録や事故の受付等のオンラインが断続的に停止する事態になった。

(3) 想定を超える負荷

想定を超える負荷とは、システムの本番稼働中に、システムに想定を超える負荷がかかった（主に外的要因によるもの）ため、リソースが不足し、システム障害となった事例をまとめてみたものである。

この原因は、さらに分析すると以下の 5 つに分類できる。

(3－1) 機能追加、リニューアル

システムの機能追加・リニューアルにともない、システムを更新し、本番稼働に望んだが、想定を超える負荷がシステムにかかったために、システム障害となった事例である。

【事例 11】販売会社・ネットサービス

断続的なアクセス障害が発生した。アクセスがし難い不安定な状態が続いた。2102 万人のユーザに影響した。原因は、新機能追加が引き金となって高負荷が発生し、データキャッシュ制御ソフトのバグが顕在化したことであった。

【事例 12】鉄道事業者・IC カード

同日から開始された新サービスの影響で、サービスへのアクセスが集中し、利用がしにくくなった。IC カードの新サービスへの登録や、IC カードを使った座席予約券の購入など、オンラインサービスが利用しづらくなった。

【事例 13】官庁・電子申告・納税システム

インターネットを利用した地方税の電子申告で、当初は、繋がりにくい状態が発生したが、企業などが送信したはずの申告データが自治体に届かない事態が発生した。

a) 予想以上のアクセスが集中し、システムの負荷上限超え、
b) 一部の通信機器の再起動繰返しによるレスポンス遅延、
c) この期間中、正常終了でないにも関わらず、手続きを終えるケースが多発

教訓集の中では、この分類に入る事例は、【教訓 T14】が該当する。

(3－2) 災害の発生

台風などの災害の発生にともない、システムに想定を超える負荷がかかったために、システム障害と

なった事例である。

【事例 14】通信事業者・通信サービス

東日本全域の通信が不能になり、最大 70 万人に影響した。台風の影響などでアクセスが増大した。認証サーバの応答が遅延して大量のリトライ処理を繰り返し、制御サーバが過負荷状態に陥りシステムが停止した。再起動を試みるも、輻輳状況下で内在していたソフトバグにより再度システム停止が発生した。負荷を抑止しつつ再起動を順次行ったため復旧に長時間を要した。

【事例 15】自治体・Web サイト

台風の詳細情報が掲載されている自治体の Web サイトがダウンし、アクセスができなかった。台風の接近にともない、約 370 万人が住む自治体のほぼ全域の携帯電話に、緊急速報メールが配信された。「土砂災害のおそれがある」という内容だったが、システムの文字数制限が 200 文字のため、対象の地区は自治体の Web サイトを参照するよう案内された。その結果、Web サイトにアクセスが集中し、サーバがダウンした。

(3－3) 処理の極端な集中

処理の極端な集中とは、株式相場の急変やイベントの開催など、外的要因により処理が極端に集中し、システムに想定を超える負荷がかかったため、システム障害となった事例である。

【事例 16】金融機関共同センター・法人インターネットバンキング

全国 142 の金融機関で、企業向けのインターネットバンキングに接続し難くなり、取引できない状態になった。約 15 万社の企業に影響した。取引の極端な集中により、サーバの処理が遅延。本来は、トラフィック監視プログラムが自動的にサーバ負荷を調整すべきところ、正常に作動せず、取引処理能力が低下し、つながりにくい状態となった。

【事例 17】自治体・住民記録／国民健康保険システム

住民票発行や転入／転出などの手続きができなくなった。待機用のサーバを起動させ、証明書の発行業務に対応したが、25 件の手続きが滞った。トラブル前日の 2 月 18 日に、同システムのデータベースに対してディスク容量を上回るデータ量の書き込み処理を実行したため、システムが停止状態となった。

【事例 18】商品取引所

全商品の取引を一時停止した。注文件数が処理能力の上限に達したため。この日は米国大統領選で A 氏が市場予想に反して当選したことで、為替相場が大きく変動していた。

教訓集の中では、この分類に入る事例は、【教訓 G12、G13、G14】が該当する。

(3－4) 設定ミス ＋ 処理の集中

「設定ミス+処理の集中」とは、本番稼働中に、システムの設定ミスと処理の集中が重なったため、シ

ステムのリソースが不足し、システム障害となった事例である。このような複合要因が重なると、障害の影響は大きくなる。

【事例 19】通信事業者・ID 認証決済システム

ID を使った全サービス（ID の新規登録、変更、サービス利用）をはじめ、かんたん決済、電子メール設定、SNS サイトにおける年齢確認サービスなど ID 認証決済のサービスが利用できなくなった。月初めに行われる通信事業者の決済利用限度額のクリア処理によるアクセス集中に加え、データベースサーバ群のメモリ割付処理パラメータに誤りがあり、CPU に過剰な負荷がかかった。

【事例 20】金融機関共同センター

全国の銀行 7 行にてシステム障害が発生し、ATM の取引ができなくなるトラブルがあった。ATM の利用中に障害が発生した取引は、7 行合わせて約 4000 件だった。

1 回目（2/15）…取引制御プロセスの処理が遅延し、待ち行列が滞留。ATM の処理がタイムアウトした。原因は、ベンダから提供された製品マニュアルにミスがあり、取引量増加に対応するための作業に設定ミスがあった。

2 回目（2/26）…プロセスの滞留発生を、重点監視体制にて監視した結果、ログの書き込みが大量となった。大量ログデータの転送を開始した際、メモリが足りなくなりプロセスがタイムアウトし、ATM の停止に至った。

（3－5）サービス再開後の過負荷

サービス再開後の過負荷は、システムが停止していた後、サービスを再開しようとした際、システムに処理が集中し、一時的にリソースが不足した事例である。サービスの停止時間が長いと、サービス再開時に一挙に大量の取引や処理が集中するため、このようなことが起きやすいと考える。

【事例 21】証券所・売買システム

デリバティブ売買が全面停止した。A 売買停止にはじまり、10 時 51 分 B 先物、11 時過ぎには C 先物、D 先物、E 先物と次々と取引停止が拡大した。原因は、売買処理プログラムの不具合だった。注文処理直後のあるタイミングで一時的な通信障害が発生すると注文受口が開放されないままとなり、新規の注文受付けができなくなった。このため、取引参加者がログインとログアウトを繰り返したためシステムへの接続が困難な状況が続いたため、すべてのデリバティブの売買を停止した。

【事例 22】通信事業者・携帯電話システム

携帯電話のうち、E メールリアルタイム受信設定を行っていた端末について E メール送受信サービスが利用できなくなった。一旦解消するも同様事象が再発した。その後、解消したが利用しづらい状況が数日間継続した。影響を受けた端末は最大 288 万台だった。新機能追加のための保守作業においてミスがあり、ユーザ情報に不一致が生じた。不一致を解消した上で新設備への切替えを実施したが、その作業中に片系にハード障害が発生、残った正常系が過負荷に陥りダウンした。その後、再起動に成功するが、滞留したメールなどの処理で過負荷状態が継続した。

(4) 環境の変化への対応遅れ

開発の終了後、システムは長期にわたって運用されるため、その間にシステムを取り巻く環境は大きく変化する。開発時点では妥当であった性能条件が、最新の利用条件を満足しなくなったために、システムの処理能力を超えたケースである。これは、「(3) 想定を超える負荷」と似たケースであるが、(3) は、ピークがある特定な場合に来る現象であるのに対し、この場合は定常的に過負荷になるケースである。

この原因は、さらに分析すると具体的に以下の 3 つに分類できる。

(4－1) システムの限界値を超過

システムの限界値を超過とは、システム設計時に見積もったシステムの限界値が、システム環境の変化（物理量、運用、環境）によって変化していたため、限界値を超えないと思っていたシステムが、限界値超過したために起きたシステム障害事例である。

【事例 23】鉄道会社・運行管理システム

列車運行管理システムでダイヤの変更入力を行った際、予想ダイヤが表示されなくなったため、確認のため全列車を停止させた。列車 8 本が立ち往生した。運休・遅延本数は 139 本となり、8 万 1,200 人に影響した。列車運行管理システムにおいて、ダイヤ変更入力時に、修正データ数がシステムの限度値 600 件を超えると、予想ダイヤを表示出来ない実装となっていた。また、このことに関する情報共有が出来ていなかった。

【事例 24】銀行・勘定系システム

オンライン取引の開始が大幅に遅延した。一部の取引はその後も利用できなかった。前日の夜間バッチ処理の異常終了により、決済処理が 38 万件未処理となった。16 日以降も決済未処理件数は増加し、3 連休中（19 日～ 21 日）はオンライン業務を完全停止し、未処理の解消を計り、22 日時点で大半が解消したが、完全な解消は 24 日となった。震災の義援金振込みが、特定の支店口座に集中し、あらかじめ設定してあった夜間バッチ処理の一口座当りの処理件数の上限値を越えたため、夜間バッチ処理のエラーが多発し、処理の大幅な遅延を招いた。

教訓集の中では、この分類に入る事例は、【教訓 T4、T18、T32】が該当する。

(4－2) 容量設計の不備 (性能不備による処理の遅延)

容量設計の不備（性能不備による処理の遅延）とは、環境の変化への対応遅れから、データ容量や処理能力の不足に対する対策を行わなければならない事態であったにも関わらず、その対応を怠ったために起きたシステム障害事例である。

【事例 25】通信事業者・携帯電話システム

携帯電話の音声通信やメールの送受、インターネットへの接続などデータ通信がしづらくなり、約 252 万人に影響した。スマートフォン契約者の増加に対応するために、新型パケット交換機への切替えを実施した。 トラフィックの上昇にともない、新型パケット交換機の動作が不安定な状態となり、ネットワー

クの自動規制により、繋がりにくい状況となった。 スマートフォンのアプリケーションによる制御信号のトラフィックが増加しパケット交換機の処理能力がオーバフローした。

【事例 26】鉄道会社・IC カードシステム

IC カード一部会員に対する 2014 年 2 月分の請求が 1 カ月遅れた。影響を受けた会員は約 32 万人、件数は約 80 万件だった。会員が他のクレジットカード会社の加盟店で IC カードを使った金額の請求業務に必要なバッチ処理が、期日どおりに完了できなかった。2 月 26 日から滞留が発生し、合計約 80 万件の請求処理が 3 月 4 日までに完了せず、一部会員への請求が 1 カ月間遅れた。対策として、バッチ処理に要した時間について、計画と実績の乖離状況を毎月チェックするなどの運用体制を強化した。

【事例 27】動画配信会社・動画配信システム

スポーツリーグのライブ動画が見られないトラブルが起きた。また、見逃し映像の配信もストップした。16 時ごろに終了したスポーツリーグ 7 試合を配信直後、アクセスが集中した。原因は、ログが急増し、リソース不足により処理が滞留したことだった。

教訓集の中では、この分類に入る事例は、【教訓 G12、G13、G14】が該当する。

(4－3) 業務処理量の予測不備

業務処理量の予測不備とは、業務の変化への対応遅れから、データ容量や処理能力の不足に対する対策を行わなければならない事態であったにも関わらず、その対応を怠ったために起きたシステム障害事例である。

【事例 28】共同センター

業界共同システムで、稼働後 1 年が経過し、業界普及率がシステム稼働当初から 2 倍に増加する時期に差し掛かっていた。業務の特性上、X システムは年度末に処理が集中する傾向があるが、キャパシティ予測が不十分であったため、想定を大幅に上回る負荷がサーバにかかったことによりミドルウェアの潜在バグが発現し、システムダウンとなった。潜在バグの修正はすぐには対応できず、またキャパシティの増強も時間がかかるため、年度末、数日間にわたってシステムがダウン、あるいはレスポンスが極度に悪化する事態となり、運用上、大幅な利用制限を行わざるを得ず、業務が著しく停滞した。

教訓集の中では、この分類に入る事例は、【教訓 G5】が該当する。

(5) 外部要因による高負荷・過負荷

システムが高負荷・過負荷になる要因として、外部環境からの影響によるものが存在する。一つは、外部システムの不具合が自システムに影響した事例と、二つめは、サイバー攻撃によるものである。

これらの要因は、自システム開発時に判明している場合であれば、当然設計時に対策を組み込めるが、運用時に起きた要因であれば、システムの次の開発や更改時に対策を組み込むしかなく、むしろ、そのような現象が起きたことを、素早く正確に把握できなかったことが問題となる。

(5－1) 外部システムの影響

近年のインターネットの普及は、思わぬところで起きたシステム障害が多くの事業者の基幹システムに影響を及ぼすことがある。

【事例 29】米グーグル

多くの企業では 8 月 25 日 12 時 24 分から 17 時ごろまで、一部のネットサービスが利用しにくくなるシステム障害が発生。スマートフォン向け電子サービスや電子マネーのチャージ等多くの機能が不安定になった。そのほか、金融機関、証券会社などのネットサービスなども一時使えなかった模様。

米グーグルから大量の経路変動があり、国内通信事業者を介してインターネットに接続していた企業のルータが大量の経路情報を受け取り高負荷となり、通信障害につながった。原因は、米グーグルが 8 月 25 日 12 時 22 分ごろに、国内通信事業者とピアリング（対等な関係でネットワークの経路情報をやり取りすること）していた時、誤った経路情報を大量に送ったことによる。

(5－2) サイバー攻撃

サイバー攻撃（意図した要因で起こる）は、システム障害（意図しない要因で起こる）とは別の観点での原因、対策が取られるものであるため、本教訓集では、対象外としていたが、システムの高負荷・過負荷の問題での外部要因として考える場合、システム障害として一緒に考えることができると考え、事例として、取り上げた。

【事例 30】DoS 攻撃[23]

DoS とは "Denial Of Service" のことで、提供するサービスを妨害したり、停止させたりするものを指す。サービスを妨害する攻撃は、「過負荷をかけるもの」と「例外処理ができないもの」の 2 種類に分けることができるが、このうち、「過負荷をかけるもの」としては、以下のような事例がある。

- 巨大なメールや大量のメールを送りつけメールサーバのディスクや CPU 資源、ネットワークの帯域を潰す。
- SYN パケット[24] を送信し相手側を無限ループに陥らせる。
- 相手に対し多くのコネクションを攻撃するサーバに張り運用ができないようにするもの。

【事例 31】DDoS 攻撃[25]

これは、"Distributed Denial Of Service" の略で、DoS 攻撃を行うホストがネットワーク上に分散している形態である。つまり、DoS 攻撃の場合は、攻撃側と相手側の 1 対 1 で行われるが、DDoS は攻撃側が複数存在し、1 台のサーバを攻撃する。例えば、1,000 台が 1 台のサーバを攻撃する。

DDoS の攻撃を受けた側は、1,000 台、10,000 台といった多くのホストから一斉に攻撃を受けるため、サービスやネットワークが過負荷となり、停止することになる。

---

[23] IPA セキュリティセンター「DoS 攻撃、DDoS 攻撃」
[24] 接続を確立するために送られるパケット
[25] IPA セキュリティセンター「DoS 攻撃、DDoS 攻撃」

（原因のまとめ）

以上のように、ひと口に「高負荷／過負荷」とは言っても、障害が発生した状況や、システムが稼働している環境まで含めて考えてみると、その原因は非常に多岐にわたることがわかる。

4.4.3 対策

前章までに示してきたように、高負荷・過負荷によって障害が発生する原因は様々であるため、何か一つ実施すればすべて解決するような、特効薬的な対策があるわけではない。開発、運用にわたるすべての工程において、適切な対策の積み重ねが必要である。そのため、どのシステム障害対策は、（図 4.4－2）のように、フェーズごとに対策を実行していくことが重要である。

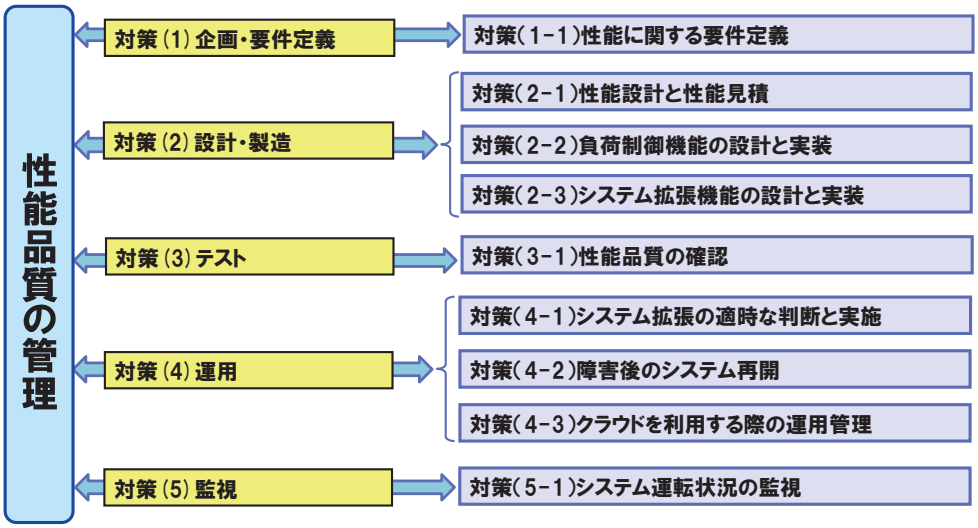


図 4.4－2　性能品質の管理のための工程ごとの対策

往々にして、システム性能に関しては、開発の終盤に入ってから確認・調整を行えば足りるとする傾向があるが、それでは全く不十分であり、すべての工程において性能品質を作り込むことが重要である。

以下、工程ごとに主要な対策を示す。さらに、教訓集では、性能問題に関係する教訓がいくつかあるので、それらを「紐付く教訓」として参照できるようにしたので、具体的な対策については、それらの教訓もあわせて参照して頂きたい。

（1）企画・要件定義工程

（1－1）性能に関する要件定義

企画・要件定義の工程において、システム性能に関する非機能要件を定めることは不可欠な作業項目である。しかし、非機能要件はその項目が多様であり、技術的にも複雑な内容を持つものであるため、機能要件に比べると、要件を抜けなく適切に定義することはやさしくない。このため、IPA では非機能要件を比較的簡単に定義できる手法として、非機能要求グレードを公開している。この中では、性能に

関する非機能要件として定義すべきものとして、以下の 4 項目をあげており、その詳細が示されている。

① 業務処理量
② 性能目標値
③ リソース拡張性
④ 性能品質保証

また、オンライン及びバッチの性能を設計する際に、通常時とピーク時、それぞれの場合について検討すべきことが示されている。

これらを参考に、性能に関する非機能要件を抜けなく設定し、設計へインプットすることが必要である。

関連する教訓

- T28　パッケージを更新するときは、変更内容の詳細確認と回帰テストで二重に安全を確保せよ

(2) 設計・製造工程

(2－1) 性能設計と性能見積

設計にあたっては、機能要件の実現のための検討に終始し、性能要件をはじめとする非機能要件の実現に向けた検討が疎かになる傾向がある。処理方式の選択などにおいては、性能要件の実現の見地からの検討も重要である。処理方式についての性能見積りを机上で行い、常に設計にフィードバックすることが必要である。

(2－2) 負荷制御機能の設計と実装

非機能要件として決めた業務処理量や性能条件は、過去のデータからの推定をもとに、実現性やコストなどを総合的に判断して定めたものであり、実際に生ずるトラフィックがこれを超えない保証はどこにもない。設計条件を超えるトラフィックが実際に生じたときにも、システムへの入力を適切に制御する負荷制御機能により、システムが安定して動作するよう制御することが重要となる。このための機能の実装が必要である。

(2－3) システム拡張機能の設計と実装

原因の項で述べた通り、環境の変化にともない運用期間中に、当初に想定した性能条件を越えてシステムの拡張が必要になるケースもある。システムの運用期間中に拡張が想定されるシステムにおいては、あらかじめシステム拡張のための機能を組み込んでおくことも必要となる。ハードウェアの能力拡張だけであれば、より性能の高いハードウェアに置き換えること（スケールアップ）やクラウドの利用によってスケールアウトも可能であるが、ソフトウェア的な制約についてはあらかじめ設計に反映しておくことが必要であり、注意を要する。

(3) テスト工程

(3－1) 性能品質の確認

開発の最終段階では、実際に運用を行うシステム構成の最終確認を行う必要がある。性能条件を満足するために、CPU、メモリ、ファイルなど、各種計算機資源の適切な容量設計の確認を行う。

可能ならば、実負荷による性能の確認、さらに負荷変動に対する耐力の確認など、それ以前の工程で実施した対策が所期の目的を果たすべく正しく実装されていることを、実システムにおいて確認する。また、障害発生時に縮退運転を想定している構成の場合には、縮退運転時に想定した性能が正しく発揮されるかも確認しておくと良い。

テストに利用できる資源の制約や運用上での制限などにより、実運用と同じ環境でのテストは困難な場合が多いが、工夫をしてできる限り実環境に近い状態での確認が重要である。

(4) 運用工程

(4－1) システム拡張の適時な判断と実施

せっかく拡張機能が実装されていても、それを適切なタイミングで実施しなければ意味がない。そのためには、システム拡張の判断基準をあらかじめ決め、上記の運転監視の状況に照らし合せるなど、拡張判断が適時・迅速に行えるような手順や体制を整備しておくことが重要である。また、設定した拡張条件は環境の変化に合わせて見直すことも必要となる。

関連する教訓

- G14　設計時に定めたキャパシティ管理項目は、環境の変化にあわせて見直すべし
- T4　システムに影響する変化点を明確にし、その管理ルールを策定せよ！
- T32　周期処理、「時間」と「変化」を監視せよ！

(4－2) 障害後のシステム再開

システムが障害から復旧し、サービスを再開した直後にトラフィックが集中して、再びトラブルになるという事例もある。多くの処理要求がサービスの再開を待って待機した状態であり、サービス再開と同時に処理要求が集中するなど、通常のサービス開始時とは異なったトラフィック特性となる。システムへの負荷を制御し、システムの処理能力を超えないようにコントロールしながら徐々に再開するなどの慎重な対応が必要である。

関連する教訓

- T24　サービス縮退時の対策を考慮せよ

(4－3) クラウドを利用する際の運用管理

教訓 T8 に見られるように、システムをクラウドに移行する際、各サーバの運用要件を整理せず移行すると、リソース不足や性能不足を起こし、クラウド利用の効果が上がらない事態に陥ることがある。クラウドへの移行を行う場合、リソース管理、性能監視を設計時に十分に考慮する必要がある。特に、サーバ同士をグループ化する場合、仮想化ソフトウェアが追加されることにより、オーバヘッドが増大することを見逃してはならない。仮想サーバグループの性能は、「サーバ台数分＋仮想化ソフトウェアのオーバヘッド」として見積もる必要がある。

また、このような運用設計を実施する中で、システム部門は、サービス開始までに要員のスキルを十分に高めるために、障害対策の検討、障害対策マニュアルなどの作成を行い、運用要員教育、障害訓練（移行時、稼働時）を実施する必要がある。

関連する教訓

- T8　仮想サーバになってもリソース管理、性能監視は運用要件の要である

(5) 監視

(5－1) システム運転状況の監視

システムの運転状況の監視に当たっては、システムへの処理要求の推移や、それに対する応答時間の状況などを監視する必要がある。高負荷／過負荷の状態に至るには、それ以前に入力の急激な増加や応答時間の延伸などの兆候が見られるケースもあり、場合によっては障害発生の前に対策が可能である。

また、システムの高負荷／過負荷に係る障害は、一見システムは正常に運転されているように見えるにもかかわらず、システムに接続しづらい、応答が極端に遅い、といった利用者からの申告によって検出される、いわゆるサイレント障害の例もあり、適切なシステム運転状況の監視が求められる。

さらに、外部環境からの影響による、外部システムの不具合が自システムに影響する場合とサイバー攻撃による場合の対策は、この監視機能が重要となる。具体例としては、システムで流れるネットワークトラフィックを分析・可視化し、通常と異なるトラフィックの動きを監視し、攻撃や異常を早期に発見する機能をもたせ、気がついた異常個所を止めたり、トラフィックを迂回させたり、などの機能を持つ仕組みが必要になる。

関連する教訓

- G5　サービスの拡大期には業務の処理量について特に入念な予測を実施すべし
- G12　キャパシティ管理は、業務部門とシステム部門のパートナーシップを強化するとともに、管理項目としきい値を設定して PDCA サイクルをまわすべし
- G13　キャパシティ管理は関連システムとの整合性の確保が大切
- T11　サイレント障害を検知するには、適切なサービス監視が重要
- T14　Web ページ更新時には、応答速度の変化等、性能面のチェックも忘れずに
- T18　新たなサブシステムと老朽化した既存システムとを連携する場合は両者の仕様整合性を十分確認すべし
- T22　隠れたバッファの存在を把握し、目的別のしきい値設定と超過アラート監視でオーバフローを未然に防止すること

(6) 原因と対策のまとめ

（図 4.4－3）に、「原因と対策一覧」を示す。原因から対策に「→」が出ていない個所もあるが、一切結びつかないというわけではないが、関連性が弱いので、線がある点を中心に考えて頂きたい。

4.4.4 本取り組みによる効果

問題の項で述べた通り、高負荷／過負荷をきっかけとする障害は一定の比率で発生している。これら

の対策によって、障害を少なくすることが期待される。

　また、高負荷／過負荷状態となった結果、ソフトウェアに内在していた潜在バグが顕在化した例もある。性能確認テストの実施が、バグの叩き出しにも効果をもたらし、結果的にシステムの安定性の向上に寄与する効果も期待できる。

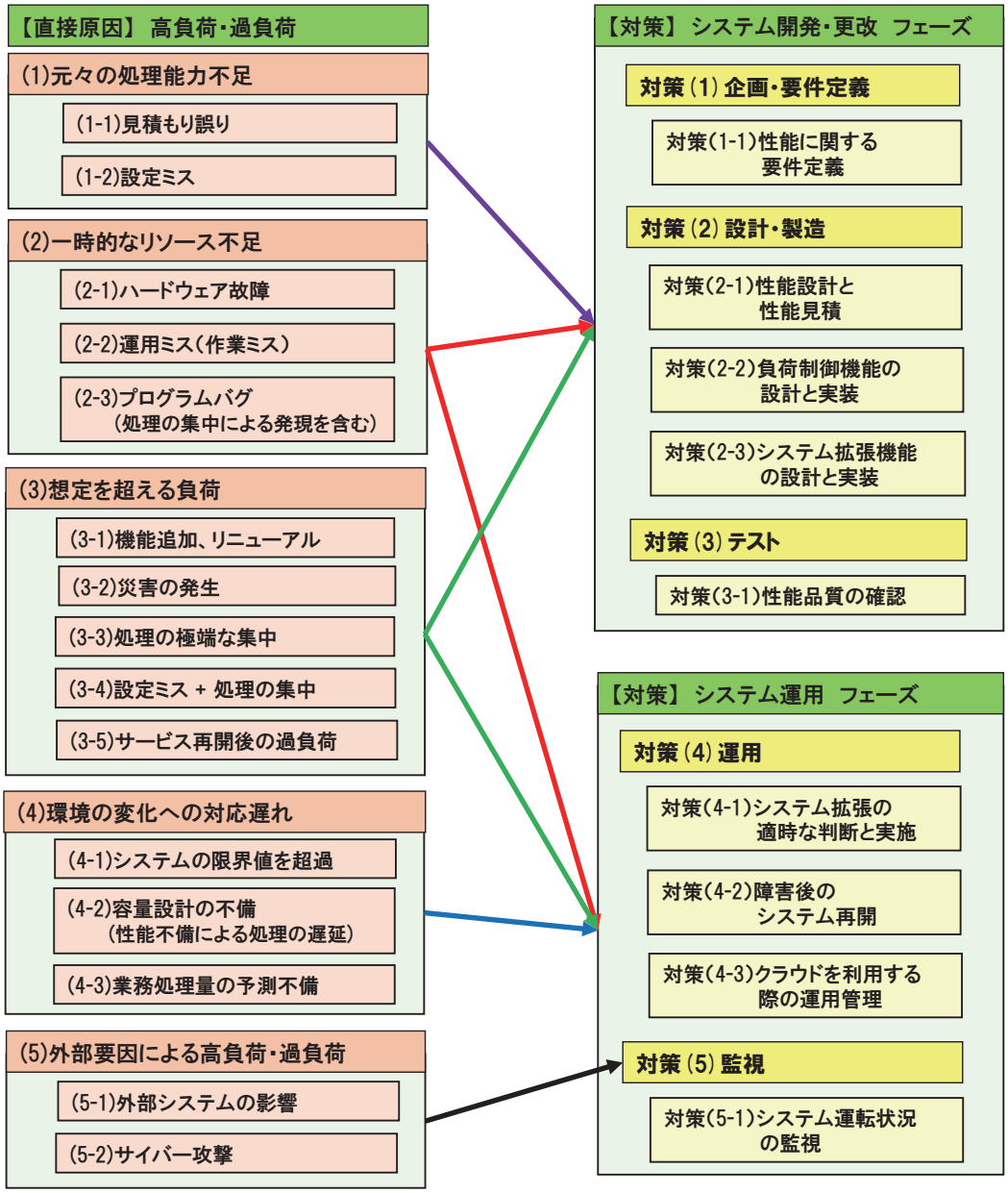


図 4.4－3　原因と対策一覧

4.5 「注意すべき観点」に基づく障害の分類

障害内容には多種多様な分類方法（業種別、工程別、発生個所別、原因別、影響別等）が考えられるが、読者に気づきを与える「注意すべき観点」に基づいて分類した。

公開している障害事例一覧表では、4 段階の分類に整理して障害事例を掲載している。大分類を JIS Q 20000-1:2012 に基づいた分類、中分類以下の 3 段階を「注意すべき観点」で分類しており、それぞれに該当する事例をあげている。

障害事例一覧表は下記の Web ページを参照されたい。

▼「注意すべき観点」に基づいた障害事例の分類
URL: https://www.ipa.go.jp/sec/system/index.html#shougaijirei

ここでは、小分類段階の注意すべき観点のうち、特に類似点の多い、以下の 10 種の注意すべき観点および該当事例について紹介する。

① 計算処理の誤り
② 検知条件の想定もれ
③ テストによる副次作用
④ 待機系への設定もれ
⑤ 障害発生ケースの想定もれ
⑥ しきい値の超過
⑦ ログの肥大化
⑧ 製品仕様の誤解
⑨ 不完全な作業実施
⑩ 作業中偶発事象への考慮不足

それぞれの項の表には、注意すべき観点（詳細段階）ごとに事例を記載している。事例番号のうち、G または T で始まるものは情報処理システム高信頼化教訓集（IT サービス編）、数字 4 桁は「情報システムの障害状況一覧」のものである。

4.5.1 計算処理の誤り

処理条件がもれる、処理対象を誤る、処理条件を誤る、実装時に変数名を誤る等の原因で、計算処理を誤った障害が発生している。

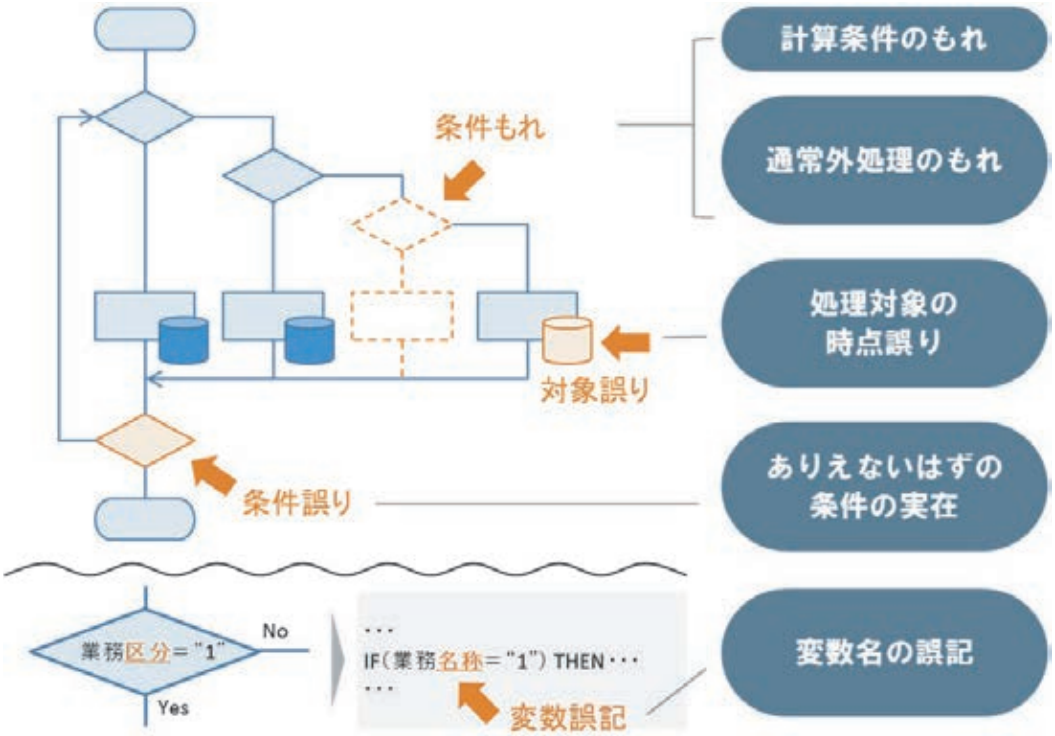


図 4.5.1－1 「計算処理の誤り」の分析

表 4.5.1－1 「計算処理の誤り」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
計算条件のもれ	1703	電力需要の計画と実績の過不足量（インバランス）算定時に、本来計算に加える必要がある値（域外分）が欠落。全国的な事業者の精算取引に影響した
	1704	スマートメーター設置の顧客には振込用紙郵送を行わないという判定条件を漏らして設計し、該当顧客に振込用紙を重複送付してしまった
	1728	測定時に特定事象が発生すると、測定データが保存されないまま計算プログラムが進行し、一部の放射性廃棄物の放射能量が少なめに評価された
通常外処理のもれ	T5	加算を主体とした業務処理（使用料計算）で減算処理が発生し、誤請求を行ってしまった
処理対象の時点誤り	1425	高額療養費は診療月時点での世帯単位で計算する必要があるが、診療月後に世帯変更があった世帯に対して変更後の世帯単位で計算してしまった
	1727	あるインターチェンジを利用した自動車に対して、ETC のプログラムミスにより利用時刻が実際より 1 時間早く記録され、料金が誤請求された
ありえないはずの条件の実在	1419	1,000 件連続で「データなし」を処理終了とする仕様に対して実データで当該条件が発生したため、後段の送金処理が未完了となった
変数名の誤記	1705	条件分岐処理において、区分を示す変数を入れるべき場所に名称を示す変数を入れてしまい、分岐が機能せず、臓器移植患者の待機日数計算を誤った

4.5.2 検知条件の想定もれ

業務上の様々なイベントをシステムが検知する際に、検知処理の設計に誤りが埋め込まれやすく、これを原因とした障害が発生している。

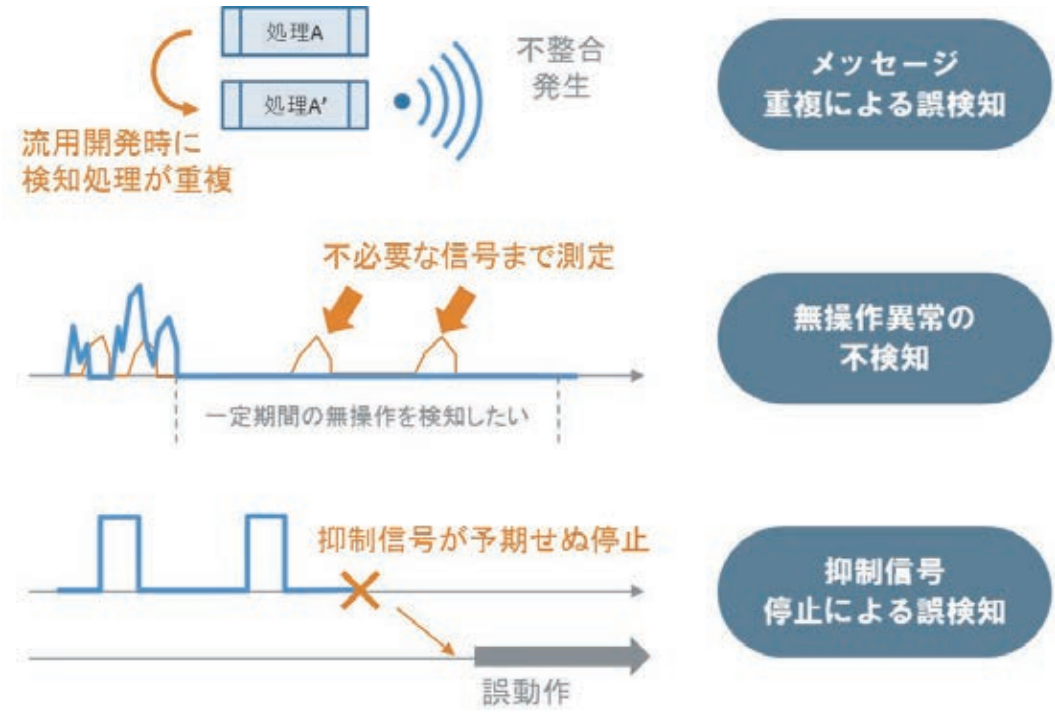


図 4.5.2－1 「検知条件の想定もれ」の分析

表 4.5.2－1 「検知条件の想定もれ」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
メッセージ重複による誤検知	T26	流用開発時に同一のジョブ完了メッセージを重複作成したため、ジョブ実行順序が変わりバッチ処理に論理矛盾が発生した
無操作異常の不検知	1431	運転士の一定時間無操作を検知する仕組みで、自動列車制御による減速を乗務員操作と誤って検知し、本来の異常検知を行えていなかった
抑制信号停止による誤検知	1231	発信用サーバに定期的に受信していた通行止め情報が途切れたため、復旧したと誤判断し、通行止め解除のメールが誤って自動送信された

4.5.3 テストによる副次作用

本番環境を用いたテストや切替実施時に、テスト用の設定を残存させてしまったこと等を原因とする障害が発生している。

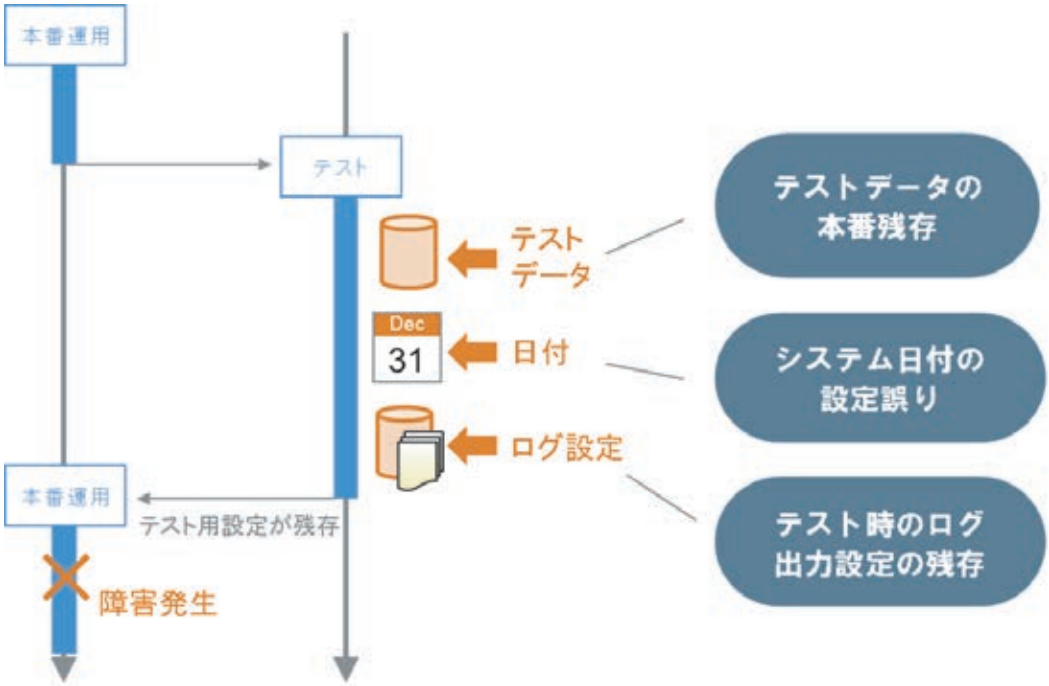


図 4.5.3－1 「テストによる副次作用」の分析

表 4.5.3－1 「テストによる副次作用」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
テストデータの本番残存	1405	改札機に対して消費税改定を想定したテストを実施後、データを戻す作業を怠り、利用者から料金を過剰に収受した
	1013	テストデータの削除を忘れてバッチ処理を実行したため、残高証明書の発行手数料を二重に引き落としてしまった
	1739	リハーサルのみに使用するプログラムが本番環境へ適用されてしまい、取引反映の処理に異常が発生
システム日付の設定誤り	1409	消費税改定を想定した切替で、1台の路線バスにのみ誤って日付を1日早くセットし、消費税改定1日前から増税後の運賃を徴収した
テスト時のログ出力設定の残存	1215	システムテスト中にログを詳細に出力する設定としたまま、それを修正せずに本番運用を行い、カード発行処理が大幅に遅延

4.5.4 待機系への設定もれ

稼働系と待機系は基本的に同じ設定内容に保つ必要があるが、待機系での設定を誤ったために、稼働系への障害発生時に待機系も起動できなかったという障害が発生している。

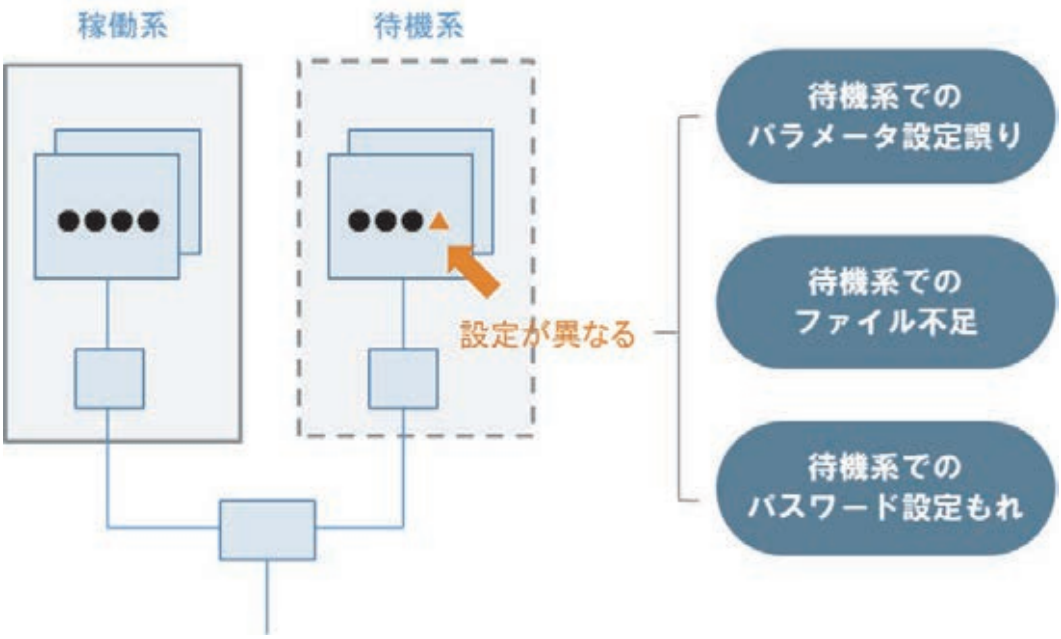


図 4.5.4－1 「待機系への設定もれ」の分析

表 4.5.4－1 「待機系への設定もれ」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
待機系でのパラメータ設定誤り	T7	稼働系と待機系の同期を取るべきソフトウェアパラメータにもかかわらず、待機系への設定を怠り、待機系が起動しなかった
待機系でのファイル不足	G11	保守作業での設定ミスで待機系側に必要ファイルが存在せず、障害時に切替失敗
待機系でのパスワード設定もれ	1640	稼働系のみパスワードを変更し待機系のパスワードを変更せず、データ同期時にエラーが発生

4.5.5 障害発生ケースの想定もれ

部分的な故障が発生した際に、複数の処理が競合して障害が拡大したり、エラーメッセージが大量に出続けることで二次障害を誘発する事例が発生している。

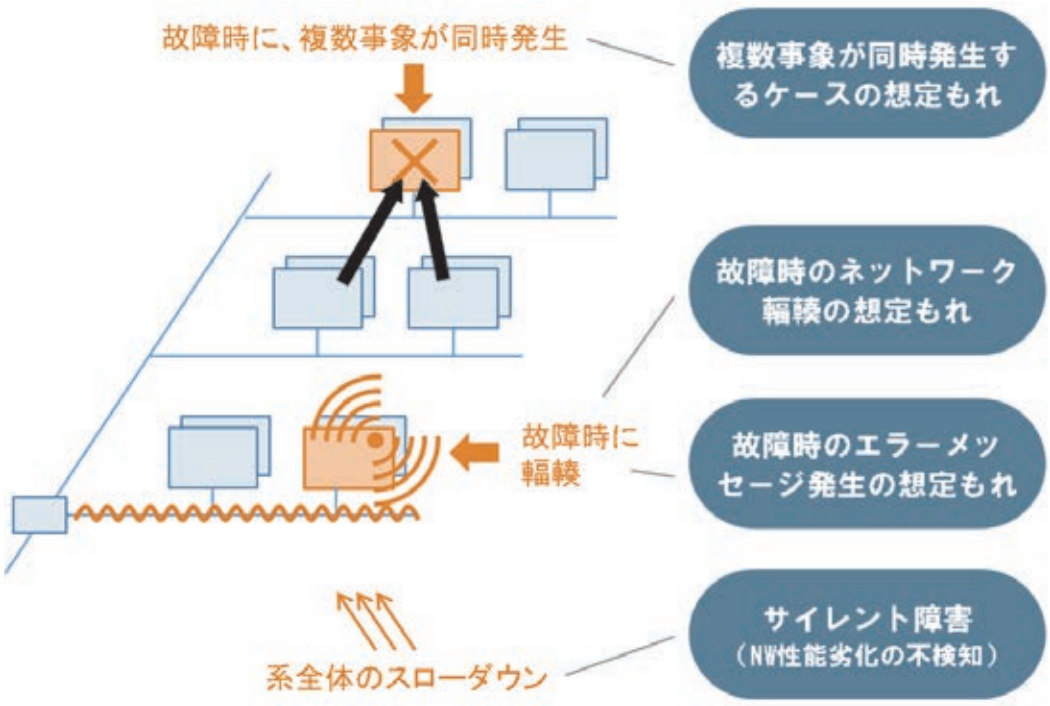


図 4.5.5－1 「障害発生ケースの想定もれ」の分析

表 4.5.5－1 「障害発生ケースの想定もれ」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
複数事象が同時発生するケースの想定もれ	T23	2 つの監視機能（DB 同期、自ノード監視）が偶然重複したため、待機系切替用の最後の DB サーバまでが停止
	T20	2 つの telnet 接続（障害情報収集、自動監視）が競合し、無限ループが発生
故障時のネットワーク輻輳の想定もれ	T2	ハードディスクの故障で「リセット通知」が出続け、処理渋滞で一部の通信が途切れ、制御監視端末からの系切替えが行えなかった
故障時のエラーメッセージ発生の想定もれ	1007	ディスク装置故障によりシステムがダウンした際にエラーメッセージが大量に発生し、連携先のシステムもダウンした
サイレント障害（NW 性能劣化の不検知）	T11	負荷分散装置でリクエストが廃棄されていたが、廃棄数がしきい値未満であったため検知できなかった
多数同時故障の想定もれ	1801	ハードディスクが 2 台故障しても自動復旧可能な仕組みにしていたものの、3 台が同時故障したため、自動修復できず復旧に時間を要した
ケーブル切断時の想定もれ	T30	2 重化配線されていたのにもかかわらず、両系のケーブルが束になっていたため同時に切断され、サービスが停止

4.5.6 しきい値の超過

システム内部には、関係者が認識している明示的なしきい値もあれば、関係者が認識できていない暗黙のしきい値もある。外部環境の変化や、長期の継続運用の結果、これらのしきい値を超過したことによる障害が発生している。

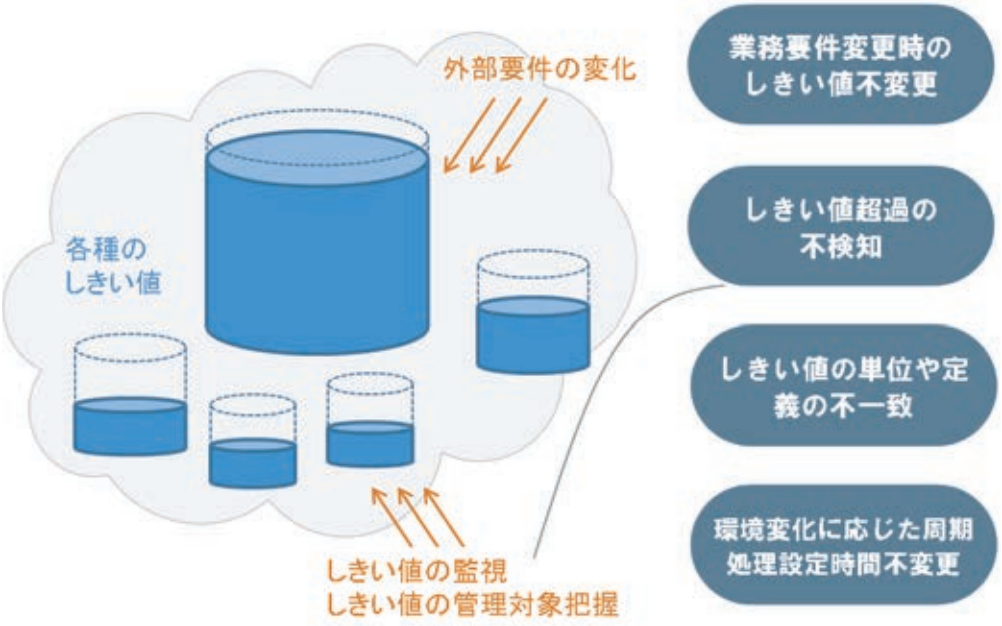


図 4.5.6－1 「しきい値の超過」の分析

表 4.5.6－1 「しきい値の超過」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
業務要件変更時のしきい値不変更	T4	予測可能時間を増大した際に予測対象列車数の上限値を変更せず、予測ダイヤを表示できなくなった
しきい値超過の不検知	T22	スローダウンを契機に、重要性を認識していなかった予約処理のバッファが連鎖的に満杯となり、全体機能が停止
しきい値の単位や定義の不一致	T29	システム間で転送されるデータの制限値について、当日分のみか、当日と翌日を含めたものかの理解が異なり、制限値を超えた結果システム障害を招いた
環境変化に応じた周期処理設定時間不変更	T32	周期処理の時間設定が機器やデータ増に追随できていなかったため、処理が時間内に完了せず機器動作停止および制御ネットワーク障害が発生

4.5.7 ログの肥大化

しきい値の一種でもあるが、ログの容量が想定以上に増加したことによる障害が数多く発生している。

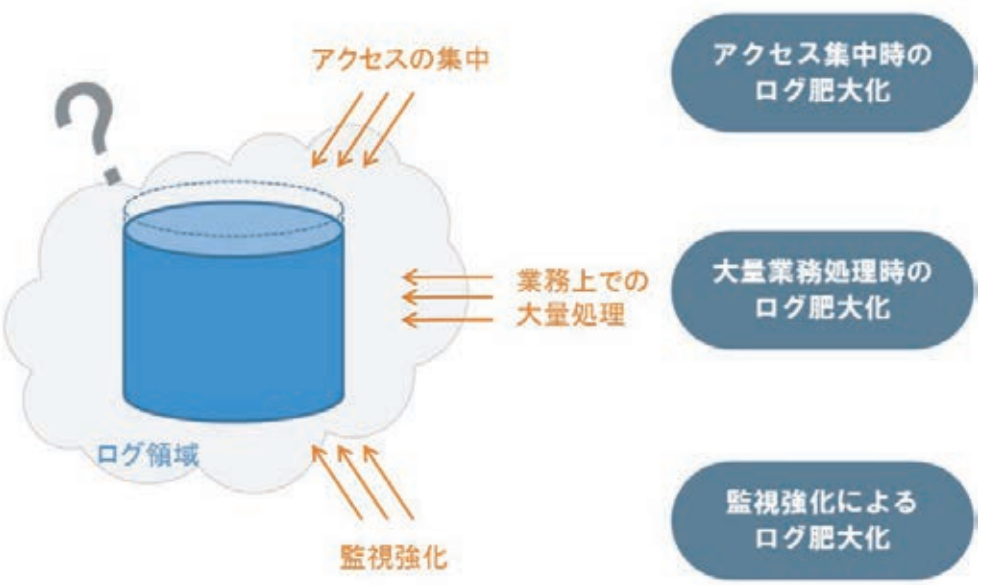


図 4.5.7－1 「ログの肥大化」の分析

表 4.5.7－1 「ログの肥大化」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
アクセス集中時のログ肥大化	1710	動画配信サービスでアクセス集中時にログが急増し、リソース不足により処理が滞留。配信予定のライブ中継映像が提供できなかった
大量業務処理時のログ肥大化	1507	大規模マンションの住民の地番修正時に同マンションに入居する他住民もログに記録するため、ログ容量が超過し障害発生
監視強化によるログ肥大化	1611	滞留プロセスを重点監視した結果、ログが大量に記録され、ログデータ転送時にメモリ不足となり ATM が停止

4.5.8 製品仕様の誤解

ハードウェア、ソフトウェア等の各製品には、独自の制約事項や仕様条件が存在することがある。この点を熟知せずに製品を利用したことによって、障害が発生している。

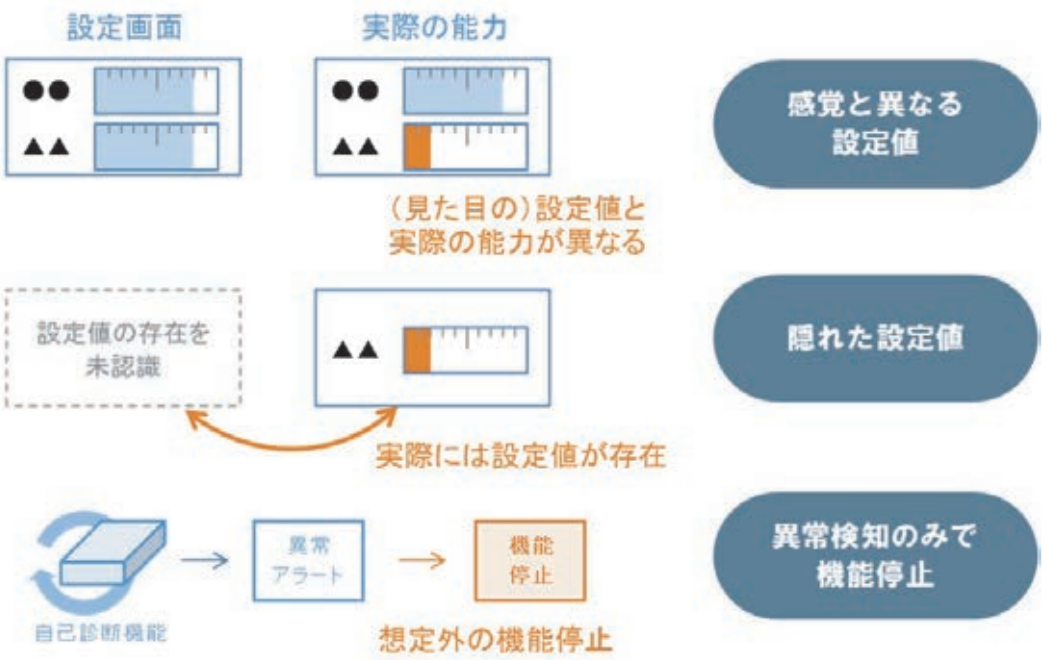


図 4.5.8－1 「製品仕様の誤解」の分析

表 4.5.8－1 「製品仕様の誤解」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
感覚と異なる設定値	T11	負荷分散装置のセッション数が設定値の 1/4 となる「仕様」のため、応答速度が低下した
隠れた設定値	1501	帳票作成用パッケージの仕様を把握しておらず、同時に実行できる印刷命令数の設定を誤り、証明書発行システムで障害発生
異常検知のみで機能停止	G11	ディスクモジュールの自己診断機能で、異常検知のみで機能停止する仕様となっていた

4.5.9 不完全な作業実施

システムの運用作業の中で、主要となる作業は確実に実施しつつも、最終的な作業や確認がもれたことによって障害が発生している。

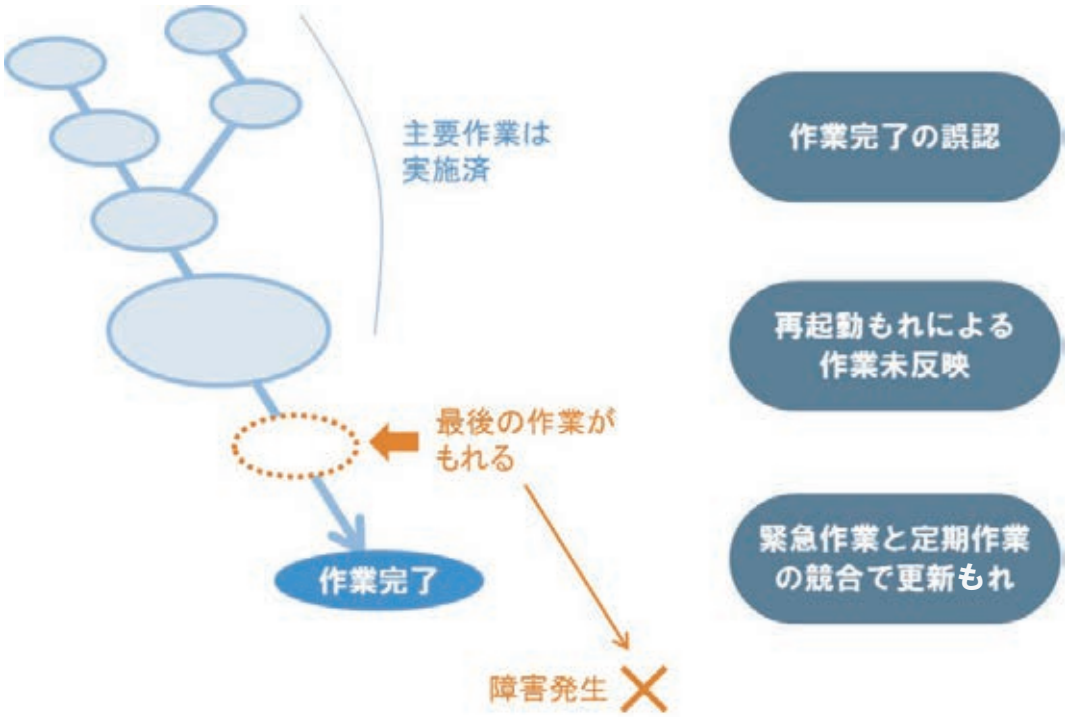


図 4.5.9－1 「不完全な作業実施」の分析

表 4.5.9－1 「不完全な作業実施」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
作業完了の誤認	G16	動作確認が完了したと誤認してプロシージャを強制終了してしまい、未完了の書込み機能が繰り返し起動してディスクを一杯にした
再起動もれによる作業未反映	1411	運賃切替のためのシステム更新時に、駅員が券売機1台の電源を切り忘れたため更新できず、切符の販売金額を誤った
緊急作業と定期作業の競合で更新もれ	T15	顧客データの定期修正時に、緊急作業更新前のデータを対象としたため、緊急作業結果が反映されなかった

4.5.10 作業中偶発事象への考慮不足

システムの一時的な運用作業の中で、偶発的に電力供給やハードウェアの故障が発生したことによって障害が発生している。

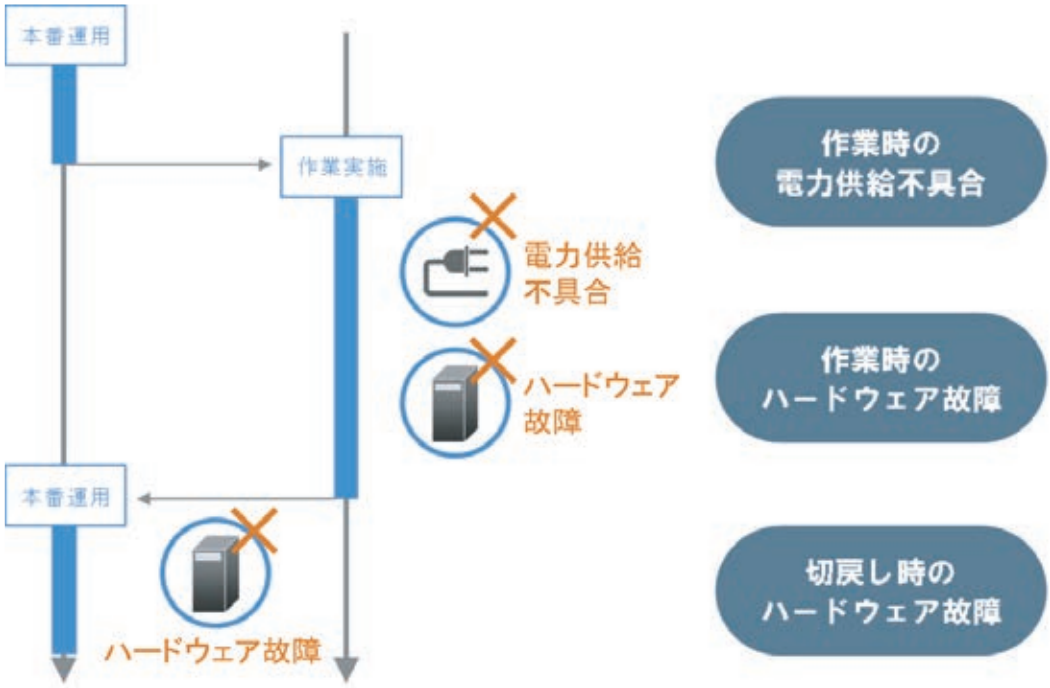


図 4.5.10－1 「作業中偶発事象への考慮不足」の分析

表 4.5.10－1 「作業中偶発事象への考慮不足」の事例

注意すべき観点（詳細）	事例番号	事例における障害発生内容
作業時の電力供給不具合	1305	電源設備の定期点検中に、システムへの電力供給に不具合が発生し、システムがダウンした
	1708	列車指令所内の電源装置のバッテリー交換時に不具合が発生し、運行管理システムへの電力供給が止まり、列車が約1時間運休した
作業時のハードウェア故障	G15	保守作業のため自動切替を解除した時にハードウェア故障が発生し、サービスが 10 分間停止
切戻し時のハードウェア故障	1314	新設備へのバージョンアップに失敗し、現行設備への切戻し中に新設備の片系でハード障害が発生し、残りの片系も過負荷でサービスがダウン

表 4.5.11－1 障害事例一覧表

「注意すべき観点」を中心とした障害事例の分類

・大分類は、JIS Q 20000-1:2012 に基づいて分類している。
・中分類以下については、個々の障害事例を再分析した上で、「注意すべき観点」として第三者が教訓を得るために有効と考えられる部分を中心に分類している。
・分析対象事例は、情報処理システム高信頼化教訓集にある54事例（G系:21事例、T系:33事例）、「情報システムの障害状況一覧」における8年半分（2010-2018年度前半）の事例の一部（詳細原因が理解できて教訓を得られるもの）である。
・事例番号のうちGまたはTで始まるものは、情報処理システム高信頼化教訓集、数字4桁は「情報システムの障害状況一覧」のものである。

【対象者】
特に注意すべき内容について、対象者ごとに強調表示している（◎の箇所）

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
新規サービスまたはサービス変更	不適切な要件定義	関係者の要件定義不参加	事業部門の要件定義不参加	G1	事業部門による要件確定が遅く、要件変更も多く、要件の設計への反映も正確に確認できていなかった	アプリケーション・オーナー制度による各事業部門の「態勢」の構築		◎	◎	◎			2.2.2 利害関係者の識別
			発注者の要件定義不参加	G2	注文処理の取消不可等、基本的な仕様に大きなもれがあることがシステム本稼働後に判明	要件定義と受入テストの発注者責任明確化、開発プロセス標準の見直し		◎	◎	◎			2.2.2 利害関係者の識別
			システム運用部門の要件定義不参加	G3	オペレータ操作に関する運用要件検討が不十分で、運用担当者の作業ミスが多発	運用者が要件定義に参加			◎	◎		◎	2.2.2 利害関係者の識別
	不適切な設計	計算処理の誤り	計算条件のもれ	1703	電力需要の計画と実績の過不足量（インバランス）算定時に、本来計算に加える必要がある値（域外分）が欠落。全国的な事業者の精算取引に影響した	プログラムの修正等	北海道電力託送業務システム（及び中部電力）			◎			2.4.4 ソフトウェア詳細設計プロセス
				1704	スマートメーター設置の顧客には振込用紙郵送を行わないという判定条件を漏らして設計し、該当顧客に振込用紙を重複送付してしまった	設計もれに対する社内組織間の役割分担明確化、マネジメント強化	中部電力料金請求システム			◎			2.4.4 ソフトウェア詳細設計プロセス
				1728	測定時に特定事象が発生すると、測定データが保存されないまま計算プログラムが進行し、一部の放射性廃棄物の放射能量が少なめに評価された	データ欠損等の発生時にエラー信号を発信し、測定を停止する機能の追加	原子力発電所 放射能測定プログラム			◎			2.4.4 ソフトウェア詳細設計プロセス
			通常外処理のもれ	T5	加算を主体とした業務処理（使用料計算）で減算処理が発生し、誤請求を行ってしまった	サービスの視点で見渡した変更管理				◎			2.4.2 ソフトウェア要件定義プロセス
			処理対象の時点誤り	1425	高額療養費は診療月時点での世帯単位で計算する必要があるが、診療月後に世帯変更があった世帯に対して変更後の世帯単位で計算してしまった	（対策については言及なし）	国民健康保険共同電算システム			◎			2.4.4 ソフトウェア詳細設計プロセス
				1727	あるインターチェンジを利用した自動車に対して、ETCのプログラムミスにより利用時刻が実際より1時間早く記録され、料金が誤請求された	（対策については言及なし）	西日本高速道路自動料金収受システム			◎			2.4.4 ソフトウェア詳細設計プロセス
			ありえないはずの条件の実在	1419	1,000件連続で「データなし」を処理終了とする仕様に対して実データで当該条件が発生したため、後段の送金処理が未完了となった	（対策については言及なし）	三菱東京UFJ銀行			◎			2.4.2 ソフトウェア要件定義プロセス
			変数名の誤記	1705	条件分岐処理において、区分を示す変数を入れるべき場所に名称を示す変数を入れてしまい、分岐が機能せず、臓器移植患者の待機日数計算を誤った	旧システムとの比較検証等	日本臓器移植ネットワーク患者検索システム			◎			2.4.5 ソフトウェア構築プロセス
		処理の不整合	システム連携時刻の不整合	T13	システム間の連携タイミングに15分間の差異があり、処理に矛盾が発生	システム全体でのウォークスルーレビュー				◎			2.4.4 ソフトウェア詳細設計プロセス
			キー項目の不整合	1527	匿名化のためキーとなるIDを暗号化したが、元データに全角、半角等が混在していたため暗号化後のIDでの突合ができなくなった	（対策については言及なし）	厚生労働省メタボ健診システム			◎			2.4.4 ソフトウェア詳細設計プロセス

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
新規サービスまたはサービス変更（続き）	不適切な設計（続き）	異常発生時の処理誤り	設計外処理の想定もれ	T3	列車出発後にも誤って制御信号が出続けるという想定外事象が発生し、後続列車が自動停止してしまった	設計された動きだけでなく、新しい動きを追加登録する「知識データベース」化				◎			2.4.2 ソフトウェア要件定義プロセス
			送信不能時のエラー表示誤り	1707	電子申告のシステムに予想以上のアクセスが集中した際に、受付が未完了であるにもかかわらず「送信完了」と表示されるケースがあり、事後対応に苦慮した	システムの負荷上限の拡大、利用者への注意喚起等	地方税電子化協議会 電子申告・納税システム			◎			2.3.3 システム方式設計プロセス
		検知条件の想定もれ	無操作異常の不検知	1431	運転士の一定時間無操作を検知する仕組みで、自動列車制御による減速を乗務員操作と誤って検知し、本来の異常検知を行えていなかった	（対策については言及なし）	JR東日本非常ブレーキ			◎			2.4.4 ソフトウェア詳細設計プロセス
			メッセージ重複による誤検知	T26	流用開発時に同一のジョブ完了メッセージを重複作成したため、ジョブ実行順序が変わりバッチ処理に論理矛盾が発生した	まるごとコピーではなく、一意性担保に注意				◎			2.4.4 ソフトウェア詳細設計プロセス
			抑制信号停止による誤検知	1231	発信用サーバに定期的に受信していた通行止め情報が途切れたため、復旧したと誤判断し、通行止め解除のメールが誤って自動送信された	（対策については言及なし）	中日本高速道路交通情報サイト			◎			2.3.3 システム方式設計プロセス
		回復時処理の想定もれ	通信障害からの回復失敗	1306	サーバ間通信に一時的な通信障害が発生した際に、通信が回復してもサービスを開始できないという不具合が顕在化	（対策については言及なし）	大阪証券取引所 売買システム			◎			2.6.3 修正の実施
		同時アクセスへの応答誤り	排他制御の考慮不足	T33	同一ファイルへの排他制御処理が実装されておらず、同時にファイルにアクセスした複数ユーザに不適切データを提供	レビュー／試験項目にリソース競合対策を明記				◎			2.4.7 ソフトウェア適格性確認テストプロセス
			キャッシュ制御の誤り	1725	新旧のCDNプロバイダのキャッシュ制御方法の違いに対処しなかったため、不適切な情報がキャッシュされて他人の個人情報が見えてしまう状態になった	外形監視、意図しないキャッシュを検知できる仕組み	メルカリ 個人間取引アプリ			◎	◎		2.3.5 システム結合プロセス
	不適切なテスト	テスト未実施	送信テストの未実施	1541	市民向けの緊急速報メールで、市民を対象とするため送信テストを実施していなかったが、1年半後に通信できないことが判明	（対策については言及なし）	福島市緊急速報メール			◎			2.4.7 ソフトウェア適格性確認テストプロセス
			負荷テストの未実施	1506	地震発生時のメールサービスが運用5年目に初めて実使用された時に、メールサーバに過負荷がかかりメール配信が遅延	（対策については言及なし）	徳島県牟岐町防災サービス			◎			2.4.7 ソフトウェア適格性確認テストプロセス
		テストによる副次作用	テストデータの本番残存	1405	改札機に対して消費税改定を想定したテストを実施後、データを戻す作業を怠り、利用者から料金を過剰に収受した	（対策については言及なし）	京成電鉄ICカードシステム			◎			2.4.7 ソフトウェア適格性確認テストプロセス
				1013	テストデータの削除を忘れてバッチ処理を実行したため、残高証明書の発行手数料を二重に引き落としてしまった	（対策については言及なし）	南都銀行システム			◎			2.4.7 ソフトウェア適格性確認テストプロセス
				1739	リハーサルのみに使用するプログラムが本番環境へ適用されてしまい、取引反映の処理に異常が発生	テスト実施時のチェック体制強化	東日本銀行			◎			2.4.7 ソフトウェア適格性確認テストプロセス
			テスト時のシステム日付の本番残存	1409	消費税改定を想定した切替で、1台の路線バスにのみ誤って日付を1日早くセットし、消費税改定1日前から増税後の運賃を徴収した	（対策については言及なし）	京急バス運賃システム			◎			2.4.7 ソフトウェア適格性確認テストプロセス
			テスト時のログ出力設定の残存	1215	システムテスト中にログを詳細に出力する設定としたまま、それを修正せずに本番運用を行い、カード発行処理が大幅に遅延	（対策については言及なし）	法務省入国管理局 在留カード等発行システム			◎			2.4.7 ソフトウェア適格性確認テストプロセス

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
新規サービスまたはサービス変更（続き）	不適切な移行	移行作業のミス	移行作業完了の誤認	16前別4	住民データの移行時にシステムが一時停止したが、「終了」と表示されたため作業完了と誤認。住民へのカード郵送が滞った	（対策については言及なし）	J-LISマイナンバーカード管理システム			◎			3.1.3 業務及びシステムの移行
			移行データのマッチング誤り	1502	データ移行時にミスがあり、住民票の「個人の住定日」を記載すべき位置に「世帯の住定日」を記載してしまった	（対策については言及なし）	大阪市住民基本台帳システム			◎			3.1.3 業務及びシステムの移行
サービス継続・可用性管理	不必要な待機系切替	意図しない待機系切替	自動切戻し機能の意図せぬ発動	1108	不具合発生時に自動で切戻す設定でのテスト実施中に、偶発ハード故障が発生。業務ピーク時に切戻し処理が発生し輻輳状態に陥った	システム切替判定処理の最適化等	NTTドコモ システム				◎		2.3.3 システム方式設計プロセス
			対応完了事象に対する切替再発動	1108	待機系切替判定ソフトをオフにした障害対応の後、オンにした判定ソフトが直近履歴を基に処理遅延中と判断し再度待機系に切り替えてしまった	切替判定ソフトが故障履歴を参照しないように修正	NTTドコモ システム				◎		2.3.3 システム方式設計プロセス
		障害メッセージの誤発出	正常稼働時の障害メッセージ誤発出	T1	ミドルウェア・OSの潜在バグで稼働系が障害との誤ったメッセージが出され、稼働系と待機系の両方が誤って稼働し、処理に不具合が発生	停止コマンドの送信等、フェールソフト（自系＆他系＆手動）の対策の追加					◎		2.4.4 ソフトウェア詳細設計プロセス
			軽微事象での障害メッセージ発出	1302	軽微な異常（通信ログを予備系にリアルタイムコピーする機能の遅延）を重大な異常と誤報してしまい、サービスの全面停止が発生	（対策については言及なし）	KDDI au LTEサービス				◎		2.4.4 ソフトウェア詳細設計プロセス
	待機系への切替失敗	障害メッセージの不発出	不完全停止によるメッセージ不発出	G4	メモリコントローラの障害時に、現用ノードが実質的には停止していたが不完全な停止状態であったため障害メッセージを発出できなかった	運用部門による主体的なシステム状態の確認					◎		3.1.4 システム運用
				T23	スイッチのキャッシュメモリ故障時に不完全停止となり障害メッセージが発生せず、障害を検知できなかった	複数観点からの監視機能追加					◎		2.3.3 システム方式設計プロセス
		待機系への設定もれ	待機系でのパラメータ設定誤り	T7	稼働系と待機系の同期を取るべきソフトウェアパラメータにもかかわらず、待機系への設定を怠り、待機系が起動しなかった	保守運用でのパラメータ確認、障害訓練					◎		2.6.3 修正の実施
			待機系でのファイル不足	G11	保守作業での設定ミスで待機系側に必要ファイルが存在せず、障害時に切替失敗	重要なシステムは、設定の作業指示書も優先的に確認					◎		2.6.3 修正の実施
			待機系でのパスワード設定もれ	1640	稼働系のみパスワードを変更し待機系のパスワードを変更せず、データ同期時にエラーが発生	（対策については言及なし）	横浜市住基ネット				◎		2.6.3 修正の実施

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
サービス継続・可用性管理（続き）	待機系への切替失敗（続き）	障害発生ケースの想定もれ	複数事象が同時発生するケースの想定もれ	T23	2つの監視機能（DB同期、自ノード監視）が偶然重複したため、待機系切替用の最後のDBサーバまでが停止	監視機能の組み合わせ確認とテスト実施					◎		2.4.4 ソフトウェア詳細設計プロセス
				T20	2つのtelnet接続（障害情報収集、自動監視）が競合し、無限ループが発生	″システム全体を俯瞰する鳥の目の対策機能追加時のリスク発見と共有″					◎		2.4.4 ソフトウェア詳細設計プロセス
			多数同時故障の想定もれ	1801	ハードディスクが2台故障しても自動復旧可能な仕組みにしていたものの、3台が同時故障したため、自動修復できず復旧に時間を要した	機器故障への監視強化、バックアップの強化、復旧手順の見直し	三菱UFJ NICOSカード				◎	◎	3.1.1 運用の準備
			故障時のネットワーク輻輳の想定もれ	T2	ハードディスクの故障で「リセット通知」が出続け、処理渋滞で一部の通信が途切れ、制御監視端末からの系切替えが行えなかった	停止制御による系切替を他装置からもできるようにした					◎		2.4.4 ソフトウェア詳細設計プロセス
			ケーブル切断時の想定もれ	T30	2重化配線されていたのにもかかわらず、両系のケーブルが束になっていたため同時に切断され、サービスが停止	施工標準の見直しと、両系とも障害になった場合の対策マニュアル作成					◎		2.3.3 システム方式設計プロセス
			故障時のエラーメッセージ発生の想定もれ	1007	ディスク装置故障によりシステムがダウンした際にエラーメッセージが大量に発生し、連携先のシステムもダウンした	（対策については言及なし）	ゆうちょ銀行システム				◎		2.3.3 システム方式設計プロセス
			サイレント障害（NW性能劣化の不検知）	T11	負荷分散装置でリクエストが廃棄されていたが、廃棄数がしきい値未満であったため検知できなかった	サービス監視条件の変更					◎		3.1.4 システム運用
		連鎖的障害への想定もれ	稼働不能機器の切り離し失敗	T10	ファームウェアの不具合でNASの故障ディスクを切り離せず、メッシュ構成であったため故障が波及して全サーバがダウンした	メッシュ構成を解除しグルーピング方式を採用					◎		2.3.3 システム方式設計プロセス
	縮退時の性能不足	縮退運転への検討不足	縮退時の一部サービス停止	T24	DBサーバ縮退時に性能が不足し、データ連携会社への情報提供等のサービスが停止	ディペンダビリティの確保					◎		2.3.3 システム方式設計プロセス
			縮退時の必要処理が遅延	1523	故障したサーバを切り離したが、未処理チェック機能の対象が想定以上のデータ量となり、長時間のサービス停止となった	（対策については言及なし）	外為どっとコム外貨ネクストネオ				◎		2.3.3 システム方式設計プロセス
	業務継続への準備不足	システム過信	代替業務の訓練不足	G9	システム障害発生時に業務窓口の処理が行えず、顧客に帰って頂く対応となった	障害規模に合わせた事務処理マニュアルの作成、訓練					◎		3.1.5 利用者教育
		作業影響の認識不足	後続処理影響の誤認識	1406	バッチ処理が期限内に終わらなかった場合の影響をシステム運用担当者が正しく認識しておらず、カード会員への請求が遅延	担当者の運用面の役割分担明確化	ビューカード基幹システム「VENUS」				◎		3.1.5 利用者教育
		障害対応手順の検討不足	障害発生の連絡が遅延	G17	重要なサービスにて障害が発生したが、障害発見者から運用担当者への連絡ルートが確立されておらず、障害対応の初動が遅れた	通常とは別の連絡系統を追加						◎	3.1.1 運用の準備
			要求に満たないサービス復旧時間	G18	障害発生時のサービス停止の影響が大きく、復旧時間の短縮が必要となった	第二本番系を構築						◎	3.1.1 運用の準備

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
容量・能力管理	大量処理への能力不足	想定を超える処理発生	突発的事象に起因する大量アクセスの発生	G12	突発的な事象により、処理能力を超えた注文が殺到し、サービス時間を短縮して対応を行った	業務部門がキャパシティ管理に責任を持ち、管理項目としきい値を設定		◎			◎		2.2.3 要件の識別
				1634	国勢情勢（米国大統領選）の変化を受け為替相場が大きく変動し、全商品の取引を一時停止	（対策については言及なし）	東京商品取引所	◎			◎		2.2.3 要件の識別
			アクセス数の予測失敗	1629	チケット購入希望者の短期間連続アクセスを想定できず、発売開始直後から想定の6倍を超えるアクセスが集中しシステムが停止	チケット販売期間を前半と後半に分ける等、処理の分散化	東京国際映画祭電子チケット販売システム	◎			◎		2.3.2 システム要件定義プロセス
				1631	同日から開始された新サービスにアクセスが集中した影響で、既存の切符購入等のサービスが利用しづらくなった	（対策については言及なし）	JR東日本モバイルSuica	◎			◎		2.3.2 システム要件定義プロセス
			アクセス影響先の検討不足	1432	携帯電話向けに緊急速報メールを配信したところ、メールから参照したWebサイトにアクセスが集中し、サーバがダウンした	暫定的には、Webサーバから容量の大きい地図データを削除	横浜市Webサイト	◎	◎	◎	◎		2.3.2 システム要件定義プロセス
		性能限界値の不整合	システム間での性能限界値不整合	T18	バッチ処理の上限を超えたデータ量を、オンラインで受け付けてしまった	制限値、制限値超過時の動作の確認			◎	◎	◎		2.3.3 システム方式設計プロセス
				G13	システム間で連携データ、連携時間の統一管理がなく、処理増加時にサービス時間を短縮	データディクショナリで定義を揃えた上で、データ連携内容を可視化			◎	◎	◎		2.3.3 システム方式設計プロセス
		性能限界値の設定不備	監視時間間隔が粗い	G14	処理件数を1分間隔で監視しており、秒単位での瞬間的な処理増大に対応できなかった	監視の時間間隔を含むキャパシティ計画の修正					◎	◎	2.3.3 システム方式設計プロセス
		要件変更時の性能考慮不足	コンテンツサイズの増大による性能劣化	T14	コンテンツ更新時にデータサイズが4倍になり、レスポンスが低下	業務部門作業をIT部門が確認することを一ルール化						◎	2.3.2 システム要件定義プロセス
	設定許容値の超過	しきい値超過	業務要件変更時のしきい値不変更	T4	予測可能時間を増大した際に予測対象列車数の上限値を変更せず、予測ダイヤを表示できなくなった	変化点の管理指標化				◎			2.3.2 システム要件定義プロセス
			周期処理時間への環境変化未反映	T32	周期処理の時間設定が機器やデータ増に追随できていなかったため、処理が時間内に完了せず機器動作停止および制御ネットワーク障害が発生	周期時間管理ルールおよび新機能追加時の運用ルールを制定						◎	3.1.4 システム運用
			しきい値の単位や定義の不一致	T29	システム間で転送されるデータの制限値について、当日分のみか、当日と翌日を含めたものかの理解が異なり、制限値を超えた結果システム障害を招いた	システム間の制限値の再点検と一元管理、機能仕様書の定義見直し				◎			2.3.2 システム要件定義プロセス
			しきい値超過の不検知	T22	スローダウンを契機に、重要性を認識していなかった予約処理のバッファが連鎖的に満杯となり、全体機能が停止	各バッファの蓄積状況監視、アラート設定				◎			2.3.2 システム要件定義プロセス

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
容量・能力管理（続き）	設定許容値の超過（続き）	無制限連続送信	試験信号の無制限連続送信	G10	作業誤りにより回線試験信号が送信され続け、3週間で共通バッファの空きがなくなり障害発生	しきい値超過後にアラートを表示するように改善				◎			2.6.3 修正の実施
			システムエラーの無制限連続送信	1628	磁気情報が消失したキャッシュカードの利用時に、プログラムミスでシステム内部でエラーが連続発生し、ATMの処理が停止	事前チェックで磁気情報消失の場合に取引を不成立とするようプログラムを修正	横浜銀行等勘定系システム「MEJAR」			◎			2.4.4 ソフトウェア詳細設計プロセス
		ログの肥大化	大量業務処理時のログ肥大化	1507	大規模マンションの住民の地番修正時に同マンションに入居する他住民もログに記録するため、ログ容量が超過し障害発生	ログ容量を削減する対策	埼玉県富士見市住民記録／国民健康保険システム			◎		◎	2.3.3 システム方式設計プロセス
			アクセス集中時のログ肥大化	1710	動画配信サービスでアクセス集中時にログが急増し、リソース不足により処理が滞留。配信予定のライブ中継映像が提供できなかった	（対策については言及なし）	DAZN動画配信サービス			◎		◎	2.3.3 システム方式設計プロセス
			監視強化によるログ肥大化	1611	滞留プロセスを重点監視した結果、ログが大量に記録され、ログデータ転送時にメモリ不足となりATMが停止	（対策については言及なし）	NTTデータ地銀共同センター			◎		◎	2.3.3 システム方式設計プロセス
事業関係管理	共同利用における問題	要件定義への責任希薄化	性能要件の見込み不足	G5	業界共同システムで、各社がピーク処理量を責任をもって予測せず、システムがダウン	利用各社による運営協議会を立上げ		◎	◎				2.3.2 システム要件定義プロセス
		障害発生時の対応制限	遠慮による再起動延期	G8	共同利用する他社への影響を考慮して機器の再起動ができず、障害が長期化	非常時対応を含めた利用者間の情報共有		◎	◎			◎	3.1.4 システム運用
	クラウド利用における問題	クラウドにおけるリソース管理の軽視	仮想サーバの割当ミス	T8	クラウドにサーバを集約したが、リソース管理が不十分で作業ミスを誘発した	仮想サーバのリソース管理、性能監視徹底					◎	◎	3.1.4 システム運用
		障害発生時の体制未整備	トラブル対応体制の未整備	G7	複数の利用者が同一のクラウドサービスを利用する環境下で、障害対応の事前取決めがなく、障害発生時に再起動等の対応判断ができずに復旧が長期化した	クラウドサービスにおいても役割、責任の明確化		◎			◎	◎	3.1.1 運用の準備
	システム利用者における問題	システム利用者の操作ミス	誤入力データの波及	G3	誤入力データが複数企業へ連携され、復旧に1週間を要した	コンティンジェンシープランの企業間共有				◎			3.1.5 利用者教育
			誤入力データの他者偶然一致	1328	誤入力したクレジットカード番号と有効期限が他者と偶然に一致し、誤った料金引き落としを行った。セキュリティコードは入力していなかった。	（対策については言及なし）	ヤフー公金支払システム			◎			2.4.4 ソフトウェア詳細設計プロセス
			データの未入力を原因とした判定誤り	1601	潮の満ち引きのデータを職員が入力せず、潮位変化を津波と誤認識して緊急速報メールを誤配信	（対策については言及なし）	和歌山県緊急速報メール			◎			3.1.5 利用者教育

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
供給者管理	製品仕様の理解不足	製品仕様の理解不足	わずかな仕様差異	T12	HDDからSSDに交換するとタイマ監視が200msで完了せず、制御装置が動作停止	ユーザによるベンダ側障害予防対策の確認、補完					◎		2.3.5 システム結合プロセス
		製品の制約事項	有効期限の超過	G21	サーバ証明書の有効期限を知らされないまま運用を開始し、期限切を迎えて通信が不通となりサービスが停止した	有効期限の台帳管理および定期的確認、開発委託先との引継ぎ事項チェックリストへの反映						◎	3.1.4 システム運用
		製品仕様の誤解	感覚と異なる設定値	T11	負荷分散装置のセッション数が設定値の1/4となる「仕様」のため、応答速度が低下した	サービス監視条件の変更					◎		2.3.3 システム方式設計プロセス
			隠れた設定値	1501	帳票作成用パッケージの仕様を把握しておらず、同時に実行できる印刷命令数の設定を誤り、証明書発行システムで障害発生	（対策については言及なし）	大阪市住民基本台帳システム			◎	◎		2.3.3 システム方式設計プロセス
			異常検知のみで機能停止	G11	ディスクモジュールの自己診断機能で、異常検知のみで機能停止する仕様となっていた	異常検知後の機能停止条件を見直した制御プログラムを適用					◎		2.3.3 システム方式設計プロセス
		製品の設定不備	データベースサーバの設定不備	T19	データベースサーバでSQLの実行計画を自動最適化で設定したところ、却って性能が悪化した	自動最適化を使用しない等、運用を見直し				◎	◎		2.3.3 システム方式設計プロセス
				1301	データベースサーバでメモリ割当処理を定義するパラメータの設定を誤り、メモリ割当・解放の処理がCPUに過剰負荷を与え障害が発生した	（対策については言及なし）	KDDI au ID認証決済システム			◎	◎		2.3.3 システム方式設計プロセス
	製品の不適切な利用	長期連続運転	負荷分散装置の長期連続運転	T17	負荷分散装置を8か月連続運転し、メモリ不足エラーで停止	定期的な再起動実施					◎	◎	3.1.4 システム運用
			ネットワークスイッチの長期連続運転	T25	スイッチの故障、コマンドによる再起動でなく物理的再起動で復旧。	障害原因切り分け基準の決定、ログの確実取得					◎	◎	3.1.4 システム運用
		不適切なパッチ適用	遅すぎるパッチ適用	T16	負荷分散装置の既知の障害が、修正パッチ適用前に顕在化	修正パッチ等の確認サイクルの早期化						◎	2.6.4 保守レビュー及び／又は受入れ
			拙速なパッチ適用	1616	開発元提供のパッチ内にキャッシュ排他制御の変更があり、もともとのディスク排他制御との間でデッドロックが発生	（対策については言及なし）	日本航空重量管理システム					◎	2.6.4 保守レビュー及び／又は受入れ
			開発元のテストに対するレビュー不足	T28	パッケージソフトウェアの修正（アップデート）に伴って混入したバグの影響により、システム停止が発生	開発元の全修正箇所に対するレビューおよび回帰テストの実施				◎			2.4.7 ソフトウェア適格性確認テストプロセス
	適切でない製品供給者管理	適切でない製品サポート	機動的でない製品サポート体制	T27	障害対応に必要な知識を持つ製品開発チームメンバへの連絡体制ができておらず、トラブル発生から回復までに多大な時間を要した（特に海外サポート拠点との時差がある場合に注意が必要）	製品開発チームメンバへの24時間サポート体制を構築						◎	3.1.1 運用の準備

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
インシデント管理	適切でない不具合管理	不具合の長期放置	機能誤りの長期放置	1641	特定条件下における保険料計算の誤りを5年前に認識したが、問合せへの個別対応のみで全体対応が遅延した	複数の担当者での確認徹底	後期高齢者医療制度保険料計算システム	◎				◎	3.1.7 システム運用の評価
			機能不足の長期放置	16前別2	キャッシングサービスの利息日割り計算機能がなく手作業で請求を行い、10年間で2,401件の過剰請求が発生した	（対策については言及なし）	イオン銀行イオンカードキャッシングサービス	◎				◎	3.1.7 システム運用の評価
		不具合発生時の対応遅れ	不具合の認識遅れ	G10	一部の電話コール接続異常を通信回線事業者の問題と誤認し、障害対応の初動が遅れた	連携先からの障害問合せには、自システムの問題を優先して調査						◎	3.1.7 システム運用の評価
				G17	一部の通信の断絶が発生したが、他の通信は正常であり、また装置本体からのアラートも発生しなかったため、障害発生の認知が遅れた	他の周辺機材等を活用した代替アラートの仕組みを追加						◎	3.1.7 システム運用の評価
問題管理	未想定の障害発生時の混乱	想定不可能な障害	製造元が未認識の不具合発生	T9	タイマ制御処理ソフトのバグでTCP切断時のタイムアウトを検知できず、処理が停滞	障害箇所を切り離しての業務継続性確保						◎	3.1.4 システム運用
			原因不明の不具合発生	T25	スイッチの故障。コマンドによる再起動でなく物理的再起動で復旧。	障害原因切り分け基準の決定、ログの確実取得						◎	3.1.4 システム運用
構成管理	システム環境管理	システム環境間の差異	本番環境とテスト環境の差異	T6	本番環境のみに存在するデータベース・オプションのバグが顕在化	環境差異分析に基づくリスク対策						◎	2.4.7 ソフトウェア適格性確認テストプロセス 5.5.2 構成管理の実行
	施設管理	室温上昇	室温上昇によるサーバ自動停止	1522	施設で小規模火災が起き、火災報知器が作動して空調が停止。この影響で室温が上昇しメールサーバが自動停止	（対策については言及なし）	auキャリアメール				◎	◎	3.1.4 システム運用
			室温上昇によるネットワークスイッチ異常	1536	サーバルームの空調故障で、ネットワークスイッチに異常が発生し空港の各システムが停止	（対策については言及なし）	中部国際空港空港内システム				◎	◎	3.1.4 システム運用
		過電流対策	配電盤の定格遮断電流設定ミス	1120	配電盤におけるブレーカーの定格遮断電流が許容値よりも小さく設定されており、過電流でない状況でブレーカーが落ち通信機能が停止	ブレーカーの定格遮断電流設定の修正	気象情報伝送処理システム				◎	◎	2.3.3 システム方式設計プロセス

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
変更・リリース管理	不適切な変更管理ルール	変更管理ルールの形骸化	一律的なチェックの弊害	G11	保守作業チェックや修正プログラムの適用がシステムによらず一律となっており、重要なミスを見落とし	システムの重要度に応じて、運用・保守の体制・作業の濃淡を						◎	3.3.4 統合的制御管理
	システム担当者の作業ミス	作業準備のもれ	作業時の本番環境分離不足	1304	ネットワーク工事において、本番環境と接続したまま工事を行ったので、不要なデータが送信されシステム異常が発生	（対策については言及なし）	原子力安全基盤機構緊急時対策支援システム					◎	5.5.2 構成管理の実行
		不完全な作業実施	作業完了の誤認	G16	動作確認が完了したと誤認してプロシージャを強制終了してしまい、未完了の書込み機能が繰り返し起動してディスクを一杯にした	本番環境に適用するものの動作環境の確認徹底						◎	2.6.3 修正の実施
			再起動もれによる作業未反映	1411	運賃切替のためのシステム更新時に、駅員が券売機1台の電源を切り忘れたため更新できず、切符の販売金額を誤った	（対策については言及なし）	大阪市営地下鉄券売機					◎	2.6.3 修正の実施
			緊急作業と定期作業の競合で更新もれ	T15	顧客データの定期修正時に、緊急作業更新前のデータを対象としたため、緊急作業結果が反映されなかった	影響データの把握、後続業務でのデータ同期						◎	2.6.3 修正の実施
		作業によるデータ不整合	データの二重登録	1617	操作ミスで顧客の二重登録やファイルの二重作成を行い、本来の使用料の2倍で料金を請求するケースが生じた	（対策については言及なし）	四国電力・検針・電気料金関連システム					◎	3.1.4 システム運用
				1609	くじの発券端末機を入れ替えた際に、くじの通し番号に重複が発生し処理結果が不整合	（対策については言及なし）	みずほ銀行宝くじシステム					◎	2.6.3 修正の実施
			データの不完全なコピー	1314	機能改修作業において、現行設備のユーザ情報を新設備にコピーする作業コマンドに誤りがあり、情報が不一致となりサービスが停止	（対策については言及なし）	KDDI au 携帯電話サービス					◎	2.6.3 修正の実施
			作業対象の取り違え	1216	B群サーバに誤ってA群サーバ用のソフトウェアを適用したため、B群側ユーザがA群ユーザの情報を参照・更新可能となってしまった	（対策については言及なし）	NTTドコモ 携帯電話サービス					◎	2.6.3 修正の実施
		作業内容の誤り	誤解を生む作業指示	T21	ゲートウェイ設定の指示書でパラメータをローマ字で記載しており、これを誤読した結果、電話コールが異なる拠点へ転送された	ミスが発生する状況、環境の改善						◎	3.1.4 システム運用
				T31	障害対策マニュアルの記述が曖昧だったため不要な機能を実行してしまい、復旧の遅延を招いた	PDCAサイクルで障害対策マニュアルを維持管理する運用ルールの制定						◎	3.1.4 システム運用
			不注意によるデータ削除	1213	勘定系システムの端末テーブル修正時に、誤って本来変更する必要のないATM37台分の内容をクリアしてしまい、翌日にATM停止が多発	（対策については言及なし）	山陰合同銀行					◎	2.6.3 修正の実施
			不注意による設定誤り	1748	料金システムの設定を間違え、平常料金とすべきと日の駐車料金を、繁忙期の料金で請求した	作業立ち合い、設定結果のダブルチェック	関西エアポート駐車料金システム					◎	2.6.3 修正の実施

大分類	注意すべき観点			事例番号	事例における障害発生内容	主要な対策	【参考】企業等名称（報道事例のみ）	発注者	PM	アプリ	インフラ	運用	共通フレーム該当箇所
	中分類	小分類	詳細分類										
変更・リリース管理（続き）	システム担当者の作業ミス（続き）	担当者の独断作業	保守担当者の独断作業	G16	保守担当が、運用担当の了解なく本番環境へリリースを行い、全サーバが停止	保守担当へのログインIDの都度払い出し						◎	2.6.3 修正の実施
			ミスの上塗り	G6	初期ミスを回復するために手順を逸脱した操作を行い、全データを消去してしまった	組織としての判断基準、作業規定の整備						◎	2.6.3 修正の実施
		作業中偶発事象への考慮不足	作業時の電力供給不具合	1305	電源設備の定期点検中に、システムへの電力供給に不具合が発生し、システムがダウンした	（対策については言及なし）	共同通信社記事編集・配信システム					◎	2.6.3 修正の実施
				1708	列車指令所内の電源装置のバッテリー交換時に不具合が発生し、運行管理システムへの電力供給が止まり、列車が約1時間運休した	（対策については言及なし）	JR九州運行管理システム					◎	2.6.3 修正の実施
			作業時のハードウェア故障	G15	保守作業のため自動切替を解除した時にハードウェア故障が発生し、サービスが10分間停止	保守作業中に自動切替を解除しない前提で保守作業マニュアルを修正						◎	2.6.3 修正の実施
			切戻し時のハードウェア故障	1314	新設備へのバージョンアップに失敗し、現行設備への切戻し中に新設備の片系でハード障害が発生し、残りの片系も過負荷でサービスがダウン	（対策については言及なし）	KDDI au 携帯電話サービス					◎	2.6.3 修正の実施
	組織内のコミュニケーションミス	組織内の情報共有不足	特定担当者しか知らない仕様	G19	制御装置が制限値を持っていることがチーム全体に共有されておらず、障害原因を見誤り復旧の遅延増を招いた	原因、対策を「教訓」としてまとめ、チーム内で継続的に唱和						◎	3.1.4 システム運用
	組織間のコミュニケーションミス	組織間の連携不足	担当者の気づきの不連携	G4	運用担当者が機器の切替失敗に気づいていたが、SEに伝達されなかった	状況判断できる社員への連絡体制強化						◎	3.1.4 システム運用
			作業実施情報の不連携	1434	システムを管理する関係会社間でメンテナンス実施情報が共有されず、利用者へ通告がないまま切符予約等のサービスが停止	（対策については言及なし）	JR東日本 えきねっとモバイル Suica	◎				◎	3.1.4 システム運用

5

おわりに
（障害事例に学ぶ教訓の共有）

IT サービスを担う情報処理システムの障害事例関連情報を収集し、6 年間にわたって IPA 内に設置した部会における、多様な分野の有識者・専門家による分析や対策手法の整理・体系化を通して得られた「教訓」を取りまとめた。本教訓集が様々な業界・分野で幅広く活用され、また、世代を越えて伝承され、類似障害の再発防止や影響範囲縮小に結びつくことを期待する。

(1) 教訓化の奨め

障害発生時には、その復旧対応が一段落した頃に、なぜなぜ分析などの手法を用いて、直接原因から根本原因にまでさかのぼるような分析は、多くの組織で行われているものと思われる。その結果は、「教訓」の形にまで抽象化・普遍化し、ぜひとも『広く長く使える』智恵として蓄積していただきたい。本書がその参考となれば、喜ばしい限りである。

(2) 共有の奨め

本書に掲載した教訓、あるいは各組織でまとめられた教訓は非常に有用なものである。しかし、その特徴故に、活用可否の判断や活用方法の検討に際して次のような課題がある。

(a) 実際の事例に基づいて作成されているが、機密保持の理由から、情報提供元を特定できるような内容を取り除いて一般化・抽象化されている。
課題 a：事例の具体的詳細までは記載されていない。そのため、活用可否の判断や活用方法の検討がしにくい。

(b) トラブル事例から学べることの重要な一部を切り取り、それに焦点を当てて教訓化されている。
課題 b：他に原因や再発防止策を深堀りすべき事項があっても、それは十分には記載されていないことがある。

(c) 汎用的かつ普遍性のある障害事例情報を広く収集して教訓化している。
課題 c：関心のある特定の業務・業種に特有の障害事例がカバーされていないことがある。

これらの課題を解決し、貴重な経験である障害事例をより詳細かつ具体的な内容で入手し、より多面的かつ高度な再発防止策を構築するために、「共有」が有効である。

共有の手段の一つは、同一業種、業務など同じ特性を持つ IT システム / サービスを開発 / 運用する組織の関係者が集まって障害情報の共有グループを形成することである。そこでは、実際に発生した障害情報と直接の原因、発生時対策、根本原因の分析結果とそれに基づく再発防止策などのより具体的な内容を、そのグループ内に限定して（場合によっては生の情報のまま）情報交換する。このようなグループの構築により、各参加者は生の事例からより具体的な対策を検討することができる。コンテキストの同じ関係者が集まる同分野・業界におけるグループでは、より精緻な議論が行われるものと期待される。

願わくは、共有グループで作成された「教訓」が、さらに社会で広く共有されて欲しいものである。

(3) 今後の期待

今後、様々な業界・分野で教訓の共有が行われ、また、世代を越えて教訓が伝承され、結果として、社会全体の IT サービスの信頼性が高まることを期待する。そのような社会の実現に向けて、IPA としても、情報共有に関する支援など、引き続き貢献していきたい。

参考文献

1章

（文献 1－1） 松田、目黒「情報システムの障害状況 2017 年後半データ」、SEC ジャーナル 52 号 IPA、2018 年 3 月
https://www.ipa.go.jp/files/000064398.pdf

（文献 1－2） 経産省、IPA、JUAS 「重要インフラ情報システム信頼性研究会報告書」、IPA、2009 年 3 月
https://www.ipa.go.jp/sec/softwareengineering/reports/20090409.html

（文献 1－3） IPA、JUAS「障害を発生させない、被害を拡大させないための対策ガイド」、JUAS, 2009 年 6 月 , 他

（文献 1－4） IPA「教訓活用ガイドブック（IT サービス編）」
https://www.ipa.go.jp/files/000051041.pdf

2章

（文献 2.2－1） IPA/SEC「共通フレーム 2013」、IPA、2013 年 3 月
https://www.ipa.go.jp/sec/publish/tn12-006.html

（文献 2.2－2） IPA/SEC「ソフトウェア開発データ白書 2018-2019」、IPA、2018 年 10 月
https://www.ipa.go.jp/sec/publish/tn12-002.html

4章

（文献 4.1－1） ITSMS ユーザーズガイド　～導入のための基礎～
JIPDEC 一般財団法人日本情報経済社会推進協会、2013 年 11 月 20 日
https://isms.jp/itsms/doc/JIP-ITSMS112-21.pdf

（文献 4.2－1） 江見　明弘 「重大システム障害の背景と対策」、NTT データ、2012 年 3 月 27 日

（文献 4.2－2） IPA/SEC「高信頼化ソフトウェアのための開発手法ガイドブック」、IPA、2011 年 3 月
https://www.ipa.go.jp/sec/softwareengineering/reports/20100915.html

（文献 4.2－3） IPA/SEC「実務に活かす IT 化の原理原則 17 ヶ条」、IPA、2010 年 10 月
https://www.ipa.go.jp/sec/publish/tn10-001.html

（文献 4.3－1） 河野龍太郎「医療におけるヒューマンエラー第 2 版」医学書院　2014 年

※ URL は、変更されたり、削除されたりしている場合があります。

謝辞

本教訓集の作成にあたり、情報提供と適切な助言・示唆を含む積極的な議論にご尽力頂いた中村英夫主査（日本大学名誉教授）をはじめとする部会の方々及び事例の提供にご協力いただいた各企業・団体等に深く謝意を表します。

IPA 関係者一同

【部会】　　2019 年 2 月現在　（敬称略）

主査	中村　英夫	日本大学
委員	安達　芳則	株式会社フジテレビジョン
	今井　元一	株式会社オリジネィション
	気賀　健司	日本生命保険相互会社
	北田　　啓	株式会社三菱ＵＦＪ銀行
	五味　　弘	一般社団法人電子情報技術産業協会
	坂本　　忍	株式会社証券保管振替機構
	関　　邦夫	東京海上日動火災保険株式会社
	高木　徳生	オムロンソーシアルソリューションズ株式会社
	築嶋　健輔	KDDI 株式会社
	鍋田　芳則	三菱電機株式会社
	長谷川 和人	内閣官房情報通信技術（IT）総合戦略室
	松本　雅行	松本信号システムコンサルタント
オブザーバ	高橋　　聖	日本大学
	宮下　　清	一般社団法人 日本情報システム・ユーザー協会
事務局	山下　博之	独立行政法人情報処理推進機構
	目黒　達生	独立行政法人情報処理推進機構
	村岡　恭昭	独立行政法人情報処理推進機構
	齋藤　　毅	独立行政法人情報処理推進機構

上記以外にも、2013 年以降に委員としてご協力いただき、その後交代された方々に深謝します。

【ご協力いただいた企業・団体等】（五十音順）
ITA（Information Technology Alliance）
TFOS.SG（Total Flight Operation System Study Group）
小田急電鉄株式会社
九州旅客鉄道株式会社
株式会社 JR 東日本情報システム
全日本空輸株式会社
国立大学法人電気通信大学
電気事業連合会
電力広域的運営推進機関
東京電力ホールディングス株式会社
東京都世田谷区
東京都中野区
日本航空株式会社
東日本旅客鉄道株式会社

また、株式会社安全推進研究所　所長（自治医科大学名誉教授）　河野龍太郎氏から頂いたヒューマンエラーの分析における多大な御教示にも深く謝意を表します。

情報処理システム高信頼化 教訓集
IT サービス編

2019 年 2 月 28 日　1 版 1 刷発行

監修者　独立行政法人情報処理推進機構 (IPA)
社会基盤センター

発行人　片岡　晃

発行所　独立行政法人情報処理推進機構 (IPA)
〒113-6591
東京都文京区本駒込二丁目 28 番 8 号
文京グリーンコート センターオフィス
https://www.ipa.go.jp/ikc/

ISBN 978-4-905318-68-2　Printed in Japan

ISBN978-4-905318-68-2
C3055 ¥2407E

定価　本体2,407円【税別】

IPA Better Life with IT
独立行政法人 情報処理推進機構
社会基盤センター