オープン系システムを対象とした障害の原因調査手法と対策立案に関する研究

平成３１年 １月

日本大学大学院理工学研究科博士後期課程

情報科学専攻

D3001 篠原昭夫

目次

第１章　　序論

1.1　本論文の背景

インターネットの普及とともに急速に導入が進んだオープン系システムは，複数ベンダから提供された製品で構成されている。それまでのシステムは少数ベンダから提供された製品で構成されていることが一般的であった。代表的なものに IBM System/360[1] がある。システム構成品が少数ベンダ製であることは，障害原因を統合的に調査することを容易にしていた反面，システムの多様化を困難にする要因となっていた。また製品が高価であると共に，新機能の追加に時間を要するなどの問題があった。これらの製品群はメインフレーム系(汎用機)と呼ばれ，1960 年代から 90 年代までシステムの主流であった。システムのユーザは，高額な導入費用に加えて，運用，稼働維持費用も負担し続ける必要があった。また，製品数の少なさからシステム規模の小型化も困難で，電子化された情報によって業務を遂行するユーザは一定の規模より大きな組織であることが条件となっていた。

一方で，特定用途向け機器での使用を前提として機能を大幅に制限した製品は，1971 年の Intel 4004 マイクロプロセッサ[2]に代表されるように，メインフレーム系の製品とは比較にならないほど安価であった。このことは電子化された情報をより多くのユーザが利用する引き金となった。しかし特定用途向けの製品はプログラミングによる機能変更を考慮しておらず，情報処理システムを構成できるまでには至らなかった。このような状況は 8bit マイクロプロセッサ[3]の出現で変化する。8bit 製品はパーソナルコンピュータ(PC)に搭載されて広く普及し，一般家庭レベルにおいてもプログラミングが行える環境が形成されていった。

小規模システムでメインフレーム系と同様に無停止での稼働を前提とした製品は，UNIX[4][5]の出現で可能となった。また，UNIX は機器同士をネットワークで接続する機能を早くから実装したことが特徴で，インターネットが広く普及するための原動力となった。UNIX の普及と同じ時期に小規模システム向けのハードウェア，ソフトウェアを開発するベンダ数は増加しはじめ，現在のマルチ・ベンダ市場が形成されてゆく。マルチ・ベンダ市場では１社で広範な製品を開発することは多くなく，ほとんどのベンダが少数の製品を提供している。

これらのベンダが提供する製品は，他ベンダで開発された製品とともに使用されることが前提のため通信規約などの規格に準拠した設計となっている。一方で規格と関係し

ない内部構造は，各ベンダ独自の設計方針で開発されている。このことは新技術の導入を容易にして製品の進歩を促している。例えば，PC 用の CPU とオペレーティング・システム(OS)は別ベンダが開発する。しかし互いの接続仕様が統一されているため，次世代の CPU が先に製品化されても OS を変更することなく使用し続けることが可能で，双方が同期して世代を更新する必要はない。

しかしマルチ・ベンダ環境は，障害発生時の原因特定を困難にする要因となる。何故なら，システム障害の調査では全体を統制することが必要となるためである。汎用機で構成されたシステムの時代にはなかった問題である。さらに Dog Year という表現に代表されるように各製品の開発・販売期間は短く，事例報告の無い未知障害の発生件数が減少しにくい状況が生じることとなった。同一バージョン製品の販売期間が 2 年を超えることはほとんどなく，12 ケ月から 18 ケ月程度が一般的である。

複数ベンダから提供された製品で構築されたシステムは，オープン系システムと呼称される。オープン系システムでは，障害の原因部位特定の殆どは人手で行われる。何故なら，製品寿命の短さから既知の障害事例参照が困難で，未知障害の発生率が高いためである。しかしながら，オープン系システムに特化した人手による調査手法の研究は緩慢であるのが実状である。

1.2 本論文の目的

システム障害の原因部位を人手により特定する手法は過去にも存在する[6]。しかしオープン系システムに適した方法とは言えない。何故なら，前節に述べたオープン系システムの特徴が考慮された手法ではないためである。オープン系システムでは，障害原因調査をユーザが主体となって実施する。このことは，それまでのメインフレーム系システムの障害調査と著しく異なる点である。さらにユーザは，ベンダが製品の詳細情報を公開しない条件下で障害原因を究明しなくてはならない。そこで本論文は，マルチ・ベンダ製品で構成されていて，かつ多様な形態を持つオープン系システムで発生する障害の原因部位を特定することを主眼とした新たな手法を確立することを目的とする。具体的な要件は次の通りである。

・複数ベンダから提供される製品を統一された手法で調査可能なこと

・多様なオープン系システムに適用可能であること

・オープン系システム向け製品の特徴が反映された手法であること

・製品の詳細仕様が不明な条件下でも障害原因部位を特定可能なこと

・製品開発中よりも原因究明のための情報が不足していても調査可能なこと

・各システムの特徴を考慮した調査が行えること

現在のオープン系システムでは，相当に大きな規模を持つものも存在する。しかし本論文では，障害発生件数が多くかつ人手で原因調査を実施することが多い中・小規模システムを対象とする。

以上を踏まえて本論文で提案する障害原因部位調査手法は，上に挙げたオープン系システム特有の条件に適合できるよう工夫されている。提案する手法は，実際に障害原因部位の調査を人手で実施する過程が反映されており，実用的なものである。

1.3　本論文の構成

本論文の構成は７章より成る。

第１章　序論

本論文の背景と目的，論文構成を述べる。本論文で研究対象とする，オープン系システム障害原因部位調査に関する条件を示し，障害原因部位調査手法に求められている要件について述べる。

第２章　オープン系システム運用の現状

本章ではオープン系システム運用の現状分析を行う。はじめにシステム構築から運用に至るまでの流れ，加えて障害発生時に実際に行われる対応を説明する。ここでは，オープン系システムで発生した障害原因の調査がどのような条件下で行われているかを知ることで，障害原因調査に必要な要因を明らかにする。

次にシステムを構成する個々の製品が，障害に対してどのような方針で開発されているかを述べ，障害原因調査と対策立案に関する条件を説明する。

最後に，製品開発の手法および障害発生に対するベンダの方針について説明する。特に本論文の目的に大きく影響する，複数ベンダによる製品提供の実状とエラー情報

ロギング機能については詳細を説明する。これらに加えてオープン系システムで発生する障害とその対策についての分析を行い，障害発生傾向と修正プログラムおよび機能拡張との関係を説明する。さらに，修正プログラムの提供実態の分析も行う。

次章では，これらの分析結果からオープン系システムに求められている障害原因調査方法を明らかにし，機能線による調査手法を提案する。

第３章　機能線による障害原因調査手法の提案

本章では人手によって障害原因調査が必要である理由と，実際の原因調査を実行する場面で用いられている手法の共通点を抽出する。抽出した結果から，人手で実行されている障害調査手法のモデル化を行う。モデル化した調査手法から機能線を提案し，その詳細を説明する。さらに，機能線と併せて用いることで調査対象を削減する手法として補助機能線を提案する。

次章では，機能線上のどの部位で障害が発生したかを判定する手法として比較法を提案する。

第４章　コンポーネントでの障害発生有無判定

前章で提案した機能線を用いて障害原因部位を特定するには，各コンポーネントで障害が発生したかを判定する必要がある。そこで本章では，コンポーネント内での障害発生有無判定手法として比較法を提案する。さらに，比較法の実行手順および留意点について述べる。

第５章　機能線による障害原因調査の効果および FTA との比較

本章では，機能線を用いた障害原因調査手法の効果を検証する。検証は，実際の障害事例を記録した障害コールセンターのデータベースを参照し，これに機能線を適用した場合の効果を分析することで行う。さらに，機能線の他に人手により障害原因部位を特定する手法である FTA(Fault Tree Analysis)との論理的な比較を行う。

第６章　機能線を用いた障害分類と対策立案

提案した機能線では，障害の概要が視覚的に記述することができる。そこで，この性質を障害分類に応用する方法について述べる。障害を分類することで，障害型と対策法を関連付けることが可能となる。この結果，障害発生から対策立案，結果判定までの流れを一元化することができる。

さらに本章では，障害対策実行時に障壁となるDOAとRegressionについての分析を行う。こにより，二次障害の発生を抑止またはその検出を容易にする対策方法を立案することを可能とする。

第７章　結論

結論を述べ，本論文の成果をまとめる。

〔参考文献〕

[1] IBM system360 概要:

http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/system360, (2018-11 閲覧)

[2] Intel 4004 概要:

http://www.st.rim.or.jp/~nkomatsu/intel4bit/i4004.html, (2018-11 閲覧)

[3] Motorola 6809 概要:

http://www.6809.net/6809/?6809%A4%CB%A4%C4%A4%A4%A4%C6, (2018-11 閲覧)

[4] UNIX 概要:

Ritchie, D.M.; Thompson, K., “The UNIX Time-Sharing System” (The Bell System Technical Journal), Vol. 57, No. 6, Part 2, July-August 1978

[5] “A History of UNIX before Berkeley”: UNIX Evolution: 1975-1984

http://www.darwinsys.com/history/hist.html

[6] Larsen, Waldemar. “Fault Tree Analysis”. Picatinny Arsenal. Technical Report 4556, (January 1974), http://www.dtic.mil/dtic/tr/fulltext/u2/774843.pdf, Retrieved 2014-05-17

第２章　　オープン系システム運用の現状

2.1　はじめに

現在稼働しているオープン系システムの総数は，その実態を把握することが困難なほどに多い。統計によれば，2011 年時点の国内サーバ稼働台数(Install Base)は 242 万台である[1]。世界の統計では，2012 年時点でおおよそ 4500 万台のサーバが稼働しているとしている[2]。１つのシステムで使用されている平均サーバ数が不明であるため，これらの数値からシステムの総数を知ることは困難であるが，これを 10 台と仮定しても全世界で約 450 万のシステムが運用中であると推定される。膨大な数のシステム全てが問題なく稼働することは難しく，日々発生する障害の件数は相当なものになることは容易に想像できる。

一方でオープン系システムを構成する製品に注目すると，ハードウェア，ソフトウェアの区別なく共通の開発方針が見える。例えば，新技術をいち早く実装するために開発試験が必ずしも十分とは言えない製品が市場に投入される。また，障害調査時に必要となるログ情報収集機能は実装の優先度が低く，製品発売後に段階的に仕様の拡張と変更を加えながら拡充されていく。このため新製品では調査情報が入手困難なことが多い。さらにログ機能の仕様は非公開が一般的であり，この情報を参照するだけでは障害原因を特定することが難しい。このような背景から，オープン系システムでは未知障害の発生率が低下せず，人手で原因調査を行う件数が比例して大きくなっている。

以上に加えて，製品ベンダが自社製品に対してのみ障害原因調査を実施することも見逃せない問題となっている。システムユーザは，各ベンダからの調査結果を自身で統合し，システムのどの部位で障害が発生したかを判断する必要がある。これは，少数ベンダの製品でシステムが構成されていたメインフレーム系の時代にはなかった問題で，オープン系システムの特徴である。そこで本論文では，多様な形態を持ちかつ障害発生件数が多く人手で調査が行われている中・小規模システムを対象とする。

本章ではこのようなオープン系システムについての分析を行い，システム運用に与える影響について考察する。

2.2　システム構築から運用まで

オープン系システムの平均的な運用寿命は数年である。これは個々の製品寿命が短いことが影響している。ベンダのサポート打ち切りがシステムの運用寿命に与える制約は大きく，同一製品で 10 年を超えてサポートを受けられることは稀である。また，近年ではセキュリティ要件からシステムの更新を行わざるを得ない状況が発生することもある。例えば，広範囲に使用されたオペレーティングシステムである WindowsXP のサポート終了理由の一つは，セキュリティ脆弱性への対応が困難になったことが挙げられる。

このような背景から，システム構築，更新は数年毎に行われることが多い。システム構築の流れは以下が一般的である。

・システム要件定義・設計，実現可能性確認

・構築，導入作業

・運用試験，検査

・運用維持，障害対応

・システム変更（実施されない場合もある）

図 2.1 はシステムの構築から運用開始までの流れを示したものである。システムの規模が大きくなれば，これに従事するベンダ数も増加し役割は細分化される。図 2.1 ではシステム規模に応じて各 Step に従事するベンダの数が異なる様子を示している。

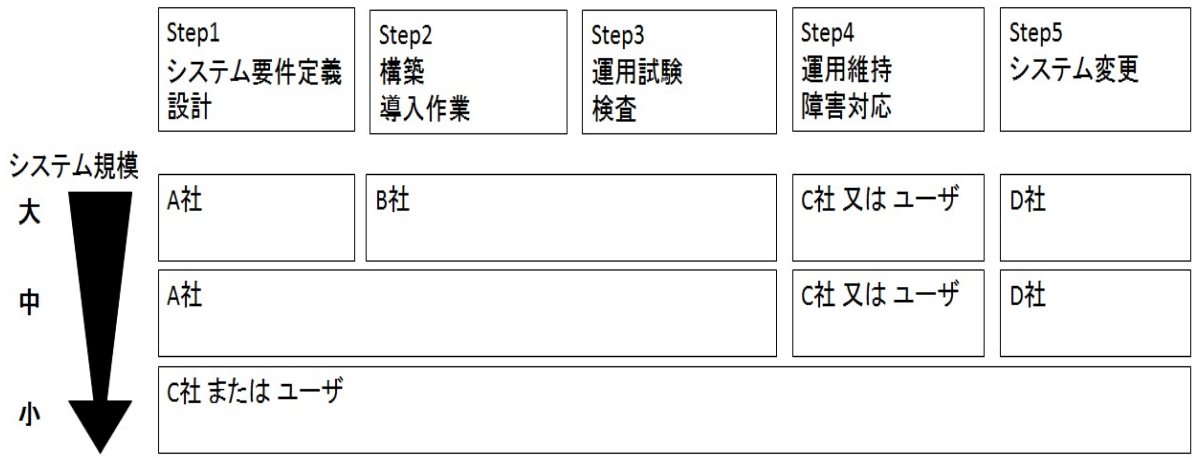

図 2.1　システムの構築から運用開始までの流れ

大規模システムではStep1のシステム要件定義・設計が，システムを導入する組織の根幹業務に影響を与える。このため大規模システムでは，Step1に特化したベンダがこれを担当する(図2.1中のA社)。さらに大規模システムでは，Step2の構築・導入作業は前出のA社と異なるベンダが実施する(図2.1中のB社)。また，Step3の運用維持・障害対応段階でもこれに特化した別ベンダにより実施される(図2.1中のC社)。

一方で小規模システムでは，Step1のシステム要件定義・設計から始まり，Step5のシステム変更までの全てを1社が行う。この規模のシステムでは，各Stepの実行をユーザ自身が行っていることも多い。

本研究で対象としている中・小規模システムでは，運用維持，障害対応の段階で人員不足傾向があることが特徴である[3]。また，システム構成の中核であるサーバ台数は，中規模システムでは数台から数十台，小規模システムでは1台から数台である。さらに，中・小規模システムでは同一製品を用いて構成されていても使用目的が多様であるため，障害事例の相互参照が困難なことが多い。このため人手によって障害調査が行われる比率が高くなっている。

2.3 運用開始後の保守体系

システムの運用開始後に必要な項目は，以下の3つに分類されると考えられる。

(1) 運用維持

システムの運転・障害監視，システム利用者への対応等

(2) 障害対応

障害原因調査，対策立案と実行

(3) ベンダの製品サポート

製品ベンダによる技術サポート(単体障害調査)

図2.2にオープン系システムの典型的な保守体系を示す。図中の方式1は大・中規模，方式2，3は中・小規模システムに多く見られ，主に予算規模に応じて選択され

る。３つの方式はどれもシステム運用，障害対応に必要な項目を全て網羅しているように見える。しかしながら実際の障害対応では不十分であることが多い。これは保守方式そのものにではなく各契約内容，障害調査に必要な情報の伝達方法，障害対応の全般過程の統制に問題があるためである。詳細を次節以降で分析する。

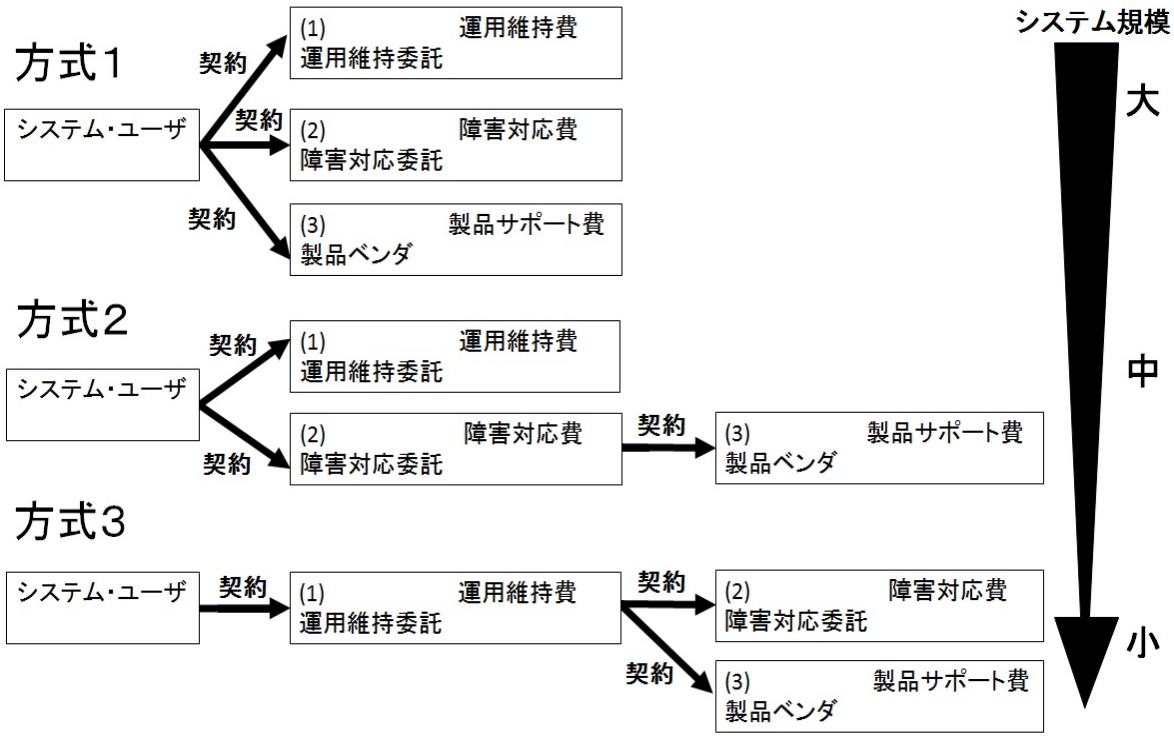

図 2.2　オープン系システムの典型的な保守体系

2.4　費用分析

運用開始後の費用分析は次のようになる。

(1) 運用維持費

システム規模に比例する。運用が長期間となりシステムの安定性が向上するのに伴い削減されることがある。このことはシステムに習熟した人員の減少を招き，障害発生時の対応時間の増加要因となる。

(2) 障害対応，対策実施費

不規則に発生し，障害調査および対策実行ともに所要工数の事前予測が困難である。小規模システムでは未契約，予算未取得となっていることが多く，障害対策実施時の所要時間増加につながる。

(3) 製品サポート費

製品価格に連動するのが一般的で，その割合はソフトウェアで年間 20%程度[4][5]，ハードウェアで 15%程度[6][7]である。ソフトウェアの方がハードウェアより比率が高い理由は，修正プログラム提供期間がより長期に渡るためと考えられる。また，ハードウェア保守契約には，交換部品の無償提供が含まれることが多い。さらに文献[7]からは，運用開始後３～５年の間と、６年以降では価格に大きな差があることが読み取れる。これはサポート期間の長期化が，ベンダにとってのコスト増加を招いているためと考える。

ソフトウェア，ハードウェアのサポート契約は，どちらもシステム規模が小さいほど未契約である傾向があり，製品購入時の無償サポート期間終了後に契約延長が実施されないことも多い。これらが未契約であることは障害対応時間の増加要因となる。

実際の運用ではこの他にシステム設置施設の維持費，通信回線費用等も発生するが，ここでは考慮しない。

つぎに(2)，(3)の共通事項として障害発生の受付時間帯をあげる。殆どが 24 時間 365 日方式，または平日 8 時間方式であり，これ以外の方式はあまり見られない(文献[7])。以上の方式は技術的な要因からではなく，システムを運用するユーザの営業時間帯から決定されていると考える。

システム規模が小さいほど平日 8 時間方式が選択されやすく，この結果，対策実行完了までの待ち時間が発生し，総対応時間が増加する。

最後に見落としてはならないのが，オンサイト費用である。オンサイトとは障害調査，対策実行の過程で必要となる現場作業である。オンサイトが障害対応時間に影響する例としてハードウェア障害の部品交換をあげる。オープン系システムで使用されるハードウェアは，サポート契約が締結されていれば交換部品の提供は無償であるが，オンサイト費用は別途(有償)であることが多い。またオンサイト対象部品(FRU *1)と，非対

象部品(CRU *2)が混在した製品が存在し，交換作業時に契約上の問題を起こすことがある。この結果，部品交換に要する時間の増加を招く。

***1** FRU (Field Replacing Unit)： オンサイト作業を製品ベンダが実施する交換部品

***2** CRU (Customer Replacing Unit)： 部品の無償提供のみが行われる交換部品

2.5 障害原因調査に必要な情報

オープン系システムの障害調査には，以下の(A)から(C)の情報が必要であると考えられる。ここではログ情報などが注目されがちである。しかし実際に障害対応が完了するまでの全体を見渡すと，パスワードや操作記録などの注目されにくい情報が重要であることがわかる。例えば，パスワードが不明であるために障害対策作業が実行できない，操作記録が不明であるために人的ミスによって障害が発生したことが判断できない、などが挙げられる。以下は，障害原因調査に必要な情報のまとめである。

(A) 固定情報

(A-1) システム構成情報

設定値，製品バージョン情報，ネットワーク構成など

(A-2) システム管理情報

パスワード，定期ジョブ運行スケジュールなど

(B) 障害に特化した情報

エラー情報，発生頻度情報，操作記録，ログ情報など

(C) 障害発生履歴と対策履歴

対象システムで発生した障害一覧とその対応履歴

これらの情報はシステム習熟者が不在の場合は，障害発生時に即時に参照，利用できないことが多く，対策実行の時間ロスにつながる。(A-1)は運用中に定期，不定期に変更される情報である。(B)は障害調査に必要な情報の主たるものである。近年ではセキュリティ対策の一環としてこれらの情報を扱う組織，人員を厳しく制限する傾向があり，調査に必要な情報が有効活用できない，調査即時性が著しく損なわれる等の問題が

発生している。(C)は障害調査に必須の情報とはいえないが，適切に管理されていれば対応時間の短縮につなげられる。

2.6　製品ベンダの開発方針と障害対応

製品ベンダは，競合他社よりも早く製品化を実現するために短期間で開発を完了する。このことは，新技術を採用した製品がいち早く提供される利点となっている。反面，製品使用開始までに十分な試験をする時間を確保できていないことは，様々な問題を発生する要因となっている。これらの要因を挙げると以下のようになる。

・製品化優先のために完成度が低い

・製品ベンダが異なると，同一機能を同じ手順で利用することが困難

・機能拡張で差別化をするため，バージョンアップの頻度が多い

・修正プログラム提供による製品販売後の品質向上が常態化している

・不具合修正のために，ユーザは頻繁に修正プログラムを適用しなければならない

・製品発売開始からサポート終了までの期間が短い

オープン系システムでは，上に挙げた要因を考慮した原因部位調査手法が求められる。本研究ではこれらの要因に対応するために，調査対象としているシステムが正常に稼働している際の情報を事前に採取しておき，この情報を障害発生判断材料に用いることを考える。

各ベンダは自社製品以外の障害原因調査は受付けない。このため，ベンダ保守のみでシステム障害原因の全てを究明することはできない。システムユーザは，障害発生時にシステムで使用されている製品を供給した各ベンダに障害調査依頼を行うが，これを自ら統制して障害発生原因を究明する必要がある。

2.7　エラーロギング機能

障害発生時に原因調査を行うための手段として，各製品にはエラーロギング機能がある。この機能に実装規約等はなく，各ベンダは独自の方針で開発する。ログ情報は障害

原因調査の重要な手掛かりとなる。ところが，障害原因調査時のログ情報収集には以下の問題が生じる。

・ベンダ独自の実装形態のために，採取や参照の手順が統一されていない

ログ情報の採取方法は，定期自動採取，コマンドによる採取，GUI 操作による採取など，統一されていない。

・OS に標準装備されたログ機能を避ける傾向がある

大規模なアプリケーションは，OS が提供する標準ログ機能を使用しない。このため，各製品のログ情報の採取手段，保存形式が共通化されていない。

・セキュリティ事情による提供拒否

採取された情報の中には，ユーザ固有の情報(IP アドレス)などが含まれていることがある。このような場合は有益な障害情報が採取されていても，ユーザはログ情報を原因部位調査に用いることを拒否することがある。

したがって，複数ベンダから供給された製品でシステムを構築しているユーザは，障害原因調査時にこれらの違いの全てを把握した上でログ情報を採取する必要がある。そこで，障害発生に備えて以下を事前に実施しておく必要がある。

・各製品のログ情報採取方法の把握

・障害発生時のログ情報採取フローの策定

2.8　修正プログラムの提供方法

オープン系システムは複数ベンダの製品を統合して構築されるため，個々の構成品の保守方式は異なっている。修正プログラムの提供方法はその好例で，提供頻度・周期，期間，方式，有償・無償など様々な違いがみられる。

ここでは修正プログラムの提供方式とその適用状況に焦点を当て，実際のシステム保守の実態を分析する。個々の製品への修正プログラム提供は頻繁に行われるが，システムに組み込まれて運用段階に入ったものへの適用率は非常に低いのが実状である。これはユーザが必ずしも技術的要因のみから修正プログラムの適用可否を決定できないことが理由となっている。以降では，これらの意思決定の背景について言及する。

修正プログラムの提供形態は，システム継続運用に大きな影響を与える。ここでは修正プログラム（以降パッチと呼称する）の提供形態について説明する。

(A) パッチ提供周期

パッチの提供周期は大きく２つの形態に分類される。

(A-1) 定期的＋臨時提供の併用

(A-2) 不定期提供

(A-1)は比較的規模の大きい製品(OS，大型周辺装置等)によくみられる方式で，３ヵ月，６ケ月などの間隔で定期パッチが提供される。併せて大きな問題に対しては臨時パッチが提供される。提供間隔は技術的な理由からではなく，４半期管理と同期する目的から決定されている。(A-2)は比較的規模の小さい製品でよくみられる。例としてPCのBIOS，拡張カードのドライバ等が挙げられる。セキュリティ関連のパッチは不定期提供であることが多い。

(B) パッチ提供期間

パッチの提供期間は様々な要因の影響を受けるが，製品販売開始時にはベンダでも決定されていないことが殆どである。一般的に提供期間はその製品の市場での流通数(Install Base)に比例すると考えられ，長い場合は提供期間が10年を超えることもある。提供期間を２バージョン・サポート方式とするベンダもある。図2.3にこの方式を示す。同方式では，提供期間の定義が前出のものと異なることに注意を要する。２バージョン・サポート方式の製品は，後継品が長期に渡り開発・提供されていることが多い。

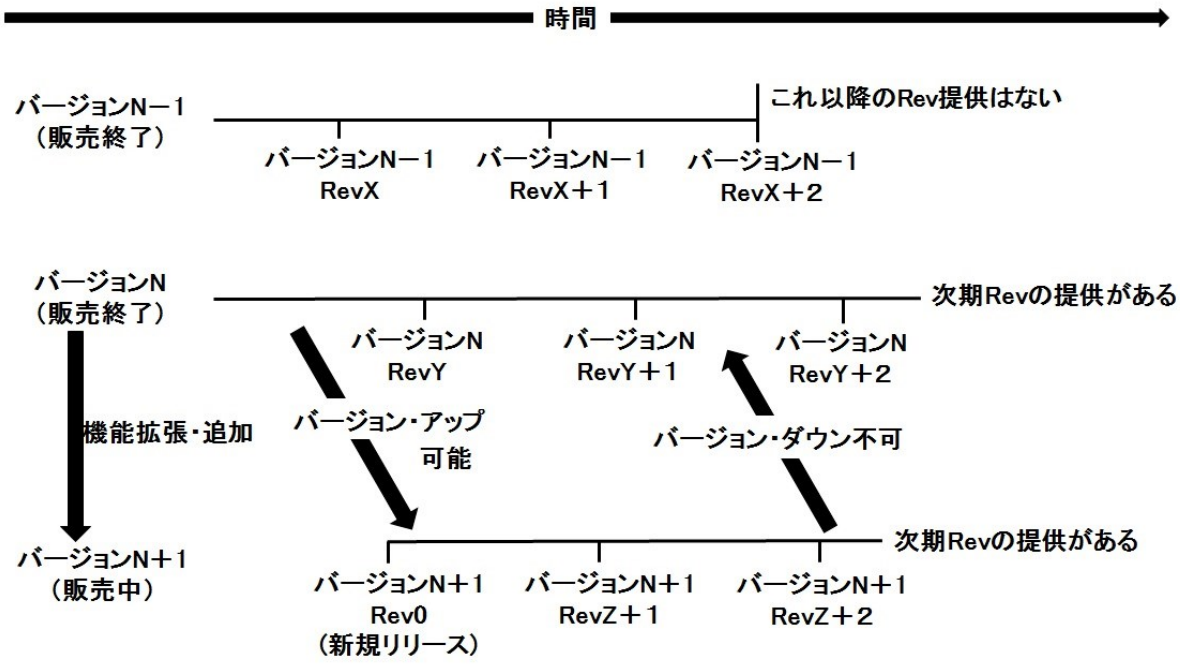

図 2.3　２バージョン・サポート方式

(C) パッチ配布方法

(C-1) 個別配布方式

(C-2) 一括配布方式

(C-1)は特定の問題に対する修正を行ったパッチである。この方式は複数の個別パッチを適用する際に前提条件管理が煩雑になる問題があり，近年では比較的規模の大きい製品では採用されなくなってきている。(C-1)に代わり増加してきたのが(C-2)の一括配布方式である。これは複数の個別パッチを一括で配布する。ユーザは特定の個別パッチを選択適用することはできない。しかし一括配布方式は前出の(A-1)定期提供となっていることが多く，ユーザは不定期に提供されるパッチの検索・調査を定常的に実施する必要がなくなる。またベンダ側はソース・コードのバージョン管理負荷低減となる。ただし一般的にユーザは適用するパッチの個数を少なくしたいと考える傾向が強く，一括配布方式で提供されたパッチの適用率を下げる要因となっている。

(D) システムへの影響度と緊急性

(D-1) セキュリティに関係する問題の修正

(D-2) データ破壊に関係する問題の修正

(D-3) システム停止を引き起こす問題の修正

(D-1),(D-2),(D-3)はユーザがパッチの適用を決定する要因として影響の大きいものである。これらは運用時間に比例して問題が発生する可能性が大きくなると考えられ，適用の緊急性が高いと考えられる。その一方で，PC のオンライン・アップデート等は影響度，緊急性に関係なく適用率が高いことにも注意を要する。

本研究では，パッチ適用の影響度と緊急性は考慮しない。なお，オンライン・アップデートが原因で発生する新規障害の調査に関しては，別途研究を行う必要があると考える。

(E) 有償，無償(パッチ・ファイルの提供)

パッチ・ファイルの提供には，有償と無償のものがある。ベンダの方針は様々であるが，一般的には規模の大きい製品は有償の保守サービスに加入することがパッチ・ファイル提供の前提となっていることが多い。

(F) 有償，無償(パッチの適用作業)

オープン系システム向け製品は，ほぼ例外なくパッチの適用作業は有償である。すなわち製品購入およびパッチ・ファイルの提供サービスと，パッチ適用作業は切り離されている。このこともパッチの適用率を低下させる要因となっている。これについては 2.10 節にて詳述する。

(G) システム停止の必要性

パッチ適用に際してシステムの停止が必要かどうかは，適用率に密接に影響する。近年では，中・小規模のシステムでも 24 時間稼働を行っていることが多く，パッチの適用に伴うシステム停止の必要性は重要な要因である。

2.9　システムの構築・運用とパッチの関係

システム構築時には，そのシステムを構成している製品間での互換性は保証されていることがほとんどである。これは主に次の理由による。

(A) 構築時に選択された製品はベンダが販売を継続・開発中のものであるため，ベンダが他社製品との接続互換試験を開発の一環として実施していることが多い

(B) システム構築者が独自に試験を実施したため稼働実績がある

ただし(A)の接続互換の意味は，以下のような差異があることに注意を要する。

・ベンダで実際に広範囲の相互接続試験を実施済み

・相互接続試験は実施済みであるが，試験範囲が限定されている

・公開されている試験条件が不明確かつ広範囲のため，実際には動作に問題があることがある。ただしベンダは，公開した条件に合致していれば障害発生時の対応は受け付ける。(例：ベンダの試験環境で使用した製品のバージョンは明記されているが，パッチ適用有無の記述がない等)

構築が完了して運用を開始した時点では，互換性が確認された製品で構成されていたシステムでも，その後のパッチ適用で互換性を失って行く。このことから互換性の精度はシステムを構成する製品の種類・個数，パッチの適用回数に反比例すると考えられる。この事実はパッチの適用率を下げる大きな要因であり，オープン系システムの特徴と言える。これらの理由から，一旦運用を開始したシステムに対して一切のパッチ適用を実施しないユーザも多く存在する(バージョン固定運用)。ただしこの方針は，ハードウェア障害による交換作業で新しいファームウェアがインストールされた部品が使用されることには対応できない。このためバージョン固定運用をソフトウェア製品に限定した方法が採用されている場合もある。また小規模システムではファームウェアのバージョンが管理されていないことも多い。

2.10　パッチ適用の判断基準

提供済パッチが，どの程度の割合で稼働中システムに適用されているかの実態を把握することは非常に困難である。特に運用開始から一度もパッチを適用していないが障害

発生がないシステムでは，外部との接点がないために使用中のパッチバージョン情報はベンダ側に伝わらない。また小規模システムではメーカとの長期保守契約を持っていないこともある。これらがパッチ適用率の把握をより困難にしている。

ユーザ，パッチ適用を決断する要因は以下である。推定適用率は，筆者らが特定の製品に対してどのバージョンで稼働中であるかを調査した結果による(文献[8])。

・発生した障害の原因が提供済のパッチで修正されている場合

（推定適用率：30%）

・対象システムが提供済のパッチで修正された問題の発生条件に一致している場合

（推定適用率：30%）

・一括パッチの適用をシステム保守の一環として実施することが決定されている場合

（推定適用率：20%以下）

最新パッチの適用率はさらに低く，10～15%程度と推定される。適用率の低さには次の理由がある。

・一括パッチの中から個別パッチを選択して適用できない

（システムに合致するパッチのみを適用したいと考える）

・パッチ適用により未知の問題が発生すると考えること

・ユーザ独自の適用判断基準を満たさない

例：ソフトウェア変更に対して，ユーザが独自の事前試験実施を条件としている場合など

・システムに組み込まれている他製品との互換性

・不可逆性

例：ファームウェアは，ほとんどがバージョン・ダウンをサポートしない

・適用作業実行者の不在

例：運用管理・保守を，外部組織へ発注している場合で，その契約にパッチ適用作業費が含まれていない

・新規パッチの存在がユーザに伝わっていない

パッチ提供のアナウンスはベンダの Web が基本であり，ユーザはこれを能動的に検索する必要がある

・システム停止の必要性

パッチ適用作業手順に，システムの再起動が含まれている

以上の要因のうちシステム停止の必要性は比重が大きい。近年では小規模システムでも 24 時間 365 日稼働を前提としているものが多い。システム停止に伴う周囲への影響の考慮と，日程調整の煩雑さがパッチ適用に伴うシステム停止の阻害要因となっている。このためベンダ側でも各パッチのシステム停止の必要性が比較的容易に確認できるようにしている場合もある。

一方で比較的適用率が高いのは，セキュリティ対策，データ消失や破壊に関連するものである。これらのパッチはシステム運用の前提に影響を与えるものであり，結果的に高い適用率になっている(例：クライアント PC へのセキュリティ脆弱性対策)。

2.11 まとめ

オープン系システムは複数ベンダから提供された複数の製品で構成されていることが特徴といえるが，このことが障害原因調査，対策実行時には負の要因となる。特に，システム構成に習熟し障害対応全般を統制する管理者の存在は障害対応時間に大きく影響する。しかしながら，中・小規模システムではこのような管理者の確保は予算面から困難なことが多い。情報共有に関連した技術の発達は著しいが，本章での分析結果によれば，障害対応時に必要な情報を迅速かつ有効利用するためには依然として人手を必要としている。

次に述べたシステム保守の現状も，オープン系システム特徴である。新製品開発が速いことで頻繁に提供される修正プログラムに対して，ユーザは自身で適用可否の判断を行わなければならない。また，複数ベンダから供給された製品でシステムが構成されているために，障害発生時にはユーザが全体を統制しながら調査と対策を実行しなければ

ならない。このこともオープン系システム障害に人手による対応が必要な理由となっている。

修正プログラムの適用判断は，システムを構成する製品の個数に比例して困難になる。また修正プログラムの中には，セキュリティ対策など適用必然性が高いものもある。これらの技術的な要因に加え，予算制約，心理的要因，管理問題などの非技術的な要因も加わり，結果としてパッチの適用率は低い水準に留まっているのが実状である。一方でパッチの未適用が既知の障害を発生させる原因ともなっているため，適用判断の方法を探究してゆく必要がある。

次章では，本章で分析したオープン系システムの特徴を考慮した障害原因部位特定手法を提案する。

〔参考文献〕

[1] 国内でのサーバ稼働数：

https://www.sbbit.jp/article/cont1/24976, (2018-11 閲覧)

[2] 世界でのサーバ稼働数：

https://www.quora.com/How-many-servers-exist-in-the-world, (2018-11 閲覧)

[3] 篠原昭夫，泉隆：「オープン・システム障害対応の現状分析」，情報処理学会第 77 回全国大会，3ZE-05 (2015-03)

[4] ソフトウェア価格と保守費用：

https://it.impressbm.co.jp/common/dld/pdf/3c2d5c5f2d2463cc199da85331ebee83.pdf, (2018-11 閲覧)

[5] ソフトウェア保守費用の平均値に関する記述：

https://it.impressbm.co.jp/articles/-/7468, (2018-11 閲覧)

[6] ハードウェア保守内容の説明資料：

http://www.hitachi.co.jp/products/it/server/portal/pcserver/pdf/sp-790.pdf, (2018-11 閲覧)

[7] ハードウェア保守費用の資料：

http://www.hitachi.co.jp/Prod/comp/OSD/pc/ha/products/hardware/ns/pdf/m1l2/2_extoption-NS_L2.pdf, pp2-4-1-21 - 2-4-1-29, (2018-11 閲覧)

[8] 篠原昭夫, 泉隆：「オープン・システムでの修正プログラム適用実態報告」, 電気学会全国大会, 3-063 (2015-03)

第３章　　機能線を用いた障害原因部位特定手法の提案

3.1　はじめに

前章までに，オープン系システムの運用・保守・障害対応の現状を分析した。さらに，障害原因調査においては，オープン系システム特有の背景があることにも言及した。本章では，これにより明らかにされたオープン系システム障害調査に求められる要因を考慮した，機能線による障害原因部位特定手法を提案する。

提案する機能線は，オープン系システムの特徴を反映した障害原因部位特定に用いるグラフ図形である。機能線は，障害をシステム内のある二点間でのデータ転送に問題が発生したと捉えることに基づくもので，データの流れに沿って配置されるコンポーネントを結んだグラフ構造で表わされる。この機能線は，システムを構成しているコンポーネントの関係を図表現することで，障害調査をシステマティックに行うことを目指したものである。また機能線はその分解レベルを段階的に上げることで，障害原因部位をより詳細かつ効率的に特定することが可能である。なお，機能線は人手による原因調査に従事する技術者に筆者らが聴き取り調査を行い，障害原因部位特定に至るまでの過程をモデル化したものである。本章では，はじめにオープン系システム障害調査での課題を示し，それを解決する手段として機能線を提案する。同時に調査を効率的に進めるためのコンポーネント分解レベルと，調査対象を削減するための補助機能線についても述べる。

3.2　人手による障害原因特定調査に求められる要件

オープン系システムは，複数ベンダの製品でシステムが構成されているため障害原因部位特定が困難なことが多い。また，各製品は販売期間が短いうえにパッチが多く提供されるため，事例報告の無い未知障害の発生件数が減少しない状況が続いている。表１は，ある PC 向け OS に対する修正件数の比較である。Update(*1)と Hot Fix(*2)の合計を比較すると，後継製品に対してより多くの修正が行われたことがわかる。

表１ OS の世代による修正件数の違い

	Update	Hot Fix
Windows7 SP1	63	10
Windows8/8.1	168	5

*1 Update：　製品の不具合修正および機能変更・拡張

*2 Hot Fix：　緊急性のある更新が含まれた製品不具合の修正

一部のベンダでは，特定の製品に対して自動化された原因調査法を導入している例が見られる。この他に原因調査を自動化する研究も存在するが，特定の規格に準拠した製品の使用を前提とするなど，多様なオープン系システムの全てに適用することは難しい[1]。より大規模なシステム向けの自動化原因調査手法として文献[2]がある。これらの自動化手法では，調査対象をシステム全体とした前提で調査を開始していることに対して，本章では中・小規模システムを対象とし，かつ調査対象がシステム全体とならない手法を提案する。また，文献[1][2] では共にハードウェア障害の原因部位特定に限定しているが，本章ではハードウェア,ソフトウェアを区別せずに全てを調査対象としている点が異なる。

さらに，開発が完了してシステム上で運用されている製品では，障害調査時にデバッグ情報などは参照できず調査情報が不足することが多い。文献[3]では，復旧を最優先としながら原因調査を進行させる試みをしているが，多様な障害に適用可能な手法になっていない。

前章までに分析した結果を踏まえると，人手による障害原因部位特定手法では次の要件を満たすことが望まれる。

・オープン系システムの多様性：

　オープン系システムの障害原因部位特定に特化し，多様な障害調査に適用可能

・ハードウェア，ソフトウェアの違いを意識しない：

　ハードウェアおよびソフトウェアを統一的に扱い，様々な製品を調査可能

・調査情報の不足：

　調査情報が不足する条件下でも障害原因部位を特定可能

3.3　人手による障害調査手法のモデル化

3.3.1　人手による障害調査により原因特定に至るまでの過程

前節で定義した要件を満たす障害原因部位特定手法を確立するために，実際に人手でオープン系システムの障害を調査する技術者への聴き取り調査を行った。対象者は特定のハードウェア，ソフトウェア製品の調査を専門に実施している技術者，およびシステム全体の障害調査を統制する技術者である。この結果から，以下のような障害原因部位特定に至るまでの共通事項を抽出した。

・多くの場合，システム全体を対象に調査を行っていない

・障害原因部位に関連する対象を，情報(データ)の流れに注目して調査している

・おおまかな障害原因部位を特定したのち，より詳細な部位へと段階的に調査を進めている

・システム正常稼働時のログ情報を障害発生有無の判定基準に利用している

・上記はハードウェアとソフトウェアの違いに関係なく行なわれている

以上のことを踏まえ，抽出した共通事項から調査過程をモデル化することを考えた。図 3.1 は比較的簡単な障害を視覚的に表記したものである。それぞれの障害と原因は以下の通りである。

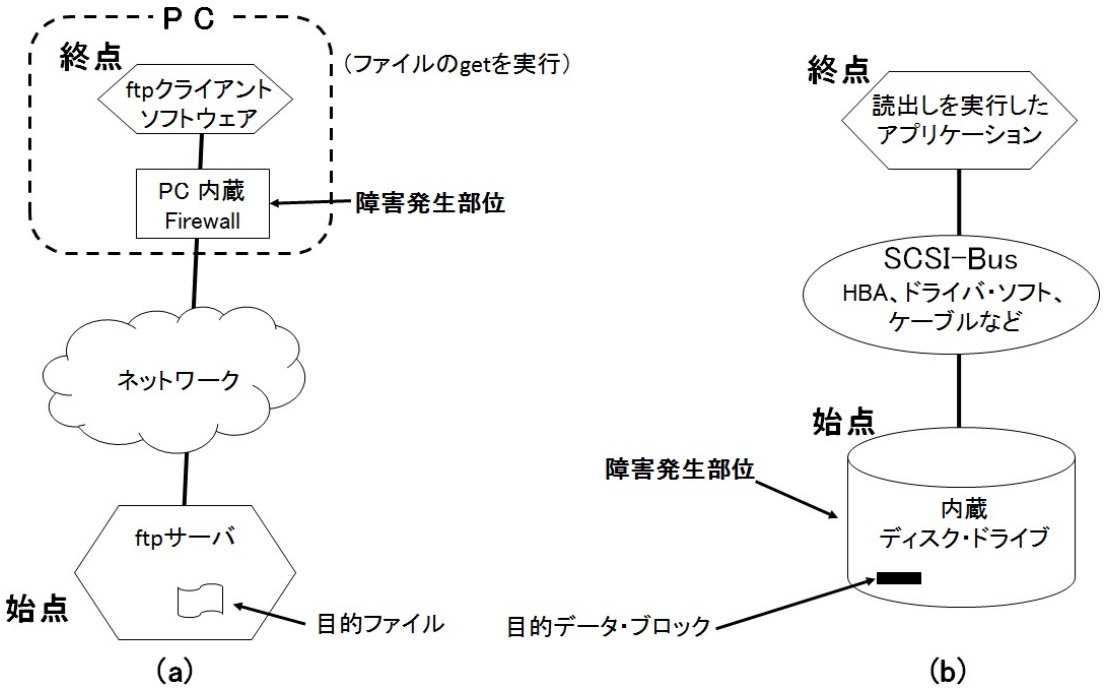

図 3.1　障害の視覚的表記

図 3.1(a)の障害：ftp クライアントでファイル get が失敗した

図 3.1(a)の原因：ftp クライアント PC の内蔵 Firewall に設定されたデータの同時転送制限量を超えたために強制中断された

図 3.1(b)の障害：PC 内蔵ディスク・ドライブの read に失敗し ASC/ASCQ＝03/11[4] を報告した

図 3.1(b)の原因：PC の内蔵ディスク・ドライブで目的データ・ブロックが破壊されている

このように視覚的に表現すると，障害全体像が容易に把握でき調査の方向性が明確になる。

3.3.2　障害原因部位と発動点

システム利用者は，何らかのエラーメッセージを受け取り障害の発生を認識することが一般的である。ほとんどの障害調査は，このエラーメッセージを発出した箇所から開始される。この箇所を発動点と呼ぶ。発動点は必ずしも障害原因部位と一致しているとは限らない。障害原因部位と発動点の関係を図 3.2 に示す。

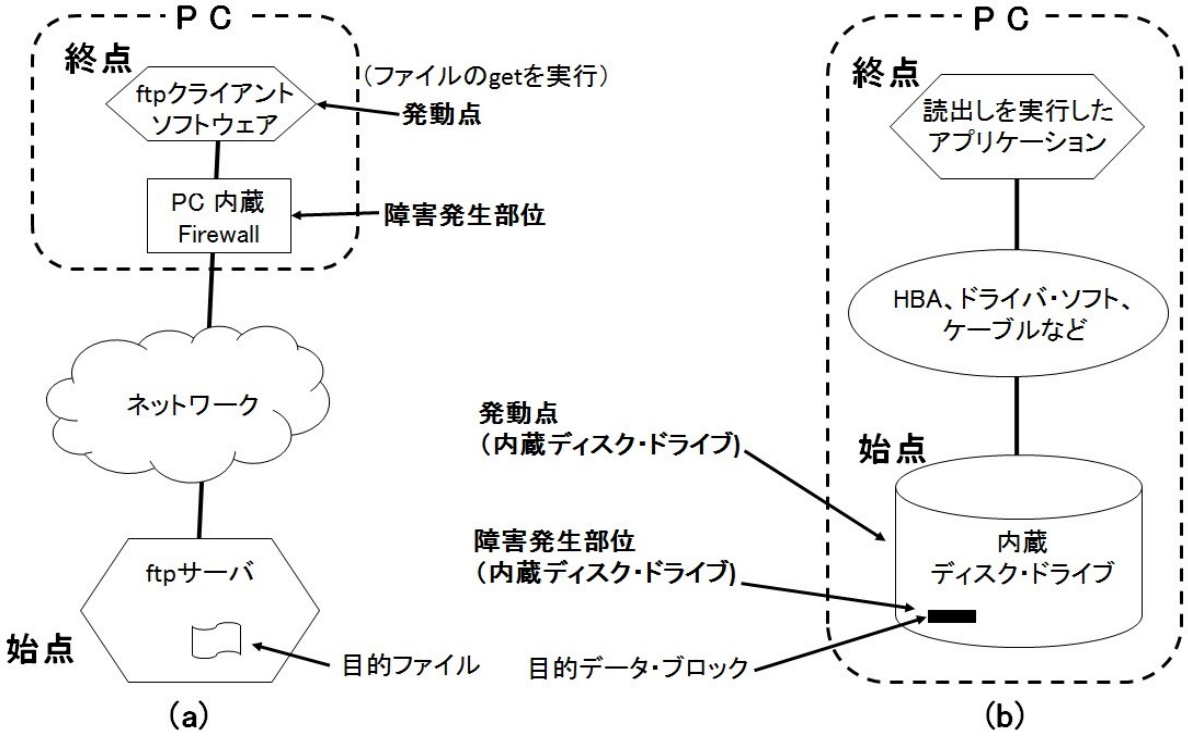

図 3.2　障害原因部位と発動点

図 3.2(a)の発動点は ftp クライアント・ソフトウェアで，障害発生原因となった PC 内蔵 Firewall とは異なる部位である。一方で図 3.2(b)では ASC/ASCQ を報告したのは障害が発生した PC の内蔵ディスク・ドライブ自体で，発動点と障害発生原因部位が一致している。

3.3.3　コンポーネント

図 3.1 および図 3.2 の(a)と(b)を比較すると，異なる障害であっても視覚的表記は類似していることに気付く。すなわち(a)では目的ファイルの転送に対して３個の要素，

PC，ネットワーク，ftp サーバが関与している。また(b)では，目的データ・ブロックの転送に対して，アプリケーション，SCSI-bus，内蔵ディスクの３要素が関与している。これを踏まえて，データ転送に関与する要素を次のように定義する。

［定義］ 目的とするデータ(target data)の転送に関与する要素を「コンポーネント(component)」と定義する。

コンポーネントの具体例をいくつか挙げる。

・ネットワーク・カード（ハードウェア）

・デバイス・ドライバ（ソフトウェア）

・SCSI ケーブル（ハードウェア）

・ディスク・アレイ装置（ハードウェア）

・ftp クライアント・ソフト（ソフトウェア）

図 3.3 にコンポーネント 3 個が関与する目的データの流れの例を示す。これを図 3.2(a)に対応させると，それぞれのコンポーネントは，PC，ネットワーク，ftp サーバとなる。

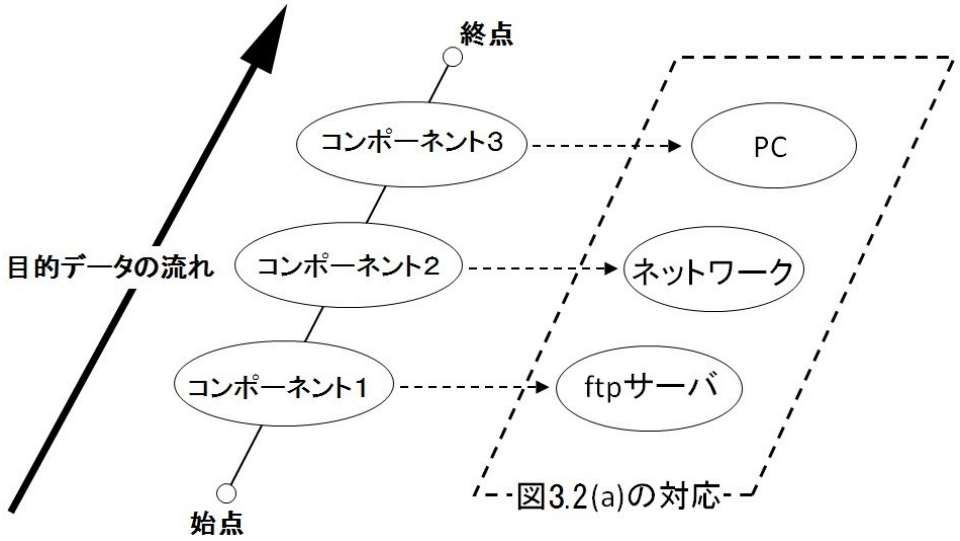

図 3.3　コンポーネントの例

3.3.4　機能線の提案

図 3.3 の表現を踏まえ，オープン系システムの障害原因部位特定に用いるグラフ図形として「機能線（Function Chain)」を提案する[5]。

［定義］ オープン系システムの障害をシステム内のある二点間で目的データ転送中に問題が発生したものと捉え，目的データの流れに沿って配置されたコンポーネントを結んだグラフ構造の図形を「機能線」と定義する。

機能線は以下のようなシンプルな構造からなる。

1. 分岐のない１本のグラフである
2. 始点(source)には目的データが格納されている
3. 終点(recipient)は目的データの到達する点である

機能線においては，障害は目的データ転送の失敗であると捉える。すなわち障害は目的データの転送が正しく行われなかったか，またはそれが期待する場所に存在していなかったために発生すると考える。この考えに基づくと，機能線上には障害原因部位が必ず存在する。１つの機能線に複数の障害原因部位が存在する多重障害もあるが，ほとんどは障害原因部位が一か所の単純障害である。そこで本研究では単純障害のみを対象とする。なお多重障害に関しては，別途検討を行っており６章において述べる。

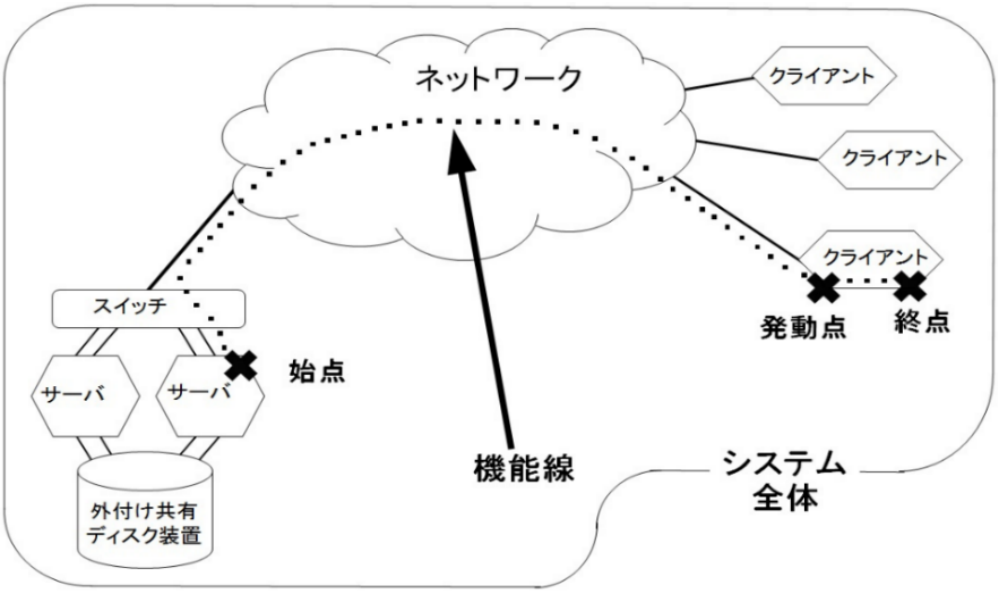

図 3.4　システム全体と機能線

図 3.4 は調査対象システムと，機能線の関係を示したものである。機能線では始点から終点までの経路上のみを調査すればよく，システム全体を対象とする必要がない。

3.3.5　コンポーネント分解レベル

障害調査の目的は対策立案である。しかし，コンポーネントの規模は大小様々であり，障害原因部位であるコンポーネントが特定されても対策立案には不十分なことがある。この場合，コンポーネントをより細かく分解し，さらにその中でどのコンポーネントが障害原因部位かを絞り込む必要がある。そこで，詳細に分解する度合いを「分解レベル」と呼ぶことにする。ここで目的データを元にして最初に作成する機能線を分解レベル 0 とする。

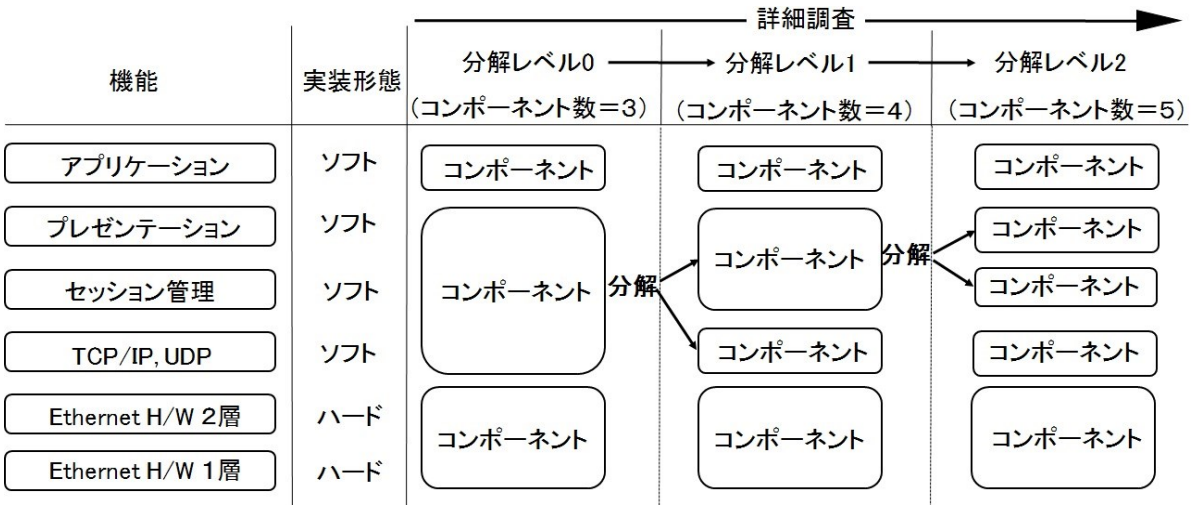

図 3.5　コンポーネント分解レベルの例

図 3.5 はコンポーネントと分解レベルの例を示したものである。この例では分解レベルが低い場合は３個の，より高い場合には５個のコンポーネントに分解している。

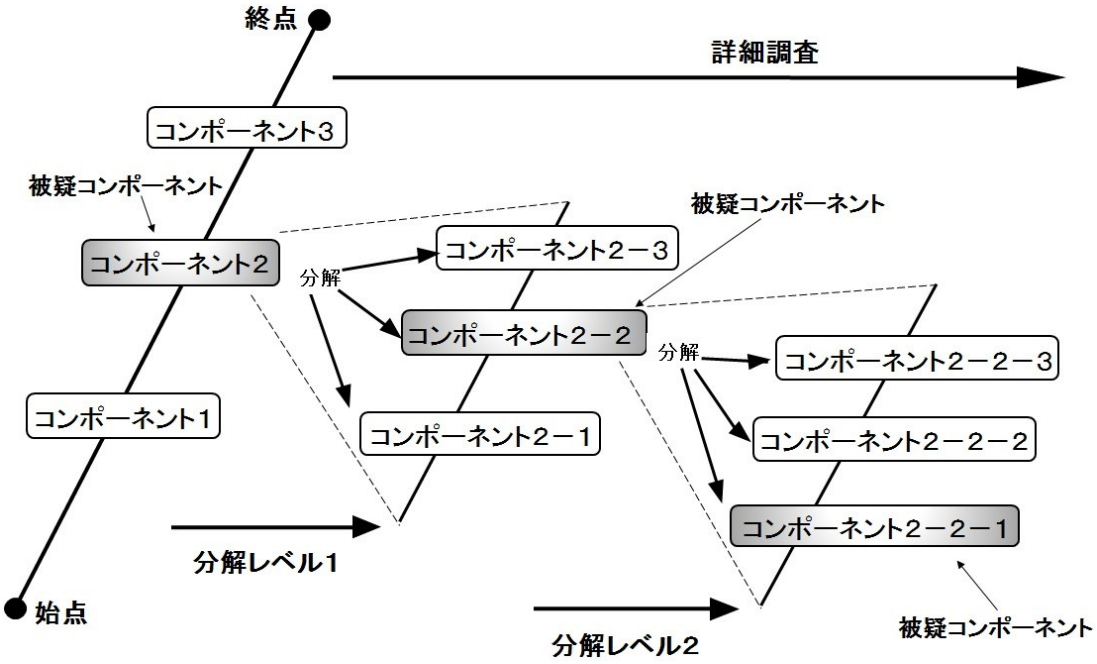

図 3.6　コンポーネント分解と詳細調査

図 3.6 は被疑コンポーネントの分解レベルを上げて障害原因部位を調査し，より詳細に障害原因部位を特定する様子を示している。

調査開始時点では少ないコンポーネントで機能線を作成し，より詳細な障害原因部位特定が必要な場合に限りコンポーネント分解を実施するのが最も効率的である。

なお，機能線作成は人手で行われるため実行者により分解レベルに差が生じる可能性がある。しかしこのことは，調査対象コンポーネント数が異なるのみで障害原因部位を誤認する要因とはならない。

3.3.6　コンポーネント分解レベルを製品実装の関係

図 3.7 にコンポーネントの詳細例として iSCSI 構成をあげる[6][7]。iSCSI はハード基盤となる Ethernet カードの上位に TCP/IP が動作するためのコンポーネントがあり，さらにその上位に SCSI 転送に関連する iSCSI Core が実装されている。図 3.7 で iSCSI の中核であるコンポーネント３は，製品 A ではソフトウェアで，製品 B ではハードウェアで実装されている。しかし，機能線ではこの違いを意識する必要はない。

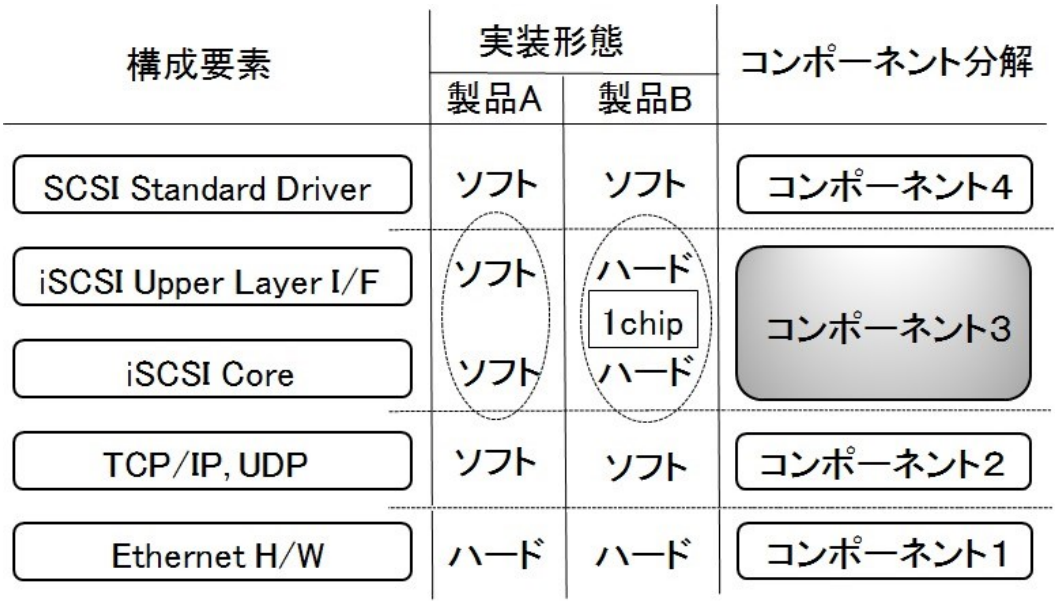

図 3.7　コンポーネント分解と実装形態

コンポーネントの分解レベルを上げる際には，製品構成を意識するべきである。これは障害調査目的が，対策立案であるためである。例として Firmware をあげる。一般に Firmware は，image(*1)と NVRAM(*2)が一体で提供される。詳細調査で NVRAM に問題があると特定できても，互換性の制約から NVRAM のみを変更・更新することは困難で，分解レベルを上げたことが必ずしも効果的でない。一方で再発防止までを考慮した詳細な障害原因部位特定が必要な場合には，分解レベルを相応に上げる必要がある。この傾向はハードウェア製品の障害調査で多く見られる。例えば，製造不良が原因のハード障害では，部品交換では再発可能性があり問題が解決しないためである。

*1 image：　Firmware の実行バイナリ・ファイル本体

*2 NVRAM：　Firmware の設定値のみを分離して集約したファイル

図 3.8 は，分解レベルとコンポーネント数の関係を示したものである。図中の(a)と(b)を比較するとわかるように，分解レベルを上げると，調査対象コンポーネント数の

急激な増加を招く。したがって，分解レベルは可能な限り低い状態から調査を開始することが望ましい。

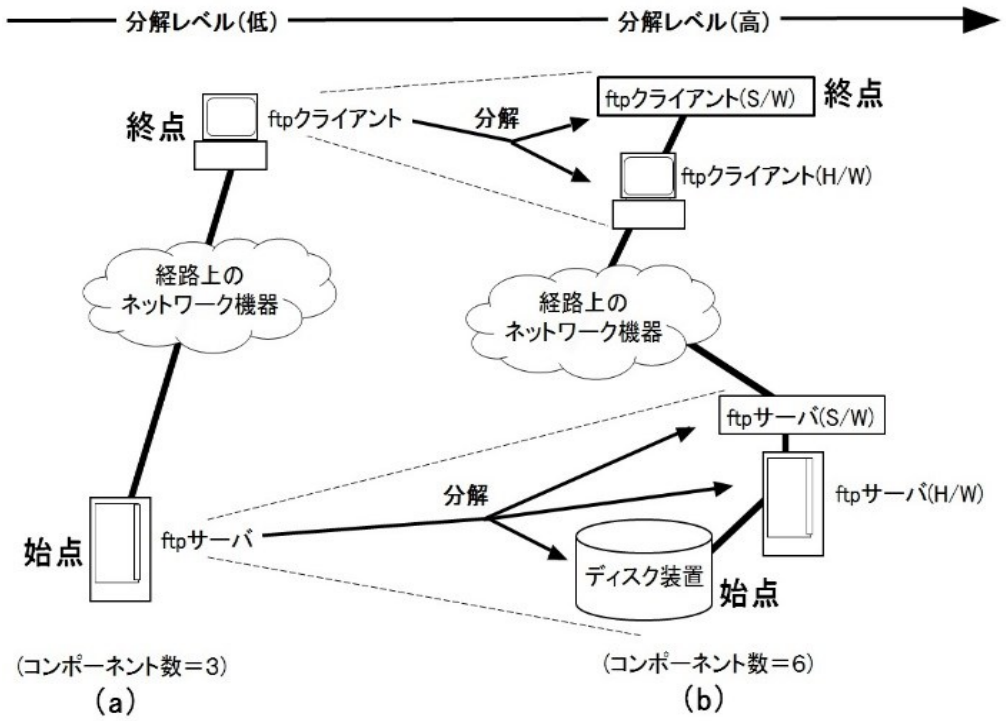

図 3.8　分解レベルとコンポーネント数の関係

分解レベルを厳密に定義して機能線を作成する必要はなく，また機能線上の全てのコンポーネントが同等レベルに分解されている必要もない。障害調査の初期段階では分解レベルを低く抑え，調査の進展や要求にあわせ被疑コンポーネントのみ分解レベルを上げるのが最適な方法である。具体的には，被疑コンポーネントが特定された場合に，その分解レベルで対策を立案可能であるかどうかを判断する。つぎに，そのコンポーネントの分解レベルを上げた場合に，より詳細な被疑コンポーネントの特定ができて対策を実行可能であるかを判断しながら調査を進めて行く。

3.3.7　補助機能線

機能線はデータの流れをもとに記述しているため，調査対象機能線と一部が重複し，かつ障害が発生していないことが判明している他の機能線が存在する場合がある。この情報を用いれば，調査対象機能線と重複する区間を調査する必要がなくなる。そこで調査対象を削減する手段として「補助機能線(Auxiliary Function Chain)」を定義する。

［定義］ 調査対象の機能線と一部が重複した機能線で，かつ障害が発生していないことが判明している機能線を「補助機能線」と定義する。

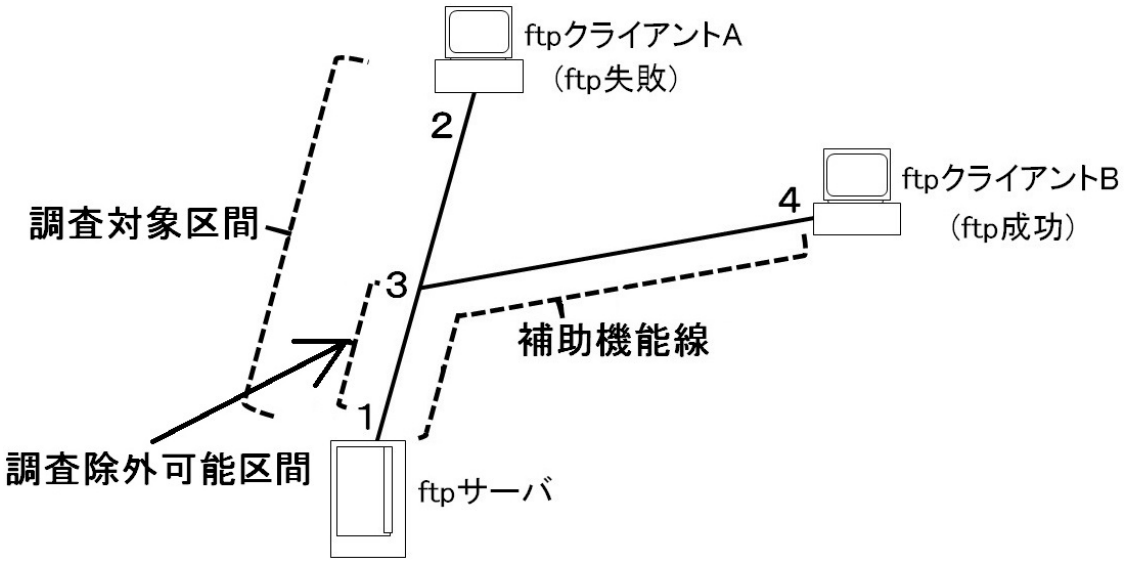

図 3.9　補助機能線

図 3.9 に補助機能線の例を示す。図中で ftp サーバから出発して ftp が成功したクライアント B に至る区間 1-3-4 が補助機能線である。補助機能線上では障害が発生していないため，重複した区間 1-3 は調査対象から除外できる。

3.4　機能線の作成手順と適用可能条件

3.4.1　補助機能線の作成手順

機能線の作成手順をまとめると，次のようになる。

Step1：障害検知エラー情報から目的データを抽出する

Step2：目的データから始点と終点を決定する

Step3：始点－終点間に存在し得るコンポーネントを低い分解レベルで列記する

Step4：列記したコンポーネントを目的データの流れ（始点から終点方向）に沿って並べ初期機能線(分解レベル 0)を作成する

Step5：障害未発生情報を元に補助機能線を作成する

なお，分解レベルは調査の進展にあわせ必要に応じて上げてゆく

3.4.2　機能線の適用可能条件

機能線を適用可能な条件は以下の通りである。

・人手による実行を前提とするため，中・小規規模システムを対象とする

・システム構成情報が採取可能であること

3.5　機能線を用いた障害原因部位特定の例

機能線作成後は，始点から終点方向へ順次コンポーネント内での障害発生有無を判定して行く。具体的な判定方法は次章で述べる。そして原因コンポーネントを特定した後は，対策立案可能となるまで当該コンポーネントの分解レベルを上げて調査を続ける。

図 3.10 は，複数サーバが関係した障害原因調査に機能線を適用した例である。この障害は，あるユーザ ID を用いた NFS クライアントが認証失敗により NFS サーバ上のファイルを参照できなかったものである。原因は認証サーバ群の中に，使用されたユーザ ID 情報を他サーバと同期できていないサーバが存在したためである。

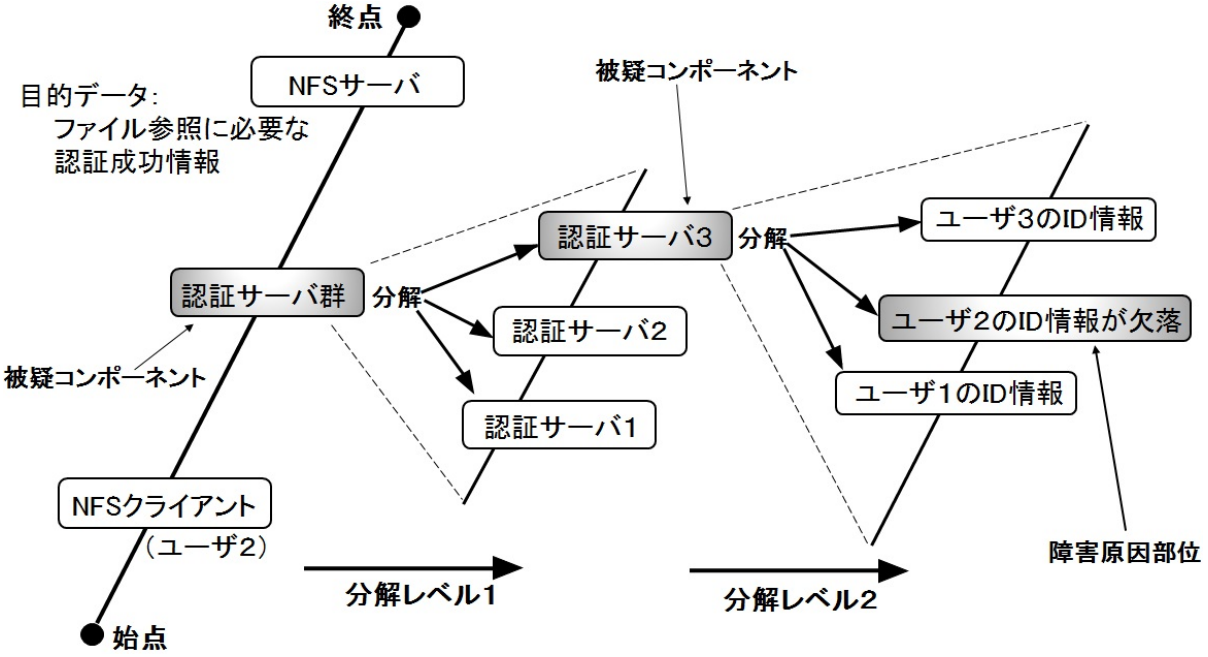

図 3.10　機能線を用いた障害調査の例

図 3.10 の例では，障害検知時の認証失敗情報から，目的データはファイルの参照に必要な認証成功情報となる。そこから機能線を作成すると，コンポーネントは，NFS クライアント，認証サーバ群，NFS サーバとなる。始点から調査を開始し，NFS クライアントに問題がないと判明したとする。そこで次は認証サーバ群全体の管理ログを調査し，認証失敗記録を発見する。ここで分解レベルを上げて，認証サーバ 1，2，3 を順に調査する。認証サーバ 3 上に不明 ID 情報によるアクセス・エラー記録を発見したとすると，このサーバが被疑コンポーネントであると特定される。最後に認証サーバ 3 上の ID 情報一覧に NFS クライアントが使用したユーザ 2 の ID 情報が欠落していることを確認して障害原因部位特定が完了する。

なお，図 3.10 の例では，認証サーバ群は並行処理によって機能線が分岐していると解釈することも可能である。しかし機能線では，このようなサーバ群は一つのコンポーネントと捉える。また，実際の調査では認証サーバ群全体の管理ログから，認証に失敗したサーバは容易に特定でき，全ての認証サーバを精査する必要はない。さらに，全ての認証サーバを調査した場合にも，本研究で対象としている中・小規模システムでは膨大な調査量とはならない。

3.6　機能線を用いた障害原因部位特定手順のまとめ

機能線を用いた障害調査は以下の手順で進める。

（１）障害検知時のエラー情報から目的データを抽出する

（２）システム構成情報から目的データに関与するコンポーネントを抽出する

（３）抽出したコンポーネントを目的データの流れに沿って列記する

（４）補助機能線を用いて重複区間を調査対象から除外する

（５）始点から終点方向に向かい各コンポーネントでの障害発生有無を，後述する 4.3 節の比較法の適用手順に従い判定する

特定された部位情報が対策立案に不十分である場合は，コンポーネントを分解し再度（４）と（５）を実行する。対策立案可能な場合は，調査を完了する。

3.7　まとめ

オープン系システムで発生した障害を記述する機能線を提案した。これにより，人手で行われるオープン系システムの障害原因調査手法をモデル化した。機能線は，障害発生原因となった目的データに注目して，これに関与する要因であるコンポーネントをグラフ構造で表記したものである。機能線を用いることで障害を視覚的に捉えられ，かつ調査の方向性を明確にすることができる。

次に機能線での障害調査対象を削減する手法として補助機能線を提案した。これにより，障害調査対象システム内で障害が発生していない部分の情報を利用して，障害調査対象から除外可能な区間を特定し，調査対象を削減することができる。

機能線を用いることで障害の全体像を視覚的に把握し，さらに補助機能線を併用することで調査対象を効率的に絞り込むことができる。これによって，実際の障害調査の場面で用いられている調査手法を論理的に表現することが可能となった。

〔参考文献〕

[1] 工藤裕,森村知弘,菅内公徳,増石哲也,薦田憲久：「障害原因解析のためのルール構築方法と解析実行方式」, 電気学会論文誌Ｃ（電子・情報・システム部門誌）, Vol.132, No.10, pp.1689-1697 (2012)

[2] 永井崇之,名倉正剛：「迅速な危機回避を目的とする大規模環境向け障害原因解析システム」, 情報処理学会論文誌 Vol.54, No.3, pp1109-1119 (2013-03)

[3] 情報処理システム高信頼化教訓のリンク集 (IT サービス編), T25 「原因不明の障害への対応に関する教訓」, http://www.ipa.go.jp/files/000057347.pdf, T25 (2017-11)

[4] SCSI 規格の ASC/ASCQ:

http://www.t10.org/lists/asc-num.htm#ASC_03, (2012-02)

[5] 篠原昭夫, 泉隆：「オープン・システム上での障害発生部位特定方法の提案」

第 13 回情報科学技術フォーラム, RO-010, 第４分冊 pp.75-80 (2014-09)

[6] John L. Hufferd: “iSCSI The Universal Storage Connection”, Addison Wesley, pp.52-54 (2003)

[7] iSCSI 規格 (RFC3270):

http://tools.ietf.org/html/rfc3720, section3.4 (2004-04)

第４章　　コンポーネントでの障害発生有無判定法

4.1　はじめに

前章において，障害原因部位特定に用いる機能線を提案した。本章では機能線を用いた障害原因調査で必要となる，コンポーネント内での障害発生有無の判定法として比較法を提案する。この比較法は，調査対象システム正常稼働時のログ情報を障害発生時に正常であったかの判定基準として用いる。これにより，各システムの運用特性を反映した障害原因調査を行うことが可能となる。

4.2　比較法

4.2.1　比較法の導入

オープン系システムを構成する製品は詳細仕様が非公開であり，単にログ情報を参照するだけではコンポーネントの正常，異常を判定することが難しい。そこで，対象システムの正常稼動時ログ情報を採取しておき，同じシステムのログ情報を障害発生時に採取したものと比較して，不一致部分があれば障害発生に関連したログ情報であると判定することを考える。本研究では，これを「比較法（comparison method)」と呼ぶ。

オープン系システムを構成する製品は，テキスト形式でログ情報を採取できるものがほとんどである。一方で，比較法はテキスト形式のログ情報を前提としているため，専用の解析ツールがなくても多くのコンポーネントに適用できる。さらに各システムでの正常稼働時ログ情報を基準に異常を判定する本比較法を用いれば，製品の詳細仕様が非公開であってもこれを補完できる利点がある。

4.2.2　比較法を使うにあたり事前採取するログ情報

比較法を使用するにあたりシステム正常稼働時に事前採取すべきログ情報を，以下に示す。

A: システム起動時の各コンポーネントからの報告

B: 各コンポーネントから得られるログ情報

C: OS のシステム・ログ

D: 対象システムが目的とするサービスの正常処理時間

E: 統計的に数値化可能な情報

A は，あるコンポーネントが障害発生時に自動的に再起動したかの判定に利用できる。一般に，正常起動時のログ・パターンの仕様は非公開であるが，A が事前採取されていることで障害時に再起動が発生したコンポーネントがあったかを判定可能となる。B と C はログ情報そのものである。D は処理遅延障害における正常処理時間の基準値となる。D がないと，パフォーマンス障害は比較対象がないため調査困難となる。また E は，正常稼働状態から逸脱した事象が発生したかを判定する際の基準値となる。

4.2.3　比較法を用いる場合の留意点

（1）ログ情報とコンポーネントの関連付け

OS のシステム・ログは，複数コンポーネントから個別に報告された情報がログ記録ソフトウェアに到着した順に保存される。このため採取されたログ情報を使用するには個々のコンポーネントとの対応付けが必要である。図 4.1 は OS のシステム・ログ中に，３つのコンポーネントのログ情報が混在している例である。このように OS レベルで収集されるログ情報では，対応コンポーネント別に分類する操作が必要となる。

図 4.1　システム・ログとコンポーネントの関係

（2）ログ情報記録機能のないコンポーネントへの適用

ログ機能のないコンポーネントには比較法を適用できない。しかし，他のコンポーネントから採取したログ情報，または統計的に数値化された情報の中に，ログ機能のないコンポーネントに関連する情報が含まれていることがある。この場合には間接的に対象コンポーネントのログ機能不足を補完することができる。

4.2.4 比較法の適用方法

比較法には，テキストマッチングを用いる。正常稼働ログ情報とマッチした記録を除去することで，残されたログ記録は調査対象の障害原因に関係しているものであると考えることができる。すなわち手順自体は正常稼働時に見られないパターンを抽出するという単純なものである。

マッチング対象は以下の通りである。

・反復する同一書式の記録

・システム起動時のバナー情報などの固定書式記録

・時刻情報から判断できる定期的なイベントの記録

図 4.2 の例を用いて比較法の適用について説明する。図中の①と②は，ともにシステム正常稼働時に記録されたログ情報である。この結果，③が障害に関係したログ情報として残され，この部分を調査すれば良いことになる。

比較法により正常が異常と判定されることはない。なぜなら異常発生の判断に用いるログ情報は，システム正常稼働時に採取されているためである。しかし，ログ機能の不具合によりシステム正常稼働中に採取したログ情報に，障害発生時にのみ記録されるべきログ情報が含まれている場合は，異常が正常と判定される。このような不具合には，システム正常稼動中に誤って記録された異常ログ情報を事前に抽出しておくことにより対応する。

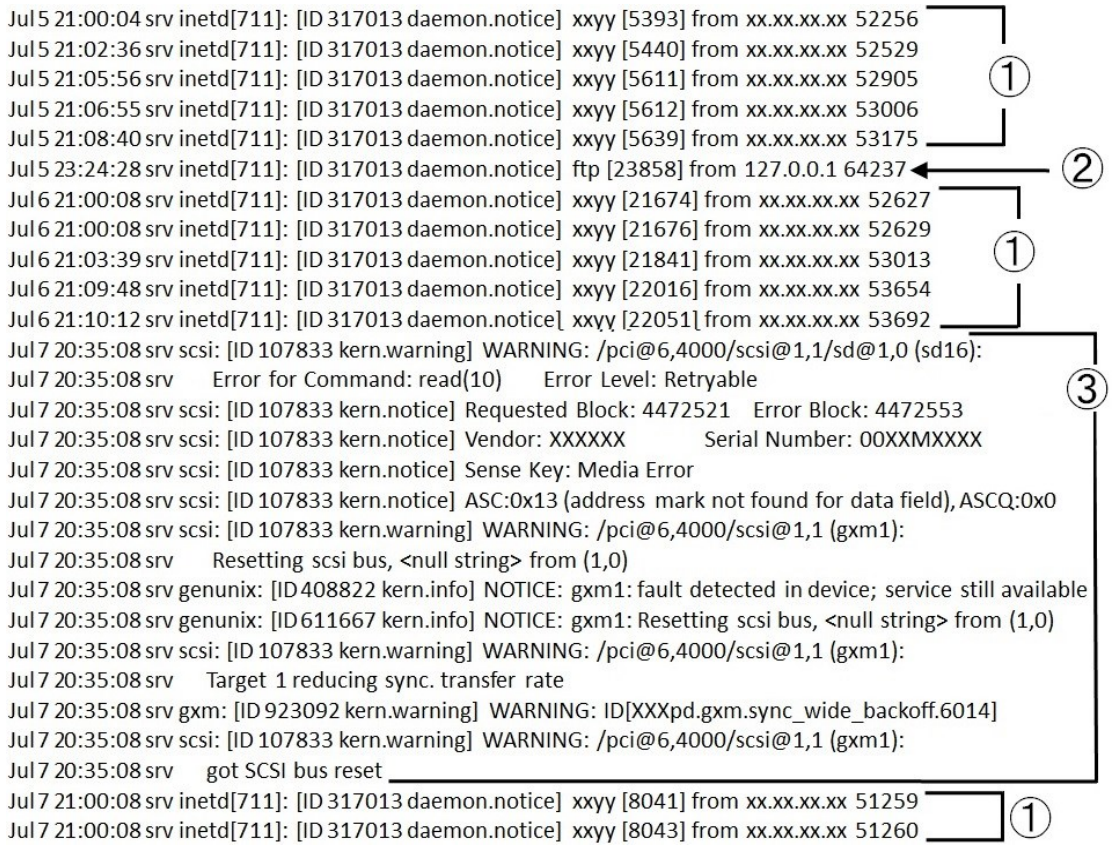

図 4.2　障害発生時ログの例

4.3　比較法適用手順のまとめ

比較法の適用手順をまとめると以下のようになる。

（1）システム正常稼動中のログ情報を事前採取する

（2）コンポーネントとログ情報の関連付けを行っておく

（3）ログ機能不具合で記録されるログ情報を解析しておく

（4）障害発生時に（1）と同じ手順でログ情報を採取する

（5）正常稼動中には見られないログ情報から，各コンポーネントでの障害発生有無を判定する

4.4 まとめ

提案した比較法は，機能線による障害原因調査手法を補完する。すなわち，機能線は障害原因調査の方向性を明確にするとともに，補助機能線により調査対象を絞り込む。これに対して比較法は，システム正常稼働時のログ情報を利用して，機能線上に存在する各コンポーネントでの障害発生有無を効果的に判定する手法である。両者を併用することで障害原因発生部位特定を容易に実現することができる。

本比較法を用いることで，製品毎に記録方式が異なる場合や、詳細仕様が非公開であっても，ログ情報から障害発生原因に起因した痕跡情報のみを見つけ出すことが可能となる。

〔参考文献〕

[1] 篠原昭夫，泉隆：「オープン・システム上での障害発生部位特定方法の提案」，第 13 回情報科学技術フォーラム，RO-010，第４分冊 pp.75-80（2014-09）

第５章　　機能線を用いた障害原因調査の効果およびFTAとの比較

5.1　はじめに

障害事例の分析には文献[1][2]による調査，教訓集などがある。しかしこれらは，システム管理者の視点で分析されているため機能線有効性検証には不向きである。そこで本章では，障害コールセンターの障害記録1998件を調査した。記録にはコール受付け時点でユーザにより障害原因部位特定が完了していた1822件が含まれており，これを除外した。すなわち，ユーザから提供されたコール時の情報のみでは障害原因部位の特定が困難である障害記録176件を抽出し調査対象とした。これらの障害記録を用いて，機能線の適用可能率，障害原因部位の特定可能率，比較法の適用可能率，補助機能線の作成可能率を検証する。

次に，人手によりシステム障害の原因部位調査を行う手法として，機能線に最も近いと考えられるFTA(Fault Tree Analysis: 故障木解析)[3]との比較を行う。

5.2　検証結果

検証は，調査対象の各障害記録に機能線を用いた手法を適用して再度障害原因部位特定を実施し，機能線を用いない場合との調査内容を比較することにより行った。はじめに、以下の（A）から（D）について分析した。

（A）機能線の作成が可能かどうか

（B）障害原因部位が特定できて対策立案可能かどうか

（C）比較法を適用可能かどうか

（D）補助機能線の作成が可能かどうか

表5.1　障害の種類と件数

障害の種類	件数
ハードウェア／ソフトウェアの問題	145（82.4%）
ネットワークの問題	16（9.1%）
クライアント側の問題	15（8.5%）

表 5.1 に調査対象障害の種類を，表 3 に機能線による手法の有効性検証結果を示す。表 5.2 の(A)から，機能線作成の成功率が高いこと，表 5.2 の(B)から約 3 割の障害で原因部位特定に至っていないことが判明した。表 5.2 の(B)で障害原因部位特定に失敗した 47 件のうち，38 件はログ情報不足が原因であった。この内訳は，ログ情報の未採取，ラップ上限数超過による上書きのためのログ情報消失，保安上の制約によるログ情報の提供拒否などであった[4][5]。残る 9 件は，調査対象の障害に関連するログ情報の記録機能がないためであった。

表 5.2 の(C)で比較法が適用できなかった 33 件は，正常稼働時のログが採取されていなかったことが原因である。表 5.2 の(D)で補助機能線作成率が低い理由は，調査対象中にネットワーク系，複数クライアント系の障害記録が少なかったためである。

表 5.2 機能線の適用効果

評価項目	成功/ Failed	成功率
(A) 機能線作成可否	成功: 161 失敗: 15	91.4%
(B) 障害原因部位特定に成功	成功: 129 失敗: 47	73.2%
(C) 比較法の適用可否	成功: 143 失敗 33	81.3%
(D) 補助機能線の適用可否	成功: 12 失敗: 164	6.8%

5.3 機能線適用効果の評価

表 5.3 は，機能線適用による調査コンポーネント数削減効果を示したものである。機能線未適用の場合に調査が必要であったコンポーネント総数が 274 であるのに対し，機能線を適用した場合のコンポーネント総数は 224 であった。機能線を適用することで，システマティックに障害調査を行うことができるとともに，約 2 割の調査対象コンポーネントが削減できた。

表 5.3 機能線に適用による調査コンポーネントの削減効果

機能線使用／未使用	調査対象コンポーネント数
機能線使用	224
機能線未使用	274

5.4 FTA による障害原因部位調査手法との比較

本節では，人手で行う既存の障害原因部位特定手法の中で機能線に最も近いと考えられる FTA との比較，検討を行う。

実際の障害調査では，調査を実行する技術者の習熟度が原因部位特定に要する時間に大きく影響する。従って，この条件下で客観的な比較を行うことは困難である。そこで本章では，機能線と FTA について，調査対象の個数，情報量の差，対策立案の容易さにより比較する。

システム内で発生した障害の調査に広く使用される FTA は，障害の要因となる事象を FT 図と呼ばれる木構造形式に展開し障害原因部位を特定する手法である。この手法は機能線と同様，ハードウェアやソフトウェアなどの実装形態を意識する必要がない。そこで提案した機能線の，FTA との対応について検討する。

FTA では障害となった事象を，第一次要因，第二次要因へと順次展開してゆくが，この過程は機能線では分解レベルを上げることに対応する。すなわち FTA の各要因は，機能線でのコンポーネントと類似している。さらに機能線は分岐のないグラフ構造であるため，AND ゲートを持たない。すなわち，機能線は OR ゲートのみで構成された FT 図に近いものと考えられる。図 5.1 は，以上に述べた機能線と FT 図との対応を示したものである。

FT 図では同一 OR ゲート内に属する要因の間に因果関係の情報はない。一方で，機能線上のコンポーネント間には，目的データ転送を通して因果関係の情報が含まれている。つまり機能線では FT 図の OR ゲート内に属する各要因に，目的データに関連した因果関係の情報が付加されている。

３章で述べたように一つの機能線上に存在する障害原因部位は，多重障害の場合を除き一か所である。また分解レベルを上げる必要があるのは，より詳細な調査結果が必要な場合のみである。一方で，FT 図では発生確率の合計が１になるまで要因の全てをトレースする必要があるが，機能線では目的データに関与するコンポーネントのみを調査対象とする。以上の理由から機能線では FT 図よりも調査対象数が少ない。さらに，機能線にはコンポーネント間の因果関係の情報が含まれているため，障害原因部位特定後の対策立案が容易になる。

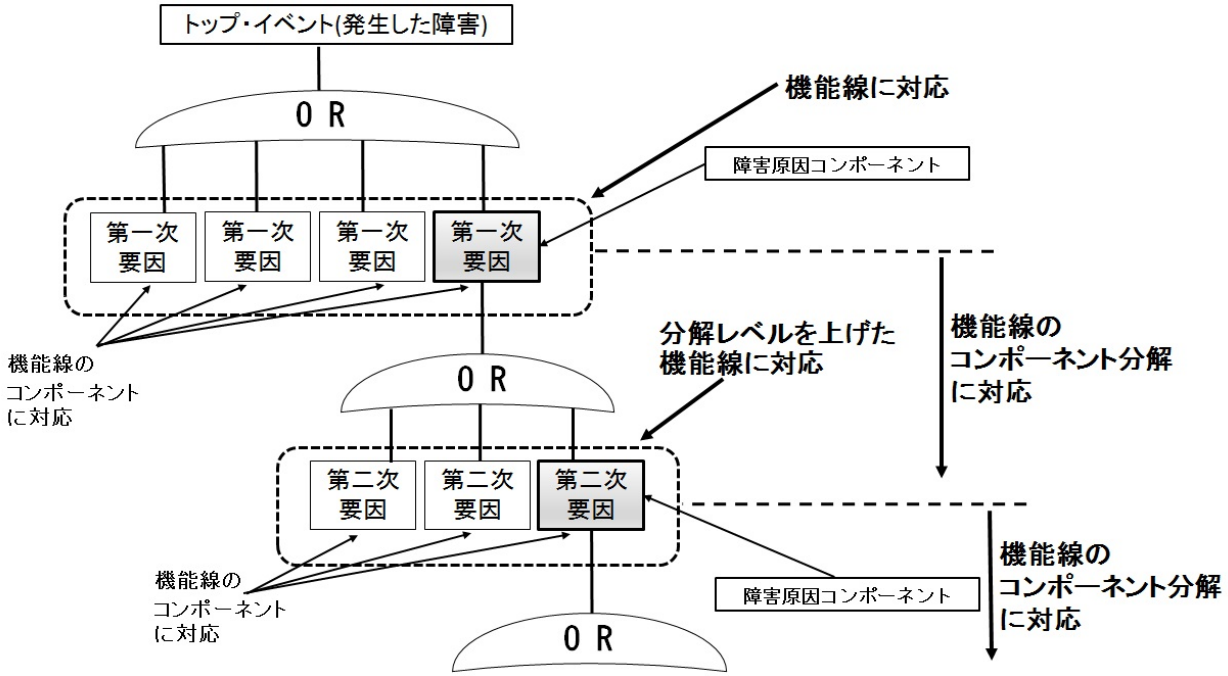

図 5.1　機能線と FT 図の対応

5.5　まとめ

本章では障害コールセンターの障害記録を調査して，実際の障害調査事例に機能線を利用した分析を行い，障害原因部位調査での効果を評価した。はじめに，機能線の適用可能率，障害原因部位の特定可能率，比較法の適用可能率，補助機能線の作成可能率を検証した。次に機能線を適用していない場合に調査を行う必要があったコンポーネント数と，機能線を適用した場合の調査コンポーネント数の比較を行い，機能線適用の効果を評価した。結果として，機能線を利用することで調査対象コンポーネントの約２割を削減できることを示した。

さらに機能線と FTA との比較を行い，機能線は FTA よりも障害に関係した情報を多く含むことと，調査対象数が少ないことを示した。

〔参考文献〕

[1] IPA 独立行政法人 情報処理推進機構，海外における IT 障害の影響及び 対応策に関する事例調査 - 報告書，http://www.ipa.go.jp/files/000026797.pdf，障害事例集，pp.1-29（2013-04）

[2] IPA 独立行政法人 情報処理推進機構，重要インフラ障害情報の分析に基づく「情報処理システム高信頼化教訓集（IT サービス編）」～障害の再発防止のため，業界を越えて幅広く障害情報と対策を共有する仕組みの構築に向けて～，

http://www.ipa.go.jp/sec/reports/20140513.html，pp.I-18-39，（2014-05）

[3] Larsen，Waldemar. “Fault Tree Analysis”. Picatinny Arsenal. Technical Report 4556，(January 1974)，
http://www.dtic.mil/dtic/tr/fulltext/u2/774843.pdf，(Retrieved 2014-05-17)

[4] 篠原昭夫，泉隆：「オープン・システム障害対応の現状分析」,情報処理学会 第 77 回全国大会，3ZE-05（2015-03）

[5] 篠原昭夫，泉隆：「オープン系システム保守の現状報告」,情報処理学会 第 76 回全国大会，4ZE-4（2014-03）

第６章　　機能線を用いた障害分類と対策立案

6.1　はじめに

本章ではオープン系システムの障害について，障害対策実行時に発生する二次障害を防止または対策を容易にする方法を提案する。

オープン系システムで発生した障害の原因調査と対策実施は，障害案件毎にユーザ自身またはシステムサポートベンダにより行われている。またその障害記録や調査結果は外部に公開されることがほとんどなく[1]，障害事例を統一した視点で論理的に分析する試みはなされていない。さらに障害対策実施中に発生した二次障害への対応も，前章で調査した障害コールセンターの障害記録データによると，発生原因調査を含めて一次障害案件の対応範囲に含まれた形で扱われている。

一方で，二次障害の発生はシステムのダウンタイムを長期化させる，障害対策が不完全な状態でのシステム運用期間を長くするなどの原因となる。また二次障害の一つである DOA(*1)の発生は，実施した対策方法が有効でないと誤判断する原因となり，追加対策方法の策定を困難にする。

以上の理由から二次障害を防止する，および DOA への対処を容易にする方法が求められている。

前章までに筆者らが行った障害事例の分析に際し，障害はいくつかの障害型に分類可能であることが判明した。さらにこの中で，不適切な対策方法が選択されたために二次障害が発生する場合があることが分かった。そこで，各障害型に最適な対策方法を関連付けることで，二次障害発生抑止または二次障害の対処を容易にできると考えた。

本章でははじめに二次障害の発生原因を分析し，防止可能または対処を容易にできる二次障害の種類を明らかにする。この結果から対象とする二次障害を，不適切な対策方法の選択により発生するもの，および DOA の２つとする。

つぎに階層構造と機能線を用いて，異なる障害同士を比較可能となるよう障害を分類する。この分類結果から３種類の障害型を定義し，最後に各障害型に最適な対策方法を関連付ける。なお，ここで提案する対策方法は，DOA の発生を考慮したものである。

*1 DOA (Death on Arrival)：　交換作業で投入した新品部品に障害があること

6.2　障害対策方法立案時の課題

オープン系システムの多くは中・小規模なものである。この規模のシステムは，障害発生から対策完了までをユーザとサポートベンダ間でのみで対応する。このため，障害情報が外部に公開されることが稀であることが特徴である。障害原因調査と対策実施は案件毎に行われ，統一した視点で障害事例を論理的に分析する試みがなされていない。また対策実施にあたりDOAなどの二次障害が発生することがあるが，DOA発生を判断する方法は確立されていないため，対策作業自体が完遂困難になる事例が見受けられる。さらに発生した障害に対して不適切な対策方法が選択される事例が散見され，このことも二次障害の発生原因となっている。

以上の問題の解決を目的として次の２つを提案する。

(1) 障害原因調査結果から適切な対策法を割り当てる手法

(2) DOA発生を判断する方法

(1)に対しては障害を障害型で分類し，それぞれに最適な対策方法を割り当てることで，不適切な対策方法が選択されることを抑止する。(2)に対しては，(1)で割り当てる各対策方法にDOAの発生を判断する方法を組み入れることで実現する。さらに(1)によりDOAが誤検出されることを防止できる場合があることにも言及する。

6.3　障害分類の事例

システム障害を分類する試みは，文献[1][2][3]などに見られる。文献[1]における分類は，システム管理，ユーザが障害発生により受ける影響などの視点のみで行われており，障害自体の論理的構造には言及していない。また文献[2]では，障害事例の原因について技術的な部分に触れた分析を行っているが，異なる障害同士の論理的構造を直接比較するまでには至っていない。一方で，文献[3]は特定の製品で発生する可能性のある障害を予測し，さらに障害の分類を試みている。しかし対策方法への関連付けは行っていない。

上記のように，障害を障害型に分類してこれに最適な対策方法を関連付ける試みは見あたらない。

6.4　二次障害発生原因の分析

二次障害の発生原因について分析し，対象とする二次障害を明確にする。二次障害が発生する原因には以下が考えられる。

(1) 障害対策で変更する部位が他に及ぼす影響の考慮不足によるもの

(2) 作業誤りによるもの

(3)　DOA によるもの

(4) 不適切な対策方法の選択によるもの

なお(1)と(4)は，(1)は対策方法自体に誤りはないが対策実施によってシステム内の他部位で新規に障害が発生するもの，(4)は選択した対策方法自体が発生した障害に対して不適切である点が異なっている。

(1)と(2)は共に人為的要因であるために，論理的な分析が困難である。そこで，本章では(3)と(4)を対象とする。

6.5　DOA と Regression

6.5.1　DOA による二次障害

DOA（Death on Arrival）とは，ハードウェア障害で交換した新品部品に障害があることをいう。DOA により発生する二次障害は，内容により２種類に分けられる。

(1) 発生した二次障害が交換前と同じである場合：

交換した部位が障害発生部位でないと誤判断する原因となる。これは，交換した部品には問題がないと暗黙に仮定しているため，対策実行箇所が元々の障害の原因部位ではなかったと誤判断してしまうことによる。

(2) 発生した二次障害が交換前と異なる場合：

交換した部品が，対策実行前の障害と異なる問題を引き起こしている場合であり，対策結果の判断を誤る原因となる。

DOA の発生は実施した対策方法の効果を誤判断する原因になる。また，DOA を誤検出することは，その後の対策実施を困難にする。したがって，DOA の発生抑止に加えて，DOA の発生を正しく検出する方法が求められている。

6.5.2 Regression と DOA 定義の拡張

Regression とは，修正を行ったソフトウェアで修正対象と同一または新規の問題が発生することを言う。ここで発生する問題の内容は，ハードウェアでの DOA と異なるところがない。一方で，本章で障害分析手段に用いる機能線では，ハードウェアとソフトウェアの違いを意識する必要がない。そこで，DOA の定義をソフトウェアにまで拡張する。

[定義]「DOA」とは，ハードウェアとソフトウェアの違いに関係なく，障害対策により変更を実施した部分で障害が発生することをいう。

以降 DOA には，この定義を用いる。

6.5.3 DOA 発生率の推定

DOA の発生率を製品ベンダが公開することはないため，その把握は困難であるがここでは推定を試みる。図 6.1 は DOA 発生原因の分類である。

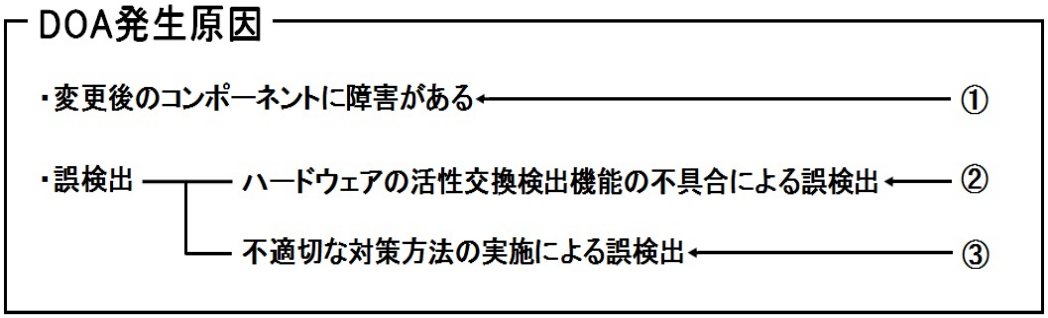

図 6.1　DOA 発生原因の分類

筆者らの経験では，図 6.1 中の①～③の割合は，①が数%，②が約 90%，③が数%である。①のうちハードウェア DOA は，ハードディスクドライブの故障率調査[4]によると，使用開始後 3 か月以内の障害発生率では 3%以下である。交換可能な単位の部品自体に注目した DOA 率を調査した結果[5][6]によれば，その発生率は 1～2%前後であると考えられる。DOA を使用開始から極めて短期間に発生する障害と捉えれば，その発生率は 3%より低いと考えられる。

次にソフトウェア DOA は，ソフトウェア修正作業時の Regression 率に言及した文献[7][8]によると 5～6%程度である。ユーザがソフトウェア DOA を検出するのはベンダからリリースされた後の製品であるため，その発生率は 5%よりも低いと考えられる。以上から①の発生率は 1～2%程度と推定する。

③の発生率は筆者らの経験では 1%程度である。

以上をまとめると，実際の DOA 率は 1～2%の範囲であると考える。

6.5.4 活性交換による DOA 誤検出

図 6.1 中で，②の活性交換検出機能の不具合による DOA 検出は，DOA 発生全体に占める割合が非常に高く 90%程度である。システムを停止せずに部品を交換する要求が高まるのに伴い，活性交換(*2)可能なハードウェア部品点数は増加している。しかし，活性交換が行われたことを自動検出する機能に不具合が内在した製品が多く，結果として活性交換件数の増加に比例して②の割合が高くなっている。

*2 活性交換(Online Replacing, Hot Plugging)： 装置稼働中に部品を交換すること

6.6 障害原因部位の個数による分類

機能線を使うと，障害は一本のグラフとして視覚的に記述される。この性質を用いると，異なる障害同士を比較することが容易となる。そこで本節では，機能線を障害の論理的分析に用いる[9]。

はじめに，図 6.2 に示す A から D の４つの階層を定義する。各階層では，機能線を用いた分析の妨げとなる要因を除外してゆく。すなわち，階層 A から C を通して論理的に分析が困難な障害を順次除外する。この結果，階層 D に属する障害は全て障害発生部位が一か所である単純障害の集合となり，分析が容易になる。

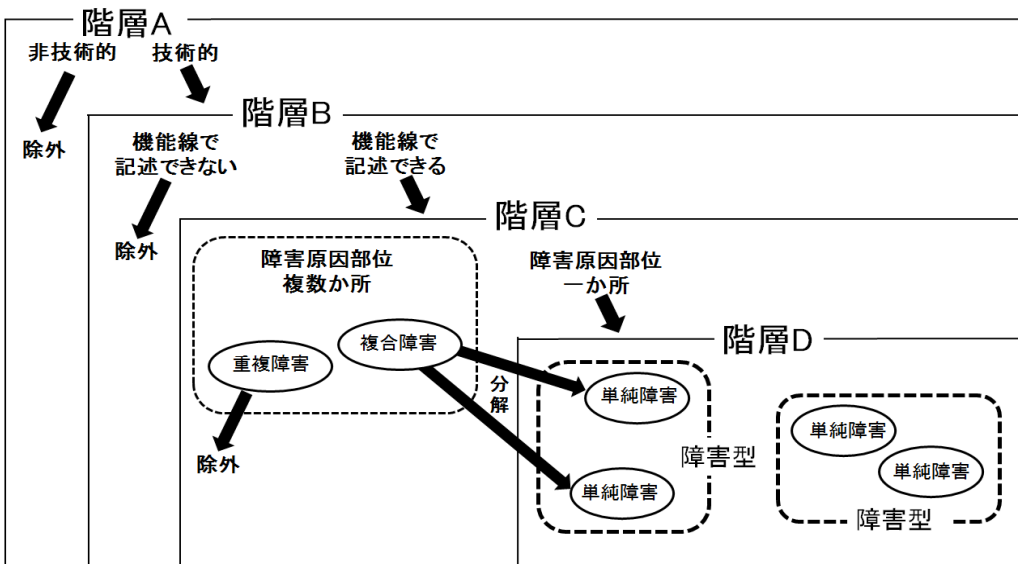

図 6.2　障害の分類階層

各階層における分類は以下の通り。

[階層 A]

障害が技術的または非技術的であるかによる分類。この階層に属する障害は，オープン系システムで障害発生が報告されたもの全てが対象となる。階層 A から非技術的な障害を除外した残りを階層 B とする。

[階層 B]

障害が機能線により記述可能であるかによる分類。この階層に属する障害は，技術的な視点で取り扱えるもの全てである。階層 B から機能線で記述できない障害を除外した残りを階層 C とする。

[階層 C]

検出された障害原因部位の個数と，障害部位の依存関係による分類。この階層に属する障害は，機能線により障害発生部位が記述できる障害全てである。階層 C から単純障害でない障害を除外した残りを階層 D とする。

[階層 D]

単純障害のみの分類。階層 A から C までの分類の結果，階層 D に属する障害は単純障害のみとなる。6.7 節では階層 D を分析し，単純障害の全てを 3 種類の障害型に分類する。

6.6.1 階層 A： 非技術的な障害の除外

階層 A では，障害が技術的または非技術的であるかにより分類する。非技術的な障害には人為的要因が含まれているため，論理的に取り扱うことが困難である。そこで，論理的に分析が行えないものを除外する。

階層 A での障害分類基準は，次の(1)から(3)である。

(1) 必ずしも製品の不具合や故障などが発生していないが，システム利用者が

問題と指摘する障害　（例）パフォーマンス障害

(2) 作業誤りによって発生する障害

(3) 製品の不具合や故障などの問題が表面化する障害

(1)には利用者の主観が，(2)には人為的な要因が含まれるため，非技術的な障害に分類する。残る(3)は技術的要因による障害である。

次の階層 B では，(1)と(2)を除外し(3)のみを対象とする。

6.6.2 階層 B： 機能線で記述可能かによる分類

階層 B では，障害が機能線により記述可能であるかにより分類する。

前章で述べた障害事例分析結果では，約 80%の障害が機能線により記述可能であった。

次の階層 C では，機能線により記述可能な障害のみを対象とする。

6.6.3 階層 C： 障害発生部位の個数と依存関係による分類

階層 C では，検出された原因部位の個数と原因部位の依存関係により分類する。この階層に属する障害は次の(1)から(4)が考えられる。

(1) 単純障害

機能線上の障害発生部位が一か所のみの障害を単純障害と呼ぶ。図 6.3 に機能線で記述された単純障害の例を示す。この障害は最も多く発生する形態である。

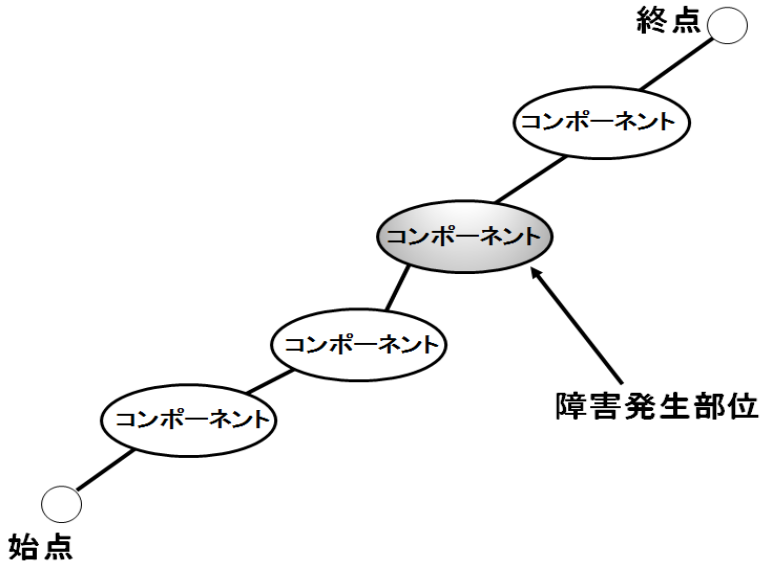

図 6.3　単純障害の例

(2) 複合障害

複合障害とは，互いに依存関係にない複数の単純障害が，システム内で同時または同じ期間に発生したものである。図 6.4 に機能線で記述した複合障害の例を示す。この障害は，図 6.4 中の単純障害１と単純障害２のように，独立した複数の障害に分解できる。これによって，複数の単純障害として個別に扱うことができる。

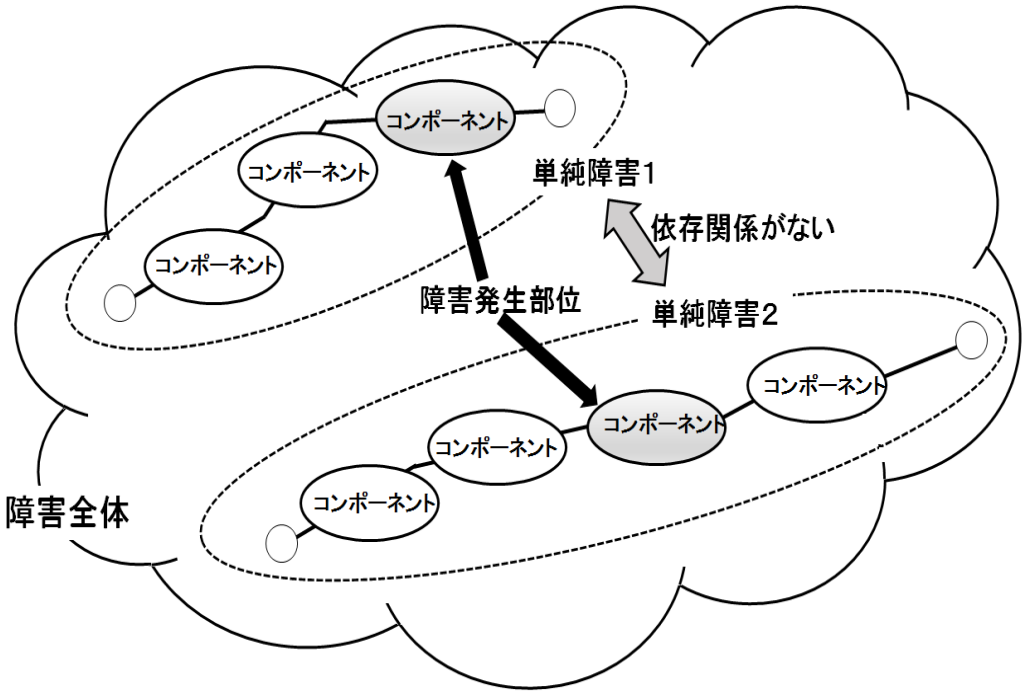

図 6.4　複合障害の例

(3) 重複障害

重複障害とは，機能線の発動点エラー報告の原因となる障害発生コンポーネントが，複数存在するものである。図 6.5 に機能線で記述した重複障害の例を示す。この障害は，複数のコンポーネントで独立して障害が発生し，かつそれらが同じ発動点エラー報告の原因となる場合である。なお，重複障害の発生確率は極めて低いため，本章では対象としない。

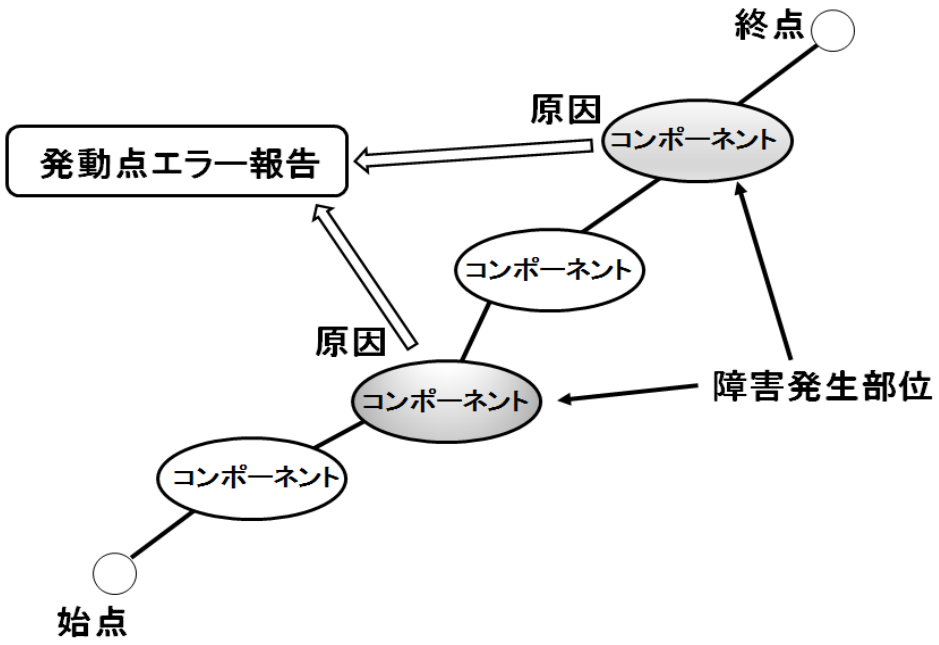

図 6.5　重複障害の例

(4) (1)から(3)以外の障害

(1)から(3)以外の障害は，次の場合が考えられる。

・複合障害が同時に複数発生する

・重複障害が同時に複数発生する

・複合障害と重複障害が同時に複数発生する

筆者らの障害事例調査では，これらの障害の存在は確認できていない。このため本章では対象としないが，(3)の重複障害も含めて研究する必要があると考える。

6.6.4 階層 D： 単純障害のみが属する階層

階層 D に属する障害は，単純障害のみである。多くの障害は階層 D に属する。

6.7　単純障害の分類

前節までに行った障害分類の結果，階層 D に属する障害は障害発生部位が 1 か所である単純障害のみに絞り込まれた。本節ではこれに続き，単純障害の分類を行う。具体的には，階層 D に属する障害を次の 3 種類の障害型で分類する[7]。

・通常型障害　　：障害発生部位が一か所の障害

・両端決定型障害　　：障害発生部位は一か所に特定できない障害

・タイムアウト型障害　：タイムアウトに関連して発生する障害

これらの障害型は，筆者らが障害事例を分析することで抽出したものである。なお，現在までにこれ以外の型の存在は確認されていない。

以下に，機能線で記述された 3 種類の障害型とその具体例を説明する。

6.7.1　通常型障害

「通常型障害」とは，障害発生部位が唯一に判定できる最も一般的なものである。図 6.6 は機能線で記述された通常型障害の例である。

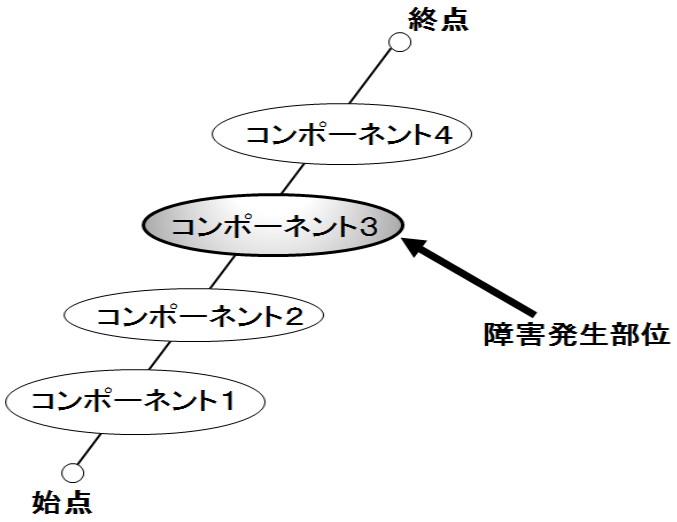

図 6.6　通常型障害の例

図 6.7 に通常型障害の具体例として，ディスク・ドライブに読み出し不可ブロックがある障害を示す。通常型障害では，障害発生コンポーネントの分解レベルを上げたとしても検出される障害発生部位は一か所のみである。

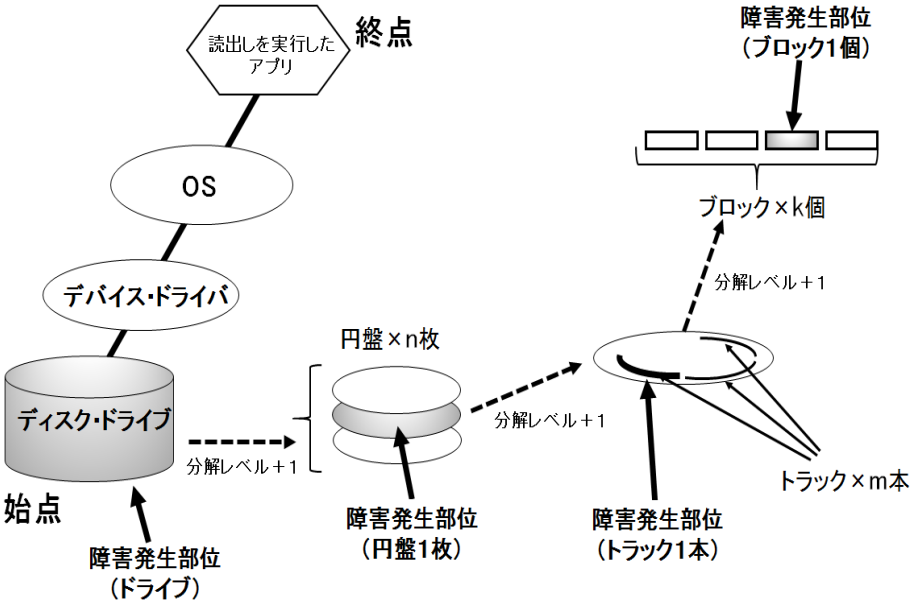

図 6.7　通常型障害の具体例

6.7.2　両端決定型障害

「両端決定型障害」とは，被疑コンポーネントが隣接する２つまで絞り込まれたが，そのどちらで障害が発生したかを判定できないものである。

図 6.8 は機能線で記述した両端決定型障害の例である。コンポーネント３－１と３－２はともに被疑部位であるが，どちらが障害発生部位かを判定できない。

図 6.8 に示すように，この型は分解レベルが低い場合には通常型障害に分類される。しかし，コンポーネント３が原因部位であるという調査結果のみでは対策立案には不十分なために，分解レベルを上げる。その結果，両端決定型障害に分類されたものである。なお，この状況が起こらないものが通常型障害に分類され，分解レベルを上げた場合にも障害発生部位が判定できる。

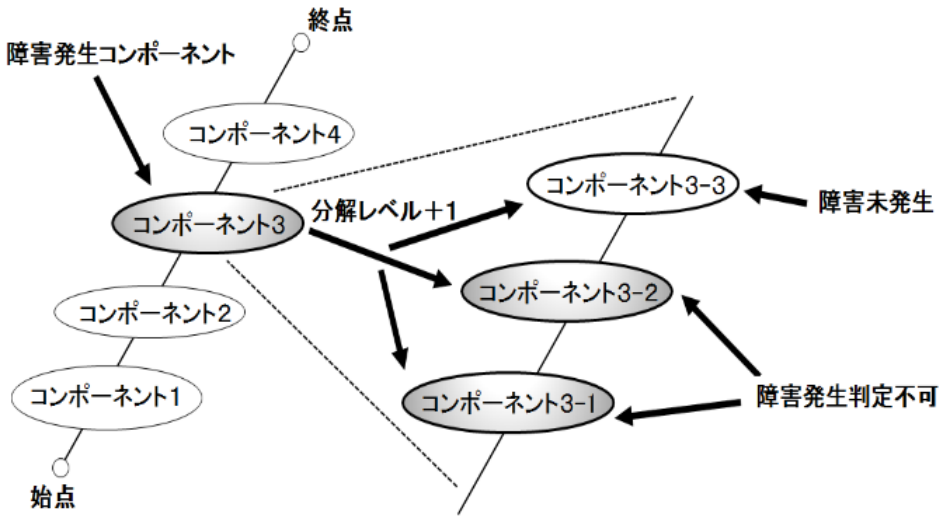

図 6.8　両端決定型障害の例

図 6.9 に両端決定型障害の具体例として，Fibre Channel 接続の光転送で 8B/10B 変換[10][11][12]の受信パターン・エラーが検出された障害を示す。図中で分解レベル N の状態ではコンポーネント３が障害発生部位で，これは通常型障害である。しかしコンポーネント３の分解レベルを上げると，SFP(*3)(コンポーネント３－１)はログ機能がないため，障害発生部位が SFP の出力部エラーであるか，8B/10B 変換モジュール（コンポーネント３－２）の受信部エラーであるかを判定できない。

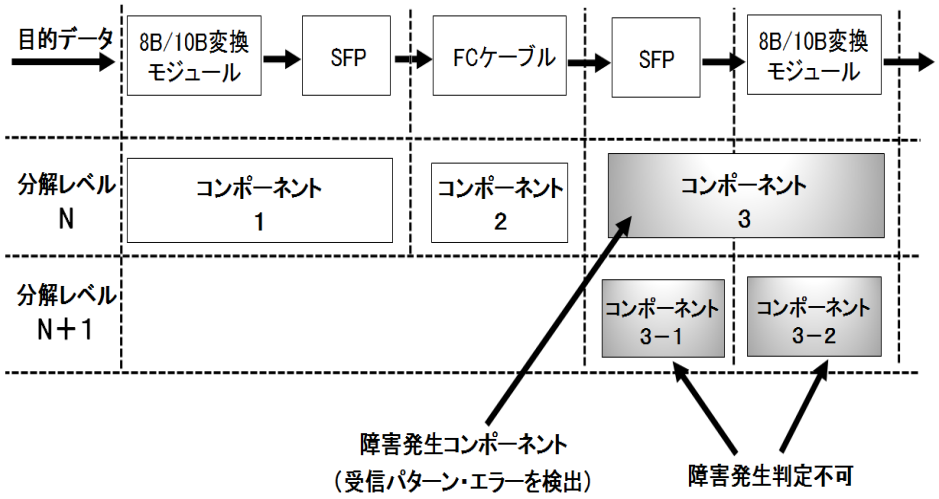

図 6.9　両端決定型障害の具体例

*3 SFP (Small Form Factor Pluggable)：　　電気信号と光信号の変換モジュール

6.7.3　タイムアウト型障害

「タイムアウト型障害」とは，あるコンポーネントが他のコンポーネントで障害が発生したと記録しているにも関わらず，当該のコンポーネントでは障害が発生していないものである。この型の障害は，殆どの場合タイムアウトに起因して発生するため，この名称とした。図 6.10 は機能線で記述したタイムアウト型障害の例である。

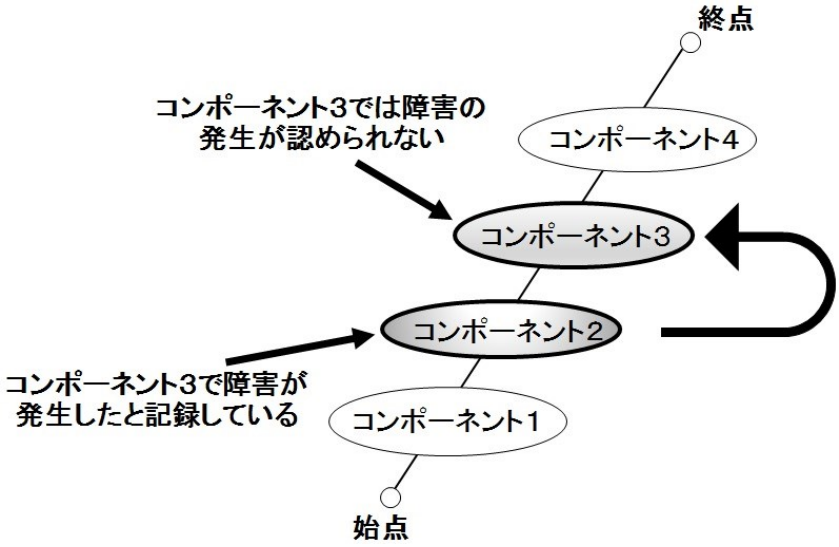

図 6.10　タイムアウト型障害の例

図 6.10 の障害の原因は障害ログ情報を生成したコンポーネント２の論理的不具合で，製品やシステムの設計不良である。この障害は通常型障害に分類することもできる。しかし障害発生部位がコンポーネント２であることを判定するには，コンポーネント２に障害記録があるコンポーネント３も調査し，これに問題がないことを確認する必要がある。このため，通常型障害とは異なる障害型に分類する。

タイムアウト型障害は，複数のコンポーネントが異なるベンダから提供され，コンポーネント間の連携確認不足から発生する。このことは複数ベンダの製品で構成されるオープン系システムの特徴でもある。

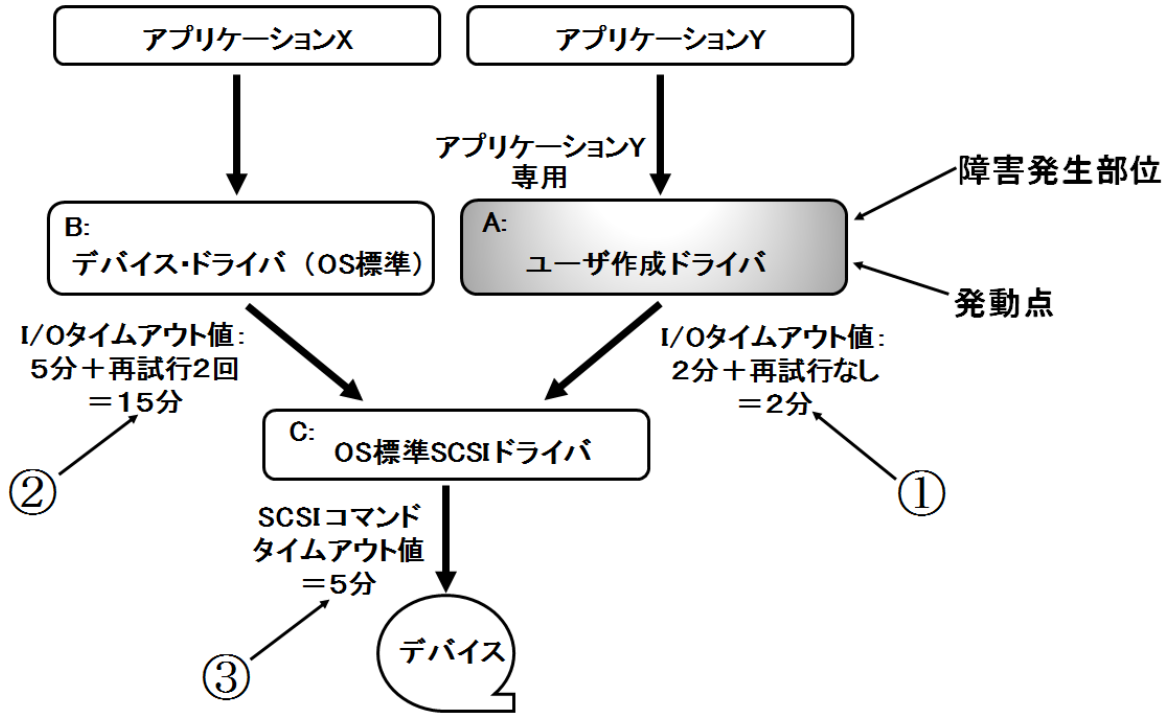

図 6.11　タイムアウト型障害の具体例

図 6.11 にタイムアウト型障害の具体例を挙げる。図中で，アプリケーション Y を発してデバイスに至るまでの経路が調査対象の機能線である。

デバイス・ドライバ B の I/O タイムアウト値 15 分に対し，A での値は 2 分と短い。しかし A はこの下位層で C を流用するため，ここでタイムアウト判定値の逆転状態が生じる(①＜③)。すなわち上位層の A で検出されたタイムアウトが，下位層の C ではタイムアウトでない状態となる。これは①の値と③の値の連携がなく，それぞれ独自に決定されているためである。なお，この例ではアプリケーション X を発してデバイスに至るまでの経路は補助機能線である。

この型の障害では，機能線での発動点がタイムアウトを生成したコンポーネントになる（図 6.11 で A は発動点かつ障害発生部位となる）。

また，この型の障害はタイムアウト値の設定誤りによっても起こるため，製品自体が不具合を持つ割合よりも多く発生する。したがって，タイムアウトを検出する仕組みを持つコンポーネントは，検出後の処理と障害調査を考慮して設計されるべきである。

6.8　各障害型に対する対策方法

障害が属する障害型を判定した後は，その障害型に最適な対策方法を割り当てる必要がある。そこで，各障害型に適切な対策方法を関連付けることで，不適切な対策方法が選択されることを抑止する方法を提案する。提案する対策方法では，DOA の発生検出を容易にすることと誤検出防止が考慮されている。

6.8.1　通常型障害の対策方法

通常型障害では，障害発生コンポーネントのみに対策すればよい。ここでは，対策するコンポーネントが周辺に及ぼす影響をも考慮しつつ変更する必要がある。そして，対策実施後に同じ障害が発生，または新たな障害が発生した場合には，対策実施コンポーネントを DOA と判断する。

6.8.2　両端決定型障害の対策方法

両端決定型障害では，同時またはいずれかの被疑コンポーネントに対策を実施するかにより２通りの対策方法が考えられる。

(1) 同時投入法

２つの被疑コンポーネントを同時に対策する。これにより，どちらのコンポーネントが障害発生部位かを特定することはできないが，一回の作業で対策を完了できる利点がある。

(2) 順次投入法

被疑コンポーネントの一方のみに対策して経過観察を行い，障害が解決しなければ，次に他方のコンポーネントを変更する。順次投入法は同時投入法よりも対策完了までに時間を要するが，障害発生部位を特定できる利点がある。

[障害型誤認による DOA 誤検出の具体例]

両端決定型障害を通常型障害と誤判断する場合について述べる。

図 6.9 で分解レベルが N の状態で調査を終了すると，通常型障害と判定される。ここで対策方法としてコンポーネント３－２のみを変更すると，DOA 誤検出の二次障害発生に繋がる。この例では，コンポーネント３全体を変更するのが正しく，分解レベル N+1 で現れるコンポーネント３－２のみの変更は不適切な対策方法となるためである。

6.8.3　タイムアウト型障害の対策方法

タイムアウト型障害では，タイムアウト・エラー記録を生成したコンポーネント，またはタイムアウト・エラー記録が示す被疑コンポーネントのどちらに対策を実施するかにより２通りの対策方法が考えられる。

(1) タイムアウト・エラー記録を生成しているコンポーネント（図 6.10 ではコンポーネント２）に対策する。

通常型障害と同様に，対策実施後に同じ障害が発生，または新たな障害が発生した場合には，対策実施コンポーネントを DOA と判断する。

(2) タイムアウト・エラー記録を生成しているコンポーネントが，エラー発生源と記録しているコンポーネント（図 6.10 ではコンポーネント３）に対策する。

対策実施後に障害が発生した場合には，対策実施部位が DOA であるかないかの判断に注意する必要がある。

[障害型誤認による DOA 誤検出の具体例]

タイムアウト型障害を通常型障害と誤判断する場合について述べる。

図 6.11 の例において，アプリケーション Y がデバイスに対して I/O タイムアウトを記録している場合に，これをもってデバイスに問題があると判断し，デバイス交換を行うことは不適切な対策方法となる。この例ではユーザ作成ドライバのタイムアウト値が短いことが障害原因であり，デバイス交換は効果がない。さらにデバイス交換を実施した後にも同一障害が再発する可能性が残るが，これは交換したデバイスを DOA と誤判断する原因となる。タイムアウト型障害では，単一コンポーネントの調査結果のみから対策方法を選択することは，不適切な対策方法の選択に繋がる可能性がある。

6.9　提案手法の有効性検証

本節では，障害型に対策方法を一意に関連付ける手法の効果について述べる。

具体的には，不適切な対策方法を選択することへの有効性，DOA 対処への有効性および二次障害全体に対する有効性を検証する.

6.9.1　不適切な対策方法の選択防止への有効性

6.8 節で不適切な対策方法が選択されるのを抑止するために，障害を分類し各障害型に一意の対策方法を関連付けた。対象とする障害は，機能線で単純障害と判定できたものである。この結果，不適切な対策方法が選択されることを抑止できる。

6.9.2　DOA 発生対処への有効性

6.8 節で提案した各障害型の対策方法は，全て DOA 発生判断方法を含んでいる。一方で，提案手法の対象となる障害は必ず３種類の障害型に属している。このため本章で対象としている障害全てにおいて DOA 発生の判断が可能である。

さらに 6.8.2 と 6.8.3 で述べた不適切な対策手法実施により発生する DOA 誤検出は，提案した対策手法への関連付けにより防止することができる。

6.9.3　二次障害全体に対する有効範囲

提案手法で発生防止可能な二次障害と，発生判断方法が適用可能な DOA の範囲をまとめると図 6.12 になる。

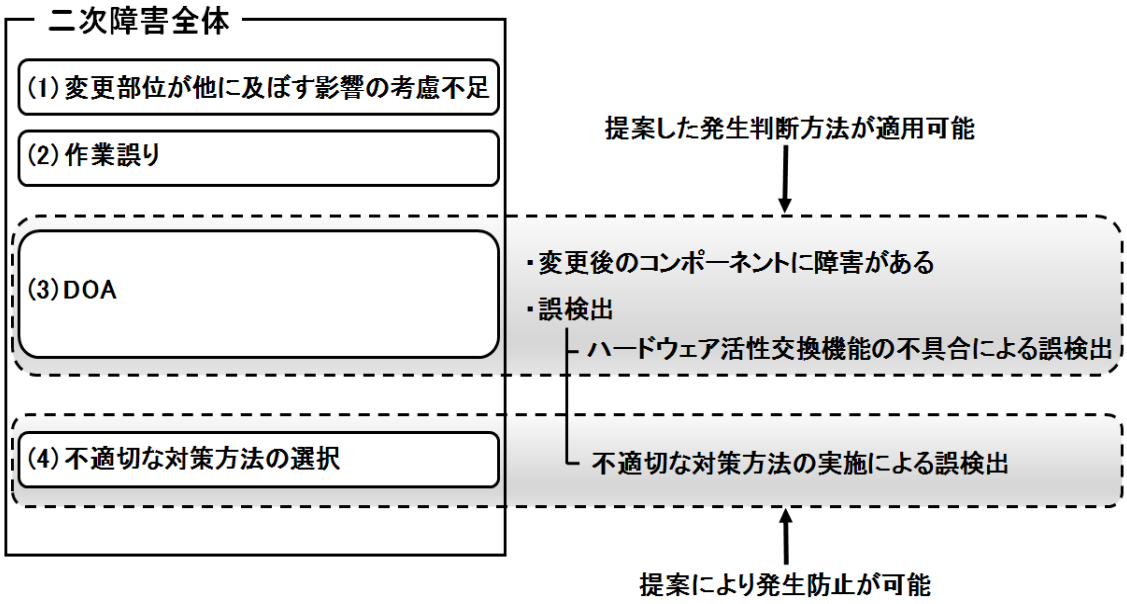

図 6.12　提案手法の有効範囲

6.10　まとめ

オープン系システムで発生する障害は，その発生件数が多いにも関わらず，統一した視点で分析する試みがあまりなされていない。これは，障害原因調査と対策実施がユーザまたはベンダで案件毎に行われるため，システム構成や使用製品に依存しない調査方法が研究されないためと考えられる。さらに障害事例をユーザが外部に公開することがほとんどないことも，統一した視点で障害を論理的に分析することを困難にしている。また対策実施に際して二次障害が発生することがあるが，その原因究明と対策方法はほとんど研究されていなかった。

本章では，オープン系システムで発生する障害を階層的に分類した。この分類結果から，対象を単純障害のみに絞り込み，機能線を利用することで単純障害を３種類の障害型に分類した。今後の調査により新しい型が発見された場合には，別途検討する必要がある。

次に，各障害型に最適な対策方法を関連付けした。これにより，オープン系システムの障害対策実施中に発生する二次障害の一つである，不適切な対策方法の選択を抑止できる。さらに不適切な対策方法と DOA 誤検出の関係についても述べた。

本章の結果をまとめると次のようになる。

・オープン系システムの障害を論理的に分析した

・障害発生部位が一か所である単純障害を分析，ここから共通の障害型を定義した

・不適切な対策方法が選択される原因を究明した

・各障害型に最適な対策方法の関連付けを提案し，不適切な対策方法が選択されることを抑止できることを示した

・不適切な対策方法と DOA 誤検出の関係を究明した

・DOA の定義をハードウェアのみから，ソフトウェアまで拡張し，両者の違いを意識せずに扱えるようにした

オープン系システムの障害対策方法立案では，修正プログラム（パッチ）の投入決定指針[13]までを考慮に入れる必要がある。また，本章では検討を行わなかった重複障害は，原因調査および対策方法立案に時間を要する事例が多いため，今後，引き続き分析・検討を行う必要がある。

〔参考文献〕

[1] IPA 独立行政法人 情報処理推進機構，海外における IT 障害の影響及び 対応策に関する事例調査 - 報告書，http://www.ipa.go.jp/files/000026797.pdf，障害事例集，pp.1-29（2013-04）

[2] IPA 独立行政法人 情報処理推進機構，重要インフラ障害情報の分析に基づく「情報処理システム高信頼化教訓集（IT サービス編）」～障害の再発防止のため，業界を越えて幅広く障害情報と対策を共有する仕組みの構築に向けて～，

http://www.ipa.go.jp/sec/reports/20140513.html，pp.I-18-39，（2014-05）

[3] 8.2. Failure classification:

https://access.redhat.com/documentation/en-US/JBoss_Enterprise_SOA_Platform/4.2/html/SOA_ESB_Programmers_Guide/SOA_ESB_

Programmers_Guide-_Fault_tolerance_and_Reliability_-_Failure_classification_.html, (2018-11 閲覧)

[4] Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz Andr´e Barroso:

“Failure Trends in a Large Disk Drive Population”,

http://static.googleusercontent.com/media/research.google.com/ja//archive/disk_failures.pdf p4 Fig-2, (2007)

[5] ハードウェア単体部品の DOA 率：

https://www.reddit.com/r/buildapc/comments/5s2wf6/doa_how_often, (2018-11 閲覧)

[6] CPU 単体の DOA 率：

https://www.pugetsystems.com/labs/articles/Most-Reliable-PC-Hardware-of-2016-872, (2018-11 閲覧)

[7] Software testing: What is an acceptable level for software regression rate?:

https://answers.yahoo.com/question/index?qid=20070503111729AAXExT8, (2007)

[8] “Los Altos Workshop on Software Testing”: http://www.developsense.com/blog/category/regression/, (1996)

[9] 篠原昭夫, 泉隆：「オープン・システム障害の分類と対策立案方法の提案」, 第 14 回情報科学技術フォーラム, RO-011 (2015-09)

[10] Robert W. Kembel:“Fibre Channel A Comprehensive Introduction”, Northwest Leading Associates, Inc, chapter10, (2001)

[11] 8B/10B encoding:

http://www.snia-j.org/dictionary/storage_network_keywords/2.html, (2013)

[12] Byte oriented DC balanced (0,4) 8B/10B partitioned block transmission code, http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=4,486,739.PN.&OS=PN/4,486,739&RS=PN/4,486,739, (1982-06)

[13] 篠原昭夫，泉隆：「オープン・システム障害対応の現状分析」，情報処理学会第 76 回全国大会，4ZE-4（2014-03）

第７章　　結論

オープン系システムで発生する未知障害は人手で調査されているが，その手法はいまだ確立されていない。そこでこれを解決するために機能線を用いた障害原因調査手法を提案した。さらに，機能線での調査を効率的に行うために分解レベルを，そして調査対象を削減するために補助機能線を提案した。機能線は人手による障害原因調査に従事する技術者に筆者らが聴き取り調査を行い，障害原因部位特定に至るまでの過程をモデル化したものである。

続けて，機能線上のどのコンポーネントで障害が発生したかを判定する手法として比較法を提案した。これはシステム正常稼動時に事前収集しておいたログ情報と障害発生時に同じ方法で採取したログ情報とを比較し，正常時にはみられないログ情報を障害の痕跡として注目するものである。

機能線はコンポーネントを用いて障害を視覚的に記述し，比較法は，機能線上のコンポーネントで障害が発生したかを判定する。両者を用いることにより，障害原因部位を特定することができる。

次に機能線が障害を視覚的に記述している性質を利用して，障害の分類を行った。分類した各障害型には適切な対策方法を関連付ける。これにより調査対象の障害に最適な対策方法を選択することが可能となる。各々の対策方法では，DOA の検出方法が考慮されている。また，これが発生した場合にも判定方法が付随しているため、二次障害原因調査が容易になる。

オープン系システムを構成する製品が市場へ投入される速度は依然衰える兆候はなく，加えてバージョンアップ，修正プログラムの提供間隔は短くなる傾向がある。これらの要因が未知障害発生件数を高めている。このことは人手による障害調査の必要性につながり，機能線は調査コンポーネントの抽出に有効に機能するものである。

本論文の成果をまとめると次のようになる。

(1)　機能線による障害の記述と原因部位特定

機能線によって障害を記述すること，さらにこれを障害原因部位特定に用いることの利点は以下の通りである。

・視覚的表現により障害全体像の把握が容易

・多様なオープン系システムに適用可能

・コンポーネントを調査単位とするため，ハードウェアとソフトウェアの違いや製品の規模を意識する必要がない

・コンポーネントの分解レベルによって効率的に調査を実行可能

・補助機能線によって調査対象コンポーネントを削減可能

・実障害事例から機能線の適用可能率が高いことを示した

・機能線を用いることにより，調査対象コンポーネントを約２割削減できた

・FTA との比較を行い，違いおよび優位性を明らかにした

(2)　比較法による障害発生有無判定

比較法によりコンポーネントでの障害発生有無判定を行うことの利点は，以下の通りである。

・システム毎の運用特性を反映した障害発生判定が可能

・製品仕様の詳細が非公開でも適用可能

・開発環境よりも調査情報が不足していても適用可能

(3)　障害分類による対策立案選択手法

機能線を用いて障害を分類し，各障害型に対策手法を関連付けることの利点は以下の通りである。

・オープン系システムで発生する障害を分類可能にした

・分類された障害型に一意の対策手法を関連付ける手法を示した

・障害型と対策手法の関連付けにより，二次障害発生を抑止することができる

本論文では，中・小規模のオープン系システムに対して障害原因部位特定をシステマティックに行うことを可能にした。今後はより大規模のシステムに適用可能なよう機能線の改良を行ってゆきたい。また，コンポーネント間の関連情報に時間要素を加えることで，パフォーマンス障害へ適用することも検討したい。

謝辞

本研究を遂行するにあたり，指導教授として長期に渡り懇切なご指導を賜りました日本大学理工学部応用情報工学科 泉隆教授(現在，特任教授)に心から感謝の意を表します。

本論文をまとめるにあたりご指導いただいた日本大学理工学部応用情報工学科 香取照臣教授，細野裕行教授，中村英夫名誉教授に感謝いたします。

機能線と実際の障害原因調査との関連について，意見交換の場を作っていただき，貴重なアドバイスをいただいた辻村尚夫氏に感謝いたします。

機能線の実効性評価を行うにあたり，コールセンターの障害記録を提供して下ったデータライブ株式会社に感謝いたします。

著者発表論文等

〔査読付き論文〕

(1) Akio Shinohara, Hisao Tsujimura and Takashi Izumi, ”A New Problem Analysis Method in Open Systems Using ″Function Chain″”, Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems in conjunction with Intelligent Systems Workshop 2018, pp.1014-1020 (2018-12)

(2) 篠原昭夫, 泉隆：「オープン・システム障害の分類と対策立案方法の提案」, 第14回情報科学技術フォーラム, RO-011, 第４分冊 pp.93-98 (2015-09)

(3) 篠原昭夫, 泉隆：「オープン・システム上での障害発生部位特定方法の提案」, 第13回情報科学技術フォーラム, RO-010, 第４分冊 pp.75-80 (2014-09)

〔学会発表〕

(1) 篠原昭夫, 小寺建輝, 泉隆：「マルウェア亜種の分類手法に関する検討」, 日本知能情報ファジィ学会　第33回ファジィシステムシンポジウム, WE1-4 (2017-09)

(2) 篠原昭夫, 泉隆：「オープン・システムでの修正プログラム適用実態報告」, 電気学会全国大会, 3-063 (2015-03)

(3) 篠原昭夫, 泉隆：「オープン・システム障害対応の現状分析」, 情報処理学会第77回全国大会, 3ZE-05 (2015-03)

(4) 篠原昭夫, 泉隆：「オープン系システム保守の現状報告」, 情報処理学会　第76回全国大会, 4ZE-4 (2014-03)

(5) 篠原昭夫, 泉隆：「システム障害発生部位判定方法と障害の分類」, 第12回情報科学技術フォーラム, B007 (2013-09)

(6) 篠原昭夫，泉隆：「オープン・システム上での障害発生部位特定方法の提案」，情報処理学会　第 75 回全国大会，5A-2（2013-03）

(7) 篠原昭夫，泉隆：「システム障害原因究明方法の考案」，日本知能情報ファジィ学会　第 35 回ファジィ・ワークショップ，pp. 83-84（2010-03）

〔表彰〕

(1) 情報処理学会　第 77 回全国大会学生奨励賞，(2015-03)（篠原昭夫，泉隆，オープン・システム障害対応の現状分析）

(2) 情報処理学会　第 76 回全国大会学生奨励賞，(2014-03)（篠原昭夫，泉隆，オープン系システム保守の現状報告）