HITACHI
Inspire the Next

ソフトウェア・メインテナンス研究会

情報システムの障害に対する準備と対処方法

2016/5/19

株式会社 日立ソリューションズ

鈴木　勝彦

HITACHI
Inspire the Next

情報システムの障害に対する準備と対処方法

1. 障害にも準備が必要
2. 環境の変化があったか
3. 障害に対しての体制を事前に決めておく
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

1. 障害にも準備が必要

HITACHI
Inspire the Next

障害が発生した時に素早く対応するには、事前の準備しておくことが重要です。
準備の状況によって、初動が大きく変わります。

(1)障害対応マニュアルを作成しておく。
- (a) 障害の重要度の判定基準
  重要度によって体制を変える
- (b) 担当者の連絡先
- (c) コールセンタの連絡先とOS/ミドルなどの問合せ時の契約番号
- (d) 関連する部署の連絡先
- (e) 社内のエスカレーションリスト
- (f) ハードウェアの一覧
- (g) ハードウェア構成図
- (h) OS/ミドルなどのソフトウェア一覧
- (i) ソフトウェア構成図
- (j) ソフトウェアの設定パラメター覧
- (k) ログ採取ツール(OS用、ミドルソフト用、UP用)

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

2. 障害に対しての体制を事前に決めておく

HITACHI
Inspire the Next

致命的でかつ緊急度の高い障害が発生した時には、事前に体制を決めておかないと混乱状態となり調査がうまくいかないことがある。緊急時には、2時間間隔ぐらいで障害対策会議を開催する。

■障害時の体制

- ・指揮者(正/副) :以下の担当のアサインと障害対策会議の運営を推進する。
  幹部などから担当に直接指示が出ないようにコントロールする。
- ・記録者(正/副) :会議中は、ホワイトボードなどに記載し全員に周知できるようにする。
  ホワイトボード以外の担当からの調査結果メモなども管理する。
  採取したログなども管理する。
- ・障害回避責任者 :回避方法があるかを検討する。
- ・障害回復責任者 :回復作業が必要な障害は、回復方法があるかを検討する。
- ・原因調査責任者 :複数の製品が関係する場合には、製品毎に調査責任者を設置する。
  全体の調査責任者は、製品毎の調査結果の共有を図る。
- ・再現テスト対応者:現地と同様/類似の環境を作成し、再現テストを試みる。
- ・同件調査責任者 :既知のOS/ミドルソフトなどの不良の調査を実施する。
- ・報告書作成者 :原因が判明しない状況でも定期的に中間報告書を作成する。
- ・現地連絡窓口 :現地との情報連絡係りを一本化し、現地作業の優先度も管理する。
- ・現地対応者 :マシンルームでは携帯電話などが使えないことがあるので、
  必ず2名以上にする。進捗が無くても定期的に連絡する。

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

3. 環境の変化があったか

HITACHI
Inspire the Next

障害の原因には、いろいろな要因があります。
ソフトウェアの場合には、ハードウェアのような経年変化に起因することはほとんどありません。ソフトウェアは、分岐の塊のようなものなので、条件によって動作が変わります。
今まで動作していたシステムであれば、障害が発生したということは、条件が変わったことが起因している可能性が高いと考えるのが一般的です。ただし、時々ではあるが、タイミングで発生することもあります。
つまり、いつもと違う条件(=環境)の変化があったかを並行して調査することが大切です。

(1)特定の時刻・時間(time)
- ・時刻(time points):うるう日
- ・時間(time intervals):長時間運転

(2)規模が大きくなった時(scale out)
- ・マシンの増設
- ・支店の増加

(3)データ量の増加(scale up)
- ・処理件数の増加
- ・処理データのサイズの肥大化

(4)データの変化(data)
- ・処理するデータの内容の変化
- ・ウイルスパターンファイルの変化

(5)パラメタの変更(parameters)
- ・OSのパラメタを変更
- ・ミドルソフトの環境設定の変更
- ・UPの環境設定の変更

(6)システム構成の変更(configuration)
- ・周辺装置の変更
- ・通信経路の変更

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

4. 事実を正しく捉える

HITACHI
Inspire the Next

クリティカルな障害が発生すると正しい情報だけでなく、推測の情報も混ざり、混乱状態になることがある。

【事実を正しく捉える-その1】
・現地での情報を入手する体制を確保する
日本では、エンドユーザーがプログラム開発して、運用しているケースは少ないので、開発者が現地にいることはあまりない。クリティカルな障害が発生している時ほど、不思議と情報が入ってこないことがある。2章で説明したような緊急時の連絡窓口が確立されていないと、情報が発信されない事態に陥ってしまう。
このため、クリティカルな業務を運用している顧客で重大障害との一報を受けた時点で現地に人を向かわせることが大切である。

【事実を正しく捉える-その2】
・採取する資料は事前に決めて、採取するためのツールを作成しておく
プログラムのログなどは、無限に残すことができないので、いつかはラップしてしまう。このため、障害発生時には、できるだけ速やかにログの採取が必要である。調査を開始すると次々に必要となる資料が判明して、追加でログを採取することがあるが、想定されるログは、事前に準備して採取する。発生直後に同時にログを採取することで、同じ時間帯のログが残り、システム全体の動作も把握できる。

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

5. 事例:性能トラブル時の調査観点

HITACHI
Inspire the Next

障害調査は、現象とログを基に実施することになる。経験値の高いベテランであれば、勘で原因が究明できることもあるが、長期化することもある。

今回は、性能トラブル時の調査観点についての事例を説明する。

性能トラブルは、長期化することが多い。 性能トラブルに対して、経験豊富な特別な技術者でなく、普通の技術者でも4W1Hによる体系的な観点に基づいて調査すれば、試行錯誤することなく、解決までの時間短縮に有効である。

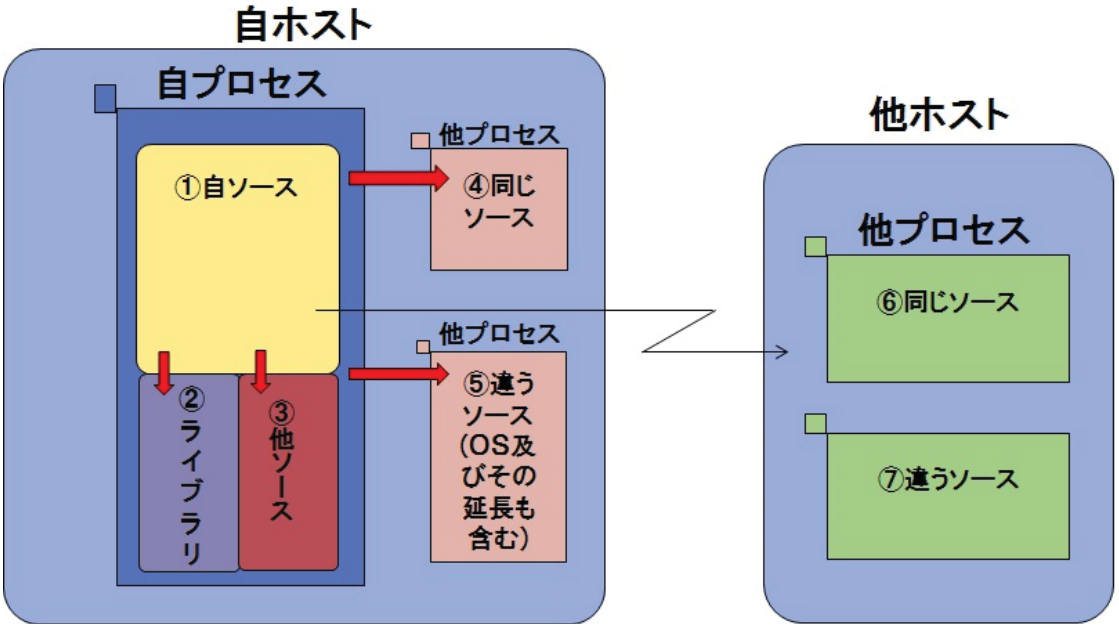
一般的に「5W1H」は、When(いつ)、Where(どこで)、Who(誰が)、What(何を)、Why(なぜ)とHow(どのように)であるが、今回は以下のように「Where」と「Why」を除き、Whomを追加して「4W1H」とした。

- Who(誰が)は、原因となる犯人(プロセス)を究明する。
- How(どのように)は、原因となる動作を究明する。
- What(何を)は、原因となる資源を究明する。
- When(いつ)は、原因となるきっかけを究明する。
- Whom(誰によって)は、原因を誘発したものを究明する。

5.1 Who（誰が）は、原因となるプロセスを究明（1）

HITACHI Inspire the Next

・性能トラブルがどのプロセスでどの場所に起因して発生しているかを特定する。



5.1 Who（誰が）は、原因となるプロセスを究明（2）

HITACHI Inspire the Next

・性能トラブルがどのプロセスに起因して発生しているかを特定する。

1. プロセスを特定するには、システム全体及びプロセス単位でのCPU使用率、DISK入出力回数とバイト数、データ通信量を確認して絞り込む。

1.1 自プロセスのCPU使用率が高い、DISK入出力回数が多い、または転送バイト数が多い、データ通信量が多い場合は、自プロセスの可能性が高い時は次の3つが考えられる。
- ・自プロセスの自分で作成したソースで発生。
- ・自プロセスだが、取り込んだライブラリで発生。
- ・自プロセスたが、callしているAPIの延長で発生。（システム関数の場合も）

1.2 他プロセス（自分で作成したソース）のCPU使用率が高い、DISK入出力回数が多い、または転送バイト数が多い、データ通信量が多い場合は、他プロセスの可能性が高く、次の2つが考えられる。
- ・自プロセスの延長で他プロセスが実行されるが、他プロセスからの戻りが遅いことで自プロセスの性能がでないことがある。この場合は、他プロセスに関して調査する。
- ・自プロセスは、他プロセスと通信しながら処理しているが、他プロセスからの戻りが遅いために性能がでないことがある。この場合は、他プロセスと通信の状態に関して調査する。

1.3 他プロセス（他人が作成したソース）のCPU使用率が高い、DISK入出力回数が多い、または転送バイト数が多い、データ通信量が多い場合は、他プロセスまたは、システム全体の可能性が高く次の2つが考えられる。
- ・特定の他プロセスだけ（システムプロセスも含む）がリソースをたくさん使用している場合には、自プロセスからcallしている延長で発生しているかを確認する。
- ・特定の他プロセスだけ（システムプロセスも含む）がリソースをたくさん使用している場合には、自プロセスが確保するリソースと競合が発生していないかを確認する。

1.4 システム全体のCPU使用率が高い、DISK入出力回数が多い、または転送バイト数が多い、データ通信量が多い場合は、他プロセスまたは、システム全体の可能性が考えられる。
- ・システム全体がリソースをたくさん使用している場合には、自プロセスが確保するリソースと競合が発生していないかを確認する。

5.1 Who（誰が）は、原因となるプロセスを究明（3）

HITACHI
Inspire the Next

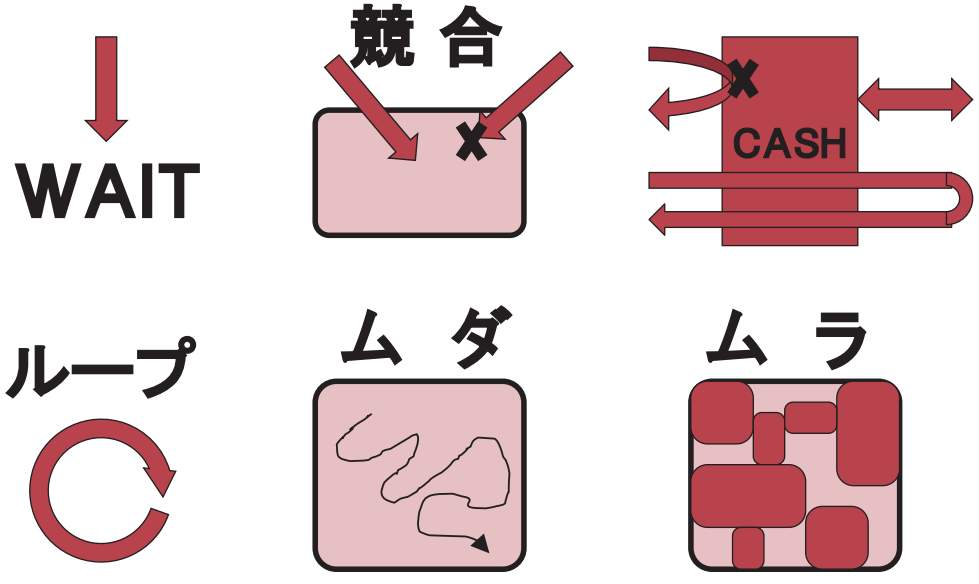
2. プロセスを特定した後、さらにどのソースコードで発生しているかを追究する。

2.1 自プロセスでかつ自分のソースで発生
　自分のソースなので、トレースログをあれば、場所の特定も容易。
2.2 自プロセスであるが、取り込んでいるライブラリ部分で発生
　ライブラリの前後でトレースログを出力すれば、絞込み可能。
2.3 自プロセスであるが、APIの延長(他人のソース)で発生
　APIの前後でトレースログを出力すれば、絞込み可能。
2.4 他プロセスであるが、自分と同じソースで発生
　自分のソースなので、トレースログを強化すれば、場所の特定も容易。
2.5 他プロセスでかつ、他人のソースの部分で発生
　APIの前後でトレースログを出力し、該当する他プロセスが自分の発行しているAPIの延長(OSなども含む)であるかを確認する。
　他プロセスの場合は、他プロセスが特定のリソースを大量に消費していることで影響を受ける場合もある。
2.6 他ホストであるが、自分と同じソースで発生。
　通信処理の前後でトレースログを出力すれば、絞込み可能。
　他ホストとの通信がある場合には、他ホストからのレスポンス待ちで自ホストの自プロセスが遅く見える場合があるが、原因の特定は他ホストがキーになる。
2.7 他ホストでかつ、他人のソース
　通信処理の前後でトレースログを出力すれば、絞込み可能。
　しかし、他ホストの他人のソースが起因する場合には、原因の究明は難しい。

5.2 How（どうした）は、原因となる動作を究明（1）

HITACHI
Inspire the Next

・性能トラブルがどうやって発生しているかを特定する。



5.2 How（どうした）は、原因となる動作を究明（2）

HITACHI
Inspire the Next

・性能トラブルがどうやって発生しているかを特定する。
性能トラブルの原因のメカニズムに関しては、大きく「ループ」、「WAIT」、「キャッシュ」、「ムダ」、「ムラ」、「競合」に分類できる。性能トラブル解析時には、どのように動いているかを特定する必要がある。

WAIT
- デッドロック
- タイマー値大
- タイマー値大(リトライ)
- タイマー設定でのリトライ多発
- 相手の処理待ち
- 無限待ち

競合
- 排他資源
- CPU
- メモリ
- DISK
- 通信

キャッシュ
- ヒット率低下
- キャッシュ不足
- 遅いキャッシュ
- キャッシュが無効

ムラな処理
- プライオリティ
- 断片化
- バッファサイズの不適切

ループ
- 無限ループ
- リトライ

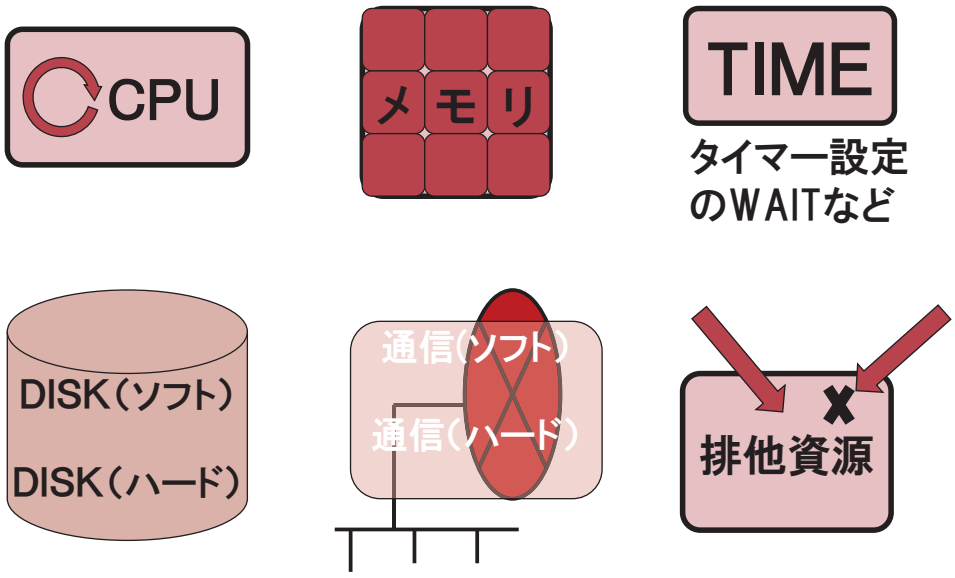
ムダな処理
- ループ中のループ
- 輻輳
- メモリ確保関数を頻発
- I/O処理を頻発
- 同期処理
- コピーの実装が悪い
- ソート、コンペアの実装が悪い
- サーチ、インデックスの実装が悪い
- キューなどの実装が悪い

その他
- 同居製品固有の実装ロジック
- OS固有の実装ロジック
- ライブラリ固有の実装ロジック
- 自プロセスの延長の固有の実装ロジック
- ハードウェア特性

5.3 What（何を）は、原因となる資源を究明

HITACHI
Inspire the Next

・性能トラブルの原因となる資源を特定する。



5.4 When（いつ）は、原因となるきっかけを究明

HITACHI
Inspire the Next

・性能トラブルが発生するきっかけ(いつから)を特定する。

エラー

・システムコールエラー
・I／Oエラー
・通信エラー
・排他エラー

経年変化

・大規模になった時
・長時間の経過後
・データ量が多くなった時
・機器構成を変更
・OSのパラメタを変更
・ミドルソフトの環境設定を変更
・UPの環境設定を変更
・通信経路を変更

通信相手が起因

・相手がビジーによるリトライ
・相手がビジーによる処理待ち
・相手がnot ready時のリトライ

DISK

・DISKビジーによる処理待ち
・DISKキャッシュの電池切れ

不良

・他の業務UPの不良
・OSの不良
・VMウェアの不良
・ファイル管理ソフトの不良
・ハードウェアの不良

5.5 Whom（誰によって）は、原因を誘発したものを究明

HITACHI
Inspire the Next

・性能トラブルを誰によって誘発されたかを特定する。

ミドルウェア

・ウイルス監視製品
・暗号化製品
・バックアップ、ディザスタ製品
・ファイル転送製品
・DB製品

業務UP

・同様の業務UP
・他の業務UP

その他

・他ホスト
・その他

5．6 実践で使えるために現象から絞り込む方法

HITACHI
Inspire the Next

性能トラブル発生時は、4W1Hの観点で調査していくことで絞り込みが可能である。
また、現象から具体的に絞り込む方法については、以下のようなExcel形式でまとめた資料があるので活用していただきたい。

「性能トラブル解決の手引き-事例編_」の全文は、sercのHP に掲載してある。　http://www.serc-j.jp/

HITACHI
Inspire the Next

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

6. 原因究明より復旧を優先する

HITACHI
Inspire the Next

障害が発生すると原因究明の調査を開始するが、同時に復旧も考える。

障害のレベルによって対応の方法も変える必要がある。マシンの再起動を実施するとしばらく稼働が止まるので躊躇するが、原因究明が長期化した場合のために、事前に再起動するタイミングを決めておくとよい。

1. システム全体がダウンしている状態
 (1)プロセスの状態などを確認して、資料採取してマシンのリブートをする。
  OSのコマンドなども実行されない場合には、システムダンプも採取する。

2. ミドルソフトが動かない状態
 (1)該当するミドルソフトに関する資料採取してミドルソフトの再起動をする。
 (2)ミドルソフトの再起動でも回復しない場合には、マシンのリブートをする。

3. 部分的にエラーが発生する状態
 (1)エラーの起因元を追究して対処する。
 (2)ミドルソフトの再起動でも回復しない場合には、マシンのリブートをする。

4. マシンの再起動で復旧しない場合
 (1)周辺機器のディスクや通信装置の再起動する

HITACHI
Inspire the Next

1. 障害にも準備が必要
2. 障害に対しての体制を事前に決めておく
3. 環境の変化があったか
4. 事実を正しく捉える
5. 事例:性能トラブル時の調査観点
6. 原因究明より復旧を優先する
7. 長期化したらKT法を使え

7. 長期化したらKT法を使え

HITACHI
Inspire the Next

社会心理学者のチャールズ・ケプナー(Dr. Charles Kepner：1922-)と
社会学者のベンジャミン・トリゴー(Dr. Benjamin Tregoe：1927-2005)の
名前に由来する。

KT法には、4つの手法があるが、障害調査は、
「問題の明確化と原因究明」
になるので「問題分析(PA)」を適用するのがよい。

http://jp.kepner-tregoe.com/
http://www.monodukuri.com/gihou/article/390

HITACHI
Inspire the Next

ソフトウェア・メインテナンス研究会

END

情報システムの障害に対する準備と対処方法

2016/5/19

株式会社 日立ソリューションズ

鈴木　勝彦

HITACHI

Inspire the Next