障害予測における最適な障害回避手段の提示法

加藤 裕[1,a)]　敷田 幹文[2]

概要：本稿は，情報システムの可用性を維持・向上するためのアプローチとして障害予測技術に着目し，予測された障害からそれを回避する最適手段の自動的な生成・提示を行う仕組みを提案するものである．情報システムの大規模化や複雑化が進むなか，障害発生後の早期復旧だけでなく，障害の兆候を事前に検出し回避する運用管理手法への期待が増大すると考えられるが，障害の回避には高度な運用管理のスキルや経験が必要である事が多い．そこで，システムの大規模化や複雑化に対応しうる自動的な障害回避策の提示法を提案し，障害回避の機会を増やして障害そのものを減らす事で，情報システムの可用性の向上に繋げるのが本研究の狙いである．

Presentation method of optimal avoiding way of failure prediction

Abstract: In this paper, we propose a method for automatic generation and presentation of the optimal way to avoid from failure prediction. We have focused on the failure prediction technique is one of the approaches to maintain and improve the availability of information systems. In a large-scale and complexity of information systems has progressed, the expectation of failure prediction method has increased. However, to avoid failure from failure prediction, it is necessary to operators with advanced management skills and experience. Therefore, by increasing the opportunity to avoid the failure by the proposed method to reduce the failures, we aim to improve the availability of information systems.

1. はじめに

情報化社会が発展するにつれ，情報システムに要求される役割は日々増大している．今日の情報システムは，利用者数の増加，扱うデータの大容量化，サービスの 24 時間 365 日提供など様々な要求に答えながら，サービスレベルの維持・向上を目指して運用を行っている．一方で，情報システムが及ぼす障害の影響もこれまで以上に大きくなっており，銀行，証券取引所，交通機関，情報通信網などにおける情報システムの障害は我々の社会生活の妨げとなっている．加えて，クラウドコンピューティングの普及に伴い，1 つの障害が多数のサービスへ同時に影響を及ぼす事態も発生している．よって，情報システムの可用性向上が不可欠であるといえる．

[1] 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292, Japan
[2] 北陸先端科学技術大学院大学 情報社会基盤研究センター
Research Center for Advanced Computing Infrastructure, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292, Japan
[a)] y-kato@jaist.ac.jp

情報システムの可用性 (Availability) は，MTBF (平均障害間隔) と MTTR (平均修復時間) の式によって表される．

$$Availability = \frac{MTBF}{MTBF + MTTR}$$

このように，可用性を向上するには，MTBF を向上する施策と，MTTR を削減する施策の 2 通りがあることがわかる．運用管理においては，サービスの機能的・非機能的要件を満足できなくなることが障害であり，障害を未然に防ぐことが MTBF 向上に，障害から早期に復旧することが MTTR 削減に繋がる．特に MTTR 削減は，障害検出技術，根本原因解析技術，そして障害復旧手順の最適化など，多くの施策が研究され，また製品化されてきた．

一方で，情報システムの障害が引き起こす社会的影響が増大している今，これまでの高信頼なシステム設計や実装といった大規模障害を防ぐための様々な施策に加え，運用管理による MTBF 支援も求められるようになると考えられる．障害の予兆を事前に発見する障害予測技術を用いると，サービスダウンに繋がる障害をいち早く見つけ，回避することが可能になる．ただし，運用管理者が実際に障害を回避するには，システム構成や現在状態などを把握する能力など，高度なスキルや経験が求められる．

そこで本研究では，大規模化・複雑化する情報システムにも適用可能で，かつ高いスキルや経験を持たない運用管理者でも可用性の向上に貢献できる，障害予測における最適な障害回避策の生成・提示法を提案する．

以下，2 章で運用管理に関する製品や研究について述べ，3 章で障害の最適回避策の提示法の提案を行う．4 章でその考察を述べ，最後に 5 章でまとめと今後の課題を述べる．

2. 関連製品・研究

既存の運用管理製品や，障害予測における研究について述べる．

2.1 既存の運用管理製品

運用管理を支援するソフトウェア等は数多く存在するが，特に有名な製品として HP OpenView や IBM Tivoli[1] が挙げられる．これらは以下のような機能を有している．

- サーバ管理機能
- ネットワーク管理機能
- 状態監視
- 障害分析
- ジョブ管理
- ワークフロー管理

これらの機能を有した製品は統合運用管理とも呼ばれ，日本国内の複数のベンダーも開発している．ある程度の規模の情報システムであれば，運用において必要不可欠なツールであり，幅広く利用されている．

しかし，今日の情報システムにおける非機能要件の複雑化，運用ポリシーの複雑化などに柔軟に対応しきれているとは言えず，運用管理者のスキル・経験で補われている側面が否定できない．また，可用性向上に注目すると，ダウンタイム削減 (MTTR 削減) に重点が置かれており，運用における MTBF 向上には乏しい．

2.2 障害予測技術

運用における MTBF 向上の施策として，障害を予測し回避する技術が挙げられる．今日までの障害予測技術は発展途上であるが，様々なアプローチで研究がなされている．

Sahoo ら [2] は，クラスタリングシステムに対し，複数の確率モデル的手法を用いてイベントを予測する研究を行った．また 2005 年には Bodik ら [3] が統計的手法を用い，予測に繋がる障害の分析手法を提案した．後者は実際の Web アプリケーションサーバのログに対して実施されており，より実用的な評価がなされている．予測可能な障害の種類としては，サーバダウンに繋がるリソースの問題やパフォーマンスの問題などがあり，事前回避可能であれば可用性向上に寄与すると考えられる．

2.3 運用管理製品の最新動向

2012 年 3 月に，HP は障害予測技術を統合した新しい運用管理製品 HP Service Intelligence を発表した [4]．ハードウェア・ソフトウェアにおける様々な障害を過去の障害発生パターンと照合する等の複数の独自手法により予測し，運用管理者へ通知することができる．ただし MTBF 向上という観点からは，予測された障害の回避判断・措置は引き続き運用管理者に任されており，高いスキル・経験を持つ運用管理者が必要である．

2012 年 4 月には，IBM がエキスパート (専門家) の知見をパターン化し，顧客に合わせてそれらのパターンを組み合わせて最適な運用管理を提供する PureSystems を発表した [5]．運用管理者の負担削減という意味で価値があるが，専門家の知見を顧客のニーズに摺り合わせるのは IBM のエンジニアによる人手であり，熟練技術者からの真の依存脱却という課題が残っていると言える．

3. 最適回避策の提示法

既存の運用管理製品における課題として，可用性を維持・向上するための障害回避措置は高度な運用管理者に頼っている事を 2 章により明らかにした．そこで本研究では，高いスキルや経験を持たない運用管理者でも扱え，大規模化する情報システムや複雑化する運用ポリシーにも適用しうる自動的な障害回避策の提示法を提案する．

初めに，障害回避策生成・提示の自動化が可能である事を示すために既存の手法を本分野に適用する予備実験を行い，問題点を議論・整理する．次に，本研究の目的を達成するための提案方式として，以下の 2 点を説明する．

- ルール生成自動化
- 最適回避策提示法

3.1 エキスパートシステムを用いた予備実験

障害を回避するには，対象のシステムに関する情報や障害予測情報，過去の事例などを総合的に判断した上で適切な障害回避策を下す必要がある．しかし，熟練の運用管理者に依存せずにこれを可能にするにはこれら高度な判断を自動化する必要がある．そこで，熟練者の知識・判断能力を再現するために，知識情報処理における既存の手法の 1 つであるエキスパートシステム [6] を用いてこの問題を分析する．

3.1.1 回避策推論システムの実装

エキスパートシステムを用いた回避策推論の有用性を判断するために，図 1 に示すシステムを実装した．

発生した障害予測データと，そのシステムの構成データ等を既知事実群として登録し，それに回避パターンをまとめた知識ベースを前向き推論によって連鎖的に適用させていく事で，現状に即した回避策を得る仕組みである．

最終的な実装は Java SE 7[7] を用いて行った．

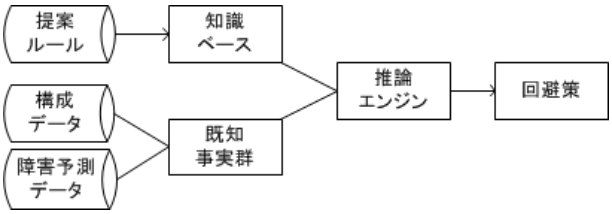

図 1 回避策推論システムの構成

3.1.2 既知事実群の実装

既知事実群は事実の集合であり，対象システムの次の情報が含まれる．

- システムの静的な情報
  - 各構成要素の詳細情報 (性能など)
  - 構成要素間のネットワークに関する情報
  - 各構成要素が提供するサービスの情報
- システムの動的な情報
  - 構成要素間の依存関係
  - 各構成要素の動作状態
  - リソース状態
- 障害予測情報

今回実装する上でこれらの情報の一部を表現した例を図 2 に示す．

```
<?xml version="1.0"encoding="UTF-8"?>
<facts>
  <fact>HOST-FS1 は稼働中</fact>
  <fact>HOST-FS2 は休止中</fact>
  <fact>HOST-Web1 は稼働中</fact>
  <fact>HOST-Mail1 は稼働中</fact>
  <fact>HOST-FS1 は SERVICE-FS を提供中</fact>
  <fact>HOST-Web1 は SERVICE-Web を提供中</fact>
  <fact>HOST-Web1 は SERVICE-FS を利用中</fact>
  ...
</facts>
```

図 2 既知事実群 (システム構成・状態データ) の記述例

各事実は自然言語で記述し，これらが後述する提案ルールを適用するための条件になる．

障害予測情報に関しては，用いる障害予測技術によって大きく内容が変わり，提案ルールの策定・適用にも大きな影響を与える事が考えられる．既存の障害予測技術である Web アプリケーションにおけるアクセス時間解析方式による障害予測技術 [8] の研究では，障害予測技術の各障害予測の出力として次の情報が得られる事が読み取れる．

- 異常度 (障害予測の疑わしさ)
- 障害予測の発生原因場所

また，複数の障害予測の結果を分析する事により，次の情報も得られる．

- 障害予測の精度
- 障害予測の発生原因場所特定の精度
- 障害発生の大まかな予想時刻

本システムでは，これらを障害予測発生時に収集・既知事実群へ追加する事を想定している．

3.1.3 知識ベースの実装

知識ベースは IF-THEN 形式のルールの集合であり，今回実装する上で提案ルールを表現した例を図 3 に示す．

```
<?xml version="1.0"encoding="UTF-8"?>
<rules>
  <rule name="FS切り換えルール1">
    <if>HOST-FS1に障害予測が発生</if>
    <if>HOST-FS1はSERVICE-FSを提供中</if>
    <if>HOST-FS2はSERVICE-FSを提供可能</if>
    <if>HOST-FS2は停止中</if>
    <then>HOST-FS2を起動する</then>
    <then>HOST-FS2へ切り替える</then>
  </rule>
  <rule name="FS切り換えルール2">
    ...
  </rule>
  ...
</rules>
```

図 3 知識ベース (提案ルール) の記述例

内容である提案ルールは，既知事実群である対象システムの構成・状態及び障害予測情報に基づいたものである．各ルールの if 要素が推論エンジンによって既知事実群と照合され，条件に合う then 要素が新たな既知事実群に追加される．推論エンジンは新たな既知事実が追加されなくなるまでこれを繰り返す事で，最終的な推論を導く事ができる．

知識ベースは自由にルールを増減させる事ができる拡張性，自然言語が利用できる事による可読性に加え，複数のルールや条件を組み合わせた複雑な運用ポリシーが容易に記述可能である．記述例では XML をベースとした記法を採用しているが，この場合 1 つの rule 要素中に複数ある if 要素や then 要素は論理積に相当し，複数の rule 要素に同一の if 要素がある場合は論理和に相当する．これによって，複雑な条件式でもルールを単位とした柔軟な制御が可能である．

3.1.4 動作結果と諸問題

20 件の既知事実群と提案ルールを手動で作成し，今回実装した回避策推論システムに入力したところ，事実に適合する 3 件の障害回避策が特に問題なく出力され，本予備実験によって自動化できる事を確認した．そこで，この回避策推論システムが研究目的を達成するために充分であるかについて議論を行った．

議論の結果，懸念された問題点は以下の 3 点である．

問題 1 システムが大規模化した場合に，提案ルールの管理が追いつかなくなる可能性

問題 2 提示する回避策は様々な判断指標を考慮しなけれ

ばならない

問題 3 誤ったルールや他のシステムのルールが混ざった場合，正しい回避策を提示できない可能性

問題 1 については，システムの規模によっては提案ルールの数が膨大になり運用管理者の手に負えなくなる問題や，構成変化に応じてルールの策定し直しが必要になるといった問題を指す．熟練でない運用管理者が扱えるようにする為にも，こうした問題は解決しなければならない．

問題 2 については，熟練の運用管理者が実施する回避策の判断を再現するために，システムが最適な回避策を提示する必要があるという事である．回避策推論システムのみでは実施可能な回避策が複数提示される可能性があり，その場合それらの優劣は運用管理者が評価し最終的に 1 つの回避策を決定する必要がある．判断指標とは，時間的・金銭的コストやサービスレベルといった指標で，最善の回避策を選択する上で必要になる．また，障害回避においては対象となる障害はその時点では発生していないため，障害発生予想時間や予測精度も考慮した上で回避策を選択すべきである．

問題 3 については， 熟練でない運用管理者が扱うシステムを想定している以上，ルール記述に誤りがあった場合でもシステムを誤った回避策の実行から守る必要がある事と，ルールをテンプレート化して共有することで，他のシステム向けのルールに従った回避策が提示される可能性があるという事である．なお，ルール記述法の誤りは文書型定義[*1] により事前に確認されるほか，論理的に矛盾するルールはエキスパートシステムで採用されずに回避できる．しかし，システムに適合しないルールがエキスパートシステムにより採用され提示されてしまう可能性は残るため，これを排除しなければならない．

3.2 ルール生成自動化

3.2.1 概要

前述の回避策推論の問題 1 を解決するため，提案ルールにおいて実効的なルールを最初から全て用意するのではなく，必要最低限のルールから不足している情報やルールを自動的に展開し統合する仕組みを提案する．概要を図 4 に示す．

3.2.2 基本ルール・詳細ルール

基本ルールは，運用管理者が対象システムの運用方針に基づき必要最小限の情報を入力する．どのホストがどのサービスを提供しているかといったシステムの構成に関する部分は自動的に詳細化するため，基本ルールでは回避策の大まかな記述で済む．更に，あらゆるシステムに適用可能な様々な運用措置をテンプレートとしてまとめることで，システム間でそれらを流用することも可能である．

[*1] XML DTD (Document Type Definision) や XML Schema によって実行する．

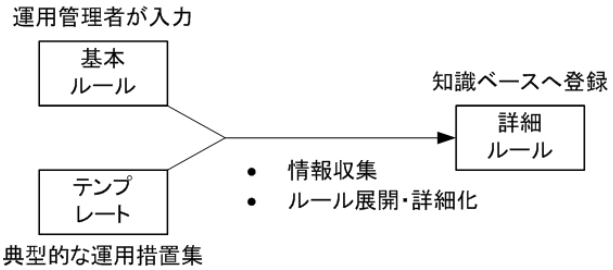

図 4 基本ルールから詳細ルールの生成

具体的なルール詳細化の例を図 5 に示す．

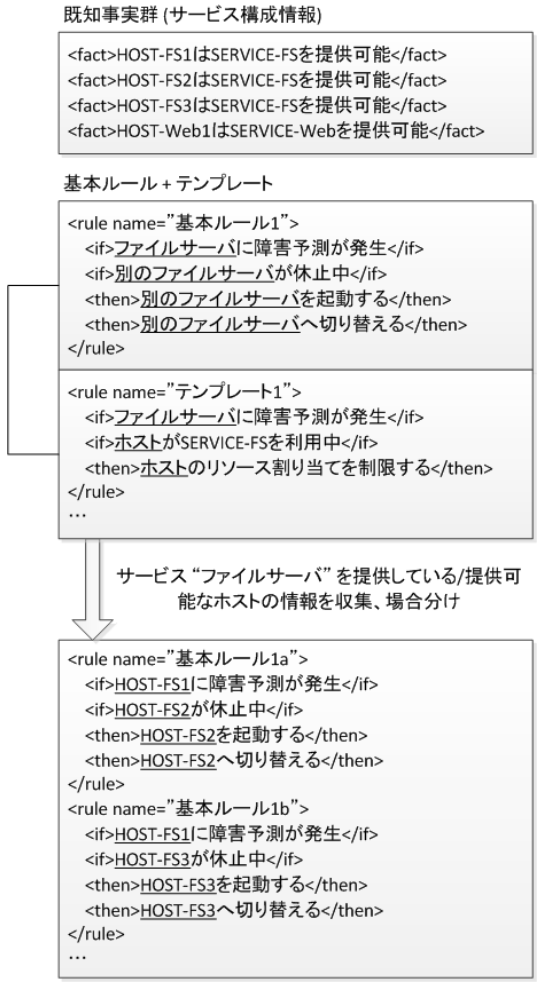

図 5 ルール詳細化の動作例

基本ルールやテンプレートでは適用対象は決定されていない．このようにする事で，運用管理者はシステムの詳細な構成を把握する必要がなく，また変更点がある場合にメンテナンスが困難になる可能性を排除する．詳細化処理では，ルール中の曖昧性を自動的に収集した構成情報を基に照合し，必要であればルールを場合分けする．これにより，最終的に既知事実群に適用可能な実効的な詳細ルールが生成される．なお，このような詳細化を行うために予め初期設定で「ファイルサーバ」が「SERVICE-FS を提供するホストである」といった意味的な定義を与えているが，図中

では省略している．

3.2.3 詳細ルールによる回避策推論

このように生成した詳細ルールによって，実際に障害予測が発生したときに最終的にどのような出力になるかを示したものが図 6 である．なお，ここまでは結果に影響を与える部分のみ図示しているが，実際にはシステムの規模に応じた多数の既知事実群の中から状況に適した回避策を選択・提示している事に注意されたい．

既知事実群 (障害予測情報)

```
<fact>HOST-FS1に障害予測発生</fact>
```

既知事実群 (システム状態情報)

```
<fact>HOST-FS1は稼働中</fact>
<fact>HOST-FS2は休止中</fact>
<fact>HOST-FS3は稼働中</fact>
<fact>HOST-Web1がSERVICE-FSを利用中</fact>
```

出力結果 (回避策)

```
1. 基本ルール1a
   1-1. HOST-FS2を起動する
   1-2. HOST-FS2へ切り替える
2. テンプレート1d
   2-1. HOST-Web1のリソース割り当てを制限する
```

図 6 詳細化ルールに基づく回避策推論結果

障害予測が発生した時点で，既知事実群に障害予測に関する情報，及び発生時点でのシステムの動的な情報が追加される．全ての既知事実群がそろったところで回避策推論システムにそれらが入力され，詳細ルールの適用を行う．回避策推論システムの出力は，推論エンジンによって新たに追加された既知事実の集合であり，これが現状に即した有効な回避策の候補という事になる．

3.2.4 自動化の効果

詳細ルール生成を自動化する事により，少ない基本ルール・テンプレート記述で，多くの構成要素からなるシステムに適用可能になる．実際にどれほどの記述量の削減が見込めるかは，基本ルール記述の抽象度と対象システムの規模に応じて変化するが，実際のシステムの詳細な構成や，台数などに依存しないルール記述が可能になるため，問題 1 として挙げた提案ルールの管理が追いつかなくなる可能性を排除できると考えられる．

3.3 最適回避策提示法

3.3.1 概要

前述の問題 2 は，提示する回避策は様々な判断指標を考慮しなければならないというものであった．回避策推論システムが出力する結果は、どれも障害回避に繋がると推論された有効なものだが，この中には SLA[*2] に不適合な回避策や，障害予測における回避可能時間内に達成できないと見込まれる回避策などが含まれる可能性がある．さらに，熟練でない運用管理者でも扱うことを可能とするには，有効な回避策の候補から実際にどの回避策を実行に移すのが適切かを判断するための支援も必要である．

同問題 3 は，誤ったルールや他のシステムのルールが混ざった場合を考慮しなければならないというものであった．誤ったルールによってシステムに効果のない，あるいは悪影響を与える回避策を提示してしまう可能性を排除するためにも，同様に回避策候補の適・不適を判断する必要がある．

そこで，最適な回避策を提示するための仕組みを提案する．図 7 に回避策推論システムの出力から最適回避策を選出する過程を示す．

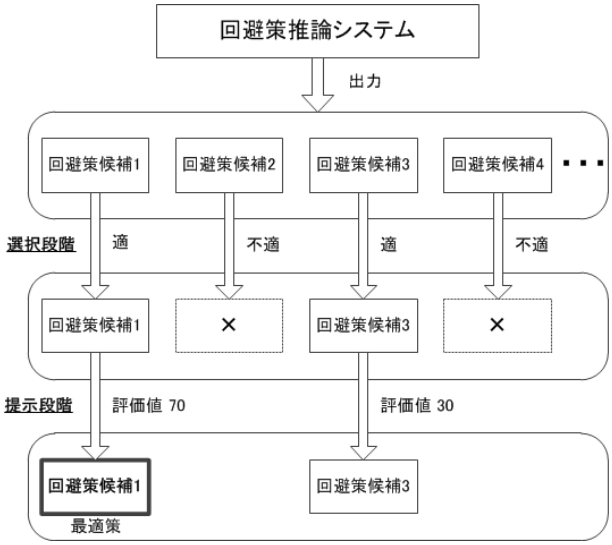

図 7 最適回避策提示法

最適回避策を決定する処理は，大きく分けて選択段階と提示段階に分かれる．以後それぞれについて詳しく説明する．

3.3.2 選択段階

まず，各回避策の評価項目の情報収集を行い結果を照合する．前小節で示した出力例を用い，最適策を判断するための必要な情報の収集と照合の様子を示したのが図 8 である．

各構成要素の時間的コスト，金銭的コスト，サービスレベルは予め対象システムを設計した者が与える．また，障害予測情報は発生時点で得られたものである．

次に，これらを表にまとめ，基準に基づいて適・不適を判断する．各回避策候補と収集した判断指標から最適回避策を選出する様子の例を表 1 に示す．

ここで言う基準とは，各評価項目における次のようなものを指す．

- 予想される最短猶予時間内に回避策を実行できるか

[*2] Service Level Agreement

出力結果 (回避策)

```
1. 基本ルール1a
   1-1. HOST-FS2を起動する (項目A)
   1-2. HOST-FS2へ切り替える (項目B)
2. テンプレート1d
   2-1. HOST-Web1のリソース割り当てを制限する (項目C)
```

収集した判断指標の情報

```
障害予測情報
  予想される最短猶予時間: 45分 ← (項目A, Bで使用)
  信頼度: 6 (0-10)
HOST-FS2の時間的コスト
  起動に必要な時間: 150秒 ←(項目Aで使用)
  停止に必要な時間: 40秒
  切り替えに必要な時間:10秒 ←(項目Bで使用)
HOST-FS2の金銭的コスト
  交換に必要なコスト: ¥120,000
SERVICE-FSのサービスレベル
  優先度: 7 (0-10)
  可用性: 99.999%
  …
SERVICE-Webのサービスレベル ←(項目Cで使用)
  優先度: 4 (0-10)
…
```

図 8 回避策候補における判断指標の情報収集・照合

表 1 選択段階における回避策選出のイメージ

回避策	実施時間	費用	SLA	時間内	結果
策 1	50 分	¥50,000	OK	NG	不適
策 2	30 分	¥150,000	OK	OK	適
策 3	30 分	¥150,000	NG	OK	不適
策 4	2 分	¥0	NG	OK	不適

- 予め設定した金銭的コストの限度
- サービスレベルの各基準，SLA

基準に合わない回避策候補は不適と判断され，最終的な回避策の提示には反映しない．

3.3.3 提示段階

基準に合わない回避策を消去したとしても，その中からどれが現状に最適な回避策なのか判断するのは容易ではない．そこで，熟練でない運用管理者でも最終的な障害回避策 (1 つ) を決定できるよう支援するため，評価値を算出してランキングを行う．

回避策候補 x の評価値 $Score$ の算出方法を次に示す．ここで，t を各コストの種類 (時間的コスト，金銭的コスト，サービス優先度など)，T をその集合，C をコストの値，B をコストの基準値 (適・不適を判断する上で最低条件となる値)，および W をコストの重みとおく．

$$Score_x = \sum_{t \in T} W_t(C_t x - B_t x)$$

これにより最適回避策は，最適策評価値を $Best$ とすると，

$$Best = max(Score_x)$$

となる回避策候補 x である．

考慮すべき点として，コストの種類ごとの重み W_t をどのように設定するかで最適策が変動する点が挙げられる．標準的な運用では，$W_t(B_t)$ が t ごとに一様になるような W_t を設定する事を想定しているが，例えばサービス優先度 $priority \in T$ をより強く反映したい場合は $W_{priority}$ を引き上げて調整するといった措置が考えられる．

3.3.4 運用管理者への回避策の提示

提案手法の出力として運用管理者へ最適策のみを提示するという方式も考えられるが，今回研究で目指しているのは熟練でない運用管理者の支援であり，自律運用ではない．そこで最終的な運用管理者への提示は複数の回避策を $Score_x$ で並び替え，各評価項目の評価結果も併せて提示する事を想定している．実際の運用管理者へ回避策の候補を示す様子を以下の図 9 に示す．

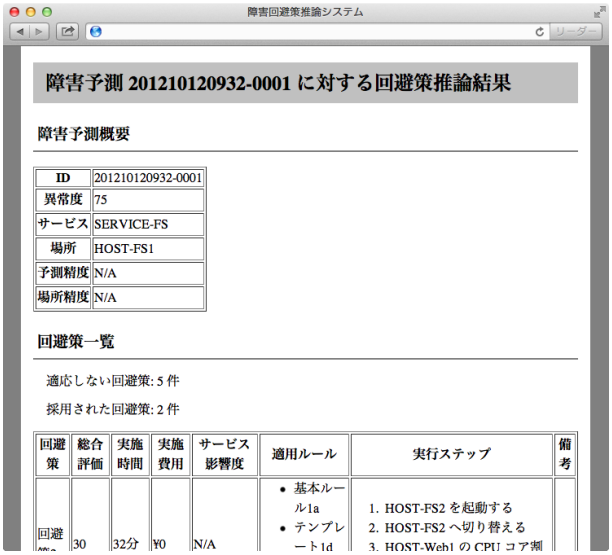

図 9 回避策提示法による提示例

回避策一覧に含まれる主な提示項目は，

- 回避策の内容 (実施手順)
- 評価値，評価値に最も影響を及ぼした評価項目[*3]
- 実施時間，費用などの評価項目の内容

これにより，もし熟練でない運用管理者が近い $Score_x$ 同士の回避策で迷った際には，他の運用管理者に相談して最終的な回避策を決定したり，またある程度スキルや経験を持った運用管理者が提案手法の出力を参考にしてより最適な回避策がないか検討したりといった運用支援が期待できる．

4. 考察

提案手法の利点について，提示された回避策の有効性，大規模化への対応，複雑化への対応の 3 つの観点から考察する．

[*3] $max(W_t(C_t x - B_t x))$ となるコストの種類 t を示す．

4.1 提示された回避策の有効性

本研究では，高いスキルや経験を持たない運用管理者でも障害回避に寄与できる事を目指し，予備実験で示したシステムに提案手法の最適回避策提示法を適用する事で，システムの構成や状態に応じて自動的に最適な障害回避策を提示可能である事を示した．提示段階では，サービスレベルやコストといった判断指標を考慮し，さらに評価値を算出・用いることで，経験の浅い運用管理者でも効果の見込める回避策を素早く把握，実行することが可能になった．

4.2 大規模化への対応

システムが大規模化した際の懸念として，ルール数やルールの関係が膨大になり管理しきれなくなる点を指摘した．予備実験で用いたエキスパートシステムの利点と，提案手法のルール生成自動化を組み合わせることで，実際に管理対象となるルール数を大きく減らせる事が期待できる．これにより運用管理者は，システムの構成や状態に強く依存しない基本的なポリシーの策定に集中することが可能となり，運用コスト削減や運用品質向上に寄与する．

4.3 複雑化への対応

今日の情報システムは構成や採用技術が複雑化し，また多様なニーズに応えるため運用ポリシーの複雑化も問題となっている点を指摘した．提案手法は基本ルールの策定・入力さえ済めば高度な運用管理者に代わって自動化する仕組みであるため，非機能要件・運用ポリシーの複雑化がルール管理へ与える影響を最小限にする事ができると考えられる．最終的な回避策提示段階においても，運用管理者はシステムの複雑性に関わらず評価指標の算出結果を比較しながら効果を判断可能である．

5. まとめ

本研究では，障害予測における最適な障害回避策の生成・提示法を提案した．熟練の運用管理者に任されていた障害回避の判断を自動化し，なおかつ大規模化・複雑化する情報システムにも適用可能な構成とすることで，障害回避措置を容易にし，可用性向上の機会拡大を実現する．

今後の課題としては，実際の障害予測技術を用いた実験の検証が挙げられる．本研究では，障害予測情報が正確であればあるほど，効果的な回避策を提示できると考えられる．しかし現状では，あらゆる情報システムに適用可能な汎用的で高精度な障害予測技術は確立されていない．そのため，既存の障害予測技術の中からなるべく精度の高いものを選択，組み合わせる必要がある．Gainaru ら [9] による統計的・データマイニング的手法を HPC システムに適用した例などを参考にしたい．

参考文献

[1] International Business Machines Corporation: Tivoli, http://www-01.ibm.com/software/tivoli/.

[2] Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., Ma, S., Vilalta, R. and Sivasubramaniam, A.: Critical event prediction for proactive management in large-scale computer clusters, *KDD* (Getoor, L., Senator, T. E., Domingos, P. and Faloutsos, C., eds.), ACM, pp. 426–435 (2003).

[3] Bodik, P., Friedman, G., Biewald, L., Levine, H., Candea, G., Patel, K., Tolle, G., Hui, J., Fox, A., Jordan, M. I., Patterson, D. A. and Patterson, D. A.: Combining Visualization and Statistical Analysis to Improve Operator Confidence and Efficiency for Failure Detection and Localization., *ICAC*, pp. 89–100 (2005).

[4] 日本ヒューレット・パッカード株式会社：HP Service Intelligence, http://www8.hp.com/jp/ja/hp-news/article_detail.html?compURI=tcm:191-1198948.

[5] International Business Machines Corporation: PureSystems, http://www.ibm.com/ibm/puresystems/.

[6] Hayes-Roth, F., Waterman, D. A. and Lenat, D. B.: *Building expert systems*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1983).

[7] Oracle Corporation: Java SE Overview - Oracle Technology Network, http://www.oracle.com/technetwork/java/javase/overview/index.html.

[8] 中村友洋：Web アプリケーションの障害を予測する “アクセス時間解析方式” の提案, 情報処理学会論文誌. コンピューティングシステム， Vol. 47, No. 12, pp. 349–357（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110004782253/⟩ (2006).

[9] Gainaru, A., Cappello, F., Fullop, J., Trausan-Matu, S. and Kramer, W.: Adaptive event prediction strategy with dynamic time window for large-scale HPC systems, *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, SLAML '11, New York, NY, USA, ACM, pp. 4:1–4:8 (online), DOI: 10.1145/2038633.2038637 (2011).