OUKA
The University of Osaka
Institutional Knowledge Archive

Title	情報システムの開発及び運用における知識活用手法に関する研究
Author(s)	工藤, 裕
Citation	大阪大学, 2011, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/1101
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

情報システムの開発及び運用における
知識活用手法に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2011 年 7 月

工 藤　裕

研究業績

A. 学術論文誌論文

1. 工藤裕, 平井千秋, 川辺博史, 降旗由香理, 大野治: “議事録を利用した設計レビュー管理システムの開発と評価”, 情報処理学会論文誌, Vol.44, No.5, pp.1404-1412 (2003).

2. 工藤裕, 森村知弘, 増岡義政, 薦田憲久: “情報システムの運用自動化に向けたポリシー記述形式とポリシー実行スケジューリング方式”, 電気学会C部門論文誌, Vol.131, No.10 (掲載予定) (2011).

B. 国際会議

1. Y. Kudo, C. Hirai, Y. Furuhata, T. Watanabe, and O. Ohno: “A Review-Report-Oriented Knowledge-Management System”, in *Proc. of the 3rd Workshop on Software Engineering over the Internet (in the 22nd International Conference on Software Engineering)*, pp.80-88 (2000).

2. Y. Kudo, C. Hirai, Y. Furuhata, T. Watanabe, and O. Ohno: “A Proposal of a Review-Report-Oriented Knowledge-Management Model”, in *Proc. of the 2nd World Congress for Software Quality*, pp.199-204 (2000).

3. H. Komuro, O. Ohno, Y. Furuhata, Y. Makuta, A. Harada, Y. Kudo, K. Yasuda: “Project Management Tool utilizing Review Reports for Software Development”, in *Proc. of the International Conference on Project Management*, pp.171-176 (2002).

C. 学会講演

1. 工藤裕, 小泉忍, 川辺博史, 降旗由香理: “ソフトウェアの分散開発拠点に対応した問題

点管理システム”, 情報処理学会 第 55 回全国大会 講演論文集(1), pp.439-440 (1997).

2. 渡辺正, 降旗由香理, 宮崎肇之, 工藤裕: “システム開発事例共有システムの開発”, 情報処理学会 第 57 回全国大会 講演論文集(4), pp.268-269 (1998).

3. 工藤裕, 平井千秋, 森田靖, 矢野理: “再利用支援データベース「AWARD」の開発”, 情報処理学会 第 61 回全国大会 講演論文集(4), pp.15-16 (2000).

4. 永井愛之, 矢野理, 森田靖, 高木勝則, 平井千秋, 工藤裕: “文書分類方法および文書再利用システム”, 情報処理学会 第 63 回全国大会 講演論文集(3), pp.3-4 (2001).

5. 村田大二郎, 工藤裕, 平井千秋, 伊野谷祐二, 三富篤: “ソフトウェア開発プロジェクト内での情報流通インフラの開発と評価”, 情報処理学会研究報告（グループウェアとネットワークサービス研究会）, Vol.2001, No.48, pp.35-40 (2001).

6. 村田大二郎, 平井千秋, 工藤裕, 藤波武起, 鐘ヶ江博: “ソフトウェア開発における約束懸案管理の方法論の提案”, プロジェクトマネジメント学会 2003 年度秋季研究発表大会予稿集, pp.156-159 (2003).

7. 工藤裕, 森村知弘, 菅内公徳, 薦田憲久: “障害原因解析のためのルール記述方法とその実行方式”, 電気学会 情報システム研究会, IS-09-71, pp.1-6 (2009).

8. 工藤裕, 森村知弘, 増岡義政, 薦田憲久: “業務システム動的構成変更のためのポリシー実行スケジューリング方式”, 電気学会 情報システム研究会, IS-10-74, pp.5-10 (2010).

D. その他

1. 平井千秋, 工藤裕, 降旗由香理: “事例 3:ナレッジマネジメントのソフトウェア開発への適用－日立製作所の事例－”, 人工知能学会学会誌 特集「ナレッジマネジメントとその支援技術」, Vol.16, No.1, pp.59-63 (2001).

2. 宮川伸也, 佐治信之, 工藤裕, 田崎英明: “ビジネスグリッド技術解説”, 情報処理, Vol.47, No.9, pp.953-961 (2006).

3. Y. Kudo and S. M. Batra: “Hitachi IT Operations Analyzer – Root Cause Analysis for Supporting Fault Identification”, Product White Paper, http://itoperations.hds.com/Documents/en/Product%20Resources/IT%20Operations%20Analyzer%20Root%20Cause.pdf (2009).

4. Y. Kudo and S. M. Batra: “Hitachi IT Operations Analyzer – Topological List View Enhances Ability to Monitor Complex IT Systems”, Product White Paper, http://itoperations.hds.com/Documents/en/Product%20Resources/Hitachi%20IT%20Operations%20Analyzer%20Topological%20List.pdf (2009).

内容梗概

本論文は，筆者が 1995 年から現在まで（株）日立製作所システム開発研究所，横浜研究所，ならびに 2009 年 10 月から現在まで大阪大学大学院情報科学研究科マルチメディア工学専攻在学中に行った，情報システムの開発及び運用における知識活用手法に関する研究成果をまとめたものである。

情報システムは，現在では経済活動や日常生活を支える社会基盤となっており，その品質向上，及び，信頼性向上は，これまで以上に重要な課題となっている。このような背景のもと，開発現場における設計事例や設計ノウハウなどの知識を活用して設計品質を高めたり，運用時の知識・ノウハウをポリシーとして形式化して運用を自動化することで，信頼性を向上するニーズが高まっている。本論文では，次の 3 点を課題として設定し，解決方法を検討する。

課題(1): 設計品質向上のための設計レビュー管理プロセスの効率化

課題(2): 障害原因解析における解析処理の高速化

課題(3): 運用自動化のためのポリシー記述の容易化

本論文は，全 5 章から構成される。

第 1 章の序論では，情報システムの開発と運用のそれぞれについて，知識を活用する際に考慮すべき点をまとめ，解決すべき課題を述べ，従来研究を概観するとともに，本論文の目的と位置付けを明らかにする。

第 2 章では，情報システムの設計開発において，知識やノウハウが日々扱われる設計レビューに着目し，そのプロセスを効率よく推進させるための支援方式について述べる。具体的には，設計レビューの議事録を活用することによって，決められた設計項目に対して十分な設計レビューを実施したか，レビューでの議論は充実したものだったかを確認する作業を支援する方式について述べる。その上で，現場開発者の意見に基づいた考

察，及び，レビューの十分性の評価にかかる時間について従来方式と比較し，提案方式の実用上の有効性を示す。

第 3 章では，障害原因解析方式について，解析処理時のオーバヘッドを低減する解析ルールのデータ構造と解析方式について述べる。解析ルールの条件式の要素をルール間で共通化して，条件要素とルール間に直接リンクを張ることによって，入力と条件のマッチングを高速化し，障害イベントの発火状態を一定時間保持させることで，障害イベントの再入力を行うことなく，より少ない計算量で解析を行う方式について述べる。さらに従来方式と比較することで，本方式の有効性を示す。

第 4 章では，ポリシー間の競合を回避するための実行スケジューリングについて，ポリシーの同時実行可否条件を業務システムの論理構成ツリーに基づいて，より容易に指定するための方式について述べる。さらに，指定した同時実行条件に基づいてポリシーの同時実行スケジューリングを行う方式について述べ，試作システムによる評価実験から，その有効性を示す。

最後に，第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる。

目次

第 1 章　序論 …… 1
1.1　研究の背景 …… 1
1.2　従来研究 …… 5
1.2.1　開発プロジェクトにおける知識活用の効率化に関する研究 …… 6
1.2.2　情報システムの障害原因解析技術に関する研究 …… 7
1.2.3　ポリシーベース運用自動化技術に関する研究 …… 8
1.3　研究の方針 …… 9
1.4　本論文の構成 …… 10
第 2 章　議事録を利用した設計レビュー管理支援方式 …… 13
2.1　緒言 …… 13
2.2　設計レビュー管理の現状と課題 …… 14
2.2.1　設計開発現場における設計レビューの位置付けとその管理方法 …… 15
2.2.2　レビュー計画立案(Plan) …… 15
2.2.3　レビュー実施，及び，議事録作成と配布(Do) …… 16
2.2.4　レビューの十分性(充実度と質)の評価(See) …… 18
2.3　設計レビュー管理の課題と解決のための支援方式 …… 18
2.3.1　設計レビューの課題 …… 18
2.3.2　課題を解決するための支援方式 …… 19
2.3.3　支援 1: 議事録フォーマットの策定と議事録作成支援 …… 21
2.3.4　支援 2: レビュー実施回数集計支援 …… 23
2.3.5　支援 3: 設計経緯の確認支援 …… 23
2.4　設計レビュー管理システムの設計 …… 24
2.4.1　システム構成 …… 24
2.4.2　議事録データベースの構造 …… 25
2.4.3　レビュー項目表管理部 …… 26
2.4.4　議事録雛形作成部 …… 26
2.4.5　議事録登録部 …… 27
2.4.6　レビュー実施回数集計部 …… 29
2.4.7　串刺し検索部 …… 30

2.5　提案方式の有効性に関する評価と考察 ……33
2.5.1　議事録の登録状況に基づく分析 ……33
2.5.2　設計レビュー管理の効率化に関する評価と考察 ……35
2.5.3　議事録作成工数，レビューの十分性の評価にかかる時間の評価 ……38
2.5.4　本システムの実用上の有効性に関する評価 ……41
2.6　結言 ……41
第 3 章　情報システムの障害原因解析のためのルール記述方法と解析実行方式 ……43
3.1　緒言 ……43
3.2　障害原因解析システムの概要 ……44
3.2.1　情報システムの障害原因解析 ……44
3.2.2　従来技術による障害原因解析 ……47
3.2.3　RCA システム ……47
3.3　汎用解析ルールの記述方法 ……50
3.3.1　汎用解析ルール記述の文法 ……50
3.3.2　汎用解析ルールの作成手順 ……53
3.4　汎用解析ルールとトポロジーによるルールメモリ構築方法 ……55
3.4.1　ルール展開処理の動作概要 ……55
3.4.2　ルール展開処理手順 ……56
3.4.3　ルールメモリのデータ構造 ……56
3.5　機器の状態異常の検出方法 ……59
3.6　ルールメモリに基づく解析方式 ……60
3.6.1　解析方式の要件 ……60
3.6.2　結論に対する確信度の計算 ……61
3.6.3　障害イベントの生存期間と発火状態のキャンセル処理 ……62
3.6.4　ポーリング間隔と障害イベントの生存期間の関係 ……63
3.6.5　異常と正常を繰り返す場合の検出方法とルール記述方法 ……65
3.7　ユーザインタフェース ……66
3.8　提案方式の有効性に関する評価と考察 ……66
3.8.1　障害原因解析のためのイベント処理時間の評価 ……67
3.8.2　累積解析処理時間に関する評価 ……70
3.8.3　解析ルールの事前展開に関する考察 ……72
3.9　結言 ……73

第 4 章　情報システムの運用自動化に向けたポリシー実行スケジューリング方式 ……75
4.1　緒言 ……75
4.2　ポリシー制御システムの概要 ……76
4.2.1　ポリシー制御システムの基本動作 ……76
4.2.2　ポリシー制御のシナリオ例 ……76
4.3　ポリシー実行制御機能の課題と解決方針 ……78
4.3.1　ポリシー実行制御機能の課題 ……78
4.3.2　課題を解決するポリシー実行制御方式 ……81
4.3.3　優先度を考慮した実行スケジューリング ……84
4.4　試作システムの設計 ……84
4.4.1　試作システムのアーキテクチャと処理概要 ……84
4.4.2　ポリシー実行のスケジューリング ……86
4.5　提案方式の有効性に関する評価と考察 ……87
4.5.1　ポリシー実行制御機能の性能の評価 ……87
4.5.2　ポリシー総記述工数の評価 ……90
4.5.3　業務システムの論理構成ツリーの利用に関する考察 ……91
4.6　結言 ……95
第 5 章　結論 ……97
5.1　本研究のまとめ ……97
5.2　今後の課題 ……98
謝　辞 ……99
参考文献 ……101

第1章

序論

1.1 研究の背景

コンピュータの処理能力の飛躍的な向上やインターネット技術の普及により，近年の情報システムは，情報通信，金融，交通，電力，ガス，水道，医療など，経済活動や日常生活を支える社会基盤となっている。そのため，情報システムの障害によるサービスの停止やサービス品質の低下は，これら経済活動や日常生活に大きな影響を与える[1][2][3]だけでなく，社会問題として発展する可能性も出てきており，情報システムの品質向上，及び，信頼性向上はこれまで以上に重要な課題となっている。

情報システムのライフサイクルは，要求仕様の定義から必要な機能の設計，実装までの開発フェーズと, 実際に情報システムを稼働させる運用フェーズに大別できる[4]。開発フェーズでは，情報システムの要件を定義し，設計，開発，構築，テストを行って情報システムを完成させる。これらの作業は，人間による知的作業が中心であり，開発フェーズ全体で数ヶ月を要するなど長期にわたる。そのため，開発成果物の品質を高めるために，過去の事例や各段階での中間成果物，各メンバーが持つ経験や知識を活用することが重要となる。一方，運用フェーズでは，情報システムを稼働環境に構築し，運用・保守を行う。IT (Information Technology) 機器の保守作業，システム異常時の復旧作業など，複雑で手間のかかる作業が中心となる。そのため，このような作業を間違いなく遂行するための知識・ノウハウを形式化し，計算機に実行させられるようにしていくことが有用となる。

開発フェーズに関しては，近年のソフトウェアの高度化，大規模化によって，開発プロジェクトの管理や開発プロセス改善の重要性が認識され，PMBOK (Project Management Body of Knowledge)[5]や CMM (Capability Maturity Model)[6][7] /CMMI (Capability Maturity Model Integration)[8], ISO (International Organization for Standardization) 9000 などの知識体系や枠組みが注目されている。これらは，プロセス

ベースの体系であり，手順にしたがって処理を進めていくことを前提とした方法論である。そのため，プロセスの進捗や成果物の出来高の計測により，計画と実績の差分を把握することによって，プロセスを改善していく考え方を基本としている。多くの企業では，このような開発方法論が統合された CASE (Computer Aided Software Engineering) ツールと呼ばれるソフトウェアを導入し，成果物の登録状況，設計やプログラム開発の進捗情報，プロジェクト内の問題点情報などを可視化することで，開発プロセスの改善に役立てている[9][10]。近年では，設計開発の対象となる情報システムの規模が，一人ではすべてを把握できないほど大きくなっている。そのため，設計開発に必要なあらゆる情報，例えば，顧客提案時の要件定義資料，仕様書，設計書，レビュー議事録などのドキュメント類，ソースコードやデータ定義などの開発成果物，問題点管理票やプログラム変更管理票などの管理帳票類について，必要な時に必要な箇所が即座に閲覧できるように，これらの情報を関連付けることで閲覧性を高める研究が進められている[11][12][13][14]。

しかし，情報システムをフルスクラッチで開発する時代とは異なり，近年では，サードベンダが提供する OS (Operating System)やミドルウェアの利用が一般的となっている。ところが，これらには，仕様書やマニュアルは存在するものの，実際にはその通りに動作しないことが多い。仕様通りに動作しなければ，それらをベースにして構築した情報システムは，信頼性の観点で，顧客に提供できなくなってしまう。そのため，それらを取り込んで開発する開発者が，独自に試行錯誤を行って，それらの挙動を調べ上げなければならなくなる。しかし，同じソフトウェアを前提にする開発プロジェクトが他にも存在し，そこで得たノウハウを提供してもらうことができれば，品質向上や生産性向上に役立たせることができる。このような発想のもと，WWW (World Wide Web)技術などを利用して，分散開発拠点間でソフトウェアの利用ノウハウを共有したり，開発事例や過去の開発物の再利用情報を共有するための研究が進められている[15][16][17][18][19]。

しかしながら，これらの研究で提案されている開発事例の共有や再利用支援では，共有される情報が終了したプロジェクトの情報であったり，完成後の仕様書・設計書であるなど，情報の鮮度の点で問題がある。つまり，日々の検討，日々の開発の中から生まれる知識・ノウハウを効率的に形式化して，確実，且つ，効率的に活かしていくための枠組みが重要であるが，現在行われている研究では，これを支援する方式に

ついての検討が十分に行われてはいない。

一方，運用フェーズに関する研究の関心事は運用コストの低減である。経済産業省発行の 2006 年版 情報処理実態調査報告書[20]にあるように, 企業における情報処理投資のうち，既存システムの運用保守にかかる費用は約 7 割を占めており，今後も増加傾向にある。情報システムの運用とは，さまざまなサービスを提供しているシステムが停止することなく，利用者に対してサービスを提供できるよう当該環境を維持管理することである。サービスが提供できなくなる事象の発生を防止する作業や，サービスを提供するために必要となる日々の作業，及び，発生してしまった事象に対して迅速に対処する作業が必要となる。このような運用作業については，提供するサービスの信頼性要件や性能要件が運用現場毎に異なることから，運用管理者のノウハウへの依存度が高い。そのため，ITIL (Information Technology Infrastructure Library)[4]のような運用作業のベストプラクティスを取り込み，現場ごとにカスタマイズすることによって運用の効率化を図っているのが現状である[21]。しかしながら，人手作業による効率化は限定的であるため，人手で行ってきた運用プロセスを計算機に代行させる運用自動化技術が注目を集めている。情報システムの運用自動化については，2003 年頃から IBM 社のオートノミックコンピューティング[22][23][24][25]を中心に，各所でさまざまな研究が進められている。オートノミックコンピューティングでは，対象の情報システムの状態維持を，監視・分析・計画・実行のループによって行う。このループを計算機, すなわち運用管理システム[26][27][28][29]に実行させるための指針をポリシーと呼ぶ。ポリシーは，対象システムがサービスを提供できなくなる事象が発生しそうな状況かどうか，または，そのような事象が発生しているかどうかを判定する条件と，発生しそうな場合や発生している場合に，その状態から回復するためのアクションで構成される。この条件とアクションの組を定義したものをポリシー記述と呼ぶ。ポリシー記述は，監視・分析・計画・実行のそれぞれについて，運用管理者の知識・ノウハウを反映して作成する。

しかしながら，現在のポリシーベース運用自動化技術では，監視・分析・計画・実行のループを, 人間が行う場合と同じように実行させることは簡単ではない。例えば，監視・分析については，人間であれば，問題が発生した IT 機器だけでなく，それと接続関係にある周辺機器にも調査範囲を拡大し，さらに，確認するメトリックの種類や時間間隔を変化させるなどして，試行錯誤的に調査を行うことができる。しかし，ポ

リシーによって原因調査を自動化する場合は，あらかじめ定義した範囲や手順でしか調査することができない。計画・実行についても同様に，人間であれば，他の作業者の作業状況を把握して，管理対象全体を考慮した作業の優先度付けを行うことができるが，ポリシーによる自動化では，他のポリシーの実行状態を考慮してアクションの実行を制御することは難しい。

一方で，計算機による自動化は，人間には不可能なことを可能にする。例えば，監視・分析については，情報システムを構成する多数のサーバやストレージ，ネットワークスイッチの接続関係を短時間で検出し，それぞれの IT 機器の稼働性能状態を計測することができる。また，接続関係と稼働性能状態を組み合わせることで，障害の原因を特定することができる。また，計画・実行では，多数の IT 機器に対するバックアップなどの運用操作を同時並行に実行できる。このように，運用自動化技術の研究においては，定型的な作業の自動化の利点を損なわない形で，運用管理者のノウハウを如何に組み込むかが重要となる。

監視・分析・計画・実行のループのうち，監視・分析では，対象システムの状態異常を検知し，原因を特定する方式の研究が中心となる。この障害原因特定の方式は大きく 2 つに分けられる。1 つは，対処事例をデータベースに蓄積し，障害発生時の情報システムの状態に基づいてデータベースを検索し，運用管理者に該当する対処事例を提示する方式である。この方式では，対処事例が蓄積されていなければ利用できないため，対処事例データベースの構築工数が課題となる。もう 1 つは，情報システムの障害伝播に関する知識・ノウハウを解析ルールとして記述し，エキスパートシステムなど知識処理に基づくアルゴリズムによって，障害の原因を推論する方式である。複数の障害イベントの関連付け(コリレーション処理)によって障害原因を特定する方法であり，障害原因解析(RCA: Root Cause Analysis)技術として，多くの研究が行われている[30][31][32]。しかしながら，RCA に用いられるルールベースシステムでは，対象システムの構成(トポロジー)に合わせて解析ルールを構築する必要があるため, 解析ルールの作成にコストがかかる。また，情報システムの規模が大きくなると，解析しなければならない障害イベントの数が増大するため，解析処理に時間がかかり，実用に耐えられないなどの課題が残っている。

また，監視・分析・計画・実行のループのうち，計画・実行では，検出した異常の内容に応じて，異常を修復するためのアクションを計画・実行する方式の研究が中心

となる。ところが，1 つの情報システムに複数のポリシーを設定した場合には，条件によっては複数のポリシーが同時に実行されることになり，アクションの対象や情報システムの構成によっては，ポリシー設定者の意図と異なる結果を生むことがある。そのため，同時実行に関する制約条件の指定が容易に行えるポリシーの記述形式と，ポリシー実行制御が現実的な時間で行える実行スケジューリング方式の確立が課題である。

本研究では，このような状況を考慮し，図 1-1 のように位置付けられる以下の課題を解決することを目的とする。

課題(1): 設計品質向上のための設計レビュー管理プロセスの効率化

課題(2): 障害原因解析における解析処理の高速化

課題(3): 運用自動化のためのポリシー記述の容易化

フェーズ	開発		運用	
活用者	設計者，開発者	管理者	運用管理者	
活用のタイミング	設計時，開発時	レビュー計画の立案時	監視→分析	計画→実行
活用すべき現場知識ノウハウ	・前提ソフトの利用ノウハウ ・仕様の実現方式(設計内容) ・設計の経緯(採用案と却下案)	・進捗遅延からの挽回方法 ・問題解決方法 ・設計管理のやり方	障害箇所特定ノウハウ(装置間障害伝播知識)	・運用のベストプラクティス ・システムの構成要素間の依存関係と構成変更時の注意点
記録形式記述形式	・開発ノウハウ共有メモ ・仕様書，設計書 ・レビュー議事録	・管理ノウハウ・メモ ・問題点管理票 ・設計項目一覧	解析ルール	ポリシー記述
課題	(1) 設計品質向上のための設計レビュー管理プロセスの効率化		(2) 障害原因解析における解析処理の高速化方式	(3) 運用自動化のためのポリシー記述の容易化

図 1-1 情報システムの開発及び運用における知識活用の研究課題

1.2 従来研究

本節では，開発及び運用における知識活用に関する，従来研究のアプローチとその問題点について述べる。

1.2.1 開発プロジェクトにおける知識活用の効率化に関する研究

前節で見たように，開発プロセスを，IT の利用によって効率化するための研究が行われている。例えば，成果物の登録状況，設計やプログラム開発の進捗情報，設計上の問題点やプログラムのバグに関する情報から，プロジェクトの進行状況を可視化する研究である。可視化の対象は，ソフトウェア工学やプロジェクト管理手法で扱われているプログラムの欠陥密度や修正工数，プログラム・コードの変更量，開発者の作業時間などであり，CASE ツールから自動で各種データを収集・分析を行うことで可視化し，改善サイクルの速度を上げることを主目的としている。ただし，これらのデータは定量的な指標が中心で，ソフトウェアの内容に関する知識の活用に関する検討はなされていない。このようなプロジェクト管理支援環境では，設計ドキュメントや設計レビューの議事録のような設計の内容や設計の経緯が記載されたドキュメントも共有されているため，これを参照することによって知識活用が促進されることもある。しかし，例えば，設計レビューの特徴，すなわち，1 つの設計項目については数回のレビューを要し，一回のレビューでは複数の設計項目が討議されるという特徴を考慮した議事情報の提示・共有を支援するなど，設計ノウハウを活用するという観点での検討は十分になされていない。

また，ソフトウェア設計に関する議論過程をモデル化して可視化することで，設計ノウハウとして活用するための研究がある[33][34][35][36]。これらの研究では，設計上の意思決定とその根拠(Design Decisions and Design Rationale)に関する情報を IBIS (Issue Based Information System)と呼ばれるモデルで表現し，新たなソフトウェアを設計する際の議論と意思決定に利用できるとしている。しかし，データ入力の手間が大きく，議論構造が複雑になると参照が困難になるという点で，一般に時間に追われがちなソフトウェア開発プロジェクトにおいては，現実的ではない。

また，一般的な企業活動における業務知識の共有のために，XML (eXtensible Markup Language)を利用して，蓄積されたメモやドキュメント間のリンク付けを行う方式[37][38][39]が提案されている。しかしながら，設計レビューの効率向上という観点では，レビュー計画に対する実施管理の容易性向上，及び，設計項目に対するレビューの充実度の確認作業に対する容易性向上についての検討はなされてはいない。

以上のことから，情報システムの開発フェーズにおいて，プロジェクト管理者と設

計者の作業の手間を低減し，知識・ノウハウの活用を効率化する現実的な方式が十分に検討できているとは言えない。

1.2.2 情報システムの障害原因解析技術に関する研究

知識処理に基づいたアルゴリズムを使用して，複数の障害イベントの関連付け(コリレーション処理)を行い，障害原因を特定する障害原因解析技術が注目されている。障害原因解析技術は，古くは製鉄工場などプラントにおける障害原因箇所の特定を主目的にエキスパートシステム[40][41]として発展し，現在では，情報システムを対象として研究が進められている。適用対象は異なるものの，推論方式として用いられている技術は，基本的には，Forgy の Rete アルゴリズム[42]をベースとしており，ルールやデータの依存関係をノードのネットワークから構成される内部表現に置き換えることが必要となる。そのため，大規模化や複雑化が進む情報システムへの適用を考えた場合には, 対象システムの構成(トポロジー)に合わせて解析ルールを構築する工数が大きな問題となっている。

また，情報システムの障害原因解析に用いられるイベントコリレーションの手法として，症状と原因を縦横の軸にとったマトリックス表現による方式が注目を集めている。Code Book Correlation Technology[43][44]と呼ばれ，EMC 社の障害解析製品に適用され，実用化されている。しかしながら，管理対象の規模が大きくなると，マトリックスのサイズが大きくなり，解析処理に時間がかかるという問題がある。

一方，サーバ間の部品の依存関係に基づくログ管理支援方式や障害通知方式に関する研究[45][46][47][48]がある。障害が発生したサーバを起点にして部品の依存関係を辿ることで，障害の影響範囲を特定し，運用管理者への通知レベルやログの管理方法を変化させる方式である。しかし，これらの研究では複数の障害を別々に扱っており，1 つの障害が複数の障害を引き起こした場合に，依存関係に基づいて関連を行い，原因を特定する方式については，述べられていない。

さらに，標準化団体 OASIS (Organization for the Advancement of Structured Information Standards)内に 2010 年 2 月に設立された SAF (Symptoms Automation Framework) TC (Technical Committee)[49]では，情報システムが示すさまざまな兆候とその対策の関係を知識として蓄積するためのフレームワークを標準化する作業が進められているが，具体的な適用事例はまだ存在していない。

1.2.3 ポリシーベース運用自動化技術に関する研究

ポリシーベース運用自動化技術は，ネットワークの QoS (Quality of Service)制御[50]をはじめとするネットワーク管理分野や，アクセス制御をはじめとするセキュリティポリシーの分野で広く研究されている。また, 今日では, サーバの負荷分散手法[51] やネットワークアプリケーションの応答性能を維持管理するためのサービスレベル管理手法[52]など，情報システムの運用管理分野に広く適用されている[53][54][55][56][57][58][59]。IBM 社は, オートノミックコンピューティング[60]という，生物が無意識のうちに心拍や温度を保つ自律神経系の働きを情報システムにも適用するコンセプトを提示している。このコンセプトでは，情報システムの状態を自律的に維持するために，監視・分析・計画・実行のループを繰り返す。つまり，システムの稼働状況を監視し，システムに対する負荷変動や障害が発生した場合に，あらかじめ設定したポリシーに基づいて，リソースの再割当てや切替えなどの対処を自動的に行う。近年のサーバ仮想化技術の発達により，IT リソースの再割当てや切替えなどのシステム構成変更は，以前の物理サーバを対象にしたものよりも容易に行えるようになっている。しかし, そこで使われるポリシーは, 情報システムの応答時間や CPU (Central Processing Unit)利用率，メモリ使用量などが閾値を超えた場合に，アラートを出すといった比較的単純なものであり，基本的にはサーバ仮想化技術の出現前後で変わっていない。

ポリシー記述言語としては，DMTF (Distributed Management Task Force)の CIM (Common Information Model) を用いてモデル化された管理対象に対してポリシーを記述する CIM-SPL (CIM - Simplified Policy Language) [61][62]や，PCIM (Policy Core Information Model) [63]などがある。いずれのポリシー記述言語でも，1 つのポリシー記述には，ポリシーの動作条件と，その条件を満たした場合に実行する対処アクションを IF-THEN 形式で記述する。ポリシーの動作条件には，情報システムの応答時間や CPU 利用率，メモリ使用量などの値に対する閾値や，システムの状態名を記述する。複数のポリシーを 1 つの情報システムに対して設定したい場合は，ポリシー記述を複数作成する。しかし，1 つ 1 つのポリシーは正しく設定できたとしても，複数のポリシーの動作条件が同一であった場合や，システム構成変更の対象が同一であった場合には，複数のポリシーが同時に動作を開始し，同じ IT リソースに対して変更操作を実

行してしまうため，ポリシー設定者の意図通りの結果とならないこともある。

イベントと条件とアクションの組みで表現する ECA ルール(Event Condition Action Rule)の並行実行制御に関しては，古くより，Active Database Management System におけるトランザクション処理の並列実行制御[64][65]の分野で議論されてきた。しかし，これらの研究は，ルール作成者が制御すべき条件をすべて把握していることを前提としてルールの記述能力を高める研究であり，運用現場において複数の別々の運用管理者が 1 つの業務システムに対して容易に安全なポリシーを設定することは想定されていない。

1.3 研究の方針

前節までに述べた課題を踏まえ，それぞれの課題を解決し，情報システムの開発及び運用における知識活用手法の高度化を図る。図 1-2 は，上記課題に対する本研究の位置付けを整理した図である。以下，図 1-2 にしたがい，本研究の方針について説明する。

情報システムの設計の質を高めるには，仕様書や設計書，プログラム・コードなどの成果物の管理だけではなく，個人の持つ知識やノウハウを組織的に増幅し，実現化させていくプロセスも重要である[66][67][68][69][70]。本研究では，情報システムの設計開発において，知識やノウハウが日々扱われる設計レビューに着目し，そのプロセスを効率よく推進させるための支援方式を提案する。具体的には，設計レビューの議事録を活用することによって，決められた設計項目に対して十分な設計レビューを実施したか，レビューでの議論は充実したものだったかを確認する作業を支援することで，レビューのプロセスを効率化する。

情報システムの障害原因解析に関しては，解析に必要な障害イベントがすべて揃わなくても結論を導くことができ，且つ，管理対象の規模が大きくなっても解析時間が増加しない解析ルールのデータ形式と解析処理方式を提案する。具体的には，エキスパートシステム等の推論アルゴリズムとして用いられてきた Rete アルゴリズムを情報システムの障害解析に応用することで，把握できている障害情報のみから，障害原因箇所を高速に推論する方式について議論する。

情報システムの稼働状態を維持するポリシーベース運用自動化システムでは，同時

並行に複数のポリシーを実行させると，ポリシー設定者の意図と反する結果となることがある。この問題を回避するためには，同時並行に実行させるべきではないポリシーの ID (Identifier)をポリシー記述で指定する方式が一般的である。しかしながら，新規ポリシーを追加する際の調整工数が膨大になるため，対象となる情報システムの構成をモデル化した業務システムの論理構成ツリーに対して，各ポリシーから同時実行条件を指定するポリシー記述形式を提案する。また，ポリシー実行スケジューリングに関しては，同一の業務システムに対して複数のポリシーが同時に実行可能な状態になったとき，それぞれのポリシーの対象リソースの範囲の重なりを業務システムの論理構成ツリーの構造に基づいて判定し，同時に実行させるか，1 つずつ実行させるかを決定するポリシー実行スケジューリング方式を提案する。

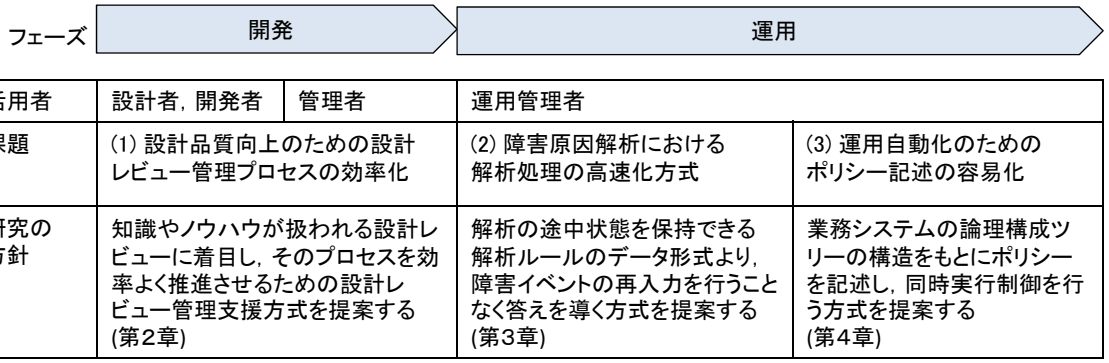


図 1-2 情報システムの開発及び運用における知識活用のための要件と研究方針

1.4 本論文の構成

本論文では，2 章以降を以下のように構成する。

第 2 章では，文献[71][72][73][74][75]に基づき，設計レビュープロセスを効率良く推進させるための設計レビュー管理支援方式を提案する。はじめに，研究対象とした設計開発現場での設計レビュー管理の方法を説明し，設計レビュー管理における議事録の利用方法，及び，議事録を利用したレビュー管理支援の課題について述べる。次に，課題を解決する支援方式を提案し，この方式に基づいて開発した設計レビュー管理支援システムについて述べる。さらに，本システムの利用者の意見に基づいた考察と，

議事録の作成工数，レビューの十分性を確認する時間による定量的な評価により，本システムの有効性を評価する。

第 3 章では，文献[76][77][78]に基づき，情報システムの障害原因解析のための解析ルールのデータ形式と解析処理方式について述べる。はじめに，障害原因解析システムの概要と，その実現に向けての課題，本研究のアプローチを述べる。その上で，提案する解析ルールの記述方法，解析ルールをトポロジーに合わせて展開する方法について述べ, 機器の状態異常の検出方法と解析処理の実行方式について述べる。さらに，ユーザインタフェースについて説明し，最後に，提案方式の評価と考察を行う。

まず，解析ルールの条件式の要素を，ルール間で共通化してオブジェクト構造化して，条件要素とルール間に直接リンクを張ることで，解析処理時のオーバヘッドを低減するデータ形式について説明する。さらに，障害イベントの発火状態を一定時間保持することで，障害イベントの再入力を行うことなく，確信度を計算して答えを導く方式について説明する。従来手法として実用化されているルールマトリクス方式と比較して，少ない計算量で障害原因解析処理が行えることを示す。

第 4 章では，文献[79][80][81]に基づき，業務システムの運用自動化のためのポリシーについて，複数のポリシー間の同時実行条件を容易に指定できるようにするためのポリシーの記述形式，及び，そのポリシー記述に基づく実行スケジューリング方式について述べる。まず，ポリシーの同時実行条件を，ポリシーを設定する業務システムの論理構成ツリーに基づいて指定する方法について説明する。次に，監視対象の業務システムの状況変化により複数のポリシーが同時に実行可能な状態になったとき，それぞれのポリシーの対象リソースの範囲の重なりを業務システムの論理構成ツリーの構造に基づいて判定し，同時に実行させるか，1 つずつ実行させるかを決定するポリシー実行スケジューリング方式について説明する。試作システムを実装し，実際にポリシーを作成，動作させることで，ポリシーの記述工数が低減できること，1,000 業務システムが稼働する大規模環境に対してもポリシー実行スケジューリングが実行可能であることを示す。

第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる。

第2章

議事録を利用した設計レビュー管理支援方式

2.1 緒言

本章では，情報システムの設計開発において知識やノウハウが扱われる設計レビューに着目し，そのプロセスを効率よく推進させるための支援方式を提案する。

設計レビューの目的は，複数の設計案から最適案を決定すること，及び，設計内容の品質を高めることである。設計作業は，開発組織内で定められた開発基準に基づいて進められる。設計レビュー管理では，はじめにレビューすべき設計項目を定義し，この設計項目に対するレビューの実施状況や必要な設計者の出席状況などを，設計レビューの議事録に基づいて確認する。

設計レビューの議事録は，次のように作成され管理される。レビュー後，議事録担当者によりテキスト形式で電子的に作成される。担当者は作成した議事録をプロジェクトの共有フォルダに格納すると共に，電子メールで関係者へ送付する。顧客への提示のため，印刷した議事録も作成し，プロジェクト管理者の承認印，顧客の承認印を押し，ファイルに綴じて保管する。過去の議事録を参照する際は，共有フォルダに格納された電子的な議事録ファイルを全文検索機能によって探し出して参照するか，紙に印刷された議事録を手捲りで探して参照する。

議事録は，設計の経緯や品質の良い設計を設計ノウハウとして共有するために，発言集という形ではなく，設計項目(議題)ごとに，結論と結論に至った理由を記載する。独自開発した議事録作成アプリケーションや表計算ソフト上で作られた議事録作成ツールが利用可能であるが，手間がかかるため，ほとんど利用されておらず，単にテキスト形式で作成し電子メールで配布するように簡略化するプロジェクトが多い。

プロジェクト管理者は，レビューの十分性の評価を行う。このとき，定められた設計項目に対応する議事内容が記載されていることを確認する必要があるが，1つの設計項目については，一回のレビューで結論が出ることは少ないため，複数の議事録を

参照しなければならない。そのための手間が大きいという課題がある。

そこで，本研究では，設計レビューの議事録を活用することによって，決められた設計項目に対して十分な設計レビューを実施したか，レビューでの議論は充実したものだったかを確認する作業の手間を軽減する設計レビュー管理支援方式を提案する。具体的には，設計レビューの議事録を設計項目単位に分割して登録するようにし，分割登録された各設計項目に対応する議事内容部分と，設計レビューの計画表であるレビュー項目表の各項目との突合せ処理により，レビュー項目表に対するレビューの実施回数を表示する。また，レビューの質の評価を行う際の設計経緯の確認作業の容易化のために，議題毎に分割格納された議事録データベースから，特定項目に関する議題内容のみを抽出する方式を提案する。

以下，2.2 節で，研究対象の設計開発現場で実施されている設計レビュー管理の方法について説明し，設計レビュー管理における議事録の利用方法について述べる。2.3 節では，議事録を利用したレビュー管理の課題について述べ，課題を解決する支援方式を提案する。2.4 節では，提案方式を実際の設計現場に適用するために構築したシステムの設計について述べ，2.5 節で，現場開発者の意見に基づいた考察と，議事録の作成工数，レビューの十分性を確認する時間による定量的な評価により，提案方式の有効性を評価する。

2.2 設計レビュー管理の現状と課題

ソフトウェアの品質向上は，一般に，設計レビュー[82][83]とテスト&デバッグによってなされる。設計レビューは，設計フェーズの成果物(主に設計ドキュメント)を評価し，技術的な問題を解決することで品質を向上させる作業である。また，テスト&デバッグは開発フェーズで実装されたプログラムを評価して不良除去・品質確認を行う作業である。不十分な設計は後工程のプログラム実装に影響を及ぼすため，設計の品質向上施策としての設計レビューは重要な活動[84][85][86]の 1 つであると言える。

本節では，実際の設計開発現場に対して行ったヒアリング結果に基づき，設計開発現場における設計レビューの位置付けとその管理方法について述べる。

2.2.1 設計開発現場における設計レビューの位置付けとその管理方法

設計レビューでは，技術的なトピックと管理的なトピックの両方を扱っている。技術面での目的は設計の妥当性の確認と技術的問題の解決であり，管理面での目的は設計の進捗とアクションアイテムの設定である。設計レビューの管理にあたっては，Plan-Do-See モデルを導入しており，図 2-1 に示すように，レビュー計画を立案し(Plan)，計画に沿ってレビューを実施して，議事録を作成，配布する(Do)。次に設計経緯の確認とレビューの十分性の評価を行っている(See)。

以降，この Plan-Do-See のそれぞれで実施されている内容について詳細に説明する。

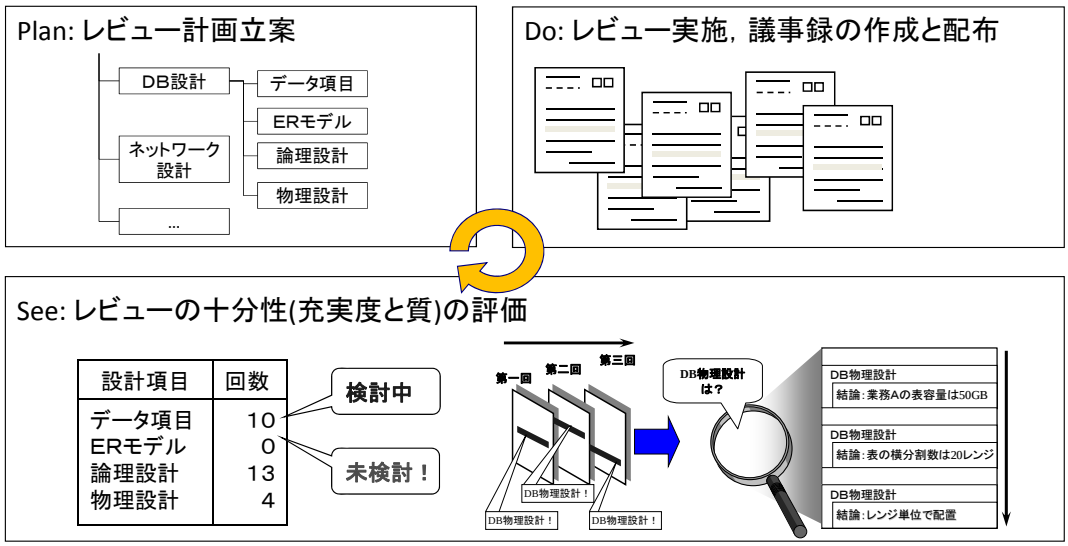


図 2-1 設計レビュー管理の Plan-Do-See モデル

2.2.2 レビュー計画立案(Plan)

レビュー計画の立案にあたっては，ソフトウェア品質特性[87]への網羅性を考慮し，ソフトウェア設計でレビューすべき項目(レビュー項目)の一覧表(レビュー項目表)を作成している。設計レビューでは，広く浅い議論が行われる場合もあれば，狭く深い議論がなされる場合もあり，議題の頻度にばらつきがある。これらの議論の状況を的確に把握し，効率良く設計・検討が進められるように，レビュー項目表の形式は，レビュー項目を階層的に表現できるツリー形式を採用している。設計開発現場で実際に

利用しているレビュー項目表を図 2-2 に示す。レビュー項目表には，大項目，中項目，小項目の 3 段階でレビュー項目が定義されている。また，1 番右のカラムには，大項目毎に，設計レビューの対象となるドキュメントが定義されている。

また，レビュー項目表は，設計状況によってレビュー項目に過不足が生じることがあるため，プロジェクト進行中であっても，レビュー項目を追加または削除してもよいことになっている。設計部門，生産技術部門，品質保証部門などで異なるレビュー項目表を用意して，レビュー実施状況をそれぞれの立場から確認できるようにしている[88]。

大項目	中項目	小項目	ドキュメント
システム構成設計	1.ハード・ソフトェアの決定		1.1.アプリケーション処理方式図※
	(C/Sシステム開発手順書参考)		1.2.ハードェア構成表
	2.開発言語・支援ツールの決定		1.3.ハードェア構成図
	(C/Sシステム開発手順書参考)		1.4.ソフトェア構成表
業務機能設計	1. システム要件の洗い出し	1.1.既存資産の流用	1.1.SDF
		1.2.ハード/ソフトの指定	1.2.入出力情報一覧表
		1.3.平常時/ピーク時のデータ量	1.3.業務機能定義書
		1.4.平均/ピーク時のレスポンスタイム	1.4.業務機能構成図
		1.5.外部システムとの接続性	1.5.業務処理概要図
		(通信規約　データフォーマット　タイミング等)	1.6.コンピュータ処理内容定義書(HIPO図) ※
		1.6.外部システムとの性能要求	1.7.システム要件定義表※
		(外部システムとのレスポンス 処理時間 業務量)	
		1.7.コンピュータセキュリティ	
		1.8.制度的対応	
		(システム監査　法制対応)	
開発方針・プロジェクト規約の作成	1.開発環境・稼働環境に関する基準	1.1.ライブラリ(PBL含む)管理	1.1.アプリケーション開発基準書
		1.1.1.バージョン管理	
		1.1.2.構成管理	
		1.2ドキュメント設計管理	
		1.3.ネーミングルール	
		1.4.コメント記述ルール	
	2.ユーザインタフェースに関する基準	2.1.画面・帳票レイアウト	
		2.2.画面操作方法	

図 2-2 レビュー項目表

2.2.3 レビュー実施，及び，議事録作成と配布(Do)

情報システムの設計は，機能要件，性能要件，コスト，使い勝手など，全体のバランスをとりながら決定する必要があるため，各部門の責任者や有識者を交えて，関連性のあるレビュー項目(例えば，システム処理方式設計とネットワーク構成設計など)をあわせてレビューしている。また，設計ドキュメントなどの成果物が一通り完成したときには，品質保証部門の担当者を交えて成果物のチェックを行っている。議事録には，議論した設計項目名とその結論，結論に対する理由を漏れなく明記されている。

レビュー議事録の例を図 2-3 に示す。図 2-3 の左上は，レビュー議事録の表紙の例であり，配布先，顧客名，システム名，実施日，実施場所，ミーティングのタイトル，概要，対象ドキュメント，今後の検討項目，主な成果，出席者などを記入するようになっている。また，図 2-3 の右下は，レビュー議事録の次紙の例であり，議事内容を，テーマと呼ばれる設計項目毎に，テーマ，結論，理由の組で記入される。

システム(デザイン、デシジョンレビュー報告書)

受付DR番号	
(マス), (コイミ), (平井), 203uインフラグループ	CBExplorer_0007

部長	課長	担当

1. 顧客名	(情公共)		11. 出席者
2. システム名	システム開発事例共有システム	レビューメンバー	(情公共)X(除旗), (宮崎), (渡辺), (シ研)(平井), 工藤
3. 作番・整番	S96A015-2	4. 工程記号、イベント番号	
5. 実施日・場所	98/04/02,14:00～,新砂13F南執務室		
6. タイトル	CBExplorer:トピックス表示機能と性能対策案		
7. 概要	CBExplorer開発は事例共有システムプロジェクト サーバとして洗出しとプロト タイプ作成はほぼ完了している。今回のテーマは現状のCBExplorerに残されている		
8. 対象ドキュメント			
9. 参考資料		担当システム	(情公共)X(宮崎) (シ研)203u工藤
10. 件数	検討 件 対策 件 持越 件		

12. 今後の検討項目	未定(1) マルチサーバ 未定(2) データ格納方 未定(3) 事例分析の具
13. 主な成果	

テーマ	結論	理由
1. CBExplorer現行機能についての検討 1.1 トピックス表示機能について	メイン画面において議事録のタイトル・日付(必要最低限の情報)を表示項目としトピックスを表示させることにした。 そこで、トピックスの表示に関しての仕様を以下のように設定した。 ・トピックスへの表示は議事録を登録する際に、表示する/しないの指定、表示期限の指定を行うものとする。 ・トピックスの情報の表示数上限は環境iniファイルでサーバ毎に管理をおこなうものとする。 ・表示件数は環境iniファイルのデフォルト値を更新することで変更可能となる。 ・作成には、省力化のため、(シ研)203uのWEB掲示板での技術を流用させて頂く。	現状では最近登録された議事録への参照が多いと考えられるが、最近のDR票という切り口で参照するには日付による検索を行わなければならない。検索処理が多用されるとパフォーマンスが悪化すると考えられるため。 また、(シ研)203uのWEB掲示板で既に実績ある技術であり、システムへの作り込みが容易であるから。
1.2 管理者・特権者のためのフリーアクセスについて	ユーザ認証を通すとアクセスレベルのチェックをパスする機能を	現状ではアクセスレベル設定において社外秘となるプロジェクト

図 2-3 レビュー議事録の例

2.2.4 レビューの十分性(充実度と質)の評価(See)

レビューの十分性の評価は，レビュー項目表とレビュー議事録に基づいて，充実度と質の 2 つの観点で行っている。

レビューの充実度は，レビュー項目に対するレビュー実施回数により，定量的に評価する。レビュー項目毎のレビュー実施回数は，レビュー議事録に記述されている内容を手掛かりにカウントする。レビュー実施回数が少ない場合は議論が不十分である可能性があり，逆にレビュー回数が多い場合は議論が収束せず長引いている可能性がある。また，レビューが実施されていない項目については，検討漏れの可能性がある。

レビューの質は，議事録を参照することで，設計の考え方を定性的に評価する。議事録には，設計に関する結論とその理由が記述される。また，採用された案だけでなく，棄却された案も記述される。レビューの対象となった資料や議論の材料になった参考文献への参照が記録される。出席者も記録される。これらの情報に基づいて，どのような案を比較検討し，どのような根拠で設計したか，どのような資料や参考文献をもとに設計したのか，責任者や有識者を交えて議論したか，といった観点で評価する。結論に対する根拠が曖昧であったり，責任者や有識者が出席していなかったり，結論が「次回までに検討」であったり，理由が明記されていなかったりする場合は，レビューが十分でないと判断される。

2.3 設計レビュー管理の課題と解決のための支援方式

2.3.1 設計レビューの課題

レビュー計画の立案から議事録の作成，設計の十分性の評価を円滑に進められるように支援するためには，以下の課題を解決する必要がある。

課題 1 議事録作成者への負荷が大きい

レビューの十分性を評価するために必要な記述項目や議事録作成にあたっての注意事項が多くなりがちで，議事録の作成に手間がかかっている。議事録は，設計レビューのたびに作成することになるため議事録作成者への負荷が大きい。

課題 2 レビューの充実度を定量的に評価する手間が大きい

レビューの充実度の評価を行うには，レビュー項目表に定義されたレビュー項目に対して，レビュー議事録にそのレビュー項目に関する記述が存在するかについて，内容を目視で確認しながら，レビュー回数をカウントする作業が必要となっている。そのため，プロジェクト管理者の手間が大きい。

課題 3 レビューの質を定性的に評価する手間が大きい

レビューの質の評価を行うには，レビュー項目表に定義されたレビュー項目についての議事内容をすべて参照しなければならない。設計レビューには，1 つの議題は数回のレビューを要し,1 回のレビューでは複数の議題が討議されるという特徴がある。そのため，複数の議事録を参照し，該当する議事段落部分を 1 つ 1 つ目視で追っていかなければならず，プロジェクト管理者の手間が大きい。

このレビューの充実度と質の評価については，評価量が多く，また，実施の手間が大きいため，設計現場のほとんどのプロジェクトが実施できていなかった。実施できていたプロジェクトの中でも，レビューが開催される度に実施するのではなく，数ヶ月に一回，まとめて実施しているケースがほとんどであった。

2.3.2 課題を解決するための支援方式

本項では，前項で挙げた課題を解決するための，議事録を利用した支援方式について説明する。議事録を利用した設計レビュー管理支援方式の全体像を図 2-4 に示す。図 2-4 にしたがって，支援 1 から支援 3 までの各支援方式の概要を説明する。

支援 1 議事録作成支援

課題 1 に対しては，従来のテキスト形式は継続した上で，議事録作成者の負担にならない程度の記述ルールを定める。記述ルールは，レビューの充実度の評価，レビューの質の評価を行うために最低限必要なものとする。また，記述ルールに準拠した議事録の雛形を生成する機能により，議事録作成の手間を軽減する。

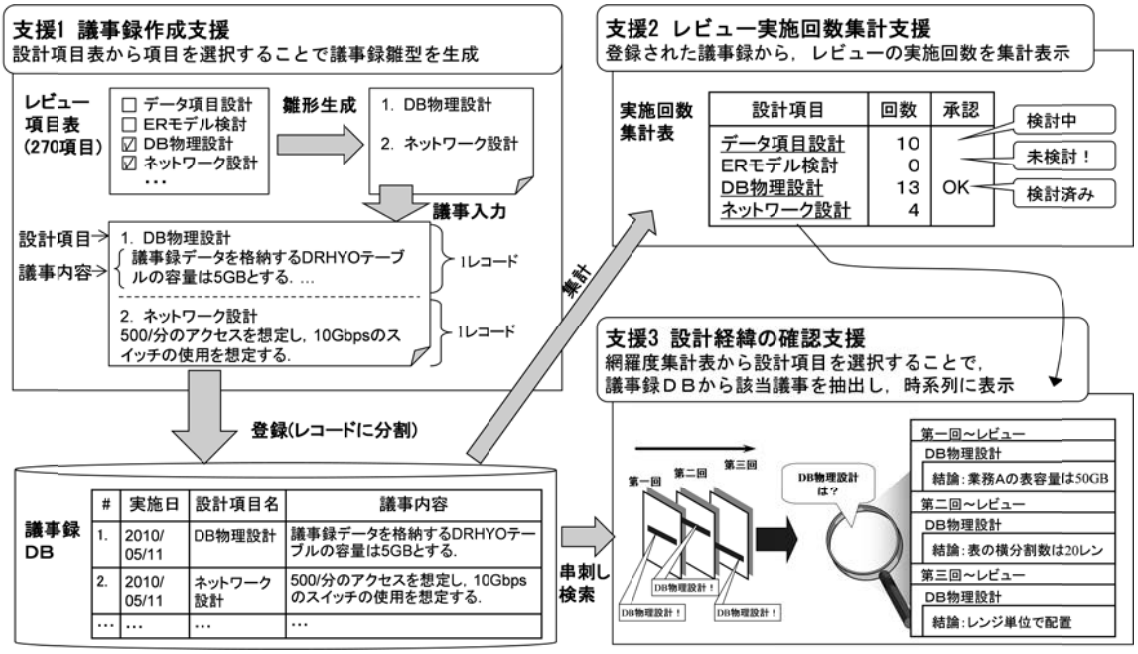


図 2-4 設計レビュー管理支援方式の概要

支援 2 レビュー実施回数集計支援(レビューの充実度の評価支援)

課題 2 に対しては，登録された議事録の情報にしたがってレビュー項目に対する登録議事録数をカウントする機能により，プロジェクト管理者がレビュー項目表の各項目に対する設計レビューの実施回数を集計する手間を排除する。

支援 3 設計経緯の確認支援(レビューの質の評価支援)

課題 3 に対しては，ある設計項目に関する議事内容を容易に評価できるように，確認したい設計項目に対する議事内容部分のみを議事録データベースから抽出し，時系列に連結して一画面に表示する。一画面に表示することで，マウス操作などによってリンクを辿って複数の議事録を試行錯誤的に参照する手間を排除する。

次の項以降，支援 1 から支援 3 までの各支援方式について，詳細に説明する。

2.3.3 支援 1: 議事録フォーマットの策定と議事録作成支援

定めた議事録フォーマットの記述項目と記述ルールを図 2-5 に示す。議事録に記述すべき項目は，大きく二つある。1 つはレビューの開催情報で，もう 1 つはレビューにおける議事内容である。レビューの開催情報には，タイトル(会議名)，実施日時，出席者，報告者などを記述する。議事内容には，議題名を見出しとして，1 つの議題に対する議事内容が 1 つの段落になるように記述する。議事内容部分には，1 つ以上の議題を記述する。

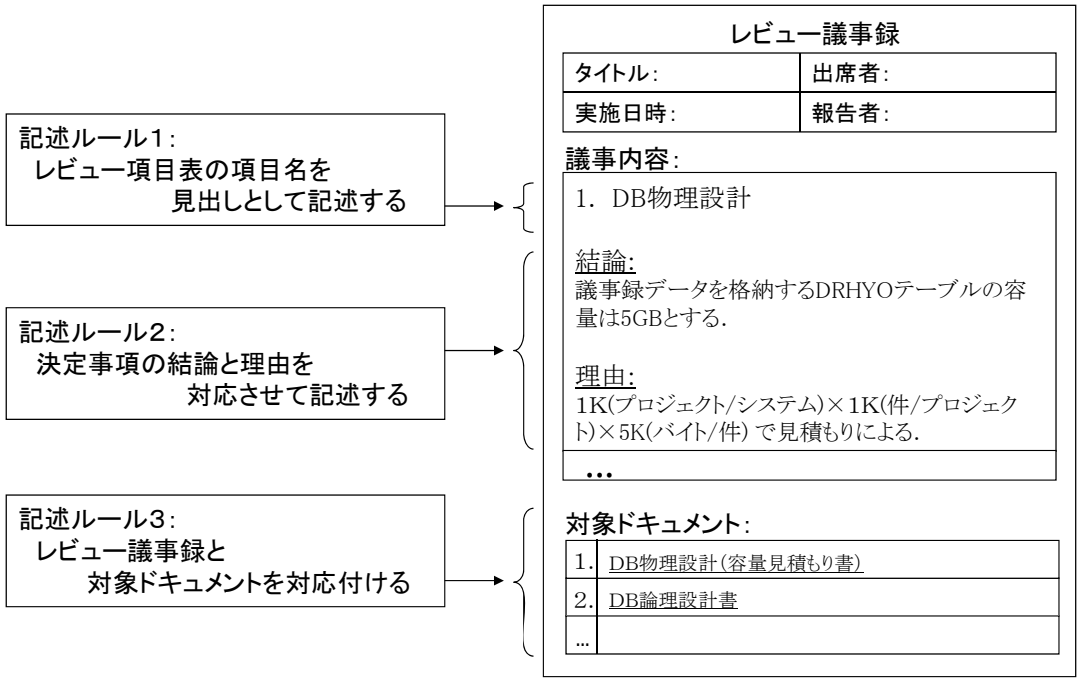


図 2-5 議事録の記述ルール

記述ルール 1 レビュー項目表の項目名を見出しとする

情報システムに関する設計レビューでは，図 2-6 に示す通り，1 つの議題について，数回のレビューを要する。そのため，議事録においても，1 つの議題に対する記述部分が複数の議事録にまたがって記述される。レビュー項目表の項目名を見出しとする(見出しを統一する)というルールを定めることで, 複数の議事録にまたがって存在する同一議題に対する記述部分を追跡しやすくする。

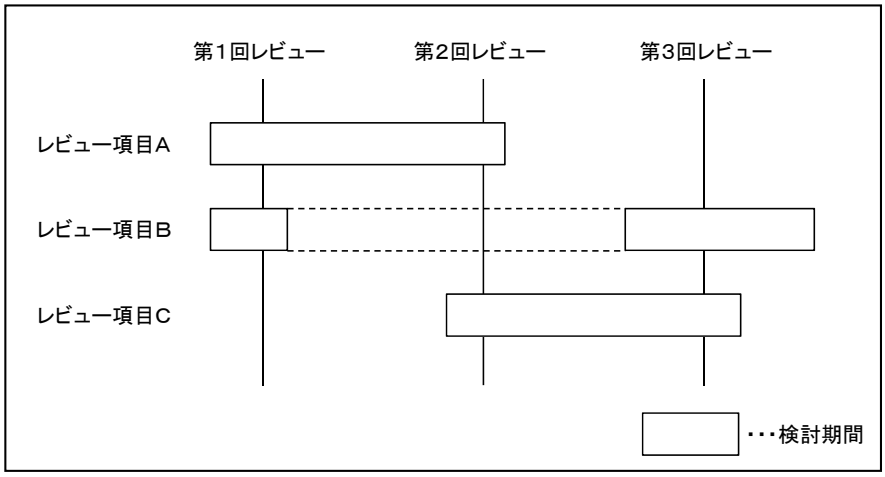


図 2-6 設計レビューの特徴

記述ルール 2：結論と理由を対応させて記述する

設計書や仕様書は，設計した機能がどのように動作するかという点(How 情報)に重点をおいて作成されることが多い[89]。しかし，How 情報だけでは設計意図を正しく伝達することは難しい。設計ノウハウ活用の観点からも，なぜそのように設計したのかという設計根拠(Why 情報)が重要となる。そのため，議事録には，レビューで結論づけた設計内容(How 情報)と，その設計根拠(Why 情報)を対応させて記述するようにする。

記述ルール 3：レビュー対象文書を添付する

議事録記載の情報だけでは，設計経緯の把握は十分に行えない。何をレビューしたときの記録なのか即座に確認できるようにするために，議事録にレビュー対象文書を添付する。

さらに，これらの記述ルールに準拠した議事録の作成を支援するために，レビュー項目表の画面からレビュー項目を選択することで，「見出し」「結論：」「理由：」を含めた議事録の雛型を生成するようにする。また，ルール 3 に対する支援として，議事録入力フォーム上で対象文書名(ファイル名)を指定することで, 議事録と関連付けた形でサーバにファイルを登録できるようにする。

2.3.4 支援 2: レビュー実施回数集計支援

レビューの充実度は，レビュー項目に対するレビュー実施回数で評価する。図 2-7 に示すように，各レビュー項目に対する検討の充実度の評価を容易にするために，レビュー項目表の各項目に対して，議事録の登録件数，有識者参加回数等を集計して表示する。

また，議事録が 1 件でも登録されている場合は，レビュー項目名を時系列串刺し表示へのリンクとして表示する。これをクリックすることで，この項目に対する対応議事内容部分を抽出して，時系列串刺し形式で表示する。

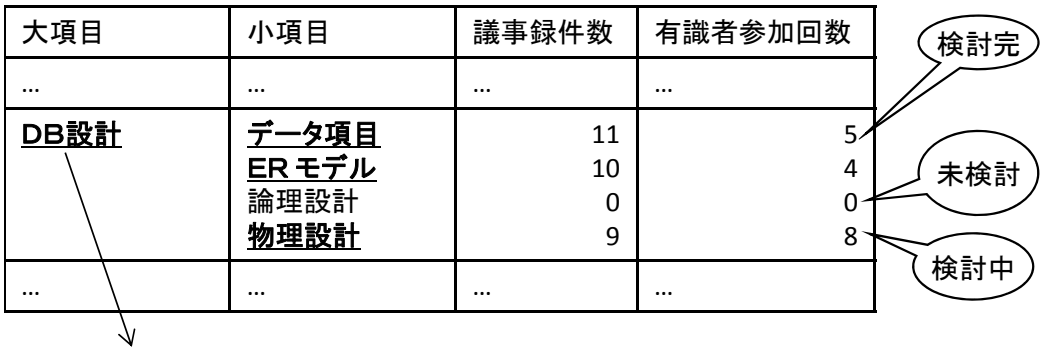


図 2-7 レビュー実施回数集計表の例

2.3.5 支援 3: 設計経緯の確認支援

レビューの質は，レビュー項目表に定義されたレビュー項目に対する議事内容を参照して評価する。複数の議事録にまたがって記述された議事内容を容易に，且つ，漏れなく目視確認できるように，図 2-8 に示すように，多数の議事録から，該当箇所を，議事録の単位ではなく，段落の単位で抽出し，レビュー実施日時の順にソートして一覧表示する。一覧表示には，各議事内容の出展(どの議事録から抽出されているか)，そのレビューの実施日時，出席者などの情報を含めることで，レビュー時の状況を推察しやすいようにする。

レビューの充実度と質の評価によってレビューの十分性が確認できたときは，当該レビュー項目に対して，完了宣言を行う。完了宣言はプロジェクト管理者が行う。

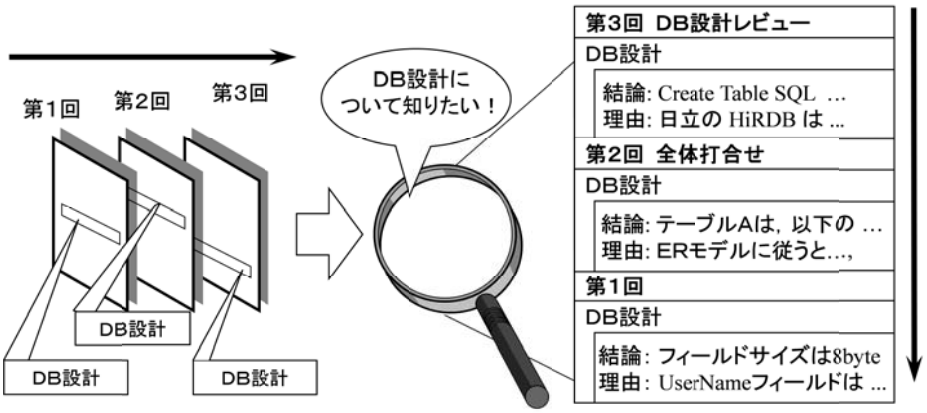


図 2-8 時系列串刺し表示の例

2.4 設計レビュー管理システムの設計

本節では, 2.3 節で述べた各支援方式を実際の設計開発現場に適用するために開発した設計レビュー管理システムの設計について述べる。

2.4.1 システム構成

設計レビュー管理システムは，分散開発拠点にある 100 以上のプロジェクトが利用できるように Web アプリケーションとして開発する。システムの構成を図 2-9 に示す。レビュー項目表を登録・編集・削除するためのレビュー項目表管理部，レビュー項目表から議事録の雛形を生成する議事録雛形生成部，作成した議事録を議事録データベースに登録する議事録登録部，登録した議事録を用途に合った形式に変換して表示する議事録表示部，登録された議事録からレビュー実施回数集計表を作成するレビュー実施回数集計部，指定されたレビュー項目に関連する議事内容を時系列に連結して表示する串刺し検索部，で構成する。また，プロジェクトに属するメンバーだけがそのプロジェクトの議事録を登録できるように，また，議事録登録者と議事録参照者，プロジェクト管理者を区別できるように，アクセス制御部を持たせる。

本システムは, Microsoft 社の ASP (Active Server Pages)と Visual Basic 6.0 で開発した。Web サーバに Microsoft IIS (Internet Information Services), データベースサーバは,（株）日立製作所の HiRDB で，OS には，Microsoft Windows 2000 Server を用いた。サーバのスペックは，Intel 社の PentiumⅢプロセッサ 747 MHz，512 MB RAM である。

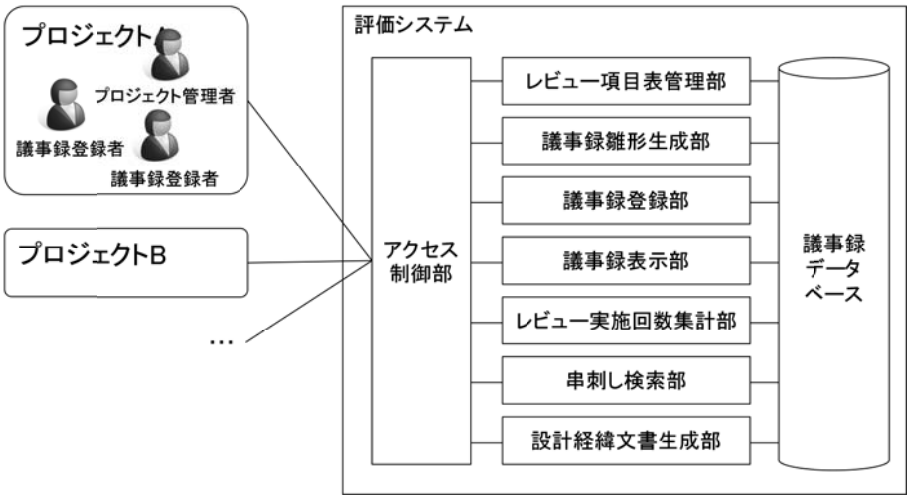


図 2-9 設計レビュー管理システムの構成

2.4.2 議事録データベースの構造

議事録情報を正確に検索・加工できるように，議事録データベースの構造を図 2-10 のようにする。プロジェクトに関する情報を格納するプロジェクトテーブル，設計レビューの開催情報を格納する議事録テーブル，議事内容を格納する議事内容テーブル，出席者を格納する出席者テーブル，対象ドキュメントに関する情報を格納する対象ドキュメントテーブルの 5 つのテーブルで構成する。1 つのプロジェクトには，複数の議事録を関連付けて登録し，1 つの議事録は複数の議事内容で構成されるようにする。また，1 つの議事録は，複数の出席者や対象ドキュメントを登録できるようにする。

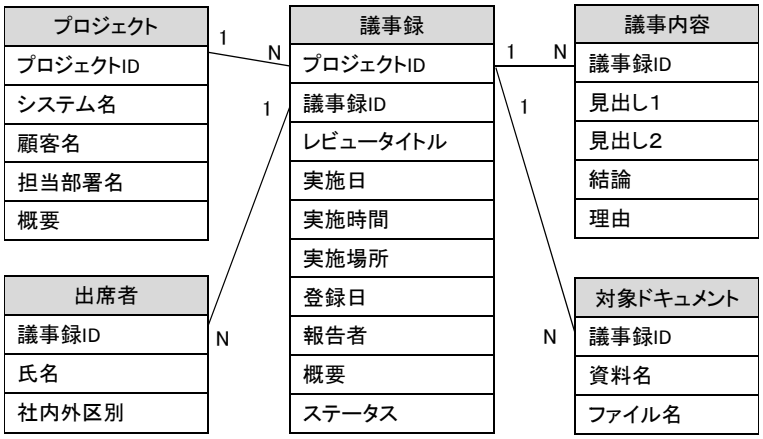


図 2-10 プロジェクト情報及び議事録情報のデータ構造

2.4.3 レビュー項目表管理部

プロジェクト管理者が，レビュー項目表に相当する情報をシステムに定義できるように，レビュー項目表の定義機能を提供する。図 2-11 にレビュー項目定義機能のスクリーンショットを示す。レビュー項目をツリー形式で構成する作業をグラフィカルなユーザインタフェースで行える。各レビュー項目の作成にあたっては，各項目の属性情報として，レビュー観点や承認者，承認日，期待するレビュー回数(図中の「登録 DR 件数」) (DR: Design Review / Decision Review)，及び，必須項目か否かのフラグを設定する。

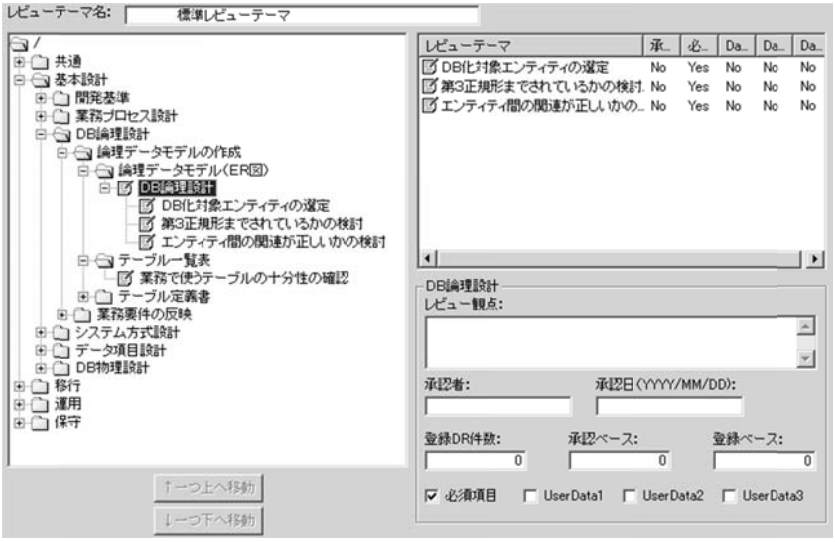


図 2-11 レビュー項目定義機能のスクリーンショット

2.4.4 議事録雛形作成部

ルールにしたがった議事録の作成支援として，レビュー実施回数集計表(レビュー項目表ベース)からレビュー項目を選択することで，「見出し」「結論：」「理由：」を含めた議事録の雛型を生成する。雛形は,「見出し=レビュー項目名」として生成する。スクリーンショットを図 2-12 に示す。議事録作成者は，レビューが終了して議事録を作成する際に，レビューした設計項目を画面上で選択して議事録の雛形を作成する。または，レビュー実施回数集計表と時系列串刺し表示によりレビューの充実度と質を確認して，次回レビューのアジェンダを決める要領で議事録の雛形を作成する。

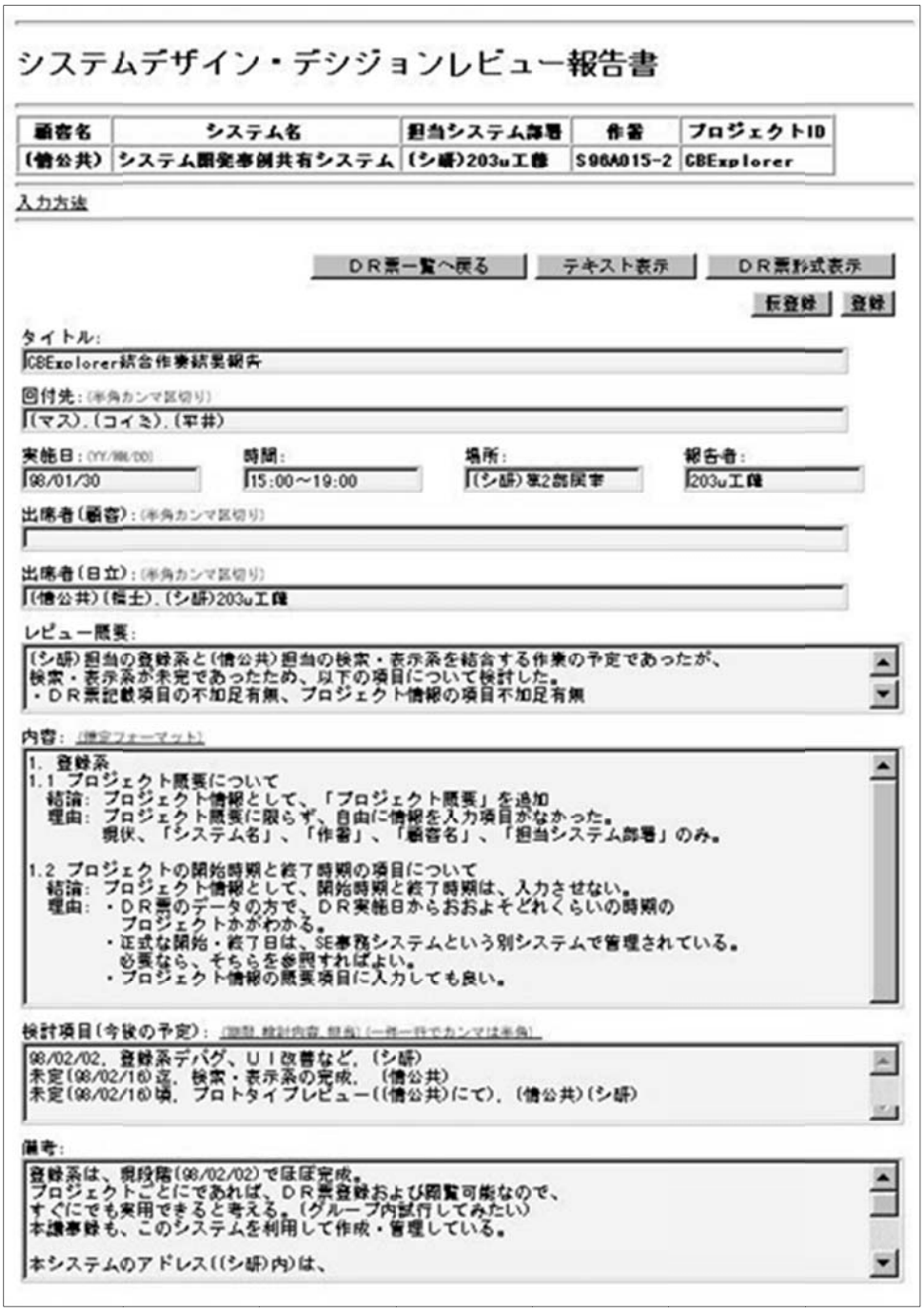
図 2-12 議事録の雛形作成機能のスクリーンショット

2.4.5 議事録登録部

議事録は，図 2-13 に示すような，HTML(Hyper Text Markup Language)のフォームに入力して登録する。雛形を生成して議事内容を記述するか，何も記入されていないところから新たに記述してもよい。

議事録を登録する際は，議事録データベースの構造に合わせて各項目の内容を登録する。レビュータイトルや実施日，実施時間，実施場所などのレビュー開催情報は，入力時点で入力フォームの項目が分かれているため，対応するカラムにデータとして格納する。議事内容は，1 つのテキストとして入力されるため，テキストの分割処理を行う必要がある。高度な自然言語処理を行わない代わりに，図 2-14 に示すような議事内容部分の書式を定めることとした。テキストの分割処理を行う際には，見出し 1 または見出し 2 の書式のマッチングにより，議事内容を分割して議事録データベース

に格納する。そのとき，レビュー網羅度の集計，串刺し検索の各処理が文字列のマッチングのみで行えるように，見出し 1 や見出し 2 に含まれる見出し番号(数字とピリオド)と見出し文字列は，分離して格納する。



システムデザイン・デシジョンレビュー報告書

顧客名	システム名	担当システム部署	作者	プロジェクトID
(情公共)	システム開発事例共有システム	(シ研)203u工藤	S98A015-2	CBExplorer

入力方法

DR票一覧へ戻る　テキスト表示　DR票形式表示

仮登録　登録

タイトル:
CBExplorer結合作業結果報告

回付先:(半角カンマ区切り)
(マス),(コイミ),(平井)

実施日:(YY/MM/DD)
98/01/30

時間:
15:00～19:00

場所:
(シ研)第2部屋事

報告者:
203u工藤

出席者(顧客):(半角カンマ区切り)

出席者(日立):(半角カンマ区切り)
(情公共)(桜士),(シ研)203u工藤

レビュー概要:
(シ研)担当の登録系と(情公共)担当の検索・表示系を結合する作業の予定であったが、
検索・表示系が未完であったため、以下の項目について検討した。
・DR票記載項目の不加足有無、プロジェクト情報の項目不加足有無

内容:(推定フォーマット)

1. 登録系
1.1 プロジェクト概要について
結論: プロジェクト情報として、「プロジェクト概要」を追加
理由: プロジェクト概要に限らず、自由に情報を入力項目がなかった。
現状、「システム名」、「作者」、「顧客名」、「担当システム部署」のみ。

1.2 プロジェクトの開始時期と終了時期の項目について
結論: プロジェクト情報として、開始時期と終了時期は、入力させない。
理由: ・DR票のデータの方で、DR実施日からおおよそどれくらいの時期の
プロジェクトかがわかる。
・正式な開始・終了日は、SE事務システムという別システムで管理されている。
必要なら、そちらを参照すればよい。
・プロジェクト情報の概要項目に入力しても良い。

検討項目(今後の予定):(項目,検討内容,期限)(一件一行でカンマは半角)

98/02/02, 登録系デバグ、UI改善など, (シ研)
未定(98/02/16)迄, 検索・表示系の完成, (情公共)
未定(98/02/16)頃, プロトタイプレビュー((情公共)にて), (情公共)(シ研)

備考:
登録系は、現段階(98/02/02)でほぼ完成。
プロジェクトごとにであれば、DR票登録および閲覧可能なので、
すぐにでも実用できると考える。(グループ内試行してみたい)
本議事録も、このシステムを利用して作成・管理している。

本システムのアドレス((シ研)内)は、

図 2-13 議事録の入力フォーム

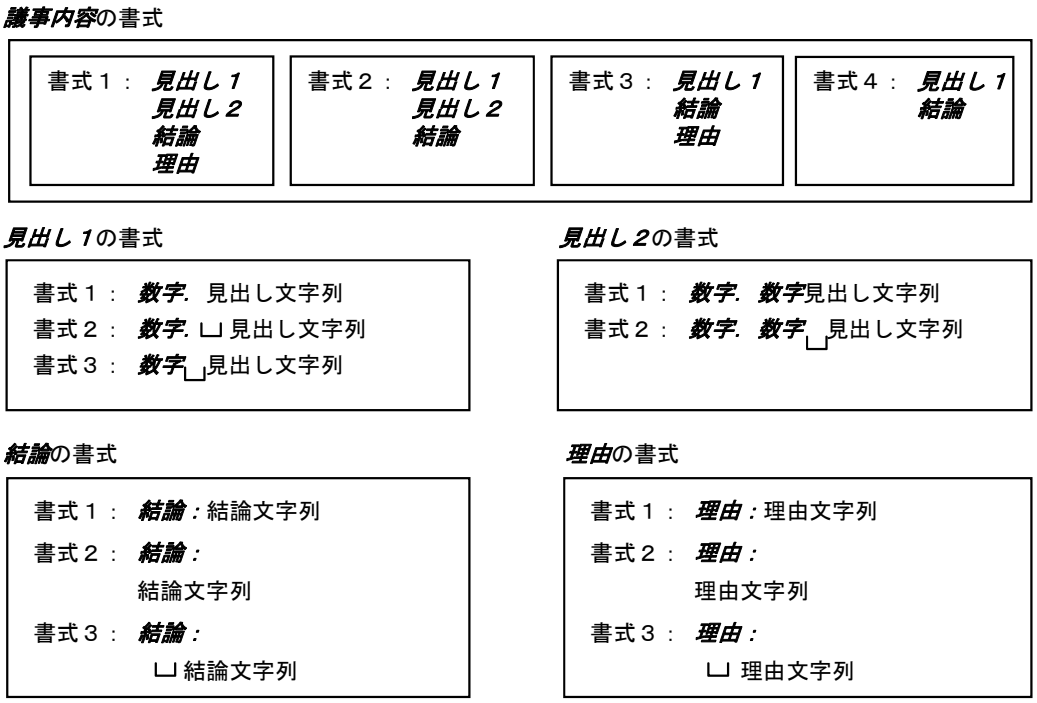


図 2-14 議事内容部分の書式

2.4.6 レビュー実施回数集計部

議事録データベースに対し，レビュー項目表の各項目を検索キーにして検索処理を行い，各項目に対する議事録件数をカウントして一覧表を表示する。レビュー実施回数集計表のスクリーンショットを図 2-15 に示す。この画面は，議事録の登録状況に基づく評価とプロジェクトマネージャの承認状況に基づく評価により，レビューの実施状況を確認するためのものである。画面中央の DR 票登録ベースのカラムには，レビュー率と DR 件数，責任者参加回数，有識者参加回数を表示する。DR 票とは，デザインレビュー報告書，または，デシジョンレビュー報告書の呼称であり，レビュー議事録のことである。責任者参加回数と有識者参加回数は，それぞれ，レビュー議事録の出席者欄に記載があれば一回とカウントする。一方，画面右側の PM 承認ベースのカラムには，レビュー率と承認記録を表示する。PM とは，プロジェクトマネージャのことである。その設計項目のレビューが十分に行えたと判断した場合，チェックボックスにチェックを入れ，登録する。ログイン情報から承認者名を参照し，システムクロックから承認処理時点の日付と時間を参照して登録する。

図 2-15 レビュー実施回数集計表のスクリーンショット

2.4.7 串刺し検索部

議事録データベースに対して，検索条件と表示条件を指定することで，該当する設計項目に対する議事内容，すなわち「結論」と「理由」を，指定したプロジェクトのすべての議事録から抽出し，時系列で表示する。例えば，図 2-16 のような議事録データが議事録データベースに格納されているとき，「クライアントのスペック」というキーワードで検索した場合，見出し 1 または見出し 2 に「クライアントのスペック」という語が含まれるレコードを抽出し，図 2-17 のように時系列の一覧表を表示する。

串刺し検索の検索条件が「見出し=クライアントのスペック」であった場合，議事録データベースに対して，以下の SQL (Structured Query Language)文が実行される。この結果を加工することで，の時系列串刺し表示を作成する。

議事録テーブル

議事録ID	プロジェクトID	レビュータイトル	実施日	…
DR-00131	CBExplorer	第3回プロトレビュー	09/03/20	…
DR-00132	CBExplorer	第4回プロトレビュー	09/04/01	…
DR-00133	CBExplorer	デモ構成打ち合わせ	09/04/05	…
…	…	…	…	…

出席者テーブル

議事録ID	氏名	…
DR-00131	工藤裕	…
DR-00131	平井千秋	…
DR-00131	降旗由香理	…
…	…	…

議事内容テーブル

議事録ID	Seq.	見出し1	見出し2	結論	理由
DR-00131	1	機器構成	クライアントのスペック	CPUは…、メモリは…	低コストを目指…
DR-00131	2	機器構成	サーバのスペック	CPUは…、メモリは…	レスポンス向上…
DR-00131	3	機器構成	ネットワーク機器	帯域は…	
…	…	…	…	…	…
DR-00134	1	機器構成	ディスプレー	解像度は…	操作画面が…
DR-00134	2	機器構成	クライアントの台数	最大10,000台とする。	
DR-00134	3	機器構成	クライアントのスペック	CPUは…、メモリは…	クライアント側の処理が
DR-00134	4	開発体制	設計作業	3つのサブシステムに分…	並行に設計できると…
…	…	…	…	…	…

図 2-16 議事内容データのデータベースへの格納例

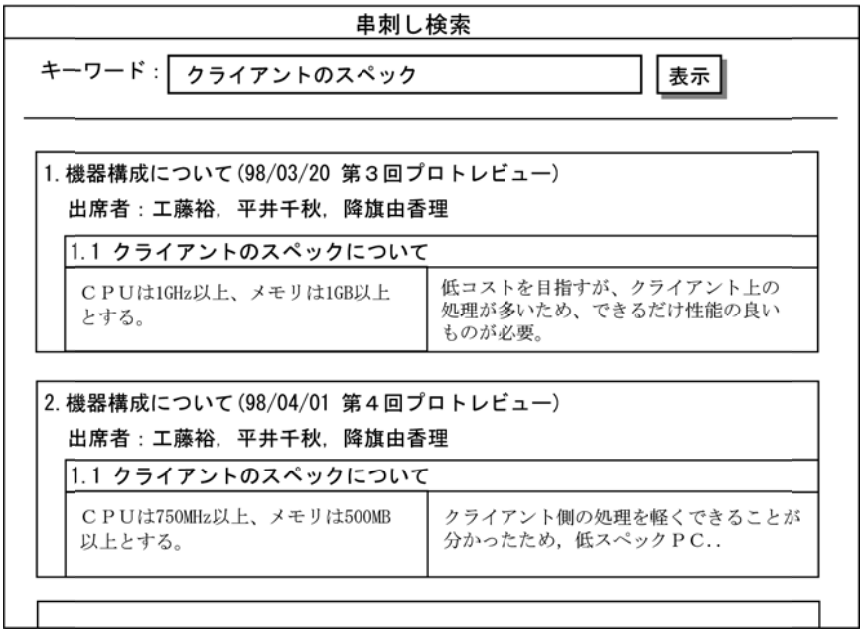


図 2-17 時系列串刺し検索の検索結果の表示例

SELECT 議事内容テーブル.見出し 1，議事内容テーブル.見出し 2,
議事内容テーブル.結論，議事内容テーブル.理由,
議事録テーブル.プロジェクト ID，議事録テーブル.レビュータイトル,
議事録テーブル.実施日，出席者テーブル.氏名
WHERE (議事内容テーブル.見出し 1 LIKE '*クライアントのスペック*' or
議事内容テーブル.見出し 2 LIKE '*クライアントのスペック*') and
議事内容テーブル.議事録 ID = 議事録テーブル.議事録 ID and
議事内容テーブル.議事録 ID = 出席者テーブル.議事録 ID

スクリーンショットを図 2-18 に示す。図 2-18 は，検索条件として，議事内容に「データベース設計」を指定し，表示条件として，タイトル，出席者，内容を指定した際の串刺し検索の結果の表示例を示したものである。議事録のタイトルの文字列は，リンクとして生成し，議事録単位の表示をポップアップで行える。設計の経緯を確認しながら，その時の設計レビューで同時に議論した他の設計項目に対する議事内容の確認が即座に行えるようになっている。表示条件として指定された項目は，出所の議事録の情報をマッシュアップする形で表示される。例えば，検索条件にヒットした議事内容の議事録 ID をもとに，議事録テーブルからレビューのタイトルを取得し，出席者テーブルから，出席者名を取得し，これらをつなぎ合わせることで時系列串刺し表示を作る。

図 2-18 時系列串刺し表示のスクリーンショット

2.5 提案方式の有効性に関する評価と考察

設計レビュープロセスの効率向上については，常に多方面からのアプローチが試されるため，全体として効率が向上したとしても，本研究での提案方式を実装した設計レビュー管理システムが直接的に貢献したことを証明するのは難しい。そこで，本支援方式の評価については，はじめに議事録の登録状況を分析し，その上で定性的な評価を，本研究で開発した設計レビュー管理システムの利用者の意見に基づいて行う。次に，定量的な評価として，(1) 設計レビュー議事録の作成工数，(2) レビューの充実度を評価する時間，(3) 1 つの設計項目についてレビューの質を評価する時間，の 3 点について，従来方式と比較する。

2.5.1 議事録の登録状況に基づく分析

本システムへの議事録の登録状況から，議事録の登録頻度，議事録記述ルールの遵守状況を分析した。表 2-1 に議事録の登録状況を示す。評価にあたっては，開発対象や期間の異なる 5 つのプロジェクトを対象とした。

最も利用頻度の高いプロジェクトは文書管理(中規模)で，1 ヶ月あたりの議事録登録件数が 20 件以上であった。このことから，このプロジェクトでは，毎日 1 件以上，日によっては 2 件以上の議事録登録があり，議事録作成が積極的に行われたことが分かった。他のプロジェクトについても，単純計算で 2 日に 1 件の登録があることから，本システムの利用による議事録の作成，登録が徹底されたことが分かった。

また，1 回のレビューでは，どのプロジェクトでも 10 件前後の議題(レビュー項目)について議論されていることが分かり，議事録単位の議事録管理ではなく，項目単位の議事内容管理が必要であることが確認できた。

また，プロジェクトの進捗率に対する議事録の登録件数の変化を図 2-19 に示す。文書管理(中規模)において，進捗率 40%の時に急激に議事録登録数が減少しているのは，プロジェクトがコーディング工程に入ったからである。進捗率 80%にはコーディング工程が終わって，テストを実施し，不良に対する対策会議が増え始めている。文書管理(中規模)は公共機関向けの情報システムであり，完全なウォーターフォールモデルで開発が進められているため，議事録登録件数がこのようにはっきりした形で推移したものと考えられる。一方，グループウェアはパッケージソフトウェアであり，開発環

境は主には社内向けツールであるため，明確なウォーターフォールモデルで開発されてはいない。そのため，開始から完了までおおよそ変化なく議事録登録件数が推移したと考えられる。

表 2-1 議事録の登録状況

対象プロジェクト	利用期間[月]	議事録件数	議事録件数/月	議題数	議題数/件	理由記述率[%]	文書添付率[%]
文書管理(中規模)	18	385	21.4	3,555	9.2	37.7	52.5
文書管理(大規模)	20	319	16.0	2,842	8.9	85.0	79.3
グループウェア	39	418	10.7	4,181	10.0	91.1	54.5
公共系アプリケーション	39	352	17.6	3,913	11.1	30.0	9.9
開発環境	40	374	9.4	3,405	9.1	64.2	29.9

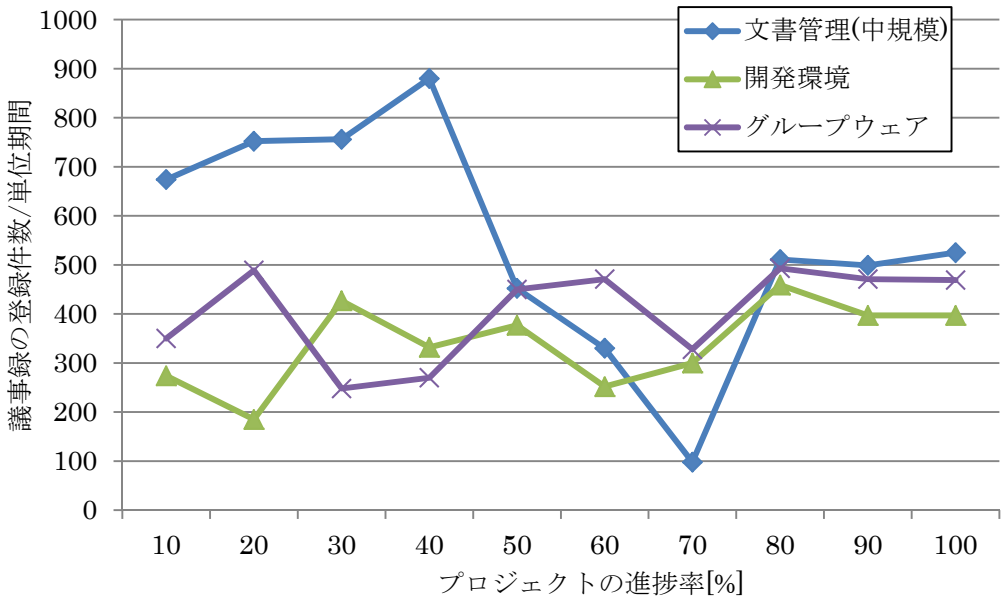


図 2-19 プロジェクトの進捗率に対する議事録の登録件数の変化

2.5.2 設計レビュー管理の効率化に関する評価と考察

本システムの各支援機能により，設計レビュー管理の効率が向上したかどうかヒアリングを行った。利用者からは，「日々検討を進めるという点ではこれまでと変わらないが，検討の進み具合が視覚的に確認できるため，これが目標となり，意欲的に仕事を進めることができるようになった」，「レビュー項目表がツリー形式なので，広く浅く議論する場合でも，細部を深く議論する場合でも，適切な項目を選びやすい」という肯定的な意見を得た。また，プロジェクト管理者からも，「レビュー実績が細かく把握でき，検討漏れの防止に役立っている」という意見を得た。これら利用者の反応から，レビューの十分性を評価しやすくし，レビューを効率化するという目的を達成できたと考える。

以下，Plan-Do-See の各フェーズに対して，個別に考察する。

(1) レビュー計画立案(Plan)

レビュー計画立案に関しては，現状，各プロジェクトを支援する生産技術部門が標準的なレビュー項目の策定を行い，各プロジェクトは，これをカスタマイズして利用するという運用形態をとっている。しかし，プロジェクトの特性(開発形態，開発規模など)が多様化する中では，このカスタマイズが困難な作業となっており，この作業に対する支援が不十分であることが分かった。管理ノウハウという点では，このレビュー項目表自体が重要な知識であり，今後は，各プロジェクトでの試行錯誤の結果を知識化し，プロジェクト特性ごとのテンプレート化を図る必要がある。

また，プロジェクト管理者からは，「レビュー項目表を随時変更できるのがうれしい」という意見を得た。このプロジェクトでは，プロジェクト開始時に 270 項目だったレビュー項目表が，プロジェクト終了時には約 400 項目に増加していた。内容としては，「アクセス制御リストの統合方式」，「認証サーバの連携方式」などプロジェクト固有の設計項目や，「プロナビ連携方式」，「HiRDB 移行」など，製品名を含んだ項目である。これらの設計項目は，設計を進めていく過程で，設計の必要性が生じ，その都度，レビュー項目表に追加されていったとのことであった。このように，状況に応じて柔軟に設計活動を最適化していける Plan-Do-See モデルの特徴が現場で理解され実行されていることが確認できた。

(2) レビュー実施，及び，議事録作成と配布(Do)

レビュー実施に関しては，「あらかじめ作成した議事録の雛型に対して，レビューの場で結論と理由を書き込んでいけるので，議論効率が上がった」という意見を得た。これは，選択したレビュー項目について，「見出し」「結論：」「理由：」の形式で議事録の雛形を作成する議事録作成支援機能をアジェンダ作成機能として利用した例である。これまでのレビューでは単に対象文書に対して議論を進行させることが多かったが，レビュー観点を明確にしつつ対象文書をレビューするという，レビュー実施方法に対する考え方の変化が確認できた。

結論に対する理由を明記するというルールに関しては，「このルールはこのシステムの導入前からあったが，あまり守っていなかった。他のプロジェクトの時系列串刺し表示を見て，理由が書かれていないことを残念に思い，自分は理由を書くようになった。そのおかげで，つねに理由を考慮しながら結論付ける習慣がつき，理論的な設計ができるようになった」という意見を得た。また，レビュー項目を見出しとするというルールに関しては，「1 つの段落で 1 つの検討内容を記述しなければならないため物事を簡潔に文章化する訓練になっている」という意見を得た。このことから，設計者の思考プロセスの質的向上にも効果があったと考えられる。

(3) レビューの十分性(充実度と質)の評価(See)

品質保証部門の担当者からは，「重点設計項目について，レビューを実施したかどうかを確認できるのがうれしい」という意見を得た。1 つのプロジェクトに対して複数のレビュー項目表を用意し，これらを切り替えてレビューの実施管理を行うことによる利点が理解され実行されていることが確認できた。また，「品質保証部門としては，1 人あたり複数のプロジェクトを担当することが多いため，複数プロジェクトの状況を同時に把握できるようにしてほしい」という要望が出された。より便利に使えるようにするための提案が，利用者側からなされたことから，本機能の有用性が確認できたと考える。

時系列串刺し表示に関しては，「Know-Who を知るのに便利」という意見が多かった。検索結果として得た出席者情報から，さらにその出席者が，どのようなプロジェクトに参加してきたかを調べることで，その出席者が持つ技術的背景を知ることができる。この時系列串刺し表示は当初，議論過程を表示する目的で設計したが，実際に

は，Know-Who 検索や添付文書検索など，議論過程表示以外の目的で使用されることが多かった。また，「レビュー会議の冒頭で，プロジェクタで当該レビュー項目の時系列串刺し表示を投影することによって，前回までの議論過程を出席者全員で確認できて無駄な議論が排除できるようになった」という意見を得た。この意見から，現場での工夫により，これまで想定していなかった使い方がなされていることが分かり，本方式の IT による会議支援方式への拡張可能性を確認できた。

時系列串刺し表示の利用状況を直接的に示すデータはないが，議事録の「理由」の記入率の変化により，間接的に，どの程度使われたかを推測する。図 2-20 は，プロジェクトの進捗率に対する理由記入率の推移である。プロジェクトによって差はあるが，おおよそプロジェクトが進行していくにつれ，理由を記入する率が増加している。理由の記入率が増加しているのは，時系列串刺し表示で「理由」が記入されていない場合に，正しい評価ができないためである。つまり，次回より理由の記入を心がけようとフィードバックがかかり理由の記入率が増加したものと考えられる。理由の記入率が増加したことから，それ相応の時系列串刺し表示が利用されたものと推測する。

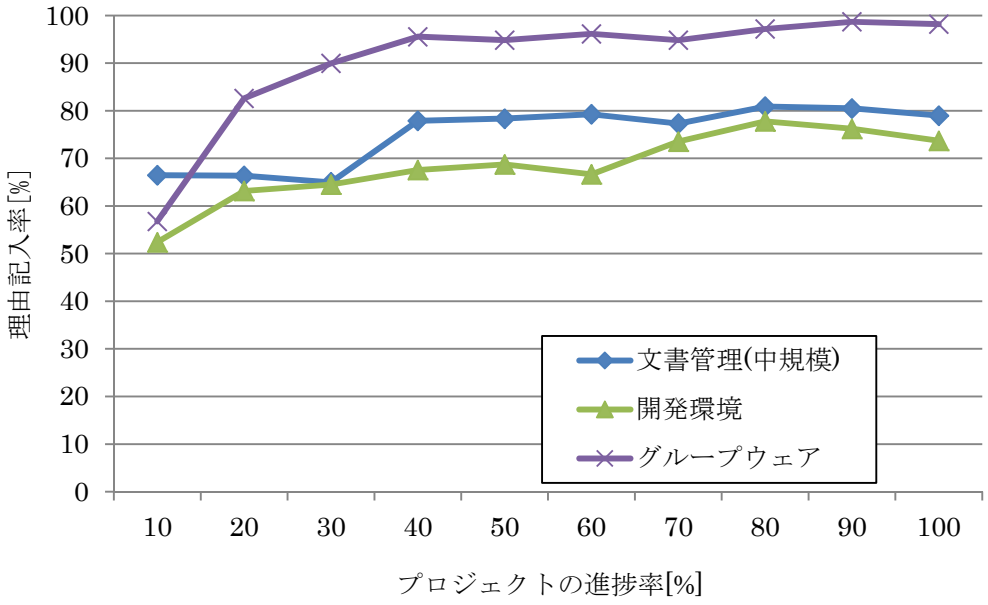


図 2-20 プロジェクトの進捗率に対する理由記入率の推移

2.5.3 議事録作成工数，レビューの十分性の評価にかかる時間の評価

本項では，議事録の作成工数，レビューの十分性の評価にかかる時間について評価する。表 2-1 に示した通り，プロジェクト当たりの議事録件数は平均で約 350 であった。また，議事録一件当たりの議題数は平均で約 10 件であった。そこで，350 回分の設計レビューに対して，(1) 10 件の設計項目を含む設計レビュー議事録の作成工数，(2) レビューの充実度を評価する時間，(3) 1 つの設計項目についてレビューの質を評価する時間，の 3 点について，従来方式と比較し，評価することとする。提案方式により従来方式に比べて十分短い時間でその作業が実施できることで，設計レビューのプロセスの効率化がなされたと評価する。なお，ここでの従来方式とは，レビュー実施単位にテキストファイルで議事録を作成・蓄積する方式とする。

(1) 10 件の設計項目を含む設計レビュー議事録の作成工数

10 件の設計項目を含む設計レビュー議事録の作成にかかった工数の計測結果を図 2-21 に示す。従来方式における工数は，設計開発現場の設計者がテキストエディタで議事録を作成する場合の平均工数とした。一方，提案方式における工数は，レビュー項目表から項目を選択して議事録の雛形を生成し，議事内容を記述することで議事録を完成させるまでの工数とした。議事録の雛形を生成することによって，見出し部分の入力が省けるため，議事録作成時間は短縮したものの，見出しよりも議事内容の記述時間の割合が多いため，効果は限定的であった。

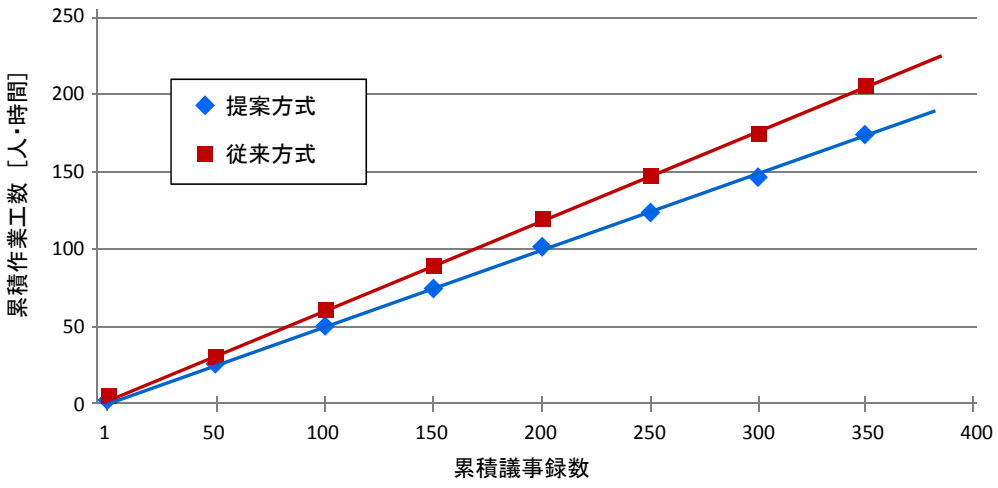


図 2-21 設計レビュー議事録の作成工数を比較

(2) レビューの充実度を評価する時間

レビューの充実度の評価にかかった時間の計測結果を図 2-22 に示す。従来方式におけるレビュー充実度の評価時間は，レビュー実施回数集計表の更新を表計算ソフトウェアによって，議事録が登録されるたびに目視にて行う場合の平均時間とした。一方，提案方式における評価時間は，本研究で開発した設計レビュー管理システムがレビュー実施回数集計表を表示する時間とした。

前述の通り，レビュー実施回数を手作業で集計する場合，各回のレビューの議事録が提出されたタイミングで，該当する設計項目を確認して表計算ソフトウェアに記録する。この作業は，おおよそ一件当たり 30 秒程度で終わるというヒアリング結果を得ている。一方，本研究で開発した設計レビュー管理システムでは，プロジェクト管理者は能動的に議事録件数をカウントすることはせず，本システムの表示画面を確認するのみである。処理方式としては，毎回，議事録データベースから集計する方式であり，件数が増えれば集計時間は増えるが，プロジェクト終盤の登録済み議事録件数の平均 350 件であっても，4 秒程度でレビュー実施状況集計表を表示でき，実用上問題ないことが確認できた。

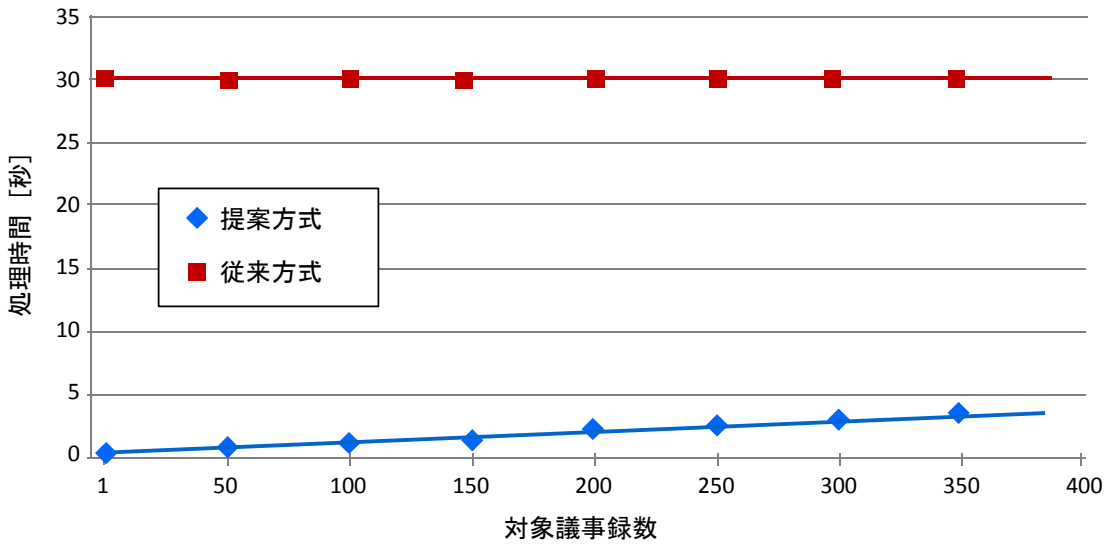


図 2-22 レビューの充実度の評価にかかる時間の比較

(3) 1 つの設計項目についてレビューの質を評価する時間

1 つの設計項目に対するレビューの質の評価にかかった時間の計測結果を図 2-23 に示す。従来方式におけるレビューの質の評価時間は，テキストエディタで作成された議事録を共有フォルダに蓄積した状態で，OS の全文検索機能を使用し，該当する設計項目を含む議事録を検索し，レビュー実施日の古い順に，目視にて当該議事内容部分を確認する時間とした。一方，提案方式における評価時間は，本研究で開発した設計レビュー管理システムが時系列串刺し表示を表示する時間とした。

表 2-1 から，1 つの設計項目に対するレビューは，平均 13 回程度で完了することが分かっており，そのため，OS の全文検索機能を使ってもヒットするのは平均 13 件程度となる。ヒットした 13 件に対して 1 件当たり 10 秒で該当議事を発見できるため，1 つの設計項目のレビューの質の評価にかかる時間は，単純計算で 130 秒となる(読んで理解する時間は含めない)。

一方で，本システムの時系列串刺し表示は，議事録データベースの見出しを格納したカラムから，指定された設計項目に該当する議事情報を抽出して，設計レビューの実施日の古い順に連結して表示する。ユーザから指示されるたびにこの処理を実行するため，議事録の登録数が増加すると処理時間は長くなるが，議事録登録数が 350 程度であれば，4 秒以内に処理でき，実用上問題ないことが確認できた。

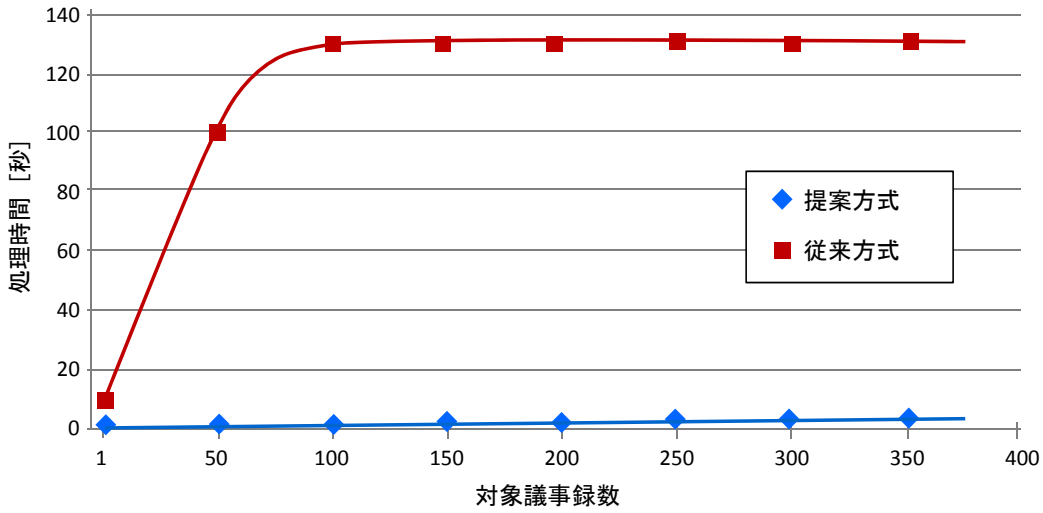


図 2-23 レビューの質の評価にかかる時間の比較

2.5.4 本システムの実用上の有効性に関する評価

本システムは 1999 年 8 月の適用開始後，2 度のエンハンスを行い，2002 年 8 月時点で，（株）日立製作所内の公共向け事業部において，累計 900 の開発プロジェクトにて利用され，レビュー議事録の登録件数は約 20,000 件に達した。これらのプロジェクトは，官庁や地方自治体，大学，病院，金融機関，流通・サービス業など多岐にわたる業種の財務会計，税金，給与といった基幹業務，及び，業務用の特注システムを開発する大規模プロジェクトから，社内の事務や開発支援のツールを開発する小規模プロジェクトまでさまざまであったが，基本的には情報システムの開発プロジェクトであった。

それから 9 年，適用開始から 12 年たった，本論文執筆中の 2011 年 6 月現在では，日立グループ内の 6 つの事業部で利用され，利用しているプロジェクトは情報システムの設計開発部門だけでなく，調達部門や企画部門，総務部門にも広がっている。1999 年 8 月から 2011 年 6 月までの期間に本システムを利用したプロジェクトは，累計 2,469 となっており，累計議事録数は，100,724 件に達している。

以上の実績により，提案方式の実用上の有効性を証明できているものと考える。

2.6 結言

本章では，設計レビューのプロセスを効率化するために，設計レビューの議事録を活用することによって，決められた設計項目に対して十分な設計レビューを実施したか，レビューでの議論は充実したものだったかを確認する作業を支援する方式を提案した。さらに，本方式を実装した設計レビュー管理システムを現場適用し，評価を行った。

本システムに対して，品質保証部門の担当者からは，「重点設計項目について，レビューを実施したかどうかを確認できるのがうれしい」という意見を得ることができ，提案方式の有効性が確認できた。加えて，「レビュー会議の冒頭で，プロジェクタで当該レビュー項目の時系列串刺し表示を投影することによって，前回までの議論過程を出席者全員で確認できて無駄な議論が排除できるようになった」という意見から，提案時点で想定していなかったレビュー会議を IT によって支援するシステムへの拡

張可能性も確認できた。また，本システムは，当初より設計開発現場への適用を想定して，開発したが，最初のサービス提供から 12 年たった現在では，開発プロジェクトだけではなく，調達部門や企画部門，総務部門にも利用されるようになった。この実績から，提案方式の実用上の有効性を証明できているものと考える。

今後の課題は，議事内容と設計ドキュメントの細粒度のマッピングによって，設計経緯の把握をさらに支援することである。議事録には，その時のレビューの対象となる設計ドキュメントを添付することをルールとしているが，設計ドキュメントの記載内容と議事録の記述のマッピングは記録していない。そのため，議事録作成者にとって，結論の要約を記述する手間が余計にかかり，且つ，議事録閲覧者にとって，議事録から設計ドキュメントの該当箇所を参照する手間の低減は，本章で提案した方式だけでは十分であるとは言えない。設計ドキュメントの該当箇所にコメントを付けていく要領で議事録をより容易に作成できるようにするなど，さらなる改善が必要である。

第3章
情報システムの障害原因解析のためのルール記述方法と解析実行方式

3.1 緒言

本章では，情報システムの障害原因解析のための解析ルールのデータ形式，及び，解析実行方式について述べる。

情報システムの運用管理は，サーバやストレージ，ネットワークスイッチなど，IT機器ごとに専門の管理者が置かれ，個別に管理することで行われてきた。しかし，これらのIT機器はお互いに依存し合って動作するため，ある機器で障害が発生すると，別の機器にも障害が波及する。このため，各機器の管理者が個別に障害対応していては効率が悪い。近年の情報システムは，オープン化や仮想化により，以前にも増して，大規模化，複雑化してきており，IT機器間のつながりを把握して，連鎖的に発生する障害イベントの因果関係を整理することが，障害復旧時間短縮のために必要となってきている。また，障害が発生した情報システムからは，障害解析に必要十分な情報が得られない場合があることも，その難しさを増加させている。例えば，情報収集に必要なネットワークが故障した場合，期待した障害イベントが通知されないことがある。また，障害により，IT機器が完全に反応しなくなることもあり，この場合は，反応しなくなったことが検知できるだけで，より詳細な情報は期待できない。このような課題に対しては，障害解析ルールを表形式で表現して解析を行う手法が提案されているが，ルールの数が多くなると解析時間が増加するという問題がある。

そこで，本研究では，Reteアルゴリズムをベースとし，解析に必要な障害イベントがすべて揃わなくても結論を導くことができ，且つ，ルール数に依存して解析時間が増加しない解析ルールのデータ形式と解析処理方式を提案する。

以下，3.2節で障害原因解析システムの概要と，その実現に向けての課題，本研究の

アプローチを述べ，3.3 節で提案する解析ルールの記述方法，3.4 節でトポロジーに合わせてルールを展開し構築する方法，3.5 節で，機器の状態異常の検出方法，3.6 節で解析処理の実行方式について述べる。3.7 節でユーザインタフェースについて述べ，3.8 節で提案する方式の評価と考察を行う。

3.2 障害原因解析システムの概要

3.2.1 情報システムの障害原因解析

情報システムはサーバ装置やネットワーク機器，ストレージ装置など，複数の IT 機器で構成され，それぞれが依存し合って動作する。例えば，ストレージ装置が提供するボリュームを，ネットワークを介して複数のサーバ機器で共有する NAS (Network Attached Storage)の構成や，Web サーバと Application サーバ，DB (Database)サーバが連動してサービスを提供する Web3 階層アプリケーションの構成などである。これらの構成において，構成要素の 1 つが障害を起こしたときには，依存動作していた別の構成要素からもエラーが報告されるなど，構成要素(機器)の接続関係に沿って障害が伝播し，監視サーバに障害イベントが多数報告されることになる。運用管理者は，報告された多数の障害の中から，どの機器のどの問題が根本原因であるのか早急に特定しなければならない。

本項では，従業員数約 500 人の研究所における IT 環境の例をリスト 1 に示し，発生し得る障害の種類とそれに対する障害原因解析手順の具体例を示す。また，リスト 1 に示した研究所のモデルを IT 機器の接続関係として図示すると，図 3-1 のようになる。図 3-1 に示した管理対象システムを，接続関係(トポロジー)の観点で分析すると，次に示すように，複数種類のトポロジーが存在することが分かる。そして，それぞれのトポロジーに対する障害原因解析は次のように行う。

リスト 1：

■リモートデスクトップ提供モデル

従業員 500 人分のデスクトップ環境が社内データセンタ内のサーバで稼動しており，各従業員は自席 PC (Personal Computer) からリモートデスクトップ経由で利用している。

■ネットワーク経由ディスクスペース提供モデル

各デスクトップ環境には，10GByte の iSCSI(Internet Small Computer System Interface)ドライブが割り当てられている。

■ファイル共有ディスクスペース提供モデル

部署内のファイル共有向けに，上記デスクトップ環境，及び，自席 PC からアクセス可能なファイル共有ディスクスペースを利用している。

■仮想サーバ提供モデル

一部の従業員は，研究開発実験用として，仮想サーバを利用している。仮想サーバへのアクセスは，各従業員が持つリモートデスクトップ環境から行う。

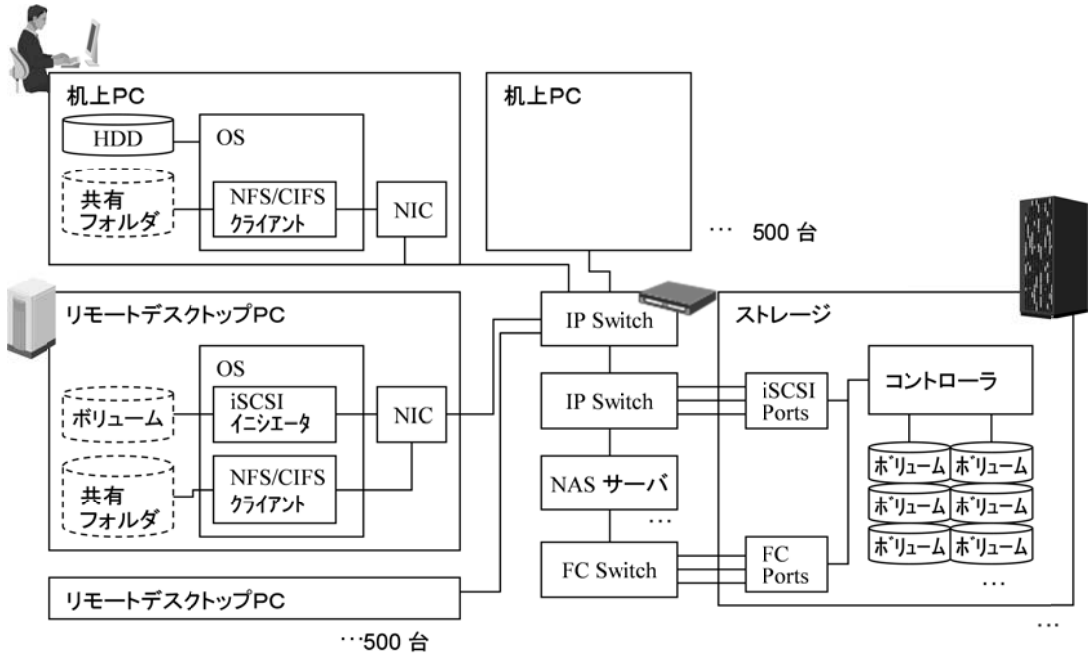


図 3-1 500 人の研究所における IT 環境

(1) IP-Network (Internet Protocol based Network)トポロジー

リモートデスクトップ提供モデルでは，サーバ装置で動作するデスクトップ環境を，1 つ，または，複数の IP-Switch を経由して，リモートデスクトッププロトコルを使って自席 PC に提供する。このような IP-Network を利用した接続形態を IP-Network トポロジーと呼ぶ。このトポロジーでの典型的な障害は，自席 PC からサーバに接続できないという障害である。原因としては，経由するいずれかの IP-Switch のエラー，自席 PC の NIC (Network Interface Card)エラー，サーバの NIC エラーが考えられ，運用管理者は，これらの可能性を 1 つ 1 つチェックすることで原因を特定する。

(2)IP-SAN (Internet Protocol based Storage Area Network)トポロジー

ネットワーク経由ディスクスペース提供モデルでは，ストレージ装置のボリュームを 1 つまたは複数の IP-Switch を経由して，iSCSI 技術を使ってサーバに提供する。このような iSCSI 技術を利用した接続形態を IP-SAN トポロジーと呼ぶ。このトポロジーでの典型的な障害は，サーバ側で，提供されたディスクにアクセスできないという障害である。原因としては，サーバの NIC エラー，サーバ OS で動作する iSCSI イニシエータの設定ミスまたはプログラムエラー，サーバとストレージ装置間の IP-Switch のポートエラー，ストレージ装置の iSCSI ポートエラー，ストレージ装置のコントローラエラー，ストレージ装置のボリュームエラーが考えられる。運用管理者は，これらの可能性を 1 つ 1 つチェックすることで原因を特定する。

(3)File-Share トポロジー

ファイル共有ディスクスペース提供モデルでは，ストレージ装置のボリュームを NAS 装置またはファイル共有サーバを経由して，さらに 1 つまたは複数の IP-Switch を経由して，複数のサーバに提供する。このような NAS 装置やファイル共有サーバを使ってファイル共有する接続形態を File-Share トポロジーと呼ぶ。このトポロジーでの典型的な障害は，それぞれのサーバでディスクがアクセスできないという障害である。原因としては，サーバの NIC エラー，サーバ OS で動作する Samba クライアントの設定ミスまたはプログラムエラー，サーバと NAS 装置間の IP-Switch のポートエラーが考えられる。運用管理者は，これらの可能性を 1 つ 1 つチェックすることで原因を特定する。

(4)FC-SAN トポロジー

ファイル共有ディスクスペース提供モデルでは，File-Share トポロジーに加えて，ストレージボリュームを NAS 装置やファイル共有サーバに，FC-Switch (Fibre Channel Switch)を経由して提供する。このような FC-Switch を使ってボリューム提供する接続形態を FC-SAN トポロジーと呼ぶ。このトポロジーでの典型的な障害は，NAS 装置やサーバでボリュームにアクセスできないという障害である。原因としては，ストレージ装置と NAS 装置またはサーバ間の FC-Switch の FC ポートエラー，ストレージ装置のコントローラエラー，ストレージ装置のボリュームエラーが考えられる。運用管理

者は，これら IT 機器とそれを構成するコンポーネントを 1 つ 1 つチェックすることで障害原因箇所を特定する。

(5) VM-Host トポロジー

仮想サーバ提供モデルでは，物理サーバ上で動作するサーバ仮想化ソフトウェア(ハイパーバイザ)が，1 つまたは複数の仮想サーバを実行する。このようなハイパーバイザを利用した形態を VM-Host (Virtual Machine - Host)トポロジーと呼ぶ。このトポロジーでの典型的な障害は，仮想サーバの性能障害である。原因としては，ハイパーバイザの負荷上昇，仮想サーバの負荷上昇が考えられる。運用管理者は，ハイパーバイザとその上で動作するすべての仮想サーバの負荷を 1 つ 1 つチェックすることで，障害原因を特定する。

3.2.2 従来技術による障害原因解析

運用管理者は，一般に，監視ツールを用いて対象の IT 機器の稼動監視を行う。監視ツールは，監視対象の IT 機器で何らかの不具合が発生した場合に運用管理者に障害イベントを通知する。運用管理者はこの障害イベントの一覧から障害の原因を特定し，これを取り除いてシステムを復旧させる。

図 3-2 は，典型的な監視ツールの構成であるが，障害は，サーバ装置，ネットワーク機器，ストレージ装置，それぞれの監視ツールで検出するため，検出した障害イベントを一覧表にまとめることはできても，それぞれの障害の因果関係の集約を行って，障害原因を特定することはできない。そのため，運用管理者は 1 つ 1 つの障害イベントを目視で確認し，障害の原因を特定する必要があるが，数百，数千の IT 機器を有するデータセンタでは，IT 機器とそれを構成するコンポーネントを 1 つ 1 つチェックするには数が多く，障害原因箇所の特定は一日掛かりの作業となっている。

3.2.3 RCA システム

本章で提案する障害原因解析システム(以降，RCA システム)は，サーバ装置，ネットワーク機器，ストレージ装置に跨る接続関係(トポロジー)を予め検出しておき，さらに，対象となる情報システムから連鎖的に通知される障害イベントを収集し，図 3-3

に示すように，3.2.1 項で述べた手順を自動化することで，障害原因を示す障害イベントの特定作業の手間を軽減する。RCA は Root Cause Analysis の略であり，障害原因解析と訳される。

RCA システムは，図 3-4 に示すように，汎用記述された解析ルールを対象システムのトポロジーに即した形にインスタンス展開し，ルールメモリを構築する。ルールメモリは，汎用記述された解析ルール群と監視対象システムのトポロジーを掛け合わせて展開された解析用のデータである。監視対象から障害イベントを受信した際には，その障害イベントをルールメモリに書き込み，条件の成立具合で障害原因を推論する。

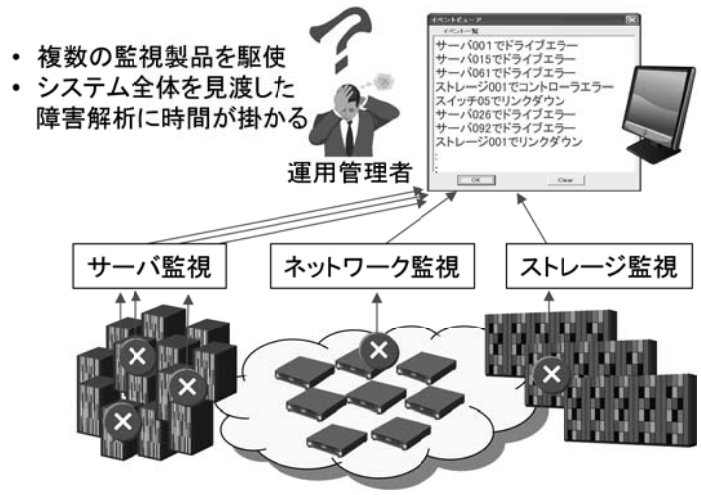


図 3-2 個別ツールによる障害箇所の特定

図 3-3 障害原因解析機能による障害箇所の特定

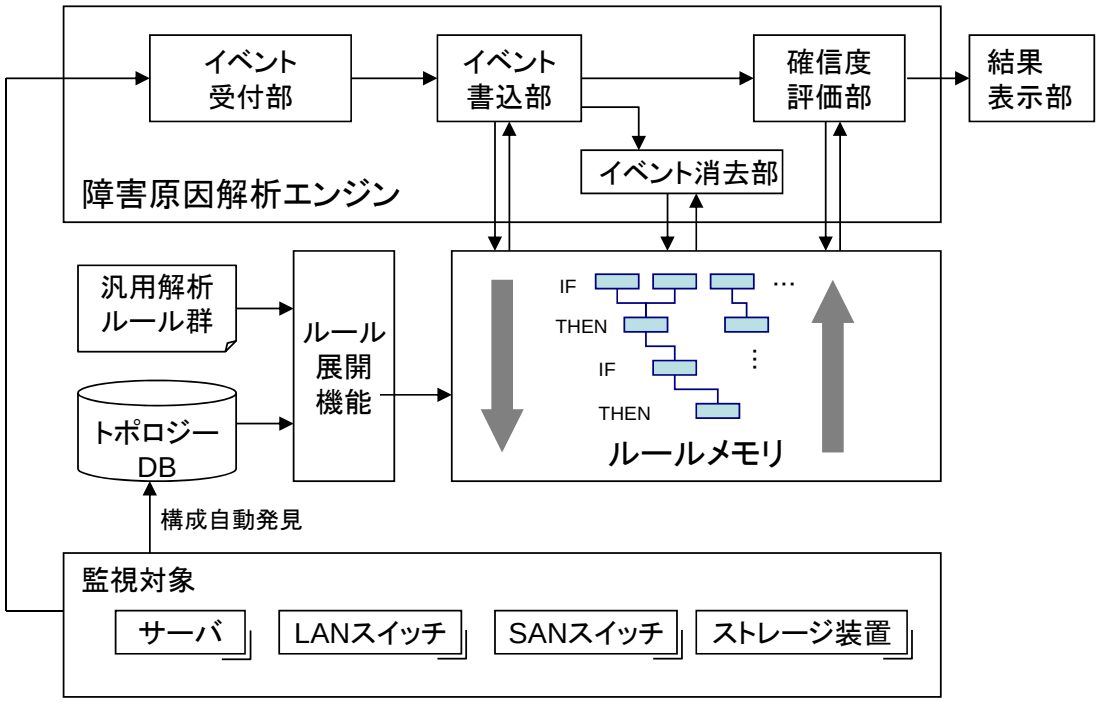


図 3-4 RCA システムのアーキテクチャ

情報システムにおける障害は，3.2.1 項で述べた通り，トポロジーを構成する各 IT 機器に伝播する。監視対象システムを構成する機器間の依存関係にしたがって解析ルールを手動で構築するとなると，IT 機器の組み合わせの数が多すぎて，ルールメモリの構築に時間がかかり過ぎる。そのため，障害伝播パターンの知識のある運用管理者が直感的に記述でき，且つ，監視対象システムのトポロジーに強く依存しない汎用記述可能な「解析ルールの記述方法」と「解析ルールを監視対象システムに内在するトポロジーに合わせてルール展開する方法」が求められる。

また，情報システムにおいては，障害の程度によって，障害イベントが生成されない場合や，通信経路上の機器が故障する等の問題で障害イベントが RCA エンジンに届かない場合，さらには，障害イベントの到着に時間差が生じ，順序性の保証が期待できない場合がある。このような悪条件の中であっても，与えられた障害イベント情報のみから，可能な限り確からしい解析結果が得られる「解析処理方式」が求められる。

3.3 汎用解析ルールの記述方法

3.3.1 汎用解析ルール記述の文法

3.2.1 項で述べたように，障害原因解析はトポロジーに基づいて行う。人手で行っていた解析を自動化するための汎用解析ルール記述の要件は，以下のように導き出せる。

(1) トポロジーに対して条件を記述できること。
(2) 障害イベントの共起条件を記述できること。

これらの要件を満たす汎用解析ルール記述の文法を表 3-1 に示す。ひとつの汎用解析ルールは，条件部と結論部のペアで構成する。一般的に馴染みがあり，直感的に理解できることから，IF-THEN の形式とする。また，サーバ，ネットワーク機器，ストレージ装置など，装置をまたがって障害原因解析を行うため，IF 部で指定する条件には，装置種別と，その装置を構成するリソースの種別，そのリソースから生成されるイベント種別を指定する。また，THEN 部には，解析の結果として，装置種別と，その装置を構成するリソースの種別とそのリソースから生成されるイベント種別を記述する。

一方，要件(1)を満たすためには，監視対象システムのシステム構成を表現する標準的なモデルが必要となる。システム構成を表現する方法として，業界標準の情報モデルである DMTF の CIM[90]が利用できる。CIM を利用した管理モデルは，サーバ系では SMASH (Systems Management Architecture for Server Hardware)[91]，デスクトップ/モバイル系では DASH (Desktop and mobile Architecture for System Hardware)[92], 仮想サーバ系では VMAN (Virtualization Management)[93]，ストレージ系では SNIA (Storage Networking Industry Association)の SMI (Storage Management Initiative)[94]にてそれぞれ標準化されている。CIM を用いることで，IT 機器やそれを構成するリソースを管理オブジェクトとして表現でき，管理オブジェクト間の関連や包含関係を表現することが可能となる。

表 3-1 汎用解析ルール記述の文法

<汎用解析ルール> ::= <IF 部> <THEN 部>

<IF 部> ::= 'IF' <条件>+

<条件> ::= '<' <装置指定部> ('[' <リソース指定部> <イベント指定部> ('FREQ('数値')')? ']')* '>'

<装置指定部> ::= <装置種別> | <装置種別> '.' <関係演算子>

| <装置種別> '.' <関係演算子> '(' <演算子条件> ')'

<装置種別> ::='server' | 'ip-switch' | 'fc-switch' | 'storage'

<関係演算子> ::= 'contain' | 'belong' | 'associate' | 'providedIpSAN' | 'providedFcSAN' | 'providedFileShare' | 'directConn' | 'hostedVM'

<リソース指定部> ::= <リソース種別>

| <リソース種別> '.' <関係演算子> '(' <演算子条件> ')'

| <リソース種別> '.' <プロパティ名> '.' <関係演算子> '(' <演算子条件> ')'

<リソース種別> ::= 'DiskDrive' | 'IscsiDrive' | 'IscsiInitiator' | 'Port' | …

<イベント指定部> ::= <イベント演算子>? <外部イベント>

<イベント演算子> ::= 'NOT'

<外部イベント> ::= 'Error' | 'CommErr' | 'LinkDown' | …

<THEN 部> ::= 'THEN' <結論部>

<結論部> ::= '<'<装置種別> 'Result(' <リソース種別> <外部イベント> ')>'

ただし，CIM の管理オブジェクトのみでルールを表現するには複雑であり，直感的に記述しづらいため，ルール記述者が物理的に捉えやすいサーバ，スイッチ，ストレージ装置については，IT 機器間の接続関係を単純に縦に並べて記述することとし，IT 機器を構成するリソースについては，オブジェクト指向言語で良く見られるドットを用いた表現で関係演算子を用いて，ルールを適用するトポロジーを指定できるようにする。また，サーバ装置，IP スイッチ，FC スイッチ，ストレージ装置，仮想サーバなどで構成されるトポロジーについては，いくつかのパターンに分類できるため，パターン化できるトポロジーについては，記述工数低減，記述ミス防止を目的として，トポロジーパターンを指定する関係演算子を導入する。関係演算子を表 3-2 に示す。リソースの包含関係，接続関係を指定する際に用いる。

表 3-2 関係演算子

#	演算子	意味と記述例
1	contain	ある管理対象を包含する管理対象。 例) LAN ポートを持つネットワークアダプタ： NetworkAdapter.contain(LanPort)
2	belong	ある管理対象に包含されている管理対象。 例) NetworkAdapter に属している LAN ポート： LanPort.belong(NetworkAdapter)
3	associate	ある管理対象に関連がある管理対象。 例) iSCSI イニシエータと関連のあるドライブ： DiskDrive.associate(IscsiInitiator)
4	provided IpSan	ストレージ装置から IP-SAN を介して LUN が提供されているサーバ。 例) Server.providedIpSan
5	provided FCSan	ストレージ装置から FC-SAN を介して LUN が提供されているサーバ。 例) Server.providedFCSan
6	provided FileShare	ファイル共有サーバからファイルが提供されているサーバ。 例) Server.providedSharedFile
7	hostedVM	仮想化機構により実行されている仮想マシン。 例) Server.hostedVM
8	hosted VirtualDisk	仮想ハードディスクを提供されている仮想マシン。 例) Server.hostedVirtualDisk
9	directConn	サーバもしくはストレージ装置に直接接続している IP スイッチ。 例) IpSwitch.directConn(Server)

また，要件(2)を満たすために，解析ルールには，「ある IT 機器のある部位(リソース)からの障害イベントと，別の装置の別のリソースからの障害イベントが同時に発生したとき」という形でルールを記述できるようにする。「ある IT 機器のある部位からの障害イベント」に対応するルール記述としては，リソース指定とイベント指定を単純に横並びに記述することで直感的に記述できるようにし，「障害イベントが同時に発生した」に対応するルール記述としては，単純に，1 つのルールに，リソース指定とイベント指定の組を複数記述すれば良いようにする。

条件を記述する際に一般に用いられる AND や OR などの演算子は不要とし，1 つのルールに出現する条件は，すべて AND として扱う。複数ルール間の関係は OR として扱うことで，OR を表現したい場合は，条件の異なるルールを別途記述すれば良いようにする。要件(1)，及び，(2)を満たす汎用解析ルール記述の概略を図 3-5 に示す。

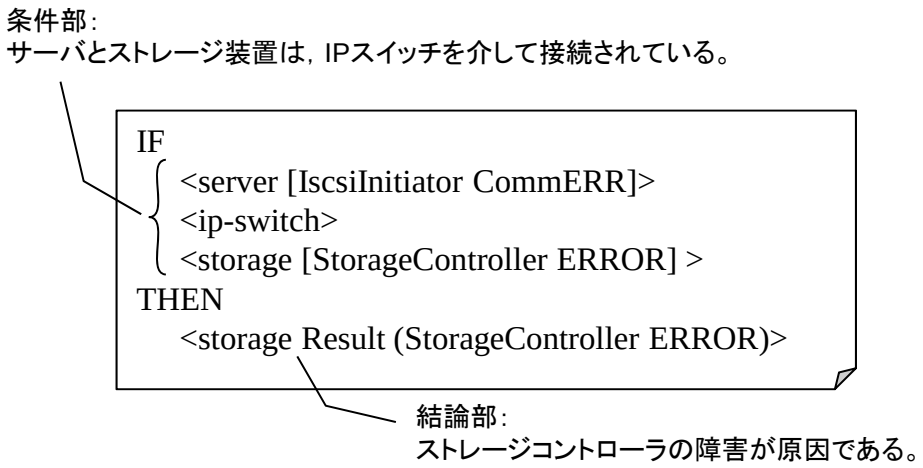


図 3-5 汎用解析ルール記述の概略

3.3.2 汎用解析ルールの作成手順

本項では，3.2.1 項で挙げた 5 つのトポロジーのうち，IP-SAN トポロジーに対する解析ルールを例として，その作成手順を示す。

障害原因解析はトポロジーに基づいて行うため，はじめにトポロジーの管理モデルを作成する。IP-SAN トポロジーの管理モデルは，図 3-6 のように表現できる。図 3-6 によれば，サーバは iSCSI イニシエータを有しており，LAN (Local Area Network)ポート経由で IP 通信を行い，ストレージ装置内のボリュームを利用している。ストレージ装置は，装置内のボリュームを iSCSI ポート経由でサーバに提供している。

次に，障害伝播範囲の確認を行う。管理モデルに基づいて，トポロジーを構成する各管理クラスから関連を辿っていくことで障害原因となり得る不具合を列挙する。例えば，サーバにおいて iSCSI ドライブが参照できなくなる障害は，図 3-6 の IscsiDrive クラスで障害が発生することを意味する。そのため，IscsiDrive クラスから関連を辿っていって，障害原因となり得る不具合を列挙する。すなわち，(a) iSCSI ドライブの動作の前提となる iSCSI イニシエータの通信障害(IscsiInitiator クラスの CommERR)，(b) iSCSI イニシエータの動作条件になっている LAN ポートの通信障害(LanPort クラスの CommERR)，(c) LAN ポートの通信相手である IP スイッチのポートの通信障害(Port ク

ラスの LinkDown)，(d) IP スイッチのポートに接続された，ストレージの iSCSI ポートの通信障害(IscsiPort クラスの LinkDown)，(e)ストレージの iSCSI ポート経由でボリュームを提供するストレージコントローラの状態異常(StrageController クラスの ERROR)，の 5 つが列挙できる。

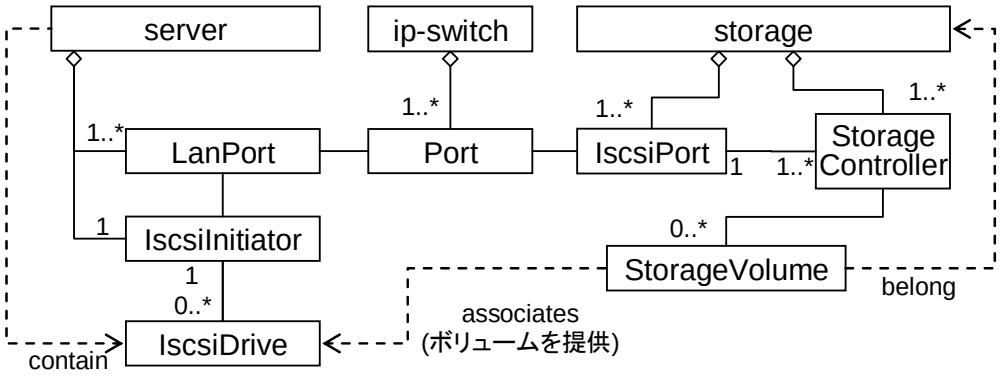


図 3-6 IP-SAN トポロジーの管理モデル

次に，共起条件の列挙を行う。列挙した原因候補それぞれについて，それが障害原因であるための共起条件を挙げる。例えば，(c)の IP スイッチのポートの通信障害の場合，IP スイッチのポートの通信障害と iSCSI イニシエータの通信障害が報告され，ストレージの障害は報告されないことが，IP スイッチのポートの通信障害が障害原因であると特定する条件となる。

最後に，共起条件と障害原因を対にして汎用解析ルールを記述する。例を図 3-7 に示す。(c)と(e)に対する例であり，共通して iSCSI イニシエータの通信エラーが条件として指定されているが, (c)として結論づけるルールには, IP スイッチのリンクダウン，(d)として結論づけるルールには，ストレージコントローラエラーを，それぞれ条件として加えている。

3.2.1 項で挙げた他のトポロジーについても同様に，3. 4. 2 項の汎用解析ルールの作成手順により，汎用解析ルールを作成する。表 3-3 に示すように，5 つのトポロジーに対する汎用解析ルールと装置単体に対する汎用解析ルールを合計して 337 個の解析ルールを作成した。

(c) IPスイッチのポート障害を根本原因として検出するルール

```
IF
    <server.providedIpSan [IscsiInitiator CommERR]>
    <ip-switch [Port LinkDown]>
    <storage>
THEN
    <ip-switch Result (Port LinkDown)>
```

(e) ストレージ装置のコントローラエラーを障害原因として検出するルール

```
IF
    <server.providedIpSan [IscsiInitiator CommERR]>
    <ip-switch>
    <storage [StorageController ERROR] >
THEN
    <storage Result (StorageController ERROR)>
```

図 3-7 トポロジーパターン指定による解析ルール

表 3-3 作成した汎用解析ルールの数

トポロジーパターン	汎用解析ルール数
(1) IP-Network トポロジー	6
(2) IP-SAN トポロジー	71
(3) FC-SAN トポロジー	63
(4) File-Share トポロジー	10
(5) VM-Host トポロジー	27
単体，その他	160
合計	337

3.4 汎用解析ルールとトポロジーによるルールメモリ構築方法

3.4.1 ルール展開処理の動作概要

汎用解析ルールを実トポロジーに合わせて展開し，ルールメモリを構築する処理をルール展開処理と呼ぶ。ルール展開処理は，図 3-4 に示した RCA システムのアーキテ

クチャのルール展開機能(Rule Expander)によって実行され, 汎用解析ルール群とトポロジーを入力とし，ルールメモリを構築する。ルール展開処理に先立って，トポロジーが自動発見機能によってトポロジーDB に格納されていることと, 汎用解析ルールが定義されていることが前提条件となる。

3.4.2 ルール展開処理手順

ルール展開処理の流れを図 3-8 のフローチャートに基づいて説明する。図 3-8 に示すように，ルール展開処理は，はじめに汎用解析ルール群を読み込んで，ルール記述内で指定されているトポロジーを重複なく抽出して列挙する(Step1)。そして，列挙したトポロジーが, 実際の監視対象システムに存在するかどうかトポロジーDB を検索してチェックする(Step2)。具体的には，providedIpSan のトポロジー指定された汎用解析ルールがあった場合には, 図 3-9 に示すような IP-SAN の実トポロジーが存在するかチェックする。存在する場合は，そのトポロジーを指定している汎用解析ルールをすべて読み込んで(Step3)，機器の識別子，例えばサーバ名やスイッチ識別子など具体的な識別情報を，汎用解析ルールの装置種別指定部分に当てはめ(Step4)，条件オブジェクトと結論オブジェクト，ルールオブジェクトとして構造化する(Step5)。これを Step1 で列挙したトポロジー全てに対して繰り返し(Step6)，ルールメモリを完成させる(Step7)。

3.4.3 ルールメモリのデータ構造

ルールメモリは，汎用記述された解析ルール群と監視対象システムのトポロジーを掛け合わせて展開された解析用のデータである。ルールメモリのデータ構造を図 3-10 に示す。ルールメモリは，解析ルールの IF 部を構成する各要素に相当する条件オブジェクトと，THEN 部の結論に相当する結論オブジェクト，条件と結論を結びつけるルールオブジェクト，及び，それらの接続関係によって表現する。図 3-10 のルールメモリを IF-THEN 形式で表現すると，図 3-11 のようになる。

図 3-11 のルール 1 は，IF 部の要素として「A」と「B」と「C」を持ち，THEN 部に「結論①」を持つ。図 3-10 において，ルールオブジェクト「ルール 1」には，条件オブジェクト「条件要素 A」と「条件要素 B」，「条件要素 C」が関連付けられている。

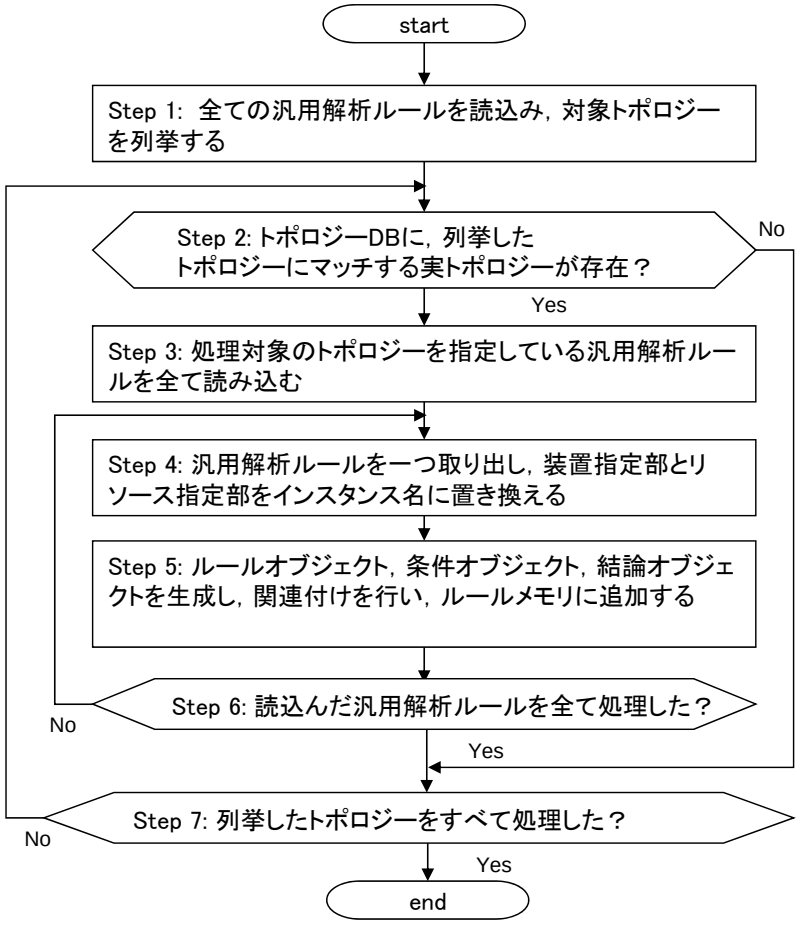


図 3-8 ルール展開処理手順

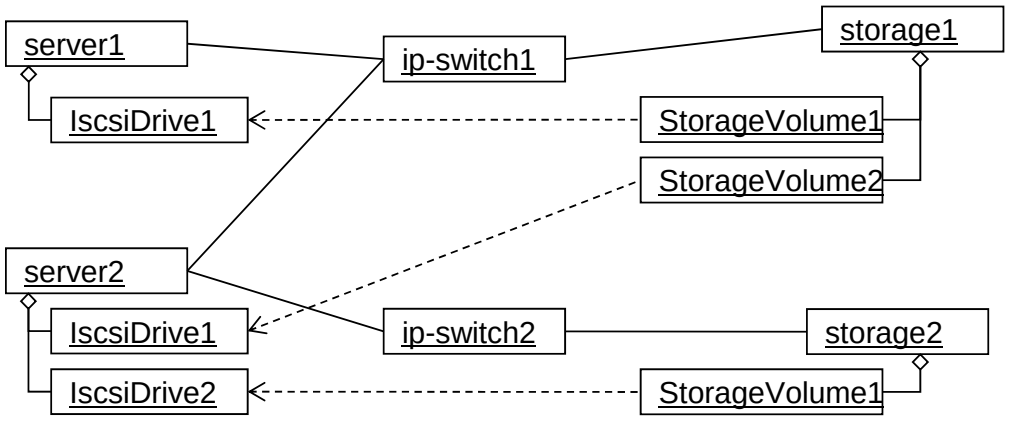


図 3-9 IP-SAN の実トポロジー

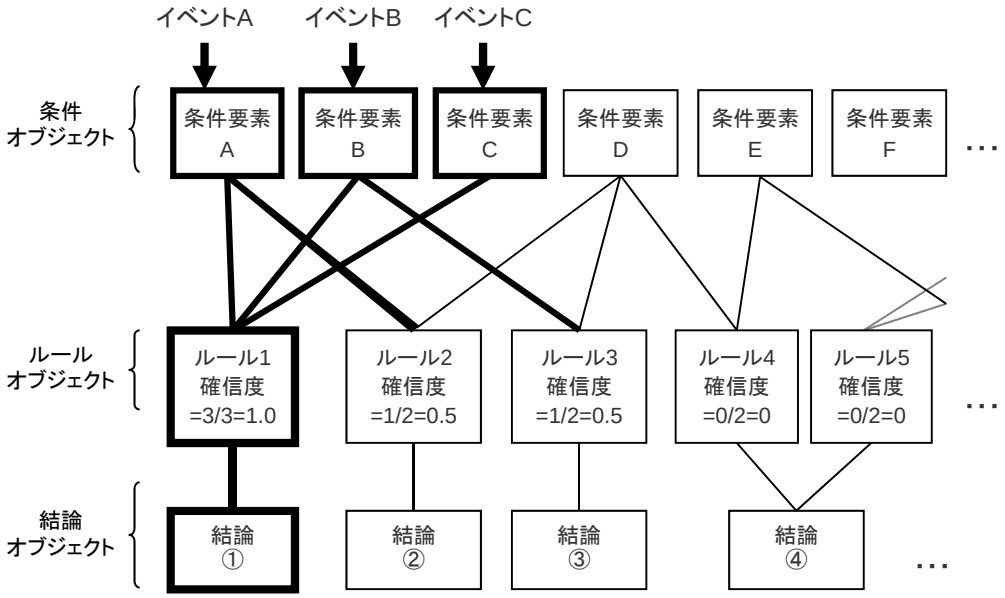


図 3-10 ルールメモリの構造

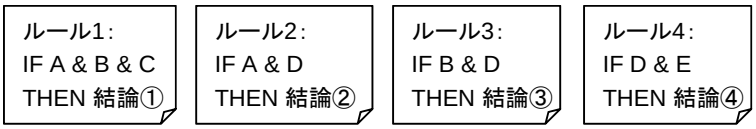


図 3-11 ルールメモリの IF-THEN 表現

また，ルールオブジェクト「ルール 1」には，結論オブジェクト「結論①」が関連付けられている。また，図 3-11 のルール 2 は，IF 部の要素として「A」と「D」を持ち，THEN 部に「結論②」を持つ。図 3-10 において，ルールオブジェクト「ルール 2」には，条件オブジェクト「条件要素 A」と「条件要素 D」が関連付けられている。また，ルールオブジェクト「ルール 1」には，結論オブジェクト「結論②」が関連付けられている。図 3-11 で，「ルール 1」と「ルール 2」で共通して条件要素となっている「条件要素 A」が，図 3-10 の表現では，1 つの条件オブジェクトとすることが重要である。イベント A を受信したときには，条件オブジェクト「条件要素 A」を発火させる。「条件要素 A」が発火することによって，「条件要素 A」と「ルール 1」の結線，及び，「条件要素 A」と「ルール 2」の結線を活性化する。このように，複数の

ルールに共通して指定される条件要素を 1 つの条件オブジェクトとすることで，対応するイベントを受信したときに，どの条件オブジェクトを発火させるかの検索処理を一回に限定することができる。

3.5 機器の状態異常の検出方法

本節では，監視対象機器の状態異常を検出する方法について述べる。一般的に，監視対象機器の状態異常を検出する方法は，大きく分けて次の二つがある。1 つは，監視対象機器で動作する監視エージェントに状態を監視させて，監視結果を監視サーバに通知してもらう方法で，エージェント方式と呼ばれるものである。もう 1 つは，監視サーバが監視対象機器に直接アクセスして状態を取得する方法で，エージェントレス方式と呼ばれるものである。

本研究では，主にネットワーク機器に対する監視に，エージェント方式として広く用いられている SNMP (Simple Network Management Protocol)の Trap を利用する。サーバに対しては，監視対象が Microsoft 社の Windows が稼働するサーバであれば，WMI (Windows Management Instrumentation)というインタフェースを利用し，Linux などの Unix 系サーバであれば，SSH (Secure Shell)の仕組みを利用する。ストレージ装置に対しては，SMI-S (Storage Management Initiative – Specification)で規定された業界標準インタフェースを利用する。エージェントは，監視対象機器の中で動作するため，一般に，監視精度が高いとされている。しかし，エージェントをインストールしなければならず，また，バージョンアップに手間がかかることや監視対象機器の不具合を引き起こすなどの理由で，望まれないケースが増えている。そのため，本研究では，もともとエージェントが組み込まれているネットワーク機器については，エージェント方式で監視を行い，サーバなど，稼働する OS 別にエージェントを別途組み込まなければならない機器に対しては，エージェントレス方式を採用するものとし，エージェント方式とエージェントレス方式の混在による監視を行う。

図 3-12 に示すように，エージェント方式では，監視対象機器に状態変化が検出された段階で，障害解析エンジンのイベント受信部(監視サーバ相当)にイベントが通知される。エージェントレス方式では，障害解析エンジンのポーリング部が一定時間毎に監視対象機器へアクセスして，状態変化がないかどうか確認する。取得した状態が，前

回取得した状態と一致しなければ，変化したものとして，イベントを生成する。イベントを受信した場合も，生成した場合も，イベント受付部を経由し，イベント書込部に渡され，ルールメモリにイベントが書き込まれる。

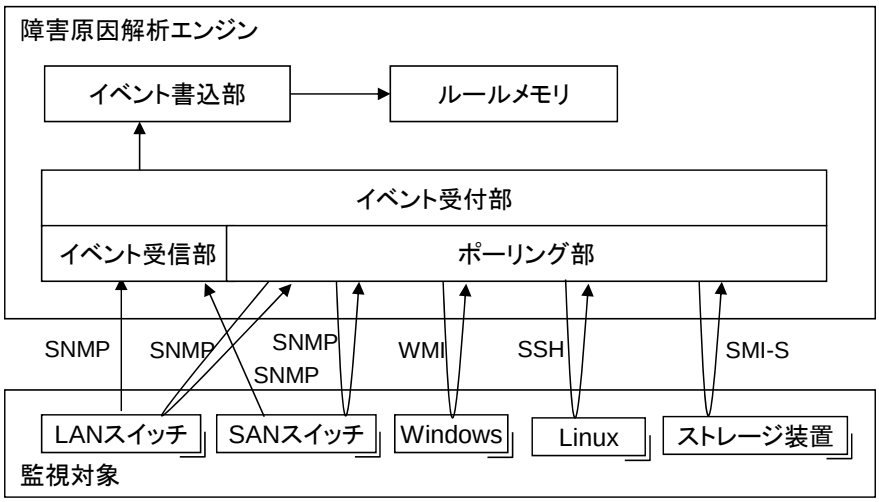


図 3-12 機器の状態の取得方法

3.6 ルールメモリに基づく解析方式

3.6.1 解析方式の要件

実際の情報システムでは，通信経路上のネットワークスイッチが障害になる場合もあり，障害イベントが RCA システムに通知されないことがある。また，ストレージで障害が発生し，サーバにボリュームを正しく提供できない事態になっても，サーバでこのボリュームにアクセスするまで，サーバからのアクセスエラーが発生しないなど，障害イベントの RCA システムへの通知が遅延することもある。さらに，情報システムからは，解析ルールという観点とは無関係に様々な障害イベントが頻繁に通知されるため，どのイベントが，障害連鎖の最初のイベントかを判定することは難しい。また，異常と正常を繰り返すような障害パターンもある。このような悪条件に対しても，適切な解析結果を得るための解析方式の要件は以下となる。

(1) 障害イベントがすべて揃わなくても，最も確からしい解析結果を導けること。
(2) 障害連鎖の最初のイベントを特定しなくても，解析結果を導けること。
(3) 対象機器の状態が異常と正常を繰り返す場合も解析の条件として扱えること。

要件(1)は, 完全な結論を得るために必要な条件のすべてが揃わない状況であっても, 揃っているいくつかの条件から，可能な限り確からしい結果を導けなければならないという要件である。この要件を満たすために，本研究では，条件に対する成立具合を計算して，最も成立率の高い結論を結果として導くようにする。

要件(2)は，どの障害イベントが一連の障害伝播の最初のイベントかを特定できなくても，解析結果を導けなければならないという要件である。従来の障害解析方式のほとんどは，最初のイベントから一定期間内に受信したイベントを用いて解析を行う。しかし，絶えず障害イベントが報告されるような大規模な IT 環境では，どのイベントが最初のイベントかを特定することは難しい。この要件を満たすために，障害イベントを受信するたびに，確信度を計算し，その時々で，例えば 70%以上の確信度であれば，結論とするようにする。

要件(3)は，IT 機器によっては，異常と正常を繰り返すものもあり，繰り返していることを，障害原因を解析するための条件として扱えるようにする。この要件を満たすために, ある一定回数, 異常と正常を繰り返すことを検出できるようにするとともに, 障害解析ルールの条件に設定できるようにする。

以降，要件(1)(2)(3)を満たすための方式について，それぞれ詳しく説明する。

3.6.2 結論に対する確信度の計算

要件(1)を満たすために, 解析手順を次のようにする。障害イベントを受信したとき，対応する条件オブジェクトの状態を変更し，次に，ルールオブジェクトに関連を持つ条件オブジェクト数に対する発火状態の条件オブジェクト数の割合を確信度として式(1)を用いて計算し，確信度の値の大きいものを障害原因として結論付ける。3.4.3 項で説明した通り，図 3-10 において，イベント A を受信したときには，条件オブジェクト「条件要素 A」を発火状態にする。次に，ルールオブジェクトへの結線を辿って，接

続されているルールオブジェクト「ルール 1」と「ルール 2」に注目する。次に「ルール 1」に接続されたすべての条件オブジェクトの発火状態を調べる。対応する条件オブジェクトは「条件要素 A」と「条件要素 B」，「条件要素 C」であるが，イベント A のみ受信した段階では，「条件要素 A」のみが発火状態である。「ルール 1」に結線された条件オブジェクトの数は 3 であり，そのうち発火状態の条件オブジェクトの数は 1 であるので，式(1)により確信度を計算すると，1/3 であり，0.3 となる。「ルール 2」についても同様に確信度を計算すると，1/2 であり，0.5 となる。したがって，イベント A を受信した段階では，ルール 2 の方が，より確からしいということになる。続いてイベント B とイベント C を受信したときには，同様に確信度を計算し，ルール 1 の確信度は，3/3 = 1.0 となり，ルール 1 に関連付けられている結論オブジェクトの内容，すなわち，結論①が最も確からしいと判定できる。

$$\text{確信度} = \frac{\text{成立した条件オブジェクト数}}{\text{条件オブジェクト数}} \quad \cdots\cdots\text{式（1）}$$

3.6.3 障害イベントの生存期間と発火状態のキャンセル処理

要件(2)を満たすために，障害イベントを受信するたびに，3.6.2 項に示した方法で確信度を計算する。解析ルールの条件に指定されている障害イベントの受信数が増えていくことで，確信度が 1.0 に近づいていくため，障害連鎖の最初のイベントがどのイベントだったのかを知る必要がなく，結論を導くことができる。

反面，この方式では，長期にわたってイベントを受信し続けると，多くのルールオブジェクトの確信度が 1.0 になってしまう。そこで，一定期間経過後に，障害イベントを解析の対象外にする処理を行う。つまり，各障害イベントに生存期間を定め，一定期間経過後に発火状態を取り消すとともに，確信度の計算をやり直す。こうすることで，イベントを一定期間収集してから解析処理を行う従来方式と異なり，ストリーム処理的に障害原因解析を行うことができ，イベントを受信するか，生存期間が終了するかの，その時々での最も確からしい解析結果を利用者に提示することが可能となる。また，解析にかかる計算量の観点では，一度投入したイベントを再投入するとい

った無駄を排除できる。

イベントを受信するか，生存期間が終了することによる各ルールの確信度の変化を図 3-13 に示す。イベント A を受信したとき，ルール 1 の確信度は 1/3，ルール 2 は 1/2 である。イベント B を受信すると，ルール 1 の確信度は 2/3，ルール 3 は 1/2 となる。イベント C を受信すると，ルール 1 の確信度は 3/3 となる。その後，イベント A が生存期間を終了し，ルール 1 の確信度が 2/3，ルール 2 は 0 となる。さらに，イベント B が終了し，ルール 1 が 1/3，ルール 3 が 0 となる。イベント C が終了すると，ルール 1 が 0 となる。

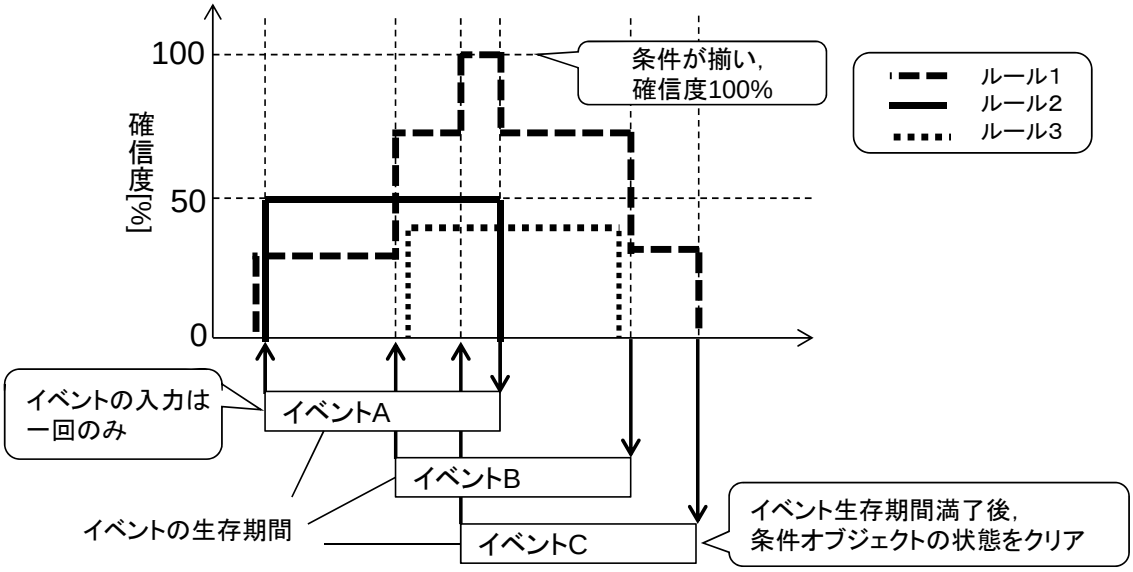


図 3-13 障害イベントの入力と確信度の変化

3.6.4 ポーリング間隔と障害イベントの生存期間の関係

ここで，エージェントレス方式で監視対象機器にアクセスして状態変化を検出するポーリングの間隔とイベントの生存期間の関係について説明する。3.5 節で説明した通り，エージェント方式で検出した状態異常に対応するイベントは随時受信するものとし，一方で，ポーリングによる状態監視も並行して行う。

障害イベントの生存期間は，ポーリング間隔の 2 倍の期間とする必要がある。以下，この理由を説明する。ポーリングとイベントの生存期間の関係を図 3-14 に示す。ポー

リングは監視対象機器に 1 台ずつ問い合わせを行う形になるため, 1 台目と i 台目では, 状態確認にタイムラグが生じる。そのため，例えば，N 回目のポーリングで，2 台目の機器の状態確認をしている最中に，既に確認が終わっている 1 台目に障害が発生した場合，N 回目のポーリングでは，1 台目の状態異常は検出されない。1 台目の状態異常が検出されるのは，次のターンである N+1 回目のポーリングである。そのため，イベントの生存期間は最低でも 15 分必要となる。しかし，これは，ポーリングの順序がいつも同じである場合であり，実際には，N 回目のポーリングで，1 台目だった機器へのポーリングが，N+1 回目では最後になることもある。

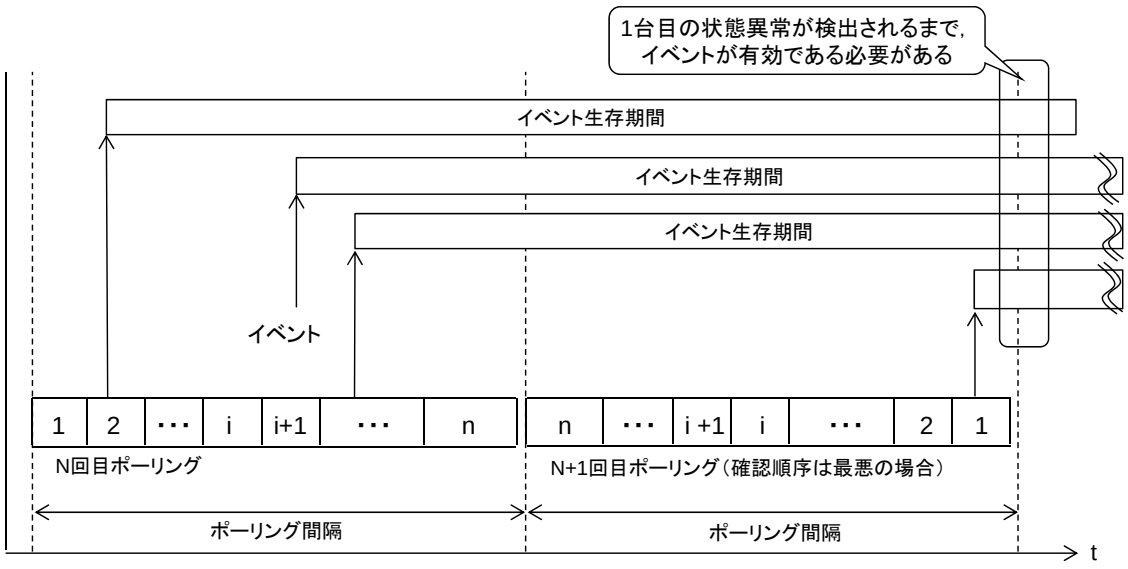


図 3-14 ポーリング間隔とイベントの生存期間の関係

実際には，これら 2 つの障害イベントは，ほぼ同時に発生した障害であるから，これらの障害イベントの両方を使って障害解析を行う必要がある。しかしながら，上記の説明の通り，最悪の場合を考慮して，障害イベントの生存期間はポーリング間隔の 2 倍の時間としなければならない。

ポーリングは, 監視サーバから多数の監視対象に問い合わせを行う必要があるため, 1 台の監視サーバが監視できる対象機器数には限界がある。ポーリング間隔を短くすれば，状態を細かくチェックできるため，状態異常の検出時間も短くなるが，反面，1

台の監視サーバで監視できる対象機器数が少なくなる。ポーリング監視間隔を長くすれば，一台の監視サーバで監視できる対象機器数を多くすることができるが，状態の監視は粗くなり，状態異常の検出時間も長くなる。ポーリング間隔を短くした場合，監視対象での状態取得のためのプログラムが頻繁に実行されるため，監視対象への管理負荷増大や，監視サーバと監視対象機器の間のネットワークの負荷増大というデメリットがある。ポーリング間隔は，これらの関係を考慮して，バランスよく決めなければならない。

3.6.5 異常と正常を繰り返す場合の検出方法とルール記述方法

要件(3)を満たすために，ある一定回数，異常と正常を繰り返すことを検出できるようにするとともに，障害解析ルールの条件に設定できるようにする。

ここで，生存期間内に，監視対象の状態が異常と正常を繰り返す場合のイベントの生成の方法と確信度計算の考え方について図 3-15 により説明する。図 3-15 の左側は，監視対象の異常状態と正常状態の遷移の様子を時間軸で示した図である。図中，p1～p7 は，ポーリングのタイミングである。p1 のポーリングの結果，監視対象が異常状態にあることを検出し, 次の p1 のポーリングでは, 正常状態に戻ったことを表している。このような場合は，ルールメモリの条件要素オブジェクトのカウンター属性の値をカウントアップする。図では，p3 のポーリングで再び異常状態にあることが検知されたので，条件要素 A のカウンターを 2 にインクリメントしている。このインクリメント処理は，イベントの生存期間でのみ実行し，生存期間の終了のタイミングで 0 にクリアする。

一方，図 3-15 の右側は，ルールメモリによる確信度計算の様子を示した図である。確信度計算では，この条件要素のカウンターの値の大きさを条件として定義する。例えば，カウンターの値が x 以上であれば，その条件要素オブジェクトの状態を発火状態にする, というルールを記述する。IF-THEN 形式で表現すると, 例えば,「IF A freq(3) and B THEN A. Flooding」のようになる。

このように，監視対象機器の状態が正常と異常を繰り返す場合には，汎用解析ルールの条件部に「freq(n)」属性を記述することで，障害原因解析のルールとして取り扱うことができるようになる。

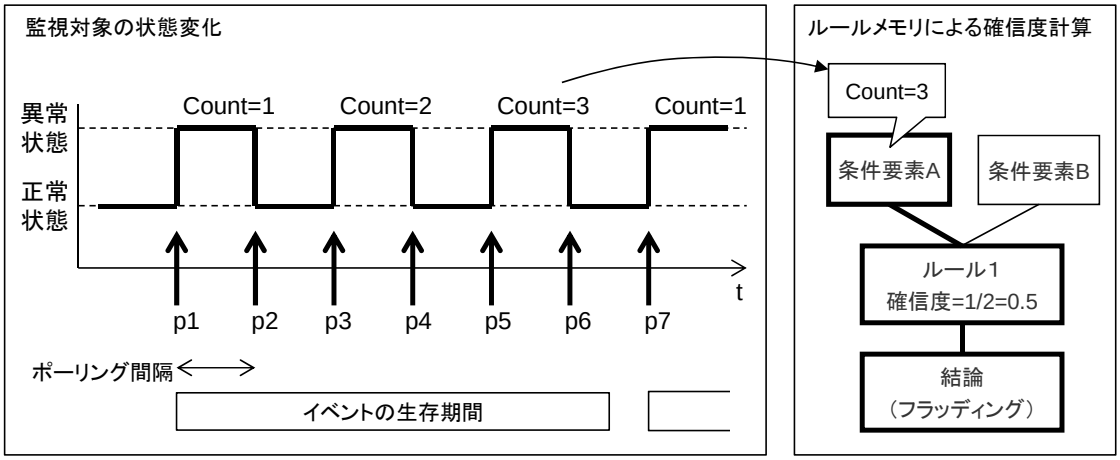


図 3-15 監視対象の状態変化とルールメモリによる確信度計算

3.7 ユーザインタフェース

サーバ装置，ネットワーク機器，ストレージ装置で構成される情報システムに対して，障害原因解析結果を表示するためのユーザインタフェースの例を図 3-16 に示す。

画面の上部には，対象となる IT 環境から自動発見した IT 機器とそれらのつながりを示すトポロジーが表示される。障害が発生した機器を表すアイコンにはステータス異常を示す「×」のマークが表示される。障害原因解析の結果，すなわち，障害原因と推定された機器を表すアイコンには雷のマークが表示される。図 3-16 の例では，Computer カラムの「DBServ01」と「WebServ03」というサーバが異常状態であり，Storage カラムの「Storage02」というストレージ装置が障害原因として特定されたことを示している。また，図の下部には，対象となる各 IT 機器から通知された障害イベントの一覧が表示される。

3.8 提案方式の有効性に関する評価と考察

本節では，本研究で提案したルールメモリと解析処理方式が，従来方式として実用化されているルールマトリクス方式と比較して，少ない計算量で障害原因解析処理が

図 3-16 ユーザインタフェース

行えることを示す。検証環境は，「CPU: Core2 Duo 2.4GHz，Memory: 2GByte，Java ヒープメモリ 256Byte」であり，監視対象システムは，「ストレージが提供するボリュームをサーバから iSCSI 接続する形態」とした。

3.8.1 障害原因解析のためのイベント処理時間の評価

従来方式の 1 つであるルールマトリクス方式と提案方式について，解析処理にかかる時間の比較を行う。ここで，ルールマトリクス方式について説明する。図 3-17 に示すように，縦軸に「症状」，横軸に「原因」として表現したマトリクスにより解析を行う方式である。「症状」として障害イベント群を入力し，行から列へと辿り，「X」の多いものを「原因」として出力する。内部状態を保持することはできないため，過去一定期間の障害イベント群を入力する必要がある。そのため，イベント到着間隔が短い場合，同じイベントを重複解析することになり，計算量が多くなる。

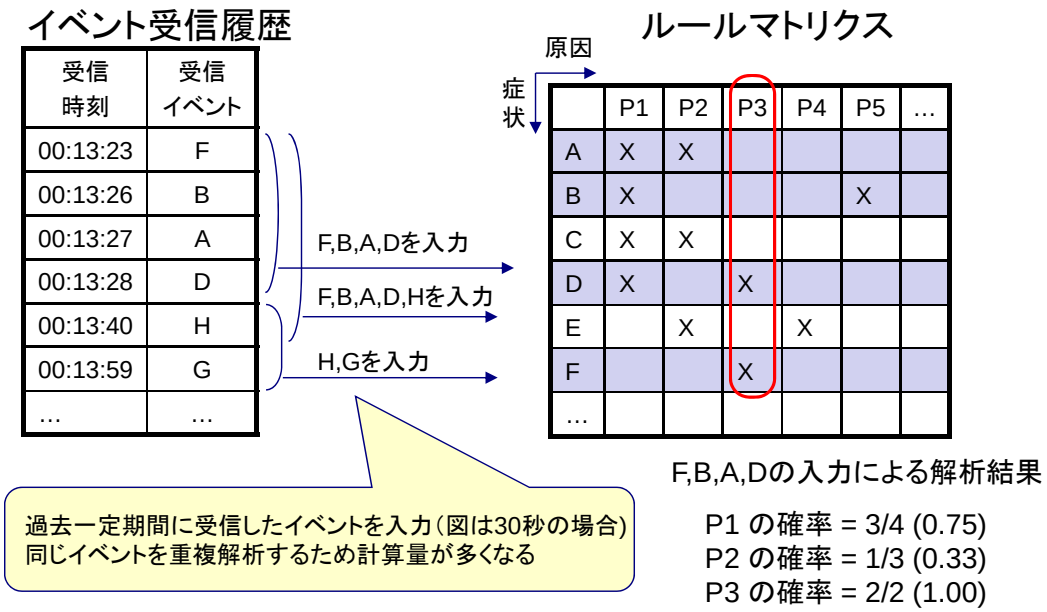


図 3-17 ルールマトリクス方式の概要

はじめに，ルール数を 100 から 500 まで変化させたときの，1 つの障害イベントに対する解析処理時間の変化を比較する。結果を図 3-18 に示す。従来方式では，縦軸から入力された「症状」に対する交点を探す処理が必要となるため，解析のための計算量はマトリクスの列数(ルール数)に比例して大きくなる。一方，提案方式では，あらかじめルール展開を行い，解析ルールをオブジェクトの結線関係を含めて構築済みとしているため，入力された障害イベントに対して次に辿るべきルールオブジェクトが直接的に決定される。そのため，計算量は極めて小さい。実測の結果，ルール数に比例せず，一定で，約 400 ミリ秒となった。

また，図 3-18 によれば，解析ルール数が 200 を超える場合に，従来方式に比べ，提案方式の解析時間が短くなる。そこで，表 3-3 に示した汎用解析ルールと 3. 2. 1 項に示した従業員 500 人の研究所のモデルに内在するトポロジー数により，ルールメモリ内に構築される解析ルール数を計算し，ルール数がどの程度になるか確認する。計算結果を表 3-4 に示す。

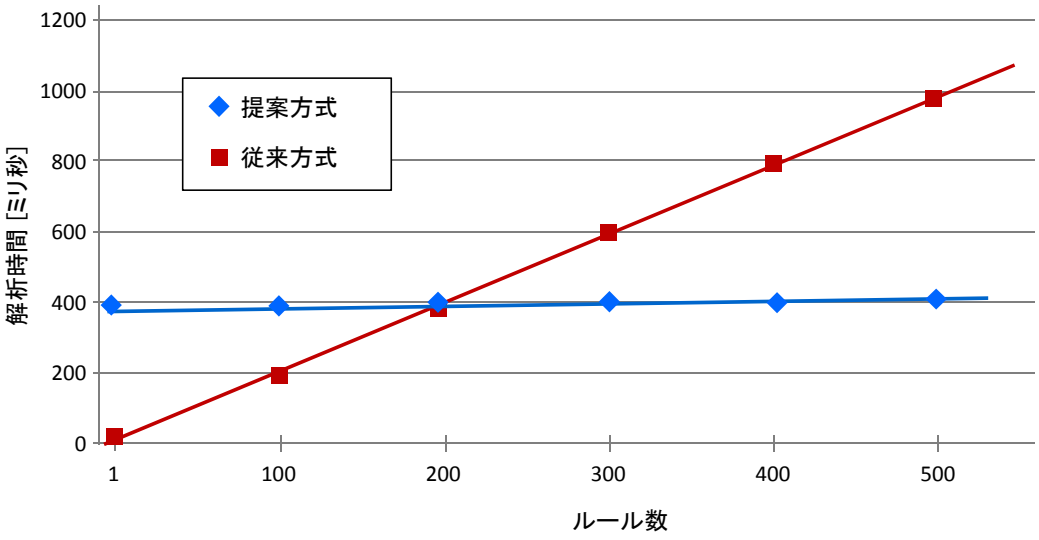


図 3-18 1 つの障害イベントの解析処理時間の比較

表 3-4 ルールメモリ内の解析ルール数

トポロジーの種類	汎用解析ルール数	トポロジー数	ルールメモリ内の解析ルール数
	(a)	(b)	(c)=(a)×(b)
IP-Network トポロジー	6	3	18
IP-SAN トポロジー	71	2	142
FC-SAN トポロジー	63	2	126
File-Share トポロジー	10	2	20
VM-Host トポロジー	27	3	81
Server 単体	97	500	48,500
IP-Switch 単体	9	20	180
FC-Switch 単体	6	3	18
Storage 単体	39	5	195
RCA サーバとの通信	9	3	27
合計	337	543	49,307

IP-Network トポロジーは，社内データセンタ内のサーバで稼動しているデスクトップ環境に，各従業員は自席 PC からリモートデスクトッププロトコルで接続している形態である。従業員数は 500 であるため，デスクトップ環境は 500，自席 PC も 500，

デスクトップ環境と自席 PC を接続する IP スイッチの段数を 3 段とすると，解析ルールの条件要素数は 500 + 500 + 1 の合計 1001 個となる。これは，IP スイッチが多段に接続されている場合に，解析ルールとしては，IP スイッチは 1 つであるとして，ルール化するためである。1 段目の IP スイッチが障害を起こすか，または，2 段目の IP スイッチが障害を起こすか，または，3 段目の IP スイッチが障害を起こすか，という，OR の解析ルールを 3 つ作成するためである。そのため，IP-Network トポロジー数は 3 と数える。IP-Network トポロジー向けの汎用解析ルール数は 6 であるので，ルールメモリ内に構築される解析ルール数は 18 となる。

IP-SAN トポロジーは，各デスクトップ環境に対して，ストレージ装置から iSCSI ドライブを割り当てるトポロジーである。デスクトップ環境とストレージ装置は同じデータセンタに存在するため，IP スイッチの段数は 2 であるとする。IP-Network トポロジーの場合と同様に計算すると，IP-SAN トポロジー向けの汎用解析ルール数は 71 であるため，ルールメモリ内に構築される解析ルール数は 142 と計算できる。同様に，FC-SAN トポロジー，File-Share トポロジーについても，スイッチ段数を 2 とすると，ルールメモリ内に構築される解析ルール数は，それぞれ，126，及び，20 と計算できる。VM-Host トポロジーについては，仮想サーバ基盤ソフトウェア(ハイパーバイザと呼ばれる)によって動作させる仮想サーバの数によって，解析ルール数が異なる。1 つのハイパーバイザで動作させる仮想サーバの数を 3 とする。この場合，ルールメモリ内に構築される解析ルール数は，81 となる。

その他，装置単体に対する解析ルール数も同様に計算すると，ルールメモリ内の解析ルール数は合計で 49,307 となる。

以上，従業員 500 人の研究所のモデルで示した環境においては，1 つの障害イベントを処理する時間で比較した場合，提案方式では 400 ミリ秒，従来方式では 10 秒程度かかることになり，提案方式の方が優位であると言える。

3.8.2 累積解析処理時間に関する評価

3.8.1 項で比較対象として挙げたルールマトリクス方式に限らず，従来の障害原因解析方式では，最初のイベント受信からタイマーをスタートさせ，一定の時間内，イベントを収集し，収集したイベント群を用いて解析処理を実行する。そのため，一定時

間が経過しないと解析結果が得られないという問題がある。

そのため，ここでは，一定時間内に受信したイベントについても，イベント受信のたびに，解析処理を実行した場合に，解析結果が出るまでの累積解析時間の比較を行うこととする。これら従来の方式では，障害イベントによる発火状態を内部状態として保持することはしないため，2 つ目以降の障害イベントを解析処理する際に，過去の障害イベントも併せて入力することとする。その場合，4 つ目の障害イベントを受信したときには，「Σi(i=1..4)×(1 回の処理時間)」の解析時間がかかることになる。一方，提案方式では，過去の障害イベントについてルールメモリ内に一定時間保持するため，2 つ目以降の障害イベントを処理する際に，過去の障害イベントを入力する必要はない。そのため，単純に「i×(1 回の処理時間)」で解析が可能である。

3.8.1 項と同様，ルールマトリクス形式と比較を行う。ルール数を 300 に固定して，4 つの障害イベントの解析で結論が導ける解析ルール，すなわち，条件要素数が 4 つのルールについて，障害イベントを受信するたびに解析を行う場合の累積解析時間を比較する。結果を図 3-19 に示す。

同じイベントを複数回入力するため，累積の解析時間はイベント数が多くなれば，指数関数的に増加する。一方，提案方式では，単純に「i×(1 回の処理時間)」で解析が可能であるため，累積の解析時間が指数関数的に増加することはない。

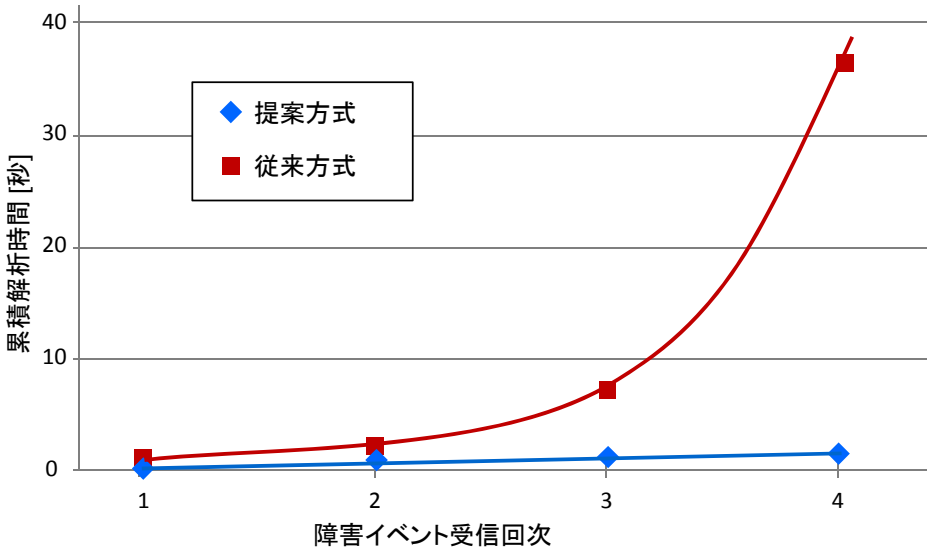


図 3-19 累積解析時間の比較

3.8.3 解析ルールの事前展開に関する考察

提案方式は, 汎用解析ルールとトポロジーにより, 事前にルールメモリを構築する。プログラム実行方式に例えれば，ソースプログラムを実行基盤である OS に合わせてコンパイルする方式と言える。障害イベントを受信し，解析処理を行う時点で，ルールメモリには，対象のシステム環境のトポロジーが織り込まれているため，解析処理は高速に行える。反面，ルール展開時間に時間がかかること，ルールメモリのサイズが大きくなること，という 2 つのデメリットがある。

図 3-20 は，ノード数に対するルール展開時間の変化を示したグラフである。ルール展開時間はノード数に比例して増加することが分かる。300 ノードの環境の場合，ルール展開にかかる時間は約 10 分であり，この間に受信した障害イベントはキューに入れられる。つまり展開処理が完了するまで解析を始めることができない。ルール展開処理は，汎用解析ルールが追加されるか，対象環境のトポロジーに変化があった場合に実行される。最近では仮想サーバ技術が普及しており，仮想マシンのライブマイグレーション(ハイパーバイザを跨って仮想マシンを無停止で移動させる技術)も利用されるようになってきている。仮想マシンがマイグレーションするたびにトポロジーは変わることになる。そのたびに約 10 分間，障害原因解析が無効の状態になるため，このようなマイグレーションを多用する環境では，提案方式は不十分である。今後，トポロジーで変化した部分のみをルール展開しなおす方式を検討していく。

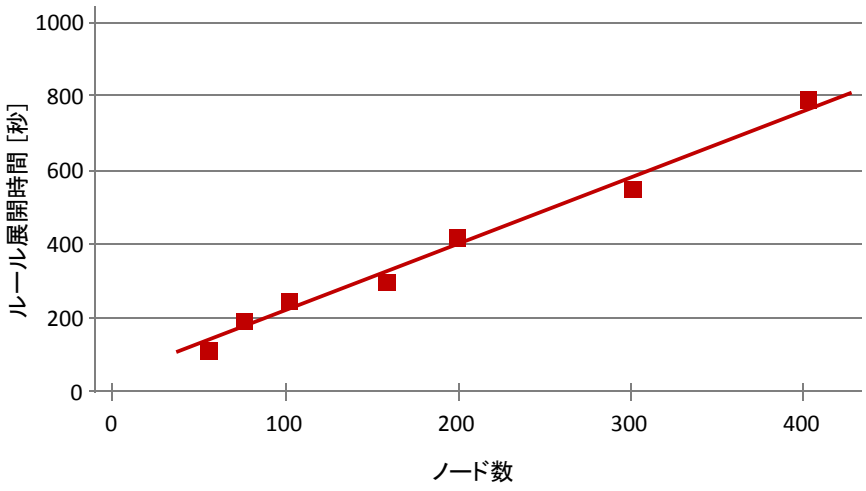


図 3-20 ノード数に対するルール展開時間の変化

また，図 3-21 は，ノード数に対するメモリ消費量の変化を示したグラフである。ルール展開処理の最中に消費するメモリの最大値と，ルール展開処理完了後のルールメモリの常駐サイズについて，それぞれの変化を示している。図 3-21 によれば，両者共に，ノード数に比例して増加することが分かる。また，ルール展開処理には，ルールメモリ常駐サイズの約 2 倍が必要となる。ノード数がさらに多くなった場合に，ルールメモリ常駐用の空きメモリ容量は十分にあるにもかかわらず，ルール展開処理が実行できないという事態にならないように，ルール展開処理の省メモリ化を図っていく必要がある。

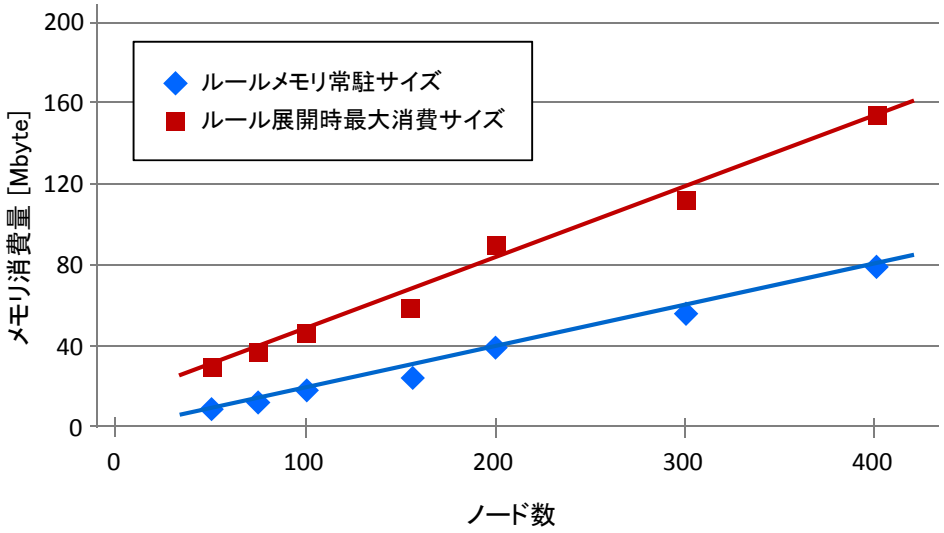


図 3-21 ノード数に対するメモリ消費量の変化

3.9 結言

本章では，情報システムの障害原因解析のための解析ルールのデータ形式，及び，解析方式について述べた。解析ルールの条件式の要素を，ルール間で共通化してオブジェクト構造化して，条件要素とルール間に直接リンクを張ったルールメモリのデータ形式により，障害イベント数に比例する解析処理が行えることを示した。本方式に

よれば，解析の対象となる情報システムの規模が大きくなっても，処理速度に影響しない。また，イベントを受信するたびに，確信度の計算を行うため，イベント受信のたびに，解析の途中経過を利用者に提示できる。

今後の課題は，ルールメモリの構築時間短縮である。提案したルールメモリのデータ形式によりイベント発火状態を保持することで，イベント受信時の処理を高速化できる一方で，汎用解析ルールと対象システムのトポロジーからルールメモリを構築する処理に多くの時間とメモリを要するという問題がある。近年のサーバ仮想化技術の普及により，情報システムのトポロジーが変わる頻度が増大している。トポロジーが変わればルールメモリもそれに合わせて構築しなおさなければならないため，イベント受信時の処理の高速性だけでなく，ルールメモリの再構成速度についても改善が必要となる。今後，トポロジーが変更された部分に対応するルールメモリの該当箇所のみを再構築する部分展開方式について検討を行う。

なお，本章で述べた障害原因解析方式は，（株）日立製作所のソフトウェア製品 (製品名 : Hitachi IT Operations Analyzer)[77][78]に組み込まれ, 2009 年 4 月にリリースされ，世界 10 ヶ国以上で利用されている。日本国内では，2010 年 10 月にリリースされており，2011 年現在もその機能拡張や性能向上のための研究を進めている。

第4章
情報システムの運用自動化に向けたポリシー実行スケジューリング方式

4.1 緒言

本章では，業務システムの運用自動化に向けたポリシーを安全に実行させるためのポリシーの記述方法，及び，ポリシーの実行スケジューリング方式について述べる。

WWW による一般消費者向けサービスが普及している。このようなサービスは，利用者数の予測が困難であり，想定外のアクセス集中による性能低下が起きやすい。このような状況になった場合に，Web サーバを増設するか，新規ログインを抑止するなど，状況に応じた対処を自動実行させる，ポリシーベース運用自動化技術に関する研究が各所で進められている。しかし，ポリシーベース運用自動化では，対象システムの状況によって，設定したポリシーのうち複数が同時に動作を開始してしまうことがある。しかし，もしもそれらが，同一の IT リソースに対して構成変更を行うポリシーであった場合，構成変更矛盾を起こしてしまうなど，ポリシー設計者の意図に反した結果を引き起こす。そのため，ポリシー制御システムは，あるポリシーが実行中，つまり，起動されて対処操作が完了するまでの間に，別のポリシーが実行可能状態になったとしても，そのポリシーと実行中のポリシーの同時実行可否を判定して，そのポリシーのアクションを実行するか，待ち状態にするように制御しなければならない。

複数のポリシーの同時実行可否を指定するもっとも単純な方式は，同時に実行すべきでないポリシーの識別子を個々のポリシー記述の制約条件部に列挙する方式である。しかし，この方式では，既にいくつかのポリシーが設定された業務システムに対してポリシーの追加・削除を行う際に，他の設定済みポリシーの制約条件指定を見直して修正しなければならず，ポリシーの記述工数が膨大になるという問題がある。

そこで本研究では，対象となる情報システムの構成をモデル化した業務システムの

論理構成ツリーに対して，各ポリシーから同時実行条件を指定するポリシー記述形式と，同一の業務システムに対して複数のポリシーが同時に実行可能な状態になったとき，それぞれのポリシーの対象リソースの範囲の重なりを業務システムの論理構成ツリーの構造に基づいて判定し，同時に実行させるか，1 つずつ実行させるかを決定するポリシー実行スケジューリング方式を提案する。

以下，4.2 節で，ポリシー制御システムの基本動作とポリシー制御のシナリオ例について説明し, 4.3 節で, ポリシー実行制御機能の課題と解決方針について述べる。次に，4.4 節で，試作したポリシー制御システムの設計について説明する。4.5 節では，ポリシー実行制御機能を実際に動作させたときの性能の評価，及び，提案するポリシー制御システムにおけるポリシーの記述性に関する評価を行う。

4.2 ポリシー制御システムの概要

4.2.1 ポリシー制御システムの基本動作

ポリシー制御システムは，ポリシー記述の内容にしたがって，制御対象の情報システムの状態を監視し，状態が条件に一致した場合には，情報システムに対して構成変更などの対処操作を実行し，情報システムの稼動状態を維持する。

ポリシー記述には，対処実行の契機となる発火条件と，その条件を満たしたときに実行する対処操作を組にして記述する。発火条件には，情報システムが障害になった際に発行されるイベントや，過負荷になったと判定するための閾値条件を指定し，対処操作には, 構成変更などの運用管理オペレーション(アクションと呼ぶ)を実行するためのフローを記述する。ポリシー制御システムは，イベントの受信を契機に，設定されているポリシー群から，そのイベントが発火条件として指定されているポリシーを検索し，そのポリシーで指定されているアクションを自動的に実行する。

4.2.2 ポリシー制御のシナリオ例

ここでは，Web3 階層システムにポリシーを適用することを想定して，ポリシー制御のシナリオについて，以下，例を挙げて説明する。ここでの Web3 階層システムは，

Web/AP サーバがロードバランサーに接続され，スケールアウトによる負荷分散が可能であり，一方，DB サーバはスケールアウト不可能であるものとする。図 4-1 は，ここで説明する 3 つのシナリオを図示したものである。

(1) 障害時のサーバ切替え

Web3 階層システムを構成する各サーバの稼動状況を監視し, 異常が発生した際には, 現用系から待機系に切り替えるサーバ代替処理を行う。

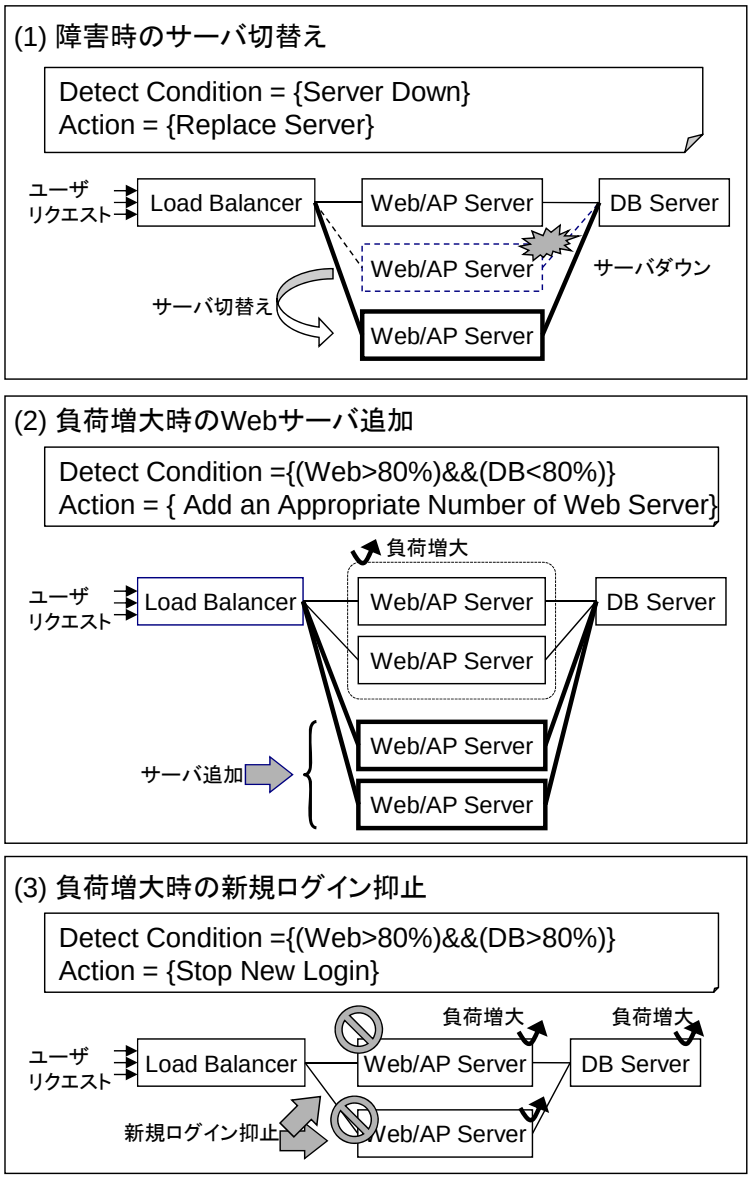


図 4-1 Web3 階層システムに対するポリシー制御のシナリオ

(2) 負荷増大時の Web サーバ追加

ロードバランサーにより複数の Web/AP サーバで負荷分散する形態では，Web/AP サーバ単体ではなく，Web 層としての性能値(スループット，応答時間など)を監視する。性能低下が検出された場合は，適切な数の Web/AP サーバを追加することで，性能維持を図る[95][96]。

(3) 負荷増大時の新規ログイン抑止

Web/AP サーバの負荷が高く，さらに DB サーバの負荷も高い場合は，Web/AP サーバをスケールアウトさせても DB サーバの負荷がさらに高まってしまうため，ユーザ数を増加させないための新規ログイン抑止操作を行う。

その他，サーバのディスク容量に応じてバックアップ処理やデータ削除処理を行うポリシー，ネットワークの負荷に応じて帯域調整を行うポリシー，サービス異常時にサービスのリスタートを行うポリシー，定期的にサーバのリブートを行うポリシー，ハイパーバイザの負荷に応じて仮想サーバを別のハイパーバイザにライブマイグレーションさせるポリシーなど,1 つの業務システムに対し複数のポリシーが設定される。

4.3 ポリシー実行制御機能の課題と解決方針

4.3.1 ポリシー実行制御機能の課題

運用現場において，管理対象となる情報システムに対するポリシーは，対象となる情報システムの稼働後に随時見直され，改善されていく。既に設定されているポリシーを修正する場合，新しくポリシーを追加する場合，不要になったポリシーを削除する場合などさまざまである。ポリシーを設定する運用管理者も組織における配置換えなどで，同一であるとは限らない。そのため，既に設定されているポリシーの制御対象や他機器への影響をすべて把握するのは困難であり，そのような状況下で，ポリシーの見直しを行わなければならない。そのため，1 つのポリシーとして，完結した形でポリシーを設計しがちであり，その場合，1 つずつシーケンシャルに実行させるの

であれば有効に働くが，同時に複数が実行されると，以下(1)に示す通り，ポリシー設計者の意図に反する結果となってしまう。

また，1 つのポリシーとして，完結した形でポリシーを設計したつもりでも，ポリシーの起動のメカニズムをよく理解せずに設計すると，以下(2)に示す通り，思わぬ結果を生むことがある。

(1) 同じ対象への競合操作

Web/AP 層の負荷増大時に，稼働中の Web/AP サーバ台数に基づいて，負荷が適正になるようなサーバ台数を求めて，Web/AP サーバを追加するポリシーを考えた場合，先に起動されたポリシーによるサーバ追加処理の完了を待たずに，同じポリシーが実行されると，後発のポリシーは，誤った「Web/AP サーバ稼動台数」に基づいた適正台数計算を行ってしまう。

(2) 同じポリシーの重複実行

ロードバランサーによってユーザアクセスを複数の Web/AP サーバに振り分ける Web3 階層システムでは，アクセス数の増大によって Web/AP サーバの負荷はどれも同様に高くなる。Web/AP サーバ毎に CPU 負荷を監視し，負荷増大に対して Web/AP サーバを追加するアクションを実行するポリシーを適用した場合，高負荷を示すイベントが複数の Web/AP サーバから重複して報告されることになる。その結果，1 台の Web/AP サーバの追加で Web/AP 層としての負荷低減が可能な場合であっても，イベントの数だけ Web/AP サーバが追加されてしまう。

よく知られているポリシー記述言語として，DMTF の CIM-SPL がある。CIM-SPL によれば，ポリシーをグループ化することができ，さらに，グループを入れ子にして定義することも可能である。各グループにはグループ内のポリシーの実行方針として，「Execute_All_Applicable」と「Execute_First_Applicable」のどちらかを指定する。前者はグループ内の実行可能状態のポリシーを全て実行させることを意味し，後者は最初に実行可能状態になったポリシーを 1 つだけ実行させることを意味する。すべてを実行させるか，1 つだけ実行させるかの指定であり，ここで議論する，同時に実行させるか，1 つずつ順番に実行させるかの指定はできない。他のポリシー記述，例えば，PCIM や，PCIM を拡張した ACPL (Autonomic Computing Policy Language)についても，

2 つのポリシー間の同時実行可否を指定することはできない。また，複数のポリシーが同時に実行可能となった場合に，ポリシーの実行時間やポリシーの優先度などを考慮して，実行順序をスケジューリングする研究として文献[97]がある。しかし，複数ポリシーの同時実行による実行対象となるリソースの競合を考慮に入れた同時実行制御については取り扱われていない。

イベントと条件とアクションの組みで表現する ECA ルール(Event Condition Action Rule)の並行実行制御に関しては，古くより，Active Database Management System におけるトランザクション処理の並列実行制御[64][65]の分野で議論されてきた。しかし，これらの研究は，ルール作成者が制御すべき条件をすべて把握していることを前提としてルールの記述能力を高める研究であり，運用現場において複数の別々の運用管理者が 1 つの業務システムに対して容易に安全なポリシーを設定するような使い方は想定していない。

複数のポリシーの同時実行条件を指定する最も単純な方式としては，同時に実行すべきでないポリシーの識別子を個々のポリシー記述の制約条件部に列挙する方式である。例えば，図 4-2 は，業務システム ABC にポリシーA とポリシーB，ポリシーC が設定され，ポリシーA とポリシーB は同時実行不可，ポリシーA とポリシーC は同時実行可能，ポリシーB とポリシーC は同時実行可能な場合に，ポリシーA には，同時実行不可のポリシー識別子「ポリシーB」を，ポリシーB には，同時実行不可のポリシー識別子「ポリシーA」を指定する様子を示した図である。

しかし，この方式では，ポリシーB，及び，ポリシーC と同時実行不可能なポリシーX を新たに追加するときに，既に設定されているポリシーA とポリシーB の記述を変更しなければならない。設定済みのポリシーからいくつかのポリシーを削除する場合も同様である。このことは，ポリシーの記述ミスを誘発するだけではなく，ポリシー追加・削除作業の工数を引き上げる原因となる。したがって，本研究では，操作対象のリソース競合を考慮した複数ポリシーの同時実行制御方式と，ポリシーを追加・削除をより少ない工数で行えるポリシー記述形式の実現を課題とする。

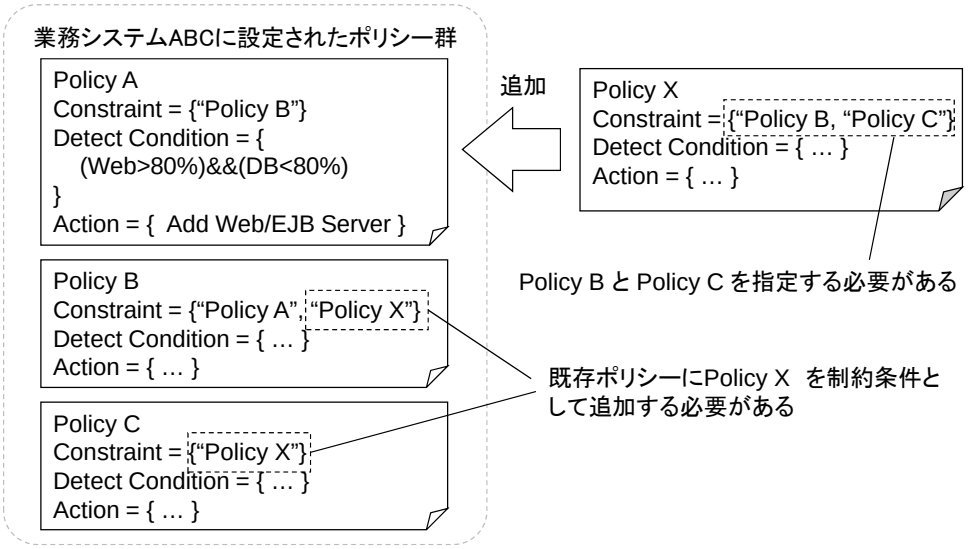


図 4-2 従来方式による新規ポリシー追加時の同時実行条件の調整

4.3.2 課題を解決するポリシー実行制御方式

4.3.1 項で述べた課題を解決するために，本研究では，図 4-3 に示すような業務システムの論理構成ツリーを導入し，同時に対処操作を行うべきでない管理対象リソースの範囲をポリシー記述から指定する方式を提案する。業務システムの論理構成ツリーとは，業務システムを論理的な役割に基づいてブレークダウンしてツリーとして表現した論理オブジェクトツリーである。ルートノードとして業務システム全体を表す論理オブジェクトを配置し，下位オブジェクトは上位オブジェクトの構成要素であるように，つまり，オブジェクト指向でいう「has-a」関係となるようにツリーを構成する。

例として, Web3 階層モデルで構成されたオンライン購買システムの業務システムの論理構成ツリーを図 4-3 に示す。図 4-3 において，業務システムの論理構成ツリーのルートとして，この業務システム全体を示すオンライン購買システムのオブジェクトを配置し，これを構成する要素として，フロントエンド層とバックエンド層を示すオブジェクトを配置し，関連付ける。さらにフロントエンド層には，これを構成する要素として，ユーザからのリクエストを Web/AP サーバに振り分けるロードバランサーを示すオブジェクトと，ユーザリクエストを処理する Web/AP 層を示すオブジェクトを配置し，関連付ける。Web/AP 層は，スケールアウト構成が可能な Web/AP サーバ群を表すオブジェクトで構成していることを示している。同様に，バックエンド層は，DB サーバを表すオブジェクトで構成している。

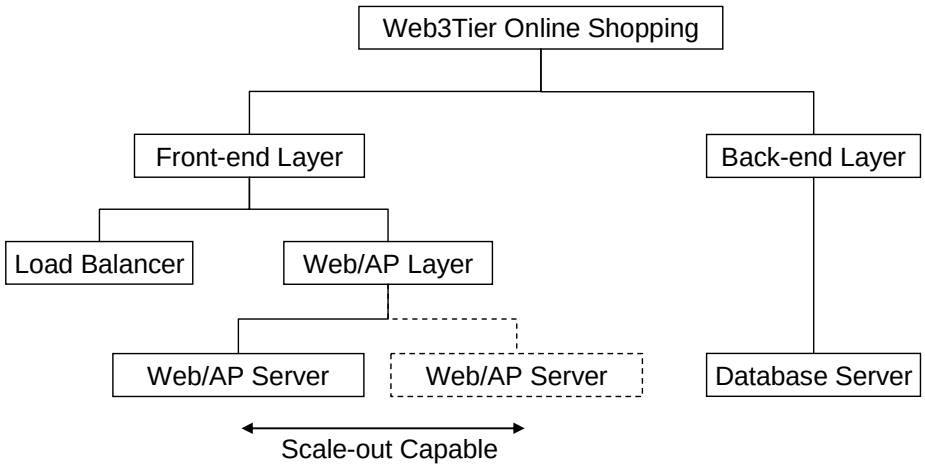


図 4-3 業務システムの論理構成ツリーの例

ポリシー記述には，対処実行の契機となる発火条件と，その条件を満たしたときに実行する対処操作を組にして記述する。さらに，ポリシーに記述された対処操作によって変更が及ぶ範囲を，当該ポリシーの適用先として，業務システムの論理構成ツリーのオブジェクト ID によって指定する。この業務システムの論理構成ツリーが示す，業務システムにおける論理構成の階層的なつながりを利用することで，ポリシーの対処操作の対象リソースを絞りこみ，同時実行可否の判定を行う。

具体的には，図 4-4 に示すように，業務システムの論理構成ツリーを用いて，ポリシーA で指定したポリシーの適用先である，フロントエンド層のオブジェクトを頂点として，ロードバランサーオブジェクト，Web/AP 層オブジェクト，Web/AP サーバオブジェクトから成る部分ツリーを，対処操作の対象リソースとして抜き出す。抜き出した対処操作の対象リソースに対して対処操作を実行中のポリシーが存在しなければ，同時実行が可能であると判定し，そうでない場合に，同時実行が不可能であるとして，片方のポリシーを待たせるなどの排他制御を行う。また，例えばフロントエンド層を適用先とするポリシーA とバックエンド層を適用先とするポリシーB は，ツリー上で独立した部分ツリーを適用先とするポリシーであるため，並列実行が可能となる。

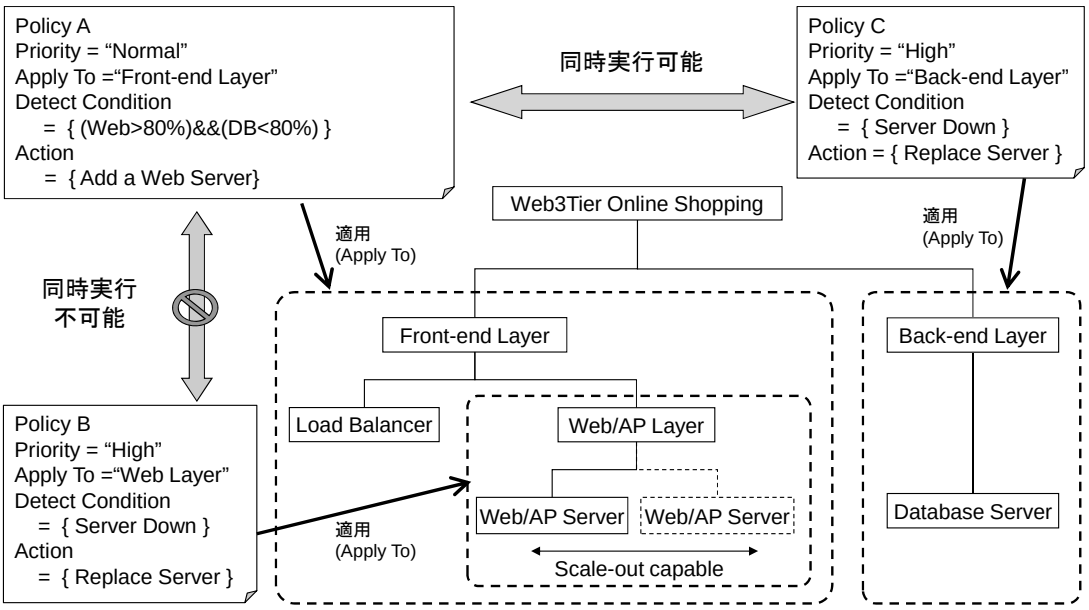


図 4-4 業務システムの論理構成ツリーへのアクション対象リソースの指定によるポリシーの同時実行制御

業務システムの論理構成ツリーの作成は，前述の通り，ルートノードとして業務システム全体を表す論理オブジェクトを配置し，下位オブジェクトは上位オブジェクトの構成要素であるようにツリーを構成する。ポリシー記述には，ポリシーの操作対象のオブジェクト ID を記載する必要があるため，業務システムの論理構成ツリーにその対象オブジェクトが定義されていない場合は，ポリシーの追加に併せて，論理オブジェクトも追加定義する。多くの場合，定義済みのオブジェクトをさらにブレークダウンする形でツリーを拡張することで，設定済みのポリシーの変更は不要にできる。例えば，「バックエンド層」をブレークダウンし，「集計バッチサーバ」を追加することで，「DB サーバ」に設定したポリシーと同時実行可能な「集計バッチサーバの設定変更を行うポリシー」を追加できる。同時実行を許さない場合は，「集計バッチサーバの設定変更を行うポリシー」の適用先をバックエンド層にすればよい。

このように，業務システムの論理構成ツリーを用いることで，ポリシー記述をシンプルに記述できるだけでなく，業務システムの論理構成ツリーの再定義とポリシーの追加・削除が，他の設定済みポリシーの内容を修正することなく行えるようになる。言い換えれば，設定済みのポリシーを修正して同時実行条件を指定しなおす作業，すなわち，既設定ポリシーの調整は不要となる。

4.3.3 優先度を考慮した実行スケジューリング

同時実行が不可能な場合でも，迅速な対処を要するポリシーが効率よく実行されるように，前項で述べた同時実行制御に加えて，ポリシーの重要度・緊急度に応じた優先実行機能も合わせて実現する。

ポリシーの優先度をポリシー設計者に記述してもらい，ポリシー制御システムのポリシー実行制御機能によって，実行させるポリシーの順序付けを行えるようにする。すなわち，ポリシーA とポリシーB が同時に起動可能な状態で，対処操作の対象リソースの競合があって同時実行が不可能な場合には，ポリシーA とポリシーB のポリシーの優先度により，優先度の高いポリシーを先に実行させるよう制御する。さらに，ポリシーC の実行中に，より優先度の高いポリシーD が実行可能状態になった場合，実行中のポリシーC を中断させて，先にポリシーD を実行させてからポリシーC の実行を再開するといった，優先度によるポリシー実行の追い越しも行えるようにする。

4.4 試作システムの設計

本節では，業務システムの論理構成ツリーに基づくポリシー実行スケジューリングの評価を行うために試作したシステムの設計について述べる。

4.4.1 試作システムのアーキテクチャと処理概要

ポリシー実行制御機能を実現するコンポーネントをポリシーコーディネータと呼ぶ。ポリシーコーディネータを含むポリシー制御システムのアーキテクチャを図 4-5 に示す。ポリシー制御システムには，個々のポリシーを実行するポリシーエンジンと，ポリシーが対象としている業務システムを監視するモニタリングオブジェクトがあり，対象業務システムのシステム構成情報を管理するオブジェクトマネージャが接続されている。ポリシーエンジンは，ポリシーや監視対象の業務システムの構成情報をオブジェクトマネージャより取得して動作する。さらに，個々のポリシーを実行するポリシーエンジンを制御するポリシーコーディネータを，ポリシーエンジンの上位コンポーネントとして設ける。ポリシーコーディネータは，オブジェクトマネージャが保持

する業務システム構成ツリーを用いてポリシー間の同時実行可否を判定し，排他制御が必要なポリシーの組み合わせに対して，それらの優先度やイベント発生時間に基づいて実行をスケジューリングする。

図 4-5 には，ポリシーエンジンが，イベント通知を受信してから対象のポリシーを実行するまでの大まかな流れも示している。図 4-5 の 1，2，及び，3 の処理は，イベント通知毎に個別に行われ，その都度，ポリシーコーディネータが，ポリシーエンジンに呼び出されて，同時実行可否の判定と実行スケジューリングを行う。

ポリシー実行時の基本的な動作サイクルは，図 4-6 に示す通り，ポリシーエンジンのイベント通知受信を契機に実行を開始し，アクション実行の必要性について対処実行条件と監視情報の解析により判定し，ポリシーに指定されたアクションを実行するという流れになる。アクションの実行は，効果判定によって効果が認められるか，指定されたアクションが存在しなくなるまで繰り返し実行される。この繰り返しを N 回アクションループと呼ぶ。アクション実行が不要と判定されるか，実行するアクションがなくなった場合にポリシーの実行を終了する。

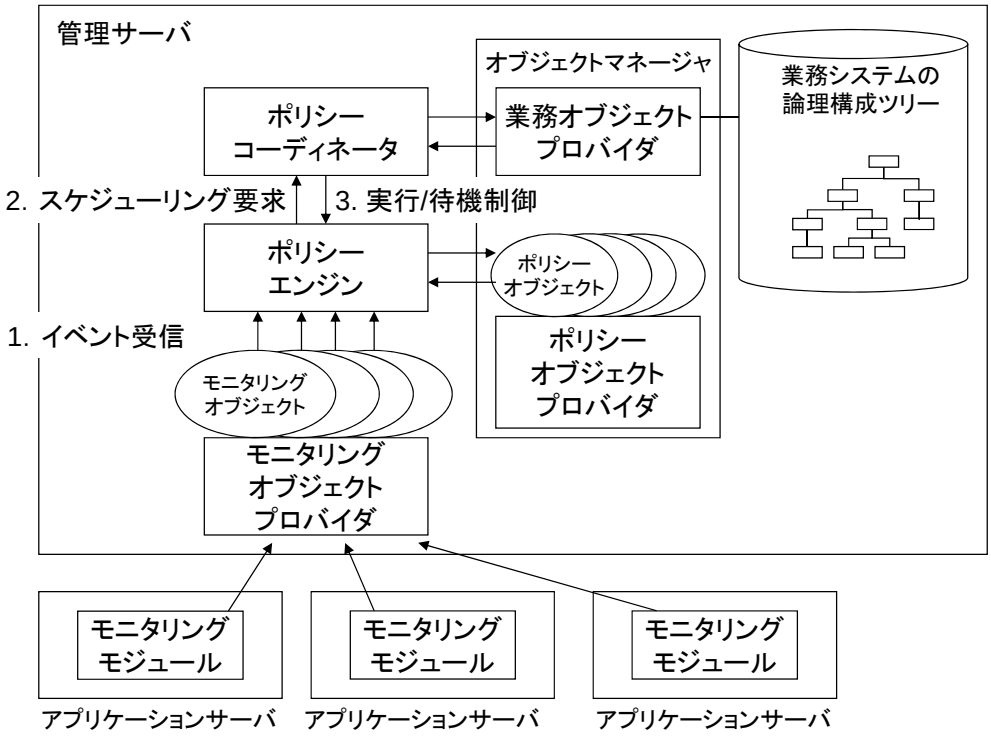


図 4-5 ポリシー制御システムのアーキテクチャ

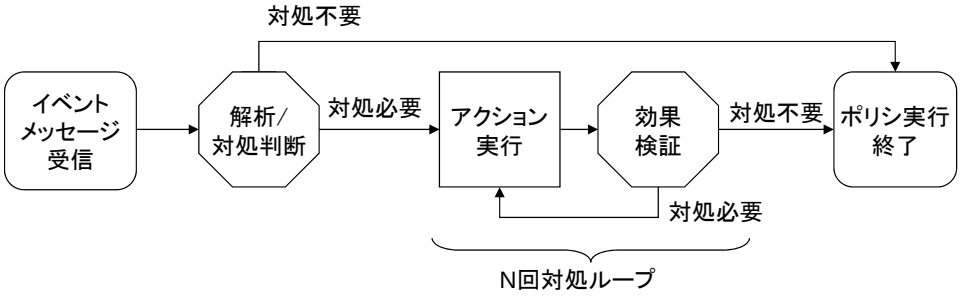


図 4-6 ポリシー実行時の動作サイクル

4.4.2 ポリシー実行のスケジューリング

ポリシーの同時実行が不可能な場合は，ポリシーの優先度に応じて実行するポリシーを決定する。実行中のポリシーよりも優先度が高いポリシーは，実行中のポリシーを追い越して実行されるようにスケジュールする。この追い越し実行のために，ポリシーエンジンは実行制御する対象のポリシーを切り替える動作を行う。ポリシーの実行制御対象の切り替え動作では，ポリシー実行のどのタイミングで制御対象ポリシーの切り替えるかが重要である。切り替えの間隔を長くすると，優先度の高い他のポリシーの実行をブロックする時間が長くなってしまうが，切り替え間隔を短くしてしまうと，切り替えオーバヘッドが大きくなり効率的なポリシー実行の妨げになる。

そこで本研究では，実行制御対象の切り替え間隔を，緊急度あるいは重要度の高いポリシーの実行が長時間ブロックされない範囲で，ポリシー実行処理のトランザクションが成立する最小単位である，「ポリシー実行における解析/対処判断から 1 つのアクションの実行をして効果判定するまで」とした。この様子を，図 4-7 を参照して説明する。ポリシーA は，優先度が低のポリシーで，アクション 1 及びアクション 2 の 2 つのアクションを持つ。ポリシーB は，優先度が高で，1 つのアクションしか持たない。ここで，優先度は，数値が少ない方が優先度は高いものとする。これら 2 つのポリシーが，ポリシーA，ポリシーB という順番で，図 4-7 の時間軸の t0，t1 でそれぞれイベント発生したときのポリシー実行の制御対象の切り替えは，効果判定の完了後の t2 そして t3 で行なう。また，待機させられたポリシーA については，待機中のシステム状態の変化に追従できるように，t4 にて実行を再開する際に，再度最新の監視状態を取得して対処の必要性について再評価する。

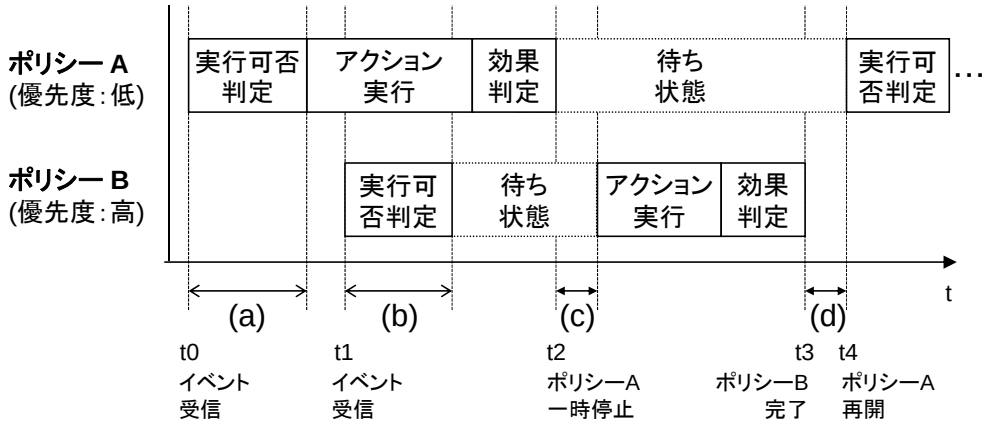


図 4-7 ポリシー実行のコンテキストスイッチ

4.5 提案方式の有効性に関する評価と考察

本節では，ポリシー実行制御機能の性能に関する評価と，ポリシーの記述工数についての評価を行う。性能の評価に関しては，試作したポリシーコーディネータを動作させた際の計測結果から，ポリシーの同時実行可否判定が実用的な時間内で処理できているかを評価する。また，ポリシーの記述の工数については，ポリシーの総記述工数がポリシー設定数に対して指数関数的に増加しない方式となっているかを評価する。

4.5.1 ポリシー実行制御機能の性能の評価

性能の評価では，イベント受信後の同時実行可否の判定に要する時間と優先度によるポリシー実行の切替えに要する時間を計測する。

評価対象は，ロードバランサーと，Web サーバとアプリケーションサーバが同一のサーバ上で稼動する Web/AP サーバで構成されるフロントエンド層と, DB サーバで構成されるバックエンド層とで構成される Web3 階層のオンライン購買システムとする。評価環境のシステム構成を図 4-8 に示す。複数のサーバがネットワークで接続されており，管理 LAN 経由でポリシー制御システムから管理される。ポリシー制御システムが動作するサーバのスペックは, CPU が Intel 社の Pentium4 2GHz で, メモリは 2GByte である。ポリシー制御システムは J2EE で実装したものである。

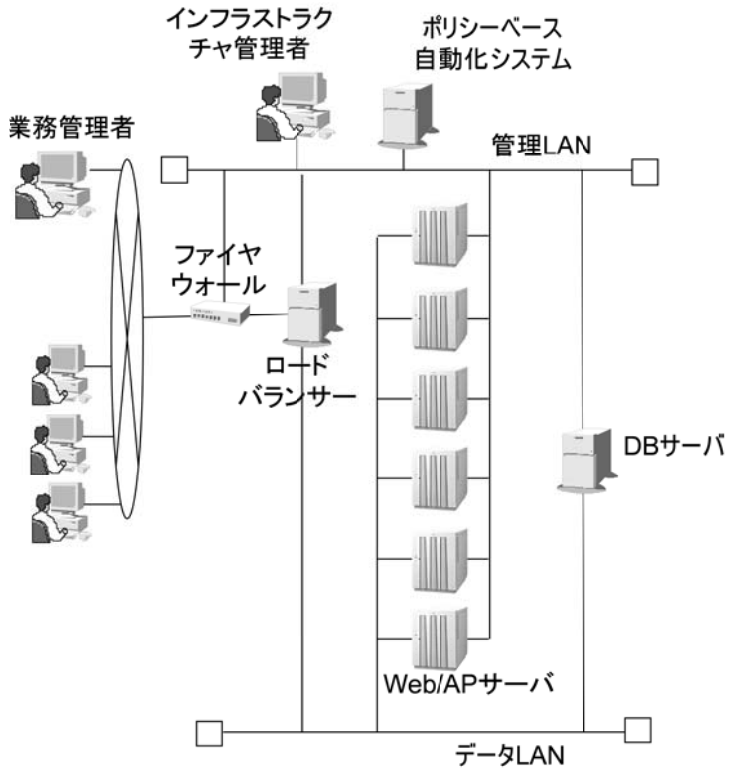


図 4-8 評価環境のシステム構成図

Web3 階層のオンライン購買システムは，Web アプリケーションをベースとした典型的な業務システムであり，広く普及していることから，評価対象として適切であると考える。評価対象の Web3 階層のオンライン購買システムの業務システムにおける，論理構成を表す業務システムの論理構成ツリーは，図 4-3 であるものとする。評価対象の業務システムでは，ユーザとのやり取りを担当するフロントエンド層と，バックエンド層が，オンライン購買システムオブジェクトの配下に接続される。フロントエンド層を示すフロントエンド層オブジェクトの配下には，Web/AP サーバへのリクエストを振り分けるロードバランサーオブジェクトと，ユーザからのリクエストを受け付ける Web 層と，サービスのロジックを実行する AP 層が一体となった Web/AP 層の 2 つのオブジェクトが接続されている。また，バックエンド層には，データを管理する DB オブジェクトが接続されている。Web/AP 層の配下には，さらにスケールアウト可能な各物理サーバに対応した Web/AP オブジェクトが接続される。

この業務システムに，表 4-1 に示す優先度の異なる 2 つのポリシーを設定する。この場合，図 4-7 に示したように，複数のアクションを持つポリシーA のアクション実

行の合間に，ポリシーA よりも優先度の高いポリシーB が割り込んで実行される。この状況において，(1)受信後の同時実行可否の判定にかかる時間と(2)よるポリシー実行の切替えにかかる時間を計測した。

表 4-1 適用するポリシー

ポリシーID	名称	適用先	優先度	アクション
ポリシーA	負荷増大時に Web/AP サーバ追加	Web3Tier Online Shopping	低	1. 適切な台数の Web/AP サーバを追加する
				2. 管理者に通知する
ポリシーB	障害時に Web/AP サーバを切替え	Web/AP Server	高	1. サーバを切替える

同時実行可否の判定に要する時間の計測結果を表 4-2 に示す。イベントを受信して同時実行可否判定を行って，アクションを起動するまでの時間，すなわち，図 4-7 の(a)の時間は 230 ミリ秒となり，同時実行不可の判定となり待ち状態に移行するまでの時間，すなわち，図 4-7 の(b)の時間は 371 ミリ秒となった。また，同時実行可否の判定処理に要した時間はそれぞれ 10 ミリ秒と 16 ミリ秒であった。

表 4-2 同時実行可否の判定時間

フェーズ	トータル処理時間 (ミリ秒)	同時実行可否判定時間 (ミリ秒)	同時実行可否判定時間の割合 (%)
ポリシーA: イベント受信からアクション起動まで	230 (a)	10	4.34
ポリシーB: イベント受信から待ち状態になるまで	371 (b)	16	4.31

次に，優先度によるポリシー実行の切替えに要する時間の計測結果を表 4-3 に示す。ポリシーA からポリシーB への切り替えに要する時間，すなわち，図 4-7 の(c)の時間は 220 ミリ秒，ポリシーB からポリシーA への切り替えに要する時間，すなわち，図 7 の(d)の時間は 365 ミリ秒であった。

表 4-3 優先度によるポリシー実行の切替え時間

ポリシー実行切替え	切替え時間(ミリ秒)
ポリシーA からポリシーB (c)	220
ポリシーB からポリシーA (d)	365

次に，この値の妥当性について考察する。例えば，サーバ数を 10,000，業務システムの数を 1,000，設定されているポリシーの総数を 50,000 であるとする。また，ポリシーの起動契機となるイベントの発生頻度を 3 日に 1 回(16,666 回/日 = 694 回/時 = 12 回/分)，ポリシー実行時間を平均 30 分とし，その 30 分(1,800 秒)の間に 360 回のポリシー起動があるとする。この場合，ポリシー実行制御機能による，イベント受信後の同時実行可否の判定に要する時間と優先度によるポリシー実行の切替えに要する時間の合計が 5 秒(1,800 秒/360 回)以内であれば，想定したイベントの同時実行可否判定が処理できていることになる。

計測の結果，イベント受信後の同時実行可否の判定に要する時間と優先度によるポリシー実行の切替えに要する時間の合計は，長い場合でも 1 秒以下であった。上記で想定した 1,000 業務システムの環境に対するポリシー同時実行可否判定処理は，図 4-5 に示したアーキテクチャでの業務オブジェクトプロバイダの性能が問題となる。しかしながら，業務システムの論理構成ツリーのデータベースへのエントリー数は，業務システム当たり，数十であり，それが 1,000 倍になったとしても，極端に性能が悪くなるとは考えにくく，1,000 業務システムに対しても，ポリシーの同時実行可否判定が実用的な時間内で処理できると考えることができる。

4.5.2 ポリシー総記述工数の評価

設定済みポリシーと追加ポリシー間の同時実行可否指定の調整工数を含めた，ポリシーの総記述工数を評価する。業務システムに対して設定するポリシーの総数を 0 から 50 まで，10 刻みで変化させたときのポリシー総記述工数について，従来方式と提案方式の場合について比較する。

比較結果を図 4-9 に示す。従来方式では，ポリシーの総数が N のとき，N 個から 2

個を取り出して同時実行可否を決定する必要があるため，C(n, 2)の組み合わせの個数分の調整工数がかかる。個々のポリシー記述の初期記述工数をαとすると，n 個のポリシーを記述する際の総工数は，$n\alpha + n(n-1)/2$ と定式化できる。

一方，提案方式では，はじめに業務システムの論理構成ツリーを作成する工数(βとする)が発生するが，その後は，設定済みポリシーの変更は不要で，追加するポリシー記述のみの工数となる。そのため，設定済みポリシー数に依存せず，n 個のポリシーを記述する際の総工数は，$n\alpha + \beta$ となり，ポリシー設定数に対して指数関数的に増加しない方式が実現できたと言える。また，図 4-9 により，業務システムに対して設定するポリシーの数が 15 を超えたとき，業務の論理構成の記述工数を含むポリシーの総記述工数は，従来形式の場合と比較して少なくなることが分かった。

上記において，提案方式で設定済みポリシーの変更が不要としたのは，ポリシーの実行効率を考慮せず，排他制御の指定の簡単さを優先させた場合を考えたからある。実行効率を考慮する場合は，目視での確認による調整が必要で，その場合，ポリシー実行制御機能の動作と同様に，ポリシーを業務システムの論理構成ツリーにマッピングさせた表示が有効である。このような表示を見ながら，ノードの包含関係も考慮することで，従来よりも容易にポリシー間の調整が行えると考える。

4.5.3 業務システムの論理構成ツリーの利用に関する考察

本研究では，業務システムの論理構成ツリーを利用してポリシーの同時実行条件を指定する方式について述べた。しかし，業務システムの論理構成ツリーを用いる場合，ポリシー記述の柔軟性という意味ではデメリットもある。ポリシーの適用先オブジェクトが，業務システムの論理構成ツリーのルートに近い位置にある場合，ロックがかかる範囲が広くなり，同時実行効率が低下する。

また，ツリーという制約により，効率の良い同時実行条件の指定ができない場合がある。その場合の状況を，4.3.2 項の図 4-3 を例にして示す。図 4-3 は，もともと「Load Balancer」と「Web/AP Layer」の間で同時実行不可の指定をすることが多く，「Front-end Layer」と「Back-end Layer」は同時実行可能であることが多い場合の業務システムの論理構成ツリーの構成の仕方である。これに加えて，「Web/AP Layer」と「Database Server」について，同時実行を不可とした同時実行条件を指定することを考える。

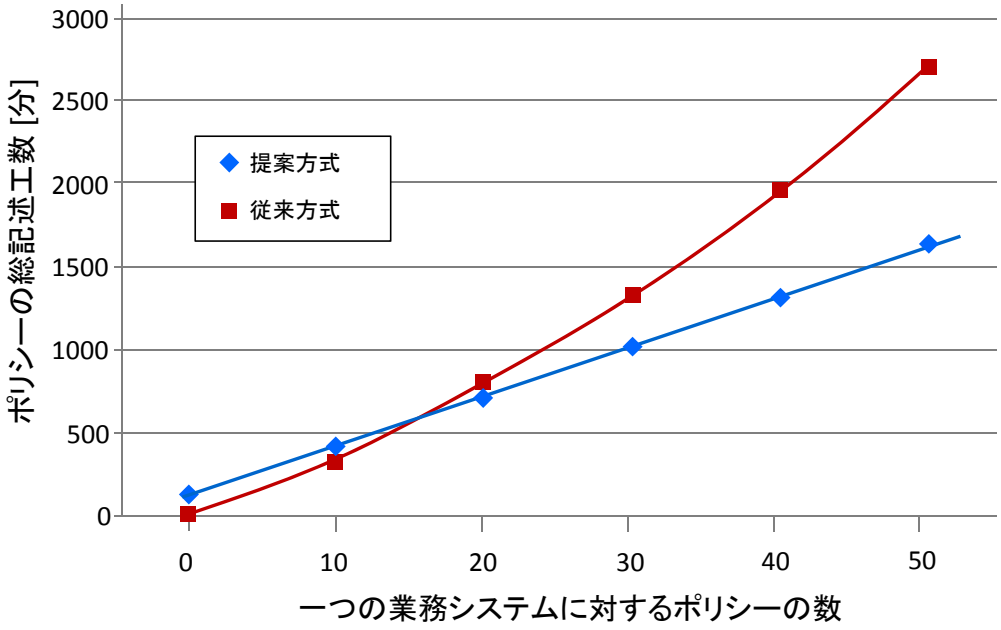


図 4-9 ポリシーの総記述工数

図 4-3 を変更しない場合は，ポリシーの適用先オブジェクトを「Web3Tier Online Shopping」とするのが簡単である。しかし，ロック範囲が広くなるという問題がある。一方，図 4-3 を変更する場合は，図 4-3 に対し，「Mid Layer」を追加配置し，「Web/AP Layer」と「Database Server」を関連付けたくなる。ところが，図 4-10 に示す通り，「Web/AP Layer」は，既に「Front-end Layer」に関連付けられているため，「Mid Layer」に関連付けるとツリー構造ではなくなってしまう。同様に，「Database Server」は，既に「Back-end Layer」に関連付けられているため，「Mid Layer」に関連付けることはできない。

そのため，ツリー構造とするためには，図 4-11 に示す通り，「Web/AP Layer」を「Front-end Layer」の配下に配置するか，「Back-end Layer」の配下に配置するかを決定しなければならない。この問題については，ポリシーの実行頻度を考慮することで，ポリシーのアクションの実行待ち時間が最小になるように決定することになる。

ここでは，ポリシーのアクションの実行頻度の仮定を，表 4-4 のようにおくものとする。表 4-4 は，ポリシーのアクションとそのロック対象，アクションの実行時間，

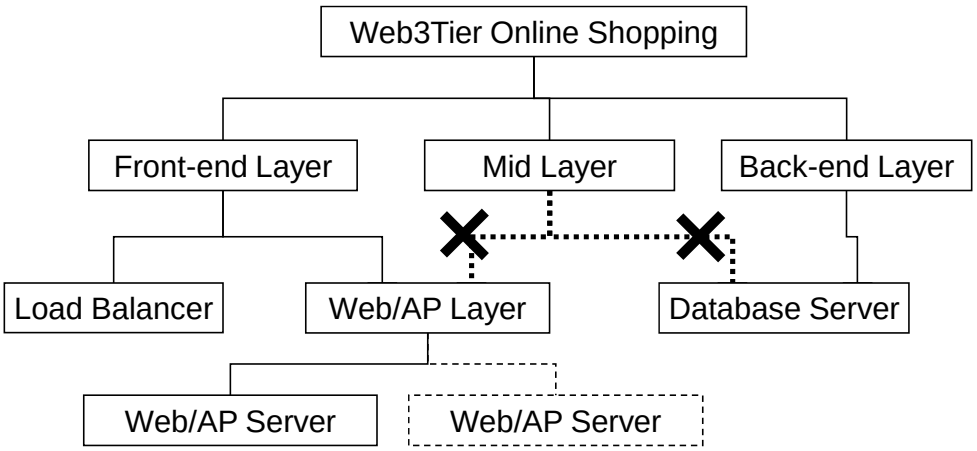


図 4-10 業務システムの論理構成ツリーで効率的な条件指定ができない例

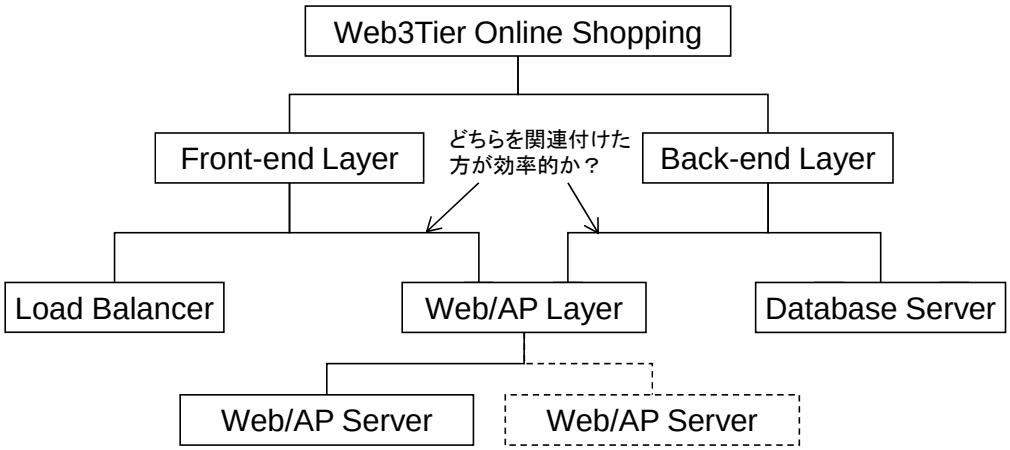


図 4-11 Web/AP Layer の配置問題

実行頻度の仮定値を示した表である。仮定値は，これまでの研究の経験によって定めたものである。Web/AP サーバの動的スケールアウトは，業務システム全体として応答時間などが基準値を上回った場合に，Web/AP サーバを追加することで回復できる場合に行うアクションである。一方，DB サーバの動的スケールアップは，業務システム全体として，応答時間などが基準値を上回った場合に，DB サーバの CPU 割り当て率やメモリ割り当て率を高めることで回復できる場合に行うアクションである。また，ロードバランサー，Web/AP サーバ，DB サーバ単体に対するアクションについても示した。単体に対するアクションは，性能劣化時のチューニングパラメータの変更や，

バックアップ，パッチ適用などあるが，ここでは，まとめて単体アクションとし，これらの実行頻度を 1 ヶ月に 1 回程度とした。

表 4-4 ポリシーのアクションの実行頻度(仮定)

ポリシーのアクション	ロック対象	アクションの 平均実行時間 [分]	実行頻度 [回/月]
Web/AP サーバの 動的スケールアウト	Load Balancer Web/AP Layer	10	30
DB サーバの 動的スケールアップ	Web/AP Layer Database Server	20	1
Load Balancer 単体アクション	Load Balancer	5	1
Web/AP Layer 単体アクション	Web/AP Layer	5	1
Database Server 単体アクション	Database Server	5	1

次に，業務システムの論理構成ツリーの構成案を 3 つ挙げ，ロック時間を比較することで，どの構成案が良いか検討する。3 つの構成案を図 4-12 に示す。

構成案 1 は, 4.3.2 項の図 4-3 と同じ構成であり,「Web/AP Layer」を「Front-end Layer」の配下に配置した構成である。この場合，Web/AP サーバの動的スケールアウトのアクションの 1 ヶ月当たりのロック時間は 300 分となり, DB サーバの動的スケールアップのロック時間 20 分／月よりも長い。ロックの範囲は, それぞれ図示した通りであり, よりロック時間が長い方のアクションのロック範囲を狭くできている。

構成案 2 は，「Web/AP Layer」を「Back-end Layer」の配下に配置した構成である。この場合，よりロック時間の長いアクションのロック範囲の方が広くなっている。

構成案 3 は,「Load Balancer」と「Web/AP Layer」,「Database Server」を同位に「Web3Tier Online Shopping」の直下に配置した構成である。この場合は，常に最もロック範囲の広い「Web3Tier Online Shopping」でロックしなければならい。

構成案 1 から 3 までの，1 ヶ月当たりのロック時間とロック範囲の比較から，構成案 1 が最も他のアクションを待たせる確率が低い構成であることが分かる。

このように，業務システムの論理構成ツリーの構成の決定には，上記のようなアクションの実行頻度，実行時間を考慮し，他のポリシーのアクションの待ち時間を最小にするための比較検討が必要となる。

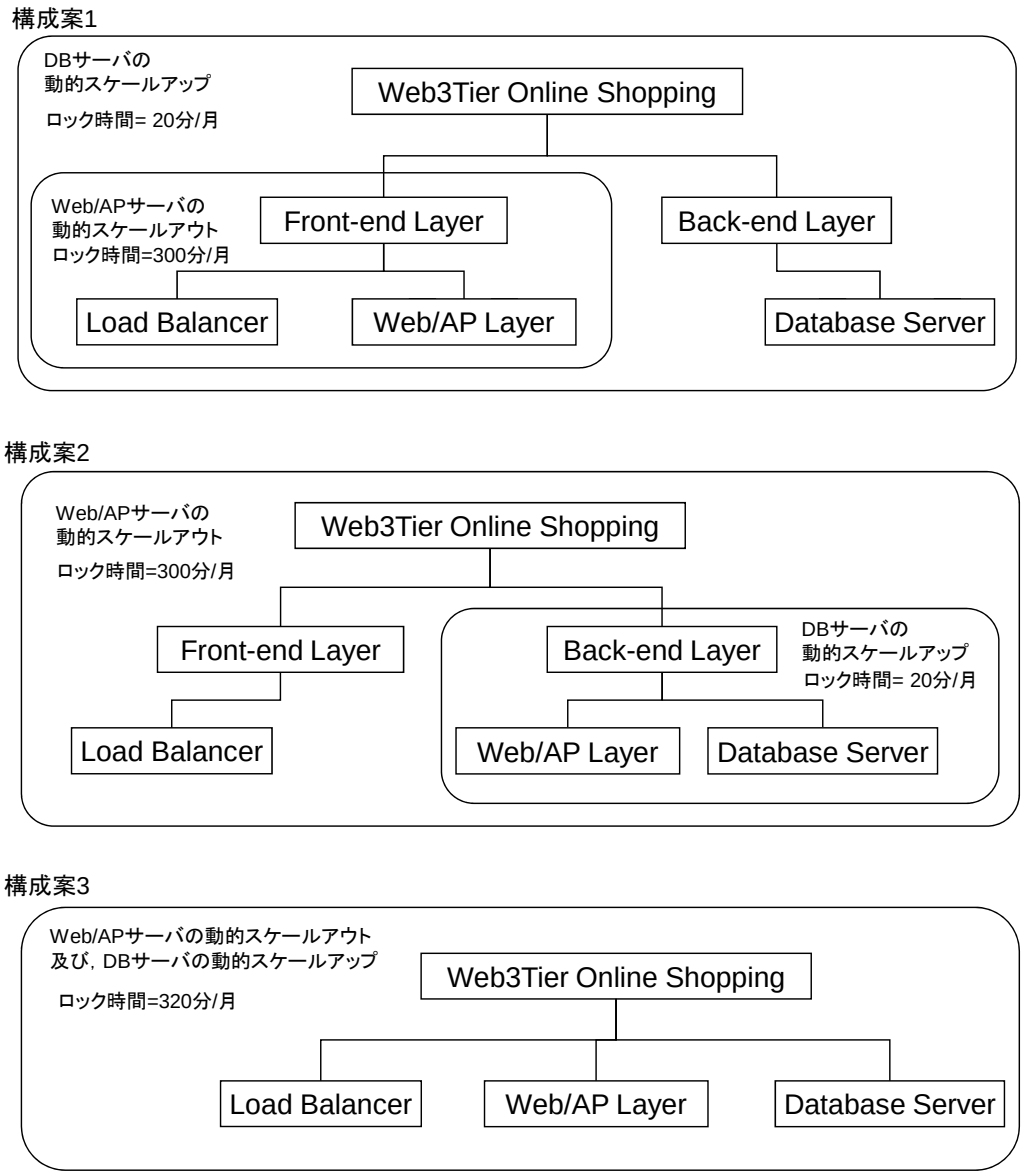


図 4-12 業務システムの論理構成ツリーの構成案とロック時間の比較

4.6 結言

本章では，ポリシー制御システムにおいて，ポリシーの同時実行可否の判定のための情報を業務システムの論理構成ツリーを用いて指定するポリシー記述方式とそれに基づいて動作するポリシー実行制御機能について述べた。

イベント受信後の同時実行可否の判定に要する時間と優先度によるポリシー実行の切替えに要する時間の合計は，長い場合でも 1 秒以下とすることができ，1,000 業務システムに対して，設定されているポリシーの総数が 50,000 の環境に対しても同時実行可否の判定処理が可能であることを示した。また，業務システムの論理構成ツリーを利用してポリシーの同時実行条件を指定する方式を提案し，業務システムに対して設定するポリシーの数が 15 を超えたとき，業務の論理構成の記述工数を含むポリシーの総記述工数が，従来形式の場合と比較して少なくなることを示した。

また，評価の節で述べた通り，業務システムの論理構成ツリーを用いる場合，ポリシーの適用先オブジェクトが，業務システムの論理構成ツリーのルートに近い位置にある場合にロックがかかる範囲が広くなり，同時実行効率が低下するというデメリットがある。本研究では，ポリシーの同時実行の危険を回避するための指定をより容易に行えるようにすることを優先させ，業務システムの論理構成の表現形式にツリー形式という制約を設けた。今後は，他の表現形式による同時実行効率向上のための検討を行っていく。

なお，本研究は，経済産業省主導の下で平成 15 年度より 3 年間実施された「ビジネスグリッドコンピューティングプロジェクト」[79][98][99][100][101][102][103]の成果である。

第5章

結論

5.1 本研究のまとめ

本論文では，大規模化，複雑化が進む情報システムの開発及び運用における知識活用手法の高度化の実現方式について述べた。具体的には，開発フェーズにおける設計レビュー管理の支援方式，運用フェーズにおける障害原因解析処理の高速化方式，運用自動化に向けたポリシー記述と実行方式ついての研究成果を，以下の 4 章に分けて述べた。

第 1 章では，情報システムの開発及び運用における知識活用手法について，設計品質向上のための設計レビュー管理プロセスの効率化，運用フェーズにおける障害原因解析処理の高速化，運用自動化のためのポリシー記述の容易化の 3 つを課題として取り上げ，それぞれの解決方針を示した。

第 2 章では，情報システムの設計開発において知識やノウハウが扱われる設計レビューに着目し，そのプロセスを効率良く進めるための設計レビュー管理支援方式について述べた。具体的には，設計レビューの議事録を活用することによって，決められた設計項目に対して十分な設計レビューを実施したか，レビューでの議論は充実したものだったかを確認する作業を支援する方式を提案した。提案方式を実装した設計レビュー管理システムを実際の開発プロジェクトに適用し，提案方式の実用上の有効性を示した。

第 3 章では，障害原因解析方式について，解析ルールの条件式の要素をルール間で共通化して，条件要素とルール間に直接リンクを張ることによって，入力と条件のマッチングを高速化し，障害イベントの発火状態を一定時間保持させることで，障害イベントの再入力を行うことなく，より少ない計算量で解析を行う方式を提案した。試作システムによる計測結果に基づいて，従来方式として実用化されているルールマトリクス方式と比較して，少ない計算量で障害原因解析処理が行えることを示した。

第 4 章では，ポリシー間の競合を回避するための実行スケジューリングについて，ポリシーの同時実行可否条件を業務システムの論理構成ツリーに基づいて，より容易に指定するためのポリシーの記述方式を提案した。さらに，指定した同時実行条件に基づいてポリシーの同時実行スケジューリングを行う方式について述べ，試作システムによる性能計測に基づく見積もりにより，業務システム数 1,000 の環境に対しても，ポリシー実行スケジューリングが遅延なく動作できることを示した。

5.2 今後の課題

情報システム運用の自動化範囲の拡大に向けて，現場の知識・ノウハウの活用手法をさらに高度化していく必要がある。情報システムの「監視」に関しては，より人にやさしい監視方式が必要となる。現在の監視システムでは，1 日に数百を超えるイベントが報告される。ノウハウを持つ管理者であれば，ある種類のイベントは 1 日に 10 回までなら無視しても問題ないことを知っている。1 日に 10 回以上発生すれば，関連する別の監視項目を調査するなど，分析対象の幅を広げた方が良いことを知っている。このような監視を行うことは，現在の監視技術でも不可能ではないが，監視の設定が複雑になる。業務システムの監視方法に関する知識やノウハウを取り込んで活用するためには，情報システムの種類や特性に応じた監視設定のパターン化や動的な監視制御のための方式を検討する必要がある。

また，監視・分析に基づくアクション(計画・実行)に関しては，業務システム固有の扱いが必要になることが課題であり，これを解決するためには，設計フェーズで作られた運用手順の運用フェーズでの活用が必要となる。例えば，業務システムの負荷増大時に，スケールアウトすべきかスケールアップすべきかなど，業務システムに固有の運用手順を形式化し，動作環境はこれを解釈して，自律的に稼働状態を維持するための研究が必要である。

開発チームが作り込んだ業務システムの特性や癖のような情報を，運用チームにノウハウとして伝達して，運用の効率化に役立てる。逆に，運用チームが持つ業務システムの挙動に関する情報を開発チームにフィードバックすることで，開発の効率向上に役立てる。この相互プロセスの効率化のための情報共有方式に関しても検討を行う必要がある。

謝　辞

本研究の全過程を通じ，懇切なるご指導とご鞭撻を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 薦田憲久教授に心から感謝申し上げます。

情報科学研究科博士後期課程において，情報工学全般に関して親切なるご指導とご助言を賜るとともに，本研究をまとめるにあたって貴重なお時間を割いて頂き，丁寧なるご教示を賜りました大阪大学大学院情報科学研究科 マルチメディア工学専攻 下條真司教授，原隆浩准教授に謹んで深謝致します。

情報科学研究科博士後期課程において，情報工学全般に関して親切なるご指導とご助言を賜りました，大阪大学大学院情報科学研究科マルチメディア工学専攻 西尾章治郎教授，藤原融教授，細田耕教授，秋吉政徳准教授に深く感謝申し上げます。

本研究の機会を与えていただくとともに，温かいご指導とご鞭撻を賜った，（株）日立製作所 横浜研究所 所長 堀田多加志博士，前所長 前田章博士，情報プラットフォーム研究センター 松並直人センター長，主管研究長岩嵜正明博士，前主管研究長 新井利明博士，研究開発本部 技師長 山本彰博士に心から御礼申し上げます。また，研究を進めるにあたり日々様々なご支援とご配慮を頂きました（株）日立製作所 横浜研究所の先輩，同僚，後輩の方々に心から御礼申し上げます。

第 2 章の研究にあたり，研究の機会を与えて頂くとともに，共同研究者として様々なご討論ご助言を頂きました（株）日立製作所 旧公共情報事業部 大野治博士，小室彦三氏，幕田行雄氏，降旗由香理氏，宮崎肇之氏，川辺博史氏，福士有二氏，渡辺正氏，（株）日立製作所 旧システム開発研究所 小泉忍氏，平井千秋博士に心から御礼申し上げます。

第 3 章の研究に当たり，研究の機会を与えて頂くとともに，米国出向からの帰国に際し，学位取得を勧めて頂きました（株）日立製作所 ソフトウェア事業部 の増石哲也博士（日立データシステムズ出向中）に深く感謝致します。また，共同研究者としてご討論ご助言を頂きました，（株）日立製作所 横浜研究所 運用管理システム研究部 菅内公徳氏，藤田高広氏，名倉正剛博士，永井崇之氏，國井雅氏に感謝致します。

第 4 章の研究に当たり，研究の機会を与えて頂くとともに，共同研究者として様々

なご討論ご助言を頂きました（株）日立製作所 ソフトウェア事業部 吉野松樹博士，（株）日立製作所（情報）経営戦略室 渡邉友範氏，研究開発本部 佐川暢俊氏（日立アジア出向中），スマートシティ推進本部 石﨑健史氏に深く御礼申し上げます。また，共同研究者として，ご討論ご助言頂きました（株）日立製作所 横浜研究所 運用管理システム研究部 増岡義政部長，増田峰義氏，飯塚大介博士，ソフトウェアプラットフォーム研究部 羽賀太氏（日立アメリカ出向中），（株）日立製作所 ソフトウェア事業部 鈴木友峰氏，富沢広幸氏，畠山敦氏，溝手裕二氏，（株）日立製作所 ＲＡＩＤシステム事業部 里見充則氏に感謝致します。

第 3 章，及び，第 4 章の研究にあたり，共同研究者として様々なご討論ご助言を頂きました（株）日立製作所 ソフトウェア事業部 森村知弘博士（日立データシステムズ出向中）に感謝致します。

最後に，いつも体調を気遣ってくれた父母に感謝致します。また，本論文の執筆に当たり，温かく励まし支えてくれた妻 朋子と子供たち 晴彦，拓海，優維に心から感謝致します。

参考文献

[1] 社団法人 日本情報システム・ユーザー協会(JUAS): “重要インフラ情報システム信頼性研究会報告書”, http://sec.ipa.go.jp/reports/20090409.html (2009).

[2] 市嶋洋平: “ダウンに動ぜず”, 日経コンピュータ, 2009.2.15 号, pp.36-51 (2009).

[3] 市嶋洋平, 小原忍: “障害発生！被害拡大と復旧の分かれ目”, 日経コンピュータ, 2007.10.29 号, pp.44-57 (2007).

[4] U. K. Office of Government Commerce: “*Applications Management: ITIL (Information Technology Infrastructure Library)*”, Stationery Office, London, United Kingdom (2002).

[5] Project Management Institute: “*A Guide to the Project Management Body of Knowledge (PMBOK Guide)*”, Project Management Institute (2000).

[6] W. S. Humphrey: “*Managing the Software Process*”, Addison-Wesley (1989).

[7] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber: “Capability Maturity Model, version 1.1”, *IEEE Software*, Vol.10, No.4, pp.18-27 (1993).

[8] CMMI Product Team: “*CMMI for Development, Version 1.2*”, Software Engineering Institute, Carnegie Mellon (2006).

[9] 大平雅雄, 横森励士, 阪井誠, 岩村聡, 小野英治, 新海平, 横川智教: “ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム”, 電子情報通信学会論文誌 D-I, Vol.J88-D-I, No.2, pp.228-239 (2005).

[10] 松村知子, 森崎修司, 勝又敏次, 玉田春昭, 吉田則裕, 楠本真二, 松本健一: “問題の早期発見・改善を支援するインプロセスプロジェクト管理手法の実プロジェクトへの適用”, 電子情報通信学会論文誌 D, Vol.J92-D, No.11, pp.1974-1986 (2009).

[11] 大平雅雄, 横森励士, 阪井誠, 松本健一, 井上克郎, 鳥居宏次: “Empirical Project Monitor: プロセス改善支援を目的とした定量的開発データの自動収集・分析システムの試作”, 電子情報通信学会技術研究報告, ソフトウェアサイエンス Vol.103, No.708, pp.13-18 (2004).

[12] 工藤裕, 小泉忍, 川辺博史, 降旗由香理: “ソフトウェアの分散開発拠点に対応した問題点管理システム”, 情報処理学会 第 55 回全国大会 講演論文集(1), pp.439-440 (1997).

[13] 村田大二郎, 工藤裕, 平井千秋, 伊野谷祐二, 三富篤: “ソフトウェア開発プロジェクト内での情報流通インフラの開発と評価”, 情報処理学会研究報告（グループウェアとネットワークサービス研究会）, Vol.2001, No.48, pp.35-40 (2001).

[14] 村田大二郎, 平井千秋, 工藤裕, 藤波武起, 鐘ヶ江博: “ソフトウェア開発における約束懸案管理の方法論の提案”, プロジェクトマネジメント学会 2003 年度秋季研究発表大会予稿集, pp.156-159 (2003).

[15] 門脇憲治, 小泉寿男: “知識情報管理によるソフトウェア再利用方式の一提案”, 情報処理学会研究報告（ソフトウェア工学研究会）, Vol.2002, No.23, pp.141-146 (2002).

[16] 横森励士, 藤原晃, 山本哲男, 松下誠, 楠本真二, 井上克郎: “利用実績に基づくソフトウェア部品重要度評価システム”, 電子情報通信学会論文誌 D-I, Vol.J86-D-I, No.9, pp.671-681 (2002).

[17] 渡辺正, 降旗由香理, 宮崎肇之, 工藤裕: “システム開発事例共有システムの開発”, 情報処理学会 第 57 回全国大会 講演論文集(4), pp.268-269 (1998).

[18] 工藤裕, 平井千秋, 森田靖, 矢野理: “再利用支援データベース「AWARD」の開発”, 情報処理学会 第 61 回全国大会 講演論文集(4), pp.15-16 (2000).

[19] 永井愛之, 矢野理, 森田靖, 高木勝則, 平井千秋, 工藤裕: “文書分類方法および文書再利用システム”, 情報処理学会 第 63 回全国大会 講演論文集(3), pp.3-4 (2001).

[20] 経済産業省: “平成 18 年度情報処理実態調査報告書”, http://www.meti.go.jp/statistics/zyo/zyouhou/result-2/pdf/3_H18report_rev071204.pdf (2007).

[21] 後藤邦仁, 望月秀樹, 宗像勉, 泉全: “IT サービス運用を最適化するソリューション”, 日立評論, Vol.87, No.4, pp.359-362 (2006).

[22] G. Ganek and T. A. Corbi: “The Dawning of the Autonomic Computing Era”, *IBM System Journal*, Vol.42, No.1, pp.5-18 (2003).

[23] G. Lanfranchi, P. D. Peruta, A. Perrone, and D. Calvanese: “Toward a New Landscape of Systems Management in an Autonomic Computing Environment”, *IBM System Journal*, Vol.42, No.1, pp.119-128 (2003).

[24] J. O. Kephart and D. M. Chess: “The Vision of Autonomic Computing”, *IEEE Computer*, Vol.36, No.1, pp.41-50 (2003).

[25] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland: “A Concise Introduction to Autonomic Computing”, *Advanced Engineering Informatics*, Vol. 19, pp.181-187 (2005).

[26] IBM: “Tivoli Software”, http://www-01.ibm.com/software/tivoli/ (2011).

[27] （株）日立製作所：“統合システム運用管理 JP1”, http://www.hitachi.co.jp/Prod/comp/soft1/jp1/ (2011).

[28] 富士通（株）：“統合運用管理ソフトウェア Systemwalker”, http://systemwalker.fujitsu.com/jp/ (2011).

[29] 日本電気（株）：“統合運用管理 WebSAM”, http://www.nec.co.jp/middle/WebSAM/ (2011).

[30] M. Hasan, B. Sugla, and R. Viswanathan: “A Conceptual Framework for Network Management Event Correlation and Filtering Systems”, in *Proc. of 6th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 233–246 (1999).

[31] V. Holub, T. Parsons, and P. O’Sullivan: “Run-Time Correlation Engine for System Monitoring and Testing”, in *Proc. of the IEEE International Conference on Autonomic Computing (ICAC-09)*, pp.9-17 (2009-6).

[32] A. Hanemann, M. Sailer, and D. Schmitz: “Assured Service Quality by Improved Fault Management (Service-Oriented Event Correlation)”, in *Proc. of the 2nd International Conference on Service Oriented Computing (ICSOC’04)*, pp.183-192 (2004).

[33] M. Davis: “*201 Principles of Software Development*”, McGraw-Hill (1995).

[34] J. Conklin and M. L. Begeman: “gIBIS: A Hypertext Tool for Exploratory Policy Discussion”, in *Proc. of the 1988 ACM Conference on Computer-Supported Cooperative Work (CSCW’88)*, pp.140-152 (1988).

[35] 古宮誠一, 西野光: “ソフトウェア設計上の意思決定と判断根拠の記録/再利用のモデル”, 電子情報通信学会技術研究報告, 知能ソフトウェア工学 Vol.93, No.211, pp.41-49 (1993).

[36] 古宮誠一: “ソフトウェア設計上の意思決定とその根拠の情報を再利用する方法”, 電子情報通信学会論文誌 D-I, Vol.J84-D-I, No.1, pp.78-89 (2001).

[37] 数野顕, 高林貴美, 石井宏次, 合田信久, 大野邦夫: “XML を用いた簡易ビジネス情報ライブラリの検討”, 情報処理学会研究報告（情報基礎とアクセス技術研究会）, Vol.2000, No.71, pp.41-48 (2000).

[38] 梅田恭子, 安田孝美, 横井茂樹: “XML を用いた知識管理方式の提案と考察”, 情報処理学会研究報告（情報システムと社会環境研究会）, Vol.2000, No.32, pp13-18 (2000).

[39] 小嶋弘行, 岩田健: “業務知識の共有に向けた紙, 電子文書のシームレス管理方式”, 電気学会 C 部門論文誌, Vo1.130, No.4 pp.598-606 (2010).

[40] 田野俊一, 増位庄一, 坂口聖治, 舩橋誠壽: “知識ベースシステム構築用ツール EUREKA における高速処理方式”, 情報処理学会論文誌, Vol.28, No.12, pp.1255-1268 (1987).

[41] 田野俊一, 増位庄一, 大森勝美: “高速双方向推論のための ST-NET 生成アルゴリズム”, 情報処理学会論文誌, Vol.30, No.9, pp.1092-1102 (1989).

[42] L. Forgy: “Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”, *Artificial Intelligence*, Vol. 19, pp.17-37 (1982).

[43] S. A. Yemini, S. Kliger, and E. Mozes: “High Speed and Robust Event Correlation”, *IEEE Communications Magazine*, Vol.34, No.5, pp.82-90 (1996).

[44] M. Gupta and M. Subramanian: “Preprocessor Algorithm for Network Management Codebook”, in *Proc. of the Workshop Intrusion Detection and Network Monitoring*, pp. 93-102 (1999).

[45] 敷田幹文: “大規模サーバ間の部品依存関係に基づく障害通知方式の提案”, 情報処理学会論文誌, Vol.49, No.3, pp.1185-1193 (2008).

[46] 佐藤彰洋, 長尾真宏, 小出和秀, 木下哲男, 白鳥則郎: “ネットワーク管理におけるイベント発生状況の効率的な把握を実現するイベント分析価値評価手法の提案と評価”, 情報処理学会論文誌, Vol.50, No.3, pp.992-1001 (2009).

[47] 敷田幹文, 後藤宏志: “大規模サーバ間の部品依存関係に基づくログ管理支援法”, 情報処理学会論文誌, Vol.49, No.3, pp.1081-1089 (2008).

[48] 後藤宏志, 敷田幹文: “サーバの依存関係を考慮したログ情報による障害管理支援の提案”, 情報処理学会研究報告（マルチメディア通信と分散処理研究会）, Vol.2006, No.121, pp.37-42 (2006).

[49] OASIS (Organization for the Advancement of Structured Information Standards): “SAF (Symptoms Automation Framework) TC (Technical Committee)”, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=saf. (2010).

[50] 中野功一: “「信頼できるインターネット」を実現する IP QoS とポリシー管理”, 情報処理, Vol.40, No.10, pp.1004-1006 (1999).

[51] 松原正純, 鈴木和宏, 勝野昭: “自律コンピューティングに向けた HPC 向け動的負荷分散機構(動的負荷分散)”, 情報処理学会論文誌, Vol.44, SIG 11(ACS3), pp.89-100 (2003).

[52] E. Kandogan, C. S. Campbell, P. Khooshabeh, J. Bailey, and P. P. Maglio: “Policy-based Management of an E-commerce Business Simulation: An Experimental Study”, in *Proc. of the 3rd International Conference on Autonomic Computing (ICAC’06)*, pp.33-42 (2006).

[53] G. Kaiser, J. Parekh, P. Gross, and G. Valetto: “KinestheticseXtreme: An External Infrastructure for Monitoring Distributed LegacySystems”, in *Proc. of the Autonomic Computing Workshop 5th Annual International Workshop on Active Middleware Services (AMS’03)*, pp.20-30 (2003).

[54] S. M. De Franceschi, K. S. Borges, R. Moraes, and F. Vasques: “Autonomic Computing Systems: Using AI Techniques for the Development of Agents in the Network Management Domain”, *WSEAS Transaction on Communications*, Vol.5, No.8, pp.1353-1360 (2006).

[55] S. Dobson, R. Sterritt, P. Nixon, and M. Hinchey, “Fulfilling theVision of Autonomic Computing”, *IEEE Computer*, Vol.43, No.1, pp.35-41(2010).

[56] J. O. Kephart and W. E. Walsh: “An Artificial IntelligencePerspective on Autonomic Computing Policies”, in *Proc. of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pp.3-12 (2004).

[57] 佐藤善郎，更田洋吾，長田充弘，増岡義政，谷村武洋: “ポリシーベース自律運用を実現する運用管理ソリューション”，日立評論，Vol.87, No.4, pp.45-50 (2005).

[58] R. Boutaba and I. Aib: “Policy-Based Management: A Historical perspactive”, *Journal of Network and Systems Management*, Vol.15, No.4, pp.447-480 (2007).

[59] H. Chan and T. Kwok: “A policy-based management system with automatic policy selection and creation capabilities by using a singular value decomposition technique”, in *Proc. of the 7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’06)*, pp.96-99 (2006).

[60] IBM: “An Architectural Blueprint for Autonomic Computing", http://www-01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf (2006).

[61] DMTF: “CIM Simplified Policy Language (CIM-SPL) 1.0.0”, http://www.dmtf.org/sites/default/files/standards/documents/DSP0231_1.0.0.pdf (2009).

[62] D. Agrawal, S. Calo, K. W. Lee, and J. Lobo: “Issues in Designing a Policy Language for Distributed Management of IT Infrastructures”, in *Proc. of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, pp 30-39 (2007).

[63] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen: “Policy core information model

(PCIM) version 1 specification", *Request for Comment 3060*, Network Working Group (2001).

[64] 小島功: "ECA アーキテクチャを支援するための SQL サーバの拡張", 情報処理学会研究報告（データベースシステム研究会）, Vol.1996, No.11, pp.57-64 (1996).

[65] P. Kangsabanik, R. Mall, and A. K. Majumdar: "Concurrency control of nested cooperative transactions in active DBMS", in *Proc. of the 4th International Conference on High-Performance Computing*, pp.4-9 (1997).

[66] 野中郁次郎, 遠山亮子, 紺野登: "『知識創造企業』再訪問", 組織科学, Vol.33, No.1, pp.35-47 (1999).

[67] I. Nonaka and H. Takeuchi: "*The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*", Oxford University Press (1995).

[68] I. Nonaka, R. Toyama, and N. Konno: "SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation", *Long Range Planning*, Vol.33, No.1, pp.5-34 (2000).

[69] ゲオルク・フォン・クロー, 野中郁次郎, 一條和生: "ナレッジ・イネーブリング", 東洋経済新報社 (2001).

[70] G. M. Weinberg: "*The Psychology of Computer Programming*", Dorset House Publishing (1971).

[71] 工藤裕, 平井千秋, 川辺博史, 降旗由香理, 大野治: "議事録を利用した設計レビュー管理システムの開発と評価", 情報処理学会論文誌, Vol.44, No.5, pp.1404-1412 (2003).

[72] Y. Kudo, C. Hirai, Y. Furuhata, T. Watanabe, and O. Ohno: "A Proposal of a Review-Report-Oriented Knowledge-Management Model", in *Proc. of the 2nd World Congress for Software Quality*, pp.199-204 (2000).

[73] 平井千秋, 工藤裕, 降旗由香理: "事例 3:ナレッジマネジメントのソフトウェア開発への適用－日立製作所の事例－", 人工知能学会学会誌 特集「ナレッジマネジメントとその支援技術」, Vol.16, No.1, pp.59-63 (2001).

[74] Y. Kudo, C. Hirai, Y. Furuhata, T. Watanabe, and O. Ohno: "A Review-Report-Oriented Knowledge-Management System", in *Proc. of the 3rd Workshop on Software Engineering over the Internet (in the 22nd International Conference on Software Engineering)*, pp.80-88 (2000).

[75] H. Komuro, O. Ohno, Y. Furuhata, Y. Makuta, A. Harada, Y. Kudo, and K. Yasuda: "Project Management Tool utilizing Review Reports for Software Development", in *Proc. of the*

International Conference on Project Management, pp.171-176 (2002).

[76] 工藤裕, 森村知弘, 菅内公徳, 薦田憲久: “障害原因解析のためのルール記述方法とその実行方式”, 電気学会 情報システム研究会, IS-09-71, pp.1-6 (2009).

[77] Y. Kudo and S. M. Batra: “Hitachi IT Operations Analyzer - Root Cause Analysis for Supporting Fault Identification”, Product White Paper, http://itoperations.hds.com/Documents/en/Product%20Resources/IT%20Operations%20Analyzer%20Root%20Cause.pdf (2009).

[78] Y. Kudo and S. M. Batra: “Hitachi IT Operations Analyzer – Topological List View Enhances Ability to Monitor Complex IT Systems”, Product White Paper, http://itoperations.hds.com/Documents/en/Product%20Resources/Hitachi%20IT%20Operations%20Analyzer%20Topological%20List.pdf (2009).

[79] 宮川伸也, 佐治信之, 工藤裕, 田崎英明: “ビジネスグリッド技術解説”, 情報処理, Vol.47, No.9, pp.953-961 (2006).

[80] 工藤裕, 森村知弘, 増岡義政, 薦田憲久: “情報システムの運用自動化に向けたポリシー記述形式とポリシー実行スケジューリング方式”, 電気学会 C 部門論文誌, Vol.131, No.10 (掲載予定) (2011).

[81] 工藤裕, 森村知弘, 増岡義政, 薦田憲久: “業務システム動的構成変更のためのポリシー実行スケジューリング方式”, 電気学会 情報システム研究会, IS-10-74, pp.5-10 (2010).

[82] E. Yourdon: “*Structured Walkthroughs*”, Prentice-Hall (1989).

[83] T. Gilb and D. Graham: “*Software Inspection*”, Addison-Wesley (1993).

[84] S. McConnell: “The Best Influences on Software Engineering”, *IEEE Software*, January/February 2000, pp.10-17 (2000).

[85] W. S. Humphrey: “*Managing the Software Process*”, Addison-Wesley (1989).

[86] 堀内純孝: “役に立つデザインレビュー”, 日科技連 (1992).

[87] International Standardization Organization: “*ISO/IEC 9126*” (1991).

[88] F. Shull, I. Rus, and V. Basili: “How Perspective-Based Reading Can Improve Requirements Inspections”, *IEEE Computer*, Vol.33, No.7, pp.73-79 (2000).

[89] 西田正吾: “知識の伝承を支援する情報技術の動向とその電力分野への応用”, 電気学会 B 部門論文誌, Vol.114, No.9, pp.839-842 (1994).

[90] DMTF: “Common Information Model (CIM) Standards”, http://www.dmtf.org/standards/cim/ (2009).

[91] DMTF: “Systems Management Architecture for Server Hardware (SMASH) Initiative”, http://www.dmtf.org/standards/mgmt/smash/ (2009).

[92] DMTF: “Desktop and mobile Architecture for System Hardware (DASH) Initiative”, http://www.dmtf.org/standards/mgmt/dash/ (2009).

[93] DMTF: “Virtualization Management (VMAN) Initiative”, http://www.dmtf.org/standards/mgmt/vman/ (2009).

[94] SNIA: “Storage Management Initiative”, http://www.snia.org/forums/smi/ (2009).

[95] J. Allspaw: “*The Art of Capacity Planning: Scaling Web Resources*”, O'Reilly Media (2008).

[96] D. A. Menasce, L. W. Dowdy, and V. A. F. Almeida: “*Performance by Design: Computer Capacity Planning by Example*”, Prentice Hall (2004).

[97] R. R. M. Lotlikar, R. R. Vatsavai, M. Mohania, and S. Chakravarthy: “Policy schedule advisor for performance management”, in *Proc. of the 2nd International Conference on Autonomic Computing (ICAC’05)*, pp.183-192 (2005).

[98] 吉野松樹, 阿部成, 中誠一郎: “ビジネスグリッドの狙い”, 情報処理, Vol.47, No.9, pp.947-952 (2006).

[99] 幕田幸男, 佐々木一陽, 阿部秀哉, 藤野修司: “ビジネスグリッドの実証実験”, 情報処理, Vol.47, No.9, pp.962-969 (2006).

[100] 福井恵右, 岸本光弘, 佐川暢俊, 中田登志之, 森拓也, 舘村純一: “ビジネスグリッド関連技術動向と標準化活動”, 情報処理, Vol.47, No.9, pp.970-977 (2006).

[101] 独立行政法人 情報処理推進機構: “ビジネスグリッドが切り開く次世代 IT 基盤”, アスキー (2006).

[102] 独立行政法人 情報処理推進機構: “ビジネスグリッドコンピューティングプロジェクト”, http://www.ipa.go.jp/software/bgrid (2005).

[103] 吉野松樹: “日立製作所の「Harmonious Computing」を中心としたビジネスグリッドコンピューティングへの取り組み (特集 1 ビジネスグリッドコンピューティング)”, COMPUTER & NETWORK LAN, オーム社, Vol. 22, No.8(通巻 250 号), pp.54-62 (2004).