DEWS2004 I-12-02

ワークフローの自動実行における障害対策

加藤　英之† 　小林　隆志†† 　横田　治夫††,†††

† 東京工業大学 工学部 情報工学科　〒 152–8552 東京都目黒区大岡山 2–12–1
†† 東京工業大学 学術国際情報センター　〒 152–8550 東京都目黒区大岡山 2–12–1
††† 東京工業大学 大学院 情報理工学研究科 計算工学専攻　〒 152–8552 東京都目黒区大岡山 2–12–1
E-mail: †katoh@de.cs.titech.ac.jp, ††tkobaya@gsic.titech.ac.jp, †††yokota@cs.titech.ac.jp

あらまし　ワークフローは，複数の処理ノードで実行されることから，処理ノードの障害についての考慮が必要である．本稿では，処理ノードの単一故障に耐えることのできる Web サービスを用いたワークフロー制御の実現を目指す．ワークフロー管理を 1 つのノードで行うと，そのノードの負荷が大きくなり，また，そのノードの故障に耐えられない．そこで，ハッシュ計算によりワークフロー管理タスクを分散し，さらにバックアップ用のノードを用意することで，単一故障に耐えることのできる信頼性の高いワークフロー制御を提案する．ここでは，提案手法の適用例として，これまで研究してきたマルチメディアコンテンツの蓄積・配信を行う UPRISE のワークフローを対象に，その有用性・信頼性について考察する．

キーワード　ワークフロー，Web とインターネット，並行処理とリカバリ，e-service，バックアップ

Failure Handling in Automatic Workflow Executions

Hideyuki KATOH†, Takashi KOBAYASHI††, and Haruo YOKOTA††,†††

† Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology
Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8552 Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8550 Japan
††† Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology　Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8552 Japan
E-mail: †katoh@de.cs.titech.ac.jp, ††tkobaya@gsic.titech.ac.jp, †††yokota@cs.titech.ac.jp

Abstract　As workflow consists of multiple processing nodes, it is necessary to consider the failure of them. In this paper, it is the goal to control workflow with Web Services which is able to tolerate Single Point Of Failure(SPOF) of processing nodes. When a processing node manages whole workflow, the node is heavy, and becomes SPOF of the workflow. So we propose reliable workflow control by hashing and setting up backup nodes to tolerate a failure through the distributed tasks in a workflow. We consider reliability and availability with the use of UPRISE workflow as an example of application, which is an experimental system to store and deliver the multimedia contents.

Key words　workflow, Web and the Internet, concurrency and recovery, e-service, backup

1. はじめに

ワークフローとは，工程 (アクティビティ) の集合として作業 (プロセス) を表現し，作業の一部もしくは全てを自動化することである [1]．また，ジョブとはワークフロー内を実際に流れる作業の一つである．ワークフロー管理システムとは，自動化された作業を管理することでワークフローを管理するシステムである．ワークフロー管理システムを導入することにより，人的コストの削減など様々なメリットが期待できる．現在，商用・研究用を問わず多数のワークフロー管理システムが存在する．

一方で，近年，Web サービスが注目されている．Web サービスとは，ネットワークを介して HTTP と SOAP を用いて，ソ

フトウェアシステム間で通信を行うオープンな技術である [2]. Web サービスは，実行環境に依存しないアプリケーションの連携が可能なため，CORBA や DCOM などの分散オブジェクト技術よりも利用が容易である．

Web サービスが登場した当初は，SOAP [3], WSDL [4], UDDI [5] などの基盤となる技術についての検証が行われ，その後 WS-Security, XML Signature などのセキュリティ・相互接続性の確保について研究されてきた．そして現在，BPEL4WS, WS-Transaction, WS-Coordination, WS-CAF などを用いた Web サービスの実用化に向けた仕様の策定が行われている．BPEL4WS [6] は XML ベースのワークフロー定義言語であり，WS-BusinessActivity や WS-AtomicTransaction [7] や WS-Coordination [8] はアプリケーションの動作やトランザクションの信頼できる結果を調整する方法を提供する．また，WS-CAF 仕様は，複数の Web サービス間でトランザクションに関する重要な情報を共有できるようにシステムの設定を行う [9]．これらの仕様の登場により，ワークフローを Web サービスを用いて実現することが盛んに研究されている．しかし，障害対策に関しては検討課題が多い．例えば，BPEL4WS では故障検出までしか定義されておらず，耐故障性を考慮したワークフローの定義を記述することができない．

そこで本研究では， Web サービスを用いたワークフローの障害対策について検討する．本稿では Web サービスを用い，単一故障に耐えられる信頼性の高いワークフローを実現するための方法を提案する．

ワークフロー管理の実現方法については，

（１） 集中管理手法

（２） 分散管理手法

（３） エージェントによる手法

などがある．分散管理によって複数 Web サービス間のトランザクションを実現する研究 [10] などがあるが，ここでは耐障害性，トランザクション処理を行うことが容易であることから，ジョブ管理部による集中管理手法を用いる．そして，アクティビティ実行ノードの冗長化とジョブ管理部のバックアップを用いた耐故障構成とそれを Web サービスを用いて実現する方法を提案する．具体例として，我々が以前より提案している UPRISE(Unified Presentation Slides Retrieval by Impression Search Engine) [11], [12] という，プレゼンテーションと動画コンテンツの統合と高度な検索機能を実現するシステムのコンテンツ処理プロセスに適用し，実験システムを試作する．

以下では，まず障害を考慮したワークフローについて説明する．その中で，ワークフローでのジョブの管理，耐故障構成，処理の流れ，Web サービスの利用方法を説明する．次に実験システムの対象である UPRISE のコンテンツ処理プロセスについて説明する．そして，それらを踏まえて実験システムの構成，処理の流れを説明し，提案手法が実現可能であることを示す．最後にまとめと今後の課題を述べる．

2. 障害を考慮したワークフロー

ここでは，はじめにワークフローの実現方法を説明し，それを基に本研究で提案する耐故障ワークフローの実現方法について，アクティビティ実行ノードの冗長化とバックアップジョブ管理部について述べる．

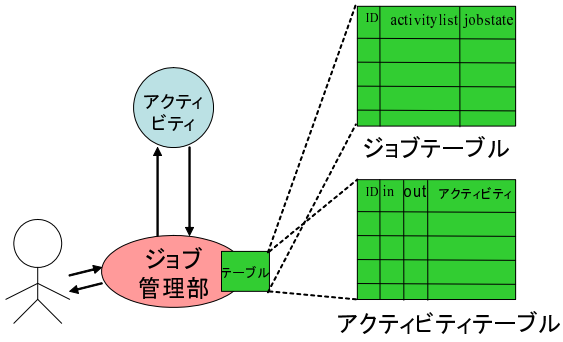

図１　ジョブの管理

2.1 ワークフローでのジョブ管理

ワークフローを実現するために，まず，テーブルを用いたジョブの管理について説明する (図 1)．ジョブをワークフローで処理する前に，まずはジョブにユニークな ID を割り振る．ID については，ID 発行を行うアクティビティを別に用意する．ID が重複しないように ID を決定することは，整数をインクリメントして割り振るなどの簡単な方法で十分である．ID を割り振った後，ユーザがジョブ管理部に ID を添えてジョブを送信する．そして，ジョブの進行状況 (どの処理をどこで行っているか，どのファイルが作成されたか) を 2 種類のテーブルで管理する．一つは，ジョブに対するテーブルであり，ジョブがどのアクティビティをどういう順番で行うかが書かれている．もう一つは，ジョブに対して行うアクティビティの為のテーブルであり，各アクティビティでの入出力と処理状況が書かれている．テーブルはジョブ管理部が管理し，テーブルでジョブの進行状況を見て，アクティビティの呼び出しなどの次の行動を決定する．

ワークフローの実行は以下のように行われる．

- ジョブ管理部はジョブを受け取ったとき，ジョブテーブルにジョブのエントリを書き込み，ジョブテーブルの内容を基にアクティビティのエントリを，実行に必要な環境情報と共に，アクティビティテーブルに書き込む．
- アクティビティの実行条件が整った場合，ジョブ管理部はそのアクティビティを呼び出し，処理状況を実行中とする．
- アクティビティの実行が終了した時，ジョブ管理部はそのアクティビティの処理状況を終了とし，次に実行するアクティビティの処理状況を更新する．
- これらを繰り返すことで各ジョブを実行する．

アクティビティ実行ノードの呼び出しに失敗した場合は，失敗した旨をテーブルに書き込むか，もしくはテーブルを変更しないかのどちらかを行う．そして，再びジョブ管理部がテーブルを見たときにそのアクティビティの再実行を行う (もしくは，失敗を返す)．しかし，アクティビティの故障が長期に渡る場合には，ジョブがそこで止まってしまう．そこで，アクティビ

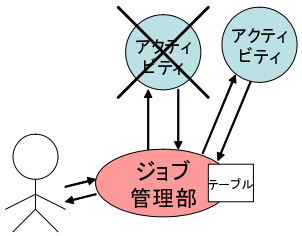

図 2　アクティビティ実行ノードの冗長化

ティが故障したときでも処理が続行するような構成にする必要がある．

2.2　耐故障構成

ここでは，アクティビティ実行ノードの冗長化やジョブ管理部のバックアップを用いて障害を考慮したワークフローを説明する．前述したように，ワークフローでのジョブの管理はジョブ管理部がワークフローで処理を行うジョブに対し適切な処理アクティビティを呼び出すことで，処理が行われる．

2.2.1　アクティビティ実行ノードの冗長化

アクティビティ実行ノードの故障による処理の中断を回避する方法を説明する．単純な処理を行うアクティビティ実行ノードに関しては，アクティビティ実行ノードの冗長化を行う．図 2 はアクティビティ実行ノードを冗長化したときの処理の流れについて示している．ジョブ管理部がアクティビティの呼び出しに失敗した場合にそこで処理が終了しないようにする．そこで，そのアクティビティ実行ノードを複数用意する．そして，同じ処理を行う別のアクティビティ実行ノードを呼び出すことで処理を続けることができる．アクティビティ実行ノードを選択する際は，アクティビティ実行ノードを登録・検索するネームサービスを用意し，処理を行うアクティビティ実行ノードを検索し，検索結果からアクティビティ実行ノードを選択する．その際，そのアクティビティ実行ノードの負荷を尋ね，低負荷のアクティビティ実行ノードを選択する．

一方，ID 発行アクティビティ実行ノードに関しては，バックアップ実行ノードを用いた故障の回避が必要である．ID 発行アクティビティはユニークな ID を割り振るため，単純に冗長化を行う場合，冗長化したそれぞれでの ID の統一が非常に困難な事になるからである．そこで ID 発行アクティビティ実行ノードは次に発行する ID をバックアップ実行ノードに伝える．ユーザが ID を取得するときに ID 発行アクティビティ実行ノードが故障していた時は，そのバックアップ実行ノードを呼び出す．

2.2.2　バックアップジョブ管理部によるバックアップ

また，ジョブ管理部の故障にも耐えるために，ジョブ管理部のバックアップ (バックアップジョブ管理部) を用意する．図 3 は，そのときの動作の流れである．ジョブ管理部は自分のテーブルを変更した後に，バックアップジョブ管理部が持つテーブルも変更する．バックアップはジョブ管理部を監視し，ジョブ管理部が故障した場合バックアップジョブ管理部がジョブ管理部の処理を引き継ぐ．そのためにバックアップジョブ管理部は各アクティビティ実行ノードにジョブ管理部が変わったことを伝える．また，アクティビティ実行ノードが処理を終えた際に，ジョブ管理部のテーブルを変更することができなかった場合，バックアップジョブ管理部のテーブルを書き換えるようにする．

アクティビティの冗長化と違い，ジョブ管理部におけるバックアップジョブ管理部は，ジョブ管理の負荷を分散することまではできない．ジョブを分散するためにはジョブ管理部自体を複数用意する必要がある．バックアップジョブ管理部も同数用意する．このようにすることで，負荷分散を行うことができる．ジョブ管理部の選択は，クライアントプログラムに共通のハッシュ関数 (例えば，$h(ID) = ID \bmod n$：n は自然数) を持たせ，ID にそのハッシュ関数を適用することで，決定する．また，この時のバックアップジョブ管理部の選択方法は，ハッシュ関数 ($h(ID) = (ID + k) \bmod n$：n，k は自然数) とし，ジョブ管理部を実行するノードとバックアップジョブ管理部を実行するノードが同一マシン内に配置されて，マシンダウンにより同時に故障することがないようにする．

アクティビティ実行ノードやジョブ管理部の冗長化の際には，ノードの配置に注意をする必要がある．これについての詳しい説明は，「Web サービスの配置」の節で行う．

2.3　障害を考慮したワークフローの動作

これまでの手法をまとめてワークフロー全体の流れを説明する．図 4 はアクティビティA とアクティビティB を行うワークフローである．まず，ユーザは ID 発行アクティビティ実行ノードでユニークな ID を取得する．ID を取得後，そのハッシュ値 $h(ID)$ により決定されるジョブ管理部にジョブと ID を渡す．

ジョブ管理部は，バックアップジョブ管理部のテーブルにジョブの ID とその現在処理中のノードの情報と ID を書き込み，

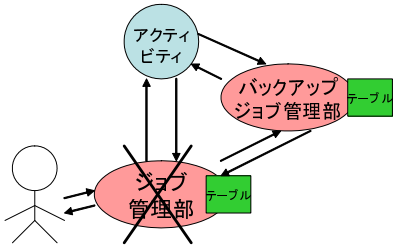

図 3　バックアップジョブ管理部

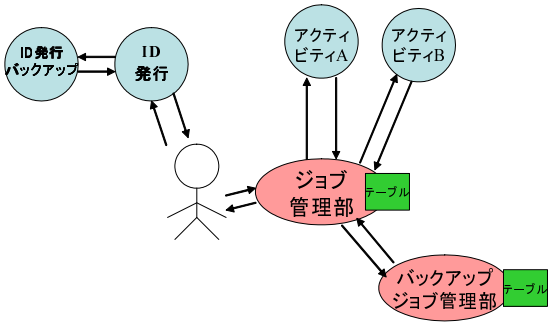

図 4　耐故障ワークフロー

ジョブで次に行われる処理を担当するアクティビティA 実行ノードをネームサービス実行ノードを介して決定し，そこにジョブと ID を送る．ジョブ管理部は，どこにジョブを送ったかについての情報をテーブルに保持し，バックアップジョブ管理部のテーブルも変更する．

アクティビティA 実行ノードでのジョブの処理が終了した時，終了報告と出力結果 (ファイルのアドレスなど) をジョブ管理部のテーブルを変更することで伝える．ジョブ管理部はバックアップジョブ管理部のテーブルも変更する．その後，ジョブ管理部は次のアクティビティB 実行ノードを決定し，同様の動作を行う．アクティビティB 実行ノードでのジョブの処理が終了し，アクティビティB 実行ノードがジョブ管理部のテーブルを変更する．ジョブ管理部はテーブルを見て，すべてのアクティビティが終了したとき，ユーザに終了報告をする．

3. Web サービスを用いたワークフロー

3.1 Web サービスによる実現

ワークフローを Web サービスにより実現することの利点は，個々の Web サービスを連携したものは，また 1 つの大きな Web サービスとなるため拡張性があることと，アクティビティの組み換えが簡潔な点にある．また，異なる部署や会社にまたがるワークフローも可能となる．

前述したワークフローを Web サービスで実現するために，各ノードを 1 つの Web サービスとして実現する．その際，考慮に入れる点は，処理時間の長いアクティビティや，高負荷のアクティビティについて，サーバ側でタイムアウトになったり，接続が切断される可能性がある．そこで，非同期に処理を行う必要がある．

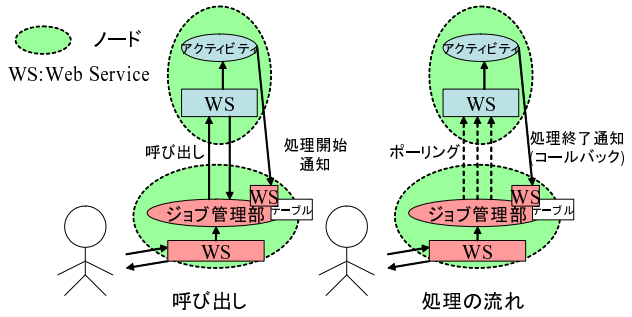

図 5　Web サービスによるアクティビティの実行

図 5，図 6 は，Web サービスを通じたアクティビティの呼び出しを表している．まず，ユーザから Web サービスを通じてジョブ管理部にジョブを送信する．ジョブ管理部はアクティビティ実行 Web サービスを呼び出し，その Web サービスは実際の処理 (アクティビティ) を起動する．この際 Web サービスはアクティビティが終了するまで待機するのではなく，呼び出しの可否のみを返し，アクティビティは非同期に実行される．アクティビティはジョブ管理 Web サービスに処理を開始した旨を通知する．ジョブ管理部は処理中の Web サービスの実行状態をポーリングすることで監視する．実行されたアクティビティ

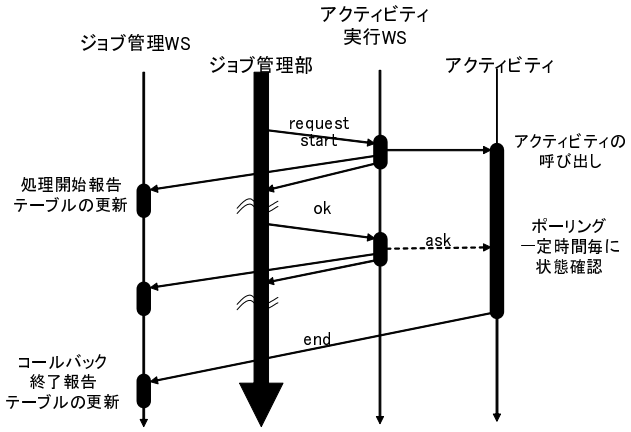

図 6　Web サービスによるアクティビティの実行の流れ

は処理が終了する際に，ジョブ管理 Web サービスにコールバックしテーブルを変更する．ジョブ管理部が処理を行う Web サービスの呼び出しに失敗した場合，同じアクティビティを実行する別の Web サービスを呼び出し，ポーリングで故障を検出した場合も同じアクティビティを実行する別の Web サービスを呼び出す．また，ジョブ管理部が故障した場合，バックアップジョブ管理部が代わりにジョブ管理部の働きをする．

3.2 ワークフローでのトランザクション処理

ワークフローでは，複数の処理を行うため，ある ID を持ったジョブの処理の結果が整合性の取れたものとなっているか，正確に把握する必要がある．そこで，トランザクション処理が必要になる．また，処理にかかる時間も長いことから，長時間トランザクションを実現する必要がある．本稿では，入れ子トランザクションを用い，モデルとして入れ子トランザクションを Web サービスに適用した WS-SAGA [10] を想定する．

また，Web サービスはステートレスであるため，本稿の提案手法では，ジョブ管理部／バックアップジョブ管理部のテーブルの内容によってジョブの状態を管理することでワークフローを実現する．そして，ジョブ管理部／バックアップジョブ管理部のテーブルを利用し，前述したような長時間トランザクションを実現する．

3.3 Web サービスの配置

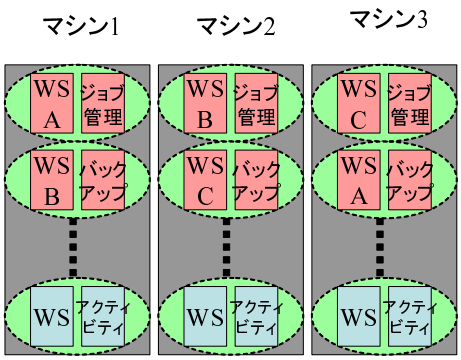

図 7　マシン内の構成

ここでは，Web サービスをマシンに配置するときの割り振り方について説明する．耐故障性を考えてアクティビティ実行ノードの冗長化を行うため，1 つのアクティビティを実装する Web サービスを複数作成し，その Web サービスを複数のマシンに配置する．1 台のマシンに 1 つの Web サービスを配置することはリソースコストが高く効率が悪いため，1 台のマシンに複数の Web サービスを配置する．その際，ジョブ管理部を実行する Web サービスとそのバックアップジョブ管理部を実行する Web サービスは，同時に故障することを避けるために，同じマシンに配置しないようにする．例えば，図 7 のように，管理 Web サービス A はマシン 1 にあるが，そのバックアップジョブ管理 Web サービス A はマシン 3 にあるようにする．この条件を守り，さらにアクティビティ実行ノードに関しても，あるアクティビティを実行するすべてのノードが同一マシン内に配置されることがないように，そして，できる限り特定マシン内に重複して配置されないようにする．

3.4 故障時の動作

障害については，アクティビティを実行しているプロセスの故障と Web サービス側の故障について考える．障害時の動作について以下で障害箇所別にまとめた．マシンの故障については前述した Web サービスの配置を行うことで対応する．

ID 発行 Web サービス ID 発行アクティビティは処理が ID を与えるのみであるため，非同期処理にはしない．そのため，Web サービスの故障のみについて考える．ID 発行アクティビティ実行 Web サービスが故障しているときは，ユーザはバックアップ ID 発行 Web サービスを呼び出して，ID を取得する．

ID 発行 Web サービスが処理中に故障した場合，ID 発行バックアップ Web サービスが異なる値を持っている可能性がある．例えば，ID 発行 Web サービスが ID10 を返し，その後 ID 発行バックアップ Web サービスに報告する前に故障した場合，ID 発行バックアップは次に与える ID を 10 だと認識してしまう．このとき，次に ID 発行バックアップ Web サービスを呼び出したとき，今与えた ID10 をもう一度与えてしまう．このため，ID 発行バックアップ Web サービスを 2 つ (A，B) 用意する．そして，ID 発行 Web サービスがバックアップ A には ID を返す前に，バックアップ B には ID を返した後に通知を行う．その場合，A の方には ID 発行 Web サービスから通知がきているため，2 つのバックアップの間で A と同じ状態になるように ID の情報の同期を行う．

ID 発行バックアップ Web サービス そのまま続行する．

ジョブ管理 Web サービス(Web サービス故障) バック アップジョブ管理 Web サービスが尋ねにきたときに呼び出しに失敗するため，バックアップジョブ管理が代わりにジョブ管理の処理を行う．

また，ジョブ管理 Web サービスが自分のテーブルを変更し，バックアップジョブ管理を変更する前に故障したときについて考える．例えば，アクティビティA から終了報告が返ってきて，ジョブ管理部のテーブルを変更した後で故障したとする．このとき，バックアップジョブ管理 Web サービスが代わりにジョブ管理を行うが，A からの終了報告がないため，A はまだ実行中であると認識する．そして，ポーリングするが，実際はすでに処理を終えているため，尋ねるプロセスが存在しない．この場合，処理の再試行を行う．

ジョブ管理 Web サービス(ジョブ管理部故障) アクティビティでの処理が終了し，ジョブ管理部に報告して返事がない場合は，バックアップジョブ管理部に報告する．その後，バックアップジョブ管理部は自分の場所を他のアクティビティに通知し，バックアップジョブ管理部がジョブ管理部の代役を務める．また，バックアップジョブ管理 Web サービスが尋ねにきたとき，ジョブ管理部が応答しなかった場合についてもバックアップジョブ管理部がジョブ管理部の代役を務める (図 8)．また，ジョブの登録の時点で故障している場合も，バックアップジョブ管理部に報告し，同様の動作を行う．

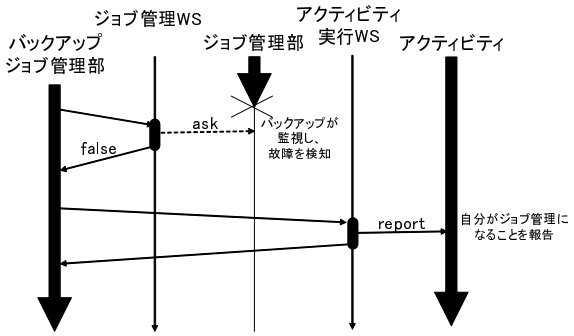

図 8 ジョブ管理部故障時の処理の流れ

バックアップジョブ管理 Web サービス 別のジョブ管理部のバックアップジョブ管理部のテーブルにジョブ管理部が保持しているテーブルの内容を書き加える．バックアップジョブ管理部はそのジョブ管理部の監視も行う (図 9)．例えば，バックアップジョブ管理部 A が故障した場合を考える．ジョブ管理部 A はテーブルの情報をバックアップジョブ管理部 B のテーブルに加える．そして，バックアップジョブ管理部 B がジョブ管理部 A，ジョブ管理部 B を監視する．

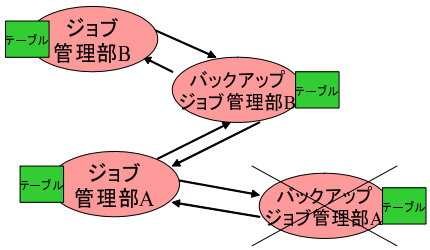

図 9 バックアップの故障

処理アクティビティ実行 Web サービス (Web サービス故障) 呼び出しの時点で故障している場合は，同じアクティビティの処理をする別のアクティビティ実行 Web サービス (例，処理 B で失敗したら処理 B ’に渡す等) を呼び出す．また，処理中に故障した場合は，ジョブ管理部が再試行の要請をする．処理を行っているプロセスが故障していない場合，重複実行になる．どこで処理を行っているかをテーブルから読み取り，再試行した方にのみ処理の後のテーブルの書き換えを許す．

処理アクティビティ実行 Web サービス (プロセス故障)　非同期で処理しているプロセス (アクティビティ) が故障した場合，呼び出しのときは，別のアクティビティ実行 Web サービスを呼び出し，処理中に故障したときは，ジョブ管理部のポーリングでの応答が false になるため，それを確認したときそのプロセスを強制終了させ，再試行させる (図 10).

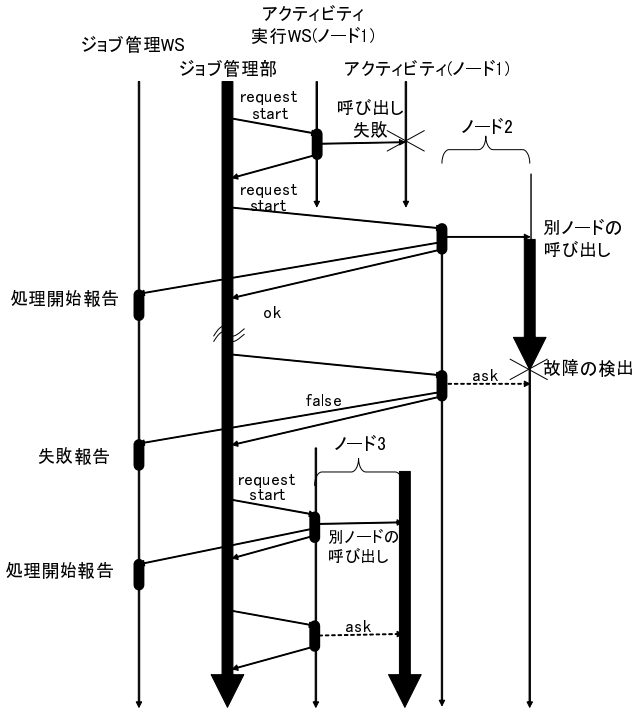

図 10　アクティビティ実行ノード故障時の処理の流れ

データの損失　そのデータを出力するアクティビティの処理をもう一度行う (図 11). あるアクティビティ A での処理が終わって，出力されたデータが損失する場合がある．その場合は，次の処理がそのデータを取得する際にデータの有無を調べ，データが損失している場合，そのことをジョブ管理部に報告し，ジョブ管理部は再びアクティビティ A 実行 Web サービスを行う.

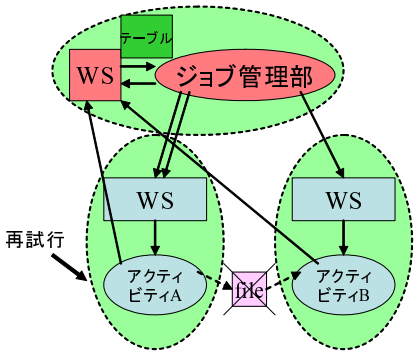

図 11　データ損失

4.　UPRISE のコンテンツ処理プロセス

2 節で説明した障害を考慮したワークフローを実現する際の

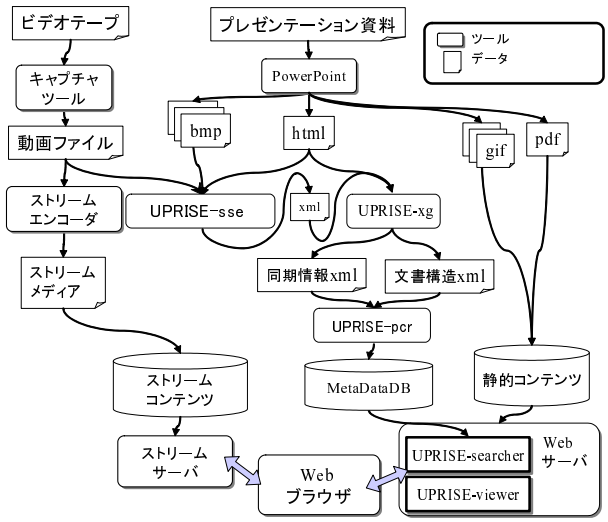

図 12　UPRISE のコンテンツ処理プロセス

適用例である UPRISE のコンテンツ処理プロセスについて説明する．UPRISE では，ビデオテープとプレゼンテーション資料からそれぞれストリームコンテンツ，静的コンテンツを作成する．一方で，ビデオテープとプレゼンテーション資料の同期情報を作成し，メタデータのデータベースに格納する．図 3 は，UPRISE のコンテンツ処理プロセスである [13].

以下に，各処理について，その処理内容を述べ，Web サービスの適用箇所について説明する.

キャプチャ　ビデオテープから動画 (avi) をキャプチャツール (AdobePremiere，RealProducer 等) でキャプチャする.

ストリームエンコード　動画ファイル (avi) をストリームエンコーダ (AdobePremiere，RealProducer 等) で RealMedia 形式にする.

ファイル変換 (PowerPoint)　プレゼンテーション資料 (ppt ファイル) を PowerPoint の「形式を指定して保存」を使用して，bmp ファイル，html ファイル，pdf ファイル，gif ファイルに変換する.

同期情報作成 (uprise-sse)　avi ファイル，bmp ファイル，html ファイルから同期情報 XML を作成する．ツールには，富士通研究所との共同研究で開発している movie-recog [14] を利用している.

XML 変換 (uprise-xg)　同期情報 XML ファイルを登録用同期情報 XML ファイルに，html ファイルを登録用文書構造 XML ファイルに変換する.

格納 (uprise-pcr)　同期情報 XML ファイルと文書構造 XML ファイルをメタデータのデータベースに格納する.

4.1　作業の流れ

まず，ビデオテープをキャプチャし，動画ファイル (avi ファイル) に変換し，ストリームエンコーダで RealMedia 形式のファイルを作成し，データベースに格納する.

一方，プレゼンテーション資料 (ppt ファイル) を PowerPoint で bmp ファイル，html ファイル，gif ファイル，pdf ファイルに変換する．その中のうち，gif ファイル，pdf ファイルは静的

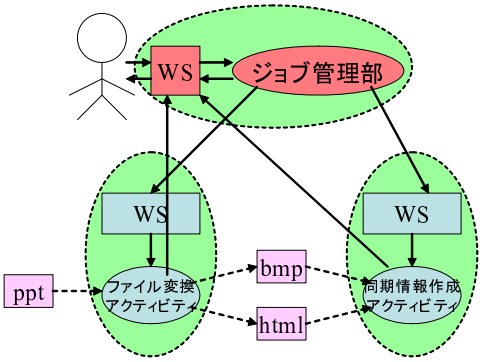

図 13　実験システム (コンテンツ処理システム) の構成

コンテンツとして格納する．

次に，avi ファイル，bmp ファイル，html ファイルを使って uprise-sse で同期情報 XML ファイルを作成する．同期情報 XML ファイルと html ファイルはそれぞれ，uprise-xg で登録用の同期情報 XML ファイル，文書構造 XML ファイルに変換する．そして，それらを uprise-pcr でメタデータのデータベースに格納する．

本稿では，ビデオテープとプレゼンテーション資料を入力して，同期情報 XML を出力するまでの部分を実験システムに適用する．キャプチャについてはテープを手動で挿入する作業があるため，この部分については今回の実験では，適用しない．この部分を除いて PowerPoint 部分と同期情報作成部分の自動化を行う．この部分は処理が連続しており，処理と処理の間の手間が省け，全体の処理がスムーズになることが期待できる．

5.　実験システム

提案手法を評価するために実験システムの試作をし，その動作を確認した．実験システムには，統合プレゼンテーション登録ワークフローである 4 節で説明した UPRISE のコンテンツ処理ワークフローを適用した．実装は *C#.NET* で行った．

作成した Web サービスはジョブ管理 Web サービス，ID 発行 Web サービス，ファイル変換 Web サービス，同期情報作成 Web サービスである．図 13 は，Web サービス全体の構成を簡潔に示したものである．

5.1　構　　成

この実験システム (図 13) では，ファイル変換アクティビティと同期情報作成アクティビティを実行する Web サービスを作成した．そして，この連続した 2 つのアクティビティをジョブ管理 Web サービスによって自動実行することを可能にした．

表 1，表 2 がジョブ管理部が持っているテーブルの情報であり，本システムではこの情報を XML ファイルとして保持する．

ジョブ管理部はジョブテーブルの activitylist を見て，どのアクティビティをどの順番で行うか，また，そのアクティビティがどのノードで実行されるかとその入出力ファイルについて把握する．そして，アクティビティの数だけアクティビティテーブルにエントリを追加する．

またジョブ管理部はアクティビティテーブルの情報を見て，Web サービスの呼び出しや，状態確認などを行う．

今回の実験システムでは，activitylist はファイル変換アクティビティと同期情報作成アクティビティの 2 つのアクティビティに固定し，その処理順序も既知のこととし，ジョブに対してアクティビティのリストを記述する部分については今後の課題とする．

5.2　実験システムの動作

以下の流れで実験システムが動作することを確認した．この枠組みを用いることで，UPRISE コンテンツ処理ワークフローだけでなく，他のワークフローについても適用可能である．

5.2.1　通 常 処 理

通常処理は以下のように行われる．

- まず，ジョブ管理 Web サービスのテーブルの初期化のための initialize メソッドによって，ジョブテーブルとアクティビティテーブルが作成される．
- 同時に，アクティビティテーブルを一定時間ごとに監視するジョブチェックメソッドが非同期で呼び出される．
- ユーザは ID 発行 Web サービスを呼び出して，ID を取得する．
- 取得した ID と ppt ファイルと avi ファイルを入力とし，ジョブ管理 Web サービスの登録メソッドを呼び出すことで，ジョブの登録を行う．
- 登録メソッドでは，アクティビティテーブルにそのジョブに対するファイル変換アクティビティと同期情報作成アクティビティのエントリが作成され，その際，アクティビティテーブルの activityID，activityname，activitystate，nextactivity などが書き込まれる．
- ジョブチェックメソッドがアクティビティテーブルを見て activitystate が ready になっているアクティビティを実行する．この場合，最初に実行するファイル変換アクティビティの状態が ready になっているため，それを実行する．実行する際には，inputlist のファイルの存在を確かめて，inputlist のファイルのパス名を入力値としてファイル変換アクティビティを実行する Web サービスのアクティビティ呼び出しメソッドを実行する．
- アクティビティ呼び出しメソッドはアクティビティの処理を行うスレッドを作成して，アクティビティを非同期に実行する．また，そのスレッドが処理を開始した際に，ジョブ

表 1　テーブルの構造 1

ジョブテーブル	
ID	ID 発行 Web サービスで取得したジョブの ID.
activitylist	そのジョブで行うアクティビティのリスト．そのアクティビティの前後のアクティビティ，入出力データ，そのアクティビティを実行できるノードについても書かれている．
jobstate	そのジョブの全体的な処理状況．全部の処理が終わったとき，commit となる．未終了時は abort.

表 2　テーブルの構造 2

アクティビティテーブル	
ID	ID 発行 Web サービスで取得したジョブの ID.
activityID	エントリする際にジョブテーブルでローカルに決めたアクティビティの ID.
inputlist	そのアクティビティで必要となる入力ファイルのパス名のリスト.
outputlist	そのアクティビティで必要となる出力ファイルのパス名のリスト.
activitytype	行うアクティビティの名前.
activitystate	そのジョブの各アクティビティでの処理状況を表す. waiting, ready, executing, finish がある. waiting では，どの activityID のアクティビティを待っているかについても記述される.
node	どのアクティビティ実行ノードで処理しているかについて書かれている.
pid	アクティビティのプロセス ID.
nextactivity	次に行うアクティビティの activityID のリスト. 分岐の場合は複数ある.
count	ジョブ管理 Web サービスがそのジョブを何回監視したかを示すものである. ある数値を超えた場合，タイムアウトとする.

管理 Web サービスのテーブル書き換えメソッドを呼び出し，activitystate の要素を executing に書き換え，そのスレッドが呼び出したプロセス ID を pid の要素に書き込む.

- スレッドの処理が終了したとき，スレッドはジョブ管理 Web サービスのテーブル書き換えメソッドを呼び出して，activitystate 要素を finish に書き換え，出力ファイルのパス名を outputlist に書き込む. そして，この outputlist の中身を，同じジョブで activitytype が同期情報作成アクティビティとなっているところの inputlist にも書き込む.
- nextactivity に書いてあるアクティビティの activitystate 要素を ready に変更する.
- ジョブチェックメソッドがアクティビティテーブルを確認した際に，activitystate が ready になっている同期情報アクティビティを実行する. ファイルが揃っているとき，同期情報作成アクティビティを実行する Web サービスのアクティビティ呼び出しメソッドを入力ファイルのパス名を入力値として呼び出し，同様の処理を行う.
- ジョブチェックメソッドはすべての activity 要素が finish になっているとき，jobstate 要素を commit にし，処理を終了する.

5.2.2　アクティビティ実行ノードの故障時の処理

- 呼び出しに失敗した場合，別のノードを呼び出すか，別のノードがなかった場合，ユーザに失敗を返す.
- アクティビティでの処理中，ジョブ管理 Web サービスのジョブチェックメソッドは，executing となっている処理について，pid を入力値としたアクティビティ実行 Web サービスの応答メソッドを呼び出し，そのスレッドの呼び出したプロセス状態を true か false で返す. false の場合，その呼び出したプロセスを強制終了させる. そして，そのアクティビティの activity 要素を ready に戻し，再試行する.
- 一定回数ポーリングしても処理が終了しない場合，強制終了させて再試行する.

6.　まとめと今後の課題

6.1　ま　と　め

本稿では，Web サービスを用いて耐故障性を考慮したワークフローの実現方法を提案した. 具体的には，テーブルを用いたジョブ管理によりワークフローを実現し，アクティビティ実行ノードの冗長化によりアクティビティ実行ノードの故障に対応し，バックアップのテーブルを更新することによりジョブ管理部の故障に対応する方法を示した. また，Web サービスを用いてワークフローのジョブ管理を実現するために，ポーリングとコールバックを組み合わせたジョブ管理を提案した. そして，実験システムを試作した. その際，適用例として，UPRISE のコンテンツ処理ワークフローを用い，通常処理とアクティビティ実行ノードの故障時の処理を実現した.

6.2　今後の課題

今後の課題としては，まず提案手法の未実装部分であるジョブ管理部の耐故障の実装が挙げられる. また，提案手法の故障の種類別の対応やバックアップジョブ管理部のテーブルへのデータの変更についての有効性・信頼性の詳細な検証を行う予定である. 次に，複数個所の同時故障や，1 個所故障しているときにさらに故障することに耐えることができる手法を提案する. そして，提案手法を別のワークフローに適用するなどの適用範囲の拡大や，既存のワークフローとの連携を図る. また，関連研究の参照や提案手法の他手法との比較・検証も行う必要がある. さらには，障害対策だけではなく，待ち合わせ手法などの提案を行うことも重要である.

7.　謝　　辞

本研究の一部は，文部科学省科学研究費補助金特定領域研究 (15017233)，独立行政法人科学技術振興機構 CREST，および 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた.

文　　献

[1] Workflow Management Coalition. http://www.wfmc.org/.
[2] 株式会社日本ユニテック DigitalXpress 編集部. SOAP UDDI WSDL Web サービス技術基礎と実践 徹底解説. 技術評論社, 2002.
[3] SOAP1.1 に関する W3C の技術ノート. http://www.w3.org/TR/SOAP/.
[4] WSDL1.1 に関する技術ノート. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
[5] UDDI. http://www.uddi.org/.
[6] BPEL4WS. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.
[7] WS-AtomicTransaction. http://www-106.ibm.com/developerworks/library/ws-atomtran/.
[8] WS-Coordination. http://www-106.ibm.com/developerworks/library/ws-coor/.
[9] WS-CAF. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf.
[10] Neila Ben Lakhal, Takashi Kobayashi, and Haruo Yokota. Distributed architecture for reliable execution of web services.

Dbs-131-17, 情報処理学会 研究報告, 7 2003.

[11] 村木太一, 吉田誠, 小林隆志, 直井聡, 横田治夫. メタデータによる講演資料と動画の統合と検索. In *Proc. of DBWeb*. 情報処理学会.

[12] 村木太一, 吉田誠, 小林隆志, 直井聡, 横田 治夫. メタデータによる教育資料の統合における検索絞り込み指標の評価. Issn 1347-4413, DEWS2003, 5-c, 電子情報通信学会データ工学ワークショップ, 3 2003.

[13] 小林隆志, 村木太一, 直井聡, 横田治夫. 総合プレゼンテーションコンテンツ蓄積検索システムの試作. In *Proc. of DBWeb*. 情報処理学会, 2003.

[14] 小澤憲秋, 武部浩明, 勝山裕, 直井聡, 横田治夫. 文字認識を利用した講義動画中のスライド固定. 第一回 情報科学技術フォーラム (FIT), LI-5. 情報処理学会／電子情報通信学会, 2002.