Web サービスにおけるトランザクションの隔離性

Isolation Property of Web Service Composition Transactions

徐 海燕☆ 古川 哲也☆☆ 史 一華☆☆☆

Haiyan XU Tetsuya FURUKAWA Yihua SHI

Web サービスの普及に伴って、インターネット上で公開された様々な Web サービスを構成部品として利用し、企業の情報システムを組み立てるという時代が到来する。このため、複合 Web サービスのトランザクションが並行実行時に互いに与えうる影響について検討する必要がある。本論文では、複合 Web サービスに従って単独で実行された各トランザクションが満たさなければならない性質を一貫性、並行実行時に同じ性質を満たすための性質を隔離性と定義し、隔離性を保証するための ACID 属性よりも緩められた条件を提案する。また、BPEL4WS のようなフローエンジンが他のサービスを呼び出す場合における隔離性の実現法について検討する。

Web services have quickly become a very important technology for enterprise system development and it has become possible to build distributed systems using web services as components. This paper discusses the isolation property of web service composition transactions. By discussing the data interaction between different instances, we propose a condition to guarantee the isolation property in web service composition transactions, which is weaker than the conventional serializablity property. The mechanism for the isolation is also discussed.

1. はじめに

ビジネス取引の自動化を目指すために、Web サービスについての研究が盛んに行われている。複数の Web サービスをインターネット上で統合して新たな Web サービスを定義するために、BPEL4WS (Business Process Execution Language for Web Services) が発表された[1]。複合 Web サービスの設計と分析に際しては、どのような複合 Web サービスは正しく動くのか、どのように Web サービスの実行の正しさを保証すればよいのか、などのように、多くの課題が残されている[4]。本論文では、Web サービスの密結合によって企業内の情報システムを組み立てる際に必要となるトランザクションの隔離性の管理について検討し、一般的な疎結合の場合まで触れる。

密結合による企業内の情報システムにおいて、複数のインスタンスが同時に実行されるとき、トランザクションの隔離性の管理が必要となる場合がある。

例 1 融資の申込みを処理するプロセスが図 1 のように構築されているとする。顧客の融資申込みを受け付け、その顧客に関する収入／財産／ローンなどの経済状況をデータベースから検索し、金融機関の Web サービスであるローン審査を呼び出してその顧客の融資へのリスクを判断する。融資のリスクが低いと判断された場合は、融資を承認し、ローンの貸出しを行う。それ以外の場合は、融資ができない旨の返信を行う。しかし、ある顧客が複数件の融資申請を同時に行った場合、貸出しできないローンを融資してしまう場合が生じうる。 □

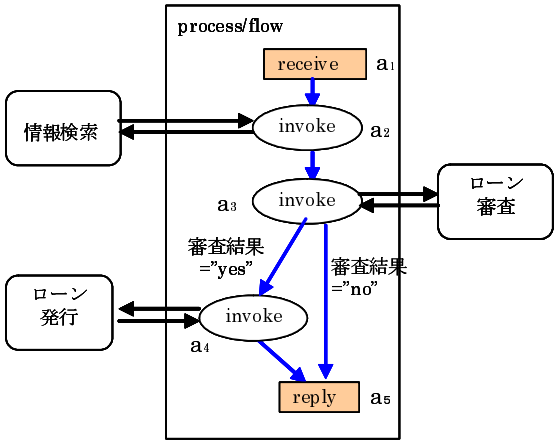

図 1 融資申込みプロセス
Fig. 1 A loan approval process

従来のトランザクション管理は、原子性（Atomicity）、一貫性（Consistency）、隔離性（Isolation）、耐久性（Durability）によって表される ACID 属性に基づいている[2]。しかし、この ACID 属性は Web サービスのような疎結合や密結合がある結合環境においては、柔軟性に欠け、制限が多すぎるという問題が広く認識されている[5]。このため、標準化組織 OASIS による Business Transaction Processing (BTP) や、BEA、IBM、Microsoft による WS-Transaction とその付属仕様である WS-Coordination が公開されている[3),6]。BTP は疎結合システムへの適用を前提としており、WS-Transaction は密結合用の ACID トランザクションと疎結合のための長大トランザクションの両者をサポートしている。これらの仕様に共通しているのは、全て実行されるかまったく実行されないかという後退復帰方式ではなく、ビジネスロジックに合わせた個々の例外処理を行う補償方式を使用することである。しかし、Web サービスの連携によって企業内の情報システムを組み立てるという密結合の場合においても厳密な議論はなされておらず、ACID 属性が必ずしも適しているとは限らない。

従来の企業におけるワークフローシステムにおいて、トランザクションの隔離性が内部設計によって実現されている場合が多い。しかし、インターネット上で公開している Web サービスを利用する場合に入手できるのはインタフェースに関する情報のみで、内部ロジックに頼ることは現実的でない。本論文では、Web サービスにおける基本的な操作単位であるアクティビティの入出力データの種類について分析し、トランザクションにおける一貫性上で関連するデータ項目の集合を明確にする。それに基づいて Web サービストランザクションにおける隔離性は ACID 属性よりも緩められることを示す。

本論文は、次のように構成される。2 節で Web サービスとその連携であるフローモデルを定義し、3 節では密結合と疎結合の場合に分けて、Web サービスのトランザクションの隔離性の正当性基準について検討する。4 節では BPEL4WS エンジンのようなフローエンジンとデータベースの連携による隔離性の実

☆ 正会員 福岡工業大学情報工学部情報工学科
xu@cs.fit.ac.jp
☆☆ 正会員 九州大学大学院経済学研究院
furukawa@en.kyushu-u.ac.jp
☆☆☆ 正会員 西南学院大学商学部
shi@seinan-gu.ac.jp

現方法を、施錠方式の立場から提案する。5 節は全体のまとめである。

2. Web サービスモデル

Web サービスは XML フォーマットのメッセージデータを交換する。使用されるメッセージのデータフォーマットや、メッセージの処理ルールは、通信規約 SOAP によって定められている。Web サービスのインタフェースは、WSDL によって記述される。複数の Web サービスを連携して新たな複合 Web サービスを作成する際は、パートナーによって提供される Web サービスを呼び出す $< invoke >$、パートナーからのメッセージを受け取る $< receive >$、パートナーへメッセージを送信する $< reply >$、大域変数の引き渡しを行う $< assign >$ が基本的なアクティビティになる。

個々の基本的なアクティビティのインタフェースを定義する WSDL には、名前、パートナー、ポートタイプ、操作、入出力データといったパラメータについての記述が含まれている。本論文では、基本的なアクティビティ a を $a(I,O,p)$ として抽象化して記述する。ここで、I は a によって検索される大域変数、O は a によって値が代入される大域変数である。例えば、基本的なアクティビティでは、$< receive >$ は出力データのみを、$< reply >$ は入力データのみを持つ。$< invoke >$ と $< assign >$ は入力データと出力データの両方を含む。また、p は a のパートナーを記述しているが、パートナーが重要でない場合は省略する。

I と O の和集合をアクティビティ $a(I,O,p)$ の操作データ集合という。実際の場合には、利用される Web サービスによってインタフェースの粒度が異なってくる。一般的に企業内ではメッセージが必要なビジネス情報を全て含むような粒度の細かいインタフェース、外部利用者に対してはキーワード情報しか含まない粒度の粗いインタフェースが使われている。本論文では、企業内の連携においては、必要とする全ての入出力情報をインタフェースに含むとする。

例 2 図 1 には、a_1, a_2, a_3, a_4, a_5 の 5 つの基本的なアクティビティがある。それらの記述を次に示す。

- 申請受付 a_1:
  $I_1 = \phi$,
  O_1 =\{ 顧客 ID:x_1, 申請額:x_2\},
- 情報検索 a_2:
  I_2 =\{ 顧客 ID:x_1\},
  O_2 = \{ 経済状況:x_3\},
- 審査 a_3:
  I_3 =\{ 顧客 ID:x_1, 申請額:x_2, 経済状況:x_3\},
  O_3 = \{ 審査結果:x_4\},
- ローン発行 a_4:
  I_4 =\{ 顧客 ID:x_1, 申請額:x_2\},
  O_4=\{ 経済状況:x_3\}
- 結果返信 a_5:
  I_5 =\{ 顧客 ID:x_1, 審査結果:x_4\},
  $O_5 = \phi$.

すなわち、a_1 では顧客の申請を受け付け、顧客 ID や申請額を大域変数に代入している。a_2 では企業データベースに記憶されている申請者の経済状況に関する情報を、a_1 で代入された大域変数 x_1 をキーワード にして検索し、その結果を大域変数 x_3 に代入している。それらによって得られた情報に基づいて、申請者に対する審査を a_3 によって行う。a_4 で申請額のローンを発行するが、申請者の経済状況にも今回のローンの発行を反映する。a_5 では顧客への審査結果の返信を行う。 □

各節点の作業によってデータベースに対しても検索や変更がなされているので、I_{DB} と O_{DB} でそれぞれ各節点によって検索と変更がなされたデータベースの項目を表すとする。例 2 においてデータベースに対する入出力データは次のようになる。

I_{2DB} =\{ 顧客 ID:x_1, 経済状況:x_3\}
I_{4DB} =\{ 顧客 ID:x_1\}
O_{4DB}=\{ 経済状況:x_3\}

データベースからの入力データとデータベースへの出力データに関する情報は、アクティビティのインタフェースからは得られない。本論文では、データベースレベルでの介入によってそれらの情報を入手する方法を導入する。

一方、基本的なアクティビティ間の実行順序は、制御フローによって記述する。BPEL4WS では $< sequence >$, $< flow >$ などの構造化されたアクティビティによって、基本的なアクティビティのインスタンス間の実行順序を定義している（図 1）。

制御フローに従う実行であるトランザクションを記述するために、データ項目 x およびデータ項目集合 X のインスタンスを $\mathcal{I}_x$ および $\mathcal{I}_X$ とし、$\mathcal{A}(\mathcal{I}_I, \mathcal{I}_O, p)$ で基本的なアクティビティ $a(I,O,p)$ の実行を記述する。トランザクションは、基本的なアクティビティの実行を表す節点と分岐節点／結合節点からなる節点集合と、それらの間の実行順序を表す枝集合からなる有向グラフ $WT(TN,TE)$ である。分岐節点に対しては分岐枝は 1 つしか存在せず、真となる経路条件に関わるデータ項目とその値が、分岐節点の出力データとそのインスタンス $\mathcal{I}_O$ となる。トランザクション $WT(TN,TE)$ は、有向非巡回グラフである。これは、複数回実行される基本的なアクティビティは異なる節点に展開され、条件分岐は特定の経路のタスクに従って実行されるためである。

トランザクション $WT(TN,TE)$ におけるデータの流れは、通常データフローと呼ばれる有向グラフ $DT(TN,DE)$ によって記述される。節点 $\mathcal{A}_i$ が $WT(TN,TE)$ において祖先である $\mathcal{A}_j$ の出力データを入力データとして利用するなら、データフロー $DT(TN,DE)$ に $\mathcal{A}_j$ から $\mathcal{A}_i$ への枝がある。分岐節点に対しては、真となる経路条件に関わるデータ項目によって分岐節点から分岐経路上の節点への枝が存在するとする。すなわち、データフロー $DT(TN,DE)$ 中の枝はトランザクション $WT(TN,TE)$ の枝の推移的閉包に含まれる ($DE \subseteq TE^*$)。

異なるトランザクションが同時に実行される場合に、スケジュールが生成される。$WT_i(TN_i,TE_i)$ $(i = 1,2,\cdots)$ からなるスケジュールは、次のような有向非巡回グラフ $WH(HN,HE)$ である。

- $HN = \bigcup_i TN_i$, $HE \supseteq \bigcup_i TE_i$,
- 異なるトランザクションの節点 $\mathcal{A}_s$ と $\mathcal{A}_t$ が競合する（$(O_{sDB} \cap (I_{tDB} \cup O_{tDB}) \neq \phi) \vee (O_{tDB} \cap (I_{sDB} \cup O_{sDB}) \neq \phi)$）なら、 それらの間の実行順序を示す枝か推移的な枝が HE に含まれる。

例 3 図 2 は例 1 の Web サービスに従う実行である WT_1 と WT_2 からなるスケジュール WH_1 を示している。一方、$WT_1(TN_1,TE_1)$ におけるデータフロー $DT_1(TN_1,DE_1)$ には、$\mathcal{A}_2^1(\mathcal{A}_3^1, \mathcal{A}_4^1, \mathcal{A}_5^1)$ の入力データ x_1 は $\mathcal{A}_1^1$ の出力データを利用しているので、枝 $(\mathcal{A}_1^1, \mathcal{A}_2^1)((\mathcal{A}_1^1, \mathcal{A}_3^1), (\mathcal{A}_1^1, \mathcal{A}_4^1), (\mathcal{A}_1^1, \mathcal{A}_5^1))$ を含む。同じように、$\mathcal{A}_3^1$ の入力データ x_3 は $\mathcal{A}_2^1$ の出力データを利用しているので、枝 $(\mathcal{A}_2^1, \mathcal{A}_3^1)$ が存在する。一方、$\mathcal{A}_3^1$ の出力データ x_4 は、経路条件にも $\mathcal{A}_5^1$ の入力データにも利用されたので、枝 $(\mathcal{A}_3^1, \mathcal{A}_4^1)$ と $(\mathcal{A}_3^1, \mathcal{A}_5^1)$ が存在する。 □

3. 隔 離 性

本節では、トランザクションの隔離性を、密結合と疎結合の二つの場合に分けて検討する。

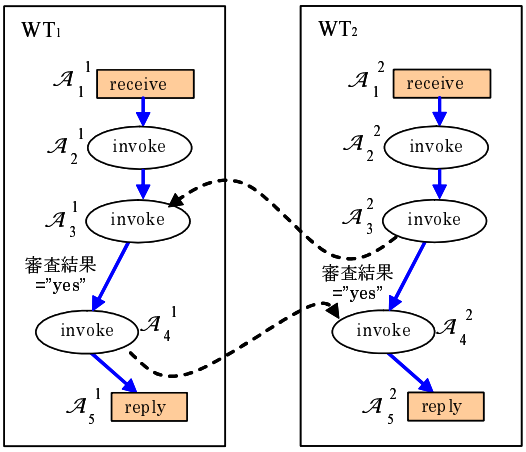

図 2 スケジュール WH_1
Fig. 2 Schedule WH_1

3.1 密結合の場合における隔離性

複合 Web サービスに従う実行であるトランザクションが、複数個同時に実行されるときに単独実行と同様に正当な実行でなければならない。本節では、単独実行時に各トランザクションが満たさなければならない性質を一貫性として形式的に定義し、並行実行時に同じ性質を満たすようにするための隔離性について検討する。

節点の集合であるトランザクションの一貫性は、各節点の一貫性と節点間の一貫性に分けられる。

- 各節点の一貫性は、それぞれの節点の入出力データ間の一貫性を意味する。
- 節点間の一貫性は、各節点とトランザクションのこれまでの実行環境との対応を表す。

基本的なアクティビティは入力データに対して正しく出力データを生成できると仮定するので、原子性の単位となる基本的なアクティビティの実行である節点は、入出力データ間の一貫性を満たす。節点間の一貫性を分析するためには、各節点の作業によってデータベースに対して検索や変更がなされていることを考慮する必要がある。大域変数は並行実行の影響を受けないが、データベースに対する入出力データは例 1 で示したように並行実行の影響を受ける。したがって、大域変数からの入力の場合も、データフローの親の実行環境が並行実行時に受ける影響を再帰的に考慮する必要がある。また、分岐節点は分岐経路上の各節点のデータフロー上の親となるため、分岐条件に使われるデータ項目のデータフロー上の親の実行環境が受けた影響も考慮する必要がある。このため、トランザクションの一貫性は次のように表せる。

定義 1 入出力データの一貫性を満たす個々の節点からなるトランザクションは、トランザクションが単独で実行する場合に各節点が実行されるときにデータフローの祖先節点のデータベースに対する入出力データ間で成り立っている性質を満たしているならば、一貫性を満たしている。 □

並行実行において、各々のトランザクションの節点の大域変数が、他のトランザクションの実行によって変更されることはない。このため、トランザクションの一貫性にそのような入力データとの一貫性を含めていない。一方、WH_1 で示しているように、データベースに対する入出力データはトランザクションの実行中に他のトランザクションによって変更されることによって一貫性上で問題が生じうる。隔離性は、そのような問題に対処するための性質である。

定義 2 単独実行において一貫性を満たすトランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、一貫性を満たさないトランザクションが存在しなければ、各々のトランザクションは隔離性を満たす。 □

WH_1 で示したように、WT_2 によって操作されたデータ項目が他のトランザクションによって変更されたため問題が生じた。このため、問題を引き起こしうるすべての変更を禁止することで対処する。

定義 3 トランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、各々のトランザクションに対して、データベースに対する入出力データが操作されてからデータフローの子孫節点が全て終了するまで他のトランザクションによって変更されていなければ、各々のトランザクションは論理性を満たす。 □

例えば、スケジュール WH_1 においては、$\mathcal{A}_2^2$ のデータベースに対する入出力データ x_3 が、$\mathcal{A}_4^2$ が終了するまで WT_1 中の節点 $\mathcal{A}_4^1$ によって変更されている。このため、WT_2 は論理性を満たさない。

トランザクション WT_i $(i = 1, \cdots, n)$ からなる 2 つのスケジュール WH と WH' で、各トランザクションの実行と最終結果がそれぞれ同じであれば、WH と WH' は等価であるという[2]。従来では、直列実行と等価なものを直列可能スケジュールといい、直列可能性を隔離性の基準としてきた。入出力データの一貫性を満たす個々の節点からなる Web サービストランザクションに対して、隔離性の基準は次のようになる。

定理 1 単独実行において一貫性を満たすトランザクション WT_i $(i = 1, \cdots, n)$ からなるスケジュール $WH(HN, HE)$ において、WH と等価で各 WT_j $(j \in \{1, 2, \cdots, n\})$ が論理性を満たすスケジュール WH' が存在すれば、各 WT_j は一貫性を満たす。

証明： トランザクション $WT_j(TN_j, TE_j)$ が一貫性を満たさないとすると、定義 1 より節点間の一貫性を満たさない節点 $\mathcal{A}$ が存在することになる。すなわち、$\mathcal{A}$ が実行されるときにデータフローの祖先節点のデータベースに対する入出力データはデータベースの一貫性を満たさない。$\mathcal{A}$ のデータフローの祖先節点のデータベースに対する入出力データを DB' とすると、単独実行時に WT は一貫性を満たすため、WT によって操作された DB' 中のデータ項目は $\mathcal{A}$ が実行されるまで他のトランザクションによって変更されたことになる。しかし、これは各トランザクション WT_j $(j \in \{1, 2, \cdots, n\})$ は WH' において論理性を満たすことと矛盾する。 □

論理性を満たす等価なスケジュールが存在する基準は直列可能性よりも緩和した基準であり、施錠方式を用いる場合は、専有施錠と共有施錠間の競合を緩和できることが、論文[7] によって示されている。また、定理 1 により、各トランザクションが一貫性を満たす単位は、トランザクション単位ではなくデータフローの経路単位でよい。通常トランザクションは複数のデータフローの経路に細分されるため、定理 1 の条件は従来のトランザクション単位での ACID 属性よりも緩和されている。

3.2 疎結合の場合における隔離性

疎結合環境におけるトランザクションは、一般的に複数個の密結合環境のサブトランザクションから構成される。$a_i(I_i, O_i, p_i)$ $(i = 1, 2, \cdots, k)$ を基本的なアクティビティとするフローに対して、集合 P に属する企業間の疎結合環境においては、P 中の要素をパートナーとするアクティビティの入出力メッセージだけが残され、P 以外のパートナー間の入出力データに関する情報は利用できないことになる。例えば、例 1 では、a_1 と a_5 の入出力データが残されることになる。

一般的に、企業内の情報システムを組み立てるためのWebサービスは密結合で連携し、企業間のWebサービスは疎結合で連携する。疎結合におけるトランザクションは複数個の密結合で構成されたサブトランザクションから構成される。

各々のサブトランザクションに関して疎結合環境は、さらに次の二つの場合に分けられる。

(1) 異なる企業のデータベースに共通部分が存在しない場合。例えば、旅行予約の場合、カード会社、航空会社、ホテルのような個々の共通部分が存在しないデータベースの場合である。

(2) 異なる企業のデータベースは共通するデータを扱っている可能性がある場合。例えば、銀行とローン審査会社のデータベースの場合である。

前者の場合は、異なるトランザクションの異なる企業のデータを扱うサブトランザクション間には競合が存在しない。すなわち、個々の密結合環境におけるサブトランザクションが隔離性を満たせばよい。したがって、本論文の隔離性に関する結果はそのまま疎結合環境におけるサブトランザクションの管理に適用できる。一方、後者の場合は、個々のサブトランザクションや個々のデータベースに関する詳細な情報がない限り、隔離性については従来のACID属性に留まることにならざるをえない。

4. 連携エンジンとデータベースによる実現法

従来のデータベースの並行処理制御方式は、他のトランザクションの問題となる実行を禁止する施錠による保守的な方式から時刻印方式のような楽観的な方式までである。本節では、密結合環境において、BPEL4WSエンジンのようなフローエンジンとデータベースの連携による施錠方式の実現方法について検討する。

各々のトランザクションが論理性を満たすスケジュールのみを生成させる制御方式としては、従来の2相施錠方式に対して専有施錠と共有施錠間の競合を緩和した拡張2相施錠方式が利用できる[7)]。すなわち、すでに専有施錠または共有施錠されたデータ項目の他のトランザクションによる専有施錠は禁止されるが、共有施錠は禁止されることなくいつでも施錠できる。本論文のモデルへ適用すると、次のようになる。

定義 4 基本モデルに対する施錠に基づく方式 (Locking protocol) は、次のように施錠と解錠を行う方式である。

- データベース中のデータに対して検索操作／変更操作を行う前に、それぞれ専有施錠／共有施錠を行う。ただし、他のトランザクションによってすでに専有施錠または共有施錠されているデータ項目には再び専有施錠することはできない。
- 大域変数によるデータの引き渡しによって得られるデータフロー情報を活用して、各節点による施錠は、データフロー上の子孫節点がすべて終了するまで保持する。 □

以降、施錠に基づく方式によって生成されるスケジュールをLPスケジュールという。例えば、スケジュールWH_1において、$\mathcal{A}_2^1$によって経済状況x_3に対する共有施錠は$\mathcal{A}_4^1$まで保持されることになる。同じように$\mathcal{A}_2^2$によって経済状況x_3に対する共有施錠も$\mathcal{A}_4^2$まで保持される。したがって、$\mathcal{A}_4^1$や$\mathcal{A}_4^2$によるx_3に対する専有施錠は禁止されることになり、WH_1はLPスケジュールではない。

定理 2 トランザクションWT_i $(i = 1, \cdots, n)$からなるスケジュール$WH(HN, HE)$がLPスケジュールならば、各トランザクションWT_iは論理性を満たす。

証明： スケジュール$WH(HN, HE)$に論理性を満たさないトランザクションWT_iが存在するとする。定義3よりデータベースに対する入出力データは、WT_iによって操作された後データフローの子孫節点が全て終了するまでに他のトランザクションによって変更されている。しかし、それは施錠に基づく方式の定義4と矛盾する。 □

5. おわりに

本論文では、インターネット上で公開しているWebサービスを部品として組み立てる時代の到来に備えて、Webサービスの連携によって企業内の情報システムを構築する場合に考慮すべき並行処理制御に関する問題について検討を行った。Webサービスに対しては内部ロジックが隠されているため、従来の企業内のワークフローシステムに用いられた内部ロジックによる方法は適用できなくなる。本論文では、Webサービスの連携によって企業内の情報システムを構築する場合のモデルを、疎結合と密結合の場合に分けて与えた。また、密結合のトランザクションに対して、各節点の入出力データを分類することによって、従来のACID属性よりも緩和した隔離性を保証するための条件を提案した。さらに、その実現方法は、フローエンジンとデータベースの連携によって行えることを示した。

[**謝辞**] 本研究の一部は、文部省科学研究費補助金基盤研究(C)15500079の援助を受けている。

[文献]

[1] Business Process Execution Language for Web Services (BPEL), Version 1.1, http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp

[2] P. A. Bernstein, V. Hadzilacos and N. Goodman: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).

[3] Business Transaction Protocol (BTP), http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=business-transaction

[4] R. Hull and J. Su: Tools for Design of Composite Web Services, ACM SIGMOD Inter. Conf. on Management of Data, pp. 958-961 (2004) .

[5] M. Little: Transactions and Web services, Communications of the ACM, Vol. 46, No. 10, pp. 49-54 (2003).

[6] Web Services Transaction (WS-Transaction), http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/ (August 2002).

[7] 徐海燕，古川哲也，史 一華：並行処理制御方式による独立化可能性クラスと直列可能クラスの比較、情報処理学会論文誌：Vol. 37, No. 8, pp. 1600-1609 (1996).

徐 海燕　　Haiyan XU
福岡工業大学情報工学部情報工学科教授。平成2年九州大学大学院博士後期課程修了。工学博士。並行処理制御、WEB型情報システムなどの研究に従事。ACM、IEEE、情報処理学会、電子情報通信学会、日本データベース学会等会員。

古川 哲也　　Tetsuya FURUKAWA
九州大学大学院経済学研究院教授。昭和63年九州大学大学院博士後期課程修了。工学博士。データベースの設計論・質問処理論、情報システムの研究に従事。情報処理学会、電子情報通信学会、ACM、IEEE、日本OR学会等会員。

史 一華　　Yihua SHI
西南学院大学商学部教授。平成4年年九州大学大学院博士後期課程終了。理学博士。WEB型教育支援システム、Webサービス、知識ベースシステムなどの研究に従事。電子情報通信学会、人工知能学会、日本データベース学会各会員。