サービス連携によるユビキタスネット社会の構築技術（継続）

松　本　啓之亮　　大阪府立大学大学院工学研究科教授

1 はじめに

近年，インターネット網の急速な整備により Web の利用目的が単なる情報流通基盤からサービス流通基盤へと変化しつつある．現在，インターネットを利用したオンラインショッピングや銀行・証券取引，ホテル検索･予約などのサービスが提供されている.それに伴い Web の利用者の要求もより複雑になってきており，一つのサービスだけではユーザのすべての要求に対応することができず，複数のサービスを利用する必要がある．この場合，組み合わせるサービスやサービス間の入出力の対応を考慮しなければならないが，あらかじめ利用する個々のサービスやサービス間の連携を決定すると固定されたサービスとなってしまう．また，ユーザにサービスやサービス間の連携を指定させることも考えられるが，サービス間の連携が利用するサービスのインタフェースに依存するため，同じ要求であってもサービスを変更する場合には再び設計し直さなければならないといった問題点がある.

そこで本研究では，ユーザの要求をサービス連携モデルとして実際のサービスとは切り離して保存，再利用する仕組みを提供し，そのモデルと実際のサービスを結合させる仕組みを提供することにより，ユーザの要求に柔軟に対応しつつ，ユーザの負担を軽減することを目的とする.

さらに，多くの人がサービスを検索・連携することを想定した環境下でシステムの信頼性を自律的に向上させるために，エージェント間のメッセージを監視し，エージェント相互の依存関係を反映させた相互依存グラフを提案する．この相互依存グラフを分析し，得られた大域的な情報をエージェントの複製（レプリカ）をもつレプリケーションシステムに応用することで信頼性向上を目指す.

2 サービス連携支援方式

本研究[1],[2]では，ユーザがブラウザを用いてサービスを実行し結果として HTML が返ってくるシステムを Web アプリケーション，それに対してユーザがブラウザを用いずサービスを実行し結果として XML (eXtensible Markup Language) が返ってくるシステムを Web サービスと呼び，これら二つをあわせて Web ベースシステムと呼ぶ．本研究では Web ベースシステムを連携の対象とする.

2-1 概要

ユーザの要求と実際のサービスとを切り離すために，サービスの抽象化を行う．サービスの抽象化とは，同種の内容であるが異なったインタフェースで提供されている個々のサービスを，インタフェースの違いを吸収し同じインタフェースで実行できるようにすることである．まず，同種のサービスから内容が共通の入出力項目を抽出し，サービス実行に必要な項目として定義する．そして定義された入出力項目と実際の個々のサービスの入出力項目との名前や値の変換方法を変換ルールとして保存する．サービスを実行する場合には，定義に従って入力された項目を変換ルールを用いて個々のサービスに対応する入力に変換し，サービスの実行を行い,返される出力を再び変換ルールを用いて定義された出力に変換する.これによりユーザは個々のサービスにおけるインタフェースの違いを意識することなく，サービスを実行することができる．本方式ではこの変換に XSLT (XML Stylesheet Language Transformations) を使用し，変換ルールを XSLT のスタイルシートとして保存する．また抽象化したサービスの定義は XML Schema を用いて定義する．そしてサービスの抽象化で定義されたそれぞれのサービスの入出力項目を関連付け，全体としてユーザの要求に対応したサービスの連携を行う．サービスの抽象化によって，連携の定義は実際の個々のサービスのインタフェースと切り離されるため，同じ要求で違うサービスを使用する場合にも連携の定義を変更する必要がない．使用する個々のサービスは実行時に指定する．本方式では，入出力の関連付けを含めたサービス連携の定義をユーザが定義することができ，サービス連携モデルとして保存，再利用することが可能となっている.

(1) サービスの抽象化

提案方式ではサービスを抽象化し，ユーザは抽象化されたサービス同士を連携することで要求をモデルとして記述する．これにより連携を行う時点では具体的にどのサービスを使うかを決定せず，実行時に決定す

ることができ，ユーザの要求と実際のサービスとを分離することができる．同じ種類の Web ベースシステムでも入力パラメータや出力されるデータのタグ名などが異なる．このインタフェースの違いを，XSLT を用いて同じインタフェースに変換することをサービスの抽象化と呼ぶ．各サービスごとにサービス登録者が，変換ルールを記述してサーバサイドに保存する．

(2) 抽象化されたサービス同士の連携による要求モデルの記述

ユーザは抽象化されたサービス同士の値の受け渡しや固定値などを設定することで連携を行いモデルを記述する．このとき，サービスの実行順を階層を用いて決定する．上位階層からサービスを実行していき，下位階層のサービスへと値を渡していく．

(3) サービスファイル

本システムでは，具体的なサービス一つ一つにサービスファイルを用意している．ここで，サービスファイルには具体的なサービスの抽象化に用いる入出力変換ルールや，具体的なサービスのリクエスト URL，入力データを送信する際の文字エンコーディング等サービス実行に必要な情報を記述している．

サービスファイルは XML ファイルであり，サービス名，サービス利用に必要な情報，変換ルールを保存している．サービスファイルの構成を図１に示す．サービス名は，本システムの連携実行において，具体的なサービスを選択するときに表示される名前である．サービス利用に必要な情報は，リクエスト URL（具体的なサービスにパラメータを送信するあて先)，文字エンコーディング（パラメータを送信する際に利用する文字エンコーディング)，通信方法(Web サービスサーバと通信を行う際に用いる通信方法)で構成されている．

(4) 変換ルール

変換ルールは，図１に示すような形式で入出力の変換ルールが XSL で記述されている．表１に，商品検索サービスの入力変換例を示す．

2-2 構成

提案システムは，サービス連携モデル記述・保存とサービス連携実行からなる．また提案システムは Web の利用者を対象としているので，Web アプリケーションとして実装した．機能は大きくクライアントサイドとサーバサイドに分かれる．図２に提案システムの構成を示す．サービス連携モデル記述・保存では，クライアント側でユーザがサービス連携モデルを記述し，そのサービス連携モデルをサーバ側で保存する仕組みを提供する．まず，ユーザはサーバから抽象化されたサービス一覧を取得する．次に，要求に合わせてそれらのサービスの入出力を Web ブラウザ上で関連付け，サービス連携モデルを記述する．そして保存を選択するとデータがサーバに送られてサービス連携モデルが保存される．サービス連携実行では，サービス連携モデルをもとに実際のサービスを実行させサービス連携を行う．まずユーザは実行させたいサービス連携モデルと実際に使用するサービス名を選択する．次に，必要な項目を入力し，サービス連携を実行する．サーバ側

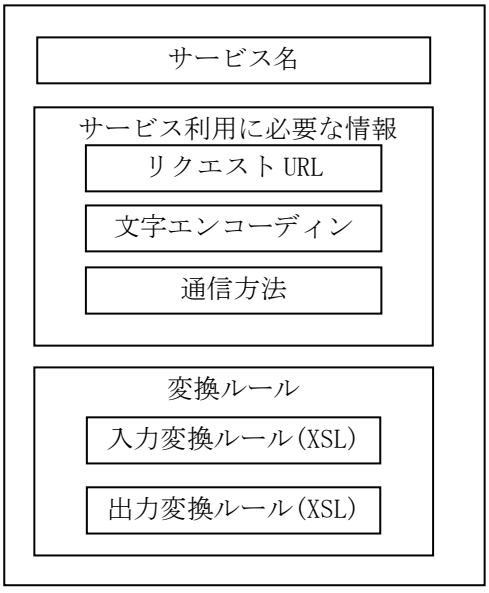

図１ サービスファイルの構成

表１： 商品検索サービスの入力変換

サービス名	変換前	変換後
価格.com	Keyword	keyword
楽天 Web サービス	keyword	
Yahoo オークション	query	

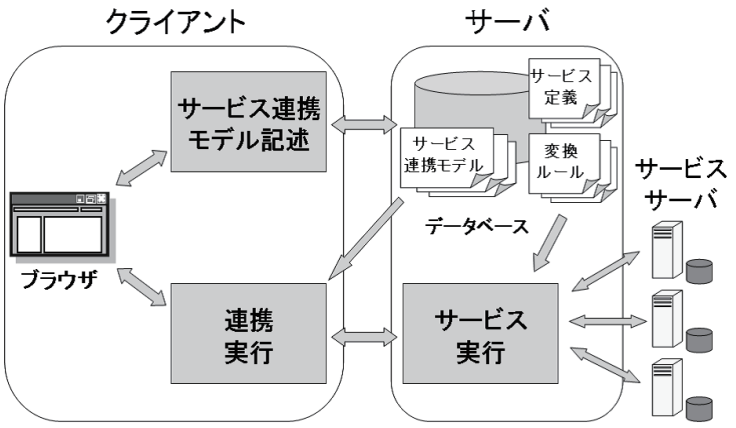

図２ 提案システムの構成

では，サービス連携モデルと指定されたサービス名をもとに変換ルールを呼び出し，入力を個々のサービスに合わせて変換しサービスを実行，そして返ってくる出力を変換し次のサービスに受け渡すことを繰り返し最後までサービス連携を行う．そして結果をクライアント側に返しユーザに表示する．

(1) **クライアントサイド**

クライアントサイドでは，ブラウザ上でのマウス操作で連携し，サーバサイドと要求モデルの送受信を行う要求モデル記述機能と，連携されたユーザの要求を実行する連携実行機能からなる．

要求モデル記述機能は，ブラウザ上でマウスのドラッグ＆ドロップなどで抽象化したサービスを連携させてユーザの要求モデルを記述する機能である．上部のメニューに階層追加のボタンがあり，これをクリックすると新しい階層のサービス画面が表示される．追加された階層には登録されている抽象化したサービス一覧とサービス追加ボタンがある．ここで階層とは，モデルの実行順をあらわすもので，左の階層から順に実行されていく．サービス一覧で抽象化したサービスを選びサービス追加ボタンをクリックすることで，階層に抽象化したサービスが追加される．抽象化したサービスには，実際のサービス一覧や実行ボタン，入力ボックス，出力ボックスがある．出力ボックスをドラッグして，その階層より右の階層の入力ボックスへドロップすることによって，抽象化したサービス同士の連携を行う．連携が完了すると，入力ボックスと出力ボックスの間に直線が描画される．また，連携後の入力ボックスや出力ボックスにマウスを重ねると，連携した直線が太く強調されどの値がどの値に連携しているかが分かりやすくなっている．サービス同士の連携を削除するときには，入力名をクリックする．また，上部のメニューに保存するファイル名を入力し保存ボタンをおすことで作成したモデルをサーバサイドに保存することができる．

連携実行機能は，抽象化したサービスを連携し実行する機能である．ユーザは必要な値を入力し，そして実際にどのサービスを実行するかを実際のサービス一覧から選択する．その後実行ボタンをクリックすることで，サーバサイドに値を送信し，結果を受信する．結果を受信したら，そのサービスと連携されているものに値を受け渡す．実行ボタンには，一つのサービスだけを実行するもの，階層すべてのサービスを上から実行するもの，そして全てのサービスを左の階層から実行するものの三種類がある．連携実行機能は要求モデル記述機能と同じインタフェースで利用する．

(2) **サーバサイド**

ユーザがブラウザ上で作成したモデルをサーバサイドに保存する．連携データはブラウザ環境があれば，読み込むことができ編集や実行，再保存などができる．

サービス実行機能は，クライアントサイドから受け取った値から実際のサービスを実行する．まず，クライアントサイドからどのサービスを実行すればよいかを読み取り，そのサービスの変換ルールファイルを用いて受け取った値を変換して，Web サービスサーバに送信し，HTML または XML ファイルを結果として受信する．同様に変換ルールを用いて結果の変換を行い，クライアントサイドに結果を返す．

3 サービス連携例

提案システムを用いた連携の例として，ホテル検索とホテルまでの路線検索及び，ホテル周辺のレストラン検索と天気予報調査を行う連携を考える．図３に提案システムのインタフェースを示す．このサービス連携モデルでは，ホテル検索サービスの出力である住所とチェックイン時刻を路線検索サービスの入力である

到着場所と到着時刻に設定する．以降同様に，路線検索サービスの到着駅をレストラン検索サービスの最寄り駅に，ホテル検索サービスの住所を天気予報サービスの場所に設定する．そして初期入力を設定し，サービス連携モデルを保存する．

サービス連携を実行する場合は，保存しておいたサービス連携モデルを呼び出し，実際に実行するサービスの指定と必要項目の入力を行う．そしてサービス連携を実行すると設定に従って値が受け渡され，ホテル検索，路線検索，レストラン検索，天気予報調査の順にサービスが実行される．

以下に”ホテル検索と選んだホテルまでの路線検索，現地の天気予報調査，そしてホテル周辺のレストラン検索”を例として提案システムを用いたサービス連携の動作の流れを述べる．

1. 階層追加ボタンをクリックして階層を３つ追加する．
2. 階層の１つ目に”ホテル検索”，２つ目に”路線検索”，３つ目に”お天気サービス”と”レストラン検索”を追加し，これらを連携させる．

ここから連携動作に入る．

3. ホテル検索が出力する”住所”をドラッグし，路線検索に入力する”到着”にドロップする．同様にお天気サービスに入力する”場所”にドロップする．
4. ホテル検索が出力する”チェックイン”を路線検索に入力する”到着時間”にドロップする．
5. 路線検索が出力する”到着駅”をレストラン検索に入力する”最寄り駅”にドロップする．
6. これらの連携を行うと，連携した出力と入力の間に線が描画される．またサービスが実行され出力が表示されると連携した入力へ値がコピーされる．

ここから連携実行に入る．

7. ホテル検索に入力を行い，実際に利用するサービスを一覧から選択し，実行ボタンをクリックする．同様に路線検索，お天気サービス，レストラン検索と順に実行する．また，連携されている入力以外の項目全てを入力し，全ての実際のサービスを選択することで，連携モデル全体を順に実行する全体実行もできる．

図３はここまでの実行例を示している．

このようにしてユーザは Web ブラウザを用いて要求モデルの記述と，要求の実行ができる．また，作成した要求モデルをサーバサイドに保存することにより，一度作成した要求モデルはブラウザで Web ページを閲覧する機能のあるコンピュータであればどこでも再利用することができる．

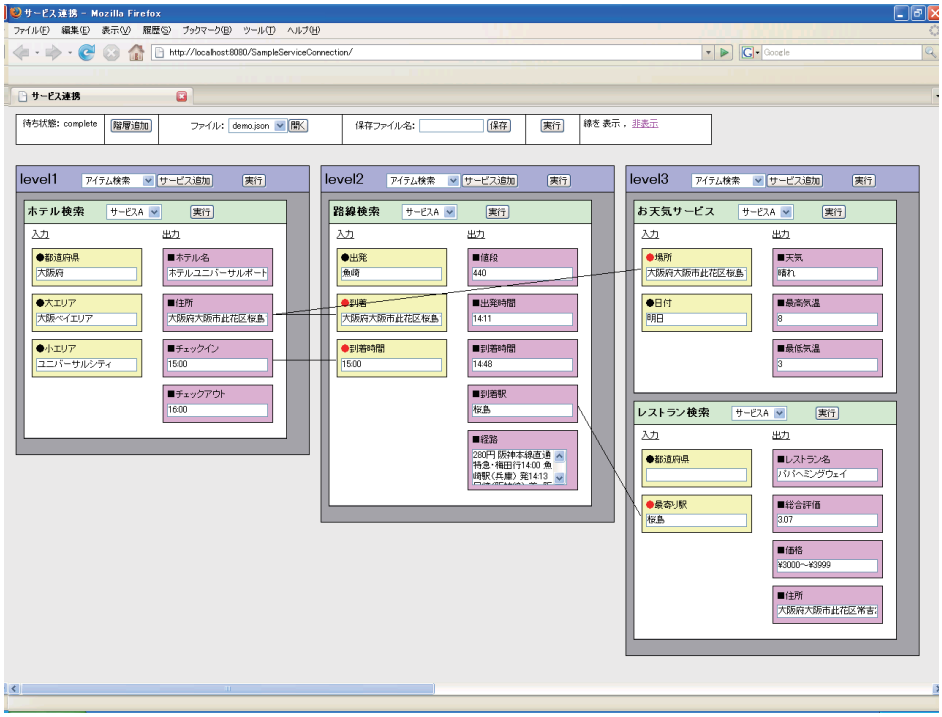

図３ 提案システムのインタフェース

4 提案システムの評価

提案システムと他のサービス連携手法の特徴を述べる．WS-BPEL[3]は，Webサービスを組み合わせたビジネスプロセスフローを記述することにより，複雑なサービス連携ができる．しかし，WSDL(Web Service Description Language) を必要とするためSOAP を用いたWeb サービス以外のサービスを連携させることができない．オントロジーを用いたサービス連携[4]は，オントロジーによるサービスの自動連携でユーザの負担を軽減する．しかし，ユーザの要求が変化したときには，オントロジーを再構築してサービスを連携させなければならない．ユーザ嗜好を考慮したサービス連携[5]は，ユーザが連携させるサービスやサービス間の値の受け渡しを行うことにより，ユーザの要求に柔軟に対応している．しかし，ユーザの要求は同じだが,同種の別サービスを利用したいといった場合,再連携を行わなければならずユーザにとって負担となる．PatchService[6]は，ユーザがブラウザを用いてサービスの登録やサービス同士の連携を行い，それらをサーバに保存する．これにより，同じ要求がすでに作成されていたら検索して利用することができ，ユーザの負担が軽減される．しかし，実際の具体的なサービスを連携しているので，同種の別サービスを利用したいといった場合や同じ要求でも利用するサービスの内一つだけが異なるようなよく似た連携が行われる場合には，再連携や再検索をしなくてはならない．

提案システムでは，従来システムは行っていなかったサービスの抽象化を行うことで，ユーザの要求と実際の具体的なサービスとを切り離した．これにより，ユーザの要求は変化しないがサービスの故障などで同種の別サービスへの切り替えが必要なときに，サービスの再連携をすることなく元の要求モデルを利用することができ，ユーザの負担を軽減することができた．

5 信頼性向上手法

近年，分散システムにおける競合や協調を解決する手法としてマルチエージェントシステム(Multiagent System: MAS) が注目されている．しかし，MASは故障の伝搬による脆弱性のために，実際に稼働しているシステムは小規模にとどまっているのが現状である．そこで既存の MAS に対し，容易にその信頼性を向上させるための一般的な構成手法[7]を提案する．

5-1 相互依存グラフ

図４のようにドメインのエージェントをそれぞれノードに関連付け，そのノード集合を相互依存グラフと呼び，ラベル付けされた(N, L, W) で表現する．N はグラフのノード集合，L はリンク集合，W はラベル集合である．n をノード数とすると，

$$N = \{Ni\}\ i=1,\text{---},\ n \qquad (1)$$

$$L = \{Li,j\}\ i=1,\ \text{---},\ n,\ j=1,\ \text{---},\ n \qquad (2)$$

$$W = \{Wi,j\}\ i=1,\ \text{---},\ n,\ j=1,\ \text{---},\ n \qquad (3)$$

Li,j は，ノード Ni から Nj へのリンクであり，Wi,j は Li,j にラベル付けされた実数値である．Wi,j は関連するエージェント間の相互依存の重要度を反映したものである．また，ドメインのエージェントが追加，もしくは削除された場合，動的に更新される．

5-2 モニタリングアーキテクチャ

効率性と信頼性を向上させるために観測メカニズムを分散させ，ドメインエージェントと監視エージェントを１対１で関連付ける．また，提案するモニタリングシステムは以下の２つの役割をもつ．

・ドメインエージェントの観測と制御
・大域的情報の生成

これらの役割は監視エージェントとホストモニタエージェントの２種類のエージェントに割り当てられ，監視エージェントは各ドメインエージェントに，ホストモニタエージェントは各監視エージェントにそれぞれ関連付けられる．

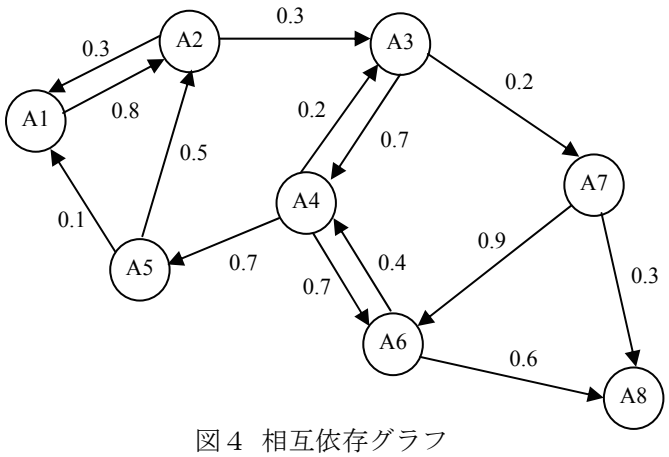

図４ 相互依存グラフ

5-3 重み更新アルゴリズム

監視エージェントはドメインエージェントの

変化を相互依存グラフに反映させる．重み更新アルゴリズムに用いる指標を以下に示す．

・時間インターバルΔt

・通信負荷 $Q(\Delta t)$

$$Q(\Delta t) = operator1(Q1,1(\Delta t), \ldots, Qn,n(\Delta t)) \quad (4)$$

・送信メッセージ数 $NM(\Delta t)$

$$NM(\Delta t) = operator2(NM1,1(\Delta t), \ldots, NMn,n(\Delta t)) \quad (5)$$

$Qi,j(\Delta t)$ および $NMi,j(\Delta t)$は，時間インターバルΔtにおけるエージェント i, j 間の通信量および送信メッセージ数，$operator$ は集合演算子である．一般的なものには平均演算がある．相互依存グラフに用いる重み Wi,j を更新する適応アルゴリズムの概要を以下に示す．

1) エージェント j ($j \neq i$) について以下を繰り返す

2) 式(6), (7) の計算

$$Qtemp = (Qi,j(\Delta t) - Q(\Delta t))/Q(\Delta t) \quad (6)$$

$$NMtemp = (NMi,j(\Delta t) - NM(\Delta t))/NM(\Delta t) \quad (7)$$

3) 重み Wi,j の更新

$$Wi,j(t+\Delta t)=Wi,j(t) + \alpha \times operator3(Qtemp,NMtemp) \quad (8)$$

4) 繰り返しの終端

このアルゴリズムは関連ノードを管理するために各監視エージェントにより実行される．

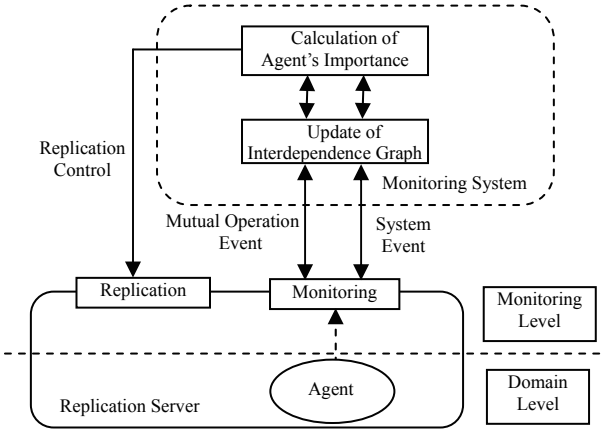

図５　適応型マルチエージェントアーキテクチャ

5-4 適応型マルチエージェントアーキテクチャ

提案するエージェントアーキテクチャは，図５に示すようなモニタリングシステムと，適応型レプリケーションシステムを実装したレプリケーションサーバで構成される．

(1) 適応型レプリケーションシステム

レプリケーションは，分散システムの耐故障性を向上させる効果的な手法である．提案システムでは相互依存グラフから得られる各エージェントの重要度に応じてレプリカの配置数を動的に調整する適応型レプリケーションシステムを提案する．

(2) 相互依存度を用いたレプリカ数決定アルゴリズム

相互依存グラフの分析は，エージェントの重要度と故障の影響，MAS の耐故障性を把握するのに役立つ．提案システムではエージェントの重要度を評価するために，各エージェントの入出力リンクにラベル付けされている重み Wi,j を $operator4$ により集合演算することでエージェントの重要度を算出する．その式を，m を次数として式(9) に示す．

$$wi = operator4(Wi,j\ j=1, ---, m) \quad (9)$$

また，エージェント i の重要度 wi は，適応型レプリケーションシステムにおいてレプリカの数 $repi$ を算出するために用いられる．その算出式を式(10) に示す．

$$repi = round\,[r_0 + r_{max} \times wi / W\,] \quad (10)$$

ここで用いられるパラメータ W は，全エージェントの wi の合計値，r_0 はレプリカ数の初期値，そして，r_{max} は設計者が設定するレプリカの総数の上限値である．

5-5 シミュレーション実験

(1) 対象システム: eMarket MAS

eMarket は，製品工場の部品購入エージェントが部品工場に属する部品販売エージェントから部品を購入し，それを加工した製品を製品販売エージェントが小売業者エージェントに売却するというシナリオをモデル化したものである．市場は A と B の２種類存在し，エージェントはそれぞれの市場に参加し，異なる役割を果たすことができる．また取引はオークション形式で行われ，市場管理エージェントがその管理を行

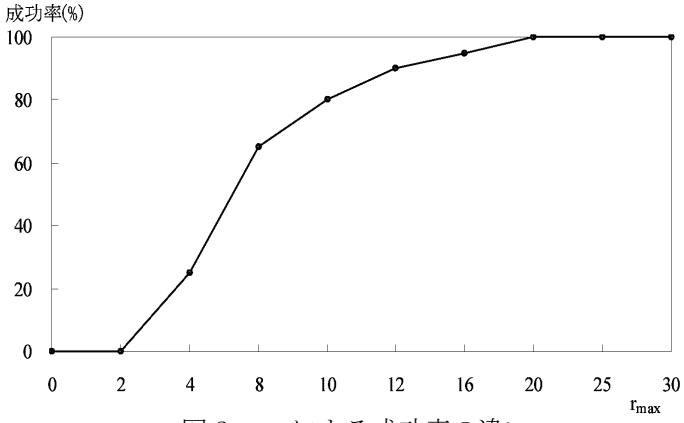

図６　r_{max} による成功率の違い

う．この eMarket MAS に対して提案システムを実装し，検証を行った．

(2) 実験結果と考察

耐故障性を検証するために，全 100 エージェントで構成されるシステムで，10 分間に故障を計 100 個発生させる実験を r_{max} を変化させながらそれぞれ 40 回行った．図 6 はそのシミュレーションの成功率を表したものである．この実験環境の場合は r_{max} を 20 以上に設定することで，システムの信頼性を維持できることを示している．

6 あとがき

本研究では，以下のような成果が得られた．

1) サービスの抽象化と抽象化されたサービスの連携をサービス連携モデルとして記述することにより，ユーザの要求と実際のサービスを切り離すことができた．
2) サービスの抽象化と抽象化されたサービス同士の連携によりサービスを連携し実行する仕組みを提供することによって，サービスの故障への対応や別のサービスへの切り替えといったユーザの要求に柔軟に対応しつつ，ユーザの負担の軽減が可能となった．
3) 提案の連携方式を Web アプリケーションとして作成し，要求モデルをサーバ側に保存することによって，複数のユーザと要求モデルを共有したり，場所を選ばず要求モデルの使用が可能となった．
4) 多人数がサービスを検索・連携することを想定した複雑かつ動的な環境下で，システムをモニタリングすることにより，大域的情報を表現できる相互依存グラフを生成し，またこの大域的情報を耐故障性の向上に利用した適応型レプリケーションシステムを提案した．シミュレーション結果より，自律的に信頼性を向上させることが可能であることが検証された．

【参考文献】

[1] 市川 正志:ユーザ要求モデルを用いたサービス連携支援システムの提案, 大阪府立大学工学部情報工学科卒業研究報告書, 2008.

[2] 有江 弘朗, 市川 正志, 松本 啓之亮:モデルを用いたサービス連携支援システムの提案, 第 52 回システム制御情報学会研究発表講演会論文集, pp. 601-602, 2008.

[3] URL: http://www.oasis-open.org/home/index.php

[4] 福田 直樹, 肥塚 八尋, 和泉 憲明, 山口 高平:オントロジーに基づく Web サービスの自動連携, 電子情報通信学会技術研究報告. KBSE, 知能ソフトウェア工学, Vol.104, No.49(20040510) pp. 13-18, 2004.

[5] T. Suda, T. Itao, T. Nakamura and M. Matsuo, “Adaptive Networking Architecture for Service Emergence,” Trans. Inst. Electronics and Communication Engineers of Japan (IECEJ), Vol.J84-B, No.3, p.310-320, 2001.

[6] URL: http://dev.patchservice.net/trac/

[7] A. Tanimoto, K. Matsumoto and N. Mori, “An Adaptive and Reliable System Based on Interdependence between Agents,” Electrical Engineering in Japan, Vol. 164, No. 1, pp. 62-68, 2008.

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
エージェントの相互依存関係を用いた適応型高信頼性システム	電気学会論文誌 C, Vol. 126, No.4, pp. 451-456	2006.4
A Dependable System Considering Interdependence between Agents	Proc. of the 2007 IEEE Systems Conference, pp. 349-354	2007.4
エージェント情報を基にしたディペンダビリティ向上手法の提案	平成１９年電気学会電子・情報・システム部門大会講演論文集，pp. 892-896	2007.9
An Adaptive and Reliable System Based on Interdependence between Agents	Electrical Engineering in Japan, Vol. 164, No. 1, pp. 62-68	2008.4
モデルを用いたサービス連携支援システムの提案	第 52 回システム制御情報学会研究発表講演会論文集，pp. 601-602	2008.5