2J-05 シナリオを用いたサービス連携システムの実現と評価

吉田 英嗣† 横山 和俊† 元田 敏浩‡ 畑島 隆‡ 箱守 聰†
†NTT データ 情報科学研究所 ‡NTT 情報流通プラットフォーム研究所
e-mail: {eiji, yokoyama, hako}@rd.nttdata.co.jp, {motoda.toshihiro, hatashima.takashi}@lab.ntt.co.jp

1. はじめに

インターネットの進展により、航空機予約サービスやオンラインバンキングサービスなどネット上に様々なサービスが提供されるようになっている。更に、このようなネット上に散在するサービス群を相互の連携させ1つの新しいサービスとして提供するというニーズも高まりつつある。そこで我々はネット上に提供されている既存のサービスを容易にかつ信頼性を保ちつつ連携させるプラットフォーム「DCAIS(Distributed and Cooperative Architecture for Information Sharing)」の開発を行っている。本報告ではDCAISによるサービス連携システムの実現について述べる。

2. サービス連携の問題点

連携させるサービス群はもともと別システムとして構築されたものである。例えば、個別に「航空機予約サービス」や「ホテル予約サービス」がネット上で提供されており、これらを連携させ新たな連携サービス（例えば「旅行予約サービス」）を実現することを考える。このとき、既存サービスを変更せずに連携させることに加えて、以下のような要求条件がある。

[要求条件 1] 連携サービスが、1つのサービスとして整合性を保たなければならない。例えば、「旅行予約サービス」では連携させる予約サービスのどちらか一方でも失敗すれば、もう一方のキャンセル処理が必要になる。個別のサービスは、各サービスの提供元で運用されており、連携サービス全体で従来のようなトランザクション処理を行うことは困難であるため、連携サービスの整合性を保証する機能が必要である。

[要求条件 2] 連携サービスに参加する個別のサービスは、連携サービスの発展に伴って徐々に追加される。例えば、上記例に加え、「鉄道予約サービス」や「レンタカー予約サービス」が後に追加され、サービス範囲が徐々に拡大することが考えられる。このとき、既に実現している連携サービスのロジック（サービスの組み合わせや実行順序）からの変更が容易でなくてはならない。

3. サービス連携プラットフォーム

現在、我々は上述の要件を満足するサービス連携プラットフォーム「DCAIS」を開発中である。DCAISの構成を図1に示し、以下に説明する。

[シナリオ] サービス連携のロジックは、XML(Extensible Markup Language)ベースの「シナリオ記述言語」により記述され、シナリオリポジトリに登録される。

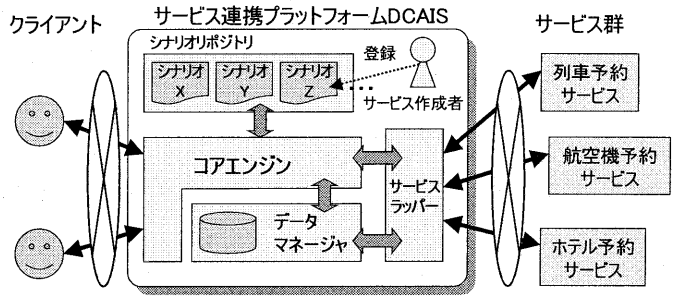

図1. サービス連携プラットフォーム"DCAIS"構成

[コアエンジン] シナリオで定義されたサービス連携ロジックをクライアントからの要求に従い実行する。サービス連携の実行状態を管理し、連携サービスの整合性を保証する。また、データマネージャを通して、データの永続化を行い、高信頼化を実現している。

[サービスラッパー] 個別サービスは、サービスラッパー経由で実行される。連携したい各サービス固有のインターフェースをDCAISサービスラッパーに登録することでDCAISとの接続を可能にしている。

さらにDCAISは、サービス連携の要求条件に対し、以下の方法で対処している。

[対処1] 連携サービスの整合性保証

DCAISは、連携サービストランザクション方式[1]により、連携サービスの整合性を保証する。これは、正常系の動作を打ち消す逆操作を用いて擬似的にトランザクションを構成し、整合性を保証する。

[対処2] 連携サービス変更の容易性

DCAISシナリオは、個別サービスに関する記述をコンポーネント化することにより、個別サービスの追加／変更をシナリオの局所的な変更で可能にしている。

4. DCAISによるサービス連携の実現

4.1. アプリケーション例

DCAISの有効性を検証するため、DCAIS上に連携サービスを実現した。実現したアプリケーションは、「旅行予約サービス」である。旅行予約サービスで連携させる個別サービスを表1に示す。また、ホテル予約と航空機予約を行うシナリオ例を図2に示す。シナリオは、個別サービスを定義する部分（<Flow>タグ部）と連携サービスの整合性を

"Realization of Scenario-based Service Collaboration System"
Eiji Yoshida†, Kazutoshi Yokoyama†, Toshihiro Motoda‡, Takashi Hatashima‡, Satoshi Hakomori†
† Laboratory for Information Technology, NTT DATA ‡ Information Sharing Laboratories, NTT

記述する部分（<Constraint>タグ）から構成されている。

表1 個別サービス

項番	個別ｻｰﾋﾞｽ名	項番	個別ｻｰﾋﾞｽ名
①	ホテル予約	③	鉄道予約
②	航空機予約	④	ﾚﾝﾀｶｰ予約

```
<Scenario Name="TravelAgency">
 <Flow>
  <Service Name="ホテル予約">
   <AdaptorID>Adaptor2</AdaptorID>
   <Input>ユーザリクエスト情報</Input>・・・
   <Output>ホテル予約情報</Output>・・・
   <Reverse>ホテル予約キャンセル・・・</Reverse>
  </Service>
  <Service Name="航空機予約">                    (α)
   <AdaptorID>Adaptor1</AdaptorID>
   <Input>ユーザリクエスト情報</Input>・・・
   <Output>航空機予約情報</Output>・・・
   <Reverse>航空機予約キャンセル・・・</Reverse>
  </Service>
 </Flow>
 <Constraint>ホテル予約→航空機予約</Constrain>
</Scenario>                                     (β)
```

図2. 旅行予約サービスシナリオ例

4.2. 連携サービスの整合性保証

DCAIS の連携トランザクションサービスを検証するため、図３に示す２つの制約について検討する。制約１は、①～④の個別サービスがすべて成功しなければ、既に行った予約処理をすべて打ち消す制約である。制約２は、（①②④）か（①③）のいずれかが成功すれば良いという制約である。それぞれの場合の整合性記述を図４に示す。

制約１の場合、最後のレンタカーサービスが不成功であれば、制約条件の左辺である鉄道予約が打ち消され、キャンセル処理が実行される。以降、順次制約条件に従い、「航空機予約」と「ホテル予約」がキャンセルされる。一方、制約２の場合、レンタカー予約に失敗した場合、航空機予約がキャンセルされるが、もう一方の候補である鉄道予約が実行される。鉄道予約が成功すると整合性が保証されるため、ホテル予約のキャンセル処理は実行されない。

これにより、DCAIS の連携サービストランザクション方式は、連携サービス全体を実行前の状態に戻す制約に加え、

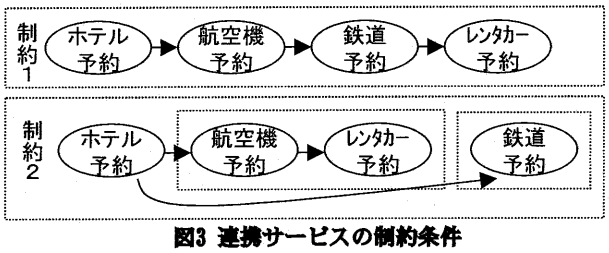

図3 連携サービスの制約条件

```
<Constraint> ホテル予約 → 航空機予約 </Constraint>
<Constraint> 航空機予約 → 鉄道予約 </Constraint>
<Constraint> 鉄道予約 → レンタカー予約 </Constraint>
```

(a) 制約1の場合の記述

```
<Constraint> ホテル予約 → 航空機予約 | 鉄道予約 </Constraint>
<Constraint> 航空機予約 →レンタカー予約 </Constraint>
```

(b) 制約2の場合の記述

図4 制約記述例

図2中の破線部（α）に挿入

```
<Service Name="列車予約">
 <AdaptorID>Adaptor3</AdaptorID>
 <Input>ユーザリクエスト情報</Input>・・・
 <Output>列車予約情報</Output>・・・
 <Reverse>列車予約キャンセル・・・</Reverse>
</Service>
```

図2中のアンダーライン部分（β）を変更

```
<Constraint> 航空機予約→列車予約 </Constraint>
<Constraint> 列車予約→ホテル予約 </Constraint>
```

図5. シナリオ変更例

一部のみを元に戻す制約が記述でき、柔軟な制約を記述できることが分かる。

4.3. サービスの拡張性

DCAIS のシナリオにおける個別サービスの記述は、<service>タグで囲まれた範囲のみで記述され、個別サービス毎にコンポーネント化されている。そのため、図 2 に示すシナリオに「列車予約サービス」を追加する場合、シナリオ中の(α)、(β)に図５に示すシナリオを追加・変更するのみでよい。これにより、連携サービスの拡大に伴い、サービス拡張を容易に行うことができることが分かる。

4.4. 性能オーバヘッド

連携サービスの実現を容易にし、高い拡張性を持つ一方で、DCAIS シナリオを解釈し実行するためには、その制御のためのオーバヘッドが大きいと考えられる。DCAIS 上で 8 個のサービスを連携させ、さらにそれらサービス間に 3 組の一貫性保証を行う仮想シナリオを記述し実行を行い、サービス実行時のフロー制御にかかる時間を測定した。個別サービスの実行時間が 0.5 秒と 5 秒の時、DCAIS オーバヘッドが全体処理時間に閉める割合を表２に示す。

表2 DCAIS のオーバヘッド

個別サービス実行時間	0.5 秒	5 秒
実サービス実行	43.2%	86.6%
フロー制御	40.1%	9.5%
データ永続化	16.7%	3.9%

実際、個別サービスの実行時間が短いイントラネットなどにおけるサービス統合を考えると、必ずしもオーバヘッドが小さくはなく、更なる高速化が必要である。しかし、インターネット上のサービスの多くは現状で実行に数秒かかるのが普通であり、DCAIS を利用することで連携サービスの市場投入が素早くでき、提供後のもービス拡張が容易であることを考えると、その代償としてのこのオーバヘッドはあまり大きなものではない。また現在、更なる高速化の目処もたっており、フロー制御時間を 1/10 以下に短縮できる見込みである。

5. おわりに

サービス連携プラットフォーム「DCAIS」を用いたサービス連携システムの実現について述べた。今後はさらに DCAIS の機能拡張やシナリオ記述性の定量的評価を行う。

【参考文献】

[1] 元田, 畑島他, 連携サービストランザクション方式の提案, pp.65-70, 信学技報 KBSE2000-29, 2000