LL-001

携帯電話実装型サービス合成エンジンの研究開発と評価

Study of Service Composition Engine Implemented on Cellular Phone

山登庸次† 田中洋平† 徳元誠一† 大石哲矢† 武本充治†
Yoji Yamato, Yohei Tanaka, Seiichi Tokumoto, Tetsuya Oh-ishi, and Michiharu Takemoto

1. はじめに

IT技術の進歩に伴い、PCだけでなく、携帯電話、家電、センサなど様々な機器がネットワークにつながりユーザ生活を支援する、ユビキタス環境が現実になりつつある。ユビキタス環境では、従来は実現が困難だった、ユーザ状況・嗜好に適したコンテキストアウェアサービスの提供が期待されている。

ユーザニーズは、状況に応じて変化するため、その場の状況に応じてサービスコンポーネントを発見・組み合わせてサービスを実現する、サービス合成技術（[1][2]等）が注目を集めてきた。Web Services（WS）をはじめ、サービス合成関連技術は、基本的に、サービス設計図（本稿ではサービステンプレート：STと呼ぶ）をもとに、コンポーネント（本稿ではサービス要素：SEと呼ぶ）を発見・バインドすることでサービスを実現する。

本研究はサービス合成技術に関する研究の一つで、特にエンドユーザが自由にサービス合成することを目標にしており、そのために意味的情報を用いて動的にインタフェースを解決するサービス合成エンジンの研究開発をしてきた[3][4]。本稿では、特に、サービス合成機能を軽量化して、携帯電話でサービス合成するための、携帯電話実装型サービス合成エンジンについて説明する。さらに、エンジンの性能測定を行い、評価を行う。

2. サービス合成概要と携帯電話実装メリット

2.1 サービス合成技術の目的とアプローチ

本研究の目的は、エンドユーザが、ユーザ端末上のサービス合成エンジンを用いて、ネットワーク上のSEを自由に合成して、合成サービスを実現できる基礎技術の確立である。

本稿では前提として、対象とするSEはWSとUPnPと携帯電話機能に限定、SEはサービス記述をOWL-S（Web Ontology Language for Services）[5]にてネットワークに公開、を想定する。WSとUPnP以外のコンポーネント、例えばセンサやコンテンツを利用する際は、全てWSに変換し、WSとして呼び出すこととする。これらの、公開され利用可能なSEを、STに基づいて動的に発見、バインドすることで合成サービスが実現される。

既存関連技術として、個々のWSを連携させる手法である、BPEL（Business Process Execution Language for Web Services）[6]がある。しかし、BPELは元来B2Bが対象で、ユーザが未知のWSをその場で発見してバインドするには向かない。具体的には、設計図であるBPEL文書には、個々のWSのポートタイプ名及びオペレーション名が記述され、それらが完全一致のWS、すなわちほぼ一つのWSにしか適用できず、状況適応性がない。

そこで、著者らはサービス設計図であるSTにネイティブなインタフェース名を記述するのでなく、インタフェースの意味的なメタデータをSemantic Web技術を用いて抽象的に記述し、（OWL-Sのプロセス名で記述）、インタフェースは異なるが機能は同等なSEを選択・切換可能にすることで、状況に応じたサービスの実現を目指している。サービスを利用する際は、欲する機能のメタデータを用いてSEのOWL-S記述を検索し、その際に取得したGrounding記述を用いてSEのネイティブなインタフェース（WSDLやUPnP等）に、グラウンディングして呼び出す形となっている。これらの検討は[3]を参照されたい。

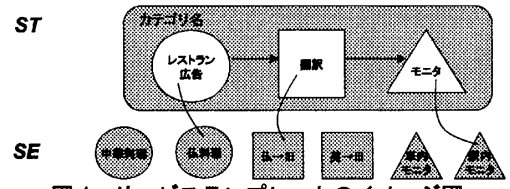

図1：サービステンプレートのイメージ図

2.2 サービス合成エンジンの携帯電話実装メリット

携帯電話は、多くのユーザが持ち歩くデバイスであり、また端末技術の進歩により赤外線やBluetooth、QRコードなど多機能を搭載しており、いつでもどこでもサービス利用するためのコントローラとして、最も適したデバイスといえる。そこで、私たちはサービス合成の入り口として、携帯電話をコントローラとしてサービス合成することを目標に検討を行っている。

携帯電話をコントローラとする形態に以下の2つの案がある。

案1：ネットワーク上のサーバでサービス合成処理を行い、携帯電話はそこにアクセスするGUIのみ提供する。

案2：携帯電話にサービス合成処理機能を実装し、不足機能はネットワーク上のサーバでカバーする。

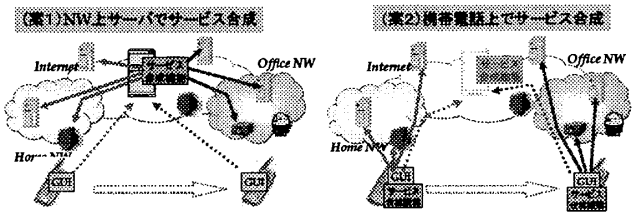

図2：サービス合成機能の実装形態案

案1は、サーバ側ですべての処理を行ってしまうため、携帯電話に対する負荷は少ないという利点がある。しかし、

・目前のデバイスもわざわざインターネット経由で制御

・ホームNWやオフィスNW内のサービスやデバイスの利用を許可する場合、その起動インタフェースをインターネットに公開する必要があり、SE提供者のセキュリティ上の不安

・ユーザ情報をインターネット上サーバで管理する必要

などの課題が存在する。

案2は、携帯電話端末に合成機能を載せるため、携帯電話端末に負荷がかかるという、短所があるが、

・端末の無線技術や非接触IC等により、身近なデバイスを直接に制御可能で、実行速度・レスポンスに優れる

・インターネットに非接続のSEのアドホックな合成が可能

・サービスの適切な選択に必要なユーザ情報を携帯電話で管理できるため、プライバシ観点でユーザ抵抗が少ない

・処理が携帯端末に閉じるため集中サーバへの負荷が少ない

などの利点がある。

すなわち、案2で行うことで、様々なメリットがあるが、携帯電話性能に依存するという点が懸念される。そこで、私たちはまず、携帯電話でサービス合成できるかどうかの評価を行うことを目的に、携帯電話に一部合成機能を実装し、測定をした。

† 日本電信電話株式会社 NTTネットワークサービスシステム研究所
NTT Network Service Systems Laboratories, NTT Corporation

3. 携帯電話実装サービス合成エンジン

現状の携帯電話では、PCに実装したサービス合成エンジン[3]の全機能（ST検索、SE検索、SE自動選択、サービス実行）を搭載するのは無理と想定される。そのため、ネットワーク上のサービス合成プロキシと機能分担することで、サービス合成することをファーストステップに検討を行った。機能配備にはいくつかの考え方があるが、本研究では、サービス合成に最も重要な実行制御機能を携帯電話に配備した際に、どの程度のユーザレスポンスを達成できるか検証することを重視して、実装を行う。実行制御機能を携帯電話にのせることで、インターネットに繋がっていない赤外線デバイスやBluetoothデバイス利用が可能、携帯電話上の様々な機能（QRコード、指紋認証機能など）を利用可能などのメリットが生まれる。

上記方針をもとにした、処理フロー概要を以下に示す。ST検索と選択、SE検索と選択をブラウザ経由でプロキシ側で行い、起動対象のSEとその順番を決定した後、それを記述したサービスシナリオを携帯電話側にダウンロードして、携帯電話側でサービスシナリオを実行する。これらの処理を実行する、携帯電話型サービス合成システムの内部処理概要を図3に示す。

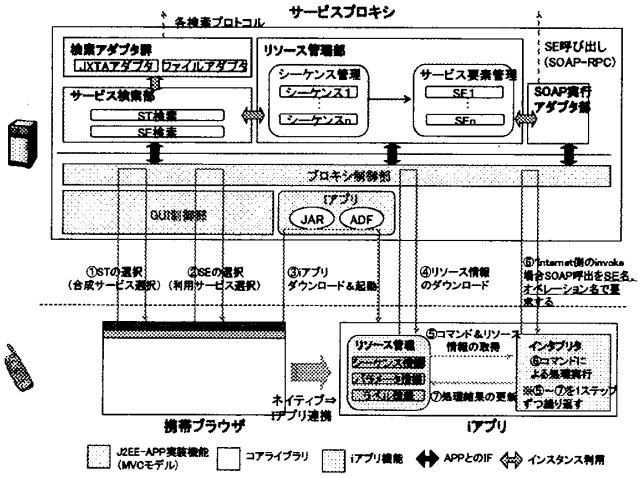

図3：携帯電話実装型サービス合成システム内部処理概要

図3の①②にあるように、ST選択、SE選択、サービスシナリオの生成までは、携帯電話端末のブラウザによる操作を行い、サービスシナリオが生成された時点で、iアプリを起動し（③）サービスシナリオをダウンロードして、それ以降はiアプリにより携帯電話端末主導で、サービス実行を行う。

合成サービスを実行するには各SEの動的なバインドが必要であり、そのためにはSEのネイティブインタフェース情報など多くのリソース情報を保持する必要がある。しかし、現状の携帯電話端末ではこのようなデータ量が多くかつ可変長の情報を保持することは適さないため、必要最低限のリソース情報のみを携帯側に構築するようにする（④）。もともとインターネット側のSEを実行する場合は現状のiアプリ仕様ではダウンロード元（本システムではプロキシ）を経由して実行をする必要があるため、SEのインタフェース情報や型情報などの詳細情報についてはプロキシ側で保持し、携帯端末側からの実行要求に応じて実際のSE呼び出しを行う（⑥）ゲートウエイ機能を提供する。一方、携帯電話側にあるSEについては必要な詳細情報をはじめから携帯電話側で保持することでiモード接続が切断されても携帯電話側SEのみの合成サービスを実行可能であるが、これらのリソース配置についてはサービス入換え等と関連させた検討も必要であるため、本設計においては携帯電話内の組み込みデバイス（赤外線等）を用いたSE実行制御を除いて対象外とする。

4. アプリケーション例

携帯電話実装合成エンジンを用いたアプリケーション例として、「どこでもカラオケサービス」を実現した（図4参照）。本アプリケーションは、好きな音楽コンテンツのURLが入ったQRコードを携帯電話で読み取ると、携帯電話実装エンジンが、該音楽コンテンツをインターネットから取得し、音楽コンテンツ内のキーワードを基にその音楽にあった画像や歌詞カードを取得し、取得画像をプリンタやモニタに出力し、音楽コンテンツを赤外線通信でスピーカに出力することを一括で行うサービスである。ここで、取得コンテンツは音楽や画像に限らないし、出力デバイスもプリンタやスピーカに限らない。

このアプリによる確認ポイントは、以下の2つである。

・携帯電話をコントローラにしてサービス合成できる

・携帯電話実装エンジンにより、QRコードや赤外線デバイス等、インターネットに繋がってないSEも利用できる。

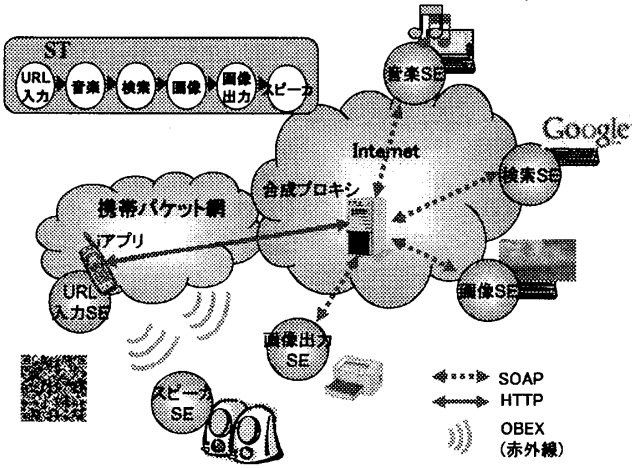

図4：どこでもカラオケサービス構成図

5. 携帯電話実装合成エンジンの性能測定

5.1 性能測定概要

前章のように実装した携帯電話実装合成エンジンに関して、処理時間、使用メモリ量、ファイルサイズの測定を行い、評価を行った。測定概要は以下のとおりである。

●測定項目

合成プロキシ、合成iアプリを対象に、以下の項目の測定を行った。測定シナリオは「どこでもカラオケサービス」である。

・処理時間：処理区間ごとの経過時間

・使用メモリ利用： プログラム全体の所用メモリ量

●測定方法

上記項目の測定データを、実行ログに埋め込み、記録する。

●処理区間

(1)合成プロキシ

・ST要求～ST候補表示

・ST決定～（1つ目の）SEリスト表示（全SE候補の収集）

・SE決定～次のSEリスト表示

・SE決定～サービス実行確認表示

(2)合成iアプリ

・iアプリ起動～リソースダウンロード開始

・リソース情報ダウンロード開始～終了

・ST実行開始～ST実行完了（SE呼び出し～リターンの区間も測定するが、機種依存やユーザ操作が入るため参考データ程度）

●測定環境（図5参照）

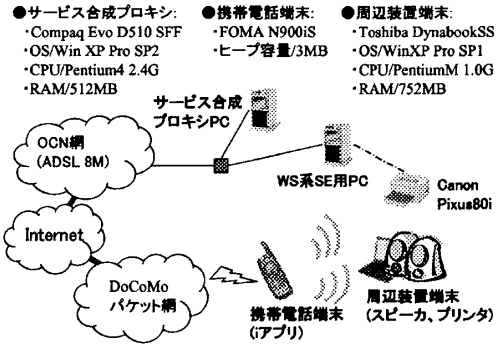

図5：性能測定環境

5.2 性能測定結果

評価の中心となる測定結果を図6、図7に示す。

5.2.1 処理時間

【全体】

どこでもカラオケサービス一連処理にて、ユーザのGUI操作（候補SEからの選択など）が1分程かかるが、全体として2分以内に実行できるなど、顕著な問題は見受けられない。ただし、これはSEの候補が10個に限定されているためで、このSE個数が増加するにつれて処理時間に影響がでると思われる。ST、SEの検索機能の改善や、ST、SEを検索した段階で候補を10個程度まで絞り込めるようにする方法が今後の課題と考える。また、ユーザが候補SEを選択しやすいGUIの工夫も必要である。

【プロキシ】

処理の中でSE実行(Invoke)が比較的時間がかかっているが、その多くはSE固有の機能（Google画像検索、MIDIファイルダウンロード等）によるもので、プロキシの実行性能は問題ない。

【iアプリ】

iアプリに対して、オブファスケータを利用したクラスファイルサイズ削減を施すor施さない2種類のJARファイルを作成し、測定したが変化は無かった。ファイルサイズ項で言及するが、オブファスケータの利用により、より機能搭載が可能である。尚、利用したツールは、RetroGuard Java Obfuscator v2.0.1である。

5.2.2 使用メモリ量

【全体】

処理時間と同じく、特に問題は見受けられなかった。

【プロキシ】

1. 初期サイズ7MBから、全SE候補の収集時に13MBまで増加（ガーベージコレクションのため、その後微減）。
2. その後もリソース情報（シーケンス情報、オペレーション情報等）処理やInvokeコマンド実行時に多少の増減はあるものの、ほぼ横ばいで推移。

1の状態からほぼ変動がないことからもわかる通り、プロキシは全SE候補の情報（SEのOWL-Sファイル、WSDL等）を保持し続けている。多ユーザ対象のプロキシを想定するため、不要SE情報の破棄など、管理方式の検討が今後の課題と考える。

【iアプリ】

1. 初期サイズ700KBから、リソース情報ダウンロード時に2.2MBまで増加。
2. 音楽配信サービスからMIDIファイルを取得時にピークの2.6MBまで増加。
3. 2の直後、1.2MB前後まで減少（N900iSのヒープサイズ3MBに近づいたため、ガーベージコレクションが働いた）。
4. 画像配信サービスからJPEGファイルを取得時に第二のピーク2MB前後まで増加。
5. 4の直後、再び減少し、その後は横ばいで推移。

以上より、iアプリについてはInvokeなどの各機能より、音声ファイルや画像ファイルなどのメディアデータを扱う場合にメモリへの影響が大きいと考えられる。ST実行時のパラメータ管理方法や、メディアデータの扱い（携帯ネイティブ機能の利用等）がiアプリでの課題になると考える。

参考：iアプリが保持したメディアデータのサイズ

MIDI => 23KB、 JPEG（Google検索） => 20KB前後

5.2.3 ファイルサイズ

iアプリのJARファイルサイズは以下の通りである。

・iアプリA => 64.8KB ：オブファスケータ処理なし

・iアプリB => 46.3KB ：iアプリ全体をサイズ削減

DoJa3.5のJARファイル上限は100KBなので、今後iアプリへ、kSOAPなどの機能を追加する余地があると言える。

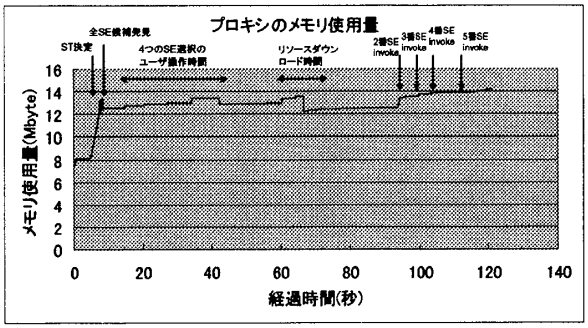

図6：プロキシメモリ使用量（携帯に閉じたinvokeは含まない）

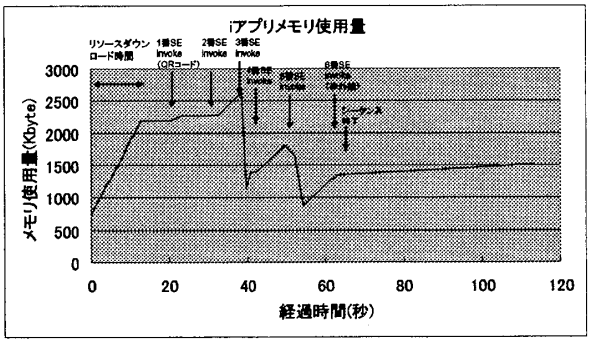

図7：iアプリメモリ使用量（携帯に閉じたinvokeを含む）

6. まとめ

本稿では、合成エンジンの実行機能を携帯電話に搭載した、携帯電話実装合成エンジンについて述べた。性能測定の結果、携帯実装エンジンの十分な性能と、更なる機能搭載が可能なことを確認した。今後は、携帯電話にあわせた、リソース管理方式等の残課題検討を行い、実用的レベルに上げることを目指す。

関連図書

[1] S. Gribble, et al., "Ninja Architecture for Robust Internet-Scale Systems and Services," Special Issue of Computer Networks on Pervasive Computing, 2000.
[2] R. Masuoka, et al., "Ontology-Enabled Pervasive Computing Applications," IEEE Intelligent Systems, vol.18, no.5, pp.68-72, 2003.
[3] 武本充治, et al., "ユビキタスコンピューティング環境に適したサービス提供アーキテクチャにおけるサービス合成方式とその実装," 情報処理学会論文誌, Vol.46, No.2, pp.418-433, Feb. 2005.
[4] T. Oh-ishi, et al., "Network Services using Service-Composition Technology", International Symposium of Networks2004, Jun. 2004.
[5] OWL-S web site, http://www.daml.org/services/
[6] BPEL web site, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/