サービス連携システムの設計と実装

萩野浩明† 藤井邦浩† 村上純子†
原未來† 木本勝敏† 栄藤稔†

本稿では，サービス連携フレームワークを提案し，その実装システムである「BLOCCO」について説明する．筆者らが提案するフレームワークはイベント駆動型サービス実行と，パラメータハンドオーバの 2 つの基本的なメカニズムから構成される．BLOCCO は既に一般公開されている．本稿では，BLOCCO を利用したスケジューラ駆動型経路検索や手書きジェスチャー認識 Twitter などの実例により，サービス連携フレームワークの利便性と拡張性を示す．

Design and Implementation of the Service Facilitation System

Hiroaki HAGINO† Kunihiro FUJII†
Junko MURAKAMI† Mirai HARA†
Katsutoshi KIMOTO† Minoru ETOH†

We present a service facilitation framework and describe its implementation called “BLOCCO.” The framework consists of two basic mechanisms: 1. even-driven service invocation, and 2. parameter handover among services. By having those mechanisms, users can combine various services so as to program concatenated service scenarios. The implementation, BLOCCO, is completed and commercially available with Android phones. BLOCCO shows that service facilitation concept with practical examples such as scheduler-driven route search and a hand-written gesture driven twitter. Those examples prove convenience and extensibility of our framework.

1. Introduction

マッシュアップ[5]やコンテキストアウェアネス[3]と呼ばれる技術は様々なサービスを産み出す可能性を秘めた技術である．しかし，ユーザにとってはサービス開発者が提供するサービスを一方的に享受する立場であることは変わらず，それゆえ多種多様なユーザニーズに応えるには必ず限界が存在する．

本稿ではこの問題を解決するために，サービスのマッシュアップやコンテキストアウェアサービスをユーザが自由にカスタマイズできる「サービス連携」のフレームワークを提案する．さらにこの概念を具現化したシステムである「BLOCCO‡」について説明する．

以下では，まず 2 章においてサービス連携のフレームワークについて説明する．次に 3 章でサービス連携システムの設計について述べる．4 章では，筆者らが提案するフレームワークを具現化したシステムである BLOCCO について実例を交えながら議論する．最後に 5 章において本稿のまとめを行う．

2. Proposal of Service Facilitation Framework

2.1 Definition of Service Facilitation

本節において，まずサービス連携の定義を行う．筆者らはサービス連携を以下の 3 つの概念もしくは技術の組合せとして考える．

- マッシュアップ
- コンテキストアウェアネス
- ユーザカスタマイズ

すなわち，サービス連携とは，開発者ではなくユーザが，自由にサービスとサービスを組合せる（ユーザカスタマイズ）ことで，サービス間でのパラメータの受け渡しとそれに伴うサービスの振る舞いの制御（マッシュアップ・コンテキストアウェアネス）を実現可能な環境や技術を指す．

従来のマッシュアップはサービス開発者にとっての開発手段の 1 つに過ぎず，多種多様なユーザニーズに応えることは困難である．この問題の解決方法として，これまでにも，ユーザによるマッシュアップを実現する研究やサービスが存在する［1,6,7,9］が，ユーザにとっての設定の複雑さや，その仕様に対応するためにサービス開発者に求められるオーバヘッドのため，普及には至っていない．

† 株式会社 NTT DOCOMO
NTT DOCOMO, Inc
‡ BLOCCO は(株)GClue の登録商標

コンテキストアウェアネスの分野においても，あらかじめ開発者が設定したマッピングルールではなく，ユーザが自由にマッピングルールを設定可能とし，設定されたルールに従ってサービスの振る舞いを変更するサービスが存在する[4,8]．特に Locale [4]はサービスとサービスを組合せることで，実質的にユーザによるイベント駆動型プログラミングを可能にするものであり、筆者らが提案するコンセプトに近い．また，ユーザ設定は比較的シンプルな操作で行うことができる．しかし，あるサービスの状態を他のサービスの実行トリガとして利用はできるが，サービス間でパラメータを引き継いで利活用することはできない．一般的に，高機能であることと，操作のシンプルさや、開発者が対応サービスを開発する際のオーバヘッドとは、トレードオフの関係にある．筆者らが提案するコンセプトは，Locale に比べて高い機能を提供するが，それによって Locale のシンプルさを著しく損なうことは避けなければならない．

以上の分析から，筆者らはサービス連携フレームワークが満たすべき条件を次のように考える．

- 対応サービスを開発するための開発オーバヘッドを最小限に留める
- 高機能化に伴う操作オーバヘッドを最小限に留める

2.2 Service Facilitation Framework

サービス連携のフレームワークを図 1 に示す．図中のサービス実行シナリオは，以下の特徴をもつ．

- イベント駆動でサービスを実行できる
- サービス間のパラメータを受け渡しできる

サービス実行シナリオは，Event，Action，Input の 3 種類の要素で構成される．ユーザはサービス実行シナリオの各要素に対して，サービスを割り当てることができる．Event，Action，Input として利用できるサービスを，それぞれ Event サービス，Action サービス，Input サービスと呼ぶ．なお，このサービス分類はサービス連携システムから見たインタフェースの分類であり，例えば Event サービスと Input サービスを兼ねるサービスも存在しうる．以下に，3 種類のサービスについて説明する．

Event サービス: サービス実行シナリオの実行トリガを発行するサービス．1 つのサービス実行シナリオの中に，少なくとも 1 つ設定される必要がある．複数の Event サービスが設定されている場合，実行トリガは複数の Event サービスの AND 条件で定義される．

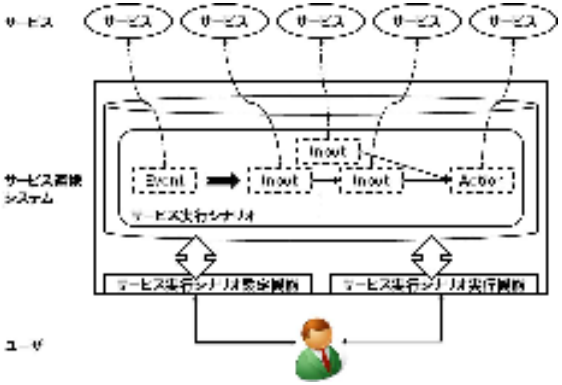

図 1 サービス連携のフレームワーク

Action サービス: サービス実行シナリオによって実行されるサービス主体．1 つのサービス実行シナリオの中に少なくとも 1 つ設定される必要がある．複数存在する場合，実行トリガが発行されると，全ての Action サービスが実行される．実行に必要なパラメータに対して，後述の Input サービスを設定できる．

Input サービス: Action サービスに対してパラメータを提供するサービス．Action サービスがサービス実行に必要とするパラメータに対して設定できる．また，Input サービスは複数連結することが可能である．例えば，他の Input サービスからパラメータを受け取り，そのパラメータを加工した上で Action サービスにパラメータを提供するサービスも，Input サービスとして定義する．Action サービスが複数のパラメータを必要とするサービスの場合，それぞれのパラメータに対して異なる Input サービスを設定することができる．また，サービス実行シナリオによっては Input サービスが存在しない場合もある．

サービス連携システムは，サービス実行シナリオの設定および実行の役割を担う．ユーザはサービス連携システムを介して，様々なサービスを Event，Input，Action に割り当ててサービス実行シナリオを設定できる．ユーザによって設定されたサービス実行シナリオはサービス連携システムによって管理される．サービス連携システムは管理するサービス実行シナリオの Event サービスが発行する実行トリガを監視し，実行条件が満たされたサービス実行シナリオを実行する．サービス実行シナリオの実行時は，Input サービスから Action サービスへのパラメータ受け渡しを仲介する．

3. Design of Service Facilitation System

本章では，サービス連携システムの設計について述べる．

3.1 Screen Transition

2 章で述べたように，筆者らが提案するサービス連携フレームワーク，およびそれを具現化するシステムでは，ユーザがサービス実行シナリオを設定する際の，シナリオ設定の容易さが重要なポイントとなる．また，筆者らはサービス連携システムは，携帯電話上で利用されることを想定する．これは常時ユーザの傍らにあることが多い携帯電話がコンテキスト獲得に適しているからである．しかし，携帯電話は画面が小さいため, 複数の情報を同時に表示することが困難である. そこでまず，サービス実行シナリオを設定する際の画面遷移を定め，その上でサービス連携システムの設計を行う．なお，シナリオ実行時はバックグラウンドでサービス間のパラメータ受け渡しを実行するため，特に画面遷移は発生しない．シナリオ設定の際の画面遷移を図 2 に示し，以下で説明する．

1. ユーザがサービス実行シナリオを設定する際，画面には Event 設定と Action 設定を行うためのボタンもしくはリンク（以下ではボタンで統一）が表示される．
2. ユーザが Event 設定のボタンを押下すると，Event サービスとして利用できるサービスの一覧が表示される．
3. ユーザが, 表示された中から Event サービスを選択すると, 選択された Event サービスの詳細設定画面に遷移する．
4. ユーザが Event サービスの詳細設定を終えると，再び最初の画面に戻る．ユーザが次に Action 設定のボタンを押下すると, Action サービスとして利用できるサービスの一覧が表示される．
5. ユーザが表示された中から Action サービスを選択すると, 選択された Action サービスの詳細設定画面に遷移する．Action サービスの詳細設定画面には，Action サービスを実行するために必要なパラメータを設定するためのボタンが表示される．
6. ユーザがパラメータを設定するためのボタンを押下すると，そのパラメータと同じパラメータ型のパラメータを提供する Input サービスの一覧が表示される．

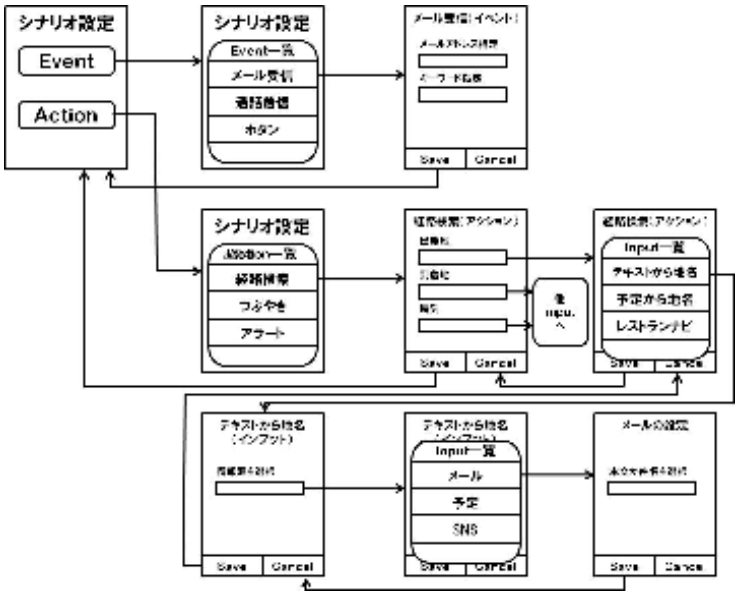

図 2 シナリオ設定時の画面遷移

7. ユーザが表示された中から Input サービスを選択すると，選択された Input サービスの詳細設定画面に遷移する．Input サービスには，さらに他の Input サービスからパラメータを受け取ることができるものもある．その場合，Action サービスと同様に，ユーザがパラメータを設定しようとすると，Input サービスの一覧が表示される．
8. Input サービスの設定が終了すると, Action サービスの詳細設定画面に戻る．Action サービスの詳細設定画面において必要なパラメータを全て設定すると，最初の画面に戻り，サービス実行シナリオの設定が終了する．

以下では, この画面遷移に基づき, サービス連携システムと Event サービス, Action サービス，Input サービスとの間のプロトコルインタフェース，およびパラメータハンドリングについて説明する．

3.2 Protocol Interface

サービス連携システムは Event サービス，Action サービス，Input サービスと通信することで，サービス連携を実現する．図 3 にサービス連携システムと各種サービス間のプロトコルインタフェース一覧を示し，以下でそれぞれのプロトコルインタフェースについて説明する．

3.2.1 Event サービスのプロトコルインタフェース

Event 呼び出し/設定インタフェース: ユーザがサービス実行シナリオを設定する際に Event を設定しようとすると，サービス連携システムはユーザが指定した Event サービスの詳細設定画面を呼び出す．ユーザが Event サービスの詳細設定を終えると，Event サービスは設定終了の通知とともに設定された値をサービス連携システムに送信する．

トリガ発行インタフェース: Event サービスが Event の発生をサービス連携システムに通知する．

状態問合せ/応答インタフェース: サービス実行シナリオに複数の Event サービスが設定されている場合，サービス連携システムがその中の 1 つからトリガを受信すると，同じサービス実行シナリオに設定されている他の Event サービスに対して状態を問合せる．問合せを受け取った Event サービスは問合せ内容と状態が一致するかどうかを判断し，その結果をサービス連携システムに通知する．これによって，Event サービスは状態が変化したタイミングでのみ能動的に動作すればよい．

3.2.2 Action サービスのプロトコルインタフェース

Action 呼び出し/設定インタフェース: ユーザが Action サービスを設定しようとすると，サービス連携システムはユーザが指定した Action サービスの詳細設定画面を呼び出す．ユーザが詳細設定を終えると，Action サービスは設定終了の通知とともに，設定された値をサービス連携システムに通知する．

パラメータマッチングインタフェース: ユーザが Action サービスのパラメータを設定しようとすると，Action サービスはサービス連携システムに対して，同じ型のパラメータを提供できる Input サービスのリストを要求する．サービス連携システムは要求に対して，該当する Input サービスのリストを生成し，Action サービスに送信する．

図 3 プロトコルインタフェース

Action 実行インタフェース: サービス連携システムは，サービス実行シナリオを実行する際，Input サービスから取得したパラメータを引数として，Action サービスに対して Action の実行を指示するメッセージを送信する．

3.2.3 Input サービスのプロトコルインタフェース

Input 呼び出し/設定インタフェース: ユーザが Action サービスのパラメータとして Input サービスを設定しようとすると，サービス連携システムはユーザが指定した Input サービスの詳細設定画面を呼び出す．ユーザが詳細設定を終えると，Input サービスは設定終了の通知とともに，設定された値をサービス連携システムに通知する．

パラメータマッチングインタフェース: Action サービスのパラメータマッチングインタフェースと同様．

パラメータ問合せ/取得インタフェース: サービス連携システムは，サービス実行シナリオを実行する際，Input サービスに対して，他の Input サービスや，Action サービスに渡すためのパラメータを問合せる．その際，先に他の Input サービスから取得したパラメータを引数として渡すこともできる．Input サービスは問合せを受け取ると，パラメータを生成してサービス連携システムに対して送信する．

3.3 Parameter Handling

サービス連携システムはサービス間でパラメータをやりとりするために，パラメータの型を定義する．パラメータの型には，text 型などのいわゆるデータ型に相当

するものや，駅名，地名，時刻，日付などの意味を表すものが存在する．Action サービスを開発する開発者は，Action サービスを実行する際に必要となるパラメータごとに，受け取ることができるパラメータの型を定義する．また，input サービスを開発する開発者は，Action サービスと同様に，Input サービスが必要とするパラメータごとに，受け取ることができるパラメータの型を定義し，さらに Action サービスや他の Input サービスに渡すことができるパラメータの型も定義する．サービス連携システムはこれらの定義に基づいて，シナリオ設定時のパラメータマッチングやシナリオ実行時のパラメータ受け渡しを実行する．パラメータ型の詳細は，文献 2)に示している．

4. Service Facilitation System “BLOCCO”

筆者らは 3 章に述べた設計に基づいて，Android アプリケーションとして，サービス連携システム「BLOCCO」を（株）GClue と共同で開発した．BLOCCO は Android マーケットからダウンロード可能である．本章では BLOCCO について説明し，筆者らが提案するフレームワークの有効性を検証する．

4.1 Outline of BLOCCO

BLOCCO は，他の Android アプリケーションを Event サービス，Action サービス，Input サービスとして扱うことができる．BLOCCO では，これらのサービスをプラグインと呼んでいる．ユーザがダウンロード後すぐに利用できるように，BLOCCO はあらかじめいくつかの基本的なプラグインを同梱している．

BLOCCO の画面例を図 4 に示す．ユーザは BLOCCO を用いてサービス実行シナリオを設定する．その際の画面遷移は図 2 に従う．ユーザがサービス実行シナリオを設定しておくと，BLOCCO が各 Event プラグインの状態を監視し，実行条件が満たされたシナリオを実行する．

BLOCCO は，プラグインとなるアプリケーションがインストールされていることを自動的に検出できる．これによって，ユーザがサービス実行シナリオを設定する際にプラグインの一覧を提示することができる．

シナリオ設定時に各種プラグインの設定画面を呼び出す機能や，シナリオ実行時の Event プラグイン監視やプラグイン間のパラメータ受け渡しは，Android OS が提供する intent および broadcast receiver の仕組みを用いて実現している．

4.2 Convenience of Service Facilitation

BLOCCO を利用することで，アプリケーションの組合せで以下のようなシナリオを実現できる．

図 4 BLOCCO の画面例

シナリオ例1.

スケジューラに件名「飲み会」，時間「19-22」，場所「六本木」と登録しておくと．1 時間前に今いる場所から六本木に 19 時に到着できるような電車の経路検索を自動で実行してくれる．

シナリオ例2.

家に帰ると，手書きジェスチャーアプリケーションを起動し，画面に”○”を描くと自動的に Twitter に「帰宅なう」とつぶやいてくれる．

シナリオ例 1 における Event，Action，Input の割り当てを図 5 に示す．シナリオ例 1 では，Event プラグインにスケジューラアプリ，Action プラグインに経路検索アプリが割り当てられている．また，経路検索アプリが必要とするパラメータのうち，「到着駅」と「時刻」については，Event プラグインであるスケジューラアプリが Input プラグインを兼ねている．一方「出発駅」には GPS アプリが割り当てられている．このように，BLOCCO を利用すると，なんらかのイベントが発生したときに，あらかじめシナリオに設定しておいたアプリケーションを自動的に実行することができる．同様の例として，例えば友人からイベントのお誘いメールが届いたときに自動的に経路検索するなど，様々な応用が考えられる．また，Input プラグインがシナリオ内に単一ではなく，Action プラグインが必要とするパラメータごとに設定可能なことも BLOCCO の特徴である．同様の仕組みにより，例えば複数のアプリケーションから別々に文字列を取得し，それらを繋ぎ合わせて一つの文字列を生成することなどが可能となる．

シナリオ例 2 における Event, Action の割り当てを図 5 に示す. シナリオ例 2 では，手書きジェスチャーアプリケーションが Event プラグイン，Twitter クライアントア

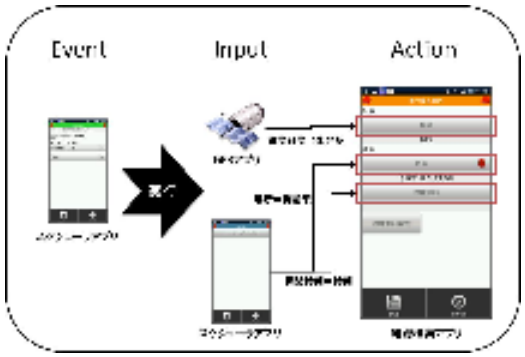

図 5 シナリオ例 1 におけるプラグインの割り当て

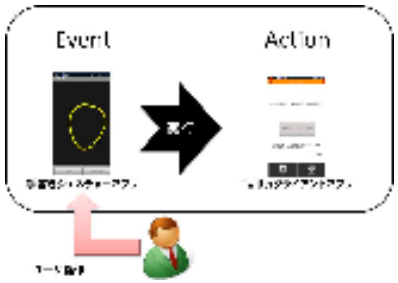

図 6 シナリオ例 2 におけるプラグインの割り当て

プリが Action アプリに該当する．この例は，BLOCCO がアプリケーションの自動実行だけでなく，半自動実行やユーザ操作のショートカットも可能であることを示している．BLOCCO ではシナリオの実行トリガを Event プラグインが発行するものとして定義している．これにより端末等の状態を監視するアプリケーションが Event プラグインに設定されている場合はアプリケーションの自動実行が可能であり，ユーザの操作によって実行トリガを発行するアプリケーションが Event プラグインに設定されている場合は，半自動実行やユーザ操作のショートカットとして利用できる．

このように，柔軟なサービス実行シナリオを設定できる機能やサービス間のパラメータ受け渡し機能を実現することで，BLOCCO はユーザがマッシュアップやコンテキストアウェアサービスを自由にカスタマイズできるプラットフォームとなっている．

4.3 Extensibility of Service Facilitation

アプリケーションがプラグインとして動作するためには，3 章で述べたインタフェースが必要となる．筆者らは 3 章で述べたプロトコルインタフェースの仕様と，BLOCCO 対応プラグインを開発するための BLOCCO SDK を公開している[2]．この SDK を利用すると，インタフェース部分のソースコードの雛形を自動生成できるため，プラグイン開発にかかるオーバヘッドは大幅に小さくなる．

本来サービス連携フレームワークがもつ，独立するサービス同士の連携という概念に加え，プラグイン開発に必要な仕様を公開し，開発環境を提供することで，BLOCCO は高い拡張性を提供している．

5. Conclusion

本稿では，サービス連携のフレームワークを提案し，フレームワークの具現化のために実装したサービス連携システム「BLOCCO」の設計と実装について述べた．BLOCCO の利便性と拡張性について検証することで，筆者らが提案するフレームワークの有効性を示すことができた．

謝辞 本研究を進めるにあたり，フレームワークの具現化に尽力頂いた（株）GClue の佐々木氏に深く感謝する．

参考文献

1) Accelerators, http://www.microsoft.com/windows/internet-explorer/readiness/developers-new.aspx
2) Blocco, http://www.blocco.jp
3) Dey, A.K. and Abowd, G.D.: Toward a Better Understanding of Context and Context-Awareness, Proceedings of the CHI2000 (2000)
4) Locale for Android, http://www.twofortyfouram.com/
5) O'Reilly, T.: What Is Web 2.0, http://oreilly.com/web2/archive/what-is-web-20.html (2005)
6) Plagger, http://plagger.org/
7) 田中譲: ミームメディア・アーキテクチャ IntelligentPad とその応用，情報処理，Vol.38, No.9, pp.222-231 (1997).
8) Toggle Setting, http://andronavi.com/2010/01/5948
9) Yahoo! Pipes, http://pipes.yahoo.com/pipes/