修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報システム学研究科　情報ネットワークシステム学専攻　博士前期課程		
氏　　名	重信　裕政	学籍番号	1252020
論 文 題 目	包括的マッシュアップフレームワーク IDUMO の拡張と評価		

要　旨

Web サービスと情報端末の発展が，我々の生活に豊かさをもたらしている．各地の天気予報や鉄道運行状況がリアルタイムに提供されるように，スマートフォンやタブレット端末の普及とモバイルネットワークの発展が，受信場所を問わない遠隔リソースへの柔軟なアクセスを実現している．外部の情報リソースと連携したデータ保管およびインターネットの双方向性を生かしたユーザからの情報発信が，ソーシャルメディアサービスの活発化を促進している．

これらの情報基盤の発展は，アプリケーションの利用形態や個人で扱える情報量を増加させた．主体的に情報にアクセスすることで，RSS リーダやガジェットツールのような Web サービスやリソースを組み合わせて，新しいサービスを作成するマッシュアップの概念が生まれた．マッシュアップを利用する利点としては，低い開発コストでユーザの望んだことが実現できる点である．また，マッシュアップツールを用いることで自らの手でスマートフォンで実行できるアプリケーションを開発することも可能である．

しかし，既存のマッシュアップツールはエンドユーザを対象としたものは，開発容易性を優先することで，開発可能なアプリケーションの自由度に制限がある．プログラミングユーザを対象としたものは，汎用性が高いものの，プログラミング等の知識が不可欠なため，エンドユーザには扱えないものとなっている．先行研究においてこの容易性と汎用性の両立を目指したマッシュアップフレームワーク IDUMO を提案，実装を行い，あらゆるユーザがアプリケーション開発可能となるように複数の開発環境を提供する仕組みを提案した．

先行研究段階では，開発環境の分割化を提案しただけにとどまり，実装を行っていなかった．また，エンドユーザが開発環境を使用していくうちに次のレベルへと成長するためのユーザ補助等を検討していなかった．

本研究では，IDUMO で提案していたユーザのレベル別開発環境の実装，エンドユーザが開発環境を使うことで次のレベルへと成長するためのユーザ補助の実装を行った．また，既存のマッシュアップツールとの比較，実際にユーザに使用してもらい，アンケート評価を行った結果より，IDUMO の有用性を確かめた．

平成２５年度修士論文

包括的マッシュアップフレームワーク IDUMOの拡張と評価

大学院情報システム学研究科
情報ネットワークシステム学専攻

学 籍 番 号 :　1252020
氏 名:　重信 裕政
主任指導教員: 吉永 努 教授
指 導 教 員 :　入江 英嗣 准教授
指 導 教 員 :　笠井 裕之 准教授
提出年月日 :　平成２６年１月２７日

(表紙裏)

目 次

図 目 次

表 目 次

第1章　序論

Web サービスと情報端末の発展が，我々の生活に豊かさをもたらしている．各地の天気予報や鉄道運行状況がリアルタイムに提供されることが示すように，スマートフォンやタブレット端末の普及とモバイルネットワークの発展が，受信場所を問わない遠隔リソースへの柔軟なアクセスを実現している．外部の情報リソースと連携したデータ保管と計算処理，およびインターネットの双方向性を生かしたユーザからの情報発信が，ソーシャルメディアサービスの活発化を促進している．

これらの情報基盤の発展は，アプリケーションの利用形態や個人で扱える情報量を増加させた．天気情報を扱うアプリケーションを例にあげると，発信された天気情報を画一的に受け取るほかに，自ら主体的にアクセスして情報を取得できる．取得できる情報も現在地情報を利用したリアルタイムな天気情報や，勤務先の決まった時刻の天気情報など様々である．また，このように主体的に情報にアクセスすることで，RSS リーダやガジェットツールのような Web サービスやリソースを組み合わせて，新しいサービスを作成するマッシュアップの概念が生まれた．マッシュアップを利用する利点は，低い開発コストでユーザの望んだことが実現できる点である．また，BLOCCO[1] や on{X}[2] などのマッシュアップツールのように自らの手でスマートフォンで実行できるアプリケーションを開発することも可能である．

しかし，既存のマッシュアップツールの開発ではプログラミングユーザを対象としたものが多く，プログラミング知識のないエンドユーザにとって必ずしも容易ではない．一方，エンドユーザを対象としたツールは機能を限定しており，自由なアプリケーション開発が行えない．既存のマッシュアップツールでは，汎用性と開発容易性を兼ね備えていないといえる．そこで先行研究において，自由なアプリケーション開発が行える汎用性とエンドユーザが簡単に開発を行える容易性を両立するマッシュアップフレームワーク IDUMO[3][4][5][6] を提案した．

IDUMO を用いて，汎用性と開発容易性を兼ね備えたアプリケーション開発を行うことが可能となったが，一方で，エンドユーザが簡単にアプリケーション開発可能な支援が十分ではない．エンドユーザにとってモジュールを接続するだけというプログラミングレスで簡単な開発であっても，どのモジュールとモジュールが接続可能なのかという操作に関して一切の補助がないと，開発が行えない可能性がある．エンドユーザがスムーズにアプリケーション開発を行えるような補助を適切に設計する必要がある．

本研究では，先行研究で提案していたユーザのレベル別開発環境の実装，IDUMO をよりエンドユーザが利用しやすいようにモジュール接続をわかりやすくするための機能の追加や，新たにモジュールを開発する際の支援を行うシステムを作成し，実際にユーザに使用してもらった評価をもとに IDUMO の有用性を示す．

本論文は以下のように構成される．まず 2 章にて既存のマッシュアップの関連研究と先行研究における IDUMO の設計を述べる．3 章にて IDUMO の改善点をまとめ，4 章にて追加した実装の説明をする．5 章では IDUMO を使用したユーザの評価と既存のマッシュアップツールとの比較を行い，考察する．最後に 6 章にて本研究をまとめる．

第2章　マッシュアップ関連技術

2.1　既存のマッシュアップツールの分類

現在，様々な種類のマッシュアップツールが提供されている．本論文では，これらのマッシュアップツールを以下の 3 つに分類する．

- データを集約するマッシュアップ
- データを生成するマッシュアップ
- アプリケーションを作成するマッシュアップ

2.1.1　データを集約するマッシュアップ

データを集約して，情報を提示するマッシュアップツールとして，Netvibes[7] や MyYahoo![8] がある．これらのマッシュアップツールでは，自分が望む情報を選択し，1 つの画面に集約する．ユーザは画面を遷移することなく，様々なサービスの利用や情報の取得が可能となるが，自プラットフォーム以外のサービス間連携を行うことはできない．

2.1.2　データを生成するマッシュアップ

サービスで提供されているリソースから，自らが望むデータを生成するマッシュアップツールとして Yahoo!Pipes[9] や Dapper[10] などがある．Yahoo!Pipes は，WebAPI や RSS 配信されているニュースサイトの XML や JSON で表現されるデータを自らの望む形式に変換，加工を行うフィードアグリゲータである．Dapper は，Web ページの HTML を解析して，自らが望むデータ構造に変換することが可能なサービスである．

このマッシュアップ手法は，他のサービスリソースを利用・閲覧しやすくするアプローチをとっている． これにより，自らの望むデータ形式に変換し利用することができるが，実際に動作するアプリケーションを作成することはできない．

2.1.3　アプリケーションを作成するマッシュアップ

WebAPI や端末のセンサを用いて特定の端末上で動作するアプリケーションを開発できるマッシュアップツールとして，on{X}，BLOCCO，JointApps[11]，IFTTT[12]，Plagger[13] などがある．本論文で提案している IDUMO もここに分類される．これらはツール側が提供する開発環境内にて，モジュールの接続や設定を行いアプリケーションの開発を行う． このマッシュアップ手法では，WebAPI やデバイス上のセンサを組み合わせることで，端末上で動作するアプリケーションが

開発可能であるが，前述のマッシュアップツールに比べ，開発環境やマッシュアップ操作がより複雑になる．

2.2 マッシュアップ関する研究

マッシュアップツールを用いることでアプリケーションを開発することができるが，(1) マッシュアップを実現するための問題や (2) ユーザが使用する際の問題がある．問題 (1) では，複数のサービスの連携や，やり取りするデータ形式をユーザが意識しなければならないということがあげられる．問題 (2) では，ユーザのレベルに合わせた視覚的な開発環境を統合的に提供できていない点や，エンドユーザにも開発可能なマッシュアップツールでは，機能に制限があるということがあげられる．

複数のサービスを連携する場合，それぞれのアクセス方式が異なるため，連携が複雑になる．例えば，ネットワーク上のサービスを用いる場合に REST を用いた XML ベースのアクセスを行い，デバイス上のセンサを用いる場合は，シリアル通信によって生ビットストリームを取得するなどがある．森ら [14] はマッシュアップによるアプリケーション開発を，対象と接続に分割するプログラミングスタイルを提案している．横山ら [15] は，Javascript プログラムを機能単位でカプセル化するモジュールを提案し，複数のモジュールに対して属性情報を結合するのみでサービス連携を可能とする手法を提案している．既存のマッシュアップツールでも，多くがこの方式を採用しており，内部をブラックボックスとして，データの接続を行うことでサービス間連携を行っている．しかし，様々なサービスの連携を行いやすくなる一方で，やり取りするデータ形式をユーザが把握しなければならず，エンドユーザが利用するには敷居が高い．

やり取りするデータ形式は，使用する API によって異なり，それを自動変換することができない．下條ら [16] は，様々な種類のライフログを構成するデータ項目を分析し，ライフログの標準データモデルの提案と変換 API の実装を行っている．森ら [17]，Sabbouth ら [18]，Maximilien[19] らは異なる API を結合する際にドメイン固有言語を用いたデータ接続を提案している．これらのアプローチは，作成されたデータフォーマットを用いることで多くのデータ構造を統一的に扱えるが，既存のマッシュアップツールでは統一的なデータフォーマットを使用せず，ツール内のプログラミングにおいて扱いたい構造に変換したり，そのまま生データを扱うなど用途に応じて様々である．BLOCCO では，マッシュアップモジュール間で授受するデータを位置情報を表すデータ構造といったような，抽象化された高機能なデータ構造を定義して，データの受け渡しを自動化している．このことが原因で接続するモジュールを限定的にして自由なアプリケーション開発を行えない．

マッシュアップのモジュールを作成する際の開発に関して，幸城ら [20] はマッシュアップ開発の際に重複する処理を簡潔化のために，アスペクト指向プログラミングを導入し，マッシュアップ開発を容易にする手法を提案している．横山ら [15] は，Jaxascript を用いたマッシュアップを行うためのモジュール自身の開発を容易にするために，Ajax や JSON といった処理を行う際の典型的なコードを自動生成する仕組みが必要であると述べている．

このようにマッシュアップモジュールを作成する際には共通の処理が含まれていることが多く，開発者のモジュール作成の負担を減らす必要がある．

利用するユーザのレベルに合わせた開発アプローチにおいて，今入ら [21] は，ユーザを一般ユーザ，開発経験者，開発者と分類し，それぞれのユーザが利用するマッシュアップツールが異なり，統合的な開発環境をマッシュアップツールが提供できていないことが問題であると述べている．ま

た，適切な視覚的マッシュアップ手法の提案を行えば，すべてのユーザが統一的に利用可能なマッシュアップツールが開発可能であると提案している．

既存のマッシュアップツールのうち，アプリケーションを開発するマッシュアップツールを，開発容易性と開発できるアプリケーションの汎用性とで配置した図が図 2.2.1 のようになる．例えば，on{X} では，開発者がプログラミングによって作成したアプリケーションを，一部カスタマイズ可能な形でユーザに提供している．しかし，エンドユーザがサービスの連携構造を変えることはできず，アプリケーション開発者にステップアップする敷居は高い．開発容易性は高いが，アプリケーションの自由度は低いといえる．BLOCCO は，開発者が作成したモジュールを用いて，エンドユーザが視覚的な環境でマッシュアップすることが可能である．ただし，機能を限定しているため自由な開発を行えているとは言えない．

ほとんどのマッシュアップツールがユーザ獲得のために開発容易性に重きを置いている．一方で，Plagger のように汎用性が高いマッシュアップツールも存在している．しかし，Plagger は導入や開発に Perl を必要としているため，開発容易性は高いとは言えない．このように，開発容易性に重きを置いたマッシュアップツールは開発の自由度，汎用性はあまり高くなく，汎用性を優先すると開発容易性に乏しくなるという特徴がある．

2.3 先行研究における IDUMO

2.3.1 IDUMO フレームワーク

前節にてあげた問題点を解決するために，IDUMO ではマッシュアップによりあらゆるデバイスや情報を包括的に扱い，アプリケーション開発を可能とする汎用性と，エンドユーザでも簡単にマッシュアップを可能とする容易性を両立を行った．IDUMO の全体像を図 2.3.1 に示す．

問題 (1) を解決するために,「統一的入出力インタフェース」を備えたモジュールを導入する．これを用いて，端末や Web サービスの機能特有の処理に依存しない形で，様々なサービスやセンサを接続可能とする．そして内部で扱うデータの授受は「抽象化データフォーマット」を用いてやり取りする．これによって，データ形式を気にしない簡単なサービス接続と生データでのやり取りによる自由な接続を両立させる．問題 (2) の解決のためにマッシュアップの記述を最少化する簡単な「マッシュアップ記述仕様」を定義する．また,「統一的実行モデル」を備えた IDUMO バーチャルマシンにより，アプリケーションを様々なプラットフォーム上で実行できるようにする．

統一的入出力インタフェース

マッシュアップで扱う様々な種類のサービスのリソースを統一的に扱うために，リソース間の接続をデータのやり取りのみとするデータフォーマットを用いる．インターフェースはデータを受け渡す Sendable と，データを受け取り Receivable に分割している．他のモジュールにデータを受け渡すモジュールは Sendable を，他のモジュールからデータを受け取るモジュールは Receivable を実装する．

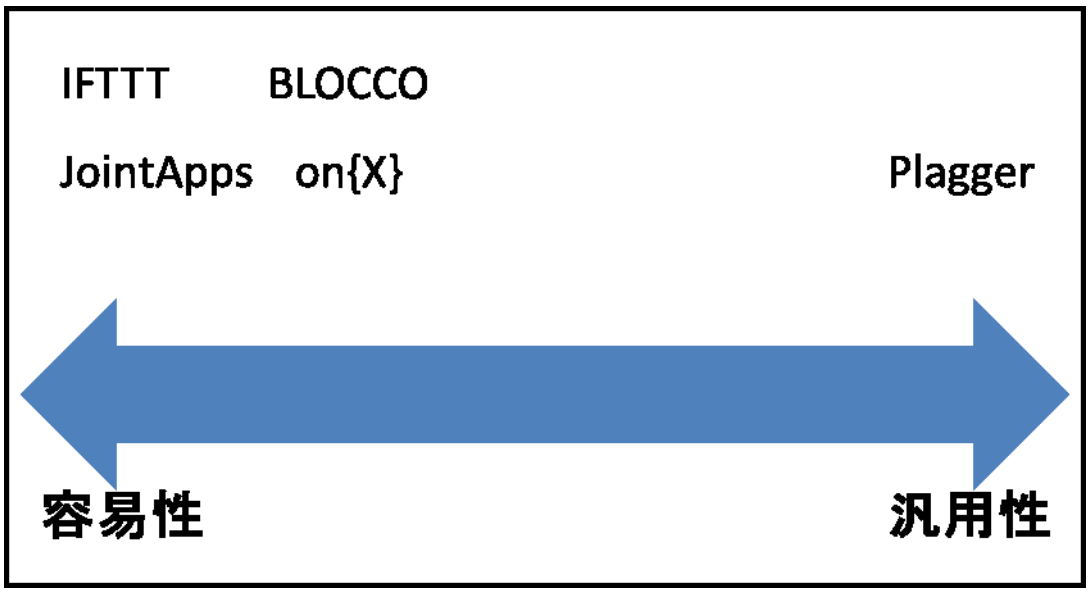

図 2.2.1: 既存マッシュアップツールの分別

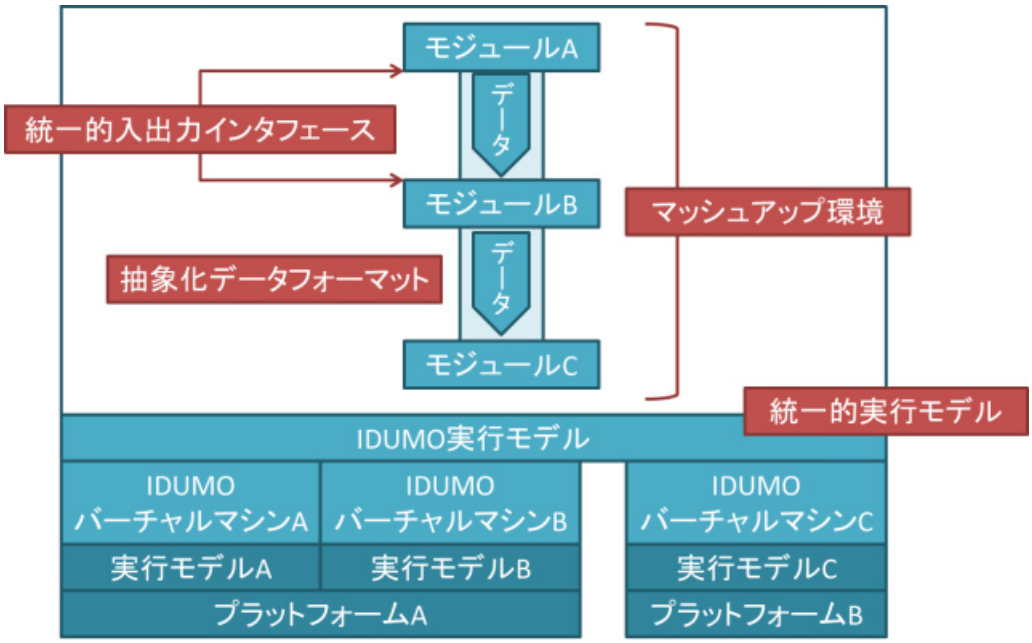

図 2.3.1: IDUMO フレームワークの全体像

抽象化データフォーマット

モジュール間でやり取りされるデータは抽象化データフォーマットを用いる．これはデータの保存自体をハッシュデータとして保存することで生データへアクセス可能とする．さらにデータ構造をインタフェースとして定義し，各データフォーマットにインタフェースを実装することによって，そのデータフォーマットがどのような型情報を表すのか示す．データ構造を扱うとともに，生データへのアクセスも可能であり，容易性と汎用性を兼ね備えている．

マッシュアップ記述仕様

マッシュアップに情報を簡単に記述するために，XML を中間言語として用いる．ここではマッシュアップの記述を最少に抑えるために，開発の際に「どのような機能を持つアプリケーションを作成したいか」という情報と,「どのようにアプリケーションを動作させるか」という情報を記述できるようにする．そこでどのようなアプリケーションの機能を表す「モジュールの定義」,「モジュール間の接続情報」とアプリケーションの動作を表す「繰り返し実行の可否」が記述されるようになっている．

統一的実行モデル

実行モデル毎に非依存なアプリケーションの開発を実現するため，アプリケーションをマッシュアップにより記述されたアプリケーション手順を表す実行モデルに非依存な部分と，アプリケーション手順を実際にデバイス上で実行させる部分に分離している．デバイスに非依存な部分にはモジュール定義，モジュールの接続，モジュールの実行が単一回か無制限かといったアプリケーションそのものの処理を記述する．その記述をデバイスに依存する部分が解釈することでアプリケーションを実行する．

2.3.2 開発手法

IDUMO の開発は,「GPS を取得する」,「テキストに表示する」などといった目的ベースのモジュールを Web アプリ上に実装したグラフィックユーザインタフェース (GUI) を用いてブロックを線でつないでいくというスタイルで行っていく．ユーザはモジュールをつなぎ，繰り返し回数，アプリケーション名を設定し，Download ボタンを押すことでダウンロード可能となる．ダウンロードボタンが押された後，サーバはユーザのマッシュアップ情報を XML に変換し，実際のプログラミング言語へ変換する．プログラミング言語へ変換されたマッシュアップ情報を IDUMO のモジュールやフレームワークと関連付け，実行可能なアプリケーションを作成し，ユーザへ提供する．作成されたアプリケーションは端末にインストールされ，IDUMO サーバとは独立して動作するため，サーバが稼働していなくてもユーザには影響を与えない．また，開発環境を「マッシュアップユーザ」,「カスタマイズユーザ」,「一般ユーザ」とユーザのレベルに分けて提供することで，それぞれのユーザが自分のレベルに合わせた開発環境を提案している．また，使用されるモジュールは IDUMO の開発用 SDK を利用することで，プログラミングができるユーザによって新しく追加することが可能となっている．

第3章　IDUMOの改善点

3.1　既存マッシュアップツールの問題点

JointApps や on{X}，IFTTT といったマッシュアップツールは商用利用を目的としている．商業的にマッシュアップツールを成功させるために，開発容易性を追求し，どのようなユーザでも簡単にアプリケーションを開発できるようにしている．そのため，エンドユーザにでも簡単にアプリケーションが開発でき，ユーザの獲得に成功している．しかし，開発容易性を追求する過程で，ユーザが利用できる機能や開発できるアプリケーションの種類を限定しているため，汎用性を損なっている．

一方で Yahoo!Pipes や Plagger といったマッシュアップツールはプログラミングができるユーザが利用することを想定しているため，汎用性が高いアプリケーションを開発可能である．しかし，プログラミングの知識，データのやり取りの形式の把握といった技術的な知識を持っていることを前提としているため，それらの知識のないエンドユーザには敷居が高い．

開発容易性を優先することで，汎用性が限定され，汎用性を優先することでエンドユーザにとって利用しがたいツールとなってしまっている．この問題として，既存のマッシュアップツールのほとんどの開発環境が 1 種類であることが原因となっていると考える．ユーザのレベルによってプログラミング等の知識量は様々である．開発環境を 1 種類しか提供しなければ，その開発環境を利用できるレベル以上のユーザしか継続して利用しないと考えられる．そのレベルに到達していないユーザの中にはアプリケーション開発に関する興味があったとしても,「自分には難しすぎる」と結論付けてしまう．マッシュアップツールをあらゆるユーザに利用してもらうためには，幅広いレベルのユーザが利用できる開発環境が必要だと考えられる．

3.2　先行研究段階の IDUMO における評価・考察

先行研究の段階における IDUMO について，評価・考察を行い，問題点と改善点を挙げていく．先行研究において，IDUMO はマッシュアップにおけるデータのやり取りやサービス間の連携といった問題点を 2 章で記述した手法にて解決している．一方で，実際のアプリケーション開発環境においては改善の余地が残っている．まず，IDUMO ではユーザのレベルに応じた開発環境を提供すると提案しているが，実装には至っていない．様々なレベルのユーザがアプリケーション開発を行うために，このユーザレベルに応じた開発環境を実装する必要がある．また，IDUMO で実装されている開発環境ではマッシュアップに用いるモジュールの種類を Provider，Handler，Adaptor，Receiptor とデータの授受の種類によって分別している．しかし，モジュール同士の接続を行うにあたって，どのモジュールとモジュールが接続可能かどうかという部分は，実際に接続して動かしてみないとわからない．接続可能かどうかの判定を実際に動作するまでわからないというのは，効率的ではないし，ユーザの混乱を招く．事前に接続可能なモジュールをユーザに提示する仕組みが必要である．また，IDUMO の特徴として，ユーザ自身が新たにモジュールを作成可能という

ものがある．しかし，モジュール作成に関しては，プログラミング知識のあるユーザが，既存のモジュールを見ながら見よう見まねで作成しなければならない．IDUMO のデータのやり取りといった共通の記述をその度に書かなければならないのは，ユーザにとって負担となるといえる．

先行研究における IDUMO の問題点をまとめると次のようになる．

- ユーザのレベルに応じた開発環境が実装されていない
- 接続可能なモジュールがユーザにわかりにくい
- モジュール作成を簡略化できていない

これらの問題点をもとに IDUMO の改善点を提案していく．

第4章　改善案の実装

4.1　IDUMOにおける提案

3章で述べた問題点を解決する手法として，ユーザのレベルに応じた開発環境を提供するという方法を提案する．IDUMOでは，エンドユーザでも簡単にアプリケーションが開発できる容易性を持ちながら，汎用的なアプリケーション開発を行うことを目標としている．既存のマッシュアップツールのように開発環境が1種類では容易性と汎用性を兼ね備えたアプリケーション開発は行うことができない．

IDUMOでは「マッシュアップユーザ」,「カスタマイズユーザ」,「一般ユーザ」にユーザのレベルを分けて開発環境を提案している．また,「一般ユーザ」がIDUMOを利用していくことで「カスタマイズユーザ」,「マッシュアップユーザ」と成長することでより自らが開発したアプリケーションで自らの要求を満たすことが可能となると考え,「一般ユーザ」向けの開発環境から「カスタマイズユーザ」向けの開発環境へ，さらに「マッシュアップユーザ」へ向けた開発環境へと成長を促せるような実装を目指す．モジュール開発者が新たにモジュールを作成する際にも開発者の負担を軽減する仕組みも実装していく．

4.2　開発者のレベル分け

IDUMOでは幅広いレベルのユーザがアプリケーションを容易に開発することが求められる．よって開発ユーザを次のようなレベルのユーザとして分類していく．

4.2.1　一般ユーザ

一般ユーザは全くプログラミングの知識もマッシュアップの経験もないユーザと定義する．このレベルのユーザはアプリケーションの開発を行うことは非常に難しいと考えられる．IDUMOでは，このレベルのユーザにはアプリケーション開発の流れを確認して次のレベルへ成長してもらうことを目的とした開発環境を提供する．

4.2.2　カスタマイズユーザ

カスタマイズユーザは一般ユーザから成長したレベルのユーザである．このユーザは，ある程度マッシュアップによるアプリケーション開発の経験と知識を取得しているユーザと定義する．また，プログラミング知識についてはある程度あるレベルのユーザと定義する．これは，プログラミングの処理の流れなどを理解していることでマッシュアップの接続の流れ等がスムーズに理解できると考えられるからである．このレベルのユーザにはモジュールの付け替えや追加といった

マッシュアップ開発の作業の一部分を体験することで，マッシュアップ開発の経験やノウハウを習得し，次のレベルへと成長してもらうことを目的とした開発環境を提供する．

4.2.3 マッシュアップユーザ

マッシュアップユーザはマッシュアップ開発を十分に経験したユーザと定義する．プログラミング知識を十分に持ったユーザであれば，モジュールの接続の流れ等が容易に理解できると考えられるため，いくつかの開発例を確認するだけでこのレベルからのスタートが行える．しかし，IDUMOではプログラミングレスな開発環境でのアプリケーション開発なため，高度なプログラミング知識はこのレベルのユーザに対しても求めてはいない．このレベルのユーザは，1 からモジュールを選択し，接続をしていくことでアプリケーション開発が行えるような環境を提供する．

4.3 ユーザのレベルに合わせた開発環境の実装

「一般ユーザ」には Example,「カスタマイズユーザ」には Customize,「マッシュアップユーザ」には Mashup という開発環境を実装した．各開発環境を図 2.2.1 に追加すると図 4.3.1 のようになる．図 4.3.1 のようにユーザのレベルによって開発環境を提供することによってあらゆるユーザがアプリケーション開発を行うことが可能となる．また，ユーザの成長を促すうえで，開発環境の見た目が変化するとユーザに抵抗感を与えると考え，開発環境の GUI のレイアウトは同じにしている．

4.3.1 一般ユーザに向けた開発

一般ユーザは，全くマッシュアップが行えないレベルのユーザと定義する．このユーザでもマッシュアップを利用するために，図 4.3.2 のような Example という開発環境を実装した．このレベルのユーザに対しては，図の左側にあるメニューから使用したいアプリケーションを選択することでアプリケーションをダウンロード可能な仕組みを提供する．図の左側にはメニューがあり，Example 内でユーザが使用したいアプリケーションをクリックすることで，呼び出すことができる．クリックしたアプリケーションは，右側のワークスペースに使用するモジュールが接続された形で表示される．図の右上の Download ボタンをクリックすることで，ユーザはアプリケーションをダウンロードすることができる．ユーザはアプリケーションの名前を決め，Download ボタンをクリックすることでアプリケーションを開発できる．一般ユーザにとってマッシュアップによるアプリケーション開発は敷居が高いため，Example では細かいモジュール接続の変更等は行えない．しかし，Example ページでは IDUMO の他の開発環境で行うようなモジュールの接続の完成図を確認することができる．

これにより，一般ユーザは IDUMO のモジュールの接続の様子を知ることができ，次のカスタマイズユーザへの成長が期待できる．

4.3.2 カスタマイズユーザに向けた開発

一般ユーザより，ある程度マッシュアップに慣れたユーザが実際にマッシュアップアプリケーションのモジュールの変更等の操作を行える環境として，カスタマイズユーザには図 4.3.3 のような

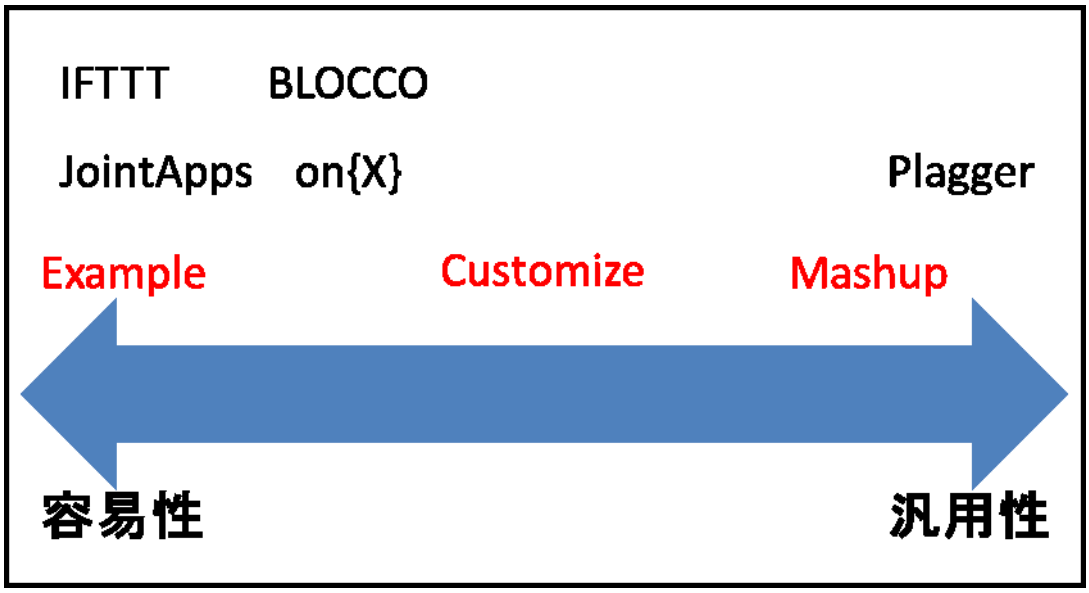

図 4.3.1: IDUMO の開発環境の分別

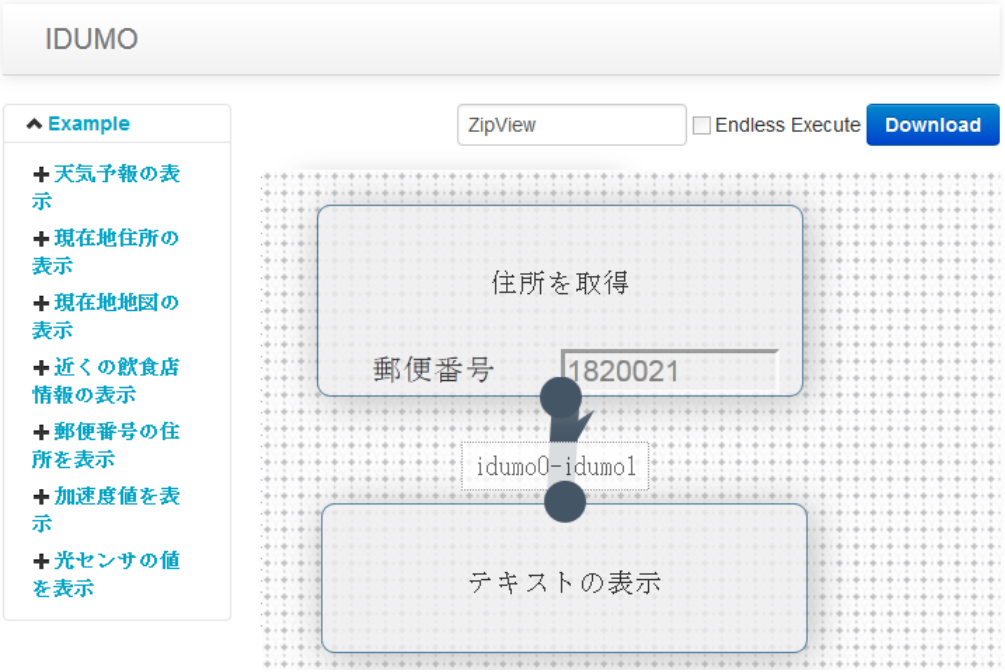

図 4.3.2: 開発環境 Example

Customize という開発環境を実装した．Example との違いとしては，メニューの種類が「Customize」,「History」,「Provider」,「Handler」,「Adaptor」,「Receiptor」と複数あることである．Customize は Example メニューと同じものである．しかし，モジュールの接続の解除や，数値入力を必要とするモジュールに数値を入力することができるようになっている．このレベルのユーザはある程度アプリケーション開発の様子が理解できているものと考える．そこで，Customize メニューからアプリケーションを選択し，表示された設計図をもとにそれを改変する作業を行うことができる．ユーザは，Provider，Handler，Adaptor，Receiptor の中にあるモジュールを選択し，Customize メニューから選んだモジュール接続情報を改変することができる．

例えば，付近の飲食店を地図に表示するアプリケーションをカスタマイズすることを考える．付近の飲食店の情報を地図に表示するものからテキストに表示するという結果に変更する．図 4.3.4 は Customize メニューから付近の飲食店情報を地図に表示するアプリケーションを選択したものである．この実行結果は図 4.3.5 のように実行され，地図上に飲食店の情報が表示されていることがわかる．カスタマイズユーザは,「地図に表示」という結果表示部分を Receiptor メニューの中にある「テキストの表示」に変更する．Receiptor メニュー内の「テキストの表示」をクリックすることで，対応するモジュールが GUI 上に表示される．このモジュールに接続をつなぎかえた結果が，図 4.3.6 のようになる．この実行結果は図 4.3.7 のようになる．実行結果の飲食店情報がテキストとして表示されていることがわかる．このように，カスタマイズユーザは 1 からモジュールの接続を行うことなく，アプリケーションの開発を行うことができる．また Customize では，図 4.3.8 のように History メニューから他のユーザが作成したアプリケーションを選択し，それをもとにカスタマイズすることも可能である．

これにより，モジュールの接続のノウハウを体験することができ，マッシュアップユーザのように 1 からモジュールを接続する開発を行うことが可能となることが期待できる．

4.3.3 マッシュアップユーザに向けた開発支援

マッシュアップユーザは図 4.3.9 のような Mashup という開発環境が提供される．このレベルのユーザは 1 からモジュールを接続することで，自由にアプリケーションを開発することができる．「Provider」,「Handler」,「Adaptor」,「Receiptor」のメニューから作成したいアプリケーションに必要なモジュールを選択し，それを接続していくことでアプリケーションを作成できる．しかし，カスタマイズユーザから成長したばかりのユーザなどは，どのモジュールとどのモジュールが接続可能なのかといったことがわからない可能性がある．そこで選択したモジュールに対して接続されたモジュールが接続可能でない場合に知らせるアラート機能，接続可能なモジュールを推薦する Recommend メニューの実装を実装した．

接続の可否を知らせる機能

IDUMO のモジュールには様々なデータを扱うものがある．しかし，中には明らかに接続することのできない組み合わせが存在する．例えば緯度経度の数値を地図に表示するモジュールに対し，文字列データを受け渡すことはできない．そういったデータのやり取りのできないモジュールを接続したときに図 4.3.10 のようにアラートで知らせる．このとき図 4.3.11 のような動作を行う．IDUMO のサーバ上に全てのモジュールが扱うことのできるデータ形式をデータベースとして保存し，モジュールが接続しようとした際にデータの受け渡し側の Sendable と受け取り側の Receivable

図 4.3.3: 開発環境 Customize

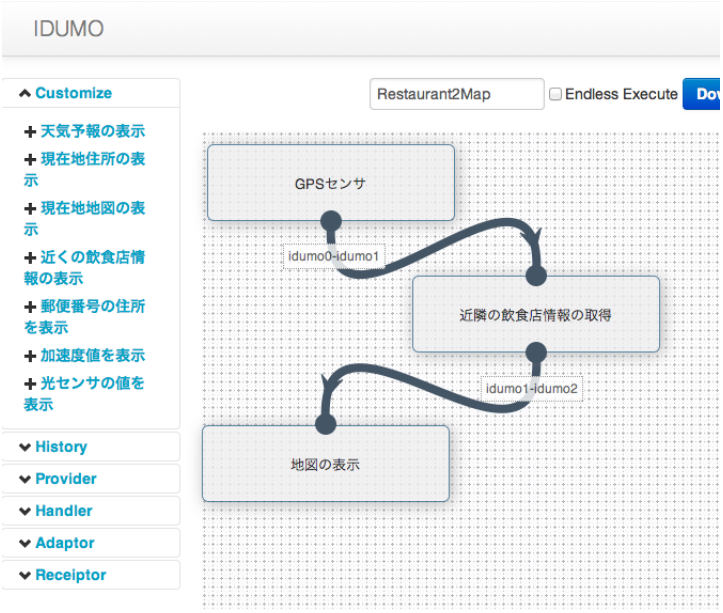

図 4.3.4: 付近の飲食店情報を地図に表示する接続図

図 4.3.5: 付近の飲食店情報を地図に表示した実行結果

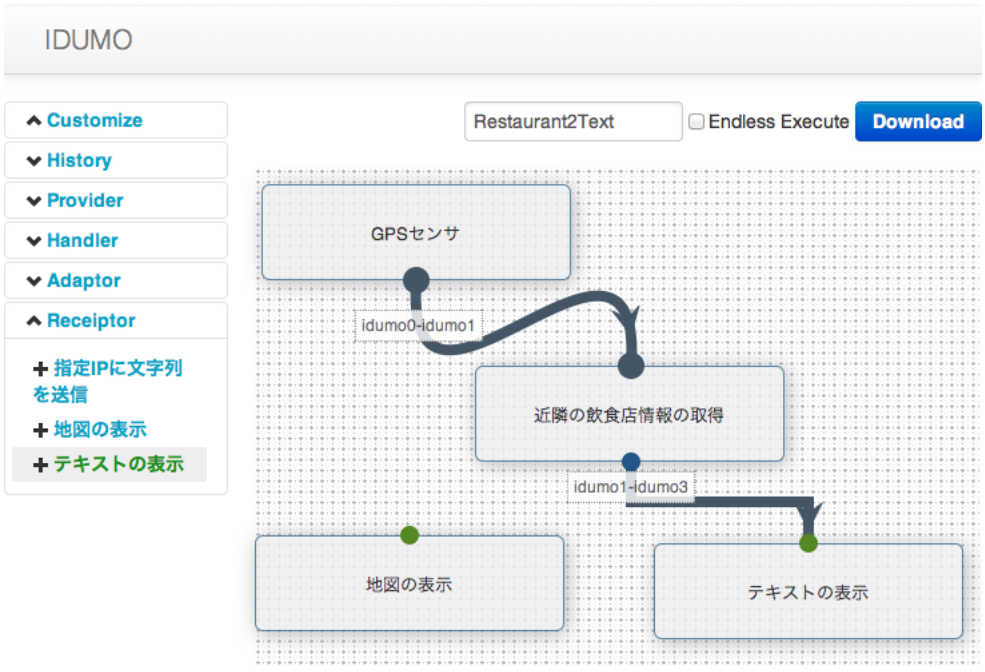

図 4.3.6: カスタマイズユーザによるモジュールの変更

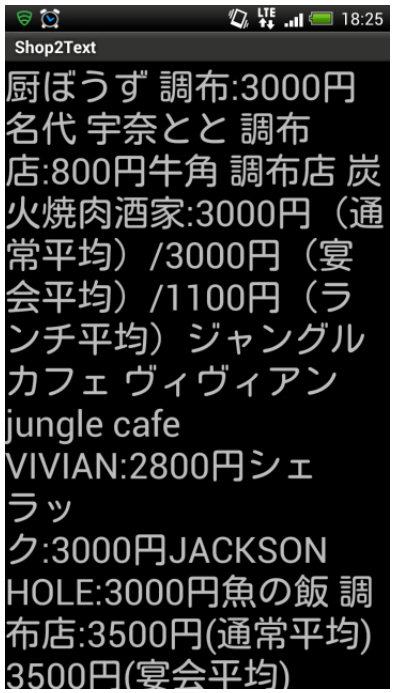

図 4.3.7: 付近の飲食店情報をテキストに表示した実行結果

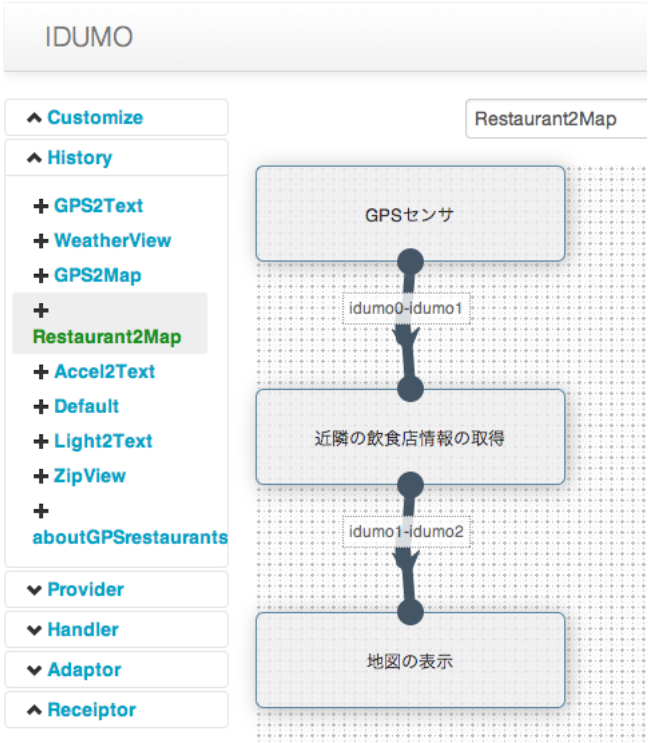

図 4.3.8: History メニューからの開発

のデータ形式のマッチングを行う．データ形式が対応していない場合，アラート画面を表示することで接続できないものであるとユーザに知らせる．これにより，ユーザが動作しないアプリケーションを開発することを防ぐことができる．

Recommend メニュー

ユーザがモジュールを接続する際に，選択するモジュールと接続可能なモジュールを適切に選ぶ必要がある．Recommend メニューでは図 4.3.12 のようにユーザがモジュールを選択した際に，選択されたモジュールの受け渡すデータ形式と受け取ることのできるデータ形式を調べる．そしてそのデータ形式に対応したモジュールを図 4.3.13 のように Recommend メニュー内に表示する．これにより，モジュールの接続に迷った場合でも，接続可能なモジュールを選択できるようになる．

4.4 モジュール開発者に対する補助

IDUMO では，マッシュアップ開発で使用するモジュールをユーザ自身が作成することができる．これにより，新しく追加されたモジュールを利用することで，以前作成できなかったアプリケーションが開発できるといったケースも起こると考えられる．

IDUMO のモジュール開発において，先行研究の段階においては，IDUMO の開発 SDK の中に入っている既存のモジュールを参考にしながらモジュールを開発する必要があった．しかし，モジュール開発者がモジュールを作成する際，モジュールによっては内部処理が似ているものがある．例えば，WebAPI にアクセスし，XML データを受け取るモジュールの場合，リクエストする URL を定義する部分，受け取ったデータをパースする処理，処理されたデータにデータ構造を持たせる処理などはほとんど同じ処理を行っている．また，データを受け渡すインタフェースや受け取るインタフェースの定義や，モジュールの名前や概要等の情報を記述する部分に関してはすべてのモジュールで共通である．このように類似する処理をモジュール開発を行う度に記述するのでは，モジュール開発の負担が大きい．

そこでテンプレートモジュールを作成することでモジュール開発者の負担を減らすことができた．テンプレートモジュール生成プログラムを実行することで，IDUMO 特有の処理を行う部分やモジュールの情報を記述する部分などの共通するコードを自動生成する．これにより，モジュール開発ユーザはテンプレートモジュールで記述された IDUMO 特有の処理を行う部分に作成するモジュールの情報を書き加えるだけにとどまり，ほとんどモジュール特有の処理を記述するのみでモジュールを作成することができる．

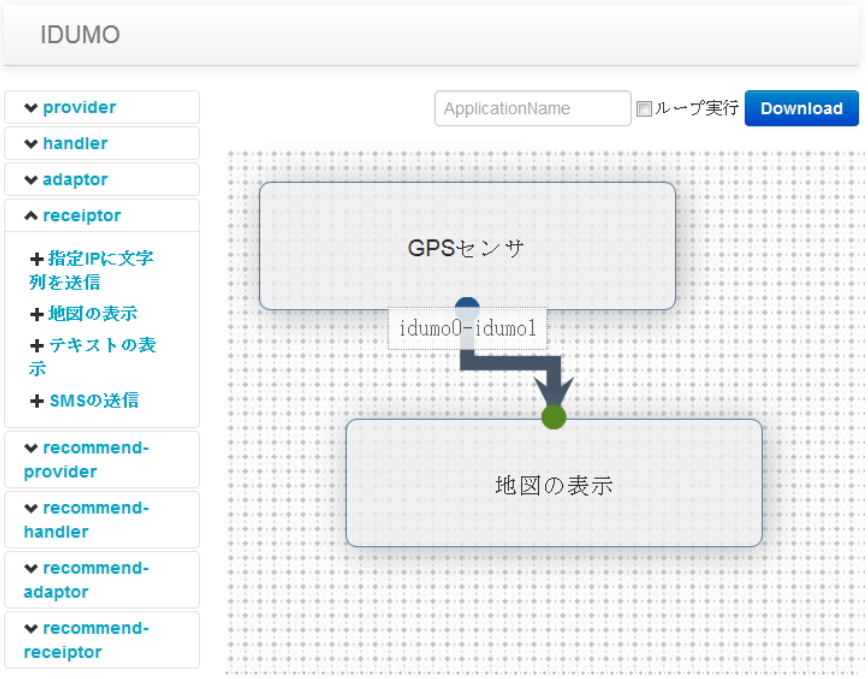

図 4.3.9: 開発環境 Mashup

図 4.3.10: 接続の可否を知らせる機能

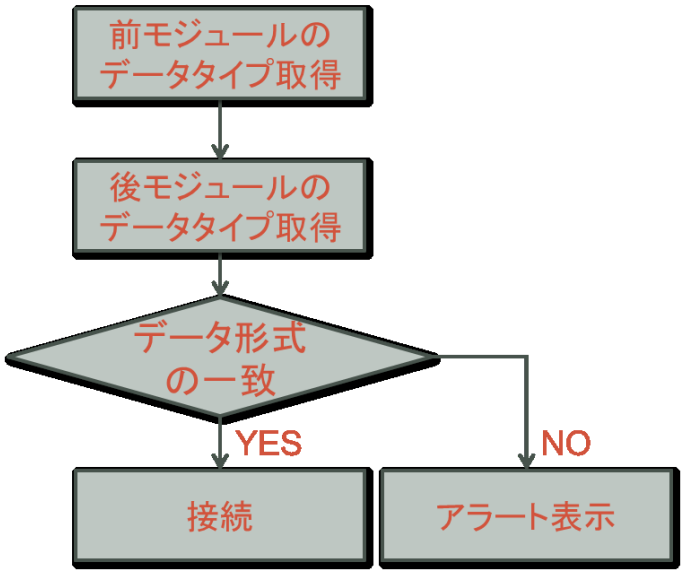

図 4.3.11: 接続の可否を知らせる機能のフローチャート

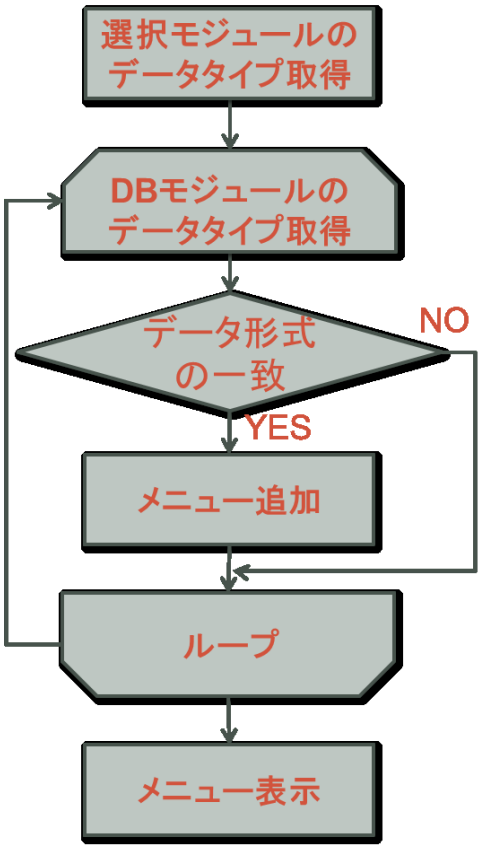

図 4.3.12: Recommend メニューのフローチャート

図 4.3.13: Recommend メニュー

第5章　IDUMOの評価・比較

5.1　ユーザによる評価

5.1.1　IDUMO 全体の評価

改善した IDUMO を実際にユーザに使用してもらいアンケートを実施した．ユーザのアンケートの結果は図 5.1.1 から 5.1.5 のようになった．今回，アンケートを実施したユーザのプログラミングレベルは次の表 5.1.1 のようになっている．

表 5.1: ユーザのプログラミングレベル

	情報系の学生 or 仕事	数回行ったことがある	プログラミング経験なし
人数 (人)	9	3	5

図 5.1.1 より，3 分の 2 のユーザがモジュールの接続のしやすさについてとても使いやすかった，まあまあ使いやすかったと答えている．少し使いづらかったと答えたユーザについては，理由を聞いたところ「使えない組み合わせがあった」と答えており，接続可能だと思ってつないだモジュールでアラートが表示されたためにスムーズに接続できなかったことが原因であった．図 5.1.2 より，ほとんどのユーザがモジュールを問題なく接続することができたと答えている．残りのユーザも Recommend メニューを用いることで問題なく接続できたと答えている．ユーザの中には「Recommend メニューがないと何を使っていいのかわからなかったので，メニューがなければ mashup は使えなかった」と答えている．このことから，ユーザ補助として Recommend メニューから接続可能なモジュールを推薦する手法は有効であるといえる．

図 5.1.3 より，想定通りのアプリケーションを作成できたかという問いにほとんどのユーザが想定通りもしくはほぼ目的を達成できるアプリができたと回答している．思ったような動作をしなかったというユーザはモジュールの接続的には地図に表示するというアプリケーションを作成し，接続情報通りの動作をしていたものの，マップに表示した後にルート案内等を行うことを期待していたために，想定と異なるアプリケーションが出来上がった結果となった．

図 5.1.4 より，GUI の使用感についてほとんどのユーザが使いやすいと応えており，少し使いづらかったと答えたユーザは「メニューが英語や専門語で書かれたいたために，プログラミングをやったことのないものとしては意味が分からなくてやりにくい部分がありました」と答えている．エンドユーザがスムーズに利用することを考えたときに，日本語で書かれたメニューを用いるといったアプローチを行うべきだと感じた．

図 5.1.5 より IDUMO を 5 段階で評価してもらったところ過半数のユーザがとても良い，良いという評価をし，残りは普通，1 人あまり良くないという評価となった．エンドユーザにとって英語でメニューが表示されていたり，モジュールや GUI といった単語を交えた説明は難しさやとっつきにくい印象を与えてしまうということがわかった．

図 5.1.6 より，ほとんどのユーザがまた IDUMO を利用しようと思うと答えている．あまり利用しようと思わないと答えたユーザは「やはり難しい」,「アプリをうまく構成できる自信がない」ということを理由として挙げていた．

IDUMO を使用しての感想としてはプログラミングレスでアプリケーションが開発できるという部分に非常に興味を持っているユーザが多かった.「プログラミングすることに対する興味は以前から持っていたので，入り口のハードルを下げてくれたと思う」という感想を述べたユーザもいたことからマッシュアップという技術をエンドユーザがあまり知らないため利用者は少ないが，このようなシステムに対する需要はあると考えられる．

これらのアンケートより考察を行う．モジュールの接続もスムーズに行うことができたユーザが多く，それ以外のユーザも Recommend メニューを用いてスムーズに接続が行えた．これは Recommend メニュー，接続の可否を知らせるアラートが有用なものであることを示している．GUI の使用感については，英語のメニューがわかりづらいというエンドユーザの意見があった．多くのユーザは英語で書かれているというだけで，抵抗があるということがわかった．エンドユーザの開発容易性を高めるためには，メニュー等も日本語で表示することが必要条件であるといえる．5 段階での評価では，半数以上のユーザからとても良い，良いという評価を得られている．また，IDUMO をまた利用しようと思うというユーザも一定数いることから，GUI 上でモジュールを接続する形式でのアプリケーション開発はユーザ評価により有用であることを示している．

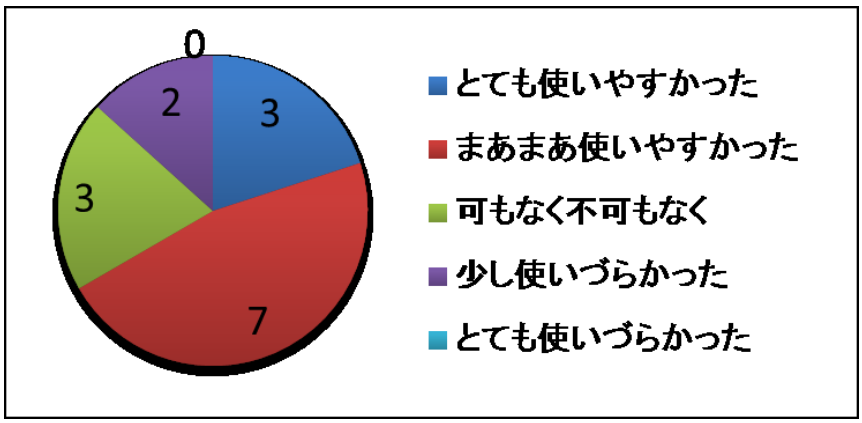

図 5.1.1: モジュールの接続のしやすさについて

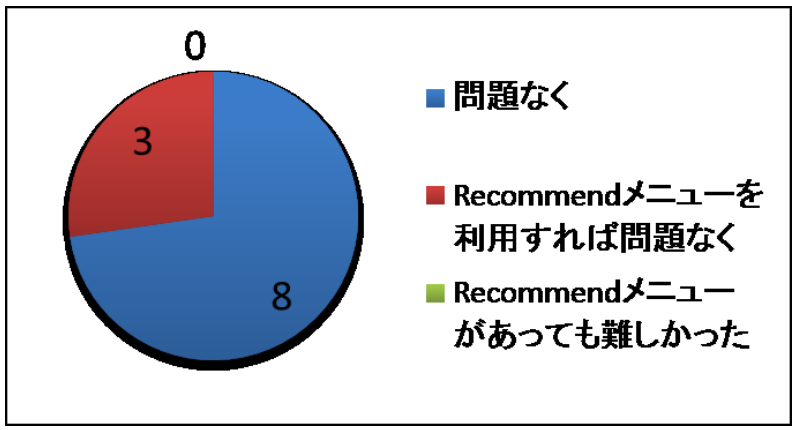

図 5.1.2: 適切なモジュール接続が行えたか

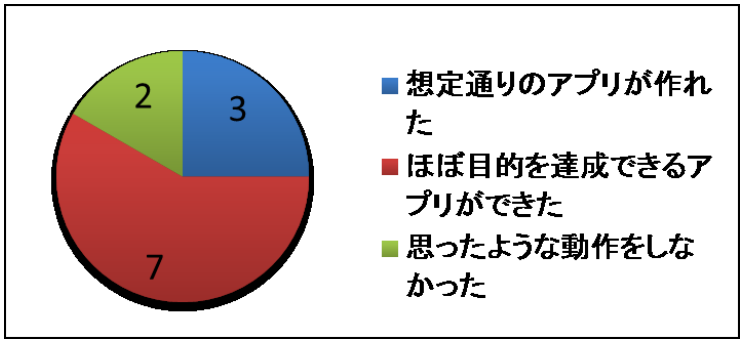

図 5.1.3: 想定通りのアプリが作成できたか

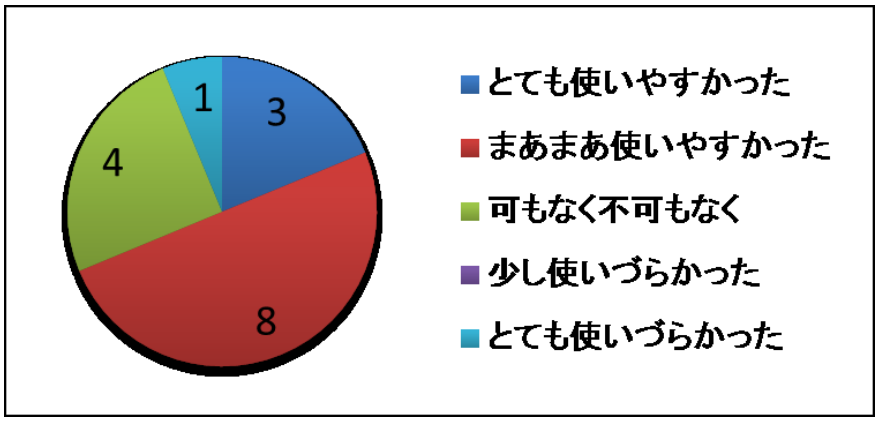

図 5.1.4: GUI の使用感について

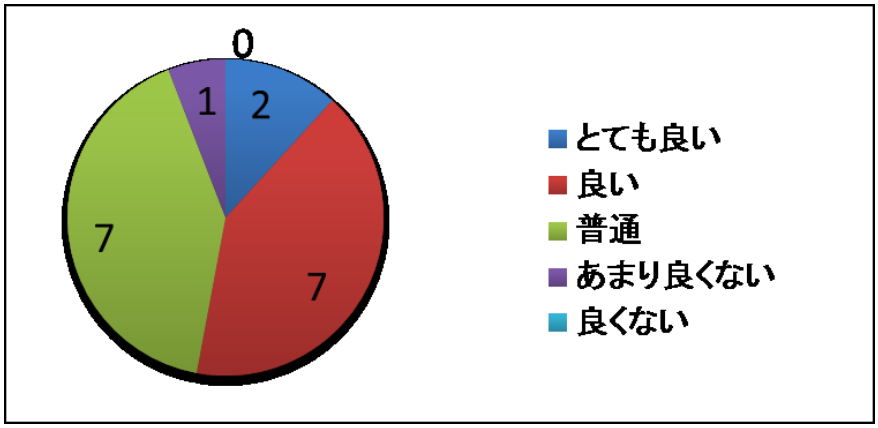

図 5.1.5: IDUMO を 5 段階で評価してください

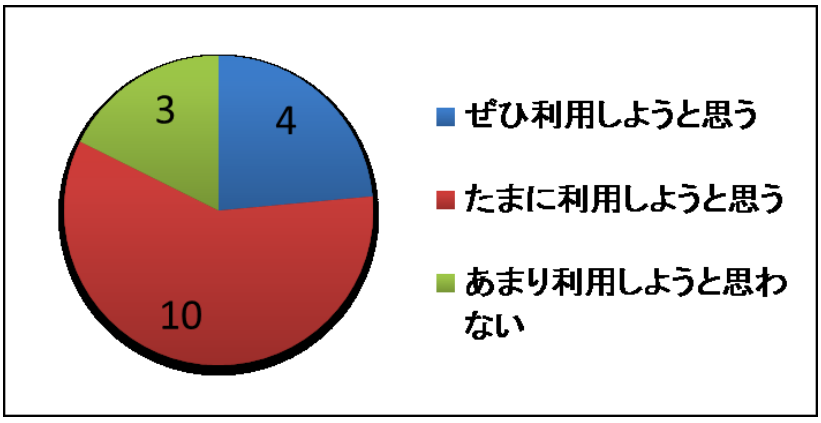

図 5.1.6: また IDUMO を利用しようと思うか

5.1.2 Customizeに関する評価

次にCustomizeを用いることで，エンドユーザがマッシュアップが用いる開発環境Mashupを利用できるユーザへと成長可能かということに関するアンケート結果を図5.1.7から5.1.10に示す．今回アンケートに答えたユーザはマッシュアップツールを使用したことのないユーザであり，大学時の専攻学科は次の表5.1.2の通りである．

表5.2: ユーザの大学時の専攻学科

	情報系学科	その他の理系学科	文系学科
人数(人)	3	4	2

図5.1.7より，Customizeをほとんどのユーザが使いやすいもしくは可もなく不可もなくと応えている．少し使いにくいと答えたユーザは「Mashupの方がRecommendメニューがある分使いやすかった」と答えている．

図5.1.8より，1人を除いたユーザがCustomizeがMashupを使うに当たって適切であると答えている．あまり感じないと答えたユーザは,「CustomizeからMashupがどのようにつながっているのかわからない．ゲーム感覚だったら面白そう」と答えている．

図5.1.9より，開発環境のレベル分けは適切であるとほとんどのユーザが答えている．あまり適切ではないと答えたユーザからは「何も知らないユーザにとってはまだわかりにくい」という意見を頂いた．

図5.1.10より，ほとんどのユーザがCustomizeを利用後にMashupを利用したいと答えている．Customizeによりモジュールの接続のノウハウがわかり，実際に1からの接続にも興味を持っているユーザが多いことがわかる．Customizeを通してユーザの成長が促せ，開発環境のレベル分けが有用であることが確認できる．

ユーザの使用した感想としては「文系人間には敷居が高かった」,「疎い分野なので，難しく感じた」という声もあったが,「モジュール同士の接続は情報の流れが分かりやすくてよい」という評価もあり，IDUMOの開発スタイルでアプリケーションを作りやすいと感じているユーザもいた．

Customizeに関するアンケート結果をまとめる．ほとんどのユーザがCustomizeを使いやすいと答えている．可もなく不可もなくと答えているユーザや，少し使いにくいというユーザもいた．Customizeには，History等他のメニューもあるため，Exampleから来たばかりのユーザにはとっつきにくさがあるのかもしれない．また，Mashupにある Recommendメニューがないため，使いにくいという意見もあった．CustomizeからMashupへの成長を促すにあたって適切であると感じたユーザが大半であった．あまり感じないというユーザは，IDUMOのレベル別開発環境の全体像が伝わっていなかったと考えられる．ドキュメント等で図を用いた説明をしっかりと提示する必要性があるといえる．しかし，多くのユーザがCustomizeを利用後にMashupを利用したいと答えているため，CustomizeからMashupへの成長を適切に促せていることがユーザ評価より示せている．したがって，ユーザのレベル別開発環境を提示するという本研究の手法は有効であるといえる．

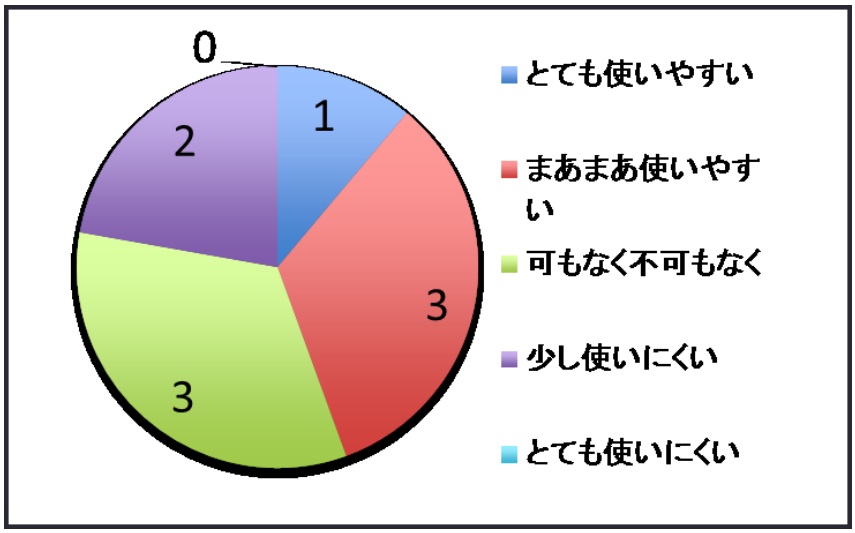

図 5.1.7: Customize の使いやすさについて

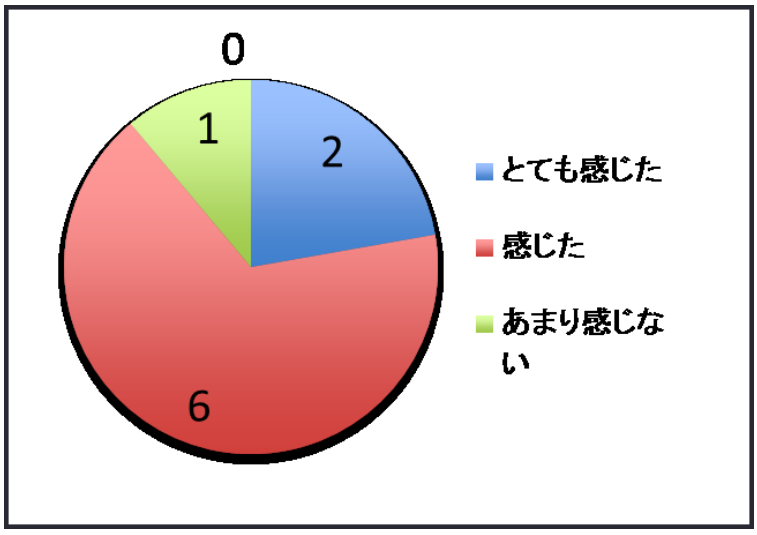

図 5.1.8: Customize は Mashup を使うにあたって適切かどうか

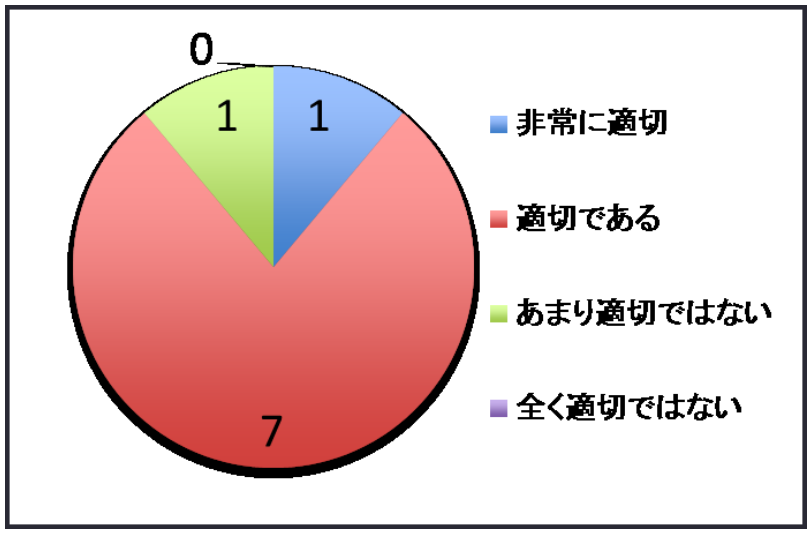

図 5.1.9: 開発環境のレベル分けは適切だと感じたか

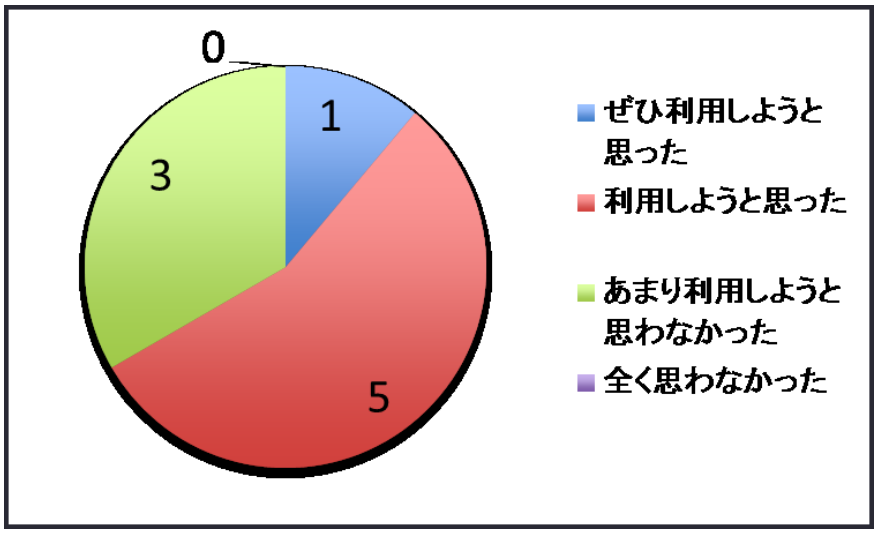

図 5.1.10: Customize を利用した後に Mashup を利用したいか

5.1.3 ユーザの成長に関する評価

ユーザが IDUMO を通してマッシュアップユーザへと成長できるか確認した．実験の手順は次のとおりである．

1. Example,Customize の説明を行う
2. Example にて“ 住所を検索するアプリケーション ”を作成および Customize にて“ 付近の飲食店情報を地図に表示するアプリケーション ”の出力をテキストに変更してもらう
3. Mashup にて“ 付近の飲食店情報を地図に表示するアプリケーション ”を再現してもらう
4. 課題アプリケーションとして“ アニメ情報の表示 ”,“ BAR 情報を平均価格でソートして表示 ”というアプリケーションを作成してもらう

それぞれ実験で作成してもらったアプリケーションの作成例のうち,“ 住所を検索するアプリケーション ”は図 4.3.3,“ 付近の飲食店情報を地図に表示するアプリケーション ”は図 4.3.4，テキスト表示に付け替えたものが図 4.3.6 となる．また,“ アニメ情報の表示 ”,“ BAR 情報を平均価格でソートして表示 ”というアプリケーションの作成例は図 5.1.11，5.1.12 のようになる．評価方法は，アプリケーション作成に使用するモジュール数と接続数をそれぞれ 1 手としてカウントする．例えば,“ BAR 情報を平均価格でソートして表示 ”を作成する場合，必要なモジュール数が 4 つ，接続数が 3 つである．このアプリケーションを詰まることなく作成した場合，モジュールを呼び出し，接続する手順を合計して 7 手となる．ユーザが作成に手間取ったり，異なったモジュールを呼び出したりした場合，手順が増加する．この手順が最短のものよりどのくらい離れるかでスムーズにモジュールの接続が行えるかを確認する．

この実験を通してマッシュアップを利用したことのないユーザでも Example，Customize といった開発環境を用いてチュートリアルを行うことで，開発環境 Mashup 上で 1 からアプリケーション開発を行うことができるようになるかを確認する．

今回実験に協力してくれた被験者は 4 人であり，それぞれマッシュアップツールの使用経験はないユーザである．実験の結果は次の表 5.1.3 のようになる．

表 5.3: ユーザの成長に関する実験結果

	再現	アニメ情報の表示	BAR を平均価格でソートして表示
被験者 A	5	3	7
被験者 B	5	3	9
被験者 C	5	3	7
被験者 D	5	3	8

表の結果より，被験者全員が“ 付近の飲食店情報を地図に表示する ”アプリケーションの再現に最短手順で成功していることがわかる．これは Customize にて作成例を確認し，モジュールの付け替え作業を実際に行ったためだと考えられる．次に“ アニメ情報を表示 ”アプリケーションも全てのユーザが最短手順 3 手で作成できている．これは使用するモジュールが少なく，接続が 1 つのため，とても簡単に作成できるためだと考えられる．そして“ BAR を平均価格でソートして表示 ”アプリケーションは 2 人の被験者が最短手順である 7 手で作成でき，被験者 B が 9 手，被験者 D が 8 手かかった．被験者 B は出力方法を“ 地図に表示 ”モジュールを選択し，接続してし

まったため 2 手余分にかかった．被験者 D は出力方法を“ 地図の表示 ”モジュールを選択したが，接続する前に気付いたため 1 手多くかかる結果となったが，スムーズにアプリケーション開発が行えたと言える．

これは，IDUMO のモジュール名が“ GPS を取得する ”，“ 地図に表示する ”といった形の目的ベースで表示されているため，マッシュアップ未経験の被験者でも，作りたいアプリケーションの処理の流れがイメージしやすいことが挙げられる．

この結果より，Example や Customize にてモジュールの接続の様子を確認することでマッシュアップ開発の様子を知り，モジュールの付け替えといった操作にて実際のマッシュアップ開発を体験することができることがわかる．それによってマッシュアップの経験のないユーザであっても，開発環境 Mashup にてアプリケーションの作成をスムーズに行うことができるようになった．

5.2 既存マッシュアップツールとの比較

既存のマッシュアップツールと IDUMO を比較することで IDUMO の有用性を示す．比較対象としたマッシュアップは JointApps，IFTTT，BLOCCO，on{X}，Plagger である．

5.2.1 JointApps との比較

JointApps はボタンのクリックや時間経過による画面遷移を行う Android アプリケーションの開発ができるツールである．表示される画面ごとに背景画像やボタンなどのレイアウトの設定やボタンを押したときの動作を設定することでアプリケーションを作成できる．また，他者が作成したアプリケーションをダウンロードして使用することもできる．IDUMO では背景画面の設定やボタンのレイアウトといった設定は行うことができないため，JointApps のようなアプリケーションの作成は行えないが，JointApps ではボタンの入力等の動作を起点として動作を行うアプリケーションなので，IDUMO で作成できるイベントドリブンなアプリケーションや自動実行して結果を表示するタイプのアプリケーションの作成は行えない．また，開発において画像の配置設定など細かく指定できる反面，操作が複雑なため，エンドユーザの敷居が少し高いといえる．ユーザのレベルに応じた開発ツールを提供している IDUMO の方が様々なレベルのユーザの開発を促せるといえる．

5.2.2 IFTTT との比較

IFTTT は AndroidOS や iOS 上で動作するイベントドリブン型のアプリケーションを開発できる．「〜なら…を行う」という形式のアプリケーションを Facebook や Evernote といった既存のアプリケーションを組み合わせて簡単に開発することができる．エンドユーザでも発生の条件と行われる動作を既存のアプリケーションから選択するだけで開発できるため，とても利用しやすいが，イベントドリブンという制約により，ユーザの実現に対して制約が発生する場合がある．また，既存のアプリケーション同士の組み合わせを前提としているため，使用できるアプリケーションを新しく追加することができない点が IDUMO とは異なる．

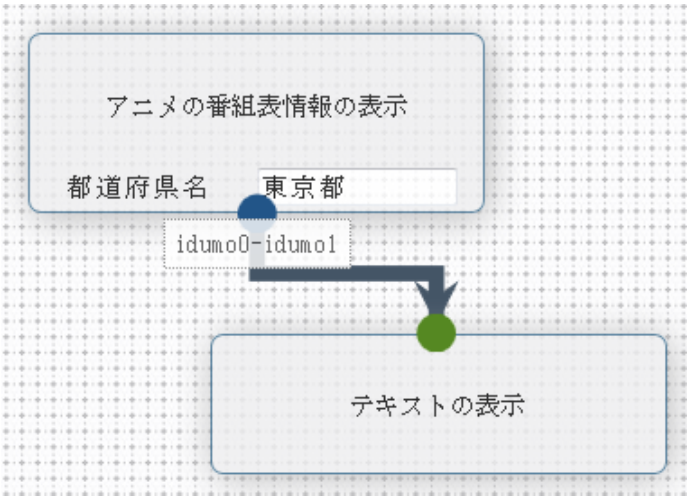

図 5.1.11: アニメ情報の表示のアプリケーション例

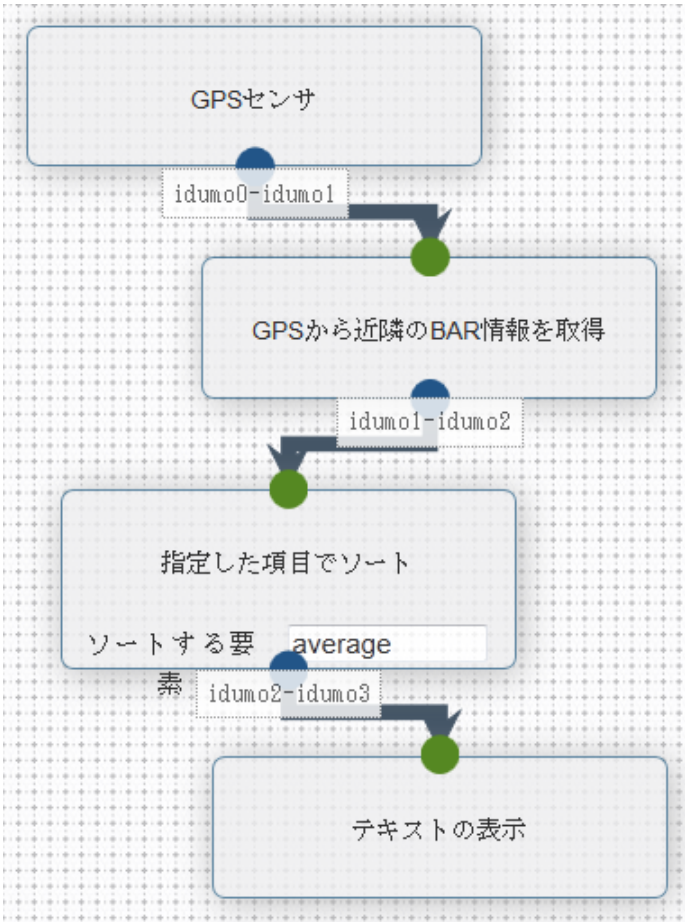

図 5.1.12: BAR 情報を平均価格でソートして表示

5.2.3 BLOCCO との比較

BLOCCO は AndroidOS 上で動作するイベントドリブン型のアプリケーションを作成できる.「〜なら…を行う」という形式の条件に当てはめて動作の設定を行うことができる．開発者は公開されている BLOCCOSDK を用いることで，条件やサービス等のモジュールをユーザが開発することができる．やはりこれもイベントドリブンという制約によって，ユーザの実現に対して制約が発生する．

5.2.4 on{X} との比較

on{X} では AndroidOS 上で動作するアプリケーションを作成できる．エンドユーザは用意されているレシピの数値やテキストを変更することでアプリケーションを利用することができる．しかしアプリケーションのレシピを作成したり，レシピの一部の動作を変更する場合，Javascript によってプログラミングを行う必要があり，エンドユーザが利用できるサービスには限界がある．IDUMO ではエンドユーザでもモジュールの接続を行って 1 からアプリケーションを開発することも，既存のものを一部分カスタマイズすることも可能である．

5.2.5 Plagger との比較

Plagger は Perl モジュールのマッシュアップにより，アプリケーションを開発することができる．モジュールを組み合わせることで様々なアプリケーションを開発することができる．使用されるモジュールも Perl を用いて作成することができるため，Perl を扱えるユーザは自由にモジュールを作成し，柔軟なアプリケーション開発が行える．IDUMO と非常によく似たアプリケーションであるが，Perl を扱うことができるユーザにしか導入，設定等を行うことができない．よってエンドユーザにとっては敷居が非常に高いものとなっている．

5.3 考察

アンケート結果より，多くのユーザがマッシュアップフレームワーク IDUMO について使いやすいという評価をしており，マッシュアップという技術に興味を持ってくれていることがわかる．モジュールの接続に関しても Recommend メニューがなければ接続ができなかったというユーザもおり，ユーザがモジュールを接続する際にどのモジュールと接続可能なのかということを迷った際に適切にユーザを補助することができていることがわかる．既存のマッシュアップツールでは，開発できる内容を制限されていたエンドユーザも，IDUMO のレベル別の開発環境を順番に体験することによって一般ユーザからカスタマイズユーザ，マッシュアップユーザと成長することが可能であることがわかる．

一方で Customize から Mashup へ成長するために必要だと考えられる要素をユーザに聞いてみたところ,「慣れ」,「マッシュアップへの興味」という声が多く，エンドユーザがマッシュアップを用いてアプリケーションを開発するという行為に興味があるがアプリケーションを開発するという行為そのものにある種の抵抗を持っていることがうかがえる．また，興味があるユーザは色々と思考を凝らしたくさんの質問をしてくれたが，あまり興味のなさそうなユーザもおり温度差が

あったため，マッシュアップを用いたアプリケーション開発ということに興味を持ってもらうためのアプローチ方法を検討するという新しい課題もあることがわかる．

そして，実際に被験者にアプリケーション開発がスムーズに行えるかという実験をしてみたところ多くの被験者が最短手順でアプリケーション作成を行うことができ，残りの被験者も 1，2 度の間違いでアプリケーション作成が行えた．モジュール名が目的ベースで表示されており，モジュール名から作成したいアプリケーションをイメージし，スムーズに開発が行えたことからもユーザのレベル別に開発環境を提供する試みは有効であることがわかる．

既存のマッシュアップツールとの比較に関しては，多くのマッシュアップツールは商用的にエンドユーザに利用してもらう必要があるため，開発容易性を重視している．IFTTT や BLOCCO，on{X} などの開発できるアプリケーションに制限があるが，簡単にアプリケーションを開発できる．エンドユーザが成長し，より複雑な開発を行うということは想定していないため，現状の開発可能な範囲で満足できないユーザには不満が残る．一方で Plagger のようにプログラミングを扱えるユーザを対象としている場合は，非常に柔軟な開発を行うことができる．エンドユーザにとって敷居が高いが，プログラミングができるユーザとしてはこちらの方が自らの要望を満たすことができる．IDUMO では，マッシュアップを通してより複雑なアプリケーションを作成したいが，プログラミング知識がないため次のステップへ進めないといったユーザでも複雑なアプリケーション開発をプログラミングレスで簡単に行うことができるツールを目標にしているため，商用的にユーザを獲得しようとしているマッシュアップツールや，使える人だけが便利なアプリケーションを作成できることを目的としたマッシュアップツールとはまた違う目線で開発を行っており，新規性があるといえる．そしてユーザによるアンケートの結果からその試みは有用であることもわかった．

第6章　結論

本研究では，既存のマッシュアップツールにおける問題点，先行研究におけるマッシュアップツール IDUMO における課題をもとに改善手法の提案と実装，ユーザによる評価を行った．既存のマッシュアップツールにおいては，エンドユーザへの開発容易性の追求のために汎用性が失われたツールと，開発の自由度を高めるために開発容易性の低いツールが存在し，中間を担うマッシュアップツールが存在しなかった．IDUMO では，開発容易性を損なわずに柔軟なアプリケーション開発を行えるマッシュアップツールを目標とし，ユーザのレベル別に開発環境を実装した．また，マッシュアップの使いやすさにおいて，ユーザ補助として接続可能なモジュールを推薦する Recommend メニュー，接続できないモジュールを知らせるアラートの実装，モジュール開発を行う際の共通の記述部分を省略できるテンプレートの作成等を行った．これにより，エンドユーザであっても自らのレベルにあった開発環境でアプリケーションを開発し，さらに次のレベルへと成長することも可能である．さらに，ユーザに実際に使用してもらっての評価，そしてアプリケーション作成をスムーズに行えるかを実験を通して，それらの有用性を確認することができた．

IDUMO は次の URL で公開されている [6]．公開したことで，ユーザから作成してほしいモジュールや，挙動のおかしいモジュールの報告があったりと，自分だけではわかりえなかった問題点，改善点が見つかったため，非常に有意義なものとなった．

電気用品安全法の改正に伴い，スマート家電の外出先からの遠隔操作が可能となったため，今後スマートフォンを用いた家電製品の遠隔操作によってさらに扱えるリソースが増加していくと考えられる．ユーザによって日常的に行う行動パターンは様々であり，家電製品との連携を行えるようになっても全てのユーザの要望を満たすアプリケーションというのは難しい．マッシュアップにより，家電製品の操作をモジュール化し，「自宅の最寄駅に着いたら，エアコンの電源を入れる」といったアプリケーションや，「天気予報を取得し，雨だった場合洗濯機を回さない」といったアプリケーションをエンドユーザ自らの手で作成するためのマッシュアップツールのデファクトスタンダードとなれることを目指す．

今後の展望として，ユーザ評価によりわかった専門用語を多用しすぎない開発環境や，マッシュアップという技術に興味を持っていないユーザに対するアプローチを検討していきたいと考えている．

謝辞

本研究を進めるにあたり，ご指導，ご助言をいただきました，吉永努教授に感謝の意を表します．大学院から情報系の学科に進学し，知識の乏しい私を，導いていただきありがとうございます．また，同講座入江英嗣准教授，吉見真聡助教授にも研究を進めるに当たり多くの助言をいただき，研究がより良いものとなりました．ありがとうございます．

先行研究を行い，卒業後も研究に関する助言をしていただいた放地宏佳氏，二年間日々の生活の中で議論を交わし，ともに切磋琢磨した同期の大木裕太，小野澤清人，黒田修平，佐久間大輝，佐藤遼，中村勇勝，藤原大輔，一年間ではありますが，ともに研究生活を過ごした先輩方，後輩達の皆様のおかげで充実した研究生活を送ることができました．本当にありがとうございました．

参考文献

[1] 萩野浩明，藤井邦浩，村上純子，原未來．ユーザ設定によりサービスの組み合わせが可能なサービス連携システム「BLOCCO」．NTT DOCOMO テクニカル・ジャーナル，Vol.18，No.4，Jan 2011．http://www.blocco.jp/.

[2] on{X}．https://www.onx.ms/.

[3] 放地宏佳，三好健文，入江英嗣，吉永努．ネットワークコンピューティングのための包括的マッシュアップフレームワークの検討．情報処理学会研究報告．UBI，[ユビキタスコンピューティングシステム]，Vol.2011，No.11，pp.1-8，nov 2011．

[4] 放地宏佳，三好健文，入江英嗣，吉永努．ネットワークコンピューティングのための包括的マッシュアップフレームワーク IDUMO の設計．マルチメディア，分散，協調とモバイル（DICOMO2012）シンポジウム，pp.1573-1582，jul 2012．

[5] 放地宏佳．IDUMO：ネットワークコンピューティングのための包括的マッシュアップフレームワーク．平成 24 年度修士論文．

[6] IDUMO.
http://idumo.comp.is.uec.ac.jp/builder/index.html.

[7] NetVives．http://www.netvibes.com/ja-jp.

[8] MyYahoo!．http://my.yahoo.co.jp/.

[9] Pipes：Rewite the web．http://pipes.yahoo.com/pipes/.

[10] Dapper:The Data Mapper．http://open.dapper.net/.

[11] JointApps．http://www.jointapps.net/.

[12] IFTTT．https://ifttt.com/.

[13] Plagger．http://plagger.org/.

[14] 森雅生，中藤哲也，廣川佐千男．マッシュアップ・リソースとマッシュアップ・グルー．Web-DBForum 2008，2A-1.

[15] 横山昌平，的野晃整，サイドミルザパレビ，小島功．Web2.0 における jsvascript コードのモジュール化とマッシュアップの枠組み．日本データベース学会 Letters，Vol.5，No.3，pp.1-4，Dec 2006.

[16] 下條彰，福田将之，井垣宏，中村匡秀．異なるライフログをマッシュアップするためのデータ変換・集約アクセス api の実装（不均質なライフログからのデータ間イニングおよび一般）．電子情報通信学会技術研究報告．LOIS，ライフインテリジェンスとオフィス情報システム，Vol.109，No.450，ｐｐ.85-90，feb 2010．

[17] 森雅生，中藤哲也，廣川佐千男．マッシュアップの軽量実装のための提案．In Proceedings of Data Engineering Workshop 2007．IEICE．C7-152，2007．

[18] Marwan Sabbouth，Jeff Higginson，Salim Semy，and Danny Gagne．Web mashup scripting language．In Proceedings of the 16th international conference on World Wide Web，WWW'07，pp.1305-1306，New York，NY，USA，2007．ACM．

[19] E．Michael Maximilien，Ajith Ranabahu，and Stefan Tai．Swashup : situational web applications mashups．In Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion，OOPSLA'07，ｐｐ.797-798，New York，NY，USA，2007．ACM．

[20] 幸城祐樹，三村次朗，上田賀一．アスペクト指向を用いたマッシュアップ構築支援システムの開発 (web サービス・システム連携 (学生セッション))．情報処理学会．ソフトウェア工学研究会報告，Vol.2008，No.29，ｐｐ．171-177，mar 2008．

[21] 今入庸介，小坂隆浩．モバイルマッシュアップによる柔軟なサービス連携の仕組み（学生特別セッション，モバイル/放送融合技術・システムおよびアプリケーション品質，モバイルコンテンツ，モバイル映像配信，p2p/アドホックネットワーク，一般）．電子情報通信学会技術報告．MoMuC，モバイルマルチメディア通信，Vol.110，No.199，ｐｐ.1-5，sep 2010．