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	放地 宏佳	学籍番号	1152032
論 文 題 目	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		

要　旨

通信基盤と計算性能の発展は遠隔リソースへのアクセスと計算環境の拡張を可能とし，ネットワーク接続可能なデバイスが利便性の高い新しい可能性を提供するようになっている．例えば，モバイルネットワークの発展がリアルタイムな天気予報や鉄道運行情報の提供を実現し，スマートフォンやタブレット端末は，デバイス自身の機能に加え，遠隔リソースと連携したデータ保管や計算処理を行なっている．また，インターネットの双方向性が，個人からの情報発信や，情報家電との連携などを可能とし，ユーザはサービスから画一的に配信される情報を得るだけでなく，主体的にサービスを扱えるようになった．

これらの情報基盤の発展により，サービスを最大限に活用し，ユーザの要望に答えることが，最大公約数的なアプリケーションでは実現できなくなった．その結果，自らの手でサービスを組み合わせてアプリケーションを開発するマッシュアップのアプローチが増えてきている．しかし，既存のマッシュアップ手法では，開発に汎用性を求めると専門知識を必要とし，容易性を求めるとサービスを限定してしまう．

本論文では，汎用性と容易性を兼ね備えたマッシュアップフレームワーク IDUMO を提案する．

- 種々のサービスのマッシュアップ
  - コントロールフローとデータフローの分割により，サービスの利用するプロトコルに依存しない設計
- 簡単かつ柔軟なサービス間連携
  - データフォーマットの定義により，サービス間の自動変換データ変換(簡単)と生データ同士の接続可能(柔軟)な設計
- 非限定的な動作環境
  - 実行モデルの定義により，デバイスの実行モデルとアプリケーションの切り離した設計
- 簡単なマッシュアップ環境
  - ユーザレベルごとの開発手法の定義により，ユーザレベルに依存しない開発環境の設計

上記設計に基づいたマッシュアップフレームワークの実装を行った. IDUMO を用いたアプリケーション開発を通して，汎用性と容易性を兼ね備えたフレームワーク実現を確かめた．

平成 24 年度修士論文

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

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 ： 1152032

氏　　名 ： 放地 宏佳

主任指導教員 ： 吉永 努

指 導 教 員 ： 入江 英嗣

指 導 教 員 ： 加藤 聰彦

提出年月日 ： 平成 25 年 1 月 24 日

(表紙裏)

目 次

図 目 次

(図目次裏)

第1章　序論

Web サービスは，様々な形態のデバイスがネットワーク接続を前提として設計されるほどに，多様化の速度を増しつつある．通信基盤と計算性能の発展は遠隔リソースへのアクセスと計算環境の拡張を可能とし，ネットワーク接続可能なデバイスが利便性の高い新しい可能性を提供するようになっている．例えば，モバイルネットワークの発展がリアルタイムな天気予報や鉄道運行情報の提供 [1][2] を実現し，スマートフォンやタブレット端末は，デバイス自身の機能に加え，遠隔リソースと連携したデータ保管 [3][4] や計算処理 [5][6] を行なっている．また，ソーシャル・メディアサービスに代表されるインターネットの双方向性が，個人からの情報発信 [7][8] や，操作者の位置を問わない情報家電との連携 [9][10] など，ネットワークを介したリソースの利用やサービスの活用方法が提案され，普及が進んでいる．

これらの情報基盤の発展は，アプリケーションのありかたに変化を与え始めている．現在までに一般的に提供されてきたアプリケーションは，最大公約数的な機能の実現が求められていた．しかし今では，個人で扱えるリソースが増えたことで，アプリケーションに対する利用者の望む要求が千差万別となり，実際のアプリケーションとのギャップが生じるようになっている．また，利用者がアプリケーションを動作させるプラットフォームの多様化が，統合的なソフトウェアの提供を困難にしている．

その一方で，新しい形態の Web サービスの利用方法における大きな特徴の 1 つとして， ユーザの情報に対する姿勢が，画一的にデータを受信するだけでなく，主体的にアクセスする形へと変化していることが指摘できる．このユーザの姿勢の変化は，既存の Web サービスやリソースを組み合わせ， 新たなサービスを形作るマッシュアップの概念を生み出し，大きな展開を見せている [11][12].

マッシュアップには，自分の望む情報をまとめて見れるようにするガジェットツールから，ニュースサイトの更新情報を取得可能な RSS リーダなど，様々な種類がある．これらのマッシュアップに共通して見られる特徴は,「利用者ごと自ら望んだことが」,「低い開発コスト」で得られるという点である．個々人の望む要求を自ら解消可能なマッシュアップによるアプローチは今後も盛んに行われることが想像される．

しかしながら，既存の手法では，汎用的なマッシュアップはプログラミングユーザを対象，エンドユーザを対象としている手法は限定的な機能しか持たないといった，エンドユーザがマッシュアップにより容易にアプリケーションを開発できる基盤が提供されていない．そこで我々は，汎用性と容易性を兼ね備えたマッシュアップフレームワーク IDUMO を提案する．IDUMO は，以下に示す機能を有するマッシュアップ環境をユーザへ提供する．

- デバイス上の機能や Web 上の機能といった種々のサービスのマッシュアップ環境
- ユーザの所有する端末上で動作するアプリケーションの提供
- ユーザレベルに合わせたプログラミング環境

本論文では，IDUMO フレームワークを実現するにあたっての課題と実装を述べる．以降，本論文は以下のように構成される．第２章で IDUMO の基盤となるマッシュアップの関連技術を述べ，第３章で IDUMO の設計方針を述べる．第４章でマッシュアップフレームワークの実装を，第５章では開発手法の実装を述べる．第６章では，実際のアプリケーション開発のケーススタディを示し考察を行う．最後に第７章で本論文をまとめる．

第2章　マッシュアップ関連技術

本章では，既存のマッシュアップツールの分類とマッシュアップを用いた研究について述べる．本マッシュアップ手法の位置付け，および，既存のマッシュアップの問題点をまとめる．

2.1　既存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．本節では，これらマッシュアップ手法を以下の３つに分類し，まとめる．

- 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開発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，ユーザへ情報を提示するマッシュアップツールとして，iGoogle[13] や WindowsGadgets[14] がある．これらマッシュアップツールは，自分が望む情報を選択し，それを 1 つの画面に集約する．その結果，ユーザは画面を横断することなく種々のサービスを閲覧可能となる．

このマッシュアップ手法は．他のサービスリソースを自プラットフォーム上に表示するアプローチを取る (図 2.1.1)．これにより，様々なサービスリソースをユーザフレンドリに閲覧することが可能となるが，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[15] や Dapper[16] がある．Yahoo!Pipes は，WebAPI で提供される XML や JSON, Atom で表現されるデータを自らの望むデータに変換するフィードアグリゲータである．Dapper は，Web ページの HTML を解析して自らの望むデータ構造へ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・閲覧しやすく加工するアプローチを取る (図 2.1.2)．これにより，様々なサービスリソースを，利用しやすく加工することが可能となるが，実際のデバイス上で動くようなアプリケーションの開発はできない．

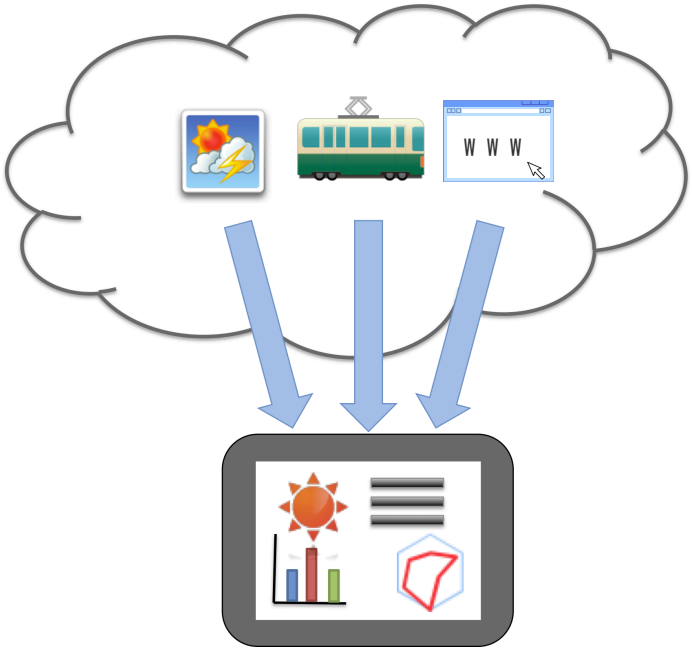

図 2.1.1: データを集約するマッシュアップ

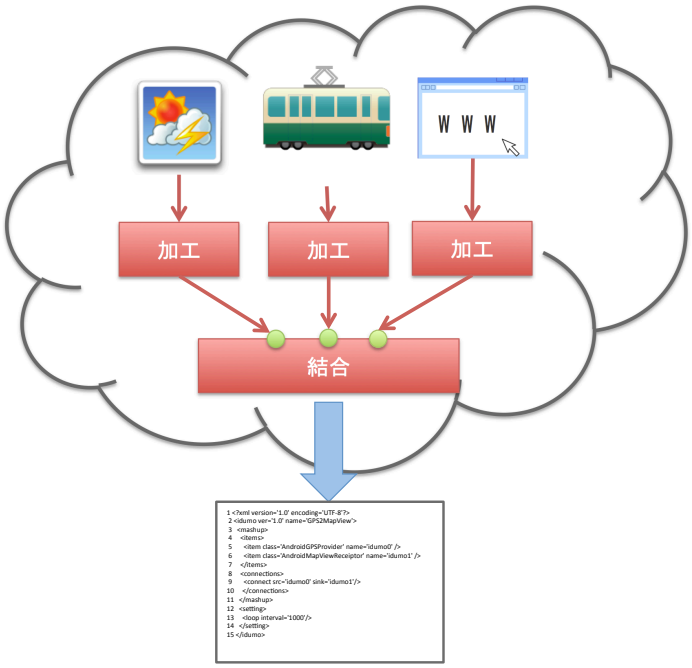

図 2.1.2: データを生成するマッシュアップ

2.1.3 アプリケーションを開発するマッシュアップ

API の機能やデバイスの機能を結合して，アプリケーションを開発するマッシュアップツールとして，BLOCCO[17] や Plagger[18] がある．これらマッシュアップツールは，開発者から提供されたモジュールの接続順序を定義することで，サービス間を横断したアプリケーションを開発可能な環境である．我々が提案する IDUMO もこの分類のマッシュアップ手法を対象としたツールである．

このマッシュアップ手法は，ネットワーク上の機能やデバイスの機能を扱い，実際のデバイス上で動くアプリケーションを開発可能である (図 **??**)．この手法は，開発者が作成し，実際に公開されるようなアプリケーションを作成出来る反面，他のマッシュアップ手法に比べてマッシュアップ環境が複雑である．

2.2 マッシュアップに関する研究

マッシュアップが有用な開発手法であることが注目されている一方，API で提供されるデータフォーマットが統一的ではないことや，マッシュアップアプリケーションの作成には専門知識が必要である点が指摘されており，より簡単なマッシュアップ開発手法が求められている．

2.2.1 サービス間連携の複雑性

マッシュアップによる開発手法は有益である一方，他のサービスとの連携の複雑さが問題となっている．森ら [19] は，マッシュアップによるアプリケーション開発をマッシュアップの対象と，マッシュアップの接続という観点に分割するプログラミングスタイルを提案している．マッシュアップ対象は入力と出力を持ち，それを接続することでサービス連携を行う．横山ら [20] 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手法でも，多くがこの方式を採用しており，内部処理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なっている [15][17][18]，サービスのモジュール化を行い，サービス間の連携をデータのみとする手法は，不特定多数のリソースを扱うのに適している一方で，利用者はモジュールがやり取りするデータを意識する必要が生じるため，データフォーマットは十分に考える必要があると考えられる．

2.2.2 サービス間のデータフォーマットの差異

API で提供されるデータフォーマットが XML や JSON のように構造化されたデータが利用されている一方，API 間で利用される構造情報が違うことにより，自動変換ができない問題が指摘されている．下條ら [21] は，様々な種類のライフログのマッシュアップを行うために，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を提案と変換 API の実装を行なっている．森ら [22]，Sabbouh ら [23]，Maximilien ら [24] は異なる API 同士を結合する際のデータフォーマットとしてドメイン固有言語を作成することで容易なデータの接続を提案している．

これらアプローチは，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うことが可能である．しかし，既存のマッシュアップ手法においては，統一的なデータ

フォーマットは使われておらず，ツール内のプログラミングにおいて自分の扱いたい構造に変換するもの [13][14][17] や，そのままの生のデータを扱うもの [15] と手法に応じて様々である．これは，提案されているデータフォーマットが多数あり，未だ統一的なものができていないことによる．原因として．個人レベルで API が作成可能な現在において，すべてのデータフォーマットを統一的に扱うことは現実的ではない点や，適応させるには今まで作成された API をすべて変更する必要もある点であると考えられる．

2.2.3 プログラミングユーザの負担軽減

マッシュアップのモジュールを作成する際の開発の煩雑さも指摘されており，幸城ら [25] は，マッシュアップ開発の際に重複する処理を簡潔化するため，アスペクト指向プログラミングを導入し，マッシュアップアプリケーション間髪を容易にする手法を提案している．横山ら [20] は，Javascript を用いたマッシュアップを行うモジュール自身の開発を容易に行うために，Ajax や JSON 等の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の作成には共通処理が多く含まれており，開発者のモジュール開発の負荷を減らす必要性がある．

2.2.4 ユーザレベルの差異

今入ら [26] は，一般ユーザ，開発経験者，開発者，というユーザ形態に対して，ユーザのレベルに応じたマッシュアップの形態が，独立したツールとして提供されている点が問題であると述べている．また，視覚的なマッシュアップ手法の提案を行うことで，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である，on{X}[27] と Blocco[17] は，ユーザのレベルに応じてツールを提供している．on{X} は，開発者がアプリケーションを作成し，一般ユーザに対してアプリケーションを一部カスタマイズ可能な形で提供する．Blocco は，開発者がマッシュアップモジュールを作成し，一般ユーザがモジュールを利用して視覚的なマッシュアップを可能とする．

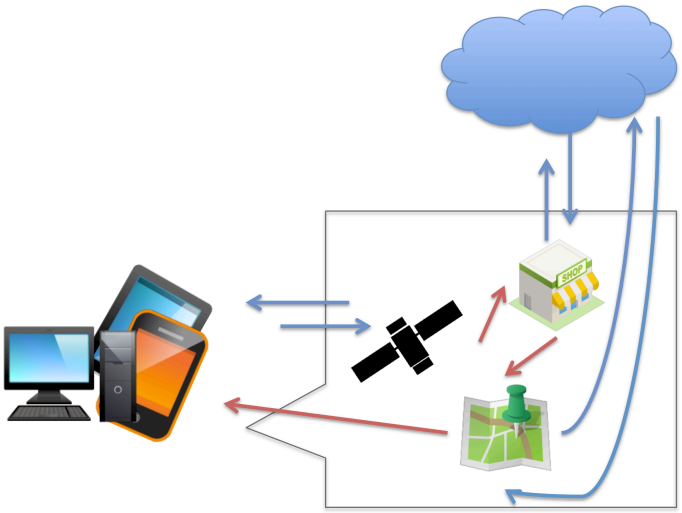

図 2.1.3: アプリケーションを開発するマッシュアップ

第3章　IDUMOの設計方針

本章では，IDUMO を汎用性と容易性を持つマッシュアップフレームワークと，実際にそれを用いた開発手法に分け，それぞれの設計方針を述べる．以降，マッシュアップフレームワーク設計を第 3.1 節において，開発手法の設計を第 3.2 節において議論する．

3.1　マッシュアップフレームワーク

本節では，IDUMO の基盤となるマッシュアップフレームワークの設計方針について述べる．IDUMO の望むマッシュアップフレームワークは以下の要件を満たす．

- すべてのデバイスや情報を包括的に扱い，マッシュアップによるアプリケーション開発を可能とする (汎用性)
- 簡単なマッシュアップを可能とする (容易性)

マッシュアップフレームワークの全体像を図 3.1.1 に示す．定められた「マッシュアップ記述手法」によりマッシュアッププログラムを記述する.「統一的入出力インタフェース」を備えたモジュールという単位により，機能特有の処理に依存しない形で，ネットワーク上の機能やデバイス上の機能を接続可能とする．モジュール間のデータの授受は「抽象化データフォーマット」によりやり取りされる．記述されたマッシュアッププログラムは,「統一的実行モデル」を備えた IDUMO バーチャルマシンにより，特定のプラットフォーム上で実行される．

以降，汎用性設計 (第 3.1.1 節) において「統一的入出力インタフェース」と「統一的実行モデル」を，容易性設計 (第 3.1.2 節) において,「抽象化データフォーマット」と「マッシュアップ記述手法」を述べる．

3.1.1　汎用性設計

現在利用されているアプリケーションは，様々なサービスやデバイスを連携させて作成されている．また，ユーザが持つデバイスもパソコン，スマートフォン，情報家電と様々である．そのため，自由なアプリケーション開発を可能とするフレームワークの実現には，様々なサービスを制限なく扱え，自らの持つデバイス上で実行可能であることが望まれる．しかし，このようなマッシュアップフレームワークの実現には,「リソース毎のアクセス方法の差異」と「デバイス間の実行モデルの差異」が存在する．

ネットワークに接続されたデバイス上の機能や，ネットワーク上のサービスをマッシュアップの対象とする際，多種多様なリソースを取り扱うことになるが，リソースへのアクセス方法はそれぞれ異なる．例えば，ネットワーク上のサービスを用いる場合は REST を用いた XML ベースのアクセスを行い，センサデバイス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

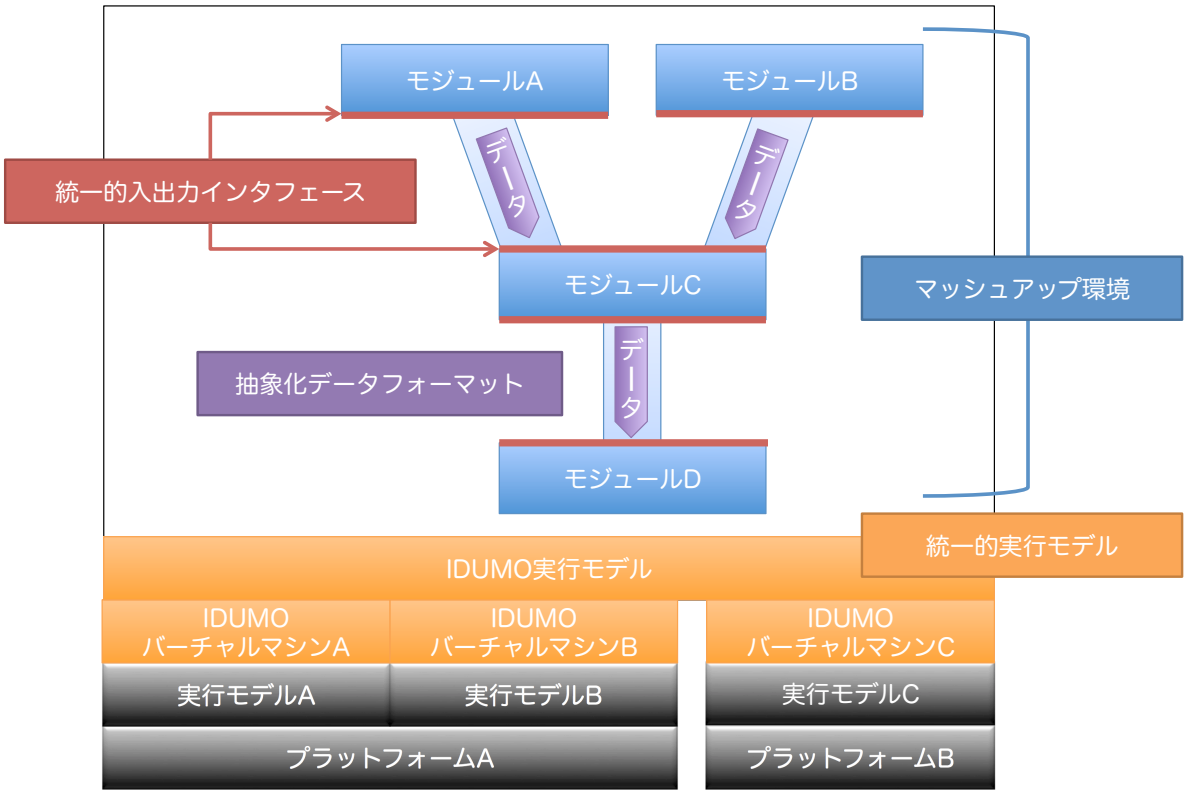

図 3.1.1: マッシュアップフレームワーク

得するといった，それぞれの異なったリソースへのアクセス方法が存在する．実際のプログラミングにおいてこれらリソースを扱う際には，それぞれのリソースの仕様を確認しながら開発が行われる．しかしながら，複数のリソースを組み合わせるマッシュアップを行う際に，リソースごとの複雑な仕様を逐次確認することは，開発コストの増加と，メンテナンス性の低下を招く．すなわち，自由にマッシュアップを行いアプリケーションを作成することが妨げられる．この問題を解消するためには，他のリソースとの接続部分は共通のインタフェースを用いつつ，それぞれのリソースへの取り扱いには個別のプロトコルを扱えるようにする必要がある．IDUMO では，他のリソースとの接続は「データを受け渡す」「データを受け取る」といった，データの受け渡しのみで行うこととする．リソース間連携をデータの受け渡しとした統一的な入出力インタフェースを備えることにより，ストリーミングデータの取得や，ネットワーク上のデータの取得といったリソース特有の処理をモジュール内に隠蔽化し記述することが可能とする．

各自の持つデバイスでアプリケーションを実行可能とする際に，それぞれのデバイスの実行モデルが異なることが問題となる．例えば，パソコン上のコンソールアプリケーションでは，アプリケーションの処理が逐次的に実行されるのに対して，AndroidOS 上のアプリケーションでは，アプリケーションの処理がシステムの発行するイベントにより制御される．このような実行モデルの差異は，実行デバイスのシステムの制約により発生するため，実行モデル自身の変更は不可能である．そのため，マッシュアップで作成されたアプリケーションをデバイス上で実行する際は，それぞれの実行モデル上で動作させる必要がある．そこで，まずデバイスの実行モデルと非依存な，マッシュアッププラットフォーム自体の実行モデルの定義を行う．デバイスの実行モデルとマッシュアッププラットフォームの実行モデルは，デバイスの実行モデルごとに実装される VirtualMachine によって差異の吸収が行われ，デバイスに影響されないプラットフォームを設計する．

3.1.2 容易性設計

ユーザがアプリケーションを開発する際には，開発容易性は非常に重要となる．しかし，その結果アプリケーション開発の自由度を下げることは望ましくない．そこで，汎用性を限定することなく，容易な設計も担保する必要性がある．

様々なサービスを対象としたマッシュアップでは，データをマッシュアップする手法とは異なり，テキスト処理を行うための文字列情報のみ，統計処理を行うための数値情報のみなど，扱うデータを限定することができない．例えば，各種センサデバイスを利用する際には生の数値を扱う一方で，ネットワーク上のサービスを利用する際には画像を扱うことも考えられる．そのため，ユーザはアプリケーションを開発する際には，モジュールがやり取りするデータを意識し，適切に接続する必要性が生じ，複雑な行為を要求される．一方で，地図を表示するといったモジュールを考えると，このモジュールは入力として緯度の数値情報と経度の数値情報を必要とする．また，GPS センサのモジュールは緯度の数値情報と経度の数値情報を持っており，飲食店情報検索サイトの API においても，緯度の数値情報と経度の数値情報も持っているであろう．このようにデータ構造には自明なものも存在するが，これらを一つ一つ接続を行うことは非常に手間のかかる行為となる．そこで，モジュール間で授受されるデータはデータ構造を持つものとしておき，自明なものは単純なモジュール同士の接続で受け渡し可能とする．その上で，必要な際には生データまで意識させたデータの接続を可能とすることで，アルゴリズムの汎用性をもたせたまま，マッシュアップ構築を容易なものとする．

マッシュアップの記述方法の複雑化は，利用できるユーザが限定されてしまい，望ましくない．

そこで，マッシュアップの記述を最小限に抑えアプリケーションを開発可能であることが望まれる．アプリケーション開発を行う際には,「どのような機能を持ったアプリケーションを作成したい」といった情報と「どのようにアプリケーションを動作させる」といった情報を記述できれば良い．そこで，マッシュアップ環境に必要な情報を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「繰り返し回数」,「繰り返し間隔時間」のみとすることで，マッシュアップ構築を容易なものとする，

3.2 開発手法

本節では，IDUMO の開発手法をエンドユーザとプログラミングユーザにわけ設計方針を述べる．開発手法の関係を図 3.2.1 に示す．IDUMO では，エンドユーザとプログラミングユーザを以下のように分ける．

- エンドユーザ (第 3.2.1 節)
  - マッシュアップユーザ
  - カスタマイズユーザ
  - 一般ユーザ
- プログラミングユーザ (第 3.2.2 節)
  - フレームワーク作成ユーザ
  - モジュール作成ユーザ

以降，エンドユーザ設計 (第 3.2.1 節) において「一般ユーザ」と「カスタマイズユーザ」と「マッシュアップユーザ」を，プログラミングユーザ設計 (第 3.2.2 節) において「モジュール作成ユーザ」と「フレームワーク作成ユーザ」を述べる．

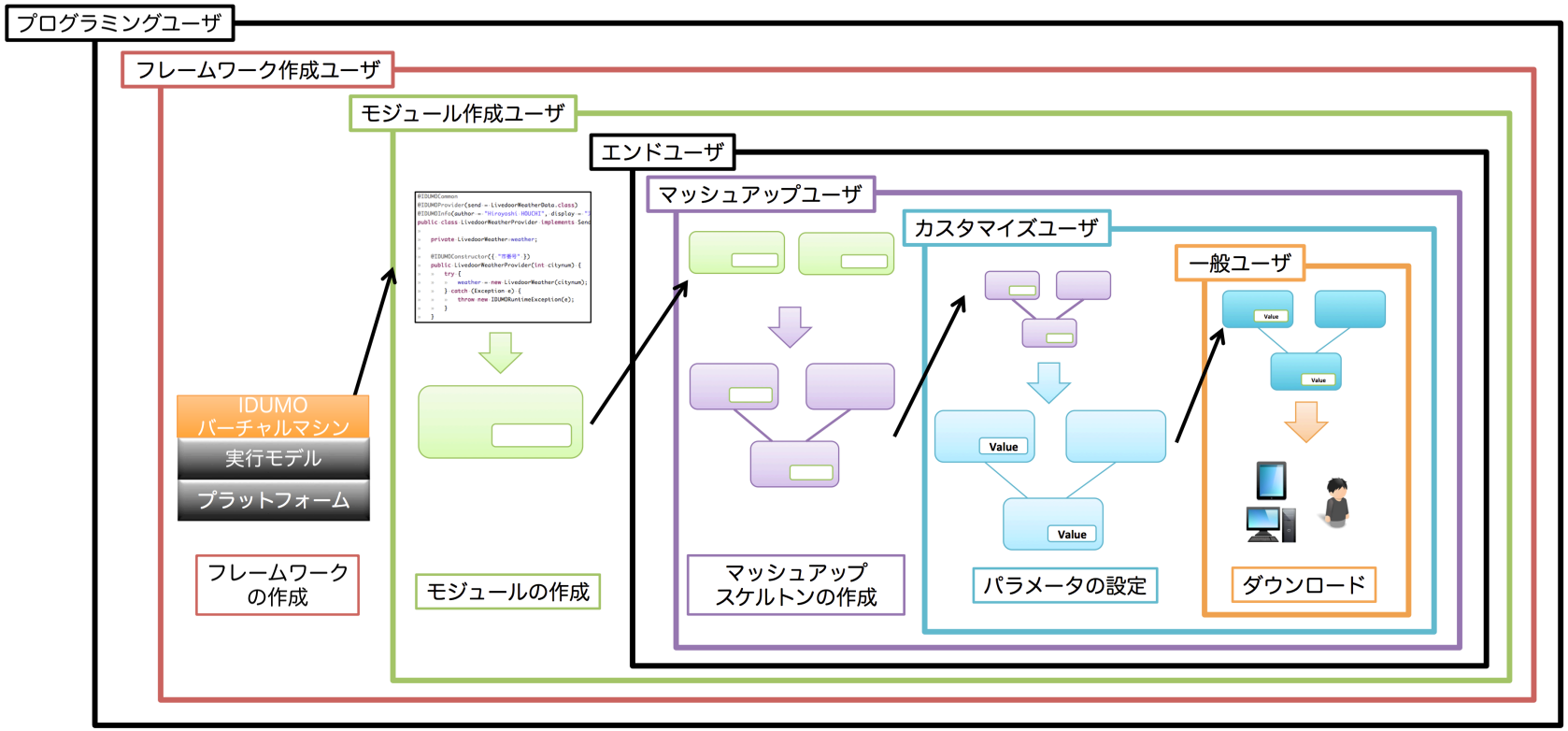

図 3.2.1: 開発者レベルに応じたツールの活用手法

3.2.1 エンドユーザ設計

本節では，エンドユーザのレベルを「一般ユーザ」と「カスタマイズユーザ」と「マッシュアップユーザ」に分け，設計方針を述べる．各ユーザは，グラフィカルな環境を用いることで，簡単にマッシュアップフレームワークの活用を可能とする．

一般的に，パソコンやスマートフォンを利用するユーザは，すでにでき上がったアプリケーションをダウンロード，インストールすることで，そのアプリケーションの利用を行う．このようなユーザに対して，いきなりマッシュアップを行うアプローチのみを提供しても，機能を組み合わせることができず，アプリケーションを開発することができないと考える．そこで，一般ユーザには，マッシュアップユーザによって作られたアプリケーションをダウンロード，インストール可能な仕組みを提供する．また，仕組みの提供の際に，マッシュアップ環境を見せることで，今後のマッシュアップユーザへの転向を目指す．

パソコンで提供されるアプリケーションや，スマートフォンに標準でついているアラームアプリケーションなど，多くのアプリケーションには，ユーザに一部カスタマイズ可能な環境を提供することで，自分にあったアプリケーションを動作させることが可能な仕組みが搭載されている．例えば，アラームアプリケーションでは，はじめから時間が決め打ちされているわけではなく，ユーザが自分の望む時間にアラームを鳴らすことを可能とする入力ボックスが存在する．このことは，多数のユーザは，アプリケーションを一部変更して自分の望むアプリケーションを開発可能であると考える．そこで，一般ユーザの次の段階として，マッシュアップ内容の一部をカスタマイズ可能な仕組みを提供する．マッシュアップユーザによって作られたアプリケーションを一部変更可能な状態で提供することで，天気予報で表示する地域や，メールの送信先を変更することを可能とする．一般ユーザと同様に，マッシュアップ環境を見せることで，マッシュアップユーザへの転向を目指す．また，カスタマイズユーザがカスタマイズしたアプリケーションを公開する仕組みの提供を行うことで，一般ユーザに対してアプリケーションを提供できるよう支援する．

マッシュアップユーザは，IDUMO のマッシュアップによりアプリケーションを開発するユーザである．第 3.1.2 節で述べたように,「モジュールの定義」「モジュールの接続」「繰り返し回数」「繰り返し間隔」の定義のみでアプリケーションを開発する．この定義をグラフィカルに行える環境を作ることで，ユーザにとって簡単にマッシュアップ可能とする．ユーザは，モジュール開発者によって作成されたモジュールを用いることでマッシュアップを行う．また，ユーザによって作られたアプリケーションを「カスタマイズユーザ」「一般ユーザ」に提供できるようにすることで，他のユーザの開発を支援する．

3.2.2 プログラミングユーザ設計

プログラミングユーザは，IDUMO フレームワークやモジュールをプログラミングにより作成するユーザである．エンドユーザは，プログラミングユーザによって作成されたフレームワークやモジュールを用いて開発を行う．

モジュール作成ユーザは，決められた規約に従い，マッシュアップモジュールを作成するユーザである．ユーザはオブジェクト指向プログラミングのクラスを作るように，特定の機能を持つモジュールを作成する．作られたモジュールは，マッシュアップユーザに提供され，実際にマッシュアップフレームワークで使われることとなる．また，自分の必要とし，かつ，提供されていないモジュールを作成することで，自身のアプリケーション開発を簡略化することも可能である．

フレームワーク作成ユーザは，特定の実行モデルに合わせたバーチャルマシンを作成するユーザである．ユーザは，マッシュアップフレームワークの仕様を理解し，新しいデバイス向けにバーチャルマシンを提供する．マッシュアップアプリケーションは，このユーザが作成したバーチャルマシン上で実行されることになる．

第4章　マッシュアップフレームワークの実装

第 3 節で述べた設計に基づき，マッシュアップフレームワークの実装について述べる．なお，マッシュアップフレームワークの実装には Java を用いているため，本章の説明では，Java の用語を用いる．また，今回は処理のフローをイベントドリブンではなく，逐次処理の方式を持たせて実装を行った．

4.1　統一的な入出力インタフェース

IDUMO では，デバイス上の機能やネットワークサービスといったリソースを統一的に扱うため，リソース間の接続をデータの授受のみとするインタフェースを用いる．インタフェースは，データを受け渡す機能を持つインタフェース (Sendable) と，データを受け取る機能を持つインタフェース (Receivable) に分割される．他のサービスにデータを受け渡すサービスは Sendable を実装し，他のサービスからデータを受け取るサービスは Receivable を実装する．また，他のサービスからのデータを受け取り，他のサービスへデータを受け渡すサービスは Sendable と Receivable の両方を実装する．授受されるデータは，次節で述べるデータフォーマットを用いる．

4.2　データフォーマット

アルゴリズムの汎用性と，マッシュアップの容易性設計のためには，データ構造を用いたアクセス，および，生データへ直接アクセスする方法両方を備えた，データフォーマットが必要となる．この要求を満たすため，データの保存自体は，ハッシュデータとして保存を行い生データへアクセスできるようにする．データ構造は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で，そのデータフォーマットがどの型情報を表すかを示す．

例として緯度経度を表すデータフォーマット LatLngModel を図 4.2.1 に示す．1 行目において，IfLatLngElement の実装定義を行い，LatLngModel が緯度経度の情報を持っていることを明示している．10 行目，14 行目において，実際に緯度経度返すメソッドを実装しており，これにより緯度経度情報を必要とするモジュールが適切に緯度経度情報にアクセス可能となる．11 行目，15 行目で確認できるように，データ要素へ直接アクセスできる機能も持っており，これにより汎用的な統計処理や，利用者がデータ型の一部のみを取り出すことも可能としている．

4.3　プログラム実行モデル

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，その「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」部分に分離する (図 4.3.1)．デバイスに

非依存な部分 (Component) 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限回といったアプリケーションそのものの処理を記述する．その後デバイスに依存する部分 (VirtualMachine) が Component を解釈して，アプリケーションを実行させる．

4.4 マッシュアップ情報の記述

簡単に，かつ，特定のプラットフォーム非依存なマッシュアップ情報を記述するため，マークアップ言語である XML を中間言語として用いる．XML の例として，GPS センサの値を用いて，現在地の地図を表示させるマッシュアップを図 4.4.1 に示す．3 行目から 11 行目において，モジュールのマッシュアップ情報が記述される．5 行目と 6 行目で,「利用するモジュールの定義」を行い，9 行目で,「モジュールの接続」を行う．また，12 行目から 14 行目ではアプリケーションの動作方法が定義され，13 行目では 1000ms のインターバルで「ループ実行」されることが定義されている．このような簡単な情報のみでマッシュアップを可能とする．また次章において，より簡単な GUI 環境についても述べる．

```
1  public class LatLngModel extends AbstractData implements IfLatLngElement {
2   private static final String LATITUDE = "latitude";
3   private static final String LONGITUDE = "longitude";
4   public LatLngModel(double lat, double lng) {
5    add(new IDUMODataTypeRawNumber(LATITUDE, lat, "Latitude"));
6    add(new IDUMODataTypeRawNumber(LONGITUDE, lng, "Longitude"));
7   }
8
9   @Override
10   public double getLatitude(){
11    return (Double) getValue(LATITUDE);
12   }
13  @Override
14   public double getLongitude(){
15    return (Double) getValue(LONGITUDE);
16   }
17 }
```

図 4.2.1: 緯度経度のデータ型を扱うクラス

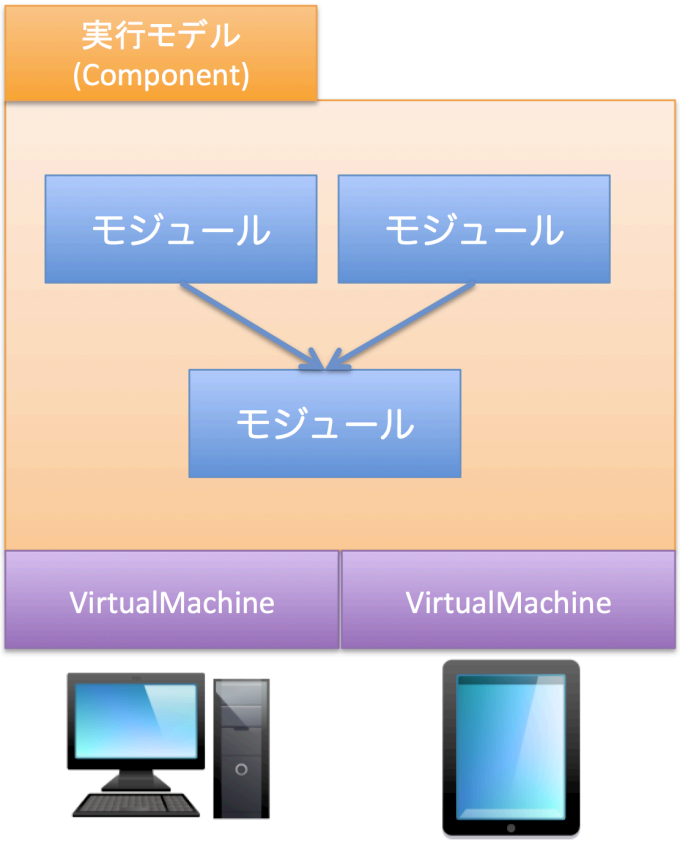

図 4.3.1: 実行デバイスに応じたラッピング

```
<?xml version='1.0' encoding='UTF-8'?>
<idumo ver='1.0' name='GPS2MapView'>
 <mashup>
  <items>
   <item class='AndroidGPSProvider' name='idumo0' />
   <item class='AndroidMapViewReceiptor' name='idumo1' />
  </items>
  <connections>
   <connect src='idumo0' sink='idumo1'/>
  </connections>
 </mashup>
 <setting>
  <loop interval='1000'/>
 </setting>
</idumo>
```

図 4.4.1: 現在地を GoogleMap に表示する Android アプリケーションの XML 表現

第5章　開発手法の実装

本章では，エンドユーザに向けた開発手法についての実装を述べる．マッシュアップ環境は，Web上で実装し，開発ツールのインストール作業など，ユーザに要求される煩雑な手順を回避する．

5.1　マッシュアップユーザ

マッシュアップユーザに対して，図 5.1.1 のような環境でマッシュアップフレームワークを提供する．ユーザは左手にあるモジュールをクリック動作により選択することで，モジュールがグラフィカルに表示される．モジュールを配置したあと，それぞれのモジュールに定義される点同士を，ドラッグ操作によりつなぎ合わせることで，モジュールの連携を可能とする．マッシュアップの記述をグラフィカルな環境で行ったあと，アプリケーションの名前の設定と，繰り返しを行うかを選択し，Download ボタンを押すことで，ユーザはアプリケーションをダウンロードが可能となる．

ダウンロードボタンが押されたあと，サーバはユーザのマッシュアップ情報を XML に変換し，そこから，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．

5.2　カスタマイズユーザ

カスタマイズユーザに対して，図 5.2.1 のような環境を提供する．ユーザは左手にあるアプリケーション名をクリックすると，右手のようにマッシュアップされた情報が表示される．マッシュアップ情報は書き換え可能な形で提供されており，ユーザはモジュールに与える引数の値の変更や，実際につながるモジュールを変更することが可能である．マッシュアップの記述をカスタマイズしたのち，Download ボタンを押すことで，ユーザはアプリケーションのダウンロードができる．

5.3　一般ユーザ

一般ユーザに対して，図 5.3.1 のような環境を提供する．ユーザは左手にあるアプリケーション名をクリックすると，右手のようにマッシュアップされた情報が表示される．カスタマイズユーザとは異なり，ユーザはマッシュアップ情報を変更することはできない．マッシュアップ情報を定義した状態で Download を押すと，ユーザはアプリケーションのダウンロードが可能となる．

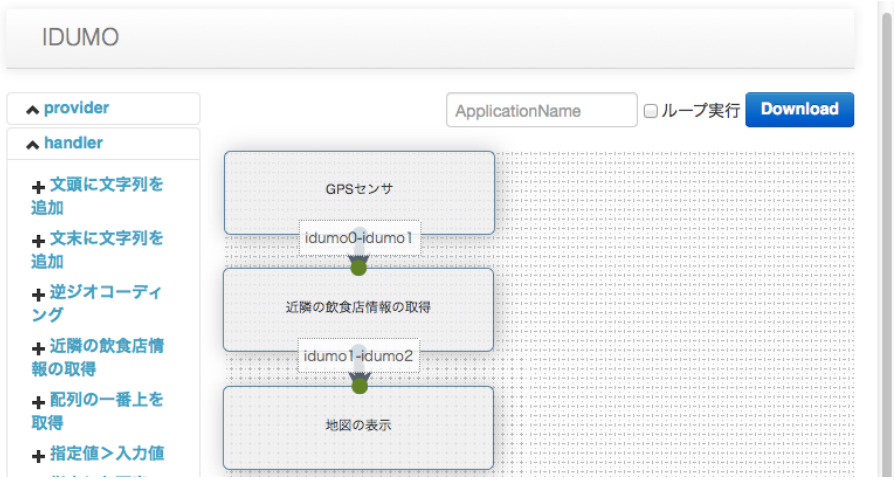

図 5.1.1: マッシュアップユーザ環境

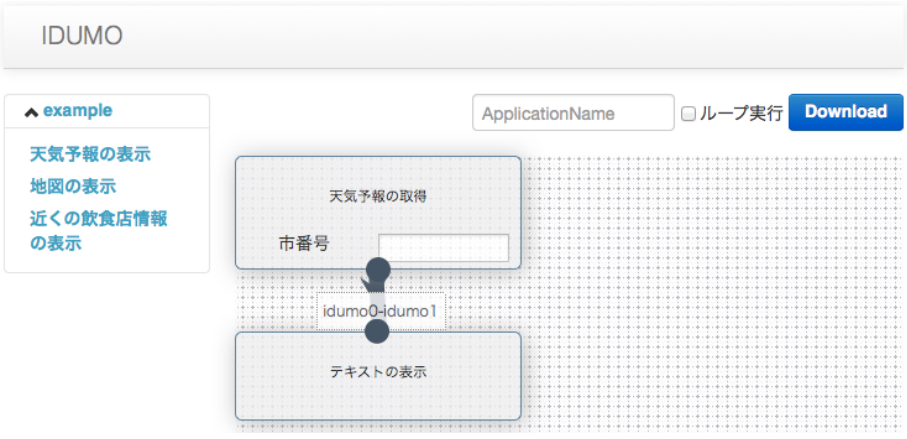

図 5.2.1: カスタマイズユーザ環境

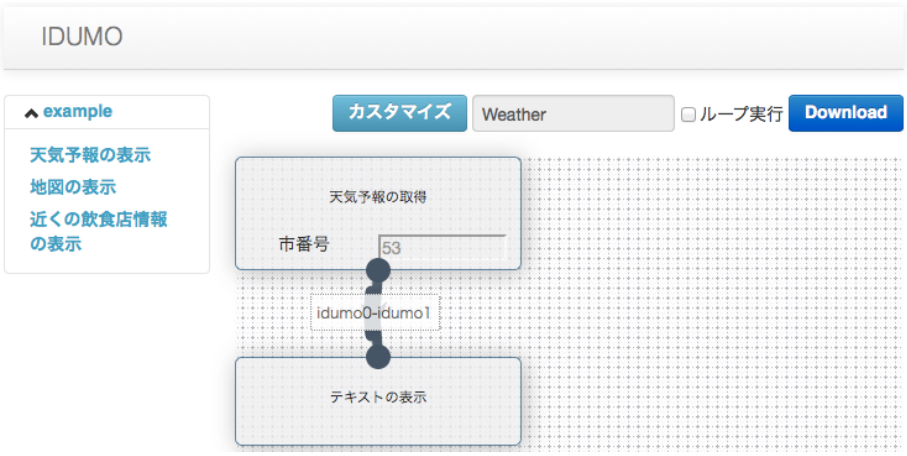

図 5.3.1: 一般ユーザ環境

第6章　ケーススタディ

本章では，IDUMO フレームワークを用いた，実際のアプリケーションの開発と考察を行う．

6.1　一般ユーザによる現在地住所の取得

ケーススタディとして，IDUMO で提供されるアプリケーションをインストールするケースを想定する．このケースでは，ユーザは既存の現在地住所取得アプリケーションをマッシュアップフレームワーク上に表示し，ダウンロードすることで，アプリケーションを活用する．

IDUMO を用いてダウンロードを行う環境を図 6.1.1 に示す．ユーザは左側にある example から「現在地住所の取得」をクリックすると，右側のマッシュアップ環境が表示される．ユーザは何も変更することなく，Download ボタンを押してアプリケーションのインストールを行う．

インストールされたアプリケーションの実行結果を図 6.1.2 に示す．実行結果として，現在地の情報「東京都調布市富士見町二丁目」という情報が表示され，アプリケーションが実行できていることがわかる．簡単な動作でマッシュアップフレームワークによって作られたアプリケーションがインストールできた．

6.2　カスタマイズユーザによる東京の天気予報の取得

ケーススタディとして，IDUMO で提供されるアプリケーションの一部をカスタマイズして，自分の望む情報を表示するケースを想定する．このケースでは，ユーザはマッシュアップユーザにより作成された,「天気予報情報の取得」アプリケーションを，自分の地域に変更して，インストールすることで，アプリケーションを活用する．

IDUMO を用いてカスタマイズを行う環境を図 6.2.1 に示す．ユーザは左側にある example から「天気予報情報取得」をクリックし，その後，ページ内にあるカスタマイズを行うことで，既存のマッシュアップを編集することが可能となる．その後，東京を表す「63」を入力し，Download ボタンを押してアプリケーションのインストールを行う．

インストールされたアプリケーションの実行結果を図 6.2.2 に示す．実行結果として，東京の天気予報情報が表示される．モジュール内の一部書き換えのみで，自らが望むアプリケーションが開発可能であることがわかる．

6.3　マッシュアップユーザによる近隣の飲食店の取得

ケーススタディとして，ユーザが新しい土地で食事をするケースを想定する．このケースでは，ユーザはデバイスの GPS の情報と近隣の飲食店と地図上に表示するモジュールをマッシュアップし，アプリケーションを開発する．

図 6.1.1: 現在地住所を表示するダウンロード環境

図 6.1.2: 現在地住所を表示するアプリケーション

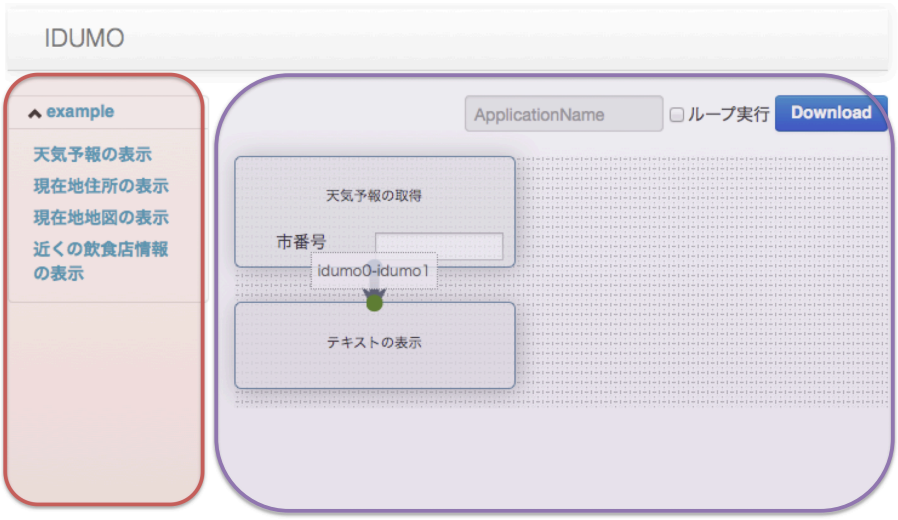

図 6.2.1: 東京の天気予報を表示するカスタマイズ環境

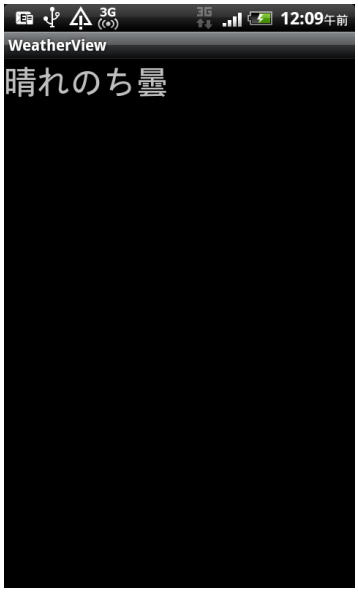

図 6.2.2: 東京の天気予報を表示するアプリケーション

IDUMO を用いてマッシュアップを行った図を図 6.3.1 に示す．ユーザは，左側にある provider から「GPS センサ」，handler から「近隣の飲食店情報の取得」，receiptor から「地図の表示」を選択し，フレームワーク上に表示する．その後,「GPS センサ」と「近隣の飲食店情報の取得」,「近隣の飲食店情報の取得」と「地図の表示」をドラッグ動作により接続する．アプリケーションの名前を設定し，ループ実行にチェックを入れ，Download ボタンを押すと，IDUMO によって作られたマッシュアップアプリケーションが，ユーザにダウンロード，インストールされる．

インストールされたアプリケーションの実行結果を図 6.3.2 に示す．実行の結果，ユーザの付近にある飲食店が地図上にグラフィカルに表示されていることがわかる．これにより，ユーザは近くの飲食店を知ることが可能である．

6.4 モジュール作成ユーザによる新しいモジュールの開発

ケーススタディとして，自らが望むモジュールが存在しないケースを想定する．このケースでは，プログラミング可能なユーザが,「数字を緯度経度に変換」する新しいモジュールの開発と，それを用いたマッシュアップアプリケーションの開発を行う．また，マッシュアップユーザに提供されるグラフィカルフレームワークを使わず，XML を用いてパソコン上で動作するコンソールアプリケーションを開発する．マッシュアップアプリケーションの内容は「特定位置の飲食店を平均価格順に表示する」アプリケーションである．

ユーザは IDUMO のソースコードを取得し，自らで「数字情報を緯度経度情報に変換」するモジュールを作成する．作成したモジュールを図 6.4.1 に示す．実際にデータの授受を表すメソッドは 24 行目に定義されている onCall である．25 行目と 26 行目において，データを受け渡すモジュールからデータの受け取りを行なっている．27 行目において，受け取ったデータを緯度経度情報を表す LatLngModel に変換し，38 行目において，次のモジュールへデータを受け渡している．

その他の部分は，IDUMO の特有の処理である．16 行目に定義されている isReady は，モジュールの準備ができているかどうかを表す．32 行目に定義されている receivableType は受け取るデータ型を表す．37 行目に定義されている sendableType は，受け渡すデータ型を表す．42 行目に定義されている setSender は，接続されるモジュールが適切であるかのチェックと接続を行う．

マッシュアップの記述情報を図 6.4.2 に示す．5 行目と 8 行目において，数値を入力するための NumberProvider の定義を行い，6 行目と 9 行目で，実際の座標を入力している．11 行目では，今回作成された数値を緯度経度に変換する，Number2LatLngAdaptor の定義を行い，12 行目では，座標から飲食店を取得する，HotpepperHandler の定義を行う．13 行目では，得られたデータの特定要素についてソートを行う，SortHandler の定義を行い，14 行目で実際にソートを行う要素の指定を行なっている．16 行目では，受け取ったデータをコンソール画面に表示する，ConsoleViewReceiptor の定義を行なっている．また，18 行目以降では，それぞれのモジュールの接続を行なっている．

XML を変換し，生成されたアプリケーションの実行結果を図 6.4.3 に示す．アプリケーションが実行され，店舗名と価格が表示されていることがわかる．

6.5 考察

本節では，マッシュアップフレームワークに必要な要件である汎用性設計と容易性設計が満たされたかどうかを，ケーススタディを通して考察する．

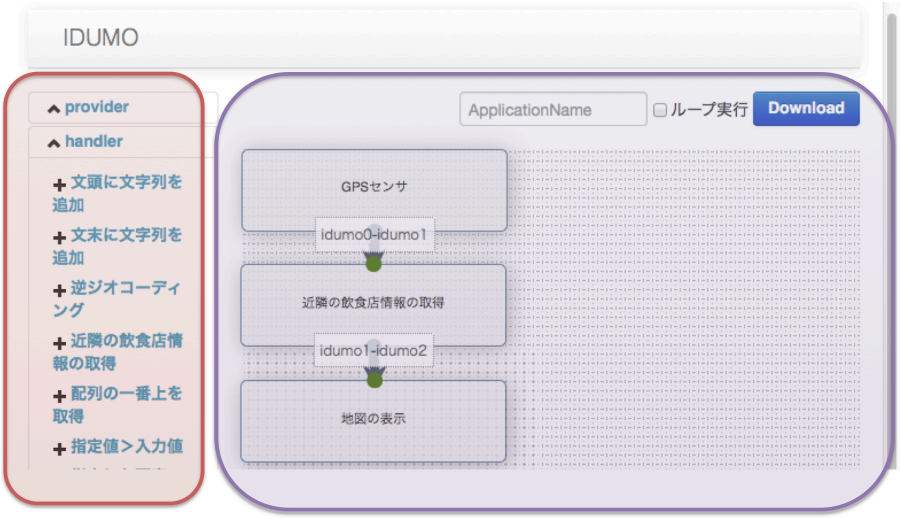

図 6.3.1: 近隣の飲食店を地図上に表示するマッシュアップ

図 6.3.2: 近隣の飲食店を地図上に表示するアプリケーション

```java
@IDUMOCommon
@IDUMOAdaptor(receive = {IfNumberPrimitiveElement.class,
    IfNumberPrimitiveElement.class}, send = LatLngModel.class)
@IDUMOInfo(author = "Hiroyoshi HOUCHI", display = "数字を緯度経度に変換
    ", summary = "")
public class Number2LatLngAdapter implements IfSendable, IfReceivable {

 private IfSendable   sender1;
 private IfSendable   sender2;

 private IfReceiveValidator vSize = new ReceiveValidatorSize(2);
 private IfReceiveValidator vType1 = new ReceiveValidatorType(1,
     IfNumberPrimitiveElement.class);
 private IfReceiveValidator vType2 = new ReceiveValidatorType(2,
     IfNumberPrimitiveElement.class);

 public Number2LatLngAdapter() {}

 @Override
  public boolean isReady() {
   boolean flag = true;
   flag = flag && sender1.isReady();
   flag = flag && sender2.isReady();
   return flag;
  }

 @Override
  public FlowingData onCall() {
   IfNumberPrimitiveElement num1 = (IfNumberPrimitiveElement) sender1.
       onCall().next();
   IfNumberPrimitiveElement num2 = (IfNumberPrimitiveElement) sender2.
       onCall().next();
   LatLngModel latlng = new LatLngModel(num1.getNumber(), num2.getNumber
       ());
   return new FlowingData(latlng);
  }

 @Override
  public ConnectDataType receivableType() {
   return new MultiConnectDataType(IfNumberPrimitiveElement.class,
       IfNumberPrimitiveElement.class);
  }

 @Override
  public ConnectDataType sendableType() {
   return new SingleConnectDataType(LatLngModel.class);
  }

 @Override
  public void setSender(IfSendable... senders) throws IDUMOException {
   vSize.validate(senders);
   vType1.validate(senders);
   vType2.validate(senders);
   sender1 = senders[0];
   sender2 = senders[1];
  }

}
```

図 6.4.1: 数字を緯度経度に変換するモジュール

```
<?xml version='1.0' encoding='UTF-8'?>
<idumo ver='1.0' name='HotpepperSortTest'>
 <mashup>
  <items>
   <item class='NumberProvider' name='idumo0'>
    <argument id=0>34.67</argument>
   </item>
   <item class='NumberProvider' name='idumo1'>
    <argument id=0>"135.52"</argument>
   </item>
   <item class='Number2LatLngAdaptor' name='idumo2'/>
   <item class='HotpepperHandler' name='idumo3'/>
   <item class='SortHandler' name='idumo4'>
    <argument id=0>"average"</argument>
   </item>
   <item class='ConsoleViewReceiptor' name='idumo5' />
  </items>
  <connections>
   <connect src='idumo0' sink='idumo2'/>
   <connect src='idumo1' sink='idumo2'/>
   <connect src='idumo2' sink='idumo3'/>
   <connect src='idumo3' sink='idumo4'/>
   <connect src='idumo4' sink='idumo5'/>
  </connections>
 </mashup>
 <setting>
  <loop interval='1000'/>
 </setting>
</idumo>
```

図 6.4.2: 特定位置の飲食店を平均価格順に表示するマッシュアップ情報

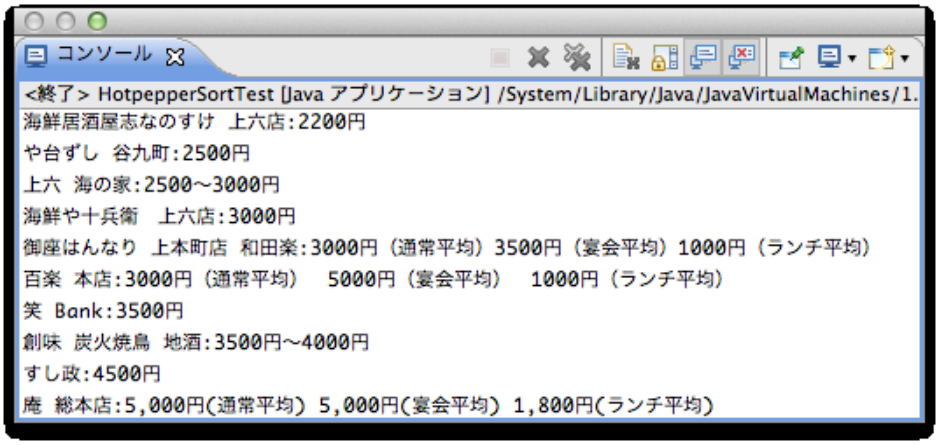

図 6.4.3: 特定位置の飲食店を平均価格順に表示するアプリケーション

ネットワークに接続されたデバイス上の機能や，ネットワーク上の機能をマッシュアップ対象とする際は，それぞれの異なるアクセス方法を持つため，統一的な入出力インタフェースの定義が必要であることを述べた，ケーススタディ 1（第 6.1 節）に注目すると,「天気予報の取得」というネットワーク上の機能 (REST を用いて XML を取得する) と「テキストの表示」というデバイス上の機能 (デバイス上の画面に文字を出力する) が利用されている．それぞれは，ネットワークやデバイスに異なるアクセス方法を持つモジュールであるが，結果を見ると適切にアプリケーションが実行されていることがわかる．

各自の持つデバイスでアプリケーションを実行する際は，それぞれが異なる実行モデルを持つため，統一的な実行モデルの定義が必要であることを述べた．ケーススタディ 1 (第 6.1 節) とケーススタディ 4 (第 6.4 節) に注目すると，AndroidOS 上で動作するアプリケーション (イベントドリブンな実行モデルを持つ) とパソコンのコンソール上で動作するアプリケーション (逐次的な実行モデルを持つ) がそれぞれ開発されている．それぞれは，異なる実行モデルを持ち，アプリケーションを動作させる環境であるが，結果から，適切にアプリケーションを実行されていることがわかる．

サービス間を連携させる際は，マッシュアップの汎用性を削ぐことなく，それぞれのデータの授受を容易とするため，適切なデータフォーマットの定義が必要であることを述べた．ケーススタディ 3 (第 6.3 節) に注目すると,「GPS センサ」と「近隣の飲食店情報を取得」と「地図上の表示」といったサービス間の連携が，モジュールの接続のみで行えていることがわかる．また，ケーススタディ 4 (第 6.4 節) に注目すると,「SortHandler」が「HotpepperHandler」の average という任意の要素に対してソートできていることがわかる．このように，データフォーマットを扱い容易なサービス間を実現している一方で，汎用的なマッシュアップも可能となっている．

マッシュアップの記述が複雑であった際，利用されるユーザが限定的となるため，マッシュアップの記述を最低限とすることが必要であることを述べた．ケーススタディ 3 (第 6.3 節) に注目すると，ユーザはモジュールをクリックし，表示されたモジュールを接続する，および，アプリケーション名と繰り返し実行をするかのみでマッシュアップを可能としている．また，ケーススタディ 1 (第 6.1 節)，ケーススタディ 2 (第 6.2 節) のマッシュアップ情報と実行結果から，様々なアプリケーションを作成できることがわかる．このように簡単にかつ様々なアプリケーション開発が可能となっている．

ユーザ毎の開発手法に注目する．エンドユーザはそれぞれのレベルに合わせた開発手法を利用することができ，自分の望むアプリケーションが容易に取得，開発可能となっている．また，プログラミングユーザであっても，自分の望むモジュールの開発のみでアプリケーションを開発可能であり，開発したいアプリケーションを容易に開発できると考える．例えば，ケーススタディ 4 (第 6.4 節) で開発したモジュールのコード量は 54 行であり，また，実際の処理を行なっている部分は 4 行に過ぎない．モジュール分割されていることで，他のモジュールとの依存性も低くなるため，開発が容易になると考えられる．

第7章　結論

本論文では，増えゆくネットワークサービスやデバイスを包括的に扱えるマッシュアップフレームワーク IDUMO の提案と実装を行った．既存のマッシュアップ手法では，以下の要件を満たすことができなかった．

- ネットワークサービスやデバイスの機能を気にせずにマッシュアップ
- マルチプラットフォーム対応
- エンドユーザの簡単なマッシュアップ

そこで IDUMO では，「統一的入出力インタフェース」「統一的実行モデル」「データフォーマット」を定義することで，ネットワークサービスやデバイスを包括的にマッシュアップ可能とした．また，ユーザの開発環境を複数提案することにより，それぞれのユーザに対してアプリケーション開発可能な環境を提供した．ケーススタディの結果，エンドユーザであっても自分のレベルに応じて簡単にアプリケーションを開発可能となった．IDUMO は，家に帰ったら「電気を付ける」「エアコンを付ける」「お香をたく」「家の鍵をしめる」といった個人の要望に応じたアプリケーションがユーザ自身の手により自由に作成できるようなアプリケーションプラットフォームのデファクトスタンダートになることを目指す．

今後の展望として，各開発手法で作成されたコンポーネントを他開発手法へ受け渡せる仕組みを作り，マッシュアップフレームワークの拡充を目指す．

謝辞

本論文は，筆者が電気通信大学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過程に在籍中の研究成果をまとめたものである．

本研究を進めるにあたり，日々の研究指導を戴いた，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ネットワークコンピューティング講座，吉永努教授に感謝する．研究の進め方が稚拙な著者を，導いて戴けたことを深謝する．

また，株式会社イーツリーズ・ジャパン三好健文氏，同講座入江英嗣准教授，同講座吉見真聡助教から，研究を進める際の助言を多く戴いたこと，感謝する．御三方の助言により，研究がより良い物となった．

最後に，講座のメンバーと環境に感謝する．修士２年間という短い期間だったが，合宿や合同ゼミ，研究と充実した生活を送り，貴重な財産を得ることができた．

参考文献

[1] お天気 Web サービス仕様 - Weather Hacks - livedoor 天気情報. http://weather.livedoor.com/weather_hacks/webservice.html.

[2] 提供 API 一覧 — ロケタッチ Developers. http://tou.ch/developer/api_all?uri=railway.

[3] Dropbox - Simplify your life. https://www.dropbox.com/.

[4] Gmail : Email from Google. http://mail.google.com/.

[5] Apple - iOS 6 - Use your voice to do even more with Siri. http://www.apple.com/jp/ios/siri/.

[6] コンシェル — サービス・機能 — NTT ドコモ. http://www.nttdocomo.co.jp/service/customize/iconcier/.

[7] Twitter. https://twitter.com/.

[8] Facebook. http://www.facebook.com/.

[9] ケータイホームシステム. http://www.docomo.biz/html/service/homesystem/.

[10] Pluto. http://pluto.io/.

[11] Mashup Awards. http://mashupaward.jp/.

[12] 藤井章博, 河本洋. 科学技術動向　2010 年 1 月号. pp. 16–26, 2010.

[13] iGoogle. http://www.google.co.jp/ig.

[14] Desktop gadgets - Microsoft Windows. http://windows.microsoft.com/en-US/windows/downloads/personalize/gadgets.

[15] Pipes : Rewire the web. http://pipes.yahoo.com/pipes/.

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

[17] 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合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/.

[18] Plagger. http://plagger.org/.

[19] 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. *WebDB Forum 2008, 2A-1*.

[20] 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 javascript 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.

[21] 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マイニング及び一般). 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.

[22] 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.

[23] Marwan Sabbouh, 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.

[24] 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.

[25] 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)). 情報処理学会研究報告. ソフトウェア工学研究会報告, Vol. 2008, No. 29, pp. 171–177, mar 2008.

[26] 今入庸介, 小板隆浩. 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・システム及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研究報告. MoMuC, モバイルマルチメディア通信, Vol. 110, No. 199, pp. 1–5, sep 2010.

[27] on{X}. `https://www.onx.ms/`.

発表文献

[1] 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.

[2] 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.

[3] 小貫貴央，神田尚子，放地宏佳，吉永努，入江英嗣．関連データ先読みとスマートフォンの消費電力に関する研究，第 10 回情報科学技術フォーラム（FIT2011），函館大学，2011 年 9 月 9 日

[4] 入江英嗣, 放地宏佳, 小木真人, 樫原裕大, 芝星帆, 眞島一貴．AirTarget: 光学シースルー方式 HMD とマーカレス画像認識による高可搬性実世界志向インターフェース, マルチメディア, 分散, 協調とモバイル (DICOMO2012) シンポジウム, pp.1295-1304, Jul. 2012.

[5] 入江英嗣，放地宏佳，稲場朋大，眞島一貴，藤原大輔，吉見真聡，吉永努．配線アクティビティを考慮した 3 次元積層プロセッサ向けフロアプランナー，情報処理学会論文誌コンピューティングシステム.【投稿中】

[6] 稲場朋大，放地宏佳，藤原大輔，眞島一貴，吉見真聡，入江英嗣，吉永努．配線アクティビティを考慮した 3 次元積層プロセッサ向けフロアプランナーのための熱評価手法，情報処理学会研究報告. 計算機アーキテクチャ研究会報告.

[7] Hirotaka Kashihara, Hiroki Shimizu, Hiroyoshi Houchi, Masato Yoshimi, Tsutomu Yoshinaga, Hidetsugu Irie. A Real-Time Gait Improvement Tool Using a Smartphone. Proc. of the 4th Augmented Human International Conference, 2013.