2ZF-4

複数の SNS を利用可能なアプリケーション開発のための データ入出力統合フレームワーク

岡崎　亮介†　　毛利　公美‡　　白石　善明†

† 名古屋工業大学　　‡ 岐阜大学

1.　はじめに

自然災害に対して，「防災」の取り組みだけでは被害を完全に防ぐことはできない．被害を最小限に留めるための「減災」の取り組みもまた重要である．減災のための取り組みのひとつに，災害が発生した際に適切な情報を提供するシステムによって被災者の行動を支援するものがある．

このような被災者を支援するシステムが SNS（Social Networking Service）を用いて提供されるとき，システムの利用者が普段使っているものと支援システムが用いる SNS とが異なる場合が出てくる．被災者支援システムを利用する機会は，事前の利用者情報登録時，および災害発生時と限られており，災害時には使い慣れていない状態でシステムを使うことになる利用者がいることになる．使い慣れない SNS を使っているシステムを利用者が使い難いと感じると，システムが利用されないことにつながる．したがって，システムは利用者にとって使い慣れた GUI で提供することが望ましい．

利用者が普段使っている SNS を模したクライアントアプリケーション（以下，クライアント）から，被災者情報が提供される SNS を操作することができれば，使い慣れた GUI でシステムを使えることになる．しかしながら，SNS は複数存在し，それぞれが異なるデータ形式で通信するため，各 SNS に対応したクライアントを開発するにはコストがかかる．

本稿では，クライアントの開発を支援するために SNS のデータ入出力を統合するフレームワークを提案する．フレームワークを用いて複数のクライアントを試作し，フレームワークの再利用性と拡張性について評価する．

2.　被災者支援システムと SNS

2011 年 3 月に発生した東日本大震災の際の情報伝達手段として，Twitter や Facebook などの SNS が活用されたことが報告されている．SNS は日常的に使われるサービスであったため，災害発生時のような場合でも積極的に使われたと考えられる．また，電話による通信が不通であった中でも，SNS によるやり取りは可能であり，メールや災害伝言板よりも早く情報を入手できた[1]ことからも，SNS は災害時の情報共有基盤として適しているといえる．

他方，消防庁による Twitter での災害情報タイムラインの提供[2]や，自治体のホームページを Facebook 上で運用する事例[3]など，公共機関が提供するシステムに SNS が用いられるようになってきている．SNS を用いることは，システムの構築・管理コストが軽減されることや，発信した情報が広がりやすいこと，既存の会員基盤を利用できることなどの利点があるため，公共機関が SNS を用いてシステムを構築する事例は増加してゆくことが考えられる．加えて，SNS の利用者は年々増加している[4]が，大手 SNS が利用者年齢の制限を緩和する動き[5]もあることから，今後幅広い世代に浸透することで，利用者数がさらに増加してゆくことも予想される．

以上から，被災者支援システムもまた，SNS を用いて提供される事例が増加すると考えられる．この場合に，使い慣れている SNS の GUI を模したクライアントによってシステムの利用につなげたいが，クライアントの開発は容易ではない．

The Data Input-and-Output Integrating Framework for the Development of Applications to Use Plural SNSs

† Ryosuke OKAZAKI and Yoshiaki SHIRAISHI・Nagoya Institute of Technology

‡ Masami MOHRI・Gifu University

3.　データ入出力統合フレームワーク

3.1　クライアントアプリケーション開発の課題

クライアントを開発するには，各 SNS がそれぞれ公開している API を呼び出さなければならない．SNS 運営事業者は，クライアント開発者に API を提供するとともに，API の仕様を Web サイトなどで公開しているが，その仕様は複雑であることに加え，API の数も多い．クライアント開発者は，API の仕様を把握した上でクライアントを実装せねばならず，複数の SNS に対応するには，多くの時間と作業が必要になる．また，API の仕様はしばしば変更されることがあり，その場合は改めて仕様を把握し直し，クライアントに反映させなければならない．複数の SNS の API を容易に呼び出すことができ，仕様変更の際にも柔軟に対応できるような枠組みが求められる．

3.2　フレームワークの動作

API の仕様を把握することは，SNS サーバへ入力するデータと，SNS サーバから出力されるデータを把握することである．入出力データが SNS ごとで異なることは，複数の SNS に対応したクライアントの実装を難しくしている．したがって，各 SNS のデータ入出力を共通化できれば，複数の SNS への対応が容易になり，クライアントの開発を支援することができる．

これを実現するフレームワークの動作は，以下のようになる．図 1 に示すように，クライアントと SNS サーバの間にデータのやり取りを仲介するインターフェースを挿入する．クライアントからのデータ入力は共通とし，インターフェースが各 SNS に合わせた形式に変換した後，API を呼び出し，SNS サーバへ入力する．SNS サーバからのデータ出力は，インターフェースが共通形式へと変換し，クライアントへ出力する．

4.　フレームワークの設計

4.1　設計方針

フレームワークは，再利用することを主眼に設計されたソフトウェア部品である．また，機能や処理の追加・修正・削除といった変更が容易であること，すなわち保守性が高い構成となっていることも特徴である．ライブラリと混同されることもあるが，上記の点において異なるものである．

再利用性と保守性の高いフレームワークを開発するには，適切な設計が求められる．そのためにはパターンの適用が不可欠となる[6]．1990 年代より様々なパターンが，提案・主張されてきたが，それらはすべて同じレベルの抽象度で考えられるものではない．ブッシュマンら[7]は，既存のパターンを抽象度レベルに応じて 3 つに分類している．抽象度の高い順に，アーキテクチャパターン，デザインパターン，イディオムである．本研究では，はじめにアーキテクチャパターン，その後デザインパ

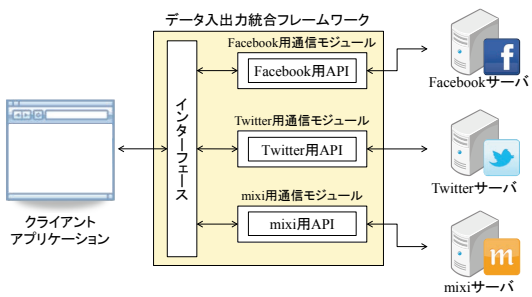


図 1　フレームワークの動作

ターンを適用し，段階的な具体化を図っている．

また，フレームワークを設計する際に重要となるのは，フレームワークの利用者が，どういった機能を実現したいという要求を持っているかを分析し，利用者が機能を実現しやすい構成とすることである．このような，機能の追加・変更が容易にできるように定めた箇所をホットスポットといい，それに対して今後，追加・変更しないと定めた箇所をフローズンスポットという．本研究が対象とするクライアントの開発には，自由な画面設計，および，被災者支援システム独自の機能追加ができるようにホットスポットを定めるべきだと考えた．

4.2 設計

4.1 節で述べた設計方針を踏まえ，設計を行った．

ソフトウェアの基礎的な構造を定めるアーキテクチャパターンには，MVC（Model-View-Controller）パターンを採用する．MVC パターンは，多くのフレームワークで用いられているパターンであり，処理（Model），入出力（View），入力に応じた処理の制御（Controller）を切り分けることで，M，V，C 間の依存性を下げることが可能になる．

デザインパターンは，アーキテクチャを構成するサブシステムが持つ，さらに小さなアーキテクチャである．本フレームワークでは，Observer パターンによって Model から View への通知を疎結合で実現し，Strategy パターンによって処理の切り替えを容易にしている．

フレームワークの構成を図 2 に示す．GUI を司る画面入力部と画面出力部，および，独自機能の追加を可能とするためデータ整形部をホットスポットとしており，クライアント開発者は，これらのみを実装すればよい構成となっている．

5. 実装と評価

5.1 フレームワークの実装

4 章で述べた設計をもとに，データ入出力統合フレームワークを実装した．開発言語には，マルチプラットフォームでの動作を考え，Java を用いた．対応する SNS として，本稿では Facebook，および Twitter を選択した．開発環境を表 1 に示す．

実装したフレームワークでの入出力統合の例を，以下に示す．

- ライブラリによる Twitter への投稿
  twitter.updateStatus(message);
- ライブラリによる Facebook への投稿
  parameter = Parameter.with("message", message);
  facebook.publish("me/feed", String.class, parameter);
- フレームワークによる投稿（*は SNS に応じたオブジェクト）
  *.sendMessage(message);

従来は Facebook と Twitter では個別にコードを記述しなくてはならないが，フレームワークを用いることで，いずれの SNS においても同一のコードで，各 SNS サーバとやり取りができる．

5.2 クライアントアプリケーションの試作

開発したフレームワークを用いて，クライアントを試作した．災害発生時には，携帯端末が多く利用されることを考え，Android 上で動作する Java クライアントを開発した．クライアントの開発環境を表 2 に，操作画面を図 3 に示す．Facebook の GUI を模したものと，Twitter の GUI を模したものの 2 種類を開発しており，どちらも Facebook と Twitter の両方を同時に操作可能である．

5.3 評価

2 種類のクライアントのフローズンスポットとホットスポットの比率の平均は表 3 のようになった．ホットスポットが 54.6%と，フローズンスポットよりも多くの箇所を占めている．これは，コード数が多くなりがちな GUI 部分がホットスポットであること，および，クライアント開発者が独自機能を追加・拡張できる設計となっているためであり，意図した結果となっている．クライアント開発者は本フレームワークを用いることでライブラリのみを用いては難しいカスタマイズを容易に行うことができるのである．

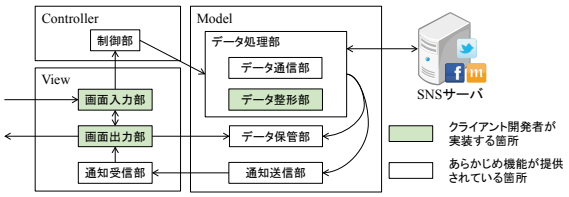


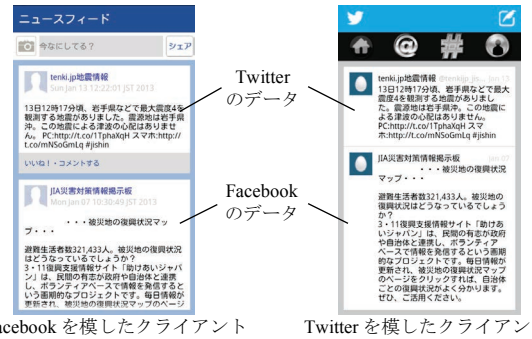
図 2 フレームワークの構成

表 1 フレームワークの開発環境

言語	Java (JDK 1.7.0_09)
ライブラリ	RestFB 1.6.11
	Twitter4J 3.0.3

表 2 クライアントアプリケーションの開発環境

言語	Java (JDK 1.7.0_09)
SDK	Android SDK rev. 21.0.1



Facebook を模したクライアント　　Twitter を模したクライアント

図 3 クライアントアプリケーションの操作画面

表 3 フローズンスポットとホットスポットの比率

フローズンスポット	45.4%
ホットスポット	54.6%

また，前述の通り，フローズンスポットは変更を加えることのないよう定めた箇所である．フローズンスポットに変更を加えることになった場合は，設計に不備があったということになる．今回のクライアント開発では，フローズンスポットを変更することなく開発できることを確認した．したがって，適切な設計がなされていたといえる．

6. おわりに

複数の SNS に対応したクライアントアプリケーションの開発は，SNS ごとに異なる API を把握しなければならない．本稿では，複数の SNS への対応を容易にするため，SNS のデータ入出力を統合するフレームワークを提案した．

パターンを用いて設計した後，フレームワークを実装し，その動作を確認するためクライアントを試作した．開発したクライアントのフローズンスポットとホットスポットについて計測し，再利用性と拡張性を備えていることを確認した．

参考文献

[1] ウェザーニューズ：「東日本大震災」調査結果，〈http://weathernews.com/ja/nc/press/2011/pdf/20110428_2.pdf〉（参照 2013-01-09）．
[2] 総務省消防庁：災害時におけるツイッターの活用を開始～「災害情報タイムライン」スタート～，〈http://www.fdma.go.jp/ugoki/h2207/2207_04.pdf〉（参照 2013-01-09）．
[3] 武雄市役所，〈http://www.facebook.com/takeocity〉（参照 2013-01-09）．
[4] 総務省：ICT インフラの進展が国民のライフスタイルや社会環境等に及ぼした影響と相互関係に関する調査研究，〈http://www.soumu.go.jp/johotsusintokei/linkdata/h23_06_houkoku.pdf〉（参照 2013-01-09）．
[5] Facebook Explores Giving Kids Access, THE WALL STREET JOURNAL, available from <http://online.wsj.com/article/SB10001424052702303506404577444711741019238.html> (accessed 2013-01-09).
[6] Ralph E. Johnson，中村宏明，中山裕子，吉田和樹：パターンとフレームワーク，共立出版，（1999）．
[7] F.ブッシュマン，R.ムニエ，H.ローネルト，P.ゾンメルラード，M.スタル：ソフトウェアアーキテクチャ，近代科学社，（2000）．