Vol.2014-IS-129 No.5
2014/9/11

処理部品の組み合わせでフローを設計可能なオープンデータマッシュアップツールの提案

北原 圭† 屋代 聡†

近年，公共機関が保有する様々な情報がオープンデータとして公開されている．オープンデータは CSV や XML 等の機械可読な形式で公開されている場合が多く，複数のデータを組み合わせたり，見やすく表示したりと言った使われ方をする．そのため統計スキルやプログラミングスキルを持たない者にとっては，オープンデータを十分に活用することが難しい状況にある．本論文では，処理部品を組み合わせてフローを設計することで，オープンデータのマッシュアップが容易に行えるツールを提案する．

Proposal of Flow-designable Mashup Tool for Processing Open Data

Kei Kitahara† Satoshi Yashiro†

In recent years, many kinds of open data have been released. Open data are released in a machine-readable format such as CSV and XML. They are usually combined with other open data and are used for visualisation of information, and thus people who are unfamiliar with statistics and programming cannot easily utilise open data. In this paper, we introduce a mashup tool in which people can easily design process-flow for open data by assembling components.

1. 背景と目的

クラウドコンピューティング，クラウドサービスの普及と共に，クラウド上には構造化された業務データばかりでなくセンサデータや映像データのような非構造データも蓄積され，多種・多様・大量のデータを利用し高度なサービスを生み出そうとするビッグデータ利活用が進んでいる．ビッグデータの利活用は，クラウド上に蓄積されたデータばかりでなく，企業や公共機関などの組織内に蓄積されたデータへも対象を広げており，クラウド上のデータと組織内のデータの掛け合わせにより，更に高度なサービスが生み出されると期待されており，いかに有用なデータを手に入れるか，いかに有用なデータの活用方法を生み出せるかが高度なサービスを実現する鍵となっている．

有用なデータの入手という観点では，公共機関の保有するデータを市民へ開放し自由に利用してもらうことで経済の活性化や社会課題の解決を後押しするオープンデータの推進が，日本を含め世界的な潮流となっている．オープンデータ推進政策で先行する米国や英国では，気象や土壌，作物の作付け・収穫実績に関するオープンデータを活用してリスク計算の精度を高め農家向け収穫保険を販売している Climate Corporation[1]の事例や，バスの運行情報・現在位置情報を地図と重ね合わせて観光客の旅程策定支援サービスを提供する BusIt London[2]などの活用事例が報告されている．国内では，2011 年 03 月に発生した東日本大震災の際，政府が保有する防災情報，被害状況情報，被災実績情報などが適切に公開されていれば，被害の波及をもっと小さくできたのではないかとの反省からオープンデータ推進に着手し，2012 年 07 月に「電子行政オープンデータ戦略」が公表されたのをきっかけに，2013 年 06 月に閣議決定された「世界最先端 IT 国家創造宣言」[3]では最重要課題として位置づけられている．

オープンデータとして公開されたデータは，編集・加工され，様々な用途に二次利用されることが期待されている．しかし，オープンデータを自由に編集・加工することができるのは主に IT スキルを持ったデベロッパに限定されてしまう．オープンデータ推進が成功するためには，オープンデータとして公開されたデータが IT スキルを持たないデベロッパを含め，より多くのデベロッパによって編集・加工され，より多くの用途に二次利用されることが重要である．

そこで，上記の背景を鑑み，本研究では IT スキルを持たないデベロッパであっても，オープンデータを容易に編集・加工することができるオープンデータマッシュアップツールを提案する．

2. 課題と解決策

IT スキルを持たないデベロッパがオープンデータの編集・加工等を行う際には，以下のような課題がある．

2.1 課題 1：プログラミングスキルが必要

データの形式は，CSV，XML，RDF 等，機械可読なフォーマットで提供される．様々なフォーマットで提供されるオープンデータを編集・加工するためのプログラムを記述しなければならず，オープンデータを処理するためにはプログラミングスキルが必要となる．

† (株) 日立製作所
Hitachi, Ltd.

2.2 課題 2：適切なデータ処理方法またはデータ処理方法の組合せが不明

オープンデータとして，様々な種類・フォーマットのデータが公開されている．オープンデータの種類・フォーマットに適したデータ処理方法やデータ処理方法の組合せを選択することは，IT スキルを持たないデベロッパにとって容易ではない．

3. 解決策

課題に対する解決策を下記のように示す．

3.1 課題 1 に対する解決策

プログラミングスキルを持たないデベロッパでもオープンデータの種類・フォーマットに適したデータ処理技術やデータ処理技術の組合せを選択でき，オープンデータを編集・加工できるようなオープンデータマッシュアップツールを構築する．

本ツールでは，インターネット上に存在するオープンデータを提供するサイト(以降，オープンデータポータルサイト)から任意のオープンデータを取得したり，取得したオープンデータを編集・加工したり，編集・加工した結果を可視化したりするなど，種々の処理を行う処理部品(以降，処理コンポーネント)を組み合わせ，処理フローを設計し，プログラミングレスでオープンデータの処理を可能とする．

また，本ツールは，グラフィカルに操作可能な GUI ベースのフロー設計インタフェースを提供することで，より簡単に処理フローを設計することを可能とする．

3.2 課題 2 に対する解決策

適切なデータ処理方法やデータ処理方法の組み合わせの選択を容易にするために，本ツールにて処理コンポーネント補完機能を提供する．本機能により，処理フローを設計していく中で，すでに配置されている処理コンポーネント間に配置可能な処理コンポーネントの候補を提示することが可能になり，より簡単に処理フローを設計することを可能とする．

4. オープンデータマッシュアップツール

本章では，オープンデータマッシュアップツールの構成や，デモシナリオを利用した本ツールによる処理フロー設計例について示す．

4.1 構成

図 1 は，本ツールの構成を示すものである．本ツールでは一般的な Web アプリケーションと同様，クライアント側で Web ブラウザを利用して Web サーバにアクセスし，オープンデータの処理を実行する．

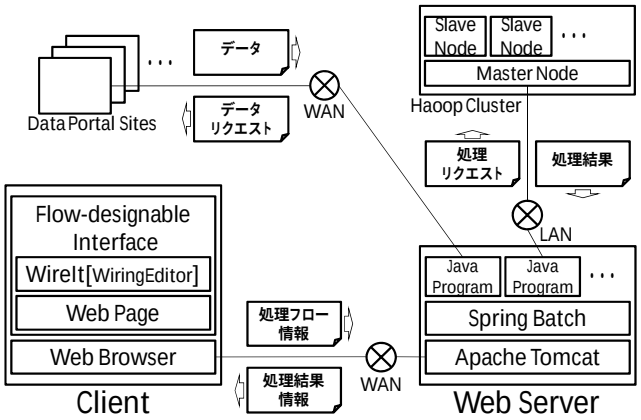

図 1 オープンデータマッシュアップツールの構成

Fig. 1 Configuration of Open Data Mashup Tool

デベロッパが操作するフロー設計インタフェースを図 2 に示す．本インタフェースは JavaScript[a] ライブラリの WireIt[4]のサンプルである WiringEditor[5]をベースにしている．

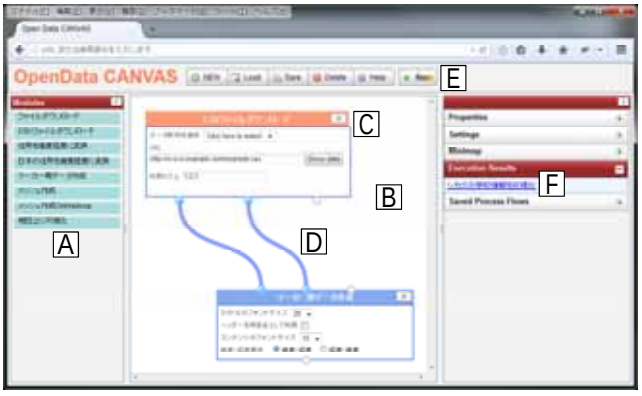

図 2 フロー設計インタフェース

Fig. 2 Flow-designable Interface

本インタフェースには，主に以下の領域やボタンがある．

- 処理コンポーネントアイテムリスト領域

処理コンポーネントアイテムリスト領域(A)には，オープンデータポータルサイトから任意のオープンデータを取得したり，取得したオープンデータを編集・加工したりする種々の処理に対応した処理コンポーネントアイテムが配置されている．

- 処理コンポーネント配置領域

処理コンポーネントアイテムリストのアイテムを右側の処理コンポーネント配置領域(B)にドラッグアンドドロ

a JavaScript は，Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です．

ップすると処理コンポーネント(C)が表示される．図 3 は処理コンポーネントの例である．処理コンポーネントには，テキストボックスやチェックボックスなどの引数入力部(C.a)があり，引数入力部の情報は引数として処理の中で利用される．処理コンポーネントの上部または下部にはターミナル(C.b)がある．上部のターミナルは入力ターミナルであり，処理対象となるデータを入力する．下部のターミナルは出力ターミナルであり，処理結果を出力する．処理コンポーネント同士は図 2 にあるように，ターミナルを接点として接続線(D)で接続することができる．

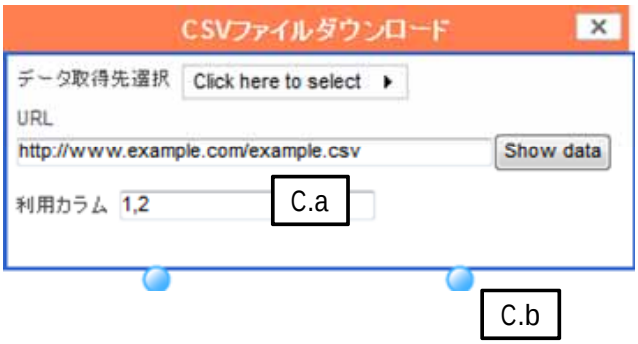

図 3 処理コンポーネント例

Fig. 3 Example of Process-component

- 処理実行ボタン

図 2 にあるように，本ツールの上部には処理実行(Run)ボタン(E)が配置されている．処理実行ボタンを押下すると，本インタフェース上で設計された処理フロー情報がサーバ側に送信される．

- 処理結果リスト領域

クライアント側から送信された処理フロー情報を基にサーバ側で実際の処理が行われる．処理終了後，図 2 にあるように，処理結果へのリンクが処理結果リスト領域(F)に表示される．

図 1 にあるように，Web サーバの機能はサーバ側の Apache Tomcat[b][6]が提供する．サーバ側には各処理コンポーネントに対応する Java[c] プログラムがそれぞれ用意されている．Java プログラムによっては，データポータルサイトからデータを取得したり，Hadoop[d][7]クラスタで大規模なデータを効率的に処理したりすることも可能である．

b Apache Tomcat および Tomcat は，Apache Software Foundation の商標です．
c Java は，Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です．
d Hadoop は，Apache Software Foundation の登録商標です．

サーバ側はクライアント側から送信された処理フロー情報を基に，処理フローを構成する処理コンポーネントの配置や接続状況を鑑み，各処理コンポーネントに対応する Java プログラムを実行する順序および各 Java プログラムの引数を決定し，Java プログラムの実行計画をバッチ処理用のフレームワークである Spring Batch[8]用の実行定義ファイルとして作成する．Spring Batch は実行定義ファイルの実行計画に沿って次々に Java プログラムを起動する．

最終的な処理結果へのリンクを処理結果情報としてクライアント側に返しサーバ側の処理が終了する．

4.2 デモシナリオによる処理フロー設計例

本シナリオでは，実際に公開されているオープンデータを利用した処理フローの設計例を示す．

本デモシナリオのアプトプットとして提供される情報は，シカゴ市内の学校に子供を入学させたいと考えている親が利用する．本デモシナリオを実施するデベロッパは，本デモシナリオのアウトプットの利用者と同一でもいいし，異なっていてもいい．

本デモシナリオのアウトプットを利用する親は，シカゴ市のどこにどのような学校があるかわからず，また，子供を入学させるのにより適した学校はどの学校かということも分からないとする．そこで，親が自分の子供をどの学校に入学させるか決定するときの判断材料となるような情報を分かりやすく提供する．具体的には以下の 2 点を実施する．

(1) シカゴ市にある学校の位置や学校に関する情報を地図上に表示．

シカゴ市のどこにどのような学校があるかどうかわからない親の為に，シカゴ市にある各学校の位置や学校についての基本的な情報を提供する．より分かりやすい形で提供するために，情報は地図上に表示する．

(2) 学校周辺の治安に関する情報を地図上に表示．

学校周辺の治安がいい(学校周辺で発生した犯罪の件数が少ない)方が親はより安心できる．そこで，なるべく周辺の治安がいい学校を探せるように，シカゴ市で発生した犯罪の件数についての情報をメッシュ状に区切り提供する．より分かりやすい形で提供するために，情報は地図上に表示する．

4.2.1 利用する処理コンポーネント

本項では，4.2(1)および 4.2(2)で利用する処理コンポーネントについて説明する．"入力"はターミナルから入力されるデータとする．"引数"は引数入力部に入力する値とする．"出力"は出力ターミナルから出力されるデータとする． "処理内容"は，処理コンポーネントが行う処理とする．

4.2 (1)では，以下の処理コンポーネントを利用する．

(A) 処理コンポーネント「CSV ファイルダウンロード」

入力： なし．

引数：データ取得先 URL，利用カラム．

出力：学校名，学校の住所，学校に関する情報．

処理内容：データ取得先 URL に入力された URL から，シカゴ市にある学校に関する CSV 形式のデータ[9]を取得し，引数で指定されたカラム(学校名，学校の住所，学校に関する情報)を出力する．

(B) 処理コンポーネント「住所を緯度経度に変換」

入力： 学校の住所．

出力：学校の緯度経度．

処理内容：学校の住所を緯度経度に変換し出力する．住所を緯度経度へ変換する機能はサーバが提供してもいいし，外部の API(e.g. Google Geocoding API[10])を利用してもいい．

(C) 処理コンポーネント「マーカー用データ作成」

入力：学校名，学校の緯度経度，学校に関する情報．

出力：マーカー用データ．

処理内容：学校名，学校の緯度経度，学校に関する情報から地図上に表示されるマーカー用のデータを作成し出力する．

4.2 (2)では，以下の処理コンポーネントを利用する．

(D) 処理コンポーネント「CSV ファイルダウンロード」

入力： なし．

引数：データ取得先 URL，利用カラム．

出力：犯罪が発生した経度緯度．

処理内容：データ取得先 URL に入力された URL から，シカゴ市で発生した犯罪に関する CSV 形式のデータ[11]を取得し，引数で指定されたカラム(犯罪が発生した経度緯度)を出力する．

(E) 処理コンポーネント「メッシュ作成 OnHadoop」

入力：犯罪が発生した緯度経度．

引数：メッシュサイズ，メッシュカラー．

出力：犯罪発生件数メッシュデータ．

処理内容：犯罪が発生した緯度経度から犯罪が発生した件数が多い場所・少ない場所をメッシュサイズ単位でメッシュカラーの濃淡で色分けした犯罪発生件数メッシュデータを作成し出力する．処理対象となる犯罪発生件数が 500 万件以上あるため Hadoop を利用して効率的に処理を行う．

4.2 (1)と 4.2(2)共通で，以下の処理コンポーネントを利用する．

(F) 処理コンポーネント「地図上に可視化」

入力：マーカー用データ，犯罪発生件数メッシュデータ．

出力：HTML ページ．

処理内容：学校の位置にマーカーを表示し，犯罪発生件数の多い場所・少ない場所をメッシュ上に区切り表示する地図を作成する．マーカーおよびメッシュを表示する地図には OpenStreetMap[e][12]を使用する．

4.2.2 フロー設計インタフェース上での操作と結果

4.2(1)および 4.2(2)における処理フローの設計手順を，図 4 を利用して説明する．

1. 処理コンポーネント「CSV ファイルダウンロード」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(A)を表示させる．シカゴ市にある学校に関するデータを取得するための URL 及び学校名，学校の住所，学校に関する情報に対応するカラム番号を引数として入力する．
2. 処理コンポーネント「住所を緯度経度に変換」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(B)を表示させる．
3. 処理コンポーネント「マーカー用データ作成」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(C)を表示させる．
4. 処理コンポーネント「CSV ファイルダウンロード」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(D)を表示させる．シカゴ市の犯罪に関するデータを取得するための URL 及び犯罪が発生した緯度経度に対応するカラム番号を引数として入力する．
5. 処理コンポーネント「メッシュ作成 OnHadoop」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(E)を表示させる．メッシュサイズとメッシュカラーを引数として入力する．
6. 処理コンポーネント「地図上に可視化」に対応するアイテムをアイテムリストから配置領域へドラッグアンドドロップし，(F)を表示させる．
7. 処理コンポーネント A の出力(学校名)と処理コンポーネント(C)の入力(学校名)，処理コンポーネント(A)の出力(学校の住所)と処理コンポーネント(B)の入力(学校の住所)，処理コンポーネント(A)の出力(学校に関する情報)と処理コンポーネント C)の入力(学校に関する情報)，処理コンポーネント(B)の出力(学校の緯度緯度)と処理コンポーネント(C)の入力(学校の緯度経度)，処理コンポーネント(C)の出力(マーカー用データ)と処理コンポーネント(F)の入力(マーカー用データ)，処理コンポーネント(D)の出力(犯罪が発生した緯度経度)と処理コンポーネント(E)の入力(犯罪が発生した緯度経度)，処理コンポーネント(E)の出力(犯罪発生件数メッシュデータ)と処理コンポーネント(F)の入力(犯罪発生件数メッシュデータ)，とを接続線で接続する．

e OpenStreetMap は，OpenStreetMap Foundation の欧州連合及びその他の国における商標です．

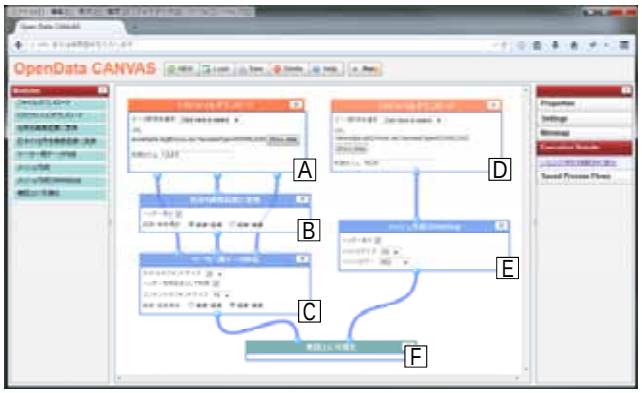

図 4 処理フロー例

Fig. 4 Example of Process-flow

8. 処理実行ボタンを押下することにより，サーバ側に処理フロー情報が送信され，処理フローに沿って処理が実行され，処理結果へのリンクが処理結果情報としてクライアント側に返される．クライアント側では，処理結果リストに表示された処理結果へのリンクを押下することにより，処理結果を別ウインドウに表示する(図 5)．

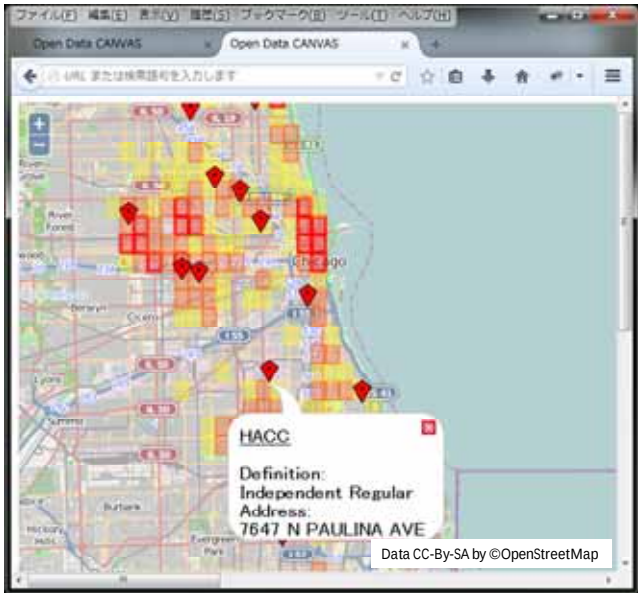

図 5 処理結果

Fig. 5 Process Result

処理結果では，学校がある位置にマーカーが表示されており，マーカーを押下すると，学校名や学校に関する情報を見ることができる．また，犯罪発生件数が多い場所が赤く，少ない場所が無色，中間の場所が黄色く色分けされたメッシュにより，学校周辺の治安の良し悪しを知ることができる．

本デモシナリオのアウトプットであるシカゴ市にある学校の位置や学校に関する情報および学校周辺の治安に関する情報は，親が自分の子供をどの学校に入学させるか決定するときの判断材料として利用することができる．

5. 処理コンポーネント補完機能

オープンデータマッシュアップツールを利用することで，処理コンポーネントを組み合わせてデータを処理することが可能となった．しかし数ある組み合わせの中から適切な組み合わせを適宜選択しなければならず，利用可能な処理コンポーネントの数によっては処理コンポーネントの組合せのパターンが増大してしまうため、適切な処理コンポーネントを選択することは容易ではない．そこで，すでに配置されている処理コンポーネント間に配置可能な処理コンポーネントの候補を提示する機能を提供し，処理フローの設計をより簡単にする．

5.1 処理コンポーネント補完機能の動作原理

本節では，処理コンポーネント補完機能の動作原理について示す．

図 6 では，処理コンポーネント配置領域にすでに処理コンポーネント「CSV ファイルダウンロード」と「マーカー用データ作成」が配置されている状態にある．

ここで，二つの処理コンポーネント間を右クリックすることによって処理コンポーネント候補のリスト(A)を表示し，上下に配置された処理コンポーネント間に配置可能な処理コンポーネントの候補(B)を提示する．

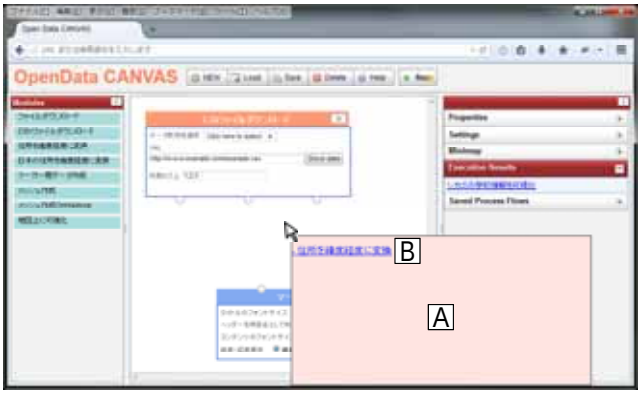

図 6 処理コンポーネント候補のリスト

Fig. 6 List of process-component candidates

処理コンポーネント候補を選択することで，処理コンポーネント(C)を処理コンポーネント配置領域に配置することができる(図 7).

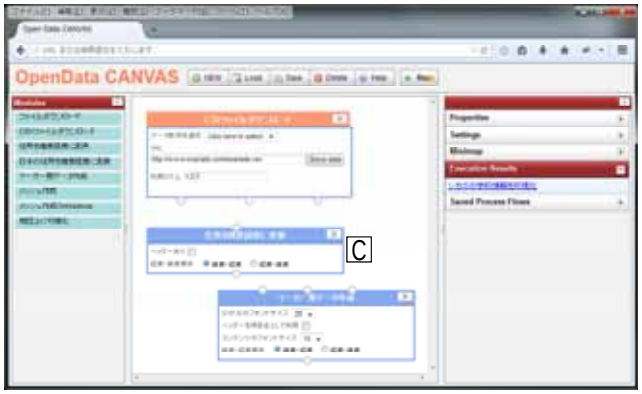

図 7 配置された処理コンポーネント候補
Fig. 7 Placed Process-component Candidate

図 8 を例に，処理コンポーネントの候補を提示する際の条件を示す．

処理コンポーネント配置領域には 2 つの処理コンポーネントが配置され，任意の位置にマウスカーソルが配置されている．

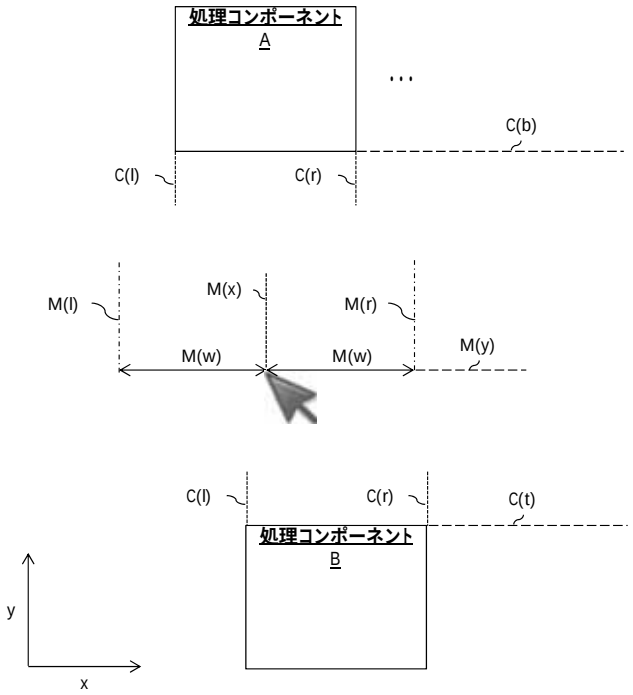

図 8 処理コンポーネント補完機能における条件
Fig. 8 Conditions for Process-component-suggesting Function

処理コンポーネントの幅や処理コンポーネント配置領域の広さ等から以下の情報を予め決定しておく．

- M(w): x 方向の任意の距離

右クリックをした瞬間に，すでに配置されている処理コンポーネント及びマウスカーソルに関する以下の情報を取得する．

- C(l): 各処理コンポーネントの左側の x 座標
- C(r): 各処理コンポーネントの右側の x 座標
- C(t): 各処理コンポーネントの上側の y 座標
- C(b): 各処理コンポーネントの下側の y 座標
- M(x): マウスカーソルの x 座標
- M(y): マウスカーソルの y 座標
- M(l): M(x)から負の方向に M(w)移動した x 座標
- M(r): M(x)から正の方向に M(w)移動した x 座標

上記の情報が，以下の条件 R を満たし，かつ，それぞれ処理コンポーネントが条件 R1～R4 のいずれかを満たす場合に，すでに配置されている処理コンポーネント間に配置可能な処理コンポーネントの候補を提示するものとする．

- 条件 R: M(y)は上方にある各処理コンポーネントの C(b)よりも小さく，下方にある各処理コンポーネントの C(t)よりも大きい．
- 条件 R1: M(l)は C(l)よりも大きく，C(r)よりも小さい．
- 条件 R2: M(r)は C(l)よりも大きく，C(r)よりも小さい．
- 条件 R3: M(l)は C(l)よりも小さく，M(r)は C(r)よりも大きい．
- 条件 R4: M(l)は C(l)よりも大きく，M(r)は C(r)よりも小さい．

5.2 処理コンポーネント候補のより適切な順序での提示

処理コンポーネント候補リストに表示する処理コンポーネント候補が複数ある場合，どの処理コンポーネント候補がより適切かということを適宜判断する必要がある．

そこで，過去に実際に配置された回数の多い処理コンポーネントの組み合わせはより適切であるということを前提とし，過去の配置履歴情報を利用して処理コンポーネント候補の順位付けを行い，より適切な処理コンポーネントを容易に選択できるようにする．

処理コンポーネントの配置履歴情報を利用して処理コンポーネント候補の順位付けを行うために，処理コンポーネントテーブル，入力処理コンポーネントテーブル，出力処理コンポーネントテーブル，処理コンポーネント候補テーブル，候補リストテーブル，に記憶されている情報を利用する．

上記の各テーブルはサーバ側の DB に作成される．クライアント側から処理フロー情報が送信される度，処理フローを構成する処理コンポーネントの組み合わせが DB に登録されていない場合には処理コンポーネントの組み合わせを表すレコードを追加し，既に処理コンポーネントの組み合わせが登録されている場合には，当該の処理コンポーネントの組み合わせが配置された回数を 1 増加する．

それぞれのテーブルの関係は図 9 の ER ダイアグラムのとおりである．

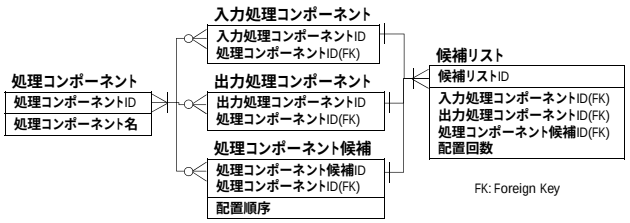

図 9 ER ダイアグラム
Fig. 9 ER Diagram

以下に，各テーブルに記憶される項目などについて説明する．

処理コンポーネントテーブルは, 処理コンポーネント ID と処理コンポーネント名を記憶している．

入力処理コンポーネントテーブルは，入力処理コンポーネント ID と，処理コンポーネント ID を記憶している．本テーブルは，処理コンポーネント候補に処理結果を渡す処理コンポーネントまたは処理コンポーネントの組み合わせを表している．

出力処理コンポーネントテーブルは，出力処理コンポーネント ID と処理コンポーネント ID を記憶している．本テーブルは，処理コンポーネント候補の処理結果を受け取る処理コンポーネントまたは処理コンポーネントの組み合わせを表している．

処理コンポーネント候補テーブルは処理コンポーネント候補 ID と処理コンポーネント ID と配置順序を記憶している．本テーブルは，処理コンポーネント候補または処理コンポーネント候補の組み合わせと処理コンポーネント候補の配置順序を表している．処理コンポーネント候補が複数の処理コンポーネントの組み合わせである場合，配置順序の数が小さい処理コンポーネント候補がより上方の位置に配置される．

処理コンポーネント候補リストテーブルは，処理コンポーネント候補リスト ID，入力処理コンポーネント ID，出力処理コンポーネント ID，処理コンポーネント候補 ID，配置回数を記憶している．配置回数は，クライアント側から処理フロー情報が送信された際に，処理フローを構成する処理コンポーネントの組み合わせが既に DB に登録されている場合には 1 増加する. 本テーブルを利用することで，入力処理コンポーネント ID で紐付けられた入力処理コンポーネントと出力処理コンポーネント ID で紐付けられた出力処理コンポーネントを基に，処理コンポーネント候補 ID で紐付けられた処理コンポーネント候補を取得することが可能となる．複数の処理コンポーネント候補が取得できる場合には，それぞれの配置回数を比べ，配置回数が多い処理コンポーネント候補がより適切であると判断される．

6. 評価

オープンデータマッシュアップツールのフロー設計インタフェースを利用することにより，処理フローをグラフィカルに設計することが可能となる．

そこで，本章では，フロー設計インタフェースを利用した処理フローの設計時間と，フロー設計インタフェースを利用しない処理フローの設計時間とを比較し，本ツールで利用するフロー設計インタフェースの効果について検討する．

処理フローは，4.2 に記載のシナリオを 4.2.2 に記載の手順で設計する．

フロー設計インタフェースを利用する場合，フロー設計インタフェース上で設計された処理フローは Spring Batch 用の実行定義ファイルに自動的に変換されるものとし，フロー設計インタフェースを利用しない場合には，実行定義ファイルをテキストエディタ等で作成するものとする．

6.1 フロー設計インタフェースを利用した処理フローの設計時間

フロー設計インタフェースを利用して，4.2 に記載のシナリオを 4.2.2 に記載の手順で処理フローを設計した場合，処理コンポーネント配置領域には，合計 6 つの処理コンポーネントを配置し，処理コンポーネント同士を合計 7 本の接続線で接続する．

処理コンポーネント補完機能を利用することを考慮し，引数入力部への値の入力も含め処理コンポーネントを 1 つ配置するのに 20 秒要すると仮定すると, 処理コンポーネントを全て配置するには 120 秒要することになる．

処理コンポーネント同士を接続線で接続するのに 1 本あたり 10 秒要すると仮定すると, 接続線で全て接続するには 70 秒要することになる．

全ての処理コンポーネントを配置し，全ての処理コンポーネント同士を接続線で接続するには，190 秒(120 秒 + 70 秒)要することになる．

6.2 フロー設計インタフェースを利用しない処理フローの設計時間

フロー設計インタフェースを利用せずに，6.1 で作成される実行定義ファイルと同じファイルをテキストエディタ等で作成する．

実行定義ファイルは形式が決まっており，ファイルを 1 から作成するのではなく，ある程度共通化できる部分は予め記述してあるものを利用することとする．

実行定義ファイルには，バッチ処理全体に関わる部分と各処理に関わる部分とに分かれている．バッチ処理全体に関わる部分では，バッチ処理名やリスタート可・不可の設定等を記述する．各処理に関わる部分では，処理に利用する Java プログラムのクラス名，処理に必要な引数，エラー時の再試行回数の設定等を記述する．

バッチ処理全体に関わる部分は予め記述してあるものと仮定し，数ある処理の中から適切な処理を選択したり，テキストエディタを利用して処理を記述したりすることを考慮し，各処理に関わる部分の記述には 1 処理あたり 120 秒要すると仮定すると，6 つ全ての処理に関わる部分を記述するには 720 秒要することになる．

上記の結果から，本ツールのフロー設計インタフェースを利用して処理フローを設計した場合には，フロー設計インタフェースを利用せずに処理フローを設計した場合に比べ設計時間を約 73.6%短縮すると仮定する．尚，利用可能な処理コンポーネントの数が多いほど，処理コンポーネント補完機能の効果が高まり，フロー設計インタフェースを利用した場合と利用しない場合との処理フロー設計時間の差が大きくなると考えられる．

7. 関連技術

(1) Talend Open Studio[13]

上記は，データの入力，加工，出力に関する処理の流れをグラフィカルに設計し，実際に処理を実行することが可能である．しかし，加工後のデータをどのようにして表示するかという可視化部分についての処理を別途異なる環境や製品で実施する必要がある．

一方，本研究で提案しているツールでは，データの可視化部分も含めて設計することができ，処理結果を地図上等に容易に可視化することが可能である．

(2) Google Fusion Tables[14]

上記は，CSV 形式のデータを WEB 上で操作でき，グラフやチャートとして表示できる．特に，データを地図上に可視化する場合には， Google マップ[15]等，Google Fusion Tables を提供する Google 社に関する技術を利用できるため，スムーズかつ容易に実施することが可能である．しかし，テーブル形式のデータの簡易的な操作は容易にできるが，より複雑な処理を実行することが難しく，また，予め用意されていない方法やスタイルでデータを可視化することが難しい．

一方，本研究で提案しているツールでは，処理コンポーネントを用意しさえすれば複雑な処理を実行することが可能である．また，様々な可視化用の処理コンポーネントを追加することにより，自由な方法・スタイルでデータを表示することが可能である．

8. おわりに

本報告では，ある程度の仮定を基に評価を行ったため，今後，定量的な有用性等の評価を行う必要がある．

参考文献

1 Climate Corporation 社, Total Weather Insurance, http://climate.com/products/total-weather-insurance
2 Chris Haynes, BusIt London, http://www.busitlondon.co.uk/
3 高度情報通信ネットワーク社会推進戦略本部,世界最先端 IT 国家創造宣言, http://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20130614/siryou1.pdf
4 Eric Abouaf, WireIt, http://dev.lshift.net/james/wireit/wireit/
5 Eric Abouaf, WiringEditor, http://dev.lshift.net/james/wireit/wireit/examples/WiringEditor/
6 Apache Software Foundation, Apache Tomcat, http://tomcat.apache.org/
7 Apache Software Foundation, Apache Hadoop, http://hadoop.apache.org/
8 Pivotal Software 社, Spring Batch, http://docs.spring.io/spring-batch/
9 City of Chicago Data Portal, Schools, https://data.cityofchicago.org/api/views/kqmn-byj8/rows.csv?accessType=DOWNLOAD
10 Google 社, Google Maps API ウェブサービス, https://developers.google.com/maps/documentation/geocoding/?hl=ja
11 City of Chicago Data Portal, Crimes - 2001 to present, https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD
12 OpenStreetMap Foundation, OpenStreetMap, http://www.openstreetmap.org/#map=5/35.588/134.380
13 Talend 社, Talend Open Studio, http://jp.talend.com/products/talend-open-studio
14 Google 社, Google Fusion tables, http://www.google.com/drive/apps.html#fusiontables
15 Google 社, Google Maps, https://maps.google.co.jp/