データフロービジュアル言語を用いた情報可視化システムの開発環境

伊藤 隆朗[1,a)]　三末 和男[2,b)]　田中 二郎[2,c)]

概要：視覚的表現を効果的に活用するには，データや利用目的に応じて可視化手法を設計する必要がある．この方法としては，テンプレートから選ぶ方式や，テキストベースのプログラミングよって記述する方法があるが，両者は習得の容易さと表現力においてトレードオフの関係にある．そこで本研究では，多彩な可視化手法を容易に記述できるようにすることを目指し，データフロービジュアル言語を備えた情報可視化システムの開発環境を構築した．この環境では，情報可視化システムの開発者に対してデータ変換の様子を積極的に提示することで，可視化手法の記述と把握を容易にした．データフロービジュアル言語には，可視化処理の構成要素に基づいたビルディングブロックのほか，演算ブロックも用意することで，多彩な可視化手法の記述を行えるようにした．さらに，開発者がデータと視覚要素のブロックを指定した場合に，両方のブロックの間にデータ変換のブロックを自動的に挿入する機能を備えることで，データと視覚要素の関連付けを直感的に行えるようにした．構築した開発環境の評価として，いくつかの可視化手法を題材に，開発に要する操作量と表現力の高さについて確認を行った．

1. はじめに

情報可視化分野では様々な可視化手法が開発されており，これらは情報を人にわかり易く提示する手段として有用である．現在でも分析ツールや Web サイトで利用されており，今後もシステムに情報可視化を取り入れたいという需要は増えていくと考えている．このような情報可視化を取り入れたシステムのことを情報可視化システムと呼ぶ．

情報可視化を効果的に利用するためには，利用目的やデータに応じた適切な可視化手法の選択や設計を行い，それを記述する必要がある．記述の方法として，テンプレートから選ぶ方式やテキストベースのプログラミングよって記述する方法がある．テンプレート方式では，手軽に可視化手法を選択できるものの，限られた手法しか利用することができない．一方，プログラミングは高い表現力を持ち，様々な可視化手法を記述することができる．この記述を支援するためのツールキット [1][2][3][4] も開発されている．しかし，テキストによる記述は，データと視覚要素の結びつきを理解しづらいと言われている [5]．また，最終的な可視化結果の調整のために，コードの記述と実行結果の確認を何度も繰り返す必要があり，位置や配色に関わるパラメータの調整は扱いづらいものである [6]．

テンプレート方式とプログラミングとの間を埋めるため，ビジュアルインタフェースによる可視化手法の記述方法が研究されている．Lyra[5] や iVisDesigner[6] は，テキストでは記述がしづらいタスクを支援するため，ドローツール風のインタフェースによって可視化手法を記述するツールである．視覚要素の位置や色などを直接編集することを可能としている．また，データ，データ変換，視覚要素などの情報可視化の構成要素 [7] に基づいたビルディングブロック（以下，ブロックと呼ぶ）が用意されている．ツールの利用者は，これらのブロックを組み合わせることで基礎的な可視化手法（棒グラフ，散布図）を容易に記述することができ，その手法に対して，例えば棒グラフの棒にデータに応じた色をつけるなどのカスタマイズを行うことができる．テンプレート方式と比べ高い表現力を持ち，プログラミングに比べて容易に多彩な可視化手法を記述することができる．

先行研究は，テンプレート方式とプログラミングの間を埋める方法として有用であるものの，以下の 2 つの問題点があると考えている．

Lyra や iVisDesigner を用いて多彩な可視化手法を記述できるようになるためには，これらのツールが持つ内部フレームワークの概念を理解し，用意された多くのインタ

[1] 筑波大学大学院 コンピュータサイエンス専攻
[2] 筑波大学 システム情報系
[a)] ito@iplab.cs.tsukuba.ac.jp
[b)] misue@cs.tsukuba.ac.jp
[c)] jiro@cs.tsukuba.ac.jp

フェースが内部フレームワークにどのように作用するかを理解する必要がある．例えば，Lyra では，矩形に対して 2 つのデータをドラッグ・アンド・ドロップすることで簡単に棒グラフを作成することができる．この操作によって，システムは自動的に 2 つのデータ変換のためのブロックと 2 つの軸（X 軸，Y 軸）を生成し，それぞれを関連付け，さらに矩形の X 座標，Y 座標，高さを関連付ける．積み上げ棒グラフを記述することも可能であるが，先のデータ変換のためのブロックや軸，矩形の概念とそれを関連付けるための別のインタフェースを学習する必要がある．iVis Designer においても同様である．このように，基礎的な可視化手法を制作できるようになった段階から，それをカスタマイズし，さらに独自の可視化手法を記述できるようになるためには依然隔たりがあると考えている．

Lyra や iVisDesigner は，ツールに用意されたデータ変換と視覚要素のブロックを組み合わせることで多彩な可視化手法を記述可能にしている．そのため，ツールの概念と機能を十分に理解したとしても，記述可能な可視化手法はツールのサポートする機能に限られる．ツールのサポートしていない可視化手法を記述するためには，プログラミングによって一から記述する必要がある．しかし，Lyra や iVisDesigner では容易に記述できた多くのことをテキストで記述しなくてはならない．

そこで本研究では，先行研究の利点を取り入れつつ，上記 2 点を解消した情報可視化システムのための開発環境を開発する．その要件として以下を定めた．

(1) データと視覚要素を直感的に関連付けられること
(2) 視覚要素のパラメータを直感的に編集できること
(3) ツール内のデータ変換の様子を確認できること
(4) ブロックを同じインタフェースで連携させられること
(5) データ変換方法を記述できること

要件 (1) (2) は，先行研究の大きな利点であるため要件として設定した．要件 (3) (4) は，基礎的な可視化手法を制作できるようになった段階から多彩な可視化手法を記述できるようになるまでの隔たりを埋めるために設定した．ツール内のデータ変換の様子を積極的に利用者に提示することで，内部フレームワークの理解やインタフェースの学習を促すことができると考えられる．また，ブロック間の連携インタフェースを統一することで，インタフェースの学習コストを軽減できると考えられる．要件 (5) は，多彩な可視化手法を記述できるようにするためには，データ変換方法の記述を行える必要があると考え設定した．

これらの要件を満たすため，本研究ではデータフロービジュアル言語により可視化手法の記述が行える開発環境 Iv Studio を開発した．先行研究は，ドローツールのインタフェースを基本とし，それに加えデータを割り当てられるようにする設計がなされている．これに対し本研究は，表現力が高くデータフローの記述に適したデータフロービジュアル言語 [8] を基本とし，その記述を支援するためにドローツール風のインタフェースを加えた．これにより，利用者が手軽なドローツールのインタフェースから入り，多彩な可視化手法の記述へと順にステップアップできると考えた．

データフロービジュアル言語には，データ，データ変換，視覚要素 3 種類のブロックのほか，四則演算や条件分岐のブロックを用意し，データ変換の記述も可能とした．加えて，ブロック自体をスクリプト言語によって拡張可能とした（要件 (5)）．

可視化手法の記述方法としては，ブロック間をドラッグ操作でつなぐインタフェースに統一した（要件 (4)）．また，利用者がデータと視覚要素のブロックを指定した場合，両ブロック間にデータ変換のブロックを自動的に挿入することで，データと視覚要素の関連付けを行えるようにした（要件 (1)）．この際に，アニメーションを用いて挿入する様子を提示することで，利用者に内部動作を提示するよう配慮した（要件 (3)）．そして，記述した可視化手法を実行して結果を表示し，それらを直接操作することで，視覚的なプロパティ（位置，大きさ，色）などを編集することができるようにした（要件 (2)）．

Iv Studio を用いて設計した可視化手法は，開発環境上や専用の実行環境にて実行することができる．また，Lua[*1] のソースコード出力機能を備えており，既存のデータ処理システムなどに組み込むことができる．

2. 関連研究

可視化手法の記述方法として，ツールによる記述とプログラミングによる記述がある．本章では，この 2 つの手法に関して述べたのち，我々の採用したデータフロービジュアル言語に関する関連研究を述べる．

2.1 可視化手法記述のためのツール

複数の可視化手法を連携させたビューを制作するツールとして，FLINA[9] や Improvise[10]，Snap-Together Visualization[11] が挙げられる．FLINA は，キャンバス上に軸を描くことで，散布図と PCP を柔軟に組み合わせた可視化結果を制作することが出来る．Improvise や Snap-Together Visualization は，用意された可視化手法を互いに連携させることのできるインタフェースを提供することで，データをドリルダウンしながら分析していくことのできるシステムの構築を可能としている．本研究では，予め用意された可視化手法や連携方法ではなく，より多彩な可視化手法を記述可能な環境を目指している．

可視化手法の記述のためのシステムも提案されている．iVis Designer[5] や Lyra[6] は，キャンバス上に図形を描く

[*1] http://www.lua.org/

ことで，ビジュアルマッピングの手順を設計することができる．これらのツールはドローツール風のインタフェースを備えており，非プログラマでも利用することができるうえ，情報可視化の広い範囲をカバーする設計を可能としている．本研究もこれらのインタフェースを取り入れている．しかし，可視化手法間の連携のような複雑なデータ変換の流れを記述するためには，その流れ自体を開発者に提示する方が有用であると考え，データフロービジュアル言語を採用した．

2.2 情報可視化向けのツールキット

情報可視化システムの利用者の操作に応じてインタラクティブに可視化結果が変更するようなシステムの開発には，プログラミングが用いられている．これを支援するため，情報可視化向けのツールキットが開発されている．Processing[*2] はプログラミング初心者に向けたプログラミング環境であり，短いコードで可視化システムを作成することができる．InfoVis toolkit[1] は，典型的な可視化手法と可視化システムでよく使われる GUI コントロールを提供する．Prefuse[2] と Flare[*3] は，可視化手法を構築するための粒度の細かいブロックを提供する．Protovis[3] は，矩形や線，ラベルといったシンプルな視覚的要素を提供する．このライブラリは，抽象度の高い可視化向けのツールキットと，抽象度の低い描画 API との間のギャップを埋めるために開発された．D3[4] は，Protovis のコンセプトに加え，更に開発環境や将来的な CSS への互換性を持たせたツールキットである．

我々はデータフロービジュアル言語を設計するに当たり，これらのツールキットの設計を参考にしている．一方で，要件 (1) (2) の観点からテキストによる可視化手法のデータ変換の手順の記述ではなく，データフロービジュアル言語による記述方法を採用している点でこれらの研究とは異なる．

2.3 データフロービジュアル言語

Haeberli ら [12] は，動的に変化するグラフィックアプリケーションを構築するためのビジュアルプログラミング言語を開発した．Bencina[13] はデータフロービジュアル言語を用いてオーディオプロセッシングシステムを構築する AudioMulch を開発した．

可視化を用いてデータ変換を記述するシステムも存在する．Hansaki ら [14] は，データフロー図を描くことで SQL クエリを生成するシステムを開発した．Zgraggen ら [15] は，無限ホワイトボード上で可視化結果を連携させていくことでデータのブーリアン演算を行いながらデータ分析を可能とするシステム PanoramicData を開発した．

サイエンティフィックビジュアリゼーションの分野では，古くからデータフロービジュアル言語が用いられている．例えば，DataVis[16]，AVS[17]，GeoVISTA Studio[*4]，SRIRun[18] が挙げられる．これらの環境は，ノードリンクダイアグラム形式の図を描くことでデータから要素への変換方法を記述する方法を採用しており，サイエンティフィックビジュアリゼーションでよく用いられる可視化手法のためのデータ変換を行うことに注力している．これに対し本環境は，非空間データを扱う情報可視化を対象としており，データテーブルの集計や集約，データの座標空間への変換といった情報可視化に必要な機能をサポートし，可視化手法自体の記述も可能としている．

3. Iv Studio

Iv Studio は，情報可視化システムの開発環境である．利用者は，システムの開発で主要なタスクである可視化手法の設計を，データフロービジュアル言語とドローツール風インタフェースを用いてインタラクティブに行うことができる．設計した可視化手法を，専用の実行環境にて実行できるほか，Lua のソースコードとして出力することが可能であり，別のソフトウェアに組み込んで使用することもできる．本章では，まず開発したインタフェースの利用例を交えながら使用方法を説明した後，インタフェース設計について説明する．

3.1 散布図と棒グラフの記述

Iv Studio を用いた散布図の記述方法を紹介する．Iv Studio を起動すると，図 1(a) の画面が表示される．まず，ここにテーブルのファイル（CSV や TSV 形式）をドラッグ・アンド・ドロップすることでデータを読み込ませ，データテーブルを示す “ノード” を生成する (図 1(b))．ここで読み込んだデータは，3 つの数値データのカラム（C0〜C2）と一つのテキストデータのカラム（C3）を持つ，3 レコードのテーブルである．次に，画面上部のメニューから Circle を選択し，円を描くノードを生成する (図 1(c))．そして，データテーブルのノードにある “出力ピン” と円のノードの “入力ピン” をドラッグ操作でつなぐことで，データを視覚変数に割り当てる (図 1(d))．ここでは，データカラム C0 のデータを円の X 座標に割り当てている（出力ピン C0 を入力ピン X につないでいる）．データを割り当てると，開発環境によってデータ変換に必要なノードが自動的に間に挿入され，その様子がアニメーションによって利用者に提示される (図 1(e))．ここでは，データカラム C0 の数値データを，座標軸を介して座標値に変換するノードが自動的に挿入されている．記述したデータフローは即座に実行され，プレビューキャンバス上に表示される．同じ手順で，

[*2] https://processing.org/
[*3] http://flare.prefuse.org/
[*4] http://www.geovistastudio.psu.edu/jsp/

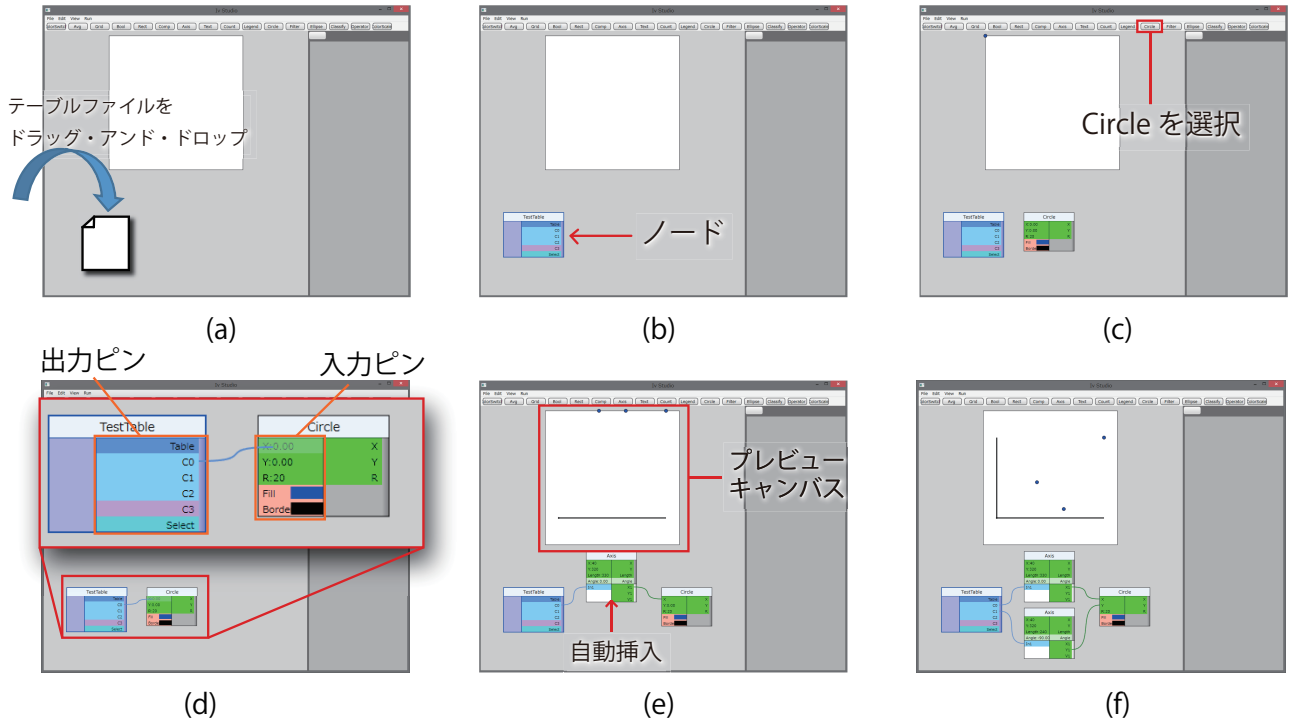

図 1　Iv Studio を用いた散布図の制作手順

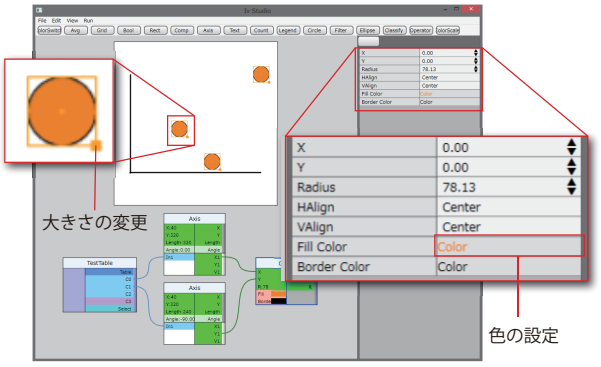

図 2　ドローツール風インタフェースによる見た目の調整

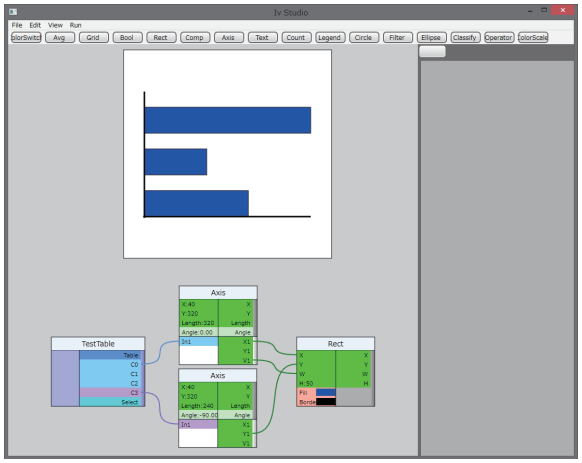

図 3　棒グラフを制作した例

データカラム C1 を円の Y 座標に割り当てることで，散布図を記述することができる (図 1(f))．

各ノードは，そのノードの動作を決定するプロパティを持っている．ノード自体かノードの生成した図形をクリックすることでノードが選択され，プロパティ一覧が画面右のプロパティパネルに表示される（図 2）．ここでは，利用者が，円をクリックして選択し，プロパティパネルから円の色を設定（青からオレンジに変更）し，図形に表示されるハンドルを使って大きさを調整した．（ノードの詳細については，4.1 節にて詳しく述べる．）

次に棒グラフの記述方法を紹介する (図 3)．まず，散布図同様データを読み込み，画面上部のメニューから Rect を選択し，矩形を表すノードを生成する．棒グラフは，矩形の幅でデータ値を表す可視化手法である．そのため，データテーブルのノードの C0 ピンを矩形の幅を表す “W ピン” につなぐ．すると，システムによって横軸が自動生成され，矩形の X 座標と幅に適切な値が割り当てられる．次に，矩形を縦方向に並べるために C3 を Y 座標に割り当てる．C3 はテキストデータのカラムのため，システムによってレコードを順に並べる軸が自動的に生成される．

3.2　棒グラフと散布図との連携

Iv Studio で開発した情報可視化システムは，実行時にシステムの利用者が図形を選択することで関連付けられているデータレコードを選択できるようにしている．システム開発者はデータレコードの選択状態をデータテーブルのノードの “Select ピン” から取得することができる．この機能を利用して，ハイライト表示を記述することができる．

図 4 は，散布図か棒グラフの視覚要素を選択すると，互

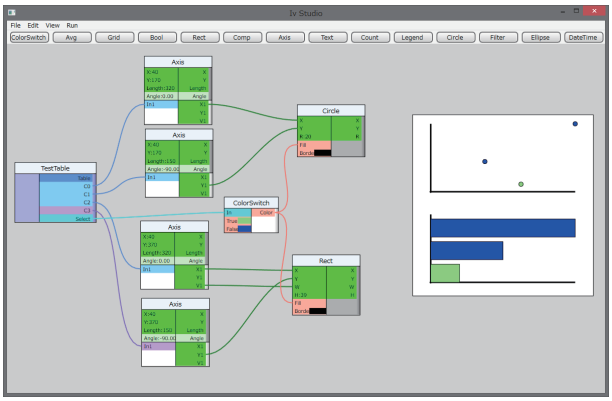

図 4　散布図と棒グラフにハイライト機能を追加した例

いの同じレコードの要素をハイライト表示する可視化手法（ブラッシング＆リンキング [19]）を作成した例である．ハイライト表示を作成するために，利用者はデータテーブルの Select ピンを散布図で用いている円の “Fill ピン” につなぐ．するとシステムによって，選択状態を色へ変換するノードが自動的に追加される．この変換ノードの出力を棒グラフの矩形の Fill ピンにつなぐことで，色を用いた棒グラフと散布図の連携を記述することができる．

3.3　インタフェース設計の意図

Iv Studio では，利用者が学習を必要とするインタフェースを少なくするため，

(1) データ，データ変換，視覚要素 のノードの追加と削除
(2) ピンの接続と切断
(3) プロパティパネルによるノードのプロパティの設定
(4) 視覚要素のハンドルインタフェースによるノードのプロパティの設定

の 4 種類の操作を用いて，データ変換の手順を設計するようにした．(1)〜(3) は，データフロー記述に最低限必要な機能である．(4) は，iVis Designer や Lyra で採用されているドローツール風のインタフェースであり，可視化結果の編集を直感的に行うために必要と考え採用した．

データフロー記述の操作を簡略化する機能として，システムによる自動的なノード追加とピン間の接続機能を用意した．ピンをつないだ際のシステムによる自動生成機能を “オートキャスト” と呼ぶ．（オートキャストの動作に関しては 4.2 節で詳しく述べる．）

オートキャストがない場合，ノードの追加とピンの接続の操作を多く行う必要がある．また，例えば棒グラフを作る場合，「棒グラフは矩形の幅で量を表すものだから，まずは軸を介して座標値に変換し，その値を矩形の X 座標と幅に割り当てる．」といった思考を利用者に要求してしまう．この思考には，Iv Studio の内部フレームワークの概念と変換のためのブロックに関して深い理解が必要となる．そこで，オートキャストを用意することで，「棒グラフは矩形の幅で量を表すものだから，データを幅に割り当てる」という考えで棒グラフを作成することができるようにした．これは，Lyra で採用されているインタフェースであり，有用と考え採用した．我々は先行研究のインタフェースに加え，システムによる支援が何を行ったのかを利用者に提示するため，その過程をアニメーションによって提示するようにしている．システムの動作を利用者に提示することで，システムの内部概念に沿った考え方の習得につながるものと考えている．

インタラクションの設計に関しては，レコードの選択状態をどのように視覚要素で表現するか記述する方式にしている．これにより，データの視覚要素への割り当てと同じインタフェースを用いてハイライト表示などのインタラクティブな可視化手法の記述を可能としており，学習が必要なインタフェースを少なくしている．

4.　データフロービジュアル言語

多彩な可視化手法を記述可能にするため，可視化ツールキットの設計を参考に，情報可視化の構成要素に基づいたデータ，データ変換，視覚要素 3 種類のブロックを用意した．データ変換のブロックとして，データの集計やフィルタリング，集約（平均値やレコード数の算出）を行うものと，データを視覚変数に変換するためのもの（軸やカラーマッピング）を用意した．これらに加え，四則演算，比較演算子，条件分岐もブロックとして用意することで，独自のデータ変換方法も記述できるようにした．

データフロービジュアル言語のスタイルとしては，サイエンティフィックビジュアリゼーションの分野では古くから用いられており，その有効性が確認されているノードリンクダイアグラム形式を採用した．ノードの位置は，利用者が自由に配置できるようにした．これは，可視化の流れを記述する際に，利用者のイメージ通りの配置を行える方が良いと考えたためである．

本章では，ノードと情報可視化用に考案したデータ型，およびオートキャスト機能について紹介し，拡張性に関して述べる．

4.1　ノードとデータ型

我々のデータフロービジュアル言語では，ブロックをノードとして用意した．ノードには，左側に入力ピンを，右側に出力ピンを持たせ，データの流れの方向を統一するようにしている．プログラム言語で考えた場合，ノードは関数，入力ピンは引数，出力ピンは返り値に該当するものである．

ピン同士の接続ミスを防ぐ目的と，データの流れの把握を支援することを目的に，ピンにはデータ型を設定しており，その型を色で表している．オートキャスト機能を除き，同じデータ型のピンしか接続することができないようにしている．データ型は，そのデータの利用目的に応じて設計

している．特徴的な設計としては，データレベルの数値を表す数値型（水色）と座標値を表す座標型（緑色）を分けているところである．この 2 つは，一つの実数であるが，可視化設計を行う上で区別する必要のある数値として別の型を用意した．

入力ピンには，接続されていない場合のデフォルトの値が指定されており，未接続のピンにはその値をピン内に表示するようにしている（図 1(d)）．表示方法は，数値の場合は数字を，色の場合はその色を表示するなど，型ごとにデザインしている．

4.2 オートキャスト

オートキャストは，利用者が異なるデータ型のピンを接続しようとした場合に，型変換を行うためのノードを自動的に挿入する機能である．型変換の方法は，入力ピンに設定されている．これは，単に 2 つのデータ型だけでは適切な変換方法を決められないためである．例えば，円の X 座標に数値が入力された場合は横向きの軸を経由して変換する必要があるが，Y 座標の場合は縦向きの軸を経由して変換する必要がある．この動作は，入力を受ける側のピンによって異なる動作が必要と考え，入力ピンに接続されるデータ型に応じた動作を設定するようにした．

オートキャストの種類としては，数値から座標値への変換だけではなく，文字列から座標値への変換（列挙），数値や文字列，真偽値から色への変換などが用意されており，データを視覚要素に割り当てる際のピンの接続作業を簡略化するために役立つ機能である．

4.3 スクリプト言語によるノードの拡張

我々の開発環境では，データフロービジュアル言語を採用することで，先行研究と比べて高い表現力を実現していると考えている．しかしながら，プログラミングによる記述に比べて表現力が劣ることは否定できない．そこで，利用者がスクリプト言語を記述することで新たなノードを作成できるようにした．

ノード記述に用いるスクリプト言語には，複数の返り値を返す構文を持ち，組み込みスクリプトとしては広く用いられている Lua を採用した．ノードのプロパティや動作を記述できるほか，プレビューキャンバス上での編集やオートキャストの動作も記述できる．

スクリプト言語による拡張は，利用者にプログラミングのスキルを要求してしまうものの，システム全体を全て記述する場合に比べて必要とする技術レベルは低いと考えている．我々の開発環境を習得した利用者が，より多くの場面で環境を活用できるようにするためには必要な機能であると考えている．

5. 評価

開発環境の評価として，いくつかの可視化手法を題材に，可視化手法の記述に要する操作量と表現力の高さについて確認した．

5.1 基礎的な可視化手法の記述にかかる操作量

図 1 や図 3 で示したとおり，基礎的な可視化手法を 4 ノードで記述することができる．このようなよく利用される可視化手法は，オートキャスト機能によって 1 回のノード追加操作と 2 回のドラッグ操作によって記述可能である．この他，折れ線グラフとエリアチャートの記述も同じ操作量で記述することができる．さらに，散布図のプロットの色や大きさに対して別のデータを割り当てるようなよく行われるカスタマイズも，1 回のドラック操作で可能である．この操作量は，先行研究の Lyra と同じ操作量であり，iVis Designer よりも少ない．これは，Iv Studio が先行研究に劣らない少ない操作量で記述できることを示している．

図 4 は，Lyra では記述することのできないインタラクティブな可視化手法を記述した例である．iVis Designer はこれに類する可視化手法を記述することができるものの，ブラッシングのためのオブジェクトの追加など多くの操作が必要である．これに対し，Iv Studio では先のデータ割り当ての延長として 2 回のドラッグでインタラクティブな手法の記述も行うことができる．

5.2 データ変換方法の記述

データ変換を記述した例として，ChronoView[20] を示す（図 5）．ChronoView はタイムスタンプを持った多くのイベントグループを可視化する手法である．各タイムスタンプがアナログ時計のように円周上に配置され，イベントグループがタイムスタンプの位置の重心に配置される．この手法は，我々の研究室で開発した独自の手法であるため，ツールキットや先行研究が対応していない有用な可視化手法の一つである．

図 5 は，購買データを可視化した例である．購買データは，売上のテーブルと商品のテーブルを持つ．時間情報を円周状に配置することや，値の平均の計算はよく行われる変換のため，Iv Studio では予めノードとして提供されている．この 2 つを組み合わせることで，ChronoView を作成した．時間情報を円周状に配置する軸のノードを追加し，その出力を平均を計算するノードにつなぐことで重心を計算した．そして，重心を円の座標に入力した．

Iv Studio では，データフロービジュアル言語を導入することで，先行研究のビジュアルインタフェースでは行えない独自の座標変換を記述することが可能である．

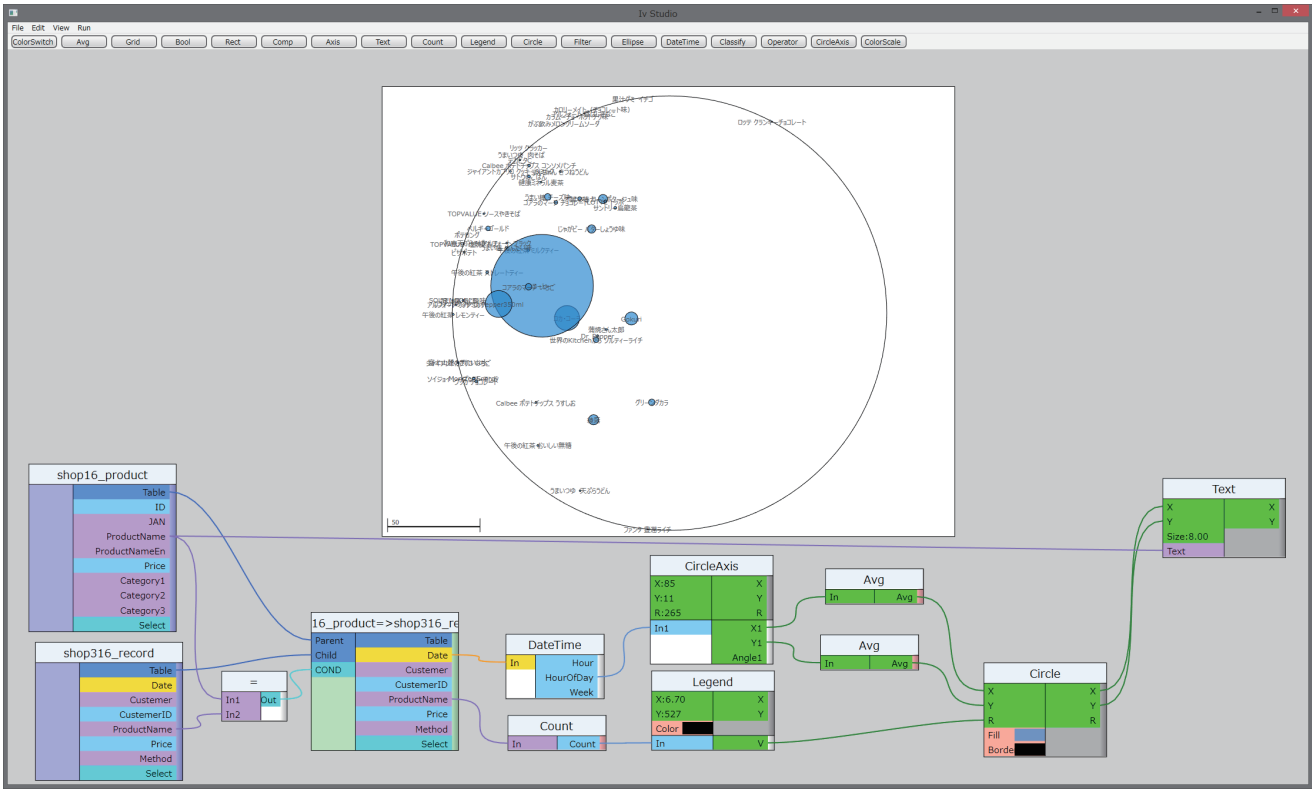

図 5　ChronoView を制作した例

5.3　フィルタリング機能を備えた可視化手法

第 4 章で述べたとおり，Iv Studio ではデータフロービジュアル言語を用いてデータの集約やフィルタリングを行うことができる．この条件式の記述も可能であり，データテーブルの Select ピンとこれらの機能を組み合わせることで，フィルタリング機能を備えた可視化手法の記述が可能である．

図 6 は，左に顧客の一覧を表示し，右には選択された顧客の曜日と時間帯の購入傾向を表す図を表示する可視化手法である．縦軸に時刻，横軸に曜日をとっており，商品を購入した部分に半透明の矩形を描くことで，購入した商品の曜日と時間帯の関係を提示している．データには前節で用いた購買データを用いている．図 7 はこの可視化手法の実行例であり，(a) は一番上の顧客のみを選択した図，(b) は上から 4 人の顧客を選択した図である．開発環境の利用者が，この可視化手法の記述が行えるようになるためには，相応の学習コストが必要になると考えられるものの，開発環境を使いこなすことで先行研究では記述できないインタラクティブな可視化手法も記述できることを示している．

6.　結論

本研究では，データフロービジュアル言語による可視化手法の記述インタフェースを備えた情報可視化システムの開発環境 Iv Studio を開発した．開発したインタフェースは，基礎的な可視化手法を制作できるようになった段階から

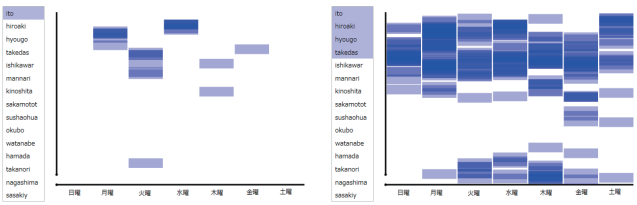

図 7　フィルタリング機能の実行例

ら多彩な可視化手法を記述できるようになるまでの隔たりを埋めるため，ツールの内部状態を利用者に対して積極的に提示するよう設計した．また，データと視覚要素への直感的な関連付けから，より詳細な手法の設計までの道筋を示すため，オートキャスト機能を開発した．

本論文では，Iv Studio を用いた可視化手法の記述例を通じて，我々の開発環境の表現力の高さを示した．しかし，開発したインタフェースが利用者の内部動作の理解をどの程度助け，多彩な可視化を設計できるようになるまでにどの程度の学習が必要かを調査する必要がある．そのため，既存のビジュアルインタフェースとの比較が今後の課題である．

参考文献

[1]　Fekete, J.-D.: The InfoVis Toolkit, *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, IEEE Computer Society, pp. 167–174 (2004).

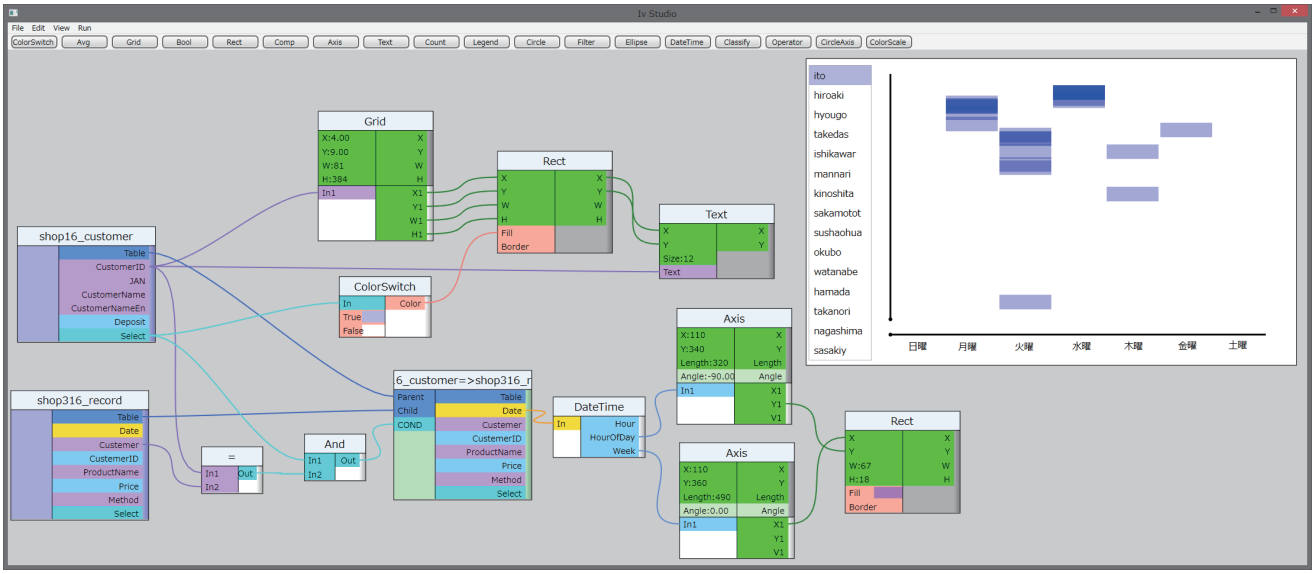

図 6　フィルタリング機能を備えた可視化手法

[2] Heer, J., Card, S. K. and Landay, J. A.: Prefuse: A Toolkit for Interactive Information Visualization, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, ACM, pp. 421–430 (2005).

[3] Bostock, M. and Heer, J.: Protovis: A Graphical Toolkit for Visualization, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, No. 6, pp. 1121–1128 (2009).

[4] Bostock, M., Ogievetsky, V. and Heer, J.: D3 Data-Driven Documents, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 2301–2309 (2011).

[5] Satyanarayan, A. and Heer, J.: Lyra: An Interactive Visualization Design Environment, *Computer Graphics Forum (Proc. EuroVis)*, Vol. 33, No. 3, pp. 351–360 (2014).

[6] Ren, D., Hollerer, T. and Yuan, X.: iVisDesigner: Expressive Interactive Design of Information Visualizations, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2092–2101 (2014).

[7] Card, S. K., Mackinlay, J. D. and Shneiderman, B.(eds.): *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publishers Inc. (1999).

[8] Johnston, W. M., Hanna, J. R. P. and Millar, R. J.: Advances in Dataflow Programming Languages, *ACM Comput. Surv.*, Vol. 36, No. 1, pp. 1–34 (2004).

[9] Claessen, J. H. T. and van Wijk, J. J.: Flexible Linked Axes for Multivariate Data Visualization, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 2310–2316 (2011).

[10] Weaver, C.: Building Highly-Coordinated Visualizations in Improvise, *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, IEEE Computer Society, pp. 159–166 (2004).

[11] North, C. and Shneiderman, B.: Snap-together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata, *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, ACM, pp. 128–135 (2000).

[12] Haeberli, P. E.: ConMan: A Visual Programming Language for Interactive Graphics, *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, ACM, pp. 103–111 (1988).

[13] Bencina, R.: Oasis Rose the composition - real-time DSP with AudioMulch, *Synaesthetica '98: Proceedings of the Australasian Computer Music Conference*, pp. 85–92 (1998).

[14] Hansaki, T., Shizuki, B., Misue, K. and Tanaka, J.: FindFlow: Visual Interface for Information Search Based on Intermediate Results, *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, APVis '06, Australian Computer Society, Inc., pp. 147–152 (2006).

[15] Zgraggen, E., Zeleznik, R. C. and Drucker, S. M.: PanoramicData: Data Analysis through Pen & Touch, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 12, pp. 2112–2121 (2014).

[16] Hils, D. D.: Datavis: A Visual Programming Language for Scientific Visualization, *Proceedings of the 19th Annual Conference on Computer Science*, CSC '91, ACM, pp. 439–448 (1991).

[17] Upson, C., Faulhaber, Jr., T., Kamins, D., Laidlaw, D. H., Schlegel, D., Vroom, J., Gurwitz, R. and van Dam, A.: The Application Visualization System: A Computational Environment for Scientific Visualization, *IEEE Comput. Graph. Appl.*, Vol. 9, No. 4, pp. 30–42 (1989).

[18] Parker, S. G. and Johnson, C. R.: SCIRun: A Scientific Programming Environment for Computational Steering, *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, Supercomputing '95, ACM (1995).

[19] Keim, D. A.: Information Visualization and Visual Data Mining, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, No. 1, pp. 1–8 (2002).

[20] Shiroi, S., Misue, K. and Tanaka, J.: ChronoView: Visualization Technique for Many Temporal Data, *Proceedings of the 2012 16th International Conference on Information Visualisation*, IV '12, IEEE Computer Society, pp. 112–117 (2012).