情　報　学　基　礎　70－9
デジタル・ドキュメント　38－9
（２００３．３．２８）

情報視覚化記述手法の開発

田辺 祥，砂岡 憲史，佐藤 克己，横山 節雄，宮寺 庸造

東京学芸大学

あらまし 本研究では，インタラクションを伴う情報視覚化をコンピュータで汎用的に扱うことを目的として，情報視覚化記述手法の開発を行った．記述手法の開発にあたり，まず情報視覚化の体系化に取り組んだ先行研究をもとに，情報視覚化に必要な表現要素を一般化した．さらに，一般化した表現要素を XML 文書として記述するための文法を定義することにより，情報視覚化記述をコンピュータで処理することを可能にした．また提案手法を用いて過去の代表的な視覚化技法を記述することにより，提案手法の適用可能性を確認した．本手法は，インタラクションを伴う情報視覚化を自動生成可能な汎用視覚化環境の構築に貢献するものと考えられる．

A Method for Describing Information Visualizations

Sho Tanabe, Norifumi Sunaoka, Yoshiki Sato, Setsuo Yokoyama
and Youzou Miyadera

Tokyo Gakugei University

Abstract This paper aims at treating information visualizations, include some interactions, on computers generally, and develops a new method for describing information visualizations. First, the paper generalized elements for expressing information visualizations based on the past researches that try out to systematise information visualizations. Second, the paper defined a descriptive grammer to express information visualizations as XML documents with XML-DTD, and it enables to process the information visualizations on computers. Third, the paper confirmed an appricapability of the proposed method by describing some past popular information visualization techniques with the method. It is expected that the melthod will contribute to implement a general environment for information visualizations that can automatically visualize various data with suitable interactions.

1　はじめに

近年，問題解決や意思決定の手段として情報視覚化が普及しつつある．情報視覚化は通常は目に見えない構造やデータを視覚化する技術であり，最近ではさまざまな情報視覚化を行うことのできる汎用視覚化ツールも登場している [3][13]．しかし既存の汎用視覚化ツールでは，対象オブジェクトを図的表現に変換することはできるものの，より深い分析に不可欠である，ノードをつまんでの移動，視点の変更，ズーミングといったようなインタラクションを実装できていない．この問題は，情報視覚化におけるインタラクションを，コンピュータ上で汎用的に扱う枠組みがないことに起因している．

これまでにも情報視覚化に伴うインタラクションを特徴付けようとする試みはあったが，これを一般化し，コンピュータ上で汎用的に扱うことを実現した研究はなかった．上記の問題を解決するためには，インタラクションを伴う情報視覚化を，コンピュータに解釈可能な形で表現する新たな枠組みが必要である．

本研究では，情報視覚化をコンピュータで自動処

理することを前提として，インタラクションが記述可能な新たな情報視覚化記述手法を開発する．

2　関連研究

本研究の目的達成の観点から，汎用視覚化ツールと情報視覚化の体系化に関する研究についてその問題点を議論する．

2.1　汎用視覚化ツールに関する研究

汎用視覚化ツールには，既存の視覚化手法を統合して汎用性を高めた統合視覚化ツール [4][9][1][13]，視覚化に必要な部品を繋ぎ合わせて視覚化を実現するモジュラー視覚化ツール (MVE: Modular Visualization Environment)[14][2][3][20]，ユーザの視覚化要求と対象データをもとに自動的に視覚化を行う自動視覚化ツール [7][19] がある．

統合視覚化ツールは既存の独立した技法をひとつのシステムにまとめあげたもので，内部的に情報視覚化を汎用的に扱う枠組みを持たない．それぞれの技法に特有のインタラクションが実装されてはいるものの，特定の目的での使用に限られる．

モジュラー視覚化ツールでは，自由度の高い視覚化が行えるものの，部品が細かく分かれており設定が複雑な上，どのように視覚化されるのか，その結果を予想することが難しい．またユーザインタラクションを扱う仕組みがないため，異なる視点による視覚化を行うには部品を選び直さなければならず，試行錯誤的な操作が難しい．

自動視覚化ツールでは，モジュラー視覚化ツールにおける設定の複雑さを解消した研究があるが [7]，ユーザの選択項目と設定のパターンファイルをひとつひとつ用意する必要があり，拡張性や汎用性に欠ける．また，実現可能な視覚化の範囲を絞ることにより汎用性の高い自動視覚化を実現した研究もあるが [19]，ユーザインタラクションを扱う枠組みを持たない．

2.2　情報視覚化の体系化に関する研究

一方，情報視覚化の体系化に関する研究には，既存の情報視覚化技法に分類観点を与えるものと，既存の情報視覚化の特徴を記述するものとがある．

視覚化技法の分類を目的とした研究のうち Wehrend の分類 [17] では，情報視覚化におけるユーザの目的に焦点を当てて分類観点が示されているが，インタラクションという観点からは一貫した視点を得ることが難しい．Shneiderman の分類 [12] では，タスクとしてインタラクションに深く関わる項目が整備されているが，それぞれのタスクの具体的な操作に関する議論がなく，実際の動作との関連付けが必要である．Keim の分類 [10] では，視覚化技法を分類する 3 つの視点のひとつとしてインタラクションが挙げられているが，分類観点を与えるに過ぎない．

一方，視覚化技法の記述を目的とした研究のうち Tweedie の記述 [16] では，代表的な視覚化技法に含まれるインタラクションを特徴付けているが，インタラクションの具体的な内容を言葉で記述するにとどまっている．Card の記述 [5] では，グラフィカルな要素に関して深く一般化の試みが行われているものの，インタラクションの一般化についてはほとんど議論されていない．Chi の記述 [6] では視覚化技法のオペレーションに焦点が当てられており，一部インタラクションに関する記述もあるが，非常に簡単な表現にとどまっている．

2.3　関連研究に関する議論

扱う表現要素の種類と自動化の 2 つの観点から，関連研究の動向をまとめた (図 1)．横軸は扱う表現要素の種類を表している．縦軸は，コンピュータによる自動化を目的としているかどうかの観点から，あくまで理論的な分類・記述にとどまるのか，さらに実システムとしての自動化を目指したのかどうかを表している．図 1 からわかるように，動的要素の自動化に関する研究はいまだにない．本研究は，この研究領域を主な対象とする．

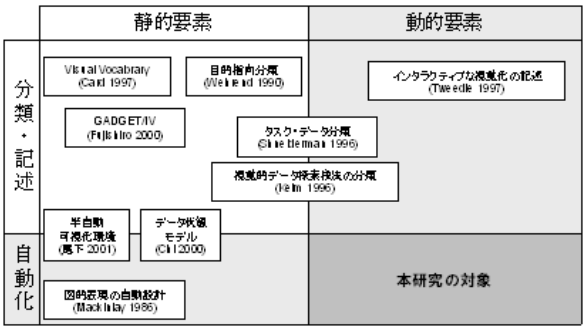

図 1: 関連研究の動向

総じて，情報視覚化におけるインタラクションの重要性は十分に認識され，インタラクションという観点から視覚化技法の特徴を表現しようという試みは既にあるものの，これを一般化し，自動生成につなげようとする試みは少ない．これには，情報視

覚化におけるインタラクションに関する共通要素の発見や，プログラム記述の困難などの問題が関係していると思われる．従って，情報視覚化におけるインタラクションをコンピュータに解釈可能な形で記述できる記述手法を開発するためには，インタラクションに関する要素の一般化と，一般化されたインタラクションを実行可能なコードとして記述するための枠組みが必要である．

また一般に，情報視覚化を自動的に行うためには，
1) ユーザの視覚化要求と視覚化対象データの抽出
2) 抽出した情報に応じた視覚化の自動生成
の 2 つの作業が必要である．さらにこれらの作業を実際にコンピュータ上で実現するためには，抽出した情報を保持するための何らかの内部表現手段が必要である (図 2)．これはあくまでコンピュータで取り扱うために必要なものでユーザが直接書くものではないが，自動視覚化の実現や，異なるシステム間での情報視覚化技法の流通のために不可欠であると思われる．しかし現在のところ，情報視覚化に伴うインタラクションも含めてこれらを表現するための仕様を明確に定めた研究はない．本研究の目的は，このための情報視覚化記述手法を開発することである．

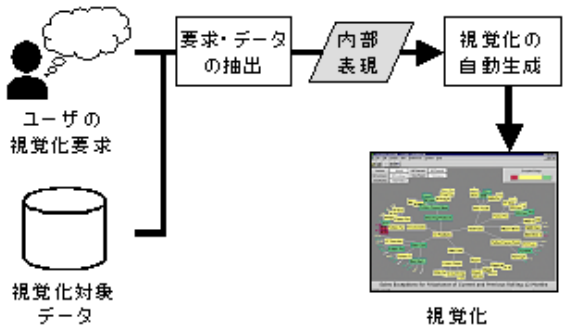

図 2: 自動視覚化の流れ

3　開発に関する検討

情報視覚化におけるインタラクションの主な目的の一つは，対象データの構造を多角的に観察することである．これより，構造の視覚化に重点を置いたツリーの視覚化とグラフの視覚化に研究の対象を絞った．また構造表現されたものに対するインタラクションは視覚化の構成要素ごとに表現する必要があると考え，代表的なツリーやグラフの視覚化事例をもとに視覚化の構成要素を抽出した．その結果，視覚化の構成要素として，視覚化対象オブジェクトを表すノード (Node)，ノード間の関係を表すコネクタ (Connector)，これらを配置，描画するための描画領域 (Area) の 3 つを定義した (図 3)．

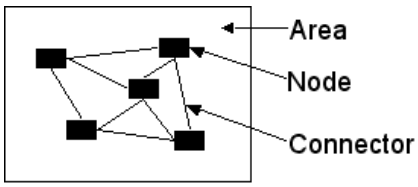

図 3: 情報視覚化の構成要素

ここで情報視覚化におけるインタラクションは，各々の構成要素の別に実装されるものと捉えることができる．またこれらを，各構成要素に含まれる静的要素を属性，動的要素をメソッドとしたオブジェクトとして捉えることにより，各構成要素をオブジェクト指向言語におけるクラスとして定義することも可能である (図 4)．各構成要素をオブジェクト指向言語におけるクラスとして定義することができれば，オブジェクト指向言語を用いて，これらを実行可能なコードとして実装することも可能である．

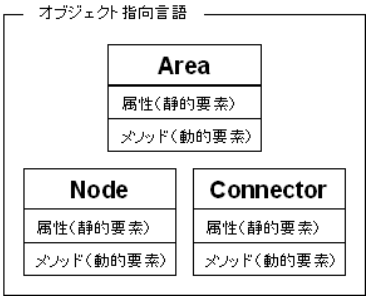

図 4: 視覚化構成要素のクラス定義

本研究は，様々な情報視覚化の特徴を Area, Node, Connector の 3 つのオブジェクトにおける属性とメソッドを詳細に定義し，さらにこれをオブジェクト指向言語におけるクラスとして定義した上でプログラム中から呼び出して利用するという枠組みで行う．すなわち本研究の目的は，この枠組みの中で，各クラスに関するインタラクションを表現するために必要な属性とメソッドを明らかにし，これらを詳細に定義するための記述言語を開発することである．このようにして情報視覚化を記述するための仕様を明確に定めてはじめて，自動視覚化を実現できると考える．

4　記述手法の開発

まず過去の研究をもとに，情報視覚化を表現するために必要な表現要素 (属性とメソッド) を一般化し，次にその記述の詳細を定め，最後にこれを XML 文書として記述するための文法として XML-DTD を定義した．

4.1　情報視覚化の表現要素

まず過去の研究をもとに情報視覚化を表現するための表現要素を一般化した．一般化の作業は，過去の研究で議論されている表現要素の抽出 (第 1 段階) と，これをもとにしたインタラクションの表現のための独自の拡張 (第 2 段階) の二つの段階で行った．

第 1 段階では主に，静的要素は Visual Vocabrary[5] と自動可視化環境 [19] から，動的要素は目的指向分類 [17] とデータ状態モデル [6], Gadget/IV[7]，半自動可視化環境 [19] から表現要素の抽出を行った．これらの研究の中ではそれぞれ独自の視点で情報視覚化の表現要素に関する議論が行われていた．自動視覚化という観点から見て具体的なものから概念的なものまで様々であったが，表現要素として議論されていると思われるすべての項目を抽出し，意味的に重複しているものはまとめた．

第 2 段階では，我々がこれまでに開発してきた学術論文関係視覚化システム [15] と Web 視覚化システム [18]，さらには，既存の代表的な実装済み視覚化システムをもとに，実システムとして実装することを前提として拡張を行った．そのためにまず，動的要素についてこれまでの議論では曖昧だった部分を排除し，不足しているものを加え，ツリーやグラフの視覚化で実装可能なインタラクションを網羅するように動的要素を決定した．さらに，決定した動的要素をオブジェクト指向言語におけるメソッドとして実装できることを要件として，静的要素に関して不要なものを排除し，また不足しているものを補った．

第 1，第 2 段階の一般化の作業の結果得られた表現要素の一覧を表 1 に示す．

表現要素は，Area，Node，Connector の別に，静的要素 (属性)，動的要素 (メソッド) の順に示した．ここで，末尾に「＋」がついている要素はさらに細かい要素に分けられることを意味する．

表現要素	説明
Area	
Dimension	描画領域の次元
VisualizationKind	視覚化の種類
RetinalEncodings+	エリアの視覚的要素
Overview	全体を概観する
History	履歴を参照する
Rotate	全体，またはツリーの一部を回転させる
Zoom	ズーム・イン／アウトする
Move	全体を移動させる
Focus	任意の点にフォーカスする
Node	
Expression+	ノードの配置方法
Shape	ノードの形
Entity	実体として関連付けるデータ
Originals	ノードに関連付けるデータ
Variables	動的変更を許可するデータ
Details	詳細情報として関連付けるデータ
RetinalEncodings+	ノードの視覚的要素
Zoom	任意のノードにズーム・イン／アウトする
Move	ノードを移動させる
Focus	任意のノードにフォーカスする
Extract	ノードの実体 (Entity) を抽出する
Relate	任意の 2 ノードを関連付ける
Cluster	任意の複数ノードをグループ化する
ChangeExpressions	ノードの配置方法を変更する
DetailsOnDemand	ノードの詳細情報 (Details) を参照する
Filter	動的変更を許可されたデータに関してフィルターを行う
Connector	
Condition+	ノード間に線を引く条件
Motion	Move に伴うノードの動き
Draw	線の引き方
Weight	コネクタの重みとして関連付けるデータ
Originals	コネクタに関連付けるデータ
Variables	動的変更を許可するデータ
Details	詳細情報として関連付けるデータ
RetinalEncodings+	コネクタの視覚的要素
ChangeConditions	ノード間に線を引く条件を変更する
DetailsOnDemand	コネクタの詳細情報 (Deatails) を参照する
Filter	動的変更を許可されたデータに関してフィルターを行う

表 1: 視覚化構成オブジェクトの表現要素一覧

Area の表現要素

Area クラスは，属性として描画領域の次元数 (2 または 3)，視覚化の種類 (ツリー，またはグラフ)，視覚的要素を持つ．Node クラスや Connector クラスにも共通する視覚的要素には，色や形，大きさ，線の色，太さ，フォントに関する情報などが含まれる．Area クラスのメソッドは全部で 6 つあり，どれも描画領域に含まれるすべてのオブジェクトに対して行われる．History は，ユーザが行ってきたインタラクション操作の履歴を遡ってそのときの状態を表示するメソッドである．その他のメソッドは，Node オブジェクトの属性を操作することによって実現される．

Node の表現要素

Node クラスは，属性として配置方法 (描画アルゴリズム) や，Node オブジェクトに関連付けるデータなどを持つ．本記述手法では視覚化対象データは

RDBMS 上に蓄積されているものと仮定し，関連付けるデータはすべてデータベース名とテーブル名，フィールド名を用いて指定する．またここでいう実体 (Entity) とは，たとえば一つの Web ページが一つの Node として視覚化されている場合に，Web ページそのものを意味する．すなわち Extract メソッドは，Node オブジェクトに関連付けられている実際のファイルなどを取得する操作である．Node クラスのメソッドには他に Node オブジェクトの位置や視点を変更するものや，Node オブジェクト同士を関連付ける操作，Node オブジェクトに関連付けられた情報を参照する操作などがあり，基本的にある一つの Node オブジェクトに対して行われる．

Connector の表現要素

Connector オブジェクトはすべての Node オブジェクト間に定義されるものであり，これに伴って，線を引くための条件を与える属性を持つ．Connector クラスは他にも属性として線の引き方や，Node オブジェクトが移動した場合の動きなどを持つ．またメソッドとしては線を引く条件の変更や，関連付けられた情報の参照などを持つ．

これらすべてのメソッドを実システムとして実装する場合には，それぞれのクラスで属性として与えられた変数の値を操作する関数として与えられる．本研究では，これを前提として，メソッドと属性の関係を明確に定義している．

またこれらとは別に，インタラクション (メソッド) のきっかけとなりうるマウスイベントを定義し (表 2)，各々のメソッドに割当可能なマウスイベントを定めた (表 3)．表 3 において◎印は，割り当てるマウスイベントを特に指定しなかった場合に割り当てられるマウスイベント (デフォルト値) を表している．ここですべてのメソッドは，マウスイベントを割り当てられるほかに，描画領域とは別のコントロール領域にボタンやスライダなどの GUI コンポーネントとして実装することもできる．表 3 においてデフォルト値が設定されていない，あるいはマウスイベントが割り当てられていない要素は，基本的にはデフォルトでコントロール領域に実装される．

4.2 情報視覚化記述の詳細

次にこれらの表現要素のすべてについて記述の詳細を定義した．その最大の目的は，各々の表現要素に与えることのできる値とその意味を明確にすることである．記述の詳細は，以下の表記法に従って定義した．

ID	マウスイベント	説明
MO	MouseOver	マウスによるポインティング
LMC	LeftMouseClick	左ボタンによるクリック
LMD	LeftMouseDrag	左ボタンによるドラッグ
LDC	LefttDoubleClick	左ボタンによるダブルクリック
RMC	RightMouseClick	右ボタンによるクリック
RMD	RightMouseDrag	右ボタンによるドラッグ
RDC	RightDoubleClick	右ボタンによるダブルクリック
BMD	BothMouseDrag	両ボタンによるドラッグ

表 2: マウスイベント一覧

	MO	LMC	RMC	LDC	RDC	LMD	RMD	BMD
Area								
Overview		○	○	○	○			
Focus		◎	○	○	○			
History		○	○					
Rotate						○	◎	
Zoom						◎	○	
Move						○	○	◎
Node								
Focus		◎	○	○	○			
Extract				◎	○			
Zoom						◎	○	
Move						○	○	◎
Relate						○	○	
Cluster						○	○	
ChangeExpressions								
DetailsOnDemand	◎	○	○	○	○			
Filter								
Connector								
ChangeConditions								
DetailsOnDemand	◎	○	○	○	○			
Filter								

表 3: メソッドに割当可能なマウスイベント (◎印はデフォルト値)

<表現要素名> 記述項目の説明．(デフォルト値)
[記述項目に与える値] 値の説明
[記述項目に与える値] 値の説明
...

記述項目がさらに詳細な記述項目に分けられる場合には，これを入れ子にして記述する．図 5 にその一部を示す．この例では，Area クラスの Dimension，VisualizationKind と，RetinalEncodings の一部，Node クラスのメソッドの一部を定めている．RetinalEncodings では，Size や Color など，さらに詳細な項目が定義される．またメソッドについては，実装することのできるメソッドと，メソッドを呼び出す際に必要なパラメタなどが定義される．

4.3 情報視覚化記述言語

前節で定めた記述の詳細に基づいて XML-DTD を定義した．XML-DTD は XML 文書に文法を与える言語である．ここでは，記述する項目や記述の順番，与えることのできる値，デフォルト値などを明確に定義した．図 6 にその一部を示す．

図 6 は，XML-DTD の冒頭部分と，インタラクションに関する項目の定義の一部を示している．記述

```
<Dimension> 次元数．(2)
    [2] 2 次元．
    [3] 3 次元．
<VisualizationKind> 視覚化の種類．(network)
    [scatter] 散布図．
    [tree] ツリー．
    [network] ネットワーク．
<RetinalEncodings> 視覚的要素．
    <Size> サイズの設定．
        <Width> 幅．ピクセルまたは Window 幅に対する割合で指定．(100%)
        <Height> 高さ．ピクセルまたは Window 幅に対する割合で指定．(100%)
        <Depth> 奥行き．ピクセルで指定．(600)
    <Color> 色の設定．
        <BackgroundColor> 前景色．RGB の組で指定する．(255.255.255)
        <ForegroundColor> 背景色．RGB の組で指定する．(0.0.0)
    <Border> 線の設定．
        <BorderWidth> 線の太さ．ピクセルで指定する．(1)
        <BorderColor> 線色．RGB の組で指定する．(0.0.0)
        <BorderKind> 線の種類．(solid)
            [solid] 実線．
            [broken] 破線．
            ....
<Zoom> ズーム．すべての Node オブジェクトの中心座標について，ポインティングされた
    ノードの中心からの距離を伸縮させる．(RightMouseDrag)
<Move> 移動．ノードの座標を連続的に変化させることによってノードを移動させる．
    (LeftMouseDrag)
<Extract> 抽出．entity で指定された実体を抽出する．(LeftDoubleClick)
<DetailsOnDemand> 詳細情報の表示．Details で指定されている情報を，指定された
    方法で表示する．(MouseOver)(popup)
    [popup] ポップアップで表示．
    [textarea] コントロールパネルのテキストエリアに表示．
    [atatusbar] ステータスバーに表示．
            ....
```

図 5: 記述の詳細 (部分)

の全体は「Visualization」タグで囲まれる．さらにその中に，Area，Node，Connector それぞれのクラスの特徴を記述する．記述の全体は階層構造になっており，Area クラスに関する記述はすべて「Area」タグによって囲まれる必要がある．例えば XML-DTD 中の 3〜5 行目は，Area クラスには RetinalEncoding と AreaInteraction が含まれ，その他にパラメタとして Dimension と VisulaizationKind を指定することができることを意味している．RetinalEncoding や AreaInteraction などの項目の記述文法についても，同じように定義される．

この XML-DTD に従って XML 文書を生成することにより，コンピュータに解釈可能なデータとして情報視覚化におけるインタラクションを表現することが可能である．

5 記述手法の適用

提案手法の適用可能性を議論するため，既存の代表的な視覚化技法の記述を試みた．以下，提案手法による各視覚化技法の解釈と，各視覚化技法で実装されているインタラクションに焦点を当て，記述の詳細を述べる．

5.1 Cone Trees の記述

Cone Trees[11] は，ファイル階層を 3 次元空間に視覚化する技法である．ここでは，テキストの書かれた長方形を Node，Node と Node をつないでいる線を Connector，Node と Connector が描画されて

```
<!DOCTYPE Vislualization [

    <!ELEMENT Visualization (Area, Node, Connector)>

    <!ELEMENT Area (RetinalEncoding, AreaInteraction?)>
    <!ATTLIST Area Dimension (2 | 3) #REQUIRED>
    <!ATTLIST Area VisualizationKind (graph | tree)
     #REQUIRED>

    <!ELEMENT Node (Expressions+, DataSet, RetinalEncoding, NodeInteraction?)>
    <!ATTLIST Node Shape (circle | rectangle) "rectangle">

    <!ELEMENT Connector (ConditionVisible+, DataSet, RetinalEncoding,
     ConnectorInteraction?)>
    <!ATTLIST Connector Motion (normal | shrink | spring) "normal">
    <!ATTLIST Connector Draw (straight | folded | curved) "straight">

    <!ELEMENT Expressions (#PCDATA)>
    <!ATTLIST Expressions CoordinateSystem (polar | rectangular | spring)
     #REQUIRED>
    <!ATTLIST Default (true | false) #REQUIRED>

    <!ELEMENT Conditions (#PCDATA)>
    <!ATTLIST Default (true | false) #REQUIRED>

    <!ELEMENT DataSet (Originals*, Variables*, Details*, NodeInfo?, Entity?,
     Weight?)>
    <!ELEMENT Originals (#PCDATA)>
    <!ELEMENT Variables (#PCDATA)>
    <!ELEMENT Details (#PCDATA)>
    <!ELEMENT NodeInfo (Root, Parent, Child)>
    <!ELEMENT Entity (#PCDATA)>
    <!ELEMENT Weight (#PCDATA)>

    <!ELEMENT Root (#PCDATA)>
    <!ELEMENT Parent (#PCDATA)>
    <!ELEMENT Child (#PCDATA)>

    <!ELEMENT RetinalEncoding (Size?, Color, Border, Label?, Font?)>

    <!ELEMENT Size ()>
    <!ATTLIST Size Width (CDATA) "auto">
    <!ATTLIST Size Height (CDATA) "auto">
    <!ATTLIST Size Depth (CDATA) "auto">

                        (中略)

    <!ELEMENT AreaInteraction (Overview?, History?, Rotate?, Zoom?, Move?,
     Focus?)>
    <!ELEMENT NodeInteraction (Zoom?, Move?, Relate?, Cluster?, Extract?,
     ChangeExpression?, DetailsOnDemand?, Filter?)>
    <!ELEMENT ConnectorInteraction (DetailsOnDemand?, Filter?, ChangeCondition?)>

    <!ELEMENT Overview ()>
    <!ATTLIST Overview Apply (true | false) "false">
    <!ATTLIST Overview Event (leftMouseClick | rightMouseClick | leftDoubleClick |
     rightDoubleClick | onControlPanel) "onControlPanel">

    <!ELEMENT History ()>
    <!ATTLIST History Apply (true | false) "false">
    <!ATTLIST History Event (leftMouseDrag | rightMouseDrag | onControlPanel)
     "onControlPanel">

    <!ELEMENT Rotate ()>
    <!ATTLIST Rotate Apply (true | false) "false">
    <!ATTLIST Rotate Event (leftMouseDrag | rightMouseDrag | onControlPanel)
     "rightMouseDrag">

    <!ELEMENT Zoom ()>
    <!ATTLIST Zoom Apply (true | false) "false">
    <!ATTLIST Zoom Event (leftMouseDrag | rightMouseDrag | onControlPanel)
     "leftMouseDrag">

    <!ELEMENT Move ()>
    <!ATTLIST Move Apply (true | false) "false">
    <!ATTLIST Move Event (leftMouseDrag | rightMouseDrag | bothMouseDrag |
     onControlPanel) "bothMouseDrag">

    <!ELEMENT Focus (#PCDATA)>
    <!ATTLIST Focus Apply (true | false) "false">
    <!ATTLIST Focus Event (leftMouseClick | rightMouseClick | leftDoubleClick |
     rightDoubleClick | onControlPanel) "leftMouseClick">

                        (中略)

]>
```

図 6: XML-DTD による記述文法の定義

いる 3 次元空間を Area と解釈する．Area における VisualizationKind は Tree である．各ノードの配置アルゴリズムは，Cone Trees 独自の関数が呼び出される．ノード間における親子関係の存在がコネクタの可視条件である．

Cone Trees では，親ノードと階層化に配置された子ノードを結ぶ円錐形を回転させることができる．

この回転によって，奥に配置されているノードを参照することができる．

以下に，このインタラクションを XML 形式で記述した部分を示す．

```
<Visualization>
    <Area Dimension = "3" VisualizationKind = "Tree">
            ....
    </Area>

    <Node Shape = "rectangle">
<Expressions/>

        <RetinalEncoding>
            <Size Width = "80"/>
            <Color BackgroundColor = "204.204.204"/>
            <Border/>
            <Label LabelPosition = "center">
                hoge.hoge.node_name
            </Label>
            <Font/>
        </RetinalEncoding>

        <DataSet>
             ....
        </DataSet>

        <NodeInteraction>
            <Rotate
             Apply = "true"
             axisY = "this.parent.y"
             axisZ = "this.parent.z"
            />
        </NodeInteraction>
    </Node>

    <Connector>
            ....
    </Connector>
</Visualization>
```

Cone Trees におけるインタラクションは，ノードの Rotate(回転) メソッドとして実装できる．ノードの Rotate には，XML-DTD によって，デフォルトでマウス左ドラッグがイベントとして割り当てられている．上記ではとくにイベントを指定していないので，このイベントが割り当てられる．また，回転軸を与える必要があるため，3 次元上で直線を決定するための 2 座標が与えられている．

5.2 Fish Eye Views の記述

Fish Eye Views[8] は，魚眼レンズのような効果によって，描画領域中の特定の部分を非線形に拡大・縮小する視覚化技法である．ここでは，丸で囲まれた数字を Node，Node と Node を結ぶ線を Connector，これらが描画されている平面を Area としてとらえる．Area は 2 次元であり，Tree の描画が行われている．Node の配置は Fisheye Views 独自の関数で与えられる．また，線が引かれる条件は Node 間に親子関係があった場合で，折れ線による線画である．

インタラクションとしては，Area の任意の点に対する Focus が実装されている．マウスクリックによるポインティングをイベントとする．

以下に，このインタラクションを XML 形式で記述した部分を示す．

```
<Visualization>
    <Area
     Dimension = "2"
     VisualizationKind = "Tree">
        <RetinalEncoding>
            <Size/>
            <Color/
            <Border/>
            <Label>
                Graphical Fisheye Views
            </Label>
            <Font FontSize = "20"/>
        </RetinalEncoding>

        <AreaInteraction>
            <Focus
             Apply = "true"
             Function = "fisheye"/>
        </AreaInteraction>
    </Area>

    <Node>
       ....
    </Node>

    <Connector>
       ....
    </Connector>
</Visualization>
```

Fisheye Views のインタラクションは，Area における Focus メソッドとして実装できる．Focus メソッドでは，とくに指定がない場合には標準の Focus 関数がライブラリから呼び出され，フォーカスの中心座標と Area のサイズ，ノードの中心座標が引数として渡される．Fisheye Views の場合には Fisheye 専用の Focus 関数である”fisheye”が呼び出され，同様の引数が自動的に渡され，各ノードの座標がリターンされる．

5.3 適用可能性に関する議論

このように，提案手法を用いることにより，XML 文書として情報視覚化におけるインタラクションを記述することができる．さらに本研究では，先にも述べた通り，この XML 文書から自動的に視覚化を行うための枠組み関しても議論を行った．まず先に示した視覚化構成オブジェクトの各々を，オブジェクト指向言語におけるクラスとしてとらえる．この考えに基づき，XML 文書として記述された視覚化記述から，オブジェクト指向言語のクラスを自動生成する．このクラスを用いることにより，自動視覚化を実現する．

現在，上記の枠組みに基づいて，オブジェクト指向言語 Java におけるクラスを自動生成するモジュールや各視覚化に固有の配置アルゴリズムなどのライブラリを開発中である．将来的には，この枠組みに基づく自動視覚化システムの実現を目指している．

6 おわりに

本研究では，情報視覚化におけるインタラクションを，コンピュータに解釈可能な形で記述可能な記述手法を開発した．提案した記述手法では，ツリー・グラフベースの情報視覚化に伴うインタラクションを記述することができる．また，提案した記述手法は XML-DTD によって記述文法が定義されており，XML 文書を処理できる環境さえあれば，これをコンピュータ上で解釈することができる．また，過去の代表的な視覚化技法のインタラクションを実際に提案手法を用いて記述することにより，提案手法の適用可能性を示した．さらに，提案手法を用いた記述を，オブジェクト指向言語におけるクラスに変換する枠組みについても議論した．提案手法は，インタラクションを扱う汎用視覚化ツールを構築するための基礎を形作るものである．

今後の課題としては，表現要素と記述文法のさらなる精錬，適用可能性の評価の充実，オブジェクト指向言語におけるクラスライブラリの開発などが挙げられる．

参考文献

[1] C. Ahlberg and E. Wistrand. IVEE: An Information Visualization & Exploration Environment. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '95)*, pp. 66–73, 1995.

[2] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff. Tioga2: A direct manipulation database visualization environment. In *Proceedings of 12th International Conference on Data Engineering (ICDE '96)*, pp. 208–17, 1996.

[3] AVS. http://www.avs.com.

[4] Stuart K. Card and Jock D. Mackinlay George G. Robertson. The Information Visualizer, An information Workspace. In *Proceeding of the ACM Conference on Human Factors in Computing Systems (CHI '91)*, pp. 181–186, 1991.

[5] Stuart K. Card and Jock Mackinlay. The Structure of the Information Visualization Design Space. In *Proceedings of IEEE Symposium on Information Visualization 1997 (InfoVis '97)*, pp. 92–99, 1997.

[6] Ed H. Chi. A Taxonomy of Visualization Techniques using the Data State Reference Model. In *Proceedings of IEEE Symposium on Information Visualization 2000 (InfoVis '00)*, pp. 69–75, 2000.

[7] Issei Fujishiro, Rika Furuhata, Yoshihiko Ichikawa, and Yuriko Takeshima. GADGET/IV: A Taxonomic Approach to Semi-Automatic Design of Information Visualization Applications Using Modular Visualization Environment. In *Proceedings of IEEE Symposium on Information Visualization 2000 (InfoVis '00)*, 2000.

[8] G. Furnas. Generalized Fisheye Views. Proceeding of the 1886 ACM Conferrence on Human Factors in Computing Systems (CHI‘ 86), pp. 18–23, 1986.

[9] Inxight. http://www.inxight.com.

[10] Daniel A. Keim. An Introduction to Information Visualization Techniques for Exploring Very Large Databases. In *Tutorial Notes of IEEE Information Visualization '00*, 2000.

[11] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '91)*, pp. 189–194, 1991.

[12] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of IEEE Symposium on Visual Languages 1996 (VL '96)*, pp. 336–343, 1996.

[13] Spotfire. http://www.spotfire.com.

[14] M. Stonebraker, J. Chen, N. Nathan, C. Paxson, A. Su, and J. Wu. Tioga: A Database-Oriented Visualization Tool. In *Proceeding of the IEEE International Conference on Visualization 1993*, pp. 86–93, 1993.

[15] Sho Tanabe, Kayoko Oyobe, Norifumi Sunaoka, Setsuo Yokoyama, and Youzou Miyadera. A Visualization System of Relationships among Papers Based on the Graph Drawing Problem. In *Proceeding of the IEEE Sixth International Conference on Information Visualisation (IV'02)*, pp. 202–210, 2002.

[16] Lisa Tweedie. Characterizing Interactive Externalizations. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI '97)*, pp. 375–382, 1997.

[17] Stephen Wehrend and Clayton Lewis. A Problem-oriented Classification of Visualization Technics. In *Proceedings of IEEE Visualization '90*, pp. 139–143, 1990.

[18] 砂岡憲史. Web 視覚化システム. 東京学芸大学修士論文, 2002.

[19] 尾下真樹, 牧之内顕文. オブジェクト指向データベースの半自動可視化環境. 情報処理学会論文誌:データベース, Vol. 42, No. SIG1(TOD 8), pp. 56–69, 2001.

[20] 磯辺成二, 黒川清, 塩原寿子, 飯塚哲也. 視覚化によるデータ分析システム：INFOVISOR. 情報処理学会論文誌, Vol. 40, No. 5, pp. 2417–2428, 1999.