オブジェクト指向方法論に基づくオブジェクト図の自動レイアウト

中島 哲[†] 田中二郎[††]

グラフを見やすく配置するグラフ描画アルゴリズムを用いて，オブジェクト指向方法論 OMT に基づくオブジェクト図を対象とした自動レイアウト手法を提案する．オブジェクト図は複数の木構造を含む複雑なグラフとして表されるため，従来の単純な描画アルゴリズムでは適用が困難であった．我々の手法では，オブジェクト図を2段階の階層構造と見なし，木構造描画アルゴリズムと無向グラフ描画アルゴリズムの2種類を適用する．これにより，継承構造を強調したオブジェクト図を自動描画できることを確認した．また，そのレイアウト手法を用いて，複数のアルゴリズムを選択・利用できる自動レイアウトシステムを開発し，オブジェクト図の描画に有効であることを確認した．

Automatic Layout of Object Diagrams Based on Object Oriented Methodology

SATOSHI NAKASHIMA[†] and JIRO TANAKA[††]

We have developed an automatic layout algorithm for object diagrams based on OMT methodology with graph drawing algorithms. Existing graph drawing algorithms can not be applied to object diagrams because a object diagram is a complicated graph which includes multiple tree structures. In our algorithm, an object diagram is considered as a graph which consists of two hierarchies. Simple drawing algorithms for tree and heirarchical graph are applied to the object diagram. We show that our algorithm can produce good layout emphasizing inheritance tree structures. We have also developed an automatic layout system where we can choose and utilize multiple algorithms for drawing object diagrams.

1. はじめに

近年，オブジェクト指向方法論が注目され，それにともない各種の方法論を支援する CASE ツールが数多く開発されている[1),2)]．それらの CASE ツールの多く[3),4)]は，ダイアグラムを効率良く描画するための図形エディタを備えている．一般にソフトウェア開発においてダイアグラムは複数の開発者間のコミュニケーションの道具ともなり，見やすく可読性の高いレイアウトが要求される[1),5)]．しかし，それらの図形エディタでは，ダイアグラムを構成する基本部品をすべてユーザが配置するといった操作環境となっている．そのため，ユーザは見やすいレイアウトをつねに考慮しながら配置していく必要があり，ダイアグラムが複雑化するに従って作図に手間がかかる．そこで本研究では，OMT 法[6)]で用いるオブジェクト図を対象とし，自動描画のためのアルゴリズムを提案する．オブジェクト図の描画では，継承の木構造を分かりやすく示すことが重要と考えられる．しかし，オブジェクト図のような複数の木構造を含む複雑なグラフでは，従来の描画アルゴリズムでは木構造を分かりやすく描くのは困難であった．そこで，我々はオブジェクト図を2段の階層グラフと見なして，木描画アルゴリズムと無向グラフ描画アルゴリズムを適用することにより，継承の構造を強調したレイアウトを得るアルゴリズムを考案した．

また，従来の CASE ツールにもレイアウトアルゴリズムを備えたもの[3),4),7)]は存在するが，いずれも単一のレイアウトアルゴリズムがツールに組み込まれた形で実装されているのみである．最近ではレイアウトアルゴリズムのライブラリ[8)]の開発が進み，優れたアルゴリズムも提案されつつあるため，ツールに対して新たなアルゴリズムを容易に追加できる環境が望ましい．しかし，従来のツールのようなアルゴリズム内臓型の実装では新たなアルゴリズムの実装は困難である．そこで我々は，複数のアルゴリズムの実装の容易化お

† 筑波大学理工学研究科
Master's Program in Science and Engineering, University of Tsukuba
†† 筑波大学電子情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

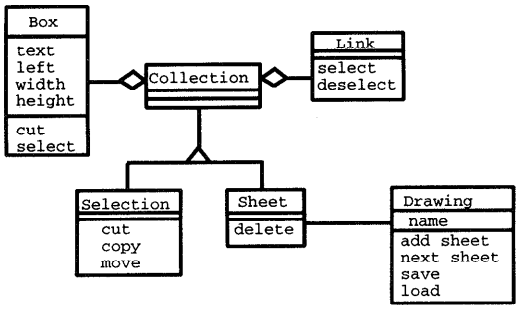


図1 オブジェクト図
Fig. 1 Object diagram.

よびアルゴリズムを選択・利用できる操作環境の構築を目的として，オブジェクト図レイアウトアルゴリズムを実装した自動レイアウトシステムを開発した[9]．

2. オブジェクト指向方法論 OMT

OMT 法[6]は，図式表記の充実度や具体的な作業プロセスの明記といった点から，特に広く使用されている方法論である．OMT 法ではシステムのモデルとして以下の 3 種類のモデルを用いる．

オブジェクトモデル システムの静的な構造を表現．
動的モデル システムの制御構造を表現．
機能モデル システムの計算に関する構造を表現．

3 種類のモデルの中でも特にソフトウェア開発の中心となるのがオブジェクトモデルである．オブジェクトモデルは**図 1** に示すようなオブジェクト図を用いて表される．

本研究では，オブジェクト図中のクラスをノード，クラス間の関係をエッジとしてとらえ，グラフ描画アルゴリズムを適用した．

3. グラフ描画アルゴリズム

各種のダイアグラムを自動生成するための基本技術として，美しさや可読性を考慮してグラフの自動配置を行うグラフ描画アルゴリズムが数多く提案されている[10),11)]．グラフ描画アルゴリズムにおいて描画対象となるグラフは，木，有向グラフ，無向グラフなどに分類される．各グラフによって美的基準が大きく異なるため，グラフ描画アルゴリズムは通常ある特定のグラフを対象として開発される．

現在グラフ描画アルゴリズムは様々な分野で応用されている．ソフトウェア工学の分野では，実体関連図[12)]やデータフロー図[13)]の自動描画などの応用例がある．

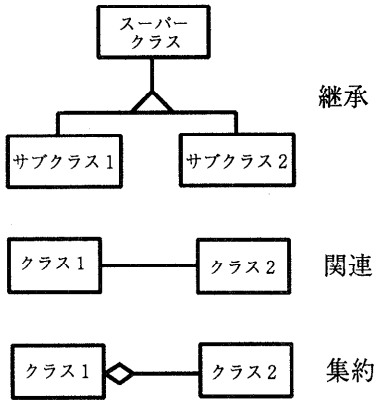


図2 オブジェクト図の表記法
Fig. 2 Description method of object diagrams.

4. オブジェクト図レイアウト手法

オブジェクト図で扱うクラス間の関係には，継承，関連，集約がある．中でも継承は，クラスの持つ属性と操作を知るには親クラスをたどる必要があるため，クラスの親子関係を明確に示すことが重要である．

しかし，オブジェクト図は複数の木構造を含む複雑なグラフであり，従来の描画アルゴリズムでは，木構造を分かりやすく描くのは困難であった．そこで，我々はオブジェクト図を 2 段の階層グラフと見なすことで，継承の木構造を強調して描く自動レイアウト手法を考案し実装した．以下では，このオブジェクト図自動レイアウト手法について述べる．

4.1 オブジェクト図のグラフ構造

グラフ描画アルゴリズムにおいて描画対象となるグラフは，一般に有向グラフ，無向グラフ，木などに分類され，それぞれ数多くのアルゴリズムが開発されている[10)]．OMT 法で用いられる主なダイアグラムのうち，状態遷移図，イベントトレース図，データフロー図はいずれも比較的単純なグラフ構造で表されるため，既存の有向グラフまたは無向グラフ描画アルゴリズムの適用が容易である．一方，オブジェクト図では，継承の関係にあるクラスは一般に木構造として描画されるため，このような描画規則を考慮すると他のダイアグラムに比べて非常に複雑なグラフ構造となる．**図 2** にオブジェクト図の主要な表記法を示す．オブジェクト図ではクラス間の関係には継承のほかに，関連（オブジェクト間の参照，利用関係），集約（オブジェクト間の全体–部分関係）がある．そのためオブジェクト図は 3 種類のエッジを持つグラフといえる．図 2 の表記法に従って描かれるオブジェクト図の一般的なグラフ構造を**図 3** に示す．図 3 に示したように，継承の

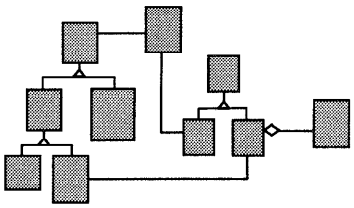


図3 オブジェクト図のグラフ構造
Fig. 3 Graph structure of object diagrams.

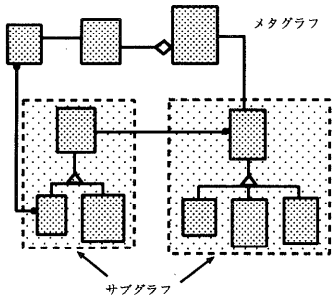


図4 オブジェクト図の階層表現
Fig. 4 Hierarchical structure of object diagrams.

関係にあるクラスは木構造で表されるため，オブジェクト図は1つのグラフ内に複数の木構造が含まれる．

また，グラフを構成する3種類のエッジは，継承と集約は有向エッジ，関連は無向エッジであると見なされるため，オブジェクト図を構成するグラフは，複数の木構造を含有し，有向エッジと無向エッジの混在するグラフとなる．

4.2 オブジェクト図の2段階の描画法

先に述べたように，オブジェクト図は複数の木構造を含有し，有向エッジと無向エッジが混在する特殊なグラフとして表されるため，既存の木描画アルゴリズムや有向グラフ描画アルゴリズムでは，継承木を反映させたレイアウトを得るのは困難である．そこで本研究では，継承木を反映させたレイアウトを得るために，オブジェクト図を図4に示すような2段の階層構造としてとらえ，継承木に木描画アルゴリズムを適用した．1つのグラフに異なる2種類のアルゴリズムを適用する手段として，グラフを2段の階層として扱った．図5に我々の提案するオブジェクト図レイアウト手法を示す．

本手法は，サブグラフのレイアウト，メタグラフのレイアウト，ルーティング（エッジの配線）の3工程から成る．木構造となるサブグラフにはWalkerの木描画アルゴリズム，メタグラフには無向グラフ描画アルゴリズムであるEadesスプリングモデル，エッジ配線には2点間を結ぶ経路を求める線分探索法をそれぞれ用いた．以下の4.3，4.4，4.5節では，各行程で用いるアルゴリズムについてそれぞれ述べる．

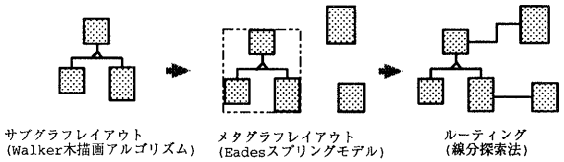


図5 オブジェクト図レイアウト手法
Fig. 5 Layout method for object diagrams.

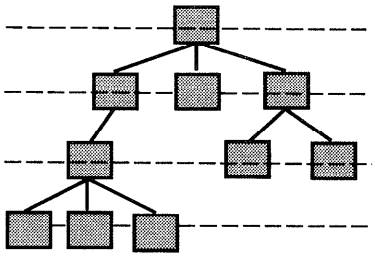


図6 階層に対応した平行線上へのノードの配置
Fig. 6 Node placement on parallel lines corrsponding to hierarchy.

4.3 サブグラフレイアウト手法

我々の提案するオブジェクト図レイアウト手法では，継承木をサブグラフとして扱う．木描画アルゴリズムには様々なものが提案されているが，現在最も優れているといわれるものにWalker[14]のアルゴリズムがある．そこで，本研究ではサブグラフのレイアウトには，このWalkerのアルゴリズムを用いた．このアルゴリズムでは，木の根となるノードは描画の最上の位置に配置され，各ノードはその階層に対応した平行線上に，あらかじめ与えられた順序に従い左から右に並べられる（図6）．ノードの配置には以下のような描画規則が用いられる．

- 親ノードは子ノードの中央に配置される．
- 同形な部分木は同一の描画とし，対象なものは鏡像とする．
- 描画幅は最小とする．

4.4 メタグラフレイアウト手法

無向グラフ描画アルゴリズムには様々なものがある．メタグラフではノード数が多くなる可能性が大きいため，我々は効率の面で優れているEadesスプリングモデル[15]を用いた．ここではEadesのスプリングモデルとそれに対して行った拡張について述べる．

4.4.1 Eadesスプリングモデル

Eadesのスプリングモデルは，力学的シミュレーションを用いてグラフを配置する力指向アプローチと呼ばれる無向グラフ描画アルゴリズム[16),17)]の1つで

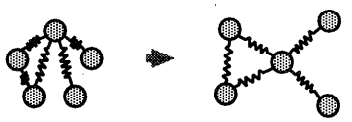


図7 スプリングモデル
Fig. 7 Spring model.

ある．Eadesのモデルでは，ノードは鉄のリングに，エッジは力学系を形成するバネに例えられる．バネによって連結される2つのノードは，ある一定距離以上近づくと斥力が働き遠ざけられる．逆に一定距離以上遠ざかると引力が働き近づけられる．また，バネによる力以外にも，すべてのノード間に斥力が定義される．スプリングモデルではノードにある初期状態を与え，各ノードをバネによる力に従って移動させる．以下のアルゴリズムに示されるように，各ノードはイタレーションにより少しずつノード間に働く力を緩和する方向に動かされる．

```
algorithm SPRING(graph:G);
  Gのノードをランダムな位置に配置;
  repeat M times
    各ノードに働く力を計算;
    各ノードをC*(そのノードに働く力)だけ移動;
  Gを描画;                    /* M,Cは定数 */
```

結果として得られる配置は初期配置に依存するため，スプリングモデルでは初期配置の与え方により得られるレイアウト結果が異なることがある．Eadesのアルゴリズムでは初期配置はランダムに決定しているが，本研究では鈴木ら[17]のアルゴリズムに習い，初期配置にはノードを円周上に配置する手法を用いた（図7）．Eadesのスプリングモデルではノードはすべて無限小の点として表されるため，本研究では任意の大きさのノードを扱うための拡張手法[18]を用いてオブジェクト図に適用した．

4.5 ルーティング

ルーティングとは従来VLSI設計などで使用されている配線手法である[19),20]．最も単純で拡張の容易なルーディングアルゴリズムとして，線分探索法がある．本研究では，この線分探索法[21),22]をグラフのエッジ配線に用いた．

5. 自動レイアウトシステム

我々は，前述したレイアウト手法を用いて，オブジェクト図を自動描画するためのシステムを開発した．以下では，本研究と関連のある自動レイアウト技術と，我々の開発したシステムについて述べる．

5.1 既存の自動レイアウト技術

ソフトウェア開発で用いるダイアグラムを対象とした自動レイアウト手法に関する研究には，グラフ描画アルゴリズムによるデータフロー図のレイアウト手法[13)]や，実体関連図のレイアウト手法[12)]がすでに提案されている．これらのアルゴリズムは階層構造を持たない単純なグラフを対象としている．オブジェクト指向方法論に基づくダイアグラムでは，イベントトレース図を対象としたアルゴリズムが提案されている[23)]．オブジェクト図については，遺伝アルゴリズムを用いたレイアウト手法が提案されている[24),25)]．対象とするオブジェクト図は原田ら[26)]の提案するSOMMの表記に基づくものであるが，グラフ配置に用いられる描画規則は比較的単純なものであるため，OMT法に基づくオブジェクト図のような木構造を含む複雑なグラフには適さない．

様々なダイアグラムを対象にアルゴリズムが開発されているが，一部の単純なグラフに関しては，Rational Rose[3)]，ReTree-C++[4)]，Graphical Desinger[7)]など，すでに商用のCASEツールに実用化されている．これらのツールでは，エディタ上に表示されているダイアグラムを見やすく再配置するためや，プログラムから設計情報を抽出しオブジェクト図を逆生成する機能などに，単純な有向グラフ描画アルゴリズムを採り入れている．

5.2 システム構成

本システムは，オブジェクト指向分析やプログラムからのリバースエンジニアリングによって得られたモデル要素（クラス，属性，関連，イベントなど）を基に，オブジェクト図の自動描画を行うシステムである．本システムでは，レイアウト計算部と，レイアウト結果を描画する表示部を分離した構成とするため，クライアントーサーバ形式で実装した．レイアウト計算部と表示部を分離することで，以下のような利点があるといえる．

- 表示部にはアルゴリズムを実装する必要はなく，単に規定のプロトコルに従って利用するアルゴリズムを指定するのみで，レイアウト計算部に用意された様々なアルゴリズムを利用できる．そのため，各種のCASEツール，WWWブラウザ上での表示など，様々なツールから分散環境でアルゴリズムを共用することが容易になると考えられる．
- レイアウト計算部では，アルゴリズムの実装は表示部には依存しないため，新たなアルゴリズムの追加が容易になる．アルゴリズムの実装はブラッ

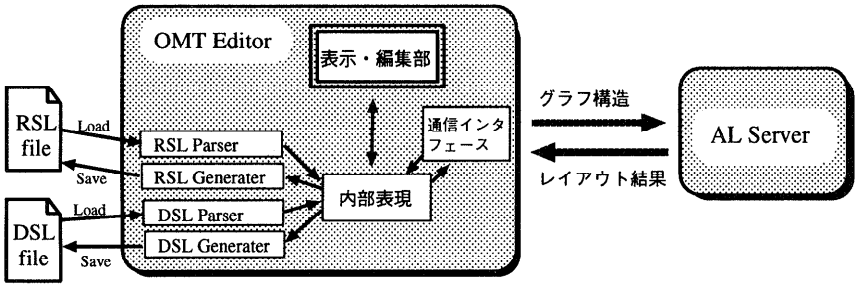


図8 システム構成
Fig. 8 System structure.

クボックス化でき，ユーザは実装の詳細を知らなくてもアルゴリズムを利用できる．そのため，レイアウト計算部では，アルゴリズムを蓄積していき，より多くのアルゴリズムを使用できる環境を提供できると考えられる．

図8にシステム構成を示す．ここで，レイアウト計算部はAL（Automatic Layout）Server，表示部はOMT Editorとして実装している．

AL Serverは，各種のデータ構造が実装されたLEDAライブラリ[27]を用いてC++により実装し，OMT EditorはTcl/Tk[28]により実装した．また，両ツール間の通信にはソケットを用いた．

オブジェクト図のレイアウトはAL Serverにより計算され，その結果がOMT Editor上に表示される．システムの入力は要求記述リスト言語RSLにより記述されたモデル要素である．システムはRSLコードからオブジェクト図のグラフ表現を抽出し，グラフの自動レイアウトを行う．自動レイアウト後のオブジェクト図はOMT Editor上に表示され，ユーザはこのオブジェクト図に対する編集をエディタ上で行う．自動レイアウトおよびユーザによる編集によって得られたオブジェクト図を保存し再度利用するために，設計図記述言語DSLを用いてオブジェクト図をテキスト形式で記述し出力する．DSL表現はRSLにより記述されたオブジェクト図の構成要素に対して位置情報を付加したものとなる．

5.3 システムの入出力形式

本システムでは入力および出力形式には，原田ら[29]によって設計された要求記述リスト言語RSLと設計図記述言語DSLを一部改良し用いた．以下ではRSLとDSLについて述べる．

5.3.1 要求記述リスト言語RSL

RSLはオブジェクト指向分析により得られたモデルを記述するために開発された仕様記述言語である．RSLはオブジェクトモデルおよび動的モデルの記述に用いられる．ここではオブジェクトモデルの記述例を以下に示す．

```
[cls1, クラス, Sequencer]
[cls2, クラス, Action]
[cls3, クラス, Scene]
[att1, 属性, start'time, class(Sequencer)]
[att2, 属性, end'time, class(Sequencer)]
[att3, 属性, text, class(Action)]
[evt1, イベント, start, sender(), receiver(Scene),
  argument(), return()]
[evt2, イベント, tick, sender(), receiver(Scene),
  argument(),return()]
[ass1, 関連,,from(Sequencer), to(Action), mul(+)]
[agg1, 集約, class(Cue), hasA(Scene), mul()]
[inh1, 継承, class(Scene), isA(Sequencer), mul()]
```

RSLではオブジェクトモデルを構成するクラス間の関係や個々のクラスの詳細がリスト形式で記述される．

5.3.2 設計図記述言語DSL

現在種々のCASEツールが開発されているが，ツール間でのダイアグラムの相互利用は困難である．DSLはツール間でダイアグラムを相互利用するための共通表現として提案されたデータ互換用言語であり[29]，OMT法に基づくオブジェクトモデル，動的モデルの記述に用いられる．以下にオブジェクト図の記述例を示す．

```
OOD(g1)[nodes-n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11",
 arcs-a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11"];
OODN(n1)[loc(651:336),size(87:68),oodnAttr(name:Sequencer,
 (dataType:string,name:start'time))];
OODN(n2)[loc(506:442),size(59:51),oodnAttr(name:Action,
 (dataType:string,name:text))];
OODA(a1)[from(n1,side:LEFT,off:34),to(n2,side:TOP,off:29),
 oodaAttr(arcType:assoc,name:(),forwardMult:1,reverseMult:*)];
```

DSLではダイアグラムはノードとエッジからなるグラフとして記述される．DSLはRSLにより記述されるモデル要素に位置情報を付加したものであり，実際のダイアグラムと1対1に対応している．

5.4 RSLからのオブジェクト図生成方式

本システムではOMT Editor上に描かれるオブジェクト図のレイアウトはすべてAL Serverにより計算

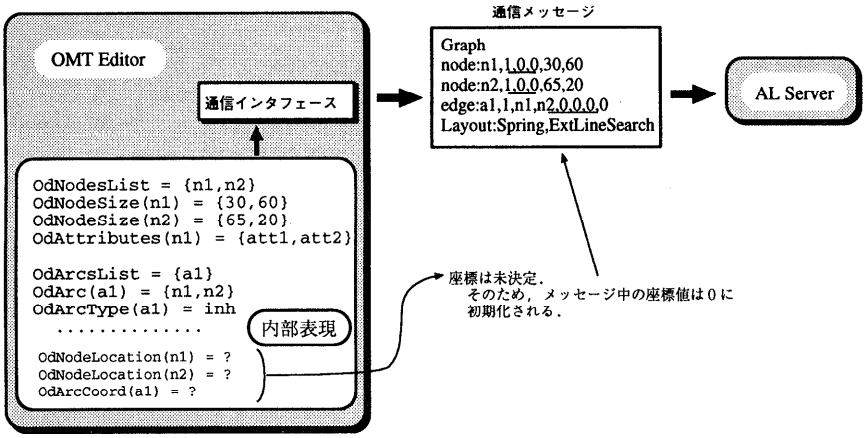


図9 AL Server に対するグラフの送信
Fig. 9 Sending graph to AL Server.

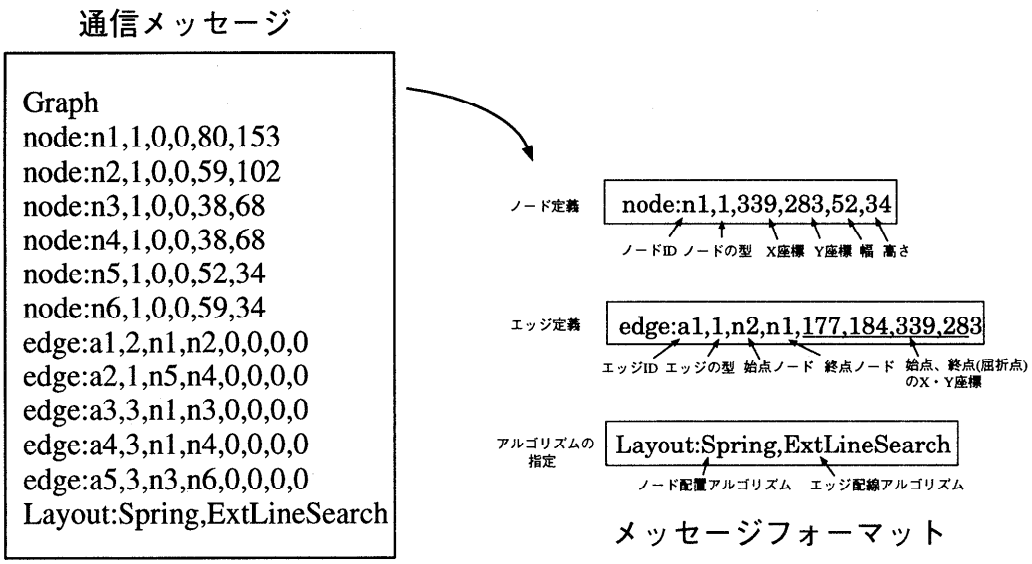


図10 通信メッセージ
Fig. 10 Commnication message.

される．OMT Editor から AL Server へはオブジェクト図のグラフ表現が送信される．AL Server はこのグラフ表現を基にレイアウト計算を行い，レイアウト結果を返信する．OMT Editor から AL Server に対してレイアウト計算の要求が起こるのは以下の2種類の状況においてである．

- RSL 形式ファイルのロード
- ユーザによるオブジェクト図の変更（クラスの追加，消去など）

以下では前者の RSL 形式ファイルからのオブジェクト図の生成について述べ，5.5 節において後者のオブジェクト図のレイアウトの自動修正について述べる．

5.4.1 RSL から内部表現への変換

本システムでは RSL 形式ファイルが入力されると，OMT Editor の RSL Parser が RSL 形式ファイルを解析してグラフ構造を抽出し，内部表現として保持する．OMT Editor では内部表現は Tcl のリストおよび連想配列として実装されている（図 **9**）．

5.4.2 AL Server に対するメッセージ送信

OMT Editor はオブジェクト図のグラフ表現を AL Server に送信する．OMT Editor と AL Server 間の通信には図 **10** のような形式のメッセージを用いた．図 10 に示したように，メッセージはノードとエッジに関する定義と利用するレイアウトアルゴリズムの指

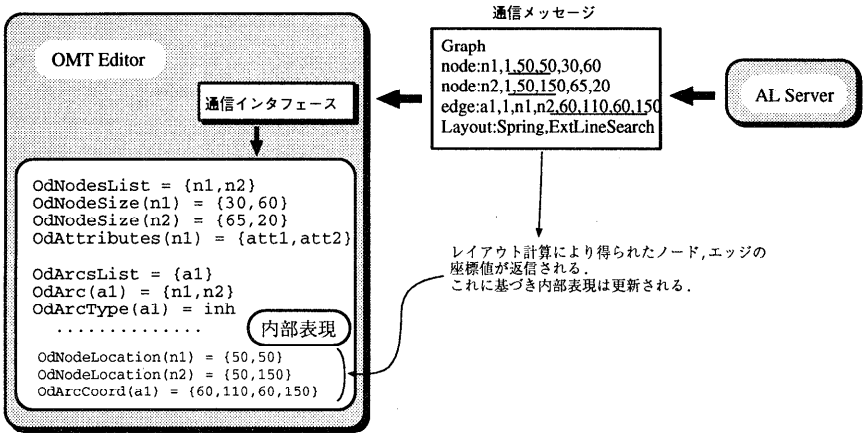


図 11 レイアウト結果の返信
Fig. 11 Reply of layouting result.

定から構成される．メッセージにより AL Server に送信されるレイアウト計算に必要な情報は，個々のノードの ID，型，X 座標，Y 座標，幅，高さ，個々のエッジの ID，型，始点ノード，終点ノード，端点（屈折点）の座標である．また，メッセージにはレイアウトアルゴリズムの指定が含まれる．図 9 に示すように，RSL 形式ファイルを RSL Parser が解析した段階では，ノードおよびエッジの座標値は未決定の状態である．そのため AL Server に対するメッセージでは，これらの座標値に関してはすべて初期値 0 に設定している．すでに OMT Editor 上にオブジェクト図が描画されており，そのオブジェクト図の再レイアウトを行う場合は，メッセージには現在配置されているノード，エッジの座標値が設定される．

ダイアグラムのレイアウト計算は，4 章で述べたようにノードの配置とエッジの配線の 2 工程から成る．図 9 のメッセージの例ではノードの配置には Spring という名のアルゴリズム，エッジの配線には ExtLineSearch という名のアルゴリズム（それぞれ 4 章で述べたノード配置アルゴリズムとエッジ配線アルゴリズム）を用いるということを表している．

5.4.3 AL Server からのレイアウト結果の返信

AL Server は OMT Editor から送信されたメッセージを基に，レイアウト計算を行い，**図 11** のように送信されたメッセージと同様の形式でレイアウト結果を OMT Editor に返す．AL Server からのメッセージに基づき OMT Editor の内部表現は更新され，表示・編集部にオブジェクト図が描画される．**図 12** にシステム実行画面（OMT Editor 表示・編集部）を示す．表示されているオブジェクト図は RSL 形式ファイルから自動レイアウト機能によって得られたものである．

5.5 オブジェクト図のレイアウト自動修正

本システムでは，クラスの追加や消去などオブジェクト図の書き換えをともなう操作を実行すると，変更前の位置情報を基に新たなレイアウトが計算されオブジェクト図は再配置される．AL Server に対するメッセージの送信は，そのような操作の実行がトリガとなり行われる．また，OMT Editor では AL Server に実装されているアルゴリズム群から起動するアルゴリズムを選択し設定することができる．以下では，レイアウトの自動修正に用いるアルゴリズムの選択機構について述べる．

5.5.1 アルゴリズム選択機構

ユーザが複数のアルゴリズム群から好みのアルゴリズムを選択し利用できるような操作環境を目指し，本システムでは**図 13** に示すようなアルゴリズム選択機構を用いた．

前述したように，通信メッセージには利用するアルゴリズムの指定が含まれるため，OMT Editor から AL Server に実装されたアルゴリズムを選択することができる（本研究で作成したプロトタイプでは，AL Server に実装されているアルゴリズムは，ノード配置アルゴリズムが 2 種類，エッジ配線アルゴリズムが 2 種類である）．OMT Editor 上では図 13 のようなダイアログボックスによりアルゴリズムの選択・設定を行う．OMT Editor は AL Server と接続されると利用可能アルゴリズムの問合せを最初に行い，AL Server が返すアルゴリズムのリストをダイアログボックス下段のリストボックスに表示する．ユーザはこのリストボックスから利用するアルゴリズムを選択する．本システムではノードの追加，消去，移動などのダイアグラムの書き換えをともなう操作に対して，それらの操

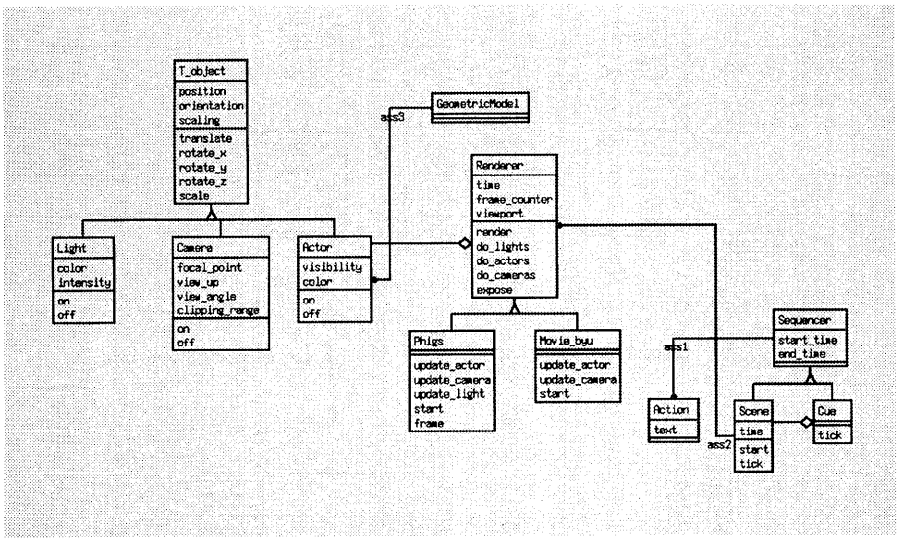


図12 自動レイアウト結果
Fig. 12 Automatic layout result.

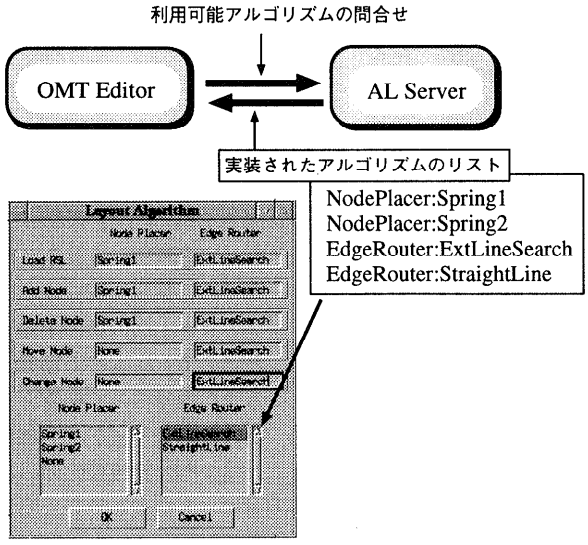


図13 アルゴリズム選択機構
Fig. 13 Algorithm selection mechanism.

作が実行されたとき起動されるアルゴリズムをそれぞれ別に設定することができる．

5.5.2 ノード追加にともなうレイアウト自動修正

ここではオブジェクト図に対するクラスの追加を例にあげ，アルゴリズム選択機構に基づくレイアウトの自動修正について述べる．ノードの追加☆では，終点となるノードの指定（マウスによるクリック）がALServerに対するメッセージ送信のトリガとなり，新たなノードとエッジを含むグラフがAL Serverに送信される．AL Serverでは指定されたアルゴリズムを用いて新たなレイアウトを計算し，OMT Editorはその計算結果を基にオブジェクト図を再配置する．**図14**にレイアウトの自動修正の実行例を示す．ノード配置アルゴリズムの指定がNoneである場合はノードは再配置されない．そのため図14の例（左下の実行結果）では，レイアウト前と後では各ノードの座標は同一であり，エッジのみ再配線されている．

☆ ここではノードの追加とは，新たなノードを生成して他のノードとの間に新たなエッジを生成する操作を指す．

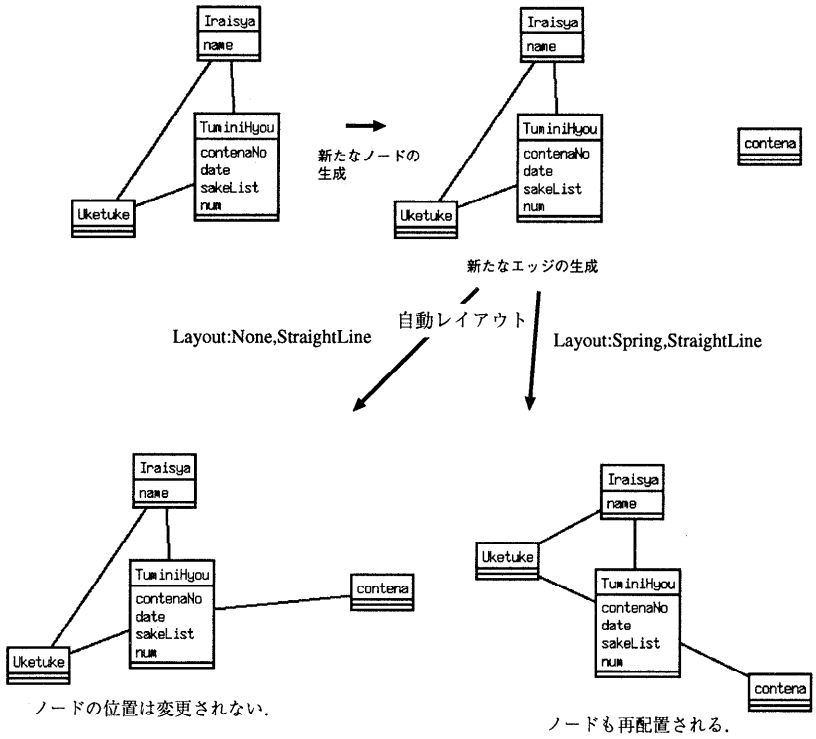


図14 ノードの追加
Fig. 14 Adding nodes.

6. 評 価

本研究では，自動レイアウト機能を持つ既存のCASEツールを対象に，我々の開発したシステムとの比較実験を行った．比較対象として市販のツールであるRational Rose[3]を用いた．以下では，Rational Roseとの比較実験について，実験方法，結果と考察の順に述べる．

6.1 実験方法

本研究では，以下のような2種類の比較実験を行った．

比較実験1 マウスによる直接操作のみの従来型の描画法と，直接操作と自動レイアウト機能を併用する描画法との比較．

比較実験2 我々の提案するアルゴリズムと既存のアルゴリズムのレイアウト結果の比較．

両実験とも5人の被験者に対する課題として，以下のような同規模の3種類のオブジェクト図について，テキスト表現（RSL形式）を用意した．

課題1 ノード数10，エッジ数14のオブジェクト図
課題2 ノード数14，エッジ数14のオブジェクト図
課題3 ノード数15，エッジ数13のオブジェクト図

比較実験1では，被験者に3種類のオブジェクト図をRSL形式を見ながら一から描くといった作業を行ってもらい，それぞれのツールにおいて描き終わるまでの作業量を測定する．この作業量が少ないほどオブジェクト図の描画を効率的に行うことができているといえる．本実験ではRational Roseではマウスによる直接操作のみで描画を行い，OMT Editorでは直接操作と自動レイアウト機能を併用して描画を行う．これにより，直接操作のみの従来型の描画法と，自動レイアウト機能を用いた描画法を比較する．また比較実験2では，両ツールにおいて被験者に3種類のオブジェクト図の自動レイアウト結果を与え，被験者が満足する（被験者が理解しやすい）レイアウトとなるようにオブジェクト図を修正する．この修正に費やした作業量を測定する．この作業量が少ないものほど，良いレイアウト結果が得られていると考えることができる．両実験における作業量として，以下の3項目についてデータ収集を行った．

操作時間 オブジェクト図描画に費やした時間
移動回数 ノードおよびエッジを移動した回数
移動距離 ノードおよびエッジを移動した距離

6.2 実験結果と考察

実験結果を**図15**，**図16**に示す．図中の1，2，3は

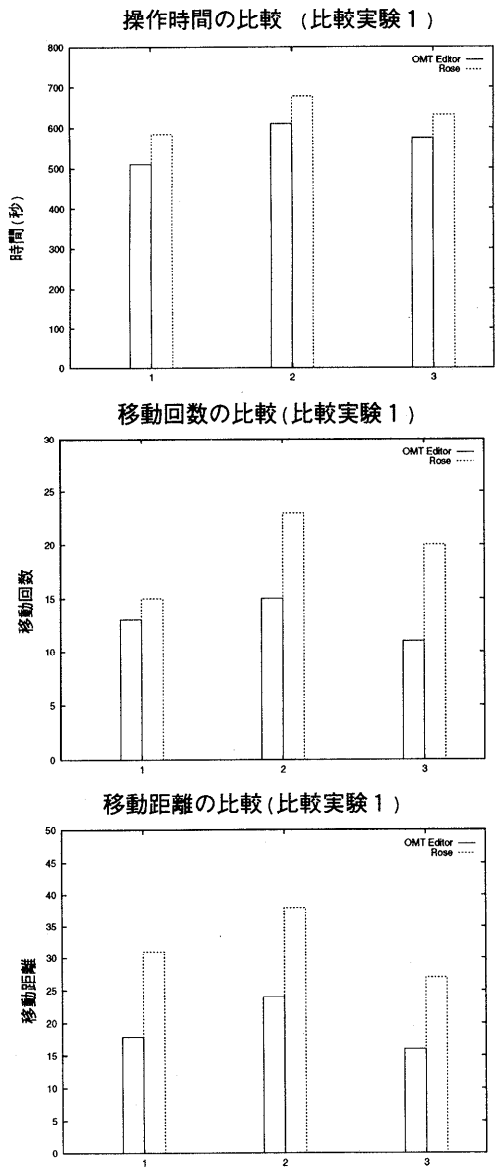


図15 実験結果(実験1)
Fig. 15 Experiment result (Experiment 1).

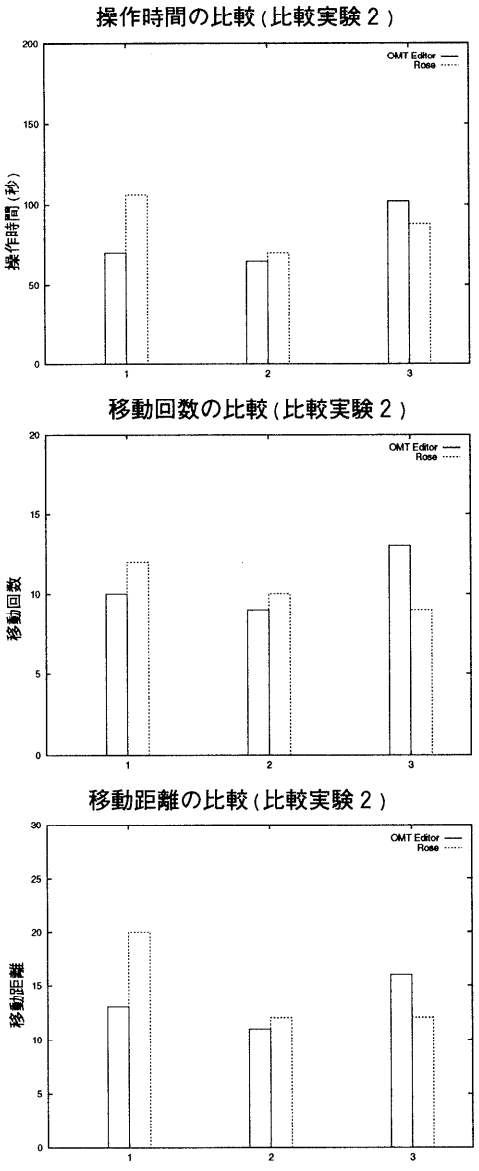


図16 実験結果(実験2)
Fig. 16 Experiment result (Experiment 2).

それぞれ課題1，課題2，課題3のオブジェクト図についての値を表す．3種類のグラフの示す値は，5人の被験者の測定値をそれぞれ平均したものである（移動距離の測定では100ドットを距離1とする単位を用いている）．

比較実験1では，3種類のオブジェクト図のいずれに対しても，操作時間，移動回数，移動距離に関してOMT Editorの方がRational Roseに比べて優れた結果が得られた．これにより，手動と自動レイアウト機能を併用することができる操作環境が，手動のみの描画に比べ効率的であり，自動レイアウト機能が有効であることが確認できた．

比較実験2では，3種類のオブジェクト図のうち1つを除いて，OMT Editorの方が操作時間が短くなっている．また，操作時間が短いものは，いずれも移動回数と移動距離も小さな値となっている．逆の傾向を示すオブジェクト図も1つあることから，対象とするオブジェクト図により優劣が異なっていることが分かる．我々のアルゴリズムでは，商用のツールで用いられているアルゴリズムと同等以上のレイアウトが得られるといえる．

本実験は，3つの課題について，5人の被験者に対

して行ったものである．各被験者はオブジェクト図中の各クラスの意味は理解しておらず，美的基準により描画を行っている．より詳細な評価のためには，測定項目や比較対象を増やすことや，オブジェクト図中のクラスの意味を深く理解している開発者を被験者として，分析する必要があると思われる．しかしながら，今回のレベルの評価でも，レイアウトアルゴリズムの優位性を示すことができたと考えられる．

7. おわりに

本研究では，グラフ描画アルゴリズムを用いたOMT法に基づくオブジェクト図のレイアウトアルゴリズムを提案した．また，そのアルゴリズムを実装したALServerと作図を行うOMT Editorからなる自動レイアウトシステムを開発し，複数のレイアウトアルゴリズムからユーザが好みのアルゴリズムを選択・利用できるような操作環境を実現した．

我々の自動レイアウトシステムと商用のCASEツールであるRational Roseの自動レイアウト機能との比較実験を行った．その結果，直接操作と自動レイアウト機能の双方を併用することができる我々のシステムの操作環境は，ダイアグラムの描画に有効であることが確認できた．

今回は，オブジェクト図の持つ継承関係に着目したレイアウト手法を提案した．今後の課題として，継承だけでなく関連や集約関係にも着目し，レイアウト手法を改良することがあげられる．また，グラフ描画の美的基準とクラス構造を理解することの関係についても，今後さらに検討していく必要がある．なお，本研究で開発したシステムは，青山学院大学で現在使用していただいている．今後多くのユーザを対象にさらなる評価を行いたいと考えている．

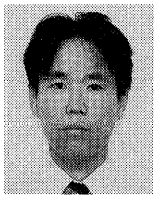
謝辞 本研究に関連して日頃から助言をいただく原田実，伊藤潔，大西淳らの諸氏に感謝いたします．

参考文献

1) Martin, J.（著），竹林則彦（監修）：オブジェクト指向CASE技法：OOIEコンセプト，トッパン (1995).
2) 本位田真一，青山幹雄，深澤良彰，中谷多哉子：オブジェクト指向分析・設計開発現場に見る実践の秘訣，共立出版 (1995).
3) http://www.rational.com/pst/products/rosecpp.html.
4) http://www.dur.ac.uk/dcs3pjb/re3-cpp.html.
5) Martin, J.（著），松山一朗（訳）：自動システム設計のための標準ダイヤグラム作成技法，近代科学社 (1991).
6) Rumbaugh, J., et al.（著），羽生田栄一（監訳）：オブジェクト指向方法論OMT：モデル化と設計，トッパン (1992).
7) http://davinci2.csn.net/jefscot/gdover.html.
8) Friedrich, C.: *The ffgraph Library*, University Passou (1995).
9) 中島 哲：オブジェクト指向方法論に基づくダイヤグラムの自動レイアウト，筑波大学修士論文 (1997).
10) Battista, G.D., Eades, P., Tamassia, R. and Tollis, I.G.: Algorithms for Drawing Graphs: an Annotated Bibliography, ftp.cs.brown/pub/papers/compgeo/gdbiblio.ps.Z (1994).
11) Ding, C. and Mateti, P.: A Framework for the Automated Drawing of Data Structure Diagrams, *IEEE Trans. Softw. Eng.*, Vol.16, No.5, pp.543–557 (1990).
12) Batini, C., Talamo, M. and Tamassia, R.: Computer Aided Layout of Entity-Relationship Diagrams, *The Journal of Systems and Software*, Vol.4, pp.163–173 (1984).
13) Batini, C., Nardeli, E. and Tamassia, R.: A Layout Algorithm for Data Flow Diagram, *IEEE Trans. Softw. Eng.*, Vol.SE-12, No.4, pp.538–546 (1986).
14) Walker, J.Q.: A Node-positioning Algorithm for General Trees, *Software Practice and Experience*, Vol.20, No.7, pp.685–705 (1990).
15) Eades, P.: A Heuristics for Graph Drawing, *Congressus Numerantium*, Vol.42, pp.149–160 (1984).
16) Fruchterman, T. and Reingold, E.: Graph Drawing by Force-Directed Placement, *Software Practice and Experience*, Vol.21-11, pp.1129–1164 (1991).
17) 鈴木和彦，鎌田富久，榎本彦衛：単純無向グラフ描画アルゴリズム，コンピュータソフトウェア，Vol.12, No.4, pp.45–55 (1994).
18) 小野寺秀俊，栗原俊彦，田丸啓吉：力学的モデルに基づくブロック配置手法，信学技報，Vol.CAS86-194, pp.47–54 (1986).
19) Sherman, A.T.: *VLSI placement and routing: the PI project*, Springer-Verlag (1989).
20) Aoudja, F., Laborie, M. and Saint-Paul, A.: CASE: Automatic Generation of Electrical Diagrams, *Computer Aided Design*, Vol.18, No.7, pp.356–360 (1986).
21) Muroga, S.（著），渡辺 誠（監訳）：VLSIシステム設計，ワイリー・ジャパン (1984).
22) 可児賢二，川西 宏，船津重宏：超LSI CADの基礎，オーム社 (1983).
23) 原田 実：SOMEにおける動的モデリングの詳細化と設計図の自動レイアウト CAMEO/Dと

C++プログラムからの設計図の逆生成 OORE, 情報処理学会オブジェクト指向'96 シンポジウム論文集, pp.111–118 (1996).
24) 原田 実, 高橋史郎：オブジェクト指向分析によるモデル要素集合からの遺伝アルゴリズムを用いた設計図作成支援システムの研究開発, 人工知能学会第9回全国大会, pp.363–366 (1995).
25) 原田 実, 高橋史郎：オブジェクト指向分析による設計要素集合からの自動レイアウトによる設計図の作成研究, EAGL 戦略助成研究成果報告書, EAGL 事業推進機構 (1996).
26) 原田 実, 澤田隆志, 藤澤照忠：構造化オブジェクトモデリング環境 SOME-SOMM に基づく OOA/OOD, 情報処理学会ソフトウェア工学研究会資料, Vol.94-SE-101, pp.1–8 (1994).
27) Mehlhorn, K., Naher, S. and Uhrig, C.: The LEDA User Manual Version R3.4.1.
28) Ousterhout, J.O.: *Tcl and Tk Toolkit*, Addison-Wesley (1994).
29) 原田 実, 伊藤 潔, 田中二郎, 大西 淳：オブジェクト指向開発の総合的な自動化をねらう HITO プロジェクト（全体総括）, EAGL 戦略助成研究成果報告書, EAGL 事業推進機構 (1996).

(平成 9 年 3 月 28 日受付)
(平成 10 年 9 月 7 日採録)



中島 哲（正会員）

1972 年生. 1995 年山梨大学工学部電子情報工学科卒業. 1997 年筑波大学大学院理工学研究科修士課程修了. 現在,（株）富士通研究所に所属. ソフトウェア開発環境, ヒューマンインタフェース等に興味を持つ. 日本ソフトウェア科学会会員.

田中 二郎（正会員）

1951 年生. 1975 年東京大学理学部卒業. 1978 年から米国ユタ大学計算機科学科博士課程に留学. Ph.D. in Computer Science. 新世代コンピュータ技術開発機構および富士通研究所を経て, 1993 年より筑波大学電子・情報工学系助教授. 研究領域はビジュアルプログラミング, インタラクティブコンピュテーション, ヒューマンインタフェース. 最近はオブジェクト指向に基づくソフトウェアの設計論に興味を持っている. 1998 年 7 月に開催された本学会主催の Asia Pacific Human Computer Interaction 1988（APCHI'98）ではプログラム委員長をつとめた. 本学会グループウェア研究会連絡委員. ACM, IEEE Computer Society, 日本ソフトウェア科学会, 電子情報通信学会, 人工知能学会, 計測自動制御学会各会員.