平成17年度

筑波大学第三学群情報学類

卒業研究論文

題目
履歴レイヤー構造を持つ知識創造支援ツールの研究

主専攻　　情報科学主専攻

著者　　中園 長新

指導教員　田中二郎　三末和男　志築文太郎　高橋伸

要　旨

知識創造支援ツールなどでアイデアを扱う際、履歴が記録されているとアイデアがいつどのような経緯で産み出されたかが把握できて便利である。一般に思考過程の履歴を振り返ることは、アイデアの更なる発展に寄与すると考えられる。本研究ではこの履歴管理に「レイヤー構造」という概念を導入した。アイデア断片の作成、修正などのイベントを時系列の概念に基づくレイヤーで管理することによって、任意の時点におけるアイデアの状態を振り返ったり、ある期間においてアイデアがどのように発展あるいは整理されていったのかを確認することができる。本研究ではレイヤー構造ネットワークを操作するための描画ツール「NeL^2」を開発し、その応用としてレイヤーによる履歴管理機能を備えた知識創造支援ツール「IdeaCrepe」を開発した。

目 次

図 目 次

表 目 次

第1章　序論

1.1　知識創造活動とその手法

我々は普段の生活において様々なことを思いつき、そして忘れていく。しかしながら一方では、思いついたアイデアの断片を忘れる前に記録したり、その断片を元に発想をふくらませていきたいという欲求がある。そのため、たくさんのアイデア断片を産み出し記録してそれらを発展させていく、いわゆる「発想支援」あるいは「知識創造支援」と呼ばれる活動の技法が多く考案され、また実際に活用されてきた [25]。

我々の知識創造を支援するための技法としては、今日に至るまで数多くの提案がなされてきた。有名なものとしては川喜田二郎の提唱した KJ 法[1][24]、Alex Osborn が創始したブレインストーミング、Tony Buzan の発案による Mind Maps [2]の技法などがあり、広く用いられている。これらの技法の多くは元来、紙とペンによって実施されることを前提にしてきた。

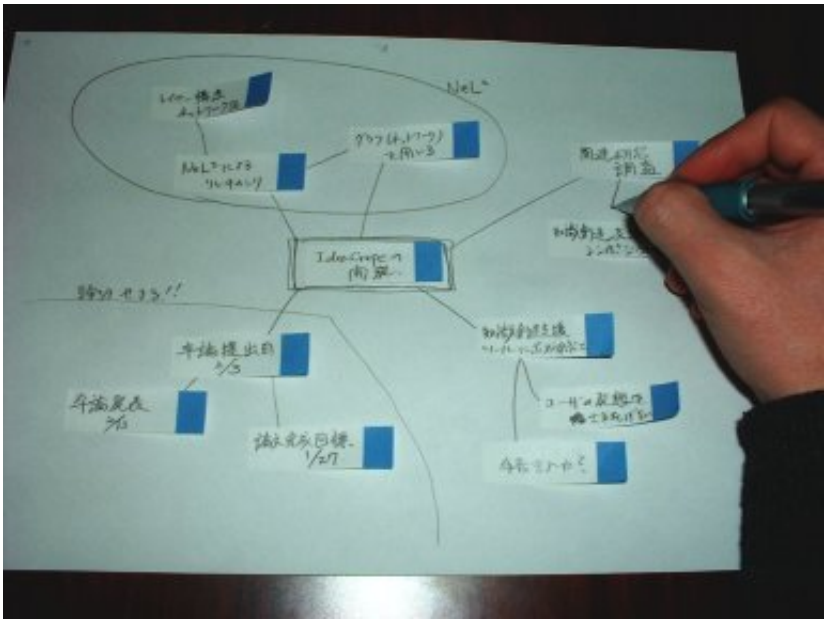

図 1.1: 紙とペンによる知識創造活動の様子

[1]KJ 法は株式会社川喜田研究所の登録商標である。

[2]Mind Maps は Buzan Organisation, Ltd. の米国および英国における登録商標であり、日本では「マインドマップ」と表記される。

実際に紙とペンを用いて行う知識創造活動の様子を図 1.1 に示す。この例では、アイデアの断片は付箋紙に記入されて大きな紙の上に貼られており、付箋紙はペンで描かれた線によって関係づけられている。これらは広い意味でネットワーク図と見なすことができる。このようなネットワーク図は KJ 法 (カードを用いる) や Mind Maps(枝の上にアイデアなどを描く) をはじめとする多くの知識創造技法の基礎に通じるものであるため、本研究ではネットワーク図を用いた知識創造支援ツールに焦点を合わせる。

1.2 履歴の振り返りから新しいアイデアへ

知識創造支援で大切なことは数多く考えられるが、本研究ではその一つとして、過去に行った知識創造活動の振り返りに着目する。「温故知新」という言葉に代表されるように、我々は古来から自分たちの過去を振り返り、その振り返り活動を通して新しい活動における指針を見いだしてきた。これは計算機上で行われる、現代の知識創造活動にも十分に当てはまると考えられる。

多くの知識創造活動は、一つのテーマに対して何回も考察を重ねていくことで行われる。そのため、知識創造活動の成果は日々変化し、成熟に向かっていく。このように時系列に沿って発展していく活動において、アイデアがいつどのような経緯で生まれたのかといった過去の履歴を振り返ることは、今後の活動指針を決める上で重要な要素であると考えられる。過去にどのようなアイデアが生まれ、そのアイデアに対してどのような考察を試みたのか、そしてその結果がどうであったのか。そのような履歴を把握することで、過去に行った活動が新しい活動の原動力となる。

1.3 既存の知識創造支援ツール

近年は計算機の爆発的な普及に伴い、計算機上で電子的に知識創造支援を行う、いわゆる知識創造支援ツール[3]も数多く開発されるようになった。それらは商用のもの [1] から個人が作成したフリーウェア [2, 4]、シェアウェア [5] まで幅広く、様々な特徴で他との差別化を図っているものも多い。たとえば FreeMind[3] は Mind Maps の技法による知識創造支援に特化したツールであり、机上での活動である Mind Mapping を計算機上で実現できる。既存ツールは多くの機能を持ち実用性の高いものが多いが、その一方で過去のアイデアを記録し履歴を保存することは重視されておらず、データは修正を加えるたびに上書きされる仕様になっているツールがほとんどである。

1.4 研究の目的と構想

一般的な多くのアプリケーションで採用されている「上書き保存」の手法は、常に最新版のスナップショットを保存しているといえる。過去の状態を参照したい場合は各時点のデー

[3]発想支援ツール、アイデアプロセッサなどとも呼ばれる

タ (スナップショット) を上書きせずに保存し、各データを比較することが必要となる。しかしながらこのような機能はあまり実装されていない。また、仮に各時点のスナップショットを保存したとしても、それらを比較して変化 (差分) を発見することは容易ではなく、ユーザは複数のスナップショットを同時に閲覧する必要がある。

この問題は知識創造支援ツールにも当てはまる。そこで本研究では、知識創造支援ツールにおいて履歴を管理できるデータ処理の枠組みを提案し、実際に知識創造支援ツールとして実装することを目的とする。

本研究ではまず、開発ツールの基本となるレイヤー構造ネットワークを提案し、実際にシステムを実装する。次にレイヤー構造を用いて履歴管理を行う知識創造支援ツール “IdeaCrepe” を開発し、それぞれのシステムについて評価と考察を行うこととした。

1.5 論文の構成

本論文の構成は次の通りである。本章では知識創造支援における現在の背景と問題点を説明し、研究の目的と構想を述べた。第 2 章ではネットワーク履歴管理のために本研究で提案するレイヤー構造と、その実装である “NeL^2”について説明し、第 3 章で履歴レイヤー構造を用いた知識創造支援ツール “IdeaCrepe”の設計と試作について説明する。第 4 章では実際に NeL^2 と IdeaCrepe を試用し、評価を行う。第 5 章で関連研究を紹介し、第 6 章でまとめる。

第2章　レイヤー構造を備えたネットワークの履歴管理

本章では、レイヤー構造を備えたネットワークの描画と履歴管理を行うツール “NeL^2”[15] について説明する。レイヤー構造ネットワークは知識活動支援ツールのみに特化したものではなく、あらゆるネットワークデータの可視化に応用できる概念である。本章ではレイヤー構造ネットワークを、知識創造支援にとらわれない一般的な見地に立って紹介し、次章で知識創造支援ツールへの応用について述べる。

2.1　レイヤー構造ネットワークの提案

2.1.1　レイヤー構造の概念

本研究では、時間と共に変化するネットワーク図の表現に対してレイヤー構造を導入する。これはネットワーク上のデータを 1 枚の図としてではなく、更新時期の違いによってレイヤー別に管理し、それらの表示や色分けを柔軟に行うことでネットワーク図から多くの情報を読み取ることを可能にするものである。

本研究で扱うネットワーク図は図 2.1 に示すようなレイヤーの概念を用いており、これは OHP シートのような透明なシートを積み重ねたものとして考えることができる。ある期間での更新は 1 枚のシート (レイヤー) に記録され、その次の期間で行われた更新は既存のレイヤーの上に新しいレイヤーを重ねてそれに記録する。それぞれのレイヤーには各期間での更新箇所すなわち差分のみが記録されていくことになり、これらを用いて次のような閲覧操作が可能となる。

- ある時点までのレイヤーを全て積み重ねることで、その時点におけるネットワーク図を見ることができる。レイヤーを順に積み重ねることで過去から未来への変化をたどることができ、逆にレイヤーを順に取り除くことで未来から過去への変化をたどることができる。
- 任意のレイヤーを単独で閲覧することで、その時点におけるネットワークの変化を見ることができる。
- 連続するレイヤーを複数枚重ね合わせることで、ある期間におけるネットワークの変化を見ることができる。

図 2.2, 2.3, 2.4 はレイヤー操作の一例を示している。図 2.2 のネットワーク図に対し、図 2.3 で示されるレイヤーを重ねる。操作後のネットワーク図は図 2.4 のようになり、既存のネットワーク図と新しいレイヤーの情報が積み重ねられていることが確認できる。

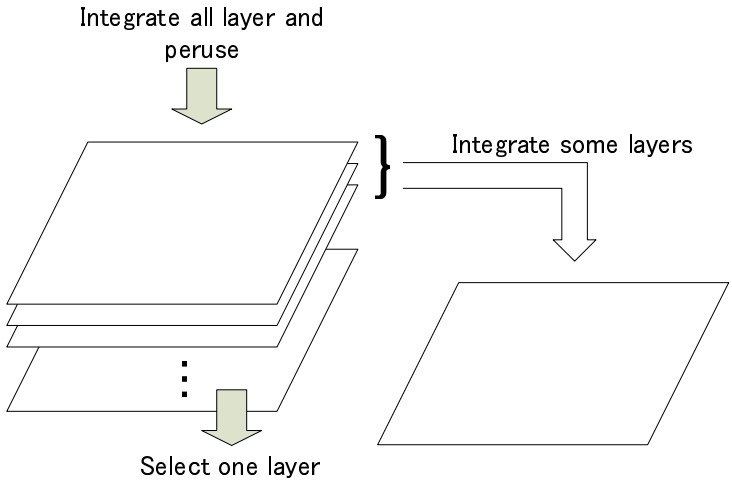

図 2.1: レイヤー構造の概念図

2.1.2 レイヤー構造を用いる利点

レイヤー構造を導入したネットワーク図には、開発者とユーザ双方にとっていくつかの利点がある。

第一に、ユーザにとってネットワーク図の操作が分かりやすく直感的になる。我々は普段から、例えば「書類の束から 10〜16 枚目を抜き出す」といったレイヤー構造とよく似た行動を自然に行っている。レイヤー構造を用いたネットワーク図でも同様に「10 日前から昨日までの変化を取り出す」といった操作が可能である。このような直感的な操作により、ユーザは簡単にネットワーク図を変化させることができる。

第二に、レイヤーを使うことでネットワーク図の差分生成が不要になるということが挙げられる。従来は差分を記録する代わりに、各時点のスナップショットから差分を生成するという手法がとられており、表示のたびに多くの処理を行う必要があった。レイヤー構造ネットワークはデータレベルで差分表現が実現され、レイヤー情報により「差分の集合」として構成されているために、指定されたレイヤーを取り出すという処理だけで差分を表示できる。この特性により、開発者は差分生成の処理を気にすることなくネットワーク図操作をユーザに提供できる。また、差分表示時の処理が減ることで、ユーザにとっては表示時の待ち時間を短縮されることが期待できる。

第三に、ユーザはネットワークの特徴を把握することが容易になる。レイヤーに分割されていないネットワーク図はすべての要素が同時に表示されているため、時間による変化を捉

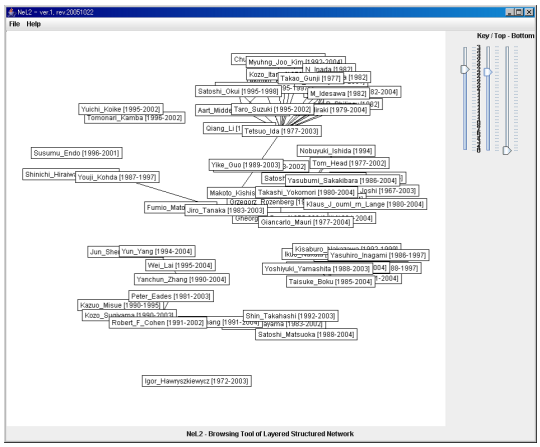

図 2.2: 差分を重ねる前のネットワーク図

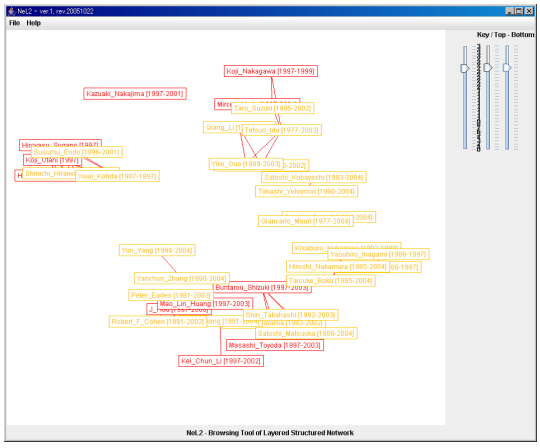

図 2.3: 差分を記録したレイヤーの表示

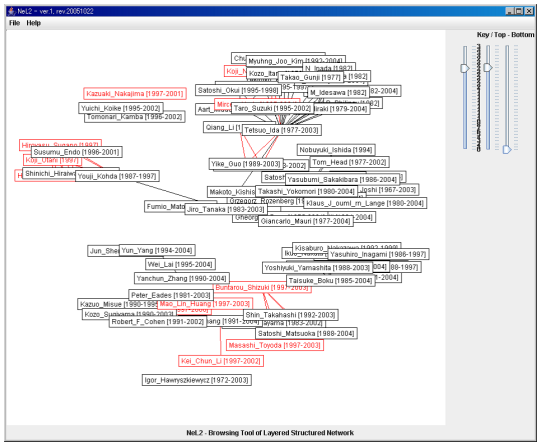

図 2.4: 差分を重ねた後のネットワーク図

えることはできない。レイヤー構造を導入することによってノードやエッジはレイヤーごとに表示を切り替えることができる。この特徴によってユーザは必要とする期間のデータだけに注目することが可能となり、求めている情報を効率よく入手できるようになる。

また、レイヤー構造ネットワークの導入により、開発者はこのような利点を持ったツールを容易に開発可能になる。

2.2 レイヤー構造の数学的定義

本研究で提案するレイヤー構造は、従来のネットワーク図にはない新しい概念であるため、ここで数学的定義を行う。

2.2.1 グラフ、ノード、エッジ

レイヤー構造ではレイヤーが順に積み重ねられ、各時点においてそれぞれネットワーク図として見える。ある時点で見えるネットワーク図はグラフとなり、次のような三つ組で表現することができる。

$$G = (N, E, \varphi)$$

N と E はそれぞれノードの集合とエッジの集合である。関数 $\varphi : E \to N \times N$ はエッジの結合関数であり、$\varphi(e) = (v, w)$ のとき、エッジ e はノード v と w に接続しているとする。

2.2.2 レイヤー

レイヤーは各時点におけるグラフの変化 (差分) を表現する。l_i は下から第 i 番目のレイヤーであり、次のような組で表現する。

$$\begin{aligned} l_i &= (N_i^+, E_i^+, N_i^-, E_i^-, \varphi_i^+) \\ \varphi_i^+ &: E_i^+ \to N' \times N' \\ N' &= (N_{i-1} \cup N_i^+) \backslash N_i^- \end{aligned}$$

N_i^+ と E_i^+ はそれぞれ第 i 番目のレイヤーにおいて追加、N_i^- と E_i^- はそれぞれ削除されたノードとエッジの集合である。ただし、一つのレイヤー上で同じノードまたはエッジについて、追加と削除は同時に起こらないものとする。すなわち、$N_i^+ \cap N_i^- = \phi, E_i^+ \cap E_i^- = \phi$ とする。関数 φ_i^+ は追加されたエッジを定義域とする結合関数である。

2.2.3 レイヤーの積み重ね

既存のグラフにレイヤーを積み重ねることで、新しいグラフが生成される。このようなレイヤーの積み重ねを $\oplus$ で表す。$i-1$ 番目までのレイヤーを重ねて見えるグラフ G_{i-1} の上に次のレイヤー l_i を積み重ねる操作は、次のように表される。

$$G_{i-1} \oplus l_i = G_i = (N_i, E_i, \varphi_i)$$

N_i, E_i, φ_i は次の通りである。ただし G_0 は空のグラフ (ノードもエッジも無いグラフ) とする。

$$\begin{array}{lcl} N_i & = & (N_{i-1} \cup N_i^+) \backslash N_i^- \\ E_i & = & (E_{i-1} \cup E_i^+) \backslash E_i^- \\ \varphi_i & : & E_i \to N_i \times N_i \\ \varphi_i(e) & = & \begin{cases} \varphi_i^+(e) & \text{if } e \in E_i^+ \\ \varphi_{i-1}(e) & \text{if } e \in E_{i-1} \end{cases} \end{array}$$

2.2.4 レイヤーの統合

連続する複数のレイヤーを統合するときも、レイヤーの積み重ねと同様に $\oplus$ で表し、この操作によって一定期間におけるデータを扱ったサブグラフが形成される。連続する 2 枚のレイヤー l_i, l_{i+1} の統合を次のように定義する。

$$l_i \oplus l_{i+1} = (N_i^{+\prime}, E_i^{+\prime}, N_i^{-\prime}, E_i^{-\prime}, \varphi_i^{+\prime})$$

追加および削除されるノード $N_i^{+\prime}$, $N_i^{-\prime}$, エッジ $E_i^{+\prime}$, $E_i^{-\prime}$, および結合関数 $\varphi_i^{+\prime}$ は次の通りである。

$$\begin{array}{lcl} N_i^{+\prime} & = & (N_i^+ \cup N_{i+1}^+) \backslash N_{i+1}^- \\ E_i^{+\prime} & = & (E_i^+ \cup E_{i+1}^+) \backslash E_{i+1}^- \\ N_i^{-\prime} & = & (N_i^- \cup N_{i+1}^-) \backslash N_i^+ \\ E_i^{-\prime} & = & (E_i^- \cup E_{i+1}^-) \backslash E_i^+ \\ \varphi_i^{+\prime} & : & E_i^{+\prime} \to N'' \times N'' \\ N'' & = & (N_{i-1} \cup N_i^+ \cup N_{i+1}^+) \backslash (N_i^- \cup N_{i+1}^-) \\ \varphi_i^{+\prime}(e) & = & \begin{cases} \varphi_i^+(e) & \text{if } e \in E_i^+ \\ \varphi_{i+1}^+(e) & \text{if } e \in E_{i+1}^+ \end{cases} \end{array}$$

2.2.5 レイヤーのビュー

レイヤーを単独で閲覧することでネットワーク図の差分を見ることができる。しかしながら、両端あるいは片端のノードを持たないエッジが存在する可能性があるため、レイヤーは

それ単独ではグラフとして成り立たない。そのため、エッジ両端のノードを補完することでレイヤーのビューを作成する。ビューはグラフである。また、差分表現では追加要素と削除要素がそれぞれ違う色で描かれる。ネットワーク図の差分を表現するための色付きグラフ G^c を、次のような五つ組で表現する。

$$G^c = (N, E, \varphi, \gamma_N, \gamma_E)$$

N, E, φ はグラフ G の定義と同様であり、$\gamma_N : N \to C$ と $\gamma_E : E \to C$ がそれぞれノードとエッジの色を指定する関数である。C は色の集合であるが、ここでは抽象的な色として c_n(変化のないノードの色)、c_a(追加されたノードやエッジの色)、c_d(削除されたノードやエッジの色) の 3 色のみを考える。

レイヤー $l_i = (N_i^+, E_i^+, N_i^-, E_i^-, \varphi_i^+)$ のビューである色付きグラフ $G^c = (N, E, \varphi, \gamma_N, \gamma_E)$ は次のように表される。

$$\begin{array}{lcl}
N & = & N_i^+ \cup N_i^- \cup N^u \\
N^u & = & \{v | (v,w) \in \varphi_i(E) \vee (w,v) \in \varphi_i(E)\} \\
 & & \backslash (N_i^+ \cup N_i^i) \\
E & = & E_i^+ \cup E_i^- \\
\varphi & : & E \to N \times N \\
\varphi(e) & = & \begin{cases} \varphi_i^+(e) & \text{if } e \in E_i^+ \\ \varphi_{i-1}(e) & \text{if } e \in E_i^- \end{cases} \\
\gamma_N(v) & = & \begin{cases} c_n & \text{if } v \in N^u \\ c_a & \text{if } v \in N_i^+ \\ c_d & \text{if } v \in N_i^- \end{cases} \\
\gamma_E(e) & = & \begin{cases} c_a & \text{if } e \in E_i^+ \\ c_d & \text{if } e \in E_i^- \end{cases}
\end{array}$$

N^u はそのレイヤー上で変化のあったエッジに接続するノードのうち、そのレイヤー上で変化のなかったノードの集合である。そのようなノードは N_i^+ や N_i^- には含まれないため、ノードの集合を $N_i^+ \cup N_i^-$ とするだけではグラフを構成できない。グラフ構成に必要なノードを N^u によって補っている。

2.3 ネットワーク描画ツール “NeL[2]”の開発

本研究ではレイヤー構造ネットワーク図を閲覧・操作するためのツール “NeL[2]” [1]を Java (JDK 5.0) を用いて実装した (図 2.5)。

[1]NeL[2] という名称は Networks, Layer, Layout それぞれの頭文字を取ったものであり、ネルスクエアと読む。

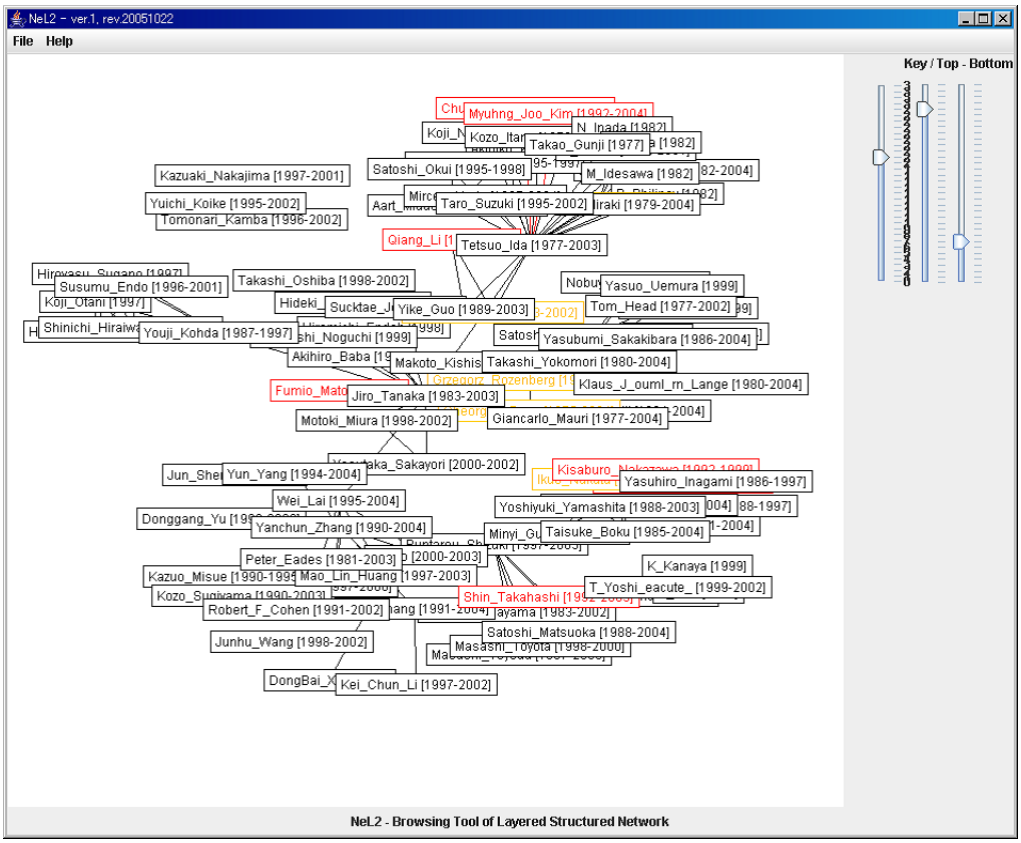

図 2.5: NeL^2 のスクリーンショット

2.3.1 ネットワークのデータ形式

NeL^2 では、ネットワーク図で表現するためのデータを独自の XML 文書として記述する。従来のネットワーク可視化手法が採用しているデータ形式にはレイヤーの概念が存在しないため、レイヤー情報を表現できる新しいデータ形式を考案した。レイヤー構造を持つネットワークデータの記述例を図 2.6 に示す。

`layer` 要素はレイヤーを定義する。ネットワーク図が更新されるたびにレイヤーは追加されるため、追加された日時などをデート属性として ID とともに持つ。`node` 要素はノードを定義し、数学的定義の $n \in N$ に対応する。自分自身の ID を属性として持ち、ラベルと x, y 座標の情報を含んでいる。`edge` 要素はノード同士を繋ぐ関係線 (エッジ) を定義し、数学的定義の $e \in E$ に対応する。属性として自分自身の ID を持ち、両端のノード ID とラベル情報を持つ。両端のノード ID は数学的定義の φ と対応する。実装では便宜を図るため、`edge` 要素の内部で用いる。ノードとエッジの ID は各要素を一意に定めるために導入した。

ネットワーク図が更新されるときにノードやエッジが削除されることがある。このような場合は該当する `node` や `edge` など要素の変更を行う代わりに、`node_event` 、`edge_event` 要素の追加を行う。これらはそれぞれ数学的定義の $n' \in N^-, e' \in E^-$ に相当する。これらの

```
<?xml version="1.0" encoding="UTF-8"?>
<graph>
  <layer id="0" date="1959">
  .....
  </layer>
  .....
  <layer id="12" date="1983">
    <node id="40">
      <label>Jiro_Tanaka</label>
      <position x="411" y="439" />
    </node>
    .....
    <edge id="0">
      <from>36</from>
      <to>1</to>
      <label>1983</label>
    </edge>
    .....
    <node_event id="0" node_id="2" command="del" />
    .....
  </layer>
  .....
</graph>
```

図 2.6: XML によるネットワークの記述例

要素にはノードやエッジに対する操作が記録され、描画時はノードやエッジそのもののデータを上書きする形で用いられる。このデータ形式を用いることにより、削除を行っても過去の状態が保持されるという特長が生まれる。

2.3.2 データの操作体系

XML 文書を読み込んだ直後の初期段階においては、ネットワーク図はレイヤーを全て重ね合わせた状態で表現されており、全てのノードとエッジが表示されている。

表示するレイヤーを変更するためのユーザインターフェースとして我々はスライダーを採用し、直感的に操作が行えるように配慮した (図 2.7)。スライダーは 3 本用意し、右側 2 本のスライダーは表示するレイヤーの範囲 (表示する最も上のレイヤーと最も下のレイヤー) をそれぞれ指定する。左端のスライダーを用いることで、ユーザは注目したいレイヤー (キーレイヤー) を指定でき、ネットワーク図では指定されたキーレイヤーに所属するノードとエッジがハイライト (赤色で表示) される。この機能により、キーレイヤーと他の部分を容易に比較することができる。また、レイヤーのビューによって補完されるノードはオレンジ色で示した。

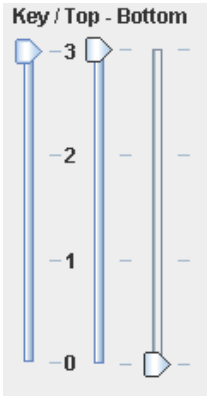

図 2.7: スライダー

2.3.3 利用例

論文共著関係の可視化

レイヤー構造ネットワーク図の利用例として、論文共著関係の可視化への応用を取り上げる。

論文共著関係ネットワークは年々変化しており、多くの研究者が分析を手がけている [6, 18]。一般に、共著関係の分析は数学的手法を用いて定量的に行うが、その前段階として視覚的な図から分析結果を推測することは効果的な活動であるといえる。このような場合にレイヤー構造ネットワーク図は有用である。

ここでは科学論文の共著関係ネットワークを使用した。用いたグラフは WWW 上で公開されている科学論文データベースである DBLP[2]を利用し、Misue[13] と同様の手法で作成した (著者 A と著者 B が共著の論文が一つ以上あれば共著関係 1 と数え、3 名共著の論文が一つあれば共著関係は 3 となる)。

著者をノード、共著関係をエッジとし、例として Jiro Tanaka(本研究の指導教員の一人) を開始ノードとしてエッジにつながっているノードを距離 2 まで探索することで、118 ノード、535 エッジのサブグラフを抽出した。このデータをもとに別のツールを用いてレイアウトを計算しておき、Ruby で記述したスクリプトで文字列処理を行って NeL[2] が読み込める形式の XML 文書に整形した。

レイヤー構造による可視化

このデータを NeL[2] に読み込ませ、可視化を行ったものが図 2.8 である。なお、レイヤーは 1 年を 1 枚とした。データ読み込み直後は自動的に、キーレイヤー (左端のスライダー) は最新の 2004 年を指す。また、すべてのレイヤーが表示され、右側 2 本のスライダーはそれぞれ 1959 年、2004 年を指している。

[2]Digital Bibliography and Library Project. http://www.informatik.uni-trier.de/~ley/db/

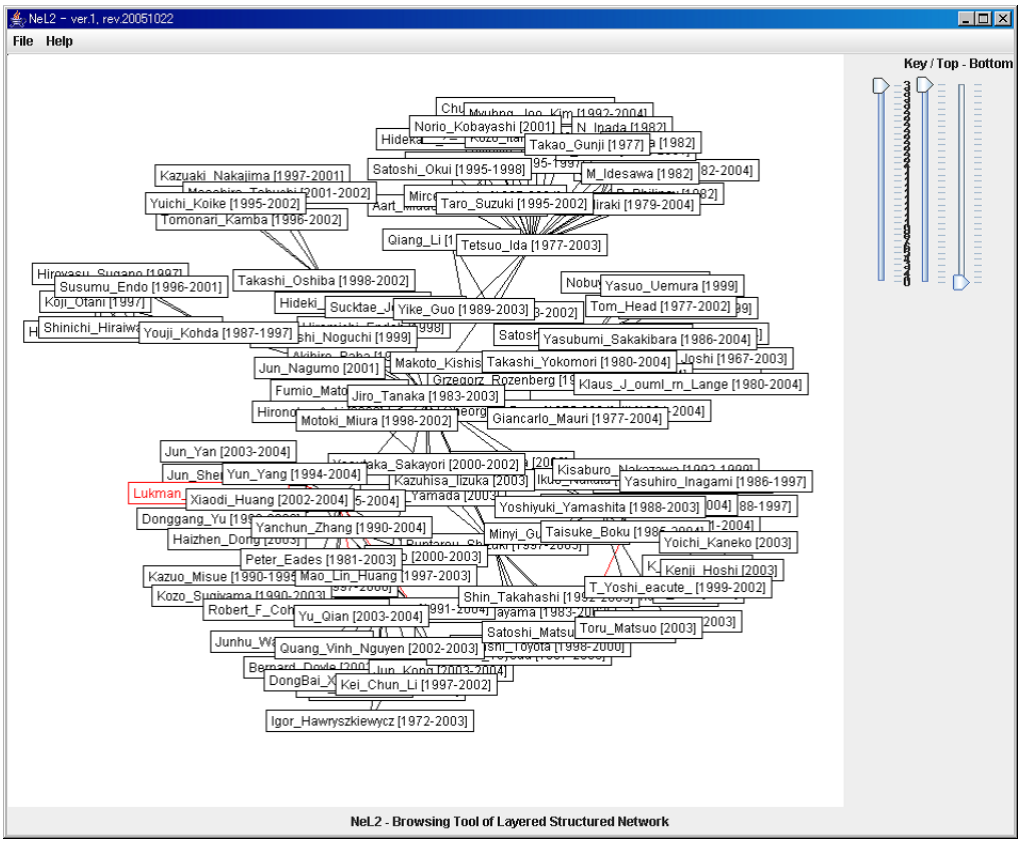

図 2.8: 論文共著関係のネットワーク図

ネットワーク図を眺めると、中央に本データの基準となる Jiro Tanaka のノードがあり、上・右・下にそれぞれ比較的大きなクラスタが存在していることが分かる。しかしながら、この状態ではどのクラスタがいつ頃活発な活動を行ったのかは分からない。

そこで右側 2 本のスライダーを操作し、表示レイヤーを Jiro Tanaka が初めてデータベースに登録された 1983 年から 1992 年までの 10 枚に限定した。また、左端のスライダーでキーレイヤーを 1990 年に設定してみた (図 2.9)。このように、表示の変更には複雑な操作を必要とせず、簡単な説明を受けるだけで誰もが自然に使うことができる。この表示から、1983 年から 1992 年の 10 年間は、上のクラスタに対応するコミュニティと密接な関係にあり、下側は比較的最近になって結びつきが生じたコミュニティであることが分かる。また、キーレイヤーを 1990 年に設定することで左下に位置するノードが多くハイライトされたが、これはこの期間において左下のクラスタに対応するコミュニティが活発に活動していたことを示唆している。

この他にもスライダーを操作してレイヤーを古い順に 1 枚ずつ重ねていくことにより、ネットワークの変化が動きとして表現できる。最初に、3 本のスライダすべてを 1959 年 (最も古いレイヤー) に合わせると、キャンバスには 1959 年のみのデータが表示される。この状態で中央のスライダー (表示する最も上のレイヤーを指定する) を上方向に操作し、レイヤーを古い順

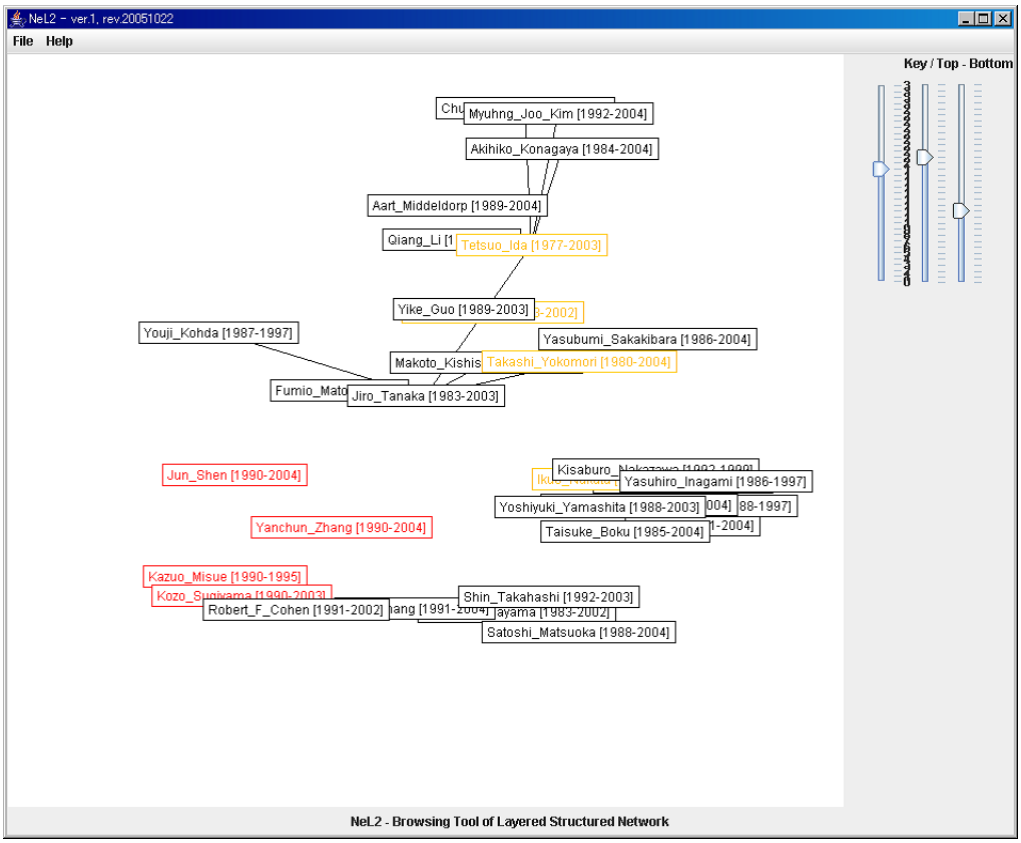

図 2.9: レイヤーの表示状態変更

に 1 枚ずつ追加していった。初期段階では著名な研究者が現れ、1983 年に中心の Jiro Tanaka が現れる (図 2.10)。時代が進むにつれて著名な研究者との繋がりが生まれ、研究室などのコミュニティが形成されていく。例えば、図 2.10 の時点ではキャンバスの上部に位置するコミュニティが活発な活動を始めており、下部に位置するコミュニティはまだ形成されていないことが分かる。このようなネットワーク変化を、レイヤー構造ネットワーク図ではスライダー操作だけで簡単に閲覧することができる。

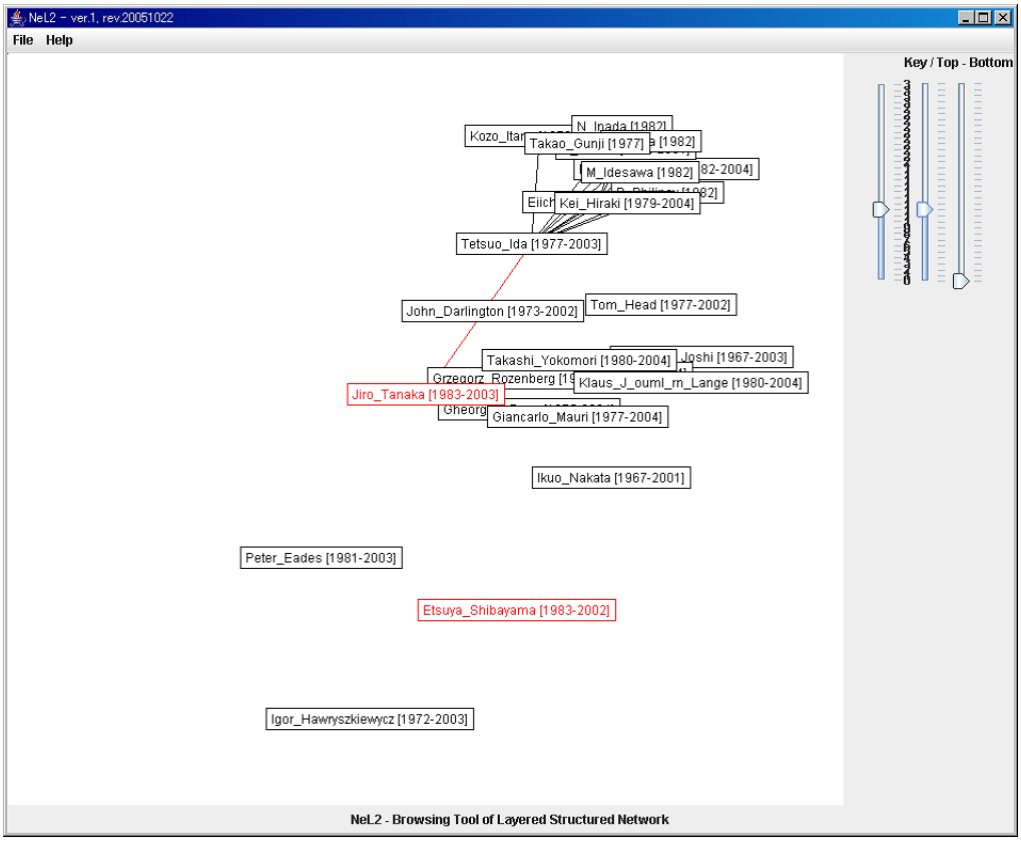

図 2.10: 基準ノードの出現とコミュニティの形成

第3章　知識創造支援ツール“IdeaCrepe”

本章では、前章で提案したレイヤー構造をもつ、ネットワークを用いて知識創造支援を行うツール “IdeaCrepe” [1]の設計と試作について述べる。

3.1　IdeaCrepe の設計方針

3.1.1　アイデアを扱う形式

多くの知識創造支援ツールと同様に、IdeaCrepe はネットワークを可視化したグラフを用いてアイデアを管理する。グラフによってアイデアを扱う場合、紙とペンによる知識創造技法と計算機上の作業の間には、一般的に次のような対応付けがなされる。

- 一つのアイデアはカードや付箋紙など、移動操作を行える紙の断片に記録することが多い。計算機上のネットワーク図では一つのノードとなり、画面上に描画される。
- アイデア同士の関係は、カード同士を線で結ぶことで表現される。計算機上でも同様に、ノード間にエッジを描画することで表現する。

これらの対応付けは一般的かつ直感的であるため、IdeaCrepe においても同様の形式を採用した。

3.1.2　履歴レイヤー構造

知識創造支援ツール上で、ユーザは様々な操作を行う。その操作例としては次のようなものが考えられる。

- アイデア断片をノードとして生成する (ノード生成)
- アイデア断片間の関係線としてエッジを生成する (エッジ生成)
- ノードやエッジのラベルを変更する (ラベル変更)
- ノードの位置を移動する (ノード移動)

[1]この名称には、アイデアがクレープ生地のように広がっていくこと、そしてミルクレープのようにアイデアが積み重なって大きな成果を生み出すことをサポートするツールでありたいという願いが込められている。

- ノードやエッジを削除する (削除)

本研究ではこれらの操作を「イベント」と呼ぶ。なお、ノードとエッジを総称して「オブジェクト」と表現する。

従来の知識創造支援ツールの多くは、アイデアを保存する際は既存データに上書きして保存するという手法をとっていた。この手法では、たとえばあるノードのラベルを変更したり位置を移動するといったイベントが発生した場合、過去のラベルや位置を参照することは不可能になる。

本ツールではデータの保存形式としてレイヤー構造を採用する。レイヤー構造は従来、時間軸をある一定の期間ごとに区切り、各区間にレイヤーを割り当てるものである。しかしながら多くの知識創造活動は断続的に行われる活動であり、一定期間ごとに区切るのは難しい。また、一つの期間内で同一オブジェクトに複数のイベントが発生した場合、この形式では全てのイベントを保存することができないという問題が生じる。

そこで本ツールでは、レイヤーは一つのイベントを一枚とし、イベントが発生するたびにレイヤーが追加されていくという概念を採用した。一枚のレイヤーにはただ一つのイベントが保存される。そのレイヤーを積み重ねていく活動は知識創造活動に等しい。

3.1.3 アイデア記録のデータ形式

NeL^2 と同様に、IdeaCrepe 上のイベントは XML 文書として記述する。NeL^2 ではレイヤー構造を持つために `layer` 要素を定義していたが、IdeaCrepe は一つのイベントが一枚のレイヤーに対応するため、レイヤーに対応する要素を明示的に用いる必要がない。このような XML 文書の例を図 3.1 に示す。

まず、それぞれの親要素について説明する。`node` および `edge` 要素は、それぞれノードとエッジを定義する。その他の親要素 (`move`, `del`, `relabel`) はイベントを定義しており、それぞれ移動、削除、ラベル変更を定義する。

次に、それぞれの親要素に共通する子要素・属性を説明する。`date` 属性はイベントの発生時間を記録し、`id` 属性はイベントの ID を記録する。この ID はイベントごとに独立してカウントされる。`label` 要素は各オブジェクトのラベル文字列を格納する。`position` 要素はオブジェクトの位置を指定するものであるが、`node` 要素や `move` 要素に対しては x, y 座標を持ち、`edge` 要素に対しては始点と終点ノードの ID を格納するようにした。これは、ノードは座標を指定することで位置が絶対的に定められるが、エッジは両端のノード位置に依存して位置が変化する可能性があるためである。

イベントを定義する親要素は、子要素として `ref` を持つ。これはイベントを適用するオブジェクトを指定する要素で、`ref_com` 属性で適用先のオブジェクト種類、`ref_id` 属性で適用先の ID を指定する。この組み合わせにより、ID がイベントごとに独立してカウントされていてもオブジェクトを一意に識別できる。

```
<?xml version="1.0" encoding="UTF-8"?>
<graph>
  <node date="1138036083031" id="0">
    <label>IdeaCrepe</label>
    <position x="311" y="235"/>
  </node>
  .....
  <edge date="1138036148453" id="0">
    <label>nextstep</label>
    <position from="7" to="10"/>
  </edge>
  .....
  <move date="1138036148502" id="0">
    <position x="248" y="420"/>
    <ref ref_com="node" ref_id="5"/>
  </move>
  .....
  <del date="1138045728102" id="0">
    <ref ref_com="edge" ref_id="14"/>
  </del>
  .....
  <relabel date="1138045879910" id="0">
    <label>NewLabel</label>
    <ref ref_com="node" ref_id="8"/>
  </relabel>
</graph>
```

図 3.1: XML による IdeaCrepe イベントの記述例

3.1.4 レイヤー構造を用いたアイデアの記録

本ツールが持つ最大の特長は、オブジェクトに対して移動、削除、ラベル変更といった操作を行っても、過去の情報が保存され参照できるという点にある。そのため、XML 文書において一度作成された要素 (イベント) は変更されたり削除されたりすることはなく、操作による変更は別の要素を追加することによって記録される。

例えば、ラベル「January」を持つあるノードを生成した後、ラベルを「February」に変更するという操作を考える。最初にノードを生成すると図 3.2 のような `node` 要素が XML 文書に追加される。ここでラベル変更を行った場合、上書き保存を行うと図 3.3 のように変更され、以前のラベルが「January」であったという情報は失われてしまう。IdeaCrepe はこのような保存方法ではなく、新規に `relabel` 要素を追加する。すなわち、`node` 要素が保存されている以前のレイヤーに対して、`relabel` 要素を持つ新規レイヤーを重ねることで情報の更新を行う (図 3.4)。

```
<node date="1138036083050" id="1">
  <label>January</label>
  <position x="271" y="175"/>
</node>
```

図 3.2: ノード生成により追加される XML 要素

```
<node date="1138036084175" id="1">
  <label>February</label>
  <position x="271" y="175"/>
</node>
```

図 3.3: 上書き保存による XML 要素の変更

```
<node date="1138036083050" id="1">
  <label>January</label>
  <position x="271" y="175"/>
</node>
.....
<relabel date="1138036084175" id="3">
  <label>February</label>
  <ref ref_com="node" ref_id="1"/>
</relabel>
```

図 3.4: レイヤー構造を用いた XML 要素の追加

3.2 IdeaCrepe の機能

3.2.1 ツールのユーザインタフェース

開発したツール “IdeaCrepe”のスクリーンショットを図 3.5 に示す。本ツールは基本的な操作を全てメインウインドウで行う。ウインドウ右の部分が描画キャンバスであり、知識創造活動はこの領域にオブジェクトを配置することで行う。ウインドウ左の部分にはファイル操作や表示変更に対応するボタンを配置し、基本的な機能はクリックするだけで呼び出せるようにした。これはメニュー操作などの煩雑な操作をできるだけ減らすことで、知識創造中のユーザの思考をできるだけ妨げないようにする配慮である。

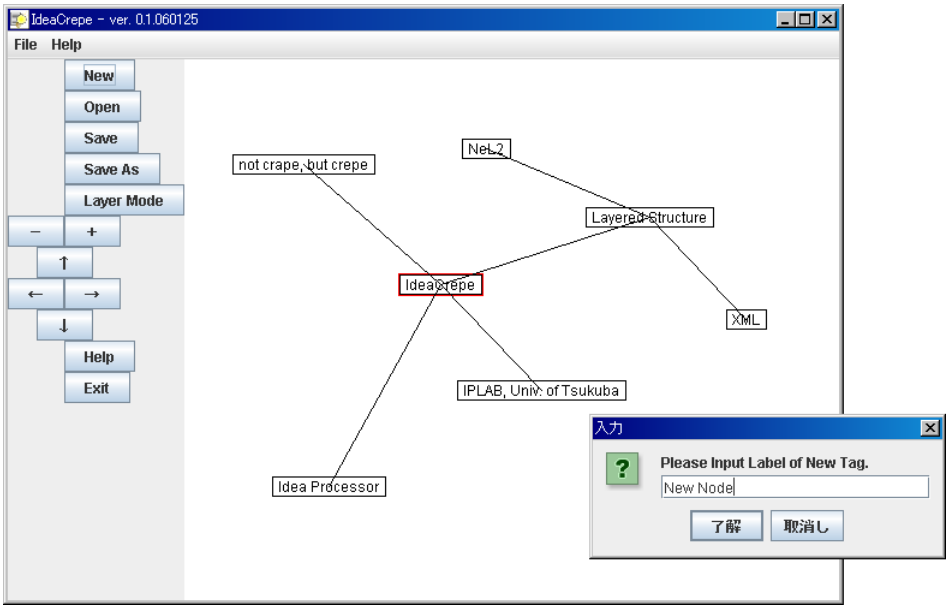

図 3.5: IdeaCrepe のスクリーンショット

3.2.2 ユーザの操作

本ツールは知識創造活動を支援するものであり、活動を妨げるものであってはならない。そのため、オブジェクト生成や選択などといったモード切替を意識せずに操作できるように配慮した。本研究ではユーザの知識創造活動を妨げないためにユーザインタフェースが備えるべき用件として、次の 2 点を考慮した。

- 操作はユーザにとってできるだけ簡単なものであり、作業を選択するための労力ができるだけ少ないものであること
- 操作はユーザにとって直感的であり、深く考えることなく自然に発想できる動きで実行できるものであること

前者の例としては、メニュー階層を深く辿るよりもボタンをクリックする方が操作としては単純である。後者の例としては、キャンバスをクリックすればそのクリックした位置にノードが生成されるような事例が挙げられる。

なお、操作に用いるデバイスとしてはマウスとキーボードの併用を想定する。

実際に IdeaCrepe を利用するユーザは、次のような操作を組み合わせることで知識創造活動を行っていく。

ノード生成

アイデアの断片を思いついたユーザはキャンバスの任意の位置をクリックする。すると図 3.6 に示すダイアログボックスが現れ、ここにラベルを入力することでアイデア断片を記録した新しいノードが生成される。このノードはキャンバスのクリックした位置に生成される。

図 3.6: 新規ノード生成ダイアログ

ノード移動とエッジ生成

紙とペンによる知識創造活動において、ノードに対応するカードを移動させる場合、我々はカードを指でつかみ、引きずることで配置を変更する。カード間に関係線を生成する場合は、カード間でペンを引きずることで線を描画する。このように、ノード移動とエッジ生成は「引きずる」という似通った操作で行っており、この操作は計算機上でのマウスドラッグと対応する。移動とエッジ生成の違いは、手にペンを持っているかどうか、そして引きずった先に別のカードがあるかどうかである。この作業を計算機上で実現する場合、ペンの有無はモード変更をユーザに意識させることになり、操作が複雑になる。そこで本ツールでは、引きずった (ドラッグした) 先に別のノードがあるかどうかで操作を判別することとした。

キャンバス上に配置されたノードをキャンバス上でドラッグすることにより、ノードの位置を変更することができる。このとき、移動したノードにエッジが接続している場合、それらの配置も自動的に変更される (図 3.7)。ノードをドラッグした先がキャンバス上ではなく他のノードだった場合、2 つのノード間にエッジが生成される (図 3.8)。

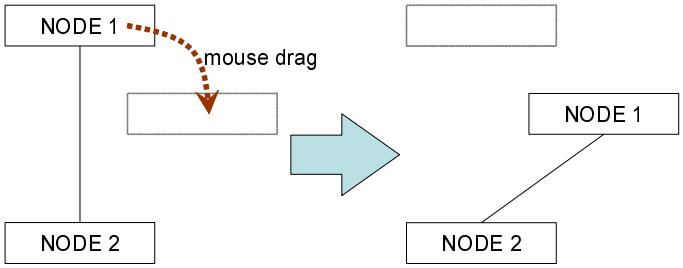

図 3.7: キャンバスへのマウスドラッグによるノードの移動

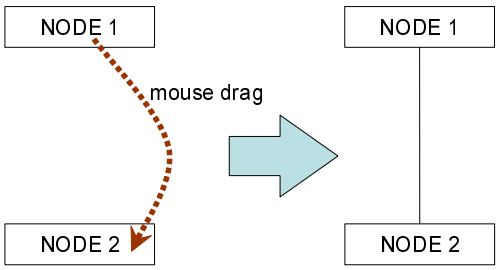

図 3.8: ノード間のマウスドラッグによるエッジの生成

ラベル変更とオブジェクト削除

ラベルの編集については、編集するオブジェクトを指定することによって行われる。本ツールでは移動時などの選択と区別するため、ノードやエッジをダブルクリックすることでラベルが編集できるようにした。

また、削除に関しても削除するオブジェクトを指定することで行われるが、描画とは異なる操作であることだけはユーザが意識する必要があると考えられる。そこで、左クリックではなく右クリックすることで削除を行うことができるようにした。

このように、オブジェクトの生成、編集に関してはユーザはツールのモードを意識することなく、簡単なマウス操作の使い分けで操作を行うことができる。

履歴レイヤー表示モードへの移行

ウインドウ左の “Layer Mode”ボタンをクリックすることで、IdeaCrepe は履歴レイヤー表示モードに移行する。このモードはユーザの期待する操作が通常のモードと明らかに違うため、ボタンをクリックして明示的にモード変更することが必要であるように実装した。このモードは NeL[2] と同様に、スライダ操作によってレイヤーの表示状態を変化させる。

3.3 IdeaCrepe の実装

本研究では、履歴レイヤー構造を持つ知識創造支援ツール “IdeaCrepe”を、Java (JDK5.0) を用いて実装した。本ツールは 26 個のクラスにより構成される。

3.3.1 レイヤー構造を用いたアイデアの可視化

レイヤー構造を用いてアイデアを XML 文書に保存することにより、以前のデータを保持したままにアイデアを更新することが可能になった。しかしながらこの手法は、たとえば 3.1.4 の例では `node` 要素 (ノードのデータ) と `relabel` 要素 (イベントのデータ) のように、一つのオブジェクトに対して複数の要素が関連することが想定される。描画時には最新あるいは

指定した期間の情報が表示される必要があるため、ノードやエッジのデータをイベントデータで上書きして更新しなければならない。また、IdeaCrepe は一つ一つのイベントをそれぞれ別々のレイヤーとして保存するため、イベント発生時に画面描画とデータ更新を同時に行う必要がある。また、既存のイベントを変更する代わりに変更を表す新たなイベントを追加するため、オブジェクトは複数のイベントに上書きされる可能性がある。

このような仕様を満たすため、IdeaCrepe では図 3.9 のような流れでデータを扱う。さらに、イベントの更新時刻を XML の `date` 属性に格納することで、時系列に沿ったイベントの管理ができるようにした。本研究では、1970 年 1 月 1 日 0 時 0 分 0 秒 GMT を起点とした時間をミリ秒で表した値を用いている。なお、この値を例えば「2006-01-10 15:43:24 GMT」などの文字列とし、この文字列を扱えるようにプログラムを改変すれば、より可読性の高い XML 文書とすることも可能である。

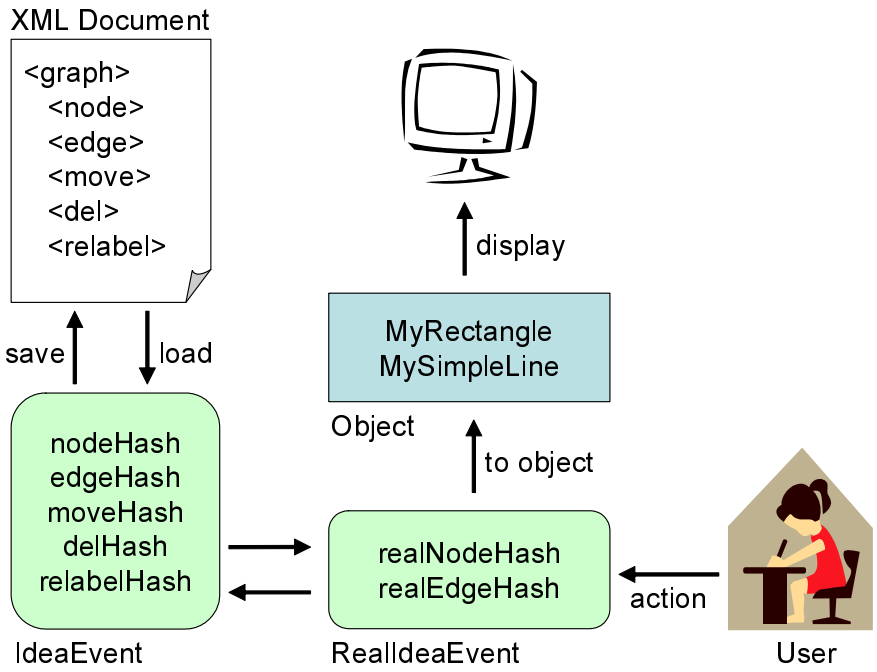

図 3.9: IdeaCrepe でのイベントの流れ

XML 文書を読み込むと、IdeaCrepe はそのデータを IdeaEvent クラスのインスタンスに記録し、ハッシュテーブルに格納する。格納されたデータのうち、移動、削除、ラベル変更のイベントに関しては対応するノードやエッジの情報に上書きしてデータを更新し、新しいハッシュテーブル (RealIdeaEvent) に格納する。このハッシュテーブルに格納されたデータを Java の Graphics オブジェクトに変換することで、データが可視化される。

3.3.2　システムの構成

本システムは大きく分けて、アイデアプロセッサ部、ファイル入出力部、履歴レイヤー表示部からなる (図 3.10)。

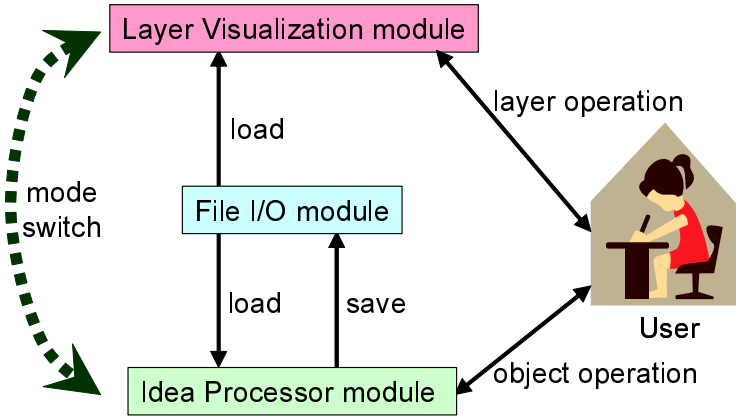

図 3.10: システムの構成図

アイデアプロセッサ部はツールのウインドウ上にオブジェクトを作成したり修正を加えるなど、一般的な知識創造支援ツールと同等の機能を提供する。

ファイル入出力部はアイデアプロセッサ部と密接に関連している。XML 文書からデータを読み込んでアイデアプロセッサ部に渡したり、アイデアプロセッサ部のデータを XML 文書に保存する部分である。

履歴レイヤー表示部は、履歴レイヤーの表示を変更することで過去のアイデア変遷などを振り返ったりできる機能を提供する。この部分については NeL^2 とほぼ同様の機能を実装した。

ユーザは主にアイデアプロセッサ部を用いて、知識創造活動を行う。活動によって産み出されたアイデアなどはファイル入出力部によって保存・読込が行われる。蓄積された履歴を振り返りたい場合は履歴レイヤー表示部が提供する機能を用い、履歴レイヤーを操作して活動を振り返ることができる。

第4章　評価と考察

本研究では、ネットワーク描画ツール “NeL^2”と知識創造支援ツール “IdeaCrepe”のそれぞれについて評価を行った。

4.1　NeL^2 の評価と考察

NeL^2 については、実際に被験者にツールを試用してもらい、いくつかの作業を通して親しみやすさや有用性を評価する実験を行った。

また、日本ソフトウェア科学会 インタラクティブシステムとソフトウェア (ISS) 研究会の第 13 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2005) [1] においてデモ発表を行い、参加者の方々からコメントをいただいた。

4.1.1　評価実験の概要と結果

評価実験は 10 人の大学生・大学院生を被験者とした。被験者は最初に約 5 分の時間を使い、レイヤー構造ネットワークがどのようなものであるかの紹介、および NeL^2 の使用方法の説明を受けた。同時にノードとエッジがそれぞれ数個の簡単なネットワークデータを入力として用い、自由に操作してツールに慣れてもらった。

その後、2.3.3 で用いた論文共著関係データを入力として与え、NeL^2 を用いて次のような 3 種類の作業を実施した。これらは 2.1.1 で述べた閲覧操作とそれぞれ対応する。

1. 基準ノードの著者がデータ上に現れる年までの共著関係を表示する。
2. 2002 年はどのコミュニティが活発に活動していたかを読み取る。
3. 1990 年代の共著関係ネットワークを表示し、その図から特徴を読み取る。

この実験を実施した後に各被験者に対してアンケート調査を行い、レイヤー構造ネットワークの有用性 (役立つか)、わかりやすさ (概念が理解しやすいか)、ツールの親しみやすさ (使ってみたいと感じるか)、使いやすさ (思うように操作できるか)、および各作業におけるツールの有用性について、それぞれ 5 段階で評価してもらった。各設問の回答の平均値を表 4.1 に示す。また、レイヤー構造ネットワークおよび NeL^2 に対する意見や感想を自由記述形式で回答してもらった。

[1] 平成 17 年 12 月 7〜9 日、香川県にて開催。http://www.wiss.org/WISS2005/

表 4.1: 評価実験のアンケート結果

対象	評価項目	評価の平均
レイヤー構造ネットワーク	有用性	3.8
	わかりやすさ	2.8
NeL^2 (ツール)	親しみやすさ	3.6
	使いやすさ	2.7
作業 1	有用性	4.0
作業 2	有用性	3.8
作業 3	有用性	4.0

4.1.2 WISS2005 デモ発表での参加者からのコメント

WISS2005 におけるデモ発表においても、評価実験とおおむね同様のコメントを多くいただいた。

簡単なスライダ操作だけで表示が変化する部分については、操作が簡単である点、行った操作によってインタラクティブに表示が変化する点に好感を持つ人が多かった。一方、「レイヤー構造」の概念が NeL^2 のインタフェースから感じられるかどうかについては否定的な意見が多く寄せられた。どの部分にレイヤーの概念が用いられているのか一目見ただけでは分からず、概念とインタフェースが一致するようなツールに発展させるべきであるとの考えをいただいた。

4.1.3 考察と課題

評価実験の結果、多くの被験者がレイヤー構造ネットワークは有用であると回答した。自由記述欄においても、スライダー操作と連動して表示が変化する点が理解を助けると評価された。また、レイヤー構造ネットワークの閲覧ツールである NeL^2 についても被験者が親しみを持って操作したことが明らかになり、自由記述欄においても「おもしろい」という感想が複数見受けられた。ネットワーク図を自在に変化させて部分的に取り出せるという NeL^2 が持つ特長についても、自由記述欄の感想などから、被験者が親しみを感じて作業に対する有用性を感じ取ることができたことが読み取れた。さらに作業 1、2、3 を行った際の有用性についても調査したが、有用であるとの意見が多かった。

これらの有用性については WISS2005 のデモ発表でも多くの閲覧者が感じ取ったことであり、インタラクティブな変化のおもしろさは高く評価されたといえるだろう。

一方、NeL^2 のユーザインタフェースではレイヤー構造ネットワークがどこに使われているかがわかりにくいという意見があった。NeL^2 の使いやすさに関しては評価結果が低く、デモ発表でも否定的意見が多く寄せられた。これは現在の NeL^2 のインタフェースが、我々が概念として持っているレイヤー構造のデザイン (図 2.1 がそれに近い) を完全には再現していない

ためであると考えられる。実際、アンケートの自由記述欄において 4 人の被験者が「レイヤー構造のイメージが見た目から分からず使いにくい」とコメントしており、今後はレイヤー構造の概念をより分かりやすく再現できるインタフェースを実装することが必要である。概念をツールのデザインとして再現することにより、レイヤー構造ネットワークがより分かりやすい概念となり、NeL^2 が使いやすいツールになることが期待される。

以上の結果より、レイヤー構造ネットワークはネットワーク図操作において有用な概念であることが示唆され、NeL^2 はユーザに親しみを持って受け入れられた。また今後の課題として、NeL^2 のインタフェースの見直しを図り、ユーザにとって分かりやすいツールに改善していくことの必要性が明らかになった。

4.2 IdeaCrepe の評価と考察

4.2.1 ツールを試用しての所見

IdeaCrepe については、実際に簡単な知識創造活動を実施し、操作感などを調査した。

オブジェクトの生成や編集に関しては、ユーザはマウス操作の使い分けによって行いたい操作を容易に選択できる。多くの既存ツールがツールボタンなどによるモード切替を採用している中、本ツールはより直感的な操作で知識創造活動が行えると考えられる。

レイヤー構造によって履歴を参照できることによる利点は現在明らかにされていない。しかしながらユーザの主観的な感想としては、過去の活動を振り返ることで今後の発想の指針を決める参考になったといえる。今後は履歴を参照することが知識創造活動においてどのような効果を上げるのか、定量的な評価が望まれる。

ツールのユーザインタフェースおよび操作体系については大いに議論の余地があるといえるだろう。NeL^2 の評価実験で指摘されたように、スライダを用いたレイヤー操作はユーザに対してレイヤーを操作しているという印象を与えにくい。このインタフェースを改善することでレイヤーの概念がより分かりやすくなり、履歴レイヤー構造の有用性が増すのではないかと考えられる。

4.2.2 入力デバイスの考察

現在の IdeaCrepe はマウスとキーボードによる入力を想定して作成している。しかしながら近年の計算機環境は実世界指向の流れにより、ペン (スタイラス) を用いることで我々が紙とペンを用いた実世界での活動により近い操作を行うことを可能にした (図 4.1)。このような現状を踏まえ、創造作業支援においても手書きのメリットを生かす試みが研究されるようになってきた [14]。多くのデザイナは高機能の計算機が存在しても、発想の初期段階においては紙とペンによる自由なデザインを好み、計算機上で行う場合も素早く自由に描くことを求めている [12]。

本研究で開発した IdeaCrepe も、マウスとキーボードによる入力だけでなく、ペンを用いた手書き入力による利用も視野に入れたいと考えている。クリックで整形されたノードを生

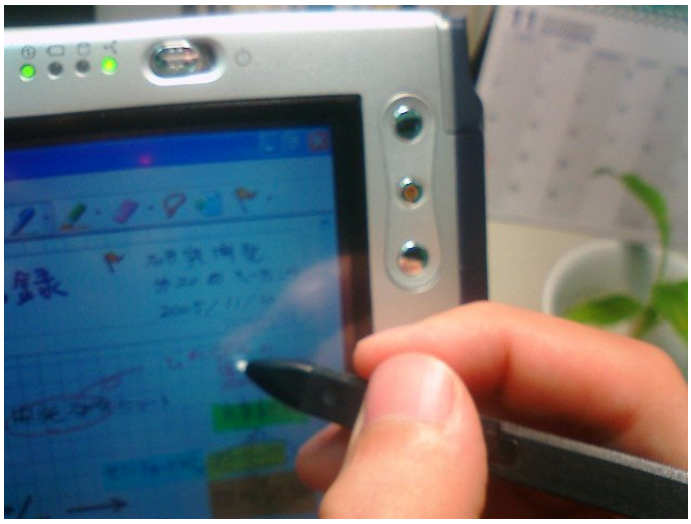

図 4.1: ペン型入力デバイス

成するのではなく、キャンバス上でフリーハンドにより矩形を描画すればノードを生成し、ノード間を線で接続すればエッジが生成されるような機能の実装が考えられるだろう。現在の IdeaCrepe はすでにモード切替を意識しない操作体系を考慮に入れており、フリーハンドの自由な入力に対応させることは十分可能であると考えている。

第5章　関連研究

5.1　ネットワーク可視化に関する研究

ネットワークの可視化に関連した研究としては様々なアプローチが試みられてきた [9]。Chen らは科学論文を対象に、著者間の参照関係およびその発展過程を可視化する手法を提案している [8]。豊田らは WebRelievo というシステムを提案し、差分ビューやクラスタビューなどを用いてウェブページの発展過程を可視化するという研究を行っている [21]。Erten らは変化するグラフを並列に配置したり三次元的に重ねるなどの枠組みを提案しており、学術的文献の関係を可視化することに応用した [10]。

また、ネットワークの可視化には様々なグラフの形状が考えられ、そのようなグラフ形状に関する研究も数多くなされている。Heer らは様々な形状でインタラクティブな可視化を行うための prefuse というツールキットを開発した [11]。Carlis ら [7] や Yee ら [17] は従来のグラフレイアウトを発展させ、螺旋や同心円に基づく配置を行う手法を提案している。

先行研究はある時点におけるネットワークのスナップショットを保存していくことで、ネットワークの変化を表現していた。これに対し本研究で開発した NeL^2 は各時点での状態ではなく、その時点に至るまでの差分に着目した。すなわち、データを差分の列として保持し、ある時点におけるネットワークは差分の統合によって表現するという、従来と異なる新しいアプローチを試みた。

5.2　知識創造支援に関する研究

発想支援、知識創造支援と呼ばれる分野の研究は長い歴史を持つ。初期の研究として有名なものに Stefik ら [16] などがあり、これは複数人による協調作業において計算機による支援を考える研究であった。一方で、「発想支援」と呼ばれるものにはどのような形態が考えられるのかについても様々な研究・提案がなされており、折原 [19] や國藤ら [20] によってよくまとめられている。

知識創造支援ツールの開発は様々な研究者によってなされており、国内においては特に KJ 法の支援に特化したツールが多く見受けられる。三末らは KJ 法の過程を図的思考展開過程としてとらえた対話型支援システム D-ABDUCTOR[22] を開発し、計算機能力を活用することで発想支援の新しい可能性を実現した。由井薗らは発想支援グループウェア郡元 [23] を開発し、複数人が計算機を通して KJ 法を行うシステムを提案している。

先行研究はツール上でアイデアをいかに扱うかを主眼において提案・開発がなされたもの

が多い。本研究はこのような先行研究とは別の着眼点、すなわち履歴保存の重要性を中心に知識創造支援を考えた。IdeaCrepe によって実装された履歴レイヤー構造は、機能拡張によって先行研究の様々なツールに応用することができると考えられる。

第6章　結論

本研究ではまず、ネットワーク図へのレイヤー構造の導入を提案し、その数学的定義を行った。その定義を元に、レイヤー構造ネットワークを扱うツール“NeL^2”を開発し、評価実験を行った。

従来のスナップショットを保存していく手法とは異なり、NeL^2 は時系列におけるネットワークの差分をレイヤーという単位で管理することで変化や差分の可視化に重点を置いている。また、自然な操作でユーザの意図した表示を得ることができる。レイヤー構造ネットワーク図により、従来はわかりにくかった特定の時期のネットワークにおける傾向や進化の過程など、様々な情報が把握できるようになった。

次に、このレイヤー構造ネットワークを知識創造支援ツールに応用し、“IdeaCrepe”を実装した。本ツールは様々な知識創造支援活動を支援する汎用的なツールであり、レイヤー構造によってイベントの履歴を保存している。この履歴を参照することで過去の発想過程を思い出すことができ、新しいアイデアを生み出す力になると考えられる。

今後の研究としては、評価実験で明らかになった問題を解決するためにグラフそのものを「レイヤー」と分かるような形で描画されるように改良を加える予定である。また、スライダーを使った操作に代えてレイヤーそのものをピックアップしたりドラッグするといった操作体系を導入するために研究を継続している。将来的にはレイヤー単位だけではなく注目ノードに関連するデータを色づけしたり、レイアウトの方法を多様化するなど、さらに多くの機能を実装していくことを計画している。また、より客観的な評価を行うことで本システムの特性や利点をより明らかにしていくことも必要であると考えられる。

さらに、現在の整形されたオブジェクト生成ではなく、フリーハンドでより自然な入力をサポートすることにより、ユーザの発想をさらに自然に引き出すことが可能になると考えられる。

謝辞

本論文を執筆するに当たり、指導教員である田中二郎先生、三末和男先生をはじめ、志築文太郎先生、高橋伸先生には、丁寧なご指導と適切な助言を多数いただきました。心より感謝申し上げます。

また、IPLAB(田中研究室) の皆様にもゼミなどを通じて大変貴重なご意見をいただきました。特に NAIS グループの皆様にはグループでのミーティングだけでなく、日常的に多くのご意見やご指摘を頂きました。ありがとうございました。

最後に、物心両面に渡って遠く福岡から私を支えてくれた両親・家族、様々な面で力になってくれた多くの友人たちに、心より感謝いたします。

参考文献

[1] JUSTSYSTEM アイデアマスター. http://www.justsystem.co.jp/ideamaster/

[2] iEdit. http://homepage3.nifty.com/kondoumh/software/iedit.html

[3] FreeMind. http://freemind.sourceforge.net/

[4] IdeaFragment2. http://homepage3.nifty.com/neko33/lzh/ideafrg2.htm

[5] IdeaTree. http://www.dicre.com/soft/itree.htm

[6] U. Brandes & T. Willhalm. Visualization of Bibliographic Networks with a Reshaped Landscape Metaphor. *Proceedings of the 4th Joint Eurographics-IEEE TCVG Symposium on Visualization 2002*, pp. 159–164, 2002.

[7] J. V. Carlis & J. A. Konstan. Interactive Visualization of Serial Periodic Data. *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pp. 29–38, 1998.

[8] C. Chen & L. Carr. Visualizing the Evolution of a Subject Domain: A Case Study. *Proceedings of the conference on Visualization '99*, pp. 449–452, 1999.

[9] H. Chi, J. Pitkom, J. Mackinlay, P. Pirolli, R. Gossweiler & S. K. Card. Visualizing the Evolution of Web Ecologies. *Proceedings of the SIGCHI conference on Human factors in computing systems 1998*, pp. 400–407, 1998.

[10] C. Erten, S. G. Kobourov, V. Le & A. Navabi. Simultaneous Graph Drawing: Layout Algorithms and Visualization Schemes. *Proceedings of the 11th Symposium on Graph Drawing*, pp. 437–449, 2003.

[11] J. Heer, S. K. Card & J. A. Landay. prefuse: a toolkit for interactive information visualization. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems 2005*, pp. 421–430, 2005.

[12] J. A. Landay & B. A. Myers. Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*, Vol.34, No.3, pp. 56–64, 2001.

[13] K. Misue. Visual Destruction for Understanding Small-world Networks. *Proceedings of HCI International 2005*, CD-ROM, 2005.

[14] K. Misue & J. Tanaka. A Handwriting Tool to Support Creative Activities. *Proceedings of 9th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems*, pp. 423–429, 2005.

[15] N. Nakazono, K. Misue & J. Tanaka. NeL2: Network Drawing Tool for Handling Layered Structured Network Diagram. *Proceedings of Asia Pacific Symposium on Information Visualization 2006 (APVIS2006)*, pp. 109–115, 2006.

[16] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning & L. Suchman. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, Vol.30, No.1, pp. 32–47, 1987.

[17] K. -P. Yee, D. Fisher, R. Dhamija & M. A. Hearst. Animated Exploration of Dynamic Graphs with Radial Layout. *Proceedings of INFOVIS*, pp. 43–50, 2001.

[18] F. Yoshikane, T. Nozawa & K. Tsuji. Comparative Analysis of Co-authorship Networks Considering Authors' Roles in Collaboration: Differences between the Theoretical and Application Areas. *Proceedings of 10th International Conference of the International Society for Scientometrics and Informetrics*, Vol.2, pp. 509–516, 2005.

[19] 折原良平. 発想支援システムの動向. 情報処理, Vol.34 No.1, pp. 81–87, 1993.

[20] 國藤進, 山下邦弘, 西本一志, 藤波努, 宮田一乗. 知識創造支援システム研究開発の動向と JAIST における開発の現状. 第 1 回知識創造支援システムシンポジウム報告書, pp. 1–10, 2004.

[21] 豊田正史, 喜連川優. WebRelievo: ウェブにおけるリンク構造の発展過程解析システム. 第 12 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2004), pp. 89–94, 2004.

[22] 三末和男, 杉山公造. 図的発想支援システム D-ABDUCTOR の開発について. 情報処理学会論文誌, Vol.35, No.9, pp. 1739–1749, 1994.

[23] 由井薗隆也, 宗森純, 長澤庸二. カード型データベースをもつ KJ 法一貫支援グループウェアの開発と適用. 情報処理学会論文誌, Vol.39, No.10, pp. 2914–2926, 1998.

[24] 川喜田二郎. 発想法. 中公新書, 1967.

[25] 高橋誠. 問題解決手法の知識. 日本経済新聞社, 1984.