Vol. 30 No. 4　　情　報　処　理　　Apr. 1989

解　説

4. 新しい技術の応用

4.2 ハイパテキストとそのプログラミング環境への応用†

高　田　広　章††

1. はじめに

近年，ハイパテキストと呼ばれるデータの管理手法が注目を集めており，その応用の一つとしてプログラミング環境への適用に関する研究も広く行われている．ハイパテキストの考え方の起源は意外と古く，1945年の Bush による論文[1] にはじまるとされている．その後1960年代には，T. Nelson らのグループと D. Engelbert らのグループにより基礎的な研究やシステムの設計/実現が進められた．このように古くから知られている手法がここ数年で急速に注目を集めた背景には，ビットマップディスプレイを備えた高性能なワークステーションやコンピュータネットワークの普及があることはいうまでもない．

ハイパテキストの応用範囲は非常に広く，文書作成/読解支援やアイデアプロセッシング，電子辞典，電子出版，テレコンファレンシング，マルチメディアデータベースなど多岐にわたる．その中にあって，プログラミング環境への応用は，比較的未開拓な分野であるということができる．これは，プログラミング環境を構成するツールの中には，テキストをリニアな構造以外のなんらかの構造をもつものと捉えているものが多く，あえてハイパテキストという考え方を導入する必要性が認識されなかったためと考えられる．しかし，プログラミング作業と密接な関係にある文書作成/読解やアイデアプロセッシングといった作業がハイパテキストを用いたシステムによって効果的に支援される環境においては，プログラミング作業自身もおのずと変化していくであろうし，また複雑になりがちなプログラミング環境の構成を整理/統合していくためにも，ハイパテキストは重要な鍵となろう．逆に，複雑なデータ構造の管理を必要とするプログラミング環境への応用を研究することは，ハイパテキスト自身の発展にとっても，一応用分野の研究以上の意味をもつ．

本稿では設計/実現されている各種のシステムを紹介することにより，ハイパテキストシステムとそのプログラミング環境への応用について，現状の概観を与え今後の展望について考察する．まず最初に，ハイパテキストとは何かについて述べ，代表的なハイパテキストシステムおよびハイパテキストモデルを紹介する．次に，プログラム開発支援のためのハイパテキストシステムをいくつか紹介し，最後にハイパテキストのもつ利点や問題点について議論する．

2. ハイパテキストシステム

2.1 ハイパテキストとは？

ハイパテキストという言葉は，T. Nelson によってはじめて使われたものであるが，何をもってハイパテキストとするかは研究者によって異なり，厳密な定義はなされていない．もっとも広義においては，非線形テキストという別名が示すように，紙の上に表示されるということからテキストに課せられている一次元（ないしは二次元）の構造にとらわれずに，コンピュータ上でテキストをより有効に扱うための構造のことをいう．通常は，テキストを意味上まとまりのある小部分（これをノードと呼ぶ）に分割し，それらの間をリンクにより関係づけたネットワーク構造を基本に考える．このような構造を導入することにより，自然言語で書かれた文書のようにマシンがその意味を解釈できないテキストを扱う場合にも，なんらかの支援を行うことが可能になる．逆にプログラムテキストのように，マシンがテキストからその意味や構造を抽出できる場合には，それと重複する明示的な構造を導入することによる効果は，効率の問題を除いては小さい．

ハイパテキストシステムとは，ハイパテキストの考

† Hypertext and its Applications to Programming Environment by Hiroaki TAKADA (Department of Information Science, Faculty of Science, The University of Tokyo).

†† 東京大学理学部情報科学科

えを活用して構築されたアプリケーションシステムのことである．J. Conklin はハイパテキストに関するサーベイ論文[2]の中で，典型的なハイパテキストシステムのもつ特徴として次のような項目をあげている．

- まとまった情報を含むノードが，ネットワーク構造に組織される．
- ディスプレイ上のウィンドウとノードが，基本的には一対一に対応する．
- 標準的なウィンドウ操作機能を備えている．
- ウィンドウは，任意個のリンクアイコンを含むことができる．リンクアイコンとは，リンクをたどる際に手掛かりとなる図形(ないしは文字)のことをいう．
- 新しいノードやリンクが容易に作れる．
- 次の三つの方法でデータにアクセスできる．(1)リンクをたどることによって，(2)文字列，キーワード，属性値のサーチによって，(3)ネットワーク構造を視覚的に表示するブラウザによって．ブラウザとは，テキスト（より一般には情報）のもつ構造を利用して，テキスト（情報）の必要な箇所を効率よく探し出し，読むためのツールのことをいう．

これらの特徴はハイパテキストシステムに必須のものではないが，ほとんどのシステムがこれらの特徴の多くを共有している．

また，ハイパメディアという言葉が使われることがあるが，これはハイパテキストの構造をもつマルチメディアシステムのことをいう．本稿では，ハイパテキストという言葉を，ハイパメディアまで含んだ意味で用いる．

以下本章では，ハイパテキストシステムについての理解を深めるために，テーマのプログラミング環境とは離れて，既存のハイパテキストシステムの中から代表的なものを4つ紹介する．

2.2 NLS/Augment[3]

SRI の D. Engelbert らによって設計/実現された NLS (oN Line System) は，初期のハイパテキストシステムとしてはきわめて革新的なものであった．彼は NLS を，ユーザとコンピュータの共同作業により，ユーザの知性をより有効に引き出すためのシステムとして構築した．

NLS においてファイルは，全体として一つの階層構造を成しているステートメントと呼ばれる小部分から成っている．各ステートメントは，階層構造中での位置を表す名前をもち，同じファイル中または異なったファイル中にあるステートメントの間に，参照リンクを付けることができる．すなわち，NLS では階層構造をテキストの基本的な構造と考え，階層構造をまたぐ参照リンクは補助的なものと位置づけている．ユーザはこれらの構造を作成/編集することにより，アイデアの整理を行い，その結果としての文書を作成する．テキストを効率良く読むための機構としては，表示されるステートメントをいろいろな方法で制限することができるビューの制御機能をもっている．また，これらの作業を離れた場所にいる複数の構成員から成るグループで行うための機構も用意している．

NLS の開発プロジェクトにおいては入出力デバイスについても研究/開発が行われており，その成果の一つとして現在ではポインティングデバイスの代表格となったマウスが発明された．また，システムの開発を通じて多くのソフトウェアツールが作られ，今日いうところのソフトウェア工学の基礎を築いたことについてはあまり知られていない．

研究目的のシステムであった NLS は，現在では Augment システムに発展し，“知識労働者 (knowledge worker)” のためのツールの統合環境として実用化されている．

2.3 Xanadu

Xanadu プロジェクトは，T. Nelson らにより，世界規模のハイパテキストデータベース構築を目指してすすめられているプロジェクトである．Xanadu では NLS とは逆に，グラフ構造を基本と考えており，階層構造に対する特別なサポートはない．

Xanadu の設計は，バックエンドとなるハイパテキストデータベースとユーザインタフェース部分に分離されており，特に前者に重点をおいて研究が行われている．たとえば，ハイパテキスト構造に統合された版管理手法，著作権の保護や電子的な課金といった問題が研究されている．

2.4 NoteCards[4]

NoteCards は，Xerox PARC で開発されたハイパテキストシステムで，現在までででもっとも成功したシステムの一つということができる．NoteCards は汎用のハイパテキストシステムとして知られているが，もともとはアイデアプロセッシングと研究者のための文書作成支援ツールとして設計／開発された．

NoteCards では，テキストまたはグラフィックデータを含むことができるノートカードと呼ばれる単位を基本として，それらを型をもったリンクによりグラフ状につないだ構造を扱う．リンクはあるカード中

のアイコンから，別のカード全体を指すことができる．また特別なカードとして，カードとリンクの作る構造を図示するブラウザや，多くのカードを組織しておくためのファイルボックスがある．

NoteCards は，Xerox の Lisp マシンの上に構築されており，100 以上の Lisp の関数によりハイパテキスト構造を操作することを可能にしている．そのため，InterLisp を使ってシステムを拡張したり，他の Lisp ベースのシステムと統合することも可能である．

2.5 HyperCard[5]

HyperCard は，Apple 社が Macintosh 用に開発したシステムで，もっとも広く使われているハイパテキストシステムである（HyperCard はハイパテキストシステムではないとする立場もあるが，ここでは他と同様に議論する）．

HyperCard では情報の基本単位はカードと呼ばれ，カードの上には任意のビットパターンを描くことができる．カードの集まりをスタックと呼び，スタック中のいくつかのカードは共通のバックグラウンドをもつ．また，カードやバックグラウンドの上には，フィールドと呼ばれるテキスト入力のためのエリアや，ボタンを置くことができる．HyperCard では，ボタン，フィールド，カード，バックグラウンド，スタックはオブジェクトと呼ばれ，各オブジェクトはその上にカーソルが置かれた状態で行われた操作（たとえば，マウスボタンのクリック）に対するアクションを表すスクリプトをもつ．行われた操作に対するスクリプトがない場合は，親のオブジェクト（通常はそのオブジェクトを含むオブジェクト）のスクリプトが起動される．このスクリプトに他のカードを表示させるよう記述することで，そのオブジェクトをリンクの起点とみなせる．

以上で述べたシステム以外にも，ZOG/KMS[6]，Intermedia[7]，Textnet[8] といった多くのハイパテキストシステムが研究/実現されている．

3. ハイパテキストモデル

ハイパテキストモデルとは，ハイパテキストシステムのバックエンドとなるハイパテキストデータベースの構造を次のような目的でモデル化したものである．

A）モデル化を行い，ハイパテキストのデータ構造とアクセス方法を明確にすることにより，そのハイパテキスト構造のもつ表現力が明確になり，限界や他の構造との比較などの評価が可能になる．

B）モデル化した構造を実現するデータベースサーバを構築し，アプリケーションをその上で開発するための基盤とすることにより，各種のハイパテキストアプリケーションの実験/開発を容易にし，またそれらのアプリケーション間でデータの交換を可能にする．

どのハイパテキストシステムも，実現されている以上なんらかのモデルをもっているわけであるが，この章では特にハイパテキストモデルに重点をおいた研究をいくつか紹介する．

3.1 実身/仮身モデル[9]

実身/仮身モデルは **BTRON** の統合操作環境における情報の蓄積/表現/管理のためのモデルであり，ハイパテキストのモデルであると同時に，ユーザインタフェースのモデルでもある．ここではハイパテキストモデルの面に絞って紹介をすすめる．

実身/仮身モデルは一種の（高レベルな）ファイルシステムと考えられる．従来のファイルに相当する情報のまとまりのことを実身と呼び，それを参照するタグを仮身とよぶ．仮身は文字や図形データと同様に実身の中に埋め込むことができ，実身中では一つの文字（ないしは図形）であるかのように扱われる．仮身は，その内部に自分が指している実身を扱うためのデフォルトアプリケーションの ID やアプリケーションに渡すパラメータをもつことができ，それを手掛かりにして実身を参照/編集できることから，ハイパテキストのリンクの起点に対応する．

ファイルシステムとして使われるためには，実行効率が良いことが不可欠である．そのため，**BTRON** の実現においては次のような工夫が行われている．まずカーネル内のファイルマネージャは，一つの実身を，通常のファイルシステムのディレクトリに相当する役割をもつリンクレコードと，ファイルの役割をもつその他のレコード（データレコードなど）とに分割して管理し，前者についてのみレコードの内部まで管理を行う．仮身に関する情報は，他の実身を指すのに必要な最小限のもののみをリンクレコード中にもち，仮身の位置や表示属性などはデータレコード中に入れる．つまりカーネルは実身間のリンクしか管理せず，仮身の構造には関与しない．それに対して，リンクレコード中の実身を指すリンクとデータレコード中の仮身に関する情報を結びつけるのは，外核に位置づけられる実身/仮身マネージャの役割である．さらにこのマネージャの機能は，将来的には TACL[10] と呼ばれる言語を用いて拡張することができ，たとえばある実身

中のある位置から別の実身中のある位置を指すようなリンクを作ることも可能になる.

実身/仮身モデルの最大の特徴は,それがOSのファイルシステムとなっていることである.これにより,真に統合されたハイパテキスト環境の構築が可能になる.

3.2 HAM[11),12)]

HAM (Hypertext Abstract Machine) は,後で紹介するNeptuneシステムの最下位レイヤとして構築された汎用のハイパテキストデータベースである.HAMは,トランザクションベースでネットワークからもアクセス可能なマルチユーザ対応のデータベースサーバとして実現されている.一般的で柔軟性のあるサーバを指向しており,実際いくつかの異なったハイパテキストのアプリケーションに適用可能である.

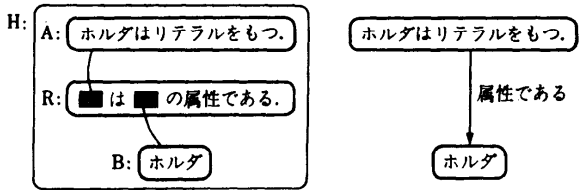
HAMでは情報の単位はノードと呼ばれ,ノード間にはそれらの関係を表すリンクをつけることができる.ノードやリンクをまとめるための機構をコンテキストという.各コンテキストは一つの親コンテキストをもち,任意個のコンテキスト,ノード,リンクを含む.コンテキストは,作業領域の切り分けやソフトウェアシステムの構成管理,版管理木などを実現するために使われる.コンテキスト,ノード,リンクには属性をつけることが可能である.属性はそれらのオブジェクトに意味を与える働きをする.また,これらのオブジェクトが作る全体の構造をグラフと呼んでいる.

また,汎用のハイパテキストサーバとして機能するように,版履歴の管理を行い,アクセスするオブジェクトを制限するためのフィルタ機構やセキュリティ確保のためのアクセスコントロールリストの機構も組み込まれている.

3.3 ホルダ・ネットワーク[13),14)]

ホルダ・ネットワークは,アイデア・プロセッサのためのデータ構造として,東京大学の西尾らによって考案されたハイパテキストモデルである.ホルダ・ネットワークにおいては,通常のハイパテキストシステムのノードとリンクがともにホルダという概念に統合されているのみならず,HAMのコンテキストのようなノードやリンクをまとめて扱う機能までもホルダにもたせることに成功している.

各ホルダはリテラルを含むことができる.リテラルとは,テキストの小部分や図形など,その環境で扱える任意のデータの集まりである.また,リテラルはその中にスロットと呼ばれる一種の変数記号を含むことができる.リテラル中にスロットをもたないホルダは通常のハイパテキストのノードに対応し,スロットを二個もつホルダはリンクの型に対応する.また,ホルダはいくつかのホルダを中に含むことができる.この状況を前者のホルダが後者のホルダをホールドしていると呼ぶ.ホールド状況を表すために各ホルダには,そのホルダがホールドしているホルダの一覧,およびそのホルダをホールドしているホルダの一覧が記憶されている.さらにホルダは,そのホルダがホールドしているホルダやその中のスロット,そのホルダ自身のスロットを単一化するための,単一化環境表をもつ.



(a) ホルダ・ネットワーク (b) 通常のハイパテキスト

図-1 ホルダ・ネットワークによるリンク構造

以上で述べたホルダ構造を用いてリンクを実現した例を図-1に示す.この図にはホルダが4つ描かれており,ホルダHが他の三つのホルダをホールドしている.ホルダRは,スロットを二つもつことからリンクの型を表すホルダであることが分かる.ホルダR中のスロットをホルダA,Bとつなぐ二本の線は,ホルダHにおける単一化の状況を表す.ホルダHは,ホルダRの表すリンクの型のインスタンスを表す.結果として,これらの4つのホルダは,通常のハイパテキストシステムにおける二つのノードと一つの(型付き)リンクに対応する.

ホルダ・ネットワークは,任意個の端点をもつリンクを作れる,ある型のリンクのすべてのインスタンスを容易にリストアップできる,(ここでは述べなかったが)ホルダによって表現された概念を合成した概念を表すホルダを合成元のホルダをホールドさせることにより表現できる等々,きわめて強力な表現力をもつハイパテキストモデルである.ただし,一般に実行効率は悪いと考えられる.実用的なシステムの実現が待たれる.

4. ハイパテキストとプログラミング環境

本章では,ハイパテキストのプログラミング環境への応用例を4つ紹介する.プログラミング環境にハイ

パテキストの概念がもち込まれたのは，オンラインヘルプシステムが最初であろう．その後，大規模なソフトウェア開発の際に作られる大量の文書やプログラムを保存するためのデータベースとしての使われ方が出てきた．さらに最近では，プログラムテキストをハイパテキスト上に展開したり，さらにはプログラミング言語そのものをハイパテキストベースにする研究も行われている．

4.1 Neptune と Dynamic Design[12),15),16)]

Neptune は，Tektronix 社の研究所で開発されたCAD 応用のためのハイパテキストシステムである．CAD の中の一つとして，CASE (Computer-Aided Software Engineering) が特に重視されており，その延長としてやはり Tektronix 社で開発された CASE システムとして Dynamic Design がある．

Neptune の最大の特徴は，下位レイヤであるハイパテキストデータベース（前章で紹介した HAM）と上位レイヤとなるユーザインタフェース部を分離したレイヤ構造になっていることである．ユーザインタフェース部としては，ノードとリンクの作るグラフ構造を扱うためのグラフブラウザ，階層構造に組織された文書を扱うためのドキュメントブラウザ，一つのノードを扱うためのノードブラウザなどが，HAM とプロセス間通信により接続されたアプリケーションの形で実現されている．

Dynamic Design は，C 言語をベースとした CASE 環境で，やはり HAM の上に構築されている．Dynamic Design では，要求記述や仕様書，設計ノートやドキュメント，実現ノート，ソースコードやオブジェクトコード，テスト仕様とその結果，ユーザ向けのドキュメントなど，ソフトウェアプロジェクトにかかわるすべての文書をハイパテキストデータベースを用いて管理する．これらの文書をそれぞれノードとして HAM に格納し，文書間の関係をリンクで表す．またコンテキストを，構成管理やプロジェクト用/個人用の作業領域の切り分けに用いる．それぞれのノード，リンク，コンテキストの意味は属性を用いて表現する．

また Dynamic Design では，C 言語のソーステキストをハイパテキストの枠組みの中で保存/編集し，各種の支援を行うユーティリティを用意している．具体的には，一つの関数やひとまとまりの変数宣言を一つのノードに入れ，その間の呼び出し関係や参照関係をリンクで表現するなどの方法を用いている．これにより，ある関数を呼び出している場所はどこか，ある変数を参照している場所はどこかという問合せに，ソーステキストの編集中でも答えることが可能になる．

4.2 DIF[17)]

DIF (Documents Integration Facility) は，ソフトウェアシステムの開発にともなう情報を単一の枠組みの中で管理するために，USC の System Factory で作られたシステムである．System Factory では，DIF を使って大規模なシステム開発の実験/研究が行われている．

System Factory においては，どのプロジェクトに関するドキュメントも同じ構造をもつよう考えられている．そのために，すべてのプロジェクトに共通な基本テンプレートとフォームが，スーパユーザによって定義される．基本テンプレートとは，そこに情報をうめることによってドキュメントを作成することができる雛型のことで，フォームとは，基本テンプレートを木構造に組織したものである．一般ユーザは，フォームに従ってドキュメントを作成していく．またユーザは，基本テンプレートから作成したドキュメントの間にリンクをつけ，それをたどってドキュメントを読み進めることや，ドキュメントにキーワードをつけ，それを使ってドキュメントの検索を行うことができる．

DIF は，ソフトウェアのライフサイクルの全局面を支援するシステムである．そのため DIF は，各種のツールに統一的なインタフェースを提供している．また新たなツールを，環境に容易に取り込めるようになっている．

さらに現在は，DIF における経験に基づき，ユーザや環境についての知識をもつことができ，ソフトウェア開発の過程の中で単なる静的なデータベースではなく，アクティブな参加者となりうる，I-SHYS (Intelligent Software Hypertext System) の研究/開発が行われている．

4.3 PDN[18)]

PDN (Program Document Network) システムは，筆者によって構築された実験用のハイパテキストシステムである．主な応用としては，ソフトウェアの保守作業効率化のためのドキュメントの整備や，再利用のためのソフトウェア部品データベースを考えている．たとえば，ソーステキストのある部分とドキュメントのそれを説明している部分をリンクでつないでおくことにより，ソースを読んでいるときに容易にドキュメントの関連部分を引き出せるばかりでなく，ソー

ステキストを改訂した際にドキュメントのどの部分の修正が必要かが判別でき，またソーステキスト中の着目していない部分を短い説明文に置き換えて表示することにより，ソーステキストの一覧性をあげることも可能になる．

PDN システムは，完成したソーステキストやドキュメントの管理の支援に重点を置いており，PDN システムの各ノード内のテキストをユーザが自由に編集することは禁止されている．これは，あるソーステキストを不注意に編集すると，それに依存している他のソーステキストの正当性を保障できなくなるためである．そのため，PDN システムのノードの内容を更新する場合は，そのノードの新バージョンを別に作ってからそれを追加登録するという手順をふまなければならない．

PDN システムも，ハイパテキストファイルシステムを構成する PDN 基本システムと，ユーザインタフェース部からなるレイヤ構造を採用している．また，PDN の実験システムは，すべて Emacs 上で Emacs Lisp を用いて実現されており，通常の文字端末からも扱えるユーザインタフェースをもっている．

4.4 TIPE[19]

TRON プロジェクトのサブプロジェクトの一つである TIPE (TRON Integrated Programming Environment) プロジェクトは，**BTRON** マシンをホストとするソフトウェア開発環境の構築を目指している．前節で述べたように，**BTRON** マシンはハイパテキストの考えをファイルシステムの基本としているため，その上で作られる TIPE システムもそれを積極的に活用するよう設計されている．

具体的には，大規模なソフトウェア開発の際に重要となる構成管理・版管理をハイパテキストの考えを用いて簡潔に実現する方法が提案されている．UNIX 上のプログラミング環境では，多くのソースファイルからなるソフトウェアを開発する場合には，モジュールごとにサブディレクトリを作って管理する方法が用いられる．構成管理ツールとしては make が一般的で，各サブディレクトリごとに makefile (make のための構成情報ファイル) を作る．この構造をハイパテキスト (TIPE システムの場合は実身/仮身モデル) 上に実現すると，サブディレクトリと構成情報ファイルを一つのノードにでき，より直感的に扱えるようになる．つまりあるモジュールを表すノード内に，そのモジュールを構成するソーステキストないしはサブモジュールを指すリンクと，コンパイル/リンクする際のオプションなどそのソーステキストの扱い方に関する情報を合わせて記述することになる．

版管理については，リンクに付属情報が付けられることを利用している．すなわち，各ノードは通常の方法で版管理をし，リンクの付属情報としてそのリンクの指しているノードのバージョン番号をいれておくことにより，ユーザは通常の方法でノードを開くだけで必要なバージョンが取り出せる．さらに，ディレクトリに対応するノードをも版管理することで，ルートのノードを開く際に必要なバージョンを指定すれば，それ以下のノードは，ルートノードと適合する版が開かれるようにすることが可能になる．

TIPE システムにおける標準言語として設計を進めている TIPE/L[20] は，このような構成管理法を意識した設計となっている．具体的には，たとえば Modula-2 などの従来のモジュラ言語では，一つのファイル内にネストしたモジュール構造をもち，ファイル間には平坦な構造しかなかったのに対し，TIPE/L においては，一つのファイル (ないしはノード) は一つのモジュールしか含まないが，ハイパテキスト構造を活用してファイルを階層状に構成することができる．多くの言語ではファイルはコンパイル単位でもあるため，この構造はコンパイル単位を小さくすることに役立つ．

5. ハイパテキストの問題点と今後の展望

前章までで，主要なハイパテキストシステム，ハイパテキストモデル，プログラミング環境への適用の例を紹介してきた．その中では，何が可能になったか，何が容易になったかを中心に述べてきたが，この章では逆にハイパテキストのもつ問題点について議論し，プログラミング環境への応用について今後の展望を述べる．

J. Conklin サーベイ[2] では，現在のインプリメントに関する問題以外に，ハイパテキストに固有の問題点として次の 2 点をあげている．

(1) 迷子の問題 (disorientation problem)

複雑なハイパテキスト構造の中で，今どのあたりにいるのか，あるノードへいくにはどうすればいいのかが，分からなくなるという問題．

(2) 心理的負荷 (cognitive overhead)

ハイパテキストにはリンクという形で文書の分岐が数多く存在する．文書の分岐は思考をも分岐させ，ユーザによけいな心理的な負荷を強いることになる．

これは通常のテキストの場合でも，注釈/脚注が多い場合などに起こる問題である．

Conklin は，これらの問題は，ハイパテキストシステムの性能とユーザインタフェースの改善や，ユーザが必要とする情報のみをシステムが選択してみせるための機構の研究を通じて，部分的には解決できるとしている．

ここでは，これらの二つの問題以外に，特にプログラミング環境との関連で出てくる問題点として，版管理における問題点を指摘したい．一般にハイパテキスト構造は版管理と相性が良いと考えられており，現にいくつかのシステムで版管理がサポートされている．しかし，サポートされているのは，もっとも単純なリニアな版構造のみであるのがほとんどで，プログラミング環境として用いるには十分でない．たとえば HAM の場合，基本的なオブジェクトであるノード，リンク，コンテキストに対して作成/修正した時刻が記憶されており，任意の時刻の状態を復元することができる．この方法を枝分れのある版管理に拡張するには，ある時刻での状態を枝の分かれる点として記憶し，分かれた枝はそれ以降別のノードとみなす方法があるが，自然なやり方とはいえない．また TIPE システムのようにリンクに版情報をもたせる方法では，枝分れのある版を扱うことは容易であるが，ネットワーク状になったデータ構造中で，二つのノードからリンクされているノードで版の不一致が起こりうる．TIPE システムではこの問題を回避するために，バックボーンとなる木構造を作るリンクにのみ版情報をもたせ，他は名前により参照する方法をとっているが，ハイパテキストの記述力を弱める結果となっている．その他にもハイパテキストにおける版管理の方法は提案されているが，プログラミング環境として十分な表現力と直感的な分かりやすさを合わせもつ方法は，今後の研究課題であろう．

次に，ハイパテキストのプログラミング環境への適用の将来展望について述べる．4．のはじめで，ハイパテキストのプログラミング環境への応用を，ヘルプシステムへの応用，ソフトウェアデータベースとしての応用，プログラミング言語への応用の三つに分類した．このうちヘルプシステムについては現在でも広く使われており，より一般的な枠組みの中に統合されて，今後とも広く用いられるであろう．

ソフトウェアデータベースとしての応用については，先に述べた版管理の問題など，解決すべき問題点がいくつか残されている．しかし，ソフトウェア開発においてシステム設計や各種の仕様書作成作業の占める比重を考えると，ハイパテキストに基づいた文書作成/読解ツールやアイデアプロセッサの利点が認められそれらが一般的になるにつれて，ソフトウェア開発にかかわるすべての文書をハイパテキストデータベースを用いて管理することになるであろう．その際に，複雑になりがちなプログラミング環境を，いかに整理し分かりやすいものにするかが今後の課題である．

三つめのプログラミング言語への適用については，ノードの粒度により大きく分けて二つのアプローチがある．一つは，TIPE/L のように，ハイパテキストのノードが従来のファイルに対応するというアプローチで，この場合ハイパテキスト構造により記述されるのはモジュール間の関係ということになる．C言語のインクルード機構をハイパテキストのリンクで表すというのも同様のアプローチである．もう一つのアプローチは，さらにノードの粒度を小さくし，アルゴリズムやデータ構造の記述までもハイパテキスト化しようというものである．しかし Dynamic Design にみられるように，元来マシンが解釈できるプログラミング言語にさらにハイパテキストによる明示的な構造を導入することは，効率の問題を除くと冗長な情報を追加しているにすぎないことになり，CPU パワーのさらなる向上が期待される状況においては有力なアプローチとは考えられない．むしろ，ハイパテキストが作る構造を解釈して動くような，新しい概念のプログラミング言語の登場が期待される．ハイパテキストシステムではないが，WEB[21] のアプローチは一つの方向を示している．また，多くのハイパテキストシステムがもつマルチメディア機構を活用した，ハイパテキストベースでより視覚的な，広い意味でのビジュアルプログラミング言語も，今後の研究課題として期待される．

6. ま と め

本稿では，ハイパテキストに関する主要なシステムや研究を概観することで，ハイパテキストとそのプログラミング環境への応用についての現状を紹介した．ハイパテキストが文書の作成/読解ツールをはじめとして普及するのは確実と考えられ，プログラミング環境にもさまざまなインパクトを与えるであろうが，具体的な方法などの研究はまだはじまったばかりで，今後に負うところが大きい．

謝辞 日頃ご指導くださる坂村健先生，ならびに草稿を読み貴重な助言をいただいた西尾信彦氏に感謝いたします．

参考文献

1) Bush, V.: As we may think, Atlantic Monthly 176, 1, pp. 101-108 (July 1945).
2) Conklin, J.: Hypertext: An Introduction and Survey, Computer, Vol. 20, No. 9, pp. 17-41 (Sep. 1987).
3) Engelbert, D.C.: Authorship Provisions in Augment, Proc. CONPCON Spring 1984, IEEE, pp. 465-472 (1984).
4) Halasz, F.G., Moran, T.P. and Trigg, R.H.: Note Cards in a Nutshell, Proc. ACM Conf. on Human Factors in Computing Systems (CHI+GI 1987), pp. 45-52 (Apr. 1987).
5) Goodman, D.: The Complete Hyper Card Handbook, Bantam Books (1987).
6) Akscyn, R.M., McCracken, D.L. and Yoder, E.A.: KMS: A Distributed Hhypermedia System for Managing Knowledge in Organizations, Comm. ACM, Vol. 31, No. 7, pp. 820-835 (July 1988).
7) Yankelovich, N., Meyrowitz, N. and van Dam, A.: Reading and Writing the Eelectronic Book, Computer, Vol. 18, No. 10, pp. 15-30 (Oct. 1985).
8) Trigg, R.H. and Weiser, M.: Textnet: A Network-based Approach to Text Handling, ACM Trans. Office Information Systems, Vol. 4, No. 1, pp. 1-23 (Jan. 1986).
9) Sakamura, K.: BTRON: The Business-oriented Operating System, IEEE Micro, Vol. 7, No. 2, pp. 53-65 (Apr. 1987).
10) Sakamura, K.: TACL: TRON Application Control-flow Language, Proc. 5th TRON Project Symp. 1988, Springer-Verlag, pp. 79-92 (Dec. 1988).
11) Campbell, B. and Goodman, J.M.: HAM: A General Purpose Hypertext Abstract Machine, Comm. ACM, Vol. 31, No. 7, pp. 856-861 (July 1988).
12) Delisle, N. and Schwartz, M.: Neptune: A Hypertext System for CAD Applications, Proc. SIGMOD 1986, pp. 132-143 (1986).
13) 西尾信彦，山田尚勇：発想の計算機支援，情報処理学会研究会文書処理とヒューマンインタフェース，15-2, pp. 1-8 (Nov. 1987).
14) 西尾信彦，小野芳彦，山田尚勇：アイデア・プロセッサには何ができるか？ Proc. 4th Symp. on Human Interface, pp. 1-8 (Nov. 1988).
15) Bigelow, J. and Riley, V.: Manipulating Source Code in Dynamic Design, Proc. Hypertext 1987, pp. 397-408 (Nov. 1987).
16) Bigelow, J.: Hypertext and CASE, IEEE Software, Vol. 5, No. 2, pp. 23-27 (Mar. 1988).
17) Carg, P.K. and Scacchi, W.: On Designing Intelligent Hypertext System for Information Management in Software Engineering, Proc. Hypertext 1987, pp. 409-432 (Nov. 1987).
18) 高田広章：ハイパテキストを用いたプログラムとドキュメントの管理システム，日本ソフトウェア科学会第4回大会論文集，pp. 283-286 (Nov. 1987).
19) 高田広章，坂村 健：TIPE システムにおける構成管理・バージョン管理，トロン技術研究会資料，Vol. 1, No. 2, (社)トロン協会，pp. 21-32 (Oct. 1988).
20) 高田広章：プログラミング言語 TIPE/L の型機構，情報処理学会第30回プログラミング・シンポジウム報告集，pp. 169-180 (Jan. 1989).
21) Knuth, D.E.: Literate Programming, Comput. J., Vol. 27, No. 2, pp. 97-111 (1984).

(平成元年1月23日受付)