HCP チャートエディタ PAN/HCP†

塩見彰睦†† 竹田尚彦†††
河合和久†††† 大岩元††

ソフトウェア開発の上流工程におけるKJ法とHCPチャートの有効性を指摘し，その作業環境支援ツールとして，図式エディタPANを提案する．図式エディタPANは，従来の図式エディタにありがちであった清書作業や，プログラムの自動生成のためのものでなく，実際の図式製作を通じた設計作業自体を効率的に行い，上流工程の試行錯誤的な知的思考を支援するように設計・実現されている．図式エディタPANの画面表示では，カード操作ツールKJエディタで用いた方法を一般化して，計算機上に200文字×150行分の作業領域を用意し，その上に図式をテキスト単位で表現している．この表示形式を導入することによって，統一した操作環境でKJ法の「A型図解」のみならずHCPチャート等のさまざまな図式に対するエディタを実現することが容易になる．PAN/HCPは，自由に図式を配置，編集するための表現形式を用いたHCPチャートエディタであり，紙上で手作業でHCPチャートを記述するのと同様に，設計者が図式の空間配置を考慮しながら，トップダウン記述に拘束されず，自由に記述できる．同エディタのプロトタイプを試作し試用した結果，いくつかの問題点が判明した．現在，これらの欠点を改善したPAN/HCPが完成している．

1. はじめに

ソフトウェア製作には，従来からウォータフォール・モデルによるトップダウン開発が適用されてきた．このモデルでは，要求分析，システム設計，モジュール設計，開発・コーディング，テスト，そして保守・運用という各生産工程に製作が分けられ，それぞれ分業体制で作業が進められる．このうち，要求分析，システム設計やモジュール設計の段階を，特に上流工程と呼ぶ（図1）．

上流工程は，最も慎重に行わなければならないフェーズであり，ソフトウェア開発が成功するか否かを大きく左右する．ところが，実際のソフトウェア開発においては，開発期間が短く設定されることが多く，上流工程の要求分析や設計工程に十分時間がとられないことが少なくない．この結果，必要な分析や設計が十分に行われないまま下流工程へ進んでしまい，仕様変更が生じて，開発効率の低下を招きがちである．

また，多人数で設計作業を進める場合，上流工程で得られた仕様や構想を，いかに誤りなくすべての開発関係者に伝えるかということも重要となる．開発関係者間の知識や価値観の違いによるコミュニケーション・ギャップもまた開発効率を下げる一つの要因となる．

筆者らは，上述の問題を解決するために，上流工程における知的作業を計算機で有効に支援するツールの研究を進めている．図2に筆者らの提案する設計モデルと支援ツールの関係を示す．まず要求分析工程を支援する手法としてKJ法[3]に着目し，ボトムアップな思考過程であるKJ法の作業過程を計算機上で行うことができるカード操作ツールKJエディタ[5]をパーソナルコンピュータ上に実現した．

要求分析作業は，利用者から提出された混沌とした問題意識や曖昧なニーズから，真の目的や要求を秩序ある仕様としてまとめるボトムアップな作業であり，KJ法のパラダイムが有効である[8]．さらにKJエディタの試用実験より，KJ法にこだわらずにKJエディタを単なるカード操作ツールとして使用すれば，モジュールの相互関係や機能の詳細化を行う概略機能設計にも利用できることがわかった[9]．このように，KJエディタを用いてシステム分析を行うと，要求されたソフトウェアの概念や，大まかに分割・詳細化された各機能が，目的（What）として明確にできる[9]．これに加えてインタフェース情報などの各モジュールの関係が記述できれば，目的（What）を仲介とすることで，トップダウンに機能の詳細化を進めることができる．

筆者らは，上流工程の詳細機能設計以降の設計工程

† An HCP Chart Editor PAN/HCP by AKICHIKA SHIOMI (Department of Information and Computer Sciences of Engineering, Toyohashi University of Technology), NAOHIKO TAKEDA (Computer Center, Toyohashi University of Technology), KAZUHISA KAWAI (Department of Knowledge-based Information Engineering, Toyohashi University of Technology) and HAJIME OHIWA (Department of Information and Computer Sciences of Engineering, Toyohashi University of Technology).
†† 豊橋技術科学大学情報工学系
††† 豊橋技術科学大学情報処理センター
†††† 豊橋技術科学大学知識情報工学系

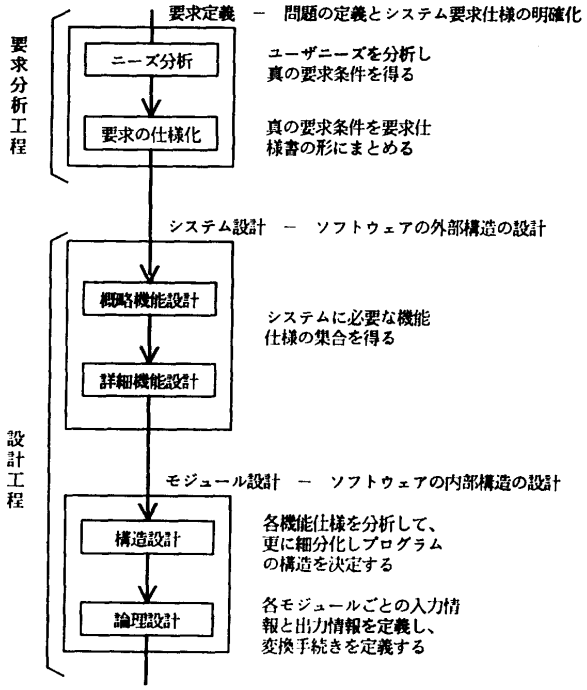

図 1 ソフトウェア製作における上流工程
Fig. 1 Overall design of software product.

を支援する構造化チャートとして HCP チャート設計図法[1]に着目した．HCP チャートは，(1)トップダウンな設計法であり，(2)要求分析においてボトムアップに構築した概念を，目的 (What) と手段 (How) の関係をもとに図式と自然言語を用いて記述することができる．また，(3)設計記述が実際のプログラム構造と対応しているため，設計ドキュメントとして利用できる．(4)HCP チャートは記述に関する規則が少ないので HCP チャートの規則が憶えやすい，という利点がある．特に，規則が少なく，主に自然言語を用いて記述されることから，よく書けた HCP チャートは，プログラミングをしない利用者を含めて，開発関係者全員が読んでわかりやすいという特徴がある．

一方，設計という発想や構想を伴う知的作業は，試行錯誤的に行われることが多く，それは即，図式の書き直しにつながる．しかし，紙上に手書きで作成した図式を修正する場合，追加・挿入・削除といった操作に手間がかかり，ある規模以上の大きさの図式の編集は容易でない．計算機上の図式編集機能は，紙上で作成した図式を加筆修正するよりも，明らかに時間の節約になる．上流工程に図式エディタを導入することで，図式の編集に費やしていた時間を，設計作業に当てることができるようになる．

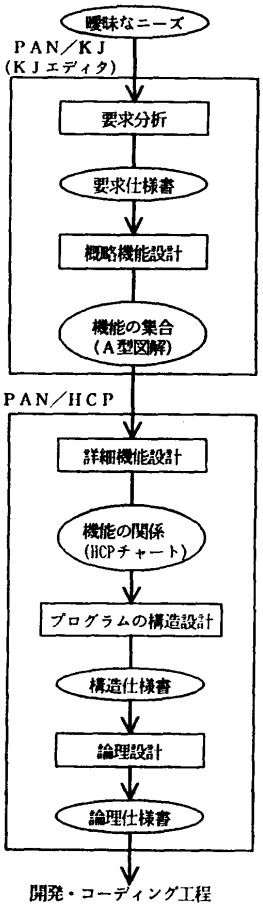

図 2 上流工程の設計モデルとその支援ツール
Fig. 2 Overall design model and supported tools.

既に，プログラムの構造を表現するために，HCP チャートなど構造化チャートが広く用いられ，それらを記述するチャートエディタが実現されているが，それらはプログラムの自動生成や設計文書の清書作業の計算機支援を目指しているものが多い[2),4)]．これらは，プログラムの自動生成の観点からチャートの構造上の誤りを防がなければならないという理由などから，トップダウンにしか記述できない．

また，最近では本研究と同様に，上流工程の設計作業を支援することを目的としたツール——たとえば，SA/SD 手法に基づく Teamwork や Excelerator——も開発されているが，そうしたツールにおいても，図式の記述はトップダウンかボトムアップのいずれかに限定されている．

これに対し，優れた設計者は，経験を活かして既存のアイデアと新しいアイデアの組合せを行いながら新しいソフトウェアを設計することが多い．この時，設

計者の思考は必ずしもトップダウンではなく，当該ソフトウェアの目的に合せたトップダウンな部分と，既存のアイデアと新しいアイデアが錯綜するボトムアップな部分とが共存していると考えられる[9]．

筆者らは，従来の図式エディタにありがちであったプログラムの自動生成や設計文書の清書作業の計算機支援でなく，上流工程における実際の設計作業そのものを計算機で支援することを目指している．そこで，個人の豊かな発想を設計作業に反映できるように，自由に図式を配置，編集するための広い仮想的な作業領域を計算機によって実現する図式の表現形式を提案した[7]．

この表現形式は，カード操作ツール KJ エディタで使用したテキスト単位での図式記述の画面表示を一般化し汎用性を持たせたものであり，HCP チャートをはじめとして，データフロー図，状態遷移図などの図式エディタの画面表示に用いることができる．

この表現形式を用いた HCP チャートエディタ PAN/HCP は，紙上に手作業で HCP チャートを記述するのと同様に，計算機上で設計者が図式の空間配置を自由に記述できる．PAN/HCP を用いると，設計者は必ずしも完全なトップダウン記述をする必要はなく，実現の階層レベルに関係なく書けるところから書いていくことができる．

2章では，設計工程を支援する技法として，HCP チャートが有効であることを指摘する．3章で上流工程の作業支援ツールとして，図式エディタ PAN の図式表現と HCP チャートエディタ PAN/HCP について述べる．4章では，試作した PAN/HCP の試用実験から得られた問題点とその解決について述べる．

2. HCP チャート設計図法による設計作業

HCP チャート設計図法（図 3）は日本電信電話株式会社 (NTT) 電気通信研究所で開発されたプログラム設計法[1] であり，トップダウン詳細化を忠実に実現している．HCP チャートを使うとモジュール設計が効率的に行えるので，DIPS システム等の大規模なソフトウェアの設計に実績がある．

HCP チャートの主な特徴は以下のとおりである．

（1） 処理の階層と制御の明示

目的と手段（What-How）の関係を明確に記述することができる．

（2） データの階層的明示

レコード型のようなデータ構造の階層的意味を表現できる．

（3） データと処理の関係の明示

入出力データ，作業用データを区別し，データと処理とを関係線で結び情報の流れを明確に記述することができる．

（4） 実際のプログラミング構造と対応

できあがった HCP チャートに基づいてコーディングを行うと，設計記述が直接プログラミング構造に反映されるので，ソースプログラムの保守用ドキュメントとしても使用することができる．

ソフトウェア設計のために，さまざまなプログラム図法が提案，利用されているが，その多くは，ソースコードとの対応を重視し，コーディングに近い実体レベルの記述を行うものである．また，そうしたプログラム図法の記述を支援するエディタの多くも，プログラムの自動生成を行うために用いられる場合が多く，一層，コーディングに近い記述をしなければならない．

一方，HCP チャートは，詳細が十分にわかっていない概念レベル（上位レベル）での記述に適しており，手順や方針が非常に細かい実現レベル（下位レベル）の記述には適していない．プログラムのステートメント単位に対応する記述などは，HCP チャート本来の特徴を生かしていないことになる．こうした意味で，HCP チャートは，他のプログラム図法に比較して，概念の詳細化を行う上流の設計工程の支援ツールの枠組みとして，より有効である．

また，HCP チャートは，「目的 (What) をどう (How) 実現する」．さらに，そのためには上位の実現手段を下位の（副）目的ととらえて，「その

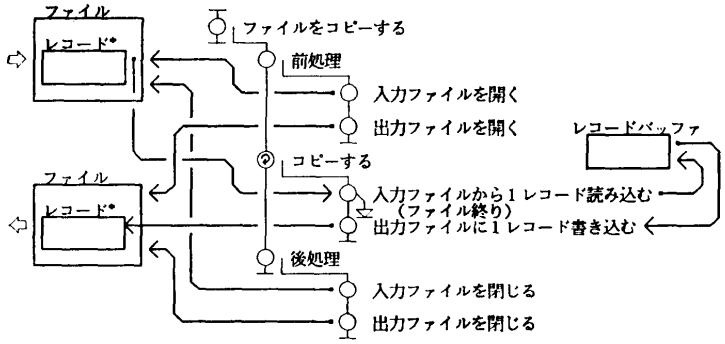

図 3 HCP チャートの例
Fig. 3 A sample HCP chart.

(副) 目的をどう実現する」という具合に，目的と手段を自然言語によって記述する．このため，対象となるシステムの利用者のようなプログラミングをしない者も含めて，関係者全員が読んでわかりやすい設計文書ができる．

また，前章で述べたように要求分析工程で KJ 法を用いて得られた「目的 (What)」を HCP チャートで「どのように (How) 実現するか」という記述がトップダウンにできるため，KJ 法との整合性が高い[6]．逆に，KJ 法を用いた要求分析やトップレベルの機能分析を十分行わないで，HCP チャートのみで設計を行うと，対象ソフトウェア全体の構造が把握されていないため，独立性・階層性の低い枝葉末節にとらわれた設計になる可能性がある．

3. 図式エディタ PAN

筆者らが最終的に目指すところは，種々の専用図式エディタの実現と，それらの有機的な結合である．すなわち，これらの専用図式エディタをシステム的に統合し，ソフトウェアの設計環境を整備することである．現在，開発・計画中の専用図式システムの一覧を表 1 に示す．これらの図式専用エディタを用いて，KJ エディタで要求分析工程および概略設計工程，HCP チャートエディタ，ER 図エディタ，データフロー図エディタで構造化設計工程，状態遷移図エディタでリアルタイム構造化分析工程を支援することが有効と考えている．

ここで重要な点は，それぞれの図式エディタが使いやすいユーザインタフェースを提供し，なおかつ，各エディタ間でそのユーザインタフェースが統一されていることである．ユーザインタフェースを統一することで，どの図式エディタも同じような操作で図式を編集することが可能になり，容易に使い方を学習できる．

3.1 節では，設計工程の観点から図式の特性と，その計算機化における問題点について議論し，3.2 節ではその問題点を解決するための図式エディタの表現形式について述べる．さらに 3.3 節では，この表現形式を用いた HCP チャートエディタ PAN/HCP の特徴と機能について述べる．

表 1 PAN 図式専用エディタ
Table 1 Chart editing tools of PAN.

図式種別	専用システム
KJ法 (A型図解)	PAN/KJ(KJ エディタ)
HCP チャート	PAN/HCP
状態遷移図	PAN/STN
データフロー図	PAN/DFD
E・R図	PAN/ER

3.1 図式の特性とその計算機化

設計作業の観点から見たとき，以下に詳述するように図式は，1)広い作業領域，2)図式の規格化，という二つの特性を持っている必要がある．これらの特性は，図式編集の計算機化において，助長（進展）あるいは保持されなければならない．

KJ 法では大きな机を，HCP チャート記述でも最低 A3 程度の用紙を用いなければ，作業が効率的に行えない．これは，取り扱う問題の大小にかかわらず，ある問題を図式で表現するために，図式要素を操作し，試行錯誤しながら，図式全体をまとめていくのに必要な最低限の広さであり，経験的に得られた広さである．

HCPチャート設計図法では，これよりも大きな作業領域を用意すると，1 枚の図に多くの記述を書きすぎてしまい，適切なモジュール化が行われなくなるという弊害が起る．一方，小さすぎると図式の編集過程での図式の移動，交換等の作業ができなくなる．編集作業に必要な作業領域を確保し，実用に供し得る記述ができる広さとして，筆者らは，200 字×150 行（全角）の作業領域を用意している．

図式は，図式要素間の隣接関係の表現力に優れており，さらに一覧できることで各図式要素間の関係や全体の構造を把握することが容易になる．したがって，何ページにもわたる文章による一次元的な記述を読むよりも，図式を用いてよく整理された二次元的な記述を読む（見る）方がはるかに理解しやすい．

重要な点は，描いている図式の全体を一覧できることである．図式全体が一覧できなければ，図式の空間配置や図式要素間の関係を把握することが設計者の負担となり，設計作業の妨げとなる．図式の持つ特性のうち計算機化に際して最大の問題となるのは，いかにして図式を一覧できる広い作業領域を実現するかである．

一方，規格化された図式を用いることで，図式とその記述に対する定義が明確に行われるため，厳密でかつ，わかりやすい記述が可能となる．

多人数で設計作業を進める場合に，関係者が図式に対する共通な知識を持っていれば，図式を通して設計作業および設計レビューを行うことで，互いの意志疎

通が正確に行える．規格化された図式を用いた設計作業は，ソフトウェア開発時のコミュニケーション・ギャップを埋める有効な方法となる．

3.2 図式エディタ PAN の表現形式

計算機上の限られた画面の枠内で，広い作業領域の一覧性を確保するための方法として，市販ソフトウェア等に広く用いられているものに，縮小表示とスクロールバーの二つの方法がある．

縮小表示は，グラフィック（ビットマップ）画面を使用し，文字や図を縮小して作業領域を表示する表示方法である．縮小表示方式では，図式の空間配置を把握するために，作業領域全体の図式の構造を容易に見ることができるという利点がある．しかし，解像度が粗く図や文字がわかりづらくなり，編集作業におけるポインティング作業が難しくなる．

図式の配置が把握できるという最低限の要求に対して，パーソナルコンピュータの標準的な画面解像度640×400ドットでは，縮小表示を用いてもＢ４用紙１枚分が限度であり，図式エディタとしては十分ではない．さらに，文字は小さくなり判読するには不十分である．

また，縮小表示方式では，一般に複数の縮小率が用意され，図式全体の概要と，一部分の詳細な内容をそれぞれ表示できるよう工夫されていることが多いが，縮小率を切り換えて表示するための操作や画面の書き換えなどのインタラクションが生じ，設計作業の思考を妨げ快適な作業環境とは言えない．

一方，スクロールバー方式は，垂直，水平２本のスクロールバーが用意され，これを操作することで画面表示位置を変更する方法である．スクロールバー方式は，画面全体に図式の一部分を詳細に表示できる利点がある．しかし，作業領域全体を把握するために表示位置を変えるには２本のスクロールーバーを操作しなければならず，縮小表示と同様にインタラクションが生じ，思考を妨げてしまう．スクロールバー方式でも Mac Draw などは，マウスカーソルが画面外に出たきにも画面がスクロールするが，ビットマップ画面の書き換えのため画面にちらつきが生じる．

上流工程を支援するツールである，Teamwork や Excelerator においても，Teamwork はスクロールバー方式を，Excelerator は縮小表示方式を用いており，上述のような操作面での問題点も有している[11]．

このほか「多視点遠近画法[12]」などの画面表示方法があるが，図を写像関数を用いて画面にマッピングするため描画速度に実用上の問題を残している．

図式エディタ PAN の表現形式では，グラフィック画面に比べ，表示速度の早いテキスト画面を用いる．PAN で使用する図式はすべて，テキスト画面で表示できる文字列の集合で構成することとし，特殊な記号，文字，図形は外字として取り扱う．図をテキスト画面に表示するため，グラフィック画面に描く図式に比べ表現力や品質は落ちるが，HCP チャートのような図式の表記では，実用上十分な表現が可能である．

PAN では，この表示方法を実現するために，まずメモリ上に全角文字で 200 文字×150 行分の仮想画面領域を取り，Ａ３サイズを越える図式を表現できるようにした．そして，実際に画面に表示する図式は，仮想作業領域の約 1/30 の大きさで，40 文字×23 行（全角文字）をテキスト画面に送り，画面上に表示する（図 4）．

全領域を見るために，テキスト画面が表示する内容を上下左右の全方向にスクロールさせる機能[5]が用意されており，この機能をマウスの動きに連動して，仮想画面領域からテキスト画面に送るデータを更新していくことによって実現する．この機能を筆者らはパニングと呼ぶ．図形要素の移動は仮想画面上の対応するデータを移動方向に書き換え，その後仮想画面の対応領域をテキスト画面に送ればよい．テキスト画面には，文字コードを転送するので，グラフィック画面よりも少ないデータの転送で済むため，高速な画面書き換えが可能になる．PAN では，画面の動きがマウス

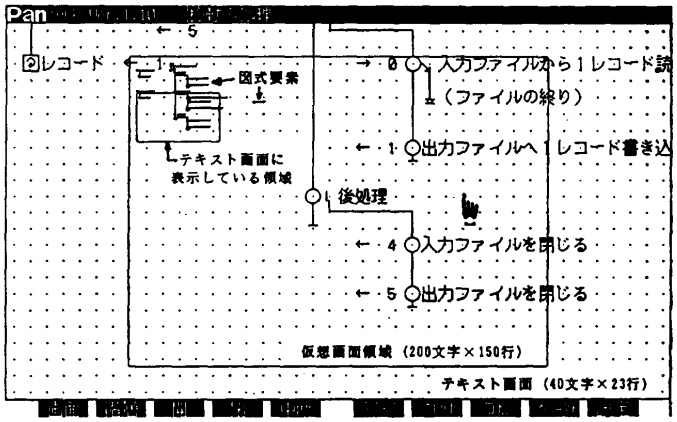

図 4 仮想画面とテキスト画面
Fig. 4 Virtual screen and text screen.

の指示に対して，ほとんど遅れなしに実行される．

しかし，テキスト画面を用いたのでは図式全体を表示できないので，全体の位置関係をグラフィック画面に表示し，上述のテキスト画面とグラフィック画面を同時表示することによって，一覧性の一助としている*．この表現形式は，あくまで，一覧性の実現に対する次善の策と筆者らは考えており，理想的には，仮想作業領域全体をそのまま表示できる（3200×2400ドット）ような，高解像度ディスプレイを用いることが望まれる．

以上のような画面表現は，KJ エディタ[5]で実現されていた．しかし，システムの論理構造は，表示速度とメモリの使用効率を重視しているため，システムが完全にモジュール化されてはいない（**図 5**(a)）．

特に，KJ エディタでは，画面表示部のデータ構造にセルリンクと呼ばれるリスト構造を用いて図式要素の重なりを表現しているが，そのリスト構造には，KJ エディタのラインセルや関係線セルといった各種図式に依存した図式専用のデータ構造が用いられている．このため，ディスプレイ表示モジュールも図式に依存し，画面表示の時，各セルからどの図式を表示するかを判断し文字コードを画面に表示しなければならなかった．画面表示部が図式に依存してしまうと，各種専用図式エディタを実現する際に，それぞれの画面表示部を新たに作らなければならない．これは，既にあるものを改修するだけでもかなりの労力を要する．したがって，パニング，ディスプレイ表示の手続きを図式から独立させることが，一連の図式エディタの開発の効率を向上させる上で重要な課題となる．

KJ エディタのシステムの内部は，画面表示部とカード等の図式要素管理部に分類することができる．そこで，図式エディタ PAN の表現形式では，画面表示部の図式に依存していたラインセル，関係線セル，フェースセル，エッジセルのデータ構造を図5(b)に示すように統一して図式セルとする．図式セルには，図式の種類を識別する図式識別子と画面表示文字コードを持たせる．

ディスプレイ表示を行うモジュールは，画面表示を行う時，そのセルがどの図式であるか知る必要がなく，画面表示文字をそのままテキスト画面に転送するだけでよい．

さらに，KJ エディタでの操作性を保ち，かつ，汎用性を持たせるために，KJ エディタの論理構造を見直し，画面表示部と図式要素管理部を切り離すようにモジュール化を行い図5(b)に示すようなシステム構成にした．セルリンクへのアクセスは画面表示部のモジュールを通して行われるため，図式要素管理部から独立することができる．

この結果，画面表示速度とメモリの使用効率が多少犠牲になるが，すべての図示要素を統一したデータ構造で取り扱うことができることになった．さらに外字セットと図式要素管理部を差し換えるだけで，HCP

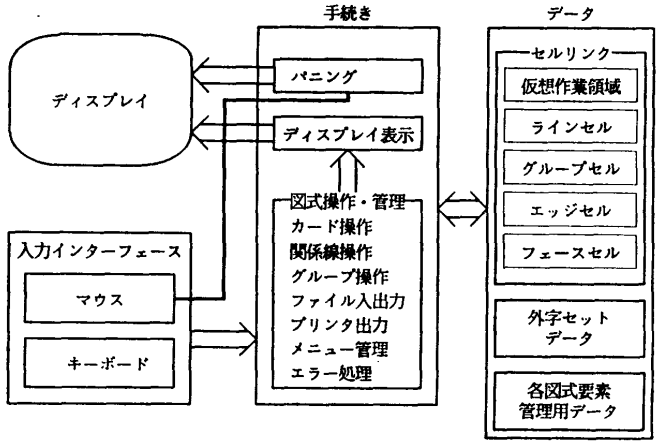

図 5(a) KJ エディタのシステム構成
Fig. 5(a) System configuration of the KJ-editor.

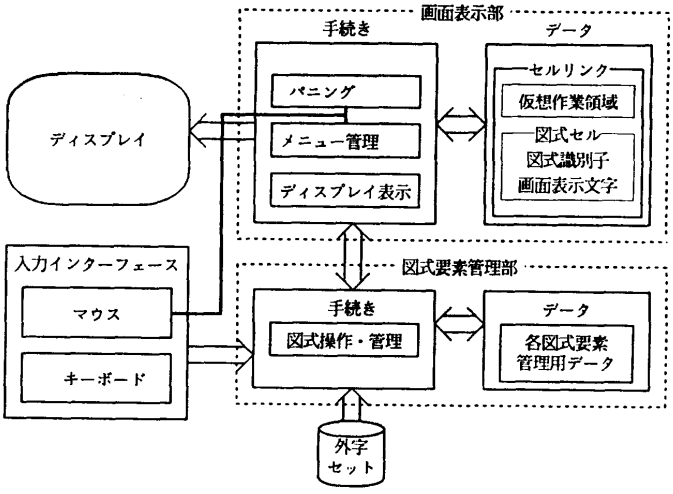

図 5(b) 図式エディタ PAN のシステム構成
Fig. 5(b) System configuration of PAN.

* 表現形式 PAN の一覧性を評価する実験を行ったところ，紙上で得られるような一覧性が，PAN では得られないことが確認されている[10]．

チャート，KJ 法の A 型図解，データフロー図などの表示が可能になり，他の図式エディタの画面表示に柔軟に対応できるようになった．また，パニングやコマンドの選択といった基本操作の統一を図ることができる．操作を統一することは，要求分析工程から設計工程までのソフトウェア製作の統合的な環境へと発展させていくうえで実用上重要である．

KJ 法の A 型図解，HCP チャートをはじめとして，データフロー図，状態遷移図などの，今後開発予定の図式エディタでは，すべてこの表現形式を使用することで，外字の定義と図式要素管理部を開発するだけですべての図式エディタを開発できることになった．

3.3 HCP チャートエディタ PAN/HCP

筆者らは，設計時における設計者の試行錯誤的な設計作業そのものを図式エディタで支援することを目標としている．この試行錯誤的な知的作業を有効に支援するために，HCP チャートの紙上での手作業による記述方法に着目した．手作業による記述方法の特徴に対して，これを支援するには以下に述べる機能が必要となる．

（1） 紙上における手作業と同等の操作性

HCP チャートは，定規やテンプレートを使わない(フリーハンド)ことを原則としているため，設計に集中でき，チャートの記述性が向上する．しかし，設計工程において，設計者の思考，知的作業は試行錯誤的に行われるので，その結果チャートは何度もの書き換えが必要となる．ところが，紙上における手作業での修正は手間がかかるため，設計作業自体の修正を最小限にすませるといった視点から行ってしまいがちである．PAN/HCP では，チャートの修正・保守のために階層をまとめて移動したり，別のチャートにモジュール化するような編集機能を用意した．

（2） 配置情報を利用するための自由な空間配置

現在，HCP チャートエディタとしては，筆者らの研究室で以前開発した HCP チャートエディタ[2]や NTT ソフトウェア研究所の HD システム[4]などがある．これらのシステムの画面表示は限られた画面内にできるだけ多くの情報を表示するように HCP チャート内の 1 項目を必ず 1 行に表示し，それを詰め合せて空行などを設けられないようにしている．

しかし，設計者が，実際に紙上で HCP チャートを記述していく時には，下の階層で詳細化するチャートの量を見積もり，詳細化する部分の空白を取りながら記述を進めていく(**図 6**)．そこで筆者らは，思考の過程が図式の空間位置に反映されると考え，図式を記述する際の図式の配置の制限を極力少なくし，設計者が記述したいように記述できるように努めた．また，これによりチャートの記述を途中で中断しても，チャートの空間配置に設計途中の思考過程が反映されているので，再開したときに図式の位置情報を手がかりに中断した時の思考の過程を思い出すことができる．

こうした配置情報を保持する機能は，図式を用いた設計作業の反復においては，特に有効と一般的に考えられているようで，上流工程支援ツール Teamwork や Excelerator[11]等も，同様の機能を有している．

（3） トップダウンにこだわらない．

筆者らがかつて開発した HCP チャートエディタの記述方法は，HCP チャートを完全にトップダウンな形でのみ記述するように，書き込める場所が制限されている．しかし，設計にともなう思考は必ずしもトップダウンなものではなく，ときには，一番下の階層の処理手順や，より下の階層の一連の処理が思い浮んだりすることがある．

PAN/HCP では，**図 7** に示すような関係付けがされていない図式の記述を許し，上述のような場合，トップダウンの記述をするのではなく，より下の階層であると考えられる処理を，右側に書いておき，チャートを構成していく際に，その処理手順の断片を移動，

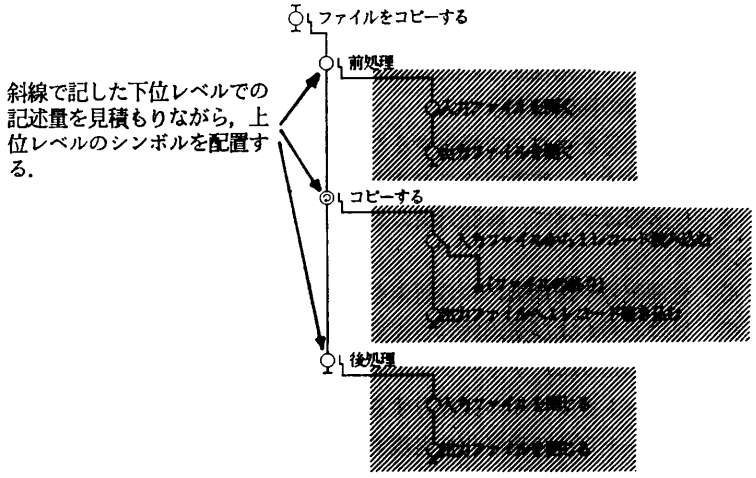

図 6 詳細化の空間配置の見積り
Fig. 6 Estimation of layout detail.

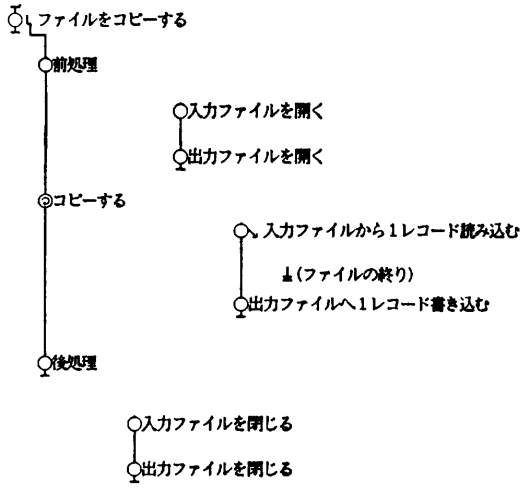

図 7 非トップダウン記述
Fig. 7 Non-topdown description.

関係付けを行って，記述を進めることができる．すなわち，設計者が書けるところ，あるいは，書きたいところから自由に記述できる方法を，PAN/HCP では採用した．この方法により，処理手順の断片は関係付けが行われていなくても，それにふさわしい階層の位置に配置できるので，処理内容の展開が進めやすい．また，思い浮んだ処理をその都度，チャート中に記述しておけば，設計者は，憶えておかなくてもよく，それだけ設計に集中することができる．

PAN/HCP では，紙上における手作業と同様の記述環境を計算機上で提供するために次に挙げる機能を用意した．

1) 処理記号をポップアップメニューから選択し，処理の内容である処理記述をキーボードから入力する．これは，作業領域中の任意の位置にいつでも自由に行える．

2) 処理記号と処理記号の間に線を引き，処理の流れや処理内容の展開を明示する．

3) 処理記号をマウスで操作することによって任意の場所に移動する．移動対象の処理記号に付属している下位階層はすべて，移動対象の処理記号に追従する．

4) 処理記号や関係線等の各要素を消去する．

5) 処理記号および処理内容を変更する．

6) 編集した HCP チャートを印刷する．

7) チャートをファイルに保存する．

8) チャート自体の階層化を実現するある処理内容を，別のチャートで展開，詳細化している場合，その処理記述内に，その内容を記述しているチャート名(厳密にはファイル名) を入れられるようにしている．これにより，現在編集中の HCP チャートから，別の HCP チャートの参照，編集が可能になる．すなわち，エディタを終了させることなく，別のチャートでモジュールを詳細記述できる．

4. HCP チャートエディタの試用と評価

3章で述べた機能を持つ HCP チャートエディタ PAN/HCP のプロトタイプをパーソナルコンピュータ PC-9801 上に，C言語を用いて試作した．さらに，その評価のために，プロトタイプ版 PAN/HCP のいくつかの関数を同エディタで記述・編集し，要求内容が満足されているか，また，ソフトウェア設計を効率よく支援できるかどうかを考察した．

その結果，次のような，機能的要因，人間的要因による問題点があることが分かった．以下に問題点とその解決を述べる．

4.1 機能的要因について

(1) データ構造の記述ができない．

プロトタイプ版では，HCP チャートを用いて思考やアルゴリズムを整理することを目標としたため，とりあえずデータ構造の記述機能を組み込まなかった．しかし，これではチャートを更新するたびに出力されたチャートに対して，手書きで加筆しなければならず大変不便であった．

そこで，PAN/HCP ではデータ構造の記述機能を新たに組み込んだ．HCP チャート本来のデータ構造の書き方は，図8に示すように，データ構造を表す枠とそれに付随するデータの名称を示す文字列で表される．この表記ではデータが多重の入れ子構造になると，データ構造の表現が繁雑になる．そこでデータの記号を新しく設け，処理の記述と同様にデータ構造を

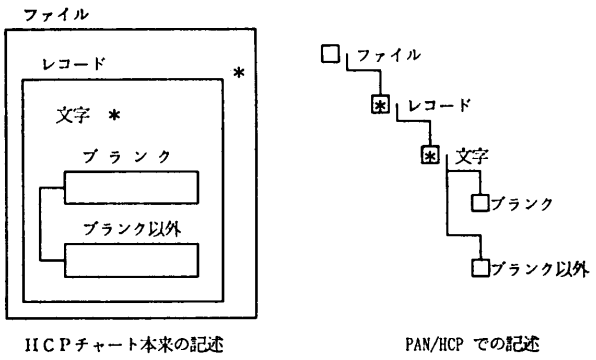

図 8 データ構造の記述
Fig. 8 Notation of the data structure.

表現することにした．この記述表現は，HDシステム[4]で使用されているものである．これにより，処理とデータ構造の記述や編集操作は統一され，一つのドキュメント体系の中で一貫した記法で記述できる．この記述方式は，従来の HCP チャートのもつデータ記述書式と同等の表現力を有している．

しかし，上流工程では，システム全体でのデータや情報の流れを把握した上で，データ構造を設計する必要があり，HCP チャートの表現力では力不足である．したがって，データの流れを記述できるデータフロー図をまず利用するのが有効と考えており，その作成を支援する DFD エディタ PAN/DFD を統一した表現形式で実現することを計画している．

（2） 大規模なチャートでは空間配置機能が有効でない．

処理の記述が 20 個程度の小規模なチャートであれば，空間配置をたよりに設計途中の思考過程を思い出すことができるが，それを越える大規模なチャートになると思考過程を思い出すことが難しくなることが，実際の使用によって明らかになった．設計時に設計者が，どの程度処理を詳細化するかを見積もって空けた空間を生かせないことになる．

この問題を解決するために，設計者の意図や思考過程をより効果的に残せるようチャート上に，設計者のコメント，メモや補足説明などが書けるカードを導入し，改善を図った．設計者が思いついたことやメモを積極的にカードに書き込むことで思考過程をチャートに残すことが可能になった．

また，大規模なチャートでは，ディスプレイ上で空間配置そのものを把握することが難しく，紙に出力したものが必要となる*．これは，表現形式 PAN によっても，一覧性が十分確保されないことを示すものであり，今後の大きな研究課題である．

（3） 使い勝手が悪く思考が中断するときがある．

チャート編集の際に，メニューや処理記号を選択するマウスの操作が頻繁に行われ，その結果，思考が中断され設計作業に集中できない．

例えば，処理を記述する場合には，プロトタイプ版では以下の手順が必要であった．

1) 処理記号を置く位置を指定する（クリック1回）．

2) 処理記号をポップアップメニューから選択する（ドラッグ1回）．

3) 処理内容を書き込む範囲を指定する（マウスの移動とクリック1回）．

4) 処理内容をキーボードから入力する．

5) 処理内容の入力が終了したことを示す（クリック1回）．

この場合では，処理内容を記述できるまでに，クリック2回とマウスの移動2回（ドラッグ1回を含む）のマウス操作が必要になる．

このように，ツールのユーザインタフェースが悪いと，操作が繁雑になる．繁雑な操作をすると，思考が編集操作のために中断してしまい，本来の設計作業に集中できない．

編集時のユーザインタフェースに関しては，設計作業中に思考活動の妨げとなるマウスやキー操作の回数が最小限となるようにユーザインタフェースを改善した．具体的には，前述の処理記述は現在の PAN/HCP では，以下の手順で行われる．

1) 処理記号をポップアップメニューから選択する（ドラッグ1回）．

2) 処理記号を置く位置を指定する（クリック1回）．

3) 処理内容をキーボードから入力する．

4) 処理内容の入力が終了したことを示す（クリック1回）．

プロトタイプ版に比べマウスの移動とクリックの回数をともに1回ずつ削減することができた．さらに，2回目以降の書き込みは 2),3),4) の手順を繰り返すことで，1) の手順も削減できる．

4.2 人間的要因について

（1） 印刷時のイメージを意識すると，設計に集中できないことがある．

PAN/HCP では，記述した図式を WYSIWYG (What You See Is What You Get) 形式で表示するため，アルゴリズムの設計や概念の詳細化などの作業中に，図式を美しく書こうとする，いわゆる清書作業のほうに意識が集中してしまい，本来の設計作業のほうに集中できないことがある．これは，ユーザである設計者の設計に対する意識の問題であり，本質的な解決には，ユーザの教育などから検討せざるをえない．

この問題に関して，記述された複雑に入り組んだ HCP チャートの木構造の関係を保持したまま，図式を再配置，整理して出力する機能，いわゆる自動清書機能が有効である．しかし，設計を進めてきたチャー

* これは，3.2 節の脚注に述べた評価実験と同様の結果である．

トを清書してしまうと，チャート中に残されたそれまでの思考過程は消えてしまうこととなり，当初の設計方針に反する．そこで，HCP チャートの完成時に清書出力を得るような自動清書機能を設ける方向で考えている．

（2） 不完全なチャートや誤ったチャートができてしまう．

図式の配置に関しては設計者に任せてあるため，出口の記号がなかったり，振分け処理でないにも関わらず，下階層で振分けの記述が書かれてしまうような不完全なチャートや誤ったチャートができてしまう．また，出口の記号が設計者の意図していない場所に戻るような記述になっているチャートができてしまう．

完成したチャートが，不完全であったり，誤っていたりすると，システム設計以降の工程に大きく影響し，効率のよいプログラム作成ができない．そればかりか，システム設計自体の信頼性に関わる問題となる．

そこで，チャートの構造的な誤りをチェックする機能が必要となる．

このチェック機能は編集時に逐次チェックするのではなく，利用者が編集を終ったときに，利用者の指示により構造の誤りを検出するのが良い．これは，利用者が自由に記述できるという利点を生かしながら，チャートの構造的な誤りをシステム側でチェックするための配慮である．

チェックをする事項はエラー事項と警告事項とに大別できる．

エラー事項は，HCP チャートとして不完全であったり，HCP チャートの記述規則とは明らかに誤っているものである．これは，まだチャートが書きかけであったり，HCP チャートのシンボルの意味を取り違えて書いた場合にチャート上に存在する．

一方，警告事項は，HCP チャートとしては形になっているが「作法」として適当でなかったり，エラーかどうか判断できないが間違っている可能性がある部分である．これは，まだ HCP チャートの書き方をよく把握していない人がそれらしく書いたが読みにくいチャートとなっている場合や，本来の HCP チャートの提唱している書き方からはずれるが便利なので故意に変形した方法で書いた等の場合がある．

現在，PAN/HCP に組み込まれているチェック事項は以下のとおりである．

（i） エラー事項

・出入口の有無
出入口のつけ忘れがないかを調べる．

・戻りの深さ
多段戻りで，戻り位置があるか（段数が大きすぎないか）を調べる．
出口の書き間違いを発見する．

・振分け処理が書かれている処理シンボルが正しいか

・最下位レベルにのみ表れるシンボル（モジュール呼出し）に下階層がないか

・最下位レベルとはならないシンボル（振分け，判断）に下階層があるか
下階層があるべきシンボルに下階層がなければ，それは，主に書き忘れていることが原因と思われる．

・処理構造の分断
処理構造の木が一つ以上あるかどうかを調べる．このチェックにより，後で接続しようと思っていて忘れていた処理構造や，単独で存在する処理シンボルを発見する．

（ii） 警告事項

・第1レベルの処理が二つ以上あるか

・処理の連接数の過多
処理の連接数が多すぎるのは，処理を階層的に記述できていないことが原因であることが多い．よって，ある数以上処理が連接していると警告を発生させることにした．当然，いくつもの処理を逐次実行する場合は連接する処理数が多くなる．

（3） 階層性のないチャートを書いてしまう．

設計時に処理の細部にこだわってしまうと，全体の構造よりも，部分的な低い階層での枝葉末節な構造にとらわれてしまう．その結果，上の階層で記述した本来の目的が下の階層で展開されていない曖昧な，意味のない，ただ形だけの階層構造になってしまう．また，実現方法のみを連続的に記述したような，階層性のないチャートができる可能性もある．

階層性のないチャートを書いてしまうことに関する対策としては，設計者が階層性を意識しやすいように階層ごとに色分けをして表示するモードを設けた．

5. おわりに

ソフトウェア製作の上流工程である要求分析工程と設計工程を支援するための手法に関して議論し，その

手法を計算機で図式として効果的に支援する図式エディタ PAN の表現形式を提案した.

この表現形式は,カード操作ツール KJ エディタの画面表示部分と外部仕様的には同等であるが,HCP チャート等の他の図式エディタにも使用できるように一般化し,汎用性を持たせた操作・画面表現の形式である.筆者らはこの表現形式を用いて,統一した操作環境で図式一般の編集を可能にすることを目指している.規格化された図式で記述した図は,可読性が高いため他の設計問題への再利用が可能であり,そのためのデータベース化が進めやすいことが期待される.

HCP チャート設計図法を用いて設計工程を支援する HCP チャートエディタ PAN/HCP のプロトタイプを試作し,試用・評価を行った.

その結果,HCP チャートエディタに紙上における手作業と同等の操作性を持たせることで,小規模なチャートでは設計者の思考過程を反映させることができた.しかし,大規模なチャートにおいて,これだけでは設計作業が効率よく行えない場合があることがわかったので,メモ書き用カードの機能を付け加えることとした.

一方,人間的な要因により,設計や詳細化の作業中に清書作業のほうに意識が集中してしまい,本来の設計作業に集中できない場合があることもわかった.また,階層性のないチャートや不完全なチャートのできてしまう問題点があることがわかった.この問題点に対して,構造的に誤りのあるチャートをチェックする機能を付加し,設計者に階層性を意識させるための補助機能として階層ごとに色分けするモードを設けた.

今後は,3章で述べた各種図式専用エディタを実現し,要求分析工程から設計工程までのソフトウェア製作の上流工程の支援環境を整備していく予定である.

謝辞 筆者らの図式エディタに関する考えを検討し,その実現にご協力いただいた日本電信電話株式会社の故花田収悦,寺島信義,米田実男,小原永の諸氏に深く感謝します.

PAN/HCP の実現にご協力いただいた豊橋技術科学大学 情報工学系 大岩研究室の中神明氏,本研究に関し討論いただいた同研究室の諸氏に感謝します.

多くの有益なご助言をいただいた査読者の方々に感謝します.

参考文献

1) 花田収悦:ソフトウェア設計図法,企画センター (1983).
2) 山本貴博,大岩 元:ソフトウェア設計開発支援システム,情報処理学会ソフトウェア工学研究会, No. 40, 40-5 (1985).
3) 川喜田二郎:KJ 法—混沌をして語らしめる—,中央公論社 (1986).
4) NTT ソフトウェア研究所:HD システムマニュアル (1986).
5) 小山雅庸,河合和久,大岩 元:発想支援ツール KJ エディタの設計,第 34 回情報処理学会全国大会論文集, 5K-9, pp. 1529-1530 (1987).
6) 大岩 元,河合和久,竹田尚彦,小山雅庸,塩見彰睦:ソフトウェア基本設計のためのソフトウェア・ツール,日本ソフトウェア科学会第 4 回大会, C-4-5 (1987).
7) 河合和久,塩見彰睦,竹田尚彦,大岩 元:ソフトウェア設計のための図式エディタシステム,第 36 回情報処理学会全国大会論文集, 2L-1, pp. 879-880 (1988).
8) 藤野喜一,花田収悦:ソフトウェア生産技術, pp. 15-31,電子情報通信学会 (1989).
9) 竹田尚彦,河合和久,大岩 元:KJ エディタを用いたソフトウェア設計者の思考過程分析の一方法,ソフトウェア・シンポジウム '90, pp. 105-109 (1990).
10) Ohiwa, H.: Idea Processor and the KJ Method, *J. Inf. Process.*, Vol. 13, No. 1, pp. 44-48 (1990).
11) Fisher, A. S.(著),黒田純一郎,中島 誠(訳): CASE—ソフトウェア開発には CASE ツールを使って—, pp. 134-139, 146-151,共立出版(1990).
12) 三末和男,杉山公造:図的思考支援を目的とした図の多視点遠近画法について,情報処理学会論文誌, Vol. 32, No. 8, pp. 997-1005 (1991).

(平成 3 年 7 月 8 日受付)
(平成 3 年 12 月 9 日採録)

塩見 彰睦（正会員）

1965 年生．1985 年徳山工業高等専門学校情報電子工学科卒業．同年豊橋技術科学大学情報工学課程 3 年次編入．1989 年同大大学院工学研究科情報工学専攻修了．工学修士．現在，同大大学院工学研究科博士後期課程に在学中．ソフトウェア工学，HCI，日本語入力の研究に従事．CASE 環境，HCI に興味をもつ．日本ソフトウェア科学会会員．

竹田 尚彦（正会員）

1958 年生．1982 年名城大学工学部電気工学科卒業．同年，(株)金陵入社．メカトロニクス関連ソフトウェアの開発に従事．1984 年より社員資格のまま豊橋技術科学大学大学院へ進学．1990 年同大学博士後期課程単位取得退学．工学修士．現在，同大学情報処理センター助手．HCI，ソフトウェア工学に関する研究に従事．ソフトウェア開発における人間的要因について興味をもつ．

河合 和久（正会員）

1958 年生．1981 年大阪大学基礎工学部情報工学科卒業．1986 年同大大学院基礎工学研究科博士課程修了．工学博士．同年豊橋技術科学大学情報工学系助手．1991 年同大知識工学系講師，現在に至る．計算機を用いた教育，創造的活動における計算機支援，HCI などの研究に従事．電子情報通信学会，人工知能学会，AAAI，日本ソフトウェア科学会などの会員．

大岩 元（正会員）

1942 年生．1965 年東京大学理学部物理学科卒業．1971 年同大学院博士課程修了．理学博士．同年東京大学理学部助手．1978 年豊橋技術科学大学情報工学系講師．1980 年同助教授．1985 年同教授．1974〜76 年英国ケンブリッジ大学キャベンディッシュ研究所客員研究員（ブリティッシュ・カウンシル・スカラー），1979〜80 年米国コーネル大学応用物理学科客員準教授．荷電粒子光学系，キーボード入力，ソフトウェア工学などの研究に従事．