卒業論文　　2001年度(平成13年度)

入力履歴を利用したユーザ入力支援機構の構築

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　　政樹

慶應義塾大学 環境情報学部

大藤　徹

卒業論文要旨　　2001年度(平成13年度)

入力履歴を利用したユーザ入力支援機構の構築

本研究ではコンピュータシステム上でユーザ入力効率を改善するためにユーザ入力の履歴を利用したユーザ入力支援機構を提案する．また本研究で提案する手法の有用性を実証するためシステムの設計，構築，評価を行う．

従来の入力支援機構では，ユーザ入力の先読み機構が予測に失敗した場合，ユーザの労力が増加する問題点があった．また機能のショートカットにはユーザに多量の記憶を要請する．このような入力支援機構ではユーザの入力を支援する反面，ユーザにとって機構を利用するデメリットが無視できない．

本研究では上記の問題点を解決するため，以下に述べる四点に重点を置き，ユーザ入力支援機構を構築した．第1にコンピュータの利用環境をユーザが交換した場合にも交換前のユーザ入力支援機構の利用経験を有効に活用することができる，第2にユーザ入力支援機構を利用するために高度な技術，記憶を必要としない，第3にユーザ入力支援機構の動作がユーザにとって予測可能，第4に過去のユーザ入力を再利用可能，の4点である．

ユーザは本システムを利用することにより過去に行った入力を即座に再利用できる．そのためユーザは繰り返し作業を行う必要が少なくなり入力効率を改善できる．また本システムの方式は以下のような今までの入力支援機構では解決できなかった状況においても効果的である．第1に複数のアプリケーションにまたがる状況，第2にスクリプトファイルを作成するほど複雑でなく，かつ比較的よく使用するけれども集中して行わない作業を行う状況である．

本論文では，まず既存の入力支援機構の特徴と問題点を挙げ，問題点を解決する手段としてユーザ入力履歴の利用を提案する．次に本研究で提案されたシステムの設計と実装方法を述べる．最後にシステムの評価を行い本研究の有用性を実証する．

慶應義塾大学　環境情報学部
大藤 徹

Abstract of Bachelor's Thesis

A User Input Support Mechanism Using Input History

In this research, we propose a user input support mechanism to improve user input efficiency by using user input history. To prove the userfulness of the system proposed, we present the design, implementation and evaluation of the system.

In former input support mechanisms, users were burdened when the system failed to predict the input correctly, or when much memorizing was necessary to use the system. To solve these problem above, we propose a user input support mechanism, which focuses on four features. First, even if the user's environment changes, the user can continue to use this system. Second, the system does not require user technique or user memory. Third, the user can predict the system operation. Fourth, the user can utilize past user inputs by the system.

Using the system proposed in this paper, past user input is utilized to simplify user input. This system is effective in situations where the user has to repeat a certain routine. It is not necessary for the user to write script files, or perform difficult tasks in order to use this system.

In this paper, we describe the problems and characteristics of former input support mechanisms and propose a method of utilizing the user input history to solve these problems. Next, we describe the system design and implementation. Finally we evaluate the system and prove the usefulness of this research.

Tetsu Ohtou

Faculty of Environmental Information
Keio University

目 次

図 目 次

表 目 次

第1章　序論

本章では，まず本研究に関連した技術背景を挙げ，それらの技術を利用することによって生じる問題点を示す．そして本研究の目的と本論文の構成について概略を述べる．

1.1 本研究の背景

近年，コンピュータが一般に普及し，洗濯機，掃除機，アイロンなど生活必需品にもマイクロプロセッサが搭載されるのが通常となった．それら家庭用品に使われているものも含めると，身の回りでコンピュータと関係のないものは存在しないと言っても過言ではない．今後コンピュータ機器はネットワークを介して情報をやり取り可能となり，それに伴って扱う情報量も増えてくることが予想されるため，今まで以上に効率的なコンピュータの利用方法が求められる．

一般にユーザがコンピュータを利用する場合の手順は図 1.1 のようになる．

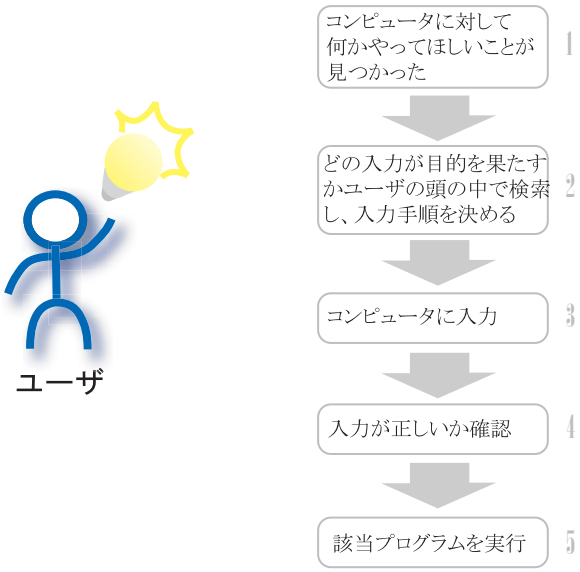

図 1.1: ユーザがコンピュータを利用するときの手順

1, 5 の手順はユーザがコンピュータを使用してプログラムを実行するプロセスにおいて行う必要がある．しかし 2, 3, 4 の手順は合理化することによって限りなく簡素化することができる．ユーザは直接コンピュータプログラムを実行する手段を持たないので，2，3，4 のような手順を踏んでプログラムを実行する必要がある．それぞれの手順には一定の時間を必要とする．例えば携帯電話で文字入力をする場合，煩雑だと感じながらこれをコンピュータのキーボード入力と同程度の効率で入力したいと思うことは多々ある．携帯電話に内蔵されているコンピュータの処理性能ではなくユーザインタフェースが未成熟であるため，煩雑な文字入力作業を行う必要がある．よって図 1.1 の 2, 3, 4 の手順に該当する部分に対するユーザの割く時間を減らすことができればコンピュータの作業効率は大きく向上する．

現在，図 1.1 の 2，3，4 のそれぞれの手順においてコンピュータシステムのユーザ利用環境を改善するための方法が考案されている．図 1.1 の手順 2 の改善方法の例としてユーザの入力履歴の利用が挙げられる．例えば Microsoft Internet Explorer ではユーザの過去の入力履歴を保存し，履歴情報を利用する事によって，URL 入力の補完，フォームの入

力における候補の表示を実現している．このことによりユーザは入力途中で過去の入力履歴を利用することができるのであらかじめ入力手順を把握してから入力を開始しなくてもよい．図 1.1 の手順 3 における改善方法の例として第 1 に取り消し，やり直し，繰り返し機能，第 2 にショートカット技術が挙げられる．ショートカットを用意することによってユーザはより少ない入力数で命令を実行できる．もう一つの例として AppleScript[1] ではユーザの行動を記録し，あとから同じ手順を自動実行できる．またその記録結果を修正できるようになっている．AppleScript を使用することによって多くの入力手順を一つにまとめることができる．図 1.1 の手順 4 の部分の改善の例としては以下のものが挙げられる．Repeat and Predict[7] は繰り返しキーと予測キーの 2 種類のキーを使うことによってユーザによる繰り返し入力の手間を省いている．このシステムではシステムが行う先読みを取捨選択できるので，ユーザは取捨選択の時間がかかる代わり，予測誤りの場合の修正を行わなくてよい．

1.2 本研究の目的

本研究は既存のユーザ入力支援技術の問題点を分析し、それらの問題点を解決する入力支援システムを開発し，評価を行う事を目的とする．そのためグラフィカルユーザインタフェースにおいて効果的な履歴検索を用いた入力支援システムの開発を行う．

本節では前節で述べた既存のユーザ入力支援技術を取り上げ，それらの問題点を挙げる．そして問題点を解決するために必要な要件を述べる．

1.2.1 既存の入力支援技術

現在様々な入力支援技術が普及している．本節では本研究に関連する入力支援技術として，取り消し機能，やり直し機能，ショートカット，履歴検索，スクリプト処理を取り上げる．

従来の取り消し機能や繰り返し機能には問題点が存在する．[6] で述べられているように，システムに対しての入力が複雑となると，状況によって取り消し，やり直し操作に複数の解釈が可能となる．例えば Microsoft Windows においてアプリケーションで作成したファイルを保存したい場合，一般にウィンドウの上端のメニューバーから「ファイル」を選択し,「保存」を選択する．ファイル名を入力して決定すると操作が終了する．この動作を取り消したいと考えたとき，システムに備わっている取り消し命令を利用することになる．しかし，取り消す操作はアプリケーションによって異なる.「ファイルの保存」という行為自体を取り消すのか，それともファイル名の文字入力を一文字取り消すのか，状況によって必要な取り消し操作は異なる．ショートカット技術の問題点として，システムによって同じ機能でも違うキーに割り当てられている場合があるということ，またショートカットキーの機能を覚える負担をユーザに強いるということが挙げられる．履歴検索技術の問題点として，一般に履歴を保存する形式が統一されていないため汎用性がない事が挙げられる．AppleScript[1] では GUI でのアプリケーションにまたがったスクリプトの

構成が可能である．しかし使用範囲は AppleScript に対応したアプリケーションに限定される．そしてユーザの目的に合致した AppleScript の実行ファイルを探しだすのに時間を必要とする．bash[4] では複数のアプリケーションの処理をまとめる事ができる．しかし個々のプログラム単位を組み合わせることはできるがグラフィカルユーザインタフェースのアプリケーションなど，一つのプログラム内で多数のユーザ入力を処理するプログラムでは実用的でない．このように個々のユーザ支援技術を利用する場合には，それぞれに不十分な点が存在する．

1.2.2 先行システムの動向

既存の入力支援技術の問題点を解決する手法として様々な技術が提案されている．

PBD(Programming by Demonstration) や PBE(Programming by Example) は，例を示すことによってシステムが人間の入力パターンを学習し，ユーザ入力の先読みをする入力支援システムである．代表的な研究は Watch What I Do[2] や Your Wish Is My Command[5] に掲載されている．この手法にはいくつかの問題点がある．第一の問題は学習までの時間の長さである．この方法はシステムを一時的に利用する場合には適用できない．第二の問題はシステムのユーザ入力予測がユーザに通知されないことである．ユーザにとって，システムの予測結果を確認する手間が生じ，効率の良さを追求できない．

一方で，システムによる予測失敗によるユーザの修正リスクを下げる手法としては Repeat and Predict[7] における「predict」キーが挙げられる．この方法では「predict」キーによって文字列の繰り返し候補を示し，ユーザが同意した場合にのみ実行する．確認を行うことによりシステムの予測失敗によるユーザの修正をなくすことができる．Repeat and Predict[7] は適用範囲としてエディタ上の文字入力のような，自由に動作の取り消し，やり直しを行うことができる環境に制限される．

1.2.3 問題解決のアプローチ

上で述べた問題点を踏まえ，本研究の目的を述べる．本研究の目的はグラフィカルユーザインタフェース上で利用可能な入力支援システムの開発である．本システムは効果的な履歴検索を実装することにより入力支援を行う．そのためにユーザの入力をシステムに履歴として保存しておき，ユーザがこれから行いたい入力の一部を検索キーとして履歴の検索を可能にする．本研究で提案されるシステムを利用することによってユーザは過去に行った入力の一部を入力するだけで過去の入力を再利用できる．このことによりユーザは効率的な入力を行える．

Windows のようなグラフィカルユーザインタフェースを備えるコンピュータ環境ではユーザ入力履歴を逐一保存することは容易でない．また複数のアプリケーションが平行して動作しており，履歴の取り扱い方も複雑になる．また過去のユーザ入力を検索する手法も bash[4] などのコマンドラインインタフェースと異なり，定番の方法が確立されていない．本研究ではグラフィカルユーザインタフェースにおいても効果的なユーザ入力の履歴

検索方法を提案する．

1.3 本論文の構成

本章では本研究の背景と現存する問題点の概要，そしてその問題をふまえた上での本研究での目的を述べた．第 2 章では，既存の入力支援技術とそれらの適応範囲を述べた後，本研究の成果を利用するためのユーザ環境について述べる．第 3 章では本研究に関連した先行研究の定性的評価を述べる．そして先行研究の比較を行う．第 4 章では本論文で実装するシステムの設計方針，第 5 章ではシステムの実装環境と実装方法を述べる．第 6 章ではこのシステムの評価方針と評価方法，そして評価結果を述べる．第 7 章では本論文の結論，そして今後の課題や改良点を述べる．

第2章　ユーザ入力支援技術

本章では一般に普及している入力支援技術を取り上げる．まず入力支援の概念について述べる．次に入力支援技術について述べるにあたって使用する用語の定義を行う．そして個々の入力支援技術を挙げ，その利点と欠点を述べる．

2.1 入力支援機構

入力支援機構とはユーザがシステムを利用する状況において，通常のシステムの環境よりも容易にユーザの意図を実現する機構である．電話機で電話をかける場合を考えると，ユーザは通話相手の電話番号を調べ，次に電話番号を入力する必要がある．しかし現在では電話機に記録されている相手先の電話番号を検索し，通話ボタンを押すことによって通話が実現する．ユーザは入力支援機構を用いることでより少ない労力で同様の目的を実現できる．

2.2 用語の定義

本節では本論文で使用する用語を定義する．入力支援機構の特徴を述べるにあたり，特に重要な用語としてユーザ入力，命令，命令の効率，入力履歴，領域の５点を取り上げ，それぞれの用語について詳細を述べる。

2.2.1 ユーザ入力

ユーザがシステムに対して情報を伝達する行為を指す．例えば，マウスやキーボードによる入力はユーザがシステムに対して情報を与えている．入力は状況によって意味が変化する．マウスのクリックはアイコンを選択している場合もあり，ウェブブラウザ上のリンクを選択している場合もある．

図 2.1: ユーザ入力の例

ユーザ入力は，取り消し操作が可能なものと取り消し操作が不可能なものの二通りに分類される．また，実行するまで入力の取り消し操作が可能であっても，実行後は不可能になる．例えばアイコンを選択する場合，決定しない限り取り消し操作が可能であり，シェルに文字を入力する場合はリターンキーを押さない限りシステムは命令の実行を行わ

ない．

2.2.2 命令

命令とは，ユーザの意思を反映したコンピュータプログラムを実行する行為を指す．図 2.2 に命令の例を示す．この図の例ではボタンクリック入力を行うことによりファイルの保存命令を起動している．

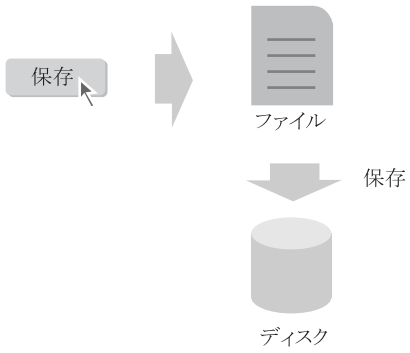

図 2.2: 命令の例

ユーザ入力と命令の違い

ユーザ入力は以下の点において命令と異なる．

- **取り消し操作の可否**
  入力は取り消し操作が可能な場合があるのに対して，命令は取り消し操作が不可能である．
- **手段と目的**
  入力はユーザの目的を達成するための手段であるのに対して，命令は目的である．

図 2.3 にユーザ入力と命令の関係を示す．例としてシェルのプロンプトにおいて，ファイルのバックアップを作成する場合を示す．ここでの命令は「thesis.tex のコピーを作成せよ」である．この命令を実行するために CP コマンドを使用する．

```
# cp thesis.tex thesis.20011115
```

ユーザが該当の命令に相当する文字列を入力し終わったところで，今日の日付は 11 月 16 日だとユーザが思い出した場合，ユーザが「Back Space」キーを押して，最後の 5 を 6

に直すことができる．この場合，修正後の「Enter」キーを押したときに画面に表示されている命令のみが実行される．この場合において個々のキー入力はファイルのコピーを作成したいというユーザの意思を実行する手順とみなすことができる．よってこの場合のキー入力はユーザ入力である．それに対して「Enter」キーを押した場合,「Enter」キー自体はユーザ入力である．しかし「Enter」キーを押した結果実行されたプログラムはユーザの意思を反映している．よって実行された結果は命令である．ここで，命令は「thesis.tex のコピーを作成せよ」で一意であるが，命令を実行するまでのユーザ入力は代替可能である．例えば，cp コマンドを入力するのではなく，グラフィカルユーザインタフェース上の「ファイルを保存」ボタンを押す入力による命令の実行方法も可能である．

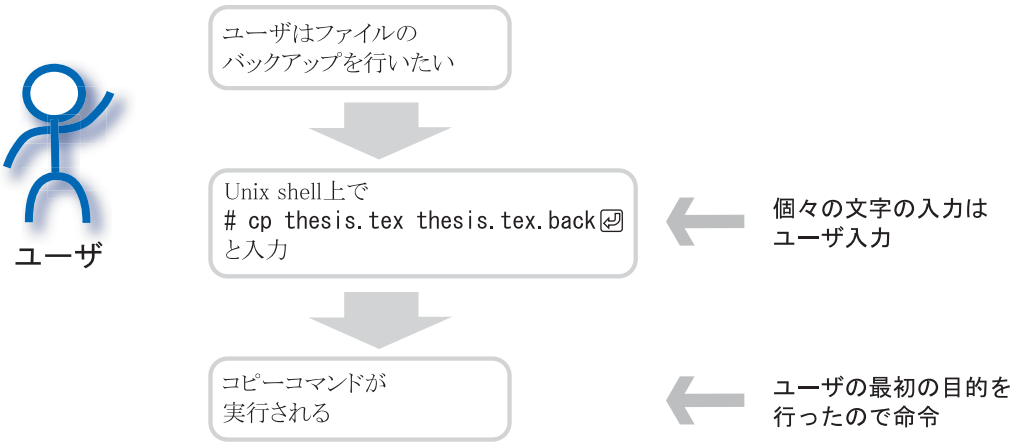

図 2.3: ユーザ入力と命令

また命令の一連の流れをひとつの命令としてみなしたものを，命令群と定義する．図 2.4 で命令群の例を示す．図 2.4 では BMP 形式の画像ファイルを開く命令，画像ファイルの形式を EPS に変換する命令，別名で保存する命令の 3 つの命令をあわせて一つの命令群とし，BMP から EPS への変換命令群として扱うことが可能ということを示している．

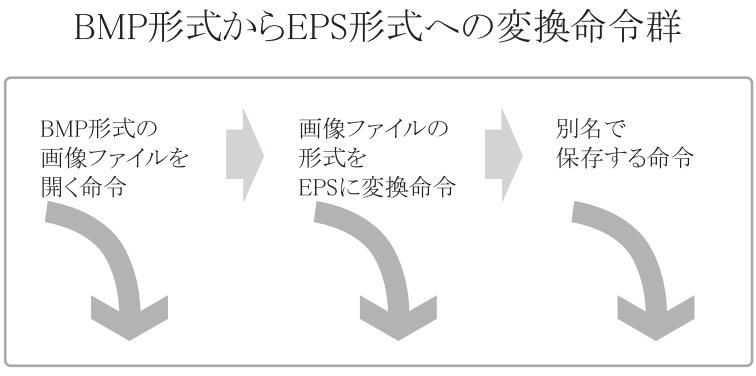

図 2.4: 命令群の例

2.2.3 命令の効率

命令の効率とは，ユーザがシステムに命令する際に費やす時間である．ここでの時間はユーザがある命令を実行したいと考えてから，命令の実行が終了するまでの時間差である．命令の効率は以下の二点に影響を受ける．

- **それぞれのユーザ入力にかかる時間**
  命令を行うまでのそれぞれのユーザ入力にかかる時間.
- **命令の実行までにかかるユーザ入力数**
  ユーザがある命令をコンピュータで実行したいと考えてから，その命令実行にかかるまでのユーザ入力の数.

上記の 2 つの要素をそれぞれ軽減する事で命令の効率は改善できる．図各要素を改善することによって，命令の効率を高める例を 2.5 に示す.

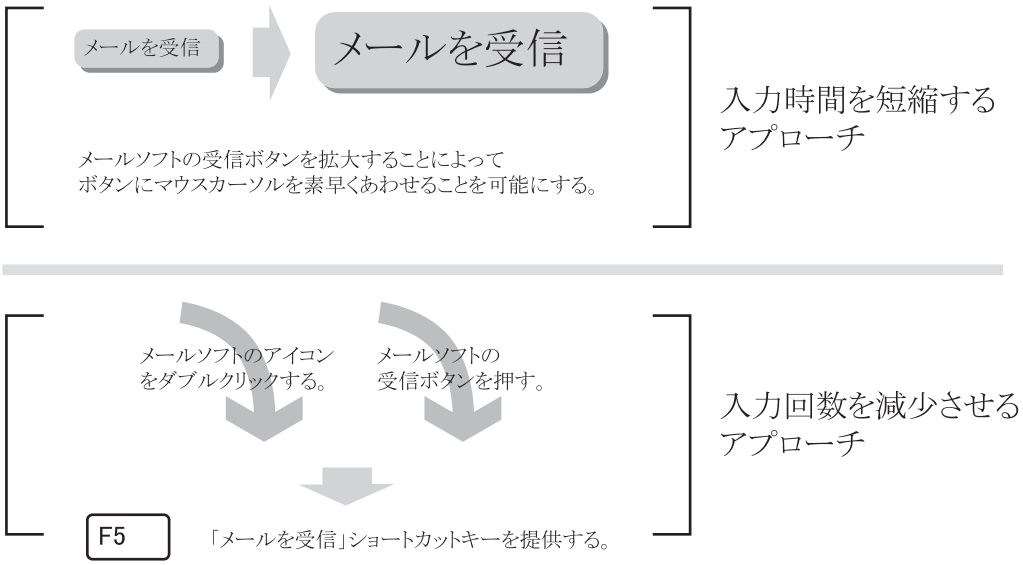

図 2.5: 効率化のための二つのアプローチ

Windows のコンピュータ環境で新着メールを確認する作業を想定する．通常はメールソフトの新着メール受信ボタンのところにマウスを移動した後，マウスをクリックする事でメールを確認できる．この作業の効率を上げるためのふたつのアプローチの例を挙げると，まず，メールソフトの受信ボタンを拡大するとボタンにカーソルを素早く合わせる事ができる．これはそれぞれのユーザ入力にかかる時間を減らすアプローチである．次に，メールを受信するショートカットがあれば 1 入力でメール確認を行うことができる．これは命令の実行までにかかるユーザ入力数を減らすアプローチである．

もう一つの例として，シェルのコマンド入力におけるユーザの文字入力補完を挙げる．シェルには途中までのコマンド名の入力に対して「Tab」キーの入力でコマンド名の候補

を挙げる機能が存在する．この機能によって全てのコマンド文字列を入力するよりも文字入力数が少なくなり，ユーザがキーボードを押す回数，命令実行までの時間ともに減少する．

2.2.4 入力履歴

入力履歴とはユーザが過去に行ったユーザ入力とそれに伴い起動した命令を，システムが保持したものである．入力履歴を保持することはユーザ入力と命令の対応の検索を可能にする．例えば Tera Term を起動した場合に通常ログインしているホストの名前を判別できる文字を入力し検索を実行すると，自動的にログイン名とパスワードを補完し，ログインする命令を候補に挙げるといったシナリオが挙げられる．

2.2.5 領域

領域とはシステム内においてユーザ入力によって直接影響を受けるオブジェクトを指す．領域は入れ子構造にすることができる．例えばウィンドウという領域の内部にボタンという領域を含むことができる．領域の情報を入力履歴に付加することによって，単純に入力履歴を保存しただけでは命令の再現に足りない情報を補完することができる．例えばマウスクリックを履歴として保存する場合を想定すると，マウスクリックが影響を与えるウィンドウを記録せずに入力の結果を再現しようと試みても，どのウィンドウ上でマウスがクリックされたか判別できないので再現ができない．図 2.6 に領域の例を示す．この図ではそれぞれのウィンドウとボタンは一つの領域である．またボタン 1，2 はウィンドウ 1 の内部にあるので，ウィンドウ 1 の下にボタン 1，2 をおいて領域を階層化している．

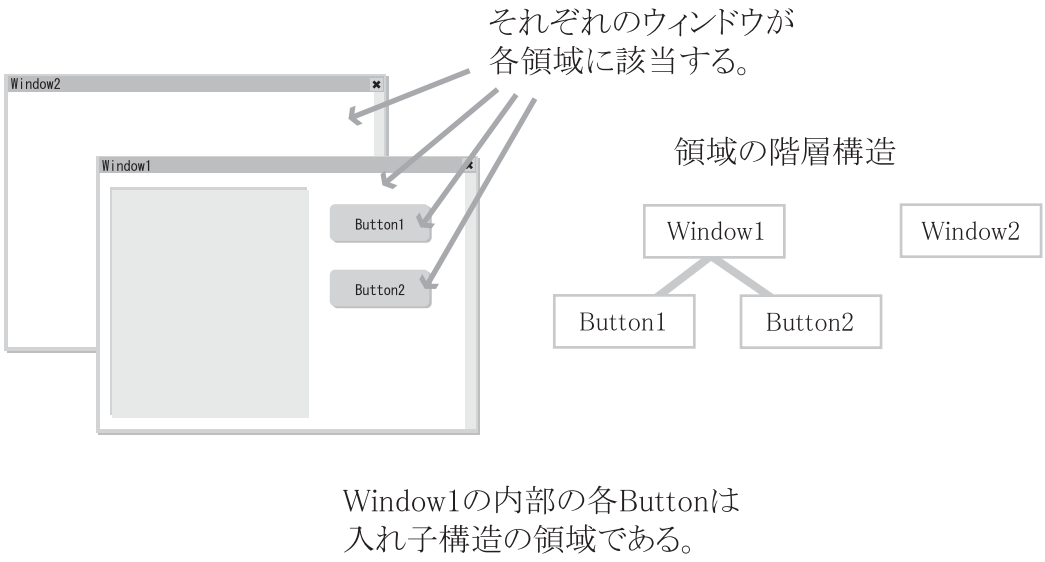

図 2.6: 領域の例

2.3　本研究に関連する入力支援技術

本節では本研究に関連する既存の入力支援技術として以下の五点を取り上げ，その分析と評価を行う．

- **機能の特化**
  機能の特化は比較的よく使用されると開発者が推測する機能に限定して機能を提供する手法である．これにより機器の利用が容易となる．

- **ショート カット**
  ショートカットは特定の機能にキーを割り当ててユーザ入力を減らす手法である．

- **ユーザ入力の予測**
  ユーザ入力の予測はシステムがユーザの入力を予測してユーザ入力の回数を減らす手法である．

- **ユーザ入力履歴検索**
  履歴検索はユーザ入力の履歴を逐一記録しておき，必要に応じてユーザ入力を再利用する手法である．

- **学習機能**
  学習機能はシステムがユーザの過去の入力履歴からユーザの入力の偏りを評価し，ユーザの入力を予測する手法である．

2.3.1　機能の特化

機能の特化とは，提供する機能をよく使われるものだけに限定することでユーザ環境の改善をはかる手法を指す．

機能の特化の利点を挙げる．

- **単純な操作の提供**
  ユーザがユーザ環境改善のためのカスタマイズを行う手間が減り，はじめから適度な使い勝手を提供できる．

例えば，一般的な電子レンジを挙げると冷凍食品，ご飯など，オーブン電子レンジを利用する代表的な製品用にカスタマイズされた機能がボタン一つに対応しており，ボタン一つでそれぞれの製品に対応した調理が実現する．

しかしこの技術の問題点として次の二点がある．

- **機能が画一的**
  用意された機能がユーザの使用するものでない場合がある．また使用頻度の高い機能がユーザによって様々である．

- **カスタマイズ不可**
  用意されていない細かな機能をユーザは使用できない．

問題点の例を挙げるとケーキを焼くという機能が電子レンジの標準の機能にない場合はケーキを焼くときに同じ設定を毎回繰り返す必要がある．このような問題点により，用意された機能とは別の機能を利用したい場合には，ユーザは機能を持つ機器を別に用意する，もしくは一つ一つの機能を手作業で行う負担がかかる．

2.3.2 ショートカット

ショートカット技術とは複数の入力をより少ない入力で置き換えるものである．一般にグラフィカルユーザインタフェースにおいてマウスを使わずにユーザ入力を行う機能を指す事が多い．マウスを使用するよりもキーボードを使用したユーザ入力の効率が良いという性質を利用したユーザ入力支援の手法である．

この技術の利点を挙げる．

- **ユーザ入力数の減少**
  ユーザの入力数を減らすことができる．

例えばWindowsで文字や画像をコピーやペーストする場合，コピーにCtrl-C，ペーストにCtrl-Vのショートカットキーが割り当てられている．マウスをウィンドウ上部のメニューバーに移動し，それらのコマンドを選択して実行する動作に比べて，ショートカットを利用するほうが入力の回数と時間が省略できる．

ショートカットの手法は2.3.1で述べた機能の特化との類似点がある．しかし機能の特化の場合は代表的な機能を中心にユーザインタフェースを実現するのに対してショートカット機能はあくまで補助的な機能である．ショートカット技術を装備する場合，ユーザがショートカットの使用法についての知識を持たない場合でもアプリケーションの利用に支障はない．

しかし既存のショートカット技術には次の四点の問題が挙げられる．

- **習得機会の少なさ**
  補助的な機能であるためユーザがショートカット機能を習得する機会が少ない．

- **学習が必要**
  ユーザがショートカットと機能の対応を新たに覚える労力が必要である．

- **非互換性**
  システムによって，特定の機能のショートカット自体が存在しなかったり違う対応付けがなされている．

- **カスタマイズ不可**
  システムによっては，ユーザがショートカットを新規に作成することが不可能であったり，コストがかかったりする．

これらの問題点により、ユーザがショートカット機能を利用する場合、アプリケーションごとにショートカットを記憶する負担を負う.

2.3.3 ユーザ入力の予測

ユーザ入力予測とは，システムがユーザの入力を予測してユーザ入力の回数を減らす手法である．ここではあらかじめ用意された予測を示している．

ユーザ入力予測の手法の利点を挙げる.

- **ユーザ入力数の減少**
  入力予測がうまくいっている限りユーザ入力数は減少する.

しかしユーザ入力予測の手法には以下のような問題点がある.

- **システムの予測ミス**
  既存の入力予測機構は精度が低い．ユーザはシステムの予測結果を確認する手間が発生し，結果的にユーザ入力の効率が悪くなる場合がある.

例として図 2.7 に Microsoft Word におけるリスト補完を行う入力支援機能を示す．左図の状態で改行を行うと，右図のように自動的に「2.」が補完される．この機能はユーザが

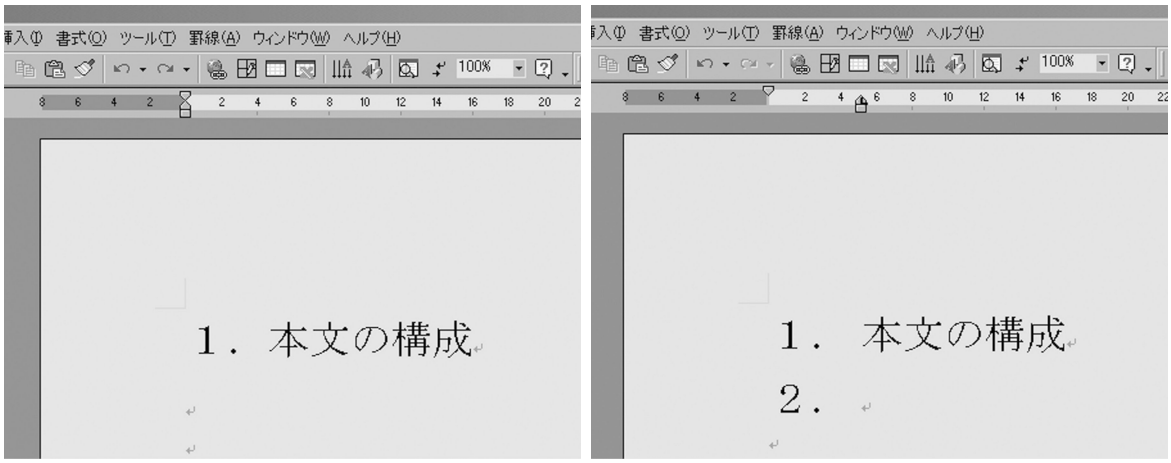

図 2.7: Microsoft Word での入力補完機能

箇条書きを想定している場合は予測が的中している．しかしユーザが章立てを意図して数字を振っている場合は次に通常の文章を書くことを想定しているので訂正するための余計な手間がかかる.

この問題点によりユーザは毎回予測ミスを訂正する負担を負う。

2.3.4 履歴検索

履歴検索はユーザ入力の履歴を逐一記録しておき，必要に応じてユーザ入力を再利用することによって過去と同じ入力を繰り返し行う必要をなくす手法である．

履歴検索の利点を二点挙げる．

- **安全性**
  ユーザが過去に実行した命令であるのでユーザの意図しない動作を行う危険が少ない．

- **再利用可能**
  ユーザが過去に入力したことを再度入力する手間を省くことが可能である．

図 2.8 に，Microsoft Internet Explorer5 は URL 入力や Form 入力で過去の履歴情報から入力を補完する．この図の例では www.ama まで入力した時点で，普段利用している www.amazon.co.jp を候補に上げている．

また bash[4] では，Ctrl-R キー入力によって過去に実行した履歴を検索できる．さらに直前に実行したコマンドを Ctrl-P キー入力によって再度入力の手間をかけずに呼び出すことができる．これによりユーザ入力を効率化させている．

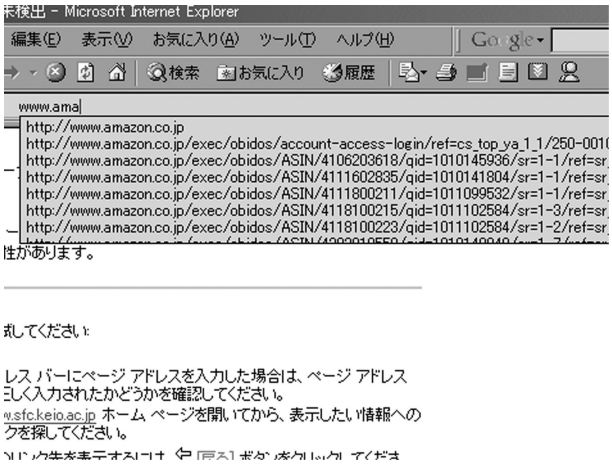

図 2.8: Microsoft Internet Explorer5 における URL 入力の補完機能

しかし，履歴検索の手法には以下の問題点が存在する．

- **履歴操作にかかるオーバヘッド**
  履歴を保存したり検索するための記憶容量が必要となる．また入力を逐一保存するためのコンピュータ処理能力が要求される．

ユーザは履歴操作を利用する入力支援機構の利用においてシステムのパフォーマンス低下を考慮しなければならない．

2.3.5 学習機能

学習機能はシステムがユーザの過去の入力履歴を利用してユーザの入力の予測を行う．学習機能の利点を挙げる．

- **カスタマイズ機能**
  学習機能を利用することによってユーザは入力支援機構をカスタマイズできる．

学習機能の例として日本語文字入力システムが挙げられる．Microsoft IME[8] では初期設定時から一般的なユーザの異口同音文字の利用頻度が学習されている状態である．その後ユーザが機能を利用していくことでユーザの利用頻度を学習する．

学習機能の手法には次に挙げる三つの問題がある．

- **ユーザの負担**
  実用に耐えられる予測の水準にシステムが達するまでに時間とユーザの労力がかかる．
- **入力予測のミス**
  ユーザにとって入力予測が最適な入力効率の状態にならない場合がある．
- **学習過程が不明瞭**
  学習機能によって生成された学習状態をユーザに理解できる形で表現することが困難である．

このような問題点によりユーザは常にシステムの入力予測と意図のずれに配慮しながら入力支援を利用するという負担を負う．しかし，思い通りの入力が可能という理由で文字コード入力で全ての日本語入力を行うよりは，時にはユーザの予測が外れる可能性を考慮しても IME などの日本語入力システムを使用したほうが効率がよい．

2.4 既存技術の評価

表 2.1 に既存技術のそれぞれの利点と問題点をまとめたものを示す．このように既存の技術はそれぞれ一長一短の特徴を持つ．機能の特化，ショートカット，ユーザ入力の予測の 3 つはあらかじめシステム開発者が備える技術である．これらの技術は基本的にカスタマイズができない．入力支援技術において開発者の予測とユーザの要望との差を埋める手段としてユーザ入力履歴検索や学習機能を利用できる．しかし表 2.1 で問題点を挙げたように，これらの手段のみでは十分なカスタマイズ機能を提供できるとは言えない．

2.5 本章のまとめ

本章ではまず本論文において必要となる用語を定義した．そして既存のユーザ入力支援技術の分析を行い評価した．その結果として既存の入力支援技術では十分な入力支援を

表 2.1: 各技術の利点と欠点の比較

技術	利点	欠点
機能の特化	設定なしに適度な使い勝手を提供	機能の均一な提供 カスタマイズ機能がない
ショートカットキー	入力時間と回数の短縮が可能	習得機会の少なさ ユーザの学習の必要性 カスタマイズ機能がない
ユーザ入力の予測	入力数の減少 （ただし予測が的中する場合）	システムの予測ミス
ユーザ入力履歴検索	安全性 再利用可能	履歴操作にかかるオーバヘッド
学習機能	カスタマイズ機能	最適化に手間がかかる 学習過程がユーザに分からない 入力予測がマッチしない

行っていないことを示した．次章ではユーザ入力の記録方法について関連研究を挙げ，分析と評価を行う．

第3章　ユーザ入力履歴記録

前章では様々な入力支援技術を挙げ，それらの問題点を述べた．本章ではユーザ入力支援機構の構築に重要な役割を果たす，ユーザ入力履歴の記録技術について関連研究を挙げ，それぞれ評価する．関連研究として Repeat and Predict[7] と A Temporal Model for Multi-Level Undo and Redo[3] の二つを取り上げる．

3.1 入力支援機構

入力支援機構を構築する場合，様々なアプローチが考えられる．以下に本研究における入力支援機構の要件を述べる．

- **ユーザの過去の入力を再現可能**
  ユーザの過去の入力に基づいた入力支援を行う．
- **ユーザ志向**
  システムはあくまでユーザの意図しない入力支援を行わない．

本章では以上の要件を満たす上で前提となるユーザ入力の履歴保存，検索技術について関連研究を挙げ，評価を行う．

3.2 関連研究

関連研究として Repeat and Predict[7] と A Temporal Model for Multi-Level Undo and Redo[3] の二つを取り上げ，分析と比較を行う．

3.2.1 Repeat and Predict

Repeat and Predict[7] は Toshiyuki Masui と Ken Nakazawa によって提案された．Repeat and Predict では，ユーザの過去の履歴を利用することにより，予測キーと繰り返しキーの二つのキーによる動的マクロ生成実行機能を実現している．

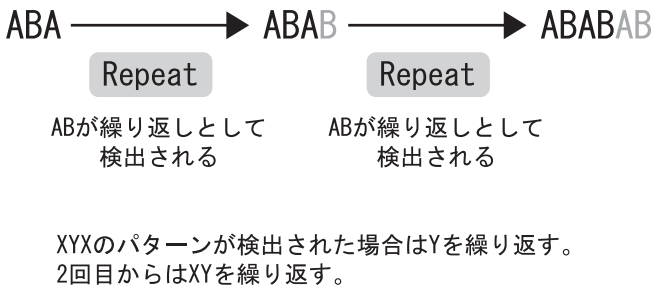

図 3.1: Repeat and Predict における Repeat 機能

図 3.1 に Repeat and Predict における Repeat 機能の例を示す．Repeat キーは XX もしくは XYX のパターンを検出し，XY を繰り返す．しかし「012345678910」のような，繰り返しのパターンに「0123456789100」と
「01234567891001234567891O」の複数の候補があり，ユーザの意図しているパターンが「01234567891001234567891O」のように，もっとも単純なものではない場合には Repeat

キーの機能だけでは十分でない．Repeat and Predict ではPredict キーという概念を導入してこの問題を解決している．

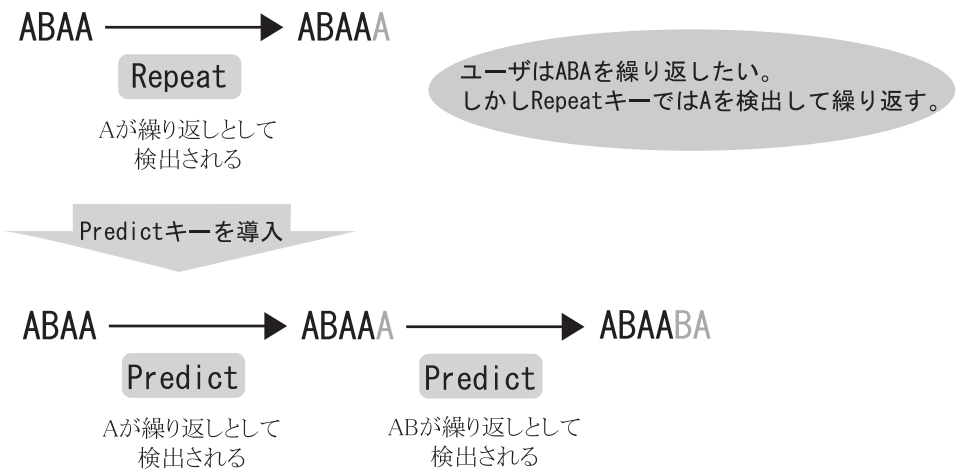

図 3.2: Repeat and Predict における Predict 機能

図3.2 に Repeat and Predict における Predict 機能の例を示す．Predict キーを利用することによって，システムがユーザの意図した候補を選択することが可能となる．

この方式は次の二つの利点が挙げられる．

- **システムが予測の確認をする**
  システムがユーザの意図と違う命令を実行する可能性がない．ユーザがシステムの誤った予測に混乱することが起きないのでシステムの入力支援をユーザは信頼することができる．

- **使用方法が単純**
  単純な二つのキーにより多様なマクロ機能が実現されるので使用方法が単純である．ユーザに支援技術の高度な知識を要求することがない．

この方式の欠点として次の三点が挙げられる．

- **マクロの記録ができない**
  単純に直前の履歴を検索して繰り返すのみであるので，過去に行った入力の繰り返しが利用できない．よって入力支援システムとしてみた場合，機能が十分でない．

- **キー入力に限定した履歴のみを扱う**
  同様にキー入力についてのみ履歴を扱っているのでシステムとしては十分でない．

- **単一の履歴を扱う**
  履歴情報を時系列で保存するので，繰り返し対象が直前の入力に限定される．

Repeat and Predict は単純で強力な機能であるが，上記のような問題点も存在する．

3.2.2 A Temporal Model for Multi-Level Undo and Redo

A Temporal Model for Multi-Level Undo and Redo[3] は W. Keith Edwards らによって提案された．A Temporal Model for Multi-Level Undo and Redo では，ホワイトボードにおいて行われた操作の履歴を線や図などのオブジェクトごとに保存することにより，それぞれのオブジェクトの取り消し機能とやり直し機能を実現している．この方式には，平行して複数の操作を行っている作業でもオブジェクト単位での取り消しおよびやり直し操作が可能である．

図 3.3 のように，アプリケーション間で横断して作業している場合の履歴を扱う方式を提案している．地図を書き込む操作と説明を書き込む操作のどちらの操作も履歴を取っておく事で個別の取り消しやり直しが可能となる．

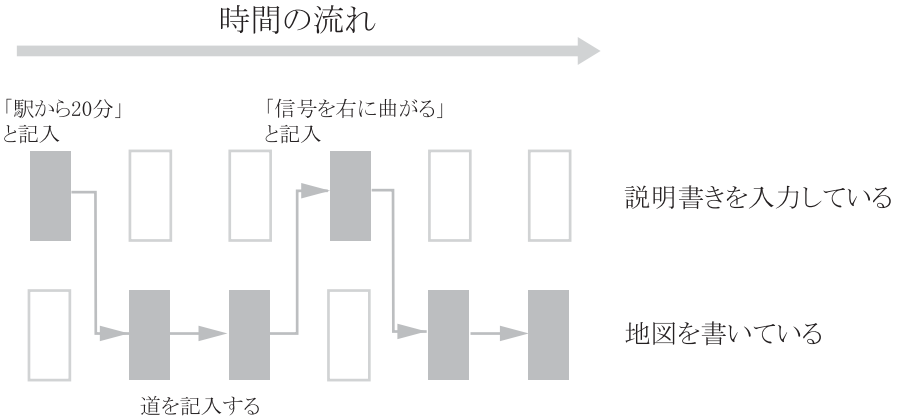

図 3.3: 地図を書く場合

この研究は，入力履歴を保存するときにマウスのクリック位置や使用しているアプリケーションなど入力された状況までを保存することによって，アプリケーションにまたがった履歴の利用が可能となるという利点を持つ．

この研究には以下に挙げる利点がある．

- **複数の階層の履歴を扱う**
  複数の階層の履歴を扱うため，過去に行った入力に対してだけ修正を加えるというような事が可能となる．

3.3 関連研究の比較

Repeat and Predict と A Temporal Model for Multi-Level Undo and Redo で前章に述べた問題が解決された点と解決されていない問題点を表 3.1 に示す．

Repeat and Predict では文字入力に対して単一の入力履歴を記録している．よって文字の入力位置に依存した履歴情報などは扱うことができず，直前の入力履歴のみの活用とな

表 3.1: 関連研究の特色

	Repeat and Predict	A Temporal Model for Multi-Level Undo and Redo
履歴検索	有り	無し
複数の履歴	無し	有り
簡易さ	有り	有り

る．A Temporal Model for Multi-Level Undo and Redo では取り消し操作とやり直し操作のみに入力履歴を利用しており，履歴検索ができない．

3.4 本章のまとめ

本章では入力支援機構に関しての先行研究の中で，本研究に特に関連のあるものを二つ取り上げた．そしてそれぞれの先行研究において問題点を挙げた．関連研究の比較を踏まえ，本研究ではキーの種類やマウスの位置だけでなく，システムの状況を含めて入力履歴を保存する事により，パターンだけでなく状況に依存した履歴検索を可能にする．このことによりユーザに対してより状況に適合した候補の提供を可能にする．次章では本研究の中で上記の問題点を解決するためのシステムの設計を述べる．

第4章　システムの設計

本章ではシステムの実現に必要な設計を述べる．まず本システムで想定するコンピュータ利用環境を述べる．次にシステムの要件を述べ，要件を満たすシステムの構成を述べる．

4.1 機能要件

本節では本システムで実現する機能を述べる．このシステムを使用することで，ユーザは検索要求後即座に過去の入力履歴から必要な入力を再利用するすることができる．また検索結果を一つづつユーザに候補として通知することにより，ユーザは入力の確認を行うことができ，入力ミスを軽減できる．

本システムを設計するにあたり重視する機能を述べる．

- ユーザ入力支援機構の使用経験によらず，即座の利用が可能である．
- ユーザ入力支援機構を使用するために高度な技術，記憶を必要としない．
- ユーザ入力支援機構の動作がユーザにとって予測可能である．
- 過去のユーザ入力を再利用可能である．

以上の点を踏まえて次節ではこの要件を満たすのに必要な設計の前提環境を，その次の節ではシステムの設計方法を述べる．

4.2 本研究で想定するコンピュータ利用環境

入力支援機構を構築する際いくつかの技術課題が存在する．現在，ネットワーク機能やディスプレイ装置を備えたワークステーションから単純な計算機能のみを持つ電卓まで様々なコンピュータ環境が存在する．本研究で前提とするコンピュータ利用環境は，以下の 2 点を満たす環境である．

- **入力の確認機能**
  ユーザ入力の結果が画面などに瞬時に表示されユーザの過去の入力が表示可能である．
- **入力の履歴保存機能**
  ユーザ入力を記録しておき，瞬時に利用が可能である．

4.2.1 入力結果の確認機能

入力結果の確認機能はユーザ入力の内容をユーザ自身による確認を可能にする機能である．システムに対する文字入力をする状況を想定すると，ユーザ入力の確認ができない場合，入力確認可能なシステムと比較してユーザ入力の正確さと効率は悪くなる．

4.2.2 入力履歴の保存機能

入力履歴を保存するためには十分な記憶装置が必要である．様々なコンピュータ環境の中には履歴保存に十分な記憶容量を備えていない環境も存在する．例を挙げると，携帯電話や電卓にはユーザが利用できる記憶容量は非常に少ない．

4.3 システム機能

本節ではまずシステムに要求される機能を述べる．次にユーザのシステム利用状態を述べる．

4.3.1 システム機能要件

本システムを構築する際，満たす必要のある要件として次の四点を挙げる．

- **ユーザ入力を逐一保存**
  入力履歴保存機能はユーザ入力履歴としてユーザ入力の時刻，種類，位置などを保存する．
- **過去の入力の履歴を検索**
  ユーザ入力の履歴の記録から入力パターンを検索する．
- **実行する命令をユーザに確認**
  検索の結果，候補に上がった命令群がユーザの意図と合致していることを確認する．
- **ユーザに確認が取れた命令を実行**
  記録されている情報を元に命令の実行を行う．

それぞれの要件を一つの機能としてモジュールに分割して設計する．

4.3.2 ユースケース

本システムに対してユーザが行う行為は以下の三点である．

- **履歴検索**
  入力支援を利用したい場合，検索命令によって検索モジュールに過去の履歴を検索させる．
- **候補を却下**
  システムの提示した候補がユーザの意図と異なった場合，候補を却下する．システムは自動的に次の候補を表示する．

- **候補の実行確認**
  システムが提示した候補がユーザの意図と合致していた場合，実行命令を行い命令実行モジュールに命令を実行させる．

これらの 3 つの入力はいずれも一回の入力とする．ユーザは入力支援を利用する場合，検索を行う．システムは検索結果を候補として表示する．候補がユーザにとって必要でない場合は検索を再度実行する．候補がユーザの意図する命令であった場合は，実行を行う．例えばユーザが F4 キーを押すとシステムは履歴検索の結果をウィンドウに表示する．そこでユーザが F5 キーを押すとシステムは候補を却下する．ユーザが F6 キーを押すとシステムは候補の命令を実行する．

4.4 設計方針

本節では本システムの設計方針を述べ，各部分の機能を述べる．

4.4.1 全体構成

本システムの構成を図 4.1 に示す．本システムは 4 つの機能から構成される．1 つ目はユーザ入力を保存する機能，二つ目は履歴から命令を検出する機能，3 つ目は命令の実行をユーザに確認する機能，4 つ目は命令の実行機能である．

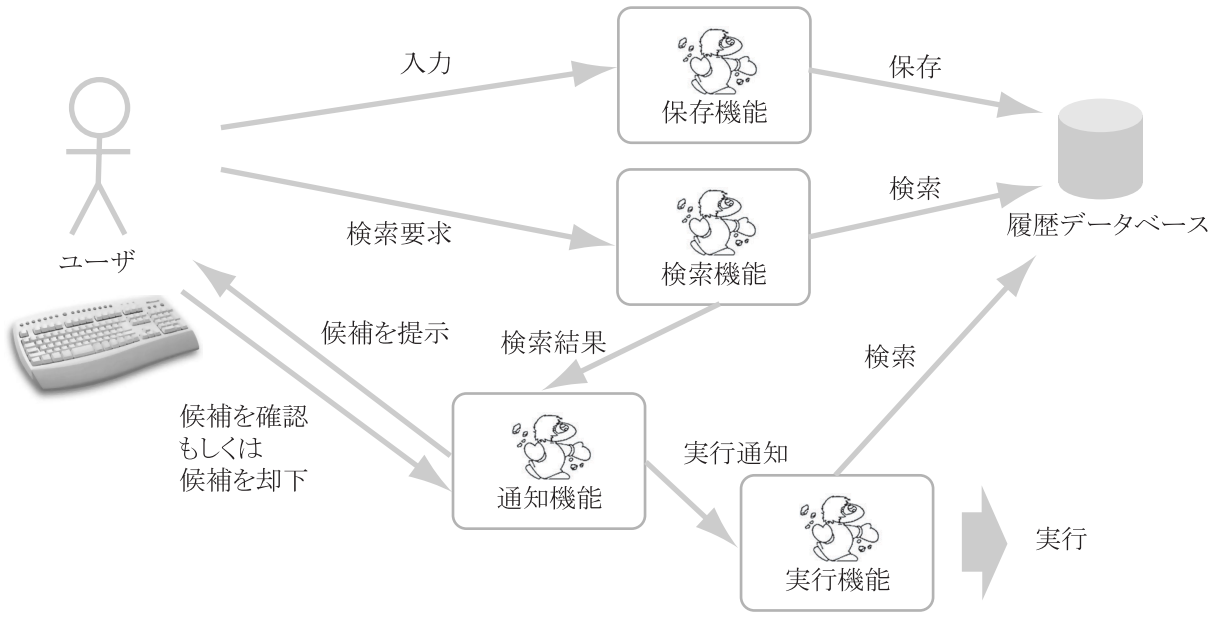

図 4.1: システムの全体構成

また図 4.2 に本システムの流れを示す．図 4.2 には 2 つの状況が混在している．1 つ目

はシステムがユーザの入力内容を保存する状況である．ユーザは入力を行い，保存機能は入力をデータベースに保存する．2 つ目はユーザがシステムに対して履歴の検索要求を行う状況である．ユーザは検索機能に検索要求を行い，検索機能はデータベースから入力パターンを検索する．検索機能はデータベースから取得した検索結果を通知機能に受け渡す．通知機能は候補としてユーザに提示する．ユーザは候補を実行してよいと判断した場合，システムに命令の実行許可を与える．この場合，通知機能は受け取った情報を実行機能に受け渡す．情報を受け取った通知機能は命令の実行を開始する．ユーザが候補を却下する場合，通知機能はもう一度検索機能に検索を要求する．

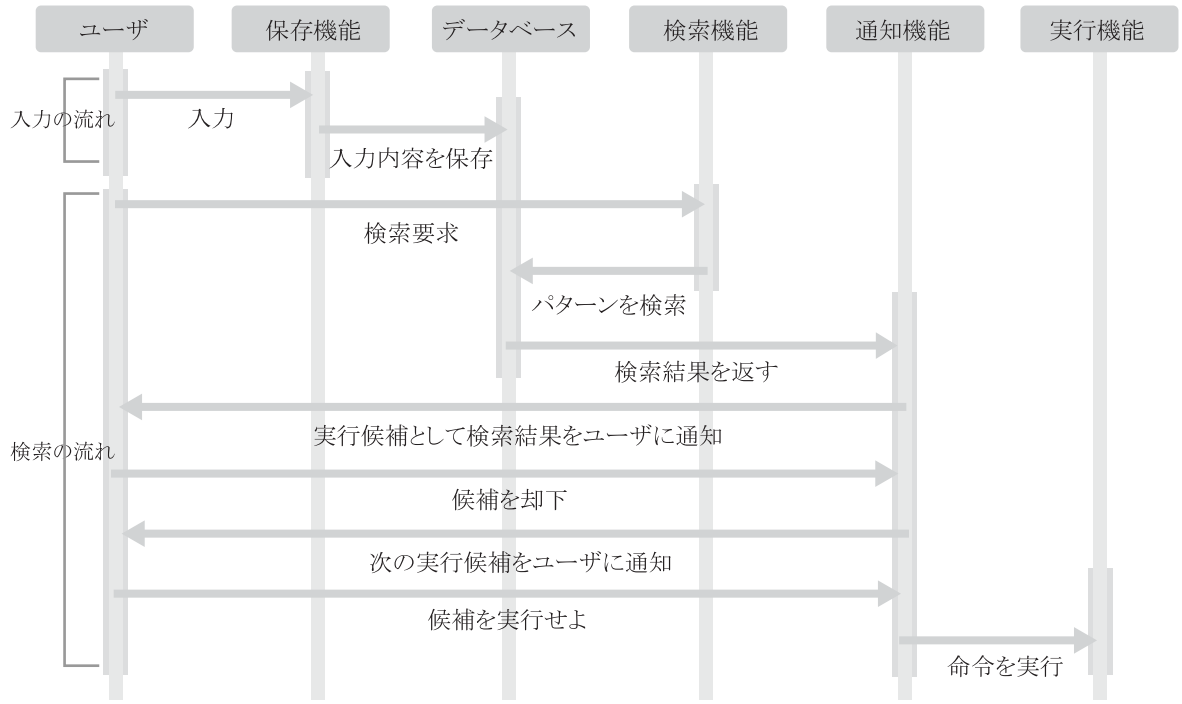

図 4.2: シーケンス図

4.4.2 データ構造

本小節では本システムを構築する場合に要求されるデータ構造を述べる．データ構造で必要な要件を以下に挙げ，各データ構造体で必要な要素を図 4.3 に示す．

- **入力装置の情報**
  このフィールドはシステムに入力を行うことが可能な装置の情報を保存する．例えばコンピュータに入力装置としてマウスとキーボードが付属している場合，マウスやキーボードの一つ一つのボタンを識別するための情報を格納する．

- **入力**
  このフィールドは個々の入力を保存する．例えば，マウスクリックが行われた場合，いつ，どのウィンドウ内でマウスクリックが行われた，という情報を格納する．

- **領域**
  このフィールドはコンピュータ環境内の領域を保存する．例えば個々のアプリケーションを一つの領域とみなす場合，現在使用しているアプリケーションを識別する情報を格納する．

- **命令**
  このフィールドは入力情報と入力の結果行われた命令の情報を対にして保存する．この入力が行われた結果，実行された命令内容を表す．

図 4.3 にデータ構造の概要を示す．

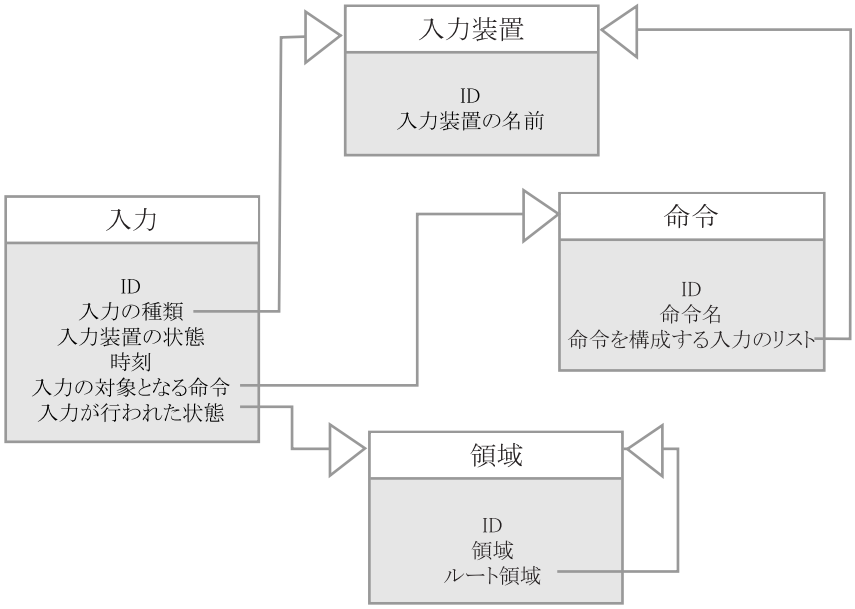

図 4.3: 基本データ構造とその関連

4.4.3 履歴保存機能

ユーザがシステムに対して入力を行った場合，履歴保存機能はユーザ入力を自動的に保存する．システムは保存するユーザ入力を利用してユーザ入力を再現できなければならない．保存するべき入力情報を次に挙げる．

- **入力の種類**
  入力の種類はキー入力やマウス入力など，入力に使う装置を識別するための情報を保持する．

- **入力装置の状態**
  マウスのカーソル位置など，入力装置に付随する情報を保持する．
- **時刻**
  入力の時刻を必要な精度で保存する．特に問題がない場合はミリ秒単位で保持する．
- **入力の対象となる命令**
  入力の結果として実行した命令の情報を保持する．
- **入力が行われた状態**
  上記の他に必要な付加情報を保存する．例えば入力の対象ウィンドウの情報を保存することによって後ほど入力ウィンドウの特定が可能となる．

4.4.4 検索機能

検索機能は次の手順で履歴から命令群を検索する．まず履歴を直前のユーザ入力をキーとして検索し，次に検索結果を候補に挙げる．本研究では複数の領域にまたがる検索を可能にした．しかしまずは単一の領域内の検索の手順を説明する．また，図 4.4 に単一の領域内の検索の手順を示す．

1. 入力を保存する．
2. 直前の入力が行われた領域を取得する．
3. 同じ領域内で同じ入力が行われた履歴情報を検索する．
4. 検索結果の中から入力の直前の入力と検索結果の直前の入力が等しいものを候補としてユーザに提示する．

上記の方法は単一領域内の検索では効果的である．しかし複数の領域にまたがる候補の検索ができない．そこで，ユーザが図 4.4 の方式で検索を行った後に，複数の領域にまたがる候補を提示するための方式を説明する．また図 4.5 に複数の領域にまたがる命令の検索方法を示す．

1. はじめは単一領域内のみを対象にして命令を実行する．
2. ユーザがさらに予測を要求した場合には，直前の検索結果に続く命令群の候補をユーザに表示する．
3. 候補がユーザの予想と合致しない場合は入力に対する検索に戻った後，次の候補を検索して続く命令を候補とする．

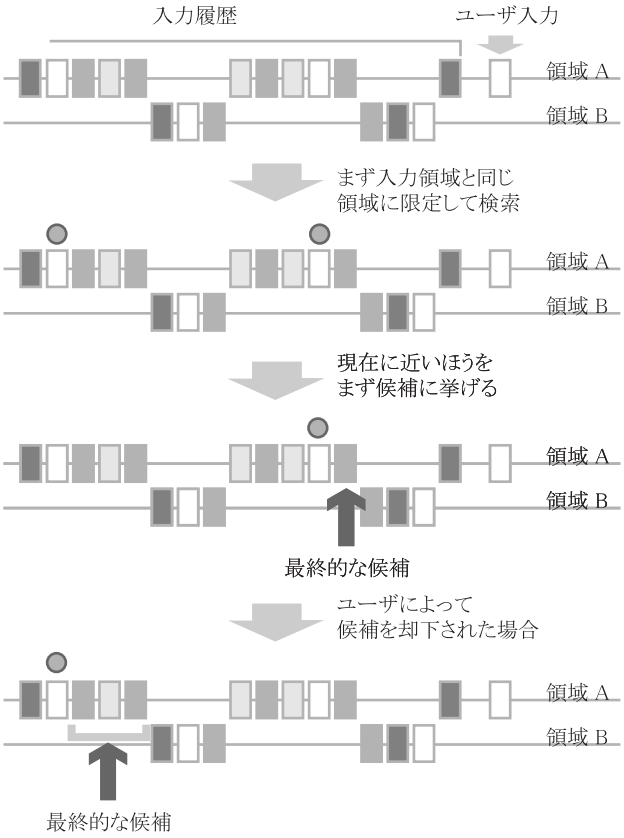

図 4.4: 単一領域の検索手法

4.4.5 通知機能

通知機能は，検索機能が検索を行った結果得られた候補をユーザに通知する．その後通知機能はユーザの反応を待つ．ユーザは通知された候補の実行の可否を判断し，実行すると判断した場合には通知機能は命令内容を実行機能へ伝達し，命令を実行する．ユーザへの通知に際しては以下の情報を付加する．

- 入力の種類
- 入力の結果実行された命令

入力の結果実行された命令をユーザの目に見える形で表現し，的確に命令内容を伝える確立された方法はない．本研究では命令に関連する図形が存在すれば表示し，存在しなければ命令内容を説明できるような文字列をユーザに表示する．

ユーザの反応

通知機能が提示した候補について，ユーザは候補を実行する，もしくは却下するのどちらかを行う．ユーザが候補を却下する場合，次の候補を検索機能に要求するか，もしくは

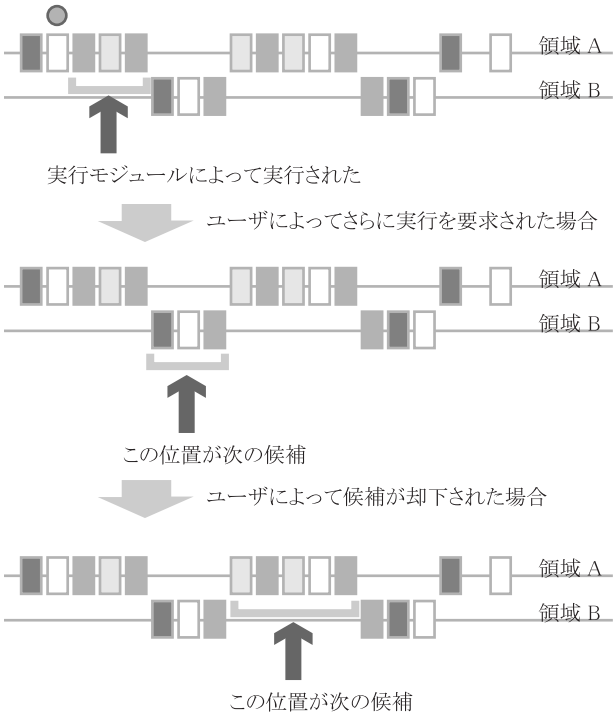

図 4.5: 複数の領域にまたがる検索手法

検索を終了する判断をする．ユーザは通知機能に対して次の二つの入力を利用することによってこの操作を行う．

- 検索入力
- 実行入力

ユーザがシステムの予測を要求する場合には，ユーザはもう一度検索入力を行う．そして通知機能から通知された候補命令の実行を判断する．ユーザは何度でもシステムに対して候補の要求を行うことができる．それに対してユーザが候補を実行すると判断した場合には実行入力を行う．システムは命令を実行する．候補を却下する場合は 2 つの場合に分けることができる．まず，第 1 にユーザが次の候補を必要としない場合には，検索入力か実行入力以外の入力を行うことによって自動的に検索は終了する．第 2 に，ユーザが次の候補を要求する場合には，ユーザは検索入力を行うことによって次の候補を検索機能に要求する．その結果，システムはもう一度候補をユーザに通知する．

4.4.6 実行機能

通知機能を介してユーザは実行の確認を行った後，通知機能は実行機能に命令内容を伝達する．命令機能は受け取った命令内容を実行する．命令内容は特定するための変数，もしくは命令を実行するための入力の集合として伝達する．

4.5　本章のまとめ

本章ではまず本研究で前提とするコンピュータ利用環境について述べた．次に本システムの要件を述べ，設計の概要を示した．次章では本設計に基づいて実際にシステムを構築し，その実装方法を述べる．

第5章　システムの実装

本章では本システムの実装に必要な事柄を述べる．はじめに本システムの実装に使用した環境を述べる．次に本システムを構築するにあたって必要とする技術的知識について述べる．次にシステムの処理の流れを解説し，おわりにデータ構造，履歴保存機能，検索機能，実行機能，通知機能の 5 点について実装方法を解説する．

5.1 実装環境

表 5.1 の実装環境で実装を行った.

表 5.1: 実装環境

	version
OS	Windows2000 Service Pack2
開発環境	Visual C++6.0
Database	MySQL3.27.47 (WindowsNT 版)
DB Interface	ODBC
ODBC Driver	myodbc2.50.39(NT 版)

本システムはWindows2000 Service Pack2 上で実装を行った. 開発環境はVisual C++6.0 を使用した. Visual C++を使用した理由として, Windows API の呼び出しが可能であることを挙げる. またユーザ入力の履歴を保存するためにデータベースとして MySQL3.27.47 (WindowsNT 版) を使用した. MySQL データベースとの入出力インタフェースにはODBC を使用した. MySQL の ODBC ドライバにはmyodbc2.50.39(NT 版) を使用した.

5.2 基本技術

本節では実装にあたって必要な知識を述べる. まず必要な用語を説明する. 次にWindows 独自の API の説明を行う.

5.2.1 実装に関連した用語

本小節では実装方法を述べるにあたって使用するWindows 独自の概念を, プロセスとスレッド, メッセージ, フック処理, ハンドル, コールバック関数, DLL の 6 点について説明する.

プロセスとスレッド

一般に Microsoft Windows のアプリケーションは単一プロセスで稼動している. アプリケーションは複数のスレッドが平行して動作している.

基本的にアプリケーション内でデータの共有を行うことは可能である. しかしアプリケーション間でデータの共有は, あるアプリケーションがメモリ内のある部分に書き込んだ情報を別のアプリケーションが読み取るといった単純な方法では不可能である. アプリケーション間の通信を行うためにメッセージを使用する.

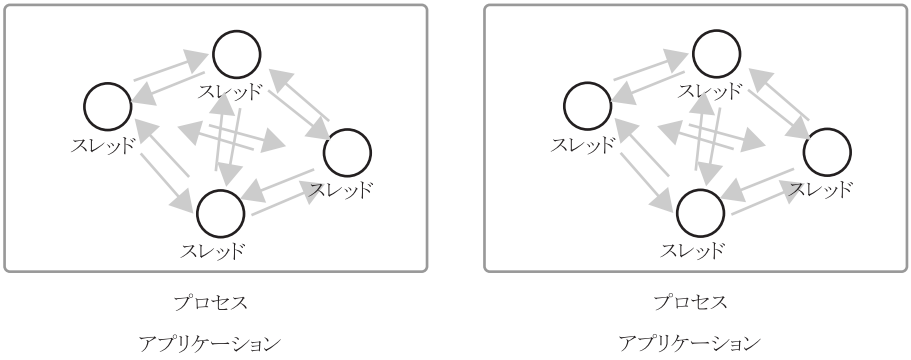

図 5.1: プロセスとスレッドの概念図

メッセージ

MS Windows ではメッセージという概念でプロセス間通信やユーザインタフェースを実現している．キー入力やマウスクリックをユーザが行った場合，対象のウィンドウにキーメッセージやマウスメッセージを通知し，ウィンドウ側がそのメッセージに対応している手続きを実行することでマウスクリックやキーボード入力を実現している．

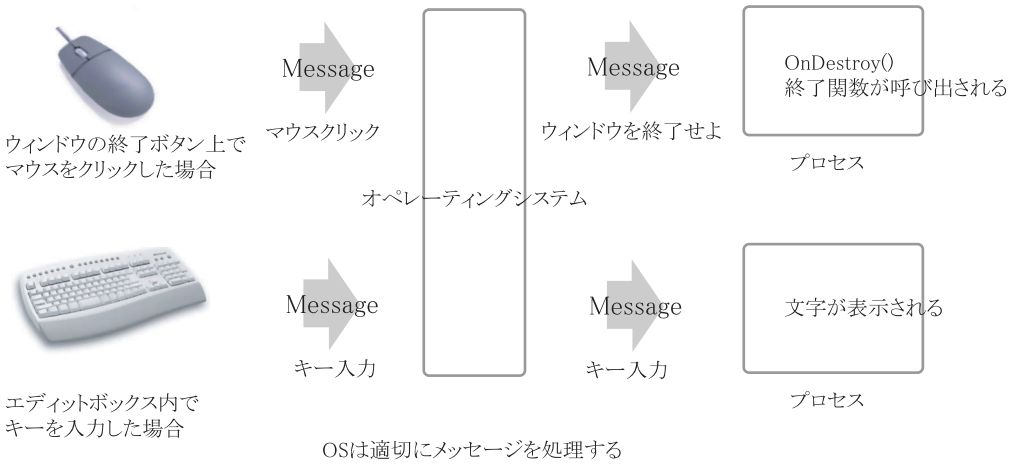

図 5.2: メッセージの概念図

図 5.2 の例では MS Windows に存在するウィンドウの右上の終了ボタンの部分でマウスをクリックすることによって，システムはマウスメッセージを受け取る．システムはマウスクリックを行ったウィンドウにメッセージを通知する．ウィンドウは終了ボタン上で起きたと判断し，コールバック関数の OnDestroy 関数を呼ぶ．

フック処理

しかし上記のメッセージによる通信の方法では他のアプリケーションに対して送られたメッセージの内容を知ることはできない．他のアプリケーションに対するメッセージを処理するためにフック処理の概念を利用する．フックを利用することで本来メッセージが送られるべきアプリケーションからメッセージを横取りし，処理を書き換えることができる．

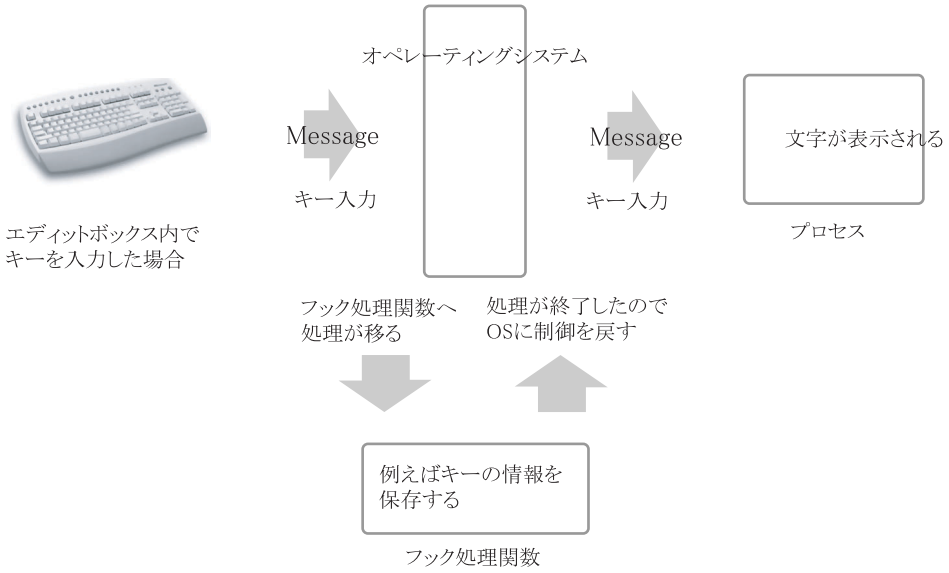

図 5.3: フック処理の概念図

図 5.3 の例では，キー入力を横取りし，キー入力を独自に処理している．図 5.3 の場合は再び処理をシステムに戻しているが，実際は必ずしも処理を戻す必要はない．

コールバック関数

コールバック関数は，関数ポインタをシステムに登録しておくことにより一定の条件が起きた場合に該当する関数を実行する機構である．フック処理を行うためにはフック処理時に処理する関数をコールバック関数として登録する必要がある．

図 5.4 の例では，あらかじめ OnDestroy 関数を登録することによって，終了メッセージが届いた場合には自動的にシステムが OnDestroy 関数を呼び出す．

ハンドル

ハンドルはウィンドウ，メニュやファイルなどのインスタンスを指し示す変数で，主にウィンドウを識別する変数として使用される．同じアプリケーションでもアプリケーショ

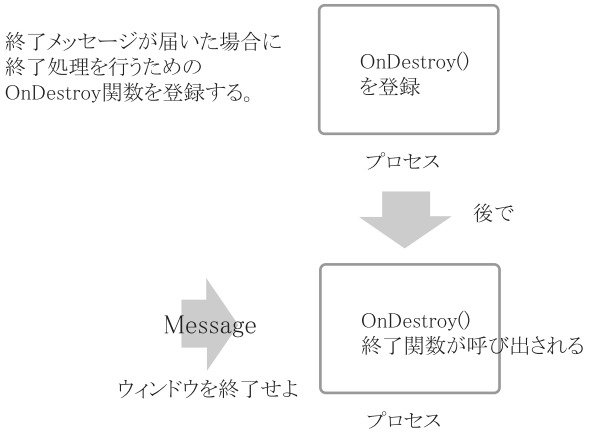

図 5.4: コールバック関数の概念図

ンが起動するごとにハンドルの値は変わるためハンドルの値はコンピュータの利用中にわたるウィンドウの識別子でなく一過的な識別子として利用する．

DLL (Dynamic Link Library)

DLL はアプリケーションでライブラリ中の関数や変数が必要となった場合にライブラリを動的に読み込む事を可能にする機構である．関数や変数をライブラリとして保存することで，複数のアプリケーションに共通して使われるような関数の実装をアプリケーションプログラム内にもつ必要がなくなる．またアプリケーション外部に関数や変数を保持することによりアプリケーション間で同じ関数や変数を利用することができる．MS Windows では DLL を介してフック関数を利用することによって，全てのプロセスに対する入力を検出して保存する事が可能となる．

5.2.2 実装技術

本小節では実装する上で重要な技術として SetWindowsHookEx 関数と，MySQL とのインタフェースである ODBC の 2 点について解説する．

SetWindowsHookEx 関数

本システムではユーザ入力を検出する手段として Windows API 関数の SetWindowsHookEx 関数を使用した．SetWindowsHookEx 関数は，プロセスに対してあるメッセージが送られた場合，状況に応じたコールバック関数を呼び出すために，呼び出されるべきコールバック関数をインストールする関数である．SetWindowsHookEx 関数を使用することによって各プロセスに送られるメッセージを捕捉し，データベースに保存することができる．本システムでは SetWindowsHookEx 関数を全てのプロセスを対象にして入

力検出を行う必要があるのでフック関数を DLL(Dynamic Link Library) 上に定義する．SetWindowsHookEx 関数の宣言は図 5.5 のようになっている．

```
HHOOK SetWindowsHookEx(

  int         idHook,      // インストールするフックの種類
  HOOKPROC    lpfn,        // フックプロシージャのアドレス
  HINSTANCE   hMod,        // アプリケーションインスタンスのハンドル
  DWORD       dwThreadId   // フックをインストールするスレッドの ID

);
```

図 5.5: SetWindowsHookEx 関数の宣言

本システムでは idHook 部分に WH_CBT を使用した．このフックはコンピュータを利用したトレーニング（CBT）アプリケーションの作成に有用なメッセージを監視するフックプロシージャをインストールする．HINSTANCE hMod の部分には lpfn パラメータで指定したフック関数が置かれている DLL のハンドルを指定する．本システムの実装では SetWindowsHookEx 関数の呼び出し元の関数とインストールされるフック関数が同じ DLL 内で定義されているため，呼び出した DLL のインスタンスハンドルを代入する．第 4 引数にはフックをインストールするべきスレッドの ID を代入する．本システムでは全プロセスへを対象に入力を検出するので 0 を代入する．CBTProc は表 5.2 のイベントを検出する．

表 5.2: CBTProc イベントの分類

イベント名	説明
HCBT_ACTIVATE	システムがウィンドウをアクティブ化しようとしている
HCBT_CLICKSKIPPED	システムがマウスメッセージを削除した
HCBT_CREATEWND	ウィンドウが作成されようとしている
HCBT_DESTROYWND	ウィンドウが破棄されようとしている
HCBT_KEYSKIPPED	システムがキーボードメッセージを削除した
HCBT_MINMAX	ウィンドウが最小化，または，最大化されようとしている
HCBT_MOVESIZE	ウィンドウの移動，または，サイズ変更が行われようとしている
HCBT_SETFOCUS	ウィンドウがキーボードフォーカスを受け取ろうとしている
HCBT_SYSCOMMAND	システムコマンドが実行されようとしている

検出されたイベントのウィンドウハンドルからユーザの入力が行われたウィンドウを特定することができる．これらのイベントを個別に処理し，取得した入力の情報をデータベースに格納する．

ODBC

本システムのデータの格納は，プロセス間のデータ通信が容易で確実であるという理由から，リレーショナルデータベースを利用することによって行う．データベースを使用するにあたって ODBC を使用することにより細かなデータベース固有の API にとらわれることなく実装することが可能とある．ODBC は異なる RDBS のインタフェースをラッピングする．ODBC を利用することによって稼動しているデータベースを意識することなくデータベースを利用することが可能となる．例えば MS SQL Server と Oracle はどちらも SQL 問い合わせによるデータ処理をサポートしているが，使用できる SQL 問い合わせには若干の違いがある．もしプログラムの内部でこれらの SQL 問い合わせを直接行っていたならば，データベースを変更する場合，プログラムの変更を余儀なくされる．しかし ODBC に準拠したプログラムを書いている場合，ODBC ドライバを変更するだけでデータベースの変更が可能である．

ODBC を利用する場合，データベースに対して行う操作は大きく分けて以下の三つの部分に分けることができる．

1. データベースに接続
2. データベースとの入出力
3. データベースとの接続解除

以下に上の 3 つの状況で行うべき手順をそれぞれ述べる．

- **データベース接続**

  データベースに接続する手順は次の手順となる．

  1. データベース環境を取得
  2. データベース接続

```
SQLAllocEnv    ( &hEnv )
SQLAllocConnect( hEnv, &hDbc )
SQLConnect     ( hDbc,
                 servername, SQL_NTS,
                 userid,     SQL_NTS,
                 password,   SQL_NTS )
```

図 5.6: データベース接続部

図 5.6 の手順でデータベースに接続する．

- **トランザクション**

本システムの方式では毎回のトランザクションに対してステートメントハンドルを取得している．なぜならWindowsはマルチスレッド実行環境であるので非同期な実行に対してもSQL 問い合わせが正常に動作するようにするには別々のステートメントハンドルを保持しなければならないからである．以下にデータベースとデータを取得する場合，または代入する場合の手順を示す．

1. ステートメントハンドルを取得
2. 準備
3. 変数を代入
4. SQL Queryを発行
5. 変数を取得
6. ステートメントハンドルを開放

呼び出すべき関数は図5.7のようになる．図5.7ではINSERT 文によってデータをデータベースに格納する例を示す．まずSQLPrepare関数はSQL 問い合わせ文の代入を行う．この例ではevent テーブルに値を代入するSQL 問い合わせ文を示す．SQLBindParameter関数はSQL 問い合わせ文に変数を代入する．この例では1番目のパラメタにLONG変数を代入することを示す．SQLExecute関数は実際にSQL 問い合わせ文をデータベースに発行する．

```
#define INSERTMOUSEEVENT \
  "INSERT INTO event VALUES(0, ?, NOW(), ?, ?, ?);";

SQLAllocStmt    ( hDbc, &hStmt )
SQLPrepare      ( hStmt, INSERTMOUSEEVENT, SQL_NTS )
SQLBindParameter( hStmt, 1, SQL_PARAM_INPUT,
                  SQL_C_SLONG, SQL_INTEGER,
                  0, 0, &type, 0, NULL )
SQLExecute      ( hStmt )
SQLFreeStmt     ( hStmt, SQL_DROP)
```

図 5.7: データベースとのトランザクション

SELECT 文でデータを取得する場合，
Executeの終了後SQLBindCol関数とSQLFetch関数を使用し，取得した値を変数に代入する．この場合，get変数に値が代入される．

```
SQLBindCol      ( hStmt, 1, SQL_C_SLONG,
                  &get, 0 NULL)
SQLFetch        ( hStmt )
```

図 5.8: データベースから情報を取得

- **データベース接続を解除**

  本システムが終了する時に ODBC のリソースを開放する．この操作によりデータベースとの接続を解除する．手順は以下のようになる．

  1. データベース接続を解除
  2. データベース環境ハンドルを開放

  呼び出すべき関数は図 5.9 のようになる．

```
SQLDisconnect ( hDbc )
SQLFreeConnect( hDbc )
SQLFreeEnv    ( hEnv )
```

図 5.9: データベースとの接続を解除

5.3 システムの処理

本節では本システムの処理手順を示す．まず本システムの開始時と終了時に行うべき処理を述べる．次にシステム稼動時に行う処理の流れを述べる．また，図 5.10 にシステムの処理の概要を示す．図 5.10 ではクライアントウィンドウがシステムの開始と終了を行う構図を示している．クライアントウィンドウによってフックされたフック関数は他のウィンドウに対して送信されたメッセージを受け取り，処理を行う．

5.3.1 システム開始処理

本小節では本システムを開始するときに必要な処理を述べる．本システムの開始時に行う必要のある処理を以下に挙げる．

1. データベース接続を開始する．

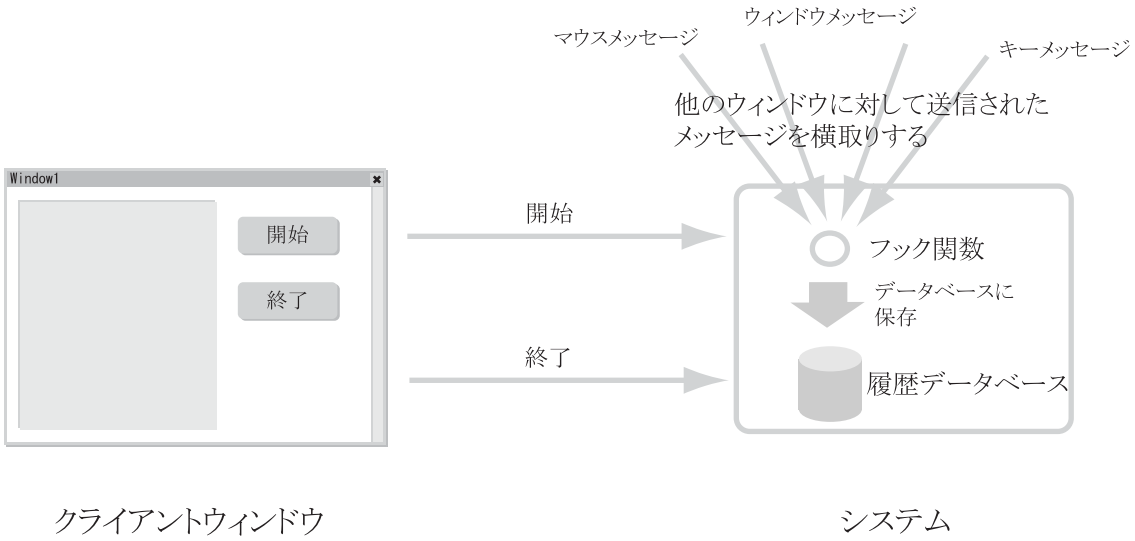

図 5.10: システム処理の流れ

2. SetWindowsHookEx 関数でフック関数を登録する.

システムが開始された後，システムは上記の手順を踏む．システムは必ずフック関数登録前にデータベース接続を開始する必要がある．なぜならフック関数を登録後は非同期にデータベースにアクセスを試みるからである．

5.3.2 システム終了処理

前小節と同様に本小節では本システムを終了するときに必要な処理を述べる．終了時に行う必要のある処理を以下に挙げる．

1. UnhookWindowsHookEx 関数でフック関数の登録を抹消する.

2. データベース接続を終了する.

システムは上記の手順を踏んでからシステムは終了する．システムの終了処理も開始処理と同様の理由でデータベース接続の終了はフック関数の登録解除後に行う必要がある.

5.3.3 システム稼動時の処理

システム稼動時にシステムが行う必要のある処理は 2 種類に分類できる．第一にユーザの入力内容を保存する処理，第二にユーザの検索要求に従い検索を行う処理である．以下にそれぞれの処理手順を述べる.

まず入力内容を保存する処理の流れは以下のようになる.

1. フック処理により捕捉された入力を受け取る．

2. 入力内容をデータベースに保存する．

3. 入力内容を本システムのウィンドウに通知する．

次に検索要求の処理の流れは以下のようになる．

1. フック処理により捕捉された検索要求の入力を受け取る．

2. 検索を行う．

3. 検索結果を本システムのウィンドウに通知する．

処理手順の詳細の説明はシステム構成要素の章で行っている．

5.4 基本データ構造

本節では実装に関わるデータ構造について詳細を述べる．本システムではデータベーステーブルとして event テーブル (表 5.3 参照) と app テーブル (表 5.4 参照) の 2 つを定義する．以下にそれぞれのテーブルの役割について詳細を述べる．

表 5.3: event テーブルの定義

名前	説明
id	ID
event_type	表 5.2 参照
time	イベントが起こった時刻
msg	イベントを再現可能な追加情報
app_id	イベントが起こったアプリケーションの ID
instset_id	入力群の ID

event テーブルは Windows の CBTProc が行う分類に従って各イベントを保存する．event_type は表 5.2 で分類される CBTProc 型のフックのフックコードを保存する．このことにより，生じたイベントの種類を記録することができる．time フィールドはイベントが生じた時刻をタイムスタンプ形式で記録する．msg はイベントを再現するために必要なデータをバイナリ形式で記録する．app_id フィールドにはイベントが起きたアプリケーションの ID を記録する．ID は app テーブルの該当するアプリケーションの ID である．instset_id フィールドには入力群の ID を記録する．このフィールドのデータはユーザが入力対象ウィンドウを変えるごとに値が変わる．

本実装ではウィンドウはボタンやサブウィンドウを含めて，一つのウィンドウを一つの領域とみなす．それぞれのウィンドウを判別するためウィンドウの名前を特定し，情報を保存しておく．ウィンドウ名の命名は以下の手順に従う．

表 5.4: app テーブルの定義

名前	説明
id	ID
root	このウィンドウの所属しているウィンドウの ID
name	ウィンドウの名前

1. 入力取得の際，取得したウィンドウハンドルを元に GetClassName 関数でウィンドウクラス名を取得する．
2. GetParent 関数を実行し，親ウィンドウが存在すれば親ウィンドウ名を GetClassName 関数を使用して取得する．
3. 2 の操作を繰り返し親ウィンドウが存在しない状態になるまで繰り返す．
4. 取得した名前を親ウィンドウから階層順に「.」でつなげて出来上がった文字列を領域名とする．

例えば「#32770.Button.」という領域名は#32770 という名前をもつウィンドウの子ウィンドウの Button という名前のウィンドウを示す．

5.5 システム構成要素

本節では各システム構成要素の実装方法を述べる．システム構成要素として，履歴保存機能，検索機能，通知機能，実行機能について述べる．

5.5.1 履歴保存機能

本小節では履歴保存機能の実装方法を述べる．履歴保存機能の基本的な手順は，以下の通りである．

1. フック処理により捕捉された入力を受け取る．
2. 入力内容をデータベースに保存する．
3. 入力内容を本システムのウィンドウに通知する．

フック処理の手順は SetWindowsHookEx 関数の章で説明したため省略する．入力を保存する関数として図 5.11 の宣言で InsertEvent 関数を作成した．

```
unsigned int InsertEvent(
                            unsigned int     eventType,
                            void*            eventStructPtr,
                            unsigned int     sizeOfEvent,
                            unsigned int     appId
                          );
```

図 5.11: InsertEvent 関数の宣言

図 5.11 の関数はデータベースの event テーブルに情報を登録する．eventType は入力の種類，appId はこの入力を受け取るアプリケーション，eventStructPtr は入力の付加情報の構造体のポインタ，sizeOfEvent は構造体のサイズを表す．返り値は挿入したデータの id となる．ここでシステムは，入力の種類により構造体のサイズや内容が異なるので，サイズや内容に応じて場合を分けて処理を行っている．この関数を実行した後，返り値の id をクライアントシステムに通知する．通知した id を元にクライアントウィンドウは次の処理を行う．

5.5.2 検索機能

本小節では検索機能について実装方法を解説する．まず検索関数として図 5.12 の宣言で SearchInstruction 関数を実装した．ユーザは Ctrl キーを押すことでシステムは SearchInstruction 関数を呼び出すことができる．

```
int  SearchInstruction ( unsigned int id );
```

図 5.12: SearchInstruction 関数

引数の id には直前の入力の id を代入する．直前の入力の id は図 5.13 の SQL 問い合わせで取得する．ここで SQL 問い合わせ文に LAST_INSERT_ID() を使用しない理由は異なるデータベース接続が直前に入力情報を挿入していた場合，その入力情報の取得を逃すからである．SearchInstruction 関数の返り値は id と同じ入力の種類で，かつ，id より前で

```
SELECT max(id) FROM event;
```

図 5.13: 直前の入力の id を取得する SQL 問い合わせ文

もっとも直近のレコードの id となる．SearchInstruction 関数内で使用している SQL 問い合わせ文は図 5.14 となっている．この結果，その過去の入力情報からその過去の入力を

```
SELECT app.root, event.instset_id, app.id \
FROM event LEFT JOIN app ON event.app_id = app.id WHERE event.id = ?;
```

図 5.14: id から入力情報を取得する SQL 問い合わせ文

含む命令を取得する．

5.5.3 通知機能

本小節では通知機能について実装方法を解説する．通知機能は検索機能で検索されたパターンからシステムが最適と判断したものをユーザに通知して，実行の判断をユーザに問い合わせる．本実装方式ではクライアントウィンドウから検索機能である SearchInstruction 関数を呼び出し，その結果受け取った値を表示する．図 5.15 は通知機能のスクリーンショッ

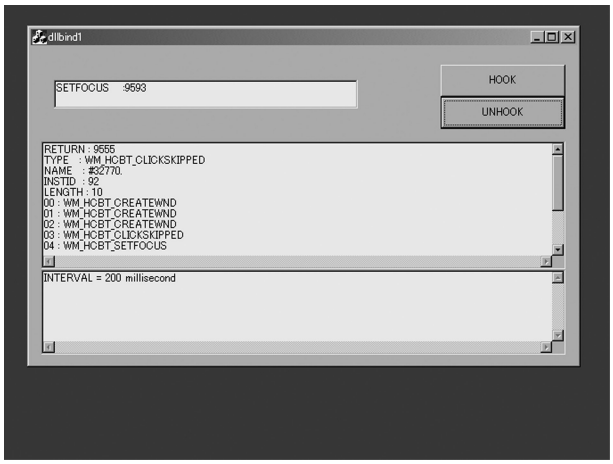

図 5.15: 通知画面

トを表している．中央のエディットボックスに通知機能が通知した候補を表示している．

通知ウィンドウに情報を伝達する方法として，WindowsAPI 関数の PostMessage 関数を利用する．PostMessage 関数の宣言は図 5.16 となっている．この API 関数は第 1 引数の hWnd が指し示す Window に対して Msg の種類のメッセージを送る．ここでは通知機能は第 3 引数の wParam に SearchInstruction 関数の呼び出しの結果得られた id を代入し，ユーザに候補を表示するウィンドウに対して PostMessage を送る．メッセージを受け取ったクライアントウィンドウは id を元に GetInstsetFromID 関数を呼び出す．ここで，GetInstsetFromID 関数は引数の入力を含む入力群を取得するための関数である．本実装では定義を図 5.17 とした．GetInstsetFromID 関数は inst_id に所属する入力群を取得する．

```
BOOL PostMessage(
  HWND    hWnd,   // handle of destination window
  UINT    Msg,    // message to post
  WPARAM  wParam, // first message parameter
  LPARAM  lParam  // second message parameter
);
```

図 5.16: PostMessage 関数の宣言

```
GetInstsetFromID(
                 unsigned int  inst_id,
                 Event         event*,
                 unsinged int* eventLength
                );
```

図 5.17: GetInstsetFromID 関数の宣言

通知機能は入力群をユーザに表示することによってユーザに確認を求める．本実装では Enter キーを押すことによってユーザはシステムに対して実行許可を与えることができる．

5.5.4 実行機能

本小節では実行機能について実装方法を解説する．命令実行の実装では図 5.18 の関数を実装した．実行機能は通知機能によってユーザの確認を取った入力群を再現することで命令を実行する．instId は event テーブルの instset_id フィールドの値である．ExecuteInstruction

```
unsigned int  ExecuteInstruction ( unsigned int instId );
```

図 5.18: ExecuteInstruction 関数

関数内では，まず前小節で説明した GetInstsetFromID 関数を使用することによって instId に所属する入力情報群を取得し，次にその入力群が行われたウィンドウを取得している．その後，WindowsAPI 関数の EnumWindows 関数と GetClassName 関数を使用して現在起動しているウィンドウの名前を取得し，該当するウィンドウの起動の有無を確認する．現在該当のウィンドウが起動している場合は，そのウィンドウに対して PostMessage 関数で入力を行う．該当するウィンドウが起動していない場合は何も行わない．

5.6　本章のまとめ

本章ではシステムの実装方法を述べた．次章では本章で実装したシステムの定性評価，定量評価を行う．

第6章　システムの評価

本章では実装したシステムの定量的評価と定性的評価を述べる．定量的評価では各機能の実行にかかる時間を計測した．定性的評価では関連研究との比較を行った．

6.1 定量的評価

本節では定量的評価を行う．まずはじめに定量的評価を行うための測定環境と測定方法を述べ，次に測定結果を述べる．

6.1.1 測定環境

本システムの各機能の実行時間を計測するために表 6.1 に示す環境を使用した．

表 6.1: 測定環境

項目	
ハードウェア	IBM ThinkPad A21p
CPU	Intel Pentium III 700 MHz
メモリ	384 MB
OS	Windows2000 Service Pack2
入力機器	キーボード，3ボタンマウス

6.1.2 測定方法

本節では定量的評価にあたって以下の評価項目を設定した．

検索時間の計測

ユーザが検索を要求して結果が得られるまでに要する検索時間の計測を行った．スケーラビリティをはかるためにデータベースのエントリ数を変え，2 回の評価を行った．1 回目の計測ではデータベースのエントリ数 500 から 1100 において無作為に 100 回計測を行った．2 回目はエントリ数 5000 から 5500 において無作為に 100 回計測を行った．そして 2 つの測定結果の違いを調べた. 計測時間は 1 回の計測値を得るのに 100 回の検索を行った. 実際の検索時間は 100 分の 1 となる．

検索時間の測定手段として Windows API の GetTickCount() 関数を使用した．この関数はミリ秒単位で時刻を測定できる．

6.1.3 評価結果

評価結果を図 6.1 と図 6.1 に示す.

1 回目の検索の平均時間は 251 ミリ秒であった.

2 回目の検索の平均時間は 1361 ミリ秒であった.

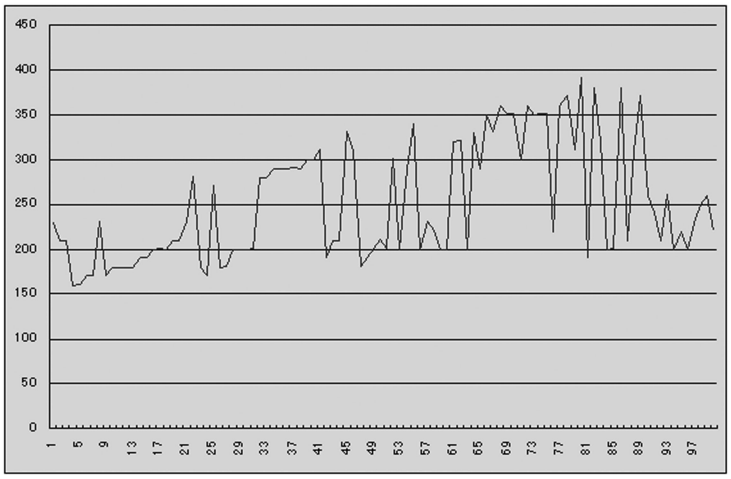

図 6.1: レコード数 500 から 1100 までで無作為に検索を行った測定結果

6.1.4 評価の考察

一回の検索にかかる時間は図 6.2 においても平均 13 ミリ秒と実用に足るものである．しかし 1 秒に平均 1 回の入力で 2 時間のコンピュータ利用を想定すると 1 日に約 5 千のデータを扱う必要がある．このことを考慮に入れると本研究の方式で履歴を扱うには最低でも 50000 エントリーを扱う必要がある．このようにより巨大なデータを検索する場合はこれ以上の負荷をシステムにかける可能性がある．

6.2 定性的評価

本節では定性的評価を行う．定性的評価を行うにあたり，まず評価項目を列挙する．次に評価の結果を述べる．

6.2.1 評価項目

本小節では定性的評価を行うにあたって，評価項目を設定する．評価結果をまとめると表 6.2 となる．個別の評価項目は次小節以降で個別に解説する．

表 6.2: 関連研究との評価

	本システム	A Temporal Model for Multi-Level Undo and Redo	Repeat And Predict
汎用性	○	×	×
システムを使用する前提技術	○	○	○

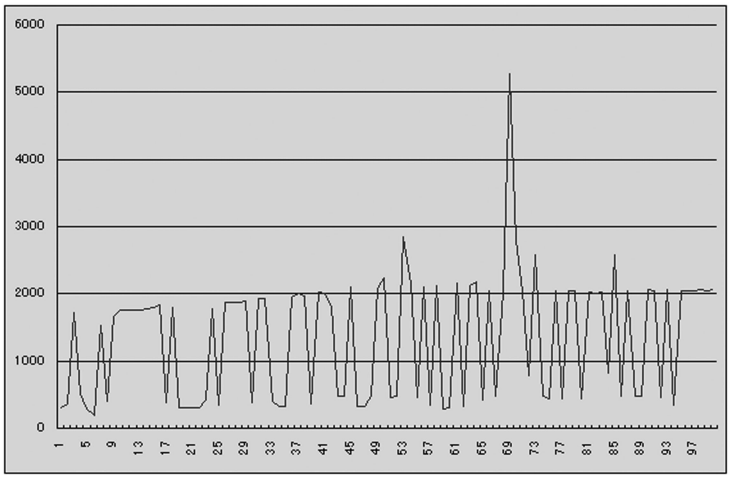

図 6.2: レコード数 5000 から 7000 までで無作為に検索を行った測定結果

汎用性

入力支援システムの汎用性を評価する．入力支援システムを使用する場合，そのシステムの適用範囲によってシステムの有用性が大きく異なる．汎用性の基準はアプリケーション依存の部分の多少である．

システムを利用する際の前提技術

入力支援システムの使用にあたって高度な前提知識を要求するシステムは新規に使用する場合に敷居が高い．ここでは評価基準としてシステムの使用に際しての前提技術の多少を評価基準とした．

6.2.2 評価の考察

本小節では汎用性とシステムを利用する際の前提技術の 2 点について考察する．

汎用性

本システムはアプリケーションの機能に依存しない．アプリケーションにまたがった入力支援を実現することが可能である．しかしコンピュータ環境にまたがった入力支援を行うことは可能でない．A Temporal Model for Multi-Level Undo and Redo ではシステム全体の履歴を扱うがシステムの適用範囲はホワイトボードとかなり限定されている．Repeat and Predict では文字入力のみかつ単一アプリケーション内での履歴を扱っている．よって先の二つのシステムは入力支援機構として汎用性が高いとは言えない．

システムを利用する際の前提技術

本システムで使用する入力は検索入力と実行入力の二種類のみであり十分に単純である．Repeat and Predict は繰り返しキーと予測キーの二種類のみで入力支援を行っており十分に単純である．これらの三つのシステムはいずれも少量の前提技術のみで高度の入力支援機能を利用することが可能である．

6.3 本章のまとめ

本章では本システムの評価を行うことによって本研究の特性を示した．定量的評価では各機能を実行する際に要する時間を測定し，オーバヘッドを検出した．定性的評価では3章で述べた関連研究と本システムを比較した．

第7章　結論

7.1　まとめ

本論文では入力履歴を利用したユーザ入力支援機構を提案し，システムの設計と実装を述べた．本機構は履歴保存機能，検索機能，通知機能，実行機能から構成される．履歴保存機能ではユーザの入力を受け取り，入力の再現に必要な情報を保存する．検索機能はユーザの検索要求に応じて履歴保存機能が保存したユーザ入力情報を検索し，候補を通知機能に伝達する．通知機能は履歴機能から受け取った候補をユーザに通知し，ユーザの判断を仰ぐ．実行機能はユーザの確認が取れた命令を実行する．このシステムを利用することによりユーザは部分的に入力した候補を検索し，効率よく命令を実行することができる．

7.2　実現した点

本システムではユーザの入力情報の保存を可能にした．本システムを構築することによって複数の領域にわたって利用できる入力支援機構を実現した．このことによりユーザは入力支援機構の違いを意識することなく入力支援機構を利用できるようになった．

7.3　今後の課題

本節では今後の展開として次の 3 点を述べる．

他人の履歴を検索

本研究では一人のユーザが本システムを利用することを前提としてシステムを構築した．複数のユーザが本システムを利用することによって，相互に履歴の検索が可能となり，他人の入力履歴を再利用できる．今後の研究では複数のユーザが利用するコンピュータ環境において履歴を共有する手法を提案する．

スケーラビリティ

本論文で述べた実装方法は多量のコンピュータリソースを消費する．今後の研究では，より少量のコンピュータリソースでユーザ入力支援機構を実現する方法を模索する．またコンピュータリソースの少ないコンピュータ環境における本システムの実現を行う．

コンピュータ環境透過性

本論文で実現したシステムは Windows に特化して実装を行った．他の OS 環境では本システムを利用することができない．今後の研究では複数のコンピュータ環境において本システムの利用を可能にする方法を模索する．

謝辞

本研究を進めるにあたり，丁寧な御指導を頂きました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝いたします．

慶應義塾大学徳田・村井・楠本・中村研究会の諸先輩方には，卒論にあたった一年を通して多くの貴重な御助言や御指導を頂きました．特に慶應義塾大学政策・メディア研究科修士 2 年の由良淳一氏，修士 1 年の石井かおり氏，修士 1 年の桐原幸彦氏には，本論文執筆にあたって絶えざる励ましと御指導を頂きました．西尾信彦氏，戸辺義人氏，慶應義塾大学政策・メディア研究科修士 2 年の岩井将之氏，楠本晶彦氏，原島章介氏，修士 1 年の松宮健太氏，青木崇行氏にも，多忙の中熱心な指導とさまざまな助言を頂きました．

ここに深い感謝の念を表します．

1 年間の研究生活の中，同じ Keio Media Space Family(KMSF) グループで研究活動を行い，多くの時間を共に過ごした柳原正氏，中西健一氏，伊藤昌毅氏には研究において多くの刺激を受け，学んだことを深く感謝します．同じく 1 年間の研究生活を共に過ごした 4 年生，卒論を提出する際に手伝って頂いた後輩方にも深く感謝します．最後に，研究会に限らず，精神的な面で支えになって頂いた多くの友人たちに深い感謝の念を表し，謝辞と致します．

平成 14 年 1 月 31 日
慶應義塾大学 環境情報学部 4 年
大藤 徹

参考文献

[1] Apple Computer, Inc. *AppleScript*, 2001. http://www.apple.com/applescript/.

[2] Allen Cypher, editor. *Watch What I Do:Programming by Demonstration*. 1993. http://www.acypher.com/wwid/index.html.

[3] W. Keith Edwards, Takeo Igarashi, Anthony LaMarca, and Elizabeth D. Mynatt. A Temporal Model for Multi-Level Undo and Redo. *13th Annual Symposium on User Interface Software and Technology, ACM UIST'00, San Diego, CA , November 5-8, 2000, pp.31-40.*, 2000.

[4] Free Software Foundation, Inc. *bash*. http://www.gnu.org/software/bash/bash.html.

[5] Henry Lieberman, editor. *Your Wish is My Command: programming by example*. 2001.

[6] David S. Kosbie and Brad A. Myers. A System-Wide Macro Facility Based on Aggregate Events. , 2000.

[7] Toshiyuki Masui and Ken Nakazawa. Repeat and Predict - Two Keys to Effcient Text Editing. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1994.

[8] Microsoft Corporation. *Microsoft IME 2000*. http://www.microsoft.com/.