修士論文

事例を用いた UNIX 問題解決支援システム CUPS の作成

指導教員
山崎 勝弘 教授

提出者
氏名 脇 武也
学籍番号 6114940103-1

提出日
1996 年 2 月 15 日

立命館大学理工学研究科情報システム学専攻

内容梗概

UNIX はネットワークに長けた高機能な OS であり、非常に便利なものである。しかし、覚えねばならぬことが多いと言う側面をもつ OS でもある。一方で、コンピュータの需要は近年ますます高まり、より多くの人が UNIX を用いるようになった。このため、UNIX の利用を助ける、あるいは UNIX 利用上の問題を解決してくれるシステムの需要が高まっている。

UNIX 使用時に生じる問題の多くは似通ったものであり、今までに他の誰かが同じ問題に遭遇し、解法を見いだしていることが多い。しかし、それにもかかわらず一般にその問題解決の経験が他人に知らされることはほとんど無いので、同じ問題に遭遇した人はまた新たにその問題を解決しているのが現状である。しかし、問題解決の経験を蓄積しておけば、これを参考に現在の問題を解決することが出来、多くの時間を節約できる。

UNIX に関する問題は多岐に渡り、また常に変化している。そのためこの問題解決システムにとって「いかにして多くの事例を迅速に集めるか」という問題が非常に重要である。本システムでは、上級ユーザーばかりではなく初級から上級までの全てのユーザーから事例を収集することとした。

事例を用いた UNIX 問題解決支援システム CUPS(Case-based UNIX Problem Solver) は、統括部、事例ベース、意味ネットワーク、GUI、メール読み込み部の５つの独立したプログラムからなる。これらのプログラムはソケットを介して互いに通信し、処理を進める。また、多くのユーザーから集められた事例を管理するため、「有用」「無用」「要改善」のマーク付け、事例への追加記入、様々な事例管理情報の表示、ファジィ検索語、意味ネットワークの５つの機能を付加した。

そして、実際にシステムを作成し（意味ネットワークを除く）、事例を収集した。今までに、研究室の１０名の方から、あわせて６６個の事例が登録された。登録された事例を見てみると、ユーザーの利用頻度が高いものに関して、あるいはトレンドな情報に関して事例が集中する傾向があった。事例の作り手と事例の利用者が同じであるので、自然と利用者に必要な事例が収集される傾向がある。

初級者から上級者まで多くの人から事例を収集した訳だが、実際に収集してみるとやはり上級者の方が多くの有用な事例を登録できた。しかし、初級者でも有用な事例を作れている例は多かった。確かに初級者が上級者向けの事例をつくることは困難である。しかし、時として初級者にしか作れない（上級者にはその必要性が理解できず作ろうとしない）事例がある。様々なレベルの人が事例を作ることで、様々なレベルの人に対応した事例が集められる。

目次

図目次

表目次

1 はじめに

UNIX はネットワークに長けた高機能な OS であり、非常に便利なものである。しかし、覚えねばならぬことが多いと言う側面をもつ OS でもある。一方で、コンピュータの需要は近年ますます高まり、より多くの人が UNIX を用いるようになった。このことにより、UNIX の利用を助ける、あるいは UNIX 利用上の問題を解決してくれるシステムの必要性は今後より増加して行くと思われる。

問題解決システム、一般的にはエキスパートシステムと呼ばれるものは、通常 専門家の持つ問題解決の知識をコンピューター化しておき問題の解決に利用する。しかし専門家の知識をコンピューター化するという作業はかなり困難なもので、知識獲得ボトルネックという問題を抱えている。これは専門家が問題を解決する場合、過去の類似した事例を思い浮かべて問題を解決することが多く、決してその専門家がルール化されているような明確な問題解決手法を用いて問題を解決している訳ではない点に起因している。そこで、専門家と同じ手順、つまり過去の経験を蓄積しておき、問題が生じたときにそれを用いて問題を解決すると言う手順を用いる問題解決のアプローチが事例ベース推論といわれる方法である。本研究ではこの事例ベース推論のアプローチを用いることとした。

UNIX 使用時に生じる問題の多くは似通ったものであり、今までに他の誰かが同じ問題に遭遇し、解法を見いだしていることが多い。しかし、それにもかかわらず一般にその問題解決の経験が他人に知らされることはほとんど無いので、同じ問題に遭遇した人はまた新たにその問題を解決しているのが現状である。また仮に同じ問題を過去に解決した人を見つけ、それに関して質問出来たとしても、その人がその問題解決の詳細を覚えているとは限らない。特に設定に関するものなどは一度解決してしまうと問題が生じない限り思い出されることがないので、忘れてしまっていることがほとんどであると思われる。そこで問題解決の経験を蓄積しておき、これを参考に現在の問題を行えれば多くの時間を節約できると思われる。

さらに本システムでは、事例ベースを構築するにあたり、事例を上級者ばかりではなく、初心者から上級者の全てのユーザーから収集することにした。UNIX に関する問題は多岐に渡り、また常に変化している。そのためこの問題解決システムにとって「いかにして多くの事例を迅速に集めるか」という問題が非常に重要である。この重要な問題を、「初心者から上級者の全てのユーザー」つまりは「大勢の人」から事例を収集することにより解決することとした。UNIX の世界では個人個人がネットワークでつながっていて相互にやり取りができるので、これを有効に用い、「大勢の人」から事例を集めることが可能である。

「大勢の人」から事例を集めると言うことは広い分野から多くの事例を大量に集められ大変魅力的であるが、事例を有効に利用することが困難になる。つまり、個々の事例のフォーマットを統一することが困難であるので、索引語を統一出来ず事例の検索が困難となったり、無用な事例が多く発生してしまったりすることが考えられる。また、多くの事例を集めるが故にそれらの事例の管理は困難なものとなり、大きな問題である。これらの問題に対して、本システムでは「ファジィ検索」や「ユーザによる事例の管理」と言う方法を用いて解決を計る。

本研究の目標としては実用的なＵＮＩＸ問題解決支援システムを作成することであり、

それは以下のようなものでなければならない。

- ＵＮＩＸに関する事例を迅速に収集できる。
- 事例を効率的に管理できる。
- 利用者が事例を容易に検索できる。

本論文では、２章でCUPS(Case-based UNIX Problem Solver)の全体構成、システム構成要素、システム実現方法について説明した後、３章で個々の事例の構成およびその具体的収集、管理方法について述べ、４章で事例の検索方法について述べ、５章で本システムを評価する。

プログラムは主にC++を用いて作成し、プロトコルの記述等の文字列解析部にはflexを併用した。また、GUI部の作成にはMotifを、ソケットへのインターフェースにはGnanasekaran Swaminathan氏の「C++ socket library Version 1.9」を利用した。

2　CUPS の構成

2.1　システム構成

事例を用いた UNIX 問題解決支援システム CUPS(Case-based UNIX Problem Solver) のシステム構成を図 1に示す。

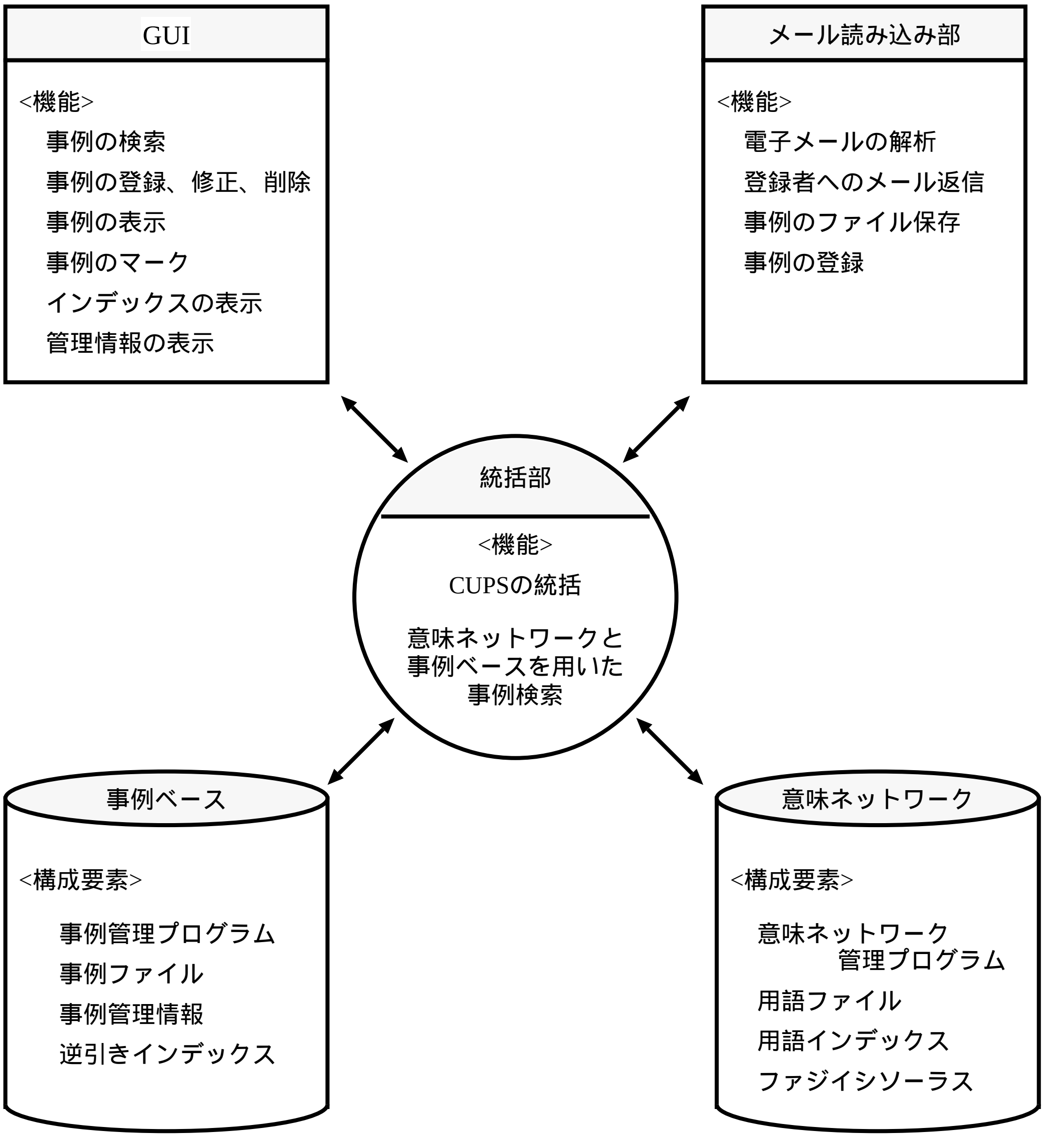


図 1: システム構成

この図が示すようにCUPSは大きく5つの部分に分類される。それぞれの主な役割は次のようである。

統括部
CUPSの中心部分で、下記の4つのパーツを統括する。

GUI
ユーザーの要求に視覚的に分かりやすい形で応じ、その要求を統括部に伝達する。

メール読み込み部
電子メールを読み込み、解釈し、事例として登録する。

事例ベース
ユーザーの作成した事例を蓄積する。

意味ネットワーク
ファジィ・シソーラスを作成する。

これらの各部分はそれぞれ独立したプログラムとして構成され、互いにソケットを用いて通信しながら処理を進める。これらを全て独立したプログラムとしたのは、CIA等の別のプログラムと事例ベース、意味ネットワークを共有することを考慮してである。

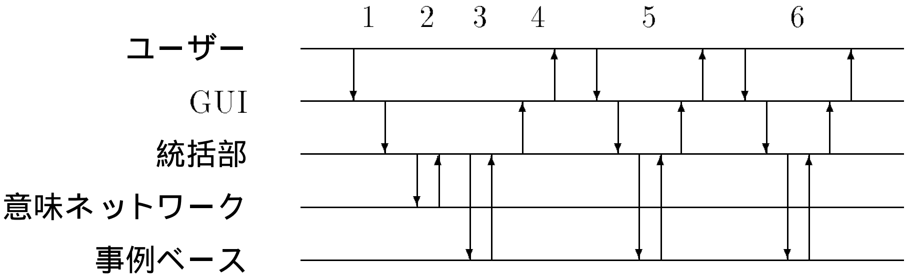
事例ベース、意味ネットワーク、統括部はデーモンプロセスであり、常にクライアントの接続を待つ。そして接続すると直ちにフォークしまた新たな接続を待つ。

2.1.1 統括部

CUPS の処理を統括する役割を担い、CUPS の全ての処理はここを介して行われる。統括部の行う具体的な処理は以下のようである。

1. 検索語による事例の検索
   - 意味ネットワークから関連語リストの読み出し
   - 関連語を事例ベースで検索
   - 対象の限定
   - 検索語との関連度による事例のソート
2. 事例ベースへの要求転送
   - 事例の登録、修正、削除、読み出し
   - 事例への追加記入
   - 事例のマーク付け
   - インデックスの読み出し
   - 管理情報の読み出し

このように、事例の検索は統括部が中心となって行われる。その事例検索処理の流れを図 2に示す。



1 検索語の入力

2 検索語を関連語リストに変換

3 関連語リストと索引語の比較

4 関連事例リストの表示

5,6 事例内容の読みだし

図 2: 事例検索処理の流れ

2.1.2　事例ベース

ユーザーの作成した事例を蓄積しておくことが、事例ベースの主な役割である。事例ベースの構成を図 3に示す。

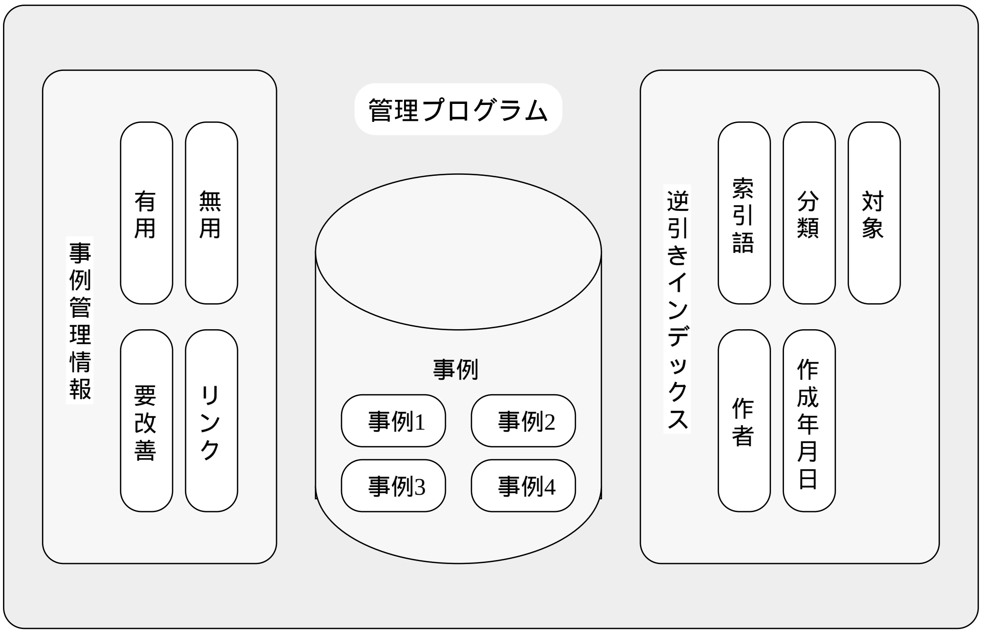


図 3: 事例ベースの構成

この図が示すように、事例ベースは以下の 4 つの要素で構成される。

1. 事例ファイル
2. 事例管理情報
    - 「有用」マーク情報
    - 「無用」マーク情報
    - 「要改善」マーク情報
    - リンク情報
3. 逆引きインデックス
    - 分類インデックス
    - 対象インデックス
    - 索引語インデックス
    - 作成年月日インデックス

    - 作者インデックス

4. 管理プログラム

そしてこれらの構成要素により、事例ベースは以下のようなサービスを提供する。

1. 事例の登録
    - 事例ファイルの作成
    - 逆引きインデックスの更新
2. 事例の修正
    - 事例ファイルの修正
    - 逆引きインデックスの更新
3. 事例の削除
    - 事例ファイルの削除
    - 逆引きインデックスの更新
4. 事例への追加記入
    - 事例の登録
    - リンク情報の作成
5. 事例の読み出し
6. 事例へのマーク付け
    - 「有用」「無用」「改善」マーク情報の更新。
7. 逆引きインデックスの読み出し、検索
8. 管理情報の読み出し、検索

2.1.3 意味ネットワーク

UNIXに関する用語を収集して意味ネットワークを形成し、それを用いてファジィ・シソーラスを作成し、検索語を重みをもった関連語リストに変換するのが役割である。

意味ネットワーク部は以下のような構成要素よりなる。ここでファジィシソーラスファイルとは、この意味ネットワークを用いて作られるファジィシソーラスを予め計算したものを格納しておくファイルである。このファイルを用いて高速に検索語を関連語リストに変換する。

1. 用語ファイル
2. 用語インデックスファイル
3. ファジィシソーラスファイル
4. 意味ネットワーク管理プログラム

そしてこれらの構成要素により、意味ネットワーク部は以下のようなサービスを提供する。

1. 関連語リストへの変換
    - ファジィシソーラスファイルの検索
2. 用語の登録
    - 用語ファイルの作成
    - 用語インデックスファイルの更新
    - ファジィシソーラスファイルの更新
3. 用語の削除
    - 用語ファイルの削除
    - 用語インデックスファイルの更新
    - ファジィシソーラスファイルの更新
4. 用語の修正
    - 用語ファイルの修正
    - ファジィシソーラスファイルの更新
5. 用語の読み出し

2.1.4 GUI

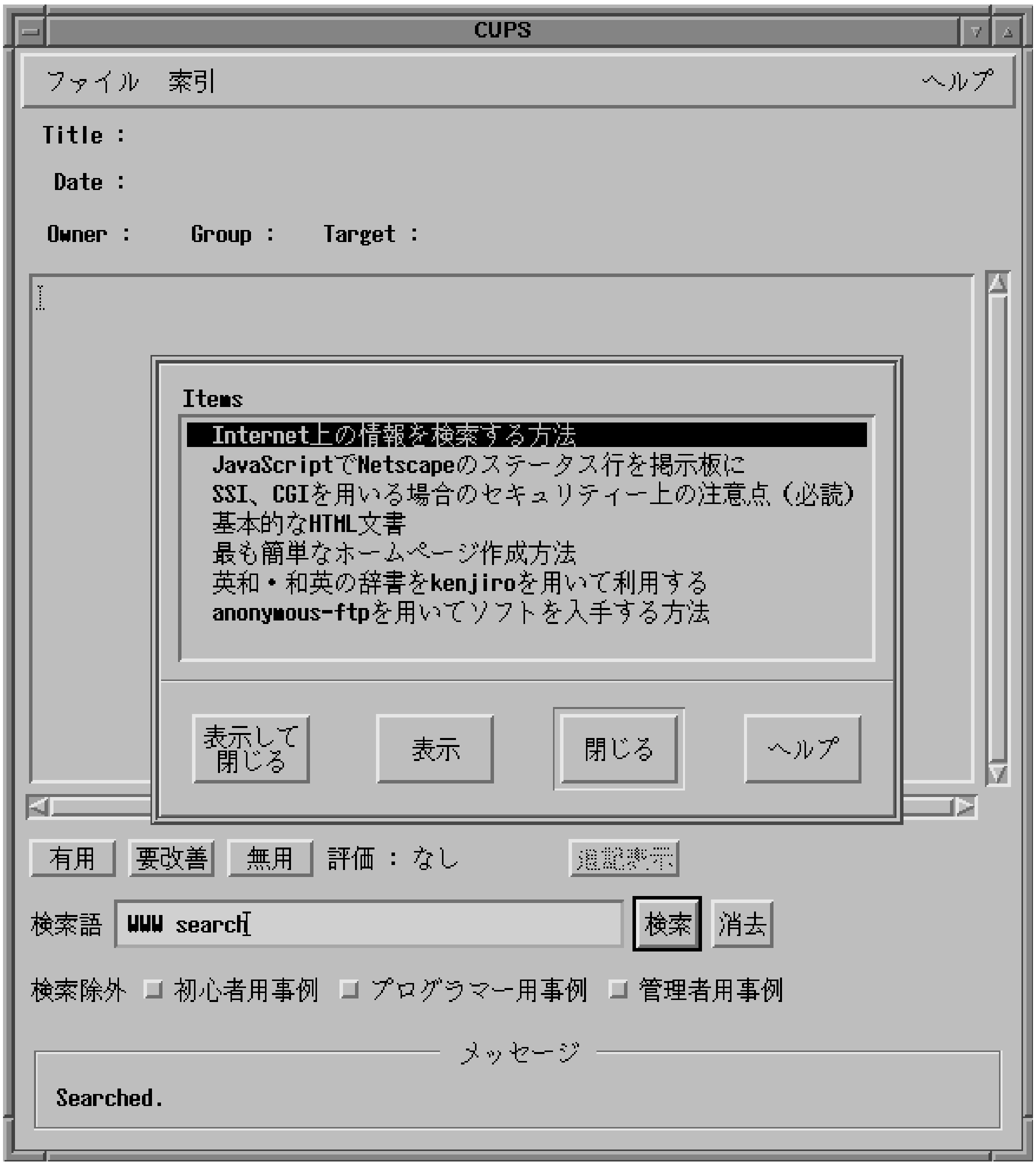
ユーザーの要求を統括部に伝達する役割を担う。Motif を用いたものと WWW 用のものの 2 つが用意されている。WWW 用のものは検索語を用いた事例検索しか出来ないが、Motif を用いたものには以下のような多くの機能がある。

1. 事例の登録、修正、削除
2. 事例への追加記入
3. 対象を限定した、検索語による事例の検索
4. 事例、追加記入事例の表示
5. 事例の詳細情報の表示
6. 事例へのマーク付け
    - 「有用」マーク
    - 「無用」マーク
    - 「要改善」マーク
7. インデックスの表示
    - 分類別
    - 対象別
    - 索引語別
    - 作成年月日別
    - 作者別
8. 管理情報の表示
    - 「有用」マークのついているもの
    - 「無用」マークのついているもの
    - 「要改善」マークのついているもの
9. 通信状態の表示

実際の Motif 版の GUI の様子を、図 4,5,6に示す。図 4は事例を検索しているところ、図 5は事例を表示しているところ、図 6は詳細情報を表示しているところである。この Motif 版の GUI はウィジェット(1)を図 7のように並べて構成されている。なお、WWW 版の GUI の外形は付録を参照。

---

(1) ラベルやボタン等の GUI の部品のこと

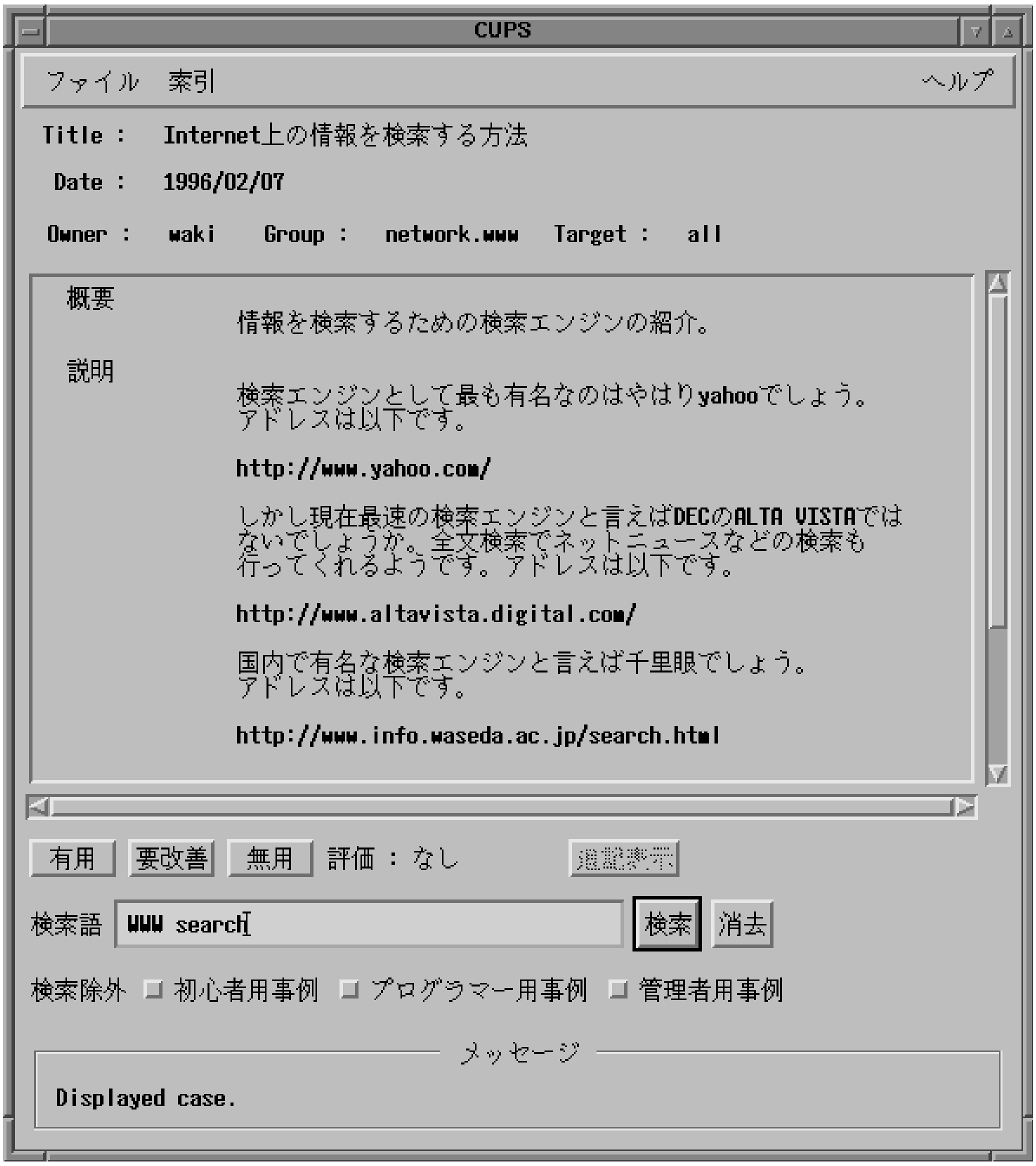


Internet上の情報を検索する方法
JavaScriptでNetscapeのステータス行を掲示板に
SSI、CGIを用いる場合のセキュリティー上の注意点（必読）
基本的なHTML文書
最も簡単なホームページ作成方法
英和・和英の辞書をkenjiroを用いて利用する
anonymous-ftpを用いてソフトを入手する方法

検索語 WWW search

Searched.

図 4: Motif 版 GUI 事例検索



Title : Internet上の情報を検索する方法

Date : 1996/02/07

Owner : waki Group : network.www Target : all

概要

情報を検索するための検索エンジンの紹介。

説明

検索エンジンとして最も有名なのはやはりyahooでしょう。
アドレスは以下です。

http://www.yahoo.com/

しかし現在最速の検索エンジンと言えばDECのALTA VISTAでは
ないでしょうか。全文検索でネットニュースなどの検索も
行ってくれるようです。アドレスは以下です。

http://www.altavista.digital.com/

国内で有名な検索エンジンと言えば千里眼でしょう。
アドレスは以下です。

http://www.info.waseda.ac.jp/search.html

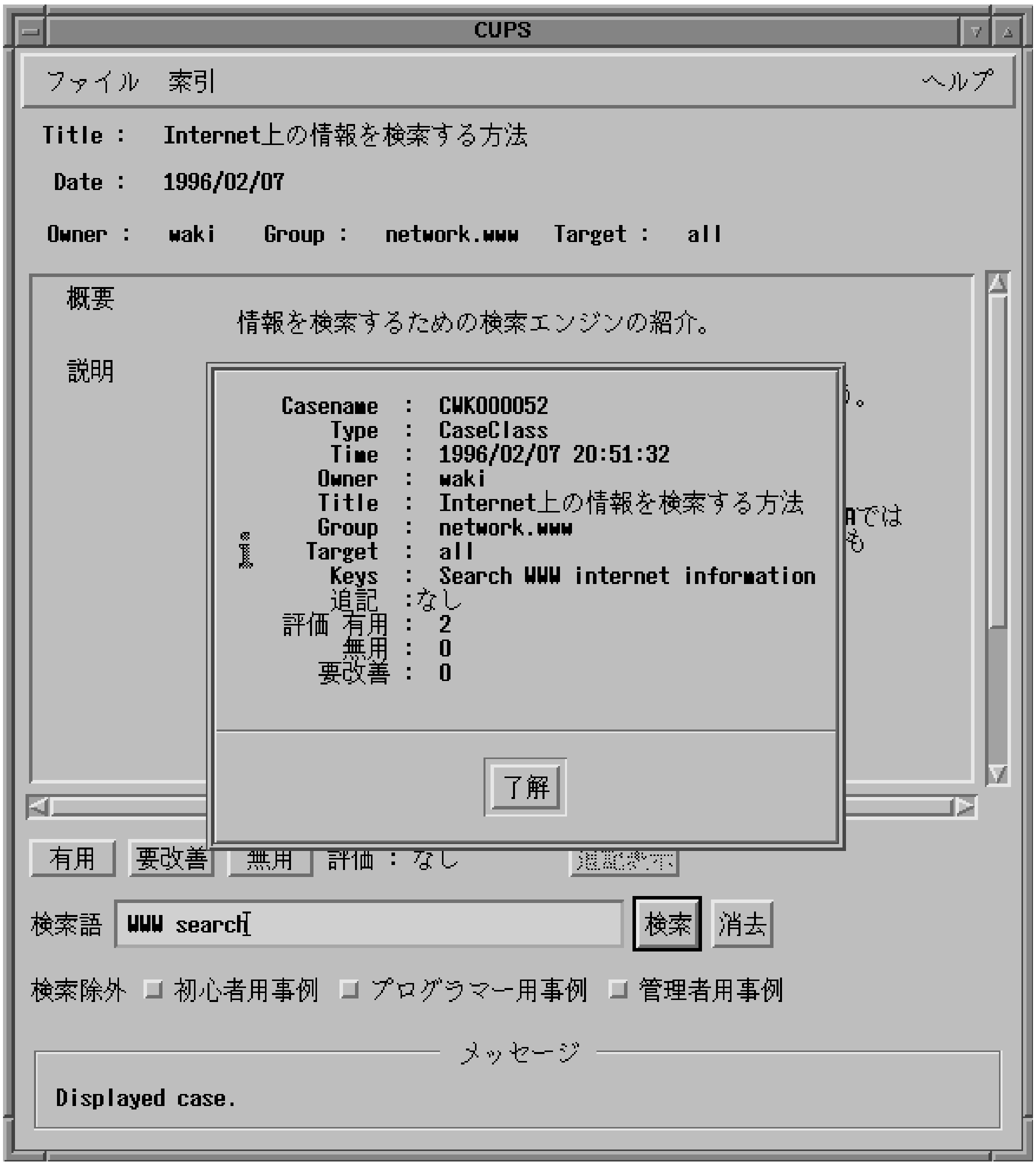
評価 : なし

検索語 WWW search

検索除外 □ 初心者用事例 □ プログラマー用事例 □ 管理者用事例

Displayed case.

図 5: Motif 版 GUI 事例表示



Title : Internet上の情報を検索する方法

Date : 1996/02/07

Owner : waki Group : network.www Target : all

概要 情報を検索するための検索エンジンの紹介。

説明

```
 Casename  :  CWK000052
     Type  :  CaseClass
     Time  :  1996/02/07 20:51:32
    Owner  :  waki
    Title  :  Internet上の情報を検索する方法
    Group  :  network.www
   Target  :  all
     Keys  :  Search WWW internet information
     追記  :なし
評価 有用  :  2
     無用  :  0
   要改善  :  0
```

評価 : なし

検索語 WWW search

検索除外 □ 初心者用事例 □ プログラマー用事例 □ 管理者用事例

Displayed case.

図 6: Motif 版 GUI 詳細情報

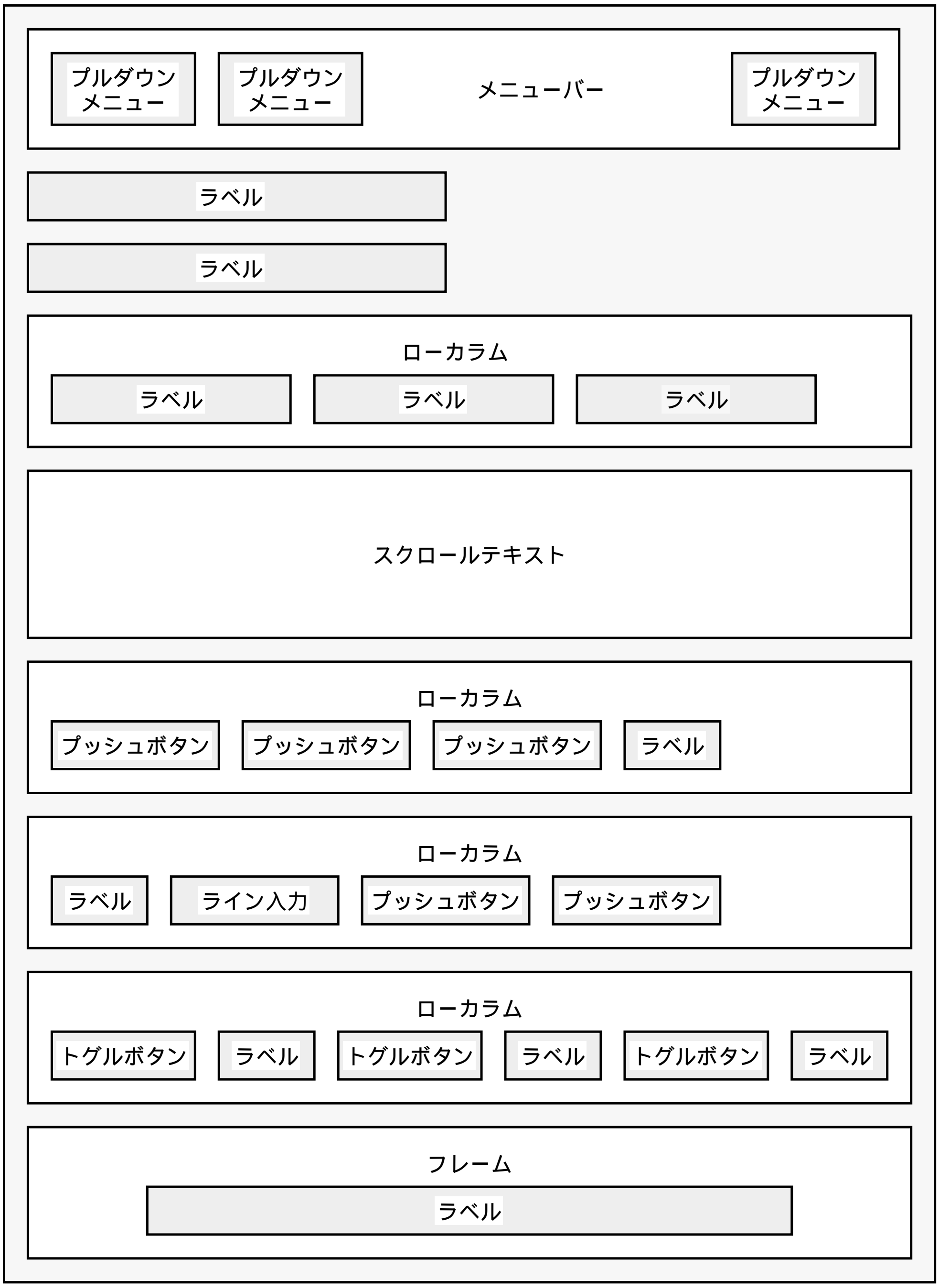

プルダウン
メニュー
プルダウン
メニュー
メニューバー
プルダウン
メニュー
ラベル
ラベル
ローカラム
ラベル
ラベル
ラベル
スクロールテキスト
ローカラム
プッシュボタン
プッシュボタン
プッシュボタン
ラベル
ローカラム
ラベル
ライン入力
プッシュボタン
プッシュボタン
ローカラム
トグルボタン
ラベル
トグルボタン
ラベル
トグルボタン
ラベル
フレーム
ラベル

図 7: GUI の構成

2.1.5 メール読み込み部

電子メールとして送られてきた事例を解釈し、統括部を介して事例ベースに事例を登録する役割を担う。CUPSに送られた事例は自動的にこのプログラムに渡されるよう「.forward」ファイルを用いて設定されている。

1. 電子メールの解析。
   - 作者の切りだし。
   - 事例の切りだし。
2. 登録者へのメール発信。
   - 事例解析エラーの通知。
   - 事例を登録したことの通知。
3. 事例をファイルに保存。
4. 事例ベースへの事例登録。

ここで事例をファイルに保存しているのは、サーバがダウンしていた等で事例ベースへの事例登録が失敗したときに事例が消滅するのを防ぐためである。

2.2 システム実現

統括部、メール読み込み部、事例ベースを C++を用いて作成した。クラス構成を図 8,9,10,11に示す。

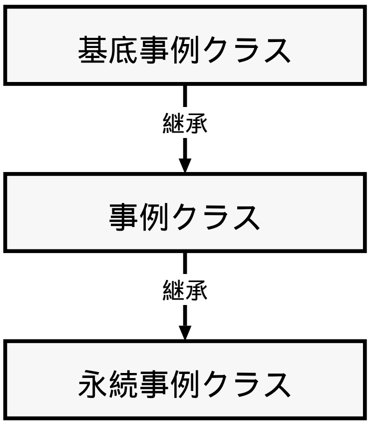


図 8: 事例のクラス構成

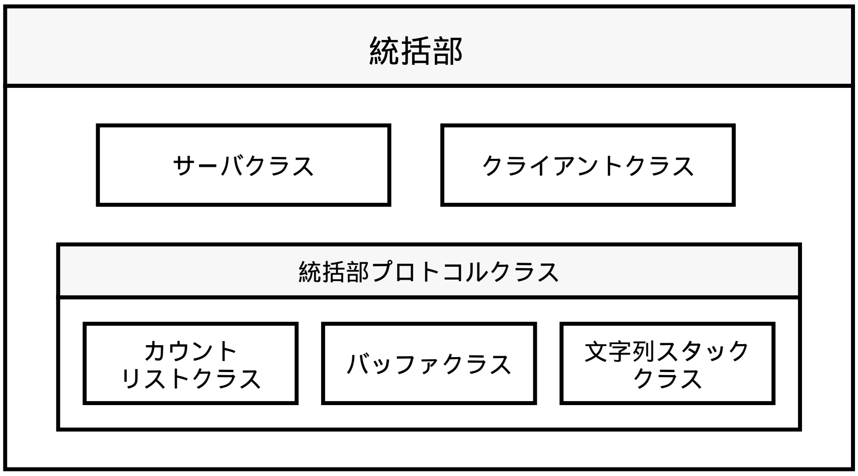


図 9: 統括部のクラス構成

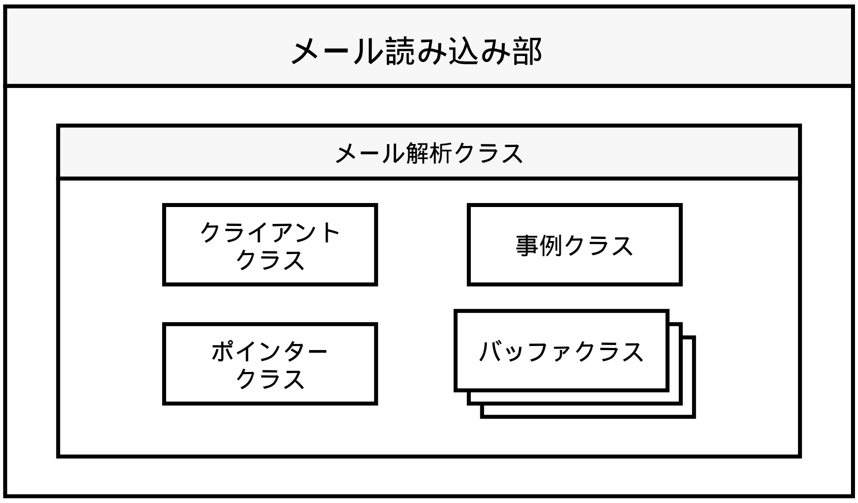


図 10: メール読み込み部のクラス構成

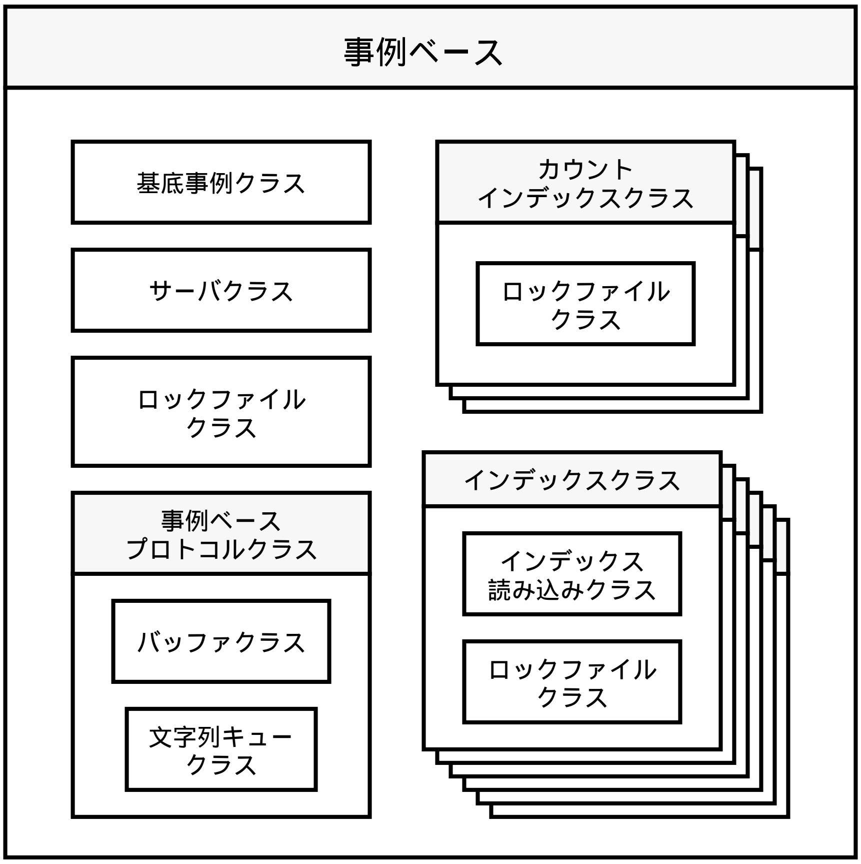


図 11: 事例ベースのクラス構成

以下、CUPS の作成に用いられた個々のクラスの概要を述べる。なお、クラスの定義に関しては付録を参照。

表 1: 「事例」クラス一覧

クラス名	概要
基底事例クラス	事例の基底クラス。CUPS で用いる事例はこのクラスを継承していなければならない。
事例クラス	本研究で収集した事例のクラス。詳細は事例の構成要素の章を参照。
永続事例クラス	事例クラスを永続クラス化したもの。このクラスの事例を変更すると自動的にその実体である事例ファイルが更新され、プログラムの終了等でデータが消滅しない。

表 2: 「ファイル処理」クラス一覧

クラス名	概要
インデックスクラス	事例ベースの逆引きインデックスを作成するためのクラスで、事例のリンク関係もこのクラスで実現されている。
ポインタークラス	文字列を入力するとその文字列が何度入力されたかを返すクラス。主に事例ファイル名の作成に用いられている。
カウントインデックスクラス	「有用」「無用」「要改善」をカウントするために作られたクラス。カウントに従ってソートされる。
ロックファイルクラス	排他処理を行うためのロックファイルを扱うクラスで、上記の 3 つのクラスなど多くのクラスのメンバークラスとなっている。

表 3: 「ネットワーク」クラス一覧

クラス名	概要
サーバクラス	サーバとして、ソケットを用いた通信を行うための手続きをクラス化したもの。
クライアントクラス	クライアントとして、ソケットを用いた通信を行うための手続きをクラス化したもの。
ソケットストリームクラス	Gnanasekaran Swaminathan 氏の作成されたクラスで、ソケットを用いた通信を iostream として実現している。

表 4: 「Flex」クラス一覧

クラス名	概要
統括部プロトコルクラス	統括部のプロトコルを記述。
事例ベースプロトコルクラス	事例ベースのプロトコルを記述。
事例読み込みクラス	事例ベースが事例を読み込むのに使用。
メール解析クラス	メール読み込み部がメールを解析するのに使用。
CGI インターフェイスクラス	WWW 版 GUI が用いる CGI インターフェイス
統括部接続クラス	WWW 版 GUI が用いる統括部との通信用クラスで、事例の検索、読み込みが出来る。
インデックス読み込みクラス	インデックスクラスのインデックスファイルの読み込みに用いられる。

表 5: 「文字列処理」クラス一覧

クラス名	概要
バッファクラス	バッファとして用いるために作成されたクラス。その大きさを自動的に変化させながら文字列を格納する。文字列の追加や、特定文字までの読み出し、char ポインタへの変換などが可能。
カウントリストクラス	統括部が事例検索時に関連事例リストを作成するのに用る。アイテムと数字を渡すと、その数字の大きさ順に並べられたリストが作成される。すでに入力されているアイテムが渡された場合は前の値に新しい数字が加算される。
文字列スタッククラス	文字列を格納し、取り出すクラス。入出力は FILO。
文字列キュークラス	文字列を格納し、取り出すクラス。入出力は FIFO。
シングルリンククラス	１つのリンクでつながれた簡単なリスト。

表 6: 「その他」クラス一覧

クラス名	概要
事例ベースクラス	事例ベース本体。
時間クラス	時間を扱うクラス。
フォーククラス	Gnanasekaran Swaminathan 氏の作成されたクラスでフォークを行う。このクラスを用いてフォークすると親プロセスの終了に伴って子プロセスを自動的に終了させることが出来る。

3　事例ベースの作成

3.1　事例の構成

個々の事例は以下のような項目で構成される。また、実際の事例の例を図 12に示す。

タイトル

事例のタイトル。主に事例の選択に用いられるので、1行に納まり、事例の内容を的確に表すもの。

分類

事例の分類を指定する項目。分類は表 7に従って行われる。

対象

誰を対象とした事例であるかを記述。上級者に初心者の事例は不用であり、プログラムを組まない人にプログラムの事例は不用である。そこでこの情報を用いて利用者に不用な事例を提示するのを防ぐ。実際には表 8のような4つの対象を設定した。

索引語

事例検索に用いられるキーワード列。

事例内容

事例のメイン部分で、以下のような項目に分けて記述される。

概要

その事例を読めば『何が分かるのか？』『何が出来るのか？』と言うことについて数行で記述したもの。事例の利用者はここを読み、本当にこの事例が自分にとって有用なものであるのかを判断する。

説明

実際の説明を記述。

例

独立性が高い例を記述。ただし、説明のなかに書いたほうが分かりやすいような短い例は『説明』に記述。

参照

事例の作者はどうやってこのことを学んだのか、より詳しいことを知るにはどうすれば良いのかについて記述される。この項目によって、その事例が直接的に利用者の問題解決に役立たない場合にも問題解決の糸口を提供できる。

付録

参考程度の例、あるいは非常に長い例を記述。

作成年月日

事例が作成された日時。システムが自動的に付加する。

作者

事例の作成者のユーザー名。環境変数 USER あるいはメールの From 行を用いてシステムが自動的に付加する。

表 7: 事例構成要素「分類」の選択肢一覧

分類名	概要
x-window	X-window 一般
x-window.terminal	X-window のターミナル (xterm,kterm)
x-window.x-tools	X-window のツール
editor	エディター般
editor.emacs	エディタ emacs
editor.mule	エディタ mule
editor.vi	エディタ vi
text	文書一般
text.tex	文書 TeX
text.latex	文書 LaTeX
text.info	文書 Info
programming	言語一般
programming.c	言語 C
programming.c++	言語 C++
programming.perl	言語 perl
shell	シェル
shell.csh	シェル csh
shell.tcsh	シェル tcsh
machine	マシン一般
machine.indy	マシン INDY
machine.ap1000	マシン AP1000
machine.ksr	マシン KSR
network	ネットワーク一般
network.mail	ネットワーク メール
network.news	ネットワーク ニュース
network.ftp	ネットワーク ftp
network.telnet	ネットワーク telnet
network.www	ネットワーク WWW
tools	ツール一般
tools.cups	ツール CUPS
tools.man	ツール man
tools.filter	ツール filter
other	その他
other.device	その他 デバイス

表 8: 事例構成要素「対象」の選択肢一覧

対象名	概要
beginner	初心者専用の事例
administrator	管理者専用の事例
programmer	プログラマー専用の事例
all	上記に該当しない事例

```
タイトル      LaTeX で「bar.sty」を用いて棒グラフを描く方法。
作成年月日    1996/02/12 18:56:15
作者          waki
分類          text.latex
対象          all
索引語        bar bar-chart chart graph style-file latex HOWTO
本文
  概要
      LaTeX で「bar.sty」スタイルファイルを用いるための設定方法。
       「bar.sty」を用いた簡単な棒グラフの例。

  説明
      LaTeX で棒グラフを描く方法として「bar.sty」があります。
      これを利用する場合は、まず以下のようにして「bar.sty」を
      読み込んでください。

      \documentstyle[12pt,a4j,bar]{jarticle}

      次に棒グラフを描きたい所で、以下のような記述をすると
      棒グラフが書けます。

      \begin{barenv}
      \setwidth{50}             % 横幅の設定
      %\setdepth{10}            % 奥行の設定
      \sethspace{.2}            % グラフ間隔の設定
      \setstretch{3}            % 縦軸のスケールを設定
      \setnumberpos{up}         % 値を表示する場所の決定 (他に inside が設定可)
      %\setxaxis{1}{4}{1}       % 目盛の設定
      \setyaxis{0}{60}{10}      %   順に初期値、終了値、ステップ
      \bar{ 5}{1}[初心者]       %
      \bar{ 6}{1}[プログラマ]   % 棒グラフの設定
      \bar{ 1}{1}[管理者]       %   順に、棒グラフの長さ、棒のタイプ、ラベル
      \bar{50}{1}[その他]       %
      \end{barenv}

  参照
      bargraph.tex と barkom.tex より、この知識を得ました。どちらも、
      bargraph.tex、barkom.lzh を archie 等で検索すれば見つけられます。
```

図 12: 事例のサンプル

3.2 事例の収集方法

本システムでは上級者ばかりではなく、初心者から上級者までのすべての利用者から事例（知識）を収集する。これにより、多くの事例を広く集めることが可能となり、またシステムの最新の物事への対応もより早くなる。これは進歩の激しいUNIXの世界において、非常に重要なことである。しかし、このように良いことが多くある反面、弱点として事例の管理が困難となる点があげられる。

表 9: 事例収集方法の比較

事例の収集対象	上級者のみ	初心者 ~ 上級者の全て
収集される事例量	少ない	多い
収集される事例の範囲	狭い	広い
最新の物事への対応	早い	より早い
事例の管理	比較的簡単	困難

具体的な事例収集は2つの方法で行われる。1つは電子メールを用いる方法で、もう1つはGUI部を介して行う方法である。これはそれぞれの方法に一長一短があるためで、電子メールを用いる方法は使い慣れたエディターを用いて事例を作成でき、またどこからでも事例を登録できるという長所があるが、すでに登録した事例に対しての処理が苦手である。逆にGUIを用いる方法ではすでに登録されている事例に対しての処理は容易だが、GUIを起動できる環境に制限があり、またGUI部のエディターの機能にも制限がある。

そのため、主に新規の事例は電子メールを用いる方法で作成し、その修正、削除、追記はGUI部を用いる方法で行う。

3.3 事例の管理方法

事例を一般ユーザーから収集することは、事例の管理を困難なものとする。具体的には、以下のような管理上の問題点が考えられる。

1. 事例の書き方を統一されたものとすることが困難。
2. 内容の適切でない事例の発生。
3. 上級ユーザーの作成した事例が初級ユーザーには理解できないかもしれない。
4. 大量の事例を管理しなければならない。

３番は一般のエキスパートシステムでも生じる問題である。また、４番は「大勢の人」から事例を集める最大の利点の１つである大量に事例が集められことよるものであり、避けられない問題点である。

これらの問題について、「一般ユーザーから収集した事例は、一般ユーザが中心となって管理すべきである」という視点で解決を計った。

１番の問題点より索引語の統一が困難となり、事例検索に支障を来たす問題点と４番の問題点に関連して、多くの事例から目的事例を探すこととなるためにやはり事例検索が困難となるこの 2 つの問題を解決するため、ファジィ検索を用いることを検討しているがこれについては次章で述べる。

3.3.1 「有用」「無用」「要改善」マーク

「無用」、「有用」、「要改善」という の 3 つのマークを導入することにした。つまり、利用者が事例を参照したときに、その事例に対して以下のような評価を与える。

「有用」
役に立った、あるいは役に立つ有用な事例であるとの評価。

「無用」
必要ない事例であるとの評価。

「要改善」
有用だとは思うが改善の必要があると言う評価。

これらの評価は個々の事例の作成者に伝えられ、その事例の作成者はそれらの情報を参考に自分の作成した事例を修正あるいは削除し、事例を管理する。

事例の作者は自分の作成した事例を読み返し、「間違ってないか」、「不用なものとなっていないか」、「アップデートする必要は無いか」などをちょくちょく確認することは時間的に不可能である。そのため自分の作成した事例の問題点にはなかなか気付けない。しかし、この方法を用いると実際に事例を利用した者がその場でその時に問題点を指摘するので、非常に小さなコストで、問題のある事例を発見できる。

3.3.2 事例への追加記入

上級ユーザーの作成した事例が初級ユーザーには理解できないかもしれないという問題の解決や、内容の適切でない事例の修正のため、事例に対して事例を追加記入出来るようにした。

例えば図13のように上級者にしか分からない事例があった場合、この事例を参照した中級者が時間をかけて理解し、この事例に新たな事例を追加記入する。また、この追加記入された事例が中級者以上にしか理解出来ないものであった場合、この事例を参照した初心者が時間をかけて理解し、さらに事例を追加記入する。

こうすることで、この事例は誰にでも理解できる事例へと成長する。上級者は最初の事例を見て理解するし、その事例で分からない中級者は、その追記事例で理解する。その追記事例でも理解出来ない初心者はさらにその追記事例で理解する。

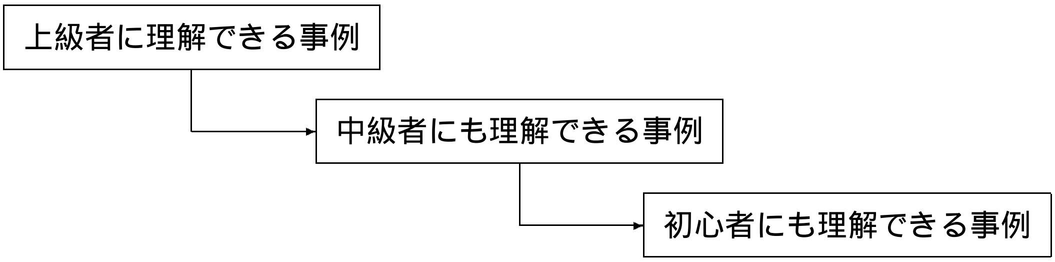


図 13: 事例の追加記入例 1

また、図14のように、不適切な内容を含む事例を発見した利用者は、その不適切な部分を補うような事例を追加記入する。こうすることで、この事例はより完成度の高いものに成長する。

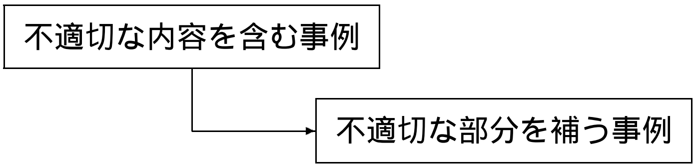


図 14: 事例の追加記入例 2

3.3.3 管理者による管理

出来る限り事例の管理は個々の事例の作成者にまかせたい。しかし、現実問題としては事例の作成者が卒業した等、どうしても管理者が事例を管理しなければならないことも多い。

そこで、以下の情報をソートして管理者に提示し、事例管理の助けとする。

「有用」とされた数
これが指定されているものは、一見、不必要であっても人によっては (ユーザーレベルによっては) 有用なものであるかもしれないので、気を付けて取り扱わねばならない。

「無用」とされた数
「無用」マークが付いているものは、不必要な事例である可能性が高い事例である。

「要改善」とされた数
「要改善」マークが付いているものは、改善する必要の高い事例である。

検索された回数
検索回数が極端に多い、あるいは少ないものは、事例の索引語指定が不適切である可能性が高い。

参照された回数
参照回数が少ない事例は、タイトルの出来が悪いか、不必要な事例である可能性が高い。

追記の回数
あまりに追記が多いものは 1 つにまとめる必要があるかもしれない。

作成年月日
作成年月日が古いものは、現在の状況に合わず、修正が必要、あるいは無用となっている可能性が高い。

3.4 事例の一覧

現在までに６６個の事例が収集された。以下にそのタイトルを分類別に示す。

x-window

BKC から研究室の X 関連のツールを利用する方法
ウインドウの大きさをワンタッチ変更

x-window.terminal

peacock から他のマシンのターミナルを開いたときの漢字表示

editor.emacs

emacs のオートセーブの間隔を調整する
emacs で編集中の行番号を表示させる
emacs で圧縮ファイルを直接編集する
emacs でファイルの最後に空行が入るのを防ぐ

editor.mule

タイプ練習 「ｔｒｒ」の使用、設定について
mule で漢字を入力するには。
egg による入力で句読点を 「,」と 「.」にする
egg のローマ字変換テーブルを変更する

text.latex

C や C++のソースファイルを LaTeX の形式に変換する
LaTeX で自由な段組みを実現する方法
LATEX における minipage 環境
LATEX における箇条書
LATEX におけるセンタリング、右寄せ、左寄せ
LATEX におけるハイフネーションの設定
LATEX における空白制御
LATEX における改行、段落、改ページ
LATEX におけるページレイアウトパラメータの変更
LATEX における目次
LATEX におけるあらまし
LATEX におけるページ番号出力形式設定
LATEX におけるヘッダの設定

LATEX におけるページのスタイル
LATEX の動作モード
LATEX で使用する特別な文字
LATEX で使用する長さの単位
LATEX における文字飾り
LATEX とは
LATEX における文章の表題
LATEX におけるタイトルの作成
LATEX における文章のスタイルオプション
LATEX における文章のスタイル
LaTeX での自作コマンド・マクロの作成
Mac(canvas) で作った絵を LaTeX にはる
LaTeX の 2 ページ分を 1 ページにする方法について
LaTeX で「bar.sty」を用いて棒グラフをかく。

shell.tcsh

tcsh のファイル名補完機能の設定
コマンドのスペルを修正する

programming

Motif プログラミングの全体像
GNU デバッガに準じた、見た目にわかり易いデバッガ

programming.c

Motif を用いた C プログラムのコンパイル方法

machine.ap1000

AP1000 へのアクセス方法とコンパイル、実行方法

machine.ksr

KSR 上での (パラレルリージョン)
KSR の使用方法

network

mesg を用いて talk を禁止する方法
Login している他人とネットワークを介して talk で会話する方法

network.mail

`.forward`を用いたメールの自動転送について

network.ftp

`anonymous-ftp`を用いてソフトを入手する方法
高機能な`ftp`である`ncftp`の紹介

network.www

`JavaScript`で`Netscape`のステータス行を掲示板に
`Internet`上の情報を検索する方法
`SSI`、`CGI`を用いる場合のセキュリティー上の注意点（必読）
基本的な`HTML`文書
最も簡単なホームページ作成方法

tools

英和・和英の辞書を`kenjiro`を用いて利用する
`top`を用いてプロセスの状態を監視する
`cron`を用いて定期的にプログラムを起動する方法
`ls`の使い方
ファイルの漢字コードを変換する
ファイルの漢字コードを調べる

tools.cups

`CUPS`の事例登録を`Mule`から行う為の設定について

other

マシンにログインする時, キーが打ち込めない時の対策
`SPARC`系の新規ユーザ登録の方法

other.device

プリンターを制御する方法

4　事例の検索

4.1　ファジィ検索語

検索語に重みを導入する。ファジィ理論の考え方では個々の検索語に[0,1]の重みを付けるのが自然であると思われるが、その方法では記述が複雑で利用者が容易に用いれるとは言えない。そこで、次のような特定の記号を検索語の頭に付加して重みを表現する。

「+ 」　その検索語を含む事例の重みを増やす。
「− 」　その検索語を含む事例の重みを減らす。

また、これらの記号を「+ + 」などのように重ねて用いてその効果を高める事も可能とする。

「+ + 」　その検索語を含む事例の重みを大きく増やす。
「− − 」　その検索語を含む事例の重みを大きく減らす。

事例検索において、これらは次のような場合に役に立つ。例えば「LaTeXを用いていて、表を書きたい」場合は

+LaTeX　表

と、検索語を指定することによりLaTeXに関係する事例の重みを上げ、これが相対的にLaTeXに関係ない事例の重みを下げることとなる。よってLaTeXに関係ある表に関する事例を検索することが出来る。逆に「表を書きたいのだが、出来ればLaTeXで書きたい」というのであれば

LaTeX　+表

と、検索語を指定すれば良いのである。また、「LaTeXを用いていて表を書きたいのだが、tabular環境を用いる方法ではうまくいかなかった」と言うのであれば

+LaTeX　表　−**tabular**

と、検索語を指定することにより、tabular環境を用いて表を書く方法について書かれた事例の重みを下げ、それ以外の方法で表を書く事例を検索できる。

このように利用者は「+ 」「− 」の記号を用いて検索語に重みをつけ、自分の要求をより正確に表現する。

4.2 ファジィ・シソーラス

事例に登録された「索引語」と、事例の利用者の入力する「検索語」の間には、この2人の間のUNIXに関する知識量の違いなどにより溝が生じる。この溝を埋めるため、「検索語」を、重みを持った関連語リストに変換してから「索引語」と照合する。具体的なその様子を図15に示す。

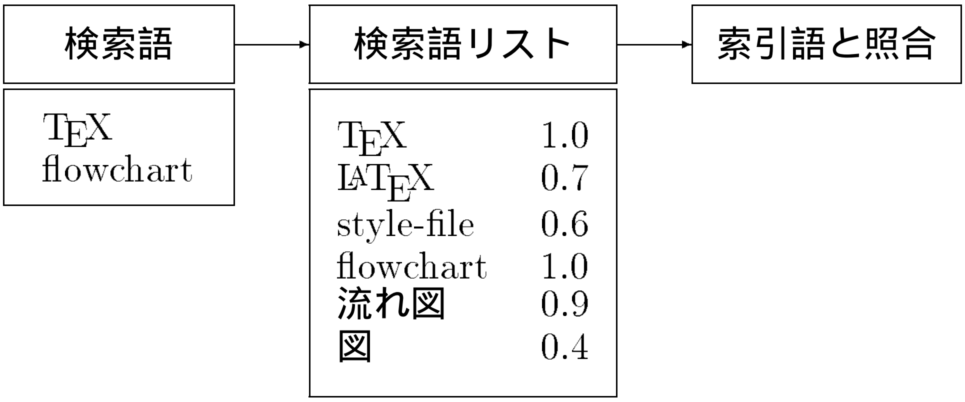


図15: ファジィ検索の流れ

この、「検索語」を、重みを持った関連語リストに変換するという作業を行うためにはファジィ・シソーラスと呼ばれる、重みをもった類義語辞書のようなものが必要となる。

本システムでは、このファジィ・シソーラスを直接作成するのではなくUNIXに関する意味ネットワークをまず形成し、それを変換して作成することとした。その特徴としては以下のようなものが挙げられる。

- 長所
  - 比較的少ない情報で、重みを持ったシソーラスを作成できる。
  - 関係の種類に与える重みを変化させたり、関係の追跡レベルを変えたりして、何通りかのシソーラスを作成し、検索の程度を変化させられる。
  - CAIシステムにとっても意味ネットワークは必要なものであり、情報を共用できる。
- 短所
  - 個々の用語間の関係に対して重みを直接設定する訳ではないので、シソーラスの重みを微調整することが困難。

4.3　意味ネットワーク

4.3.1　意味ネットワークの構成要素

意味ネットワークの、個々のノードの構成要素を以下に示す。

- 用語名
- 属性
- 別名
- 説明
- 識別子
- 記述年月日
- リンク
  - is-a 関係の親
  - is-a 関係の子
  - part-of 関係の親
  - part-of 関係の子
  - 類義語
  - 関連語
  - 動作

このようなノードをリンクを用いて接続し、意味ネットワークを形成する。

4.3.2 関連語リストへの変換

本システムではファジィシソーラスの代わりに意味ネットワークを用いて、検索語を重みを持った関連語リストの変換する。具体的には、図16のようなLaTeXに関する意味ネットワークが形成されていたとき、以下のルールのもと、検索語LaTeXに対して、表10のような関連語リストを得る。

1. 別名に対して-0.1の重みを与える。
2. 類義語に対して-0.2の重みを与える。
3. is-a関係に対して-0.3の重みを与える。
4. part-of関係に対して-0.4の重みを与える。
5. 関連語に対して-0.6の重みを与える。
6. is-a関係のみ１レベル先まで関連語を探す。

表10: 関連語リスト

関連語	関係	重み
LaTeX	検索語	1
ラテフ	別名	1 - 0.1 = 0.9
ラテック	別名	1 - 0.1 = 0.9
jlatex	類義語	1 - 0.2 = 0.8
xdvi	関連語	1 - 0.6 = 0.4
dvi2ps	慣例語	1 - 0.6 = 0.4
epsbox	part-of関係の子	1 - 0.4 = 0.6
TeX	is-a関係の親	1 - 0.3 = 0.7
テフ	is-a関係の親の別名	1 - 0.3 - 0.1 = 0.6
テック	is-a関係の親の別名	1 - 0.3 - 0.1 = 0.6
jtex	is-a関係の親の類義語	1 - 0.3 - 0.2 = 0.5
dviファイル	is-a関係の親のpart-of関係の子	1 - 0.3 - 0.4 = 0.3
styファイル	is-a関係の親のpart-of関係の子	1 - 0.3 - 0.4 = 0.3

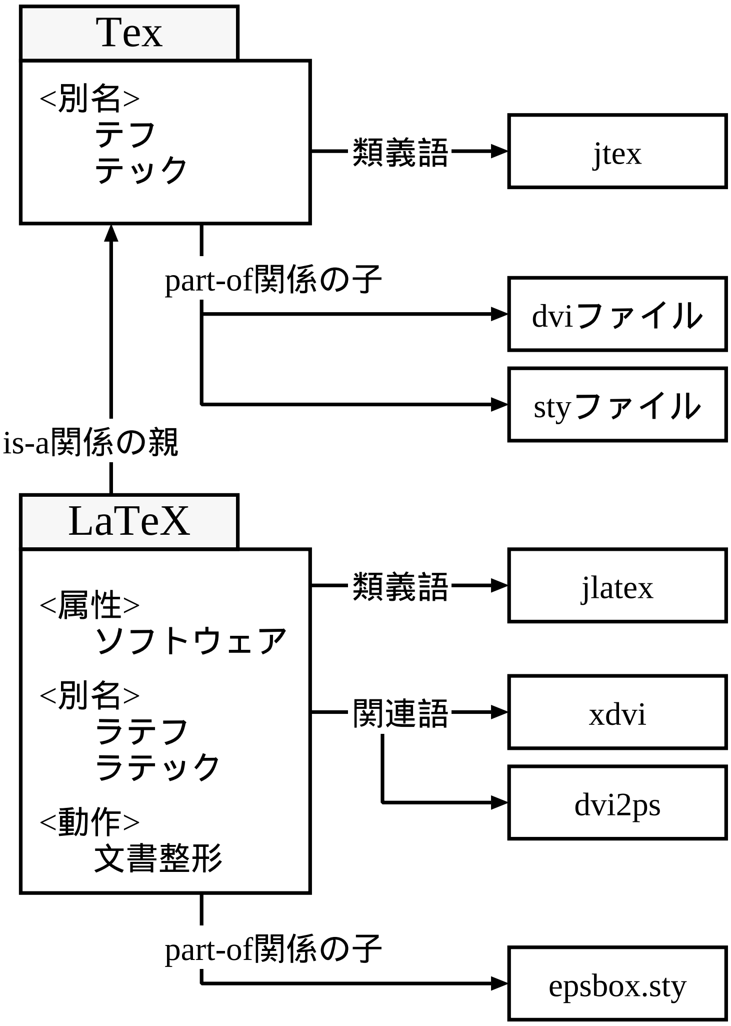


図 16: 意味ネットワーク

5 評価

現在までに収集された事例は６６個である。まだまだ少ないが本システムの事例登録は問題を解決したそのときに行うことが基本であることを考えると仕方の無い面もある。逆にレベルの異なる１０人という多くの人から事例を収集できた点は評価できる。mule での事例登録用に emacs-lisp プログラムを作成し、質問に答えるだけで事例のフレームを作成出来たり、事例の構成要素「分類」と「対象」を選択するだけで入力できるようにしたことが、非常に有効であった。

収集された事例を分類別に集計してみると図17のようになる。text に対して多くの事例が集まっているが、この text に LaTeX が含まれ、この LaTeX に関して三小田が集中的に事例を収集した(2)ためである。また同時に事例収集期間に卒論等で LaTeX が頻繁に利用されていたことにもよる。同様に network が多いのは、最近 WWW などがトレンドであるためであろう。ユーザーの利用頻度が高いものに関して、あるいはトレンドな情報に関して事例が集中する傾向がある。これは、個々のユーザーから自分にとって有用だと思ったものを事例化するためで、大きな利点である。

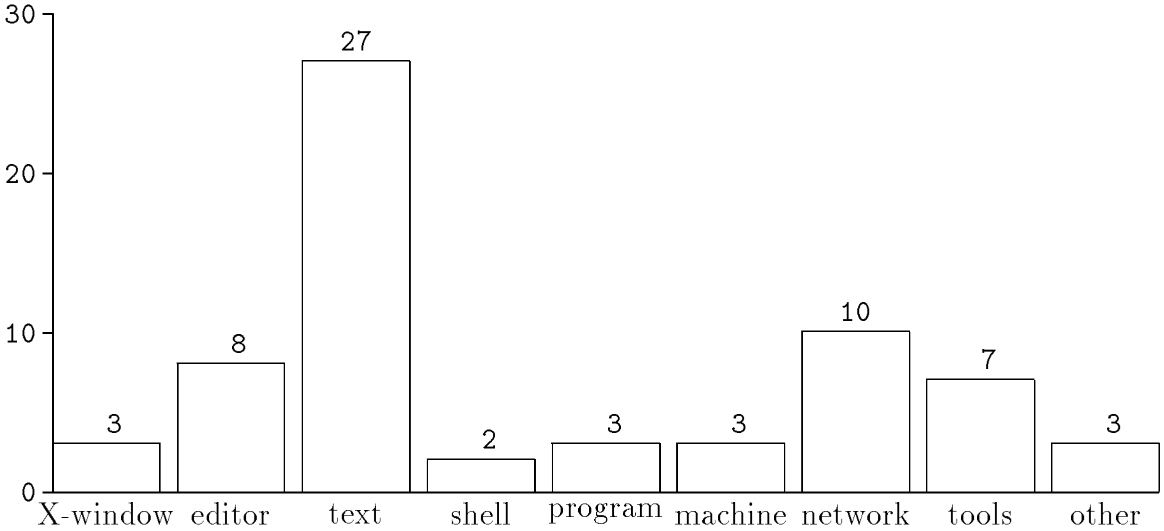


図 17: 分類別事例数

次に、収集された事例をその対象別に集計すると図18のようになる。これから分かるように「その他」が圧倒的に多い。これは予想されていたことではあるが、ここまで集中してしまうと事例検索時に対象を絞る効果が期待できず良くない。もっと対象を増やすなどして、「その他」に集まる割合を減らす必要がある。

本システムでは事例を迅速に収集するため、初級者から上級者まで多くの人から事例を収集した。実際に事例を収集してみるとやはり上級者の方が多くの有用な事例を登録できた。しかし、初級者でも有用な事例を作れている例は多かった。確かに初級者が上級者向けの事例をつくることは困難である。しかし、時として初級者にしか作れない（上級者に

---

(2) 文献 [15] を参照

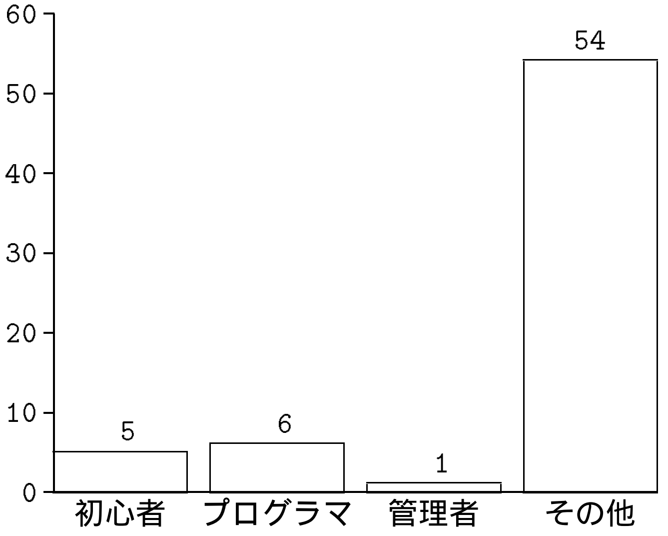


図 18: 対象別事例数

はその必要性が理解できず作ろうとしない）事例がある。初級者もその量と質で上級者には及ばないが、日々さまざまな経験をつみ、新しい知識を得ているのである。そのなかに事例とすべきことは必ずある。様々なレベルの人が事例を作ることで、様々なレベルの人に対応した事例が集められる。これは、このシステムの優れた点だと言える。

しかし、初級者には事例を作りたくてもなかなかその能力の限界によって作れないことがある。つまり、問題を解決出来ない場合や、理解度が浅すぎて事例化できない場合などである。このような場合、そのことを上級者に伝え、事例化してもらえるシステムにすると、より有用なものとなるだろう。

また、利用者から、「事例に画像を取り込みたい」と言う要望があった。テキストだけではなく、様々なバイナリデータを扱えるように事例構成を拡張していく必要がある。

次に事例検索についてだが、現状では重みを付けた複数の検索語での事例検索が可能であるが、意味ネットワーク部が機能していないため、検索語を関連語リストに変換する部分が省略されている。そのため、自分の求める事例を多くの無関係な事例とともに表示させることはそれほど難しくないが、必要な事例を絞り込んで検索することはかなり難しい。特に初心者が事例を検索する場合はなかなか的確な検索語を指定できないので、意味ネットワークの必要性を強く感じた。

また、本システムでは事例を効率的に管理するため、以下のような手法を用いた。

1. 「有用」「無用」「要改善」マーク
2. 事例への追加記入
3. 管理者に様々な事例管理情報を提示。

まだ事例数が多くないので、これらに対して評価を与えるのは時期が早いように思う。しかし、少なくとも現在程度の事例数では、これでは充分に事例を管理出来ている。今後、事例数がもっと多くなってからが楽しみである。

6　おわりに

事例を用いたUNIX問題解決支援システムCUPSを設計した。設計にあたり「いかにして迅速に事例を収集するか」と言う点を重視し、その実現のため上級ユーザーばかりでなく、初心者から上級者のすべてのユーザーから事例を収集することとした。そして、事例が容易に登録出来るシステムとなるよう心掛けた。

また、事例をすべてのレベルのユーザーから集めることよって生じる様々な問題に対処するため、CUPSに以下にような機能を付加した。

- 「有用」「無用」「要改善」マーク
- 事例への追加記入
- 様々な事例管理情報の表示
- ファジィ検索語
- 意味ネットワーク

実際にシステムを作成し（意味ネットワークを除く）、事例を収集した。そして今までに、研究室の１０名の方から、あわせて６６個の事例を登録してもらった。登録された事例を見てみると、事例の作り手と事例の利用者が同じであるので、自然と利用者に必要な事例が収集される傾向があった。

今後に課題としては、意味ネットワークを構築し事例の検索能力を強化すること、バイナリデータを扱えるような事例をサポートすること、事例作成の要望を上級ユーザーへ通知するシステムを付加がすることである。

謝辞

本研究の機会を与えて下さり、貴重な御指導を頂きました山崎勝弘教授に心より深く感謝いたします。

また、本研究室のみなさまには事例の作成に協力いただき時には貴重な意見を聞かせていただきました。その中でも上野和孝氏にはGUI部のプログラムを、三小田降幸氏にはLaTeXに関する多くの事例を作成していただきました。この場を借りて深く感謝いたします。

「C++ socket library Version 1.9」を作成されたGnanasekaran Swaminathan氏に深く感謝します。

参考文献

[1] Kolodner J.: Case-Based Reasoning, Morgan Kaufmann Publishers,1993.

[2] Mark Kriegman,Ralph Barletta: Building a Case-Based Help Desk Application, IEEE EXPERT,Vol.8,No.6,1993,pp.18-26.

[3] 戸沢義夫: 特集 事例ベース推論」にあたって, 人工知能学会誌, Vol.7,No.4,1992,pp.558.

[4] 小林重信: 事例ベース推論の現状と展望, 人工知能学会誌, Vol.7,No.4,1992,pp.559-566.

[5] Kamran Parsayem,Mar Chignell,Setrag Khoshafian,Harry Wong 著, 近谷英昭 訳, 知的データベース ~オブジェクト指向 ·演繹 ·ハイパーメディア~, オウム社,1992.

[6] 上野晴樹, 石塚満: 知識工学講座２ 知識の表現と利用, オウム社,1987.

[7] 竹垣盛一, 石岡卓也: 知的制御システム, 海文堂出版,1990.

[8] Douglas Comer,David Stevens 著, 村井純, 楠本博之 訳, TCP/IP によるネットワーク構築 Vol.III ~クライアント-サーバプログラミングとアプリケーション~, 共立出版,1995.

[9] Kevin Washburn,Jim Evans, 油井尊 訳: TCP/IP バイブル,SOFTBANK BOOKS,1994.

[10] Gnanasekaran Swaminathan: C++ Socket Classes(socket++-1.9.tar.gz),1994.

[11] 日本ファジィ学会: 講座 ファジィ 第９巻 ファジィ ·データベースと情報検索, 日刊工業新聞社,1993.

[12] Marc J.Rockkind 著, 福崎俊博 訳: UNIX システムコール ·プログラミング, アスキー出版,1987.

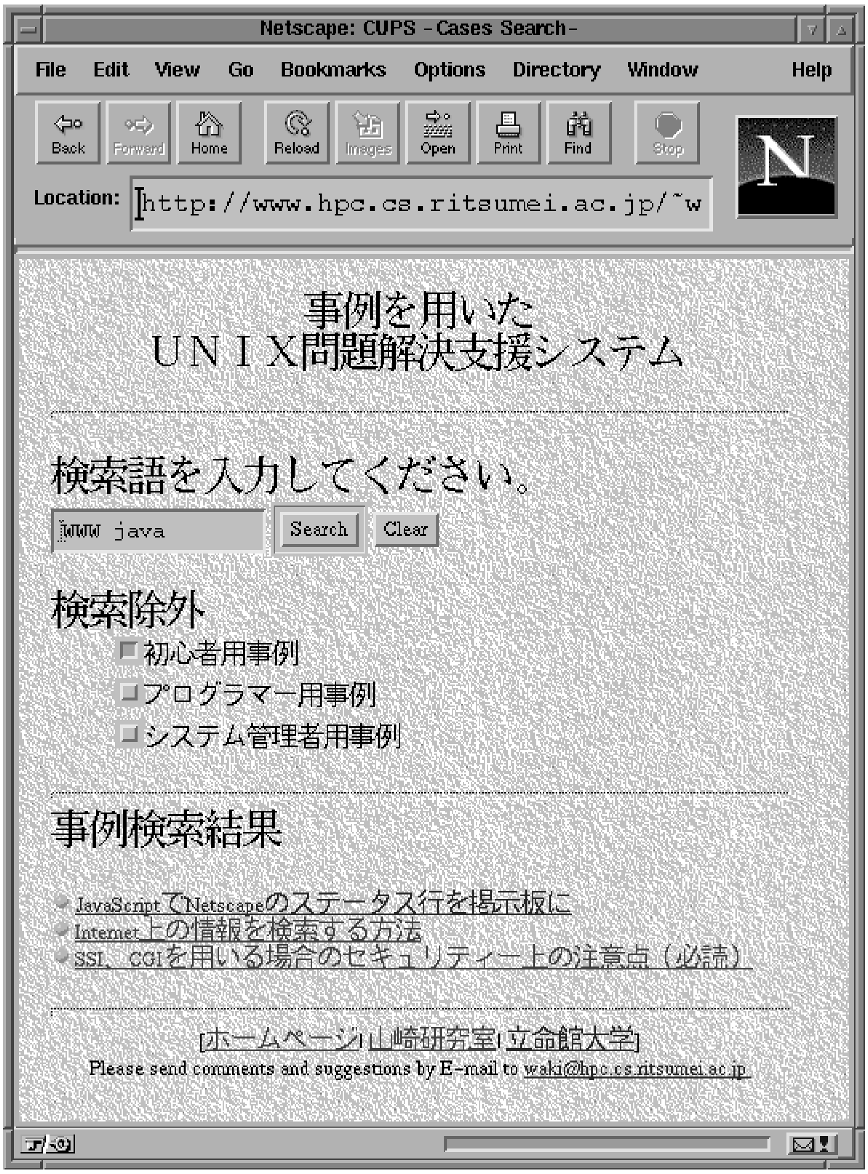
[13] 脇武也, 山崎勝弘: 類似事例を用いた UNIX 問題解決支援システムの設計、情報処理学会第５１回全国大会,7j-5,1995.

[14] 上野和孝: UNIX 問題解決支援システム CUPS のグラフィカルユーザーインターフェイス, 立命館大学理工学部情報工学科卒業論文,1996.

[15] 三小田降幸: UNIX 問題解決支援システム CUPS における事例の作成, 立命館大学理工学部情報工学科卒業論文,1996.

付録

WWW 版 GUI の外形



http://www.hpc.cs.ritsumei.ac.jp/~w

事例を用いた
ＵＮＩＸ問題解決支援システム

検索語を入力してください。

WWW java

検索除外

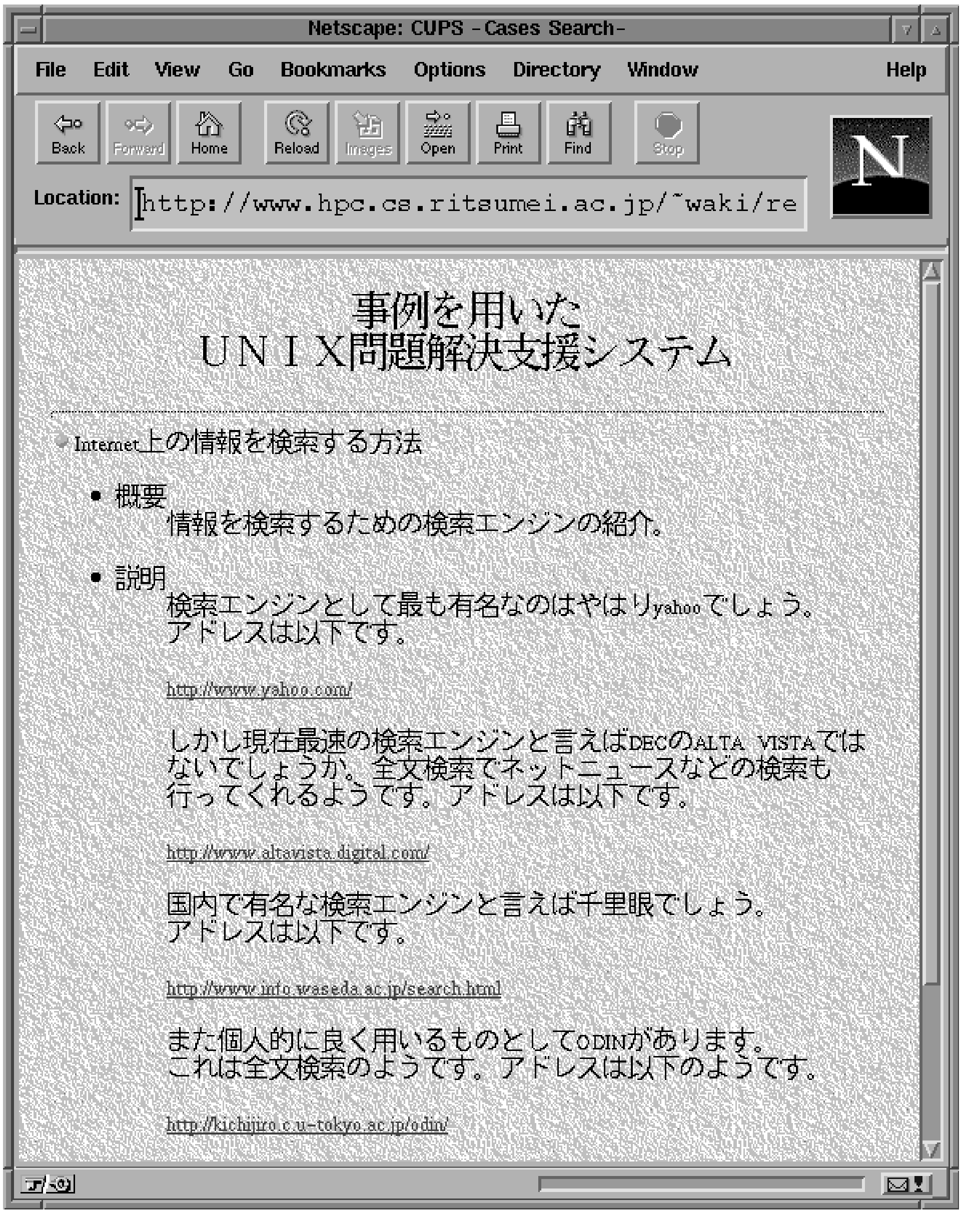
☑ 初心者用事例
☐ プログラマー用事例
☐ システム管理者用事例

事例検索結果

- JavaScriptでNetscapeのステータス行を掲示板に
- Internet上の情報を検索する方法
- SSI、CGIを用いる場合のセキュリティー上の注意点（必読）

[ホームページ| 山崎研究室| 立命館大学]
Please send comments and suggestions by E-mail to waki@hpc.cs.ritsumei.ac.jp

図 19: WWW 版 GUI 事例検索



http://www.hpc.cs.ritsumei.ac.jp/~waki/re

事例を用いた
UNIX問題解決支援システム

Internet上の情報を検索する方法

- 概要
  情報を検索するための検索エンジンの紹介。
- 説明
  検索エンジンとして最も有名なのはやはりyahooでしょう。
  アドレスは以下です。

  http://www.yahoo.com/

  しかし現在最速の検索エンジンと言えばDECのALTA VISTAではないでしょうか。全文検索でネットニュースなどの検索も行ってくれるようです。アドレスは以下です。

  http://www.altavista.digital.com/

  国内で有名な検索エンジンと言えば千里眼でしょう。
  アドレスは以下です。

  http://www.info.waseda.ac.jp/search.html

  また個人的に良く用いるものとしてODINがあります。
  これは全文検索のようです。アドレスは以下のようです。

  http://kichijiro.c.u-tokyo.ac.jp/odin/

図 20: WWW 版 GUI 事例表示

クラスの定義

基底事例クラス

```
class CaseBaseClass
{
public:
  virtual void GetAllCaseData(BufferClass *i_buffer)
    const=0;
  virtual const char *GetTime() const=0;
  virtual const char *GetOwner() const=0;
  virtual const char *GetTitle() const=0;
  virtual const char *GetGroup() const=0;
  virtual const char *GetTarget() const=0;
  virtual const char *GetKeys() const=0;
  virtual const char *GetMain() const=0;
  virtual void PutTime (const char *i_time) =0;
  virtual void PutOwner (const char *i_owner) =0;
  virtual void PutTitle (const char *i_title) =0;
  virtual void PutGroup (const char *i_group) =0;
  virtual void PutTarget(const char *i_target)=0;
  virtual void PutKeys (const char *i_keys) =0;
  virtual void PutMain (const char *i_main) =0;
  virtual const char *GetCasename() const=0;
  virtual void UpdateTime()=0;
  virtual void Update()=0;
  virtual ~CaseBaseClass() {}
};
```

事例クラス

```
class CaseClass : public CaseBaseClass
{
public:
  CaseClass();
  CaseClass(const BufferClass *all_data);
  void GetAllCaseData(BufferClass *i_buffer) const;
  const char *GetTime() const;
  const char *GetOwner() const;
  const char *GetTitle() const;
  const char *GetGroup() const;
  const char *GetTarget() const;
  const char *GetKeys() const;
  const char *GetMain() const;
  void PutTime (const char *i_time);
  void PutOwner (const char *i_owner);
  void PutTitle (const char *i_title);
  void PutGroup (const char *i_group);
  void PutTarget(const char *i_target);
  void PutKeys (const char *i_keys);
  void PutMain (const char *i_main);
  const char *GetCasename() const;
  void UpdateTime();
  void Update();
  ~CaseClass();
protected:
  void StrRenew(char **poi,const char *str);
  FILE *fp;
  TimeClass time;
  char *owner,*title,*group,*target,*keys,*main;
};
```

永続事例クラス

```
class CaseClassP : public CaseClass
{
public:
  CaseClassP(const BufferClass *all_data,
           const char *i_casename=NULL);
  CaseClassP(const char *i_casename);
  void PutTime (const char *i_time);
  void PutOwner(const char *i_owner);
  void PutTitle(const char *i_title);
  void PutGroup(const char *i_group);
  void PutTarget(const char *i_target);
  void PutKeys (const char *i_keys);
  void PutMain (const char *i_main);
  const char *GetCasename() const;
  void UpdateTime();
  ~CaseClassP();
private:
  void SetFileName(const char *i_casename=NULL);
  void SaveData();
  int SaveFlag;
  char *casename;
  char *filename;
};
```

インデックスクラス

```
class IndexClass
{
public:
  IndexClass(const char *i_filename);
  void Put(const char *i_name,const char *i_item);
  void InnerPut(const char *i_name,const char *i_item);
  int Del(const char *i_name,const char *i_item);
  char **Get(const char *i_name);
  char **GetAllItem();
  ~IndexClass();
private:
  struct key_list_struct {
    char *name;
    ListSLClass *items;
    key_list_struct *next;
  } *Top,*Now,*Tail;
  void AddTop(const char *i_name,const char *i_item);
  void AddTail(const char *i_name,const char *i_item);
  void Update();
  void LoadIndex();
  void Save();
  void Clear();
  FlexLexer *lexer;
  time_t time;
  int counter;
  char *filename;
  LockClass *lock;
};
```

ポインタークラス

```
class FilePointerClass
{
public:
  FilePointerClass(const char *i_filename);
  int GetPointer(const char *i_name);
  ~FilePointerClass();
private:
  struct PointerStruct {
    char name[256];
    int num;
```

```
    PointerStruct *next;
  } *Top,*Now,*Tail;
  void Add_Link();
  void Update();
  void LoadPointer();
  void SaveData();
  LockClass *lock;
  time_t time;
  FILE *fp;
  char *filename;
};
```

カウントインデックスクラス

```
class CountIndexClass
{
public:
  CountIndexClass(const char *i_filename);
  void Add(const char *i_item);
  int GetCounter(const char *i_item);
  int Del(const char *i_item);
  char **GetAllItem();
  ~CountIndexClass();
private:
  struct count_index_struct {
    char *item;
    int num;
    count_index_struct *prev,*next;
  } *Top,*Now,*Tail;
  count_index_struct *Find(const char *i_item);
  void AddNewItem(const char *i_item,const int i_num);
  void Clear();
  void Update();
  void LoadIndex();
  void SaveData();
  LockClass *lock;
  time_t time;
  FILE *fp;
  char *filename;
  int counter;
};
```

ロックファイルクラス

```
class LockClass
{
public:
  LockClass(const char *filename,
            int mode=(O_RDWR|O_CREAT));
  time_t GetTime();
  ~LockClass();
private:
  void GetLock();
  int lockfile;
};
```

サーバクラス

```
class ServerSocketClass
{
public:
  ServerSocketClass(int i_port);
  int Accept();
  void Read(char *str);
  void Read(BufferClass *buf_class);
  void Write(const char *str);
  void Write(BufferClass *buf_class);
  int is_eof();
  operator iosockinet& ();
  ~ServerSocketClass();
private:
  sockinetbuf si_buf;
  iosockinet *sstream;
  Fork *fork;
};
```

クライアントクラス

```
class ClientSocketClass
{
public:
  ClientSocketClass(const char *i_hostname,int i_port);
  void Read(char *str);
  void Read(BufferClass *buf_class);
  void Write(const char *str);
  void Write(BufferClass *buf_class);
  int is_eof();
  operator iosockinet& ();
  ~ClientSocketClass() {}
private:
  iosockinet sstream;
};
```

バッファークラス

```
class BufferClass
{
public:
  BufferClass(int num=256);
  char &operator[] (int index);
  void Clear();
  int Gets(char *str,int num=0) const;
  int Puts(const char *str,int num=0);
  int Add(const char *str) { return Puts(str,last_num); }
  int Strlen(int num=0) const;
  int BufferSize() const { return buffer_size; }
  int Last() const { return last_num; }
  operator char* ();
  ~BufferClass();
private:
  struct buffer_struct {
    char *buf;
    buffer_struct *next;
  } *top,*bottom;
  void MakeBuffer() {MakeBuffer( &(bottom→next) );}
  void MakeBuffer(buffer_struct **pre_next);
  int buffer_size,last_num,last_frame;
};
```

カウントリストクラス

```
class CountListClass
{
public:
  CountListClass();
  void Add(const char *i_item,const float i_num);
  int Del(const char *i_item);
  char **GetAllItem();
  ~CountListClass();
private:
  struct count_list_struct {
    char *item;
    float num;
    count_list_struct *prev,*next;
```

```
   } *Top,*Now,*Tail;
   count_list_struct *Find(const char *i_item);
   void AddNewItem(const char *i_item,
                   const float i_num);
   int counter;
};
```

文字列スタッククラス

```
class StrStack
{
public:
   StrStack();
   void Push( const char *item );
   char *Pop();
   GetCounter();
   ~StrStack();
private:
   struct str_stack_struct {
      char *item;
      str_stack_struct *next;
   } *Top;
   int counter;
};
```

文字列キュークラス

```
class StrQueue
{
public:
   StrQueue();
   void Put( const char *item );
   char *Get();
   int GetCounter();
   ~StrQueue();
private:
   struct str_queue_struct {
      char *item;
      str_queue_struct *next,*prev;
   } *Top,*Tail;
   int counter;
};
```

シングルリンククラス

```
class ListSLClass
{
public:
   ListSLClass();
   void Add(const char *i_item );
   void AddTop(const char *i_item);
   void AddTail(const char *i_item);
   int Del(const char *i_item);
   char **GetAllItem();
   int Length();
   ~ListSLClass();
private:
   struct list_struct {
      char *item;
      list_struct *next;
   } *Top,*Now,*Tail;
   int counter;
};
```

事例ベースクラス

```
class CaseServerClass
{
public:
   CaseServerClass();
   void Add( const BufferClass *i_case,
             const char *i_casename=NULL );
   const char *Add2( const BufferClass *i_case,
                     const char *parent_casename );
   void Del( const char *i_casename);
   void Get( const char *i_casename,BufferClass *i_buf);
   void Modify( const char *i_casename,
                const BufferClass *i_case);
   void MakeTitle( const char *i_casename,char *i_buf);
   void TimeSearch ( const char *i_item,
                     BufferClass *i_buf );
   void OwnerSearch ( const char *i_item,
                     BufferClass *i_buf );
   void GroupSearch ( const char *i_item,
                     BufferClass *i_buf );
   void TargetSearch( const char *i_item,
                     BufferClass *i_buf );
   void KeysSearch ( const char *i_item,
                     BufferClass *i_buf );
   void LinksSearch ( const char *parent_casename,
                     BufferClass *i_buf );
   int GetYuuyou( const char *casename );
   int GetMuyou ( const char *casename );
   int GetKaizen( const char *casename );
   void MarkYuuyou( const char *casename );
   void MarkMuyou ( const char *casename );
   void MarkKaizen( const char *casename );
   void TimeIndex ( BufferClass *i_buf );
   void OwnerIndex ( BufferClass *i_buf );
   void GroupIndex ( BufferClass *i_buf );
   void TargetIndex( BufferClass *i_buf );
   void KeysIndex ( BufferClass *i_buf );
   void YuuyouIndex( BufferClass *i_buf );
   void MuyouIndex ( BufferClass *i_buf );
   void KaizenIndex( BufferClass *i_buf );
   ~CaseServerClass();
private:
   void Search( IndexClass *i_index,
                constchar *i_item,BufferClass *i_buf );
   void GetIndex( IndexClass *i_index,BufferClass *i_buf );
   void GetInfo( CountIndexClass *i_info,
                 BufferClass *i_buf );
   LockClass *LockCase( const char *i_casename);
   CaseBaseClass *case_poi;
   LockClass *lock;
   FlexLexer *lexer;
   IndexClass
      time_idx,owner_idx,group_idx,target_idx,keys_idx;
   CountIndexClass yuuyou,muyou,kaizen;
   IndexClass links;
};
```

時間クラス

```
class TimeClass
{
public:
   TimeClass();
   void Set();
   void Put(const char *i_time);
   const char *Get() const;
private:
   char time_data[TIME_LENGTH];
};
```

事例作成ガイドライン

タイトル

・事例の内容、摘要範囲を的確に示すもの。
・一行に収まる範囲で長いもの。

グループ

以下の中から１つを選んでください。

```
"x-window"
"x-window.terminal"
"x-window.x-tools"
"editor"
"editor.emacs"
"editor.mule"
"editor.vi"
"text"
"text.tex"
"text.latex"
"text.info"
"programming"
"programming.c"
"programming.c++"
"programming.perl"
"shell"
"shell.csh"
"shell.tcsh"
"machine"
"machine.indy"
"machine.ap1000"
"machine.ksr"
"network"
"network.mail"
"network.news"
"network.ftp"
"network.telnet"
"network.www"
"tools"
"tools.upss"
"tools.man"
"tools.filter"
"other"
"other.device"
```

ターゲット

以下の中から１つ選んでください。

```
=======================================================
beginner        初心者にしか必要ない事例。
```

administrator	管理者にしか必要ない事例。
programmer	プログラマーにしか必要ない事例。
all	以上の該当しないもの。 かなりの事例がここに含まれます。

キー

・4〜8個ぐらいのキーワードの並び。
・日本語、英語の両方を用いて指定する。
・区切りはスペースを用い、二語にわたるものは「-」を用いてつなぐ。
・設定に関するものには、「SETUP」というキーを指定してください。
・方法が書かれているものには「HOWTO」というキーを指定してください。

メイン

メイン部分は以下のような見出しを用いて書いて下さい。

見出し	説明
概要	この事例を読めば「何が分かるのか？」「何が出来るのか？」ということについて数行で書く。 （利用者はここを読み、本当にこの事例が自分にとって有用なものであるのかを判断する。）
説明	実際の説明を書いて下さい。 オプションの説明など、同じようなものが並ぶ場合にはこのように表を用いるようにお願いします。
例	例は説明の中に書いても構いませんが、独立性が高い場合はここに書いてください。
参照	より詳しいことを知るにはどうすれば良いのか、または自分がどうやってこのことを学んだのかについて書いてください。（この情報は非常に重要ですので、出来る限り記入してください。）
付録	例があまり長いと読みにくくなると考えられますのでそのような場合はここに例を載せてください。

その他の注意点

・一つの事例にあまり多くの事柄を書かないください。