PCTE を用いた UNIX コマンドデータベースの作成に関する研究

田中 聡

北陸先端科学技術大学院大学 情報科学研究科

2000 年 2 月 15 日

キーワード： UNIX コマンド, PCTE, データベース, 意味的制約.

1 背景

コンピュータの普及に伴い、ソフトウェアの大きさは爆発的に増加した。巨大なソフトウェアにはコードの断片が、互いに複雑に制約し合い関係している。それゆえ、巨大なソフトウェアを効率的に開発・保守するにはソフトウェアデータベースが必要である。ソフトウェアデータベースの概念が世に出てから長い年月が経つが、未だ実用的なソフトウェアデータベースは開発されていない。それはソフトウェアデータベースの開発が難しいからである。

2 目的

本研究のゴールは、何がソフトウェアデータベースの問題を難しくしているのかを明らかにし、ソフトウェアデータベースの構築方法を確立することである。これまでの研究ではメモや非形式的な仕様書、不完全な設計図などのような非形式的なオブジェクトに焦点が当てられてきた。本研究では形式的なソフトウェアオブジェクトに焦点を当て、規模が小さく最低限の難しさを保持したソフトウェアデータベースを構築することを試みる。本研究ではソフトウェアデータベースの構築実験の実験材料として UNIX コマンド、実験道具として Emeraude PCTE を用いた。PCTE を用いた UNIX コマンドデータベースの作成を行ない、ソフトウェアオブジェクト間の制約・関係を記述することを試みる。具体的には以下の手順で行なう。

1. 対象となる UNIX コマンドを選択する。

2. UNIX コマンドから関係を抽出し、ERA 図で記述する。

3. UNIX コマンドから制約を抽出し、Emeraude PCTE 上でプログラムとして実現する。

本稿の目的は、作成した UNIX コマンドよりソフトウェアデータベースの構築方法の確立に有効な手がかりとなる分類を行なうことである。

3 UNIX コマンドの特徴

- UNIX コマンドは相互に複雑に関係し合っている。
- しかし現在のところ、その情報をユーザに伝える方法は存在しない。

UNIX コマンドの解釈実行プログラムを「シェル」という。シェルはパイプ処理を使って複数の UNIX コマンドをつなぐことができる。また UNIX コマンドは環境変数を経て他のコマンドを呼び出すことがしばしばある (例えば、man コマンドの PAGER)。それゆえに、UNIX コマンドは相互に複雑に関係し合っている。これは UNIX コマンドがオブジェクトを形式的に表現しているにも関わらず、関係を記述するのことは易しくないことを意味している。

一方、UNIX コマンドは身近であるが為に理解しやすい。これは他の本当のソフトウェアに比べて扱いが簡単であることを意味している。これらの理由により、UNIX コマンドは実験材料として適切であると判断した。

4 Emeraude PCTE

PCTE (Portable Common Tool Environment) は開放型リポジトリの為の汎用型ツールインターフェース標準である。本研究では PCTE を実現したソフトウェアの一つ Emeraude PCTE を利用した。Emeraude PCTE リポジトリはオブジェクトをリンクでつなぐことで情報を記述した、オブジェクトベースと呼ばれる。PCTE では SDS (Schema Definition Set) と呼ばれるものを用いる。SDS は ERA モデルに リンクカテゴリのような特定の性質を付加したものである。それゆえに、関係を用いてオブジェクトを生成、参照するのに適している。また、SDS は多重度などのいくつかの意味的制約をサポートしており、SDS で記述できない意味的制約は C 言語により実現できる。

以上のことから Emeraude PCTE は実験道具として適切であると判断した。

5 関係の記述

本研究ではオブジェクト間の関係を記述する際、以下の 3 階層に分かれる階層構造を用いた。

- メタ・クラス図
- クラス図
- インスタンス図

クラス図は例えば、引数とファイルの関係など静的な関係を参照するときに用いる。インスタンス図はファイル名や引数の値など、動的な情報を参照するときに用いる。メタ・クラス図は複数の UNIX コマンドの関係を統一的に記述・検索する為のメタ・情報として用いる。

6 意味的制約の実現

SDS でサポートしていない制約は C 言語を用いてプログラミングを行なうことにより実現できる。しかし、全ての意味的制約を完全に実現することは難しい (そのような制約は部分的な実現を行なう)。例えば、UNIX コマンド man は環境変数 PAGER を経て他の UNIX コマンドと関係を持つ。その際、PAGER には テキストファイルの中身をユーザに見やすく表示するコマンドがセットされることを期待する。しかし、プログラムの中でそのような意図を記述することは困難である。このとき、その意図を部分的に実現することを試みる。例えば、環境変数 PAGER に適するコマンドのリストを参照する方法が挙げられるが、これは新しいコマンドには対応していない。

7 難しさの分類

UNIX のファイルシステムは動的情報など、いくつかのソフトウェアデータベースの含んでいるが、比較的易しい形になっている。

UNIX コマンドデータベースにおける意味的制約の難しさによる分類をおこなった。意味的制約は実現可能な制約と部分的な実現可能な制約に分かれる。部分的な実現可能な制約はさらにいくつかに分かれるが現段階、得られている分類と部分的な実現方法は以下の通りである。

- 位置情報に関する制約
  - 位置情報を記述
  - find により検索

  - 他のユーザの .cshrc を検索

- 意図に関する制約
  - 定義前の適当な値のリストを参照
  - typing のような属性を与える。
  - 全ての UNIX コマンドを検索し、関係を参照する。

- 動的情報に関する制約
  - インスタンス図が参照できれば、インスタンス図を参照する。

この分類がソフトウェア開発において認められるものか、さらに議論していく。

8 まとめ と 今後の課題

まとめは以下の通りである。

- 意味的制約と関係を形式的に記述することは、しばしば非形式的に記述することと同様に難しい。
- 形式的な記述と非形式的な記述は両方必要である。意味的制約は形式的な記述だけでは記述できない。しかし、非形式的な記述をどのように扱うかという新たな問題が生じる。
- 意味的制約はそれらの性質により分類できる。本研究では現時点でプログラミングの難しい制約を 位置情報に関する制約、意図に関する制約、動的情報に関する制約に分類できている。これが UNX コマンドに限定したものではないことを議論した。

本研究のソフトウェアデータベースに対する貢献は、難しさによる分類に言及したことである。

今後の課題は以下の通りである。

- メタ・クラス図と意味的制約の分類の確立
  さらに多くの UNX コマンドを扱わなければならない。
- 分類の有効性に対する考察
  意味的制約の分類がソフトウェアデータベースの問題に対して有意義であるか考察する必要がある。

- インスタンス図の半自動作成ツールの作成
  インスタンス図により得られる動的情報は重要なので、インスタンス図の (半) 自動生成ツールが必要である。しかし、完全な自動化は現実的でない。なぜなら、コマンド の全ての情報を得るためにコマンド のプログラムのソースコード を組み込まなくてはならないからである。