4E-7 ソフトウェア開発支援のための辞書システム

宗近修久　松村一夫

(株式会社東芝　システム・ソフトウェア技術推進部)

1．はじめに

ソフトウェア生産において、複数の担当者間のコミュニケーションを正確に行なうことは重要である。しかし、実際には
・ドキュメントがわかりにくい、あいまいである
・同じ言葉を人によりちがった意味に用いる
・同じ言葉を場合によりちがった意味に用いる
などの問題があった。

我々は、ソフトウェア生産の工業化を目標とした一貫支援システムＩＭＡＰを開発しており、特に設計作業においては理解しやすく、かつ形式的表現による厳密性をそなえた設計記述法ＴＦＦとその支援ツール[1]を開発した。ＴＦＦでは、記述の一側面の形式化、つまり制御構造と記述項目の枠組を形式化しているが、形式のなかに記述する内容は自然言語であり、まだ厳密さに欠ける点がある。そこで、我々のアプローチは、制御構造や記述項目の枠組の形式化に加えて、項目の内容には限定した用語を使うこととする。本稿では、そのベースとなる用語辞書の考え方をオブジェクト指向的アプローチで述べる。これにより、用語の意味と使い方の統一を支援する。

2．用語辞書の位置付け

2．1　用語の定義

用語とは、次のものと定義する。
・設計、説明などに使う自然言語の単語（名詞、動詞）
・プログラムの中の識別子としてのネーム（名前）

2．2　用語辞書の目的

用語の意味、使い方の統一を目的として用語辞書を使う。

（１）用語の意味を統一する

ベースとなるプリミティブな用語として、分野ごとの基本単語と、その単語に対応する基本ネームを用意する。基本単語は動詞（機能名、処理名など）と名詞（処理対象データ、入出力装置名、状態など）である。

用語の意味は上記の基本単語の集まりで表現する。つまり単語の意味は、上位下位概念、類似語など単語間の関係で表現する。ネームの意味は、それを説明する単語、プログラム内の相互参照、階層構造などを用いて表現する。

（２）用語の使い方を統一する

用語について、使ってよい場所、使ってよい用語、構文の限定、新しい用語を生成登録する規則を決める。例えば、単語の使用に関して、定められた設計記述項目の欄には辞書の単語しか認めない。また、ネームに関しては、ネームの生成規則（基本ネームの組合わせの順序、長さなど）を定める、などがある。

このように、設計・プログラミング環境のベースとして位置付け、ＴＦＦの支援ツールなどと連動して、標準化設計を支援する。

3．用語辞書の構成

3．1　辞書の種類とその関係

用語辞書には標準辞書と対応辞書の２種類がある。標準辞書は標準用語の情報の集まりであり、用語の標準化に効果がある。対応辞書は工程間の用語の対応関係の情報の集まりであり、保守時などに有効である。

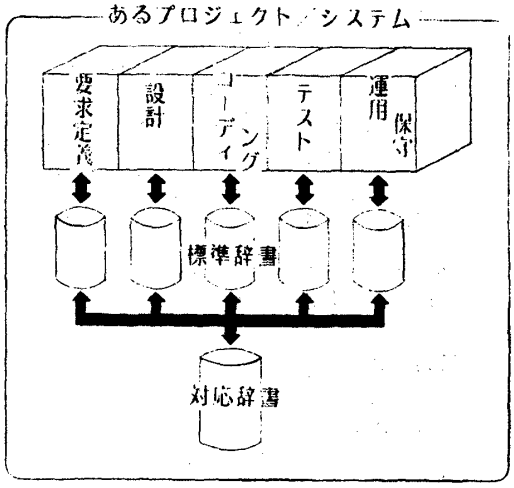

図１．　辞書の関係

２種類の辞書の関係は、図１のように１つのプロジェクト／システムに対して各工程ごとに標準辞書があり、それらの対応をとる対応辞書が１つある。

3．2　標準辞書

標準辞書は用語情報の集まりである。ここでは、用語にクラスを設定してそのクラスに属するインスタンスを用語として辞書に登録する。

（１）用語のクラス

<Class,AttributeItems,Methods>
　Class:クラス名
　AttributeItems: 属性の項目
　Methods:クラスで定義した手続き、規則

AttributeItemsは、クラスに属するインスタンスとしての用語情報の型枠を定義する。Methods は用語の使い方を規定したものである。

（例）クラス「変数」の情報は、例えば次のようになる。
　Class:変数
　AttributeItems: 型、表題、説明、サイズ、…
　Methods:変数名を生成する規則

（２）用語のインスタンス

<Name,Class,Attributes,Relations >
　Name: 用語名
　Class:用語のクラス名
　Attributes: 用語の属性とその属性値の組の集まり
　　（属性の項目はClass により決まる。）
　Relations:関連する用語名（ポインタ）と関連の種類の組の集まり

AttributesもRelations も用語の意味記述であるが、Attributesは用語自体の属性であり、Relations は他の用語との関連を表わす。

Dictionary System for Supporting Software Development

Nobuhisa Munechika　Kazuo Matsumura

TOSHIBA CORPORATION

(例) 変数「error_flag」の情報は、例えば次のようになる。
Name: error_flag
Class:変数
Attributes: 型 (int) , 表題 (エラーフラグ) ,
説明 (ステータスがエラーのときon) , …
Relations:使用モジュール名 (kkk,ggg) , …

3．3 対応辞書

<Name,Pointers >
Name: 対応関係を持つ用語名
Pointers: Nameとの対応関係の集まりで
(PointerType,Partners) からなる。
PointerType : 対応の種類
Partners: 対応先の用語名の集まり

対応辞書は工程間の用語の対応関係の情報の集まりであり、標準辞書の用語を通して情報を持つ。これにより、ある用語の前後工程への影響範囲などを正確につかめる。
(例) 用語名「辞書ファイル」の情報は次のようになる。
Name: 辞書ファイル
PointerType:対応するコーディング工程の用語
Partners: dict_file1 , dict_file2

4．辞書の運用

標準辞書の中の標準用語は
①プロジェクト／システムに依存しないある分野で共通する基本用語
②プロジェクト／システムに固有の用語
に分類できる。

①の基本用語は、基本用語辞書にあらかじめ準備する。したがって、プロジェクト開始時に基本用語辞書から標準辞書へ基本用語を取込んで使う。

基本用語辞書を作成するには
(1) 既存の設計、ソースコードからネームを収集する。
(2) そのネームを説明する単語を収集する。
(3) それらの単語、ネームを選択して編集する。
の順で行なう。

②のプロジェクト固有の用語については、プロジェクトの進行に合わせて追加、増殖していく。

辞書システムの主な機能は
・基本用語辞書の作成支援機能
・基本用語の初期化 (取込み) 機能
・編集 (登録、変更、削除) 機能
・検索参照機能
・アクセス機能
などである。編集機能により登録する用語は、承認手続き (責任者のチェック) が必要である。なお、運用上一時的に承認されていないものも認める。

辞書システムの機能を使って編集、検索などを行なう以外に他のツールからこれらの機能が使えるアクセス機能を提供する。例えば、仕様書作成エディタから、辞書にデータを自動登録したり、エディタにより変更した箇所を辞書に反映させたりすることが可能となる。

5．プロトタイプシステムの作成

標準辞書のプロトタイプシステムを、オブジェクト指向型言語TOOL[2](Talking Object Oriented Language)を用いて作成した。本プロトタイプシステムでは、4章で述べた機能のうち、編集 (登録など) 、検索参照機能を作成した。ここでは、登録機能の1つのネームの生成についてＴＯＯＬによるプログラム例 (図2) を用いて説明する。

①は、クラスの階層構造の定義である。
②は、各クラスのインスタンスを作成している。
③は、用語間のRelations を作成している。(オープンするに対応するネームはopen)
④は、クラス「関数」のMethods 「付く」と「登録さる」の定義である。「付く」がネームの生成の手続きである。この例でのネーミング規則は、簡単のため

```
① 単語 と ネーム の 種類 は 用語 。
  名詞 と 動詞 の 種類 は 単語 。
  変数 と 関数 の 種類 は ネーム 。

② オープンする の 種類 は 動詞 。
  ファイル の 種類 は 名詞 。
  open の 種類 は 変数 。
  file の 種類 は 変数 。

③ オープンする の ネーム ← open 。
  ファイル の ネーム ← file 。

④ 関数 が 付く とは
    「 (操作名 の ネーム) と
       __ と
       (対象名 の ネーム) を
       つなぐ 。」 。
  関数 が 登録さる とは
    「 関数 が 付く 。
      答 を 関数 へ コピーする 。 」 。

⑤ ある関数 の 種類 は 関数 。
  操作名 ← オープンする 。
  対象名 ← ファイル 。
  ある関数 が 登録さる 。
```

図2．ＴＯＯＬによるプログラム例

＜関数名＞::＝＜操作名＞__＜対象名＞
とする。

⑤により、クラス「関数」にメッセージを送り実際に関数名open_file を生成し登録する。

以上簡単な例ではあるけれども、個々の用語が互いの関連を含めてダイナミックに増えるような用語辞書実現にオブジェクト指向は適しているという感触を得た。

ネーミング規則は運用上大切であり、言語、ＯＳなどの制約 (ネームの長さ、大文字、小文字の区別) を考慮したバリエーションが必要である。例えば
＜関数名＞::＝＜動詞１＞__＜名詞１＞|
＜動詞２＞__＜名詞２＞ __ ＜名詞２＞
＜動詞１＞::＝動詞１に対応するネーム
＜名詞１＞::＝名詞１に対応するネーム
＜動詞２＞::＝動詞２に対応するネームの短縮形
＜名詞２＞::＝名詞２に対応するネームの短縮形
というように、ネームが長くなるときは短縮形を使うことも必要である。また、ネームの意味が自然と反映するように (短縮形の作り方を含めて) 規則を作る必要がある。

6．おわりに

ソフトウェア生産において、用語の意味と使い方の標準化を支援するための用語辞書について提案した。用語として単語とネームを同じ世界で扱うことにより、ライフサイクルを一貫して標準化支援が可能となる。

今後の課題として、用語の分析、用語辞書の効果、非標準 (個人) の用語の扱いなど、運用環境を含めて検討する。

参考文献

[1] 松村他，“ＩＭＡＰシステム (2) ー設計・コーディング支援技術５０ＳＭ－”，情報処理学会第３１回全国大会予稿集
[2] 森本他，“オブジェクト指向型知的パソコン言語ＴＯＯＬの開発”，情報処理学会第３０回全国大会予稿集