# 統合プログラミング環境（4）

## ソフトウェア設計情報用データベースPDDに基づく開発環境

5H−4

濱崎　勝久　湯原　義彦　小林　茂
（株）東芝

### 1 はじめに

一般に、ソフトウェア開発においては非常に多くのソフトウェア・オブジェクト（以下、単にオブジェクトと呼ぶ）を対象として操作を行う。例えば、レコード、モジュールあるいはそれらを処理するコマンドなどもオブジェクトの一種である。

従来のソフトウェア開発環境では、これらのオブジェクトやオブジェクト間の関係を開発者が全て意識しながら作業を行わなければならず、大きな負荷となっていた。また、各コマンドは、そのコマンド独自の出力形式でオブジェクトを生成するため、他のコマンドがその情報を利用するのも簡単ではなかった。

これらの情報の管理を効率化するためにソフトウェア設計情報用データベースPDD（Program Data Dictionary）[1][2]をスーパーミニコンピュータDS6500シリーズおよびスーパーワークステーションASシリーズ上に開発し、このDBを利用したソフトウェア開発環境を構築した。これによって情報の取り出しが統一されたオペレーション／表示形式で的確に行えるようになる。また、コマンド間の情報交換もDBを通して可能となる。本稿ではPDDを中心としたソフトウェア開発環境の構想と、そのDB機能を利用した開発支援ツールの実現例について報告する。

### 2 PDDに基づく開発環境の構想

我々は、PDDを中心とした統合的なソフトウェア開発環境を構築している。その構成を図1に示す。

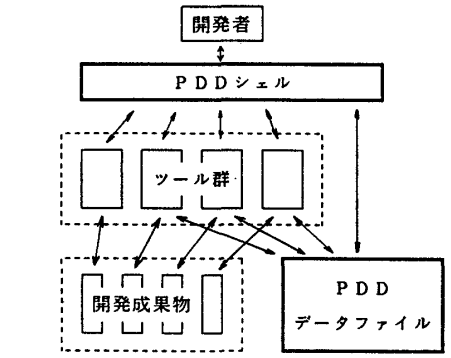


図1　PDD環境の構想

この環境における開発形態を最も特徴づけるのはPDDシェルである。PDDシェルはツールの実行や情報検索あるいはそれに伴うDBの操作をメニューによって行うためのユーザインタフェースを定義する機能を有する。本環境の主な特徴として以下の点が挙げられる。

1）ツール実行時の指示の簡略化が可能

通常の環境におけるコマンド実行時の引数の多くは、関係情報によって補うことができる。例えば、あるロードモジュールをリンクする時、それに必要なオブジェクトプログラムは「ロードモジュールと構成要素ファイル」という関係から求められる。また、あるソースプログラムをコンパイルした結果をどこに出力するかは、「ソースプログラムとオブジェクトプログラム」の関係から求められる。PDDシェルが、メニュー形式という入力情報量の限られたインタフェースによって効率の良い開発を実現できるのはこれによるところが大きい。

2）高度な情報検索が可能

PDDのRDB機能により、情報を任意の観点から柔軟に求めることが可能である。例えば、「AはBをインクルードしている」と言う関係から、「Aがインクルードしているものは」、「インクルード関係にあるファイルは」等の条件検索に見合った情報を取り出せる。また、開発途中で発生した変更の影響範囲を調べたり、設計情報について相互の関係を検証することにより矛盾を検出することも可能である。

3）開発生成物の自動生成が可能

目的の生成物を最小の手段で自動的に生成する機能（make機能）。例えば、ソースプログラムやオブジェクトプログラムを修正日時を比較することにより自動的にコンパイルを行うことなどを容易に実現できる。

### 3 PDDを用いた実用例

PDDを用いて実現されているプログラム開発ツールの例としてプログラム情報検索ツールとプロジェクトライブラリについて説明をする。

#### 3.1 プログラム情報検索ツール

情報検索ツールはソースプログラム中のクロスリファレンス情報（シンボル情報）を管理するツールである。

従来、ソースプログラムに関する情報はクロスリフ

---

Integrated Programming Environment based on Design Information Database

Katsuhisa Hamazaki, Yoshihiko Yuhara, Shigeru Kobayashi
TOSHIBA Corp.

ァレンスリストとソースプログラムを相互に参照することにより得ていたが、ソフトウェア開発の複雑化や大規模化にともなって、プログラムのシンボル情報の管理が問題となる。

情報検索ツールはＰＤＤと図２のようなシステム構成をとり、コンパイラにより作成されるシンボル情報を図３のようなオブジェクト型レコード形式に変換してＰＤＤに登録する。

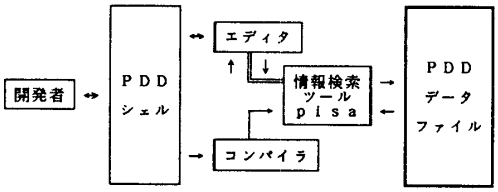


図２　情報検索ツールのシステム関連図

```
オブジェクトレコード
  ・symbol(filename, symname, category, type)
      filename・・シンボルが定義されているファイル名
      symname ・・シンボル名
      category・・シンボルの種別（関数、変数等）
      type・・・・シンボルの型（整数、実数等）
  ・module(filename, modname)
      filename・・関数が定義されているファイル名
      modname ・・関数名
関係レコード
  ・refer(symbolid, moduleid, line, etype)
      symbolid・・シンボルＩＤ
      moduleid・・モジュールＩＤ
      line・・・・出現行数
      etype ・・・出現状態（定義、参照等）
```

図３　情報検索ツールの登録レコード形式

開発者は、この登録されたシンボル情報に対しシンボル名称や属性、用法、出現ファイル名等を種々の複合条件指定することによりシンボル情報の絞り込み行い、必要とするシンボル情報の検索を行うことができる。ここで検索されたシンボル情報はマルチウィンドウ指向エディタの１つのウィンドウ上に表示される。開発者はこの表示されたシンボル情報とソースプログラムとを同時に参照することによりシンボルに関する変更、確認が容易に行えるようになる。また、ここでのエディタとの結合はエディタ上にツールの情報を表示するインタフェースを構築することにより実現しており、これにより従来、エディタには備わっていなかったＤＢ機能がツールを介在することにより実現できるとともに、特に、結合するツールを意識せずに必要な情報の交換が可能となる。

ソースプログラムに関する情報もＤＢ化することにより単なるクロスリファレンス情報でなく、他のツールからも利用することができるようになり、開発環境としてのデータ共有も可能となる。

### 3.2 プロジェクトライブラリ

プロジェクトライブラリとは当社のＡＳシリーズ上のＣＡＳＥツールを構成するツールであり、設計情報エディタ等で生成される設計書や仕様書に関する設計情報を管理する。

構造化エディタやチャートエディタにより作成される設計書や仕様書は、モジュールやファイルに関する情報を得るための資料となるが、仕様書単位には独立しており設計書や仕様書に含まれる情報の相互関係は得られない。例えば、どの仕様書のどこにどの様なモジュールが含まれていて、そのモジュールがどのモジュールから参照されているかという関係は仕様書のレベルでは分からない。プロジェクトライブラリは各設計書や仕様書に含まれる情報をＤＢ化することにより、モジュール間の定義・参照関係を要求に応じて表示する設計情報の管理ツールである。

ＣＡＳＥツールは図４のようにエディタやライブラリにより構成され、開発者はプロジェクトライブラリにより各種エディタで作成した設計書や仕様書の矛盾や抜けを検出し、プロジェクトとしてのリンケージチェックを行うことができる。また、プロジェクトライブラリはＣＡＳＥツールとして独立したインターフェースを備えているためＰＤＤシェルは利用せずＤＢ機能として利用している。

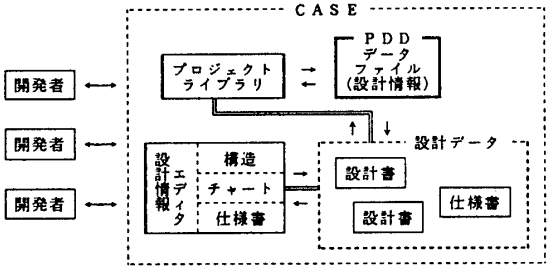


図４　ＣＡＳＥツール構成図

### 4 おわりに

設計情報用データベースＰＤＤにより実現したソフトウェア開発支援ツールついて報告した。ここで報告したＰＤＤの使用法はツール固有の機能を実現するためであり他ツールとのデータ共有には至っていない。今後、このＰＤＤをより有効に利用して行くための開発支援ツールのＤＢの使用法についてさらに考察を進め、より多くツールに適応と統合化の促進を行っていく予定である。

参考文献

[1]小林他:「統合プログラミング環境(2) - 開発支援データベース -」、情報処理学会第３９回全国大会
[2]湯原他:「統合プログラミング環境(3) - ソフトウェア設計情報用データベースＰＤＤの開発 -」、情報処理学会第４１回全国大会