ソフトウェア工学 40－20
（1985. 2. 8）

プログラミングデータベース：SODA

上原憲二，中島 毅，石川由美子，高野 彰，春原 猛
三菱電機(株)情報電子研究所

1. はじめに

近年、ソフトウェアの生産性、品質の向上が極めて重要な課題となっている。この課題解決のため統合化されたソフトウェア生産環境の研究開発が進められている[1,2]。プログラミングデータベースはこのような環境の一構成要素であり、その役割はソフトウェア開発プロジェクトについて、

- ライフサイクルの全情報を格納する。
- プロジェクトの進行を記録する。
- 格納された情報への容易なアクセスを提供する。

があげられる。

SODA(Software product DAtabase manager)はそのようなデータベースの一つであり、ソフトウェア開発者（プロジェクトに属する人員）に次のような便宜を与えることを目標としている。

- 雑作業が排除され作業が効率化される。
- 必要な情報が簡単に検査でき、または自動的に検査されることによって誤作業（不正作業、冗長作業）が防止される。
- 管理用情報が簡単または自動的に収集されることによって、管理作業を効率的に行える。

そのためSODAは実際のライフサイクルにおける生産物と作業に基づいている[3]。SODAデータベースは次の特徴をもつ。

- 生産物（仕様書，ソースコード等）を4層の木構造に階層化して管理する。
- 生産物に関し、その属性、他生産物との関係を保持する。
- 作業で使用するツールについて、その名前、入出力生産物の種類を保持する。

SODAデータベース管理は次の特徴をもつ。

- 生産物の検索は4層木構造に基づく検索と関係に基づく検索ができる。そのとき、属性によって検索される生産物に条件を付けることができる。
- ツールは簡単なコマンドで利用できる。SODAがそれをホストOSのコマンドに変換する。また、ツールに対応していくつかの管理用の属性の自動収集を行う。

2. 開発作業の分析と問題点

効率的で誤りのない作業を実現するために、まず開発作業がどのように行われているか分析する。

①組織によって標準的なライフサイクルがあり、作成すべき生産物も決められている。

②設計から製造段階においては、外部仕様書、内部仕様書、ソースコードの作成というように、各作業と生産物が対応している。工程管理も同様であるが、生産物完成度（例：仕様書の第何章まで完；ソースコード予想ステップ数の何％完等）などの情報も用いられる。

③試験段階では生産物の集合が作業に対応する。例えば、単体試験では作業はモジュール対応であり、ソースコード、オブジェクトモジュールが含まれる。更に、ドライバー、スタブが必要であり、試験の実行では被試験モジュール、ドライバー、スタブからなるプログラムが作業に対応する。工程管理はモジュールごとに試験完了を見るが、試験項目数などの情報も用いる。

④開発は多くの場合分業であるが、開発者間のインタフェースが最小になるように生産物が分担される。これは②、③で述べたような対応にできるだけ基づいて行われる。

このような作業における問題点を次に述べる。

① まず、生産物の識別が重要である。例えば、モジュールについてはソースコード、オブジェクトモジュール、またプログラムについては内部仕様書、ロードモジュールおよびその構成要素であるモジュールの対応が容易にわかることが望ましい。これは名前付け規約である程度解決できるが、複数のプログラムを同時開発している場合はそれでも不十分である。

② 作業を行うためにコマンドを入力する必要がある。リンクコマンドのように多くの生産物を指定する場合はめんどうで誤りやすい。

③ 指定生産物が少ないコマンドでも何回も入力するときはめんどうで誤りやすい。これは生産物が多い場合、原因の分りにくい誤りを生むことがある。コマンドファイルはこれを避けるために用いることができるが、融通性に欠け冗長作業を生むことが多い。必要十分な作業を容易に知るまたは指示できることが重要である。

④ 工程管理においては、何種類かの情報が生産物の担当者によって収集され管理者に報告される。管理者は情報のまとめと評価を行う。情報の収集、まとめ、評価はできるだけ自動化されることが望まれる。

3. SODAの構成

SODAの構成を図1に示す。既存ツールを変更なしで使用可能とするため、データベースにおいて生産物自身はホストOSのファイル（実ファイル）として存在する。データベース管理は、基本ルーチン、構築プロセッサ、操作報告プロセッサ、ツール起動コマンドプロセッサから構成される。

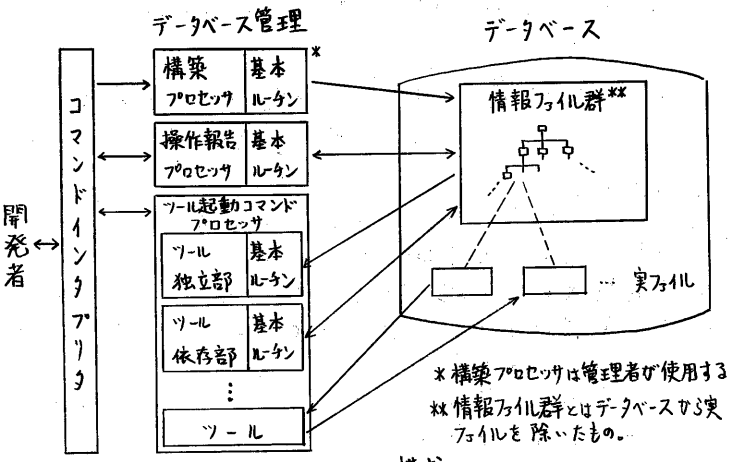

図1. SODAの構成

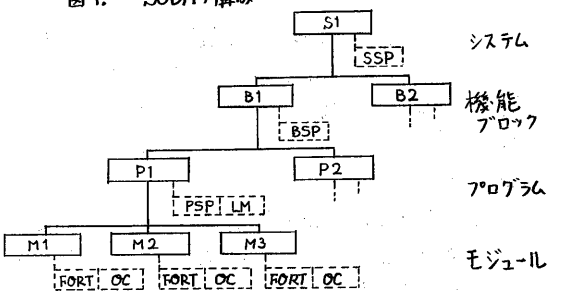

図2. SODAの4層木構造

4. SODAデータベースの概要

SODAにおけるデータの項目と構造を次に示す。

(1) 4層木構造

SODAでは生産物を4層の木構造で管理する。それは、ソフトウェアの構成がシステム、機能ブロック、プログラム、モジュールの4層をなしており、生産物も各構成要素に対応して存在している。そして、2章で述べたような作業と生産物の対応はこの構造上で素直に表現されているからである。

図2に4層木構造の例を示す。構成要

S1:B1:P1	S1システムのB1機能ブロックのP1プログラム			
S1:B1:P1:M1	S1 〃	B1 〃	P1 〃	のM1モジュール
S1:B1:P1/LM	S1 〃	B1 〃	P1 〃	のロードモジュール
S1:B1:P1:M1/FORT	S1 〃	B1 〃	P1 〃	のM1モジュールのFORTRANソースコード

(注) パス名はノード名とノード名間は':', ノード名とタイプ名間は'/'で連結する。

図3. パス名の例

素はノードとして表される。これらのノードは構成要素の層間の包含関係によって木構造をなす。例えば、P1プログラムはM1,M2,M3モジュールからなることがわかる。生産物はそれが対応する構成要素のノードに属し、生産物の種類を表すタイプ名によって表される。例えば、P1プログラムに対応して内部仕様書とロードモジュールが存在することがわかる。

このようなSODAデータベースにおいて各ノード、生産物を識別するために、システムノードからそのノードまたは生産物までのノード名とタイプ名を連ねたパス名を用いる。図3にパス名の例を示す。このようなパス名は生産物の識別を容易にし、識別法に標準を与える。これによって例えば、実際には生産物を作らない管理者でも、実際の担当者とインタフェースをとらなくとも望む生産物を識別することができる。

また、上記4層木構造やパス名は作業と対応する生産物の集合をとらえることを容易にしている。例えば、「P1プログラムを構成する生産物すべて」は4層木構造上、パス名上明白に表現されている。

更に、パス名におけるノード名、タイプ名にワイルドキャラクター('**'で表す)を導入することにより、生産物の指定をより便利にしている。図4に例を示す。

(2) SODAファイル

SODAデータベースは、各生産物に対応して生産物自身、その属性および他生産物との関係を保持する。この対応を表す

S1:B1:P1:**	P1プログラムの全モジュール
S1:B1:P1:**/FORT	P1プログラムの全FORTRANソースコード

図4. ワイルドキャラクターの例

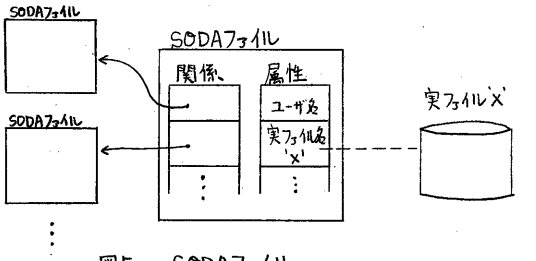

図5. SODAファイル

ために生産物はSODAデータベース上ではSODAファイルとして表される(図5参照)。生産物自身はホストOSのファイル(SODAファイルに対し実ファイルと言う)として存在する。SODAファイルと実ファイルの対応は、属性のひとつとして実ファイル名をもつことで表す。

(3) 属性

属性は生産物の性質や管理情報を格納する。それを利用して生産物の検査や工程管理が容易になる。以下に主な属性について説明する。

基本的な属性として実ファイル名の他にユーザ名がある(図5参照)。これは生産物の担当者を表し、生産物に対するアクセス制御に利用される。

生産物の新旧を判定するために最新更新日時(RDTと表記)は重要である。利用例を図6に示す。

```
CNODE   S1:B1
LIST    :P1:**/FORT,%RDT>85/2/8/12:00
```

(a) 操作報告プロセッサのSODAファイルリストコマンド

```
COMPILE
RANGE ==>*
F     :P1**
CONDITION ==>
%RDT = TODAY***
```

*下線はプロセッサ出力を示す。
**P1プログラム以下のすべてのSODAファイルを表す。
***ホストOSのもつ本日の日付

(b) ツール起動コマンドプロセッサ COMPILE

図6. 最新更新日時の利用例

図6(a)で操作報告プロセッサはSODAデータベースに対する操作を会話型で行うものである。1行目のコマンドは現在ノードを'S1:B1'に設定する。これは以後のパス名の省略記法を許す。2行目の':P1:**/FORT'は'S1:B1:P1:**/FORT'のことである。2行目はSODAファイルをリストするコマンドであり、コンマ(,)以降は属性条件を示す。この行は、本日(85/2/8を本日とする)午後更新したSODAファイルのリストを指示している。

(b)でツール起動コマンドプロセッサはツールを簡単なコマンドで利用可能にするものである。ここでは、P1プログラムのすべてのソースコードのコンパイルを指示している。ただし、最後の行がRDTを利用して、「本日修正をしたもの」という属性条件を指定している。

生産物の品質面を表す属性として、レコード数(REC)、更新回数(UCT)、コンパイル回数(CCT)、デバッグ使用回数(DCT)などがある。ソースコードにおいてREC、UCT、CCTが他より特に大きいもの、ロードモジュールにおいてDCTが特に大きいものは管理上の要注意点のひとつである。これらの属性は自動収集される。特に、UCT、CCT、DCTは対応するツールのツール起動コマンドプロセッサが収集する。例えば、図6(b)ではコンパイルされるソースコードのCCT、出力オブジェクトモジュールのUCTを1増加する。

工程管理のための属性として、予想レコード数(EREC)、レコード数 重要度(PRI)、完了予定日(PDT)、作成開始日(CDT)、完了日(FDT)、進行状態(STA)がある。ソースコードのEREC、RECは作成ステップ数の点から進捗管理の情報となる。PRIは管理上の重要度を表す。例えば、入出力モジュール、制御モジュールなど他のモジュールの工程に大きな影響をもつモジュールの生産物には高い重要度が与えられる。PDT、CDT、FDTは工程の予定と実績を表す。STAは作成作業が未着手、作成中、検査中、完了のいずれであるかを表す。これらの属性は、工程と実績との差、工程管理上の注意点の把握を容易にすることを目的としている。これらの属性のうちREC以外は自動収集されないので、生産物の担当者が操作報告プロセッサの属性設定コマンドにより設定しなければならない。これらの属性を簡単にまとめて表示するコマンドが操作報告プロセッサに用意されている(5章図8参照)。

(4)生産物間の関係

生産物間の関係には、4層木構造に沿ったものと、それとは異なる経路による生産物の指定法を与えるものがある。

前者の関係には生産物の作成順序関係(ORDERと表記)がある。これは生産物間の矛盾発見(例えば、要再コンパイルモジュールの発見)において最新更新日時属性とともに基本的な情報である。

後者の関係としてモジュール構成(呼出し)関係(KOUSEI)、インクルード関係(INC)がある。例えば、KOUSEIを用いればあるモジュールの単体試験で必要な下位モジュールをすべて知ることができる。INCを用いればあるインクルードファイル(タイプ名はINC)の変更により再コンパイル必要なソースコードをすべて知ることができる。図7は図2に加え、P1プログラムの下にM4モジュールが、M4モジュールにインクルードファイルがあり、それをM1,M3モジュールがインクルードしていることを示している。

関係にも属性と同様に自動的に設定さ

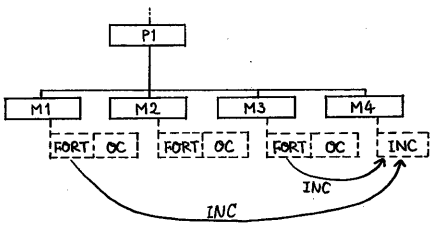

図7. インクルード関係の例

れるものとされないものがある。4層木構造に沿うものは前もってタイプ名間の関係として設定できる。例えばORDERは

PSP → FORT → OC → LM

としておけば各タイプのSODAファイルがデータベースに作られた時点でその関係が設定されたことになる。しかし、KOUSEI、INCなどは必要となった時点で個々の生産物間に設定しなければならない。これは操作報告プロセッサの関係設定コマンドを用いて行う。

(5)ツールと入出力タイプの関係

この関係はプロジェクトで使用するツールを登録するものである。登録する情報はツール名（ホストOSでそのツールを起動するコマンドの名前）とツールの入出力生産物のタイプ名である。ツール起動コマンドプロセッサはこれを用いてツールの起動コマンドを簡単化している。例えば図6(b)のCOMPILEでは入力のタイプ名を明示しなくとも良く、誤ったタイプ名指定もチェックされる。また、出力については何も指定しない。

5. データベース管理の概要

データベース管理の各構成要素について次に示す。

(1)基本ルーチン

データベースの構築、情報（実ファイルを除く）の格納、検索、削除を行うデータベースアクセスルーチンの集まりである。SODAデータベース（実ファイルを除く）へのアクセスはすべてこれらのルーチンを用いて行われる。

(2)構築プロセッサ

プロジェクトの開始にあたりSODAデータベースを構築する。構築に際し、システム名（開発対象ソフトウェアを表す）、開発者、生産物のタイプ、属性の種類、関係の種類が指定でき、従ってプロジェクトの要求に応じたSODAデータベースにすることができる。

(3)操作報告プロセッサ

開発作業中に開発者がデータベースに情報を格納したり表示させたりするための会話型プロセッサである。構築直後のSODAデータベースにはシステムノード以外何もない。開発の進行に伴い開発者はこのプロセッサのコマンドを用いてノードやSODAファイルの生成、属性や関係の設定を行う。また、これら情報を表示するコマンド、工程管理用属性の表示コマンド(図8参照)などがある。

```
PROGRESS :P1
(NODE) S1:B1:P1
S1:B1:P1/PSP
  (USER) YAMADA      (REC)  371 (CDT)85/01/30
  (STA)  COMPLETION  (EREC) 400 (FDT)85/02/01
                                (PDT)85/02/01   (a)
S1:B1:P1/LM
  (USER) YAMADA      (REC)   59 (CDT)85/02/05
  (STA)  TESTING     (EREC)  70 (FDT)
                                (PDT)85/02/12
(1-LEVEL LOWER NODES)
S1:B1:P1:M1/OC
  (USER) YAMADA      (REC)  129 (CDT)85/02/04
  (STA)  COMPLETION  (EREC) 300 (FDT)85/02/07
                                (PDT)85/02/07
S1:B1:P1:M2/FORT
  (USER) YAMADA      (REC)   97 (CDT)85/02/05   (b)
  (STA)  TESTING     (EREC) 100 (FDT)
                                (PDT)85/02/08
   ⋮
```

図8. 工程管理用属性の表示例

図8で(a)の部分はコマンドで指定されたノードについて、生産物の作成順序関係(4章(4)のORDER)の順で工程管理用属性を表示したもの。(b)は1レベル下層の各ノードについてORDERの順で進行状態属性(STA)が完了でない最初の生産物の表示をしたものである。図からはM1モジュールは検査済、M2モジュールは検査中であることがわかる。

(4)ツール起動コマンドプロセッサ

開発作業の多くは必要な生産物に必要なツール（例えば、仕様書エディタ[4]、テキストエディタ、コンパイラ、リンカー）を適用して進められる。このプロセッサは、ツール起動のコマンド、対象生産物の指定を容易にすることによって、効率的で誤りのない作業の実

現をはかるものである。

図6(b)において、'F :P1'はP1プログラム以下すべてのSODAファイルを指示している。そしてコンパイルであるのでそのうちのソースコードのSODAファイルすべてが選択され、更に各ソースコードについて言語に対応したコンパイラが起動される。出力オブジェクトモジュールは、タイプ名がOCであることを除いてソースコードとパス名が同じSODAファイルである。

図9に図7に示したインクルード関係を利用したコンパイルを指示する例を示す。'CR INC :P1:M4/INC'はM4モジュールのインクルードファイルから関係INCを逆にたどって直接または間接に得られるSODAファイルを指定している。属性条件はそのうち少なくとも作成済みのものを指定している。

```
COMPILE
RANGE ==>
CR INC :P1:M4/INC
CONDITION ==>
%STA=TESTING | %STA=COMPLETION
```

図9. インクルード関係によるCOMPILE

図10にリンクの例を示す。リンクでは対象生産物の指定をロードモジュールに対して行う。'F S1:B1'はB1機能ブロック下のロードモジュールすべてを指示している。各ロードモジュールに対しリンクされるオブジェクトモジュールは、そのロードモジュールの属するプログラムの下のすべてのモジュールのオブジェクトモジュールである。

```
LINK
RANGE ==>
F S1:B1
LIBRARIES ==>
<ライブラリの入力>
CONDITION ==>
% STA = TESTING
```

図10 ツール起動プロセッサ LINK

として、属性条件は検査中のものを指示している。

また、新しいツールをこのプロセッサが対象とするツールに追加することを容易にするために、このプロセッサはツール独立部のプログラム、依存部のプログラムから構成されている。依存部は、ツールに対応した属性の自動収集部、実際にツールを起動するホストOSのコマンドを生成する部分からなる。

6. おわりに

ソフトウェア開発者は生産物の作成において、作業を効率的にまた誤りなく進めるために幾つかの情報管理を行っている。SODAはこの情報管理を支援するものである。SODAの特徴は、ソフトウェアシステムの階層構成(システム—機能ブロック—プログラム—モジュール)に対応して生産物を4層の木構造で管理することにより、①望む生産物への容易なアクセス、②ツール利用における簡単なインタフェース、③生産物間の矛盾防止支援、を実現していることである。

SODAは現在MELCOM COSMO 900Ⅱ UTS/VS上で開発を終えたところである。今後の課題は、まず実際のプロジェクトでの試使用によるSODAの評価がある。また、誤作業防止のための検査の自動化はまだ不十分であり構成管理機能[5]の検討が必要である。更に、管理情報については、まず具体的にどのような管理情報が有効なのかを検討し、次にその自動収集、表示法について検討する必要がある。

参考文献

1) U.S. Department of Defense; Requirements for Ada Programming Support Environments: Stoneman, 1980.

2) 佐藤ほか, 機能に着目したソフトウェアツール体系化の提案, 第25回情処全大3E-8.

3) 石川ほか, ソフトウェア開発管理のためのプログラミングデータベース, 第28回情処全大1T-5.

4) 高野ほか, 仕様書作成支援ツール, 第27回情処全大3B-1.

5) 中島ほか, プログラミングデータベース上での構成管理機能, 第30回情処全大2T-8.