SSDの実用化における諸問題について

内田裕士，毛利友治（富士通研究所）

1. はじめに

近年情報化社会の急速な発展に伴い，増大する情報を正確かつ迅速に処理する必要性が増して来ている。この必要性を満たすための情報処理システムも必然的に大規模，複雑化して来ているが，そういった情報処理システムを作成するための技術は種々の改良があったにもかかわらず，社会からのニーズを十分に満たせる程には進歩していない。確かにプログラミング言語の改善（機械語レベルから ハイレベル言語に変わって来たことなど）やストラクチャードプログラミングなどに代表されるプログラミングの方法論の改善などプログラミング技術においてはかなりの進歩の跡がみられるが，情報処理システムの要求や設計などの仕様定義技術，あるいはでき上った仕様やプログラムなどの正当性を確認するテスト技術などは現時点において十分であるとは言い難い。

情報処理システムの要求の定義や設計などをより厳密にし，開発の後の段階からのフィードバックをできる限り減らすことがソフトウェア開発のコストダウンおよびソフトウェアの信頼性の向上に大いに役立つであろうという観点から諸外国，特にアメリカにおいてISDOS（ミシガン大学），SADT（Softech），SREM（TRW）などが開発され実用化されている。しかしながら実際の有効性についてはまだまだ疑問な点が多い。

我々もソフトウェアのライフサイクルの論理設計段階からコーディング段階までの一貫性を保ち，ソフトウェアの開発を支援するSSD（Software for Structured Development）と呼ばれるシステムを52年に開発した。以下2.ではSSDの概要を，3.では適用プロジェクトの概要を述べ，4.5.6.で試用結果に関する考察を行い，実用化における問題点について技術的あるいは人間工学的な観点から述べ，それらの問題を克服するための今後の計画について述べる。

2. SSDの概要[1),2)]

SSDは情報処理システムの仕様を記述するための言語SDL（System Design Language），プログラムを記述するためのプログラミング言語群SELs（FORTRAN,etc.），仕様を分析するためのSDA（System Design Analyzer），プログラムと仕様との間の一貫性のチェックを行うためのPA（Program Abstructor），仕様やプログラムを始めとしてプロジェクトに関するすべての情報が貯えられるPDB（Project Data Base）やその他の支援ツール群から構成されている。SSDの処理系の構成を図1に示す。

SSDの主な機能はソフトウェアの階層的設計を支援することである。これは階層間の一貫性をチェックすることによって行われる。ソフトウェアシステムの仕様はSDLを用いて階層的に記述され，その仕様の完全性および一貫性がSDAによってチェックされる。さらに仕様が定義されたソフトウェアシステムはFORTRANなどのプログラミング言語で作成され，仕様

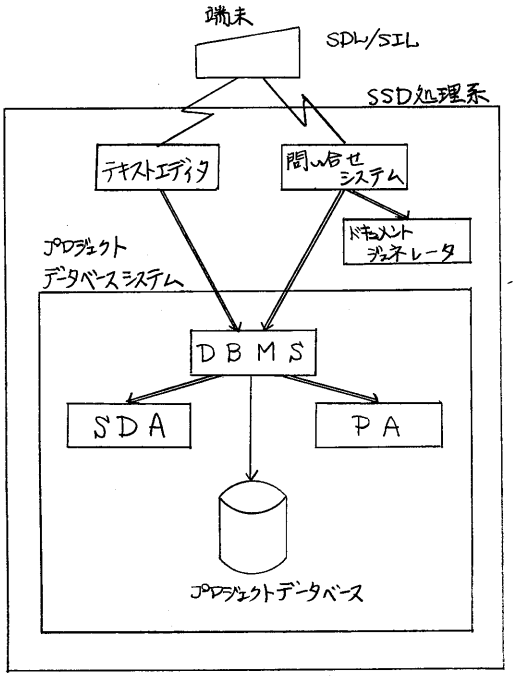
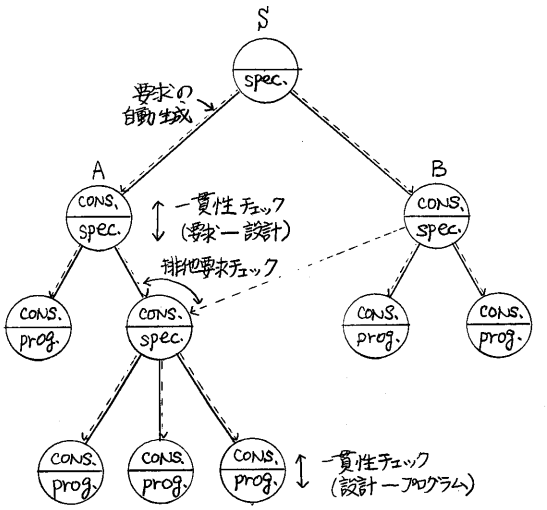


図1　SSD処理系の構成

との一貫性がPAの助けによってSDAでチェックされる。このようにして仕様定義からプログラムまでのすべての階層にわたって一貫性が保たれる。また仕様からは何種類かのドキュメントが自動的に生成されるようになっている。

ソフトウェアシステムはシステムを構成するコンポーネントおよびコンポーネント間に成立する関係（相互作用）を定義することによって記述される。これらのコンポーネントは階層的に記述されるので図2のような木構造になる。この木構造において節はコンポーネントを表わす。部分木をなすコンポーネントはシステムと呼ばれ概念的にソフトウェアシステムを構成する。葉をなすコンポーネントはモジュールと呼ばれ実際にソフトウェアシステムを構成する。システムの記述はシステム仕様と呼ばれる。システム仕様の主な内容は子コンポーネントの定義と，それらのコンポーネントがはたす役割りの定義である。コンポーネントのはたす役割りは，コンポーネント間の関係およびそれらの関係が成立する順序を定義することによって記述される。システムの仕様からは，そこで定義されたコンポーネントに対し，それらがはたすべき役割が要求条件（constraint）として各々のコンポーネントに課せられる。モジュールはFORTRANなどのプログラミング言語で定義される。

SSDは図2に示すようなやり方でコンポーネント間の矛盾をなくし，階層間の一貫性を保つことによってソフトウェアの階層的設計を支援している。



spec.　specification
prog.　program
cons.　constraint

図2　ソフトウェアの階層構造

3、適用プロジェクトの概要

ＳＳＤを適用したプロジェクトの概要を以下に述べる。

1) プロジェクトの内容
索引順次編成ファイル操作用ユーティリティの開発。但し既存のユーティリティの改良という形で開発された。

2) 適用方法および目的
既存のユーティリティを分析，整理してＳＤＬで記述し，システム像の把握を行う。設計段階に適用し，機械処理によるエラーの早期発見および，設計意図を正確にコーディング段階に伝えることを目的とした。

3) 適用結果
- システムサイズ
  ソース（ＳＰＬ／100）　　ﾁｮｳ ｶﾝ
  ＳＤＬ　　ﾖﾝﾋｬｸ ｶﾝ
- 工数
  9人月
- 計算機時間
  96Ｈ（内17ＨはＳＳＤ使用分）
- その他
  ＳＳＤを使用したことによる工程的な遅れはなく，作成段階でのバグは少なかった。その後の障害は1件のみであった。

4、適用結果に関する考察

ＳＳＤを適用した上記のプロジェクトは完全な新規開発でなく，既存のシステムを改良する形で行われたので，ボトムアップ的な開発法になりがちであった。このような環境でＳＳＤがどれぐらいの能力を発揮できるのかは興味が持たれたが，ＳＳＤの本来の目的である大規模プロジェクトの開発支援ということではなかったので，そういう意味での十分な評価はできなかった。

上記のプロジェクトにおいてＳＳＤを使用したことによる効果として確認されたのは，

1) 上位のレベルからやるべきことが要求として渡されるので階層的設計が行い易い，
2) 制御の流れなどに関するエラーが設計段階で検出される，
3) 修正による影響範囲の自動検出による副次的なエラーの未然防止，
4) ドキュメントと実際のソースプログラムとのくい違いの回避

などがあった。ボトムアップ的な開発法であったにもかかわらずほぼ期待通りの効果が得られたと思われる。しかしながら記述能力に関しては次のような問題点があった。

1) 論理システム設計と物理（プログラム）設計のギャップを埋めるのに苦労する。
2) データに関しての記述が十分でない。

これらの問題点および他のプロジェクトに適用したときの問題点について次節で詳しく述べる。

5、実用化における問題点

ＳＳＤはソフトウェアのライフサイクルにおいて論理システム設計のレベルから実際のプログラムのコーディングのレベルまでをカバーしている。したがって他の仕様定義技術と比較して従来からすでに機械処理されていた即

分，例えばプログラム管理などをSSDの中に含んでいるので，より一層現行の開発体制との適合性が問題になって来る。なぜならばプログラミング言語の場合と同様に，開発支援のためのツールも，個人にとって使い慣れたツールが，そのツールの良し悪しは別にして，最も使い易いツールであるということがしばしばあるからである。ここではSSD自身が内包している主に技術的な問題と，現行の開発体制との適合性を含めた人間工学的な問題について述べる。

• 技術的な問題

1) 論理システム設計と物理（プログラム）設計のギャップ

SSDを使用してのソフトウェアシステムの開発は論理システム記述からはじめて，すべてのコンポーネントがプログラムとして定義されるまで，順次ブレイクダウンを続けて行く。このときにSDLでのシステムの記述において中心となるのは，TRANSFORMS文と呼ばれる次のような形をした文である。

```
D1:=P(D2,D3,…)
        PURPOSE:"…";
```

この文の意味はD2，D3などのデータからプロセスPによってD1が決定されるということである。したがってこの文が何をするのかという機能を表わす動詞に相当する部分はPで表わされる。システムの記述を分かり易くするにはあるまとまった操作毎に異ったPを設定しなければならない。しかしながらPはいずれプログラムとして定義されなければならないので各操作毎にその操作に対応するプログラムが必要になり，実際のプログラムの数が猛烈に増えてしまうことになる。また逆にプログラムの設計の方に重点を置いてシステムの記述を行うとPの種類が少なくなり，システムの記述が分かりにくくなる上にチェックの厳密性も減ることになる。（もっとも記述の分かり易さだけを考えれば，上記の文に自然語による機能の説明をPURPOSE節でつけることができるので必ずしも分かりにくくはないが。）実際のシステム設計においてはこのあたりの兼ね合いがむずかしく，実用化において最も問題になるところである。データに関しても同様の問題が生じるが，次にそれについて詳しく述べる。

2) データに関する記述

SDLにおいてデータは最終的にプログラム上での変数として定義されなければならない。したがってデータはデータそのものの内容ではなく，データの入れものを表わしている。このために，論理システム記述を行うとき，データの内容の変化を明確に表現するためには，その度毎にデータの入れものを変える必要が生じる。これは次のような例からも明らかであろう。

例1、 LIST:=SORT(LIST);

例2、 SORTED_LIST:=
SORT(LIST);

上の例はあるリストをソートするという操作を記述したものであるが，例1のような表現だと，その後リストが参照された場合，そのリストがソートされているのか，いないのか

は文脈（順序関係）を見ないと分からない。例えのような表現だとそのあたりのことは明確に分かるが，実際のプログラム上で異った2つの変数を用意することは殆んどなく，プログラムと論理システム記述が合わなくなり見合が悪い。このようにデータの記述に関しては物理設計にひきずられ，論理システム記述が行いにくくなっている。

上記のように技術的な問題に関しては殆んどの問題が論理設計と物理設計のギャップに起因している。

次に現行の開発体制との適合性に関した人間工学的な問題点について述べる。

・人間工学的な問題

人間工学的な問題に関してはその殆んどが教育の問題に帰することができると思われるが，SSDを適用して実際に生じた問題を次に列挙する。

1) 階層的設計法

現行の開発体制では通常SSDが要求しているほど厳格な階層的設計法をとっていない。また一般的にソフトウェアの階層はプログラムの制御構造に基いており，SSDのように概念構造に基いてはいないので，SSDの適用の際にこの2つの違いに関して若干の混乱が見られた。

2) 言語的問題

日本語の入出力の困難さによってSSDはSDL中の自然語表現やドキュメントに関してEnglish-Orientedなシステムである。しかしながら要求定義あるいは設計段階のように自由な発想が最も必要な段階でその発想を記録するために英語を使わせることは無理があった。またドキュメントに関しても同様のことが言えた。

3) 厳格なチェック

SSDは設計段階で最大限エラーを検出することを目的としているので，厳格なチェックを行うことは当然であるという考えから，システムの仕様を記述し，プロジェクトデータベースに登録する際に完全性および一貫性のチェックを行い，エラーがある限り登録できないシステムになっている。しかしながらあまりにも厳格なチェックは使い易さの障害になりがちであった。

4) TSS専用の開発法

欧米人に比べ日本人は子供の頃からタイプライタになじんでいないため，端末の前に座ってソフトウェアシステムの設計を考えることが非常に苦手であった。

6、今後の計画

上で述べた実用化における種々の問題点を克服し，SSDをさらに実用的なものに近ずけるために現在考えている2，3の解決法について述べる。

1) 論理システム設計と物理設計のギャップについて

論理システム記述のもとになる要素は操作（プロセス／オペレーション）とその操作の対象となる論理データであると思われる。したがって

論理システム記述はこれらの要素間の関係を定義することで記述されるはずである。しかしながらこれらの操作あるいは論理データは必ずしも実際のプログラムやデータに対応しない。ここに論理システム設計と物理(プログラム)設計の間にギャップが存在するわけである。一般に論理的要素と物理的要素の間にはれ対れの写像が行われていると思われる。SSDではこれを無理に1対1に対応させているわけで,そこに書きにくさが表れて来る。したがって論理システム記述と物理設計の記述を明確に分け,その上で論理要素と物理要素との対応を記述させる方向で対処しようと考えている。

2) 日本語化

ソフトウェアの設計段階は試行錯誤の連続であるといえる。このような段階を英語的な仕様やドキュメントを見ながら設計して行くということは,あまりにも言語的ハンデキャップが大きいと言える。この問題を解決するためにカナ漢字変換入力[3]をもとにした日本語SDLを考えている。これは単にキーワードを日本語化するだけでなく,ある程度の意味解析も導入し,表現に柔軟性を持たしたいと思っている。

7. 謝辞

最後にSSDをプロジェクトに適用し,その際にデータの収集や評価を行なって下さった千葉氏に感謝いたします。

8. 参考文献

1) H. Uchida, "SSD: Software for Structured Development", FUJITSU Scientific and Technical Journal Vol.14, No.2 pp 129-142, 1978.6

2) 内田, "ソフトウェア開発支援システムSSD", ソフトウェア工学研究会資料5-2, 1978.1

3) 内田, 杉山, "カナ漢字変換入力による和文テキストエディタ", 情報処理学会第19回全国大会, pp 763-764, 1978.8

9. 附録

ＳＳＤの適用プロジェクトでのＳＤＬによる設計仕様の例を以下に示す。

```
DESIGN JSEISSI-ROUTINE (PROC );
CONSISTS OF JSEISSI(PROC), ISLDCBIN(DATA),
            ISLBUFIN(DATA), ISLBUFOT(DATA);
ENTERS TO JSEISSI          PURPOSE:"TO OPEN UNLOADED DATA-SET OF INPUT
                                    AND GET UNLOADED RECORDS AND THEN
                                    PUT THIS RECORDS TO INDEXED SEQUE-
                                    TIAL DATA-SET OF OUTPUT";
     IF FIRST ENTRY THEN DO;
        ISDCBAR1:=(ISLDCBIN, ISDDNAME)
                           PURPOSE:"TO MOVE DCB AND DD-NAME OF INPUT
                                    AND OPEN SEQUENTIAL DATA-SET";
        READS ISLBUFIN PURPOSE:"TO GET POINTER OF INPUT BUFFER";
        ISDOUBL2:=(ISLBUFIN)
                           PURPOSE:"TO SET INFORMATION FROM READ RECORD";
     END;
     ELSE DO;
        REPEAT NOT END OF DATA;
          IF NEED TO READ OF NEXT RECORD THEN DO;
             READS ISLBUFIN
                           PURPOSE:"TO GET RECORDS FROM INPUT";
             ISDOUBL2:=(ISLBUFIN)
                           PURPOSE:"TO SET INFORMATION FROM READ RECORD";
          END;
          IF OUTPUT INDEXED SEQUENTIAL DATA-SET IS NOT OPENED
          THEN DO;
             ISQISAM:=(ISLBUFIN)
                           PURPOSE:"TO EDIT OF READ ORIGINAL DCB
                                    AND THEN SAVE TO COMMON WORK AREA";
             LEAVE;
          END;
          ELSE DO;
             ISLBUFOT:=(ISLBUFIN)
                           PURPOSE:"TO EDIT OF READ RECORDS AND MOVE
                                    TO OUTPUT BUFFER";
             WRITES ISLBUFOT
                           PURPOSE:"TO PUT INDEXED SEQUENTIAL RECORDS";
          END;
        END;
     END;
     EXITS;
END DESIGN;

DESIGN JSEISSO-ROUTINE (PROC );
CONSISTS OF JSEISSO(PROC), ISUDCBOT(DATA),
            ISUBUFOT(DATA);
ENTERS TO JSEISSO          PURPOSE:"TO MAKE UNLOADED DATA-SET";
     IF THE OUTPUT SEQ. DATA-SET HAS TO OPEN THEN DO;
        ISDCBAR2:=(ISUDCBOT, ISDDNAME)
                           PURPOSE:"TO MOVE DCB AND DDNAME OF OUTPUT";
        ISDOUBL2:=(ISDOUBL1)
                           PURPOSE:"TO GET ATTRIBUTE OF INDEXED SEQ.
                                   DATA-SET AND INTIALIZE THIS AREA";
        READS ISUBUFOT
                           PURPOSE:"TO GET BUFFER PTR. OF OUTPUT";
     END;
     ELSE DO;
        IF NOT END OF DATA THEN DO;
           ISUBUFOT:=(ISRECPTR)
                           PURPOSE:"TO GET PTR. OF INDEXED SEQ. RECORDS
                                   AND MOVE TO OUTPUT BUFFER";
           WRITES ISUBUFOT
                           PURPOSE:"TO PUT OF INDEXED SEQ. RECORDS";
        END;
     END;
     EXITS;
END DESIGN;
```