ソフトウェア工学 13－3
（1980. 3. 17）

ＳＤＤ：ソフトウェアの図形表示技術

神田　泰典　　杉本　正勝
富士通株式会社

１．はじめに

ＳＤＤ（Ｓｏｆｔｗａｒｅ　Ｄｉａｇｒａｍ　Ｄｅｓｃｒｉｐｔｉｏｎ）：ソフトウェアの図形表示技術とは次のことを意味する。

（１）２次元の図面を用いるソフトウェアの新しいドキュメント形式

（２）２次元の図面に基ずくソフトウェアの新しい開発法

ＳＤＤの目的は，次の通りである。

（１）高品質のソフトウェアを設計・作成する

（２）ソフトウェアの生産性・保守性の向上

これらの目的には，読み易く，理解し易いドキュメントの方法が必須になっている。[9),(10)]

ソフトウェアは，「人間の考えを記述したもの」である。それ故，ソフトウェアの設計者・作成者の発想なり考え方が自然の形で盛り込めるような構造化されたドキュメントが必要となる。[(1),(2),(3)]

ソフトウェアは，２次元の構造化された図面として表示すべきである。これにより，視覚を十分に利用出来き，図面を介して設計者・作成者の間で正確で，速やかな情報交換ができる。

２．ＳＤＤの基本思想

人間の視覚の役割を重視して，ソフトウェアのドキュメントを２次元的に表示しようとする方向が本質的であると我々は考えている。ストラクチャドプログラミングにおける字下りの工夫（Ｉｎｄｅｎｔａｔｉｏｎ），ＩＢＭ社の考案であるＨＩＰＯチャートはそのはしりであると考えられる。

人間の思考とか，論理判断能力は視覚と大いに関係しているということも医学，心理学の研究から報告されている。[(6),(12)]

我々は２次元のソフトウェアの記述法に，より多くの研究が必要であると考える。

機械の設計には機械図面，建築の設計には建築図面，コンピュータのハードウェアの論理設計には論理回路図があるように，ソフトウェアの設計・作成・保守にも図面が必要である。[(7),(8)]

図１のＩは，現在のプログラミング手法を示す。トップダウンの段階的な詳細化を行って，最終的には高級プログラミング言語で書いたソースプログラムとなる。

このソースプログラムは，ストラクチャードプログラミングの規則に合うように，順列，ｉｆ　ｔｈｅｎ　ｅｌｓｅ，ｄｏからなっている。

上記の手法では「文章による，ソフトウェアの記述」が基本となっている。

文章は１次元の表現であり，言葉が連らなっていればよい訳である。一方，ソフトウェアの分野ではブロック構造，ネスティング等，プログラムの構造という概念が使われており，これらを分り易くするためにＩｎｄｅｎｔａｔｉｏｎの手法が導入され

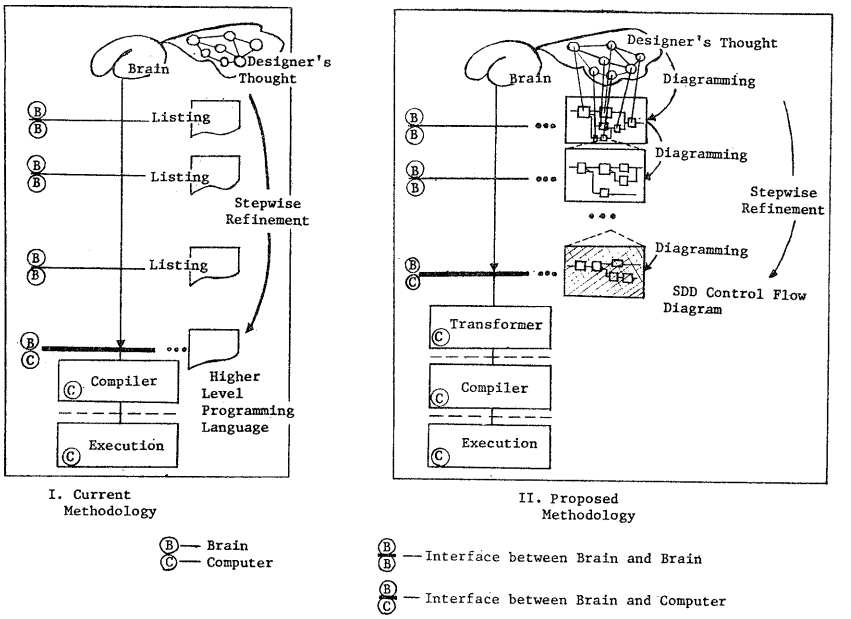


Fig. 1 Programming Methodology, Current Methodology and Proposed Methodology

ている。これは，１次元の言葉に，視覚に訴える構造を取り入れる方向ではあるが中途半端な手段にすぎないと思う。(6)

我々は，図１のＩＩに示すソフトウェアの開発手法を提案する。即ち，ソフトウェアの設計・作成の各段階で使用するドキュメントが図面の形式をとるものである。(4),(11)

コンピュータのハードウェアの論理設計は純粋に論理的思考作業であるが２０年以上前から論理図面が用いられ，図２に示すようなＤＡシステムで図面が管理されて，多くの成果を上げている。(5) この事実はＳＤＤがうまく行くであろうことの裏付けとなっている。また，最近では図形処理システムが比較的手近に利用できるようになってきており，ソフトウェアの開発作業に図面を利用するアプローチのが広まることが十分考えられる。

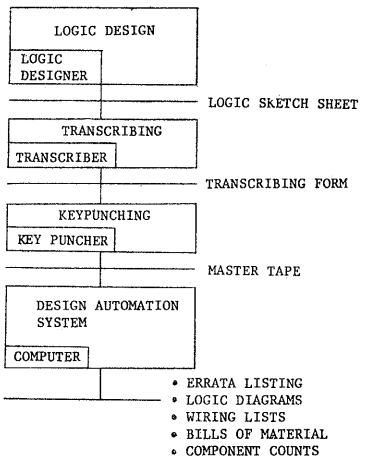


Fig. 2 Processing of Logic Diagrams in Logic Design Automation Systems

３．ＳＤＤのダイヤグラム

ＳＤＤの制御フローダイヤグラム

我々は，まずプログラム記述用のソースプログラム・リスティングに注目し，プログラムの理解し易い記述形式を検討した。そして，ＳＤＤの制御フローダイヤグラムを提案する。（図１のＩＩのＣｏｎｔｒｏｌ Ｆｌｏｗ Ｄｉａｇｒａｍである）

ＳＤＤの制御フローダイヤグラムは図３に示すような基本的な要素，ボックスをプログラム実行の順に直線で結合した図面である。

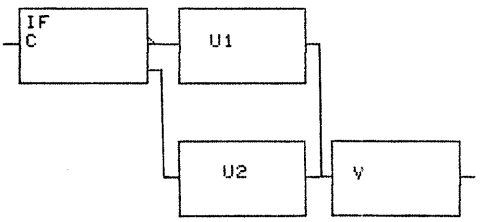
各ボックスは，高級プログラミング言語の式（Ｅｘｐｒｅｓｓｉｏｎ），または文（Ｓｔａｔｅｍｅｎｔ）を記入する。

式または文は，各ボックスに１つだけ置くこともあるし，２つ以上置くこともある。

ボックスには，図３に示す種類がある。

制御は，図の左上から始まり，左から右へ，上から下へと移動する。

ＩＦボックスは，次の形式も利用できる。



ＩＦボックスに上記の２つの形式があるので，図面上で論理が自然の形で書ける。

また，ＳＤＤの基本的な考え方は論理のまとまりを分り易く表すことなので，次の書き方をしてもよい。

１つのボックスの中で，ＤＯ文を利用

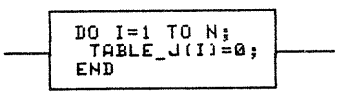


図４にはボックスの各記入欄の意味ずけを示す。

これらの図形とシンボルは，ハードウェアの設計で用いている論理素子のグラフィック・シンボル，論理図面を注意深く検討して定めた。

なお，注釈文も図面上で構造化されている。

１つのボックス全体に対する注釈はボックスの上に書き，ボックス内の各々の処理に対する注釈はボックスの中に書く。

図５にＳＤＤ制御フローダイヤグラムの実例を示す。

ＳＤＤの制御フローダイヤグラムは通常のフローチャートとソースプログラム・リスティングとの結合である。

従来の方法との大きな差はソースプログラム・リスティングは使用しないことである。

ＳＤＤの図面だけが，コンピュータに対いする入力である。

（ｉ） ＳＤＤの図面は，人間の視覚の役割を重視して，論理が分り易いように構造化してある。

（ａ）制御フローの向きを統一

（ｂ）シンボルを用いる

（ｃ）ボックスの内部，外部の各部分の記述内容に規則を設けた

（ｉｉ）図面はコンピュータによつて処理されている。

図面はＳＤＤサポートシステムと呼ぶシステムに入力され，編集され，システムにより表示され，プリントされる。

ＳＤＤサポートシステムは，図面の保守，

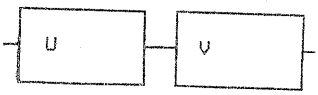
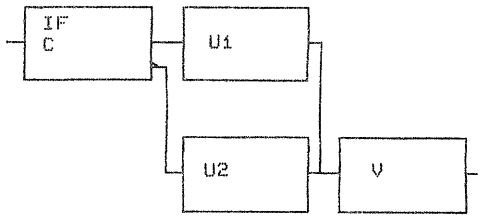
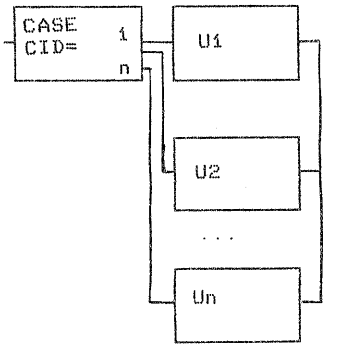
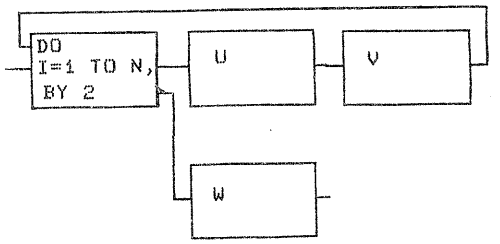
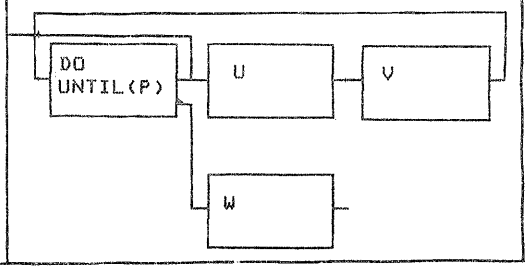
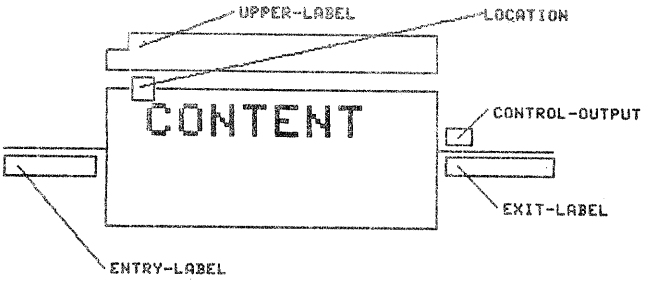
Control Structure	Representaion in Higher Level Programming Language	Representaion in SDD
Sequential Execution	U; V; ...	Simple Box:  （注）U，Vは複数個の文を書いてよい
IF	IF C THEN U1; ELSE U2; V;	IF Box: 
Case	SELECT(CID); WHEN(CID=1) U1; WHEN(CID=2) U2; ... WHEN(CID=n) Un; END;	Case Box: 
Repetition	DO I=1 TO N, BY 2; U; V; END; W;	DO Box: 
	DO UNTIL(P); U; V; END; W;	DO UNTIL Box: 

図 3　　ＳＤＤダイヤグラムの書き方



記入項目	記入内容の説明
① UPPER－LABEL （屋根ラベル）	ＥＮＴＲＹボックスを除いて，ボックスの内味に対する注釈（コメント）を記入する。必要がない場合には記入を省略する。 ＥＮＴＲＹボックスの場合は，手続名（入口名）を記入する。
② LOCATION （ボックス位置， 座標）	そのボックスの図面上の位置を２桁で記入する。
③ ＥＮＴＲＹ－LABEL （入口ラベル）	そのボックスの入口に関する注釈を記入する。必要のない場合は記入を省略する。
④ ＥＸＩＴ－LABEL （出口ラベル）	そのボックスの出口に関する注釈を記入する。必要のない場合は記入を省略する。
⑤ ＣＯＮＴＲＯＬ－ ＯＵＴＰＵＴ （制御の出口）	次に制御が移動すボックスの位置を記入する。
⑥ ＣＯＮＴＥＮＴ （内容）	（１） ＳＩＭＰＬＥボックスの場合 処理内容，宣言内容，説明文などを記入する。 （２） ＩＦボックス，ＣＡＳＥボックス，ＤＯボックスの場合 判定条件，説明文を記入する。 （３） ＥＮＴＲＹボックスの場合 パラメータ・リスト，パラメータの宣言内容，説明を記入する。

図 4 ボックスの記入欄の説明

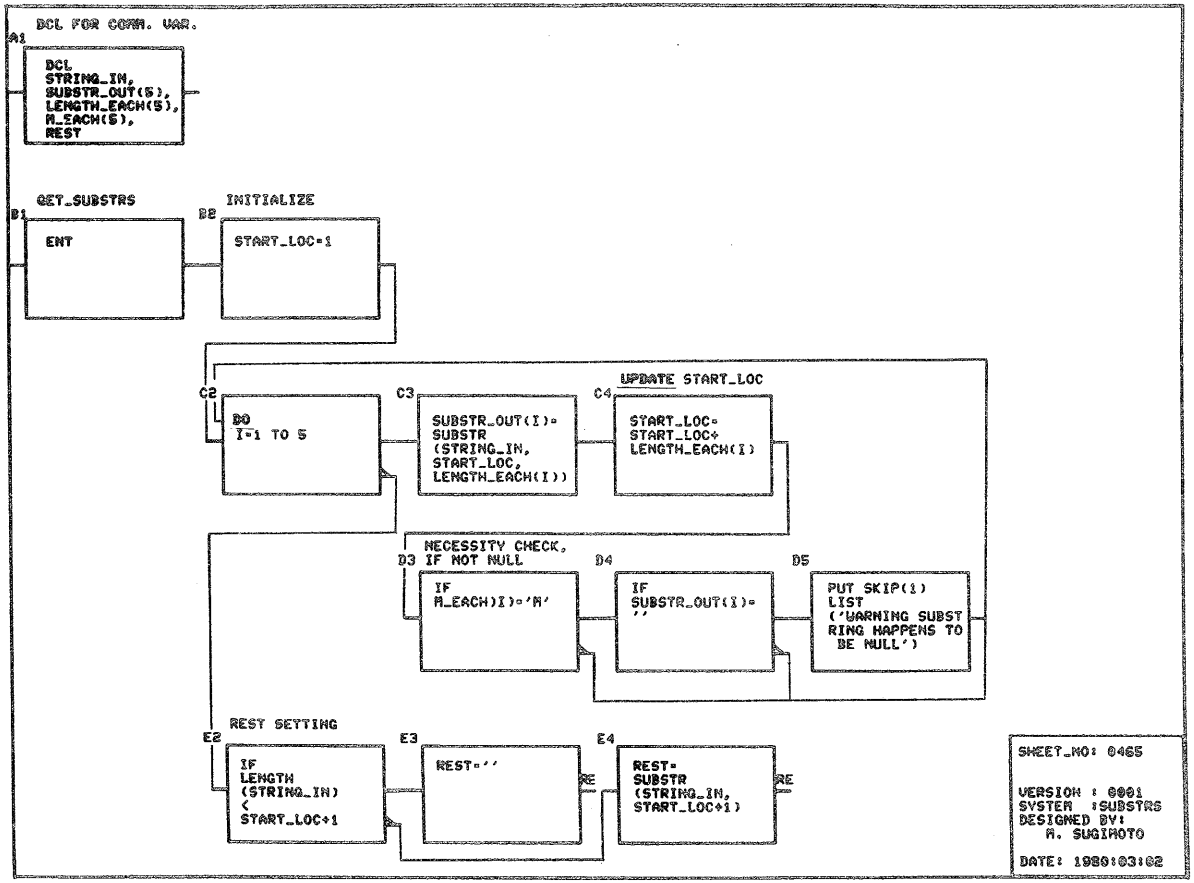


図5　　ＳＤＤダイヤグラムの例

図面のチェック，実行を行う。

ＳＤＤの図面はトランスフォーマによって変換され，実行される。

４．ＳＤＤの有効性

ＳＤＤの制御フローダイヤグラムはプログラムの制御機構が明確に表示できることから，プログラムのアルゴリズムを明確に表示することができる。また，プログラムのアルゴリズムを直観的に理解するのに役立つ。

我々の実験では，単純なプログラムについて，次の２つの形式のプログラムを用意した。

（ｉ）ストラクチャード・ソースプログラム・リスティング表現

（ｉｉ）ＳＤＤ表現

プログラムの制御の順序に関する問題とプログラムの中にバグを意図的に混入させてそれを摘出する問題の所要時間を比較したところ図６に示すような結果を得た。

これは，社内のソフトウェア作成のエクスパート１０名に対して，図５に示すようなＳＤＤの図面を用いて行った。

この結果からもＳＤＤの図面の優れていることが分る。

ＳＤＤの図面はＳＤＤサポートシステムの開発とマイクロプロセッサ用プログラムの開発に利用しているが，使用している者の感想は次のようにＳＤＤの優れていることを指摘している。

- ＳＤＤの図面はプログラムを作成する時に，論理が良く分り，プログラムの論理に無駄がなくなり，すっきりとした論理が組める。
- 自分が作成したプログラムでも，１ケ月とか時間がたつと，その詳細が分らなくなることが従来はあったが，ＳＤＤでは，容易に論理が分る。

	正解率％	作業時間（平均分）
通常のリスティング	８９	８
ＳＤＤ	１００	５

（ａ）制御の順序の問題

	デバッグ所要時間（分）
通常のリスティング	８．５
ＳＤＤ	６．５

（ｂ）　デバッグの問題

図６　通常のリスティングとＳＤＤ図面の比較

プログラムの構造を表すのにＤＯ；・・・ＥＮＤ；やＩＦ　ＴＨＥＮ　ＥＬＳＥを使用するやり方にはロスが多いということがＳＤＤの開発を進め中で実感として分ってきた。

ＳＤＤサポートシステムでＳＤＤの図面を扱う時に，最も問題とされるのは図面の入力であるが，これはＴＳＳ端末で対話型で行えば容易に行える。

5．おわりに

ソフトウェアの図形表示技術は設計者の創造的な思考を助け，設計・作成・保守の各段階で正確な情報交換に大いに役立つと確信している。

この技術を中心にソフトウェアのＤＡ（Ｄｅｓｉｇｎ　Ａｕｔｏｍａｔｉｏｎ）システムを作成中である。

また，データフローダイヤグラム等のソフトウェア作成に役立つダイヤグラムを検討して行きたい。

謝辞

いつも御指導いただいている開発事業部佐藤部長に深謝いたします。

参考文献

（１）コンピュータにおける産業革命
神田　情報処理学会計算機アーキテクチャ研究会，１９７７，１１

（２）ハードウェアから見たソフトウェア
神田　昭和５３年度電子通信学会総合大会

（３）ソフトウェアにおける日本語の役割
神田　杉本
情報処理学会ソフトウェア工学研究会資料　１９７８，１

（４）ＳＤＤ：ソフトウェアの図形・図表による記述　神田　杉本　長田
昭和５３年度情報処理学会第１９回全国大会

(5) M. Kloomok, et. al., The Recording, Checking, and Printing of Logoic Diagram, Proc. EJCC, Dec. 1958

(6) D. C. Smith, PYGMALION: A Creative Programming Environment, Report STAN-CS-75-499, Stanford Artificial Intelligence Laboratory, June, 1975

(7) R. Wirry, Dimensional Flowcharting, Software-Practice and Experimence Vol. 7, 1977

(8) Y. Chu, Introducing Software Blueprint, Proc. COMSAC, 1977

(9) R. Yeh, Current Trends in Programming Methodology, Vol. I, Software Specification and Desigin, Prentice-hall, 1977

(10) P. Nyborg et. al., Information Processing in the United States, A Quantitative Summary, AFIPS, 1975

(11) Y. Kanda and M. Sugimoto, SDD: Software Diagram Description, 3rd UJCC 1978

(12) W. J. Tracz, Computer Programming and the Human Thought Process, Software-Practice and Experimence, Vol. 9, 1979