ソフトウェア工学　56－1
（１９８７．９．９）

ソフトウエアの要求仕様作成から設計仕様作成までの標準化
ＳＡ／ＳＤ手法と構造図の定量的な評価との組み合わせ

立田種宏
ソニー・テクトロニクス㈱　ロジック設計システム部

構造化分析（ＳＡ）手法に基づいてソフトウエアの要求仕様の記述を行なうことで、データフロー図を中心とした要求仕様を作成できる。そして、構造化設計（ＳＤ）手法における変換分析を用いることで、ＳＡ手法により作成されたデータフロー図から、要求仕様を満足する設計仕様としての大まかな構造図が作成できる。ここでは、この二つの手法と、構造図の設計の信頼性について定量的に測定する試みとを組み合わせて用いることで、信頼性の高い設計仕様を効率良く求める方法を提案する。

Standarization of Specifying the Software Requirement and Design
/ (in Japanese)
Combinational Usage of Structured Methodologies with Software Metrics

Tanehiro Tatsuta
Logic Design System Division, Sony/Tektronix Corporation
5-9-31, Kitashinagawa, shinagawa, Tokyo, 141 Japan

By specifying the software requirement with Structured Analysis method, we can create the Data Flow Diagrams. And carrying out Transform Analysis, the Data Flow Diagrams will be converted to the structure charts, major product of Structured Design method. We propose the effective ways to make high quality structure charts by combining these two methods with a quantitative evaluation.

１．はじめに

ソフトウエアの要求仕様を記述する際に用いられる手法に、米国を中心に７０年代後半から広く知られるようになった構造化分析(ＳＡ)手法があり 1)､2)、ソフトウエアの設計を行なう際に用いられる手法に、７０年代前半から広く用いられている構造化設計(ＳＤ)手法がある 3)､4)。ＳＡ手法を使うことで、データフロー図を中心とした要求仕様が作成され、ＳＤ手法を使うことで、設計仕様としての構造図が作成される。このＳＤ手法には、データフロー図から構造図を導き出すアイデアが含まれている。このアイデアは８０年代に入ると、変換分析として体系付けられ紹介されている 5)。この変換分析により、ＳＡ手法により作成されたデータフロー図を大まかな構造図にうまく変換できる。このため、一から構造図を作成する手間が省け、人の手によって書き加えられる箇所が減少し、その結果、生産性と信頼性の向上が期待できる。これらの一連の手法は、ＳＡ／ＳＤ手法とも呼ばれ、要求仕様の作成から設計仕様の作成に至るまでをスムーズにつなぐことの出来る有効な手段として、米国を中心に広く受け入れられている。

さて、ＳＤ手法による最終出力結果としての設計仕様(構造図)の評価は、構造図の中のモジュールの強度や結合度等により定性的に行なわれている。そのためその評価尺度は、その設計を行なった設計者の能力、経験、信用等に大きく依存することになる。そこで、ＳＡ／ＳＤ手法と、構造図の信頼性について定量的に測定する方法 6)とを組み合わせて用いることで、より信頼性の高いソフトウエアを効率良く作成する方法について述べる。

２．ＳＡ手法

ＳＡ手法により、先ず概観図と呼ばれるデータフロー図が記述される(以下実例として、駅でよく見かける切符の自動販売機システムをターゲットとする要求仕様を記述しながら解説する)。概観図では、記述の対象外とのインターフェイスが明確にされる(図１)。次に、最上階のデータフロー図が記述され(図２)、そのデータフロー図上の各プロセスは、１階層下の各データフロー図に分解され詳細化される(図３)。この分解は、プロセスが数行のテキスト等で書き表せる迄繰り返される。この分解において、１階層下のデータフロー図は、それに対応した１階層上のデータフロー図のプロセスと同じ名前が付けられ、同様にデータフロー図に入出力するデータフローは、それに対応した１階層上のデータフロー図のプロセスに入出力するデータフローと同じ名前が付けられる。又、データフロー図やプロセスには、「そのダイアグラムがどのプロセスに対して詳細化されたものであるのか」や「そのデータフロー図がどの階層に位置するか」が分かるような番号が付けられている。このＳＡ手法独特の分解の際の平衡性や番号の付け方により、仕様書の様々な機械的検証を可能にしている。

３．ＳＤ手法

ＳＤ手法を用いて、設計仕様としての構造図が記述される。特殊な構成要素を除けば、①モジュール同士の結合状態を示す矢印、②モジュールが呼ばれる度に「何を実行するか」を一行の文章にしたモジュール名、③モジュールの入出力情報としてのデータ結合子(↕)及びフラグ(↕)、のみが構造図に記述され、それ以

外のモジュールの中身についての情報は一切記述されない。
ＳＤ手法における変換分析を用いることで、要求仕様としてのデータフロー図か

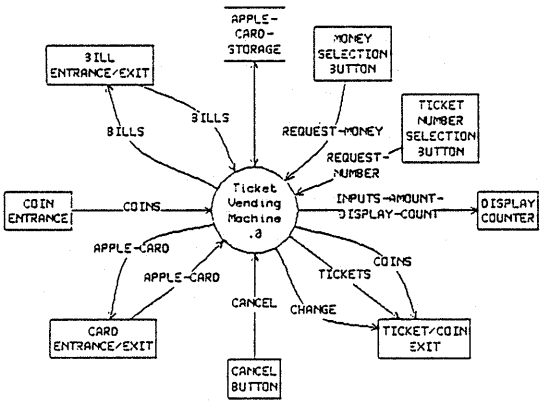

図１　概観図。特別なデータフロー図である概観図には、記述の対象となるターゲット・システム（丸で示された部分）とその外部環境（四角で示された部分）との間に流れるすべてのデータの流れが記述される。

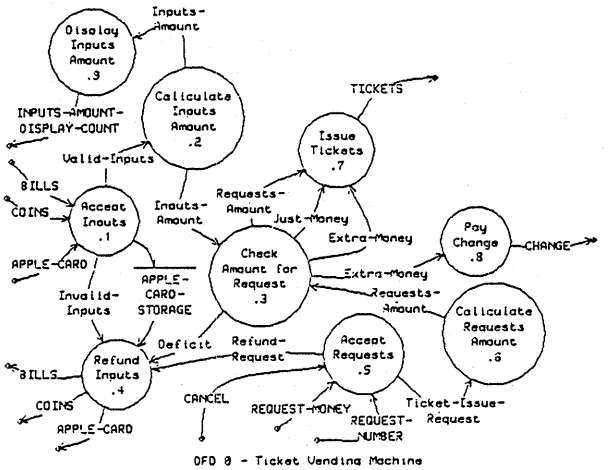

図２　最上階層（１階層目）のデータフロー図。図１の概観図の丸の部分が分解され詳細化されたもの。

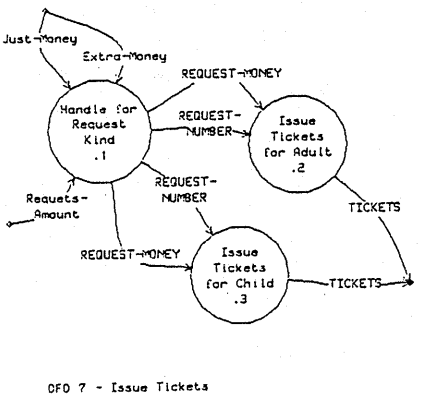

図３　２階層目のデータフロー図。図１のデータフロー図の１から９までの各プロセスは、それぞれが１階層下のデータフロー図に分解され詳細化される。この図は、図１のプロセス７が分解され詳細化されたもの。

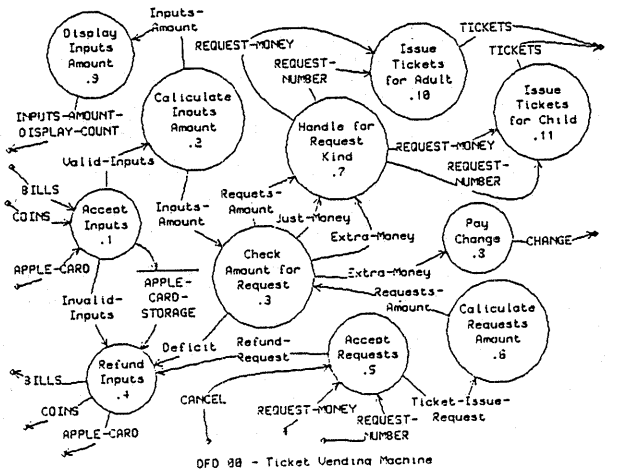

図４　図２と図３をひとまとめにしたデータフロー図。紙面の制約上、２階層目のデータフロー図の中から、代表として図３のみを使った。この場合、データフロー図の中央部分に位置する３つのプロセス（プロセス番号２、３、６）をひとまとめにして中央変換部分とするのが妥当である。

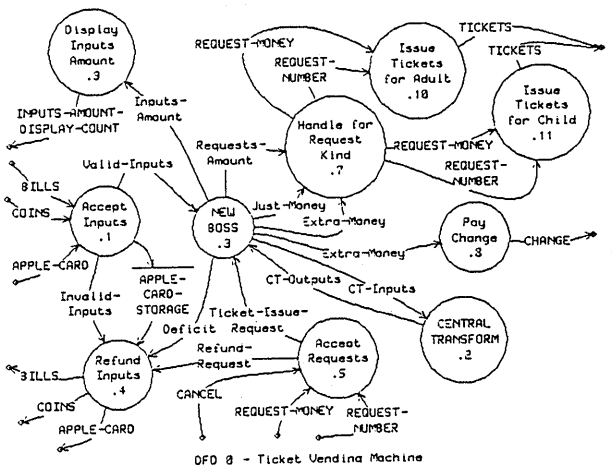

図５　図４のデータフロー図の外部からボスを導入。図４の中央変換部分の中では、データの流れの交通整理的な役割をしている点で、プロセス３が最もボスの候補として有力だが、データの流れを加工する役割も持っているので、プロセス３をボスにするのは止めて、新たに外部からボスを導入した。

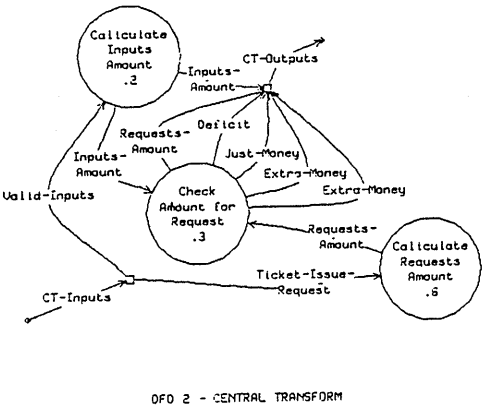

図６　図４の中央変換部分（プロセス番号２、３、６）を一まとめにしたデータフロー図。これは、図５のデータフロー図のプロセス２が分解され詳細化されたもの。

ら、大まかな構造図を求めることができる。この変換分析の元になるアイデアは、７０年始めに紹介され、後に本になった 3)。７０年後半に紹介されたものもある 7),8)。しかしその何れもが、変換分析について余り詳しく述べていないため、８０年代に入ってからこの変換分析について詳しく書かれた本が出版されることになった 9)。変換分析を行なう際には、先ず要求仕様としてのデータフロー図を、設計に利用可能な入力データとしてのデータフロー図に作り直してから、このデータフロー図の中央変換部分を決定する(図４)。次に、構造図に変換されたときの階層構造の頂点に位置するに相応しいモジュール(モジュールのボス)の候補が、中央変換部分の中にあればそこから選び、なければ外部から導入する(図５、６)。最後に、残りのプロセスをボスの下にぶら下げるようにし、幾つかの機械的な修正を加えることにより、大まかな構造図が出来上がる(図７)。

４．構造図の複雑度を定量的に測定するソフトウエア・メトリックス

構造図の複雑度を定量的に測定するソフトウエア・メトリックスがある 6)。このメトリックスでは、モジュールの再帰呼び出し、多重呼び出し、及び同階層でのモジュール間の呼び出しが一切行なわれない木構造を形成するものを、最も理想的な構造図としている。モジュール間の結合状態は、モジュールを呼び出している数に依存し 10)、プログラムの複雑度は、プログラム構造のみに依存する 11)とされている。そのことから、階層構造におけるモジュール間の呼び出しの数

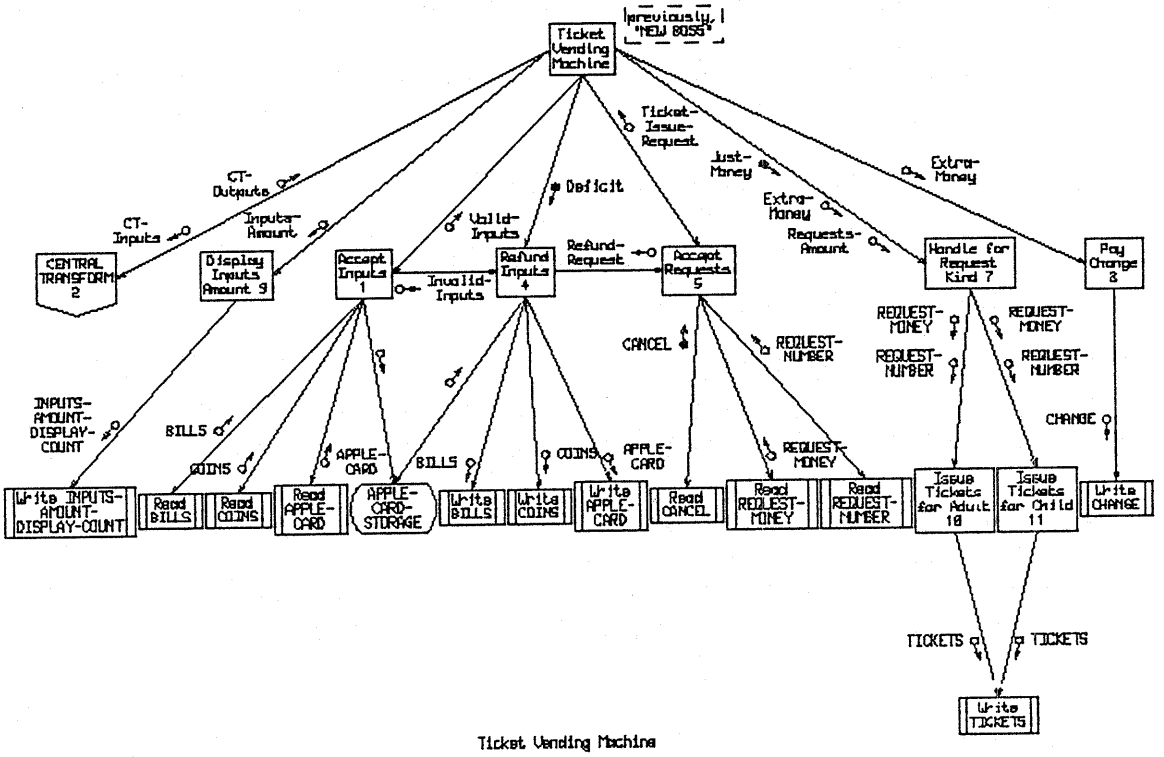

図７　図５のデータフロー図から、変換分析により導き出した大まかな構造図。

に着目したこのメトリックスは、有効であると言える。メトリックスとして利用出来る主な算出式は、次の通りである。

表１　構造図（図７）の信頼性を定量的に測定。この表は、変換分析によって求められた構造図（図７）のR，（１階層目の木構造純度）が、急に増大していることを示している。そこで、この１階層目にエラーが含まれていることが推定される。

	要素の数	呼び出しの数	複雑度	木構造純度	レベル純度
Level	Items	Calls	Complexity	Tree Impurity	Level Impurity
0	1	0	0	0.00	0.00
1	6	8	2	0.25	0.25
2	14	15	3	0.13	0.06
3	1	2	4	0.16	0.50
-----	-----	-----	-----	----	----
Total	22	25	9	0.54	0.81
Avg	7.3	8.3	3.0	0.18	0.27

- T_i に対する A_i の増加度(複雑度) $C_i = A_i - T_i$
- 理想的な木構造に対する累積偏差値(木構造純度) $R_i = 1 - T_i / A_i = C_i / A_i$
- 理想的な木構造に対する同一階層内における偏差値(レベル純度) $D_i = 1 - (T_i - T_{i-1}) / (A_i - A_{i-1}) = (C_i - C_{i-1}) / (A_i - A_{i-1})$

但し、N_i ＝0階層～i階層の全モジュール数
A_i ＝0階層～i階層の全モジュール呼び出し数
T_i ($= N_i - 1$)＝理想的な木構造での0階層～i階層の全モジュール呼び出し数

実際に、あるプロジェクトで検証した結果、プログラム・エラーの数とモジュールの複雑度との間には、高い相関関係が見られた 6)。このメトリックスを使って測定された値から得られる評価としては、次のようなものがある。

- C_i が急激に増大している階層は、その階層でのエラーの可能性が高いことを示唆している。
- R_i と D_i は、0～1の範囲内にあり(理想的な木構造の時は0)、その値が大きい程、ファン・インや結合度が大きく、理想的な木構造からかけ離れていることを示している。

5．SA／SD手法とソフトウエア・メトリックスとの組み合わせ

SD手法により構造図を作成した後、先程のソフトウエア・メトリックスを使って複雑度を測定する。その結果、木構造純度やレベル純度がある一定の値を越えたなら、その構造図の改善を検討する。そして、修正したほうが良いと思われる箇所が出てきた場合、構造図に直接変更を加えるよりも、この構造図に変換された元のデータフロー図への修正を試みる。この例では、木構造純度が1階層目で急激に増大している(表1)。それは、同階層間でのモジュールの呼び出しが原因している。そこで、その原因箇所に対応した元のデータフロー図（図2）において、構造図の複雑度の原因になっている箇所を改善する(CANCELをプロセス4の入力とすることで、Refund-Requestが不要となる）。この修正作業は、同時にデータフロー図自身の改善にもつながる。その他にも、測定結果から、構造図の改善すべき箇所があれば、それに対応した元のデータフロー図に修正

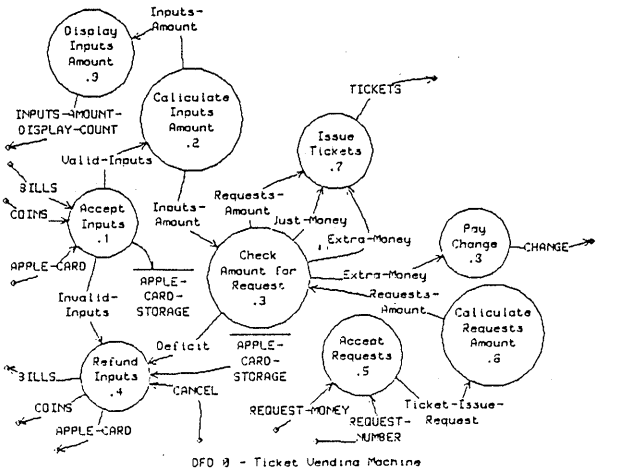

図8　データフロー図（図2）を改善する。

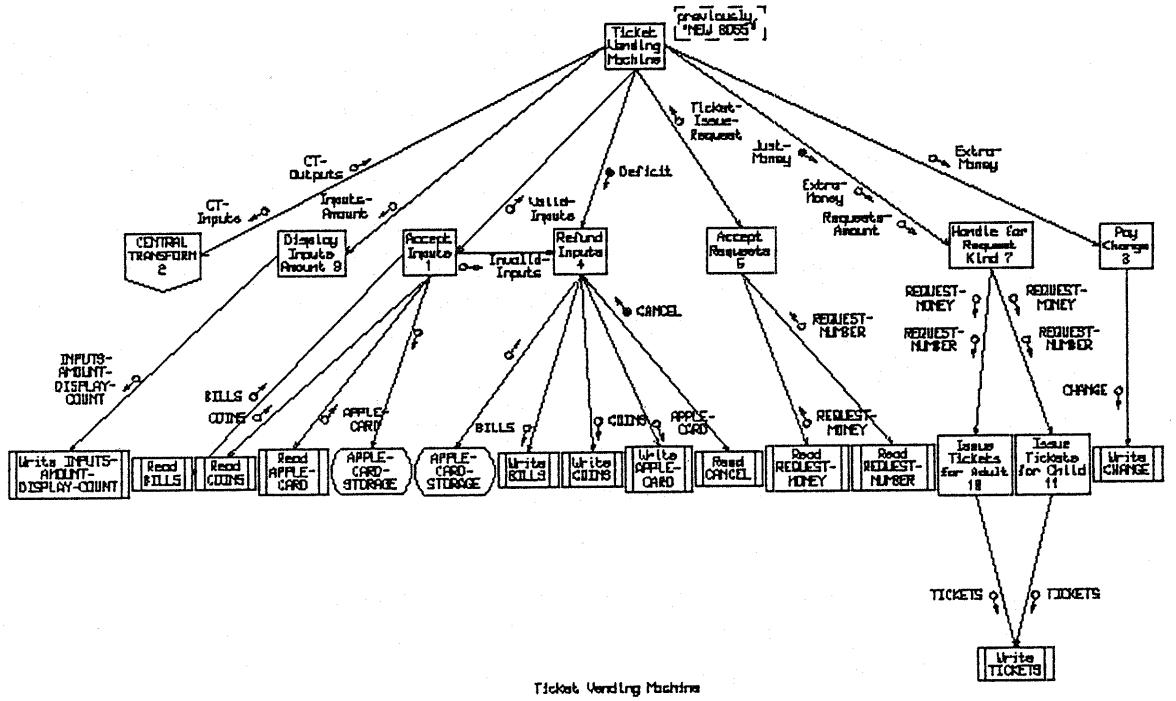

図９　改善されたデータフロー図（図８）を、変換分析により大まかな構造図に変換。同様に、中央変換部分も大まかな構造図に変換できる（図１０）

を加える（図８）。そして変換分析により、今修正したデータフロー図から、改善された構造図を導きだす(図９、表２)。このようにして求められた構造図は、一部のデータ結合子をフラグに変更し(Deficit,CANCEL,Just-Money)、最上層のモジュール名をこの構造図全体を示す名前に変更するだけで、そのまま設計仕様としても十分利用可能になる。

表２　構造図（図９）の信頼性を定量的に測定。表１と比較すると、特に木構造純度に改善が見られる。

	要素の数	呼び出しの数	複雑度	木構造純度	レベル純度
Level	Items	Calls	Complexity	Tree Impurity	Level Impurity
0	1	0	0	0.00	0.00
1	6	7	1	0.14	0.14
2	15	15	1	0.04	0.00
3	1	2	2	0.08	0.50
Total	23	24	4	0.26	0.64
Avg	7.7	8.0	1.3	0.09	0.21

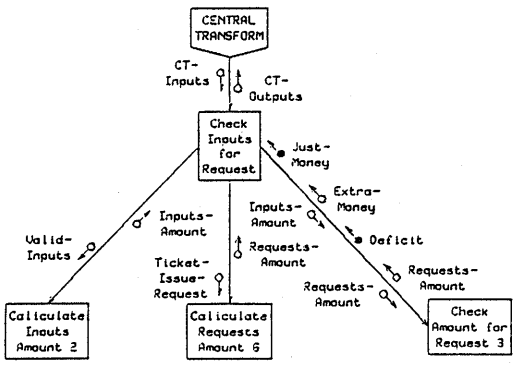

図１０　中央変換部分を大まかな構造図に変換。図６のデータフロー図が、変換分析によって変換された大まかな構造図。

６．おわりに

ここで述べたＳＡ手法は、リアルタイムシステムの制御やタイミングの記述を不得手としている。しかし最近、このＳＡ手法の欠点を改善した手法(ワードの手法やハトレーの手法)がリアルタイムＳＡ手法として登場してきた12)。今後は、このリアルタイムＳＡ手法による要求仕様を，効果的に構造図に変換する方法について検討する予定である。最後に、実例として載せた図表は、すべて当社ツールの支援により作成したものであることを付け加えておく。

参考文献

1) T. DeMarco, Structured Analysis and System Specification,Prentice-Hall/Yourdon Press, 1978.（構造化分析とシステム仕様、高梨・黒田訳、日経マグロウヒル社、1986.）
2) C. Gane and T. Sarson,Structured Systems Analysis: tools and techniques,New York: Improved System Technologies Inc.,1977.
3) W.P.Stevens, G.J.Myers, and L.L.Constantine, Structured Design,IBM Systems Journal,1974,Vol.13,No.2,pp.115-139.
4) E.Yourdon, and L.L. Constantine, Structured Design, Prentice-Hall/Yourdon Press,1978.
5) M. Page-Jones, The Practical Guide to Structured Systems Design, Prentice-Hall/Yourdon Press,1980.
6) B.H.Yin and J.W.Winchester, The Establishment and Use of Measures to Evaluate The Quality of Software Design, Proc. ACM Software Quality Workshop,San Diego,CA,November 1978,pp.45-51.
7) G.J.Myers,Composite/Structured Design,Van Nostrand Rheinhold,1978.
8) E.Yourdon, Classics in Software Engineering, Prentice-Hall/Yourdon Press,1979,pp.221-224.
9) M.Page-Jones, unpublished private letter, Wayland Systems Development Inc.,Federal Way,WA,June 26th,1987.
10) N.F.Schneidewind, Modularity Considerations in Real Time Operating System Structures,COMPSAC 1977,pp.397-403.
11) T. J. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering,Vol.SE-2,No.4,December 1976,pp.308-320.
12) 立田種宏、蘇った構造化分析手法、日経コンピュータ、1987年7月20日号.