ソフトウェア工学 58－1
（１９８８．２．４）

システム開発法ＪＳＤの定義付けの試み

野村研仁　井上克郎　鳥居宏次
大阪大学基礎工学部

本稿では，種々のソフトウェア開発手法で用いることができる一般的で簡明なプロセス及びプロダクトの概念を提案する．一つのプロダクトは，ある言語に属する“テキスト”（グラフ，表，文章，プログラム等）であり，その言語固有の定義に従ってその意味が与えられる．プロセスは，入力プロダクトを用いて出力プロダクトを生成する過程であり，前提条件，完了条件，及びサブプロセスの関係の三つの要素から構成される．前提条件は，プロセスに入力されるプロダクトが満たすべき条件を表わし，完了条件はプロセスが出力するプロダクトが満たすべき条件を表わす．また，サブプロセスの関係は，サブプロセス間の実行順序，及び，親プロセスとサブプロセス，並びに，サブプロセス間の入出力プロダクトの対応を表わす．我々は本稿で述べるプロセスやプロダクトを用いて，ソフトウェアシステム開発法ＪＳＤの定義を現在行っており，既に記述した例について本稿でふれる．

Notes on Models of Software Products and Processes in Software Development Methods through JSD

Kenji NOMURA, Katsuro INOUE, and Koji TORII

Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University
1-1 Machikaneyama, Toyonaka, Osaka 560, Japan

We discuss simple and generic models for processes and products appearing in various software development methods. A process is a text, which might be graphs, tables, sentences, programs, etc. The text is written in a language, whose meaning is given by a semantic definition of the language. A process takes some products as its inputs and produces some products as its outputs. The process is composed of preconditions, postconditions, and relations between subprocesses. We will also describe these models using a part of Jackson System Development (JSD) as an example.

1．はじめに

ソフトウェアの生産性や信頼性を向上させることを目的として種々のソフトウェア開発手法が提案されている[松本a87][松本b87]．これらの手法はソフトウェア開発プロセスの記述と開発作業中に利用又は生成されるプロダクトの記述の組によって表現することができる．

本稿では，ソフトウェアプロセス及びソフトウェアプロダクトのモデルを提案する．我々は，これらのモデルに基づいてソフトウェアシステム開発手法ＪＳＤの定義付けを実際に試みているので，この記述についても述べる．

ここでいうプロダクトとは一般に，ソフトウェア開発作業によって生成又は利用されるプログラム・仕様書・解説書（ドキュメント）などをさす．

プロセスとは，いくつかのプロダクト（入力プロダクトと呼ぶ）を利用していくつかのプロダクト（出力プロダクトと呼ぶ）を生成する作業過程のことをいう．

ソフトウェア開発過程に関する研究の代表的なものとして，プロセス・プログラミング[Oste87]，Kaiserらの研究[Kais87]，Williams らの研究[Will88]などがある．

プロセス・プログラミングは，ソフトウェア開発プロセスをプログラムのように記述し実行することを提案している．Kaiser らは，ツールを起動するための前提条件とツール起動方法及びツール起動後の完了条件をルール形式で記述することにより，プロセスを表現しようとしている．Williams らは，プロセスやプロダクトのモデルの他にツールに関するモデルを用意し，ソフトウェア開発支援ツールの統合などについて研究している．

これらの研究ではそれぞれ，プロセスやプロダクトのモデルを提案している．

プロセス・プログラミングはプロセスを手続き的なものとしてモデル化し記述しようとしている．一方 Williams らはプロセスを動作的にとらえてモデル化している．また，Williams ら及び Kaiser らは，プロセスの手続き的な側面をアクションという概念を設けて表わしている．

本稿ではこれらの研究で提案されているプロセスモデル及びプロダクトモデルよりも簡明なモデルを提案する．

プロセス PS は，三字組 <PRE,POST,REL> である．PRE は前提条件の集合，POST は完了条件の集合，REL はサブプロセスの関係の集合である．前提条件は，プロセスに入力されるプロダクトが満たすべき条件を表わす．完了条件は，プロセスが出力するプロダクトが満たすべき条件を表わす．入力されるプロダクトや出力するプロダクトが複数個存在する時は，それぞれのプロダクトに関する条件が必要である．一つのプロセスをいくつかのプロセス（サブプロセス）の集まりとして表わしたい時は，サブプロセスの関係をプロセスの定義中に記述する．

プロダクト PD は，一つの“テキスト”であり，ある言語 L_{PD} に属する．おのおのの言語には，言語自身の意味が存在する．ただし一般には，言語の意味が明示的に表わされていない事が多い．

種々のソフトウェア開発手法には，それぞれ独自の特徴がある．本稿で述べる方法によって開発手法を記述した場合，各開発法は，

(1) 一連の開発プロセスをどのように部分的なプロセスに分割し関係づけするか，

(2) どのような中間的なプロダクトを開発作業中に生成するか，

ということで違いが生じる．

また開発法が一連のソフトウェア開発作業のどの部分を対象としているかということは，外部から入力されるプロダクトの集合と生成を目的とするプロダクトの集合によって表わされる．

ＪＳＤ[Jack83][Came86]は，仕様要求分析・ソフトウエアの設計・プログラミングなどを含むソフトウェアシステム開発手法である．ＪＳＤの特徴として，仕様化と実現を明確に区別していること，ソフトウェアシステムが持つ機能の記述や仕様化を手掛ける前にシステムが対象とする現実世界についての記述とそのモデル化をすることなどが挙げられる．ＪＳＤは開発作業手順・各作業の完了基準・技法・記法といった，プロセスやプロダクトに関する記述をかなり明確に行っている．

ＪＳＤが開発手法として有効であること，開発プロセスやプロダクトがかなり明確であることから，我々はこの開発手法を記述の題材として選択した．

2．プロダクト及びプロセスのモデル

本章ではソフトウェアプロダクトとソフトウェアプロセスのモデルについて述べる．

2．1．プロダクトのモデル

プロダクトは，ある一つの言語に属するテキストである．例えばプロダクトがソースプログラムである場合，プロダクトが属する言語は，プログラムの記述に用いられている言語そのものである．プロダクトが図形で表現されているときは，プロダクトはその図形を定義するような言語に属する．

自然語などのように形式的に定義できない言語も多いが，ここではこのような言語も許す．言語にはそれぞれその意味が存在するが，意味の定義は明示的になされていないことが多い．

ソフトウェア開発作業中に利用又は生成されるプロダクトには，開発作業を開始する以前にすでに存在していることが仮定されており開発作業の際に外部から入力するプロダクト（外部入力プロダクト），開発作業によって最終的に得ることを目的とするプロダクト（目的プロダクト），開発作業の途中に生成されるプロダクト（中間プロダクト）の三種類がある．

2．2．プロセスの概念

プロセスは一つ以上のプロダクトを用いて一つ以上のプロダクトを生成する過程である．

プロセスは抽象度に応じて任意の大きさに分割することができるが，どのように分割した場合でも必ず入力プロダクトと出力プロダクトを持つ．

形式的にはプロセスを，前提条件，完了条件，サブプロセスの関係の組で表現する．

ここで前提条件とは，入力プロダクトが満たすべき条件に相当し，完了条件とは出力プロダクトが満たすべき条件に相当する．入出力プロダクトはそれぞれ複数個存在することがあるが，この場合それぞれのプロダクトごとに条件がある．

前提条件や完了条件は，それらに対応する入力プロダクトや出力プロダクトの記述言語を用いて表わす場合や，他の言語を用いて表わす場合，また，その両方を用いる場合もあろう．

一般に，一つのプロセス（親プロセス）はサブプロセスの集合であると考えることができる．サブプロセスはそれ自身プロセスとして別に定義される．サブプロセス間の関係を用いて，各サブプロセスの実行順序と入出力プロダクトを指定する．サブプロセス間の関係を定義する際の注意としては，

(1) あるサブプロセスを実行するときは，そのサブプロセスの前提条件を満たすプロダクトが存在していなければならない，
(2) サブプロセスに入力されるプロダクトは，その親プロセスの入力プロダクト，同じ親を持つ他のサブプロセスの出力プロダクト，外部入力プロダクトのいずれかでなければならない．
(3)ある親プロセスに対応するサブプロセスの実行がすべて終了した時，親プロセスの完了条件を満たすような出力プロダクトが生成されていなければならない．

一般に，最後に評価されたサブプロセスの完了条件と親プロセスの完了条件は一致しない事があるが，そのような場合でも，実行されたすべてのサブプロセスの完了条件を用いて親プロセスの完了条件が満たされるようになっていればよい．

プロダクトおよびプロセスは形式的に記述できない場合がある．形式的に記述できない場合でも，あるプロセスまたはプロダクトが必要な条件を満たしていない場合，プロセスを実行している途中で必要な作業を正しく実行することができなくなることを人間が発見できると期待される．

我々は，必ずしもすべてのプロセスが機械的に実行される必要はないと考えている．むしろ，ソフトウェア開発作業を行なう際には，かなり人手に頼らなければならないことが多いのが現状である．従ってプロダクトやプロセスを必ずしも形式的に記述する必要はなく，自然語で記述することなども許すものとする．

なお自然語のあいまいさの取り扱いに関して盛んに研究が行なわれているが，現在のところ多くの問題が残っている[辻井87]．

2．3．具体的なプロダクトとプロセスの例

ソフトウェア開発プロセスの例として，ＪＳＤの構造図から構造テキストを作成する過程について考える（図１）．

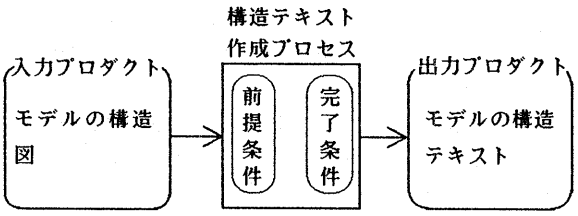

図１　構造テキスト作成プロセスと入出力プロダクト

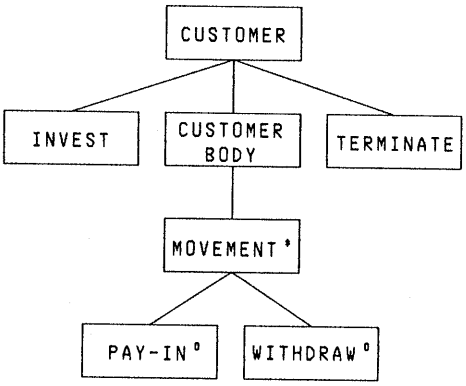

図２　顧客の構造図

```
CUSTOMER seq
  INVEST;
  CUSTOMER-BODY itr while cond-CB
    MOVEMENT sel cond-PAY
      PAY-IN;
    MOVEMENT alt cond-WD
      WITHDRAW;
    MOVEMENT end
  CUSTOMER-BODY end
  TERMINATE;
CUSTOMER end
```

図３　顧客の構造テキスト

ＪＳＤでは，作成しようとするソフトウェアシステムが対象としている世界に存在する“実体”のモデル化を行なう．実体の振舞いは，“事象”の列として表わされる．この事象の発生順序を“構造図”及び“構造テキスト”を用いて表わす．

構造図は，木構造になっており，頂点にあたる部分が箱型になっている（図２）．それぞれの箱には事象の名前が書かれている．構造図は，直感的には，箱に書かれている事象が左側から順に発生することを表わしている．事象の名前の右肩に＊印がついている時は，その事象が繰り返し発生することを表わし，０印がついている時は，いずれかの事象のうちの一つが発生すること（選択）を表わす．

従って，木構造になっており，それぞれの箱に事象の名前や＊印，０印を書くように定められた言語に構造図が属すると考えることができる．それゆえ，ＪＳＤで用いる構造図は，我々が定めたプロダクトの定義に当てはまる．

図２の構造図は，簡単な銀行システム[Jack83]の顧客（CUSTOMER）のモデルの振舞いを表わしている．この図は，顧客の振舞いを，まず最初に口座開設（INVEST）があり，それに続いて顧客本体（CUSTOMER BODY）が起こり，最後に口座閉鎖（TERMINATE）が起こる事であると表わしている．さらに顧客本体は行動（MOVEMENT）の０回以上の繰り返しであり，行動は支払い（PAY-IN）又は引き出し（WITHDRAW）のいずれかであることを表わしている．

この図を構造テキストを用いて書くと図３のようになる．図中，seq は事象が逐次的に発生することを表わす．また itr は繰り返しを表わし sel とalt の組み合わせで選択を表わす．この構造テキストは，ある文脈自由文法から生成される言語に属すると考えることができるので，本稿でいうプロダクトである．

次に構造テキスト作成プロセスについて考える．このプロセスの前提条件は，入力される構造図が構造図を記述するための言語に属することである．プロセスの完了条件は，(1) 出力される構造テキストが構造テキストを記述するための言語に属すること，(2) 構造図に出現する事象の名前が構造テキストにも出現すること，(3) 構造図に表現されている事象の発生順序が構造テキスト中の事象の順序と等しいこと，が考えられる．

完了条件のうち (2) と (3) の条件は，構造図から構造テキストを作成する手順を用いて表わすことができる．構造図を先順（preorder）で探索し，探索中に出現する順序にしたがって事象を並べ，必要に応じて，選択や繰り返しの条件を書いていくことにより構造テキストを作成することができる．この方法を用いて作成することを出力プロダクトの条件にすればよい．

また条件を表わす別の方法として，構造図中の各箱に書かれている事象に着目し，自分よりも左にある箱に書かれている事象は，構造テキスト中において自分よりも前に書かれていなければならない事などを利用することが考えられる．

後者の場合はプロダクトが持つ性質に着目して条件を表わすことになる．この場合，構造テキストを作成する手順を書く必要がなくなる．

3．ソフトウェア開発法の記述

ソフトウェア開発法は，プロダクトの記述と開発プロセスの記述の組によって表現することができる．

開発作業がいくつかの段階に分けられている場合は，それぞれの段階に対応するサブプロセスを定義する．

プロセスの完了条件として，出力プロダクトがどのような性質を満たすべきかということを記述する場合もあろう．その際，具体的にどのようにすればそのような性質を満たすことができるかということは指定しないかもしれない．

また逆に必要ならば，特定の手続きやアルゴリズムの適用，特定のツールの起動あるいは人間の特定の行動の起動などを，完了条件やサブプロセスの関係中に表現することができる．

ところで，プロセスを実行する際に人間がプロセスの記述で指定されている入力又は操作を行いその正しさも人間が保証しなければならないことがある．一般にこのような場合，入力あるいは操作が行われた時点では誤りがあってもそれが分からないことがある．誤りがある場合は後続するプロセスにおいて，人間がプロセス中のある作業を適切に実行することができなくなるので誤りがあったことを発見できる．

人間に種々の作業をさせる必要がある時には，その都度，プロセスから通信文を出力する．その答えとなる人間からの通信入力はプロセスへの入力プロダクトとして取り扱う．

4．システム開発法ＪＳＤの記述方針

4．1．システム開発手法ＪＳＤ

ＪＳＤの開発プロセスは，エンティティ・アクションステップ，エンティティ構造ステップ，初期モデルステップ，機能ステップ，実現化ステップの５つのプロセスに分割される．

エンティティ・アクションステップ，エンティティ構造ステップ，初期モデルステップの３つのプロセスで現実世界の記述とそのモデル化を行う．直感的にはこれらのプロセスで現実世界のシミュレーションモデルを作成する．

機能ステップでは，必要な出力を得るための“逐次プロセス”をシミュレーションモデルに付加する．

実現化ステップでは，前ステップまでに作成したモデルが目的とする計算機環境上で作動するようにモデルを変換する．

各ステップでは，構造テキスト・構造図・システム仕様図などＪＳＤ特有のプロダクトを生成する．

4．2．ＪＳＤの記述方針

本節では，ＪＳＤの定義付け作業の方針について述べる．

ＪＳＤは，作成したいソフトウェアシステムに対する要求を用いてソフトウェアシステムの実現図，及び，実現図内の個々のプロセスの動作を記述したものなどを作成する．従ってこれらがＪＳＤの入力プロダクト及び出力プロダクトになる．

前節で述べたようにＪＳＤの一連の作業は大きく５つのステップに分割される．これらのステップをＪＳＤのサブプロセスであると考える．ＪＳＤでは，基本的には５つのステップを逐次的に行なうので，サブプロセス間の実行順序もそのように指定する．また各ステップは，利用するプロダクトと生成するプロダクトがあらかじめ決められているので，各サブプロセス間の入出力関係も指定できる．これらの記述によって最上位のプロセスであるＪＳＤプロセスが定義される．

ＪＳＤでは，構造図・構造テキスト・システム仕様図・システム実現図など，開発作業中に生成するプロダクトの記法や意味も定めている．従って定義記述では，これらをプロダクトを記述するための言語として指定する．

なお，ＪＳＤの種々の特長を表現するためには，より詳細なレベルでプロセスをとらえ定義する必要がある．従って各ステップをさらに細かいプロセスに分割し定義しなければならない．

5．ＪＳＤの記述例

本章ではＪＳＤの開発プロセスとプロダクトの実際の記述について述べる．

5．1．ＪＳＤのプロダクトの記述例

プロダクトの記述言語の例として，ＪＳＤのプロダクト“システム仕様図”が属する言語の記述を図４に示す．

“システム仕様図”は，ＣＳＰ[Hoar78]をモデルにしている．システム仕様図の方形はＣＳＰの逐次プロセスに対応し，矢印・円・菱形を用いて表わされる方形間の関係は，ＣＳＰにおけるプロセス間通信に対応する．

図５に“システム仕様図”のプロダクト関数の定義を示す．ここでは代数的な記述を採用している．この

1．システム仕様図 <V,E> は，頂点の集合 V と有向辺の集合 E から構成されるグラフである．

2．頂点には４種類ある．

PROCESS: 

PROCESS に隣接する頂点は複数個存在してもよい．

DATA_STREAM: 

DATA_STREAM に入る辺と出る辺はそれぞれ１つのみである．なお，どちらか一方のみでもよい．

STATE_VECTOR: 

STATE_VECTOR には，入る辺と出る辺が必ず一つづつ存在する．

SYNC_PROCESS: 

SYNC_PROCESS は，DATA_STREAM から入る辺を１つだけ持ち，DATA_STREAM へ出る辺を２つ以上持つ．

3．有向辺には３種類ある．

 ：接点のうち一方が PROCESS 又は SYNC_PROCESS であり，他方が DATA_STREAM 又は STATE_VECTOR でなければならない．

 ：接点に関する規則は上記の辺と同様．

 ：この辺は DATA_STREAM から出て PROCESS に入っていなければならない．

＜注意＞

・辺 e = (u,v) (u,v は頂点) が存在するとき，辺 e は 頂点 u から出るといい，かつ，頂点 v に入るという．また，u,v を辺 e の接点という．

図４　システム仕様図記述言語

```
PRODUCT システム仕様図

FUNCTION_DECLARATION
  putprclabel:PROCESS,LABEL->PROCESS;
  getprclabel:PROCESS->LABEL;
  putdslabel:DATA_STREAM,LABEL->DATA_STREAM;
  getdslabel:DATA_STREAM->LABEL;
  putsvlabel:STATE_VECTOR,LABEL->STATE_VECTOR;
  getsvlabel:STATE_VECTOR->LABEL;
   ...
END

FUNCTION_DEFINITION
  getprclabel(putprclabel(Prc,Lbl)) == Lbl;･･･①
  getdslabel(putdslabel(Prc,Lbl)) == Lbl;  ･･･②
  getsvlabel(putsvlabel(Prc,Lbl)) == Lbl;  ･･･③
   ...
END
```

図５　システム仕様図のプロダクト関数の定義記述例

```
PROCESS JSD

INPUT_PRODUCT_LANGUAGE
  Natulal Language;

OUTPUT_PRODUCT_LANGUAGE
  Language of System Implementation Diagram;

CONDITION_OF_INPUT_PRODUCT
  contain_valid_req_spec(Req_Spec) == TRUE;･･･①

CONDITION_OF_OUTPUT_PRODUCT
  equiv(Req_Spec,JSD(Req_Spec)) == TRUE;   ･･･②

RELATION_OF_SUBPROCESS
  JSD(Req_Spec) ==
    ImplementationStep(
      FunctionStep(
        InitialModelStep(
          EntityStructureStep(
            EntityActionStep(Req_Spec)))));
END
```

図６　プロセス JSD の記述例

例では，FUNCTION_DECLARATION 部で関数の宣言を行ない，FUNCTION_DEFINITION 部で関数本体を定義している．①，②及び③は方形，円，菱形それぞれのラベルの値の代入及び参照関数の定義である．

5．2．ＪＳＤのプロセスの記述例

プロセスの記述の例として，プロセス“ＪＳＤ”の記述の例を図6に示す．図中，INPUT_PRODUCT_LANGUAGE 部で入力プロダクトの記述言語を宣言し，OUTPUT_PRODUCT_LANGUAGE 部で出力プロダクトの記述言語を宣言する．ここでは，入力されるプロダクトが自然語に属し，出力されるプロダクトは，“システム実現図”を記述する言語に属することを表わしている．

CONDITION_OF_INPUT_PRODUCT 部で，入力されるプロダクト Req_Spec に必要な仕様要求が含まれていること①を表わし，CONDITION_OF_OUTPUT_PRODUCT 部で，出力されるプロダクトが Req_Spec の要求を満たすべきこと②を表わす．

RELATION_OF_SUBPROCESS 部ではサブプロセスの関係を定義している．ここではＪＳＤの５つのステップに対応する５つのサブプロセスによってＪＳＤが分割されること，及びそれらの実行順序と入出力関係を表わしている．なおこの例では，各サブプロセスはすべて一入力一出力として表現している．本来，ＪＳＤの多くのステップは複数の入出力を持つが，ここでは簡単にするため，複数のプロダクトを一つの組にしたものを入出力とするような記述方法を採用している．

また，プロダクトが満たすべき条件の記述の更に詳細な例として，ＪＳＤの初期モデルステップによって生成される出力プロダクト“システム仕様図”，“モデルプロセス構造図”，“モデルプロセス構造テキスト”が満たすべき条件の記述を表１に示す．

6．おわりに

種々のソフトウェア開発手法に従って作成されるプロダクトの多くは，既成の汎用的な“モデル”を参照している．ここで“モデル”とは，ある目的を持った論理表現を解釈し，その真偽を論ずるために設定する概念の記述や図式のことである[松本b87][長尾83]．

既成の汎用的なモデルとしては，データフローモデル，コントロールフローモデル，有限状態機械モデル，ＣＳＰ，オブジェクトモデルなどがある．

上述したようなモデルに基づくプロダクトやこれらのプロダクトを生成するソフトウェア開発プロセスに関する概念は，いずれも本稿で述べた定義に従ったプロダクトやプロセスを用いて説明することが可能であると考えられる．

今後，我々は，ＪＳＤを含む種々のソフトウェア開発手法について，本稿で述べた方法にしたがって定義付け及び記述を試み，本手法の有効性の実証と改良点の検討を行っていく予定である．さらにこの作業を通して，ソフトウェア開発手法を記述するために用いることができる枠組についても検討していきたい．

謝辞　ＪＳＤの解釈に関し，多数の有益な御助言御協力をいただきました日本ユニバック㈱の加藤潤三氏に感謝致します．

表１　初期モデルステップの出力プロダクトが満たすべき条件

(1) システム仕様図は正当でかつ以下の各項目を満たす．
(1.1) すべてのエンティティについて対応するモデルプロセスが存在する．
(1.2) 有袋エンティティでないエンティティに対応するモデルプロセスは実世界の対応するプロセスと結合されている．
(1.3) 有袋エンティティに対応するモデルプロセスは，モデルプロセスのうちの一つ（これを親プロセスと呼ぶ）が実世界の対応するプロセスと結合されており，他のプロセスは親プロセスとデータストリーム結合されている．
(1.4) 結合はデータストリーム，ステートベクトルのいずれであるかが明示され，さらに一対一，多対一，多対多のいずれであるかも明示されている．
(2) モデルプロセス構造図とモデルプロセス構造テキストはそれぞれ正当でかつシステム仕様図に含まれているすべてのモデルプロセスについて存在する．
(3) モデルプロセスの結合とモデルプロセス内の入出力操作が正しいことをシステム作成依頼者が確認した
(注：ここで正当であるとは，形式的な定義を満足していることをいう．また正しいとは，人間が定義を満たしていることを確認しなければならないような事項を満足していることをいう)

参考文献

[松本a87] 松本正雄，”設計とプログラミングの自動化”，情報処理学会誌，Vol. 28, No. 7, pp. 862-872 (1987).
[松本b87] 松本吉弘，”ソフトウェアに対する要求の形成”，情報処理学会誌，Vol. 28, No. 7, pp. 853-861 (1987).
[長尾83] 長尾，淵，”論理と意味”，岩波講座情報科学７，岩波書店 (1983).
[大野87] 大野豊，”ソフトウェア工学の背景と展望”，情報処理学会誌，Vol. 28, No. 7, pp. 845-852 (1987).
[辻井87] 辻井，上原，”ソフトウェア工学と自然言語処理”，情報処理学会誌，Vol. 28, No. 7, pp. 913-921 (1987).
[Booc86] G.Booch, "Object-Oriented Development," IEEE Trans. Software Eng., Vol. SE-12, No. 2, pp. 211-221 (1986).
[Came86] J.R.Cameron, "An Overview of JSD," IEEE Trans. Software Eng., Vol. SE-12, No. 2, pp. 222-240 (1986).
[Hoar78] C.A.R.Hoare, "Communicating Sequential Processes," Comm. ACM, Vol. 21, No.8, pp. 666-677 (1978).
[Jack83] M.A.Jackson, "System Development," Prentice-Hall (1983).
[Kais87] G.E.Kaiser and P.H.Feiler, "An Architecture for Intelligent Assistance in Software Development," Proc. of 9th ICSE, pp.180-188 (1987).
[Myer75] G.J.Myers, "Reliable Software through Composite Design," Petrocelli/Charter (1975). 邦訳（久保，国友），”高信頼性ソフトウェア－複合設計”，近代科学社 (1976).
[Oste87] L.Osterweil, "Software Processes are Software Too," Proc. of 9th ICSE, pp.2-13 (1987).
[Tori84] K.Torii et al., "Functional Programming and Logical Programming for The Telegram Analysis Problem," Proc. of 7th ICSE, pp.463-472 (1984).
[Will88] L.G.Williams, "Software Process Modeling: A Behavioral Approach," submitted to 10th ICSE (1988).