ソフトウェア工学 73－8
（１９９０．７．１９）

交換ソフトウェアプロセスモデリングに関する一考察

白石 智

ＮＴＴ交換システム研究所

ソフトウェアプロセスを計算機で支援することを目的に議論を行っている。まず、ソフトウェアプロセスに着目する背景、問題点を述べ、要求条件を明確にしている。そして、ウォーターフォールモデルで示されるライフサイクルモデルを保守について拡張し、さらにそれの持つ外包的、内包的性質を規定したソフトウェアプロセスモデル、『３－stageプロセスモデル』を提案する。また、ソフトウェアプロセスを記述する手法としてプロセスネットを提案し、ソフトウェアプロセスが明示できることを示す。最後にこれらを適用した具体的な開発支援システムの構成例を述べ、開発支援への応用について考察している。

A Study on Switching System Software Process Modeling

Satoshi Shiraishi

NTT Communication Switching Laboratories

9-11, Midori-cho 3-Chome Musashino-Shi, Tokyo, 180, Japan

This paper discusses the software development processes for switching systems. The background for taking attention to it is described and requirements are clarified. It proposes 3-stage process model, which is the software process to expand the maintenance phase of Water-Fall life cycle model and to define its outward and internal nature. And it proposes the method to describe the software process model, which is called Process-net. It is possible to declare the software processes by this method. Furthermore it describes the example of a development supporting system and considers applications for development supporting.

1. はじめに

近年、通信サービスの多様化、タイムリーなサービスの提供が要求され、交換ソフトウェアの生産性向上に対する期待が高まっている。そして、交換プログラムへのオブジェクト指向概念、ＣＴＲＯＮインタフェースの適用等交換プログラム構造の改良[1][2]、すなわちプロダクト（生産物）そのものを良構造化しようという研究、自動プログラミング[3]等各種ツールによる開発支援、すなわちソフトウェアプロセス（開発過程）を効率化しようという研究等、両面からの研究がさかんに行われているが、多くの課題が残っているのが現状である。

最近、ソフトウェア工学の分野ではソフトウェアプロセスに関する検討が盛んに行われており、ソフトウェアライフサイクルモデルに関する提案[4]、ソフトウェア開発過程の形式的記述等[5][6][7][8]による各種開発支援への適用が試みられている。

筆者も、交換プログラムの生産性向上をねらって、特に、実際の大規模システムに適用するという観点から、交換ソフトウェアの開発プロセスに着目した開発支援手法の検討を行っている。そして、実際に運用されている交換機のソフトウェアに関して、初期開発から維持管理も含めたプロセスの記述を試み、プロダクトの制約によるプロセスのユニファイが、プロセスの明示（プロセスの定義、形式的記述等）に有効なこと、そして、プロセスに着目した開発支援手法と適用範囲等について提案を行ってきた。[9]

本稿では、２章において、ソフトウェアプロセスに着目する背景、問題点を述べ、これらに対する要求条件を明確にする。３章では、ソフトウェアプロセスの議論を、大規模な実システムの開発支援に適用するための基盤へと発展させる。ライフサイクルを中心に考察し、それの外包的、内包的性質を考慮したソフトウェアプロセスモデルとして３－STAGE プロセスモデルを、及びソフトウェアプロセスの階層記述法としてプロセスネットを提案する。そして、４章において、これらを適用した具体的な開発支援システムの構成例を述べ、開発支援への応用について考察している。

2. 背景

交換ソフトウェアが抱えている問題の１つとして、それが非常に大規模システムであるという点がある。具体的には、交換システムは一度導入されてから更改までの時間は最低１０年以上と非常に長いこと、初期開発から維持管理まで含めると、述べ数千人以上の人間が何らかの形でそのソフトウェアに携わること、単純に規模だけでみても、数１００Ｋラインから数Ｍラインに及ぶこと等が大規模であることの例としてあげられる。そして、開発の指針となる作業標準等は用意されていても、これまでは生産物の規定に関する記述が中心であった。開発プロセスについては大規模であるがゆえに記述量が膨大であり、システム全体を体系的に記述することが困難であった。そのため、開発過程を詳細に規定することは少なかった。　しかし、その結果として、ソフトウェアの初期開発時とシステムの運用中に実施される機能追加（半年に１度くらいある。）で担当者が異ならざるをえないという大規模システムの宿命による問題が顕著になってきた。たとえば、開発ノウハウが引き継がれない、属人的開発手法になりやすい等の問題の発生である。

このような問題の対策として、開発プロセスの明示とそれによる開発支援が有効と考えるのは当然の帰結とも言える。

すなわち、ソフトウェアプロセスに着目する理由は、人間に依存していた開発作業を計算機により支援し、効率化をはかろうということに集約できる。そして、このような支援の実現のためには、ソフトウェアプロセスモデルの確立、プロセスモデルを開発支援システムへ応用することが要求されてくる。

3. ソフトウェアプロセスの方法論

3.1 概要

ソフトウェアプロセスの方法論として議論する範囲は広く、また、方法論という言葉の定義もあいまいである。ここでは、『ソフトウェアプロセスの方法論とは、ソフトウェアプロセスを計算機で支援することを達成するため、それに要求される、あるいは前提となる考え方や技術』と考えることとする。

本章においてはこれに基づき、特にソフトウェアプロセスを計算機上で明示することをねらい、そのアプローチとして最も重要と考えられる下記の2点について議論する。

①システムの開発から保守までのライフサイクルを一環して捉えるソフトウェアプロセスモデル
②ソフトウェアプロセスの表現方法

3.2 ソフトウェアプロセスモデル

3.2.1 ソフトウェアプロセスモデルの分類

ソフトウェアプロセスモデルは、ソフトウェアプロセスを定義し、支援するために要求される。そして、一般にウォーターフォールモデル[10]に代表されるライフサイクルモデルを指す場合が多い。本稿においても、ソフトウェアプロセスとして、このようなライフサイクルモデルを中心に捉えるが、2章で述べた問題を解決するために保守工程をより考慮したモデルに拡張する。

さらに、ライフサイクルモデルはソフトウェア開発作業の工程の区切り、作業の種類をスタティックに表すだけで、それ自身の成長、内部状態、作業状態等ダイナミックな性質を明確にしていない。つまり、ソフトウェアプロセスの断片的性質しか規定していない。しかし、ソフトウェアプロセスの全体像を表すには、ダイナミックな性質も含めたモデル化が重要と考えられる。そこで、スタティックなライフサイクルモデルを、ミクロな視点とマクロな視点から見ることで、ダイナミックな性質を抽出し、モデル化する。そして、これらのモデルを組み合わせることで、トータルなソフトウェアプロセスモデルを構築する。

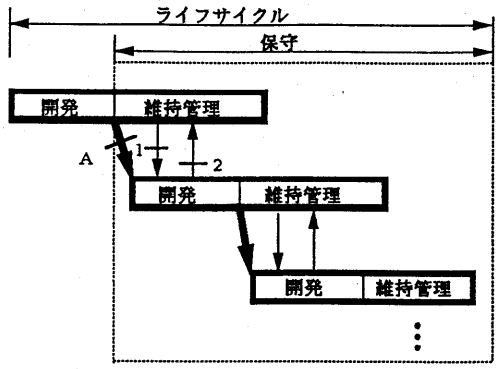
3.2.2 ライフサイクルモデル

ウォーターフォールモデルに代表されるライフサイクルモデルが有名であるが、近年各種モデルが提案されている。[11] 各モデルには、種々の改良点（工程の区切り方、仕様変更への対応の仕方・・・）、考え方等の厳密な差は多々あり、それらの有効なものは積極的に適用すべきである。しかし、ソフトウェアの生産は基本的に開発（設計、製造、テスト）、保守（維持管理、機能追加）という作業が必要である。そして、ライフサイクルモデルはこれらの作業の種類、区切り等を示す指針であると考えれば、ほとんどのモデルは、改ウォーターフォールの範疇に含まれる。

本節では、このようなライフサイクルモデルの保守に関する部分について考察する。一般に保守というのは、ソフトウェアを運用できる状態に維持管理する作業のことをいう。交換システムの場合、具体的な作業としては、運用作業、バグの修正（事後保全）、ユーザの要求に対するソフトウェアの改版（機能追加、変更）等があり、特に、事後保全や機能追加、変更の迅速性の要求はきびしい。そして、これらの作業は、ほとんどの場合開発者とは別のグループ（人）が実施している。しかし、ライフサイクルにおける保守の位置付けは、その最終段階に置かれているだけで、上記のような作業の相違、特性を考慮していない。たとえば、ソフトウェアの機能追加作業は、開発と同様にウォーターフォールモデル等に従って作業を行う必要がある。これは、言い換えればソフトウェアの寿命の範囲内で開発作業が続くということだが、それは陽になっていない。そして、これが開発ノウハウが引き継がれない、属人的開発手法が生じる一因にもなっており、この保守段階の作業の体系化がソフトウェアプロセス支援を実現するための一課題となる。

そこで、図3.1に示すような、連続ライフサイクルモデルを提案する。これは、保守工程におけ

る運用作業や事後保全等日々の維持管理作業と、機能追加、変更等の開発作業を明確に区別している。そして、それらの間の関係を規定すれば、保守段階に入っても開発が続くというソフトウェアのライフサイクルを自然な形で明示することができる。



1:維持管理から開発への規定点：：事後保全結果等（バグフィードバック等）

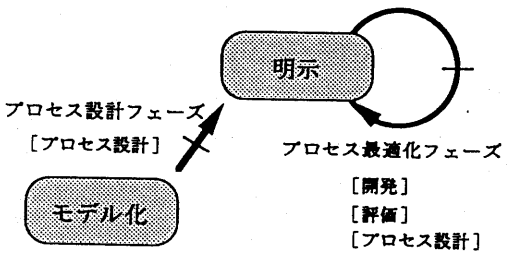
2:開発から維持管理への規定点：：予防保全施策等（教育等）

A:機能追加フェーズへの移行：：機能追加、変更

図3.1　連続ライフサイクルモデル

3.2.2　マクロなソフトウェアプロセスモデル

マクロなソフトウェアプロセスモデルとは、ライフサイクル自身が外界（ソフトウェア開発者、開発環境等）に対して見せる外包的性質を規定するためのモデルのことである。マクロなモデルとして、図3.2に示すようなソフトウェアプロセス進化モデルを提案する。



*明示：ソフトウェアプロセスが客観的に既知の状態にある。

図3.2　ソフトウェアプロセス進化モデル

ソフトウェアプロセス、特にライフサイクルモデルの具体的な内容は、本来一義的に規定されるものでなく、その時点における技術の到達度、人間のレベル、投入可能な工数、予算、方法論等さまざまな要因を背景に、各種ツールセット、作業を組み合わせることでモデル化、規定してゆくべきものである。すなわち、ソフトウェアライフサイクルはスタティックなものでなく、適宜評価を行い、常に最適なプロセスを指向して、進化を続けてゆくものと考えることが適切である。

また、ソフトウェアプロダクトは、その寿命の範囲内で機能追加により発展してゆくことは前節で述べた。そして、これは進化したソフトウェアプロセスにより出力されるものであり、プロセスとプロダクトは一体となって発展してゆくということが言える。そして、連続ライフサイクルモデルでは、このような機能追加を明確に位置付けプロダクトの発展を陽にしたが、これに対応するソフトウェアプロセスもモデルとして陽にする必要があるわけである。

このようなライフサイクルのダイナミックな性質を表すため、ソフトウェアプロセス進化モデルを明確にした。そして、このような最適なソフトウェアプロセスへ発展してゆくメカニズムがより重要になってくる。

3.2.3　ミクロなソフトウェアプロセスモデル

ミクロなソフトウェアプロセスモデルとは、ライフサイクルの内部状態、作業状態等内包的性質を規定するためのモデルのことである。ミクロなモデルとして図3.3のような開発プロセスモデルをすでに文献[9]において提案しているが、このモデルの背景と考え方を示す。

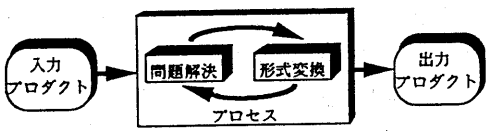


図3.3　開発プロセスのモデル化

この開発プロセスモデルでは、ソフトウェア開発作業は形式変換と問題解決の相互実行により実現されるものとしている。形式変換とは、定式的

に記述できるもの、つまり、ツール等により生産物が自動的に得られるものに、問題解決は人間が考える必要があるものに対応している。

すなわち、工程内で問題解決部分の割合が多いほど難易度が高く、少ないほど容易な工程と言える。従って、上流工程ほど問題解決部分が多くなる。また、このような問題解決部分は、定式化が可能になった時点で形式変換に置き換えることが可能である。これは、システム開発が1ラウンド終了した時点で、開発プロセスに関する技術的ブレイクスルーがあったり、ＱＣ活動等により各種作業をツール化すること等を意味する。これにより、プロセスの発展的展開が可能となる。このように、開発プロセスモデルは、ソフトウェアプロセスの最適化を指向するソフトウェアプロセス進化モデルの考え方とも合致している。

3.2.4　トータルソフトウェアプロセスの構築

ソフトウェアプロセスのスタティックなモデルとして連続ライフサイクルモデルを、ダイナミックなモデルとして外包的性質を規定するソフトウェアプロセス進化モデル、及び内包的性質を規定する開発プロセスモデルを位置付けた。そして、これらを組み合わせることで、これまでは断片的な性質のみしか規定できなかったソフトウェアプロセスをトータルなモデルとして表すことが出来る。これを3－STAGE プロセスモデルといい、そのイメージを図3.4に示す。

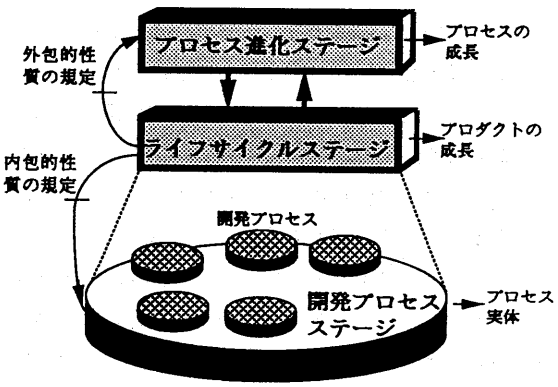


図3.4　３－STAGEプロセスモデル

この３－STAGE プロセスモデルは、開発から保守までのソフトウェアプロセスを明示するためのベースとして、開発支援システムへフィードバックが可能である。特徴として、以下のことがあげられる。

①ライフサイクルの考察では、対象プロダクトとして交換システムを意識し、保守工程を体系化した。しかし、他のシステムにも同様な問題は考えられ、本モデルの適用は可能である。

②3つのモデルを組み合わせることで、実際の作業ベースのことから、ソフトウェアプロセスの最適化の過程まで、ソフトウェア開発の全体像を表現している。

③ソフトウェアプロセスの最適化とソフトウェアプロダクトの機能追加を、ソフトウェア開発の表裏と捉え、プロセスとプロダクトの一体性を指向している。

3.3　プロセスモデルの表現法

3.3.1　プロセスモデルの表現手法

3.2章で議論したソフトウェアプロセスモデルをベースに具体的にそれを記述する手法について述べる。ソフトウェアプロセスの記述法として、以下の3案が考えられる。

案1：プロセス記述言語

案2：図形表現

案3：自然言語

計算機上でのプロセス支援を考慮すると、案1あるいは案2が望ましいが、上流工程は問題解決部分が多く形式的記述は困難である、さらに、大規模システムでは記述範囲が広く、すべてのプロセスを記述する言語、図形表現の定義は困難である。また、案3ではすべてのプロセスの記述は可能だが、計算機での支援は難しい。そこで、作業のマクロ的な流れの記述に図形表現を適用し、各作業の詳細な内容を自然言語で記述する階層記述を採用することにした。これにより、ソフトウェアプロセスを視覚で容易に捉えられ、しかも、全工程の作業内容をもれなく、無理なく記述するこ

とが可能になる。

図形表現の枠組みだが、ペトリネットを応用した記述を採用する。この理由としては、ソフトウェアの開発は、分散環境における多数の人間の作業の集合体であり、個々の作業はこの中での非同期事象ととらえられる。そして、このような非同期事象における情報の流れを記述するのにペトリネットは適しており、ソフトウェアプロセスの記述に応用出来る。たとえば、ペトリネットのプレースをプロダクトに、トランジションをプロセスに対応させることで、基本的な作業の流れを表現できるようになる。

自然言語による作業内容の詳細な記述では、当面『何を』『どのように』ということを明確にすることを原則に記述する。将来的には、制限された自然言語、プロセス記述言語による記述が課題である。

3.3.2 プロセスネットの概要

3－STAGE プロセスモデルを前節の表現手法をベースに、ライフサイクルが複数の開発プロセスの集合体によって構成されていることに着目して記述する例を図3.5に示す。これをプロセスネットと呼ぶ。

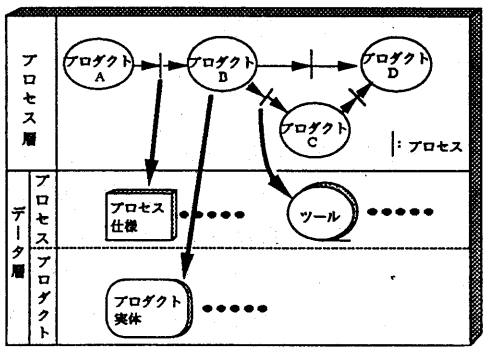


図3.5　プロセスネット概要

プロセスネットは、プロセス層において図形表現による開発作業の流れを、データ層においてプロセスとプロダクトの実体を表している。また、プロセス層とデータ層の間でリンクをとり、プロセス層の作業の流れと、そこで適用されるプロセスやプロダクトの実体の関係を示している。この時、プロセスに関しては、開発プロセスモデルにおける問題解決部分が自然言語で記述されたプロセス仕様に、形式変換部分がツールに対応している。また、プロセスの進化については、プロセスネットの上位層にその機構を配置することし、プロセスネット上には表現しない。

プロセスネットでは、基本記法に加えソフトウェアプロセスの記述に必須の各種事項を定めている。その作業、条件の記述例を表3.1に示す。

表3.1　作業、条件の記法例

項　目		記　法
プロセス実行	プロダクト	○
	プロセス	|
	実行箇所	▪
プロセス評価	レビュー	‖ (レビュー要プロセス)
	工程管理	⊖→ (データ 抽出)
	品質管理	●→ (データ 抽出)
	バグフィードバック	プロダクト を基準に戻りプロダクトへ矢印を記述
プロセス進展	判断	☆
	要求仕様変更	追加、変更項目単位に新トークンの追加

4. ソフトウェアプロセスの応用

4.1 概要

交換ソフトウェアに対する要求条件として、高信頼性、生産性向上があげられる。また、これを達成させる立場にある開発／保守サイドから見た時、維持管理期間が長い、頻繁な機能追加、多人数による開発／保守が要、という特徴があげられる。そして、これらは、2章で述べた現象面から捉えた問題点の『開発ノウハウが引き継がれない』『属人的開発手法になりやすい』ということの本質につながるものと考えられる。

従って、3章で述べたモデルや表現法によるソフトウェアプロセスの明示をベースとした開発支援システムを構築することで、交換ソフトウェアの生産性向上に貢献できる。本章では、このような開発支援システムの構成例及び具体的な開発支

援への適用についてのべる。

4.2 開発支援システム構成例

3－STAGE プロセスモデルとプロセスネットによりソフトウェアプロセスを明示し、それを開発支援環境のプラットホームとして組み込んだ開発支援システム構成例を図4.1に示す。

本システムの構築にあたり、以下に示す考え方を適用、明確化し、統一思想のもとでの発展をはかる。

①プロセスネットのイメージをそのままツールの全体構成に反映させる。（プロセスネットのプロセス層をプラットホーム、データ層をデータベースに対応させる。）

②ツールの統合化を指向し、ツールを組み込むための標準インタフェース（ＯＳ種別、ウィンドウ種別、ユーザーインタフェース、ツール間インタフェース条件等）を規定する。

③複数ＷＳをＬＡＮ等で結んだ分散環境下での適用を考慮し、プロセスの分割、統合手法を規定する。

4.3 ソフトウェアプロセスによる開発支援

ソフトウェアプロセスを明示するという枠組を明確に組み入れることで、基本的なソフトウェア開発をトータルに支援できるとともに、より高度な支援のサポートが可能になる。その例を示す。

◆開発プロセス自動実行制御

プロセスネットに沿って自動的に作業を進めるためのツール。プロセスネットを解釈し、各種ツールの自動起動、人間への作業の提示等を行う。さらに、従来は人間の経験によっていた部分、たとえば、プロセスの進展状況による次作業の決定等の判断もメタプロセス制御機構を設け自動化をはかる。このツール（プロセスエンジン）の構成を図4.2に示す。

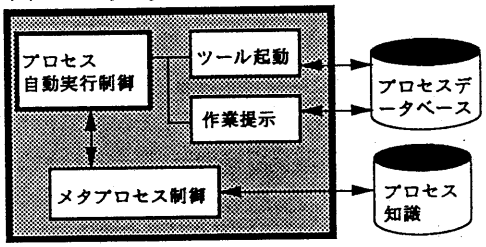


図4.2 プロセスエンジン概要

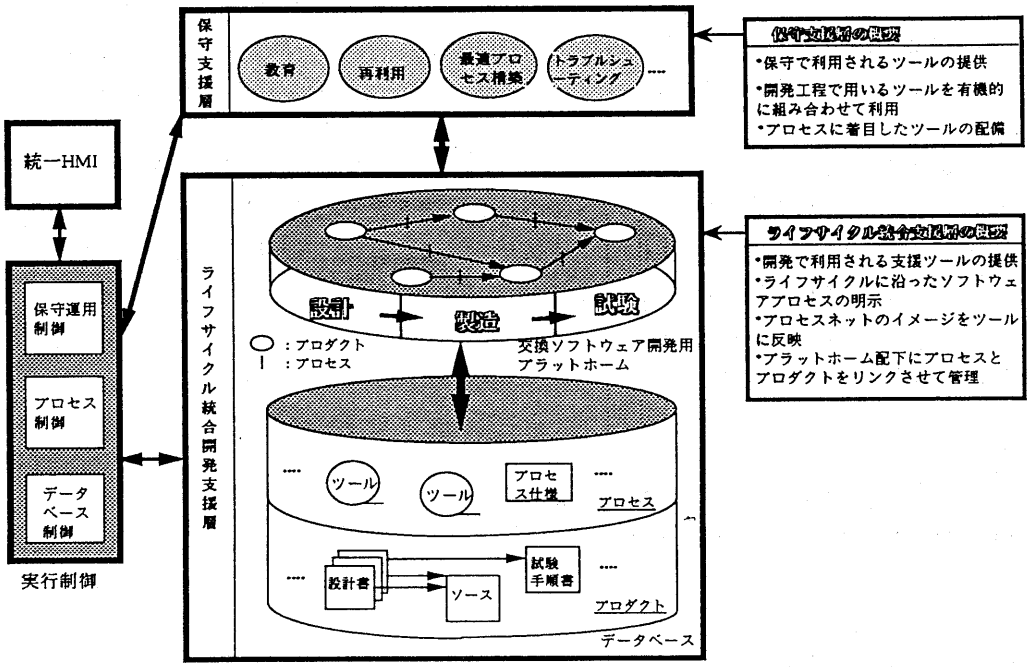


図4.1 開発支援システム構成例

このようにソフトウェアプロセスを支援することで可能になるツールの具体例を表4.1に示す。

表4.1　ｿﾌﾄｳｪｱﾌﾟﾛｾｽによる 高度開発支援例

項　目	支　援　内　容	
ﾌﾟﾛｾｽ明示	◇明示プロセスに沿った作業 ◇作業種別、ﾂｰﾙ の提示 ◇ﾒﾀﾌﾟﾛｾｽ制御による自動化	開発
ﾌﾟﾛｾｽ設計	◇ﾌﾟﾛｾｽの評価と問題点ﾌｨｰﾄﾞﾊﾞｯｸ ◇プロセスの自動学習 ◇最適プロセスの構築	保守
ﾌﾟﾛｾｽ再利用	◇ﾌﾟﾛﾀﾞｸﾄ のﾊﾟﾀｰﾝ化による ﾌﾟﾛｾｽ限定 ◇ﾌﾟﾛﾀﾞｸﾄ の類似による ﾌﾟﾛｾｽ再利用	
ﾏﾈｼﾞﾒﾝﾄ	◇ﾌﾟﾛｾｽによる 工程管理、品質管理	
教育	◇ﾌﾟﾛｾｽCAI による標準的教育	

5. まとめ

本稿においては、最初に、ソフトウェア開発における問題点、ソフトウェアプロセスに着目する背景を述べた。次に、ソフトウェアプロセスを計算機で支援することを目的に、開発から保守までのライフサイクル、それの持つ外包的、内包的な性質を考慮した3－STAGE プロセスモデル、及び図形表現と自然言語による階層的ソフトウェアプロセスの記述法プロセスネットを提案した。そして、このような考え方を実際の開発支援システムに適用した構成例、及び具体的な開発支援例を示した。

今後の課題として、3－STAGE プロセスモデル及びプロセスネットの評価を開発支援システムの試作を通して行う。また、ソフトウェアプロセスの自動学習について検討を行う。さらに、ソフトウェアプロセスとソフトウェアマネジメントとの融合が課題としてあげられる。

謝辞

日頃熱心に御討論頂く、ＮＴＴ交換システム研究所黒田グループリーダに感謝致します。

参考文献

(1)丸山、渡部：“既存並列処理言語による実時間オブジェクト指向プログラミング” 情処学会論文誌 Vol.31 No.1 pp88-97(1990-1)

(2)本田、浅村他：“高速パケット処理ソフトウェア構成法” 研実報告 38.3 pp223-230(1989)

(3)伊藤、市川：“通信分野における自動プログラミング” 情報処理 Vol.28 No.10 pp1405-1411 (1987)

(4)Boehm,B.W.：“A Spiral Model of Software Development and Enhancements " Proc. of International workshop on the Software Process and Software Environments, ACM SIGSOFT Software Engineering Notes, Vol.11 No.4 pp4-5(1986)

(5)Osterweil,L.：“Software processes are software too",Proc. of 9th ICSE,pp2-13 (1987)

(6)田村、中島他：“設計プロセスに対する支援機能についての一考察” 情処全大第39回5S-10(1989)

(7)稲田、荻原他：“ソフトウェア開発過程の形式化とその詳細化による支援システムの作成” 信学論 Vol.J72-D-1 No.12 pp876-881(1989)

(8)太田、落水：“プロセス記述言語としてのPrologの評価” 情処全大第40回 1S-2 (1990)

(9)白石：“開発プロセスに着目した交換プログラム開発支援手法に関する一考察” 信学全大春季 B-515 (1990)

(10)Royce,W.W.：“MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS” ,Concepts and Tequniques,IEEE WESCON,pp1-9 (1970)