ソフトウェア工学 71−18
（1990. 2. 9）

ソフトウェア・プロセス――
実時間処理システムにおけるケース・スタディ

望月純夫　山内顕　片山卓也　鈴木正人
三菱スペース・ソフトウェア　東京工業大学

ソフトウェア・プロセスは、設計現場における技術者の動き、その間に流れる情報を重要視し、その観察、分析、記述を基礎にして設計過程を究明しようとするものである。我々は、現在実用化されている実時間処理システムの一つを取りあげ、実際の設計作業に基づいて、その基本設計段階におけるソフトウェア・プロセスの分析を試みたので、その成果及び評価について報告する。

最終的には、ソフトウェア・プロセスを核として、さらに適切なオブジェクト例および練習問題等を含む若年技術者向教育ツールを目指しているが、今回の成果は、その基礎をなすものである。

SIG SE 71-18

SOFTWARE PROCESS ――――
―――― CASE STUDY OF A REAL-TIME PROCESSING SYSTEM

Sumio MOCHIZUKI　Akira YAMAUCHI
Mitsubishi Space Software
MSS Yamazaki Bldg.299,Yamazaki,
Kamakura-shi 247,Japan

Takuya KATAYAMA　Masato SUZUKI
Tokyo Institute of Technology
1 Ookayama,Meguroku,Tokyo 152,Japan.

In Software Process paradigm, it is very important that we observe and analyse how the engineers design software and how they exchange imformation with one another.

In this case study we take up one of the real-time processing systems and try to analyse how its design was processed.

We also give the results and evaluations of our case study.

We are planning to design an instruction tool for the inexperienced engineers, using this software process concept with some adequate sets of the objects and exercises.

1．はじめに

近年、電子計算機応用システム（以下電算機システムと略す。）の開発要員であるシステム・エンジニア、ソフトウェア・エンジニアは、慢性的に不足して居るがその傾向は今後益々顕著になっていくと予測されている。

しかし、システム・エンジニア、ソフトウェア・エンジニアの早期育成は困難であり、長期間の技術蓄積、経験の積み重ねが必須となっている。

システム・エンジニア（以下ＳＥと略す。）育成の難しさの原因は、一つにはソフトウェア設計技術そのものが未成熟であり、方法、手法が標準化されていない為、未だに技術者各個人の力量に任されている部分が相当に多いことにある。

ソフトウェア設計業務における技術者のBehavior、考えの過程（PROCESS）、手順は、あまり明らかになっていないが、もし、詳細に分析、解明することができたならば、おそらく技術者各個人の間に相当な相違が見られるであろう。

この様な現状においては、設計技術の確立、伝承は難しく、若手技術者の急速な育成は困難である。また、設計業務において問題が発生したとしても、その原因を究明することは極めて難しいことになろう。

最近、この様な状況のもとで、現場におけるソフトウェア設計作業そのものの観察、分析を中心として、その過程を明らかにしようとする、いわゆる' ソフトウェア・プロセス' の研究が盛んになってきた。

本論文では、現実のプロジェクト（チェックアウト・システム開発プロジェクト）の担当技術者のインタビュー、生成物（オブジェクト）の収集、整理、分析をすることにより、技術者の設計過程を明らかにしようとするものである。

2．ソフトウェア・プロセスの目的及び効果

ソフトウェア・プロセスは、設計の現場における作業の実態を素直に直視し、これを原点として出発することを基本としている。

ソフトウェア設計過程が明らかになるならば、これを応用することによって、次の様な効果が期待できる。

（1）ソフトウェア設計作業の標準化
（2）若年技術者の育成、早期戦力化
（3）設計技術の向上、生産性向上（支援）
（4）ソフトウェア設計作業の可視化、管理の容易化
（5）ソフトウェア・プロセスの再利用

これ等の項目は、何れも電算機システム開発部門、応用ソフトウェア開発部門にとっては、極めて重要なものである。

この数年間、ソフトウェア・プロセスに関する論文が、数多く発表されている。しかし、ソフトウェア・プロセス追求のための方法論が多く、また、実験室的なプログラミングの開発を例にとった解析結果が多い。

本論文では、現実の製品を開発するプロジェクトにおけるソフトウェア開発プロセスを分析し、評価を加えることとする。

我々の目的は、このソフトウェア・プロセスの解析を通じて、標準的な（理想的な）ソフトウェア・プロセスの存在を探り、最終的には、若年層の育成・教育ならびにシステム設計支援のためのソフトウエアツールを確立することにある。

3．実時間処理システム及びその設計

本論文においては、実時間処理システム分野における実用システム開発のプロセスを対象として取り上げることとした。

この分野の中で、我々は次の様な考えのもとに、分析対象システムを選択した。

（1）現実のプロジェクトには、様々な外部要因（コスト、納期、体制、仕様等の制約条件）が付与されるが、これがあまりに強すぎると変則的な設計経緯を渉ることになりかねない。
正常な設計プロセスを抽出するためには、そのプロジェクトの履歴、それをとりまく環境を十分吟味する必要がある。

（２）あまりプロジェクトの規模が大きく、かつ複雑になると、設計プロセスの大筋を把握することが難しくなる。

（３）初めて設計するシステムの場合、試行錯誤が多い。これも重要なプロセスの一つではあるが、理想的な設計プロセスを抽出するためには、同種のシステム設計を何回か経験した結果、設計プロセスとしてリファインされたものであることが望ましい。

以上の事項から、我々は、（人工衛星の）チェック・アウト・システムを取り上げることにした。

図１に、同システムの概念図を示す。人工衛星とチェックアウト・システムとは、近接して設置され、人工衛星各部の状況を示すテレメトリ・データならびに人工衛星を制御するためのコマンド・データを送受信し、その動作を確認するものである。

また、各部の温度等を直接計測するためのケーブル（ハード・ライン）も備えている。

本システムは、小規模とはいえ、実時間処理システムの特長を備えた典型的なシステムである。

この種のシステムを設計するためには、ソフトウェア開発作業のみならず、ハードウェア、品質管理、プロジェクト管理と多方面のプロジェクト活動が必要であるが、その作業項目（WBS:Work Breakdown Structur）を図２に示す。この説明については、ここでは省略し、本論文ではこの中の応用ソフトウェア開発（300番台の項目）を中心に論ずることとする。

従来、我々はこの様な応用ソフトウェア設計に際しては、ウォーターフォール・モデルならびに、それに対応する工程表を使用してきた。（図３）

しかし、ウォーターフォール・モデルには、次の様な問題がある。

（１）実際の設計作業の中では、ウォーターフォール・モデルにおける各設計フェイズの開始時点および終了時点が、あまり明確ではない。

（２）フェイズの分け方がマクロ的すぎるため、設計作業の実態を把えきれない。

（３）現実のプロジェクト遂行上避けることのできないリスク管理が表現されていない。

ここで、この様な課題を解決できる新しいソフトウェア設計プロセス・モデルへのアプローチが必要となってきた。

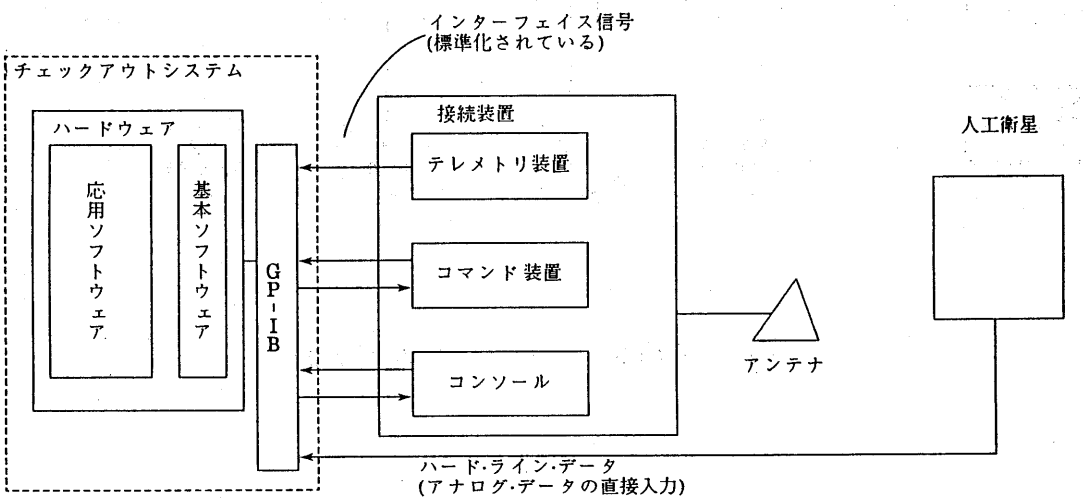


図1　DAS（Data Acquisition System）概念図

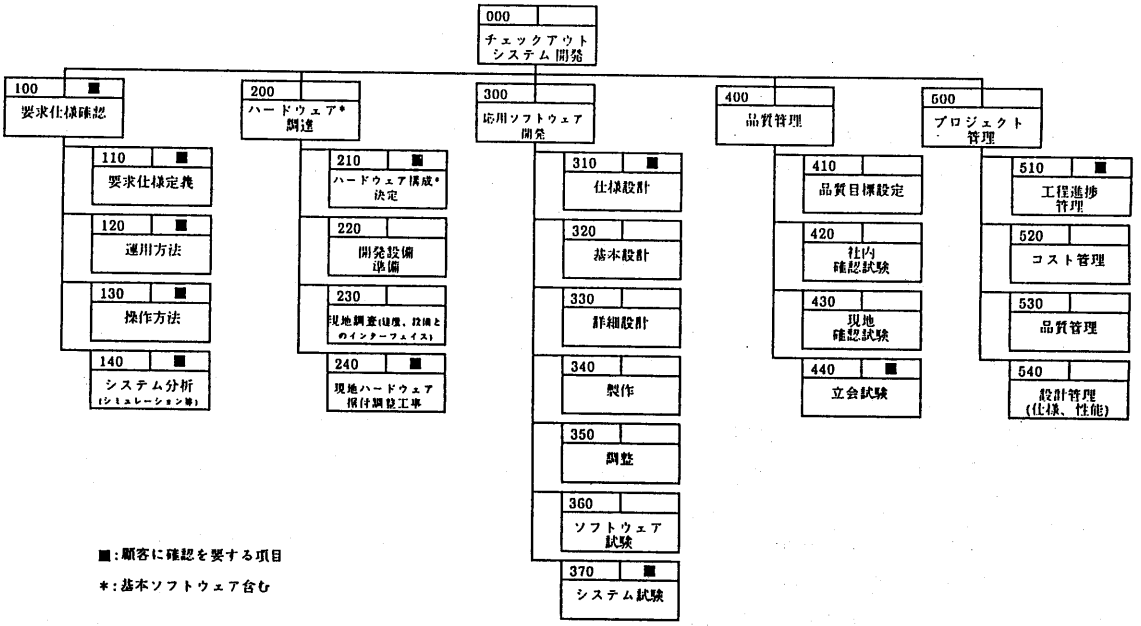


図2　チェックアウトシステムWBS

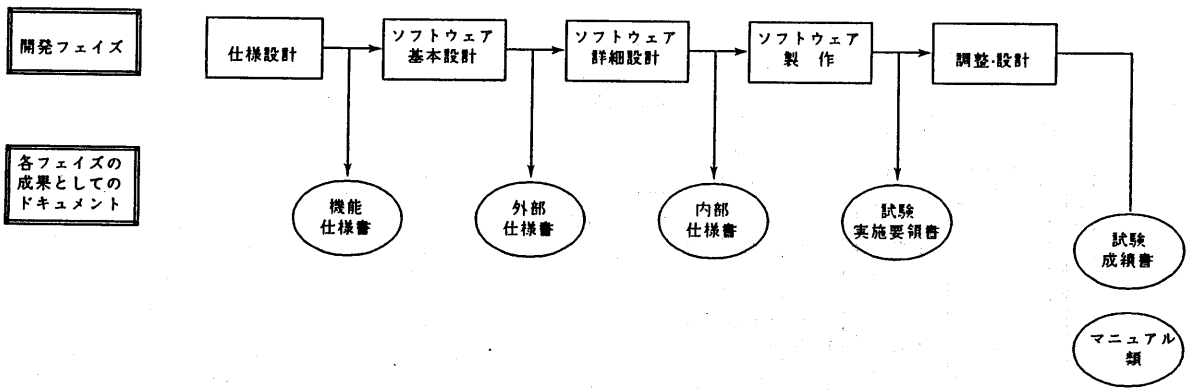


図3　応用ソフトウェア設計概念図

4．ソフトウェア基本設計

前章で述べたWATERFALL MODELの中でソフトウェア基本設計は、ユーザ側の要求機能をまとめ詳細にブレーク・ダウンし、それを実現すべくシステムの設計に反映させる重要な段階である。

この段階では次に示すようなソフトウェア設計の概要が決定され、その検討結果は、外部仕様書としてまとめられる。

（1）実現すべき機能、運用方法　（2）システム性能　（3）品質　（4）詳細工程
（5）応用ソフトウェア概要（フィージビリティ・スタディを含む。）

即ち、このフェイズは、次の意味で、システム設計上（応用ソフトウェア設計上）最も重要な階段であるといえる。

（1）ユーザの要求仕様をシステム機能として包含する。
（→ユーザの要求をどこまで含められるか、ユーザとの誤解はないか等）
（2）システム機能をソフトウェア設計に反映させる。
（→システムのフィージビリティ・スタディ、実現性チェック等）
（3）開発すべきシステムの概要（ハードウェアの構成、応用ソフトウェアの内容・構成）を決定する。

基本設計工程は、応用ソフトウェアの規模、難易度がきまる段階であり、その後の詳細設計以降の工程（外部仕様書にもとずき忠実にソフトウェアを製作する）に比較して、格段に重要な意味を持つ段階である。

システム開発プロジェクトの成否の大部分は、この基本設計フェイズにかかっているといっても良い。

我々は、システムのライフサイクルの中におけるこの基本設計の重要性を日頃から痛感しており、ソフトウェア、プロセス抽出、分析の対象として、この基本設計フェイズを取り上げることとした。

5．ソフトウェア基本設計のモデル化

チェックアウトシステムの基本設計を対象としたソフトウェア・プロセスを次の様な方針でまとめることとした。

5．1　モデル化の進め方

チェックアウト・システムの基本設計プロセス・モデルをまとめるために、過去に同じシステムの設計を担当したＳＥ６名を含むチームを編成した。

まず、最新のシステムのオブジェクト（外部仕様書及びその中間オブジェクトとしての技術検討書）を集め、その設計手順を現場で行われているようにできる限り忠実に再現することとした。

目標とするソフトウェア・プロセスのレベル（細かさ）としては、入社２～３年のシステム開発経験を持つソフトウェア技術者が理解できる程度に定めた。

5．2　ソフトウェア・プロセスの記述法

ここでは、ＨＦＳＰ（Hierarchical and Functional Software Process）Ｍｏｄｅｌ（階層的関数型ソフトウェア・プロセス・モデル（詳細は、文献２参照。））に沿ってソフトウェア・プロセスを記述する。

即ち、概要は以下の通りである。

（1）ソフトウェア・プロセスをその入出力関係からとらえ、入力から出力へ変換する小さなプロセスの集合として記述する。各プロセスの機能のみならず、その入力、出力を明確に定義する。
（2）各プロセスの機能が単純でないときは、これ等をさらに簡単な機能のサブ・プロセスに分解し、同時にその入力、出力を明確に定義する。
（3）この分解の作業を各プロセスの機能が十分単純化されるまで続ける。

この様に、プロセスの静的かつ宣言的な記述形式をとることにより、明瞭で、理解し易いプロセスの記述が可能となるのである。

5．3　基本設計のソフトウェア・プロセス・モデル

前節の方針に従ってチェックアウト・システムの基本設計手順を分析した結果、図4に示すソフトウェア・プロセスを得た。

全体は、6個のプロセスから成り立っている。

図には、各プロセスの入力ならびに出力情報が記述されており、また各プロセス間の入力、出力情報の授受関係ならびに各プロセスの実行順序が示されている。さらに、右側には各プロセスの出力がそのプロセスのどの入力から生成されるかの因果関係を示して居り、これにより、入力と出力の対応関係がより明確に表現されている。

図中、第1、2、5、6番目のプロセスは、ソフトウェアの要求仕様書を次第に詳細化し、それをもとにソフトウェア設計を進め、さらに、Feasibility Study（第6番目のプロセスに含まれる。）を行う過程を示している。

第3、4番目のプロセスは、システム全体のハードウェア、基本ソフトウェアの仕様を決定する段階であり、通常第2番目のプロセスと並行して進められる。

図4の6個の各プロセスの機能は未だマクロ的過ぎるため、各々をさらに簡単な機能のサブ・プロセスに分解した。

その内の1つ、第2番目のプロセス（チェックアウト・システム概念設計）を分解した結果を図5に示す。

これ等の設計プロセスは、標準的な（理想的な）設計手順を示しているのであり、始めてこのシステムを設計する場合の試行錯誤、繰り返し、手戻り等については、ここでは表現されていない。

従って、実際の設計手順の大筋を示しているといえよう。

6．リスク管理

最近のように、電子計算機技術が向上し、コスト次第で大概の要求仕様の実現が可能となり、またユーザ側が投資効果を強く求めるようになると、システムに対する要求仕様も非常に高度かつ厳しい内容になる。（図6）

一方、システム開発に投入できる資源（コスト、期間、人材等）には限りがあり、ここに相反する

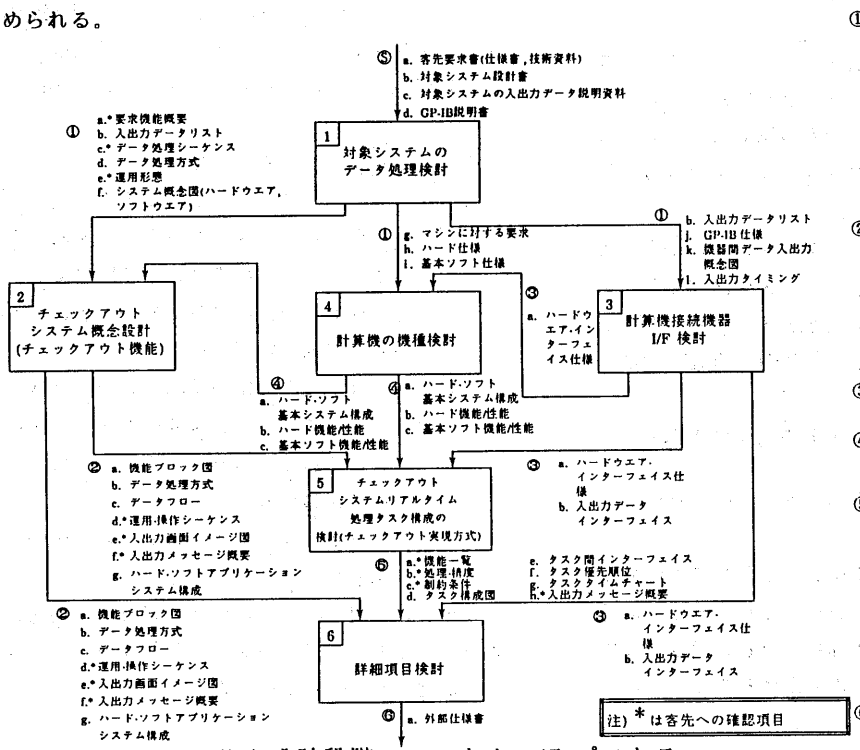
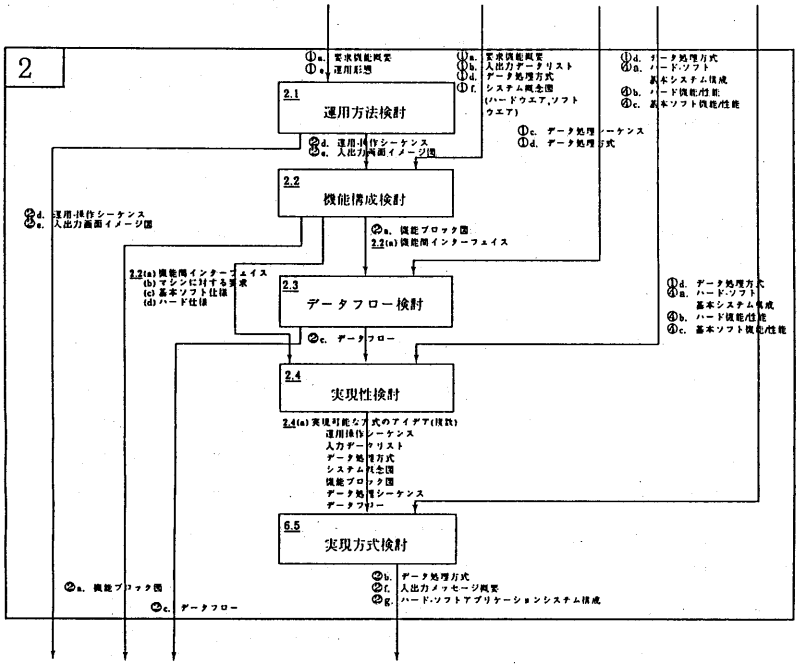


図4　基本設計段階のソフトウェア プロセス



2.2a ← ①b , ①d , ①f , ②d
b ← ①a , ①d , ②d
c ← ①a , ①d , ①f , ②d
d ← ①b , ①d , ①f

2.4a ← ①d , 2.2(a) , 2.2(b) , 2.2(c) , 2.2(d) , ②c , ④a , ④b , ④c

② a ← ①a , ①d , ①f , ②d , ②e
b ← 2.4(a) , ④a , ④b , ④c
c ← ①c , ①d , ②a
d ← ①a , ①e
e ← ①a , ①e
f ← 2.4(m)
g ← 2.4(a) , ④a

図5　第2ブロック チェックアウトシステム概念設計
(チェックアウト機能)のソフトウェア プロセス

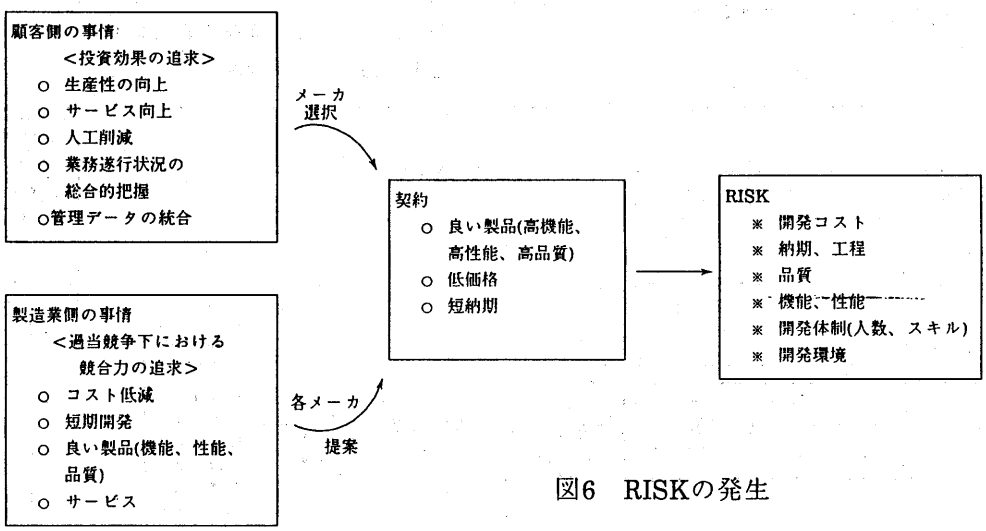


図6　RISKの発生

条件が発生し、これを解決するためにリスク管理が必要となるのである。

Boehmは、ソフトウェア設計の手順と共にリスク管理手順を定め、これらをサイクリックに繰り返してゆくモデル(Spiral model)を提案している。

現実のシステム開発においては、ソフトウェアの設計を進めつつ、その推移を監視し制御するリスク管理が必要となるのである。

特に、初期設計（仕様設計、基本設計）においては、ソフトウェア全体の仕様が明確化、詳細化される。

その設計途中段階においては、図7に示すように、最終的な仕様及びソフトウェア構想を予測し、それに伴う制約条件を見積って各々の制限値*を超えないように設計を制御するのである。

即ち、ソフトウェア・プロセスがソフトウェアに関するオブジェクトを生成する機構とすると、その生成過程において各制約条件（コスト、工程等）が各制限値を超えないように制御する仕組みがリスク管理であるといえよう。（図7）

これも一つのソフトウェア・プロセスととらえ今後さらに詳細に追求してゆく積もりである。

＊ 設計開始時点に設定される。

7．今回得たソフトウェア基本設計プロセス・モデルの評価

第5章にて得たソフトウェア基本設計プロセスならびにその作業過程について、我々は次のような評価および反省をしている。

（1）今回得たソフトウェア・プロセスは、今後さらに充実させ、具体例等を準備することにより、若年技術者教育に役立つものになると考える。

（2）本研究をスタートする前は、検討メンバが全く異なる手順で設計しているのではないかという懸念があったが、今回の結果は全員の納得、合意を得ることが出来た。これは、標準的なソフトウェア・プロセスの存在を示すものである。

（3）このプロセスの大部分は、チェックアウト・システムだけでなく、他の実時間処理システムに対しても共通である。

（4）設計プロセス各項目と、入出力オブジェクトだけでは、設計そのものを十分表現できぬ部分がある。

（5）設計する上で、設計者が過去に担当した設計の経験が生かされているところが何箇所かある。この経験の表現を工夫する必要がある。（実例等）

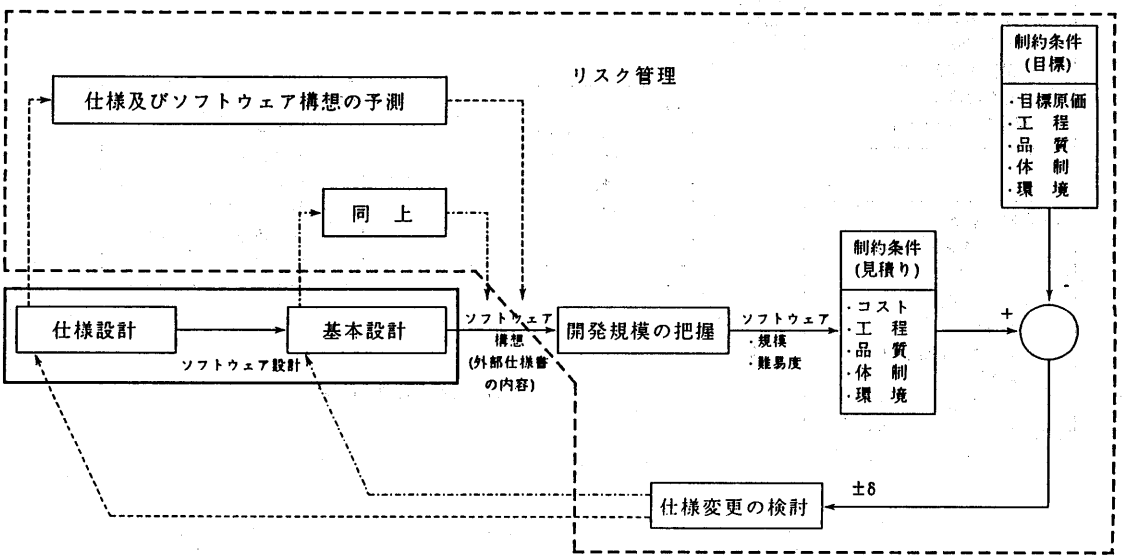


図7　ソフトウェア設計とリスク管理との関係

（6）同時に併行して進められるプロセス間のオブジェクトを網羅することが難しい。（各プロセスを分担するとしても担当者同士の会議等により頻繁な情報（オブジェクト）交換を行っている。）

（7）設計途中での手戻り作業の表現が必要。

（8）プロジェクトに異常事態が発生した時の対処も含む必要がある。（CONTINGENCY PLAN）

8．現実のプロジェクトのソフトウェア・プロセス導出作業の注意と改善。

今回の作業において、気付いた点を以下に示す。

（1）現実のプロジェクトは、規模が大きく、その中の設計プロセスの大筋を抽出することは難しい作業であり、実際にその仕事に携わったベテラン技術者が行うことが望ましい。

（2）現実のデータそのものの整理には、地味な努力が必要である。オブジェクト、手順の欠落に良く注意する必要がある。

（3）現実のデータを整理した結果、あまり一般的な結果を求めようとすると実際の設計プロセスを歪めたものになりかねない。

（4）オブジェクト同士の因果関係に関しては、次の事に注意する必要がある。

a．1つのプロセスの出力オブジェクトを生成するために必要な全てのオブジェクトがそのプロセスの入力オブジェクトにあること。

b．あるオブジェクトが一連のプロセスの途中で突然に現れたり、消えたりしないこと。

（5）実際の設計作業は、非常に複雑な思考の上に成り立っており、少数のプロセス及び入出力オブジェクトで代表させる為には、相当な割り切りが必要である。
また、一部のプロセスに対しては、次の様な補足が必要である。

・プロセスの作業内容の補足説明（詳細説明）

・解り易い設計例の表示（オブジェクトの例を含む。）

・オブジェクトのまとめ方（様式、表型式等）

9．今後の方向

本論文では、実用システム開発プロジェクトにおいて最も重要な基本設計段階における設計プロセス及びその入出力オブジェクトを分析した。

我々は、最終的には、若年のシステム・エンジニアの教育システム、さらに設計支援ツールを目標としている。未だ第一歩をふみ出しただけであり、今後に残された課題は多いが、次のような方向を目指している。

（1）最終的には、エンジニアリング・ワークステーションに組み込むことを想定した、教育用ツール、さらに設計支援ツールを目標とする。

（2）今回得たソフトウェア・プロセスは、標準的なプロセスのみを分析したものであり、今後さらに手戻り作業、繰り返し、リスク管理等を含めたプロセスの充実が必要である。

（3）各プロセスの入出力オブジェクトの実例を付加して理解し易くする。

（4）基本設計プロセス全体に対して、練習問題（初級、中級、上級用等）を用意し、学習者がこれにより、設計の実例をもとに体験できるようにする。

（5）各プロセスの出力オブジェクトをまとめるための標準様式を用意し、設計作業の効率化を計る。

（6）設計の進行と共にリスク管理がし易い様に、設計途中で制約条件の見積りを支援する機能を用意する。

（7）設計を変更した時、その影響をシミュレーションする機能も考慮する。

（8）今回求めたものは、チェックアウト・シ

ステムの設計プロセスであるが、多少異なったシステムを対象とした場合、設計プロセスをどの様に変更、調整すれば良いか。ソフトウェア・プロセスの再利用の適合範囲をどの様に拡げるか。これは今後の問題である。

１０．まとめ

実時間処理システム分野の典型的なものとして、チェックアウト・システムをとりあげ、その基本設計段階における設計過程をできる限り忠実に分析した。

作業は、数人が分担して詳細検討した結果を持ち寄りそれを全員で討議し、調整すると共に先へ進めるというステップを繰り返した。

作業開始時点では、ソフトウェア・プロセスそのものが、各エンジニア特有であり、各人全く独特のプロセスとなるのではないかと危惧の念を持っていたが、実際に進めてみると、今回の２段階のレベルにおける各プロセスは全員納得できる内容であった。

我々は、これに意を強くして、さらに追求して行く積りである。

最終的目標として、若手ソフトウェア技術者の教育ツール、さらに設計支援ツールを目指しているが、まずその基本としてのソフトウェア・プロセスをまとめ、その上に教育、設計支援等に必要な機能を実例、シミュレーション機能、練習問題等と共に追求するというように着実な進め方をしたいと考えている。

参考文献

1. L. Osterweil, "Software Processes are Software too, "Proceeding of the Ninth International Conference on Software Engineering, Montery, California, pp. 2-13, April, 1987.

2. T.Katayama, "HFP, a Hierarchical and Functional Programming," Proceeding of the 11th International Conference on Software Engineering, pp. 343-352, 1989.

3. B. W. Boehm, "A Spiral Model of Software Development and Enhancement, "ACM Software Engineering Notes, vol. 11, no. 4, pp. 14-24, August, 1986.

4. R. Balzer, T. E. Cheathan, Jr., and C. Green, "Software Technology in the 1990's; Using a New Paradigm, "Computer, vol. 16, no. 11, pp. 39-45, 1983.