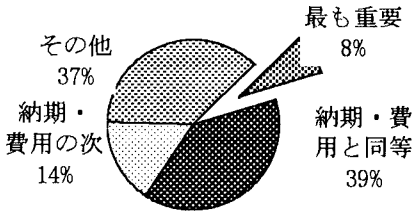
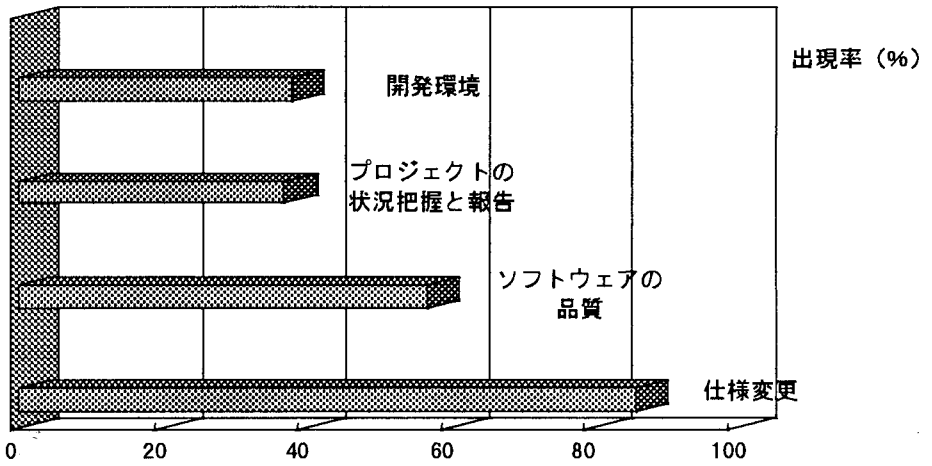
II ソフトウェアの品質管理

1 品質の重要度

ソフトウェア品質の困難な点は、利用者の要求ニーズを把握して、現有の資源で高い品質を考慮しなければならない。
ソフトウェア開発における意識調査 (図II－1)で“品質を最も重要と考えている”と答えた技術者の割合は8%と低い割合で、"納期・費用を優先し、品質はその後に"という意識が強い。しかしソフトウェア開発の現場において仕様変更などの次に品質の問題は主要なものとして高い値を示している。(図II－2)



図II－1 開発の中での品質の重要性



図II－2 プロジェクトの抱える問題点

2 ソフトウェア品質管理における開発と保守

ソフトウェア品質の管理は、新規開発時の品質向上は勿論であるが、その後のメンテナンスの容易さ、再エンジニアリング作業の取り組みやすさがとても重要である。

数年前までは、ソフトウェアの逆エンジニアリングや再エンジニアリングという言葉は、聞き慣れないものであった。以前はソフトウエアの逆エンジニアリングといえば、ＯＳやコンパイラなど、システムソフトウエアにおける内部解析の手段を意味していた。

次々と開発が行われ、ソフトウェア資産が蓄積される中、保守業務の割合が増大し、保守に費やすデータ処理部門の作業量やコストが新規開発のそれらを大きく上まわり、新規開発を圧迫される企業が出てきた。米国などでは、保守量がデータ処理部門のコストの９０%以上に達している企業が少なくないし、日本においても近年、同じ傾向が見られる。
このような背景から、保守の効率化のために、逆エンジニアリングと再構造化の技術が注目されるようになった。

また業務変更・拡張に対応して、ゼロから新たに開発するよりも、既存のプログラムを作り直した方が有利な場合が多々ある。そのために再エンジニアリングが行われるケースが増えている。一般に、修正がコードの全体の５０%以下なら、再エンジニアリングを行った方が新規に開発をするよりもコストが小さいと言われている。

注１　逆エンジニアリング…プログラムから構成要素、関係を解明すること。
（再文章化、設計復元、業務工程分析等）

注２　再エンジニアリング…既存のシステムの分析・理解→新しい形態に変更→実現
という処理の流れ

3 開発と保守関連の定義

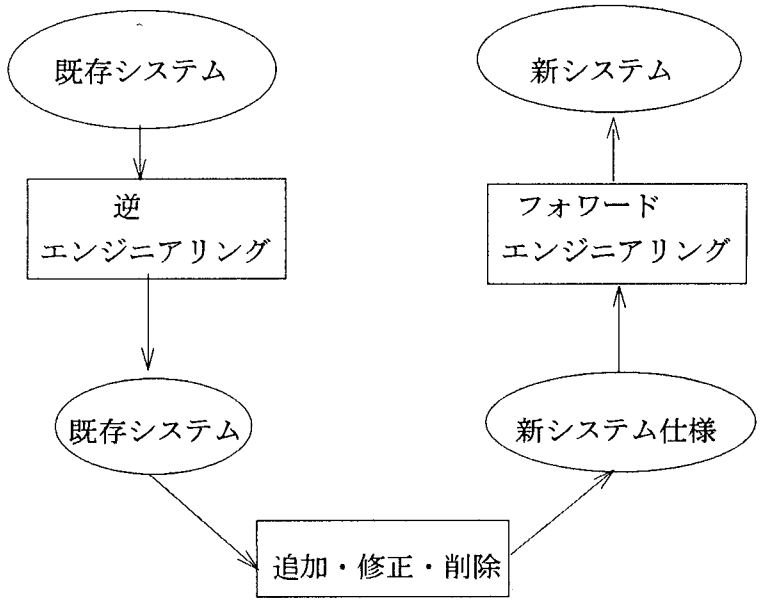
再エンジニアリング開発の流れを図示する（図Ⅱ－3）と次のようになる。

（1） 既存システムに逆エンジニアリングをかける。

（2） （1）で得られた設計仕様書に追加・修正・削除を行う。

（3） （2）の結果である新システムの設計仕様書にフォワードエンジニアリングをすることにより新システムの開発を行う。

このような手順で、新システム開発においても多くの場合、既存のシステムの理解が必要であり、日頃からきちんとプログラムを保守管理する必要がある。



図Ⅱ－3 再エンジニアリング開発の流れ

4 プログラムの保守

プログラムの保守にはまずプログラムを理解しなければならない。ところが開発後長い時間が経っている場合は、多くのパッチが当てられていたり、文章化が不十分、不正確であったりして、コードの解析が非常に困難な仕事となる。そのため一般的に保守作業の労力・時間の約５０％がプログラムを理解のために費やすといわれている。（図Ⅱ－4） したがって、広義の保守は、図Ⅱ－4に示すように新規アプリケーションと既存アプリケーションへの適合化のための作業を含めて示される。

保守作業の比率	アプリケーション理解５０％	修正・拡張の実現　５０％

図Ⅱ－4 広義の保守と仕事内容

ソフトウェアの大きさや複雑さが増せば理解が一層難しくなり、保守のコストは大幅に上昇する。大規模ソフトウェアの保守は想像をはるかに超えて高価である。

5 品質管理における標準化

保守作業を少しでも軽減し、先に述べた再エンジニアリングを効率よく行っていくためには、ソフトウェアを部品化し蓄積して必要に応じて取り出し「利用」していく体制が必要である。また、部品の再利用を行うためには、標準化が行われなければならない。

品質管理と標準は密接な関係にあり、品質を向上させてゆくためには標準化の推進が必要であり、標準化が進めば品質は自ずと高いものになっていくものである。

下記に標準化の作成を行うための例として.2つの項を記述する。

標準化作成

① 開発作業手順の効率化標準

開発作業手順におけるバグゼロのための７つの設計原理。

単純原理　巧妙なテクニックは使用しない。シンプルに。
一様原理　同一内容はワンパターンで済ませる。
対称原理　対称性を利用する。
階層原理　主従関係を常に意識した体系とする。
透過原理　フラグのON/OFFの制御は控える。
明証原理　複雑なロジックは必ず分かりやすい証明をつける。
安全原理　曖昧さのある所は、安全サイドで設計・作成する。

７つの設計原理（証明原理）は、バグを作り込まないようにしていく中で原理として確立されたものであり、バグの根本的原因の大半は、この７つの設計原理から証明できるといわれている。７つの設計原理は、「誰でもが作業の進行中、発生する可能性のある間違いを、避けることを可能にするシンプルな法則」を具体的に言い表したものであり、このことを常に念頭において、設計・レビューを行うことによりバグの発生を最小限に止めることができる。

② 品質保証作業標準

品質保証作業の標準化は、次のような項目について実施される。

品質保証項目
品質保証指標
作業標準
教育
品質の作り込み（チェックリスト、ドキュメント基準、ワークシート）
誤り検出　（レビュー基準、テスト基準、チェックリスト）
品質評価　（収集、集計、分析、評価、アクション）

6 標準化推進

標準化は、開発環境の整備、ルール作りが非常に重要な要素であり、ツール化の促進(CASE) などのプロジェクト管理技法の整備充実が有効手段である。標準化の浸透を促し、常に一定のレベルを保つためには、しっかりとしたチェック機関（人）を設計部署とは別に配置し、権限を与える必要がある。なおこの時、組織、権限等は携わるものの意見でのボトムアップで決定されることが理想的である。