Ⅱ　生産性向上の発展とシステム

１．ＩＳ部門の生産性

コード化の目的は、人間やコンピュータが情報を容易に処理することにあり、情報の体系化、処理の効率化、さらにはそれらの標準化（設計、コード自動生成、テスト）をはかろうとするところにコード化の主目的があます。その特性を生かすために、コードジェネレータと設計用ルールが必要になります。

（１）大規模開発チーム

大規模開発チームでの生産効率に関して、技術者間での意志疎通の不整合による誤りが多くなる傾向があります。その解決策として、ＣＡＳＥツールを前提におき、小規模な開発単位ごとに正規化されたデータ定義による開発を行います。各開発単位間とのインターフェースは、共通のデータモデルを経由して行われるために、同一データの連携利用による管理の容易により、効率よく早期開発が可能になります。

（２）大規模なプログラム

基本計画の段階で決定したシステム化の対象が大規模な場合、あるいはその機能が多様な場合などには、それらをさらに小さな機能を持つシステムに細分化することによって、システム１つあたりの大きさや機能を適当なレベルに抑えるのが一般的です。こうすることにより、システムは階層的な構造を持つことになります。このとき上位のシステムを構成する下位の各システムのことを、上位のシステムに対する「サブシステム」と呼びます。システムとサブシステムの関係は、あくまで相対的なものです。サブシステムを定義する際のポイントとしては、「時間的、データ的にみて関連の深いものをまとめること」「他のサブシステムとのインターフェースが最も少ないところをサブシステム分割の境界とすること」などがあげられます。

２．再利用可能な設計とコード

生産性を向上させるためには、設計情報、データモデル、プログラム・コードなどを、積極的に再利用することが重要です。再利用可能な設計情報を部品化して、ＣＡＳＥツールのエンサイプロペディアの中に登録しておき、必要に応じて取り出し、変更を加えて使用する必要があります。

「ソフトウェアの再利用」
⇒　ソフトウェアの開発に必要とされる知識を何らかの形で標準化し、これを繰り返し利用し、その知識から新たなソフトウェアを想像することです。

「標準化」の形式には、次の二つの形式があります。
⇒　ソフトウェアそのものを標準化して再利用する形
　　ソフトウェアのデータ構造や機能を標準化パッケージにし、さらに細かなモジュール群に展開した「標準部品」の形式にして、繰り返し使用します。（再利用技術）

⇒　ソフトウェアを作成する課程を標準化し、これを自動化するツールを提供する形
（自動化技術）

再利用の基本的な考え方として、「部品による再利用」、「生成による再利用」、「オブジェクトによる再利用」があげられます。

（１）部品による再利用

①　部品化方式

ソフトウェアの再利用を促進するには、ソフトウェアを構造的にとらえ、その構成要素を標準化し、再利用の単位（部品）とします。標準化する部品作成には、利用頻度の高い共通部分のセットがメーカー等から支給されるべきであり、またユーザー自信が容易に部品を作れる用に、標準的書式とそれによって部品を作りやすくするエディターが必要です。

②　部品ライブラリー

ａ．部品を系統的に蓄積したデータベース　⇒　部品ライブラリー
ｂ．部品の開発言語は、基本的には何でも良いのですが、原始プログラムのレベルで利用することがあるので、言語はなるべく統一すべきです。
ｃ．部品名は、検索時の効率を考慮して、機能別、使用頻度別などでグルーピングするというような工夫が必要です。
ｄ．部品の数が増加すれば、種類別の「サブライブラリ」が必要です。
ｅ．部品登録簿の作成　⇒　部品名、登録者名、登録年月日、利用頻度等

（2）生産による再利用

①　リエンジニアリング（再エンジニアリング）

a．対象システムを調べ、新しい形に変更してそれを実行すること。
b．リバースエンジニアリングとフォワードエンジニアリングを伴います。
c．新しい要求に対応するための変更、追加等を伴うこともあります。

②　ソフトウェアの再構造化

構造化されていないプログラムを、構造化されたプログラムツールを使用して変換すること。この考え方は、データや設計、要求仕様などにも対応できます。

（3）オブジェクトによる再利用

①　オブジェクト指向設計

a．プロセス中心設計

データはプロセス（手続き、処理主体）に付随するものと考え、まずプロセスに注目して設計を行います。従来から行われている方法です。

b．データ中心設計

まずデータに注目し、それを元にデータ構造を設計します。プロセスの構造はデータ構造への依存関係に基づいて設計します。

c．オブジェクト指向設計

データ中心設計の考え方を更に推し進め、カプセル化という考え方によって、データとプロセスを一体化し、それを「オブジェクト」としてとらえて設計を行うアプローチです。

②　オブジェクト指向と再利用

カプセル化によって、データとプロセスの関係は密接になり、データの局所性が高くなります。データ構造の変更は、プロセスを介してデータを利用する側には見えにくくなり、また、データ変更されてもその影響を受けるプロセスの範囲が限定されることになります。このようにデータ局所性が高い構造は、ソフトウェアの再利用に適した構造であります。

３．システムの発展的ライフサイクル

長期的視野にたってソフトウェアを改良するためには、データとプロセスに関する構造化されたモデルが必要であり、人間とコンピュータとのインターフェースをより効率的に行うために品質の良い開発支援ツールを使用して、システム間の整合性を維持し、標準化された再利用可能な設計情報やプログラム・コードを最大限に使用したり、継続的に機能追加を行っていけるような環境の中で、分析、設計、コード自動生成、テスト、改善を一貫した処理開発サイクルで構築する事が必要です。このことが企業情報システムを長期間にわたって発展させていくことになります。

（１）システムの保守

システムの保守は、ＣＡＳＥツールで設計変更を行い、変更された設計情報からプログラム・コードを自動生成することにより、システム変更を簡単に、しかも短時間で行うことができます。一貫した体系された構造化技法を適用して、保守の容易なシステムを早期に開発することが出来るようになります。従来の手作業による統一性のないシステムは、最新の技術を取り入れて構築された新しいシステムへ、段階的に移行することが必要です。

（２）リバース・エンジニアリング

旧（既存）ソフトウェアシステムの仕様に対し追加要求（追加、修正、削除）し、改良された新システムの仕様にして新規ソフトウェアとして再利用を行います。

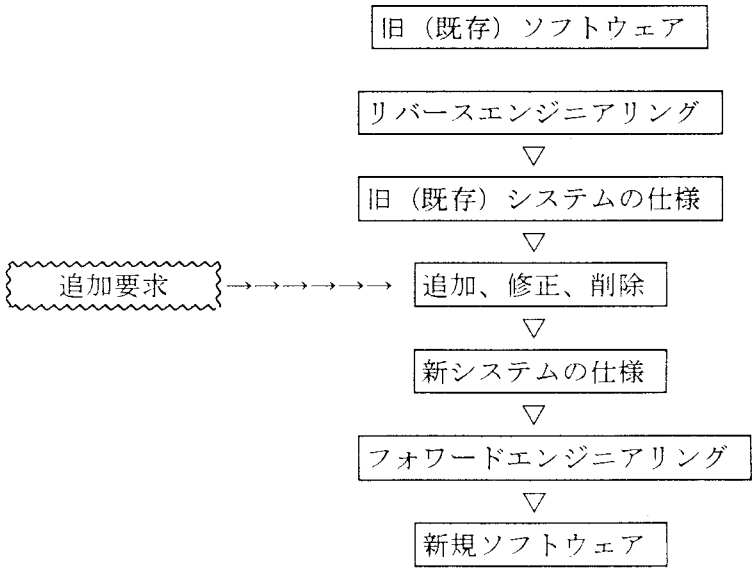

図３　リエンジニアリングによるリバースエンジニアリンクの移行