

リエンジニアリングのためのCASEツール

本 村 昭 二
ケーステクノロジー株式会社

情報システムは増え続けている。これらの情報システムを保守するのにソフトウェアを直接変更することは良案ではない。

ソフトウェアは「業務仕様」「稼働環境」「実現技術」から成りたっている。従って新しい適用業務を開発することは各要素を設計し、自動的にソフトウェアを生成する。各要素を個別に保守することを何時でも条件に合致した完全なソフトウェアが生成出来る。

また、既存ソフトウェアから「業務仕様」を抽出することで新しいソフトウェアを生成出来るようになる。結果として以降のソフトウェアの保守は不要となる。

CASE TOOL FOR REENGINEERING

Shoji Honmura

Case Technology Co., Ltd.

Data processing system has been increasing. To maintain the application system, it is not proper method to change software itself. Software consists of three components, such as application requirement, operating environment and implementation technology.

To develop new application system, we design each component, afterwards generate software automatically.

We can maintain those application system to change each component individually, and generate software to meet new specifications.

To analyze the existing software, we can abstract requirement and generate new software and consequently we do not need to change existing software.

1. リエンジニアリング要求の背景

情報システム（コンピュータ化された情報処理）は1度構築され運用が始まると原則として廃止することはない。給与計算にしろ、会計処理にしろ、組織内におけるその情報処理機能が不要になるのではない限り1度コンピュータで処理し始めたことを手計算なり、他の方法に置き替えることは實際上考えられない。従って、誕生した情報システムは存在し続ける運命である。

一方この情報システムは実務上使用に耐える状態にするには、変化に対応して絶えず維持、改善をすることが必要となる。その主な要因は大なり、小なりの業務の要求仕様の変更であるが、その他にハードウェアやシステムソフトなど稼働環境の進歩への対応、潜在的な品質不良の除去などもある。

この情報システムの存続するための維持、改善には大きく2つの方法がある。1つは保守、1つは再構築である。この2つの方法には根本的な違いがある。保守は現行の情報システムを稼働させるソフトウェアの存続を前提条としており、一方再構築は現行ソフトウェアの維持使用をやめて、全てを再度開発する方法をとる。

図1に示すように新規開発、保守、再構築の作業負荷は時間とともに保守と再構築の比率が高まっていく。情報システム部門は新規開発ではなく、保守と再構築の作業負荷問題を解決しなければならぬ状況にある。

この再構築を従来は新規開発と基本的には同じ方法をとっていた。少なくとも既存のソフトウェアの活用という手段はなかった。しかしながら、時間とともにソフトウェアは累蓄され、再構築の対象が増すに従って、現実的に不可能になりつつある。このことがリエンジニアリング既存のソフトウェアを利用した再構築が求められてくるようになった大きな要因である。

その上にエンジニアリングワークステーション中心に高性能、低価格のハードウェアが提供され、かつCASEを中心としたソフトウェアの開発環境の進歩がその実現の可能性を推進する要因となっている。

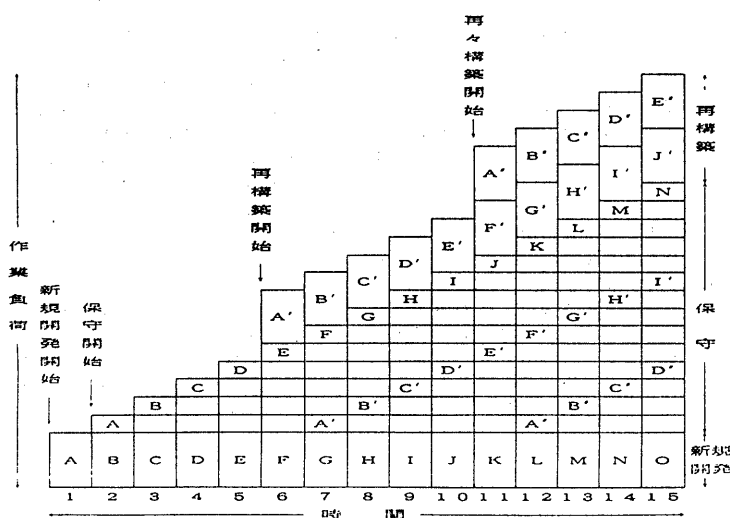


図1：(伝統的方法)による新規開発、保守、再構築の作業負荷

再構築の第一の目的は仕様、環境、技術などの変化により保守の範囲では維持不可能になったソフトウェアを最新の条件を前提で再度設計、製造することである。しかし、この再構築を従来の方法で行ったとすれば、時の経過とともにこのソフトウェアは繰り返し再構築を必要とするようになる。従って現在求められている方法は1回だけ再構築をすることで繰り返す再構築の必要のないようなソフトウェア管理状態にすることである。

情報システムの開発保守をソフトウェアを対象としないで、情報システムモデルを対象とする。

このようなことを可能にするのは、リポジトリを中核としたフォワードエンジニアリングCASEを活用することで実現できる。ソフトウェアの製造をプログラマーが行うのではなく、ソフトウェアを創り出す元となる要素を開発、保守し、ソフトウェアは常に最新の条件のものを自動生成することができればよい。従ってリエンジニアリングは従来の方法、技術による手作りのソフトウェアを手を加える必要のないソフトウェア自動生成の機能を持つCASE環境の管理下へなれば自動的に移植することである。

2. ソフトウェア開発・保守のあり方

2. 1 従来のソフトウェアの保守性の悪さ

ソフトウェアの本質的にハードウェアへの指示命令群であった。その後、システムソフトウェアや言語の進歩により、オペレーションシステムやデータベース、マネジメントシステムなどへの定義をも含めるようになった。いずれにしても従来の作成方法は機能的要素とか部品とかを結合、組み立て上げるものではなく、具体的に決められた仕様をする特定のハードウェアやシステムソフトウェアで稼働する言語仕様や技法で表現したものである。

従って仕様や環境条件の変化に対応して、必要な要素のみを変更、追加する保守作業には極めて不都合なものである。ある程度以上の変化には實際上保守で対応不可能となる。

2. 2 ソフトウェアの要素

ソフトウェアを組み立てたり、解析したりすることを想定するなら、その構成をいくつかの要素に分解して見る必要がある。この要素はソフトウェアの表現上での分類ではなく、本質的な機能という観点からでなければ用をなさない。

ソフトウェアが現実実行してその役目をはたすには次の3つの要素を考えるのを理解がしやすくなる。

- ① 業務仕様 (What)
- ② 稼働環境 (Where)
- ③ 実現技術 (How)

この3つの要素の適正に結合されたものがソフトウェアとなる。

(1) 業務仕様

業務仕様は情報システムのWHATであり、情報システム運用の目的である。ユーザの要求する具体的、詳細な内容である。たとえば「単価に数量を掛け売上額をする」といった内容で、これがソフトウェアに含まれていなければ全く意味をなさない。

システム開発の工程において要求分析から始まってプログラム仕様に至るまでこの業務仕様を具体化する作業である。

従って上流CASEが対象としているのはこの業務仕様である。

(2) 稼働環境

情報システムはいずれからのコンピュータ上で稼働する。

ハードウェアとシステムソフトウェアの上にアプリケーションシステムが稼働する。従ってソフトウェアの表現方法はそのハードウェアやシステムソフトウェアなどの稼働環境により決定される。

実務上運用されているシステムは必ずいずれか具体的なハードウェア、システムソフトウェアが前提であり、アプリケーションソフトウェアのWHEREとしての要素である。

プログラム言語はメインフレームコンピュータにおいてCOBOLを中心に標準化、共通化の方向にあったが、パーソナルコンピュータやエンジニアリングワークステーションの普及とともにBASIC、Cなど異なる言語が広まっている。

(3) 実現技術

「業務仕様」と「稼働環境」上で動かすためには、設計書やプログラムの技術が必要である。これが実現技術でソフトウェアのHOWの要素である。

ソフトウェアエンジニアリングとして構造化プログラミング以来、この技術が研究されている。

2. 3 自動生成と個別要素別保守

図2に示すように、業務仕様の論理モデル、技術モデル、環境モデルから物理モデルを設計する。この物理モデルから100%完成した全てのソフトウェアを生成する。

保守はソフトウェアの要素である、論理モデル、技術モデル、環境モデルの保守し、その結果を物理モデルに反映させ、ソフトウェアを生成させることで実行する。ソフトウェア自体の保守という概念はなくなり、常に全体的に作り直しになる。当然再構築の必要性はなくなる。

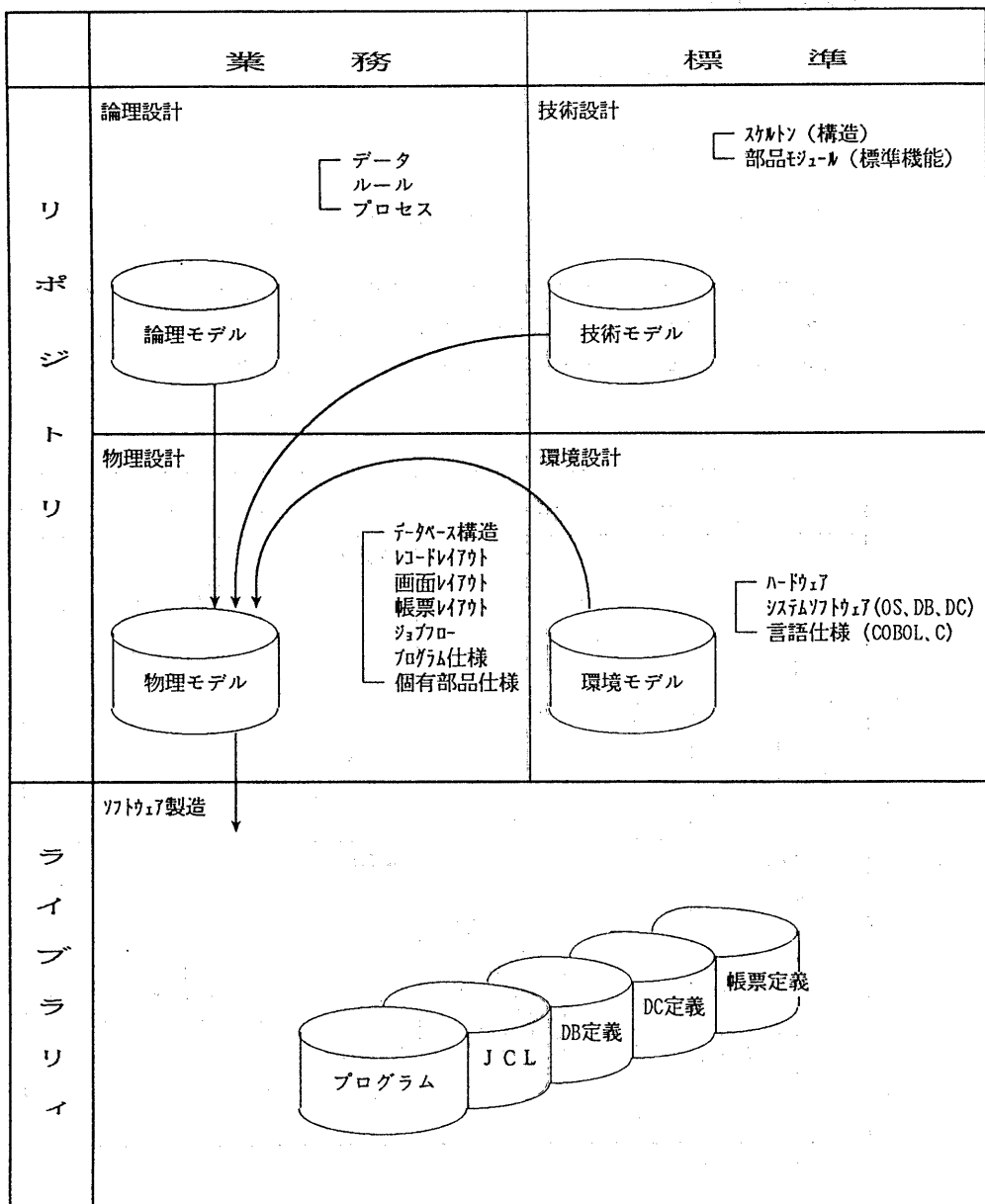


図 2 : ソフトウェア自動生成

3. 再構築のためのCASE

3. 1 リバースエンジニアリングCASE

3. 1. 1 2段階のリバース

図3に示すようにアプリケーションソフトウェアからのリバースは2段階ある。

まず、第一レベルはソフトウェアを言語解析することによりこのソフトウェアの物理モデルに抽象化する段階である。この段階でソフトウェアの管理対象と構造が重要である既存ソフトウェアの保守に有効である。

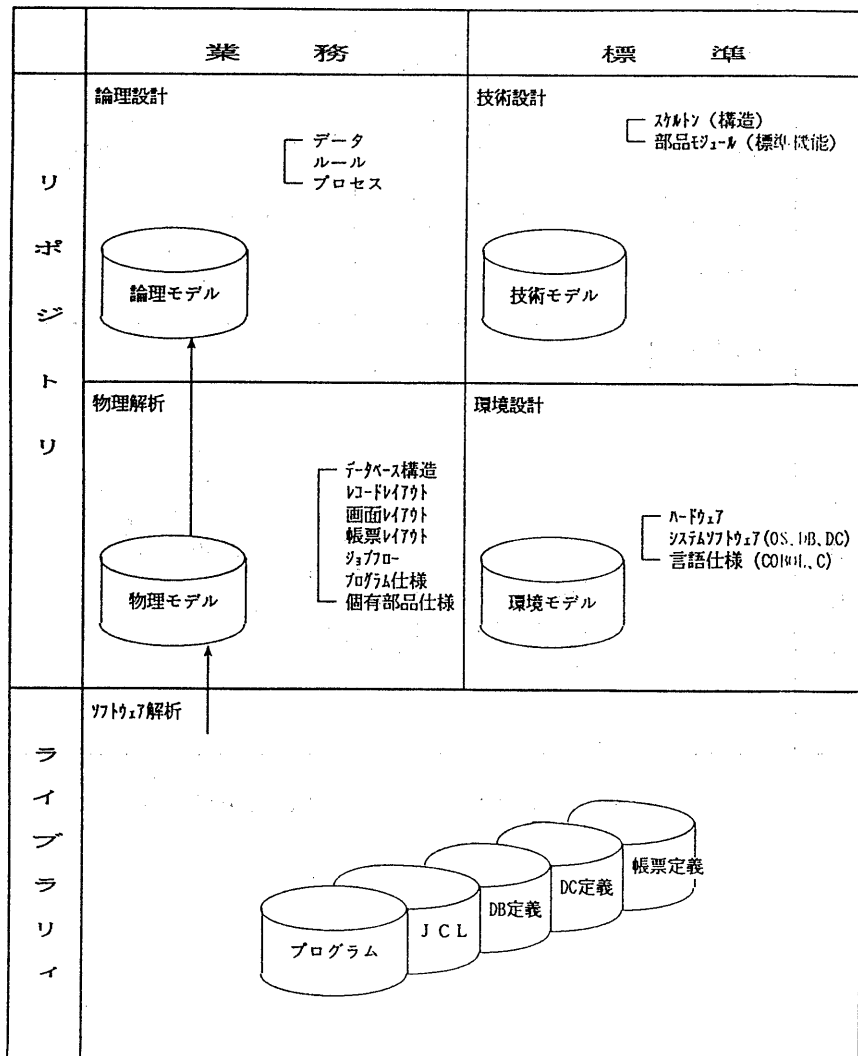


図3 : リバースエンジニアリング

3. 1. 2 業務仕様の抽出

エンジニアリングのためのリバースは第1段階の物理モデルでは不十分である。再構築にはソフトウェアの要素の1つかつ情報システムの目的である業務仕様の抽出が不可欠の条件である。

具体的な成果物は業務仕様の論理モデルで表現される。

第1番目にはデータ単項目に関するもので異名同義項目のグループ化とその項目に属する長さ、属性、値の領域などである。

第2番目にはデータ項目間に関する仕様である。自然語に近い分りやすい表現業務仕様記述言語が有効となる。

第3番目にはデータの構造配列の論理モデル表現である。

ソフトウェアの他の2つの要素である、稼働環境、実現技術はリバースで抽出する必要はない。これらはあくまでもソフトウェアの作成のための不可欠な手段であるが、古いものを継承する必要はない。

3. 2 フォワードエンジニアリングCASE

再構築との関係においてフォワードエンジニアリングCASEの機能を考える。

フォワードエンジニアリングCASEとしては

- ・ソフトウェアの要素別管理
- ・ソフトウェアの完全自動生成

は不可欠である。この機能が存在しない限りソフトウェアの保守が発生してしまう。特に業務仕様は現行ソフトウェアから抽出された情報を受け取るものであり、そのモデルは極めて重要な位置づけにする。

3. 3 品質保証

新規に開発したソフトウェアと異なり再構築の場合はオリジナルのソフトウェアとデータが存在し、いずれも運用され、実務上活用されているものである。この状況を活用して再構築したソフトウェアの機能と品質を評価できる。

アプリケーションソフトウェア、データ、コンピュータシステムにより情報システムは運用される。従ってアプリケーションソフトウェアが再構築前と後が等しいことを証明することが考えられる。両システムで同じシステムを運用し、結果としてのデータの物理構造や形でなく論理的に比較することで等しいことが分ればよい。

この方法CASEはない限り、再構築は品質保証がシステムとしてなされない。

4. 再構築の特徴

4. 1 既成事実が前提条件

再構築は既存のソフトウェアを解析するのである。従ってそのソフトウェアは現在本稼働しており、実用上問題のない水準にあるとみてよい。従ってリバースエンジニアリングの立場からソフトウェアの修正や変更を要求することは許されない。規格や標準を定めてその範囲内の解析をするのでは實際上役に立たない。既成事実が前提条件になる。

4. 2 一過性の作業

再構築は本質的に切り替え、移し替えの作業である。従ってルーチンワークではなく一回限り一過性の作業となる。また再構築に必要なソフトウェアは通常ライブラリ上に保管されていて、何時でも処理可能な状態である。

この2つの条件から図に示すように①から⑥の処理を繰り返えし実行する方法で再構築の品質を合格するまでさせる。

- ① 自動解析：リバースエンジニアリングCASEによるソフトウェアの解析を抽出
- ② 自動生成：フォワードエンジニアリングCASEによるソフトウェアの自動生成
- ③ 現行システム処理：現行システムのソフトウェアとデータによるシステムの実行とアウトプットの作成
- ④ データ変換：現行のデータから再構築用データへの構造、形の変換
- ⑤ 再構築システム処理：再構築システムのソフトウェアとデータによるシステム実行とアウトプットの作成
- ⑥ 品質保証：現行システムの更新後データと再構築システム更新後データの論理比較による評価

5. おわりに

私自身開発に係わったCASEとして、フォワードエンジニアリングは、1983年に開発したCANO-AID、リバースエンジニアリングは、1992年に開発したRESCUEがある。本稿に述べたことがある程度まで実現化されている。今後はフォワードとしてはモデルのあり方、リバースとしてはソフトウェアの解析方法が開発の対象としたい。

参考文献

- (1) 本村 昭二 システム再構築支援システムの試み
日経コンピュータ 1990. 1. 1
- (2) 本村 昭二 ソフトウェア保守用CASE RESCUEの開発
日経コンピュータ 1990. 11. 18

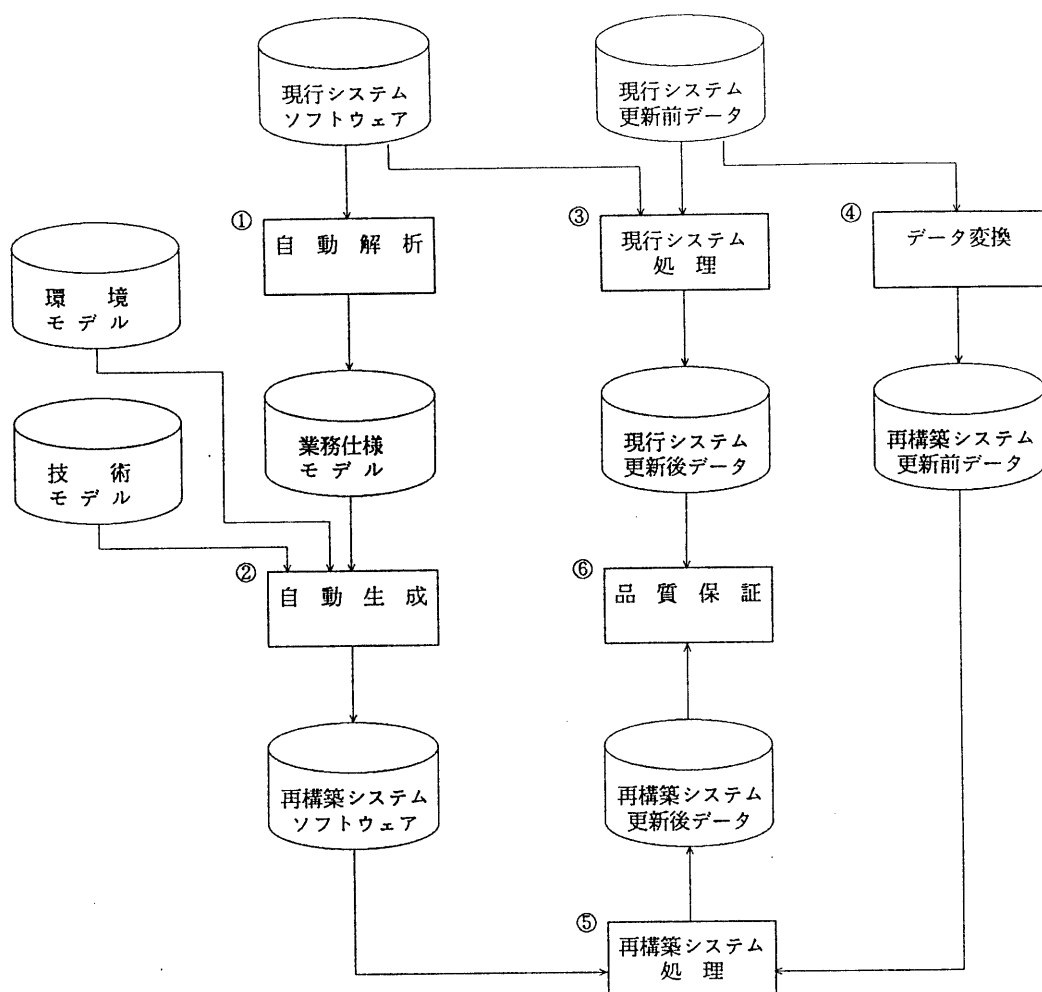


図4 : リエンジニアリングの手順