リエンジニアリング手法：RELICS *

5N－4

忠海　均　　高田 信一
NTT ソフトウェア研究所

1　はじめに

近年のダウンサイジングの普及に伴い、既存の大型汎用機を用いた集中型システムから安価なクライアント／サーバシステムへの更改ニーズが高まっている。しかし、現状では更改後のアーキテクチャが異なるために、既存の資産が使えないなどの問題があり、システム更改が高価なものになっている。そこで、リエンジニアリング技術に着目し、このようなシステム更改ニーズに応えるリエンジニアリング手法である RELICS (Re-Engineering method for Legacy Information proCessing System) を開発した。

2　RELICS の特徴

リエンジニアリングには様々な解釈があるが、これを以下のように定義する [1]。

リエンジニアリング：リバースエンジニアリング、フォワードエンジニアリングの組合せによる開発

リバースエンジニアリング：抽象度の低い情報から高い情報を復元すること

フォワードエンジニアリング：抽象度の高い情報から低い情報へ変換すること

従来のリエンジニアリング技術では、フォワードプロセス（新規システムの開発）の前にリバースプロセス（設計情報を復元する過程）が存在する。そのため、大規模システムの開発では、多額の初期投資が必要であった。そこで、これを削減するための支援ツールが数多く開発されている。しかし、アーキテクチャの異なるシステムに更改する場合、再利用可能なソースプログラムが少ないため、支援ツールでリバース可能な詳細設計レベルの設計書の大部分が使えず、初期投資額に比べて効果が少ないという問題がある。そこで、この問題を解決するために、工程毎にリバース範囲を規定し、逐次リバース作業を実施することにより、必要な設計情報のみリバースして最適なリバース投資を実現できる RELICS を開発した。

RELICS は、大規模アプリケーションプログラムのシステム更改を対象としており、作業マニュアルとツール群から構成される。

工程毎に最適な設計情報をリバースしようとした場合、どのように再利用する設計情報を定めるか、上流工程の抽象的な設計情報を如何に効率的にリバースするかが大きな問題となる。これを解決するために、RELICS では、工程毎にリバース範囲を限定するための再利用基準、抽象的な設計情報をリバースするための支援ツールを用意した。

3　工程単位の設計情報のリバース法

ある工程に着目した時のその工程でのリバース作業手順を図 1に示す。

RELICS では、中間情報項目表として、新規開発する設計書の目次項目に対し、対応する再利用情報（既存設計書の目次項目相当）およびそのリバース法を記述する。ここで、再利用情報は前工程の生産物と既存設計書から決定できるが、既存設計書の品質が悪くて再利用範囲が決定できないことが多い。そこで、RELICS では再利用判断基準を設け、既存設計書の目次項目相当の単位でこれに一致するものを再利用候補とすることにした。再利用判断基準の一部を表 1に示す。

表 1: 再利用判断基準の例

再利用項目	判断基準					結果
	言語	API	外部仕様	データ	性能 . .	
処理方式A	COBOL→C	変更なし	機能追加	DBMSは同じ	変更なし	再利用可能
モジュール構成 B	COBOL→C	変更なし	変更なし	ファイル変更	変更なし	再利用可能
モジュールX	COBOL→C	変更なし	機能変更	入出力データ変更	変更なし	再利用不可

次に、各再利用候補について、RELICS で用意しているリバース手順（情報項目対応にリバースツールを組み合わせてリバース作業を実施する手順）に従い、ソースプログラムや JCL などから設計情報をリバースする。なお、工程毎にリバースを実施するにあたり、モジュール構成やテーブル一覧などの各工程で使用する共通的なリバース情報が存在する。これらの情報は、RELICS ツールで事前に作成できるようにしている。

リバースした情報は中間情報ファイルに格納する。新規側開発者は、中間情報を再利用データと捉え、これを利用して新規側の設計書を作成する。

4　抽象情報のリバース法

上流工程の設計書をリバースするためには、ソースプログラムから抽象的な設計情報（サービス仕様や方式に係わる情報など）を復元することが必要になる。RELICS で実現している手法を図 2に示す。抽象的な情報の復元のためには、複数の視点からそれを捉え、抽象的な情報を決定する人間の知的作業が必要になる。例えば、「モジュールの機能」の決定は、それを使う上位モジュールの処理、入出力、その実現アルゴリズムから可能となる。これを支援するために、RELICS では、ドメインを限定することにより、その処理方式やパッケージインタフェースに特化したツールを用意して、複数視点での情報を生成できるようにした。

5　手戻りの扱い

新規設計情報が確定する前に中間情報を決定すると、手戻りの可能性が高い。RELICS では、これを以下のように扱っている。

- リバースした設計情報が再利用不可能と判明した場合は、それを廃棄し、新規に設計する。
- リバースした設計情報以外の情報が必要になった場合は、その都度、必要な情報をリバースする。
- 変更のない再利用情報のうち、ツールで自動出力したもの以外はレビューした上で使用する。

---

*Re-Engineering Method: RELICS,
Hitoshi TADAUMI and Shin-ichi TAKATA,
NTT Software Labs.

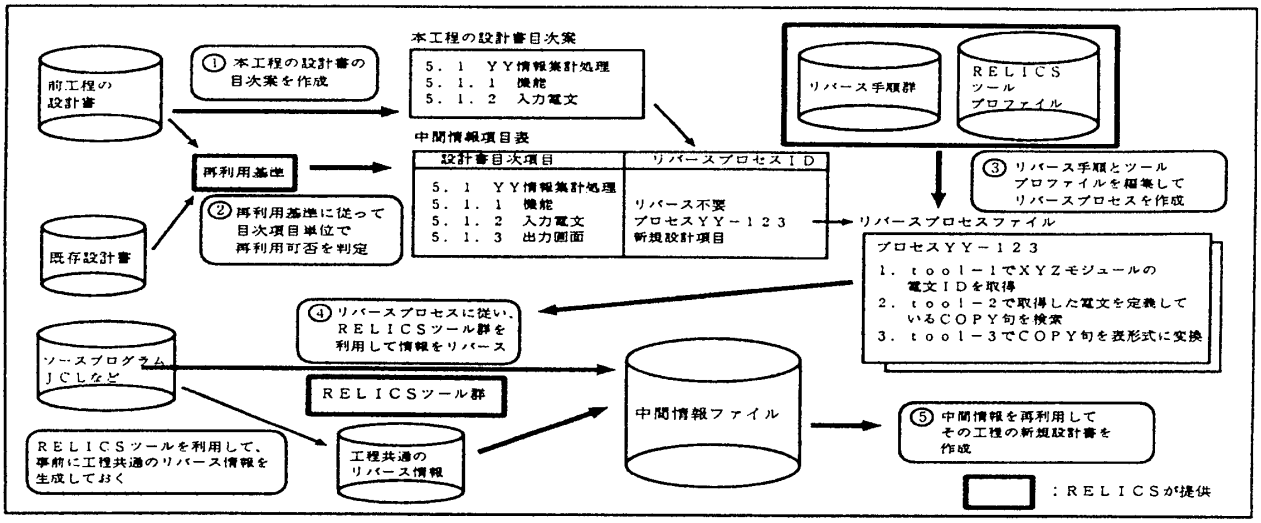

図 1: RELICS におけるリバース作業手順

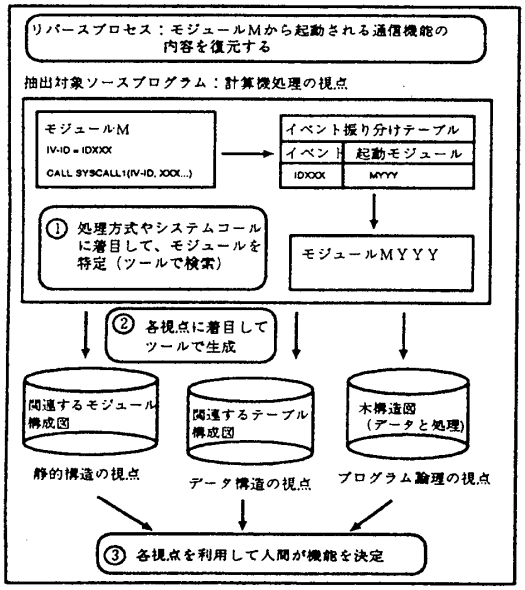

図 2: 抽象的な設計情報を復元する手法

これらの扱いでは、各設計情報に対するリバース工数が増えると、工程の遅延につながる。そこで、RELICS ではプロセス定義を可能な限り自動的に実行できるように、リバース手順群の大部分をコマンドプロシジャとして提供している。

6 実験結果

RELICS の有効性を評価するために、実験的に COBOL で記述された汎用計算機上で動作する通信系アプリケーションプログラムのリバース作業を実施した。この実験では、システム更改の仮想的な仕様を想定し、必要な部分のみの設計書をリバースした。リバース実験結果を設計情報毎に表 2に示す。再利用率や復元率は想定した仕様に依存するが、システム更改の場合、サービスに係わる仕様の変更は少ないと想定している。

今回の実験では、再利用判断基準に一致した設計情報は、新規設計書の約 4 割程度であった。これは、プログラム論理やモジュール仕様などの再利用率の低い詳細設計レベルのドキュメント量が多いことによる。これより、工程毎にリバース範囲を絞り込むことにより、リバース投資の削減が可能になることが分かる。

また、表 2から一定の復元効率（従来より良い効率）を保ちつつ、設計情報のかなりの部分がリバースできることが分かる。しかし、RELICS を利用しても、原理的にソースプログラムから復元できない外部イベントのシーケンスにより決まる機能や、ソースプログラムにコメントがなくて設計書の日本語名称と対応が付かない情報などの復元はできなかった。これは機能仕様や方式に係わる部分に多いが、ソースプログラムから設計情報を復元するアプローチの限界と考える。

表 2: RELICS を用いたリバース実験の結果

再利用情報	再利用率[1]	復元率[2]	復元効率[3]	備考
機能仕様／方式等	35	50	65	情報の欠落／リバース不可能な情報が多い
ファイル／テーブル仕様等	75	100	95	従来ツールを使用
ユーザインタフェース仕様	50	90	80	仕様変更部分が多い
モジュール構成	80	100	100	従来ツールを使用
モジュール仕様	50	60	70	−
プログラム論理	40	100	100	従来ツールを使用

1) 新規設計書に対して、再利用できたドキュメントの率（%）
2) 再利用を決定した部分のうち、実際に復元できた率（%）
3) 既存社内ツールを利用した従来法に比較したリバース作業時間の割合（%）

7 おわりに

工程毎に設計情報をリバースするにあたっては、作業をどこまで自動化できるかが重要となる。RELICS では、ドメインを限定することにより、リバースする情報をある程度定め、それに着目したリバースプロセスを用意にすることにより、作業の自動化範囲を拡大した。現在、RELICS は実際の大規模ソフトウェアの開発に適用中である。今後、その適用結果で RELICS を再評価／改善するとともに、支援ツールの整備、適用域の拡大をはかっていきたい。

参考文献

[1] R.S.Arnold: Software Reengineering, IEEE Computer Society Press, pp.3-22, 1992