ソフトウェア工学 102-2
（1995．1．26）

システム更改に適応したリエンジニアリング手法（RELICS）の提案

忠海 均　高田 信一　山本 修一郎
NTT ソフトウェア研究所

〒180 東京都武蔵野市緑町 3-9-11

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、システム更改コストを削減するために、リエンジニアリング手法であるRELICS を開発した。
本稿では、RELICS の概要を解説し、既存設計情報のリバース実験結果と実開発への適用結果に基づき、本手法の有効性を考察する。

Proposal of System Migration Oriented Re-engineering Method(RELICS)

Hitoshi Tadaumi, Shin-ichi Takata, Shuichiro Yamamoto
NTT Software Labs.

3-9-11 Midori-cho Musashino-city Tokyo 180 Japan

Accompanied with the recent propagation of Downsizing, necessity of migration from legacy mainframe systems to cheaper Client/Server systems has been increased. But since a migrated system architecture is usually differant from the existing one and it is hard to reuse assets of the existing system, nowadays the migration cost is yet expensive. Then, in order to reduce this cost, the Re-engineering method, RELICS, was developed.
This paper describes an abstract of RELICS, and discusses about its effectiveness, based on the experiment of reversing existing design information and the application of this method to a real software development.

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 [1][2]。

本稿では、RELICS の概要を簡単に解説し、本手法における再利用の考え方、抽象情報の復元法を中心に、実験結果と適用結果に基づき、本手法の有効性を考察する。

2 RELICS の概要

システム更改のコストを削減するために、既存システムの設計情報を調査し (**リバースエンジニアリング**)、それを新規システムの開発に活用する (**フォワードエンジニアリング**) 技術であるリエンジニアリング技術が開発されている。

典型的なリエンジニアリングのプロセス [3] を図 1に示す。この例では、まずリバースエンジニアリングを行い、既存システムの設計書やソースコードなどを入力とし、実装方式などの抽象度の低い情報から機能仕様や業務仕様などの抽象度の高い情報を復元する。それが完了してから、フォワードエンジニアリングにより新規システムを開発する。

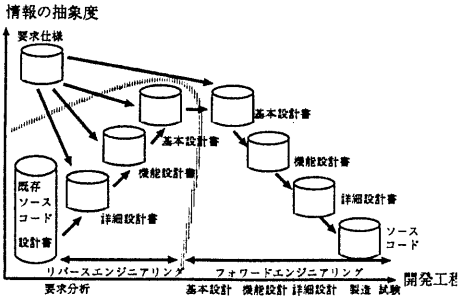

図 1: 従来のリエンジニアリングプロセス

ところが、大型汎用機のダウンサイジングのようにアーキテクチャの大幅な変更をともなう場合、既存システムの設計情報がそのまま新規システムで利用できないことがある。このような場合には、再利用できる既存システムの設計情報を選択する必要がある。しかし、このプロセスでは、フォワードプロセスが開始する前に、仕様が未確定の状態でリバースプロセスを完了させるため、再利用性の判断がほとんどできないという問題がある。

この問題を解決するために、図 2のようなプロセスを開発した。このプロセスでは、新規開発の各工程の前で、その工程分の設計情報をリバースしており、新規開発における前工程の設計内容を参考に、再利用する設計情報を選択することが可能である。これにより、再利用可能な情報のみをリバース対象とし、最適なリバース投資をすることが可能となる。

著者らはこの視点で RELICS を検討してきた。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。以下では、主な特徴を紹介する。

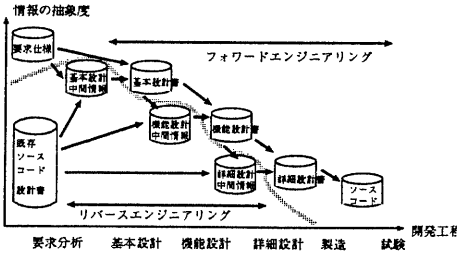

図 2: RELICS によるリエンジニアリングプロセス

2.1 アプローチ

下流工程の設計情報を利用せずに、ソースコードだけから上流工程の設計情報をリバースすることは非常に困難である。この問題に対処するため、RELICS では既存システムのドキュメントを有効に利用することにした。**RELICS では、既存ドキュメントからのリバースエンジニアリングと、ツールを使用したソースコードからのリバースを併用して、設計情報を抽出する。**

2.2 作業体制

大規模システムにおける膨大な既存ドキュメントから必要な情報を復元するためには、その情報がどこにあるかを知っており、記述内容の信頼性を評価して、正しい情報を選択できる必要がある。従って、既存システムの詳細な知識を持った技術者が必要となる。ところが、そのような技術者は限られているため、その少数の技術者の工数を有効に利用し、その作業結果を詳細な知識を持たない多数の技術者に円滑に引き継ぐ必要がある。**RELICS では、既存システムの知識を持った少数の技術者の工数を有効に利用するための体制を規定している。**この体制での役割分担を表 1に示す。

2.3 作業工程

以下ではこのような体制のもとで効率的に作業を進めるためのプロセスを説明する。RELICS では、各設計工程毎にリバースおよびフォワードプロセスを繰り返す。各工程のプロセスはほぼ等しいため、以下では一つの工程について説明する (図 3)。

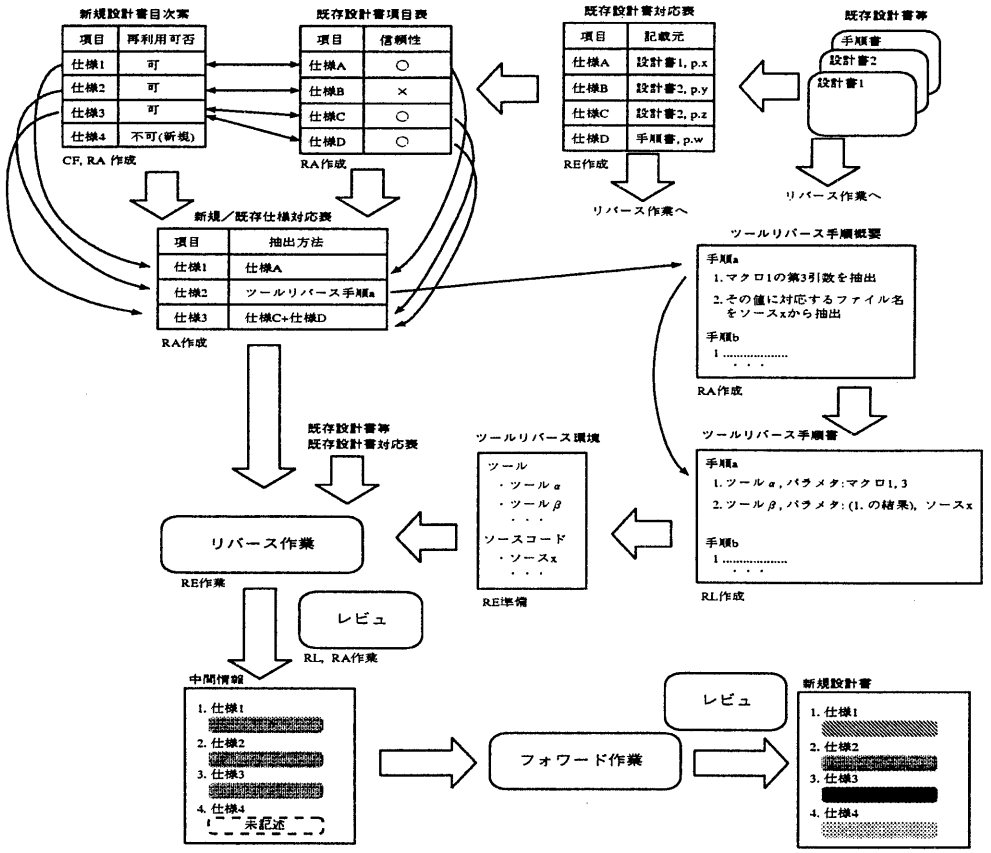

図 3: RELICS におけるリバース作業手順

表 1: 役割分担

役割名	役割分担	スキル
リバースリーダ (RL)	リバース作業全体を管理し、新規開発との整合性をとる。	既存システムと新規システムの仕様を把握できる。
リバース支援者 (RA)	既存システムの設計情報の評価や抽出方法の決定を行う。	既存システムの開発担当者などで、既存システムの詳細を熟知している。
リバース技術者 (RE)	RL の指示に従ってリバース作業を実施する。	WS、RELICS のツール群などを操作できる。

2.3.1 新規開発用設計書の目次案の検討

新規開発で作成される予定の設計書の目次案を作成する。この目次案には、新規開発で新たに追加する機能項目も含まれている。これに基づき、新規側の開発担当者と RA が、再利用対象の設計情報を抽出する。(図 3の新規設計書目次案参照)

2.3.2 リバース情報の調査

リバース対象の各項目について、RA がその情報の抽出元となりそうな既存ドキュメントの記述箇所を選択する。そして、選択した記述箇所の各々について記述内容の信頼性評価を行う。(図 3の既存設計書項目表参照)

2.3.3 リバース方法の決定

RA が新規設計書目次案の各項目に対して、既存設計項目表の仕様項目から、再利用可能な仕様項目を抽出し、それを新規設計書目次案の各項目に対応付ける。この対応がとれた項目については、リバース方法が決定したことになる。対応がとれなかった項目、つまりその設計情報をリバースするにあたっ

て信頼できる情報がなかった場合には、ツールによるリバースを検討する（後述）。なお、ツールによるリバースも不能な情報は、リバースを諦めてフォワードでの作成対象となる。（図3の新規／既存仕様対応表参照）

2.3.4 リバース作業の実施

RE が手順書で決められた手順にしたがって、必要な情報をリバースする。復元された情報を中間情報と呼ぶ（図3の中間情報参照）。

2.3.5 フォワード作業

中間情報は、新規開発設計書からフォワード作業で作成される設計情報を抜いた状態でフォワード作業に提供される。フォワード作業では、中間情報の空白項目への追加と既に記述されている項目の部分的な修正により、新規システムの設計書を完成させる。

3 RELICS における設計情報の再利用の考え方

上記のように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が問題となる。RELICS では、これを容易にするために、再利用する設計情報の視点と再利用可否の判断のためのガイドを用意している。

再利用する設計情報の観点は以下のとおりである。

- 外部仕様

  業務および機能仕様 業務フローやシステムの機能に関する仕様

  システム構成仕様 システムのネットワーク／ハードウェア／ソフトウェア構成

  外部インタフェース仕様 他システムや端末との通信インタフェース、DB、ファイルなどの外部データ、画面、コマンド、帳票などの HMI 仕様、API などリバース対象ソフトウェアとその下位パッケージや基本ソフトとのプログラムインタフェース仕様

  品質特性 性能、信頼性などの品質特性の規定

- 内部仕様

  方式仕様 システムの機能の実現方式の仕様

  内部データ仕様 システム内部のファイル／共通テーブルの仕様

  モジュール構成仕様 モジュール構成とモジュール間インタフェースの仕様

  モジュール仕様 モジュール単位の機能の仕様

  プログラム論理仕様 モジュール内のプログラム論理の仕様

これらの情報は、ドキュメント体系の違いがあってもある程度一般的に記述される情報である [5]。これらの情報のうち、信頼できる情報だけでも再利用できれば、ドキュメント作成工数がかなり削減できると期待できる。更に、外部仕様に係わる情報は、マニュアル類にも記述されていることが多いが、マニュアル類に記述された情報はかなり信頼性が高いため、貴重な再利用情報となる。

また、これらの視点を定めることによって、各々の設計情報毎に、再利用可否の判断にあたってのノウハウ相当のガイドの提供を可能としている。以下に、各視点毎のガイドでの基本的な考え方を示す。

外部データ仕様 そのシステムが処理しなければならない入出力データに変更があった場合でも、処理方式からモジュール設計まで再利用可能なことが多い。しかし、処理データ量が大幅に増加する時や、性能条件が厳しくなるときは、これらを優先的に考慮する必要がある。

業務および機能仕様 業務フローやシステムの機能仕様に変更がある場合、処理方式から全面的な見直しが必要となることが多い。しかし、再利用不可能な仕様が一部に集約されていることが多いので、再利用単位を細かく分けることにより再利用率を高くすることが可能である。

プログラムインタフェース仕様 OS、下位パッケージなどの変更はファイル、メモリ、回線などの資源へのアクセス方式に大幅な変更をもたらす。アプリケーションソフトウェアでは、OS や下位パッケージへのアクセス部分は下位モジュールとして実現されることが多いが、これらの下位モジュールが再利用不可能となることが多い。しかし、共通的なアクセスインタフェースを用意している場合、それより上位の設計情報を再利用できる場合が多い。

性能および基礎業務量 これらの大幅な変更がある場合、システム全体の処理方式に影響を及ぼし、特に、性能条件の厳しいシステムの場合、内部仕様の再利用可能部分が少なくなる。

記述言語 記述言語の変更がある場合は、ソースの再利用は困難である。しかし、言語のみの変更の場合、モジュール構成については有効に利用することができる。

現実のシステム更改では、変更箇所はこれらの複雑な組合せであり、ガイドに従った単純な判断では再利用可能範囲を決定できない。再利用可能範囲の抽出には、現状ではスキルの高い技術者の判断力が必要である。

4 抽象情報のリバース法

従来のリバースツールで汎用的に取得可能な情報は下流工程に近い設計情報がほとんどであった。RELICS においては、それらを利用する一方で、上流工程の設計情報である抽象情報を RA の知識と

RE の稼働を活用してソースコードから復元する方法をとった。そのアプローチは以下のようなものである。

- 既存システムの基本ソフトの仕様やアプリケーションの規約などの知識を活用して、特定の情報を抽出するための手順を RA が検討する。この手順は RE が手作業でできるものであればよいが、システム対応の特定のシステムコールや名標をキーに必要な抽象情報を取得できることも多く、パーサやパターンマッチングの検索のための汎用的なツールを手順に合わせて組み合わせて利用する。
- RL はその手順に従い、作業指示のための手順書を作成する。
- RE はその手順のなかでパターン化された単純作業をツールにより自動化して、リバース作業を実施する。

しかし、抽象情報の復元のためには、複数の視点からそれを捉え、人間の知的作業によりその情報を特定する場合もある。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。このような場合は、情報の抽出までは RE が実施し、最終的な情報の復元は RA が分担することになる。

このアプローチでは、リバースする情報項目対応に手順が存在するが、ドメインを特定すれば、処理方式やパッケージインタフェースに特化した手順を事前にツールとして提供することができる。現在、RELICS で実現しているツールの概念を図 4に示す。

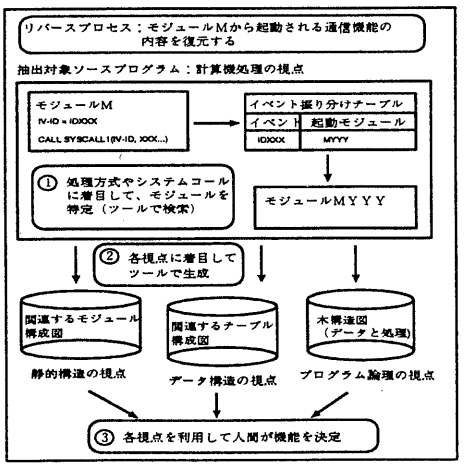

図 4: 抽象的な設計情報を復元する手法

5 考察

5.1 リバース実験

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。

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	ー
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（%）

この実験では、再利用できた設計情報は、新規設計書の約 4 割程度であった。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。なお、復元できなかった情報は、外部イベントのシーケンスにより決まる機能などであった。

5.2 実開発への適用

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。適用状況を以下に示す。

開発の概要 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約半分の機能をダウンサイジングして、UNIX ベースの分散処理システムに移行。

- 他システムとの通信インタフェースに一部変更あり。
- 移行された機能、関連する業務の変更、業務量の増加あり。
- API は既存システムと類似のものが既に存在。

進捗状況 基本設計／機能設計／詳細設計からなる設計工程のうち、基本設計まで完了。

適用結果 基本設計書の一部である他システムのと通信インタフェース仕様書（記述項目数：約 1

１０）のみをリバースして再利用した。対象として選択した理由は、ドキュメントが紙のみであり修正に耐えないこと、再利用可能な情報が多かったことである。これにあたり、リバース元は、既存通信インタフェース仕様書（約７０項目流用）と既存機能仕様書（約３０項目流用）であった。なお、２項目（電文一覧、電文マトリックス）は手作業で作成した。また、これらの既存ドキュメントの記述内容は全て信頼できた。

ここで、再利用対象外となった情報は、システム構成、基本方式、業務／機能仕様、運用仕様などであり、これらは既存情報からの流用率が２０％以下となったため、全て作り直しとなった。しかし、再利用性の評価を実施することにより、客観的に妥当なリバース投資をして、既存資産を再利用できたことは評価できる。

また、今回は、維持管理のためのドキュメント整備が既に終っていたため、既存ドキュメントの不足情報のソースプログラムからの復元作業は実施することができなかった。なお、本開発への適用は、詳細設計工程まで実施する予定である。

5.3 RELICS の適用性

以上の２つの事例の経験に基づき、RELICS の適応性について整理する。

規模 対象とするシステムが大規模（100KL 以上）であること。数 10KL 程度ででは、全ソースを人手で調査してリバースするほうが効率がよい。

システム改造内容 単なる移植や部分改造ではなく、かなりの設計変更を伴う開発であること。モジュール構成や木構造図などを復元する程度のリバースツールだけで済むような場合には、RELICS を適用する効果はほとんどなくなる。

RA の存在 RA の稼働は全リバース工数の１０％程度であった。しかし、既存システムを熟知している RA の存在は必須である。一方、下流工程の設計情報しか利用しないリエンジニアリング手法 [4] には、上流工程の仕様変更をともなうシステム更改への適用は困難であるが、RA が存在しない場合でも適用可能であるメリットがある。

RE の作業の単純性 リバース作業の実施にあたり、リバースする項目とその方法はすべて手順化されるため、RE の作業にシステムに関する詳細知識を要求しない。これにより、RE は特定の技術者である必要はなく、工数の確保は容易である。しかし、リバース手順に人間の判断を必要とする部分が存在し、これが多いと、判断方法などを手順書に記述する RA の稼働を圧迫する。

既存ドキュメントの状態 信頼できるドキュメントがほとんど存在しない状態では、リバース可能な情報が限られ、復元効率が劣化する。

RELICS は適用条件が整えば有効であるものの、実際の適用では、なかなか理想的な条件が揃わないため、リバース実験の結果からだけでその有効性を判断できない。今後も、適用実績を積む必要がある。

6 おわりに

システムダウンサイジングの要求とともに、それをサポートするリエンジニアリング手法が要求されている。

RELICS ではフォワードエンジニアリングとリバースエンジニアリングをほぼ並行して行い、リバース情報の再利用の最適化を図ると共に、従来有効利用されていなかった既存ドキュメントの活用、抽象情報をリバースする手順やツールの提供などにより、上流工程までの情報の抽出率を高めることによって、この要求に対処している。この手法は、知識が必要な作業と不要な作業を切り離し、双方の作業の間を円滑に繋ぎ、知識を有効に利用することで成立している。

しかし、実開発への適用を通じて、RELICS の効果が適用条件にかなり依存することが明らかになってきた。今後とも、適用実績を重ねつつ、技術の改善、適用域の明確化を図っていく予定である。

7 謝辞

日頃御指導頂く、細谷 僚一ソフトウェア研究所長及び長野 宏宣ソフトウェア技術研究部長に深謝致します。

参考文献

[1] 忠海 均, 高田 信一：リエンジニアリング手法: *RELICS* の概要, 情報処理学会 第４９回（平成６年度後期）全国大会 講演論文集（５）, 5-307

[2] 高田 信一, 忠海 均, 山本 修一郎：システム更改に適応したリエンジニアリング手法の提案, 電子情報通信学会技術研究報告, KBSE 94-33

[3] E. J. Byrne. : *Software Reverse Engineering: A Case Study*, Software - Practice and Experience, Vol.21, pp.1349-1364, Dec. 1991

[4] H. M. Sneed & E. Nyary. : *Downsizing Large Application Programs*, in Proc. of 9th CSM.

[5] 花田 収悦：ソフトウェアの仕様化と設計, 日科技連, 1986