# ソフトウエア変更作業の分析と支援機能

津田道夫* 永岡郁代* 青木一紀** 和栗正一***

* （株）日立製作所 ビジネスシステム開発センタ
** 日立電子サービス（株）情報システム本部
** 日立ソフトウエアエンジニアリング（株）オープンシステムプログラム部

ソフトウエア変更作業における、作業工程を分析して、必要な情報や作業内容、その結果として得られた情報を明確にした。また、実際に保守担当者が、どのような方法で仕様を理解して、プログラムの修正をしているかを、調査した。この結果、プログラム理解の方法が、保守経験値により異なることが判明した。この分析に基づき、ソフトウエア影響波及検索ツールのプロトタイプ（ツール名：ＦＩＮＤＭＡＴＥ）を開発した。ＦＩＮＤＭＡＴＥは、ユーザからの保守依頼に対応して、既存ソフトウエア・リソースの相互関連を検索する。これにより、影響波及対象のリソースを抽出して、作業見積、作業計画や修正作業に利用する。

# Analysis and Support Tool of Software Modification Operations

Michio Tsuda*, Ikuyo Nagaoka*, Kazunori Aoki**, Shouichi Waguri***

* Institute of Advanced Business Systems, Hitachi, Ltd.
** Information Systems Division, Hitachi Electronics Services Co., Ltd.
*** Open System Design Department, Hitachi Software Engineering Co., Ltd.

We analyzed processes of software modification and clarified essential information, contents of the modification operations, and information that is obtained from the operations. We also studied what kind of procedures the software maintainers use to understand programs and to modify them. As a result, we found that the procedures of understanding software depend on the years of experience of the software maintainers. According to the result, we have developed a prototype of the tool "FINDMATE", that finds the effects of software modification. FINDMATE can be used to search interrelationships across the existing software resources to meet users' maintenance requirements. FINDMATE identifies affected resources,which can be used for estimation of the amount of modification operations, planning of the operation, and the actual modification.

## はじめに

大規模な情報システムを開発、維持している組織では、標準の保守作業手順があり、それに従って保守がおこなわれている。保守作業は、仕様理解、ソフトウエアの修正と確認に分けられる。しかしながら、保守担当者が実際に、どのようにして、変更作業をしているのかは、明確ではなかった。

今回、ソフトウエア変更作業を分析して、作業内容と入力情報、出力情報を明確にした。この結果、２２の作業項目と３０の出力情報を抽出した。作業項目の６０％が、変更要求の理解や対象ソフトウエアの理解の作業に費やされている。

この作業項目に対して、保守担当者が、実際に、どの作業を、どのような順序でしているか、実態調査をした。調査対象者は、情報システムの保守担当者１５名と支援ツールの保守担当者４名である。調査の結果、プログラム仕様理解の方法が、保守担当者の経験により、異なることが判明した。保守経験の豊富な担当者は、プログラム構造の調査から始め、トップダウンに理解を進めていくのに対し、経験の浅い担当者は、ソースリストの解読などのボトムアップで理解を進めていく傾向が見られた。しかし、経験者の間でも、変更作業の入り方に相違があることも判明した。

この分析結果から、ソフトウエア影響波及検索ツールのプロトタイプ（ツール名：ＦＩＮＤＭＡＴＥ）を開発した。ＦＩＮＤＭＡＴＥは、ユーザからの保守依頼に対応して、既存ソフトウエア・リソースの相互関連を検索する。これにより、影響波及対象のリソースを抽出して、作業見積、作業計画や修正作業に利用する。検索対象のリソースは、事務処理分野システムのジョブ、プログラム、ファイル、レコード、画面、帳票、データ項目などである。あるリソースをキーにして、最大４個までネストして、関連検索ができる。また、検索結果（検索条件や検索パス）を保存しておき、再検索したり、類似の保守依頼の時に参照する機能がある。これにより、保守管理業務の効率向上と品質向上を図ることが可能となった。

## １ ソフトウエア変更作業

（１）ソフトウエア変更作業の分析

ソフトウエア変更作業は、変更要求の仕様理解から始まり、システムやプログラムの理解、変更箇所の特定化と修正作業、テストによる確認で終わる（表１）。要求仕様の理解では、要求仕様が自分のイメージと一致するか、実現できるか考える。過去の経験や見積計算式から保守作業工数を見積もる。要求者とレビューして、要求内容を確認する。

システムの全体の理解では、ドキュメントや経験からシステムの使われ方を理解して、処理パターンを推測する。処理パターンとは、オンライン処理では、データ入力や問合わせなどのパターンであり、バッチ処理では、ＤＢ更新や帳票作成などのパターンである。

システムの機能理解では、ジョブフローや画面／帳票／ファイル／レイアウトなどから、システムの動作環境、ヒューマンインタフェース部分、ファイル形式を調査する。

このように、システム全体の仕様から、プログラムの内部仕様までを理解していく。作業項目２２個のうち、１３項目（６０％）が理解に関する作業である。

変更作業は、標準手順で定められた各種のドキュメントを作成しながら進める。例えば、プログラムの変更箇所と内容を記述したプログラム変更仕様書などである。テストでは、テストケースを記述したＰＣＬ（プログラムチェックリスト）を作成して、テストレビュー、進捗と品質の管理に使う。バグが発生すれば、バグ票（Ｂ票）を発行する。

（２）保守担当者の作業分析

２２個の作業項目に対して、保守担当者の実態調査をした。対象者は、事務分野のアプリケーションシステム保守担当者１５名と支援ツールの保守担当者４名の計１９名である。この内、初級者（１ー３年）は９名で、経験者（４年以上）は１０名である。

実施している作業と作業順序、プログラム仕様理解の方法、理解能力を調査した。また、経験者３名に対して、変更箇所をどのような方法で特定化しているかを調査した。

まず、実施作業と作業順序を調査した（表２）。当然ながら、初級者はシステム理解の作業はしていない。経験者は、全作業を実施する者と、システムの仕様理解をしない者の２パターンに分かれた。これは担当しているシステムの規模によるものである。

調査の結果、プログラム仕様理解のアプローチ方法が経験の差により異なることが分かった。プログラム仕様の理解とは、プログラムの概要を掴む（外部仕様理解）ことと処理の流れを把握する（内部仕様理解）ことである。プログラムの機能概要理解は、両者共行なうが、初級者は、そこからソースステップの解読を始めてしまう。１ステップずつ読みながら、セクション単位の処理の流れ、セクションの実行順序、分岐や繰り返し条件等を理解してから、処理内容の詳細を理解する。これに対して、経験者は、他プログラムとの関連を調査して、プログラムの位置付けを把握する。次に各セクションの構成関係を明確にする。ソースリストのコメントを読んで、各セクションの概要や要求仕様に無関係と思われる処理部分を切り出す。経験者の３０％は、各セクションの構成関係を調査する前に、プログラムからデータの内容を理解している。このように、経験者はトップダウンで作業を進め、初級者はソースプログラムからボトムアップで作業をしている。

表２ 実施作業と順序

| 作業名 | 初級者 | 経験者 | |
|---|---|---|---|
| | | 1 | 2 |
| 要求仕様理解 | × | ① | ① |
| システム全体理解 | × | ② | × |
| システム機能理解 | × | ③ | × |
| ﾌﾟﾛｸﾞﾗﾑ外部仕様理解 | ① | ④ | ② |
| ﾌﾟﾛｸﾞﾗﾑ内部仕様理解 | ② | ⑤ | ③ |
| 変更仕様設定 | ③ | ⑥ | ④ |
| プログラム変更 | ④ | ⑦ | ⑤ |
| プログラムテスト | ⑤ | ⑧ | ⑥ |

表１ ソフトウエア変更作業の内容

| 作業名 | 内　　容 | 作業項目 | 出力情報 |
|---|---|---|---|
| 要求仕様理解 | 保守の目的と規模を理解する。工数を見積もる。 | 4 | 3 |
| システム全体理解 | システムの使われ方を理解する。 | 2 | 3 |
| システム機能理解 | システムの機能と動作環境、入出力部分を理解する。 | 1 | 4 |
| ﾌﾟﾛｸﾞﾗﾑ外部仕様理解 | ﾌﾟﾛｸﾞﾗﾑの機能や、他のﾌﾟﾛｸﾞﾗﾑとの関連を理解する。 | 3 | 4 |
| ﾌﾟﾛｸﾞﾗﾑ内部仕様理解 | プログラムの処理ステップを解読する。 | 3 | 6 |
| 変更仕様設定 | 修正箇所を特定化し、影響範囲を調査する。 | 4 | 4 |
| プログラム変更 | プログラムを修正する。 | 1 | 1 |
| プログラムテスト | テストケースを設定し、テストを行なう。 | 4 | 5 |
| | 合計 | ２２ | ３０ |

保守担当者が、自分の知識を、どう生かしているか、調査した（表３）。仕様の理解や変更仕様の設定を、どのようなアプローチでできているか、４段階評価をした。４は、「ほとんど的確にキーワードが思い浮かぶ」。３は、「機能レベルであたりをつけ、ドキュメントやリソースの調査で、特定できる」もので、共に保守担当者だけで作業が完結できる。２は、「機能レベルであたりをつけるが、調査に時間がかかる」もので、作業の精度は高くない。１は、「しらみつぶしに調べる」もので、上位者の指示と確認を必要とする。調査の結果、判明したのは、経験者でも、他の関連リソース／モジュールへの影響範囲の調査に自信がなく、時間をかけている点である。この変更波及調査をツールで支援すれば、保守効率が向上することが分かる。調査は、経験者３名（A，B，C）と初級者１名（D）の４名でおこなった。担当者ＡとＢは、過去に類似システムの保守を経験している。そのため、経験年数は同じでも、担当者Ｂは、Ｃと比べて、理解能力が高い。

ソフトウエア変更作業の中で、修正箇所の特定化は、担当者の経験、知識によるところが大きい。この作業に着目して、経験者３名（E，F，G）の知識と特定化作業の方法、参照資料を調査した。経験者は、リソースの知識として、処理の流れ、プログラム（名称と処理概要）、ファイル（名称と用途）を持っている。また、リソースの関連知識としては、ジョブ／プログラム／ファイルの知識を持っている。経験者の知識は、ほぼ共通であったが、変更箇所特定化の方法は、個人差が出た（表４）。

表４ 変更作業の方法

| 調査内容 | | E | F | G |
|---|---|---|---|---|
| 特定のためのきっかけ | 処理サイクル | ○ | | ○ |
| | マスタファイル | ○ | | |
| | 処理の流れ | | ○ | |
| 変更箇所までのたどりかた | ジョブ→ファイル | ○ | ○ | ○ |
| | ジョブ→ﾌﾟﾛｸﾞﾗﾑ | ○ | ○ | |
| | ファイル→COPY | | | ○ |
| | COPY→プログラム | | | ○ |
| 参照資料 | ジョブフロー図 | ○ | ○ | ○ |
| | 設計仕様書 | ○ | × | ○ |

以上の結果から、影響波及検索ツールの要件を抽出した。

・初級者、経験者が共通して使えること
・初級者が、高い品質の情報を入手できること
・経験者の個人差に対応して、多様な検索パスをサポートすること

などである。これらに基づいて、ソフトウエア影響波及検索ツールを開発した。

表３ 保守担当者の理解度

| 作業名 | 調査内容 | A | B | C | D |
|---|---|---|---|---|---|
| 要求仕様理解 | 要求仕様からキーワードが思い浮かぶか？ | 4 | 4 | 3 | × |
| | キーワードから該当設計ドキュメントが見つけられるか？ | 3 | 3 | 3 | 2 |
| システム理解 | 設計ドキュメント、レイアウトの種類が思い浮かぶか？ | 3 | 4 | 3 | 2 |
| 変更仕様設定 | 変更対象の該当リソースが見つけられるか？ | 4 | 4 | 3 | 2 |
| | 該当リソースから該当モジュールが見つけられるか？ | 4 | 3 | 2 | 3 |
| | 他の関連リソースを見つけられるか？ | 3 | 2 | 2 | 1 |
| | 他の関連モジュールを見つけられるか？ | 3 | 2 | 2 | 1 |
| | 他のシステムへの影響を洗いだすことができるか？ | 3 | 1 | 1 | 1 |

経験年数：Ａ＝７年、Ｂ＝４年、Ｃ＝４年、Ｄ＝２年

## 2 ソフトウエア影響波及検索ツール

本ツール（ＦＩＮＤＭＡＴＥ）は、保守担当者が、ソフトウエアの構成を理解したり、影響範囲を調査する作業を支援する。

主な機能を以下に示す。

・各種リソースのメンバー覧の表示
・リソース間のメンバ相互関連の検索
・連結検索（「芋づる」検索）
・検索結果の保管と再利用
・リソース・メンバに対するマーキング

対象システムは、ホストシステムのソフトウエアである。本ツールは、ＰＣで稼動する検索用のブラウザである。相互関連情報は、ホストツール（ツール名；ＶＯＳ３リエンジニアリングツール）で作成して、ＰＣにダウンロードする。

（１）リソース一覧の表示

リソース情報を一覧表の形式で表示する。リソースは、ソースプログラム、ＣＯＰＹメンバ、ロードモジュール、ＪＣＬ，ファイル、データセット、レコード、データ項目などである。図１にソースプログラムの表示例を示す。この情報は、ソースプログラムとライブラリ管理システムのデータを解析して、表示している。

表示したリソースには、印（マーク）を付けることができる。その意味や目的は、保守担当者にまかせられている。

（２）リソース相互関連の検索

相互関連の検索は、キーとなるリソース名を入力するか、一覧表から該当するリソース（複数でも、全部でも可能）を指定して行なう。図２は、検索できるソフトウエアの相互関連図である。データ項目から、プログラム、ファイルを連結検索した例を図３に示す。リソースとして「データ項目」を選択して、「ＤＴＹＪ０１」をキーとして指定する（①）。この例では、文字列検索をしている。このデータ項目を使っているプログラムとして、「ＴＹＪ０６」「ＴＹＪ１２」が検索された（②）。次に、「ＴＹＪ１２」に関連している、ファイルを検索する（③）。その結果、「ＤＴＹＪ０１」と「ＭＴＹＪ１１」の２ファイルが使われている事が分かった（④）。

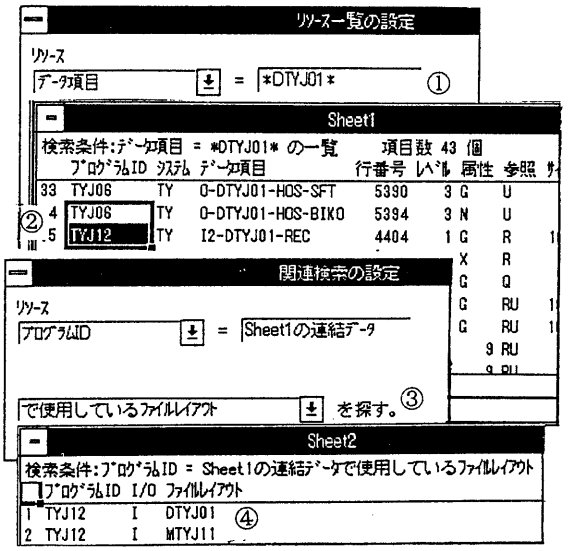


図３ 検索例（プログラム→ファイル）

| 項目 | プログラムＩＤ | システム | 言語 | STEP | コメント | 定義 | 処理 | M/S | 参照 | CALL | オンライン／バッチ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 表示例 | AI004 | AI | CO | 883 | 19 | 347 | 517 | M | 0 | 0 | B |

| 日本語名称 | 作成ユーザ | 作成日 | ソース更新日 | ソース更新ユーザ | メモリサイズ | ロード更新日 |
|---|---|---|---|---|---|---|
| 荷受け処理 | ﾀｶﾊｼﾀﾂﾛｳ | 93-06-25 | 75-02-20 | P00059 | 113160 | 93-10-13 |

図１ プログラム一覧の表示例

3）試行評価

本ツールを使った、試行を２サブシステムでおこなった（表５）。保守担当者の評価は、良好で、ツールの目的は確認された。この結果、保守作業工数の約１０％の削減を目標にすることができた。従来、熟練者でなければできなかった変更影響箇所の特定化を、若年者でも可能にした。また、副次的な効果として、若年者と熟練者とのローテーションが期待できる。

表５　評価サブシステム

| ｻﾌﾞｼｽﾃﾑ | ｿｰｽﾌﾟﾛｸﾞﾗﾑ | ｼﾞｮﾌﾞ |
|---|---|---|
| A | ８９本(33KS) | １８本 |
| B | ４４１本(311KS) | １２３本 |

課題としては、大規模システムでの実用評価がある。保守対象ソフトウエアは、数Mステップの規模のものが多く、性能などに問題が出ると予想される。また、リソースを検索キーとしているが、アプリケーションの情報（例えば、業務機能や業務ルールなど）からの検索の方が効率は良い。あいまい検索や事例検索などの検索方式も効果的である。対象言語も、ＣＯＢＯＬだけでなく、現実にはアセンブラや古い簡易言語が使われており、検索対象範囲の拡大も課題である。

## おわりに

保守作業分析と、それを支援する影響波及検索ツールについて述べた。本ツールは、現在製品化作業中である。今後は、より上流の業務情報や設計情報のリバース技術を開発して、ソフトウエアの理解支援技術を拡充させていきたい。そのひとつの試みとして、既存プログラムから業務ルールを抽出する技法を開発して、実用化評価を行っている。

### 参考文献


[1]原田 他：ＣＡＳＥのすべて,オーム社, PP.315-328,1991.11

[2]佃 他：ＣＳＳ統合開発環境(7)-リエンジニアリング-,情報処理学会第45回全国大会,PP.5-351,1992.10

[3]岡部 他：リエンジニアリング支援ツールの適用と評価,情報処理学会第46回全国大会, PP.5-231,1993.10

[6]津田：ＣＡＰＳＤＦによるリエンジニアリング,情報処理学会研究報告93-IS-2,PP.77-85,1993.1

[7]津田：ＣＡＰＳＤＦによるリエンジニアリング,情報処理学会研究報告93-SE-93,PP.165-172,1993.7


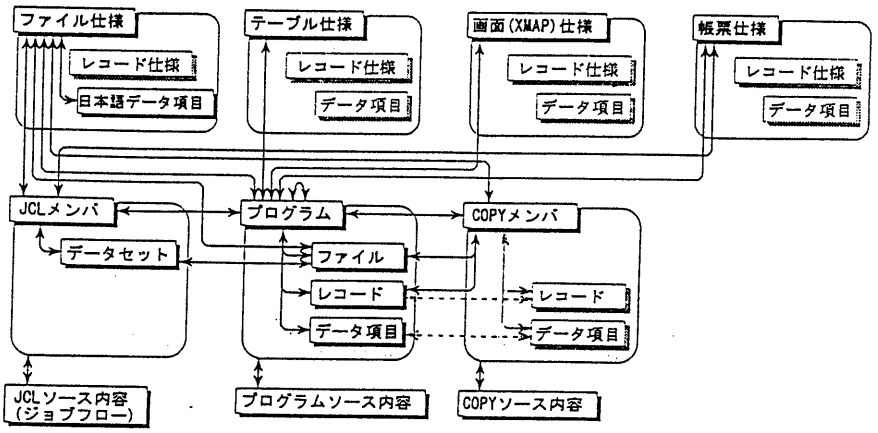


図２　ソフトウエアの相互関連図