5S-5

事務処理プログラムからの詳細仕様書生成

直田繁樹，上原三八，園部正幸
(株) 富士通研究所

1 背景

金融システムのような大規模事務処理プログラムはCOBOLで数百万ステップに及ぶ巨大なシステムであり、その保守に用いられる仕様書類も膨大な量になる。これらのシステムで、既に開発が完了しているものでは、その仕様書は大部分が手書きの文書である。このため、実際の作業においては、検索やプログラムとの対応付けに多くの労力が費やされている。こうした仕様書を計算機で管理し、その作業を効率化することで、大規模事務処理プログラムの保守作業の負担をある程度軽減することが期待できる。

しかしその前提として、仕様書が計算機で処理できる形式になっている必要がある。単純に既存の仕様書をタイプ入力するのでは、その量が膨大であるためコストが大きく、また、タイプミスの発生等、信頼性の点でも問題がある。このため、COBOLで記述された既存の事務処理プログラムから詳細仕様書を生成するシステムを実現することによって、この問題の解決を図る。

2 事務処理プログラムの特徴と本システムの機能

プログラムの処理記述部から仕様的情報を取り出す方法として、記述の日本語化、制御構造の抽出とその図式表現化等がある。一般には、これだけで仕様書と言うには不十分である。しかし、事務処理プログラムの場合、複雑なアルゴリズムはあまり使われないため、プログラムを日本語化するような比較的単純な変換でも、実際に使われている仕様書に近いものができる部分が多いことが分かった。

そこで、そのような単純な変換を行いながら、以下のような機能によってより高いレベルの変換を実現することで、現場の要求を満たすような詳細仕様書を生成することができると考えた。

1 分岐構造の表形式への変換
2 登録パターンを利用した簡略化
3 類似処理の抽出による簡略化
4 手続き呼出しの前後の処理の抽出と簡略化

一般に、仕様書とプログラムでは記述レベルが異なるので、ユーザとの対話なしでは、その機能や意図などを反映した高いレベルの仕様書を得ることはできない。しかし、一方で、大量のプログラムを低コストで処理するためには、ユーザとの対話処理は極力避けなければならないという問題もある。ここでは対象領域に依存した手法を用いることで、ユーザとの対話を極力減らしながら、生成される仕様書の記述レベルを上げることを目指している。

なお、大規模事務処理プログラムの場合、データ項目名（変数名）を、データディクショナリに登録し管理する開発形態を採用している場合が多い。このため、仕様情報生成の重点は処理記述の部分におけばよい。また、日本語化には、データディクショナリ中に登録されている日本語名を利用することができる。さらに、データ名に関する命名の規約を意味情報として利用することができる。

図1にプログラムと生成される仕様書の例を示す。これは表形式への変換と類似処理の抽出を行った例である。

3 システムの構成

このシステムの構成を図2に示す。

構文解析部では、COBOLの構文解析を行い、データディクショナリからの情報を付加して中間表現を生成する。

データディクショナリには、データ名とその日本語名、属性等の他に、命名規約から抽出したデータ名のカテゴリ情報も登録されている。

パターン処理部では、ユーザによって登録されたマクロパターンによる簡略化と類似処理の自動抽出による簡略化を行う。類似処理の抽出については後述する。

プログラムパターンライブラリには、頻繁に出現するプログラムパターンがその簡略表現と共に登録されている。

表生成部では、IFやEVALUATE等の分岐構造を表に変換して出力する。図1で示したように、同種の条件による分岐文が連続して出現する場合はマージして、一つの表にする。出力する表の形式については、生成用のテンプレートを複数用意することによって、切り替えできるようにする。また、外部手続き呼び出しのパラメタ設定処理やリターンコード判定処理を抽出し、その部分を表にまとめる処理を行う。さらに、記述の日本語化もここで行う。

Specification generation from COBOL program
Shigeki SUGUTA, Sanya UEHARA and Masayuki SONOBE
FUJITSU LABORATORIES LTD.

4 類似部抽出処理の概要

類出処理の検出のため、まず、中間表現作成時にデータ名にカテゴリ情報を付加しておく。類似処理検出部では、このカテゴリの出現頻度に着目し、最も出現頻度の高いものから順に、そのカテゴリに属するデータを含む複文を取り出す。それらを相互に比較し、その類似度（距離）を計算する。この比較において、文の出現順序はデータの依存関係に基づいて入れ替え可能なものは入れ替えて比較する。こうした比較の結果、類似度が一定値を超えるものを類似処理としてまとめる。これは、相違部をパラメタ化した上で、一意名を与えられ、プログラムパターンとして登録される。このようにして登録したパターンは、ユーザが登録したパターンと同様に扱われ、ユーザの修正により良いものにすることができる。

5 課題

今後の課題としては、大量のデータの処理を行うための効率のよい実現手法を検討することと、生成された仕様書を表形式言語への入力とすることで仕様書とプログラムを一元管理できるようにすることがある。

```
EVALUATE KOZANOCK
  WHEN 1  CONTINUE
  WHEN 2  MOVE 5 TO MSGNO
          MOVE  KOZANOIDNO TO ERRIDNO
          PERFORM SEC-ERROR
  WHEN OTHER  MOVE 6 TO MSGNO
          MOVE KOZANOIDNO TO ERRIDNO
          PERFORM SEC-ERROR
 END-EVALUATE.
 EVALUATE ZENZANCK
  WHEN 1
  WHEN 3 CONTINUE
  WHEN OTHER  MOVE 3 TO MSGNO
          MOVE  ZENZANIDNO TO ERRIDNO
          PERFORM SEC-ERROR
 END-EVALUATE.
 EVALUATE TENNOCK
  WHEN 1
     EVALUATE TENNO
           WHEN  0001 THRU 9999 CONTINUE
          WHEN OTHER
            MOVE 4 TO MSGNO
            MOVE TENNOIDNO TO ERRIDNO
            PERFORM SEC-ERROR
     END-EVALUATE
  WHEN 2
     MOVE 5 TO MSGNO
     MOVE TENNOIDNO TO ERRIDNO
     PERFORM SEC-ERROR
  WHEN OTHER
     MOVE 4 TO MSGNO
     MOVE TENNOIDNO TO ERRIDNO
     PERFORM SEC-ERROR
 END-EVALUATE.
```

図1(a) 入力プログラムの例

結果コード	1	2	3	OTHER
口座番号	処理なし	処理1 (5,口座番号)	−	処理1 (6,口座番号)
前残高	処理なし	−	処理なし	処理1 (3,口座番号)
店番号	処理2	処理1 (5,口座番号)	−	処理1 (4,口座番号)

処理1(AAA, BBB)
AAAをメッセージ番号に設定。
BBB一連番号をエラー項目番号に設定。
セクションSEC-ERRORを実行。
処理2
.....

図1(b) 生成される仕様書の例

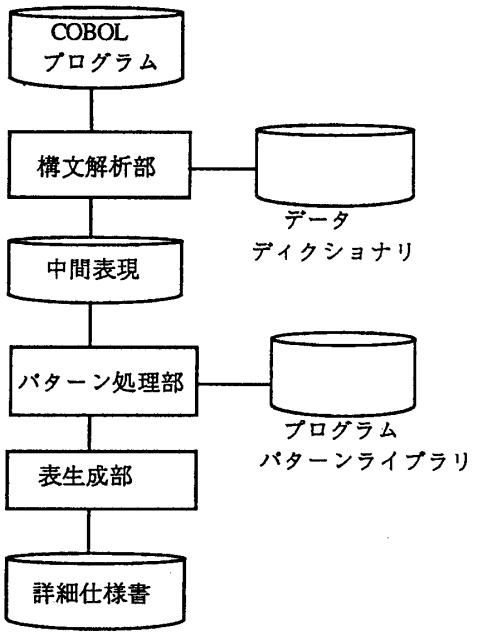

図2 システムの構成