ソフトウェア仕様解析の一方法

大槻 繁，野木兼六，葉木洋一
（日立製作所(株)システム開発研究所）　（同左 ソフトウェア工場）

1. はじめに

ソフトウェア工学の研究における一つの流れは，プログラミング言語やプログラミング支援環境に代表されるように，生産性向上を主目的とした技法やツールに関するものである。しかし，近年，開発されるソフトウェアが大規模化，複雑化するのに伴い，信頼性，保守性等の品質に関する問題がクローズアップされ，テスト技法，カバレジモニタ，あるいは，ソフトウェア科学に根ざした品質の尺度等が提案されている。

信頼性，保守性，性能，操作性等の品質に関する要求は，開発工程に沿って適切な時点で作り込まれて行くと考えられる。[1] 従って，品質を向上させるためには，各工程で要求された品質が，作り込まれているか否かの確認を行なう必要がある。また，エラーの修正コストの観点からでは，前工程で作り込まれたエラーを後工程に持ち越すと，一工程発見が遅れるごとに数倍の修正コストがかかる。よって，エラーはできる限り早期に発見しなくてはならない。これ等のことから，図1に示すように，開発工程の各段階でレビューを確実に行なうことが，いかに重要であるかが分かる。

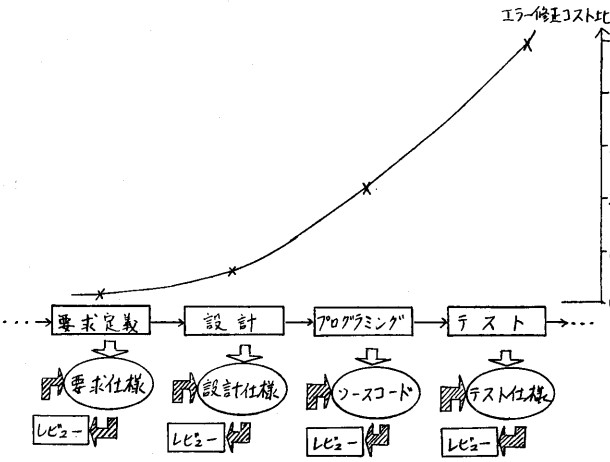

図1. レビューの必要性

レビューは，各工程で作成される仕様を対象として，その充分性，正しさ等を確認する作業である。レビュー作業には，各仕様で共通な基本的な作業と，それぞれの仕様で行なうべき独自の解析作業とがある。我々は，ここで，前者を支援する統一的な仕様解析の一手法を提案する。

本報告では，統一的な仕様解析手法の背景となる基本的な考え方，解析手法の具体例，及び，期待される効果について述べる。

2. 仕様解析手法に対する要求

2.1. 仕様解析の種類

各仕様のレビューにおいて確認すべきことは，基本的に，情報が必要かつ十分で，その上，正しく，矛盾がないことであり，おおむね，以下のように整理することができる。

（1）完全性：情報が欠落せず完備していること，すなわち，情報の十分性。
（2）最小性：余分な情報がないこと，すなわち，情報の必要性。
（3）正当性：各箇所の情報がそれ自身で正しいこと。

（4）一義性：ある記述の解釈が一通りであること。
（5）無矛盾性：2箇所以上の情報の間に矛盾がないこと。

実際のレビューにおいて，上記項目について解析しようとする場合の基本作業は，全仕様から当該解析に必要な情報，または結果としての情報を抽出することである。この抽出作業はその目的に応じて次のように分類することができる。

（1）検索：一覧表を作成するような単純な情報抽出で，指定された性質を満たすものを整理して抽出する。
（2）レビュー情報抽出：クロスリファレンス等のレビューを支援するための情報抽出で，指定された規準を満たすものを抽出する。
（3）エラー発見：エラーである規準を設定し，これにより，エラー箇所を摘出する。

我々がここで提案している解析手法は，これ等の抽出作業を統一的に支援する手法である。解析の手順そのものは，仕様の表現にのみ依存しており，仕様情報の意味上の評価や判断は基本的に人間に委ねられている。

2.2. 仕様解析手法の汎用性と柔軟性

我々がここで確立しようとしていることは，仕様の種類に依存しない仕様解析手法である。すなわち，仕様の記述方式や形式とは独立でなくてはならない。この汎用性の要求は，次節に述べるように，種々の仕様を統一的に扱うモデルを要求している。

各仕様についてレビューを行なう場合，解析項目は時と場合によって異なる。また，各種の仕様を統一的に扱うことから，解析手順や，情報抽出の規準を事前に固定化するのは難かしい。従って，解析担当者が，その場の状況に応じて，解析手順を柔軟に設定できる手法でなくてはならない。

3. 仕様情報の構造

ソフトウェアの開発で作成される仕様は，表1に示すように，さまざまな種類のものがあり，それぞれの仕様は，各仕様独自の記述様式に則っている。解析の対象となる仕様は，データフロー図，入出力データ構造図といった個々の図式や形式であることもあるし，また，より広範な解析や追跡可能性等の解析を行なうためには，いくつかの図式を組み合せた仕様情報を解析の対象としなくてはならない。このように，種々の形態を持つ仕様を統一的に解析する手法を確立するためには，仕様を統一的に把握する形式的なモデルが要求される。モデルは，表現能力が高く，自然な形のものがよいと考え，我々は，自然語を形式化したE-R（Entity-Relationship）モデル[2]を，仕様情報を把握するためのモデルとして採用した。

表1. 仕様の種類

仕様の分類	仕様書の例
要求仕様	機能階層図 データフロー図 SADT 入出力データ構造図 オペレーション・フロー 入出力フォーマット図 ⋮
設計仕様	モジュール仕様表 テーブル仕様表 モジュール関連図 ファイル構造図 メモリマップ リンクマップ ⋮
プログラム仕様	フローチャート PDL ⋮
⋮	⋮

例えば，図2(a)はモジュール関連図，(b)は階層型データフロー図によって表現されている仕様である。これ等は，それぞれ，図3(a)(b)に示すようにE-Rモデルによって記述される。図3(b)では，プロセスの階層をprt-cld，データの包含関係をd-prt-cld，データのアクセスをaccess

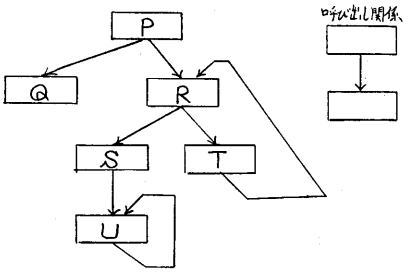

図2(a)モジュールの呼び出し関係仕様

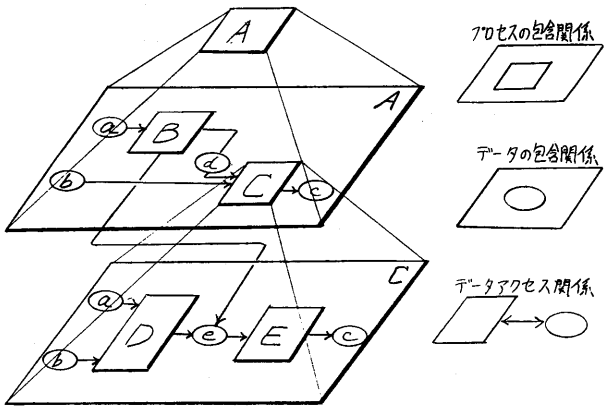

図2(b)階層型データフロー仕様

という関係によって表現している。

E-Rモデルによる仕様情報の把握は，一意に定まるとは限らないし，また，すべての情報を表現できるとも言えない。前者は，特に，エンティティー型の選択に任意性が残されており，この設定によって解析能力が左右される。後者については，特に，自然語でしか記述できない項目や，プログラム処理手順に関する情報は，たとえ，E-Rモデルによって記述できたとしても，不可解なものになってしまうことが多い。また，解析の種類によっては，元の仕様から情報を落としてE-Rモデル表現を簡潔にしておくことも考えられる。例えば，図2(b)のデータのアクセスは，入力と出力を矢印の向きによって表現しているが，データフロー解析を行なわないのであれば，図3(b)に示すように，単にアクセス関係にまとめてしまってもかまわないであろう。

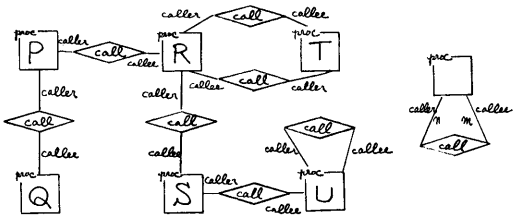

図3(a)モジュールの呼び出し関係のE-Rモデル表現

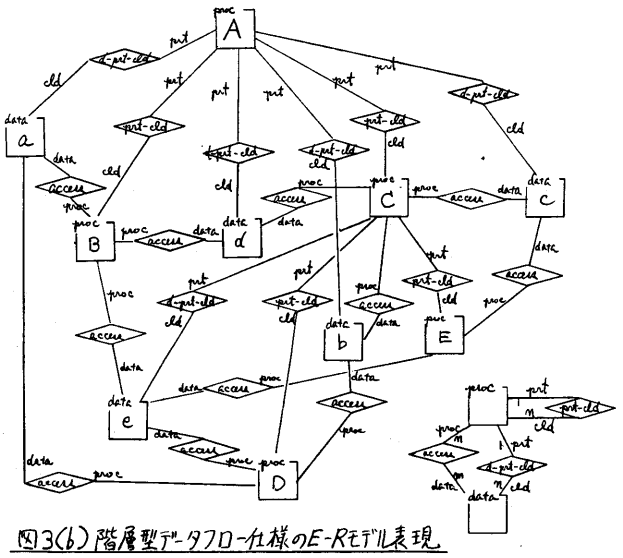

図3(b)階層型データフロー仕様のE-Rモデル表現

4. 統一的解析手法

4.1. 仕様解析のためのビュー

ここで提案する解析手法は，基本的に，仕様情報の変換手法である。従って，本手法の利用者は，仕様情報の変換過程が明確に把握できなくてはならない。

前節で，基になる解析の対象としての仕様を，E－Rモデルで表わすことを述べた。E－Rモデルは，仕様の全体像及び情報構造を把握するためには有効であるが，データ変換のオペレーションがなく，また，データそのものを管理するのがむずかしいため，E－Rモデル表現をそのまま，仕様解析時のビューとして採用するのは好ましくない。

そこで，解析時のビューとして，表形式すなわち関係形式モデル[3]を導入する。E－Rモデルから関係形式モデルへのマッピングは，エンティティとそれに結合しているアトリビュート，リレーションに結合しているエンティティとアトリビュートで，それぞれ表を作成すればよい。例えば，図3(a)(b)のE－Rモデル表現は，図4(a)(b)の表形式の記述で表わすことができる。

呼び出し関係 call

caller	callee
P	Q
P	R
R	S
S	T
T	U
R	U

図4(a) モジュールの呼び出し関係の表

プロセスの包含関係 p-cld

p	c
A	B
A	C
C	D
C	E

データの包含関係 d-p-cld

p	c
A	a
A	b
A	c
A	d
C	e

データのアクセス関係 access

proc	data
B	a
B	b
B	d
C	e
C	b
C	c
D	d
D	b
D	d
E	e
E	c
E	e

図4(b) 階層型データフロー仕様の表

E－Rモデルによって表わされた仕様情報と等価な表を基本表と呼び，図5に示すように，以後，解析をすすめるに従い，既存の表から表を変換し，新たに表を生成してゆく。この生成された表を派生表と呼ぶ。従って，要求された解析結果は，派生表の形で得られる。

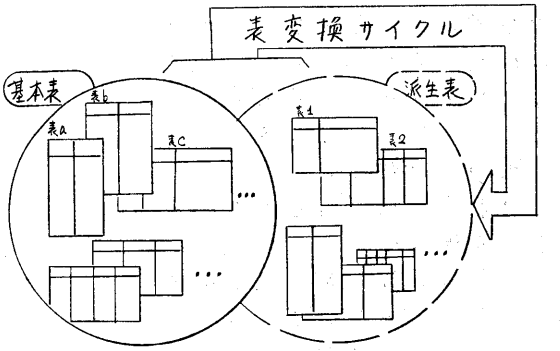

図5. 表変換サイクル

4.2. 仕様解析手順

仕様解析の手順は，表を変換してゆく過程である。表変換とは，複数個（1つでもよい）の表から，新たな1つの表を生成，定義することである。変換の種類には，設定と呼ばれるものと，これをくり返し適用した集積とがある。

設定変換の手順は，以下の通りである。

（1） 入力となる複数個の表の直積をとった表を考える。

（2） ここで得られた表の各行から，規準（表の欄，または，欄の間の条件式）を満たさない行を削除。

（3） 出力となる表の必要な欄に対し射影をとる。

（4） (1)~(3)の手順を，複数行ない，それぞれの結果に対し，集合演算を行なってもよい。

（5） ここで得られた表に対し，表名，欄名の名前付けを行なう。

集積変換は，設定変換をくり返し適用するものであり，入力の表中に，出力の表を含んでいなくてはならない。この手順は，以下の通りである。

（1）～（4） 同上

（５）　（4）で得られた表を出力表に足す。

（６）　（1）～（5）をくり返す。この停止条件は，あらかじめ設定しておいた反復回数に達するか，または，（5）で得られた表が，前回の場合と同じ時である。

（７）　表名，欄名の名前付けを行なう。

図6に，図4（b）で示した，プロセスの親子関係から，子孫関係を導き出す集積変換の例を示す。また，図7に，図4（b）の表から出発し，プロセスBが，それより下位のデータであるeにアクセスしているエラーを，抽出する過程を示す。

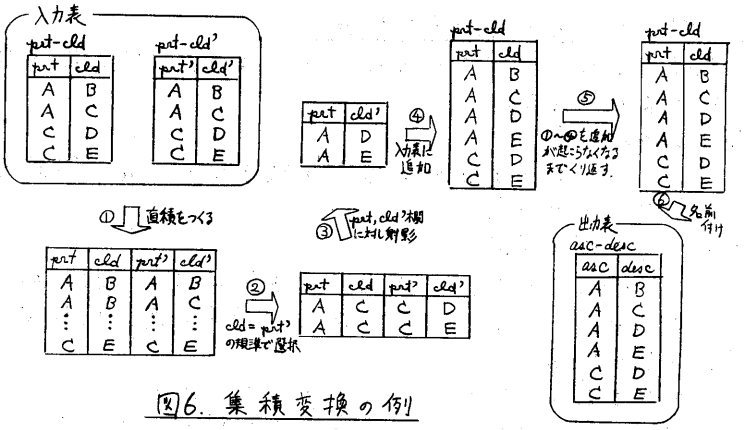

図6. 集積変換の例

このように，設定，集積変換では，入力の対象とする表，出力の表の形，選択の規準，射影する欄，及び，くり返し回数等を設定すれば，変換を完全に定義でき，これ等の変換の組み合せ順序を規定すれば，解析手順が正確に表わされる。例えば，図7に示す手順は，対象となるデータが変っても，何時でも，プロセスとデータのアクセスに関する有効範囲解析を行なうことができる。

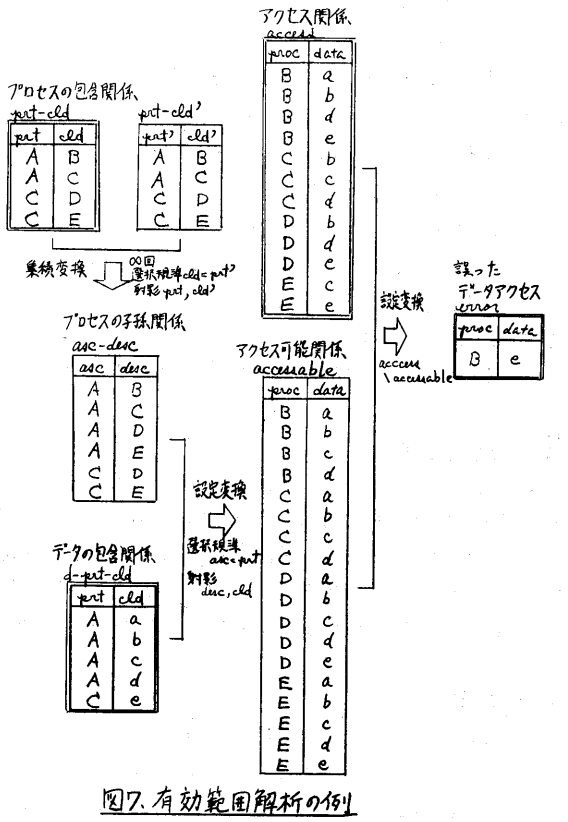

図7. 有効範囲解析の例

4.3. 解析能力

（1）エンティティの検索能力

エンティティに対するアトリビュートの値による，エンティティの抽出能力は，設定変換において，欄の条件式による行選択が可能なことにより，保証されている。

（2）リレーションの抽出能力

一つのリレーションは，一つの表に対応しており，設定変換において，欄の間の条件式による行選択が行なえることにより，

リレーションの抽出能力が保証される。

（３）リレーションの追跡能力

リレーションを表わす複数個の表から，新たな表を生成，定義することは，リレーションをたどる機能に対応しており，その上，集積変換の手順により，たどる回数を，固定回，任意回の双方を設定し解析する能力がある。

5. 設計レビューに対する効果

実際のレビューに対しては，おおむね，以下に述べるような例で，本手法が有効であると考える。

（１）単純検索

ファイルの一覧表や，作成者別モジュール一覧表といったような，整理されたデータの作成。

（２）変更波及解析

保守時において，ある場所の変更に対し，影響が及ぶ範囲を，データの参照関係や，モジュールの呼び出し関係を基に抽出する。

（３）完全性解析

参照されているにもかかわらず未定義のモジュール，必要項目が未だ記載されていない仕様等の，不完全な箇所を抽出する。

（４）無矛盾性解析

ある仕様に対し，あってはならない属性や関係を抽出する。

（５）有効範囲解析

プロセスやモジュールの包含関係により，アクセスしてはならないデータ参照や呼び出し関係を抽出する。

（６）インターフェイス解析

呼び出し側と呼ばれる側とのパラメタの個数や型に関して正合性のとれない情報を抽出する。

（７）データフロー解析

データに対する読み出し，書き込み関係をたどり，入力データと出力データの対応を解析する。

（８）構造解析

モジュール間の結合度を解析し，悪い構造をしている箇所を指摘する。

6. おわりに

仕様解析の対象をE-Rモデルによって把握し，これを関係形式モデルにマッピングした後，表の変換手順に則り解析をすすめてゆく手法を提案した。本手法の特徴は，仕様の種類によらず汎用的で，かつ，解析を手順によって示す柔軟な手法であることである。実際の解析は，設定された手順に従ってすすめられるため，解析の網羅性が要求される大規模で複雑なシステムに対し有効である。

現在，本手法を支援するシステムの開発をすすめている。

参考文献

1) 大槻：“ソフトウェア設計仕様の系統的レビュー技法”，日本科学技術連盟，第1回ソフトウェア生産における品質管理シンポジウム，'81.7月

2) P.P.Chen：“The Entity-Relationship Approach to Logical Data Base Design”，The Q.E.D. Monograph Series No.6，1977

3) E.F.Codd：“A Data Base Sublanguage Founded on the Relational Calculus”，Proc. ACM SIGFIDET, Access & Control，1971