4L-2 情報蓄積型ソフトウェア・システムにおいて安定した仕様を構築する一方法

大槻 繁
(株)日立製作所 システム開発研究所

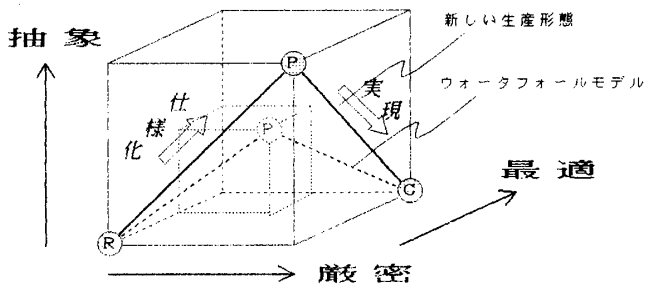
1. はじめに

ソフトウェアの生産方式は、1980年代初頭のライフサイクル論争以来、従来のウォータフォール型の生産形態では要求の抽出不能性や不確定性の問題を原理的に解決できないことが再認識され、オペレーショナル・アプローチや自動プログラミングのパラダイムに代表される新しい生産形態に移行して行く傾向にある。我々は、情報蓄積型ソフトウェア・システムを主対象として、形式的な仕様記述モデル、並びに、その仕様化技法[4)5)]を開発して来た。

この情報蓄積型ソフトウェア・システム仕様化技法に従って得られる最終的な仕様は、従来の生産形態における要求仕様やシステム外部仕様に相当するものであるが、遙かに形式的で、かつ、記述量も多い。本稿では、安定した仕様の導出方法について論ずる。ここで言う「安定」とは、ユーザのシステムに対する要求変更が起こりにくく、起こった場合でも、仕様の変更範囲が限定化され、これに迅速に対処可能なことである。

2. 情報蓄積型ソフトウェア・システム仕様化技法の概要

ここで言う「情報蓄積型ソフトウェア・システム」とは、オンライン銀行システムや販売管理システム等に見られるように、データベースを情報蓄積の中心に据え、その周りに種々の個別業務を支援するプログラムを配する複合的なソフトウェア・システムのことである。新しい生産形態では、図1に示すように、問題の定式化に相当する仕様化とその計算機による実現とを明確に区別することが特徴である[5)]。本技法は、前者の仕様化に関わるものである。



Ⓡ Real World : 曖昧な実(業務)世界のユーザ要求
Ⓟ Problem Specification : 厳密に定式化された問題仕様(ソフトウェア・システム仕様)
Ⓖ Computer World : 効率よく実行されるプログラム

図1 RPC生産形態モデル

本技法の手順は、大きく分けて3段階から構成される。ステップ1は、ソフトウェア・システム導入前の実世界の情報と作業の分析であり、ステップ2と3とは、それぞれ対象システムに対する情報蓄積要求と機能要求との抽出・仕様化である。実世界及びソフトウェア・システムいずれの記述においても、情報記述に対してはデータベースの分野において概念設計モデルとして定評のあるP.P.ChenのER(実体・関連)モデル[2)]を、機能(作業)記述に対してはM.A.Jacksonの入出力データ構造対応関係[1)]を基盤としている。技法の手順の概略は以下の通りである。

(1) ステップ1 実世界分析

ソフトウェア・システム導入前の業務分析を、既存の伝票やワークシート類を中心に行う。本ステップの詳細は、ワークシート類の収集や業務の洗い出しを行う<概要分析>、ワークシート類の抽象的な構造を分析する<情報分析>、業務手順の概要を記述する<行動分析>、情報分析と行動分析の結果をさらにデータ構造の対応関係によって詳細化する<機能分析>、及び、これ等の分析の基盤になる実世界の情報の静的な構造を実体・関連モデルで記述する<情報関連分析>から構成される。

(2) ステップ2 データベース仕様構築

データベースは、システムを構成するプログラムが共通に使用する基盤となる情報を蓄積する。このデータベース仕様は、実体・関連モデルによって記述される。安定した仕様を得るためには、実現上の理由によるその場限りの実体や関連ではなく、実世界を反映した構造を導入しなくてはならない。従って、基本的にはステップ1の<情報関連分析>の結果得られた実世界実体・関連記述を基に、その一部分か、あるいは、再構成されたものがソフトウェア・システム(データベース)実体・関連記述となる。本ステップの詳細は、実体型と関連型とを抽出しデータベース全体の枠組みを決定する<実体・関連記述>、これ等の属性を構造化して割り当てる<属性記述>、及び、関数依存性やオカレンスの制約等を決定する<制約記述>から構成される。

(3) ステップ3 機能仕様構築

最後に、ソフトウェア・システムを構成する個々のプログラムの機能をJacksonモデルに従って記述する。即ち、機能は、入力データ、出力データ、並びに、これ等の間の対応関係によって規定される。この入力、出力データのいずれかがデータベースになっていてもかまわない。この場合のデータを「部分データベース」と呼び、この記述は実体・関連モデルのデータベースに対して検索を行いその結果を木構造として得る操作指定である。本ステップの詳細は、システムの外部の入出力データ構造を決定する<データ構造記述>、部分データベースの定義を行う<部分データベース記述>、これ等の間の対応関係によって機能を定義する<機能記述>から構成される。

本技法で得られる最終的なソフトウェア・システム仕様を概念的に示すと図2のようになる。

3. 安定した仕様の構築方法

第2節で述べた技法によって安定した仕様が得られるわけであるが、本節ではその妥当性について、実世界モデルに基づくことの利点、その記述モデルとして静的なモデルを使用している理由、さらに実世界記述から仕様を導出する具体的方法について述べる。

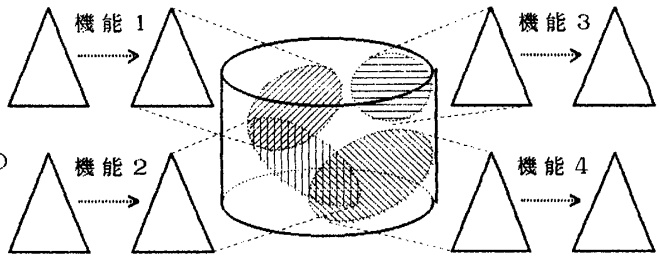


図2 ソフトウェア・システム仕様の概念

A Method of Constructing Stable Specification for Database Oriented Software Systems
Shigeru OTSUKI
HITACHI, Ltd.

3．1　実世界モデルの必要性

Jacksonのシステム開発技法ＪＳＤ[2]や、Chenのデータベース開発技法としてのＥＲアプローチにおいても実世界モデルを基礎としている。我々の技法は、これ等の技法で提唱している実世界記述を統合化したものとして位置づけることができる。ソフトウェア・システム自体の仕様を決める前に実世界の記述をし、これを基に仕様を構築して行く利点は、概ね以下のように考えられる。

(1)ユーザの介在している業務世界を具体的に記述することによって、ユーザとシステム開発者との間のコミュニケーション・ギャップを埋めることができる。

(2)ユーザの実世界で確立している広範な業務知識をソフトウェア・システムに取り込む仕掛けを作ることができる。

(3)ソフトウェア・システムを取り巻く環境（即ち個別の機能要求）より、ユーザの業務を取り巻く環境（実世界モデル）の方が遙かに安定しており、これを基に仕様を構築した方が仕様全体の保守性を高くすることができる。

3．2　静的なモデルの有用性

本技法では、実世界の情報構造（ステップ１＜情報関連分析＞）、並びに、データベース仕様（ステップ２）において実体・関連モデルという静的なモデルを使用している。一般的に、実世界すべてを記述するには、静的なモデルでは不十分であるし、実世界を観察していて第一に認識可能なものは動作（事象）である。極端な場合では、ＪＳＤにおいて、実世界の記述モデルは動的なモデルしか許していない。しかし、少なくとも情報蓄積型ソフトウェア・システムを対象とする場合には、若干事情が異なる。本技法において、静的なモデルを使用している主な理由は以下の通りである。

(1)情報蓄積型ソフウトウェア・システムでは、情報そのものが資源であり、実世界においても安定して存在している。

(2)抽象度の高い業務知識は、静的な形で集約されている。（例えば、「法人」や「個人」といった概念、さらには、これ等を一般化した「顧客」は、実体型である。また、「契約」は、関連型の典型例である。）

(3)実現工程においてデータベース管理システムやファイル・システムを使う場合が多く、この時のスキーマ設計やレコード形式への変換方式が確立している。

3．3　実世界記述との対応

静的なモデルとしての実体・関連モデルと動的なモデルとしてのJacksonモデルとを、実世界記述（ステップ１）と仕様記述（ステップ２、３）の両者に使用しているが、ここで問題になるのが、いかにして実世界記述から仕様を導出するかということである。その方法は、基本的に範囲限定あるいは境界付けによる。例えば、図3は、銀行システムにおいて基になる実世界の実体・関連記述の一部である。（「銀行」は、いくつかの「支店」を「所有」している。「支店」と「顧客」の間には「口座契約」が結ばれる。「口座契約」は「口座」として実体化される。為替業務（他行間の送金をも含む）は、「払込人」が「支店」において直接現金にて申し込んでもよいし、「口座」から直接引き落とし指定をしてもよい。逆に、「受取人」は「支店」において直接現金を受け取ってもよいし、「口座」へ直接振り込むように払込人が指定してもよい。いずれにせよ、為替業務は「払込」側と「受取」側との「為替契約」として認識される。）

これ等の実世界記述に対して、仕様を導出する典型的な方法は以下の例に示す通りである。

(1)オカレンス選択：対象とするソフトウェア・システムが、ある一つの銀行用のものである場合、実体集合「銀行」の要素（オカレンス）は唯一である。システムで管理すべき「顧客」は、この場合、口座契約を結んでいる実体要素に限られる。

(2)属性選択：「顧客」の属性には、「氏名」、「年齢」等の実世界の個人の属性が数多く考えられるが、ここでは、口座契約に必要な属性のみに限定化される。

(3)境界付け：例えば、為替送金が当該銀行から他銀行へ、あるいは、その逆の場合には、システムの境界は、関連集合「為替契約」に在る。

以上により、仕様化されたデータベース仕様を図４に示す。この様に実世界と対応したデータベースを構築し、実世界の変化に対応してデータベースを更新することによって実世界を模擬するように、機能仕様を決定して行く。

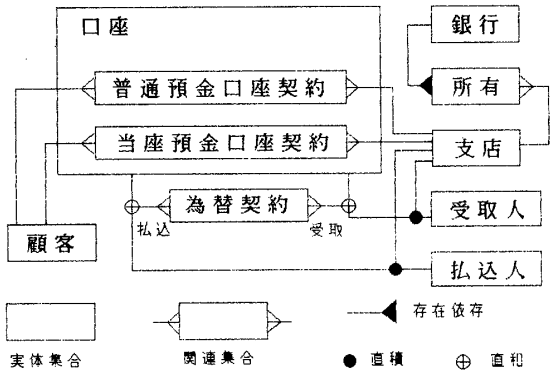


図３　実世界実体・関連記述（属性は省略）

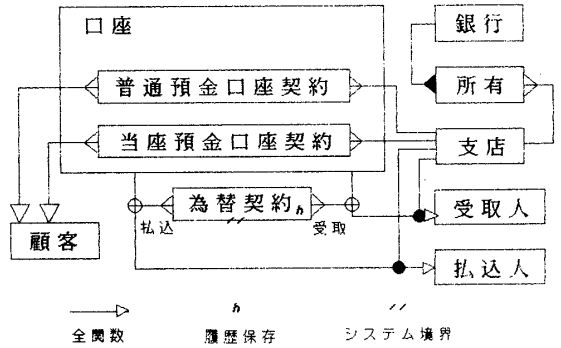


図４　データベース実体・関連記述（属性は省略）

4．　おわりに

本稿では、情報蓄積型のソフトウェア・システムを主対象とし、実世界の記述を基礎として仕様化を進めて行く技法について考察した。最終的な仕様は、実世界の情報の状態を模擬するように定式化される。実世界を明確に記述することによって、ユーザと開発者との間のコミュニケーションギャップを埋めることができ、従来の生産方式でよく見受けられた誤解に基づく要求変更は起こり難くなる。また、実世界に存在する情報自身が変わらない限り、システムの情報蓄積要求は変動しない。その意味において非常に安定した仕様を得ることができる。種々の機能要求は最後の段階で仕様に付加されるために技法全体の手戻り作業が少なくて済む。

【参考文献】

[1]Jackson, M. A.:"Principles of Program Design", Academic Press, 1979

[2]Cameron, J. R.:"An Overview of JSD", IEEE Trans. on Software Eng., Vol. SE-12, No. 2, Feb. 1986, pp. 222-240

[3]Chen, P. P.:"The Entity-Relationship Model toward a Unified View of Data", Trans. Database Syst. 1, 1, Mar. 1976, pp. 77-84

[4]大槻:「データ中心型ソフトウェア仕様化技法の一考察」, 情報処理学会第３５回全国大会, 1987 9月, pp. 1029-1030

[5]大槻:「データモデルを導入した仕様化技法」, 情報処理学会ソフトウェア工学研究会, 1987 11月