仕様獲得支援システム：K−SCORE（3）
−仕様精錬部の構成とその機能−

1G−3

°西村一彦　久野禎子　中村英夫
（株）東芝　システム・ソフトウェア技術研究所

1．はじめに

仕様獲得支援システムK−SCORE[1]は顧客の要求を構成・意味の両面から実現可能な機能仕様に変換するプロセスを支援することを目的としている。

本稿ではそのサブシステムである仕様精錬部について述べる。

仕様精錬部は設計者（本システムのユーザ）が要求仕様を詳細に理解するプロセスの支援を行う。このプロセスは仕様構成部[2]の結果を入力とし、中心的問題は
(1)要求機能の実現性を確認する
(2)実現性が見いだせない場合にその箇所を特定し、改善するための代替案を作成する
(3)要求の変更がなされる場合はそれを示す
ことにある。

本稿ではこの問題に対して、まず、仕様化プロセスを
(1)要求機能から実現可能な機能への詳細化
(2)詳細化によって生じる機能間の相互作用の発見・改善
(3)仕様の修正に伴う要求の変更を予測
という3つのタスクからなると考え、それぞれの支援を提供する。本稿では(1)-(2)について述べる。この過程では実際にシステムを設計するための詳細な知識が得られる。具体的に本システムは以下の特徴を持つ。

●機能の詳細化をガイドするために要求機能をもとにしてプランを作成する。
●プランの作成の時に生じる不具合点の発見をプランの因果的な解析によって行う。

2．仕様精錬とプランニング

ソフトウェア開発における仕様獲得フェーズの目的は「顧客の要求を実現可能な機能仕様に変換する」ことにある。つまり、顧客の要求（ゴール）を設計者の立場から分析し、基本的なシステムの動作列（機能）を得ることにある。この機能は顧客の目的を設計者が達成するためのプランに相当する。

ところで、一般にプランニングは与えられたゴールを達成するための行動に関する推論を行うことである。「ゴール＝顧客の意図」とみなすと、「行動の推論＝機能の推論」と考えることできる[3]。従って、プランニング手法を応用することによって顧客の要求に基づいたシステム機能の獲得が支援できる。具体的には
(1)要求を実現するための機能の作成
(2)仕様を詳細化のためのプランの作成
の支援が期待できる。また、プランニングの抽象化技術は類似機能の発見や、仕様の修正の能力を向上させるのに役立つものと考えられる。

プランニングではその興味の対象は達成を望んでいる目標であるのに対して、仕様では望んでいない目標＝禁止条件にも興味が置かれることがある。従って、欲する機能が禁止条件を達成しないように条件を獲得したり、新しい機能を追加するという重要な問題が残されている。

3．方法

3.1 要求機能の詳細化

仕様精錬部の最初のステップは仕様構成部から得られるオブジェクト記述をオペレータに変換する。オペレータはオブジェクトの動作に関する記述であり、オペレータの適用条件と結果からなる。さらに、結果はオペレータの適用によって成立しなくなる状態と新たに成立する状態記述である。これらは以下に述べるプランニング知識となる。

ここで、実現可能な形式とは
(1)システムの目的に対して、正しく作用すること
(2)抽象的な機能は、設計者が理解できる形式であること
と定義する。以下(1)-(2)の実現方法を述べる。

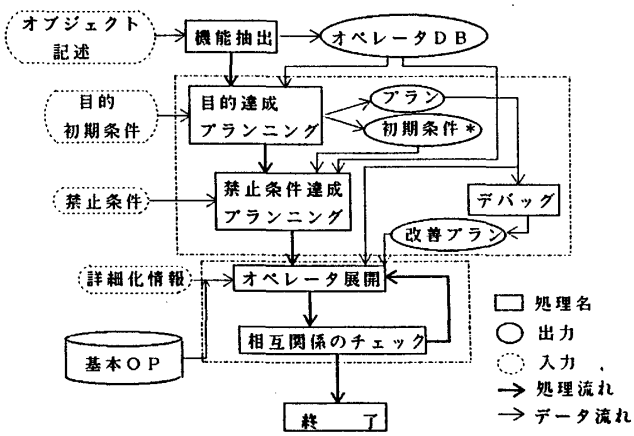

図1　精錬部の処理

図1は精錬部の処理の流れを表したものである。
(1)システムの目的には様々なレベルが存在するが、ここでは、設計の対象システムのある時点でのシステム挙動（システムの状態）とする。具体的にはシステムを構成するオブジェクトの状態変数の具現値、オブジェクト間の構成関係などである。禁止条件は達成したくないシステムの挙動であり、記述は目的と同様である。プランナは与えられた目的がオブジェクトの機能（オペレータ）によって達成できるかどうかを推論する。推論結果であるプランはゴールを達成するためのオペレータ集合とオペレータを作用させる順序、およびその初期条件である。次に禁止条件に対するプラン生成を試みる。このプラン生成に失敗した場合はプランの解析によってその原因を特定する。その方法についてはデバッグの部分で述べる。
(2)(1)で得られたプランは設計目標を達成する機能の列であり、設計をガイドするプランであることから、このプランに従って各機能を詳細化する。まず、各オペレータを構成する部分オペレータおよびそれらの順序関係を入力する。例えば、エレベータが移動するというオペレータは①移動命令を受ける、②移動先の設定、③移動するための動力系への命令、順序関係は①は②の前、②は③の前であるというように詳細化情報を入力する。こうしたオペレータの詳

Knowledge-based System for Specification Construction,Refinement and Execution:K−SCORE (3)
- Specification Refinement Subsystem -
Kazuhiko NISHIMURA, Yoshiko KUNO, Hideo NAKAMURA
Systems and Software Engineering Laboratory, TOSHIBA Corporation

細化と同時に各オブジェクトが持つ状態変数の追加や修正を行う。

支援システムはこれらの次々に入力される詳細機能や状態変数から、ユーザが次に何を入力すべきなのかをガイダンスする。すなわち、

●個々のオペレータを詳細化することによってプランが詳細化される。その結果、抽象レベルでは発見できなかった機能間の相互作用が発見される。相互作用とはある機能を作用した結果が別な機能の条件となることである。この相互作用を示すことは機能間の順序関係を獲得する上で重要な情報となる。

●設計者が完全に領域知識を持つことは希である。そこで、本システムでは領域に関する知識を基本オペレータとして与える。プランナは基本オペレータを合成し、現在のオブジェクトの機能が実現できることを保証する。

●状態変数の詳細化はオブジェクト間のインタフェースの違いとなる。従って、状態変数を詳細化した場合にその影響がどこに及ぶかを抽象レベルのプランを用いて予測することができる。

以上のように詳細化によって仕様を実現するための知識を獲得する。

3.2 不具合点の発見

詳細化をガイドするプランは禁止条件を考慮した目的を達成するものであるが、これが要求機能から完全に得られるかどうかは仕様を構成する段階で、目的が明確であり、設計上の制約をどれだけ意識しているかに依存する。設計を考慮したプランが得られることは詳細化をスムーズに行うために重要となる。よって、プランニングが失敗する場合、その失敗の直接的な原因を発見することが仕様獲得をスムーズに進行させる要因となる。この失敗の原因が何かを解明するにはそうした失敗状態を導くプランの解析が必要となる。

［失敗状態］

失敗状態とは目的が達成できない状態である。具体的にはサブゴールの未達成、サブゴールに矛盾する状態の達成をいう。

失敗プランの原因として

(1)適用すべきオペレータがない：機能そのものが存在しないことである。

(2)オペレータの適用の誤っている：機能の作用の条件が誤っている。これは条件部の誤りと結果部の誤りである。

という２種類が考えられる。

［デバッグの前提］

設計者は失敗状態を認識でき、失敗プランの各オペレータに関して完全に知っている。また、失敗状態に至るまでの中間状態は正しいものとする。

失敗状態は先の定義より望むべき記述がない場合（目標条件の欠落；LC）と望んでいない記述が存在する場合（禁止条件の存在；EP）に分類する。すると、その原因は４つに分類できる。

(1)LCの原因

(1a)LCを削除するオペレータが作用可

(1b)LCを追加するオペレータが作用不可

(2)EPの原因

(2a)EPを追加するオペレータが作用可

(2b)EPを削除するオペレータが作用不可

(1b)(2b)の場合は直接原因となるオペレータが何かを明確にすることが重要であり、失敗プランの分析とは直接無関係な手続きが必要である。ここでは、それが明らかになっている場合を考え、そのオペレータが適用されない原因を失敗プランから発見する。また、(1a)(2a)の場合は失敗プランから直接作用しているオペレータは簡単に見つけることができる。（これらのオペレータをOPとする）　あとは、これらのオペレータの前提条件についてチェックすることが必要となる。図２は単純なオペレータの列であるプランを状態間の因果構造を考慮に展開したものである。失敗原因の発見はこのプラン構造のトレースを基本とする。

(T1)OPの結果部（削除リスト、追加リスト）をチェックする。これに誤りがあれば、OPが原因である。

(T2)OPの前提条件(P)をチェックする。Pと状態Siの関係をチェックする。Siが正しい場合はPをもたらすプランに問題がある。PがSiで成り立つことが不可能であると考えた場合、Pそのものが誤りか、もしくは、Si-P（SiからPを除いた部分）に誤りがある。Pに誤りがある場合はOPそのものが原因である。Si-Pに誤りがある場合は、その誤りを導くオペレータについて(T1-2)を適当する。図２の場合、54はOP3の削除リストにある。よってOP3の前提条件22,23をチェックする。今、22がS3で成り立たないとすると、その原因はOP1以前に存在することになる。

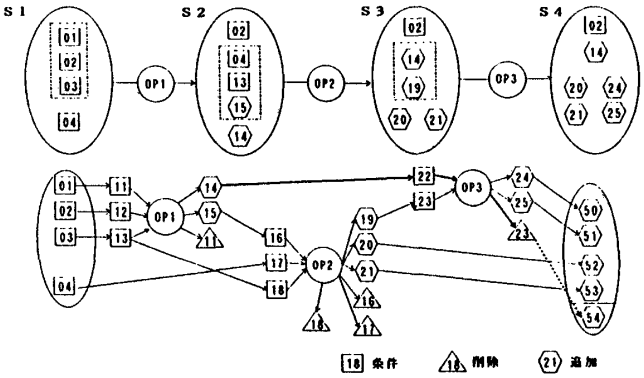

図２　プランのデバッグ

プランの因果解析は失敗原因と直接関係のない箇所をデバッグの対象からはずすことができ、無駄な手続きを省くことができる。

4．おわりに

仕様精錬部は要求機能を詳細化することによって機能の実現性を保証する。このタスクはプランニングの応用によって行われ、

●設計目的に対するシステムの機能合成支援

●プランに基づく要求機能の詳細化のガイド

●領域知識によるプラン知識の補完

という特徴がある。詳細化をガイドするためのプランニングの結果から得られる、システムの機能間の因果関係や順序関係知識は制御システムにおいては重要なものである。また、これらの情報は、機能実現の失敗や機能間の相互作用の原因を特定し、修正方法をガイドする知識として意味がある。

現在、階層プランナを用いて、これらの機能の実現を行っている。抽象デバッグ等のデバッグの戦略を拡張することが課題である。

［参考文献］

[1]中村,西村,久野,「仕様獲得支援システム：K−SCORE(1)−システム概要と仕様実行部−」，本予稿集.

[2]久野,西村,中村,「仕様獲得支援システム：K−SCORE(2)−仕様構成部の構成とその機能−」，本予稿集.

[3]Anderson,J.S. and Fickas,S.,"A Proposed Perspective Shift: Viewing Specification Design as a Planning Problem",5th IWSSD,Vol.14,No.3,1989.