仕様化支援方式に関する研究

4K−3

浜田 雅樹　竹中 豊文
ATR 通信システム研究所

1.0　はじめに

要求分析作業は、要求者が提示する、断片的、曖昧、矛盾を含んだ問題領域固有の問題を設計技術的な観点から整理･体系化する作業と言うことができる。一般に要求分析はelicitation, formalization, validation作業からなる[1]。その中でも、elicitationに関した、要求を要求者より引出し･整理する作業は効率が悪い作業となっている。この原因の１つとして、要求者は設計知識を、要求分析者は領域知識を各々持たない事が考えられる。本研究では、コンピュータに領域知識、設計知識を持たせて、要求分析者が行う要求の引出し･整理作業を支援する仕様化支援方式について提案する。以下、基本的な考え方について述べる。

2.0　仕様化支援方式の概要

一般に要求は、領域知識上の常識が省略されたり、断片的、矛盾を含んだ状態で提示される。したがって、仕様化支援として以下の機能を実現する必要がある。

- 省略、曖昧性を含んだ要求の厳密化。
- 実現可能性、矛盾の検査。
- 要求を思いつくままに構築し、それを整理するための支援機能。

Fig.01に提案する仕様化支援方式の概要を示す。要求分析者は要求者が示した要求を仕様化支援システムに入力する。システムは、持っている領域知識、設計知識を用いて、省略の予想、矛盾のチェックや要求の整理等を行う。要求分析者はその結果を基に、要求者に真意を尋ね、関連した要求を引き出す。

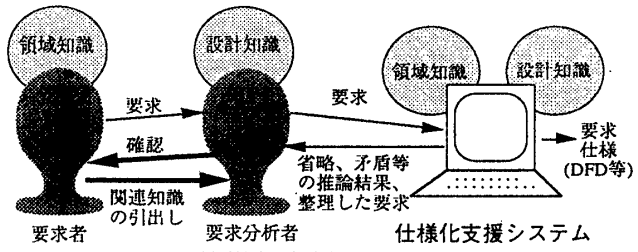

Fig.01 仕様化支援方式の概要

2.1　要求分析のプロセスモデル

設計活動の分析結果[2]などに基づき、要求仕様設計作業を以下の様にモデル化する(動作が複雑でない事務処理プログラム等を前提)。

(1) 振舞いの イメージ化

① 主処理の作成 --- その処理の基本的な振舞いを考える。

② 分岐の作成 --- 主処理から別の処理ケースを発見し、追加する。

(2) 構造化

(1)で作られた振舞いにおける冗長性の排除、条件の一般化などを行う。

本仕様化支援方式では、この設計活動のプロセスモデルに沿って要求仕様の構築を支援する。振舞いのイメージ化には、タスクに関する知識(領域知識)が、構造化に対しては設計技術的な知識が大きく関与すると考えられる。各々の知識ベースを用意し上記モデルの各ステップを支援する。

2.2　知識ベース

以下に主な知識ベースを示す。予め問題領域ごとに定義しておく。

(1)処理部品ライブラリ(領域知識)

- 処理ユニット--- その問題領域の用語で表された処理で、状況等で意味する処理内容に幅がある。要求者からの要求は処理ユニットの処理順序で表現する。
  処理ユニット:== <コア処理>, <オプション処理列>
  コア処理 :== 処理モジュール
  オプション処理列 :== 処理モジュール｜処理モジュール, <オプション処理列>
  コア処理とは、その処理ユニットが必ず含む処理を、オプション処理は状況によって含む場合もある処理を表す。
- 処理モジュール --- 処理内容が固定。属性として<pre-cond>,<post-cond>をもつ。pre-condは、その処理を実行するために満たされていなければならない条件、post-condは、その処理を実行したあと新たに成立する条件を表す[3]。pre-, post-condは条件要素名とその状態で表す。一般に、これらは処理するデータや制御情報等を表すために用いるが、状態まで記述するため、単なるデータの入出力情報と異なり、処理間の意味的な接続条件を表すことができる。

(2)条件要素(領域知識)

pre-, post-condの表現で用いる条件要素の属性として、取り得る状態や分岐属性(分岐を起こす可能性の有無等)を管理する。

(3)構造化規則(設計知識)

作成された振舞いを、処理の冗長性の排除、分岐条件の一般化などの観点から整理するための規則群。

2.3　振舞いのイメージ化支援

ユーザが振舞いとして入力した、処理ユニットの処理順序&分岐について、システムが、可能性がある

A requirement elicitation support method
Masaki Hamada, Toyofumi Takenaka
ATR Communication Systems Research Laboratories

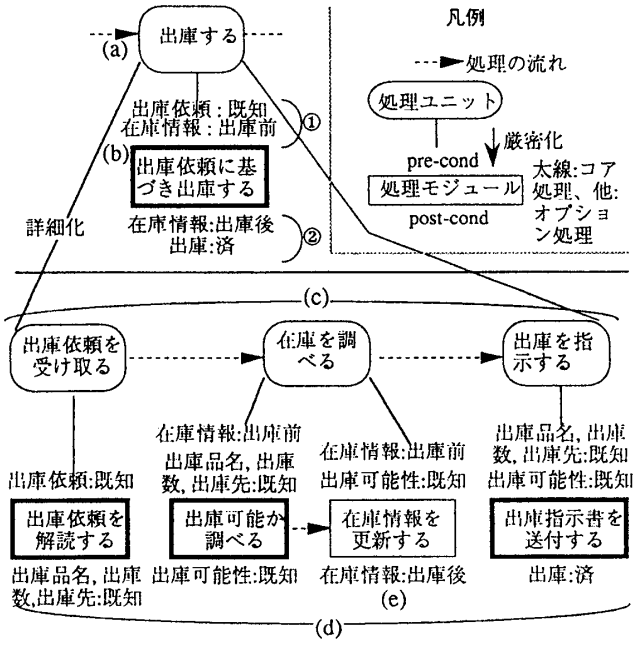

Fig.02 要求の省略、抜け等の推論

省略、抜け等を推論(=厳密化)する。

主処理の作成プロセスでは、システムが、要求として入力された各処理ユニットのオプション処理として可能なものを選択し、その意図する処理内容を表す可能性がある処理モジュールの列へ展開する。その具体例をfig.02に示す。今、要求として入力された処理ユニット"出庫する" (図中a)が本支援機能により、コア処理"出庫依頼に基づき出庫する"のみに厳密化されていたとする(b)。ユーザはさらに、"出庫する"を3つの処理ユニット"出庫依頼を受け取る"、"在庫を調べる"、"出庫を指示する"へと詳細化したとする(c)。システムは、これら3つの処理ユニットそれぞれのオプション処理の選択パターンより得られる処理モジュール列(dに相当)の中から以下の詳細化制約を満たすもの(=厳密化の解)を求める。

- 処理モジュール列(d)のpre-cond=上位の処理モジュール列のpre-cond(図中の①に相当) かつ
- 処理モジュール列(d)のpost-cond⊇上位の処理モジュール列のpost-cond(②)

処理モジュールの実行により、そのpost-condが、それまでに成立している条件の集合(以下条件集合)に追加(または書き換え)される。処理モジュールは、そのすべてのpre-condが、その直前の処理モジュールを実行した直後の条件集合により満たされる場合、実行可能であると言う。処理モジュール列のpre-condとは、列の処理モジュールをすべて実行可能にするために必要十分な、列に対する初期条件の集合を指す。処理モジュール列のpost-condとは、列最後の処理モジュールを実行した直後の条件集合を指す。

Fig.02で示した具体例では、このような条件を満たす厳密化の解(d)を示している。処理モジュール列(d)のpre-condは、(出庫依頼:既知、在庫情報:出庫前)であり、上位の処理ユニット"出庫する"のpre-cond(①)と一致している。また、処理モジュール列のpost-condは、(出庫品名,出庫数,出庫先:既知、出庫可能性:既知、在庫情報:出庫後、出庫:済)となり、上位の処理ユニット"出庫する"のpost-cond(在庫情報:出庫後、出庫:済)を含んでいる。

この解では、処理ユニット"在庫を調べる"が持つオプション処理の中の"在庫情報を更新する"を選んでいる。この結果より、ユーザが要求として示した"在庫を調べる"といった処理ユニットは、出庫可能か調べ、かつ在庫情報を更新する処理まで含んでいる(e)可能性があることをシステムが示唆する。システムは、このように厳密化された結果を基に下流工程の設計者に示す要求仕様書を作成する。また、ユーザが示した要求が上記制約を満たす解を持たない場合(=実行可能性、矛盾の検査)、ユーザに警告を発し、要求の再考を促す。

また、分岐の作成プロセスでは、処理ユニットのpost-condになっている条件要素の分岐属性(知識ベース参照)より、考えられる処理の分岐についてユーザに確認する。

2.4 構造化支援

Fig.03に構造化支援の例を示す。冗長な処理をまとめる規則などにより、思いつくままにに記述されていった要求(振舞い)を半自動的に整理する。

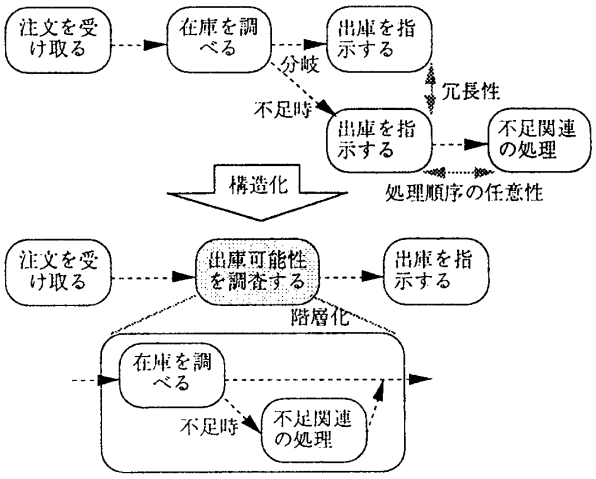

Fig.03 構造化支援の例

3.0 今後の課題

事務処理などの問題領域について、領域知識の記述力上の問題点を明かにする。さらに仕様化支援プロトタイプシステムの試作を通し、支援内容の適切性等について分析を行う。

【参考文献】

1. H.Reubenstein, R.Waters, The Requirement Apprentice：Automated Assistance for Requirements Acquisition, IEEE Transactions on Software Engineering, Vol.SE-17, No.3, 1991
2. B. Adelson, E. Soloway, The Role of Domain Experience in Software Design, IEEE Transactions on Software Engineering, Vol. SE-11, No.11, 1985
3. D. Perry, Software Interconnection Models, Proceedings of 9th International Conference on Software Engineering, 1987