ソフトウェア仕様記述の支援環境に関する研究

95B547 加藤 健

指導教員　野呂 昌満

1　はじめに

現在, ソフトウェア開発の大規模化に伴い, 初期段階の要求分析は重要であると指摘されている. 曖昧な点や矛盾が存在することなく要求分析を行なうことで, ソフトウェア開発が円滑に行なわれる. 要求分析では成果物として要求仕様書が作成される.

要求仕様書を記述する方法の 1 つとして, 数学的な理論に基づいた形式手法が知られている. 形式手法により仕様の曖昧性や矛盾を検証することができるが, 厳密に仕様を記述することは困難である. また, 過去に記述された仕様を蓄積し, それをパターン化し, 再利用することは従来まれであった.

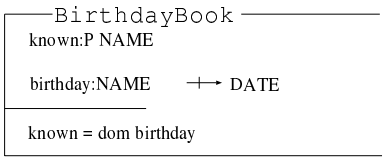
最終的な目標として, ソフトウェア設計者の仕様作成を支援するツールの作成を目指す. 従来これらのツールを作成するには, 各ツール開発者が独自に仕様の解析を行なうことが多かった. しかし仕様の解析ルーチンの作成は手間がかかる. もし各ツール間で共通に利用できる解析ルーチンが存在すれば仕様作成の手間が軽減される.

本研究では, ツール作成を容易に行なうために共通のパーザを基にした環境の構築を目指す. まず仕様記述の構造や意味をモデル化し, モデルにしたがって仕様を解析するパーザを作成する. 解析された情報はデータベースに格納される. ツールは, データベースにアクセスすることによって仕様の情報を容易に取得できる. またデータベースを利用して, 仕様でよく用いられるパターンの発見や, 仕様の再利用を支援することもできると考えられる. 仕様記述言語としては比較的分かりやすく, 広く普及している Z を用いる.

2　従来の研究との違い

従来の研究では仕様の型チェックや証明, 検証などが中心であり, 仕様の再利用支援などはあまり考えられていない. また実装の成果物をライブラリやフレームワークとして設計段階では, デザインパターンやソフトウェア・アーキテクチャとして再利用することができる. しかし要求分析の成果物である要求仕様書はほとんど再利用されていない. その理由として要求仕様書が問題領域に特化していることや, 仕様書の蓄積がほとんどされていなかったことが考えられる.

Z の表記方法としては LaTeX や troff による表記が一般的である. しかしこれらの表記法では構文が明確ではない. 本研究では, 構文が明確な PiZA による表記法を用いる.[図 1]




```
{sb BirthdayBook
  known : pset NAME ;
  birthday : NAME -+-> DATE
----
  known = dom birthday
}
```

図 1: PiZA による表記

3　処理系の構成

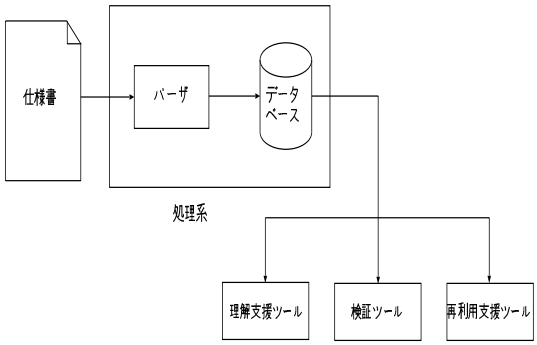


図 2: パーザとデータベースの関係

図 2 は, ユーザの仕様作成を支援する際に必要な手順を示している. これは, 仕様をパーザで解析し, そのデータをデータベースに格納することを表している.　本研究の目的としては, 仕様を解析する共通のパーザとデータベースを作成する.　これを行なうことで, ツール開発者が独自のパーザやデータベースを作成することなく支援ツールが作成できる. 例えば, 理解支援ツール, 検証ツール, 再利用ツールがある.

4　パーザの設計と実現

パーザでは一般のコンパイラと同様に仕様を字句解析, 構文解析, 意味解析する.　意味解析では, 変数があらかじめ正しく定義されているか演算子が正しい型に適用されているかなどを調べる.

パーザは図 3 のモデルにしたがって解析する. 図 3 のモデルは Z によって記述されたシステム全体の構造を表している. システムはスキーマと補助定義から構成されている.

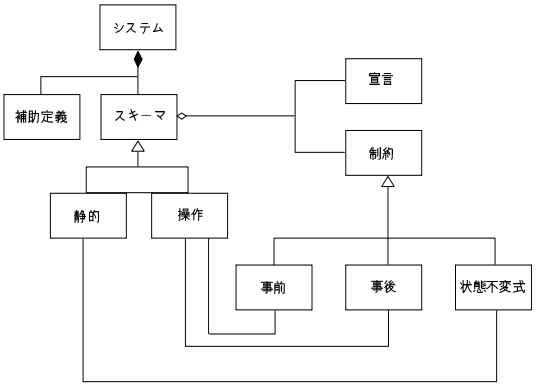


図 3: 仕様のモデル

スキーマは静的なスキーマと操作を表すスキーマに分類することができる. 静的なスキーマはシステムが常に満足すべき制約を記述する. ここで記述された制約式は状態不変式と呼ばれる. 状態不変式はどのような操作を行なっても, その状態は変わらないことを表す.

操作を表すスキーマは, システムに対する操作を表す. 操作を表すスキーマの述語部は事前条件, 実際の操作を記述する部分, 事後条件に分類される. 事前条件は, 操作を行なう時に満足しなければならない条件である. 事後条件は, 操作を行なった後で満足しなければならない条件である.

5 応用例

われわれの支援環境を用いることにより以下のことが可能になると考えられる.

1. 仕様の再利用支援

   誕生日帳の仕様が既に記述されているときにこれを再利用して住所録の仕様を作成することを考える. 誕生日帳では, 人の名前を表す NAME 型と, 誕生日を表す DATE 型を使用し, 誕生日帳に新しい人名と誕生日を追加する操作 AddBirthday や人名から誕生日を検索する操作 FindBirthday などが定義されているとする.

   この仕様を再利用して住所録の仕様を作成するには, 住所を表す $ADDR$ 型を定義し, 仕様の日付を表す $DATE$ 型を利用している部分を, $ADDR$ 型に変更する必要がある. われわれのモデルでは宣言された基本型の $DATE$ とそれを参照している部分には関連が生成されている, したがって $DATE$ を利用している部分を容易に取得でき, 変更が必要な箇所を知ることができる.

2. 類似した仕様の検索・仕様のパターン化

   例えば誕生日帳と住所録の仕様を考えると, 新しい誕生日・住所を登録する操作や, 名前から誕生日や住所を探す操作などの仕様は, 記述される型などが異なるだけで, 制約式などは類似したものになるはずである. われわれの環境では仕様は解析されて, グラフ構造としてデータベースに格納される. 誕生日帳と住所録のグラフでは, 型や変数名を表す節点のラベルが異なるだけで, その他の構造は同じになると考えられる. グラフの構造を調べることにより, 類似した仕様を検索できる. 同様に蓄積された仕様に対して, 型や変数名を表す節点を抽象化したグラフを作り, そこから頻繁に使われるグラフの構造を抽出し, パターン化することもできると考えられる.

6 おわりに

本研究では再利用を目的とした仕様記述言語 Z の支援環境について述べた. 仕様を解析するために Z の構造をモデル化した. 解析された結果はグラフとしてデータベースに格納される. 仕様の類似度の判定などはグラフに対する操作として行なうことができる.

謝辞

本研究を進めるにあたり, 一年半御指導いただいた野呂昌満教授, 有益なアドバイスをいただいた蜂巣吉成先生, 大学院生の熊崎敦司さん, 兵藤淳二さん, 中田義也さん, 池内仁さん, 宗宮健仁さんに深く感謝致します.

参 考 文 献

[1] B. ボター,J. シンクレア,D. ティル: ソフトウェア仕様記述の先進技法 -Z 言語, トッパン,1993

[2] M・A・Hewitt: *PiZA:User Guide*,1997

[3] 辻野 嘉宏: 「コンパイラ」, 昭晃堂,1996

[4] 松尾 三郎: 「データベース」, 株式会社 SCC,1993

[5] http://www.csr.ncl.ac.uk/projects/FME/InfRes/tools/method3.html