ソフトウェア工学 126−15
（2000．3．10）

# ソフトウェア要求仕様の誤り修正支援手法

大西　淳, 杉本 英昭
立命館大学大学院 理工学研究科 総合理工学専攻
525-8577 滋賀県草津市野路東 1-1-1
e-mail: {ohnishi,sugi}@selab.cs.ritsumei.ac.jp

ソフトウェア要求仕様の妥当性検証支援手法を提案する。本研究での妥当性は、開発対象となるソフトウェアのユーザが記述するソフトウェアの妥当性記述と、開発者が記述するソフトウェア要求仕様を照らし合わせることによって検証する。ソフトウェア要求仕様は我々が開発してきた日本語要求言語によって記述されていると仮定しており、要求仕様は単文に分解され、文単位でデータベース化されている。妥当性の基準を要求仕様データベースに対する質問と見なし、質問によって検索可能かどうかによって、妥当性を検証することが出来る。本報告では、ソフトウェアの妥当性の基準の記述言語、それを用いたソフトウェア要求仕様の妥当性検証手法、および具体例を用いた手法の有用性について述べる。

# A Supporting Method of Correcting Erroneous Software Requirements Specifications

Atsushi OHNISHI, Hideaki SUGIMOTO

Department of Computer Science, Ritsumeikan University
1-1-1 Noji-Higashi,Kusatsu,Shiga 525-8577,Japan

We propose a validation method of the correctness of software requirements specifications. By verifying a validation description by users with a requirements specification, the correctness of the specification is validated. The authors have been developing a requirements model named Requirements Frame and requirements languages based on the model in order to improve the quality of SRSs. Since Requirements Frame can be transformed into relational data model, each of requirement sentences can be transformed into relational data model, each of requirement sentences can be regarded as a tuple of a relational table. We have been developing both a validation description language and a relational database management system for an SRS database as a validation system. One of the features of the SRS DB system is to show an answer with a requirement sentence as an example/counter-example.

## 1 はじめに

ソフトウェア要求仕様はソフトウェア開発においてさまざまな場面で参照される。そのため、ソフトウェア要求仕様には正確さが要求される。ソフトウェア要求仕様に誤りがあると、それを参照して作成される設計仕様、プログラム仕様などに影響を与え、結果としてソフトウェア開発に膨大な時間と労力を費やさなければならなくなり、非常に効率が悪い。このため、正確な要求仕様を作成する技法の確立が望まれる。我々は、ソフトウェア要求仕様に現れる誤りを検出・修正する手法を提案してきた [7] [3] 。

要求仕様にはユーザの意図したソフトウェア要求が完全に反映されていなければならないが、要求分析工程においてユーザと開発者の間でコミュニケーションがうまくとれていない場合はユーザの考えるソフトウェアとは違ったものが開発される。例えば、パーソナルコンピュータ等をWWW上で販売するソフトウェアを新規に開発する場合を考えてみる。新規に開発するソフトウェアを使用するユーザが開発を依頼し、開発者は依頼を受けてソフトウェア要求仕様を作成する。その工程において図1に示すように、ソフトウェアのユーザと開発者の間で代金の決裁方式について考えの相違がある場合が考えられる。このような考えの相違を要求分析段階で解消する手法の確立も必要である。

また、実際にソフトウェア開発の現場のＳＥの経験によると、「仕様獲得に費やす期間は対象業務によりさまざまであるが、全体の開発期間が1.5～2年程度の場合では、通常2～4ヶ月間を必要とする」 [8] とある。仕様獲得の段階でこれだけの期間を要するのであるから要求定義後の工程をも考慮すると、ある程度の規模のソフトウェア開発のライフサイクルにおいて時間的な経過により、ある時点では正しかった要求仕様も、別の時点では必ずしも正しいとは言い切れない。したがって、時間経過により要求仕様の妥当性が保証されなくなった場合の、その検証手法の確立も必要である。

以上の問題点を解決するため、本研究ではソフトウェアのユーザにあらかじめソフトウェアの妥当性の基準を定義してもらい、その妥当性の基準を要求仕様が満たしていることを検証する手法を提案する。


図 1: ユーザと開発者の考えの相違

## 2 要求言語

要求フレームモデル [3] に基づいた要求言語として

- 日本語要求言語X-JRDL (eXtended Japanese Requirements Description Language)
- ビジュアルな要求言語 VRDL (Visual Requirements Description Language)

を開発している。

X-JRDL は次の特徴を持つ。

1. 単文だけでなく重文や複文 (主語節、対立節、連体修飾節) が書ける。
2. 文脈から前方照応可能な名詞を省略したり、代名詞が利用できる。
3. １６種のファイルシステムに関連した動作概念とそれに対応する動詞を利用できる。さらに新規の動詞も、その動作概念の格構造を定義することによって利用できるようになる。

重文や複文のように1つの文の中に複数の動詞がある場合は、一つの動詞しか含まない単文に分割し、次に代名詞や省略された名詞を格フレームに基づいて補ってからCRD表現に変換する[3]。

VRDLは以下の特長を持つ[4]。

1. アイコンの形状と意味を定義できる。
2. 複合的なアイコンを容易に作成できる。
3. アイコンと矢印をエディタ上で配置することによって、フロー要求を記述する。
4. VRDL記述中のアイコンの動作を動作記述として与えることができ、それを解釈実行することによって、アイコンの動作をアニメーションとして表示できる。
5. VRDLによる記述を標準的なアイコンを用いた記述へ変換できる。

VRDLでは、名詞に対するアイコンの形状をビットマップデータとして定義しその意味を要求フレームに基づいて、**①名詞の型**、あるいは**②名詞の名前と型**のいずれかでもって定義する。

要求フレームにおいて動作概念を関係、必須格を属性とみなすことによって格フレームを関係スキーマと見ることができる。例えばデータフローを表すDFLOWの格フレームはDFLOWを関係、動作主格・源泉格・目標格・道具格といった必須格を属性とする関係スキーマと見ることができる。これにより「小渕さんがファックスを用いてBillさんへメッセージを送る」という要求文は表1のように関係表のタプルと見なされる。

表1: 要求文の関係データモデル表現

**関係：DFLOW**

| 動作主格 | 源泉格 | 目標格 | 道具格 |
|---|---|---|---|
| メッセージ | 小渕さん | Billさん | ファックス |

日本語要求文の解析結果を表1のように変換するのは容易であり、日本語要求言語やビジュアルな要求言語で記述された要求はそれぞれの概念に対応した関係スキーマのタプルの集まりとみなすことができる。換言すれば、日本語やビジュアルに記述された要求文は容易に関係データベースに格納できる。

## 3 要求仕様の妥当性確認

本研究で取り扱う「要求仕様の妥当性」とは、ソフトウェアのユーザが、「そのソフトウェアは本来こうあるべきである」と考えている要求のことである。開発者が記述する要求仕様とは必ずしも一致するものではない。

### 3.1 妥当性記述言語

妥当性記述言語は妥当性と用語の定義が記述できる。妥当性の記述ではある名詞で名前付けられた概念集合に含まれる要素にどういったものがあるかを質問したり、概念集合の要素すべてが特定の性質を満たすかどうか、概念集合の要素の少なくとも一つが特定の性質を満たすかどうかといった記述ができる。

以下に具体的な妥当性記述例を示す。

**・解析データが 入力される 機能は なにか？**
**・入力系 かつ システム処理機能である要素が 存在するはずである。**
**・すべての 入力プロセスは ファイルを 検索しなければならない。**

最初の妥当性記述は要求記述中で「解析データ」を入力とする機能型の名詞を問う質問形式の記述である。このような質問に対しては該当する名詞を列挙する。

2番目は集合「入力系」と「システム到達機能」の共通する要素が存在すべきであることを表明している。この場合も該当する名詞があればそれらを列挙する。

3番目は集合「入力プロセス」の要素すべてはファイルを検索する動作主格に当てはまっているかどうかを問うものである。この場合も該当する要求文、もしくは反例となる要求文を提示する。

これらの妥当性記述は利用者によって記述されることを想定している。このため要求仕様中で用いられている名詞や動詞がそのまま妥当性記述でも用いられるとは限らない。このため、妥当性記述で使われる概念と要求仕様で使われる概念との間の関係を定義をすることによって対応付けられるようにする。これを「用語の定義」と呼んでいる。

用語の定義では動作概念の集合、名詞の集合、名詞の型の集合を要素の性質の記述、もしくは要素の列挙によって定義することができる。さらに定義済みの集合に対して演算子『かつ』(∩),『または』(∪),『～の否定』(¬)を適用できる。

また効率良く定義するために、動作概念を表す動詞以外に『到達可能』、『祖先』、『子孫』という予約語が用意されている。機能Aから機能Bへデータフローが存在するときAからBへ『到達可能』とする。また『祖先』と『子孫』はデータ構造と機能構造に対して、親より上すべてを『祖先』として、また子供より下すべてを『子孫』として表すために用いられる。

以下に定義例を示す。

・入力コマンドと 解析データを 受動オブジェクトと 呼ぶ。
・受動オブジェクトを 受け取る 機能を 入力プロセスと する。
・ユーザと インタフェース部を 外部オブジェクトと 呼ぶ。
・外部オブジェクトの 否定を 内部オブジェクトと 呼ぶ。
・計算部から 到達可能な機能を システム到達機能と 呼ぶ。

定義文に表れる名詞や動詞は要求仕様に用いられたものと、集合定義文によって定義済みのものに限定し、妥当性記述で現れた名詞や動詞との間の関連を定義する。妥当性記述を読んで理解した要求仕様の記述者によって集合の定義は記述される。

最初の定義文では「入力コマンド」と「解析データ」という2つの要素からなる集合「受動オブジェクト」という用語が定義される。2番目の定義文では1番目で定義した「受動オブジェクト」を入力とする機能型の名詞を要素とする集合「入力プロセス」が定義される。3番目でも2つの要素からなる集合が定義される。4番目の定義文では3番目で定義した集合の否定が定義される。否定の場合は元の集合の要素の名詞の型を持つ名詞の集合との差がとられる。この場合では要求記述中の「ユーザ」以外の人間型の名詞と「インタフェース部」以外の機能型の名詞を要素とする集合を「内部オブジェクト」として定義したことになる。5番目の定義文では「計算部」からの出力が直接もしくは一つ以上の機能を介して流れ込む機能の集合が「システム到達機能」として定義される。このように要素を列挙するか、集合の性質を示すことによって用語を定義する。

### 3.2 妥当性確認のシステム化

妥当性記述を要求仕様データベースに対する質問と見なし、要求仕様データベースを検索し、その検索結果によって妥当性を確認する。このため妥当性記述の処理と要求仕様データベースの検索システムRETを開発している[5]。

要求仕様の精製とソフトウェア設計時や保守時の要求仕様の検索と修正を支援するために、CRD内部表現に対する検索系RETをWS上でKCL (Kyoto Common Lisp)を用いて開発した。RETの特長を以下に示す。

- 質問に対して要求文を具体的な例証もしくは反例として示す。
- 要求フレームで定められた枠組みを意味制約としている。

最初の特長は要求仕様の正しさの確認に有効である。利用者は要求仕様中の名詞や動詞の集合に対して、それが満たすべき・満たすべきでない性質を質問できる。もし要求仕様が誤っているならば、具体的な例証もしくは反例が得られるので誤っている箇所が判明できる。

2番目の特長は効率的な検索に有効である。格フレームはある動作概念の持つ特定の格にはどういう型を持った名詞が当てはまるかを規定するものであるが、質問文で誤った型の名詞が表れたならば、検索することなしに答を返すようにしている。

## 4 妥当性確認例

要求仕様の解析と妥当性確認例を示す。

### 4.1 要求仕様の解析

X-JRDLによって書かれた在庫管理システム要求の一部を示す。

在庫管理システムは 入庫管理部、出庫管理部、在庫管理部から構成される。

1. 入庫管理部
   在庫管理係から制御を 受け取り、彼からキーボードを通して品名と数量からなる入庫票を受け取る。そして、それを用いて在庫マスタファイルを更新し、在庫管理係に制御を 返す。
2. 出庫管理部
   検索部と 出庫部と 発注部から なる。

最初に要求仕様の解析系は辞書に未登録の単語の意味を問い合わせる。具体的には名詞はその型を、動詞・形容詞はその概念(と格構造)を問い合わせる。次に複文・重文を概念が一つしか含まない単文に分割する。その過程を以下に示す。なお明朝体は入力された要求記述を単文に分割したものを、ゴシック体は解析系からのメッセージを、イタリック体は利用者の応答を示す。

この過程で省略された格に相当する名詞や代名詞は具体的な名詞に置き変わる。もし置き換える名詞の候補が複数存在する場合は記述者に問い合わせる。最後に必須格がすべて揃った単文を提示する。各単文は格構造をそのまま反映しており内部表現に変換される。内部表現は2章に示したように関係データモデルに変換できる。

表 2: 要求仕様の関係データモデル表現

関係： **ANDSUB**

| 動作主格 | 目的格1 | 目的格2 | 目的格3 |
|---|---|---|---|
| 在庫管理システム | 入庫管理部 | 出庫管理部 | 在庫管理部 |
| 入庫票 | 品名 | 数量 | − |
| 出庫管理部 | 検索部 | 出庫部 | 発注部 |

関係： **CFLOW**

| 動作主格 | 源泉格 | 目標格 |
|---|---|---|
| 制御 | 在庫管理係 | 入庫管理部 |
| 制御 | 入庫管理部 | 在庫管理係 |

関係： **DFLOW**

| 動作主格 | 源泉格 | 目標格 | 道具格 |
|---|---|---|---|
| 入庫票 | 在庫管理係 | 入庫管理部 | キーボード |

関係： **UPDATE**

| 動作主格 | 源泉格 | 目標格 |
|---|---|---|
| 入庫管理部 | 品名 | 在庫マスタファイル |

結果を表2に示す。表でANDSUBは機能やデータのand木構造を、CFLOWは制御の流れを、UPDATEはファイルの更新を、それぞれ表す動作概念である。ANDSUBは2つ以上の目的格を子として持てば良く、「在庫管理システム」と「出庫管理部」は3つのサブシステムを持つが、「入庫票」は2つのデータより構成されるため、目的格3は空欄となっている。

このように単一の動詞しかもたない単文に分割された各要求文は、その動詞の動作概念を関係とする関係表のタプルとみなされる。

これから単文に分解します。
在庫管理システムは 入庫管理部と 出庫管理部と 在庫管理部から 構成される。

見出し語 : 入庫管理部
在庫管理係から 制御を 受け取る。
  "入庫管理部" が "受け取る" のですね。
入庫票は 品名と 数量から なる。
彼から 入庫票を キーボードで 受け取る。
  "彼" の指す名詞は "在庫管理係" ですね。
  " 入庫管理部" が "受け取る" のですね。
在庫マスタファイルを それで 更新する。
  "入庫管理部" が "更新する" のですね。
  " それ" の指す名詞は何ですか。
    RET : "入庫票"  1 : "数量"  2 : "品名"
    ==> *2*
在庫管理係へ 制御を 返す。
  "入庫管理部" が "返す" のですね。

見出し語 : 出庫管理部:
検索部と 出庫部と 発注部から なる。
  "出庫管理部" が "なる" のですね。

図 2: 要求仕様の解析プロセス

### 4.2 要求仕様の妥当性確認

ここでは妥当性の記述と検索による妥当性確認を示す。なお質問を明朝体で検索系からの検索結果をゴシック体で示す。妥当性確認では妥当性の記述の前に用語の定義があるものとしている。

在庫管理システムの子孫を子システムと呼ぶ。

**Q1:** 在庫管理係から到達可能な機能はなにか?
**A1:** 入庫管理部 です。

**Q2:** すべての子システムの要素は受け取るか?
**A2:** いいえ。出庫管理部、在庫管理部、検索部、出庫部、発注部は条件を満たし ません。

**Q3:** 少なくとも 1 つの子システムの要素は受け取るか?
**A3:** はい。子システムの要素のうち、入庫管理部は以下のように条件を満たします。「入庫管理部は 在庫管理係から 入庫票を キーボードで 受け取る。」

**Q4:** 子システムの要素を動作主格とする ANDSUB 文はあるか?
**A4:** はい。子システムの要素のうち、出庫管理部は以下のように条件を満たしま
す。「出庫管理部は 検索部と 出庫部と 発注部から なる。」

最初の文は、「在庫管理システム」のサブシステムである「入庫管理部」、「出庫管理部」、「在庫管理部」、「検索部」、「出庫部」、「発注部」を要素とする集合を「子システム」という名前で定義している。

2 番目の文 Q1 は「在庫管理係」からデータフローによって到達する機能についての質問であり、該当する「入庫管理部」が検索結果として答えられている。

Q2 では「子システム」の要素がデータもしくは制御を受け取るかという質問に対して、受け取らない要素（つまり DFLOW, CFLOW の目標格に該当しない要素）が示されている。質問 Q3 では「子システム」の要素でデータもしくは制御を受け取る文を例示している。Q4 でも質問を満たす要求文が具体的に示されている。

このように利用者は必要とする要求文を検索できるだけでなく、要求記述に誤りがある場合でも、適切な質問を用意し、その結果例証や反例となって提示される要求文を吟味することによって要求記述の正当性を検証できる。

## 5 本手法の適用範囲

本研究で提案した手法はソフトウェア要求定義の様々な工程で適用することが出来る。以下、その例を述べる。

1. ユーザと開発者の間のコミュニケーション支援

   妥当性の基準の記述者（ユーザ）と要求仕様の記述者（開発者）との間でうまくコミュニケーションが取れていないと正確な要求仕様は作成できない。本研究で提案した手法を用いることにより、開発者とユーザの間のコミュニケーションが円滑に進むように支援することができ、正確な要求仕様作成につながる。

2. 時間経過により、要求仕様または妥当性の基準の変更があった場合の、要求仕様の妥当性検証

   一人の要求仕様記述者が作成した要求仕様であっても、時間経過とともに要求仕様が変更になる場合がある。そのような場合に、あらかじめユーザが定義した妥当性の基準に、変更後の要求仕様が合致しているかどうか検証することができる。また、要求仕様に変更がなくても妥当性の基準が時間経過とともに変更される場合がある。そのような場合も本研究で提案した手法で要求仕

様が妥当であるか否か検証することができる。

3. 立場の異なるユーザの合意形成支援

同じソフトウェアであっても、そのソフトウェアを実際に利用するユーザと、そのソフトウェアに対する予算決定権を持っている人とでは、妥当性の基準が違う場合がある。そのような場合にそれぞれが記述した妥当性の基準を用いることにより、要求分析における合意形成の支援をすることができる。

## 6 おわりに

本報告では、ソフトウェアの妥当性の基準をユーザにあらかじめ記述してもらい、ソフトウェア開発者の記述する要求仕様の妥当性を検証する手法を提案した。本手法はソフトウェア要求の仕様化工程だけでなく、要求分析工程での立場の異なるユーザ間の合意形成支援にも適用できる他、時間経過によってソフトウェアの妥当性の基準が変化した場合の、要求仕様の妥当性検証にも利用できるなど、ソフトウェア要求定義における様々な工程に適用できる。

今後の課題としては以下の点が挙げられる。本手法を適用した結果、ソフトウェア要求仕様が変更された場合、その変更が同一仕様内の他の箇所にも影響を及ぼすことが考えられる。そのような箇所を特定し、修正を支援する手法の確立が必要である。また、本手法をシステム化し様々な要求仕様に適用するとともに妥当性の基準の記述についても様々な例を適用して、本手法の有用性を確認することが必要である。

## 参考文献


[1] Davis, A.M.: *Software Requirements, Objects, Functions, & States,* Prentice-Hall, Inc., 1993.

[2] IEEE Standards Board: IEEE Recommended Practice for Software Requirements Specifications, IEEE, 1993.

[3] 大西 淳, 阿草清滋, 大野 豊: 要求フレームに基づいたソフトウェア要求仕様化技法, 情報処理学会論文誌, Vol.31, No2, pp.175-181 (1990).

[4] 大西 淳：「ビジュアルなソフトウェア要求仕様化技法」情報処理学会論文誌 36 巻 5 号, 1995, pp.1183-1191.

[5] Atsushi Ohnishi: Software Requirements Specification Database Based on Requirements Frame Model, Proceedings of Second International Conference on Requirements Engineering *(ICRE'96)*, pp.221-228 (1996).

[6] 杉本 英昭, 大西 淳: 矛盾を含んだ要求仕様の解釈手法, ソフトウェア工学の基礎 V, 近代科学社, pp.106-115 (1998).

[7] Hideaki Sugimoto, Atsushi Ohnishi: A Detecting and Interpreting Method of the Inconsistency of Software Requirements Specifications, Proceedings of 6th Asia Pacific Software Engineering Conference *(APSEC'99)*, pp.208-215 (1999).

[8] 鈴木 健蔵, 腰原 貞利, 木津谷 欣三: ＳＥの仕様獲得の実際 - システム分析工程における作業手順と問題点 -, 情報処理学会誌, Vol.33, NO.6, pp.612-616 (1992).

[9] Thayer, R., Dorfman, M.: "System and Software Requirements Engineering," IEEE Computer Society Press Tutorial, 1990.

[10] Ullman, J.D.: *Principles of Database Systems* 2nd ed., Computer Science Press, Inc. 1982.