ソフトウェア工学 90－8
（１９９３．２．４）

# ビジュアルなソフトウェア要求仕様化技法

大西 淳

京都大学大型計算機センター

記述者のイメージ通りにソフトウェア要求を仕様化できるように、記述者にビットマップエディタなどを用いてアイコンの形状を定義させ、さらにそのアイコンの意味として具体的な名詞、名詞の型、または動作概念の一つを定義させる手法を提案する。手法に基づいて開発中のビジュアルな要求言語 VRDL とその処理系について紹介する。VRDL は筆者が提案した要求フレームモデルに基づいており、その内部表現は、既開発の日本語要求言語の内部表現と同じ形式をとる。これにより、日本語とアイコン言語のどちらでも要求を記述できる。また既開発の要求定義環境 Card で用意している各種のツールが適用できるため、ビジュアルな要求記述の品質を向上できる。

# A Visual Software Requirements Specification Method

**Atsushi OHNISHI**

**Data Processing Center, Kyoto University**
**Kyoto 606-01, Japan**

The author proposes a visual software requirements specification method with which a describer can define both shape and semantics of an icon to specify his requirements just what he imagines. Visual requirements language named VRDL and its analyzer are illustrated. Since VRDL is based on the Requirements Frame model, its internal representation has the same scheme of internal code of the Japanese base requirements language X-JRDL. A Describer can write down requirements with either/both VRDL or/and X-JRDL. The requirements definition environment named CARD contributes to make visual requirements better.

## 1 はじめに

日本語によるテキストと図を併用しながら要求を定義することによって、効率良く要求を仕様化できる。構造化分析技法 [2] の DFD (Data Flow 図) や SADT[5] など仕様化のための図的表現やその支援ツールは数多く提案され開発されている [14]。特に DFD の処理系は上流 CASE ツールとして各社から製品が出されている。

しかし、これらの手法では用いられる図形の形状とその意味があらかじめ定まっており、要求定義者の頭に描いたイメージをそのまま図として表すことは出来ない。

ここでは、要求に現れる実体を任意の形状のアイコンとして定義し、実体間の関連を矢印と動作を表すアイコンによって定め、それらをエディタ上で配置していくことによって、要求を定義する手法を提案する。これにより要求定義者のイメージを反映させることができる。さらに、アイコンの意味を要求フレームモデル [7] に基づいて定義させることにより、要求仕様の意味を明確にできる。

また、記述は解析系により、既開発の日本語要求言語による記述の内部表現と同じ表現に落されるため、日本語要求言語と併用して、テキストとして書きたい部分は日本語で、図として書きたい部分はビジュアルな言語でと、切り分けて仕様化できる。

本稿では、最初に要求フレームモデルについて説明する。次に提案する手法に基づいたビジュアルな要求言語 VRDL とその処理系について紹介する。また、要求定義環境 CARD における位置付けについても言及する。

## 2 要求フレームモデル

要求フレームモデル [7] は要求記述の枠組を与えるものであり、

① 名詞と名詞の型

② 動詞と概念

③ 文と格フレーム

表 1: 概念と動詞

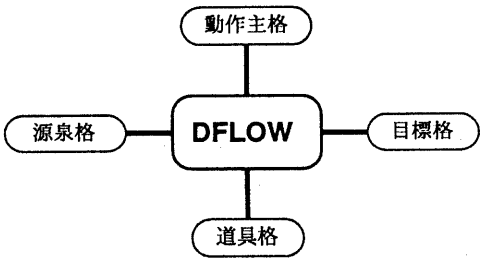
| 概念 | 対応する動詞 |
|---|---|
| data の流れ | 入れる、受け取る、等 |
| 制御の流れ | 渡す、受け取る、移す、等 |
| function, data, file の and 木構造 | 構成する、成る、含む、分割される、含まれる、等 |
| function の or 木構造か data 種別 | 構成する、成る、含む、分割される、含まれる、等 |
| file 中の data 検索 | 検索する、検索される |
| file への data 挿入 | 挿入する、加える、等 |
| file 中の data 更新 | 更新する、更新される |
| file 中の data 削除 | 削除する、除く、等 |
| file 処理 | ソートする、アペンドする、マージする、複写する、等 |
| data 生成 | 作成する、変換する、等 |

④ 文章と機能フレーム

のそれぞれについての対応関係を定める。

名詞は human, function, data, file, control, device の 6 種の型のいずれかを持つ。動詞はデータの流れ、ファイルや機能の構造、ファイルの操作など 10 種の動作に関する概念のいずれかに分類される。「入力する」、「(データを) 渡す」、「出力される」といった動詞はすべてデータの流れに相当する DFLOW という概念に分類される。表 1 に概念と対応する動詞を示す。名詞の型と動作概念については要求定義の対象をファイル処理分野に限定して用意している。なお、動詞については後述するように、記述者が新たに概念を定義し、それに対応した新規の動詞も利用できる。

文は一つの動詞とその動詞に対応する動作概念の必須格に当てはまる名詞から構成される。格フレームは動作概念と必須格に対する枠組である。各概念はそれぞれ異なった格構造を持っている。例えば DFLOW の場合は図 1のように動作主、源泉、目標、道具という格を持っている。さらに、それぞれの格に当てはまる名詞の型も、それぞれ限定される。DFLOW の場合は動作主格は流れるデータが相当し、当てはまる名詞も data 型とな




| 概念 | 格 | 名詞の型 |
|---|---|---|
| DFLOW | 動作主格 | data |
| | 源泉格 | functionかhuman |
| | 目標格 | functionかhuman |
| | 道具格 | device |

図 1: 概念 DFLOW(データの流れ) の格フレーム

る。源泉・目標格はそれぞれ流れるデータを送り出す・受けとる主体が相当するため、 human または function 型の名詞が当てはまる。道具格は device 型の名詞となる。 10 種の動作概念に該当しないような動作を記述において表したい場合は、記述者が新たにその概念と対応する格フレームを定義することによって利用できる [10]。

この格フレームによって、格の抜けや不正な型の名詞の使用を検出できる。さらに、代名詞が使われたり、格が省略された場合に、文脈から用いられるべき名詞を推定できる [10]。例えば、「利用者は検索コマンドを入力する」という要求文は「入力する」の対象である「検索コマンド」が data 型の名詞であることから、動作概念が DFLOW であるとみなされ、助詞をもとに DFLOW の格フレームに基づいて判断した結果、表 2のように解析される。この例文では、目標格 (どこに対して) と道具格 (何を用いて) が抜けていることが判明する。 DFLOW の格フレームから目標格は function か human 型の名詞、道具格は device 型の名詞が埋まるべきであり、前述の要求文や要求文の置かれた段落の見出し語などから該当

表 2: 要求文の解析結果

要求文「**利用者は検索コマンドを入力する**」

| 動作概念 | DFLOW |
|---|---|
| 動作主格 | 検索コマンド |
| 源泉格 | 利用者 |
| 目標格 | ** 不明 ** |
| 道具格 | ** 不明 ** |

する型の名詞を埋まるべき名詞の候補として記述者に問い合わせるが、埋まらない場合は抜けと判断する [7, 10]。

このように要求文は動作概念とその格フレームに基づいた内部表現 CRD (Conceptual Requirements Description) に変換される。重文や複文 (主語節、対立節、連体修飾節) 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て解析している [10]。

機能フレームはシステムが備えるべき一般的な性質を規定するものであり、「外部からの入力と外部への出力は、それぞれ少なくとも一つ存在する」、「作成されたり検索されて得られたデータは一度は参照されなければならない」といった 10 個の性質について、要求記述が満たしているかどうかをチェックするために用いられる [7]。機能フレームによって機能単位の抜けや矛盾を検出できる。

## 3 ビジュアルな要求言語: VRDL

要求フレームモデルに基づいて、ビジュアルな要求言語 VRDL (Visual Requirements Description Language) を設計した [11]。 VRDL の特徴を以下に示す。

1. アイコンを記述者が定義する
2. 定義したアイコンをエディタ上で配置していくことによって要求を定義する
3. 記述された要求は CRD 表現に変換される

4. VRDL による記述から CRD 記述を介して標準的な文書へ変換できる [9]

5. VRDL 記述に用いられるアイコンの動作を与えることによって、記述を実行する

6. VRDL による記述から変換された CRD 表現に対して、要求定義環境 CARD のツール群による検証・検索・フロー表示・設計支援ができる [6, 12]

特徴 1 ～ 5 について以下に説明する。6 については 5 章で述べる。

### 3.1 アイコンの定義

DFD に基づくような一般の要求定義用のビジュアルな言語では、利用可能なアイコンの形状も意味もあらかじめ定まっている。DFD はデータの流れを名前つきの矢印で、機能を円で、ファイルを直線で、データの源泉と吸収を四角形で表し、アイコンの種類が少ないので覚えやすい。しかしながら、能大式の業務フロー図 [3] のように 30 以上の多種のアイコンを使う図に慣れた人にとっては DFD は単純化しすぎて使いにくく、アイコンの種類が少ないので名前や説明を詳細に文章などで記述しなければならない。また、使えるアイコンが限られるために、例えばファイルを直線でなく JIS の情報処理用流れ図記号の直接アクセス記号 [4] で表現したくても出来ない。

記述者にとっては、自分のイメージにあった記号をそのまま要求記述に用いることができるならば、要求が記述しやすいし、また理解しやすい。このためには自分でアイコンの形状を定義して要求記述に用いることができるようにすればよい。一方、要求記述は記述者以外にも設計者など開発に携わる人によって参照される。他人の描いた図を理解するには、そこで使用されたアイコンの意味を的確に把握する必要がある。記述者以外の人にとってアイコンが別の意味にとられると正しく要求を理解されなくなる。

このため VRDL ではアイコンの形状と意味の両方を記述者自身で定義できるようにしている。

表 3: 定義可能なアイコンのタイプとその意味

| タイプ | アイコンの意味 |
|---|---|
| 名詞 | 具体的な名詞の名称とその型 |
| 名詞の型 | 具体的な名詞の型 (例: data) |
| 動作概念 | 既存の具体的な概念 (例: DFLOW) |

表 4: 具体的なアイコンの定義例

| アイコン | 意味 |
|---|---|
|  | human 型 |
|  | 「Fax」 device 型 |
|  | 「文書」 data 型 |

定義可能な 3 つのアイコンのタイプとその意味を表 3に示す。

このように要求フレームモデルに対応して、要求記述に現れる名詞や動詞に対する動作概念をアイコンとして定義できる。表 4にアイコンの形状と意味の定義の具体例を示す。

表 4 で最初のアイコンは human 型を持った名詞の総称を表し、2 番目は「Fax」という device 型の名詞を表し、3 番目は「文書」という data 型の名詞を表す。名詞の総称を表すアイコンには具体的な名前を与えることができる。上の例では人の頭部の輪郭をしたアイコンに「オペレータ」という名称や「顧客」という名称を与えることによって、それぞれ human 型の具体的な名詞を表すことができる。

### 3.2 アイコンによる記述と CRD への変換

定義したアイコンを配置していくことによって、要求を定義する。名詞を表すアイコンと別の名詞を表すアイコンとの間に矢印を引き、矢印に動作概念を表すアイコンをラベルとして貼り付けることによって、その動作概念を意味させる。動作概念を表すアイコンが与えられない場合は、矢印はデータの流れを表すものとしている。

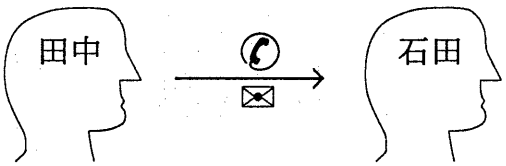



図 2: 定義したアイコンによる記述例

図 2に記述例を示す。この図は「田中さん (という人間型の実体) から石田さん (という人間型の実体) に Fax (という装置) で文書 (というデータ型の実体) を送る」という文を意味する。このように必須格に相当する名詞アイコンを矢印で表される動詞の回りに配置することによって文を定義できる。この文は格フレームに基づいており、容易に CRD 表現に変換することができる。さらに次々とアイコンや矢印を配置していくことによって、複数の文に相当する記述を表すことができる。

### 3.3 文書の標準化支援

図 2を記述者以外の人が読む場合、 という アイコンを「Fax」ではなく「電話」と誤って解釈するかもしれない。このように同じ意味内容に対して異なる記号の表現が与えられる問題や、同じ記号の表現が人によって異なる意味を持つ問題は、名詞とそれに対するアイコンを読者が定義することによって解決できる。例えば表 5のように記述者と読者によって、同じ名詞が異なるアイコンで表される場合、記述者の定義したアイコン表現を読者の定義したアイコン表現に置換することによって読者にとっても正しく解釈される [10].

しかしながら、記述で使用される名詞や動詞に対応するアイコンをすべての読者一人一人に定義してもらうのは困難であると予想されるため、記述で使用される名詞や動詞に対応する標準化されたアイコンをあらかじめ用意しておき、それらを用いた記述に変換する。これにより、記述者が自分で定義したアイコンを用いても、標準化されるため読者は誤解なく理解できる。具体的には VRDL を用いたデータフロー記述から DFD 図を導くことを考えている。

表 5: 記述者と読者のアイコンの定義

| 意味 | 記述者 | 読者 |
|---|---|---|
| 「Fax」 device 型 | | |
| 「電話」 device 型 | | |

### 3.4 ビジュアルな要求記述の実行

要求記述を解釈実行することによって、利用者に記述が正しく書かれているかどうかを確認できる。ここでの実行とは具体的には VRDL による記述からデータフローに関する記述を抽出し、さらにそこで用いられている動作可能なアイコンに対して、動作記述を利用者が与えることによって、データフローの様子をアニメーションとして表示させることを指す。

アイコンの動作は、

- アイコンの移動
- 少しずつ異なる複数のアイコンの表示の切替

によって実現する。例えば「データが渡される」という動作を、源泉格から目標格のアイコンに向かって、データを表すアイコンを座標を少しずつずらしながら表示したり消去したりすることを繰り返すことによって実現する。また、「電話が鳴る」という動作を、受話器が少し持ち上がったアイコンと平常の電話のアイコンとを切替えて表示することによって実現する。さらに、アイコンの移動と異なる複数のアイコンの表示の切替を組み合わせることによって、複雑な動作を実現する。

VRDL による記述において、 DFLOW の源泉格と目標格は動作しないと仮定し、データに相当する動作主格と道具格に対してそれらの動作記述を与える。動作でのアイコンは、 VRDL 記述で用いたアイコンと動作用に新たに定義したアイコンが利用できる。動作記述では、どのアイコンを、どの位置に、どの程度の時間間隔で、表示・消去するかが記述される。この動作記述を解釈実行す

ることにより、DFLOW文の源泉格から目標格へのデータの流れが一文ずつ逐次アニメーションとして表示される。

このために、アイコンの指定、アイコンの表示・反転・消去、表示座標設定、時間幅設定といった動作記述のための言語とその処理系を開発中である。

## 4 ビジュアルな要求言語の処理系

現状でのビジュアルな要求言語の処理系は

- アイコンの定義
- アイコンと矢印の配置
- VRDL記述の解析

に分けられる。

### 4.1 アイコンの定義処理

アイコンの形状については、bitmapコマンドやDrawing toolsといった既存のツールを用いて描いたり、スキャナーなどの装置を用いて既存の表現を読み込み、bitmapデータとして定義する。現在のところアイコンの縦横の大きさは一定となるようにしている。アイコンは表3のように3種類あるが、それぞれのアイコンデータの格納されたファイルとその意味をアイコン辞書に定義している。現在のところアイコン辞書への登録は機械化されていない。また、VRDL記述の実行においても、元のアイコンと少し異なったアイコンを定義する場合があるが、その際も既存のツール等で対処する。

### 4.2 アイコンと矢印の配置処理

次に定義したアイコンをエディタ上で配置する。エディタを起動すると、左側に定義した名詞アイコンが型ごとに分類されて表示される。上部に動作概念アイコンが表示される(現在のところこの機能はサポートされていない)。

最初に動作概念をマウスで選択すると、その動作概念の格構造に基づき、必須格に相当する名詞

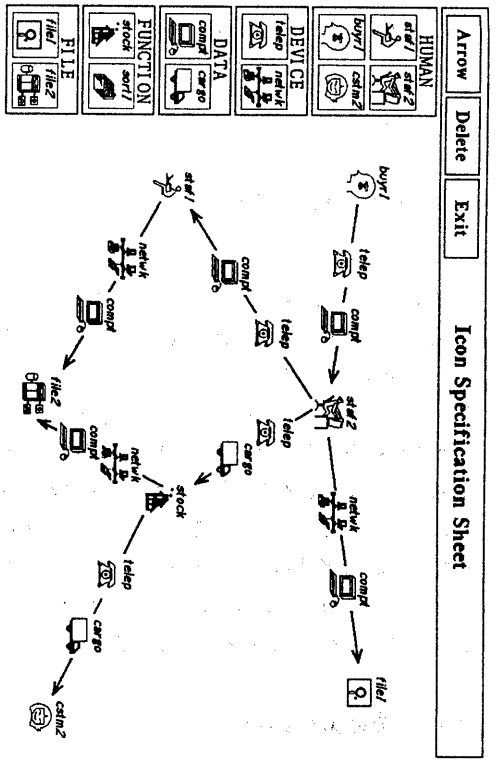



図3: VRDLによる記述例

アイコンを次々と選択するように促され、マウスでアイコンとその配置位置を指定していく。ただし、現在は動作概念としてDFLOWだけをサポートしており、動作概念の選択機能はない。

図3にエディタの画面のハードコピーを示す。図1に示したように、DFLOWには4つの必須格があるが、それぞれに対応する名詞アイコンとその表示位置を順次指定することによって1つのDFLOW文に相当する図が描かれる。DFLOWの源泉格と目標格の配置位置から矢印が自動的に引かれるようにしている。この操作を繰り返すことによって複数の要求文に相当する図が描かれる。図3で左上のDFLOW文は「顧客からアナリストに画面設計情報を電話で伝える」という日本語に対応している。図では7つのDFLOW文に相当する記述が描かれている。

このように記述者が自分で定義したアイコンを

自分で指定した位置に配置しながら、頭の中のイメージに近い形で要求を記述できる。現在のところ、問題点として以下のものがあげられる。

- アイコンの配置位置を指定する際に、過去に配置したアイコンに新たに配置しようとするアイコンが重なる場合や、他のDFLOW文と接近しすぎると、配置したアイコンが2つのDFLOW文のどちらの格に相当するかが(人間にとっては)あいまいになる場合が生じており、指定した位置情報から重なりやあいまいさが生じる場合を排除して配置する工夫が必要である
- アイコンの大きさを常に同じ大きさとしているが、視認性の向上のためには、場合に応じて大きさを変化させる工夫も必要である
- 構造的な記述をサポートしていないので、複雑な図が記述できないが、DFDのように構造的な記述もサポートする
- すべてのアイコンを利用者に定義させるのではなく、ファイルやディスプレイの形状のアイコンなど、標準的なアイコンをシステム側であらかじめ用意する

### 4.3 VRDL記述の解析

エディタによる入力が終ると、次に解析を始める。VRDL記述は1つのDFLOW文を単位として内部表現CRDに変換される。動作概念に対応する必須格はすべて入力時に指定するようになっており、代名詞の照応や格の抜けの問題はない。現在はエディタと解析部のプロトタイプをUnixワークステーション上でC言語とX Windowによって実現している。

## 5 要求定義環境: CARD

要求定義のための環境としてCARD (Computer Aided Requirements Definition)を開発している[6, 12]。CARDの構成を図4に示す。

CARDは

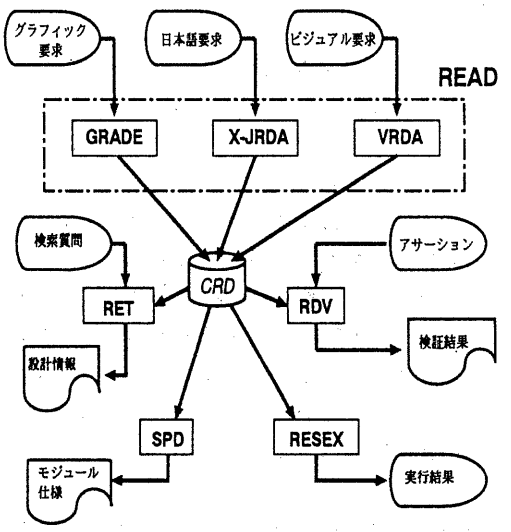



図4: 要求定義環境: CARD

- 要求記述解析系: READ
  - 日本語要求言語解析系: X-JRDA
  - 図形要求言語解析系: GRADE
  - ビジュアル要求言語解析系: VRDA
- 要求記述精製系
  - 要求記述検証系: RDV
  - 要求記述検索系: RET
  - 要求記述実行系: RESEX
- 設計支援系: SPD

から構成されている。要求は日本語・アイコン言語・グラフィック言語のどれかにより記述される。日本語で書きたい部分とビジュアルに書きたい部分を切り分けて、より書きやすい言語を用いて記述できる。どの言語による要求記述も、解析の結果としてCRD表現に変換されて、要求記述精製系や設計支援系で利用できる。要求記述精製系は、主に記述の正しさ(correctness)の向上を目的としている。設計支援系は、記述からのモジュール設計を支援する[8]。

## 6 おわりに

ビジュアルな要求言語 VRDL とその処理について紹介した。日本語要求言語と組み合わせることによって、要求記述の書きやすさや読みやすさは向上すると思われる。

VRDL の処理系のプロトタイプの使用によって判明した①アイコンや矢印の配置問題、②アイコンの大きさの変化、③構造化された記述のサポート、④標準アイコンのサポートといった問題点の解決が今後の課題である。また、開発中である VRDL による記述の実行系の、CASE 環境への統合を予定している。

**謝 辞** 処理系の開発に当たり、本学大学院工学研究科応用システム科学専攻修士2回生の程 国政君と島田淳一君に感謝する。また討論に参加し、有益なコメントを戴いた大型計算機センターの教官各位に感謝する。

## 参考文献

[1] 阿草清滋、大西 淳、久保哲也、西山 聡、飯村次郎：「上流工程支援ツール VIP の試作」、日本ソフトウェア科学会第6回大会 C3-3, 1989 (pp.177-180).

[2] Tom DeMarco: "Structured Analysis and System Specification," Prentice-Hall, 1979.

[3] 情報処理学会編：「情報処理ハンドブック」、オーム社, 1980 (pp.245)

[4] 日本規格協会編：「JIS ハンドブック情報処理 ソフトウェア編」 JIS X 0121(1986), 1992.

[5] D. A. Marca, C. L. McGowan: "Structured Analysis and Design Technique." McGraw-Hill Book Co., New York, 1988.

[6] 大西 淳、阿草清滋、大野 豊：「ソフトウェア要求定義支援技法 / 環境:Card」、情報処理学会「CASE 環境シンポジウム」論文集, 1989 (pp.49-56).

[7] 大西 淳、阿草清滋、大野 豊：「要求フレームに基づいた要求仕様化技法」、情報処理学会論文誌 31 巻 2 号, 1990 (pp.175-181).

[8] 大西 淳：「要求フレームに基づいた日本語要求仕様の設計支援」、日本ソフトウェア科学会第 8 回大会 C6-6, 1991 (pp.533-536).

[9] 大西 淳：「コミュニケーションモデルに基づくソフトウェア開発支援」、電子情報通信学会技術研究報告 KBSE92-17, 1992 (pp.25-32).

[10] 大西 淳：「要求定義のためのコミュニケーションモデル」、情報処理学会論文誌 33 巻 8 号, 1992 (pp.1064-1071).

[11] 大西 淳：「ビジュアルな要求言語」、情報処理学会第 45 回全国大会 6U-4, 1992, 第 5 分冊 (pp.379-380).

[12] A. Ohnishi, K. Agusa: "CARD: A Software Requirements Definition Environment," IEEE Proc. Int. Symp. Requirements Engineering, 1993 (to appear).

[13] N. C. Shu: "Visual Programming," Van Nostrand Reinhold Co. New York, 1988.

[14] R. H. Thayer, M. Dorfman: "System and Software Requiremenst Engineering," IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.