ソフトウエア工学 93－5
（1993．7．8）

ビジュアルなソフトウェア要求定義支援

大西 淳

京都大学大型計算機センター

ビジュアルなソフトウエア要求記述に対して、アイコンの動作を記述者にシナリオとして与えさせ、そのシナリオを解釈することによってソフトウエア要求の動的な振舞いをアニメーションにより表示する手法を提案する。手法に基づいて開発中のシナリオ記述言語とその解釈系について紹介する。ビジュアルなソフトウエア要求は筆者らが開発した要求言語 VRDL によって記述される。この言語はアイコンの形状と意味を記述者が定義でき、静的なデータや制御の流れに対して記述者の持つイメージに忠実に要求を表すことができるという特長を持つ。アニメーション表示によって、動的な振舞いを検証できるため、ビジュアルな要求記述の品質を向上できる。

Supporting method of Visual Software Requirements Specification

Atsushi OHNISHI

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

The author proposes an execution method of a visual software requirements specification (SRS). The method provides the requirements describer to specify icons' movements as an scenario. By interpreting the scenario, dynamic behaviors of the SRS are expressed as an animation. Visual SRSs are written with a requirements language named VRDL. With VRDL a describer can define both shape and semantics of an icon to specify static data/control flow requirements just what he imagines. The execution method contributes to improve visual SRSs because the decriber can check dynamic behaviors of the SRSs.

1 はじめに

データや制御の流れに関するソフトウエア要求を、要求定義者の頭に描いたイメージにできるだけ忠実に表すことを目標に、要求に現れる実体を任意の形状のアイコンとして定義させ、実体間の関連を矢印で表し、それらをエディタ上で配置させる。さらに、アイコンの意味を要求フレームモデル [6] に基づいて定義させることにより、ビジュアルに記述された要求仕様の意味を明確にする手法とそのための要求言語 VRDL を開発してきた [8, 10]。

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させることができる。VRDL によるビジュアルな要求記述にはデータや制御の流れが中心に表されており、『流れ』という動作の確認には、実際にその振舞いを提示することが有効であると考えられる。このため VRDL による記述で用いられている動作可能なアイコンに対して、動作の記述を利用者が与えることによって、データフローの振舞いをアニメーションとして表示させる手法を開発した。

本稿では、最初に要求モデルとビジュアルな要求言語 VRDL について説明する。次にビジュアルな要求の実行手法とその処理系について紹介する。また、要求定義環境 CARD [5, 9] 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と要求言語

開発してきた要求言語にはビジュアルな要求言語の他にも日本語要求言語や図式言語があるが、これらはすべて要求フレームモデルに基づいている。このモデルは要求記述の枠組を与えるものであり、記述の構成要素に応じて

① **名詞レベル**：名詞と名詞の型

② **動詞レベル**：動詞と動作概念

③ **形容詞レベル**：形容詞と深層概念

④ **単文レベル**：文と格フレーム

⑤ **機能レベル**：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ものである [6]。

名詞は『人間』・『機能』・『データ』・『ファイル』・『制御』・『装置』のいずれかの型に分類される。

一般の文章で使われる表層の動詞は『データの流れ』・『制御の流れ』・『and 木構造』・『or 木構造』・『データ作成』・『レコード検索』・『レコードの更新』・『レコードの削除』・『レコードの挿入』・『ファイルの操作』の 10 種の深層の動作概念のいずれかに分類される。例えば『（データを）入力する』、『（データを）渡す』、『表示される』といった動詞はすべて『データの流れ』という概念に分類される。10 種の概念にあてはまらない動作概念は記述者が新規に定義できる。また、大小関係などの比較を表す形容詞も 6 つの深層概念のいずれかに分類される。

格フレームは深層概念と必須格に対する枠組である。動作概念によって必須格は異なる。例えば『データの流れ』の必須格は動作主、源泉、目標、道具の 4 つであり、それぞれデータ型、機能か人間型、機能か人間型、装置型の名詞が当てはまる。このフレームを用いて必須格の抜けや格に当てはまる名詞の型誤りを検出できる。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 [7]。また、新規の動作概念を定義することも出来るが、その場合は、新規概念の格フレームも併せて定義しなければならない。

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」などの 10 個の性質について、それらを要求記述が満たしているかをチェックするために用いられる [6]。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つの要求言語による記述は、格フレームに基づいた共通の内部表現CRDに変換される。従って、『データや制御の流れといったビジュアルに記述し易い部分をVRDLで、データ構造や機能内容といった部分は日本語要求言語で』というように、用いる要求言語を切り替えて仕様化できる。

3 ビジュアルな要求言語: VRDL

3.1 VRDLの特長

VRDL (Visual Requirements Description Language) [8, 10] の特長を以下に示す。

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していくことによって要求を記述する
4. 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
5. VRDL記述に用いられたアイコンの動作を与えることによって、記述を実行できる
6. VRDL記述に対して、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 [5, 9]

特長の1～4について説明する。5, 6についてはそれぞれ4章、5章で述べる。

3.2 アイコンの定義

DFD[1] やSADT[4] 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 [2] 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 [3] で表現したくても出来ない。

表 1: 具体的なアイコンの定義例

アイコンの形状	意味	
	名称	名詞の型
	顧客	人間
	電話	装置
	資料	データ
	解析者	人間
		人間

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い。

このためVRDLでは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名詞の型をアイコンの意味として定義できる。表1にアイコン定義の具体例を示す。

表1で例えば最初のアイコンは「顧客」という人間型の名詞を表す。また最後のアイコンのように名称を省略することにより、人間型の名詞の総称を意味するアイコンも定義できる。このような総称的なアイコンには記述に表す都度、異なる名前を名称として与えることができる。このように利用者が形状と意味の両方を明示することによって定義されるアイコンを特に基本アイコンと呼ぶ。基本アイコンからは複合アイコンを作成できる。

表 2: “composing” による複合アイコン

基本アイコンの意味	複合アイコンの意味
A,B (同じ型の名詞)	A と B
A (機能か人間) B (装置)	B を用いて A
A (データかファイル) B (装置)	B を通して A

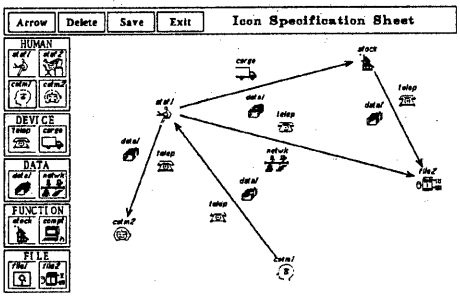

図 1: VRDL 記述例

3.3 複合アイコンの作成

基本アイコンから複合アイコンを作成するために、基本アイコンに適用される操作として、“mulitiplying” と “composing” を導入している。複合アイコンの形状と意味は基本アイコンと操作から自動的に定まるので、利用者は定義しなくて良い。

“mulitiplying” は 2 つ以上の基本アイコンを表す複合アイコンを作成する操作である。例えば『ファイル型で 1 枚のフロッピィディスク』を表す基本アイコン [♀] にこの操作を適用すると [♀] が得られ、その意味は『ファイル型で 2 枚以上のフロッピィディスク』と自動的に定まる。

“composing” は 2 つの異なる基本アイコンを組み合わせた複合アイコンを作成する操作である。組み合わせる基本アイコンによって、得られる複合アイコンの意味は表 2のように異なる。“composing” による複合アイコンの形状は 2 つの基本アイコンを括弧でくくったもので表す。

利用者はこれらの操作を用いることにより、容易に複合アイコンを作成できる。複合アイコンは基本アイコンと同様にビジュアルな要求記述に用いることができる。

3.4 アイコンと矢印による記述

定義したアイコンと矢印を配置していくことによって、要求を記述する。アイコンと別のアイコンとの間の矢印で流れを表すが、VRDL ではデータの流れを表すのに実線矢印を、制御の流れを表すのに破線矢印を用いている。要求フレームモデルの動作概念には、データや制御の流れ以外にも 8 種あり、さらに利用者が自分で新しい動作概念を定義できるので、流れ以外の動作概念を表したい場合は、その動作概念を表すアイコンを別途定義し、それを矢印にラベルとして貼り付けることによって対処するが、一般的にはビジュアルに表す対象はデータや制御の流れで十分と考えている。

図 1にビジュアルな要求記述エディタの画面を示す。画面の左には予め定義したアイコンが示される。この中のアイコンや矢印を指定して、マウスでクリックした位置に張り付けながら記述を進めていく。記述中の 1 つの矢印が 1 つのデータの流れに対応し、矢印の始点はデータの流れの源泉格に、終点は目標格に相当する。矢印の途中に置かれるアイコンは流れるデータと装置に相当する。これら 4 つの必須格に相当する名詞アイコンと 1 つの実線矢印でもって、1 つのデータフロー文が表される。例えば表 1のようにアイコンが定義されているならば、記述の最左の部分は「分析者は顧客に電話で資料を送る」ということを表す。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。

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ビジュアルに記述できる。VRDL の処理系を開発したが、改善点

として以下のものがあげられる。

- **あいまいさの解消:** 新たにアイコンを配置する際に、過去に配置したアイコンに重なる場合や、矢印同士が近過ぎると、配置したアイコンが2つの矢印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
- **アイコンサイズの変更:** アイコンの大きさは同じとしているが、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:** 複雑な要求のためにDFDのような構造的な記述をサポートする
- **複合アイコンのメニュー選択**
- **標準アイコン:** 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

3.5 文書の標準化支援

ビジュアルな要求記述を記述者以外の人が読む場合、例えば✆というアイコンを記述者は「Fax」のつもりで用いたのに読者は「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の形状を読者が定義することによって解決できる。例えば表3のように記述者と読者によって、同じ名詞が異なる形状の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 [7]。

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、標準化アイコンによる表現に変換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 3: **記述者と読者のアイコンの定義**

意味	記述者	読者
「Fax」装置型	✆	FAX
「電話」装置型	☎	☎

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(scenario)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。

このため動作を記述する言語を開発した。この言語は以下の項目を記述できる。

- 動かすアイコンの指定
- アイコンを最初に表示する位置の指定
- アイコンの表示開始
- アイコンの表示終了
- アイコン表示時間の変更

アイコンの動作は、

1. アイコンの移動
2. 少しずつ異なる複数アイコンの表示の切替

によって実現している。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

```
tel_anime(ACT src, ACT goal, ACT data)
{
    ACT tel1, tel2;
    int i;
    ICON icon1, icon2;
    LINE line;

    icon1 = ''telep1'';
    icon2 = ''telep2'';

    tel1 = {icon1, icon2};
    tel2 = {icon1, icon2};

    line = LINESTYLE1 ;

    locate(tel1, 50, 50);
    locate(tel2, 450, 50);

    actfor(i=0;i<11;++i){
        locate(data, 50 + 40*i, -50);
        if(i-i/2*2 == 0){
            display(tel1[0], ON);
            display(tel2[1], ON);
        }
        else {
            display(tel1[1], ON);
            display(tel2[0], ON);
        }
        display(line, ON);
        display(data, ON);
        display(src,  ON);
        display(goal, ON);
    }
    display(tel1, OFF);
    display(tel2, OFF);
    display(line, OFF);
}
```

図 2: **動作記述例**

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、データの流れ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。動作記述中のアイコンは、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、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。

図 1に対する動作記述例を図 2に示す。この例では電話に対してビジュアルな要求記述で使われた ☎ とアニメーション表示のために新たに定義した 📞 の 2つの異なるアイコンを切替えながら表示することによって電話がなるという動作をアニメーションで実現している。

また、動作記述でアイコンの移動の始点、終点、時間間隔を指定することによって、そのアイコンの移動をアニメーションとして表示できる。図 3 ～ 6はアニメーションのスナップショットを示したものである。

動作記述言語を解析し、動作記述を実行するプロトタイプを作成した。このプロトタイプは動作記述を C 言語と X Toolkit によるプログラムに変換し、これを Unix ワークステーション上の X Window(Ver.11) 環境下で動作させている。プロトタイプの使用を通して、以下の改善点が明らかになった。

- **装置の動作指定の簡略:** 「電話が鳴る」、「プリンタに出力される」といった頻繁に記述されるような装置の動作については明に指定しなくても済むようにしたい。
- **データ移動指定の簡略:** データの移動は源泉格のアイコンから目標格のアイコンへの移動が殆んどであり、これも明に指定しなくても済むようにしたい。
- **動作記述の自動生成:** さらには動作を記述しなくても済むようにしたい。例えば状態遷移のような動的な振舞いのモデルが明らかになっている場合は、動的モデルから動作記述を自動生成したい。この場合、アニメーションでデータの流れを見せることにより、データの流れ(機能) と状態遷移 (動的な振舞い) の両方を検証できる。

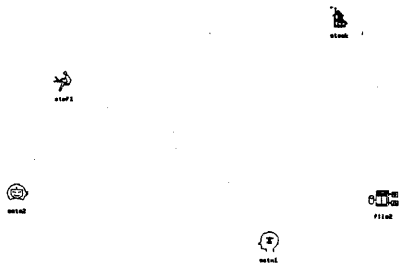

図 3: アニメーションのスナップショット (1)

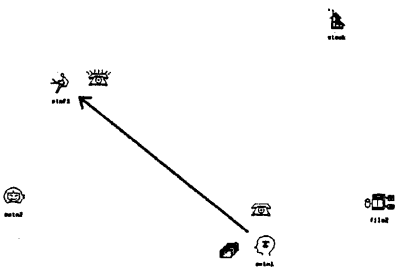

図 4: アニメーションのスナップショット (2)

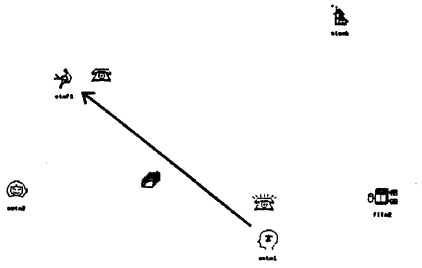

図 5: アニメーションのスナップショット (3)

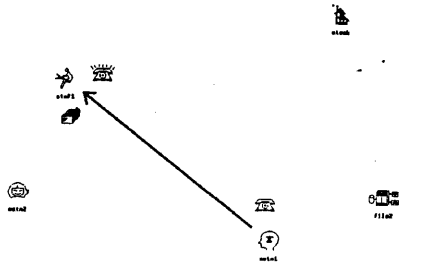

図 6: アニメーションのスナップショット (4)

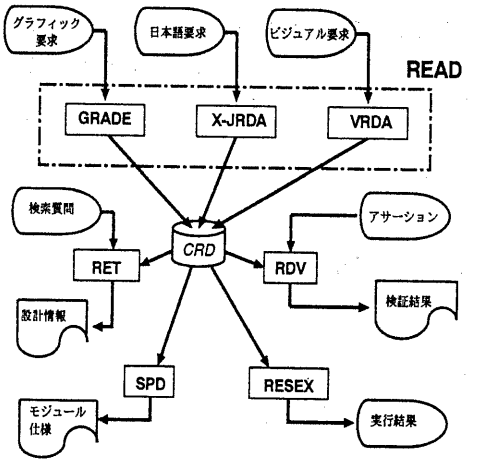

図 7: 要求定義環境: CARD

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 [5, 9]。 CARD の構成を図 7に示す。

CARD は

- 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。

ビジュアルな要求記述の実行系の CARD 環境への統合は今後の課題である。

6 おわりに

ビジュアルな要求言語 VRDL とその記述の実行手法について紹介した。VRDL と日本語要求言語とを場合に応じて使い分けることによって、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。さらにビジュアルな要求記述の実行によってデータフロー記述の確認ができる。

3.3 節と 4 章で示した VRDL の処理系の使用を通して判明した問題点の解決が今後の課題である。特に、動的なモデル (状態遷移) 記述からのアイコン動作記述の自動作成手法を今後中心に進めていく予定である。

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修了の程國政君 (現在中華民国中央銀行) と島田淳一君 (現在郵政省) に感謝する。

参考文献

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

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

[3] 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.

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

[5] 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).

[6] 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).

[7] 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).

[8] 大西 淳：「ビジュアルなソフトウェア要求仕様化技法」情報処理学会研究報告, Vol.93, No.13, SE90-8, 1993 年 2 月, pp.57-64.

[9] A. Ohnishi, K. Agusa: "CARD: A Software Requirements Definition Environment," Proc. of IEEE Int. Symp. Requirements Engineering, San Diego, CA, U.S.A., 1993, pp.90-93.

[10] A. Ohnishi: "Visual Software Requirements Specification Language: VRDL," Proc. of the fifth International Conference on Software Engineering and Knowledge Engineering (SEKE93), San Francisco, CA, U.S.A., June 1993 (to appear).

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

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

[13] Shi-Kuo Chang: "A Visual Language Compiler," IEEE Trans. Softw. Engnr. Vol.15, No.5, 1989, pp.506-525.

[14] St-Denis, R.: "Specification by example using graphical animation and a production system," Proc. IEEE 23rd HICSS, Vol.2, 1990, pp.237-246.