ビジュアルなソフトウェア要求仕様化技法

大西 淳†

記述者のイメージにできるだけ忠実にソフトウェア要求を仕様化することによって，要求の仕様化が容易になる．本稿では，ビジュアルなソフトウェア要求仕様化技法を提案する．本技法により，記述者は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でき，さらにそのアイコンの意味として具体的な名詞，名詞の型を定義できる．アイコンの形状と意味があらかじめ定まっている既存の言語を使う場合に比較して，記述者のイメージがより正確に要求仕様に反映されるとともに，記述した要求の意味をより正確に読み手に伝えることができる．またアイコンの動作をシナリオとして与え，動作をアニメーションとして表示することによって，要求記述の正当性を検証できる．手法に基づいて開発し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．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，日本要求言語とビジュアルな要求言語のどちらでも要求を記述できる．このため，日本語で書きやすい要求は日本語で，図で表しやすい要求はビジュアルな言語で記述でき，仕様化を一層容易にしている．またビジュアルな要求記述に対して，既開発の要求定義環境CARDで用意している各種のツールを適用できるため，ビジュアルな要求記述の品質が向上される．

A Visual Software Requirements Specification Technique

ATSUSHI OHNISHI †

The author proposes a visual software requirements specification technique including describing a visual software requirements specification (SRS) with a visual requirements language and executing the SRS. The technique proposed here provides a describer for defining both the shape and semantics of icons to specify the requirements just what he imagines. The technique also provides for describing icons' movements as a scenario description. This scenario enables an execution of the SRS with animation. The describer can confirm his requirements by checking the animation. A visual requirements language named VRDL and tools are illustrated with examples. 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. This feature improves the easiness to write requirements. The visual requirements specification technique proposed here also contributes to improve the understandability and the correctness of an SRS. The author also touches on the requirements definition environment named CARD.

1. はじめに

効率良くソフトウェアを開発するためには，高品質なソフトウェア要求仕様が必須となる[2]．本稿では要求の書きやすさ・読みやすさと正しさの向上を目標としたビジュアルな要求仕様化技法を提案する．技法により (1) ビジュアルな要求の記述と (2) シナリオによるビジュアルな要求記述の実行が可能となる．さらにビジュアルな要求記述の正当性の検証ができる．

ソフトウェア開発の上流工程を支援するCASEツールのほとんどは，データフロー図 (DFD) の描画機能を備えている[4]．DFDではデータの流れ，プロセス，ファイル，データ源泉とデータ吸収を表すために4つの図形が用いられる[3]が，場合によっては，用意された4つの図形以外に例えばフローチャートのシンボルといった異なった図形や，頭に描いたイメージをそのまま表現するアイコンを用いたほうが素直に要求を表すことができる．

ビジュアルな要求仕様化技法の第一の特長は，データや制御の流れに関するソフトウェ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せることである．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で表し，それらを

† 立命館大学理工学部情報学科
Department of Computer Science, Faculty of Science & Engineering, Ritsumeikan University

エディタ上で配置することによって，要求を仕様化する．

第二の特長は，要求仕様の誤りの検出ができることである．定義したアイコンの意味は我々が開発した要求フレームモデル[9]に基づいて定義される．要求フレームに基づいた要求記述の正当性検証手法については既に確立している[9),10)]．

ラピッドプロトタイピングを用いたエンドユーザによるプロトタイプの使用経験は要求の確認に有効な手段であるが，第三の特長はアイコンの動作を表したシナリオを用いて，要求仕様を実行できることである．

最初に要求言語の基礎を与える要求フレームモデルについて説明する．次にビジュアルな要求言語と仕様化技法について説明する．さらに，ビジュアルな要求言語の処理系と要求定義環境CARDについて述べる．

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があるが，これらは要求フレームモデル[9]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・形容詞レベル：動詞・形容詞と動作概念
③単文レベル：文と格フレーム
④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．

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つの動作概念のいずれかに分類される．

格フレームは動作概念と必須格に対する枠組である．動作概念によって必須格は異なる．例えば「データの流れ(DFLOW)」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，機能型か人間型，機能型か人間型，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．また，新規の動作概念を定義することもできるが，その場合は，新規概念の格構造も併せて定義しなければならない[10)]．

日本語要求言語の場合，複文や重文も記述できるが，これらは動作概念を1つしか含まない単文に分割されてから，格フレームに基づいて解析される．

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1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[9)]．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．

2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やすい部分を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

VRDL (Visual Requirements Description Language)[12)]の目標を以下に示す．

i データや制御の流れに関するソフトウェ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せるようにする
ii ビジュアルな要求を記述した本人だけでなく他の人間にもわかるようにする
iii 要求フレームによる誤り検出以外にビジュアルな要求仕様におけるデータの流れをアニメーション表示することによって誤りを検出する

VRDLの機能的な特長は以下に示したとおりであり，特長の1，2，3によって目標iを，特長の4によって目標iiを，特長の5によって目標iiiを達成している．

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

表1 アイコンの定義例
Table 1 Example of icons definition.

アイコンの形状	アイコンの意味
	人間型
	「ファックス」、装置型
	「メッセージ」、データ型

3.1 アイコンの定義

DFD[3]やSADT[8]に代表される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と意味は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，図形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[6]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[7]で表現したくてもできない．

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，要求を記述しやすいし，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できない．

このためVRDLでは単にアイコンの形状を定義できるだけでなく，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で最初のアイコンは人間型の名詞の総称を意味する．このような総称的なアイコンには記述に表すたびに，異なる名前を名称として与えることができる．2番目のアイコンは装置型の「ファックス」を意味する．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．

3.2 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”を導入する．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．

表2 “composing”による複合アイコン
Table 2 Elementary icons and “composed” icon.

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

“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．

3.3 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の中で特に流れを選んだ理由はビジュアルに表しやすいと考えたからである．流れ以外の動作を含んだ要求は日本語要求言語によって表すことができる．

例えば「富市さんからBillさんへメッセージをファックスで送る」という要求をDFDで表すと**図1**のようになる．同じ要求を表1で定義したアイコンを用いてビジュアルに表したものを**図2**に示す．図1と比較すると，人の頭部の輪郭をアイコンとして用いることによって富市さんやBillさんが人間であることが直感的に理解できる．このように適切なアイコンを用いることで，要求をより分かりやすく，また頭の中のイ

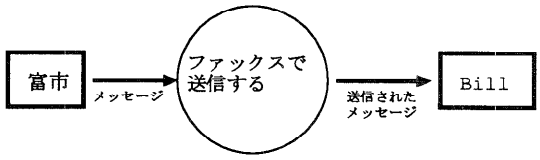


図1 DFDによる要求
Fig. 1 Simple example with DFD.

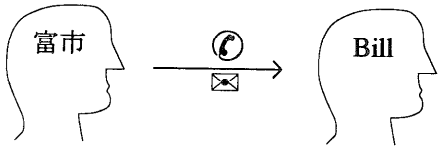


図2 ビジュアルな要求文
Fig. 2 Simple example of a visual requirement.

メージに近い形で表すことができる．

VRDLでは要求フレームモデルにおける動作概念の必須格にアイコンを，概念に矢印を対応させて，要求文をビジュアルに表している．従って，各々の要求文は格フレームを正確に反映している．同じ要求を日本語要求言語によって「富市さんがBill氏へメッセージをファックスを使って送る」と表しても同一の内部表現に変換される．

このようにして，要求は日本語でもビジュアルな言語でも表される．それぞれの要求記述が内部表現に変換されてから統合され，全体として誤りがないか機能フレームによって解析される．

3.4 ビジュアルな要求記述の標準化

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「ファックス」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．

この問題は標準化されたアイコンの導入により解決できる．例えば**表3**のように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，読者は標準化アイコンの意味をあらかじめ知っておくことにより正しく理解できる．

表3 記述者のアイコンと標準化アイコン
Table 3 Describer's icons and standard icons.

意味	記述者	標準化アイコン
「ファックス」装置型	✆	📠
「電話」装置型	☎	☎

3.5 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．

- 動作させるアイコンの指定
- アイコンを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
- アイコン表示速度の変更

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り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．

4. ビジュアルな要求記述の処理系

提案したビジュアルな要求仕様化技法の有効性を確認するためにプロトタイプを試作した．プロトタイプは

- ビジュアルな要求の記述と解析系
- ビジュアルな要求記述の実行系

に分けられる．

4.1 アイコンの定義

基本アイコンの形状については，X windowシステムのbitmapコマンドといった既存のツールを用いて描いたり，スキャナなどの装置を用いて，bitmapデータとして定義する．プロトタイプではアイコンの縦横のサイズは固定としている．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登録することにより基本アイコンを定義する．現在のところ複合

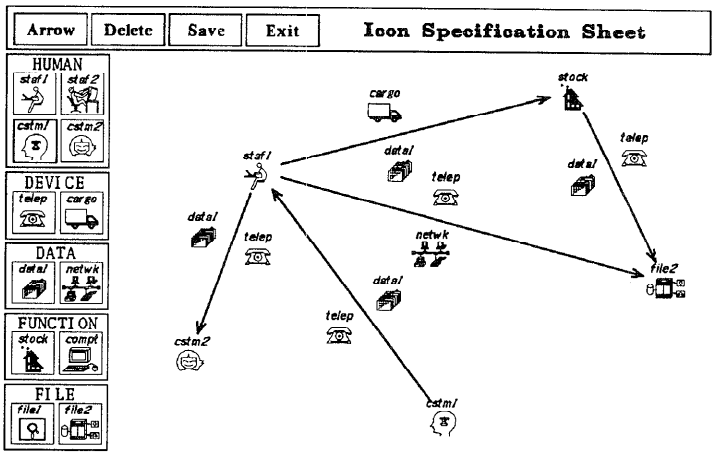


図3 VRDL 記述例
Fig. 3 VRDL example.

アイコンはサポートしていない．

4.2 ビジュアルな要求の記述と解析

次に定義したアイコンをエディタ上で配置する．図3にビジュアルな要求記述エディタの画面を示す．画面の左にはあらかじめ定義した名詞のアイコンが型ごとに分類されて表示される．最初に動作概念としてデータまたは制御の流れか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ながら，記述を進めていく．記述中の1つの実線矢印が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DFLOW文が表される．

例えば**表4**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．各々の文は格フレームに基づいており，日本語要求言語と共通の内部表現に変換される．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．

同じ要求をDFDで表すことはできるが，その場合はバブルや矢印に付けるラベルに細かい説明が必要となる．本手法では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より近い形で要求をビジュアルに記述できる．

4.3 ビジュアルな要求記述の実行

VRDLによる記述において，データの流れ

表4 記述中のアイコンの定義例
Table 4 Icons in a visual SRS.

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
	分析者	人間

```
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4 動作記述例
Fig. 4 Scenario description.

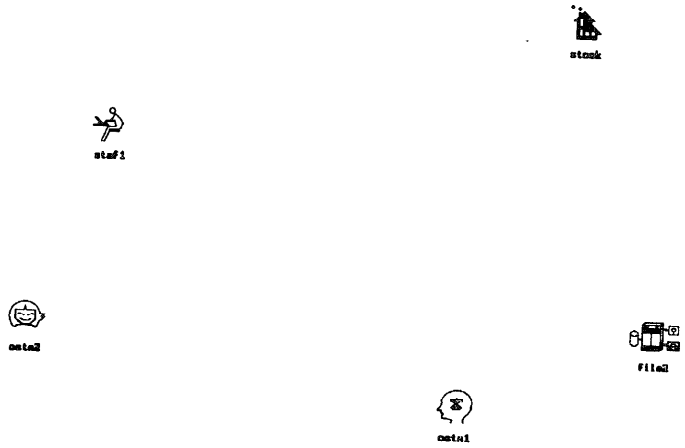


図5 アニメーションのスナップショット(1)
Fig. 5 Snapshot of the animation(1).

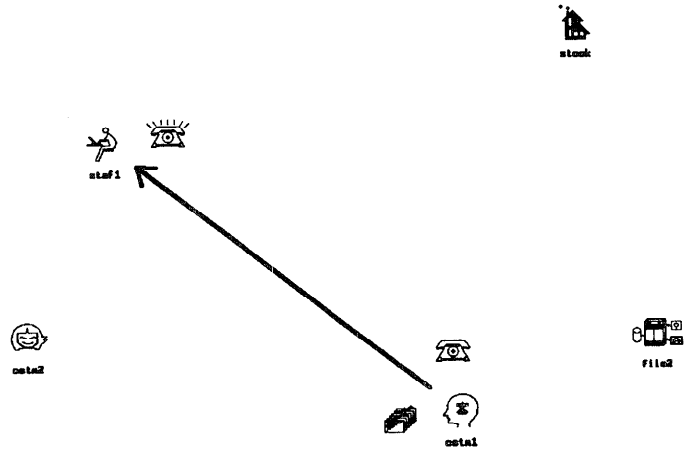


図6 アニメーションのスナップショット(2)
Fig. 6 Snapshot of the animation(2).

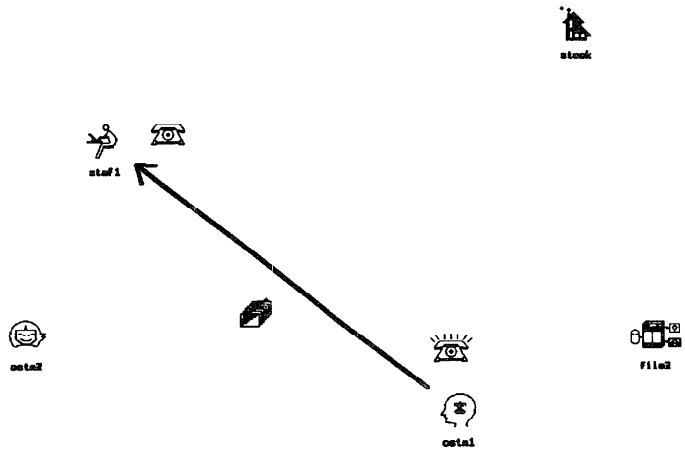


図7 アニメーションのスナップショット(3)
Fig. 7 Snapshot of the animation(3).

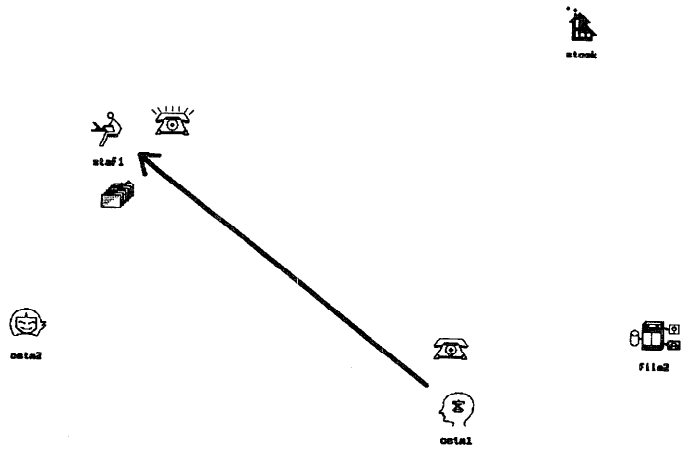


図8 アニメーションのスナップショット(4)
Fig. 8 Snapshot of the animation(4).

(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3に対する動作記述の一部を図4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り替えながら表示することによって電話が鳴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5〜8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 Window System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環境下で動作させている．「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ライブラリ化して明に指定しなくても済むようにしている．

現在は動作記述を用意しなくても済むように技法を研究中である．具体的には状態遷移で表された動的な振舞いのモデル（動的モデル）が明らかになっている場合は，動的モデルにおける状態遷移を順序付けすることによって動作記述を自動生成する．

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[11]．CARDの構成を図9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
  - 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

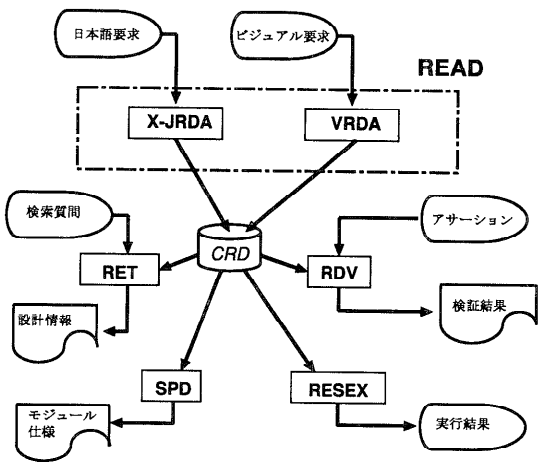


図9 要求定義環境：CARD
Fig. 9 Card environment.

－要求記述実行系：RESEX
●設計支援系：SPD
から構成されている．要求は日本語かビジュアルな言語のいず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．どちら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. 関連研究

関連した研究として，文献5)ではシナリオに基づいたプロトタイピングと画面ベースのシナリオ生成手法が提案されている．生成されるのはシナリオに基づいたディスプレイ画面のシーケンスであり，これによって人間と計算機のやりとりが正しいかを確認するものである．我々の技法では確認の対象はデータの流れに関する要求であり，流れはアニメーション表示されて，その向きや順序が正しいかを確認する．

文献1)ではSIL-ICONと名付けられたビジュアル言語コンパイラが提案されており，アイコンの構文と意味，ならびに言語の文法を定義できる．我々の技法ではアイコンの意味は要求フレームに基づいており，複合アイコンの意味は自動的に定まる．

文献13)ではアイコンのアニメーション化によるプロトタイピング手法が提案されている．アイコンは初期状態と終了状態の形状が定められ，初期状態から終了状態までの動きは垂直方向か水平方向の移動か色の塗り潰ししかない．我々の技法ではアイコンの動きは任意の方向への移動と，形状の異なる複数のアイコンの切り替え表示によって実現され，しかも動作速度を指定できるため多彩な動きが表現できる．

7. おわりに

ビジュアルな要求仕様化技法とそれに基づいた処理系について紹介した．DFDのような記法に比べると，頭の中のイメージにより近く仕様化でき，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が向上すると考えている．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要求の正しさの確認ができる．さらにCARD環境に適用することで要求記述の品質を高めることができる．

VRDLのエディタの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，矢印同士が近すぎると，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
- **複合アイコンのサポート**
- **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

これらの問題点の解決と動作記述の作成支援手法の確立，ならびにソフトウェア開発への適用が今後の課題である．特にソフトウェア開発への適用に関しては，より簡便にアイコンを作成するためのツールを用意したり，エディタの使い勝手をより向上させるために，利用可能なアイコン一覧の表示機能や記述したビジュアルな要求文を日本語で表示することによる確認機能の付加といった機能の充実と編集操作の簡便化を支援する必要があると考えている．

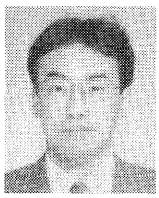
謝辞 処理系の開発に当たり，京都大学大学院工学研究科応用システム科学専攻修士課程修了の程國政，島田淳一，服部芳明，および立命館大学理工学部情報工学科4回生の岩倉次郎の各氏に感謝する．本研究は一部文部省科学研究費補助金一般研究(C)06680318による．

参考文献

1) Chang, S.K.: A Visual Language Compiler, *IEEE Trans. Softw. Eng.*, Vol.15, No.5, pp.506-525 (1989).
2) Davis, A.M.: The Analysis and Specification of Systems and Software Requirements, *IEEE Tutorial System and Software Engineering* (Thayer, R. H. and Dorfman, M.(eds.)), pp.119-144, IEEE-CS Press (1990).
3) DeMarco, T.: *Structured Analysis and System Specification*, p.352, Prentice-Hall (1978).
4) Fisher, A.S.: *CASE Using Software Develop-*

ment Tools, p.287, John Wiley (1988).
5) Hsia, P. and Yaung, A.T.: Screen-Based Scenario Generator: A Tool for Scenario-based Prototyping, *Proc. IEEE HICSS*, Vol.2, pp.455-461 (1988).
6) 情報処理学会編: 情報処理ハンドブック, p.1166, オーム社 (1980).
7) 日本規格協会編: JISハンドブック情報処理ソフトウェア・符号編 JIS X 0121 (1986), p.1699 (1992).
8) Marca, D. A. and McGowan, C. L.: *Structured Analysis and Design Technique*, p.393, McGraw-Hill Book (1988).
9) 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, Vol. 31, No. 2, pp.175-181 (1990).
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, Vol. 33, No. 8, pp. 1064-1071 (1992).
11) Ohnishi, A. and Agusa, K.: CARD: A Software Requirements Definition Environment, *Proc. IEEE Requirements Engineering* (*ISRE*), pp.90-93 (1993).
12) Ohnishi, A.: A Visual Software Requirements Definition Method, *Proc. IEEE Requirements Engineering* (*ICRE '94*), pp.194-201 (1994).
13) St-Denis, R.: Specification by Example Using Graphical Animation and a Production System, *Proc. IEEE HICSS*, Vol.2, pp.237-246 (1990).

(平成6年 7 月 22 日受付)
(平成6年10月13日採録)



大西 淳 (正会員)

1957年生. 1979年京都大学工学部情報工学科卒業, 81年同大学院工学研究科修士課程情報工学専攻修了. 83年同博士後期課程情報工学専攻中退. 83年から89年まで京都大学助手. 89年から94年まで京都大学助教授. 94年から立命館大学教授. 理工学部情報学科に勤務. この間90年から91年までカリフォルニア大学客員研究員. 京都大学工学博士. ソフトウェア工学, 特に上流工程の要求定義や設計技法に興味がある. IEEE Computer Society, ACM, 日本ソフトウェア科学会各会員.