

インタラクティブシステムの設計における タスクの形式的記述とその実現

地平 稔 池田瑞穂 高田喜朗 関 浩之

奈良先端科学技術大学院大学 情報科学研究科

現在アプリケーション開発工程において、ユーザインタフェース部分の開発の占める割合が大きくなって
いる。しかし、従来の設計アプローチには、設計のガイドラインが断片的で設計方法論として体系化されて
いないなどの問題がある。

著者らは、設計の上流工程から適用できる、体系化されたインタラクティブシステムの設計法の研究を行っ
ている。この設計法では、ユーザタスクの形式的記述を一連の設計プロセスへの入力とする。そして、前段
階の仕様を満たすような変換に基づいて詳細化する。仕様に現れるイベントをユーザが行うものとシステム
が行うものに分類し、それらが使う UI 部品などを指定することで、プロトタイプを自動生成する。

本稿では書籍の販売システムを例題として本設計法の説明を行う。

Formal Specification and Implementation Using Task Flow Diagram in Interactive System Design

Minoru Jihira Mizuho Ikeda Yoshiaki Takata Hiroyuki Seki

Graduate School of Information Science
Nara Institute of Science and Technology

In development of an application system, designing the user interface comes to occupy a larger part.
Various techniques have been proposed and used for interactive system design. However, it is often pointed
out that a design process of an interactive system highly depends on designers' experience and lacks a
systematic methodology.

In this paper, a systematic method of interactive system design is proposed. First, the user's task is
specified as a task flow diagram. The diagram is refined by decomposing each task into several subtasks
which represent system's actions and user's ones. A prototype can be generated automatically from a
detailed task flow diagram augmented by information on user interface. The proposed method is explained
by using a book purchasing system.

1 まえがき

ソフトウェアシステムの多機能化、ユーザの多様
化により、アプリケーション開発工程においてユー
ザインタフェース部分の開発の重要性が高まってい
る。そのようなインタラクティブシステムの開発に
おける、生産性や品質の向上を実現するための設計
法が必要である。しかし、従来の設計法は、システ
ムの機能を実現することに重点が置かれており、使
いやすいシステムを実現することがその目的ではな
い。そのためインタラクティブシステムの開発のため
の、特有のさまざまな設計法や設計技術が提案さ
れている [1, 6]。

しかし、これらの手法や技術にも以下のような問

題がある。

- 要求定義、ユーザ挙動のモデル化、メンタルモ
デルの構築などの個別技術が場当たりに用い
られ、設計プロセス全体の整合性が無い。
- 設計のガイドラインは断片的であり、設計法と
して体系化されていない。
- プロトタイプの作成とそのテストを繰り返す方
法がよく用いられるが、この方法は開発コスト
が大きい。

これらの原因から、アプリケーションの品質は開
発者や設計者の経験や感性に依存しがちになり、ユー
ザの要求を十分に満たしていないことが多い。

そこで著者らは、設計の上流工程から適用できる

体系化されたインタラクティブシステムの設計法の開発・提案を行ってきた [3].

提案する設計法は以下の 2 つの特徴を持つ.

(1) ユーザタスクの記述を初期仕様とする.

従来の設計技法ではシステムの機能の仕様を重視していたため, ユーザタスクの構造や流れが設計プロセスに反映されにくかった. ユーザタスクの形式的記述を, 設計プロセスへの入力とすることにより, 初期段階からユーザタスクの流れを意識した設計が可能になる.

(2) 仕様の形式的記述と詳細化を行う.

初期仕様を形式的記述法を用いて記述し, 前段階の仕様を満たすように詳細化を行う. これにより詳細化前後の不整合を防止できる. また, 形式的記述を行うことで, プロトタイプの自動生成や仕様の形式的検証が可能となり, 設計作業の効率を計ることができる.

著者らは, [3] で述べた設計法についてさらに考察し, 特にタスク間のデータの授受に関する記述法やプロトタイプ生成法を開発することで, 本設計法を実際にプロトタイプが生成できるものにした. 本稿ではこれらを含め, 本設計法の概要を説明する.

以降, 2 節では本設計法の概要について述べる. 3 節～5 節では設計法の各ステップについて詳しく述べる. 6 節では本設計法で用いる形式的記述言語であるタスク図について述べる. 最後に, 7 節でまとめと今後の課題について述べる.

1.1 関連研究

形式的記述をユーザインタフェース設計に応用する研究は多くなされているが, 現実のシステム設計において仕様を記述する目的に使えるものは少ない.

グラフィカルユーザインタフェースの仕様を形式的に与えることでその形式的検証とプロトタイプ自動生成を行う方法が研究されている [7]. これは, 上述の特徴 (2) に関して本研究と同じ目的を持つものである. しかし [7] では, 仕様を作成する段階について特に方法論が与えられていないという問題がある.

また, [7] を含め従来のほとんどの設計法では, システムに要求されている各機能を実現し, その機能の実行に必要な情報をユーザから与えてもらうことを目的として, インタフェースの設計が行われる. しかし, 上の特徴 (1) で説明したとおり, ユーザインタフェースの設計は, ユーザの行いたいタスクの構造や流れに着目して行われるべきである.

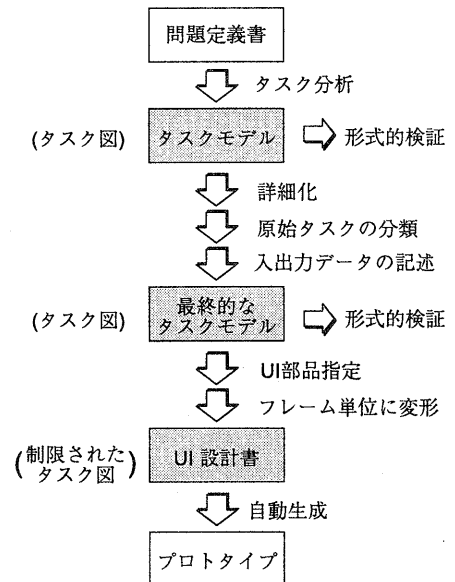


図 1: 提案する設計法

ユーザの振る舞いの形式的モデルを設計に利用する試みとして [5] がある. これは, ユーザおよびシステムのモデルをそれぞれベトリネットで記述し, ユーザがシステムを使用する際に起こる不具合をモデル上で検出する, という枠組について述べたものである. しかし, [5] は設計法として体系化されておらず, またベトリネットはタスク構造を自然に書き表す目的には適していない.

2 提案する設計法の概略

図 1 に本設計法の概略を示す. ここで, 矩形は作成されるドキュメント, 矢印はドキュメント間の変換処理を表す.

本設計法における設計手順は以下のようになる.

- (1) 問題定義を記述する.
- (2) タスクモデルを作成する. …タスクとサブタスクの関係や実行手順などのタスクの構造を記述する. これには 7 節で定義するタスク図と呼ぶ形式的記述法を用いる.
- (3) タスクモデルの詳細化および実装に必要な情報の付加を行うことで, 最終的なタスクモデルを得る. …各タスクを十分細かいレベルまで繰り返し詳細化する. 同時に, システムが行うタスク

クとユーザが行うタスクの分類や、タスク間のデータの授受に関する記述を行う。

- (4) ユーザインタフェース (UI) 設計書を作成する。… UI 設計書は、タスクモデルに、現実の入出力操作や操作画面との対応に関する情報を加えたものである。また、システムの実装を可能とするため、UI 設計書では記述可能なタスクの構造が制限されている。即ちタスクモデルの構造を変形する形で UI 設計書を作成する。
- (5) プロトタイプを自動生成する。

また、ステップ (2)、(3) の後に必要に応じて以下を行う。

- (6) タスクモデルの検証を行う。…満たすべき性質を時相論理式で記述し、作成されたタスクモデルがその性質を満たすかどうか形式的に検証する。

以下では「通信販売による本の購入」を例題として、上記の各ステップを説明する。

3 タスクモデルの作成

3.1 問題定義書の作成

問題の定義を自然言語で記述する。すなわち、設計するシステムの目的や必要とされる機能、対象とするユーザなどを記述する。

この例では以下ようになる。

ユーザが目的の書籍を見つけ、注文し、入手できるような計算機システムを作成する。対象とするユーザは、WWW を利用でき、通信販売を利用したことがあるとする。

3.2 初期タスクモデルの作成

問題定義を基に、タスクの構造を表現したタスクモデルを作成する。これにはタスク図という記述法を使用する。この記述法については 7 節で詳しく述べる。

図 2 はこの例におけるタスクモデルの一部である。ここでは、「通信販売で本を買う」タスクは、「本と冊数を指定する」「発送先を指定する」「支払方法を指定する」の 3 タスクの並行実行で実現できることが記述されている。また、「本と冊数を指定する」タスクは、検索によって「本を探し」た後「冊数を指定する」ことで実現されること、「冊数を指定」した後、任意の回数「冊数を変更」できること、なども記述されている。

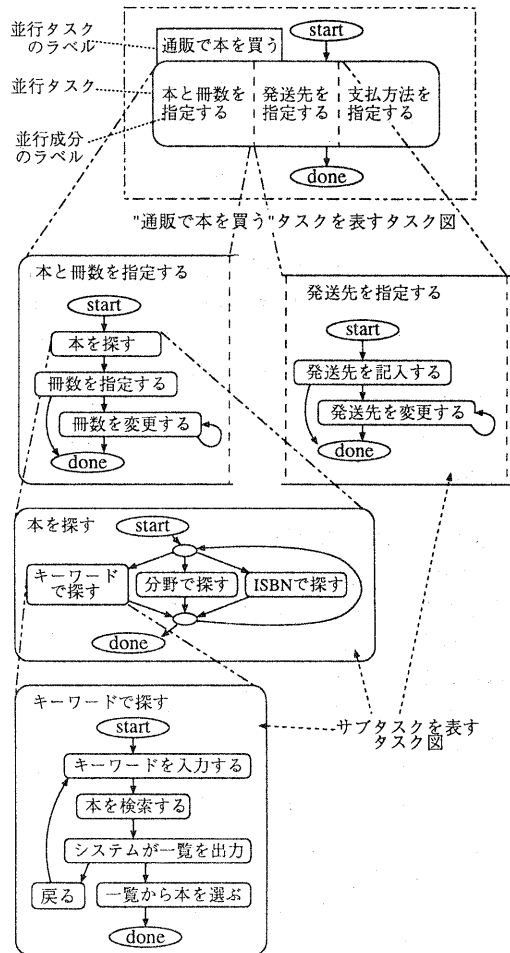


図 2: タスクモデル

このように、タスク図はタスクを階層的に表現できるので、概略を記述した後に各部分を記述していくような段階的な詳細化が行いやすい。

3.3 タスクモデルの詳細化と実装に必要な情報の付加

タスクモデルにおいて、タスクを複数のサブタスクに分解したり、新しいタスクを追加したりすることで、より詳細なタスクモデルを作成する。最終的にタスクモデル中の各原始タスクがユーザやシステムの 1 動作となるまで詳細化する。このようなタスク (ユーザやシステムの 1 動作) をイベントと呼ぶ。

また詳細化と並行して、以下のような情報をタスクモデルに付加していく。

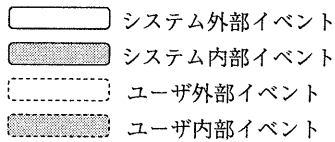


図 3: イベントの分類

- (1) イベントの分類. 各イベントを, システムが行うものとユーザが行うものとに分類する. また, システムとユーザの間で送受信をするイベントとそれ以外のイベントにも分類する.
- (2) タスク間のデータの授受に関する記述.

以下では, 詳細化手順とこれらの付加情報について説明する.

3.3.1 イベントの分類

イベントにはユーザの意思決定によるものと自動化できるものの双方が含まれるので, イベントをユーザが行うものとシステムが行うものとに分類する. これらをそれぞれユーザイベント, システムイベントと呼ぶ. さらにそれらを, ユーザとシステムの間で送受信が行われる外部イベントと, それ以外の内部イベントに分類する. すなわち, タスクモデル中の各イベントを以下の4つに分類する.

- システム内部イベント… e.g., ~を計算する, ~を検索する
- システム外部イベント… e.g., ~を表示する
- ユーザ内部イベント… e.g., ~することに決める
- ユーザ外部イベント… e.g., ~を入力する, ~を選ぶ

タスク図中では, 図3のように記述する.

3.3.2 データの入出力に関する記述

タスクモデルにおけるタスク間のデータの授受を表すために, 各タスクに対しデータの入出力に関する記述を行う. ここでは, タスクは変数を介して他のタスクとデータの授受をするものとし, 各タスクについてそれが値を参照する変数や値を更新する変数を記述する.

各タスクにおけるデータの入出力は, 以下の4種類に分けて記述する.

in, out, see, show

これらはそれぞれ変数の集合である. in, see がそのタスクの入力, out, show は出力に該当する. また, 各変数にはそのデータ型を指定する.

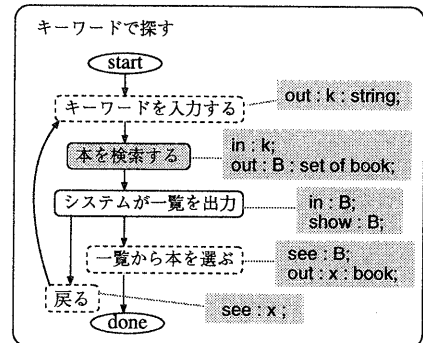
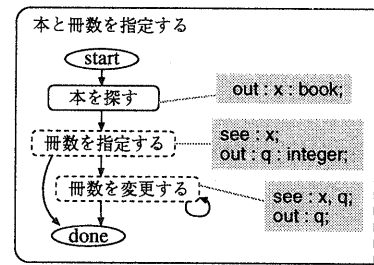


図 4: 入出力データの記述

図4では, in, out などの種別, 変数名, 型名を並べて記述している. ただし, その変数の型が他の箇所で指定されている場合は型名を省略している.

入出力データの種類 データが格納される記憶領域に関して, システムが参照できる記憶領域とユーザが参照できる記憶領域を分けて考える. 前者に対する作用を in と out で表し, 後者に対する作用を see と show で表す (図5). 図5の中央にある2つの矩形が記憶領域を表し, 上側はシステムが参照できる領域, 下側はユーザが参照できる領域を表している. 矢印はデータの流れる向きを表す. ユーザが参照できる記憶領域とはすなわち画面上における各種の表示である.

例えば図4において, タスク「本を検索する」は, 変数 k の値を参照し, 変数 B の値を更新することが記述されている. 更新後の B の値は, 以降に実行されるシステムイベントによって参照され得る. また, タスク「システムが一覧を出力」は, 変数 B の値を参照し, 画面にその値を表示する¹.

図5および上記の説明から明らかな通り, イベ

¹「out: B 」が「記憶領域 B に何らかの値を書き込む」ことを表すのに対し, 「show: B 」はより単純に「(in で参照した) 変数 B の値を画面に表示する」ことを表す.

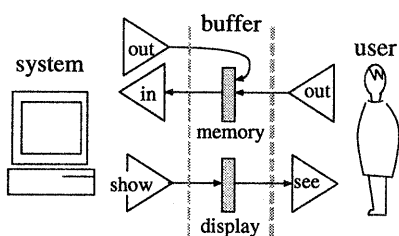


図 5: 入出力データの種類

ントについてはその種類に応じて、指定可能な入出力データが以下のように決まっている。

- ユーザ外部イベント: see, out
- ユーザ内部イベント: see
- システム外部イベント: in, show
- システム内部イベント: in, out

また、イベント以外のタスクについては、全サブタスクの in の和集合（またはその部分集合）をそのタスクの in とし、以下 out, see, show についても同様とする。

データ型 後述の UI 部品型と区別するため、データを格納する変数の型をデータ型と呼ぶ。データ型には、integer, string などの単純組み込み型、設計者が定義する型、および、型 T の要素の集合を表す set-of(T) 型がある。

各データ型は独自のメンバ関数を持つ。各型は、少なくとも、値の代入を行う演算と、自身の値を画面に表示する関数をメンバ関数として持つ。

代入およびメンバ関数呼び出し 各イベントにおける out の要素（変数）に対して、その変数に代入される値や式を記述することができる²。例えば「out: $q = 1$: integer」という記述は、このイベントの実行によって、変数 q に 1 が代入されることを表している。代入される値として、定数、組み込み演算、in に指定した変数などからなる式を指定できる。

4 UI 設計書の作成

システムの実装可能な記述としてユーザインタフェース (UI) 設計書を作成する。これは、タスクモデルに対して以下のように情報の追加および構造の変形を行ったものである。

²最終的にプロトタイプ実装段においては、out に指定された代入演算がそのイベントの意味そのものとなる。

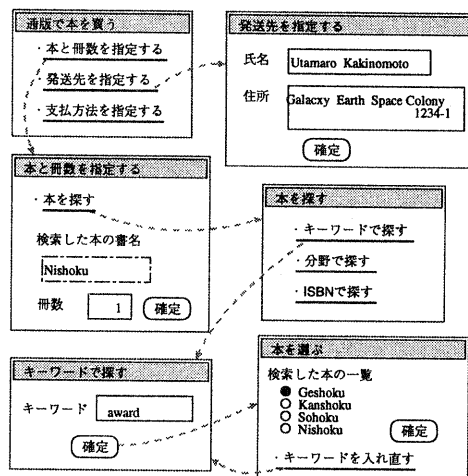


図 6: プロトタイプユーザインタフェース

- (1) 各ユーザ外部イベントにおいて使用される UI 部品を指定する。
- (2) タスクの階層構造をなくし、平坦な構造に展開する。同時に、イベントの集合を各フレームに対応する部分集合に分割する。

UI 部品とは、入力欄やボタンなど、ユーザからの入力を受け付けるオブジェクトである。フレームとは、UI 部品を表示するための画面の一領域を指す。本設計法で得られるプロトタイプは、ひとつのフレームとそれに属する UI 部品を画面に表示し、ユーザの入力に応じて表示するフレームを変えていくようなプログラムである (図 6)。

以下、(1), (2) について説明する。

4.1 UI 部品の指定

ユーザ外部イベントは、システムにおいてはユーザからの入力を受信するイベントである。ユーザからの入力は UI 部品を介して行われる。そこで、各ユーザ外部イベントに、そこで使用される UI 部品を指定する。図 7 は、図 4 と同じタスクモデルに対して UI 部品を指定した図である。図 7 では、イベント「冊数を指定する」に UI 部品 t_1, s_1 が指定されている。また、これらの型 (UI 部品型) も指定されている。

UI 部品型 UI 部品型には、単純 UI 部品型と、単純 UI 部品型 U の要素の集合を表す set-of(U) 型がある。

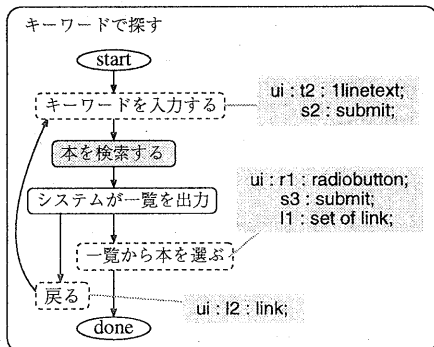
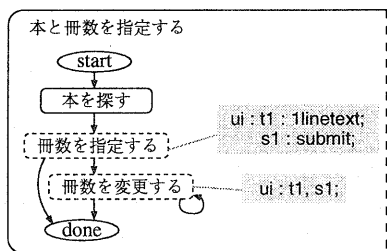


図 7: UI 部品の指定

使用可能な単純 UI 部品型の集合は、対象とする実装の枠組によって決まる。実装の枠組とは、使用可能な UI 部品型の集合や、プロトタイプを記述するプログラミング言語などを規定するものである。WWW, Visual BASIC などが実装の枠組の例として考えられる。例えば、WWW を実装の枠組とした場合、単純 UI 部品型として以下のようなものが考えられる。

- 1linetext …一行のテキスト入力欄
- textarea …複数行のテキスト入力欄
- radiobutton …選択ボタン（複数選択が不可能）
- checkbox …選択ボタン（複数選択が可能）
- submit …上記の各部品に入力した値をシステムに通知するボタン
- link …ハイパーリンク（選択がただちにシステムに通知されるボタン）

各 UI 部品型は独自のメンバ関数を持つ。例えば各単純 UI 部品型は、ユーザが入力した値を返す関数 (entered) と、その UI 部品自身を画面に表示する関数をメンバ関数として持つ。

UI 部品の表示 各 UI 部品に対し、それを表示するシステム外部イベントをタスクモデルに追加する。図 8 は、図 7 を基に作成した UI 設計書である。図

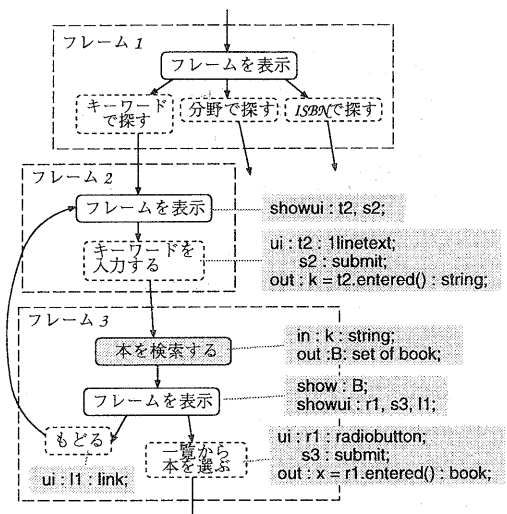


図 8: UI 設計書

8 では showui という項目をシステム外部イベントに付記し、そのイベントが表示すべき UI 部品を記述している。

UI 部品からの値の読み出し ユーザ外部イベントの out には、出力変数に代入する値として、UI 部品のメンバ関数呼び出しを記述できる。この記述は、ユーザが入力した値をシステムが参照できる変数に代入することを表す。

図 8 では、イベント「キーワードを入力する」において、UI 部品 t_2 にユーザが入力した値を変数 k に代入している。

4.2 タスク構造の変形

タスクモデル中のイベントの集合とフレームとを対応づけるため、まずタスクの階層構造をなくして平坦な構造にし、さらに、イベントの集合をフレームに対応した部分集合に分割する。

このとき、1 フレームに対応するイベント集合は、次のような構造でなければならない (図 9)。

- イベントの実行順序は、任意個のシステム内部イベント、システム外部イベント、任意個のユーザ内部イベント、ユーザ外部イベント、の順でなければならない。
- システム外部イベントは高々 1 個である。

タスクモデルがこのような構造でないときは、実行順序を入れ換えたり、複数のシステム外部イベント

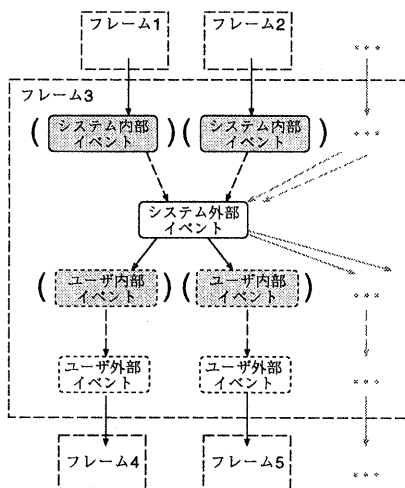


図 9: フレーム内のタスク構造

を 1 個に結合したりしてこのような構造に変形する。

ユーザとシステムの対話は、「システムが必要な内部処理をした後に表示を行い、続いてユーザが操作を行う」ことの繰り返しで構成されと考えられる。上記のタスク構造はこのことを反映している。

5 プロトタイプの自動生成

UI 設計書に基づいてプロトタイプを自動生成する。すなわち、フレームおよび UI 部品の表示や、ユーザとの間の入出力を制御するプログラムを生成する。

プロトタイプ生成の基本方針として、1 フレームに対してプロトタイプの 1 モジュールを生成し、モジュール内は各イベントに対応するプログラムコードを並べることで構成する。モジュール間のデータの授受はグローバル変数を介して行う。

プロトタイプへの変換アルゴリズムは、対象とする実装の枠組ごとに与えられる。以下では例として、WWW を実装の枠組としたときの変換手順を説明する。

WWW 枠組でのプロトタイプ生成 各フレーム f に対して、以下のような 2 つの CGI プログラムを生成する。

- $P_{f,1}$: 必要な内部処理の後、フレームを表示する。
- $P_{f,2}$: ユーザの操作によって起動され、ユーザの入力を変数に代入するなどの処理をする。

$P_{f,1}$ はフレームの入口に当たるイベントからシステム外部イベントまで、 $P_{f,2}$ はユーザ外部イベントから、次のフレームに対応するモジュールの実行開始までを担当する。ユーザ内部イベントはシステムの外で行われるイベントなので、どちらのプログラムも担当しない。

各 CGI プログラムが共有すべきグローバル変数は、ファイルを媒体に使う実装する。

$P_{f,1}$ の生成アルゴリズムは、UI 設計書上で、入口に当たる各イベントから深さ優先探索を行い各イベントに応じたプログラムコードを生成していく。 $P_{f,2}$ の生成アルゴリズムは、各ユーザ外部イベントから探索を行い、後は $P_{f,1}$ の場合と同様である。

各イベントに応じたプログラムコードは以下のようになる。

- システム内部イベント: out の各要素に対する、代入すべき式の評価と評価値の代入。
- システム外部イベント: show, showui の要素である各変数や UI 部品に対する、表示メンバ関数の呼び出し。
- ユーザ外部イベント: out の各要素に対する、代入すべき式の評価と評価値の代入。代入すべき式は通常、UI 部品に対する入力値読み出し関数の呼び出しを含む。

6 タスクモデルの形式的検証

得られたタスクモデルに対し、ユーザビリティに関する性質を満たしているかどうか形式的に検証する。ここでは以下のような手順で検証を行う。

- (1) 検証したい以下のような性質 ϕ を時相論理式で記述する。
 - ユーザはいつでもタスクを中断できる。
 - ユーザは何回でも本を検索できる。
- (2) タスクモデル T の下で ϕ が成り立つ、すなわち $T \models \phi$ であることを、モデルチェッキング [4] などの手法を用いて検証する。

7 タスク図

タスクモデルの記述およびその後の詳細化のために、タスク図という図的言語を導入する。

定義 1 (タスク図) タスク図は、有向グラフ $G = (V, A)$ である。頂点 $v \in V$ は次の 6 つのうちいずれかである。

- 原始タスク
- 並行タスク
- タスク図
- start
- done
- 空頂点

ただし, start, done はそれぞれちょうど 1 個 V に含まれる. start の入次数および done の出次数は 0 である. 各頂点 $v \in V$ について, start から v へのパスおよび v から done へのパスが存在する. □

定義 2 (タスクモデル) タスクモデルはタスク図で与えられる. □

タスク図において, 各頂点は全体のタスクを構成するサブタスクを表し, 辺はその実行順序を表している. start は実行開始点, done は終了点を表す.

タスク図の記述例を図 2 に示す. 角の丸い長方形および楕円が頂点を表し, 矢印が有向辺を表す. 頂点に図を埋め込めることによる階層構造の表現, および並行成分からなる頂点 (並行タスク) の表現は, ステートチャート [2] になっている.

定義 3 (並行タスク) 並行タスクは, 原始タスク, 並行タスク, タスク図を要素とする空でない集合である. □

図中では並行成分を破線で区切って表す (図 2).

階層構造 タスク図や並行タスクを頂点とすることでタスクの階層構造を表現できる. 記述の際は, 頂点や並行成分を表す領域内にタスク図を埋め込んで描いてもよいし, 親図と分けて別の領域に描いてもよい.

定義 4 (タスク, サブタスク) タスクとは, 原始タスク, タスク図, 並行タスクのいずれかとする. また, タスク図 t の頂点 (ただし start, done, 空頂点を除く) および並行タスク t' の成分をそれぞれ t, t' のサブタスクと呼ぶ. □

ここで, タスクとサブタスクの関係は木構造を構成するものとする. つまり, 自分自身を内部に含むタスクや, 互いに要素を共有するようなタスクは存在しないとする.

空頂点 空頂点は, 辺の数を減らして図を見やすくするために用いられる. 図中では, 中が空白の楕円で表される (図 2).

8 あとがき

本研究では, 設計の上流工程から適用できる体系化されたインタラクティブシステムの設計法を提案した.

今後の課題として以下が挙げられる.

(1) タスクモデルの記述におけるガイドライン

適切なタスクモデルを記述するのはどのような記述法を用いようとも熟練を要する作業であり, タスクモデルの詳細化, UI 設計書などに関する雛型を検討する必要がある.

(2) タスクモデルの形式的検証

タスク図に対する形式的検証法について検討する. モデルチェッキング [4] などの検証法の応用を考えている.

(3) 設計支援環境の実現

上記 (1), (2) をふまえ, 本設計法に基づく支援環境を実現する. この支援環境が提供すべき機能として, タスクモデルの記述, 詳細化, および UI 設計情報の指定を支援するための編集機能, プロトタイプ自動生成機能, タスクモデルに対する形式的検証機能などがある.

参考文献

- [1] Dix, A. J., Finlay, J. E., Abowd, G. D. and Beale, R.: *Human-Computer Interaction*, Prentice Hall Europe, 2nd edition, 1998.
- [2] Harel, D.: Statecharts: A Visual Formalism for Complex Systems, *Science of Computer Programming*, Vol. 8, pp.231-274, 1987.
- [3] 池田, 高田, 関: ユーザタスクの形式的記述に基づくインタラクティブシステム設計法の提案, 情報処理学会研究報告, SE122, pp.25-32, 1999.
- [4] Kesten, Y. and Pnueli, A.: Modularization and Abstraction: The Keys to Practical Formal Verification, in *MFCS '98*, LNCS1450, pp.54-71, 1998.
- [5] Moher, T., Dirda, V., Bastide, R. and Palanque, P.: Monolingual, Articulated Modeling of Users, Devices, and Interfaces, in *Design, Specification and Verification of Interactive Systems '96*, pp.312-329, Springer-Verlag, 1996.
- [6] Newman, W. M. and Lamming, M. G.: *Interactive System Design*, Addison-Wesley, 1995.
- [7] 辻野, 井上: 形式的仕様に基づく GUI プロトタイプの自動生成について, 情報処理学会研究報告, HI78, pp.13-18, 1998.