仕様獲得実験システムにおける仕様獲得技法

松沢忠幸　鈴木宏文　町田和広　蔵田全弘*　土田賢省*

日本電気技術情報システム開発（株）　*日本電気（株）ソフトウェア生産技術開発本部

1　はじめに

ユーザインタフェース仕様獲得において，仕様表現が何らかの対話モデル（[Green],[Hartson],[Myers]）に基づいていると，次のような利点がある：(1) 表現能力の形式的限界が明確になる。(2) モデルが目的とするユーザインタフェースにとって自然であれば，意図した仕様を容易に表現できる。(3)（獲得された）仕様の妥当性をモデルに基づいて検証できる。(4) モデルに基づいて必要な対話処理を予測したり，不適切な対話を抑制することができる。また，要求者とユーザインタフェース開発者との間に交換される情報や，それぞれが持つ知識にも注目する必要がある。利用者（要求者）は，例を通じて要求仕様を提示する場合が多い。検証のために開発者から要求者へ示される情報も，画面イメージなど例である場合が多い。従って，ユーザインタフェース仕様獲得を（半）自動化する方法として以下の様なものが考えられる：要求者からユーザインタフェースの例示を受け，その妥当性を判定し，それから一般的な仕様を推論して，その結果を再び例示により要求者に検証させる。この場合，ユーザインタフェースモデルは，妥当性の判定や一般化において重要な役割を果たすことができる。

本論文では，ユーザインタフェースモデルに基づいた例示によるユーザインタフェース仕様獲得の方法について論ずる。以下，ユーザインタフェース仕様の獲得のために利用した対話モデルを説明し，そのモデルに基づいた，例示と一般化による仕様獲得法を説明する。将来の拡張方向と本手法の問題点とについても触れる。

2　ユーザインタフェースモデル

診断型エキスパートシステムにおけるユーザインタフェースのように，機械と人とが協調的に問題解決を行うための対話の機能として，次のようなものが本質的である：機械（もしくは人）が情報不足のため計算（推論）を継続できなくなり，計算再開のため相手の情報を獲得するための機能である。

このような機能に注目した場合，計算過程を監視して，いかなる情報が不足しているために計算が継続できないかを判断できなくてはならない。そこで，機械（人）を" グレイボックス" としてモデル化する。

計算は，機械（人）の内部状態と入力とにより完全に決定されるものとする。外部には，計算が中断した時点毎の内部状態（の表現，以降計算状態と呼ぶ）を見せる。この内部状態から計算の停止理由を推論する。また，内部状態が機械（人）からのメッセージを構成するための原子トークンとなる。トークンを組み合わせて要求元の計算に必要な情報を構成することが対話処理における主たる計算である。また，妥当性のないトークンを拒否したり，脱出処理の制御も対話処理の一部と考えることもできる。

ユーザインタフェースは，対話メディアを通してトークンの取り出し・変換・発信を繰り返すものと見なす。取り出し・変換・発信などは，それぞれ基本対話処理を組み合わせて実現する。従って，ユーザインタフェースを基本対話処理の具体化としてモデル化する。そして，基本対話処理をノードとする木を処理フローといい，この木を獲得することをユーザインタフェース仕様の獲得とみなす。対話処理の逐次制御は，処理フロー上の道（パス）として表現される。以降において，情報を構成する文脈を与える計算状態をマクロな計算状態，トークン単位に処理していく個々の状態を記述する計算状態をミクロな計算状態と呼ぶ。

基本対話処理は，ユーザインタフェース開発者が持つ対話処理知識の抽象化である。基本対話処理には，計画問題における動作記述と同様な構造を与える：(1) 処理記述，(2) その処理を呼び出すための必要条件を記述した前提条件，(3) その処理を実行した場合（文脈に依存せずに）ミクロな計算状態をどのように変更するかを記述した変換記述，及び (4) その処理を実行した後に満足されているべき遷移後条件。

【例】ウィンドウに基づく対話メディアとしては，テキストウィンドウ，メニューなどがある。例えば，メニューは，矩形状表示エリアを画面上に開き，文字列のリストを一覧表示して，マウスで選択させトークンを出力する機能を持つメディアと見なすことができる。基本対話処理としては，メディアの生成・消去・表示停止・表示再開・位置移動・サイズ変更などがある。更に，例えば，メニューに特有な対話処理として，文字列のリストの一覧表示・マウス選択がある。マウス選択処理に付随する前提条件として「メニューに選択項目が設定済みである」こと，変換記述として「ミクロな計算状態にトークンが選択されたことを記録する」ことなどが考えられる。更に，計算状態それぞれをトークン保持用のエリアに取り出す処理群・ウィンドウ／メニューにトークンを表示する処理・メニューからトークンを選択する処理・トークンを計算部に渡す処理なども必要である。

3　一般化手法

利用者からの例示により，処理フローの具体例を獲得する。例示は，基本対話処理を逐次的に具体化することにより行う。基本対話処理は，以下のように具体化される：(1) 利用者が一つの基本対話処理を選択する。(2) 現在の計算状態がその処理の前提条件を充足しているかをチェックする。(3) 充足している場合，前提条件を現在の計算状態で置換する。以降，置換した計算状態の記述を，その処理を呼び出す条件と解釈する。(4) 処理記述に従って（対話インタプリタが）処理を実行し，計算状態を変更する。(5) 遷移後条件をチェックする。実際には，計算状態の表現に対して，充足のチェックや変更を行う。従って，(3) や (4) において計算状態の表現が条件を充足していない場合，その処理は具体化されない。変換は，一般的には，ミクロな計算状態をのみ変更する。従って，一つの処理フロー内では，マクロな計算状態は不変である。

対話が必要になった時点で，既に獲得された処理フローの中に，呼び出し条件（もしくはそれを一般化したもの）を現時点の計算状態が充足しているものがある場合，その処理フローを現在の対話処理の候補として選択する。そして，利用者と協調的に基本対話処理単位でその妥当性を検証してゆく。この場合，処理の呼び出し条件を現在の計算状態が充足できるまで一般化する。もし，このような処理フローが存在していて，利用者が明示的に分岐を指示した場合，候補の処理フローの呼び出し条件と現在の計算状態（新たに例示された処理フローの呼び出し条件となる）を弁別して，それぞれの呼び出し理由を特殊化する。

計算状態は，一階のリテラル（0変数の命題リテラルも含む）の集合（連接と解釈する）として表現する。述語引き数としては，アトムもしくはアトムのリストもしくは変数を取る。実際の計算状態と基本対話処理に付随した前提条件（具体化された基本対話処理の呼び出

A Prototyping System for Dialog Specifications with Acquisition from Examples,

Tadayuki Matsuzawa, Hirofumi Suzuki, Kazuhiro Machida, Kensei Tsuchida, Morahiro Sakata**, NEC Scientific Information System Development, Ltd, * NEC Corporation.

し条件）とから以下のような6種のリテラル集合を生成する：(A)完全に一致するリテラル・(B)実際の計算状態の方が一般的なリテラル（通常は空集合）・(C)呼び出し状態の方が一般的なリテラル・(D)一般化により単一化できるリテラル・(E)実際の計算状態にのみ存在するリテラル・(F)呼び出し状態にのみ存在するリテラル。実際の計算状態においてある呼び出し条件を持つ対話処理を呼び出す（あるいは，具体化できる）ためには，Fが空集合でなければならない。また両者が弁別できるためには，BからFの集合の内少なくとも一つが非空集合である必要がある。一般化は，Dに対して行う。

ある計算状態において，二つ以上の処理フローが呼び出し可能であった場合，いずれか一つを選択するために，上述の6種のリテラル集合の状態から決定されるヒューリスティックな順序関係を利用する。

【例】以下の様な診断知識を考える：

```
q1:a&q2:b=>d1.
q1:a&q2:c=>d2.
...
```

ここで，q:xは，qを発問することにより，xなる回答を得ることを表す。いま，発問q1により，回答aを得て，q2に関する情報を得るために計算を停止したとする。この場合，マクロな計算状態として，例えば，リテラル集合

```
{is_in_query_state,
is_current_node(q2),
is_current_branch([b,c]),...}
```

を持つ（型は無視する）。既に，発問q1の段階で，発問から回答を得て再び推論を行うまでの例示が与えられていた場合，その処理フローの呼び出し条件は，

```
{is_in_query_state,
is_current_node(q1),
is_current_branch([a,...])}
```

なる形式である。もし例示された処理フローを発問q2においても適用するとすると，呼び出し条件を一般化して，

```
{is_in_query_state,
is_current_node(X),
is_current_branch(Y)}
```

なる形にする（X,Yは変数）。更に推論が進行して，結論まで到達したとする。その時点での計算状態は，

```
{is_in_conclusive_state,
is_current_node(r),
is_current_branch([]),...}
```

のような形である。もしこれまでの処理フローが，結論における対話として不適切であるとして，新たに処理フローが例示された場合，その処理フローの呼び出し条件は，この計算状態そのものである。そして，ヒューリスティックな順序付により，結論では新たに例示された処理フローを選択するようになる。（図1参照）

4 議論

本方法の適用可能性として，以下のようなものが考えられる。まず，(1)実装により議論されたように，プロトタイピングの一方法として本方法を利用できる［土田］。(2)現実装では，獲得されたユーザインタフェース仕様をインタプリタが解釈実行している。この場合，個々の処理フローの呼び出し条件の内で，相互に弁別する部分のみが制御にとり意味がある。この部分のみを取り出すことにより，イベントハンドラ群として仕様をコンパイルすることができる。(3)本

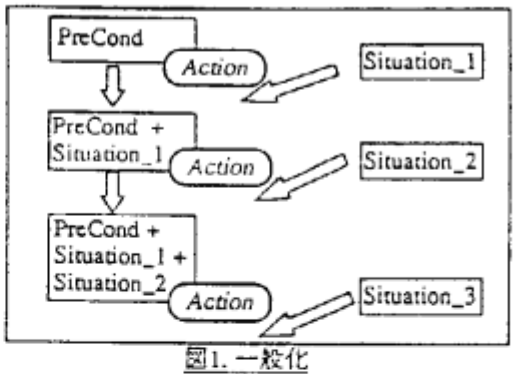

図1. 一般化

方法では，対話メディアとその機能を既存のものとしてそれらを組み合わせて対話処理を構成する。しかしながら，仕様獲得過程で得たユーザインタフェース仕様自身が，類型的な対話メディアとしての機能を持つと考えることもできる。これにより，「ユーザインタフェース部品」を獲得することができる。

一方，現段階では様々な問題点もある。例えば，計算に関する知識やユーザインタフェースに関する知識の質などである。これに対しては，計算に関する知識を模倣的に持ち込むことにより，曖昧要求などを自動合成することができるようになると思われる。また，一般化の程度をあげることにより，より細かい制御を例示から獲得できるようになると思われる。これに関連して，「最小汎化」[Plotkin]が注目される。

5 結論

本報告では，まず，ユーザインタフェースのモデルについて報告した。そしてモデルに基づいた例示と一般化によるユーザインタフェース仕様獲得方法を提案した。

ユーザインタフェース仕様の獲得と実現とは，非常にコストが掛かる。システムの利用し易さの大部分を決定する因子がユーザインタフェース部にあるとすると，システム開発においていかに効率的にユーザインタフェース仕様を獲得するかが重要になる。そのため，ユーザインタフェース仕様化の工程を開発方法論の一部として組み入れることも提案されている[Hartson]。この場合，プロトタイピング（とそれによる検定方法）を併せもつことも重要であるとされている。このような課題を解決する一方法として，本論文で提案したような仕様獲得方法が有効ではないかと思われる。

謝辞

本研究は新世代コンピュータ技術開発機構(ICOT)の委託研究として進められた。御支援頂きました長谷川ICOT第一研究室室長を初めとする方々に深く感謝いたします。また，研究にあたり，御指導下さったNECソフトウェア生産技術開発本部川越課長ならびにC&Cシステム研究所宮下課長に感謝します。

［参考文献］

[Green] Green, M.: A Survey of Three Dialog Models, ACM Transaction on Graphics, Vol.5, No.3, pp.244-275(1986).

[Hartson] Hartson, H.R & Hix, D.: Human-Computer Interface Development- Concepts and Systems for its Management, ACM Computing Surveys, Vol.21, No.1, pp5-92(1989).

[Myers] Myers, B.A.: Creating User Interfaces by Demonstration, Academic Press(1988).

[Plotkin] Plotkin, G.D: A Note on Inductive Generalization, Machine Intelligence, Vol.5, pp.153-163(1970).

［土田］ 土田，他：仕様獲得実験システム，第39回情処全国大会講演論文集(1989).