使いやすい計算機システムのための対象の属性に注目したモデル化枠組

古宇田 フミ子　　近山 隆†
東京大学 情報理工学系研究科，新領域創成科学研究科† *

1 はじめに

利用者インタフェース (UI) は、(a) 多くの CUI のように、単機能のプリミティブを多数提供し、利用者がこれを自由に組み合せることで種々の処理を可能とするものと、(b) GUI のように利用者は与えられたメニューから選ぶことで定型的処理をたやすく実行できるものが知られる。

利用者の目的は種々異なるので、この二つのタイプだけでは満足できないことも起こり得る。(a) タイプでは計算機システム側が提供するプリミティブを組み合せることで処理を実行させるので、組合わせの自由度が増し、きめ細かく制御が可能となるものの、多くの利用者にはやりたい処理をどのようなプリミティブの組合わせで実現できるかを「発見する」のは容易ではない。また、利用者はプリミティブに関する詳細を理解する必要がある。この難しさは利用者インタフェース (UI) で提示するプリミティブがシステムの観点に基づいて抽象化した処理を提示するので、(1) UI が OS 毎に異なる、(2) 個々の処理は比較的単純なので、利用者の望む処理との関係付けや組合せ方が解り難い、(3) 処理プリミティブの識別子は処理の解釈の一つに基づいて付けられるのでその仕様が推察しにくい、(4) 処理プリミティブの入力形式や結果の形式には柔軟性がない、等、に起因する、と考えられる。

UI でシステムの観点ではなく、利用者の知識に基づく処理指示ができるようになれば、計算機システムのことを熟知していなくても使えるようになる。即ち、処理機構の説明をするのではなく「何をするか」という処理目的を記述することで、OS や計算機の形態に依存しない記述が可能となる。このことができれば、比較的容易に処理指示ができ、しかも、(b) タイプのような定型処理に限定されること無く細かな制御も可能となる。

このような条件を満たす新しい UI 構成の一方法は、計算機側で利用者の処理目的の記述を理解できるようにすることである。この実現手段として利用者の処理記述を考慮し得る処理のモデル化を行ない、このモデルに基づいて利用者の処理記述と UI 上の (複数) プリミティブとの対応付けを実現させたい。本稿では、モデル化の枠組について述べる。

*An attribute-based modelling framework for user-friendly computer systems
Fumiko Kouda, Takashi Chikayama†
Graduate School of Information Science and Technology, School of Frontier Sciences†, University of Tokyo

2 モデル化の方針

2.1 モデル化に組み込む要件

利用者が処理指示を行なう場合、処理の途中経過は重要ではない。必要なのは処理の入口と結果であるので、処理は処理の入口の対象と結果の対象との対応関係、と見る。利用者の処理指示と UI のプリミティブでは処理粒度の差があるので、複数の処理を合成する必要もある。

処理では複数の側面が同時に変化する。このことを明示することで利用者の多様な処理の見方に対処する。変化に関わるものは対象の持つ性質なので、処理は対象の属性の変化として捉え、処理に関係する属性をすべて表示した属性間の対応付けとする。

利用者に関しては処理機構の記述は無くしたい。そこで、処理機構を表す属性は無視し、処理機能を表す属性のみに注目する。

2.2 既存のモデル

概念の説明にモデルを立てることで物事の本質を明瞭、簡明、共通に説明できることがある。利用者が目的とする処理の本質を適切に記述する記述モデルがあれば、利用者には計算機処理の理解を助ける点で役立つと考えられる。

処理に関しては処理効率を上げるモデルや処理機構として見るモデルは存在する。利用者のモデル [Kay1995] も存在するが、利用者自体の趣向の分類であり、利用者が処理をどう見るか、の観点ではない。利用者の方向を向いた処理のモデルは存在しない。

解決法の一つとして処理手順を細かく提示するのではなく、処理目的が何であるかを記述することで計算機での処理の指示を可能としたい。この実現のために、UI で提示される処理を 2.1節の要件を満たすような観点から見直した処理モデルを構成する。

将来的には、利用者が自然言語ないし、それに準ずる

形で記述すると、ここで構築する処理モデルを通して、現実に存在するコマンド (列) を (OS の種類の違いに応じて) 返すような構成を目指す。例えば、利用者が「latx 構文の文書を画面に出したい」と記述すると、「latex → dvi2ps → gv」又は「latex → xdvi」のようにプリミティブ列を返す。

2.3 処理モデルの枠組概要

利用者の抱く処理の概念は計算機の処理を抽象的に捉えた [Ramscar1997] ものであるので、 UI が提供する処理を処理機能の面から捉え直すことには意味がある。即ち、 UI が提供する処理を考察範囲とし、利用者の処理目的に対応し得るように処理機構 (how) ではなく処理機能 (what) について考える。更に、簡単のため、処理の入口では処理の対象はファイル上にあるとし、予め、処理は計画され、インタラクティブではないものとする。

処理は対象から別の対象への写像と捉え、対象の処理に関係する属性で記述する。利用者の処理記述の粒度と UI の処理プリミティブのそれは必ずしも一致しないので処理の写像の合成法も明示する。

3 属性に注目した処理の観察と解釈

実際の処理を観察し属性の観点から解釈した。

計算機上の処理の対象に関しては、 (1) 一次元ファイル上で構文が作られる、 (2)$\{1, 0\}$ を基本単位としてすべて表現可能 [渡辺 1992], (3) 処理に応じて構文として見る単位が異なる、 (4) 異なる構文の種類の間には包含関係による半順序関係がある、と見做せる。 (3) と (4) の構文の例は、「単語、文、行、頁」、等が挙げられる。

利用者の概念と計算機上の処理の対象の関係では、処理の対象を抽象化した属性が利用者の処理概念上の対象の属性と対応し得る (co)。

処理の写像に関しては、 (m1) 処理の写像は領域が属性となる。写像の定義域の属性を持つ対象と値域の属性を持つ対象がこの写像の適応となる。

(m2) 処理の写像で扱う属性には、変化、不変化、見ない、不問、がある。写像には不変であるべき属性 (軸属性) がある。軸属性に関係する属性 (隣接属性) が変わることで処理の写像を表す (例、文字'a' は同一で表現法を S-JIS から EUC コードにする)。

(m3) 写像で注目する範囲の外枠 (構文の半順序として) の属性 (枠属性) があり、それより外側の属性は見ない (知らない属性)。同時に内枠の属性 (カプセル属性) もあり、この中の属性は不問 (気にしない属性) となる。

(m4) 処理の属性間の対応には、幾つかのカプセル属性又は枠属性があり、不変化属性の有無により、以下のタイプが存在する。

(I-a) 処理の写像でカプセル属性が変化しない。

(I-b) 処理の写像でカプセル属性が変化する。

(I-b-i) カプセル属性に軸属性が含まれない。

(I-b-ii) カプセル属性に軸属性が含まれる。

(II-a) 処理の写像で枠属性に含まれる属性が変化しない。

(II-b) 処理の写像で枠属性に含まれる属性が変化する。

(II-c) 処理の写像で枠属性に対応する軸属性が存在して、別の枠属性になる。

(m5) 処理の写像では変域と値域は複数の属性が関係し、それぞれ (m4) のタイプの組み合せの写像となる。

(m6) 写像には変化と不変化の属性間に何らかの一対一対応がある場合と一部の属性が消失する場合がある。前者は逆写像を持ち得るので正則 ($f^{-1} \circ f = Id$) と見做せる。写像の合成では、軸属性で繋ぐ場合と、お互いに干渉しない共役 ($g = f^{-1} \circ g \circ f$) な場合がある。

4 処理のモデル化の枠組の提案

処理を属性間の写像と見て $\{Q\}P\{R\}$ で表す。ここで Q は処理の入り口に関係する属性、 R は処理の結果に関係する属性、 P は Q から R への対応関係を表す。 (m2) の処理で不変化と変化属性の関係を $ef(*)$ で、不問と見ない属性を $dn(*)$ で表すと、 $Q = ef(Q) \cup dn(Q)$, $R = ef(R) \cup dn(R)$ となる。

処理の概念と計算機上の構文は対応付け (3章 (co)) できるので、 Q には異なる種類の枠属性 (や軸属性) がある。各々 (m4) をテンプレートとして作られる。

二つの処理 $\{Q_1\}P_1\{R_1\}$ と $\{Q_2\}P_2\{R_2\}$ の合成には以下の関係がある。 (a) P_1 と P_2 の軸属性が存在して同じ場合、 $ef(R_1) \cap ef(Q_2) \neq \phi$ となり、合成可能。 (b) P_1 と P_2 の軸属性が存在して異なる場合、 $ef(R_1) \subset dn(Q_2)$ ならば P_1 と P_2 は影響しないので、 P_1 と同じ軸属性を持つ処理は P_2 と処理順を入れ替え得る。 (c) P_1 と P_2 の軸属性が存在して異なる場合、 $ef(R_1) \cap ef(Q_2) \neq \phi$ ならば処理は一方向性となる。

5 おわりに

利用者には個別化を可能とするような処理のモデル枠組を実際の処理の観察に基づき、特に、処理の写像の部分を示した。本当にこの枠組で十分かの確認が必要となる。これに対して、枠組を固め、プロトタイプを構成し検証する予定である。現在は、 C 言語でプロトタイプに取りかかっている。

参考文献

[Kay1995] Judy kay "The um toolkit for cooperative user modelling," UMUAI4, pp.149–196, 1995.

[Ramscar97] Michael Ramscar et al., "Do We Know ...", CISM1997, pp.429–431, 1997.

[渡辺 1992] 渡辺治 計算可能性・計算の複雑さ入門、近代科学社 1992