利用者の視点に基づく処理記述に向けて

3V−04

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

計算機の利用者インタフェースは(1)処理の抽象化の方向が利用者本位でなく、(2)抽象化の粒度が細か過ぎ、目的を達成するための組合わせ方が分かりにくい、という問題がある。この背景には、インタフェースは種々の処理に適用できるように小さな処理モジュールを提供し、これを利用者が自由に組み合わせることができるように設計されているという特徴があるためである。

処理の粒度が細かいと目的の処理を得るために利用者には知識が必要となる。この欠点を克服する目的で処理の粒度を粗くして予め幾つかの処理を組み合わせ統合化した形で利用者に処理を提供するシステムも存在するが、こちらは、逆に、想定した使い方以外は不可能なので、利用者の求める処理に沿わないことも起こる。

この問題の解決法の一つとして、利用者がシステムで提示する処理インタフェースに合わせるのではなく、利用者の処理の捉え方を理解できるようなミドルウェアを追加して、セマンティックギャップを埋める方法が考えられる。これには利用者の描く処理目的とシステムの処理記述の枠組が必要である。処理対象の処理による変化が属性に基づいて記述できる点に注目し、属性を中心に処理を捉え、これに基づき、実現のための第一歩として、何が必要であるかを例題を基に抽出する。精密化は今後の課題とする。

2　処理の記述の枠組と仮定

2.1　前提項目

処理では処理目的である対象の属性を変える、と理解して以下の枠組で捉える。

(1) 処理の対象(例、ファイル)は属性により記述される。処理動作は入り口の処理の対象の属性と結果の対象の属性との対応関係(写像)とする。処理記述を(処理動作, 入り口の属性, 結果の属性)の3つ組(a-dr)で表す。

(2) 処理の対象の属性は処理動作の対応で変化するものと不変化のものがある。

(3) 処理は組合わせ可能である。処理の合成は(1)の写像の合成として表される。

(4) 利用者は意図する処理目的を何らかの方法で記述する。この記述から処理動作、即ち、処理対象の開始時の属性と結果の属性の対応関係が切り出せるものとする。

(5) 利用者の意図の記述では処理対象のすべての属性を見渡せる訳ではないので、関心のある属性のみについて記述する。

2.2　解決すべき課題

以下の二点が主要な課題と考える。

[1] 属性の捉え方: 利用者の視点から捉えた処理の対象の属性と現在のインタフェースで利用者に提供している処理の捉え方が異なる。即ち、対象を見る視点が異なり、捉える属性が異なる。利用者は処理の意図や目的に基づいた属性を記述する。また、インタフェースで提示する処理は具体的なスキーマ(構造枠組)も扱うので構造に基づく属性も考慮する。

[2] 処理の粒度: 利用者の視点では目的の処理結果を重視する。途中経過、つまり、処理のやり方、は問わない。システムの見方では処理動作はある規則に従って一組の属性を切り出して、この属性について対応する属性に変換する。

この解決のために、[1] 属性の表現法、[2] 処理の表現法、[3] 利用者的見方とシステム的見方との対応付けアルゴリズム、を明確にすることが必要である。

3　処理と処理目的の記述の対応

利用者、システムの何れの視点においても2.1節の(1)より処理は属性に基づいて記述される。これらの関係明確化のために、対象の属性を表す属性空間を導入する。

2.1節の(2)は処理の視点を考慮すると属性の変化に関して見方が変わる。

処理を一つ固定した場合、属性空間では、仮に全知(全てを見通せる)ならば変化する属性の領域と不変化の二種類に分けられる。処理の視点が定まると、関係する属性の範囲が決まり、属性空間の一部を切り取って見るこ

---

*Toward a Description of computer processings from the viewpoint of users

Fumiko Kouda and Takashi Chikayama
Graduate School of Information Science and Technology and School of Frontier Sciences†　of University of Tokyo
3-1 Hongo 7-chome, Bunkyo-ku, Tokyo 113-0033, Japan

とになる。この範囲の中で、処理の対象の属性が変化属性、不変化の属性、不明の属性 (don't know: まったく知らない)、に分けられる。

同じ処理に対し、(利用者の意図を表す) 処理目的に基づいた記述の場合は、変化属性は中心変化 (して欲しい) と見る属性、この属性が変化することで連動して変化した副作用の変化属性、不変化であるべき属性、不問 (don't care: 知っているけれども気にしない) の属性、に分けられる。目的により立場が変わり得る。例えば、副作用の変化を強調したい場合は変化属性と見るが、そうでない場合は不問の属性と見ることもできる。また、切り取られた属性空間の範囲内の属性で、処理目的として考慮外の属性は実際には変化していても、不問の属性となる。

利用者の見方とシステムの処理各々で用いる属性の間で、異なる表現であっても属性同士の異同の対応関係が付けられるものと仮定 (a1) する。すると、処理に関係する属性を変化と不変化の 2 組に分け、不問や不明の属性は場合に応じて都合のよい方の組に入れることで、同じ処理を表すかどうかの判定ができる。利用者の処理目的記述の属性とシステムの処理記述の属性の対応関係がこの判定法で同じになれば、二つの記述は同じ処理を表す可能性がある。

4 粒度差と対応

利用者は「文書を整形し美文書化して紙に出したい。」とする。この記述を、利用者の見方でより具体的にすると、例えば、「与えられた文書を利用者の意図に基づいて整形するために、文章を latex 形式で書いて印刷する」となる。この処理は幾つかの処理の連鎖で記述される。例えば、

[0] 文書を入力する、
[1]latex の構文規則に基づき、利用者の意図を反映するように与えられた文書の指示された個所に latex の規則を埋め込むことで文書を編集し、人間に可読な latex 構文を含んだ形式の文書にする、
[2] その結果の文書を latex の規則に従って、二次元表現形式 (dvi) に変換する、
[3] 二次元表現形式 (dvi) をプリンタの受入形式 (postscript) に形式変換する、
[4] 変換結果の形式の文書 (postscript) を媒体を紙として二次元表示に変換する、

となる。ここで、[0] と [1] は計算機が自動的に行なうのではなく、人手が関わる。

これを 2.1節の三つ組 (a-dr) で表すと、

[0] (編集, ϕ, 文書である)
[1] (編集, 文書である, latex 構文の文書である)
[2] (latex 処理, latex 構文の文書である, dvi 形式文書である)
[3] (ps 化, dvi 形式文書である, ps 形式文書である)
[4] (印刷, ps 形式文書である, 二次元形式化文書である)

となり、前段の処理結果の属性が次の入り口に使われることが分かる。何れの場合も「文書の内容」という属性は不変化の属性として存在するが、ここでは省いた。

利用者的見方を (a-dr) で表すと、

(整形して印刷, 文書である, 意図の形式を持つ文書である)

となり、[1] の入り口の属性と [4] の結果の属性が対応する (図 1)、途中の属性の変化は利用者には知らなくてもよい、複数の処理動作を一つのものと見ている、ことが確認できる。

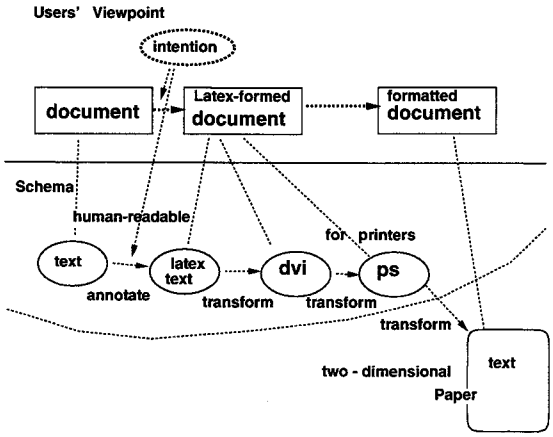

図 1: 属性の捉え方と処理

このことから、以下の方針が得られる。仮定 (a1) の下で典型的な場合、(1) 利用者記述の入り口の属性とシステム処理の入り口の属性が一致するものを探す、(2) 存在すれば、これを始点としてシステムの処理は 2.1節 (3) の合成で可能な組み合わせを作る、(3) 組み合わせ結果の属性と利用者記述の結果の属性の一致を調べる、不一致ならば (2) に戻り繰り返す、ことで対応付けの可能性を判断する。

5 おわりに

現在のシステムの持つセマンティックギャップを埋める目的で、その第一歩として、利用者の処理の捉え方とシステムの処理の記述を属性を中心に捉え、処理を三つ組 (a-dr) で表すことで対応付けの可能性を調べた。属性空間を利用することで処理の粒度の違いを埋めるための方向付けを得た。

今後の課題は、属性空間の精密化、2.2節の課題の明確化にある。