ソフトウェア工学 106－1
（1995．11．17）

ソフトウェアプロセスに連係するユーザインタフェースの組み立て

満田 成紀　鯵坂 恒夫

京都大学工学部

統合型CASE環境におけるUI統合サービスの実現を目的とし，環境と人間の界面となるユーザインタフェース(UI)の設計方針を考える．この方針は，UIの中で管理されるサブジェクトの分類を基本とし，コンポーネントウェア化された環境を提供することで，UI設計者は，ウィンドウの構成部品とツールへの入出力プロセスを独立して設計することが可能となる．構成部品とプロセスの間の関係を明らかにし，この変換サービスをコンポーネント間のインタフェースによって実現することで，UI部品がプロセスに連係することが示される．

An Architecture for User Interfaces Cooperating with Software Processes

MITSUDA Naruki　AJISAKA Tsuneo

Faculty of Engineering, Kyoto University
Yoshida honmachi, Sakyo-ku, Kyoto 606-01, JAPAN

This paper describes an architecture for user interfaces(UIs) to realize UI integration services in a CASE environment. This architecture is based on a classification of subjects managed in UIs. Using this architecture, the UI designer can model processes of tool operation separately from physical structure of UI window. Cooperation between UI parts and processes of tool operations is established by implementing translation services between subjects.

1 はじめに

統合型CASE環境の実現とは，多種多様なCASEツールに対して，それらを一貫して管理する共通のプラットフォームの実現を要求するものである．共通プラットフォーム実現に向かうアプローチとして，CASE環境に対するコンポーネントウェア的枠組みを与えることが考えられる．

環境において一貫性管理の対象となるサブジェクトを階層的に分類[1][2]し，それぞれの階層に対応したサブジェクト管理コンポーネントを用意する．異なる階層のサブジェクト間に関係がある場合は，隣接するコンポーネント間にインタフェースが用意される．この時，個々のツールは利用するサブジェクト定義を各コンポーネントに与えるだけで，サブジェクトの一貫性管理を実現することが可能となるのである．

本研究は，CASE環境のコンポーネントウェア化実現に向けて，特にユーザインタフェース(UI)に注目したものである．まず，UI管理に必要なサブジェクト階層として**UIオブジェクト**と**操作プロセス**を示し，階層内でのサブジェクト管理，両階層にまたがるサブジェクト間関係の管理について考察をすすめることで，UI管理コンポーネントの設計が可能となる．

本研究で扱っているプロセスとは，一般にソフトウェアプロセスと呼ばれ研究対象となっているものとは，その粒度において異なるものである．ソフトウェアライフサイクルやワークフローといったものが扱うレベルに比べてかなり小さく，ツールに対するユーザの入出力操作をプロセスと呼んでいる．しかし，粗粒度プロセス(一般のソフトウェアプロセス)も，細粒度プロセス(本研究のプロセス)から構成的に組み立てられるものであり，その実働にはユーザ操作との連係が欠かせない．

さらに，ソフトウェア開発におけるプロセスというものをツールからみた入出力操作としてのみ捕らえることで，人間による活動をカプセル化し，その曖昧さを排除することが可能になると考える．人間の活動も含めた一様なモデル化を提供することによって，複数のユーザやツールによる協調的活動(＝パフォーマティブ)も，いくつかの操作プロセスにまたがる関係として容易にモデル化できる．

2 UI管理コンポーネントの構成

UIに対して管理コンポーネントを設計するために，まず，UIの提供すべき機能をもとにそこで利用されるサブジェクトを分類する．

現在では，UIはX-Windowのようなウィンドウシステム上で実現されることが一般的となっている．したがって，UIの提供すべき機能とは，ボタンやフィールドといった，ウィンドウシステムによって提供される構成部品を用いてツールに対する入出力操作を行うことである．この時，ウィンドウ構成部品と入出力操作という二つのサブジェクト層を考えることが可能である．これらは独立に管理可能なものだからである．

ユーザ側のサブジェクト層を**UIオブジェクト**とし，ツール側のサブジェクト層を**操作プロセス**とする．その結果，図1に示すように，UIエンジンとプロセスエンジンの二つのコンポーネントが用意されることになる．

2.1 UIオブジェクトモデル

UIエンジンによって，ボタンなどのウィンドウ構成部品の**種類**や**配置**といった物理的属性が管理される．これを定義したものがUIオブジェクトモデルである特に，ボタンやメニュー

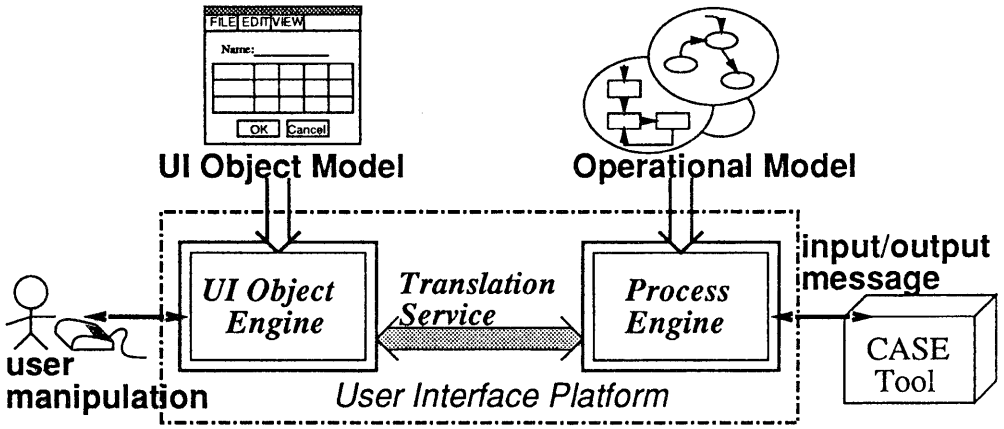

図 1: UI エンジンとプロセスエンジン

のように入力に用いられるものである場合，必ずマウスの位置やクリックに対応した部品状態というものを持つまた，メッセージフィールドやリストボックスのように出力に用いられるものは，それが表現する部品値を持つ．

2.2 操作モデル

プロセスエンジンによって，ツールに対する入出力操作の持つ論理 (= プロセス) が管理される．操作モデルは，ツールに対する入出力の持つ論理を，ツール設計者が自由にデザインしたものである．入出力パラメータをデータ，その値の変遷をアルゴリズムと考えることで，一般のソフトウェア同様のモデル化が可能である．ただし，データの構成 (静的側面) よりも，変遷の過程 (動的側面) に重点を置いたモデルである．

論理の動的側面を表現するためによく用いられるのが状態遷移モデルであり，プロセスモデルの場合もこれを利用できる．状態とイベントという抽象的な概念を持ち込むことで，プロセスの中に潜む因果関係を明確に示すことができる．

2.3 サブジェクトの分類と統合化サービス

コンポーネントウェア的枠組みを用いることで，トースターモデル [3](図 2) における統合化サービスに対しても明確な定義を与えることができる．統合化サービスとは，隣接した階層にまたがるサブジェクト間の関係をもとに，サブジェクトの相互変換を行うサービスである．例えば，本研究で注目している UI オブジェクトと操作プロセスの階層間で為されるサブジェクト変換 (図 1の Translation Service) は，UI 統合に他ならない．

2.4 操作プロセスによる UI オブジェクトの制御

コンポーネントとしては，UI オブジェクトと操作プロセスを別々に管理するだけでなく，サブジェクト変換を行うことで，二つのモデ

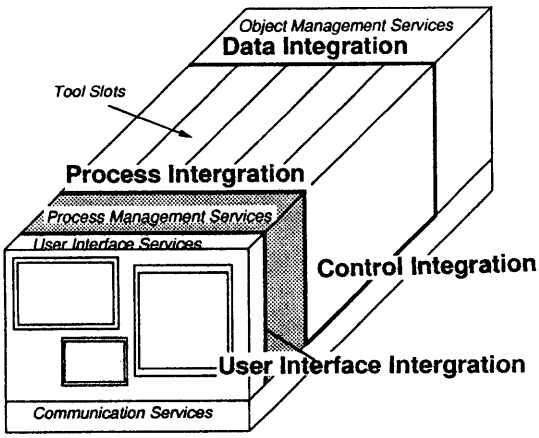

図 2: 統合型 CASE 環境の参照モデル

ルを連結する必要がある．すなわち，それぞれの要素間の対応関係を明確にし，一貫性を管理する必要がある．

これが実現されることによって，プロセスの実働によるサブジェクトの変化が，結果として，UI オブジェクトの属性を制御することになる．

操作プロセスで扱うサブジェクトの内，UI オブジェクトの属性に影響するのは，唯一，ツールに対する入出力パラメータのみである．また，プロセスによって制御される UI オブジェクトの属性は，そのオブジェクトの部品種類に依存する．整数値のパラメータに対して，数字が表示されるかもしれないし，スライディングバーが表示されるかもしれない．

プロセスモデルのイベントの多くは，UI オブジェクトにおける部品の状態変化に対応する．例えば，メニューの状態変化をもとに「メニュー項目 A の選択」を意味するイベントの生成を考えることができる．ただし全てのイベントがそうとは限らない．ツールからの出

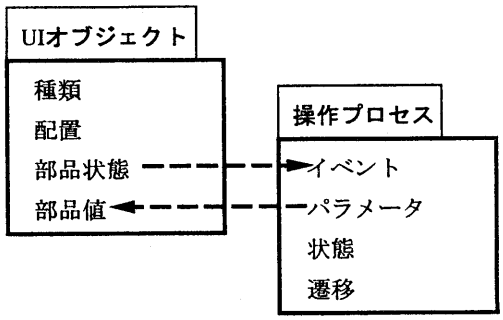

図 3: 両階層内のサブジェクトとその間の変換

力要求もイベントであるし，因果関係を表すために便宜上導入された仮想のイベントもあり得る．

結果として，UI エンジン－プロセスエンジン間で必要な変換サービスとは，

- 入出力パラメータの部品値化
- 部品状態変化のイベント化

の二つであると言える (図 3)．

3 UI オブジェクトの定義

定義すべき対象サブジェクトは，ウィンドウを構成する部品の種類，配置，部品状態，部品値である．これらを独自のスクリプトで定義することも可能であるが，本研究では Ousterhout 氏らによって提唱されている Tcl/Tk 言語 [4] が同等の定義機能を備えていると考え，これを用いて UI オブジェクトの定義を行う．これによる大きなメリットとして，UI オブジェクトにおけるサブジェクト管理が tcl/tk のインタプリタによって実現できるので，UI エンジンを自前で作成する必要がなくなることがあげられる．

Tcl/Tk を用いるうえで注意すべきことは，言語として操作プロセスを記述する能力をも

```
1: set cell {
2: sinh cosh tanh log   exp ON/OFF
3: asin acos atan log10 /   MR
4: sin  cos  tan  abs   *   M-
5:  7    8    9   sqrt  -   M+
6:  4    5    6    ,    +   MC
7:  1    2    3    0    =   C
8: }
9:
10: entry .field -relief sunken -textvariable v
11: frame .frame -width 300 -height 300
12: pack .field .frame -side top -fill x
13:
14: set r [expr 1.0/6.0]
15: for {set i 0} {$i < 36} {incr i} {
16:   set n [lindex $cell $i]
17:   button .frame.b$n -text "$n" \
18:     -command "msg $n"
19:   place .frame.b$n -relx [expr ($i%6)*$r] \
20:     -rely [expr ($i/6)*$r] \
21:     -relwidth $r -relheight $r
22: }
```

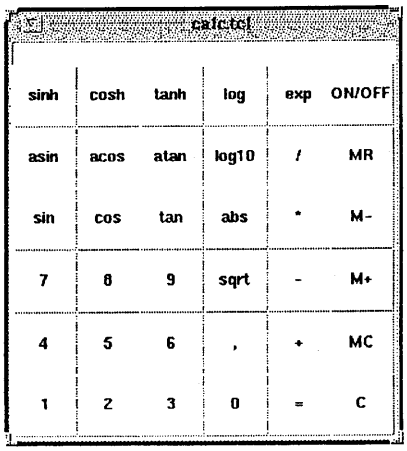

図 4: UI オブジェクトの定義例

持ち合わせていることである．これを使ってしまうと，コンポーネント化した意味が無くなってしまう．そこで UI オブジェクトの定義においてはウィンドウ部品に操作プロセスを割付ける，-command オプションや bind コマンドの使用に制限を持たせることにする．

3.1 定義例

UI オブジェクトモデルの定義例を図 4に示す．ボタンの定義に-command オプションを使用しているが，ここではボタンの状態変化を通知するメッセージを生成することに限定されており，操作プロセスの記述とはなっていない．

4 操作プロセスのモデル定義

操作プロセスは状態遷移モデルによって定義する．定義すべき対象サブジェクトは，状態，遷移，入出力イベント，入出力パラメータの四つである．定義の手法は様々なものが考えられるが，ここでは遷移を羅列することでモデル定義とする．遷移以外のサブジェクトは遷移の構成要素として定義される．

各遷移は図 5に示す文法に従って定義される．イベント名やパラメータ名に変数 (? で始まる) を用いることができ，この場合，入力イベントでマッチした値が出力イベントに使用される．また，出力イベントには向きを明示するためのタグをつけることができる．<T> タグはプロセスエンジンからツール機能に向かうイベントを意味し，<U> タグはユーザに向かうイ

```
<trans_list> := 'trans' '{' <transition>* '}'
<transition> := <origin_state> '[' <input_event> {':' <output_event>} ']' <dest_state>
<input_event> := ('entry' | <message>)
<output_event> := { <direction> } <message>
<direction> := '<' ('T' | 'U') '>'
<message> := '"' <event> <parameters>* '"'
```

図 5: 状態遷移モデルの定義文法

ベントを意味する．これによって，ユーザがUIを介してツール機能と結合されることになる．

4.1 定義例

操作モデルの定義例を図6に示す．状態としてponとpoffがあり，ON/OFFイベントによって状態が切り替わる．pon状態の場合のみ，その他のイベントを受けつける．

5 操作プロセスとUIオブジェクトの間の変換サービス

実際の定義例をもとに，2章で分析したサブジェクト間の変換サービスの実現方法を解説する．

5.1 入出力パラメータの部品値化

操作モデルの定義において，イベントのパラメータに対して変数(?<名前>)を使用することで，入力イベントから出力イベントへのパラメータ渡しが可能となっている．さらに，ユーザに向けての出力イベント(<U>付き)は，パラメータをともなったまま，UIエンジンに対するメッセージとして送信される．もし，UIエンジンがメッセージを解釈した結果，パラメータが部品値に反映されれば，変換サービスが実現されたことになる．

一方，Tcl/Tkにおいて部品値は，グローバルに宣言された変数を用いて管理することが可能となっている．つまり，変数の値の変更が部品値に反映するのである．図4の例の場合，テキストフィールドの部品値がvというグローバル変数を用いて管理されることが定義されている(10行目)．したがって，この管理変数の値を変更するメッセージを発行すれば良いことになる．

もっとも簡単な実現方法は，Tcl/Tkコマンドをそのままメッセージにする方法である．UIエンジンは送られてきたメッセージをそのままコマンドして解釈するように設計すれば良い．図6の例では，writeイベントを受けとった場合には，結果としてset v <パラメータ>というコマンドが実行されるように定義されている(4行目)．

5.2 部品状態変化のイベント化

Tcl/Tkには，ボタンやメニューのような入力用部品に対して，その状態が変化した場合に実行されるコマンドを-commandオプションによって定義することが可能となっている．この機能を使うことによって，部品状態変化のイベント化を実現している．

-commandオプションの引数には必ずmsgコマンドを使用する．msgコマンドは特別に用意した拡張コマンドであり，引数の文字列をメッセージとしてプロセスエンジンに向けて送信する．メッセージを受けたプロセスエンジンは，これをイベントとして解釈することになる．

```
1: trans {
2:     poff ["ON/OFF"]  pon
3:     pon  ["ON/OFF"]  poff
4:     pon  ["write ?x" : <U>"set v ?x" ] pon
5:     pon  ["?m" : <T>"button ?m" ]      pon
6:          [entry] poff
7: }
```

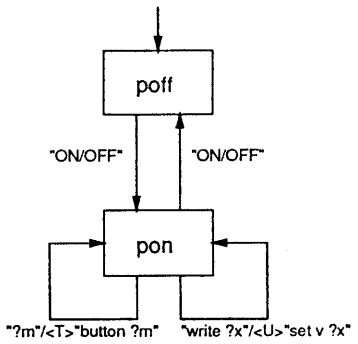

図 6: プロセスモデルの定義例

図 4の例では，ボタンが押された時にボタンの名前をメッセージとして発行することが定義されている (18 行目)．例えば ON/OFF ボタンが押されれば，"ON/OFF"というメッセージが発行され，これは図 6の 2,3 行目で定義されているように，イベントとして解釈される．

6 考察

6.1 プロセス中心の UI 設計

本研究で考案した UI 管理環境を利用することで，UI 設計者はウィンドウ構成部品とツール操作プロセスを独立に設計することが可能になる．これは従来の部品中心の設計とは異なる，プロセス中心の設計が可能であることを意味する．

部品中心の設計とは，まずツールの入出力のために必要と思われる部品を配置し，その部品が使用された時に実行すべき機能を部品に割りつける方法である．ここで問題となるのは，機能間に依存関係がある場合である．つまり，機能の実行順序が想定されている場合である．あくまで部品が主なので，実行順序を守るためには，ツール機能の中に想定順序通りかチェックしこれを解決 (無視，つじつま合わせ等) するコードを埋め込まなくてはならない．

一方，プロセス中心の場合，まず始めにツール機能が想定する実行順序を操作プロセスとしてモデル化することになる．操作モデルが主となることで，部品の使用に関しては，このモデルに記述されている部品使用しか認識されないことになる．また，機能の実行に関しては，想定順序通りの実行であることが保証される．

6.2 操作モデル同士の協調

イベントと状態の概念を持ち込んだことによって，操作モデル間にまたがる制約をイベントの伝播として記述することが可能となると考える．具体的には，二つの操作モデルにまたがるイベントを定義することで，操作モデル同士の協調が定義されるのである．これはすなわち，あるツールに対する操作イベントが，別のツールに対する操作イベントを引き起こすことであり，複数ツールの同期を実現していることになる．さらに連結された操

作モデル群を協調させることで，人間同士の協調作業の実現へとつながると考えられる．

複数の操作モデルを対象とした管理を考える時に注意すべきこととして，各モデルごとに定義における視点や粒度が異なる場合があることである．この時，サブジェクト間に潜む関係を明らかにしておく必要がある．考えられるのは，イベント－イベント間，イベント－状態間，状態－状態間の関係であり，例えば，イベント－イベント間には，構成関係(複数の副イベントへの分割)，順序関係，排他関係が考えられる．

7 おわりに

統合型 CASE 環境で管理されるサブジェクトを分類することで，環境のコンポーネントウェア化が可能となる．本研究は特に UI の管理に注目し，ウィンドウの構成部品とツールに対する操作プロセスを独立して管理することを可能とするシステムを構築した．サブジェクト間の関係を明らかにし，この変換サービスがコンポーネント間のインタフェースによって実現されることで，UI 部品がプロセスに連係することを示した．

今後は，操作プロセス間の協調を基に，より粒の大きなプロセス片を構成的に組みあげていくことで，一般にソフトウェアプロセスと呼ばれるレベルのプロセス管理を実現することを目標としている．

参考文献

[1] IEEE Computer Society's Task Force on Professional Computing Tools. "A Standard Reference Model for Computing System Tool Interconnections". Trial-Use Standard P1175, IEEE, 1991.

[2] Naruki Mitsuda, Tsuneo Ajisaka, and Yoshihiro Matsumoto. "A Semantic-Directed Graph Editor on PCTE". In *JCSE '93*, 1993.

[3] ECMA and NIST. "Reference Model for Frameworks of Software Engineering Environments". Technical Report ECMA TR/55 2nd Edition, NIST Special Publication 500–201, December 1991.

[4] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley Publishing Company, 1994.