知識型設計方法論に基づくインタフェース設計法の形式化と設計支援システムの構成†

木下哲男†† 岩根典之††
菅原研次††† 白鳥則郎††††

高度情報処理システムを実現するための重要な構成要素として，ヒューマンインタフェースが挙げられる．一般に，優れたヒューマンインタフェースを設計できるのは，豊富な経験を持つ熟練した設計者が設計を行った場合に限られることが多い．これは，利用者の要求をヒューマンインタフェースに反映させるための効果的な手段，あるいは熟練した設計者の持つ様々な設計知識を有効に利用するための手段が与えられていないことによる．このような問題を解決するために，本論文では，利用者の要求や熟練した設計者の設計知識を効果的に活用する知識型設計方法論の枠組みに基づいて，ヒューマンインタフェース設計のための知識モデル，およびそれを用いたヒューマンインタフェース設計方式を提案する．本方式では，ヒューマンインタフェース設計過程をシステムの機能設計過程とは独立にモデル化する．そして，その設計過程に含まれる各設計ステージが，利用者の要求や設計仕様などの知識で与えられる知識モデルとして表現され，また，各設計ステージを連結する設計プロセスが，熟練した設計者の持つ種々の設計知識で与えられる知識モデル間のマッピングとして表現される．こうして，要求仕様定義からヒューマンインタフェースの実現に至る設計過程が，マッピングを用いた知識モデルのインタラクティブな変換系列としてモデル化される．このようなモデル化によって，利用者の要求が効果的にヒューマンインタフェースに反映され，同時に熟練した設計者の知識を有効に活用した設計過程によりヒューマンインタフェースが半自動的に生成できる．さらに，本論文では，本方式に基づいて試作されたヒューマンインタフェース設計支援システムと設計例について述べ，その有効性を示す．

1. はじめに

コンピュータシステムの提供する機能とシステムの利用者の境界面となっているヒューマンインタフェース（以下，インタフェースと略記する）は，高度情報処理システムを実現するための重要な要素として認識されているが，優れたインタフェースを設計することは，次のような理由から極めて難しい問題となっている[12]．すなわち，従来のインタフェース設計は，目的とするシステムの機能の一部としてインタフェースの設計仕様が定義され，システムが提供する機能との密接な関連のもとに設計が行われてきた．これは，システムの機能設計を主体とするインタフェース構成手法であり，設計目標となるインタフェースに対して利用者の真の要求を反映させることが困難となる[2)~8)]．一方，近年，インタフェースを構成する部品要素とその直接的な操作機能を，利用者，あるいは設計者に提供することによってインタフェースを構築する手法が提唱されている[9)~11)]．これは，利用者の視点を重視してインタフェースを設計する一手法となっているが，種種の要求に応じて多数の部品を組み合わせるための方法論が確立されておらず，また利用者自身が設計を進める際の設計知識なども明示的に与えられていない．上述した2つの設計手法は，それぞれ異なる立場からインタフェースの構築を目指すものである．しかし，いずれの場合も，現状では，利用者の持つ様々な要求を，設計仕様，およびそれに基づいて実現されたインタフェースに十分反映させるための効果的な手段，また，熟練した設計者の経験に基づく設計知識を有効に利用する手段などがほとんど与えられていない．そのため，優れたインタフェースが構築できるのは，熟練した設計者が設計を行った場合に限られることが多い．したがって，インタフェース設計の高度化のための課題は，1)利用者の要求を設計目標となるインタフェースに効果的に反映させるための手段，および，2)熟練した設計者の持つ高度な設計知識を有効に利用するための手段，をどのように与えるか，という問題に帰着される．

インタフェース設計に関する上記の問題を解決するために，本論文では，知識型設計方法論と呼ぶ枠組み

† Formalization of the Interface Design Method and Construction of the Design Support System Based on the Knowledge-based Design Methodology by TETSUO KINOSHITA, NORIYUKI IWANE (Systems Laboratory, OKI Electric Industry Co., Ltd.), KENJI SUGAWARA (Faculty of Information Engineering, Chiba Institute of Technology) and NORIO SHIRATORI (Research Institute of Electrical Communication, Tohoku University).

†† 沖電気工業(株)総合システム研究所
††† 千葉工業大学情報工学科
†††† 東北大学電気通信研究所

に基づいたインタフェースの設計支援方式を提案する．本方式では，インタフェース設計のための知識モデルを提案し，これに基づいて利用者の要求知識，インタフェースの要求仕様，および設計仕様が明確に表現される．また，本方式では，複数の設計ステージとそれを連結する設計プロセスによって構成されるインタフェースの設計過程が，システムの機能設計とは独立した設計タスクとしてモデル化される．そして，各設計ステージが上記知識モデルとして表現され，また，各プロセスが，熟練した設計者の持つ種々の設計知識によって定義される知識モデル間のマッピングとして表現される．こうして，要求仕様定義からインタフェースの実現に至る設計過程は，マッピングによって連結された知識モデルの変換過程として形式化される．このようなモデル化により，利用者の種々の要求が，知識モデルを媒体として設計対象のインタフェースに十分反映され，同時に，熟練した設計者の持つ設計知識が，知識モデル間のマッピングとして設計過程の中で有効に利用される．さらに，本方式に基づいてインタフェース設計者の設計作業をインタラクティブに支援することによって，利用者の要求に適合したインタフェースが柔軟に，かつ半自動的に生成できる．以下，2章では本方式の基礎となる知識型設計方法論の概要を述べる．3章では，はじめにインタフェース設計のための知識モデルを提案し，次にそれを利用した知識型設計方法論に基づくインタフェース設計方式を提案する．そして，4章では本方式に基づくインタフェース設計支援システムを試作し，設計例を通してその有効性を示す．

2. 知識型設計方法論

高度な設計支援システムの実現を目指して多くの研究が行われている．しかしながら，熟練した設計者の持つ種々の設計知識を効果的に利用するための方法論を与える研究は極めて限られている[12)~21)]．著者らによって提案された知識型設計方法論[1)]は，設計型タスクで利用される種々の知識を設計過程において統一的かつ効果的に利用するための枠組みである．

一般にシステム設計過程は，**図1**に示すように，多くの設計ステージと設計プロセスから構成される．知識型設計方法論では，以下の2つの基本要素によって設計過程をモデル化する．すなわち，

［A］ 設計ステージを知識モデルとして表現する．

［B］ 設計プロセスをマッピング（写像）として表現する．

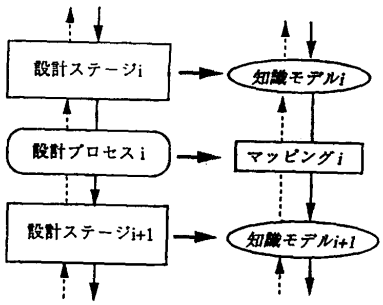

図1 知識型設計方法論の枠組み
Fig. 1 Framework of the knowledge-based design methodology.

ここで，知識モデルは，利用者要求／設計仕様／設計部品情報／設計状態情報など，設計対象や設計問題によって与えられる知識（問題依存型知識）によって定義される．また，マッピングは，その設計ドメインの経験的知識／設計規則／基本原理などの設計知識（領域依存型知識）によって，1つの知識モデル上，あるいは2つの知識モデル間において定義される．こうしたモデル化により知識型設計方法論に基づく設計過程は，マッピングによって連結された知識モデルの変換系列として形式化される．また，本設計方法論では，設計者がマッピングの過程にインタラクティブに関与することにより，柔軟性の高い設計形態を実現する．

3. 知識型設計方法論に基づくインタフェース設計

3.1 ジェネリックファンクションとジェネリックオブジェクト

高度インタフェースの実現においては，まず，利用者の持つ要求（要求タスク）を抽出し，それを設計対象となるインタフェースに効果的に反映させることが重要である．そこで，本方式では，インタフェースに関する要求を次の3種類の知識（インタフェースの基本要求知識）を導入し，これに基づいて利用者の要求を抽出・整理する．

（1） インタラクション：利用者とシステムがインタフェースを介して行う様々な相互作用（操作，応答）に関する知識．

（2） プレゼンテーション：利用者がインタフェース上で直接的／間接的に操作／観測が可能な対象の形態や構造に関する知識．

（3） アクティベーション：利用者に提供されるサービス機能の実体となるシステムの内部機能モジ

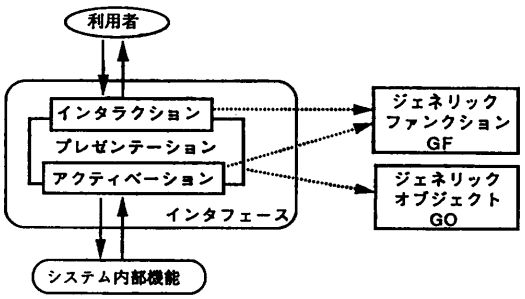

図 2 インタフェースの基本知識と GF/GO
Fig. 2 Basic knowledge for interface and its relations to GF and GO.

ュール（サービス機能実体）との相互作用に関する知識.

例えば,「コマンドメニューから処理コマンドを選択する」という要求タスクは,“処理コマンドの読み込み”がインタラクションとして,“コマンドメニュー”がプレゼンテーションとして抽出される. このようにして, 利用者の要求は, 上記3種類のインタフェースの基本要求知識の集合として抽出される.

上述したインタフェースの基本要求知識の性質から明らかなように, インタラクションとアクティベーションはインタフェースが備えるべき機能に関する要求を, プレゼンテーションはインタフェースの構成要素に関する要求を表現するものとなっている. そこで, インタフェースの基本要求知識として抽出・整理された要求は, インタフェースの機能とその構成要素という観点に基づいてさらに抽象化を行い, 形式化された要求仕様の知識として記述することができる. そのための手段として, 本論文では,

- ジェネリックファンクション GF (Generic Function : 汎化機能)
- ジェネリックオブジェクト GO (Generic Object : 汎化要素)

という, 要求仕様の知識を形式的に表現する2種類の論理部品要素を導入し, これによって要求知識レベルから要求仕様知識レベルへの抽象化を行う. **図2**は, インタフェース基本知識と, GF, および GO の対応関係を示している. 以下, GF と GO について説明する.

GF は, 利用者, あるいはシステムの論理的な振舞いを表す機能部品である. GF は, インタラクションに対応した G_IN, G_OUT, アクティベーションに対応した G_ACT, および, 要求知識を構造化するための機能 G_CTL という4種類のクラスに分類され, それぞれ**図3**に示すような概念階層 (is_a) 木 T_{GF} として表現される. 一方, GO は, プレゼンテーションとして与えられるインタフェースの構成要素に対応する部品である. GO は, 視覚的に観測可能 (Visible), 間接的に観測可能 (Internal), 論理的に存在 (Logical), 物理的に存在 (Physical) という, 部品の持つ性質に基づいて分類した, V&L, V&P, I&L, I&P という4つのクラス, および要求仕様記述で使用されるデータを表す G_Data クラス, という5種類のクラスによって構成され, それらは GF と同様に, **図4**に示すような概念階層木 T_{GO} として表現される.

T_{GF} と T_{GO} の各ノードでは, それぞれ論理部品の特性が定義され, 各概念階層木の構造から明らかなように, 各階層木の上位レベルは下位レベルに比べてより一般的な概念を表している. また, T_{GF} と T_{GO} のリーフ（最下位ノード）は, インタフェースを構築する際に使用される具体的な部品に対応付けられた論理部品となっている.

本方式では, 要求の表現単位を素要求仕様知識 r_i $(i=1,2,\cdots,n)$ と呼び, 各 r_i は, T_{GF} と T_{GO} から選

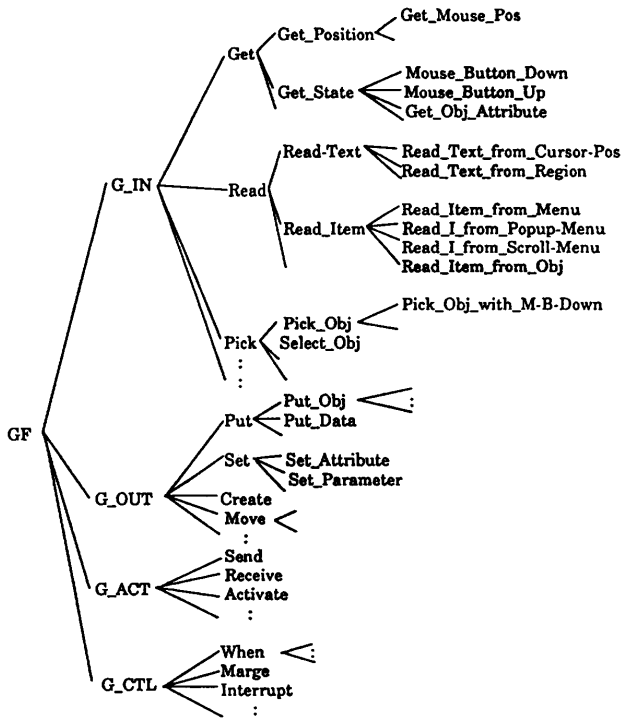

図 3 GF 概念階層木 T_{GF}
Fig. 3 Conceptual tree of GF; T_{GF}.

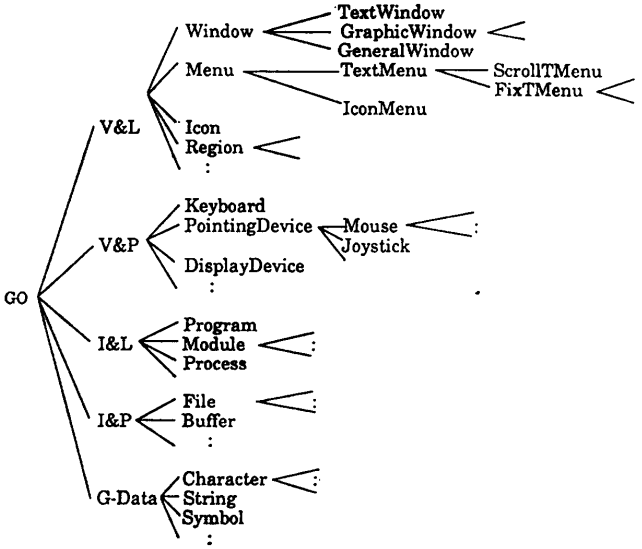

図 4 GO 概念階層木 T_{GO}
Fig. 4 Conceptual tree of GO; T_{GO}.

択される一対の GF と GO の組として表現される．また，要求仕様に関する知識Rは，これら素要求仕様知識の集合 $\{r_i\}$ として表現される．例えば，前述したコマンドメニューからコマンドを選択するという要求では，“処理コマンドの読み込み” の概念に対応した “Read” という GF，および，“コマンドメニュー” の構成概念に対応した “Menu” という GO を選択することによって，素要求仕様知識が構成される．以下，3.2 節では，素要求仕様知識を表現する知識モデルを提案し，3.3 節では，素要求仕様知識の集合として与えられる要求仕様表現のための知識モデル，および設計仕様表現のための知識モデルを定義してゆく．

3.2 素要求仕様知識モデル

3.1 節で導入した素要求仕様知識の構成要素は，インタフェースの論理部品となる GF と GO である．本節では，これら論理部品を形式的に記述する2種類の知識モデルを提案すると共に，その具体的な表現形式を与える．

（1） I-Model (Interaction-Model)

I-Model は，インタフェースを介して行うタスクの機能に関する仕様を，GF を主体として記述するための知識モデルである．I-Model は，Iモデル記述 (IMD) の集合であり，各 IMD は，GF によって規定されるインタフェースの機能要素を表すIモデル要素 (IME) の系列として定義される．

$$\text{I-Model} = \{\text{IMD}^j \mid j = 1, 2, \cdots, m\}$$

$$\text{IMD}^j = \{\text{IME}^j_i \text{(以下 IME}i \text{と略記)} \mid i = 1, 2, \cdots, n\}$$

$$\text{IME}i = \langle \text{GF}i, \text{PM}i, \text{GIR}i \rangle$$

ここで，GFi は，IMEi で選択された GF の識別名，PMi は，GFi がアクセスする P-Model へのポインタである．また，GIRi は，以下に示すような IMEi の詳細属性情報である．

GIRi=〈Task_Group_ID, Pre_IME,
Post_IME, GO_Argument,
Activity_Mode, Local_Proc_Time,
Total-Proc_Time, Side_Effects,
Interrupt_Type & Cue,
GF_Class_Own_Attributes〉

ここで，Task_Group_ID は IMEi が属するタスクグループ，Pre_IME，および Post_IME は IMEi の前後に位置する IME，Go_Argument は GFi が PMi との間で交換するパラメータ情報，GF_Class_Own_Attributes は GFi として指定された GF の固有の属性群を表す．また，Activity_Mode は GFi の処理モード，Local_Proc_Time は IMEi の処理時間，Total_Proc_Time は IMEi に至るまでの累積処理時間，Side_Effects は IMEi の実行に伴う副作用，Interrupt_Type & Cue は IMEi における割込み制御情報を表す．

（2） P-Model (Presentation-Model)

P-Model は，インタフェースを介して行うタスクの表現形態や構成に関する仕様を，GO によって記述するための知識モデルである．P-Model は，Pモデル記述 (PMD) の集合であり，各 PMD は，GO によって規定されるインタフェースの構成要素を表すPモデル要素 (PME) の集合として定義される．

$$\text{P-Model} = \{\text{PMD}^j \mid j = 1, 2, \cdots, k\}$$

$$\text{PMD}^j = \{\text{PME}^j_i \text{(以下 PME}i \text{と略記)} \mid i = 1, 2, \cdots, l\}$$

$$\text{PME}i = \langle \text{GO}i, \text{IM}i, \text{GPR}i \rangle$$

ここで，GOi は，PMEi として選択された GO の識別名，IMi は，GOi を参照している IMD 群へのポインタ集合である．また，GPRi は，PMEi の属性で次のように与えられる．

GPRi=〈Task_Group_ID, Super_PMD,
Sub_PMD, Adj_Rels,
GO_Own_Attributes〉

ここで，Task_Group_ID は GOi が属するタスクグループ，Super_PMD は GOi が含まれる PMD, Sub_PMD は GOi を構成する PMD, Adj_Rels は GOi に隣接する GO との相互関係，GO_Class_Own_

Attributes は GO*i* として指定された GO の固有の属性群を表す．

（3） 素要求仕様知識の表現形式

I-Model と P-Model に対する表現形式を定義する．厳密な表現形式は，各々のモデルの構成要素に対応したスロットを持つフレーム型知識表現を用いるが，本論文では，フレームとして表現される各モデルの構成要素の内容を簡潔に表現するために，**図5**（a）に示す知識記述テンプレートを用いる．また，特に I-Model では，より簡潔な表現として図5（b）に示す文表現形式を用いる．

ここで，利用者から，**図6**に示すインタフェースのイメージを利用するタスクに関する要求が与えられる場合を考える．このインタフェースのプリミティブメニューに関する要求を P-Model によって記述した例が**図7**（a）である．また，このインタフェースにおいて，コマンドメニューから “Select” コマンドを選択し，次にプリミティブメニューからアイコンを選択し，その結果としてウィンドウ Win1 上にアイコンのコピーを張り付ける，というタスクに関する要求の一部を I-Model によって記述した例が図7（b）である．

Statement	IME-ID
	Task-Group-ID
GF/ Class	Type
GF-Own-Attribute	
PM-ID	GF-Arguments
Activity-Mode	
Interrupt-Type&Cue	
Local-Proc-Time	Total-Proc-Time
Pre-IME-ID	Post-IME-ID

（1）Iモデル記述用

GO / Class	PME-ID
	Task-Group-ID
Super-PMD	Sub-PMDs
IM-ID [GO-Arguments]	
GO-Own-Attribute	
Adjacency Rels	

（2）Pモデル記述用

（a） モデル記述用テンプレート

（a） Templates for describing knowledge model.

文表現形式： **GF** *GO* prep [GF's Arg.] prep [GO's Arg.]

<Notation>
Bold String : Generic Parts (GP)
Italic String : Generic Object (GO)
Bracketed String : Arguments for GP/GO
Lowercase String : Preposition (to be ignored)

例： **Get** *Cursor-Position* **from** [Mouse] **on** [Window1]

（b） 文表現形式（I モデル用）

（b） Statement form for I-Model.

図5 素要求仕様知識表現形式

Fig. 5 Representation forms for primitive requirements knowledge.

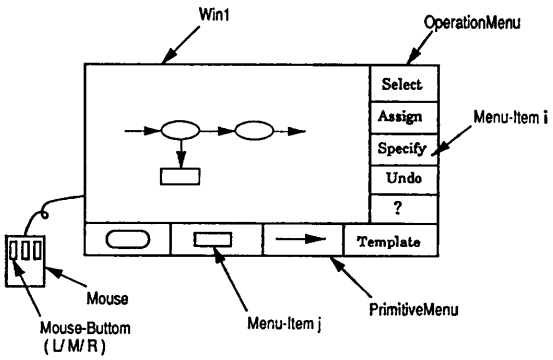

図6 インタフェース設計の例題（要求イメージ）

Fig. 6 Example of interface design (Image of user's requirements).

Primitive-Menu / IconMenu	PME-002 Ex.1
RDF-Window	Icon1 Icon2 Icon3 Icon4
IMDs :	
Alignment : Holizontal Num-of-Item : 4 Item-List : (Icon1 Icon2 Icon3 Icon4) Title : nil	Size : BorderWidth : MasterView :
((Upper Win1))	

（a） Pモデルによる要求仕様知識の記述例

（a） Example of requirements specification knowledge of P-Model.

```
IMD21** "Task Ex.1"
  When [Operation = 1]
    When [Primitive = 1 | 2 | 3]
      Create Icon-Object from [Primitive] on [Win1]
      Show Icon-Object on [Win1]
      Move Icon-Object on [Win1]
      Put Icon-Object at [Mouse-Position] on [Win1]
      Exit

    When [Primitive = 4]
      Put PopUpMenu1 on [Win1]
      Read Template from [PopUpMenu1]
      Show Template-Object on [Win1]
      Move Template-Object on [Win1]
      Put Template-Object at [Mouse-Position] on [Win1]
      Exit
```

（b） Iモデルによる要求仕様知識の記述例

（b） Example of requirements specification knowledge of I-Model.

図7 要求仕様知識の記述例

Fig. 7 Example of requirements specification knowledge.

3.3 インタフェース設計過程のモデル化

本節では，知識型設計方法論に基づくインタフェース設計過程の枠組みを提案する．同時に，そこで利用される2種類の知識モデルを提案する．これらの知識モデルは，3.2節で提案したI-Modelと P-Modelによって定義される．

(1) インタフェース設計過程の切り出し

知識型設計方法論に基づくシステム設計 Dsの枠組みを図8に示す．Dsは，複数の設計タスクから構成される．Dsでは，まず要求知識獲得タスクにおいて，利用者から設計対象システムに関する様々な要求（要求知識）を収集・獲得する．このレベルでは，システム機能やその利用形態に関するものなど，様々な要求知識が混在している．こうした形式化されていない要求知識を表現するのが初期要求知識モデルである．初期要求知識モデルに基づいて設計目標システムの要求仕様／設計仕様が定義され，設計が進められる．Dsでは，以降の設計を，システム機能設計タスクとインタフェース設計タスク Diに分割し，この Diを知識型設計方法論に基づいてモデル化する．このように，インタフェース設計タスクをシステム機能設計タスクから分離し，インタフェースを独立した設計対象として扱えるのは，インタフェースとシステム機能の接続知識が，初期要求知識モデルからアクティベーションとして抽出され，GFと GOを用いた素要求知識として表現されることに依っている．すなわち，アクティベーションの機能は GFにマッピングされ，さらにインタフェースのプロトタイプ生成時に具体的なプログラム部品に置換される．こうしたプログラム部品の実体として，システム機能設計によって与えられるプログラムモジュールを割り当てることにより，インタフェース表層での論理機能とシステム機能が接続される．

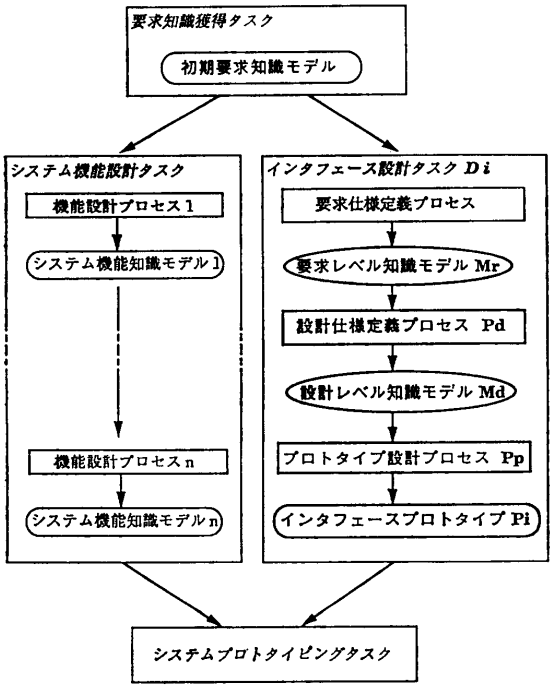

図 8 システム設計タスク Ds
Fig. 8 System design task; Ds.

(2) インタフェース設計過程の構成と知識モデル

インタフェース設計タスク Diにおける設計過程は，知識型設計方法論に基づいて次のようにモデル化される．まず，初期要求知識モデルからインタフェースの要求仕様の知識Rが抽出され，要求レベル知識モデル Mrによって形式的に記述される（要求仕様定義プロセス）．ここで，Mrは，

$$Mr=\langle \text{I-Model, P-Model} \rangle$$

として定義される．すなわち，3.1節で述べたように，Rは素要求仕様知識の集合 $\{r_i\}$ によって与えられ，各 r_i は3.2節で定義した I-Model および P-Modelによって表現される．したがって，要求知識の詳細度に応じて T_{GF}, T_{GO} (GF/GP 知識）から適切な GFと GOを選択し，素要求仕様知識を I-Modelと P-Modelとして記述することにより，Rに対する知識モデル Mrが生成される．

Mrで選択された論理部品 GFと GOが T_{GF} と T_{GO} の中間ノードの場合には，詳細な論理部品の選択などを行うことによって，要求知識の補完／抽象化／詳細化が可能である．こうした Mrの要求仕様知識の洗練によって，設計レベル知識モデル Mdを生成するのが設計仕様定義プロセス Pdである．本プロセスにおいて，要求知識の詳細化の作業を積極的に支援することにより，要求仕様設計者の負担を軽減する．Pdで利用される Mdは，Mrと同様に，

$$Md=\langle \text{I-Model, P-Model} \rangle$$

として定義されるが，GFや GOとして割り当てられる論理部品が，T_{GF} と T_{GO} のリーフノードとなっている点が Mrと異なる．なお，Pdの処理は，Mrの内容を洗練する知識（Mr-Mr マッピング，Mr-Md マッピング）を，設計者がインタラクティブに利用することによって行われる．

(3) プロトタイプの生成

Mdで与えられる設計仕様に基づいて，インタフェースのプロトタイプ Piを生成するのがプロトタイ

プ設計プロセス Pp である．ここでは，まず，Md の GF と GO によって指定された機能と構造（形態）を持つ論理部品 GP (Generic Parts : 汎化部品) を選択することにより，いったん Md を GP を用いた表現（ジェネリック知識モデル）に変換する．次に，この GP に対応して作成された具体的なプログラム部品 OP (Operational Parts : 動作部品) の集合から，Md の実現に最適な OP を選択することによって Pi を生成する．なお，Pp では，Md を，GP, および OP による表現に変換する知識 (Md-Pi マッピング) を利用してインタラクティブに設計を行う．これによって，要求仕様知識に基づく部品の選択／割り付けなどの設計作業を代替することができると同時に，部品検索の手間や部品割り付けにおける誤りなどを大幅に軽減することができる．

一方，インタフェース単体の動作の確認は Pi を用いて行われる．そして，Ds のプロトタイピングタスクで，システム機能設計タスクの設計結果と Pi を統合して設計対象システムのプロトタイプを生成し，システム全体の機能的な検証が行われる．

4. インタフェース設計支援システムの試作

4.1 システム構成

本論文で提案した方式に基づくインタフェース設計支援システムを，Smalltalk-80 上で試作した．本システムは，3.3 節で与えたインタフェース設計タスク Di の設計仕様定義プロセス Pd とプロトタイプ設計プロセス Pp を実現したインタフェース設計者用のシステムである．本システムの構造を**図 9** に示す．

モデル設計支援モジュール MDM は，要求仕様知識 Mr を格納した Mr 知識ベース (Mr-KB) を受け取り，GF/GO 知識を格納した GF/GO 知識ベース (GF/GO-KB)，および，Mr-Mr/Mr-Md マッピング知識を格納した Mr マッピング知識ベース (Mr-Map-KB) を用いて，Md 知識ベース (Md-KB) を生成する．次に，プロトタイプ設計支援モジュール PDM では，この Md-KB を受け取り，OP 知識ベース (OP-KB)，および Md-Pi マッピング知識を格納した Md マッピング知識ベース (Md-Map-KB) を用いて，Pi を生成する．

現在，MDM と PDM の設計者用インタフェースは Smalltalk-80 の環境とブラウザを流用し，マッピングを実行する推論エンジンは Smalltalk-80 上のルールベースシステム[22]を利用している．なお，本試作では，GO を Smalltalk-80 のオブジェクトクラスに，GF をそのクラスのメソッドに対応付けることによって GP と OP を一体化している．したがって，PDM では，GP を省略して OP のみを提供し，Md-Pi マッピングも，Md(GF, GO) から Smalltalk-80 の MVC モデルへの直接的変換 (MVC 変換) として実現している．こうした実装上の手法は，本論文で与えた知識モデルの定義やその利用形態に影響を与えるものではない．また，Pi は MVC 構造を持つ Smalltalk-80 のプログラムとして生成され，Smalltalk-80 の環境で実行される．

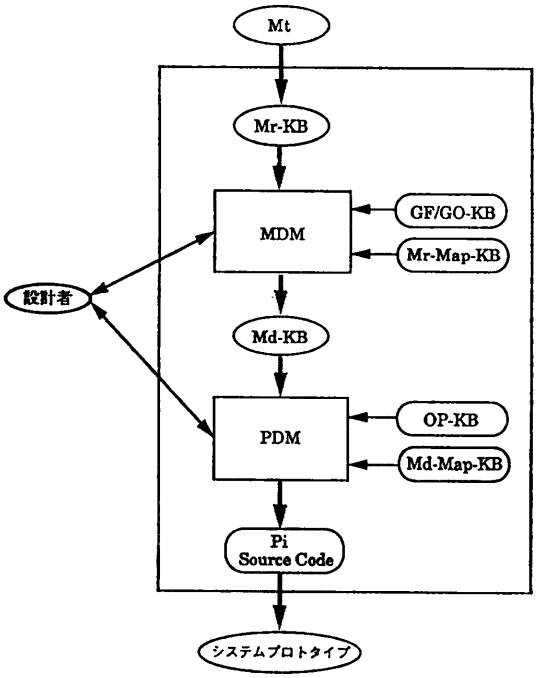

図 9 インタフェース設計支援システムの構成
Fig. 9 Structure of interface design support system.

4.2 設 計 例

4.1 節の設計プロセスに従って，図 6 に示したインタフェースイメージに基づく要求が利用者から与えられる場合のインタフェース設計例を示す．はじめに，このインタフェースイメージに関する利用者の要求が，3.2 節(3)の図 7 に示した形態の要求仕様知識 Mr として Mr-KB 上に生成される．この Mr に対して，まず，MDM の設計仕様定義プロセス Pd によって設計仕様知識 Md が導出される．Pd では，図 3，図 4 に示した GP/GO の部品知識階層を格納した G-KB に基づいて，Mr-Map-KB のマッピングを効果的に利用することにより，**図 10** に示すような Md

Primitive-Menu / GFixIconMenu		PME-002 Ex.1
RDF-Window	PME-I-011 PME-I-02 PME-I-03 PME-I-04	
IMD-01 IMD-03 IMD-7 IMD-10		
Alignment : Holizontal Num-of-Item : 4 Item-List : (Icon1 Icon2 Icon3 Icon4) Title : nil	Size : ((0.0 0.9) (1.0 0.1)) BorderWidth : 1 MasterView : ApView	
((Upper-Adjacent Win1))		

(a) Pモデルによる設計仕様知識の記述例

(a) Example of design specification knowledge of P-Model.

```
IMD21**  "Task Ex.1"
IME2101  Cond-Branch-1 [Menu-Item1 = 1]
IME2102    Cond-Branch-2  [Menu-Item2 = 1 | 2 | 3]
IME2103      Create-by-Copy-Obj Icon-Object [Menu-Item2] [Win1]
IME2104      Show-Obj Icon-Object [Win1]
IME2105      Reverse-Presentation Icon-Object [Win1]
IME2106      Move-Obj-with-Mouse-Pos Icon-Object [Win1]
IME2107      Put-Obj-at-M-Pos-with-M-B-Clilck Icon-Object [M-Pos] [Win1]
IME2108      Reset-State Operation-Menu
IME2109      Reset-State Primitive-Menu
IME2110      Exit

IME2111    Cond-Branch-2  [Menu-Item2 = 4]
IME2112      Put-Pop-Menu-at-Pos PopUpMenu1 [Pos1] [Win1]
IME2113      Read-Item-from-Popup-Menu Menu-Item3 [PopUpMenu1]
IME2114      Show-Obj Template-Obj [Menu-Item3] [Win1]
IME2115      Move-Obj-with-Mouse-Pos Template-Object [Win1]
IME2116      Put-Obj-at-M-Pos-with-M-B-Click Template-Object [Pos2] [Win1]
IME2117      Reset-State Operation-Menu
IME2118      Reset-State Primitive-Menu
IME2119      Exit
```

(b) Iモデルによる設計仕様知識の記述例

(b) Example of design specification knowledge of I-Model.

図 10 設計レベル知識モデル Md の記述例

Fig. 10 Example of design specification knowledge.

```
TR004
(ST #SEQ ?in) &
(IME #INUM ?in #GF_N ?gp #GF_A ?gpa #GO_N ?go #CLS ?op #GO_A ?goa) &
(OPKB #OPID ?op  #GFID ?gp #MSG ?msgid #TYPE 4) &
(CHK #GOID ?op #MSGID ?msgid #ARG1 ?a1 #ARG2 ?a2) &
(Class-INC #DEF ?a1 #INST ?gpa) &
(Class-INC #DEF ?a2 #INST ?goa)
->
(Generate "?msgid <- ?go ?msgid: ?goa")
(Modify ST #SEQ (?in + 1))
```

(a) Md マッピング知識の例 (変換ルール形式)

(a) Examples of Md mapping knowledge (Translation rule form).

```
(OPKB #OPID GFixTextMenu  #GFID Reset_State  #MSG resetEvent  #TYPE 1)
(OPKB #OPID GFixTextMenu  #GFID Read_Item_from_Menu  #MSG getSelectItem  #TYPE 3)
(OPKB #OPID GFixTextMenu  #GFID DisplayTMenu  #MSG (init ptNum)  #TYPE 5)
(OPKB #OPID GFixIconMenu  #GFID Read_Item_from_Menu  #MSG getSelectItem  #TYPE 3)
(OPKB #OPID GWindow  #GFID ShowObj  #MSG setSelectedObj  #TYPE 2)
(OPKB #OPID GWindow  #GFID GetEvent  #MSG getEvent  #TYPE 3)
(OPKB #OPID GWindow  #GFID PickObj  #MSG setNewPic  #TYPE 4)
(OPKB #OPID GWindow  #GFID DisplayWindow  #MSG (new label)  #TYPE 5)
(OPKB #OPID GIconGen  #GFID Generate_Icon  #MSG num  #TYPE 4)
(OPKB #OPID GOutRemoveRegion  #GFID PutText  #MSG putText  #TYPE 3)
(OPKB #OPID GPopUpMenu  #GFID Read_Item  #MSG selectItem  #TYPE 3)
(OPKB #OPID GStringInput  #GFID ReadText  #MSG getString  #TYPE 4)
                    :
(CHK #OPID GWindow  #MSGID (new label)  #ARG1 variable  #ARG2 (number  string))
(CHK #OPID GIconGen  #MSGID num  #ARG1 variable  #ARG2 number)
(CHK #OPID GStringInput  #MSGID getStrign  #ARG1 variable  #ARG2 string)
(CHK #OPID GOutRemoveRegion  #MSGID putText  #ARG1 string  #ARG2 nil)
(CHK #OPID GFixTextMenu  #MSGID resetEvent  #ARG1 nil  #ARG2 nil)
(CHK #OPID GFixTextMenu  #MSGID getSelectItem  #ARG1 number  #ARG2 nil)
                    :
```

(b) OP 部品知識の例

(b) Example of operational parts (OP) knowledge.

図 11 PDM で使用する設計知識の記述例

Fig. 11 Examples of design knowledge used in PDM.

がインタラクティブに生成される．次に，PDM のプロトタイプ設計プロセス Pp において，図 11(a) のような変換ルールとして定義された Md-Map-KB の Md-Pi マッピングと図 11(b) に示す OP-KB の動作部品知識を用いて，Md に対する MVC 変換をインタラクティブに行う．例えば，図 11(a) のルールは，変換タイプ 4 というカテゴリに属する要求知識の変換を行う知識であり，6 個の条件部と 2 個の実行部から成る．この条件部は，変換対象となる IME*i* のパターン (述語 IME を持つ記述)，OP 部品知識 (述語 OPKB の記述)，部品選択条件 (述語 CHK，および述語 Class-INC の記述)，および設計状態情報 (述語 ST の記述) から成り，同図中，？記号を付したルール変数を介して，図 11(b) の部品知識や選択情報などとユニフィケーションが行われる．ユニフィケーションが成功しルールが発火すると，その実行部によって変換結果が生成される．この場合，IME で指定された GO (変数 ?go によって表される) の持つ機能 (変数 ?msgid) に対して引数 (変数 ?goa) を送り，その結果を受け取る (変数 ?msgid) 実行文を生成し (述語 Generate の記述)，次の状態に遷移すること (述語 Modify の記述) を表している．こうした変換結果は逐次ワークスペースに出力され，それらを統合することによりプロトタイプ Pi のプログラムが半自動的に生成される．最後に，生成された Pi のプログラムを実行することによって図 12 に示すようなインタフェース表現が得られ，Smalltalk-80 環境上でインタフェース単体の動作の確認が行われる．

4.3 評価と検討

知識型設計方法論に基づいて，インタフェース設計タスクに知識モデルを導入し，インタフェース設計過程のモデル化を行った．知識モデルによる形式化された要求表現やインタフェース構成部品表現，およびそれらを利用する設計知識の適用によって，

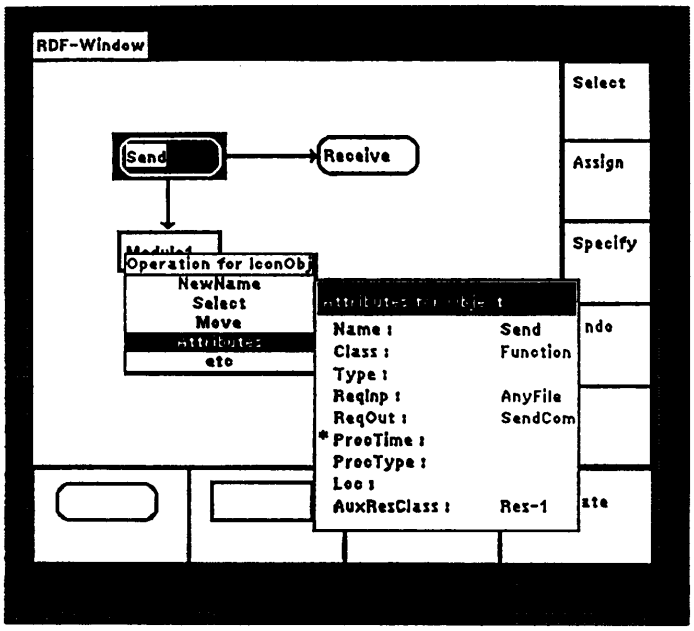

図 12 インタフェースプロトタイプ実行例
Fig. 12 Run-time example of interface prototype.

- インタフェース設計が独立した設計タスクとしてシステム設計から分離できる,
- 要求仕様や設計仕様に関する知識が明示的／宣言的に表現できる,
- 熟練した設計者の持つ知識がマッピングとして効果的に利用できる,
- インタフェースの生成が半自動的に可能となる,
- インタフェースの設計と検証がより容易に行える,

などの利点が得られることが確認された．また，本方式に基づいて試作されたインタフェース設計支援システムは，現段階では限られた設計知識と部品知識のみを有する小規模なものであるが，その構成は知識型システムの枠組みに準拠していることにより，

- 設計知識の追加により段階的に性能が向上できる,
- GF や GO, あるいは GP のセットを変更することにより，提供する機能や性質の異なるインタフェースの設計も扱える,

という性質を内包するものとなっている．

一方，知識型設計方法論の枠組みは，その基礎となる知識モデルとマッピングをドメインで定義することにより，インタフェース設計やコンピュータコミュニケーションシステムの設計[1]など複数の設計タスクに対して有効に機能することが確認された．

5. おわりに

本論文では，知識型設計方法論に基づくインタフェース設計方式と，そこで用いられるインタフェース設計のための知識モデルを提案した．本方式によって，インタフェース設計タスクとシステム機能設計タスクとが明示的に分離され，要求仕様定義からインタフェース実現に至るインタフェース設計プロセスが，知識モデル間のマッピングを効果的に利用した設計者のインタラクティブな設計プロセスとしてモデル化されることを示した．さらに，本方式に基づいて Smalltalk-80 上に試作されたインタフェース設計支援システムの構成を述べ，設計例を通してその有効性を示した．今後の課題としては，インタフェース設計の知識モデルや設計知識の表現方式の改良と検証，知識型設計方法論に基づく効率的な設計プロセス構成法の検討，より高度なインタフェース設計支援システムの実現などが残されている．

謝辞 日頃ご討論いただく沖電気工業(株)総合システム研究所，羽下雄之輔所長，ならびに東北大学電気通信研究所，野口正一教授に深謝する．

参考文献

1) Kinoshita, T., Sugawara, K. and Shiratori, N.: Knowledge-based Design Support System for Computer Communication System, *IEEE Journal SAC*, Vol. 6, No. 5, pp. 850-861 (1988).
2) Kinoshita, T.: A Knowledge Acquisition Model with Applications for Requirements Specification and Definition, *ACM SIGART Newsletter*, No. 108, pp. 166-168 (1989).
3) Rzepka, W. and Ohno, Y. (eds.): Special Issue: Requirements Engineering Environments, *IEEE Comput.*, Vol. 18, No. 4 (1985).
4) Rossen, M. B. et al.: The Designer as the User: Building Requirements for Design Tools from Design Practice, *CACM*, Vol. 31, No. 11, pp. 1288-1298 (1988).
5) Bailin, S. C.: An Object-Oriented Requirements Specification Method, *CACM*, Vol. 32, No. 5, pp. 608-623 (1989).
6) Sugawara, K. and Kinoshita, T. et al.: A Consideration of an End User Interface of Computer Communication Systems, *Proc. 2nd Int. Conf. Comput. Appl.*, IEEE, pp. 23-29

(1987).
7) 木下，菅原，白鳥：コンピュータコミュニケーションシステム設計における初期要求知識獲得について，情報処理学会マルチメディア通信と分散処理研究会資料，DPS 37-9 (1989).
8) Grimes, J. D. (ed.): Special Issue: Human Factors-Part 2, *IEEE CG & A*, Vol. 4, No. 12 (1984).
9) Hix, D. (ed.): Special Issue: User Interface *IEEE Softw.*, Vol. 6, No. 1 (1989).
10) Czuchry, A. J. et al.: KBRA: A New Paradigm for Requirements Engineering, *IEEE Expert*, Vol. 3, No. 4, pp. 21-35 (1988).
11) Lewis, T. G. et al.: Prototypes from Standard User Interface Management System, *IEEE Comput.*, Vol. 22, No. 5, pp. 51-60 (1989).
12) Norman, D. and Draper, S. (eds.): *User Centered System Design*, LEA (1986).
13) Urban, J. (ed.): Special Issue: Building Intelligence into Software Tools, *IEEE Expert*, Vol. 1, No. 4 (1986).
14) Mostow, J. (ed.): Special Issue: Artificial Intelligence and Sofware Engineering, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 11 (1985).
15) Mostow, J.: Toward Better Models of the Design Process, *AI Mag.*, AAAI, Spring, pp. 44-57 (1985).
16) Mitchel, T. et al.: Knowledge-based Approach for Design, *IEEE Trans. PAMI*, Vol. PAMI-7, No. 5, pp. 502-510 (1985).
17) Chandrasekaran, B.: Towards a Functional Architecutre of Intelligence Based on Generic Information Processing Tasks, *Proc. IJCAI-87*, pp. 1183-1192 (1987).
18) Brewer, F. D.: An Expert System Paradigm for Design, *Proc. 23rd DA Conf.*, IEEE, pp. 62-68 (1986).
19) Luqi: Knowledge-based Support for Rapid Software Prototyping, *IEEE Expert*, Vol. 3, No. 4, pp. 9-18 (1988).
20) 木下：設計プランの概念に基づく設計過程の形式化の検討，人工知能学会第3回全大予稿，11-33 (1989).
21) 木下，菅原，白鳥：マンマシンインタフェース設計のための知識モデル，電子情報通信学会オフィスシステム研究会資料，OS 89-22 (1989).
22) Fukushima, M., Sugawara, K. and Oizumi, J.: A Smalltalk Implementation of Distributed Rule Base System for Distributed Problem Solving, *Proc. 4th Intl. Joint Workshop Comput. Comm.*, pp. 249-258 (1989).

(平成元年8月30日受付)
(平成2年4月17日採録)

木下 哲男（正会員）

昭和28年生．昭和54年東北大学大学院情報工学専攻博士前期課程修了．同年沖電気工業(株)入社．現在，同社研究開発本部総合システム研究所勤務．知識表現モデル，知識型システムとその構築支援環境の研究開発に従事．情報処理学会平成元年度研究賞受賞．電子情報通信学会，人工知能学会，日本認知科学会，AAAI, ACL各会員．

岩根 典之（正会員）

1959年生．1983年広島大学総合科学部卒業．1986年同大学院環境科学研究科修士課程修了．現在，沖電気工業(株)総合システム研究所勤務．知識表現，ニューラルネットワークの研究に従事．

菅原 研次（正会員）

昭和25年生．昭和48年東北大学通信工学科卒業．同年富士通入社．昭和55年東北大学博士課程中退．同年千葉工業大学電子工学科助手．工学博士．研究テーマは知識工学，計算機ネットワーク，分散AI, CAI, ヒューマンインタフェース．IEEE, 電子情報通信学会，人工知能学会，ソフトウェア科学会，日本ロボット学会各会員．

白鳥 則郎（正会員）

昭和21年生．昭和52年東北大学大学院博士課程修了．同年，東北大学電気通信研究所勤務，現在同大助教授．知的分散処理システム，コンピュタネットワーク，プロトコルソフトウェア設計，ネットワークOS, プロトコル仕様記述言語，検証法などの研究開発に従事．情報処理学会25周年記念論文賞受賞．IEEE, 電子情報通信学会各会員．