ICS-99B-53

オブジェクト指向ソフトウエア自動設計用
分散型知識ベースシステムの開発：
システム設計

指導教官　　B.H. Far 助教授

平成 11 年 2 月 9 日

工学部情報システム工学科 55762

Romi Satria Wahono

埼玉大学工学部情報システム工学科 Far 研究室

埼玉県浦和市下大久保 255

目次

図目次

概要

本研究は、オブジェクト指向方法論によるソフトウエア自動設計のための設計知識抽出方式と、それを再利用する分散型知識ベースシステムの研究である。本研究の最終目的は、ソフトウエアの生産性及び信頼性の向上を狙い、オブジェクト指向方法論によるソフトウエア自動設計を行なうために分散型知識ベースシステムを実現する確立にある。

オブジェクト指向方法論によるソフトウエア自動設計方式は人間の設計結果から設計知識を抽出し再利用して、広い範囲の対象に適用可能とすることを狙う。そのソフトウエア自動設計のための分散型知識ベースシステムを容易化し、効率的に開発できるように、次の考え方を採用する。

1. オブジェクト指向分析・設計方法論を利用して、システムを基本的設計から各エキスパートユニットの設計まで、全体的に設計する。システム設計を完成したら、システム設計をチェック・テストし、設計バグを検討する。最後に、Java 言語で、分散型知識ベースシステムを実装・実現する。
2. 知識ベースには、基本的に人間によるオブジェクト指向設計の思考に対応したソフトウエア設計ルールと設計パターンを蓄積する。グループウエアでの作業を簡単化し、処理速度を上がるためなど、分散型知識ベースアプローチを採用する。
3. エンドユーザとシステムとの対話を提供するために、インタラクティブユーザインタフェイスを、分散オブジェクト CORBA を用いるコミュニケーションエンジンを実現する。

本論文の内容は、

1. 第１章では、研究背景、研究の目的、研究アプローチについて述べる。
2. 第２章では、知識ベースシステムの基本概念や本研究と関連する AI 技法などについて述べる。
3. 第３章では、本研究で開発している分散型知識ベースシステムの全体的な分析と設計について述べる。
4. 第４章では、本研究で開発している分散型知識ベースシステムの各エキスパートユニットの分析と設計について述べる。
5. 第５章では、設計されたシステムを実装・実現するために、採用した技術やアプローチなどについて詳しく述べる。
6. 第６章では、本研究の結論と今後の課題をまとめる。

である。

第1章 序論

本章には、研究背景、研究の目的、研究アプローチについて述べる。現在には、オブジェクト指向分析設計プロセスの問題、新たな分散 CASE システムへの要求、大規模知識ベースシステムでの問題があるため、本研究の背景となる。それで、その問題を解決するするのは研究の目的である。解決方針としては、研究アプローチでまとめる。

1.1 背景

1.1.1 オブジェクト指向分析設計プロセスの問題

オブジェクト指向分析設計は、実世界の概念を中心に組織化されたモデルを利用した問題解決のための新しい思考法である。その基本要素はオブジェクトであり、単一の実体の中にデータ構造とその振舞いの両方を持っている。したがって、分析の中心は、実世界からオブジェクトを抽出し、オブジェクト間の関連を定義するオブジェクトモデルの構築である。オブジェクト指向設計は、問題の解決や適用分野の専門家とのコミュニケーション、企業のモデル化やドキュメントの作成、さらにプログラムやデータベースの設計に有効である。オブジェクト指向分析設計方法論として、Coad/Yourdon 法、OMT 法、Shlaer/Mellor 法、James Martin の OOA&D、UML 法などがある。

いずれの方法論も、実体関連図から派性したオブジェクトモデルを構築する

点では共通している。そのオブジェクトモデルを構築するために、次のようなプロセスが不可欠なことになる。

1. **オブジェクトクラスの識別**
2. **オブジェクト関連の識別**
3. **属性の識別**
4. **継承による洗練**

しかし、オブジェクト識別をはじめ、上記のプロセスが困難だと専門家や技術者の認識が一致している。その要因はオブジェクト指向パラダイムが本来持つ問題ではなく、その各プロセス方法自体に誤りがあると考えている。つまり、現在のオブジェクト指向分析設計方法論では、良い設計というのは個人の経験や発見的手法により決められている。

1.1.2 新たな分散 CASE システムへの要求

分散環境は、通常図 1.1 に示すような２つの意味で用いられている。１つは、従来の集中ホストマシンに代わり、ワークステーションとネットワークからなる分散処理による開発環境である。一方開発組織そのものが、地理的に離れた複数の開発拠点に分散する開発環境がある。これを分散組織環境と呼ぶ。両者に共通の場合に、分散開発環境よ呼ぶ。実際には、両方の分散形態が相互に推進力となり、分散処理環境上で分散組織による開発へ移行している[13]。

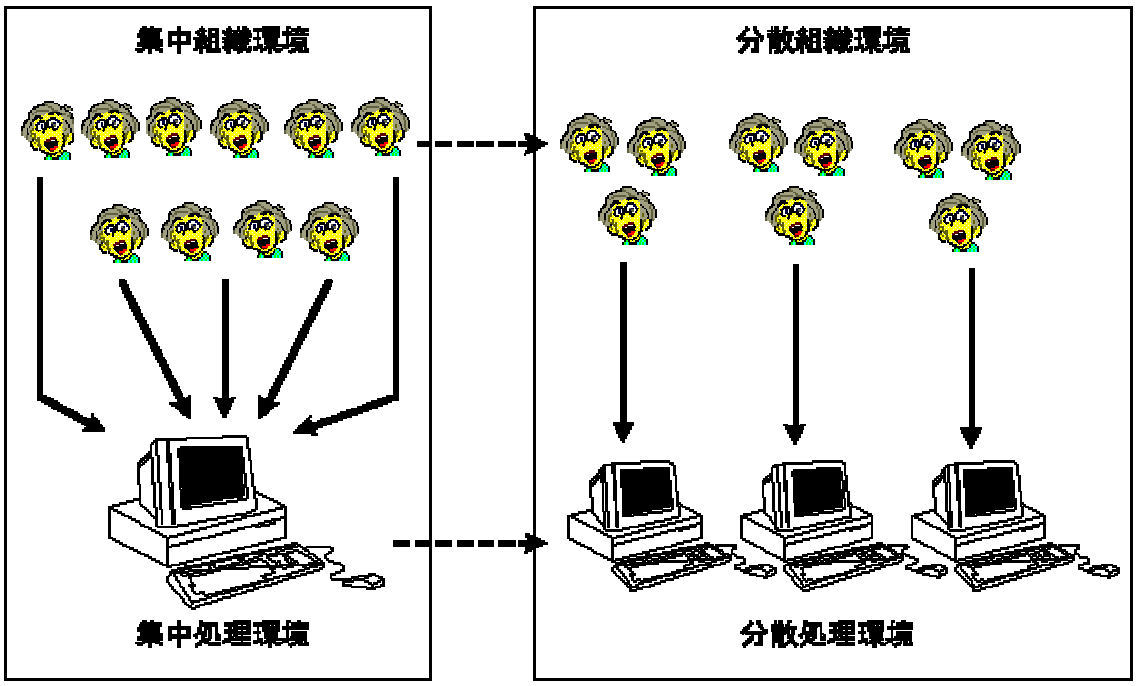

図 1.1： 分散開発環境

ソフトウエア開発環境が分散する背景には、技術的背景と社会社会的背景という要因が指摘されている。

1. 技術的背景

情報処理技術の変化、例えばコストパフォーマンスの優れたワークステーションと高速ネットワーク技術の発展核として、今や一人一人のソフトウエア技術者が１台のワークステーションを占有する一人１台のワークステーション環境の時代になっている。一方、開発環境に対する要求も変化している。従来の TSS（Time Sharing System）による集中処理環境におけるレスポンス時間がプログラム生産性に大きな影響を及ぼすことが実験的に指摘され、生産性向上のためにレスポンス時間の短縮が要求された。分散処理環境ではこの問題を経済的に解消できる。

2. 社会的背景

開発ソフトウエア規模に絶えざる増大にともない、ソフトウエア開発要因が依然不足している。多数の開発要因を集中してい各保することが経済的にも社会的にも困難となっているため地域的に分散して開発する。分散組織環境

へ移行している。その他、Drucker [14]は、21 世紀のワークスタイルは従来の階層的組織における命令型からチームやグループを中心とするネットワーク型組織へ移行すると指摘している。

上記のような分散開発環境や分散組織環境などにより、グループウエアという概念が提案された。グループウエアというのは、複数の人が互いに相談したり助け合いながら全体として１つの処理を実行し、個人あるいはグループの間での協調を支援する機構である。その機構に基づき、グループウエアでソフトウエア開発するため、情報共通(data sharing)、相互運用性(interoperability)、互換性(compatibility)、可搬性(portability)などの問題を解決できる CASE システムを要求している。

1.1.3 大規模知識ベースシステムでの問題

知識ベースシステムは、 問題解決を知識表現言語によって表現した知識を格納した知識ベースと、 その内容を解釈実行する推論エンジンから構成される。知識ベースシステムの研究開発・実用化が進められて大規模知識ベースシステムの開発が行なわれるようになると、 いくつかの問題が顕在化してきた[5]。

知識獲得ボトルネック　知識システムを開発するためには、 既存の知識ベースシステム構築用ツールの求めているレベルまで、専門家の知識を分析、抽出体系化詳細化手続き化する必要がある。これまでは、専門家にインタビューをすることによって知識を抽出し、それを知識表現言語で表現し、プロトタイプを構築してそれを再び専門家に提示して詳細化・洗練するという知識獲得の作業を繰り返してきたが、 それは膨大な労力を要するものであった。

狭さ　初期の知識ベースシステムの開発では、知識獲得が非常に効率の悪いものであったため、 デモンストレーション効果のあるシステムを作るためには特定の狭い専門家領域に特化せざるをえなかった。そのようなシステムは、設計時に考慮していなかった現象に対して大変もろく、専門家に比べて急激に問題解決能力が低下する傾向があった。

既存システムとの統合の困難さ　初期の知識ベースシステムはどれも既存のデータベースシステム、CAD ツール、リアルタイムシステムなどとの統合を考慮したものではなかったので、孤立的な性格が強く、システムとしての有用性を低下させるものであった。

知識メディアとしての弱さ　知識ベースシステムに蓄積される知識は基本的にはコンピュータ主体の問題解決を指向したものであるため、蓄積された知識自体は専門家や作業員にはブラックボックスに近く、蓄積されている知識そのものは役に立たない。また、知識表現言語で記述できない知識やノウハウは放置される傾向があった。そのため、知識ベースシステムは問題解決に必要な知識をほとんど完全に知識ベース化するまでは役に立たない。

方法論の欠如　小規模なプロトタイプ開発と異なり、大規模で実用的な知識ベースシステムの構築には、一定の開発方法論が不可欠である。

知識ベース処理の速度　知識ベースが大きければ大きいほど、知識ベースを処理する速度を減少し、新たに効率的な知識ベースアーキテクチャを導入する必要がある。

大規模知識ベースシステムの研究では、上に述べたような現在の知識ベースシステムの限界を超えるため、どの領域において知識ベースを構築する場合にも必要になってくる共通的知識を体系化して供給することによって新たに知識獲得すべき知識の量を大幅び削減することを狙っている。

大規模知識ベースシステムの研究は、集積指向アプローチ(図 1.2-a 参照)と共有・再利用指向のアプローチ(図 1.2-b 参照)の２つに大別できる[2]。集積指向のアプローチでは、標準的な知識表現言語やモデリング言語を設定して、常識的な知識や基礎的知識を包括的に収集し、蓄積することに重点を置いている。一方、共有・再利用指向のアプローチではことなる方法論や概念化に基づく知識ベースシステムやソフトウエアシステムを協調させるための枠組の構築に重点を置いている。

本研究は共有・再利用アプローチに基づき、分散型知識ベースシステムを開発する。

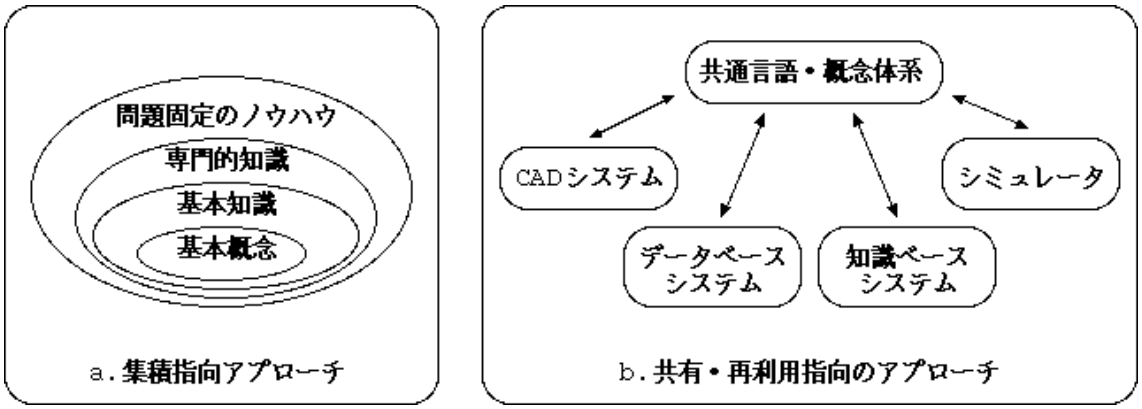

図 1.2： 大規模知識ベースシステムの研究アプローチ

1.2 目的

本研究の目的は、下記のようになる。

1. オブジェクト指向設計での困難を理解して、解決方法を研究する。人間によるオブジェクト指向ソフトウエアの設計パターンや設計ルールなどの有効な設計知識を獲得する。
2. 獲得した設計知識を蓄積するための効率的な知識ベースのアーキテクチャーを求める。
3. 実需的なユーザインターフェイスを持つ、ネットワーク技術に対応できる、グループウエアで利用できるシステムを実現する。

本研究の目的は、このようなオブジェクト指向ソフトウエア自動設計のための分散型知識ベースシステムを開発することである。

1.3 研究アプローチ

研究の背景と目的に基づき、本研究では、次のような研究アプローチを採用する。

1. オブジェクト識別・関連の識別・属性の識別・継承による洗練での困難に対するアプローチは、まずそのプロセスに対応するルールを求めて、知識ベースに蓄積する。そしてそのルールに基づいて、困難な設計プロセスを自動化する。
2. 分散オブジェクトを用いる分散システムアプローチを採用する。その理由は、まず、グループウエアでの情報の共通や相互運用性の問題を解決できる。そのほかスケーラビリティ、可搬性、柔軟性を持つシステムを実現することができるからである。
3. 分散型知識ベースアプローチを採用する。それは、効率的な知識ベースであり、ネットワーク及び C/S システム技術を提供でき、地理的に離れた場所からも設計知識を利用でき、知識獲得ボトルネックの問題を解決できるという理由で採用した。

1.4 本章のまとめ

本章には、研究背景、研究の目的、研究アプローチについて述べた。研究の背景は、オブジェクト指向分析設計プロセスの問題、新たな分散 CASE システムへの要求、大規模知識ベースシステムでの問題というが現状であり、本研究の背景といえる。それで、各問題及び要求を分析して、包括的にまとめられるシステムを開発する必要があると考えた。結局、1.2 の研究の目的に述べた特徴を持つオブジェクト指向ソフトウエア自動設計用分散型知識ベースシステムを開発すると決定した。具体的な解決方針としては、1.3 の研究アプローチでまとめた。

第2章 知識ベースシステムの基本概念

本章には、知識ベースシステムの基本概念を述べる。知識ベースシステム(Knowledge Based System)とは、人工知能研究で見出された考え方、 技術を、現実の問題に適用することから生まれた新しい情報システム作成の手法である。知識ベースシステムの基本的な構成は知識ベースと推論機構に加えて、ユーザインタフェースからなる。 知識ベースシステムを、 現在、情報処理技術やソフトウエア開発の変化などによって、分散人工知能応用としての分散型知識ベースシステムに拡張する研究も出てきている。

知識ベースシステムの特徴は、下記のようにまとめられる。

- 悪構造・悪定義問題を対象すること
- 知識情報処理を基本とする
- 人工知能や知識工学を基盤技術とすること
- プロトタイピング的接近を必要とすること
- システムの透明性が要請されること
- ユーザとの対話性が重視されること
- 保守や拡張が容易に行なえること

2.1 知識ベースシステムとは

知識ベースシステム(Knowledge Based System)とは、人工知能研究で見出された考え方、技術を、現実の問題に適用することから生まれた新しい情報シ

ステム作成の手法である[25]。1960 年代後半に知識ベースシステムの相先に当たるシステムの開発がはじまり、1970 年代半ばに医療診断の分野のシステムが大きな注目を集めることになった。その後、1970 年代後半にはいくつかの分野で知識ベースシステムの開発が進められるようになり、ワークステーションといった形でコンピュータの高性能化が進んだことも背景となり、1980 年代に実用化の時代を迎え、大きく発展した。 知識ベースシステムは、もっと広い意味でエキスパートシステムとも言われる。

知識ベースシステムを規定するためには、少なくとも次の 2 項目は不可欠の要素である[5]。

1. 明示的な知識構造に基づくシステムであること(形式)。
2. 特定の専門的分野の知識をもち、 それを利用してその分野の専門的な問題解決や作業を知的に行なうシステム(内容・能力)。

１の形式に関しては、従来型のシステムなりプログラムも対象問題に関する知識に基づいて作成されているのである。知識ベース型である知識ベースシステムでは“ 知識の表し方 ”や“ 知識の使い方 ”が異なっており、知識とこれを利用する推論機構とを分離して扱う。

２の内容・能力に関しては、理想的には特定の分野の問題に対して、専門家並みの能力で知的に対処することが期待されている。知識ベースシステムでは現状の人工知能技術の基礎上に、知識を集積し実用性のあるシステムとすることが重要となるが、幅の広い知識である常識や、直観的な認知能力、学習能力などの利用は今後の知識ベースシステム、さらには人工知能研究の大きな問題である。

知識ベースシステムの基本的な構成は図 2.1 に示されるように、知識ベースと推論機構に加えて、ユーザインタフェースからなる。

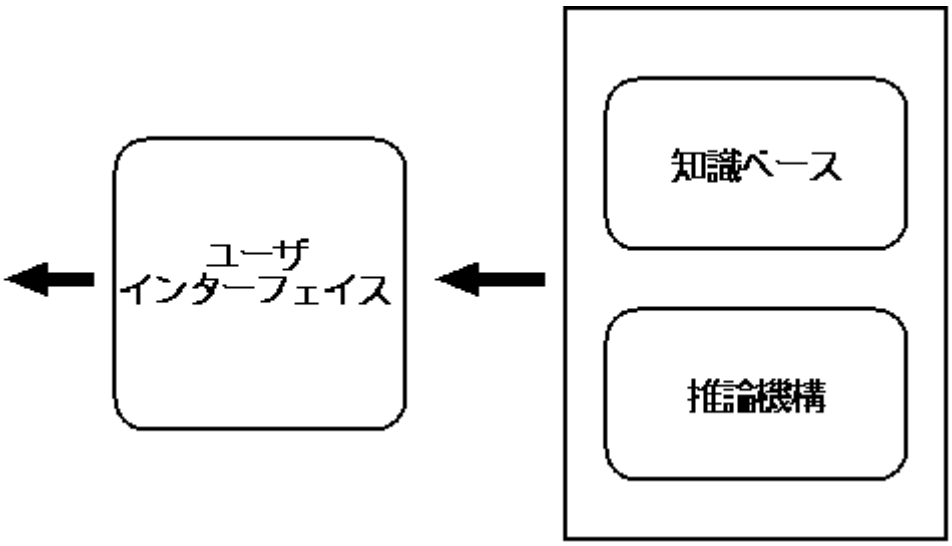

図 2.1：知識ベースシステムの基本構成

2.2 知識ベースシステムの研究進化

分散人工知能の応用として分散型知識ベースシステムの研究を現在話題になりつつある。スタンドアローン型知識ベースシステム(Standalone Knowledge Based System)から分散型知識ベースシステム(Distributed Knowledge Based System)へ拡張する研究も出て来ている[21][22]。さらにインターネット・WEBをベースにする分散型知識ベースシステムを開発している研究者も少なくない。例としては、Ex-W-Pert System[23]と XPert Web System[24]をあげられる。

現在の知識ベースシステムは、情報処理技術やソフトウエア開発の変化などによって、スタンドアローン型知識ベースシステムから分散人工知能応用としての分散型知識ベースシステムに進化している。そのスタンドアローン型知識ベースシステムの研究のプロセスは分散型知識ベースシステムとの比較すると、図 2.2 のようになる[25]。

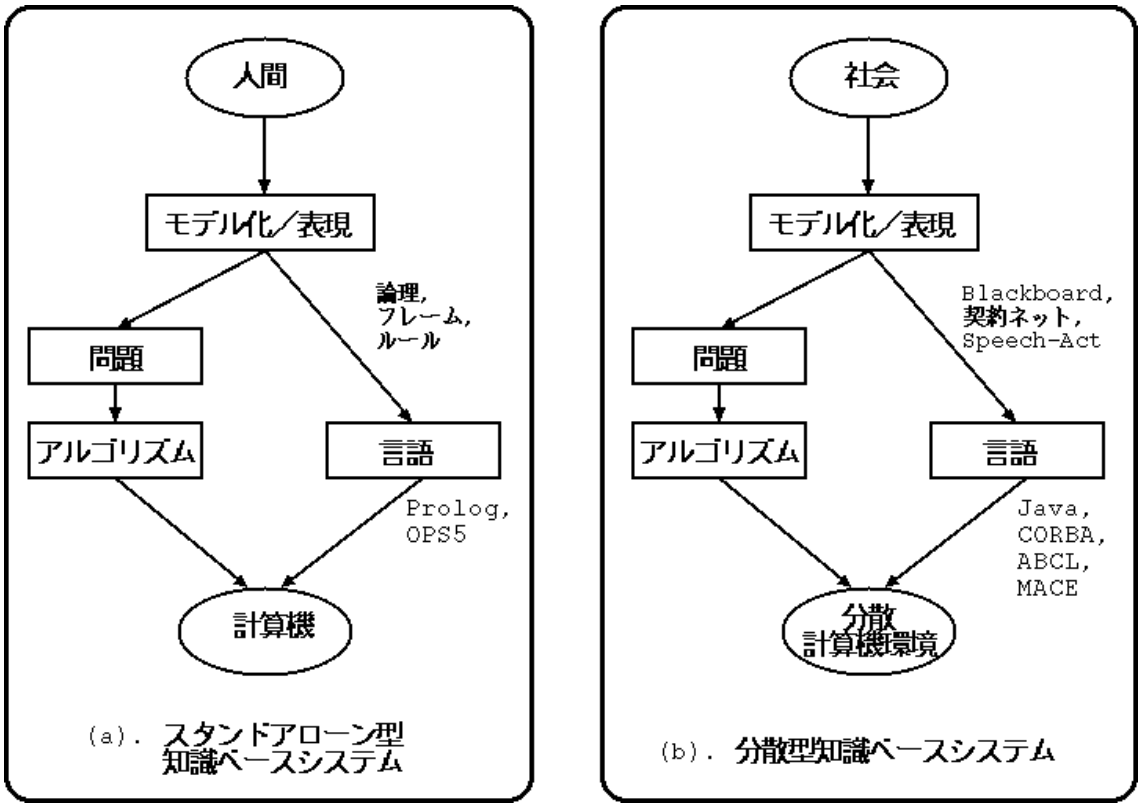

図 2.2：知識ベースシステムの研究プロセスの進化

1. スタンドアローン型知識ベースシステム

図 2.2(a)は、スタンドアローン型知識ベースシステム研究のフローを表している。スタンドアローン型知識ベースシステムの研究では観察の対象は「人間」個々知性である。機械 ではとうてい実現できない処理を人間が行なう様子 を観察し、「モデル」を作り「表現」を与える。例えば、理論、フレーム、プロダクションシステムなどがこれにあたる。表現が与えられると、それ以降の手順は２つ分かれる。１つ道筋では解くべき「問題」が定式化され「アルゴリズム」が研究される。例えば、論理を表現として選択すると定理証明アルゴリズムが研究され、プロダクションシステムを選択するパターンマッチアルゴリズムが研究される。もう１つ道筋は、問題が容易に定式化できない（ill-defined）場合である。このときは、表現をプログラミング言語化し問題の解法を直接プログラムすることが試みられる。論理型言語（Prolog)、プロダクションシステ言語(OPS5)、フレーム言語などが提案され、これまで多くの知識ベースシステムが記述されてきた。いずれかの道筋をとるにせよ、結局「計算機」上に人間の知性の一端が模倣される。

2. 分散型知識ベースシステム

図 2.2(b)に分散型知識ベースシステムの研究のフロー示す。観察対象は人間個々の知性ではなく、その集団である「社会」全体の知的な振舞いである。「モデル」としては黒板(blackboard)、契約ネット、発話行為(Speech-Act)などが提案されてきた。「表現」が与えられた後の道筋はスタンドアローン型知識ベースシステムと変わるところはないが、「問題」「アルゴリズム」に関しては、多くの人々を納得させる結果はまだ得られていない。「言語」に関して は、分散環境を提供出来る言語を Java、CORBA、さらに分散問題解決を提供する ABCL/1、MACE を取り上げられる。「計算機」環境は自然と「分散計算機環境」になることが多いが、別々に作られたプログラム郡を 1 台の計算機上で協調させる場合を排除するものではない。

分散型分散型知識ベースシステムの基本的な構成は図 2.3 に示されるように、分散知識ベースと分散推論機構に加えて、ユーザインタフェースからなる。

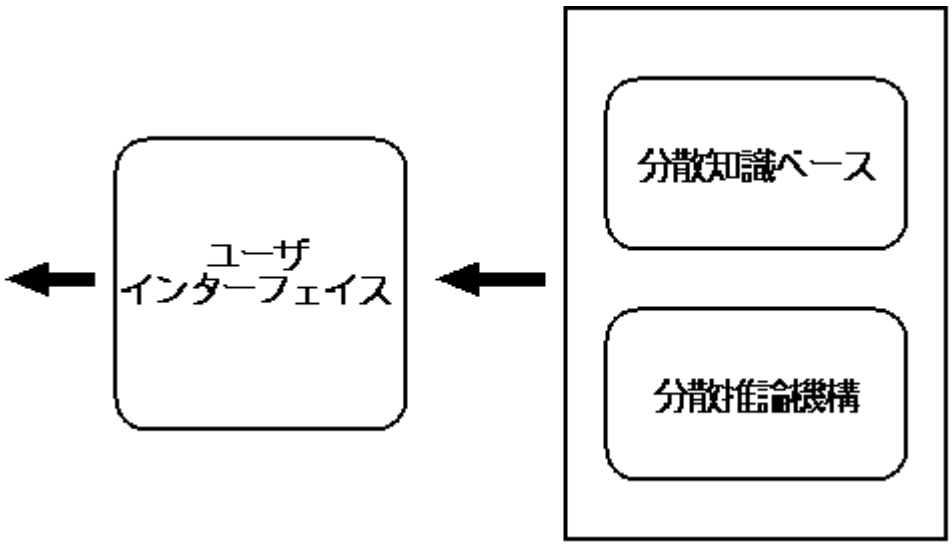

図 2.3：分散型知識ベースシステムの基本構成

2.3 知識ベースシステムの特徴

知識ベースシステムは図 2.4 に示すように、次のような特徴をもつ[25]。

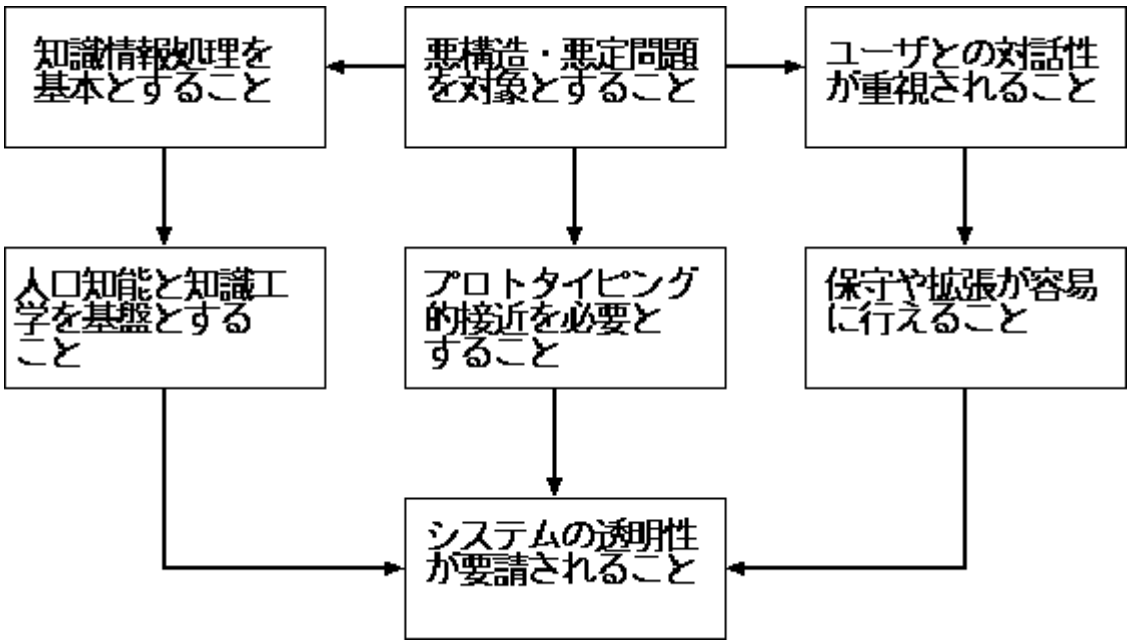

図 2.4 : 知識ベースシステムの特徴

- **悪構造・悪定義問題を対象すること**
  対象とするシステムの機能や構造が明確であり、数理的あるいはアルゴリズム的な解決が可能な問題を良構造(well-structured)問題という。また、問題解決の範囲（ユーザの要求仕様）を明確に設定できる問題を良定義（well-defined）問題という。一方、アルゴリズム的な解決が難しい問題を悪構造的（ill-structured)、また問題解決の範囲を明確に設定することが難しい問題を悪定義的(ill-defined)という。知識ベースシステムは、一般に、悪構造的または悪定義的な性質をもつ問題を対象とすることを特徴とする

- **知識情報処理を基本とする**
  知識ベースシステムの基本的枠組はプロダクションシステムであり、対象に関する情報が格納されるワーキングメモリ、問題解決に使われる知識が格納される知識ベース及び問題解決の過程を制御する推論機構から構成される。

したがって、知識の表現、推論の制御、知識の獲得と管理など、知識情報処理が問題解決において主要な役割を演じる。

- **人工知能や知識工学を基盤技術とすること**
  知識ベースシステムの基盤技術は、人工知能と知識工学に求めることができる。すなわち、人工知能は探索や問題解決戦略、知識の表現と利用などの要素技術を提供する。一方、知識工学は構築方法論、知識獲得支援、知識プログラミングなどのシステム技術を提供する。

- **プロトタイピング的接近を必要とすること**
  悪構造・悪定義問題では、問題解決の範囲及び問題解決の方法を事前に明確することが困難であることから、とりあえず動作可能なプロトタイプシステムを作ってみるという方法は有用である。すなわち、プロトタイプをユーザに呈示することにより、ユーザの要求仕様を次第に明確化することができる。また、プロトタイプを作ることにより、知識秘術者自身が問題解決方法の新たな手掛りを見出することを期待できる。したがって、プロトタイピング的接近は知識ベースシステムの開発において有用かつ実現的手段といえる。

- **システムの透明性が要請されること**
  プロトタイピング的方法に基づいて知識ベースシステムを階段的に開発するためには、システムをブラックボックスにしてはならない。すなわち、システムの透明性(transparency)を確保することが重要である。知識ベースと推論機構を分離しておくことは、システムの透明性を確保するための必要条件である。しかし、複雑な問題解決を必要とするときには、推論過程を間接的に制御するためのメタ的知識を知識ベースにおかざるを得ないこともあり、これはシステムの透明性を阻害する要因となるので、十分な注意が必要である。

- **ユーザとの対話性が重視されること**
  知識ベースシステムはユーザから見てもブラックボックスであってはならない。ユーザに対し、推論結果を示すだけではなく、結果に至る推論筋道をその根拠と共に明確に示すことにより、ユーザとシステムの間の対話性を向上させる必要がある。

- **保守や拡張が容易に行なえること**
  知識ベースシステムは、悪構造・悪定義問題を対象とすることから、システムとして完成したと呼ばれる段階に達するのはまれである。言い換えれば、知識ベースシステムは永遠にプロトタイプシステムであり続けるともいえる。したがって、知識ベースシステムの保守と拡張は、ある時期を境として、システム開発者からユーザの手に円滑に委譲されることが望ましい。そのためには、知識ベースシステムは手離れの良いシステムとしておく必要がある。そのためにも、システムの透明性やユーザとの対話性の確保は重要である。

2.4 本章のまとめ

本章には、知識ベースシステムの基本概念を述べた。知識ベースシステム(Knowledge Based System)とは、人工知能研究で見出された考え方、技術を、現実の問題に適用することから生まれた新しい情報システム作成の手法である。知識ベースシステムの基本的な構成は知識ベースと推論機構に加えて、ユーザインタフェースからなる。そして、知識ベースシステムの特徴と現在の進化などについては、2.2 と 2.3 のところに詳しく述べた。

第3章 分散型知識ベースシステムの分析と設計

本章には、システムの分析と設計について述べる。本システムは根本的に OMT によるオブジェクト指向方法論に基づいて設計されている。まず、オブジェクト指向パラダイムと OMT 法との基本概念を説明し、その後システムの全体てきな設計に進める。本システムの設計手順としては、下記のようになる。

1. 問題を分析して、システムアーキテクチャを設計する
2. オブジェクトモデルを設計する
3. 機能モデルを設計する
4. 動的モデルを設計する

本研究で開発しているシステム全体構成は、ユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 からなる。本章には、その全体的なシステムの分析と設計について説明する。

3.1 オブジェクト指向の基本概念

オブジェクト指向アプローチの発想の原点は、現実のもの及びもの同士の関係をそのままソフトウエアで表現することによって、現実世界の仕組みをコンピュータ上で再現しようとするものである。現実に近い形でシステムの構造

を表すことによって、その構造がより直感的でわかりやすくなる。現実の世界は、いろいろなものが役割を分担しながら機能している。自分でできることは独立して作業を進め、自分のテリトリ以外の仕事は他にまかせて、結果だけをもらう。そこで、現実の世界のもの（オブジェクト）ともの同士のつながり（関係）に着目し、個々のオブジェクトに作業を割り振り、オブジェクト同士がお互いに作業を依頼しながら機能するようにしたにが、オブジェクト指向アプローチである。

オブジェクト指向システムの基本的な構造は、図 3.1 に示すように、「オブジェクト」「クラス」「属性」「メッセージ」「メソッド」「関係」からなる[20]。

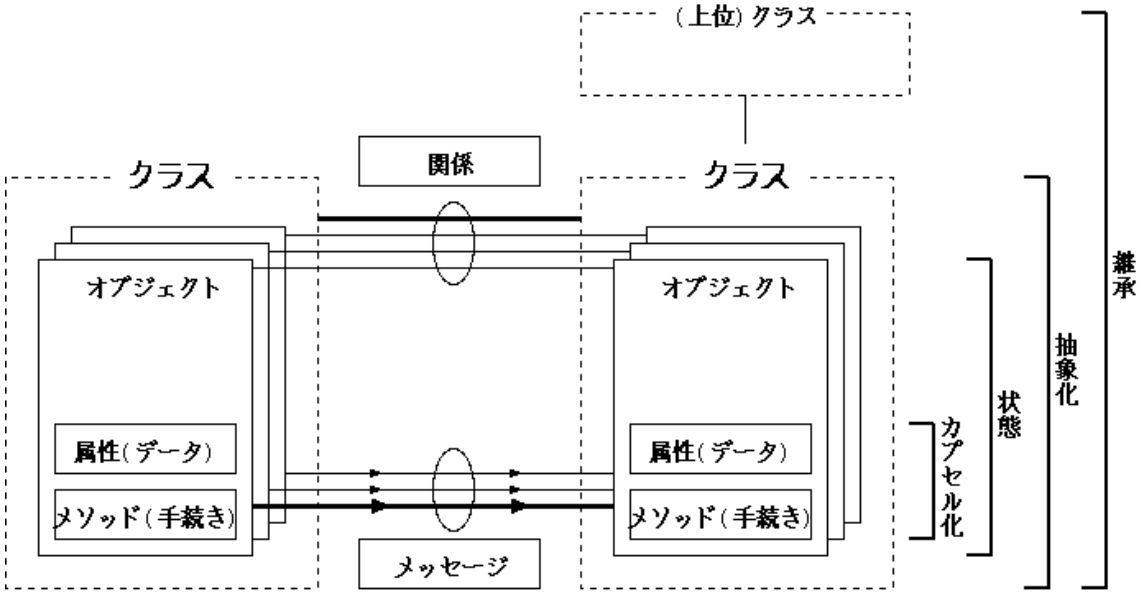

図 3.1： オブジェクト指向システムの基本的な構造

システムはオブジェクトが中心となって構成される。個々オブジェクトはその役割（作業）を果たすのに必要な「属性（データ）」と「メソッド（手続き）」を内部に持っている。この属性とメソッドを１つにまとめた構造をカプセル化と呼び、オブジェクト指向の特徴の１つとなっている。このような構造を持つオブジェクト同士は「メッセージ（作業依頼）」を送ることで、協調的に作業を進めることができる。そのメッセージを受けたオブジェクトは作業した結果を必要に応じて他のオブジェクトに渡す。オブジェクト指向のもう１つの特徴と

して、「個々のオブジェクトをさらに抽象化した概念（クラス）」を取り入れている。クラスはオブジェクト（インスタンス）という実対のテンプレートである。さらに、オブジェクトはお互いの位置づけを明確にする必要がある。この位置付けを表現するのが「オブジェクト同士のつながり（関係）」である。

オブジェクト指向の概念として、「抽象化」「カプセル化」「継承」「状態」がある[20]。

- **抽象化**
  抽象化には、現実世界のものをオブジェクトとして抽象化するという意味のほかに、同じ性質を持つオブジェクト群をさらにクラスとして抽象化する、あるいは、クラス群をさらに抽象的なクラスとして抽象化するという意味がある。このようなものの抽象化階層を作り上げることによって、ものの構造を体系的にとらえることも可能になる、これによって実世界を自然に把握することができる。

- **カプセル化**
  カプセル化はオブジェクトの内部にデータを隠蔽し、外部には手続きの仕様のみを見せることで、オブジェクトの仕様と実装を分離するアプローチである。このカプセル化による最も大きな効果はオブジェクトの提供者とオブジェクトの利用者の切り分けを明確にできる点である。

- **継承**
  継承は、上位クラスの概念を下位クラスが引き継ぐとともに、下位クラスでは上位クラスに存在しない新たな性質を追加することができる。この継承によって、既存の資源を有効に再利用して生産性を向上させる手段を提供している。

- **状態**
  オブジェクトに定義されている手続き群の利用可否は、実際には内部の属性値によって決まる場合がある。そこで、オブジェクトを特定の内部データ値によって複数の異なる状態に分け、オブジェクトの状態ごとに、その時点で利用可能な手続きを明示する必要がある。状態の概念を取り入れることの効果は、オブジェクトの提供者がメッセージの正しい受信手順（メソッドの利用手順）を状態遷移図として利用者に提供できることである。

3.2 オブジェクト指向方法論 OMT

オブジェクト指向方法論は、1990 年頃から世界で大きな注目を浴びている。その特徴としては、実世界の概念を中心に組織化されたモデルを利用した問題解決のための新しい思考法であり、分析から実現まで一貫したモデルが使用できることなどである。その基本要素はオブジェクトであり、単一の実体の中にデータ構造とその振舞いの両方を持っている。オブジェクト指向モデルは、問題の理解や適用分野の専門家とコミュニケーション、企業のモデル化やドキュメントの作成、さらにプログラムやデータベース設計に有効である。

OMT(Object Modeling Technique)法は、1992 年に Rumbaugh ら[7]に提案されたオブジェクト指向方法論の１つである。Rumbaugh らは３つの関連しているが異なった視点からシステムをモデル化することが有効であることを発見した。それらの視点はおのおのシステムの重要な側面をとらえているのだが、完全な記述のためにはそれらすべてが必要である。オブジェクトモデル化技法（OMT）はシステムをモデル化するための、これら３つの見方を組み合わせた方法論に対して Rumbaugh らが付けた名前である。オブジェクトモデルはシステムの静的、構造的、「データ的」な側面を表現している。動的モデルはシステムの時間的、動作的、「制御的」な側面を表現している。機能モデルはシステムの変換的、関数的、「機能的」な側面を表現している。

典型的なソフトウエアの手続きは３つの側面すべてを組み込んでいる。それはデータ構造を使い（オブジェクトモデル）、操作を適切なタイミングで順に並べ（動的モデル）、そして値を変換する（機能モデル）。各モデルは他のモデルの中の実体に対する参照を含んでいる。この３つ種類のモデルは、システムを直交する３つの見方に分割し、それぞれ統一的な記法で表現し操作することを可能とする。それぞれのモデルは完全に独立ではない、つまりシステムは独立した部分の集まり以上のものである。けれども各モデルは、それだけで独立してかなりの部分まで検査したり理解したりすることが出来る。異なるモデル間の相互関係は限定されており、陽に示されるからである。もちろん３つのモデルがあまりに組み合っていて分離できないような悪い設計をすることはいつでも可能である。しかし、良い設計はシステムの異なる側面を分離し、それらの間の結合を制限する。それぞれ３つのモデルを詳しく下記に説明している。

3.2.1 オブジェクトモデル

オブジェクトモデルはシステム中のオブジェクトの構造を記述する。記述する内容は、オブジェクトのアイデンティティ、他のオブジェクトとの関係、オブジェクトの属性、オブジェクトの持つ操作である。オブジェクトモデルは、動的モデルや機能モデルを配置するための本質的な枠組を与える。というのも、動的モデルや機能モデルで表現される変化や変換は、その対象となるべき何ものかが存在して初めて意味を持つからである。

オブジェクトモデルの構成における目標は、アプリケーションにとって重要な概念を実世界から捕まえてくることである。工学的な問題のモデル化においてはオブジェクトモデルは技術者の楽しんでいる用語を含むことになるだろう。ビジネス分野のモデル化においてはビジネスの用語、ユーザインターフェイスのモデル化においてはアプリケーション領域の用語が含まれることになろう。オブジェクトモデルはオブジェクトクラスを含むオブジェクト図によって視覚的に表現される。クラスは、共通の構造や振舞いを共有するために階層として整理されると同時に、他のクラスとも関連付けられる。クラスは各オブジェクトインスタンスの保持する属性値と各オブジェクトが実行/実施すべき操作を定義する。

3.2.2 動的モデル

動的モデルは時間と操作の順序にかかわるシステムの側面を記述する。それは変化を促す事象、事象例、事象にとっての文脈を定義する状態、そしてそれら事象と状態とがどう組み合わさっているかからなる。動的モデルは「制御」すなわち起こるべき操作の例を記述するというシステムの側面をとらえたものである。それは操作が何を行なうかとか、それらが作用する対象はなにかとか、それらがどのように実装されているかに関係しない。

動的モデルは状態図によって視覚的に表現される。各状態図はオブジェクトの１つクラスに対してシステム中で許されている状態と実装例を示している。状態図はまた他のモデルを参照する。状態図中の動作は機能モデル中の機能に

対応しているし、状態図中の事象はオブジェクトモデル中のオブジェクトに対する操作になる。

3.2.3 機能モデル

機能モデルは値の変換にかかわるシステムの側面を記述する。それは関数、写像、制約そして機能（関数）依存性といったものからなる。機能モデルはシステムが何を行なうかについて、それがいつかにして行なわれるかとは無関係に表現したものである。機能モデルはデータフロー図で表現される。データフロー図は機能がいつ実行されるのかに関することは抜きにして、値の間の依存性、入力値からの出力値の計算、そして機能を示している。演算式のパーズ木のような伝統的な計算の概念は機能モデルの例になっているし、あまり伝統的でない概念であるがスプレッドシートのようなものも機能モデルの例である。機能は、動的モデルの中では動作として起動されるし、オブジェクトモデルにおいてはオブジェクトに対する操作として示される。

本研究で開発しているシステムは根本的にOMTによるオブジェクト指向方法論に基づいて設計されているが、場合によって独特な方法論や他の方法論などを利用することもある。

3.3 システムアーキテクチャの設計

本研究のシステムはユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 から構成されている(図 3.2 参照)。

1. **ユーザインタフェイス部**
   本システムとユーザとの対話が、ユーザインタフェイス部で行なわれる。ユーザインタフェイス部では、オブジェクト指向設計のエディタやそのシステムの処理結果を表示するための機能を管理する。

2. **コミュニケーションエンジン部**
   ユーザインタフェイス部とシステムとの間のコミュニケーションがコミュニケーションエンジン部で行なわれる。本システムでは、分散オブジェクト CORBA を利用して、ユーザインタフェイス部とシステムとの通信をやりとる。

3. **推論エンジン部**
   推論エンジン部は、システムのカーネルになり、分散型知識ベース部に蓄積しているルールとユーザが入力する依頼を推論する役割をもっといる。つまり、推論エンジンでは、ユーザインタフェイス部からもらったオブジェクト指向設計を分散型知識ベースにある設計ルールと比較して、対応するルールがあったら、そのルールを呼び出して利用して、依頼を処理する。

4. **分散型知識ベース部**
   分散型知識ベースは、いくつかの知識ベースから構成され、KQML で各知識ベースを話し合う。分散型知識ベース部の窓口として、知識ベースサーバである。もし推論エンジンが検索するルールを知識ベースサーバで見つけることができなかったら、知識ベースサーバが知識ベースクライアントに検索依頼を送り、結果をもらう。

5. **ドキュメンテーションエンジン部**
   ドキュメンテーションエンジン部では、推論エンジンからもらった結果に対応するドキュメントを追加して、その結果をユーザインタフェイス部に送る。その他、ユーザがオブジェクト指向設計を行うとき、その設計プロセスを自動的に残してドキュメントとして保存する。設計プロセスより新しいルールを発見すれば、再利用することが出来るように編集して、知識ベースに蓄積するという役割を持っている。

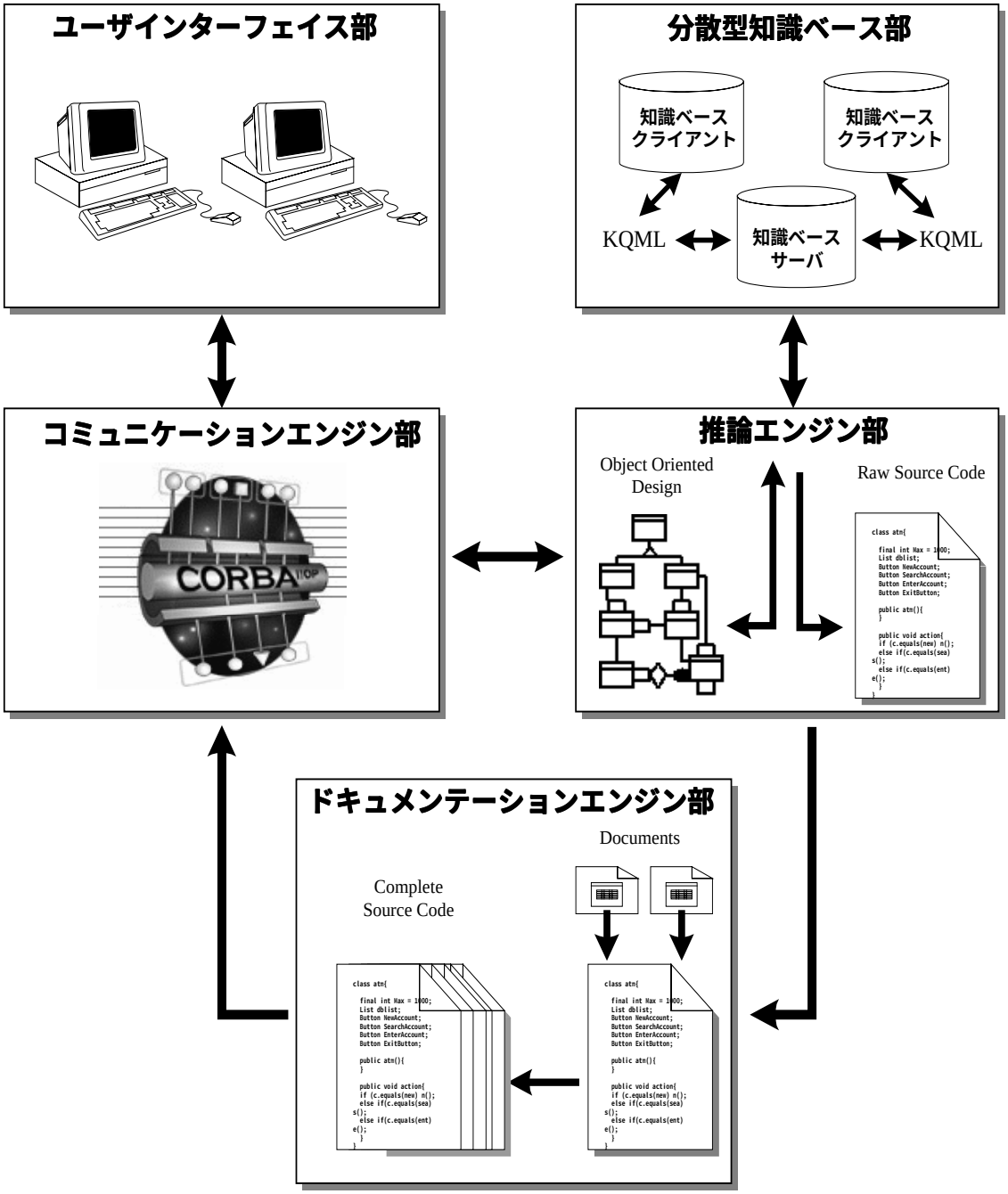

図 3.2 : システムアーキテクチャの設計

3.4 オブジェクトモデルの設計

システムアーキテクチャーをオブジェクトモデル化すると、図 3.3 のオブジェクトモデルのようになる。

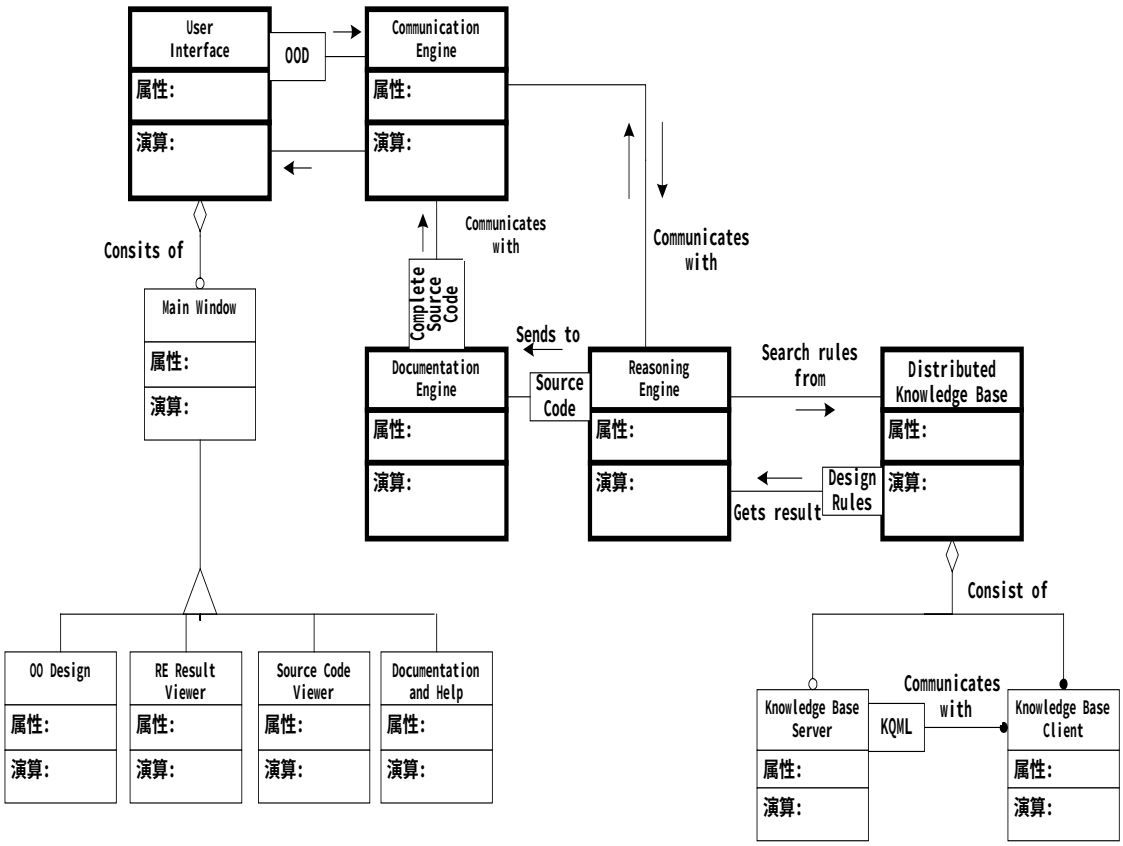

図 3.3： オブジェクトモデルの設計

3.5 動的モデルの設計

本システムの流れを図 3.4 の動的モデルのようになる。

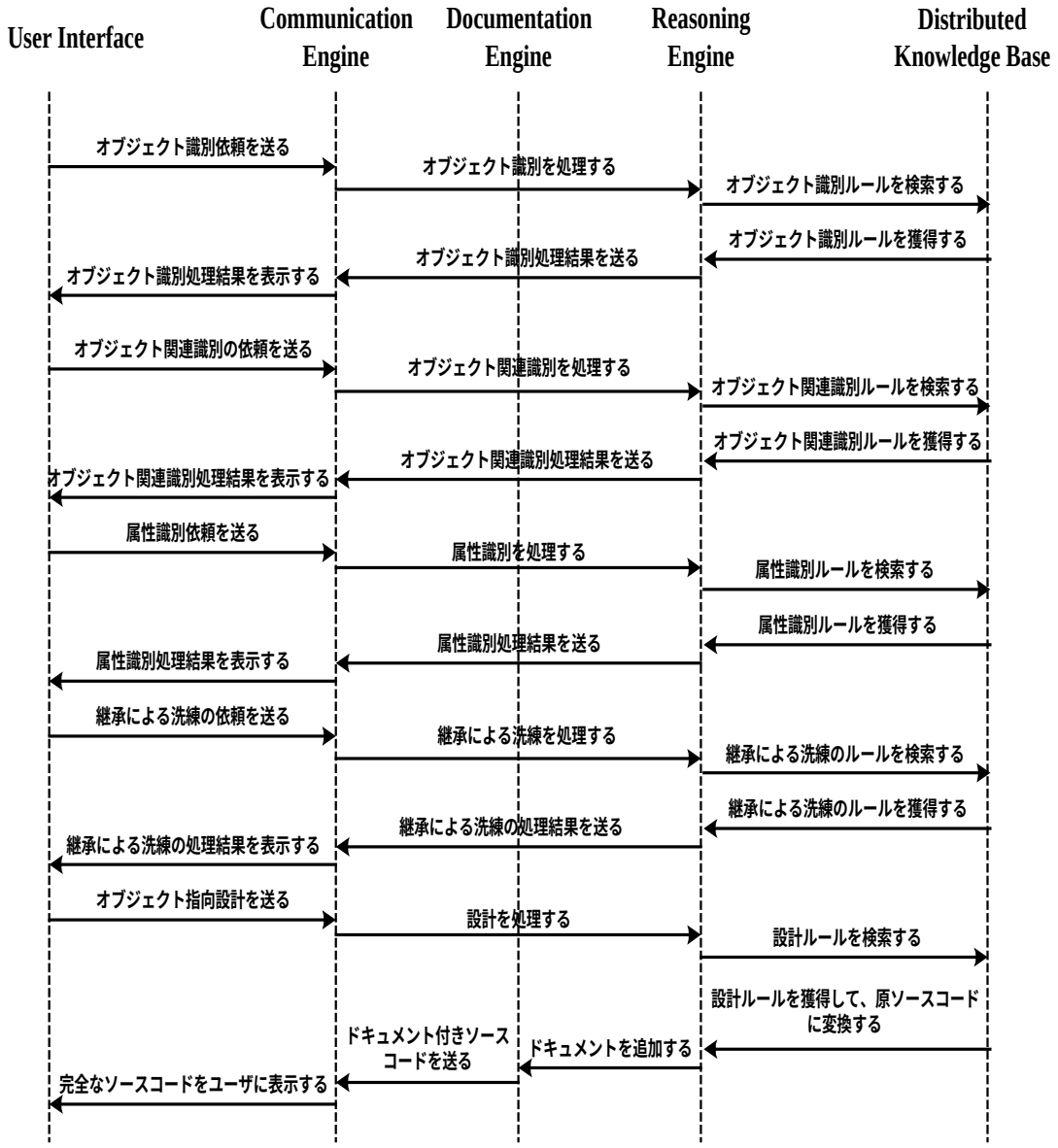

図 3.4：動的モデルの設計

図3.4に示すように、ユーザがシステムのユーザインタフェイスを利用して、

オブジェクト指向設計の依頼をシステムに送る。その依頼は、

1. **オブジェクト識別**
2. **オブジェクト関連識別**
3. **属性識別**
4. **継承による洗練**
5. **オブジェクト指向設計のソースコード化**

である。

Documentation Engine と Reasoning Engine が分散型知識ベースを利用して、オブジェクト識別、オブジェクト関連識別、属性識別、継承による洗練、オブジェクト指向設計のソースコード化という依頼を処理して、その処理結果をユーザに提案して表示する。

3.6 機能モデルの設計

システムの機能に関しては、図 3.5 の機能モデルのようにモデル化している。ユーザがまずコミュニケーションエンジン部を経由して、オブジェクト識別依頼を送ります。推論エンジン部が分散型知識ベース部でのルールを検索して、結果を処理して、ユーザに送ります。関連の識別、属性の識別、継承による洗練の依頼も同様である。最後にオブジェクトモデルを完成したら、ソースコード化依頼も送ってその結果をユーザに表示する。

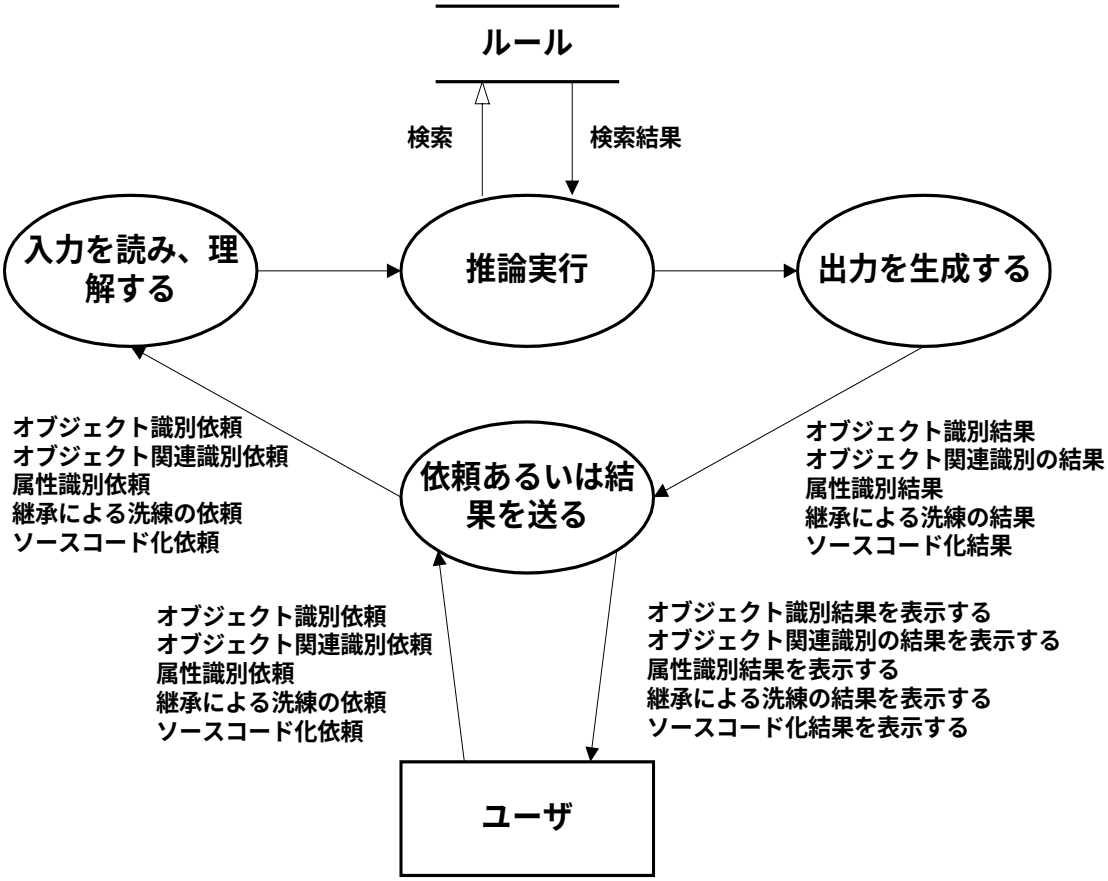

図 3.5： 機能モデルの設計

3.7 本章のまとめ

本章には、システムの分析と設計について述べた。本システムは根本的にOMT によるオブジェクト指向方法論に基づいて設計されている。OMT(Object Modeling Technique)法は、1992 年に Rumbaugh ら[7]に提案されたオブジェクト指向方法論の１つである。Rumbaugh らは３つの関連しているが異なった視点からシステムをモデル化することが有効であることを発見した。それは、オブジェクトモデル、動的モデルと機能モデルである。オブジェクトモデル化技法（OMT）はシステムをモデル化するための、これら３つの見方を組み合わせた方法論に対して Rumbaugh らが付けた名前である。

オブジェクトモデルはシステムの静的、構造的、「データ的」な側面を表現している。**動的モデル**はシステムの時間的、動作的、「制御的」な側面を表現している。**機能モデル**はシステムの変換的、関数的、「機能的」な側面を表現している。

本研究で開発しているシステム全体構成は、ユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 からなる。本章には、その全体的なシステムの分析と設計について説明した。

第4章 各エキスパートユニットの分析と設計

本章には、システムの各エキスパートユニットの分析と設計について述べる。本システムは、ユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 から構成されている。各部をエキスパートユニットと呼ぶ。本章に、ユーザインターフェイス部をはじめ、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部の詳しくその分析と設計について説明する。

4.1 ユーザインターフェイス部

本システムとユーザとの対話が、ユーザインタフェイス部で行なわれる。ユーザインタフェイス部には、次のような機能を整える。

1. オブジェクト指向設計のエディタ
2. 推論エンジン処理結果のビューア
3. Java ソースコードのビューア
4. ドキュメンテーションとヘルプ

4.1.1 オブジェクト指向設計のエディタ

オブジェクト指向設計のエディタでは、UML、OMT、Booch、 Shlaer-Mellor、Yourdon-Coad など様々なオブジェクト指向方法論を扱うことができると予定されているが、今年は OMT 方法を重視にして分散型知識ベースシステムを開発している。OMT のオブジェクトモデル記法は、図 4.1 のように詳しく示されます。

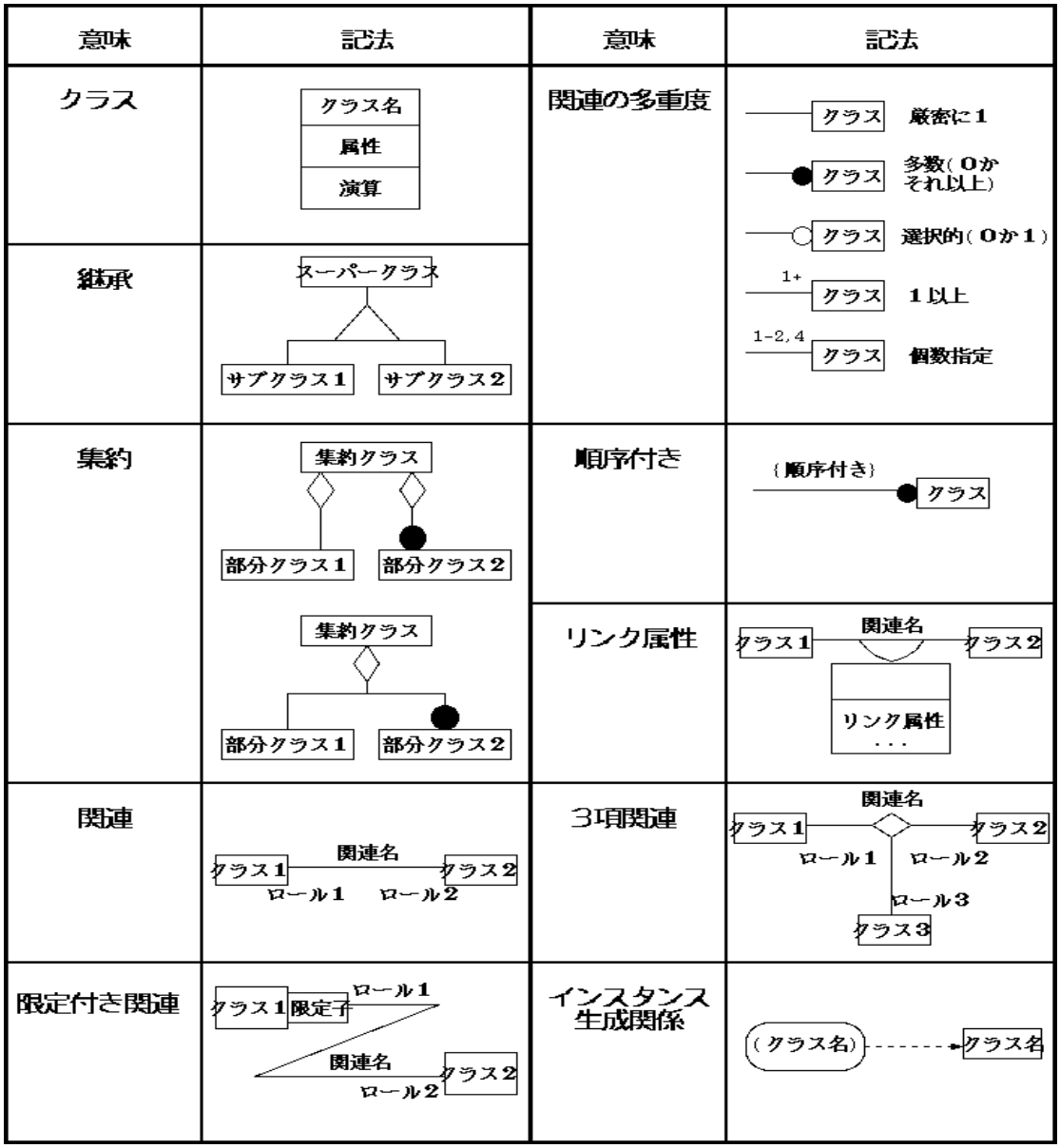

図 4.1 : OMT のオブジェクトモデル記法

オブジェクト指向設計エディタ機能は、オブジェクトモデルを設計するために、図 4.1 に示す記法を管理している。

4.1.2 推論エンジン処理結果のビューア

推論エンジン部では、ユーザインタフェイス部からもらったオブジェクト指向設計を分散型知識ベースにある設計ルールと比較して、対応するルールがあったら、そのルールを呼び出して、依頼を処理する。その処理結果は、ユーザに表示するために、処理結果のビューアという機能を付けなければならない。推論エンジン処理結果のビューアは、システムが提案するより良いオブジェクト指向設計と具体的な説明のビューアあるいはダイアローグフレームみたいな機能を管理する。

4.1.3 Java ソースコードのビューア

ユーザとシステムが、オブジェクト指向設計に関する問題を解決することができたら、最後にユーザが推論エンジンにソースコード化の依頼を送って、その結果を Java ソースコードのビューアに表示する。Java ソースコードのビューアでは、クラス木とメソッド木があって、その他具体的な全ソースコードも見れるようにする。

4.1.4 ドキュメンテーションとヘルプ

ドキュメンテーションとヘルプ機能に関しては、全体的なシステムマニュアルとそのシステム動作のトリックを表示することが出来るようにする。マルチプラットフォームに対応できるように html フォーマットと Windows の hlp フォーマットを両方作成する必要がある。

上記の分析を基づき、ユーザインタフェイス部は図 4.2 のように設計されている。

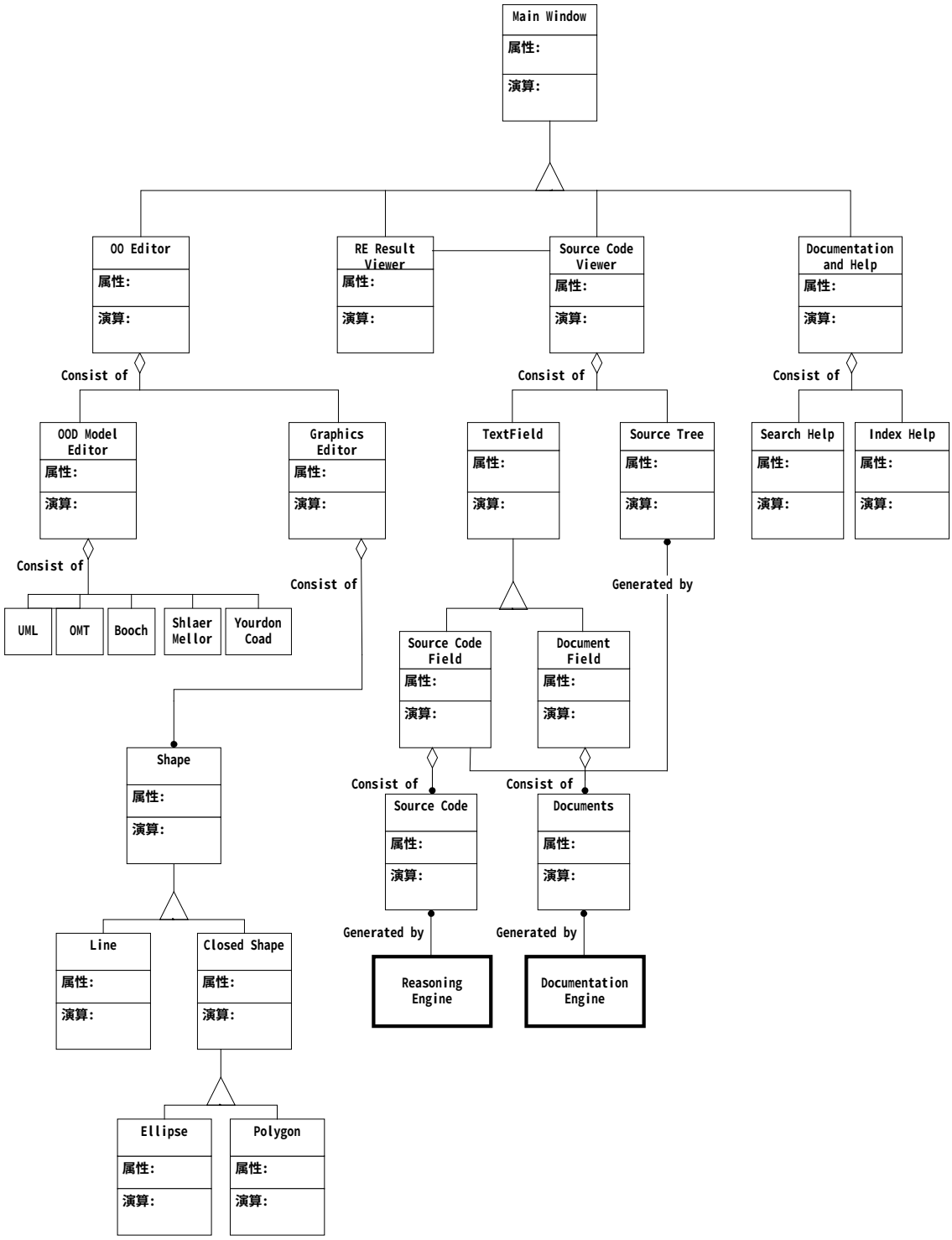

図 4.2：ユーザインターフェイス部の設計

4.2 コミュニケーションエンジン部

ユーザインタフェイス部とシステムとの間のコミュニケーションがコミュニケーションエンジン部で行なわれる。本システムでは、分散オブジェクト CORBA を利用して、ユーザインタフェイス部とシステムとの通信をやりとる。

分散オブジェクト CORBA の基盤を構成するのは基本的に下記の４つである。

1. 分散オブジェクト CORBA のオブジェクトバスを定義する ORB（Object Request Broker）
2. バスを拡張するためのシステムレベルのオブジェクトフレームワークを定義する CORBA サービス
3. ビジネスオブジェクトが直接使用する水平型と垂直型のアプリケーションフレームワークを定義する CORBA ファシリティ
4. ビジネスオブジェクトであり、かつアプリケーションでもあるアプリケーションオブジェクト

ORB はオブジェクトバスであり、ORB を用いることにより、各オブジェクトからローカルあるいはリモートのオブジェクトに対して透明的に要求を出したり、これらのオブジェクトからの応答を受け取ることが出来る。クライアントはサーバオブジェクトとの通信やサーバオブジェクトの起動あるいは保管にどのような仕組みが用いられるのがしらなくてもよい。ORB は非常に充実した分散型ミドルウエアサービスを提供する。ORB を用いることにより、実行時にオブジェクトがお互いを見つけ出し、お互いのサービスを呼び出すことが出来る。

CORBA サービスはシステムレベルのサービスの集合であり、IDL で規定されたインタフェイスを備えている。オブジェクトサービスは ORB の機能を拡張し、補うものと解釈すればよい。これらのサービスを利用することにより、コンポーネントを作成し、それに名前を付け、それが使用される環境に公開することが出来る。

CORBA ファシリティは IDL で記述されたフレームワークの集合であり、アプリケーションオブジェクトは CORBA ファシリティの提供するサービスを直

接利用することできる。

分散オブジェクト CORBA に基づき、コミュニケーションエンジン部は、図 4.3 のように設計されている。

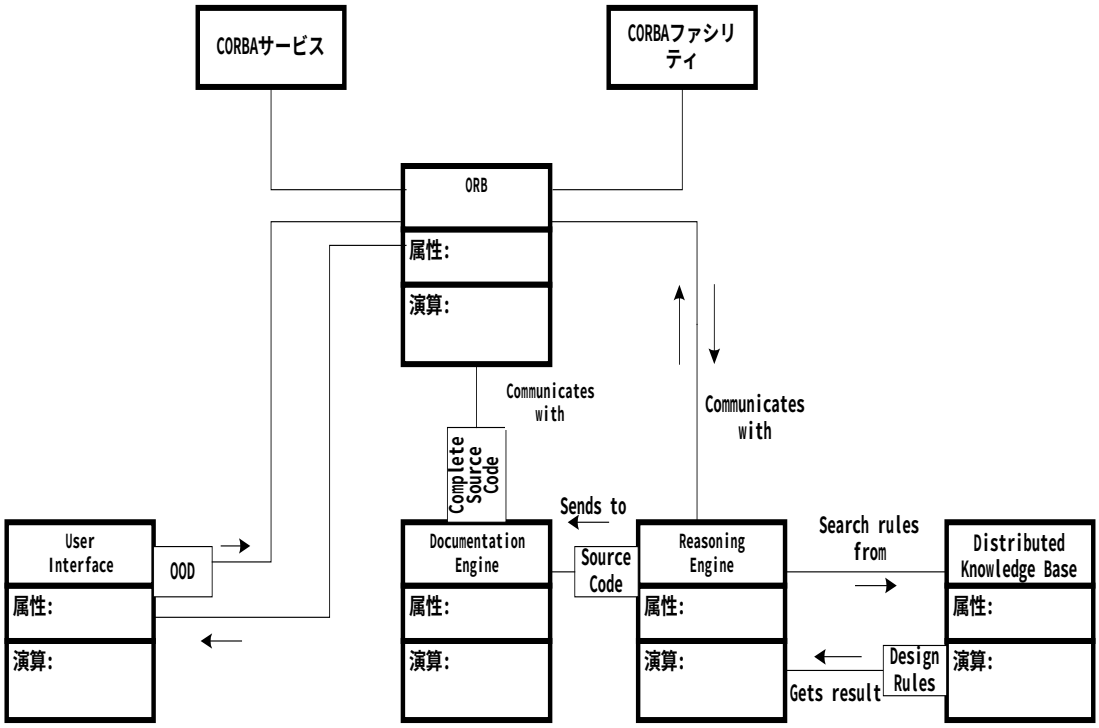

図 4.3：コミュニケーションエンジン部の設計

4.3 推論エンジン部

推論エンジン部は、システムのカーネルになり、分散型知識ベース部に蓄積しているルールとユーザが入力する依頼を推論する役割をもっといる。つまり、推論エンジンでは、ユーザインタフェイス部からもらったオブジェクト指向設計を分散型知識ベースにある設計ルールと比較して、対応するルールがあったら、そのルールを呼び出して利用して、依頼を処理する。

推論エンジン部は、オブジェクト指向設計の推論とソースコードの推論という機能を管理している。オブジェクト指向設計の推論とは、ユーザから送ったオブジェクト指向設計を分散型知識ベース部に蓄積しているルールを参考にして、チェック･推論する。つまり、オブジェクト指向設計の推論は、

1. **オブジェクト識別**
2. **オブジェクト関連識別**
3. **属性識別**
4. **継承による洗練**

という依頼の担当になる。

一方、ソースコードの推論は、

オブジェクト指向設計のソースコード化

という依頼の担当となり、完成したオブジェクト指向設計を Java ソースコードへ変換する役割をもっている。

推論エンジン部は、図 4.4 のように設計されている。

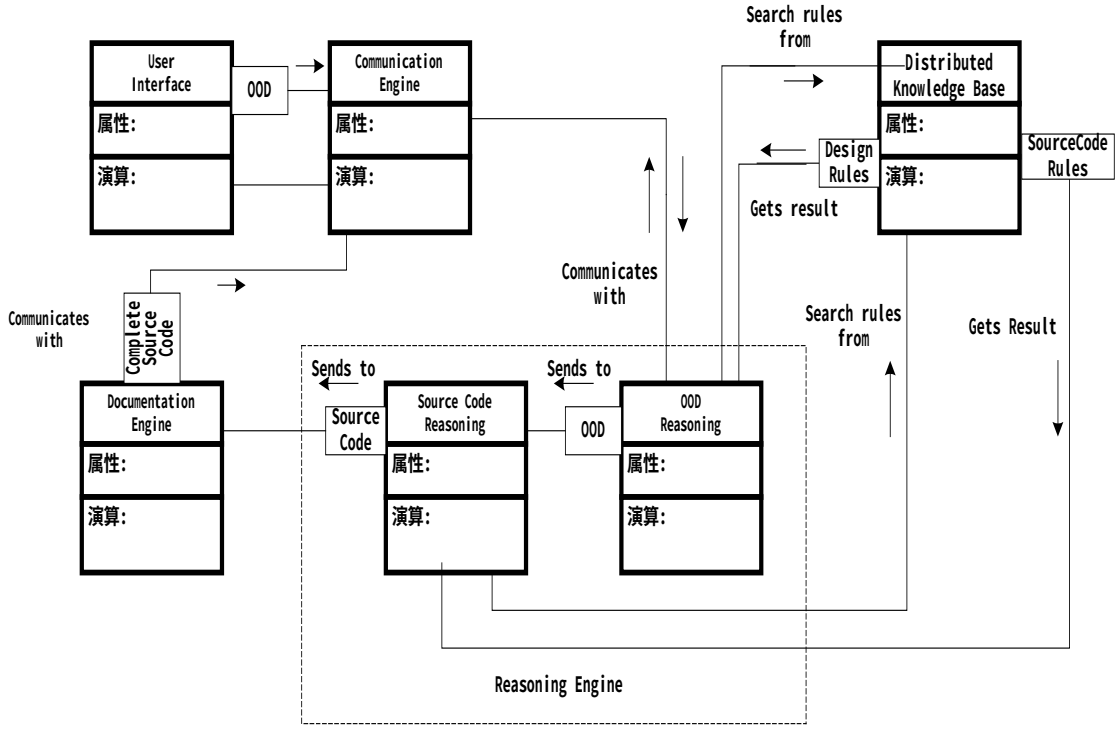

図 4.4：推論エンジン部の設計

4.4 分散型知識ベース部

分散型知識ベースは、いくつかの知識ベースから構成され、KQML で各知識ベースを話し合う。分散型知識ベース部の窓口として、知識ベースサーバである。もし推論エンジンが検索するルールを知識ベースサーバで見つけることができなかったら、知識ベースサーバが知識ベースクライアントに検索依頼を送り、結果をもらう。

推論エンジンを対応することできるために、２種類ルールを分散型知識ベース部に蓄積する。それは、

1. **オブジェクト指向設計用ルール**
2. **ソースコード化用ルール**

である。

分散型知識ベース部の設計は、図 4.5 のように設計されている。

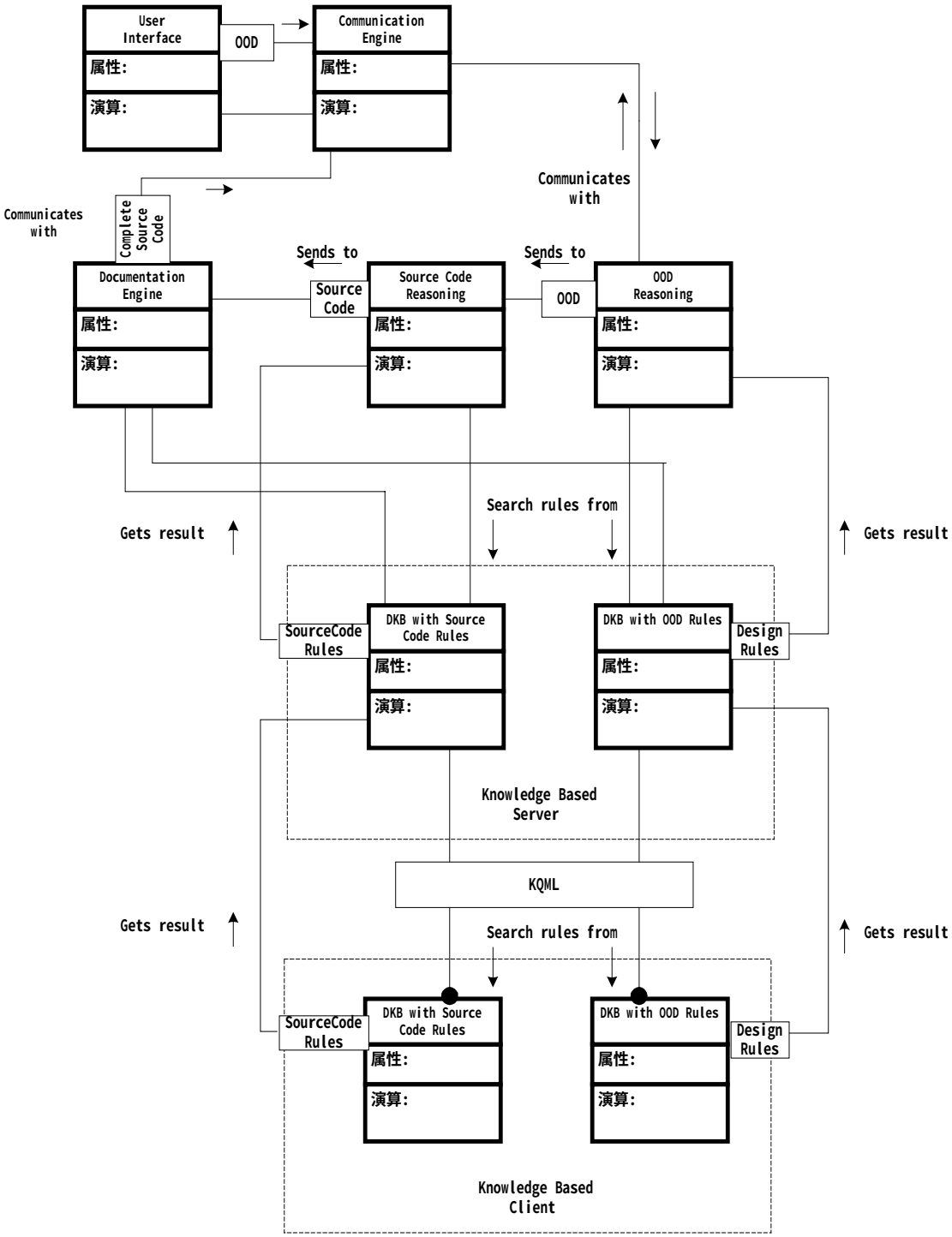

図 4.5： 分散型知識ベース部の設計

4.5 ドキュメンテーションエンジン部

ドキュメンテーションエンジン部では、以下の役割を持っている。

1. 推論エンジンからもらったソースコードに対応するドキュメントを自動的に追加･生成して、その完成したソースコードをユーザインタフェイス部に送る。
2. ユーザがオブジェクト指向設計を行うとき、その設計プロセスを自動的に残してドキュメントとして保存する。設計プロセスより新しいルールを発見すれば、再利用することが出来るように編集して、知識ベースに蓄積する。

ドキュメンテーションエンジン部は、図 4.6 のように設計されている。

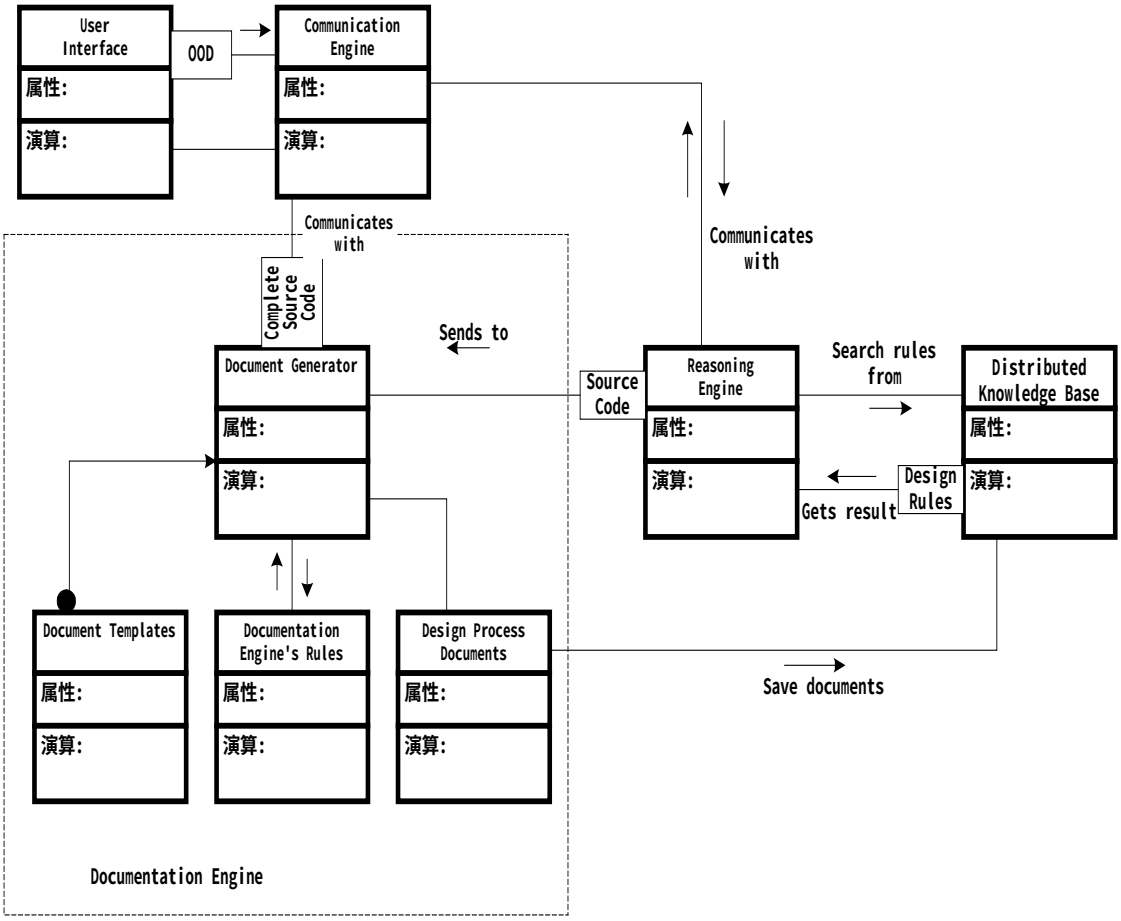

図 4.6：ドキュメンテーションエンジン部の設計

4.6 本章のまとめ

本システムは、ユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 から構成されている。各部をエキスパートユニットと呼ぶ。本章には、システムの各エキスパートユニットの分析と設計について述べた。

第5章 分散型知識ベースシステムの実現用枠組み

本章には、設計されたシステムを実装・実現するために、採用した技術やアプローチなどについて詳しく述べる。本システムは、オブジェクト指向言語、分散システムに対応、シンプルな言語仕様、マルチプラットフォーム、言語仕様のセキュリティ機構などの理由で、Java言語を採用して実現される。そのほかコミュニケーションエンジン部では、分散オブジェクトを用いて、データをやりとることが可能な分散システム型を実現する。

5.1 Java 言語の採用

本研究では、設計するシステムを実装するために、Java プログラミング言語を利用する。さらに、本研究のソースコード自動生成するところも Java 言語のソースコードを採用している。その理由は次のようになる。

1. **オブジェクト指向プログラミング言語**
   オブジェクト指向プログラミング言語を利用によって、システムの拡張や変更など容易になる。

2. **分散システムに対応している**
   現在世の中にあるオブジェクト指向言語と比べると、Java 言語が分散システムには完全に対応している。それは、Java 言語はもともと分散システム

向けプログラミング言語であるので、対話性・拡張性・安全性の特徴を持つ分散システムの実現も可能になる。本研究では、分散システムを実現するので、Java 言語は的確な選択と思っている。

3. シンプルな言語仕様

Java 言語は C＋＋言語によく似た表記を使うプログラムであるが、ポインタ型がないなど様々な点で C＋＋言語とは異なっている。オブジェクト指向という観点で見れば中途半端な存在であった構造体もなくなっている。Java 言語では数値や論理型などの基本型を除けばすべてオブジェクトと極めてシンプルな言語設計を採用している。このように Java 言語は言語仕様を単純化することで、本研究のシステム設計や知識ベースの設計ルールのところでは、容易になる[6]。

4. マルチプラットフォーム

Java 言語でプログラムを書くと、Unix マシンや PC など様々なプラットフォームで実行できるようになっている。それは、プログラマ側では、Java 言語を利用すると、開発コストを削減することができる。一方ユーザ側では、他のプラットフォームバージョンのソフトウエアを購入する必要がなく、１つプログラムでどんなマシンでも実行できる。

5. マルチスレッドに対応している

１つのアプリケーションプログラムのの中で複数の処理を並行動作させることをマルチスレッドと呼ぶ。

6. 言語仕様レベルのセキュリティ機構を持っている

Java 言語は言語仕様レベルからセキュリティについて考慮されている。例えば、Java 言語にはポインタ型がないため、プログラマが意図的にシステムメモリを操作できない。バイトコードファイル実行時に Java インタープリタは変数を正しい型で使っているかなど、バイトコードに不審な点がないことをチェックした上で実行する。また、ネットワークから読み込んできたプログラムからはユーザのファイルは操作できないようにしており、ユーザのファイルの内容がユーザの知らない間に書き変わらないようにしている。このように Java 言語では様々なレベルで何重ものセキュリティ機構を持つことで、ネットワークを通じて転送したプログラムの実行という一見非常に危険に思われる仕組みを安全に実現できるようにしている[6]。

7. **独自のガベージコレクション機能をもつ**
   他のオブジェクト指向言語では、delete 演算子などを使って、ユーザが明示的にメモリ管理をする必要があったが、Java 言語では使われなくなったメモリは Java 言語インタープリタが自動的に解放してくれるので、ユーザはメモリ管理をする必要がなくなる。そのような機能はガベージコレクションと呼ばれる。ガベージコレクション機能をもつことで、プログラムのバグを減少することができる。

5.2 オブジェクト指向方法論の採用

本研究では、構造化あるいは手続きアプローチを利用せず、オブジェクト指向アプローチを採用する理由は、次のようになる。

- **オブジェクト指向分析から実装まで一貫した関連があり、ギャップは全く存在しない**
  構造化パラダイムによるソフトウエア開発プロセスでは、プログラミングフェイズで用いる構造化プログラミングや構造化チャートといった技法への連動がうまく取れなくて、そのフェイズの間にギャップが存在する。

- **オブジェクト指向方法論では、プログラムとデータベースとのインピーダンス・ミスマッチが存在しない**
  構造化パラダイムによるソフトウエア開発プロセスでは、プログラムとデータベースとの親和性が弱く、インピーダンス・ミスマッチが生じる[1]。例えば、データベースのスキーマ設計に構造化パラダイムを適応することはできない。

- **オブジェクト指向方法論を利用すると、機能追加及び保守の生産性が向上することができる**
  対象世界に近いモデル化やカプセル化により、変更範囲が局所化され、機能追加の生産性が向上する。

5.3 分散知識ベースアプローチの採用

本研究では、設計ルールを蓄積するため、分散型知識ベースを利用する。その理由は、次のようになる。

- 大規模知識ベースを取り扱うため、分散化アプローチを採用した方が効率的な知識ベースを得られる[2、3]。
- コンピュータネットワーク及びクライアント・サーバシステム技術を提供することができる。
- 地理 的に離れた場所からも設計知識を利用できる。
- 大勢の人が設計知識を共同で蓄えることによって、利用できる知識が増える。

5.4 分散オブジェクトの採用

分散オブジェクトは、リモートからアクセスすることができるオブジェクトである。この事が意味するのは、分散オブジェクトは普通のオブジェクトと同じように、ネットワーク上のどこからということは関係なく、使うことができるということである。

オブジェクトは、一般的にデータと振舞いをカプセル化したものであると考えられる。分散オブジェクトのある場所は、オブジェクトのユーザにとって問題ではない。分散オブジェクトは、ユーザにオブジェクトに関係する機能のセットを供給する。機能のセットを供給するアプリケーションは、しばしばサービスと呼ばれる。ビジネスオブジェクトは、ローカルオブジェクトまたは分散オブジェクトになるかもしれない。ビジネスオブジェクトという用語は、あるビジネスプロセスに関連した仕事の集合を行うオブジェクトを指す。

本システムのコミュニケーションエンジン部では、分散オブジェクトを採用する。分散オブジェクトを利用する利点は以下のようにあげられる、

1. **柔軟なシステムを構築できること**

分散システムの特徴は、多くのマシンから構築され、アプリケーションはい

くつかのマシンにまたがって動作したり、他のアプリケーションと連携するということである。このようなシステムでは、マシンの移動や拡張、追加、アプリケーションの追加及び変更といったシステム変更に対する柔軟性が必要である。分散オブジェクトを使った分散システムは、このようなシステム構成の変更にも柔軟に対応できる。具体的に、

① システムの拡張/変更/組合せに伴って、連携された個々のアプリケーションを修正する必要はない。例えば、システムを変更しても、クライアント側の環境の定義の変更は必要ない。

② アプリケーションの配置が分散システム上で自由に選択できる。

2. マルチプラットフォーム

分散システムは、Windows、 Macintosh、 Unix など様々なマシン/OS を組み合わせて構築される。各マシン上で動作しているアプリケーションをすべて１つのシステムとして扱うためには、これらすべてのマシンをカバーするような分散環境が必要である。分散オブジェクトは、ワークステーションやパソコンなど様々なプラットフォーム上で動作する製品が提供されている。

3. インターオペラビリティ

異なる ORB 間のインターオペラビリティも保証できるため、ORB を使ってプラットフォームやヘッダに依存しないシステムを構築することができる。

4. スケーラビリティ

カプセル化によってもたらされるスケーラビリティである。オブジェクト同士のインタフェースさえ明確に定義しておけば、 それぞれのオブジェクトのインプリメンテーションは自由である。 後に、 改良や強化が必要になっても、 他のオブジェクトをプログラミングし直す必要はない。インタフェース仕様さえ守れば、 新たに増設したコンピュータにオブジェクトを配置し直してシステム全体の性能向上を図ることが可能である[4]。

5. アプリケーションまたはユーザの間で情報の共有のために使うことができる

6. いくつかのマシンの間で動作を同期させるために使うことができる

7. 特定の仕事のパフォーマンスを向上させることに使うことができる

8. 異なる町の人達が、共同してある特定のビジネスプロセスに参加することを可能にする

分散オブジェクトを開発する組織は、OMG と Microsoft である。Microsoft は、1996 年から Microsoft 製品しか提供することができない DCOM を開発している。一方 OMG は、1990 年から Microsoft と非 Microsoft 製品を提供することができる CORBA を開発している。本研究では、マルチプラットフォームをサポートする、安定した分散オブジェクト CORBA を採用した。

5.5 プログラムの全体構成と実現状況

作成するプログラムの構成とその実現状況は、図5.1のようになっている。本システムは、ユーザインタフェイス部、コミュニケーションエンジン部、推論エンジン部、分散型知識ベース部、ドキュメンテーションエンジン部 から構成されており、それをエキスパートユニットと呼ぶ。各エキスパートユニットは、パッケージとして実現される。現在コーディング化している部分は、コミュニケーションエンジン部とユーザインタフェイス部である。他の部分は、未完成となる。

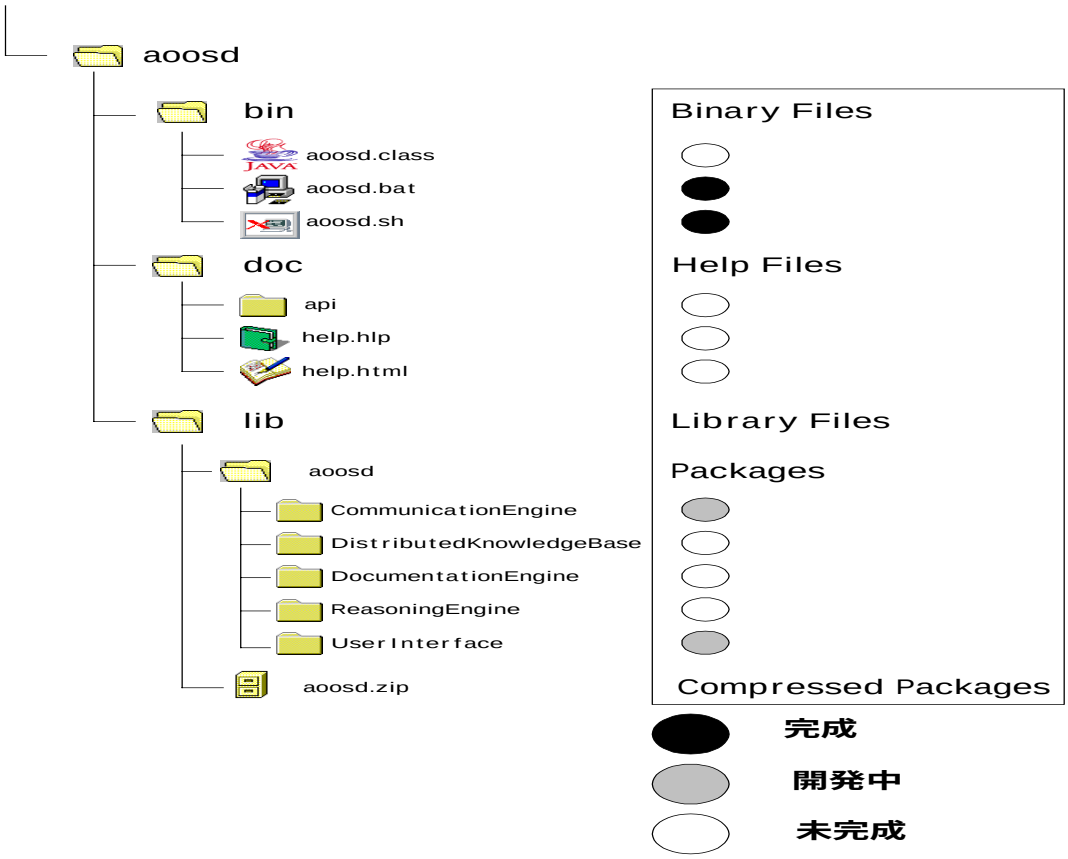

図 5.1： プログラム全体構成と実現状況

5.6 本章のまとめ

本章には、設計されたシステムを実装・実現するために、採用した技術やアプローチなどについて述べた。本システムを、Java 言語で実現する。その理由としては、

1. オブジェクト指向プログラミング言語
2. 分散システムに対応している
3. シンプルな言語仕様
4. マルチプラットフォーム
5. マルチスレッドに対応している
6. 言語仕様レベルのセキュリティ機構を持っている
7. 独自のガベージコレクション機能をもつ

である。そして、コミュニケーションエンジン部は、分散オブジェクトを用いて、データをやりとることが可能な分散システム型として実現される。本章には、システムを実現するための技術やアプローチとその理由などについて詳しく述べた。

第6章 結論と今後の課題

本研究は、オブジェクト指向方法論によるソフトウエア自動設計のための設計知識抽出方式と、それを再利用する分散型知識ベースシステムの研究である。本研究の最終目的は、ソフトウエアの生産性及び信頼性の向上を狙い、オブジェクト指向方法論によるソフトウエア自動設計を行なうために分散型知識ベースシステムを実現する確立にある。現在「オブジェクト指向分析設計プロセスの問題、新たな分散 CASE システムへの要求、大規模知識ベースシステムでの問題」という状況があって本研究の背景となった。

上記のような問題を検討・研究した後、下記のような研究アプローチを利用して、オブジェクト指向ソフトウエア自動設計用分散型知識ベースシステムを開発する必要があると結論した。

1. オブジェクト識別・関連の識別・属性の識別・継承による洗練での困難に対するアプローチは、まずそのプロセスに対応するルールを求めて、知識ベースに蓄積する。そしてそのルールに基づいて、困難な設計プロセスを自動化する。
2. 分散オブジェクトを用いる分散システムアプローチを採用する。その理由は、まず、グループウエアでの情報の共通や相互運用性の問題を解決できる。そのほかスケーラビリティ、可搬性、柔軟性を持つシステムを実現することができるからである。
3. 分散型知識ベースアプローチを採用する。それは、効率的な知識ベースであり、ネットワーク及び C/S システム技術を提供でき、地理的に離れた場所からも設計知識を利用でき、知識獲得ボトルネックの問題を解決できるという理由で採用した。

今後の課題として、

1. 分散型知識ベース用の設計ルール及び設計パターンを求める。
2. 本論文でまとめた分散型知識ベースシステムの分析と設計を基づき、各エキスパートユニットを実現する。
3. 各エキスパートユニットを統合して、全体的にシステムを実現する。
4. システムのテストとその評価を行う。

謝辞

研究ならびに生活面において御指導を賜りました B.H.Far 助教授、研究の遂行にご協力頂きました河野善彌教授、有益な御助言をいただいた陳慧助手、本当にありがとうございました。

また、先輩としていつも良きアドバイスを下さいました Sidi 氏、El-Khouly 氏、Hajji 氏、Shadan 氏、Hashimoto 氏、丸山和幸氏、上野栄一氏、そして苦労を共にした園城浩行氏、合田寿彦氏、野村行広氏、そして同期学生の皆様、並びに私を暖かく見守って頂いた、両親をはじめとする周囲の全ての人々に深く感謝致します。

参考文献

[1] 河村一樹、“ソフトウエア工学入門”、*啓学出版*、1993

[2] 西田 豊明、“ソフトウエアエージェント”、*人工知能学会誌*、Vol.10 No.5、1995

[3] 西田 豊明、“大規模知識ベースシステム”、*情報処理*、Vol.35 No.2、1994

[4] 清水降文、三輪 芳久、真島 馨、“分散オブジェクトを検討する”、*日系バイト*、７月、1997

[5] Rob Appelbaum、Marshall Cline、and Mike Girou、“The CORBA FAQ”、*http://www.cerfnet.com/~mpcline/corba-faq*、1996

[6] 有我成城、衛藤敏寿、佐藤治、白神一久、西村利浩、村上列、“一歩先行くインターネット・Java 入門”、*SHOEISHA*、1996

[7] Rumbaugh J.、Blaha M.、Premerlani W.、Eddy F. and Lorenson W.、“Object Oriented Modeling and Design”、*Prentice Hall*、1990

[8] Erich Gamma、Richard Helm、Ralp Johnson、John Vlisside、“Design Pattern-Elements of Reusable Object Oriented Software”、*Addison Wesley*、1994

[9] 太原育夫、“人工知能の基礎知識”、*近代科学社*、1988

[10] Brian Henderson and Sellers、“A Book of Object-Oriented Knowledge - Object-Oriented Analysis、Design and Implementation : A New Approach to Software Engineering”、*Prentice Hall*、1992

[11] Robert Orfali and Dan Harkey、“Client/Server Programming with Java and CORBA”、*John Wiley &Sons Inc*.、1997

[12] Joseph P.Bigus and Jennifer Bigus、“Constructing Intelligent Agents with Java”、*John Wiley & Sons Inc.*、1998

[13] 青山幹雄、“分散環境：新しい開発環境像を求めて”、*情報処理*、Vol.33 No.1、1992

[14] Drucker P.、“ The Coming of the New Organization ”、*Harvard Business Review*、Jan.-Feb. 1998、pp. 45-53、1988
[15] Jeremy Rosenberger、“ Sams' Teach Yourself CORBA in 14 Days ”、*Sams Publishing*、1998
[16] 加藤貞行、“ オブジェクト識別についての一考察とその効果 ”、*情報処理学会研究報告*、Vol.90、No.100、1998
[17] 宮崎善史、廣田豊彦、橋本正明、“ 自分自身を編集出来るオブジェクトモデルエディタ ”、*情報処理学会研究報告*、Vol.90、No.100、1998
[18] 垂水幸、“ グループウエアのソフトウエア開発への応用 ”、*情報処理*、Vol.33、No.1、January 1992
[19] 坂下善彦、“ 分散開発環境の事例と今後の展望 ”、*情報処理*、Vol.33、No.1、January 1992
[20] 本位田真一、山城明宏、“ オブジェクト指向分析・設計 ”、*情報処理*、Vol.35、No.5、May 1994
[21] N.R. Jennings、L.Z. Varga、R.P.Aarnts、J.Fuchs、P. Skarek、“ Transforming Standalone Expert Systems into a Community of Cooperating Agents ”、*Eng.Applic.Artificial Intelligent*、Vol.6、No.4、1993
[22] 石田亨、桑原和広、“ 分散人工知能（１）：協調問題解決 ”、*人工知能学会誌*、Vol.7、No.6、1992
[23] Behrouz H.Far and Zenya Koono、“ Ex-W-Pert System : A Web-Based Distributed Expert System for Groupware Design ”、*Expert Systems With Application*、Vol.11、No.4、1996
[24] Orlando Belo and Antonio Ribeiro、“ A Web-Based Framework for Distributed Expert Systems ”、*Proceedings of WWW National Conference*、1996
[25] 石塚満、小林重信、“ エキスパートシステム ”、*丸善株式会社*、1991
[26] James Martin and James J. Odell、“ Object-Oriented Methods : A Foundation ”、*Prentice Hall*、1995
[27] James Martin and James J. Odell、“ Object-Oriented Methods : Pragmatic Considerations ”、*Prentice Hall*、1996