修 士 論 文

形式仕様記述を用いた
テキストファイルの図式化の研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

手塚 隆之

2008 年 3 月

修 士 論 文

形式仕様記述を用いた
テキストファイルの図式化の研究

指導教官　東条敏 教授

審査委員主査　東条敏 教授
審査委員　島津明 教授
審査委員　鳥澤健太郎 准教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

610703 手塚 隆之

提出年月: 2008 年 2 月

概 要

これまでテキストで書かれている文字列を，自然言語処理である「形態素解析」「構文解析」「意味解析」という要素技術を用いて解析し，さまざまな方法で処理する研究が行われてきている．本研究では，テキスト文の解析結果をソフトウェア開発の世界で使われているプログラミング言語に置き換え，それを組み込みシステム開発でも導入されているUML（Unified Modeling Language の略）形式で表示する試みを行った．

形式言語とは，もっとも広義かつ素朴には，素となる記号（以下，素記号と呼ぶ）幾つかを使って，文法や構文規則など定められた規則に従って作られる記号の全体集合のことをいう．各種プログラミング言語も，素記号と構文規則によって定められた形式言語である．本研究では，形式仕様記述 ＝ 形式言語 ⊃ プログラミング言語として扱っている．

オブジェクト指向プログラミングにおける設計では，フローチャートなどで処理の手順を記述するより，クラスやオブジェクトの関係やそのインターフェイスを設計し把握することのほうが重要視されている．表記を統一し，開発者どうしの意思疎通を図るために，モデリング言語が必要となる．2007 年現在では，オブジェクト指向分析設計におけるモデル図の記法は，統一モデリング言語 (UML) が使われる事例がほとんどである．

本研究では入力として使うテキスト文として料理レシピの調理手順を用いた．料理の作成手順はプログラムの実行手順と類似性が高く，UML で言うところのユースケースに近いものがあるからである．料理レシピ文に書かれている文字列を解析し，その構造解析結果を形式言語の一種であるプログラミング言語に中間的に置き換えて，図式表現である UML へと変換を試みた．作成したアプリケーションの説明を交えて，どのように実装したかを紹介し，組み込みソフトウェア開発への適用可能性について述べる．

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景と目的

近年，組み込み機器のソフトウェア開発ではプログラムステップ数が増加の一途をたどっている．組み込み製品でも特に高機能化，ネットワーク化が必要な製品については，特に顕著である．代表的なものとして携帯電話，自動車，MFP（マルチ・ファンクション・プリンタ）などがあるが，これらは一つの製品の中にあるプログラムステップ数は100万ステップを遥かに越え，中には1000万ステップに達しようとしているものもある．このように開発が大規模化している中，高度な技術を要求されながらも開発期間は縮小傾向にあり，しかも複数機種の並行開発も行っているという非常に厳しい状況にある．

現在のソフトウェア工学の仕様記述法としてもっとも汎用的なスタイルはオブジェクト指向の手法であるが，組込みの世界でも同じ傾向を示している．従来の開発技術では対処できないこの状況に対応するため，オブジェクト指向開発技術が必要になり，それはモデリング主体の開発を意味している．ところが技術者を育てるためには時間とコストが必要になる訳で，特にオブジェクト指向開発可能な技術者不足が深刻化している．多くの企業では開発における分析工程から構築工程までの基本となる考え方であるオブジェクト指向とUML [1] を開発現場へ急速に導入を進めようとしているが，同様の問題があることが教育をする側からも報告されている [28].

オブジェクト指向に対するニーズの高まりの背景として挙げられるもの

- 新技術への適応
- 開発期間の短納期
- システムの高品質要求
- 様々な利害関係者との協業

このような背景から，ソフトウェア開発工程の上流部分の開発を支援でき，生産性をアップさせることができるツール類の必要性を感じている．また開発要員不足からオブジェクト指向開発に不慣れな人員が開発の現場に投入されることも起こっており，分析・設計品質を一定レベルまで確保する仕組みも必要になってきている．特に分散開発，海外リソースの活用などが盛んで，共通フォーマットによるドキュメントの共有も重要になっ

[1] Unified Modeling Language の略であり，1997.1 OMG にて標準化.

てきている．そこで本研究ではソフトウェアに対する要求文書（すなわちテキスト文）からUML のクラス図とシーケンス図を導出するものを試作することを検討した．開発段階のターゲットテキストとして，身近にある料理のレシピ文を使うことにした．

料理とは手続きを一定の時間順に記述したものである．手続き連鎖からある目的でプロダクトを作るという意味では料理とはプログラムの作成手順と似たものと考えることができる．この手続き化・時間順序化は何も料理に限ったことではない．さまざまなプロダクトの生成過程を統一的にソフトウェア生成過程になぞらえてプロセスを明快にするということは，次の二点で意味があると思われる．

- まずソフトウェア開発工程は世間で共通に定められた仕様記述方法があり，分野に特化した恣意的な書き方を排除することができる．
- またソフトウェアの開発工程記述とはそもそも手順の曖昧性を排除するものであり，すべての手続きがオリジナルの自然言語文による記述を上回る明快性で提示されていることが期待できる．

さて，現在のソフトウェア工学の仕様記述法としてもっとも汎用的なスタイルはオブジェクト指向の手法であると考えられる．すなわち料理のレシピの中に現れる食材や調理器具をオブジェクトと捉え，そのオブジェクトに対する操作を定義しておくことにより不合理な記述を排除し，かつ操作手順をオブジェクト間の操作のシーケンスとして記述することが考えられる．

このように本研究では入力として使うテキスト文として料理レシピの調理手順を用いることにしたが，料理の作成手順はプログラムの実行手順と類似性が高く，UML で言うところのユースケースシナリオに近いものがあると言える．そこで，料理レシピ文に書かれている文字列を解析し，その構造解析結果を形式言語の一種であるプログラミング言語に中間的に置き換えて，図式表現である UML へと変換を試みた．最終的には入力としてソフトウェア要求仕様を用いることができるようにして，生成されたクラス図とシーケンス図を，ソフトウェア開発現場のオブジェクト指向分析および設計に役立つようにしたいと考えている．

1.2　本論文の構成

本論文では，以上の背景ならびに目的から，料理レシピの構文解析の結果からUMLのクラス図とシーケンス図を導き出す手法を作成したアプリケーションを交えて提案する．

２章で料理分野を扱った過去の類似研究や，テキストからの視覚化に関する関連研究，UMLを生成する研究などについて紹介する．３章では「ソフトウェア工学からの視点」を取り入れたテキスト文解析結果からのオブジェクト指向分析，およびプログラミング言語への中間的置き換え方法を示し，そこからUMLへの変換について述べる．４章では，作成したアプリケーションのシステムの設計・仕様の紹介をする．５章では，作成したアプリケーションを実際に動かした結果について紹介し，考察を加える．最後に６章で本研究のまとめと問題点ならびに今後の課題について述べる．

第2章　関連研究

本章では，本論文に関連のある過去研究について紹介する．本論文ではテキスト文書を最終的に UML 形式に図式化しているが，変換対象としてレシピ文を用いている．従って次の観点から関連研究を調査した．

- レシピ文を解析対象としている研究
- テキスト文を解析する研究
- ソフトウェア開発支援系の研究

これらの関連研究を順次紹介する．

2.1　レシピ文を解析対象としている研究

本研究では日本語テキスト文を解析しているが，過去の研究でもさまざまな研究が行われている．浜田らは，テキストからの情報を画像・音声解析に反映させることで，実用的な精度のマルチメディア統合技術の実現を目指し，テキスト教材の付随する料理番組に着目し，料理映像とテキスト教材を対応付けシステム化している [11]．レシピの動作表現に対し基本動作辞書を用いることにより，アニメーションと結びつける研究も行われている [12]．西田 [13] は，料理教示発話分類として作業宣言，個別作業，料理状態，食品・道具提示，代替可，留意事項，その他に分類されるが，このうち，作業宣言，食品・道具提示，代替可，留意事項，その他については発話の文末表現のパターンを記述することで認識できるとしている．

レシピ文のアスペクトを分析し，平行動作や前後関係を明確にした上でタイムマップを表示する研究も行われている [14] [15]．この研究では Karlin による４つのアスペクトに分類できないレシピ動作完了を示す語などを，東条 [16] による分類である「完成相」,「達成相」,「進行相」,「完了相」の４つのアスペクトクラスに分類することにより解決している．そして，レシピ文を Web ブラウザ上でタイムマップ表示させることに成功している．

2.2　テキスト文を解析する研究

日本語テキストを題材とした研究としては，構文解析，格フレーム生成系，意味構造生成系から構成されるシステムを使って，日本語の物語文に対応する意味構造を自動的に生

成する研究が行われている [18]．本研究とテキストを図示するという点で共通点のある，小説という単一文ではない文章を「想念」「発話」「それ以外」のスペースに分けることによって構造化し，図示する研究も行われている [17]．テキスト文を解析する際に重要な技術要素となる日本語文節間係り受けに関する研究も行われており [21]，本研究でも日本語係り受け解析器である CaboCha[20] を使用している．

2.3 ソフトウェア開発支援系の研究

ソースプログラムと XML で記述されているドキュメントに含まれる識別子を自動的に対応付ける手法を提案した研究が行われている [9]．これは後藤らによるものだが，ソフトウェアの理解支援のためにソースプログラムとドキュメントの相互参照を提供するツールと, 両者の整合性を検査するツールを実装し, その有用性を確認されている．また，注連らにより日本語機能表現に対し機械学習を用いて検出する手法を提案した研究も行われている [10].

中鉢らにより，ユースケースモデリング手法，SBVA (Scenario-Based Visual Analysis) 法の提案が行われている．これは自然言語で記述されたシナリオの名詞句，動詞句に注目し，各句の意味的関連を業務鳥瞰図と称するグラフ構造に表現し，このグラフ構造を閲覧・編集することで，シナリオの不備の訂正や調整を行い，アクタとユースケースの抽出を図る研究である [22]．同氏らはこの研究に先立ち，ソフトウェアの要求定義文書に基づいて記述したシナリオに対し,「図解化エディタ」を用いて分析作業を行うことでオブジェクト指向分析 (OOA) で使用可能なモデル要素を抽出するための方法論を提案しており，これを支援するソフトウェア (SBVA エディタ) を開発している [29]．また，日本語による要求記述から, システムを特徴付ける機能およびデータを提供する部分の Java プログラムを生成することにより, プロトタイプ作成を文援するツールを開発した [23].

プログラムから UML への変換に関することは一般的にはリバースエンジニアリングと呼ばれているが，関連研究は数多く行われており，中でも Java を対象としているものが目立つ．JAVA プログラムから UML に変換する研究には大きく 2 つの傾向に分かれており，一つは静的構造を表すものであり，もう一つは動的構造を表すものとなる．静的構造を表すものとして代表的なものはクラス図であり，動的構造を表すものの代表的なものとして，シーケンス図が挙げられる．

2.3.1 UML クラス図を生成する研究例

ソースプログラムを元にしたプログラムの可視化に関する研究として，堀田らによるオブジェクトフローグラフを使った研究がある．これは Java プログラムを一旦オブジェクトフローグラフという形にしてから UML を生成しているもので，クラス図生成の紹介が行われている [25]．リバース機能によりクラス図を生成できるモデリングツールがいくつか存在するが，大抵は Java をサポートしている．

2.3.2 UML シーケンス図を生成する研究例

シーケンス図を生成することは理論的に困難である．オブジェクト指向プログラムでは，実行時に動的に生成されるオブジェクトが相互にメッセージを交換することによってシステムが動作するが，どのオブジェクトがメッセージ通信を行うかは，実行時に動的に決定される．そのため，設計や実装作業においても常に，システムが生成するオブジェクトの動的な振る舞いをイメージしながら実装することになる．

市販されているモデリングツールでもシーケンス図のリバース機能をサポートしているものはごく僅かである．研究報告事例も数少ないが，特徴的なものとして２つ紹介しておく．

一つ目は，谷口らにより提案 [24] されているもので，プログラムが実行時に生成するオブジェクト群の動作理解を支援するために，オブジェクト指向言語の１つである Java 言語のプログラムからシーケンス図を作成する手法である．これはプログラムの実行履歴を元に図の生成を行うというもので，シーケンス図を作るために必要なメソッド呼び出しに関する情報を利用している．これは，設計段階で作成されたシーケンス図と，実装されたプログラムの振る舞いとの違いを調べることや，設計ドキュメントが提供されていないプログラムのシーケンス図を作成する必要がある場合に有効なアプローチである．

二つ目は，関数呼び出しの実行パスを網羅して表現することを重視し，静的解析でベースとなるシーケンス図を作成し，そのシーケンス図上に動的解析から得られるパスを重ねあわせるというアプローチを行っている [26]．

2.4 本研究の特色

以上，過去のさまざまなテキスト解析結果ではそれぞれ独自の結果表示をしている．そして現在までのところ料理レシピ文を UML で表現するといった研究は見受けられていない．本研究では世界的に標準となっている UML 形式で料理レシピを図示することを目指しており，新しいアプローチであると考えている．

第3章　ソフトウェア工学の方法論に基づく解析

3.1　ソフトウェア工学とは

ソフトウェア工学は，コンピュータのソフトウェアの開発方法を研究対象とする情報工学の一分野である．通常、開発対象となるソフトウェアの開発をいくつかの工程に分けて考察する．この工程をソフトウェア開発ライフサイクルという．

一般的にソフトウェア工学で用いられるソフトウェア開発ライフサイクルは次のようになっている．

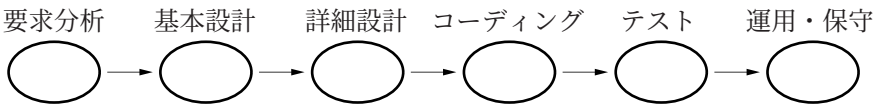

図 3.1: ソフトウェア開発ライフサイクル

- 要求分析 ：着想したソフトウェアがどのような機能を持つべきかを検討し必要に応じて文書化する．
- 設計 ：機能がソフトウェアとしてどのように実装されるべきかを検討し必要に応じて仕様化する．一般的には，基本設計と詳細設計とに分ける場合が多い．
- コーディング：仕様に従ってプログラムを作成する．
- テスト ：作成されたプログラムが機能的な要求を満たしていることを実証する．
- 運用・保守 ：ソフトウェアを使用したり、新たな要求に応じて機能を追加・変更したりする．

本研究のターゲットは要求分析および基本設計のための支援ツールの位置づけを目指している．

ソフトウェア開発方法論は，構造化技法 (構造化プログラミング) と，オブジェクト指向開発方法論 (オブジェクト指向プログラミング) とに分類される．ここでは，本論文では１章に述べた理由から，オブジェクト指向開発方法論に焦点を充てている．

3.2 オブジェクト指向開発方法論

オブジェクト指向分析とオブジェクト指向設計を含めた，オブジェクト指向開発の具体的な方法論を，オブジェクト指向開発方法論 (object-oriented methodology) という．これまで非常に多くのオブジェクト指向開発方法論が考案されている．2007 年現在では、オブジェクト指向分析設計におけるモデル図の記法は，統一モデリング言語 (UML) が使われる事例がほとんどである．

UML ではクラスという概念が重要であり，クラスの抽出に失敗するとソフトウェア開発に悪影響を及ぼす．ここでクラスを抽出するいくつかの方法を紹介する．

表 3.1: クラス抽出の方法

抽出方法	特徴	難易度
名詞句抽出法	ユースケースから名詞句抽出．中心的な概念を表すものを抽出．抽出したものをグループ化	低
責務抽出法	分類した視点から抽出．視点をかえることで名詞句抽出法の欠点を補うことができる	普通
図解抽出法	図や表を活用．そこに現れる用語や概念をクラスとする方法	やや高
直感抽出法	過去の経験ひやめきによる方法ドメイン知識と経験による直感要求分析による本質のひらめき	高

名詞句抽出法では，要求モデルから名詞句を抜き出し，それらを構造化していくやり方をすることが一般的で，ブレーンストーミングを使うことがある．複数人で名詞句をすべて抜き出す作業をするが，これを形態素解析により自動化するというものが本研究のアイデアの一つである．係り受け構造を利用してグループ化していくことも考えられるが，本研究ではそこまでには至っていない．

クラスの振る舞いに関して，コミュニケーション図を用いることがある．それはクラスを空間的に配置することで，クラス同士の役割分担を把握しやすいからである. しかし実際には時間軸に沿ってソフトウェアが実行されることになるので，シーケンス図を使って特定のシナリオを確認することが有効である．

3.3 UMLによる図式化の適用

UMLとは数ある方法論の中で使用されていたモデリング言語を統一し標準化したものである．UMLはオブジェクト指向の分析設計方法論にもとづいたモデル化技術であり，ソフトウェア開発のための設計書や仕様書の規格としてのデファクト・スタンダードとなりつつある．UMLによるモデル化は，システムの構造や振る舞いを明確化する過程で，仕様の曖昧さをある程度排除できるため利用価値は高く，分析・設計・実装などの開発フェーズのあらゆる局面で利用されている．本研究ではこのUMLの定義のうち，クラス図とシーケンス図の導出を行った．

表 3.2: UML 各図の名称 と 本研究対象

	UML 各図の名称	本研究対象
構造図	クラス図	○
	パッケージ図	
	オブジェクト図	
	コンポーネント図	
	配置図	
振る舞い図	ユースケース図	
	コラボレーション図	
	シーケンス図	○
	ステートチャート図	
	アクティビティ図	

次ページ以降に「クラス図」「クラス間の関係」「シーケンス図」それぞれについて，UMLの定義をどのように使用しているかを説明する．

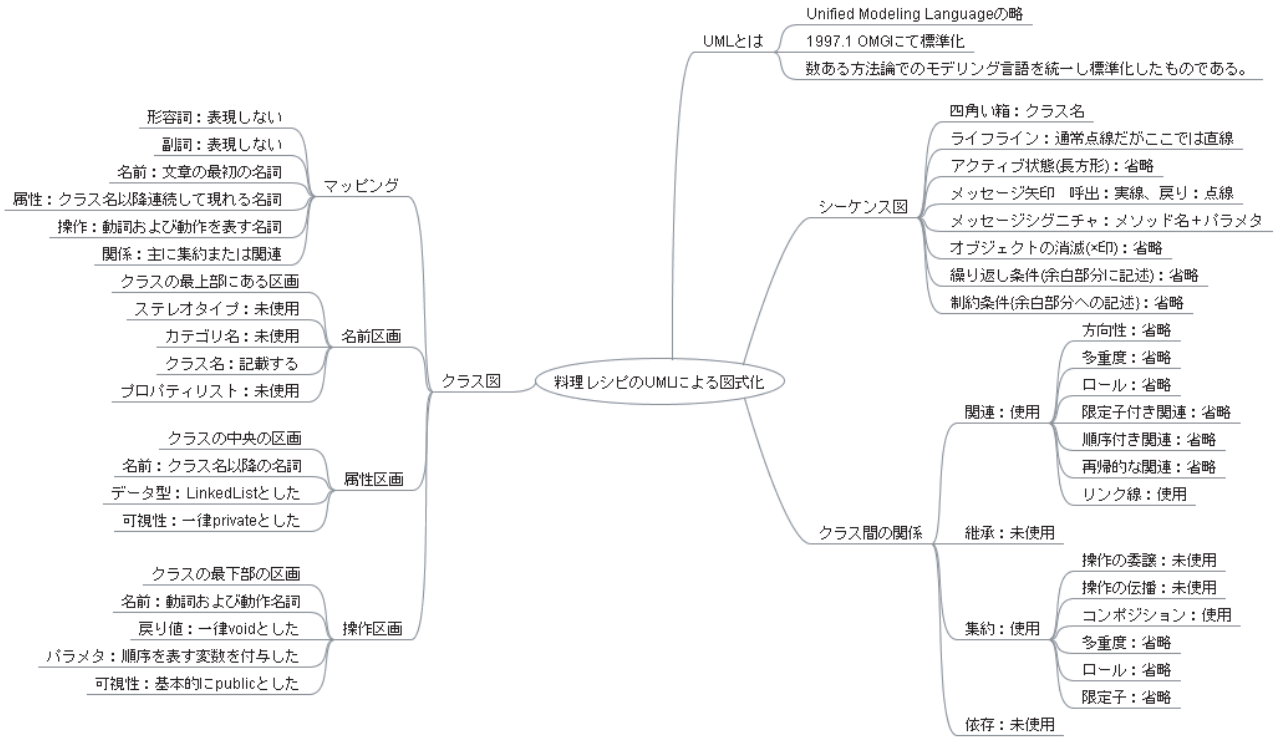

図 3.2: UML による図式化の概念全体像

3.3.1 クラス図

ここではクラス図について,「マッピング」「名称区画」「属性区画」「操作区画」それぞれについてどのような対応をしているか説明する.

マッピング

形容詞	：表現しない
副詞	：表現しない
助詞	：表現しない
名前	：文章の最初の名詞
属性	：クラス名以降連続して現れる名詞
操作	：動詞および動作を表す名詞
関係	：主に集約または関連

本研究では名詞句抽出法を採用しているため，クラス名には名詞を使っている．ただし，連続してあらわれる名詞がある場合，最初に現れた名詞との繋がりがあると考え，属性として抽出している．

名前区画

クラスの最上部にある区画

ステレオタイプ	：未使用
カテゴリ名	：未使用
クラス名	：記載する
プロパティリスト	：未使用

名前区画では，クラス名のみ使用しており，その他は未使用としている．

属性区画

クラスの中央の区画

名前	：クラス名以降の名詞
データ型	：`LinkedList` とした
可視性	：一律 `private` とした

属性区画にはクラスとして抽出された名詞以降に現れた名詞をリストアップし，データ型は LinkedList とし，外部から参照されることはないことから，可視性は Private とした．

操作区画

クラスの最下部の区画
名前　　：動詞および動作名詞
戻り値　　：一律 void とした
パラメタ　　：順序を表す変数を付与した
可視性　　：基本的に public とした

操作区画としては，動詞および動作を表す名詞（ここでは動作名詞と呼ぶ）を抽出した．戻り値は使用しないので void 型とし，外部から呼ばれるメソッドとなるため可視性は Public とした．さらにパラメタとして，出現順位を保存するために，通し番号付の変数を付与した．これは後にシーケンス図にする時に使用する番号でもある．

3.3.2 クラス間の関係

ここではクラス間の関係についてどのような対応をしているか説明する.

クラス間の関係

関連：使用
- 方向性：省略
- 多重度：省略
- ロール：省略
- 限定子付き関連：省略
- 順序付き関連：省略
- 再帰的な関連：省略
- リンク線：使用

継承：未使用

集約：使用
- 操作の委譲：未使用
- 操作の伝播：未使用
- コンポジション：使用
- 多重度：省略
- ロール：省略
- 限定子：省略

依存：未使用

クラス間の関係は，関連としてはリンク線以外使用しない．ただし料理として成立するための食材を含む主要なクラスはコンポジションの関係があると考える．コンポジションとは，集約しているクラスのオブジェクトが削除されると，必ず集約されている側のオブジェクトはすべて削除される関係があり，UML では、視覚的には黒い菱形を集約する側クラスの線の端に記述する事で表現する．

定義としては以上だが，機能的制約によりリバースエンジニアリングによるコンポジションの再現ができないため，今回作成したアプリケーションでクラス図を表示する際には，該当箇所を「継承」で表示するようにした．

3.3.3 シーケンス図

ここではシーケンス図についてどのような対応をしているか説明する．

シーケンス図

四角い箱	：クラス名
ライフライン	：通常点線だがここでは直線
アクティブ状態（長方形）	：省略
メッセージ矢印	：呼出=実線，戻り=点線
メッセージシグニチャ	：メソッド名（＋パラメタ）
オブジェクトの消滅（×印）	：省略
繰り返し条件（余白部分に記述）	：省略
制約条件{余白部分への記述}	：省略

四角い箱にはクラス名を表示している．オブジェクト名を使うことが一般的であるが，本研究ではオブジェクト名とクラス名は同等のものをして扱っている．ライフラインは通常点線で表示するが，ここではテキスト表示ということもあり，アクティブ状態も含めて表現することが困難であるためである．メッセージの呼び出しは実線矢印，戻りは点線の矢印で表現した．メッセージシグニチャにはメソッド名のみ表示しているが，内部的にはパラメタの変数情報を使っている．

3.4　本研究における対応付け

テキスト文書の言語構造解析に関する過去研究では，各々の視点により解析結果を表現していた．今回の研究ではオブジェクト指向の世界では標準規格となっている UML 表記を用い，テキスト文書の構造解析結果を表現した．オブジェクト指向の題材として，プログラム仕様に似た表現をしている料理のレシピ文を使用した．手順としては以下のようになる．

1. あらかじめ料理に関する単語集を作成しておく．
2. まず名詞句に注目し，単語集に存在するものはクラスとしてとらえる．
3. 次に動詞句に注目し，料理動作を表すものはメソッドとしてとらえる．
4. 上記の情報を，プログラム言語で出力する．（本研究ではＪａｖａ言語を使用）
5. 上記で出力したプログラムファイルをＵＭＬツールであるＪＵＤＥで取り込み，クラス図に変換する．これによりレシピ文をクラス図とし表記することが可能となる．
6. 上記と平行してシーケンス図を作成する．クラスをオブジェクトの箱に記載し，メソッド呼び出しを矢印で表現する．

以上の結果，レシピ文の全体像をクラス図として表記，料理の手順をシーケンス図として表記することになるが，ここで使ったクラス図やシーケンス図は UML1.5 に準拠した表記となっており，世界共通表記として通用するものとなっている．

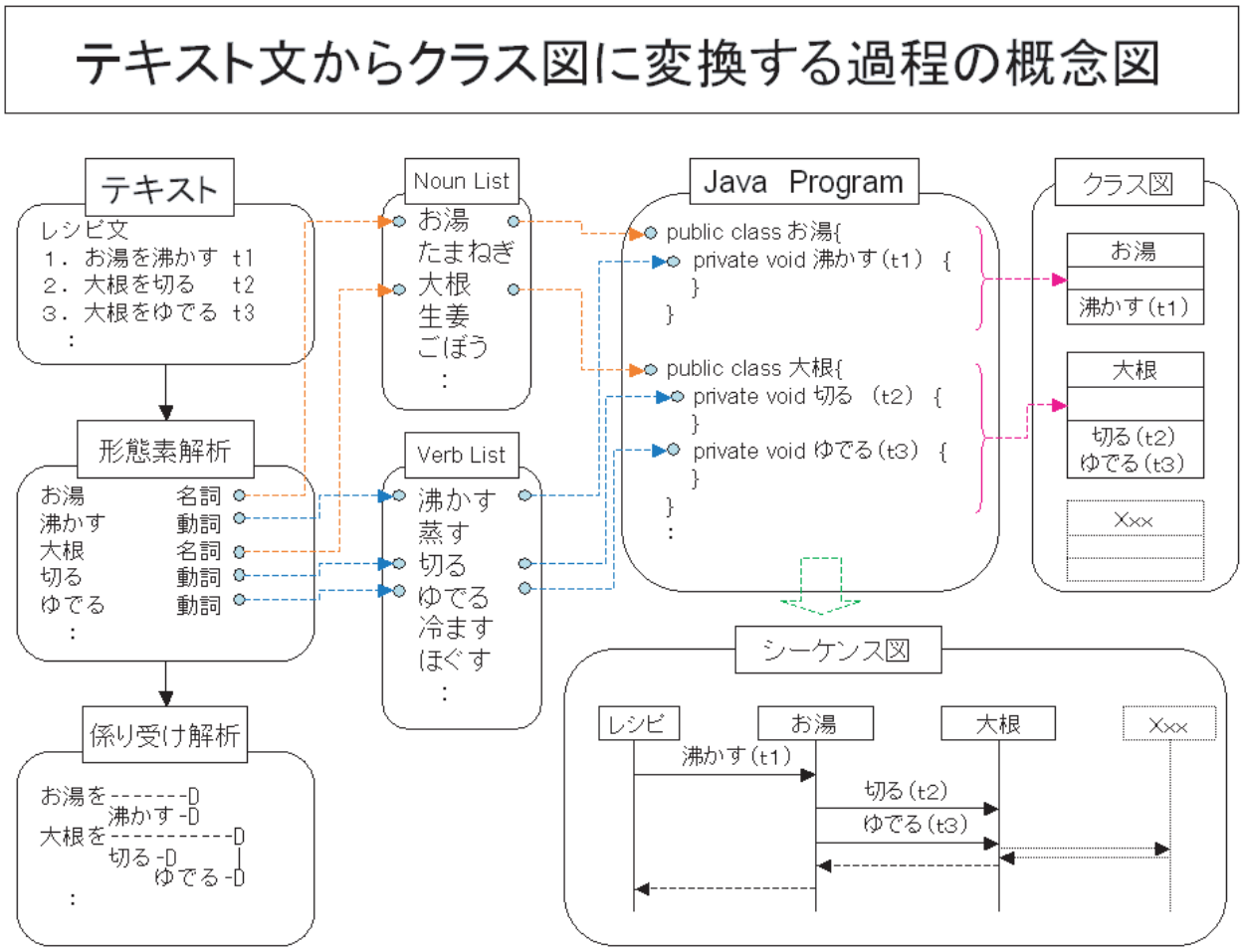

図 3.3: テキスト処理概念図

第4章　システムの設計・仕様

4.1　ソフトウェア概要

4.1.1　使用ソフトウェア

はじめに，アプリケーション作成に使用したソフトウェアを下記に列挙する．

- 形態素解析エンジン： cabocha Ver.0.53
- アプリケーション作成ツール：Microsoft VisualStudio2005 Academic Edition
- アプリケーション作成言語：C#
- UML モデリング ツール ：JUDE Community Ver.5.1.1

4.1.2　システム構成

ここに作成したアプリケーションのシステム構成図を示す.

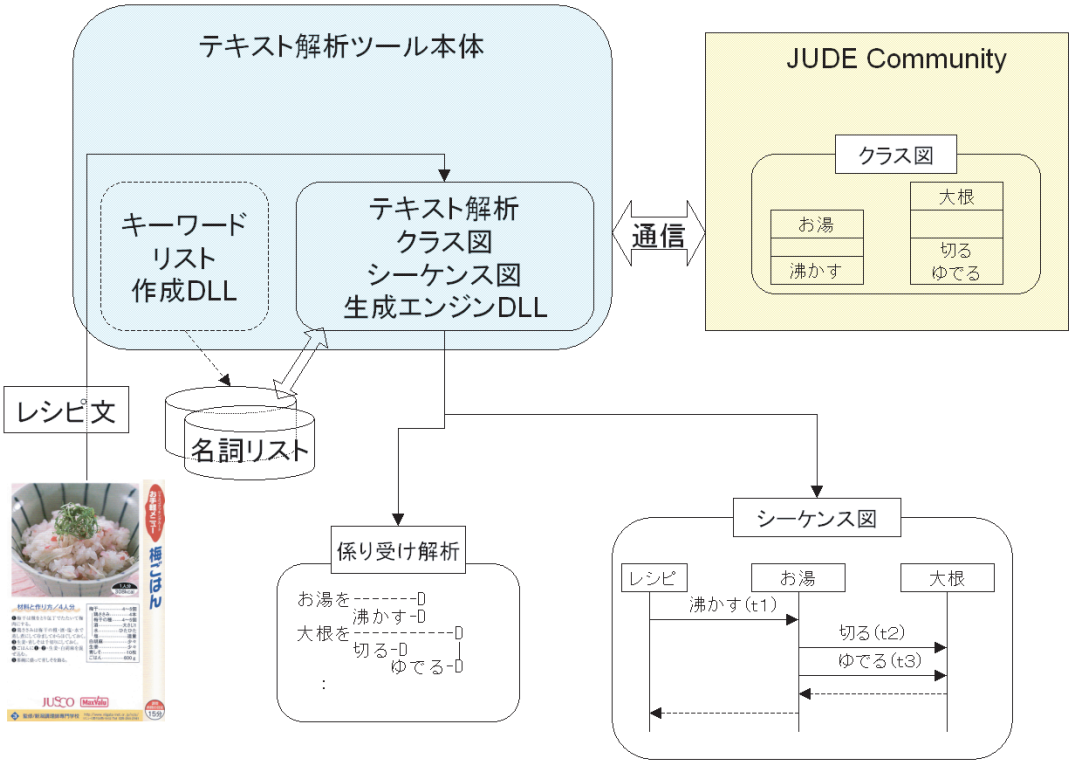

図 4.1: システム構成図

テキスト解析ツール本体

アプリケーション本体は特に機能を持たず，プラグイン構成で機能追加できる構成とした．本来の目的であるテキスト文の形態素解析/係り受け構造解析（内部的には CaboCha を使用）を行って，クラス図とシーケンス図を生成するエンジン部を DLL 形式で作成し，アプリケーション本体から呼び出して動作させている.（後述する照合用リスト類をデータベース化する処理を現在作成中であるが，未完成である.）

JUDE Community

今回クラス図を作成するのに JUDE Community というフリーの UML 作図ツールを使用した．選定理由は以下である．

- フリーである
- 日本語が扱える
- リバース機能がある

特に 3 番目のリバース機能が採用の決め手となった．リバースするためには入力として Java ソースコードが必要なため，解析結果をオブジェクト指向言語である Java 言語に変換するようにしている．JUDE で Java ソースコードを取り込んだあとは，マウス操作でクラス図を自動的に作成することができる．

係り受け解析

CaboCha により生成された係り受け解析結果を表示する機能を持っている．

シーケンス図

シーケンス図はテキストベースの罫線などを組み合わせて表示するようにした．

4.2　作成した照合用リスト類

本来「内部辞書」と呼びたいところだが，実際のところ単語の羅列リストでしかないものをいくつか作成した．

食材リスト

主に食材を抽出するための照合用リストである．形態素解析の結果「名詞」と判断された単語とこのリストを照合させ，一致したものは画面上のクラス候補リストに追加されるようになっている．このリストに登録されたものは，後にクラス図を表示した際にはクラス名として現れることになる．ただし，連続して名詞が抽出された場合には，最初の名詞つまりクラスの属性としても定義するようにしている．

食材リストの抜粋

梅干 / 手羽 / 生姜 / しそ / 胡麻 / 茶碗 / おくら / かぼちゃ / ...

調理動作 (動詞) リスト

主に調理動作を抽出するための照合用リストである．形態素解析の結果「動詞」と判断された単語とこのリストを照合させ，一致したものは画面上のメソッド候補リストに追加されるようになっている．このリストに登録されたものはクラスメソッドとなり，現れた順番がわかるようにインクリメンタルな番号を含んだ引数をもたせるようにしている．後にクラス図を表示した際にはクラスメソッドとして現れることとなる．

動詞リストの抜粋

蒸す / 煮る / 冷ます / ほぐす / 混ぜる / 盛る / 飾る / たたく / ...

調理動作 (名詞) リスト

名詞の中に調理動作を示す句が含まれていることがある．例えば「千切り」などがあるが，この場合はこの動作の属するクラスのメソッドと認識させたい．このためこのリストに含まれるものがあればクラスメソッドにもなるようにした．

動作名詞リストの抜粋

リスト登録名称	補足説明文
短冊切り	短冊のように縦長の長方形に切っていく方法です.
千切り	短冊切りした材料をさらに細く切っていく切り方です.
みじん切り	一般的には細かくせん切りにした材料を，さらに端から細かく切ります.
そぎ切り	材料に対して，包丁を斜めに寝かして入れ，そぐように切る切り方です.
薄切り	材料を端から薄く切っていく方法で，”スライスする ”と表現されている場合もあります.
輪切り	大根や，にんじんのように，切り口が丸い材料を端から円形に切っていく切り方です.
半月切り	輪切りでは大きすぎる場合に，輪切りのさらに半分の大きさにする切り方です.
いちょう切り	半月切りを更に半分にカットしたものがいちょう切りです.
...⋮	⋮

機材リスト

一般的にレシピには料理手順と用意する食材の記述があるが，用意すべき機材についてはレシピ文中に書かれていることが多い．そのためレシピ文を解析した際に，必要となる機材を抽出しておくと便利だろうと考えた．このような使い方をするためのリストであるが，中身は機材となる名詞が書かれているものである．

機材リストの抜粋

包丁 / ボール / 浅鍋 / 計量スプーン / 計量カップ / 小さじ / 大さじ / ...

4.3 変換ルール

レシピ文とクラス/メソッドの対応付けを以下のような変換ルールにより行った.

- レシピ文を形態素解析した結果，名詞，動詞，形容詞，副詞などに区分けされることになるが，ここでは名詞と動詞に注目している.
- 名詞でありかつ材料として登録されているものを，クラスと定義した.
- ただし，単に名詞が連続しているだけの場合もあり，その場合は 2 番目以降に現れた名詞は最初に現れた名詞のクラス属性とも定義した.
- 動詞であり，動詞リストに登録されているものを，直前のクラスに分類された名詞に対する操作（メソッド）と定義した.
- 名詞であるが，動作を表すものは動作名詞と区分し，メソッドと定義した.
- クラス図では，料理のタイトルをスパークラスとしており，料理素材はサブクラスとして表現されるように汎化の関係にしている.
- 名詞が連続した場合は，前後を結合してひとつの意味あるものになる場合がある. その場合は連結した状態でひとつのクラスと定義することが望ましい場合があると考えるが，開発したシステムでは対応できていないので，今後の課題である.

  この他にも下記のような制約もある.

- プログラム上に多重度を表現できないため，クラス図では多重度を省略している. 正確には再現不可能である.
- シーケンス図の矢印はプログラム実装上の都合で 7 階層までしか対応していない. したがって多くの素材を使う料理ではシーケンス表示が不十分な場合がある.

ここで，CaboCha を使ったレシピ文の解析例を紹介する.

梅ごはんの作り方

1. 梅干は種をとり包丁でたたいて梅肉にする.
2. 鶏ささみは梅干の種・酒・塩・水で蒸し煮にして冷ましてからほぐしておく.
3. 生姜・青しそは千切りにしておく.
4. ごはんに 1,2, 生姜, 白胡麻を混ぜ込む.
5. 茶碗に盛って青じそを飾る

これを次のコマンドにより解析した結果を表にした.

使用した CaboCha オプション

```
"c:\Program Files\CaboCha\bin\cabocha.exe" -f1 -n1 -o 京大コーパス.txt
```

このコマンドにより出力した結果を次の表に抜粋を記載する.（全文は付録に添付）

表 4.1: 生成された京大コーパス.txt の中身（抜粋）

分割文字	かな	文字	品詞	活用	接続
梅干	ウメボシ	梅干	名詞-一般		
は	ハ	は	助詞-係助詞		
種	タネ	種	名詞-一般		
を	ヲ	を	助詞-格助詞-一般		
とり	トリ	とる	動詞-自立	五段・ラ行	連用形
包丁	ホウチョウ	包丁	名詞-一般		
で	デ	で	助詞-格助詞-一般		
たたい	タタイ	たたく	動詞-自立	五段・カ行イ音便	連用タ接続
て	テ	て	助詞-接続助詞		
梅	ウメ	梅	名詞-一般		
肉	ニク	肉	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
する	スル	する	動詞-自立	サ変・スル	基本形
.....					
生姜	ショウガ	生姜	名詞-一般		
は	ハ	は	助詞-係助詞		
千切り	センギリ	千切り	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
.....					

このCaboChaオプションを使ったのは，コンピュータで扱いやすい形式であるためであるが，文字が細分化されるためプログラム側でひと工夫必要である．

ここで，動作名詞リストにある補足説明をクラス図に反映させるための工夫を紹介する．次に示すのは試作アプリケーションで生成したJavaソースプログラムの一部である．

千切りに関する情報のコメント化

```
package Test.Class;

public class 生姜 extends 梅ごはん {
     private String 青じそ;
     /**
      *短冊切りした材料をさらに
      *細く切っていく切り方です.
      */
     public void 千切り(Number t6) {
     }
}
```

このようにメソッドコメントとして補足説明文を残すことにより，コメント文をインポートさせることでクラス図のメソッド定義に反映させることができる．以下はJUDEでのメソッド定義の表示例である．

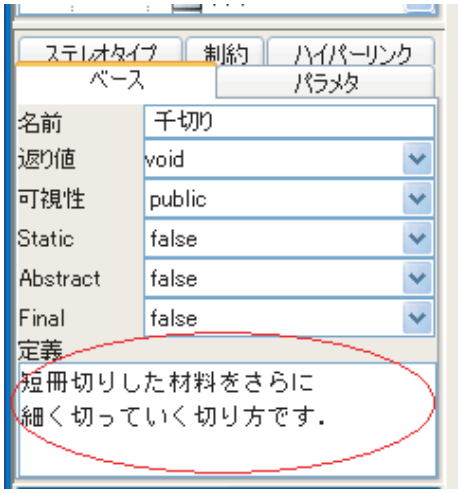

図 4.2: JUDEでのメソッド定義の表示例

実際にはアプリケーション側からJUDEにキーイベントを送って，Javaソースプログラムを取り込ませ，JUDE上のプロジェクトツリー上にクラス情報などがインポートされ

る仕組みとなっている．試作したアプリケーションでは完全自動化までは実装できておらず，クラス図の自動生成に関してマウス操作が必要になる．（右クリックしてメニューを選ぶだけ）JUDE によるクラス図自動生成の結果イメージを次に紹介する．

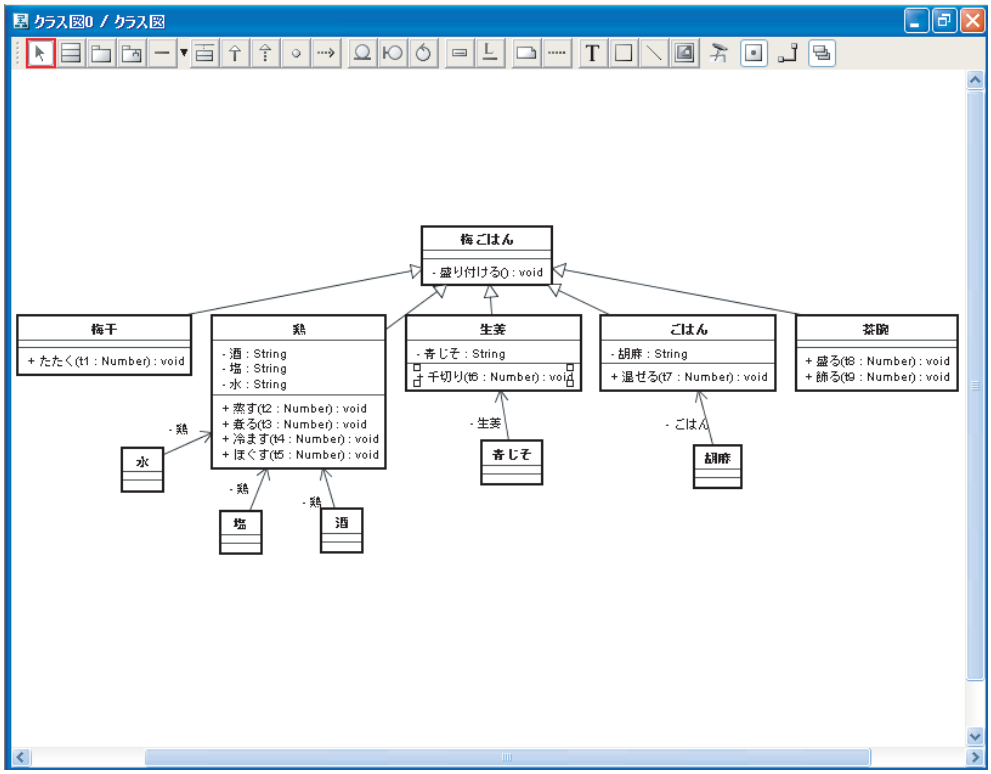

図 4.3: クラス図の例

クラス間の関係は，関連としてはリンク線を使用．料理として成立するための食材を含む主要なクラスはコンポジションの関係で表したかったが，機能的制約により継承関係で表示している．

第5章　実行例

5.1　手順

5.1.1　料理レシピの準備

まず対象テキストとして，手短にあったジャスコクッキングカード「お手軽メニュー」シリーズを選んだ．このテキストを，CaboCha[20] で係り受け構造解析し名詞や動詞を抽出したのち，図式化した．

図 5.1: ジャスコクッキングカード「梅ごはん」/「豚肉の香り揚げ」

5.1.2　料理レシピの入力

今回作成したアプリケーションの Input 部にレシピ文を入力したところ．毎度手入力するのは手間がかかるので，よく使うレシピ文はファイルにしておくと便利である．また Web 上に公開されているレシピ文はコピー＆ペーストして使うことも可能となっている．なおタイトル部には，レシピのタイトルを入力するが，ファイルから入力した場合はそのファイル名（拡張子はカット）を表示するようになっている．

このステップは必須項目である．

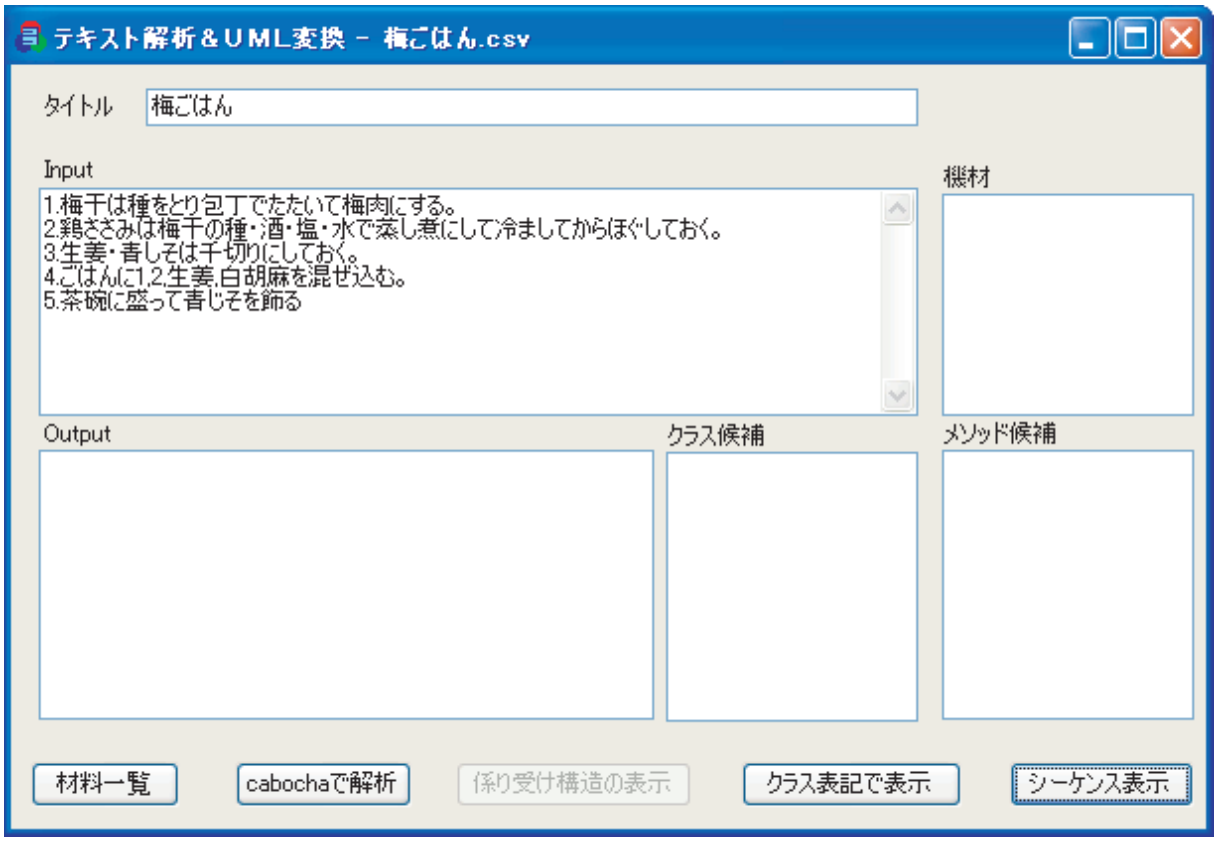

図 5.2: 作成したアプリケーションにレシピ文を入力したところ

この状態から「cabocha で解析」ボタンを押下すると「係り受け構造の表示」ボタンが有効になる仕組みになっている．

5.1.3　レシピ文の形態素解析

アプリケーションの「cabocha で解析」ボタンにより，Input 部のレシピ文を形態素解析および係り受け構造の出力を行う．Output 部には形態素解析結果の内，名詞，動詞，助詞，副詞のみ表示するようになっている．名詞の中で内部名詞リストと一致したものは「食材」の欄に，動詞の中で内部動詞リストと一致したものは「行為」の欄にそれぞれ表示するようになっている．さらに名詞の中で機材リストに一致したものは「機材」の欄に表示されるが，ここには動詞の中で機材が連想されるもの（画面の例では，例えば「たたく (動詞)」により「まな板」を導出している）を表示させる機能も追加してみたが，まだ実用レベルにまで至っていない．

このステップは必須項目である．

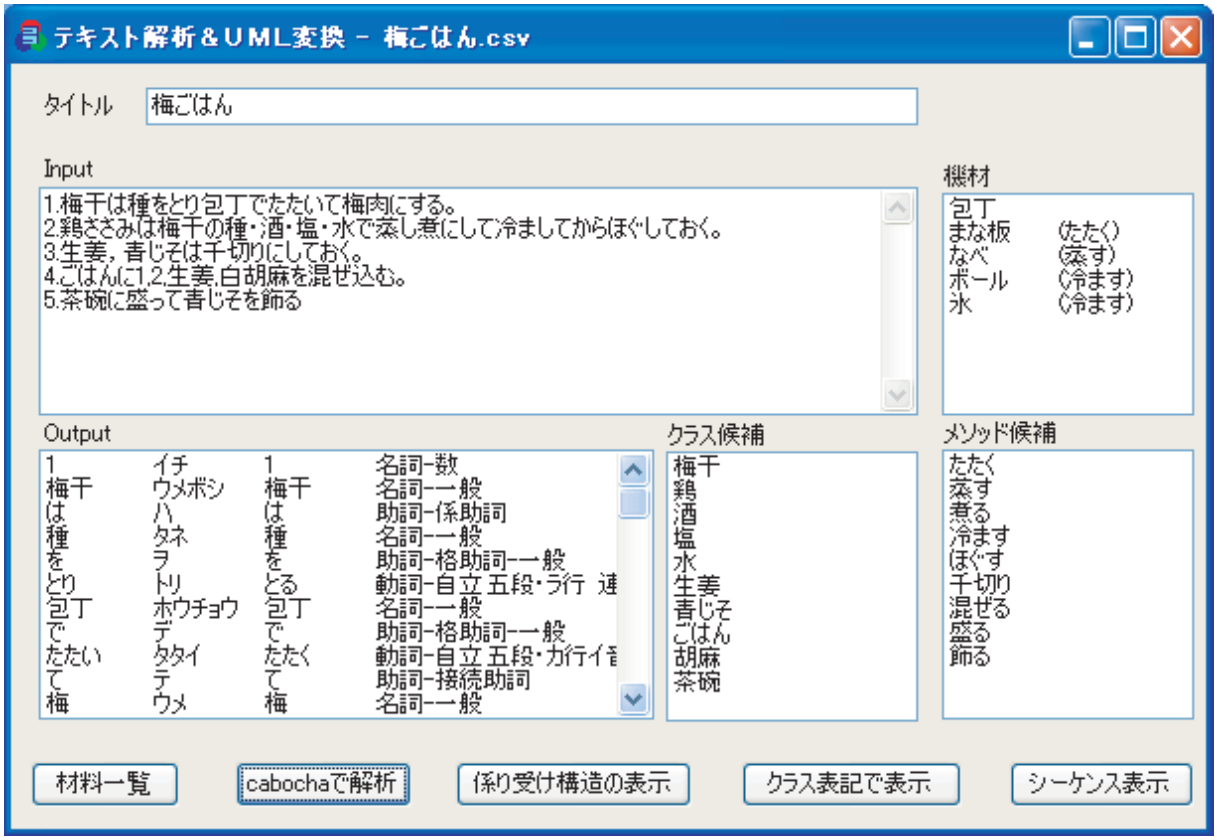

図 5.3: CaboCha で形態素解析した結果

次に行うことができるのは「係り受け構造の表示」による確認と「クラス表記で表示」によるクラス図の確認,「シーケンス表示」によるシーケンス図の表示があるが，ここでは順を追って説明する．

5.1.4　レシピ文の係り受け構造の表示

アプリケーションの「係り受け構造の表示」ボタンにより，係り受け構造の表示を行う．

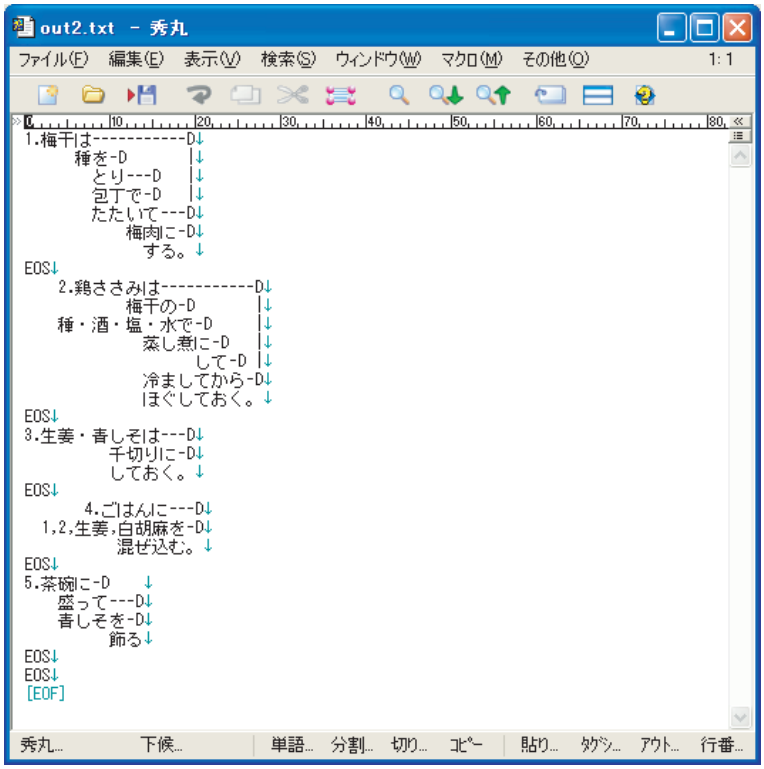

図 5.4: CaboCha による係り受け構造を表示したもの

これは cabocha により出力した係り受け解析結果のファイルを，テキストエディタで表示しているだけである．このステップは省略することも可能である．

次ページではクラス図の表示に関する説明を行っている．

5.1.5 レシピ文のクラス図への変換

アプリケーションの「クラス表記で表示」ボタンにより，UML 設計ツールである JUDE でクラス図を表示します．内部的には

(1) Java ソースコードを生成する

(2) JUDE の [ツールメニュー]-[Java ソースコードの読み込み] を選択する

(3) (1) で生成したソースコードを選択して [適用] ボタンを押す

(4) ソースコードのインポート処理が開始される

(5) インポートが修了するとプロジェクトツリーの構造ツリーにパッケージが追加される

(6) パッケージを選択し右クリックメニュー [詳細クラス図を自動生成する] を選択する

(7) ソースコードをリバースエンジニアリングした形でクラス図が表示される

という処理を行っている．なお JUDE の操作はアプリケーション側から仮想キーコマンドを送信することにより実現している．

このステップはクラス図を確認するためには必須であるが，シーケンス図だけを確認したい場合はスキップすることも可能である．

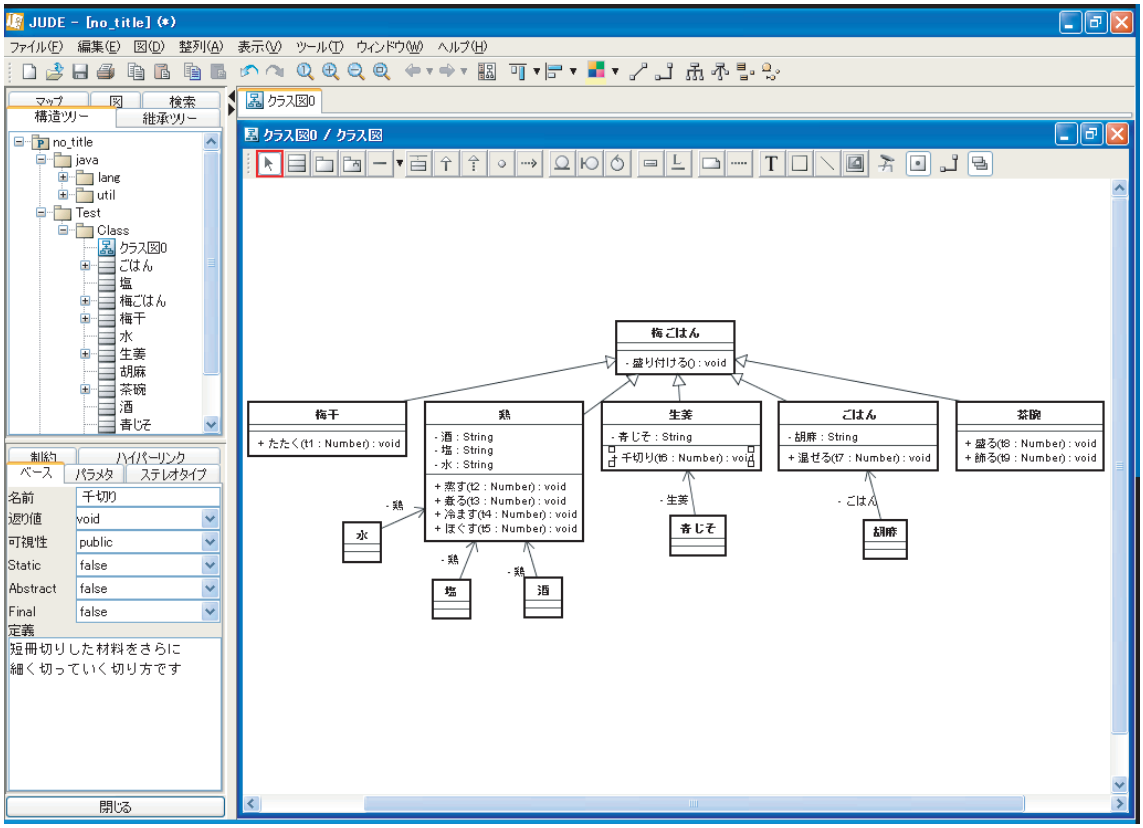

図 5.5: JUDE と連携してクラス図を自動生成した結果

5.1.6　レシピ文のシーケンス図への変換

アプリケーションの「シーケンス表示」ボタンにより，シーケンス図を生成し表示する．このステップはシーケンス図を確認するためには必須．

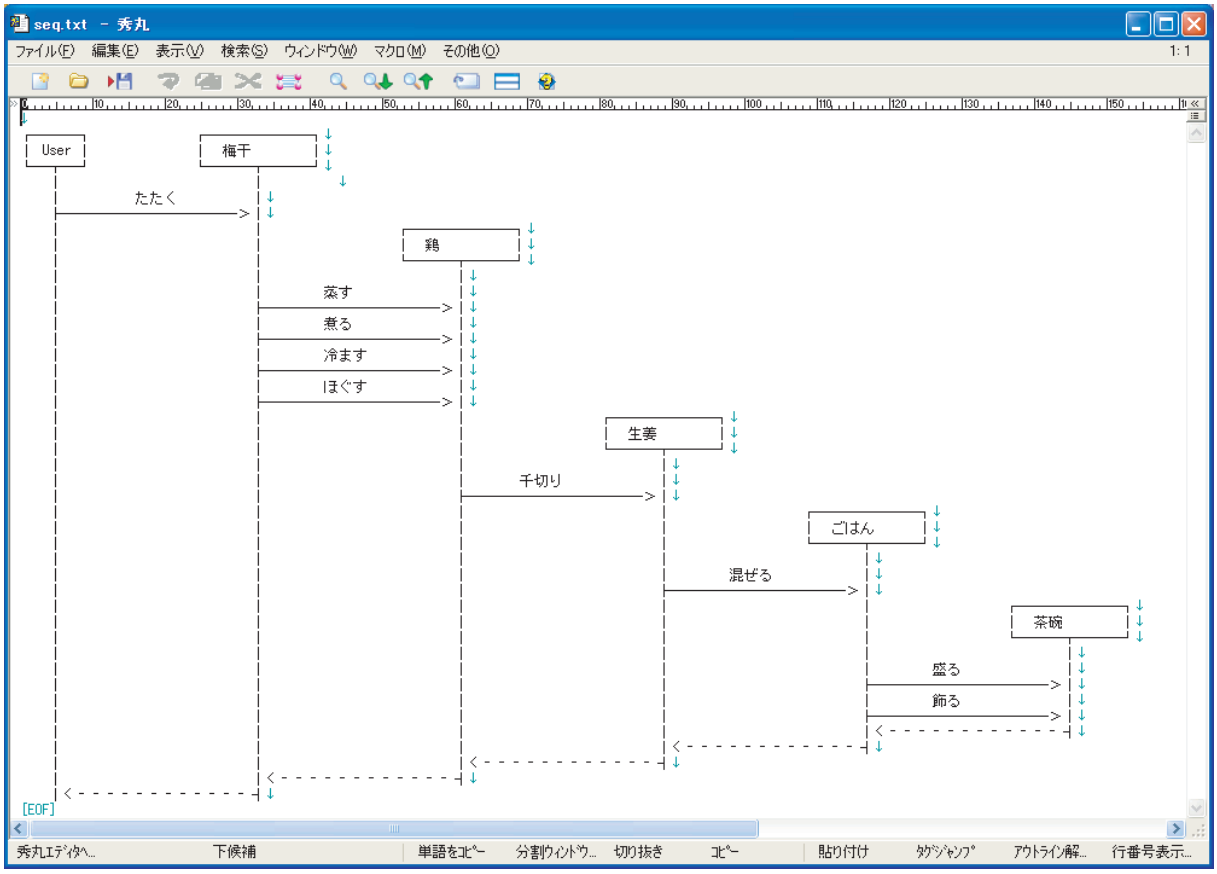

図 5.6: 生成したシーケンス図を表示した結果

本来であれば，グラフィック機能を使い，高度な表示をさせたかったが，知識と経験と時間不足のためテキスト表示になっている．（テキストならではの手軽さや使い勝手もあると考える．）

5.2 結果

以上，実行結果と検討をまとめると，以下のようになる．

- ある分野に特化したレシピ文には有効である．(現状，和食のみ)
- 係り受け解析に多少時間がかかることがある．
- クラス図はモデリングツールで作られるので便利である．
- シーケンス図は箱の位置が斜めに配置されるが特に視認性は問題ない．
- 文字をみるより図式化した方が分かりやすい．

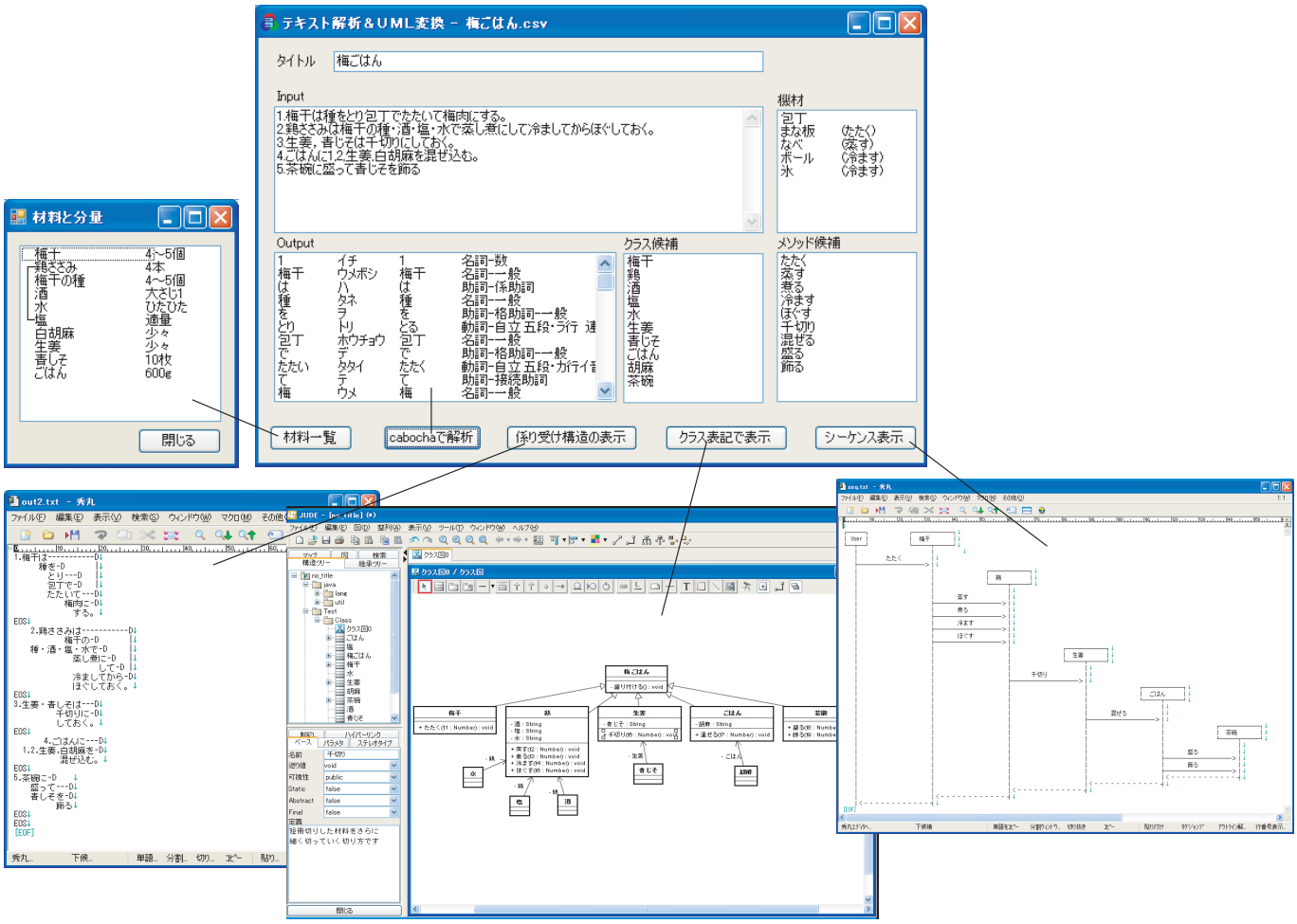

図 5.7: 料理レシピの UML 変換の全体像

5.3 考察

レシピ文解析に関する過去の類似研究では，タイムマップで表現したものがあるが，それに比べてより一般的なUMLでの表現をすることに成功したといえる．ただし今のところ和食レシピでしか試していない．現状では和食限定のアプリケーションであり，適用ジャンルを広げるには辞書類を多くのジャンルに対応できるようにさらなる充実が必要になる．さらに入力ソースは特に限定する必要がないので，次のようなことが言えると考える．

- 辞書を増やせば対応分野を広げることができそうである．(現状，和食のみ)
- 素材辞書はもう少し充実させる必要がありそうだ．
- 店頭で配布されているレシピ紙でも十分に処理可能である．
- 料理本に書いてあるレシピ文でも十分に処理可能である．
- Web上の料理レシピサイトにあるレシピ文でも十分に処理可能である．

改善点としては次のことが挙げられる
シーケンス図の矢印は7階層までしか対応していない．したがって多くの素材を使う料理ではシーケンス表示が不十分な場合がある．またシーケンス表示では内部的にユニコード(UTF8)を使っているが，4バイト文字や5バイト文字が混入した場合の検証は不十分である．ごくまれに矢印長に過不足があったり，オブジェクト（クラス）名を表示している箱が少しずれてしまったりすることがあるが，今後の要改善項目である．

第6章　おわりに

6.1　本研究のまとめ

本研究では，料理レシピ文を解析し，それをソフトウェアの世界で使われている仕様記述言語（ここでは Java 言語）に中間的に置き換えてから，最終的には UML のクラス図とシーケンス図に変換することを行った．レシピ文を使ってこのような試みを行ったのは過去に例が見当たらない．しかしながら敢えて試みたのは次の理由からである．

- オブジェクト指向開発可能な技術者不足が深刻化している．
- ソフトウェア開発工程の上流部分の開発を支援可能なツールが必要である．
- 分散開発，海外リソースの活用などの理由もあり，UML が主流である．
- 料理の作成手順はプログラムの実行手順と非常に似ている．
- 身近にある比較的手順が少ない料理レシピが入力として使えそうだ．

オブジェクト指向プログラミングにおける設計では，フローチャートなどで処理の手順を記述することよりも，クラスやオブジェクトの関係やそのインターフェイスを設計し把握することの方が重要視されている．ここでは静的な構造をあらわすクラス図，動的な構造を表すシーケンス図を使って表示することを提案したが，料理のレシピ文を使った検証では，ある一定の成果を挙げられたと考える．

その根拠としては，オブジェクト分析の経験のない分析者であっても，本アプリケーションを操作することで，オブジェクトモデリングの初期の分析作業が行えるということが挙げられる．オブジェクト指向に不慣れな人がクラス抽出すると危険な兆候を示すことが多い．例えば「クラス名が動詞句である」とか「メソッド名が説明的である」とかいったことである．しかしながら試作したアプリケーションを使用すればそのような心配はなくなるため，これは今回開発したアプリケーションの有用性と可能性を示すことができるものと考える．

クラスの抽出については４つの抽出方法があることを示したが，本研究では名詞句抽出法を採用した．テキスト文の解析結果から品詞が認識できることから，素材名詞リストと照合することにより比較的容易にクラス候補を抽出することができたと考える．クラス抽出に関して，本研究により分かったことは，以下のとおりである．

- レシピ文に現れる素材名詞は，クラスに対応させることができるということである．
- レシピ文に現れる動詞は，直前の名詞が形成するクラスに属するメソッドとして対応させることができる．
- 名詞の中の調理機材に関しては，素材名詞とは別の認識をさせることにより対処できる．
- 名詞の中の調理動作に関しては，動作名詞として認識させることにより対処できる．

ただし，全ての用語をリストに登録しておくことは，事実上不可能である．今回使用した素材名詞辞書は，ある分野に特化したレシピ文に限り有効である．あらゆるレシピ文に対応させるためには，素材単語や動作名詞を網羅的に収集し，整理する作業を押し進める必要がある．

クラス図に特記事項を反映させるために，Java ソースプログラムのメソッドコメントを用いる方法を採用した．この仕組みが JUDE 以外のモデリングツールにも使える方法であるか検証しなければならないが，間違いなくソフトウェア開発する側からみると便利な機能であるといえる．通常は複数のドキュメントを参照しなければならないことが，モデリングツール上に自動的に記載されていることになるからである．

シーケンス図に関しては，過去の他の研究にある，プログラムを実行させた結果を蓄積してシーケンス図にしたり，プログラムの静的解析と動的解析を組み合わせてシーケンスを得たりというアプローチをしているのに比べて，シンプルな構造ながら使える手法を導き出せたと考える．ただし，提案したアイデアであるテキスト文に現れた順番に番号を付けるやり方は，実行される手順を考慮した記述となっている料理レシピ文だからこそ使えるやり方とも言える．実際のソフトウェア開発の現場では，仕様を書く人間は思いつきで書いている面もあり，また仕様追加も頻繁におこなわれることからプログラムの実行手順を意識したドキュメントとなっているものは稀である．

また，研究の本質から外れるかと思われるが，外部アプリケーションに対しキーイベントを使って操作するというやり方は，操作の自動化と誤操作防止に役立つものなので，今後も使って行きたいと考えている．

応用可能性

ここで，ソフトウェア開発における分析・設計支援ツールとしての可能性について述べる．本研究における成果をソフトウェア開発に適用することを考えた場合，ユースケースシナリオまたは要求仕様相当があることが前提ではあるが，開発メンバーの習熟度が低い場合には有用性はあると考えられる．ただしこれらの記述方法については現状ではない，新たなルールを設ける必要がある．仕様追加・仕様変更が発生した際にも，常にプログラムの実行手順に沿った記述となるよう，メンテナンスしていくルールである．実用化に向けては，レシピ文を処理する際に必要だった食材リストや動作名詞リストと同様に，

ターゲットとするドメインに特化した名詞／動詞リストを作成していく必要がある．ソフトウェア開発プロジェクトへ適用する場合，初回はこれらのリストを準備するための作業が追加で必要になると見込まれるが，同様のドメインに対する 2 回目以降の開発では，分析・設計工程での工数削減，品質向上，そしてコスト削減につながることが期待できるものと考える．是非とも実際の開発プロジェクトに導入してみたいと考える．

6.2 今後の課題

研究を通して認識した課題としては次のようなものがある．

- メソッドとして認識した順番にシーケンス図に書き込む方式のため，平行動作は表現するのが困難である．
- 繰り返し動作を認識し，シーケンス図に反映することが実現できていない．
- 複数の料理をひとつのレシピの中に書かれている場合の対処については検討できていない．
- 否定的な文章を認識できず，肯定的表現であるものとして扱ってしまっている．

特に最後のものは致命的で，品詞レベルでリストと照合している限り対策できないものなので，別のアプローチが必要であると考える．さらに，ソフトウェア開発における分析・設計支援ツールとして適用する場合には，レシピ文とユースケースシナリオ記述との共通点・相違点をきちんと分析しておかなければならない．そして相違点に関しては補間する何らかの仕組みが必要であると考える．これらについては簡単には解決できない課題であると考えている．

余談であるが，レシピ文の解析には CaboCha の Windows 版を使用したが，学習機能はないものであった．最近になって学習機能付の Windows 版がリリースされたようである．機会を見つけ，是非学習機能付の CaboCha も使ってみたいと考えている．

謝辞

本研究を始めるにあたって示唆に富むアドバイスを頂いた東条敏教授および佐々木氏に感謝しております．さらに遠路にも関わらず，たくさんの打ち合わせの機会を設けてもらい，多くのアドバイスを頂いた東条敏教授には大変感謝しております．中間審査においては，研究の方向性や問題点について的確な指摘をして頂いた，鳥澤健太郎准教授ならびに島津明教授に感謝しております．また，研究終盤に会社に迷惑をかけてしまう場面がありながらも，研究に理解を示してくださった会社の上司・同僚に感謝しております．

最後に，多大な迷惑をかけながらも様々な面において支援してくれた家族に感謝しております．本当にありがとうございました．

参考文献

[1] UML, “ Unified Modeling Language (UML)1.5 specification. OMG, March 2003.

[2] 東条敏, “ 言語・知識・信念の論理 ”, オーム社，2006.

[3] 小高 知宏, “ はじめての AI プログラミング ”, オーム社，2006.

[4] 吉田 裕之，山本 里枝子，上原 忠弘，田中 達雄, “ UML によるオブジェクト指向開発 実践ガイド ”, 技術評論社，2001.

[5] Windows プログラミング愛好会, “ UML500 の技 ”, 技術評論社，2003.

[6] SESSAME WG2, “ 組込みソフトウェア開発のためのオブジェクト指向モデリング ”, 翔泳社，2006.

[7] 日本サン・マイクロシステムズ株式会社サン・サービスエデュケーションサービス部, “ JAVA プログラミング講座 ”, アスキー出版局，2000.

[8] 前橋和弥, “ Java 謎+落とし穴 徹底解明 ”, 技術評論社，2001.

[9] 後藤英斗, 大久保弘崇, 粕谷英人, 山本晋一郎, “ 文脈に基づいたソースプログラムとドキュメント間の識別子対応付け手法 ”, 情処研報, vol.2005, no.29(2005-SE-147(6)), pp.41–48, March 2005.

[10] 注連 隆夫, 土屋 雅稔, 松吉 俊, 宇津呂 武仁, 佐藤 理史, “ 日本語機能表現の自動検出と統計的係り受け解析への応用 ”, 自然言語処理, Vol.14, No.5, pp.167-197, 2007.

[11] 浜田玲子，出井一郎，坂井修一，田中英彦, “ 料理テキスト教材における調理手順の構造化 ”, 電子情報通信学会論文誌 Vol.J85-D-II, No.1, pp.79-89, 2002.

[12] 大川寛志，白井清昭, “ 料理レシピ文に含まれる動作表現からの調理アニメーション生成に関する研究 ”, 北陸先端科学技術大学院大学 情報科学研究科 修士論文，2005.

[13] 西田悠介, 柴田知秀, 河原大輔, 岡本雅史, 黒橋禎夫, 西田豊明, “ 料理教示発話の構造解析 ”, 言語処理学会 第 9 回年次大会, 601-604, 2003.

[14] 林絵梨, “ レシピ文における時間的関係構造の自動生成 ”, 北陸先端科学技術大学院大学，修士論文, 2002.

[15] 林絵梨，吉岡卓, 東条敏, “ 日本語レシピ文における時間的関係構造の自動生成 ”, 自然言語処理，Vol.10，No.2, pp.3-17, 2003.

[16] 東条敏, “ アロー論理によるアスペクト解析 ”, 自然言語処理，Vol.7，No.4, pp.3-24, 2000.

[17] 佐々木啓子，東条敏, “ 想念・発話のスペース分割による小説の構造解析 ”, 北陸先端科学技術大学院大学 情報科学研究科 修士論文，2007.

[18] 上田世志, 瀧口（乾）伸雄, 小谷善行, “ 昔話「桃太郎」を対象とする自然言語文の意味構造自動生成 ”, 電子情報通信学会言語理解とコミュニケーション技術研究報告, Vol.NLC91 7-15, pp.25-32, 1991.

[19] 土屋雅稔, 宇津呂武仁, 松吉俊, 佐藤理史, 中川聖一, “ 日本語複合辞用例データベースの作成と分析 ”, 情報処理学会論文誌, Vol. 47, No. 6, pp. 1728-1741, 2006.

[20] 工藤 拓, 松本 裕治, CaboCha, URL“ http://chasen.org/˜taku/software/cabocha/ ”.

[21] 高橋直人, “ ニューラルネットに基づいた日本語文の解析 ”, 筑波大学博士 (工学) 学位論文，1992.

[22] 中鉢 欣秀, 小林 孝弘, 松澤 芳昭, 大岩 元, “ シナリオの図解化によるユースケースモデリング ”, 電子情報通信学会論文誌, Vol.J88-D-I, No.4, pp.813-828, 2005.

[23] 小林孝弘，中鉢欣秀，大岩元, “ 日本語要求記述に基づくプロトタイプ作成支援ツールの開発 ”, 情報処理学会研究報告，Vol.2004，No.53，pp.19-26，2004.

[24] 谷口考治, 石尾 隆, 神谷年洋, 楠本真二, 井上克郎, “ Java プログラムの実行履歴に基づくシーケンス図の作成 ”, 日本ソフトウェア科学会 FOSE2004, pp. 5-15, 2004.

[25] 堀田吉彦, 大久保弘崇, 粕谷英人, 山本晋一郎, 斉藤邦彦, “ Java プログラムからのオブジェクトフローグラフの生成 ”, 情報学ワークショップ 2005(WiNF 2005) 論文集, pp.25–31, September 2005.

[26] 堀田吉彦, 大久保弘崇, 粕谷英人, 山本晋一郎, 斉藤邦彦, “ リバースエンジニアリングによる Uml をベースとした拡張シーケンス図の生成 ”, 第 6 回情報科学技術フォーラム (FIT2007) 講演論文集, October 2007.

[27] ChangeVision, JUDE, URL“ http://jude-users.com/˜ja/ ”.

[28] 中尾信明, “ オブジェクト指向、UML に関する教育の視点と実践 ”, 情報処理学会研究報告，Vol.2004，No.49，pp.9-16，2004.

[29] 中鉢欣秀，松澤芳昭，大岩元, “ シナリオ図解化分析法によるオブジェクトモデリング ”, 情報処理学会プログラミングシンポジウム, 2003.

[30] 森本 祥一, “ ソフトウェア開発工程における支援研究と実用化への課題 ”, 産業技術大学院大学紀要, No. 1, pp. 105-110, 2007.

付 録A　生成された京大コーパスの全文

表 A.1: 生成された京大コーパス.txt の中身（前半）

分割文字	かな	文字	品詞	活用	接続
*0 6D 2/3 3.82352415					
1	イチ	1	名詞-数		
.			未知語		
梅干	ウメボシ	梅干	名詞-一般		
は	ハ	は	助詞-係助詞		
*1 2D 0/1 0.52282235					
種	タネ	種	名詞-一般		
を	ヲ	を	助詞-格助詞-一般		
*2 4D 0/0 0.20093940					
とり	トリ	とる	動詞-自立	五段・ラ行	連用形
*3 4D 0/1 1.47544134					
包丁	ホウチョウ	包丁	名詞-一般		
で	デ	で	助詞-格助詞-一般		
*4 6D 0/1 5.51951039					
たたい	タタイ	たたく	動詞-自立	五段・カ行イ音便	連用タ接続
て	テ	て	助詞-接続助詞		
*5 6D 1/2 0.00000000					
梅	ウメ	梅	名詞-一般		
肉	ニク	肉	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
*6 -1O 0/0 0.00000000					
する	スル	する	動詞-自立	サ変・スル	基本形
。	。	。	記号-句点		
EOS					
*0 6D 5/6 3.26360139					
2	ニ	2	名詞-数		
.			未知語		
鶏	ニワトリ	鶏	名詞-一般		
さ	サ	さ	副詞-助詞類接続		
さ	サ	さ	副詞-助詞類接続		
み	ミ	みる	動詞-自立	一段	連用形
は	ハ	は	助詞-係助詞		
*1 2D 0/1 1.47918518					
梅干	ウメボシ	梅干	名詞-一般		
の	ノ	の	助詞-連体化		
*2 3D 6/7 0.31545529					
種	タネ	種	名詞-一般		
・	・	・	記号-一般		
酒	サケ	酒	名詞-一般		
・	・	・	記号-一般		
塩	シオ	塩	名詞-一般		
・	・	・	記号-一般		
水	ミズ	水	名詞-一般		
で	デ	で	助詞-格助詞-一般		
*3 4D 1/2 0.90800486					
蒸し	ムシ	蒸す	動詞-自立	五段・サ行	連用形
煮	ニ	煮る	動詞-自立	一段	連用形
に	ニ	に	助詞-格助詞-一般		
*4 5D 0/1 0.09795713					
し	シ	する	動詞-自立	サ変・スル	連用形
て	テ	て	助詞-接続助詞		
*5 6D 0/2 0.00000000					
冷まし	サマシ	冷ます	動詞-自立	五段・サ行	連用形
て	テ	て	助詞-接続助詞		
から	カラ	から	助詞-格助詞-一般		
*6 -1O 0/2 0.00000000					
ほぐし	ホグシ	ほぐす	動詞-自立	五段・サ行	連用形
て	テ	て	助詞-接続助詞		
おく	オク	おく	動詞-非自立	五段・カ行イ音便	基本形
。	。	。	記号-句点		
EOS					

表 A.2: 生成された京大コーパス.txt の中身（後半）

分割文字	かな	文字	品詞	活用	接続
*0 2D 5/6 4.11877221					
3	サン	3	名詞-数		
.			未知語		
生姜	ショウガ	生姜	名詞-一般		
・	・	・	記号-一般		
青	アオ	青	名詞-一般		
しそ	シソ	しそ	名詞-一般		
は	ハ	は	助詞-係助詞		
*1 2D 0/1 0.00000000					
千切り	センギリ	千切り	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
*2 -1O 0/2 0.00000000					
し	シ	する	動詞-自立	サ変・スル	連用形
て	テ	て	助詞-接続助詞		
おく	オク	おく	動詞-非自立	五段・カ行イ音便	基本形
。	。	。	記号-句点		
EOS					
*0 2D 2/3 5.22293806					
4	ヨン	4	名詞-数		
.			未知語		
ごはん	ゴハン	ごはん	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
*1 2D 7/8 0.00000000					
1	イチ	1	名詞-数		
,			未知語		
2	ニ	2	名詞-数		
,			未知語		
生姜	ショウガ	生姜	名詞-一般		
,			未知語		
白	シロ	白	名詞-一般		
胡麻	ゴマ	胡麻	名詞-一般		
を	ヲ	を	助詞-格助詞-一般		
*2 -1O 0/1 0.00000000					
混ぜ	マゼ	混ぜる	動詞-自立	一段	連用形
込む	コム	込む	動詞-非自立	五段・マ行	基本形
。	。	。	記号-句点		
EOS					
*0 1D 2/3 1.47736398					
5	ゴ	5	名詞-数		
.			未知語		
茶碗	チャワン	茶碗	名詞-一般		
に	ニ	に	助詞-格助詞-一般		
*1 3D 0/1 2.81604332					
盛っ	モッ	盛る	動詞-自立	五段・ラ行	連用タ接続
て	テ	て	助詞-接続助詞		
*2 3D 1/2 0.00000000					
青	アオ	青	名詞-一般		
しそ	シソ	しそ	名詞-一般		
を	ヲ	を	助詞-格助詞-一般		
*3 -1O 0/0 0.00000000					
飾る	カザル	飾る	動詞-自立	五段・ラ行	基本形
EOS					
EOS					