Vol.2010-SE-167 No.21
2010/3/18

オブジェクト指向記述言語 OOJ における日本語単文の自動変換機構

畠山正行 [†1] 岩坂篤志[†1] 池田陽祐[†2]
三塚恵嗣[†2] 涌井智寛[†2]

本論文は我々の研究グループが以前から開発を進めてきたオブジェクト指向記述言語 OOJ の一部として開発した日本語単文のプログラムへの変換支援システム (JTM) に関するものである．OOJ においては，オブジェクトや対象世界の全体構造は OOSF により，最小単位の記述は日本語の単文により行う．本稿では日本語の単文を個々に対応する部分プログラムに変換する仕組みを述べる．そのため日本語の単文を平叙 NJ 単文に書き直し，個々の単文がどう変換されるかのガイドラインを五文型に分け，ユーザに書き直しを要求する．書き直された NJ 単文は設計段階以降の変換に掛けられ，最後に完成プログラムとして出力される．自動変換機構は，JTM をそのサブとし，一貫相似性を持たせた OOJ(ICSP_OOJ と呼ぶ) システムにおいて最終的に実現される．現在，OOJ や JTM は完成したばかりであり，細かな改善行うと共にある程度の記述例を作って自動変換機構の機能確認と性能評価を行う予定である．

An Automatic Transformation Mechanism of Simple Japanese Sentense in Object Oriented Description Language OOJ

Masayuki Hatakeyama ,[†1] Iwasaka Atsushi ,[†1]
Ikeda Yousuke ,[†2] Mitsuka Keishi [†2]
and Wakui Tomohiro [†2]

An Automatic Transformation Mechanism of Simple Japanese Sentense in Object Oriented Description Language OOJ. In this paper, we will report an automatic transformation mechanism (called the JTM) of simple Japanese sentence in the sub-system of the OOJ. In the OOJ, the structures of the objects and the whole target world are defined and controlled by the OOSF, and the minimum unit of the descriptions is described using the Japanese simple sentences. In this paper, we will describe a mechanism to transform the Japanese simple sentences into the program fractions (= some pieces of the partial programs) that individually correspond to just the program fractions. To attain the aim, the DU analyze in detail the Japanese simple sentences, and throw into the translator to transform the descriptions into the design-stage descriptions and finally output the complete programs. The automatic transformation mechanism contains the JTM as the sub-system of the OOJ, and its automatic transformation functions are finally realized inside the whole system of the ICSP_OOJ with the integrally consistent and similar processes). At present, the new version of the OOJ (=ICSP_OOJ) and the JTM have been developed just now. Then, we will confirm the designed functions and the performance evaluations of the automatic transformation mechanism by making use of the description examples out of the DUs.

1. はじめに

現在はソフトウェア開発の専門家以外にはプログラム開発そのものを行う人々は少なくなっているとは言え，比較的小規模のプログラム開発の必要性が高い人達は未だ非常に多数存在し，プログラム開発のニーズは依然高いと言えよう．中でもプログラム開発の必要性が高い人達の一部に，自身の専門分野専用の特殊なプログラムを作って技術設計・開発や数値シミュレーションを行う専門家の人達のカテゴリーが存在する．それが本論文で扱うドメインユーザ (以降，DU) である．

我々の研究グループでは**付録 A** に示すようなある特定の専門分野 (domain) の専門家であるドメインユーザ (以降，DU と略す．) をターゲットユーザとして，小規模のプログラムを開発するシステムを開発している．しかし，1 つの問題点として，記述の最小単位である自然日本語の短い文を従来の OOJ[1),2)] においてはそれらをどの様に program fraction(部分プログラム) に変換するのか，という点の自動化する方向での解決は無く，各 DU 任せであった．しかし想定ユーザである DU としてはその特性からして当然にプログラムへの自動変換[3)] を望む．

しかし，この問題を自然言語処理の通常の方法によって解決するには，解決に近づく場合もあるが，逆に大きな問題を抱え込んでしまう可能性もある．それは，

†1 茨城大学工学部情報工学科
Department of Computer and Information Sciences, Ibaraki University
†2 茨城大学大学院理工学研究科情報工学専攻
Graduate School of Science and Engineering, Ibaraki University

⋆1 日本語文法以外に記述方法を制約されないという意味で OOJ 内部で用いる日本語を自然日本語 (Natural Japanese) と呼ぶことにし，NJ と略称する．

(1) 現状の自然言語処理の方法をそのまま使って十分な自動変換が可能かという点については，良く知られているように実用に耐える程の自動変換率 (ほぼ 100 %) に上げることは極めて困難である．また，例えそこまで上げ得ても，残り 10 %が自動変換されなければ，DU はそこで立ち往生し，結局は実行可能な完成プログラムは生成できない．

(2) DU は自然言語処理の方法を知りたいとは思っておらず，学習する気も無いことは明らかである．

すなわち，この自然言語処理の通常の方法をそのまま用いたならば，DU が望む program fraction への自動変換は無理であることは明らかである．

そこで本研究では,「DU というユーザに合わせた」即ち「DU-oriented」な自動変換の仕組みを提案することでそれを解決し，その支援機構として日本語単文の変換機構 (Japanese Transformation Mechanism，以降，**JTM** と略) を設計・実装した．

2. OOJ という視点からの JTM の全体設計

2.1 JTM 開発の発端

我々は以前から，OOJ[1),2)] と呼ぶオブジェクト指向パラダイム (Object Oriented，以降 OO と略) に基づく記述言語系 OONJ[4)]，ODDJ[5)]，OPDJ[6)] そしてそれらを統合化した OOJ なる記述言語系を研究開発してきた．これ等の骨格となるのは OOSF[7)] と呼ぶオブジェクト指向に特有な離散化したモデル記述を構造化表現する記述の仕組みである．

この OOSF はオブジェクト単位，振舞い単位，相互関係記述の単位等でその構造化を行い表現するには適切であるが，OOSF の最小単位であるサブスロットの内部に表現された NJ 記述をオブジェクトの構造を踏まえた上で program fraction (数行程度までの部分プログラムを指す．論文 9 ページの図 6 と図 7 に例を挙げてある．) に変換するのは，現状の OONJ，ODDJ あるいは OOJ においては例外的な場合を除いて DU 任せであり，自動変換は実現できていなかった．

2.2 JTM 開発の具体的な方針

記述言語系 OOJ にはその解決を図ることが出来る手がかりが存在する．それは，OOSF を構成する集約階層構成が NJ 単文より高い階層 (サブスロットの枠) から対象世界全体の階層までは OOSF で直接に書き分けて制御することが出来るが，サブスロットの枠の内部にある NJ 記述には処理も制御も出来ない設計になっている，という点である．

OOSF における集約階層はサブスロットの枠の内側と外側で大きく二分されており，互いに独立していると言っても良い [*1] ．これは DU が自在に扱える NJ を主用するためという理由の他に，最も簡単な構造にまで分解することで NJ 記述を自動変換する可能性を残す，という理由もあった．

そこで，サブスロットの枠の内側の NJ 記述については半独立的に NJ 記述を program fraction に変換可能な形にまで変換する仕組みを構築することとした．それが Japanese Transformation Mechanism (以降，JTM と略) の開発の基本方針である．

2.3 JTM の基本条件と機能設計

DU は自然日本語 (NJ) の記述に関して以下のような特性を持つと想定できる．

(1) NJ 文は自身の書いたのであるから，それらは文法的にも意味的にも如何様にでも分析や書き直しが出来る知識と実力を持っている．

(2) 専門とする分野，すなわちドメインに関する記述は如何様にでも，形式的な構造は勿論，意味的に関わる NJ 文の同値変換等も自由自在に行える．

したがって，構文解析ツールや字句解析ツールを利用する必要が無く，また，まえがきの (1) でも述べた様にかえって障害になる可能性が出て来るので，高速大量の変換の場合には利用するが，そうでない場合等は基本的には使わない方針とする．この方針から，

(1) まず DU は自身が最初に記述した NJ 文や NJ 記述を自動変換可能な形にまで自力で書き直しや変換を行うこととする．これは第 4 章で詳細に述べる．

(2) 「自動変換可能な形にまで」変換された NJ 文は文構造を形式的に計算機に示すためのディレクション付けを施されてる．ここまでが JTM の主な任務である．これについては，第 5 章で述べる．

⋆1 ただしサブスロットが他の記述単位 (サブスロット，スロット，フレーム) と相互関係を持つ際には完全に独立しているとは言えないので注意する必要がある．

(3) ディレクション付けを施された NJ 文は OONJ トランスレータに引き渡され，設計段階の記述に変換され，必要な計算情報を付加された後，OPDJ トランスレータによって各ターゲット言語となるオブジェクト指向プログラミング言語 (以降，OOPL と略) の program fraction に変換される．program fraction は OOSF の構造に組み込まれ，ODDJ を用いた設計記述へ，OPDJ を用いた実装記述へ，そして実行可能な完成プログラムに自動変換される．

という全体構成を設計することが出来た．

3. DU 側とシステム側との作業切り分け

3.1 DU 側とシステム側の作業の切り分け基準

我々の自動変換についての考え方の出発基盤として，想定した DU の特徴を詳しく纏めた表を**付録 A の表 1** に示す．その結果，切り分け基準は以下のような結果になった．

(1) 「DU がやれると想定して良い責任範囲事項」は全てユーザの責任範囲としてやるべき義務が負わされ，必ず処理し，計算機側にやらせてはならない．逆の場合は，やる義務を負わないし，また更にいえばやってはいけない．

(2) ちょうどユーザと相補的な関係で処理の切り分けを行う．ユーザにやらせてはいけない事項や範囲は 100 %システム側で処理し，ユーザに負担させてはならない．例えば計算機の専門的な事項の詳細パラメータは DU に設定数値等を求める，等．

(3) 特に，本来 DU がやれば出来る作業を計算機/システム側で処理してはいけない．特に 100 %の処理が出来ないモノについては特にそうである．例えば，構文解析や字句解析のツールである．NJ 記述に関しては DU 自身が書いた記述なのであるから，それらに関して与えるべき文法事項等は全て DU が負うべきであり，システム側が負うべき負担はゼロでなくてはならないと考える．したがって，これ等のシステムやツールは第一義的には使わない．高速大量処理が必要な場合には補助的に用いる場合は許される．

(4) したがって NJ 記述を平叙 NJ 単文に書き直し，各要素種類の記法に従った，つまりトランスレータを使えば program fraction に変換可能な形にまで記述及び変換するのは DU 責任である．JTM のシステム側は前項に述べた補助作業と 3 章で述べた NJ 文の分類と変換の支援 (ガイドラインと記入支援) を行う．

以上の切り分けに基づいて，システム側 (計算機側) に関わる部分について，プログラムを自動的に造り出す/変換することが，本論文で指す**自動生成/自動変換**である．

3.2 DU 側とシステム側からの具体的な作業切り分け

本論文のタイトルにもある自動変換，あるいは一般に「自動プログラミング」と言われる技術は，DU とシステム側両者にとって勿論，無制限ではない．まえがきでも述べたように，両者の現状を十分考慮した上で，従来は注目されていなかった事項に着目して開発することで，DU にとって無理のない要求に対して，開発プログラムの自動生成が出来ないか，を検討する．システム側からの要求事項と提供事項は以下の様になった．

システム側 (計算機側) の要求事項と提供事項

システム側からの要求事項

(1) プログラムの構成と実行に必要な事項や情報は十分に提供されること．特に，DU の専門分野に関する事項や情報は完全に提供されること．

(2) DU には十分な NJ を用いた記述能力や文法能力があるので，基本的にはこの点は DU に任せる．

システム側からの提供事項

(1) 表現言語や形式が異なる事項や部分は ICSP_OOJ あるいは JTM 支援環境として提供する．

(2) 記述する言語に関してはシステム側でトランスレータ等を用意する．

DU 側からの要求事項と提供事項は以下の様になった．

Vol.2010-SE-167 No.21
2010/3/18

DU 側の要求事項と提供事項

DU 側からの要求事項

（1） OOPL のプログラムの構造は知らないので，OO プログラミングをサポートする専用のエディタ等の支援環境ガイドが必要．

（2） PL を用いたプログラミングには関わらないで済ませたい．つまり，OOPL の知識無しで自動変換を実現，つまり，スケルトンではなく完成プログラムの自動生成を得たい．

（3） 可能な限り計算機世界の記号や言語でではなく，自身の専門分野 (ドメイン) で常用している自然日本語 (NJ) を主用したい．

DU 側から提供事項

（1） 自身のドメインについては如何なる事項についても自由自在に書ける．したがって，自身の「対象世界のモデリングと情報の提供」は完全に実施する義務を負う．例えば，

(1) そのドメインのある物理量をモデル化した変数に関するデータ型とアクセス属性は全て DU が指定可能である．

(2) 各オブジェクトの振舞いの詳細は NJ や数式 (そのドメイン常用の形式) で自在に記述可能である．

（2） DU 自身の書いた記述であるので，形式的な構造のみでなく，意味的な部分についても如何様にでも分析や書き直しすることを厭わない．

以上の条件の内，特に DU からの要求の (2) を重視した実現を本論文で言う「自動変換」であると定義する．

4. NJ 記述の分析と変換までの DU 作業の方法

本章では NJ 記述の分析と変換までを 4 つの段階に分けて述べる．各分析と変換作業の全体を**図 1** に対応させて述べる．

第 1 段階：全て NJ 文へ変換/整形/書き直す

下準備として NJ 記述 [a] があれば分解して NJ 文 [b] の集合や集約に書き直すと共に，NJ 文自体の記述の妥当性について DU 自身が推敲し書き直しておく．この段階は DU の文章力/文作り力に完全に依存し，記述支援は単にエディタ機能のみを提供する．

ただし，現実問題としては DU は記述例を参考にしながら作業することが多く，後で書き直しが発生する NJ 記述や NJ 文を最初に書くような状況には陥らさず，殆どを「(平叙)NJ 単文」から出発すると考えられる．

---

a 本論文で **NJ 記述**とは，NJ 文の集合や集約を指す．ただし稀には文脈や前後関係によっては，単に NJ で書かれた表現を指す場合もある．

b 本論文において **NJ 文**とは，形式的には 2 つの丸（。) かピリオド（.）間のひとまとまりの文字列を指す．内容的には，複雑な文 (関係代名詞文，複文，重文など)，間接話法の文，叙情文，否定文，疑問文，倒置文，強調文，最終的には不要と判断される形容詞や形容動詞等の文飾りが付いた長い文，論理的に不整合な文，文法的に正しくない文，最終的にはプログラムには変換されない要素を含む文，その他 OO 記述に不適切な文等々を指す．ただし形式上は次の段階で出てくる NJ 単文も含まれる．また，単に文と言えば，NJ 文のことを指すとする．

Vol.2010-SE-167 No.21
2010/3/18

第 2 段階：NJ 文を平叙 NJ 単文に書き直す

NJ 文をまずは形式的に 2 つの丸（。) かピリオド（.）間のひとまとまりの文字列を切り出して NJ 単文とする．次に意味内容から DU が判断して平叙 NJ 単文に書き直す．そして，平叙 NJ 単文とその集まり (集約/集合) に変換する．

2.1 NJ 単文に，そして論理的で一意性を持つ**平叙 NJ 単文**に書き直す．

2.2 書き直しを試みた段階で，以下の三つの文構造形式については，トランスレータに掛ければ変換可能な OOJ で定めた一定の形式パターン化記述の形式に書き直す．

- 各 DU の分野の記号等で記述された数式と数学関数
- 三種類の制御構文 (二分岐，多分岐，繰り返し)
- OOJ ライブラリーに関わる対象世界共通フレーム

2.3 不成功，または不明であったら，それは第 1 段階に戻す．

2.4 それらが完了すれば，その形式や意味内容に依って以下の様な書き直しを行う．

- 平叙 NJ 単文は書き直しの必要はなく次の作業に移る．
- 叙情文は内容を変更・削除・補充し，必要な内容の叙事文の列に書き直す．
- 倒置文や強調文は同等内容の平叙文の列へ書き直す．
- 否定文は平叙文または分岐選択文に変換するか，制約記述に書き直す．
- 疑問文は，疑問内容を問い合わせる mp 文の形式に書き直す．
- 命令文は mp 文に変換する．
- 感動文は平叙文か，感動を伝えたい相手への mp 文に書き直す．

2.5 書き直した後に本段階の条件を満たさなくなったら，第 2.1 段階からやり直す．

2.6 本段階を完了すれば，NJ 文変換の核心となる五文型の判定とその処置に移る．

---

a **mp 文**とは message passing を表す NJ 単文を指す．

第 3 段階：NJ 五文型を判定して DU 処置

平叙 NJ 単文に書き直し，それを NJ 五文型という特別な文型群に限定して分析を行うことが本節で提案する変換支援システムを簡潔に構築させ，DU の利用も分かり易くする．五文型は学校英語で十分に馴れているからである．

以下に各文型における処置の代表例を示す．これ等は飽くまで代表例であって，全ての場合を尽くしてはいない．DU 判断で書き直しや整形を行う．

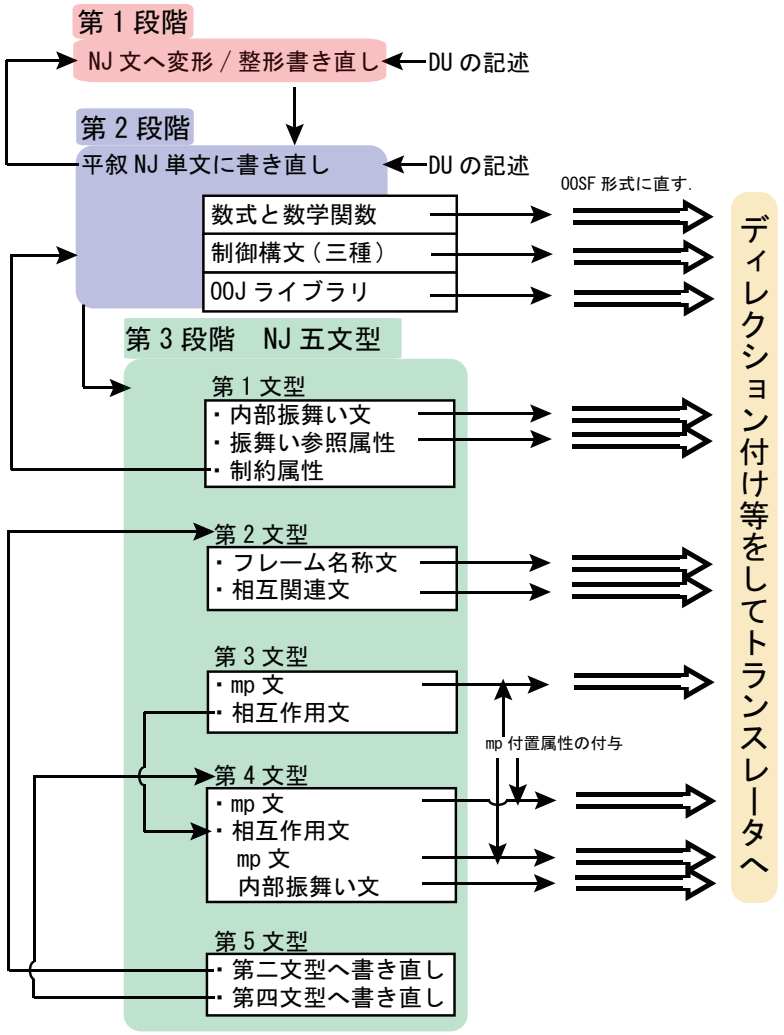

図 1　NJ 文の五文型への分析と変換の方法

第 3 段階－１：第 1，第 4 文型の場合

本文型ではトランスレータに掛けると program fraction に変換可能である．

第 1 文型 (S+V)：私 (S) は走る (V)．

- 内部振舞いを表している NJ 単文→→必要な付置属性を振舞い参照属性 NJ 単文から抽出・付置し，OONJ 形式に整形した上で，第 4 段階のトランスレータに引き渡す．
- 振舞い参照属性を表している NJ 単文→→ OONJ 形式に整形すると共に，属性名や属性値を抽出して付置記述に変更した上で，第 4 段階のトランスレータに引き渡す．(例) 私は速度＝ 40km/hour で走る．
  →→私は走る．そして「速度という属性」の実値を 40km/hour とする．
- 制約属性を表している NJ 単文→→意味内容から DU の判断で例えば何かの処理の制約条件判断を表す NJ 単文を作って内部振舞い文に付加する．
  (例) 人間は 37.5km/hour よりは遅く走る．
  →→「走る速度 V ≧ 37.5km/hour ならば警告する.」という条件文を加える．

第 4 文型 (S+Od+Oi+V)：私 (S) はボール (Od) を彼 (Oi) に投げる (V)．

- mp を表している文ならば mp 付置属性を付置し，OONJ 形式に変換して第 4 段階のトランスレータに引き渡す．
- 相互作用を表している文ならば (mp 文＋内部振舞い文)，あるいはそれ等の集まりに変換し，mp 文は前項に従い，内部振舞いを表している文は改めて分析し書き直す．
- スロット総称文は第 1〜第 5 の文型に関係なく，タイトル名詞 (名詞，動名詞) として扱う．

第 3 段階－２：第 2，第 3，第 5 文型の場合

本文型では DU がその意味等も考慮して書き直す必要がある．

第 2 文型 (S+C+V)：私 (S) は学生 (C) です (V)．

- フレーム名を指す NJ 単文→→フレーム名 (学生) のみの省略形でフレームヘッダへ記述
- 相互関連を表している NJ 単文→→相互関連の OONJ 形式に整形する．
  (例) エンジンは乗用車の部品です．
  →→エンジンは乗用車に集約されるという相互関連の OONJ 記述を行う．

第 3 文型 (S+Od+V)：私 (S) はあなた (Od) を愛する (V)．

- 間接目的語 (Oi) が未抽出か，省略した第 4 文型と見ることが出来るならば，
  (1)mp を表している文ならば mp 付置属性を検討・抽出して付し，OONJ 形式に変換して第 4 段階のトランスレータに引き渡す．
  (例) 私 (S) は彼 (Od) を推す (V)．→私 (S) は彼 (Od) を社長 (Oi) に推す (V)．
  (2) 相互作用を表している NJ 単文ならば，第 4 文型の場合へと引き渡す．
- 相手にメッセージを送るだけの NJ 単文→→本文を付置属性に変換して，Od へ mp を送る．

第 5 文型 (S+Od+C+V)：私 (S) はあなた (Od) を学生 (C) であると思う (V)．

- 第 2 文型に書き直し：主語 S と動詞 V を削除し，目的語 O を主語 S として補語 C との関係の直接表現に書き直す．→→ (例) あなた (S) は学生 (C) である (V)．
- 第 4 文型に書き直し：主語 (S) は目的語 (Od) に対して補語 C の内容を (必要なら変更し)mp 文にする．→ (例) 私 (S) は (学生であると思う)(Od) とあなた (Oi) に伝える (V)．

どちらかを DU 判断で書き直し，その文を改めて (再変換・再判断し，) 処置する．

Vol.2010-SE-167 No.21
2010/3/18

第 4 段階：OONJ トランスレータに掛ける

ディレクション付けをして，トランスレータに掛ける，つまり，分析段階の記述はこれで完成した．

前段階で DU 作業は終了しており，この第 4 段階では，第 3 段階までに DU が行った 5 文型に分類された平叙 NJ 単文に関わる情報を，自動変換させるために計算機に引き継ぐ作業を行う．それは各 NJ 単文に対して第 0 文型から第 5 文型までの文型情報と，S，Od，Oi，C，V の文要素情報を付加することである．

これで計算機は自動変換に必要な分析段階情報を受け取れる状態になる．この記述状態で分析段階記述は完成し，OONJ トランスレータに引き渡される．トランスレータは必要情報を加えた上で，設計記述に変換して出力し，次の段階へ移行する．

5. JTM：DU 作業の支援システムの構築

前章においては，特に支援システムを想定せずに DU 自身の手作業的な方式で行うように記述した．基本は DU が行うべき作業であるが，しかし，これ等の作業の中には形式的に処理できる場合や，支援システムで行う方がより DU の負担が減少する場合もある．そこで，OOJ エディタの一部のシステムとして DU の作業を支援するシステムを構築した．JTM の主な機能は以下の通りである．

(1)NJ 文の NJ 単文化機能
(2)OOJ 数式記述機能
(3)NJ 単文のディレクション機能

である．JTM の作業の流れを**図 2** に示す．図 2 は図 1 に対して，支援システムでの作業を中心に描いたものである．これらの機能は OOJ エディタの OONJ 記述の内部作業において，サブスロット内部の NJ 文を右クリックした際のメニューから選択すると起動される．

5.1 NJ 文の NJ 単文化

図 3 に NJ 文の 1 例を示す．選択したサブスロットの NJ 文が画面トップに現れるので，3 つの制御条件文かその他の単文化のボタンを押すと，3 つの制御条件文の場合は構造化した表示が下に現れるので，必要な修正を施した後に OK を押すと作業は完了する．

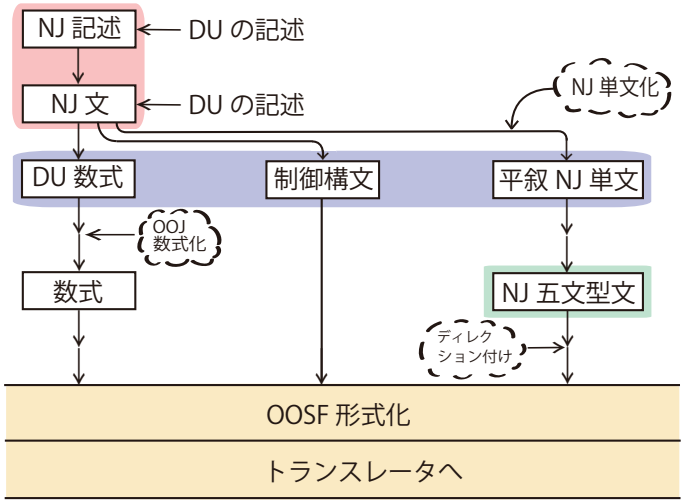

図 2　JTM の作業の流れ

5.2 OOJ 数式記述

数式に関しては，DU はまず最初に自身の分野で使われる記号や演算子，関数やその他の数式を用いて記述しておけば良い．エディタ内部のサブスロットで右クリックを行い OOJ 数式 [*1] を選ぶと，**図 4** のダイアログが現れる．最上部には選択した数式が現れるので，これを OOJ 数式に変換する．変換は基本的に手作業であるが，OOJ 数式として用意されている数学関数 (29 種類) は選択することで使うことが出来る．

5.3 NJ 単文のディレクション付け

この DU 作業は選択した各 NJ 単文に対して第 0 文型から第 5 文型までの文型情報と，S，Od，Oi，C，V の文要素情報を付加することである．エディタ内部のサブスロットで右クリックを行いディレクション付けを選ぶと，**図 5** のダイアログが現れる．最上部の文型情報

⋆1 OOJ 数式とは，OOJ において利用できる演算子，式記号，種々の表記法，数学関数，そしてライブラリーを新たに定義したものである．これに準拠した数式は program fraction に自動変換が出来るように設計してある．

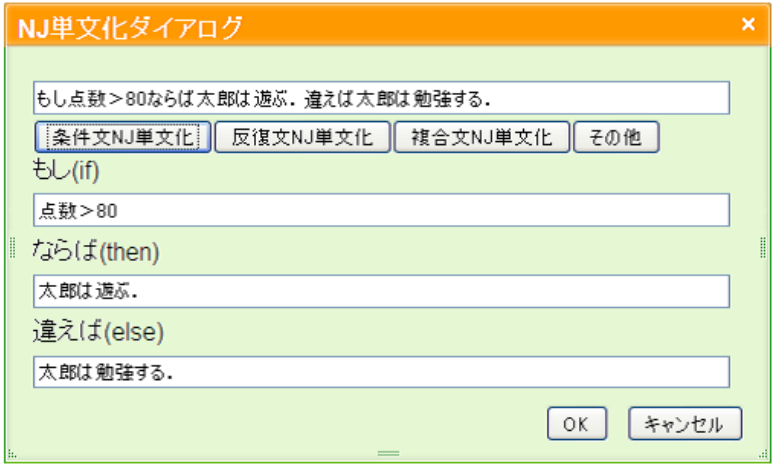

図 3　NJ 文の NJ 単文化

の内のどれかを選択すると，その文型に合わせて手動または自動でディレクション付けを行うまたは実行させる．結果は色付けされて表記されるので，それを確認して修正するか OK を押せば良い．

5.4　NJ 単文と変換した program fraction の記述例

図 6 と**図 7** に数式とメッセージパッシングの場合の分析段階の OOJ の記述と Java プログラムの fraction program の記述の変換例を示す．いずれも人工的に整形した例であるが，妥当な Java のプログラムに変換されていることが分かる．簡単のために，途中経過である設計段階の ODDJ 記述，実装段階の OPDJ 記述は省略されている．

6.　構築結果と考察

6.1　日本語単文の変換機構 JTM の構築結果

本稿では主として第 3 章で DU による対象世界についての分析段階の記述と書き直しの過程について提案し，第 4 章では DU の NJ 単文から始まり，トランスレータに引き継ぐ直前までの記述及び変換に関わる支援システムのみを示した．しかし，自動変換システムは，第 3 章と第 4 章の分析と変換の仕組み，OOJ(OONJ，ODDJ，OPDJ) の 3 つのトランスレータ等を全てを含むことに依って実現される仕組みである．

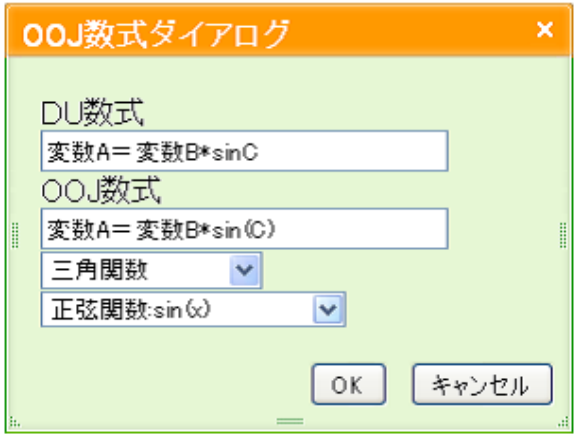

図 4　OOJ 数式記述

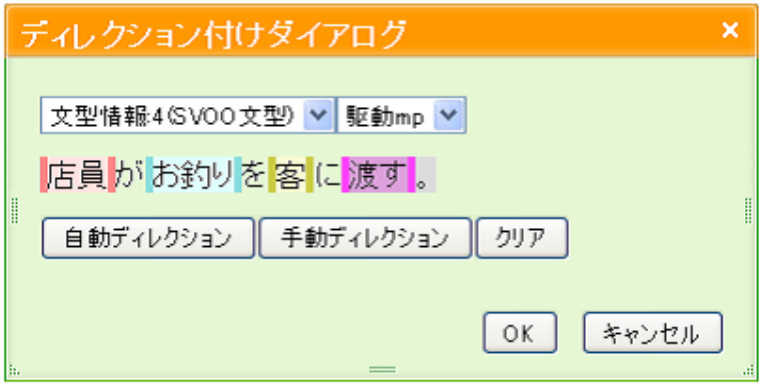

図 5　NJ 単文のディレクション付け

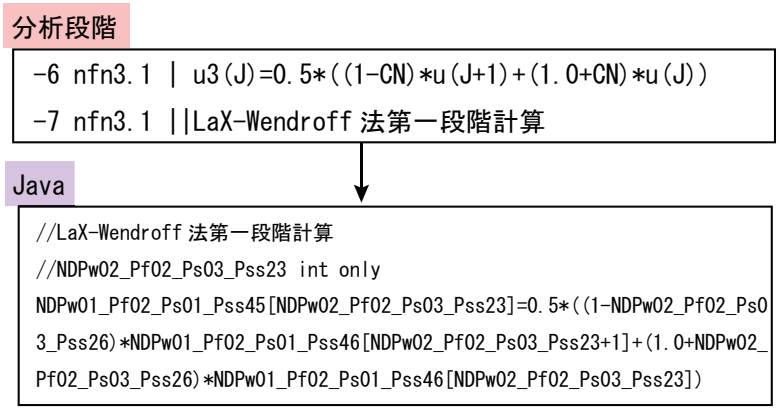

図 6　数式とその program fraction への変換例

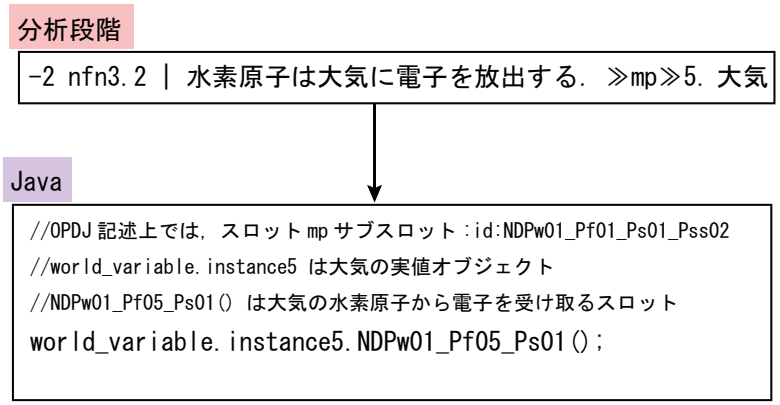

図 7　第 4 文型の平述 NJ 単文とその program fraction への変換例

本稿ではこのシステム全体を詳細に述べたわけではないが，自動変換システムの主要部分の前半部分は構築できたこと，そして第 5 章において，記述と変換の過程を経た program fraction が示されたことで，構築した仕組みが妥当な設計と実装である事が例を見る事により部分的には例証されたと言って良いであろう．

本システムを DU 側から見ると，以下の様な作業をすることになる．

(1)　OO パラダイムに基づき支援環境を用いて対象世界を分析し，
(2)　OOJ 記述支援エディタを用いて分析記述を作成し，
(3)　NJ 記述を NJ 単文化し，可能な単文は自動変換し，それ以外は DU 自身が処置する．
(4)　計算機特有のパラメータ等を同じく支援環境に従って記入する．

この 4 つの段階の中で最も煩雑な作業が (3) 項である．本論文はこの点について新たな開発によって得られた結果を報告した．

6.2　JTM が実現した主な理由

(1)　OOJ(OOSF) 側からはプログラムの内部構造とプログラム全体構成が枠組として提供される．
(2)　そのプログラムの構造内部の空白部分，即ち NJ 単文の自動変換の仕組みは JTM 側から本論文の結果として提供された．
(3)　DU の NJ 文に対する高い知識と分析及び処理能力が既存の基礎条件としてあった．

この三者が提供する三条件の組み合わせ，つまり OOJ 側と，JTM 側と，DU 側の三者のニーズと提供技術が上手くかみ合った事が JTM の実現に大きな貢献をしたと考えている．

6.3　関連研究との比較評価

【日本語プログラミング言語 (以下，J-PL と略)[8] との比較評価】

J-PL と我々の ICSP_OOJ の最も基本的な違いは，プログラミング言語 (PL) か記述言語かという点にある．J-PL のプログラムの構造がどうあるべきかを学習して理解し，使いこなせなければならないという点にある．その意味で，制約された日本語が使えるとは言え，新たなスタイルの PL を丸々一言語を学習する必要がある．その点，ICSP_OOJ は DU が理解している対象世界の OO 構造を理解していれば，OOPL のプログラム構造を気にすることなく DU のドメインでの記述に非常に近い記述を純粋な NJ を使って書くことが出来る．DU の負担という点で，ICSP_OOJ の方がメリットが大きいと言えよう．また，J-PL が科学技術分野の複雑なシミュレーションに使われた実績があるとは聞いていない．

【DEQSOL[9] との比較評価】

DEQSOL は我々のと同じ方向性や目的を持つシステムである．ただし，微分方程式に

限った数値計算プログラムを生成するという点でターゲットを絞り込んでいる点が異なる．日本語を主用するか否かという点で ICSP_OOJ に大きいメリットがある．すなわち守備範囲の広さや使い勝手の容易さという点では ICSP_OOJ からメリットを得る対象の人数が遙かに多い代わりに，生成するプログラムの規模や実用性という点で，DEQSOL の性能が遙かに勝っている．今後も DEQSOL の良い点を参考にしたい．

7. 結論と今後の課題や展望

結論として，第 2，3，4 章で述べた設計の意図はほぼ実装にまで達しており，ICSP_OOJ システム全体は未だ完全稼動ではないが，現在，新たに開発された JTM 支援システムはほぼ稼動を開始し，記述例は少数例記述できた．したがって，本論文で述べた理論的な部分に関しては，論文タイトルに近い目的をほぼ達成したと言えよう．実装した JTM 支援システムは稼動を開始しているが，記述例は未だ十分ではない．

したがって，今後の課題としては記述例を多いに増やして DU にとっての自動変換を実体として実現するためのシステムの微修正や，DU に対するガイドラインの充実等に努める．

それにより，DU と ICSP_OOJ 及び JTM 支援システムの組み合わせによる，(本論文で定義して意味での) 自動変換は実現すると考えて良いと判断している．実証が求められる．

参 考 文 献

1) 畠山正行，オブジェクト指向一貫記述言語 OOJ の構成とその概念設計，第 150 回 SE 研究会報告, 2005-SE-150，pp.49-56，(Nov. 29，2005).
2) 大木幹生，片野克紀，三塚恵嗣，沼崎隼一，涌井智寛，加藤木和夫，池田陽祐，畠山正行：三言語独立のオブジェクト指向記述言語 OOJ の実装と検証，第 163 回 SE 研究会報告，2009-SE-163，pp
3) 大野豊監修，原田実編著，自動プログラミングハンドブック，オーム社，1989 年 12 月.
4) 松本賢人，畠山正行，安藤宣晶，オブジェクト指向分析記述言語 OONJ の設計原理構築と記述環境開発，第 150 回 SE 研究会報告,2005-SE-150，pp.57-64，(Nov. 29，2005).
5) 川澄成章，野口和義，畠山正行，オブジェクト指向設計記述言語 ODDJ の設計とその記述環境の開発，第 150 回 SE 研究会報告, 2005-SE-150，pp.65-74，(Nov. 29，2005).
6) 加藤木和夫，畠山正行，オブジェクト指向実装記述言語 OEDJ の記述環境およびトランスレータの開発，第 150 回 SE 研究会報告,2005-SE-150，pp.75-82，(Nov. 29，2005). pp.49-56(2009).
7) 畠山正行，オブジェクト指向分析自然日本語構造化フレーム OOSF の設計と表現技法，日本シミュレーション学会誌，Vol.22，No.4，pp.195-209, Dec., (2004).
8) クジラ飛行机，日本語プログラム言語なでしこ公式バイブル，ソシム (株)，2008 年 6 月.
9) 佐川暢俊，金野千里，梅谷征雄，数値シミュレーション言語 DEQSOL，情報処理学会論文誌，Vol.30, No.1, pp.36-45, (1989).

付　　録

付録 A：想定 DU の特徴

表 1　想定ドメインユーザの特徴 (Domain User, DU, 特定分野の専門家)　最終改訂日：2010 年 02 月 21 日

(1) 自身が自ら深く学び，概念や方法の基礎/基盤を培い，プロの考えを積み上げ，仕事の枠組みや体系を構築して“ 最も深く影響された分野 ”を本研究では専門分野と呼ぶ.
(2) 流体や構造解析, 画像分析, 化学合成等多様な応用分野 (ドメイン) の専門家で，分野で解析，設計，シミュレーション等を行う開発技術者や研究者の多くが含まれる．計算機は利用するだけのユーザである.
(3) いわゆるソフトウェア開発/プログラム開発の
専門家ではない.
(4) 専門分野に関わる計算/解析結果だけに計算機の利用価値を認め，関心を持つ．プログラム開発については非専門家で, 実力/考え/通常の手法/特性，が千差万別．主たる関心は計算 (処理) 結果に在り，プログラムの構造や開発法等には関心は薄い/無い.
(5) 中小規模 (数千〜１万行) の複雑な自分用のプログラムの個人規模の一貫自主開発
(6) DU 自身の常用言語以外の使用は避ける.
(7) プログラムや計算機に関わる時間的/心的負担や労力は最小限にとどめたがる.
(8) 自身の専門に関しては多様で十分な実力と表現力を持つ.
(9) OO の概念は一応は理解し，使えると仮定.