Vol.2010-HPC-125 No.8
2010/6/17

ドメインユーザ・プログラミング：記述言語をベースとする仕組みの提案

畠山正行[†1]　池田陽祐[†2]　三塚恵嗣[†2]

計算機を仕事の道具として計算や処理，シミュレーションを行うことで研究や開発等を行う専門家をドメインユーザ (DU) と呼び，彼等をターゲットユーザにして，分析からプログラムを生成するための記述言語を基盤としたシステム (仕組み) をドメインユーザプログラミングと題して提案する．まず，DU プログラミングの理念や DU の特徴，DU 側とシステムを提供する側との義務と要求の切り分けや両者の要求と提供事項を仮想的に設定した上で，DU プログラミングのシステムの構築方法を提案した．これ等の結果として，本研究で提案した DU プログラミングのための OO 記述言語とその記述支援環境は目的がほぼ実現した．また第ゼロ近似の設計が比較的良い結果を生んだ結果，DU プログラミングシステムと，記述支援環境を伴った記述言語 OOJ とがほぼ同じシステムを生んだことが確認された．

Domain User Programming: A Proposal of its Mechanism based on the Description Language

Masayuki Hatakeyama ,[†1] Yousuke Ikeda [†2] and Keishi Mitsuka [†2]

In this paper, we will propose a "Domain User Programming System", that is constituted from three kinds of Object-oriented Description Languages and its Description Support Environments. The target users for this programming system are called the Domain Users (hereafter, DU) who are the specialists of a special domain. First, we have found out the characteristics of DUs and the concept of the DU-programming, search the duty and the requirements for the system between the DUs and the developer. Under these assumed design factors, we have proposed how to develop the above DU-programming system. As the results, the "DU-Programming System" has been realized. Our proposal to the assumed primary design of the DU-programming system has resulted in good products. And finally the DU-programming system we have proposed has been verified to be just equivalent to the OO description language series OOJ.

1. はじめに

計算機を専門に学んでいない人でもプログラムを作成して自身のプロとしての業務や開発，研究に用いている．彼等はプログラミングの相当の量と質の時間や神経を取られており，プログラミングからフリーであることを望むが，現実はそうは行かない[1),2)]．本研究では彼等が**図 1** の例の様なある特定の専門分野 (ドメイン) についてのプロフェッショナルである場合を想定し，彼等をドメインユーザ (Domain User，以降，**DU**) と呼び，彼等 DU が時間の消耗や負荷を最小限に押さえるプログラミングの仕組みを開発している．

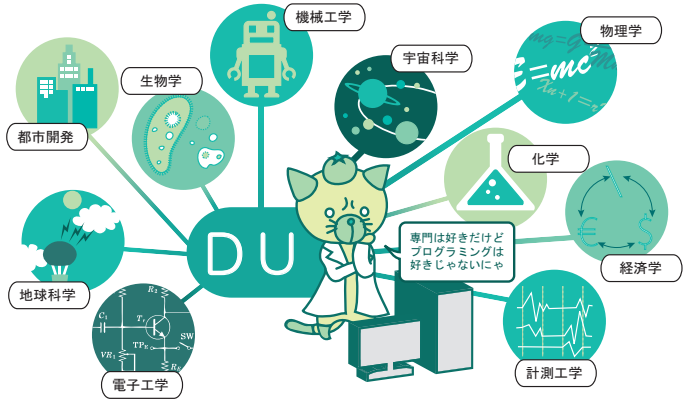

図 1　コンピュータ以外の分野の専門家である DU は「私のプログラムを代わりに作って！」と内心思っている．
Fig. 1　The DUs inwardly imagine that "I hope someone to make my programs instead of me."

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．中でもプログラム開発の必要性が高い人達の

†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
Graduate School of Science and Engineering, Ibaraki University

一部に，自身の専門分野専用の小規模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 DU である．我々の研究グループではそのような DU をターゲットユーザとして，彼等がプログラムを開発するためのシステムを開発している．

この様な DU は，その専門分野に関する記述は何でも出来ると考えて良いし，その記述を彼等の母国語の自然言語を使って記述しているであろうと考えても無理はない．そこで，彼等が自由自在に操れる自然言語を本研究では自然日本語 (Natural Japanese，以降 **NJ** と略) と設定し，NJ を使って分析記述を行い，それに多少の計算に必要な (特殊な) 記述を最小限付け加えることで，計算機用のプログラミング言語 (Programing Language，以降，**PL**) で書かれたプログラムを比較的容易に作り出せるのではないかと考えた．

2. DU プログラミングとは，そして DU とは何か？

本論文では「プログラミング」という用語を広義と狭義の二種類を区別して用いている．**狭義のプログラミング**においては，通常の計算機用のプログラミング言語を直接用いて作る作業を狭義のプログラミングと言い，その記述を狭義のプログラムという．

一方，**広義のプログラミング**においても最終的にプログラムが作られるという点では同じである．ただし，その前段階ではプログラム言語以外のある言語 A を用いて，プログラムに記述される情報と同等の情報をユーザ自身が「記述」として作成し，その記述をトランスレータを通して該当するプログラムに (半) 自動変換するという仕組みにする．この様に言語 A を用いて記述する場合の作業を広義のプログラミングと呼び，言語 A を本論文では「記述言語」と呼ばれる言語として設計して用いることとする．この記述言語の特徴は DU の考え方や思考様式，通常の業務で用いている言語や用語・用法に十分に適合させた言語に設計できる可能性が高いことにある．

2.1 DU プログラミングの理念

定義 1：「ある専門的な分野の多数のユーザ (DU) 向けに可能な限り容易にかつ効率良く，その分野のイメージに近い記述方法と環境下でプログラムを (半) 自動で生成する「広義のプログラミングの仕組み」を指す．

更に DU 側からの要求が可能ならば，

定義 2：「自身の専門分野 (Domain) の対象世界の記述やデータ等を計算機での実行にとって一意特定可能な程度に詳細に記述し，計算に必要な情報をわずかに提供すれば必要十分であり，それだけで後は，再現シミュレーション開始までの過程は DU にとってはブラックボックスである.」様なプログラミング方式を指す．

2.2 なぜ DU をターゲットユーザに設定したか？

DU をターゲットユーザにする必然性については，

(1) 著者の 1 人が以前からそして現に DU であり，本研究で目指すプログラミングの仕組みを必要としてきたという経緯がある[?),1)]．そこで想定される DU であれば，本研究で想定し設計した「仕組み」に沿ったプログラミング作業が出来るであろうし，それによりプログラムの (半) 自動生成が実現出来る可能性が高いと考えたからである．その結果として DU が本来の専門的な内容の研究開発により深くかつ効率的に専念できるであろうと考えた．

(2)DU はプログラミングについて好みはしないが多少のスキルと経験はあるので，我々が提案する分析から設計・実装・実行の一連のプログラミングの仕組みを理解し，利用する最も適した人達の一群であろうと考えた．ただし，これ等の仕組みは可能な限り DU のドメイン (＝専門分野) に近いイメージでの操作や記述が実現されていなければならない．

(3) 一方，DU にはプログラムを作成する義務や必要性が必須であるという incentive が存在する．つまり，自身の仕事に何らかの意味でのプログラムの作成と実装，それに基づく成果のアウトプットを必須としており，作らなければならない状況に置かれている．

(4) プログラムは必須であるとしても，作るためのエネルギーや神経を出来る限り割きたくないと DU は思う．つまり，DU にはしばしばプログラム作りに割く時間や神経は自身の専門分野の研究や開発にとって必要悪でしかないとの心証さえ発生することがある．そこで,「プログラムの (半) 自動生成」を実現することを想定した．その前提に立てば，ターゲットユーザ側にもある程度以上の知識や実力・経験等が必要である．第 3 章に示す DU であれば，我々が考える「プログラムの (半) 自動生成」を実現できるターゲットユーザであろうと考えた．

この (3) と (4) 二律排反をどう克服するか，について (1) と (2) を前提にすれば解決の可能性が高いと考えたのが本研究の発端であるとも言える．そこで，自身の専門分野で常用している専門用語を含んだ自然言語である日本語を用いることにした．これを最終的にはプログラムに変換しなければならない．つまり入り口は日本語，出口はプログラムである．したがってこの間を結んで変換するための仕組み，しかもターゲットユーザが DU という計算機利用に関して非専門家であるという理由から DU 向けの専用の仕組みが必須である．

2.3 小規模開発向けのプログラミングシステムの構築

小規模[⋆1]でも有効かつ効率的に機能するプログラミングの仕組みが必要である．小規模生成と小規模改修・改訂の多い DU にとってはこの方が使い勝手が良くて便利になるであろうと考える．したがって従来のソフトウェア工学に基づく大規模開発システムの様に開発自体を専門とする技術者向けのプログラム開発システムではなく，プログラムを必要とする DU 自身が自力でプログラムを作れる DU_oriented な開発法とその利用システム，即ちある種のプログラミング支援システム[⋆2]を目標とすべきであると考える．

もう一つ，重要な点がある．それは DU は分析から始めて設計，実装，そして実行 (とデータ取得，解釈) を DU 自身が一人でやらなければならない．したがってそれらを全て DU がやるには，

(1) プログラミングの作業原理は，単純明快簡潔素朴な原理に基づいていることが大前提として必要である．
(2) 分析・設計・実装・実行・管理等々において一貫して同一のモデリングと記述そしてプログラミング作業の原理に貫かれている必要がある．

と考える．そうであるとするならば，その構築原理はオブジェクト指向 (OO) パラダイム以外には有り得ないと我々は考えた．DU も離散的な考え方や扱い方には十分に馴れており，OO 採用における離散的なモデリング単位とその構造化[11] などについても十分余裕を持って理解・利用できると考えられる．

以上の状況を考え併せて，我々は「一般に言う計算科学あるいはシミュレーション技術分野」でプロ的な職業とする特定分野のプログラムが必須な DU に対して，彼等自身の分野に最も専念でき，したがって最も少ないプログラミング作業量と神経消耗で使えるような「OO に基づく何らかの言語とその記述環境」を開発・提供することを計画した．

2.4 ターゲットユーザとしての DU の特性分析

本研究の最初のテーマは本節の標題のテーマを正確に捉えることが必須である．以下の表に纏めたものを示す．これ等の纏めは，著者達の長期に亘る著者達自身，及び著者達と DU との議論の中から経験的に得られたモノを纏めたものである．なお，第 1 著者は流体ドメインの研究者である/あった．したがって，これ等の経験的データも著者の経験が多く入っているかも知れないことをお断りしておきたい．

想定ドメインユーザの特徴　1．専門家としての側面

1．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．

その依って立つ専門分野体系の基礎/基盤は，機械工学ならばいわゆる四力学，電気/電子工学ならば電気/電子回路，化学ならば化学反応論/素反応速度論，情報工学なら離散数学とプログラミングである．自身の内部に初めて確立した専門は以降に修得した分野の考え方や枠組，体系，スタイルや習慣等に大きく影響したと考えられるからである．知識の多く深いだけの分野を専門分野とは呼ばない．

2．中小規模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

中小規模 (数千〜 1 万行) の試行錯誤や改訂の多い自家用プログラムを必要に迫られ個人/小人数で一貫自主開発．そのアルゴリズムは複雑なことが多く，別言語への書き換え等は嫌がる．ただし，DU は常時新規プログラムを開発している訳ではない．例えば中規模ならば数年に 1 本，小規模ならば年 1 本という程度で，残りは改訂しつつ利用すると考えて良い．その点，常に新規開発を想定するソフトウェア開発 (OOSE) の専門家とは異なる．

3．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．

DU 自身のドメインならば，必要充分な知識があり，対象世界の詳細要素を識別でき，どんなモデリングでも縦横に行え，表現や記述も的確/正確に出来る．

⋆1 本論文で小規模とは，主として数百行から数千行の「記述」を指すものとする．一万行を超す「記述」は小規模から段階を追って改訂を重ねた後に達する規模と想定する．
⋆2 これを「小規模開発向けのソフトウェア工学」と呼びたいが，誤解を生じるだけであろう．

想定ドメインユーザの特徴　2．User としての側面

4. 応用ドメインの専門家で，計算機は利用するだけのユーザ

情報分野以外の科学/工学/技術/生産等のドメイン (専門分野) の分析/設計/計算に関わる研究/開発/技術の専門家．流体や構造解析, 画像分析, 化学合成等多様な分野で解析, 設計, シミュレーション等を行う．殆どの開発技術者や研究者が含まれる silent majority である．

5. 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.

業務支援（図書館, 経理, 銀行等の勘定系) システムや計算機特有 (通信, ウイルス対策, データベース）ソフトなどの計算機内部の擬似的な世界ではなく, 前項のような真性の実世界を対象世界とする．

6. 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, プログラムの開発法等には関心は薄い/無い.

7. DU 自身の常用言語以外の使用は避ける.

常用のプログラム言語 (多くは Fortran) と常用自然言語 (母国語，NJ) は十分に駆使でき, その知識は前提にできる．未知/異型の新規 PL は避ける．結果が出れば良く，プログラム言語に関心は無い．

8. 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.

CASE ツール，新しい PL や開発支援環境，新奇なプログラミング技法やソフトウェア開発技術等は使いたがらず，面倒臭がって避ける傾向が強い．

9. OO は概念としては一応は理解し，使えると仮定する.

OO パラダイムや概念を理解し，DU の専門分野の用語を交えた自然言語 (母国語，本テキストでは NJ(注)) でならば,OO に基づく説明や記述も充分にできる, と仮定．知識や概念として OO を知らない場合は適切に初期課程を教え, それらを前提にする必要がある．

(注)**NJ** : Natural Japanese．本来の文法を制約していない自然日本語の意．　(注) “ / ” は or を表す．)

3. DU 側と開発側双方のから義務と要求の提示と要求仕様の策定

前章で述べたように，計算機システムに対してあまり興味と知識のない DU にその設計仕様を出させようとするのは全く無理であることは経験上，十分に広く知られている[1)–3)]．

3.1 DU から要求を一セットで出させる難しさ：二すくみ関係の現出

DU にいきなり「どんなプログラム開発システムが欲しいですか？　要求をワンセットにして言ってくれたら，作って差し上げますよ.」と申し出ても，汎用性のある一セットの要求仕様に当たるような回答は帰ってこないことは経験的に知られている．

逆に，DU から「計算機システムについてどんなことがどの様に出来るかを適切に一セットにして言ってくれたら，こんなシステムが欲しいと言ってあげますよ.」と申し出られても，計算機システムに出来ることには非常に多面性がある故に「適切な一セット」を提示することなど非常に難しい．このことは開発側から見れば当然であると言えよう．

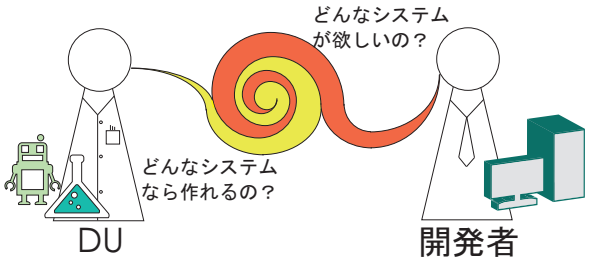

図 2　DU と開発側の義務と要求の仕様提出：二すくみ関係の現出
Fig. 2　A specifications both the duties and the requirements from the DUs and the developers

我々はこの様な状況を**図 2** に見る様に互いが他の側の当初からの完全な情報提供を前提とする「二すくみ状態/関係」に陥っていると見て，それ故にその状況から脱出するためには，DU からのシステム要求や仕様を抽出して，DU に最適なプログラミングの仕組みを提案することが難しいことを認識してきた．

3.2 二すくみ状態の解消から仕組みの設計仕様へ

前節で述べた「二すくみ関係」の解決法を前節までの議論を元に提案する．前節の分析から客観的な切り分けとして DU 側とシステム (構築) 側の両方からの最低限の要求条件と提

供条件が提出された．本節ではこれを受けて「二竦み関係」の解決する方法を以下のようなシステム構築の方針として提案する．

それは,「初期値推定 (Initial Guess) を伴う反復収束法」，すなわち数値解析などで用いることの多い iterative な解決方法である．それは，不完全な DU 要求情報，つまり前章の特性分析に加えて元 DU で現開発側にいる著者の一人からの要求仕様を織り込みながら総合し，第ゼロ要求仕様を DU の代理で構築し，それを基に設計するという方式である．

【第ゼロ近似仕様設計】前節の要求事項と提供事項に我々の推定 (初期値推定) を入れたシステムを計画設計し，これを用いて第ゼロ近似の仕様で設計と実装を行う．
初期値推定 (Initial Guess) は論理上は何でも構わない．しかし，それは収束性と収束 (＝開発) までの期間の長さを左右する．

【第一近似仕様設計】その構築結果を DU 側に示して使って貰って，その結果も取り入れて，第一近似に向けた改訂要求を DU 側と協議の上，新たな改訂仕様をこの近似仕様は協力して作り上げ，設計・実装を行う，‥‥

これを繰り返して第二近似，第三近似，‥と繰り返し，DU 側からの要求が変化しなくなったら，完成と評価する，というサイクルを行うという方針とする．以上の方法により，DU と開発側の「二すくみ状態を脱する」方法の 1 つを開発できると考える．

本節の最も重要なポイントは第ゼロ近似の仕様の作成にある．なぜならば，DU は何らかの自身向けのシステム (記述言語や記述支援環境等) を具体的に目にし，触って操作できるような“モノ”が無ければ，具体的かつシステム全体を考慮した要求を提示することは困難だからである．

なお，以上のような方法はソフトウェア工学で常識的に行われているように思われるかも知れないがそれは明らかに違う．ソフトウェア工学で開発するのは例えば，第 2.4 節で述べたような図書館情報システムのようないわゆるアプリケーションシステムであるが，DU プログラミングのために我々が開発して DU に提供するのは，DU がアプリケーションシステムを開発するための記述環境なのである．

3.3 DU 側と開発側の義務と要求の切り分け

我々の DU プログラミングについての考え方の出発基盤条件として,「想定 DU の特徴」以外に幾つかのの点についてどちらが負担すべきかの切り分け基準を設けた．これは DU の持つ実力 (例えば，自身の書いた NJ 文の文法的な分析) は彼等自身が十分に発揮・負担すべきであると考えたことに由来する．切り分けの目的は作業負担を厳密に切り分けることで，開発すべき仕組みやシステム容易あるいは簡潔にすることにある．

(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．例えば先に挙げた DU 自身の書いた日本語文の文法的な分析がそれである．

(2) ユーザと相補的な関係でシステム側の義務と負担の切り分けを行う．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なければならない．例えば計算機の専門的な事項の詳細パラメータを DU に設定させる等がそれに当たる．

(3) 計算機が処理できることであっても，本来 DU がやれば出来る作業を計算機/システム側でやってはいけない．計算機ではある程度は出来るが 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の使用である．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 が負うべきであり負うことも出来る．その場合はシステム側が行うべきではないと考える．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．高速大量一括処理が必要な場合のみ補助的に用いる．

以上の切り分けに基づいて，システム側は計算機に特有な部分についてのプログラムを自動的に生成する/変換する．一般に (半) 自動変換,「自動プログラミング[4]」と言われる技術は勿論，DU 側と開発側双方にとって無制限ではない．DU という特定のユーザと記述言語の設計を十分勘案し，従来は曖昧にされてきた事項に着目して要求と義務を切り分け，それを基に開発することで，DU というターゲットユーザに対してならば「かろうじて」プログラムの (半) 自動生成があるいは実現可能ではないかという可能性を求めて検討する．

3.4 DU 側と開発側からの要求と提供

第 3 章の DU の特徴と前節の切り分けを踏まえて，まずその基礎条件である DU 側と開発側双方からの要求事項と提供事項を述べる．これ等は，我々DU としての第 1 著者を含む開発側が想定する仮想的な要求と提供の事項の列挙である．

DU 側の要求事項と提供事項

DU 側からの要求事項

(1) 自身が常用している日本語と手続き型言語 (Fortran) 以外の言語を使いたくない．新規な OOPL の学習は避けたい．

(2) OO プログラミングに関しては，OOPL を用いた (狭義の) 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 (いわゆるスケルトンではなく) 完成プログラムの (半) 自動変換の実現を希望する．

(3) 可能な限り計算機世界の記号や用語表現を使わず，また，DU 自身の専門分野 (ドメイン) で常用している日本語や記号を主用したい．

DU 側から提供事項

(1) 自身のドメインについては完全かつ全て自由自在に書ける．したがって，そのモデリング情報と記述情報の提供は完全に実施する．例えば，
［1］振舞いの詳細記述や数式 (そのドメイン常用の形式) の記述．
［2］そのドメインのモデル化した変量に関するデータ型とアクセス属性．

(2) DU 自身の書いた記述であるので，形式的な構造や文法的な事項のみならず意味的な部分についても，如何様にでも分析，変換，書き直しは全て自己責任で行う．例えば，特定文型への変換や，主語や述語の指定等．

(3) OO 作業については本論文提案の仕組みに適合した講座等があれば学習する．

これ等を要約すると，新規な OOPL の学習やそのプログラムの作成は避けるが，自身のドメインに関わる事項や表現であれば，日本語で如何様な負担・負荷にも応える．ただし，1 つの問題点として，記述の最小単位である NJ 単文の短い文を従来の OOJ[6)–8)] 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 (半) 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[4)] を望む．

開発側 (計算機側) の要求事項と提供事項

開発側からの要求事項

(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記述という形で完全に提供されること．

(2) DU には十分な日本語 (NJ) を用いた記述能力や文法能力があるので，この点は全て DU に任せる．

開発側からの提供事項

(1) 対象世界を計算機内部の形式や文法で表現するための言語や記述形式が異なる部分については，JTM あるいはトランスレータとして可能な限り自動で変換する仕組みを提供する．

(2) DU が行う分析から設計・実装・実行までの複雑な過程を OO に基づいて作業するために，十分な記述支援環境を提供する．

(3) OO を知らない DU 向けに，特定 OOPL に依存しない一般的な広義の OO プログラミング講座を用意する．

これ等を要約すると，OOPL を知らなくても広義のプログラミングに必要な情報を全て提供されればそれが日本語で記述されていても (半) 自動変換出来る可能性があろうことが結論できる．

3.5 仮想的要求仕様に基いた“仕組み”の設計方針の構築

第 2 章，第 3 章を踏まえ，仕組みの設計戦略を以下のように第ゼロ近似仕様設計として仮決定した．DU プログラミングの最初の版はこの仕様で実現され，その結果評価に基づいてフィードバックして再設計・実装する．

1．簡潔な OO 原理に基づく広義のプログラミングの方法を用いる．
2．分析から実装まで一貫して DU に理解と記述が容易な日本語を主用した記述言語系を提供する．
3．DU は日本語で自身のドメイン (専門分野) について完全な記述や要求に対応する変換作業を提供する．
4．日本語をプログラムに変換する仕組みはシステム側が提供する．それに必要な分析や変換は DU が全て提供する．

4. OO 記述言語をベースとした DU プログラミングの提案

4.1 DU プログラミングの仕組みの構成

第 3 章までの検討を踏まえ，提案する仕組みの全体構成を**図 3** に提示し，各構成を他のとの関わりに言及しながら説明する．図 3 は DU プログラミングを表す図である．

DU プログラミングの全体構成とサブシステム

(1)OO の原理[9] ：既に述べた．
(2)OO モデリング ：離散単位要素の抽出法．既に述べた．
(3)OOSF[11] ：離散的な記述要素の構造化法
(4)OONJ[9] ：OO 分析記述言語
(5)NJ ：OONJ の主用言語で，制約されない自然日本語を指す．
(6)JTM[10] ：最小離散化単位 (NJ 単文) →→ OOPL のフルプログラムに (半) 自動変換．
(7)OOJ ：**図 3** のように分析段階の OONJ，設計段階の ODDJ，実装段階の OPDJ の 3 つの記述言語を統合化した言語を用いてプログラムに至る過程を一貫して記述・変換する記述言語
(8) トランスレータ ：同じく図 3 のように，OONJ[12)]，ODDJ[13)]，OPDJ[14)] の各段階の記述言語の後に位置し，次の段階の記述に変換する仕組み．OPDJ の後は OOPL のプログラムに変換する

a コンパイルして即，実行に移れる状態にあるという意味での完成した (完全な) プログラムを指す．ただし，ライブラリやある種の環境はこの定義範疇にはない．

上記の全体構成の中で，事項として新たな提案は (4)OONJ と (7)OOJ である．(4) と (7) は DU プログラミングのために DU 用として開発した記述言語[⋆1]である．(3) はその骨格となる OO 構造化の枠組である．DU プログラミングの仕組みの実際の中心となるのはこの「記述言語とその記述支援環境」である．本論文の主題や紙幅の関係等から (1)〜(3) や記述支援環境は割愛して別の機会に譲り，(4) と (7)，(5) と (6) の 4 項目に限定してその仕組みを順次提案する．我々は既に記述言語系 OONJ，ODDJ，OPDJ，あるいはそれらを統合した記述言語 OOJ として幾つか発表済み[7),8)] なので，詳細はそちらに譲る．

4.2 自然日本語を用いた記述とその (半) 自動変換の方法の採用

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 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．

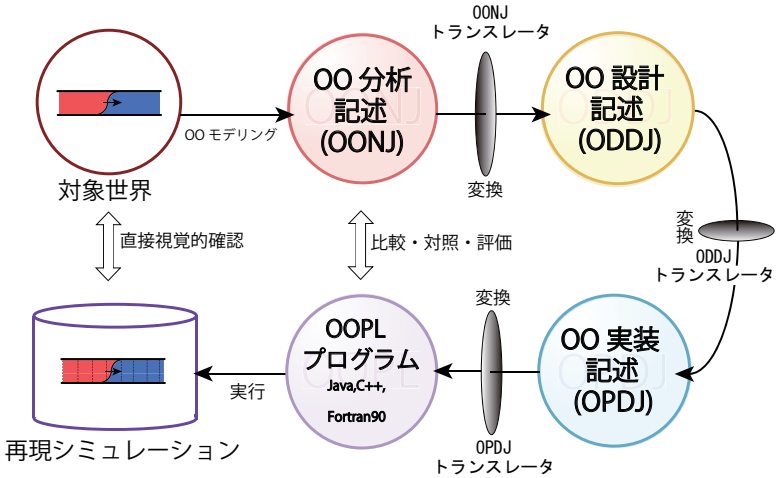

図 3 分析から実行まで一貫した過程の実現：OOJ
Fig. 3 An integrated processes OOJ throughout from the analysis stage up to the execute stage

記述言語系 OOJ には上記の二つの課題についての解決法が既に組み込まれている．OOSF に依る構造化の最小単位であるサブスロットはフレーム (オブジェクト) の中で既に OO 構造化されており，残った課題はサブスロット内部に NJ 単文によって記述された内容を OOPL のプログラムに変換することである [⋆1] ．

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．それが「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である Japanese Transformation Mechanism (以降，JTM と略) である．NJ 記述の分析と変換までを**図 4** に示す．詳細は既存の

⋆1 OONJ あるいは OOJ が正確な意味で「言語」であるか否かは議論の余地がある．しかし，体系的に纏まった OO 記述法ではあることは確かなので，言語と呼び習わしている．

⋆1 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．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 (半) 自動変換する可能性を残す，という理由もあった．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．

講演論文[10] に譲る．本研究のアプローチは機械的形式的な操作ではなく，DU という知能のある人間による知的な自力修正作業でもあるので，確実に実績を積み重ねつつある．

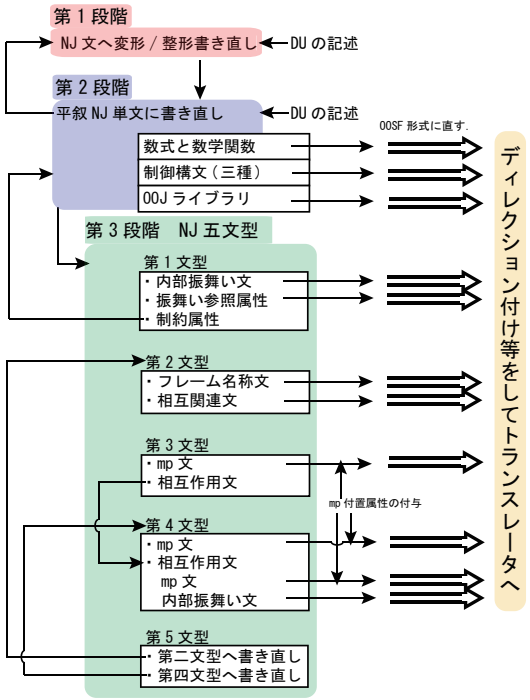

図 4　NJ 文の五文型への分析と変換の方法
Fig. 4　Analysis and transformation method from the NJ sentences into five sentence type

4.3　各段階の記述とトランスレータによる変換の例

本節では，分析・設計・実装の各段階のエディタによる記述とトランスレータによる特定の OOPL プログラムまでの変換例を**図 5** に示す．図 5 で用いた記述例の対象世界は一次元衝撃波管[15] であり，current セルは衝撃波管中の計算点を含むセルオブジェクトである．また，**表 1** に三つの記述言語間の三つのトランスレータの変換表を示す．

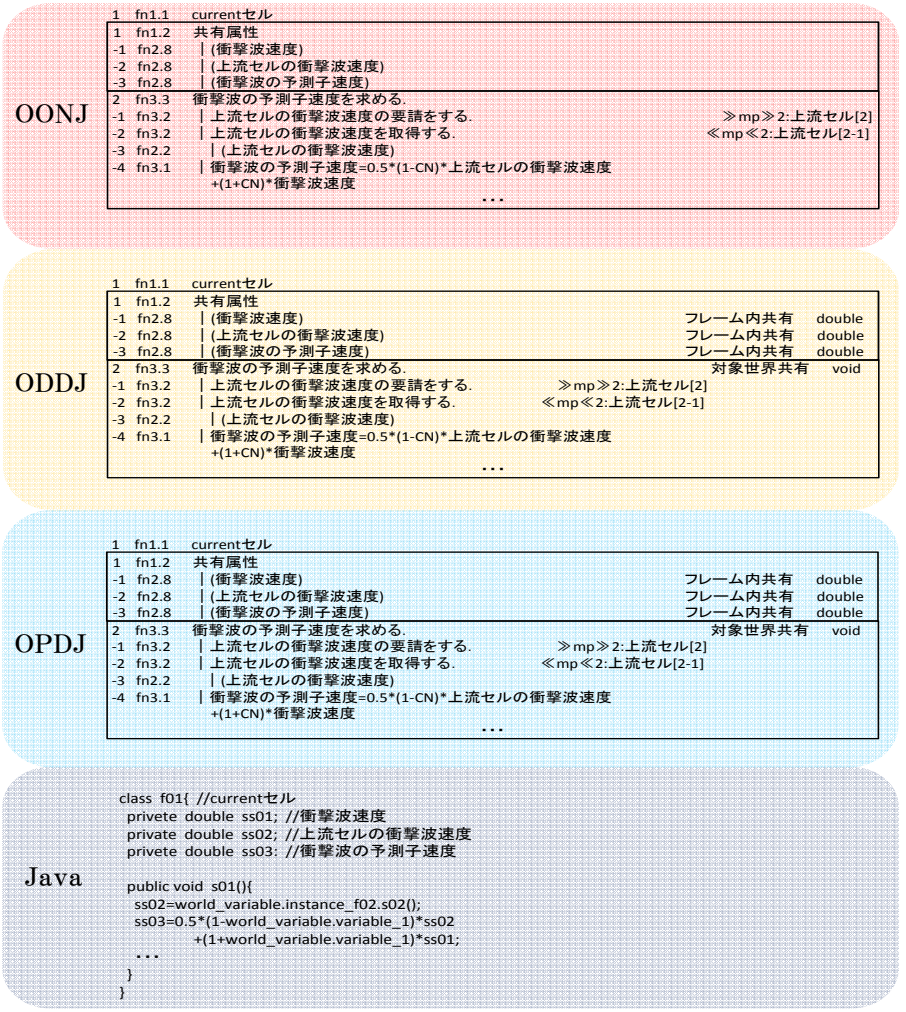

図 5　三つのトランスレータによる三つの記述言語記述の変換例
Fig. 5　Three transformation examples using three translators

表 1 OONJ から OPDJ までの各離散要素とトランスレータによる変換表
Table 1 Transformation table for the dicrete elements by the translators

OONJ の要素	ODDJ の要素	OPDJ の要素	オブジェクトベース OOPL	クラスベース OOPL
フレーム	フレーム	フレーム	モジュール	クラス
振舞い参照属性 mp 付置属性 共有属性	ローカル変数 仮引数/実引数 メンバ変数	ローカル変数 仮引数/実引数 メンバ変数	ローカル変数 仮引数/実引数 メンバ変数	ローカル変数 仮引数/実引数 メンバ変数
内部振舞い 相互作用伝達	数式/制御構文 スロット呼出し	数式/制御構文 スロット呼出し	数式/制御構文 関数呼出し	数式/制御構文 メソッド呼出し
集約 汎化/特化 実値化 引用	集約 汎化/特化 実値化 引用	集約 汎化/特化 実値化 引用	— — — 関数の上書き	— 抽象化/具象化 インスタンス化 メソッドの継承

【OONJ トランスレータ：OONJ 記述から ODDJ 記述への変換】
これは OONJ 記述を ODDJ 記述に変換する変換機構である．OONJ では計算機特有の情報が記述されない．そこで OONJ 記述を計算機の世界に持込んで適切な計算機特有の情報を付与するための共通な変換を行う．図 5 において，変換された部分は共有属性，スロット，相互作用伝達 (mp)，内部振舞いの 4 つである．共有属性とスロットにはそれぞれ可視情報と型情報が付与される．相互作用伝達と内部振舞いは表面上何も変わっていないが，記述を保存する際はそれぞれスロット呼出し，式として保存される．

【ODDJ トランスレータ：ODDJ 記述から OPDJ 記述への変換】
これは ODDJ 記述を OPDJ 記述に変換する変換機構である．ただし図 5 の例では ODDJ 記述に不備がないため，ODDJ 記述と OPDJ 記述は同一のものとなっている．特定の OOPL に特有な表現や機能を利用したい場合は，ここでその変換を行う．

【OPDJ トランスレータ：OPDJ 記述から Java までの変換】
これは OPDJ 記述から任意の OOPL のプログラムに変換する変換機構である．OPDJ 記述の要素と各 OOPL の要素との変換表を参照しながら，OPDJ 記述から各 OOPL のプログラムに変換する．現在は Java，C++，Fortran90 のプログラムに記述を変換することを想定している．

OONJ 記述から OPDJ 記述までの記述では記述上の離散単位を id で管理している．図 5 の例では，current セルは f01，current セルの共有属性である衝撃波速度は ss01 という id が割り当てられている．OPDJ 記述に登場する変数やオブジェクトは日本語名であるため，このままでは特定の OOPL プログラムに変換することができない．そこで，プログラム上では id を名前とすることで変数やオブジェクトの一意性を保ったままプログラムに変換しているのである．また，図 5 中の Java プログラムに「world_variable」という変数が登場するが，これはインスタンスと大域変数を管理するための変数である．メソッド呼出しでインスタンスを参照するときや大域変数を参照するときはこの変数から対象となるインスタンスや大域変数を参照する．

5. 提案した仕組みの設計と実装の状況，考察

5.1 DU プログラミングシステムの構築結果

システムを DU 側から見ると，**図 6** の様な手順で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) OONJ 記述支援エディタを用いて OO 分析記述 (OONJ 記述) を作成し，
(3) OONJ 記述中の NJ 記述は DU 自身が NJ 単文化し，可能な単文は JTM を用いて (半) 自動変換し，それ以外は DU 自身が処置する．
(4) ODDJ，OPDJ においても同様に順次計算機世界内部の要素の変換や OOPL 特有の記述を行う．我々の経験では OONJ 段階で 80 %，ODDJ 段階で 15 %，OPDJ 段階で 5 %の割合の負担が掛かる様である．
(5) OPDJ トランスレータからプログラムが出力されたら，そのままで即コンパイル&実行すれば良い．

5.2 DU プログラミングを支える三つの主な条件

(1) OOJ(OOSF) 側からはプログラムの全体構成とその内部構造と枠組として提供された．
(2) OOSF 内部の NJ 単文の (半) 自動変換の仕組みは JTM 側からて提供された．
(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，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本研究のシステムの実現に大きな貢献をしたと評価している．

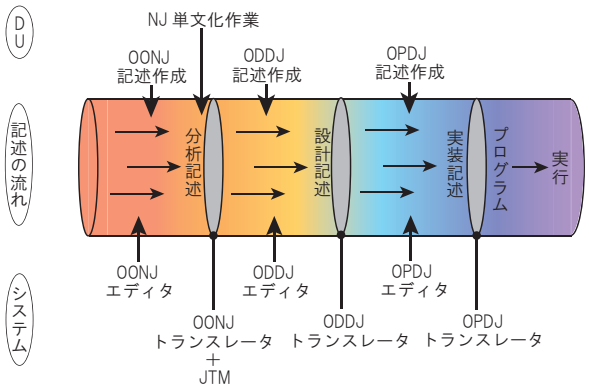

図 6 DU プログラミングの作業の構成図
Fig. 6 Constitutions for DU-programming operations

6. 結論と今後の課題

結論として，第 2〜第 5 章で述べた設計の意図はほぼ実装にまで達しており，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 や三つのトランスレータは一部稼動を開始し，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理念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するであろうことが“見えている段階”にあるといって良いと考える．

以上の成果から導かれる最も重要な結論は，DU の特徴，DU とシステム側の義務と要求の切り分けから導き出された DU プログラミングの提案した仕組みが，我々が従来から OOJ プログラミング，すなわち，記述言語 OOJ とその記述支援環境を用いたプログラミングスタイルと同義である考えても良い，という点にある．つまり，DU 側からの第ゼロ近似仕様設計と OOJ システムが近いことが十分に予想される．従って，第ゼロ近似仕様は比較的良い設計であったことになる．

今後の課題としては記述例を多いに増やして DU にとっての実用的な利用に向けたシステムの改訂やバージョンアップ等がある．

参 考 文 献

1) 畠山正行,「計算力学ユーザからの要求記述とその情報モデル構築の試み」、第 39 回応用力学連合講演会、pp.235-238，平成元年 12 月 14 日．
2) M. Hatakeyama, ”Numerical Information Environment and Its Applications to Computational Science,” Proceedings of The Eleventh Annual International Computer Software & Applications Conference (COMPSAC-87), edited by A.K. Hawkes and T.L. Kunii, pp.48-57, The Computer Society of The IEEE, l987.
3) 中所武司他,「エンドユーザ向けアプリケーション統合環境の研究開発報告書」，(財)日本情報処理開発協会 07-R003，平成 8 年 3 月．
4) 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．
5) 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).
6) 池田陽祐，大木幹夫，片野克紀，加藤木和夫，涌井智寛，畠山正行，オブジェクト指向一貫相似性記述言語 OOJ の設計と検証，第 159 回 SE 研究会報告，2008-SE-159, pp.65-74, (2008).
7) 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.
8) 池田陽祐，大木幹夫，三塚恵嗣，畠山正行，単言語方式のオブジェクト指向プロセス記述言語 OOJ の設計，第 163 回 SE 研究会報告，2009-SE-163, pp.57-64, (2009).
9) 畠山正行，池田陽祐，三塚恵嗣，加藤木和夫,「日本語を主用したオブジェクト指向分析記述言語 OONJ の設計主題と記述法」，情報処理学会第 78 回 MPS 研究会，(2010).
10) 畠山正行，岩坂篤志，池田陽祐，三塚恵嗣，涌井智寛，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，第 167 回 SE 研究会報告，2010-SE-167, (2010).
11) 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).
12) 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).
13) 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).
14) 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.
15) 池田陽祐，三塚恵嗣，加藤木和夫，大木幹生，畠山正行,「オブジェクト指向分析記述言語 OONJ の記述例を用いた簡潔な表現技法の開発」，情報処理学会第 78 回 MPS 研究会，(2010).