社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－MPS－61 (17)
2006／9／15

オブジェクト指向プログラム自動生成記述言語 OPDJ とその記述開発環境

—科学技術計算/解析/設計/シミュレーション等の応用ドメインの専門家ユーザが使えるオブジェクト指向技術の提案—

加藤木和夫* 畠山正行**

*茨城県立産業技術短期大学校 **茨城大学工学部情報工学科

情報工学から見ればアプリケーション分野である科学技術計算や設計等を自身の専門分野とするドメインユーザ(DU)向けにオブジェクト指向プログラム自動生成記述言語 OPDJ と二つのトランスレータを主要な構成とする記述開発環境を開発した. OPDJ はオブジェクト指向一貫記述言語 OOJ の実装段階に位置し, 前段階の設計記述言語 ODDJ を受けてその記述を自動変換し, DU に提示するプログラムイメージの記述言語である. DU は記述環境の支援を得て, OPDJ 上で編集作業を行い, 最小限のプログラミング情報の追記, 計算式や制御プロセスの詳細記述・修正を行う. OPDJ は ODDJ の自然日本語記述を受け継ぎ, また, DU の学習容易性を考慮した言語仕様とした. 記述開発環境はガイド機能や DU の編集作業を支援し, トランスレータは OPDJ プログラムから Java のフルプログラムを自動生成して出力する. 現時点では僅か 2 時間程度の編集作業で 3000 行程度の Java プログラムの自動生成と実行を行った例がある. これにより, トランスレータの機能が高いこと, DU の負担が従来よりも十分に軽くなったことが確認され, 本成果は DU にとって十分に有用であると評価された. 今後の課題としては, 多様な対象分野に適用して評価し, 記述言語とトランスレータ等の改良を行って実用性を高めることである.

An Object-oriented Program Automatic Generation Description Japanese OPDJ and its Description/Development Environments

--A proposal of Object-oriented technique for Domain Users in Application Domains--

KAZUO KATOUGI* MASAYUKI HATAKEYAMA**

*Ibaraki Industrial Technical College **Ibaraki University

We have designed and developed a new Object-Oriented (OO), automatic Program generation Description Japanese OPDJ and its description development environments with two-translators for the sake of the Domain Users (DUs). They analyze and simulate their target phenomena in their own expert domains. The OPDJ stands at the implementation stage of the OO integrated description languages OOJ, and have been designed so as to automatically translate the preceding ODDJ descriptions into "OPDJ program image" descriptions. The DUs edit their OPDJ program image descriptions using the description environments. The specifications of the OPDJ take over the Natural Japanese (NJ) descriptions of the ODDJ, and aim at the simple learning of the "OPDJ programming image" operations of the DUs. The description development environments support the guidance functions and OPDJ programming operations of the DUs, the translator automatically generates and output the Java full programs from the OPDJ program. At present we have one description example that 3000 lines Java program have been automatically generated and executed through only 2 hours editing operations after 40 hours analysis and 8 hours design processes. The results have shown that the load of the DUs have became sufficiently light, the functions of the translator is comparatively high, and have been confirmed to be satisfactorily useful and usable for the DUs. The future works are to apply the OPDJ to various target fields of the DU, and evaluate the validity of the OPDJ, and to improve the specifications of OPDJ and its translators, and to tuning up of its usefulness.

1. はじめに

本論文では副題に挙げた様なユーザをターゲットユーザとし, 彼等自身の記述を基にオブジェクト指向(以降, OO)プログラムを自動生成するための自然日本語ベースのプログラム自動生成の方法とシステムとを提案する. 応用ドメインの専門家ユーザ, すなわちドメインユーザ(Domain User, 以降, DU)はドメイン特有の多様で複雑な処理アルゴリズムを記述した **"プログラムの自動生成"** を望んで来た. しかし特殊な分野や特定のモデルに基づくシミュレーションシステム以外にはその様なシステムは殆ど提供されて来ず, DU は殆ど諦めかけている.

本論文では DU 向けに, ソフトウェア開発分野でその有用性が確認されている OO 技術を DU 向けに作り替えると共に, 上記の DU のプログラム開発作業システムに応用して, DU が自身の主用するプログラム言語(以降DU主言語と呼ぶ)に近いイメージで最小限のデータ入力/記述を行う

ことにより，DU 所要のプログラムが自動生成される日本語ベースの記述言語 OPDJ とその記述開発環境を提案する．本論文で提案する OPDJ は既発表の分析・設計・実装(プログラム化)の一貫記述言語系 OOJ[1]の一環として位置づけられる実装段階の言語である．

現在開発中の DU 向けの記述言語と支援環境で想定するプログラムの開発プロセスでは開発の分析／設計段階に各々の記述言語を設け，開発全体を一連の記述の変換によって最終の成果物であるプログラムを生成する．

すなわち，最初の**分析段階**では DU は分析記述言語 OONJ[2]を用いて対象世界を分析記述する．次の**設計段階**は，OONJ で分析記述された世界を具体的に計算機上で実現できるように，設計記述言語 ODDJ[3]を用いて DU と計算機世界とのインタフェースあるいはアルゴリズムに必要なデータ構造等を追記し，設計文書へと書き改める．

この設計段階までに DU が記述した内容を従来の ODDJ の言語仕様においては**表 1**，**表 2** の内容を含む．

表 1　A 群:対象世界から計算機世界への移行

・DUが対象とする世界の構成とその構造 (要素オブジェクト，属性，要素間相互関連，内部振舞い，メッセージパッシング，初期/境界条件，駆動シナリオ等々) ・計算機に格納/駆動するために必要な記述要素 (変数とデータ型，汎用的な式/メソッド/関数の表記方法，厳密に一意特定可能な駆動制御記述，アクセス制限，リンク)

表 2　B 群:プログラミング特有な作業

・ファイル入出力の詳細な手順や操作の指定 ・バッファリング操作，メモリー操作 ・入出力フォーマットの詳細な定義 ・インスタンスの生成 ・各OOPL特有な指定. (キーワード(break, return, end 等々)，演算記号，関数やライブラリの起動手順，各種の宣言文記述)

現時点で設計文書を記述する ODDJ は上記の A, B 両群の記述機能までを持つ，すなわち，現状の開発プロセスでは設計段階で DU が「できれば避けたい」と考えている B 群の「プログラミング作業」を全て DU が直接に記述しなければならない状況下にある．

そこで，本論文ではこの B 群のプログラミング作業を可能な限り避け，プログラムを自動生成させる方法論を新たに案出し，それをシステム構築に反映させ，DU に提案・提供することを目的としている．

2.　OPDJ のプログラム自動生成の原理

(I) OONJ で対象世界を充分詳細に記述し，
(II) ODDJ で計算機実行の条件を記述
していれば，後はシステム側でのプログラムの自動生成/コンパイル/実行というのが DU の希望である．

これを具現化したものが**図 1** に示す DU 向けの開発プロセスで，ここで「DU プログラミング」のイメージを持たせた実装記述言語が OPDJ であり，表現/記述された「DU プログラム」イメージが OPDJ プログラムである．

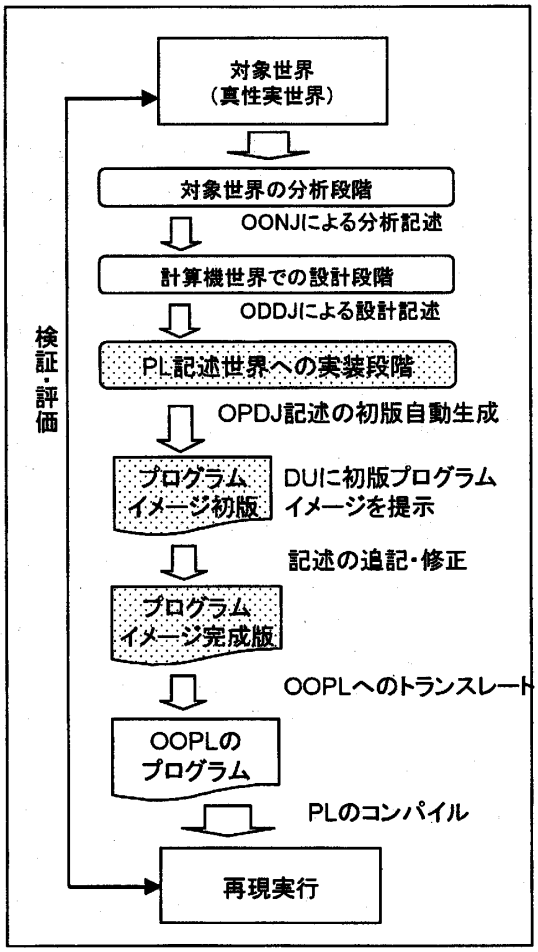

図 1「プログラムイメージ」を取り入れた DU 向けの開発プロセス

3.　OPDJ 言語仕様

3．1　DU 主言語ライクな OPDJ 表現仕様，

OPDJ の基本仕様は
(1)手続き型の Fortran ライクな記述言語の形であれば，DU は十分容易に記述指示やデータ入力を行うことが出来る，
(2)プログラムの概略はほとんど形を成した状態で初期提示される．
このことから，Fortran 風の DU 主言語を模倣する典型の“表記仕様”とした．この Fortran/DU 主言語風の OPDJ プログラム表記法で記述すると，記述開発環境の機能により，それを特定 OOPL(本研究では Java)の出力形式に変換/出力する．**これが DU プログラミングイメージである．**

3．2　OPDJ 言語仕様への要件

OPDJ の言語仕様は「DU プログラミングイメージ」に適合している必要があり，次の点が要求される．

(a) DU 主言語からの類推が容易
(b) ODDJ 文書から自動変換で出力できる
(C) 曖昧性のない一意特定識別可能な文法
(d) OOPL の仕様を意識させない**特定 OOPL 非依存**

3.3 言語仕様の特徴

OPDJ 言語仕様の詳細は OPDJ の前身となった OEDJ の仕様[4]を参照されたい. 本節ではその中で OPDJ の特徴となるものの一部を示す.

(1) プログラム構造とキーワード

表 3 に OPDJ プログラムの全体構造を示す.

表 3　OPDJ プログラムの全体構造

```
1.1 OPDJプログラム
OPDJプログラム::= フレーム+　駆動シナリオ
1.2 フレーム
フレーム::= "%フレーム" 《プログラム名》 "モジュール"
      ("%フレーム間相互関係"　関連フレーム )*
      ("%変数スロット"　変数記述 |
       "%定数スロット"　定数記述 )*
      ("%処理スロット"　処理記述 )*
      "%フレーム終了"
関連フレーム ::= 汎化 | 集約 | 一般関連
```

キーワードは ODDJ 文書からの連続性を考慮した<フレーム>等を用いる. プログラム構造は DU が対象とする世界を計算機上に実現する構造として, **表 3** に示すように OPDJ プログラムでは複数の<フレーム>と, ひとつの<シナリオプログラム>から構成する.

(2)継承は使わない

<フレーム>間の継承は許さない. これは現在想定している DU の対象世界が必ずしも継承を必要としないこと, および DU からの観点からして OOPL が持つ継承の理解しにくさや扱いの煩雑さを排除するためである.

(2) 変数とデータ型

変数とデータ型の記述例を示す.

```
%変数スロット 面積 : 実数型 = 70.0  //初期値あり
```

(3) call 文

表 4 に call 文の構文を示す.

call 文はオブジェクト指向の動作の基本であるメッセージ送信を DU に分かりやすく理解してもらうために, 手続き型の call 文とは概念的には異なるが類推し易いということで導入した. Fortran のサブルーチン呼出しに近い形とする.

表 4　call 文

```
2.3 文
call文::= "call" 参照
参照::= フレーム参照 | ライブラリ参照
フレーム参照::= 《フレーム名》(配列要素)? " . "
                《処理スロット名》 実引数
ライブラリ参照::= 《ライブラリ名》(配列要素)? " . "
                《処理名》実引数
実引数::= "(" ( 式 ("," 式 )*)? ")"
概略文::= "outline" 日本語文
マクロ文::= "inline"《マクロ文》 実引数
```

(4) アウトライン文

DU に OPDJ を「プログラムイメージ」として提示するキーとなる文としてアウトライン文を設ける. ODDJ 文書では NJ 記述になっており, そのままでは OPDJ プログラムに変換不可能な NJ 記述もあり, これらは一旦 OPDJ のアウトライン文に置換して表示した上で, OOPL への変換可能な OPDJ 文法への書き直しを DU に促す.

(5) クラス／インスタンスの取り扱い

DU は対象世界の分析／設計において OOPL におけるクラスやインスタンスを意識していない. このことに対応するために, ODDJ の OOSF 構造はオブジェクトベースの考え方[1]に基づき構成し, クラス／インスタンスの概念は導入していない.

4. 記述開発環境の設計

(1) トランスレータ

トランスレータは 2 個必要となる. ひとつは ODDJ 文書を OPDJ プログラムへ変換する. DU プログラミングイメージである「当初プログラムイメージ」(＝当初 OPDJ プログラム)を DU に提示するためのものである. もうひとつは DU によって情報の追記・修正の行われた OPDJ プログラムを OOPL のフルプログラムへと変換するためのものである.

(2) OPDJ の提示と編集

DU へ提示された「プログラムイメージ」はこの OPDJ プログラムイメージ上で追記・修正ができる編集機能が必要である. また, 追記・修正部分については DU に対して記述指示やデータ入力を促すダイアローグや確認メッセージなどを出すようにし, DU は入力要求に従って情報を追記・修正すれば OPDJ「プログラムイメージ」が完成できる記述開発機能を用意する.

5. 記述例と評価

5.1 記述例

記述例として対象世界が分かりやすく, ある程度の規模を持った「水の大気循環」を取り上げる. 記述データを**表 5** に示す. 記述時間の単位は hour である.

表 5「水の大気循環」記述データ

	フレーム	総ステップ数	記述時間
OONJ	36	921	約40
ODDJ	26	1588	約8
OPDJ	26	2483	2
Java	26	2954	0

フレーム数は設計段階以降では 26 個である. 記述時間は OONJ 記述に全体の約 8 割を費やし, ODDJ 段階では 15%のたかだか 8 時間, OPDJ に至っては僅か 2 時間(4%)でしかなく, 記述量が大幅に増えているにもかかわらず, 記述時間が極めて少なくて済んでいるという結果が非常に顕著である. その理由は, トランスレータによる自動変換機能が高く, 本記述例においては自動変換率が 95%にも達しており, 手作業で変換しなくてはならない部分が少ないことが要因である. ただし, 表 5 は DU による作業ではなく, 開発者自身によるものであり, 最良の結果を示す例と捉えるべきである. DU であれば, この 2 倍程度は掛かると見込まれる.

Java プログラムは約 3,000 行(この内 20%程度が OONJ 段階からの残存する記述等のコメント行)となり, 数千行レベルの Java のフルプログラムを本方式で自動生成可能であることが検証できたことが分かる.

「水の大気循環」の「海」フレームの一部を ODDJ と OPDJ で記述した例を**図 2** に示す. OPDJ プログラムは ODDJ 記述の NJ 記述を受け継いでいるため, 構造が構造化フレームからプレーンなテキストに変換されていても対

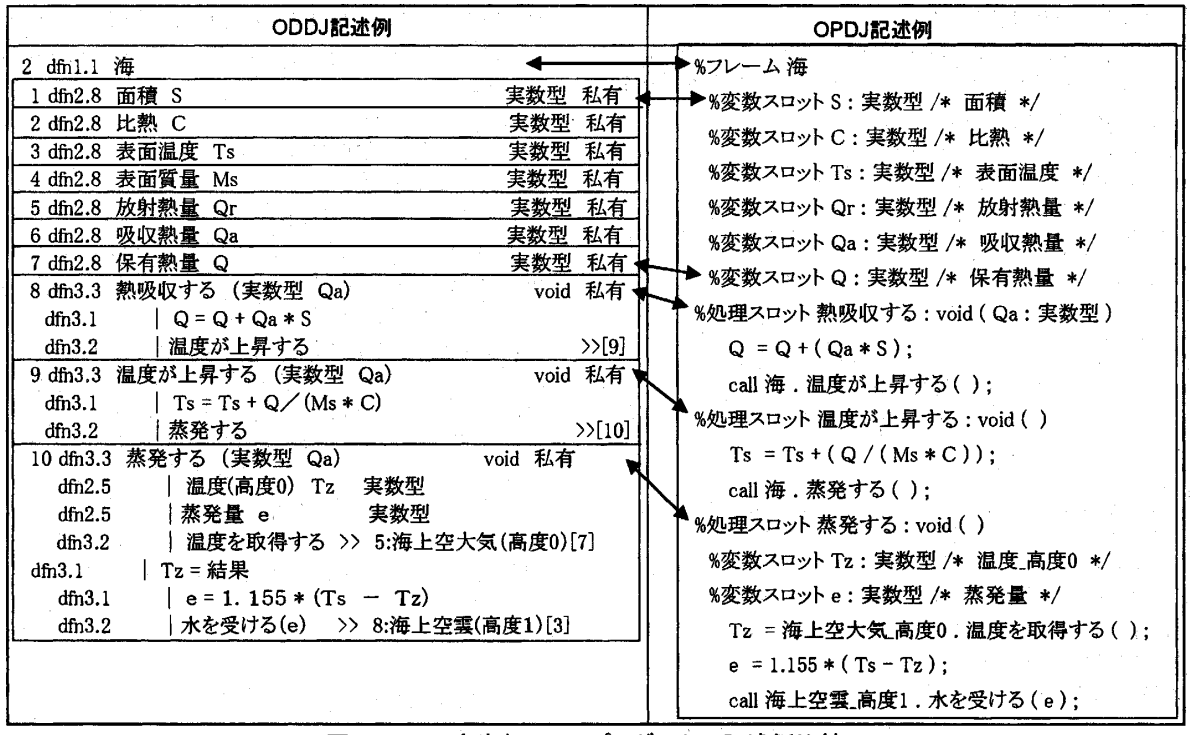

図 2 ODDJ 文書と OPDJ プログラムの記述例比較

応が分かり易いことが見て取れる. 図 2 のフレームとプログラム間ではほとんど自動変換され, その結果, OPDJ エディタには当初からほぼ全てが OPDJ 表記で表示される.

5. 2 OPDJ プログラム記述としての評価

(1) 上流の ODDJ 記述の受け入れと OPDJ 表記

NJ記述の部分はそのまま引き継ぐことで一貫性を確保できた. また, 文書構造は OOSF 構造[1]から手続き型PL風構造に変更されるが, OOSF 構造のキーワードを流用したことにより類似性が保てること, DU 自身の主言語に近い表記であること, 等のために慣れるのは容易であるとの DU のコメントを得ている.

(2) Java への変換

OPDJ には Java 特有の詳細な仕様は含まない. 特に学習容易性及び手続き型 PL 風の実現という目標から継承機能などを省いたため, プログラミングの効率性が懸念された. しかし OPDJ では効率よりも自動生成の実現を最優先にしたこと, 対象ドメインではせいぜい二万行程度を想定すれば十分であり, 継承を委譲で実現しても特に問題点は出なかった.

(3) 記述スタイル

ODDJ 記述を OPDJ 表記に表現した形式で編集するが, 最小限の Java 仕様特有の情報 (全体の一割前後) 以外は, その編集作業の量と質は ODDJ 記述の完成度により大きく変わる. これは ODDJ と OPDJ の仕様をオーバーラップさせる設計とした方針により出現した多様性であり, OPDJ 編集作業の大きな特長である. このスタイルは, DU に多様なスタイルで開発を選択させる方針になった.

5. 3 OPDJ の構成方式の評価

(1)DU 主言語にごく近い表記形式の記述言語を設計したので, DU が見知らぬ OOPL のプログラミングであることを意識することなく, 自身の DU 主言語で追記・修正するのと同じ程度の負担感で作業できるようになった.

(2)OPDJ で十分に記述すれば, OOPL(Java)のフルプログラムが自動生成されるので, Java 特有の記述法に依存したようなプログラミングの負荷は実体としては大きく削減され, ほとんど負担ではなくなった.

以上が得られたことは良好な成果と言えよう.

6. 今後の課題

今後は更に記述評価を進め, 多様な DU の多様/特殊な要求にも応えられる機能/性能を持ち合わせたトランスレータに拡張すると共に, DU が使い易い記述環境への改良を行っていきたい.

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, (2005).

2)畠山正行, オブジェクト指向自然日本語記述言語 OONJ の設計とその記述例, 第 145 回 SE 研究会報告, 2004-SE-145, pp.53-60, (2004).

3)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, (2005).

4)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).