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

2006－MPS－61 (16)
2006／9／15

オブジェクト指向ドメインユーザ計算設計記述日本語ODDJとその記述特性

–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–

畠山正行 † 川澄成章 †† 野口和義††

本論文では計算機の応用分野の専門家ユーザ (ドメインユーザ, DU) をターゲットとしたOOJと呼ぶ記述言語系の中，設計段階の記述言語ODDJの設計について述べる．従来のODDJではその記述力の不足が予測され，また記述言語としての設計基盤の妥当性や記述特性が明確ではなかった．本論文では改めてODDJ自体の設計から再検討して構築し直し，記述力を向上させると共に，その記述特性，特に“DU計算設計”の概念の明確化，汎用性の検討等を行った．その結果，記述力は向上し，設計の妥当性は検証され，ODDJのDU計算設計の概念と方法とが確立された．

An Object-oriented, Domain-User Computing Design Description Japanese ODDJ and It's Description Characteristics

MASAYUKI HATAKEYAMA ,† SHIGEAKI KAWASUMI †† and KAZUYOSHI NOGUCHI ††

In the present paper, we have developed an OO design description Japanese ODDJ that is at the design stage of the OO description language series OOJ. The OOJ is for the expert users that are called the domain user (DU) in the application domain for the computer. At present the ODDJ is lacking the description power, and the reasonability of design and the description characteristics of the ODDJ are not clarified. Therefore, we have again considered the design of the ODDJ itself and re-constituted it. The description power has been upgraded and the description characteristics and the general versatility have been re-considered. The results have shown that the reasonability of the ODDJ language design has been verified, the concept and the methodology of the "DU computing design" have been established.

1. はじめに

我々の研究グループでは，OOJ[1]と呼ぶ記述言語系とその記述支援環境とを提案している．その想定ユーザは自身の仕事を専門的な知識を持って設計/開発/研究等を行う「ドメインユーザ (以降，DUと略)」である．この記述言語系では分析段階にOONJ[2]，設計段階においてODDJ(Object-oriented Design Description Japanese)[3]，実装段階においてOPDJ[4]と呼ぶ記述言語とその専用記述エディタを配し，各言語記述間はトランスレータを用いて変換する．DUは最終的にプログラミング言語 (Programming Language 以降，PL) 記述，すなわち完成したフルプログラムを自動生成して得る．現時点において得られた分析から実装まで一貫した記述例の一つは約920行のOONJ記述，約1600行のODDJ記述，約2500行のOPDJ記述，そして約3000行のJavaプログラムへの自動変換が実現しており[4]，現在も順次その一貫記述例は増加しつつある[5]．

現時点のODDJの不十分な点は，広義の記述力の不足と，DU向けの記述言語としての妥当性や記述特性の検討である．そこで従来のODDJの言語仕様を拡張・再設計する．ODDJは対象世界の記述を計算機世界☆に変換して持ち込む機能と共に，計算機世界特有の記述を避けたいDUに対する対策も必要となる．

2. ODDJの要求仕様/設計条件の検討

ODDJへの基本要求は次の二点である．
(1)DUが容易に理解/編集/変換/修正が可能．
(2) 計算機世界で一意特定識別可能な記述を作れる．

［**上流境界条件：分析段階のOONJ**］その仕様はOONJの全ての要素や構造記述を引き継いで受け容

† 茨城大学工学部情報工学科
Department of Computer and Information Sciences, Ibaraki University
†† 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

☆ 計算機世界とは，対象世界を駆動する原理が自然法則から計算機の計算処理に代わったことを表現するための用語である．

れ可能な仕様とする．

［**下流境界条件：実装段階の OPDJ**］入力として ODDJ 記述を読み込み，DU が希望する特定のプログラミング言語記述 (現在は Java) に変換し出力する．

3. 記述言語 ODDJ の設計方針の具体化

計算機世界の記述要素種類としては全くの新規要素は現れず，全て名称か概念の変更に沿ったものである．設計結果を**表 1** に示す．

表 1　計算機世界の ODDJ 要素種類表
Table 1　ODDJ element kinds list in computing world

dfn1 フレーム	**dfn3 処理**
dfn1.1 オブジェクト	dfn3.1 内部処理メソッド
dfn1.2 –	dfn3.2 呼び出しメソッド
dfn1.3 –	dfn3.3 総称メソッド
dfn1.4 –	dfn3.4 if 文
dfn1.5 定数定義	dfn3.5 –
dfn1.6 ODDJ 関数	dfn3.6 switch 文
dfn1.7 初期設定	dfn3.7 while 文
dfn1.7.1 初期状況記述	dfn3.8 –
dfn1.7.2 境界条件記述	**dfn4 相互関連**
dfn1.8 駆動制御記述	**dfn5 定数定義**
dfn1.8.1 スクリプチャ	dfn5.1 定数定義
dfn1.8.2 駆動シナリオ	dfn5.2 単位定義
dfn2 属性	**dfn6 ODDJ 関数**
dfn2.1 参照属性	dfn6.1 ODDJ 関数定義
dfn2.2 実引数	**dfn7 初期設定**
dfn2.3 仮引数	dfn7.1 初期状況記述
dfn2.4 –	dfn7.2 境界条件記述
dfn2.5 局所変数	**dfn8 駆動制御**
dfn2.7 アクセス制約	dfn8.1 スクリプチャ
dfn2.8 共有変数	dfn8.2 駆動シナリオ

属性の扱い：計算機世界では属性の扱い方が大きく変わる．ODDJ では理論上は変量と呼ぶが，実際には DU にも変数と呼ばれることが多く，データとして扱われることになるのでデータ型や変数名も定義した．

データ型：属性にデータ型を追加する．整数型，実数型，文字 (列) 型，論理型を定義する．

アクセス制約：属性/変数/メソッドに対して，他のフレームから参照可能である“共有”と，参照ができない“私有”のどちらかを記述する．

振舞いから計算駆動処理への概念変換：OONJ における振舞いは ODDJ ではメソッドという名称に変わる．メソッドは計算に必要な計算式と制御構文等との組み合わせであり，各スロット記述が原則的には一つのメソッドである．

ODDJ 計算式の表記形式の新規設計：PL 非依存の独自の式記述形式を設計した．ODDJ における数式の表記は汎用の内部表現形式を定義/設計した．設計結果を**表 2** に示す．

ODDJ 関数群：算術関数 (三角関数，乱数，log や exp，平方根，等々)，標準入出力，ファイル入出力といった「ODDJ 関数群」として提供する．

4. 計算機世界の記述規則設計

OONJ の OO 構造化を基本的にそのまま継承した．その設計結果を**表 3** に示す．

(1) 各属性を変数 (変量) に割り当て，その変数 (変量) 毎の定義を行う．

(2)“引用スロット”のインライン展開または，別途の関数定義を行う．

(3)NJ 記述と式の並行記述 (同一内容の別表現を表す多様構造化記述)

(4) 付置属性を引数として関数の後に括弧付きで記述するので付置属性の行は消える．

(5) 抽象階層のフレームは全て無くなり，全て実働する実値フレームとなる．

(6)mp についての簡略化記法が採用されたと共に，mp の受け側の記述が省略された．

5. 記述例/記述特性/その評価

図 1 に水の大気循環現象の OODJ 記述を示す．OONJ で取り上げた抽象階層の「雲」，「海上空雲」等は設計段階では既になくなり，必要な記述は全てフレーム内にインライン展開されている．また付置属性も特別な場合以外は形式上姿を消し，メソッドの引数の形になっている．これらを除けば記述は表面上は OONJ と同様である．記述行数は OONJ 記述が 920 行程度なのに対して ODDJ 記述は 1600 行弱となった．変換に掛かった時間は正味僅か 8 時間程度である．

DU 計算設計の方法が確立：DU に対する有用な成果としては，対象世界の記述を計算機世界に変換して持ち込む設計記述の方法が明確になった．

汎用仕様の設計記述言語：ODDJ の設計およびそれに対する記述方法は，(OO)PL の影響を排除してほぼ最小限に抑え，汎用仕様に設計された．

フルプログラムの自動生成：水の大気循環の記述例については実装段階に移行した段階で ODDJ 記述の 93 %が当初 OPDJ 記述に自動変換されて直ちに表示されたことからも了解される[4]．残りの 7 %程度の記述変換は僅か 2 時間で Java のフルプログラムの自動生成され，自動生成も容易であった[4]．

6. 結　　論

ODDJ の設計は，DU の理解性と DU 自身の記述の

表 3　ODDJ 構造記述規則 (1/3)：計算機世界における拡張 OOSF
Table 3　ODDJ Structured Description Rules (1/3) : Extended OOSF in computer world

(1) **<対象世界>**
:=:<対象世界内部フレーム>$^{+}$ < 定数定義フレーム>* <初期状況記述フレーム>< 境界条件記述フレーム>
< スクリプチャフレーム>< 駆動シナリオフレーム>< ODDJ 関数フレーム>*
(2) **<フレーム>**:=:『<フレームヘッダスロット> 【スロット$^{+}$】』
(3) <フレームヘッダスロット>:=: <フレーム番号 >< sp > “dfn1.1” < sp ><フレーム名 >
(4) <フレーム番号> :=: ≪ ODDJ 記述内一連番号≫
(5) <フレーム名> :=: < NJ 記述>

(6) **<スロット>** :=: < メソッドスロット> | <変数スロット> | <式スロット>
(7) **<メソッドスロット>** :=: <「スロット総称文」><戻り値型><アクセス制約>< lf > (< mr ><サブスロット>)$^{+}$
(8) <スロット総称文> :=: 「 <スロット番号 >< sp >< “dfn3.3” >< sp >< NJ >< [“(” ><仮引数>< * “)”]
(9) <スロット番号> :=: ≪フレーム内一連番号≫
(10) **<変数スロット>** :=: 「 <スロット番号>< sp >< “dfn2.8” >< sp ><変数部>< データ型 >< “私有”|“共有” > 」
(11) **<式スロット>** :=: <「スロット総称文」><戻り値型><アクセス制約>< lf > (< mr > (<式> | <サブスロット>))$^{+}$

(12) **<サブスロット>**:=:<サブスロット記号> (< if 文> | < swicth 文> | < while 文> | <式>| <メソッド呼出し文>
| <変数サブスロット>)*
(13) <変数サブスロット> :=: 「 <行番号>< sp >< “dfn2.5” >< sp ><変数名><データ型> 」
(14) <サブスロット記号>:=:< sp ><行番号>< sp ><ファセット記号>< mr2 >< sp ><階層表記線>
(15) <階層表記線> :=: “|”

(16) <ファセット記号> :=: < “dfn” >≪ファセット番号≫
(17) <仮引数> :=: <データ型> <変数>
(18) <実引数> :=: <変数名>
(19) <アクセス制約> :=: ≪“私有” | “共有”≫
(20) <戻り値型> :=: ≪<データ型> | “void”≫
(21) <変数> :=: <変数名> [< 配列要素>]
(22) <変数名>:=:≪属性名≫ | ≪一時変数名≫ | ≪仮引数名≫
(23) <配列要素>:=: “[” <要素番号> “]”
(24) <要素番号>:=:< 算術式> //算術式のデータ型は整数型
(25) <行番号> :=: “-” ≪行番号 (スロット内一連番号) ≫

(26) < lf > :=: ≪改行して次の行の先頭へ戻る≫
(27) < rt > :=: ≪右詰め≫
(28) < sp > :=: ≪任意一定幅空白≫
(29) 『』 :=: ≪フレームを構成し，内部にスロットを収める≫
(30) 【】 :=: ≪スロットを構成し，内部に NJ 記述またはサブスロットを収める≫
(31) 「」 :=: ≪行を構成し，内部に NJ 記述を収める≫
(32) < mr > :=: ≪直上の行に 階層表記線 があれば, その位置まで右に移動する. ≫

(52) **<メソッド呼び出し文>** :=: 「(<外部メソッド呼び出し> | <内部メソッド呼び出し>)」< lf > (「<サブスロット記号>
< mr ><実引数>< [“,” ><実引数>]* 」)
(53) <外部メソッド呼び出し>:=:<相手メソッド名> “≫”<フレーム名>< [<対象スロット番号>]
(54) <内部メソッド呼び出し>:=:<相手メソッド名>“≫”< [<対象スロット番号>]
(55) <対象スロット番号>:=:< “[” スロット一意特定識別子>< [“-” <行番号>] “]”
(56) <相手メソッド名>:=:< NJ 記述 (名詞，動名詞) >

(57) **< if 文>**:=:< “if” >< sp >< if 条件文>< lf >< “if 文様式” 分岐構造>
(58) < if 条件文>:=: “(” <論理式> “)”
(59) < “if 文様式” 分岐構造>:=:< then > [< lf >< else >]
(60) < then >:=:<サブスロット記号>< “then” >< (lf ><サブスロット>)*
(61) < else >:=:<サブスロット記号>< “else” > (< lf ><サブスロット>)*

(62) **< switch 文>**:=:< “switch” >< sp >< switch 条件文 >< lf >< “switch 文様式” 分岐構造>
(63) < switch 条件文>:=:< “(” <式> “)” //ただし式のデータ型は整数型か文字型である
(64) < “switch 文様式” 分岐構造>:=:<サブスロット記号>< “case” >< sp >< case 条件文>
< “:” (< lf ><サブスロット>)$^{+}$ [< lf >< break >])*[< lf >< deafult >]
(65) < case 条件文>:=:<整数値 >| <文字値>
(67) < default >:=:<サブスロット記号>≪“default:”≫< (< lf ><サブスロット形式記号>)* [< lf >< break >]

(68) **< while 文>**:=:<サブスロット記号>< “while” >< sp >< while 条件文>< lf >< “while 文様式” 構造>
(69) < while 条件文>:=: “(” <論理式> “)”
(70) <”while 文様式”構造>:=:<サブスロット記号> (< lf ><サブスロット>)$^{+}$
(71) <戻り値>:=:<変数名>

(注 1) 二重引用符 “” で囲まれた語句や記号 (NJ 単語, ≪, ≫,fn,|,||,|||等) は語句や記号をそのまま記すことを意味する.

表 2 ODDJ 構造記述規則 (3/3) : 計算式の記述規則
Table 2 ODDJ Structured Description Rules (3/3) : Specific description rules for computing formula

```
(73) <式 >
    :=:<算術式> | <論理式> | <関係式> | <文字式>
(74) <算術式>
    :=: <算術二項式> | <算術単項式> | <基本項>
(75) <算術二項式> :=: <項>< 算術二項演算子>< 項>
(76) <算術単項式 > :=: <算術単項式>< 項>
(77) <論理式>
    :=: <論理二項式> | <論理単項式> | <基本項>
(78) <論理二項式> :=: <項><論理二項演算子><項>
(79) <論理単項式> :=: <論理単項演算子>< 項>
(80) <関係式 > :=: <項>< 関係演算子 ><項>
(81) <文字式 > :=: <文字二項式> | <基本項>
(82) <文字二項式> :=: < 項>< 文字演算子>< 項>
(83) <項> :=: < 基本項 | < "(" 式 ")" >
(84) <基本項> :=: < 変数 | リテラル
(85) <リテラル> :=: <数値> | <論理値> | <文字値>
(86) < NJ > :=: ≪自然日本語文. ≫
(87) <データ型> :=: <基本データ型 >
(88) <基本データ型>
   :=:<算術型> | <論理型> | <文字型> | <文字列型>
(89) <算術型> :=: ≪ "整数型" ≫| ≪ "実数型" ≫
(90) <整数型> :=: ≪ "整数型" ≫
(91) <実数型> :=: ≪ "実数型" ≫
(92) <論理型> :=: ≪ "論理型" ≫
(93) <文字型> :=: ≪ "文字型" ≫
(94) <文字列型> :=: ≪ "文字列型" ≫
(95) <数値> :=: <整数値> | <実数値>
(96) <整数値> :=: <数字> +
(97) <実数値> :=: <数字> + ≪ "." ≫< 数字> +
(98) <論理値> :=: ≪ "真" ≫ | ≪ "偽" ≫
(99) <数字> :=: ≪ "0" | "1" | "2" | "3" | "4"
                | "5" | "6" | "7" | "8" | "9" ≫
(100) <文字値> :=: "'" ≪英数字≫ "'"
(101) <文字列値> :=: """ ≪英数字≫ """
(102) <注釈> :=: "//" ≪任意の文字列≫
(103) <算術演算単項子> :=: ≪ "-" ≫
(104) <算術二項演算子> :=: ≪ "*" ≫ | ≪ "/" ≫
                | ≪ "+" ≫| ≪ "-" ≫| ≪ "^" ≫
(105) <関係演算子> :=: ≪ "<" | "<=" | ">"
                | ">=" | "%" "==" | "!=" ≫
(106) <論理単項演算子> :=: ≪ "not" ≫
(107) <論理二項演算子> :=: ≪ "and" ≫ | ≪ "or" ≫
```

2 dfn1.1 海

1	dfn2.8	面積S, 比熱C, 表面温度Ts, 表面質量Ms, 放射熱量Qr, 吸収熱量Qa, 保有熱量Q	実数型 私有
2	dfn3.3	熱吸収する (実数型 Qa)	void 私有
-1	dfn3.1	Q=Q+Qa*S	
-2	dfn3.2	温度が上昇する	≫ [3]
3	dfn3.3	温度が上昇する	void 私有
-1	dfn3.1	Ts=Ts+Q/(Ms*C)	
-2	dfn3.2	蒸発を行う	≫ [4]
4	dfn3.3	蒸発する	void 私有
-1	dfn2.5	温度 (高度 0) Tz、蒸発量e 実数型	
-2	dfn3.2	温度を取得する	
		≫ mp ≫??:海上空大気 (高度 0)[??]	
-3	dfn2.2	Tz	
-4	dfn3.1	Tz = 結果	
-5	dfn3.1	e=1.155*(Ts-Tz)	
-6	dfn3.2	水を受ける (e)	
		≫ mp ≫??:海上空雲 (高度 1)[??]	
-7	dfn3.2	熱放射する	≫ [5]
5	dfn3.3	熱放射する	void 私有
-1	dfn3.1	Qr=$\epsilon * \sigma$ *(Ts+0.948)^4	
-2	dfn3.2	海から熱吸収する (Qr)	
		≫ mp ≫??:海上空大気 (高度 0)[??]	
-3	dfn3.2	温度が低下する	≫ [6]
6	dfn3.3	温度が低下する	void 私有
-1	dfn3.1	Ts=Ts-Qr/(Ms*C)	
7	dfn3.3	海へ流出する	void 私有
8	dfn3.3	雨/雪を受ける	void 共有
9	dfn3.3	大気から熱吸収する (実数型 Qa)	void 共有
-1	dfn3.1	Q=Qa*S	
10	dfn3.3	表面温度を設定する (実数型 表面温度 temp)	void 共有
-1	dfn3.1	Ts = temp	
11	dfn3.3	保有熱量を設定する (実数型 保有熱量 temp)	void 共有
-1	dfn3.1	Q = temp	

図 1 ODDJ 記述例 (水の大気循環現象)
Fig. 1 ODDJ description example: atmospheric circulation of water

ための記述性と，OOPL 実装を次の段階で行うための一意特定識別記述，という相反する 2 つの要求を解決し，かつ汎用性を持たせた仕様の設計とした．

その結果，水の大気循環の一貫記述例等を通して，DU 計算設計の方法の確立が確認され，汎用仕様の設計記述言語が強い記述力を持つこと，そしてフルプログラムの自動生成に貢献したことが明らかにされ，良好な記述特性を示す設計結果を得た．

参 考 文 献

1) 畠山正行，松本賢人，川澄重章，齋藤正樹，加藤木和夫，野口和義，安藤宣晶，オブジェクトベース一貫日本語記述言語 OOJ に基づくプログラムの自動生成，第 58 回 MPS 研究会報告,2006-MPS-58, pp.55-58, (Mar. 17, 2006).
2) 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).
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, (Nov. 29, 2005).
4) 加藤木和夫，畠山正行，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，第 61 回 MPS 研究会報告,2006-MPS-61, (Sept. 15, 2006).
5) http://gaea.cis.ibaraki.ac.jp/