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

2009－SE－163 (8)
2009／3／18

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

池田陽祐 †† 大木幹生 ††
三塚恵嗣 † 畠山正行 †

非情報系の科学技術諸分野の専門家 (ドメインユーザ, DU) が扱う対象世界をオブジェクト指向 (OO) に基づき自然日本語を主用して記述し，相似性の高い再現シミュレーションを実現するために，OO 一貫相似性の概念を提案した．また，一貫相似性を検証可能にするために分析・設計・実装の各段階毎に OO 記述言語を設計し実装した．その結果，記述言語系を定義・設計し，計算機で扱えるように記述支援エディタを実現しいくつかの記述例を得る事に成功した．そこで，次の段階として各段階毎の記述言語を一つに統合した言語を設計することで，当初想定した単一仕様言語を設計するための資料が得られると考え，三段階の言語を一つに統合した単言語方式 OOJ を設計し，単一仕様言語を設計するために必要な対象世界の要素種類とプログラムの構成要素の対応関係を導出した．今後は導出した資料をもとに単一仕様言語の設計を行うことが次の段階の課題である．

A Design of Object-oriented, Integarated Singleton Process Description Language OOJ

Yousuke Ikeda ,†† Mikio Ooki ,†† Keishi Mitsuka † and Masayuki Hatakeyama †

We have proposed the concept of the Object-oriented, integrally consistent similarity to realize the reemergent simulations with high similarity. This concept will be realized for the Domain Users (hereafter, DU) who are the specialists of some expert science/technology domains except the information domain. It can be concluded that the OO description languages at each stage of the analysis (OONJ), design (ODDJ), implementation (OPDJ) are needed to make verifiable the similarity. A description support editor has been developed and verified to handle the descriptions for the analysis stage, the design stage, the implementation stage, and finally for the OO programming stage in the computer.

As the results, we have been able to be concluded that the integrally consistent similarity has been verified by making use of the same description examples throughout the OONJ, the ODDJ, the OPDJ, and finally the Java program.We have considered that these three kinds of description languages could be integrated into the singleton description language OOJ by making use of the above described examples.Then, we have designed the OO description language OOJ, and have written upsome description examples that follow the syntax (description rules) of the OOJ, and have succeeded in developing the OO programming language (Java) program.

In the future work, we have planed to develop the OOJ for the preparation of the Object-oriented, singleton specified description language with an integrally consistent similarity.

1. はじめに

本研究のそもそもの発想は，著者の１人が未だ気体力学の数値シミュレーションを専門分野 (ドメイン) としていた頃に持った疑問，すなわち「実世界の流れと，計算機内部でのシミュレーションの間に十分に高い相似性☆を持たせ，かつ形式的に検証するモデリングやプログラミング方法論はないのか？」という問いに端を発している．その疑問は情報工学を専門にし始めてからは研究テーマとして発展した．そこで辿り着いたのが，オブジェクト指向 (以降，**OO** と略) パラダイムである．

OO では対象世界 (例えば着目する流れという部分世界) を離散単位で分析し，プログラムを設計し，そ

† 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

☆ 本論文では，“同等内容の別表現”としての意味で用いる．

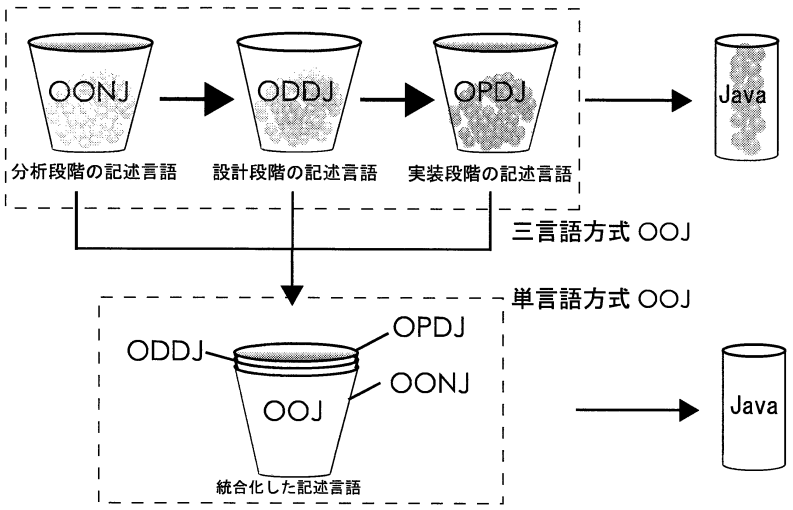

図 1 統合化された単言語方式 OOJ のイメージ図

Fig.1 An Image for the Integrated Singleton Language OOJ

れをプログラミング言語 (以降, **PL** と略す) を使って実装するという一連の過程 (プロセス) について, OO という抽象的には同一のモデリング/ ☆プログラミングのパラダイムを一貫して用いることが出来ることを見出した. その成果としてその様な様相の異なる複数の段階を含む一連の過程を仔細にわたって追跡, 比較評価, 相似性検証可能であることを見出した.

それは「OO パラダイムに基づく一貫相似性モデリング過程」という概念に集約される. つまり, 対象世界の分析・設計・実装・駆動という各段階での OO に基づく記述/プログラムを比較対照すれば, 分析→設計→実装→駆動の各段階間での相似性の有無や精度を明らかにでき, それを四段階間で一貫して実施することで対象世界の物理現象と計算機世界内部のシミュレーションとの比較対照と評価が可能になり, その結果としてシミュレーションの相似性の高さと一貫した過程であること自体もが検証が可能になる, という方法であった.[3)4)]

しかし, この方法論には大きな問題点が存在した. それは, 相似性の高さの評価についても, その相似性に関わる検証についても, 形式的な (したがって, 客観的) 検証のための道具立てが不足したのである. そこで考案したのが記述言語である.

ただし, OOJ を使うと想定されるユーザを**表 1** にあるように一定の専門分野 (ドメイン) を持つドメインユーザ (以降, **DU** と略) とした故に, 自然言語である日本語をベースとした記述言語である必要があった.

そのため, OO に基づく構造化記述のために OOSF[1)] が提案され, OO 記述の最小単位を自然日本語 (以降, **NJ** と略) で記述する方法が構築されていった. しかし, このアイデアを適用しても, 分析・設計・実装・駆動という大きく異なる四つの世界を同時に表現できる記述言語を設計することは非常に困難であった. さらに, それらの記述から PL を用いたプログラムが自動生成されなければ DU にとってのメリットは殆どないので自動生成のシステムも必須である. しかし, いまだにそういう異なった複数の広い世界を記述できる言語あるいは記述/プログラミング方法は構築されていない.

そこで我々の研究グループでは, まずは分析/設計/実装/駆動の各段階において NJ を基盤とした OO 記述言語という共通性を持たせながら, 分析/設計/実装/駆動の各々の世界に特有な事象については各々の段階の記述言語として設計し実装する方針とした. それらが本研究会で同時に発表した三言語独立のオブジェクト指向記述言語 OOJ[5)](以降　三言語方式 OOJ) である. この発表において, 一貫相似性を重視した三つの言語を実装し, これらの言語を用いて 200 行程度の記述 (一次元衝撃波管のシミュレーション記述) と他には短い幾つかの記述について, 一貫した記述 (分析→設計→実装→駆動) が得られた.

☆ 本論文では, " / " はその前後の語句の or を指し, "・" はシーケンシャルなプロセスである事を指す.

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)

Table 1　Features of Domain Users: DU

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，プロの仕事の体系を構築した“初めての分野”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．それ以降に修得した分野は本論文では専門分野とは呼ばない．人は初めて確立した専門分野の考え方，体系，習慣等に大きく依存した「自身の枠組み」を作って持ち続けるてしまうと考えられるからである．勿論，例外もいるが.

2．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.

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.
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）ソフトなどの擬似実世界ではなく，前項のような真性実世界を対象.

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は居る訳ではない．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.

6．DU 自身の常用言語以外の使用は避ける.
常用 PL(注)(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，PL に関心は無い.

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.
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.

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.
DU 自身のドメインに関することならば，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うことが出来，表現や記述も的確/正確に出来る.

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)**PL**：Programming Language,　**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.)

これによって，分析・設計・実装・駆動の段階間の接続に関する記述資料を取得できた．そこで，三言語方式 OOJ で取得した資料を元にし，一貫相似性モデリング/プログラミングの方法開発の構想段階において当初計画してきた，OOJ という単一仕様の言語を用いた一貫相似性記述作成を行うための言語仕様を探りたい.

そのためにまずは，三つの言語を一つに統合した言語 (以降　単言語方式 OOJ) を設計する事を行った．**図 1** は，三言語方式 OOJ と単言語方式 OOJ の設計のためのイメージである．コップが言語の仕組みを表し，中の玉がプログラム (フレーム) を比喩的に表す．三言語方式 OOJ は各段階毎にコップで表現された記述言語が存在し，それぞれ分析段階の OONJ，設計段階の ODDJ，実装段階の OPDJ となっている．三つのコップがよく似ている事は三言語の相似性も暗示している．その記述言語間で重複している箇所を下の図の様にコップで重ねる事で三言語を統合した言語が単言語方式 OOJ である.

本論文においては，三言語独立のオブジェクト指向記述言語から得られた資料を元に (1) 三つの言語を一つに統合した言語を設計し，そこから (2) 単一仕様言語の OOJ を設計するための設計方針や参考データの取得を試みた結果についての結論が纏まったのでその報告を行う.

2.　三言語と単言語の両方式の OOJ の違い

第 1 章で述べた単言語方式 OOJ と三言語方式 OOJ の異なる点についての記述を本章では行っていく.
第一に，最も異なる点は一貫相似性記述作成を行うために DU が学習する記述言語の数である．この問題は，以前から指摘されている点であり，三つの記述言語は類似性が高いため異なる三つの記述言語を学習するよりは DU の学習負担を軽減しているが，一つの記述言語を学ぶ負担よりは多いと言える.

第二に，三言語方式 OOJ では，三つの言語に準じた記述に変換されるため一貫相似性を形式的に検証するには各段階毎に相似性を認識する必要があった．さらに専用 Editor の画面を用いて直接に比較を行う事も困難であったため，一貫相似性の検証が困難であった．しかし，単言語方式 OOJ では一つの記述言語のみを用いるため形式的な分析段階記述からプログラミング言語までの一貫相似性の検証を行う事が容易である.

3.　単言語方式 OOJ の設計

三言語を一つの言語に統合するために三言語方式 OOJ の実証データを検証し，それを元に単言語方式 OOJ の設計方針を導き出し，単言語方式 OOJ の言語仕様の設計を本章では行う.

三言語方式 OOJ を用いた一貫記述例の実装段階の記述と Java プログラムを**図 3**，**図 4**，**図 5**，**図 6** に挙げた．対象世界は『一次元衝撃波管内の流れ』であり，1 次元の世界の中で衝撃波が伝搬して行く様子をシミュレーションするものである．記述例は『一次元

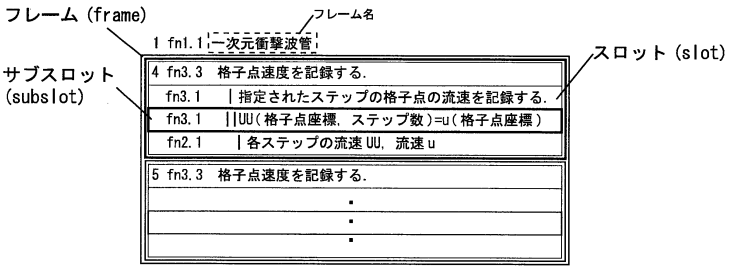

図 2 OOSF のイメージ図

Fig.2 An Image for OOSF

番号	FN	NJ	相互関係
1	fn3.3	駆動開始する	≪mp≪2:駆動シナリオ[1-4]
2	fn3.3	| (反復)任意のステップ数回	
3	fn3.2	| 衝撃波が伝播し流速が変化する	≫[3]
4	fn3.2	| ステップ数の格子点速度を取得する	≫[4]

図 3 分析段階の一次元衝撃波管の記述例

Fig.3 OONJ description example for Shock Tube flow

衝撃波管』フレームのメソッドの一部を示している. **図 3**, **図 4**, **図 5** は, 三言語方式 OOJ で実装された専用エディタを用いて作成している.

単言語方式 OOJ の設計方針を定めるために, 三言語方式 OOJ の各記述の分析を行うと, 対象世界全体からサブスロットまでの集約階層構造を表現する OOSF と, 更にその内側に表現する日本語単文の二つの層に分けて考える. そこで, 単言語方式 OOJ の言語仕様を設計するために OOSF と日本語単文について記述を個々に述べて行く.

3.1 OOSF について

まず, OOSF とは要素の離散化, 離散化した要素に対する構造化, 要素の明示形式性を実現するための記述法である.[1] **図 2** に OOSF のイメージ図を示す. OOSF は「フレーム」,「スロット」,「サブスロット」という三つの離散化した要素から構成されており, 各要素間の関係は集約階層を成している.

三言語方式 OOJ から得られた OOSF に関する資料を整理したものを**表 2**, **表 3** に示す. 表 2 は, 三言語方式 OOJ の分析段階の OONJ, 設計段階の ODDJ, 実装段階の OPDJ 間の対応関係を示しており, 表 3 に実装段階記述とプログラミング言語 (現在は, Java のみに対応している) との対応表を示す. 表 2 の設計段階の右端に表 3 の実装段階があると見ていただきたい. この表 2, 表 3 から分析段階と実装段階の記述が一貫していることが分かる.

また, 表 3 を簡潔に纏めると, フレームは Java の class, スロットは Java のメソッド, サブスロットは Java の宣言文などに対応している. つまり, OOSF はプログラムの構造を模す事が可能で, かつ実際に模しているという事である. これは, OOSF が構成要素の離散化とその構成要素が集約階層構造を成す構成をとっているためであると考えられる. 以上から, OOSF については分析段階からプログラミング言語まで一貫したプログラム構造を表現していると言える.

また, 分析段階から実装段階へは, 変換する際には設計段階と実装段階の記述言語で規定された「計算機で駆動するために必要なデータ」を DU が付与する必要がある. しかし, 計算機で駆動するために OOSF を構成要素の詳細化と判断し, 実装段階の記述言語とプログラミング言語間の関係は同等内容の別表現であるため, 内容の変化は生じない. つまり, OOSF は分析段階からプログラミング段階までに関して各段階間での相似性を持っていると判断する事ができる. そこで, 単言語方式 OOJ のプログラムの構造は, 三言語方式 OOJ の分析段階のプログラムの構造を用いる事とする.

3.2 日本語単文について

上述の様に, OOSF を用いると対象世界 (プログラム) 全体から 1 行程度の長さまでの規模の記述はプログラム構造化されてしまうことが分かるが, 宣言文とか「日本語での一文」規模の記述の実体に関するプログラム化は OOSF では扱えない. 特に OOJ で扱う日

番号	FN	NJ	データ型	アクセス制約	相互関係
1	dfn3.3	駆動開始する	void	共有	
-1	dfn 2.8	和名「反復回数」英字名「J」	整数型	私有	
-2	dfn 3.1	反復回数=0			
-3	dfn 3.1	while (反復回数<ステップ数)			
-4	dfn 3.2.1	衝撃波が伝播する > 一次元衝撃波管			
-5	dfn 3.2.1	格子点速度を記録する > 一次元衝撃波管			
-6	dfn 2.2	和名「反復回数」英字名「J」	整数型		
-7	dfn 3.1	反復回数=反復回数+1			

図 4 設計段階の一次元衝撃波管の記述例

Fig.4 ODDJ description example for Shock Tube flow

番号	PFN	OPD記述	アクセス制約	データ型	相互関係
1	pfn3.3	駆動開始する	public	void	
2	pfn2.8	反復回数=0	private	int	
3	pfn3.1	while(反復回数<ステップ数)			
4	pfn3.2	衝撃波が伝播する()			
5	pfn3.2	格子点速度を記録する(反復回数)			
6	pfn3.1	反復回数=反復回数+1			

図 5 実装段階の一次元衝撃波管の記述例

Fig.5 OPDJ description example for Shock Tube flow

```
//OPDJ/Java プログラム
public class Cls1{ // 一次元衝撃波管 一次元衝撃波管
  // 処理スロット
   public void mthd1() { // 駆動開始する
     int tempVar0 ; // 反復回数
     tempVar0=0;
     while(tempVar0<Cls4.instance.glb1) {

       this.mthd2(); // 一次元衝撃波管. 衝撃波が伝播する
       this.mthd3(tempVar0); // 一次元衝撃波管. 格子点速度を記録する
       tempVar0=tempVar0+1;
     }
     System.out.println(" X 座標, 2STEP, 4STEP");
     while(tempVar0<Cls4.instance.glb1){
       System.out.println(tempVar0+" "
      +Cls4.instance.glb4[j][2]+" "
      +Cls4.instance.glb4[j][4]);
       tempVar0=tempVar0+1;
     }
   } // ***** メソッド終了
} // ***** クラス終了
```

図 6 Java の一次元衝撃波管の記述例

Fig.6 Java description (program) example for Shock Tube flow

本語はプログラムの構造とは異なり，日本語の文法が支配する構造である．したがって，我々は OOSF 以上と NJ 文以下の二種類に分けて別途扱う方針とした．

まず，我々は DU が扱う NJ 文を DU の利用に支障の無い範囲で制約を加えることにする．

(1) 文法的に正しい NJ 文を書くことは DU の責任とする．

(2)NJ 文は必ず単文 (以降，NJ 単文) とする．

(3) 複文重文等は DU 責任で全て単文に変換する．

(4)NJ 単文を更に英語の五文型に限定する．これを以降，**NJ 五文型**と呼ぶ．

文学作品等を扱う訳ではない DU には，これらの制約はプログラムを得るための支障にはならないと考えられる．この NJ 五文型はオブジェクトにおいては更に二種類に分けられる．それは，オブジェクトの内部振舞いとオブジェクト間の振舞い (相互関係) を表すメッセージパッシング (以降，mp と略) である．これらには以下の様な分析が有効である．

(1) 第一，第二文型 (いわゆる，S+V 型と S+V+C 型) が表す内部振舞いは，最終的には関数定義に持ち込む．システム側は記述作業を支援するが，記述責任は DU にある．DU は DU 責任で振舞いの記述を詳細化して，計算機が処理可能な記述に自力で変換する．

(2) 第三，第四，第五文型 (いわゆる，S+V+O 型，

表 2 三言語方式 OOJ の各段階の要素種類の対応表

OOSF	OONJ	ODDJ
フレーム	フレーム 対象世界全体共有フレーム シナリオ 初期状況記述フレーム 境界状況記述フレーム	フレーム 対象世界全体共有フレーム シナリオ 初期状況記述フレーム 境界状況記述フレーム
スロット	スロット	スロット 変数スロット 初期化スロット main スロット 初期状況記述スロット 境界状況記述スロット
サブスロット	サブスロット	メソッド呼び出し 代入文 脱出文 応答文 空文 コメント
付置属性	付置属性	仮引数 実引数 スロット内変数記述
サブスロット	注釈 行内注釈	コメント コメント
フレーム構造記述子	フレーム間構造記述子	フレーム間構造記述子
サブスロット サブスロット サブスロット	二分岐文 多分岐文 反復文	if 文 switch 文 while 文

表 3 三言語方式 OOJ の実装段階と Java との対応表

OPDJ	Java
フレーム 対象世界全体共有フレーム シナリオ 初期状況記述フレーム 境界状況記述フレーム	public class... public class... public class Main... public static void frame-init-slot()... public static void frame-boundary-slot()...
スロット 変数スロット 初期化スロット main スロット 初期状況記述スロット 境界状況記述スロット	access type メソッド名 (...) access type メソッド名 (...) public クラス名 (...) public static void main(String[] args)... - -
メソッド呼び出し 代入文 脱出文 応答文 空文 コメント	メソッド名 (...), instance. メソッド名 (...) 代入文 break return() - // ~ //
仮引数 実引数 スロット内変数記述	access type メソッド名 (仮引数) メソッド (実引数) access type 変数名, access type id = 値
フレーム間構造記述子	extends

if 文
 	
switch 文
while 文
	if(condition)...
	if(condition)... else...
	switch(condition)...
	while(condition)...

S+V+O+O 型，S+V+O+C 型) は目的語を持つのでメソッド呼び出しに変換されるが，メッセージの送り先は一意に決まっており，また付置して送るための属性はメソッド呼び出しで順に引数として扱われるので，この文型は自動変換される/可能である．

したがって，システム側としては，第 (1) 項は記述支援だけで良く，その形式にまで記述されればそれで良い，第 (2) 項は文型通りに書かれていれば，完全自動変換される．

そこで，独自に NJ 単文を自動変換するための機構として日本語単文変換機構 (以降　Japanese Transformation Mechanism:JTM と略) を設計・実装した[6]．JTM の特性を考慮すれば，OOJ の設計としては「日本語での一文」規模の記述は JTM によりプログラム構造に変換されるとして良い．また JTM の変換規則により一貫相似性が判断できることになる．

3.3 単言語方式 OOJ の言語仕様

三言語方式 OOJ の言語仕様は，3.1 節で述べた「OOSF」と「計算機で駆動するために必要なデータの規定」から構成される．三言語方式 OOJ を統合化し単言語方式 OOJ の設計を行うため，言語仕様の構成は三言語方式 OOJ とほぼ同様になると判断し，三言語方式 OOJ の言語仕様の各構成要素毎に単言語方式 OOJ の言語仕様を設計していく．

OOSF については，3.1 節で述べたように分析段階の記述言語 OONJ の OOSF の規則を用いることにする．同じく 3.1 節で述べた計算機で駆動するために必要なデータの付与は，三言語方式 OOJ では，設計段階の記述言語で行われているため単言語方式 OOJ の言語仕様は設計段階の記述言語 ODDJ の規則を用いる．

以上の設計方針で具体的な設計作業として以下のように行った．

(1)DU の情報提供無しに設計段階，実装段階への形式的に変換可能である OOSF の構成要素は，DU 側から見ると自動変換が可能である要素に当たる．したがって，最初の記述を行う分析段階の記述言語を単言語方式 OOJ として充てる．現在の例ではメッセージパッシングがそれに当たる．

(2) 設計段階で DU から計算機が駆動するための情報提供が必要で，実装段階で不要ならば，設計段階の記述言語を単言語方式 OOJ の記述規則として充てる．

(3) 同様に，実装段階で情報提供が必要であれば，実装段階の記述言語を用いる．しかし，現段階で三言語方式 OOJ の資料から該当する要素が無かった．無かった理由は実装段階のライブラリの数が少なかったためだと考えられる．

以上の設計方針で単言語方式 OOJ の言語仕様の設計を行った．設計結果の記述規則は紙幅の関係上，割愛した．代わりに当研究グループの WebSite[7] に掲載してある．

4. 設計に基づく記述例

第 3 章で述べた方針で設計した単言語方式 OOJ の言語仕様を元に作成した記述例の実装段階を**図 7**，**図 8**，**図 9** に挙げた．対象世界は三言語方式 OOJ の記述例の『一次元衝撃波管内の流れ』[2] の『一次元衝撃波管』フレームのメソッドの一部を示している．

図 7，**図 8**，**図 9** と**図 3**，**図 4**，**図 5** の違いは，単言語方式 OOJ ではサブスロットの表記が分析から実装まで一貫しているのに対して三言語方式では各段階

```
1  nfn1.1  一次元衝撃波管
2    nfn3.3  駆動開始する
 -1 nfn3.3  |(反復)任意のステップ数回
 -2 nfn3.2  | 衝撃波が伝播し流速が変化する. ≫[3]
 -3 nfn3.2  | ステップ数の格子点速度を取得する. ≫[4]
 -4 nfn3.1  |速度の遷移を出力する.
 -5 nfn2.1  | 格子点の流速
```

図 7　分析段階の一次元衝撃波管の記述例

Fig.7 OOJ description example for Shock Tube flow

```
1  dfn1.1  一次元衝撃波管
2    dfn3.3    駆動開始する                          共有    void
 -1 dfn2.8    和名「反復回数」英字名「J」            私有    整数型
 -2 dfn3.1    反復回数 = 0
 -3 dfn3.1    |while (反復回数<ステップ数)
 -4 dfn3.2    | 衝撃波が伝播し流速が変化する. ≫[3]
 -5 dfn3.2    | ステップ数の格子点速度を取得する. ≫[4]
 -6 dfn2.2    和名「反復回数」英字名「J」                    整数型
 -7 dfn3.1    反復回数 = 反復回数 + 1
```

図 8　設計段階の一次元衝撃波管の記述例

Fig.8 OOJ description example for Shock Tube flow

```
1  pfn1.1  一次元衝撃波管
2    pfn3.3    駆動開始する                          共有    void
 -1 pfn3.1    反復回数 = 0                          私有    整数型
 -2 pfn3.1    |while (反復回数<ステップ数)
 -3 pfn3.2    | 衝撃波が伝播し流速が変化する. ≫[3]
 -4 pfn3.2    | ステップ数の格子点速度を取得する. ≫[4]
 -5 pfn2.2      和名「反復回数」英字名「J」
 -6 pfn3.1    反復回数 = 反復回数 + 1
```

図 9　実装段階の一次元衝撃波管の記述例

Fig.9 OOJ description example for Shock Tube flow

の記述言語毎に変換されている点である．具体的な部分として，**「衝撃波が伝播し流速が変化する.」サブスロット**である．この要素は，設計段階でメソッド呼び出しに当たるが，分析段階で変換するために必要な情報を DU が与えているため，分析段階から実装段階まで一貫した表現形式を取っている．つまり，三言語方式 OOJ 記述と単言語方式 OOJ 記述の違いは，表現形式のみであると推定され記述内容の違いは存在しないと言って良い．したがって Java プログラムへのトランスレータを用いる事で同様な結果を得られると判断できる．

以上の事から，三言語方式 OOJ を統合化した単言語方式 OOJ を設計できた事が少なくとも記述例を通して検証できた．

5. 評価と考察

5.1 単言語方式 OOJ の評価

第 4 章で述べたように，三言語方式 OOJ と単言語方式 OOJ の違いは，一貫した表記の有無についてのみであった．一貫相似性の形式的な検証を容易に行うには，段階毎に変化した箇所を明示的に示す事が重要である．そして，段階毎に変化する必要が無い箇所は表記を変化させない必要がある．三言語方式 OOJ では，各段階毎に記述言語が異なっていたため変化する

必要が無い箇所も表記が変化している状態であった．そのため，三言語方式 OOJ の各段階の記述言語で重複していると判断した箇所は，変化する必要が無い箇所だが表記が異なる箇所だと考えられる．すなわち，単言語方式 OOJ において一貫した表記が存在している事は三言語方式 OOJ の各記述言語で重複した箇所の統合を行われているからだと言える．

つまり．三言語方式 OOJ の記述言語間で重複した箇所を統合することによって，単言語方式 OOJ の設計を行い，一貫相似性を形式的に検証が容易になったことが分かる．

5.2 単言語方式 OOJ の設計についての考察

三言語方式 OOJ が一貫相似性を持った記述例の作成が可能になったため三言語方式 OOJ を統合化した単言語方式 OOJ を設計できた．さらに，実際の記述例をいくつか通す事によって当初から考えられていた単一仕様言語の言語仕様を設計するための参考資料を収集する事ができた．その資料は，対象世界の要素種類である．

これによって，対象世界の要素とプログラムの構成要素との対応関係の一意的な特定が可能になる事で，分析段階の構成要素と実装段階の構成要素との相似性を示す事が可能となると考えられる．さらに，当初，単一仕様言語の作成が困難だった理由の一つは，対象世界の構成要素の種類と計算機処理の命令の種類との対応関係についての実証資料が全くなかったからだと言えよう．

6. 結論と今後の課題

本論文で得られた結論を以下に示す．

(1) 一貫相似性を考慮して言語仕様を設計するには，OOSF と日本語単文を分けて考える必要がある．

(2)OOSF はプログラムの構造を模す事が可能である事が実証された．

(3) 日本語単文は JTM[6] によりそれ単独でプログラムに変換され，日本語単文とそれに対応するプログラム間には相似性が成立する．

(4) 三言語方式 OOJ を統合化することで単言語方式 OOJ の言語仕様を設計することができた．

(5) 三言語方式 OOJ を統合化した単言語方式 OOJ を設計し，一貫記述例を作成する事ができた．

(6) 単一仕様言語の OOJ を設計するために必要な資料が対象世界の構成要素の種類である事が分かった．

今後の課題として以下が挙げられる．

1. 本論文で得た参考資料を元にした単一仕様言語 OOJ の設計を行う．

2. 単一仕様言語 OOJ の記述を作成するための開発環境の実装を行う．

参 考 文 献

1) 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).

2) 峯村吉泰，流体・熱流動の数値シミュレーション 第 3 章，森北出版株式会社，ISBN4-627-91751-1.

3) 畠山正行，高度なシミュレーションのためのオブジェクトベース一貫モデリング過程論とその駆動支援環境，第 1 回情報処理学会 MPS 研究会研究報告，，(1995 年 5 月 18 日).

4) 畠山正行，一貫モデリング過程論に基づく物理世界のオブジェクトモデル，第 20 回情報処理学会 MPS 研究会研究報告，，(1998 年 7 月 24 日).

5) 大木幹生，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，情報処理学会第 163 回研究会報告，2009-SE-163.

6) 片野克紀，オブジェクト指向設計記述言語 ODDJ の設計と実装，平成 20 年度修士論文，茨城大学大学院理工学研究科 (2008).

7) 畠山研究室，http://gaea.cis.ibaraki.ac.jp/