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

2008－SE－159 (9)
2008／3／17

オブジェクト指向一貫相似性記述言語OOJの設計と検証

池田陽祐 †1 大木幹生 †1 片野克紀 †2
加藤木和夫 †3 涌井智寛 †4 畠山正行†1

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記述言語が必要であることが分かった．それ故に記述言語系を定義・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た．各記述言語の記述要素毎のの対応関係とその対応要素の相似性を検証し，それらを全要素 (構造要素も含む) に行って総合することで，分析記述され対象世界と再現シミュレーションの相似性の高さを (客観的にも) 検証可能な方法として提案した．さらに，三つの記述言語を一つの記述言語に統合化した記述言語OOJの設計方針を導出した．今後は三つの記述言語の仕様を拡張して完成度を高めると共に，それを基盤にしたOOJの詳細設計と実装を行う．

A Design of Object-oriented, Integarally Consistent and Similar Description Language OOJ and Its Validations

YOUSUKE IKEDA ,†1 MIKIO OOKI ,†1 KATSUNORI KATANO ,†2
KAZUO KATOUGI ,†3 TOMOHIRO WAKUI ,†4
and MASAYUKI HATAKEYAMA†1

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 experts of some special science/technology domains except the information domain. We can conclude that the OO description language at each stage of the analysis/design/implementation are needed to make verifications of the similarity. A description support editor has been developed to handle for the definitions, designs, and for the implementations in the computer. We have proposed a actual method to verify the similarity between the analyzed target world and its reemergent computer simulations. We have developed the method by verifying the correspondence and the similarity of each description element of each description language, and sum up them all the elements (including the structure elements). We have derived some general rules for the description language OOJ that have been integrated from these three description languages. In the future works, we aims at extending the specifications of the three description languages, and also at tuning up them. We will develop the extended design of the OOJ and its implementations.

1. はじめに

本研究のそもそもの発想は，著者の1人が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?」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

†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
†3 茨城県立産業技術短期大学校
Ibaraki Prefectural Industrial Technology Junior College
†4 茨城大学大学院理工学研究科情報・システム科学専攻
Graduate School of Information and System Science, Ibaraki University

いう抽象的には同一のモデリング/☆プログラミングに対して一貫して同じパラダイムを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．

しかし，この方法論には大きな欠点が存在した．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 OO 記述言語である．ただし，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 (Natural Japanese, NJ) をベースとした記述言語である必要があった．

そのために，OO に基づく構造化記述のために OOSF が提案され，OO 記述の最小単位を自然日本語 (以降，NJ と略) で記述すると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り，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ないので自動生成のシステムも必須であるという条件も加わる．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 (記述言語も含めた) 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 を主用する 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．それらが一昨年秋の OONJ，ODDJ，OPDJ，そして OOJ に関わる発表であった．これ等の発表においては，三つの言語における連携関係は未だ十分ではなく，1000

☆ 本論文では，"/" はその前後の語句の or を指す．

＊ **表 1** 想定ドメインユーザ (DU，特定分野の専門家) の特徴
Table 1 Features of Domain Users: DU

1. 応用ドメインの専門家で，計算機は利用だけ
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．

2. 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．

3. 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．

4. 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．

5. 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．

6. プログラムや計算機に関わる時間的/心理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．

7. 自身の専門に関しては多様で十分な表現力を持つ
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．

8. OO は概念として一通りは理解し使えると仮定
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. Ordinary designates the Japanese.

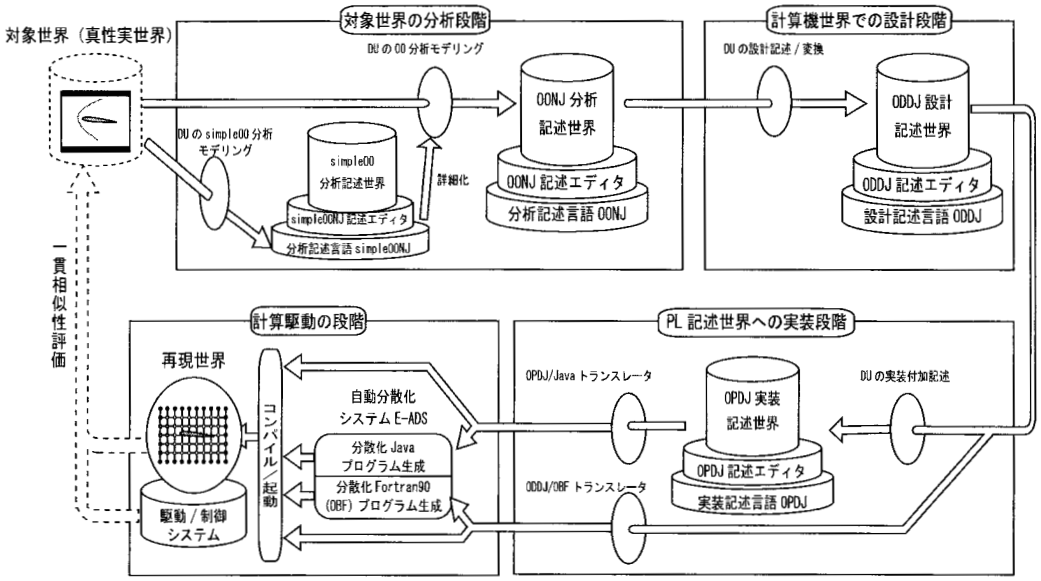

図 1　オブジェクト指向 一貫相似性モデリング/記述/変換過程の方法

Fig.1 A Method of Object-oriented, Integarally Consistent and Similar Processes

行近い記述 (水野大気循環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) により Jave プログラムが自動生成されただけであった．

その後，各記述言語の仕様の不具合や機能向上等の改訂が各々独立して行われてしまったという事情が生じたために，一貫した記述/プログラミングが得られない状況にあった．しかしその後，特に (本発表の前に行った) 設計段階の記述言語である ODDJ の仕様を開発行程でいう上流に位置する分析段階の OONJ と，下流に位置する実装段階の OPDJ との調整を考慮した上での改訂を行うことにより，OONJ・ODDJ・OPDJ の三記述言語に基づく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「一貫相似性を実現した記述」の準備が整ってきた．これと並行して三言語を順次段階的に用いる段階から OOJ という単一仕様の記述言語への移行段階への見通しも立ってきた．

そこで本論文においては，上流の OONJ と下流の OPDJ における仕様の改訂を踏まえた上で三つの記述言語 OONJ・ODDJ・OPDJ で一貫した記述を実現するシステムと OOJ の構想とを対照させ

(1) どの程度に一貫相似性を実現できる仕様になっているのか

(2)OOJ の仕様に近い記述言語が実現可能か

(3) 本来の OOJ の基盤として使えるのか

についての見解が纏まったのでその報告を行う．

2. OOJ の定義と変遷

第 1 章で述べた様に，相似性の高い (したがって，信頼性の高い) 計算機内シミュレーションを実現するために一貫相似性モデリング過程が考案され，そのモデリング/プログラミング過程を実現するために，OO 一貫相似性記述言語 OOJ の構築が計画された．本章では OOJ 構築に至るまでの経緯を述べ，それらから OOJ の実現すべき/満たすべき条件等を導出する．また，それにより OOJ の設計要件や特性を検討する．

2.1 一貫相似性モデリング/プログラミング過程と，三つの記述言語あるいは OOJ との関係

本節では，OOJ 設計の狙いであり，その根幹でもある一貫相似性を持たせたモデリングとプログラミングの過程について述べる．

図 1 に一貫相似性モデリング/プログラミング過程を模式図で示した．DU は図の左上の対象世界を定め，OO 分析モデリングを行って分析段階のモデルを作る．その記述に用いられる言語が分析段階の記述言語 OONJ である．同様にこの記述を元にして計算機世界内に移すために図右上に移って設計モデリング

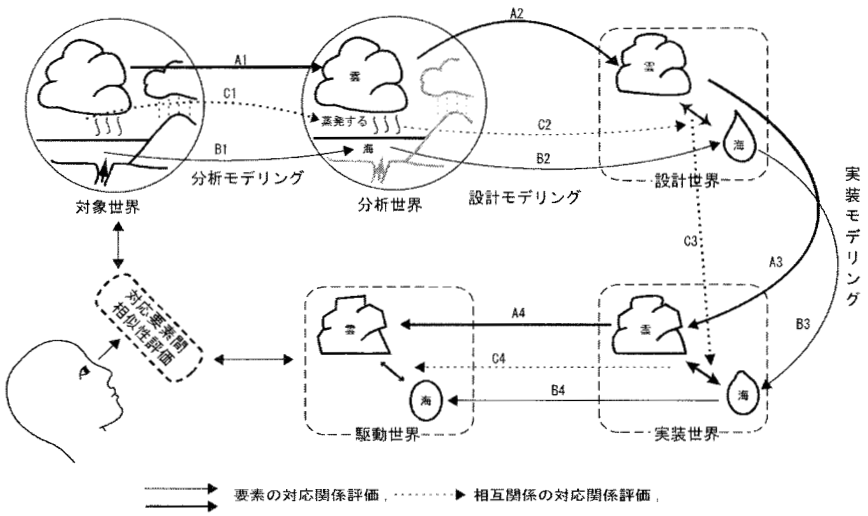

図 2　OO 一貫モデリング過程における一貫相似性の検証の方法

表 2　オブジェクト指向モデリングを前提とした場合の一貫相似性の概念の定義

相似性の有無 変換前後の対応する記述 DMU(脚注 1) の組が対応する要素間で「縦の相互関連 (脚注 2)」を同定できる事実を指す.
相似性の高さ (相似精度) 縦の相互関連が同定された記述 DMU 間の記述が指す内容実体の差が無いか，小さければ相似性が高いと判断して良い．この判断は DU に委任すべき (専権的な) 事項である.
比較の対象とすべき「構成 DMU」 全ての DMU 要素について比較対照する．具体的には OONJ の (表 3 の)DMU 種類表，(表 4 の) 汎化階層構成 DMU 種類，(表 5 の) 集約階層構成 DMU 種類，の DMU を指す．詳細には (表 6 以降の) 同 OONJ の多くの構造記述規則表に現れる記述 DMU を指す.
一貫相似性の有無 任意の複数の段階，したがって分析・設計・実装の全ての段階のモデル記述において，対応すべき記述構成 DMU 間の全ての縦の相互関連を同定できること.
一貫相似性の高さ (一貫相似精度) 縦の相互関連が同定された任意の全ての段階の記述構成 DMU 間の記述内容の差の小ささ. ドメインユーザの (専権的な) 判断事項.
上記の項目だけで OO に基づく一貫相似性が判断できる理由の簡潔な説明 本表で扱う要素の中には，必ず相互関係も全く他の要素と同じように「要素」として含まれている．この相互関係は，静的な相互関係である相互関連 (用語) と動的な相互関係である相互作用 (用語) が含まれる. したがって静的な構造を指し示す相互関連と，相互作用を指す動的構造がその定義に含まれる．故に，この相互関係要素をその最小要素に至るまで分解して比較対照した結果，上記の事項を満たすならば，最小の DMU から対象世界全体に至る静的/動的両方の構造においても相似性が成り立っていると言える.

(脚注 1) : DMU　= Discrete Modeling Unit. 即ち離散 (化) モデリング単位を指す．ただし，独立したオブジェクトだけではなく，その内部要素等も定義に含む.

(脚注 2) : 縦の相互関連とは，対象世界の DMU，分析世界 (OONJ 記述) の DMU，設計段階 (ODDJ 記述) の計算機世界の DMU，実装段階 (OPDJ 記述) の各要素．(可能であれば，)OOPL のプログラムの要素，の四つの世界間の対応する要素の相互関連を指すための用語として定義した.

(設計への変換) が行われ，設計段階の記述言語 ODDJ で記述される．同様な過程を経て実装に関する記述が付加され同じく記述言語である OPDJ で記述が行われる．この OPDJ 記述は各プログラミング言語 (以降，PL と略す) のプログラムに変換するためのトランスレータを経て，例えば Java のプログラムへと変換・コンパイル・実行される．

各モデリング/(広義および狭義の) プログラミング☆段階のモデルと，それらのモデルを記述するための記述言語 OONJ，ODDJ，OPDJ，を対応させることにより，一貫相似性のモデリング/プログラミング過程の成立を検証することが出来るようになる．このことを本論文で検証する．

2.2 三つの記述言語を用いた一貫相似性の検証方法

OO に基づく対象世界のモデリング過程の四つの世界の関係を**図 2** に示す．図の左上の対象世界のモデリングから下段の駆動世界までの過程において，分析モデリング，設計モデリング，実装モデリングと進んでゆく．その各モデリング作業の各離散要素についてのモデリング元とモデリング先の関係を「縦の相互関連」と呼ぶ．図中の A1～A4，B1～B4，C1～C4 がそれである．これ等は各世界でモデリングされた離散要素 (相互関係も含む) の対応関係も示す．

個々の要素の対応関係 (例えば，A1，A2，A3，A4) が存在すれば，元の世界とモデリング先の世界の対応要素の存在が立証されれば，その要素についてモデリングの妥当性や相似性を DU が検証するのは簡単であり，客観的に可能である．この作業を A1 から C4 に至る，即ち全ての縦の相互関連を確認し，その対応要素間の相似性を個々に評価した上で，それらを総合すれば，対象世界と再現シミュレーションとして計算される駆動世界 (図 1 参照) との相似性が (客観的にも) 検証できる．これが一貫相似性の検証方法である．この方法は OO パラダイムを採用することで実用的に可能かつ客観的 (他の DU も同様に検証可能という意味で) な検証方法となり得た．

図 1 の様な方法で一貫相似性を検証するためには，各世界間において OO 要素 (勿論その OO 要素にはその構造記述要素，つまり相互関係 (＝静的構造と動的構造 (相互作用) を含める) の対応する要素を同定して対応関係を明らかにすると共に，その対応関係のある要素間の相似性を評価する，ということが必要である．この要素間の対応関係，したがって相似性をも検証する道具立てとして分析/設計/実装の各世界における対象世界の記述を行うために，各段階の記述言語が必須なのである．かつ各要素を個々に取り出して対応関係と相似性を比較評価するためには離散的に扱われる OO パラダイムと OO 記述言語が最適である．

分析/設計/実装の各段階の各世界の要素間の対応性の確定と相似性の検証を行うには，各世界を表現する記述言語の構造記述規則/文法を用いて行えばよい．勿論，必要に応じて記述例を挿入する方がベターである．

以上から本章の結論として，一貫相似性を実現するためには，三つの記述言語 OONJ，ODDJ，OPDJ が必須であり，かつ三つの記述言語の記述力や記述の詳細さ等が，離散要素の対応性や各要素間の相似性を検証できる必要があることが分かった．そこで，次章では OONJ，ODDJ，OPDJ，Java の各記述言語と OOPL に対してこれ等を実証してゆく．以上で目的を実現する方法と道具立てが揃った．

3. 三つの記述言語とその比較検討

3.1 三つの記述言語の比較検討の必要性と比較方法

三つの記述言語は，OONJ が実世界の NJ 記述を，ODDJ が計算機世界一般の記述を，OPDJ が特定 PL 特有の記述を，それぞれ扱うと定義されている．もちろん一つの対象世界を OONJ で記述し，その情報を ODDJ を用いて，記述世界の内容の同一性 (相似性) を保ちつつ別表現に変換して計算機世界内部での記述にし，その情報に OPDJ を利用予定の OOPL(例えば，Java) に特有の記述 (例えば，Java の入出力の詳細な statements) に変換または追加してゆく，という作業である．したがって，ODDJ と OPDJ は上流からもたらされた情報をそのまま，または可能な限り「同一内容の別表現」に変換する規則 (文法) と機能を持つ必要がある．しかし，計算機世界はその構造も駆動する法則や仕組みも異なる世界なのであるから，それは保証の限りではない．そこで，改訂された三つの言語の対応関係と相似性について各言語のルール (つまり文法あるいは構造記述規則) 間での比較対照の検証が必要となってくる．

また，各記述言語の文法 (構造記述規則) は簡単ではあるが，容易に直ちに理解し暗記・適用出来るとは限らない．そこで文法に沿った記述の支援を行うエディタや NJ を主体とした記述を下流の記述言語の形式に適合した記述または PL のプログラムに変換するためのト

☆ 広義のプログラミングとは，言語を使って作られる作業の中，計算処理に必要な情報を与える全ての記述作業を指す用語である．

ランスレータといった記述支援環境が必須となってくる．したがって，計算機内部においてOONJ，ODDJ，OPDJの文法(構造記述規則)を扱う必要がある．それは，各記述言語の文法を計算機世界内部の記述法/言語を規定する必要がある事を意味する．我々はそれについてはXMLを用い，文法を(現状では)DTDを用いて規定している．したがって，OONJとODDJの構造記述規則(文法)とこのDTDとがちょうど写像関係にあるか否かをも比較検証しておく必要もある．

以上から三つの記述言語の文法と二つのDTD(OPDJはDTDを持たない設計にしてある)の間の写像関係や対応関係と相似性に関する評価を行う必要がある．それらの全体の関係を図示したのが，**図3**である．四角形が表形式に納めた文法であり，楕円形が文法間または文法とDTDの間の比較表である．それぞれ，簡単のために**表3**の様に省略符号が定められている．各表の詳細は通計で30ページにも上るので全て省略し，当研究室のWebSiteに公開してある．本講演論文では引用できないので，言及しないが，当然ながら各要素の記述規則の対応するか否かや相似性の判断も行ったことを含めた判断を行っている．

表3 OONJ，ODDJ，OPDJ，Javaの対応一覧表の表記法

対応表記	意味
SDR	Structured Description Rules の略であり構造記述規則のことである．
N-R 表	OONJ の構造記述規則
N-D 表	OONJ の DTD 規則
D-R 表	ODDJ の構造記述規則
D-D 表	ODDJ の DTD 規則
P-R 表	OPDJ の構文規則 (文法)
J-R 表	Java の文法
N-RD 表	OONJ の規則と DTD の比較表
ND-R 表	OONJ と ODDJ の記述規則の比較表
ND-D 表	OONJ と ODDJ の DTD の比較表
DP-R 表	ODDJ と OPDJ の記述規則の比較表
PJ-R 表	OPDJ と Java の記述規則の比較表

3.2 三つの記述言語とJavaプログラムの比較検討

代わりに各記述規則に出てくるOO離散要素名を纏めてリストとして挙げ，それらが各記述言語間でどの様に扱われるかを分類して**図4**に示した．

分類1に属する要素はその記述規則と共に，最後の段階のPLのプログラムにまでそのまま達する．分類2に属する要素は変換はされるが，PLプログラムにまで達する．分類3に属する要素は離散集合を行う要素であり，別の要素の一部として再生したり，複数の要素が集まって，次の段階の新規要素を形成する要素群である．分類4と5に属する要素はODDJとOPDJでの新規出現要素である．

図3から各記述要素の記述言語間での変換/出現/消滅等の扱い，つまり，対応関係と相似性保持がどの様になるかの概略を知ることができる．この多様な相互関連を十分考慮に入れた上で各要素が分析段階から実装段階に至るまでにどのように扱われるかを追跡して，その上で各要素毎にその要素が出現する当初段階における記述規則/文法を設計すればよいことになる．

例えば，カテゴリー2におけるメッセージパッシング(mp)について分析からJavaプログラムまでの変換例を**図5**に示す．OONJの記述規則と記述例の組を最上段に，ODDJについての同じ記述の組をその下段に，OPDJについて同様に下から二段目に，そしてJavaについて同様に最下段に示す．これ等四つの規則と記述例の組が，OOJの特性である一貫相似性の記述であること，すなわち，同一内容の別表現を表していることは容易に分かる．

3.3 三つの記述言語を通した変換の流れの記述例

一貫記述例の一つの記述例を**図6**，**図7**，**図8**，**図9**に挙げた．対象世界は『一次元衝撃波管内流れ』であり，1次元の世界の中で衝撃波が伝搬してゆく様子をシミュレーションするものである．記述例は『衝撃波管』フレームのメソッドの一部を示している．図7は，本研究で作成したODDJエディタを用いて作成し，ODDJのDTDを基にしたXML形式のODDJ文書をXSLTを用いて構造記述規則を基にした記述へ変換したODDJ記述である．OONJにおける『内部振舞いとしての衝撃波管伝播を開始する』が，ODDJ，OPDJでは『Lax _ Wendroff計算』，Javaでは『method5』に適切に変換されている．また，メソッドの内容もOONJ，ODDJ，OPDJ，Javaにおいて適切に変換される．これらのことより，本研究で設計されたODDJはOONJ，OPDJと連携をとれていると考えられる．

4. 三記述言語を統合したOOJの設計方針

図5の記述間の関係が「オブジェクト指向一貫相似性記述」になっていることは前章で提示した．本章ではこの例と記述規則の関係から以下のことが導出できることを考察する．即ち，OONJとODDJの記述規則とOPDJの対応する文法を比較し，各記述規則毎に検討すれば良い．

(1) 下流に位置するODDJ構造記述規則とOPDJ

文法がDUの情報提供無しに，すなわち形式的に変換可能である場合は，DU側から見れば自動変換が可能である場合に当たる．したがって，最初の記述を行う記述言語の記述規則が基本的にOOJの記述規則となる．今の例ではmpのOONJの記述規則がそれに当たる．途中の変換処理はトランスレータが行う．

(2)ODDJの段階でDUの情報提供が必要で，OPDJの段階で不要ならば，ODDJの記述規則をOOJの記述規則として充て，DU向けの記述規則としてはOONJに近づけた記述形式を新たに定義するなど，可能な限りDUの特性に沿った形の(即ちスマートな)処置を取って，この規則を決める．

(3) 同様にOPDJの段階で情報提供が必要であれば，三つの定義方法が考えられる．

(i)OPDJの当該規則をOONJ風に表現した文法規則を設計する．

(ii)OPDJの当該規則をカプセル化(抽象化)し，それをライブラリーや関数等の形で定義し，呼び出して使う形式を取る．この方式はライブラリーの引数等の与え方にも適用する．

(iii) エディタで記述基礎の記述に必要な情報をインタラクティブに入力する．

これら各々をOOJの当該規則として生成すれば良い．

5. 纏めと今後の計画

本論文で得られた結論を以下に示す．

(1) 相似性の高い再現シミュレーションを実現する方法として一貫相似性を実現するために，分析・設計・実装の各段階毎のOO記述言語が必要である．

(2) 各記述言語の構造記述規則/文法の対応関係と相似性を検証することで再現シミュレーションの相似性を保証する方法があることが分かった．

(3) 三つの記述言語を比較対照した結果，不足する仕様要素はあるものの，概ね(1)(2)に沿った仕様であることが検証された．

(4) それに基づいて，三つの言語を一つに統合化するための言語設計方針が提案された．この設計方針の具体化と詳細化は現在進行中である．

今後の課題としては，以下が挙げられる．

1. 各記述言語の仕様を改訂，機能と表現力の向上．
2. エディタやトランスレータの拡張．
3. 一貫した記述例の作成と蓄積を通じて実記述レベルでも対応性と一貫相似性の実証すること．

(参考文献は前発表と同じであることもあり，省略させていただきました．)

```
$処理スロット 計算開始:void(t 実数型) /* Masa */
  if (t>=0)
  {
    call 一次元衝撃波管「マサ」.Lax_Wendroff計算():
  }
  else {
  $戻る 終了:
};

$処理スロット Lax_Wendroff計算 : void( ) /* Masa */
  $変数スロット JMAX : 整数型: /* 格子点数 */
  $変数スロット u3[JMAX] : 実数型: /* 離散点速度(予測値) */
  $変数スロット UN[JMAX] : 実数型: /* 離散点速度(修正値) */
  $変数スロット J : 整数型: /* 離散空間座標パラメータ */
  J=1:
  while(J>=101){
    u3[J] = 0.5*{((1.0 - CN)*u)+((1.0+CN)*u)}:
    J=J+1:
  }:
  while(J>=101){
    UN[J+1] = u - (CN*(u3-u3)):
    J=J+1:
  }:
  call 駆動シナリオ.標準出力(UN[ ]):
  call 一次元衝撃波管「マサ」.継続判定( ):
```

図 8 『衝撃波管』フレームの一部のOPDJ記述例

Fig.8 OPDJ description example for Shock Tube flow

```
public void mthd4(double t) {   // 計算開始
    if(t>=0) {
        mthd5();  //  一次元衝撃波管「マサ」.Lax_Wendroff計算
    }
    else {
    }
}  //  *****メソッド終了
// 処理スロット
public void mthd5() {   // Lax_Wendroff計算
    double u3 [] = new double[20];  // u3
    double UN [] = new double[20];  // UN
    int J ;  // J
    J=0;
    while(J<20) {

            u3[J]=0.5*{((1.0-CN)*u[J+1])+((1.0+CN)*u[J])};              J=J+1;
    }
    ;
    UN[0]=1.0;
    J=1;
    while(J<20) {

            UN[J]=u[J]-(CN*(u3[J]-u3[J-1]));              J=J+1;
    }
    ;
    //u[]=UN[];
        for(int i=0; i < 20; i++ ){
            u[i] = UN[i];
        }
    Cls3.instance.mthd9(u);  //  駆動シナリオ.データ出力
    mthd6();  //  一次元衝撃波管「マサ」.継続判定
}  //  *****メソッド終了
```

図 9 『衝撃波管』フレームの一部のJava記述例

Fig.9 Java description example for Shock Tube flow

```
4       fn3.3   駆動開始する.
 -1     fn3.2     |時間T=0.0+という情報&命令を受け取る.                    ≪ rnp ≪ 6.シナリオ[2-5]
 -2     fn2.2       |時間T
 -3     (注釈:) 対象世界に対する駆動開始命令である.
 -4     fn3.2     |自身の駆動を開始する.                                  » [5]
5       fn3.3   内部振舞いとしての衝撃波伝播を開始する.
 -1     (注釈:) u3は，101個の一次元配列である.
 -2     (注釈:) UNは，101個の一次元配列である.
 -3     (注釈:) 第1波の予測波の伝播を起こす.
 -4     (注釈:) Jは，1から始まる整数型の離散化空間座標パラメータである.
 -5     fn3.1     |(反復)Jが(n+1)になるまで，1ずつ加えて計算を繰り返す.
 -6     fn3.1       |u3(J)=0.5*((1-CN)*u(J+1)+(1.0+CN)*u(J))
 -7     vfn3.1      ‖Lax-Wendroff法第1段階計算
 -8     (注釈:) 第2波の修正子波の伝播を起こす.
 -8     fn3.1     |(反復)Jが1から(n+1)になるまで，1ずつ加えて計算を繰り返す.
 -10    fn3.1       |UN(J+1)=u(J+1)-CN*(u3(J+1)-u3(J))
 -11    vfn3.1      ‖Lax-Wendroff法第2段階計算
```

図 6　『衝撃波管』フレームの一部の OONJ 記述例

Fig.6　OONJ description example for Shock Tube flow

```
2   dfn3.3   計算開始（実数型　t）                                          void      私有
    dfn3.1.1   |  if( t>=0 )
    dfn3.1.1   |  then
    dfn3.2     |  Lax_Wendroff計算                  >> 一次元衝撃波管「マサ」
    dfn3.1.1   |  else
3   dfn3.3   Lax_Wendroff計算                                              void      私有
    dfn2.8   離散点速度(予測値)　u3[20]                                     実数型    私有
    dfn2.8   離散点速度(修正値)　UN[20]                                     実数型    私有
    dfn2.8   離散空間座標パラメータ　J                                      整数型    私有
    dfn3.1     |  J=0
    dfn3.1.2   |  while ( J<20 )
    dfn3.1     |    u3[J]=0.5*((1.0-CN)*u[J+1]+(1.0+CN)*u[J])
    dfn3.1     |    J=J+1
    dfn3.1     |  UN[0]=1.0
    dfn3.1     |  J=1
    dfn3.1.2   |  while ( J<20 )
    dfn3.1     |    UN[J]=u[J]-CN*(u3[J]-u3[J-1])
    dfn3.1     |    J=J+1
    dfn3.1     |  u[]=UN[]
    dfn3.2     |  データ出力 ( u )                  >> 駆動シナリオ
    dfn3.2     |  継続判定                          >> 一次元衝撃波管「マサ」
```

図 7　『衝撃波管』フレームの一部の OONJ 記述例

Fig.7　ODDJ description example for Shock Tube flow

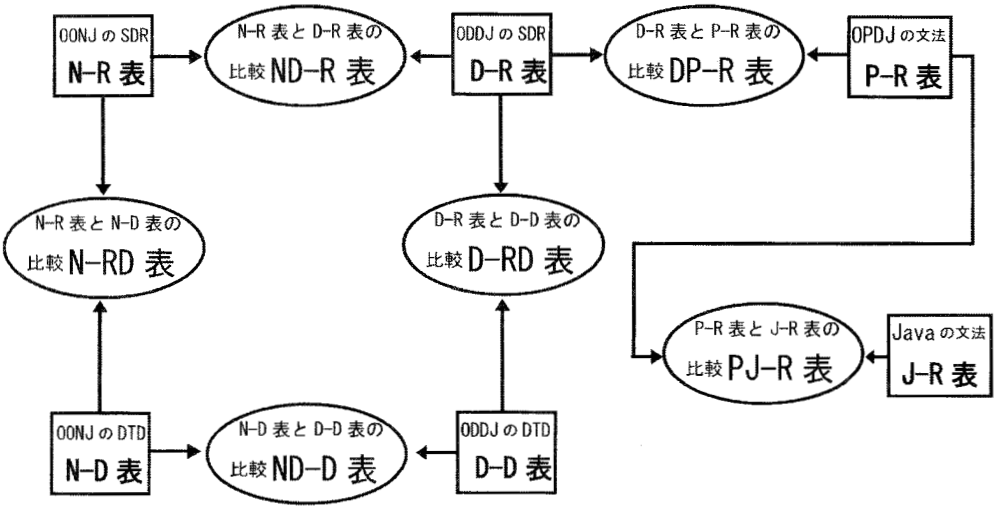

図 3 三つの記述言語の構造化記述規則，二つの DTD の比較

Fig.3 Comparisons of three descrption languages and two DTD

OONJ 記述

<サブスロット通常文> :=: <サブスロット記号> (<sp><構造化接続詞> (“, | .”)?<NJ 単位文> <行内注釈>?<フレーム間構造記述子>?
fn3.2	時間 T=0.0+ という情報 & 命令を受け取る . ≪ mp ≪ 6: シナリオ [2-5]

ODDJ 記述

<メソッド呼び出し> :=: <sp><メソッド呼び出し名><sp>“(”<実引数> (“,”<実引数>)* “)”
dfn3.2	Lax_Wendroff 計算 >> 一次元衝撃波管「マサ」

OPDJ プログラム

<call 文> :=: "call"<参照> <参照> :=: ≪フレーム名≫<配列要素>? “.”≪処理スロット名≫<実引数>
call 一次元衝撃波管「マサ」.Lax_Wendroff 計算 ( ) ;

Java プログラム

(文法は省略)
mthd5(); // 一次元衝撃波管「マサ」.Lax_Wendroff 計算

図 5 各記述言語の構造記述規則と一貫記述例の組

Fig.5 Sets of Structured description rules and descrption examples

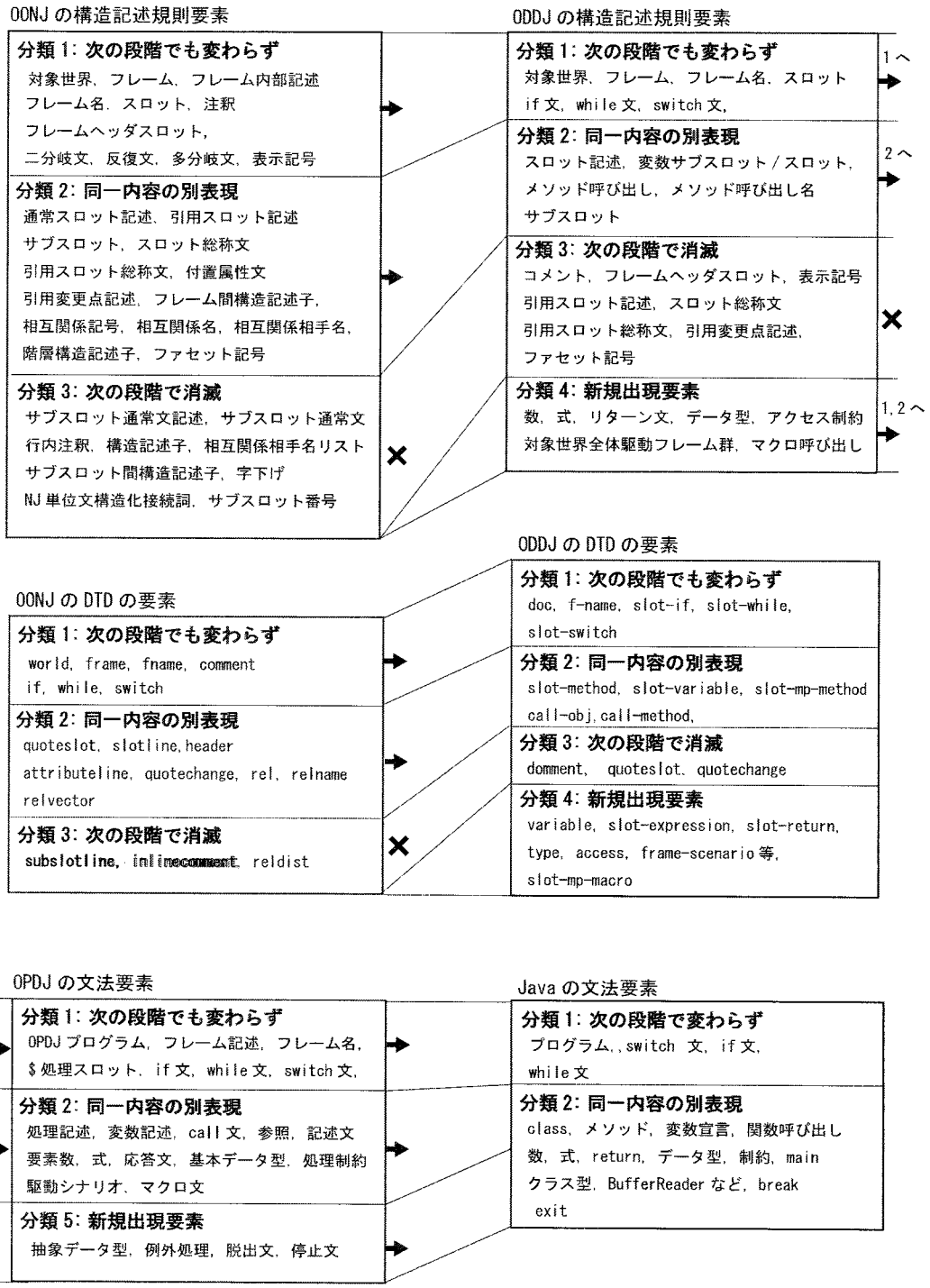

図 4　三つの記述言語と Java の間における各離散記述要素の変換関係

Fig.4　Transformation Relations of each discrete description elements among three description languages and Java