Vol.2012-MPS-90 No.19
2012/9/20

プログラム開発教育用記述言語OOJの設計と応用

—シンプルなモデルに基づく分析からプログラムまでの一貫開発—

畠山 正行[†1]　池田 陽祐[†2]　大木 幹生[†3]　三塚 恵嗣[†4]　上田 賀一[†1]　加藤木 和夫[†5]

概要：　本論文ではプログラム開発のプロではない科学技術計算分野の大学院生のために，分析から設計・実装・プログラムに至るまでを一貫して作業できる教育用の記述言語 OOJ を設計し，記述実験と実際の演習に応用した．OOJ は以前に開発した離散構造化モデルをベースとした記述言語 OONJ を起点として設計と実装の段階の記述言語を設計し，それらを統合的に構成・運用する．三つの各段階の言語の記述規則は実世界の OONJ の記述規則を計算機の世界において忠実に再現する方針で設計された．各記述言語は専用のエディタで記述を支援し，記述言語間は作成したトランスレータを使って変換する．OOJ の実際の応用と評価のデータは，数人の大学院生の記述実験と情報工学科の三年生の授業の一部として実施して得た．その結果，三年生の想定ユーザに近い受講生の大半と院生の殆どが Java プログラムまで得ることができた．院生の殆どは狙った通りの計算結果を出した．OOJ ための三年生向けの講義回数は 9 回のみ，演習は 6 回でこの結果である．この事実から OOJ の理解と記述は容易であること，学部三年生でも十分に応用可能であることが分かった．以上から，想定ユーザである大学院生向けに講義と演習として応用すれば，要求されるプログラム開発の実力養成は高い実現可能性を持つことが推定できた．

キーワード：プログラム開発教育，離散構造化モデル，オブジェクト指向，記述言語

A design and applications of descriptive language OOJ for the program development educations

Masayuki Hatakeyama[†1]　Yousuke Ikeda[†2]　Mikio Ohki[†3]　Keishi Mitsuka[†4]　Yoshikazu Ueda[†1]
Kazuo Katoug[†5]

Abstract: In the present paper, we have designed a descriptive language system OOJ that can be applied from the analysis stage up to the program transformation stage throughout the design and the implementation stage. The users have been assumed to be the graduate school students in the fields of calculations of science or engineering. They are non-professional users for the program development. OOJ has begun at OONJ that has been realized previously to represent the discreted and structured model as it is. The design and implementation stage descriptive languages have newly been designed, and they are integrally constituted. The description rules of each stage language in the computer world have been designed so as to reappear the original OONJ description rules for the real world. Every descriptive language are supported by each dedicated editor, and the transformation between languages are achieved by the translator. The actual application and estimation data have been obtained through the descriptive experiments by the graduate school students and the lesson for the third grade undergraduate students. As the results, almost third grade students and all the graduate school students have gotten the Java program. These results have been obtained through only 9 times lectures and 6 times exercises for OOJ. These results have shown that the recognition and the description of OOJ are easy, and the application to the third grade students is sufficiently possible. As the conclusion, the lectures and the exercises of OOJ for the graduate school students will be successful, and the requirements for the program development capability will be realized with high feasibility.

Keywords: program development education, discreted and structured model, object-oriented, descriptive language

Vol.2012-MPS-90 No.19
2012/9/20

はじめに

自然現象の解明や工学的問題の解決のために着目する問題領域の分析やモデリング，そしてシミュレーションは現在も活発に行われる研究方法の一つであり，その成果としての再現精度の高いシミュレーションを計算機上で行うことは分析結果やモデルの妥当性を主張するための重要な根拠となる [1], [2], [3].

このような人たちの役に立てるため，我々の研究グループでは主として自然現象等の科学技術計算のプログラム[*1]の開発を目的としたオブジェクト指向一貫記述言語系 OOJ[*2]の構築を行ってきた [4]．OOJ では対象世界に対して分析を行い，そこで得られた記述に対して計算機世界で実行・駆動させるための設計と実装を施すことで Java プログラムに自動変換する仕組みを採っている．本論文ではこの OOJ の仕組みと設計について述べ、実際の記述例や大学の授業に適用した結果を基に考察を行う。

以降、第 2 章では、基本となる離散構造化モデルについて述べ、第 3 章では OOJ の概念設計や具体的な設計方針等について述べ，第 4 章において詳細な設計を示す。第 5 章ではプログラム開発教育に応用した例を取り上げて評価する。第 6 章では関連研究との比較評価を行い、第 7 章で纏めと今後の課題を述べる。

2. 対象世界とその離散構造化モデル

2.1 想定対象世界とその一例

想定対象分野は“科学技術計算分野”である．したがって，計算式・論理式の計算処理に帰着することが多いと考えられる．そのような対象世界の一例として「一次元衝撃波流れ」を取り上げる．一次元衝撃波流れは衝撃波管内を伝播して行く衝撃波の再現シミュレーションである [5].

モデリングする衝撃波管内部の空間は**図 1** に示すようにセルという離散化された一次元の空間で区切られている [5]．図 1 の上部にモデリングにかかわる属性と振舞い及び数式を提示している．それらを用いると隣接するセル

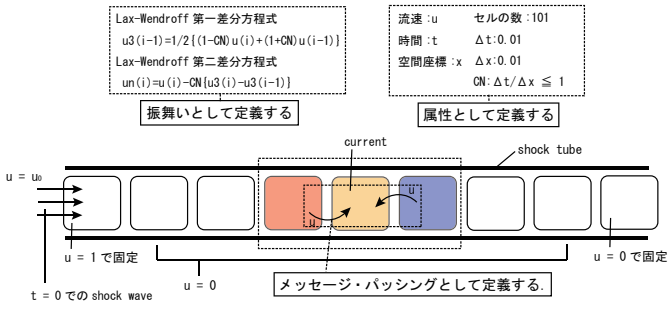

図 1 一次元衝撃波流れの離散構造化モデリング

Fig. 1 Discreted and structured modeling for one-dimensionnal shock wave flow

間で情報交換 (メッセージパッシン) グをすることで相互作用を表現できる．その相互作用によって衝撃波が伝播するモデリングを行っている．シミュレーションでは伝播して行く衝撃波に焦点を当てて近似計算式を作る．

2.2 OOJ の利用想定ユーザとその要求

まず**表 1** の【0】の想定ユーザからの要求の纏めを基にシステム設計・実装上の制約を勘案して実現可能なシステム構成を検討する．想定ユーザとしては，自身の分野に近い表現が出来る言語を望むと共に，自身の分野の情報はいくらでも提供するが，プログラミングに関わる事項には可能な限り関わりたくない，しかし実行可能なプログラムはほしい，という要求である．

そこで，分析から始まってプログラムの取得までを半期四ヶ月に収容してプログラム開発の実を挙げようとする試みが記述言語 OOJ の方法である．そのための主要な前提としては，最低限一種類のプログラミング言語を基地であり，多少の (数百行程度の) プログラムは組んだことのある学生ならば，可能であるとの見込みに基づいて開発したのが OOJ の方法 (システム) である．この条件に当てはまるのは理工系の大学院生である．

2.3 OOJ が準拠するモデリング方法とモデル

前節のような想定ユーザは，図 1 のように空間を離散化したモデルを使って計算プログラムを作ることが殆どであると言って良い．本論文ではこのような計算方法を一定の「形式化したモデル」に纏め直す．このように離散化したモデルを構造化するモデルを図で表すと，**図 2** のようである．このモデルを**離散構造化モデル**と呼ぶ．対象が微分方程式支配の世界であれば，微分方程式も同様に離散化される．差分法，有限要素法等がそれであり，図 1 の左上にその一例を示した．

離散構造化モデルは対象世界の自然現象等を出来るだけ

†1 現在，茨城大学工学部情報工学科
Presently with Department of Computer and Information Sciences, Ibaraki University
†2 現在，茨城大学大学院理工学研究科情報工学専攻
Presently with Graduate School of Information and System Science, Ibaraki University
†3 現在，群馬工業高等専門学校教育研究支援センター
Presently with Technical Support Center for Education and Research, Gunma National College of Technology
†4 現在，株式会社日立システムズ
Presently with Hitachi Systems, Ltd.
†5 現在，茨城県立産業技術短期大学
Presently with Ibaraki Prefectural Industrial Technology Junior College

*1 本論文では，プログラミング言語を用いた記述のみを「プログラム」と呼ぶ．それ以外はすべて単に「記述」と表現する．
*2 Object Oriented Japanese の略

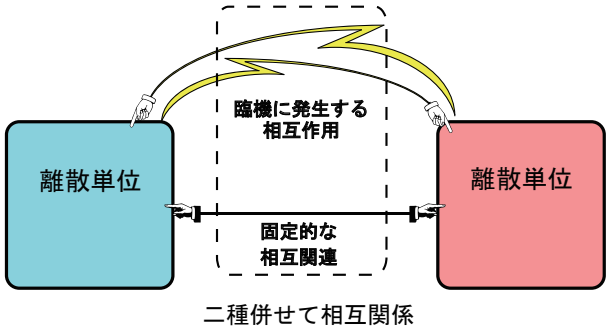

図 2 離散構造化モデルの概念図
Fig. 2 A concept expression of discreted and structured model

対象世界のままに捉える/写像すると同時に，デジタル計算機で計算可能な記述にも変換できるように，という狙いで定義され/使われてきたモデルである[*3].

そこで本論文で設計する OOJ はこの離散構造化モデルを直截に記述できる記述言語として設計する．そして OOJ を用いた記述の最終プロダクト (出力) はプログラムであるが，このプログラムは直截に記述された離散構造化モデルが忠実に反映されていることが必須条件である．

3. OOJ のシステム構成

想定ユーザの要求を概念設計という形に纏めると表 1 の【I】の (1)〜(3) となる[*4]．全体概念図を**図 3** に示す．

3.1 OOJ の内部構成：三つのサブ言語構成

OOJ の概念設計は表 1 の【I】であるが，この【I】の (1)，(2)，(3) は各々異なった要求から導かれた方針であり，言語特性としては各々がかなり大きく異なる．したがってこの 3 項目を同時に実現するためには，持つ機能が複雑になるので，当初から一つの記述言語として設計するのは困難である．そこでこの OOJ の設計を容易にするために【I】の (1)，(2)，(3) を実現する機能を分割して実現し，後で単一の言語として再統合化する，という方針とした．

また別の観点から言えば，この OOJ が大学院生のプログラム開発の教育目的に利用されることを考慮に入れ，プログラム開発の当初分析から設計や実装を経てプログラムに至るまでを経験しておくことは必須である．そこで，内部の見かけの構成として分析，設計，実装の各段階に分け，それぞれの段階でなすべき作業を特定し，明確な作業区切りを付けて OOJ を設計して行くこととした．そこで分析・設計・実装の各段階に対応する三つのサブ言語に分割して設計する．【I】の (1) が分析段階に，(2) が設計と実装の段階に，(3) が実装段階に相当する．

OOJ 内部の三つの各記述言語の設計方針の概略と役割分担を表 1 の【II】に示す．表 1 の【II】には各記述言語の重ならない個々の役割分担を割り当てた．各記述言語は【II】の設計上の役割を忠実に実現した記述規則にすることが必須である．

3.2 OOJ の設計目的とその実現すべき特性

前節の三つのサブ言語という基本構成を前提にして OOJ の設計目的を表現すると，以下のようになる．

(1) 既存の OONJ((Obect Oriented Natural Japanese)) のように対象世界の離散単位と構造に忠実な記述を作成できるような規則に設計する．

(2) 設計記述言語である ODDJ(Object-oriented Design Description Japanese) は，OONJ を計算機で計算できるような記述/表現に変換する．ただし，OONJ と相似な記述を作成可能な規則に設計する．また，OONJ の対象世界のモデルである離散構造化モデルを忠実に変換できるパラダイムとして，オブジェクト指向 (必要に応じて **OO** と略す) を使う．

(3) 実装記述規則 OPDJ((Object-oriented Program Description Japanese)) も同様の条件で設計する．更に OPDJ では変換目標言語としたオブジェクト指向プログラミング言語 (以降，必要に応じて **OOPL** と略す) に変換できる仕様にする．

(4) 三つの記述言語は，対象世界の構造の相似性を保持出来る規則でなくてはならない．本論文で**相似性**とは変換前後の個々の離散単位毎について，近似的な計算/処理結果を出す記述あるいは規則の性質を指す．

3.3 OOJ のシステム構成の全体概略

前節の概念設計を実現するため，想定ユーザの要求とシステム上の制約等を検討した結果，本論文では表 1 の【III】のようなシステム構成で設計することとした．次章以降ではこの設計方針を具体化させた記述言語を提案する．

(1) 当初は手作業で OONJ を設計する．完成した OONJ 記述規則は【III】の 1.〜3. の特性を組み込んだ設計とする．

(2) 【III】の 3. 項目および 4.1 項目のように離散構造化モデルをオブジェクト指向モデルに読み替えて ODDJ の記述規則を設計する．

(3) ODDJ では 4.2 項のように計算機世界の共通の項目や用語について，同等内容の別表現に変換するか，新規に記述する．

(4) OPDJ では 4.3 項のように目標とする OOPL に共通の表現に変換するか，新規に設計する．

*3 その事情が，ソフトウェア工学の立場から見ると，OONJ は純粋の分析モデルでは無く，設計段階のモデルでもあると評価される理由である．
*4 (4) の要求は将来必要ではあるが本論文では省く．

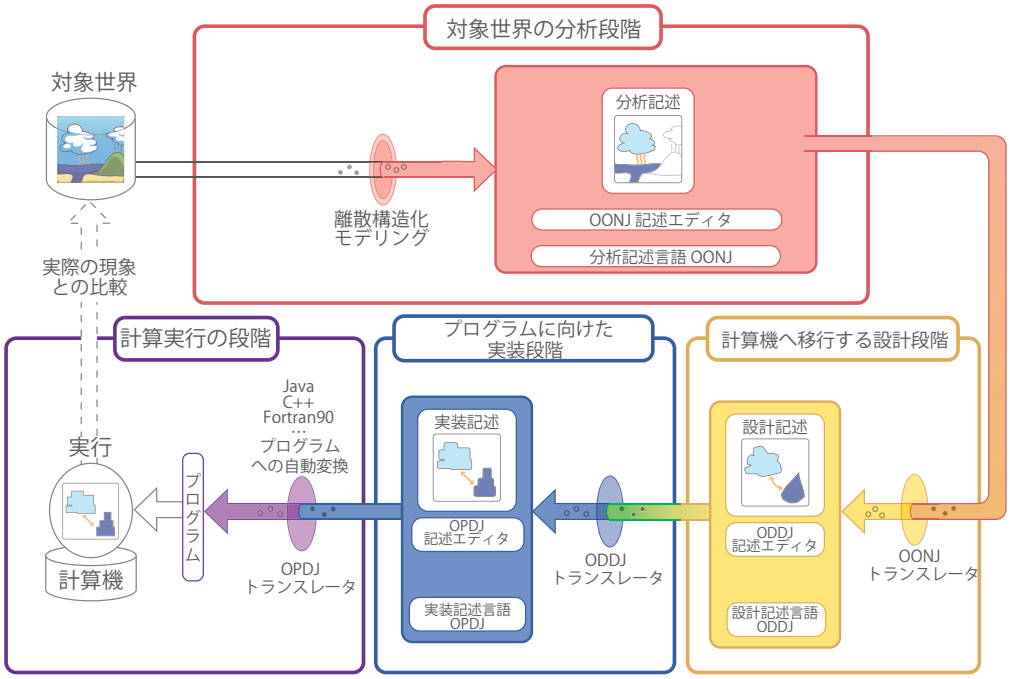

図 3 OOJ を基盤としたプログラム開発過程の全体概念図
Fig. 3 A whole concept diagram of program development processes based on OOJ

(5) OPDJ 記述の完成形は 4.4 項と 4.5 項のように，トランスレータ変換によって実行可能な OOPL プログラムに変換されて出力出来ること．そして OPJD 記述が OONJ 記述と相似になっていること．

4. 記述言語系 OOJ の設計詳細

4.1 OONJ の概要

分析段階である OONJ では離散構造化モデルを直截に記述できる必要がある．この離散化モデルの記述方法については，既に多くの方法が示されているが [5], [6]，構造化表現の方法が明示的ではないことが多い．OOJ では OOSF[7] の構造化の方法を用いる．そして OOSF の規定を OOJ のシステム構成に合うように以下の性質を追加した．まず OONJ の記述規則を**表 5** に示す．

この表 5 を理解しやすくするため，UML のメタ記述法を用いて表 5 の概要を**図 4** に示す．図 4 から OONJ が離散単位を中心とし，相互関係を用いて離散単位間を構造化する仕組み，すなわち離散構造化モデルの仕組みが規定されていることが分かる．これらのことから OONJ は，図 2 の離散構造化モデルが記述可能であるといえる．

4.2 ODDJ や OPDJ への記述規則変換

記述規則変換の具体作業を示す．そのために，**表 2** に三つの言語の規則番号とその変換の対応関係を示す．まず OONJ の規則に存在する 1〜34 までを最左段にある．表内で“←”の記号は前の段階の言語と規則が同じ (変わらない) ことを意味する．34 までは大半が変化しないので，変化しているあるいは新たに設計された規則についての相似性を個々に検討すれば良い．

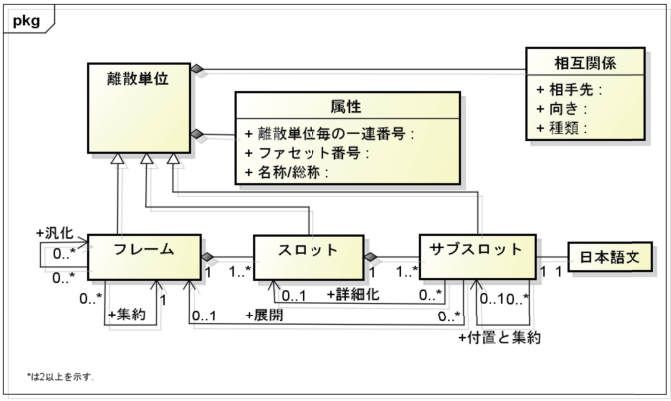

図 4 メタ記述表現法を用いた OONJ 記述規則の全体構造
Fig. 4 Whole structures of OONJ description rules using meta-descrption notations

検討すべき規則は**表 3** の (A) の部分に挙げてある．規則番号の右側にある D や P は設計段階や実装段階での変換規則であることを示し，その実体は表 3 に示されている．番号のない規則は変更されていないことを示す．これらの変換作業の内容を OONJ の規則番号と照らし合わせ相似に変換されているか否かを個々に検討すれば良い．結果はほぼ全てについて相似的な変換であることが確認できた．

次に OONJ には存在せず ODDJ から始まる記述規則がある．それが表 2 の“新”という文字を付した ODDJ 起源

表 1　想定ユーザ，OOJ の概念設計、各言語の役割分担、OOJ 詳細設計方針
Table 1　Assumed users, concept designof OOJ, role sharing of each language, detailed design of OOJ

【0】想定ユーザの定義と特徴

(A)OOJ を使う想定ユーザは理工系の大学院生とする．彼らは理工学分野の研究や技術開発のプロになるための訓練を受けており，一定以上のレベル?のプログラム作成技術を必要としている．想定ユーザに準ずる人たちには研究室配属の 4 年生，さらには実務に就いている研究者や技術者?の一部も入る．
(B) そのために最終的には数千行程度 (10000 行未満) のプログラムを作成出来る程度の技術の習得を目標としている．
(C) 彼らは計算内容的に見て複雑で精密な計算処理プログラムとその結果を必要としている科学技術計算のプロの卵ユーザである．
(D) 彼らは自身の研究開発の必要上，自身でモデリングからプログラムまでを一貫して行い計算も行う．複雑な関わりを持つ計算式やデータ構造を扱うのが特徴である．その計算結果から対象世界の知見や設計データ等を得ることを主な目的としている．
(E) 現在大学院教育中であり、自身の専門分野について一応の水準の知識やモデルを既に持っており，対象世界の分析も自力で行える．
(F) 多くは手続き型のプログラミング言語一つを必要な範囲で知っており，「他との協同を前提にすること無く」，「自身だけで」，「ゼロから一貫して自主的に」，「他人用ではなく自分用」に「小規模」のプログラム作成の経験を有する．
(G) プログラム開発の専門家ではないのでその方面の多くの知識やノウハウ，高い OO プログラミング技術は期待出来ない．また、プログラム作成の負担は最小限を，自身の分野の知識の記述については必要十分に詳細な記述に応じることが出来る．

【I】OOJ の全体的な概念設計

(1) 最初の分析記述は自身の分野に近い言葉で書きたい。：この方針は、例えばその分野の教科書や論文の記述にあるいは対象世界のイメージに可能な限り近い記述にするためである。想定ユーザの記述は分析段階でその大半を投入するので、異なった分野の言語や記号とかを使わなくて良い記述法や言語が必要である。その事情から、分析記述言語の主用言語は教科書や論文がそうであるように日本語文にするべきである。
(2) 途中の設計や実装の段階では想定ユーザのみにしか分からない/知らない情報のみを提供する。：これは勿論、想定ユーザの心理的なモノも含めて負担を少なくするためである。残りの記述は分析段階からの情報を自動変換して補うべきである、という方針とする。
(3) 最後には実行可能なプログラムへ自動変換して出力．：これも勿論、想定ユーザの負担を少なくするためである．想定ユーザはプログラミングの文法の中、細かい記法に神経を消耗させられるのを避けたい気持ちが強いので、必須の機能である．
(4) 途中段階を含めた記述に対する自在な分析とトレースを行う．：これは OOJ の設計と直接には関わらないが、各段階毎の記述とその間の変換を比較評価したり、出力されたプログラムと分析記述との相似性を検証したりするのに必要である。

【II】OOJ 内部の各言語の役割分担

分析 (設計) 記述言語 OONJ	対象世界に関する情報で，プログラム生成に至るまでに必要となる情報 (離散単位や構造についての情報) を全て記述すること．最小離散単位の日本語文は単文であることが推奨される。そしてその離散単位の構造化を要求される．数式等は各記述者の考えで記述して良い．計算機世界に関わる情報は記述しない．
設計記述言語 ODDJ	計算機世界特有な要素の追加や表現変換を行う．例えば，属性を変数と呼び変え，配列やリスト構造を導入したりする．特定の OOPL に関わる事項は記述しない．
実装記述言語 OPDJ	各特定プログラム言語 ((OO)PL) 向けのプログラムに自動変換するための記述を作る．現時点では，Java, C++，Fortran90 に変換可能である．
記述言語系 OOJ	上記の 3 つの記述言語を概念上で統合した言語の名前であると共に、三つの記述言語の設計主題と解決目標の枠組を示すモデルでもある..

【III】詳細設計の方針

1．分析用のモデリング方法として「離散構造化モデル（図 2)」[*5]を採用する。
2．この離散構造化モデルを表現する専用の記述言語を設計する。(これで、上記の要求 (1) は実現可能になる。)
3．この離散構造化モデルをオブジェクト指向モデルの限定モデルに変換する (読み替えモデル)。
この読み替えモデルは離散構造化モデルと同等内容の別表現を行えるモデルでなくてはならない。
4．その読み替えモデルを表現する専用の設計・実装段階の記述言語を設計する。ただし、以下の条件を満たすものでなくてはならない。
4.1　二つの言語は分析段階のそれと同じように「離散構造化モデルと同等内容の別表現になる」言語でなくてはならない。
4.2　設計段階の記述言語では、計算機世界の共通な事項に関する変換を行えるように設計する。
4.3　実装段階の記述言語では、ターゲット言語の共通な事項に関する変換を行えるように設計する。
4.4　各言語での記述を完成して次の段階に移る時にはトランスレータに掛けて変換させる。(以上により要求 (2) は実現可能になる。)
4.5　実装記述言語からプログラミング言語に変換するトランスレータにおいては、実行可能なプログラムに変換できることを設計条件とする。(これにより、要求 (3) は実現可能になる。)
5．以上の記述言語、トランスレータ、(実はエディタも含まれるが)、等は全て共通なデータ形式で定義しておき、そのデータを分析することで各段階の記述をの分析とトレースを実行可能にしておく。(これにより、要求 (4) が実現可能になる。)

表 2 OONJ から ODDJ，OPDJ への対応関係
Table 2 Correspondence of OONJ to ODDJ and to OPDJ

OONJ 規則番号	ODDJ 規則番号	OPDJ 規則番号	理由説明等
1〜3	←	←	
4	4D	4P	
5	5D	5P	
6	6D	6P	
7	×	×	ODDJ 以降の新規則に移行
8	←	8P	
9〜10	←	←	
11	←	11P	
12	←	12P	
13	←	←	
14	←	14P	
15	15D(注 1)	←	
16〜17	×	×	ODDJ 以降の新規則に移行
18〜19	←	×	
20	× (注 2)	×	ODDJ 以降の新規則に移行
21	21D	21P	
22	22D	22P	
23〜24	←	←	
25〜30	←	←	
31〜34	←	←	
×	35D 新	35P	計算機表現に移行する変換
×	36D 新	←	同上
×	37D 新	←	同上
×	38D 新	←	同上
×	39D 新	39P	同上
×	40D 新	40P	同上
×	41D 新	41P	同上
×	42D 新〜	←	OONJ の日本語文をライブ
	〜105D 新	←	ラリとして定義した規則.
×	×	42P 新	ターゲット OOPL へ変換
×	×	43P 新	同上
×	×	44P 新	同上
×	×	45P 新	同上
×	×	46P 新	同上
×	×	47P 新	同上
×	×	48P 新	同上
×	×	49P 新	同上
×	×	50P 新	同上
×	×	51P 新〜	三種のターゲット OOPL
		〜82P 新	から抽象化定義した規則。

(注 1)：OONJ の規則項目の一部削除.
(注 2)：ODDJ 以降ではコメント相当の日本語文として扱われる。

の規則番号である．その実体の規則は表 3 に示してある．このカテゴリーの規則は個々に計算機処理の世界で特定する何かが，OONJ 記述規則に反する，つまり相似性を崩す規則であるか否かを検討すれば良い．これらも個々に検討する．【B】の 35 から 38 に関してはデータ型とアクセス修飾子の事項であるので，相似性を崩す規則ではあり得ない．39〜41 までは OONJ では抽象的であった数式やメッセージパッシング (mp) を具体化した規則であり，同様に相似性を崩す規則ではない．以上のように【B】の ODDJ 起源の規則は段階的に計算を詳細化する過程で現れた規則であり，相似性を崩す規則ではないことが分かる．また 42D 新〜105D 新という規則は OONJ の日本語文を新たにライブラリとして定義した規則であり，選択的に置き換えて使う規則群である．これらの定義の詳細は省略した．

最後に OPDJ から始まる“新”記述規則群を表 2 に示す．いずれも三種のターゲット OOPL の規則 (文法) 対応に定義された規則である．これらも同様に表 3 に実体が示されている．ここでの【C】の OPDJ 起源の規則は，制御構文や変数や引数等の詳細化を規定した規則で，【B】と異なるのは変換先のターゲットとした複数の OOPL [*6]の共通的な文法に変換するための記述規則であることである．これらの変換作業の内容も同様に個々に検討した結果も問題なく相似的な変換であることが確認できた．また 51P 新〜82P 新までの規則は，式，関数やスロットそしてライブラリなどの呼び出し規則が定義されている．これらの定義の詳細は省略した．

以上から，三つの言語の記述規則間の記述や変換は OONJ 記述規則の相似性を崩すものではないことが分かった．なお、非常に多くの規則が存在するように見えるが、表 2 の 15 番以降は OOJ 専用に開発された三つのエディタ (図 3 を参照) が自動あるいは半自動でサポートしており、ユーザが記憶して使いこなすべき規則は実体としてその主なものは、規則番号で言えば 1 番〜14 番までに過ぎない。

4.3 OPDJ の設計方法

OOJ の最終段階である OPDJ は三種類の変換先 OOPL のインタフェース仕様でもある．本章の文法を満たした記述は OPDJ トランスレータによって実行可能なプログラムに変換される．OONJ ではプログラム相当の全体構造が仕様として定義されたが，OONJ の日本語文は OOPL の文法詳細については殆ど規定していない．文法事項の詳細は ODDJ 及び OPDJ において規定する．そこで詳細な文法事項が表 2 と表 3 の規則の変換によって段階的に詳細化した規則として定義される．つまり，OONJ で日本語文が担ってきた概略だけの規則 (表 2 の OONJ 規則の 7) は ODDJ と OPDJ の新規規則 (表 2 の下半分（35D 新以降の全て) の新規な規則に依って OOPL の文法に変換された．詳しくは参考文献 [8] を参照されたい．

5. プログラム開発教育への適用と評価

OOJ 評価のための記述実験を情報工学科三年生の講義&演習「オブジェクト指向プログラミング」に重ねて行い，受講生の一部について本来の想定ユーザである大学院生に比較的近い学生のデータを抽出した．抽出の基準は離散構造化モデリングを良く理解し分析段階の良好な記述を作った学生で，35 名中 8 名を抽出した．

OOJ 評価の対象は最終レポート，アンケート，考察，試

[*6] 現状では Java，C ++，Fortran90 である．

表 3 新規な記述規則の定義や変換作業の内容
Table 3 Newly defined descrption rules or transformation operations

【A】OONJ 記述規則からの変換

変換番号	OONJ からの変換を定義する。
4D	右辺に付置属性記述を付与.
5D	右辺にデータ型とアクセス修飾子を追加.
6D	右辺のサブスロット主文を複数の命令文へ変換.
7D	規則番号 39〜41 の規則に変換。
8D	右辺にデータ型を追加.
4P	右辺の付置属性記述を仮引数へ変更.
5P	右辺の相互関係を削除.
6P	右辺のサブスロット主文を複数の命令文へ変換.
8P	付置属性を仮引数や実引数に変換.
11P	複数の相互関係相手先を単一の相互関係相手先に変換.
12P	右辺の一部を削除 (詳細省略).
14P	右辺の一部を削除 (詳細省略).
15D	拡張日本語文や注釈文は，数式やコメントに対応するためその部分を削除.
21D	“nfn”の表記を“dfn”へ変換
22D	ODDJ の離散単位の種類表 (詳細省略) の対応へ変換
21P	“dfn”の表記を“pfn”へ変換.
22P	OPDJ の離散単位の種類表 (詳細省略) の対応へ変換.

【B】ODDJ 起源の規則

変換番号	**計算機での共通な表現への変換するための規則の新設や変更を行う**.
35D 新	＜付置属性記述＞:=:(＜ lf ＞＜付置属性文＞)$^{+}$
36D 新	＜計算機特定指定子＞:=:＜ sp ＞＜データ型＞＜ sp ＞＜アクセス修飾子＞
37D 新	＜データ型＞:=:(＜整数型＞｜＜実数型＞｜＜論理型＞｜＜文字型＞｜＜文字列型＞ ｜＜整数配列型＞｜＜実数配列型＞｜＜文字配列型＞｜＜ void ＞｜＜フレーム名＞)
38D 新	＜アクセス修飾子＞:=:(＜対象世界共通＞｜＜フレーム内共通＞｜＜スロット内共通＞)
35P	付置属性文を仮引数に変換.
39D 新	＜数式サブスロット＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜数式＞＜相互関係＞$^{?}$
40D 新	＜スロット呼出しサブスロット＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜日本語文＞ ＜相互関係 (駆動 mp) ＞＜付置属性記述＞$^{?}$
41D 新	＜戻りサブスロット＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞ ＜日本語文＞＜相互関係 (戻り mp) ＞＜付置属性記述＞$^{?}$
39P	＜数式サブスロット＞ (39D 新) を ＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜式＞に変換.
40P	＜スロット呼出しサブスロット＞ (40D 新) を ＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜スロット呼出し＞に変換.
41P	＜戻りサブスロット＞ (41D 新) を ＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞“return”＜単純式＞に変換.

【C】OPDJ 起源の規則

変換番号	**ターゲット OOPL のプログラムへの変換のための規則の新設や変更を行う**.
42P 新	＜反復脱出文＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞“break”
43P 新	＜反復飛ばし文＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞“continue”
44P 新	＜変数＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞ (＜変数名＞｜＜実値オブジェクト名＞) (“[” ＜要素数＞ “]”)$^{?}$(“=” ＜単純式＞)$^{?}$ ＜ rt ＞＜計算機特定指定子＞
45P 新	＜コメント＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞“(コメント)”＜日本語文＞
46P 新	＜そのまま出力文＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜ユーザ自由記述＞
47P 新	＜仮引数＞:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜仮引数名＞＜ rt ＞＜計算機特定指定子＞
48P 新	＜実引数＞:=:＜単純式＞
49P 新	＜式＞:=:(＜単純式＞｜＜比較式＞｜＜論理式＞｜＜代入式＞)
50P 新	＜単純式＞:=:(＜文字列式＞｜＜算術式＞) ...
79P 新	＜スロット呼出し＞:=:(＜実値オブジェクト名＞“＝＞”)$^{?}$ ＜スロット名＞“(” (＜実引数＞)(“,” ＜実引数＞)*)$^{?}$ “)”

験の中から本来の想定ユーザである大学院生が OOJ を用いた時に参考になるデータのみを抽出し，更に大学院生の記述実験の 4 例も加えて、OOJ を評価した．抽出したデータは，以下の三項目である．結果のデータを**表 4** に示す．

表 4　最終レポートの結果
Table 4　Results of the final reports

対象世界名	OONJ	ODDJ	OPDJ	Java	実行
ビリヤード	146	140	163	274	×
ミニ四駆の走行	172	193	238	283	×
斜面上の物体運動	185	195	254	280	○
直線上の三体衝突	203	203	243	×	×
メタンの燃焼	210	147	167	210	×
鉛蓄電池	225	190	205	337	×
バドミントン	237	229	234	×	×
食物連鎖	250	250	275	415	×
赤血球の生成	215	217	275	485	○
斜方投射と跳返り	260	261	284	544	○
アルコールの蒸溜	302	321	386	617	○
列車運行	834	864	1009	1414	×

(1) 作業内容の質的な容易性の指標として，到達段階
(2) 複雑さと記述力の指標として，記述行数と記述時間
(3) 理解度の指標として，考察項目の分析と試験成績
項目 1 と 2 は表 4 から見出せる．結果は大半の学生と全院生が Java プログラムに到達し，実行できた．院生は実行結果の物理的/化学的な考察も行っていた．

記述時間については平均時間で測る．モデリングと分析記述までは 7.4 時間，OONJ 記述が 10.7 時間，残りの ODDJ 記述以降の所要時間は平均 13.6 時間であり，合計 31.7 時間であった．院生の方は同順に 16.3 時間，11.5 時間，18.3 時間，46.1 時間であった．記述行数は記述力と対象世界の複雑さのバランスに依る．院生の時間数が (特に分析時間が) 多いのは三年生よりは専門的な内容であるので頷ける数字であると考えられる．またそれ故に行数があまり大きな差が付かなかったと考えられる。

項目 3 については，記述式での試験の平均点は 89.1 点で，概念や知識の理解という点では問題はなく，考察の項目も，レポート完成時の記述なので，離散構造化とオブジェクト指向の共通点等も理解していたようである．
以上，OOJ の理解と実行について半期 1 コマの授業構成にして、講義回数が 15 回中の 9 回という少ない回数でもほぼ必要な理解と記述が得られたことは大きく評価できる．

6. 関連研究や技術との比較評価

本論文では MATLAB と MDA を取り上げる．

6.1 MATLAB との比較

まずシミュレーションプログラム開発支援のツールとして MATLAB[12], [13] と比較する．MATLAB は行列計算やベクトル演算などの数値計算を得意としたシミュレーションツールであり，独自パッケージである Simulink と組み合わせることで信号処理や制御回路などのシミュレーションを行うことができる．特に Simulink を使ったプログラム開発ではビジュアルエディタを用いて開発することができ，ユーザは既に用意されている微積分ブロックや論理演算ブロックなどのブロックダイアグラムを画面中に配置するだけでシミュレーションを行うことができる．

さらに M コードと呼ばれる独自の言語とその実行環境，および各分野に特化した豊富なライブラリを持ち，シミュレーションプログラムの開発が可能である．MATLAB で開発されたプログラムは C と Fortran のプログラムに変換することが可能であり，変換されたプログラムは実際の製品の組込みプログラムとして適用することができる．

OOJ との比較の観点で見ると，MATLAB は Simulink を使ったブロックダイアグラムを用いてモデルを記述し，それを抽象化によるプログラムの自動生成技術を用いて円滑なプログラミングを提供している．そのためブロックダイアグラムを記述することでプログラム，もしくは結果を即座に得ることが可能である．さらに，MATLAB の実行系でできることは自動変換されたプログラムでもできるという強みがあり，変換先言語特有の機能を制限している OPDJ と比較すると利用できる機能が豊富である．

一方，OOJ では複数種の異なる OOPL に自動変換することに注力しているため，MATLAB のように豊富な機能は備えていない．プログラムを得るには，教育目的であるために複数の段階を忠実に経るために，MATLAB に比べて作業時間は増加する．しかしこれは教育的な側面から整備された仕組みなので比較にはならない．

以上の様にシミュレーションプログラム開発を効率良く行えるという点では MATLAB が優位であり，教育段階の大学院生ユーザにとっての使い勝手という点では OOJ の方が優位であると考えられる．両者のそのような特徴から考えると，OOJ は MATLAB に代表される Domain Specific Language の front-end としての使用も可能だと考えられる．

6.2 MDA との比較

MDA[18] とは，ソフトウェア工学におけるモデル駆動開発 (Model Driven Developments 以降，MDD と略記) の手法の一つであり，OMG で正式に定義されている．MDD では，業務過程とプラットフォームを独立して記述し，最後に結合し Implementation Model を得て，それを自動生成によって C/C ++，Java，Ada95 などのソースコードを得ることが出来る．その中でもアジャイル MDA として Excutable UML[19] があり，そのツールとして BridgePoint がある．具体的な流れとして，クラス図とステートメント図などの幾つかのダイアグラムと Action Language を用いて業務過程を記述し，各プラットフォーム毎のアーキテクチャを記述する．そしてそれらを統合し，最終的にはソー

スコードを自動生成する．

これは OONJ で対象世界を記述しそれを OPDJ を経て各種プログラムを得るとの開発過程が類似しており，開発環境としての視点による比較が可能である．MDA や MDD は企業で行われる製品開発や組込みシステムの開発，業務シミュレーション [20] の開発で使用されており，OPDJ と OONJ が対象としている教育目的とは異なる．また，MDA では要求される技術が高いことが示されている*7．さらに磯田 [21] はモデリング法そのものについても，離散構造化モデリングは業務システムの開発の際のモデリング手法が大きく異なっているので，大学院生には適切に区別して用いる技術を持つことが難しい．大学院生にはこれらの技術力は望めないので利用は難しいであろう．

7. まとめと今後の課題

纏め

(1) 三年生や院生の記述結果から、設計した OOJ がその目的の合致した結果を出しており，OOJ の設計目標はある程度実現できたと判断できる．

(2) 言語としての OOJ の習得は例えば Java 等のプログラミング言語よりも短時間で良い．また，トランスレータが使えるどの言語にも同じように出力できる．これらは想定ユーザにとっては好ましい結果である。

(3) OOJ の仕組みへの理解が三年生でもほぼ問題ないことを根拠とすれば，より専門性の高い想定ユーザには理解は十分に容易であると推定できる．

(4) 分析からプログラムまでの一貫した流れを短期間で理解することが出来るので，教育用言語として将来のプログラム開発業務に役立つことは十分に推定できる．

今後の課題

実用プログラムに近い記述例やライブラリーの整備，トランスレータの機能強化，標準スタイル等の整備によってより複雑なプログラムも開発可能になると予想される．

1．より複雑な対象世界を表現出来る仕様への改訂．

2．DSL のフロントエンドに使える言語への拡張．

参考文献

[1] 渡邉一衛：ものづくり・サービスづくりのシミュレーション，日本シミュレーション学会論文誌，2009 年 3 月 vol.28,No.1，pp.41-45(2009).

[2] 加藤廣：自動車開発のデジタル化はシミュレーションが主役，日本シミュレーション学会論文誌，2008 年 6 月 vol.27,No.2，pp.50-53(2008).

[3] 日本計算工学会編，工学シミュレーションの標準手順，」JSCESS-HQC002:2011 A Model Procedure for Engineering Simulation，日本計算工学会，2011 年 5 月 31 日．

[4] 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，pp.49-56(2009).

[5] 数値流体力学編集委員会編，数値流体力学シリーズ 2，圧縮性流体解析，東京大学出版会，1995 年．

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

[7] 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).

[8] 三塚恵嗣，オブジェクト指向一貫記述言語系 OOJ における変換機構と実装記述言語 OPDJ の開発，平成 22 年度茨城大学大学院修士論文，平成 23 年 (2011 年)2 月 10 日．

[9] 吉田紀彦，自動プログラミングハンドブック第 4 章「プログラム変換手法による自動プログラミング」，オーム社，pp. 109-120(1989).

[10] Burstall,R. M.,Darlington,J.:"A Transformation System for Developing Recursive Programs",J. ACM, 24, 1, pp. 44-67(1977).

[11] 畠山研究室，〈http://gaea.cis.ibaraki.ac.jp/〉，(accessed 2012-03-15).

[12] MathWorks MATLAB(online)，入手先〈http://www.mathworks.co.jp/products/matlab/〉，(accessed 2012-05-05).

[13] MathWorks Simulink,http://www.mathworks—.co.jp/products/simulink/description1.html，(access 2011/11/15).

[14] Michael M. Tiller，Modelica による物理モデリング入門，オーム社，平成 15(2003 年) 年 11 月．

[15] 計算力学研究センター，SAMCEF MECANO(online)，入手先〈http://www.rccm.co.jp/product/structure/samcef-mecano/〉，(accessed 2011-12-26).

[16] NewtonWorks，SimulationX(online)，入手先〈http://www.newtonworks.co.jp/caecad/simulationxgaiyou.html〉，(accessed 2012-03-10).

[17] LMS，AMESim(online)，入手先〈http://www.lmsjapan.com/imagine-amesim-suite〉，(accessed 2012-03-10).

[18] OMG(Obejct Management Group)，MDA Specifications，http://www.omg.org/mda/specs.htm，(accessed 2012-05-05).

[19] OMG(Obejct Management Group)，Building and Implementing Concurrent Specifications，http://www.omg.org/news/meetings/workshops/RT_2005/00-T2_Starrett-Mellor.pdf，(accessed 2012-05-05).

[20] Annette VINCENT，Alan BRAIN，http://anu.academia.edu/ZoeBrain/Papers/133897/Simulation_Case_Study_-_xtUML_in_agile_development，(accessed 2012-05-05).

[21] 磯田定宏，実世界モデリング有害論—オブジェクト指向モデリング技法の解明，電子情報通信学会論文誌，Vol.J83-D-I, No.9, pp.946-959, (2000).

*7 具体的には UML(各種開発環境を使いこなす能力) を十分に使いこなす能力，対象となるプラットフォームのアーキテクチャを記述できる能力，そしてソフトウェア開発の経験が必須であり，多人数による開発が前提である．

表 5 OONJ 記述規則
Table 5 OONJ descriptive rules

離散単位	(規則 2 の【 】と，4 の《 》はそれぞれが容器であることを表す幾何学的な枠線を指す．)
1 **＜対象世界＞**	:=:＜対象世界共通要素フレーム群＞ (＜ lf ＞＜フレーム＞)$^{+}$
2 **＜フレーム＞**	:=:＜フレームヘッダ＞【(＜ lf ＞＜フレームヘッダスロット＞)$^{+}$(＜ lf ＞＜スロット＞)$^{+}$】
3 ＜フレームヘッダ＞	:=:＜フレーム要素特定子＞＜ sp ＞＜フレーム名＞＜相互関係＞$^{?}$
4 **＜スロット＞**	:=:《＜スロット総称文＞ (＜ lf ＞＜サブスロット＞)$^{+}$》
5 ＜スロット総称文＞	:=:＜スロット要素特定子＞＜ sp ＞＜日本語文＞＜相互関係＞$^{?}$
6 **＜サブスロット＞**	:=:＜サブスロット主文＞ (＜ lf ＞＜付置属性文＞)$^{?}$
7 ＜サブスロット主文＞	:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞＜日本語文＞$^{?}$＜相互関係＞$^{?}$
8 ＜付置属性文＞	:=:＜サブスロット要素特定子＞＜ sp ＞＜階層構造記述子＞ (“(”＜属性名＞ (“,”＜属性名＞)* “)”)
構造記述子	(相互関係を付与することが構造を表現する方法であることの仕組みや符号を指す．)
9 **＜相互関係＞**	:=:＜ rt ＞＜相互関係記号＞＜相互関係相手名リスト＞
10 ＜相互関係記号＞	:=:((“≫”＜相互関係名＞“≫”)	(“≪”＜相互関係名＞“≪”)	(“≪”＜相互関係名＞“≫”))
11 ＜相互関係相手先リスト＞	:=:＜相互関係相手名＞ (“,”＜相互関係相手名＞)*
12 ＜相互関係名＞	:=:(≪ mp ≫ | ≪集約≫ | ≪汎化≫ | ≪特化≫ | ≪実値化≫ | ≪引用≫ | ＜その他の相互関係名＞)
13 ＜相互関係相手先＞	:=:(＜フレーム番号＞“:”＜フレーム名＞)$^{?}$(“[”＜スロット番号＞ (“-”＜サブスロット番号＞)$^{?}$“]”)$^{?}$
14 ＜階層構造記述子＞	:=:((＜ mr ＞＜ sp ＞“|”)	(＜ mr ＞“||”))
日本語文	(自然言語としての日本語の単語や文の種類を推奨するあくまで原則的な規則．)
15 ＜日本語文＞	:=:(≪自然言語としての日本語の文法のままで定義・記述する文≫ | ＜拡張日本語文＞ | ＜注釈文＞)
16 ＜拡張日本語文＞	:=:≪式 (数式，計算式，化学式等) ≫	＜和文単語＞	≪日本語文や式等に変換可能な表現≫
17 ＜和文単語＞	:=:(＜属性名＞ | ＜フレーム名＞ | ＜相互関係名＞ | ＜相互関係相手先＞ | ＜ファセット記号＞)
18 ＜属性名＞	:=:≪多くの場合は名詞を使うが，制約条件等を書く場合には日本語文でも良い．ドメインユーザ判断．≫
19 ＜フレーム名＞	:=:≪多くの場合は名詞を使うが，振舞いのフレーム等の場合には日本語文でも良い．ドメインユーザ判断．≫
20 ＜注釈文＞	:=:＜サブスロット要素特定子＞＜ sp ＞≪任意の日本語文や 和文単語を書いて良い．≫
21 ＜ファセット記号＞	:=:“fn”＜ファセット番号＞ (fn は facet number の略．要素種類の分類記号)
22 ＜ファセット番号＞	:=:≪ OONJ の離散単位の種類表に記載されている種類に対応する番号 (**f**acet **n**umber) ≫
特別なフレームやスロット	(二つは規則というより右辺が左辺の標準スタイルで提供する具体的な種類を示す．詳細省略．)
23 ＜対象世界共通要素フレーム群＞	：＜初期/境界条件フレーム＞，＜共通属性フレーム＞，＜シナリオフレーム＞，・・・
24 ＜フレームヘッダスロット＞	：＜フレーム内共有属性スロット＞，＜相互関連スロット＞，＜ mp コントローラスロット＞
要素特定子	相互関係を付与するための要素を特定する仕組みや記号を指す．
25 ＜フレーム要素特定子＞	:=:＜フレーム番号＞＜ sp ＞ $^{+2}$ ＜ファセット記号＞
26 ＜スロット要素特定子＞	:=:＜スロット番号＞＜ sp ＞ $^{+2}$ ＜ファセット記号＞
27 ＜サブスロット要素特定子＞	:=:＜ sp ＞“-”＜サブスロット番号＞＜ sp ＞＜ファセット記号＞
28 ＜フレーム番号＞	:=:≪各フレーム単位の一連番号で，通常は 1 から始まる昇順の整数≫
29 ＜スロット番号＞	:=:≪各スロット単位の一連番号で，通常は 1 から始まる昇順の整数≫
30 ＜サブスロット番号＞	:=:≪各サブスロット単位の一連番号で，通常は 1 から始まる昇順の整数≫
記述位置指定子	記述位置を指定する仕組みや記号を指す。
31 ＜ rt ＞	:=:≪その右側の要素を右詰め (右寄せ) して記述する．≫
32 ＜ lf ＞	:=:≪次の離散単位に移り，その左先頭に戻る．フレーム間の場合は任意の間隔を空ける．≫
33 ＜ sp ＞	:=:≪全角一文字空白，space ≫
34 ＜ mr ＞	:=:≪直上のサブスロットに「階層構造記述子」または NJ 単文の先頭があればそこまで右寄せ．≫

拡張 BNF 記号

(1) ＜・・・＞はケット (アングルドブラケット) で，要素の内部を展開すべき/展開可能な離散要素であることを表す．
(2) ≪・・・≫はギュメ (ギルメット) で，内部要素の展開は完了した離散要素であることを表す．
(3) ＜・・・＞^{n} は n 個 (n ≧ 0，n=1 は省略可) の要素並びを表す．　(4) ＜・・・＞$^{+n}$ は n 個以上の要素の並びを表す．
(5) ＜・・・＞* は 0 個以上の要素並びで，＜・・・＞$^{+0}$ と同等．　(6) ＜・・・＞$^{?}$ は 0 個/1 個 (無し/有り) の要素を表す．
(7) “|” はこの記号の左右にある離散単位の片方の選択を指示する．OONJ では同一記号で集約や付置属性も表す．
(8) ダブル・クオテーション “” は挟まれる文字列や記号等をそのまま記すことを指示する．