Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

ビューポイント DSL を用いたシステム仕様記述に関する考察

田中明†　　高橋修††

ビューポイントに基づく仕様記述方式はシステムの規模と複雑さの増大に対応出来るシステム仕様記述方法の一つである．その仕様記述には自然言語，UML，DSL など各種記述方式が考えられるが，テキスト型のビューポイント DSL を用いた場合の記述について UML 記述の場合との比較を行いその特徴について評価した結果を報告する．

Study on systems specification with viewpoint textual domain specific language

Akira Tanaka† and Osamu Takahashi††

This study introduces viewpoint based systems specification techniques, based on RM-ODP international standard, and focuses on the case where textual DSL is used to represent viewpoint specification. Difference among specifications with natural language, with UML, and with Domain Specific Language (DSL), especially textual DSL, is investigated.

1. はじめにa

複雑さに対応する考え方の一つに divide and conquer（分割して統治する）方式があり，本報告ではビューポイントにより分割を行うシステム仕様記述方式を取り上げる．システム仕様記述方式には自然言語に基づく方式，UML に基づく方式，グラッフィック型（Graphical）またはテキスト型（Textual）のドメイン特化言語（DSL）に基づく方式，更には形式記述言語を用いる方式等がある．本報告では試作したテキスト型のビューポイント DSL を用い仕様記述を行い UML で記述した場合と比較することで，その適用領域や課題を考察する．

2. 関連技術の現状

2.1 ビューポイントに関する関連技術及び標準

RM-ODP（Reference Model of Open Distributed Processing）は ISO/IEC 及び ITU-T の国際標準仕様であり，オブジェクト指向とビューポイントに基づいた分散システム記述の参照モデルである．参照モデル標準開発後 Enterprise Language 標準[1], Use of UML for ODP system specifications 標準[2]など継続して開発されてきた[3]．IEEE では IEEE 1741[4]によりビューポイントを一般的に規定する際のガイド標準も作成されている．

RM-ODP のビューポイントに基づくシステム設計については 1994 年に藤長，加藤，鈴木 3 氏による「ODP ビューポイントに基づく分散システムの設計法とその通信システムへの適用」[5]や「ODP ビューポイントに基づく分散システム設計法の検討」[6]がある．当時の RM-ODP 標準化状況については[7]に，また 2009 年の DPS 研究報告「ビューポイントに基づくシステム仕様記述に関する考察（１）」[8]がある．

2.2 モデル記述記法

モデル記述記法のうち RM-ODP のビューポイントに基づくシステム記述と合わせて利用される可能性の高い手法を以下にあげる．

・自然言語（オフィス文書、Web ページなど）

自然言語を用い文章，表，グラフ類など組み合わせ仕様を記述する方式で，文書形式の標準等存在するが仕様記述が十分な構造を持たず厳密な記述とはみなせない．

・UML 標準[9]

標準モデリング言語として広く利用されている．例えば Use of UML for ODP system specifications が国際標準となっており，幾つかの UML ツール向けのプラグインが公開されている．他にも OMG の UPDM 仕様[10]等幾つもの UML 拡張仕様が利用されている．

・MOF 標準[11]

†, †† 公立はこだて未来大学 Future University - Hakodate

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

MOF 標準に基づくモデルは通常概念モデルやメタモデルと呼ばれるモデル記述のためのモデルである. 広く知られた MOF サブセット実装例に eclipse EMF[12]がある.

・Domain Specific Language (DSL)

ドメインに特化した言語を設計利用する方式であるが通常幾つかにカテゴリ分けされる. 母体となるプログラミング言語を持つ場合と持たない場合を区別し Internal DSL と External DSL という分類，ダイアグラム表現を指向したものとテキスト表現を指向したものを区別した Graphical DSL と Textual DSL という分類がある.

本報告では Textual DSL の設計適用を中心とし, 報告者が過去に実施した UML 記述との比較に基く考察を行う.

3. ビューポイント DSL の設計

3.1 概要

ビューポイントに基づくシステム記述用にビューポイント DSL を設計した（63 rules）. ベースとして利用したのは ANTLR[13]に基づく Textual DSL ワークベンチである eclipse Xtext[14]である. 分量の関係から試作した文法の先頭部分の一部掲載に止める. 以降でも UML 記述と対比する形でその記述の一部分を提示する.

```
grammar org.xtext.example.ODP04 with org.eclipse.xtext.common.Terminals
generate oDP04 "http://www.xtext.org/example/ODP04"
Model:
    (elements+=ModelElement)* ;
ModelElement :
    Type | Object | Spec ;
Type:
    SimpleType | ComplexType | Enumeration ;
SimpleType:
ComplexType:
Enumeration:
Spec:
    EV_spec | IV_spec | CV_spec | NV_spec | TV_spec | Corr_spec ;
FQN:
Object:
    objectType=('EV_Object'|'IV_Object'|'CV_Object'|'NV_Object'|'TV_Object') name=ID ('extends'
        (properties+=Property)*
        ('{' stateMachine=StateMachine '}')?
        (interfaces+=Interface)*
        ('<' (innerObjects+=Object)+ '>')?
    '}' ;
Property:
    'property' name=ID ':' type=[ModelElement|FQN] (many?='[]')? ;
```

図 1　ビューポイント DSL 文法（一部）

4. ビューポイントに基づくシステム仕様記述

4.1 概要

ビューポイントに基づくシステム仕様記述とは，対象システムを複数のビューポイント（観点・視点）から記述する方式であり，各ビューポイント記述ではそのビューポイントの関心事以外の情報を捨て去ることで関心事にフォーカスした記述を行う. RM-ODP では Enterprise，Information，Computational，Engineering，Technology の 5 種類がある.

4.2 Use of UML for ODP system specifications 標準を用いた仕様記述例

Use of UML for ODP system specifications 標準は UML の拡張機能である Profile 機構を用い RM-ODP の各ビューポイント概念を UML ツールで記述出来るようにする仕様で, 仮想的な地域診療所予約管理システム記述例を以下に示す. この UML 記述では，仕様全体を一つの Package とし，その内部に各ビューポイント仕様を含む Package とビューポイント仕様間の相関を規定する相関仕様の Package 及び対応ビューポイント仕様への依存関係を表現している.

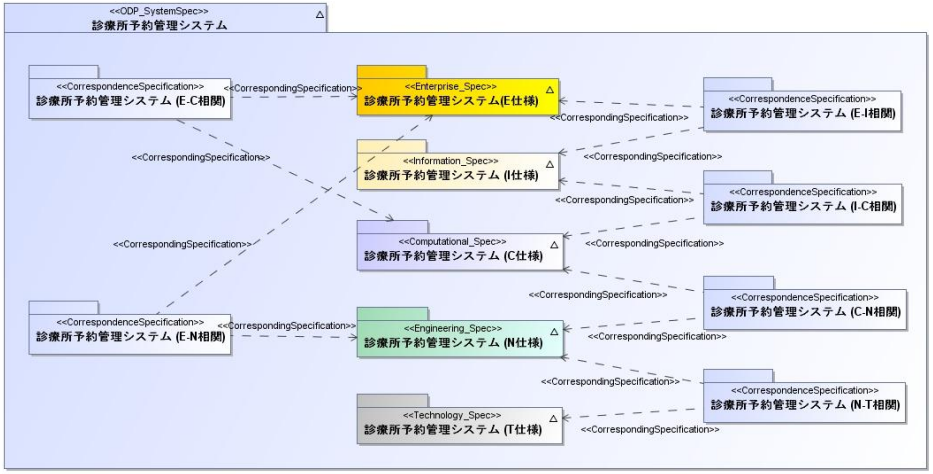

図 2　システム仕様全体像記述例

図 2 は各ビューポイント仕様とそれらの相関関係仕様を UML Package で平面的に表現している. これをビューポイント DSL で記述（部分）すると，次のようなビューポイント仕様の連続記述と相関記述という直線的記述となる.

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

```
// エンタプライズ仕様 //
enterprise Clinic_EV {
// インフォメーション仕様 //
information Clinic_IV {
// コンピュテーショナル仕様 //
computational Clinic_CV {
// エンジニアリング仕様 //
engineering Clinic_NV {
// テクノロジ仕様 //
technology Clinic_TV {
}
// 相関仕様 //
EI_correspondence Clinic_EI
```

図 3　DSL によるシステム仕様全体像記述（一部）

(1)　Enterprise ビューポイント仕様

このビューポイントにおける仕様の構造は概略次のようになる．UML での Package 構造を参考にシリアライズした（UML 図は省略）．

```
// エンタプライズ仕様 //
enterprise Clinic_EV {
// 適用分野 //
    EV_FieldOfApplication "クリニック運営"
// コミュニティコントラクト //
    EV_CommunityContract ABC_Clinic {
// コミュニティの目的 //
        EV_Objective "地域医療に貢献するクリニックの実現"
// コミュニティ定義 //
        EV_Community LocalClinic {
// エンタプライズオブジェクト（含状態遷移）定義 //
            EV_Object Person {
// ロール定義 //
            EV_Role Patient
            EV_Role Receptionist
            EV_Role Examiner
            EV_Role ClinicSystem
// オブジェクトとロールの関連定義 //
            EV_Object Person fufills Patient EV_Role
// ビジネスプロセス定義 //
            EV_Process ExaminationProcess {

// 診療ポリシー定義 //
            EV_PolicyEnvelop TreatmentPolicy {
        }
```

図 4　DSL によるエンタプライズ仕様概要

図 5 は患者，受付，診療所予約管理システムの間の貸出プロセス仕様を記述したもの．UML の Activity 図を利用．

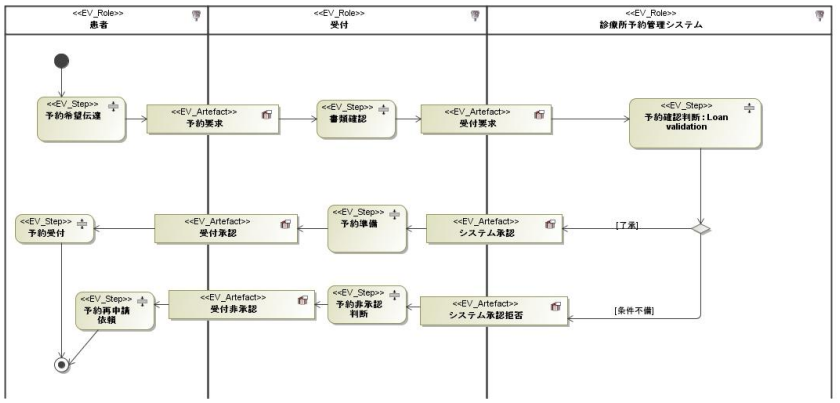

図 5　予約プロセス記述例

ビューポイント DSL で記述すると用意した状態機械記述能力を用い次のような記述となる．これはプロセスを Activity 図のレーン毎に分割記述していることに相当．

```
// ビジネスプロセス定義 //
    EV_Process ExaminationProcess {
        EV_Role Patient {
            start Normal {
            EV_Step StartOfTreatment {
            EV_Step Examination {
                incoming StartOfTreatment
                outgoing EndOfTreatment
                outgoing Treatment
            }
            EV_Step Treatment {
            EV_Step FinalExamination {
            end EndOfTreatment {
                incoming Examination
                incoming FinalExamination
            }
```

図 6　DSL によるプロセス記述例

Enterprise Object と Role の可能な対応付けは UML の場合 FulfilsRole という関連を用い Class 図で記述するが，ビューポイント DSL では次のような記述となる．

```
EV_Object Student FulfilsRole Patient EV_Role
```

図 7　DSL によるオブジェクトとロールの関連記述例

更にポリシー他についての記述もあるが紙面の関係から省略．

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

(2) Information ビューポイント仕様

Information Object の静的，動的，不変スキーマを記述する．静的スキーマは Information Objects のある時刻における値のセットであり UML の Object 図で記述，動的スキーマは Information Object の状態遷移であり UML の State Machine として記述，そして不変スキーマは常に成立する条件であり UML の Class 図と OCL による制約記述により表現する．

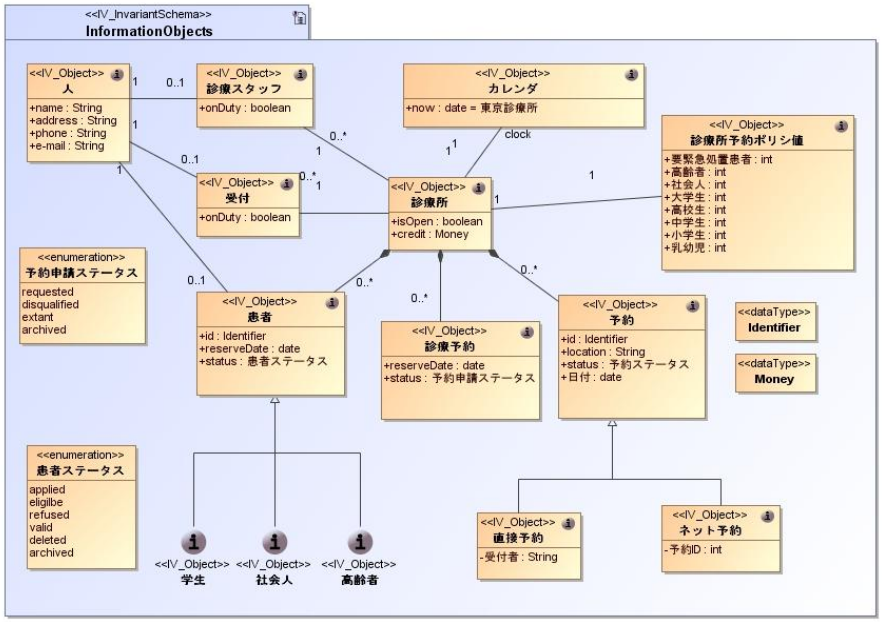

図 8 不変スキーマ記述例

図 8 は不変スキーマの例を示す．ここでは Information Object の識別とそれらの間の関係を記述している．図 9 のビューポイント DSL での記述では，不変スキーマを Class 図に相当する記述で，またこれに基づき動的スキーマを Information Object の状態遷移で記述する．図に含めていないが，静的スキーマでは不変スキーマのスナップショットとして幾つかの時刻における Information Object のインスタンスを記述する．多重度や OCL を用いた制約記述は DSL 文法規定自体とツールが備えるメカニズムを利用し外部的に制約チェックロジックを与えることで実現．

```
// インフォメーション仕様 //
information Clinic_IV {
    invariant schema Clinic_Information_Invariant {
        IV_Object Patient {
            property id : Identifier
            property reservationDate : Date
            property status : Status
        }
        IV_Object Reservation {
            property id : Identifier
            property location : String
            property status : ReserveStatus
            property date : Date
        }
        IV_Object WebReservation extends Reservation {
            property webReservationID : Identifier
        }
    }
    dynamic schema Clinic_Information_DynamicSchema {
        events
            somethingWrong
            treatmentDone
        end
        commands
            visitClinic
        end
        state initial actions { visitClinic }
            somethingWrong => underTreatment
        end
        state underTreatment actions { visitClinic }
            treatmentDone => initial
        end
    }
}
```

図 9 DSL によるスキーマ記述例

(3) Computational ビューポイント仕様

Computational Object とその提供及び要求インタフェースを Template として識別し，必要な DataType やインタフェースの Signature を記述する．Object 間のインタラクションについては UML Activity/Sequence 図を利用し記述する．RM-ODP の特徴として Computational Object はシステムの分散形態を意識しない論理的なものとなる．図 9 に代表的な Computational Object の構造記述例を示す．UML の Component 図を利用し，内部要素，インタフェース（UML Port），シグニチャ（UML Interface）を用い構造を表現している．

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

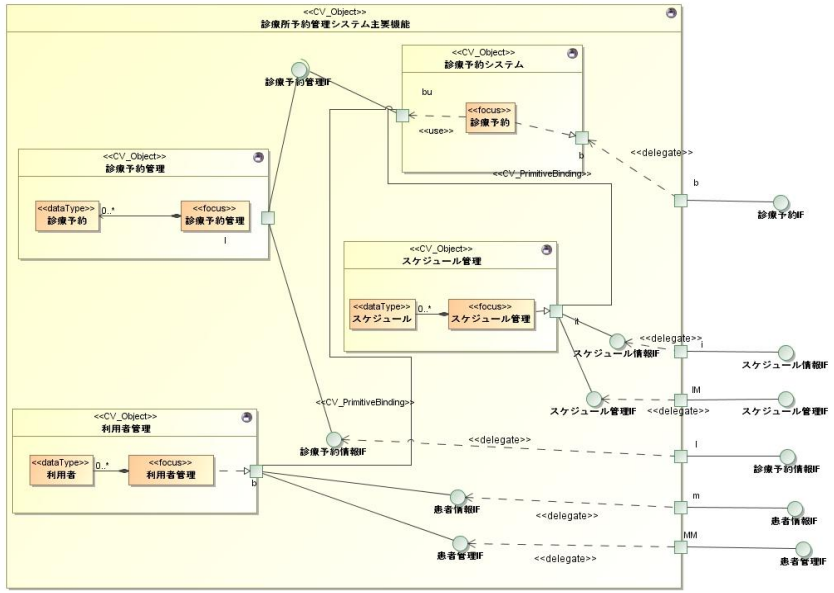

図 10　診療所予約管理システム主要機能内部構造記述例

ビューポイント DSL では次のような記述を行う．

```
// コンピュテーショナル仕様 //
computational Clinic_CV {
// コンピュテーショナル仕様データタイプ定義 //
    type TimeSlot {}
// シグニチャ定義 //
    CV_OperationInterfaceSignature AddNewSchedule {}
    CV_OperationInterfaceSignature MakeReservation {
        parameter in reservation : TimeSlot
        parameter out result : Boolean
    }
// コンピュテーショナルオブジェクト定義 //
    CV_Object ClinicReservationManagement {
// 内部オブジェクト定義 //
        <   CV_Object ReservationSystem {}
            CV_Object ScheduleManagement {
                CV_OperationInterface AddSchdule {
                    providing AddNewSchedule
                }
                <   CV_Object ScheduleManagementCore {}
                    CV_Object Schedule {}
```

図 11　DSL によるコンピュテーショナル仕様記述例

(4)　Engineering ビューポイント仕様

Engineering ビューポイントでは Computational Object の機能を複数 Node に分散配備することや，ミドルウェアが実現する各種機能の識別配備，また分散処理支援機能とそのメカニズムなどを記述．図 10 は最もハイレベルな Node 構成記述を示す．

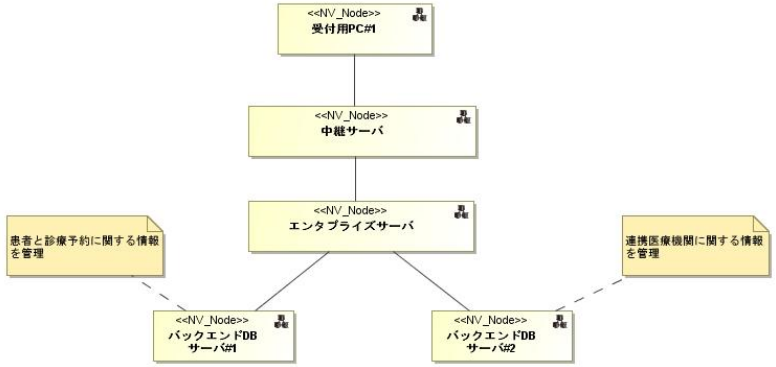

図 12　診療所予約管理システム Node 構成記述例

ビューポイント DSL で記述すると次のようになる．

```
// エンジニアリング仕様 //
engineering Clinic_NV {
    node PCforReceptionist {
        link toServer : PCServer
        NV_OperationInterfaceSignature Login {
            parameter in UId : Identifier
            parameter in PWD : String
            parameter out result : Boolean
        }
        NV_Object LoginObject {
            property name : String
            property UId : Identifier
            property PWD : String
            NV_OperationInterface Login {
                providing Login
            }
        }
    }
    node PCServer {
        link toPC : PCforReceptionist
        link toBackEnd : PCBackEnd
        NV_SignalInterfaceSignature SecurityAlarm {}
        NV_Object AlarmObject {}
    }
    node PCBackEnd {}
```

図 13　DSL によるエンジニアリング仕様記述例

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

(5) Technology ビューポイント仕様

Technology ビューポイントではシステム構成要素となるソフトウェア，ハードウェア，ネットワークに用いる具体的な製品等を記述する．図 14 に例を示す．

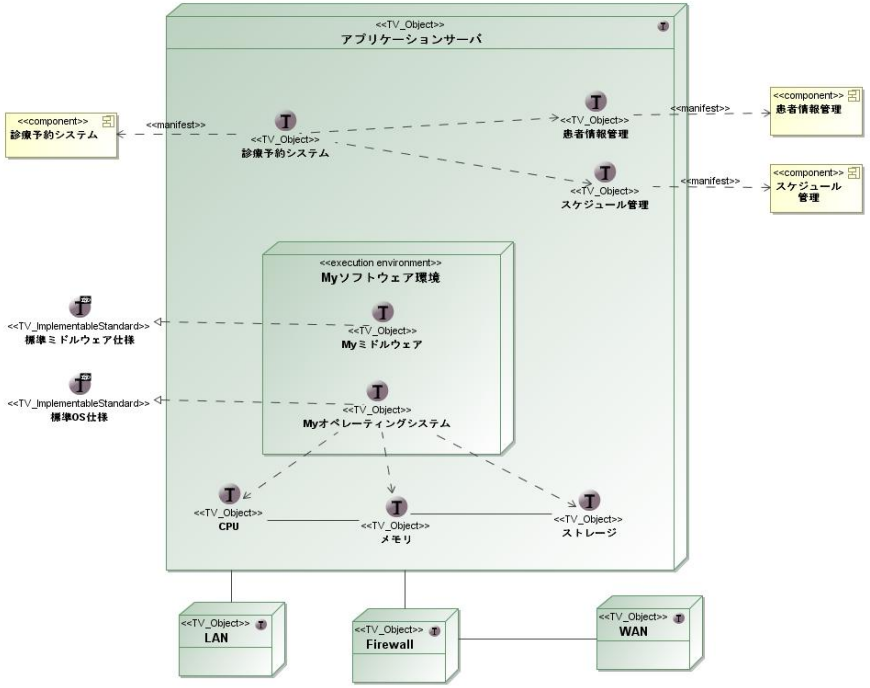

図 14 診療所予約管理システム構成要素記述例

ビューポイント DSL での記述を図 15 に示す．

```
// テクノロジ仕様 //
technology Clinic_TV {
    TV_Object PC01 {
        implementing Clinic_NV.PCforReceptionist.LoginObject
    }
    TV_Object PCServer03 {
        implementing Clinic_NV.PCServer.AlarmObject
    }
    implementable standard JEE
    implementable standard SQL
    IXIT "JVM verion higher than or equel 1.5"
}
```

図 15 DSL によるテクノロジ仕様記述例

(6) Correspondence 仕様

各ビューポイント仕様の間にある相関関係を Correspondence 仕様として記述する．例として Information ビューポイント仕様と Computational ビューポイント仕様の二つを取る．この例では診療予約 Information Object と診療予約 Computational Object の間にある相関を規定しており，

・診療予約 Information Object と診療予約 Computational Object には診療予約日程他の対応関係が存在する

・Information ビューポイントでの診療予約インスタンスが Computational ビューポイントでの診療予約管理オブジェクトが管理するオブジェクトと対応する

などを，図 16 の例のように CorrespondenceLink のステレオタイプを持つクラスに Correspondence の OCL 記述を制約として加えた形で記述できる．

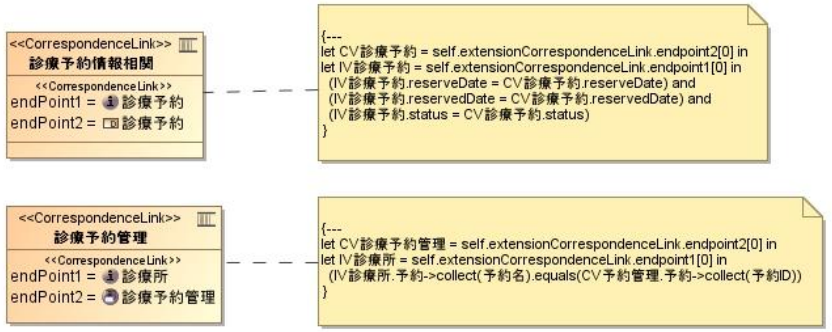

図 16 診療所予約管理システム Correspondence 仕様記述例

```
// 相関仕様 //
correspondence IC_Correspondence_ReservationDate {
    information {
        endpoint Clinic_IV.Clinic_Information_Invariant.Reservation
    }
    computational {
        endpoint Clinic_CV.TimeSlot
    }
    description "Resrvation in Information Viewpoiont corresponds to TimeSlot in Computationa Viewoint"
}
```

図 17 DSL による Correspondence 記述例

5. UML 記述とテキスト型 DSL 記述の比較検討

5.1 UML 記述

UML には標準メタクラスのサブクラス相当を stereotype として規定し，それらを標

Vol.2010-SE-168 No.13
Vol.2010-EMB-17 No.13
2010/6/2

準記法に無理なく取り込むことが出来る UML Profile という拡張機能がある．手順的には当該関心領域（ドメイン）のメタモデルを作成し，現れる概念をどの UML 要素に対応付けるかを決める．UML 記述の特徴は，グラフィカル（ダイアグラム）記法であること，多数のツールが利用できること，モデルデータ交換形式も標準化されていること，等があげられる．記法を熟知している人同士の場合，意思疎通の手段として有効．

5.2 テキスト型 DSL 記述

テキスト型 DSL の場合，Internal DSL の場合においても，テキストやソースコードで表現したいことをより簡潔に表現することが目標で, その DSL 記述のプログラムによる解釈・処理も視野に入っている．そのためには理論的基盤を備えたワークベンチが必要となる．今回は Xtext を利用したが他にも幾つかの選択肢（MPS[15], M[16], TCS[17]他）がある．

UML 記述とテキスト型 DSL 記述比較

(1) モデル開発プロセス要素

仕様記述のステップ毎に相違点を検証する．参考にグラフィカル DSL の項を追加．

表 5.1 モデル開発プロセス要素の比較

	UML/UML Profile	Textual DSL (Xtext)	Graphical DSL（GMF）
メタモデル作成	MOF モデル：UML クラス図（subset）定義	Xtext 文法定義（eclipse）	ecore モデル定義(eclipse)
ホスト言語との関連付	UML Profile を定義	対象外	対象外
モデル作成	UML ツール	生成 Textual Model Editor（eclipse）	生成 Graphical Model Editor（eclipse）
妥当性チェック	UML（OCL）	OCL 相当	OCL（OCL 相当）
モデル交換データ形式	XMI	XMI データ生成可能	Ecore（XML）
モデル変換（参考）	UML ツールの外側 (機能を持つツール有)	Xtend で可能	別ツールと組み合わせ（QVT, ATL, Xtend）
コード生成（参考）	UML ツールの外側 (機能を持つツール有)	Xpand で可能	別ツールと組み合わせ（QVT, Xpand）
データ保存	UML ツール固有	Xtext 形式（ecore）	GMF 形式（ecore）
要求されるスキル	Meta-modeling, UML, UML Profile, XML	言語定義，Xtext，Xtend, Xpand, XML, Java	Meta-Modeling, GMF, EMF, XML, Java

(2) ビューポイント記述

ビューポイントの規定はビューポイント毎の概念に基づくビュー規定を意味する．従って各ビューポイントに含まれる概念規定等がまず始めに必要であり，それはメタモデル記述であるため，最適な記述手法は MOF（記法は UML Class 図）となる．ビューポイント概念規定を終えれば，次は仕様の構造化を検討する段階となる．そこでは Textual DSL の文法定義利用が効果的である．UML の場合 UML Profile 設計を行い，その際及びその後にモデル構造（一般的に木構造）を決めることになる．

(3) モデル記述

モデルの作成において UML/UML Profile と DSL で大きく異なるのが汎用を目指して規定されている UML 標準記法の存在である．例えば状態遷移やアクティビティの記述には標準記法が規定されているため単にこれらを利用することになる．これに対し DSL で状態遷移やアクティビティの記述が必要になればそれらに対応したメタモデルや文法を DSL 設計者が作成する必要がある．この段階で作成者による差が導入される可能性が高いため，シェア出来る共通ライブラリ化などが必要となる．

(4) 習得期間

比較表の要求されるスキル欄にある通り，どの方式であってもそれぞれ習得すべきスキルセットが存在する．UML と XML を習得済みであれば UML Profile 方式が入り易いはずであるが，この用途で利用できるまで UML を理解・使いこなせるには通常かなりの時間を必要とする．これに対し Textual DSL では Meta-modeling の替わりに言語定義から入るため，UML の知識を持たずとも利用出来，同等の作業ながら習得期間は比較的短いものと想定される．

(5) スケーラビリティ

人間の視野・認識範囲で，仮に A4 の紙に収まる程度の数のグラフィカル要素（経験からは 30-40 要素程度）を扱うのであれば理解・コミュニケーション上有効である．しかし要素数が例えば 100 を超えるようになると，2 次元図であるが故に一般的には詳細の理解が困難となってくる．これに対し Textual DSL の場合，仕様記述が直線的でありモデル要素の追加なども比較的状況を把握した変更作業が可能となる．

(6) 相互運用性

相互運用性には二種類ある．一つは UML Profile の場合の UML ツール間でのデータ交換という意味での相互運用性，及びテキスト型 DSL ツール間でのデータ交換という意味での相互運用性．もう一つは UML Profile を使い作成したデータとテキスト型 DSL で作成したデータとの間の相互運用性．前者のうち，UML Profile については OMG で活動が行われている．テキスト型 DSL についての標準化活動はないが，モデル変換という文脈でみると Atlantic Zoo [18]といった研究レベル活動があり相互運用性のベースになり得る．後者については，UML Profile 規定のベースとなったメタモデルに立ち返ると前記研究レベル活動に基づく変換も原理的には可能である．但し，テキスト型 DSL の場合ダイアグラム情報が含まれないため，意味的な相互運用性に止まる．

(7)　モデル駆動開発（モデル完成後の開発活動）

原理的には差は無いが実用的に次の差がある．UML Profile に基づくモデルは UML 要素を含んでいるため，入力データ解析に UML メタモデルを反映する必要がある．テキスト型 DSL に基づくモデルではモデル要素の数がメタモデル要素数に近いため，入力データ解析が比較的容易に出来ることになる．Model to Model 変換と Model to Text 変換自体については類似の仕様・ツールを使うため，ツールチェーン全体として類似の動作結果が期待できる．

(8)　UML Profile と DSL の複合利用

大きな枠組みは UML Profile を利用し UML ダイアグラムで表現し，幾つもの関心事について DSL を利用し個別に記述しそれらを統合するような使い方が考えられるが，UML・DSL の各ツールそれぞれの立場から相手のデータを直接管理出来ず維持拡張の点で現時点では好ましくない．UML Profile 同士や DSL 同士の複合は対象領域が独立であれば何ら問題ないが，重複や強い関連がある場合その部分に継承関係を導入するか独立させるような対策が必要．但し一つでも標準が含まれると即座の対策は困難．

5.3 考察

(1)　トップダウンとボトムアップ

モデリングは仕様記述のトップダウンアプローチでだが，Ruby on Rails/Grails などのプラットフォーム側から抽象レベルを上げるボトムアップアプローチもみられる．両者が中間点でうまく接続できるようモデリング側の配慮が必要．

(2)　仕様記述・詳細化手順

利用者の立場から仕様記述に入り易いのは自然言語そして UML/UML Profile である．まず自然言語で要件や諸条件を書き出し，UML で仕様の大枠を固め，そしてテキスト型 DSL でスケーラビリティに対応した詳細記述を実施するという手順が有効である．

6. 結論と今後の課題

6.1 結論

仕様規模が大きくなる場合には，ダイアグラム中心の UML・DSL では記述した仕様の理解・維持管理が容易ではなくなる．この点，テキスト型 DSL に基づく仕様記述はシンプルである分問題も少なく，モデル駆動ソフトウェア開発の一つの手法として効果が期待出来る．

6.2 今後の課題

DSL 定義を行う際に比較的頻繁に現れる要素があり，これらについて Component として積極的に利用できるよう公開ライブラリ化するような仕組み作りが課題．

今回の報告ではビューポイント仕様の記述法となる UML Profile とテキスト型 DSL の比較に主眼を置いたが，今後は Model to Model 変換及び Model to Text 変換を含め，具体的なプラットフォームへの展開検討が課題となる．

6.3 謝辞

本研究に利用した各種オープン仕様・オープンソースソフトウェアの開発者及び提供団体，そして分散システムの仕様記述手法に関しご指導を頂いた，公立はこだて未来大学の高橋修先生に感謝致します．

参考文献

[1] ITU-T Recommendation X.911 | ISO/IEC 15414, Information technology - Open distributed processing - Reference model - Enterprise language
[2]ITU-T Recommendation X.906 | ISO/IEC 19793, Information technology - Open distributed processing - Use of UML for ODP system specifications
[3] RM-ODP Resource Site, http://www.rm-odp.net
[4] IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
[5] 藤長昌彦,加藤聰彦,鈴木健二:ODP ビューポイントに基づく分散システム設計法とその通信システムへの適用,マルチメディア通信と分散処理 63-13 (1994)
[6] 藤長昌彦,加藤聰彦,鈴木健二:ODP ビューポイントに基づく分散システム設計法の検討,情報処理学会第 47 回全国大会 (1993)
[7] 中川路哲男, 田中　明, 浅野正一郎: "開放型分散処理の標準化の概要", 電子情報通信学会誌, Vol.77, No.3, pp.277-287, (1994)
[8] 田中　明, 高橋　修:ビューポイントに基づくシステム仕様記述に関する考察（１）, マルチメディア通信と分散処理, DPS-140-5 (2009)
[9] ISO/IEC 19501, Unified Modeling Language
[10] Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defense Architecture Framework (MODAF) [UPDM], OMG
[11] Meta Object Facility (MOF™), OMG
[12] Eclipse Modeling Framework, The Eclipse Foundation, http://www.eclipse.org/modeling/emf/
[13] ANTLR, ANother Tool for Language Recognition, http://www.antlr.org/
[14] Xtext - a programming language framework, http://www.eclipse.org/Xtext/
[15] Meta Programming System, http://www.jetbrains.com/mps/
[16] The Microsoft code name "M" Modeling Language Specification, http://msdn.microsoft.com/en-us/library/dd285282.aspx
[17] Frédéric Jouault et al: TCS:: a DSL for the specification of textual concrete syntaxes in model engineering, Proceedings of the 5th international conference on Generative programming and component engineering, pp. 249-254 (2006)
[18] AtlanMod Zoo, http://www.emn.fr/z-info/atlanmod/index.php/Zoos