Vol.2015-SE-189 No.2
2015/7/22

UML によるディオンティックコンセプトに基づいた ODP エンタープライズ言語の表現

宮崎比呂志[†1]

ODP(Open Distributed Processing)エンタープライズ言語は、仕様記述をすべきシステムの目的やポリシーを記述するために用いられるものである.また、ODP エンタープライズ言語はシステムが動作する環境に付随する制約を記述するものである.エンタープライズ仕様は、システムの技術的な詳細というよりも概念レベルでの記述を行うものであるため、詳細な仕様技術というよりもステムの責務や規約に着目するものである.最新のエンタープライズ言語は、責務や他のディオンティックコンセプトを表すことができるように改訂し、ISO/IEC 19793 として発行された.本稿では、ディオンティックトークンを用いたエンタープライズ言語の基本的な概念を示すと共に、実際にディオンティックトークンを用いた仕様記述へ適用実例を紹介する.

Representing ODP Enterprise Language based on the Deontic Token in UML

HIROSHI MIYAZAKI[†1]

1. はじめに

近年、情報システムの量及び重要性が増してきており、完全なシステム仕様を記述する方法が重要となり、特に、仕様の正しさを検証、正しくない仕様をチェックする必要性が増してきている.したがって、より完全なシステム仕様を与えることにより、ビジネス要件を正しく反映した IT ソリューションを構築するようにすることが重視されてきている.

システム仕様を記述するために Reference Model for Open Distributed Processing (以下 **RM-ODP**)が ISO 標準として開発されている.これは、エンタープライズ アーキテクチャの１種であり、各ステークホルダーの観点に基づいてシステムの仕様を記述するものである.特に、RM-ODP では、エンタープライズビューポイント、インフォメーションビューポイント、コンピューテーショナルビューポイント、エンジニアリングビューポイント、テクノロジービューポイントから構成され、ビューポイント毎に仕様を記述するものである.これにより”関心事の分離”を図るものである.これは国際標準として、ISO/IEC 15414 及びその UML 表現を規定した ISO 19793 が開発された.しかしながら、旧来の RM-ODP は、単にシステムの動作、非機能に着目して記述しただけのものであった.しかしながら、SOA やクラウドの出現により、利用者の役割にフォーカスを移した、より完全なシステム仕様を与えることの重要性が増してきた.

そのために、利用者の役割を明示し、それによりシステムとの相互作用によるシステム仕様記述法方が必要とされてきている.[1,2].これは、ディオンティックコンセプト[3]と呼ばれる考えに基づきモデル化するものである.例えば、ディオンティックコンセプトの具体的な例としては、責務(Obligation),許可(Permission), 禁止(Prohibition)としてシステムに与える要件を形式化して記述するものである.

エンタープライズ言語は Reference Model for Open Distributed Processing(RM-ODP)で最初に導入され、ODP システム仕様のキーパートを構成する.システムの目的、方針、環境上の制約を別々のビューポイントに分離することによって、ビジネスを把握することが容易になる.そのようにすることにより、新しいシステムの詳細設計及び既存システムの開発を把握するための重要な基盤となる.しかしながら、システムが何をすべきかにフォーカスすることは、ディオンティックロジックを扱うことになる.ディオンティックロジックは基準や予想を扱うことができるようにする.もしくは仕様化された動作を実行するための obligation, そのような動作を実行するための permission 及び動作の prohibition である.したがって、仕様上で様々なオブジェクトに関連した obligation, permission, prohibition を扱うことができるようにし、さらにこれらがどのように処理されるかを表すことができなくてはならない.において、このような体系がトークンオブジェクトによっておのおののディオンティックアサーションを表現することに

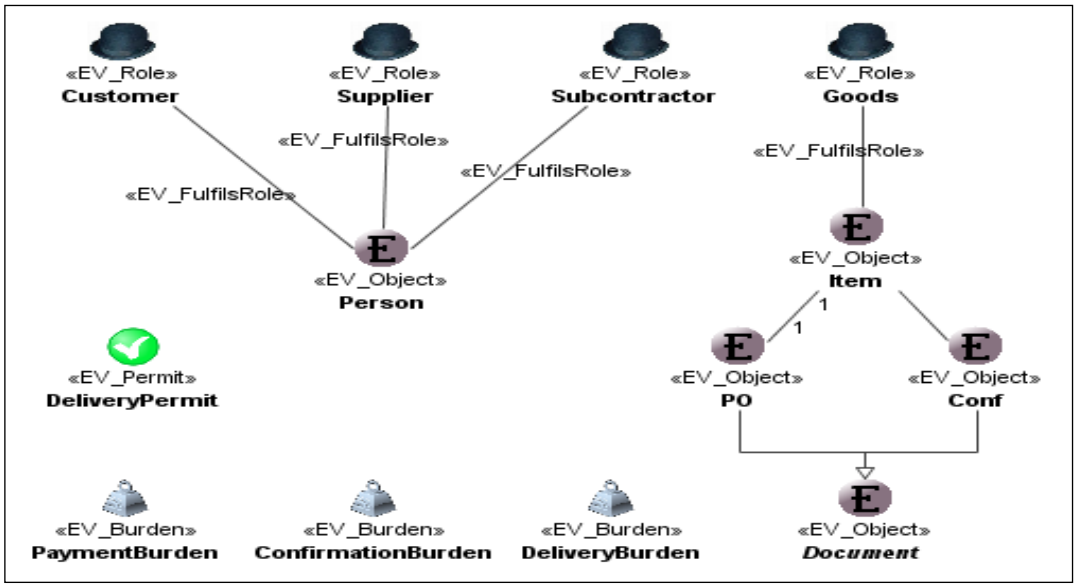

図 1　コミュニティー

よって実現できるかを示した.これらのオブジェクトは関与するドメインによって保持され、それらが保持している状況には動作をコントロールする.ディオンティック制約の仕様記述を導入することによるこの問題の扱いについては、のエンタープライズ言語を構築する方針として最初に示唆され､ISO 標準となっている.エンタープライズ言語は抽象概念のフレームワークを提供するが、実際の仕様記述は具体的な表記法で表現される必要がある.ODP においては､1 つのそのような表現は ISO/IEC19793 の”Use of UML for ODP System Specification”で定義されている.

本稿では、システム技術が UML を用いてこれらのコンセプトを表現できるようにした成果について述べる.

2．エンタープライズ言語

ODP エンタープライズ言語は､仕様記述するシステムに適用する目的と方針に記述するために用いられた.それは、システムが用いられる環境に関与する制約も捕捉する.エンタープライズ仕様はシステムの技術的詳細というよりビジネス上の課題に関与するものであるので、システムの責務や規定に重点をおくものであり、必要とされる機能の詳細には重点はおいていない.

エンタープライズ言語は関与する関係者及びそれらの予期される動作を表現する.エンタープライズ言語の主要なものは、Community の概念である.これは、１つの目的を達するために協調して作用する複数のプレヤーを表す抽象オブジェクトである.Community はそれらのタイプを使って表され､参加する community ロールを使って表される動作を表現する契約である.これらのロールは特定の community において、定義された動作に適合するように規定された enterprise object によって満足される.

エンタープライズ言語は、accountability と responsibility を表す概念を含んで、システム動作のベースが accountability に遡れるようにする.そのようにすることはビジネスとテクニカルのギャップを橋渡しする.このような仕様の書き方の背景や言語の形態の動機は、詳細が述べられている.

実用的にするためには、このような抽象言語は容易に使用できる具体的概念によってサポートされていなくてはならない.

UML4ODP は OPD ビューポイント言語の表現が、UML に馴染んでいるアーキテクトやデサイナーに容易に使用できるように定義されている.UML4ODP はサポートされていて、ODP、UML のシステム設計者が仕様を書いたり、検証したら、プロトタイプすることができるようになっている.

3. モデリングアプローチ

RM-ODP はビューポイントを用いて複雑なシステム仕様を複数の独立な関心事の分離をするものである.これらのビューポイントは、ビューポイント間の要素や言語の対応関係を確立することによって、個々に完結した仕様を構成している.対応関係は､あるビューポイントの特定の名前が別のビューポイントの異なる名前によって参照されているものを表すこともある.対応関係はタイプレベルまたはビューポイント固有言語間もしくは表記間で確立される.

これらのビューポイントを確立させる目的は、異なるステークホルダーがシステムを設計／構築を可能な限り独立にすることができるようにすることである.あるステークホルダーに関係する設計は、対応関係がビューポイント間で確立しているので、他のステークホルダーに対しても重要になる.この対応関係は､異なるビューポイントが相互に補完しあうようにするものである.

ディオンティックプロパティを表すモデルとビジネスプロ

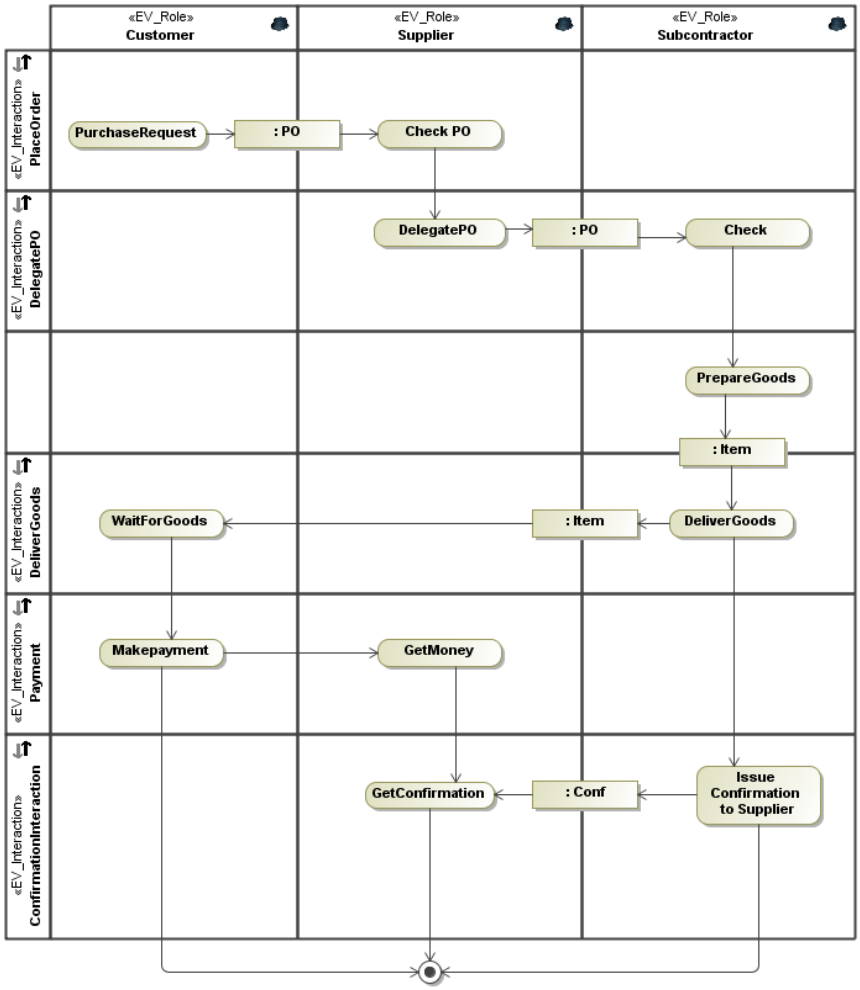

図２　アクティビティ図

セスを表すモデル間の関係は異なる.ディオンティクによる制約は一般に基本的なプロセスであり、してはならないこととなすべきことを区別する.このことは、表している動作は、リッチで複雑ことを表すが、異常／禁止／例外を正常系から区別する付加的な可能性がある.反対に、明確に表されていないどんなものでも禁止されると理解される許容される動作の表現である.スコープ内の仕様の複雑性を保つために、一般的なルールは一般的に特定の例外を識別することが好まれる.タイプレベルでの少数の制約は、多くの詳細の表現より、付加的な複雑性は少なくなる.結果として、アスペクト志向プログラミングで見出せるようなものに類似している基本的動作を装飾するメカニズムが必要とされる.広範なインターラクションのタイプがパターンへのマッチングとして認識されるところに適用できる一般ルールとして、ディオンティックコンセプトの制約を表現する.基本的もしくは拡張されたモデリングに対して異なるＵＭＬ図を使うことによって、この区別をつける.

４．売買システムのシナリオ

これに適合するためのタイプの例として、簡単な取引に関するシナリオを考えてみる.４つのロールが存在する最小限のコミュニティーを考える.図１.これらは、３つのアクティブな参加者を表すロールである.それは、顧客、製造者、下請け業者及び商品取引をする役割である.このコミュニティーは、発注、確認書、商品アイテム、許可と義務を表すトークンのような取引上のドキュメントからなる.

顧客の発注で始まり、商品の支払いで終了する主要な動作を考えてみる.図２

顧客は、製造者に発注し、その製造元は下請けに発注の実行を委譲するか決定する.下請けされる時には、下請け業者は商品を用意し、発送する.発送する時点において、下請け業者は製造元に発送することを通知する.顧客は、商品を受け取ったら、製造者に支払いを行う.UML4ODP を使って表される図では、プロセスは別々のインターラクションに分解される.付加的に記述されるディオンティックコンセプトは、そのインターラクションに結び付けられた一連のルールとして表現されるので、インターラクションが重要視される.それゆに、各インターラクションは水平の区画として、インターラクションの名前がラベルが付けられる.

これらのインターラクションは、異なる図をして描かれる.

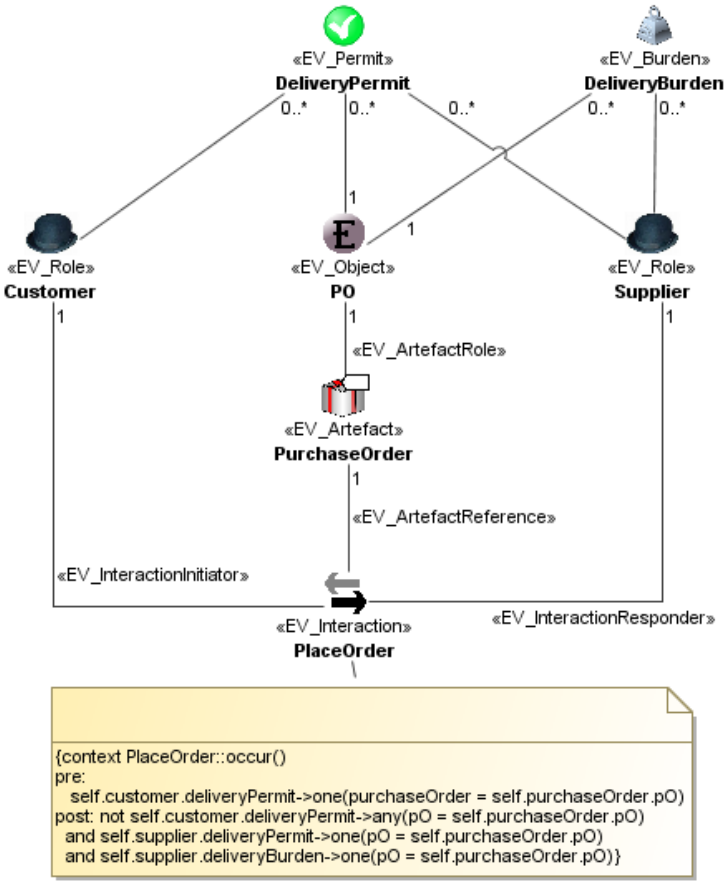

図３　ディオンティックトークンを用いたインターラクション図

アクターや成果物としてのインターラクションロールを明確にする.たとえば、図３は PlaceOrder インターラクションを表し、２つの役割(顧客と製造元)と１つの成果物（購入のための発注）とこの成果物の役割(enterprise object)を満足するオブジェクトを持っている.ODP のインンターラクションは UML4ODP で２つの方法で表すことができる.１つはインターラクションに関与する主要なロールを持つ UML のクラス図と、もう一つはそれらの間の関係である.これらの図は UML4ODP で定義されたものであり、普通の UML の要素間のメッセージフローを表す Interaction Diagram と混同しないようにする.（たとえば、シーケンス図とかコミュニケーション図）

５．ディオンティックトークンの導入

ディオンティックトークンとそれらがどのように操作されるかを表現するため、エンタープライズ言語は幾種類のエンタープライズオブジェクトを使って表される.もし、アクティブなエンタープライズオブジェクト（つまり、動作を持っている）が、関係つくディオンティクトークンを持っているならば、対応するディオンティック制約は、そのオブジェクトの動作に適用される.ディオンティックトークン、それ自身はアクティブなエンタープライズオブジェクトではなく、アクションロールを取ることによって、インターラクションに直接関与しない.各々のディオンティックトークンは唯一のアクティブなエンタープライズオブジェクトに関連付けられる.

エンタープライズ言語は３種類のタイプのディオンティックトークンを持つ.burden は結び付けられているオブジェクトの”obligation”責務を表す.permit は結びつけられているオブジェクトが保持する”permission”を表す.embargo は結び付けられているオブジェクトの prohibition を示す.

これらのオブジェクトを UML で表すために、UML のクラスを拡張するステレオタイプを定義する方法をとる.それぞれに対してアイコンを割り付ける.

重要な問題は、トークンの動態をどう表現するかということである.UML は、システムの動作的側面の詳細仕様を表現するのにあまり適していない.そこで、最もシンプルなアプローチを取ることにする.さらに、プロセスに含まれるトークン（アクティビティ図によって表される）はアクションや成果物に比べて非常に高い.すべてのアクションに対して、多くの permit, obligation の結果を記述することができるようになる.これらは、通常、オブジェクト間で転送される成果物と一緒に渡される.このすべての情報をアクティビティ図に付加することは、(図２に示すように) ディオンティックの情報を書くと、読みづらくごたごたしたものになる.

これを解決するために採られた方法は簡単になるようにこれらの情報を記述することである.ディオンティックトー

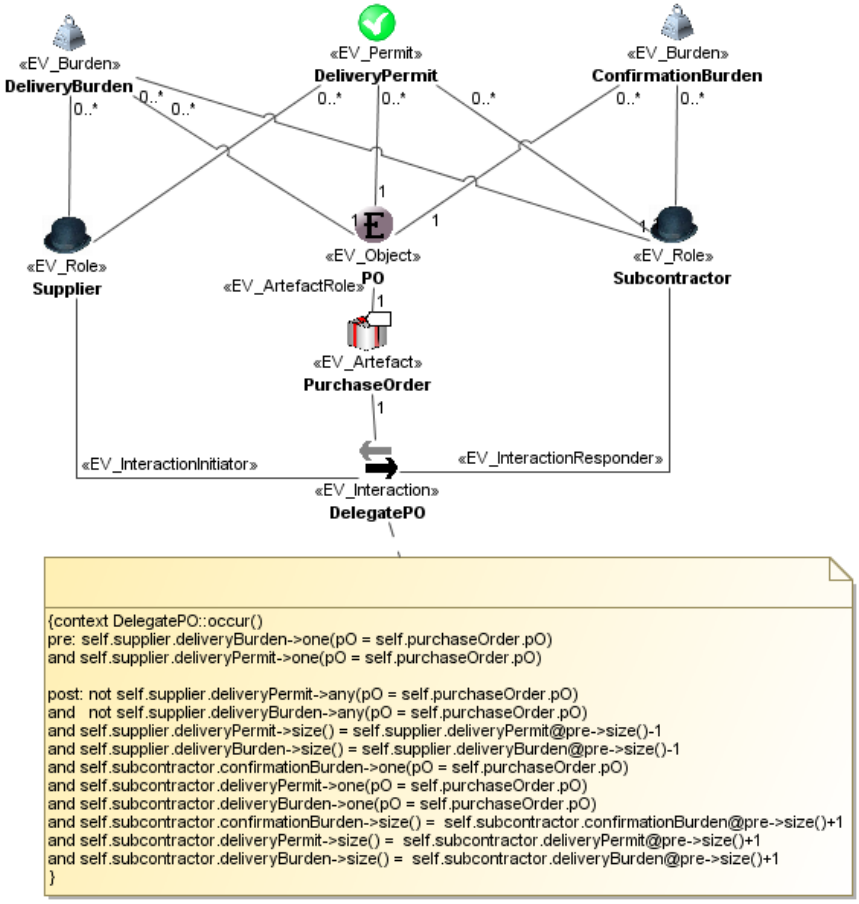

図４　Delegate Purchas order インターラクション図

クンの動態を記述するための適したところは、インターラクション図である.インターラクション図は、プロセス全体の特定のビューを提供し、インターラクションのアクターに焦点をあて、ディオンティックトークンの適切なコンテキストを提供する.これは、”関心事の分離“の仕様の明確な例であり、UML が認める複数のビューを使う.

図４は PlaceOrder インターラクションがそれに関与するディオンティックトークンを伴ってどのように表現されるかを示している.その動作は、インターラクションに結びついている occur()オペレーションの pre,post コンディションの形式で記述される.このオペレーションは、コミュニティー動作のインターラクションオカレンスと共に、静的でテンプレート風のイターラクションのビューに結び付けられる.図４において、顧客は製造元が商品を配送するためのアクセス権を取得できるようにする配送の許可を提供している.これは、アクションが起きた時に、製造元へ転送される.同時に製造元において商品を配送する obligation が生成される.この例をとおして、PO は適切なトークンを識別するためのプライマリーキーとして使われる.

さらに、興味深いインターラクション DelegatePO は、図５に示す.製造元は下請け業者にこの発注に対する実行を委譲することを仮定する.これは DelegatePO インターラクションによって行われるものである.このインターラクションは、下請け業者に“配送”permit と”配送“burden を転送する.これは、下請け業者が配送が行われたとき、製造元と確認する burden を創出する、つまり、配送 burden が実行され、支払いが行われることをわかるようにする.

購買全体に関するほかのインターラクションは同様にして記述される.結果としての仕様記述は、必要とされるトークンに関するすべての情報を組みこんで、プロセスの中でそれらがどのように生成され、委譲され、実行されるかを表している.そしてそれぞれのインターラクションについてモジュール化を行い行っている.

６．関連研究

標準的なディオンティックロジックに基づく多くの方法は、脆弱で、特定のロールを果たす特定のエージェントに関連つけることによって制約を局所化することができなかった.

これらの制約内で、多くの研究は、異なる obligation のカテゴリ分を行っており、適切な例を提供した.

サービス指向の構造に関する研究は、エージェンシーの課題を述べてきた.典型的な例を提供しているものがある.この研究は、本稿で表した方法の表現力を欠いている.なぜなら、ディオンティックコンセプトの制約は、完全に具体化

されておらず、実際に使用している開発者に馴染みのある表記による仕様を表現するのを非常に困難にしているからである.反対に、ここでは UML4ODP によりエンタープライズコンセプトを UML 標準にマッピングさせている.

ここで用いられた方法に類似したものとしては STIT("see to it that")ロジックという Belnap の研究があり、obligation のモデリングの関連した研究がある.これは、完全な基礎とはならないが、本稿の研究の基礎となった.本稿ではオブジェクトベースの方法を採り、詳細レベルでの類似性はあまりない.

いくつかの研究は、特に制約を形式化する時に、ディオンティックコンセプトを表現に UML を用いている.たとえば、[1,2]では、Alam らは UML で表される文法を持つ述語表現を用いて信頼性のあるポリシーを表す方法を示している.[15]においては、著者は、UML アクティビティ図を用いて Trust Management シナリオの negotiation の側面を表現した.しかしながら、この研究は述語の具体化がされておらず、割当／委譲の表現がされていない.この割当／委譲は本稿での主要な先進性である.ここで、オブジェクトとしてディオンティックコンセプトを具体化する方法を構築した[10].これは、ＵＭＬによりディオンティックトークンを表現する課題をうまく解決した.このような表現をしたので、このようなオブジェクトの動作仕様は UML/OCL のシンプルなアプローチの１つを用い、特に新たな方法を開発する必要はない.

７．結論＆課題

本稿では、ディオンティック制約及び UML4ODP による表現の具体化に基づく UML によるエンタープライズ仕様の表現の方法について概略を紹介した.この規格は、現在、最終審議が完了し国際規格として承認が終了している.この方法は、実際に適用するためにはさらなる改善と別の事例に適用して検証する必要がある.別の研究機関や産業界からのフィードバックを得ることは必要である.また、このように投稿するのは、このような理由である.計画どおりに進むと、この研究の成果は２，３年のうちに発行され、これを実装するツールも２、３年のうちに提供されるはずである.

UML4ODP によるシステム仕様記述に関するほかの研究もある.たとえば、この仕様記述はモジュールベースの仕様記述であるが、異なるモジュール化された部分的な仕様（ビュー）を一貫性のある統一化されたシステム仕様としてマージできるようなツールが必要とされる.２番目に、この方法は、システム仕様のデッドロック、不整合の状態、予期せぬ条件（たとえば、ポリシー違反等）の自動分析を目指したものである.このようなツールの開発は重要である.最後に、複数の表現によって対応可能な１つの抽象化仕様となるように共通化されることが必要である.エンタープライズ言語は、複数の形式を使って１つの仕様を表すことを図ってきた.BPMN 等の他の言語でも表すことが可能であることを示すことが必要である.

８．謝辞

長年にわたり、この ODP の開発に関与された多くの国々の貢献者に謝辞を表したい.しかしながら、本稿におけるエラーの責任は著者にある.この研究は部分的に Rearch ロジェト TIN2011-223795 で行われたものである.

９．参考文献

[1] M. Alam, R. Hafner. Model-Driven Security Engineering for Trust Management in SECTET. Journal of Software, 2(1):47-59, 2007.

[2] M. Alam, M. Hafner, R. Breu, S. Unterthiner. A Framework for modeling restricted delegation of rights in the SECTET. Comput. Syst. Sci, Eng, 22(5) 142-151, 2007

[3] N.D. Belnap, M. Perloff, M. Ming Xu. Facing the Future: Agents and choices in our Indeterminist World, Oxford University Press, 2001.

[4] G. Governatory and A. Rotolo. A conceptually rich model of business process compliance.

[5] ISO/IEC IS 10746, Information Technology –Open Distributed Processing, Parts 1 to 4 , 2010. Also published as ITU-T Recommendations X.901 to X.904.

[6] ISO/IEC IS 15414, Information Technology – Open Distributed Processing – Enterprise Language, 2006. Also published as ITU-T Recommendation X.911.

[7] ISO/IEC IS 19793. Information Technology – Open Distributed Processing – Use of UML for ODP System Specifications, 2008. Also published as ITU-T Recommendation X.906.

[8] P.F.Linington, Z. Milosevic, K. Raymond, Policies in Communities. Extending the ODP Enterprise Viewpoint. In Proc. 2nd Int. Enterprise Distributed Object Computing Work (EDOC'98), pages 14-24, San Diego, USA. Nov. 1998.

[9] P. F. Linington,, Z. Milosevic, A. Tanaka, A. Vallecill. Building Enterprise Systems with ODP – An Introduction to Open Distributed Processing. Chapman & Hall/CRC Press, 2012.

[10] P. F. Linington, H. Miyazaki, A. Vallecill.. Obligations and Delegation in the ODP Enterprise Language. In Proc. of the 7th International Workshop on Vocabularies, Ontologies and Rules for the Enterprise (VORTE'12), Colocated with EDOC 2012, Beijing, China, Spet. 2012. IEEE Digital Library.

[11] P. F. Linington, S. Neal. Using policies in the checking

of business to business contracts. In Proc. 4th IEEE Int. Work. on Policies for Distributed Systems and newworks(POLICY '93), pages 207-218, Lake Como, Italy, June 2003. IEEE Computer Society.

[12] J. R. Romero, J.I. Jaen, A. Vallecill. A Tool for the Model-Based Specification of Open Distributed Systems. The Computer Journal, 2013. TO appear. Published on-line, doi; 10.1093/comjnl/bxs021.

[13] D. Ronnedal. An Introduction to Deontic Logic. CreateSpace, 2010.

[14] M. P. Singh, A. k. Chopra, N. Desai. Commitment-based service-oriented architecture. Computer, 42(11):72-79, Nov.2009.

[15] H.Skogsrud, B.Benatallah, F. Casati. Model-driven trust negotiation for web services. IEEE Internet Computing, 7(6):45-52,2003.