プロジェクトのUMLモデル化
－MDADのアプローチ－

技術推進部門
シニアITアーキテクト

松山　圭一

Keiichi　Matsuyama
keiichi-matsuyama@exa-corp.co.jp

システム開発に対する「高品質、短納期、低価格」という市場の強い要求に対応するため、国内SIerは開発プロセス、開発手法に関する大胆な変革を迫られている。本論文では、この解決策として期待される「ソフトウェア工場」に着目し、これを実現する基盤となるMDAD（Model-Driven Adaptive Development）アーキテクチャの開発を行った。MDADは、プロジェクトという業務で扱われる概念をUMLモデルとして抽象化するためのアーキテクチャである。MDADアーキテクチャの基本コンセプト、MDADアーキテクチャに従ったプロジェクトのモデル化、さらにプロジェクト管理システムとして導入した際の期待される品質向上効果について述べる。

1. はじめに

「高品質、短納期、低価格」という顧客からのプレッシャーが強まる中で、国内SIerは開発プロセス、開発手法に関する大胆な変革を迫られている。即効性が期待できる施策として高品質が期待でき（例えば、CMMIレベル５を達成している）、低単価のオフショアSIerへの委託開発という取り組みがある。プロジェクトのうちの可能な限り多くの開発フェーズをオフショアのSIerに肩代わりさせることで、低コストで品質が保証されたシステムを構築し、同時に多数の開発リソースの投入により、短納期をも実現しようとするものである。しかし、オフショア開発の需要増加に伴う単価上昇が予想され、長期的な解決策にならないことが危惧される。

一方で、UMLに代表される統一モデリング言語で記述したクラス図などの詳細設計仕様から最終成果物であるソースコードを自動生成し､開発生産性を向上するというMDA（Model-Driven Architecture）アプローチが「短納期、低価格」を実現する開発手法として期待されている。しかし、MDAだけでは設計・開発プロセスに大きく依存する「高品質」という課題を解決することができない。また、MDAの実現形であるソースコード自動生成はプロジェクトのうちの実装・単体テストという比較的短いフェーズをさらに短縮化するもので、低品質の設計プロセスに起因する結合テスト以降の手戻りによって、削減された工数が容易に打ち消されてしまう懸念がある。

本論文では、MDAのスコープを設計・開発プロセスを包含する"プロジェクト全体"に拡大したMDAD（Model-Driven Adaptive Development：モデル駆動適合型開発）を提案する。MDADは、いわゆる"ソフトウェア工場"のプラットフォーム（生産ライン）に相当する。高品質は、よく計画され、制御された生産ラインによってもたらされ、生産性（短納期、低価格）は、そうした生産ラインに配備された装置によって向上する。MDAは生産性の高い装置を創り出すためのアーキテクチャであり、MDADは品質の高い生産ラインを創り出すためのアーキテクチャである。

まず、２章でプロジェクトを業務として捉え、そこで取り扱われている概念を抽出する。次に、３章でプロジェクトをモデル化するためのアーキテクチャであるMDADに関して提案する。４章では、２章で抽出した概念とMDADアーキテクチャによって構成されるプロジェクトという業務の概念モデルを示す。さらに、５章でこの概念モデルをプログラムに可読なUMLモデルへとマッピングする方法について述べる。最後に、６章でプロジェクトをMDADアーキテクチャに従ってUMLモデル化することによって期待される効果について述べる。

2. 概念の抽出

プロジェクトをプログラムに可読なUMLモデルとして表現するために、プロジェクトを「業務」として捉え、プロジェクトという一連の業務の中でどのような概念が扱われているかという視点で考察する。まず、軸となる業務は「プロジェクトの推進」であるが、これに加え、推進を効率的に行うための「プロジェクトの標準化」、推進が効果的に行われていることを確認する「プロジェクトの監視」という業務がある。監視についても標準化、例えばプロジェクト監査の根拠となるルール作成、が必要である。これらの関係を図１に示す。以下では、これらプロジェクトを構成する３つの業務について検討する。

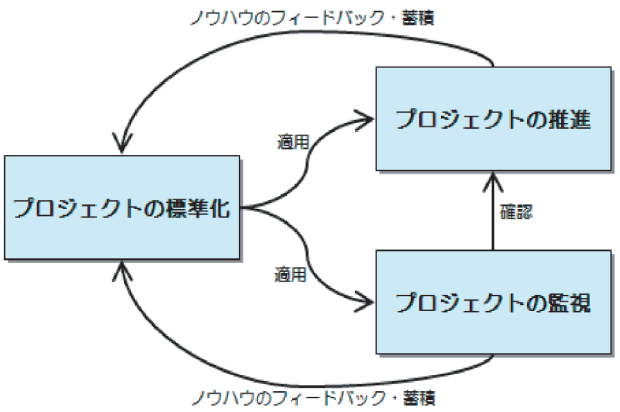

図１　モデル化の範囲

2.1. プロジェクトの標準化

標準化はプロジェクト成果物の品質を保つ上で不可欠の業務であり、プロジェクトの規模が大きくなる程、その重要性は高まる。標準化の対象は、プロジェクトのライフサイクルを定義する工程の定義とライフサイクルを通して作成される成果物の定義とに大きく分けられる。

2.1.1. 成果物の定義

要件定義書、外部設計書などプロジェクトを通して作成する成果物を定義する場合、それぞれの成果物に関して、記載するべき仕様とそれらの記載場所（章立て）を定義した仕様書テンプレートだけでなく、テンプレートに記載する仕様の意味や上流の仕様との関係、記述の粒度などを説明した仕様書ガイドラインを作成することによって、成果物をより明確に定義することができる。

仕様書テンプレートと仕様書ガイドラインという２つの概念とその関連を図２に示す。極めて簡単な図になるが、実際にプロジェクト標準として作成するには多大なコストと成功事例の蓄積を必要とする。特に仕様書ガイドラインは、仕様を単にパッケージングする仕様書テンプレートと異なり、仕様と仕様の間の依存関係を示さなければならない。ここで示している仕様間の依存関係は、ソースコード間の依存関係ではなく、要求仕様からソースコードへと至る具体化の過程を表す依存関係である。仕様書テンプレートと仕様書ガイドラインは相互に依存する。例えば、プロジェクト環境・制約により仕様書の章立てや仕様体系をテーラリングした場合は、ガイドライン中の仕様間の依存関係を修正しなければならない。

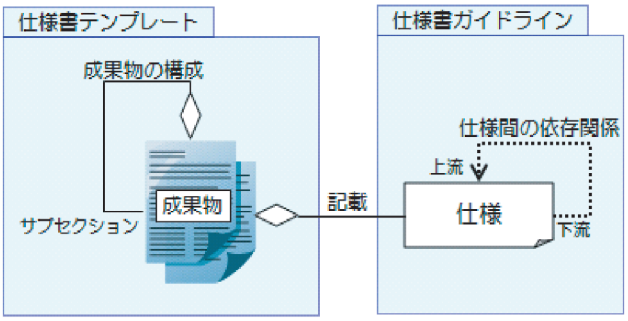

図２　成果物の定義

2.1.2. 工程の定義

プロジェクトを管理しやすい規模のタスクに細分化するだけでなく、それぞれのタスクの入出力となる成果物や前後のタスクとの依存関係を標準工程テンプレートとして定義することで、プロジェクトの実施計画をより明確に示すことができる。

Microsoft社のMS Projectを見ても分かるように、工程計画はタスクと入出力成果物および担当者を主な概念として、WBSのコンポジット構造、前後依存関係および担当者のアサインメントによって定義されている。

ただし、タスクの入出力となっている成果物は、図２に示したように依存関係を持った仕様が記述されているため、図３に示すようなタスクの前後依存関係（MS Projectでは、「終了－開始(FS)」で、他に「開始－開始(SS)」、「終了－終了(FF)」、「開始－終了(SF)」が設定可能）は、仕様書テンプレートに記載される仕様の依存関係と矛盾しないように定義しなければならない。このことは、仕様書テンプレートと仕様書ガイドラインの関係と同じように、入出力成果物のテーラリングによって標準工程テンプレートも都度修正が必要になることを意味している。

また、図３の担当者は、標準工程テンプレートの段階では、個人を指すものではなく、プロジェクトマネージャ、アーキテクト、開発リード、開発者といったプロジェクトにおける役割すべてを包含している。

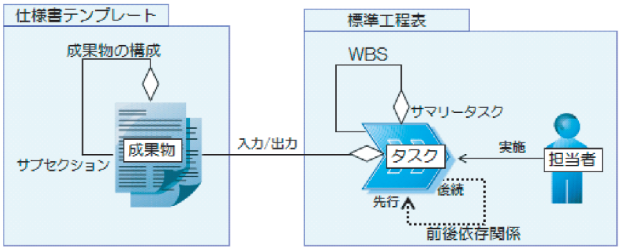

図３　工程の定義

2.2. プロジェクトの推進

ここでは、プロジェクトを推進するにあたって取り扱われる概念に注目する。プロジェクトは前後依存関係を持つ作業（タスク）の集まりであり、個々の作業を完了させることによって進捗する。個々の作業は、入力成果物が確定したことによる開始の作業指示によって始まり、担当者の完了報告に対する管理者の承認によって終了する。

作業は、プロジェクト開始時に計画された作業（以下、計画作業と呼ぶ）とプロジェクト期間中に発生する変更依頼によって都度計画される作業(以下、手戻り作業と呼ぶ)に大別できる。

計画作業にせよ手戻り作業にせよ、作業指示によって成果物の作成または修正が行われ、レビュー結果を記載したレビュー報告書が"合格"という評価で承認された場合にのみプロジェクトは進捗する。同時に、作業指示とレビュー報告書が対で存在し、これらを成果物の改訂履歴に対応付けて管理することで、成果物の品質や完成度を記録することができる。（図４）

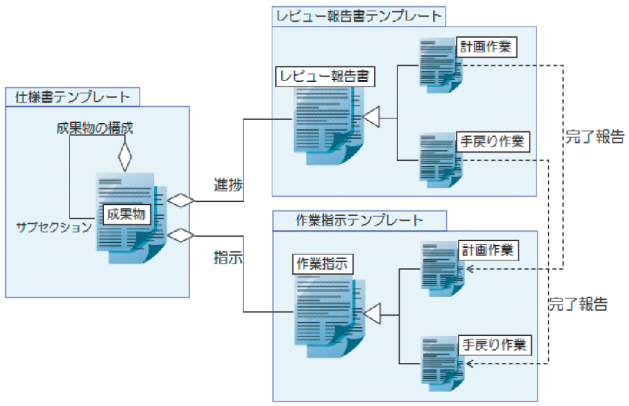

図４　作業指示と完了報告

2.3. プロジェクトの監視

プロジェクトメンバによってプロジェクトが推進される傍らで、様々な視点でプロジェクトは監視されなければならない。監視の視点は、例えば、出来高、生産性、完成度、品質、変更頻度、残業比率、リスクなど様々である。

本論文では、それぞれのメトリクスに関しての説明は行わないが、重要なことは、図５に示すように、工程表は「実施計画」であり、「実績」は成果物に記録されるということである。2.2節に述べたように、作業指示とレビュー報告書によってプロジェクトが進捗することにより、工程表に記載された実行予算等の計画値はタスクに記録され、レビュー報告書の承認によって出来高（あるいは進捗率）は推移し、レビュー報告書に障害の発生状況、除去状況、作業工数、レビュー実施工数等を記載することで、実績値は成果物に記録される。プロジェクトの監視は、こうした計画値に対する実績値を様々な視点で評価することで行われる。

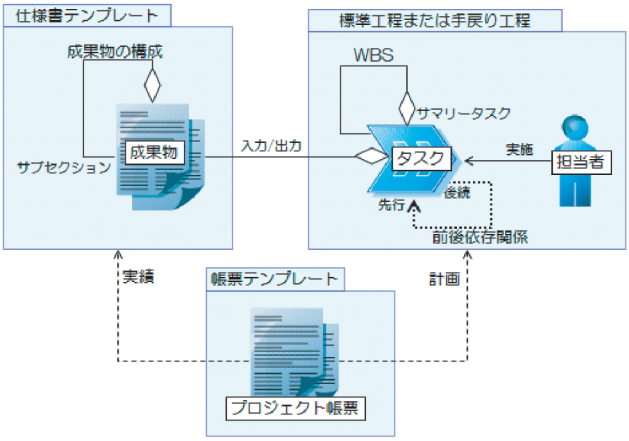

図５　プロジェクト帳票

3. MDADアプローチ

本章では、MDADアーキテクチャを構想したアプローチ方法とその基本コンセプトについて述べる。

MDADアーキテクチャの構想に際し、システムを対象とした概念モデルとして有名なRM-ODP（Reference Model for Open Distributed Processing）やMDAで用いられている「視点（Viewpoint）」という概念に注目した。RM-ODPでは、ODP（オープン分散処理）システムに対する仮説に基づいた５つの視点を設定することで、システムの抽象化を行っている。同様に、MDAでも３つの視点を設定している。これら概念モデルで用いられた視点の例を表１に示す。

表１　MDAとRM-ODPの視点

MDA	RM-ODP
CIM ： Computation Independent Business Model 計算処理に依存しないモデルであり、ユーザからの要求を表現する。	Enterprise Viewpoint 企業におけるシステムの目的、範囲、方針。ビジネスの要求や、いかにそれらの要求を満たすかに注目した視点。
PIM ： Platform Independent Model プラットフォーム技術に依存しないモデル。つまり、複数のプラットフォームで再利用可能なモデルを表す。	Information Viewpoint システムが管理する情報や、管理するデータの種類と構造に注目する視点。 Computational Viewpoint システムが提供する機能と、その機能分解に注目する視点。
PSM ： Platform Specific Model システムが利用する特定のプラットフォーム技術を想定して表したモデル。	Engineering Viewpoint 情報の管理と機能の提供を行うための処理の分散と相互連携に注目する視点。 Technology Viewpoint システムを実現するための要素技術に注目する視点。

RM-ODPやMDAは、プロジェクトというソフトウェア開発活動の結果として作成される「プロダクト」をスコープとしたもので、MDADが狙いとする「プロジェクト」全体を抽象化するものではない。しかし、概念モデルを構想していくアプローチとして、先駆者のプラクティスを参照するのは有用と考えられる。

さて、２章でプロジェクトという業務で取り扱われる概念を抽出したが、これを整理すると図６に示す３つのドメインを導出することができる。ここで、これら３つのドメインがプロジェクトをシステムとして抽象化していくための「視点」に相当すると考えた。

MDAD構想の基盤とした「視点」は、プロダクトそのものを表す「成果物構成」、成果物に内在する仕様の依存関係を表す「トレーサビリティ」、プロダクトを開発するプロセスを表す「プロジェクトライフサイクル」の３つである。これを図７に示す。

プロジェクトライフサイクル、成果物、あるいは両者を対象にした「視点」に基づく概念モデルやシステムは数多く提案されているが、MDADでは新たにトレーサビリティという「視点」を提案する。プロジェクトの進捗に伴い生成される数多くの成果物に含まれる「仕様」を追跡し、かつその依存関係を工程と関連付けて管理することで、より極めの細かい正確なプロジェクト管理の実現が期待される。トレーサビリティはその根拠を与える視点である。

4. 概念モデル

プロジェクトの標準化、推進、監視という３つの業務をモデル化する場合、それぞれの業務は抽象度が異なる概念に注目しているため、プロジェクトで取り扱う概念をOMGのモデル階層にならって、抽象度よって３つの階層に分けて記述した。これを図８に示す。抽象度の高い順に、M2レベル（メタモデル）、M1レベル（モデル）、M0レベル（インスタンス）を表す。

M2レベルは標準化の方針を規定する高次の概念で、実際のプロジェクトの標準化はM2レベルの概念に則って、M1レベルの概念である様々な仕様書テンプレートや標準工程表などの概念を制定する業務である。ここでM1レベルの概念は、開発方法論や既存プロジェクトからのベストプラクティスに基づき構成される。プロジェクトの推進はM1レベルで規定された標準に則って、M0レベルの成果物を作成する業務である。またプロジェクトの監視は、プロジェクトの推進によって作成された成果物をM1レベルに定めたビジネスロジックである評価ロジックに則って確認

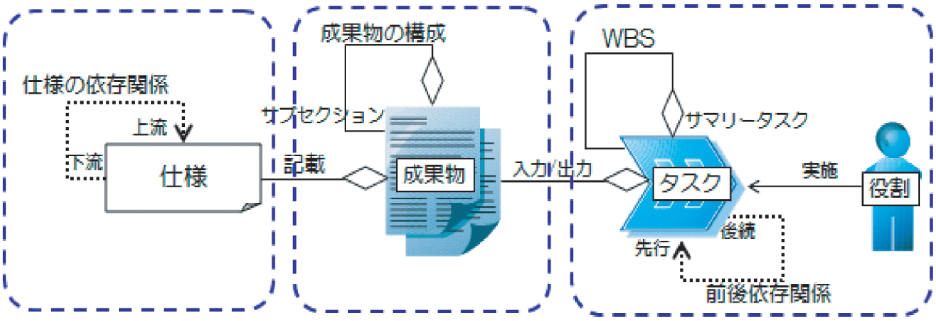

図６　プロジェクトを表す３つのドメイン

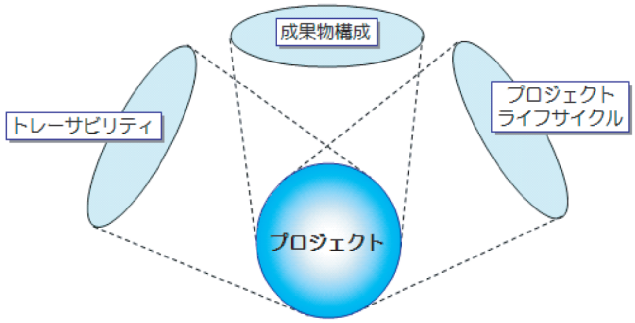

図７　プロジェクトを抽象化する３つの視点

する業務である。MDADはこれらのすべての概念をスコープとしている。

図中で概念を連結している破線矢印は依存関係を示している。例えば、M1レベルの仕様書テンプレートは成果物構成モデルに準拠して作成され、これに依存する。モデル階層間で概念を連結している△破線矢印は、具体化を表している。例えば、M1レベルの仕様書ガイドラインに則って仕様書テンプレートに仕様を記載したものがM0レベルの仕様書である。また、M0レベルの仕様書を作成するために記載された情報は、M0レベルのトレーサビリティや成果物構成をインスタンス化する際の値となる。

4.1. M2レベルモデル

プロジェクトの標準化を行う際には、会社、事業部等の単位に合わせてプロジェクト標準を策定できなければ、実際には使いにくい標準となってしまう。場合によっては、特定のプロジェクト案件ごとに標準をテーラリングしなければならないこともある。したがって、画一的なプロジェクト標準を規定するのではなく、プロジェクト標準の策定方針を規定する。図中M2レベルのトレーサビリティメタモデル、成果物構成メタモデル、プロジェクトライフサイクルメタモデルがプロジェクト標準の策定方針であり、M1レベルのモデルを作成する際の規約となる。モデリングガイドラインは、標準策定者に対して、M2レベルの３つのメタモデルを使ってM1レベルの３つのモデルを作成する際のガイドラインを記したドキュメントである。

4.2. M1レベルモデル

M2レベルの標準化方針に準拠して作成されたこの階層がプロジェクト標準となる。プロジェクトメンバは、M1レベルに定義されたモデルに則って作成された仕様書テンプレートと仕様書ガイドラインに則って、M0レベルのプロジェクトの成果物を作成する。

4.3. M0レベルモデル

モデルと呼ぶには多少違和感があるが、M1レベルモデルに定められたプロジェクト標準に従って作成された個々のプロジェクト案件の成果物を表している。プロジェクトの監視は、M1レベルで評価ロジックとともに制定したプロジェクト帳票テンプレートに、M0レベルで生成される成果物構成やプロジェクトライフサイクルの情報を集約することで、様々な視点での評価結果を得ることができる。

5. MDADモデル

MDADモデルには、メタモデル（M2レベルモデル）を

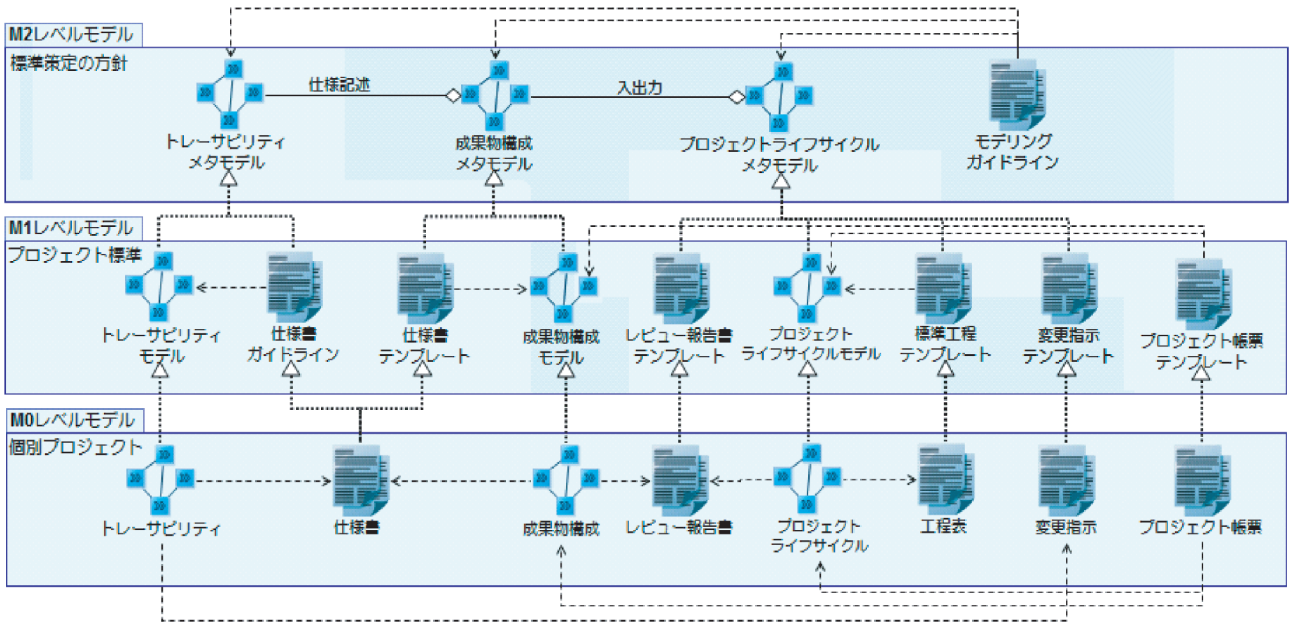

図８　概念モデル

規定することで、モデル（M1レベルモデル）として定義されるプロジェクト標準のテーラリングを可能にするという目的と、システムに可読なクラスにマッピングするという２つの目的がある。

本論文では、４章で定義した概念モデルをシステム化していくための手法を中心に説明することにとどめ、実際にMDADメタモデルを使って作成した下位モデルの詳細な説明は紙面の都合上割愛する。実際に作成されるモデルは、開発方法論とプロジェクト経験を根拠として作成されることになる。

5.1. メタモデル

4.1章でも言及したように、標準といっても様々な規模、期間、タイプのプロジェクトに適用するためには、モデルすなわち標準として固定するのではなく、標準を策定する方針を規定することで標準のテーラリングを容易にしなければならない。MDADではこのコンセプトのもとに、トレーサビリティメタモデル、成果物構成メタモデルおよびプロジェクトライフサイクルメタモデルの３つのメタモデルを定義している。

メタモデルといっても、新たなメタモデル要素を導入するのではなく、UMLメタモデルをステレオタイプやタグ付値によって拡張することによってMDADメタモデル要素を定義している。M1レベルのモデルは、このMDADメタモデル要素を使ってモデリングすることによって、標準の策定方針に従っていることが保証される。

5.1.1. トレーサビリティメタモデル

プロジェクトライフサイクルを通して作成される仕様および仕様間の依存関係を規定する。ここで定義している仕様間の依存関係は､要求からソースコードに至るまでの｢具体化や詳細化」といった仕様間の依存関係を示している。

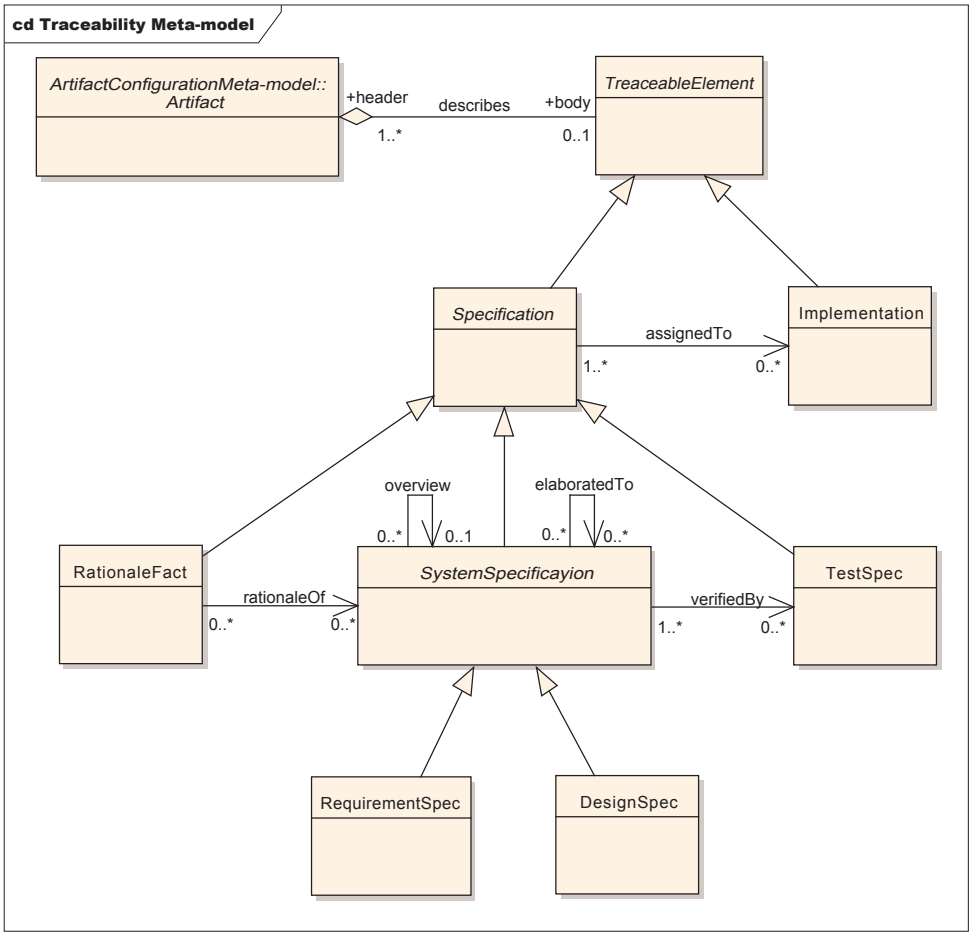

図９　トレーサビリティメタモデル

トレーサビリティというと、何らかの作業を経て作成された成果物に対してその経路(工程)を都度記録していくというように、あくまでもインスタンスに対して、言わば「後付け」で記録される履歴情報を一般的には意味する。これに対し、MDADではあらかじめプロジェクトライフサイクルに登場するすべての仕様とそれぞれの仕様間の依存関係をモデルとして定義することで、トレーサビリティ管理のシステム化を可能としている。また、このことにより、仕様書作成者に対しては仕様書ガイドラインに相当する情報が提供できるようになる。

図９にMDADのトレーサビリティメタモデルを示す。

MDADでは、仕様間の依存関係の向きを通常のUMLとは逆向きに定義している。これは、「具体化や詳細化」といった上流から下流へと向かう依存関係を「影響の向き」として捉えているためである。矢印線のソースとターゲットは、ソースの変化がターゲットの変化を引き起こすものであり、ターゲットからソースへの影響はないというコンセプトに基づいている。

5.1.2. 成果物構成メタモデル

MDADでは、成果物は仕様のパッケージングとタスクの入出力を表す。仕様と仕様間の関連を定義したトレーサビリティモデルは、トレーサビリティメタモデルに準拠し、開発方法論を根拠として定義されるものであり、個別のプロジェクト案件ごとに変わるものではない。一方で、成果物構成は工程の組み方によって容易に変化する。成果物構成メタモデルは、ドキュメント構造を表すためのコンポジット構造と仕様を記載する際のスタイルを規定する。仕様書のスタイルには、段落番号付き、箇条書き、表形式など様々な表現形式があるが、これを規定しているのが図中の網掛けをしたクラスである。（図10）

5.1.3. プロジェクトライフサイクルメタモデル

MDADにおけるプロジェクトライフサイクルメタモデルは、WBSと各タスク間の前後依存関係、および入出力成果物と担当者の割り当てを規定している。（図11）

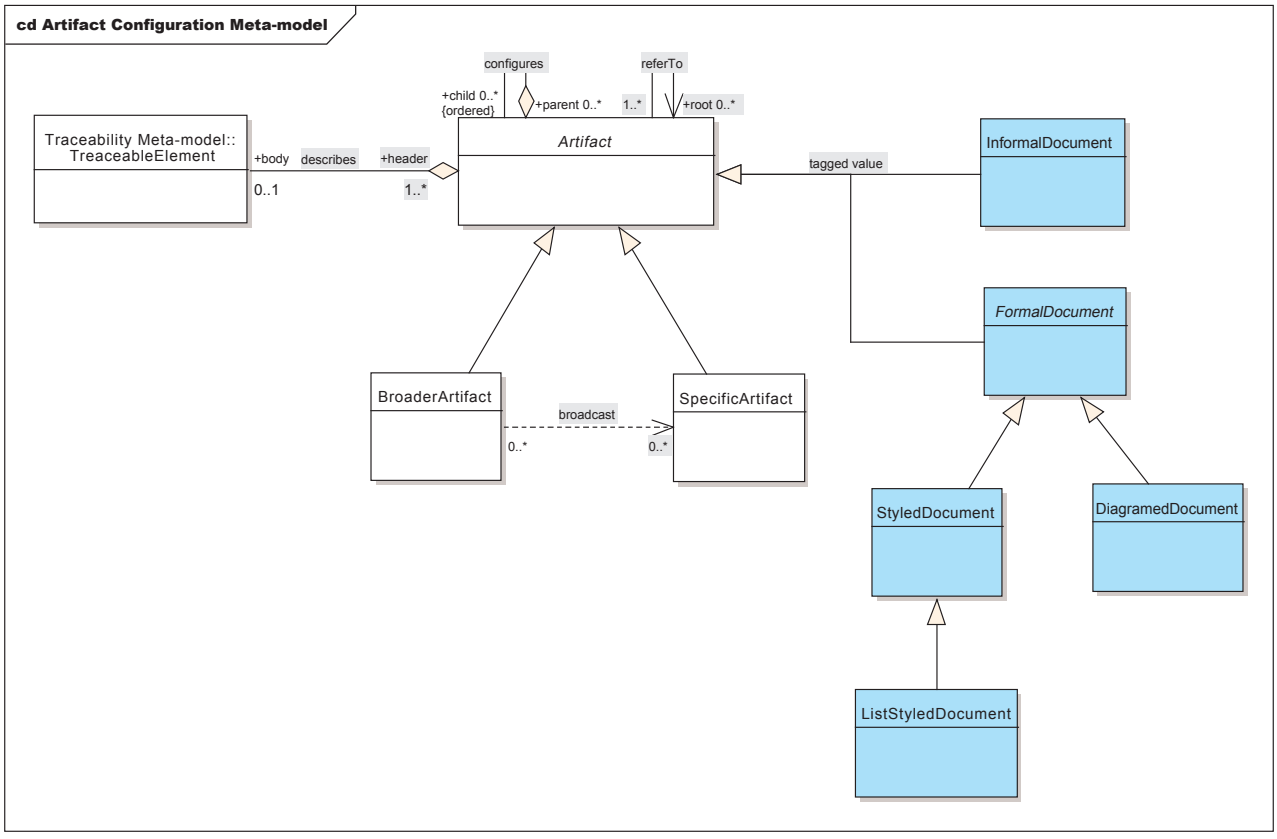

図10　成果物構成メタモデル

5.2. メタクラスからステレオタイプ付クラスへのマッピング

上記のように定義した３つのメタモデルをプロジェクト標準策定の方針として標準策定者に提供するために、各メタモデルをUMLプロファイルとして標準策定者のツールにインポートすることが可能となっている。標準策定者は、このUMLプロファイルで供給されたステレオタイプ付のクラスや関連を使って標準となるモデルを作成する。

図10の成果物構成メタモデルを例に取ると、成果物を表すSpecificArtifactとBroaderArtifactという２種類のメタクラスは、図12に示すようにステレオタイプとして表され、仕様を記述する際のスタイルと仕様書の章立てを表す章番号はそれぞれタグ付値format、orderで表される。

MDADが提供するステレオタイプ付のクラスや関連を使って作成された３つのM1レベルのモデル（トレーサビリティモデル、成果物構成モデル、プロジェクトライフサイクルモデル）は、プロジェクト標準としてシステムに読み込まれる。プロジェクトの進捗に伴って、開発者が作成した仕様書やレビュー報告書から読み込んだ情報に対して、標準である３つのモデルをもとに標準に従っているか否かの検査をしながらM0レベルのインスタンスが生成されてゆく。

6. 期待される効果

５章までに、プロジェクトという業務で取り扱われる概念の抽出とUMLモデル化について述べてきた。プロジェクトをプログラムに可読なUMLでモデル化することによって期待されるもっとも大きな効果は、開発方法論を根拠として作成した３つのモデルによって、これまで先人のベストプラクティスに頼ってきたプロジェクト計画に開発方法論という根拠を与えることができるということと、開発者に対しては成果物作成時に品質を保つためのガイドライン情報のタイムリーな提供が可能になるということである。ここでは、MDADモデルを導入することで期待できる従来のプロジェクト管理システムにはなかった効果について説明する。

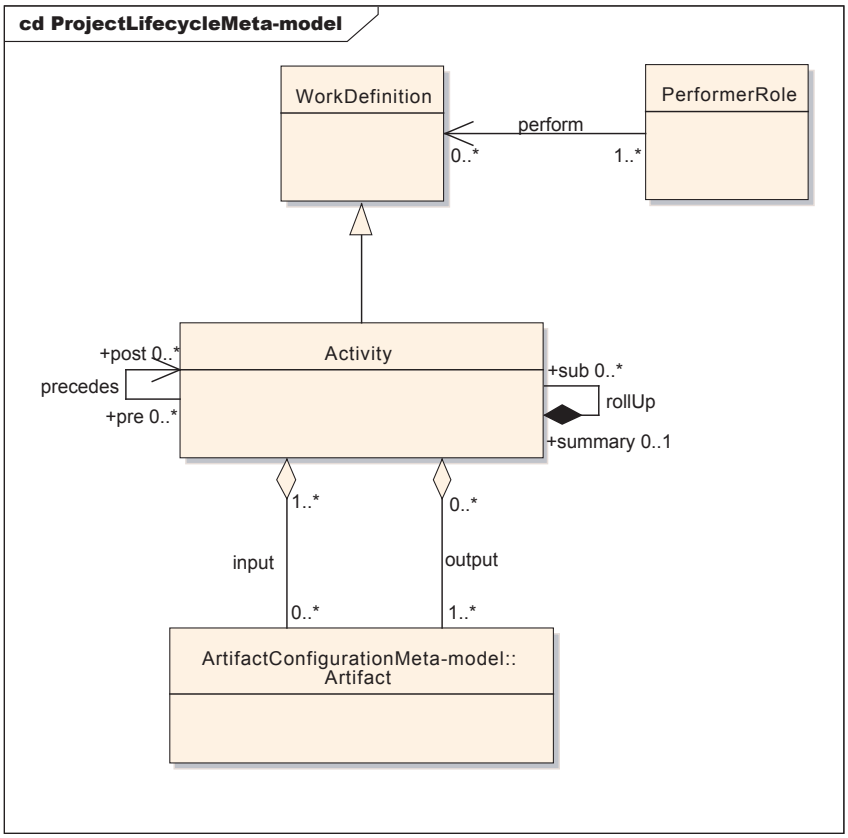

図11　プロジェクトライフサイクルメタモデル

6.1. UMLモデルによるベストプラクティスの蓄積と再利用

UMLという統一モデリング言語でプロジェクトを定義し、プログラムに可読な形式（例えば、XMI）に保存し、これらのモデルで駆動されるプロジェクト管理システムに搭載する。このシステムのもとで行われる実際のプロジェクト案件からのフィードバックを即座にモデルに反映し、新たな案件を行うプロジェクトに提供することができる。様々な開発方法論、規模、期間、プロダクトに関して、実績のあるモデルを蓄積し、再利用していくことで、迅速かつ確実なプロジェクト計画の立案が容易になる。

6.2. テンプレートの自動生成

Microsoft社のMS Office製品に見られるように、多くの開発環境ツールはXML形式で成果物を入出力できるようになっている。XML形式の採用により、プロジェクト標準として搭載されたプロジェクトライフサイクルモデルや、成果物構成モデルに準拠した標準工程テンプレートや、仕様書テンプレートを、MS ProjectやMS Wordなどの XML形式で自動生成することができる。また、これにより工程表のWBSに合わせた仕様書テンプレートの自動分割が可能になり、見かけ上大きな一塊の仕様書であっても、複数のタスクあるいは担当者が編集できるという構成管理上の利点が得られる。

6.3. プロジェクト推進の自動化

プロジェクトライフサイクルモデルに準拠して生成された標準工程テンプレートに、タスクの開始日、終了日、担当者等の個別プロジェクトの実計画情報を記載し、再度システムに読み込ませることで、プロジェクトは標準工程に準拠して推進される。本論文では述べていないが、システム化の際にプロジェクトライフサイクルとあわせてレビュープロセスや課題管理プロセスなどの業務フローを定義、実現することでプロジェクトメンバに対して、タイムリーかつ標準に準拠した開発活動をガイダンスすることができる。これにより品質向上のための作業負荷を軽減することができる。

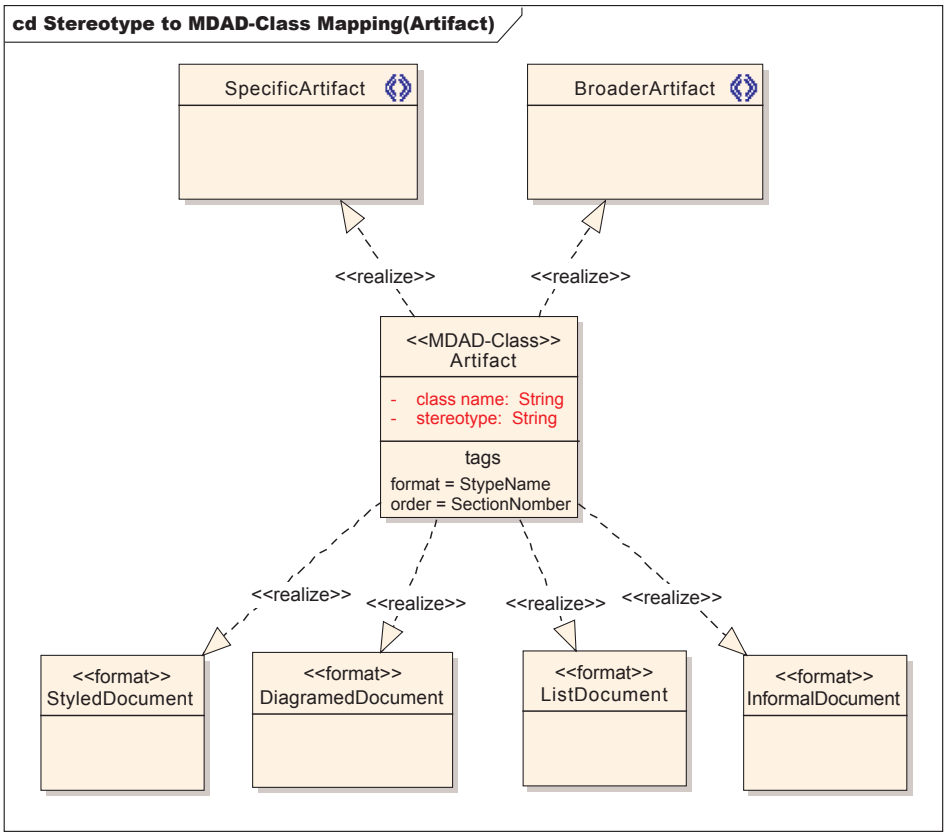

図12　メタクラスからステレオタイプ付クラスへのマッピング

6.4. 成果物検査の自動化

プロジェクト計画や仕様書は、プロジェクト標準であるモデルに準拠して生成されたプログラムに可読な形式（例えば、XML）の工程テンプレートや仕様書テンプレートを使ってPMや開発者が作成する。したがって、システムはこれらの成果物をモデルと照合することによって、テンプレートの改竄や、過不足あるいは間違った依存関係の設定を検知し、正しいガイダンスを行うことができる。

6.5. 変更要求発生時の影響分析と手戻り工程の自動生成

プロジェクト標準であるモデルに従って作成される仕様、成果物、タスクは、個々のプロジェクト案件が進捗するに伴い、それぞれM0レベルのトレーサビリティ、成果物構成、プロジェクトライフサイクルとなってインスタンス化されていく。したがって、変更の対象となっている仕様がどのタスクに該当し、どの成果物に記載され、作成済みのどの仕様に影響を与えている可能性があるかを、その工程とともに導出することができる。
プロジェクトにおいては、変更要求や追加要求は避けることができないものであり、いかに効率よく無駄のない対応をするかが重要となる。これまで述べてきたように、３つのモデルによって、プロジェクトは図13のようにモデル化され、このモデルに従ってシステム内に展開されている。

変更要求は、仕様変更依頼という形式で図中の仕様に対して行われる。システムは以下の手順を経て手戻り工程を導出することができる。

・　影響を受ける仕様を特定する。
トレーサビリティモデルに定義されている仕様の依存関係を下流へとたどり影響を受けている可能性のある仕様を特定する。
・　仕様が記載されている成果物を特定する。
成果物構成モデルに定義されている成果物と仕様の関連（記述関連）から影響を受けている可能性のある仕様書の章番号を特定する。また、成果物の構成をたどることで、当該仕様がどの仕様書に記載されているかを特定する。
・　影響を受けるタスクを特定する。
プロジェクトライフサイクルモデルに定義されているタスクと成果物との関連（入力関連）から影響を受けている可能性のあるタスク、すなわち手戻り作業を特定する。また、タスク間の前後依存関係とWBS構造から、手戻り工程を導出する。
・　担当者を特定する。
プロジェクトライフサイクルモデルに定義されているタスクと担当者の関連から、手戻り作業の担当者を特定し、手戻り工程表が完成する。

6.6. プロジェクトの統一的な評価と実績の蓄積

本論文では述べていないが、MDADモデルを導入してプロジェクト管理システムを実現する際に、個々のクラス（ArtifactやActivityなど）にプロジェクトを評価するための様々なメトリクスを持たせ、評価ロジックを定義したプロジェクト帳票テンプレートにメトリクスの値を集約する。これは、プロジェクトの評価に統一的な基準を与えるだけでなく、様々なプロジェクトの実績を蓄積することで、同タイプのプロジェクトを推進していく際の判断やリスク

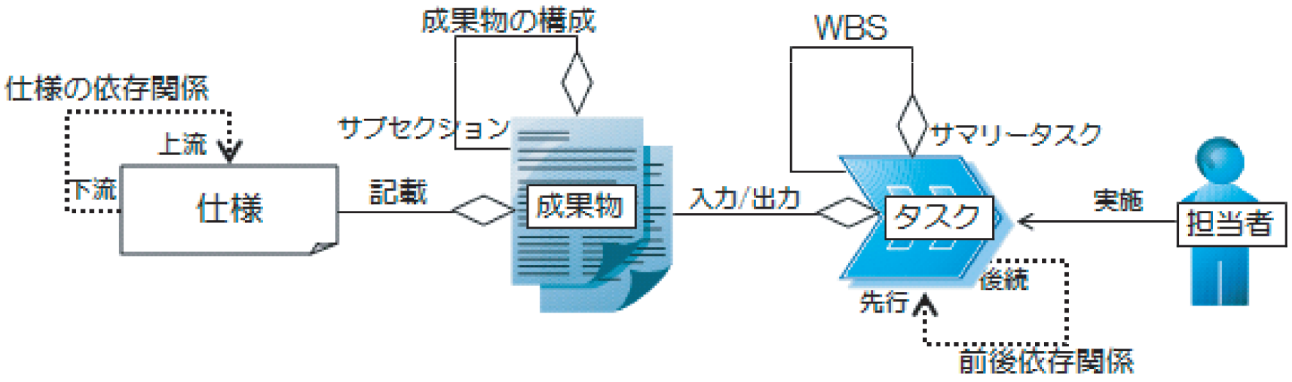

図13　モデル化されたプロジェクト（概要）

回避を助けるものとなる。

7. おわりに

プロセスに注目するプロジェクト管理システム、成果物構成に注目する構成管理システム、あるいはその両方を併せ持つ統合システムは、既に多く存在している。MDADは、さらにトレーサビリティモデルを導入することで、これらに開発方法論に基づく根拠を与えるものである。

当社では、ソフトウェア工場を実現するプラットフォームとして、MDADモデルを搭載して駆動する開発環境TowerⅡの開発を進めており、本論文執筆時点で完成間近になっている。並行して、コンポーネントベース設計とウォータフォール型開発というコンセプトでのプロジェクトのUMLモデル化と各種テンプレートの開発を進めており、MDADがプロジェクトに適用される日も近い。

しかしながら、MDAD開発は今やっとスタートラインにたったに過ぎない。MDADのモデル化が多くのプロジェクト経験者やアーキテクト、プロセスコンサルタントの知見を集めて行われてきたものであっても、実際のプロジェクトに適用することで初めて様々な規模、期間、方法論、プロダクト種類に耐え得るプロジェクト標準へと洗練されていく。また、開発環境TowerⅡに関しても同様であり、実運用からのフィードバックに基づいて、様々な機能の改善と拡張が必要となってくる。開発プロセスと手法を大胆に変革するという大きな目標に向け、継続して取り組み、実用化を推進していく所存である。

参考文献（Web公開資料）

1) OMG、"OMG Model Driven Architecture"、http://www.omg.org/mda/
2) IBM、"Rational Unified Process"、http://www-306.ibm.com/software/awdtools/rup/support/
3) OMG、"OMG Unified Modelling Language Specification"、Version 1.5、2003/03/01.
4) J-M. Comily, M.Belaunde、"Specifying distributed object applications Using the Reference Model for Open Distributed Processing And the Unified Modelling Language"、http://universalis.elibel.tm.fr/publications/edoc99.pdf.
5) Joaquin Miller and Jishnu Mukerji、"MDA GuideVersion 1.0.1"、http://www.omg.org/docs/omg/03-06-01.pdf、2003/6/12.
6) OMG、"Meta Object Facility(MOF) Specification"、Version 1.4、http://www.omg.org/docs/formal/02-04-03.pdf、2002/4.
7) Patricio Letelier、"A Framework for Requirements Traceability in UML-based Projects".
8) OMG、"Software Process Engineering Meta-model Specification"、Version1.0、http://www.omg.org/technology/documents/formal/spem.htm、2002/11.
