2005－IS－91（8）
2005／3／15

アスペクト指向によるプロセスモデリング手法

上野 浩一郎
三菱電機情報技術総合研究所

最近、ソフトウェアの開発プロセスをオブジェクト指向でモデリングするアプローチが増えつつある。オブジェクト指向で定義した開発プロセスでは、要求／設計／実装／試験などのエンジニアリング分野の成果物や作業は、クラスや操作で表現される。しかし、プロジェクト管理や構成管理といった支援分野の作業をオブジェクト指向でモデリングしようとすると、エンジニアリング分野のプロセスを表す様々なクラスや操作の中に散在して埋め込まれてしまう傾向がある。これは支援分野の作業は、エンジニアリング分野の作業や成果物と横断的に関わるためである。そこで本稿では、支援分野のプロセスを、オブジェクト指向モデルから分離独立して、アスペクトとして定義することを提案する。プロセスをアスペクトとして定義することにより、開発プロセスのテーラリングに際して、プロセスモデルの保守と再利用を高めることが期待できる。

Aspect-Oriented Modeling for Software Process

Koichiro Ueno
Mitsubishi Electric Corporation Information Technology R&D Center

Recently, the software development process is described by object-oriented modeling. The artifacts and activities of Engineering discipline, such as Requirement / Design / Implementation / Test, are expressed by the classes or operations. However, the artifacts and activities of Support discipline, such as Project Management and Configuration Management are scattered and embedded into various classes for Engineering discipline, because the activities of Support discipline are crosscutting concerned across the artifacts and activities of Engineering discipline. Then, this paper proposes that the process of a Support discipline is described with Aspect-Oriented, in order to separate from the object-oriented model. Using Aspect-Oriented process, it is expectable to raise maintenance and reuse of the process model when carrying out the tailoring of a software development process.

1. はじめに

近年、ソフトウェア開発の短期化やコスト削減の要求がより強くなり、また開発技術の変化の激しさから、開発プロジェクトのリスクが上がっている。この問題に対する１つのアプローチとして、開発の作業や成果物を標準化した開発プロセスが期待されている。本稿では開発プロセスをオブジェクト指向でモデリングする問題を考察し、その結果から開発プロセスのモデリングにアスペクト指向技術を導入することを提案する。なお本稿は筆者による参考文献[1]を発展させたものである。

2. 現在までの開発プロセス技術

2.1. パラダイムの変遷

ソフトウェア工学におけるパラダイムの変遷と、それに追随したソフトウェア技術と開発プロセス技術を図１に示す。過去、パラダイムは「構造化」から「オブジェクト指向」へと変遷し、ソフトウェア技術もそれぞれのパラダイムに応じて「プログラミング」から「分析設計」と拡大してきた。開発プロセスは、ソフトウェア技術を用いた開発者の作業を定義するものなので、ソフトウェア技術に追随している。ここで注目すべきは、開発プロセス自身の定義方法がソフトウェア技術に強く対応している点である。例えば、ソフトウェア技術として「構造化分析設計」を前提とした開発プロセス[2]は、開発作業とその成果物の関係を Data Flow Diagram で定義していた。またオブジェクト指向開発の開発プロセスである Rational Unified Process[3]は、自らのプロセスをオブ

ジェクト指向でモデル化している。このような傾向は Osterweil[4]が指摘した”Software Process Are Software Too”の考え方に基づくものである。ソフトウェア技術を開発プロセス定義に援用してパラダイムを統一することは、プロセスエンジニアと開発者が同じモデリング記法や技法を共用できるので妥当である。本稿はこの考え方を踏襲しつつ、これからのパラダイムである「ポストオブジェクト指向」に関わるアスペクト指向プログラミングと、メタモデリングに該当する SPEM(Software Process Engineering Metamodel)[5]を援用した開発プロセス技術に位置付けられる。

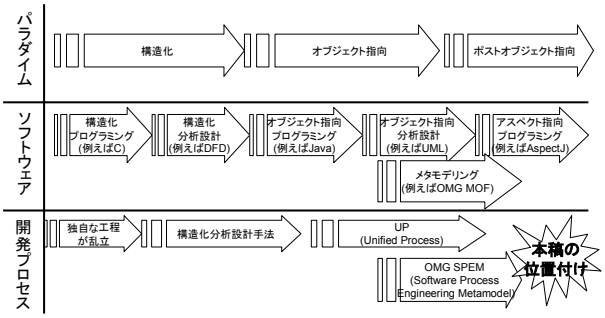

図１　技術の変遷

2.2. SPEM

SPEM は、OMG(Object Management Group)が標準化した開発プロセスのメタモデルである。SPEM は自らの位置付けを明確にするために、開発プロセスに関係するメタレベルを以下の４種類に分類している。メタレベル１「Process Model」は、具体的な開発プロセスのインスタンスが位置付けられるレベルであり、例えば RUP 等が該当する。メタレベル０「Performing process」は、具体的なプロジェクトにおいて計画して実行されるインスタンスが位置付けられるレベルである。メタレベル２「Process Metamodel」は、開発プロセスそのものを表現する構成要素(例えば Activity や WorkProdukct)が位置付けられるメタレベルであり、SPEM はこのレベルに位置する。メタレベル３「MetaObject Facility」は、メタレベル２を表現する構成要素が位置付けられるメタメタレベルである。

図２に SPEM の一部抜粋を示す。本稿で言及する要素は、図上で網掛けしており、それぞれについて表１で説明する。また表１では、SPEM 要素をステレオタイプとして設定可能な UML 要素と、その表示用のアイコンを示す。

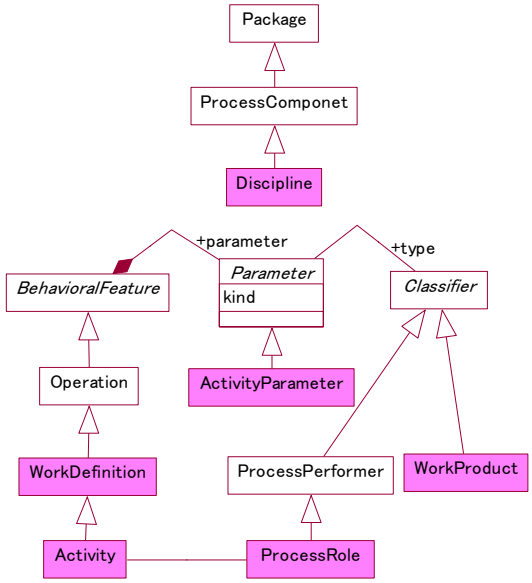

図２　開発プロセスのメタモデル SPEM

表１　SPEM の要素（一部抜粋）

SPEM 要素	説明	UML 要素	アイコン
Discipline	作業分野	パッケージ	
WorkDefinition	作業定義	ユースケース	
Activity	作業	アクターの操作、あるいはアクション状態	
ProcessRole	役割	アクター	
WorkProduct	成果物	クラス、あるいはオブジェクトフロー状態	
ActivityParameter	作業パラメータ	オブジェクトフロー	

2.3. SPEM に基づくプロセスモデルの例

本節では、SPEM に基づいて UML で定義したプロセスモデルの具体例を図３～５に示す。このプロセスモデルは統一プロセス[6]を参考にしているが、本稿の議論用に非常に簡易化している。なおこのプロセスモデル例は以後の章で、適宜参照する。

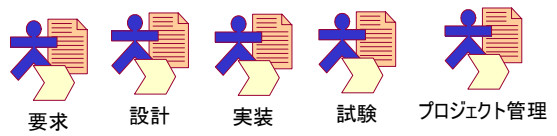

図３　Discipline

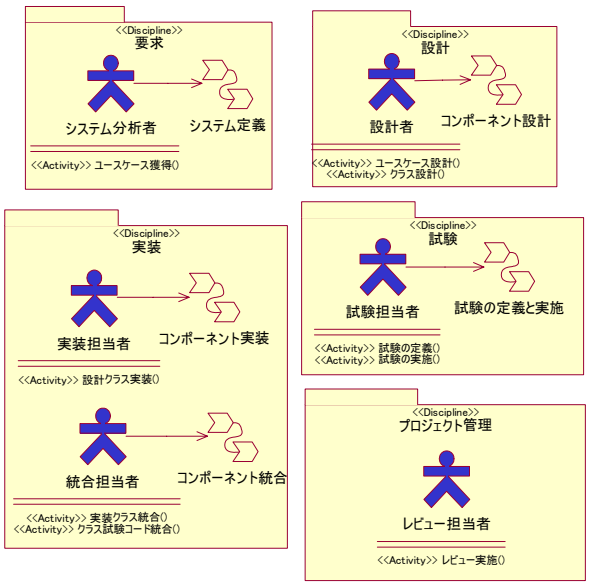

図４　ProcessRole と WorkDefinition

3. 開発プロセスに関する考察

3.1. 開発プロセス定義の特長

開発プロセスの Discipline を、開発の主体であるエンジニアリング分野と、その支援分野に分類する。例えば図３の例では、要求/設計/実装/試験がエンジニアリング分野、プロジェクト管理が支援分野となる。（なお支援分野の他の例としては構成管理、変更管理、環境構築などがある。）そしてこの２つの分野の関係から、開発プロセス定義の以下の特長を明らかにする。

特長１

支援作業は個々のエンジニアリング作業やその作業成果に対して繰り返し適用される。

特長２

シーケンシャルなのでスケジューリング可能なエンジニアリング作業と比較して、支援作業の実行はイベントドリブンである。

特長３

技術面が強いエンジニアリング作業と比較して、支援作業はプロジェクト規模などに強く依存するので、テーラリングされる割合が大きい。

特長４

組織外の業界標準あるいは市販の開発プロセスをテーラリングした開発プロセスは、オリジナルの開発プロセス改訂に追随必要となる。更に昨今はソフトウェア技術の陳腐化が早いため、開発プロセスの改訂間隔も短期化している。

3.2. オブジェクト指向によるプロセスモデリングの問題

本節では、前節で指摘した特長がオブジェクト指向によるプロセスモデリングで、どのような問題を生むかを考察する。

特長１と２のために、支援分野のプロセスを正確にモデリングしようとすると冗長になってしまう。例として、2.3 節のプロセスモデル例における Activity「レビュー実施」を取り上げる。図５では「レビュー実施」の入力は「任意の WorkProduct」と表現している。これを正確にモデリングすると、図６のようにプロセスモデルにおける全ての WorkProduct を描き表

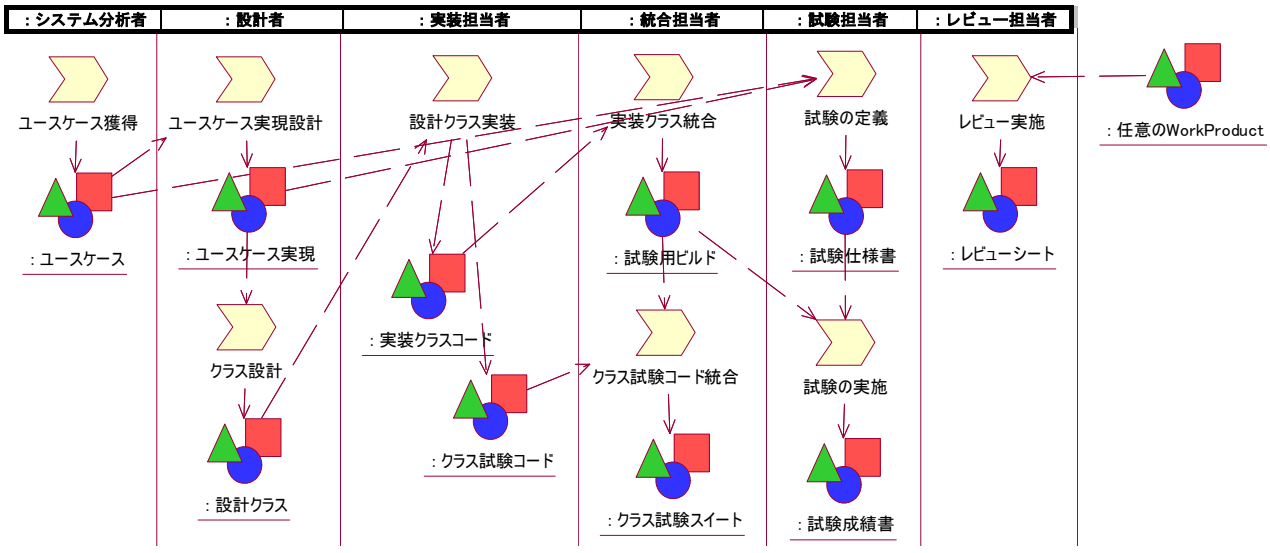

図５　Activity と WorkProduct

わさなければならず冗長である（特長１）。このことはメタレベル１のプロセスモデルに責任を持つプロセスエンジニアにとって負荷が高い。

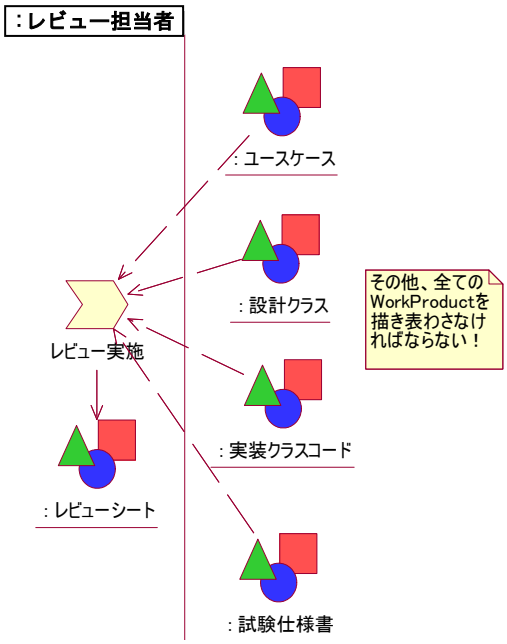

図６「任意の WorkProduct」を具体化

またメタレベル０におけるシーケンス図の例を図７に示す。これは WorkProduct「ユースケース」に対するモデルであるが、正確には他の全ての WorkProduct のインスタンスに対しても同様なシーケンス図を描き表わさなければならず冗長である（特長２）。このことはメタレベル０のプロセス実行に責任を持つプロジェクト管理者にとって負荷が高い。

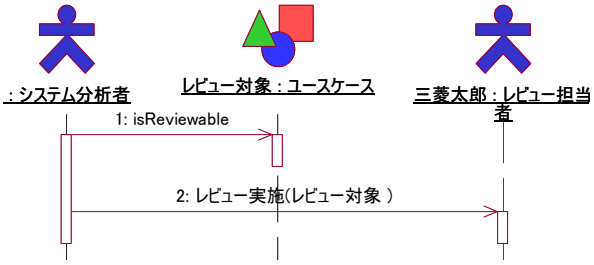

図７　WorkProduct「ユースケース」に対するレビュー実施

更に問題なのは、特長１と２に対して正確に記述したプロセスモデルは、エンジニアリング分野と支援分野が密接に一体化したものになってしまう点である。特長３で指摘した通り、支援分野はテーラリングされる割合が大きいので、密接に一体化したプロセスモデルは柔軟性に欠け、テーラリングコストが高くなってしまう。更に特長４で指摘した通り、オリジナルの開発プロセスが改訂される度に、同じ内容のテーラリングを実施しなければならない。

3.3. 開発プロセスにおける Separation of Concerns

前節で指摘した問題は、開発プロセスにおける別種の「関心事(Concern)」が「もつれている(tangling)」ことが原因である。ここで Concern や tangling は「SoC(Separation of Concerns)」[7]の用語である。そもそも SoC とは、ソフトウェアの様々な側面を視点毎に分離して、保守性や再利用性を高めようとする考え方である。対象ソフトウェアが本来目的としている関心事を「Core Concern」、Core Concern に直交してソフトウェア全体に横断する関心事を「Crosscutting Concern」と呼ぶ。そして Core Concern に Crosscutting Concern を織り込むことを「weaving」と呼ぶ。

本節ではこの SoC の考え方を開発プロセス定義に導入する。3.1 節で指摘した４つの特長から、エンジニアリング分野を Core Concern、支援分野を Crosscutting Concern と捉えるとする。

なお参考文献[8]でも開発プロセス定義に SoC の考え方を導入しており、相互に直交する複数の Concern を示している。但し本稿が示すような Core Concern に weaving する Crosscutting Concern という考え方は示されていない。

4. アスペクト指向によるプロセス定義の提案

本章では、前章で導入した概念 Crosscutting Concern を「アスペクト」としてプロセス定義することを提案する。なお以下、プロセスに関するアスペクトを「プロセスアスペクト」と称する。4.1 節は、プロセスアスペクト定義で前提にする準備事項を説明する。4.2 節と 4.3 節では、プロセスアスペクトの具体例を擬似コードで示す。

4.1. プロセスアスペクト定義の準備

① アスペクトを形式的に記述する文法として擬似的に AspectJ[9]を援用する。これはアスペクトを形式的に定義する記法が標準化されていないため便宜的に代替している。

② プロセスアスペクトにはメタレベルに応

じて 2 種類ある。メタレベル 2 の構成要素(即ち SPEM 要素)を用いて定義し、メタレベル 1 の構成要素に weaving するプロセスアスペクトを M1aspect と称する。同様に、メタレベル 1 の構成要素(即ちプロセスモデルの要素)を用いて定義し、メタレベル 0 の構成要素に weaving するプロセスアスペクトを M0aspect と称する。

③ 擬似コード中から SPEM 要素にアクセス可能と仮定する。

④ インスタンスへは名前でアクセス可能と仮定する。

⑤ 擬似コード中、***BoldItalic*** で表示している英字名は SPEM が定義している要素であることを示す。

⑥ 以下で示すプロセスアスペクトは、2.3 節で示したプロセスモデルを対象とする。

4.2. メタレベル 1 のプロセスアスペクト例

図 5 のアクティビティ図では、Activity「レビュー実施」の入力が「任意の WorkProduct」であると表現している。このことは Activity「レビュー実施」が WorkProduct を横断する Crosscutting Concern であることを意味している。この Crosscutting Concern を表すプロセスアスペクトをリスト 1 に示す。リスト 1 は、新種の WorkProduct が追加された場合にそれを Activity「レビュー実施」の入力とすることを定義している。行番号 01 ではプロセスアスペクト名を宣言する。行番号 02 では、このアスペクトが weaving されるタイミングを示し、そのタイミングとして任意の WorkProduct 生成後と指定する。行番号 03 ではプロセスモデル上の「レビュー実施」インスタンスを取得している。行番号 04 では、「レビュー実施」にセットする ActivityParameter を生成し、行番号 05～06 で、aWorkProduct を入力と設定する。行番号 07 では上記の ActivityParameter を「レビュー実施」に設定する。このプロセスアスペクトをプロセスモデルに weaving すると、図 6 相当の情報が自動設定されることになる。このようにメタレベル 1 のプロセスアスペクトによりプロセスエンジニアは、UML で定義したプロセスモデルとは分離独立して支援分野の作業を定義することができる。

4.3. メタレベル 0 のプロセスアスペクト例

リスト 2 は、プロジェクト実行時に何か成果物がレビュー可能になると、特定担当者のレビュー実施を開始する規則を定義したプロセスアスペクトである。行番号 01 ではプロセスアスペクト名を宣言する。行番号 02 では、このアスペクトが weaving されるタイミングを示し、そのタイミングとして任意のレビュー対象が isReviewable になった後と指定する。行番号

リスト 1　メタレベル 1 のプロセスアスペクト例

```
01:public M1aspect 成果物レビューアスペクト{
02: after() returning( WorkProduct aWorkProduct ) :
      call( WorkProduct.new(..)){
03:  Activity レビュー実施 =
     ProcessRole.getInstance("レビュー担当者").getActivity("レビュー実施");
04:  ActivityParameter アクティビティ入力 = new ActivityParameter();
05:  アクティビティ入力.kind = "input";
06:  アクティビティ入力.type = aWorkProduct;
07:  レビュー実施.setParameter( アクティビティ入力 );
08: }
09:}
```

リスト 2　メタレベル 0 のプロセスアスペクト例

```
01:public M0aspect レビュー開始アスペクト{
02: after ( Object レビュー対象 ) :
      レビュー対象.getClass().streotype.name == "WorkProduct" &&
         call( レビュー対象.isReviewable(..)){
03:    レビュー担当者.getInstance("三菱太郎").レビュー実施( レビュー対象 );
04: }
05:}
```

03 では「三菱太郎」というレビュー担当者に対してレビュー対象をレビュー実施することを指定する。このプロセスアスペクトが weaving されたプロジェクトでは成果物がレビュー可能になると、例えば図 7 のメッセージ番号 2「レビュー実施(レビュー対象)」に相当する内容が実行されることを意味する。このようにメタレベル 0 のプロセスアスペクトによりプロジェクト管理者は、プロジェクト実行時のスケジュール管理や構成管理のイベントドリブンな情報に応じた規則を定義することができる。

5. おわりに

本稿では、開発プロセスにおけるエンジニアリング分野と支援分野をそれぞれ Core Concern と Crosscutting Concern として捉えた。そして、支援分野のプロセスを、エンジニアリング分野のプロセスであるオブジェクト指向モデルに weaving するアスペクトとして定義可能であることを示した。このようにオブジェクト指向によるプロセスモデルからアスペクトを分離独立させることにより、支援分野のプロセスの保守と再利用を高めることが期待できる。
今後の課題としては、アスペクトとして支援分野のプロセスだけでなく、組織やプロジェクトへ導入する際のその他のテーラリング内容、例えば「計画駆動型とアジャイル型の選択による影響」[10]などをプロセスアスペクトとして定義できないかを検討する。
また本稿ではプロセスアスペクトの定義のみを扱ったが、それを weaving するツール weaver については議論していない。プロセスアスペクトの weaver としては、プロジェクト管理ツールや構成管理ツールとの連携も必要であり、開発プロセスに関わる統合的な計算機支援環境が必要である。

参考文献

[1] 上野浩一郎、アスペクト指向による開発プロセス定義、情報処理学会第 66 回全国大会 1G-4、2004.
[2] Tom DeMarco, “構造化分析とシステム仕様”, 日経 BP, 1986.
[3] Rational Unified Process: http://www-6.ibm.com/jp/software/rational/products/pdf/rup.pdf
[4] L.Osterweil: Software Process Are Software Too, Proc.9th International Conference on Software Engineering, Monterey CA, pp.2-13, April 1987.
[5] Software Process Engineering Metamodel: http://www.omg.org/technology/documents/formal/spem.htm
[6] Ivar Jacobson 他, ”UML による統一プロセス開発プロセス”, 翔泳社, 2000.
[7] Walter L. Hürsch, Cristina Videira Lopes: Separation of Concerns, Northeastern University, Boston(1995).
[8] Pavel Hruby, “Dimensions for the Separation of Concerns in Describing Software Development Processes”, First Workshop on Multi-Dimensional Separation of Concerns in Object-oriented Systems at OOPSLA '99.
[9] AspectJ: http://www.eclipse.org/aspectj/
[10] Barry Boehm & Richard Turner, “アジャイルと規律”, 日経 BP 社, 2004.