第 5 章

オブジェクト進化に基づく組織化過程

5.1 概要

オブジェクトの進化過程を実システム開発事例をもとに調査した結果，ある時点におけるオブジェクトの構造は，オブジェクトの進化の積み重ねによって得られることが明らかとなった．ここでも生物学のアナロジを用いてオブジェクトの組織化について考えてみよう．現在の生物学では，高等な生物が発生するまでには原始生物に始まる何世代にもまたがった進化が必要であると考えられている．ここでオブジェクトの組織化過程を，利用者へシステムが提供されるまでにオブジェクトが形成される過程と定義する．利用者環境に起こる将来のすべての変化を予測することは不可能であるが，利用者環境を時間的変化に伴う変化と利用者の多様化に伴う変化とを区別して考えると，後者の利用者の多様化という変化は分析段階でかなりの程度の予測が可能である．そこで，オブジェクトの組織化過程では，利用者の多様化に適応するための段階的なオブジェクト組織の構築手法を提案する．

従来の OMT 法 [51] などの分析／設計方法論ではアプリケーションのモデルを構築する際に，問題領域に関する情報を収集し，そこからオブジェクトを抽出する手法が一般的に採用されていた [9]．これらの方法論を実際のシステム分析に適用すると，小規模のシステムでも数十から数百のユースケース [29] から，ひとつのオブジェクトモデルを構築することになる．そのため，モデル化の作業は煩雑となり，適切なオブジェクトを選択する基準を決定する根拠が脆弱であり，分析者の技術力による判断に頼らざるをえないという欠点があった．

また，オブジェクトの組織が様々な利用者要求によって進化する漸進型開発では，時間軸上の変化と利用者の多様化という 2 種類の利用者環境への適応を考慮する必要があり，頑健なモデルを構築するための技術的環境への適応，すなわち，開発者への負荷が大きくなるという問題があった．しかし，オブジェクトの組織を構築する際に進化過程を模倣した組織化過程を導入し，段階的な組織化を行えば，大規模で複雑なオブジェクトモデルも系統立てたて構築することが可能となり，漸進型開発では，利用者の多様化への対応を軽減することが可能となる．

本研究では，利用者の多様化に対処するために，時間的に変化すると想定される個々の利用者環境に対応して分析の視点を導入することを検討した．この手法では，視点ごとに問題領域を絞り込んでオブジェクトモデル構築した後，順次統合してシステム全体のオブジェクトモデルを構築するため，規模が大きく複雑な問題領域でも，オブジェクトモデルを構築する際に一定の手順に従って進めることが可能となる．

本章では，多視点を用いたモデル化の手法を組織化過程と呼び，その技術的課題について議論し，解決策を提示するとともに，事例を用いて手法を適用する．

構成は次のとおりである．第 5.2節で視点を定義し，第 5.3節では多視点を用いた分析手法の利点と課題について議論する．課題を解決するための手段については，第 5.4節で，オブジェクトの組織構造の特徴を定義した後に検討する．そして，第 5.5節で組織化過程の基本方針について説明した後，残りの節で組織化過程を適用した事例を示す．

5.2 モデル化の視点

オブジェクトの組織化過程では，オブジェクトの進化に大きな影響を与える時系列で変化する利用者環境を組織化の視点と定める．組織化の視点とは，要求を発する主体，すなわち利用者がシステムを観察する視点と言うこともできる．本研究では，ある視点に基づいて定義したオブジェクトモデルを利用者モデルと呼ぶ．さらに，これらの利用者モデルを統合してシステムのオブジェクトモデルを構築することから，システムのオブジェクトモデルを統合モデルと

呼ぶ.

5.3 多視点の利点と課題

オブジェクトの組織化過程には 2 つの特徴がある．第一の特徴はモデル化に視点を導入することであり，第二の特徴は，進化過程を模倣し，利用者モデルを段階的に統合し，利用者の多様性をモデルに取り込むことである．これらの特徴によって，オブジェクトモデルの分析時に，次のような利点を得ることができると考えられる.

- 視点を定めることによってモデル化するオブジェクトの振る舞いや属性の絞り込みが容易となり，モデル化対象の問題領域の境界を明確にできる.
- 視点が複数存在するため，多様な要求変更に対応できる構造をオブジェクトに持たせることが可能となる.
- モデルの統合作業は，オブジェクトが技術的環境へ適応する過程で行われる技術的な設計変更と同様の作業である．オブジェクトの技術的環境への適応が，将来の変更に対する頑健な構造を形成していたように，開発者は統合作業を進めながら要求変更に対して頑健なオブジェクトモデルを検討することができる.
- 視点ごとに分析者を定めれば，それぞれの分析作業を並行して進めることが可能となる.

利用者モデルを構築する際は，モデル化の視点はひとつであるから，従来の分析方法論を適用することが可能である．しかし，利用者モデルを統合する時点で，いくつかの問題が発生する．多視点を用いたモデル化については，データベースのスキーマ定義に関する研究 [64, 3] や，中谷らの研究 [42] があるが，以下の問題について十分な議論が行われてこなかった.

1. オブジェクト間の情報の比較する際に，対応するオブジェクトを個々の利用者モデルから抽出することが困難である.

オブジェクトが異なる視点からモデル化されると，たとえ形式的に同一であったとしても，それらが対応するオブジェクトであるとは限らない．逆に，オブジェクトが形式的に異なる構造を持っていたとしても，それらのオブジェクトが対応しないとは限らない．OMT 法などを用いて定義された利用者モデルから，現実世界の同じ事物や事象をモデル化したオブジェクトを形式的な情報だけから対応づけるのは困難である．したがって，非形式的な現実世界と形式的なモデル世界とを関連づける記述が必要である．

2. オブジェクト間の衝突を発見するために用いる道具がない．

   対応づけたオブジェクト間で，視点の違いに由来する仕様の差異を解消するためには，オブジェクトの構造や振る舞いに関する情報を形式的に記述できる道具が必要である．

第一の問題に対処するために本研究では，現実世界とモデル世界とを対応づける記述として,「指示」を導入することを試みる．指示とは，現実世界のオブジェクトとモデル世界のオブジェクトの対応関係に着目し，オブジェクトが指し示す現実世界の事物や事象を限定的に表す認識規則とクラス名を表す述語を関係づけた記述であり [28],

$$\text{認識規則} \approx \text{指示されるクラス名}$$

で表現される．指示の左辺には，クラスがどのような現実世界の事物や事象を抽象化したかを知るための認識規則を非形式的な自然言語規則を用いて記述し，指示のためのシンボル$\approx$ の後，右辺にクラス名を用いた述語を記述する．

例えば,

$$eng\text{は開発工程を実施する人} \approx \text{技術者}\ (eng)$$

という指示は，技術者というクラスの要素である engが，開発工程を実施する人を指していることを意味する．特定の問題領域を分析している分析者や問題領域の専門家には，左辺の認識規則を理解して，右辺のクラス名を用いた述語によって指示される対象を識別可能である．もし，指示に定義された認識規則によって現実世界の事物や事象を限定できないとしたら，それは指示の記述が不完全であることになる．

第二の問題に対しては，指示の記述を含めたオブジェクトの性質を定義するためのオブジェクト辞書を用いて対処する．オブジェクト辞書には，利用者モデルに対応する利用者辞書と統合モデルに対応する統合辞書を用意する．統合辞書は，利用者モデルの統合作業と並行して利用者辞書を統合して定義する辞書である．オブジェクト辞書の詳細な構造については，第 5.4節で紹介する．

5.4 オブジェクトの組織

5.4.1 オブジェクトの組織構造の特徴

オブジェクトの組織という観点から，現実世界とモデル世界の利用者モデルの対応関係に基づき，利用者モデルが持つ構造の特徴を議論しよう．

組織化過程では，利用者モデルに定義されるクラス群を，利用者オブジェクトの推移的な参照組織の集合として定義する．ここで，参照組織を，注目するクラスが関連を持つクラスの集合と，直系の継承クラスの集合の和集合と定義する．図 5.1に，クラス c_xの参照組織を示す．図の表記には OMT 法を用いた．図中で，網かけをしたクラスがクラス c_xの参照組織を構成するクラス群である．

ここで，$R(c_1, c_2)$ はクラス c_1がクラス c_2への関連を保持していることを意味する述語，$H(c_1, c_2)$ はクラス c_1がクラス c_2の直下のサブクラスであることを意味する述語とする．ただし，R, Hには，対称律を仮定しない．H^*を Hの推移的閉包としたとき，クラス c_xの参照組織を構成するクラスの集合 $I(c_x)$ は，クラス c_xの関連クラスの集合 $IR(c_x)$ と継承クラスの集合 $IH(c_x)$ を用いて次のように定義される．

$$IR(c_x) = \{c' | R(c_x, c')\}$$
$$IH(c_x) = \{c' | H^*(c_x, c') \vee H^*(c', c_x)\}$$
$$I(c_x) = IR(c_x) \cup IH(c_x)$$

M_uを利用者モデルに定義されたクラスの集合としたとき，M_uの利用者オブジェクトを $c_{vp}(M_u)$ と表す．ここで，M_uの視点はただ一つであるから，M_uには利用者オブジェクト $c_{vp}(M_u)$ がただ一つだけ定義されることになる．

M_uの任意のクラス c は，利用者の視野に含まれる現実世界のオブジェクト

をモデル化したものである．従って，M_uの利用者オブジェクト $c_{vp}(M_u)$ とクラス c との間には，直接的，あるいは間接的にメソッドを呼び出す関係が存在すると仮定してよい．このような関係には，クラス間の関連や継承がある．ここでは，これらの関係をまとめて，クラス間のメソッド呼び出し関係と定義する．$RH(c_1, c_2)$ をクラス c_1とクラス c_2にメソッド呼び出し関係があることを意味する述語としたとき，

$$RH(c_1, c_2) = R(c_1, c_2) \vee H^*(c_1, c_2) \vee H^*(c_2, c_1)$$

となり，従って，

$$I(c) = \{c' | RH(c, c')\}$$

が成り立ち，RH^*を RHの推移的閉包として，$c_{vp}(M_u)$ と M_uの任意の要素 c との間に存在する関係は，

$$\forall c \in M_u \cdot RH^*(c_{vp}(M_u), c)$$

とならなければいけない．この式は，利用者モデル内の任意のクラスが，利用者オブジェクトの推移的な参照組織の要素となることを表している．さらにこの式は，利用者オブジェクトから，その参照組織を順に辿って統合作業を進めれば，全クラスの統合作業が完了できることを保証している．

異なる 2 つの利用者モデルに定義されたクラスが同一であるということは，クラスがモデル化している現実世界の事物や事象が同一であるということである．しかし，視点が異なるとモデル化されたクラスの属性や振る舞い，参照組織といった性質が一致するとは限らないし，粒度が異なる場合もある．そのため，モデルを統合する際には，異なる利用者モデル間に定義された同一のクラスを抽出し，逐一それぞれの性質に起きている仕様の衝突を発見して解消する必要がある．

5.4.2 衝突の種類

利用者モデルを統合する際は，対応するクラスの要素を比較し，図 5.2に示した順に衝突を発見して解消する．クラスが持つ性質には依存関係を持っているものがあるため，図 5.2には，それぞれの性質の依存関係を矢印で示してある．図中の四角形はクラスの性質を表し，矢印は，各クラスの要素間の制約関

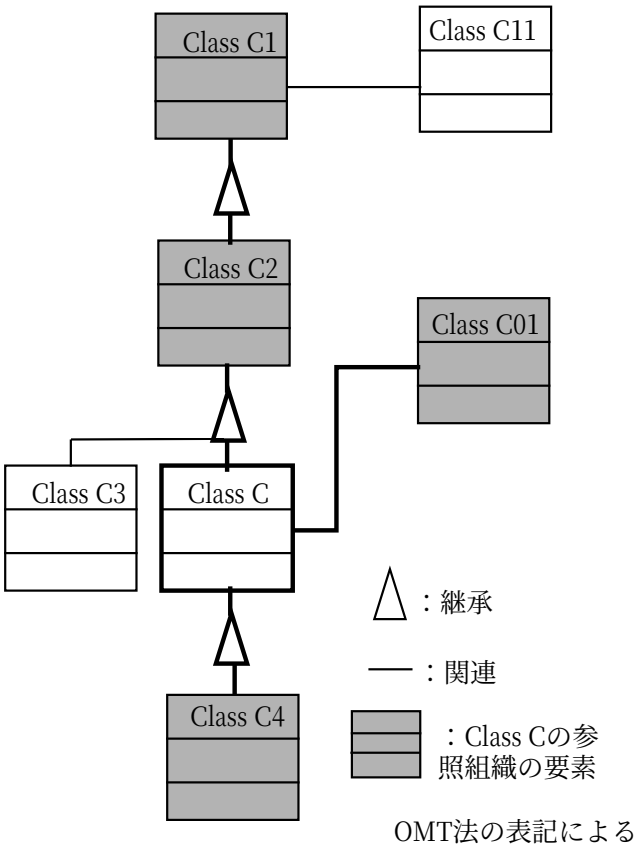

図 5.1: オブジェクト Class C の参照組織

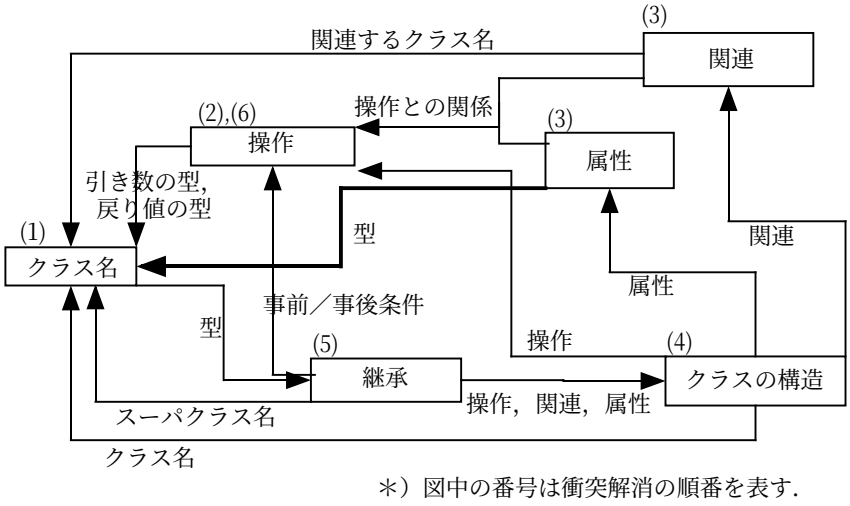

図 5.2: クラスの性質の依存関係

係を表す．たとえば，継承は，クラスの構造，スーパクラス名，操作の事前条件，事後条件が決まることによって決定される [36]．また，図中の番号は，クラスの性質を照合して仕様の衝突を解消する手順を表している．番号が 2 つつけられているものは，仕様の衝突の検証作業に繰り返しがあることを意味している．

統合時に問題となる衝突には，名前の衝突と構造の衝突が考えられる．名前の衝突とは，異なる対象に対して同じ名前が使われている，あるいは，同じ対象に対して異なる名前が使われているといったクラス間の矛盾をいう．この衝突はクラス名，操作名，属性名，関連名に起こりうる．

構造の衝突は，異なる視点で定義された同じ事物や事象を指示する 2 つのクラスにおいて，操作や属性といった性質，あるいは参照組織が一致しないことを指す．構造の衝突で，特に継承構造が異なる場合を継承の衝突という．継承の衝突は，オブジェクトの分類方法や抽象度の違いによって発生する可能性がある．

図 5.2 に示した性質の依存関係は，クラス間で発生している性質の衝突を解消する順番でもある．したがって，継承の衝突を解消するのはクラスの性質の衝突解消の最後となる．

以上の衝突を発見する道具として，利用者モデルのクラスの性質を定義する

オブジェクト辞書を用いる.

5.4.3 オブジェクト辞書の構成

オブジェクト辞書には次の 2 つの役割がある.

- クラスが現実世界の同一の事物や事象をモデル化したものであることを識別するための情報を定義できること.
- クラスの性質間の衝突を発見するための形式的な情報を定義できること.

これらの役割を満足するために，オブジェクト辞書に登録されるクラスには次の項目の情報を定義する.

1. 指示とクラス名

   クラスがデータ抽象した現実世界の事物，事象を指し示す情報で，以下の述語で表現する.

   $$\text{認識規則} \approx \text{指示されるクラス名}$$

2. 操作に関する情報

   操作は，オブジェクト間の契約のもとで実行されるから，辞書には操作のシグネチャと，契約のための表明として事前条件および事後条件 [39] を定義する．これらの情報を比較して，対応する項目の内容が等しければ，それらの操作を同一と判定する.

3. 属性および関連に関する情報

   責任駆動型設計 [62, 63] では，クラスの属性および関連はクラスの責任を果たすために定義されると考える．だから，辞書には属性や関連の意味を説明する情報として，属性の型および関連づけられたクラスの名前とともに，その属性や関連が定義された目的をクラスの操作との関係によって定義する．ここで，属性や関連とクラスの操作との関係が一致すれば同一の属性や関連と判定する.

4. 継承に関する情報

   継承を説明するために，辞書には継承するクラスの名前を定義する．継承の衝突を解消する時点で，クラス名の統一が終わっていれば，継承するクラスの名前が一致するだけで同一の継承構造であると判定できる．

利用者モデルを定義する際に使用する利用者辞書に記述する項目を次に例示する．

.....

『オブジェクト名』

- 所有者名: 利用者モデル名
- 指示: 認識規則 ≈ 指示するオブジェクト名
- ＜参照組織＞
  - 継承するオブジェクト名:

    このオブジェクトが責任を果たすために必要な属性と関連として，以下，参照組織を構成する属性数および関連の数だけ繰り返す．
  - 属性名または関連名
    - 属性の型，または参照するクラス名
    - 操作との関係: 属性，または参照関連が定義されている目的
- ＜操作＞

  このオブジェクトが他のオブジェクトに提供することを契約する操作として，以下を操作の数だけ繰り返す．

  操作名（引数の型の並び）: 戻り値の型
  - 事前条件:
  - 事後条件:

統合辞書の構造は利用者辞書とほぼ同じであるが，所有者の項目に，統合されたオブジェクトを定義したすべての利用者モデル名を記述する点に違いがある．Rubin と Goldberg らが提唱した OBA(Object Behavior Analysis) で

は，各作業段階で作成する表に情報の由来を説明する項目がある [50]．複数のツールを用いて分析作業を行う場合には，このような情報の由来を説明する項目が必要である．オブジェクト辞書に記述する所有者項目は，統合モデルのオブジェクトの由来を説明し，利用者モデルのオブジェクトと対応づけるための項目である．統合モデルと利用者モデルを関係付けることができれば，複雑化した統合モデルを理解する際に関連する利用者モデルを参照して理解を助けることが可能であるし，統合モデルのレビューに利用者モデルを使うこともできる．また，統合モデルや利用者モデルを変更した際，所有者項目を手がかりにして，その変更箇所を他のモデルに反映させる作業を進めることができる．

5.5 組織化過程の概要

以上のオブジェクトの組織化過程をまとめると次のようになる．

1. システムを構成するアプリケーションプログラムの利用者を発見し，その利用者が問題領域を観察する視点を分析の視点と定める．

2. 分析者は担当する視点から現実世界を観察し，利用者モデルと利用者辞書を定義する．

3. 利用者辞書を順次選択し，統合辞書に記述されたクラスを指示に基づいて照合し，参照組織の衝突を発見し解消して統合モデルと統合辞書を更新する．

   （3-1）クラス名を統一する．
   （以下の作業をクラスの参照組織を辿りながらクラス数繰り返す）
   （3-2）クラスごとに操作名の衝突を解消する．
   （3-3）クラスごとに属性名および関連名の衝突を解消する．
   （3-4）クラスごとに構造の衝突を解消する．
   （3-5）継承関係を持つクラスについて，継承の衝突を解消する．

クラス間の衝突を解消する過程は，一方のクラスに新たな仕様を取り込ませる設計作業であるから，オブジェクトが利用者環境の変化に適応する過程で行

われる設計作業と同じである．利用者モデルの統合時には，開発者が問題領域への理解を深めることによって，より適切なモデル構造を発見して定義することも可能である．この作業は，オブジェクトの進化過程では技術的環境への適応で行われる設計作業と同じである．第2で議論したように，オブジェクトは技術的環境への適応の過程で，より利用者環境に適応しやすい構造を取得していく．したがって，段階的な組織化過程を経ることによって，利用者へシステムが公開された後も，より利用者環境へ適応しやすい構造をオブジェクトに持たせることが可能となる．

5.6 事例

本研究で提案した組織化過程の妥当性を検証するために，ソフトウェア開発管理システムの組織化を行った．

5.6.1 システム概要

ソフトウェア開発管理システムの利用者として，プロジェクト管理者，開発者，構成管理者，品質監査者を想定する．各利用者は次の要求を持っている[27].

- プロジェクト管理者

  プロジェクトの計画立案，要員の割り当て，進捗管理，成果物の検証，妥当性確認のレビューを行い，プロジェクト進捗上の問題発見と問題解決，再発防止のための開発工程の是正を行えるようにプロジェクトに関するデータを進捗にしたがって随時参照したい．

- 開発者

  割り当てられた開発プロジェクトの業務や他の業務をスケジューリングし，開発業務を終了したら速やかに次の作業に取りかかりたい．

- 構成管理者

  プロジェクトの記録や文書，ソフトウェア部品などの成果物を適切に配布し，保管し，廃棄できるように成果物の版管理を行いたい．更に，成

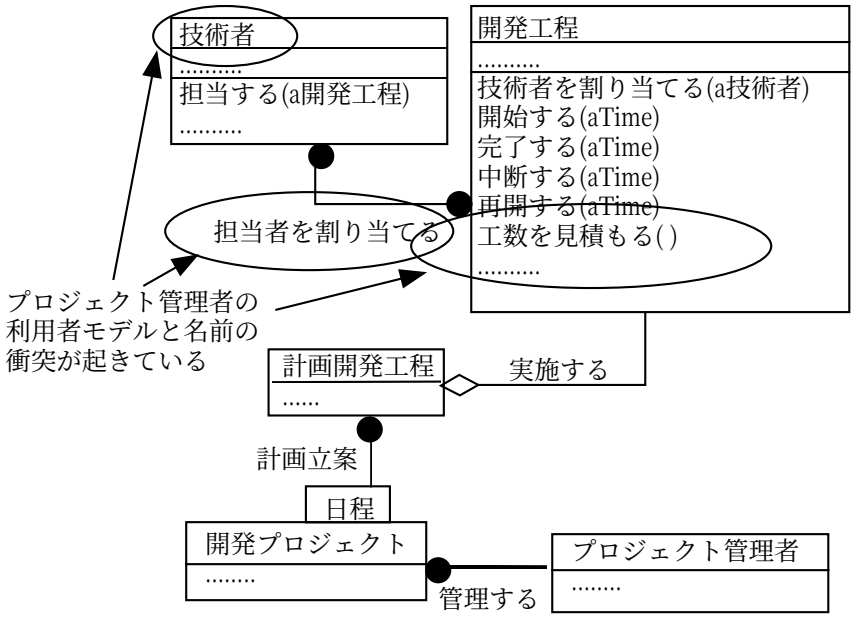

図 5.3: プロジェクト管理者から見た開発工程の参照組織

果物がどのような成果物を参照して生成されたのかといった文書間の関係も管理したい.

- 品質監査者

  構成管理が適切に行われていることを監査しなければならないので，プロジェクトの開発プロセスがその時点の最新の成果物を参照して実施されていたことを調べたい．また，業務の担当者が品質マニュアルに記述されている基準を満たしていることも確認したい.

5.6.2 衝突の発見

クラス名の衝突と解消

組織化手順にしたがって，最初にクラス名の統一を行う．プロジェクト管理者と開発者の利用者辞書を以下に示し，それぞれの利用者モデルに定義された，**開発工程**の参照組織を取り出して図 5.3と図 5.4に示した.

.....

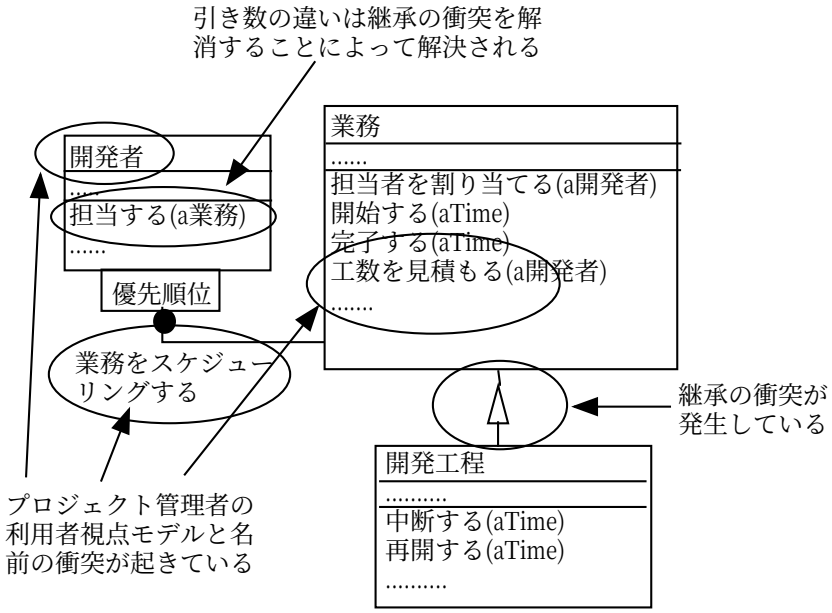

図 5.4: 開発者から見た開発工程の参照組織

『技術者』

- 所有者: プロジェクト管理者モデル
- 指示: engは開発工程を実施する人 ≈ 技術者 (eng)

.....

『開発者』

- 所有者: 開発者モデル
- 指示: devはソフトウェアを開発する人 ≈ 開発者 (dev)

.....

プロジェクト管理者と開発者の利用者辞書に定義された指示の意味を解釈すると，図 5.3の**技術者**と図 5.4の**開発者**は現実世界の同じ事物を指していることがわかる．そこで，この名前の衝突を解消するために，**技術者**を**開発者**に変更してクラス名を統一する．

1. 属性名および関連名の衝突と解消

プロジェクト管理者のモデルに定義された**開発工程**と**技術者**の間にある関連は，開発者のモデルに定義された**開発者**と**業務**との間の関連と意味が一致するので，関連名に衝突が起きていることになる．ここでは，名前を統一して衝突を解消する．名前の衝突を解消するにあたって，開発者のモデルに定義されていた限定子や多重度を調整する．

2. 構造の衝突と解消

図 5.3と図 5.4に示した開発者モデルの**開発者**およびプロジェクト管理者モデルの**技術者**，それぞれの参照組織を比較すると，属性と操作および関連の集合が一致しない構造の衝突を発見できる．この例では既に関連名の衝突は解消されているから，関連の集合を合成するだけでよい．属性名と操作名についても同様に，それぞれの集合を合成して構造の衝突を解消する．

3. 操作名の衝突と解消

開発工程と**業務**の間の継承の衝突を解消する前に，**開発工程**に発生している操作名の衝突を解消しよう．次に関係する利用者辞書の一部を示す．

.....

『開発工程』

- 所有者: プロジェクト管理者モデル
- 指示:

  $proc$ はソフトウェア開発を管理するために区切られた工程

  $\approx$ 開発工程 ($proc$)
- ＜参照組織＞
- 継承するオブジェクト名: なし

.....

- ＜操作＞

  工数を見積もる（）: 日数
  - 事前条件:
  - 事後条件:

.....

『業務』

- 所有者: 開発者モデル
- 指示: $task$は，人によって遂行される業務 ≈ 業務 $(task)$
- ＜参照組織＞
- 継承するオブジェクト名: なし

.....

- ＜操作＞

  工数を見積もる（技術レベル）: 日数

  – 事前条件:

  – 事後条件:

プロジェクト管理者はプロジェクトの計画立案を行う際，企業の標準見積り技術を使って開発工程の工数を見積もる．これに対して開発者は，自分の担当業務をスケジューリングするために，自分の技術レベルで必要となる業務の工数を見積もる．したがって，2 つのモデルを単純に合成すると，プロジェクト管理者の**開発工程**と開発者の**業務**のサブクラスである**開発工程**は，それぞれ**工数を見積もる**という名前の衝突を起こす操作を持つことになってしまう．この例では，**業務**の**工数を見積もる**を**実質工数を見積もる**に変更し，プロジェクト管理者の**開発工程**に定義されていた**工数を見積もる**を**標準工数を見積もる**に変更する．

4. 継承の衝突と解消

   プロジェクト管理者の**開発工程**と開発者の**開発工程**の属性の集合と操作の集合を合成し，**業務**のサブクラスとして**開発工程**を定義し，開発者とプロジェクト管理者の**開発工程**との間に発生している継承の衝突を解消する．

5. 統合の検証

   図 5.5に以上の手順で統合した**開発工程**の参照組織を示す．この統合モデルは開発者とプロジェクト管理者といった利用者モデルを統合したモデ

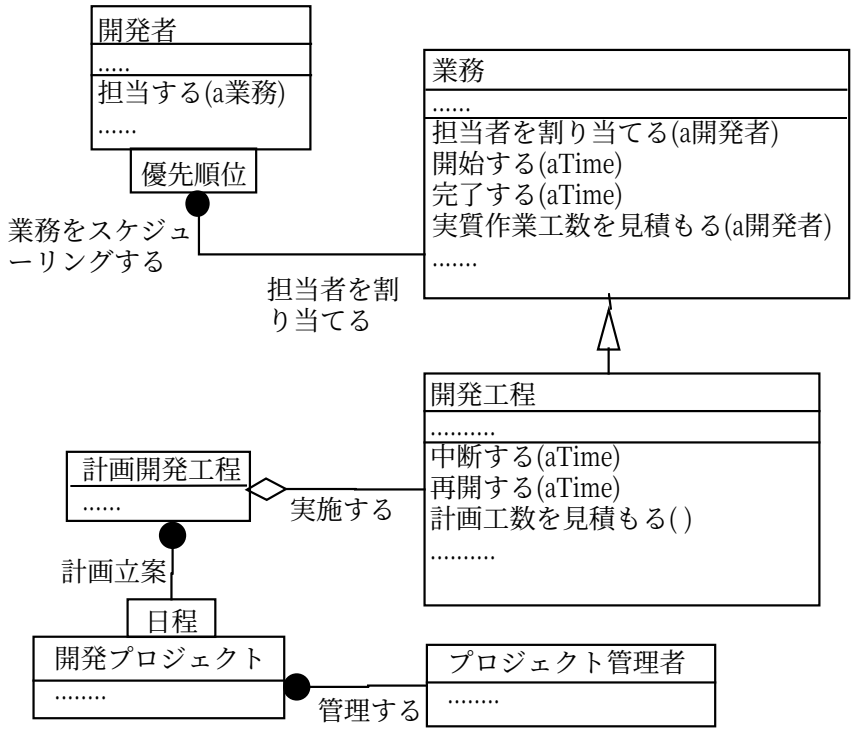

図 5.5: 衝突を解消した結果

ルであるから，**開発者**と**プロジェクト管理者**の利用者オブジェクトが含まれている．図5.5では，これらの利用者オブジェクトから統合モデルの検証を始め，参照組織を辿って，他のすべてのクラスの性質が正しく統合モデルに反映されているかを検証する．

6. その他の構造の衝突：関連の衝突と解消

その他の構造の衝突の例として，構成管理者と品質監査者の利用者辞書の中から，**記録**のサブクラスである**文書**の参照組織を示して説明する．ただし，**記録**とは，版管理されないプロジェクトの成果物を指示するオブジェクトである．

.....

『文書』

- 所有者: 構成管理者モデル

- 指示:

  doc は，版管理されているプロジェクトの成果物 ≈ 文書 (doc)

- ＜参照組織＞
- 継承するオブジェクト名: 記録

  .....

- ＜操作＞

  参照記録を得る（）: 記録の集合

  - 事前条件:
  - 事後条件:

    戻り値 $Rec = \{rec|$ 参照する $(self, rec)\}$

    ($self$は，ここで定義しているオブジェクトを指す）

  .....

『文書』

- 所有者: 品質監査者モデル
- 指示:

  doc は版管理されている開発工程の成果物 ≈ 文書 (doc)

- ＜参照組織＞
- 継承するオブジェクト名: 記録

  .....

- ＜操作＞

  参照記録を得る（）: 記録の集合

  - 事前条件:

    $\exists proc \cdot$ (新規に生成する $(proc, self)$

    $\vee$更新する $(proc, self))$

  - 事後条件: 戻り値 $Rec = \{rec|$

    $\exists proc \cdot ($(新規に生成する $(proc, self)$

    $\vee$更新する $(proc, self)) \wedge$ 記録を参照する $(proc, rec))\}$

構成管理者の文書オブジェクトに定義された**参照記録を得る**と，品質監査者の文書オブジェクトに定義された**参照記録を得る**は，事前条件，事

後条件は一致しないが，文書と記録は開発工程の成果物を指示しているから，両者の**参照記録を得る**の意味が等しいと解釈できる．したがって，次の関係が成り立つ．

参照する（doc, rec）

$= \exists proc \cdot ((\text{新規に生成する}\ (proc, doc)$

$\vee \text{更新する}\ (proc, doc)) \wedge \text{記録を参照する}\ (proc, rec))$

すなわち，ある文書 doc が記録 rec を参照するということは，記録 rec を参照し，かつ文書 doc を新規に生成するか，あるいは更新する開発工程 $proc$ が存在するということである．したがって，構成管理者の文書間の参照関連を品質監査者の開発工程を経由したモデルで置換すれば，構造の衝突を解消できる．

5.7 考察

以上の手順を経る際に定義したプロジェクト管理者，開発者，構成管理者，品質管理者の利用者モデルを付録Dに掲載した．各利用者モデルを統合して構築した統合モデルを図5.6に示す．

従来の分析方法論を用いると，最初のモデル化のゴールが図5.6のようなモデルを構築することとなる．しかし，組織化過程を用いると，最初のゴールとして限定された問題領域をモデル化対象に設定できるだけでなく，図5.6に向けて，段階的に各利用者モデルを統合する作業を進められるため，モデル化の生産性と精度を従来の方法よりも高めることができると思われる．

技術的には，モデルと現実世界の対応関係を指示によって明示でき，モデルの意味を記述したオブジェクト辞書を用いることによって，視点ごとに定義された利用者モデル間の衝突の発見と解消を容易にできた．また，オブジェクト辞書の所有者項目には，統合モデルと利用者モデルとを対応づける情報を定義するため，統合モデルと利用者モデルの対応関係を管理できる．さらに，レビューに参加する利用者にとって理解可能な利用者モデルが構築されるため，分析の信頼性を向上させることも期待される．

今後は，この組織化過程に準拠した分析手法を開発現場に導入し，漸進的開発に対する効果について研究を進めていきたい．

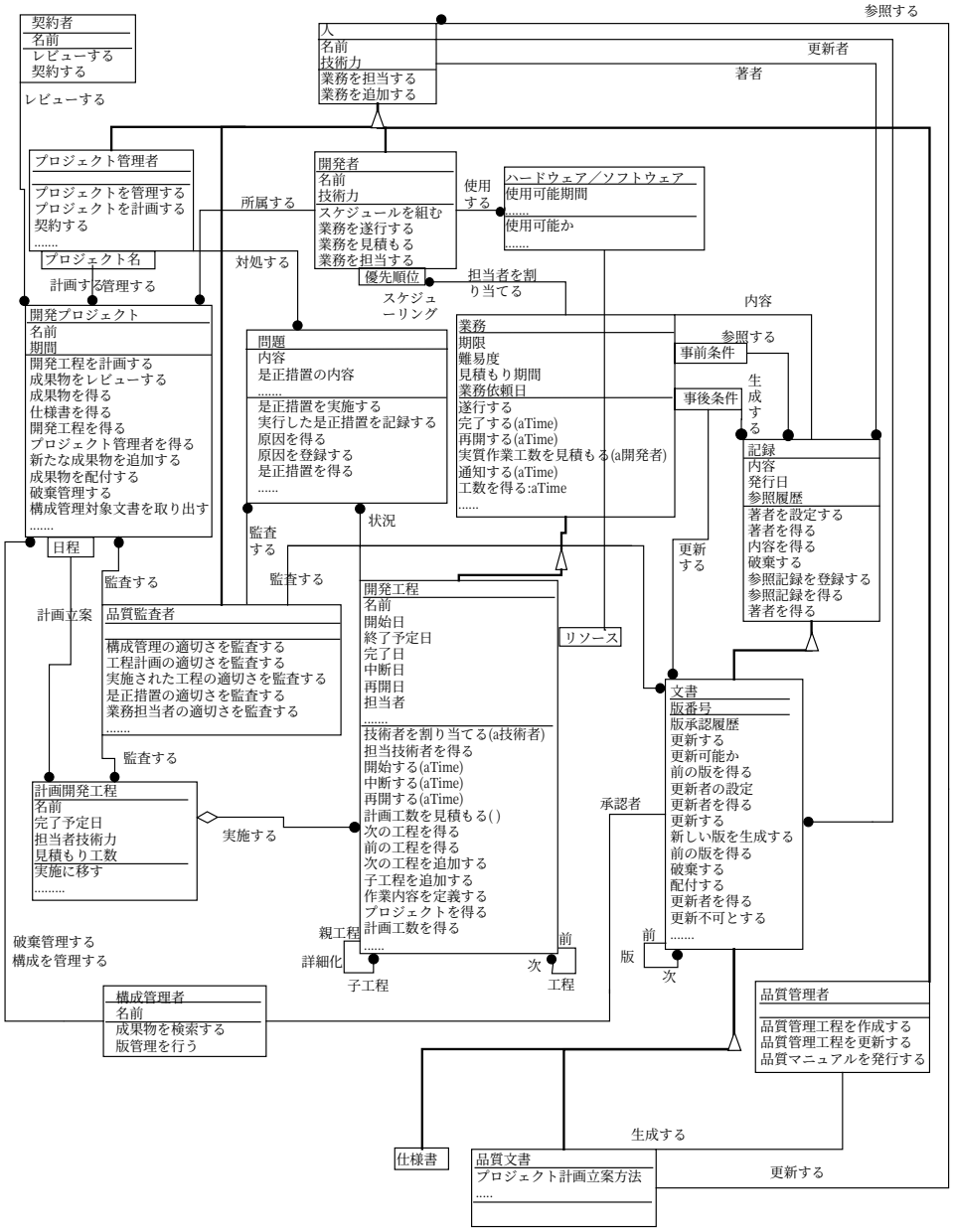

図 5.6: ソフトウェア開発管理システムの統合モデル