オブジェクト指向原価計算の基本構造

尾　畑　　裕

1　はじめに

一般的に製品原価を計算するシステムは，製品の原価をひとつの要約数字として表す．製品の原価を計算するには，製造間接費の配賦をはじめ，多くの仮定が必要であるにもかかわらず，原価情報の利用者には，要約された数字だけが手渡されるのであり，情報利用者にとって，その原価がどのようなプロセスで計算されたのか，どのような仮定をもって計算されたのかという事実が隠されている場合も多いのである．

製品という1つの要約された数字のみを媒介にして原価情報作成者と原価情報利用者が接点をもつ現在の製品原価計算のありかたに，現在の製品原価計算の限界があると感じている．もし，原価情報利用者が製品原価の構成情報を直感的に取り出すことができ，簡単な操作で特定の資源の価格を変更して製品原価を再計算するシミュレーションを行うことができたならば，原価情報利用者の製品原価にたいする理解がずいぶんと深くなると思う．原価情報利用者が特定の製品を指定して，その構成情報，内訳情報など，その製品に関する特定の情報をリクエストすると，リクエストに応じた答えが得られる．そのように原価情報利用者とくに経営者が主体的に原価情報にかかわるようになるのが理想的である．もちろん原価の構成情報，内訳情報は，あらかじめそういった情報を出力するようにプログラムしておけば，製品原価情報とともに詳細な補足情報として出力することは可能であろう．ただ，その結果，辞典のような冊子が出力されてきたのでは，原価情報利用者は，とてもそれらの詳細情報を参照する気がしなくなってしまう．

原価情報利用者の主体的問いかけに対してシステムが明快に答えてほしいのである．

このような原価情報利用者による直感的操作に対応し，原価の透明性を高めるために，オブジェクト指向の考え方で原価計算をモデル化するのが非常に役立つ．オブジェクト指向プログラミングでいうオブジェクトとは，データとメソッドをカプセル化し，内部構造を外部から隠蔽するとともに，厳密に定義されたインターフェイスを通じて，他のプログラム部分と通信できるようにしたものである．オブジェクト指向原価計算モデルでは，もはや原価は要約された１つの数字ではなく，その背後に他の構成要素への参照をもち，情報利用者からのさまざまな問い合わせに答えるすべを知っている知的な構造体となる．

2　オブジェクト指向原価計算の特徴

オブジェクト指向原価計算の基本的な特徴は以下のようにまとめることができる．この詳細については，本論文のなかで取り上げていくことにする．

① オブジェクト指向原価計算における原価とは，原価計算対象に集計された要約数値のみを意味するのではなく，インプットとアウトプットの関係の階層構造そのものを内部構造に包含した複雑な構造体であり，原価計算は要約情報のみならず，必要に応じて要約される前の情報も提供できるべきである．

② オブジェクト指向原価計算では資源の物量的消費情報が骨組みを形成する．原価計算の原子的構成要素は，資源消費である．

③ オブジェクト指向原価計算は，原価計算対象階層とプロセス階層をもつ．ただし原価計算対象階層は１つとはかぎらず，必要に応じて代替的原価計算対象階層を構築できる．原価計算システムは，原価計算対象階層を中心としたViewとプロセス階層を中心としたViewを提供すべきである．

④ 資源消費は，少なくとも１つの原価計算対象階層と少なくとも１つのプロセス階層の両方に帰属する．

⑤ オブジェクト指向原価計算は十分な透明性をもつべきで，最終原価計算対

象からもプロセスについての情報，末端の資源消費についての情報が見えるべきである．また原価計算対象から，プロセスについての情報を取り出したり，プロセスから原価計算対象についての情報を取り出したりできなければならない．

⑥　オブジェクト指向原価計算システムは，ユーザーがコストについて直感的洞察を得ることを支援するものでなければならない（原価計算対象ナビゲータ，プロセスナビゲータの採用など）．

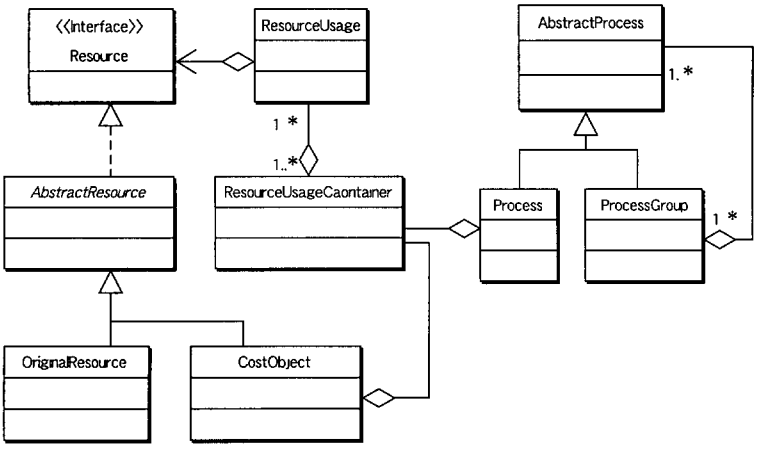
3　オブジェクト指向原価計算の全体構造

製品原価，製品原価計算をビジネスオブジェクトとしてモデリングする方法にはさまざまな方法があると思うが，筆者が考えるオブジェクト指向原価計算の基本構造を示してみよう．図1は，統一モデリング言語 UML で表現したオブジェクト指向原価計算の基本構造である．オブジェクト指向原価計算の全体構造を示すために各クラスの属性や操作は省略してある．

図1のクラス図のなかにある，資源（Resource）インターフェイス，資源消費（ResourceUsage）クラス，資源消費コンテナ（ResourceUsageContainer）クラス，原価計算対象（CostObject）クラスの4つは，オブジェクト指向原価計算の核となるもので重要である．これら4つの間の関係には，コンポジットパターンの変形が使われている．資源インターフェイス，資源消費クラス，資源消費コンテナクラス，原価計算対象クラスについては，本論文でそれぞれ1節づつを割り当てて説明することにする．ここでは，材料や労働力のようなオリジナル資源も，部門サービスや部品のような原価計算対象も，資源インターフェイスを実現しており，消費する側からみると資源型として同一視されるという点だけを強調しておこう．

なお，以下のクラス図における Resource インターフェイスを実現する AbstractResource という抽象クラスは，抽象骨格実装（skeletal implementation）であり，オリジナル資源クラス，原価計算対象クラスに共通する処理をまとめたものである．

図1　オブジェクト指向原価計算の基本構造を表すクラス図（UML 表記）



また，資源消費コンテナクラスとプロセスクラスの関係も重要である．

なお，以下の説明では，Java 言語による実装を前提として議論していくことにする．そのため UML での表現による属性，操作にかえて，フィールド，メソッドという言葉を使う．

4　資源消費クラス——原価計算の原子的構成要素——

オブジェクト指向原価計算の核となる資源（Resource）インターフェイス，資源消費（ResourceUsage）クラス，資源消費コンテナ（ResourceUsageContainer）クラス，原価計算対象（CostObject）クラスのなかでも，とりわけ重要なクラスは，資源消費クラスである．原価計算は，資源消費の事実を原価計算対象に結びつけるしくみといっても過言ではない．資源消費には，消費する対象である資源型のオブジェクトへの参照を保持するためのフィールドと，その資源の消費量を保持するためのフィールドがある．もちろん，消費対象である資源型オブジェクトへの参照を取り出すためのメソッドや消費量を取り出すためのメソッドを有していることはいうまでもない．

資源消費は，原価計算対象クラスのオブジェクトに関連付けれるとともに，その資源消費が行われるプロセスにも関連づけられる．

資源消費オブジェクトは，資源消費額を計算するために，calcCost()メソッドをもつ．

5　資源消費コンテナ

資源消費は，原価計算対象に関連づけられる．原価計算対象オブジェクトが，そのフィールドに，資源消費オブジェクトへの参照を保持することにより，資源消費オブジェクトと原価計算対象が関連付けられる．しかし，1つの原価計算対象オブジェクトは，複数の資源消費オブジェクトへの参照を保持する必要がある．たとえば製品という原価計算対象は，複数の部品や材料からなりたっている．部品という原価計算対象も，下位の部品や材料から成り立っている．

複数の資源消費オブジェクトへの参照を保持するためには，Java でいえば，Vector のような入れ物が必要になる．ここでは，資源消費コンテナという専用の入れ物を用意することにする．

その資源消費コンテナオブジェクトへの参照を，原価計算対象オブジェクトが保持したり，あるいは，プロセスクラスのオブジェクトが保持したりする．また，資源消費コンテナは，資源消費の集合であるから，原価を構成する内訳情報を表現する手段としても利用できる．すなわち原価計算対象クラスのオブジェクトから呼び出される各種の問い合わせメソッドの返し値としても利用できる．

資源消費コンテナには，以下のような基本演算が定義される必要がある．

① 資源消費コンテナオブジェクトに資源消費オブジェクトを加える
② 資源消費コンテナオブジェクトから資源消費オブジェクトを差し引く
③ 資源消費コンテナオブジェクトに資源消費コンテナオブジェクトを加える
④ 資源消費コンテナオブジェクトから資源消費コンテナオブジェクトを差し引く
⑤ 資源消費コンテナオブジェクトに float 型をかける

具体的には以下のようなイメージである．

①資源消費コンテナオブジェクトと資源消費オブジェクトの加算

((材料A, 5個), (材料B, 6個), (材料C, 3個))＋(材料A, 2個)

＝((材料A, 7個), (材料B, 6個), (材料C, 3個))

②資源消費コンテナオブジェクトと資源消費オブジェクトの減算

((材料A, 5個), (材料B, 6個), (材料C, 3個))－(材料A, 2個)

＝((材料A, 3個), (材料B, 6個), (材料C, 3個))

③資源消費コンテナオブジェクトと資源消費コンテナオブジェクトの加算

((材料A, 5個), (材料B, 6個), (材料C, 3個))＋((材料A, 3個), (材料C, 2個))

＝((材料A, 8個), (材料B, 6個), (材料C, 5個))

④資源消費コンテナオブジェクトと資源消費コンテナオブジェクトの減算

((材料A, 5個), (材料B, 6個), (材料C, 3個))－((材料A, 3個), (材料C, 2個))

＝((材料A, 2個), (材料B, 6個), (材料C, 1個))

⑤資源消費コンテナオブジェクトと float 型との乗算

((材料A, 5個), (材料B, 6個), (材料C, 3個))×3

＝((材料A, 15個), (材料B, 18個), (材料C, 15個))

上記の演算は，あくまでイメージを説明したものであり，正確ではない．正確には，資源消費オブジェクトは，資源型オブジェクトへの参照と消費量からなっているのであり，たとえば，（材料 A，5個）というのは，材料 A という資源への参照と消費量 5 個をもつ資源消費オブジェクトという意味である．

なお，資源消費コンテナクラスに，専用のイテレータをつけ，資源消費コンテナクラスのオブジェクトが収容しているすべての資源消費オブジェクトにたいしてある特定のメソッドを呼び出すなどの処理を簡単に行えるようにする．また，収容しているすべての資源消費オブジェクトについての情報を資源消費オブジェクト ID 順に出力したり，資源消費オブジェクトのそれぞれについて単価×消費量の計算をしてその結果を累積して返す calcCost()のようなメソッドを用意しておく．このようなメソッドを資源消費コンテナクラスに用意しておくと，資源消費コンテナクラスのオブジェクトへの参照をみずからのフィールドにもつ原価計算対象クラスにおいては，calcCost()メソッドは，資源消費コンテナクラスの

オブジェクトのcalcCost()メソッドを呼び出して，その返し値を返すだけですむ．すなわち，本来は原価計算対象クラスが備えるべきコスト計算や出力などの機能を，資源消費コンテナクラスに委譲させることができる．

なお，資源消費コンテナクラスは，原価計算対象クラスが，資源消費オブジェクトへの参照を保持する入れ物として機能するのみならず，プロセスクラスが資源消費オブジェクトへの参照を保持する入れ物としても機能する．また，各種の原価の構成要素についての問い合わせの返し値の型としても効果的である．

6　原価計算対象クラス

ここで，原価計算対象とは，通常考えられているような製品や部品のみに限定されない．製造部門，補助部門，部門共通費のグループなども，原価計算対象と考えることができる．このように考えるとあらゆる資源消費は，なんらかの原価計算対象に直接的に関連付けられるともいえるのである．

原価計算対象クラスは，原価計算対象IDフィールド，資源消費コンテナクラスへの参照フィールド，アウトプットフィールドをもつ．アウトプットフィールドのほかに，予定アウトプットフィールド，実績アウトプットフィールド，キャパシティーフィールドなどのフィールドも必要になる．

アウトプットフィールドが必要なのは，原価計算対象クラスのオブジェクトと関連付けられる，資源消費オブジェクトが保持する消費量が，原価計算対象のどれだけのアウトプットに対応するものであるかを明示する必要があるからである．対応するアウトプットの指定がない消費量情報はナンセンスである．

実績計算ではなく，予定計算が目的で，しかも資源消費として自製部品および買入部品の消費，材料の消費のようなものを考えるならば，原価計算対象クラスのオブジェクトのアウトプットフィールドにセットされる値は1が便利である．材料仕様書は，製品1単位，部品1単位の製造に必要な消費量が規定されている．このような場合を想定するならば，アウトプット＝1は自明のものとして，あえてアウトプットフィールドを用意する必要はないと思うかもしれない．しかしながら，製造部門，補助部門などの原価計算対象を考えてみると，資源の消費は，

それら部門の提供するサービス1単位あたりに予定することは不可能である．むしろ，一定量のサービス（期待実際操業度など）を提供する場合の各種資源の消費量が予定されるのが普通である．もっとも，変動的資源と固定的資源をわける場合には，変動的資源の消費量は，部門のアウトプット1単位あたりに予定し，固定的資源消費は，部門の予定操業度を前提に，予定することになる．

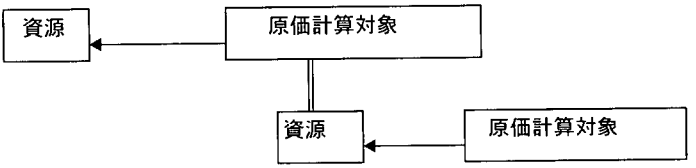
一般的にいえば，変動的要素の消費は，アウトプット1単位あたりに予定し，固定的資源の消費は，予定提供アウトプット全体にたいして予定するということになる．部門オブジェクトにたいして，資源消費コンテナクラスは，変動的資源用と固定的資源用の2種類を用意しておく必要がある．

製品，部品を表す原価計算対象クラスのオブジェクトの場合には，それが集約している資源消費オブジェクトは，具体的な物的な資源への参照をもっている．しかし，部門を表す原価計算対象クラスのオブジェクトの場合には，それが集約する資源消費オブジェクトが参照している資源は，物的な資源ではなく費目として表現されたヴァーチャルな資源である可能性がある．たとえば，減価償却費といった資源である．このような資源は，月間の資源の単価が月間の減価償却費そのものであり，それを1単位消費すると考えるのである．直接費については，基本的に消費量×単価でとらえるが，製造間接費については，消費量×単価で把握することが必ずしも現実的でない．そこで消費量＝1，単価＝発生総額そのものとして形式的に，消費量×単価の形をとらせることになる．

7　資源インターフェイス

資源インターフェイスは，外部から購入した材料のような資源と，自製部品のような原価計算対象を，消費される対象という観点から同一視するためのメカニズムである（図2参照）．資源消費オブジェクトは，消費対象への参照をフィールドに保持する．そのとき消費対象フィールドに保持されている参照の型は，資源型というふうに認識されるのみで，どういう資源なのか，すなわち外部から調達した材料なのか，自製部品なのかといったことは意識しないのである．たとえば，資源消費オブジェクトは，資源消費額を計算するために，calcCost()メソッ

図2　原価計算対象と資源の同一視



ドをもつが，資源消費オブジェクトのcalcCost()メソッドが呼ばれると参照している資源型オブジェクトにたいしてcalcCost()メソッドを呼び，その返し値に消費量をかけたものを返し値として返すことになる．そのとき，資源消費オブジェクトにとって重要なのは，資源インターフェイスを実装するオブジェクトであれば，かならずcalcCost()メソッドをもっており，その返し値をその資源の原価として受けいれるということである．実際には，資源消費オブジェクトが指している資源は，外部から購入した材料であるかもしれないし，自製部品かもしれないのである．すなわち，資源型のオブジェクト（資源インターフェイスを実装するクラスのオブジェクト）の実際の型は，オリジナル資源クラスを継承する外部購入材料クラスのオブジェクトであるかもしれないし，原価計算対象クラスを継承する部品クラスのオブジェクトであるかもしれないのである．

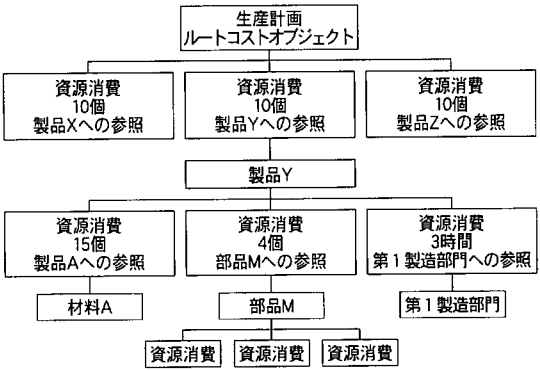
外部購入材料クラスのオブジェクトがcalcCost()メソッドを呼ばれた場合と，部品クラスのオブジェクトがcalcCost()メソッドを呼ばれた場合とでは，実際の処理は異なる．外部購入材料クラスのオブジェクトのcalcCost()メソッドは，単に，その価格フィールドにセットされている値を返すだけであろうが，部品クラスのオブジェクトのcalcCost()メソッドが呼ばれた場合には，部品クラスのオブジェクトに備わっている資源消費コンテナクラスのcalcCost()メソッドを呼んで，その返し値を返すことになる．このようにメソッドを呼ぶほうは意識せず，呼ばれたほうが適切な方法で処理するしくみが，ポリモルフィズムであり，もっともオブジェクト指向的な考え方が示されている部分である．

8　オブジェクト指向原価計算の再帰的構造

オブジェクト指向原価計算では，それぞれのクラスにおいて定義されているメソッドは，どれもそんなに複雑なものでない．非常にシンプルな仕事をしているといえる．ところが，オブジェクト指向原価計算では，資源消費オブジェクトを媒介にして，資源型のオブジェクトと原価計算対象クラスのオブジェクトが，連鎖的につながっており，究極的には製品オブジェクトへと集約されていく．製品オブジェクトのメソッドを呼び出すと，部品クラスのオブジェクト，その部品を構成する下位部品のオブジェクトと，原価計算対象を下降しながら次々と再帰的に，メソッドが呼び出され，結果的にかなり複雑な処理を行うことになる．通常の原価計算が，資源の消費を原価計算対象に関連づけるときに金額にして，中間的原価計算対象に集計しつつ，原価計算階層を上昇していくのにたいし，オブジェクト指向原価計算の場合は，再帰的にメソッドを呼び出しながら原価計算階層を下降していくことになる．

今，製品原価を計算する場合のことを考えてみよう．まず，原価計算対象クラスを継承する製品クラスのオブジェクトを1つ選択して，それに備わったcalcCost()メソッドを呼ぶとする．そうすると，製品クラスのオブジェクトが参照を保持する資源消費コンテナクラスのオブジェクトのなかに登録された1つひとつの資源消費オブジェクトにたいし，calcCost()メソッドが呼ばれて，その返し値を順番に合計して，その合計値を返し値として返すことになる．その資源消費オブジェクトの指している資源が，外部購入の資源ならば，設定された単価を返すが，もしその資源が，部品のような原価計算対象ならば，その資源消費コンテナに登録された1つひとつの資源消費オブジェクトにたいし，さらにcalcCost()メソッドを呼んでいくことになる．このようにして，末端の資源に到達するまで下降していくことになる．なお，返し値は，末端から次々と上の階層におくられ，最終的に，製品オブジェクトのcalcCost()メソッドの返し値として総合される．

図3　原価計算対象の階層



9　原価計算対象の階層構造とプロセスの階層構造

いままで，伝統的な原価計算対象の階層を念頭において議論してきた．製品，部品といった原価計算対象階層は，典型的に，図3のような関係である．ここで注目すべきは，原価計算対象階層のルートである．もしいま計画段階での原価計算の利用を考えているのであれば，原価計算対象階層のルートは，生産計画である．生産計画を表す原価計算対象オブジェクトに，各製品オブジェクトへの参照と各製品ごとの生産量を消費量にセットした資源消費オブジェクトが集約されることになる．原価計算対象階層のルートからcalcTotalCost()メソッドを呼べば，変動費と固定費を別々に計算して費用総額を計算することもできる．

実は，原価計算対象の階層は，1通りではない．とくに製造間接費の部分はさまざまな原価計算対象階層の定義が可能である．全部原価計算用の原価計算対象階層を定義すると同時に直接原価計算用の原価計算対象階層を定義することができる．また，活動基準原価計算を行うために活動の階層を定義することも可能である．1つの原価計算システムのなかで複数の代替的原価計算階層を定義するこ

図4　責任階層を中心としたプロセス階層

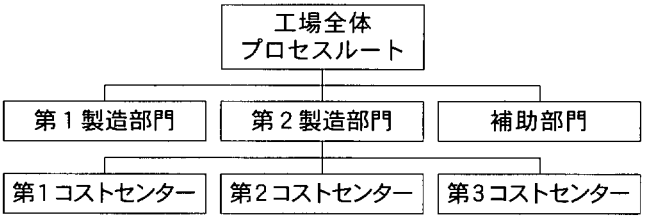
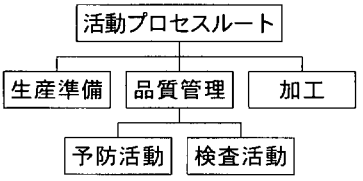


図5　機能を中心としたプロセス階層



とも可能である．また，将来的な原価計算としては，ソフトウェア部品の製作，ノウハウの蓄積など，企業内に残留するアウトプットを認識して，そこに原価を集計するタイプの原価計算を行うために企業内に残存するアウトプットを原価計算対象階層に組み入れることもできる．

いままでは，原価計算対象の階層だけに注目して議論を進めてきたが，原価計算対象とは別にプロセスの階層があることを忘れてはならない．資源消費クラスのオブジェクトは，少なくとも１つの原価計算対象クラスのオブジェクトに関連付けられるとともに，少なくとも１つのプロセスクラスのオブジェクトに関連づけられるのである．

原価計算対象が階層構造をもっていたように，プロセスもまた階層構造をもっている．しかし，原価計算対象の場合の階層構造とプロセスの階層構造とでは意

味が異なることに注意をする必要がある．原価計算対象の場合の階層構造は，消費するものと消費されるものという関係の連鎖による階層構造であったが，プロセス階層の場合には，そのような消費するものと消費されるものという関係の連鎖は存在しない．むしろ，プロセスの小分類，中分類，大分類といった，プロセスの要約過程を表す階層構造であるといえよう．プロセス階層についても，複数の代替的な階層構造を定義することができる．

図4は，責任階層を中心としたプロセス階層であり，図5は機能を中心としたプロセス階層である．

10　原価計算対象階層とプロセス階層の関係

原価計算対象の階層とプロセス階層を明確に区別することは非常に重要である．原価計算階層としての部門に帰属させることには意味がなくても，プロセス階層としての部門に帰属させることには意味があるということはよくある．たとえば，直接材料の消費は，原価計算対象の階層ということでは，コストセンターやその上位の部門に帰属させることは意味をもたない．直接材料は，製品，部品，といった原価計算対象に直接的に帰属させることができるのであり，あえて部門クラスのオブジェクトに帰属させる必要はないのである．しかし，部門ごとコストセンタごとの直接材料の消費能率を知るためには，直接材料の消費を部門やコストセンタに帰属させることが必要なのである．

また，活動基準原価計算の場合，しばしば，プロセス階層と原価計算対象が明確に区別されないために混乱が生じていることがある．ある活動が，原価計算対象として定義されているのか，それともプロセスとし定義されているのかを明確にすることはきわめて重要である．活動基準原価管理を行うためには，かならずしも活動を原価計算対象として定義しなくてもよいのである．ただ，活動基準原価計算として，製造間接費の配賦を改善しようとするならば原価計算対象として活動を定義する必要がある．

原価計算対象の階層とプロセスの階層は直交した関係にあるといえる．一般的な原価計算システムでは，製品原価をみて，そのプロセスの情報を引き出すこと

はできない．しかしながらオブジェクト指向原価計算では，製品オブジェクトにたいして，プロセスの情報を提供させるメソッドを呼ぶことが可能である．たとえば，ある製品の原価のうち，特定のプロセスに帰属する原価はどれだけかを%で示す．ある製品の生産に関与する特定のプロセスで行われた資源消費はどのようなものかを，資源消費コンテナに入れて返すといったメソッドを作りこんでおくことが可能である．これを可能にするためには，ある特定のプロセスに属する資源消費オブジェクトに対し，マーキングしていく機能が必要となる．すなわち，資源消費クラスには，マークフィールドが設けられ，ある特定のプロセスまたはプロセスグループの mark()メソッドを呼んで，マーキングをしたうえで，原価計算対象階層の特定の製品を選び，selectMarkedResourceUsage()というようなメソッドを呼ぶような形になる．

11　原価計算対象クラスに作りこむ機能の例

原価計算対象クラスには，原価計算対象の内訳，構成情報を取得するための豊富なメソッドが定義されるべきである．資源の種類を指定して，特定の資源の種類に属する資源消費オブジェクトを資源消費オブジェクトコンテナにいれて返すメソッドや，オリジナル資源をリストアップする機能などである．そのように，製品オブジェクトから呼び出すことのできるメソッドは，金額情報だけでなく物量情報をも返すことができるのである．

なお，原価計算対象クラスには，その資源消費コンテナクラスに資源消費オブジェクトを追加するための add()メソッドが備わっているのは当然のことである．

12　資源消費のデータから原価計算対象ツリーとプロセスツリーを構築する

ひとたび原価計算対象階層，プロセス階層が構築されてしまえば，その階層構造を使って，上位の原価計算対象に特定のメソッドを発行することでさまざまな問い合わせが可能になる．しかしながら，そのような階層構造を構築すること自体がかなり複雑な作業である．Main のなかで，１つひとつ資源消費オブジェクト，原価計算対象オブジェクトをインスタンス化して，関係する原価計算対象オ

ブジェクトに資源消費オブジェクトをaddしていくのは，大変なことである．このあたりは，一定の書式でかいた資源消費データ，あるいはデータベースに登録されたレコードをもとに自動的に，インスタンス化および，原価計算対象オブジェクトへの登録，プロセスオブジェクトへの登録を行えるような仕組みが必要であろう．

そのさい資源消費オブジェクトを生成するもととなる資源消費レコードには，かなり多くの情報をあわせてもたせておき，その情報に基づいて事後的に，必要に応じて新たな原価計算対象階層，プロセス対象階層を構築できるようにしておくことが有用である．

13　実績計算と計画計算

オブジェクト指向原価計算は，実績計算としても計画計算としても実行することができる．

オブジェクト指向原価計算では，資源消費と原価計算対象の関連，資源消費とプロセスの関連といった，関連レベルと，資源消費にセットされる消費量レベルと，オリジナル資源にセットされる価格レベルと，3つのレベルをそれぞれ別々に変化させることができる．そのため，関連レベル，消費量レベルを変化させずに，価格レベルだけを変化させてシミュレートさせるというようなこともできる．一般に一番変動が激しいのが，価格レベルであり，次に消費量レベルが変動し，一番変化が遅いのが関連レベルであろう．このように原価を構成する変動の激しい部分と，変動の激しくない部分を切り離すことができるというのはオブジェクト指向原価計算の非常に有利な側面である．

よく，標準原価計算が役に立たないといわれれることがある．その場合，役にたたない理由は，価格が急激に変動することであったり，新製品立ち上げ時に，消費能率が安定しなかったりといった事情である．価格の急激な変動にたいしては，あらかじめ構築された原価計算対象階層をそのままつかって，外部購入資源オブジェクトにセットされている価格をカレントな価格に変えてやれば解決する．外部購入資源オブジェクトにあらかじめ複数の価格をセットしておき，目的に応

じて使用する価格をいっせいに切り替えるようにしてもよい.

標準消費量の設定がむずかしいような場合，実績データを収集し，それをベースに，価格水準を切り替えるなどしてシミュレーションを行うことも可能である. なぜこのようなことが可能かといえば，価格データは，資源消費オブジェクトが持つことなく，資源消費オブジェクトが指ししめすオリジナル資源オブジェクトに保持されているからである.

14　オブジェクト指向原価計算と GUI

オブジェクト指向原価計算の第一の目的が原価データの直感的操作であった. であるから，使い勝手のよい GUI と結合しなければ意味がない. オブジェクト指向的に構築されたモデルは，GUI 表現とは非常に相性がよい. オブジェクトを指定して，そのオブジェクトに備わったメソッドを呼ぶという行為は，GUI 上でいえば，オブジェクトを表現するアイコンを指定して，たとえば右クリックで実行すべき機能を選択するといった動作とパラレルである. 前項まで論じたものが MVC の Model とすれば View と Controller の部分を作る必要がある. GUI の部分の開発は，Java の Swing フレームワークを利用すれば，比較的容易に行うことができる.

この論文で示した原価計算モデルを GUI で表現することにより，経営者が直感的に原価モデルを操作できるようになるし，あるいは設計者が直感的に GUI 上のモデルを操作して，原価の作りこみに役立てたりすることができよう. 原価計算対象ナビゲータやプロセスナビゲータを使って，原価の構成要素の任意の部分に焦点をあて，そこを起点に，構成情報をうることができる.

15　残された課題

本論文でとりあげたオブジェクト指向原価計算は，まだ開発途上のものであり，まだ完成したものではない.

オブジェクト指向原価計算は，実績計算にも対応すると述べた. しかしながら，現在のところ，材料単価の払出単価にたいする先入先出法や後入先出法の採用，

月初仕掛品原価の存在や月初中間部品の存在など前期の消費能率と当期の消費能率が混在するような場合には，うまく対応できない．そもそも，先入先出法や後入先出法の採用，月初仕掛品原価の存在や月初中間部品の存在など前期の消費能率と当期の消費能率の混在は，原価の透明性を阻害する要因であり，それらを考慮することはオブジェクト指向原価計算の理念に反するようにも思われる．しかしながら原価計算制度に対応させるためには，上記の点はなんらかの形で解決されなければならいないと思うが，これは今後の課題である．

＊　UML は，Object Management Group の登録商標である．Java, Swing は米国および，その他の国における米国 Sun Microsystems, Inc. の商標である．

＊　この論文で展開したオブジェクト指向原価計算システムのプロトタイプを作成する実験を尾畑研究室で行っており，以下の URL で公開している．
http://obata.misc.hit-u.ac.jp/oocm/

（一橋大学大学院商学研究科教授）