パターン：部分的オブジェクト＋プロトコル　(Half－Object＋Protocol)

―――――――――――――――――――――――――――――――――――――――

著者：Gerard　Meszaros
所属：BNR, Canada
PO　Box　3511, Station　C,
Ottawa, Ontario Canada K1Y 4H7
E-mail：gerard@bnr.ca
Phone：(613)763-7860
Fax：(613)765-4855

現連絡場所：gerald.meszaros@acm.org , 87 Connaught Dr. NW, Calgary
Alberta, Canada T2K 1V9,403-264-5840,FAX 403-233-2021

―――――――――――――――――――――――――――――――――――――――

適用可能性：

システムの中のオブジェクトとそれらの関連を決定したら、どのようにして、それらのオブジェクトが実行されるアドレス空間に割り当てるか決定すべき時です。このパターンは、オブジェクトが配置される場所と、アドレス空間の境界に対してのオブジェクトの位置を決定するのに役立ちます。

問題：

オブジェクトは、複数の計算処理の場面（例えば、アドレス空間のような）に現れなければならない場合があります。どうすれば、1 つのアドレス空間と複数のアドレス空間（例えば、単体処理 vs 分散処理のような）の違いを意識しなくてもよくなりますか。

制約：

複雑さ、分散、情報の有用性、コスト、性能

多くのコンピュータシステムでは、コスト、大きさ、物理的な分散、プログラミング環境の違い、規定などから、複数のアドレス空間をまたがって実装しなければなりません。こういったシステムを、1 つのアドレス空間だけに存在するオブジェクトとして、簡単に分解できる場合もあります。いくつかのオブジェクトは、ハードウェア（センサ、ディスク

ドライバなど）と組み合わされているため、ある特定のアドレス空間にしか配置できません。その他のオブジェクトは、ハードウェアオブジェクトとの結びつきに基づいて、簡単に配置できます。しかし、ある概念が両方の空間に存在するため、簡単に分解できない場合もあります。また、オブジェクトがその振る舞いを実行するためには、複数のアドレス空間から情報を要求する必要があるので、複数のアドレス空間のオブジェクトと相互作用しなければならない場合もあります。同様に、他のアドレス空間のオブジェクトから届いた要求を（非同期に）処理する必要があります。

この場合、どのアドレス空間も単独では、仕事を実行させることはできません。そこで、（少なくとも）１つのオブジェクトは、複数のアドレス空間になければなりません。問題になっているオブジェクトは、１つのアドレス空間につき、少なくとも 1 つ以上のオブジェクトに分割しなければなりません。アドレス空間をまたがったオブジェクトを分割すると、いっそう複雑になります。それは、１つのオブジェクトとして扱うほうが、あるプロトコルで連携し合う 2 つの部分的オブジェクトとして扱うよりも簡単だからです。ある部分的オブジェクトが、もう一方の部分的オブジェクトから情報を取ってくる、または、作業をしてもらう必要があるときは、それらを相互作用させれば、その部分的オブジェクトに全ての振る舞いを実現できます。

これらの相互作用が頻繁に起こると、メッセージオブジェクトの構築や他のアドレス空間のオブジェクトにメッセージオブジェクトを送るためにかかるコストは、容認できなくなります。オブジェクトを 同一な２つに分割すると、それぞれの部分が、もう片方のアドレス空間とは独立に、多くの局所的な要求に応じることができます。しかし、その結果として、重複した機能性と 2 つのオブジェクト間の同期を必要とします。同一な２つに分割することで、部分的オブジェクト間で使用されるプロトコルは、オブジェクトのインタフェースから見えなくなります。

そこで：

オブジェクトを、プロトコルで連携し合う 2 つの相互依存の部分的オブジェクト（各アドレス空間で唯一とする）に分割しましょう。各アドレス空間で、もう一方の部分的オブジェクトと効果的に相互作用するために必要な機能はすべて、実装しましょう。（これによって、例えば、両方の空間で実装されるような、重複した機能が生じるかもしれません。）2 つの部分的オブジェクトの間で、プロトコルを定義しましょう。それによって、両方のオブジェクトの活動が上手く連携でき、アドレス空間を越える必要のある不可欠な情報を運べるようにしましょう。

関連パターン：

それぞれの部分的オブジェクトの機能が決まれば、そのプロトコルを定義できます。そこで、プロトコルの設計についてのパターン（オブジェクトとしてのメッセージ（Message as Object)、オブジェクトとしてのメッセージパラメータ（Message Parameter as Object)）と、さらに、メッセージの作成（情報の収集、形式の設定）と受取（解析、処理）を扱うための仕組みを設計するためのパターンの使用が考えられます。

例：

- 物理的な分散：
  他のマシンで動作させるデバッガ、全国的な電話網に分散している電話サービス。ハードウェア装置にアクセスするセンサのようなシステム。情報源（中心データベース）からかけ離れたユーザインタフェース。分散（複製された）データベース。
- 大きさ：
  機能の実時間コストのために、それをスレーブマイクロプロセッサで実行しなけばならないシステム
- プログラミング環境の違い：
  Smalltalk で実装された C++デバッガ用のユーザインタフェース
- 規定：
  電話網のサービス制御地点には、電話会社が作成した、電話網の交換機を通してやりとりされる通話量を緩和させるためのソフトウェアが入っています。このソフトウェアは、交換機で動作させることを許可されないかもしれません。（また、サービス制御地点は、通常、UNIX に基づいた環境であるのに対して、交換機には交換機特有の環境が存在するので、プログラミング環境の違いの例とも言えます。）