役割・能力・規則
-抽象オブジェクトによるソフトウェア構成の検討-

鈴木　優輔　重見　大輔　柴田　直樹　坂下　善彦
湘南工科大学　情報工学科

1.はじめに

ソフトウェアを開発する手法として、手続き型プログラミングやオブジェクト指向などの方法論が存在するが、現在の主流としてオブジェクト指向技術が用いられている。オブジェクト指向を用いたソフトウェアの開発において、何を「物」としてオブジェクト化するかという考え方はそのソフトウェアを開発するプログラマに任されている。

我々は、オブジェクト指向を用いたソフトウェア開発において個々のソフトウェアのオブジェクトには、役割・能力・規則に相当する要素が何らかの形態で存在すると仮定し、「物」としてのオブジェクトの役割・能力・規則の要素の部分に注目し、そのオブジェクトから要素を取り出して抽象オブジェクトとして構成するソフトウェア開発の手法を研究している。

このために、個々に対象とするアプリケーションのそれぞれの要素をオブジェクトから取り出して抽象オブジェクトとして定義し、具体的な適応に向けてアプリケーションを製作する検討を進めた[1]。

この手法で定義した構成に基づき、対象とするアプリケーションを製作して実際に機能するかを検証する。

2.我々の考える「役割・能力・規則」

(1)役割

役割は「役をそれぞれに割り当てること。割り当てられた役目」とされている。ここでの役目とはアプリケーションの持つ仕事そのものを意味する。

例えば、検索や計算といった作業もアプリケーションに与えられた仕事である。これらの仕事はアプリケーションの作成目的によって千差万別である。電卓は計算をするという仕事は出来るが、検索という仕事が出来ないのもこれに当てはまる。

(2)能力

ソフトウェアを構成するオブジェクトの持つ能力とは、そのソフトウェアの中でオブジェクトが何かを「することができる」ということに着目し、一般的に「能力」を「することができる」ことであると定義する。

この定義を元に、本研究の場合では実体化されたオブジェクトが「上下に動くことができる」、「左右に動くことができる」、「上下左右に動くことができる」という 3 つの能力を持つソフトウェアの製作を想定した。この場合のソフトウェアに対するオブジェクトの能力とは「動くことができる」ということであると定義する。

(3)規則

「規則」は、手順や行為などを行う際のきまりで一種の「制約」ともいえる。例を挙げると、「サッカーでは、手はキーパー以外使ってはいけない」、「ラクビーで、パスするとき自分の前にパスを出してはいけない」、「バレーボールでは、3 タッチ以内で相手側にボールを返す」など、目的を実現する上で、制約事項である。一般的に「規則」は、オブジェクトを制約するものと定義する。規則で、手順を命令することで役割がそれぞれ動作し、制約を付けることによって能力が動作する。そうすると、アプリケーションを作成するときに、規則が、全体がどのように関連しているかを検討した[2]。

3.ソフトウェアの構成

(1) 役割

Role という基底クラスに仕事クラス、動作クラスという二つの派生クラスをもつ。それぞれに複数のメンバ関数・メンバ変数を持っている。仕事クラスは 2・(1)の通りアプリケーションに与えられた仕事を意味し、動作クラスによって Role クラスの制御やデータの送受信を行う。二つの派生クラスは基底クラスを継承し、アプリケーション目的を達成するためにサポート体制

Role, Ability, and Rule - The software composition by the abstract object-
Yusuke Suzuki, Daisuke Shigemi, Naoki Shibata, Yoshihiko Sakashita
Information Science, Shonan Institute of Technolgy

を取っている[3]。

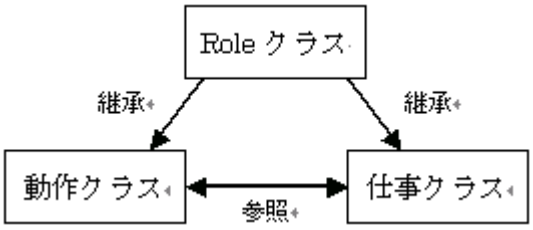

図 1.「役割」におけるクラスの関係

(2)能力

実体化させるオブジェクトを Object1 クラスと定義する。この Object1 クラスから「動くことができる」能力を取り出して抽象オブジェクトとし、Ability クラスとして定義した。Object1 クラスは Ability クラスから「動くことができる」能力を得る。また、一種類のクラス型オブジェクトだけがこの能力を得るのではなく、条件の異なる Object2 クラス、Object3 クラスを定義する。Object2 クラス、Object3 クラスも Ability クラスから能力を得る。この 4 つのクラスの関係を図 2 に示す。図 2 におけるクラス間の矩形は、Object1 クラス、Object2 クラス、Object3 クラスが Ability クラスから能力を得ることを示す[4]。

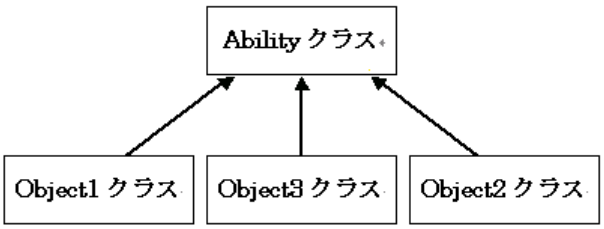

図 2.「能力」におけるクラスの関係

(3)規則

Rule という 2 つの派生クラスを持っている抽象クラスを作成。まったく別として、Agent1 クラス・Agent2 クラスがある。規則が制約することによって、制約に基づいて、Agent1 クラス・Agent2 クラスが動作する。Agent1 クラス・Agent2 クラスは、動作を参照しながら、規則に従って動作する。Agent1 クラス・Agent2 クラスは、お互いに動作を参照する。3 つの関係を図 3 に示す。

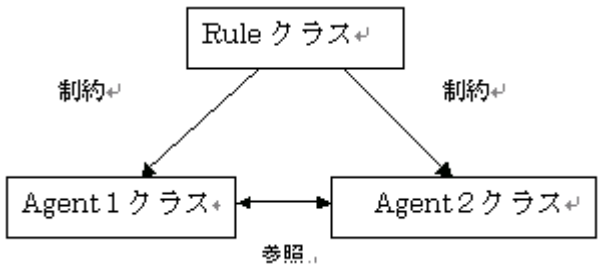

図 3.「規則」におけるクラス

4.評価

これまでに定義した概念に基づき、検証のためのアプリケーションを作成した。

(1)役割

鉄道信号を使い、信号の現示によって列車を制御するという役割で全体がどのように関連するかを検討した。

青信号の時に列車がさしかかると、列車はそのまま進行するが、赤信号の場合に列車が差し掛かると列車はその場で停止する。これは、動作クラスの中に flag というメンバ関数があり、青信号の場合は flag0 で列車には何も及ぼさないが、赤信号の場合は flag1 に設定してあり、flag1 の場合には列車はその場で停止するようになっている。

(2)能力

上記の定義した概念を基に、検証のために特定のキーボード入力により実体化されたオブジェクト(bitmap 画像)が、ウィンドウ上を移動するソフトウェアを製作した。「上下に動く」、「左右に動く」、「上下左右に動く」という 3 つの能力をそれぞれ変更することによって、オブジェクトの振舞いの違いによる動きを観測した。

(3)規則

規則に基づいてアプリケーションを作成。この時、規則 1「ペナルティーエリアに入ってはいけない」、規則 2「3 タッチ以内でパスを出す」。Agent をそれぞれ人・ボールに設定した。規則 1・規則 2 を変更によって、人・ボールが違うふうに動作を確認した。

5.終わりに

ソフトウェアには、役割・能力・規則に相応する要素部分が何らかの形態で存在すると仮定して、これらの要素の互いの関係を論ずる前に、個々の要素を単独で抽出する試みを実施した。極めて小規模なソフトウェアでの検討を行ったが、より大きい規模での検証と、併せてこれらの要素が共存する構成の場合の検討を進める。

参考文献

[1]坂下,井手口,佐藤,水野:役割に基づくアプリケーションの振舞い制御,電子情報通信学会論文誌,Vol.J82-B,No.5,1996.6

[2]Tucker:C++による実践的ソフトウェア構築入門 1998.5.29

[3]矢沢久雄著:C++クラスと継承完全制覇 2002.10.6

[4]杉浦賢著:最新 C++ハンドブック 2002.7.10