オブジェクト指向言語Smalltalkによる仕様記述

4S-3

浜野博,南摩英明,有沢誠
山梨大学

1 はじめに

オブジェクト指向言語ｓｍａｌｌｔａｌｋで2つの例題の仕様を記述し、Ｃ++言語での記述と比較してｓｍａｌｌｔａｌｋによる仕様記述の可能性について考える。

2 交通信号制御システム

動的な例題として交通信号の問題(文献[3])をとりあげる。まず、何をオブジェクトとし、それらオブジェクト間の関係をどうするかを考える。より自然なモデル化を重視して考えると、制御系、信号機、計測機の3つのオブジェクトに分けることができる。ここでは、それぞれのオブジェクトの関係を制御系を中心に考える。制御系の必要な機能(メソッド)は次の2つがある。

①車の混雑をなるべく減らす。
②双方向の車が2つの交差点をスムーズに通過できるようにする。

①の実現には、問題では明確には定義していない混雑の度合いの明確な定義が必要となる。ここでは、次のように定義した。一定時間内の通過した車の台数と通過した車の平均速度をそれぞれ目安と比較して混雑度(3段階)を定義する。たとえば平均速度が目安より大きく通過台数が多い場合は交通量は普通である。目安はシステムを記述した後、シミュレーションを行い決める。この混雑度により信号機の各色の継続時間を変えて(混雑している場合は青色の継続時間を長くする)、混雑をなるべく減らす。

②の実現には信号機のタイミングを考える。双方向(A→B,B→A方向)の車が2つの交差点(1,2)をスムーズに通過するには、双方向の混雑度が同じ場合2つの信号機(A,B)が同時に青になればいい。A→B方向の混雑度が大きい場合には信号機Aの変化を数秒早めるようにする。双方向の混雑度は①の方法で比較することも可能であるが、タイミングは長いサイクルで変えたいため、①で変えている信号機の青色の継続時間で混雑度を比較する。この比較でも目安を使うことにし、目安はシミュレーションを行い決める。

①,②の2つの機能(メソッド)をもつ制御システムの実現の為には、信号機,計測機には次のような機能が必要となる。

信号機
- o 各色の継続時間を変える
- o 信号機の変化の時間を早める
- o 青色の継続時間を返す

計測機
- o 1秒間に通過した車の速度を計測
- o 通過台数と速度の合計を返す

このような制御系,信号機,計測機をインスタンスとして発生させるクラス(すべて継承を使わずオブジェクトのサブクラスとしている)を記述したも

Smalltalk as Specifications Language
Hiroshi HAMANO , Eimei NANMA , Makoto ARISAWA
YAMANASHI University

のの例として制御系を次に示す。変数名などは日本語にしてある。

```
Object variableSubclass: #制御系
  instanceVariableNames:
    '平均速度目安 通過台数目安 同期目安 '
  classVariableNames: ''
  poolDictionaries: '' !

!制御系 class methods ! !

!制御系 methods !

混雑解消: 信号機1 and: 信号機2 use: 計測機
  | 台数 速度 |
  台数 := 計測機 get通過台数.
  速度 := 計測機 get速度合計.
  ((台数 // 速度) < 平均速度目安)
   ifTrue:  [信号機1 plus. 信号機2 minus]
   ifFalse: [
    (台数 > 通過台数目安)
     ifFalse: [信号機2 plus. 信号機1 minus]]!

目安初期化: 目安配列
  平均速度目安 := 目安配列 at: 1.
  通過台数目安 := 目安配列 at: 2.
  同期目安 := 目安配列 at: 3.!

同期: 信号機配列
  | 青A 青B |
  青A := (信号機配列 at: #A) get青.
  青B := (信号機配列 at: #B) get青.
  (青A > (青B + 同期目安))
   ifTrue: [(信号機配列 at: #A) 時間経過.
            (信号機配列 at: #C) 時間経過].
  (青B > (青A + 同期目安))
   ifTrue: [(信号機配列 at: #B) 時間経過.
            (信号機配列 at: #D) 時間経過].! !
```

3　酒屋の在庫管理問題

次に、静的な例題として酒屋の在庫管理の問題(文献[2])の仕様をｓｍａｌｌｔａｌｋで記述してみる。受付係の仕事を記述するのが目的なので、まず受付係の機能(メソッド)を考えて、関係するオブジェクトを挙げる。

受付係　：

①倉庫係から届いた積荷票の品物が在庫不足リストにあったら削除して出庫指示書を作成する。在庫不足リストにない場合は、在庫ファイルに追加する。

②小売から出庫の依頼を受けた時には、在庫ファイルにその品物がないなら、そのことを小売に連絡し在庫不足リストに追加する。

③在庫不足リストの登録がある程度になったら問屋へ注文する。

④出庫指示書がある程度まとまったら倉庫係へ出庫指示をする。

以上の４つが最低必要で、在庫ファイル、在庫不足リストは受付係の内部変数とする。

問屋　：受付係からの注文を受けると倉庫係へ商品を発送する。

倉庫係：コンテナが届いたら、積荷票を作成して受付係へ渡す。

小売　：受付係へ出庫を依頼する。

各オブジェクトの機能を実際に記述するには、在庫ファイル、在庫不足ファイルの詳細を決める必要がある。在庫ファイルは、コンテナ(積荷票)の集合として扱う。

4　おわりに

ｓｍａｌｌｔａｌｋはＣ＋＋と違いオブジェクト指向を徹底しているので、特に在庫管理の問題ではより自然なモデル化ができた。ｓｍａｌｌｔａｌｋによる仕様記述で重要となるのは、いかにうまくシステムの既存のクラスを利用した差分プログラミングを行い記述量を減らすかにあると思う。

5　参考文献

[1]Adele Goldberg,Devid Robson: Smalltalk-80 The Language and Its Implementation,1983

[2]山崎 利治:設計方法解説のための共通例題,情報処理学会誌,1984

[3]南摩 英明:Ｃ＋＋による仕様記述,情報処理学会全国大会誌,１９９０