BridgePoint アクション記述言語
抜粋版

（株）東陽テクニカ
ソフトウェア・ソリューション

1. インスタンスの選択

1.1. 関係を使用しないインスタンスの選択 (5.1.2)

◆あるオブジェクトの中からインスタンスを一つ任意に選択する
select any インスタンス名 from instances of オブジェクトのキー文字
（例）select any cat from instances of C ;
◆あるオブジェクトの全インスタンスを選択する
select many インスタンス名 from instances of オブジェクトのキー文字
（例）select many cats from instances of C ;

◆あるオブジェクトの中からある条件を満たすインスタンスを一つ任意に選択する
select any インスタンス名 from instances of オブジェクトのキー文字
where selected.属性名　条件
（例）select any cat from instances of C where selected.color == “black” ;
◆あるオブジェクトの中からある条件を満たすインスタンスを全て選択する
select many インスタンス名 from instances of オブジェクトのキー文字
where selected.属性名　条件
（例）select many cats from instances of C where selected.color == “black” ;

1.2. 関係をたどったインスタンスの選択 (6.1.4)

◆あるインスタンスに関係のあるオブジェクトのインスタンスを選択する
select [one | any | many] インスタンス名 related by
関係元 -> オブジェクトのキー文字[関係の番号] ;
（例）select one cat related by owner->C[R1] ;
select any dog related by owner->D[R2] ;
select many dogs related by owner->D[R2] ;

※関係は複数回たどることができる。
（例）select any dog_food related by owner->D[R2]->F[R3] ;

◆あるインスタンスに関係があり、かつある条件を満たすオブジェクトのインスタンスを選択する。

select [one | any | many] インスタンス名 related by
関係元 -> オブジェクトのキー文字[関係の番号]
where selected.属性　条件

（例）select any dog related by owner->D[R2] where selectd.color == “black” ;

2.　インスタンス操作

2.1.　インスタンスの属性へのアクセス (5.1.3 , 5.1.4)

インスタンス名.属性名

（例）　color = dog.color　（読み込み）
dog.age = 5 ;　（書き込み）

2.2.　インスタンスの生成

create object instance インスタンス名 of オブジェクトキー文字 ;

（例）create object instance new_dog of D ;

2.3.　インスタンスの削除

delete object instance インスタンス名 ;

（例）delete object instance old_dog ;

2.4.　関係を張る

relate 関係元のインスタンス名 to 関係先のインスタンス名 across 関係の番号 ;

（例）relate new_dog to owner across R2 ;

◆関連付けオブジェクトを使用して関係を張る

relate 関係元のインスタンス名 to 関係先のインスタンス名 across 関係の番号
using 関連づけオブジェクトのインスタンス名 ;

（例） create object instance new_enrollment of E;
relate new_dog to class across R4 using new_enrollment ;

2.5. 関係の消去

unrelate 関係元のインスタンス名 from 関係先のインスタンス名 across 関係の番号；

（例）unrelate dog from owner across R2；

◆関連付けオブジェクトを使用した関係を消去

unrelate 関係元のインスタンス名 from 関係先のインスタンス名 across 関係の番号
using 関連づけオブジェクトのインスタンス名；

（例） unrelate dog from class across R4 using enrollment；

3. 制御文

3.1. IF 文 (4.1.1)

if(条件) end if；(elif と else は必要時のみ)

（例）

```
if ( dog.color == “black” )
 x = 1；
elif ( dog.color == “white” )
  x = 2；
else
  x = 3；
end if；
```

3.2. For Each Loop (4.1.2)

for each インスタンス名 in インスタンス集合
＜実行文＞
end for；

（例）select many dogs from instances of D；

```
  for each dog in dogs
     dog.feed_time = 9；
  end for；
```

3.3. While Loop (4.1.3)

条件が TRUE の間中、ループし続ける

while (条件式)
< 実行文 >
end while ;

（例）j = 1 ;
while (j <= 20)
d.ID = j ;
j = j + 1 ;
end while ;

3.4. その他の制御文

◆Break (4.1.4)
for each ループ、while ループから抜けるときに使用する。

◆Continue (4.1.5)
for each ループ、while ループで、残り行を実行せずにループの初めに戻る時に使用する。

◆ネスト（4.1.7）
IF 文、for each ループ、while ループはネストすることができる。

4. イベント

4.1. イベントの生成 (7.1.2)

generate イベントラベル to <ターゲット> ;
generate イベントラベル:`イベント名` to <ターゲット> ;
generate イベントラベル(補足データ) to <ターゲット> ;
generate イベントラベル:`イベント名`（補足データ） to <ターゲット> ;

<ターゲット>
存在しているインスタンスに対するイベント：*送り先インスタンス名*
生成イベントの場合： *オブジェクトのキー文字 creator*
割当子に対するイベント： *オブジェクトのキー文字 assigner*

（例）generate D1:'go to bed' to dog ;

4.2. 受け取ったイベントの補足データへのアクセス (7.1.1)

rcvd_evt.補足データ名

（例） dog.age = rcvd_evt.age ;

5. 式

5.1. 論理式(8.1.4)

= =　　!=　　<　　>　　<=　　>=

and　　or　　not

5.2. 文字列操作(8.1.6)

（例） a = “Hello, world “ ;

a = “Shlaer” + “Mellor” ;

a = dog.name + “;“ + owner.name ;

※3020 では、3 項の文字列加算はできません。

5.3. 集合操作 (8.1.9)

empty < handle >

not_empty <handle>

cardinality<handle>

5.4. Enumeration の取得

Enumeration 名::Enumrator

（例）a = enum::a1 ;

6. Operation/Function/Bridge

Operation:クラスベースのメッソド

インスタンスベースのメッソド

Function:ドメイン内でのグローバル関数。

Bridge:他のドメインの処理を呼び出す。

6.1. Operation の呼び出し

ClassOperation:クラスベースのメソッド

◆戻り値のない ClassOperation の呼び出し

クラスキー文字 :: 関数名（パラメータ名：値）；

（例） B::pressed() ;

◆戻り値のある ClassOperation の呼び出し

<value> = クラスキー文字 :: 関数名（パラメータ名：値）；

（例） return_value = B::pressed()；

InstanceOperation:インスタンスベースのメソッド

◆戻り値のない InstanceOperation の呼び出し

インスタンス名.関数名（パラメータ名：値）；

（例） select any button from instances of B;
button.pressed();

◆戻り値のある InstanceOperation の呼び出し

<value> = インスタンス名.関数名（パラメータ名：値）；

（例） select any button from instances of B;
return_value = button.pressed();

6.2. Function の呼び出し

◆戻り値のない Function の呼び出し

:: 関数名（パラメータ名：値）；

（例） :: function();

◆戻り値のある Function の呼び出し

<value> = :: 関数名（パラメータ名：値）；

（例） return_valude = :: function();

6.3. Bridge の呼び出し (9.1.2)

bridge ExternalEntity のキー文字 :: 関数名（パラメータ名：値）；

“bridge”は書かなくても可

6.4. Operation/Function/Bridge の記述

Operation/Function/Bridge の記述の中にアクション言語の文法で処理を記述することができる。

◆return 文が存在する

（例）select any dog from instances of DOG；
return dog.weight；

◆パラメータが存在する

param.パラメータ名

（例）return param.x * param.x；

7. タイマ (11.1)

（全て、“bridge”の記述はなくても可）

7.1. タイマを開始する

bridge <timer_handle> = TIM::timer_start(microseconds:時間,event_inst:イベントインスタンス名)

◎タイマ使用時は、タイムアップした時に発行されるイベントをあらかじめ定義し、そのイベントを timer_start で指定する。

（例）create event instance ev of V2:Timeout to self ;
self.Timer_ID = TIM::timer_start
(microseconds:self.cooking_period ,event_inst: ev) ;

◆ 一定時間毎にタイムアップするタイマの開始

bridge <timer_handle> =
TIM::timer_start_recurring
(microseconds:時間,event_inst:イベントインスタンス名)

7.2. その他のタイマの処理

◆タイマの残り時間を得る

bridge 戻り値 =
TIM::timer_remaining_time (timer_inst_ref:<timer_handle>)

◆タイマをリセットする

bridge 戻り値 =
TIM::timer_reset_time (timer_inst_ref:<timer_handle>, microseconds:時間)

戻り値： TRUE - タイマが存在する場合
FALSE - タイマが存在しない場合

◆タイマのキャンセル

（キャンセルとタイマインスタンスの削除）

bridge 戻り値 =
TIM::timer_cancel (timer_inst_ref:<timer_handle>)

戻り値： TRUE - タイマが存在する場合
FALSE - タイマが存在しない場合

◎timer_start_recurring 以外のタイマは発火すると自動的に削除される。