Action Data Flow Diagrams

Leon Starr
November 1, 2021
mint.sm-meta.action.tn.1
Version 0.1.1

Copyright © 2021 by Leon Starr / MIT Open Source
MODEL INTEGRATION, LLC

Change Log

Version	Date	Changes	Modified by
0.1.0	Nov 1, 2021	Copy notes in from tablet made in August/Oct 2021	Leon Starr
			Leon Starr

There are four places in Shlaer-Mellor where an Activity may be defined.

Each Activity type specifies a set of zero or more input parameter:type pairs

State Activity

OPEN

UI.Door opened(Shaft)
my Bank .= /R4/R2/Bank
Time to close -> me @ Blocked ?
    my Bank.Block clear time : my Bank.Passenger load time

Only synchronous Activities may specify a set of zero or more output parameter:type pairs

State Activities and Asynchronous External Entity Operations may not specify outputs

Class Method

Bank Level

Bank {I, R29}
Floor {I, R29}

Choose shaft(calling floor : Level Name,
service dir: Direction) : Shaft ID

=>> /R29/R1/Shaft/R2/Cabin(1, ^-Estimate delay(calling floor: Floor, in.service dir)).Shaft

Arrived at floor(cabin : Shaft ID)

Domain Operation

Arrived at floor -> Cabin(Shaft : in.cabin)

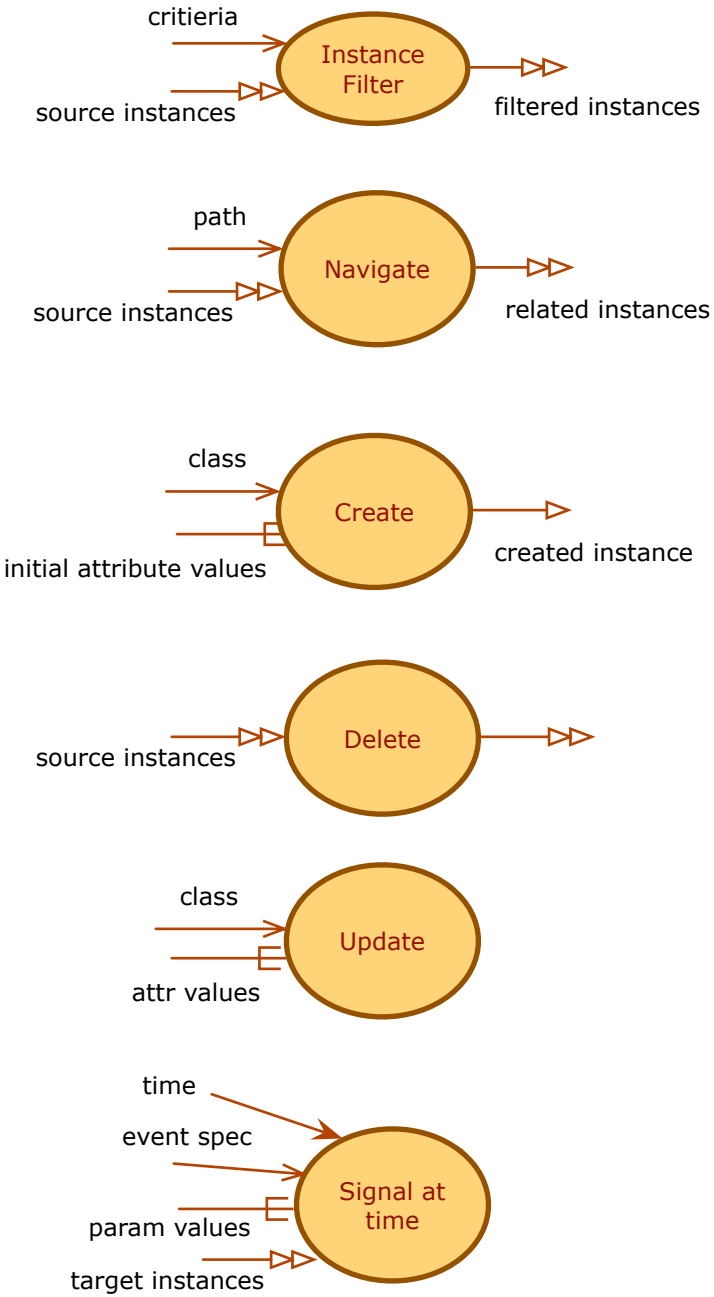
External Entity Operation

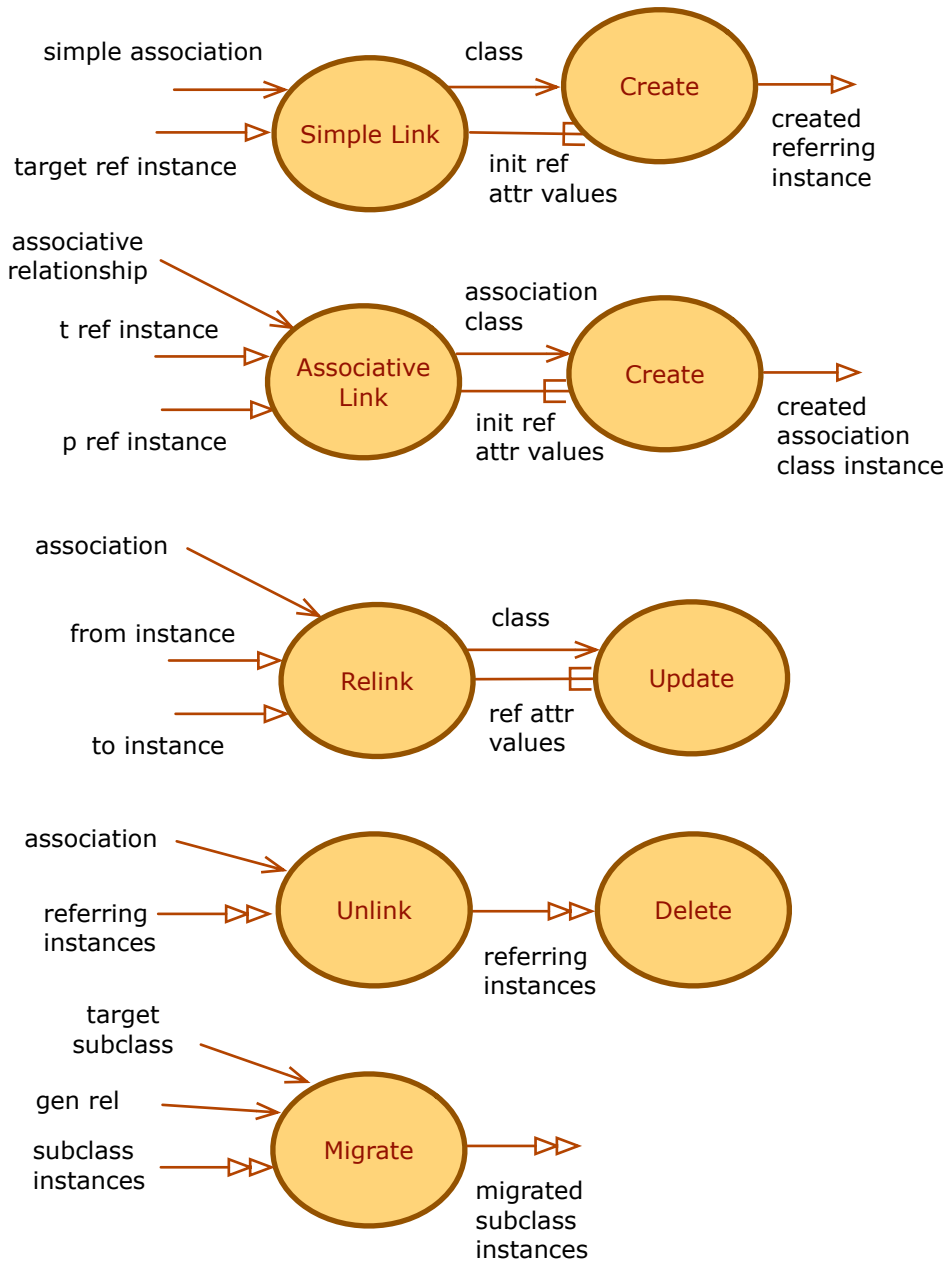
(asychronous or synchronous)

UI.Cabin arrived(shaft: Shaft ID, direction: Direction)

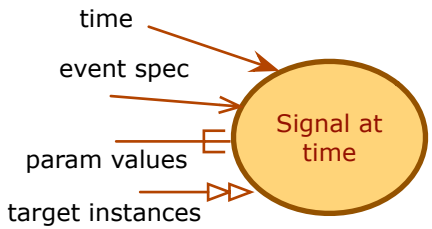
// Tell UI that a cabin has arrived

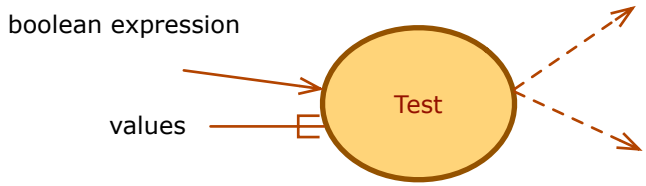
Copyright © 2021 by Leon Starr / MIT Open Source

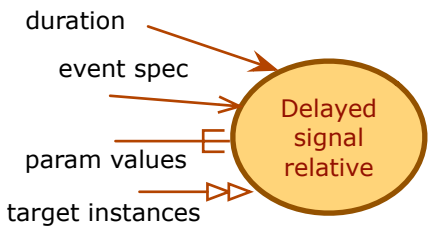


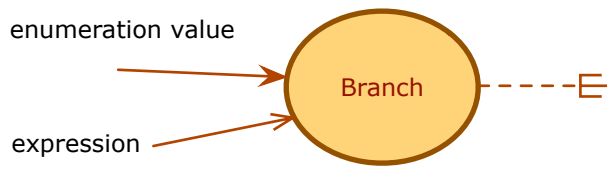


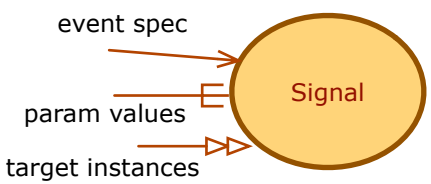
Copyright © 2021 by Leon Starr / MIT Open Source

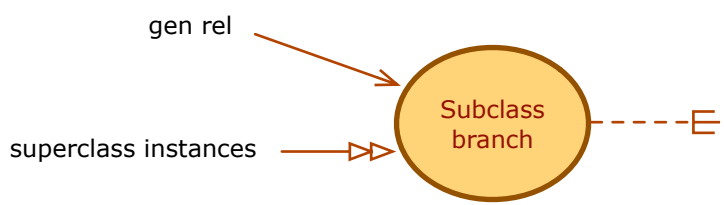












Copyright © 2021 by Leon Starr / MIT Open Source

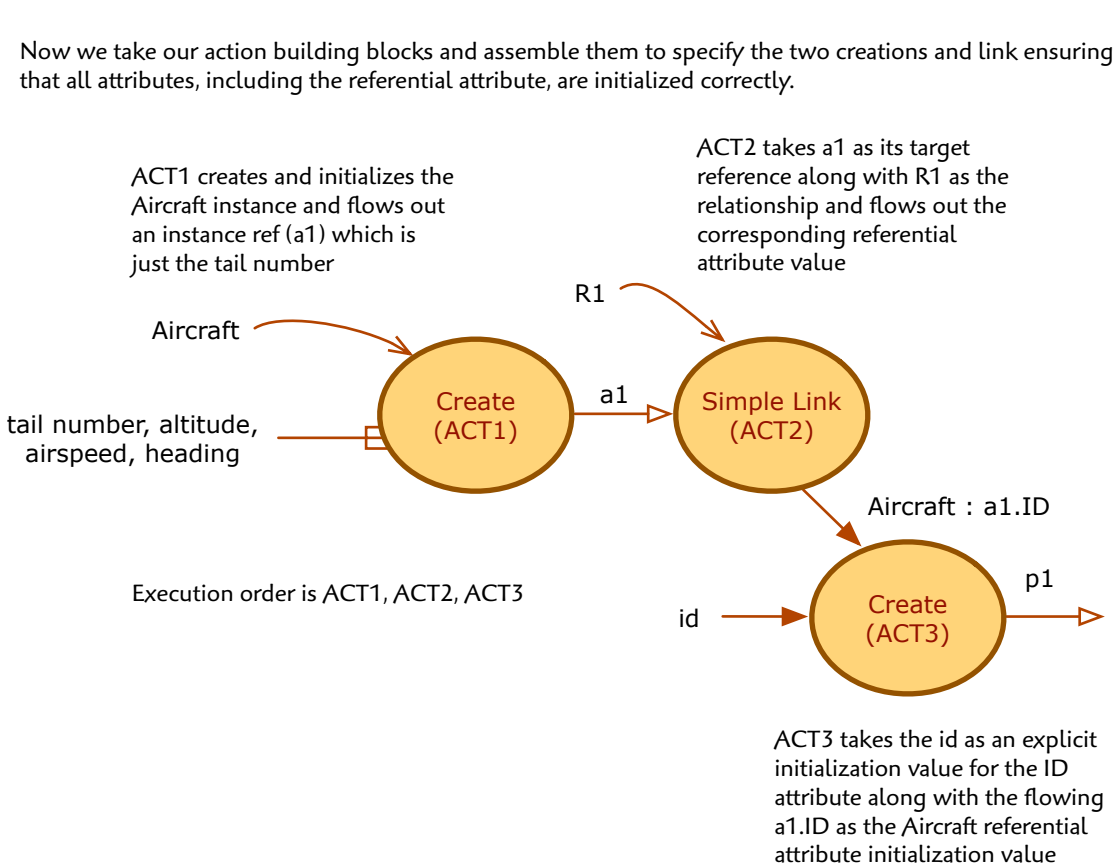
Let's examine how the following Scrall statement is broken down into metamodel components (in the Link Action Subsystem)

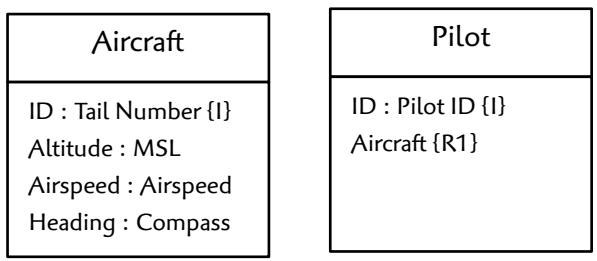
```
*Aircraft( tail number, altitude, airspeed, heading ) &R1 *Pilot( id )
```

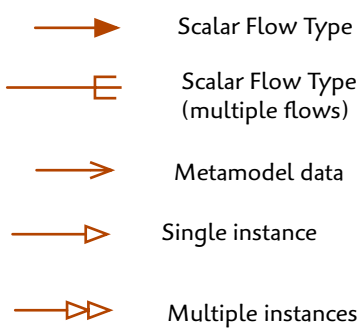
We'll assume that the parameter values and names match in the attribute initialization expressions

We'll assume that the parameter values and names match in the attribute initialization expressions

Now we take our action building blocks and assemble them to specify the two creations and link ensuring that all attributes, including the referential attribute, are initialized correctly.



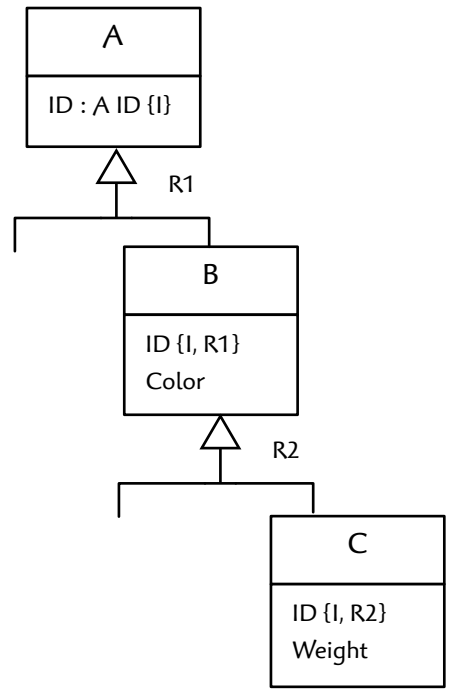




Copyright © 2021 by Leon Starr / MIT Open Source

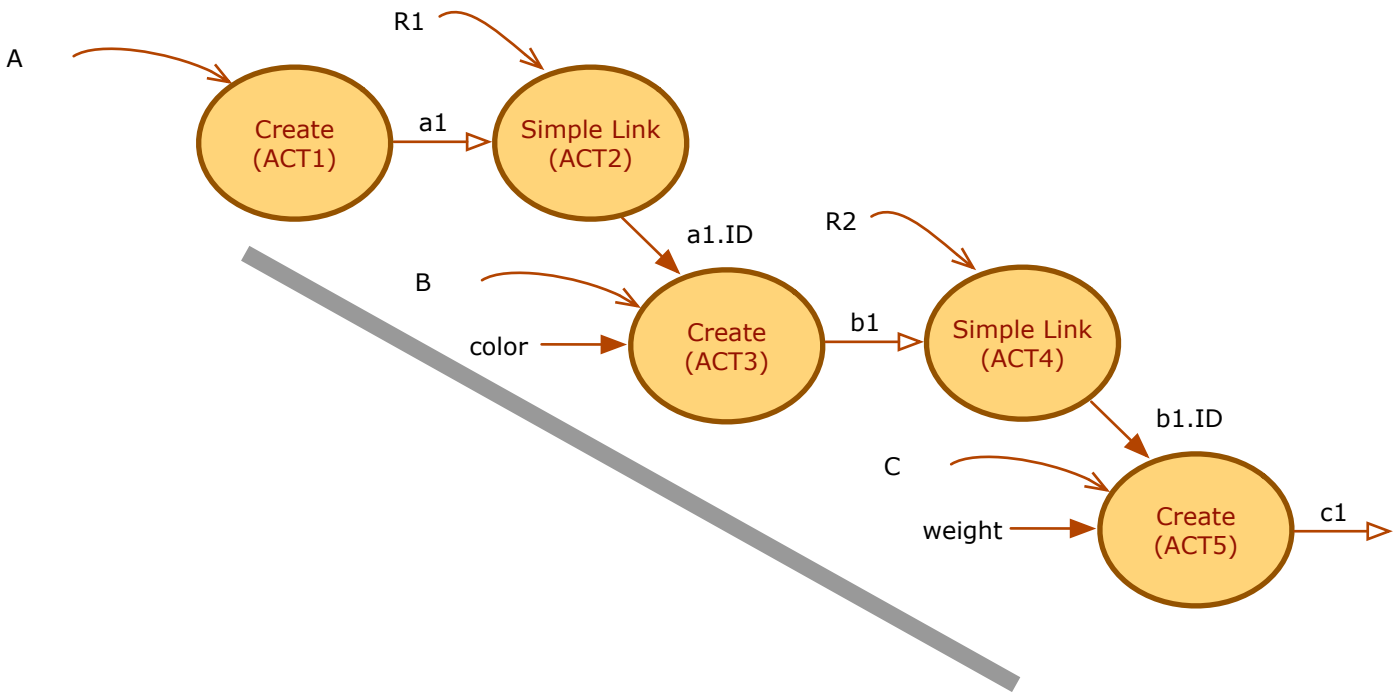
```
*C(weight) &R2 *B(color) &R1 *A
```

When creating an instance in a composition of generalization relationships, the highest superclass must be instantiated with one subclass instance at each level below



We'll assume that the parameter values and names match in the attribute initialization expressions

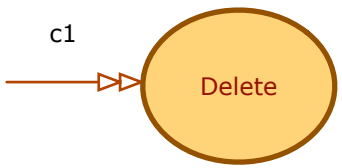
Regardless of ordering in the action language, we always build from the top down so that we have our referential attribute values as input to the downstream create action



Copyright © 2021 by Leon Starr / MIT Open Source

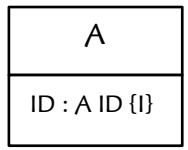
!*c1

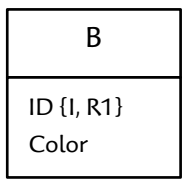
The action language will delete the instance of C and, since it is part of a generalization, remove all related superclasses as well

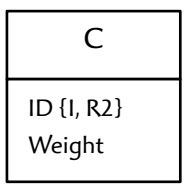


When deleting a subclass instance, the delete primitive will work its way up to all related superclasses (and up from any corresponding subclasses, if in a compound generalization)

Since the Delete Action does not output any instance flow, we can't chain them together to illustrate a separate delete on each generalization instance. Instead, we assume that all this goes on within the Delete Action.







Copyright © 2021 by Leon Starr / MIT Open Source

Here we migrate a subclass instance

ATC
ID : Employee ID {I}

```
atcOn >> Off Duty Controller( Time logged off : Date.Now hms )
```

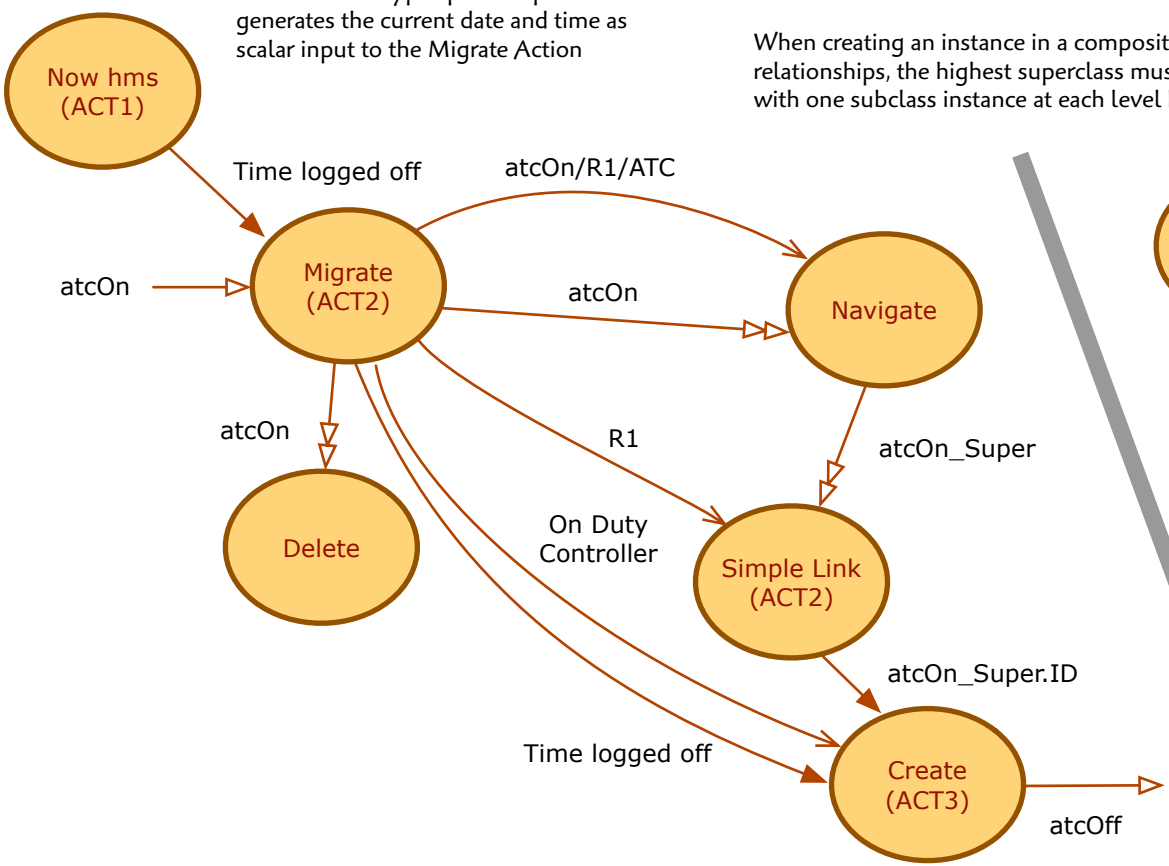
Here we delete the original subclass instance and create a new one initialized with the correct referential attribute and explicit attribute initialization.

On Duty Controller
ID {I, R1}

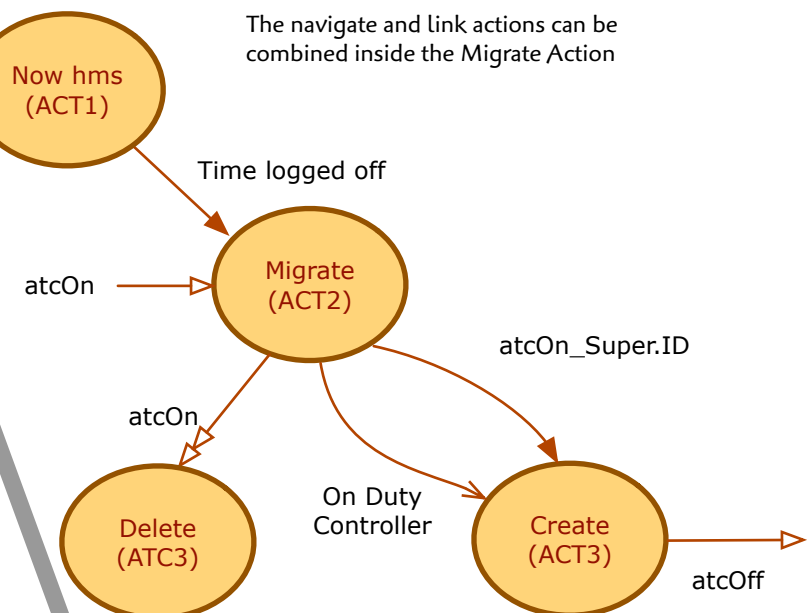
Off Duty Controller
ID {I, R1} Time logged off : Date

The Date data type specific operation generates the current date and time as scalar input to the Migrate Action

When creating an instance in a composition of generalization relationships, the highest superclass must be instantiated with one subclass instance at each level below



The navigate and link actions can be combined inside the Migrate Action

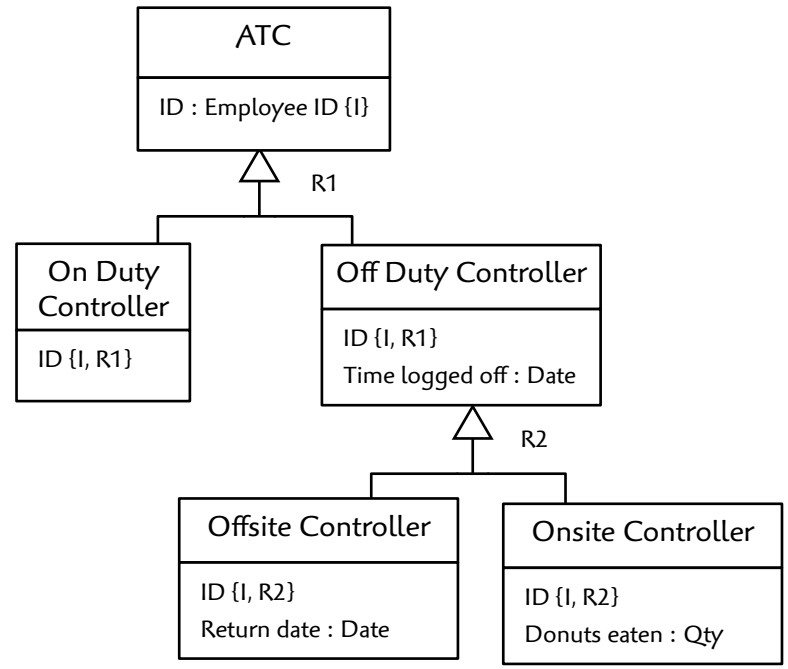


Copyright © 2021 by Leon Starr / MIT Open Source

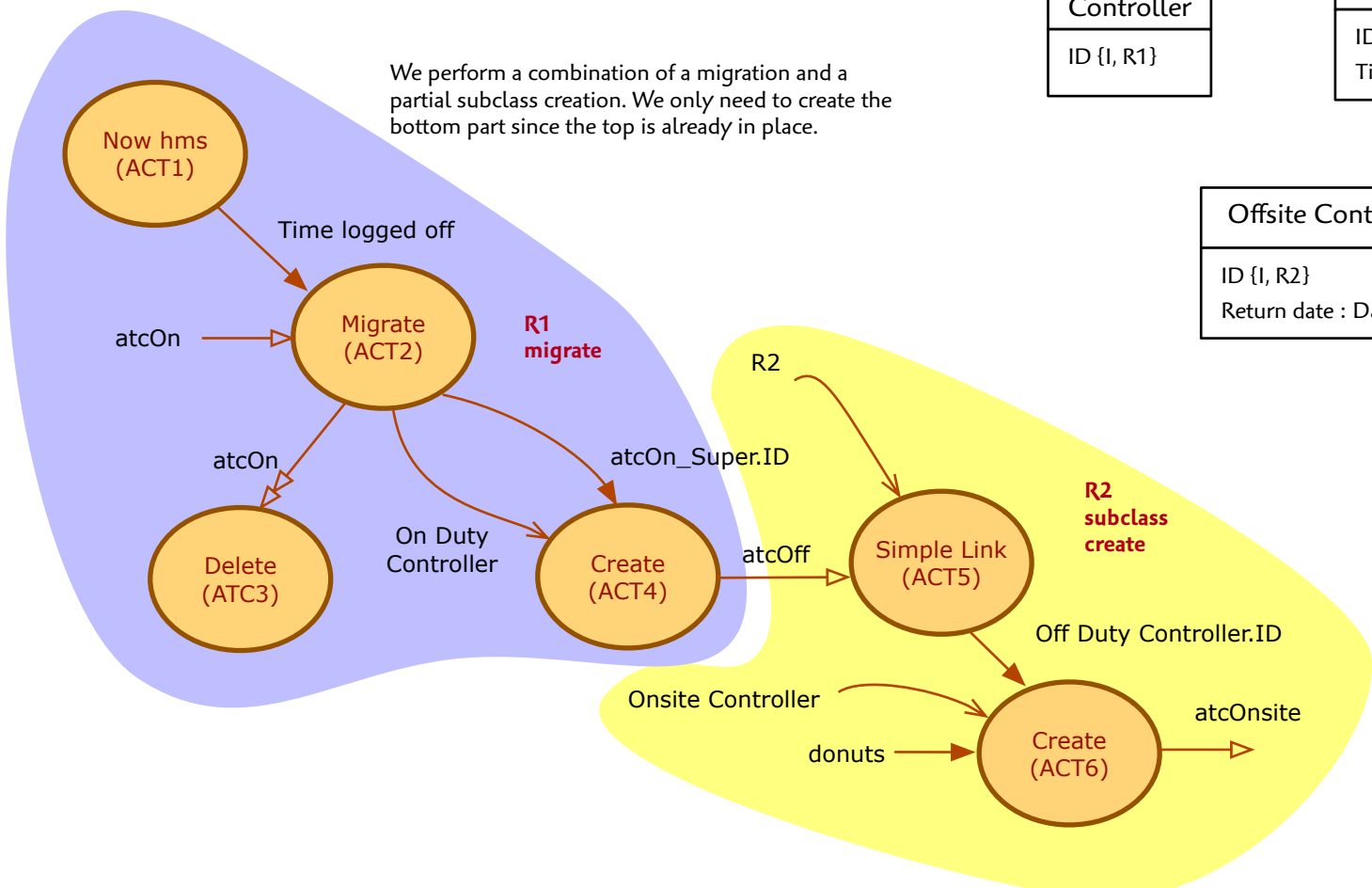
Here we migrate a subclass instance

```
atcOn >> Off Duty Controller( Time logged off : Date.Now hms )
>> Onsite Controller( Donuts eaten : donuts )
```

Here we migrate to a Composition Generalization and need to specify two subclasses for creation



We perform a combination of a migration and a partial subclass creation. We only need to create the bottom part since the top is already in place.



Copyright © 2021 by Leon Starr / MIT Open Source