A The *StockManager* Component Partial Specification

Listing 4: Kmelia specification Vendor

```
COMPONENT StockManager
INTERFACE
  provides : {newReference, removeReference, storeItem, orderItem}
  requires : {authorisation}
USES {STOCKLIB}
TYPES
  Reference :: range 1..maxRef
VARIABLES
  vendorCodes : setOf Integer; //authorised administrators
  obs catalog : setOf Reference; // product id = index of the arrays
  plabels : array [Reference] of String; //product description
  pstock : array [Reference] of Integer //product quantity
INVARIANT
  obs @borned: size(catalog)<=maxRef,
  @referenced: forall ref : Reference | includes(catalog,ref) implies
     (plabels[ref] <> emptyString and pstock[ref] <> noQuantity),
  @notreferenced: forall ref : Reference | excludes(catalog,ref) implies
     (plabels[ref] = emptyString and pstock[ref] = noQuantity)
INITIALIZATION
  catalog := emptySet;
  vendorCodes := emptySet;     //filled by a required service
  plabels:= arrayInit(plabels,emptyString); //consistent with ..
  pstock := arrayInit(pstock,noQuantity);  //..empty catalog
SERVICES
########## provided services
provided newReference () : Integer  //Result = ProductId or noReference
Interface
   calrequires : {ask_code} #required from the caller
   intrequires : {getNewReference}
Pre
   obs size(catalog) < maxRef #the catalog is not full
Variables # local to the service
  c : Integer;      # c : input code given by the user
  res:Reference;
  d : String;           # product description
Initialization
  res := noQuantity;
Behavior
Init i # the initial state
Final f # a final state
{  i -- c := __CALLER!!ask_code() --> e1,
       # gets the password on the ask_code (service) channel
  e1 -- [not(c in vendorCodes)]
        display("adding a reference is not allowed") --> end,
  e1 -- [c in vendorCodes] __CALLER?msg(d) --> e2,
       # gets the product description
  e2 -- [d = emptyString]
        display("adding an EmptySet description is not allowed") --> end,
  e2 -- [d <> emptyString] res := __SELF!!getNewReference() --> e4,
  e4 -- {catalog := including(catalog,res); //add new reference
         pstock[res] := 0; //default stock is null
         plabels[res] := d //product description is the one provided
        }--> end,
  end -- __CALLER!!newReference(res) --> f
       # the caller is informed from the Result and the service ends.
}
Post
 obs @resultRange:((Result >=1 and Result <=maxRef) or (Result = noReference)),
 obs @resultValue:(Result <> noReference) implies (notIn(old(catalog),Result)
                  and catalog = add(old(catalog),Result)),
 obs @noresultValue:(Result = noReference) implies Unchanged{catalog},
 @refAndQuantity: (Result <> noReference) implies
         (pstock[Result] = 0 and plabels[Result] <> emptyString and
         (forall i : Reference | (i <> Result) implies
            (pstock[i] = old(pstock)[i] and plabels[i] = old(plabels)[i] ))
         ),
 @NorefAndQuantity: (Result = noReference) implies Unchanged{pstock,plabels}
End
/////////internal provided services/////////////
query provided getNewReference () : Reference
Pre
  obs size(catalog) < maxRef #possible if the catalog is not full
Variables
```

```
  i: Reference;
Initialization
  i := 1 ;
Behavior
Init i
Final f
{  i -- {while (pstock[i] <> noReference) do
           i := i +1
         endwhile} --> res,
  res -- __CALLER!!getNewReference(i) --> f
}
Post
  obs notIn(catalog,Result) //the reference is really new
End
########## required services (partial description)
required ask_code() : Integer
  //default assertion = true and No LTS
End
required authorisation() : setOf Integer  //...
End
END_SERVICES
```

B The *Vendor* Component Partial Specification

Listing 5: Kmelia specification Vendor

```
COMPONENT Vendor
INTERFACE
  provides : {vending}
  requires : {addItem, removeItem, increaseItem, decreaseItem}
USES {STOCKLIB}
CONSTANTS
obs noID : Integer := -1;
VARIABLES
  obs orders : setOf ProductItem;  # observable user card
  vendorId : Integer               # vendor personal code
INITIALIZATION
  orders := emptySet;
  vendorId := noID
SERVICES
########## provided services
# The main (provided) service is vending.
provided vending ()
Interface
   extrequires : {addItem, removeItem, increaseItem, decreaseItem}
Pre true
Variables # local to the service
  choice : CommandChoice ;     # command choice : addItem, ...
  ref : Integer;               # product reference given by the user
  qty : Integer;               # product quantity given by the user
  desc : String;               # product description given by the user
  pi : Integer;
Behavior // The behaviour is specified as an infinite loop
Init i # i is the initial state
Final f # f is a final state
{ i -- {
        displayMenu();                    # call an internal action
        display("Please enter your choice");
        choice := readCommandChoice()     # call an internal action
        } --> e0,
  e0 --[choice = stop] display("bye bye") --> f,
  //final state = end of vending
  e2 --[choice = add]  _addItem!!addItem() --> e10,
  e0 --[choice <> stop] display("Product reference") --> e1,
  e1 -- ref:=readInt() --> e2,
  e2 --[choice = remove] _removeItem!!removeItem(ref) --> e20,
  e2 --[choice = store] {_increaseItem!!increaseItem(ref,readInt())} --> e30,
  e2 --[choice = order] _decreaseItem!!decreaseItem(ref,readInt()) --> e40,
  //------ add Item
  e10 <<code>>, #subservice code is available here
  e10 -- {desc:=readString(); // product description
          _addItem!msg(desc)          } --> e11,
  e11 -- _addItem??addItem(pi) --> e12,
  e12 -- {if (pi <> noReference)
```

```
            then display("New reference : "+asString(pi))
            endif } --> i
  //------ other choices ...
}
Post obs true
End

provided code() : Integer
/* daemon service that answers the code of the user */
  Pre obs true
Behavior
Init e0
Final f
{ e0 -- [vendorId = noID] display("Enter your vendor code") --> e1,
  e0 -- [vendorId <> noID] __CALLER!!code(vendorId) --> f,
  e1 -- vendorId:=readInt() --> e0
}
Post obs Result <> noID
End
########## required services (partial description)
required addItem () : Integer
Interface
  subprovides : {code}
  Virtual Variables
     catalogFull : Boolean;
     catalogEmpty : Boolean     //possibly catalogSize
  Virtual Invariant not(catalogEmpty and catalogFull)
  Pre not catalogFull
  //No LTS
  Post (Result <> noReference) implies (not catalogEmpty)
End
END_SERVICES
```

C The *StockSystem* Component Partial Specification

Listing 6: Kmelia specification Vendor

```
COMPONENT StockSystem
INTERFACE
  provides : {vending}
  requires : {authorisation}
SERVICES
  //services of the composite
  ...
END_SERVICES
COMPOSITION
  Assembly
    Components
      sm : StockManager;
      ve : Vendor
    Links ////////////assembly links/////////
      lref: p-r sm.newReference, ve.addItem
          context mapping
            ve.catalogEmpty == empty(sm.catalog),
            ve.catalogFull == size(sm.catalog) = MaxInt
          sublinks : {lcode}
      lcode: r-p sm.ask_code, ve.code
      ...
  End // assembly
  Promotion
    Links ////////////promotion links/////////
      lvend: p-p ve.vending, SELF.vending
      laut: r-r sm.authorisation, SELF.authorisation
END_COMPOSITION
```

D The derived Event-B models

D.1 StockLib

CONTEXT StockLib

EXTENDS Default

CONSTANTS

References
MaxRef
NullInt
NoQuantity
NoReference

AXIOMS

axm5 : $References = 1 \mathinner{..} MaxRef$
axm1 : $MaxRef = 100$
axm2 : $NullInt = -1$
axm3 : $NoQuantity = -2$
axm4 : $NoReference = -3$

END

D.2 StockManager

MACHINE StockManager

SEES StockLib

VARIABLES

vendorCodes
catalog obs
plabels
pstock
Result_newReference obs

INVARIANTS

inv5 : $vendorCodes \subseteq \mathbb{Z}$
inv2 : $catalog \in \mathbb{P}(References)$
obs
inv7 : $finite(catalog)$
obs
inv3 : $plabels \in 1 \mathinner{..} MaxRef \rightarrow String$
inv4 : $pstock \in 1 \mathinner{..} MaxRef \rightarrow \mathbb{Z}$
• borned : $card(catalog) \leq MaxRef$
obs
• referenced : $\forall ref1 \cdot (ref1 \in References \wedge ref1 \in catalog \Rightarrow plabels(ref1) \neq EmptyString \wedge pstock(ref1) \neq NoQuantity)$
• notreferenced : $\forall ref2 \cdot (ref2 \in References \wedge ref2 \notin catalog \Rightarrow plabels(ref2) = EmptyString \wedge pstock(ref2) = NoQuantity)$
inv6 : $Result\_newReference \in \mathbb{Z}$
obs

EVENTS

Initialisation

begin
act1 : $vendorCodes := \varnothing$
act2 : $catalog := \varnothing$
act3 : $plabels := (1 \mathinner{..} MaxRef) \times \{EmptyString\}$
act4 : $pstock := (1 \mathinner{..} MaxRef) \times \{NoQuantity\}$
act5 : $Result\_newReference := 0$
end

Event $newReference \mathrel{\widehat{=}}$

any
new_Result
$new\_catalog$
new_pstock
$new\_plabels$
where
grd8 : $card(catalog) < MaxRef$
obs
grd1 : $new\_Result \in \mathbb{Z}$
obs

grd2 : $new\_catalog \in \mathbb{P}(References)$
obs
grd11 : $finite(new\_catalog)$
obs
grd3 : $new\_plabels \in 1 \mathinner{..} MaxRef \rightarrow String$
grd4 : $new\_pstock \in 1 \mathinner{..} MaxRef \rightarrow \mathbb{Z}$
grd5 : $(new\_Result > 0 \land new\_Result \leq MaxRef) \lor new\_Result = NoReference$
obs
grd6 : $new\_Result \neq NoReference \Rightarrow$
$new\_Result \notin catalog$
$\land\ new\_catalog = catalog \cup \{new\_Result\}$
obs
grd7 : $new\_Result = NoReference \Rightarrow new\_catalog = catalog$
obs
grd9 : $new\_Result \neq NoReference \Rightarrow$
$new\_pstock(new\_Result) = 0 \land$
$new\_plabels(new\_Result) \neq EmptyString \land$
$(\forall ii \cdot (ii \in 1 \mathinner{..} MaxRef \land ii \neq new\_Result \Rightarrow$
$new\_pstock(ii) = pstock(ii) \land$
$new\_plabels(ii) = plabels(ii)$
$))$
grd10 : $new\_Result = NoReference \Rightarrow$
$new\_pstock = pstock \land$
$new\_plabels = plabels$

then

act1 : $Result\_newReference := new\_Result$
act2 : $catalog := new\_catalog$
act3 : $pstock := new\_pstock$
act4 : $plabels := new\_plabels$

end

END

D.3 Vendor_addItem

MACHINE Vendor_addItem

SEES StockLib

VARIABLES

`catalogFull`
`catalogEmpty`
`Result_addItem`

INVARIANTS

inv1 : $catalogFull \in BOOL$
inv2 : $catalogEmpty \in BOOL$
• notFullEmpty : $\neg\ (catalogEmpty = TRUE \land catalogFull = TRUE)$
inv4 : $Result\_addItem \in \mathbb{Z}$

EVENTS

Initialisation

begin

act1 : $catalogFull := FALSE$
act2 : $catalogEmpty := TRUE$
act3 : $Result\_addItem :\in \mathbb{Z}$

end

Event $addItem \mathrel{\widehat{=}}$

any

new_Result
$new\_catalogEmpty$
$new\_catalogFull$

where

pre_addItem : $\neg\ (catalogFull = TRUE)$
grd2 : $new\_Result \in \mathbb{Z}$
grd6 : $new\_catalogEmpty \in BOOL$
grd5 : $new\_catalogFull \in BOOL$
Post_addItem : $new\_Result \neq NoReference \Rightarrow$
$new\_catalogEmpty = FALSE \land$
$new\_catalogFull \in BOOL$
Post_addItem2 : $new\_Result = NoReference$
$\Rightarrow$
$new\_catalogEmpty = catalogEmpty \land$
$new\_catalogFull = catalogFull$

then

addItem_result : $Result\_addItem := new\_Result$
addItem_empty : $catalogEmpty := new\_catalogEmpty$
addItem_full : $catalogFull := new\_catalogFull$
end

END

D.4 v_addItem_sm_newReference

MACHINE v_addItem_sm_newReference

REFINES Vendor_addItem

SEES StockLib

VARIABLES

catalogEmpty
catalogFull
Result_addItem
catalog

INVARIANTS

inv1 : $catalog \in \mathbb{P}(References)$
inv6 : $finite(catalog)$
borned : $card(catalog) \leq MaxRef$
assemblyEmpty : $catalogEmpty = bool(card(catalog) = 0)$
assemblyFull : $catalogFull = bool(card(catalog) = MaxRef)$

EVENTS

Initialisation
extended
begin
act1 : catalogFull := FALSE
act2 : catalogEmpty := TRUE
act3 : Result_addItem :∈ $\mathbb{Z}$
act4 : $catalog := \varnothing$
end

Event $newReference \mathrel{\widehat{=}}$

refines *addItem*

any
new_Result
$new\_catalog$
where
pre_newReference : $card(catalog) < MaxRef$
grd11 : $new\_Result \in \mathbb{Z}$
grd64 : $new\_catalog \in \mathbb{P}(References)$
grd10 : $finite(new\_catalog)$
post_newRef1 : $((new\_Result > 0 \land new\_Result \leq MaxRef) \lor new\_Result = NoReference)$
post_newRef2 : $new\_Result \neq NoReference \Rightarrow new\_Result \notin catalog \land new\_catalog = catalog \cup \{new\_Result\}$
post_newRef3 : $new\_Result = NoReference \Rightarrow new\_catalog = catalog$
with
new_catalogEmpty : new_catalogEmpty = bool(card(new_catalog) = 0)
new_catalogFull : new_catalogFull = bool(card(new_catalog) = MaxRef)
then
addItem_result : $Result\_addItem := new\_Result$
addItem_empty : $catalogEmpty := bool(card(new\_catalog) = 0)$
addItem_full : $catalogFull := bool(card(new\_catalog) = MaxRef)$
act34 : $catalog := new\_catalog$
end

END