

This separate appendix for the FESCA@ETAPS 2010 article is available at the Kmelia website⁵.

A Kmelia specification

A.1 The Kmelia assembly StockSystem

```

ASSEMBLY
COMPONENTS
  ve : Vendor ;
  sm : StockManager
LINKS
@lref: r-p ve.addItem, sm.newReference
Context mapping
  catalogEmpty == empty(sm.catalog),
  catalogFull == size(sm.catalog) = MaxInt
  sublinks : {lcode}
End
@lcode: p-r ve.code, sm.ask_code
...
END_LINKS

```

A.2 The Kmelia component StockManager

```

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
##### required services (partial description)
required ask_code() : Integer
//default assertion = true and No LTS
End
required authorisation() : setOf Integer //...
End
##### provided services
provided newReference() : Integer //Result = ProductId or noReference
Interface
  calrequires : {ask_code} #required from the caller
  intrequires : {getNewReference}
Pre
  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,

```

⁵ http://www.lina.sciences.univ-nantes.fr/coloss/download/fesca10_app.pdf

```

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
@resultRange: ((Result >= 1 and Result <= maxRef) or (Result = noReference)),
@resultValue: (Result <> noReference) implies (notIn(old(catalog),Result)
and catalog = add(old(catalog),Result)),
@noresultValue: (Result = noReference) implies Unchanged{catalog},
local @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] ))
);
local @NorefAndQuantity: (Result = noReference) implies Unchanged{pstock,plabels}
End
//////////internal provided services//////////
query provided getNewReference () : Reference
Pre
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
notIn(catalog,Result) //the reference is really new
End
END_SERVICES

```

A.3 The Kmelia component 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
  @ref: (Result <> noReference) implies (not catalogEmpty),
  @noref: (Result = noReference) implies Unchanged{catalogEmpty, catalogFull}
End
END_SERVICES

```

B Event-B Models

B.1 The Event-B context *StockLib*

```

CONTEXT StockLib
EXTENDS Default
CONSTANTS
  References
  MaxRef
  NullInt
  NoQuantity
  NoReference
AXIOMS
  axm5 :  $References = 1..MaxRef$ 
  axm1 :  $MaxRef = 100$ 
  axm2 :  $NullInt = -1$ 
  axm3 :  $NoQuantity = -2$ 
  axm4 :  $NoReference = -3$ 
END

```

B.2 The Event-B model *StockManager_obs*

```

MACHINE StockManager_O
SEES StockLib
VARIABLES
  catalog
  Result_newReference
INVARIANTS
  type1 :  $catalog \in \mathbb{P}(References)$ 
  type2 :  $finite(catalog)$ 
  @borned :  $card(catalog) \leq MaxRef$ 
  type3 :  $Result\_newReference \in \mathbb{Z}$ 
EVENTS
Initialisation
  begin
    act2 :  $catalog := \emptyset$ 
    act5 :  $Result\_newReference := 0$ 
  end
Event newReference =
  when
    grd1 :  $card(catalog) < MaxRef$ 
  then
    act1 :  $Result\_newReference, catalog :$ 
      (
         $((Result\_newReference' > 0 \wedge Result\_newReference' \leq MaxRef) \vee Result\_newReference' = NoReference)$ 
         $\wedge$ 
         $(Result\_newReference' \neq NoReference \Rightarrow$ 
           $Result\_newReference' \notin catalog \wedge catalog' = catalog \cup \{Result\_newReference'\})$ 
        )
         $\wedge$ 
         $(Result\_newReference' = NoReference \Rightarrow catalog' = catalog)$ 
      )
  end
END

```

B.3 The Event-B model *StockManager*

```

MACHINE StockManager
REFINES StockManager_O
SEES StockLib
VARIABLES
  catalog
  Result_newReference
  vendorCodes
  plabels
  pstock
INVARIANTS
  type5 :  $vendorCodes \subseteq \mathbb{Z}$ 
  type6 :  $plabels \in 1..MaxRef \rightarrow String$ 
  type7 :  $pstock \in 1..MaxRef \rightarrow \mathbb{Z}$ 

```

```

@notreferenced:  $\forall \text{ref2} \cdot (\text{ref2} \in \text{References} \wedge \text{ref2} \notin \text{catalog} \Rightarrow \text{labels}(\text{ref2}) = \text{EmptyString} \wedge \text{pstock}(\text{ref2}) = \text{NoQuantity})$ 
EVENTS
Initialisation
  extended
  begin
    act2: catalog :=  $\emptyset$ 
    act5: Result_newReference := 0
    act8: vendorCodes :=  $\emptyset$ 
    act7: labels :=  $(1..MaxRef) \times \{\text{EmptyString}\}$ 
    act6: pstock :=  $(1..MaxRef) \times \{\text{NoQuantity}\}$ 
  end
Event newReference =
refines newReference
  when
    grd1: card(catalog) < MaxRef
  then
    act2: Result_newReference, catalog, pstock, labels : |
      (
        ((Result_newReference' > 0  $\wedge$  Result_newReference'  $\leq$  MaxRef)  $\vee$  Result_newReference' = NoReference)
         $\wedge$ 
        (Result_newReference'  $\neq$  NoReference  $\Rightarrow$ 
          Result_newReference'  $\notin$  catalog  $\wedge$  catalog' = catalog  $\cup$  {Result_newReference'})
        )
         $\wedge$ 
        (Result_newReference' = NoReference  $\Rightarrow$  catalog' = catalog)
         $\wedge$ 
        (Result_newReference'  $\neq$  NoReference  $\Rightarrow$ 
          pstock'(Result_newReference') = 0  $\wedge$  labels'(Result_newReference')  $\in$  String  $\setminus$  {EmptyString}  $\wedge$ 
          ( $\forall ii \cdot (ii \in 1..MaxRef \wedge ii \neq \text{Result\_newReference}' \Rightarrow \text{pstock}'(ii) = \text{pstock}(ii) \wedge \text{labels}'(ii) = \text{labels}(ii))$ )
        )
         $\wedge$ 
        (Result_newReference' = NoReference  $\Rightarrow$  pstock' = pstock  $\wedge$  labels' = labels)
      )
    end
  end
END

```

B.4 The Event-B model Vendor_addItem

```

MACHINE Vendor_addItem
SEES StockLib
VARIABLES
  catalogFull
  catalogEmpty
  Result_addItem
INVARIANTS
  inv1: catalogFull  $\in$  BOOL
  inv2: catalogEmpty  $\in$  BOOL
  @notFullEmpty:  $\neg(\text{catalogEmpty} = \text{TRUE} \wedge \text{catalogFull} = \text{TRUE})$ 
  inv4: Result_addItem  $\in$   $\mathbb{Z}$ 
EVENTS
Initialisation
  begin
    act1: catalogFull, catalogEmpty : |  $(\neg(\text{catalogEmpty}' = \text{TRUE} \wedge \text{catalogFull}' = \text{TRUE}))$ 
    act3: Result_addItem :  $\in$   $\mathbb{Z}$ 
  end
Event addItem =
  when
    grd1:  $\neg(\text{catalogFull} = \text{TRUE})$ 
  then
    act1: Result_addItem, catalogEmpty, catalogFull : |
      ((Result_addItem'  $\neq$  NoReference  $\Rightarrow$ 
        catalogEmpty' = FALSE  $\wedge$  catalogFull'  $\in$  BOOL)
         $\wedge$ 
        (Result_addItem' = NoReference  $\Rightarrow$ 
          catalogEmpty' = catalogEmpty  $\wedge$  catalogFull' = catalogFull)
        )
    end
  end
END

```

B.5 The Event-B model 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:  $f_{inite}(catalog)$ 
  borned:  $card(catalog) \leq MaxRef$ 
  assemblyEmpty:  $catalogEmpty = bool(card(catalog) = 0)$ 
  assemblyFull:  $catalogFull = bool(card(catalog) = MaxRef)$ 
  inv7:  $(\neg(catalogFull = TRUE)) \Rightarrow (card(catalog) < MaxRef)$ 
EVENTS
Initialisation
  extended
  begin
    act1:  $catalogFull, catalogEmpty : |(\neg(catalogEmpty' = TRUE \wedge catalogFull' = TRUE))$ 
    act3:  $Result\_addItem : \in \mathbb{Z}$ 
    act4:  $catalog := \emptyset$ 
  end
Event newReference =
refines addItem
  when
    grd1:  $\neg(catalogFull = TRUE)$ 
  then
    act1:  $Result\_addItem, catalog, catalogEmpty, catalogFull : |$ 
      (
         $((Result\_addItem' > 0 \wedge Result\_addItem' \leq MaxRef) \vee Result\_addItem' = NoReference)$ 
         $\wedge$ 
         $(Result\_addItem' \neq NoReference \Rightarrow$ 
           $Result\_addItem' \notin catalog \wedge catalog' = catalog \cup \{Result\_addItem'\})$ 
        )
         $\wedge$ 
         $(Result\_addItem' = NoReference \Rightarrow catalog' = catalog)$ 
         $\wedge$ 
         $(catalogEmpty' = bool(card(catalog') = 0))$ 
         $\wedge$ 
         $(catalogFull' = bool(card(catalog') = MaxRef))$ 
      )
  end
END

```