Refinement in UML-B

May, 2010

Colin Snook  University of Southampton
Extending Context Diagrams
Starting from a Context Diagram with a Class Type

CONTEXT
   cθ

SETS
   T  // ClassType

CONSTANTS
   a  // attribute of T

AXIOMS
   a.type : a ∈ T → BOOL
   Axiom1 : ∃t. t ∈ T ⇒ a(t) = TRUE

END
Make an Extension of this Context

1) Select context, 
2) Click Button in Properties, 
3) Makes a starting point for extending
Provides a basis for extending classtypes

An empty ExtendedClassType

```
CONTEXT
  c1
  c0
END
```
Add Attributes and Axioms to Extend ClassType

CONTEXT
  c1

EXTENDS
  c∅

CONSTANTS
  new  // attribute of T

AXIOMS
  new.type : new ∈ T → N
  newAxiom : ∀t.t∈T ⇒ new(t) < 100

END
Refining Class Diagrams
Starting from a Class Diagram

MACHINE
  m\theta
SEES
  m\theta\_implicitContext
VARIABLES
  as0   // attribute of cl0
ININVARIANTS
  as0\_type : as0 \in cl0 \rightarrow \text{BOOL}
Invariant1 : as0 \triangleright \{\text{FALSE}\} \neq \emptyset

EVENTS

INITIALISATION ▲
STATUS
  ordinary
BEGIN
  as0\_init : as0 \Rightarrow cl0 \times \{\text{FALSE}\}
END

ev0 ▲
STATUS
  ordinary
ANY
  thisCl0   // contextual instance of class cl0
WHERE
  thisCl0\_type : thisCl0 \in cl0
  ev0\_Guard1 : (as0\triangleright\{thisCl0 \Rightarrow \text{TRUE}\}) \triangleright \{\text{FALSE}\} \neq \emptyset
THEN
  ev0\_Action1 : as0(thisCl0) \Rightarrow \text{TRUE}
END

END
“Make a refinement of this Machine”

Select Machine, click Button in Properties, Makes a starting point for refinement
Basis for refinement

MACHINE
  m1
REFINES
  m0
SEES
  m1_implicitContext

VARIABLES
  as0 // inherited attribute of cl0

EVENTS
  INITIALISATION
  STATUS
    ordinary
  BEGIN
    as0.init : as0 := cl0 \times \{FALSE\}
  END
  ev0
    STATUS
      ordinary
  REFINES
    ev0
  ANY
    thisCl0 // contextual instance of refined class cl0
    WHERE
      thisCl0.type : thisCl0 \in cl0
      ev0.Guard1 : (as0\{thisCl0 \rightarrow TRUE\}) \cup \{FALSE\} \neq \emptyset
    THEN
      ev0.Action1 : as0(thisCl0) = TRUE
  END

END
Refine a Class

- Add new attributes/associations
- Refine existing events (strengthen guards, add actions, split, merge)
- Add new events (can only alter new variables)
- Add new invariants
Example – Bank Accounts and ATMs

ABSTRACT
Bank accounts have a balance which is zero when the account is opened. Money may then be deposited in the account, increasing the balance by some amount, or withdrawn, depleting the balance by some amount.

REFINEMENT
A card is associated with an account and withdrawals are made via an ATM machine. The card is inserted into the ATM and either a successful withdrawal is completed and the card is ejected or the transaction fails.

(based on a case study by Mar Yah Said)
Example – Abstract

Note: for this example it is necessary to rename the contextual class instance from *self* to *thisAccount* so that it can be disambiguated when the event *withdraw* is moved to a different class.

```
acGon: balance(thisAccount) := balance(thisAccount) - amount
```

**Parameter:** `amount ∈ N`

**Action:**
```
balance(thisAccount) := balance(thisAccount) - amount
```
Example – Refinement

\[ \text{icard} := \text{cd} \]

**Parameter**: \( \text{cd} \in \text{Card} \)

**Action**: \( \text{icard}(\text{thisATM}) := \text{cd} \)

**Action**: \( \text{icard} := \{\text{thisATM}\} \leftarrow \text{icard} \)
Example – Refinement

\[
\text{balance(account(icard(thisATM)))} := \text{balance(account(icard(thisATM)))} - \text{amount}
\]

Parameter: \(\text{amount} \in \mathbb{N}_1\)

Witness: \(\text{thisAccount} = \text{account(icard(thisATM))}\)

Action: \(\text{balance(account(icard(thisATM)))} := \text{balance(account(icard(thisATM)))} - \text{amount}\)
Example – Refinement

acGon: balance(account(icard(thisATM)))

{\text{parameter:}} \quad \text{amount} \in \mathbb{N}_1

\begin{align*}
\text{witness:} & \quad \text{thisAccount} = \text{account(icard(thisATM))} \\
\text{action:} & \quad \text{balance(account(icard(thisATM)))} := \\
& \quad \text{balance(account(icard(thisATM)))} - \text{amount}
\end{align*}
Refining Statemachines

Slides show state sets translation.
Also works using state function translation
Starting from a Simple Statemachines
“Make a refinement of this Machine”

Select Machine, click BuMon in ProperGes, Makes a starting point for refinement

Same as for Class Refinement
MACHINE
m1

REFINES
m

SEES
m1_implicitContext

VARIABLES
S1 // state from refined statemachine, sm
S2 // state from refined statemachine, sm

EVENTS

INITIALISATION ⊩
STATUS
    ordinary
BEGIN
    S1.init : S1 = TRUE
    S2.init : S2 = FALSE
END

t1 ⊩
STATUS
    ordinary
REFINES
t1
WHEN
    sm_isin_S1 : S1 = TRUE
THEN
    sm_leaveState_S1 : S1 = FALSE
    sm_enterState_S2 : S2 = TRUE
END

t2 ⊩
What we can’t do (when refining statemachines)

Cannot add new states
  state sets – would contradict the existing partition invariant
  state function – would alter the existing enumerated type

Cannot add completely new transitions
  new events must not alter old variables (e.g. state change would)
What can we do?

Refine the existing transitions
  strengthen guards
  add actions to alter any new variables
  split transitions (as long as they have same source and target state)

Can add things to a state
  Invariants
  Nested State-machines
Transition Splitting – Preparing for a Nested Statemachine

`t2a`  
`STATUS`  
`  ordinary`  
`REFINES`  
`  t2`  
`WHEN`  
`  sm_isin_S2 : S2 = TRUE`  
`THEN`  
`  sm_leaveState_S2 : S2 := FALSE`  
`  sm_enterState_S1 : S1 := TRUE`  
`END`

`t2b`  
`STATUS`  
`  ordinary`  
`REFINES`  
`  t2`  
`WHEN`  
`  sm_isin_S2 : S2 = TRUE`  
`THEN`  
`  sm_leaveState_S2 : S2 := FALSE`  
`  sm_enterState_S1 : S1 := TRUE`  
`END`

t2 has been split into 2 ‘cases’ (both refine t2)
Adding a Nested Statemachine elaborates
Translation of Refinement

**INVARINTS**

- $S21.type : S21 \in BOOL$
- $S22.type : S22 \in BOOL$
- $S23.type : S23 \in BOOL$

subStates $S21,S2 : \neg (S21=TRUE \land S2=FALSE)$
subStates $S22,S2 : \neg (S22=TRUE \land S2=FALSE)$
subStates $S23,S2 : \neg (S23=TRUE \land S2=FALSE)$

disjointStates $S22,S21 : \neg (S22=TRUE \land S21=TRUE)$

E

**t2b**

**STATUS**

*ordinary*

**REFINES**

$t2$

**WHEN**

- $S2sm_isin_S23 : S23 = TRUE$

**THEN**

- $sm_leaveSuperState_S2 : S2 = FALSE$
- $S2sm_leaveState_S23 : S23 = FALSE$
- $sm_enterState_S1 : S1 = TRUE$

**END**

**tnew1**

**STATUS**

*ordinary*

**WHEN**

- $S2sm_isin_S21 : S21 = TRUE$

**THEN**

- $S2sm_leaveState_S21 : S21 = FALSE$
- $S2sm_enterState_S22 : S22 = TRUE$

**END**
Example – Card Validation by PIN

In the ATM example, add a refinement to explain how card validation works.

There is a PIN number associated with a card.
A number is entered at the ATM.
If the number matches the inserted cards PIN the validation succeeds.
If the number doesn’t match the PIN the validation fails.
Refined Class Diagram

Account
- Attributes:
  - icard
- Events:
  - open
  - deposit

ATM
- Attributes:
  - icard
- State machine:
  - atm_sm
  - init
  - insert
  - withdraw
  - ejectCard
  - fail

Card
- Attributes:
  - account
- Events:
  - issueCard

parameter: $n \in \mathbb{N}$

action: $\text{pin(thisCard)} := n$
Refined Statemachine
New nested statemachine

parameter: \( n \in \mathbb{N} \)
guard: \( n = \text{pin(icard(thisATM))} \)

parameter: \( n \in \mathbb{N} \)
guard: \( n \neq \text{pin(icard(thisATM))} \)
New nested statemachine - elaboration
Summary

Extended Class Types

Refined Classes & Inherited Attributes

Moving events between classes

Statemachine refinement
  transition splitting
  nested statemachines
Provide feedback:

Experience  cfs@ecs.soton.ac.uk

Bug reports  http://sourceforge.net/tracker/?group_id=108850&atid=651669

Feature requests  http://sourceforge.net/tracker/?group_id=108850&atid=651672