Introduction to UML-B, UML-B Class Diagrams, UML-B Context Diagrams

May, 2010

Colin Snook  University of Southampton
What is UML-B?

A Graphical front-end for Event-B
  • Plug-in for Rodin

Not UML …
  • Has its own meta-model (abstract syntax)
  • Semantics inherited from translation to Event-B

… but has some similarities with UML
  • Class Diagrams
  • State Machine Diagrams

Translator generates Event-B automatically
What are the benefits?

Visualisation
- Helps understanding
- Communication

Faster modelling (allows you to experiment)
- One drawing node = several lines of B
- Extra information inferred from position of elements
  (e.g. if contained in a class or statemachine)

Provides structuring constructs
- Class
- Hierarchical state-machines
Getting Started

Install UML-B using the Rodin update site
► Help – Install, select the main Rodin update site, wait for it to retrieve the categories, select UML-B Modelling Environment under Modelling Extensions.

UML-B Perspective

UML-B New Project Wizard
► Opens a project diagram for you
► Add machines and contexts
► Double click on a machine to open a class diagram
► or on a context to open a context diagram

UML-B Menu
► Disable automatic translation (Recommended for larger models)

UML-B toolbar button
► Save and translate
Class-oriented problems

In Event-B models, often find a pattern

- **Set** $I$ (or could be constant or variable)
- **Variables** $v \in I \rightarrow T$
- **Events** $e(i,..)$ when $i \in I$, $\ldots$ then $v(i) := x$

$I$ is a set of instances of a class
$v$ is a set of values, one for each instance (a class attribute)
$e$ is a ‘family’ of identical events to assign values to $v$ (a class event)

I.e. trying to represent class-oriented problems
An Event-B model of a class-oriented problem

VARIABLES
allocation, rooms

INVARIANTS
inv1: rooms ∈ ℘(ROOMS)
inv2: allocation ∈ rooms → GUESTS

EVENTS
Initialisation
begin
act1: rooms := ∅
act2: allocation := ∅
end
Event Check_in ≜
any g, r where
grd1: g ∈ GUESTS
grd2: r ∈ rooms \ dom(allocation)
then
act1: allocation(r) := g
end

Event Add_Room ≜
any r where
grd1: r ∈ ROOMS \ rooms
then
act1: rooms := rooms ∪ {r}
end

Event Remove_Room ≜
any r where
grd1: r ∈ rooms \ dom(allocation)
then
act1: rooms := rooms \ {r}
end
In a university degree programme, students are registered on degree courses. Students must be enrolled to be registered in a course. Courses can be removed from the degree programme.

There is no need to consider multiple degree programmes - just assume we are modelling a single degree programme.
In a university degree programme, students are registered on degree courses. Students must be enrolled to be registered in a course. Courses can be removed from the degree programme.
Example - modelling with UML-B

In a university degree programme, students are registered on degree courses. Students must be enrolled to be registered in a course. Courses can be removed from the degree programme.
Adding an Invariant

Enrolled students must be 18.

Translation: \( \forall \text{self} \cdot ((\text{self} \in \text{enroll}) \Rightarrow (\text{age}(\text{self}) \geq 18)) \)
### UML-B Class Diagrams – Translation rules (part)

<table>
<thead>
<tr>
<th>UML-B</th>
<th>Event-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (variable instances)</td>
<td>Variable ⊆ Set</td>
</tr>
<tr>
<td>Class (fixed instances)</td>
<td>Set</td>
</tr>
<tr>
<td>Class (variable inst and has super class)</td>
<td>Variable ⊆ SuperClass</td>
</tr>
<tr>
<td>Class (fixed inst and has super class)</td>
<td>Constant ⊆ SuperClass</td>
</tr>
<tr>
<td>Attribute (card 0..n - 1..1)</td>
<td>Variable ∈ Class → Type</td>
</tr>
<tr>
<td>Attribute (card 0..n - 0..1)</td>
<td>Variable ∈ Class ↔ Type</td>
</tr>
<tr>
<td>Attribute (card 0..n - 0..n)</td>
<td>Variable ∈ Class ← Type</td>
</tr>
<tr>
<td>Etc. (try other cardinalities in UML-B)</td>
<td>Etc.</td>
</tr>
<tr>
<td>Associations</td>
<td>As Attribute but Type is another class</td>
</tr>
<tr>
<td>Class Event</td>
<td>Event(self) WHEN self ∈ Class ...</td>
</tr>
<tr>
<td>Class Constructor</td>
<td>Event(self) WHEN self ∈ SET\Class ...</td>
</tr>
<tr>
<td>Class Invariant</td>
<td>∀self.((self∈Class)⇒ Class invariant)</td>
</tr>
</tbody>
</table>
Example – Event-B produced by UML-B

```plaintext
machine m sees m_implicitContext

variables enroll // class instances
degree_course // class instances
register // attribute of enroll
age // attribute of Students

invariants
@type enroll ∈ P (Students)
@type degree_course ∈ P (degree_course_SET)
@type register ∈ enroll ↔ degree_course
@type age ∈ Students → N
@Invariant1 vself.(vself∈enroll)(age(vself) ≥ 18)

events
event INITIALISATION
  then
    @init enroll = Ø
    @init degree_course = Ø
    @init register = Ø
    @init age = Students x {Ø}
  end

event Enroll
  any self // constructed instance of class enroll
  where
    @type self ∈ Students \ enroll
    @Guard1 age(self) ≥ 18
  then
    @enroll_constructor enroll = enroll u {self}
  end

event Register
  any self // contextual instance of class enroll c
  where
    @type c ∈ degree_course
    @type self ∈ enroll
    @Guard1 self ↦ c ∈ register
  then
    @Action1 register = register u {self ↦ c}
  end
```
The ‘Implicit’ Context

Each class diagram creates an *implicit* context

- Contains the ‘basis’ of things on the class diagram
- e.g. a carrier set for the type of class instances
Context Diagrams

How can we model constants that belong to a class?

in Event-B our machine would see a Context with sets, constants, axioms

UML-B takes a similar approach

► Class Diagram (Machine) sees Context Diagram
► Similar to a Class Diagram but translates to sets, constants and axioms
► ClassType instead of Class
► Constant Attributes/Associations represent constants
► Axioms instead of Invariants
► No Events
A Context Diagram and its translation

context c

constants STUDENTS // classType instances
  Id // attribute of STUDENTS
  Name // attribute of PERSON

sets ID // ClassType
  PERSON // ClassType
  NAME // ClassType

axioms
  @type STUDENTS ∈ P (PERSON)
  @type Id ∈ STUDENTS → ID
  @type Name ∈ PERSON → NAME

end
Linking a Class to a ClassType

1. Select class
2. Click button and enter name of ClassType
3. Select ClassType in Instances combo

![Diagram showing the process of linking a class to a ClassType.](image-url)
Enumerated Types

For real enumerated types e.g.
  signal = \{proceed, warning, stop\}

also, for restricting types to an example
for model checking
Enumerated Types

CONTEXT
  Context1

SETS
  NAMES // ClassType

CONSTANTS
  Dinho    // enumeration constant
  Michael  // enumeration constant
  Colin    // enumeration constant

AXIOMS
  Dinho.type : Dinho ∈ NAMES
  Michael.type : Michael ∈ NAMES
  Colin.type : Colin ∈ NAMES
  partition enumeration of NAMES : partition(NAMES,{Dinho},{Michael},{Colin})

END
Summary

Class diagrams for class-oriented modelling
  automatically generates class structures in Event-B

Attribute and association cardinalities

Options for class instances
  variable (constructors and destructors)
  fixed

Automatically generates an ‘implicit context’

Context diagrams for class oriented modelling of sets, constants and enumerated types