From an Abstract Specification in Event-B toward an UML/OCL Model

Imen Sayar (LORIA/MIRACL) and Mohamed Tahar Bhiri (MIRACL)

In FormaliSE2014, June 3rd 2014, Hyderabad, India
Plan

1. Problematic
2. Hybrid approach of software development
3. Event-B and EM-OCL
4. Case study in Event-B : SCEH
5. From Event-B to UML/OCL
6. Conclusion and future works
Problematic
Classical Approach

Oversights and ambiguity

Errors

Errors

Errors

Errors

Requirement s Document

Specification

Design

Coding

Integration

Software not always correct (bugs)

Correct Software and Systems? Complexity?
Software Engineering:

• Formal methods:  
  - Refinement
  - Proofs
  - Logic and set theory

• Test technics:  
  - Functional Test
  - Structurel Test

• Hoare Logic  
  Precondition and postcondition

• Model-checking, ...
Formal development process

Formal process in software development encounters some difficulties as:

- Exclusion of non-expert actor in formal methods → Validation activity
- Maintenance → Reviewing of formal models
- Choice of the refinement strategy
- Difficulties related to the interactive proving
What about combining formal and semi-formal approaches?

- Initial Requirements Document
- Structured Requirements Document
  - Abstract Specification
  - Horizontal Refinement
  - Design
  - Coding
  - Integration

Formal Approach (Event-B)

Semi-formal Approach (UML/OCL)
Hybrid Approach of software development
Hybrid Process: phases

Requirements document in natural language

Phase 1

Structured requirements document

Phase 2

Initial abstract model in Event-B

Phase 3

Final abstract model in Event-B

Phase 4

Validated final abstract model in Event-B

Phase 5

Pivot UML/EM-OCL model

Phase 6

Initial UML/OCL model

Phase 7

Final UML/OCL model

Restructuring the requirements document

- Oversights

- Ambiguity, lack of informations

- Two separated texts (J.R Abrial):
  - Explicative text:
    - all system details
    - main reference
  - Reference text:
    - most important constraints
    - short, simple and labelled sentences written in natural language (traçabilité)

✗ Difficult task and needs an intense intervention of the developer
Construction of a UML/OCL B models

Requirements document in natural language

Phase 1
Structured requirements document

Phase 2
Initial abstract model in Event-B

Phase 3
Final abstract model in Event-B

Phase 4
Validated final abstract model in Event-B

Phase 5
Pivot UML/EM-OCL model

Phase 6
Initial UML/OCL model

Phase 7
Final UML/OCL model

Systematic translation rules (15 rules)

Refinement rules

Initial UML/OCL model

Design patterns and class libraries

Final UML/OCL model
Assessment

- **Coherent** and **validated** formal specification of the future software/system

- **Reuse** of design patterns and class libraries

- **Involvement of external actors** not necessarily experts in formal methods

- Possibility of automatic generation of **test data**

- **Bridge** between Event-B and UML/OCL: **UML/EM-OCL**
Event-B and EM-OCL
Mathematical approach

Formal models correct by construction

Refinement

Verified and validated models via proofs and animation/model-checking (ProB, AnimB, JeB,..)

Rodin platform open source (http://www.event-b.org/)
EM-OCL: Mathematical Extension of OCL

- Integration of mathematical concepts Pair, Binary Relation and Function

- Three existant uses (Bhiri et al.) :
  - Refinement in UML
  - Validation of class diagrams (invariant construction proposed by EM-OCL)
  - EM-OCL as a request language

- Other use UML/EM-OCL as pivot language between Event-B (the formal) and UML/OCL (semi-formal)
The EM-OCL library

- Augmentations related to the standard OCL library
## EM-OCL vs. Event-B

### Correspondences between Event-B set-logical language and UML/EM-OCL

<table>
<thead>
<tr>
<th>Event-B set-logical language</th>
<th>UML/EM-OCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \mapsto y )</td>
<td>( \text{Pair}(x, y) )</td>
</tr>
<tr>
<td>( A \leftrightarrow B )</td>
<td>( \text{BinaryRelation}(A, B) )</td>
</tr>
<tr>
<td>( A \rightarrow B )</td>
<td>( \text{PartialFunction}(A, B) )</td>
</tr>
<tr>
<td>( A \twoheadrightarrow B )</td>
<td>( \text{TotalFunction}(A, B) )</td>
</tr>
<tr>
<td>( A \hookrightarrow B )</td>
<td>( \text{PartialInjective}(A, B) )</td>
</tr>
<tr>
<td>( A \rightarrowtail B )</td>
<td>( \text{TotalInjective}(A, B) )</td>
</tr>
<tr>
<td>( A \twoheadrightarrowtail B )</td>
<td>( \text{TotalBijection}(A, B) )</td>
</tr>
</tbody>
</table>

### Correspondences between Event-B substitution language and UML/EM-OCL

<table>
<thead>
<tr>
<th>Event-B substitution language</th>
<th>UML/EM-OCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = y )</td>
<td>post : ( x = y )</td>
</tr>
<tr>
<td>( x : \subseteq \text{Set}_\text{Exp} )</td>
<td>post : ( \text{Set}_\text{Exp} \rightarrow \text{includes}(x) )</td>
</tr>
<tr>
<td>( x : \mid \text{Before}<em>\text{After}</em>\text{Predicate}(x) )</td>
<td>post : ( \text{Before}<em>\text{After}</em>\text{Predicate}(x) )</td>
</tr>
<tr>
<td>( x, y = E, F )</td>
<td>post : ( x = E ) \hspace{1em} \text{and} \hspace{1em} y = F )</td>
</tr>
<tr>
<td>( f(x) = E )</td>
<td>post : ( f \rightarrow \text{imageElt}(x) = E )</td>
</tr>
</tbody>
</table>
### Rule R_i

<table>
<thead>
<tr>
<th>Rule R_i</th>
<th>Label</th>
<th>Fundamental Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>EM-OCL constraints</td>
<td>Event-B and EM-OCL typing correspondences</td>
</tr>
<tr>
<td>R_2</td>
<td>Event-B OCL constraints</td>
<td>Passage of implicit guards to explicit constraints</td>
</tr>
<tr>
<td>R_3</td>
<td>Extracted post-conditions from the guards</td>
<td>Methods and attributes visibility</td>
</tr>
<tr>
<td>R_4</td>
<td>Extracted preconditions from the guards</td>
<td>Skip substitution</td>
</tr>
<tr>
<td>R_5</td>
<td>Applicable Methods/Operations</td>
<td>Constructor</td>
</tr>
<tr>
<td>R_6</td>
<td>Object attributes and invariants typing</td>
<td>Static attributes and invariants typing</td>
</tr>
<tr>
<td>R_7</td>
<td>Static Attributes</td>
<td>Object attributes and invariants typing</td>
</tr>
<tr>
<td>R_8</td>
<td>Data Types</td>
<td>Static Attributes</td>
</tr>
<tr>
<td>R_9</td>
<td>Structured requirements</td>
<td>Rule document in a natural language</td>
</tr>
<tr>
<td>R_10</td>
<td>Requirements document</td>
<td>Initial abstract model in Event-B</td>
</tr>
<tr>
<td>R_11</td>
<td>Rule R_11</td>
<td>Initial UML/OCL model</td>
</tr>
<tr>
<td>R_12</td>
<td>Rule R_12</td>
<td>Pivot UML/EM-OCL model</td>
</tr>
<tr>
<td>R_13</td>
<td>Rule R_13</td>
<td>Final abstract model in Event-B</td>
</tr>
<tr>
<td>R_14</td>
<td>Rule R_14</td>
<td>Final UML/OCL model</td>
</tr>
<tr>
<td>R_15</td>
<td>Rule R_15</td>
<td>Validated final abstract model in Event-B</td>
</tr>
</tbody>
</table>
Rule 10: Every substitution in an Event-B event is converted to a post-condition

check_out ≜
...
THEN
...
act3:cards_ad=\{c\}<cards_ad
END

EM-OCL constraints

context Hotel::check_out(...)
...
post act3:cards_ad=cards_ad@pre -> soustractionDomain(c)
An Electronic Hotel Key System (SCEH) in Event-B
The purpose of this system is to ensure the unicity of access to hotel rooms by their current clients. This is not the case of hotel with metallic key system since a previous user of the room may have duplicated the metallic key. Therefore, access to the corresponding rooms may be possible at any time by any previous client. The judicious use of an appropriate electronic key system could guarantee unicity of access to the rooms by their current clients... (From “Modeling in Event-B: System and software Engineering” of J-R Abrial)
<table>
<thead>
<tr>
<th>Reformulated constraint</th>
<th>Constraint type</th>
</tr>
</thead>
<tbody>
<tr>
<td>The access to a room is limited to the user who has booked it.</td>
<td><strong>FUN-1</strong></td>
</tr>
<tr>
<td>Each hotel room door is equipped with an electronic lock which holds an electronic key and which has a magnetic card reader.</td>
<td><strong>ENV-1</strong></td>
</tr>
<tr>
<td>A magnetic card holds two distinct electronic keys: k1 and k2</td>
<td><strong>ENV-2</strong></td>
</tr>
<tr>
<td>Hotel employees can enter in the rooms with identical cards to those of clients</td>
<td><strong>FUN-2</strong></td>
</tr>
<tr>
<td>The first access of a client to his room is followed by an update of the key stored in the lock</td>
<td><strong>FUN-3</strong></td>
</tr>
<tr>
<td>Access to rooms is controlled by magnetic cards</td>
<td><strong>FUN-4</strong></td>
</tr>
</tbody>
</table>
Adopted Refinement Strategy

Hotel_M0 sees Hotel_Ctx0

Hotel_M1 sees Hotel_Ctx1

Hotel_M2 sees Hotel_Ctx1

Hotel_M3 sees Hotel_Ctx3

Hotel_M0 refines Hotel_M1

Hotel_M1 refines Hotel_M2

Hotel_M2 refines Hotel_M3

Hotel_Ctx0 extends Hotel_Ctx1

Hotel_Ctx1 extends Hotel_Ctx3

Level 1

Level 2

Level 3

Level 4
Formal Event-B models

Initial Abstract Model

MACHINE Hotel_M0
  SEES Hotel_Ctx0
  VARIABLES
    owns
  INVARIANTS
    inv0_1 : owns ∈ ROOM GUEST
  EVENTS
    INITIALISATION
      STATUS
        ordinary
      BEGIN
        act1: owns = Ø
      END
    check_in
      STATUS
        ordinary
      BEGIN
        act1: owns = owns \ {r↦g}
      END
    check_out
      STATUS
        ordinary
      BEGIN
        act1: owns = Ø
      END
END
SCEH : From Event-B models toward UML/OCL class diagram
Rule 1

Rule 2

Rule 3

Rule 4

Rule 7

Rule 8
context Hotel:: check_in3(g: GUEST, r: ROOM, c: Pair(Key, Key), a: ADMINISTRATOR)

pre grd2: Room->includes(r)
pread grd3: (owns->domain())->excludes(r)
pread grd4: Card->includes(c)
pread grd5: (first->imageElt(c)) = (currk->imageElt(r))
pread grd6: issued->excludes(second->imageElt(c))
pread grd7: (currk->range())->excludes(second->imageElt(c))
pread grd8: (cards->domain())->excludes(c)
pread grd9: roomk->imageElt(r) = (currk->imageElt(r))
pread grd10: Administrator->includes(a)
pread grd11: owns_adm->imageElt(r)=a
pre grd12: (cards_adm->domain())->excludes(c)

post act1: owns->imageElt(r)=g
post act2: issued=issued@pre->including(second->imageElt(c))
pread act3: cards->imageElt(c)=g
post act4: currk->imageElt(r)=second->imageElt(c)
pread act5: cards_adm->imageElt(c)=a
Conclusion

- **Hybrid** development process: **formal** (Event-B) and **semi-formal** (UML/EM-OCL and UML/OCL)
- Essential software qualities: correctness, reusability, scalability...
- **Various actors**: Event-B specifiers, OO designers, OO implementers and testers
- **Translation rules** between Event-B and UML/EM-OCL
- **Refinement rules** of UML/EM-OCL by UML/OCL models
- Case study of electronic hotel key system
Future works

- identity (id) and Cartesian product

- Properties related to vivacity

- Automate the Event-B transition to UML/EM-OCL

- Automate the transition from UML/EM-OCL to UML/OCL
THANK YOU