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

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

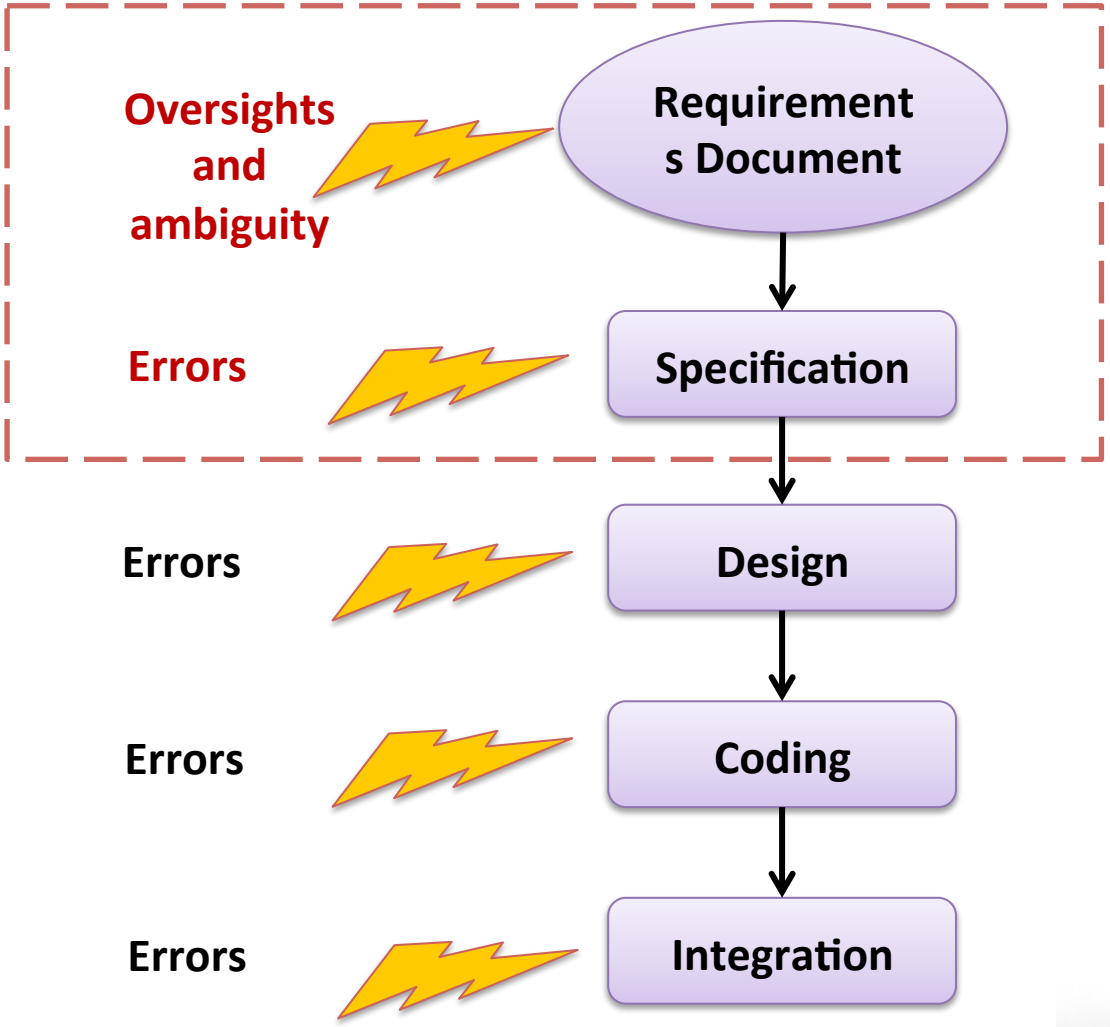
In FormaliSE2014, June 3rd 2014, Hyderabad, India

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



Software not always correct (bugs)

Correct Software and Systems ? Complexity ?

Problematic

Software Engineering:

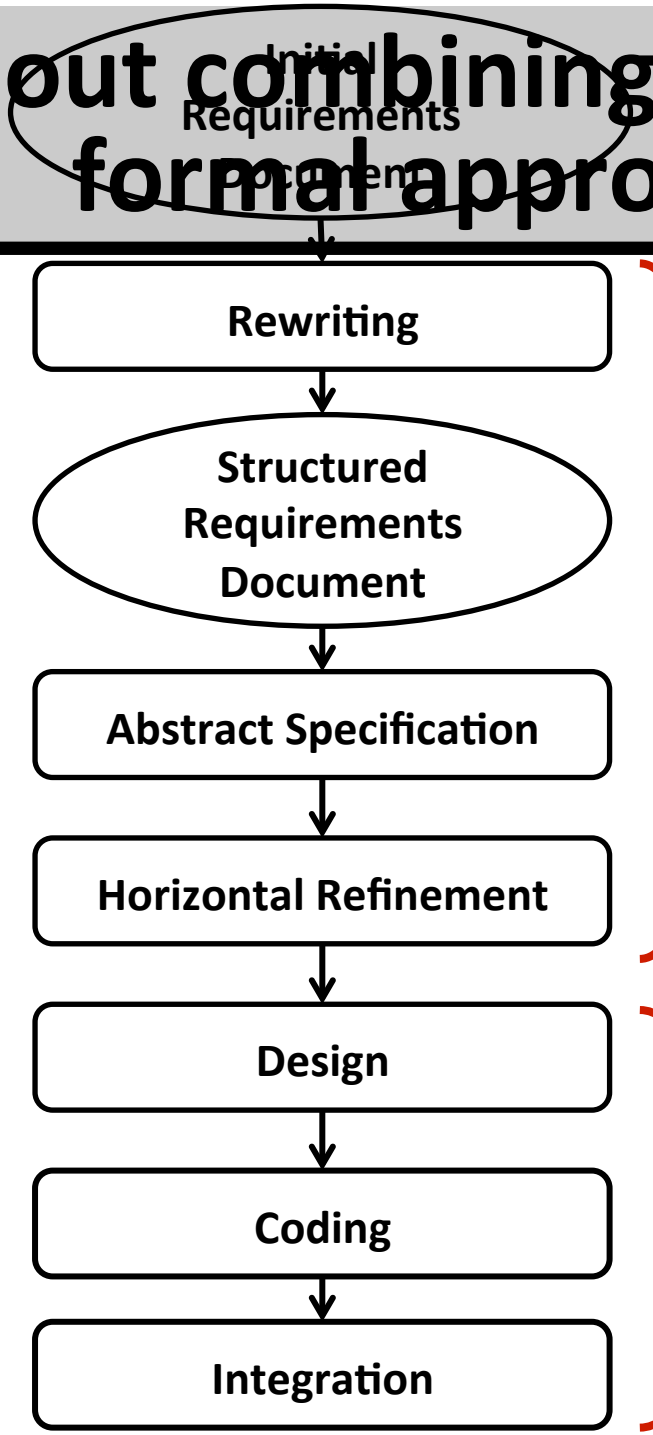
- **Formal methods:**
  - Refinement
  - Proofs
  - Logic and set theory
- **Test technics:**
  - Functional Test
  - Structurel Test
- **Hoare Logic** → Precondition and postcondition
- **Model-checking, …**

Formal developement process

Formal process in software developement encounters some difficulties as:

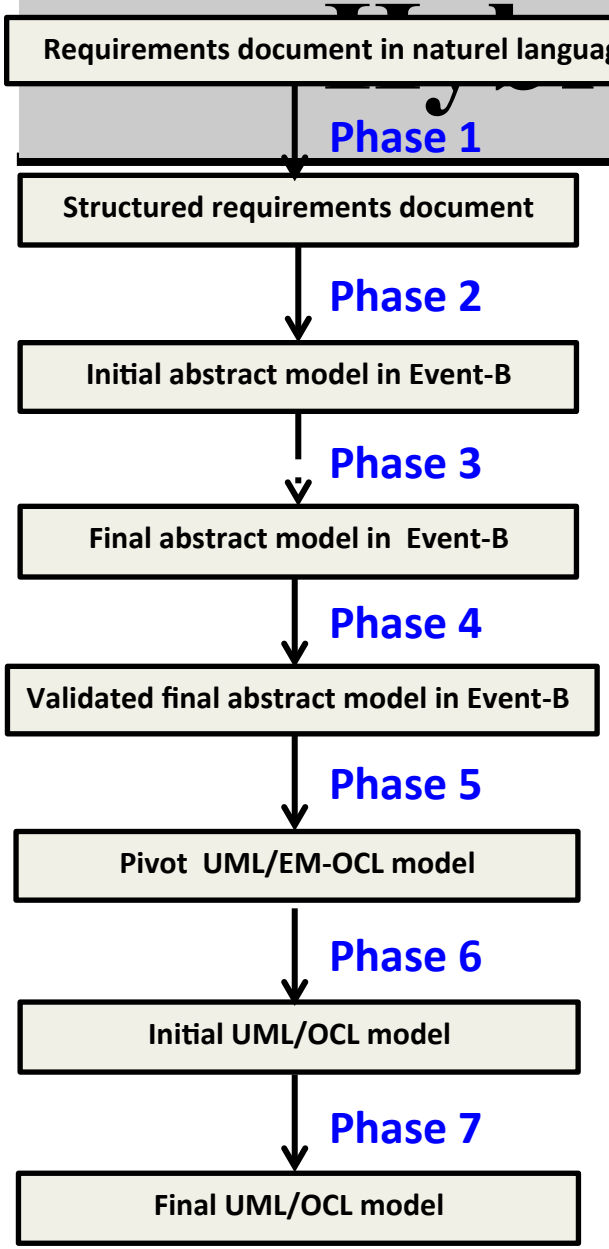
- **Exclusion of non-expert actor** in formal methods ⟶ **Validation** activity
- Maintenance ⟶ **Reviewing** of formal models
- Choice of the **refinement strategy**
- Difficulties related to the **interactive proving**

What about combining formal and semi-formal approaches ?



Hybrid Approach of software development

Hybrid Process: phases

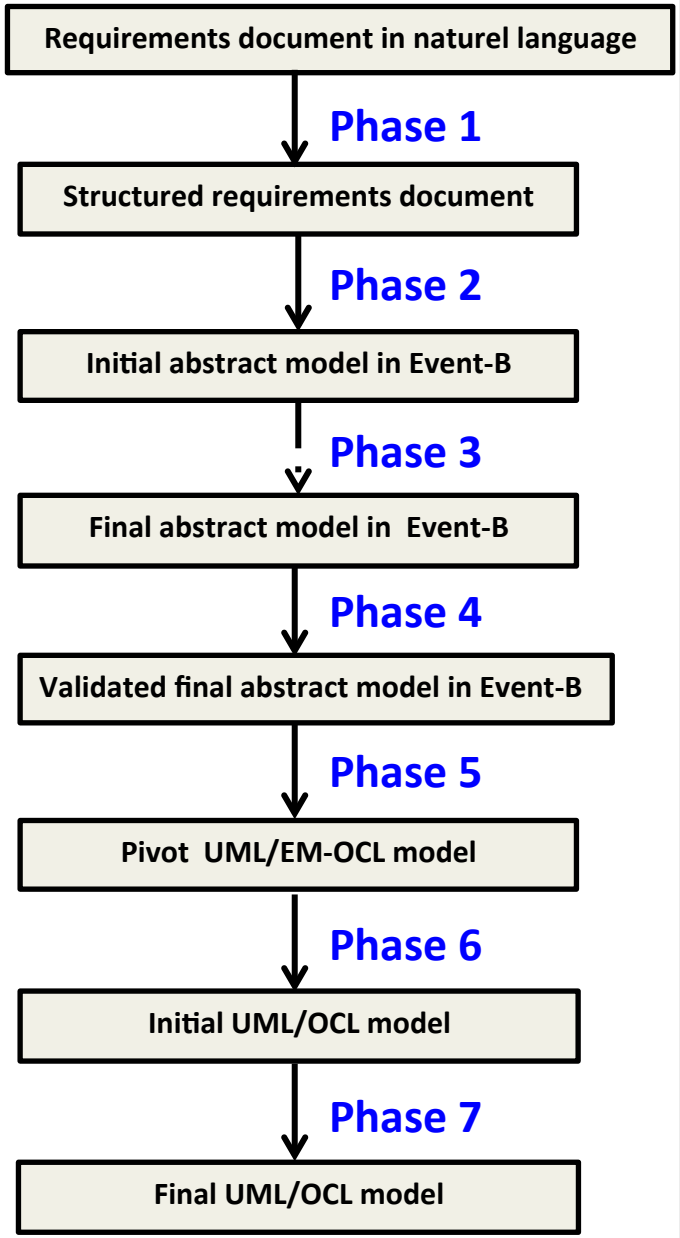


Restructuring the requirements document

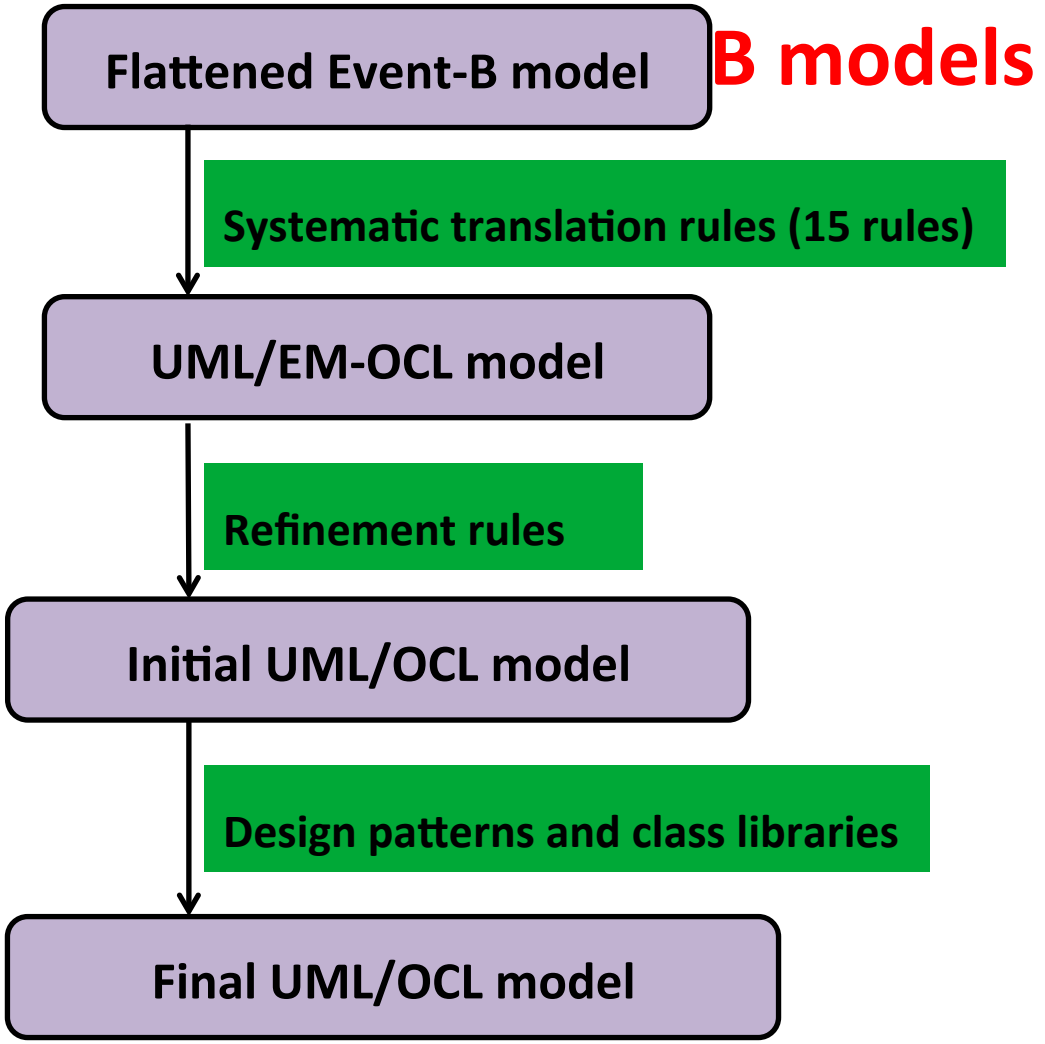
- Oversights
- Ambiguity, lack of informations

- 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 (**traçability**)

✖ Difficult task and needs an intense intervention of the developer



Construction of a UML/OCL B models



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

Event-B

- 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

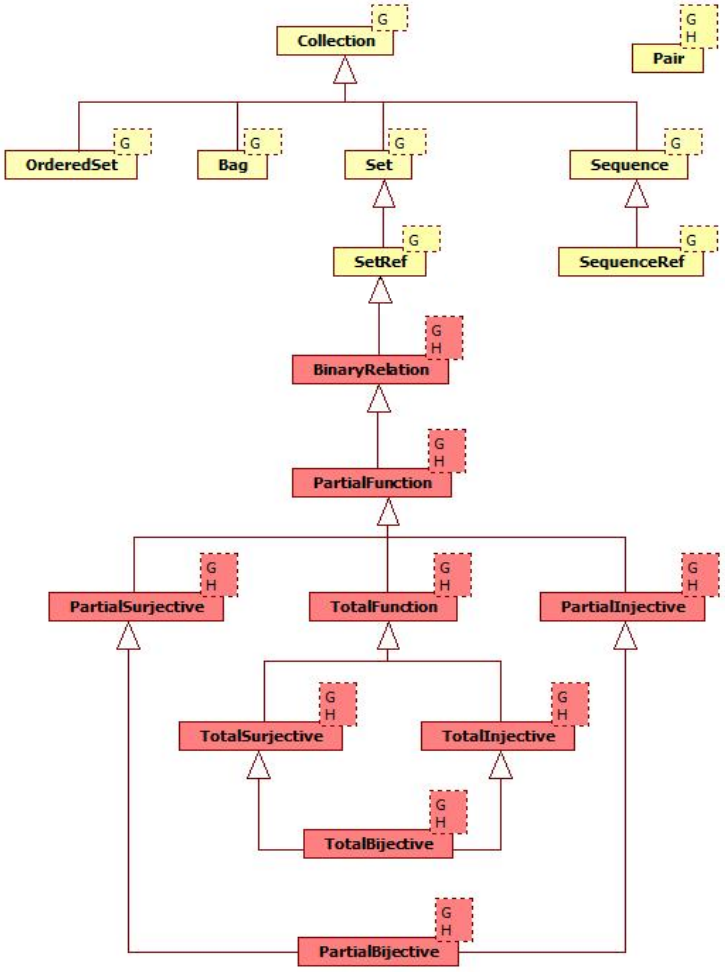
- Integration of mathematical concepts **Pair, Binary Relation** and **Function**

- 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



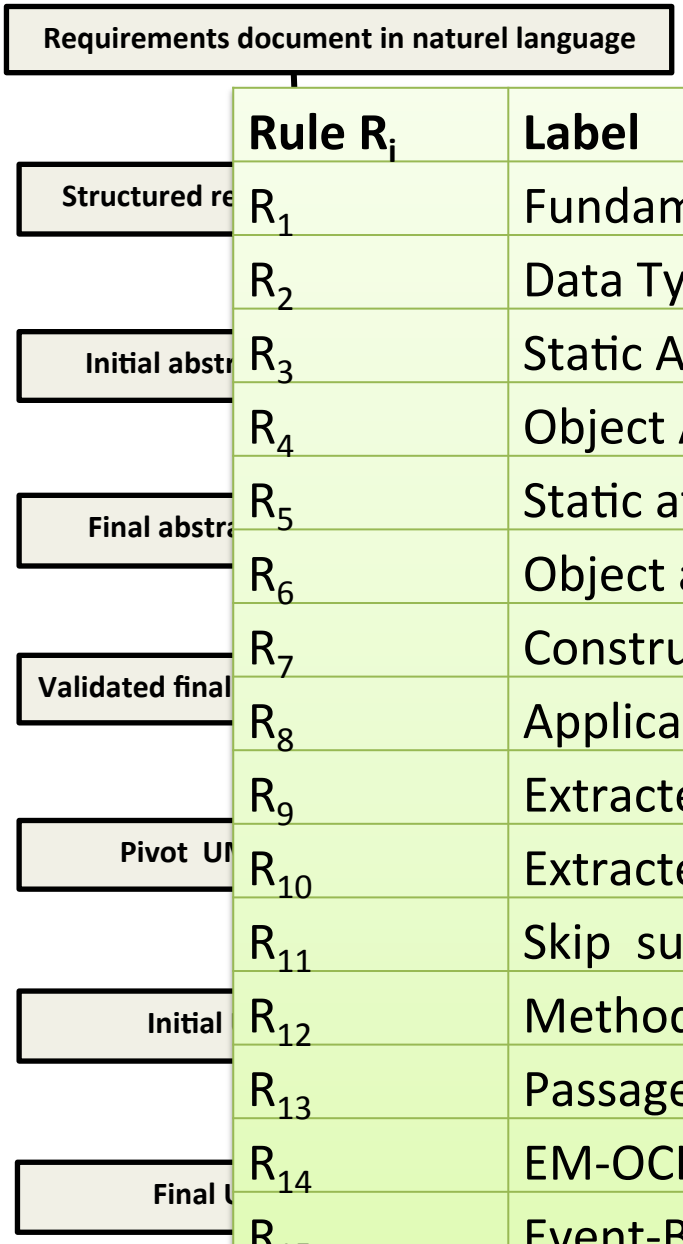
EM-OCL vs. Event-B

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

Event-B set-logical language	**UML/EM-OCL**
$x \mapsto y$	Pair(x, y)
$A \leftrightarrow B$	BinaryRelation(A, B)
$A \nrightarrow B$	PartialFunction(A, B)
$A \rightarrow B$	TotalFunction(A, B)
$A \rightarrowtail B$	PartialInjective(A, B)

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

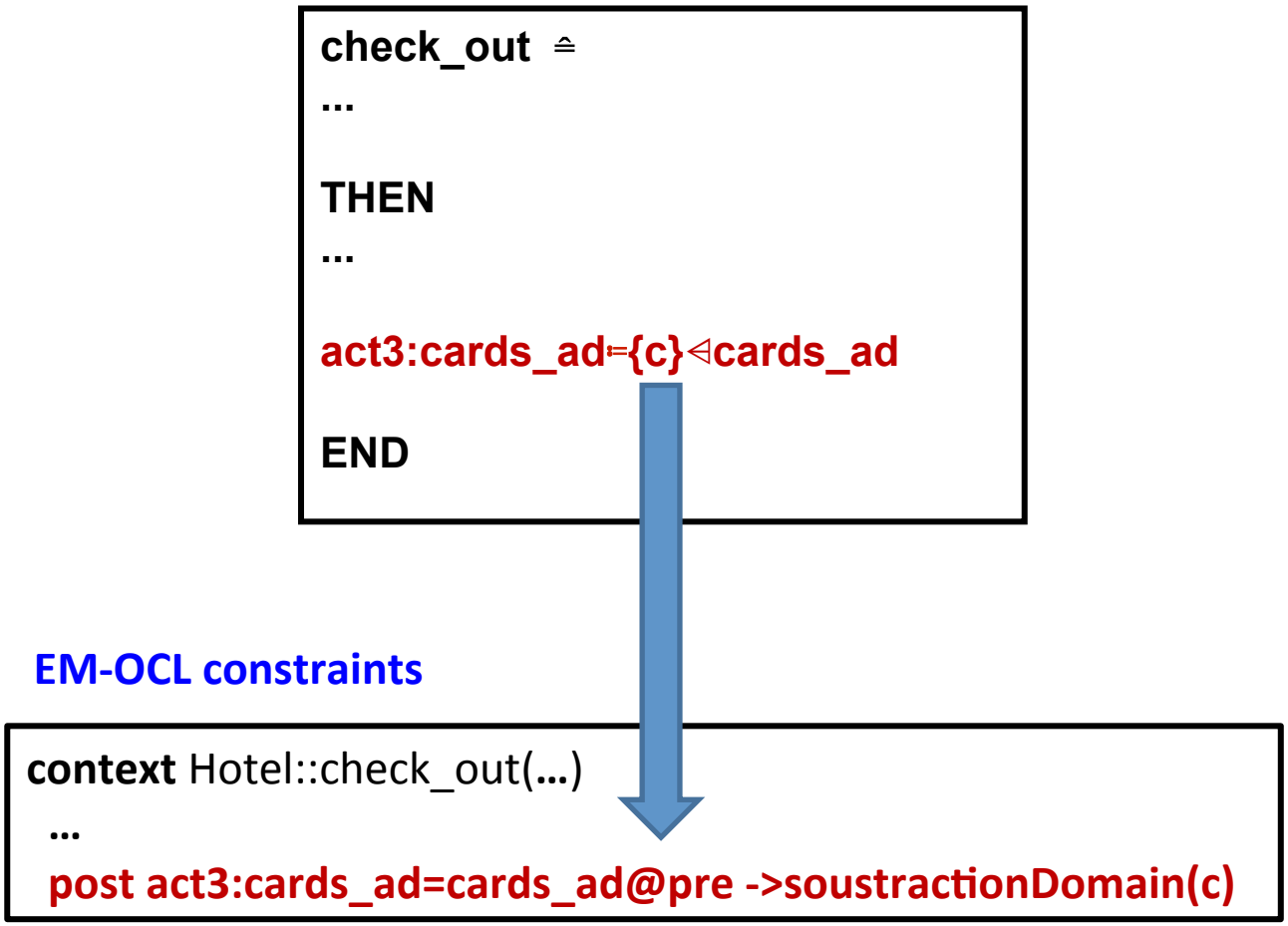
Event-B substitution language	**UML/EM-OCL**
$x := y$	post : x=y
$x :\in Set\_Exp$	post :Set_Exp-> includes(x)
$x :\mid Before\_After\_Predicate(x)$	post : Before_After_Predicate(x)
$x,y := E,F$	post : x= E and y= F
$f(x) := E$	post : f->imageElt(x)= E



Rule R_i	Label
R_1	Fundamental Class
R_2	Data Types
R_3	Static Attributes
R_4	Object Attributes
R_5	Static attributes and invariants typing
R_6	Object attributes and invariants typing
R_7	Constructor
R_8	Applicable Methods/Operations
R_9	Extracted preconditions from the guards
R_{10}	Extracted post-conditions from substitutions
R_{11}	Skip substitution
R_{12}	Methods and attributes visibility
R_{13}	Passage of implicit guards to explicit constraints
R_{14}	EM-OCL constraints
R_{15}	Event-B and EM-OCL typing correspondences

Illustration

Rule10: Every **substitution** in an Event-B **event** is converted to a ***post-condition***



An Electronic Hotel Key System (SCEH) in Event-B

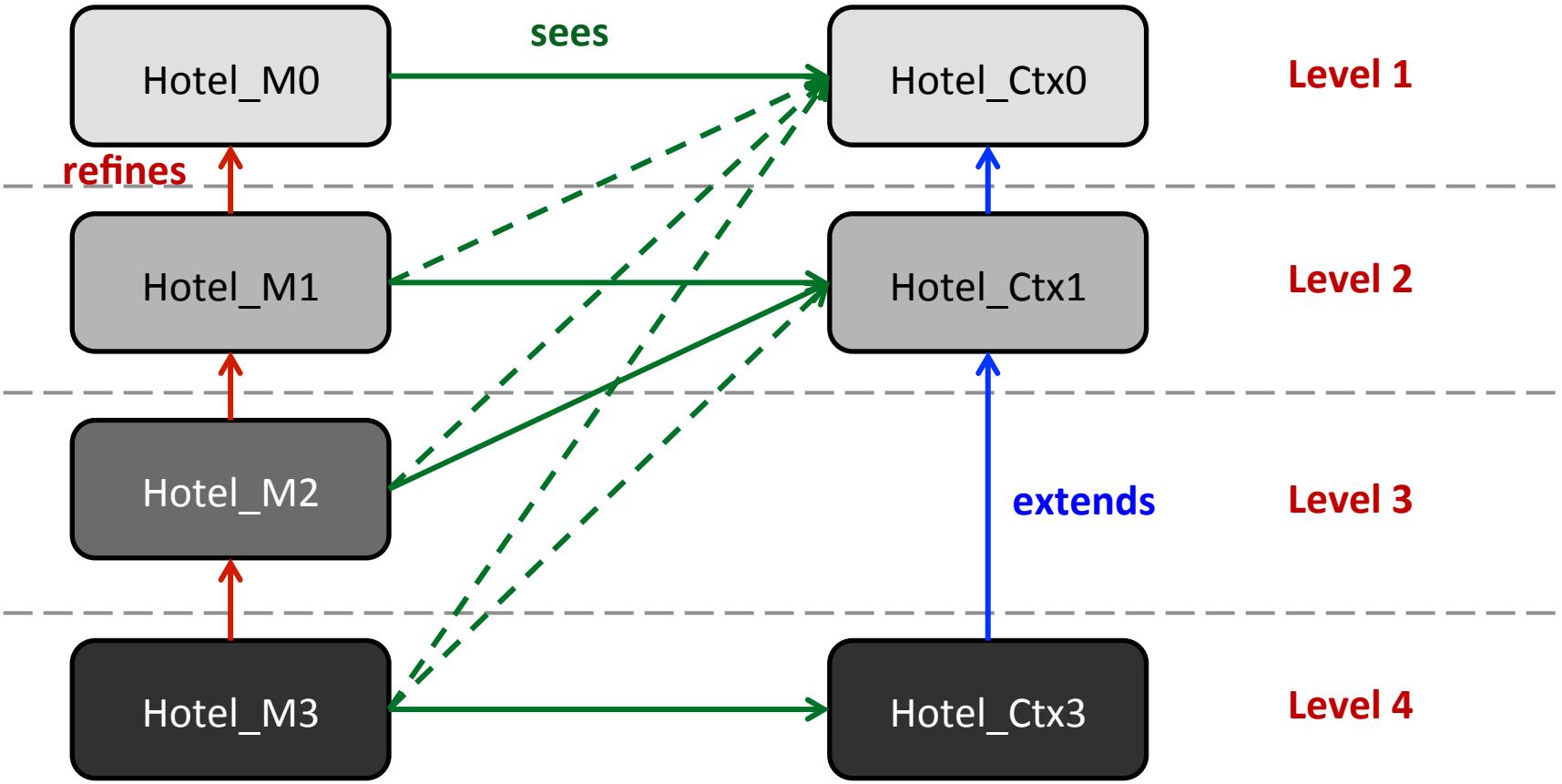
SCEH: informal presentation

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)

Structured Requirements Document: Referential Text

Reformulated constraint	Constraint type
The access to a room is limited to the user who has booked it.	***FUN-1***
Each hotel room door is equipped with an electronic lock which holds an electronic key and which has a magnetic card reader.	***ENV-1***
A magnetic card holds two distinct electronic keys: k1 and k2	***ENV-2***
Hotel employees can enter in the rooms with identical cards to those of clients	***FUN-2***
The first access of a client to his room is followed by an update of the key stored in the lock	***FUN-3***
Access to rooms is controlled by magnetic cards	***FUN-4***

Adopted Refinement Strategy



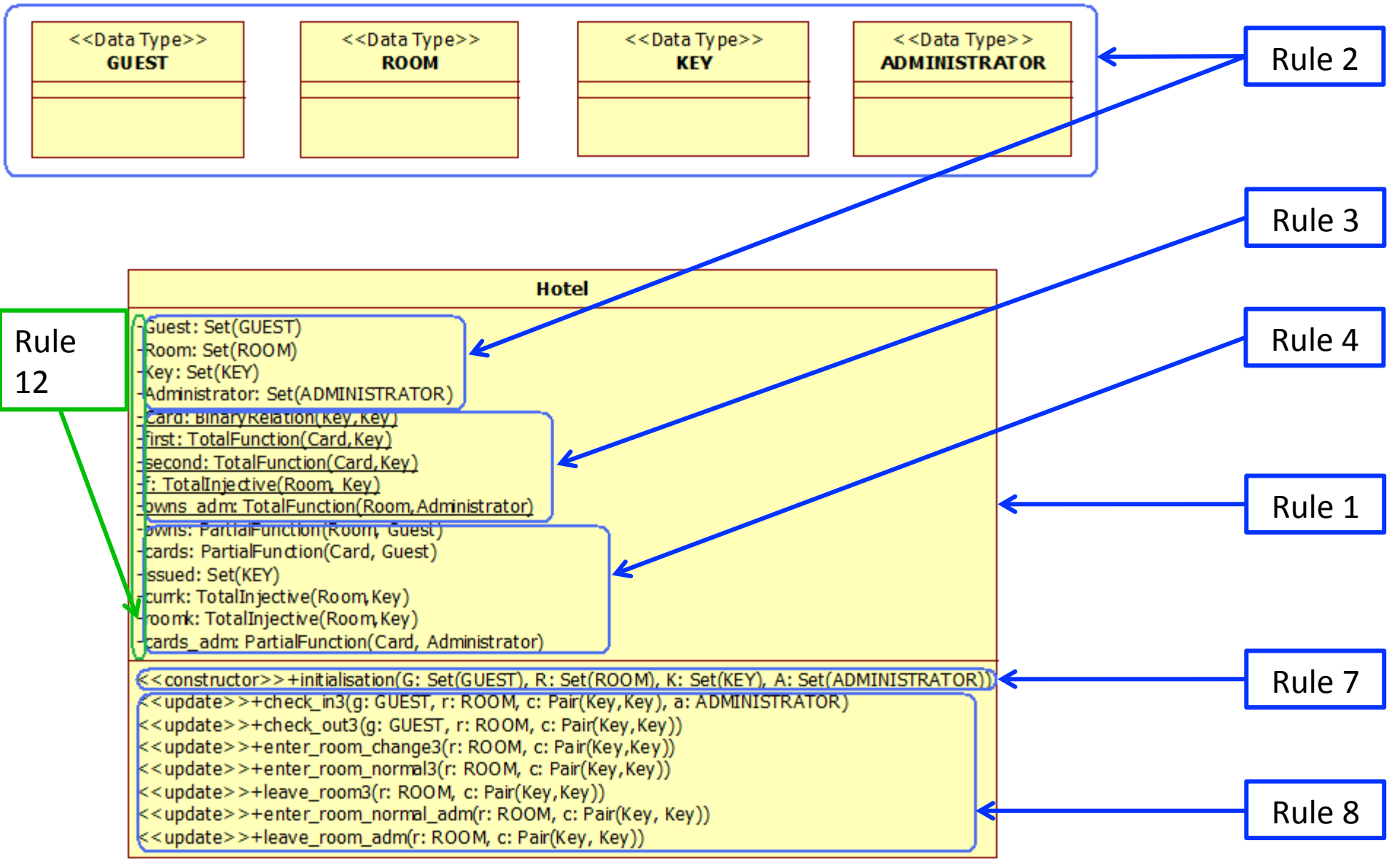
Formal Event-B models

Initial Abstract Model

```
CONTEXT ♪
    Hotel_Ctx0 ♪
SETS♪
    GUEST♪
    ROOM♪
END♪
```

```
MACHINE ♪
    Hotel_M0♪
SEES ♪
    Hotel_Ctx0♪
VARIABLES♪
    owns♪
INVARIANTS♪
    inv0_1    :
owns∈ROOM⇸GUEST    ♪
EVENTS♪
  INITIALISATION  ≙♪
    STATUS♪
      ordinary♪
    BEGIN♪
      act1: owns≔∅      ♪
    END♪
  check_in  ≙♪
    STATUS♪
      ordinary♪
    ANY♪
      g♪
      r
    WHERE♪
      grd1: g∈GUEST♪
      grd2: r∈ROOM♪
      grd3  : r∉dom(owns)♪
    THEN♪
      act1: owns(r)≔♪
    END♪
 ♪
  check_out  ≙♪
    STATUS♪
      ordinary♪
    ANY♪
      g♪
      r♪
    WHERE♪
      grd1  : r↦g∈owns♪
    THEN♪
      act1: owns≔owns∖{r↦g}  ♪
    END♪
 ♪
END♪
```

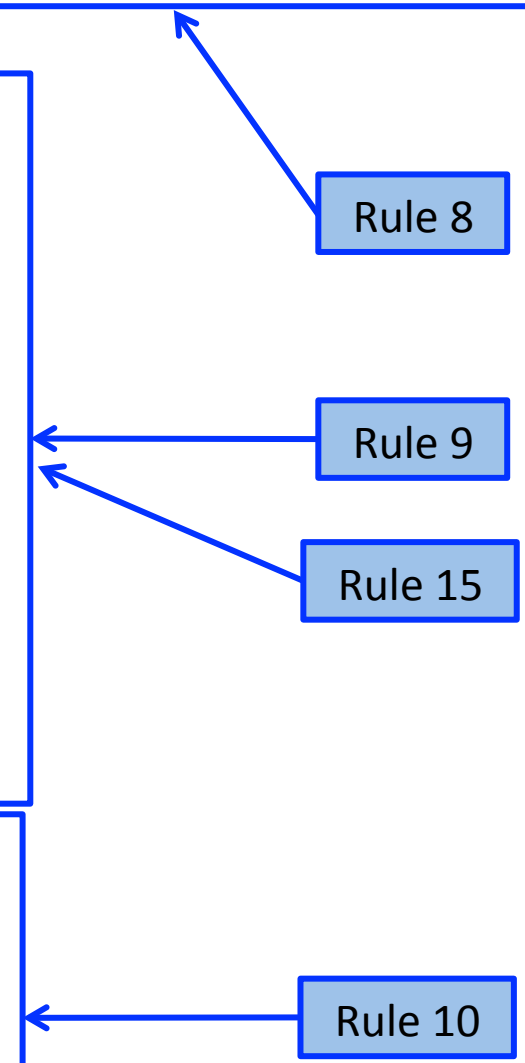
SCEH : From Event-B models toward UML/OCL class diagram

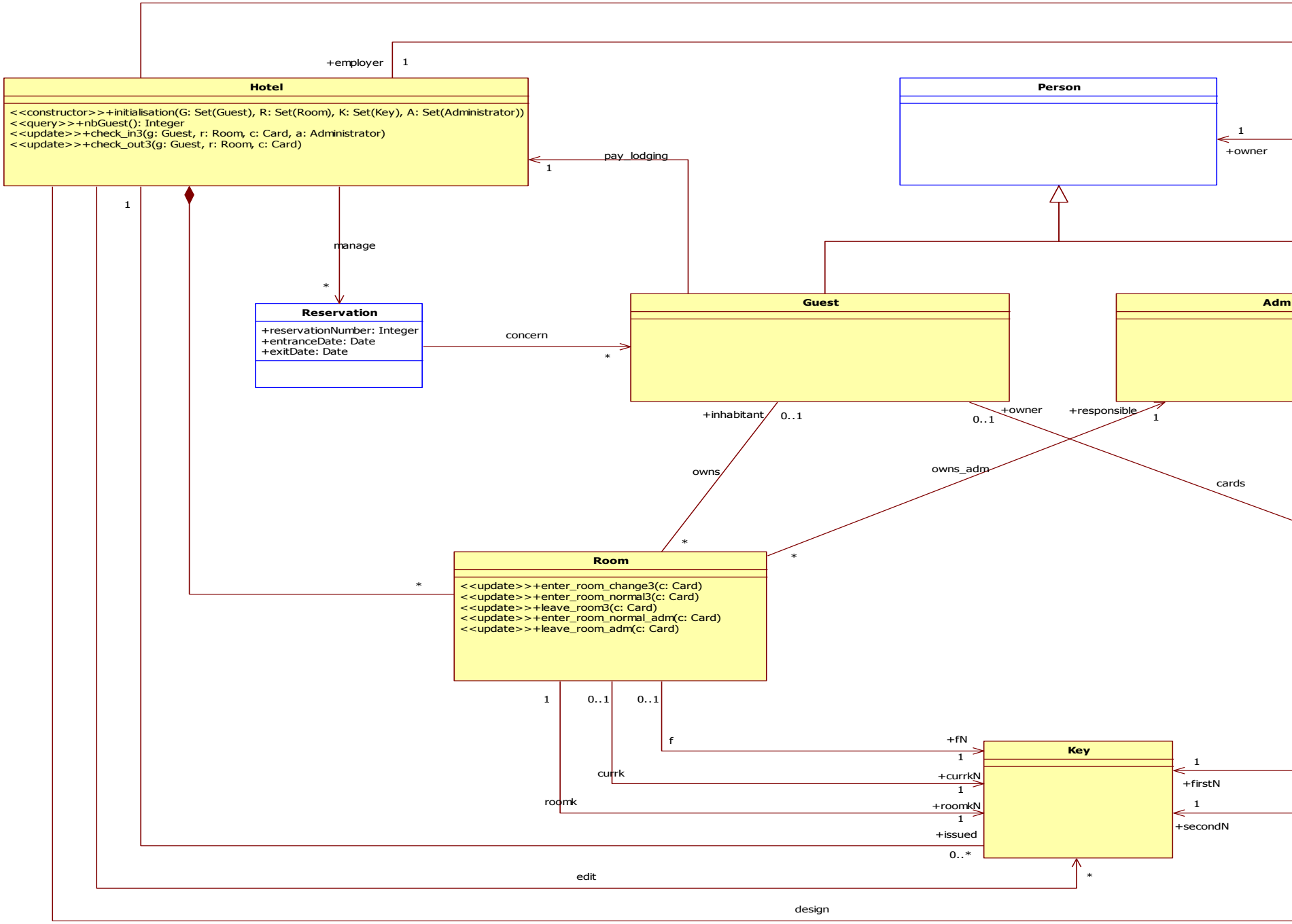

<<Data Type>>
GUEST
<<Data Type>>
ROOM
<<Data Type>>
KEY
<<Data Type>>
ADMINISTRATOR
Hotel
-Guest: Set(GUEST)
-Room: Set(ROOM)
-Key: Set(KEY)
-Administrator: Set(ADMINISTRATOR)
-Card: BinaryRelation(Key,Key)
-first: TotalFunction(Card,Key)
-second: TotalFunction(Card,Key)
-f: TotalInjective(Room, Key)
-owns_adm: TotalFunction(Room,Administrator)
-owns: PartialFunction(Room, Guest)
-cards: PartialFunction(Card, Guest)
-issued: Set(KEY)
-currk: TotalInjective(Room,Key)
-roomk: TotalInjective(Room,Key)
-cards_adm: PartialFunction(Card, Administrator)
<<constructor>>+initialisation(G: Set(GUEST), R: Set(ROOM), K: Set(KEY), A: Set(ADMINISTRATOR))
<<update>>+check_in3(g: GUEST, r: ROOM, c: Pair(Key,Key), a: ADMINISTRATOR)
<<update>>+check_out3(g: GUEST, r: ROOM, c: Pair(Key,Key))
<<update>>+enter_room_change3(r: ROOM, c: Pair(Key,Key))
<<update>>+enter_room_normal3(r: ROOM, c: Pair(Key,Key))
<<update>>+leave_room3(r: ROOM, c: Pair(Key,Key))
<<update>>+enter_room_normal_adm(r: ROOM, c: Pair(Key, Key))
<<update>>+leave_room_adm(r: ROOM, c: Pair(Key, Key))
Rule 2
Rule 3
Rule 4
Rule 1
Rule 7
Rule 8
Rule 12

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

pre grd2: Room->includes(r)
pre grd3: (owns->domain())->excludes(r)
pre grd4: Card->includes(c)
pre grd5: (first->imageElt(c))=(currk->imageElt(r))
pre grd6: issued->excludes(second->imageElt(c))
pre grd7: (currk->range())->excludes(second->imageElt(c))
pre grd8: (cards->domain())->excludes(c)
pre grd9: roomk->imageElt(r)=(currk->imageElt(r))
pre grd10: Administrator->includes(a)
pre 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))
post act3: cards->imageElt(c)=g
post act4: currk->imageElt(r)=second->imageElt(c)
post act5: cards_adm->imageElt(c)=a




Hotel
<<constructor>>+initialisation(G: Set(Guest), R: Set(Room), K: Set(Key), A: Set(Administrator))
<<query>>+nbGuest(): Integer
<<update>>+check_in3(g: Guest, r: Room, c: Card, a: Administrator)
<<update>>+check_out3(g: Guest, r: Room, c: Card)
+employer
1
Person
+owner
1
pay_lodging
1
1
manage
*
Reservation
+reservationNumber: Integer
+entranceDate: Date
+exitDate: Date
concern
*
Guest
Adm
+inhabitant
0..1
owns
*
+owner
0..1
+responsible
1
owns_adm
cards
*
Room
<<update>>+enter_room_change3(c: Card)
<<update>>+enter_room_normal3(c: Card)
<<update>>+leave_room3(c: Card)
<<update>>+enter_room_normal_adm(c: Card)
<<update>>+leave_room_adm(c: Card)
*
1
0..1
0..1
f
+fN
1
Key
currk
+currkN
1
roomk
+roomkN
1
+issued
0..*
1
+firstN
1
+secondN
edit
*
design

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