Modelling, Refining, and Proving with Event-B

Jean-Raymond Abrial

Outline

- General concepts and comments

- An illustrating example

- A demo with the Rodin Platform

- Slides can be distributed
- Text of example can be distributed

An Ambition in System and Software Developments

- Being correct by construction

- Some simple ingredients:

  1. Informal (but precise) Requirements
  2. Modeling vs. programming
  3. Refining
  4. Proving

1. Requirements

Purpose of the Informal Requirement Document

- Contains the properties of the future system

- Allowing us to judge eventually that the final product is correct

- Made of short labeled “fragments” (traceability)

- Should be easy to read (different font) and easy to extract (boxed)

Place of the Requirement Document in Development Cycle

1. Feasibility Study
2. Requirement Document
3. Technical Specification
4. Design

4. Coding
5. Test
6. Documentation
7. Maintenance

Difficulties and weak point

- Importance of this document (due to its position in the life cycle)

- Obtaining a good requirement document is not easy:

  - missing points

  - too specific (over-specified)

- Requirement document are usually difficult to exploit

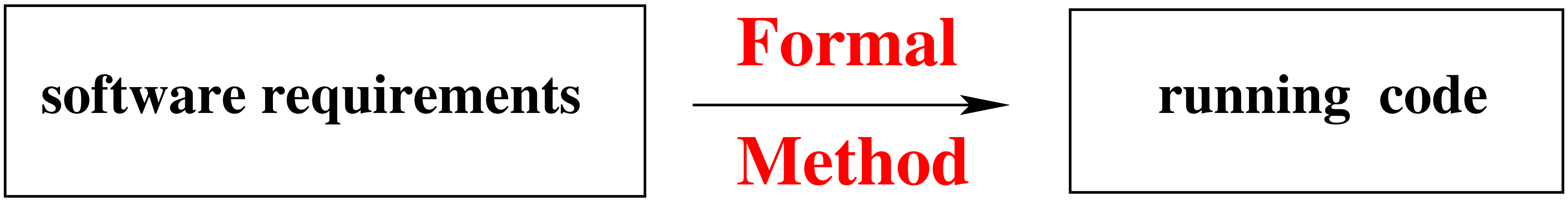
Some Structuring Rules

- Two separate texts in the same document:
  - explanatory text: the why
  - reference text: the what
- Embedding the reference text within the explanation text
- The reference text eventually becomes the official document
- Must be signed by concerned parties

2. Modeling vs. Programming

Purpose of Formal Methods

- Helping people in doing the following transformation:



- It does not seem to be different from ordinary programming

- A formal method is a systematic approach

- It is used to determine whether a program has certain properties

- Different kinds of formal methods (according to this definition)
  - Type checking
  - Static analysis
  - Model checking
  - Theorem proving

Theorem Proving: Model Construction

- This is the approach developed here

- It concentrates on the construction of models by refinements

- The properties to be proved are parts of the models

- The most refined model is automatically translated into a program

How About Other Engineering Disciplines

- Some mature engineering disciplines:
  - Avionics,
  - Civil engineering,
  - Mechanical engineering,
  - Train systems,
  - Ship building,
  - . . .

- Are there any equivalent approaches to Formal Methods with Proofs?

- Yes, BLUE PRINTS

More Precise Definition of Formal Methods Used Here

- Formal methods are techniques for building and studying blue prints

- These blue prints are ADAPTED TO OUR DISCIPLINE

- Our discipline is the design of hardware and software SYSTEMS

- Such blue prints are now called formal models

Conventions for Model Writing and Reasoning

- Models allow to reason about a FUTURE system

- The basis is lacking (hence you cannot “execute” all models)

- Using pre-defined conventions in order to facilitate reasoning:
    - Classical Logic (Predicate Calculus)
    - Basic Set Theory (sets, relations and functions)

Characterizing the Systems we Intend to Build

- These systems operate in a discrete fashion

- Their dynamical behavior can be abstracted by:

    - A succession of steady states (enriched by invariants)

    - Intermixed with sudden jumps (events)

- Usually such systems never halt

- They are called DISCRETE TRANSITION SYSTMS

3. Refinement

Refinement

- Refinement allows us to build models gradually

- We shall build an ordered sequence of more precise models

- A useful analogy: looking through a microscope

- Spatial as well as temporal extensions

- Data refinement

4. Proving

Reasoning about Discrete Transition Systems

- Test reasoning (a vast majority): VERIFICATION

- Blue Print reasoning (a very few): CORRECT CONSTRUCTION

Test Reasoning: VERIFICATION

- Based on laboratory execution

- Obvious incompleteness

- The oracle is usually missing

- Properties to be checked are chosen a posteriori

- Re-adapting and re-shaping after testing

- Reveals an immature technology

Blue Print Reasoning: CORRECT CONSTRUCTION

- Based on a formal model: the “blue print”
- Gradually describing the system with the needed precision
- Relevant Properties are chosen a priori
- Serious thinking made on the model, not on the final system
- Reasoning is validated by proofs
- Reveals a mature technology

Blue Print Reasoning: Outcome of Proving

- The proof succeeds

- The proof fails but refutes the statement to prove

    - the model is erroneous: it has to be modified

- The proof fails but is probably provable

    - the model is badly structured: it has to be reorganized

- The proof fails and is probably not provable nor refutable

    - the model is too poor: it has to be enriched

An Illustrating Example

Purpose of the Example

- Illustrating the previous points:

1. Informal (but precise) Requirements

2. Modeling

3. Refining

4. Proving

Informal Requirements: Explanations

- We want to build a business protocol

- A seller S wants to order a product from a warehouse clerk C.

- The seller may reserve some products before making a final choice

- S and C communicate by means of messages

A Business Protocol: Abstract Informal Requirements

This protocol involves a seller S and a warehouse with products	ENV-1

S may reserve products (one at a time)	FUN-1

S may delete the reservation of products (one at a time)	FUN-2

At the end, S may order one of the reserved products	FUN-3

A Business Protocol: Concrete Informal Requirements (1)

The protocol also involves a warehouse clerk C	ENV-2

S and C communicate by means of messages	ENV-3

S can make a reservation by sending a message to C.	FUN-4

C always confirms a reservation by sending a message to S.	FUN-5

A Business Protocol: Concrete Informal Requirements (2)

S can delete a reserved product by sending a message to C.	FUN-6

S can order a reserved product by sending a message to C.	FUN-7

Order, deletion or reservation messages cannot be sent by S beteween a reservation message and its confirmation.	FUN-8

The ordered product must be the same for both partners	FUN-9

A Business Protocol: Concrete Informal Requirements (3)

Messages can be reordered before being treated by C	ENV-4

Messages are never lost	ENV-5

At the end, all pending messages must be treated by C	ENV-6

1. Formalising the overall purpose of the protocol: FUN-3
2. The Seller S is alone in the Warehouse: FUN-1 and FUN-2
3. Introducing the Warehouse Clerk C: other requirements
4. Technical refinement (to make things implementable)

Initial Model: the State

- We just formalise what the seller can eventually do: order a product.

- First, we define a carrier set PRD: the product information.

sets: PRD

- A single variable S_ORD denoting the set of ordered product.

variables: S_ORD

- S_ORD is at most a singleton set

inv0_1: $S\_ORD \subseteq PRD$

inv0_2: $S\_ORD \neq \varnothing \;\Rightarrow\; \exists x \cdot S\_ORD = \{x\}$

Initial Model: the Events

- The INIT event making S_ORD empty at the beginning

- The Order event making S_ORD a singleton set

INIT
$$S\_ORD := \varnothing$$

Order
any p **where**
$p \in PRD$
$S\_ORD = \varnothing$
then
$S\_ORD := \{p\}$
end

At the end, S may order one of the reserved products	FUN-3

-We define two more variables:

- S_USD is the set of used products (already reserved in the past)

- S_RES is the set of reserved products (candidates for ordering)

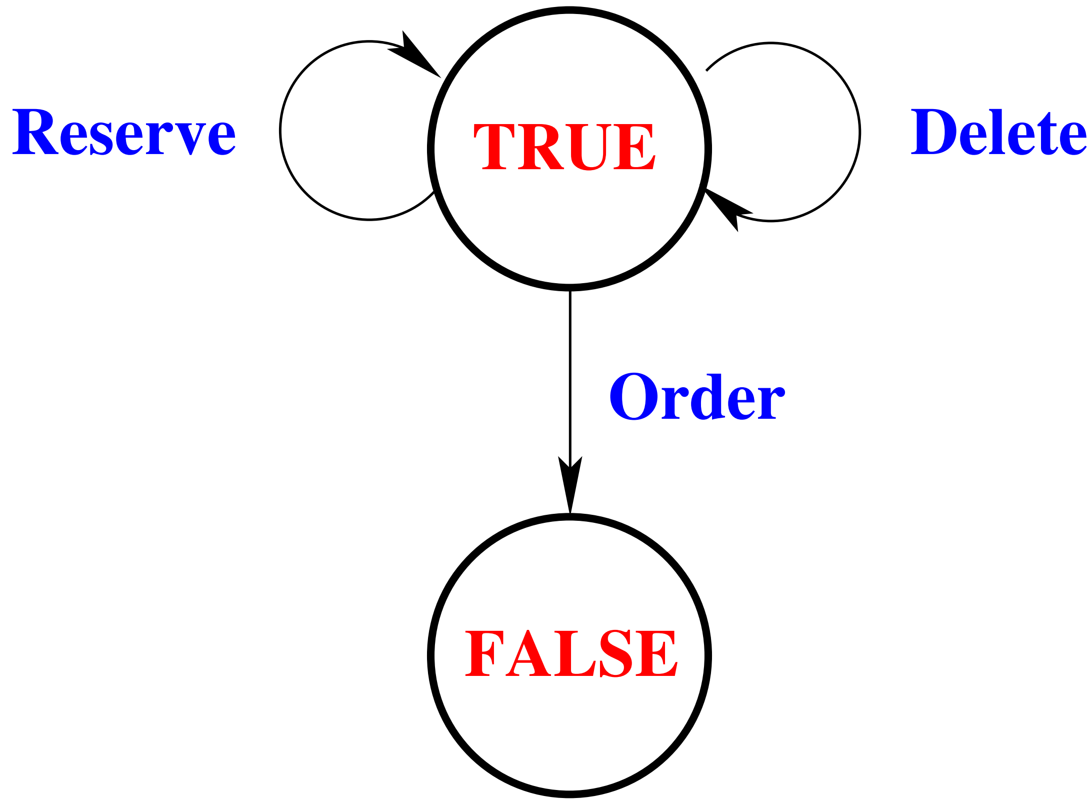
First Refinement: Invariants

inv1_1: $S\_USD \subseteq PRD$

inv1_2: $S\_RES \subseteq S\_USD$

inv1_3: $S\_ORD \subseteq S\_RES$

First Refinement: Events (1)



TRUE **means protocol active**

FALSE **means protocol terminated**

First Refinement: Events (2)

Initially: The protocol is made active and no products are used, reserved, or ordered

Reserve: When protocol is active, choose a new product (not used) and make it used and reserved

Delete: When protocol is active, choose a reserved product and make it not reserved

Order: When protocol is active, choose a reserved product and make it ordered. Make protocol inactive

INIT

$S\_USD := \varnothing$

$S\_RES := \varnothing$

$S\_ORD := \varnothing$

Order

any p **where**

$p \in S\_RES$

$S\_ORD = \varnothing$

then

$S\_ORD := \{p\}$

end

Adding New Eevents

- The protocol is active while S_ORD is empty

Reserve
any p **where**
$p \notin S\_USD$
$S\_ORD = \varnothing$
then
$S\_USD := S\_USD \cup \{p\}$
$S\_RES := S\_RES \cup \{p\}$
end

Delete
any p **where**
$p \in S\_RES$
$S\_ORD = \varnothing$
then
$S\_RES := S\_RES \setminus \{p\}$
end

First Refinement: Proofs

- Invariant preservation by the events requires 7 proofs

- All proved automatically by the prover of the Rodin Platform.

Taking Account of Requirements FUN_1, FUN_2, and FUN_3

- Event Reserve

During the protocol, S may reserve a product	FUN-1

- Event Delete

During the protocol, S may delete the reservation of a product	FUN-2

- Event Order

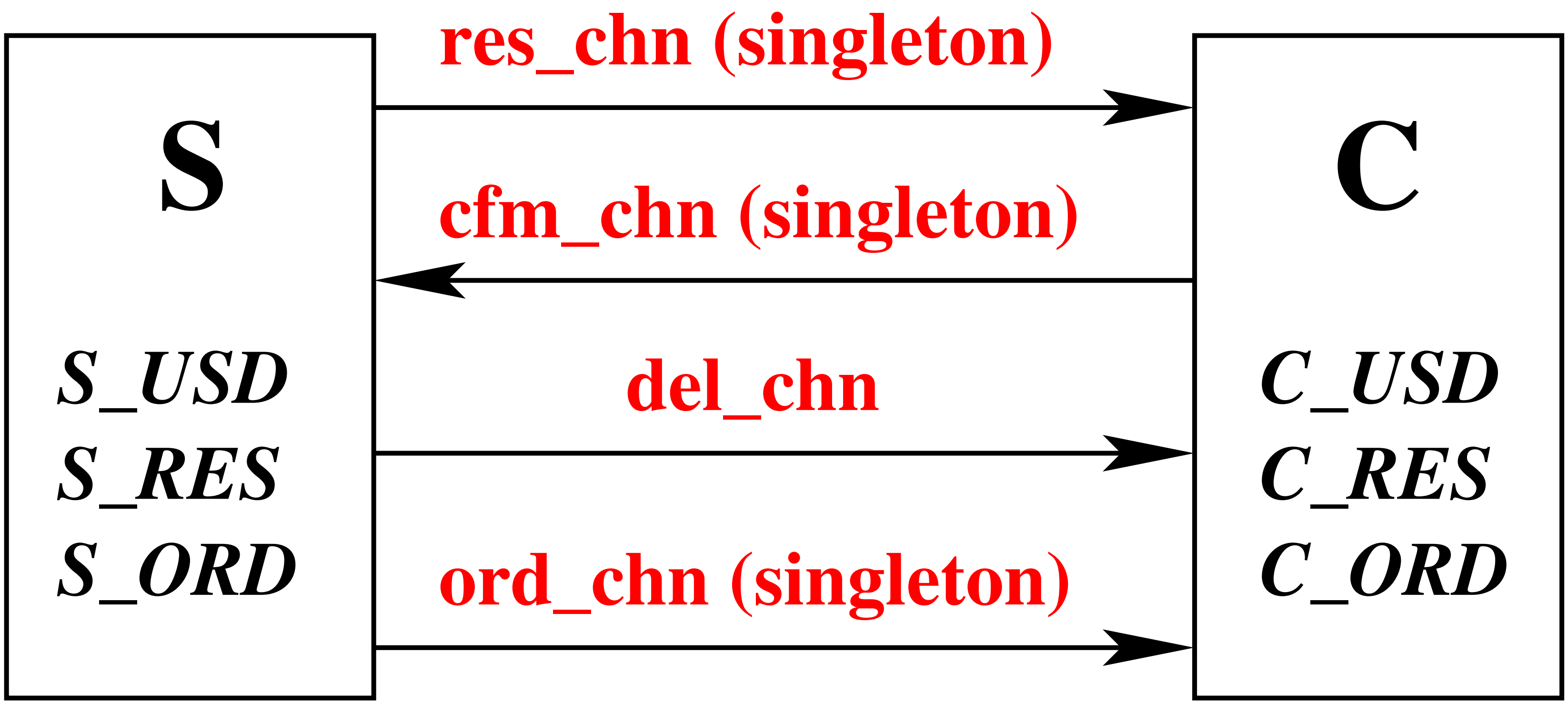
At the end of the protocol, S may order a reserved product	FUN-3

- Adding C variables:

  C_USD, C_RES, and C_ORD

- Adding channel variables:

  res_chn, cfm_chn, del_chn, and ord_chn



Invariants (1)

inv2_1: $C\_USD \subseteq PRD$

inv2_2: $C\_RES \subseteq C\_USD$

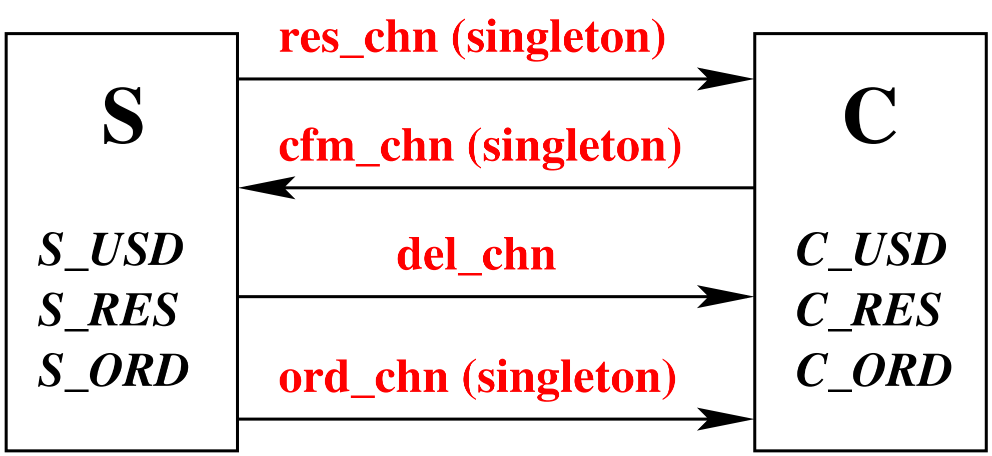
inv2_3: $C\_ORD \subseteq C\_RES$

Invariants (2)

- Connecting S and C variables

inv2_4:	$S\_USD = C\_USD \cup res\_chn$
inv2_5:	$C\_USD \cap res\_chn = \varnothing$

- C_USD and res_chn partition S_USD



Invariants (3)

- We have similar connections for other channels

inv2_6: $S\_RES \cup del\_chn = C\_RES \cup res\_chn$

inv2_7: $S\_RES \cap del\_chn = \varnothing$

inv2_8: $S\_ORD = C\_ORD \cup ord\_chn$

inv2_9: $C\_ORD \cap ord\_chn = \varnothing$

Refinement of the Abstract Events

snd_Reserve
  refines
    Reserve
  any p **where**
    $p \notin S\_USD$
    $S\_ORD = \varnothing$
    $res\_chn = \varnothing$
    $cfm\_chn = \varnothing$
  then
    $S\_USD := S\_USD \cup \{p\}$
    $S\_RES := S\_RES \cup \{p\}$
    $res\_chn := \{p\}$
  end

snd_Delete
  refines
    Delete
  any p **where**
    $p \in S\_RES$
    $S\_ORD = \varnothing$
    $res\_chn = \varnothing$
    $cfm\_chn = \varnothing$
  then
    $S\_RES := S\_RES \setminus \{p\}$
    $del\_chn := del\_chn \cup \{p\}$
  end

snd_Order
  refines
    Order
  any p **where**
    $p \in S\_RES$
    $S\_ORD = \varnothing$
    $res\_chn = \varnothing$
    $cfm\_chn = \varnothing$
  then
    $S\_ORD := \{p\}$
    $ord\_chn := \{p\}$
  end

New Events (rcv_Confirm in S, others in C)

rcv_Confirm
when
$cfm\_chn \neq \varnothing$
then
$cfm\_chn := \varnothing$
end

rcv_Reserve
when
$res\_chn \neq \varnothing$
then
$C\_USD := C\_USD \cup res\_chn$
$C\_RES := C\_RES \cup res\_chn$
$cfm\_chn := res\_chn$
$res\_chn := \varnothing$
end

rcv_Delete
any p **where**
$p \in del\_chn$
then
$C\_RES := C\_RES \setminus \{p\}$
$del\_chn := del\_chn \setminus \{p\}$
end

rcv_Order
when
$ord\_chn \neq \varnothing$
then
$C\_ORD := ord\_chn$
$ord\_chn := \varnothing$
end

Proofs (1)

(1) We cannot prove that the event rcv_Delete

rcv_Delete
  any p **where**
    $p \in del\_chn$
  then
    $C\_RES := C\_RES \setminus \{p\}$
    $del\_chn := del\_chn \setminus \{p\}$
  end

preserves invariant **inv2_6**:

inv2_6: $S\_RES \cup del\_chn = C\_RES \cup res\_chn$

- We have to introduce the following new invariant:

inv2_11: $res\_chn \cap del\_chn = \varnothing$

Proofs (2)

(2) Then we cannot prove that the event rcv_Order preserves invariant **inv2_3**

rcv_Order
when
$ord\_chn \neq \varnothing$
then
$C\_ORD := ord\_chn$
$ord\_chn := \varnothing$
end

inv2_3: $C\_ORD \subseteq C\_RES$

- This amounts to proving:

$$ord\_chn \subseteq C\_RES$$

- We, simply add this statement as a new invariant:

inv2_12: $ord\_chn \subseteq C\_RES$

Taking Account of the Requirements(1)

S can make a reservation by sending a message to C.	FUN-4

snd_Reserve
  refines
    Reserve
  any p **where**
    $p \notin S\_USD$
    $S\_ORD = \varnothing$
    $res\_chn = \varnothing$
    $cfm\_chn = \varnothing$
  then
    $S\_USD := S\_USD \cup \{p\}$
    $S\_RES := S\_RES \cup \{p\}$
    $res\_chn := \{p\}$
  end

Taking Account of the Requirements (2)

C always confirms a reservation by sending a message to S.	FUN-5

rcv_Reserve
when
$res\_chn \neq \varnothing$
then
$C\_USD := C\_USD \cup res\_chn$
$C\_RES := C\_RES \cup res\_chn$
$cfm\_chn := res\_chn$
$res\_chn := \varnothing$
end

Taking Account of the Requirements (3)

S can delete a reserved product by sending a message to C.	FUN-6

```
snd_Delete
  refines
    Delete
  any p where
    p ∈ S_RES
    S_ORD = ∅
    res_chn = ∅
    cfm_chn = ∅
  then
    S_RES := S_RES \ {p}
    del_chn := del_chn ∪ {p}
  end
```

Taking Account of the Requirements (4)

S can order a reserved product by sending a message to C.	FUN-7

snd_Order
  refines
    Order
  any p **where**
    $p \in S\_RES$
    $S\_ORD = \varnothing$
    $res\_chn = \varnothing$
    $cfm\_chn = \varnothing$
  then
    $S\_ORD := \{p\}$
    $ord\_chn := \{p\}$
  end

Order, deletion or reservation messages cannot be sent by S beteween a reservation message and its confirmation.	FUN-8

The ordered product must be the same for both partners	FUN-9

- This is achieved thanks to the following theorem:

thm2_2: $ord\_chn = \varnothing \;\Rightarrow\; S\_ORD = C\_ORD$

- Requirements
- Refinement strategy
- Successive refined models:
  - constants
  - variables
  - invariants
  - events
  - proofs
  - requirements meeting

Conclusion

- I shortly presented a practice of formal modeling

- It is done with an approach called Event B

- *Modeling in Event-B: System and Software Engineering*
  by J-R. Abrial. Cambridge University Press (2010)

- It is developed within some European Projects: Rodin and Deploy

- Loading the free software of the Rodin Platform: http://event-b.org

- An illustrating demo