Sequential programs, refinement, and proof obligations[1]

Manuel Carro
manuel.carro@upm.es

Universidad Politécnica de Madrid &
IMDEA Software Institute

[1] Several slides, examples, borrowed from J. R. Abrial

All you ever wanted to know about installing Rodin...

...is at

https://wp.software.imdea.org/cbc/#tools

and

https://wp.software.imdea.org/cbc/rodin-installation-and-tips/

Sequential programs and Event B

- Sequential programs can be transpiled into Event B.
- Correctness, termination, etc. proven with Event B tools.
- However, underuse of Event B.
  Other approaches are very good at this.

- Better approach: design with Event B from the beginning.
- Apply to reactive and concurrent systems – strong points of Event B.
- For illustration: will develop several sequential programs.

Appetizer

Let us use Rodin with the *Integer Division* example.

CONSTANTS b, c
AXIOMS $b \in \mathbb{N}, c \in \mathbb{N}_1$

VARIABLES a, r
INVARIANTS $a \in \mathbb{N}, r \in \mathbb{N}, a \times c + r = b$

INITIALISATION
$a, r := 0, b$
END

EVENT Progress
WHERE $r >= c$ THEN
$r, a := r - c, a + 1$
END

EVENT Finish
WHERE $r < c$ THEN
skip
END

Two types of components in a Rodin project:

Context(s) Contains constants and axioms.

Machine(s) Variables, invariants, and events (and some other things). Machines *see* Contexts.

Switching to Rodin. The example I type is available as part of the course material.