Rigorous Development of Fault-Tolerant Systems through Co-Refinement

Ilya Lopatkin & Alexander Romanovsky

Newcastle University, UK

ilya.lopatkin@gmail.com
alexander.romanovsky@newcastle.ac.uk

Motivations

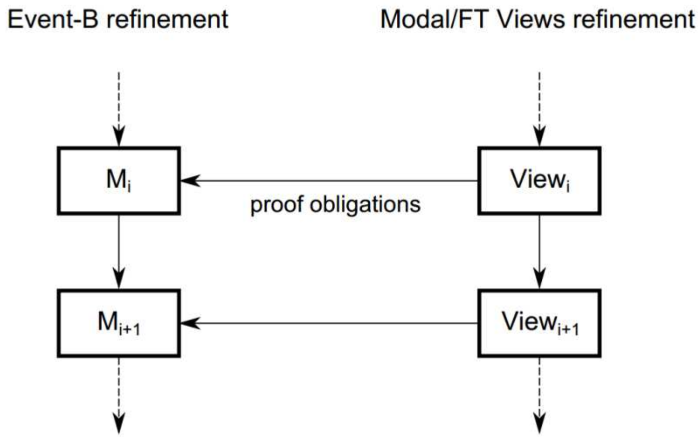
- Tackle systems complexity
- Facilitate industrial acceptance of formal methods
- Improve formalisation of FT requirements
  - High proportion of FT-related requirements to critical systems
  - Fault tolerance requirements are typically intertwined with functional ones
  - Fault assumptions and rigorous definitions of FT requirements rarely make their way to formal models

Introduction

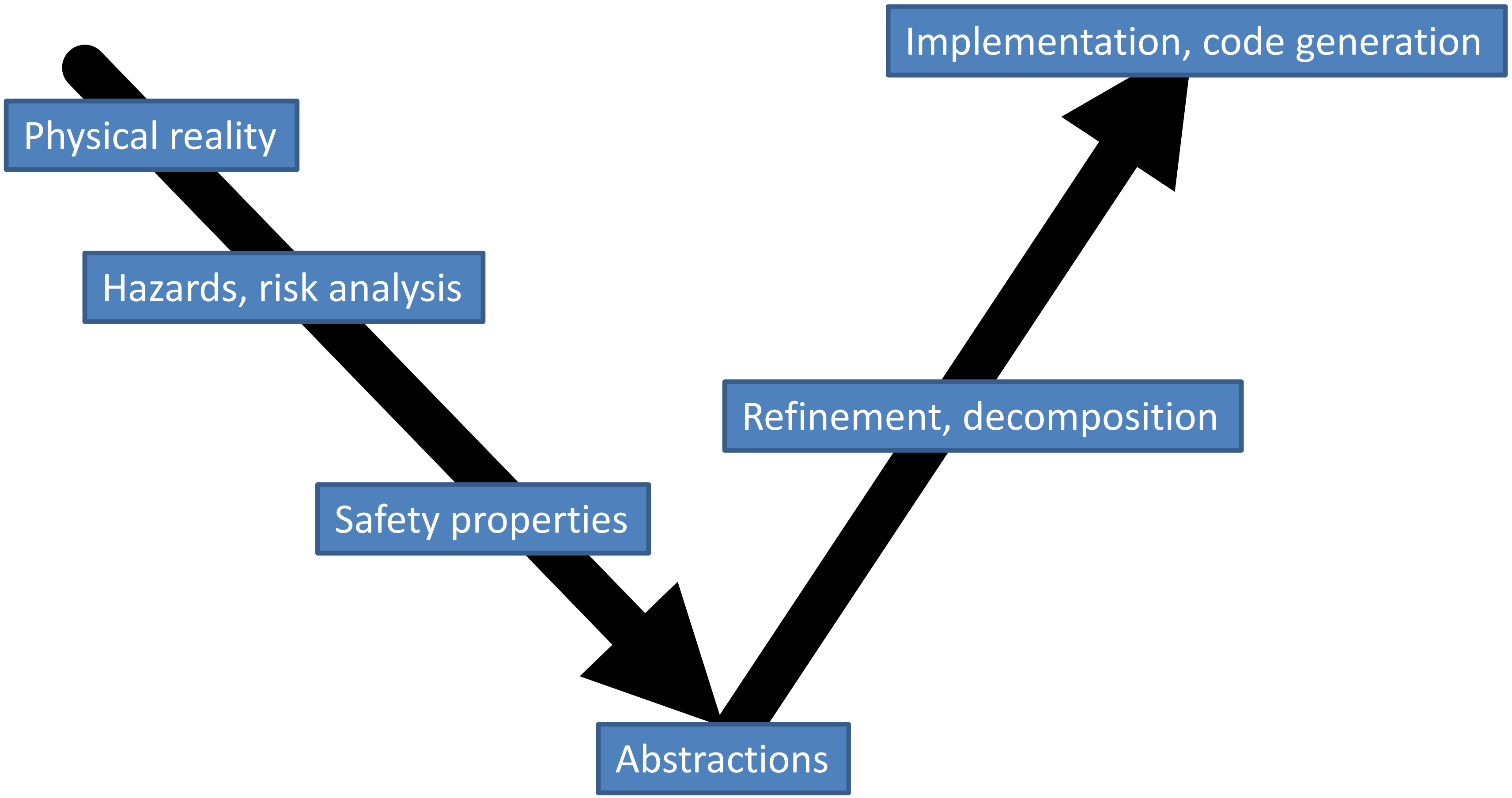
- Refinement
  - Structure complex requirements
  - Correctness-preserving steps
- Separation of concerns & multiple notations
  - Main code vs test cases
  - Process description vs temporal properties
  - State machine vs safety properties
  - UML
  - Multiple views with mutual dependencies

Overview

- Refinement-based formalism (Event-B)
- Diagrammatic formalism (Mode Views)
- Co-refinement
- Focus on proving safety properties



Development process



Constituents

- Modelling principles
- Refinement strategy
- Modelling patterns

Principle

Pattern

Modelling principles

- **Reactive style**: *cause => reaction* (properties)
  - *Cause* is typically a state of environment
  - *Reaction* is a system state
- **Behaviour restriction**
  - Start with *unconstrained behaviour*
  - Add *constraints* during refinement

Modelling principles

- **Implementable causality rule** (behaviour)
  - *Cause* (environmental change) must not depend on a *reaction* (system change)
  - Careful with system actions

when door = CLOSED

then sensor = true

– will prove the invariant but won't implement

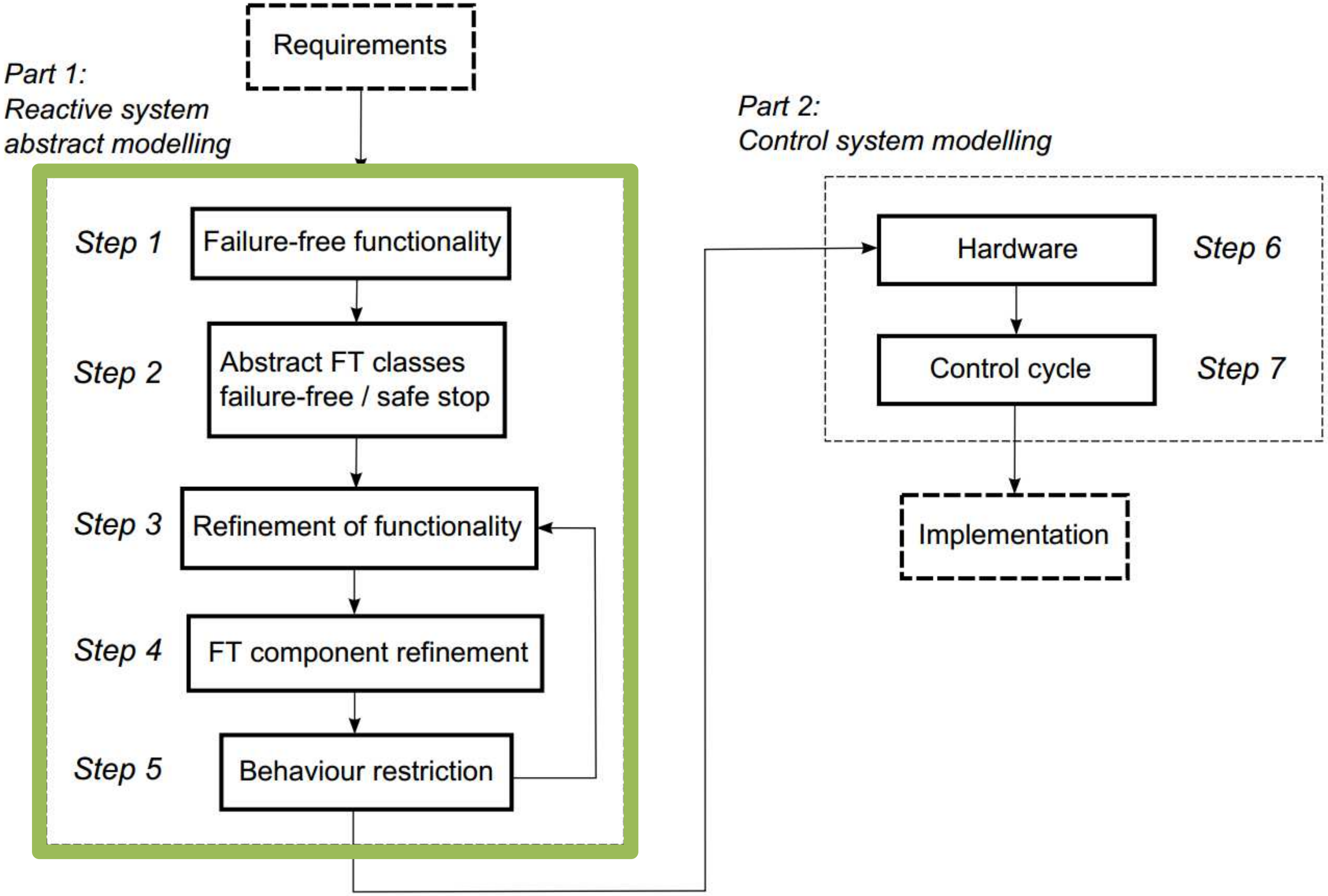
Modelling principles

- **Fault tolerant component**
  - Structuring mechanism
  - Hierarchical definition of system components via *functional* and *error* state variables

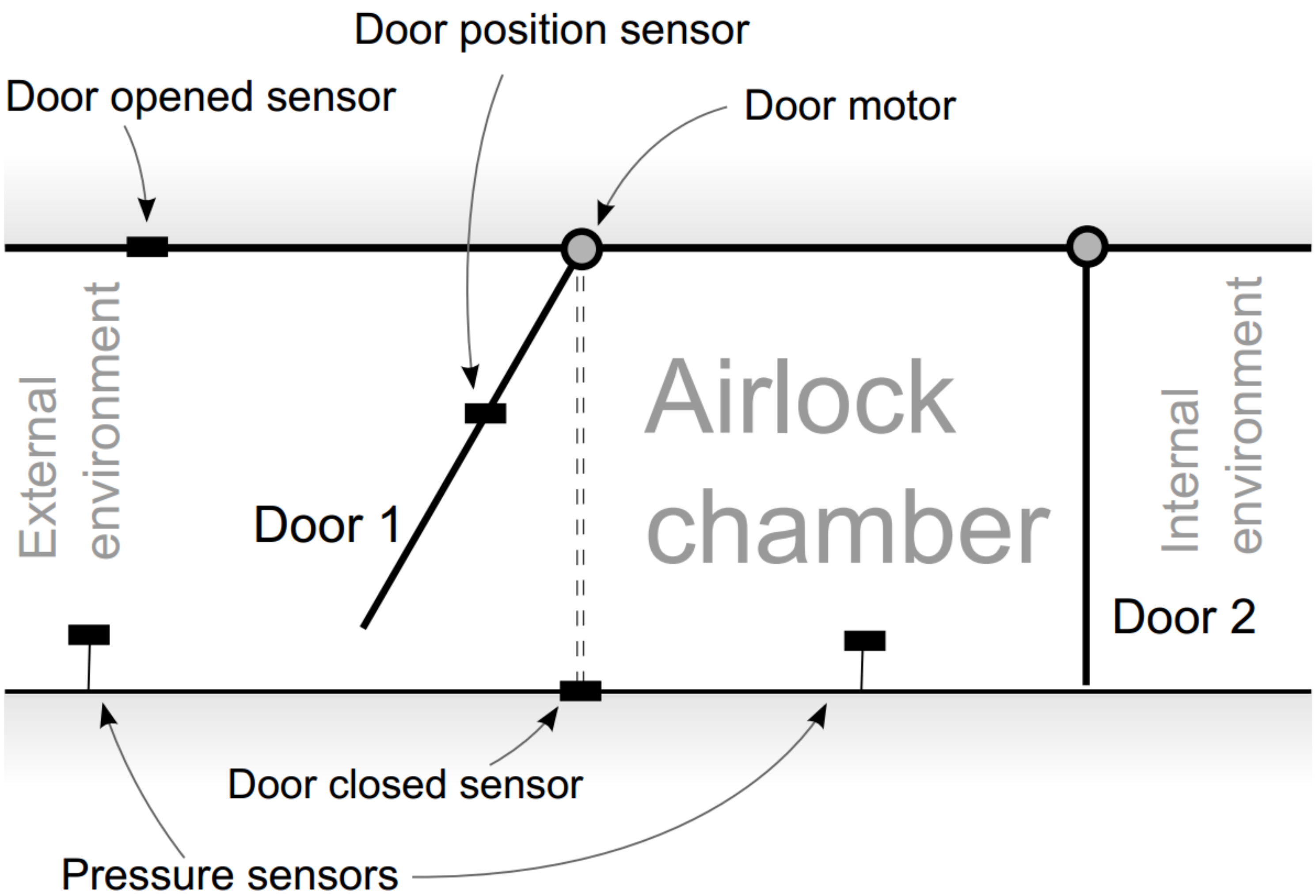
door_state: {OPENED, CLOSED}

door_condition: {OPERATIONAL, BROKEN}

Refinement strategy



Example system



Safety requirements

- (SAF1) The pressure in the chamber must always be between the lower external pressure and the higher internal one
- (SAF2) A door can only be opened if the pressure values in the chamber and the conjoined environment are equal
- (SAF3) At most one door is allowed to be opened at any moment of time
- (SAF4) The pressure in the chamber shall not be changed unless both doors are closed

Failure-free functionality (1)

axioms

axm1: $partition(DOOR\_STATE, \{OPENED\}, \{CLOSED\}, \{OPENING\}, \{CLOSING\}, \{STOPPED\})$
axm2: $LOW\_PRESSURE = 0$
axm3: $HIGH\_PRESSURE = 2$

World

invariants

inv1: $door1 \in DOOR\_STATE$
inv2: $door2 \in DOOR\_STATE$
inv3: $pressure \in \mathbb{N}$
inv4: $door1 \neq CLOSED \Rightarrow pressure = LOW\_PRESSURE$ ← SAF2
inv5: $door2 \neq CLOSED \Rightarrow pressure = HIGH\_PRESSURE$ ← SAF2
inv6: $door1 = CLOSED \vee door2 = CLOSED$ ← SAF3
inv7: $pressure > LOW\_PRESSURE \Rightarrow door1 = CLOSED$ ← SAF4
inv8: $pressure < HIGH\_PRESSURE \Rightarrow door2 = CLOSED$ ← SAF4
inv9: $pressure \geq LOW\_PRESSURE \wedge pressure \leq HIGH\_PRESSURE$ ← SAF1

events

event open1 $\widehat{=}$

when

grd1: $door1 = CLOSED \vee door1 = STOPPED$
grd2: $pressure = LOW\_PRESSURE$
grd3: $door2 = CLOSED$

then

act1: $door1 := OPENING$

end

Behaviour

Abstract class of system FT

2

event open1 $\widehat{=}$ **extends** open1

  when grd_stopped: $stopped = FALSE$

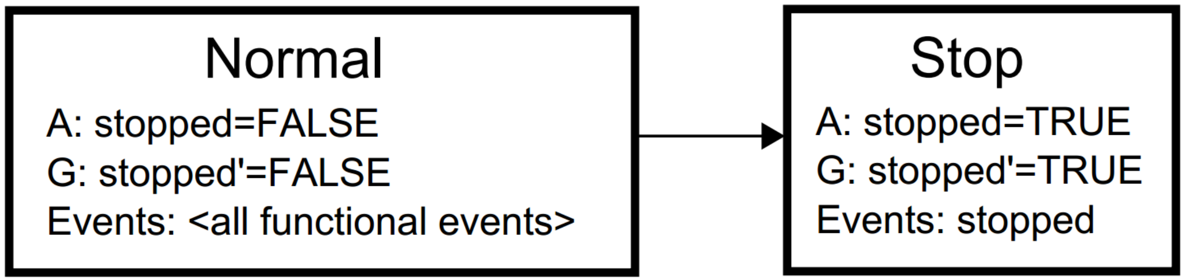
event stop $\widehat{=}$

  when grd_stopped: $stopped = FALSE$
  then act_stopped: $stopped := TRUE$

event stopped $\widehat{=}$

  when grd: $stopped = TRUE$
  then $skip$

Safe stop pattern



Safe stop template

FT requirements

- (FT1) The system shall disallow opening a degraded door
- (FT2) The system shall stop if at least one of the doors is broken
- (FT3) If both doors are degraded, the system shall stop unless there is a user in the chamber. If the user is present in the chamber, the system shall allow opening the inner door

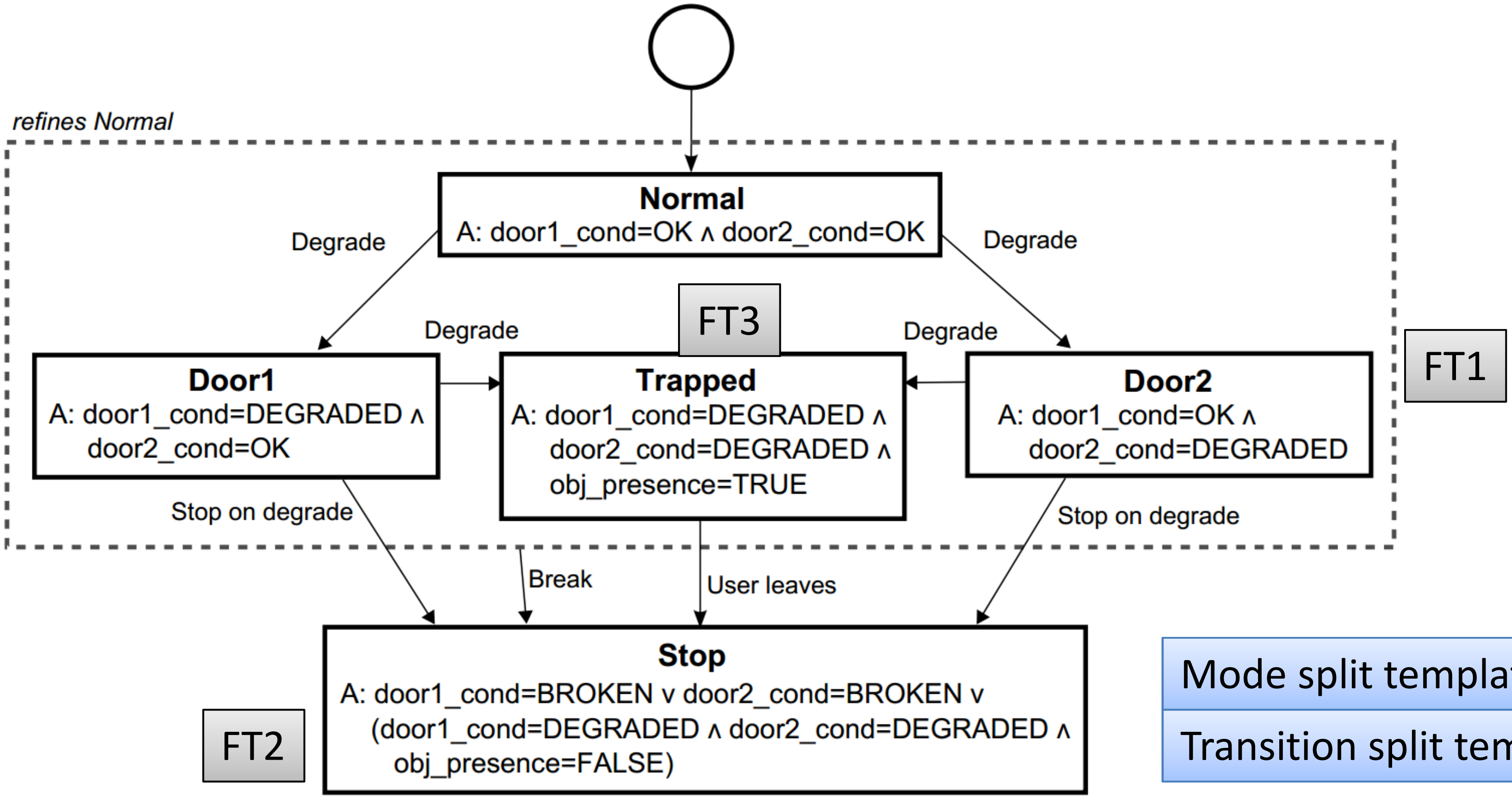
3
4

Fault tolerant component refinement

Fault tolerant component

Error state variable

$$door1\_cond, door2\_cond : \{BROKEN, DEGRADED, OK\}$$



Mode split template

Transition split template

Fault tolerant component refinement

4

Error state invariant

$$door1\_cond = BROKEN \lor door2\_cond = BROKEN \lor (door1\_cond = DEGRADED \land door2\_cond = DEGRADED \land obj\_presence = FALSE) \Leftrightarrow stopped = TRUE$$

event `break` $\widehat{=}$ **extends** `stop`

event `degrade` $\widehat{=}$

event `stop_on_degrade` $\widehat{=}$ **extends** `stop`

when

grd1: $door1\_cond = DEGRADED \lor door2\_cond = DEGRADED$

grd2: $obj\_presence = FALSE$

grd4: $door1\_cond = OK \lor door2\_cond = OK$

then

act1: $door1\_cond := DEGRADED$

act2: $door2\_cond := DEGRADED$

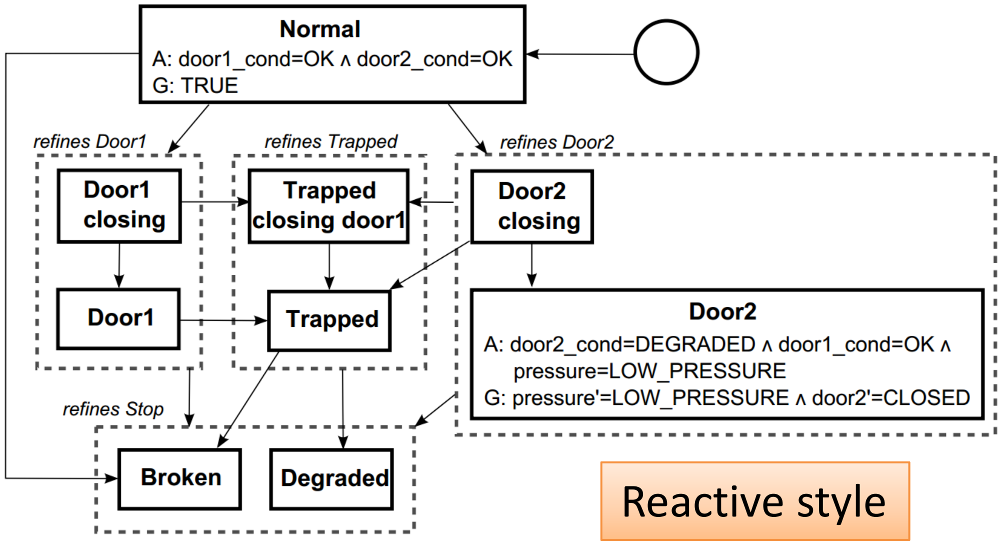
Fault tolerant behaviour

Implementable causality

Behaviour restriction

5

event open1 $\widehat{=}$ **extends** open1
when $door1\_cond = OK$



Behaviour restriction pattern

Implementable causality

Reactive style

Low-level features

- 6 Sensors, actuators through refinement of fault tolerant components
- 6 Environment (who changes pressure?)
- 7 Control cycle
- Implementation

Numbers

- Rodin environment
- 5 Event-B machines
- 3 Modal views
- 356/417 proof obligations proven automatically
- 61 are Event-B POs

Conclusions

- Another medium-scale case study (AOCS) and a number of smaller ones
- Streamlined approach to refinement-based modelling
- Focus on demonstrating safety properties
- Additional viewpoint
  - Adds rigour to the development process
  - Represents system-level FT behaviour
  - Captures FT requirements

Some links

- More details about FT views:

http://wiki.event-b.org/index.php/Mode/FT_Views

- Previous works:

I. I. Lopatkin. A Method for Rigorous Development of Fault-Tolerant Systems. PhD thesis, Newcastle University, 2013

II. Lopatkin, A. Iliasov, and A. Romanovsky. Rigorous Development of Dependable Systems using Fault Tolerance Views. ISSRE'11

III. I. Lopatkin, A. Iliasov, A. Romanovsky, Y. Prokhorova, and E. Troubitsyna. Patterns for Representing FMEA in Formal Specification of Control Systems, HASE'11

IV. F. L. Dotti, A. Iliasov, L. Ribeiro, and A. Romanovsky. Modal systems: Specification, refinement and realisation, ICFEM '09