UMONS
Université de Mons

inforTech
UMONS RESEARCH INSTITUTE
FOR INFORMATION TECHNOLOGY
AND COMPUTER SCIENCE

Advanced support for
executable statechart modelling

Tom **Mens &** Alexandre **Decan**
Software Engineering Lab
Department of Computer Science

informatique.umons.ac.be/genlog

Research Context
Model-driven software engineering

Goal

Increase quality and reliability of software systems *before implementation phase* through use of visual design models

How?

- Specify structure and behaviour of software-intensive systems
  - at high level of abstraction
  - without considering technical details
- Allow formal reasoning over the system
- Test and simulate system behaviour
- Facilitate system evolution
- Explore design alternatives
- Automated code generation

Research Context
Model-driven software engineering

Activities: Model execution, model simulation, automated testing, code generation, …

Modeling languages: UML models, business process models, …

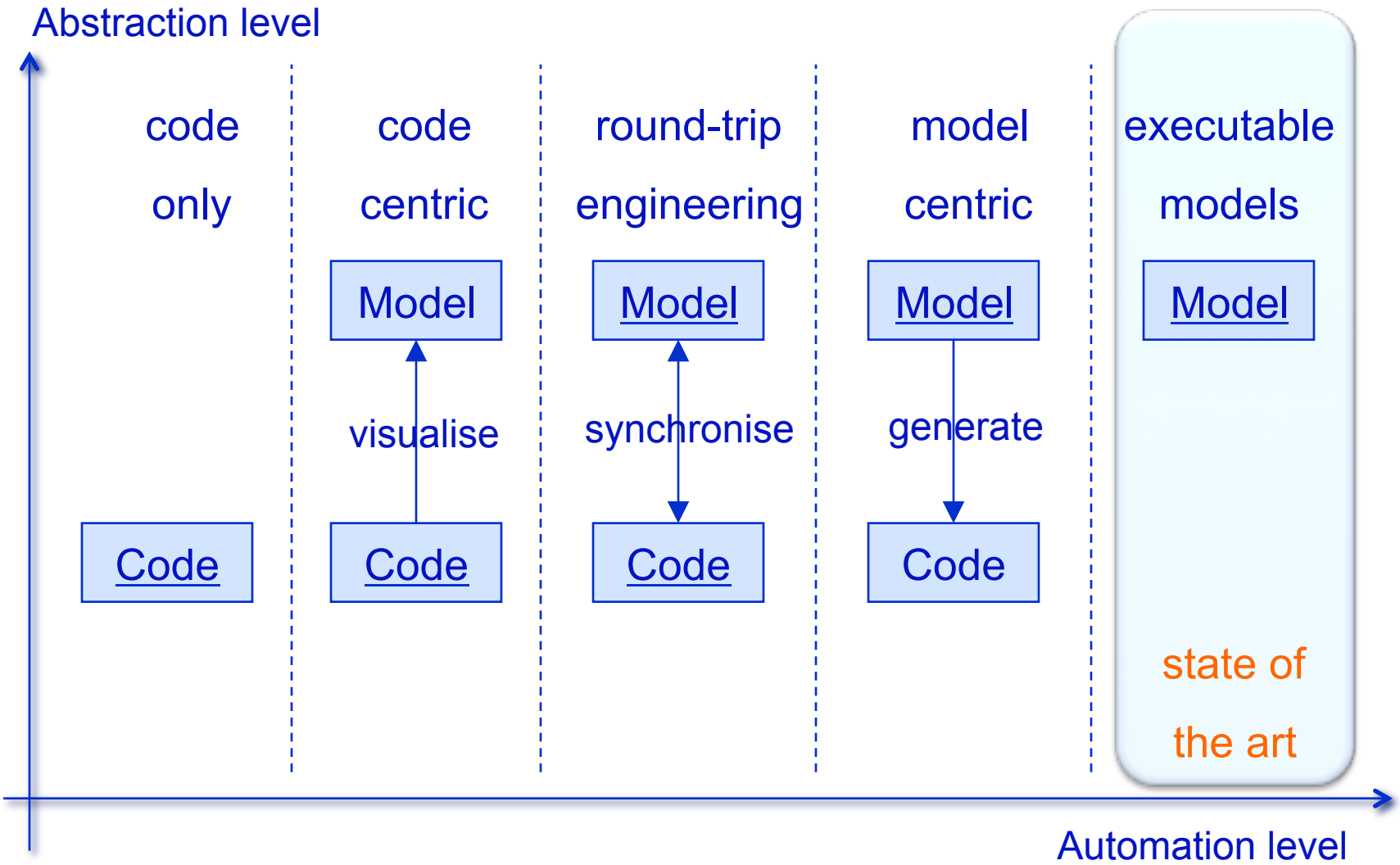
Bottom Up

Top Down

Activities: Formal verification, model checking, theorem proving, …

Formalisms: temporal logics, automata, Petri nets, game theory, …

Research Context
Executable modelling



Research Context
Executable modelling

Focus on statechart models

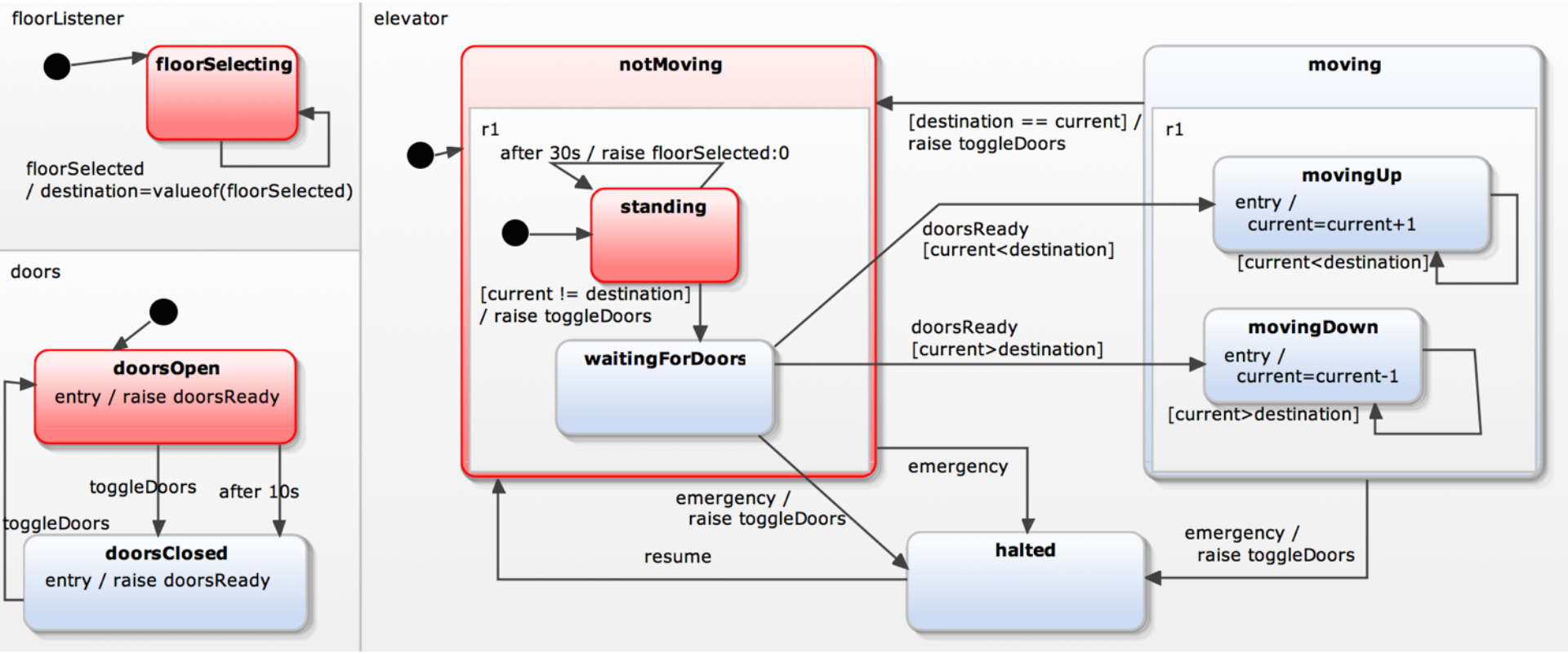
- Frequently used in industry

- Well-suited for describing event-driven behaviour of concurrent, real-time systems

Executable statechart modelling

Elevator example

UMONS Université de Mons



Pros and cons

- Commercial tool support available
  - IBM Statemate, IBM Rhapsody, MathWorks Stateflow, Yakindu Statechart Tools
- "Standardisation" through UML

- Many semantic variations
- No open source solutions
- Limited support for advanced development techniques

Research goals

Provide more advanced support for statecharts

- Dealing with semantic variation
- Automated testing and test generation
- Design by contract
- Behaviour-driven development
- Formal verification and model checking
- Composition mechanisms
- Design space exploration
- Detecting quality problems
- Applying model refactoring
- Model evolution

Sismic

UMONS Université de Mons

- Interactive Statechart Model Interpreter and Checker
  - Python library available on Python Package Index (PyPI)
  - released under open source licence LGPL v3
  - Source code
    - github.com/AlexandreDecan/sismic
  - Documentation
    - sismic.readthedocs.org

Executable statechart modelling

Sismic

UMONS Université de Mons

- *Executing statechart behaviour*

```
simulator = Interpreter(my_statechart)
simulator.execute_once()
simulator.queue(Event(’floorSelected’, floor=4))
simulator.execute_once()
```

- *Defining and running a story*

```
story = Story([Event('floorSelected', floor=1),
               Pause(10),
               Event('floorSelected', floor=4),
               Pause(10)])
story.tell(simulator)
print(simulator.time)                    # 20
print(simulator.context.get('current'))  # 4
```

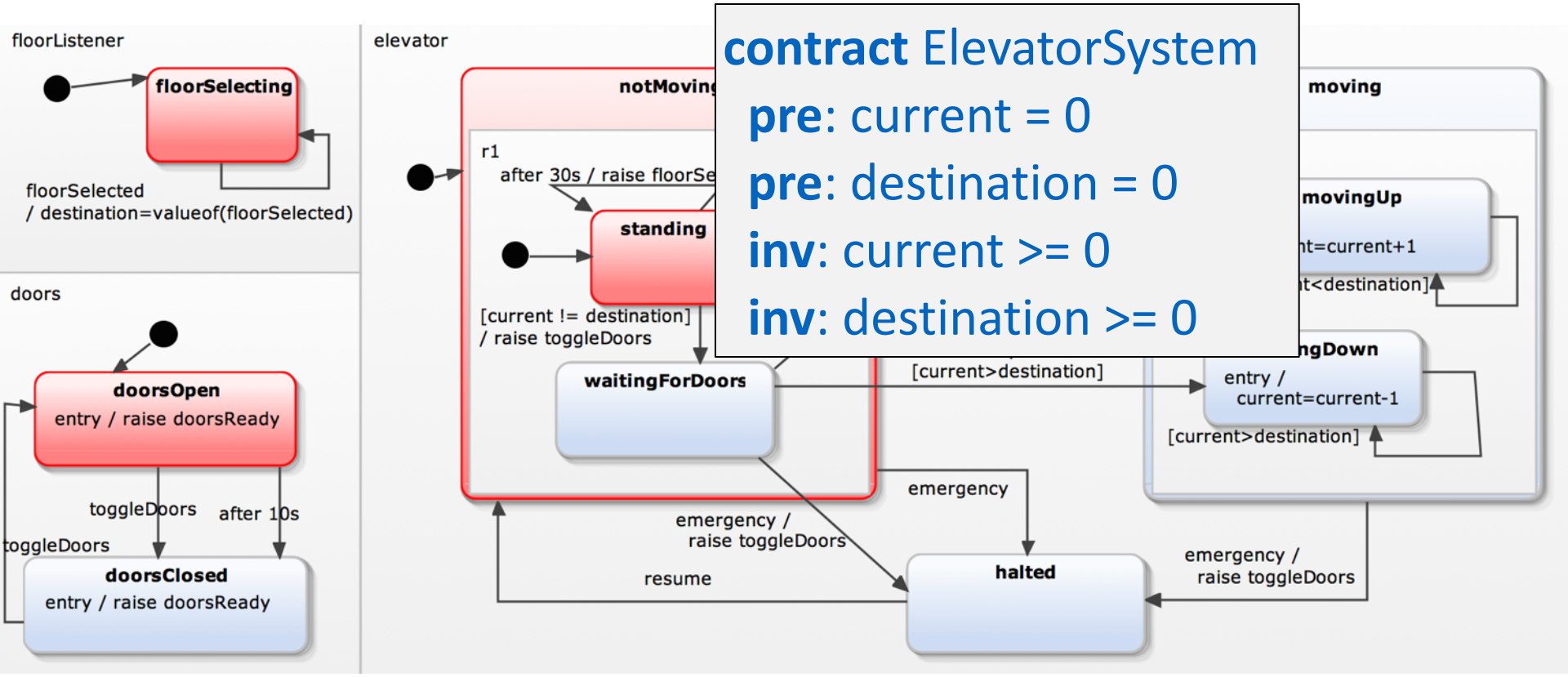
Contract-driven development

UMONS Université de Mons

- Add precise and dynamically verifiable specifications to executable software components (e.g., methods, functions, classes)
- Based on Bertrand Meyer's "*Design by Contract*"
- The code should respect a *contract*, composed of
  - *preconditions*
  - *postconditions*
  - *invariants*

```
class DICTIONARY [ ELEMENT ]
 feature
  put (x : ELEMENT; key : STRING ) is
  require
    count <= capacity
    not key.empty
  ensure
    has (x)
    item (key) = x
    count = old count + 1
 end
 invariant
   0 <= count
   count <= capacity
 end
```

Example of statechart contract



Example of statechart contract

context notMoving
pre: destination = current

context moving
pre: destination <> current
post: current = destination**@pre**

floorListener
floorSelecting
floorSelected
/ destination=valueof(floorSelected)

doors
doorsOpen
entry / raise doorsReady

notMoving
r1
after 30s / raise floorSelected:0
standing
[current != destination]
/ raise toggleDoors
waitingForDoors

[destination == current] /
raise toggleDoors

doorsReady
[current<destination]

doorsReady
[current>destin

moving
r1
movingUp
entry /
current=current+1
[current<destination]

emergency

emergency /
raise toggleDoors

resume

halted

emergency /
raise toggleDoors

context movingUp
pre: current < destination
post: current > current@pre

context doorsOpen
inv: not oclIsInState(moving)

Detecting contract violations

InvariantError

Object: BasicState(doorsOpen)
Assertion: not active('moving')
Configuration:
['doors', 'elevator', 'floorListener', 'doorsOpen', 'floorSelector', 'moving', 'movingUp']

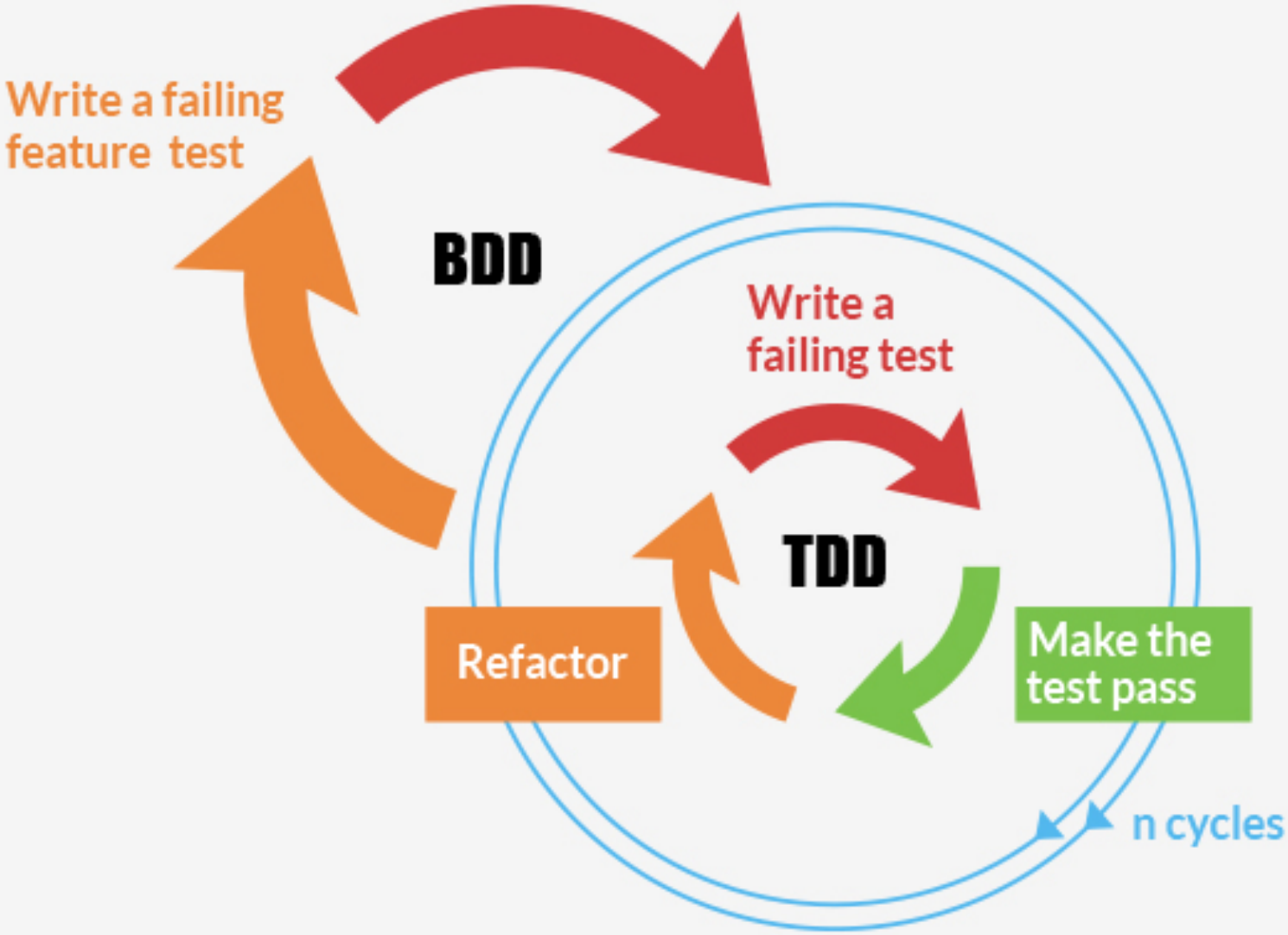
Step: MacroStep@10(
InternalEvent(doorsReady),
[Transition(waitingForDoors, movingUp, doorsReady)],
>['moving', 'movingUp'],
<['waitingForDoors', 'notMoving'])

Evaluation context:
- destination = 4
- current = 2

Executable statechart modelling

Test-driven & behaviour-driven development

UMONS Université de Mons



Behaviour-Driven Development

UMONS Université de Mons

- Include acceptance test and customer test practices into test-driven development
- Encourage collaboration between developers, QA, and non- technical stakeholders (domain experts, project managers, users)
- Use a domain-specific (non-technical) language to specify how the code should behave
  - By defining feature specifications and scenarios
  - Using Gherkin language
- Reduces the technical gap between developers and other project stakeholders

Behaviour-driven development

UMONS Université de Mons

Example
(taken from docs.behat.org/en/v2.5/guides/1.gherkin.html)

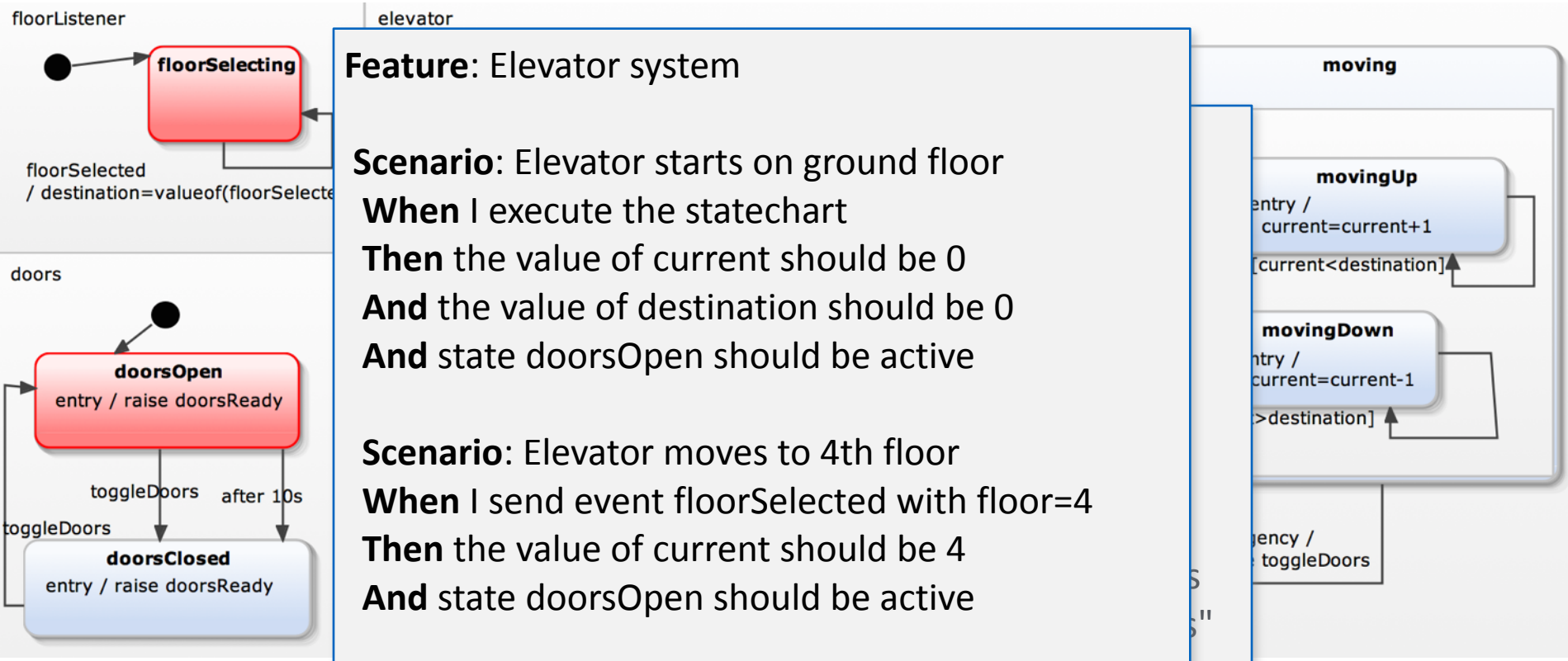
Feature: Serve coffee
In order to earn money customers should be able to buy coffee

Scenario: Buy last coffee
Given there is 1 coffee left in the machine
And I have deposited 1 dollar
When I press the coffee button
Then I should be served a coffee

Behaviour-driven development

UMONS Université de Mons

Example: Feature specification for Elevator statechart



Feature: Elevator system

Scenario: Elevator starts on ground floor
When I execute the statechart
Then the value of current should be 0
And the value of destination should be 0
And state doorsOpen should be active

Scenario: Elevator moves to 4th floor
When I send event floorSelected with floor=4
Then the value of current should be 4
And state doorsOpen should be active

Then the value of current should be 0

Behaviour-driven development

UMONS Université de Mons

- *Supporting BDD*

```
Feature: Elevator System
[...]

1 feature passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
13 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.017s
```

Behaviour-driven development

UMONS Université de Mons

- *Supporting BDD*

```
Failing scenarios :
  Elevator moves to ground after 30 secs

Assertion Failed:
  Variable current equals 4 != 0

0 features passed , 1 failed , 0 skipped
3 scenarios passed , 1 failed , 0 skipped
12 steps passed , 1 failed , 0 skipped , 0 undefined

Took 0m0.014s
```

Sismic

- *Coverage analysis*

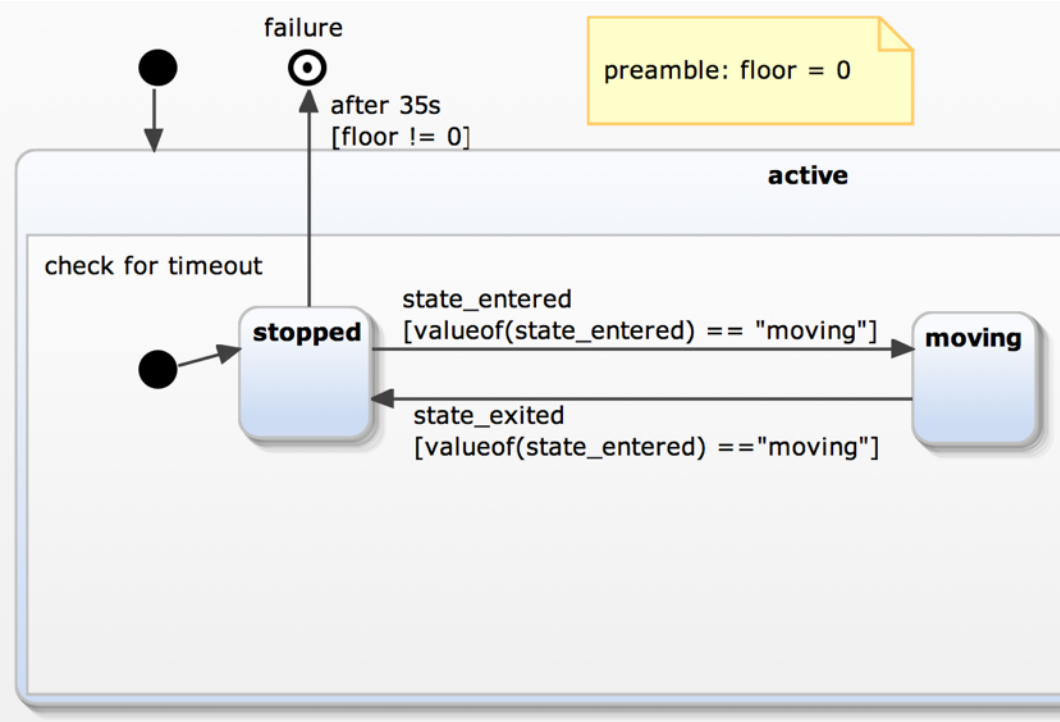
```
State coverage: 92.86%
 Entered states:
  root (4) | elevator (4) | moving (4) | movingUp (12) | movingDown (4) |
  notMoving (8) | standing (9) | waitingForDoors (4) |
  doors (4) | doorsOpen (8) | doorsClosed (6) |
  floorSelector (4)   floorListener (4) |
 Remaining states: halted

 Transition coverage: 73.33%
 Processed transitions:
  movingUp [None] -> movingUp (9) |
  moving [None] -> notMoving (4) |
  standing [None] -> waitingForDoors (4) |
  ...
```

Executable statechart modelling

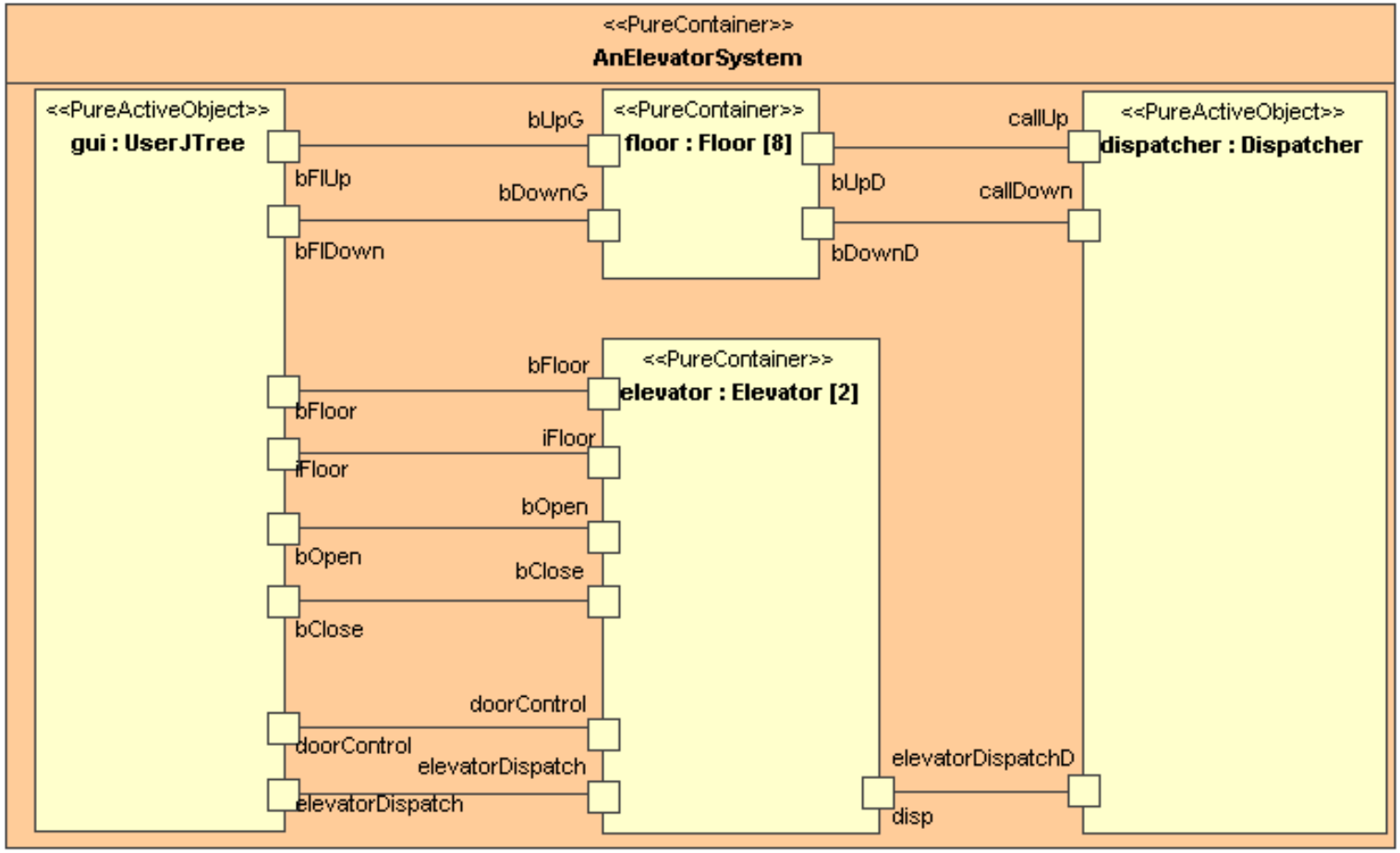
Sismic

- *Defining properties over statecharts*
  - If elevator does not receive floorSelected event during 30 seconds, ground floor should be reached 5 seconds after
  - Can be checked dynamically by means of runtime monitoring



Future work

- Composition and communication mechanisms



Future work

UMONS Université de Mons

- Automated detection of contracts, based on
  - dynamic analysis of statechart executions
  - static symbolic analysis of actions and guards
- Automated test generation
  - Based on contract specifications
  - Based on mutation testing or concolic testing
- Formal verification and model checking
  - Based on temporal logic properties
  - Expressed in domain-specific language (e.g. Dwyer specification patterns)

Future work

UMONS Université de Mons

- Support for quality analysis
  - Detection of *model smells*
- Support for quality improvement
  - Automated (behaviour preserving) *model refactoring*

Future work

- Software product family design and variability analysis
- Example: feature model of an elevator control system product line