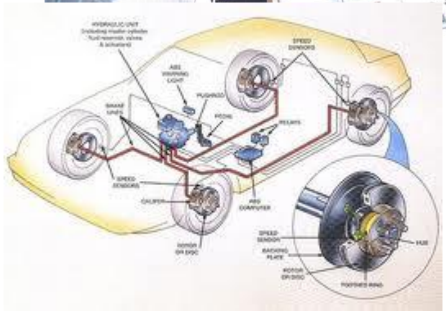
ALE
Agile Language Engineering

(2017 – 2019)

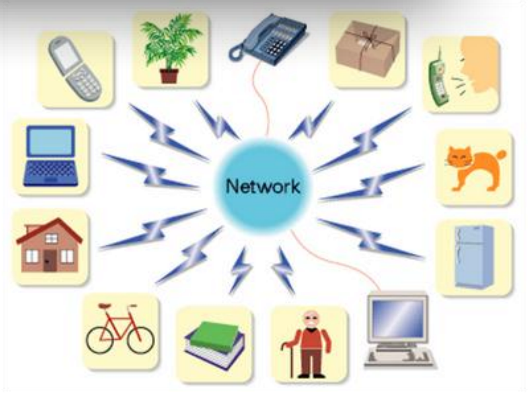
Thomas Degueule
CWI – Inria Workshop
September 19 – 20, 2017
CWI, Amsterdam

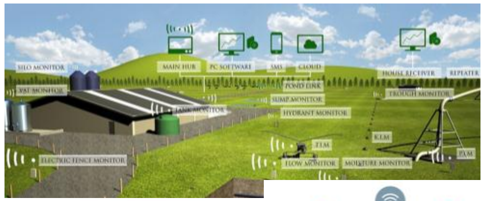
http://gemoc.org/ale/

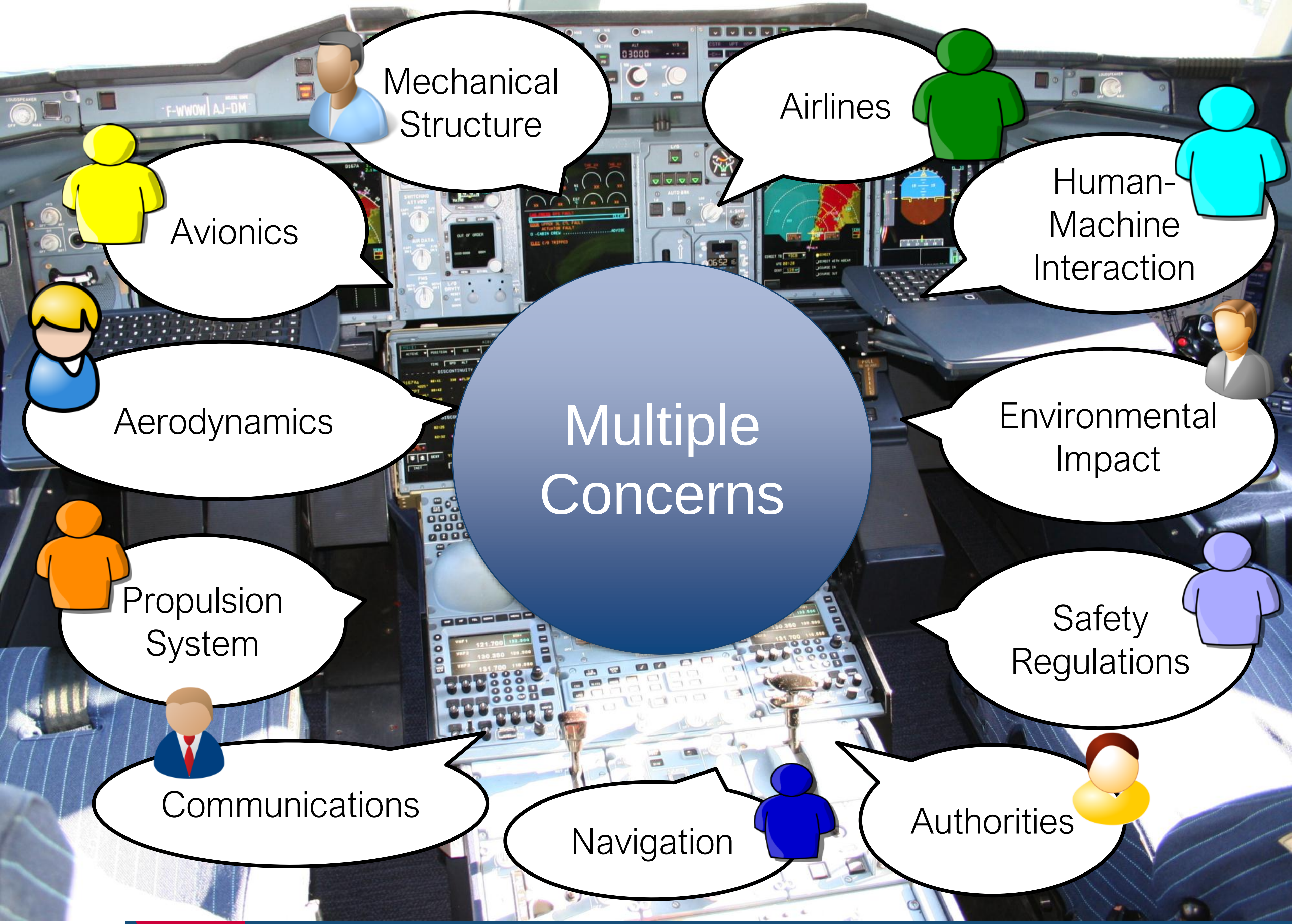
Context

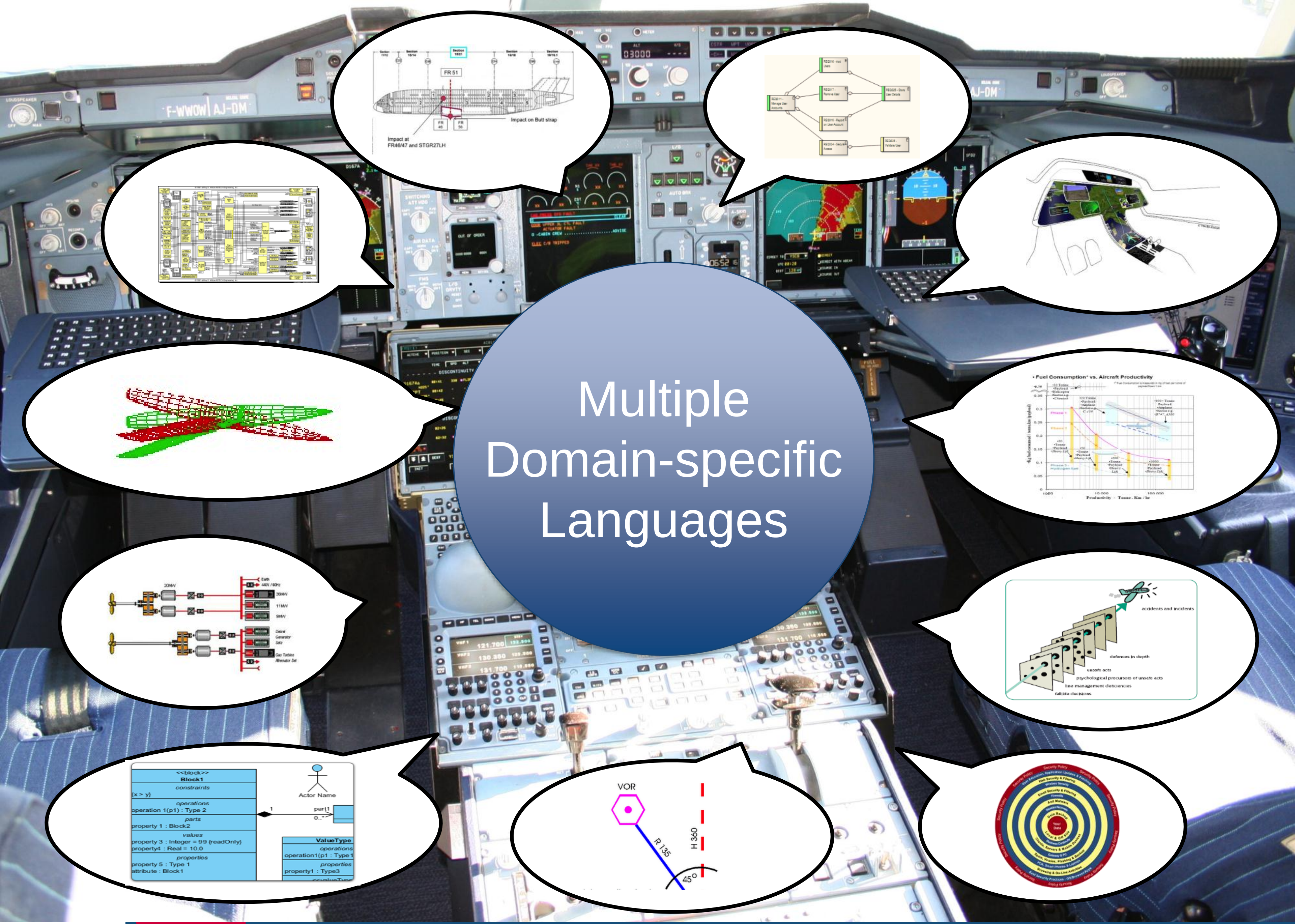


Software intensive systems

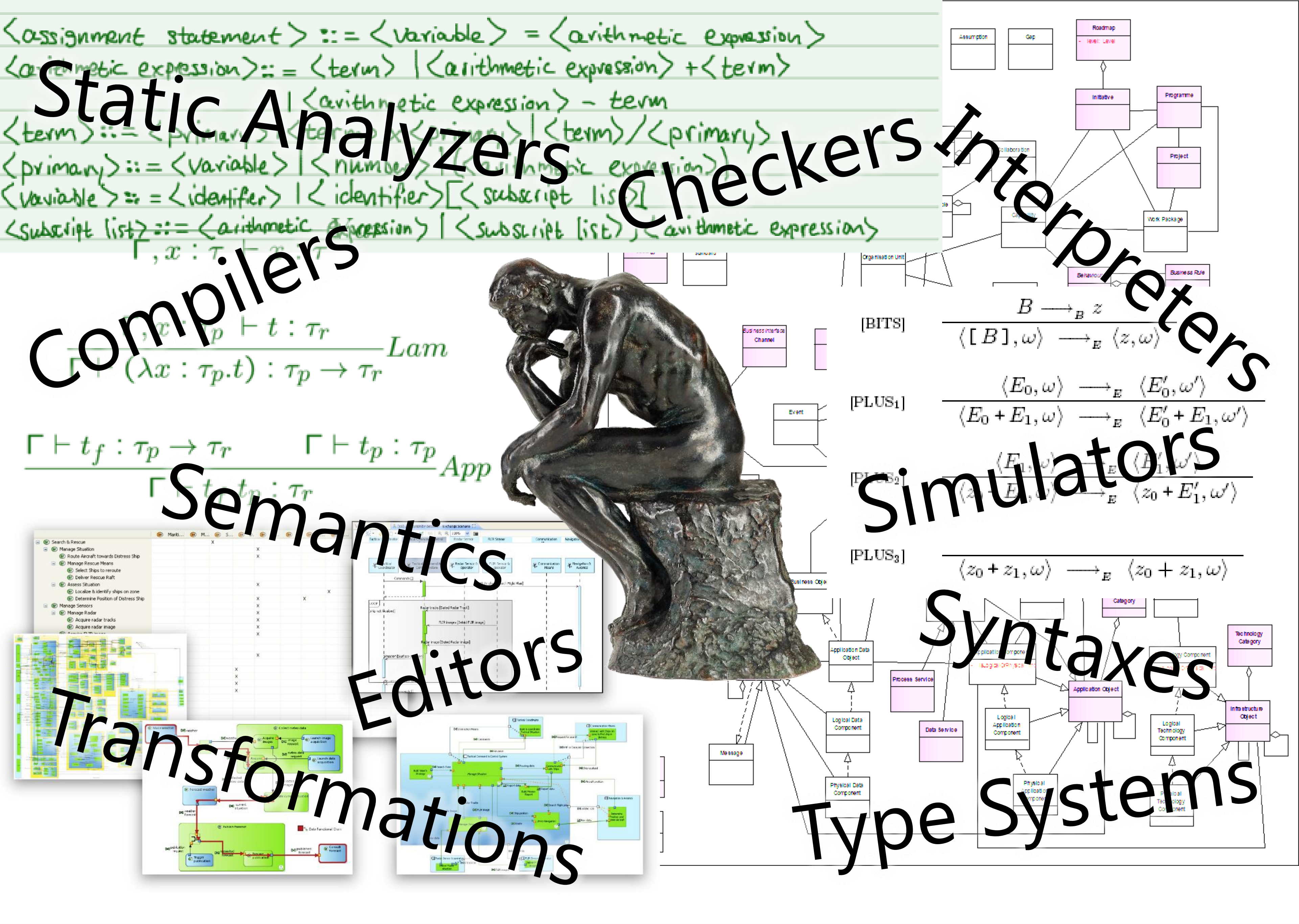





Mechanical Structure
Airlines
Avionics
Human-Machine Interaction
Multiple Concerns
Aerodynamics
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorities



Multiple Domain-specific Languages



Static Analyzers

Checkers

Interpreters

Compilers

Simulators

Semantics

Syntaxes

Editors

Transformations

Type Systems

Software Language Engineering Challenges

- **Challenge #1**: Language Modularity & Reuse
  - Modular extension
  - Incremental compilation
  - Language modules
  - Language interfaces

DSL & Tools Designer

- **Challenge #2**: Live Languages
  - Incremental modeling
  - Immediate feedback

DSL User

CWI SWAT

- Software Analysis and Transformation
- Software analysis, reverse- and re-engineering
- Strong background in metaprogramming, static analysis
- SLE: mainly *technical* DSLs (GUIs, web, configuration, etc.)

Jurgen J. Vinju
Group Leader

Tijs van der Storm
ALE Coordinator

http://rascal-mpl.org/

Ensō

http://enso-lang.org/

Inria DiverSE

- Diversity-centric Software Engineering
- Diversity of platforms, languages, features, failures
- Strong background in model-driven engineering
- SLE: mainly *business* DSLs (avionics, IoT, agronomy, etc.)

Benoit Baudry
Group Leader

Benoit Combemale
ALE Coordinator

Gemoc

http://gemoc.org/

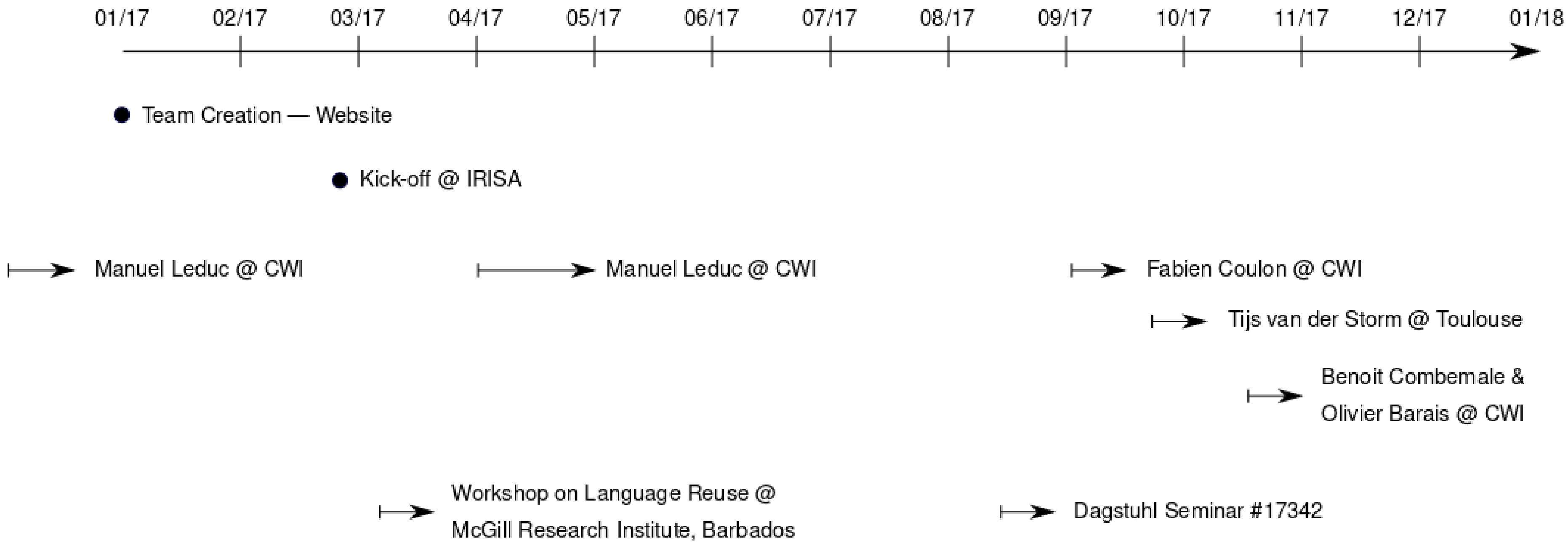
Melange

http://melange-lang.org/

ALE Members

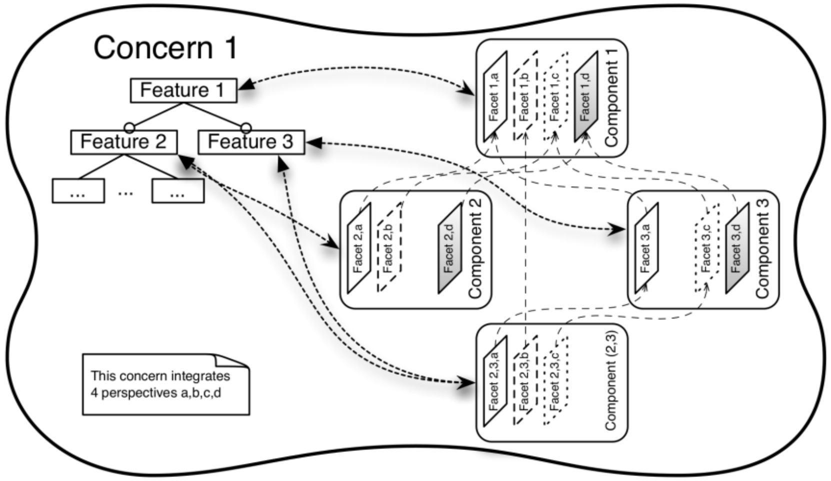
- Olivier Barais, Professor, Inria and Univ. Rennes 1, France
- Benoit Baudry, Research Scientist, Inria, France
- Benoit Combemale, Associate Professor, Inria and UR1 1, France
- Fabien Coulon, Research Engineer, Inria and UR1, France
- Thomas Degueule, Associate Research Scientist, CWI, The Netherlands
- Manuel Leduc, PhD Student, Inria and Univ. Rennes 1, France
- Riemer van Rozen, PhD Student, CWI, The Netherlands
- Tijs van der Storm, Professor, CWI, The Netherlands
- Pablo Inostroza Valdera, PhD Student, CWI, The Netherlands
- Jurgen Vinju, Professor, CWI, The Netherlands
- Didier Vojtisek, Research Engineer, Inria, France

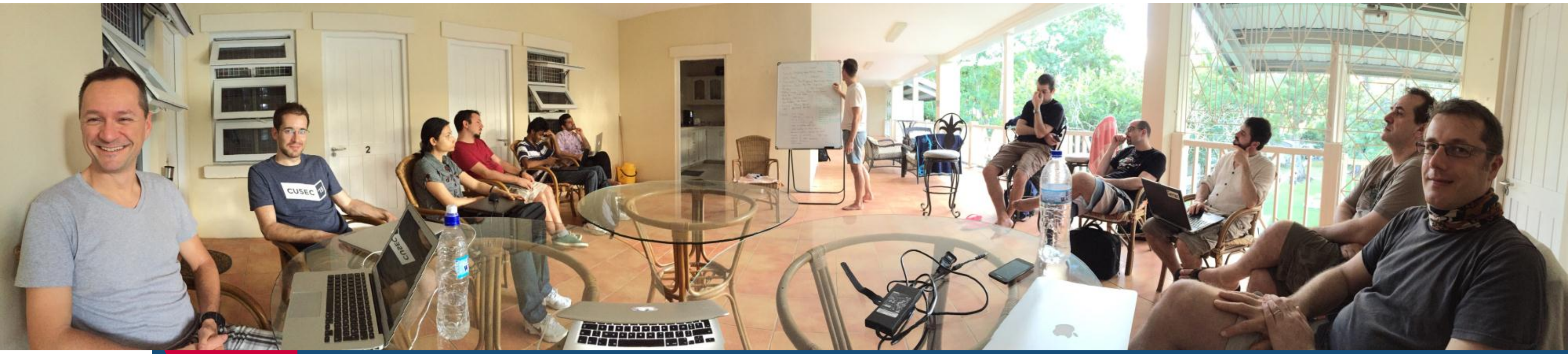
Timeline


01/17
02/17
03/17
04/17
05/17
06/17
07/17
08/17
09/17
10/17
11/17
12/17
01/18
Team Creation — Website
Kick-off @ IRISA
Manuel Leduc @ CWI
Manuel Leduc @ CWI
Fabien Coulon @ CWI
Tijs van der Storm @ Toulouse
Benoit Combemale & Olivier Barais @ CWI
Workshop on Language Reuse @ McGill Research Institute, Barbados
Dagstuhl Seminar #17342

Events

- Workshop on Language Reuse, March 17 – 24, 2017
- McGill's Bellairs Research Institute – Holetown, Barbados



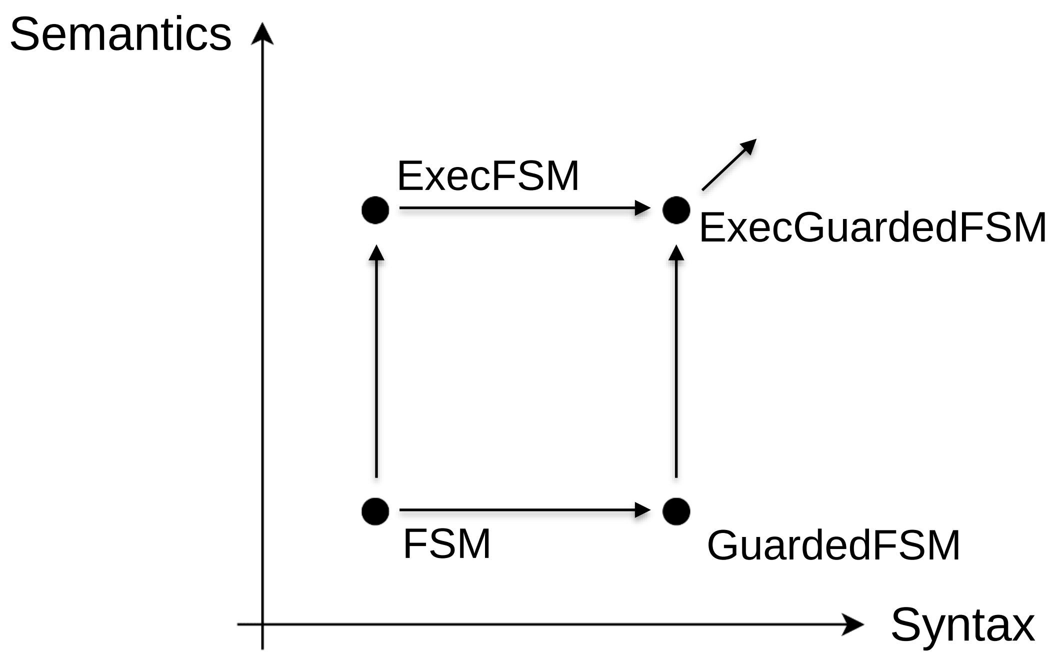


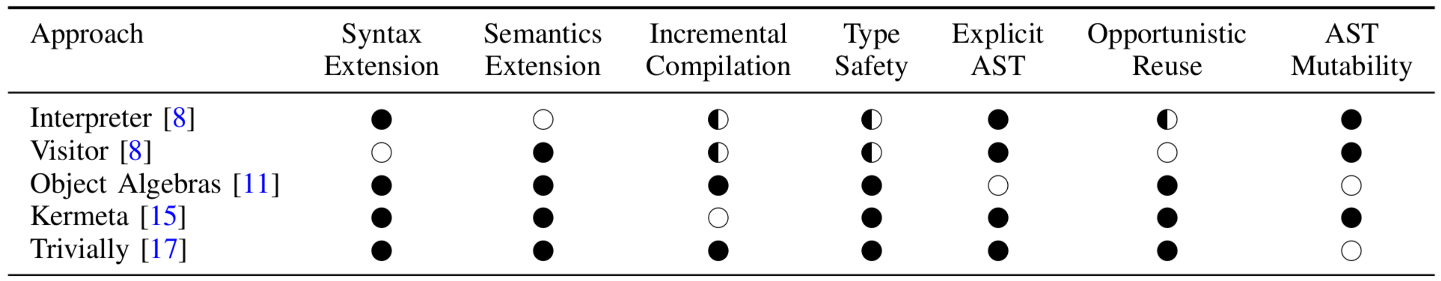
Events

- Dagstuhl Seminar #17342 (SLEBoK)
  - The *Software Language Engineering Body of Knowledge*
- August 20 – 25, 2017
- Schloss Dagstuhl – Wadern, Germany
- https://www.dagstuhl.de/17342

Results

Modular Language Extension



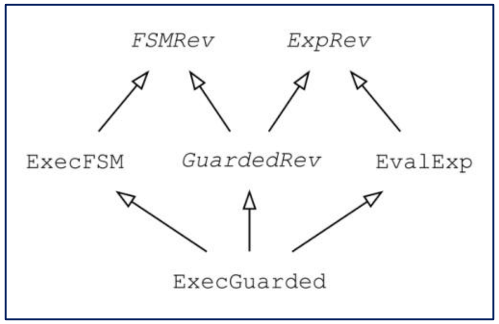


Approach	Syntax Extension	Semantics Extension	Incremental Compilation	Type Safety	Explicit AST	Opportunistic Reuse	AST Mutability
Interpreter [8]	●	○	◐	◐	●	◐	●
Visitor [8]	○	●	◐	◐	●	○	●
Object Algebras [11]	●	●	●	●	○	●	○
Kermeta [15]	●	●	○	●	●	●	●
Trivially [17]	●	●	●	●	●	●	○

The REVISITOR Pattern

MODELS 2017
austin, tx

- A language implementation pattern that enables
  1. Independent extensibility of syntax and semantics
  2. With incremental compilation
  3. Without anticipation



```
interface Pr { String print(); }
interface PrintFSM extends FSMRev<Pr, Pr, Pr, Pr> {
  default Pr machine(Machine it) {
    return () -> it.states.stream().map(s ->
        $(s).print()).collect(joining());
  }
  default Pr state(State it) { /* ... */ }
  default Pr finalState(FinalState it) {
    return () -> "*" + state(it).print();
  }
  default Pr transition(Transition it) {
    return () -> it.event + " => " + it.target.name;
  }
}
```

```
interface FSMRev<M, S, F extends S, T> {
  F finalState(FinalState it);
  S state(State it);
  default F $(FinalState it) {
    return finalState(it);
  }
  default S $(State it) {
    if(it instanceof FinalState)
      return finalState((FinalState) it);
    return state(it);
  }
}
```

Revisiting Visitors for Modular Extension of Executable DSMLs
Manuel Leduc, Thomas Degueule, Benoit Combemale, Tijs van der Storm, Olivier Barais
In 20th *International Conference on Model Driven Engineering Languages and Systems* (MODELS), 2017

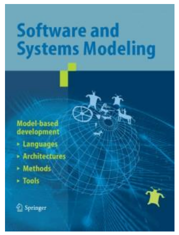
The Action Language for Ecore (ALE)

- A high-level semantics definition language that compiles to the REVISITOR pattern
- Currently transferring the technology to Obeo
- Ultimately to http://eclipse.org/ecoretools/

```
behavior execution;

import ecore “GuardedFsm.ecore”;
import ale execfsm;
import ale evalexp;

open class GuardedTransition {
  def void step(String ch) {
    if ($[self.guard].eval().equals(true))
      $[super].step(ch);
  }
}
```

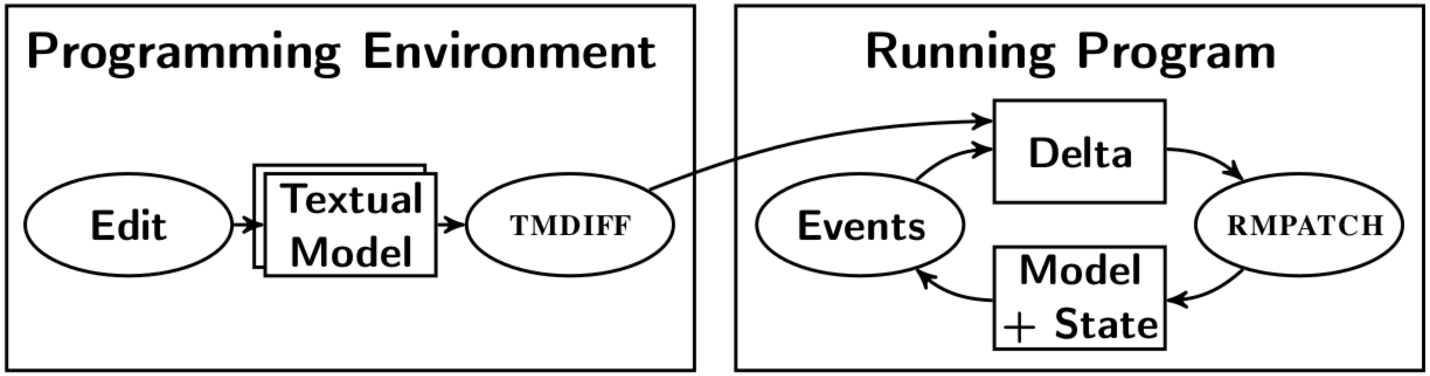
```
behavior timedprinting;

import ecore “TimedFsm.ecore”;
import ale printing;

open class TimedTransition {
  def String print() {
    return self.time + “@” + $[super].print();
  }
}
```

EcoreTools-Next: Executable DSLs made (more) accessible
Cédric Brun, Yvan Lussaud, Benoit Combemale, Fabien Coulon
Presented at *EclipseCon France, Toulouse, 2017*

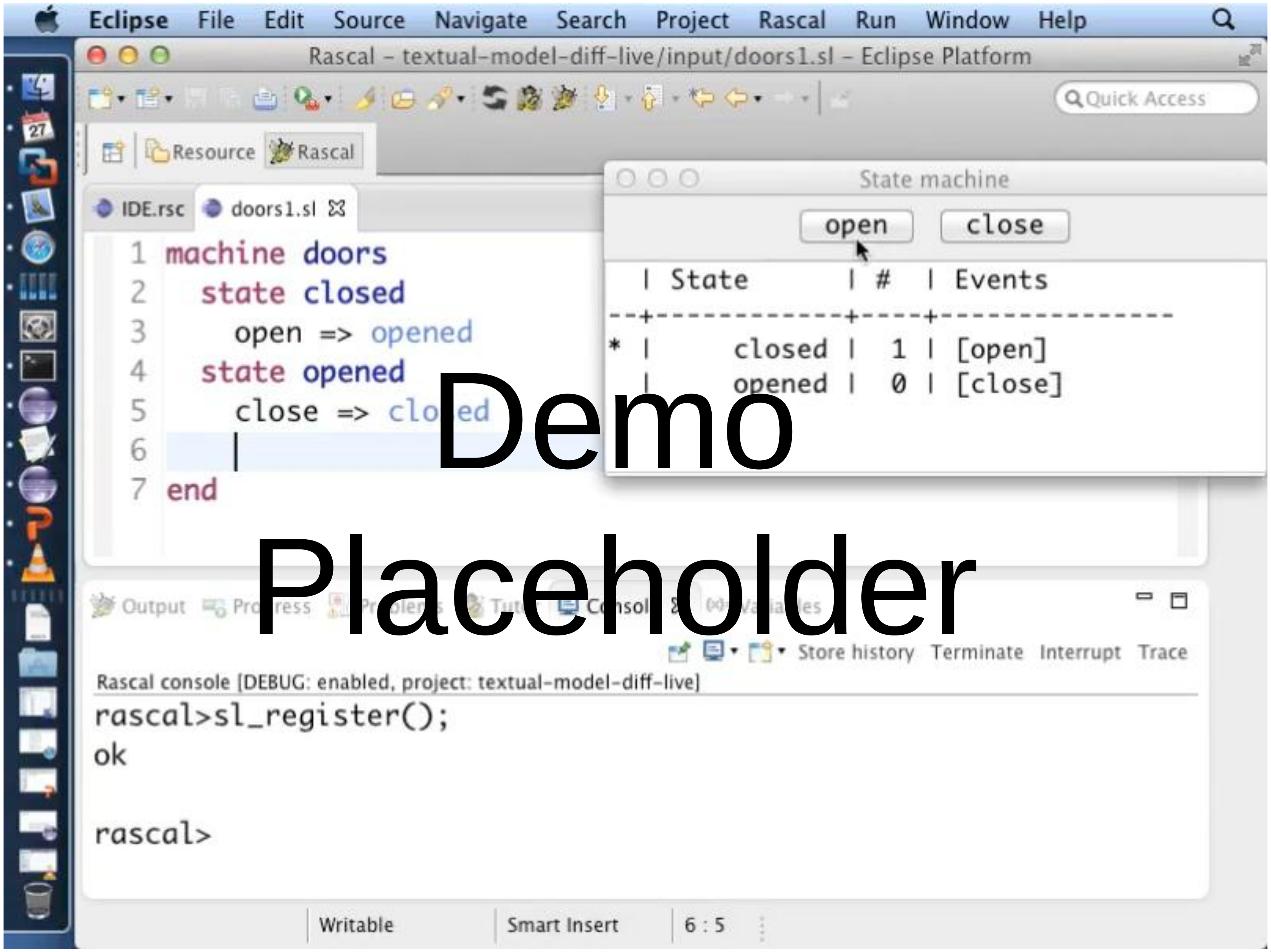


Live Textual Domain-specific Languages

- Bridge the *gulf of evaluation* between the edition of a model and its execution
- Live DSLs: Shorten the feedback loop between a model and its execution (avoid the edit-compile-restart cycle)
- The running model is updated instantly after every change to the model



Towards Live Domain-specific Languages: From text differencing to adapting models at run time
Riemer van Rozen, Tijs van der Storm
In *Software and Systems Modeling* (SoSyM), 2017



```
machine doors
  state closed
    open => opened
  state opened
    close => closed

end
```

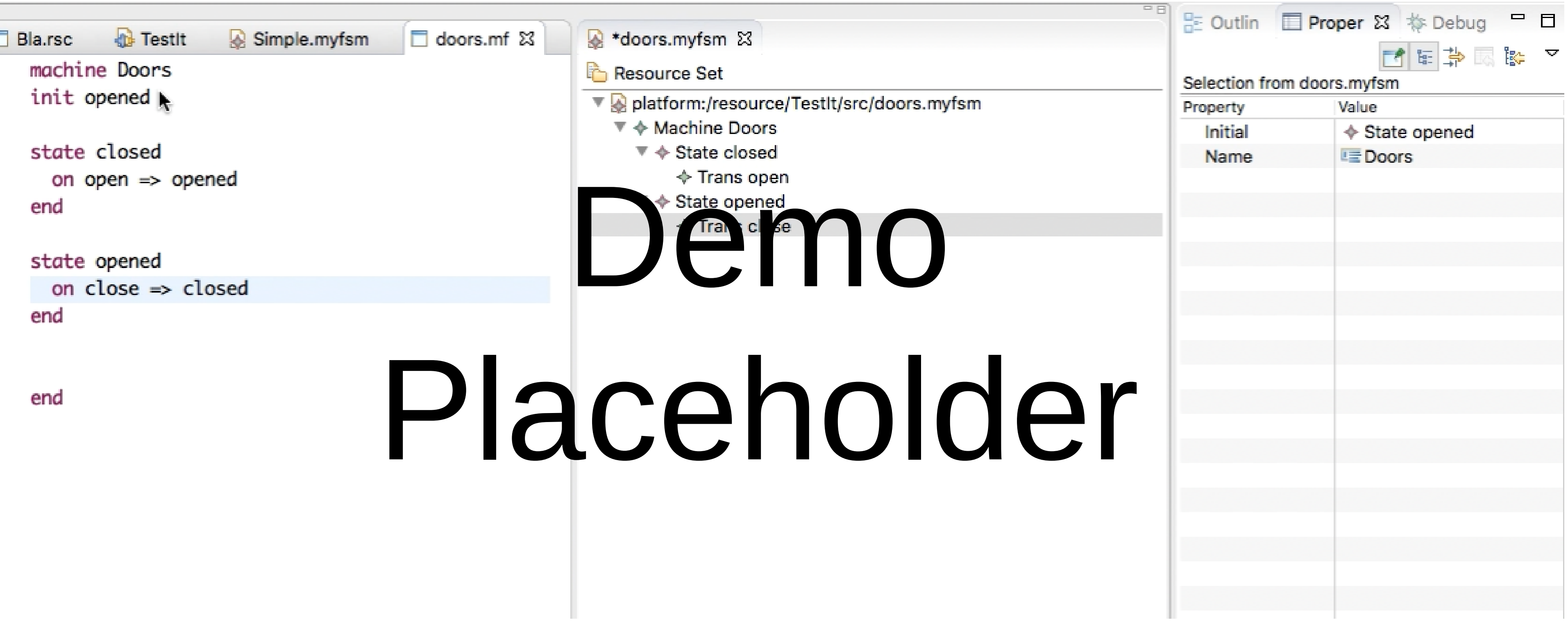
```
  | State      | #  | Events
--+------------+----+-------------
* |     closed |  1 | [open]
  |     opened |  0 | [close]
```

Rascal console [DEBUG: enabled, project: textual-model-diff-live]

```
rascal>sl_register();
ok

rascal>
```

Demo Placeholder

Ongoing: Bridging Technological Spaces



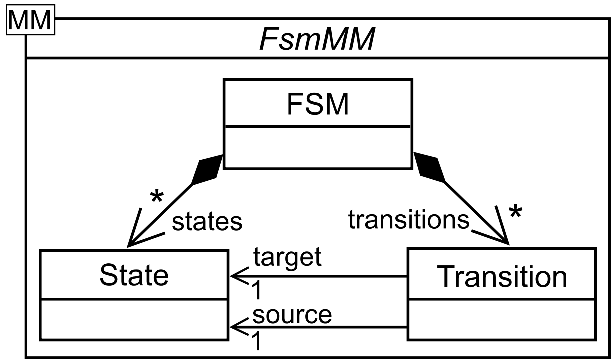
```
machine Doors
init opened

state closed
  on open => opened
end

state opened
  on close => closed
end


end
```

Resource Set
- platform:/resource/TestIt/src/doors.myfsm
  - Machine Doors
    - State closed
      - Trans open
    - State opened
      - Trans close

Selection from doors.myfsm

Property	Value
Initial	State opened
Name	Doors

```
data Machine =
    machine(str name, list[State] states);
data State =
    state(str name, list[Trans] transitions);
data Trans =
    trans(str event, Ref[State] to);
```



Future Work

- Incremental compilation is the first step towards the definition of *language modules*
- With proper provided/required interfaces
- Towards *Component-Based Software Language Engineering*
- As a support for Concern-Oriented Language Development (Manuel Leduc’s PhD @ DiverSE)

Thank you!

http://gemoc.org/ale/