



On Turning Domain Knowledge into Tools

Prof. Jean-Marc Jézéquel

Director of IRISA

jezequel@irisa.fr

<http://people.irisa.fr/Jean-Marc.Jezequel>



Writing Software
vs.
Creating a Scientific Theory

Scientific Theories & Models

- A scientific theory is an explanation of an aspect of the natural world that can be repeatedly tested, in accordance with the scientific method, using a predefined protocol of observation and experiment
 - "The Structure of Scientific Theories" in The Stanford Encyclopedia of Philosophy
- The scientific method involves
 - the proposal and testing of hypotheses,
 - by deriving predictions from the hypotheses about the results of future experiments
 - then performing those experiments to see whether the predictions are valid
- A Model is an abstraction of an aspect of the world for a specific purpose. Therefore a Scientific Theory is a Model.
 - But a Model is not always a Scientific Theory



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Creating a Scientific Theory is (evermore) Writing Software

- Mathematics used to be the language of science...
 - ... when science was simple enough
 - Newton's gravity
 - 2 bodies problem has an analytical solution (Maths)
 - 3+ bodies problem?
 - Solution: model it into software and run it on a computer
 - aka Simulation
 - Idem for nuclear reactions, QCD, meteo, climate ...
 - Informatics is the language of science of the 21th
 - Of course Math still has a role to play



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



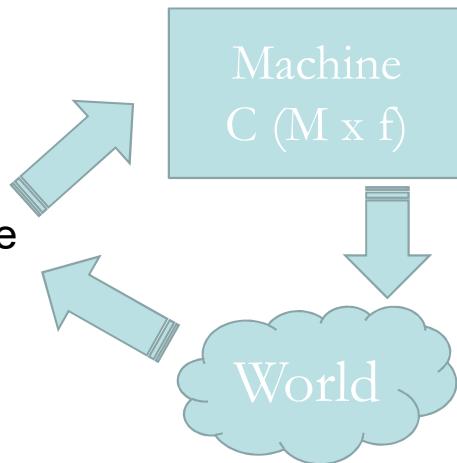
Conversely writing (usefull) Software is like Creating a Scientific Theory

➤ A Machine is made of

- A computer C
- Model M
- Function f } Software

➤ Does it do what I want?

- Test it w.r.t. the World!



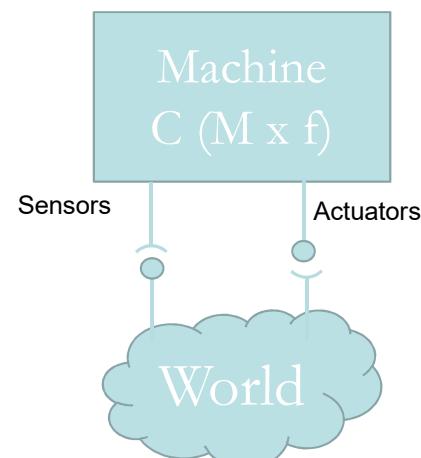
The World and the Machine [Jackson]

➤ A Machine is made of

- A computer C
- Model M
- Function f } Software

➤ Typical evolution

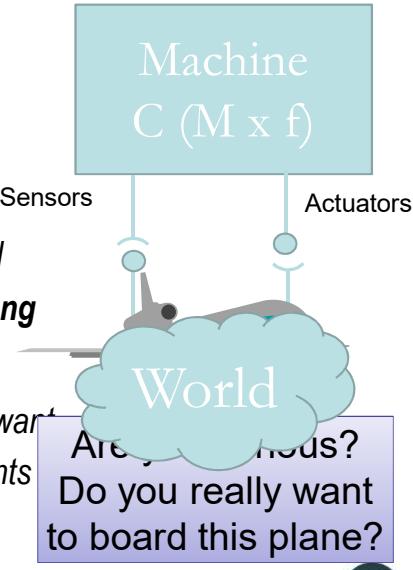
1. Model the world
2. Monitor the world
3. Control the world
4. Sometimes *becomes* the world (but that's not the point of this talk)
 - Bank accounts, Expedia...



Caveat

➤ A Machine is made of

- A computer C
 - *Nobody knows any longer how modern processors work*
- Model M
 - *Abstraction of an aspect of the world*
 - *it is incomplete, partial and thus wrong*
- Function f
 - *Users do not really know what they want*
 - *Many bugs traced to bad requirements*



Solution: Abstraction + Separation of Concerns

➤ Of course [Dijkstra]!

- Abstraction on hardware
- Models are SoC + abstraction of the world
- Functions must be understood => abstracted

➤ Are all these abstractions consistent?

- Do the thing right [Brooks]: applied maths, eg. Proofs

➤ Are they close enough to reality for the purpose?

- Do the right thing [Brooks]: Test!

Evolution towards better abstractions

➤ The historical approach (50's->80's)

- Machine = C(M x f) : M x f is « compiled »
 - Typical in Fortran, C, control automation, ...
 - Most efficient, but no SoC thus brittle wrt f->P

➤ The object oriented revolution (70's -> 2000's)

- Machine = C(M) x C(f) : M x f is « interpreted » (M still there)
 - Then it makes it easy to have Machine' = C(M) x C(f')
- Still hard to keep model separated from technical concerns
 - persistency, security, FT, speed...

➤ One DSL (Domain Specific Language) per concern (90's -> ?)

- Machine = C(M1) x C(M2) x C(M3) x C(f1) x C(f2) ...



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



**Writing software
is (will be)
composing abstractions
described in different languages**

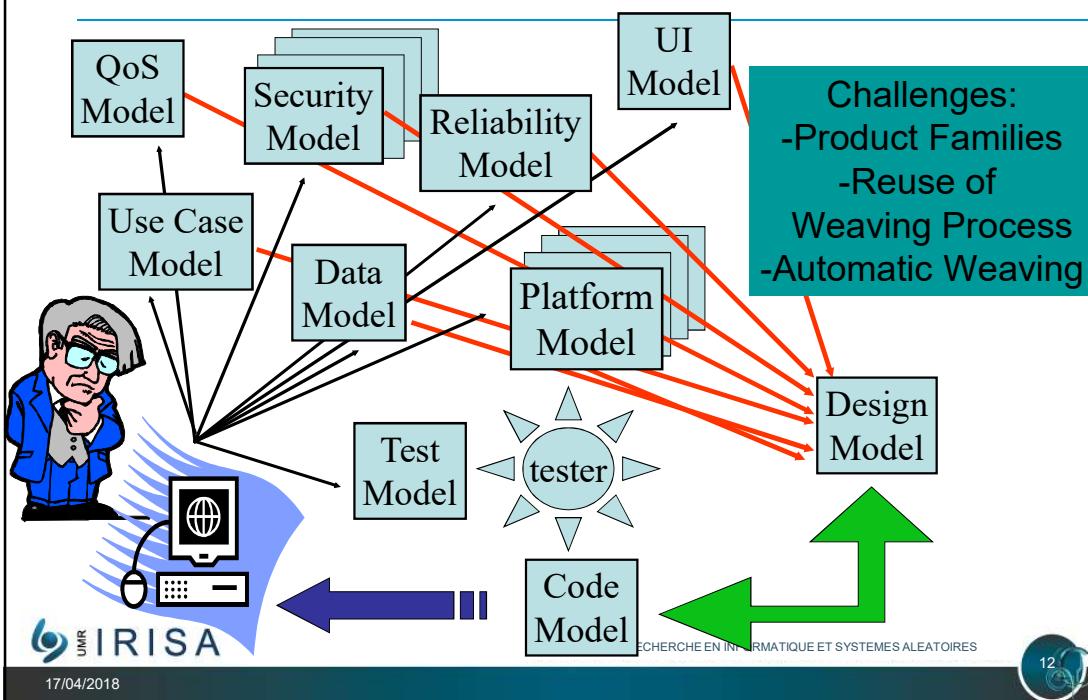


17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Modeling and Weaving



Complex Software Intensive Systems

- Multiple viewpoints & stakeholders
- Multiple concerns (technical...)
- Multiple domains of expertise
- => Need languages to express them!
 - In a meaningful way for experts
 - With tool support (analysis, code gen., V&V..)
 - Which is still costly to build
 - At some point, all these concerns must be *integrated*

Example: jHipster



- jHipster is a development platform to generate, develop and deploy Spring Boot + Angular Web applications and Spring microservices.
- Goal is to generate a complete and modern Web app or microservice architecture, unifying:
 - A high-performance and robust Java stack on the server side with Spring Boot
 - A sleek, modern, mobile-first front-end with Angular and Bootstrap
 - A robust microservice architecture with jHipster Registry, Netflix OSS, ELK stack and Docker
 - A powerful workflow to build your application with Yeoman, Webpack/Gulp and Maven/Gradle
- Use of 40+ different DSLs!



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Limits of General Purpose Languages (1)

- Abstractions and notations used are not natural/suitable for the stakeholders
 - Even with the best languages, impossible to keep all concerns separated down to the implementation



17/04/2018

```

if (newGame) resources.free();
s = FILENAME + 3;
setLocation(); load(s);
loadDialog.process();

try { setGamerColor(RED); }
catch(Exception e) { reset(); }
while (notReady) { objects.make(); }
if (resourceNotFound) break;

byte result; // Сменить на int!
music();
System.out.print("");

```



SYSTEMES ALÉATOIRES



Limits of General Purpose Languages (2)

- Not targeted to a **particular** kind of problem, but to any kinds of software problem.



GLP



DSLs

General-Purpose Languages

« Another lesson we should have learned from the recent past is that the development of 'richer' or 'more powerful' programming languages was a mistake in the sense that these baroque monstrosities, these conglomerations of idiosyncrasies, are really unmanageable, both mechanically and mentally.

I see a great future for very systematic and very modest programming languages »

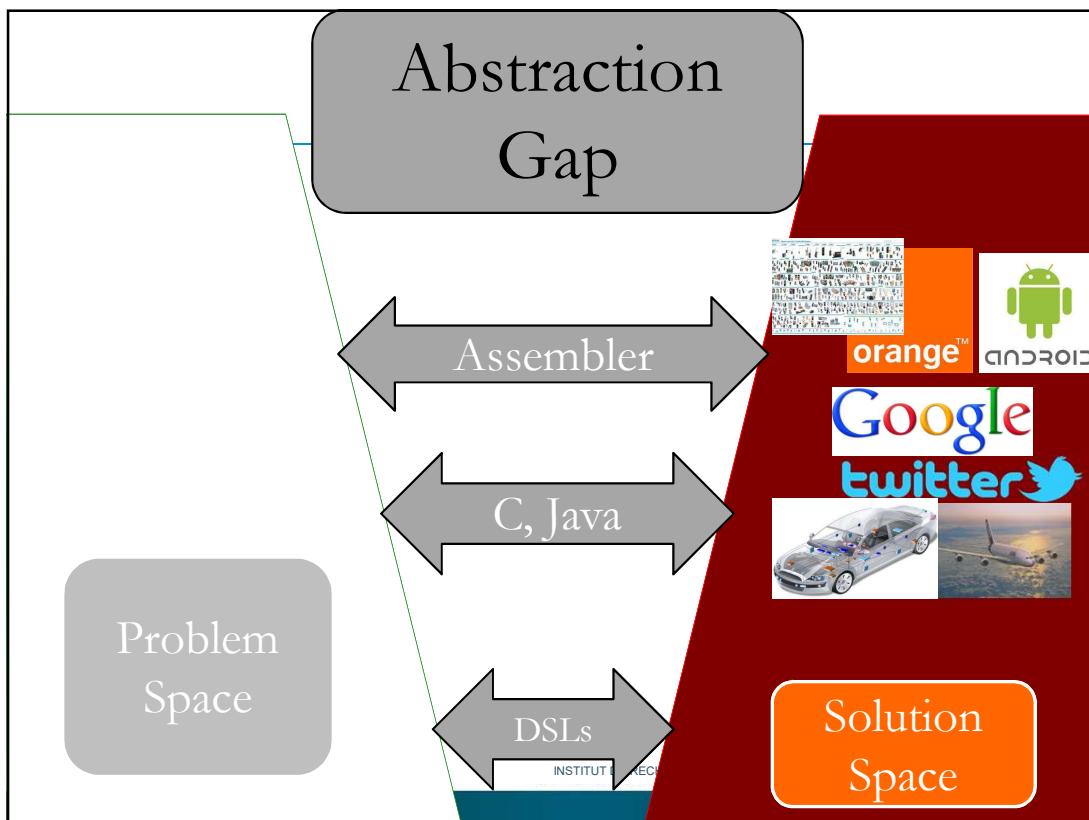
1972

ACM Turing Lecture,
« The Humble Programmer »
Edsger W. Dijkstra

aka Domain-Specific Languages

Domain Specific Languages

- Targeted to a **particular** kind of problem
 - with dedicated notations (textual or graphical), support (editor, checkers, etc.)
- Promises: more « efficient » languages for resolving a set of specific problems in a domain
- Each concern described in its own language => reduce abstraction gap



Domain Specific Languages (DSLs)

- Long history: used for almost as long as computing has been done.
- You're using DSLs in a daily basis
 - Even if you do not recognize them as DSLs (yet), because they have many different forms [Fowler]
- More and more people are building DSLs
 - How can we help them?



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xml:lang="en" lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>My first Web page.</p>
  </body>
</html>
```

Domain: web (markup)



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



CSS

```
.CodeMirror {
    line-height: 1;
    position: relative;
    overflow: hidden;
}

.CodeMirror-scroll {
    /* 30px is the magic margin used to hide the element's real scrollbars */
    /* See overflow: hidden in .CodeMirror, and the paddings in .CodeMirror-sizer */
    margin-bottom: -30px; margin-right: -30px;
    padding-bottom: 30px; padding-right: 30px;
    height: 100%;
    outline: none; /* Prevent dragging from highlighting the element */
    position: relative;
}
.CodeMirror-sizer {
    position: relative;
}
```



17/04/2018

Domain: web (styling)

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Makefile

```
PACKAGE      = package
VERSION      = `date "+%Y.%m%d%" `
RELEASE_DIR  = ..
RELEASE_FILE = $(PACKAGE)-$(VERSION)

# Notice that the variable LOGNAME comes from the environment in
# POSIX shells.
#
# target: all - Default target. Does nothing.
all:
    echo "Hello ${LOGNAME}, nothing to do by default"
    # sometimes: echo "Hello ${LOGNAME}, nothing to do by default"
    echo "Try 'make help'"

# target: help - Display callable targets.
help:
    egrep "^# target:" [Mm]akefile

# target: list - List source files
list:
    # Won't work. Each command is in separate shell
    cd src
    ls

    # Correct, continuation of the same shell
    cd src; \
    ls
```



17/04/2018

Domain: software building

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Regular expression

```
<TAG\b[^>]*>(.*?)</TAG>
```



Domain: strings (pattern matching)

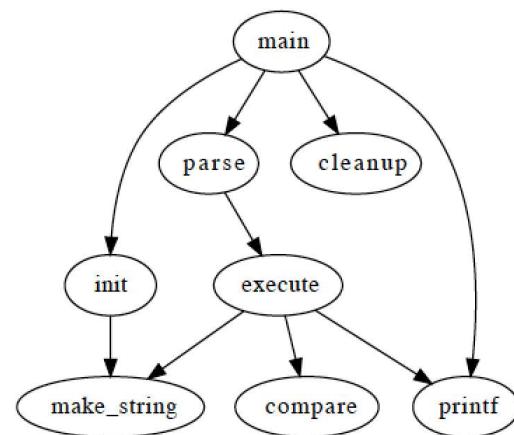
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES

17/04/2018



Graphviz

```
digraph G {
    main -> parse -> execute;
    main -> init;
    main -> cleanup;
    execute -> make_string;
    execute -> printf;
    init -> make_string;
    main -> printf;
    execute -> compare;
}
```



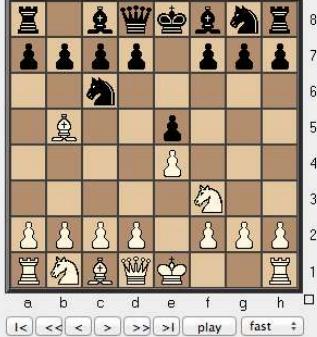
Domain: graph (drawing)

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES

17/04/2018



PGN (Portable Game Notation)

[Event "F/S Return Match"]
 [Site "Belgrade, Serbia Yugoslavia|JUG"]
 [Date "1992.11.04"]
 [Round "29"]
 [White "Fischer, Robert J."] 
 [Black "Spassky, Boris V."] 
 [Result "1/2-1/2"]

```
1. e4 e5 2. Nf3 Nc6 3. Bb5 {This opening is called the Ruy Lopez.} 3... a6  

  4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8 10. d4 Nbd7  

  11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nbl h6 16. Bh4 c5 17. dxе5  

  Nxе4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6  

  23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxе7 Rxе1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5  

  hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxh4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5  

  35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6  

  Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



17/04/2018

Domain: chess (games)

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



R: a DSL for statisticians

- a dynamic, lazy, functional, object-oriented programming language
 - with a rather unusual combination of features
- designed to
 - ease learning by non-programmers
 - enable rapid development of new statistical methods.
- base is estimated to be as high as 2 million users.



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



SQL

```

SELECT Book.title AS Title,
       COUNT(*) AS Authors
  FROM Book
 JOIN Book_author
    ON Book.isbn = Book_author.isbn
 GROUP BY Book.title;

INSERT INTO example
(field1, field2, field3)
VALUES
('test', 'N', NULL);

```



17/04/2018

Domain: database (query)

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



There are more DSLs than you think...

... because DSLs exist with different *shapes*



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Different shapes for a DSL: External

- External DSLs with their own syntax and domain-specific tooling
 - Nice for the non-programmers
 - Good for separation of concerns
 - Bad for integration
- Example: SQL

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue;
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```



17/04/2018

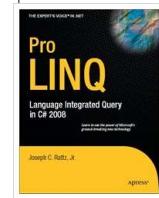
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Different shapes for a DSL: Internal/Embedded

- Internal/Embedded DSLs, blending their syntax and semantics into host language (C++, Scala, C#)
 - Splendid for the gurus
 - Hard for the rest of us
 - Excellent integration
- Example: SQL in LINQ/C#

```
// DataContext takes a connection string
DataContext db = new DataContext("c:\\northwind\\northwnd.mdf");
// Get a typed table to run queries
Table<Customer> Customers = db.GetTable<Customer>();
// Query for customers from London
var q =
  from c in Customers
  where c.City == "London"
  select c;
foreach (var cust in q)
  Console.WriteLine("id = {0}, City = {1}", cust.CustomerID, cust.City);
```



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Different shapes for a DSL: Implicit [Fowler]

- Implicit = from plain-old API to more fluent APIs
 - Good for Joe the Programmer
 - Bad for separation of concerns, V&V
 - Good for integration
- Example: SQL

```
Connection con = null;

// create sql insert query
String query = "insert into user values(" + student.getId() + ","
+ student.getFirstName() + "','" + student.getLastName()
+ "','" + student.getEmail() + "','" + student.getPhone()
+ ")";

try {
    // get connection to db
    con = new CreateConnection().getConnection("che
        "root");

    // get a statement to execute query
    stmt = con.createStatement();

    // executed insert query
    stmt.execute(query);
    System.out.println("Data inserted in table !");
} catch (SQLException e) {
    e.printStackTrace();
}

Result<Record> result =
create.select()
    .from(T_AUTHOR.as("a"))
    .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
    .where(a.YEAR_OF_BIRTH.greaterThan(1920))
    .and(a.FIRST_NAME.equal("Paulo")))
    .orderBy(b.TITLE)
    .fetch();
```



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



C: a DSL for System Programming?

- Hum, wait a minute, is not C a GPL?
 - Yes, but where it is really good is when you write low level stuff
- But what is the difference between a GPL and a DSL then?
 - Turing incompleteness not considered anymore as a criteria
 - Not always black & white, large greyscale



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



33

Where does Software go with DSL?

- A DSL program is a 3D abstraction
 - from the domain (cf Newton vs Relativity),
 - from the function (requirements)
 - from the platform
- Embrace uncertainty
 - Smaller abstractions than with GPL
 - We better know and control the unknown
 - Apply rigorous methods to uncertain systems so that we get known uncertainties



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



DSLs for the masses:

Helping people turn domain knowledge
into tools



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES





Issues of DSL Engineering *DSL is Software!*

- Versions
- Variants
- Separation of concerns / Composition



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Versions of a DSL: a Typical Lifecycle

- Starts as a simple ‘configuration’ mechanism
 - for a complex framework, e.g.; video processing
- Grows more and more complex over time
 - `ffmpeg -i input.avi -b:v 64k -bufsize 64k output.avi`
 - Cf <https://www.ffmpeg.org/ffmpeg.html>
- Evolves into a more complex language
 - `ffmpeg config file`
 - A preset file contains a sequence of option=value pairs, one for each line, specifying a sequence of options. Lines starting with the hash (#) character are ignored and are used to provide comments.
- Add macros, if, loops, ...
 - might end up into a Turing-complete language!



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Variants of a DSL

➤ Abstract syntax variability

- functional variability
 - E.g. Support for super states in StateCharts
 - 50+ variants of StateCharts Syntax have been reported!

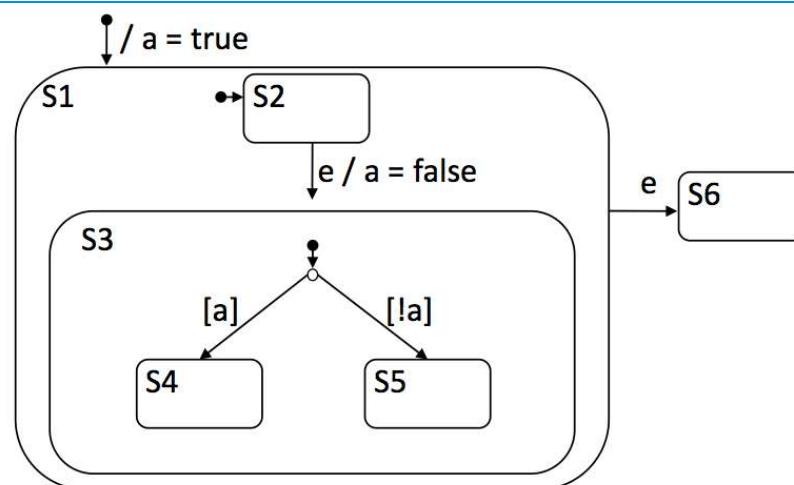
➤ Concrete syntax variability

- representation variability
 - E.g. Textual/Graphical/Color...

➤ Semantics variability

- interpretation variability
 - E.g. Inner vs outer transition priority

Variants Also at Semantic Level

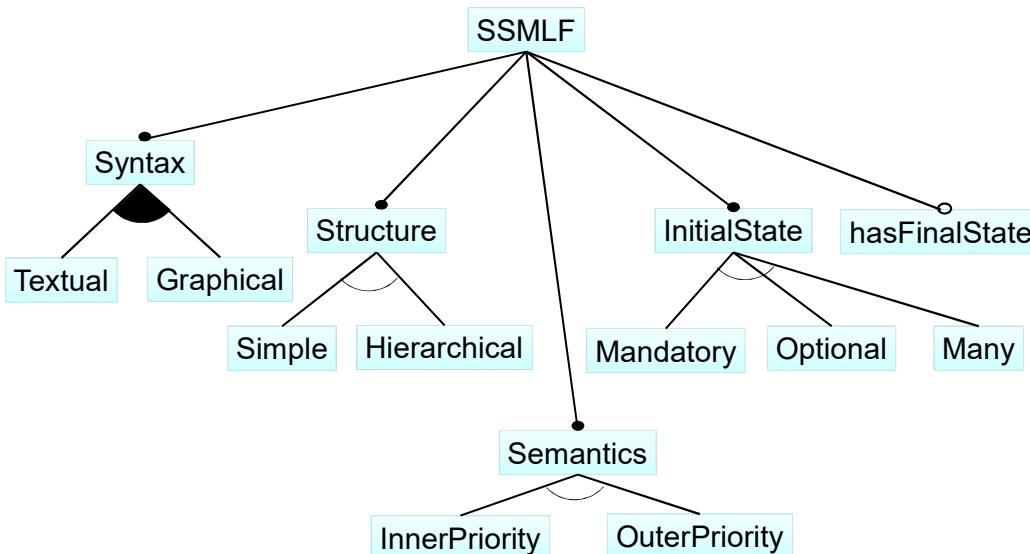


Event “e” leads to

S4 (UML), S5 (Rhapsody), or (S6) Stateflow

"UML vs. Classical vs. Rhapsody Statecharts: Not All Models are Created Equal ", Michelle Crane, Juergen Dingel

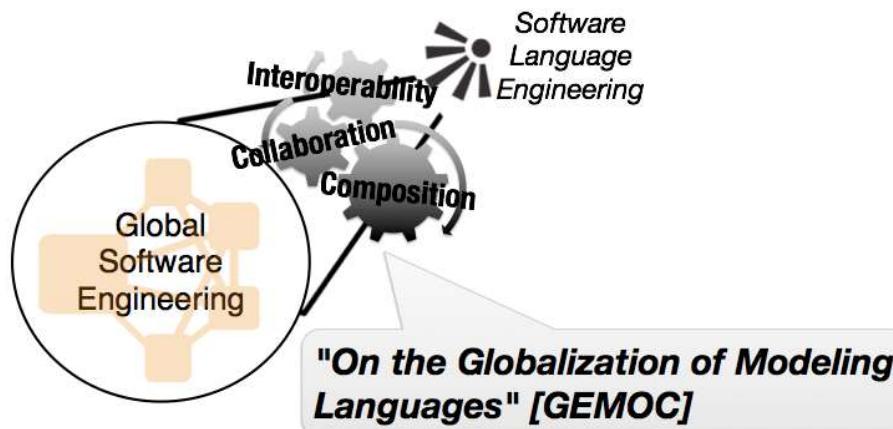
A (Simplified) State Machine Language Family



Gemoc Initiative

Visit <http://gemoc.org>

Focuses on SLE tools and methods for interoperable, collaborative, and composable modeling languages



DSL: From Craft to **Engineering**

➤ From supporting a single DSL...

- Concrete syntax, abstract syntax, semantics, pragmatics
 - Editors, Parsers, Simulators, Compilers...
 - But also: Checkers, Refactoring tools, Converters...

➤ ...To supporting Multiple DSLs

- Interacting altogether
- Each DSL with several flavors(variants)
- And evolving over time (versions)

➤ Product Lines of DSLs!

- Safe reuse of the tool chains?
- Backward compatibility, Migration of artifacts?



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Our Goal

• Ease the definition of tool-supported DSL families

- How to ease and validate the definition of new DSLs/tools?
- How to correctly reuse existing tools?

⇒ Bring external DSL design abilities to the masses

⇒ Use abstractions that are familiar to the OO
Programmer to define languages

⇒ set of DSL to build DSLs

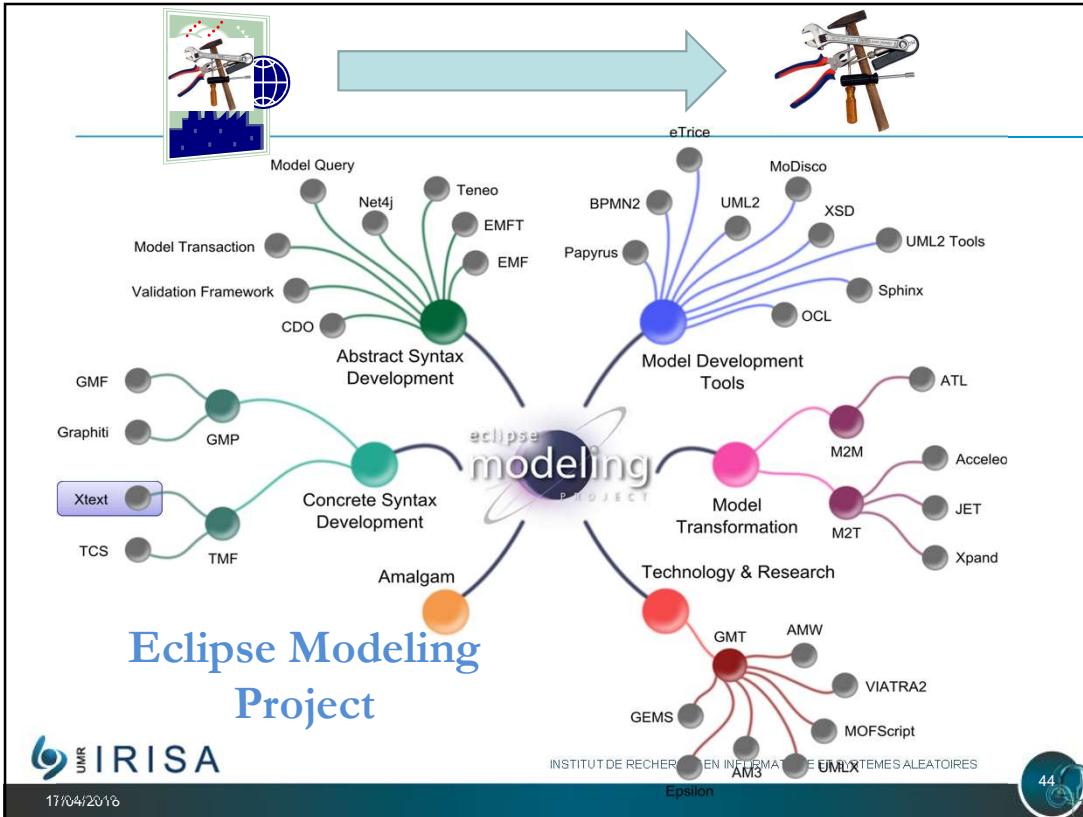
⇒ Leverage static typing to foster safe reuse
⇒ With an appropriate definition of type



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES





Tools built with MDE

➤ A tool (aka Model Transformation) is just a program working with specific OO data structures (aka meta-models) representing abstract syntax trees (graphs).

- Kermeta approach: organize the program along the OO structure of the meta-model
- Any software engineer can now build a DSL toolset!
 - No longer just for genius...

➤ Product Lines of DSLs = SPL of OO programs

- Safe reuse of the tool chains -> Static typing
- Backward compatibility, Migration of artifacts -> Adaption

IRISA

17/04/2018

45

Melange: a Meta-language for Modular and Reusable Development of DSLs

with Thomas Deguenni, Benoit Combemale, Arnaud Blouin, Olivier Barais

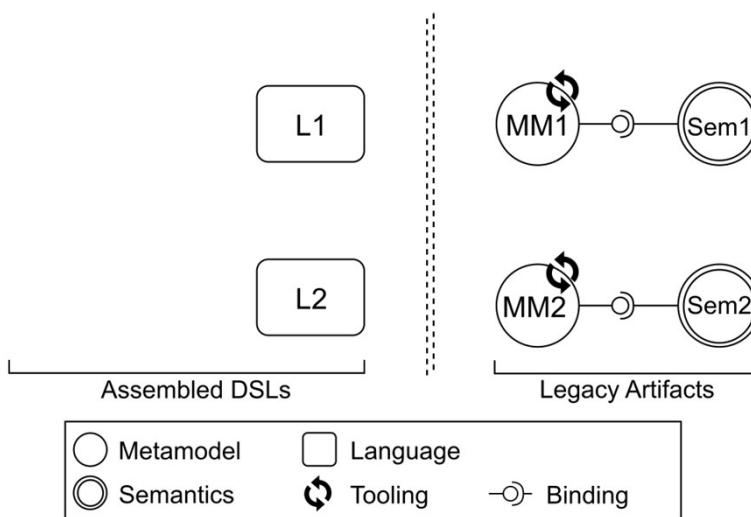


17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Approach Overview

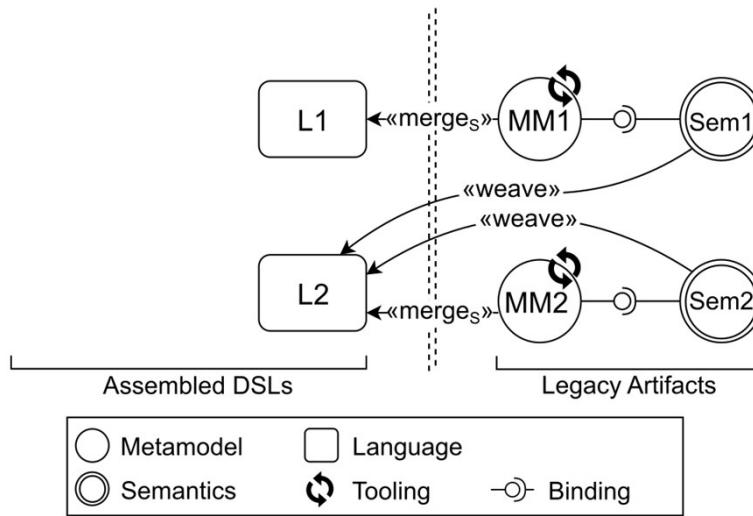


17/04/2018

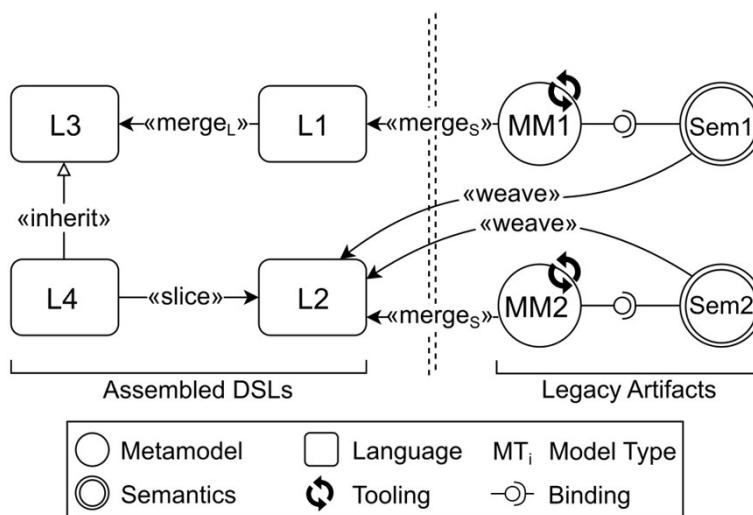
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Approach Overview

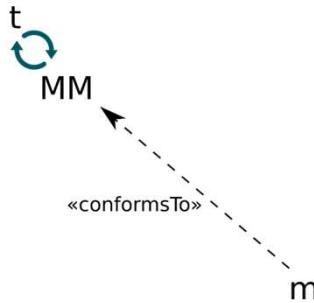


Approach Overview

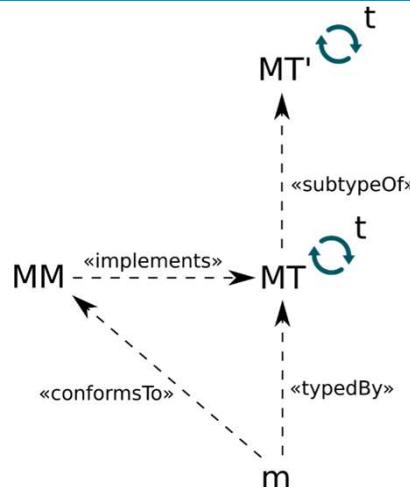


Inspired by eg. Erdweg et al., *Language Composition Untangled*, LDTA, 2012

TOOL REUSE THROUGH MODEL TYPING



TOOL REUSE THROUGH MODEL TYPING



Steel et al., On Model Typing, SoSyM, 2007
 Guy et al., On Model Subtyping, ECMFA, 2012

LANGUAGE DEFINITION

$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t$$

$$\forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j$$

$Sem \bullet Sem' \equiv Sem \frown Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem}^{\circ} sig(A_i^t)$$

$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle$$
 where

$$MT'' = MT \circ MT'$$
 and

$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

LANGUAGE DEFINITION

$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t$$

$$\forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j$$

$Sem \bullet Sem' \equiv Sem \frown Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem}^{\circ} sig(A_i^t)$$

$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle$$
 where

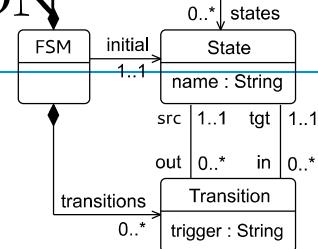
$$MT'' = MT \circ MT'$$
 and

$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$


language Fsm {
syntax 'Fsm.ecore',



}

LANGUAGE DEFINITION

$$\mathcal{L} \triangleq (AS, Sem, MT)$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_-(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

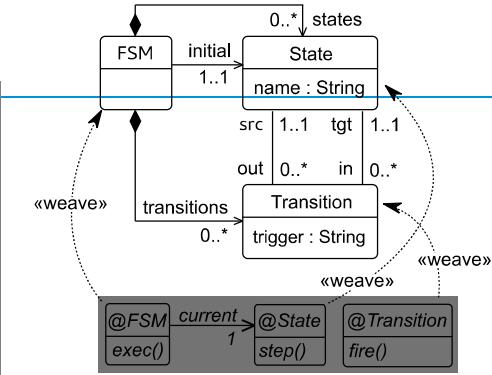
$$AS_2 \triangleq \lambda_-(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{ A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2 \},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language Fsm {
    syntax 'Fsm.ecore'
    with ExecutableFsm
    with ExecutableState
    with ExecutableTransition
}
  
```

LANGUAGE DEFINITION

$$\mathcal{L} \triangleq (AS, Sem, MT)$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_-(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

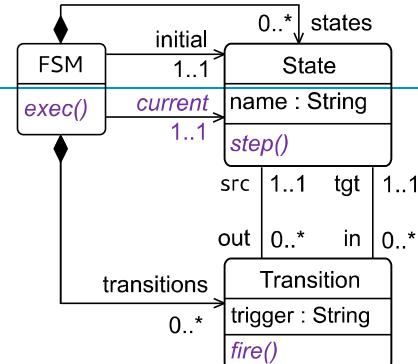
$$AS_2 \triangleq \lambda_-(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{ A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2 \},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language Fsm {
    syntax 'Fsm.ecore'
    with ExecutableFsm
    with ExecutableState
    with ExecutableTransition
    exactType FsmMT
}
  
```



SYNTAX MERGING

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \frown Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

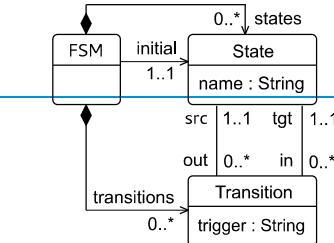
$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language GuardedFsm {
  syntax 'FSM.ecore'
  with ExecutableFsm
  with ExecutableState
  with ExecutableTransition

  exactType GuardedFsmMT
}
  
```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



SYNTAX MERGING

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \frown Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

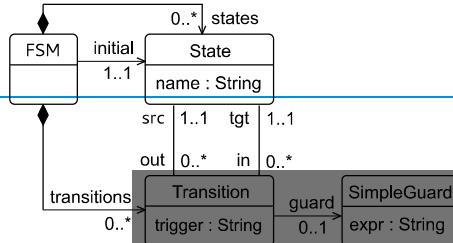
$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language GuardedFsm {
  syntax 'FSM.ecore'
  syntax 'Guard.ecore'
  with ExecutableFsm
  with ExecutableState
  with ExecutableTransition

  exactType GuardedFsmMT
}
  
```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



SEMANTICS WEAVING

$$\mathcal{L} \triangleq (AS, Sem, MT)$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT'$$

$$MT'' \lhd MT'$$

$\Lambda_-(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

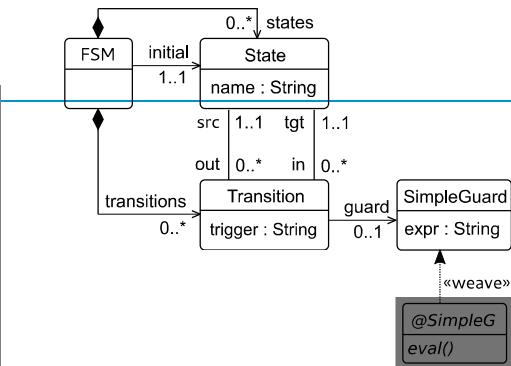
$$AS_2 \triangleq \lambda_-(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language GuardedFsm {
  syntax 'FSM.ecore'
  syntax 'Guard.ecore'
  with ExecutableFsm
  with ExecutableState
  with ExecutableTransition
  with EvaluateGuard
  → exactType GuardedFsmMT
}
  
```

SEMANTICS WEAVING

$$\mathcal{L} \triangleq (AS, Sem, MT)$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \ominus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT'$$

$$MT'' \lhd MT'$$

$\Lambda_-(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

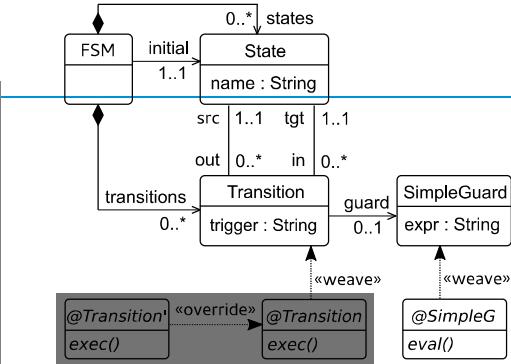
$$AS_2 \triangleq \lambda_-(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language GuardedFsm {
  syntax 'FSM.ecore'
  syntax 'Guard.ecore'
  with ExecutableFsm
  with ExecutableState
  with ExecutableTransition
  with EvaluateGuard
  with OverrideTransition
  → exactType GuardedFsmMT
}
  
```

LANGUAGE MERGING

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \uplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT'$$

$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

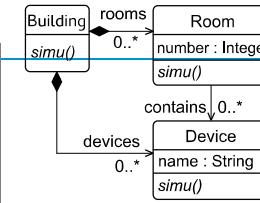
$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language Building {
    syntax 'Building.ecore'
    with SimulatorAspect...
}

exactType BuildingMT
}

```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



LANGUAGE MERGING

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \uplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT'$$

$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

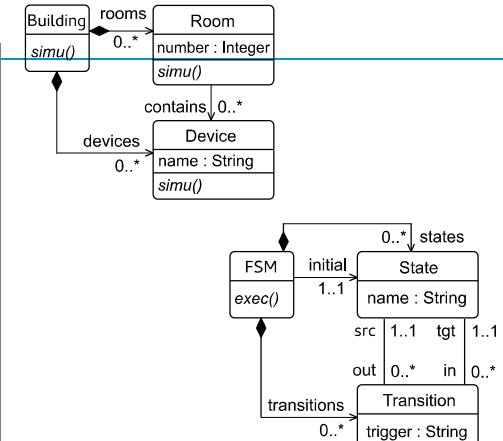
$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language Building {
    syntax 'Building.ecore'
    with SimulatorAspect...
    merge Fsm
}

exactType BuildingMT
}

```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



LANGUAGE MERGING

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \uplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

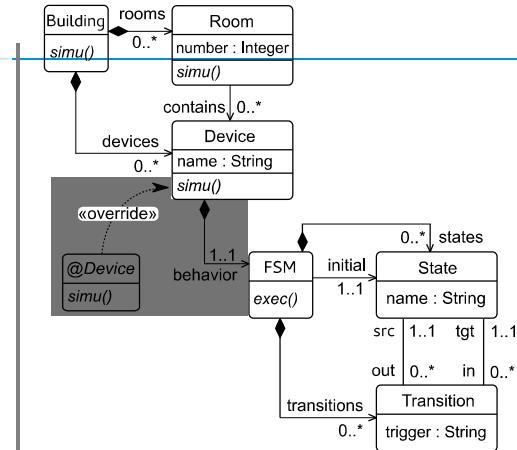
$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018



```

language Building {
  syntax 'Building.ecore'
  with SimulatorAspect...
  merge Fsm
  with GlueDeviceToFsm
  → exactType BuildingMT
}
  
```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



LANGUAGE INHERITANCE

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \uplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

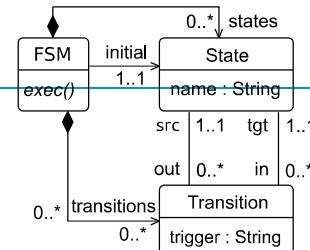
$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\}, \\ MT_1 \lhd MT_2,$$



17/04/2018



```

language TimedFsm inherits Fsm {
  exactType TimedFsmMT
}
  
```

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



163

LANGUAGE INHERITANCE

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

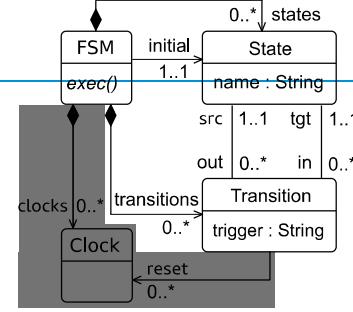
$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018

Melange: a Meta-language for
Modular and Reusable
Development of DSLs



language TimedFsm inherits Fsm {
syntax 'Clocks.ecore'

exactType TimedFsmMT
}



LANGUAGE INHERITANCE

$$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\begin{aligned} \forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t \\ \forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j \end{aligned}$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where}$$

$$MT'' = MT \circ MT' \text{ and}$$

$$MT'' \lhd MT'$$

$$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle, \text{ where:}$$

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

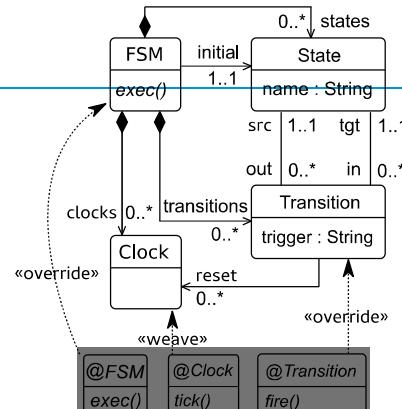
$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 \lhd MT_2,$$



17/04/2018

Melange: a Meta-language for
Modular and Reusable
Development of DSLs



language TimedFsm inherits Fsm {
syntax 'Clocks.ecore'
with ClockTick
with OverrideFsm
with OverrideTransition
exactType TimedFsmMT
}



LANGUAGE SLICING

$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t$$

$$\forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$

$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$

$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$

$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$

$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle$ where

$$MT'' = MT \circ MT'$$
 and

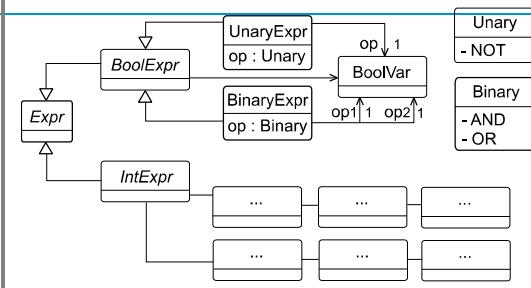
$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$

$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$

$MT_1 \lhd MT_2,$



```
language Expressions {
syntax 'Expressions.ecore'
with EvaluateBoolean
with EvaluateInteger
exactType ExpressionsMT
}
```



Melange: a Meta-language for
Modular and Reusable
Development of DSLs

17/04/2018



LANGUAGE SLICING

$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t$$

$$\forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j$$

$Sem \bullet Sem' \equiv Sem \sim Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$

$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$

$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$

$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$

$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle$ where

$$MT'' = MT \circ MT'$$
 and

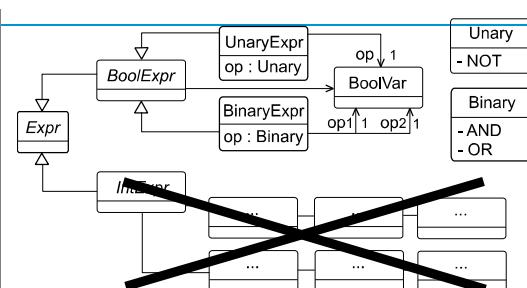
$$MT'' \lhd MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$

$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$

$MT_1 \lhd MT_2,$



```
language Expressions {
syntax 'Expressions.ecore'
with EvaluateBoolean
with EvaluateInteger
exactType ExpressionsMT
}
```

```
language BooleanExpressions {
slice Expressions on
['BoolExpr']
exactType BooleanExpressionsMT
}
```



Melange: a Meta-language for
Modular and Reusable
Development of DSLs

17/04/2018



LANGUAGE SLICING

$\mathcal{L} \triangleq \langle AS, Sem, MT \rangle$

$Sem(\mathcal{L}) \triangleq (A_i^t \in Aspects)$ where

$$\forall A_i^t \in Sem(\mathcal{L}), \exists c \in AS(\mathcal{L}) : c \text{ match } t$$

$$\forall A_i^t, A_j^t \in Sem(\mathcal{L}) : A_i^t \lhd A_j^t \implies i > j$$

$Sem \bullet Sem' \equiv Sem \setminus Sem'$

$$sig(Sem) \triangleq \bigcup_{A_i^t \in Sem} sig(A_i^t)$$

$MT(\mathcal{L}) \triangleq AS(\mathcal{L}) \circ sig(Sem(\mathcal{L}))$

$$\mathcal{L} \xleftarrow{m} AS' = \langle AS \circ AS', Sem, MT \circ AS' \rangle$$

$$\mathcal{L} \xleftarrow{w} Sem' = \langle AS, Sem \bullet Sem', MT \circ sig(Sem') \rangle$$

$$\mathcal{L} \uplus \mathcal{L}' = \langle AS \circ AS', Sem \bullet Sem', MT \circ MT' \rangle$$

$$\mathcal{L} \oplus \mathcal{L}' = \langle AS \circ AS', Sem' \bullet Sem, MT'' \rangle \text{ where } MT'' = MT \circ MT' \text{ and } MT'' <: MT'$$

$\Lambda_+^+(\mathcal{L}_1, c) = \langle AS_2, Sem_2, MT_2 \rangle$, where:

$$AS_2 \triangleq \lambda_+^+(AS_1, c), AS_2 \subseteq AS_1,$$

$$Sem_2 \triangleq \{A_i^t \in Sem_1, fp(A_i^t, AS_1) \subseteq AS_2\},$$

$$MT_1 <: MT_2,$$

```

classDiagram
    class Fsm {
        initial
        1..1 states
        0..* transitions
    }
    class State {
        name : String
    }
    class Transition {
        src : 1..1
        tgt : 1..1
        trigger : String
        guard : Expr
    }
    class Expr
    class BoolExpr
    class UnaryExpr {
        op : Unary
    }
    class BinaryExpr {
        op : Binary
        op1↑1
        op2↑1
    }
    class UnaryExp {
        eval()
    }
    class BinaryExp {
        eval()
    }

    Fsm "1..1" --> "0..*" State : initial
    Fsm "0..* --> Transition : transitions
    Transition "0..1" --> Expr : guard
    Expr "0..1" --> UnaryExpr : UnaryExpr
    Expr "0..1" --> BinaryExpr : BinaryExpr
    UnaryExpr "1" --> BoolVar : op1
    BinaryExpr "1" --> UnaryExp : op1↑1
    BinaryExpr "1" --> BinaryExp : op2↑1
  
```

```

language GuardedFsm inherits Fsm {
    with ...
    merge BooleanExpressions
    with AttachGuardToTransition
    exactType GuardedFsmMT
}
  
```

IRISA
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES
63

```

FsmFamily.melange
1 package fsmfamily
2
3@language Fsm {
4   ecore "FSM.ecore"
5   exactType FsmMT
6 }
7
8@language TimedFsm inherits Fsm {
9   exactType TimedFsmMT
10  with timedfsm.TimedTransitionAspect
11 }
12
13@language ExecutableFsm implements FsmMT {
14   ecore "ExecutableFSM.ecore"
15   exactType ExecutableFsmMT
16   with execfsm.ExecutableFSMSpect
17   with execfsm.ExecutableTransitionAspect
18   with execfsm.ExecutableStateAspect
19 }
  
```

```

FsmFamily.melange
22
23@transformation FsmMT createNewTimedFsm() {
24   val fact = TimedfsmFactory.eINSTANCE
25   return new TimedFsm {
26     contents += fact.createState => [
27       name = "S1"
28       outgoingTransition += fact.createTransition => [
29         input = "a"
30       ]
31     ]
32   ]
33 }
34
35@transformation flatten(FsmMT m) { /* ... */ }
36
37@transformation loadFsmModel() {
38   val m1 = Fsm.load("Lights.fsm")
39   val m2 = TimedFsm.load("Temporal.timedfsm")
40   flatten.call(m1)
41   flatten.call(m2)
42   val m3 = m2 as FsmMT
43   flatten.call(m3)
44 }
  
```

Outline

- fsmfamily
 - Fsm <| FsmMT, TimedFsmMT
 - TimedFsm <| Fsm <| FsmMT, TimedFsmMT
 - ExecutableFsm <| FsmMT, TimedFsmMT, ExecutableFSMSpect @ FSM
 - fsmt
 - execute
 - currentState
 - State
 - ExecutableTransitionAspect @ Transition
 - ExecutableStateAspect @ State
 - fsm
 - createNewTimedFsm
 - flatten
 - loadFsmModel
 - FsmMT <| TimedFsmMT
 - TimedFsmMT <| FsmMT

Melange

A Language Workbench

MELANGE

- An open-source (EPL) language workbench
- or... a language-based, model-oriented language for DSL engineering
- An implementation of the algebra
- Supported by a model-oriented type system
- Based on Xtext
- Seamlessly integrated with the EMF ecosystem
- Bundled as a set of Eclipse plug-ins



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Implementation Choices

- **Abstract syntax:** Ecore (EMOF)
- **Merging:** Customized UML PackageMerge¹
 - Trading UML specificities with EMOF specificities
 - Support for renaming
- **Slicing:** Kompren²
- **Operational semantics:** K3 (Xtend on steroids)



¹Bingel et al., *Understanding and Improving UML PackageMerge*, SoSyM, 2008

²Blouin et al., *Kompren: Modeling and generating model slicers*, SoSyM, 2012

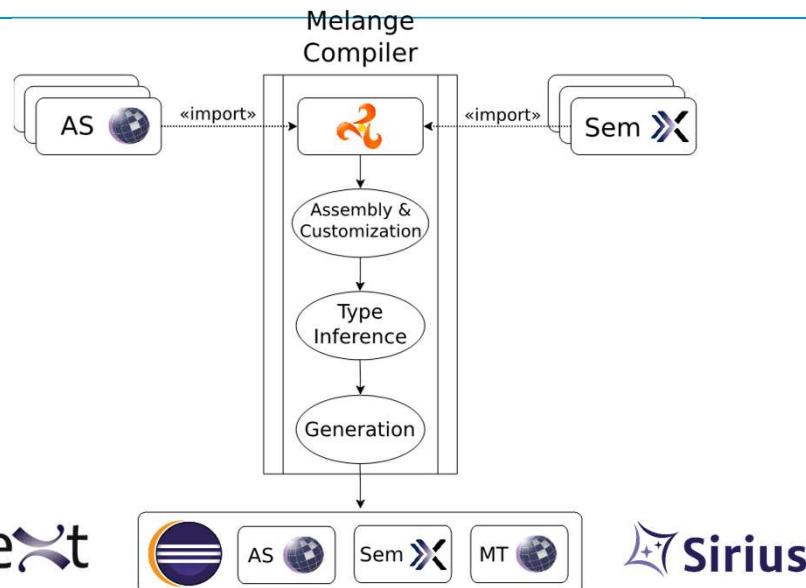


17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Compilation Scheme



IRISA

17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



OK, but does it work in practice?

Talk is cheap, show me the code!
[Linus Torvalds]

IRISA

17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES



Farming System Modeling (/ Software Defined Farming)

The screenshot shows the GEMOC Farming System Modeling interface. It includes:

- Model Explorer:** Shows the project structure with nodes like "MyExploitation", "src-gen", "analysis.scientific", "cultures.activities", "John exploitation", "representations.ard", and "schedule.simulation".
- Outline:** Displays a schedule for "John's Exp..." with tasks like "Labour scheduled on 13/apr", "SEMIS scheduled on 31/mar", and "RECOLTE scheduled on 21/apr".
- cultures.activities:** A list of activities for "culture corn":
 - activity LABOUR from 1 jan to 28 feb using 1 Tractor and 1 People
 - activity SEEDS From 15 mar to 15 apr [after LABOUR & no rain since 3 days & temp > 10°C] using 1 Tractor and 2 People
 - activity FERTILIZATION from 15 mar to 15 jun [after SEMIS & no rain since 30 days & no rain since 1 days] using 1 Tractor and 1 People
 - activity RECOLTE from 1 sept to 10 sept [grain is "mature" & after SEMIS] using 1 Tractor and 2 People
- *exploitation description:** A diagram illustrating the farming system with components like "tractor", "People", "wheat", "corn", "sheep", "Extra Water needed: 181600m³", "Massive Fertilizer", "John", "Henry", "Grazing", "Biomass Model 1.85", and "Biomass Model 1.85".
- *cortasks dependencies:** A dependency graph between tasks like "WATER", "FERTILIZER", "SEEDS", "LABOUR", "RECOLTE", and "RESULTS".
- *Hydro Data:** A table for "John's Exp..." showing data for "Rain", "Hyd", "Biomass", and "LAI" from March 21 to April 13.
- *Climate Data:** A table showing Rain (mm), Temperature (°C), and Rad (Joules/cm²) for months from April 7 to April 10.
- Properties:** A table showing properties for "Exploitation Analysis 60.0" and "Biomass Model 1.85".
- Resource Set:** A table showing resources A, B, Culture, Eb, Elmax, K, and Lmax.

Bottom:

- IRISA logo
- Link: <https://github.com/gemoc/farmingmodeling>
- Date: 17/04/2018
- Page number: 74

Water flood prediction

The screenshot shows a water flood prediction application using Padins. It includes:

- Padins:** A web-based interface for component-based development.
- Test File:** A flowchart with nodes: "Raw data", "Processing", and "Visualisation".
- List of available components:** Components include "Raw data", "Model", "Processing", "Simulation", and "Visualisation".
- Name:** A field for naming the component.
- Data available from the previous nodes:** A section stating "No data from previous nodes."
- Code:**

```

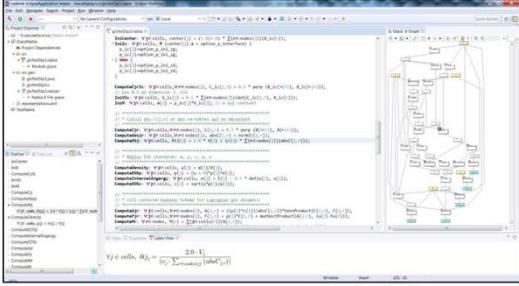
1 temperatures = [
2     [7, 6.9, 9.5, 14.5, 18.4, 21.5, 25.2, 26.5, 23.3, 18.3, 13.9, 9.6],
3     [3.9, 4.2, 5.7, 8.5, 11.9, 15.2, 17, 16.6, 14.2, 10.3, 6.6, 4.8]
4 ]
5
6 months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
7
8 # Use the function below to choose the data you want to send to the next nodes.
9 # Add each variable between the parenthesis. They will be accessible from the next nodes with
10 # the name you chose in the function call.
11 # Be careful not to override variables in the next nodes.
12 # If you do, use another identifier (foo, foo_toto-toto)
13 # will make a accessible on the next nodes with the name foo, so you can use
14 # as an already existing var, and toto will be accessible as toto
15 # !! To send a, you can't write (a), please use (ava)
16
17 sendTheDataToTheNodes(temperatures, hellomonths)
    
```
- Visualisation:** A chart titled "hello" showing monthly temperatures. The Y-axis ranges from 0 to 30, and the X-axis shows months from Jan to Dec. Data points are labeled with their values: (Jan, 3.9), (Feb, 4.2), (Mar, 5.7), (Apr, 8.5), (May, 11.9), (Jun, 15.2), (Jul, 17), (Aug, 16.6), (Sep, 14.2), (Oct, 10.3), (Nov, 6.6), (Dec, 4.8).
- Bottom:**

 - IRISA logo
 - INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES
 - Date: 17/04/2018
 - Page number: 75

High Performance Computing

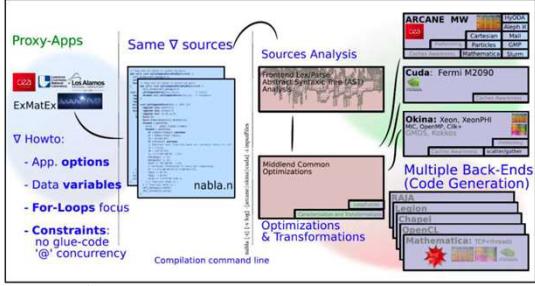
Nablab Environment

Technologies : Eclipse EMF, Xtext, Sirius
Contributors : MPO, BL, BC



Nabla Language

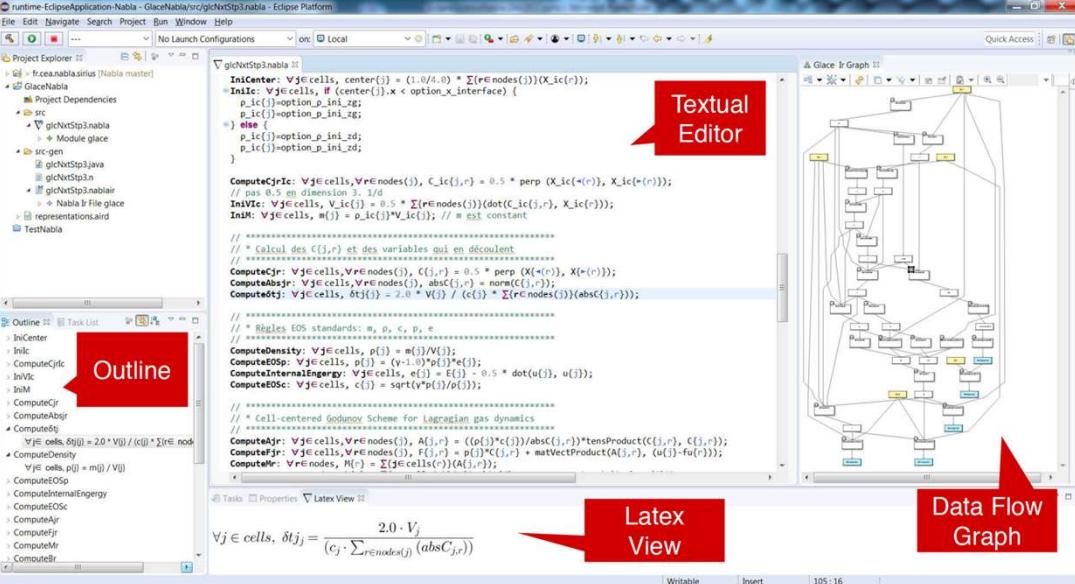
Technologies : Flex, Bison, C++
Contributors : JSC, BL



Nabla text files (.n)

 INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES
17/04/2018


High Performance Computing



The screenshot illustrates the GlaceNabla application within the Eclipse Platform. It features several views:

- Textual Editor:** Displays the Nabla source code for "glcNablaStp3.nabla". The code includes various mathematical operations and variable definitions.
- Data Flow Graph:** A graph visualization showing the data flow between different components or nodes in the program.
- Latex View:** A view showing the generated LaTeX code corresponding to the Nabla code.
- Outline:** A tree view showing the project structure and the contents of the Nabla files.

At the bottom, the IRISA logo and the date 17/04/2018 are visible.

Open Challenges

- ▶ **Diversity/complexity of DSL relationships**
 - ⌚ Far beyond structural/behavioral alignment, refinement, decomposition
 - ⌚ Separation of concerns vs. Zoom-in/Zoom-out
- ▶ **Live and collaborative (meta)modeling**
 - ⌚ Minimize the round trip between the DSL specification, the model, and its application (interpretation/compilation)
 - ⌚ Model experiencing environments (MEEs): what-if/for scenarios, trade-off analysis, design-space exploration
- ▶ **Integration of analysis and predictive models into DSL semantics**
 - ⌚ Towards unpredictable languages
 - ▶ Specify the correctness envelope to avoid over-specification
 - ▶ Identify plastic computation zones
 - ▶ Vary the execution flow of the program



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Conclusion

- **From supporting a single DSL...**
 - Concrete syntax, abstract syntax, semantics, pragmatics
 - Editors, Parsers, Simulators, Compilers...
 - But also: Checkers, Refactoring tools, Converters...
- **...To supporting Multiple DSLs**
 - Interacting altogether
 - Each DSL with several flavors: families of DSLs
 - And evolving over time
- **Product Lines of DSLs**
 - Share and reuse assets: metamodels and transformations



17/04/2018

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALEATOIRES



Acknowledgement

➤ All these ideas have been developed with my colleagues of the DiverSE team at IRISA/Inria



Formerly known as Triskell



17/04/2018

CHAPMAN & HALL/CRC INNOVATIONS IN
SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT

Engineering Modeling Languages

Turning Domain Knowledge into Tools



Benoit Combemale
Robert B. France
Jean-Marc Jézéquel
Bernhard Rumpe
Jim Steel
Didier Vojtisek

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

