What‘s new
in the MPS Ecosystem

voelter@acm.org
www.voelter.de
@markusvoelter
Markus Völter
itemis voelter ingenieurbüro für softwaretechnologie

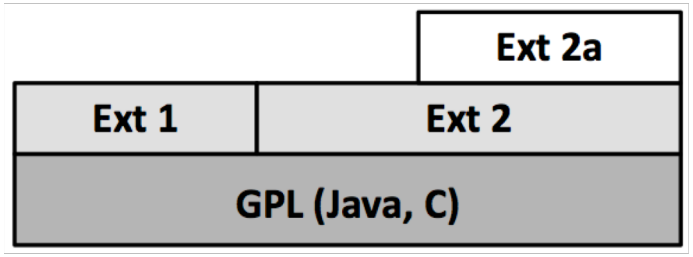
1

KernelF

DSL Development

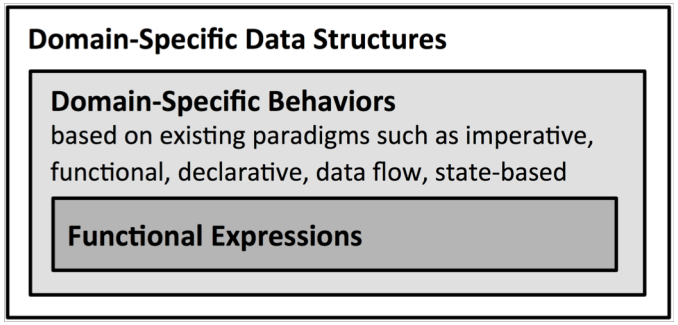
GPL Extension

Reuse GPL incl. Expressions and TS
Add/Embed DS-extensions
Compatible notational style
Reduce to GPL



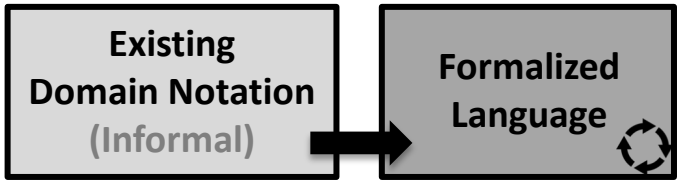
New Language

Analyze Domain to find Abstractions
Define suitable, new notations
Rely on existing behavioral paradigm
Reuse standard expression language
Interpret/Generate to one or more GPLs



Formalization

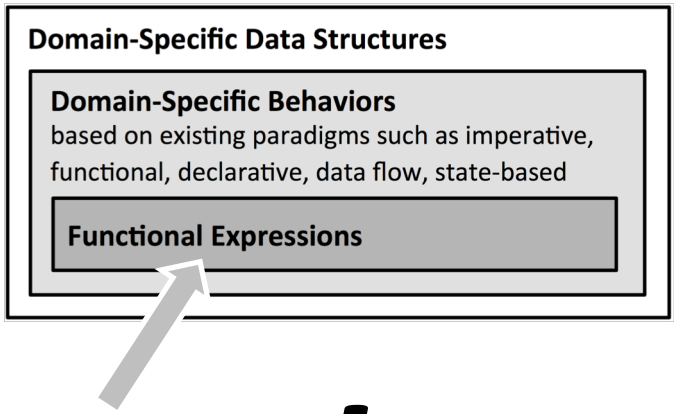
Use existing notation from domain
Clean up and formalize
Generate/Interpret
Often import existing „models“



DSL Development

New Language

Analyze Domain to find Abstractions
Define suitable, new notations
Rely on existing behavioral paradigm
Reuse standard expression language
Interpret/Generate to one or more GPLs



KernelF

Functional Features

Functional, no state at its core.

Purity + Effect Tracking

The usual types, literals and op's

Various Conditionals

Functions and Blocks

Error Handling `attempt<T|E-1,... E-n>`

```
try <e> => <s> error <E-1> => <e-1> ... error <E-n> => <e-n>
```

Immutable Collections and higher-order functions

Enums, tuples, records, all immutable

Constraints on types and functions

+ Algebraic Data Types

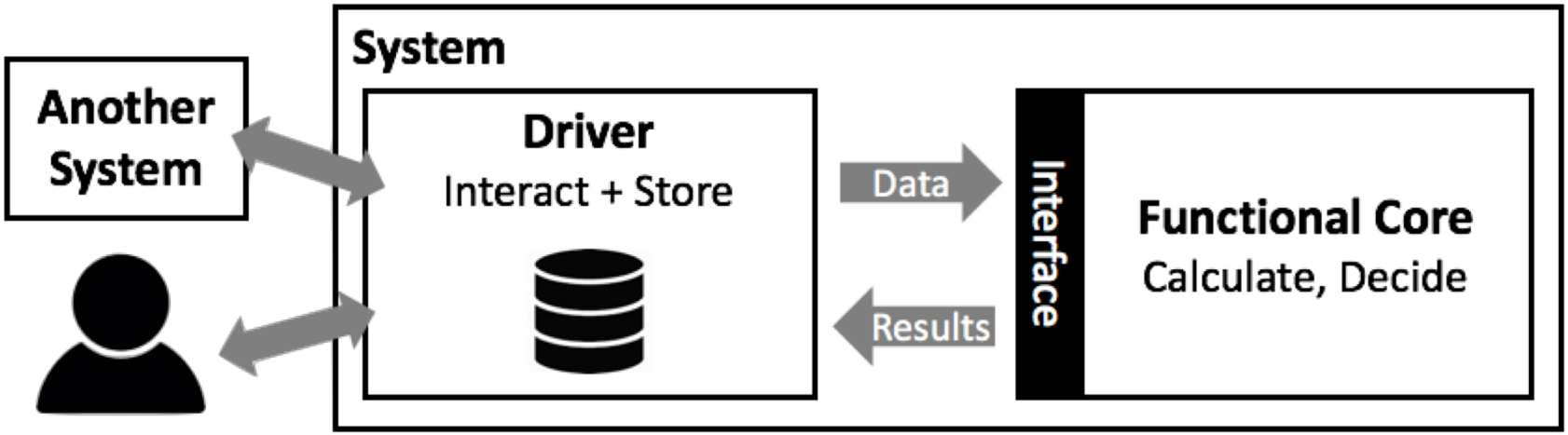
+ Datetime

+ Temporal Data

Functional Stuff only: is this useful?

A purely functional language only heats up the processor.

-- Anonymous



Functional Features

Functional, no state at its core.

Purity + Effect Tracking

The usual types, literals and op‘s

Various Conditionals

Functions and Blocks

Error Handling `attempt<T|E-1,... E-n>`

```
try <e> => <s> error <E-1> => <e-1> ... error <E-n> => <e-n>
```

Immutable Collections and higher-order functions

Enums, tuples, records, all immutable

Constraints on types and functions

Stateful Features

Boxes (like Clojure‘s `ref`)

Transactional Memory

State Machines

Interactors

Tooling: Live Execution

```
test suite PaperDescription        execute automatically  : false
                                   Only local declarations: true

val aBool: boolean = true
val anInt: number = 42
val aReal: number{2} = 33.33
val aString: string = "H"
fun f(n: number) where [post res < 10] = 2 * n

test case BasicOperators [fail] {
  assert  42 + 33                  equals 75      [1 ms]      <number[75|75]{0}>
  assert  42 + 2 * 3               equals 48      [1 ms]      <number[48|48]{0}>
  assert  aReal + anInt            equals 75.33   [1 ms]      <number[-∞|∞]{2}>
  assert  aBool && true            equals true    [0 ms]      <boolean>
  assert  aString + ", W"          equals "H, W"  [1 ms]      <string>
  confail f(20)
  assert  if aBool then 42 else 33 equals 44      actual: 42 <number[33|42]{0}>
}
```

Tooling: Coverage

```
assessment: InterpreterCoverage
query:      interpreter test coverage in current model
              problems only: ☐
              languages: language/org.iets3.core.expr.base/
              ignore:    {33 ignored concepts}
sorted: ☒    must be ok: ☐    hide ok ones: ☐
last updated: 2 minutes ago by markusvoelter
```

org.iets3.core.expr.base

TupleValue	Covered.
SomeValExpr	Covered.
LogicalImpliesExpression	Covered.
ErrorExpression	Partial. Missing: [generic]
SuccessExpression	Covered.
ParensExpression	Covered.
TryExpression	Partial. Missing: [fallthrough, generic]

total 38, **new** 38, **ok** 0

coverage 95 %

Tooling: Debugging

Trace Explorer assert { val v = getR().addr.first if v.street.length < 3 then "LessThan3" else "Bigger" }

- **assert** assert { val v = getR().addr.first if v.street.length < 3 then "LessThan3" else "Bigger" } [AssertTestIte
  - **actual:** { val v = getR().addr.first if v.street.length < 3 then "LessThan3" else "Bigger" } ⇒ **Bigger** : Strin
    - **val** v ⇒ **#Address[Oetz, #City[70327, St ...** : RecordValue (4 ms) [ValExpression]
    - if v.street.length < 3 then "LessThan3" else "Bigger" ⇒ **Bigger** : String [IfExpression]
      - **cond:** v.street.length < 3 ⇒ **false** : Boolean (1 ms) [LessExpression]
      - **else:** Bigger ⇒ **Bigger** : String [StringLiteral]
  - **expected:** getOetz() ⇒ **Oetz** : String [FunctionCall]

```
if v → #Address[Oetz, #City[70327, Stuttgart]].street → Oetz .length → 4 < 3 then "LessThen3" else "Bigger"
   → Oetz
   → 4
   → false
   → Bigger
```

Tooling: REPL

```
repl Time_repl_0 for node artificialclock(0) in Time
downstream updates ☐
Reevaluate All   Clear   Clear Tail   Close & Return

[0]  artificialclock(0)    : artificialclock, effects[reads]
         artificialclock:5
[1]  $0.advance(5)    : artificialclock, effects[modifies]
         artificialclock:5
[2]  $0.time    : number[0|∞], effects[reads]
         5
[3]  <no expression>
       <no result found>
```

Tooling: Test Case Generation

Random values + Extreme Values

```
test case TestFunctions [success] {
  vectors function add  -> producer: random 25
          results: true
```

	a	a2	b	c	s	res	status
0: valid	3	-3.06	false	BLUE	""	3	ok
1: valid	2	2.74	false	BLUE	"M/Yh-0I/ac"	2	ok
2: valid	1	0.22	false	BLUE	""	1.22	ok
3: invalid input	4	-0.45	true	BLUE	"7l:6h7sg!afLmULGU8wtI00H9"		not executed
4: invalid input	1	1.38	false	RED	"Mtoa7J66uuWTye2f2-fLhD§hj8C2K"		not executed
5: invalid input	1	-2.63	false	BLUE	"n66r7E (f0J$aQMj§"		not executed

```
vectors function plus -> producer: eqclass
        results: true
```

	a	b	c	res	status
0: valid	0	-10	GREEN	-10	ok
1: valid	0	-10	BLUE	-10	ok
2: valid	0	10	GREEN	10	ok
3: valid	0	10	BLUE	10	ok
4: valid	0	-1	GREEN	-1	ok

	a	b	status
0: valid	7	2	no expected value given; actual was 14
1: valid	1	8	no expected value given; actual was 8
2: valid	5	2	[POST] res == a * b
3: valid	7	8	no expected value given; actual was 56
4: valid	5	6	[POST] res == a * b
5: valid	8	0	[PRE] b > 0
6: valid	5	8	[POST] res == a * b

Tooling: Mutation Testing I

```
fun doodle(a: number[1|5]) = if true then a else a * 1

fun add(a: number[1|5]) = alt [ a > 3      => a + 1
                                otherwise => doodle(a) ]

test case TestFunctions [incomplete] {
  vectors function add                          -> producer: random 30
          results: true                            { 30 entries }
          mutator: # of mutations 20
                   keep all:      false
                     -> ParensExpression
                     -> NumberLiteral
                     -> LogicalNotExpression
                     -> ParensExpression
                     -> ParensExpression
}
```

Tooling: Mutation Testing II

```
fun add(a: number[1|5]) = alt [ ((a) + 1) > 3 => a + 1
                                a
                                otherwise     => doodle(a) ]

fun doodle(a: number[1|5]) = if true then a else a * 2
                                                    1

fun doodle(a: number[1|5]) = if true then a else ((a * 1) + 1)
                                                 a * 1

fun doodle(a: number[1|5]) = if !(true) then a else a * 1
                                true
```

Example Kf Extensions

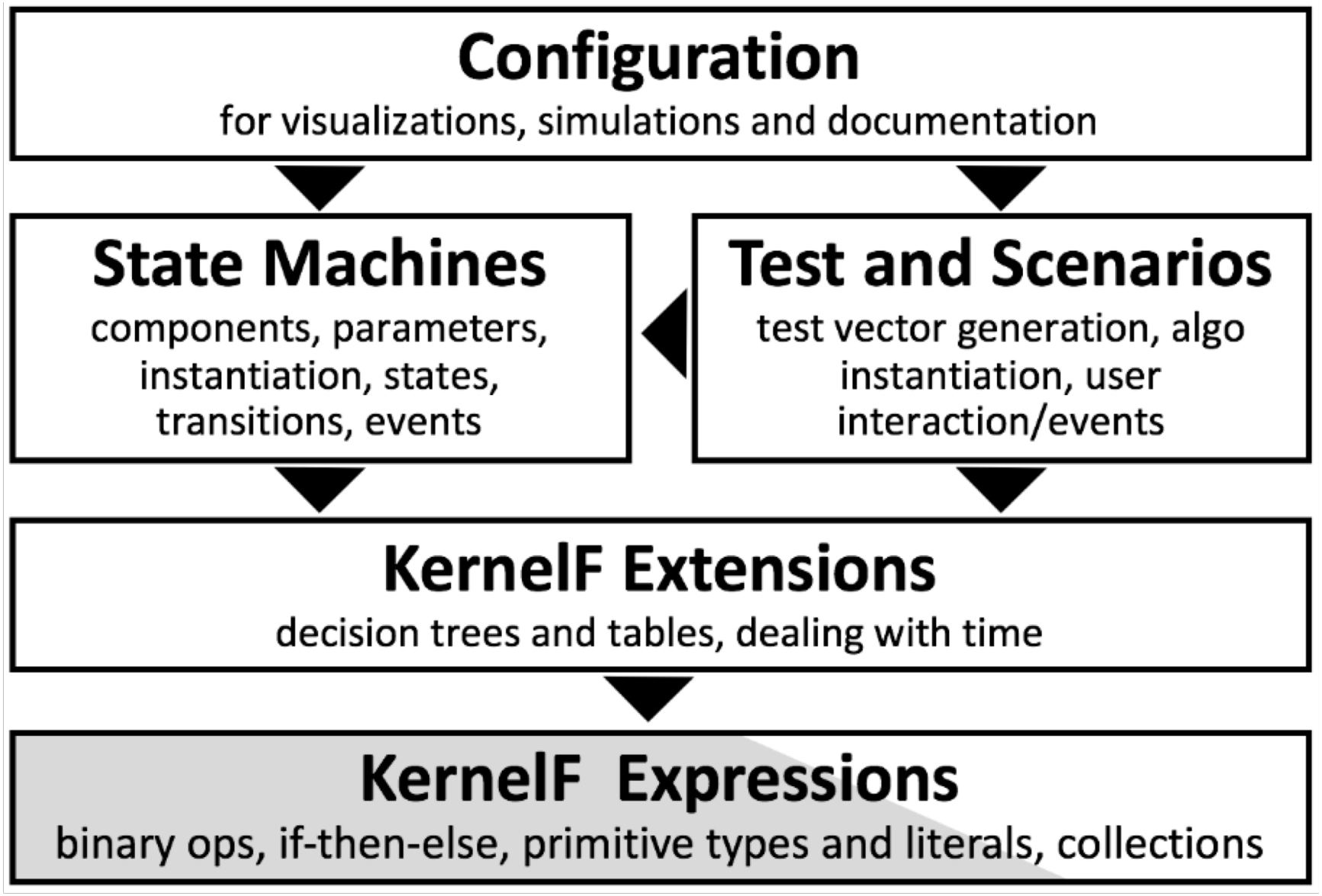
Used in Systems Engineering, Healthcare, Finance and Smart Contracts (Experimental)

	Salary/Tax	Smart Contracts	Healthcare
IDE	Alternative Rules Rules for Data Items Projection for a given Date	Live Values Diff of Live Values	Simulator with Phone UI Debugger for State Machine
	Language-Specific Extensions to Functional Debugging/Tracing and Testing		
Structure	Basic Data Result Data.		Time Series Components Test Scenarios
Behaviors	Calculation Rules & Dependencies Variants Validity	Interactors & Processes State Machines & Interceptors Context Arguments Live Values	(Special) State Machines Test Generation & Mutation Coverage Measurement Utils
Functional & Types	Date Types Currency Types Temporal Types		Time & Duration Types Decision Trees and Tables Sentence-like Function Calls
KernelF			

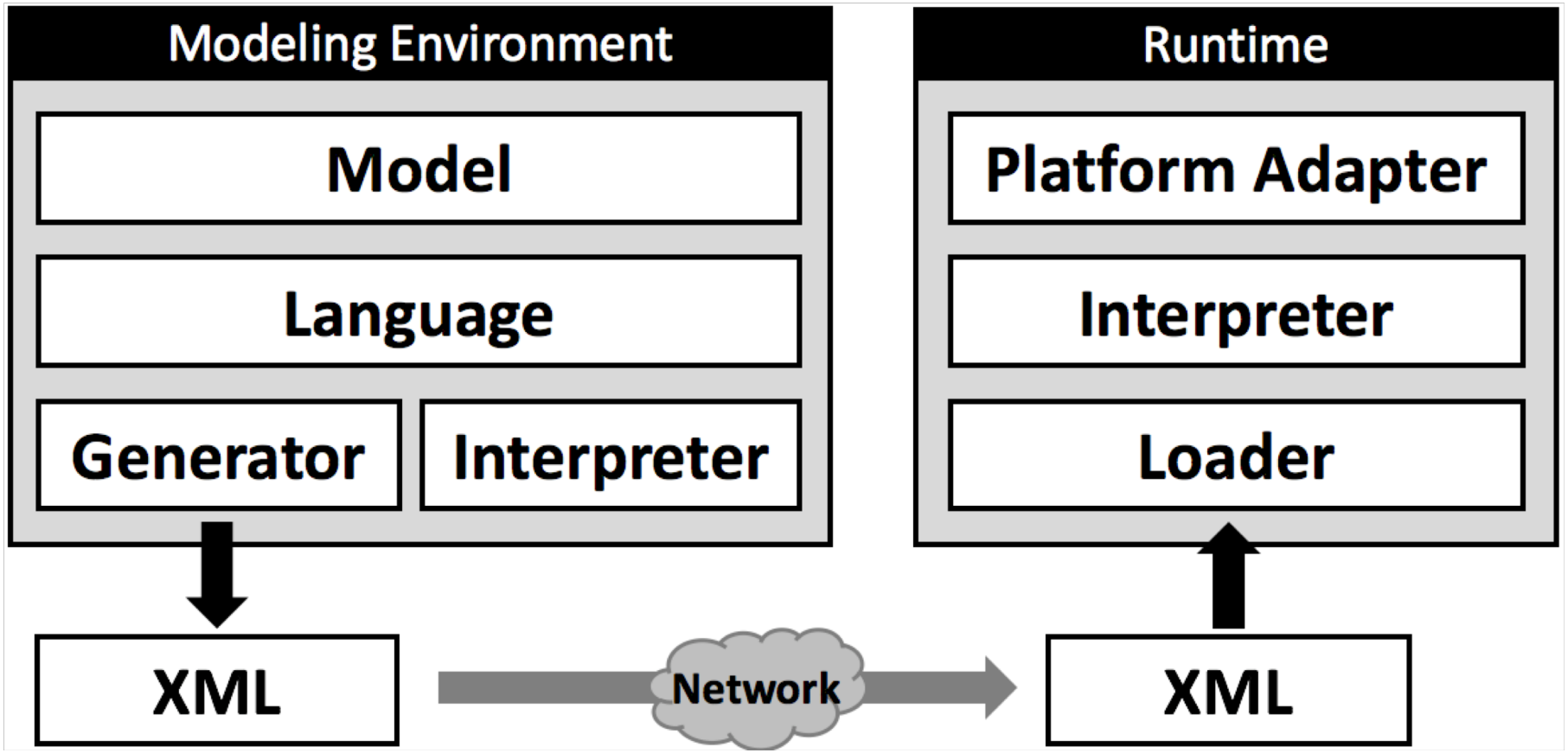
2

MPS & Safety

Languages Used

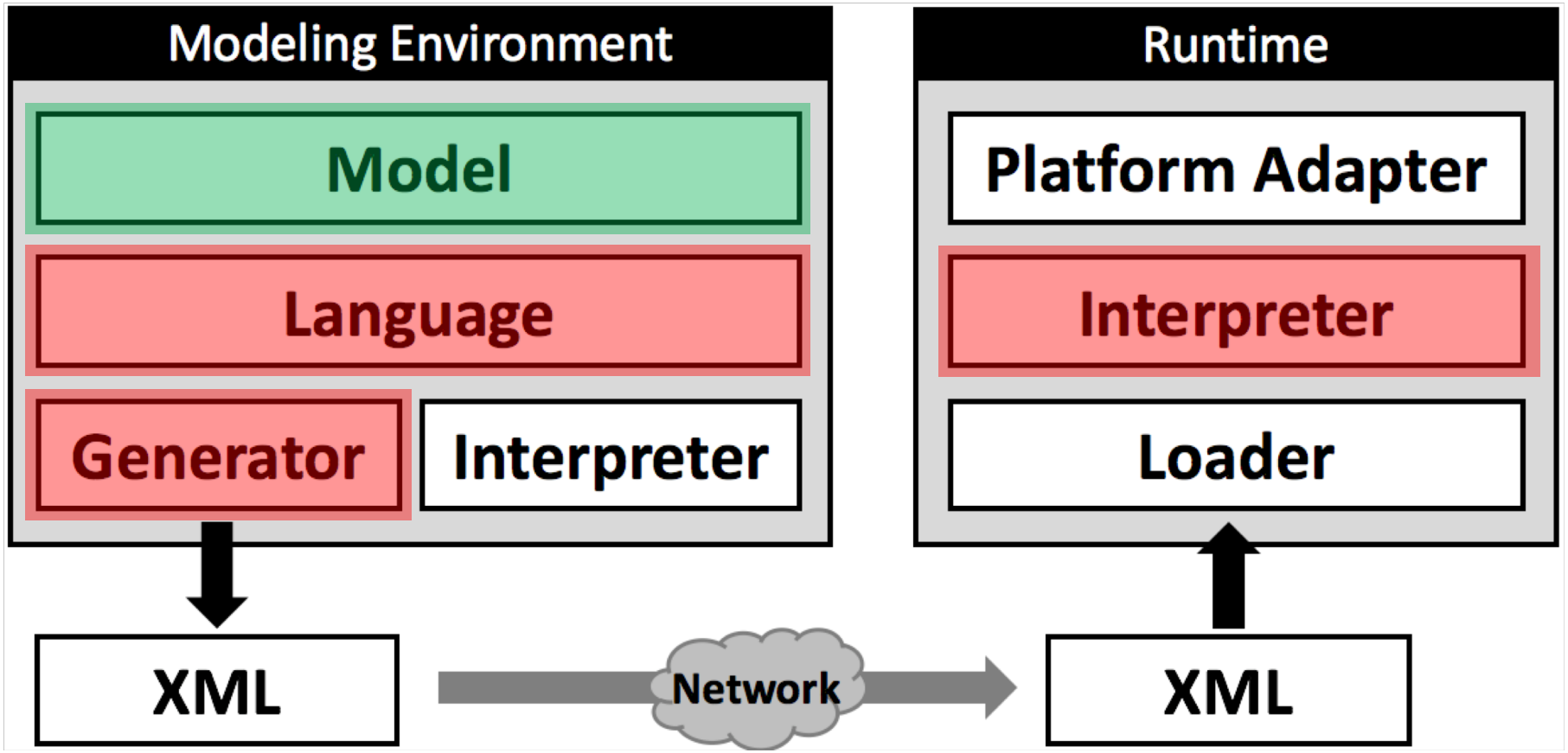


System Architecture



What good is all the abstraction if we cannot trust the translation to the implementation?

System Architecture & Safety Standards

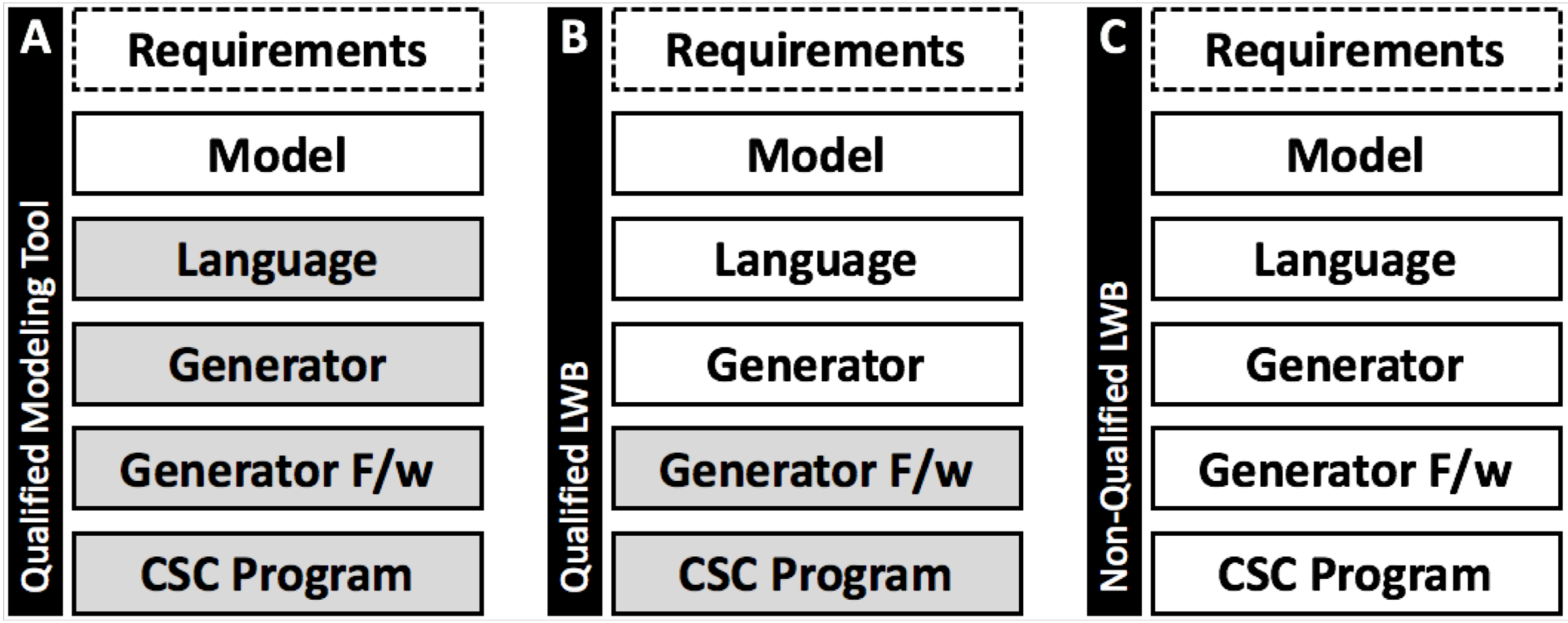


Tools may introduce *additional systematic errors* if faulty.
Safety **standards** require reliable mitigation of such errors.

DO-178C EN50129 IEC62304 ISO26262

Unqualified Tools!

What good is all the abstraction if we cannot trust the translation to the implementation?



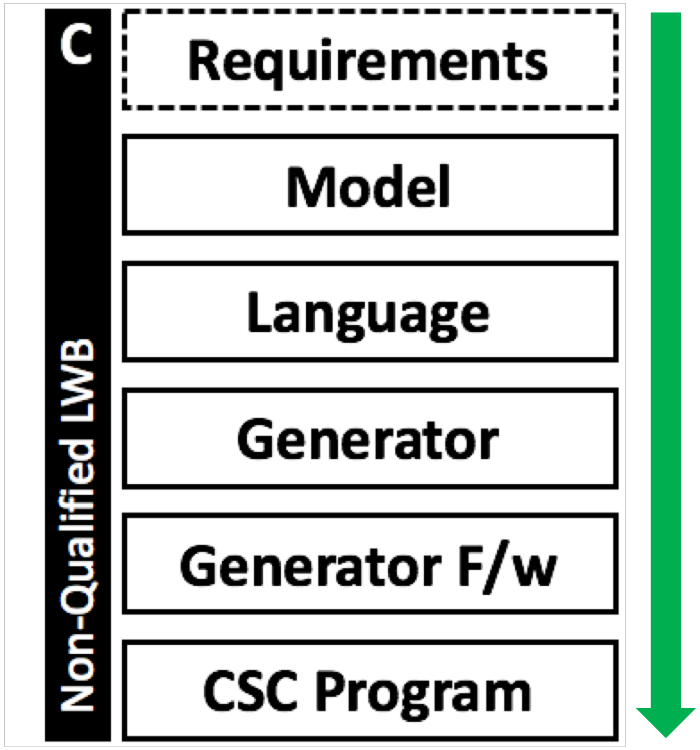
Unqualified Tools!

End-to-end testing required.
How to do this without exploding effort?

Automated Redundancy

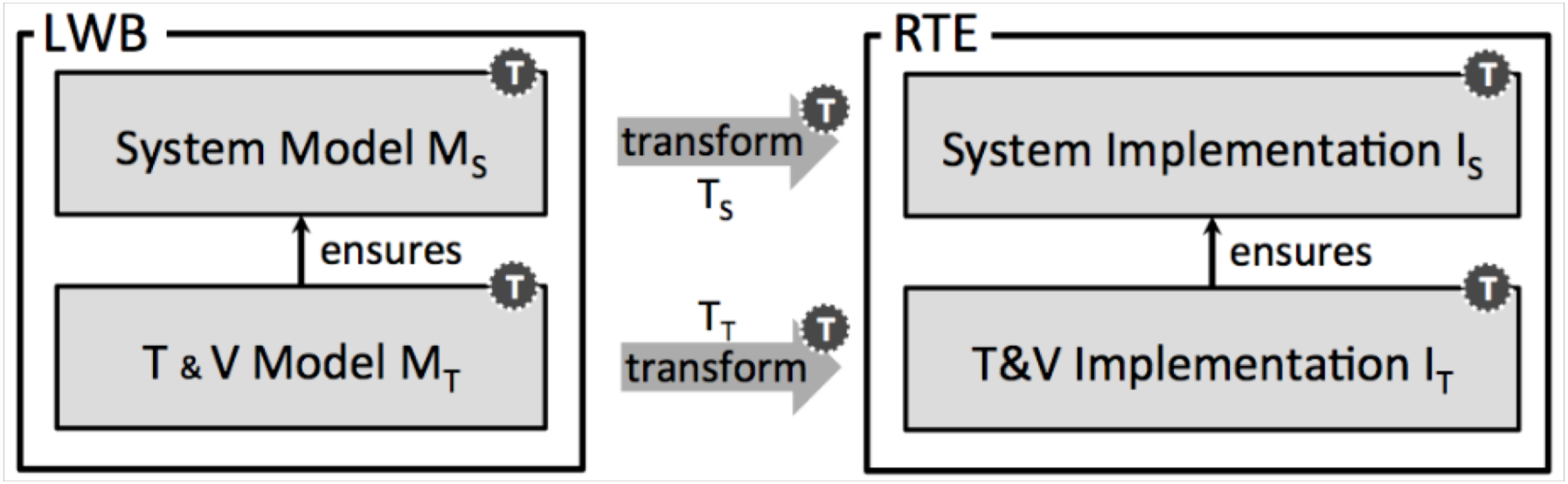
catch errors in redundant path while reducing manual effort.

+ specific risk mitigations



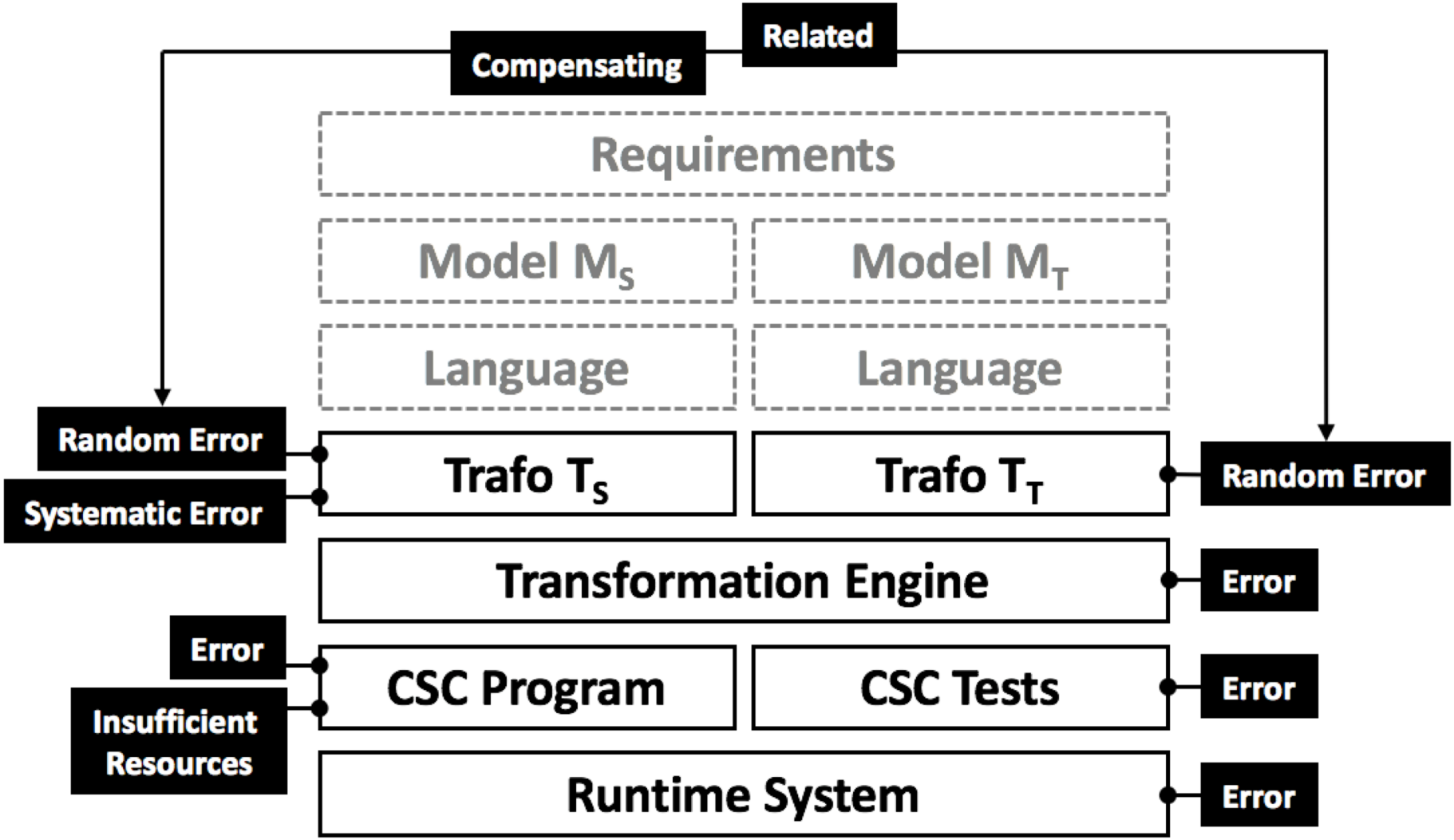
Modeling Architecture

Model the Algo/System with the DSL and also model the tests/verification. Then translate both and execute on the level of the implementation.

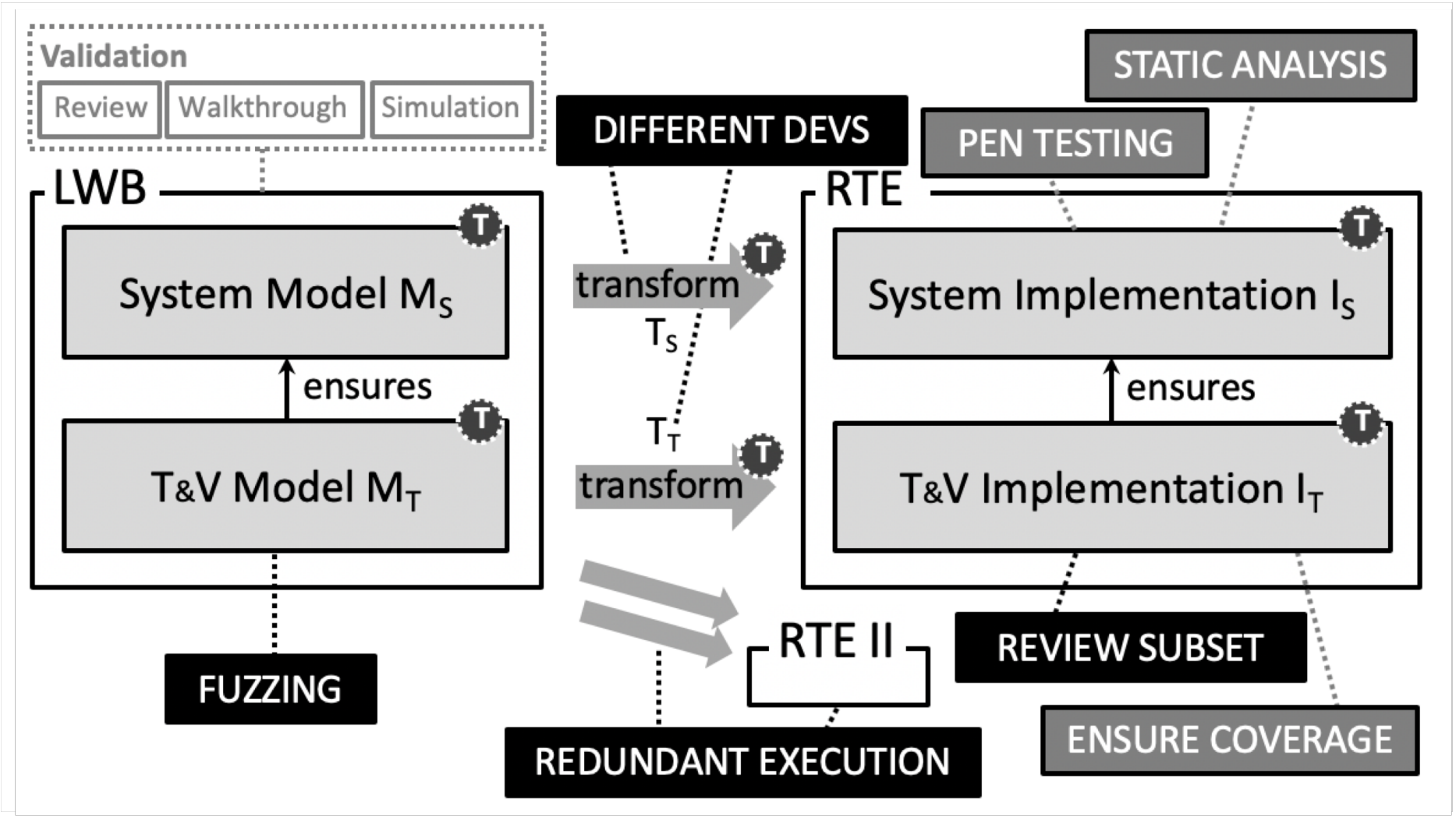


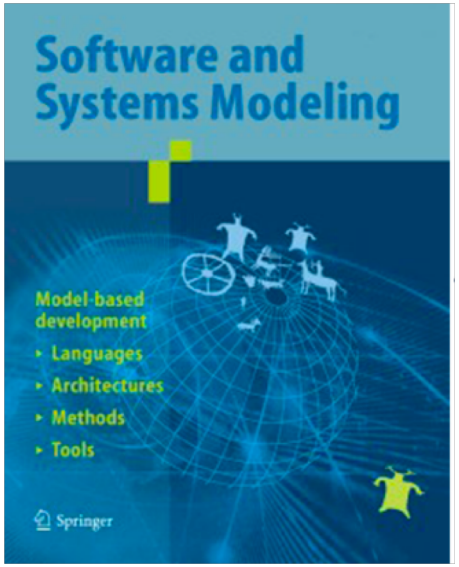
+ Risk Analysis + Mitigations

Risk Analysis



Mitigations – Safe Modeling Architecture





Software & Systems Modeling
pp 1–24 | Cite as

Using language workbenches and domain-specific languages for safety-critical software development

Authors | Authors and affiliations

Markus Voelter, Bernd Kolb, Klaus Birken, Federico Tomassetti, Patrick Alff, Laurent Wiart, Andreas Wortmann, Arne Nordmann

Regular Paper
First Online: 17 May 2018

Abstract

Language workbenches support the efficient creation, integration, and use of domain-specific languages. Typically, they execute models by code generation to programming language code. This can lead to increased productivity and higher quality. However, in safety-/mission-critical environments, generated code may not be considered trustworthy, because of the lack of trust in the generation mechanisms. This makes it harder to justify the use of language workbenches in such an environment. In this paper, we demonstrate an approach to use such tools in critical environments. We argue that models created with domain-specific languages are easier to validate and that the additional risk resulting from the transformation to code can be mitigated by a suitably designed transformation and verification architecture. We validate the approach with an industrial case study from the healthcare domain. We also discuss the degree to which the approach is appropriate for critical software in space, automotive, and robotics systems.

http://voelter.de/data/pub/MPS-in-Safety-1.0.pdf

3

Realtime Incremental Transformations

What is Shadow Models?

Functional **transformation language** with support for fixpoints

Incremental execution upon change of input model

Unidirectional, but with first-class support for **lifting** results

Fully integrated into **MPS** IDE

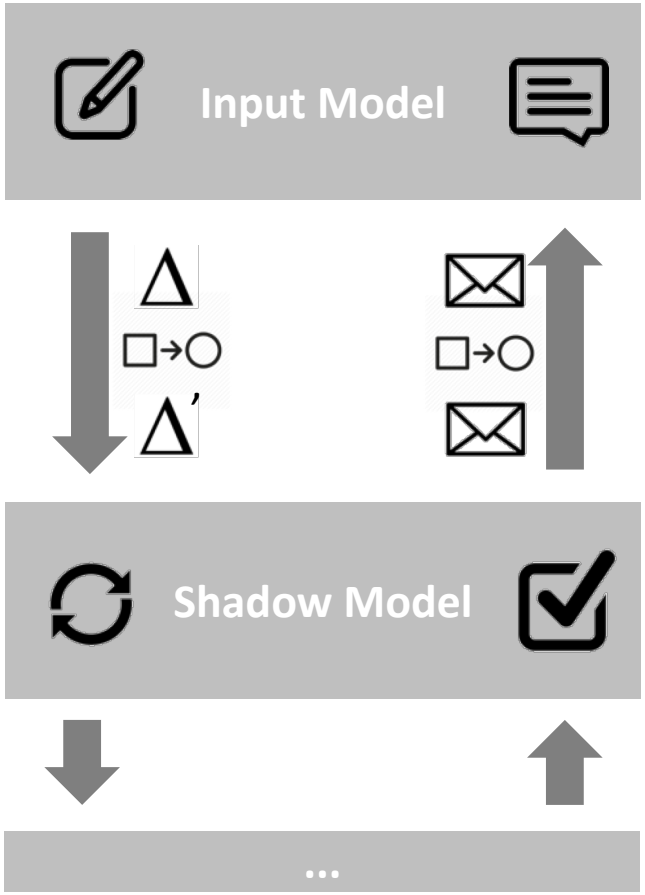
Code: https://github.com/JetBrains/MPS-extensions/tree/master/code/shadowmodels

Docs: https://jetbrains.github.io/MPS-extensions/extensions/shadowmodels/

Mechanics

Might be stacked

- User **edits** the input model
- A **delta** is propagated into the transformation engine
- A **change** on the shadow model is produced
- This **triggers** analysis or update of results in Shadow Model
- Results/messages are **lifted** back to input level
- Results are **annotated** to the input model



Example Input

```
program Example2 {

  [LIFTED] duplicate literal names                [LIFTED] this is the duplicate
  enum color { red, green, yellow, yellow }

  fun decide(a: int, b: int) = alt [ a > b      => 1
                                     a == b     => 2
                                     otherwise  => 3 ]

  val res =                ?maybe?  ?maybe?  ?maybe?
    [LIFTED] This table is not guaranteed to succeed; unhandled combinations possible.
            ?maybe? | 2          4          6
}
```

Example Shadow

```
program Example2 {

  val Color_red = -309736043
  val Color_green = -1313361145
  val Color_yellow = -1555955472
  val Color_yellow = -1555955472

  fun decide(a: int, b: int) = if a > b
                                then 1
                                else if a == b then 2 else 3

  val res = if ?maybe?
            then if ?maybe?
                 then 1
                 else if ?maybe? then 4 else <!>
            else if ?maybe?
                 then if ?maybe?
                      then 3
                      else if ?maybe? then 4 else <!>
                 else if ?maybe?
                      then if ?maybe?
                           then 5
                           else if ?maybe? then 6 else <!>
                      else <!>
}
```

Error: duplicate name Color_yellow

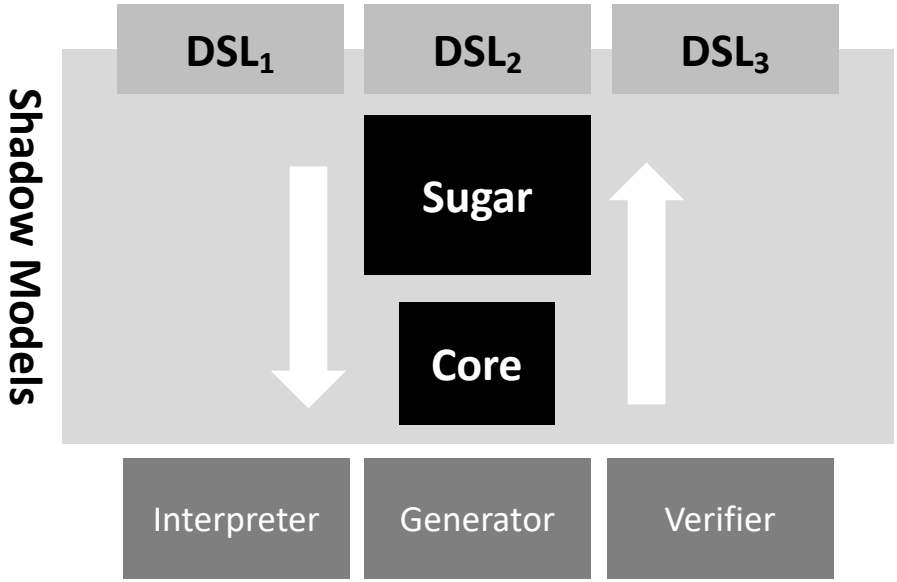
Error: this should never occur in a program

Example Use Cases

- Mapping of system models to **model checkers** for interactive verification of temporal properties
- **Flattening** of component hierarchies for type checking
- Flattening of hierarchical **feature models** for path expression evaluation
- Weaving in of **safety concerns** into C code.
- **Desugaring** of business DSLs to a core functional language; that core language would then feature an interpreter, a compiler and an integration with an SMT solver

Use Case: Kf2

- Minimal, Expressive Core
  - Functional
  - Reactive
- Sugar on top
- Interpter, Generator and Verifier below
- Realtime-Trafo with Shadow Models



Use Case: Safety Patterns

```
void PIControlTask( ) {
  // sample current spacings between the cars
  float volatile curr_d = readDistanceSensor( );
  // sample current speed of the car
  float volatile curr_v = readSpeedSensor( );
  // error integration using Newton method
  static float error_integral = 0.0f;
  error_integral += (curr_d - DESIRED_DISTANCE);
  // applying control law
  float throttle_u = PI(curr_d, curr_v, error_integral);
  // writing to the actuator
  writeActuator(throttle_u);
} PIControlTask (function)
```

Diese Arbeit ist im Projekt SAFE4I entstanden. SAFE4I wird unter dem Förderkennzeichen 01IS17032 durch das BMBF gefördert. Für den Inhalt dieser Publikation sind die Autoren verantwortlich.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Use Case: Safety Patterns

```
void PIControlTask()
  [Safe Function:StatementDuplication]
  {
    // sample current spacings between the cars
    float volatile curr_d = readDistanceSensor();
    // sample current speed of the car
    float volatile curr_v = readSpeedSensor();
    // error integration using Newton method
    static float error_integral = 0.0f;
    error_integral += (curr_d - DESIRED_DISTANCE);
    // applying control law
    float throttle_u = PI(curr_d, curr_v, error_integral);
    // writing to the actuator
    writeActuator(throttle_u);
  } PIControlTask (function)
```

Diese Arbeit ist im Projekt SAFE4I entstanden. SAFE4I wird unter dem Förderkennzeichen 01IS17032 durch das BMBF gefördert. Für den Inhalt dieser Publikation sind die Autoren verantwortlich.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Use Case: Safety Patterns

```
void PIControlTask()
  [Safe Function:StatementDuplication]
  {
    // sample current spacings between the cars
    float volatile curr_d = readDistanceSensor();
    // sample current speed of the car
    float volatile curr_v = readSpeedSensor();
    // error integration using Newton method
    static float error_integral = 0.0f;
    error_integral += (curr_d - DESIRED_DISTANCE);
    // applying control law
    float throttle_u = PI(curr_d, curr_v, error_integral);
    // writing to the actuator
    writeActuator(throttle_u);
  } PIControlTask (function)
```

+

SAFETY-PATTERN:Safe Function – *StatementDuplication*

Configuration

Delta constraints

```
constants {
  // mmap of custom VP
  #alias NOTIFY_DC_DETECTION = hex«ffffff06u»;
}
delta function {
  static int8 cnt = 0;
  if (cnt == 10) {
    if ((DELTA > 8) || (DELTA < -8)) {
      *((int8 volatile*) (NOTIFY_DC_DETECTION)) = 1;
    } if
  } else {
    cnt++;
  } if
}
```

Required capabilities

Avoid pass-by-reference

Diese Arbeit ist im Projekt SAFE4I entstanden. SAFE4I wird unter dem Förderkennzeichen 01IS17032 durch das BMBF gefördert. Für den Inhalt dieser Publikation sind die Autoren verantwortlich.

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

Use Case: Safety Patterns

```
void PIControlTask()
  [Safe Function:StatementDuplication]
  {
    // sample current spacings between the cars
    float volatile curr_d = readDistanceSensor();
    // sample current speed of the car
    float volatile curr_v = readSpeedSensor();
    // error integration using Newton method
    static float error_integral = 0.0f;
    error_integral += (curr_d - DESIRED_DISTANCE);
    // applying control law
    float throttle_u = PI(curr_d, curr_v, error_integral);
    // writing to the actuator
    writeActuator(throttle_u);
  } PIControlTask (function)
```

+

```
SAFETY-PATTERN:Safe Function - StatementDuplication
Configuration
Delta constraints
  constants {
    // mmap of custom VP
    #alias NOTIFY_DC_DETECTION = hex«ffffff06u»;
  }
  delta function {
    static int8 cnt = 0;
    if (cnt == 10) {
      if ((DELTA > 8) || (DELTA < -8)) {
        *((int8 volatile*) (NOTIFY_DC_DETECTION)) = 1;
      } if
    } else {
      cnt++;
    } if
  }
Required capabilities
Avoid pass-by-reference
```

=

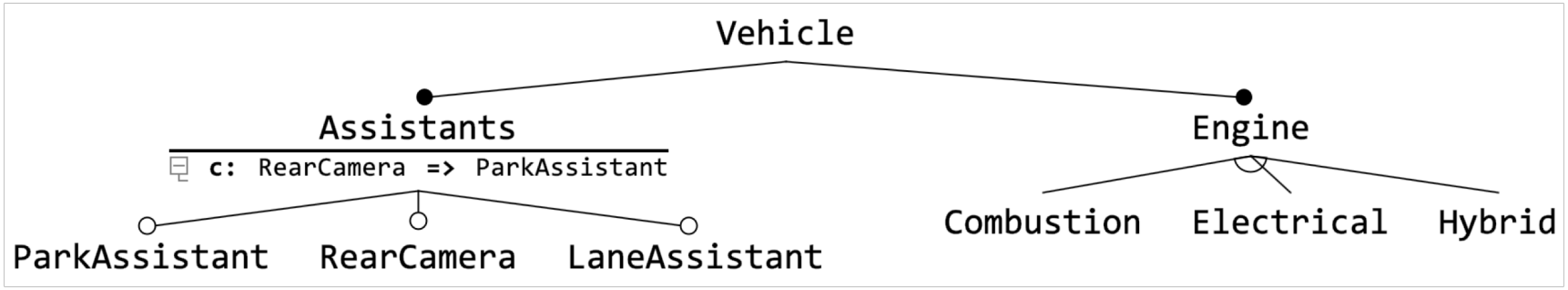
```
void PIControlTask() {
  // sample current spacings between the cars
  float volatile curr_d = readDistanceSensor();
  // sample current speed of the car
  float volatile curr_v = readSpeedSensor();
  // error integration using Newton method
  static float error_integral = 0.0f;
  error_integral += (curr_d - DESIRED_DISTANCE);
  // applying control law
  float throttle_u = PI(curr_d, curr_v, error_integral);
  // writing to the actuator
  // sample current spacings between the cars
  float volatile curr_d_dup = readDistanceSensor();
  // sample current speed of the car
  float volatile curr_v_dup = readSpeedSensor();
  // error integration using Newton method
  static float error_integral_dup = 0.0f;
  error_integral_dup += (curr_d_dup - DESIRED_DISTANCE);
  // applying control law
  float throttle_u_dup = PI(curr_d_dup, curr_v_dup, error_integral_dup);
  // writing to the actuator
  float delta = throttle_u_dup - throttle_u;
  static int8 cnt = 0;
  if (cnt == 10) {
    if ((delta > 8) || (delta < -8)) {
      *((int8 volatile*) (NOTIFY_DC_DETECTION)) = 1;
    } if
  } else {
    cnt++;
  } if
  writeActuator(throttle_u_dup);
```

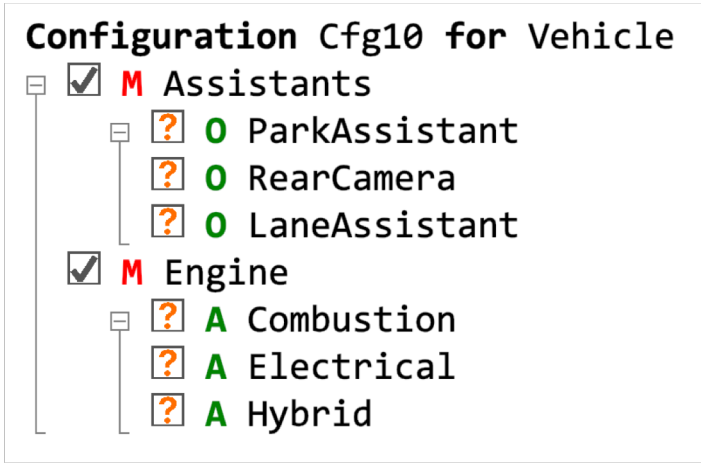
Diese Arbeit ist im Projekt SAFE4I entstanden. SAFE4I wird unter dem Förderkennzeichen 01IS17032 durch das BMBF gefördert. Für den Inhalt dieser Publikation sind die Autoren verantwortlich.

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

Use Case: Interactive FM Specialization

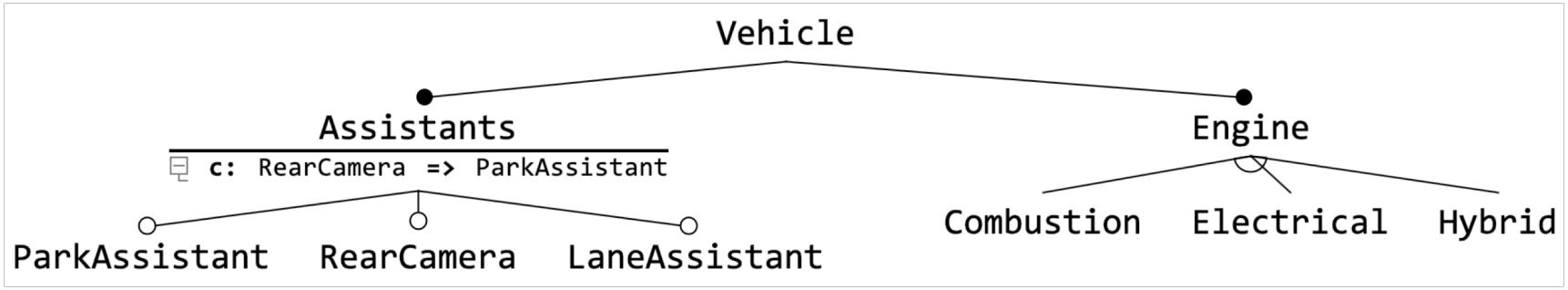
Klaus Birken

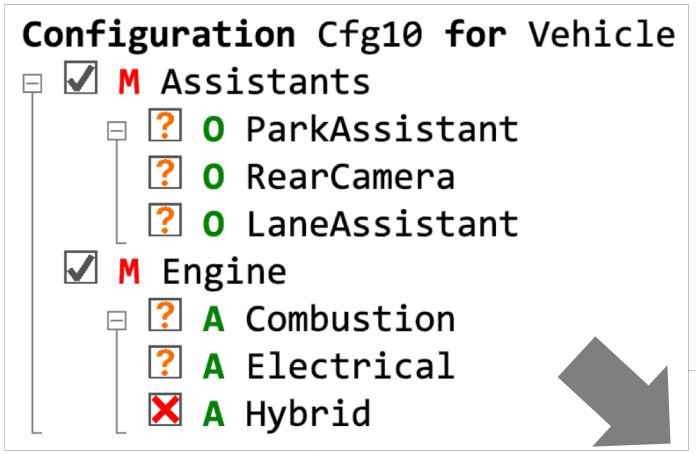




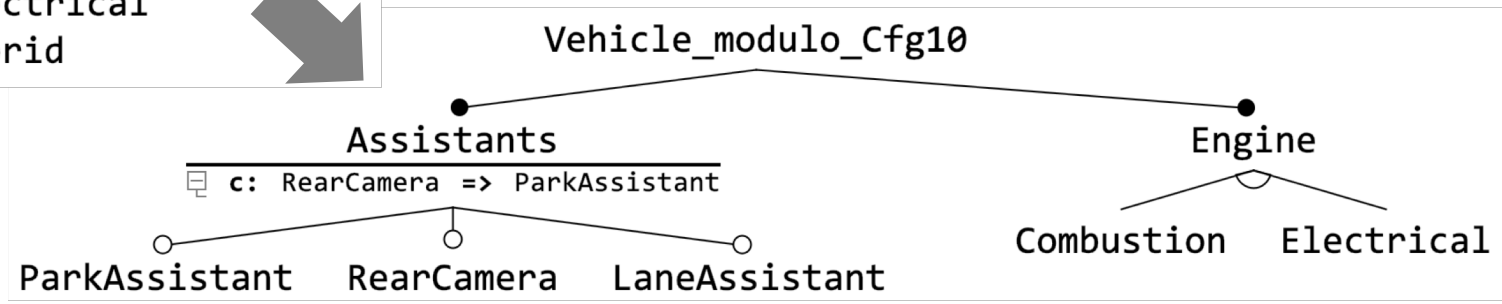
Use Case: Interactive FM Specialization

Klaus Birken



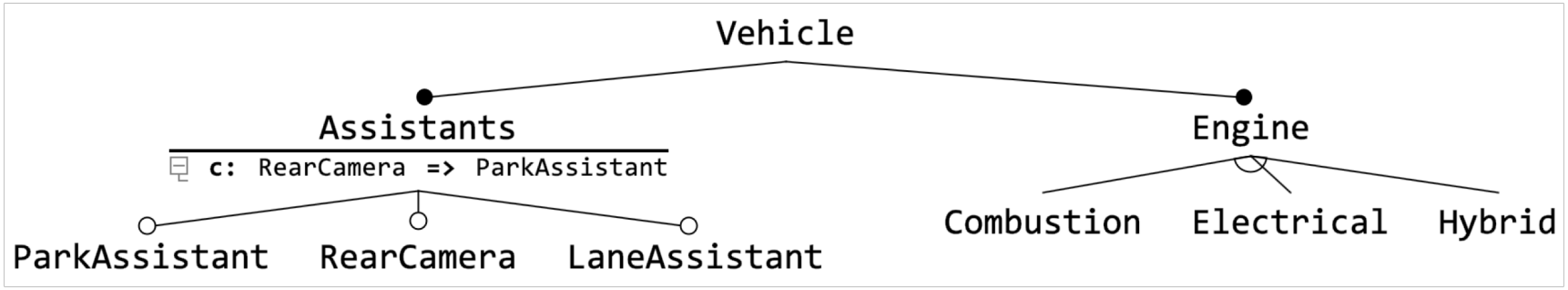


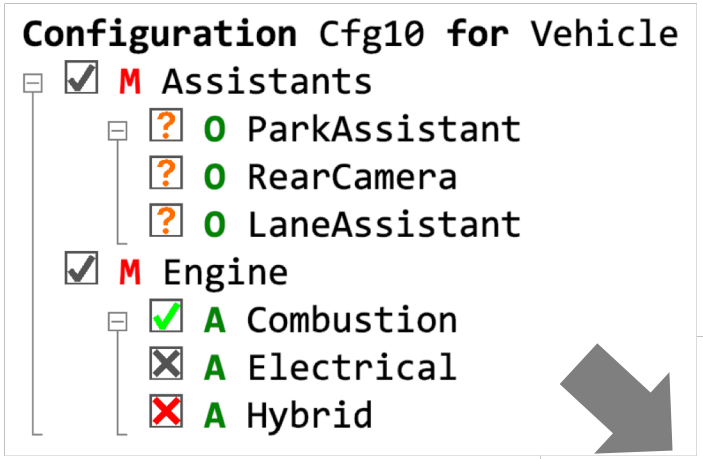
Uncheck
Hybrid



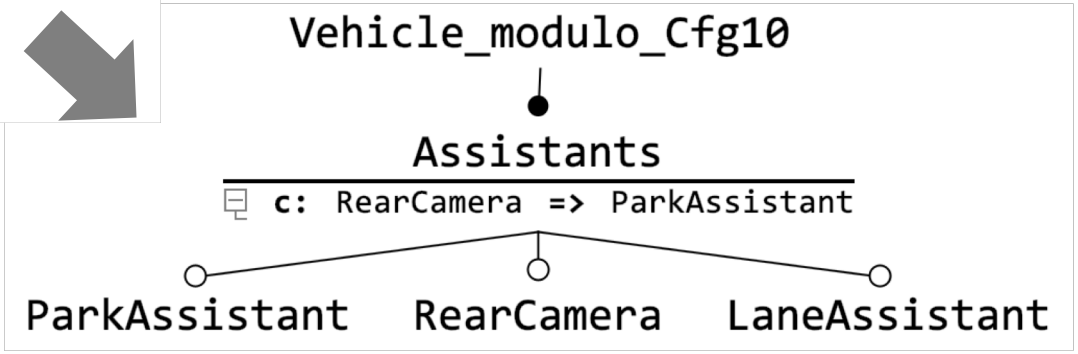
Use Case: Interactive FM Specialization

Klaus Birken



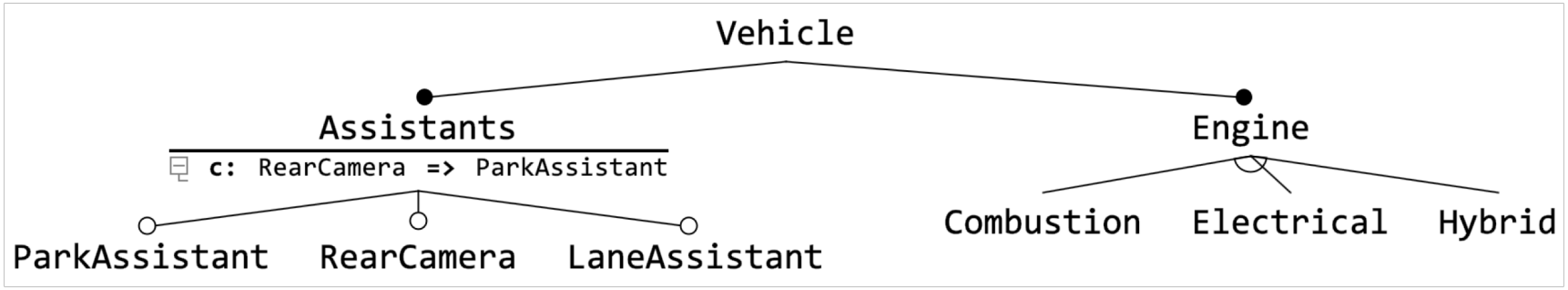


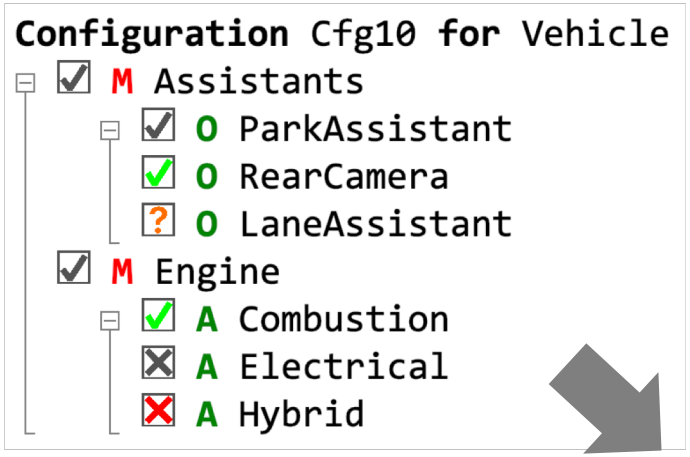
Check
Combustion



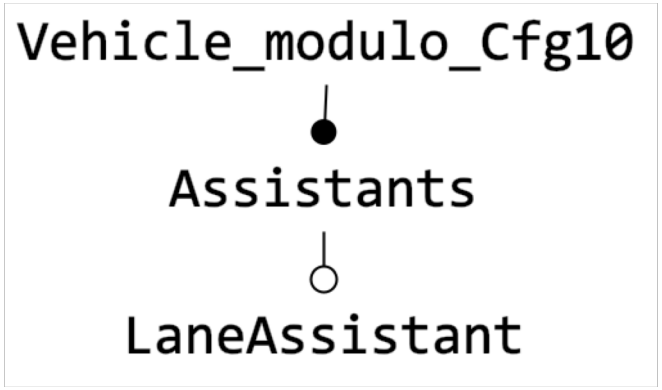
Use Case: Interactive FM Specialization

Klaus Birken





Check
RearCamera



Current State (March 2019)

Still under active development.

Primarily by Sascha Lisson.

Used for initial use cases at itemis Stuttgart.

4

Dclare for MPS

Declarative Rules for MPS

(Modeling Value Group)

Module_RuleSet

```
// attributes

model targetModel := this.findOrAddModel("target", true);

nlist<Entity> entities := this.models.roots(Entity);

// rules

targetModel.roots(Class) := entities.target;
```

Entity_RuleSet

```
node ruleset Entity {

  // attributes

  node<Class> target := <Class(
                          name: this.name,
                          attributes: # this.properties.target.concat(this.relations.target))>;
```

Entity | Entity_Editor | Entity_RuleSet

Property_RuleSet

```
node ruleset Property {

  // attributes

  node<Attribute> target := this.numeric ? <Attribute(
                                              name: this.name,
                                              type: Numeric)> : <Attribute(
                                                                  name: this.name,
                                                                  type: String)>;
```

Property | Property_Editor | Property_RuleSet

Relation_RuleSet

```
node ruleset Relation {

  // attributes

  node<Attribute> target := <Attribute(
                              name: this.name,
                              type: # this.entity.target,
                              opposite: # this.opposite.target)>;
```

Relation | Relation_Editor | Relation_RuleSet

Customer

```
entity Customer {
  properties:
    name    numeric=false
    age     numeric=true

  relations :
    order : Order <-> customer

}
```

Customer

```
class Customer {
  attributes:
    attribute name : String <-> <no opposite>;
    attribute age : Numeric <-> <no opposite>;
    attribute order : Order <-> customer;

}
```

Order

```
entity Order {
  properties:
    number    numeric=true

  relations :
    customer : Customer <-> order

}
```

Order

```
class Order {
  attributes:
    attribute number : Numeric <-> <no opposite>;
    attribute customer : Customer <-> order;

}
```

Declarative Rules for MPS

(Modeling Value Group)

Unidirectional.

Bidirectional requires complementary rules

Granularity is per **node/property** combination

Dynamic read-tracking

No scalability numbers yet.

Open Source.

https://github.com/ModelingValueGroup/DclareForMPS

5

Plain Text Gen

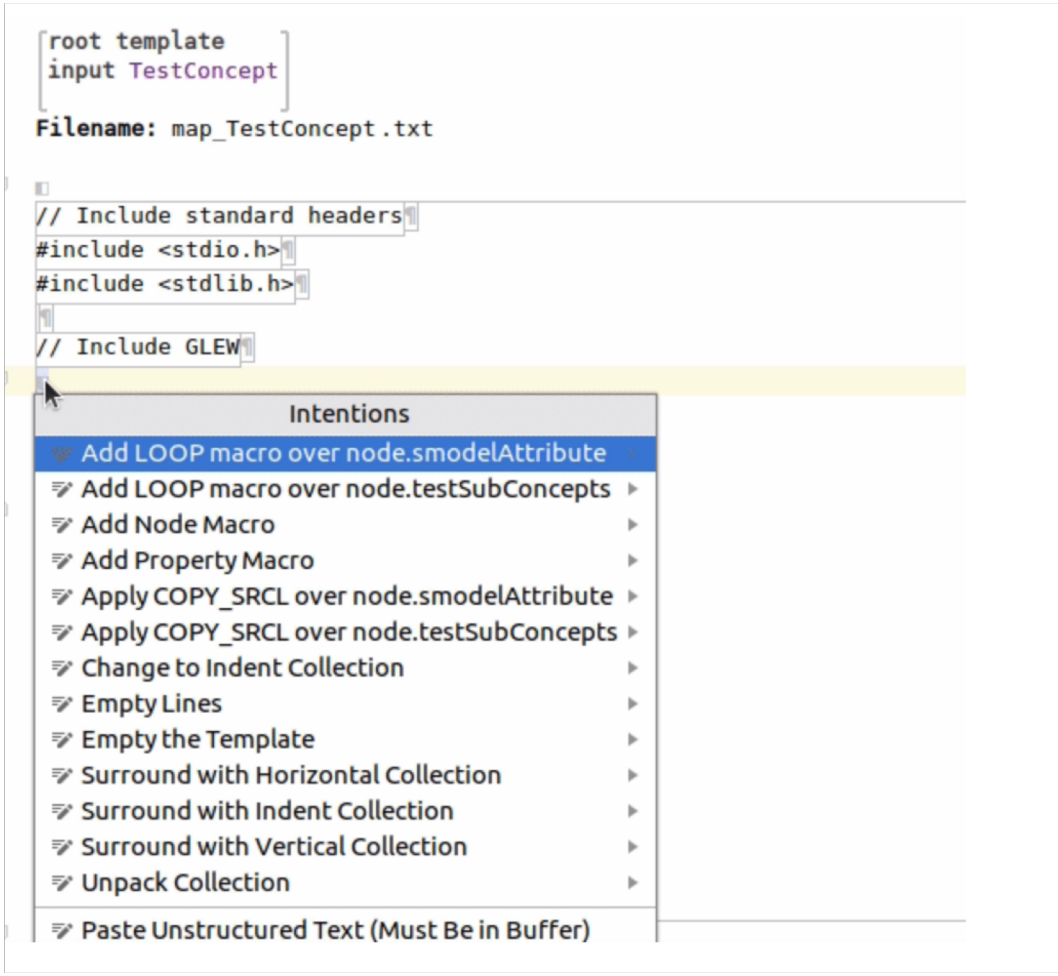
Plain Text Gen

(Schindler Universe)

„Template-engine-style“
text generator
on top of MPS textgen

Abstractions
for words, lines

Paste in literal text
then selectively replace
with string-typed
expressions



6

MPS Community

MPS Community

Meetup 2019 tentatively scheduled:

Oct 10/11 in Amsterdam

Community Guidelines online:

https://confluence.jetbrains.com/display/MPS/Community+Guidelines+of+MPS

Support Contract available for serious users

https://www.jetbrains.com/mps/support/partnership.html

Continued evolution of MPS:

- Development has obviously slowed down.
- Many of the team members work on apps on top of MPS (or support paying customers).
- Team is planning to hire two additional devs.
- Future evolution increasingly aligned with the needs of (support-paying) users.
- A process for coordinating them is being designed.

TALK TO JETBRAINS!

voelter@acm.org
www.voelter.de
@markusvoelter
Markus Völter
itemis voelter { ingenieurbüro für softwaretechnologie //