JetBrains MPS

Create a programming language that the whole company can understand

Artem Tikhomirov

Software Engineering Conference Russia
October 2017, St. Petersburg

Domain Specific Languages

- A DSL is a **focused**, **processable language** for describing a specific **concern** when building a system in a specific **domain**

- The **abstractions** and **notations** used are natural/suitable for the **stakeholders** who specify that particular concern.

Markus Voelter

Math

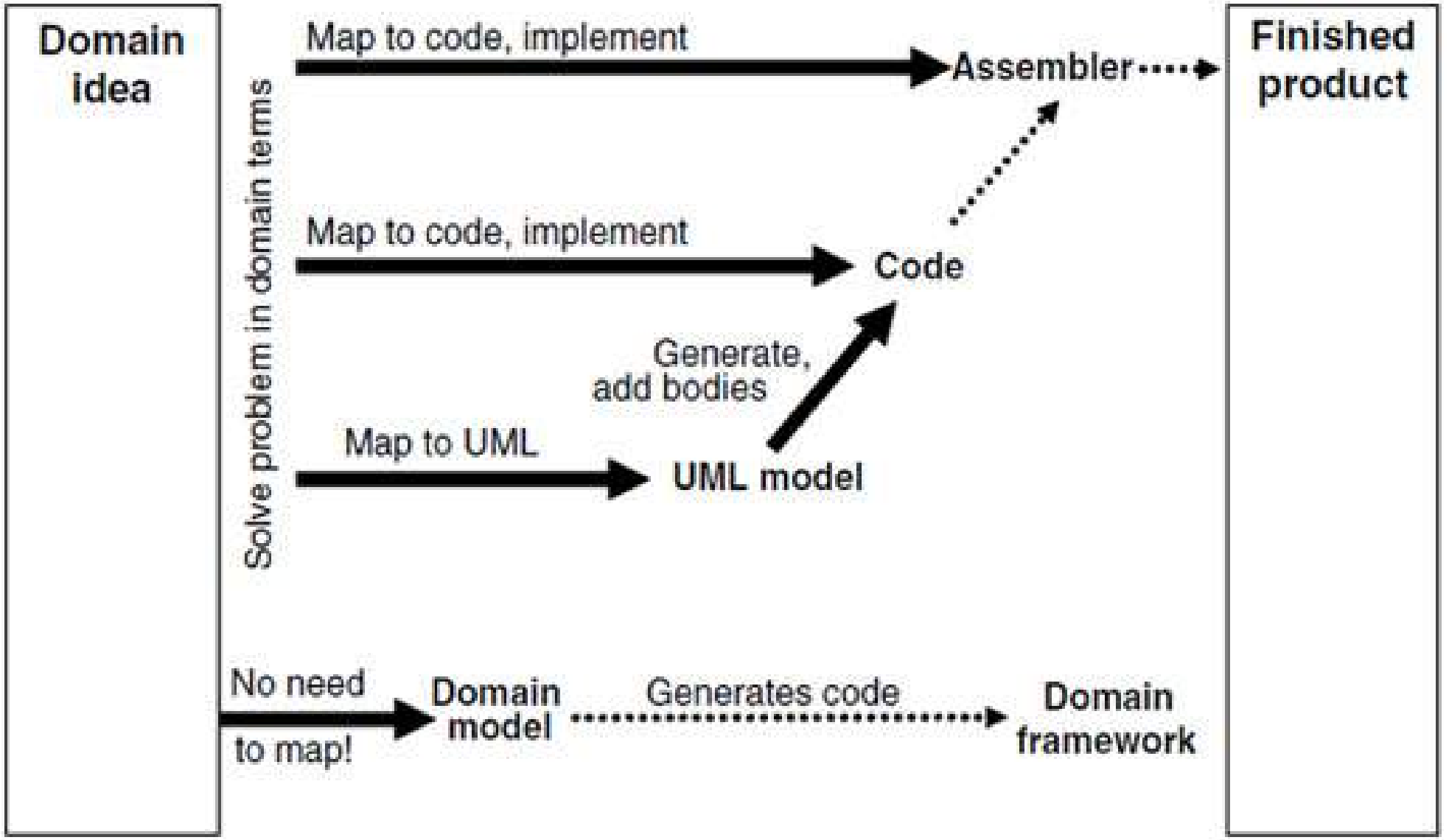
$$2\cos z = \left(1 + \frac{\sqrt{-1}z}{\infty}\right)^{\infty} + \left(1 - \frac{\sqrt{-1}z}{\infty}\right)^{\infty} = e^{\sqrt{-1}z} + e^{-\sqrt{-1}z}$$

$$\Psi(x) = \sum_{k=0}^{\infty} \frac{\cos\left(3^k x\right)}{k!}$$

$$\int_\gamma f(z)\,dz = \int_a^b f(z(t))z'(t)\,dt = \int_\gamma (u\,dx - v\,dy) + i\int_\gamma (v\,dx + u\,dy).$$

$$\sum_{1 \leqslant k \leqslant n} f(z(t_k))(z(t_k) - z(t_{k-1})).$$

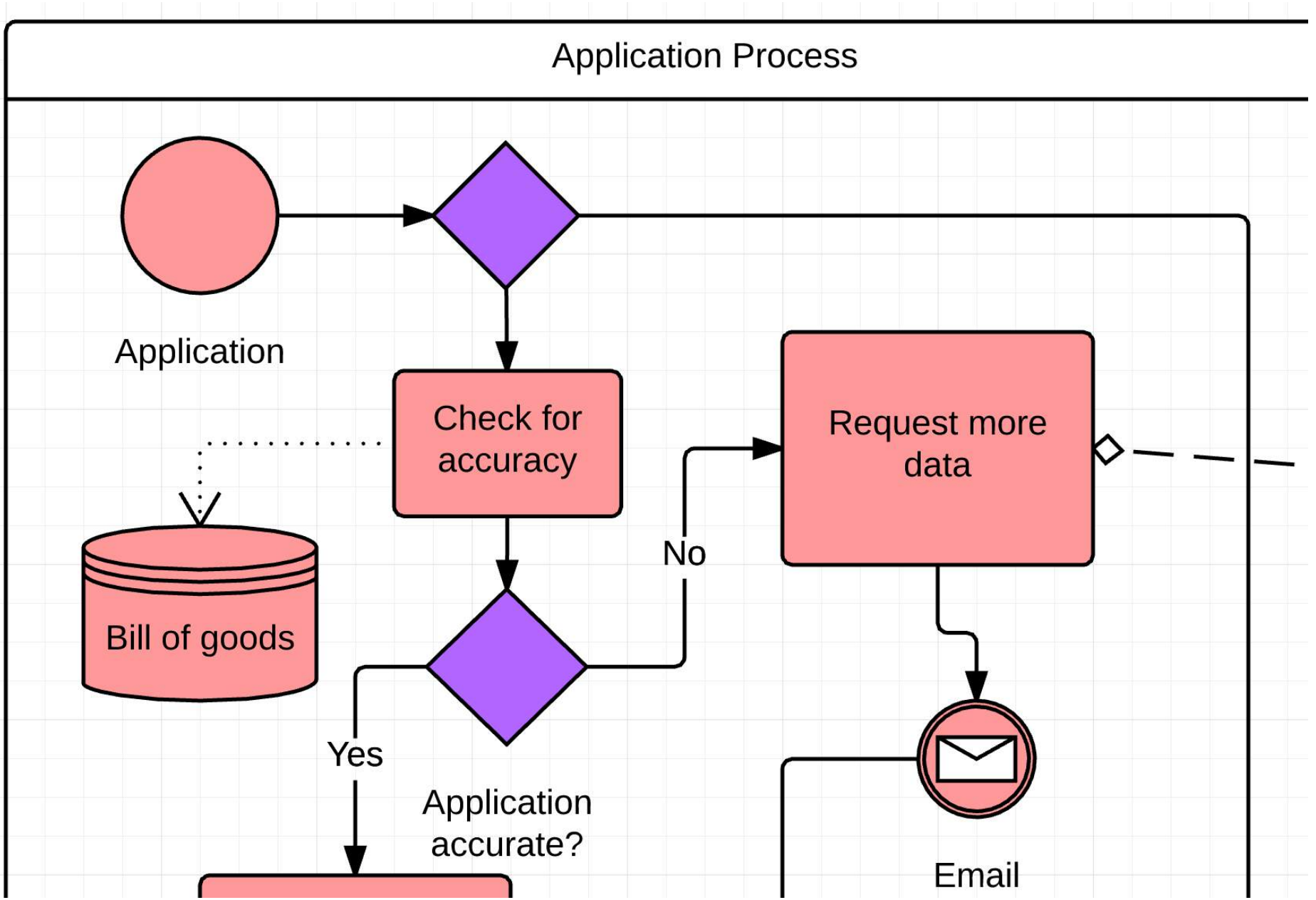
How come?


Domain idea
Solve problem in domain terms
Map to code, implement
Assembler
Finished product
Map to code, implement
Code
Generate, add bodies
Map to UML
UML model
No need to map!
Domain model
Generates code
Domain framework

DSLs? Who cares?

You! DSLs are ubiquitous!

Buisness Processes



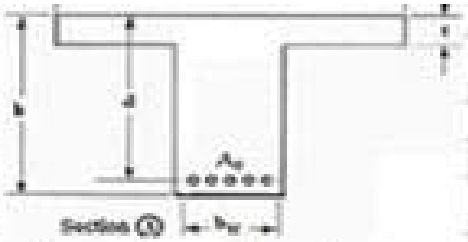
ма-ма | ра-ма | Шу-ра
ра-ма | ра-мы | мы-ла

Ма-ма мы-ла ра-му.

- Health and medicine
  - Stakeholder integration, Scalable Business, Document Generation + Certification
- Finance
  - Precise Specification and Implementation of Insurance Products („Rules“)
- Government
  - Changing Regulations, Fast Implementation, End User Empowerment
- Automotive
  - Code Complexity, Frameworks (Autosar), Product Lines
- Aerospace
  - Reduction of Accidental Complexity in Code, Process Conformance (Docs)
- Robotics
  - A powerful language and IDE for existing frameworks (Industry Robots, ROS)
- Embedded software
  - Multi-Paradigm Programming, not just Simulink and C
- Science
  - Consistent Derived Documents

Most widely adopted DSM tool

Select Strength:	fy = 60000 psi; f'c = 4000 psi			Select design ρ:	ρmax	Select Load Case:	1.2D + 1.6L
f_y =	60000	f'_c =	4000	Design ρ =	0.0069	Est w_u [k/ft] =	1.78
ρ_{min} =	0.0033	ρ_{max} =	0.0181	Design R_u =	388.79	M_u [ft-k] =	227.8
R_u at ρ_{min} [psi] =	192.2	R_u at ρ_{max} [psi] =	912.0	bd^2 =	7813.549	M_n [ft-k] =	253.2
				d =	22.10	Density [pcf]	150



Enter Givens & Estimates:					
w_D [k/ft]	0.55	b_w [in]	16	b_{eff} [in]	80
w_L [k/ft]	0.4	h [in]	27	t [in]	3
Length [ft]	32	d [in]	24	a [in]	0.49
Clear Cover [in]	3	A_s [in^2]	2.63	β_1	0.85
Est Self Wt [plf]	400	$A_{s\,min}$ [in^2]	1.214	c [in]	0.57
Actual Self Wt [plf]	450.00	$A_{s\,min}$ [in^2]	1.280	ε_t	0.1227
		ϕM_n [ft-k]	253.36		

Neutral Axis Calculations:			N.A. is in the flange
T [kips]	142.2	z [in] =	21.60
A_s [in^2]	41.8	z [in] =	22.50
$A_{s\,req'd}$ [in^2]	240	Governing z:	22.50

Select A_s in Tension:		
BAR	Qty	As [in^2]
#8	3	2.37

If a < ht	
A_s [in^2]	2.6496

If a > t	
A_{sf} [in^2]	10.88
M_{nw} [k-ft]	1224.00
M_{uf} [k-ft]	996.16
R_{nw}	1441.20
ρ_w =	0.03
a [in]	14.64
c [in]	17.22
ε_t	0.00
$A_{s\,tot}$ [in^2]	13.27
A_s [in^2]	24.15

Bar Selection:

BAR	Number of Bars in a Single Layer of Reinforcing													
	2		3		4		5		6		7		8	
	Area	Width	Area	Width	Area	Width	Area	Width	Area	Width	Area	Width	Area	Width
#4	0.39	6.80	0.58	8.30	0.78	9.80	0.98	11.30	1.18	12.80	1.37	14.30	1.57	15.80
#5	0.61	6.90	0.91	8.50	1.23	10.20	1.53	11.80	1.84	13.4	2.15	15.00	2.45	16.70
#6	0.88	7.00	1.32	8.80	1.77	10.50	2.21	12.30	2.65	14.00	3.09	15.80	3.53	17.50
#7	1.20	7.20	1.80	9.00	2.41	10.90	3.01	12.80	3.61	14.70	4.21	16.50	4.81	18.40
#8	1.57	7.50	2.35	9.50	3.14	11.30	3.93	13.30	4.71	15.30	5.50	17.30	6.28	19.30
#9	2.00	7.60	3.00	9.80	4.00	12.10	5.00	14.30	6.00	16.60	7.00	18.80	8.00	21.10
#10	2.55	7.80	3.79	10.40	5.06	12.90	6.33	15.50	7.59	18.00	8.86	20.50	10.12	23.10
#11	3.12	8.10	4.68	10.90	6.25	13.80	7.81	16.60	9.37	19.40	10.94	22.20	12.50	25.00
#14	4.50	8.90	6.75	12.30	9.00	15.70	11.25	19.00	13.50	22.40	15.75	25.80	18.00	29.20
#18	8.00	10.00	12.00	15.10	16.00	19.60	20.00	24.10	24.00	28.60	28.00	33.10	32.00	37.70

Too narrow view of programming?

Demo

- Language to configure voice menu for an answering maching
- Jetbrains MPS as a Rich Client Platform
- Business value is negligible
  - Market share of your office assistant is small

- Health and medicine
  - Stakeholder integration, Scalable Business, Document Generation + Certification
- Finance
  - Precise Specification and Implementation of Insurance Products („Rules“)
- Government
  - Changing Regulations, Fast Implementation, End User Empowerment
- Automotive
  - Code Complexity, Frameworks (Autosar), Product Lines
- Aerospace
  - Reduction of Accidental Complexity in Code, Process Conformance (Docs)
- Robotics
  - A powerful language and IDE for existing frameworks (Industry Robots, ROS)
- Embedded software
  - Multi-Paradigm Programming, not just Simulink and C
- Science
  - Consistent Derived Documents

Healthcare

Software Medical Devices Accessible to Doctors

Robustness and Correctness Required

To be FDA-certified

Needs to run on multiple target platforms

- IOS
- Android
- JavaScript

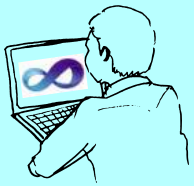
Stakeholders driving the development

Product Management
Customer Service

Physicists

System Engineering

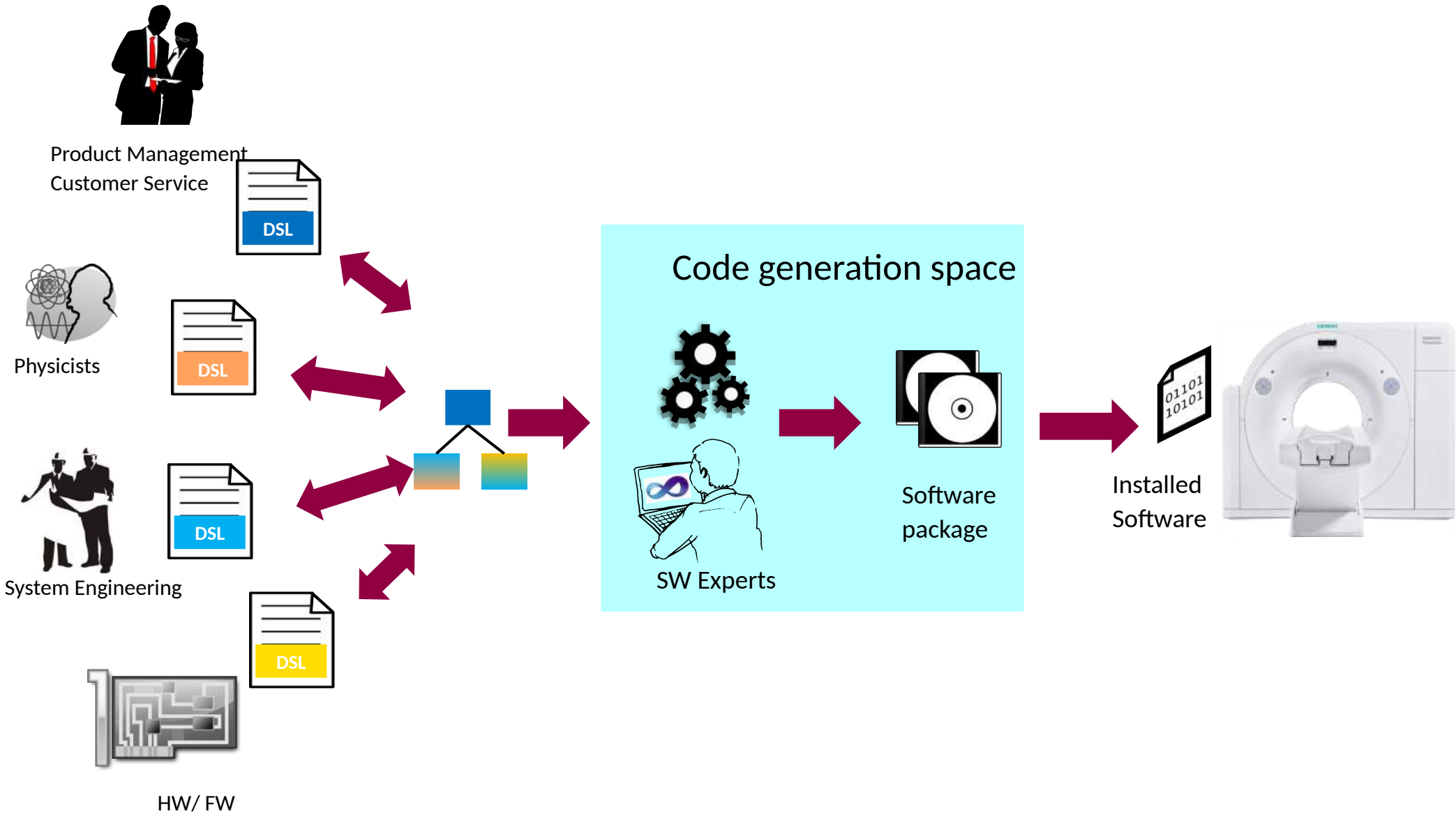
HW/ FW



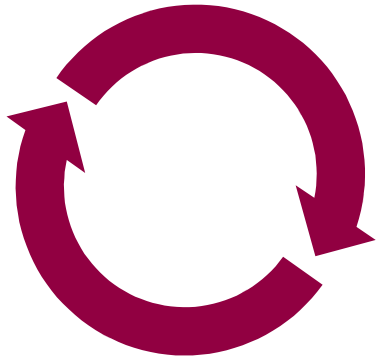
SW Experts

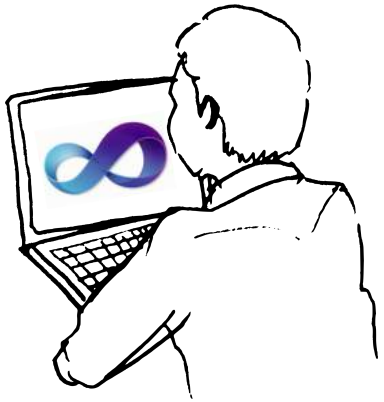
Software
package

Installed
Software

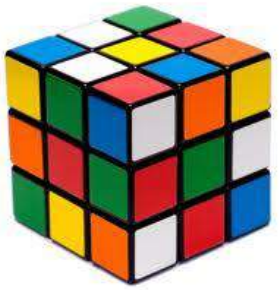

Product Management
Customer Service
DSL
Physicists
DSL
DSL
System Engineering
DSL
HW/ FW
Code generation space
SW Experts
Software
package
01101
10101
Installed
Software

Domain Experts



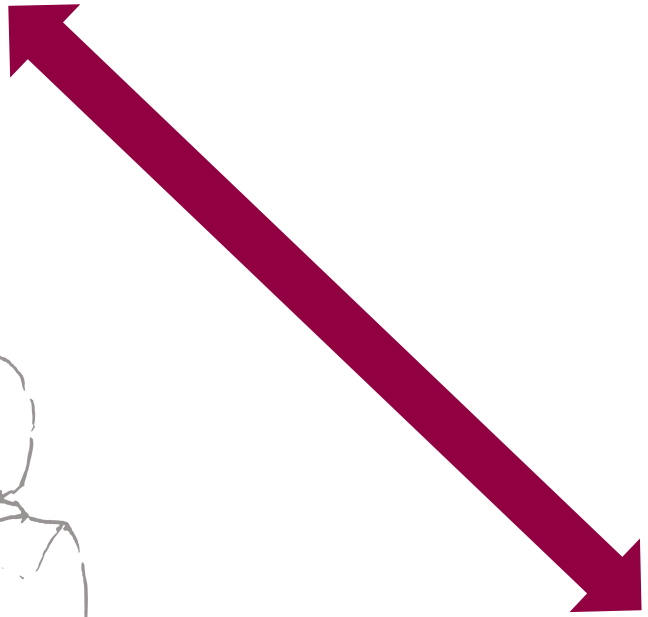


SW Experts



Scanner
Model / SW

Domain Experts



On demand
(automatically)

SW Experts

Scanner
Model / SW

Focus T01

Catalogue T01_Rules **imports** platform_Definitions

Default clinical case: <no defaultClinicalCase>

collimation tabular rule: T01 Slot plate and coll.

description: this is a description text

	Collimation [mm]	Readout DMS [mm]	Preprocessed data matrix [mm]	Collimator Opening [mm]	Soft opening inner slot plate [mm]	Soft opening outer slot plate [mm]	Focus
(1) "30x0.1"	30x0.1	34x0.1	30x0.1	1300.000	n/a	n/a	not so big, big
(2) "30x0.1Mg"	30x0.1	34x0.1	30x0.1	1300.000	n/a	n/a	not so big, big
(3) "30x0.1AgMg"	30x0.1	34x0.1	30x0.1	1300.000	n/a	n/a	not so big, big
(4) "4x0.1"	4x0.1			n/a	1	1.1	not so big
(5) "4x0.1Mg"	4x0.1			n/a	1	1.1	not so big
(6) "3x1"	3x1	32x0.1	32x0.1	n/a	1	1.1	not so big, n/a, big
(7) "1x3"	1x3	32x0.1	32x0.1	n/a	2	2.2	not so big, big
(8) "1x3Mg"	1x3	32x0.1	32x0.1	n/a	3	3.3	not so big, big

Error: parameter must be in range

formula: Formula with Parameter References

alias: S_1

description: This is a text to decribe the formula.

$$S_1 = \text{Reconstructed Slice Width} * 4 + \sum_{i=3}^{\text{Gantry rotation time}} \left(\frac{N_s}{3}\right) + \left(\cos\left(\text{Number of preped slices}\right) + \text{Tube Voltage}\right)$$

tabular rule: Tube Voltage and Filtration

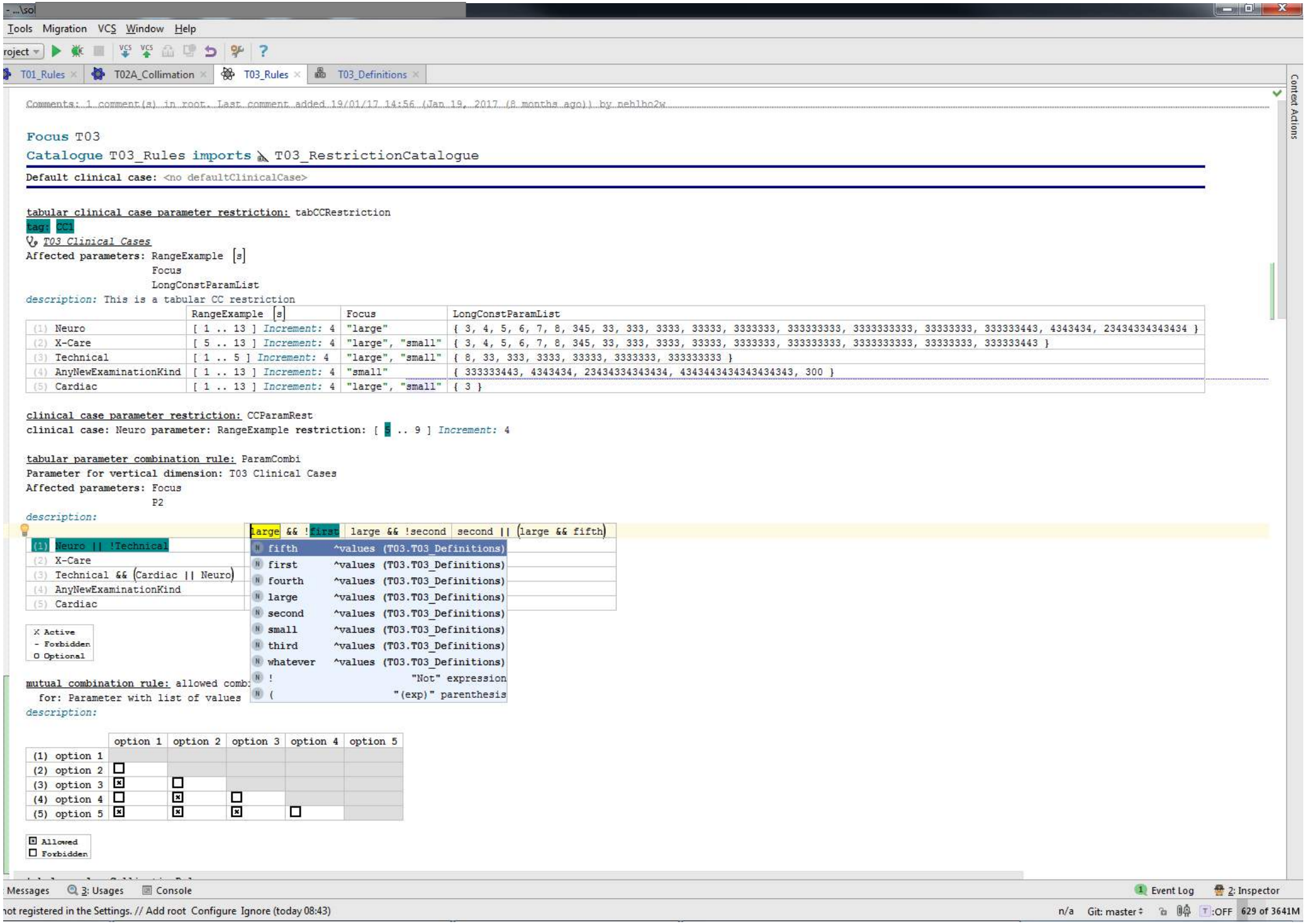
description:

	Spectral Filtration	Tube Voltage [kV]
1 <>	None	7, 8, 9, 10, 11, 12, 13, 14
2 <>	Mg	10, 11, 12, 13, 14
3 <>	AgMg	12, 14

tabular rule: Selectable wedge and filtration

description:

	Wedge Filter	Spectral Filtration	platform_Clinical_Cases
1 <>	f1	None	Regular, Topogram
2 <>	f1+f2	None	Case3, Case8, Case4, Case5
3 <>	f1	Mg	Topogram, Case4, Case6
4 <>	f1+f2	Mg	Case8, Topogram, Case4, Case6, Case5
5 <>	f1	AgMg	Case9

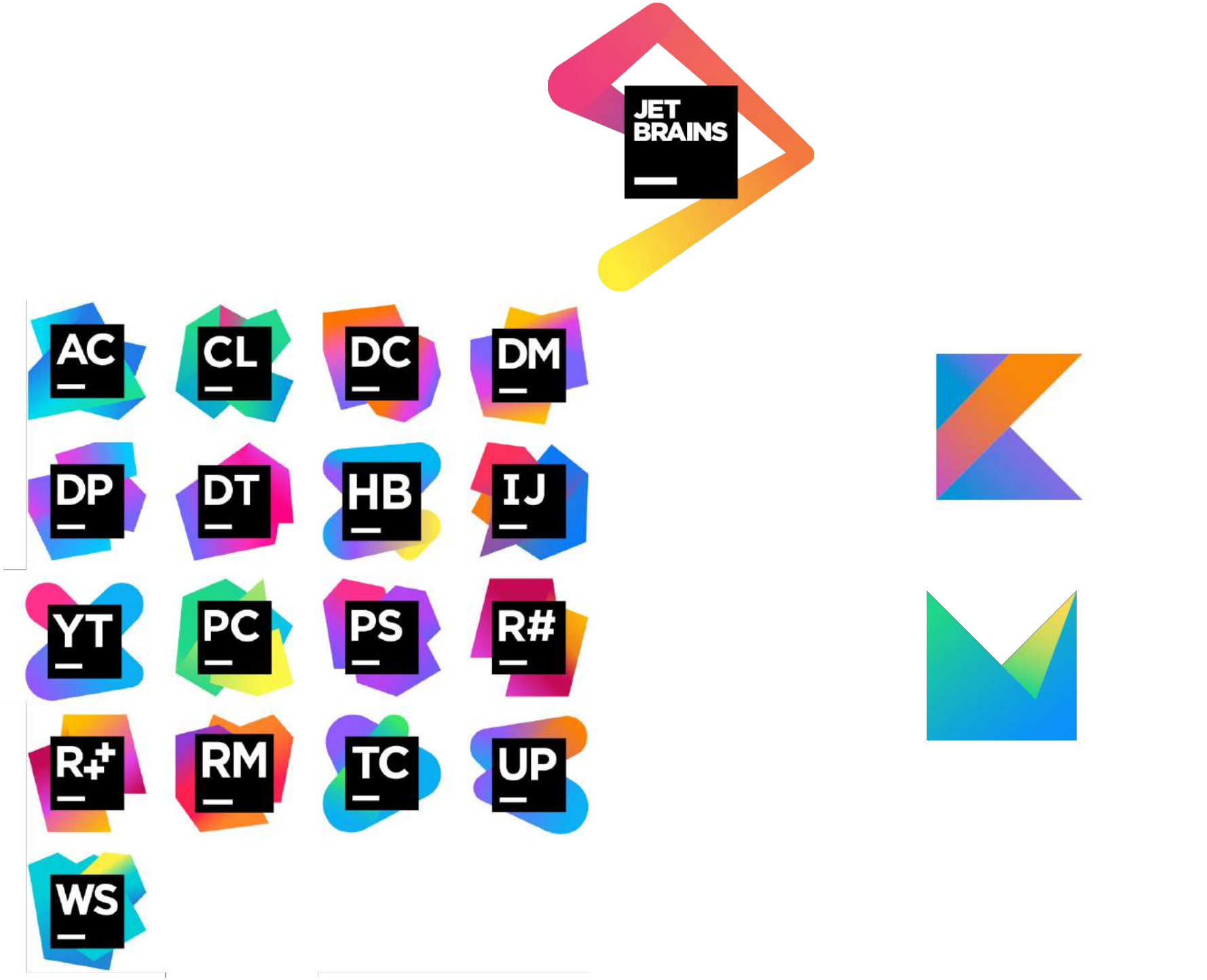

Tools Migration VCS Window Help
T01_Rules
T02A_Collimation
T03_Rules
T03_Definitions
Comments: 1 comment(s) in root. Last comment added 19/01/17 14:56 (Jan 19, 2017 (8 months ago)) by mehlho2w
Focus T03
Catalogue T03_Rules imports T03_RestrictionCatalogue
Default clinical case: <no defaultClinicalCase>
tabular clinical case parameter restriction: tabCCRestriction
tag: CC1
T03 Clinical Cases
Affected parameters: RangeExample [s]
Focus
LongConstParamList
description: This is a tabular CC restriction
RangeExample [s]
Focus
LongConstParamList
(1) Neuro
[1 .. 13] Increment: 4
"large"
{ 3, 4, 5, 6, 7, 8, 345, 33, 333, 3333, 33333, 3333333, 333333333, 3333333333, 33333333, 333333443, 4343434, 23434334343434 }
(2) X-Care
[5 .. 13] Increment: 4
"large", "small"
{ 3, 4, 5, 6, 7, 8, 345, 33, 333, 3333, 33333, 3333333, 333333333, 3333333333, 33333333, 333333443 }
(3) Technical
[1 .. 5] Increment: 4
"large", "small"
{ 8, 33, 333, 3333, 33333, 3333333, 333333333 }
(4) AnyNewExaminationKind
[1 .. 13] Increment: 4
"small"
{ 333333443, 4343434, 23434334343434, 43434434343434343, 300 }
(5) Cardiac
[1 .. 13] Increment: 4
"large", "small"
{ 3 }
clinical case parameter restriction: CCParamRest
clinical case: Neuro parameter: RangeExample restriction: [s .. 9] Increment: 4
tabular parameter combination rule: ParamCombi
Parameter for vertical dimension: T03 Clinical Cases
Affected parameters: Focus
P2
description:
large && !first
large && !second
second || (large && fifth)
(1) Neuro || !Technical
(2) X-Care
(3) Technical && (Cardiac || Neuro)
(4) AnyNewExaminationKind
(5) Cardiac
fifth ^values (T03.T03_Definitions)
first ^values (T03.T03_Definitions)
fourth ^values (T03.T03_Definitions)
large ^values (T03.T03_Definitions)
second ^values (T03.T03_Definitions)
small ^values (T03.T03_Definitions)
third ^values (T03.T03_Definitions)
whatever ^values (T03.T03_Definitions)
! "Not" expression
("(exp)" parenthesis
X Active
- Forbidden
O Optional
mutual combination rule: allowed comb
for: Parameter with list of values
description:
option 1
option 2
option 3
option 4
option 5
(1) option 1
(2) option 2
(3) option 3
(4) option 4
(5) option 5
Allowed
Forbidden
Context Actions
Messages
3: Usages
Console
Event Log
2: Inspector
not registered in the Settings. // Add root Configure Ignore (today 08:43)
n/a
Git: master
:OFF
629 of 3641M

When to DSL?

- Complex domain knowledge
- Painful translation of expertise from a domain specialist to a programmer
  - Gap in abstraction level
- Not directly related to programming
  - biology, math, insurance
  - Verification, analysis, simulation.
- Classes of applications
  - Software factories

Typical risks

- Language design takes effort
  - Adds to the cost of the project → need for reuse
- Language design skills
  - Steep learning curve
  - What goes into the language
  - How to make it elegant
- Proper tooling


JET
BRAINS
AC
CL
DC
DM
DP
DT
HB
IJ
YT
PC
PS
R#
R+
RM
TC
UP
WS

MPS is an open-source language workbench for DSL development

DSM isn't new


Xtext


{S} spoofax


OMG
OBJECT MANAGEMENT GROUP


INTENTIONAL


MetaCase

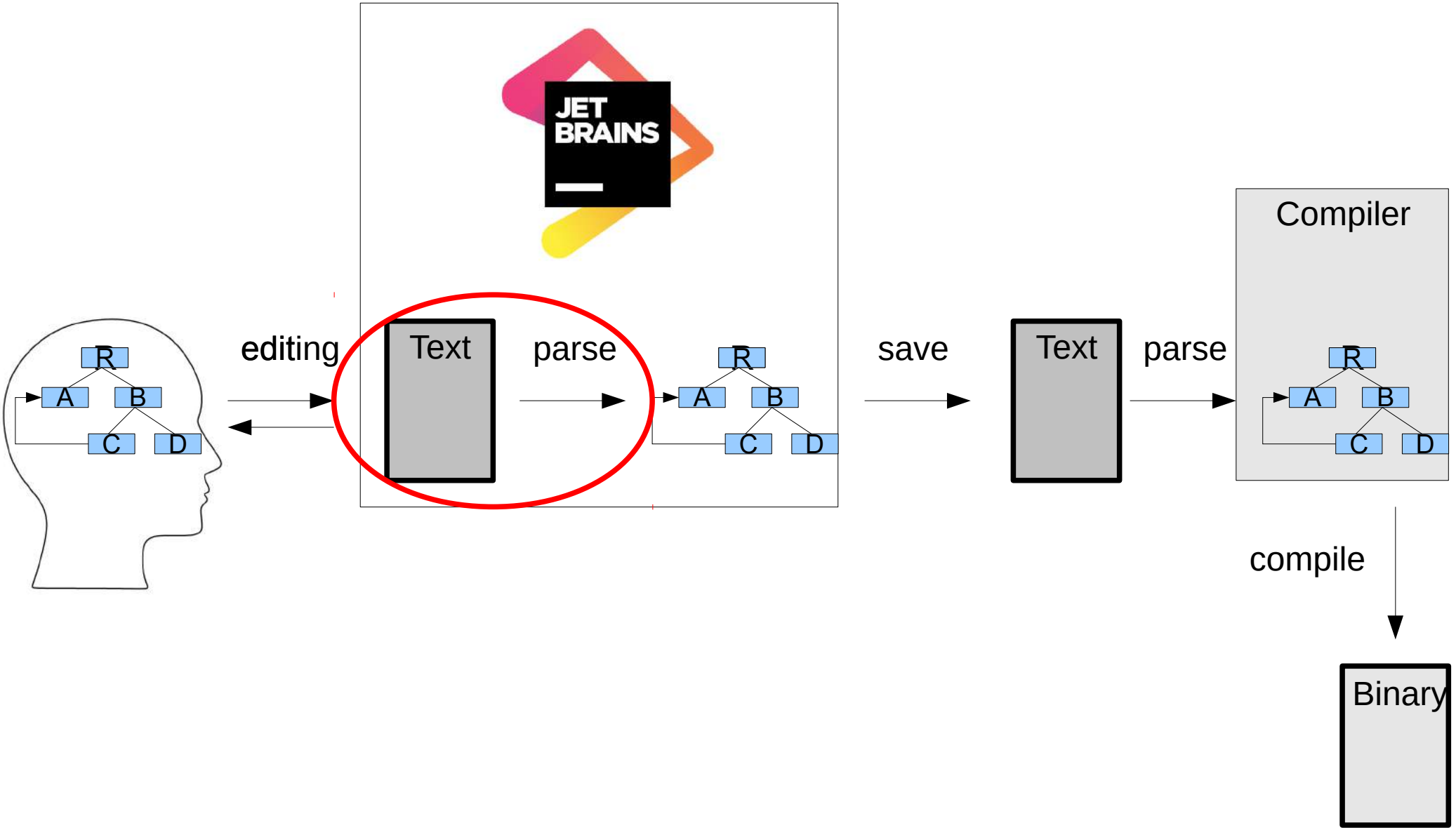
DSLs with MPS

- Abstract Syntax
- Concrete Syntax
- M2M, M2T
- Semantics: Typesystem, Dataflow
- IDE integration, UI
- Tooling: build, plugins
- Evolution/migration
- Deployment: custom IDE, RCP

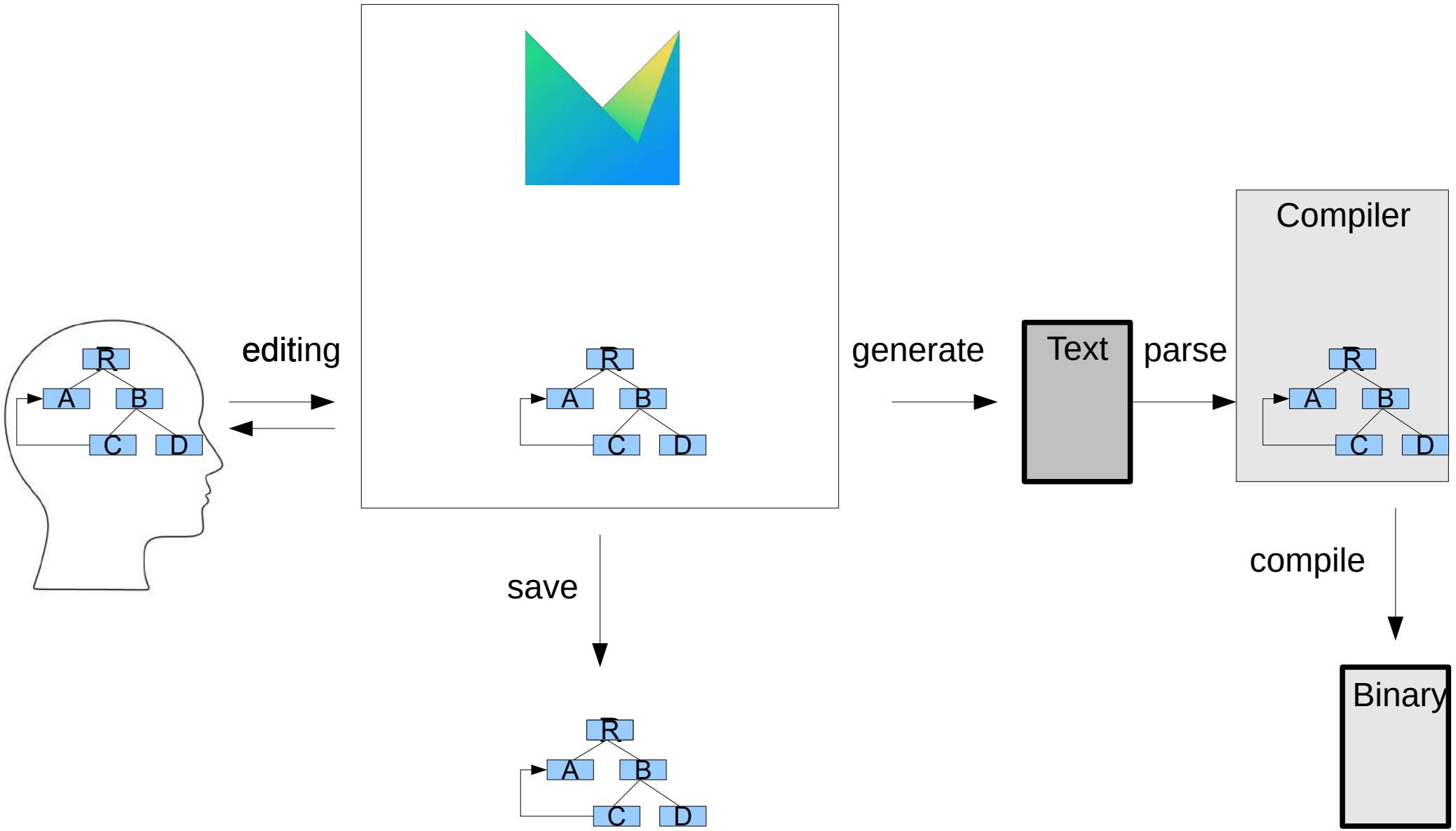
DSLs with MPS

- Abstract Syntax
- Concrete Syntax
- M2M, M2T
- Semantics: Typesystem, Dataflow
- IDE integration, UI
- Tooling: build, plugins
- Evolution/migration
- Deployment: custom IDE, RCP

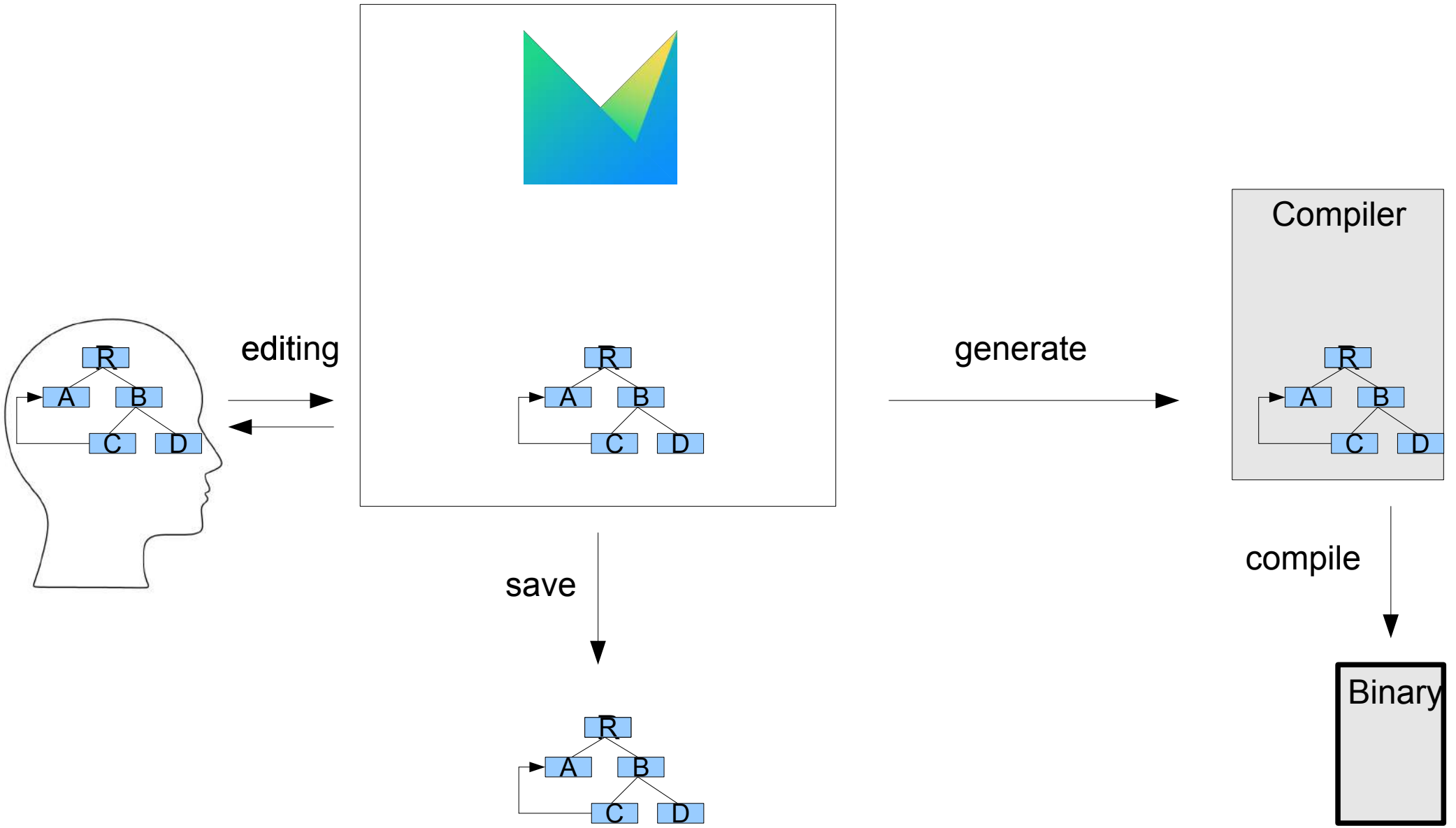
Textual editing



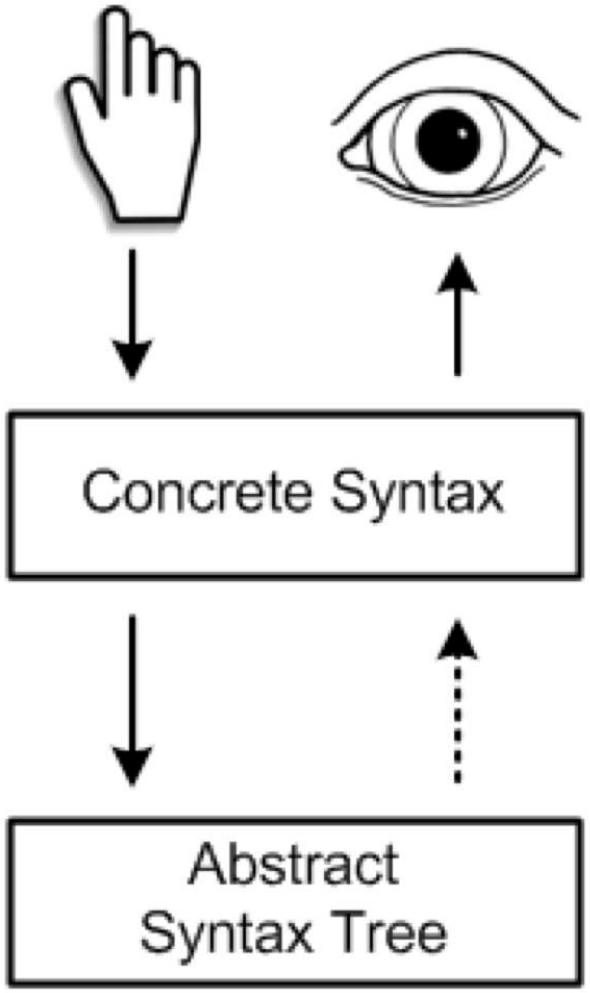
Projectional editing

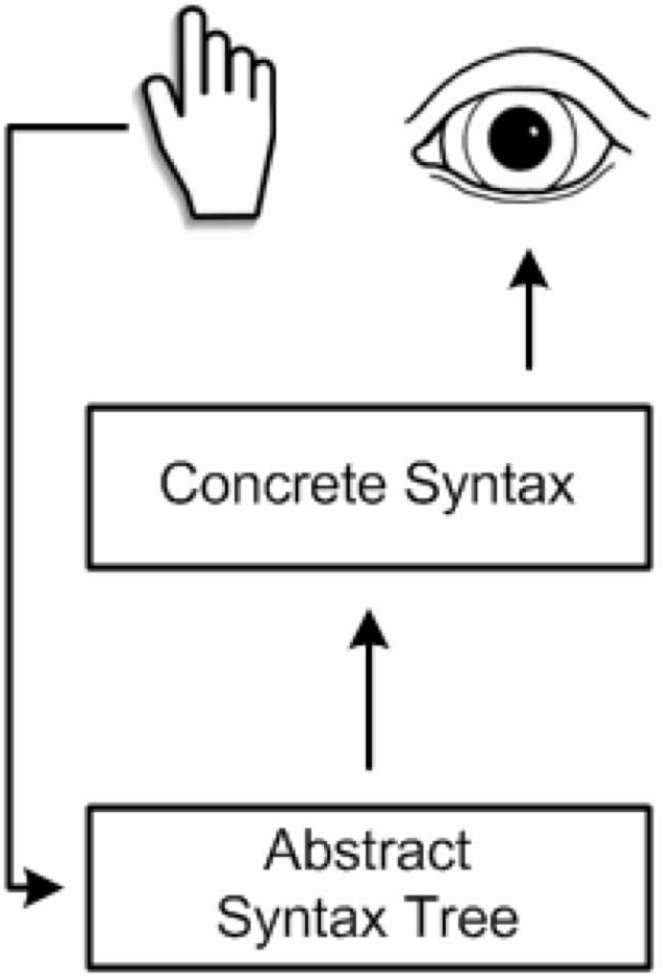


Projectional editing



Projectional editing





Rich syntaxes

```
public class Gol {
  public static void main(String[] args) {
    new Gol().run();
  }
  public void run() {
    sequence<Coordinate> generation = new arraylist<Coordinate>{(4, 5), (5, 5), (6, 5)};
    for (int i = 0; i < 5; i++) {
      System.out.println("Next generation: " + generation);
      generation = nextGeneration(generation);
    }
  }
  private sequence<Coordinate> nextGeneration(sequence<Coordinate> generation) {
    set<Coordinate> candidates = new hashset<Coordinate>;
    candidates.addAll(generation);
    generation.forEach({~it => candidates.addAll(neighbors(it)); });
    set<Coordinate> nextGeneration = new hashset<Coordinate>;
    foreach c in candidates {
      if (boolean Default: dead
```

	generation.contains(c)	!generation.contains(c)
countAliveNeighbors(generation, c) < 2	dead	dead
countAliveNeighbors(generation, c) == 2	alive	dead
countAliveNeighbors(generation, c) == 3	alive	alive
countAliveNeighbors(generation, c) > 3	dead	dead

```
) {
        nextGeneration.add(c);
      }
    }
    return nextGeneration;
  }
  private sequence<Coordinate> neighbors(Coordinate cell) {
    ((cell +
```

	left	middle	right
upper	(-1, 1)	(0, 1)	(1, 1)
middle	(-1, 0)	(0, 0)	(1, 0)
lower	(-1, -1)	(0, -1)	(1, -1)

```
) - cell).asSequence;
  }
  private int countAliveNeighbors(sequence<Coordinate> currentGeneration, Coordinate cell) {
    return neighbors(cell).intersect(currentGeneration).size;
  }

}
```

by Jens Nerche @jensnerche

Tabular notations

```
[checked]
exported statemachine FlightAnalyzer initial = beforeFlight {
```

		Events	
		next(Trackpoint* tp)	reset()
States	beforeFlight	[tp->alt > 0 m] -> airborne	
	airborne	[tp->alt == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne { points += VERY_HIGH_SPEED; } [tp->speed > 100 mps && tp->speed <= 200 mps && tp->alt == 0 m] -> airborne { points += HIGH_SP	[ ] -> beforeFlight
	landing	[tp->speed == 0 mps] [tp->speed > 0 mps] - { points--; }	
	landed		
	crashed		

```
}
```

Core Data DefaultRegions **for entity** BillingRegion

Code	Name	Base Min Price	Max Rebate Factor
BW	Baden Württemberg	0.20	0.8
BY	Bayern	0.20	0.8
BE	Berlin	0.15	0.7
BB	Brandenburg	0.10	0.7
HB	Bremen	0.20	0.7
HH	Hamburg	0.15	0.7
HE	Hessen	0.15	0.7
MV	Mecklenburg-Vorpommern	0.10	0.7
NI	Niedersachsen	0.15	0.7
NW	Nordrhein-Westfalen	0.15	0.7
RP	Rheinland-Pfalz	0.15	0.7
SL	Saarland	0.15	0.7
SN	Sachsen	0.10	0.7
ST	Sachsen-Anhalt	0.10	0.7
SH	Schleswig-Holstein	0.15	0.7
TH	Thüringen	0.10	0.7

Symbolic notations

```
int32 sumUpIntArray(int32[] arr, int32 size) {
  return ∑_{i=0}^{size} arr[i];
} sumUpIntArray (function)
```

```
int32 averageIntArray(int32[] arr, int32 size) {
  return (∑_{i=0}^{size} arr[i]) / size;
} averageIntArray (function)
```

```
double midnight1(int32 a, int32 b, int32 c) {
  return (-b + √(b^2 - 4 * a * c)) / (2 * a);
} midnight1 (function)
```

```
double midnight2(int32 a, int32 b, int32 c) {
  return (-b + √(b^2 - ∑_{i=1}^{4} a * c)) / (2 * a);
} midnight2 (function)
```

```
double sumOfProductsOfLogs(int32[] arr, int32 size) {
  return ∑_{k=0}^{size} (∏_{i=0}^{k} log_2 arr[i]) / 2;
} sumOfProductsOfLogs (function)
```

Positional notations

Rule Set Type DemoRuleSetType

Business objects

person : **Person**

Variables:

PRMI : **int**
FR : **int**
NN : **int**
TT : **int**
J : **int**
A3 : **int**
G3 : **int**
ANUI : **int**
X : **int**

Parent

<no parent>

Libraries

Standard
Extra

Rule Set Type DemoRuleSetType

Business objects

<no business objects>

Variables:

<no variables>

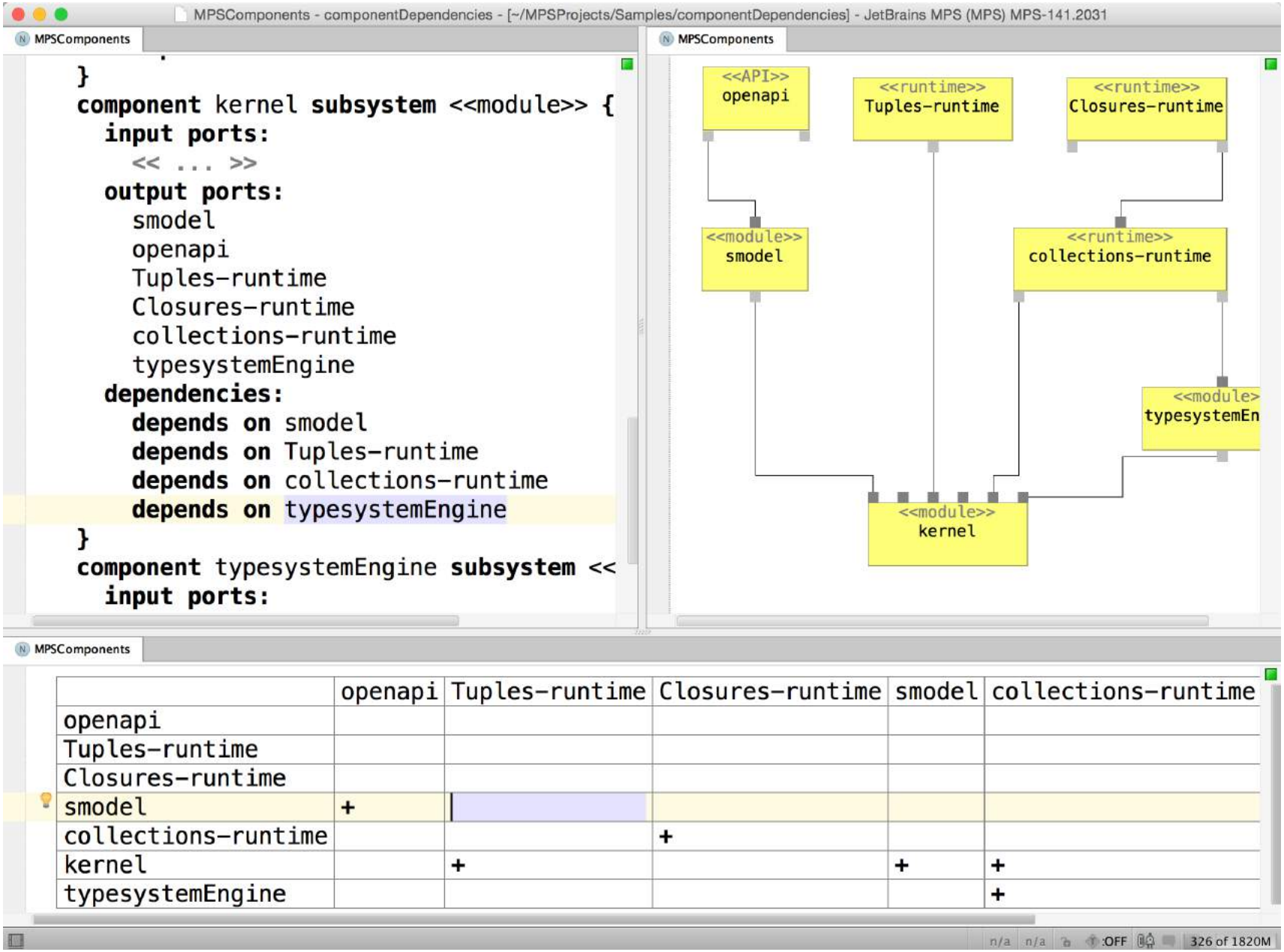
Parent

<no parent>

Libraries

<no libraries>

Multiple switchable notations



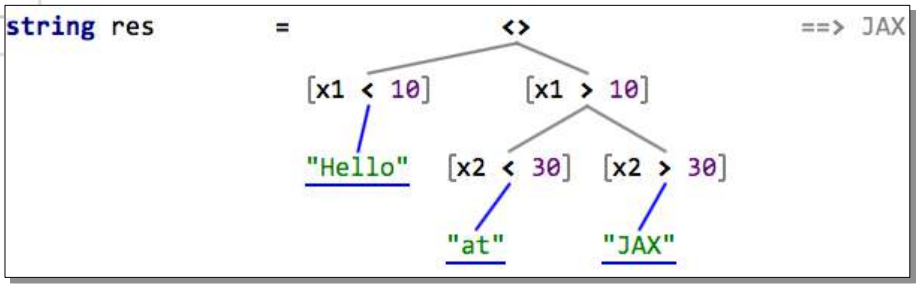
Combine languages

```
variables:
  int x1             = 10 * (1 + 2)                    ==> 30
  int x2             = 20                              ==> 20
  boolean b1         = true || !false                  ==> true
  int b2             = if [ b1 then 12 else 13 ]       ==> 12
  list<int> intList  = list(1, 2, 3)                   ==> [1, 2, 3]
  int three          = intList.last                    ==> 3
  list<int> t2       = intList.where|it > 2|           ==> [3]
  boolean allEl      = intList.all|it > 0|             ==> true
  int surprise2      = doWithTwoInts(:add, 1, 3)       ==> 4
  [int, int] tuple   = [1, 2]                          ==> [1, 2]
  int one            = tuple[0]                        ==> 1
  int c1             = alt [x1 < 0 && x2 > 1  => 2]    ==> 0
                           [x1 > 0 && x2 == 1 => 1]
  int c2             =                                 ==> 9
```

	x1 < 0	x1 == 0	x1 > 0
x2 < 0	1	2	3
x2 == 0	4	5	6
x2 > 0	7	8	9

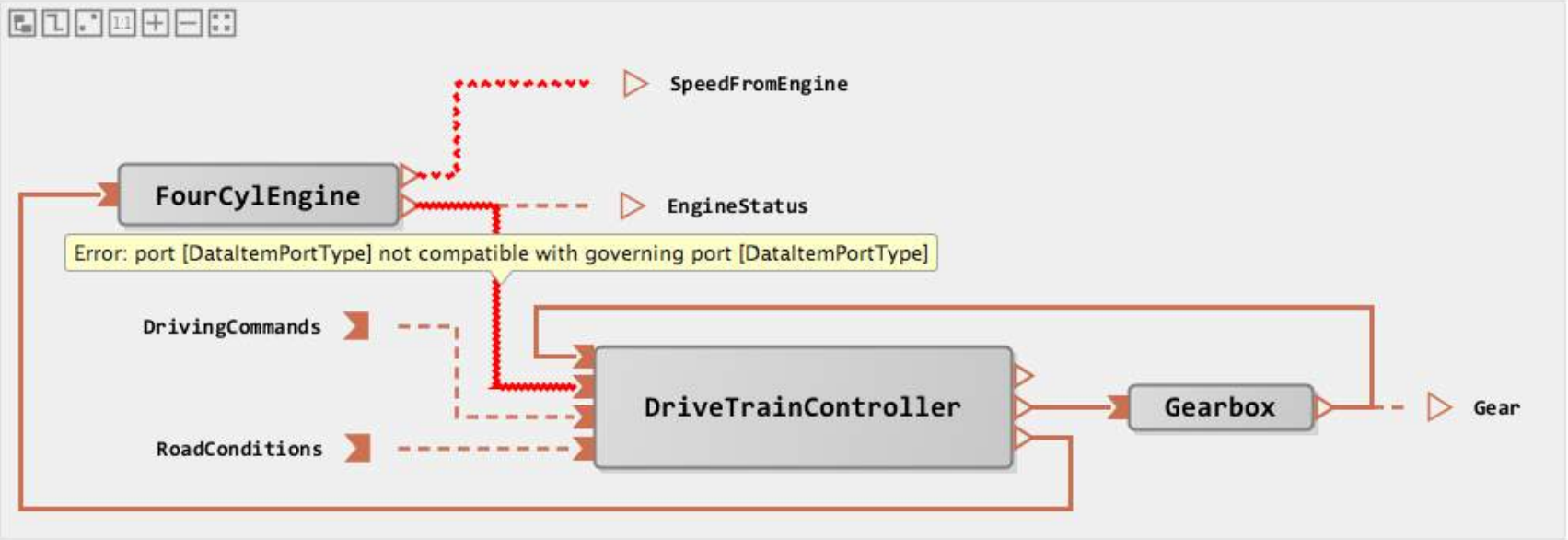
```
  int complicated    = {
                         val t1 = 10 + 20
                         val t2 = t1 + 30
                         t2
                       }
```

```
string res         =                                  ==> JAX
```

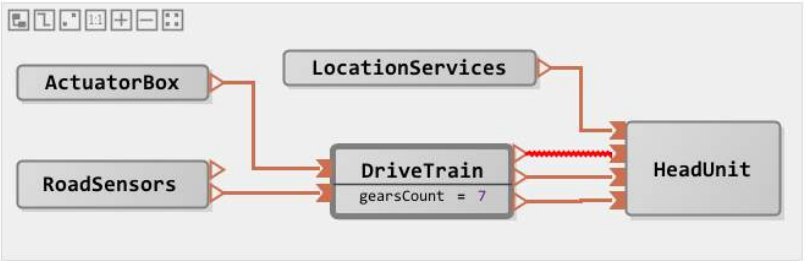



```
functions:
  fun add(int a, int b)                                    : int                = a + b
  fun doWithTwoInts((int, int => int) fun, int a, int b)   : int                = fun.exec(a, b)
  fun anotherFun(option<int> i)                            : int                = with some i as x => x + 1 none 20
  fun giveMeAnInt()                                        : int                = anotherFun(some(10))
  fun getStreets(Person p)                                 : collection<string> = p.workedAt.offices.street
```

```
public functional component DriveTrain {
  produces SpeedFromEngine
  produces EngineStatus
  produces Gear where it < gearsCount
  consumes RoadConditions
  param int gearsCount
  consumes DrivingCommands
```




```
}
functional component Car {
```




```
}
```

Parsing is the bottleneck

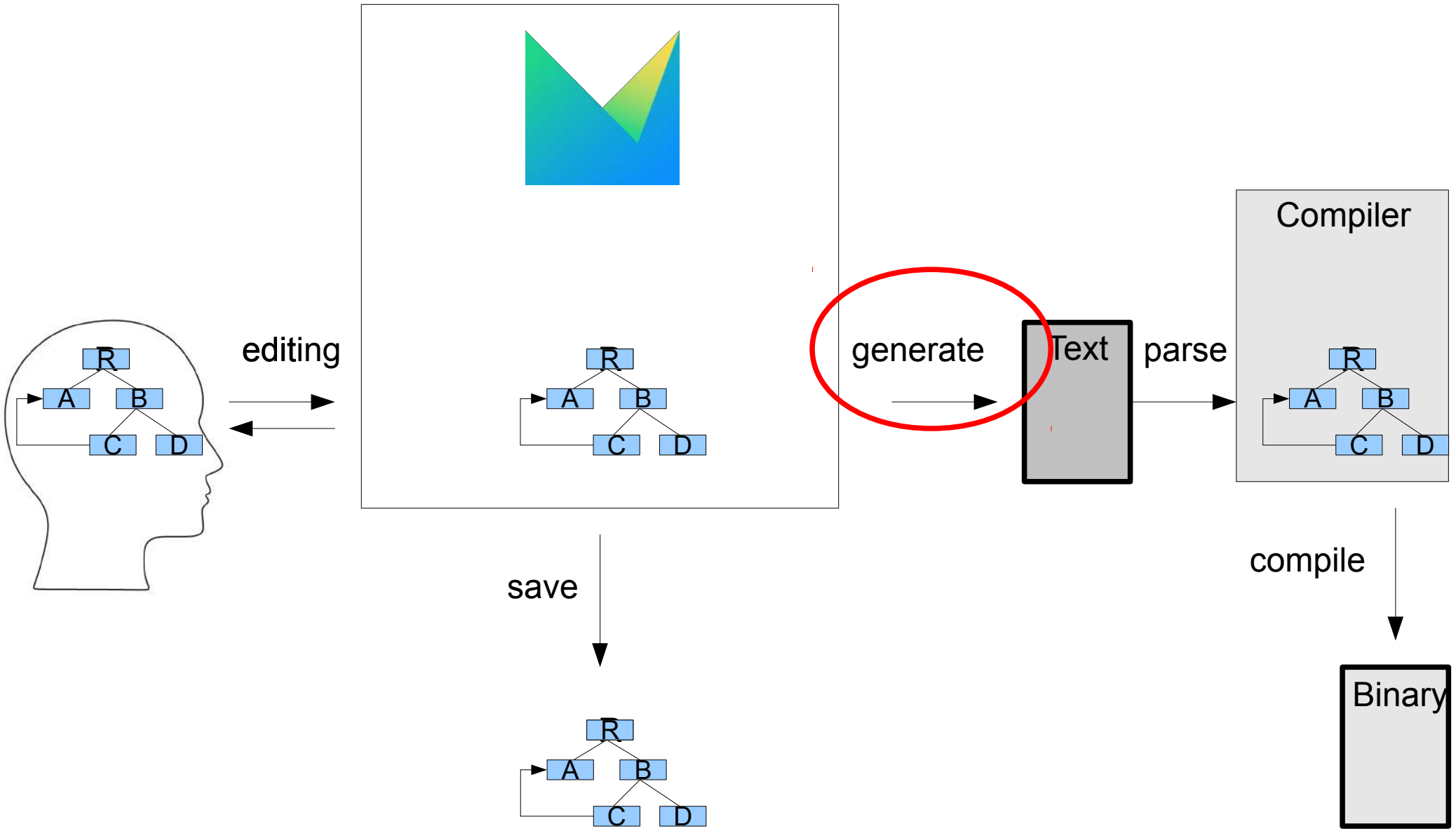
… of language expressiveness

- Limits the **possible** syntaxes
- Allows only **one** editable code visualization
- Complicates **combining** languages

DSLs with MPS

- Abstract Syntax
- Concrete Syntax
- M2M, M2T
- Semantics: Typesystem, Dataflow
- IDE integration, UI
- Tooling: build, plugins
- Evolution/migration
- Deployment: custom IDE, RCP

Code generation



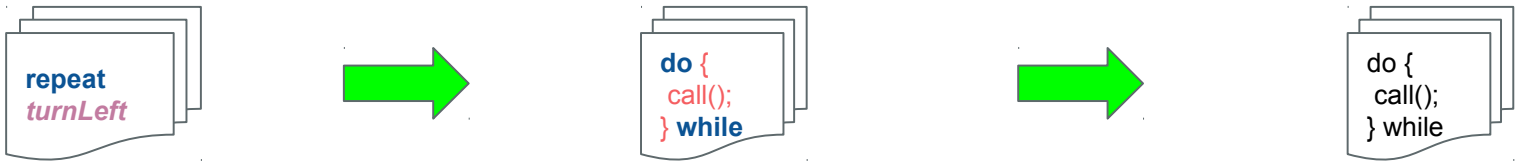
Generators

Map solutions from a **problem domain** to an **implementation domain**

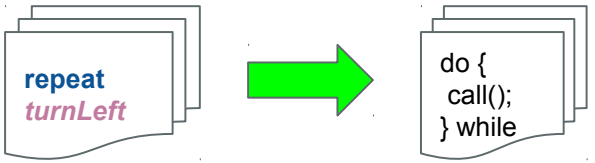
- Transform models
- Output **models** or **text**

Compiler analogy

Source code | Intermediate presentation | Machine code

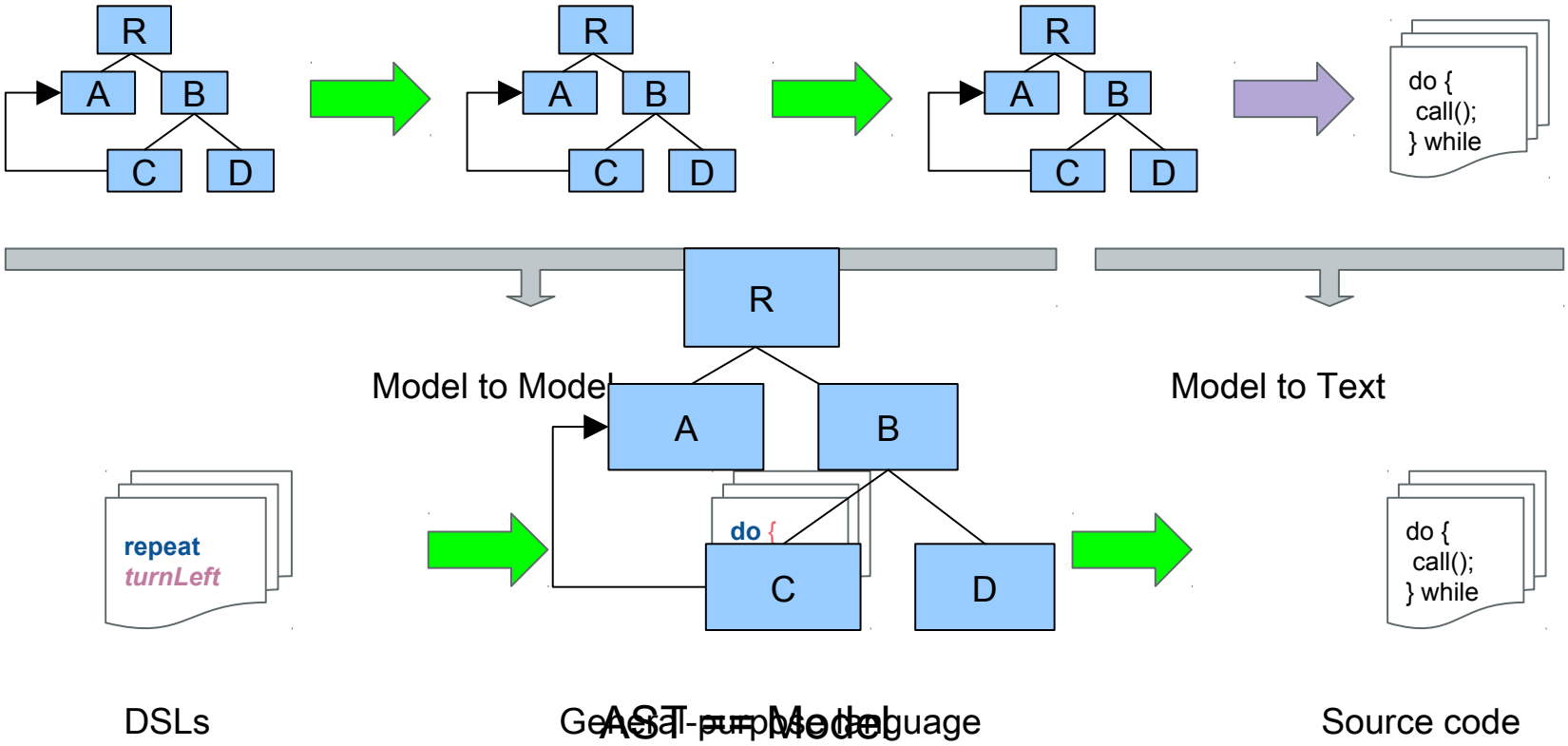


DSLs | General-purpose language | Source code



DSLs | Source code

OMG/MOF Perspective



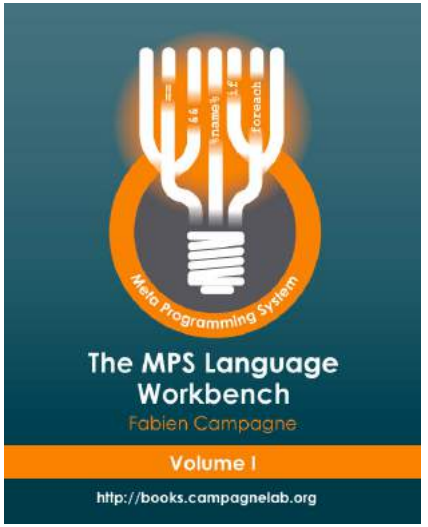

SIEMENS
Mercedes-Benz
UNIVERSITY OF AGDER
fortiss
OHB
BOSCH
HUAWEI
Continental
Whirlpool
CORPORATION
McGill
UNIVERSITY OF
WATERLOO
DYNAGEN
power controls you can trust
SIOUX
Belastingdienst
itemis
i2S
INSURANCE KNOWLEDGE

jetbrains.com/mps

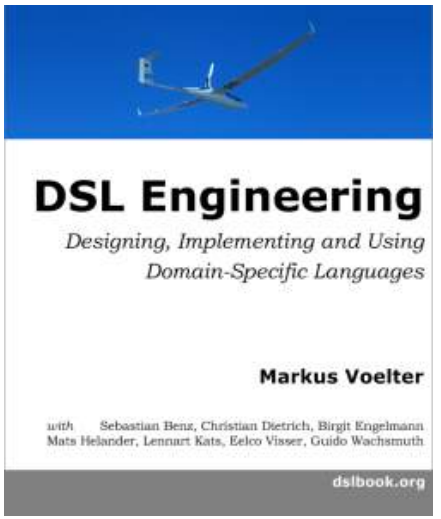
Thank you
for your attention

Books

- http://books.campagnelab.org



- http://dslbook.org



https://www.jetbrains.com/mps/publications