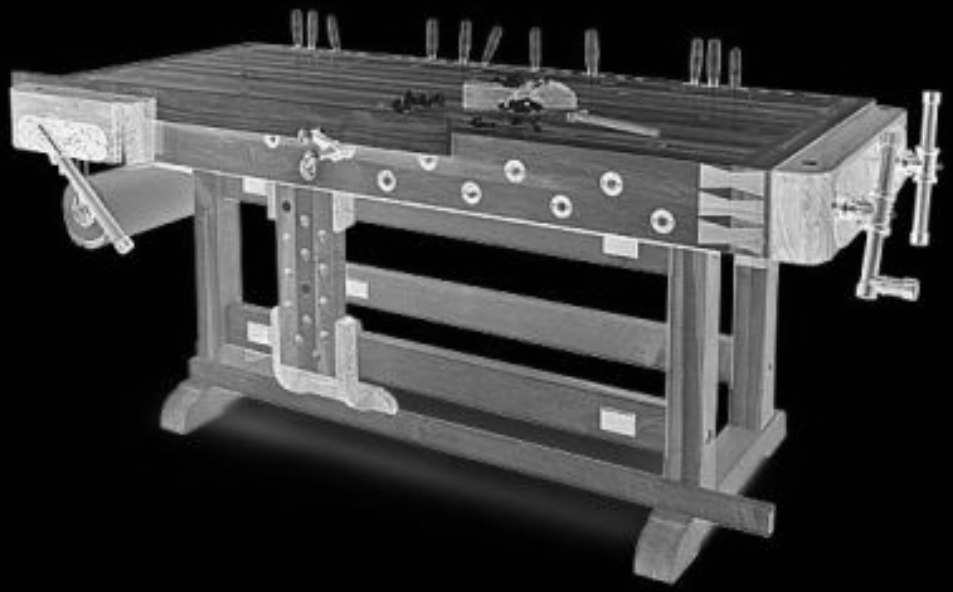
Language Workbenches

The State of the Art in 2015



voelter { ingenieurbüro für softwaretechnologie // itemis

voelter@acm.org
www.voelter.de
@markusvoelter

0 Languages, Models, Programs

1 Language Workbenches

2 Notational Flexibility

3 Language Modularity

4 Specification Languages

5 LWBs and the Web

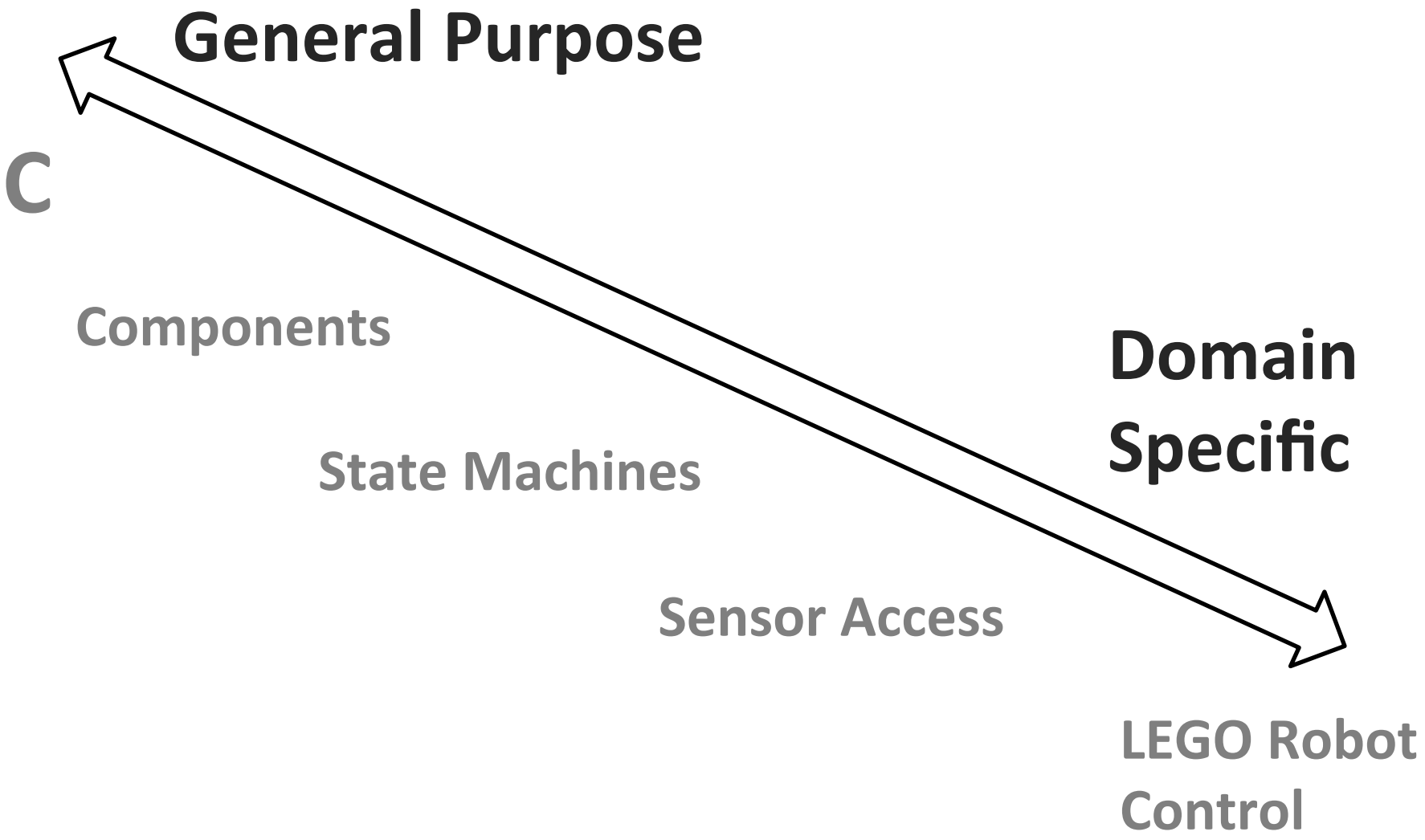
6 Miscellaneous

7 Why you should care

0

Languages, Models, Programs

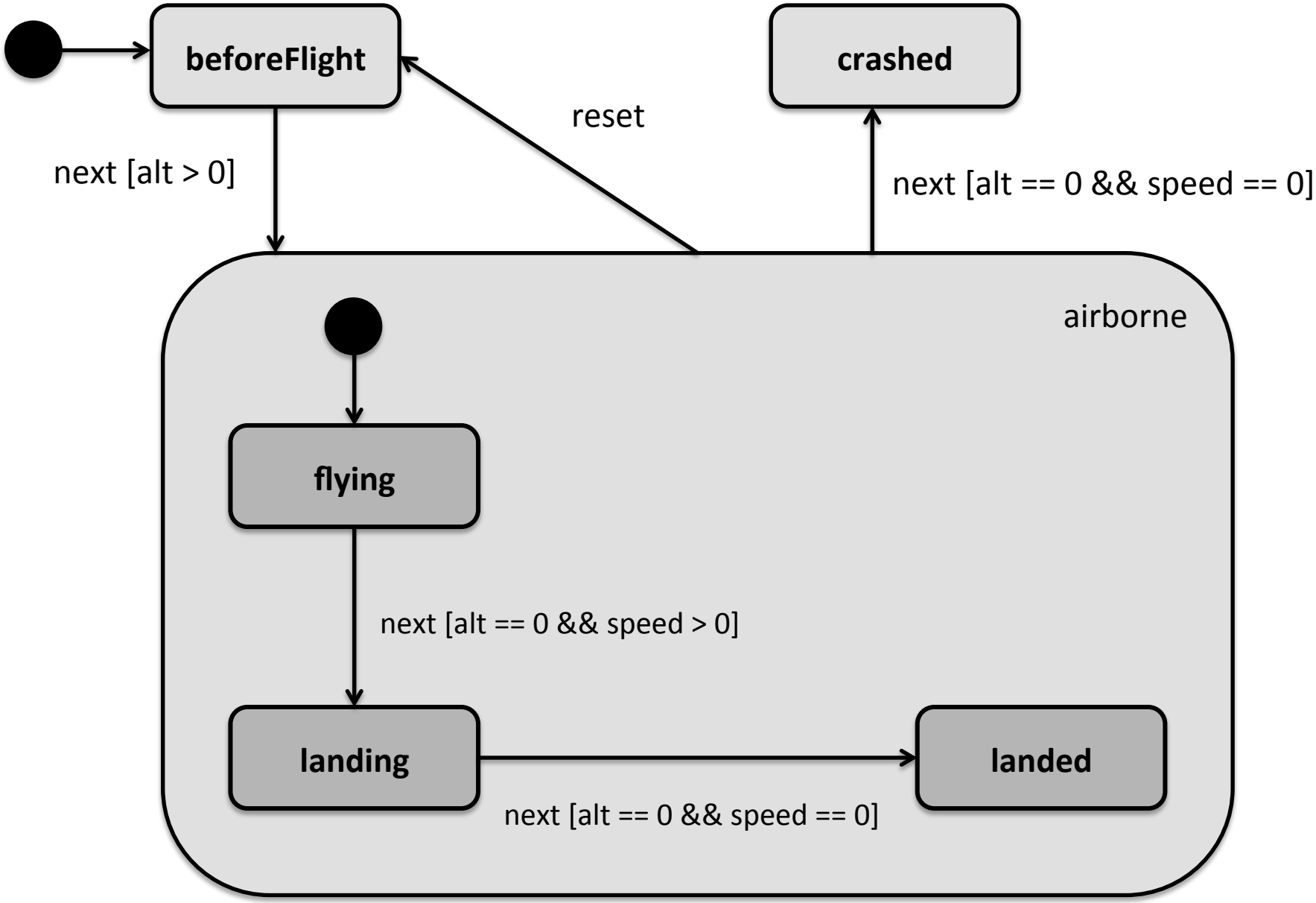
	more in GPLs	more in DSL
Domain Size	large and complex	smaller and well-defined
Designed by	guru or committee	a few engineers and domain experts
Language Size	large	small
Turing-completeness	almost always	often not
User Community	large, anonymous and widespread	small, accessible and local
In-language abstraction	sophisticated	limited
Lifespan	years to decades	months to years (driven by context)
Evolution	slow, often standardized	fast-paced
Incompatible Changes	almost impossible	feasible


General Purpose
C
Components
State Machines
Sensor Access
Domain Specific
LEGO Robot Control

[Model or Code]

```
Trackpoint* makeTP(uint16 alt, int16 speed) {
  static int8 trackpointCounter = 0;
  trackpointCounter++;
  Trackpoint* tp = ((Trackpoint*) malloc(sizeof Trackpoint));
  tp->id = trackpointCounter;
  tp->timestamp = trackpointCounter;
  tp->alt = alt
  tp->speed = speed
  return tp;
}
```

[Model or Code]



[Model or Code]

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next()
  in reset()
  out crashNotification() -> raiseAlarm
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne
      on next [tp->speed > 100 mps] -> airborne
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing
    }
    state landed {
    }
  }
  state crashed {
  }
}
```

[Model or Code]

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next(Trackpoint* tp)
  in reset()
  out crashNotification() -> raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
    exit { points += TAKEOFF; }
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight { points = 0; }
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne { points += VERY_HIGH_SPEED; }
      on next [tp->speed > 100 mps] -> airborne { points += HIGH_SPEED; }
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing { points--; }
    }
    state landed {
      entry { points += LANDING; }
    }
  }
  state crashed {
    entry { send crashNotification(); }
  }
}
```

Does it really matter?

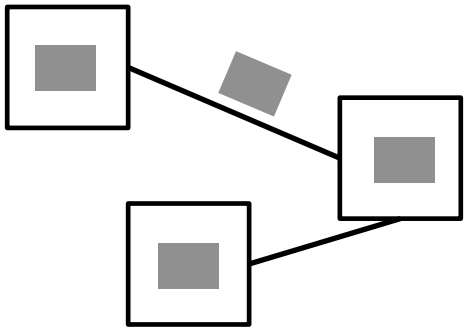
What is the difference?

Who cares?

[Model or Code: Notation!]

Textual

Graphical



Tabular

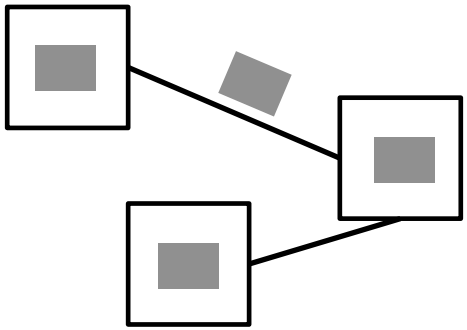
Mathematical

[Notation]

One important aspect of languages.

Textual

Graphical

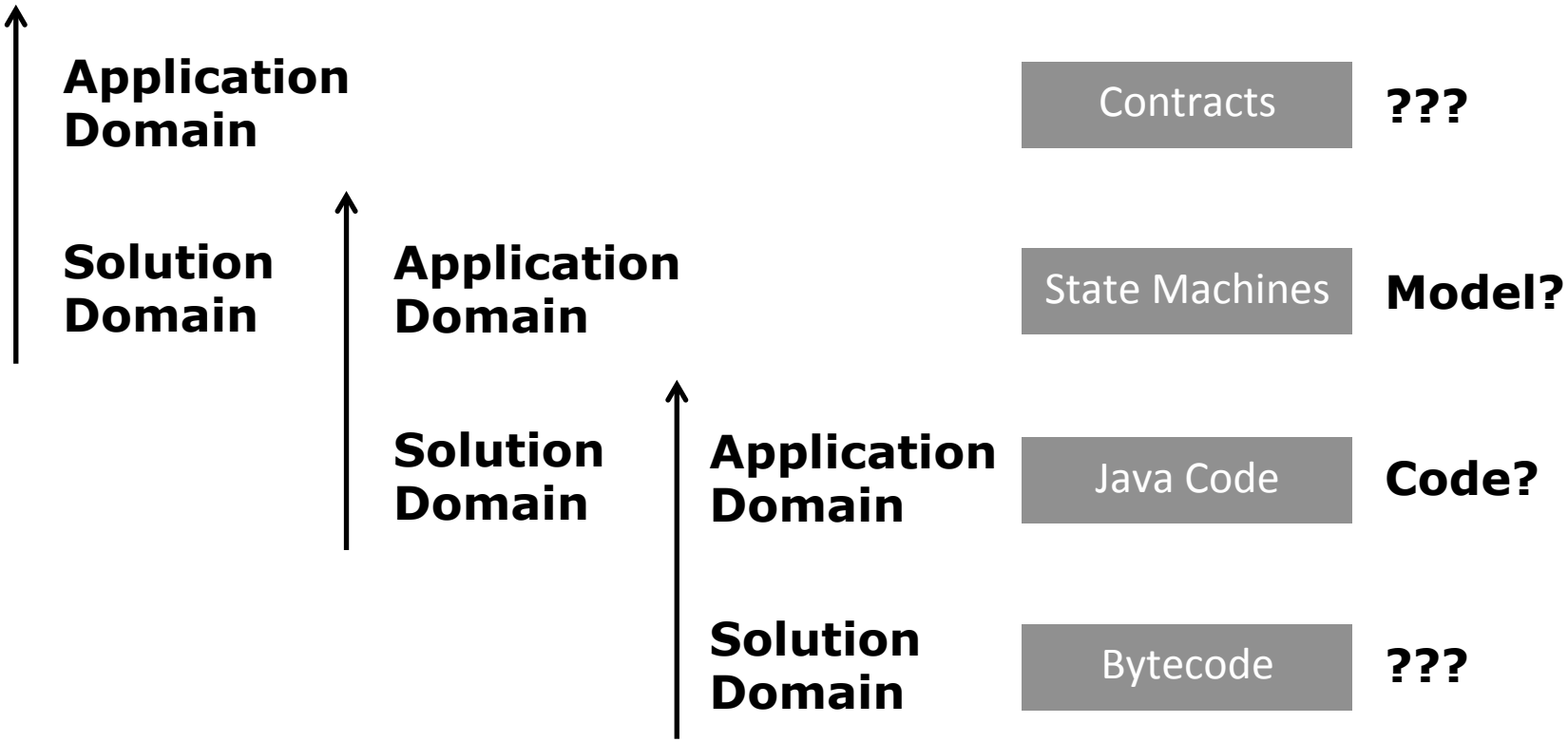


Tabular

Mathematical

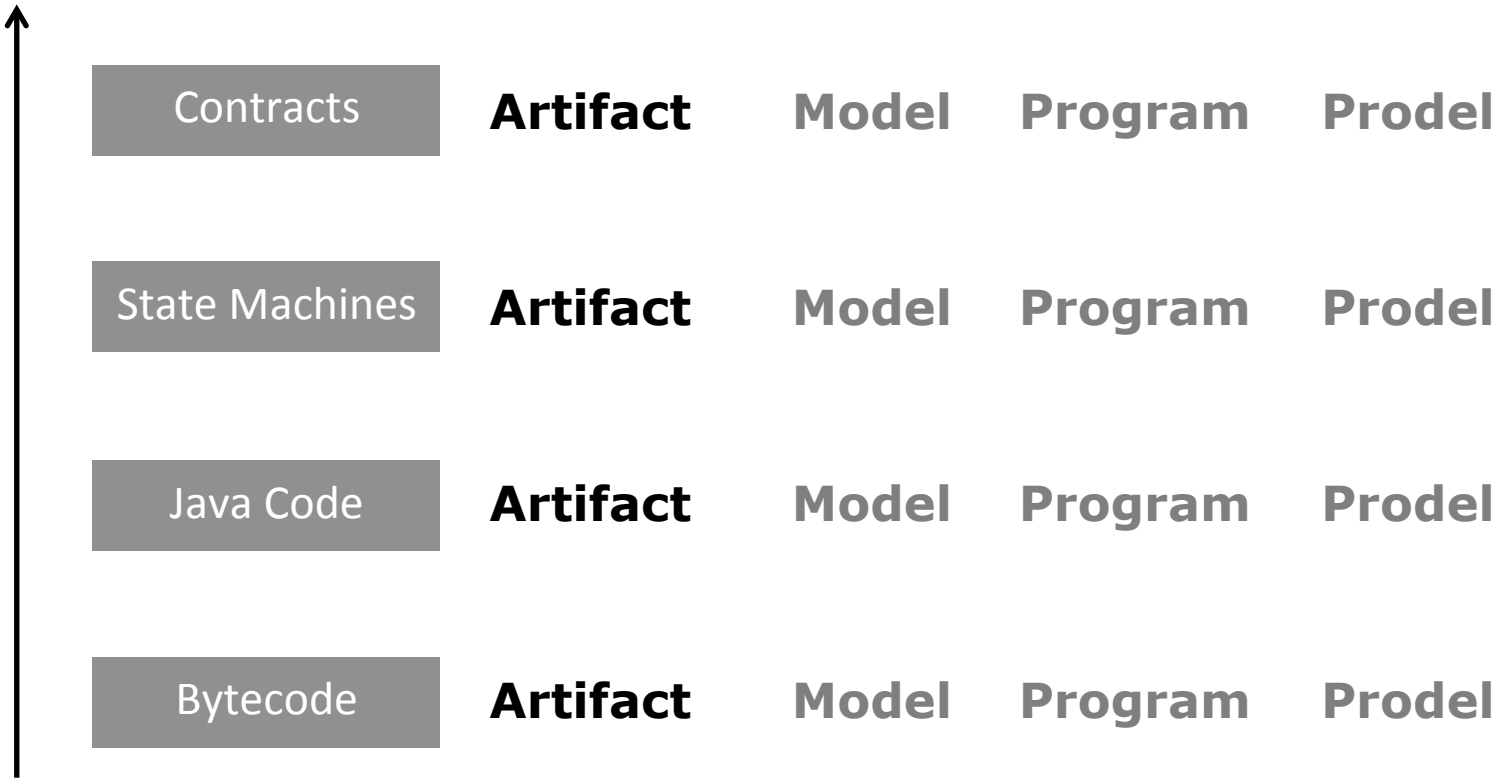
[Abstraction]

The other important aspect.



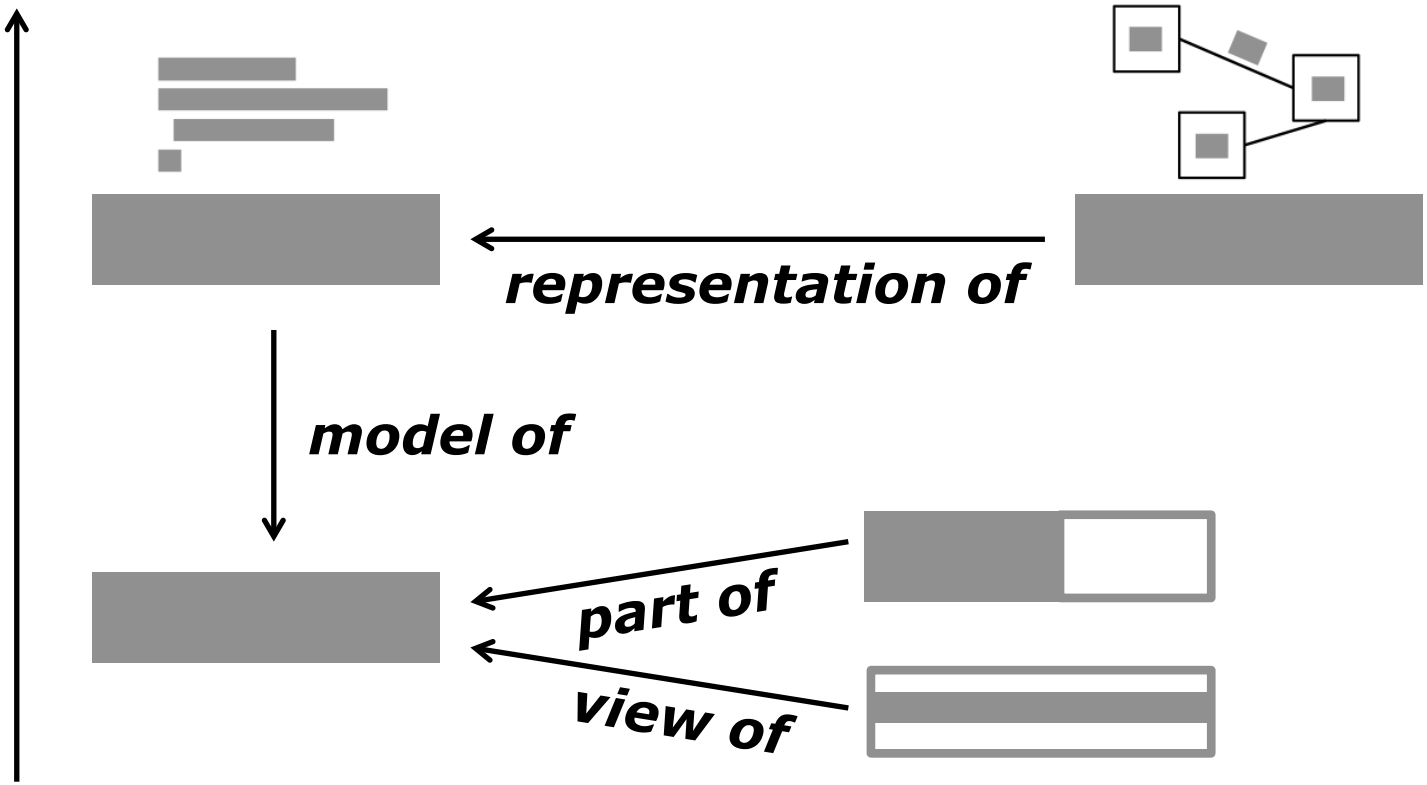
[Abstraction]

The other important aspect.



[Relationships]

[Relationships]



[Relationships]

model of Abstraction

part of Hierarchy, Partitioning

view of Concern, Aspect, Viewpoint

representation of
Presentation, Notation, Form

manifesto

man·i·fes·to \ˌma-nə-ˈfes-(ˌ)tō

noun

a written statement declaring publicly the intentions, motives, or views of its issuer.

INVOLVE STAKEHOLDERS
SEPARATE CONCERNS
AUTOMATE WHAT YOU MUST
USE APPROPRIATE ABSTRACTIONS
USE FRONT-LOADING

Model-Based Engineering • Model-Driven (Software) Development • Domain-Specific Languages • Language-Oriented Programming

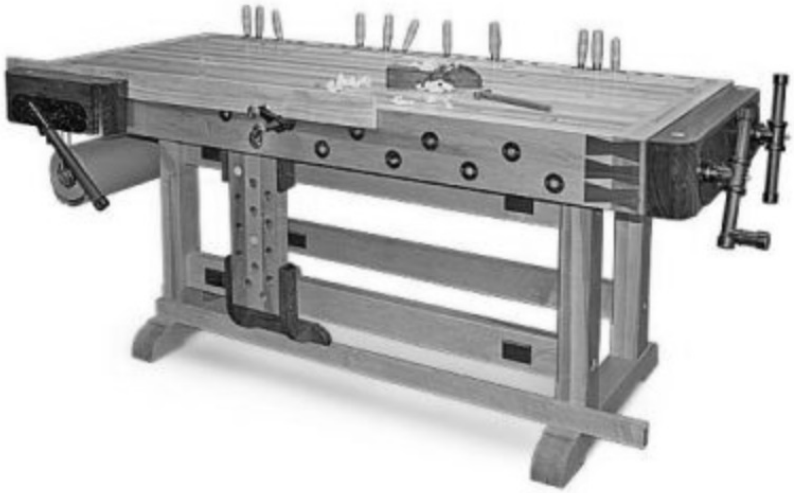
1st

Language Workbenches

Language Workbench(es)

... are the tools to do:

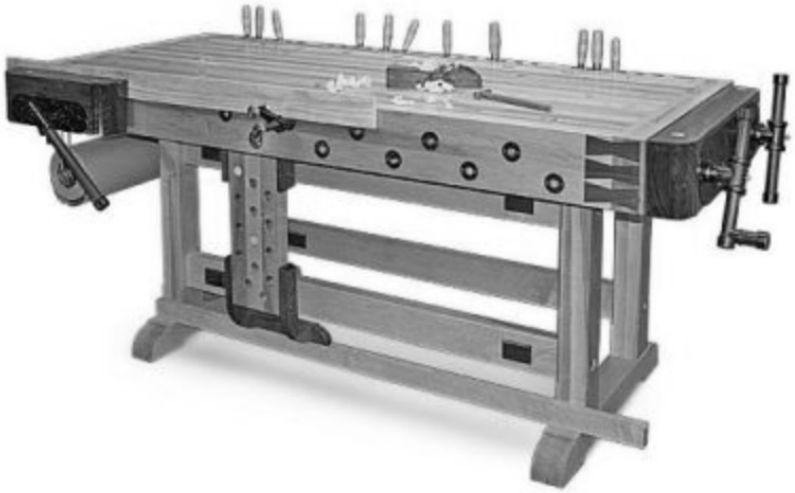
Model-Based Engineering • Model-Driven (Software) Development • Domain-Specific Languages • Language-Oriented Programming



Language Workbench

(Martin Fowler)

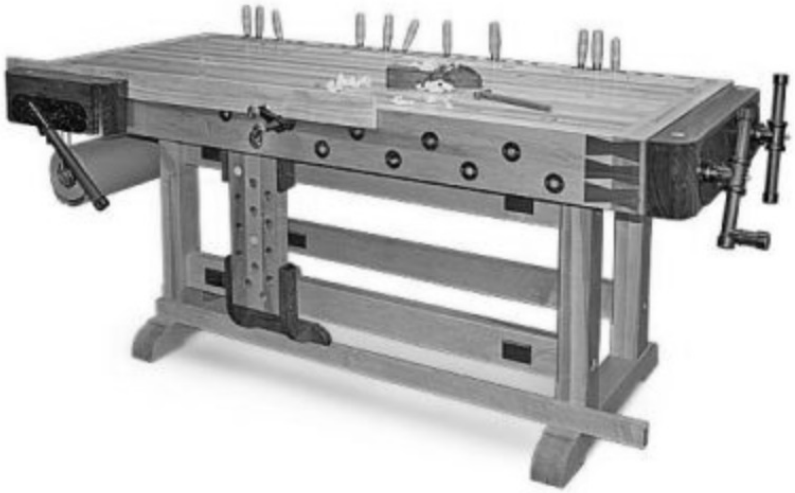
Freely
define
languages and
integrate
them



Language Workbench

(Martin Fowler)

use
persistent
abstract
representation



Language Workbench

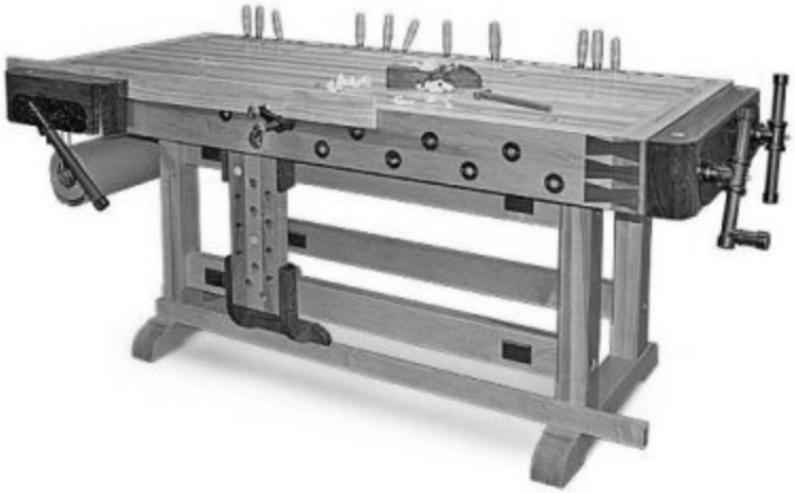
(Martin Fowler)

language ::=

schema

+ editors

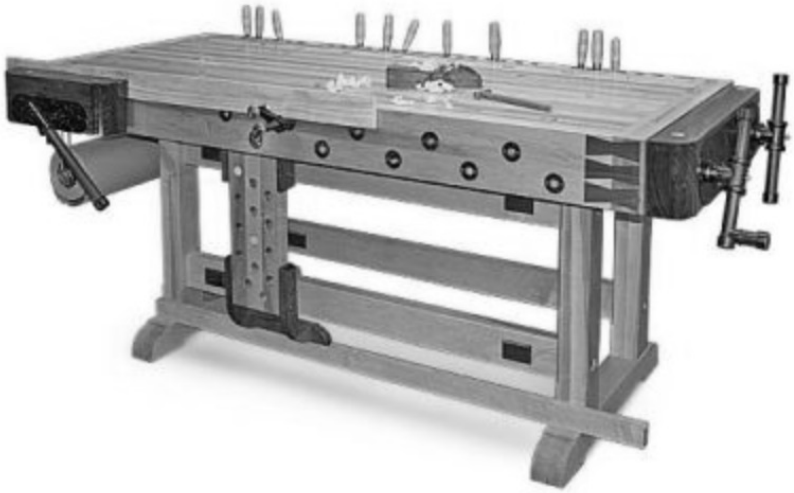
+ generators



Language Workbench

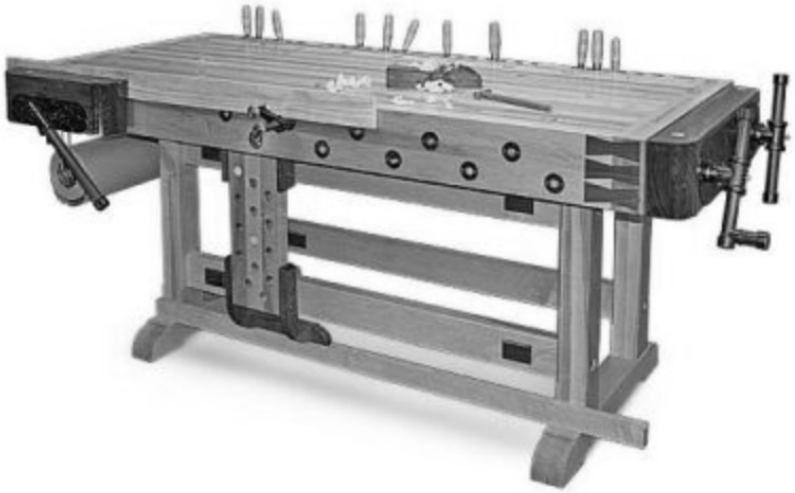
(Martin Fowler)

projectional editing ?



Language Workbench

(Martin Fowler)



persist

incomplete

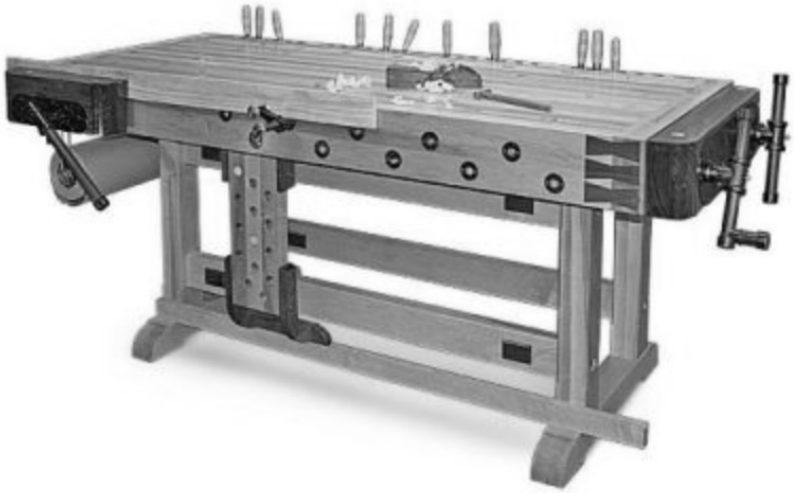
or

contradictory

information

Language Workbench

(Martin Fowler)



powerful

editing

testing

refactoring

debugging

groupware

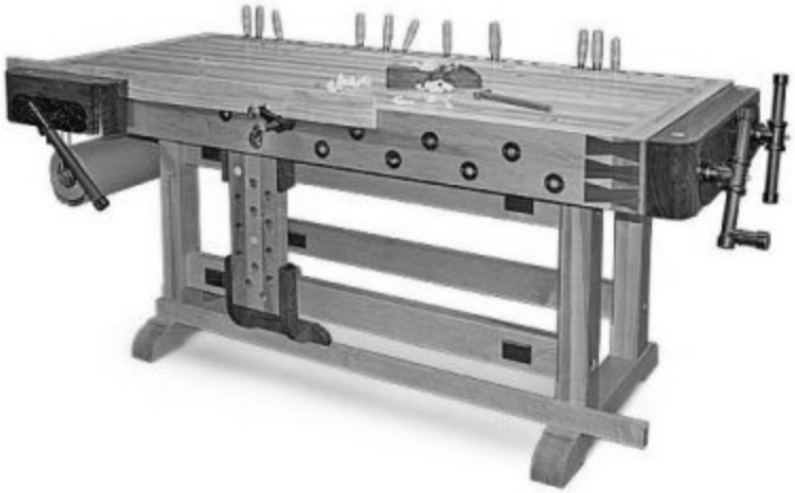
language definition

implies

IDE definition

Language Workbench

(Martin Fowler)



support for

„classical“ **programming**

„classical“ and **modeling**

Language Workbench Challenge

Comparing Tools of the Trade

About LWC | Workshop schedule | Call for participation | Past editions | Submissions

languageworkbenches.net

Language Workbench Challenge

Comparing Tools of the Trade

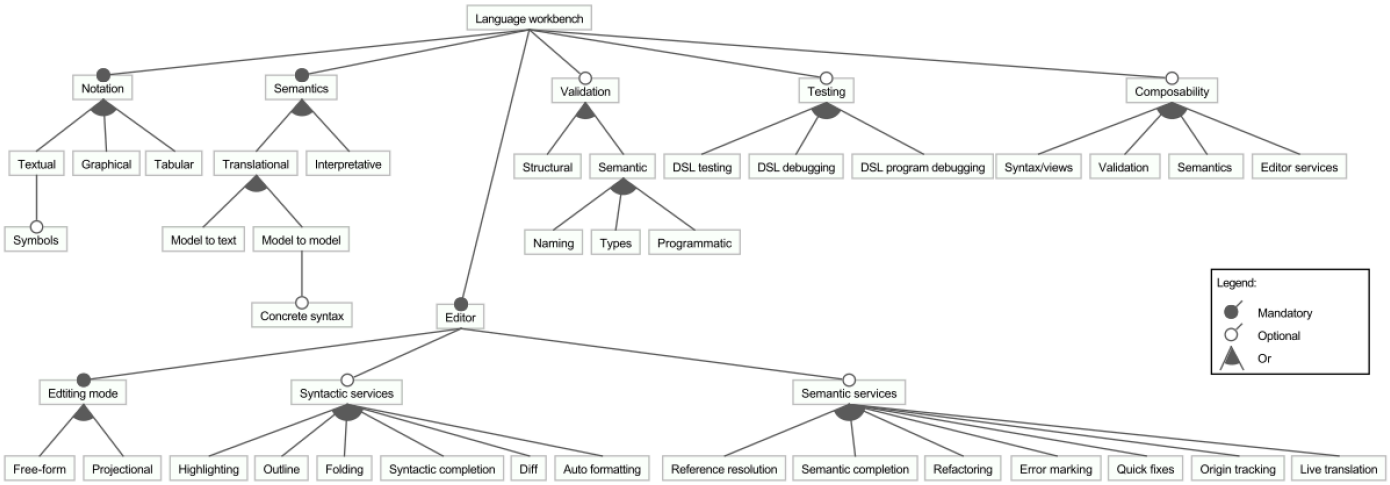
About LWC | Workshop schedule | Call for participation | Past editions | Submissions

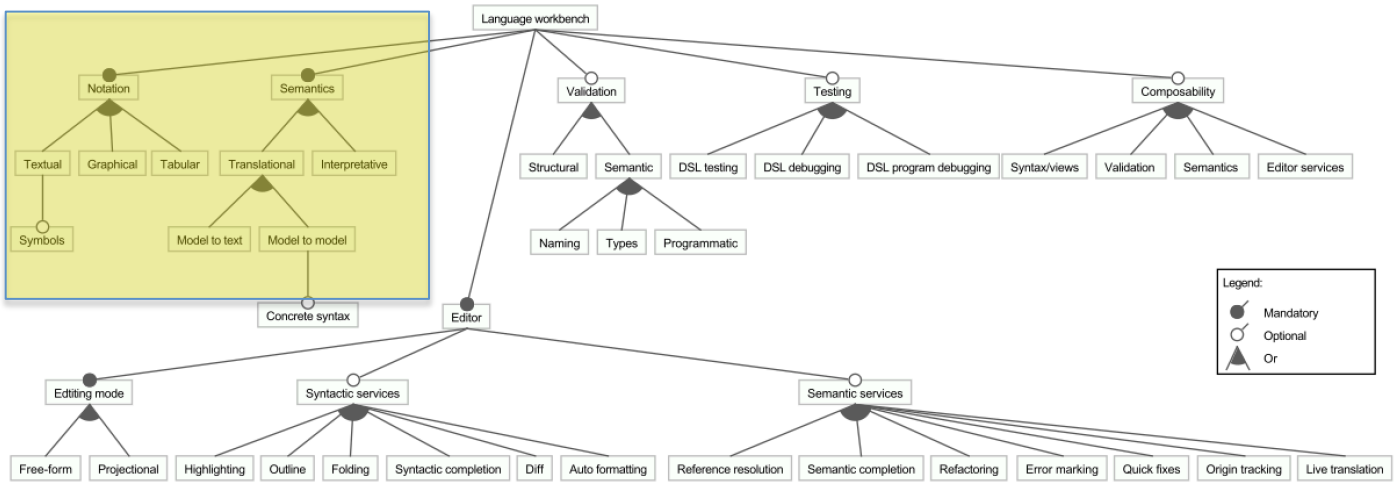
The State of the Art in Language Workbenches

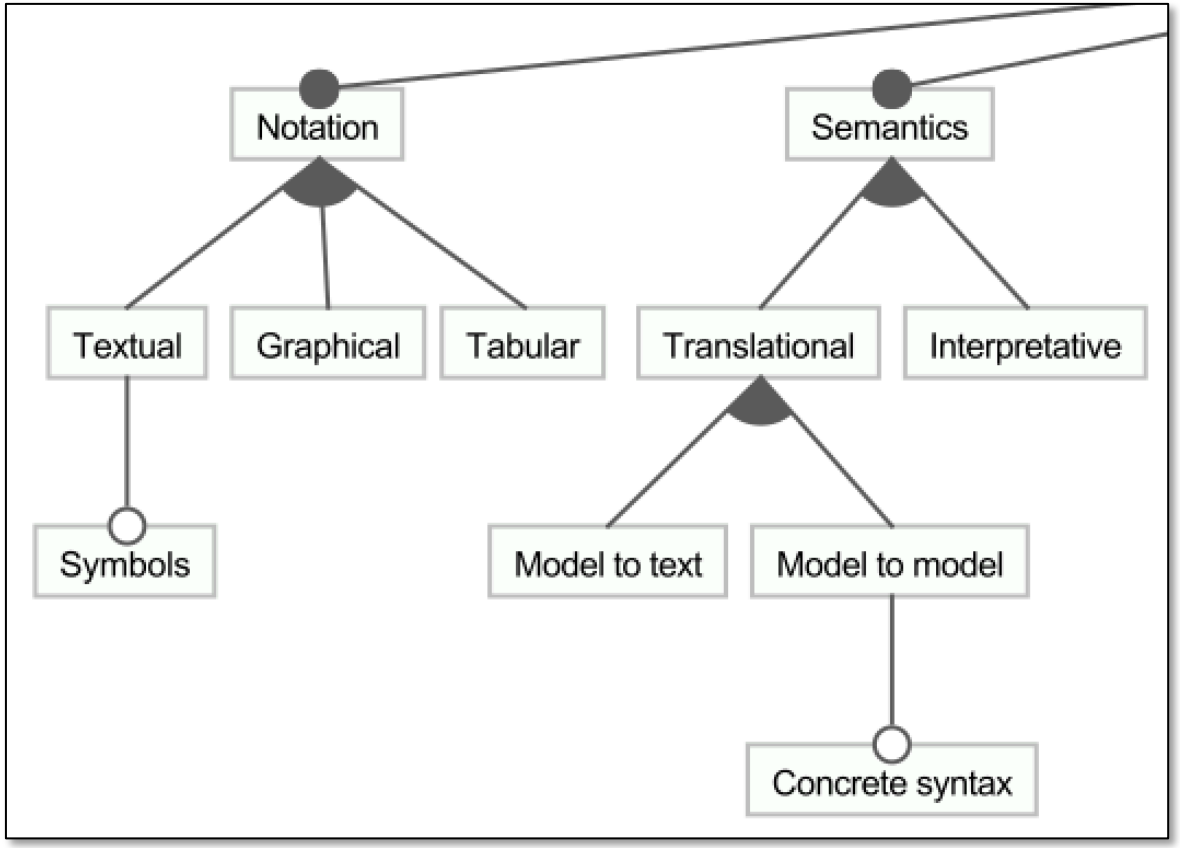
Conclusions from the Language Workbench Challenge

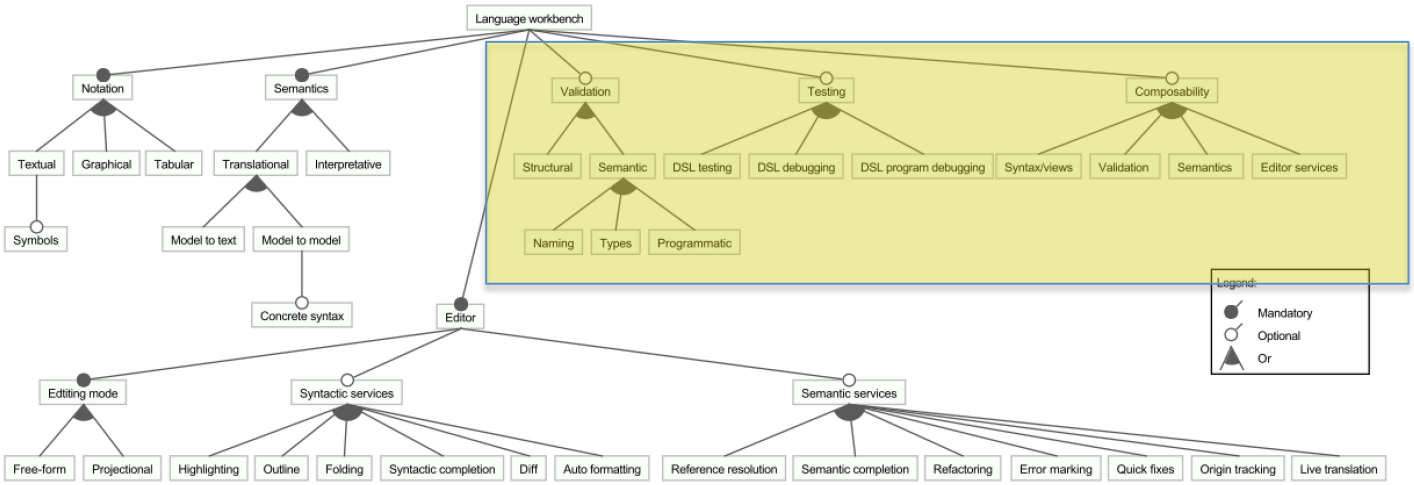
Sebastian Erdweg[1], Tijs van der Storm[2,3], Markus Völter[4], Meinte Boersma[5], Remi Bosman[6], William R. Cook[7], Albert Gerritsen[6], Angelo Hulshout[8], Steven Kelly[9], Alex Loh[7], Gabriël Konat[10], Pedro J. Molina[11], Martin Palatnik[6], Eugen Schindler[6], Klemens Schindler[6], Riccardo Solmi[12], Vlad Vergu[10], Eelco Visser[10], Kevin van der Vlist[13], Guido Wachsmuth[10], and Jimi van der Woning[13]

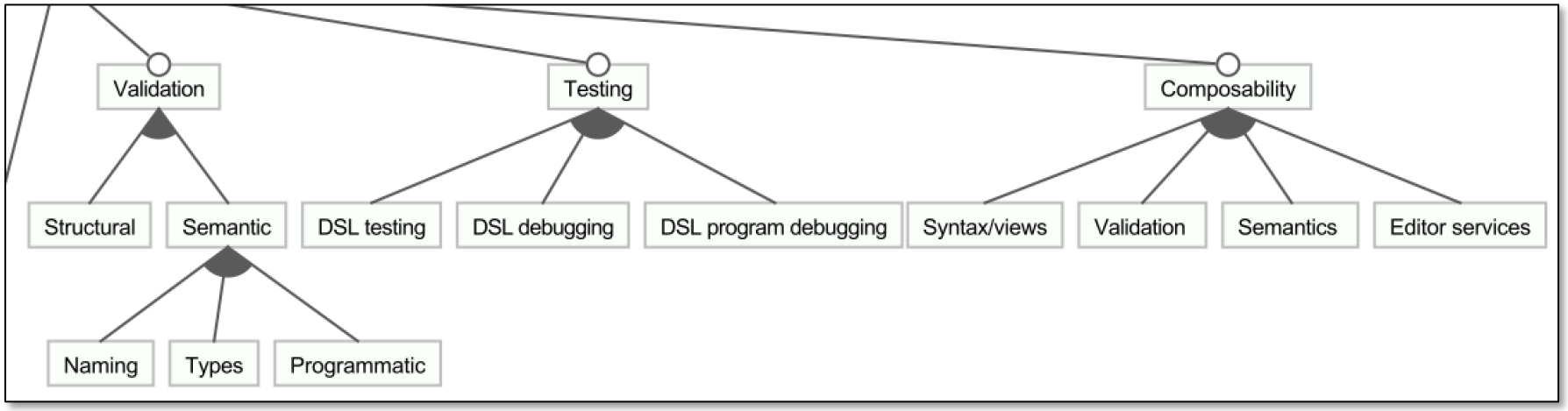
[1] TU Darmstadt, Germany [2] CWI, Amsterdam, The Netherlands [3] INRIA Lille Nord Europe, Lille, France [4] voelter.de, Stuttgart, Germany [5] DSL Consultancy, Leiden, The Netherlands [6] Sioux, Eindhoven, The Netherlands [7] University of Texas, Austin, US [8] Delphino Consultancy, Best, The Netherlands [9] MetaCase, Jyväskylä, Finland [10] TU Delft, The Netherlands [11] Icinetic, Sevilla, Spain [12] Independent, Bologna, Italy [13] Universiteit van Amsterdam

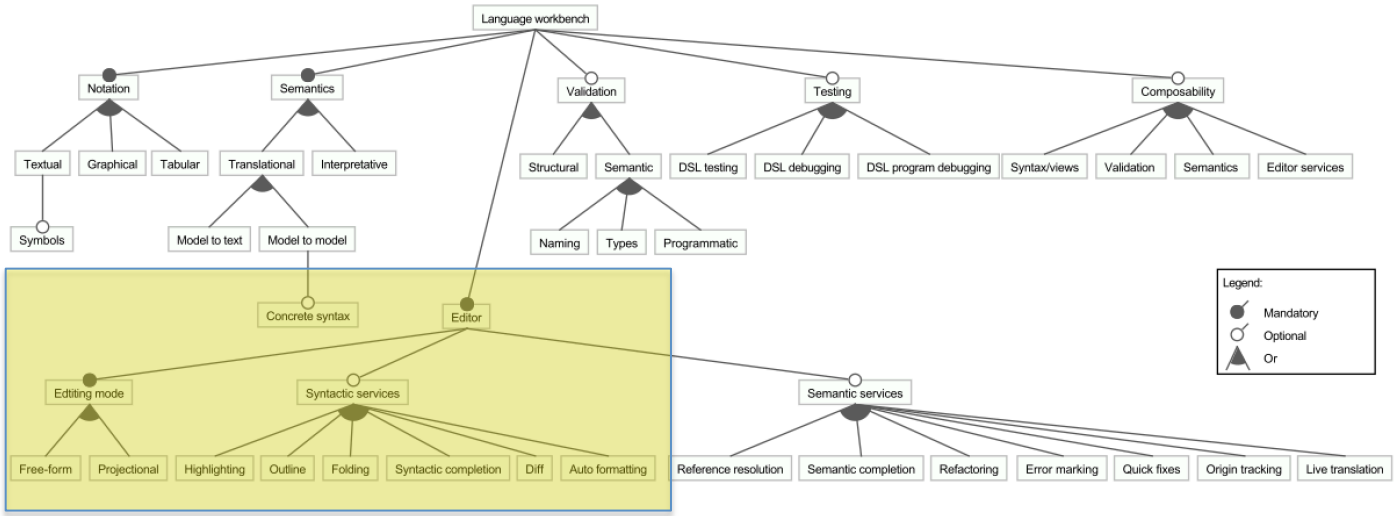

Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
Concrete syntax
Editor
Legend:
Mandatory
Optional
Or
Edtiting mode
Syntactic services
Semantic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation

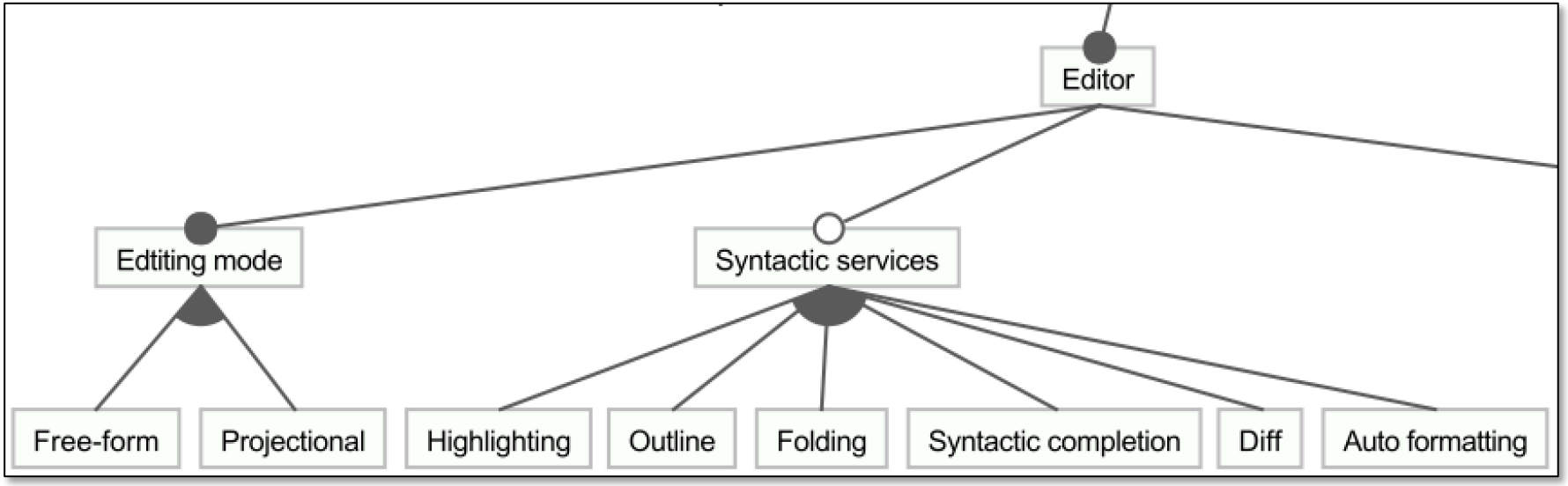

Language workbench
Notation
Semantics
Textual
Graphical
Tabular
Translational
Interpretative
Symbols
Model to text
Model to model
Concrete syntax
Validation
Testing
Composability
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Naming
Types
Programmatic
Editor
Legend:
Mandatory
Optional
Or
Edtiting mode
Syntactic services
Semantic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation

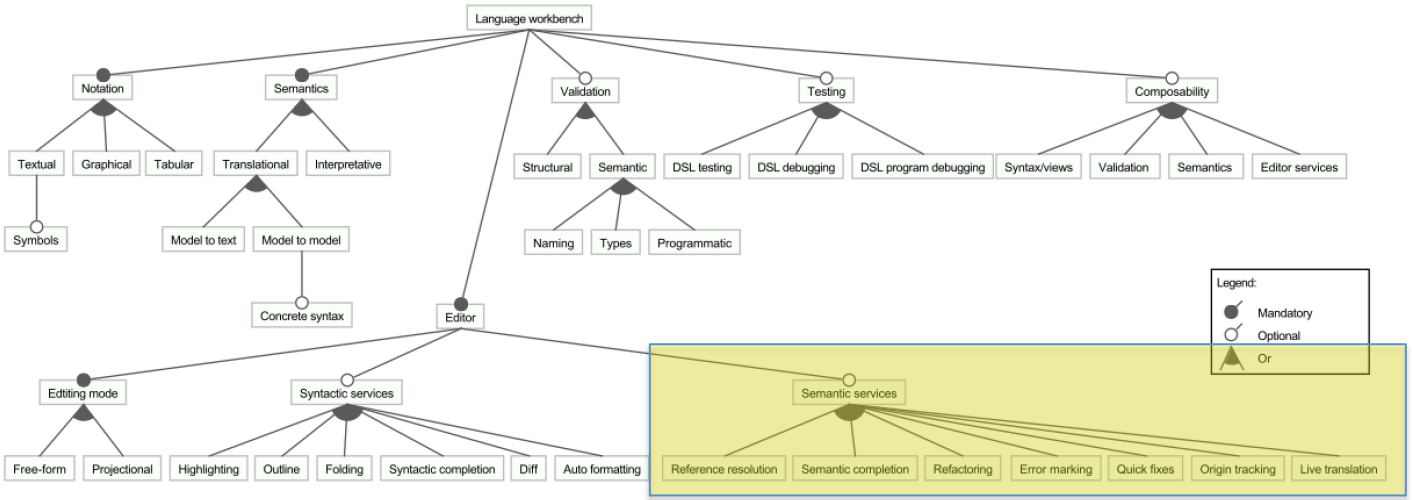

Notation
Semantics
Textual
Graphical
Tabular
Translational
Interpretative
Symbols
Model to text
Model to model
Concrete syntax

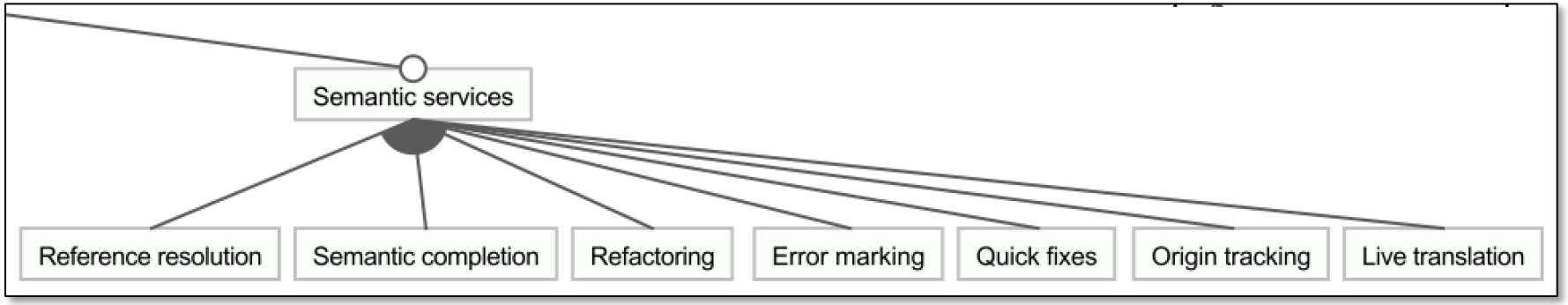

Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
Concrete syntax
Editor
Legend:
Mandatory
Optional
Or
Editing mode
Syntactic services
Semantic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation

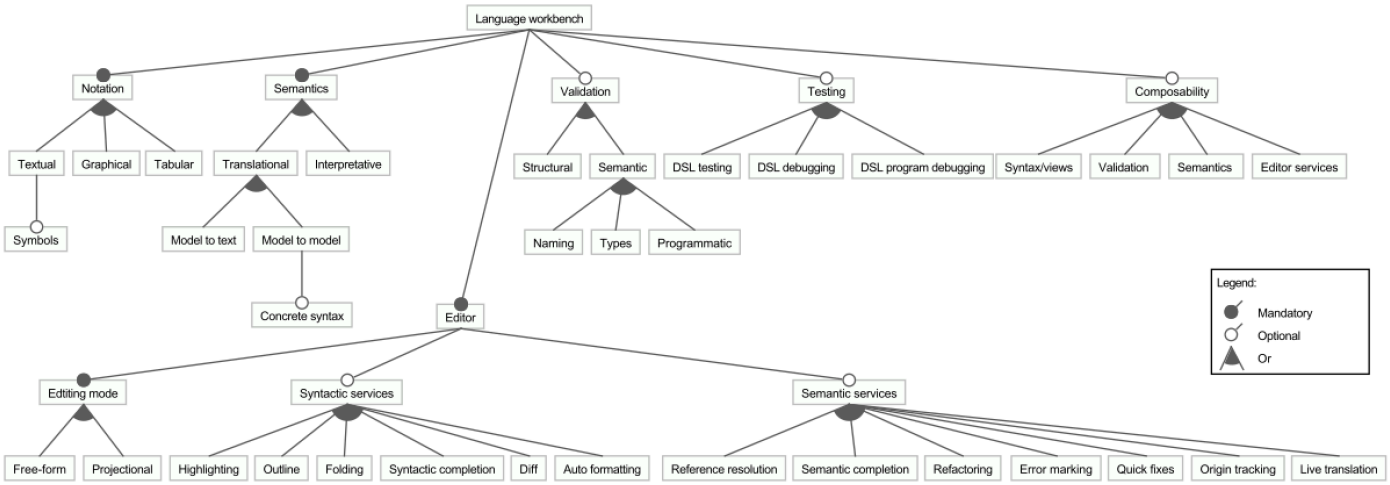

Validation
Testing
Composability
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Naming
Types
Programmatic


Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
Concrete syntax
Editor
Legend:
Mandatory
Optional
Or
Edtiting mode
Syntactic services
Semantic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation

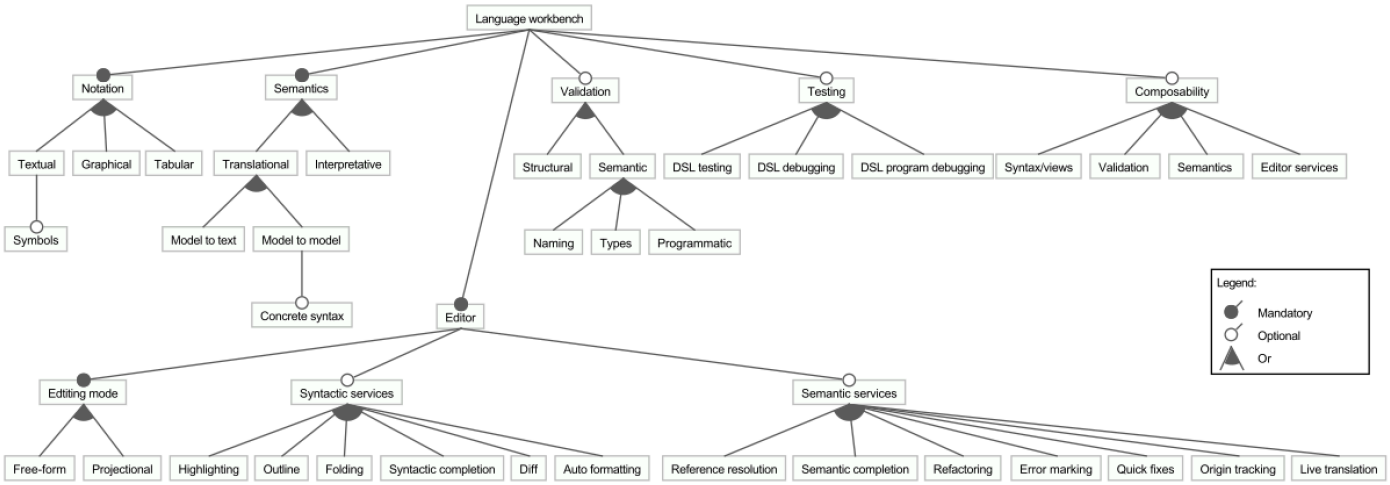

Editor
Edtiting mode
Syntactic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting


Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
Concrete syntax
Editor
Legend:
Mandatory
Optional
Or
Editing mode
Syntactic services
Semantic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation

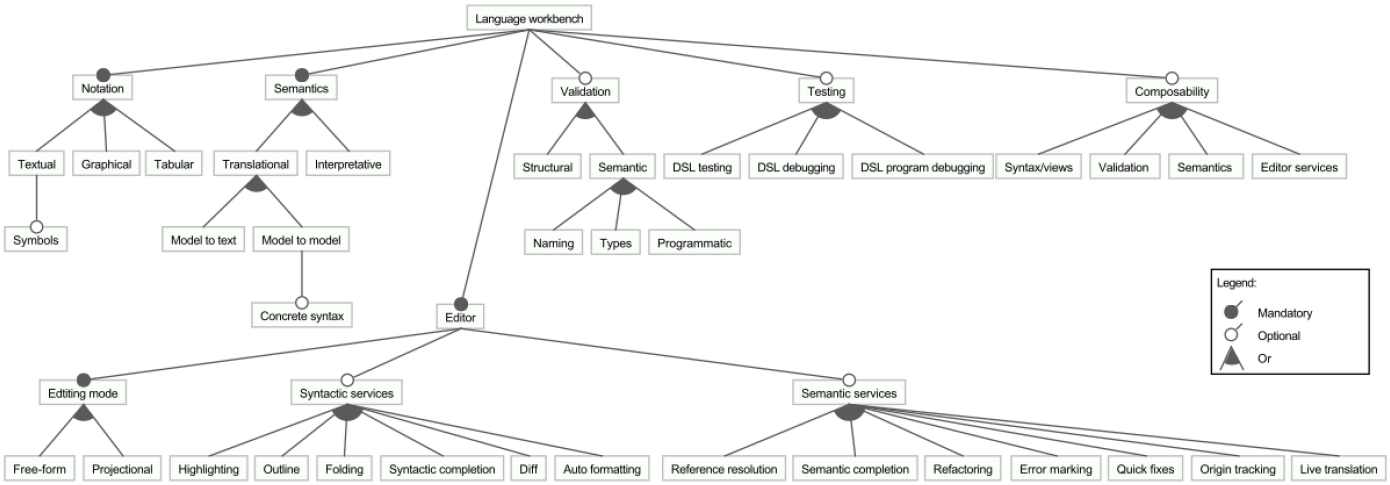

Semantic services
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation



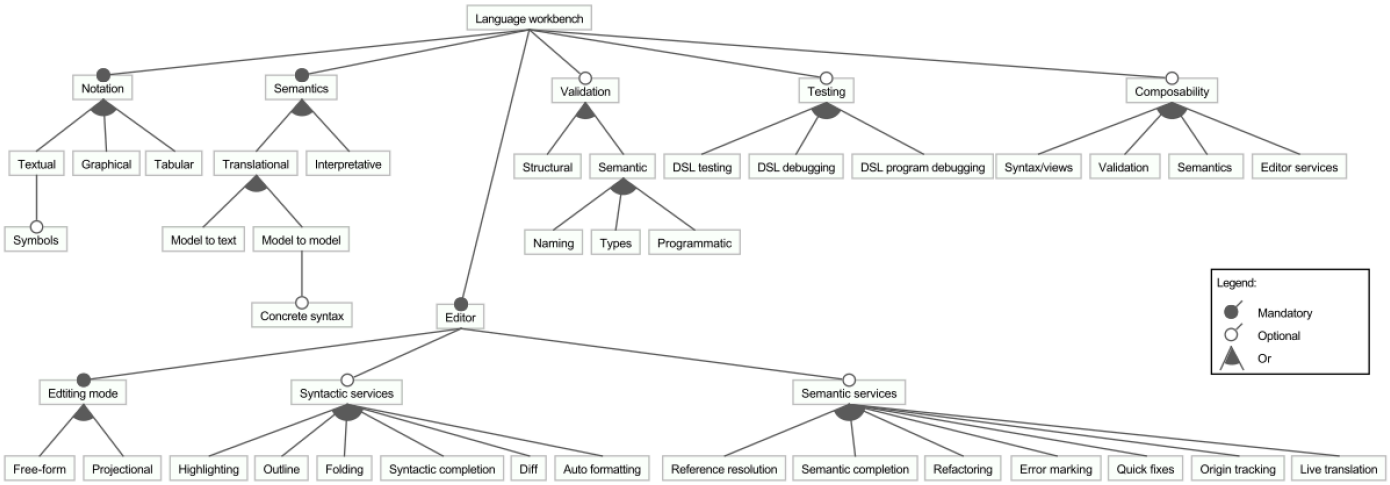
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●



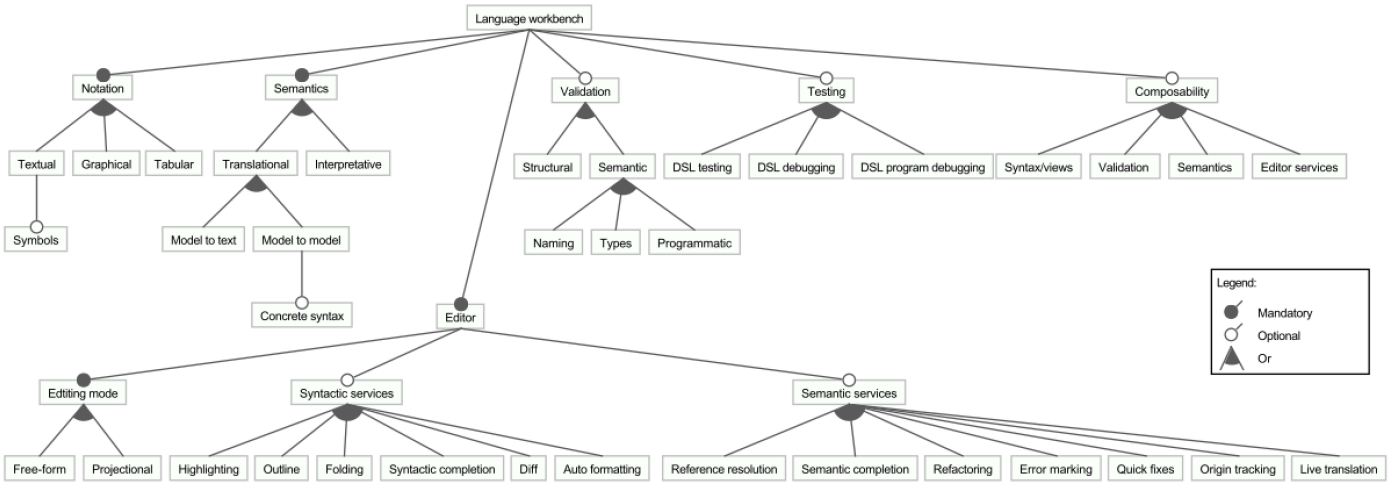
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●



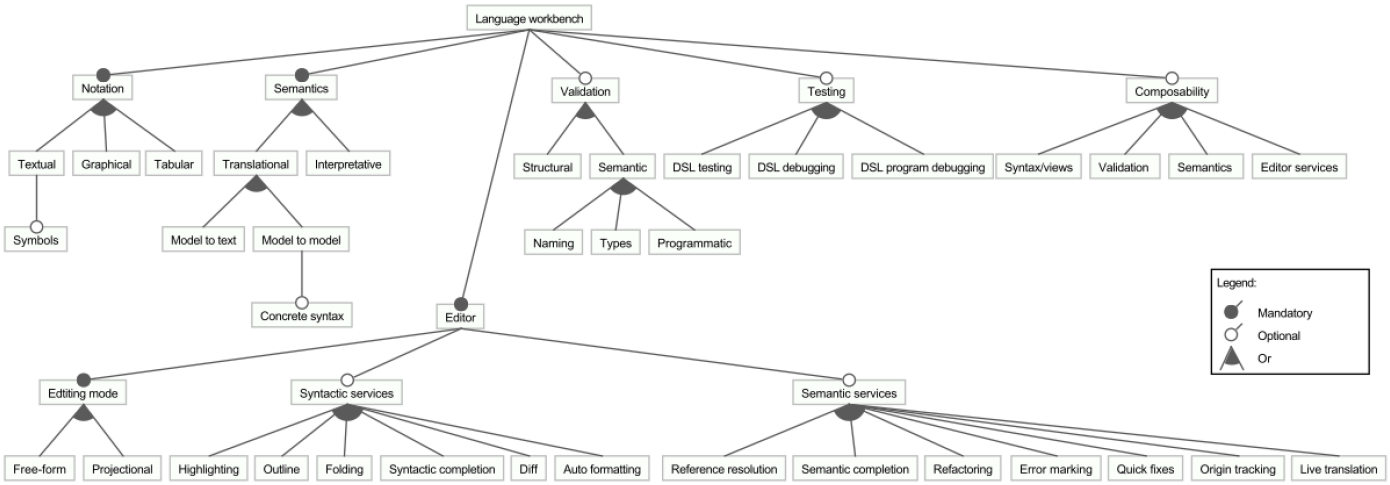
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●



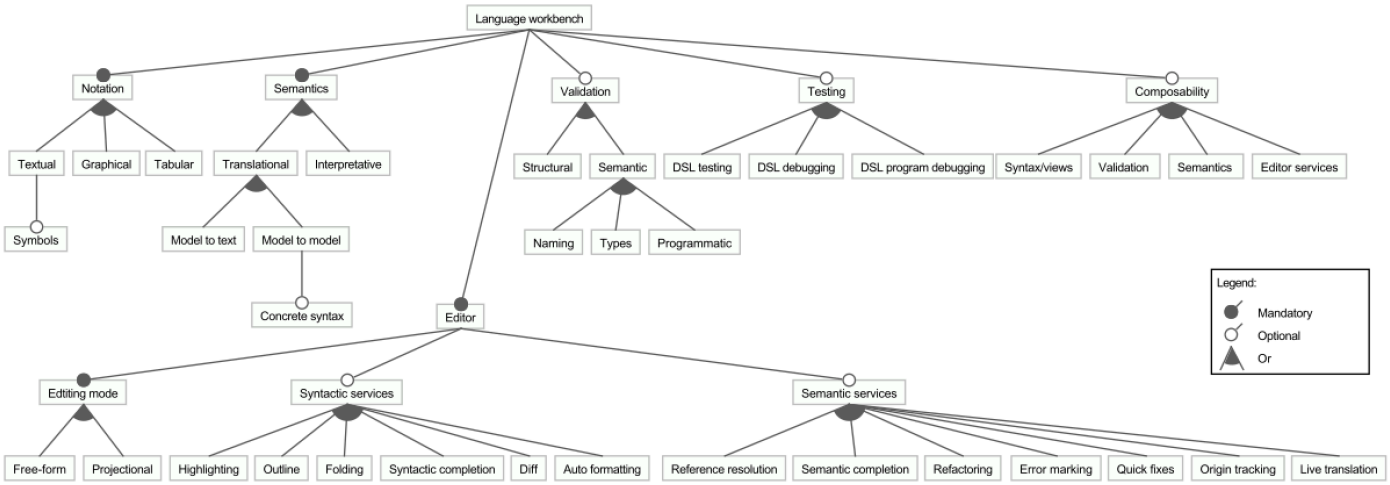
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●



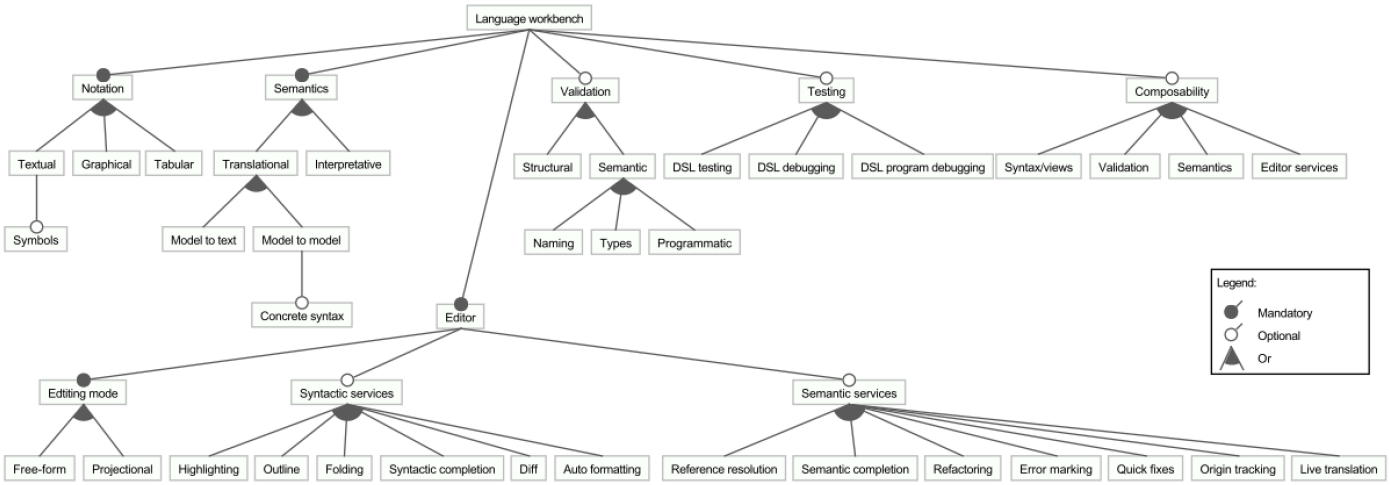
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●



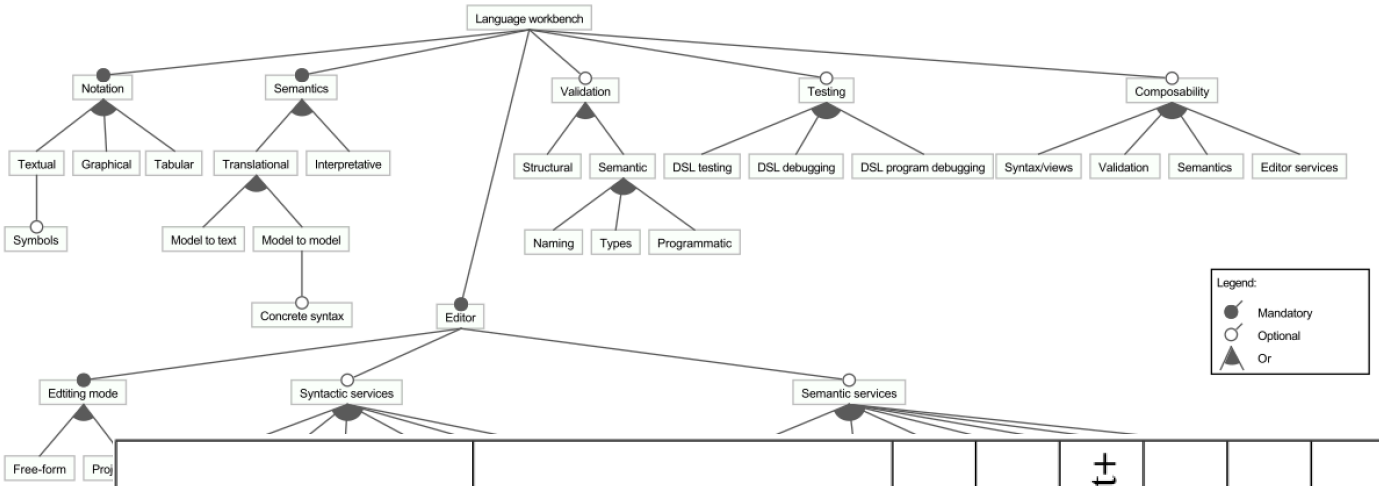
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◒	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◒
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
Editing mode	Free-form	●		●		●	●	●	●		●
	Projectional		●		●	●				●	



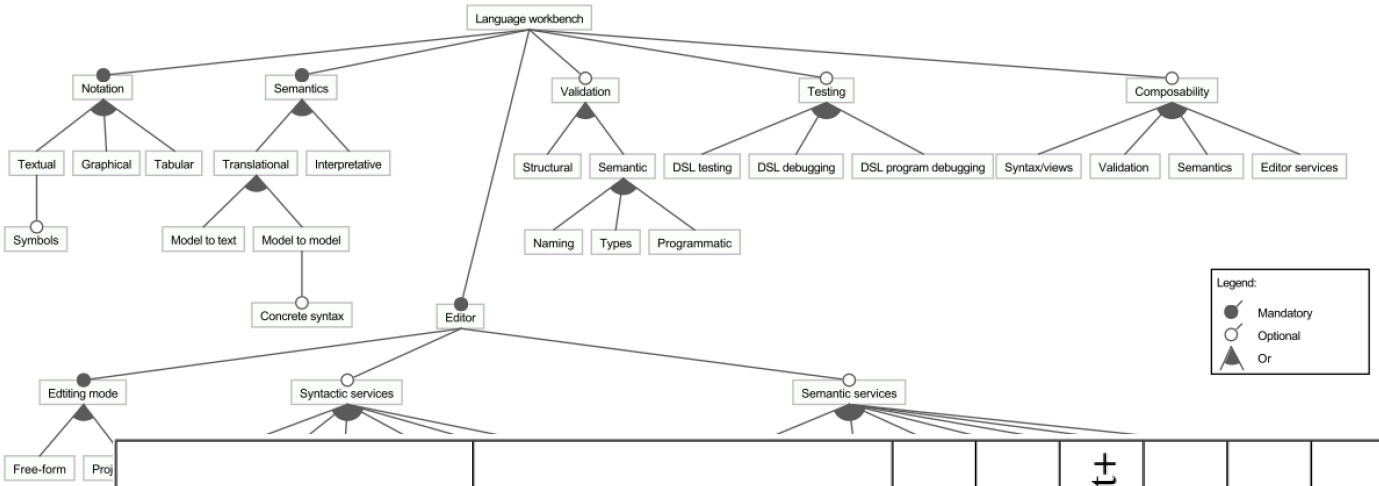
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◐	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◐
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
Editing mode	Free-form	●		●		●	●	●	●		●
	Projectional		●		●	●				●	



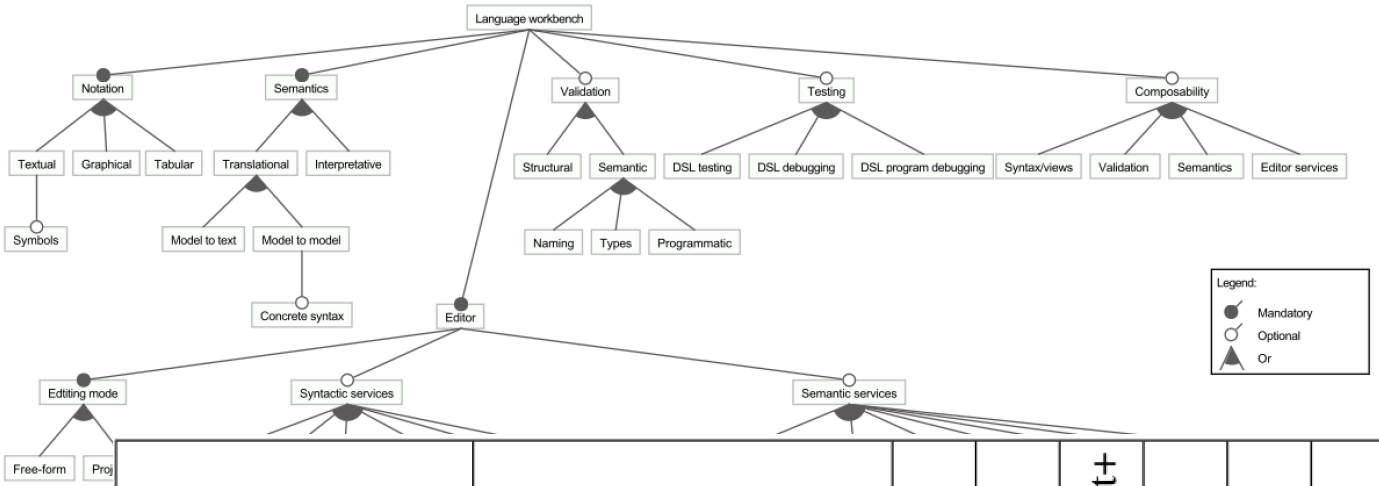
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◐	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◐
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
Editing mode	Free-form	●		●		●	●	●	●		●
	Projectional		●		●	●				●	



		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Syntactic services	Highlighting		◒	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding		●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	Semantic completion		●	●	●	●	●	●	●	●	●
	Refactoring		◒	●	●		●	●		●	
	Error marking		●	●	●	●	●	●	●	●	●
	Quick fixes				●						●
	Origin tracking	●		●	●		●	●	●		●
	Live translation			●		●	◒	●		●	●



		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Syntactic services	Highlighting		◒	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding		●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	Semantic completion		●	●	●	●	●	●	●	●	●
	Refactoring		◒	●	●		●	●		●	
	Error marking		●	●	●	●	●	●	●	●	●
	Quick fixes				●						●
	Origin tracking	●		●	●		●	●	●		●
	Live translation			●		●	◒	●		●	●



		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Syntactic services	Highlighting		◒	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding		●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	Semantic completion		●	●	●	●	●	●	●	●	●
	Refactoring		◒	●	●		●	●		●	
	Error marking		●	●	●	●	●	●	●	●	●
	Quick fixes				●						●
	Origin tracking	●		●	●		●	●	●		●
	Live translation			●		●	◒	●		●	●

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

academic

vs.

industry

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

used for real

vs.

research/prototype

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

commercial

vs.

open source

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

more
(not part of LWC)

+

JetBrains Nitra
oomega
EMFText
Eclipse Sirius
Eco

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

worth looking at

+

JetBrains Nitra

oomega

EMFText

Eclipse Sirius

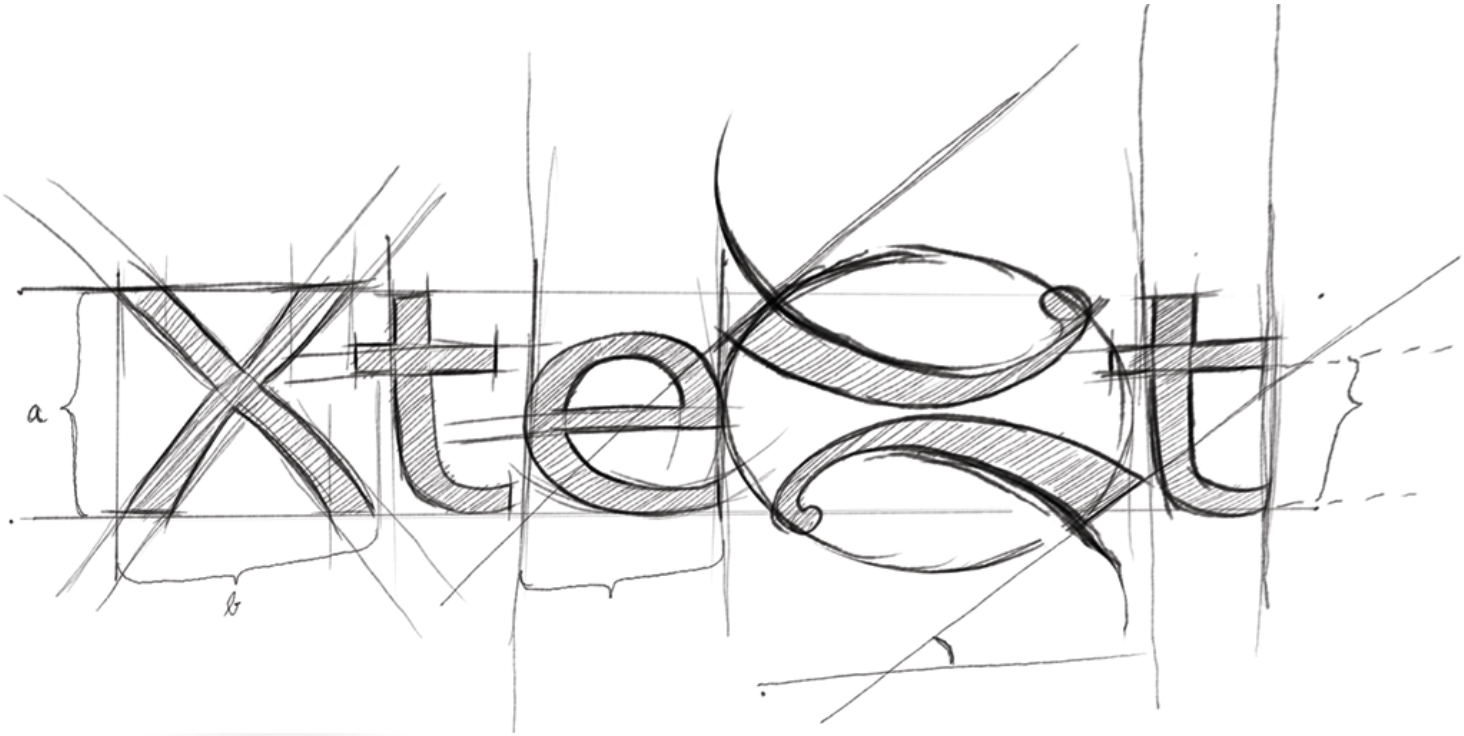
Eco

Tabular	Graphical	Textual	
	●	●	Ensō
●	◐	●	Más
●	●		MetaEdit+
●		●	MPS
		●	Onion
	◐	●	Rascal
		●	Spoofax
		●	SugarJ
●	●	●	Whole
		●	Xtext

this talk

+

JetBrains Nitra
oomega
EMFText
Eclipse Sirius
Eco



eclipse

itemis

JetBRAINS

Spoofax

TU Delft

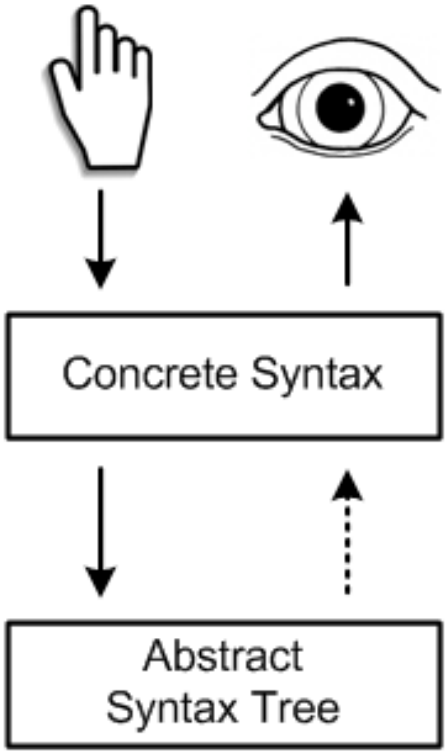
2

Notational Flexibility

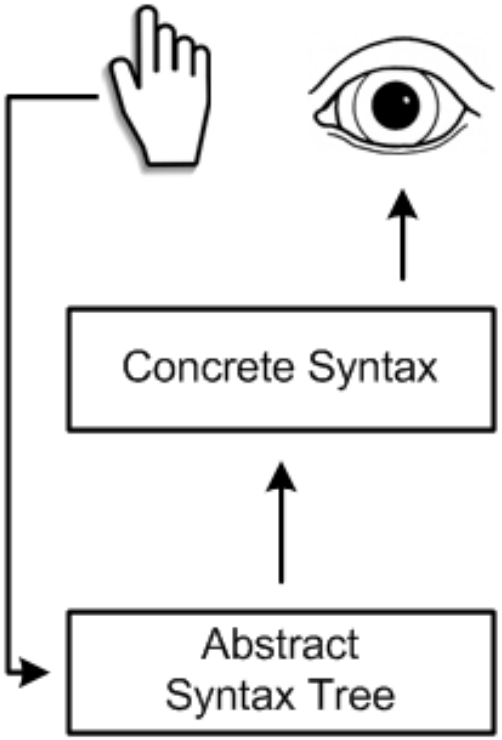
Projectional Editing

[Projectional Editing]

Parsing



Projectional Editing



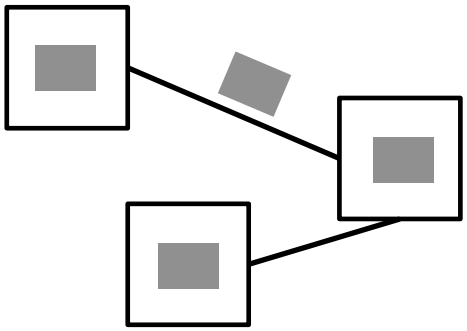
[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical



[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

```
//[ A documentation comment with references
  [ to @arg(data) and @arg(dataLen)
void aSummingFunction(int8[] data, int8 dataLen) {
  int16 sum;
  for (int8 i = 0; i < dataLen; i++) {
    sum += data[i];
  } for
} aSummingFunction (function)
```

Mathematical

```
double midnight2(int32 a, int32 b, int32 c) {
```

$$\text{return}\ \frac{-b + \sqrt{b^2 - \sum_{i=1}^{4} a * c}}{2 * a};$$

```
} midnight2 (function)
```

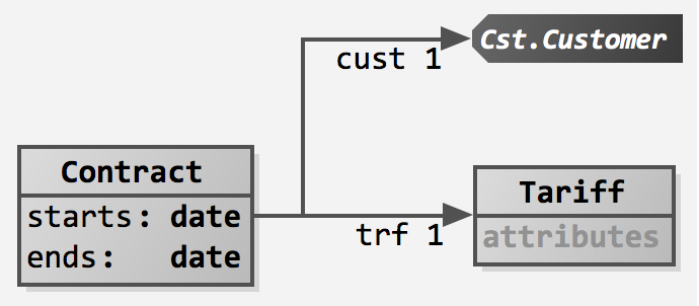
Tables

```
int16 decide(int8 spd, int8 alt) {
  return
```

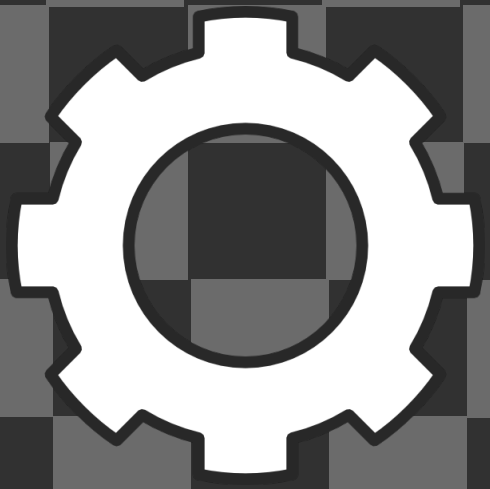
	spd > 0	spd > 100
alt < 0	1	1
alt == 0	10	20
alt > 0	30	40
alt > 100	50	60

```
otherwise 0;
} decide (function)
```

Graphical



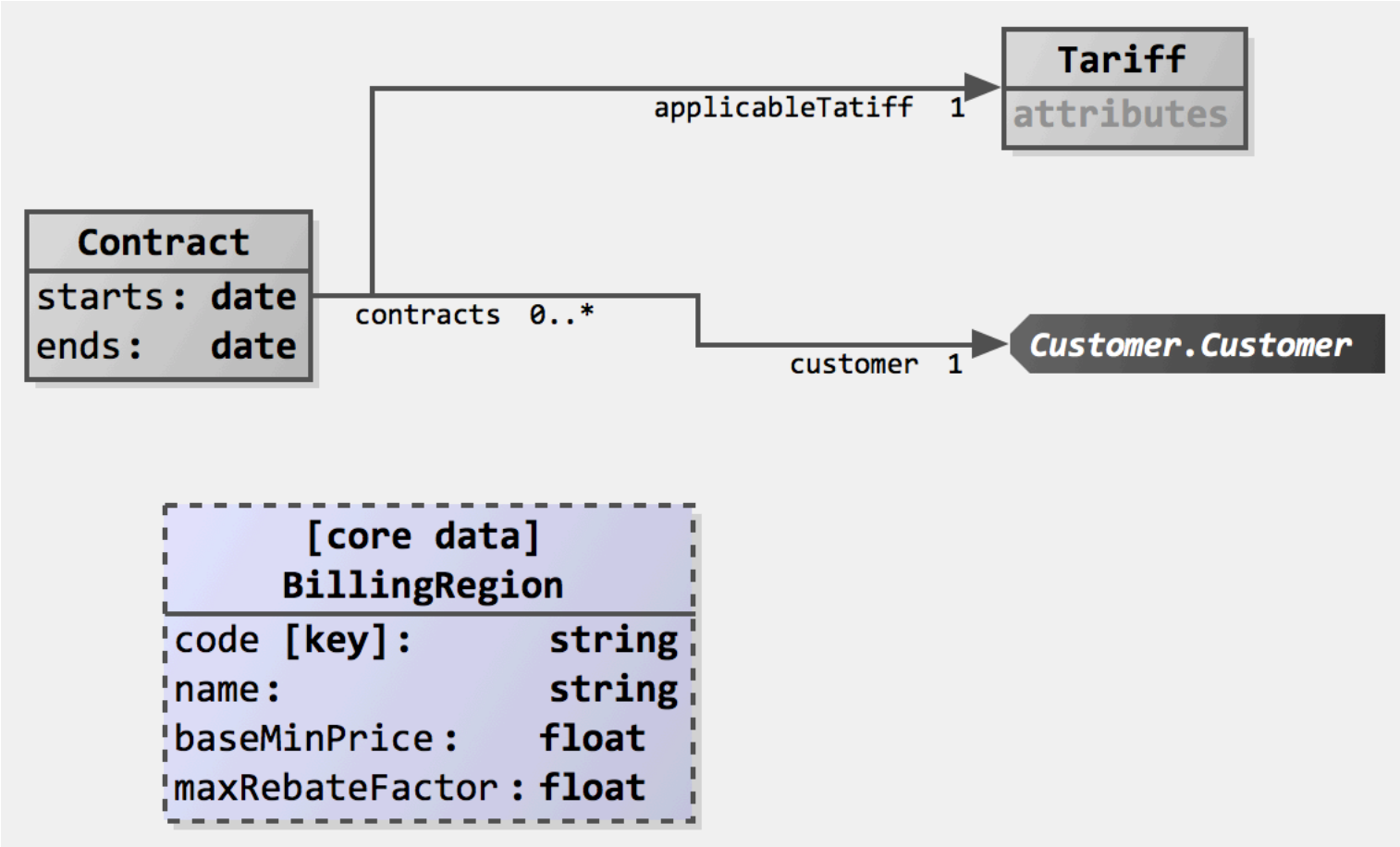
DEMO



Notations

[Business Application]

Insurance Configuration Tool



[Business Application]

Insurance Configuration Tool

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

[Business Application]

Insurance Configuration Tool

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>

[Tax/Benefits Application]

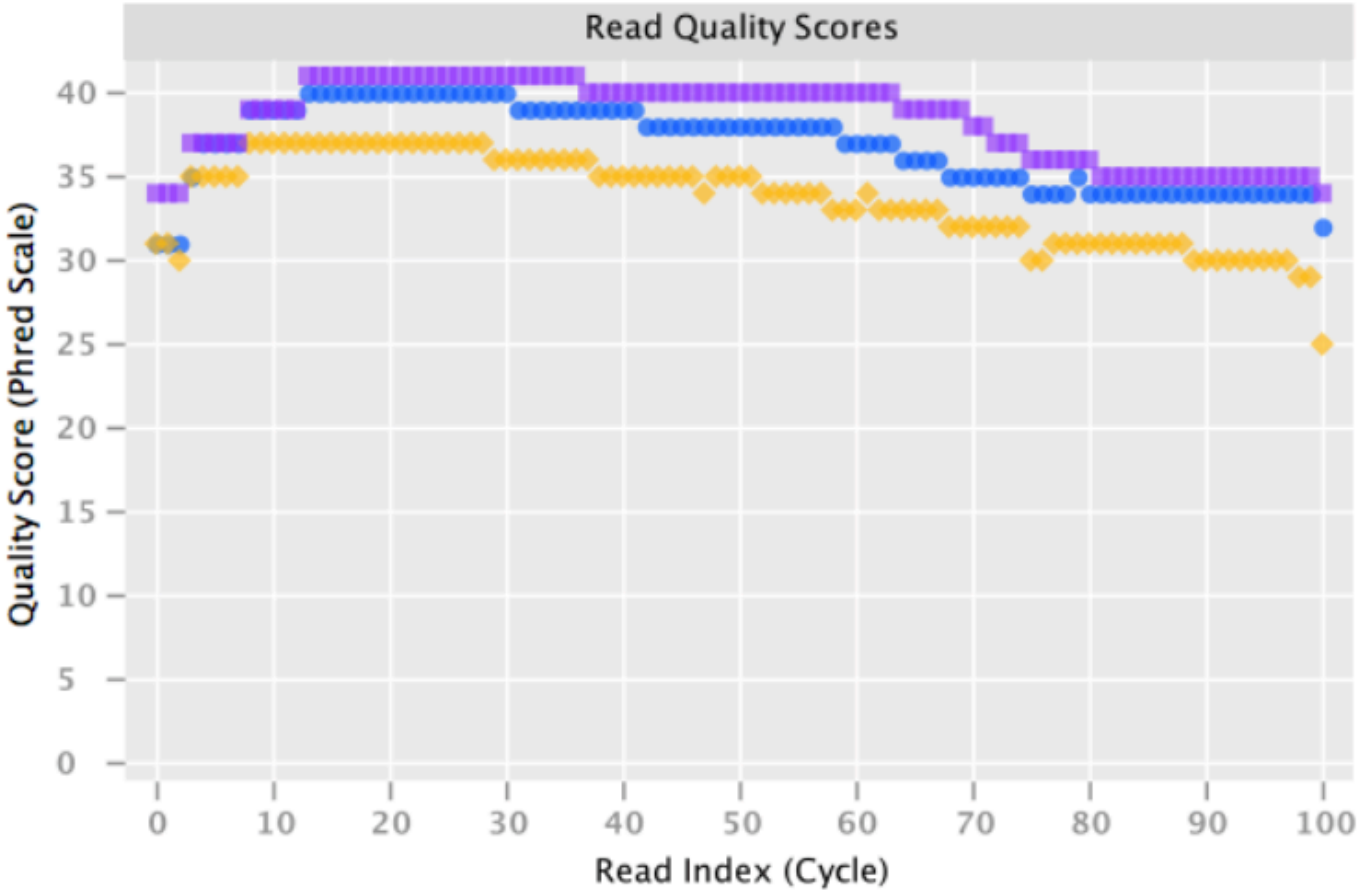
Dutch Tax Agency (Evaluation)

```
bloedverwanten : lijst van Burgers  zijn gedefinieerd als {
    Een bloedverwant is een Burger die
    bloedverwant in rechte lijn is of die
    bloedverwant in tweede graad zijlijn is
    Einde declaratie
}

bloedverwanten in rechte lijn : lijst van Burgers  zijn gedefinieerd als {
    Een bloedverwant in rechte lijn is een Burger die
    nakomeling is of die
    voorouder is
    Einde declaratie
}

bloedverwanten in tweede graad zijlijn : lijst van Burgers  zijn gedefinieerd als {
    Een bloedverwant in tweede graad zijlijn is een ouder.kind met
    ouder.kind ongelijk het actuele voorkomen
    Einde declaratie
    ' dus: broer of zus (incl. erkend kind van ouder)
}

bloed- of aanverwanten in rechte lijn : lijst van Burgers  zijn gedefinieerd als {
    Een bloed- of aanverwant in rechte lijn is een Burger die
    bloedverwant in rechte lijn is of die
    aanverwant in rechte lijn is
    Einde declaratie
}
```

[Biological Data]

CampagneLab's NYoSh

```
goby web read sample 849-WT {
  tag: NGBOXSJ
  number of reads: 29230382
  is paired: false
  is bisulfite: false
  is ready to align: true
  min length: 101
  max length: 101
  organism: mus_musculus
  platform: Illumina
  lib protocol preserve strand: true
}
```



[Embedded Software]

mbeddr

```
int32 sumUpIntArray(int32[] arr, int32 size) {
```

$$\text{return} \sum_{i=0}^{size} arr[i];$$

```
} sumUpIntArray (function)
```

```
int32 averageIntArray(int32[] arr, int32 size) {
```

$$\text{return} \frac{\sum_{i=0}^{size} arr[i]}{size};$$

```
} averageIntArray (function)
```

```
double midnight1(int32 a, int32 b, int32 c) {
```

$$\text{return} \frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a};$$

```
} midnight1 (function)
```

```
double midnight2(int32 a, int32 b, int32 c) {
```

$$\text{return} \frac{-b + \sqrt{b^2 - \sum_{i=1}^{4} a * c}}{2 * a};$$

```
} midnight2 (function)
```

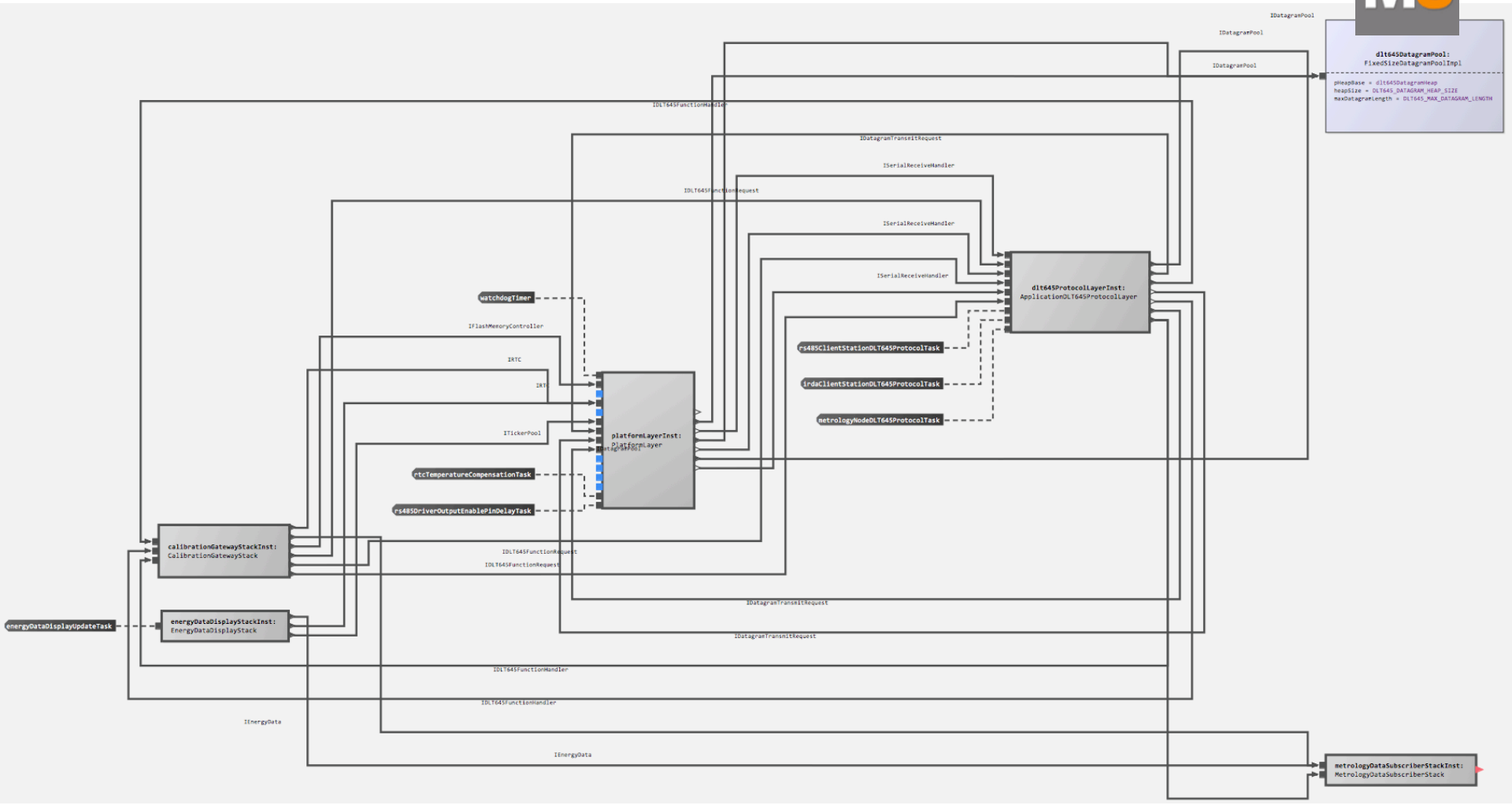
```
double sumOfProductsOfLogs(int32[] arr, int32 size) {
```

$$\text{return} \sum_{k=0}^{size} \frac{\prod_{i=0}^{k} \log_2 arr[i]}{2};$$

```
} sumOfProductsOfLogs (function)
```

[Embedded Software]

mbeddr



[Embedded Software]

mbeddr

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

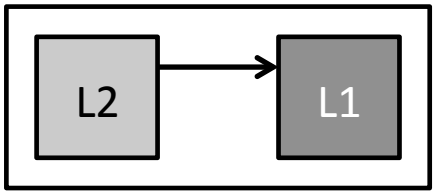
		Events	
		next(Trackpoint* tp)	reset()
States	beforeFlight	[tp->alt > 0 m] -> airborne	
	airborne	[tp->alt == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne { points += VERY_HIGH_SPEED; } [tp->speed > 100 mps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

```
}
```

3

Language Modularity

[Projectional Editing]
Language Composition



Separate Files

Type System
Transformation
Constraints

In One File

Type System
Transformation
Constraints
Syntax
IDE

mbeddr

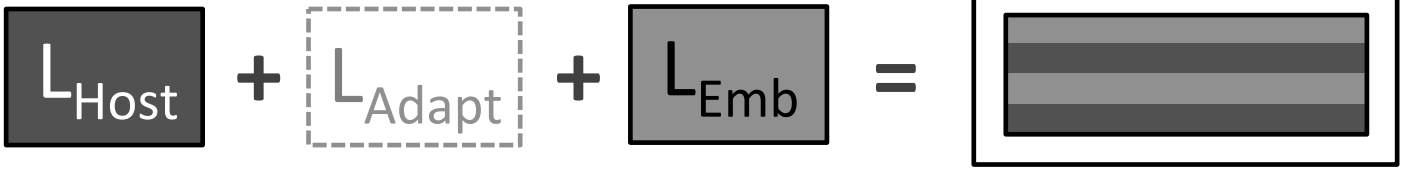
50+ extensions to C
10+ extensions to requirements lang.

[Projectional Editing]

Modular Language Composition

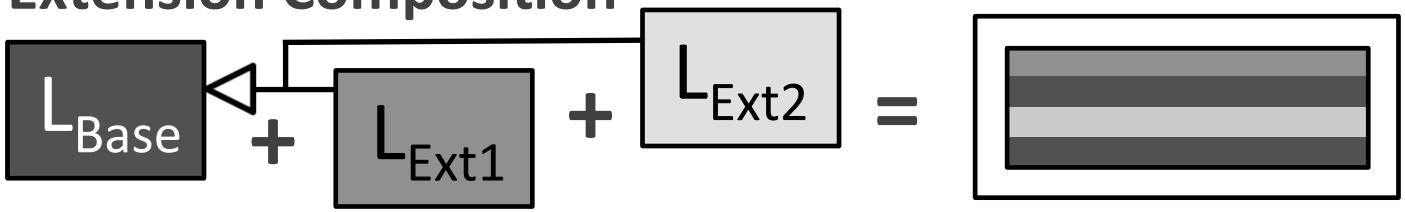
No change to definition of L1 **or** L2 **in order to use them together.**

Embedding



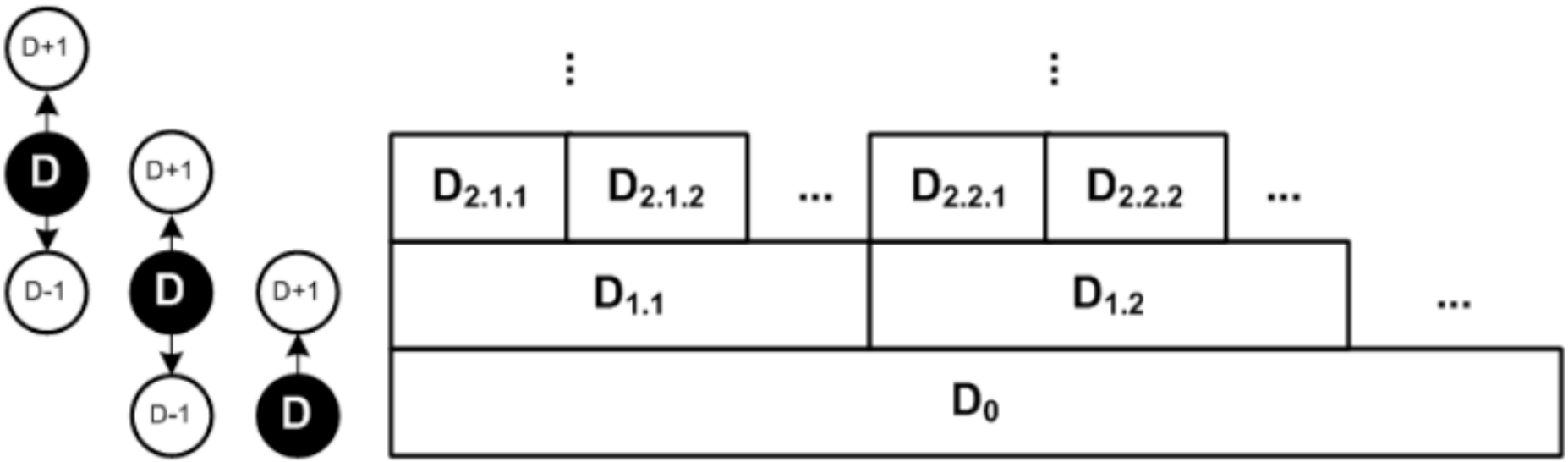
Extension

Extension Composition



[Projectional Editing]

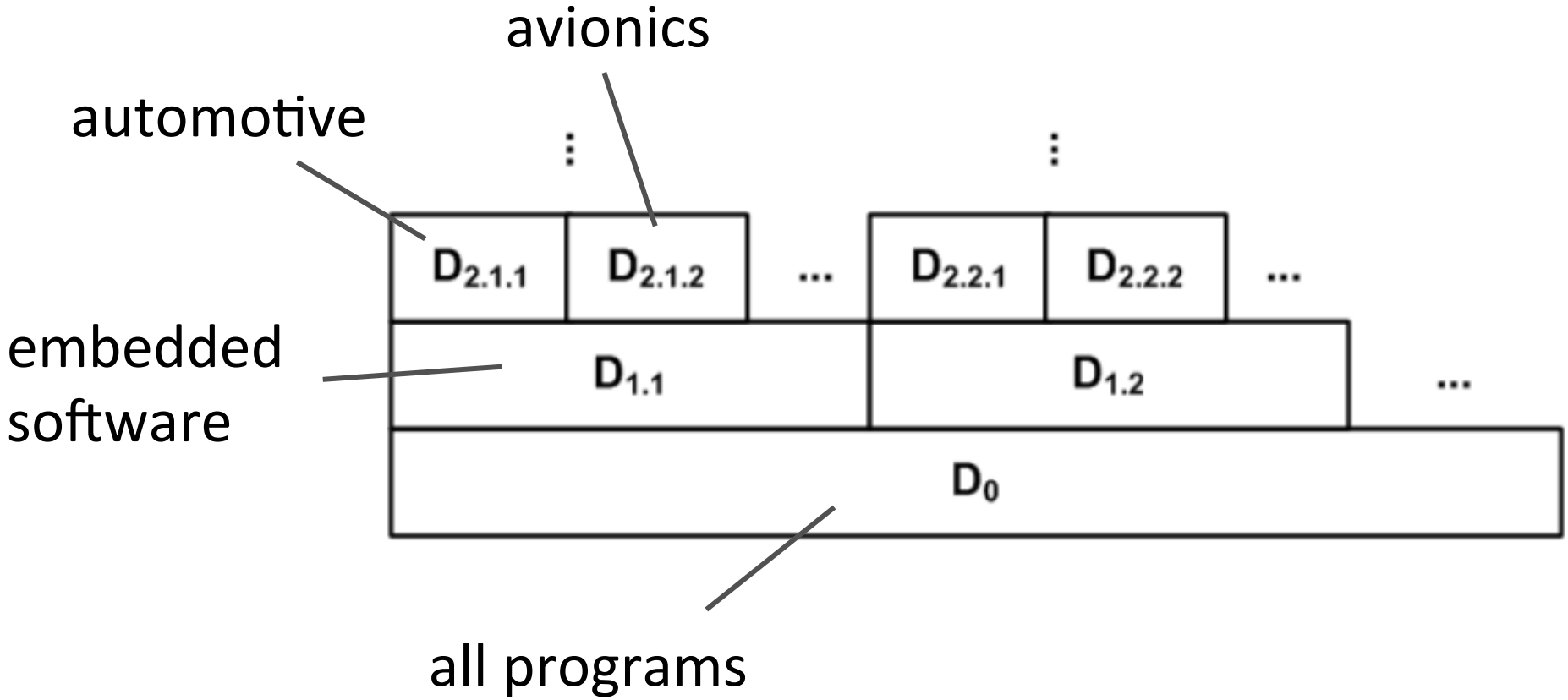
Growing a Language towards Domain



Domains Naturally form a hierarchy.
Languages can „grow“ along.

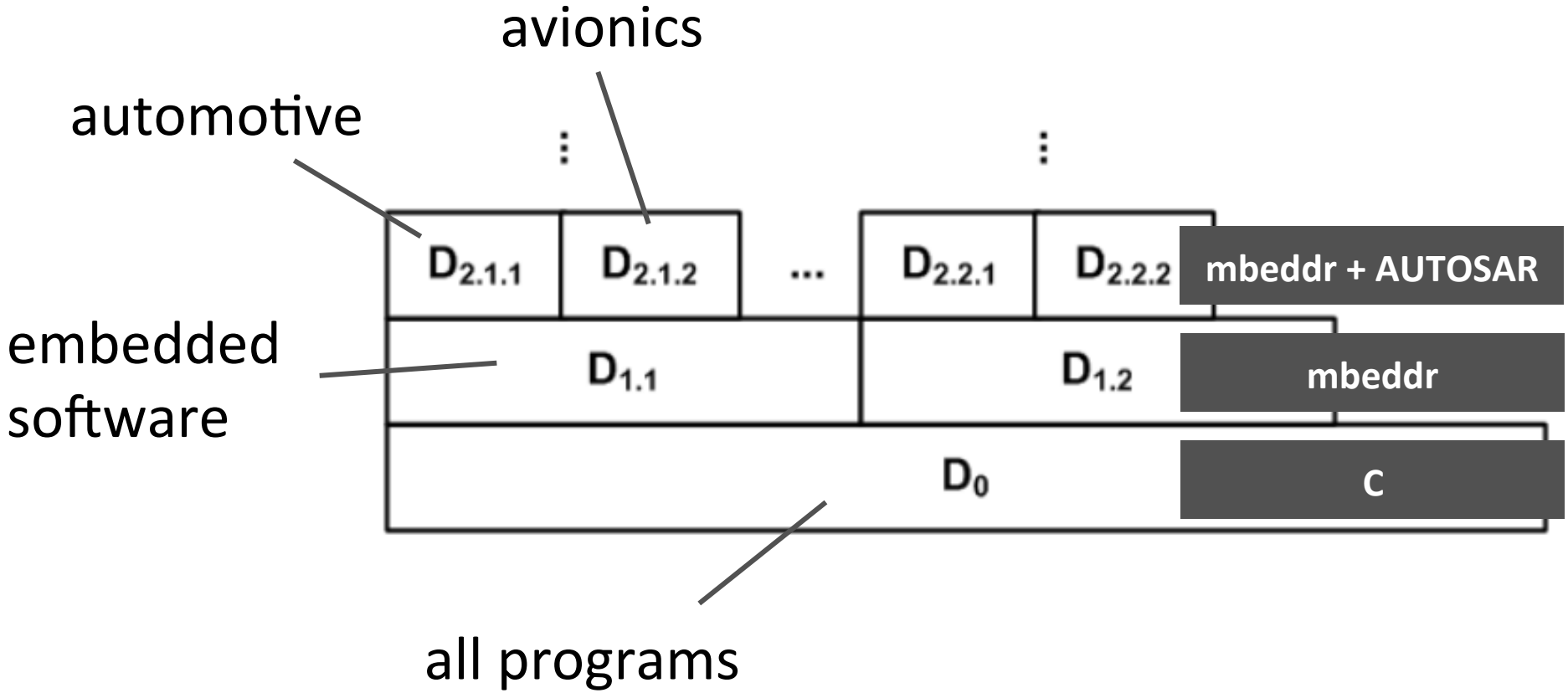
[Projectional Editing]

Growing a Language towards Domain



[Projectional Editing]

Growing a Language towards Domain



[Projectional Editing]

DSL Styles and Trade-Offs

Classic External DSL

Nicely analyzable (if done right)

Great IDE Support

Limited Expressiveness

[Projectional Editing]

DSL Styles and Trade-Offs

Classic External DSL	External + Expressions

Can still be nicely analyzable

Great IDE Support

Better expressiveness b/c Expressions

[Projectional Editing]

DSL Styles and Trade-Offs

Classic External DSL	External + Expressions	Prog Lang Extension

Harder to analyze (GPL influence)

Great IDE Support

Very expressive (Fallback to GPL)

[Projectional Editing]

DSL Styles and Trade-Offs

Classic External DSL	External + Expressions	Prog Lang Extension

Great IDE Support

[Projectional Editing]

DSL Styles and Trade-Offs

Prog Lang Extension

Built with LWB

Great IDE Support

[Projectional Editing]

DSL Styles and Trade-Offs

Prog Lang Extension

Built with LWB

~~Great IDE Support~~

Built via Meta Programming

Ruby
Scala
Xtend

Internal DSL

[Projectional Editing]

DSL Styles and Trade-Offs

Prog Lang Extension

Built with LWB

~~Great IDE Support~~

Built via Meta Programming

~~Type System~~
~~Compiler~~
~~IDE~~

Ruby
Scala
Xtend

Internal DSL

[Projectional Editing]

DSL Styles and Trade-Offs

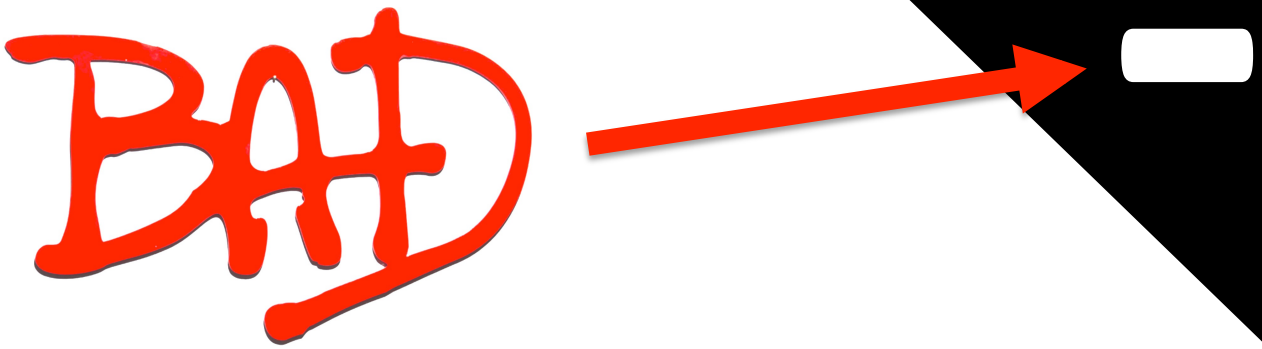
Prog Lang Extension

Potentially similar syntax, but quite different UX

Internal DSL

4

Specification Languages



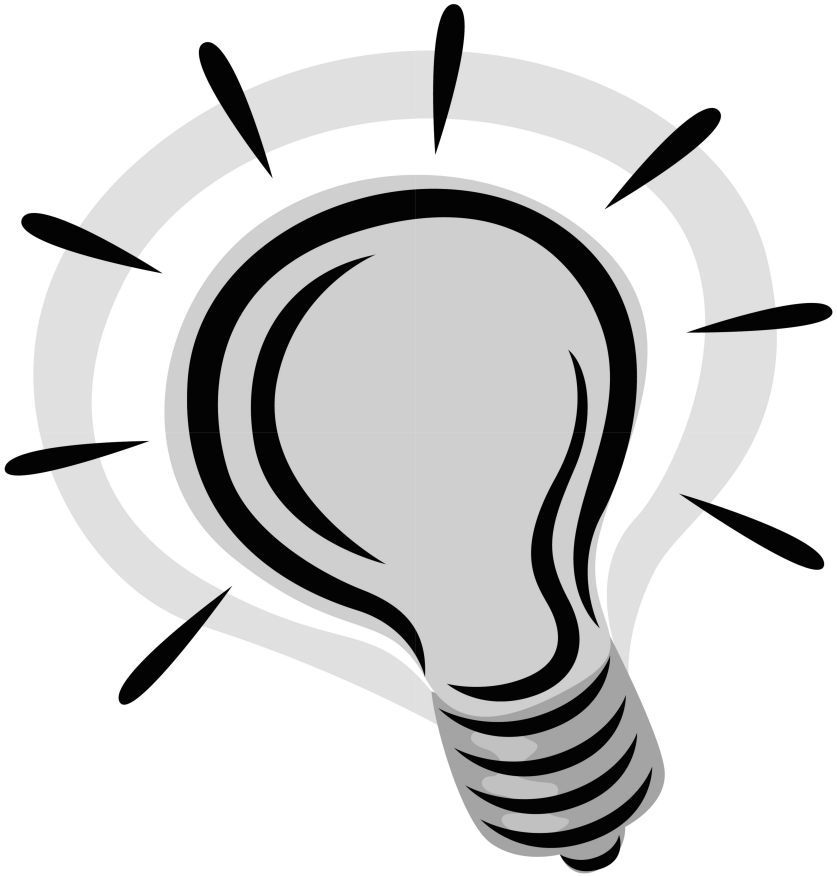
From a Talk

Lessons Learned from Building mbeddr

No direct support for detecting semantic interations between languages

Debugger definition separate from generator; leads to duplication

many aspects of language definition too „procedural" and hence hard to analyze.



Declarative!

Spoofax
SDF3

```
templates

  Definition.Function = <
    <Type> <ID>(<Param*; separator=",">) {
      <Statement*; separator="\n">
    }
  >

  Statement.If = <
    if(<Exp>)
      <Statement>
    else
      <Statement>
  >

  Statement.Return = <return <Exp>;>

  Exp.Add  = <<Exp> + <Exp>>

  Exp.Var  = <<ID>>
```

Spoofax
NaBL

```
binding rules

  Param(t, name) :
    defines Variable name

  Var(name) :
    refers to Variable name

  Function(t, name, param*, s) :
    defines Function name
    scopes Variable

  Call(name, exp*) :
    refers to Function name
```

Spoofax
TS

```
type rules

  Call(name, exp*) : t
  where definition of
                 name : (t*, t)
    and exp* : t_exp*
    and t_exp* == t*
```


Everything is wonderful.
Right?

Well, not quite.

Declarative?

Express real, „messy“ languages?

Can „normal“ people use it?

Procedural escapes?

Do they destroy the whole story?

Not clear. Needs Research.

➡ Spoofax@TU Delft

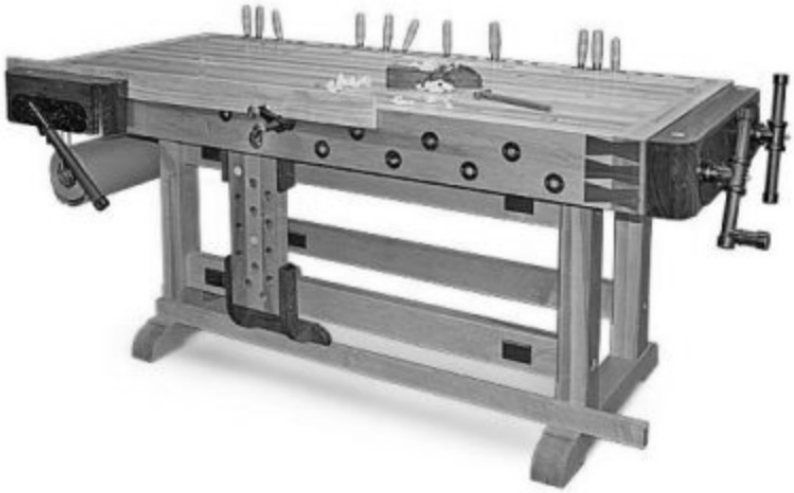
5

LWBs on the Web

Of course …

Language Workbench

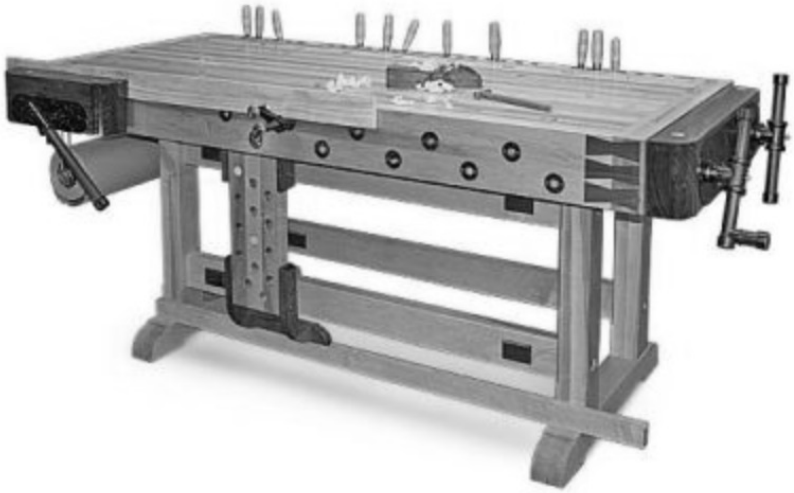
(Martin Fowler)



Freely
define
languages and
integrate
them

Language Workbench

(Martin Fowler)



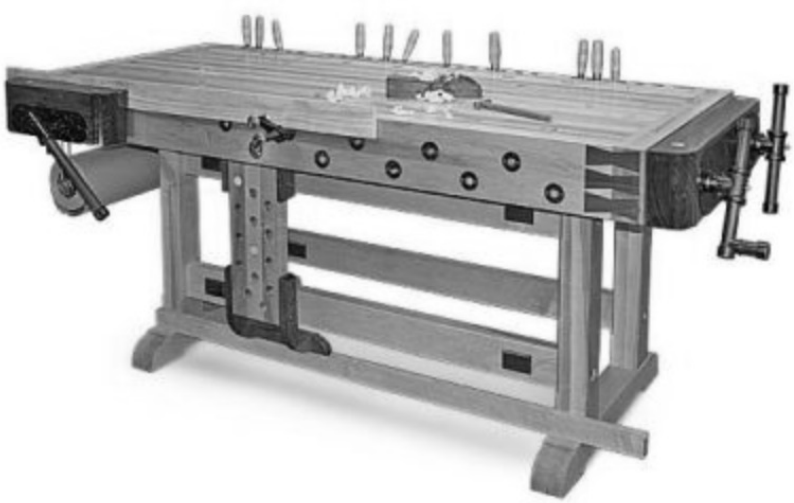
Freely
define
languages

web-based IDEs
or modeling tools
do not count ...

... but demonstrate
editing technology

Language Workbench

(Martin Fowler)



Freely

define

languages

and: initial tools will only use languages in the browser, not define them.

LWBs on the Web

Xtext and MPS teams are working on it

It will be months to years until these are finished/ robust.

Other web-LWBs (prototypes) exist: http://languageworkbenches.net

Editor Widget in the Browser

JVM on the Server

Initial Services:

Syntax Highlighting, Validation, Code Completion, Hovering, Formatting

More Services based on User Input

Use existing Xtext lang def

Q3 2015 earliest

Some Details:

vimeo.com/53935743

xrobots.itemis.de

Projectional Editor in Browser

incl. Hybrid Editing

No Server Component

No Dependency on MPS

Projectional Editor in Browser

incl. Hybrid Editing

No Server Component

No Dependency on MPS

Initial Services:

Editing, Syntax Highlighting,
Code Completion, Validation

Initially not based on MPS langs

New language stack to be back-ported eventually

Middle/Late 2015

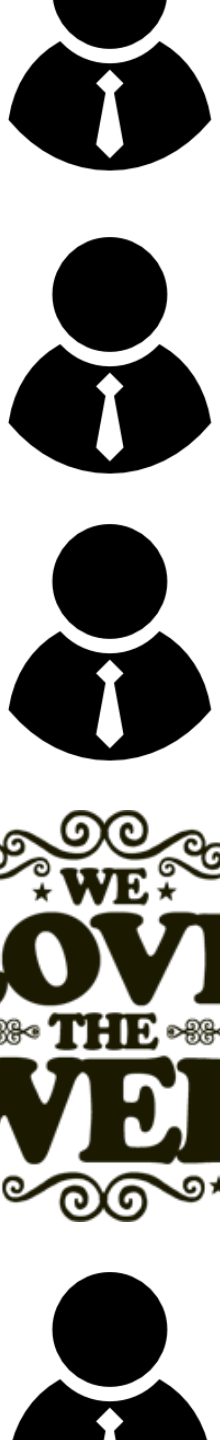
Longer Term:

Full Language Stack (Scopes, TS)

„OCL-like“ as default base language

Modular Language Aspects

LWBs on the Web

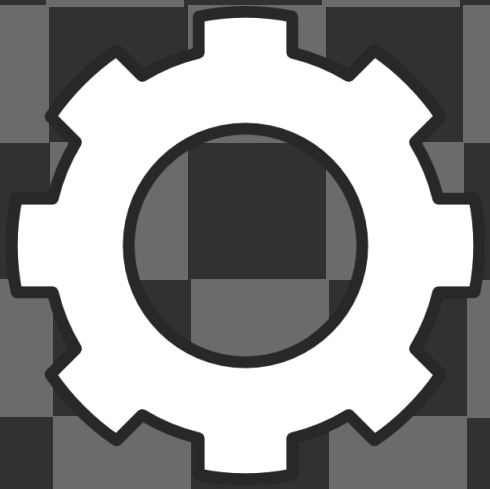


It will be years until these are as robust and feature-rich as rich client variants.

Or maybe ...

Make Web-IDEs Extensible?

DEMO



XRobots

6

Miscellaneous Trends

UI vs. Infrastructure

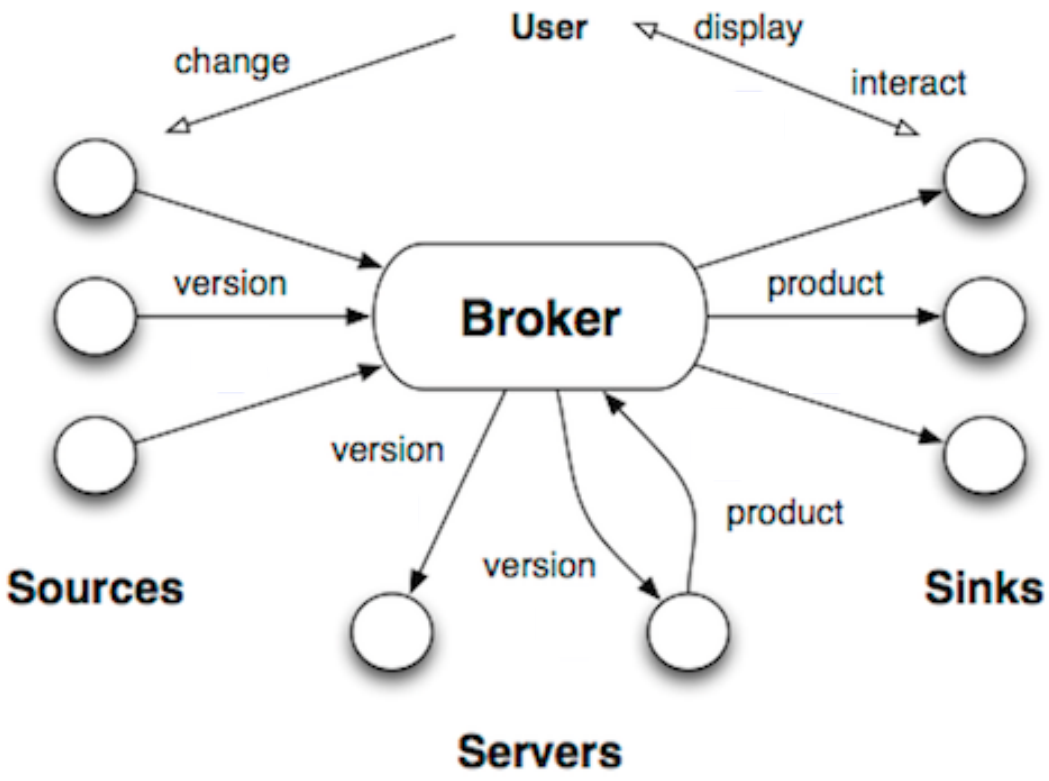
Keep the Client, Webify Infrastructure?

UI vs. Infrastructure

Monto:

Disintegrated DE, IDE Middleware

Python-based



IDE Independence

Library.domainmodel – [my-project] – my-project – [~/IdeaProjects/my-project]

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

my-project 〉 src 〉 Library.domainmodel 〉

Project

List.java × | Library.domainmodel × | BookStatus.java ×

- my-project (~/IdeaProjects/my-proje
  - .idea
  - src
    - BookStatus
    - Library.domainmodel
  - my-project.iml
  - pom.xml
- External Libraries

```
import java.util.List

/**
 * an ordinary library, Books only!
 */
entity Library {
    books : List<Book>
}

entity Book {
    title : String
    authors : List<Author>
    status : BookStatus
}

entity Author {
    name : String
}
```

Xtext on IntelliJ

JetBrains Nitra

DSLs for Visual Studio

NitraSyntax.nitra

```
syntax NitraType
{
  Typing() : NType * bool;

  | PredefinedType
    {
      override Typing
      {
        def value = match (GetText(this.PredefinedType))
        {
          | "void"   => NPredefinedType.Void
          | "int"    => NPredefinedType.Int
          | "string" => NPredefinedType.String
          | _        => assert3(false)
        };
        (NType.Predefined(value), false);
      }
    }
  | QualifiedType  = (NitraTypeName; ".")+
```

7

Why you should care

Language Engineering solves Real Problems!

Language Engineering solves Real Problems!

Improved Robustness, Testability, Maintainability and Performance in Embedded Software

Preliminary Experience of using mbeddr for Developing Embedded Software

http://voelter.de/data/pub/mbees2014.pdf

Language Engineering solves Real Problems!

Improved Security for C-based Software.

Improving Software Security using Language Engineering and mbeddr C

http://voelter.de/data/pub/mbeddr_security_voelter_2014_roadmap.pdf

Language Engineering solves Real Problems!

Better Integration of Stakeholders and more Efficiency in Business Applications.

Projecting a Modular Future

http://voelter.de/data/pub/projectingModuleFuture.pdf

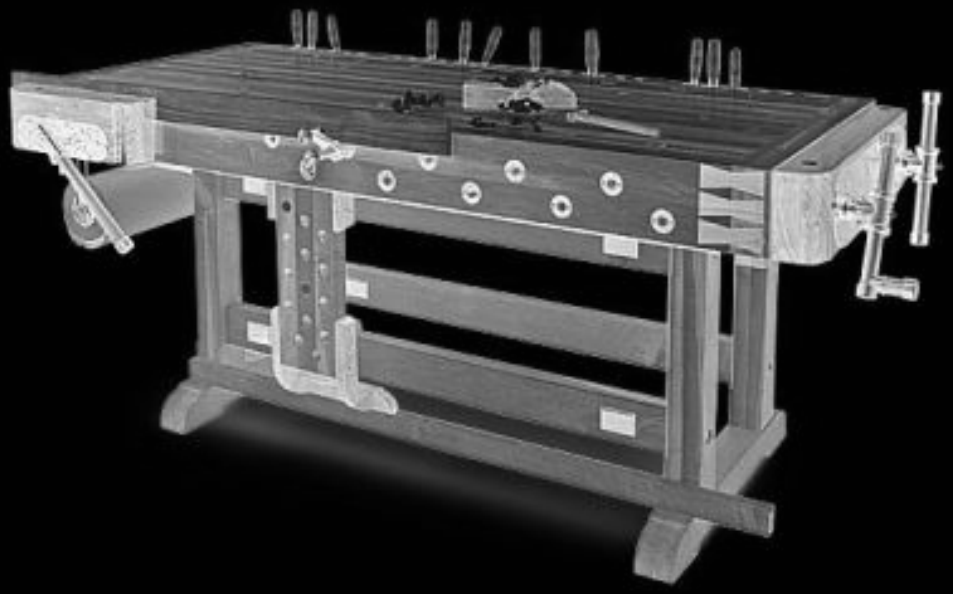
Language Engineering solves Real Problems!

More formal, checkable and hence more useful requirements and specifications.

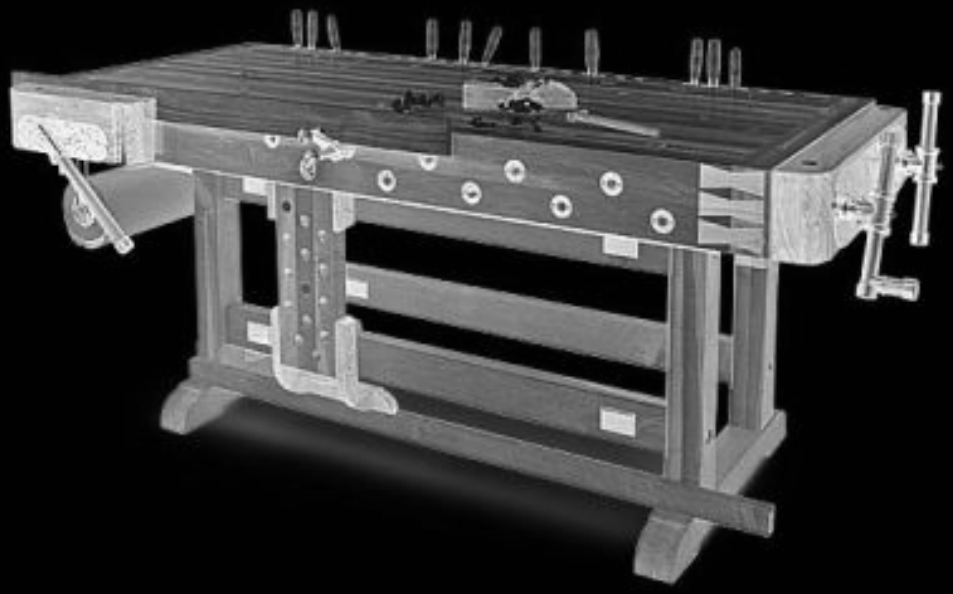
Projecting a Modular Future

http://voelter.de/data/pub/projectingModuleFuture.pdf

PROBLEM SOLVED



Thank you!



voelter@acm.org
www.voelter.de
@markusvoelter