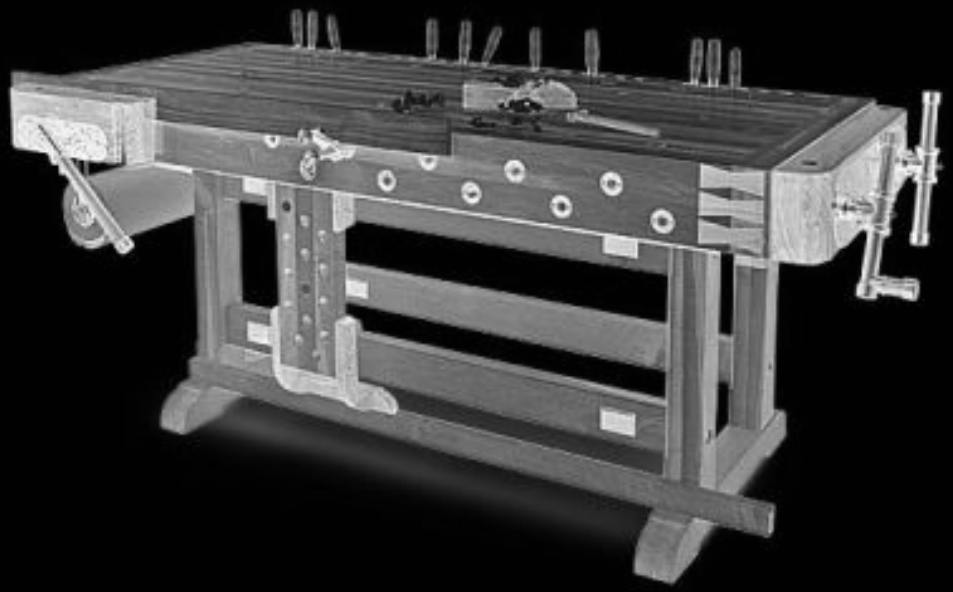
Language Workbenches
and DSLs for Embeeded S/W



voelter { ingenieurbüro für softwaretechnologie // itemis

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

This Talk

0 Languages, Models, Programs

1 Language Workbenches

2 MPS

3 mbeddr

4 Smart Meter

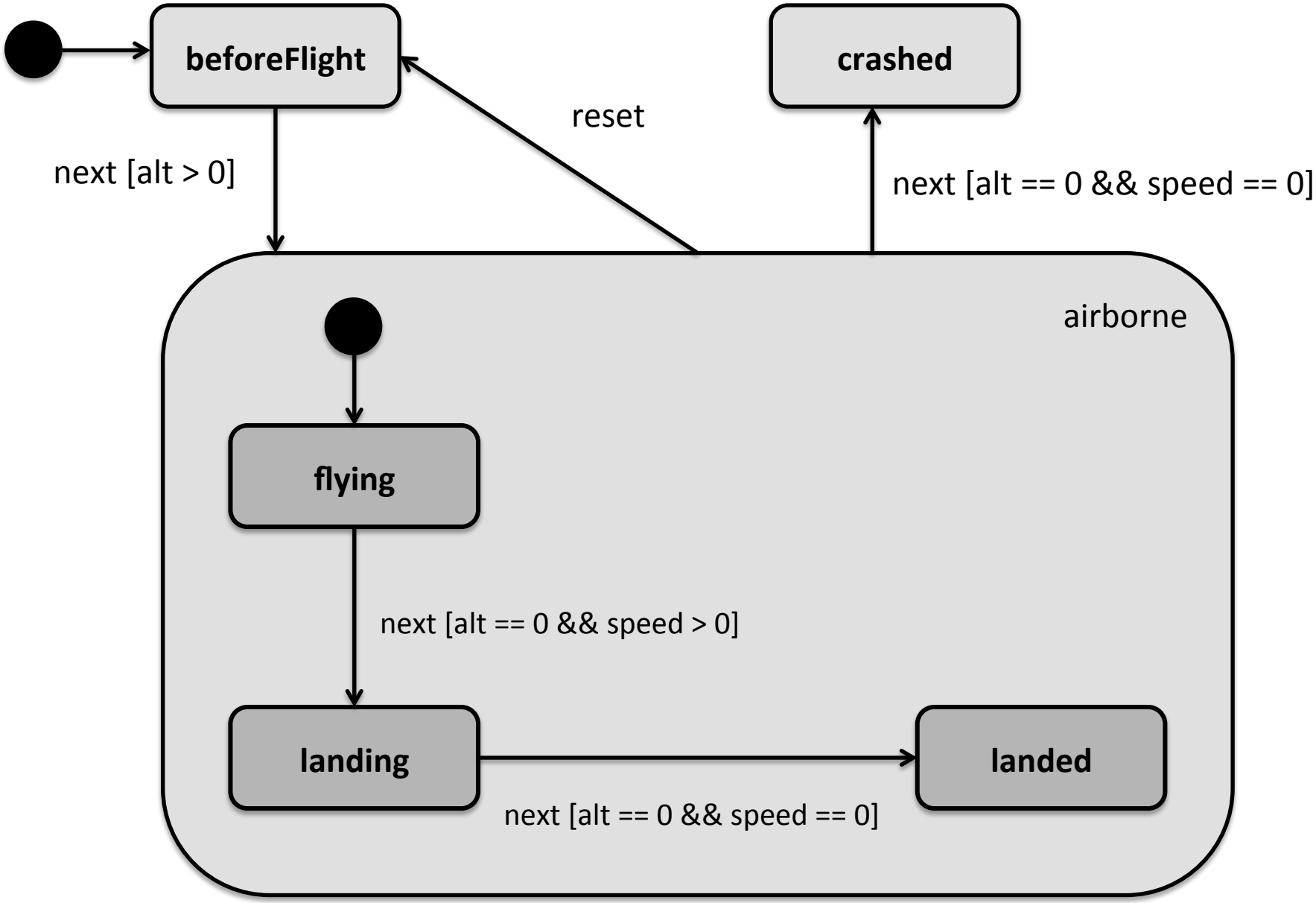
0

Languages, Models, Programs

[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?

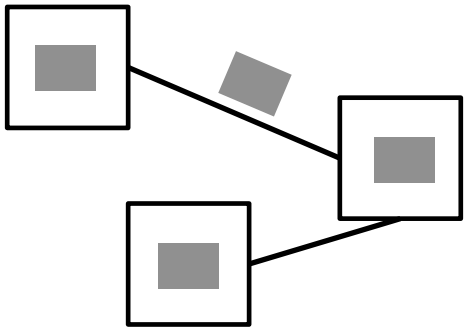
What is the difference?

Who cares?

[Model or Code]

Code

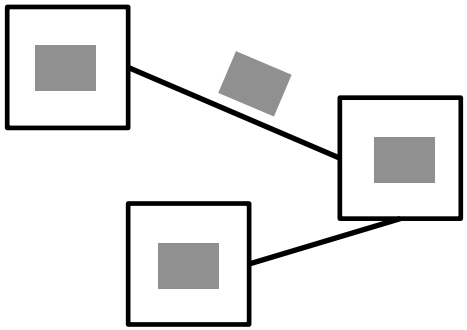
Model



[Model or Code]

Code

Model



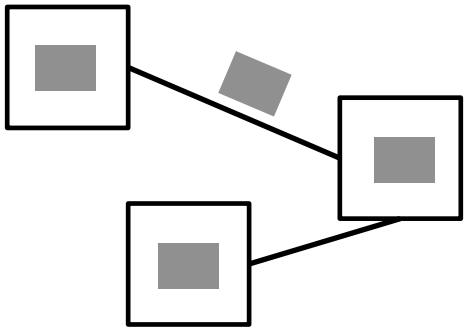
?

?

[Model or Code: Notation!]

Textual

Graphical



Tabular

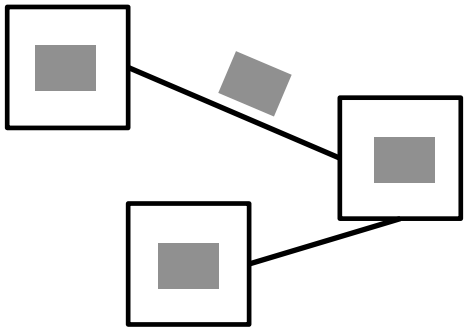
Mathematical

[Notation]

One important aspect of languages.

Textual

Graphical

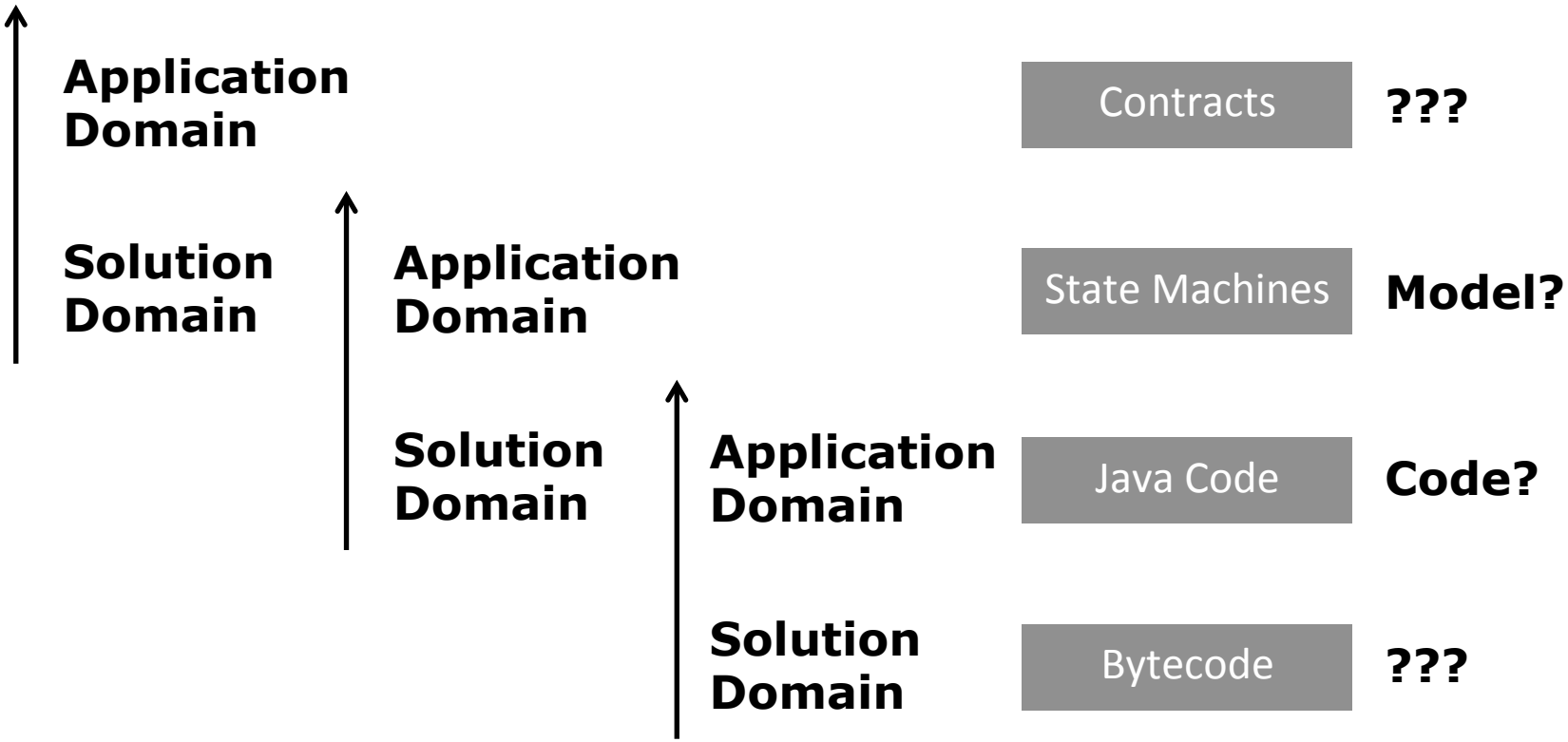


Tabular

Mathematical

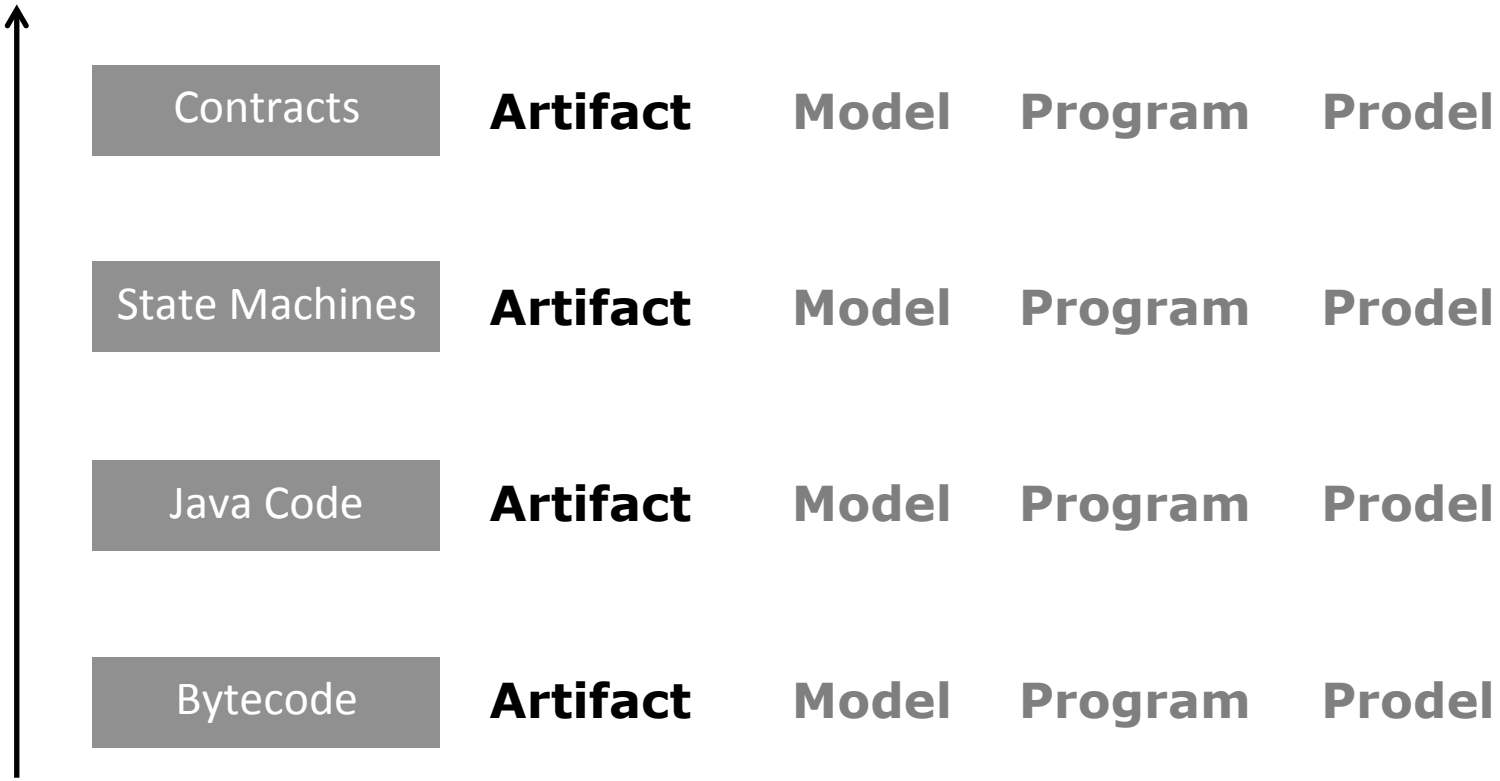
[Abstraction]

The other important aspect.

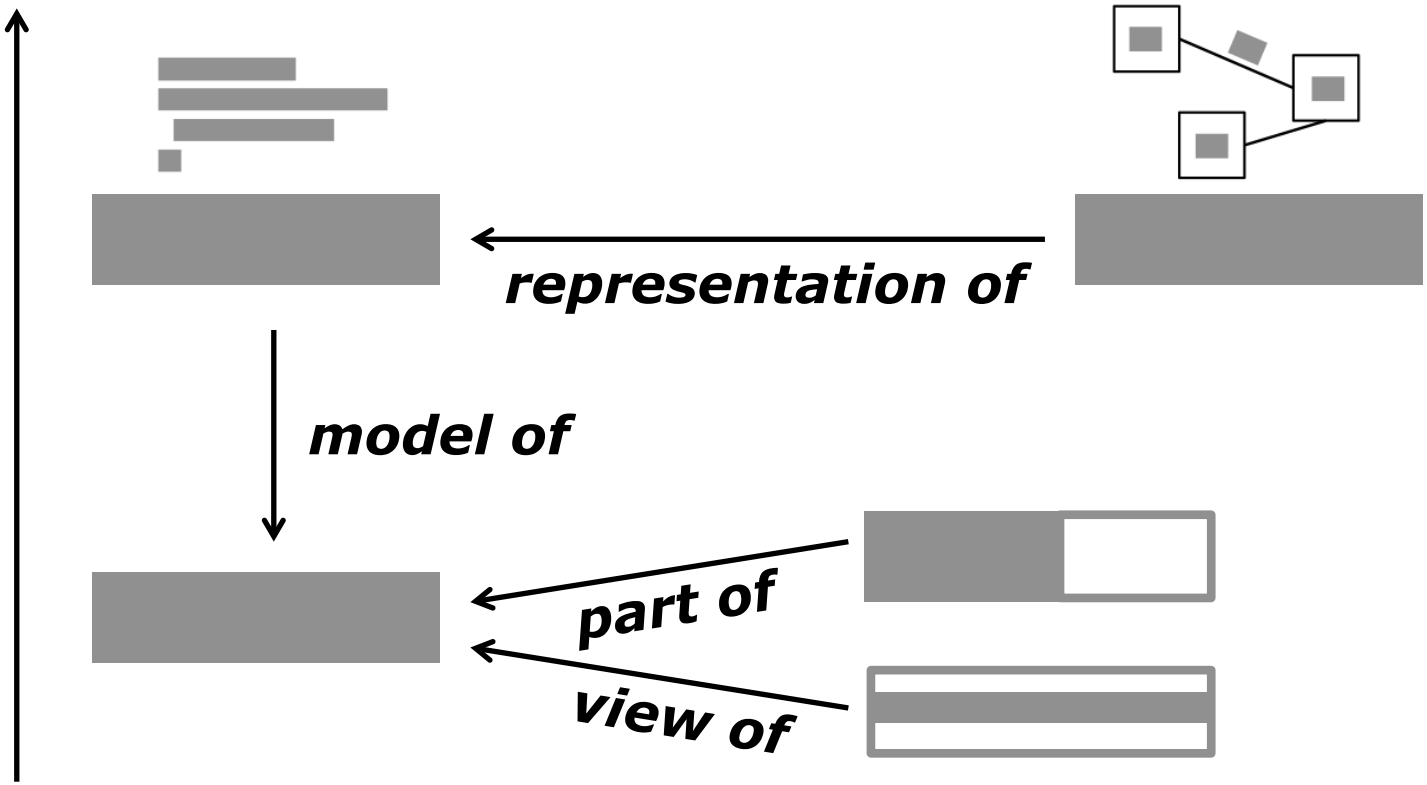


[Abstraction]

The other important aspect.



[Relationships]



[Relationships]

model of Abstraction

part of Hierarchy, Partitioning

view of Concern, Aspect, Viewpoint

representation of
Presentation, Notation, Form

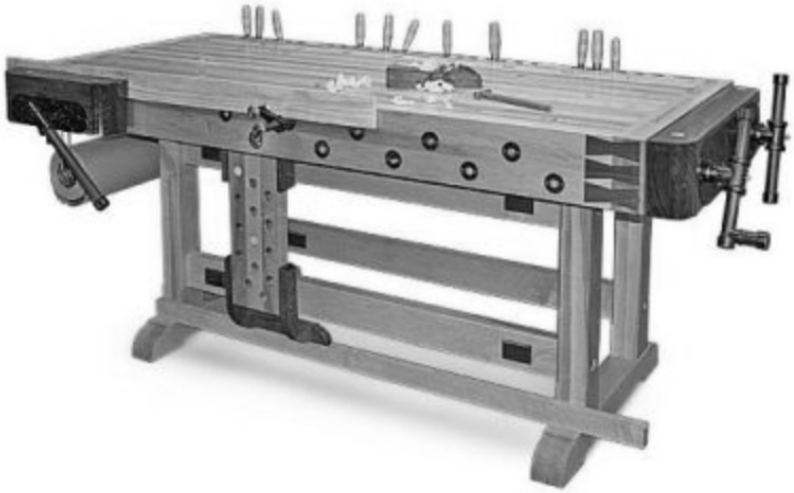
1st

Language Workbenches

Language Workbench(es)

... are the tools to do:

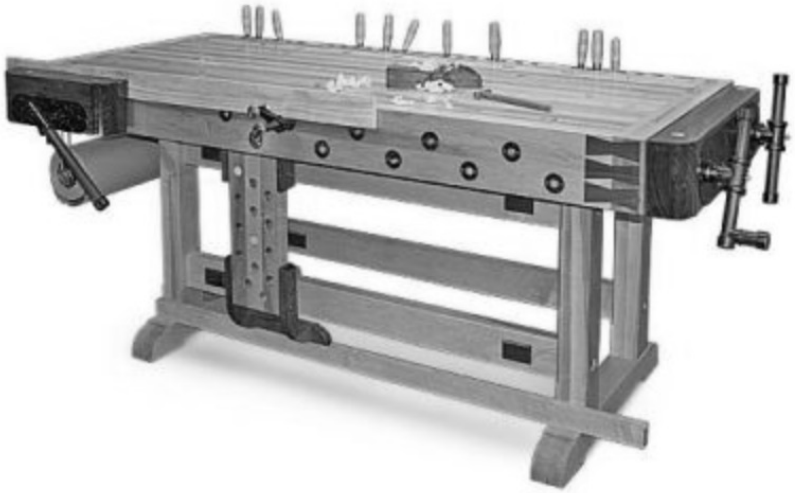
Model-Based Engineering • Model-Driven (Software) Development • Domain-Specific Languages • Language-Oriented Programming



Language Workbench

(Martin Fowler)

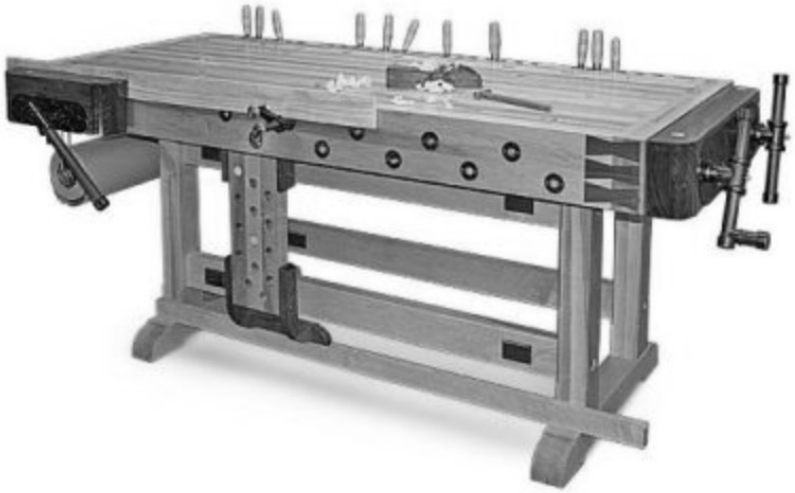
Freely
define
languages and
integrate
them



Language Workbench

(Martin Fowler)

use
persistent
abstract
representation



Language Workbench

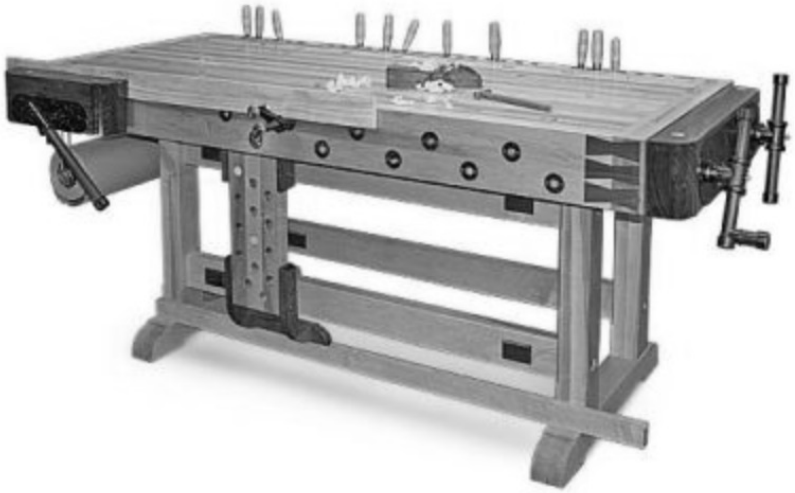
(Martin Fowler)

language ::=

schema

+ editors

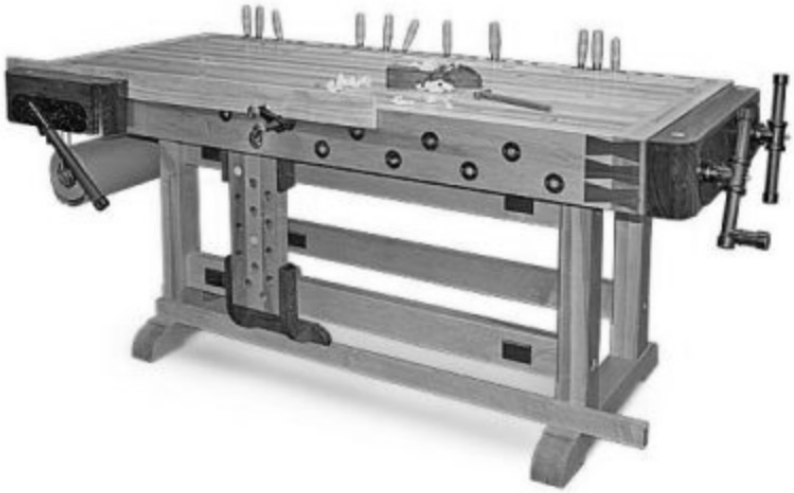
+ generators



Language Workbench

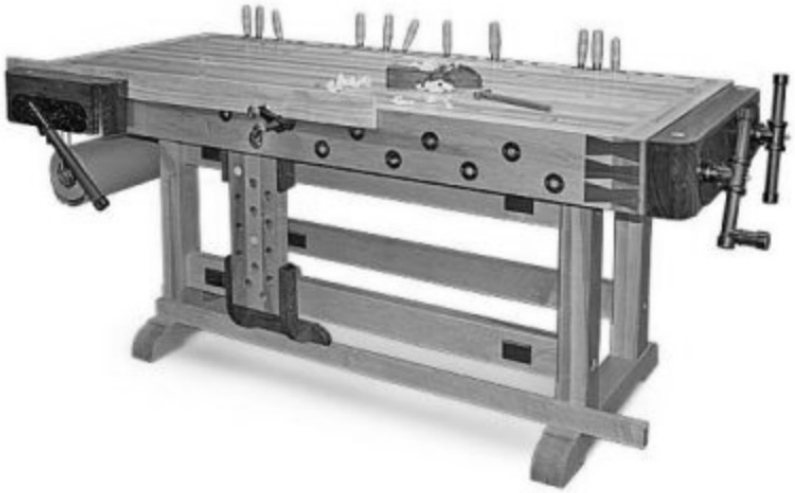
(Martin Fowler)

projectional editing ?



Language Workbench

(Martin Fowler)



persist

incomplete

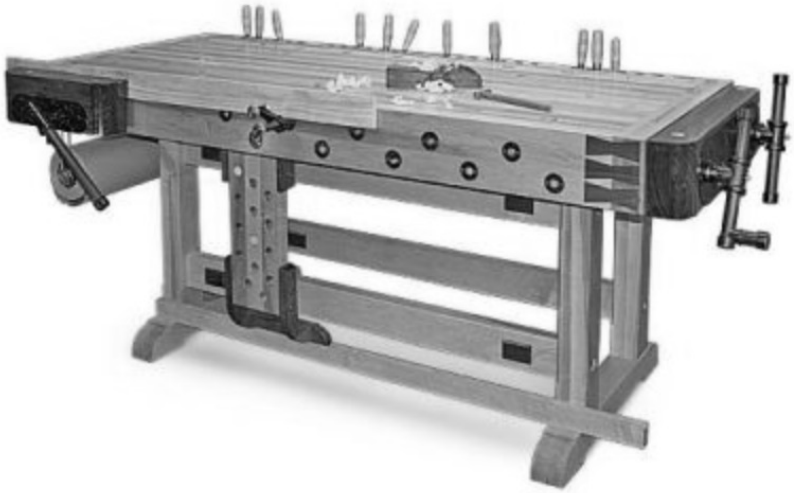
or

contradictory

information

Language Workbench

(Martin Fowler)



powerful

editing

testing

refactoring

debugging

groupware

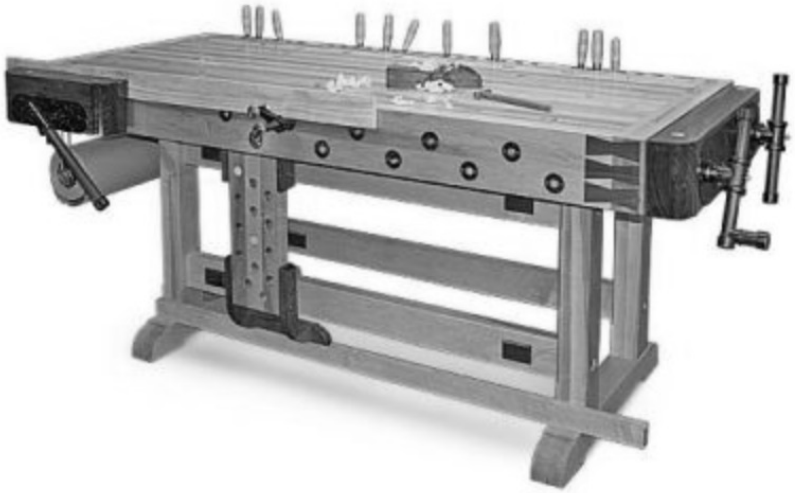
language definition

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

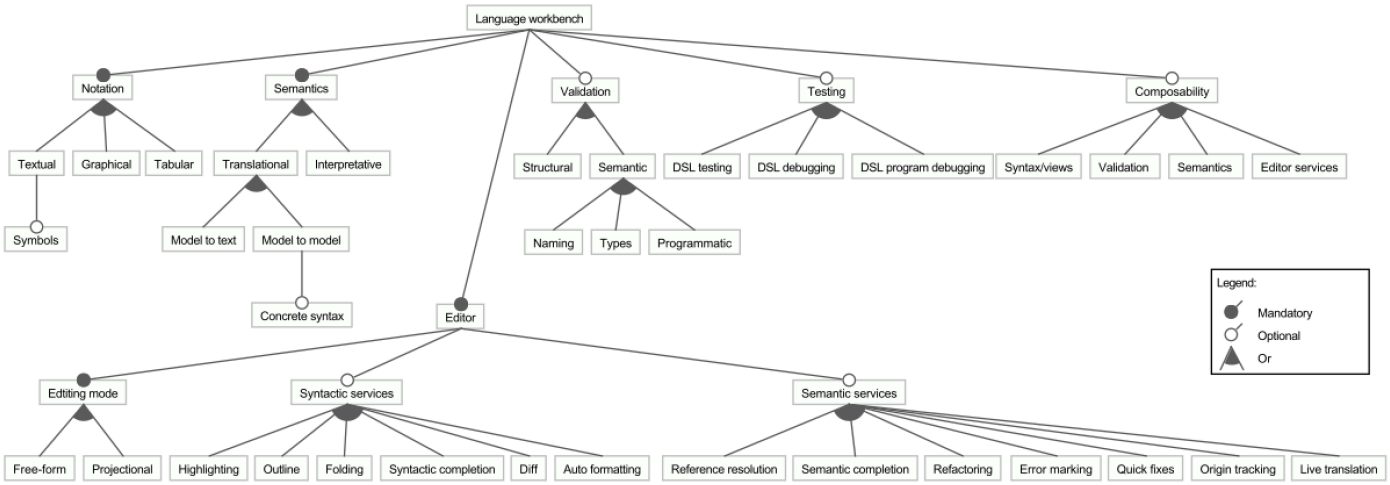
About LWC | Workshop schedule | Call for participation | Past editions | Submissions

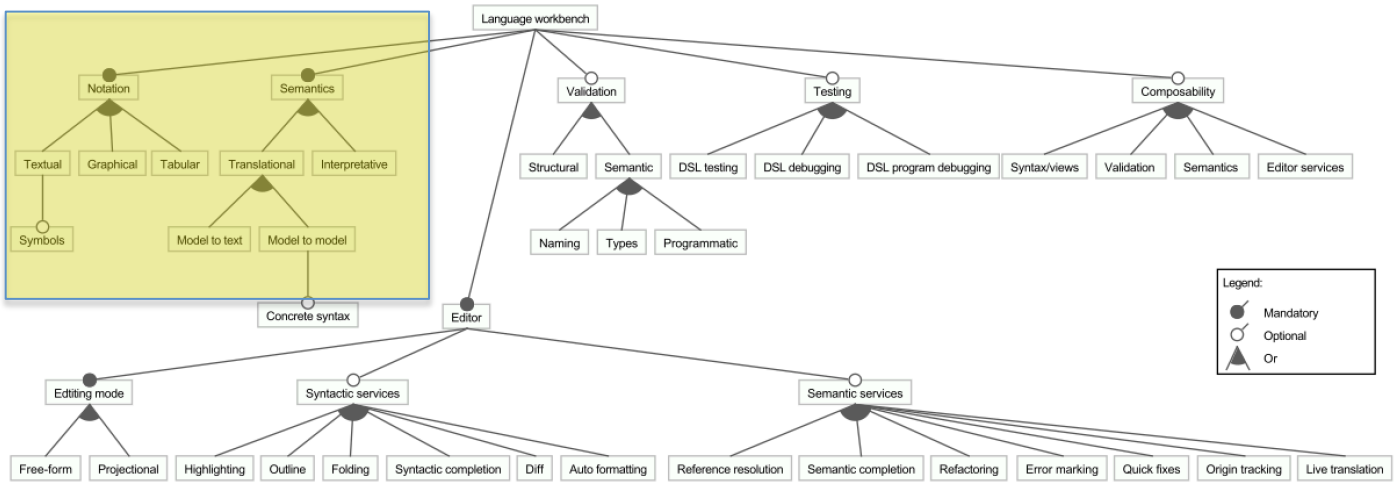
The State of the Art in Language Workbenches

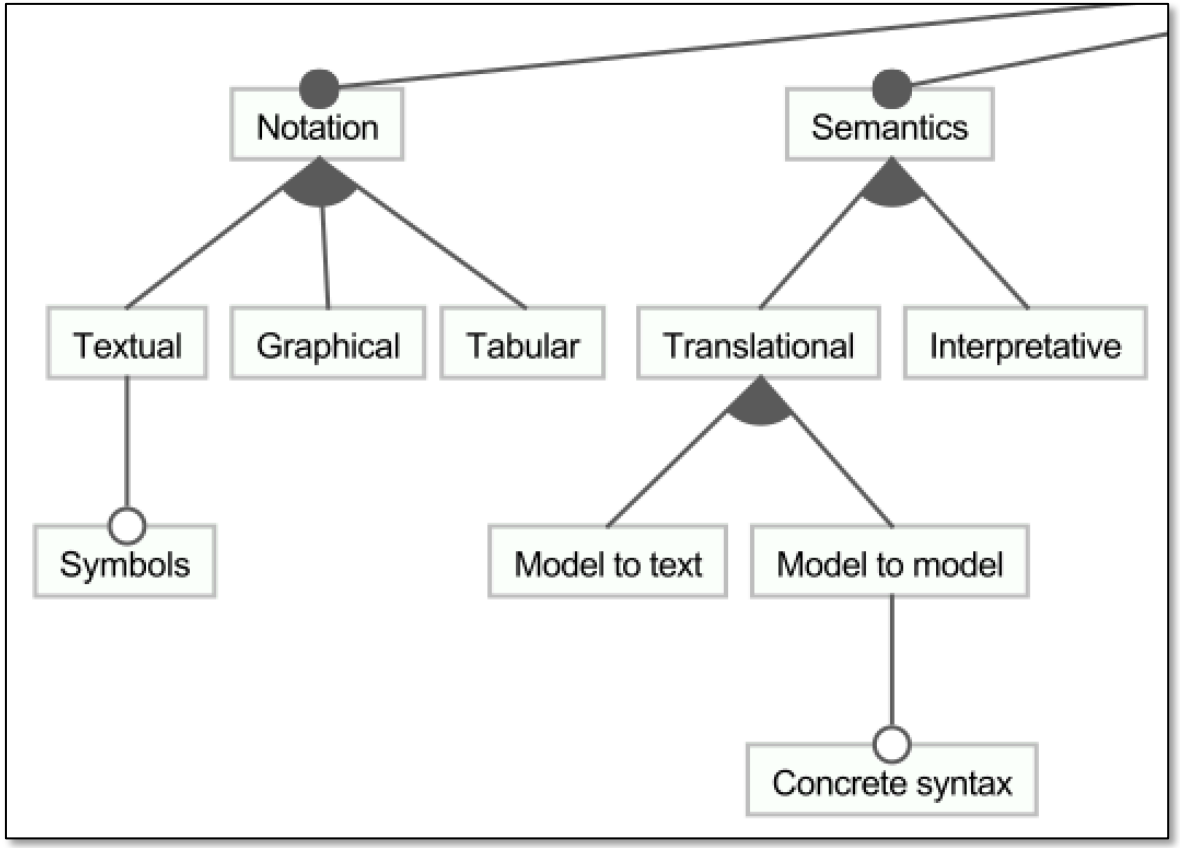
Conclusions from the Language Workbench Challenge

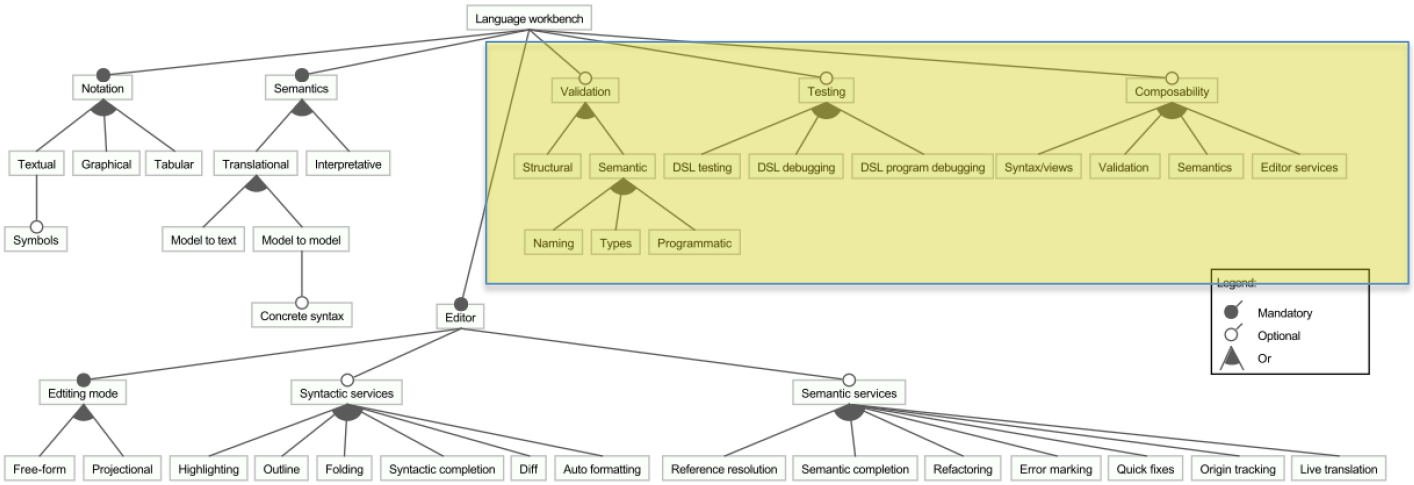
Sebastian Erdweg[1], Tijs van der Storm[2,3], Markus Völter[4], Meinte Boersma[5], Remi Bosman[6], William R. Cook[7], Albert Gerritsen[6], Angelo Hulshout[8], Steven Kelly[9], Alex Loh[7], Gabriël Konat[10], Pedro J. Molina[11], Martin Palatnik[6], Eugen Schindler[6], Klemens Schindler[6], Riccardo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

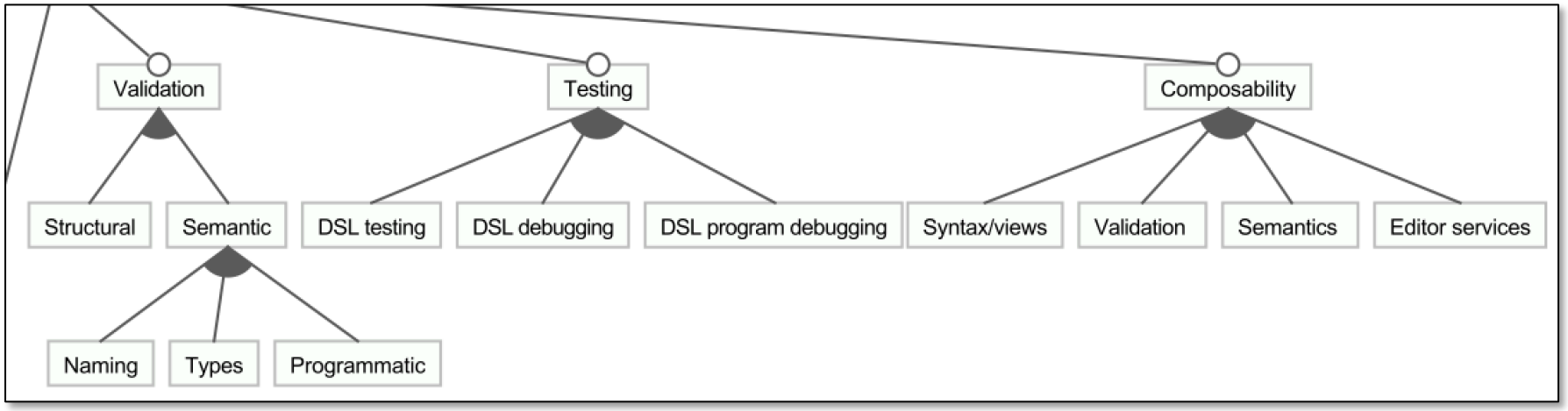
[1] TU Darmstadt, Germany [2] CWI, Amsterdam, The Netherlands [3] INRIA Lille Nord Europe, Lille, France [4] voelter.de, Stuttgart, Germany [5] DSL Consultancy, Leiden, The Netherlands [6] Sioux, Eindhoven, The Netherlands [7] University of Texas, Austin, US [8] Delphino Consultancy, Best, The Netherlands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

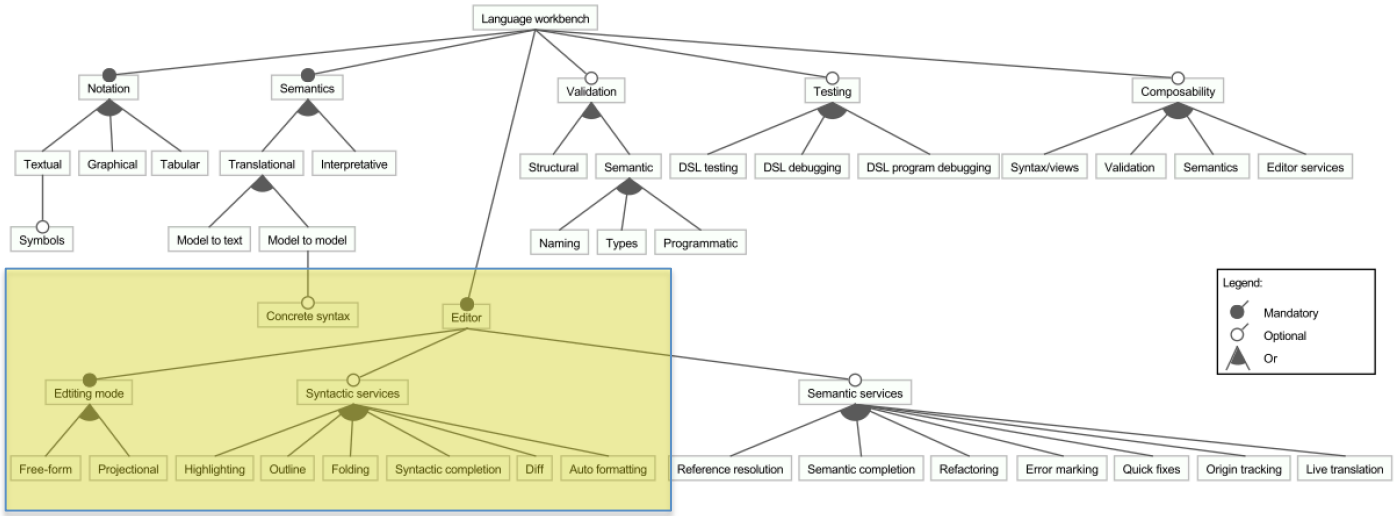

Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
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

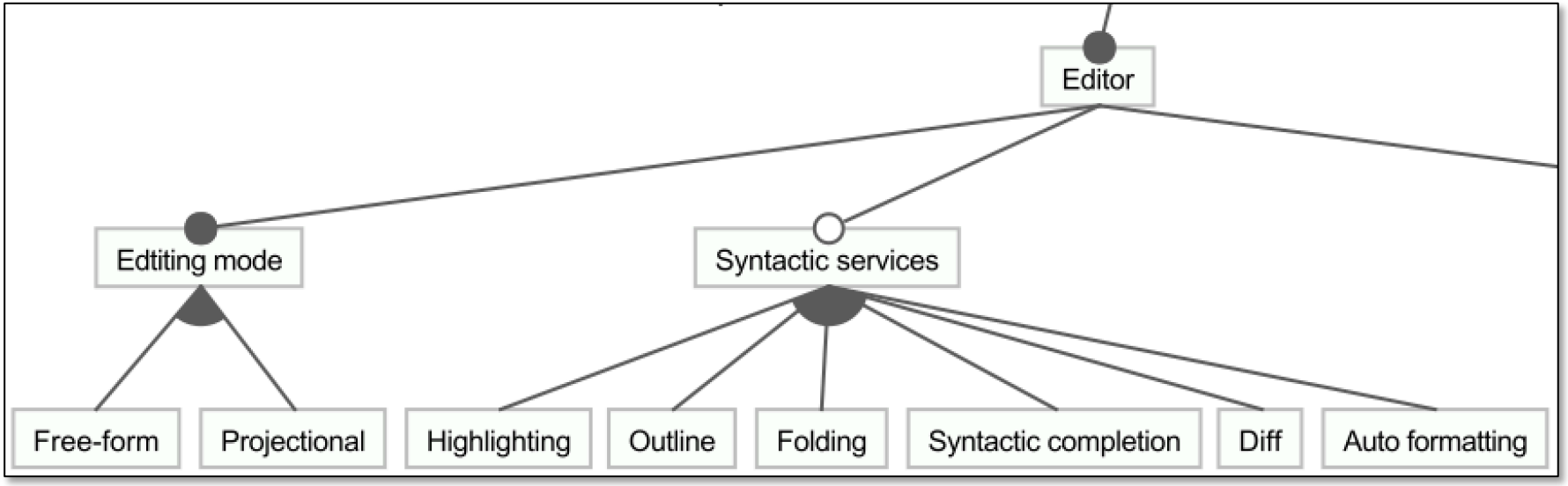

Language workbench
Notation
Semantics
Textual
Graphical
Tabular
Translational
Interpretative
Symbols
Model to text
Model to model
Concrete syntax
Validation
Testing
Composability
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Naming
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

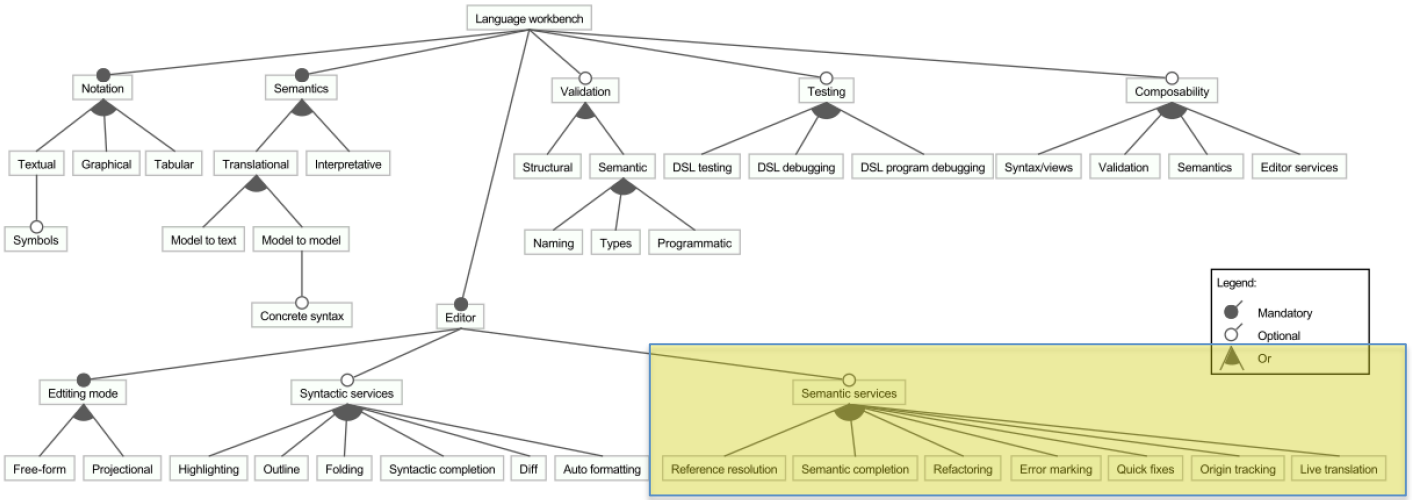

Notation
Semantics
Textual
Graphical
Tabular
Translational
Interpretative
Symbols
Model to text
Model to model
Concrete syntax

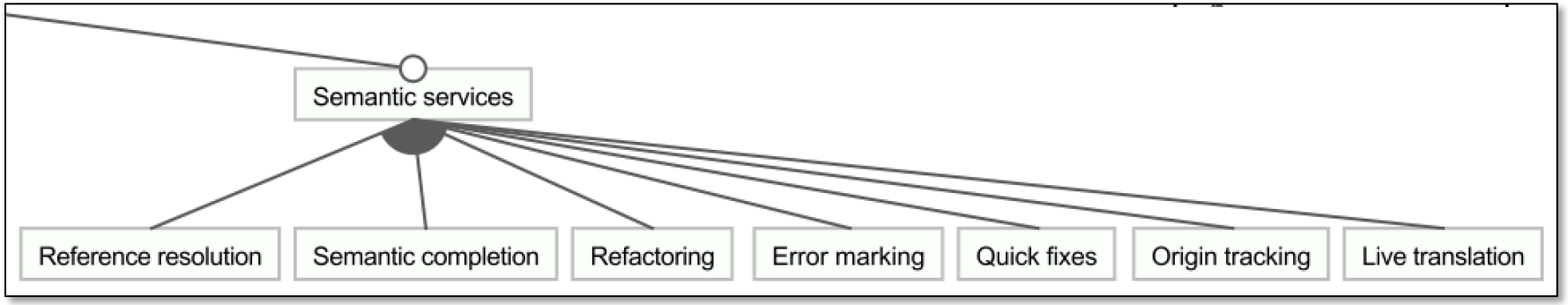

Language workbench
Notation
Semantics
Validation
Testing
Composability
Textual
Graphical
Tabular
Translational
Interpretative
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Symbols
Model to text
Model to model
Naming
Types
Programmatic
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

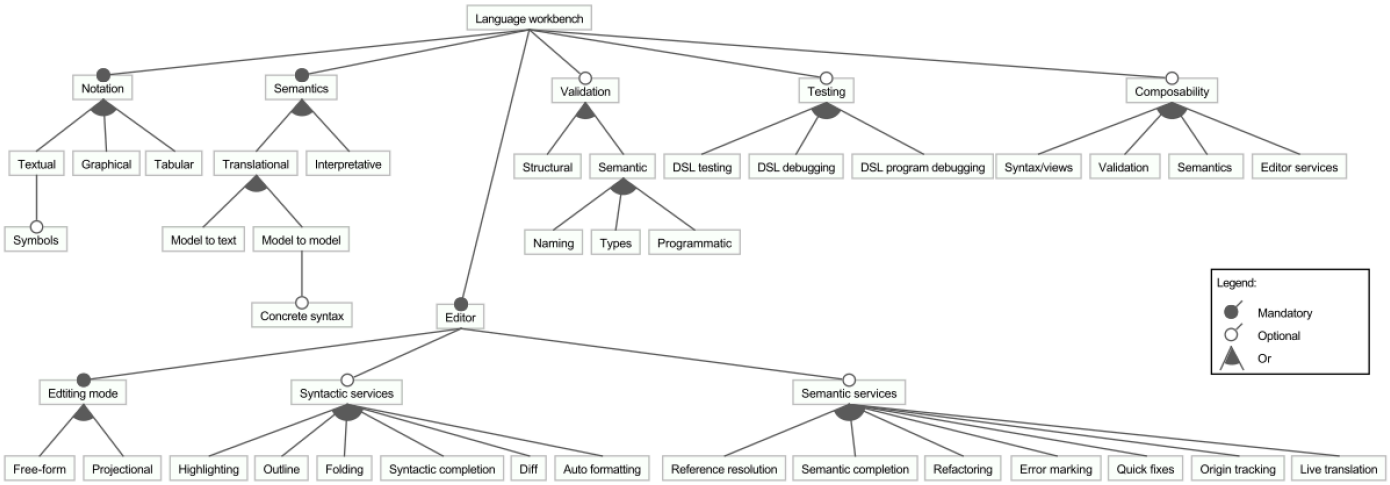

Validation
Testing
Composability
Structural
Semantic
DSL testing
DSL debugging
DSL program debugging
Syntax/views
Validation
Semantics
Editor services
Naming
Types
Programmatic


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

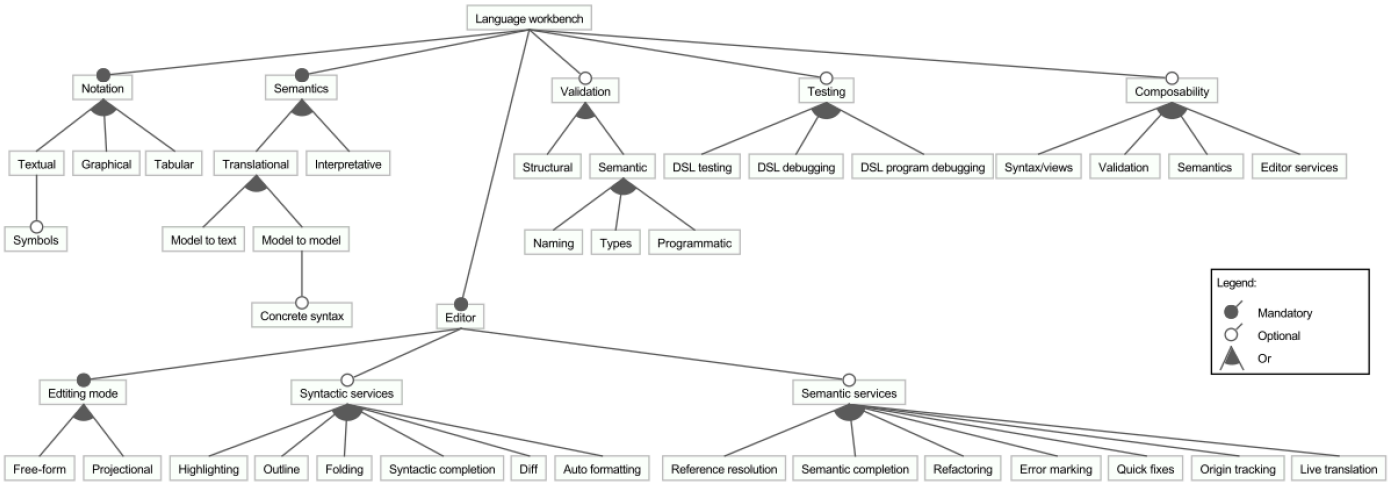

Editor
Edtiting mode
Syntactic services
Free-form
Projectional
Highlighting
Outline
Folding
Syntactic completion
Diff
Auto formatting


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

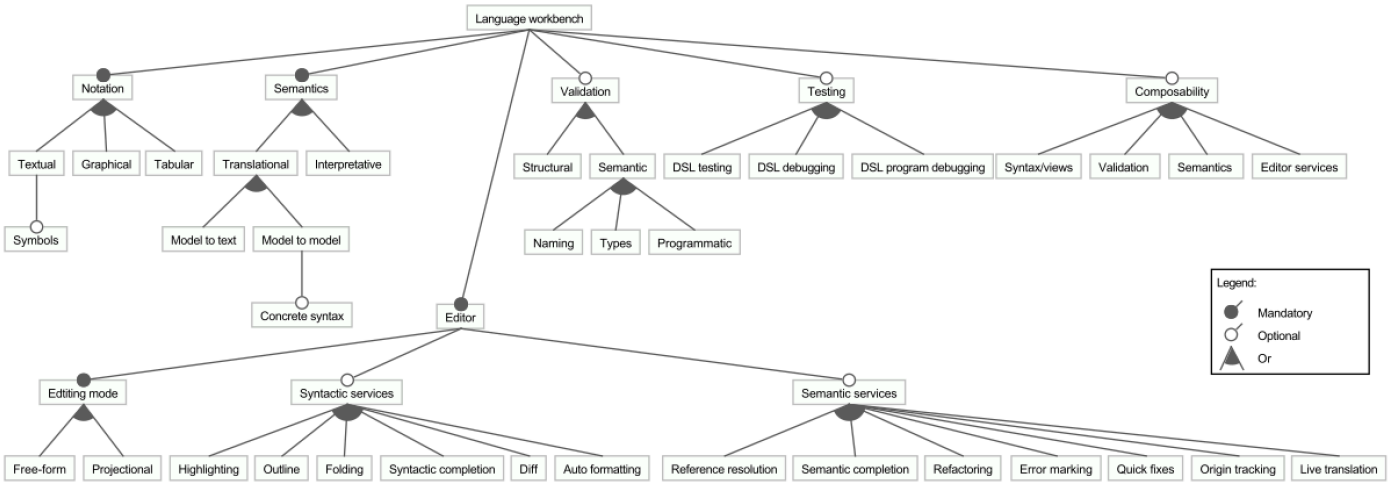

Semantic services
Reference resolution
Semantic completion
Refactoring
Error marking
Quick fixes
Origin tracking
Live translation



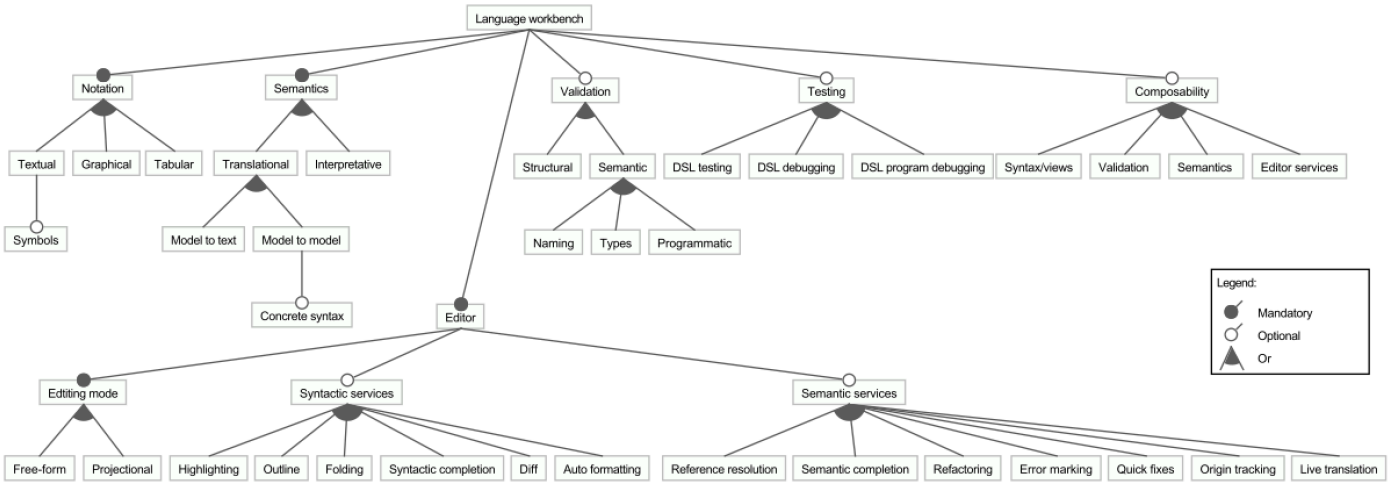
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒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



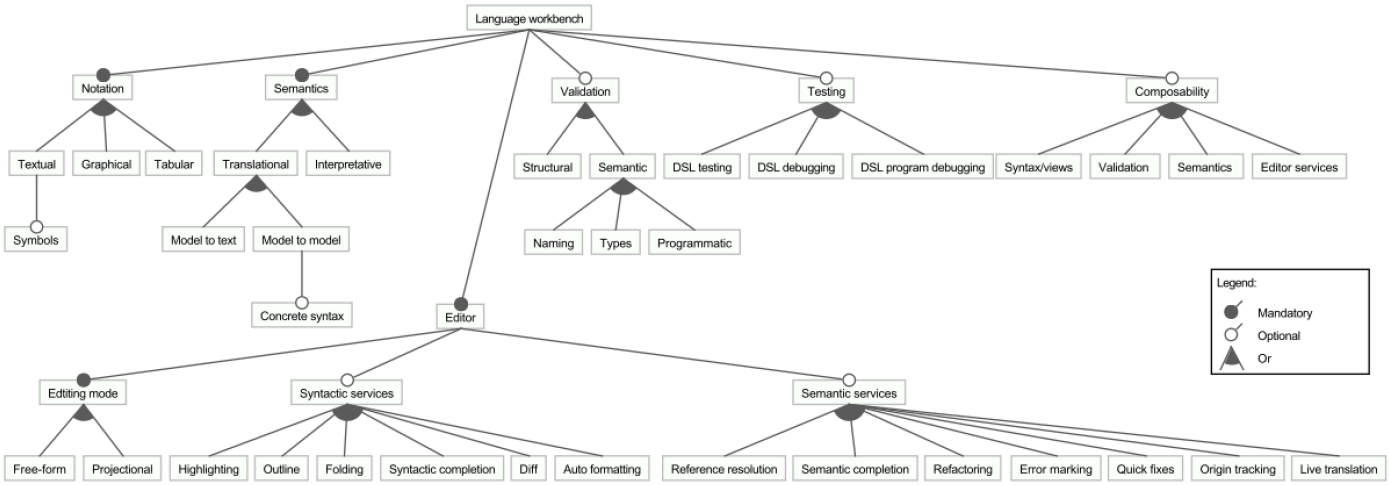
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒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



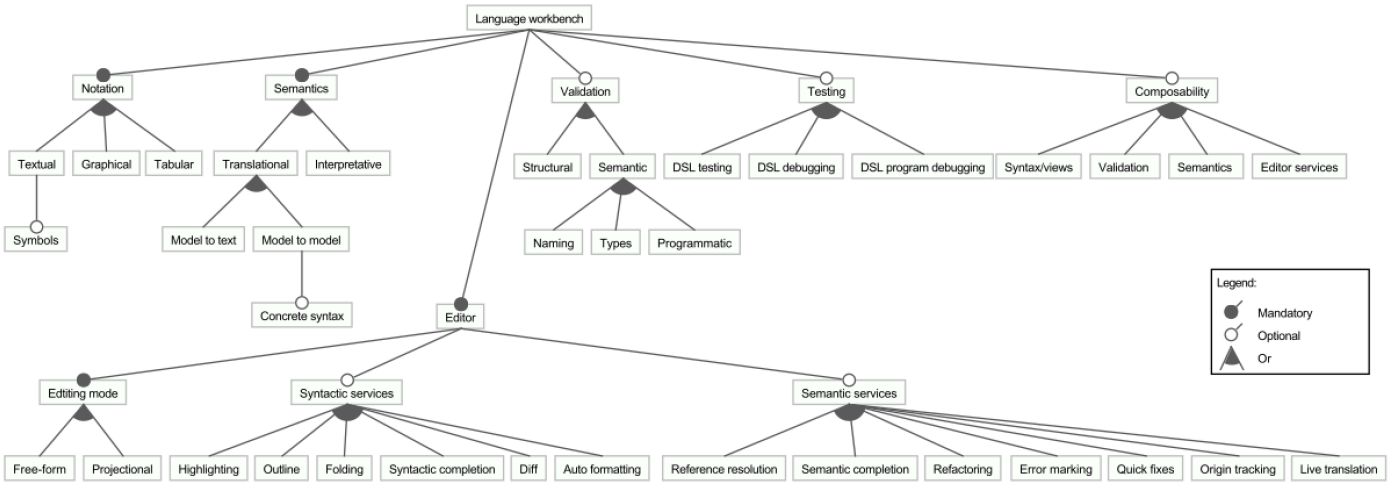
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒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



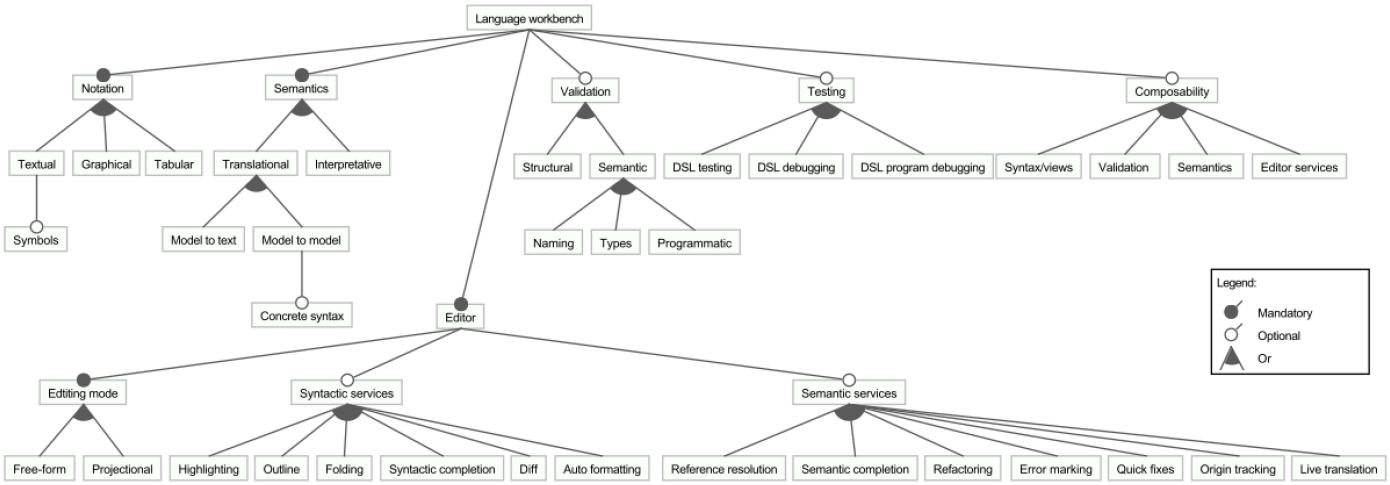
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒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



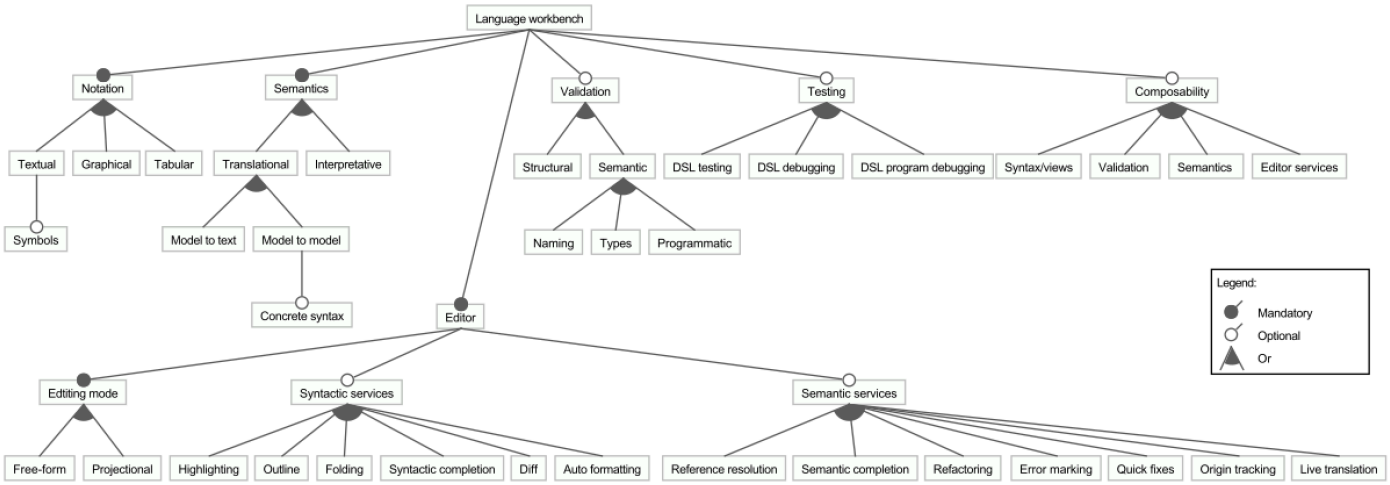
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●	●		◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒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



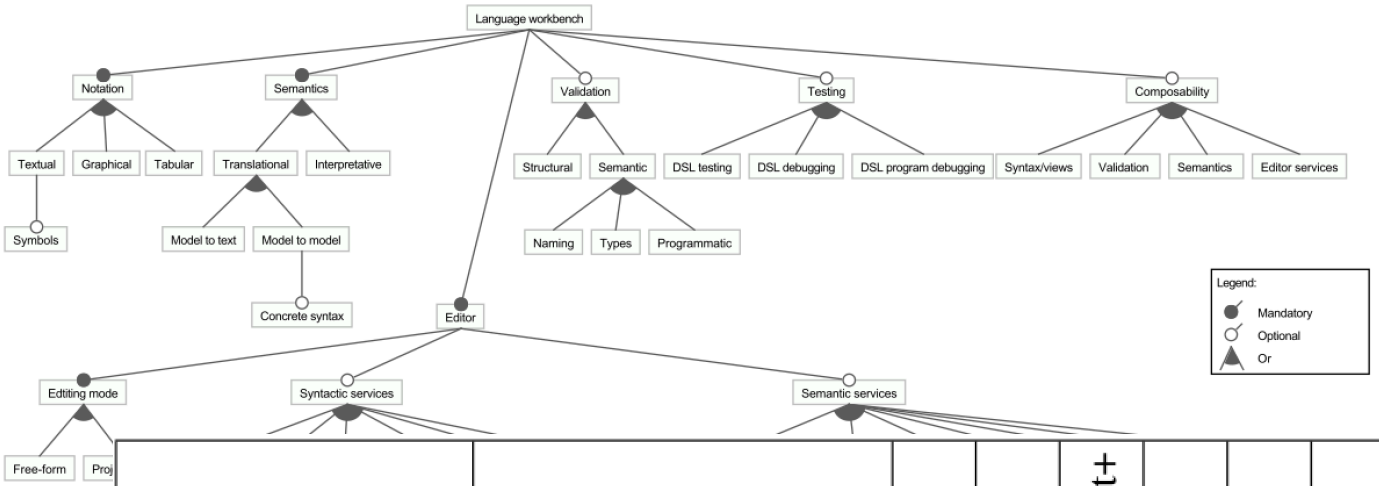
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◒	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◒
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●	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



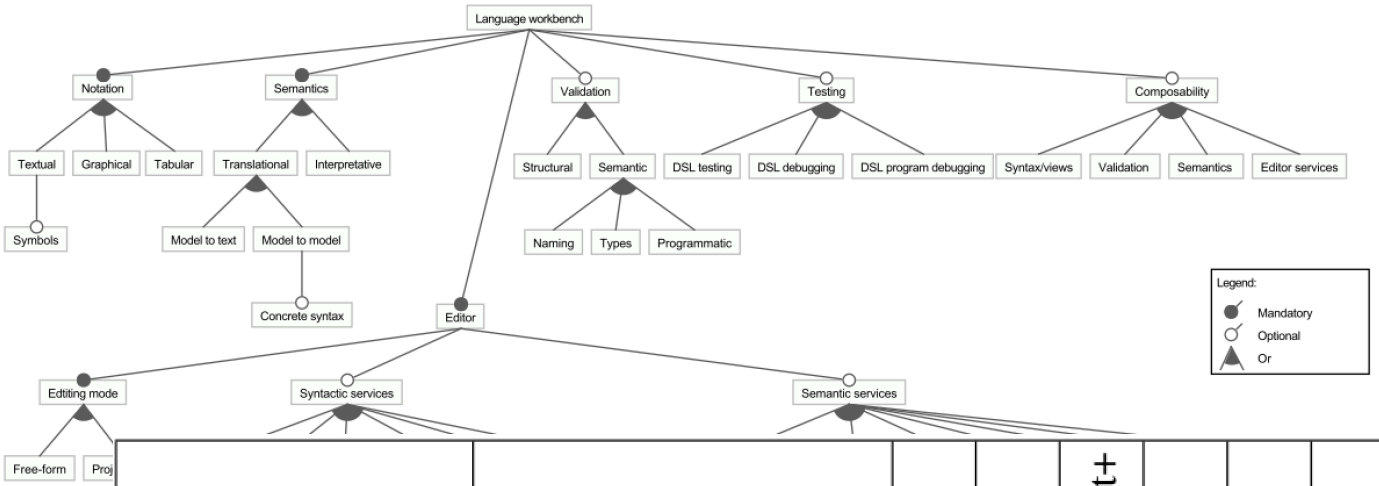
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◐	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◐
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●	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



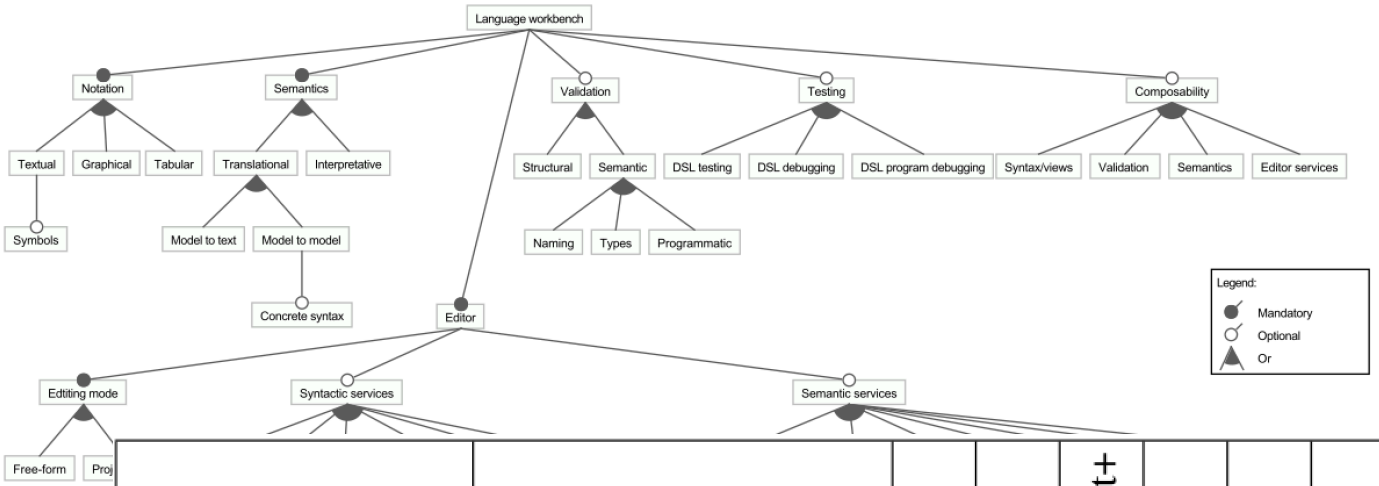
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Testing	DSL testing				●		◐	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◐
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
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			●		●	◒	●		●	●



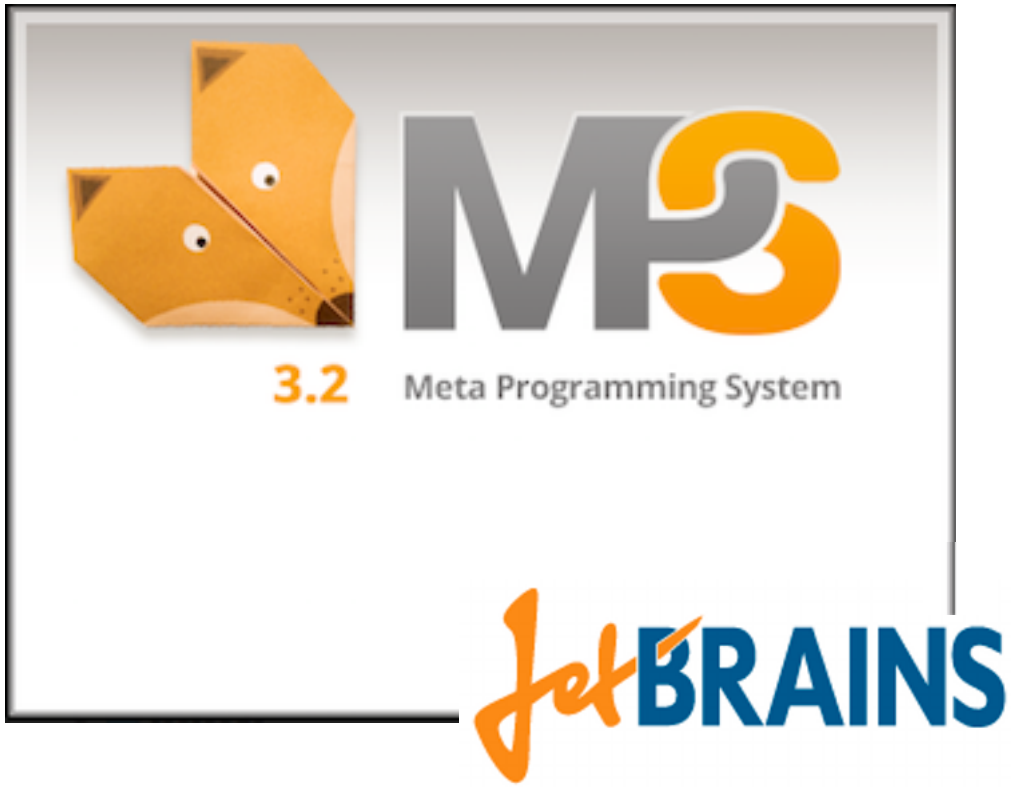
		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Syntactic services	Highlighting		◒	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding		●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	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			●		●	◒	●		●	●

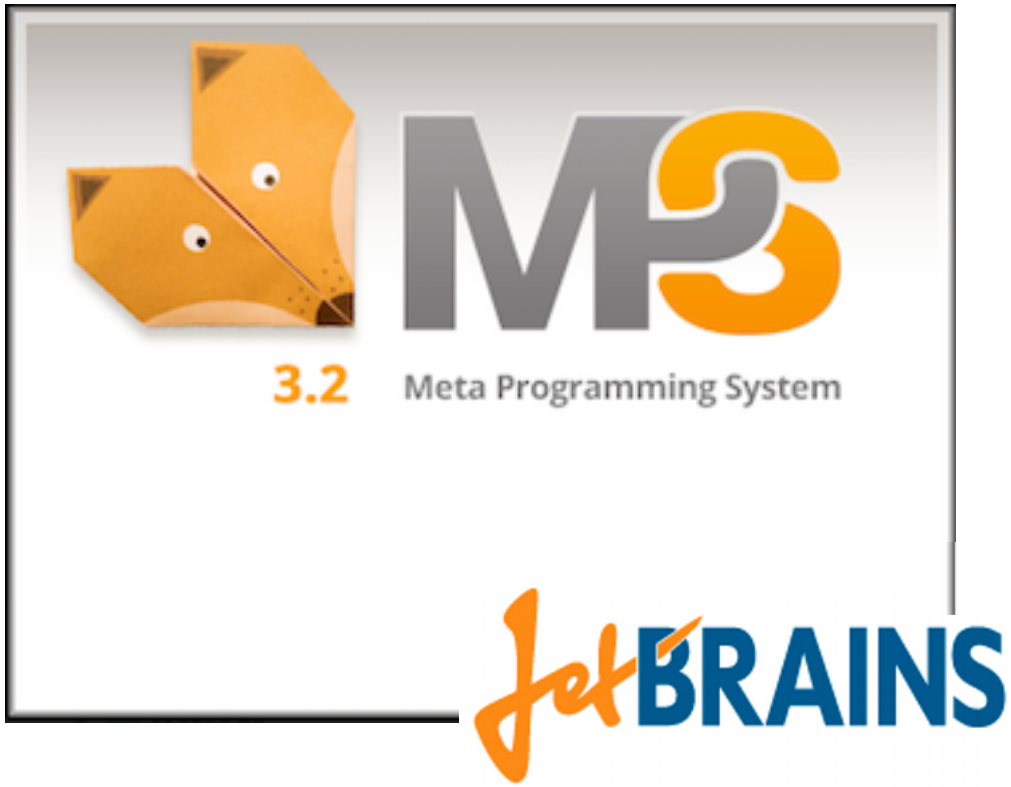
2

MPS

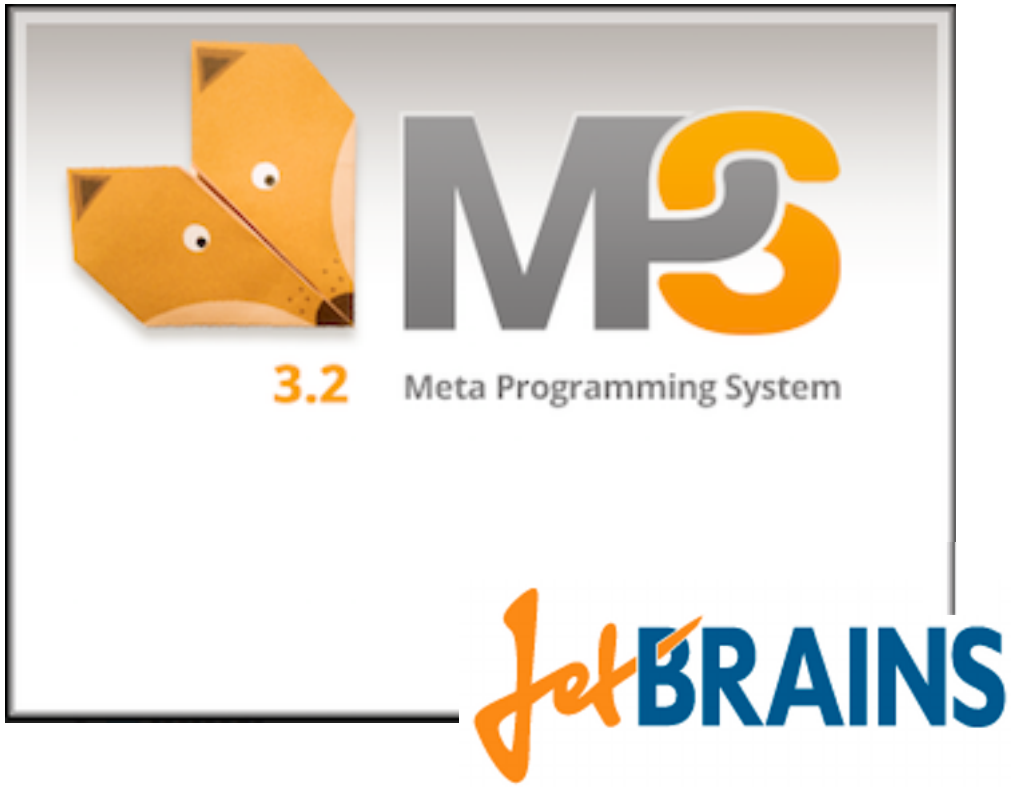


A Language Workbench –

a tool for defining, composing and using ecosystems of languages.



Open Source
Apache 2.0
http://jetbrains.com/mps

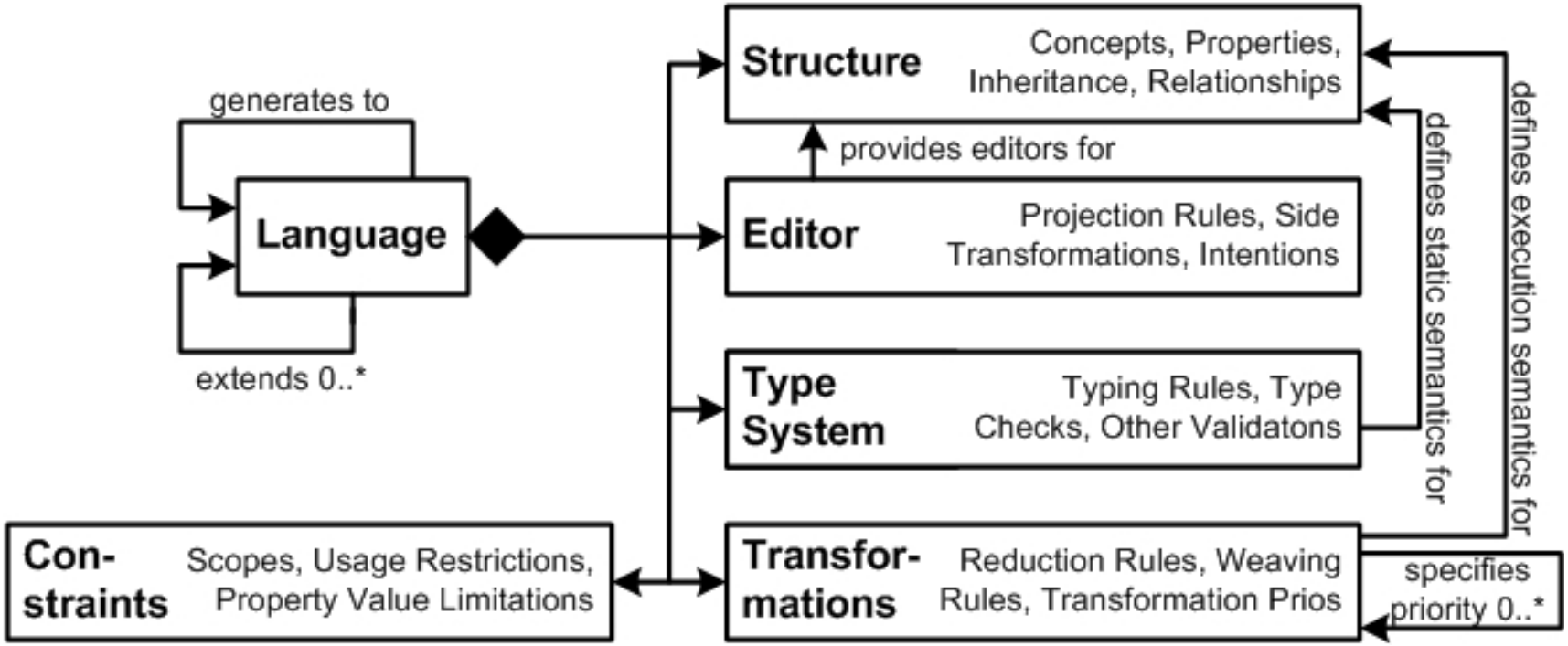


V 3.2 is current.

V 3.3 to be released in Q4 2015.

[Language Workbench]

Comprehensive Support for many aspects of Language Definition.



+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...

[Comprehensive IDE Features]

For End Users and Language Developers

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop

[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight {
  in event next(Trackpoint* tp) <no binding>
  in event reset() <no binding>
  out event crashNotification() => raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    //[ Here is a comment on a transition. ]
    on next [tp->alt == 0 m] -> airborne
    exit { points += TAKEOFF; } -> implements PointsForTakeoff
  } state bef
  state airborne {
    on next [tp->alt == 0 m && tp->speed == 0] -> crashed
    on next [tp->alt == 0 m && tp->
    on next [tp->speed > 200 mps &
    on next [tp->speed > 100 mps &
    on reset [ ] -> beforeFlight
  } state airborne
  state landing {
    on next [tp->speed == 0 mps] -> landed
    on next [tp->speed > 0 mps] -> landing { points--; } -> imp
```

Error: type int16/[m / s] is not comparable with (uint8 || int8)

alt	^DataStructures.Trackpoint.alt (Member)
crashNotification	^StateMachines.FlightAnalyzer.crashNotification (OutEvent)
id	^DataStructures.Trackpoint.id (Member)
speed	^DataStructures.Trackpoint.speed (Member)
time	^DataStructures.Trackpoint.time (Member)
x	^DataStructures.Trackpoint.x (Member)
y	^DataStructures.Trackpoint.y (Member)

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop

[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight
```

	next(Trackpoint* tp)
beforeFlight	//[Here is a comment on a transition.]
	[tp->alt == 0 m] -> airborne
airborne	[tp->alt == 0 m && tp->speed == 0] -> crashe
	[tp->alt == 0 m && tp->speed > 0 mps] -> lan
	[tp->speed > 200 mps && tp->alt == 0 m] ->
	[tp->speed > 100 mps && tp->speed <= 200 mp tp->alt == 0 m] -> airborne
landing	[tp->speed == 0 mps] -> landed
	[tp->speed > 0 mps] -> landing -> implements Sh
landed	

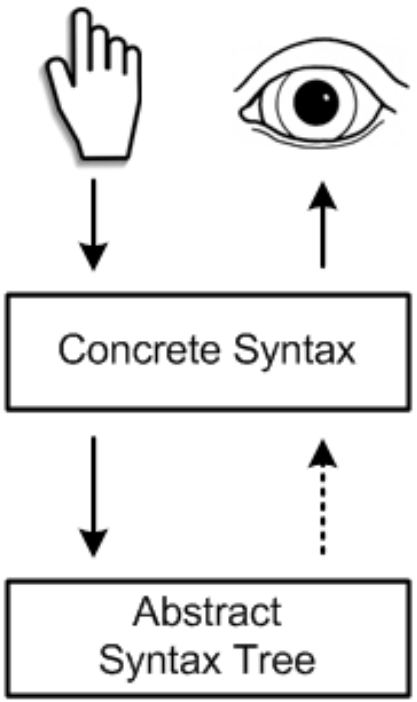
```
beforeFlight
composite state airborne initial = flying {
```

next(Trackpoi
[tp->alt > 0
[onTheGround

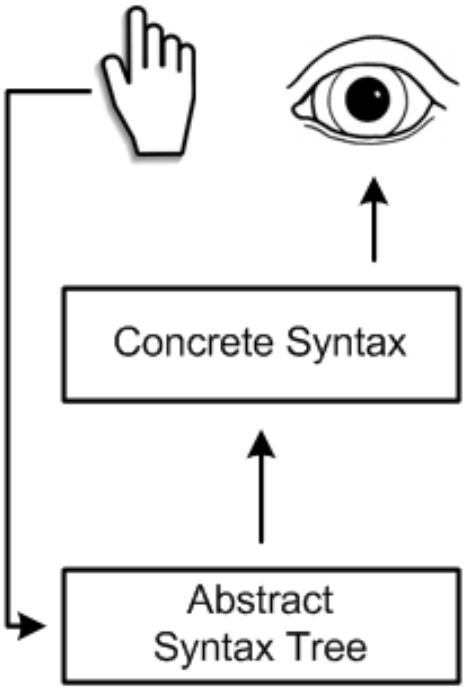
[Projectional Editing]

MPS uses a Projectional Editor

Traditional: Parsing



MPS: Projectional Editing



A Projectional Editor modifies the AST directly.
No grammars or parsers are involved.

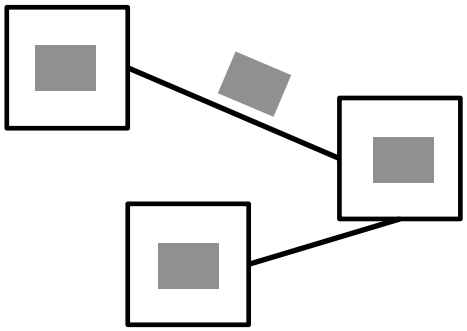
[Projectional Editing]
Advantage: Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical



[Projectional Editing]

Advantage: Syntactic Flexibility / MPS

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)
```

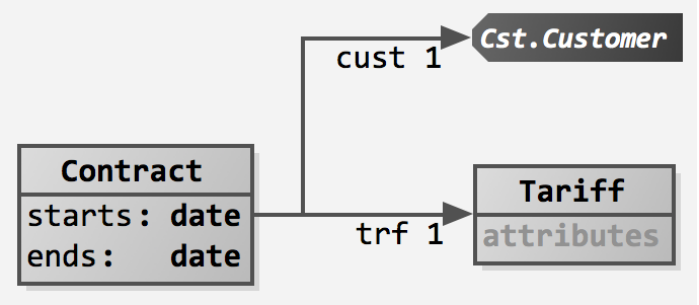
Tables

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

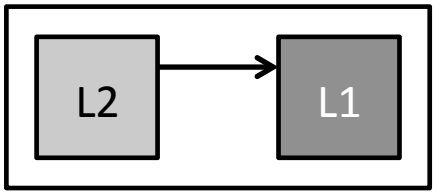
	spd > 0	spd > 100
alt < 0	1	1
alt == 0	10	20
alt > 0	30	40
alt > 100	50	60

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

Graphical



[Projectional Editing]
Advantage: Language Composition



Separate Files

Type System
Transformation
Constraints

In One File

Type System
Transformation
Constraints
Syntax
IDE

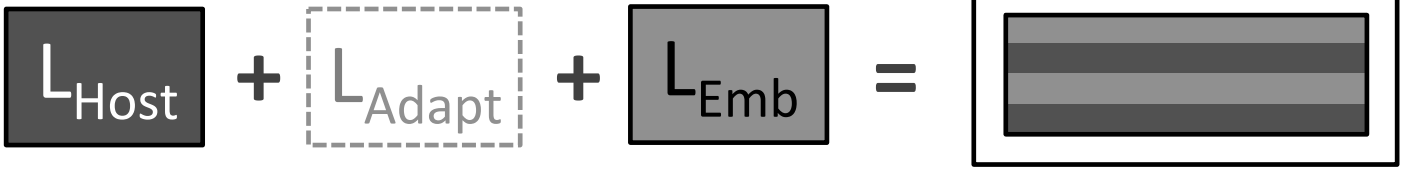
5+ base languages
50+ extensions to C
10+ extensions to requirements lang.

[Projectional Editing]

Advantage: Language Composition

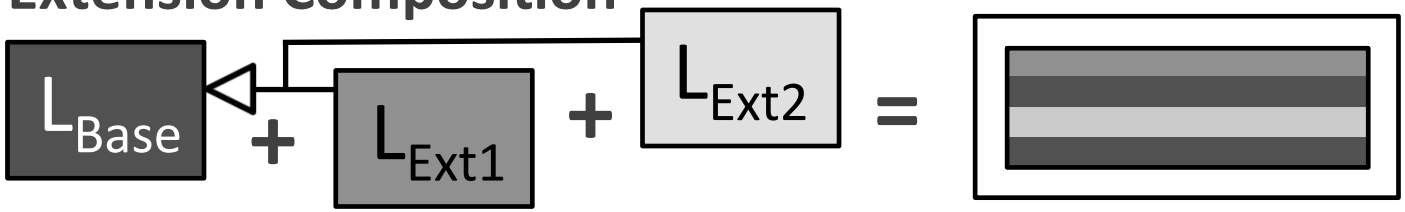
No change to definition of L1 **or** L2 **in order to use them together.**

Embedding



Extension

Extension Composition



[Projectional Editing]

Study Results on Editor Usability

“Initially, the editor feels a bit strange. But once you get used to it, it is better than a traditional text editor.”

[Projectional Editing]
Study Results on Editor Usability

People **prefer MPS** over conventional IDEs

MPS more is **more efficient** than normal IDEs

MPS more is **more productive** than normal IDEs

MPS makes it easier to create **correct programs**

MPS enforces a **structurally correct AST**

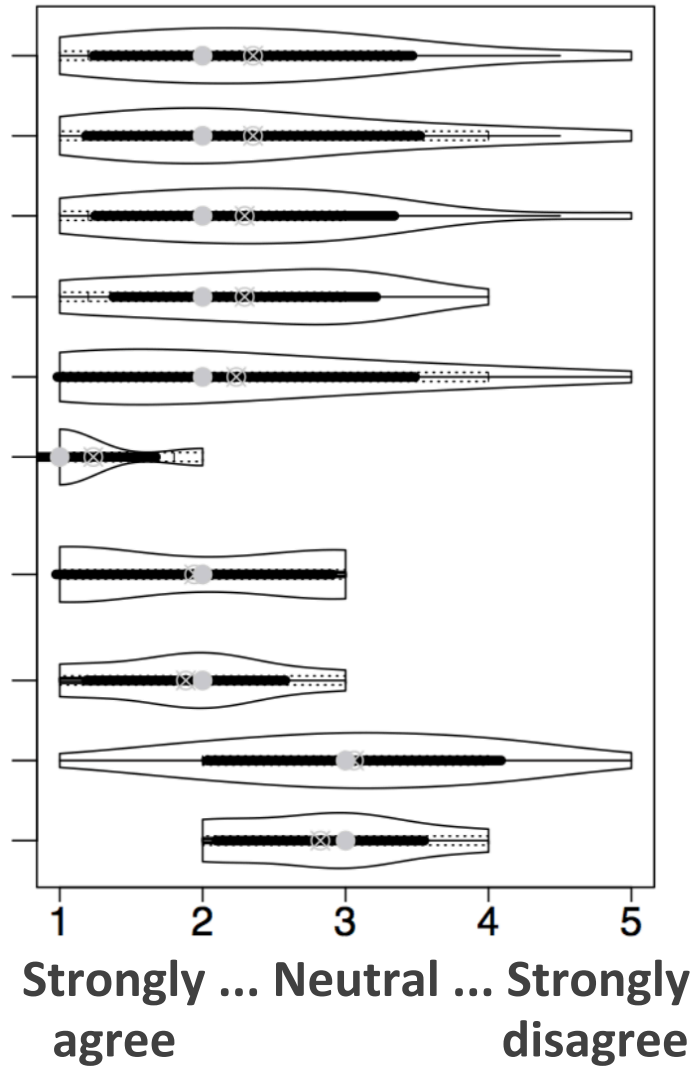
People benefit from **language modularity**

People benefit from the **flexible notations**

People benefit from **advanced navigation support**

The experience with **learning MPS** is mixed.

It takes some time to **get used to MPS**



MP3 users

SIEMENS

fortiss

HUAWEI

BOSCH

Whirlpool CORPORATION

McGill

UNIVERSITY OF WATERLOO

ERICSSON

SIOUX

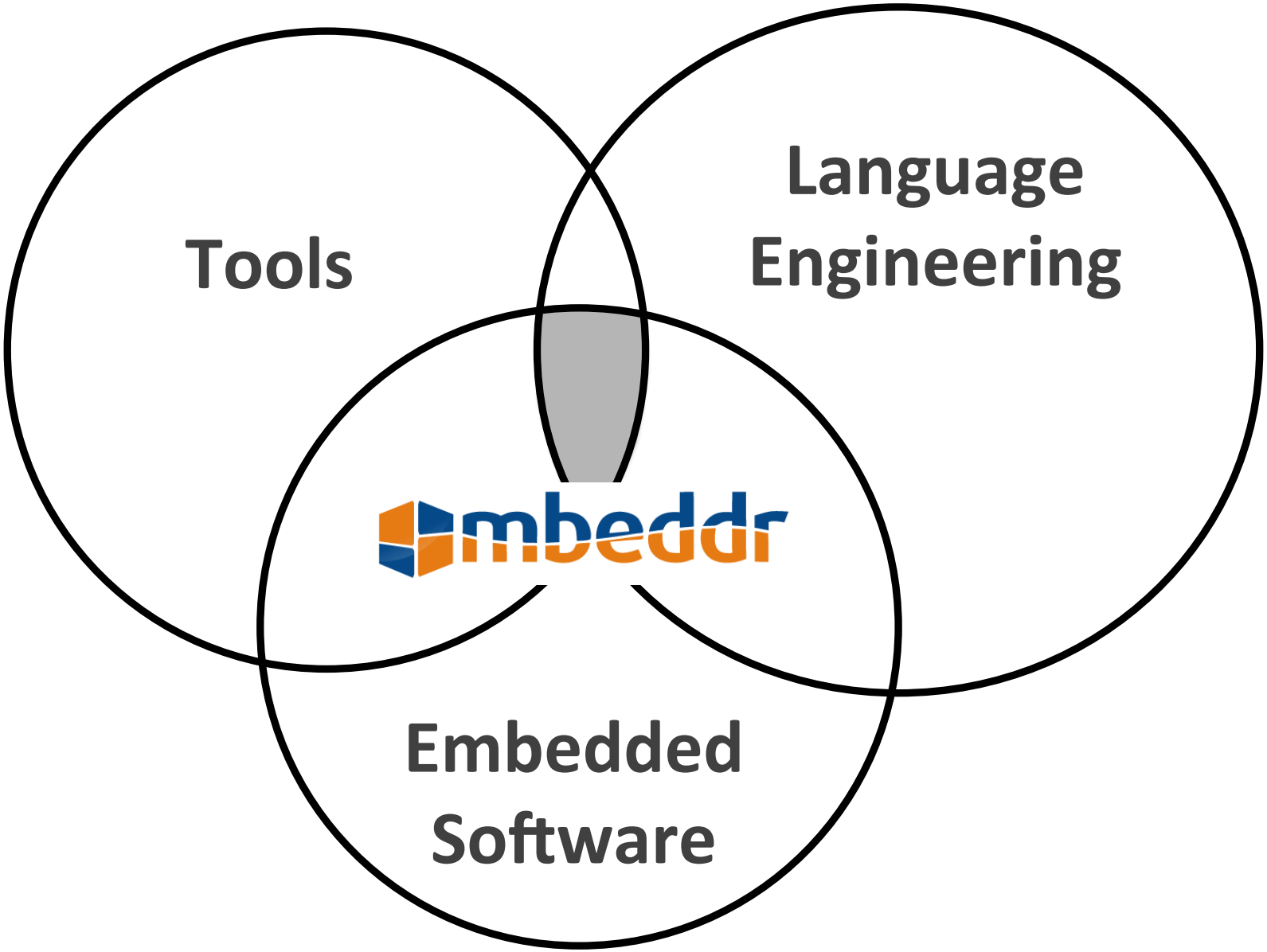
Belastingdienst

itemis

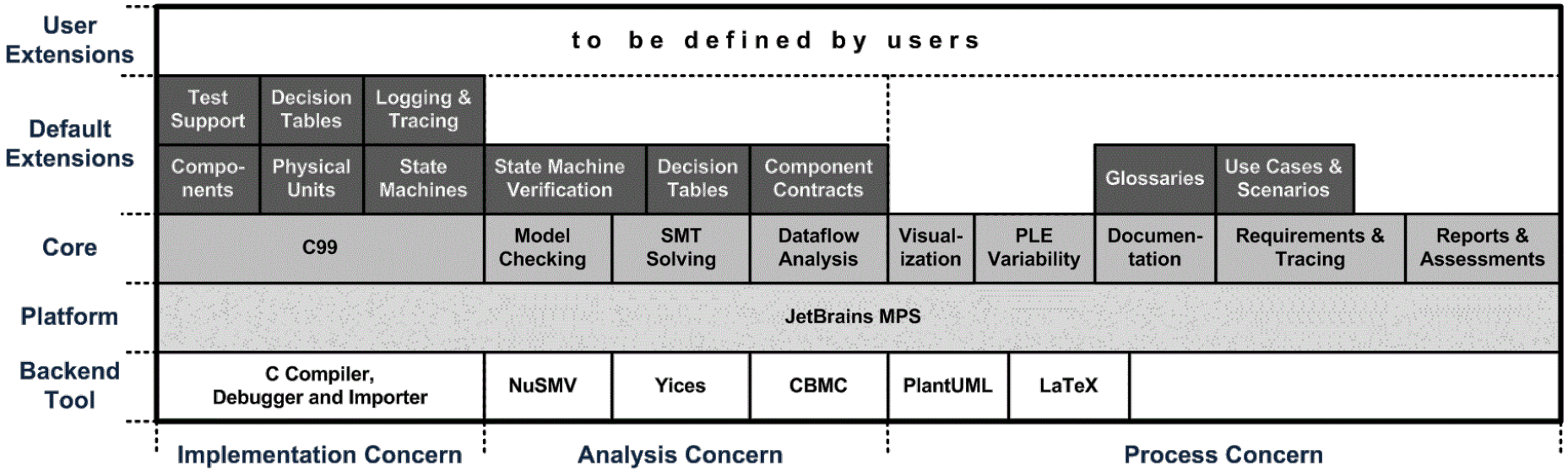
i2S INSURANCE KNOWLEDGE

3

mbeddr


Tools
Language Engineering
mbeddr
Embedded Software

An extensible set of integrated languages for embedded software engineering.



Open Source @ eclipse.org
Eclipse Public License 1.0
http://mbeddr.com

mbeddr developers

itemis — 7 developers, project management

fortiss — 2 developers, verification support

SIEMENS — 1 developer, verification support

SIOUX — 3 developers, C++

---

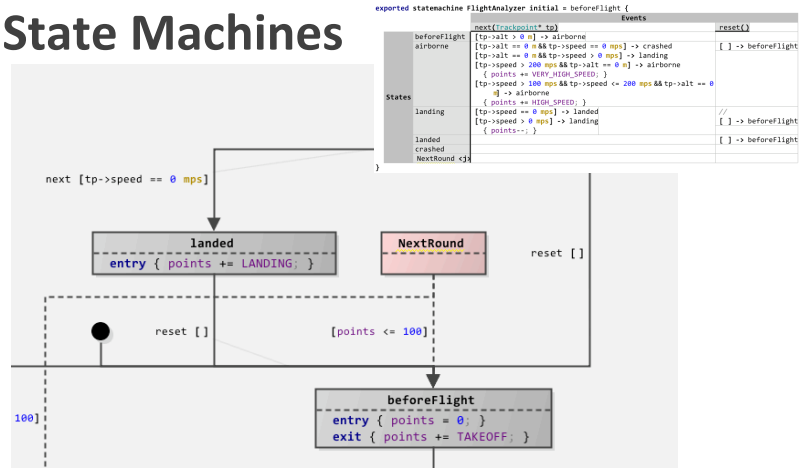
strategic collaboration with JetBRAINS

Some of the C Extensions

Units

```
void calcVerticalSpeed(TWP* prev,TWP* cur) {
  if (prev == null) {
    cur->vSpeed = 0 mps;
  } else {
    int16/m/ dAlt = cur->alt - prev->alt;
    int8/s/ dTime = cur->time - prev->time;
    cur->vSpeed = dTime / dAlt;
  }
} calcVerticalSpeed (function)
```

State Machines



Components

```
exported cs interface TrackpointProcessor {
  Trackpoint* process(Trackpoint* p)
    pre(0) p != null
    pre(1) p->id != 0
    pre(2) p->time != 0 s
    post(3) result->id != 0
}

exported component Nuller extends nothing {
  provides TrackpointProcessor processor
  Trackpoint* processor_process(Trackpoint* p) <= op processor.process {
    p->alt = 42 m;
    return p;
  } runnable processor_process
} component Nuller
```

Math

```
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)
```

An IDE for Requirements

Requirements

6.1 You should land as short as possible
ShortLandingRoll /functional: tags

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Suspendisse potenti. Etiam risus ante, bibendum ut mattis
vel condimentum velit. Quisque venenatis faucibus tellus
Phasellus rhoncus quam eu dui dictum sollicitudi

6.2 Once you land successfully, you get another
FullStop /functional: tags

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Suspendisse potenti. Etiam risus ante, bibendum ut mattis
purposes, this one references **@req**(InFlightPoints)

Rules

```
calculation PointForATrackpoint: This rule computes the points a
                                 It does so by taking into accou
                                 passed as arguments.
  parameters: [ int16 alt: current altitude of the trackpoint ] =
              [ int16 speed: current speed of the trackpoint ]
```

result = (*BASEPOINTS* * 1) *

	alt > 2000	alt > 1000
speed > 180	30	15
speed > 130	10	20

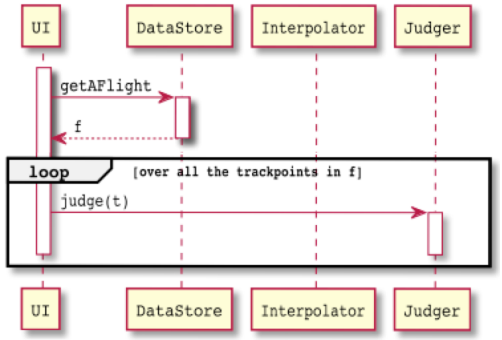
```
  tests: PointForATrackpoint(500, 100) == 0
         PointForATrackpoint(500, 1200) == 0
         PointForATrackpoint(1100, 165) == 200
         PointForATrackpoint(2100, 140) == 100
         PointForATrackpoint(2100, 200) == 300
```

Tracing

```
[#constant TAKEOFF = 100;] -> implements PointsForTakeoff
[#constant HIGH_SPEED = 10;] -> implements FasterThan100
[#constant VERY_HIGH_SPEED = 20;] -> implements FasterThan200
[#constant LANDING = 100;] -> implements FullStop

[checked]
exported statemachine FlightAnalyzer initial = beforeFli
  state beforeFlight {
    entry { points = 0; }
    on next [tp->alt > 0 m] -> airborne
    [exit { points += TAKEOFF; }] -> implements PointsForTakeoff
  } state beforeFlight
```

Visualisations



Great IDE Support

```
DLT645ProtocolImpl ▶ handleDataLength
DLT645ProtocolImpl ▷ DLT645RxFrameParser

Events
dataReceived(uint8 data)

States
idle            [data == DLT645_FRONT_LEADING_BYTE] -> awaking
                  {  }
                Error: type boolean is not a subtype of uint8
awaking         [data + DLT645_FRAME_STARTFLAG] -> receivingFrame
                  {
                    send startFrame(idx, data);
                    idx++;
                  }
receivingFrame  [idx < length - 1 && idx != DLT645_HEADER_DATALENG
                    -> receivingFrame
                  {
                    send frameByte(idx, data);
                    idx++;
                  }
                [idx == DLT645_HEADER_DATALENGTH_OFFSET] -> recei
                  {
                    send dataLength(idx, data);
                    idx++;
                    length = DLT645_FR                + DLT645_TRAILE
                        data;
                  }
                [idx == lengt

DLT645_FRAME_ENDFLAG                ^DLT645ProtocolDefinitions.DLT645_FRAME_ENDFLAG (GlobalConstantDeclaration)
DLT645_FRAME_STARTFLAG              ^DLT645ProtocolDefinitions.DLT645_FRAME_STARTFLAG (GlobalConstantDeclaration)
DLT645_FRONT_LEADING_BYTE           ^DLT645ProtocolDefinitions.DLT645_FRONT_LEADING_BYTE (GlobalConstantDeclaration)
DLT645_FRONT_LEADING_BYTE_COUNT     ^DLT645ProtocolDefinitions.DLT645_FRONT_LEADING_BYTE_COUNT (GlobalConstantDeclaration)
```

```
UnitDeclarations
FractionUnits ▶ nounit -> %

UnitDeclarations                                                    constrai
model   com.itemis.embedded.platform.implementation.intf.units      imports

section FractionUnits {
  exported unit % := <no spec> for Fraction
  exported unit ‰ := <no spec> for Fraction
  exported unit ppm := <no spec> for Fraction

  exported conversion nounit -> % {
   val as <no type> -> val * 100
  }
  exported conversion % -> nounit {
   val as <no type> -> val / 100
  }

  exported conversion nounit -> ‰ {
   val as <no type> -> val * 1000
  }
```

```
DCFilterTestHarness
DCFilterTestRunnerImpl ▶ dcFilterTestRunner_dispose

  } runnable dcFilterTestRunner_dispose
} component DCFilterTestRunnerImpl

exported composite component DCFilterTestHarnessImpl {
  provides IDCFilterTestRunner dcFilterTestRunner

  internal instances {

    exported cs interface IGraphPlotter {
      handle newCurve(Color color, string const label)
      void addPoint(handle hCurve, double x, double y)
      void plot(string const title, handle hCurve)
      void plotAll(string const title)
      void clear(handle hCurve)
      void clearAll()
      void deleteCurve(handle hCurve)
      void dispose()
    }

    IGraphPlotter
    graphPlotterInst: GraphPlotterImpl
    IMetrologyRawSignalSimulator
    dcFilterTestRunner
    dcFilterTestRunnerInst: DCFilterTestRunnerImpl
    metrologyRawSignalSimulatorInst: MetrologyRawSignalSimulatorImpl
    IMetrologyRawSignalData

  }
}
```

Diagram Palette

Event Log 2: Inspector

An IDE for Documentation

5.4 Stripping and Reintroducing Units

Let us assume we have an existing (legacy or external) function that does not know about physial units and you cannot or do not want to use generic units. An example is anExistingFunction:

```
int16 anExistingFunction(int16 x) {
  return x + 10;
} anExistingFunction (function)
```

Code from anExistingFunction

To be able to call this function with arguments that have units, we have to strip away the units before we call the function. This can be achieved by selecting the corresponding expression and invoking the **Strip Unit** intention. The type of this stripped expression will be simply the type of the original expression but without units.

```
int16/m/ someFunction(Trackpoint* p1, Trackpoint* p2) {
  int16 newValueWithoutUnit = anExistingFunction(stripunit(p1.alt));
  return newValueWithoutUnit m;
} someFunction (function)
```

How about... Overhead

CAT I: No runtime footprint

Units, Traces, Variability, any kind of metadata

CAT II: No overhead compared to C

Tests, Logging, State Machines (-> Switch/Case)

CAT III: Some overhead relative to C

Components have some overhead, depending on the system and the optimization level used during generation

„Efficient enough“: Smart Meter: runs on the target hardware selected without the use of mbeddr in terms of memory use and performance/timing (4096 Hz).

mbeddr

How is it different from...

Model-Driven-*

Fully open and extensible

Multiple paradigms, not one-size-fits-all

Mix of „Model and Code“

mbeddr

How is it different from...

Macros

More syntactic flexibility

Higher Expressivity (do more than with Macros)

Type Checking

Generally better IDE support

How is it different from...

C++

Requires no C++ Compiler

Components more suitable for Embedded

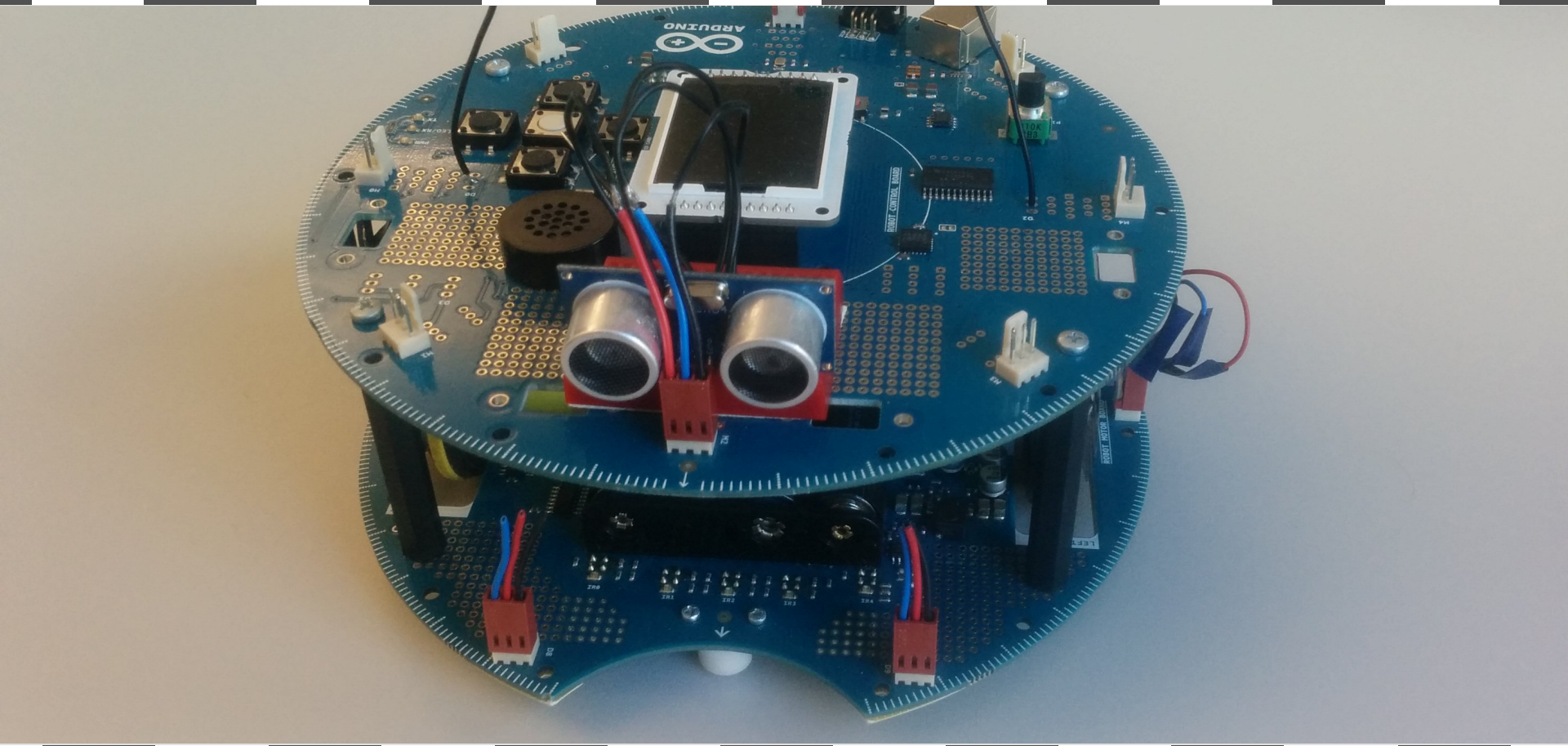
Different Features: units, state machines

TMP: Better IDE support

Better Error Messages

LE better done in LWB

DEMO


ARDUINO
ROBOT CONTROL BOARD
ROBOT MOTOR BOARD

4

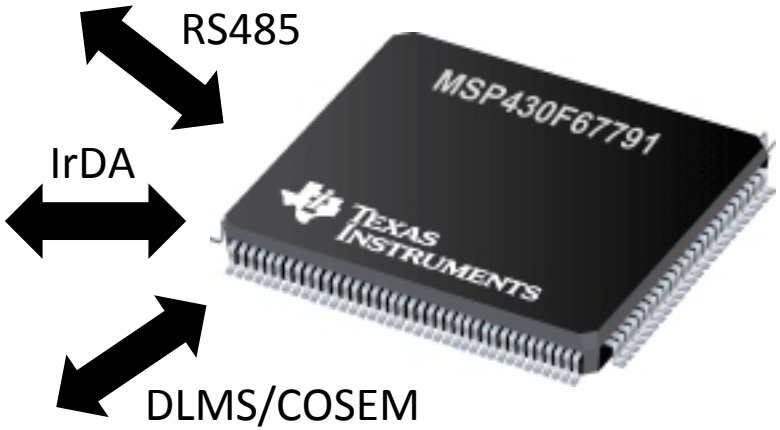
Smart Meter

What is a SM?

Measures Voltage and Current
Computes Derived Values
Shows Data on LCD Display
Communicates through Networks

Precision is critical for Certification.
Evolvability is critical for it to be a viable business.

Hardware Architecture



MQTT

UART

Application Logic

MSP430 F67791

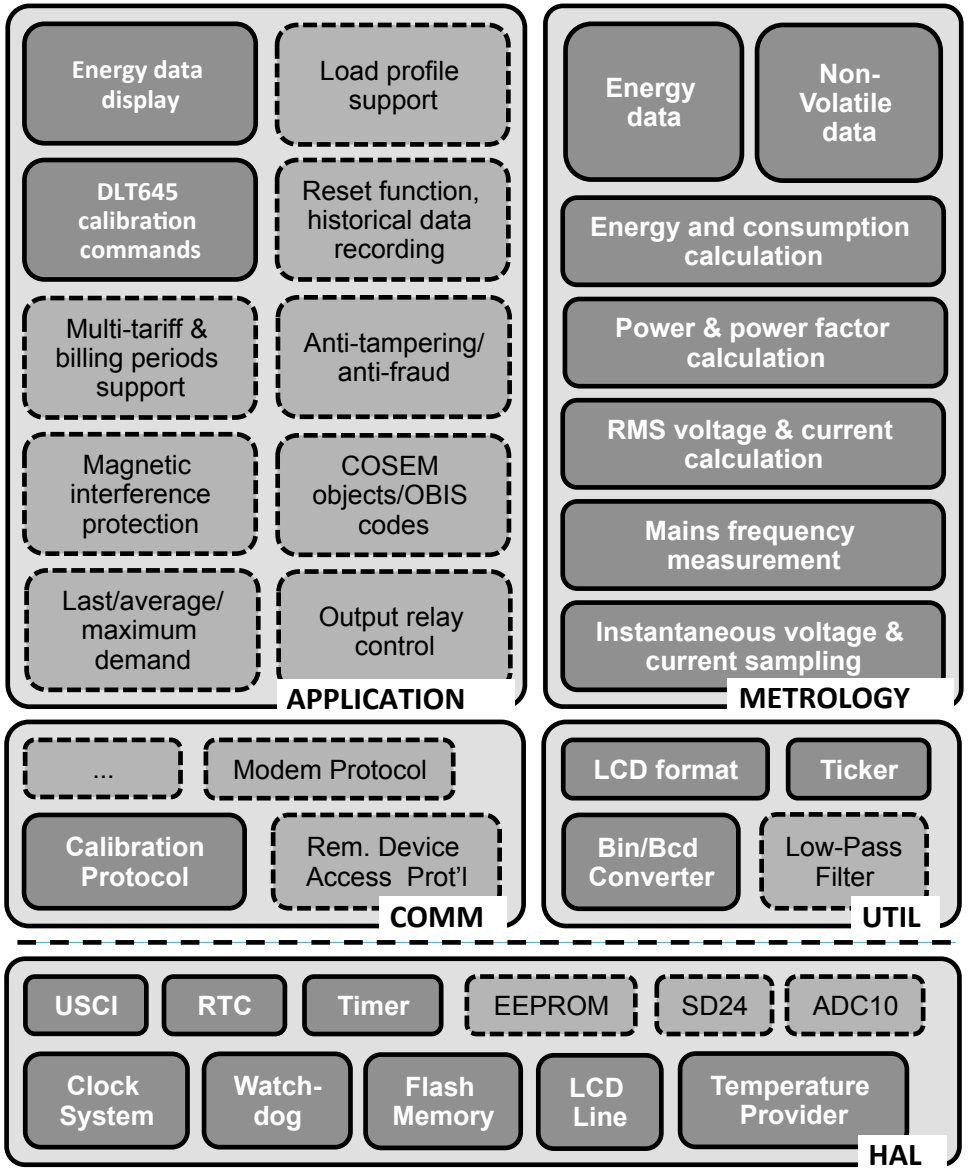
25 MHz
256K Flash ROM
32K RAM

Metrology

MSP430 F6736

25 MHz
128K Flash ROM
8K RAM

Software Architecture



No RTOS

Interrupt-Driven

One-Threaded Programming

Required Precision leads to 4096 Hz Sampling Rate

Interrupt-Triggered:

Measurement

Foreground Tasks:

App Logic, RTC

Example Smart Meter Code

From the processor vendor. But: no tests, bad structure, buggy, not all features.

Hence:

Phase 1 Reimplement with mbeddr

Phase 2 Two Processors,
Communication between the two processors,
Improved comms infrastructure (multiplexing,
two comm stacksRS485 and IrDa)
an I2C Bus driver
an EEPROM controller
a subset of the required DLMS/COSEM messages
additional application functionality (historical data rec, reset)

Size of the System

Criterion	Common	Metro	App	Total
# of Files	134	101	105	340
Total LOC	8,209	10,447	10,908	29,564
Code LOC	4,397	5,900	5,510	15,807
Comment LOC	950	2,402	2,620	5,972
Whitespace LOC	2,852	2,145	2,778	7,775

Common code runs on both processors, **Metro** runs on the metrology processor and **App** runs on the application / communication processor.

Use of Extensions

Category	Concept	Count
Chunks (≈ Files)	Implementation Modules	382
	Other (Req, Units, etc.)	46
C Constructs	Functions	310
	Structs / Members	144 / 270
	Enums / Literals	150 / 1,211
	Global Variables	334
	Constants	8,500
Components	Interfaces / Operations	80 / 197
	Atomic Components	140
	Ports / Runnables	630 / 640
	Parameters / Values	84 / 324
	Composite Components	27
	Component Config Code	1,222
State Machines	Machines	2
	States/Transitions/Actions	14 / 17 / 23
Physical Units	Unit Declarations	122
	Conversion Rules	181
	Types / Literals with Units	593 / 1,294

Category	Concept	Count
Product Line Variability	Feature Models / Features	4 / 18
	Configuration Models	10
	Presence Condition	117
Custom Extensions	Register Definition	387
	Interrupt Definitions	21
	Protocol Messages	42
Statements	Statements total	16,840
	Statements in components	6,812
	Statements in test cases	5,802
	Statements in functions	3,636
Testing	Test Cases / Suites	107 / 35
	Test-Specific Components	56
	Stub / Mock Components	9 / 8
	`assert` Statements	2,408

All mbeddr C extensions used a lot.
Some extensions built specifically for SM.

Complexity

The developers **naturally think** in terms of extensions, and suggested additional ones during the project.

mbeddr **components help structure** the overall architecture and enable reuse and configurability.

mbeddr extensions facilitate strong **static checking**, improve **readability** and help avoid **low-level mistakes**.

Testing

mbeddr components are instrumental in **improving testability** through clear interfaces and small units, leading to 80% test coverage for core components.

The custom extensions and the components **facilitate hardware-independent testing**, continuous integration and automated dry runs of the certification process.

The modularization facilitated by components **helps track down** problems during commissioning.

Overhead

The **memory** requirements of SMT are **low enough** for it to run on the intended hardware, with room for growth.

Componentization enables **deployment of only the functionality necessary** for a variant, conserving resources.

The **performance** overhead is low enough to achieve the required **4,096 Hz sample rate** on the given hardware.

Effort

Development Tasks	Effort	% Total
Implementation	200 PD	66%
Reimplementation	145 PD	48%
Additional Functionality	55 PD	18%
Tests, Simulators	48 PD	16%
Integration & Commissioning	38 PD	13%
Custom Language Extensions	14 PD	5%

Effort

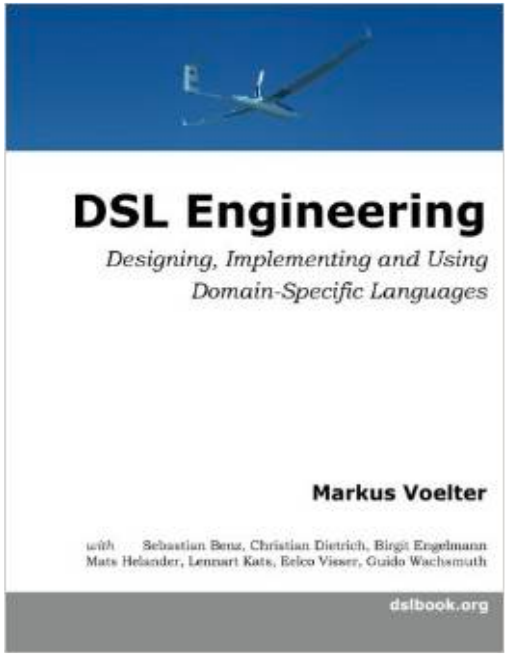
The **effort** for the additional functionality, integration and commissioning is **lower than what is common** in embedded software.

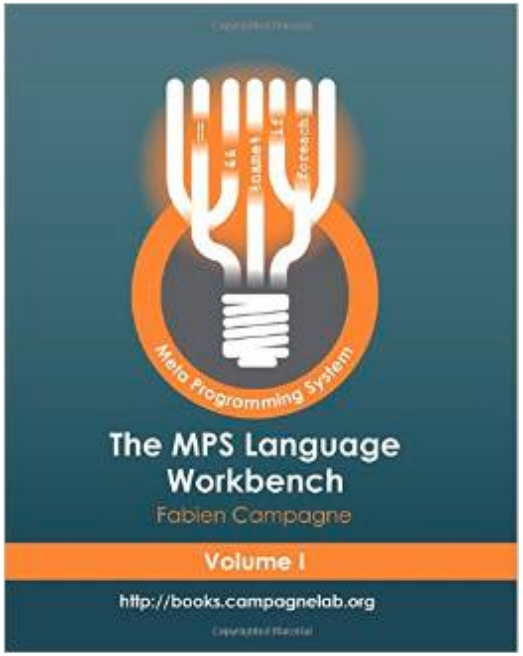
The effort for **building the extensions** is low enough for it to be absorbed in a real project.

Overall, using **mbeddr does not lead to significant effort overrun**, while resulting in better-structured software.

[open

source





mbeddr

MPS Meta Programming System

voelter { ingenieurbüro für softwaretechnologie // itemis

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