Language Workbenches and DSLs for Embedded S/W
This Talk

0  Languages, Models, Programs
1  Language Workbenches
2  MPS
3  mbeddr
4  Smart Meter
Languages, Models, Programs
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;
}
beforeFlight

crashed

reset

next [alt > 0]

next [alt == 0 && speed == 0]

flying

next [alt == 0 && speed > 0]

landing

landed

next [alt == 0 && speed == 0]

airborne
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 {
  }
}
```c
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]
[Model or Code: Notation!]

Textual

Graphical

Tabular

Mathematical

\[ \sum \]
One important aspect of languages.
[Abstraction]

The other important aspect.
[Abstraction]

The other important aspect.
[Relationships]

representation of

model of

part of

view of
model of Abstraction
part of Hierarchy, Partitioning
view of Concern, Aspect, Viewpoint
representation of Presentation, Notation, Form
1

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)

language ::= 
  schema
  + editors
  + generators
Language Workbench
(Martin Fowler)
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 and modeling
The State of the Art in Language Workbenches

Conclusions from the Language Workbench Challenge

Sebastian Erdweg¹, Tijs van der Storm²,³, Markus Völter⁴, Meinte Boersma⁵, Remi Bosman⁶, William R. Cook⁷, Albert Gerritsen⁶, Angelo Hulshout⁸, Steven Kelly⁹, Alex Loh¹⁰, Gabriël Konat¹⁰, Pedro J. Molina¹¹, Martin Palatnik⁶, Eugen Schindler⁶, Klemens Schindler⁶, Riccardo Solmi¹², Vlad Vergu¹⁰, Eelco Visser¹⁰, Kevin van der Vlist¹³, Guido Wachsmuth¹⁰, and Jimi van der Woning¹³

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 Icine, Sevilla, Spain ¹² Independent, Bologna, Italy ¹³ Universiteit van Amsterdam
<table>
<thead>
<tr>
<th></th>
<th>Enso</th>
<th>Más</th>
<th>MetaEdit+</th>
<th>MPS</th>
<th>Onion</th>
<th>Rascal</th>
<th>Spoofax</th>
<th>SugarJ</th>
<th>Whole</th>
<th>Xtext</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Textual</td>
<td>●</td>
<td>●</td>
<td>⬜</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Graphical</td>
<td>●</td>
<td>●</td>
<td>⬜</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Tabular</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Symbols</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2Text</td>
<td>●</td>
<td>●</td>
<td>⬜</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Model2Model</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Concrete syntax</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structural</td>
<td>●</td>
<td>●</td>
<td>⬜</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Naming</td>
<td>⬜</td>
<td>●</td>
<td>⬜</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Types</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Programmatic</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

*Legend:*
- ● Mandatory
- ⬜ Optional
- ⬞ Or

*Diagram Description:*
- Language workbench
  - Notation
    - Textual
    - Graphical
    - Tabular
  - Semantics
    - Model2Text
    - Model2Model
  - Validation
    - Structural
    - Naming
    - Types
    - Programmatic
  - Composability
    - Editor
      - Concrete syntax
      - Editing mode
    - Semantic services
      - Syntactic 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

*Table:*

**Notation**
- Enso: Complete
- Más: Complete
- MetaEdit+: Complete
- MPS: Complete
- Onion: Complete
- Rascal: Complete
- Spoofax: Complete
- SugarJ: Complete
- Whole: Complete
- Xtext: Complete

**Semantics**
- Model2Text: Complete
- Model2Model: Complete
- Concrete syntax: Complete

**Validation**
- Structural: Complete
- Naming: Optional
- Types: Complete
- Programmatic: Complete
<table>
<thead>
<tr>
<th></th>
<th>Ensō</th>
<th>Más</th>
<th>MetaEdit+</th>
<th>MPS</th>
<th>Onion</th>
<th>Rascal</th>
<th>Spoofax</th>
<th>SugarJ</th>
<th>Whole</th>
<th>Xtext</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Textual</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Graphical</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Tabular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Symbols</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2Text</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Model2Model</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Concrete syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Interpretative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structural</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Naming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Programmatic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Notation</td>
<td>Textual</td>
<td>Mås</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-----</td>
<td>----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Graphical</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabular</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbols</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantics</td>
<td>Model2Text</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model2Model</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concrete syntax</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interpretative</td>
<td>✔️</td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>Structural</td>
<td></td>
<td></td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naming</td>
<td></td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Types</td>
<td></td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Programmatic</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enso</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Notation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Textual</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Graphical</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Tabular</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Symbols</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2Text</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Model2Model</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Concrete syntax</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structural</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Naming</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Types</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Programmatic</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Enso</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>------</td>
<td>-----------</td>
<td>------</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Notation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Textual</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
</tr>
<tr>
<td>Graphical</td>
<td>●●</td>
<td>●</td>
<td>●●</td>
<td>●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tabular</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbols</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2Text</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2Model</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concrete syntax</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structural</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naming</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Types</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmatic</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td>●●</td>
<td>●●</td>
<td>●●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- ●● Mandatory
- ● Optional
- ○ Or
<table>
<thead>
<tr>
<th></th>
<th>Enso</th>
<th>Más</th>
<th>MetaEdit+</th>
<th>MPS</th>
<th>Onion</th>
<th>Rascal</th>
<th>Spoofax</th>
<th>SugarJ</th>
<th>Whole</th>
<th>Xtext</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL prog. debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Composability</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax/views</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Editor services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Editing mode</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free-form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projectional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enso</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL testing</td>
<td></td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSL debugging</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>DSL prog. debugging</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td><strong>Composability</strong></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax/views</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Validation</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Semantics</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Editor services</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td><strong>Editing mode</strong></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Free-form</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Projectional</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Syntactic services</td>
<td>Enso</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Highlighting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Folding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto formatting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference resolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refactoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error marking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick fixes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Live translation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic services</td>
<td>Ensō</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoonax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Highlighting</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Outline</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Folding</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Syntactic completion</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Diff</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Auto formatting</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Semantic services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference resolution</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Semantic completion</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Refactoring</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Error marking</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Quick fixes</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin tracking</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Live translation</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic services</td>
<td>Ensō</td>
<td>Más</td>
<td>MetaEdit+</td>
<td>MPS</td>
<td>Onion</td>
<td>Rascal</td>
<td>Spoofax</td>
<td>SugarJ</td>
<td>Whole</td>
<td>Xtext</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Highlighting</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Folding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto formatting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference resolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refactoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error marking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick fixes</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Live translation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- ●: Mandatory
- ○: Optional
- ▲: Or
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
MPS uses a Projectional Editor

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

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

Mathematical

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

Tables

```c
int16 decide(int8 spd, int8 alt) {
    return spd > 0 ? spd > 100 ? 1 : 0 : 10;
} decide (function)
```

Graphical

- Customer diagram with relations:
  - Contract
  - Starts: date
  - Ends: date
  - Tariff attributes
  - Cust.Customer
  - cust 1
  - trf 1
[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.
Advantage: Language Composition

Embedding

\[ L_{\text{Host}} + L_{\text{Adapt}} + L_{\text{Emb}} = \]

Extension

\[ L_{\text{Base}} + L_{\text{Ext}} = \]

Extension Composition

\[ L_{\text{Base}} + L_{\text{Ext1}} + L_{\text{Ext2}} = \]

No change to definition of \( L_1 \) or \( L_2 \) in order to use them together.
Initially, the editor feels a bit strange. But once you get used to it, it is better than a traditional text editor.
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
mbeddr
An extensible set of integrated languages for embedded software engineering.
Open Source @ eclipse.org
Eclipse Public License 1.0
http://mbeddr.com
developers

itemis  7 developers, project management

fortiss  2 developers, verification support

SIEMENS  1 developer, verification support

SIoux  3 developers, C++

strategic collaboration with
Some of the C Extensions

Units

```c
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)
```

Components

```c
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
    }
}
```

State Machines

```c
int32 averageIntArray(int32[] arr, int32 size) {
    return \sum_{i=0}^{\text{size}} \text{arr}[i] / \text{size};
} averageIntArray (function)
```

Math

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

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 account the passed as arguments.

parameters: [ int16 alt: current altitude of the trackpoint ] =

result = (BASEPOINTS * 1) *
speed > 180 30 15
speed > 130 10 20

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

Tracing

constant TAKEOFF = 100; >> implements PointForATakeoff
constant HIGH_SPEED = 10; >> implements FasterThan100
constant VERY_HIGH_SPEED = 20; >> implements FasterThan200
constant LANDING = 100; >> implements FullStop

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

Visualisations
Great IDE Support
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 physical units and you cannot or do not want to use generic units. An example is `anExistingFunction`:

```c
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.

```c
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).
How is it different from...

Model-Driven-*

Fully open and extensible
Multiple paradigms, not one-size-fits-all
Mix of „Model and Code“
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
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

**Application Logic**

**MSP430 F67791**
- 25 MHz
- 256K Flash ROM
- 32K RAM

**Metrology**

**MSP430 F6736**
- 25 MHz
- 128K Flash ROM
- 8K RAM

Connections:
- RS485
- IrDA
- DLMS/COSEM
- MQTT
- UART
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 stacks RS485 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

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

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

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

### Count

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

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.
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.
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.
<table>
<thead>
<tr>
<th>Development Tasks</th>
<th>Effort</th>
<th>% Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>200 PD</td>
<td>66%</td>
</tr>
<tr>
<td>Reimplementation</td>
<td>145 PD</td>
<td>48%</td>
</tr>
<tr>
<td>Additional Functionality</td>
<td>55 PD</td>
<td>18%</td>
</tr>
<tr>
<td>Tests, Simulators</td>
<td>48 PD</td>
<td>16%</td>
</tr>
<tr>
<td>Integration &amp; Commissioning</td>
<td>38 PD</td>
<td>13%</td>
</tr>
<tr>
<td>Custom Language Extensions</td>
<td>14 PD</td>
<td>5%</td>
</tr>
</tbody>
</table>
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
Thank you!