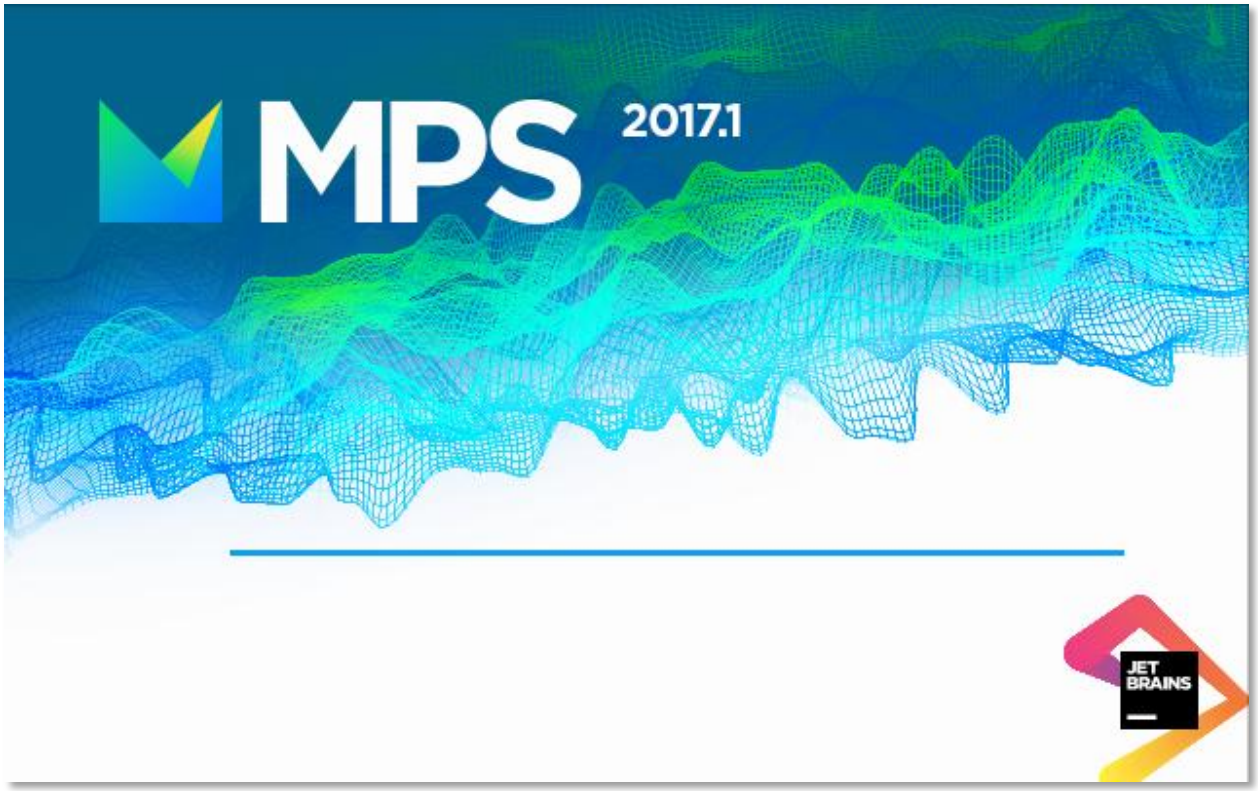
Lessons Learned from Developing mbeddr

A Case Study in Language Engineering with MPS

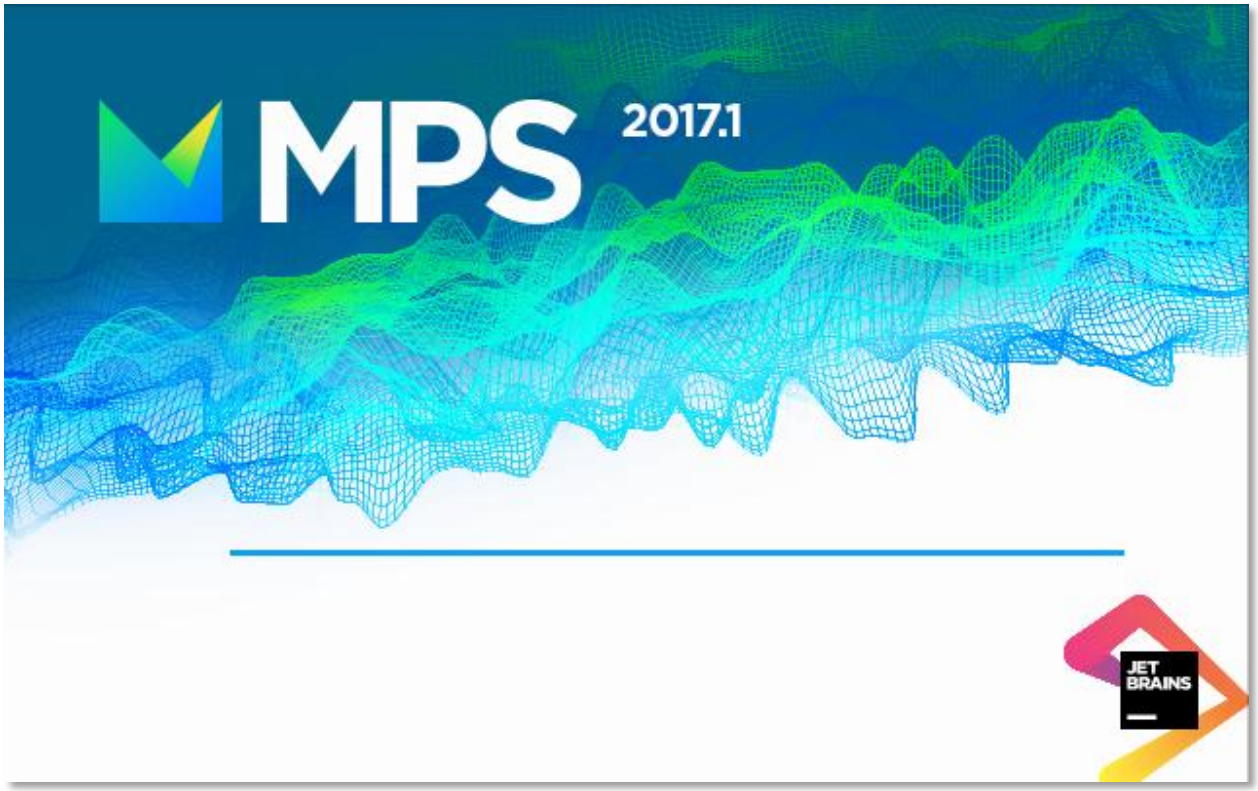
Markus Völter voelter@acm.org
Bernd Kolb bernd.kolb@itemis.de
Tamas Szábó tamas.szabo@itemis.com
Daniel Ratiu daniel.ratiu@siemens.com
Arie van Deursen Arie.vanDeursen@tudelft.nl

MPS

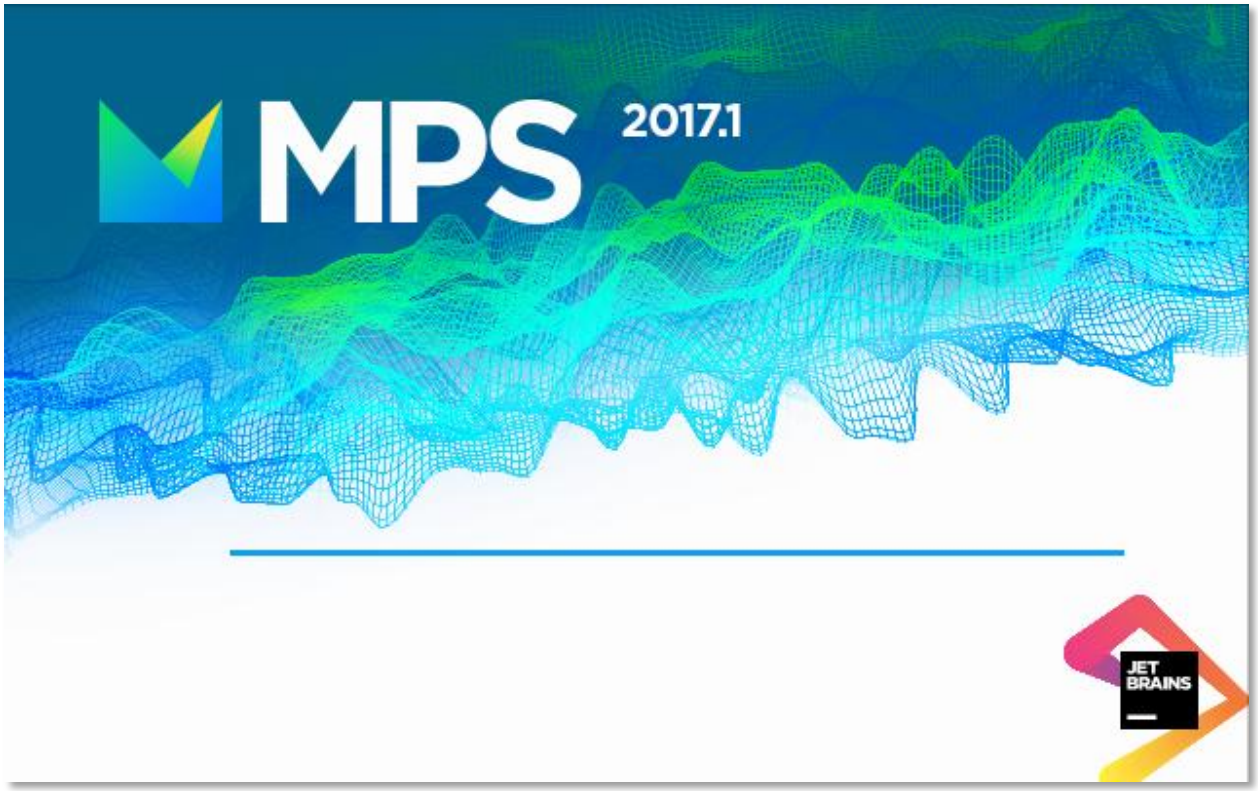


A Language Workbench –

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



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

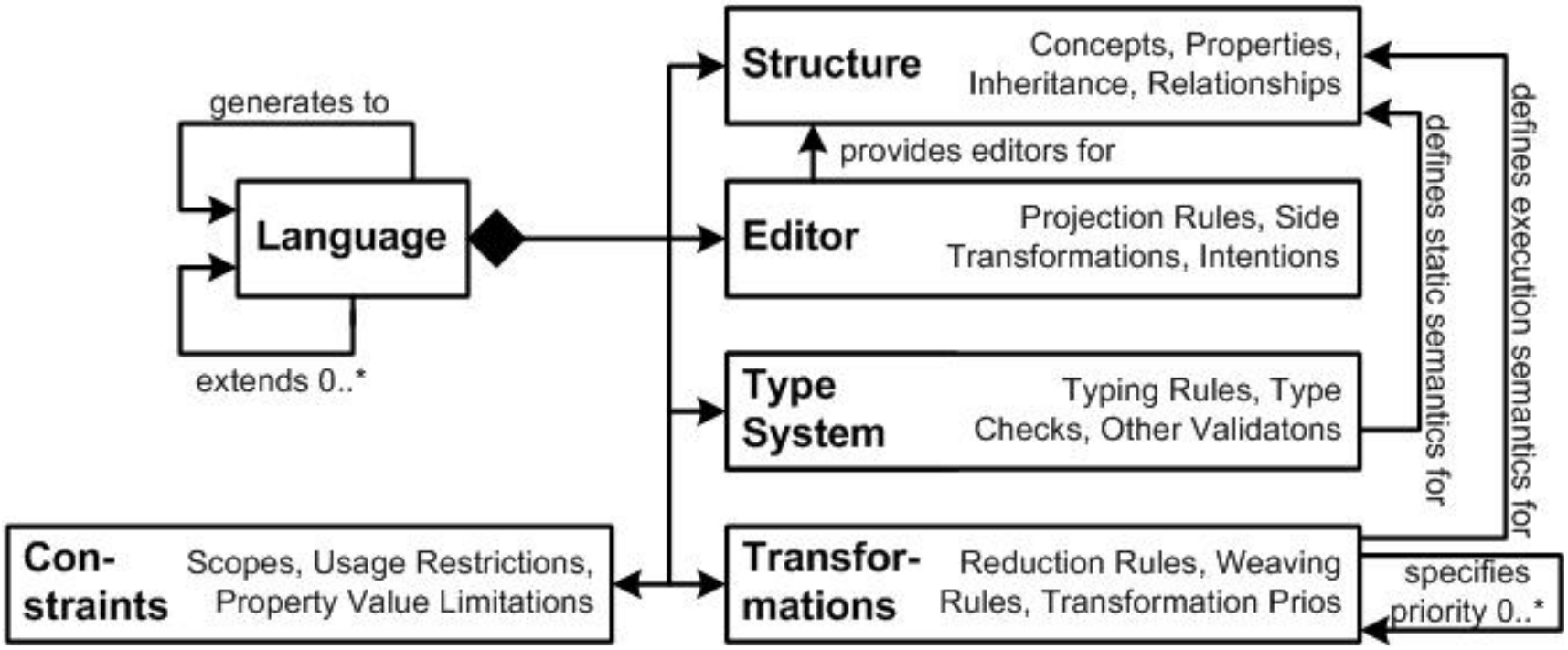


V 2017.2 is current

V 2017.3 released later in 2017

[Language Workbench]

Comprehensive Support for many aspects of Language Definition.



+ Refactorings, Find Usages, Syntax Coloring, Debugging, Language Evolution and Models Migration, Configuration Management, ...

Mercedes-Benz

SIEMENS

fortiss

BOSCH

McGill

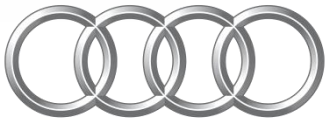
UNIVERSITY OF
WATERLOO

ERICSSON

Belastingdienst

itemis

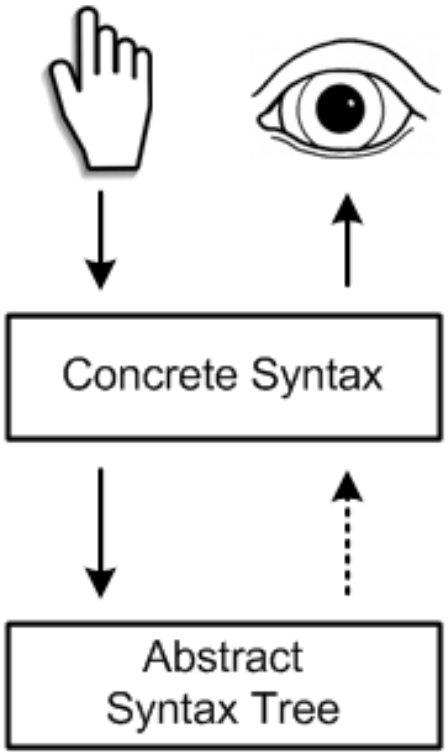
INSURANCE KNOWLEDGE



Audi

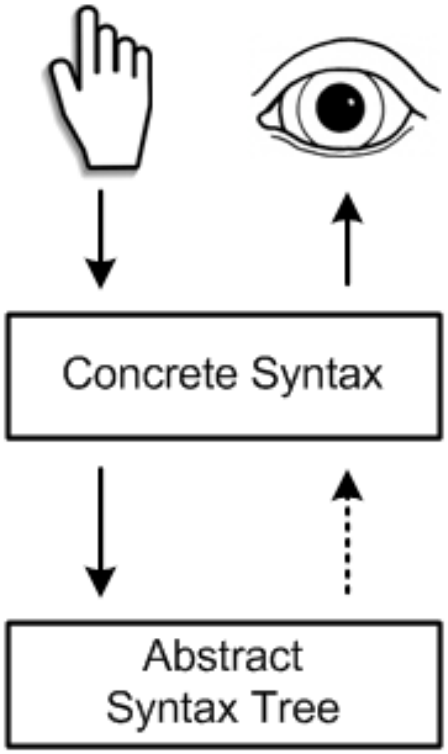
[Projectional Editing]

Parsing

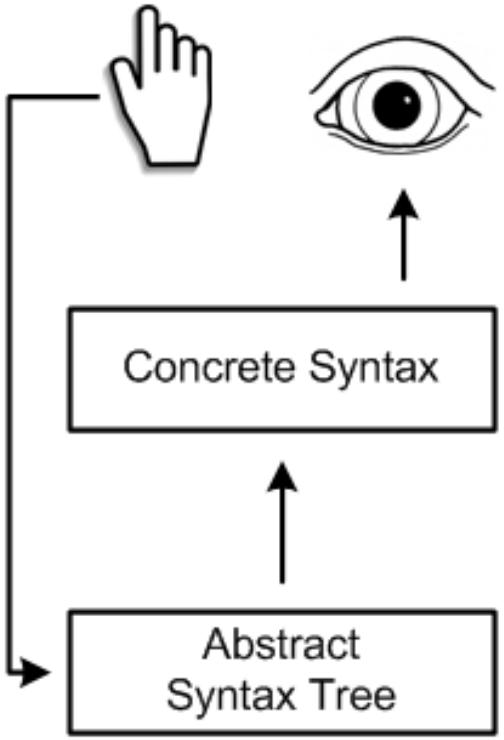


[Projectional Editing]

Parsing



Projectional Editing



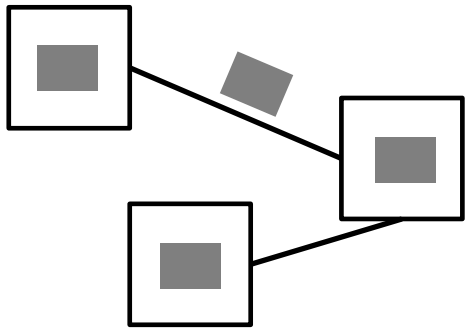
[Projectional Editing]
Syntactic Flexibility

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) {
  return (-b + √(b² - Σ_{i=1}^{4} a * c)) / (2 * a);
} midnight2 (function)
```

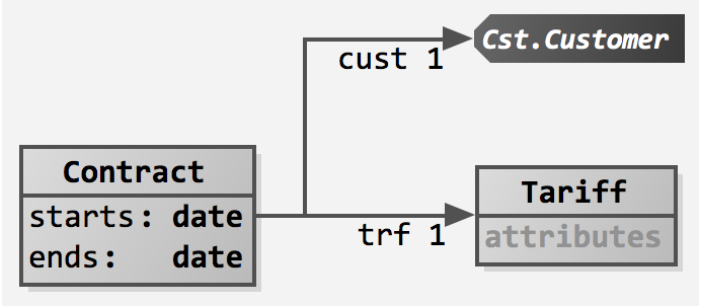
Tables

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

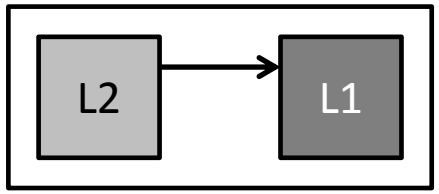
	spd > 0	spd > 100
alt < 0	1	1
alt == 0	10	20
alt > 0	30	40
alt > 100	50	60

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

Graphical



[Projectional Editing]
Language Composition



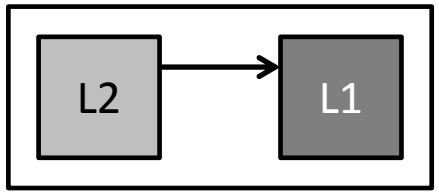
Separate Files

Type System
Transformation
Constraints

In One File

Type System
Transformation
Constraints
Syntax
IDE

[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.


mbeddr

Origins

„Language Workbenches for Embedded Systems“

Research project publicly funded between 2011 – 2013

Goal: show that it is cost effective to build domain specific languages and tools even for small companies or teams

… by using language workbenches

itemis

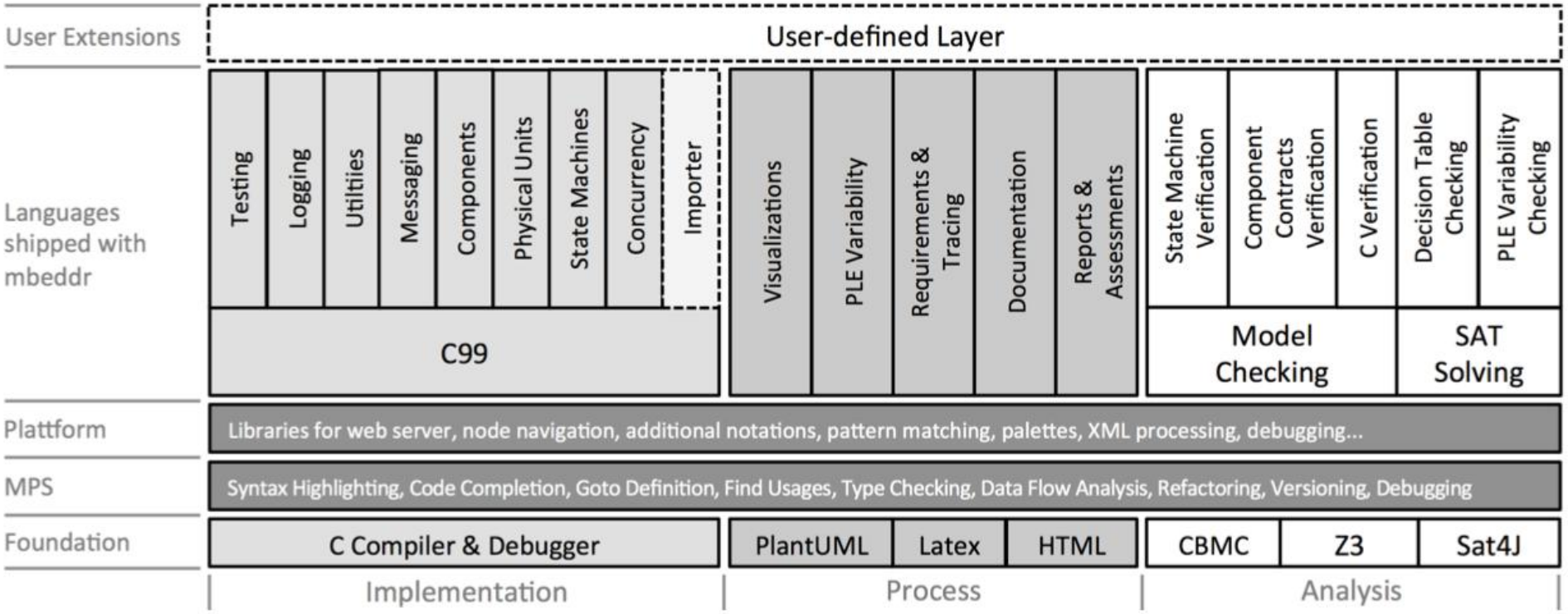
fortiss

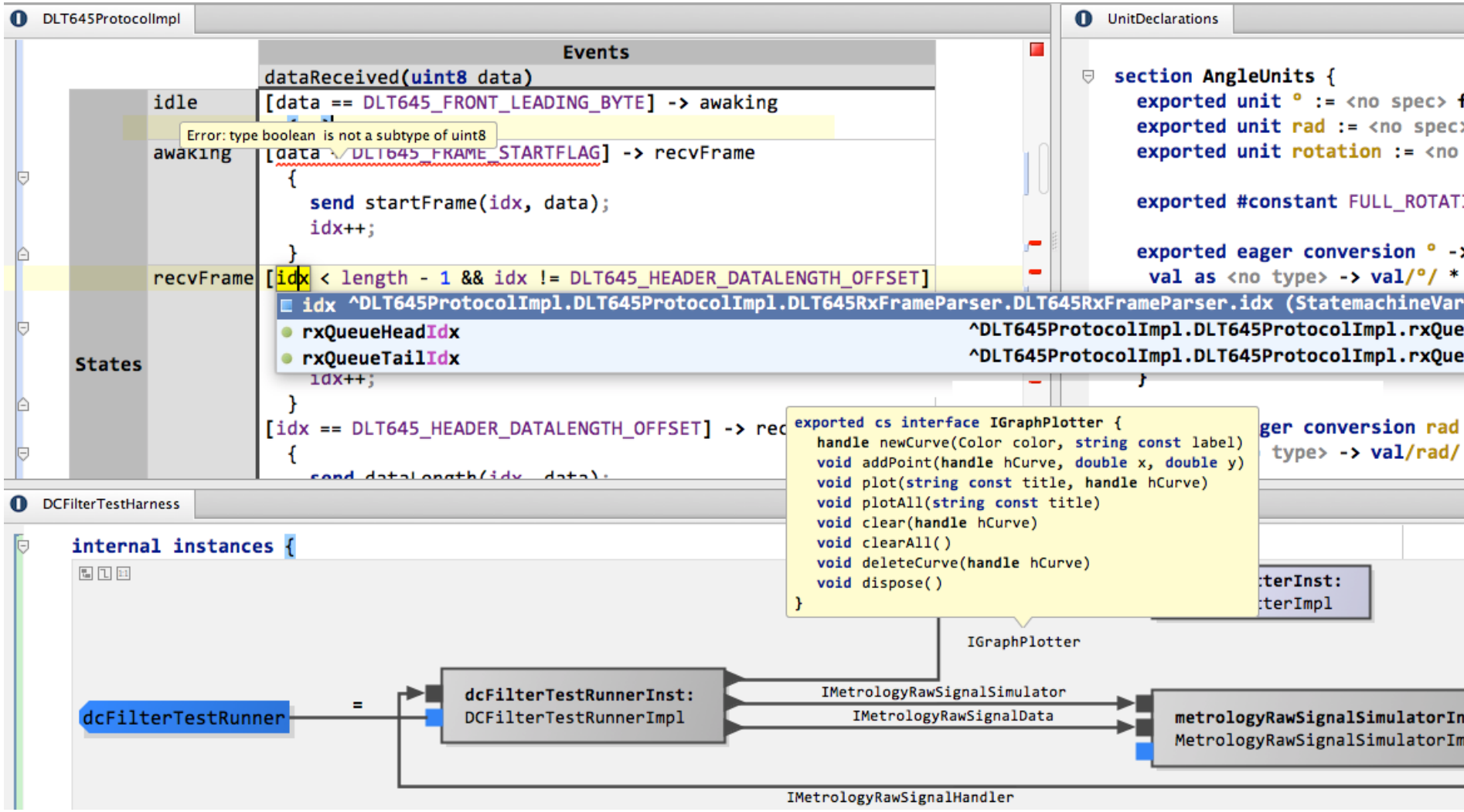
BMW Car IT

Bundesministerium
für Bildung
und Forschung

mbeddr

An extensible set of integrated languages for embedded software engineering.




mbeddr
DLT645ProtocolImpl
Events
dataReceived(uint8 data)
idle
[data == DLT645_FRONT_LEADING_BYTE] -> awaking
Error: type boolean is not a subtype of uint8
awaking
[data == DLT645_FRAME_STARTFLAG] -> recvFrame
{
send startFrame(idx, data);
idx++;
}
recvFrame
[idx < length - 1 && idx != DLT645_HEADER_DATALENGTH_OFFSET]
idx ^DLT645ProtocolImpl.DLT645ProtocolImpl.DLT645RxFrameParser.DLT645RxFrameParser.idx (StatemachineVar
rxQueueHeadIdx ^DLT645ProtocolImpl.DLT645ProtocolImpl.rxQue
rxQueueTailIdx ^DLT645ProtocolImpl.DLT645ProtocolImpl.rxQue
States
idx++;
}
[idx == DLT645_HEADER_DATALENGTH_OFFSET] -> rec
{
UnitDeclarations
section AngleUnits {
exported unit ° := <no spec> f
exported unit rad := <no spec>
exported unit rotation := <no
exported #constant FULL_ROTAT
exported eager conversion ° ->
val as <no type> -> val/°/ *
}
ger conversion rad
type> -> val/rad/
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
DCFilterTestHarness
internal instances {
terInst:
terImpl
dcFilterTestRunner
=
dcFilterTestRunnerInst:
DCFilterTestRunnerImpl
IMetrologyRawSignalSimulator
IMetrologyRawSignalData
metrologyRawSignalSimulatorIn
MetrologyRawSignalSimulatorIm
IMetrologyRawSignalHandler

Components (mbeddr)

```
// ADC is the analog-digital converter
interface IADC {
  int16 read(uint8 addr)
}
```

```
component ADCDriver {
  provides IADC adc
  int16 adc_read(uint8 addr) <= op adc.read {
    int16 val = // low level code to read from addr
    return val;
} }
```

```
component CurrentMeasurer {
  requires IADC currentADC
  internal void measureCurrent() {
    int16 current = currentADC.read(CURR_SENSOR_ADDR);
    // do something with the measured current value
} }
```

State Machines (mbeddr)

```
statemachine FrameParser initial = idle {
  var uint8 idx = 0
  in event dataReceived(uint8 data)
  state idle {
    entry { idx = 0; }
    on dataReceived [data == LEADING_BYTE] -> wakeup
  }
  state wakeup {
    on dataReceived [data == START_FLAG]
       -> receivingFrame { idx++; }
  }
  state receivingFrame { .. }
}
```

```
// create and initialize state machine
FrameParser parser;
parser.init;
// trigger dataReceived event for each byte
for (int i=0; i<data_size; i++) {
  parser.trigger(dataReceived|data[i]);
}
```

Testing & State M. (mbeddr)

```
testcase testFrameParser1 {
  FrameParser p;
  assert(0) p.isInState(idle);
  // invalid byte; stay in idle
  parser.trigger(dataReceived|42);
  assert(0) p.isInState(idle);
  // LEADING_BYTE, go to awakening
  parser.trigger(dataReceived|LEADING_BYTE);
  assert(0) p.isInState(awakening);
}

testcase testFrameParser2 { ... }
testcase testFrameParser3 { ... }

int32 main(int32 argc, char* argv) {
  return test[testFrameParser1,
              testFrameParser2,
              testFrameParser3];
}
```

Mocks & Units (mbeddr)

```
mock component USCIReceiveHandlerMock {
  provides ISerialReceiveHandler handler
  Handle* hnd;
  sequence {
    step 0: handler.open { } do { hnd = handle; }
    step 0: handler.dataReceived {
        assert 0: parameter data: data == 1 }
    step 1: handler.dataReceived {
        assert 1: parameter data: data == 2 }
    step 2: handler.dataReceived { .. }
    step 3: handler.dataReceived { .. }
    step 4: handler finsihed { } do { close(hnd); }
} }
```

```
unit V :=      for voltage
unit A :=      for Amps
unit Ω := V·A^-1 for resistance

uint16/Ω/ resistance(uint16/V/ u, uint16/A/[ ] i, uint8 ilen) {
  uint16/A/ avg_i = (sum p = 0 .. ilen : i[p]) / ilen;
  return avg_i / u;
} resistance (function)
```

Error: type uint16 /V^(−1) · A/ is not a subtype of uint16 /Ω/

Product Lines (mbeddr)

```
feature model SMTFeatures
  root opt
    Data_LEDs opt
      DataReadLED
      DataWriteLED [DigitalIOPortPin pin]
    DISPLAY xor
      DISPLAY_V10
      DISPLAY_V22
    WRITABLE_FLASH_MEMORIES
```

```
exported composite component MetrologyPlatformLayer {
  provides IWatchdogTimer watchdogTimer
  ? {DataReadLED && WRITABLE_FLASH_MEMORIES}
  ?provides IDigitalOutputPin pin1
  ? {DataWriteLED}
  ?provides IDigitalOutputPin pin2
```

Registers (smart meter)

```
exported register8 ADC10CTL0 compute as val * 1000

void calculateAndStore( int8 value ) {
   int8 result = // some calculation with value
   ADC10CTL0 = result; // stores result * 1000 in reg.
}
```

Interrupts (smart meter)

```
module USCIProcessor {
  exported interrupt USCI_A1
  exported interrupt RTC

  exported component RTCImpl {
    void interruptHandler() <- interrupt {
      hw->pRTCPS1CTL &= ~RT1PSIFG;
} } }
```

```
instances usciSubsystem {
  instance RTCImpl rtc;
  bind RTC -> rtc.interruptHandler
  connect ... // ports
}
```

Messages (smart meter)

```
// a field representing a timestamp for 10:20:00
uint8[6] f_time = {0x00A,  // field type identifier
                   UNIT_TIME24, // unit used: time
                   3, // 3 payload bytes follow
                   10, 20, 00 // the time itself
                  };

// a field repre
uint8[4] f_value

// a message that uses the two fields
uint8[5] message
```

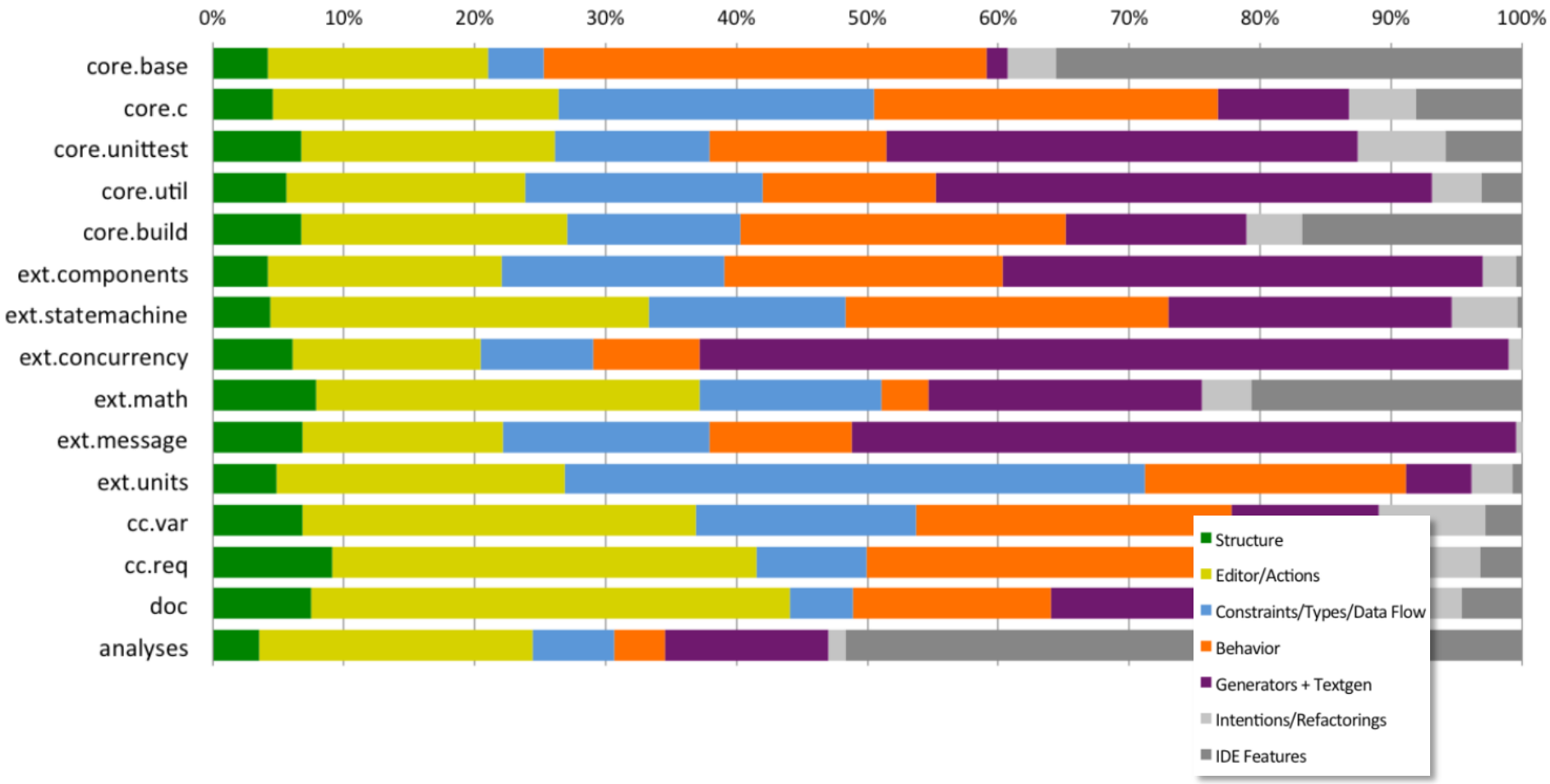
```
message CurrentMeasuredValue:42 {
  int32     timestamp; // time of measurement
  uint16/A/ value;     // measured value in Amps
  uint16    accuracy;  // accuracy in 1/100 %
}
message ... { ... }
...
```

```
atomic component CoreMeasurer {
  field uint16/A/ lastValue = 0;
  message data 42 {:currentTime, &lastValue, 100};
  void measure() {
    lastValue = // perform actual measurement
} }
```

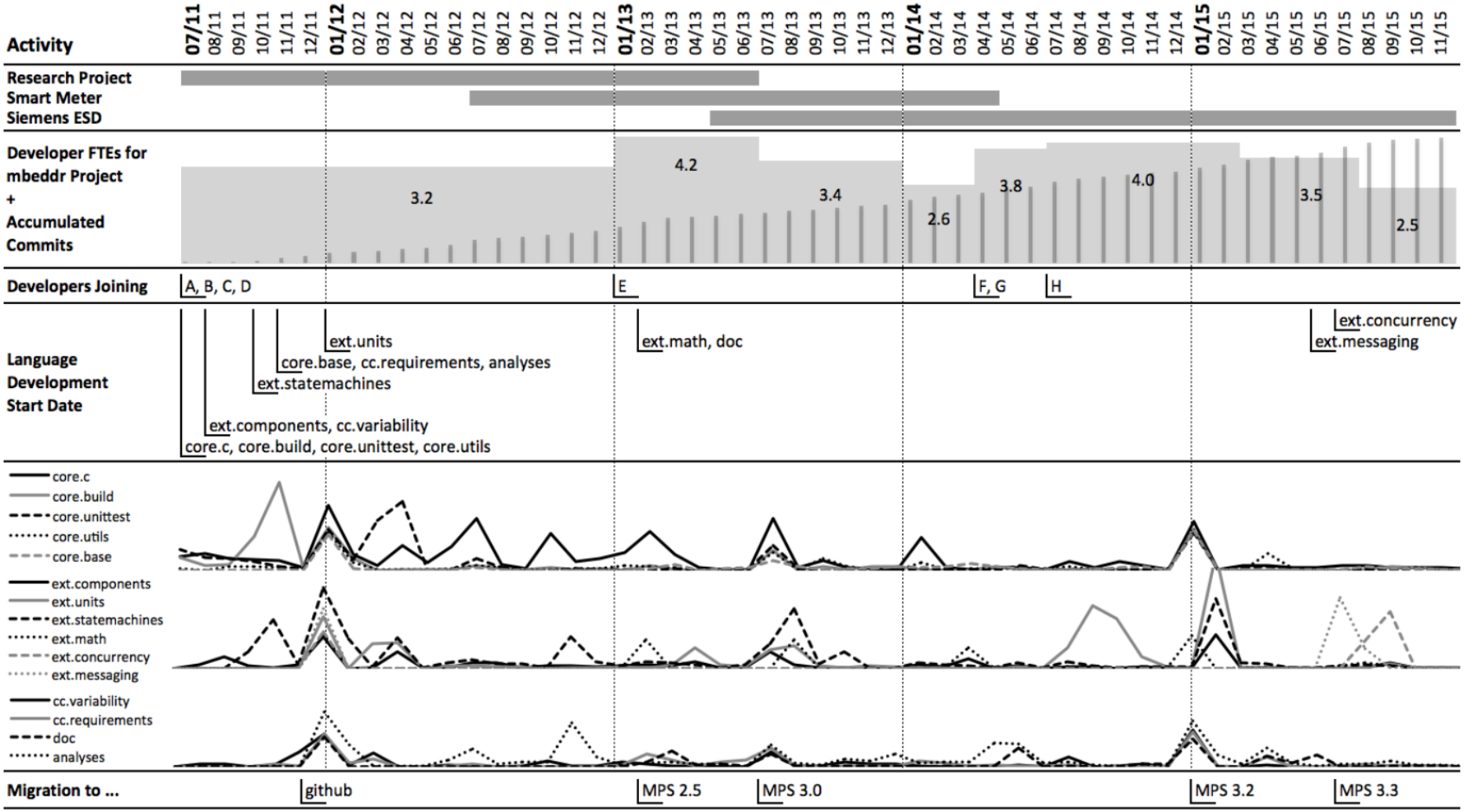
mbeddr Size of the System

Part	Language	#L	#LC	#S	#C	#LOC
core	base	1	-	1	98	6,163
	C	8	-	7	354	20,114
	unittest	1	1	1	26	1,014
	utils	3	3	1	116	6,306
	build	2	-	1	48	2,080
ext	components	9	9	1	160	11,173
	state machines	1	1	1	48	3,194
	concurrency	3	3	0	65	3,078
	math	1	1	0	11	446
	messaging	2	2	0	60	2,151
	units	1	1	0	30	1,884
cc	variability	7	3	1	87	3,638
	reqmts/tracing	9	1	2	171	5,563
doc	doc	10	-	1	153	6,355
analyses	analysis	18	10	18	170	15,235
Total		81	34	38	1,597	88,394

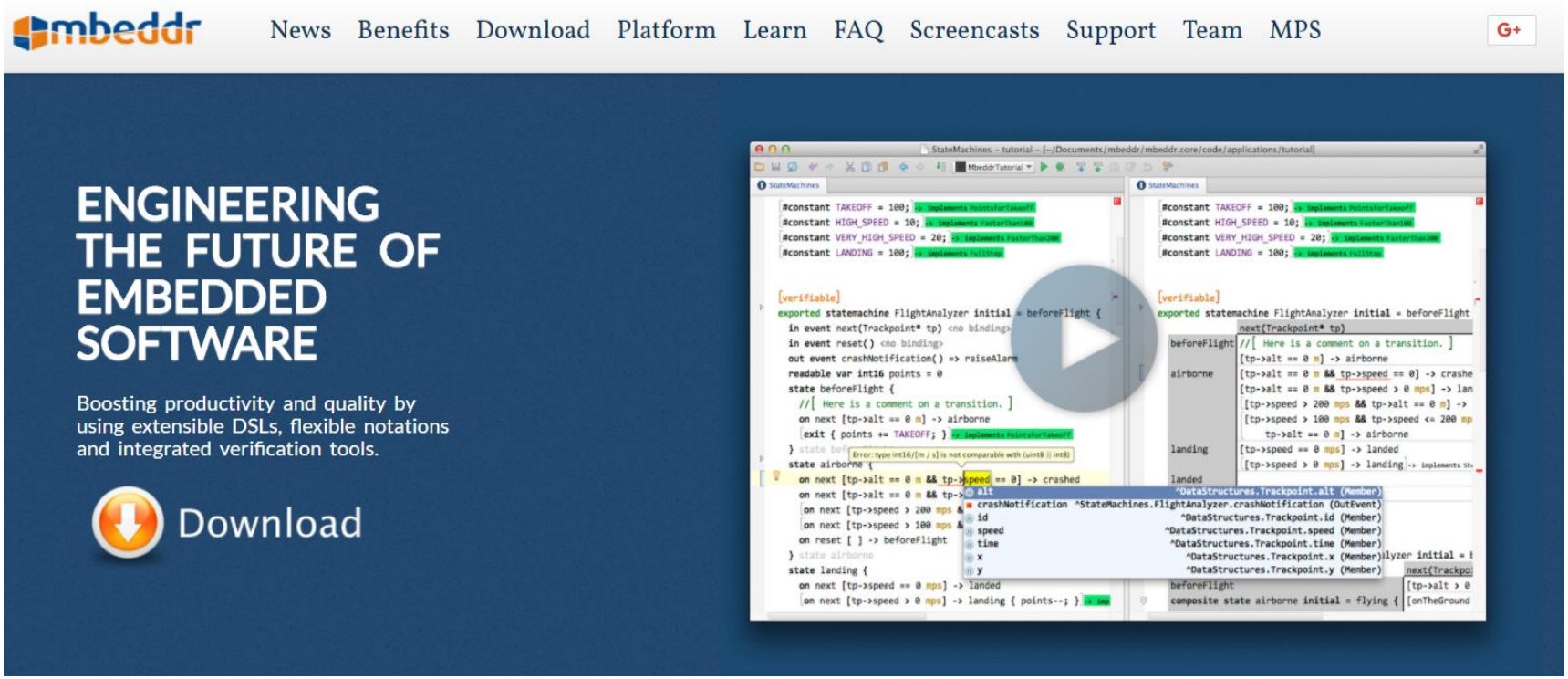
mbeddr Variety of Languages



mbeddr Dev Timeline



mbeddr Homepage



www.mbeddr.com

Open Source, Eclipse Public License

https://github.com/mbeddr/mbeddr.core

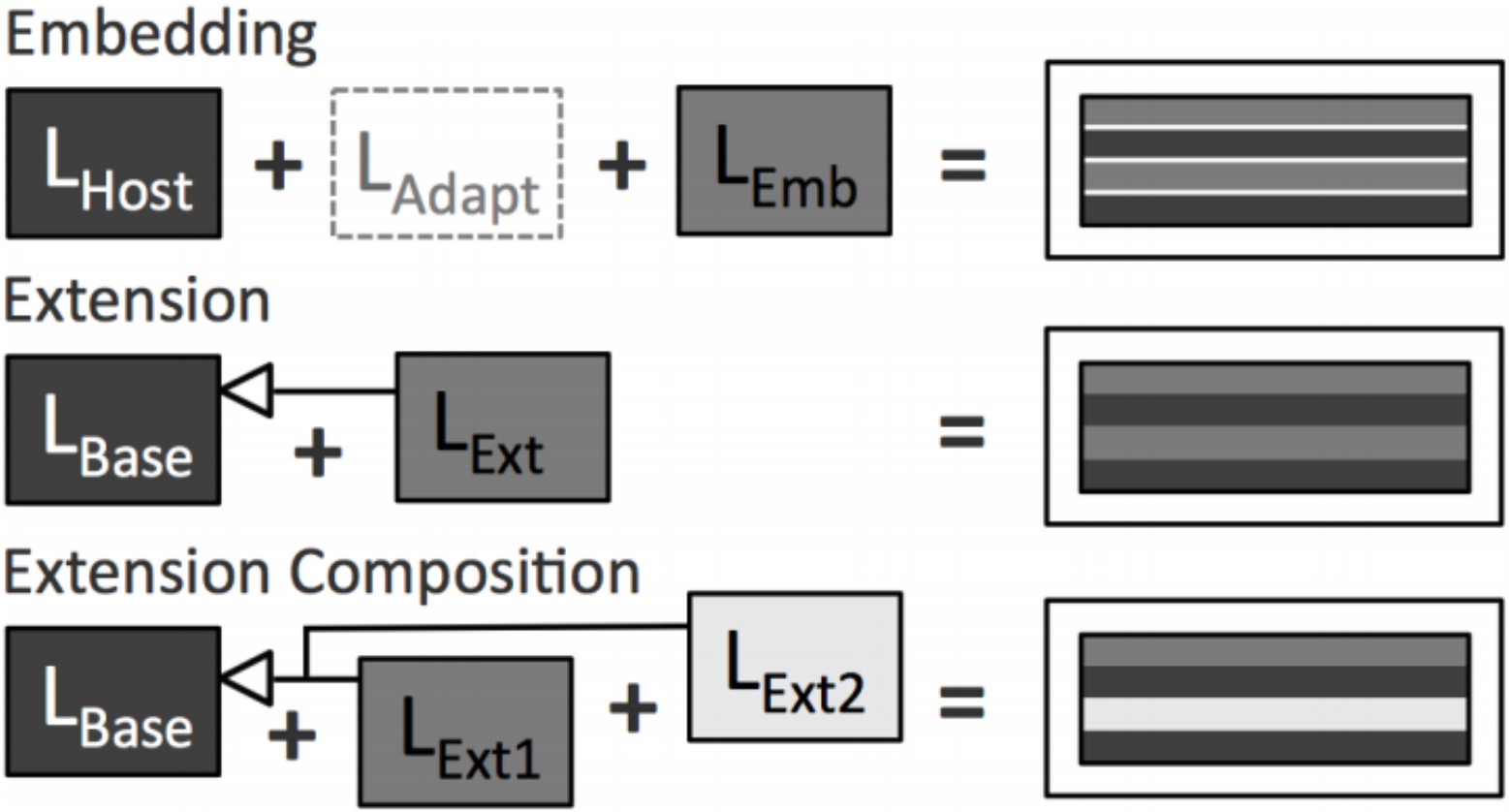
Research Findings

RQ1: Modularity

Is it practically feasible to define a modular set of languages of the size of mbeddr?

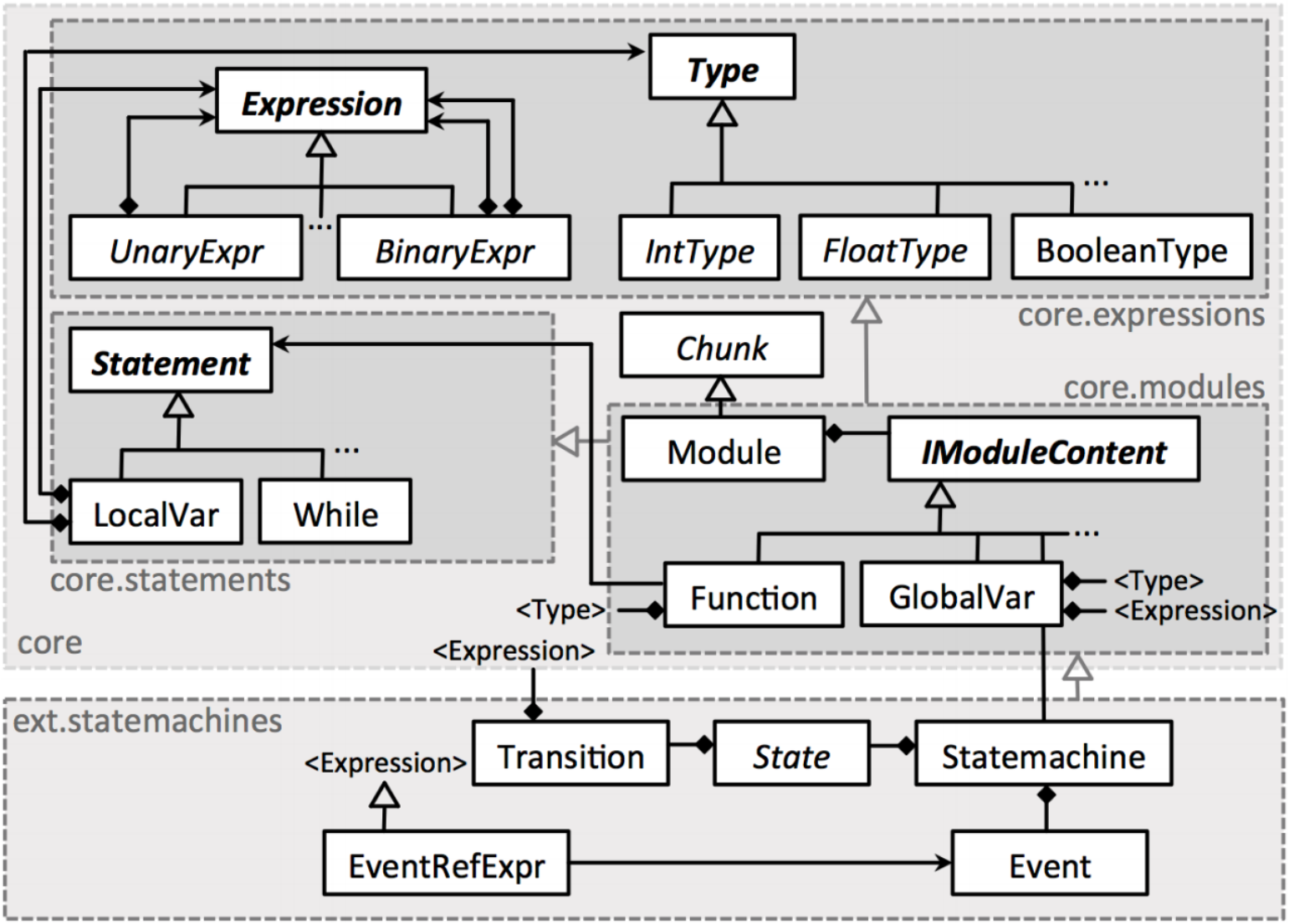
Modularity

Language Composition Mechanisms used



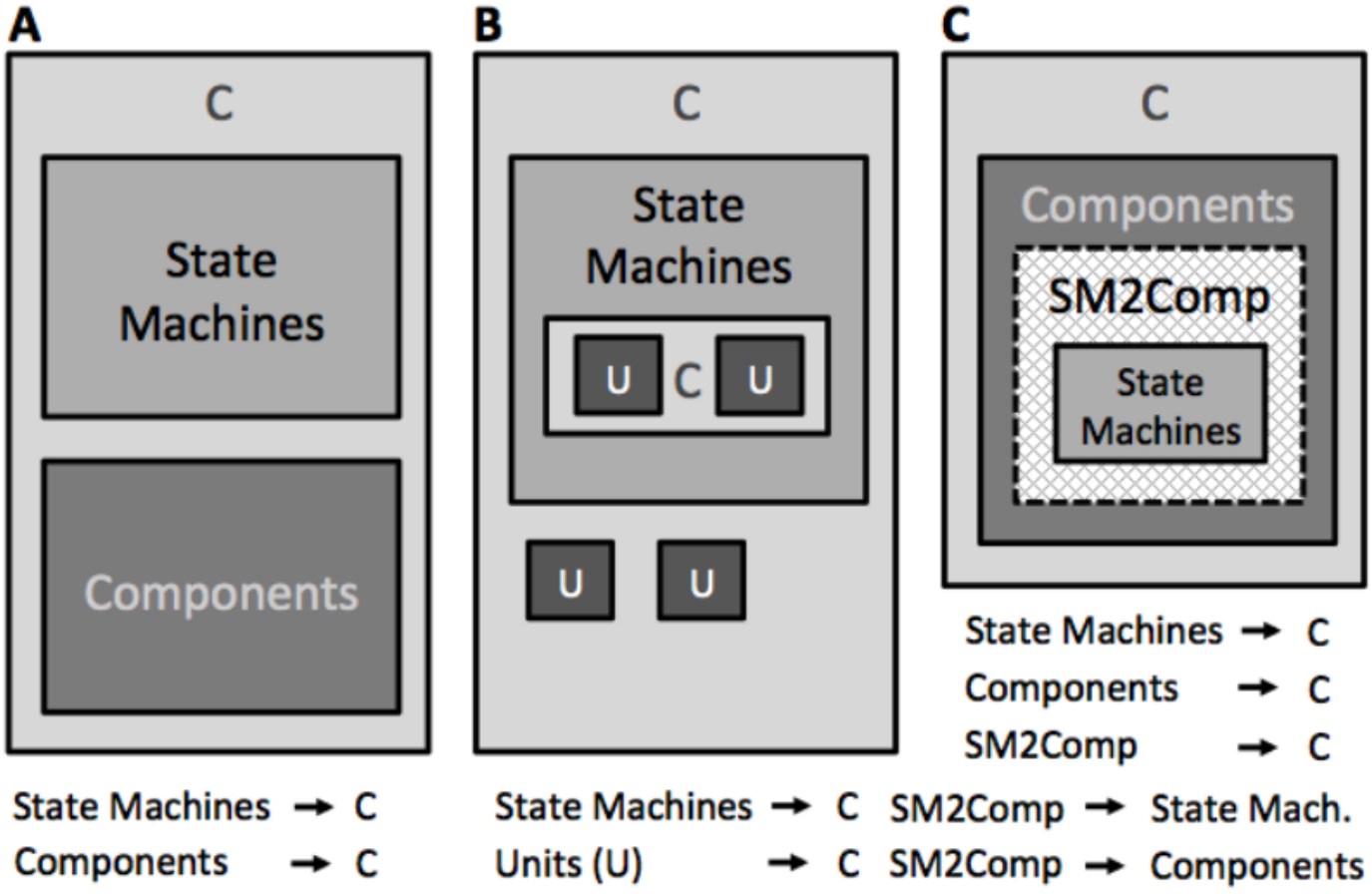
Modularity

OO-style composition for structure and syntax



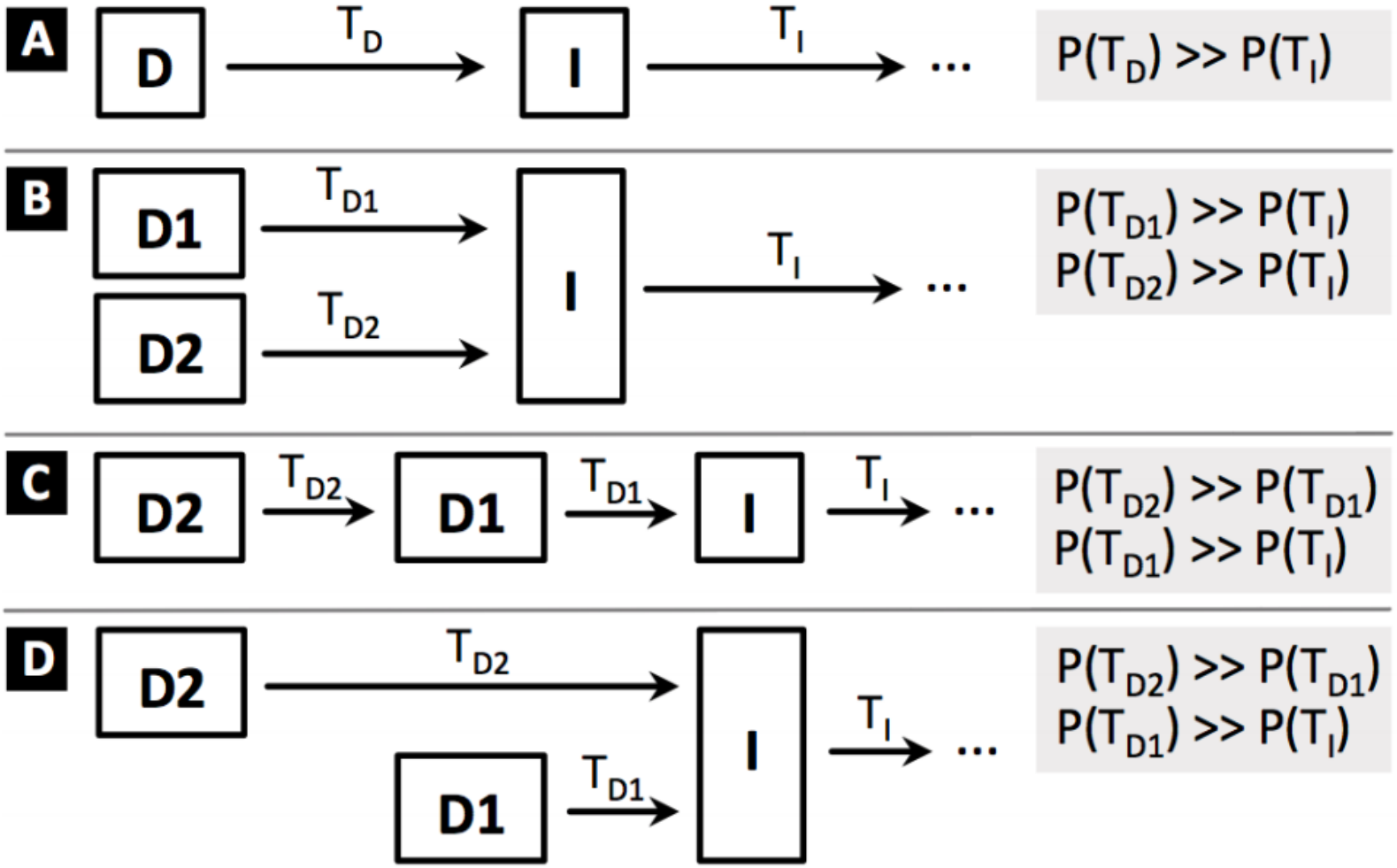
Modularity

Fine-grained nesting



Modularity

Modular semantics/generators



Modularity Wrap Up

mbeddr's 34 extensions to C are a clear indication that MPS' language modularity works. Modularity is useful for language understanding, testing and reuse.

In rare cases, modularity is compromised by necessary changes to the base language and unwanted dependencies between independent extensions.

Currently there is no way to detect (unwanted) semantic interactions between independent language extensions through analysis of their transformations.

RQ2: Projection

What is the contribution of projectional editing to the success of mbeddr?

Projection

Variety of notations used in mbeddr

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) {
  return (-b + √(b^2 - Σ_{i=1}^{4} a * c)) / (2 * a);
} midnight2 (function)
```

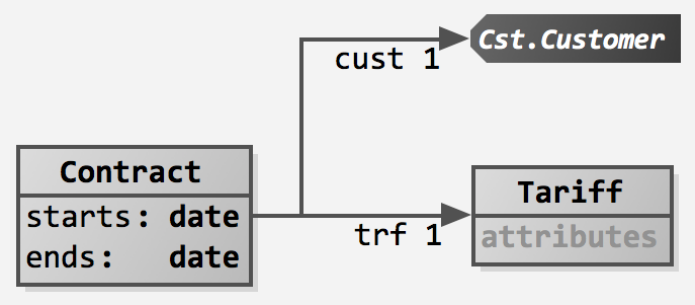
Tables

```
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



The two main benefits of projectional editing – language modularity and a range of combinable notations – have been used extensively in mbeddr. The anticipated benefits have been observed.

The editor can be flexibly extended with new notational styles with acceptable effort, as exemplified by the support for math, tables and diagrams.

The ability to use multiple and partial projections must be further improved by integrating with other language aspects, in particular, editor actions and type checks.

RQ3: Complexity

How effective are MPS' mechanisms for managing the complexity inherent in language development?

Complexity Wrap Up

The approach of using a DSL for each language aspect works well based on our experience, even though some aspects are missing and some are not declarative enoughto support meaningful analyses.

The support for debugging is spotty: it works well for transformations, but debugging generator macros, behaviors and type system rules is very tedious.

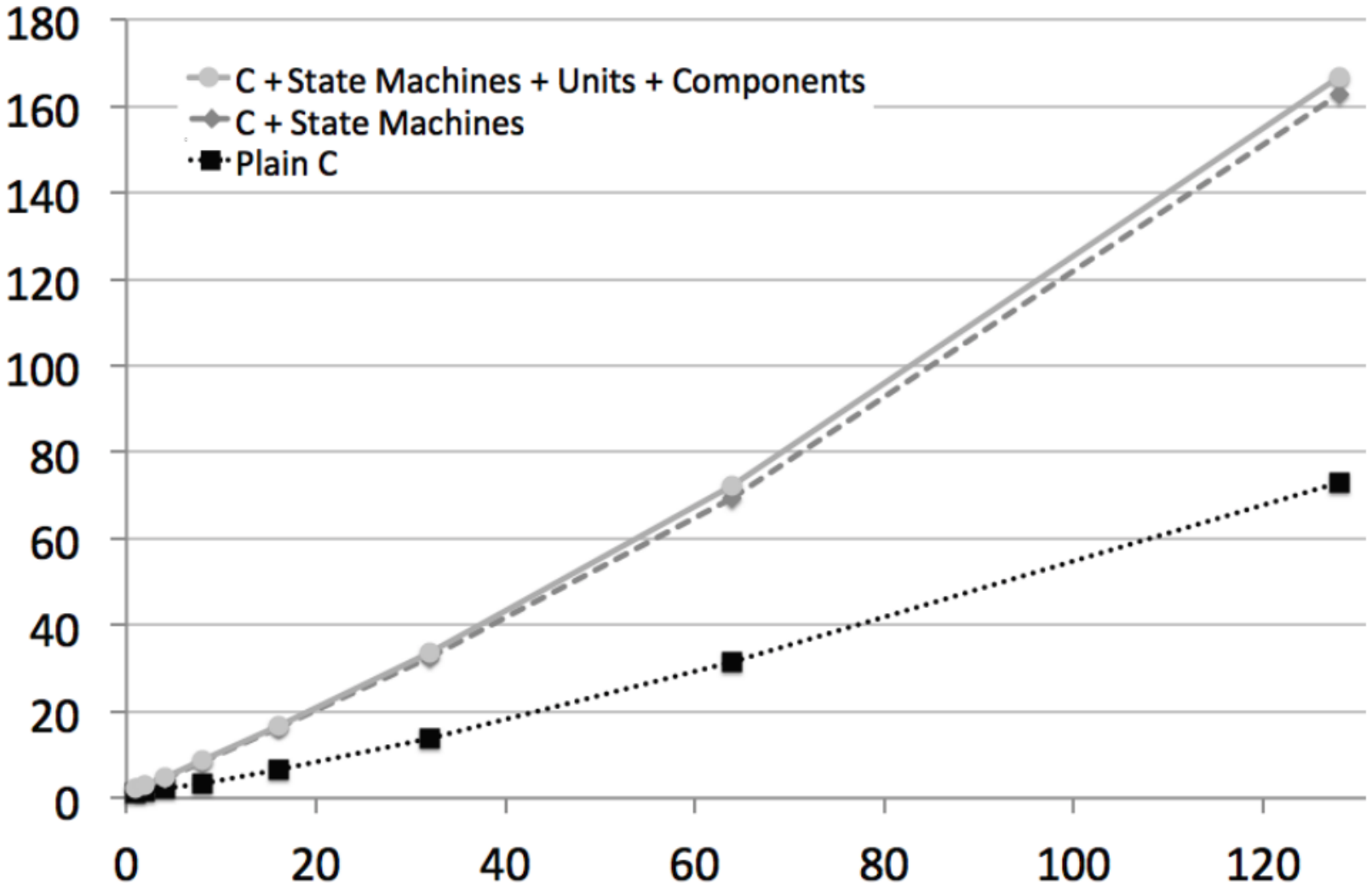
The ability to extend MPS' language definition DSLs with MPS itself is a powerful approach for managing complexity, and we have used it extensively, even though it has some limitations.

RQ4: Scalability

What are the performance and scalability implications?

Scalability

Generator Performance



Scalability Wrap Up

If attention is paid to the size of roots and the distribution of code over multiple models, then systems of significantsize can be built with MPS.

The performance of the type system (as it is evaluated in realtime in the editor) and support for cross-model generation are the two most critical ways of improving MPS performance.

During the development of languages we have not run into any problems regarding performance or scalability (of editor, type system or generator definitions).

RQ5: Process

What are the interactions with the development process?

Process

Language Testing Support

```
for ( int8 i = 0; i < 10

int8 add(int8 x, int8 y) {
  int8[3] a = {1, 2, 3};
  int8[3] b;
  struct
  return x + y;
} add (function)
```

```
Test case testSideTransformations
nodes
  ( AnImplementationModule   constraints
                             imports     nothing

    void f() {
      <expr  4 + 3 * 2>;
    } f (function)

test methods
  test testPrecedence {
    assert expr.isInstanceOf(PlusExpression);
    assert expr.left.isInstanceOf(NumberLiteral);
           expr.right.isInstanceOf(MultiExpression);
```

```
Test case testingTypeChecksForLocalVariables
nodes
  ( AnImplementationModule    constraints
                              imports     nothing

    void f() {
      int8 anInt = 42;
      int8 aFloat = <check 3.3 has error>;
      int8 aString = <check "hello" has error>;
    } f (function)
```

```
statemachine SM initial = S1 {
  in event e()
  in event f()
  state S1 { on e [] -> S2 }
  state S2 { on e [] -> S3 }
  state S3 { on e [] -> S1 }
}
```

```
exported testcase testState {
  SM sm;
  assert(0) sm.isInState(S1);
  test statemachine sm {
    e → S2
    f → S2
    e → S3
    e → S1
  }
}
```

Except for the missing test support for model migrations and single-step transformations, language testing works well, and we have achieved good coverage as demonstrated by a stable code base.

We have successfully integrated mbeddr's build, test and packaging with the Teamcity CI server, but the effort to get there was significant, partially as a consequence of the inadequacy of MPS' build language.

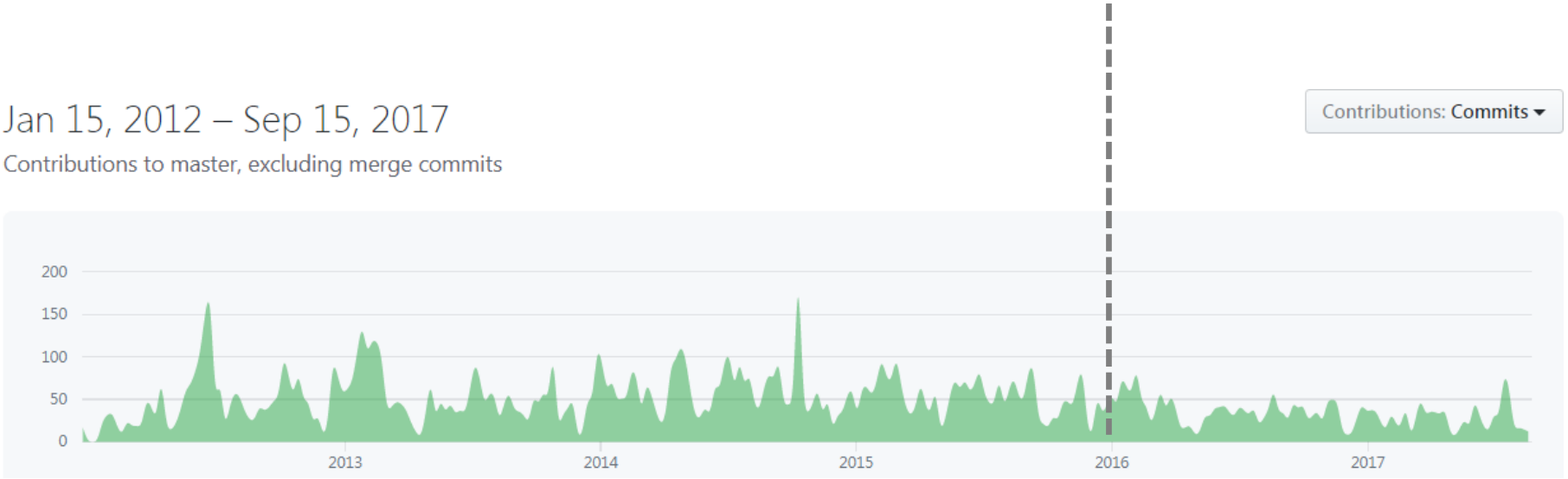
Migrating instance models as the underlying languages change incompatibly is feasible with manually scripted migrations and their automatic execution based on implicitly-maintained language version numbers.

Back to the Present

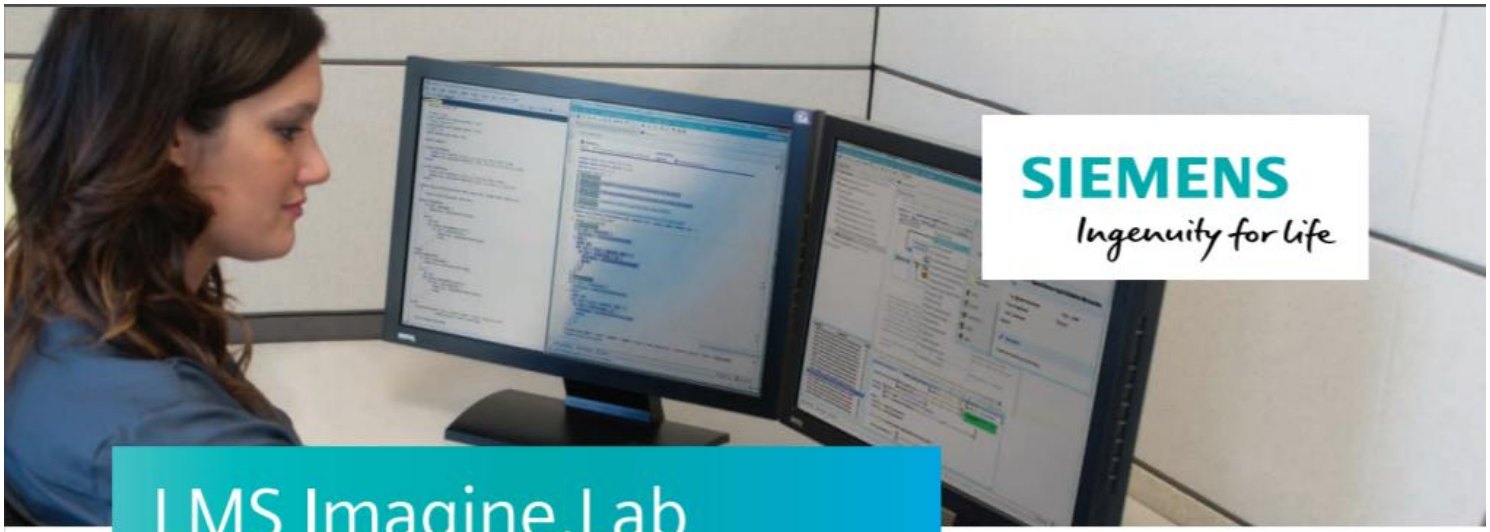
mbeddr Today

Our SoSym paper “ends” in 12.2015

“new” developments



Beyond mbeddr …



LMS Imagine.Lab Embedded Software Designer

Accelerating on-board software development with a model-based, test-driven approach

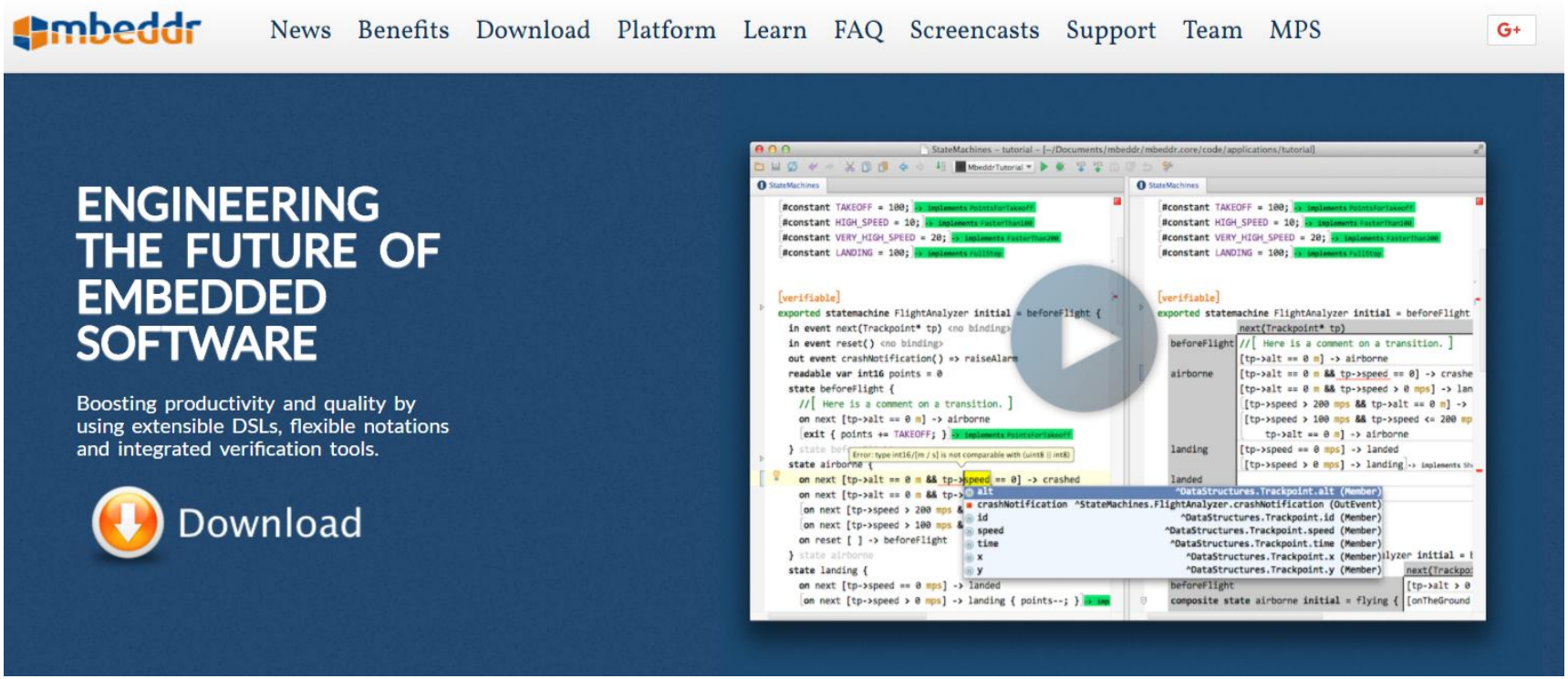
LMS™ Imagine.Lab™ Embedded Software Designer is an integrated development environment (IDE) for efficient model-based test-driven development of on-board software. Application domains include massively customized software-intensive and cyber-physical systems such as smart vehicles, home appliances or buildings, families of smart adaptive utilities, and transportation and tourism services based on continuous analysis of data generated by the Internet of Things (IoT).

mbeddr today is 20% bigger than in 12.2015

ESD doubles the size of mbeddr today

All Findings Remain Valid

mbeddr Thank you!



www.mbeddr.com

Open Source, Eclipse Public License

https://github.com/mbeddr/mbeddr.core