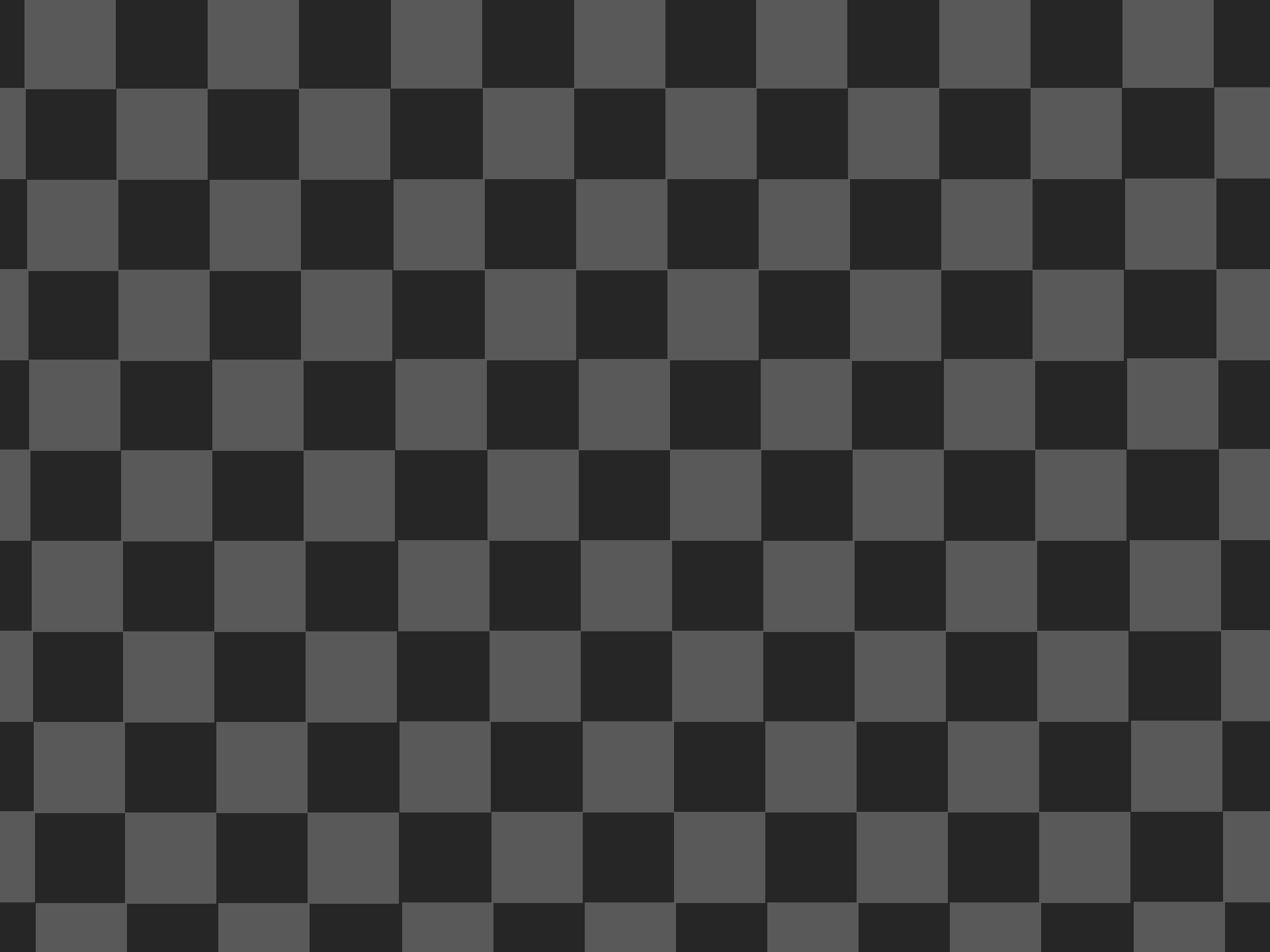


# Generic Tools - Specific Languages On the Art of Software Building Development Tools

Markus Voelter  
independent/itemis

<http://voelter.de>  
[voelter@acm.org](mailto:voelter@acm.org)  
[@markusvoelter](#)





**1** Tools

**4** GTSL

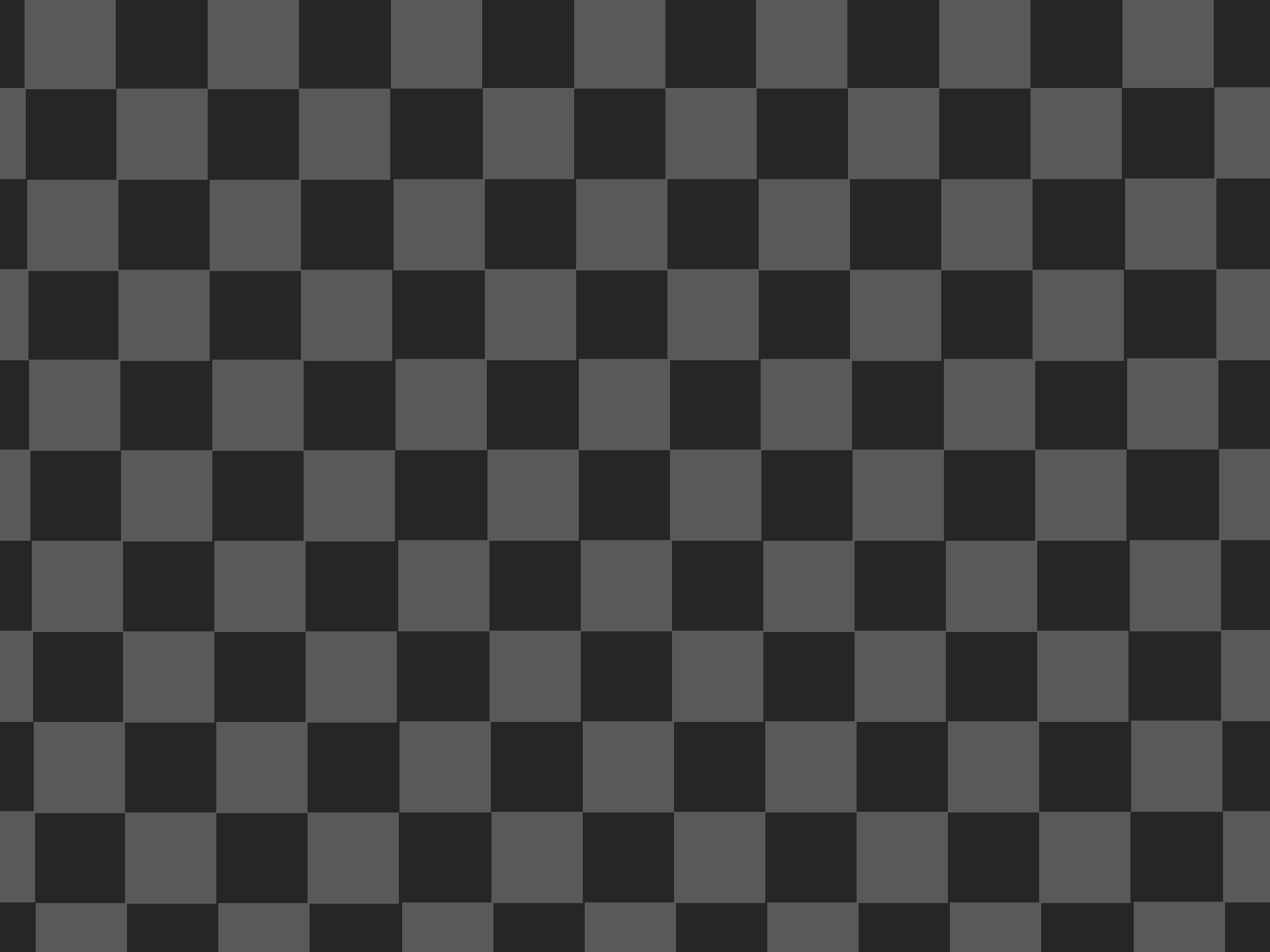
**2** Extensibility

**5** An Example

**3** Challenges

**6** Wrap Up



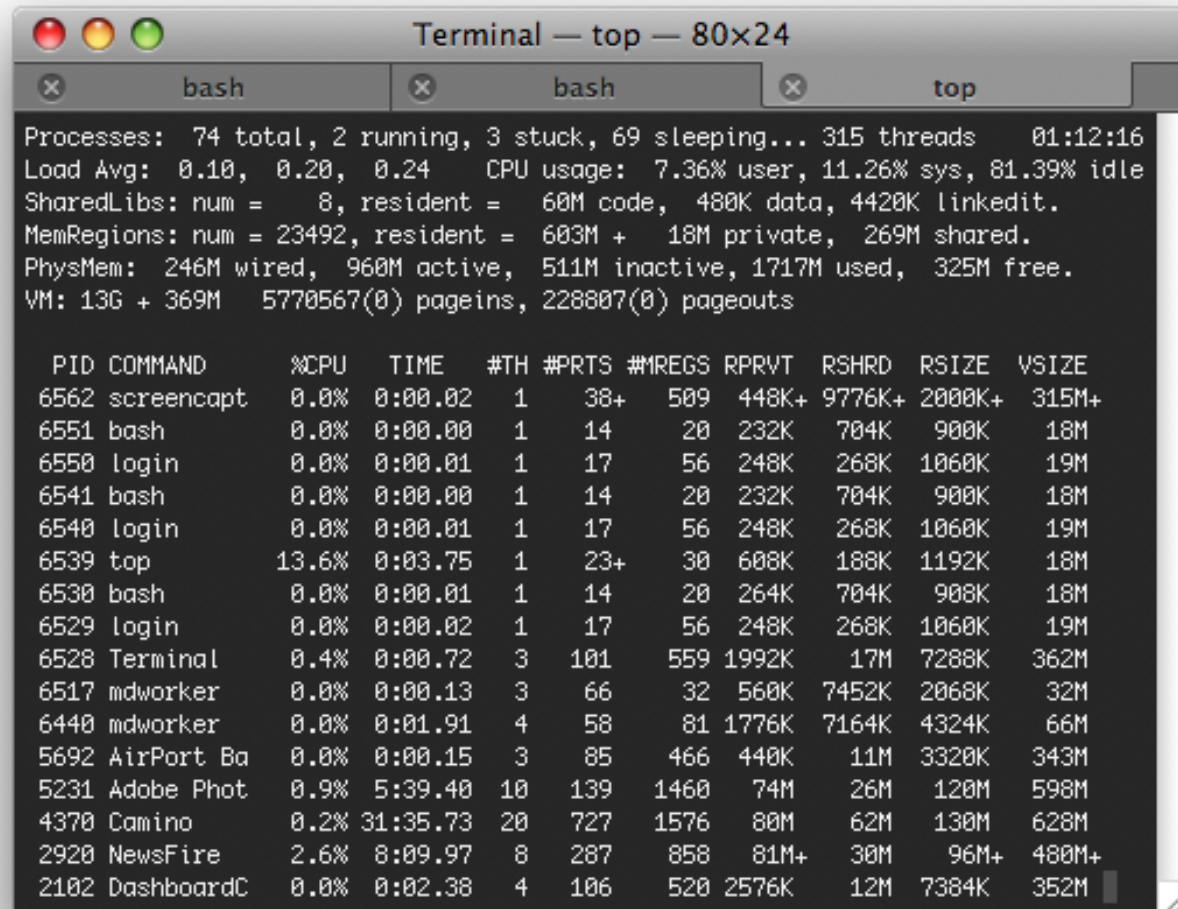




# Tools

# Tools

## Command-Line Tools



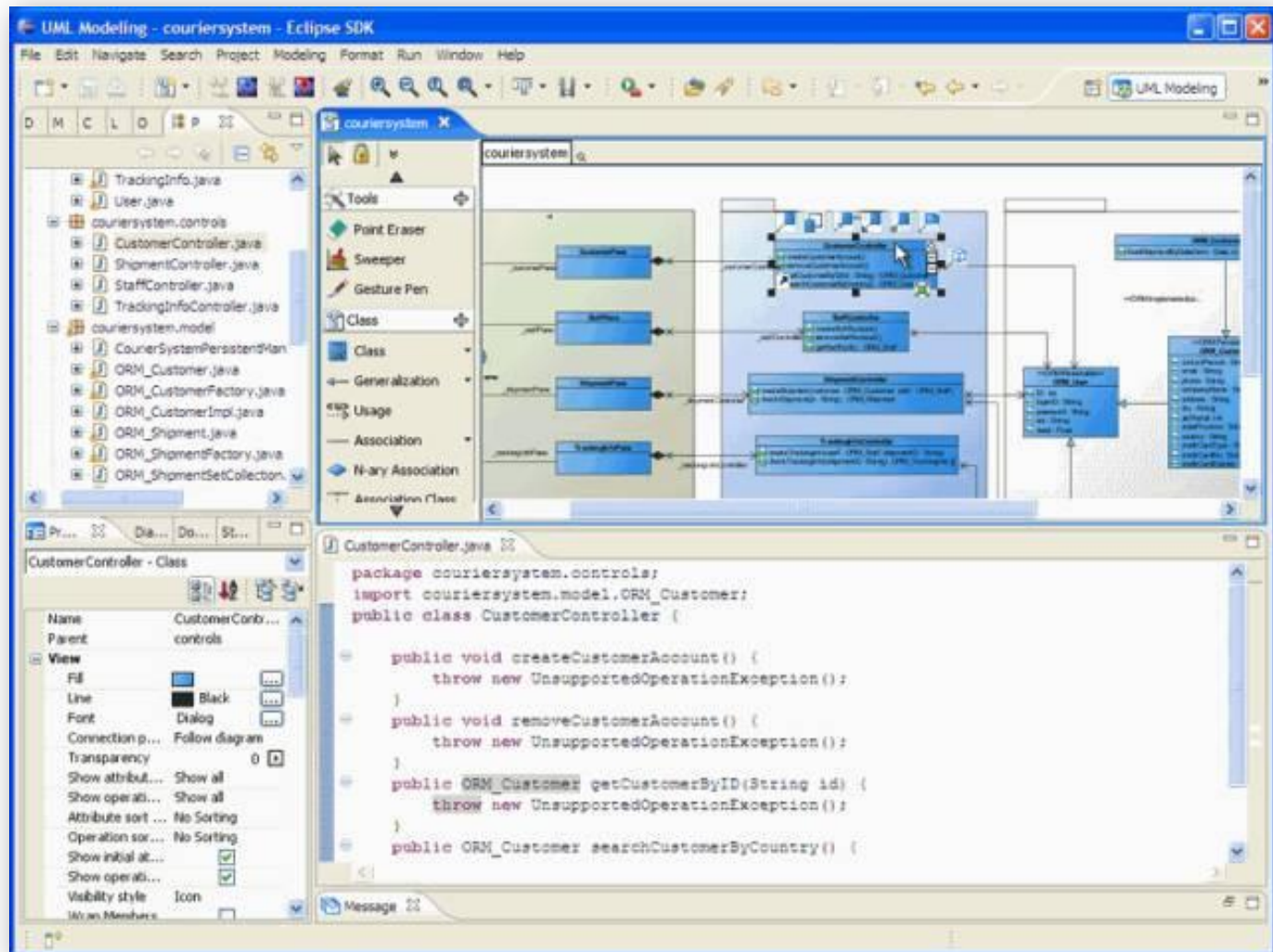
A screenshot of a macOS Terminal window titled "Terminal — top — 80x24". The window has three tabs: "bash", "bash", and "top". The "top" tab is active, displaying system statistics and a list of running processes. The statistics section shows 74 total processes, 2 running, 3 stuck, and 69 sleeping. CPU usage is 7.36% user, 11.26% sys, and 81.39% idle. The process list is a table with columns: PID, COMMAND, %CPU, TIME, #TH, #PRTS, #MREGS, RPRVT, RSHRD, RSIZE, and VSIZE. The processes are sorted by CPU usage, with "top" at the top (13.6%) and "DashboardC" at the bottom (0.0%).

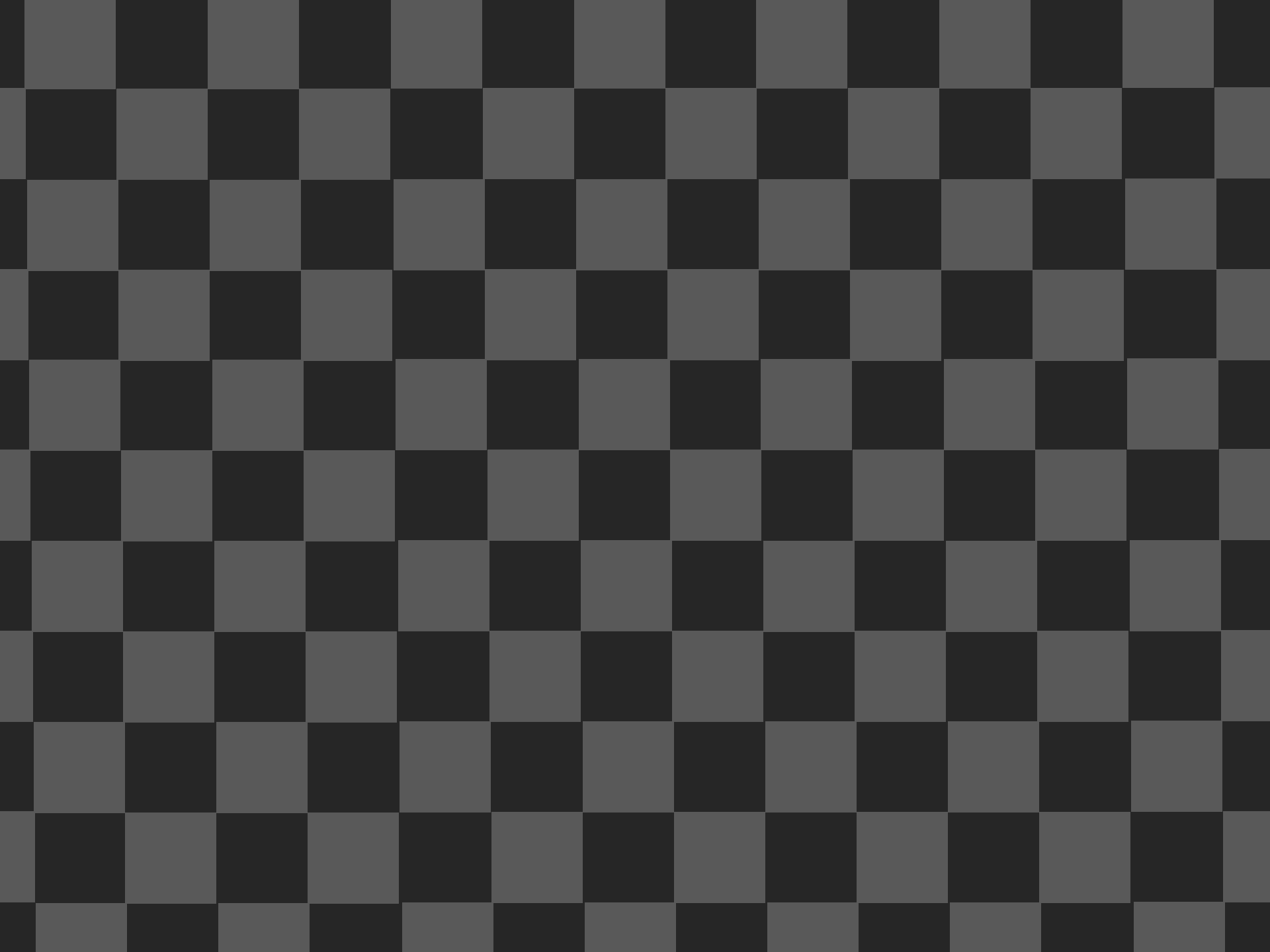
```
Processes: 74 total, 2 running, 3 stuck, 69 sleeping... 315 threads    01:12:16
Load Avg: 0.10, 0.20, 0.24    CPU usage: 7.36% user, 11.26% sys, 81.39% idle
SharedLibs: num =    8, resident =    60M code, 480K data, 4420K linkedit.
MemRegions: num = 23492, resident = 603M + 18M private, 269M shared.
PhysMem: 246M wired, 960M active, 511M inactive, 1717M used, 325M free.
VM: 13G + 369M    5770567(0) pageins, 228807(0) pageouts

  PID COMMAND      %CPU   TIME   #TH  #PRTS  #MREGS  RPRVT  RSHRD  RSIZE  VSIZE
  ---  ---
 6562 screencapt    0.0%   0:00.02    1   38+    509   448K+  9776K+ 2000K+ 315M+
 6551 bash          0.0%   0:00.00    1   14     20   232K   704K   900K   18M
 6550 login         0.0%   0:00.01    1   17     56   248K   268K  1060K   19M
 6541 bash          0.0%   0:00.00    1   14     20   232K   704K   900K   18M
 6540 login         0.0%   0:00.01    1   17     56   248K   268K  1060K   19M
 6539 top           13.6%   0:03.75    1  23+     30   608K   188K  1192K   18M
 6530 bash          0.0%   0:00.01    1   14     20   264K   704K   900K   18M
 6529 login         0.0%   0:00.02    1   17     56   248K   268K  1060K   19M
 6528 Terminal       0.4%   0:00.72    3   101    559  1992K    17M  7288K  362M
 6517 mdworker       0.0%   0:00.13    3    66     32   560K   7452K  2068K    32M
 6440 mdworker       0.0%   0:01.91    4    58     81  1776K   7164K  4324K    66M
 5692 AirPort Ba    0.0%   0:00.15    3    85    466   440K    11M  3320K   343M
 5231 Adobe Phot     0.9%   5:39.40   10   139   1460    74M    26M   120M   598M
 4370 Camino        0.2%  31:35.73   20   727   1576    80M    62M   130M   628M
 2920 NewsFire      2.6%   8:09.97    8   287    858   81M+    30M    96M+  480M+
 2102 DashboardC    0.0%   0:02.38    4   106    520  2576K    12M  7384K   352M
```

# Tools

## UI Tools









**Tool**

# Tool Extensibility

## Study Findings I

The majority of our interviewees were very successful with MDE but all of them either built their own modeling tools, made heavy adaptations of off-the-shelf tools, or spent a lot of time finding ways to work around tools. The only accounts of easy-to-use, intuitive tools came from those who had developed tools themselves for bespoke purposes. Indeed, this suggests that current tools are a barrier to success rather than an enabler.

# Tool Extensibility

## Study Findings II

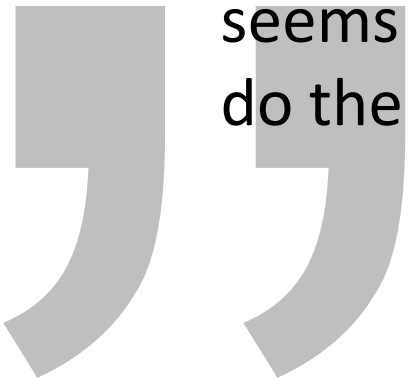
Complexity problems are typically associated with off-the-shelf tools. Of particular note is accidental complexity – which can be introduced due to poor consideration of other categories, such as lack of flexibility to adapt the tools to a company's own context [..]

# Tool Extensibility

## Study Findings III



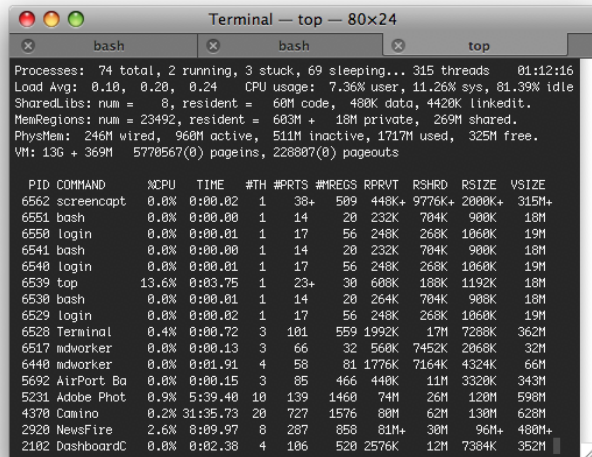
Our interviews point to a strong need for tailoring of some sort: either tailor the tool to the process, tailor the process to the tool, or build your own tool that naturally fits your own process. Based on our data, it seems that, on balance, it is currently much easier to do the latter.



Jon Whittle, John Hutchinson, Mark Rouncefield, Hakan Burden, and Rogardt Heldal. Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS) 2013*. ACM, 2013.

# Tool Extensibility

## Command-Line Tools



Terminal — top — 80x24

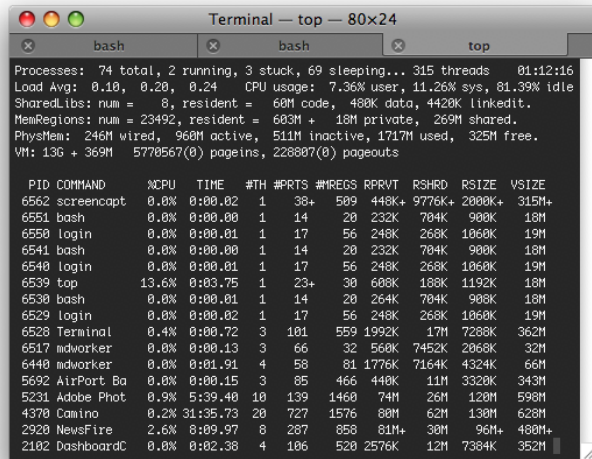
Processes: 74 total, 2 running, 3 stuck, 69 sleeping... 315 threads 01:12:16  
Load Avg: 0.10, 0.20, 0.24 CPU usage: 7.36% user, 11.26% sys, 81.39% idle  
SharedLibs: num = 8, resident = 60M code, 480K data, 4420K linkedit.  
MemRegions: num = 23492, resident = 603M + 18M private, 269M shared.  
PhysMem: 246M wired, 968M active, 511M inactive, 1717M used, 325M free.  
VM: 13G + 369M 5770567(0) pageins, 228807(0) pageouts

PID	COMMAND	%CPU	TIME	#TH	#PRTS	#MREGS	RPRVT	RSHRD	RSIZE	VSIZE
6562	screencapt	0.0%	0:00.02	1	38+	509	448K+	9776K+	2000K+	315M+
6551	bash	0.0%	0:00.00	1	14	20	232K	704K	900K	18M
6550	login	0.0%	0:00.01	1	17	56	248K	268K	1060K	19M
6541	bash	0.0%	0:00.00	1	14	20	232K	704K	900K	18M
6540	login	0.0%	0:00.01	1	17	56	248K	268K	1060K	19M
6539	top	13.6%	0:03.75	1	23+	30	608K	188K	1192K	18M
6530	bash	0.0%	0:00.01	1	14	20	264K	704K	908K	18M
6529	login	0.0%	0:00.02	1	17	56	248K	268K	1060K	19M
6528	Terminal	0.4%	0:00.72	3	101	559	1992K	17M	7288K	362M
6517	mdworker	0.0%	0:00.13	3	66	32	560K	7452K	2068K	32M
6440	mdworker	0.0%	0:01.91	4	58	81	1776K	7164K	4324K	66M
5692	AirPort Ba	0.0%	0:00.15	3	85	466	440K	11M	3320K	343M
5231	Adobe Phot	0.9%	5:39.40	10	139	1460	74M	26M	120M	598M
4370	Camino	0.2%	31:35.73	20	727	1576	80M	62M	130M	628M
2920	NewsFire	2.6%	8:09.97	8	287	858	81M+	30M	96M+	480M+
1102	DashboardC	0.0%	0:02.38	4	106	520	2576K	12M	7384K	352M

New File Formats  
New Processors

# Tool Extensibility

## Command-Line Tools



A screenshot of a macOS Terminal window titled "Terminal — top — 80x24". The window shows the output of the 'top' command, which displays system statistics and a list of running processes. The processes are sorted by CPU usage, with 'top' being the most CPU-intensive at 13.6%.

```
Processes: 74 total, 2 running, 3 stuck, 69 sleeping... 315 threads 01:12:16
Load Avg: 0.10, 0.20, 0.24 CPU usage: 7.36% user, 11.26% sys, 81.39% idle
SharedLibs: num = 8, resident = 60M code, 480K data, 4420K linkedit.
MemRegions: num = 23492, resident = 603M + 18M private, 269M shared.
PhysMem: 246M wired, 968M active, 511M inactive, 1717M used, 325M free.
VM: 13G + 369M 5770567(0) pageins, 228887(0) pageouts

  PID COMMAND           %CPU   TIME    #TH  #PRTS  #MREGS  RPRVT  RSHRD  RSIZE  VSIZE
6562 screencapt         0.0%  0:00.02    1   38+   509   448K+  9776K+ 2000K+ 315M+
6551 bash                0.0%  0:00.00    1    14    20    232K   704K   900K   18M
6550 login              0.0%  0:00.01    1    17    56    248K   268K  1060K   19M
6541 bash                0.0%  0:00.00    1    14    20    232K   704K   900K   18M
6540 login              0.0%  0:00.01    1    17    56    248K   268K  1060K   19M
6539 top                13.6%  0:03.75    1   23+    30    608K   188K  1192K   18M
6530 bash                0.0%  0:00.01    1    14    20    264K   704K   908K   18M
6529 login              0.0%  0:00.02    1    17    56    248K   268K  1060K   19M
6528 Terminal           0.4%  0:00.72    3   101   559  1992K   17M   7288K  362M
6517 mdworker           0.0%  0:00.13    3    66    32    560K   7452K  2068K   32M
6440 mdworker           0.0%  0:01.91    4    58    81  1776K   7164K  4324K   66M
5692 AirPort Ba         0.0%  0:00.15    3    85   466   440K    11M  3320K   343M
5231 Adobe Phot          0.9%  5:39.40   10   139  1460    74M    26M   120M  598M
4370 Camino              0.2% 31:35.73   20   727  1576    80M    62M   130M  628M
2920 NewsFire           2.6%  0:09.97    8   287   858    81M+   30M   96M+  480M+
2102 DashboardC         0.0%  0:02.38    4   106   520  2576K    12M   7384K  352M
```

# New File Formats

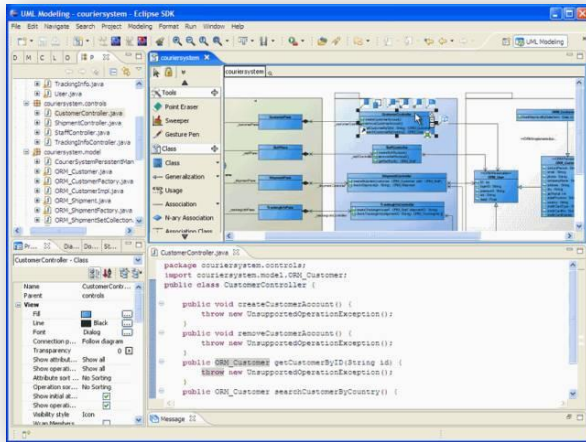
# New Processors

---

## Assemble Components (Pipes & Filters)

# Tool Extensibility

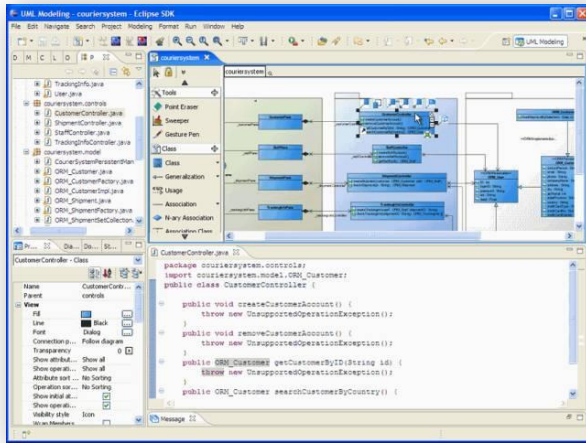
## UI Tools



Buttons Views  
Menus Actions

# Tool Extensibility

## UI Tools

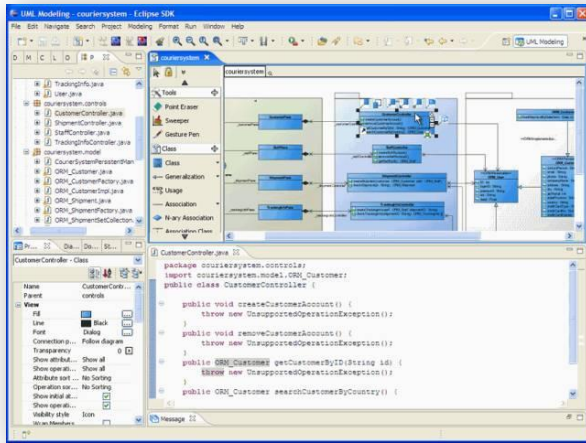


Buttons Views  
Menus Actions  
New Languages  
New Editors



# Tool Extensibility

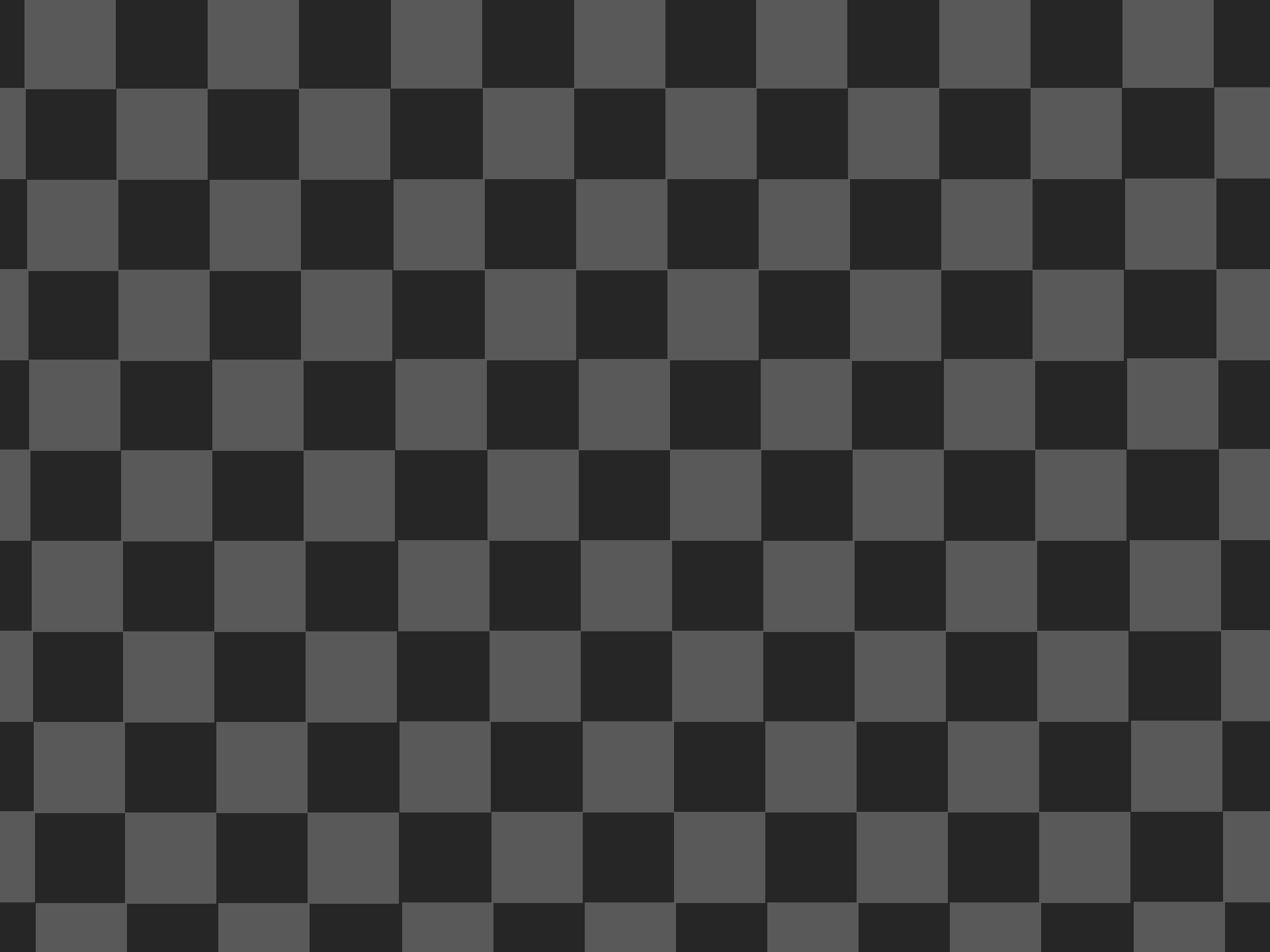
## UI Tools



Buttons Views  
Menus Actions  
New Languages  
New Editors

---

**Platform/Plugin Systems**





**3**

**Challenges**

# Overview

**Context: embedded programming**

Examples are from Embedded  
Programming and use C as the  
„data format“

But applies similarly to other  
or other data formats/languages

# Extensibility Examples

## Physical Units: The Challenge

How do you work with  
physical units in your c

# Extensibility Examples

## Physical Units: The Challenge

```
// in file example.c
int distance = 10;
int time      = 1;
int speed     = distance / time;
```

# Extensibility Examples

## Physical Units: The Challenge

```
// in file example.c  
int distance = 10;  
int time      = 1;  
int speed     = distance / time;
```

```
int speed = time / distance;
```

How do you **detect this error**

# Extensibility Examples

## Physical Units: The Challenge

```
int speed = time / distance;
```

How do you **detect this error**

You **need to do the  
checking)**

**Data (the units in  
the code)**



# Extensibility Examples

## Physical Units via Comments

```
int/*#m*/ distance = 10 /*#m*/;  
int/*#s*/ time     = 1  /*#s*/;  
int/*#mps*/ speed  = distance / time;
```

Bao

# Extensibility Examples

## Physical Units via Macros

```
UT(int, m) distance = UV(10, s);  
UT(int, s) time      = UV(1, s);  
UT(int, mps) speed   = distance / time;
```

Bac

# Extensibility Examples

## Physical Units via external XML

```
<unitdeclarations>
  <unit name="m" for="distance"/>
  <unit name="s" for="time"/>
  <unit name="mps" for="speed" calculateAs="m/s"/>
</unitdeclarations>
<programmarkup>
  <globalvar file="example.c" name="distance" unit="m"/>
  <globalvar file="example.c" name="time" unit="s"/>
  <globalvar file="example.c" name="speed" unit="mps"/>
</programmarkup>
```

Bao

# Extensibility Examples

## Physical Units via Extension

**int**/m/ distance = 10 m;

**int**/s/ time = 1 s;

**int**/mps/ speed = distance / time;

Good

# Extensibility Examples

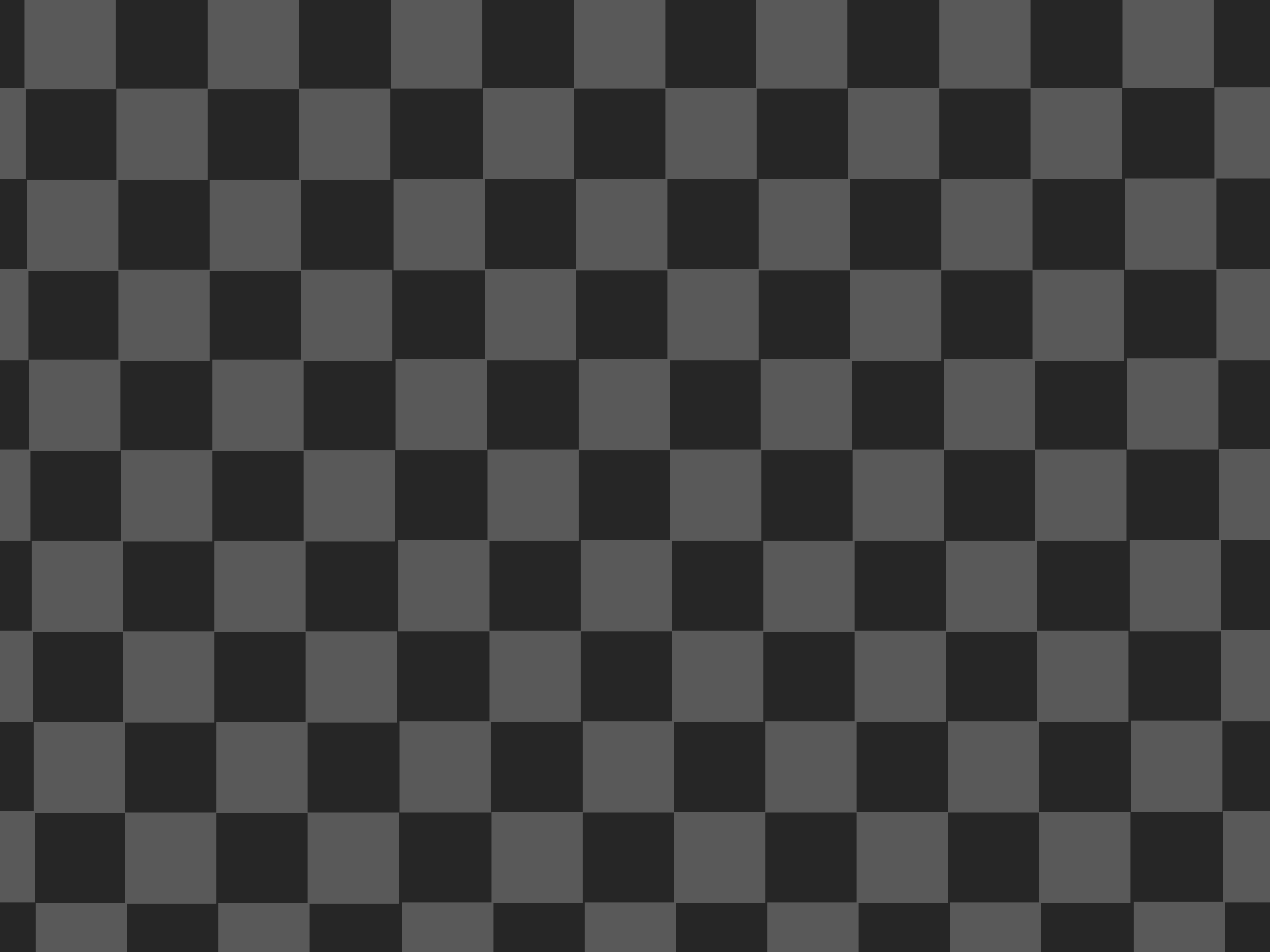
## Physical Units via Extension

```
int/m/ distance  = 10 m;  
int/s/ time      = 1 s;  
int/mps/ speed   = distance / time;
```

You get a **TypeChecker** (to do the checking)

Program Code (the units in the code)

Good



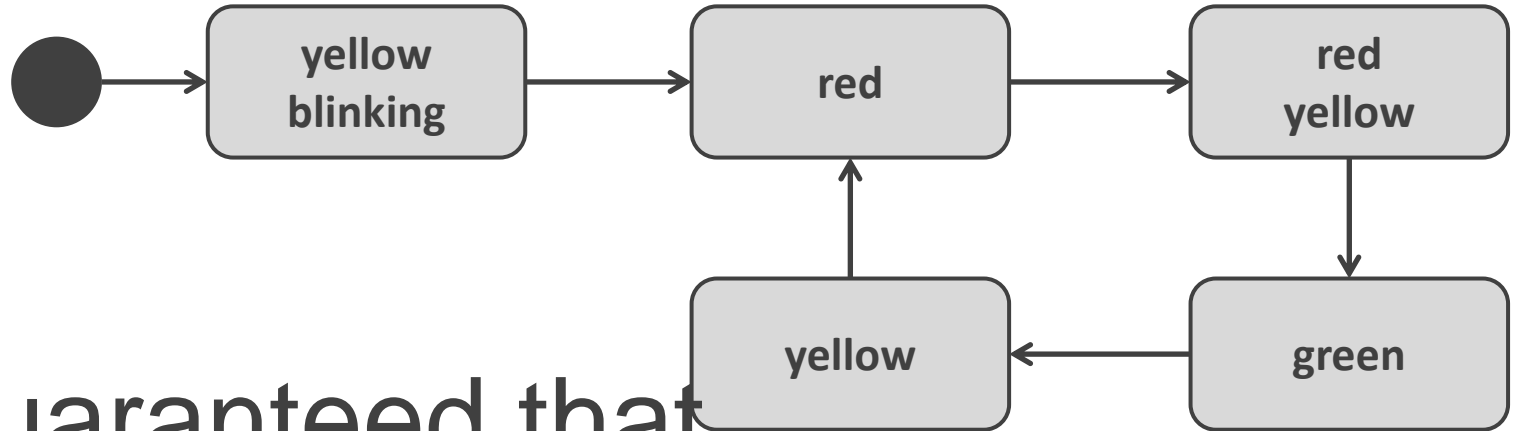
# Extensibility Examples

## State Machines: The Challenge

How do you represent  
state-based behavior in  
and support analyses?

# Extensibility Examples

## State Machines: The Challenge



Is it guaranteed that ...

... the TL get green eventually?

... if the TL is turned off/on, it starts in

... the TL never goes from yellow to g



# Extensibility Examples

## State Machines via C idioms

```
// a state machine that transitions into S2
// when E1 is received while the machine is in S1
void execute_StateMachine( Event_Enum evt ) {
    switch (currentState) {
        case S1: switch (evt) {
            case E1: if ( guard for E1 in S1 ) {
                // execute exit actions for S1
                currentState = S2;
                // execute entry actions for S2
                break;
            }
        }
        case S2: ...
        ...
    }
}
```

Bao

# Extensibility Examples

## State Machines via C idioms

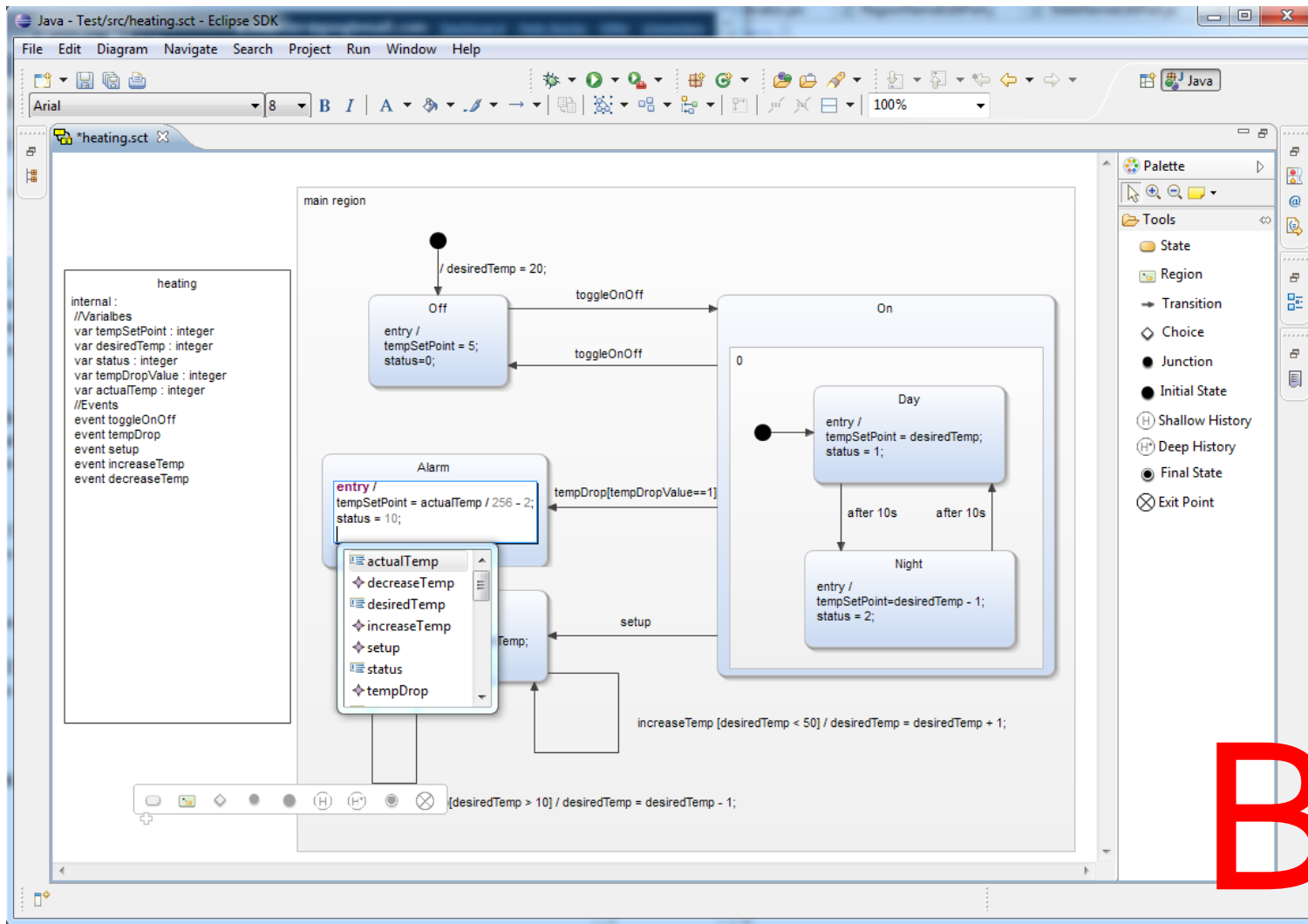
```
// a state machine that transitions into S2
// when E1 is received while the machine is in S1
// -1 means "do nothing".

//      S1  S2  S3  ...
int[N_EVT][N_STATE] = { { 1, -1, -1 } // E1
                        { -1, -1, -1 } // E2
                        ...
                        };
```

Bao

# Extensibility Examples

## State Machines via External Tool



Bao

# Extensibility Examples

## State Machines: The Challenge

How do you **perform**  
**analyses** on state machines

You need to do the  
checking)

Data („clean“ state  
machines)

# Extensibility Examples

## State Machines via Extensions

```
statemachine SM {  
  event E1  
  state S1 {  
    entry { // entry action for S1 }  
    on E1 [guard for E1 in S1] -> S2  
    exit { // exit action for S1 }  
  }  
  state S2 {  
    ...  
  }  
  ...  
}
```

Good

# Extensibility Examples

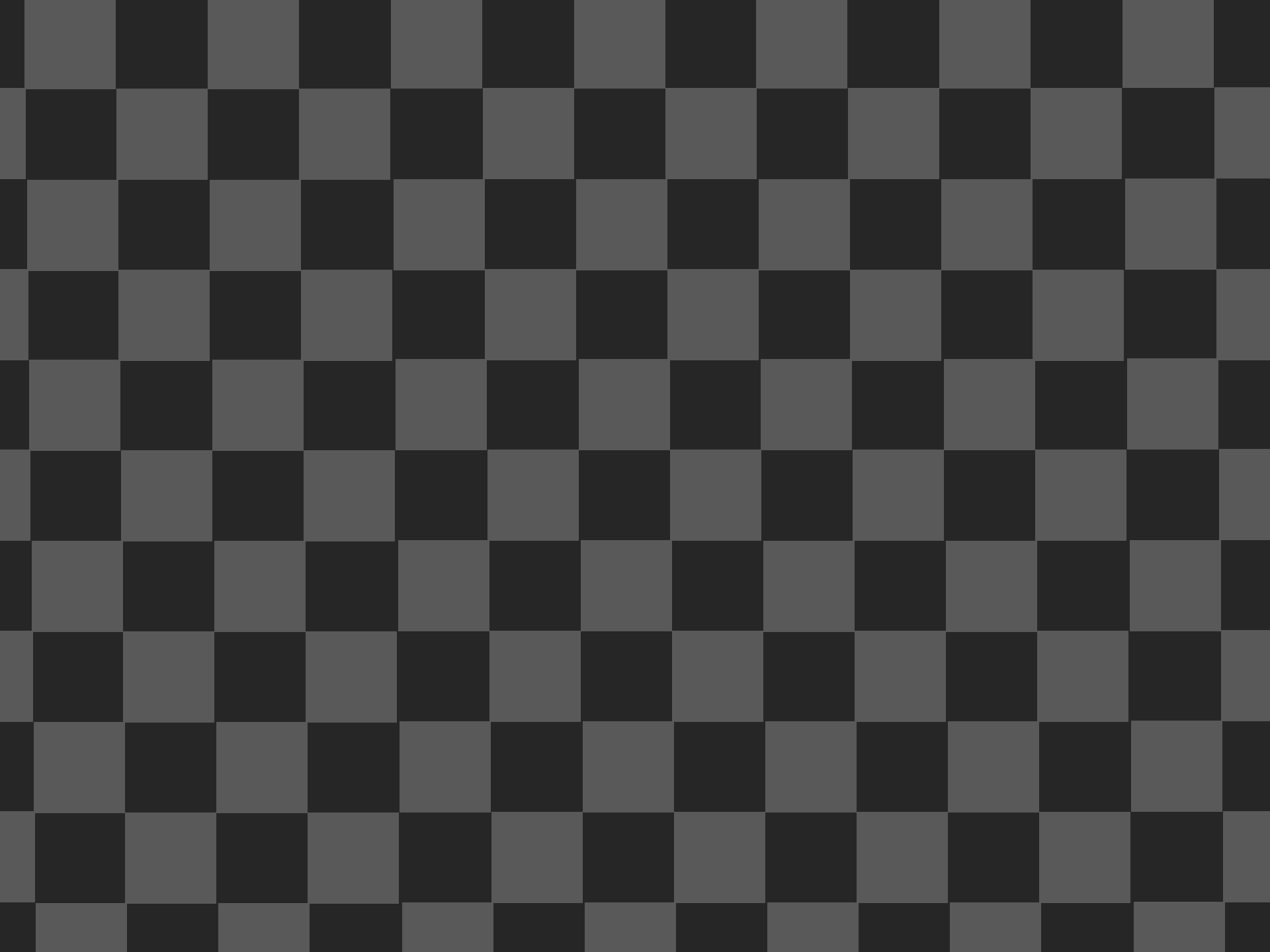
## State Machines via Extensions

```
statemachine SM {  
  event E1  
  state S1 {  
    entry { // entry action for S1 }  
    on E1 [guard for E1 in S1] -> S2  
    exit { // exit action for S1 }  
  }  
  state S2 {  
    ...  
  }  
  ...  
}
```

**You get a Constraint Checker (to do the checking)**

**Program Code (the units in the code)**

**Good**



# Extensibility Examples

## Tracing: The Challenge

How do you add trace requirements anywhere in your code, robustly?



# Extensibility Examples

## Tracing via Macros

```
TRACE(REQ_CALIBRATION)
int calibrate( int measurement ) {
    return measurement * FACTOR + OFFSET;
}

int getValue() {
    int raw = readFromDriver(ADC1_ADDRESS);
    TRACE(REQ_CALIBRATION)
    return calibrate(raw);
}
```

Bao

# Extensibility Examples

## Tracing via Macros

```
TRACE(REQ_CALIBRATION)
int calibrate( int measurement ) {
    return measurement * FACTOR + OFFSET;
}

int getValue() {
    int raw = readFromDriver(ADC1_ADDRESS);
    TRACE(REQ_CALIBRATION)
    return calibrate(raw);
}
```

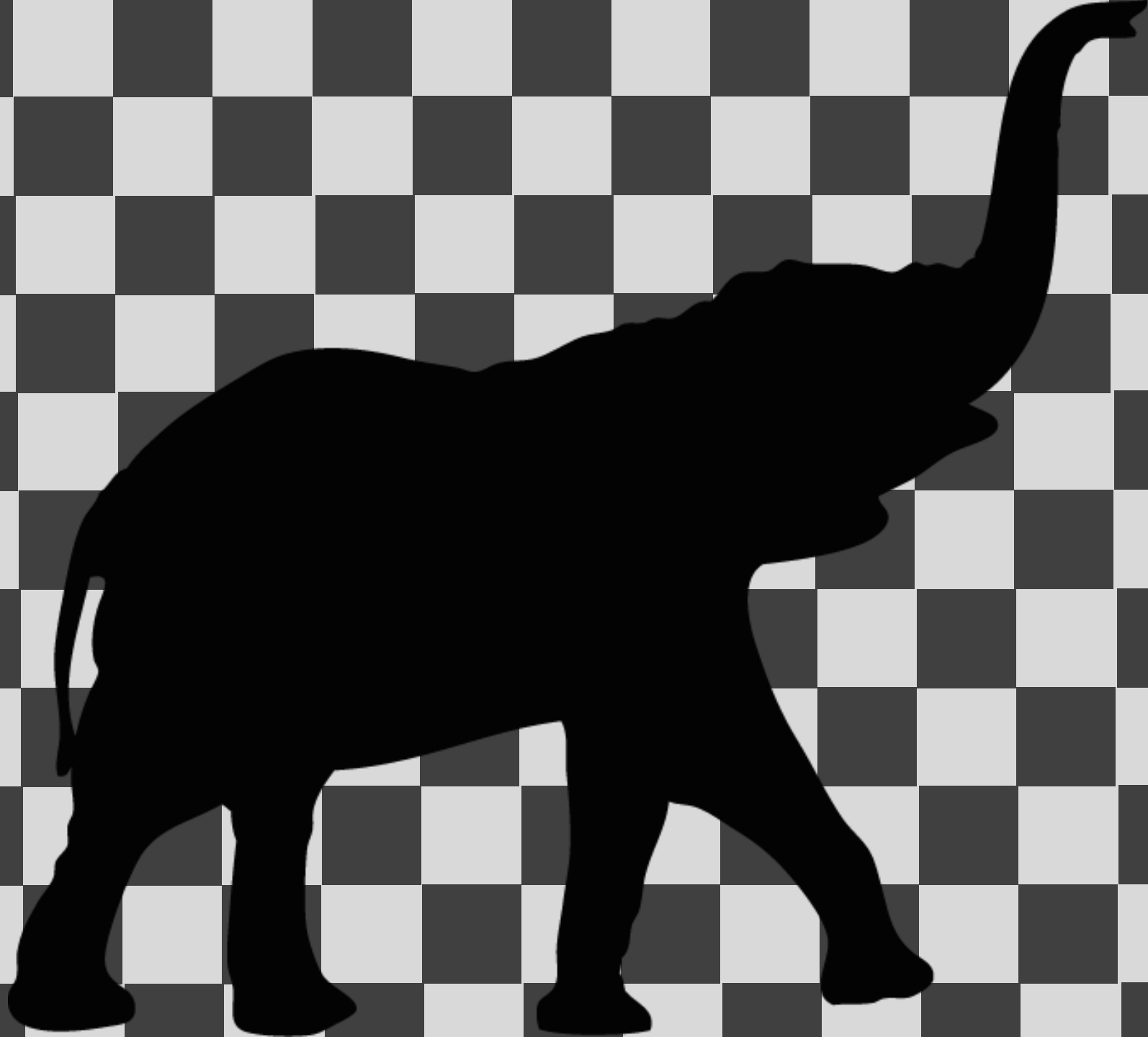
You need to create trace  
reports)

Data (robust trace  
annotations)

# Extensibility Examples

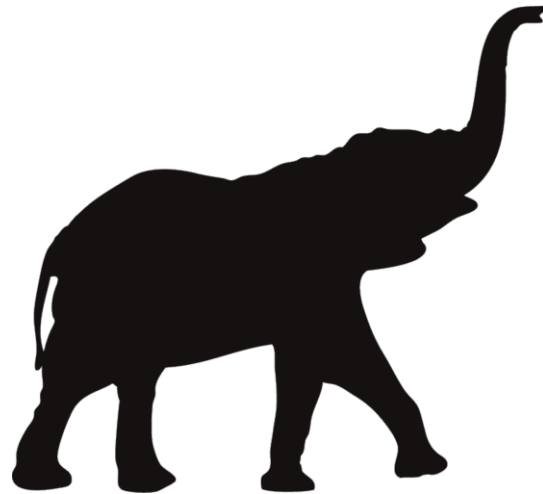
## Tracing via Language Extensions

You get the idea :-)



# Extensibility Examples

## Combinations

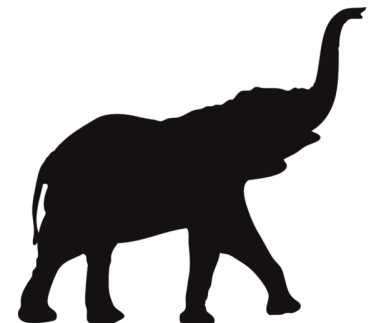


How do you combine these  
(and other) extensions?

# Extensibility Examples

## Combinations

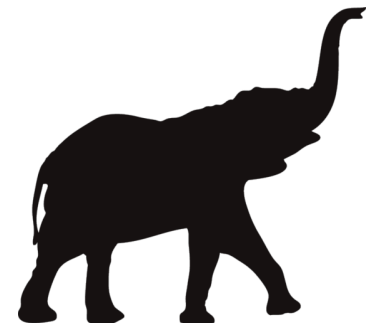
```
statemachine TrainDoorController {  
  event DOOR_BUTTON;  
  state DOORS_CLOSED {  
    trace REQ_BUTTON_OPENS_DOORS_ONLY_OPEN_WHEN_STOPPED  
    on DOOR_BUTTON [speed > 0 mps] -> DOORS_OPEN  
  }  
  state DOORS_OPEN {  
    entry { openDoors(); }  
    trace REQ_BUTTON_CLOSSES_DOORS_WHEN_OPEN  
    on DOOR_BUTTON [] -> DOORS_CLOSED  
    exit { closeDoors(); }  
  }  
  ...  
}
```



# Extensibility Examples

## Combinations

```
stateMachine TrainDoorController {  
    event DOOR_BUTTON;  
    state DOORS_CLOSED {  
        trace REQ_BUTTON_OPENS_DOORS_ONLY_OPEN_WHEN_STOPPED  
        on DOOR_BUTTON [speed > 0 mps] -> DOORS_OPEN  
    }  
    state DOORS_OPEN {  
        entry { openDoors(); }  
        trace REQ_BUTTON_CLOSSES_DOORS_WHEN_OPEN  
        on DOOR_BUTTON [] -> DOORS_CLOSED  
        exit { closeDoors(); }  
    }  
    ...  
}
```

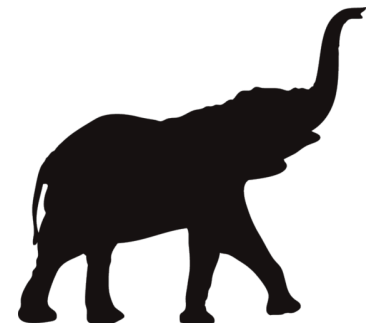


# Extensibility Examples

## Combinations

### State machines

```
statemachine TrainDoorController {  
  event DOOR_BUTTON;  
  state DOORS_CLOSED {  
    trace REQ_BUTTON_OPENS_DOORS_ONLY_OPEN_WHEN_STOPPED  
    on DOOR_BUTTON [speed > 0 mps] -> DOORS_OPEN  
  }  
  state DOORS_OPEN {  
    entry { openDoors(); }  
    trace REQ_BUTTON_CLOSSES_DOORS_WHEN_OPEN  
    on DOOR_BUTTON [] -> DOORS_CLOSED  
    exit { closeDoors(); }  
  }  
  ...  
}
```





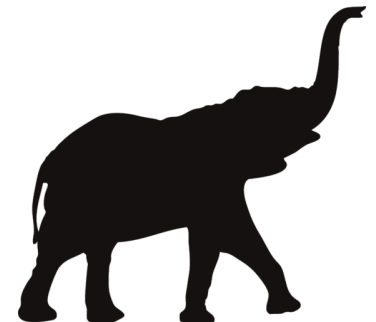
# Extensibility Examples

## Combinations

State  
machines

```
statemachine TrainDoorController {  
  event DOOR_BUTTON;  
  state DOORS_CLOSED {  
    trace REQ_BUTTON_OPENS_DOORS_ONLY_OPEN_WHEN_STOPPED  
    on DOOR_BUTTON [speed > 0 mps] -> DOORS_OPEN  
  }  
  state DOORS_OPEN {  
    entry { openDoors(); }  
    trace REQ_BUTTON_CLOSSES_DOORS_WHEN_OPEN  
    on DOOR_BUTTON [] -> DOORS_CLOSED  
    exit { closeDoors(); }  
  }  
  ...  
}
```

Tracing



# Extensibility Examples

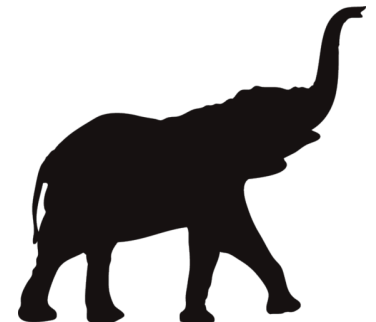
## Combinations

State  
machines

```
statemachine TrainDoorController {  
  event DOOR_BUTTON;  
  state DOORS_CLOSED {  
    trace REQ_BUTTON_OPENS_DOORS_ONLY_OPEN_WHEN_STOPPED  
    on DOOR_BUTTON [speed > 0 mps] -> DOORS_OPEN  
  }  
  state DOORS_OPEN {  
    entry { openDoors(); }  
    trace REQ_BUTTON_CLOSSES_DOORS_WHEN_OPEN  
    on DOOR_BUTTON [] -> DOORS_CLOSED  
    exit { closeDoors(); }  
  }  
  ...  
}
```

Units

Tracing



# Extensibility Examples

## Combinations (in an actual tool)

[verifiable]

```
// [ This state machine implements a way to grade flights.  
  It has separate states for the important flight phases,  
  such as @child(beforeFlight) or @child(airborne). ]
```

```
statemachine FlightAnalyzer initial = beforeFlight {
```

```
  in next(Trackpoint* tp) <no binding>
```

```
  in reset() <no binding>
```

```
  out 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 beforeFlight
```

```
// [ This represents the state in which the airplane flies.
```

```
  It has several substates. Note how it uses the @top(VERY_HIGH_SPEED)  
  and @top(HIGH_SPEED) constants. These constants are defined in the  
  same module @module(StateMachines). ]
```

```
  state airborne {
```

```
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
```

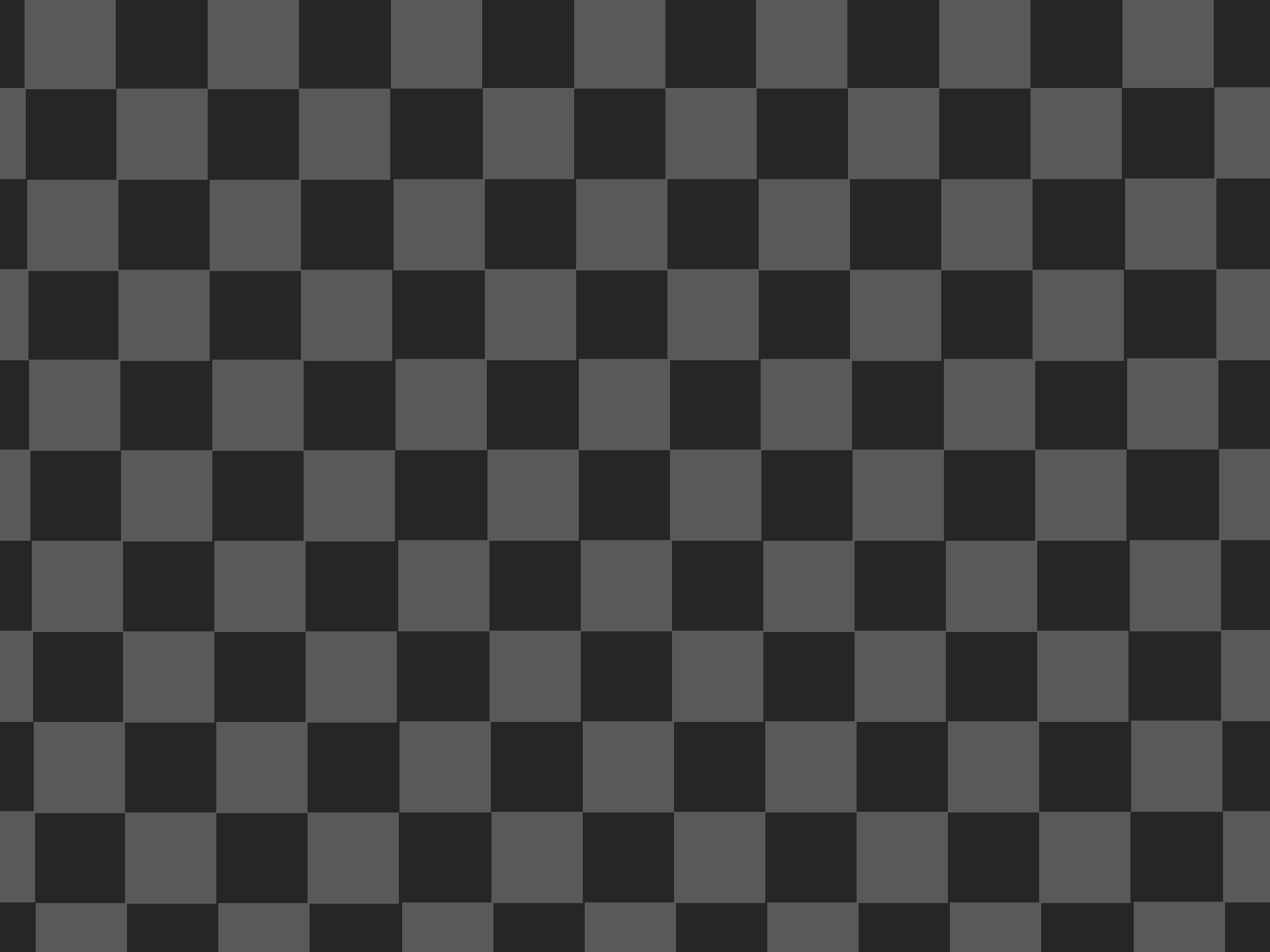
```
    on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
```

```
    [on next [tp->speed > 200 mps && tp->alt == 0 m] -> airborne { points += VERY_HIGH_SPEED; }]-> implements FasterThan200
```

```
    [on next [tp->speed > 100 mps && tp->speed <= 200 mps && tp->alt == 0 m] -> airborne]-> implements FasterThan100  
      { points += HIGH_SPEED; }
```

```
    on reset [ ] -> beforeFlight
```

```
  } state airborne
```



# **Key Takeaway Point**

**Tool Extension is not enough!**

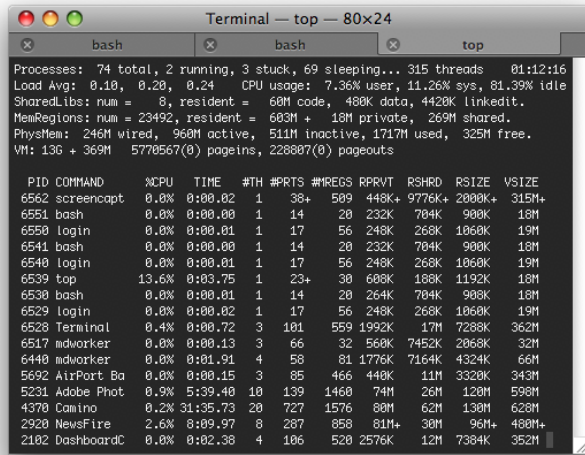
# Key Takeaway Point

**Tool Extension is not enough!**

**Focus on the  
data first!**

# Key Takeaway Point

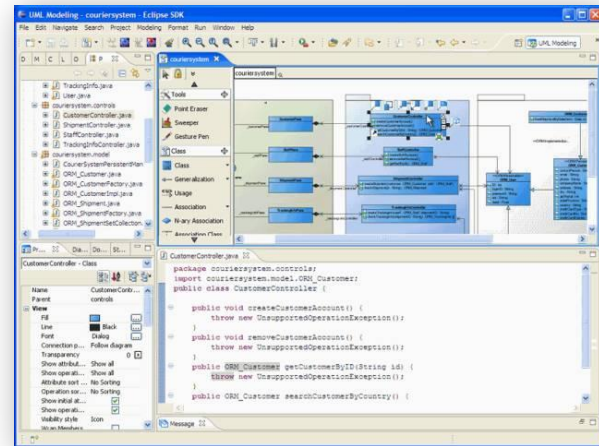
## Tool Extension is not enough!



Terminal — top — 80x24

```
Processes: 74 total, 2 running, 3 stuck, 69 sleeping... 315 threads 01:12:16
Load Avg: 0.10, 0.20, 0.24 CPU usage: 7.36% user, 11.26% sys, 81.39% idle
SharedLibs: num = 8, resident = 60M code, 480K data, 4420K linkedit.
MemRegions: num = 23492, resident = 603M + 19M private, 269M shared.
PhysMem: 246M wired, 960M active, 511M inactive, 1717M used, 325M free.
VM: 13G + 369M 5770567(0) pageins, 226887(0) pageouts
```

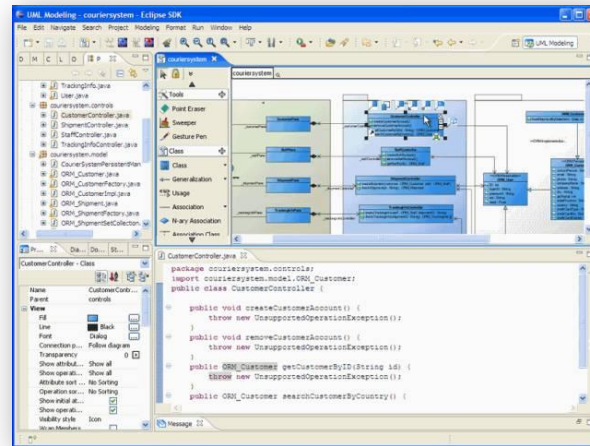
PID	COMMAND	%CPU	TIME	#TH	#RPTS	#REGS	RPRVT	RSHRD	RSIZE	VSIZE
6562	screencapt	0.0%	0:00.02	1	38+	509	448K+	9776K+	2000K+	315M+
6551	bash	0.0%	0:00.00	1	14	20	232K	704K	900K	18M
6550	login	0.0%	0:00.01	1	17	56	248K	268K	1060K	19M
6541	bash	0.0%	0:00.00	1	14	20	232K	704K	900K	18M
6540	login	0.0%	0:00.01	1	17	56	248K	268K	1060K	19M
6539	top	13.6%	0:03.75	1	23+	30	608K	188K	1192K	19M
6530	bash	0.0%	0:00.01	1	14	20	264K	704K	900K	18M
6529	login	0.0%	0:00.02	1	17	56	248K	268K	1060K	19M
6528	Terminal	0.4%	0:00.72	3	101	559	1992K	17M	7288K	362M
6517	mdworker	0.0%	0:00.13	3	66	32	560K	7452K	2066K	32M
6440	mdworker	0.0%	0:01.91	4	58	81	1776K	7164K	4324K	66M
5692	AirPort Ba	0.0%	0:00.15	3	85	466	440K	11M	3320K	343M
5231	Adobe Phot	0.9%	5:39.40	10	139	1460	74M	26M	120M	596M
4370	Camino	0.2%	31:35.73	20	727	1576	80M	62M	130M	626M
2920	NewsFire	2.6%	8:09.97	8	287	858	81M+	30M	96M+	480M+
2102	DashboardC	0.0%	0:02.38	4	106	520	2576K	12M	7384K	352M



# These both do not explicitly support

# Key Takeaway Point

**Tool Extension is not enough!**



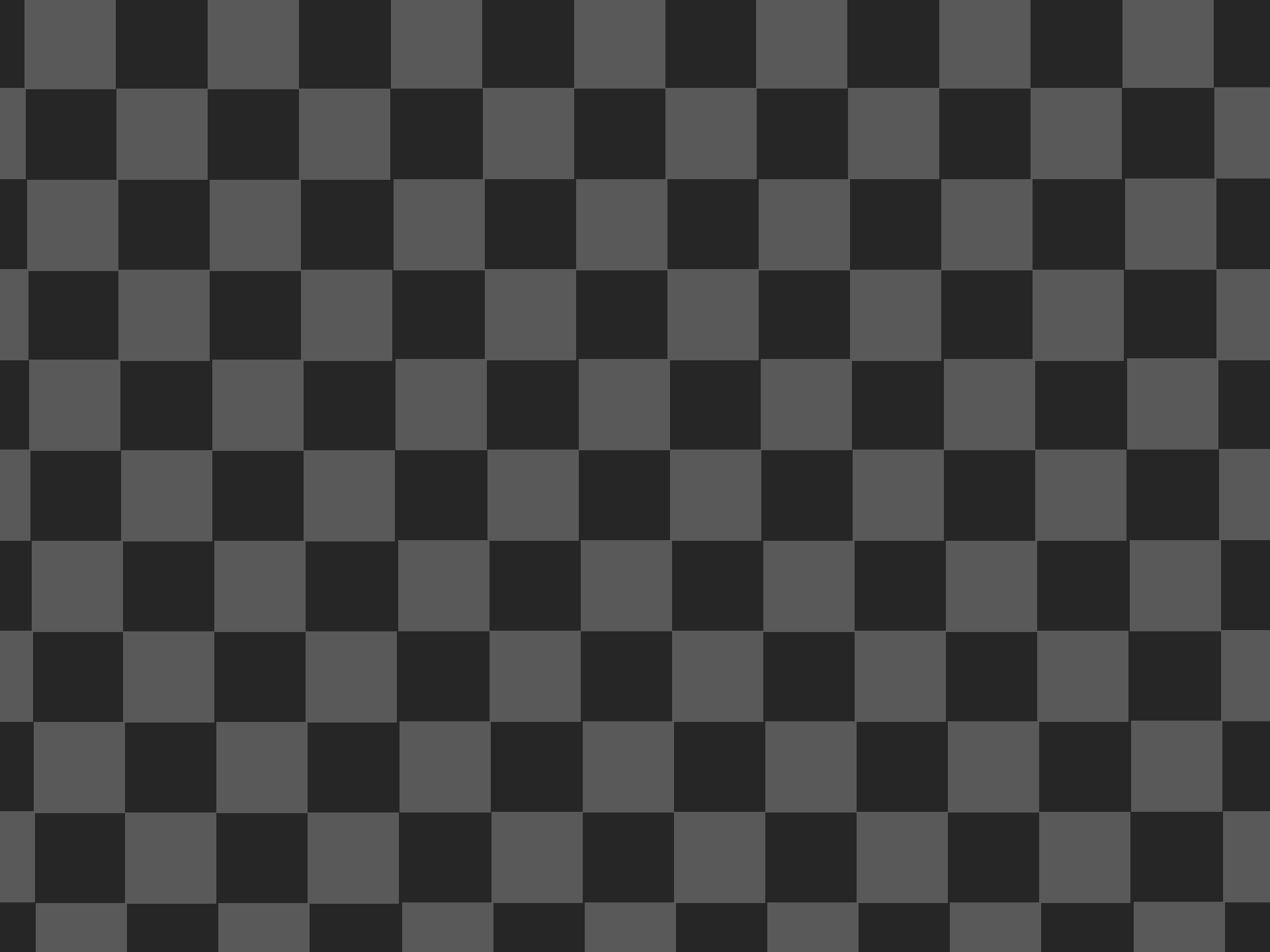
**Relatively high effort to  
reimplement editors**



# Key Takeaway Point

**Tool Extension is not enough!**

**Focus on the  
data first!**





Generic Tools **G T S L** Specific Language

# Thought Process

## From Data Formats To Languages

# Thought Process

## From Data Formats To Languages

# Structure, Constraints, Semantics

---

## Data Format

# Thought Process

## From Data Formats To Languages

Structure, Constraints, Semantics

---

**Data Format + Syntax**

---

**Language**

# **Thought Process**

## **From Data Formats To Languages**

### **Languages**

**Thought Process**

**Language Engineering**

**Language Rev** Languages

**Language**

**Modularization**

**Language**

**Composition**



**Thought Process**

**Language Engineering**

**Language Reuse Languages**

**Language**

**Modularization**

**Language**

---

**Composition**  
Language Engineering

**Thought Process**

**Language Engineering**

**Languages**

**Language Engineering**

**Thought Process**

**Language Engineering**

**Languages**

**Language Engineering**

**Text Math Graph**  
**Tables Symbols**  
**s ols Forms**

**Thought Process**

**Language Engineering**

**Languages**

**Language Engineering**

**Text Math Graph**

**Table Symbols**

**sols Forms**

**Syntactic Diversity**

**Thought Process**

**Language Engineering**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**Thought Process**

**Language Workbenches**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**But does this  
really work?**

**Thought Process**

**Language Workbenches**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**But does this**

**~~really work?~~**

**Language Workbenches**

**Thought Process**

**Language Workbenches**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**Language Workbenches**



# **Generic Tools, Specific Languages**

**Ingredients**



**Languages**

**Language Engineering**

**Syntactic Diversity**

**Language Workbenches**

# Generic Tools, Specific Languages

Ingredients



**Specific  
Language  
s**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**Generic  
Tools**

**Language Workbenches**

# Generic Tools, Specific Languages

Ingredients



**Specific  
Language  
s**

**Languages**

**Language Engineering**

**Syntactic Diversity**

**Generic  
Tools**

**Language Workbenches  
(we don't have to  
reimplement editors  
and synchronizers)**

# Generic Tools, Specific Languages

Ingredients

Specific  
Languages

Languages

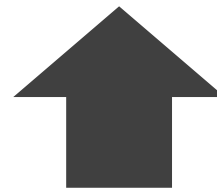
Language Engineering

Syntactic Diversity

support

Generic  
Tools

Language Workbenches



# Language Workbenches

## Typical Features



# Language Workbenches


## Typical Features



**Language Definition,  
Reuse, Extension,  
Composition**

# Language Workbenches


## Typical Features



**Language Definition,  
Reuse, Extension,  
Composition  
Mixing Notations**

# Language Workbenches

## Typical Features



**Language Definition,  
Reuse, Extension,  
Composition  
Mixing Notations  
Type Systems,  
Constraints,  
Transformation,  
Interpretation**



# Language Workbenches

## Typical Features

**Goto Definition/Find  
Usages**



# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Usage Markup/Quick**

**Fixes**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Browser/Markup/Quick**

**Fixes**

**Syntax Highlighting**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Error Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Browser/Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Browser/Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace**

**Refactoring**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Image Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace**

**Refactoring**

**Debugging**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Error Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace/Reporting**

**Refactoring**

**Debugging**



# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Error Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace/Reporting**

**Refactoring Visualization**

**Debugging**

# Language Workbenches

## Typical Features

**Goto Definition/Find**

**Error/Markup/Quick**

**Fixes**

**Syntax Highlighting**

**Code Completion**

**Search/Replace/Reporting**

**Refactoring Visualization**

**Debugging**

**Version**

**Control**

# Language Workbenches


## Typical Features



**for *any*  
Language!**

# Language Workbenches

## Typical Features



**Language  
Workbenches are  
IDEs for arbitrary  
languages.**

**Languages / Language Extensions**

**Contribute Customizations**

漢  
字  
漢  
字

# Languages / Language Extensions

## Contribute Customizations

漢  
字  
漢  
字

**Language Definition,  
Reuse, Extension,  
Composition**

**Mixing Notations  
Type Systems,**

**Constraints,**

**Transformation,**

**Interpretation**

# Languages / Language Extensions

Contribute Customizations

Goto Definition/Find

Error Markup/Quick

Fixes

Syntax Highlighting

Code Completion

Search/Replace/Reporting

Refactoring/Visualization

Debugging

Version

Control

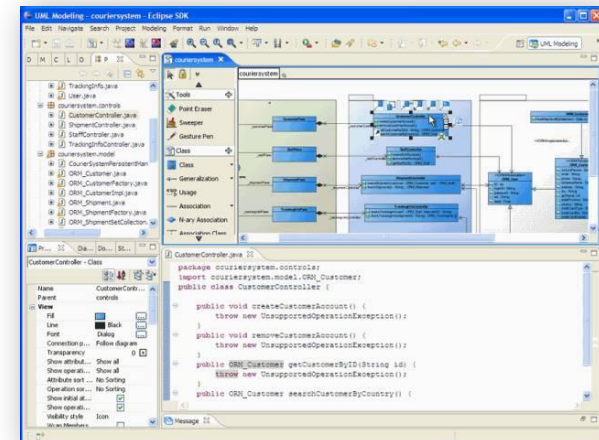
漢  
字  
漢  
字

# Languages / Language Extensions

## Additional Stuff

漢  
字  
漢  
字

Buttons Views  
Menus Actions





# Tool Extensibility

## Study Findings I

The majority of our interviewees were very successful with MDE but all of them either built their own modeling tools, made heavy adaptations of off-the-shelf tools, or spent a lot of time finding ways to work around tools. The only accounts of easy-to-use, intuitive tools came from those who had developed tools themselves for bespoke purposes. Indeed, this suggests that current tools are a barrier to success rather than an enabler.

# Tool Extensibility

## Study Findings I

The majority of our interviewees were very successful with MDE but all of them either **built their own** modeling tools, **made heavy adaptations** of off-the-shelf tools, or spent a lot of time finding ways to work around tools. The only accounts of easy-to-use, intuitive tools came from those who had developed tools themselves for bespoke purposes. Indeed, this suggests that current tools are a barrier to success rather than an enabler.

# Tool Extensibility

## Study Findings II

Complexity problems are typically associated with off-the-shelf tools. Of particular note is accidental complexity – which can be introduced due to poor consideration of other categories, such as lack of flexibility to adapt the tools to a company's own context [..]

# Tool Extensibility

## Study Findings II

Complexity problems are typically associated with off-the-shelf tools. Of particular note is **accidental complexity** – which can be introduced due to poor consideration of other categories, such as **lack of flexibility to adapt the tools** to a company's own context [..]

# Language Workbenches

## Typical Features

Used by the tool vendor  
to

**build** the initial tool  
(languages).



# Language Workbenches

## Typical Features

Used by the tool vendor  
to

**build** the initial tool

Used by the end user to  
(language)

**adapt** the tool (lang  
extensions)!



# Language Workbenches

## Typical Features

Used by the tool vendor  
to

**build** the initial tool

~~Used by the~~ end user to  
(language)

**adapt** the tool (lang

extensions)!

~~Extensions~~ are first-

class!



**Generic Tools, Specific Languages**

**Adaptability is built-in!**

**Extensio**

**ns are first-class**



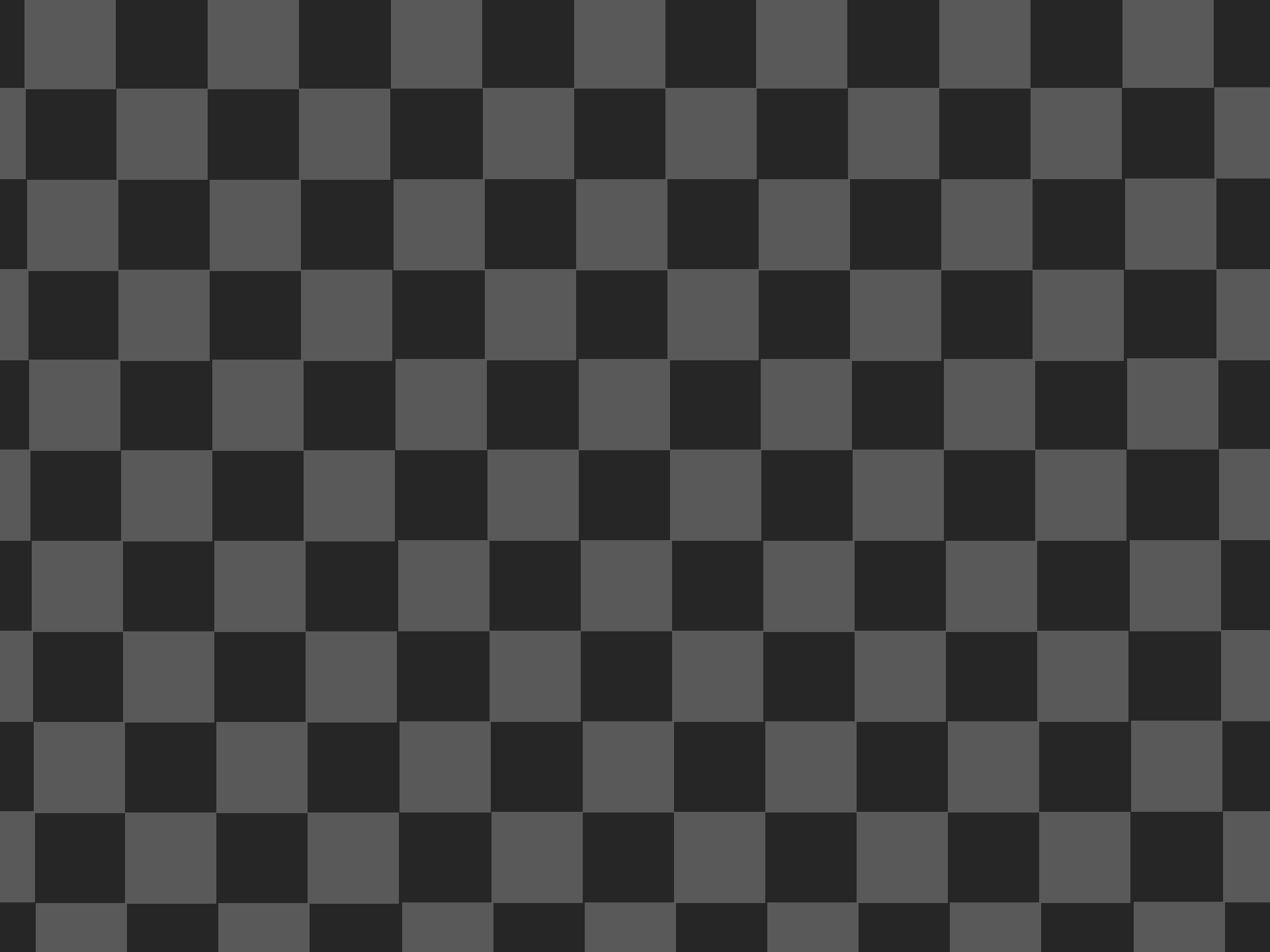
**Generic Tools, Specific Languages**

**Adaptability is built-in!**

**Extensio**

**ns are first-class**

**Fundamentally different from  
Today's State-of-the-Art in Tools**





**An Example**

# An Example System

## Language Engineering Embedded Software



Specific  
Languages

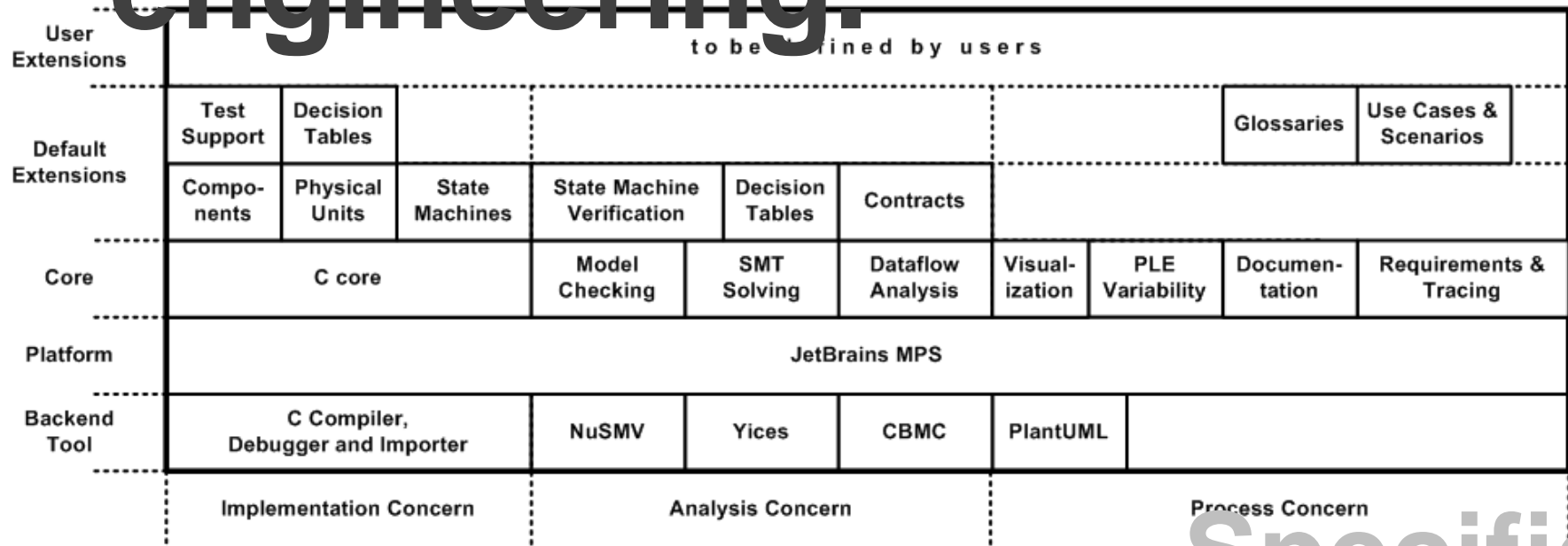
Language Engineering Embedded Software

**A collection of integrated  
for embedded software  
languages  
engineering.**

Specific  
Languages

## Language Engineering Embedded Software

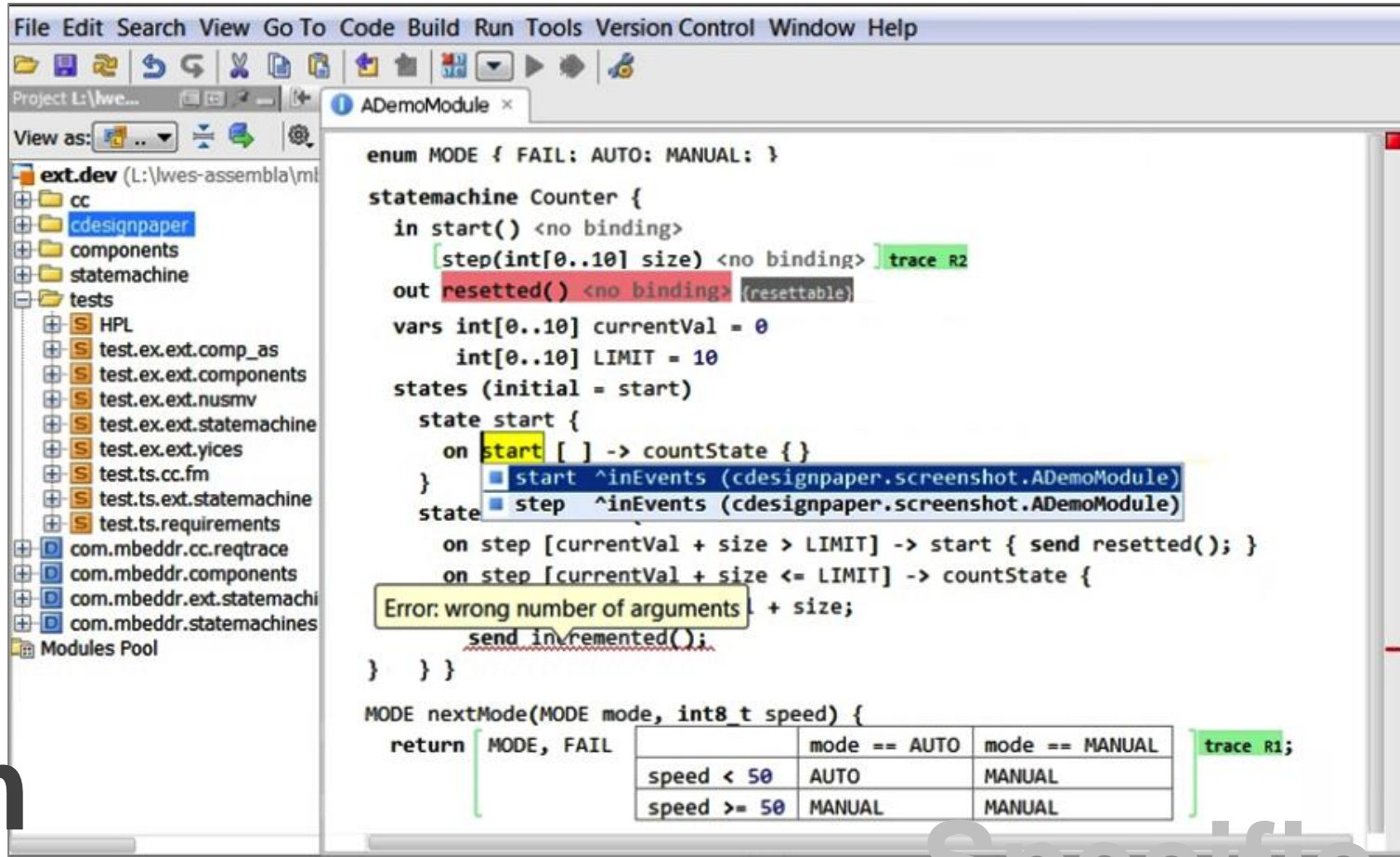
# A collection of integrated languages for embedded software engineering.



Specific  
Languages

## Language Engineering Embedded Software

An  
IDE  
for  
all  
Of  
them



```
enum MODE { FAIL: AUTO: MANUAL: }

statemachine Counter {
  in start() <no binding>
  [step(int[0..10] size) <no binding> trace R2]
  out resetted() <no binding> [resettable]
  vars int[0..10] currentVal = 0
  int[0..10] LIMIT = 10
  states (initial = start)
  state start {
    on start [ ] -> countState {
      start ^inEvents (cdesignpaper.screenshot.ADemoModule)
      step ^inEvents (cdesignpaper.screenshot.ADemoModule)
    }
    on step [currentVal + size > LIMIT] -> start { send resetted(); }
    on step [currentVal + size <= LIMIT] -> countState {
      + size;
      send incremented();
    }
  }
}

MODE nextMode(MODE mode, int8_t speed) {
  return [ MODE, FAIL
```

	mode == AUTO	mode == MANUAL
speed < 50	AUTO	MANUAL
speed >= 50	MANUAL	MANUAL

trace R1;

Specific

languages

**Open Source  
Eclipse Public License**

**<http://mbeddr.com>**

**itemis fortiss**



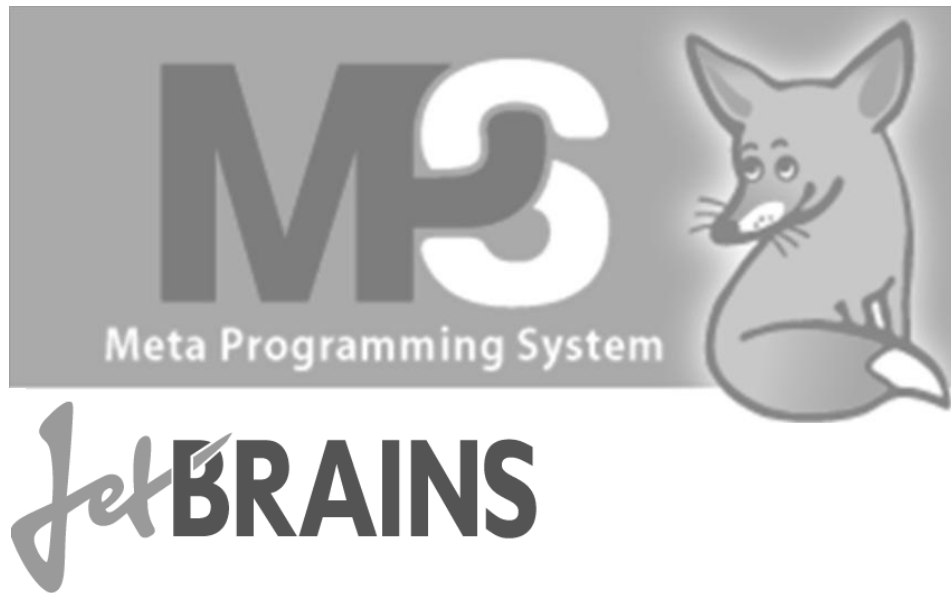
Bundesministerium  
für Bildung  
und Forschung

**Specific  
Languages**



# An Example System

**Built on JetBrains MPS**



**Generic Tool**

# An Example System

## Built on JetBrains MPS



Projectional Editing

Textual/Symbolic/Tabular/(soon Graphical)

Multiple projections for the same language  
(in 3.0, due soon)

Modular language development, extension  
and embedding

# Generic Tool

# An Example System

## Built on JetBrains MPS



Support for language aspects such as type system, scopes, code completion, find usages, dataflow

Template-based approach for transformation and code generation with IDE support for target language in templates

Support for building extensible debuggers

# Generic Tool

# An Example System

## Built on JetBrains MPS



```
enum MODE { FAIL: AUTO: MANUAL: }

statemachine Counter {
  in start() <no binding>
  [step(int[0..10] size) <no binding> trace R2
  out resetted() <no binding> {resettable}]
  vars int[0..10] currentVal = 0
    int[0..10] LIMIT = 10
  states (initial = start)
    state start {
      on start [ ] -> countState {
        start ^inEvents (cdesignpaper.screenshot.ADemoModule)
        step ^inEvents (cdesignpaper.screenshot.ADemoModule)
      }
      on step [currentVal + size > LIMIT] -> start { send resetted(); }
      on step [currentVal + size <= LIMIT] -> countState {
        Error: wrong number of arguments + size;
        send incremented();
      }
    }
  }

  MODE nextMode(MODE mode, int8_t speed) {
    return [
      mode == AUTO | mode == MANUAL |
      speed < 50 | AUTO | MANUAL |
      speed >= 50 | MANUAL | MANUAL |
    ] trace R1;
  }
}
```

# Generic Tool

# **An Example System**

**Built on JetBrains MPS**



*Jet*BRAINS

**Open Source  
Apache 2.0**

**<http://jetbrains.com/mps>**

**Generic Tool**

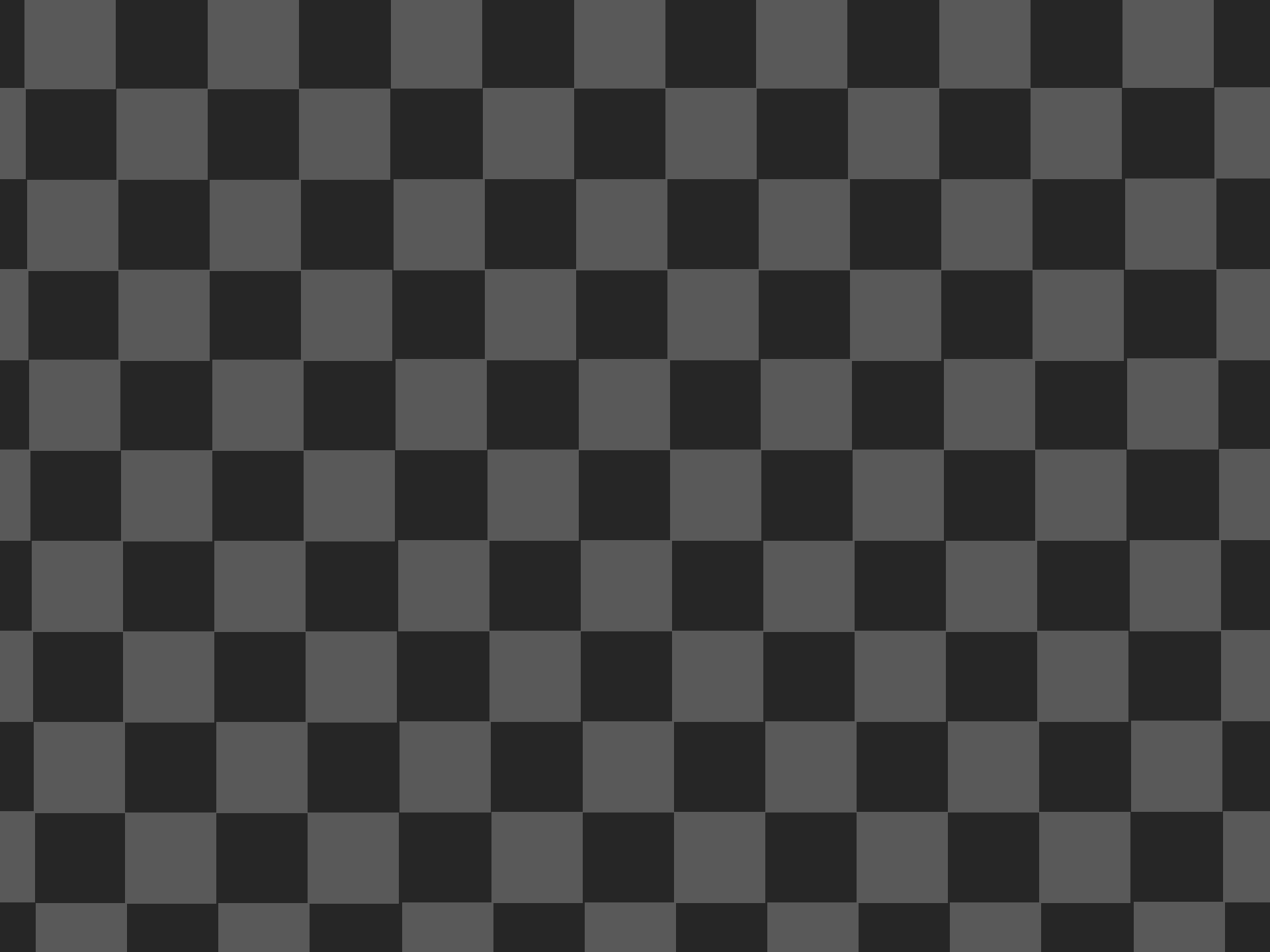


**DEMO**











6

Summing up

# Summing Up

## Key Points

**To build meaningful tools, the data must be extended.**

Extending the tool  
(views, buttons, etc.) is not

# Summing Up

## Key Points

**Structured Data can  
be expressed as  
languages.**

**Languages are data  
plus syntax**

# Summing Up

## Key Points

**Language Engineering  
supports extension  
and composition**

**This supports  
for specific domains  
adapting tools**

# Summing Up

## Key Points

**IDE-style tools are very good for editing data/programs.**

**We've got a lot of experience from regular**

# Summing Up

## Key Points

**Language Workbenches  
are the key enabling  
technology.**

**MPS is IMHO the most  
powerful, but it's not the**

# Summing Up

## Key Points

**Let's build new classes  
of tools!**

... which make  
meaningful extensibility a reality





**The End.**



**voelter.de**  
**dslbook.org**  
**mbeddr.com**  
**jetbrains.com/mps**

# The End.