Domain-Specific Languages Projectionnal Edition

Sébastien Mosser (UNS, I3S)

JDEV,Bordeaux, 2 Juillet 2015

pictures: sxc.hu

Sébastien Mosser

"geek, snowboarder & composition boy"

- Since 2012 : Associate Professor, UNS
- 11-12 : Research Scientist, SINTEF (NO)
- 10-11 : Postdoc, Inria Lille Nord-Europe
- 2010 : PhD Thesis

Team work

2 to 3 persons

Practicing is an essential part of this workshop

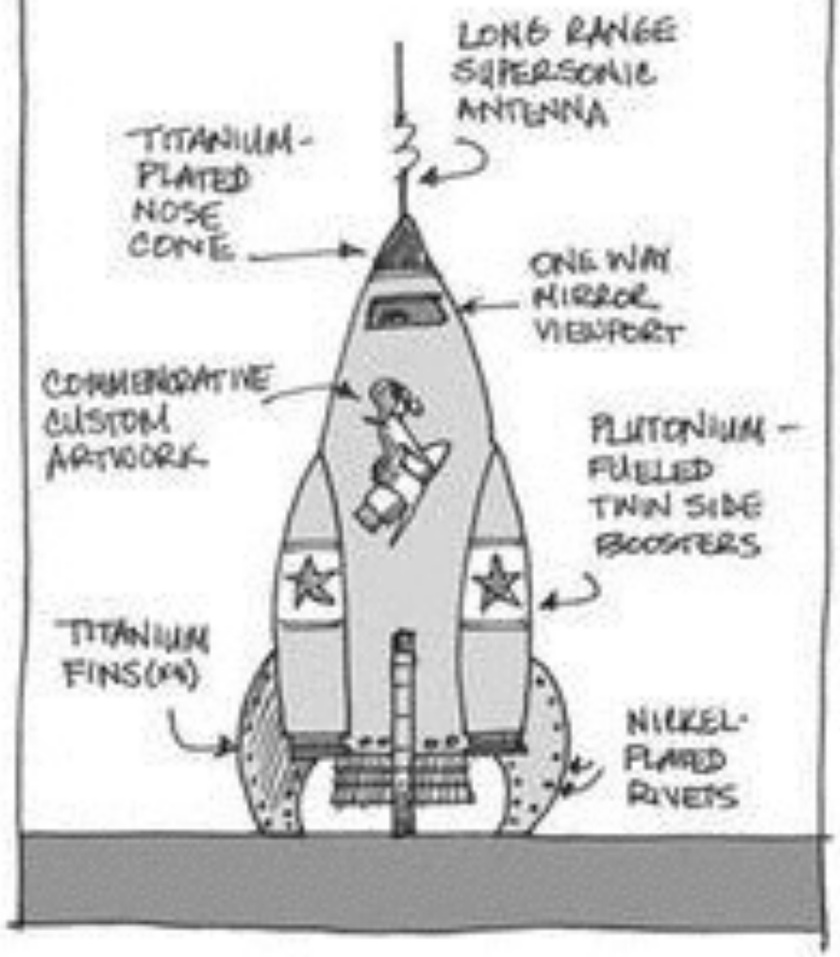
Download MPS

(right now)

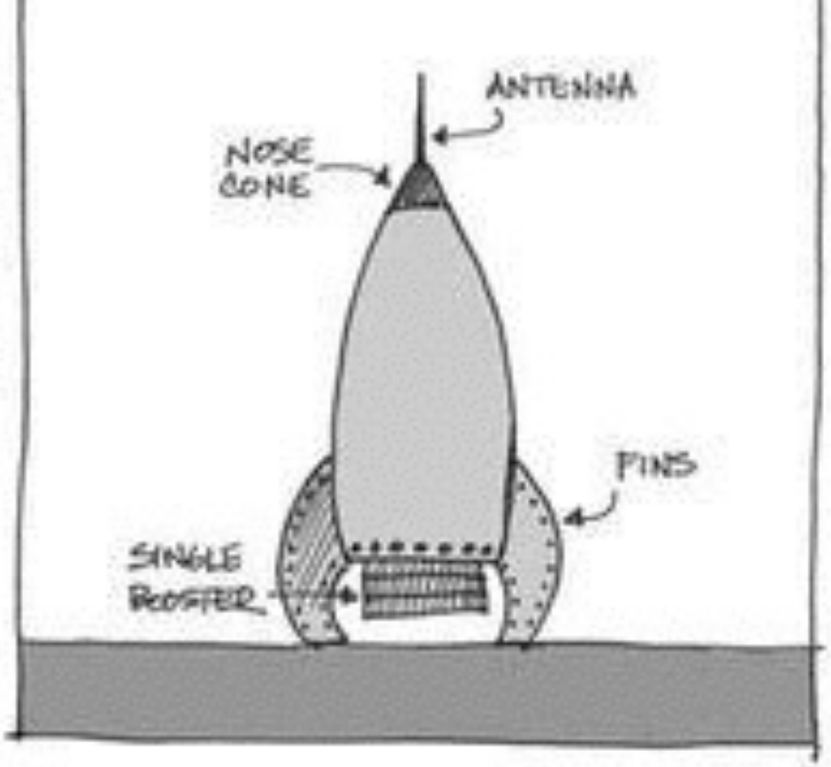
https://www.jetbrains.com/mps/download/

THE UX DESIGNER PARADOX

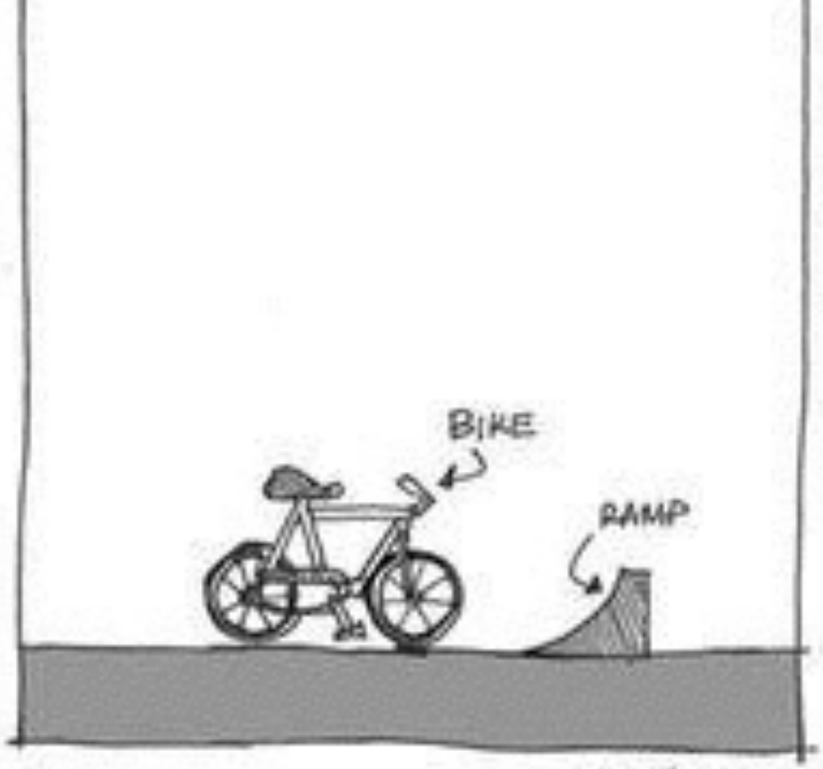
WHAT WE DREAM UP AT KICKOFF



WHAT WE SETTLE FOR AT LAUNCH

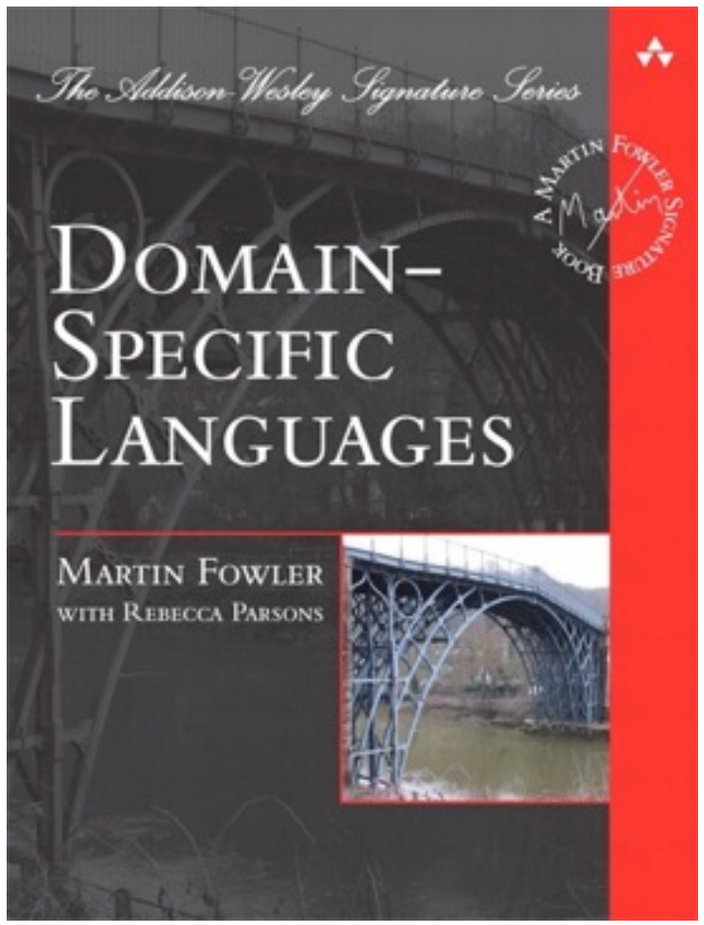


WHAT THE USER NEEDS

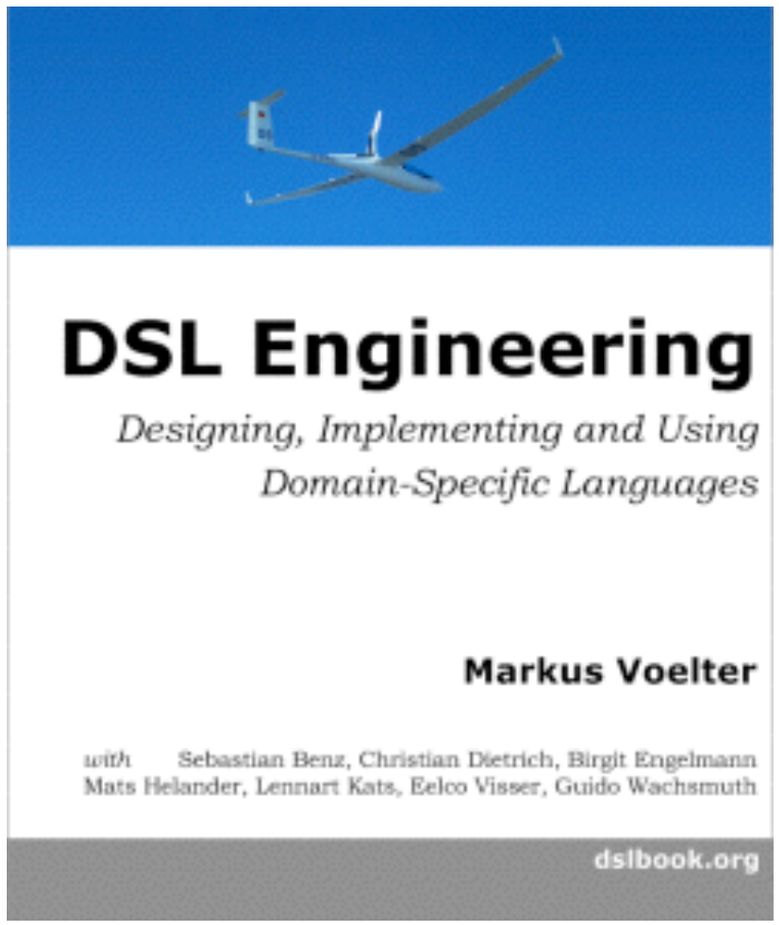


BONUS 2015

Bibliography



[Fowler]



[Voelter]

Domain Specific Language ?

```
SELECT login, pwd
FROM users
WHERE login = «?»
  AND pwd   = «?»
```

```
appliance KIR {

  compressor compartment cc {
    static compressor c1
    fan ccfan
  }

  ambient tempsensor at

  cooling compartment RC {
    light rclight
    superCoolingMode
    door rcdoor
    fan rcfan
    evaporator tempsensor rceva
  }
}
```

```
cooling test for Standardcooling {
  prolog {
    set cc.c1->standstillPeriod = 0
  }
  // initially we are not cooling
  assert-currentstate-is noCooling
  // then we say that RC needs cooling, but
  // the standstillPeriod is still too low.
  mock: set RC->needsCooling = true
  step
  assert-currentstate-is noCooling
  // now we increase standstillPeriod and check
  // if it now goes to rcCooling
  mock: set cc.c1->stehzeit = 400
  step
  assert-currentstate-is rcCooling
}
```

[DSL engineering]

All

¤ 3.3 Commutatiegetallen op 1 leven¶

$$D_x = v^x * \frac{l_x}{100}$$ ¤6 Dec(3)¶

Implemented in ¤ V9401¶

¶

$$N_x = \sum_{t=0}^{\omega - x} D_{x+t}$$ ¤7 Dec(3)

¶

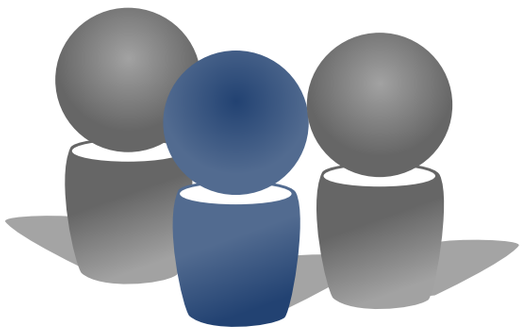
¤ 3.6 Contante waarde 1 leven/ 2 levens¶

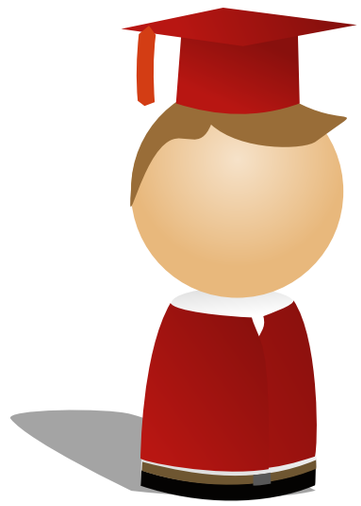
$${}_nE_x = \frac{D_{x+n}}{D_x}$$ ¤19 Dec(4)

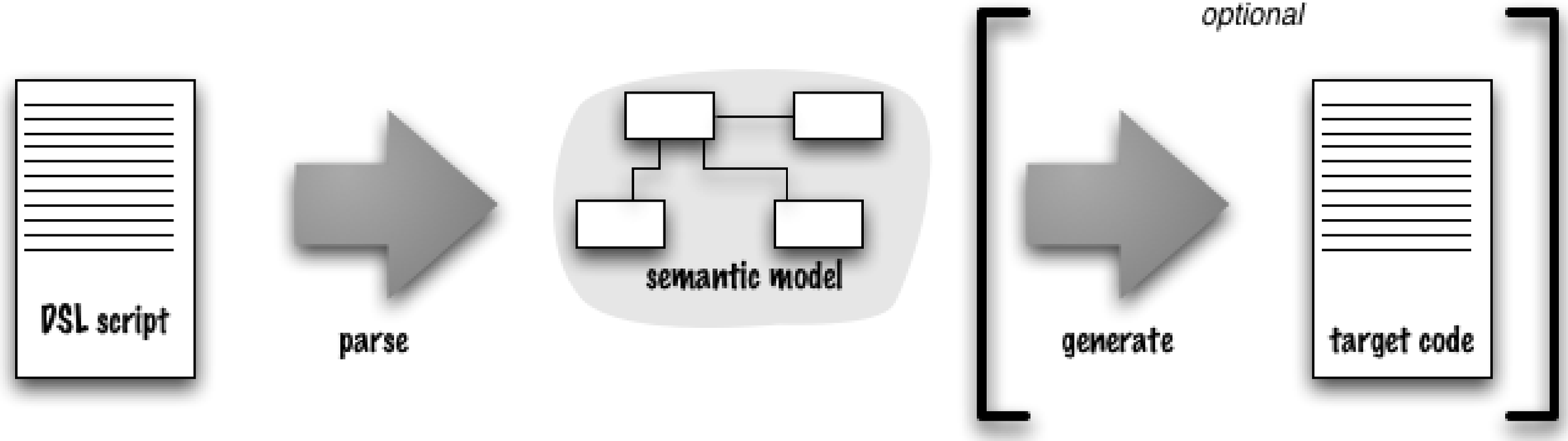
$$\ddot{a}_{xn\rceil} = \frac{N_x - N_{x+n}}{D_x}$$ ¤23 Dec(3)

$$\bar{a}_{xn\rceil} = \ddot{a}_{xn\rceil} - 0{,}5 + 0{,}5 * {}_nE_x$$ ¤25 Dec(3)

[DSL engineering]







[Domain-Specific Languages]

Miss Grant's drawer

Secret Compartments

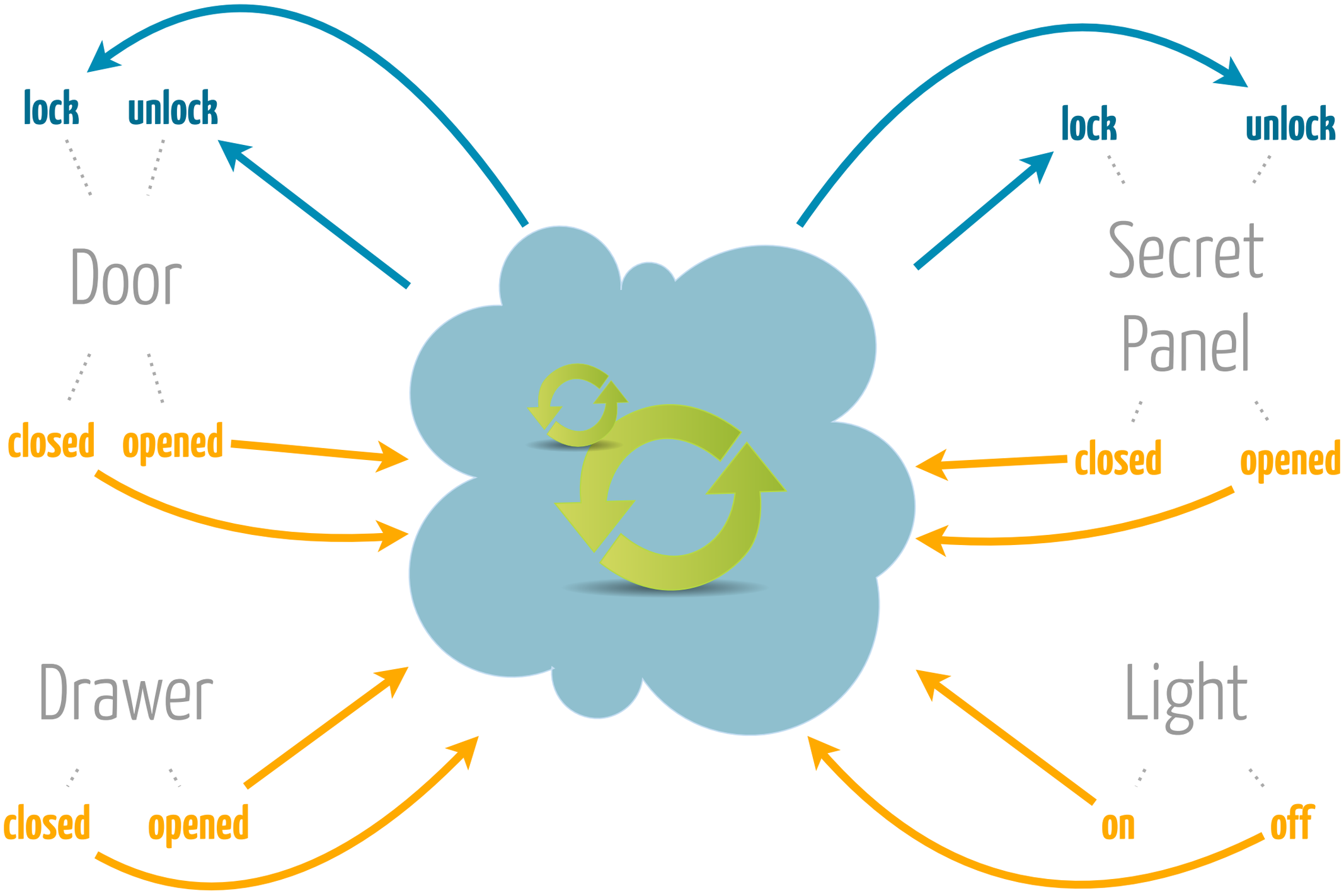
- To open the **Diagon Alley** wall:
  - «From the trash can,
  - Tap three bricks up,
  - and two across.»
- To open the **secret door**:
  - Pull the left candlestick,
  - Push the 3rd brick from the right,
  - Pull Vol. 7 of the Encyclopedia.

Miss Grant's Secret Drawer

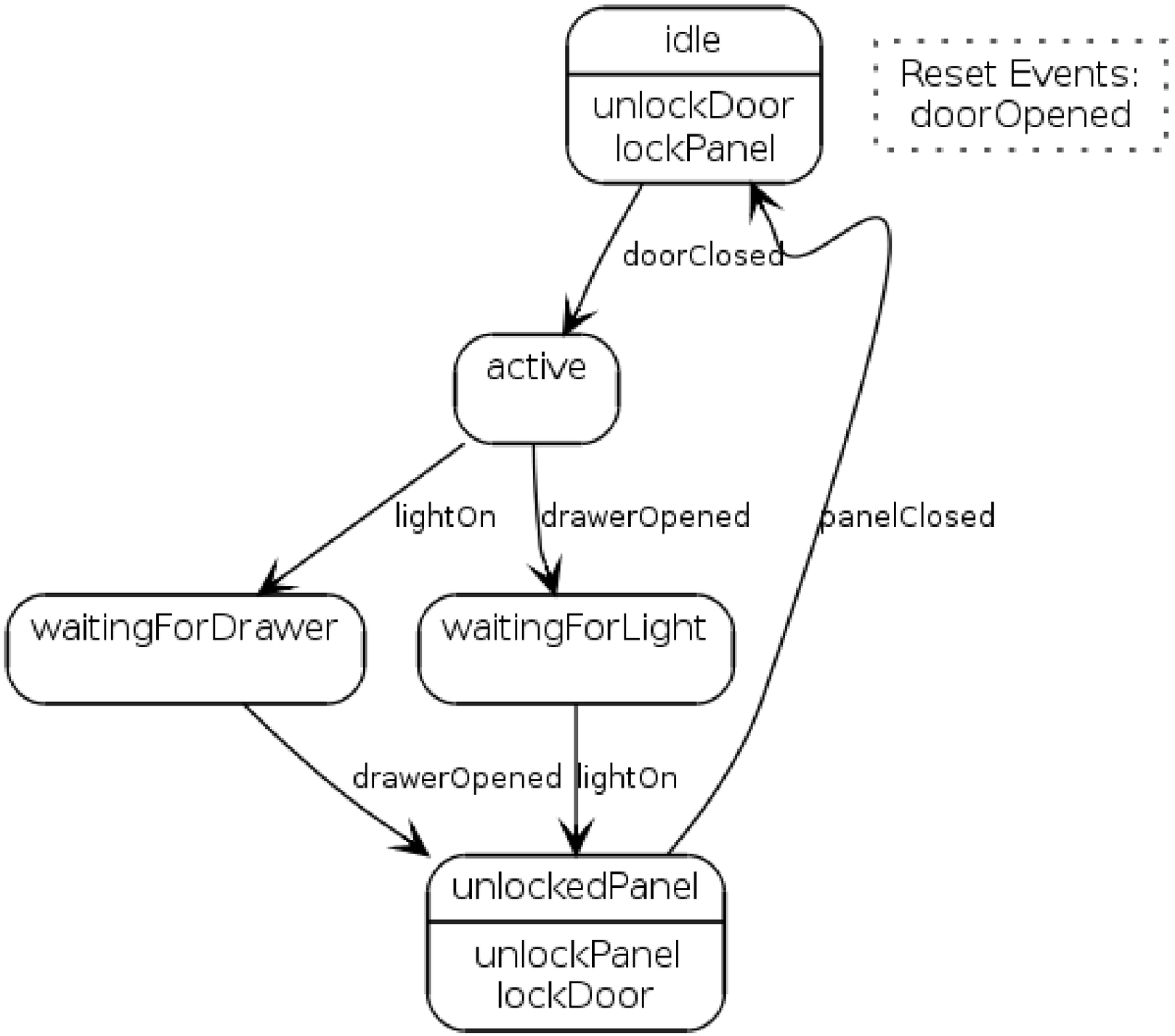
«To **open** it, she has to **close the door**, then **open the second drawer** in her chest and **turn her bedside light on**—in either order. Once these are done, **the secret panel is unlocked** for her to open.»

[Fowler]

Miss Grant's Environment



Formalizing Miss Grant's Controller

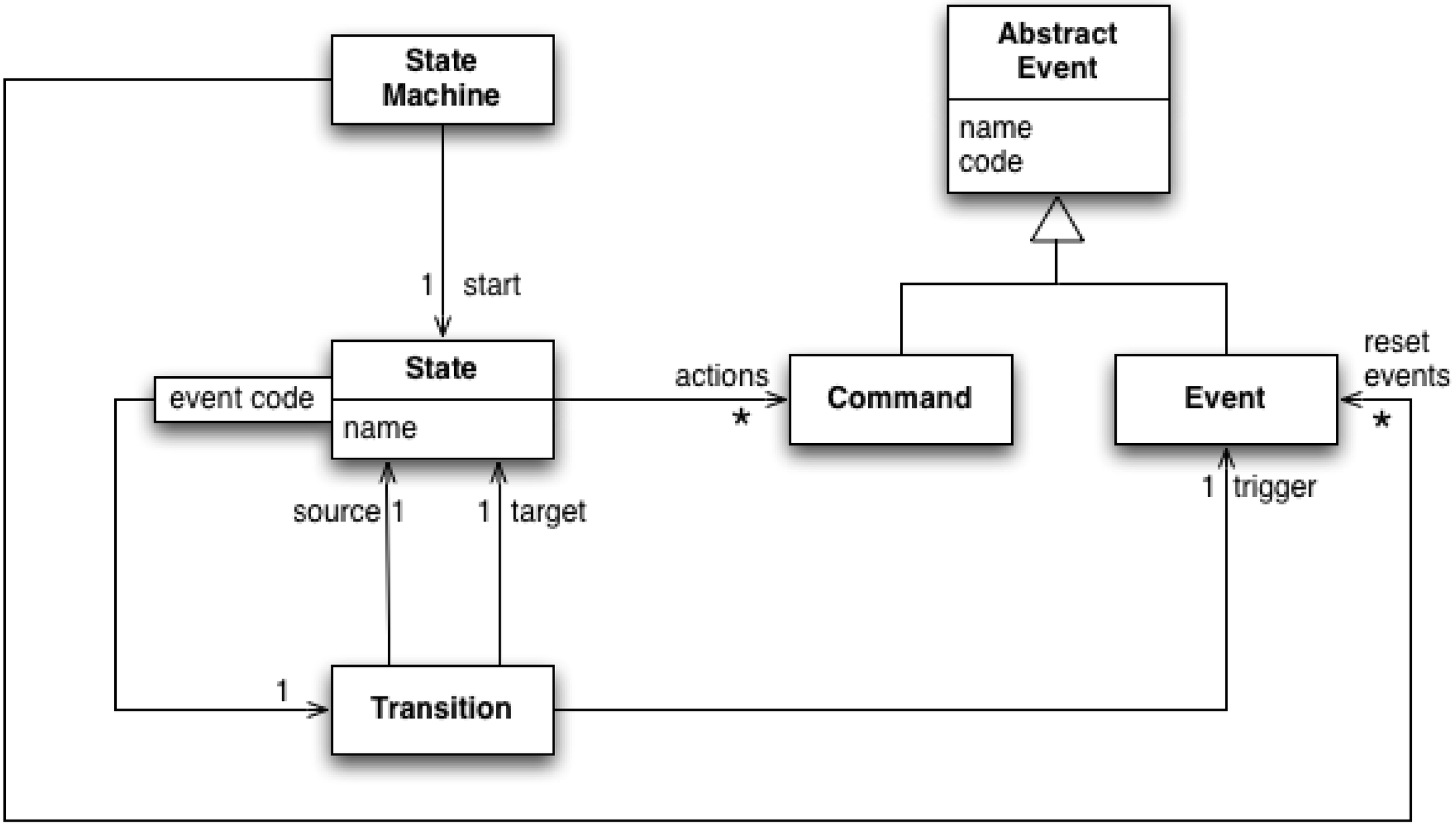


[Fowler]

Modeling Controllers

- **Events**:
  - A code sent **by** the environment
  - Inform about context changes
- **Commands**:
  - A code sent **to** the environment
  - Change the context (e.g., unlock)
- **States**
  - Controller's current situation
  - Might trigger Commands
  - Reacts to events with Transitions
- **Transitions**
  - Associate an event to a next State

Associated Structural Concepts



[Fowler]

Programming the Controller [1/2]

```
Event doorClosed = new Event("doorClosed", "D1CL");
Event drawerOpened = new Event("drawerOpened", "D2OP");
Event lightOn = new Event("lightOn", "L1ON");
Event doorOpened = new Event("doorOpened", "D1OP");
Event panelClosed = new Event("panelClosed", "PNCL");

Command unlockPanelCmd = new Command("unlockPanel", "PNUL");
Command lockPanelCmd = new Command("lockPanel", "PNLK");
Command lockDoorCmd = new Command("lockDoor", "D1LK");
Command unlockDoorCmd = new Command("unlockDoor", "D1UL");


State idle = new State("idle");
State activeState = new State("active");
State waitingForLightState = new State("waitingForLight");

State waitingForDrawerState = new State("waitingForDrawer");
State unlockedPanelState = new State("unlockedPanel");

StateMachine machine = new StateMachine(idle);
```

[Fowler]

Programming the Controller [2/2]

```
idle.addTransition(doorClosed, activeState);
idle.addAction(unlockDoorCmd); idle.addAction(lockPanelCmd);


activeState.addTransition(drawerOpened, waitingForLightState);
activeState.addTransition(lightOn, waitingForDrawerState);
```

```
waitingForLightState.addTransition(lightOn, unlockedPanelState);
waitingForDrawerState.addTransition(drawerOpened,
unlockedPanelState);


unlockedPanelState.addAction(unlockPanelCmd);
unlockedPanelState.addAction(lockDoorCmd);
unlockedPanelState.addTransition(panelClosed, idle);


machine.addResetEvents(doorOpened);
```

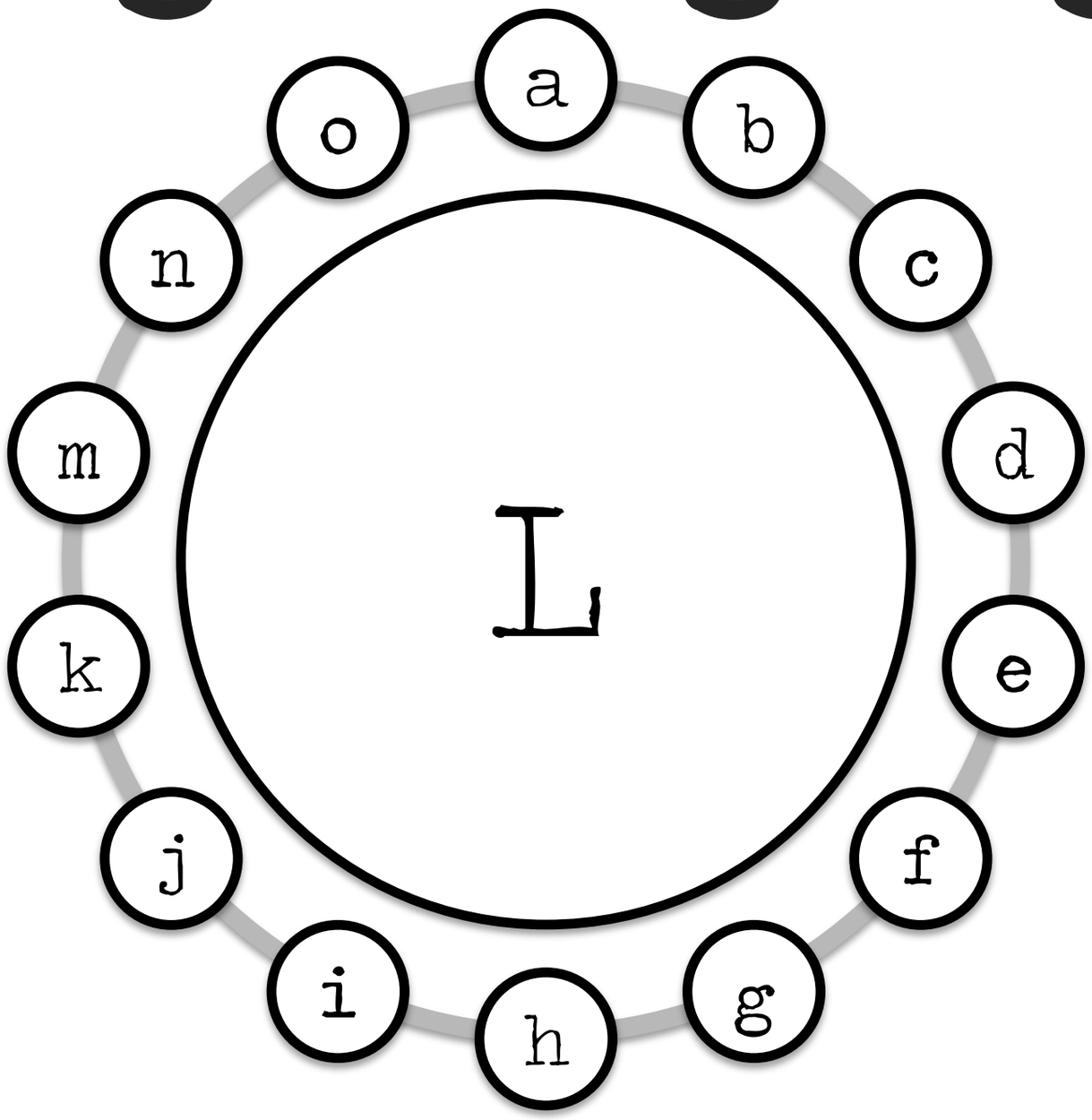
[Fowler]

Wait!

Where the **hell**

is my **domain**?

Big Language



with **many** first class concepts!

[Voelter]

What the hell is my domain?

Controller

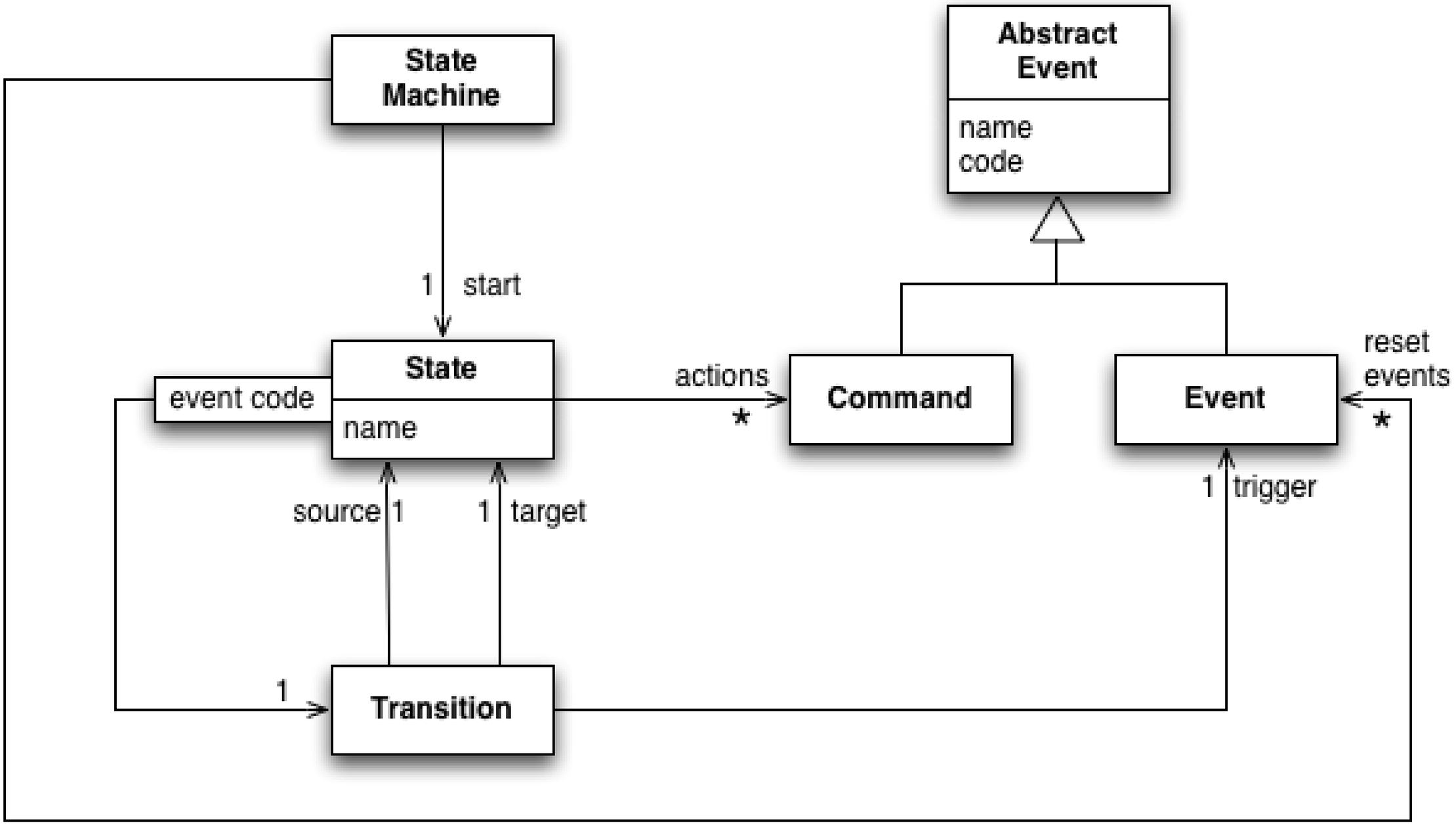
From «**Programming** the **model** of the **Controller**» ...

… to

«Modeling

the Controller»

Associated Structural Concepts



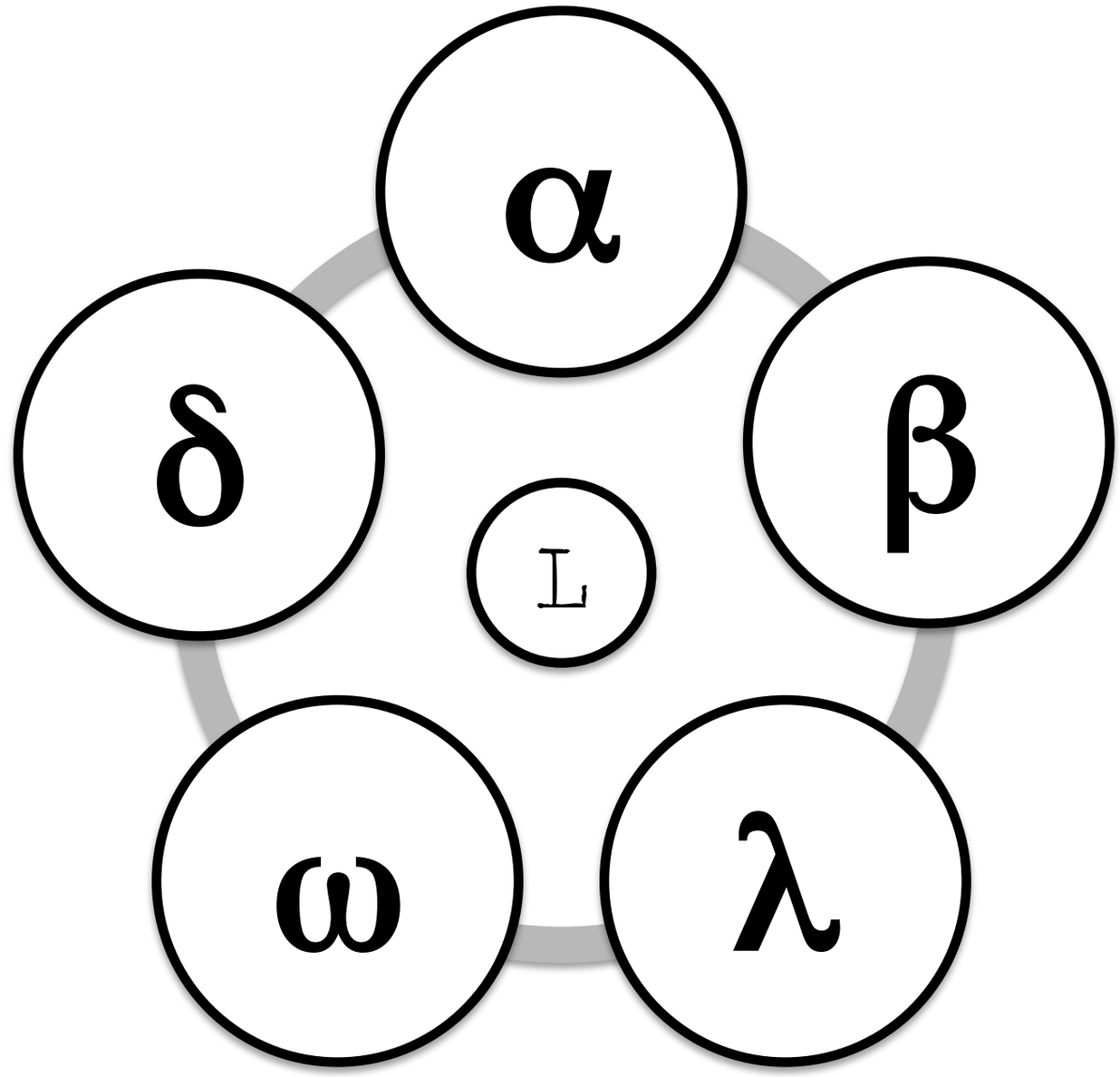
[Fowler]

```
events
  doorClosed      D1CL
  drawerOpened    D2OP
  lightOn         L1ON
  doorOpened      D1OP
  panelClosed     PNCL
end


resetEvents
  doorOpened
end


commands
  unlockPanel PNUL
  lockPanel   PNLK
  lockDoor    D1LK
  unlockDoor  D1UL
end
```

```
state idle
  actions {unlockDoor lockPanel}
  doorClosed => active
end

state active
  drawerOpened => waitingForLight
  lightOn => waitingForDrawer
end

state waitingForLight
  lightOn => unlockedPanel
end

state waitingForDrawer
  drawerOpened => unlockedPanel
end

state unlockedPanel
  actions {unlockPanel lockDoor}
  panelClosed => idle
end
```

[Fowler]

Small Language

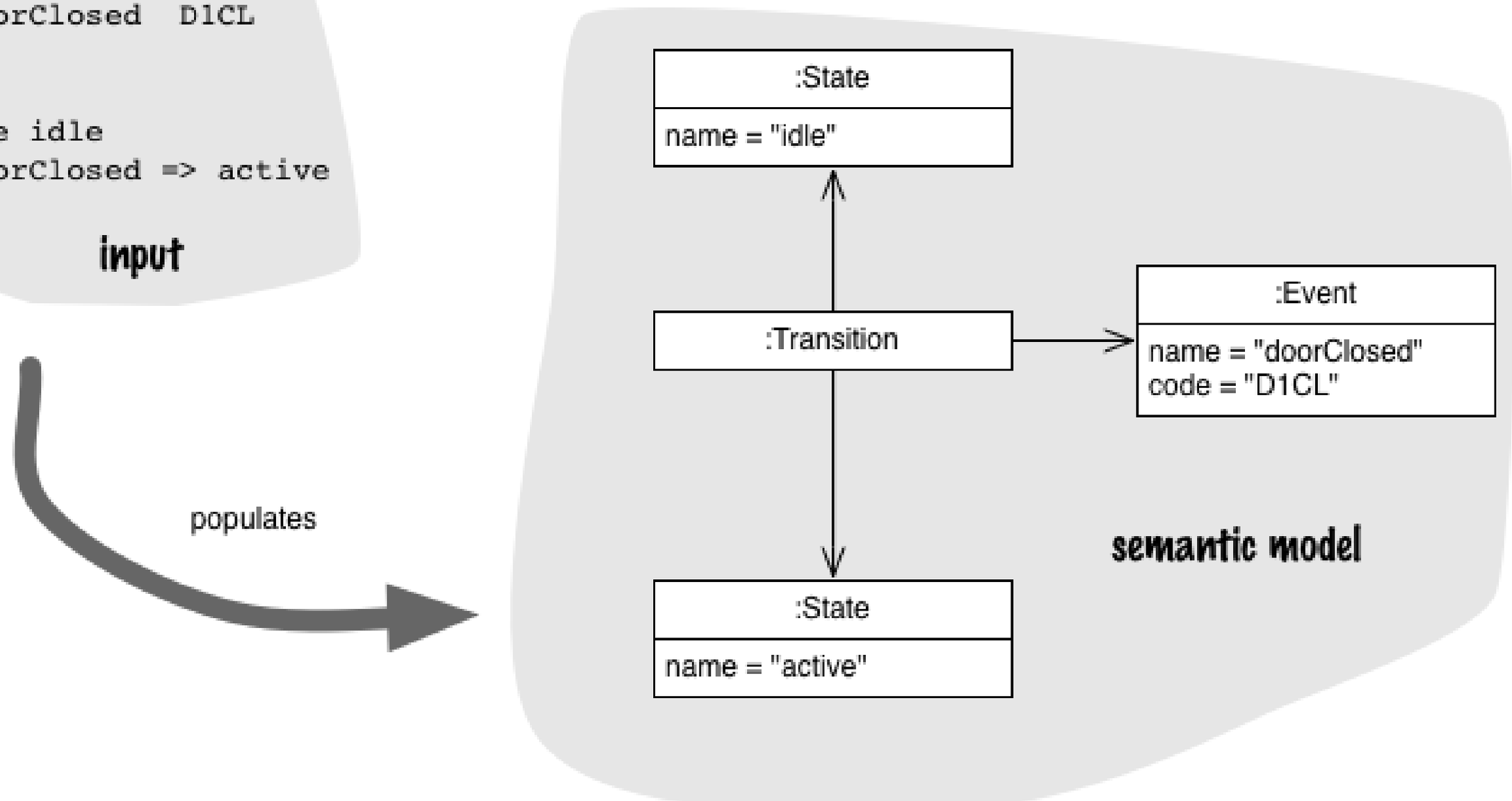


with a few, orthogonal and poweful concepts

[Voelter]

```
events
  doorClosed  D1CL
end

state idle
  doorClosed => active
end
```

input



[Fowler]

External DSLs

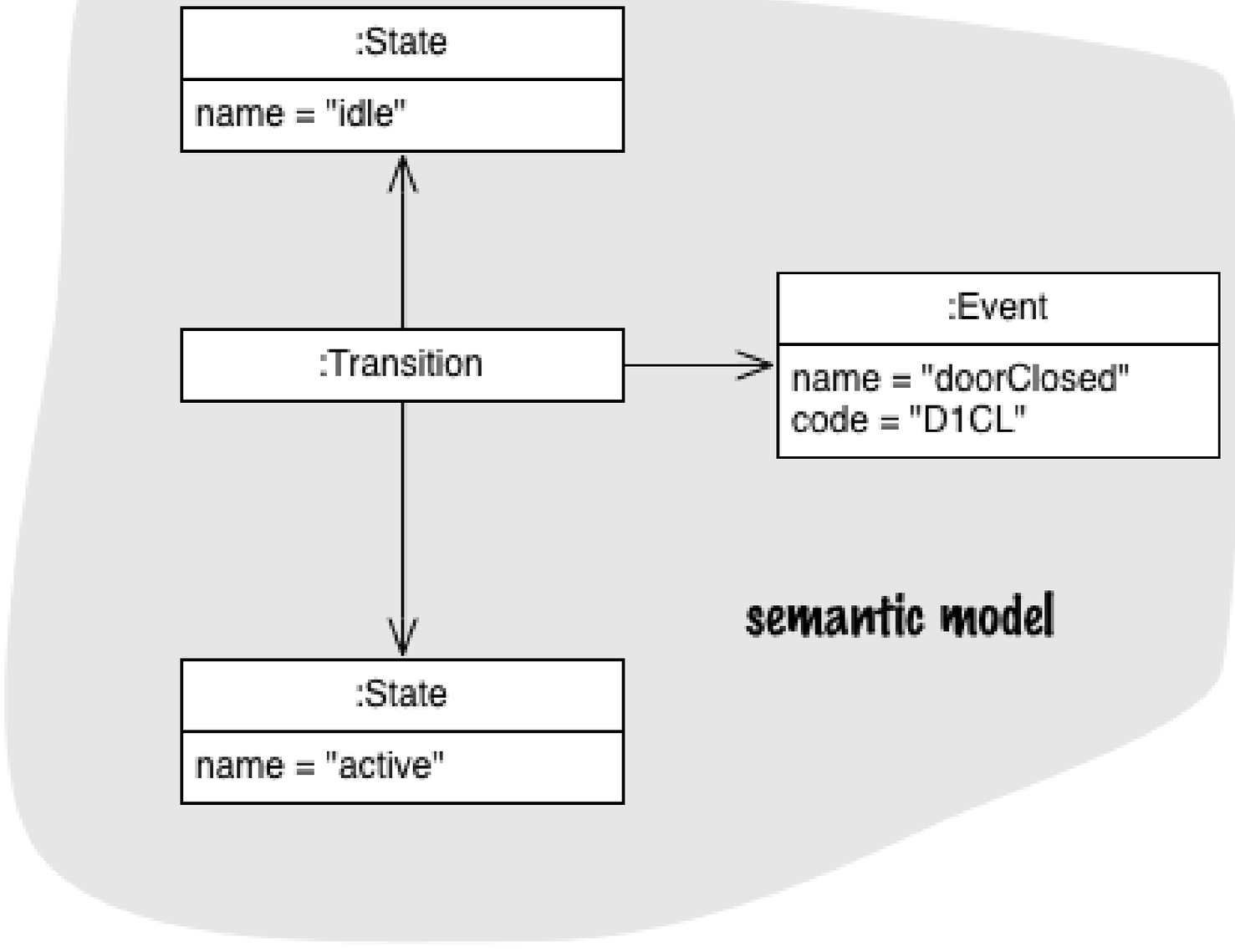
```
events
  doorClosed     D1CL
  drawerOpened   D2OP
  lightOn        L1ON
  doorOpened     D1OP
  panelClosed    PNCL
end

resetEvents
  doorOpened
end

commands
  unlockPanel PNUL
  lockPanel   PNLK
  lockDoor    D1LK
  unlockDoor  D1UL
end
```

```
state idle
  actions {unlockDoor lockPanel}
  doorClosed => active
end

state active
  drawerOpened => waitingForLight
  lightOn => waitingForDrawer
end

state waitingForLight
  lightOn => unlockedPanel
end

state waitingForDrawer
  drawerOpened => unlockedPanel
end

state unlockedPanel
  actions {unlockPanel lockDoor}
  panelClosed => idle
end
```

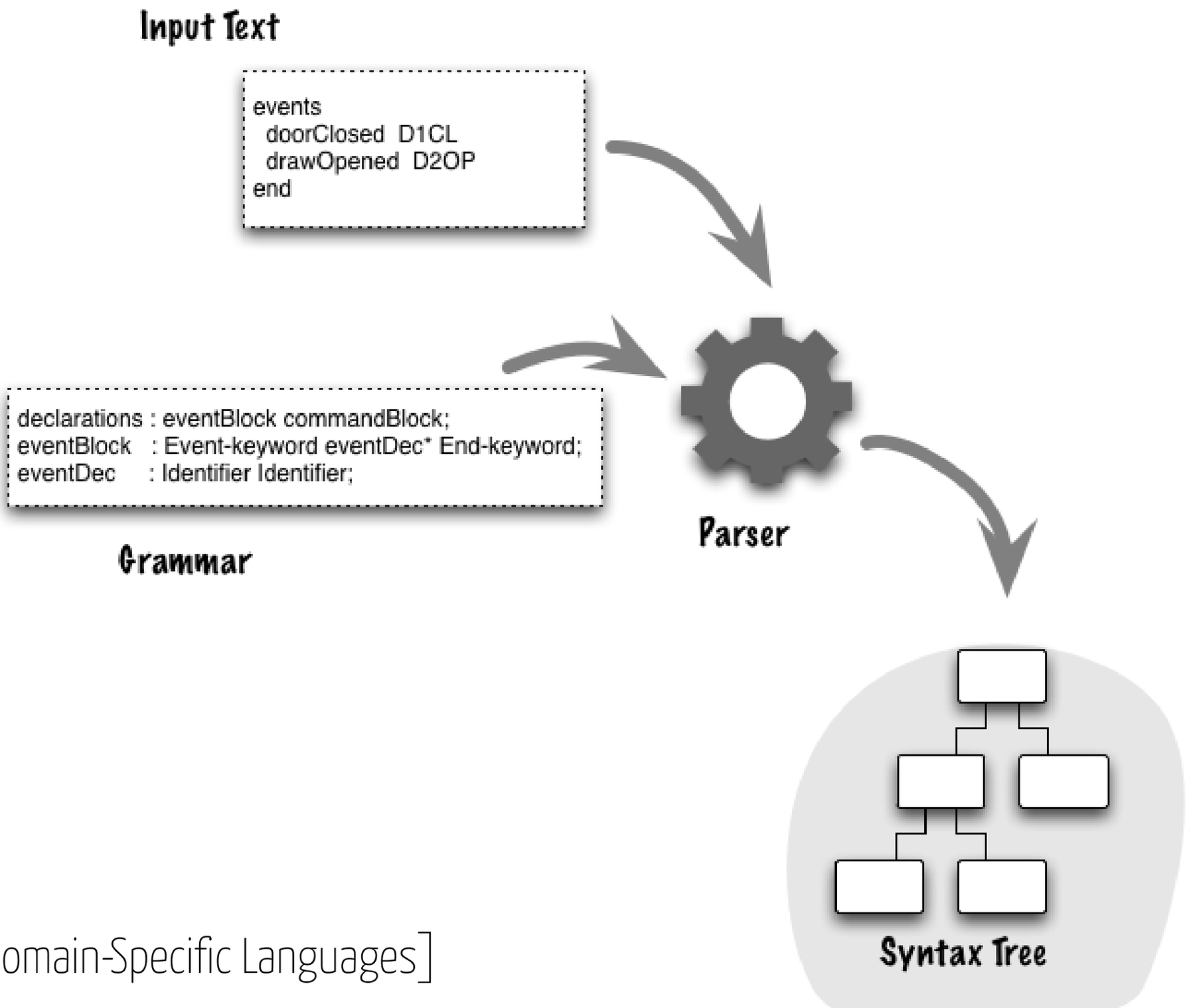
```
events
  doorClosed  D1CL
end

state idle
  doorClosed => active
end
```

input

populates



Syntax-directed Translation



Introducing Grammars

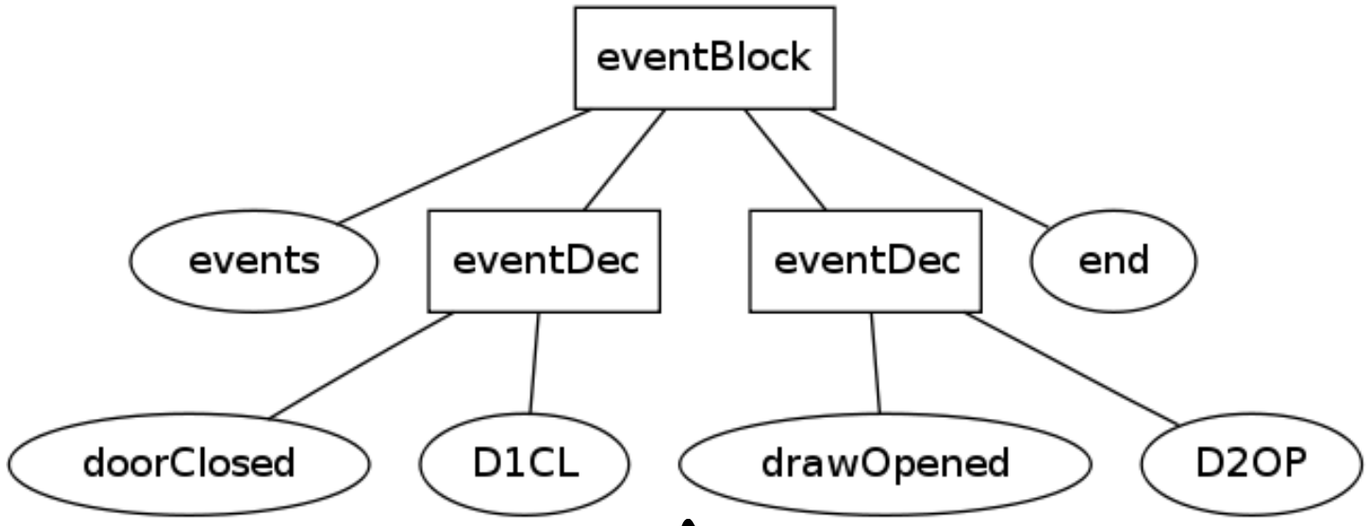
```
declarations : eventBlock commandBlock;

eventBlock : Event-keyword eventDec* End-keyword;

eventDec : Identifier Identifier;

commandBlock : Command-keyword commandDec* End-keyword;

commandDec : Identifier Identifier;
```

This is not a compilation course!

From **Words** to **Parse Tree**

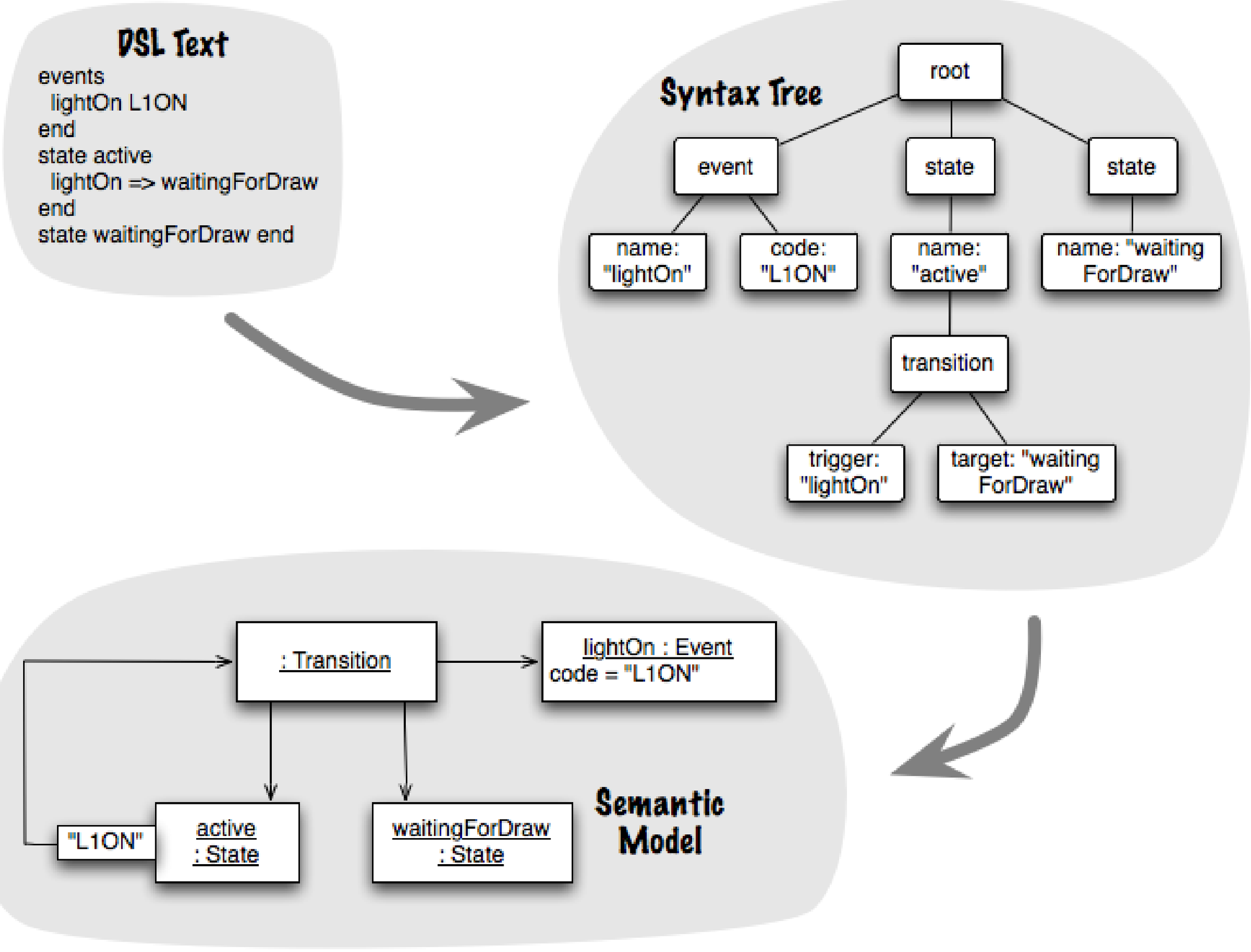
```
events
   doorClosed  D1CL
   drawOpened  D2OP
end
```

lexer



parser

```
[Event-keyword: "events"]
[Identifier: "doorClosed"]
[Identifier: "D1CL"]
[Identifier: "drawOpened"]
[Identifier:"D2OP"]
[End-keyword: "end"]
```


DSL Text
events
lightOn L1ON
end
state active
lightOn => waitingForDraw
end
state waitingForDraw end
Syntax Tree
root
event
state
state
name: "lightOn"
code: "L1ON"
name: "active"
name: "waiting ForDraw"
transition
trigger: "lightOn"
target: "waiting ForDraw"
: Transition
lightOn : Event
code = "L1ON"
"L1ON"
active
: State
waitingForDraw
: State
Semantic Model

	more in GPLs	more in DSL
Domain Size	large and complex	
Designed by	guru or committee	
Language Size	large	
Turing-completeness	almost always	
User Community	large, anonymous and widespread	
In-language abstraction	sophisticated	
Lifespan	years to decades	
Evolution	slow, often standardized	
Incompatible Changes	almost impossible	

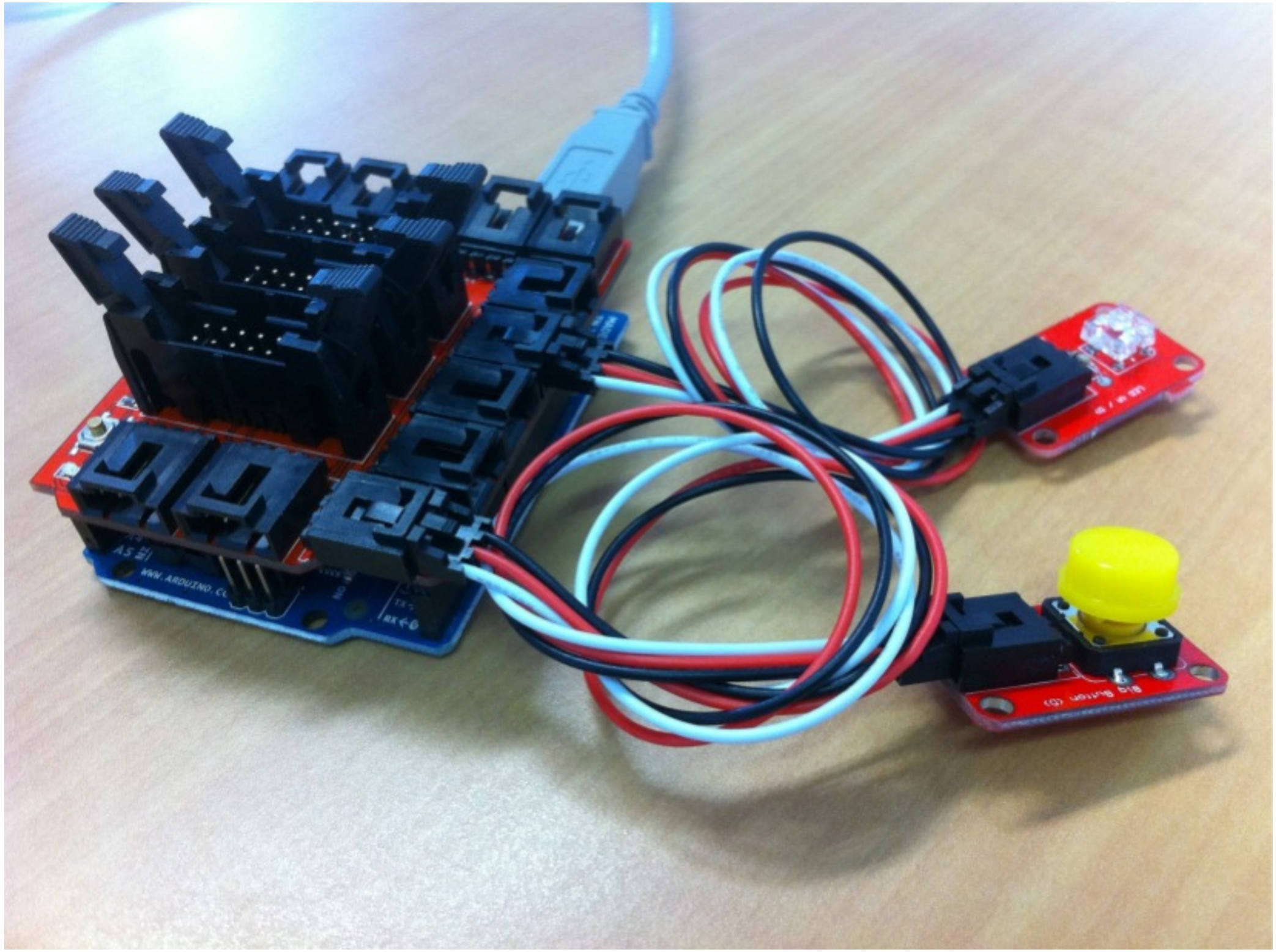
?

[Voelter]

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

[Voelter]

The ArduinoML Language



Sébastien Mosser, Philippe Collet, Mireille Blay-Fornarino. “Exploiting the Internet of Things to Teach Domain Specific Languages and Modeling” in Proceedings of the 10th Educators' Symposium at MODELS 2014 (EduSymp'14), ACM, IEEE, pages 1-10, Springer LNCS, Valencia, Spain, 29 september 2014

```
int state = LOW; int prev = HIGH;
long t = 0; long debounce = 200;
void setup() {
  pinMode(8,INPUT);
  pinMode(11, OUTPUT);
}
void loop() {
 int reading = digitalRead(8);
 if (reading == HIGH && prev == LOW
    && millis() - t > debounce) {
   if (state == HIGH) {
     state = LOW;
   } else { state = HIGH; }
   time = millis();
 }
 digitalWrite(11, state);
 prev = reading;
}
```

Concepts involved in ArduinoML

20 minutes

Semantic Model

4

"To abstract does not mean to remove details, it means to remove what is not necessary.

- Perdita Stevens [Panel MODELS'13]

details

≠

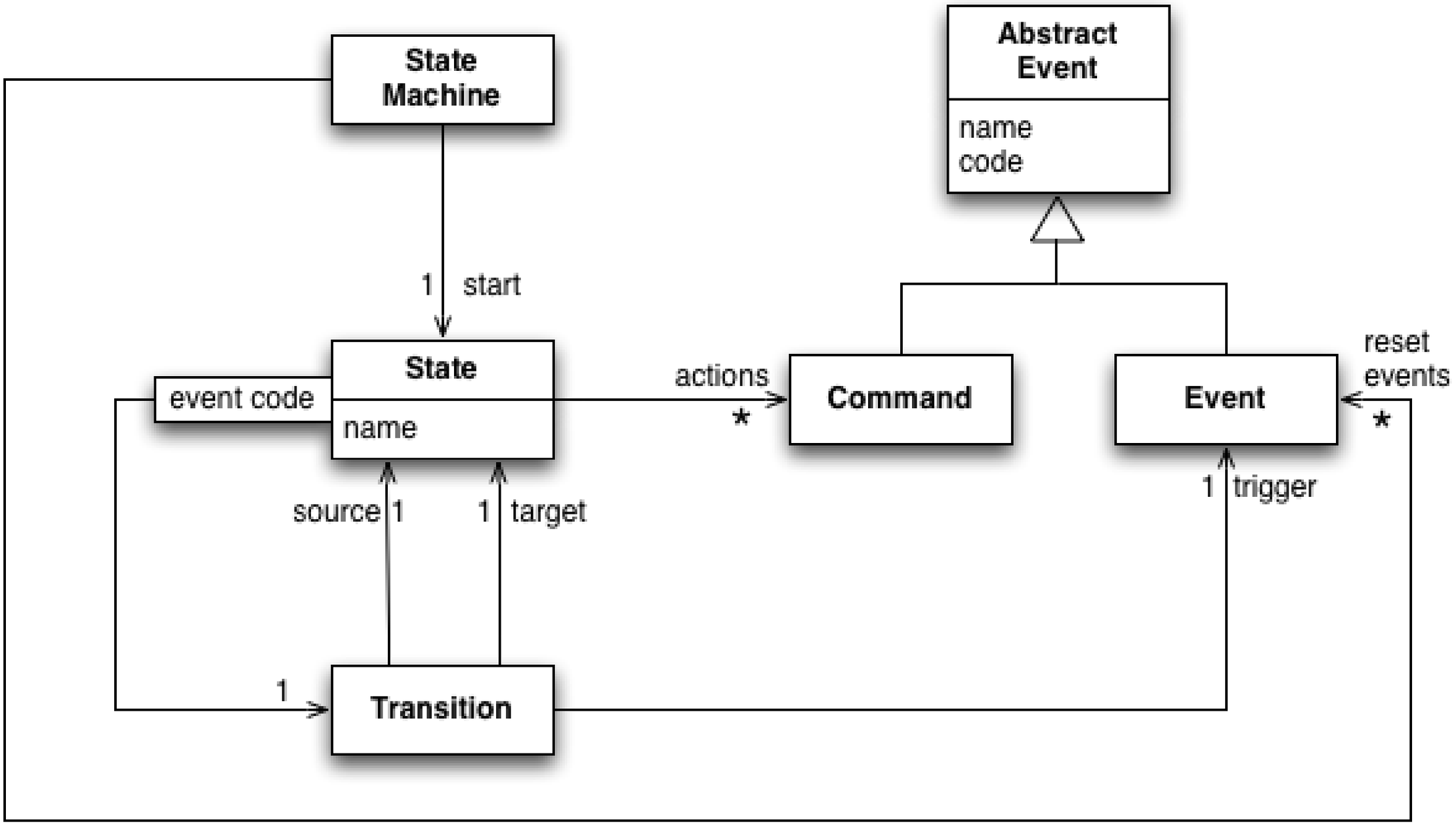
not necessary

«Modeling the **domain**»

≡

«Creating a **Semantic Model**»

Metamodel for controllers

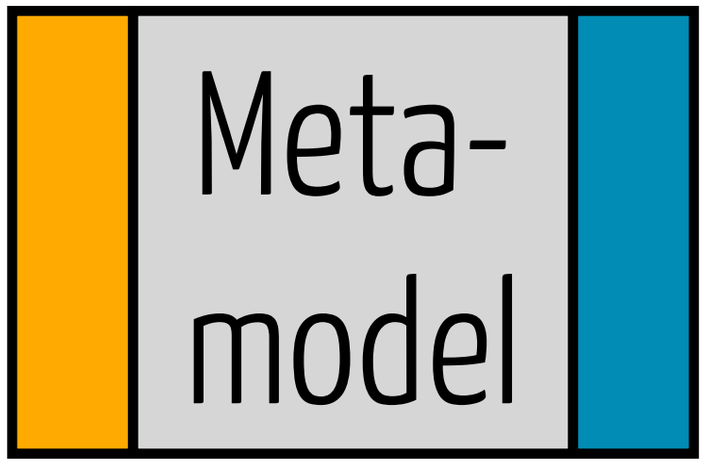


[Fowler]

Semantic Model ≡ Meta-model

- The «**essence**» of a domain, according to a **given viewpoint**
- Should be **usable with or without a DSL**
- Interfaces to **exploit** the meta-model

Population
Interface

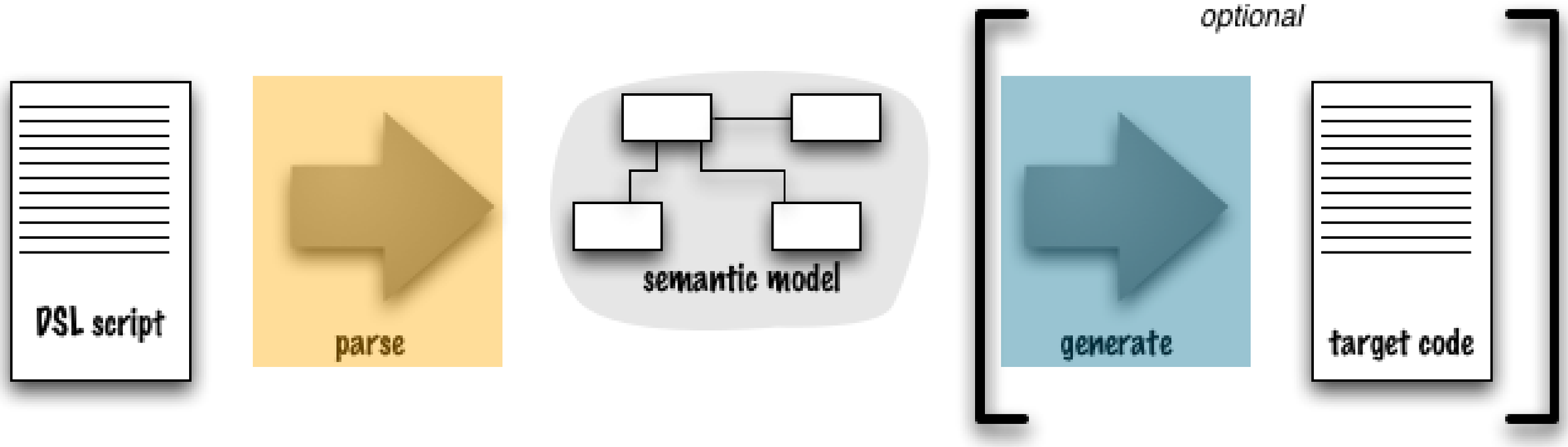


Operational
Interface

Population
Interface

Meta-
model

Operational
Interface

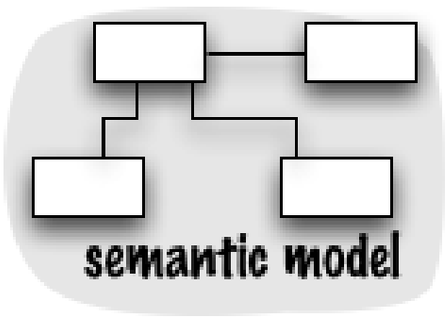


Behavioral concepts in ArduinoML

15 minutes

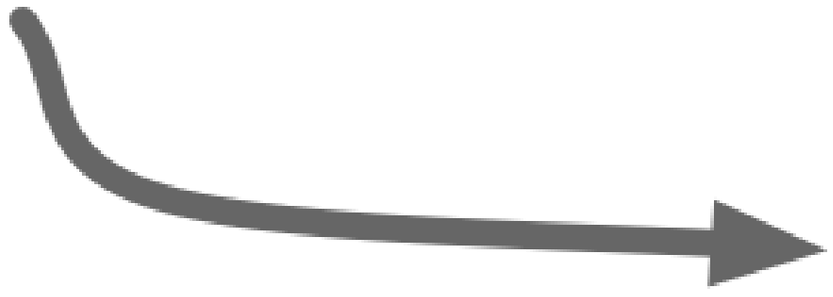
Code Generation

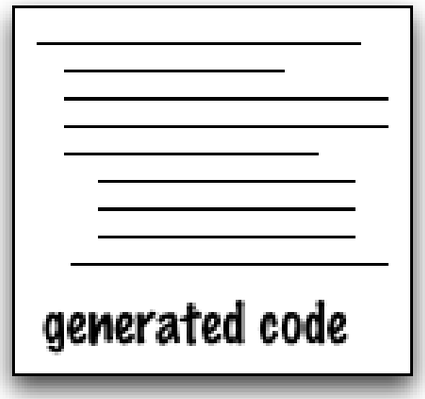
Transformer Generation



ToArduinoCode.java

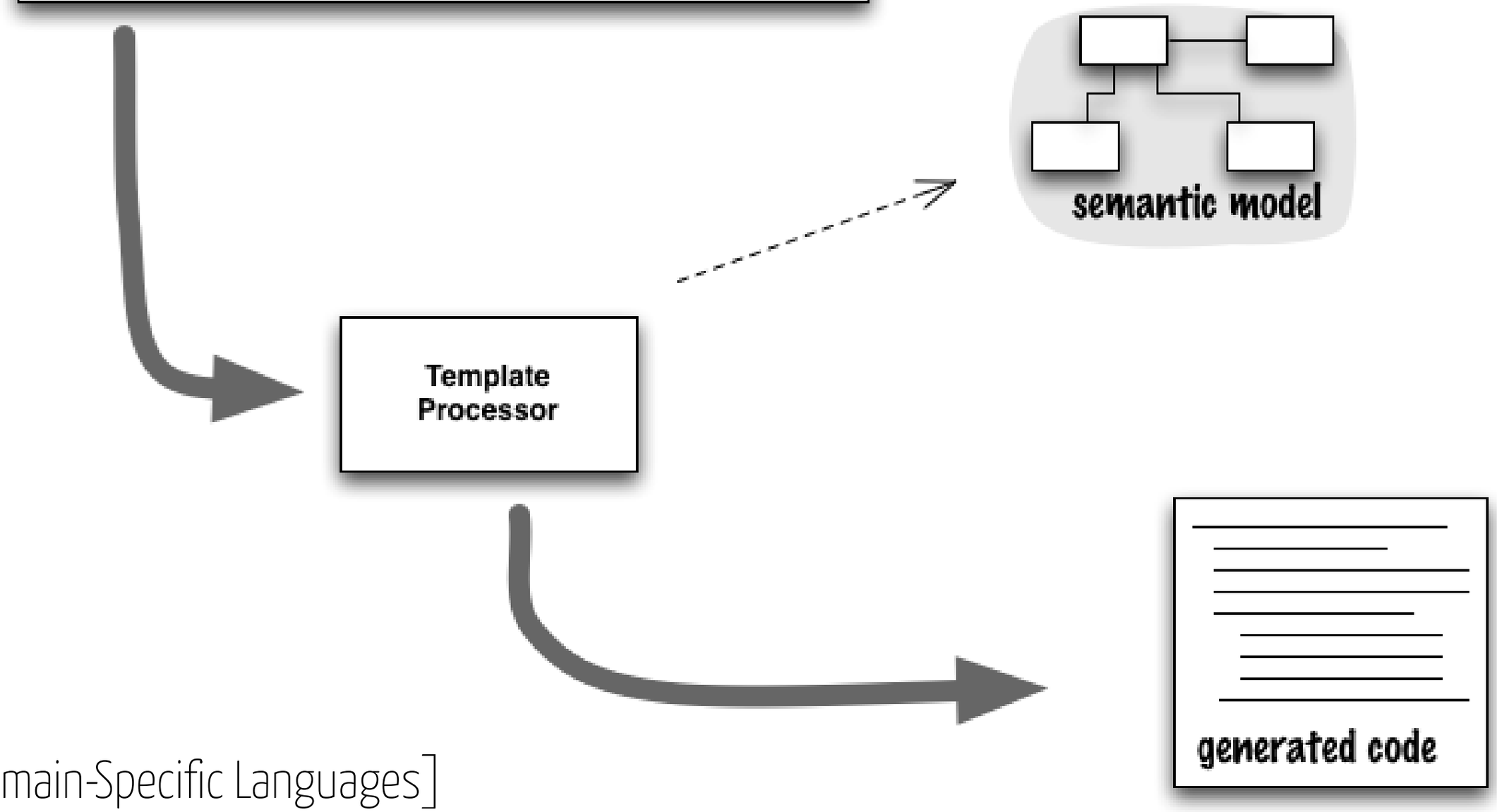
```
private void generateEvents(Writer output) throws IOException {
  for (Event e : machine.getEvents())
    output.write(String.format("  declare_event(\"%s\", \"%s\");\n",
                               e.getName(), e.getCode()));
  output.write("\n");
}
```



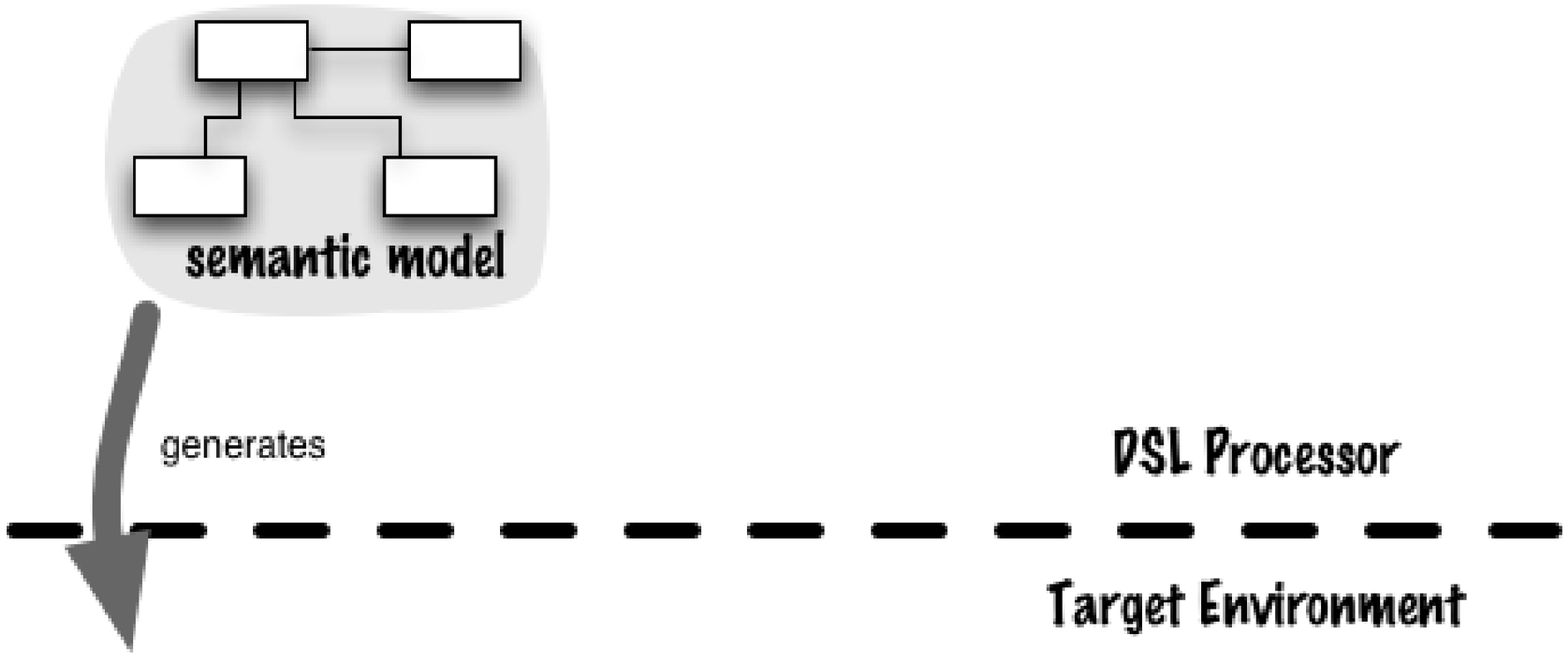


Template-based generation

```
#foreach($e in $helper.events)
  declare_event("${e.name}", "${e.code}");
#end
```

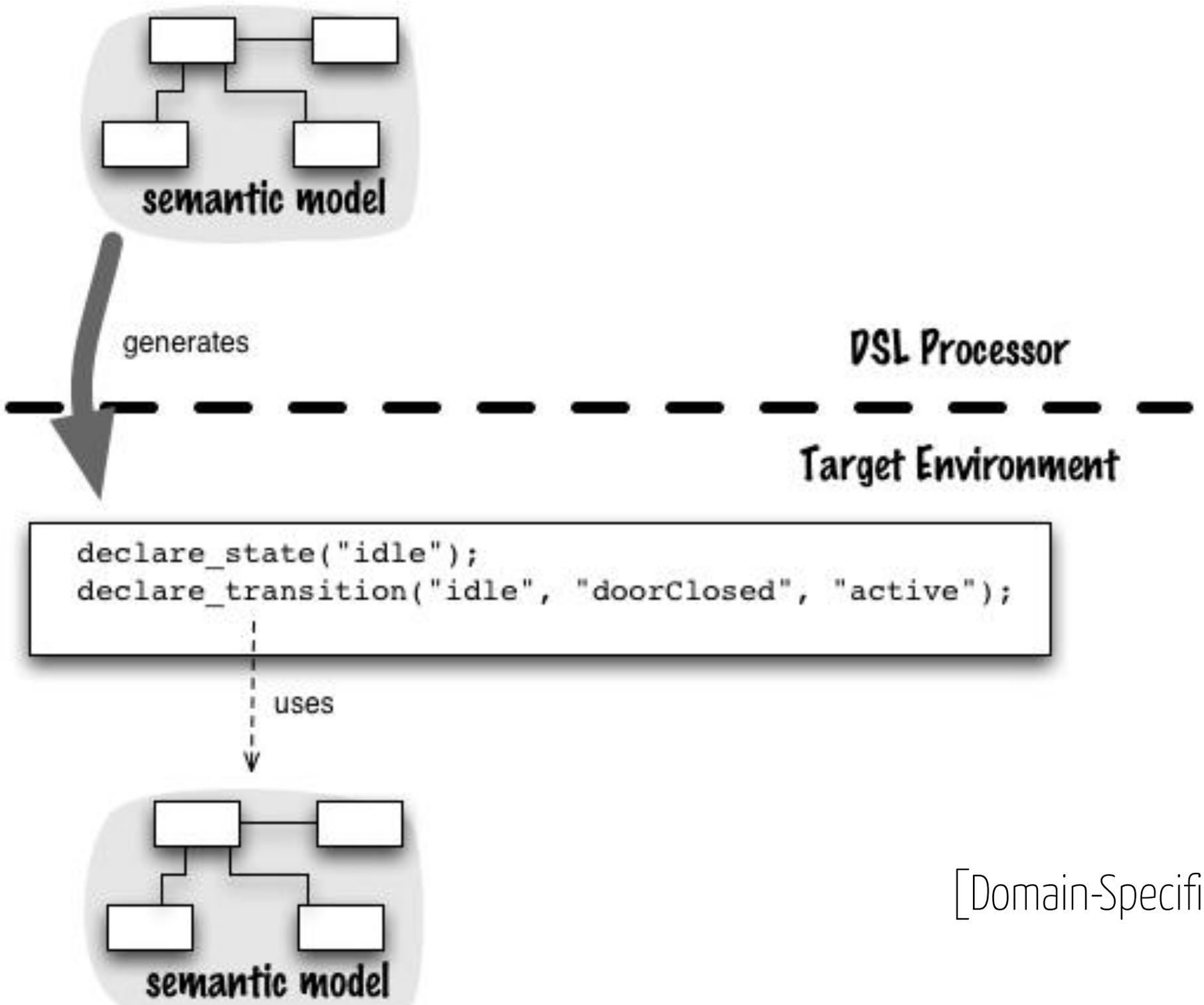


Model-ignorant Generation




```
void handle_event(char *code) {
  switch(current_state_id) {
  case STATE_idle: {
    if (0 == strcmp(code, EVENT_doorClosed)) {
      current_state_id = STATE_active;
    }
    return;
  }
  case STATE_active: {
 ...
```

Model-aware Generation



Projectional Edition

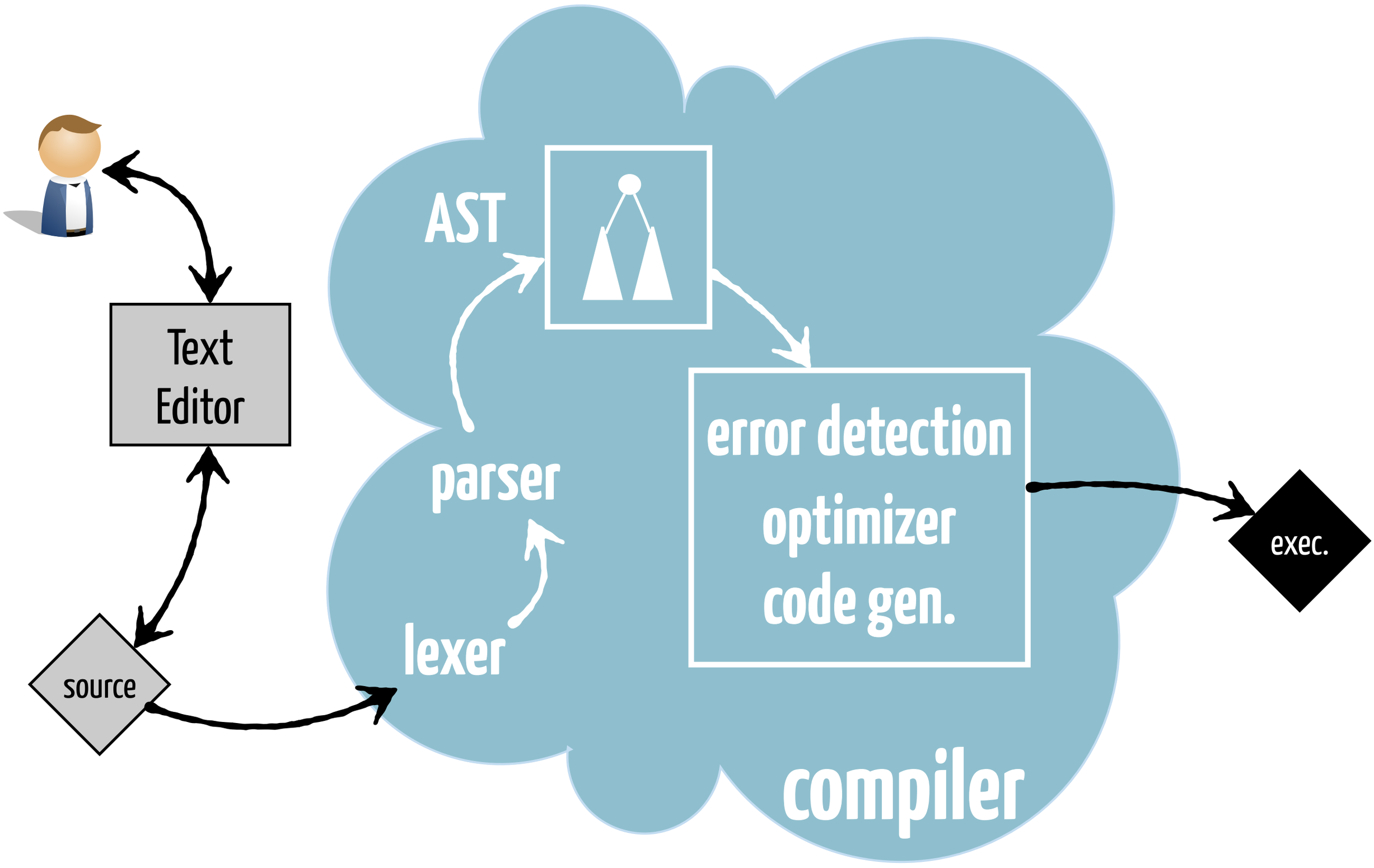
Until last year ...

model-driven grammars

meta-modeling framework

"Eclipse emphasizes a medieval approach of modeling, based on incantations

- Don Batory, MODELS'13



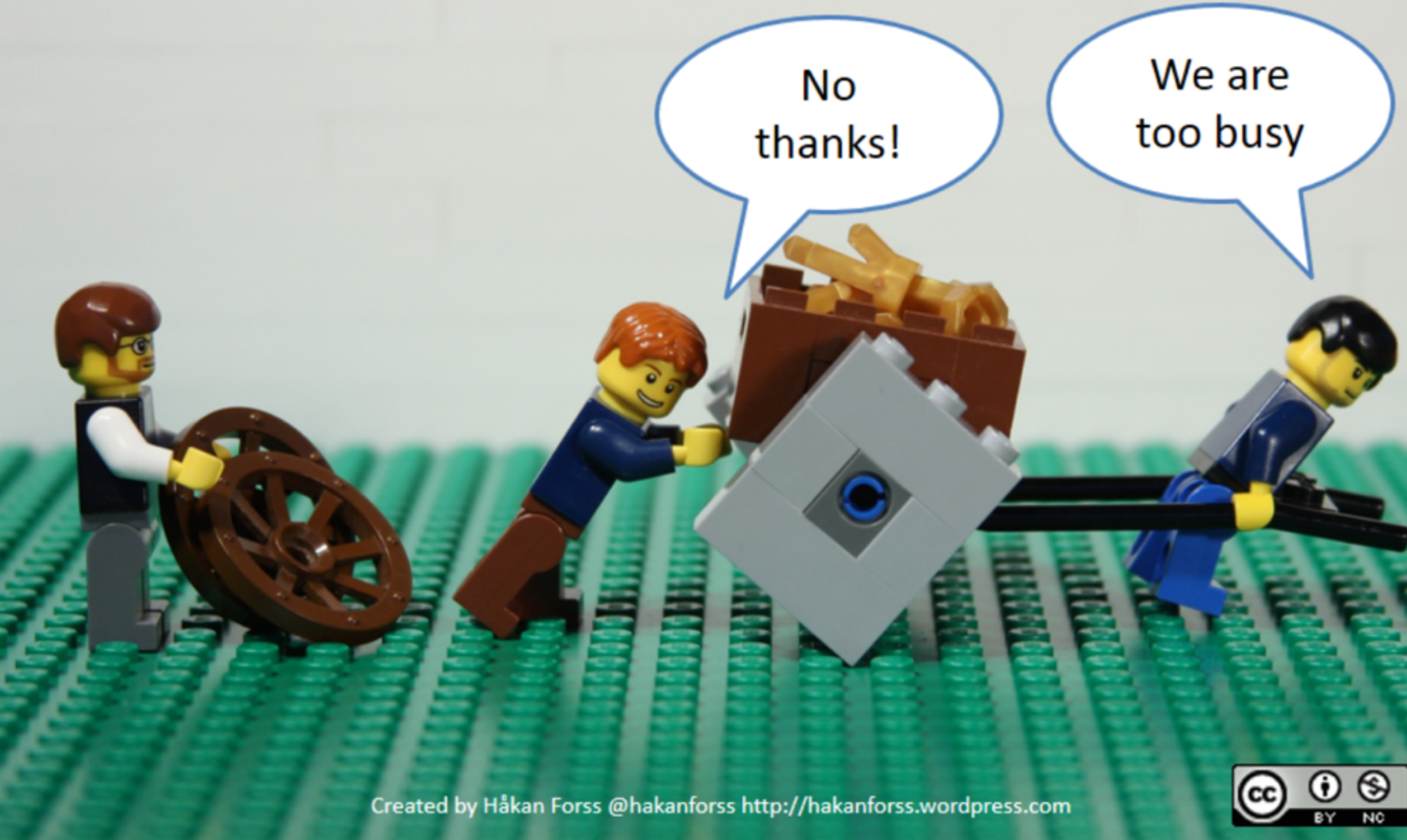
[based on Campagne's slides]

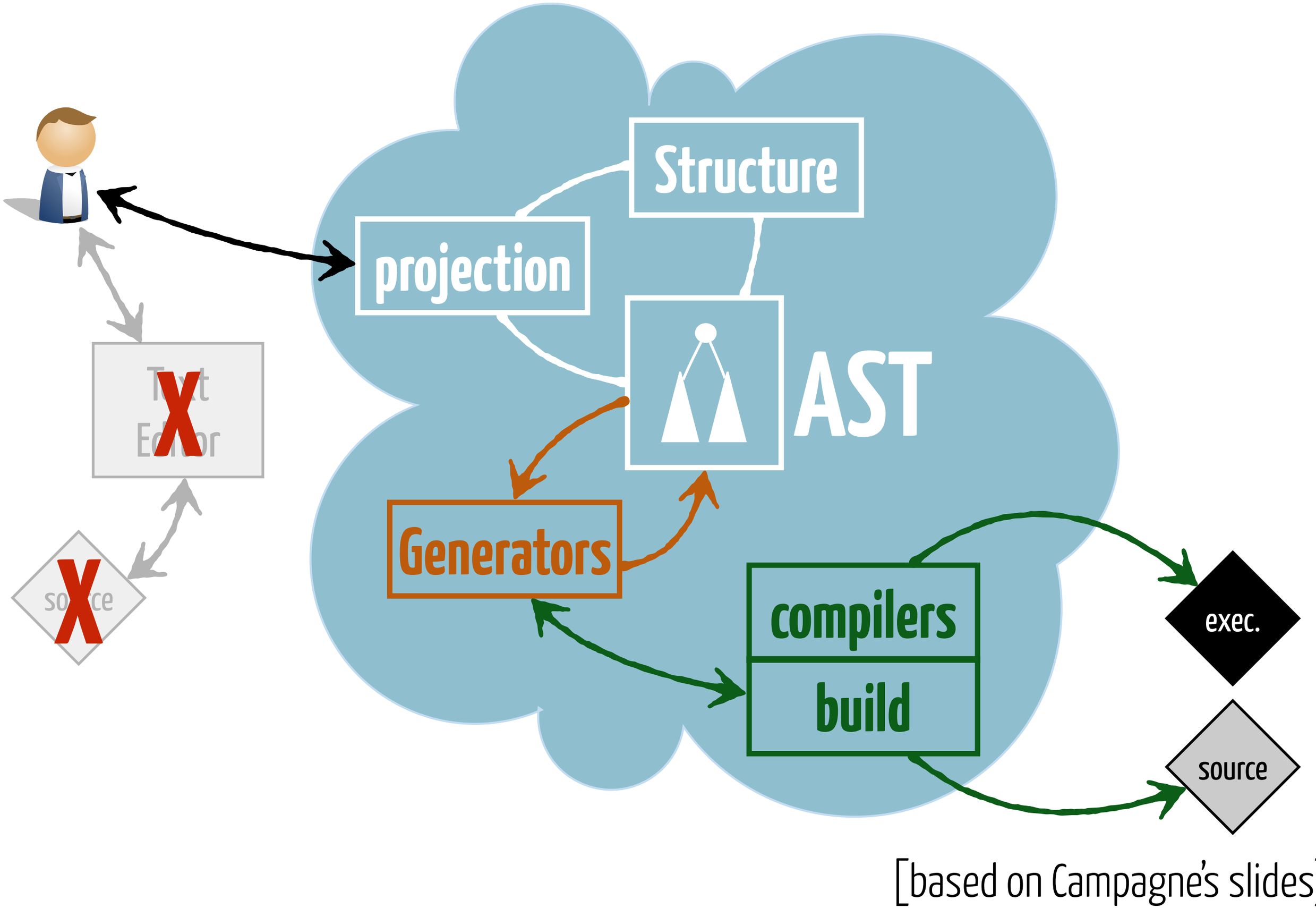
Storing **programs**

as **text**

is so **1950**

Are you too busy to improve?





[based on Campagne's slides]

Programs are **not text** anymore!

The **AST** is the **key**

copy/paste?

moving fragments?

Is projectional edition an hipster trap?

http://jeffgreenspan.com/projects/personal-work/urban-traps

ArduinoML MPS workshop

120 minutes