

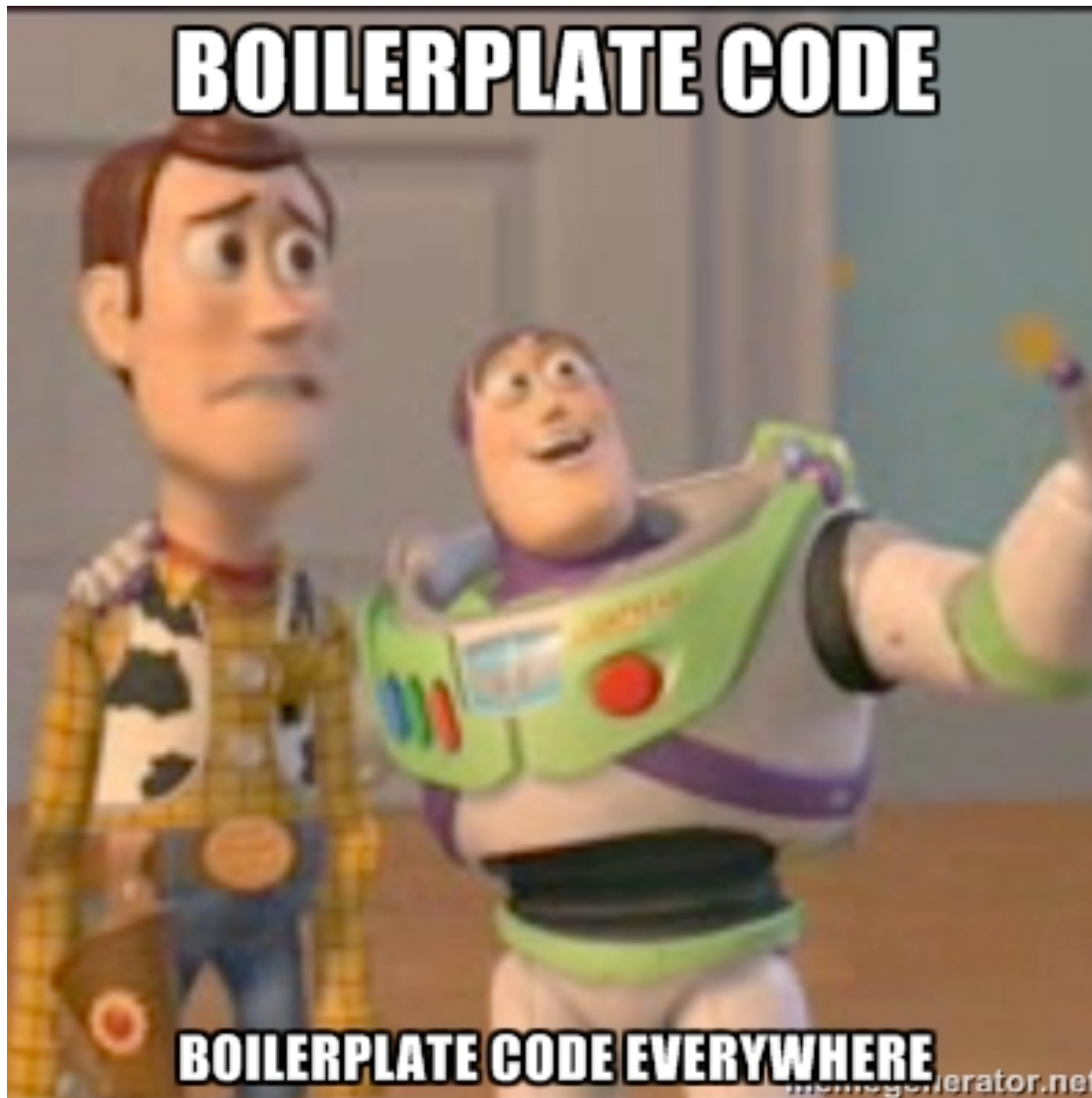
# Domain-Specific Languages

Tijs van der Storm  
([storm@cwi.nl](mailto:storm@cwi.nl) / @tvdstorm)

# Programming?



**BOILERPLATE CODE**



**BOILERPLATE CODE EVERYWHERE**

[memegenerator.net](http://memegenerator.net)

A programming language is low level when its programs require attention to the irrelevant

# Some facts

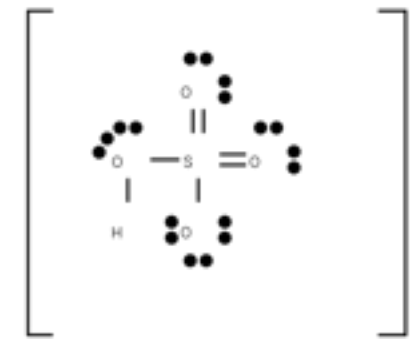
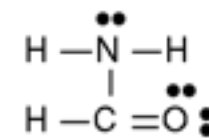
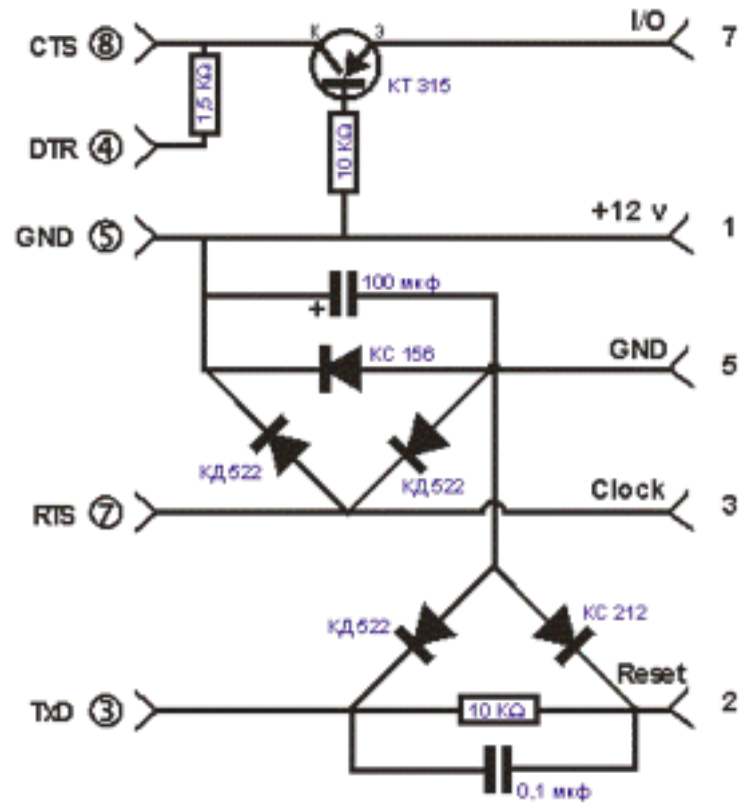
Fact 41. Maintenance typically consumes 40 to 80 percent of software costs. It is probably the most important life cycle phase of software.

Fact 44. Understanding the existing product is the most difficult task of maintenance.

Fact 21. For every 25 percent increase in problem complexity, there is a 100 percent increase in solution complexity.

## Domain Specific Languages!

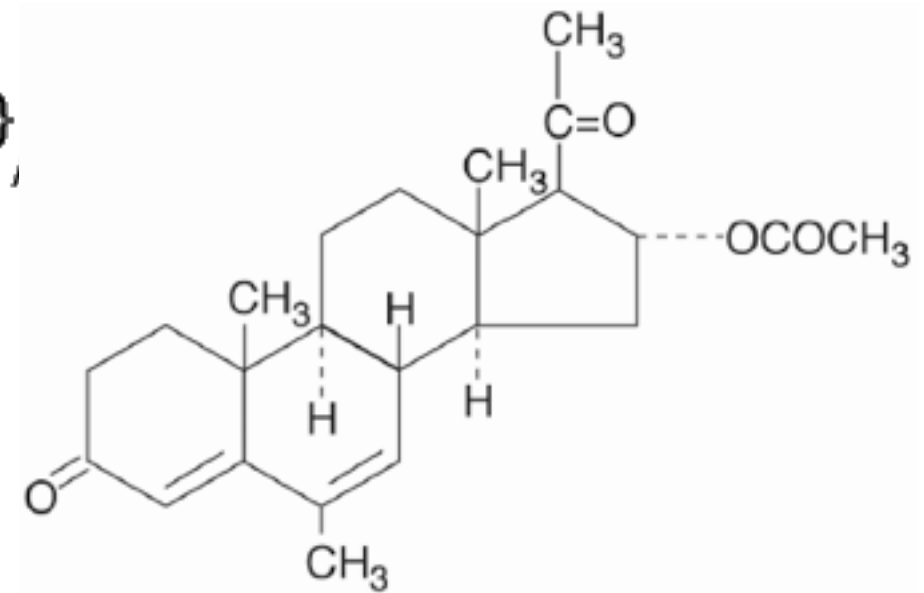
# Domain specific languages



$$c_t = S_t N(h) - X e^{-rt} N(h - \sigma \sqrt{\tau})$$

where

$$h = \left\{ \ln\left(\frac{S}{X}\right) + r\tau + \frac{\sigma^2 \tau}{2} \right\}$$





# Domain specific languages

E	-----	-----	-----	-----
B	-----	-----	-3--3-0--	-1--1-----
G	-0-2-0-----	-0-2-0-----	-----	-----0-----
D	-----2-----	-----2-----	-----	-----
A	-----	-----	-----	-----
E	-----	-----	-----	-----

Copyrighted Material

## Lieder eines fahrenden Gesellen

Arranged for Chamber Ensemble by Arnold Schoenberg

Nr. 1

### Course Materials

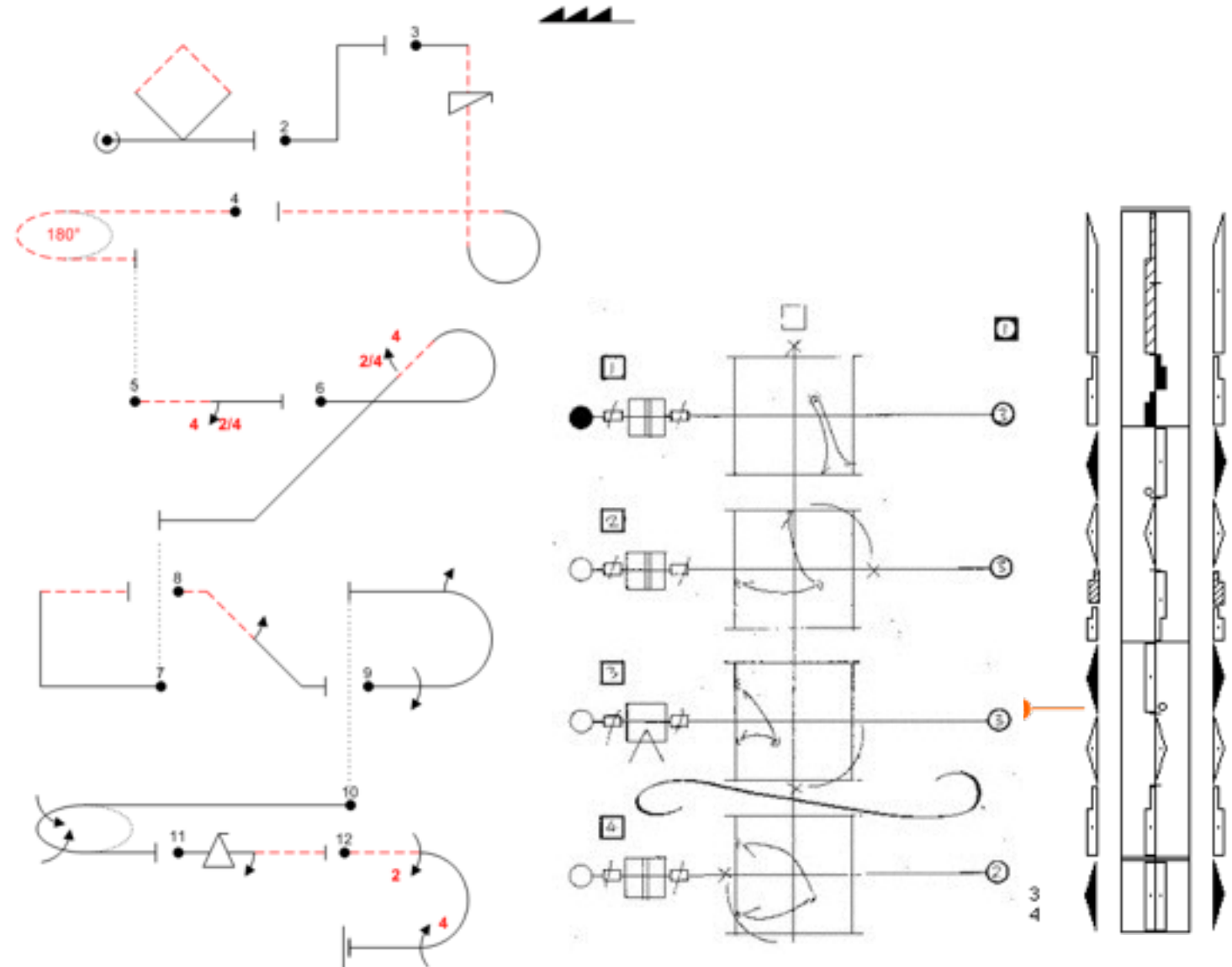


© Copyright 1979 by WILLIAM HOLT MUSIC PUBLICATIONS, Los Angeles. All rights reserved.

White: G. A. Anderssen  
Black: J. Dufresne  
Opening: Evans Gambit  
Location: Berlin, 1854

White	Black
-------	-------

White	Black
1. P-K4	P-K4
2. Kt-KB3	Kt-QB3
3. B-B4	B-B4
4. P-QKt4	BxKtP
5. P-B3	B-R4
6. P-Q4	PxP
7. O-O	P-Q6
8. Q-Kt3	Q-B3
9. P-K5	Q-Kt3
10. R-K1	KKt-K2
11. B-R3	P-Kt4
12. QxP	R-QKt1
13. Q-R4	B-Kt3
14. QKt-Q2	B-Kt2?
15. Kt-K4	Q-B4?
16. BxQP	Q-R4
17. Kt-B6 ch!	PxKt
18. PxP	R-Kt1
19. QR-Q1!	QxKt
20. RxKt ch	KtxR
21. QxP ch!	KxQ
22. B-B5 dbl ch	K-K1
23. B-Q7 ch	K-B1
24. BxKt mate	



# Observations

- Special purpose
- Restricted
- Concise
- Expert usage
- Formalized
- Textual or graphic or combination



# General purpose languages (GPLs)



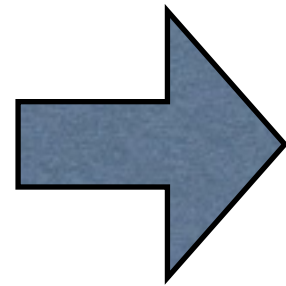
# DSLs



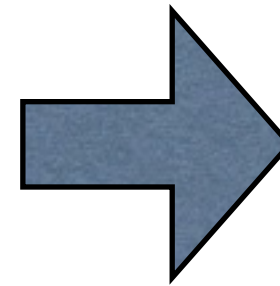
# Programming



Domain



Programmer



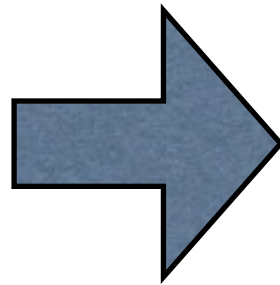
```
.text:131411EF ; FUNCTION CHUNK AT .text:13141239 SIZE 000
.text:131411EF ; FUNCTION CHUNK AT .unp0:13143000 SIZE 000
.text:131411EF ; FUNCTION CHUNK AT .unp0:13143522 SIZE 000
.text:131411EF ; FUNCTION CHUNK AT .unp0:13145C99 SIZE 000
.text:131411EF 68 F1 4F 5B FF      push    0FF5B4FF1h
.text:131411F4 E9 A0 4A 00 00      jmp     loc_13145C99
.text:131411F4
.text:131411F4
.text:131411F9 64 69 31          byte_131411F9 db 64h, 69h, 31h
.text:131411FC 60 16 42 75 9A C4+ dd 75421660h, 0CACDC49Ah, 7
.text:131411FC CD CA FA 7B 9D 7B+ dd 6EEE544Dh, 10D69610h, 0F
.text:131411FC D5 04 4B 2B 85 4B+ dd 0EB6323E4h, 82B58465h, 0
.text:13141238
.text:13141238 58
.text:13141239
.text:13141239
.text:13141239 52
.text:1314123A 55
.text:1314123B 53
.text:1314123C 51
.text:1314123D 9C
.text:1314123E 57
.text:1314123F 50
.text:13141240 56
.text:13141241 51
.text:13141242 68 00 00 00 00
.text:13141247 8B 74 24 28
.text:1314124B BF F9 11 14 13
.text:13141250
.text:13141250
.text:13141250 89 F3          loc_13141250:  mov     ebx, esi
.text:13141252 03 34 24          add     esi, [esp+2Ch+var_2
.text:13141255
.text:13141255
.text:13141255
.text:13141255
loc_13141239:
push    edx
push    ebp
push    ebx
push    ecx
pushf
push    edi
push    eax
push    esi
push    ecx
push    esi, [esp+2Ch+var_4
mov     edi, offset byte_13
loc_13141250:
mov     ebx, esi
add     esi, [esp+2Ch+var_2
loc_13141255:
; C
; S
```

Code

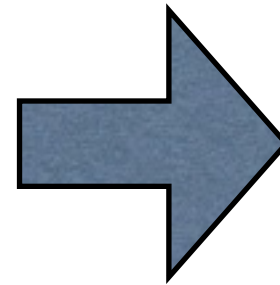
# Programming



Domain



Programmer



Code

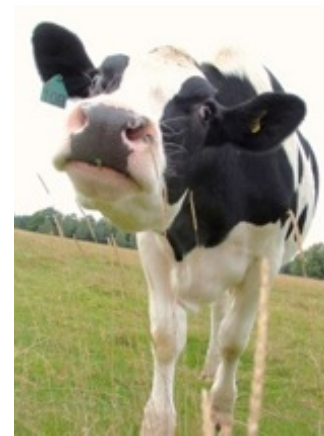


# Programming is “lossy”

- encoding
- obfuscating
- encrypting
- dispersing
- tangling
- distorting



# Cognitive distance



?



# The problem

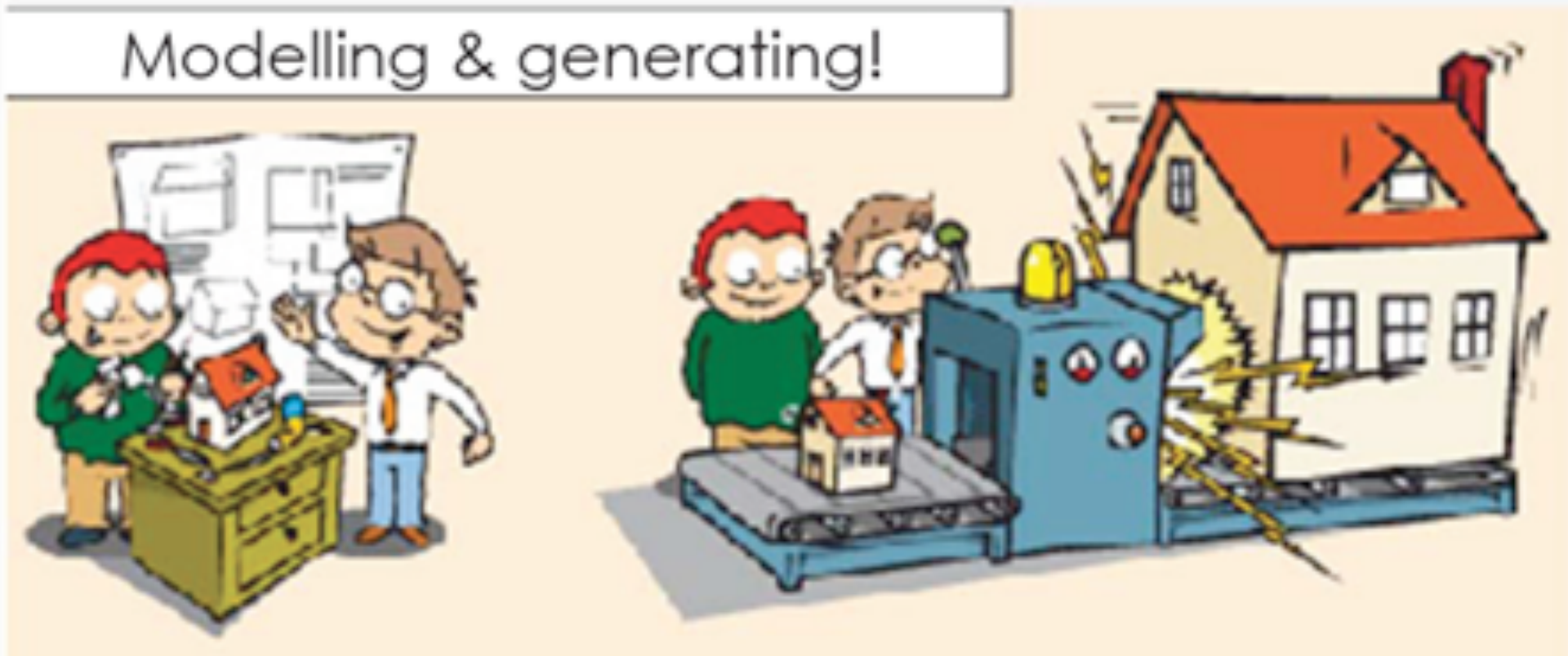
- a lot of code,
- low level code,
- characterized by lack of abstraction
- encoding domain knowledge
- and encoding design knowledge



Programming?



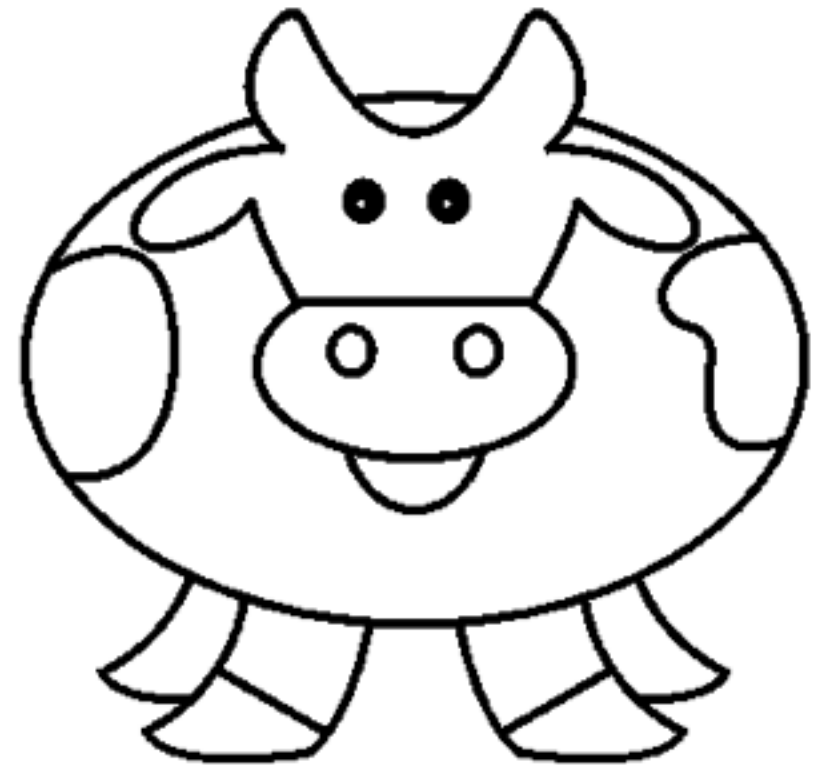
Modelling & generating!



# Modeling the domain



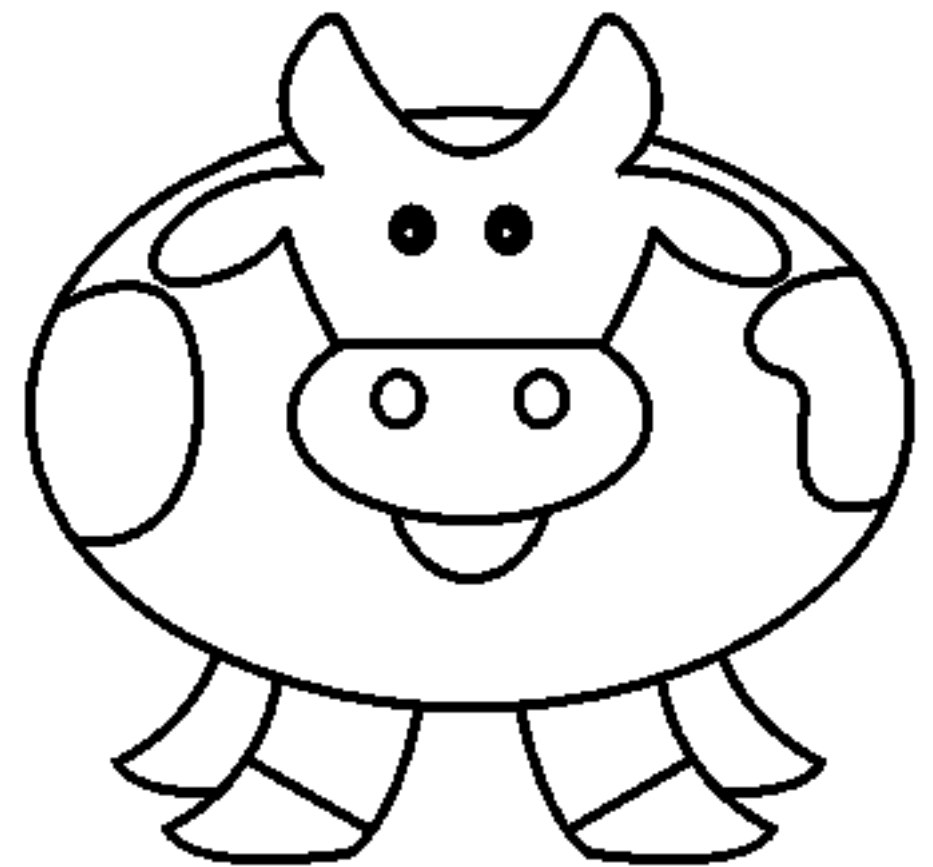
domain analysis



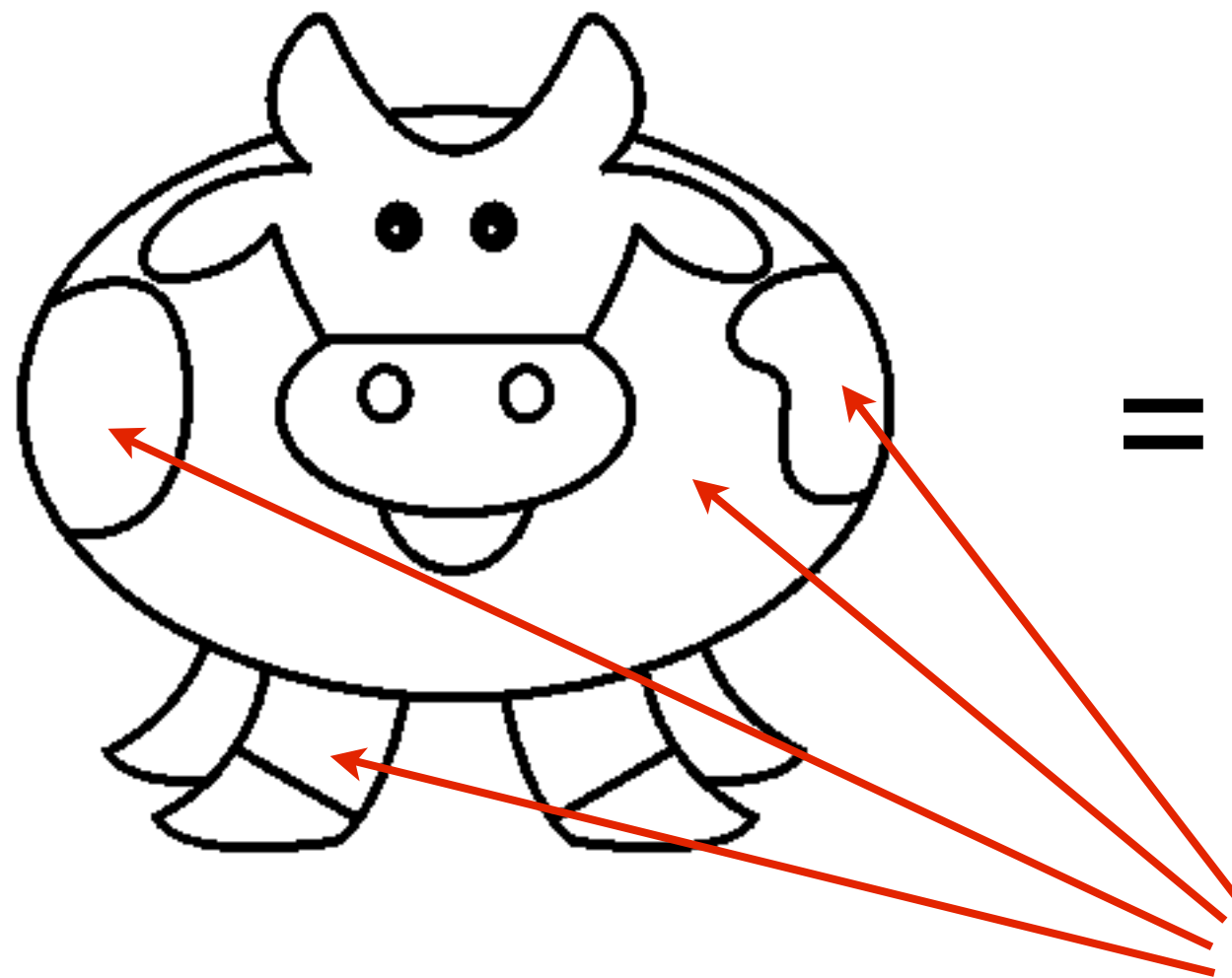
Ceci n'est pas une vache



# System families



# Domain Specific Language

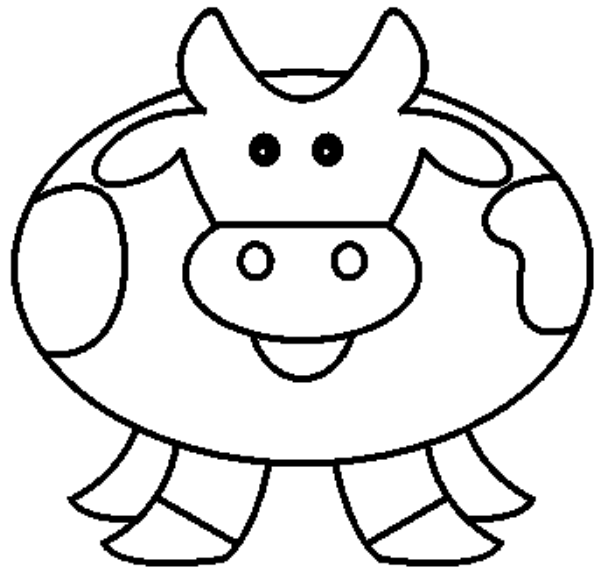


=

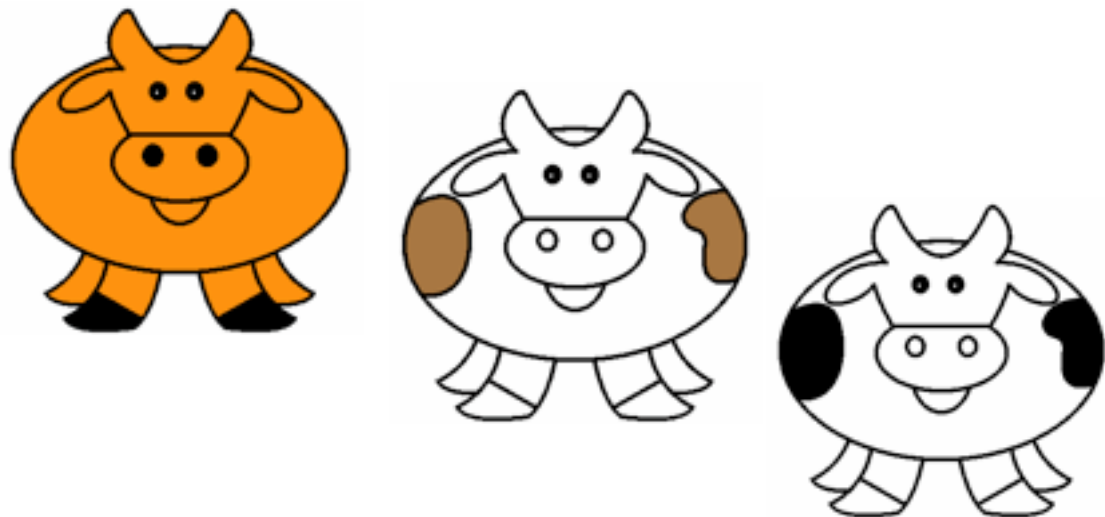
formalized  
notation  
capturing  
“Cows”

variation  
points

# Domain Specific Languages

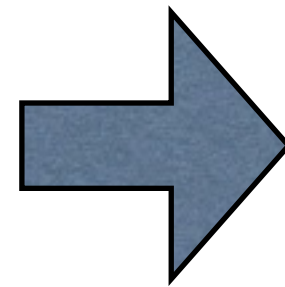
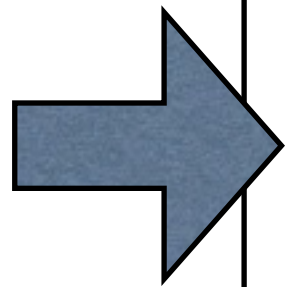


=  
grammar,  
template,  
metamodel



=  
sentence,  
instance,  
model

# Code generation

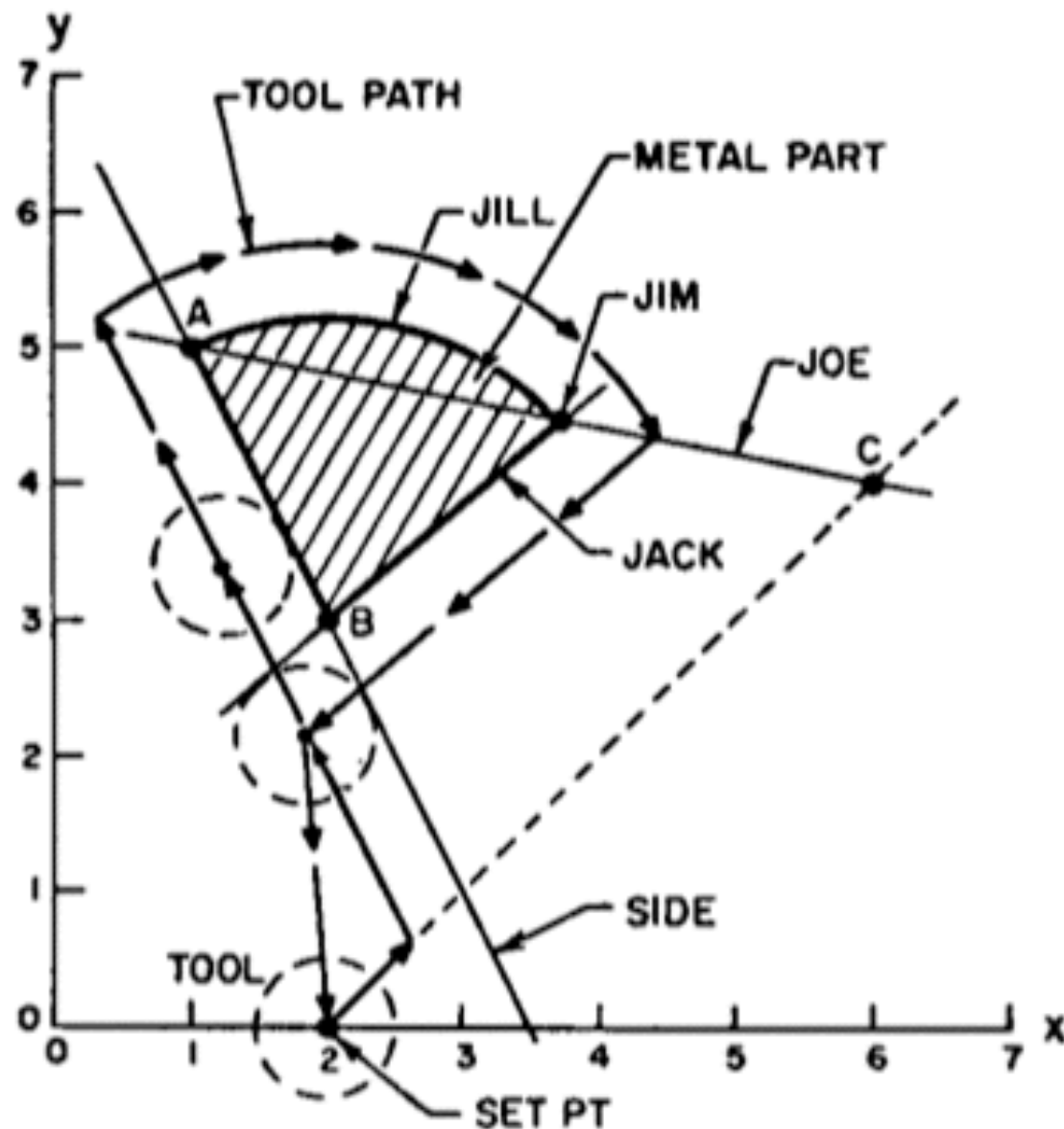


Code generator

Code



# APT: numerical control



from the '50s (!)

A = POINT / 1, 5  
 B = POINT / 2, 3  
 C = POINT / 6, 4  
 TL DIA / +1.0, INCH  
 FEDRAT / 30, IPM

SET PT = FROM, POINT / 2, 0  
 IN DIR, POINT / C

SIDE = GO TO, LINE / THRU, A, AND, B  
 WITH, TL LFT, GO LFT, ALONG / SIDE

JILL = GO RGT, ALONG, CIRCLE / WITH, CTR AT, B, THRU, A

JOE = LINE / THRU, A, AND, C

JIM = POINT / X LARGE, INT OF, JOE, WITH, JILL

JACK = LINE / THRU, JIM, AND, B  
 GO RGT, ALONG / JACK, UNTIL, TOOL, PAST, SIDE  
 GO TO / SET PT  
 STOP, END, FINI



# LaTeX: document preparation

```
\subsection{Application à un exemple: la coévolution proies-prédateurs}
\subsubsection{Étape 1: Modèle écologique et stationnarité}
Nous nous intéresserons dans ce cas au modèle simple de Lotka-Volterra,
énoncé par le système-\ref{eq:lotka_volterra}.
```

Dans ce modèle de base, il faut introduire une dépendance au trait sujet à évolution qui nous intéresse. Ici, nous considérons la taille corporelle  $x$  comme trait d'intérêt et supposons que la compétition intraspécifique,  $\alpha$ , et la prédation,  $\beta$ , en dépendent ainsi:

```
\begin{eqnarray}
\alpha(x_1) &= & \alpha_0 + \alpha_2(x_1 - x_{1_0})^2 \\
\beta(x_1, x_2) &= & \beta_0 \\
&\exp\left[-\left(\frac{x_1}{\beta_1}\right)^2 + \right. \\
&2\beta_3\left(\frac{x_1}{\beta_1}\right)\left(\frac{x_2}{\beta_2}\right) \\
&\left. - \left(\frac{x_2}{\beta_2}\right)^2\right] \\
\end{eqnarray}
```

```
\begin{figure}[p]
\begin{center}
\includegraphics[width=0.45\textwidth]{figures/func_alp}
\includegraphics[width=0.45\textwidth]{figures/func_bet}
\caption{Les fonctions choisies pour  $\alpha$  et  $\beta$ }
\end{center}
\end{figure}
```

# VHDL: hardware description

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClkDiv is
    Port ( InByte : in STD_LOGIC_VECTOR(3 downto 0);           --<-- Seq_CPLD
           RegSel  : in STD_LOGIC_VECTOR(1 downto 0);         --<-- Seq_CPLD
           RegStrb  : in STD_LOGIC;                           --<-- Seq_CPLD
           MClk     : in STD_LOGIC;                           --<-- OSC
           SeqReset  : in STD_LOGIC;                           --<-- Power Monitor
           ADC_Clk  : out STD_LOGIC);                          -->-- ADC
end ClkDiv;

architecture Behavioral of ClkDiv is
    signal ADC_div : STD_LOGIC_VECTOR(5 downto 0) := "001111";
    signal ADCClk  : STD_LOGIC := '0';
    signal ClkSel  : STD_LOGIC_VECTOR(2 downto 0) := "100";

begin
```

# Risla: financial products

**product** LOAN

**declaration**

**contract data**

PAMOUNT : amount	<i>%% Principal Amount</i>
STARTDATE : date	<i>%% Starting date</i>
MATURDATE : date	<i>%% Maturity data</i>
INTRATE : int-rate	<i>%% Interest rate</i>
RDMLIST := [] : cashflow-list	<i>%% List of redemptions.</i>

**information**

PAF : cashflow-list	<i>%% Principal Amount Flow</i>
IAF : cashflow-list	<i>%% Interest Amount Flow</i>

**registration**

*%% Register one redemption.*  
RDM(AMOUNT : amount, DATE : date)



Developed  
at CWI

# QL

```
form Box1HouseOwning {  
  "Did you sell a house in 2010?" hasSoldHouse: boolean  
  "Did you buy a house in 2010?" hasBoughtHouse: boolean  
  "Did you enter a loan for maintenance?" hasMaintLoan: boolean  
  if (hasSoldHouse) {  
    "Private debts for the sold house:" privateDebt: money  
    "Price the house was sold for:" sellingPrice: money  
    "Value residue:" valueResidue = sellingPrice - privateDebt  
  }  
}
```

# Other examples

- Make: software building
- Dot: graph visualization
- SQL: relational querying
- SWUL: Swing GUIs
- HTML: hypertext
- CLOPS: commandline options
- GNUPlot: plotting
- R: statistics
- CML: kernel config
- Lex: lexical scanning
- Excel: spreadsheets
- Rascal: meta-programming
- ...

# Domain-specific languages

- Better languages, for specific domains
- Capture families of systems
- Higher level of abstraction
- Focus on “what” vs “how”
- Reuse designs, not just code
- Language workbenches (e.g., Rascal)

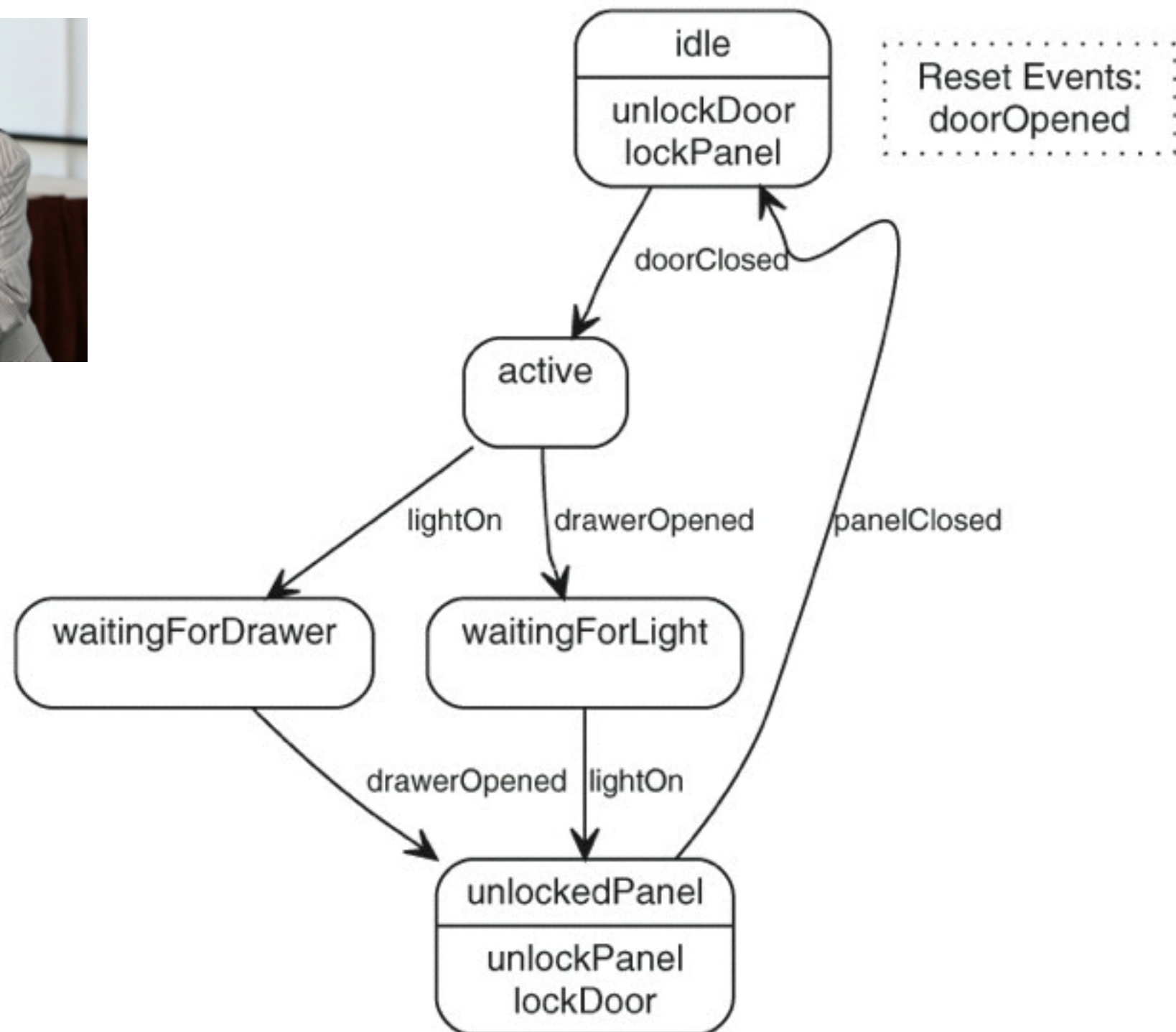


**Demo**





# State machines



# Textual notation

## events

```
doorClosed D1CL
drawerOpened D20P
lightOn L10N
doorOpened D10P
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
```

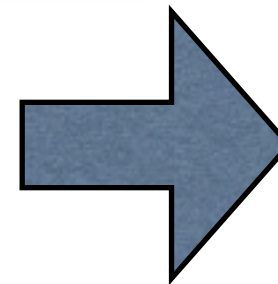
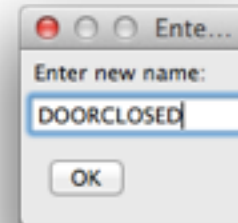
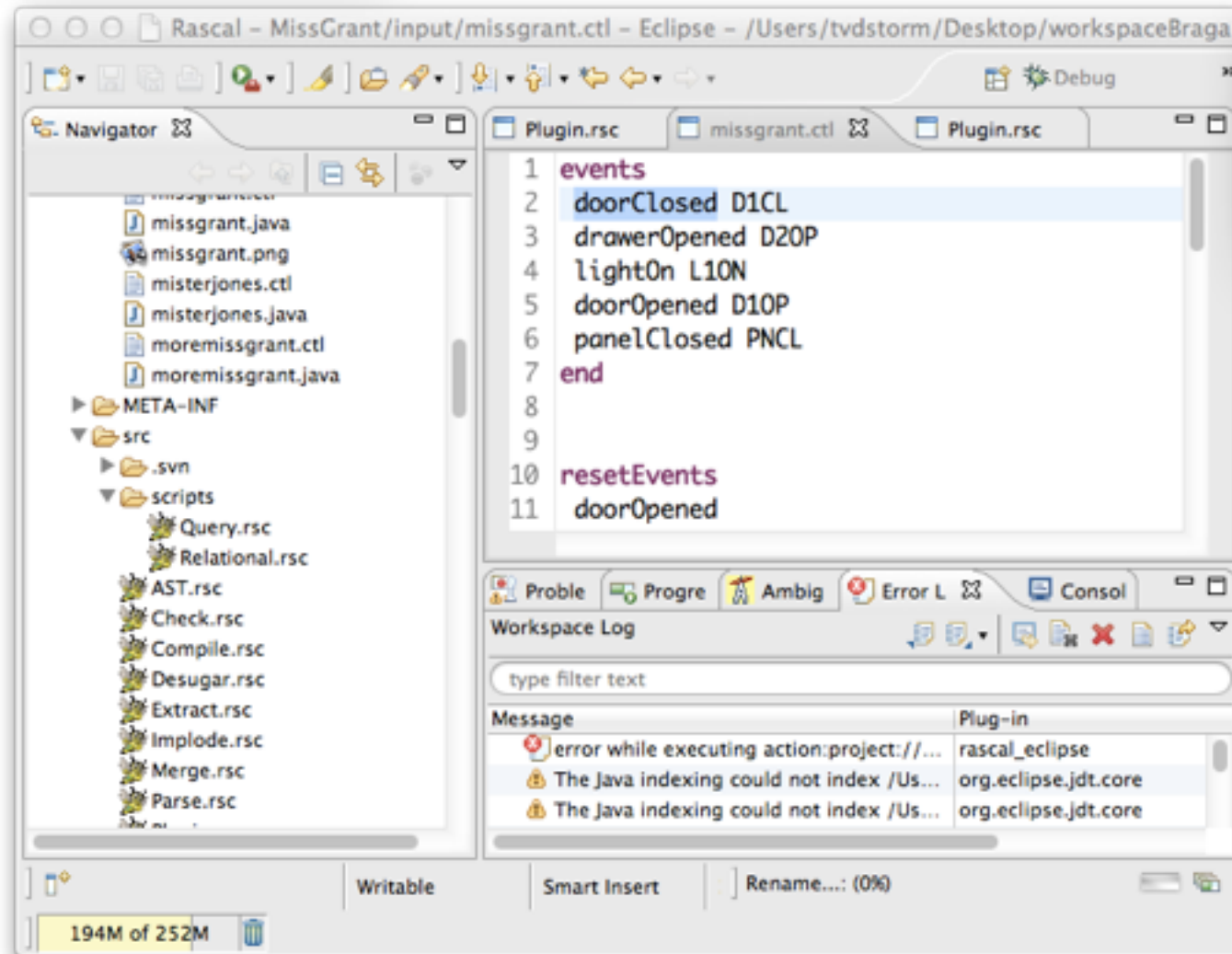
end

## state unlockedPanel

```
actions {unlockPanel lockDoor}
panelClosed => idle
```

end

Visualize  
Rename...

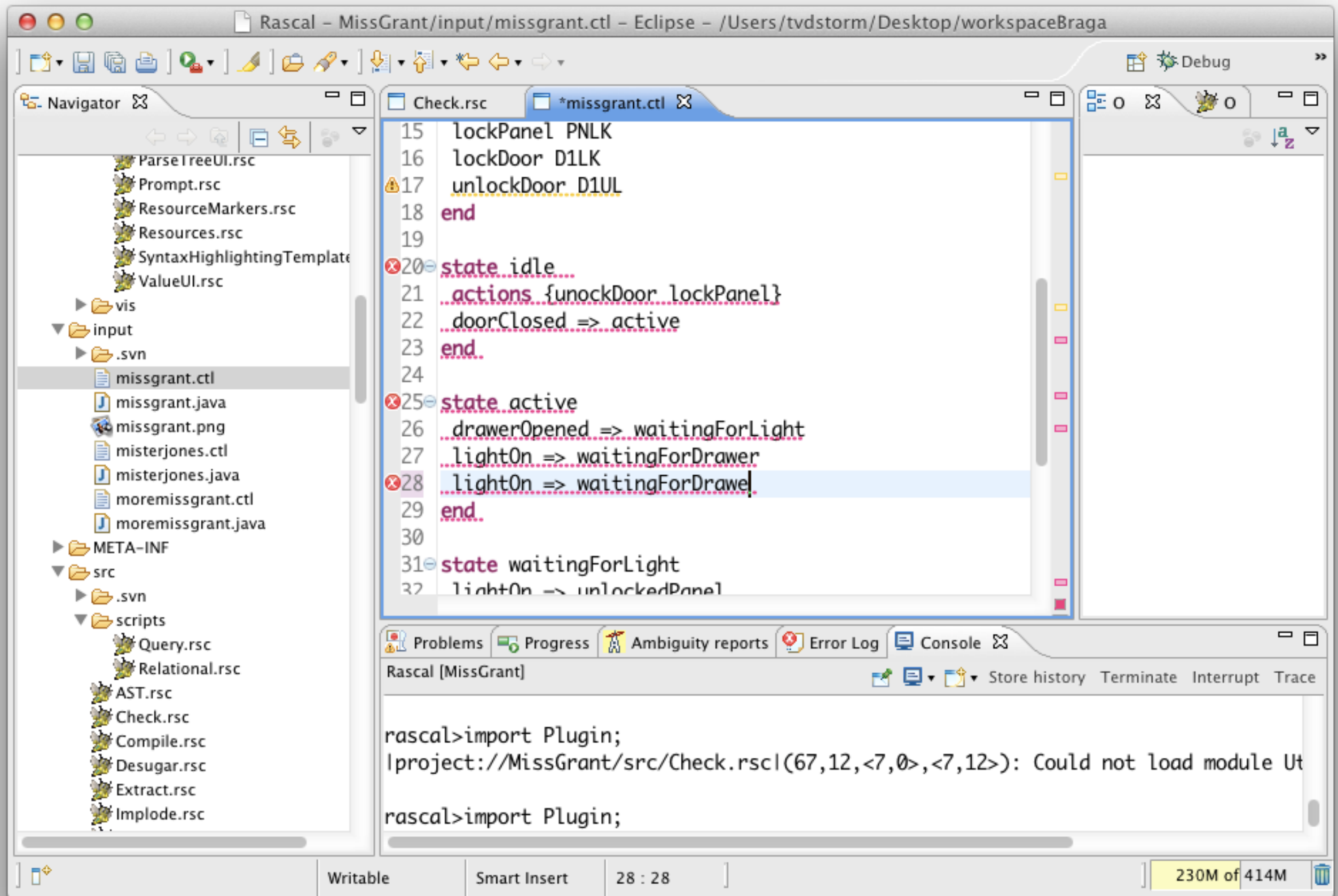


```
events
  DOORCLOSED D1CL
  drawerOpened D2OP
  lightOn L1ON
  doorOpened D1OP
  panelClosed PNCL
end

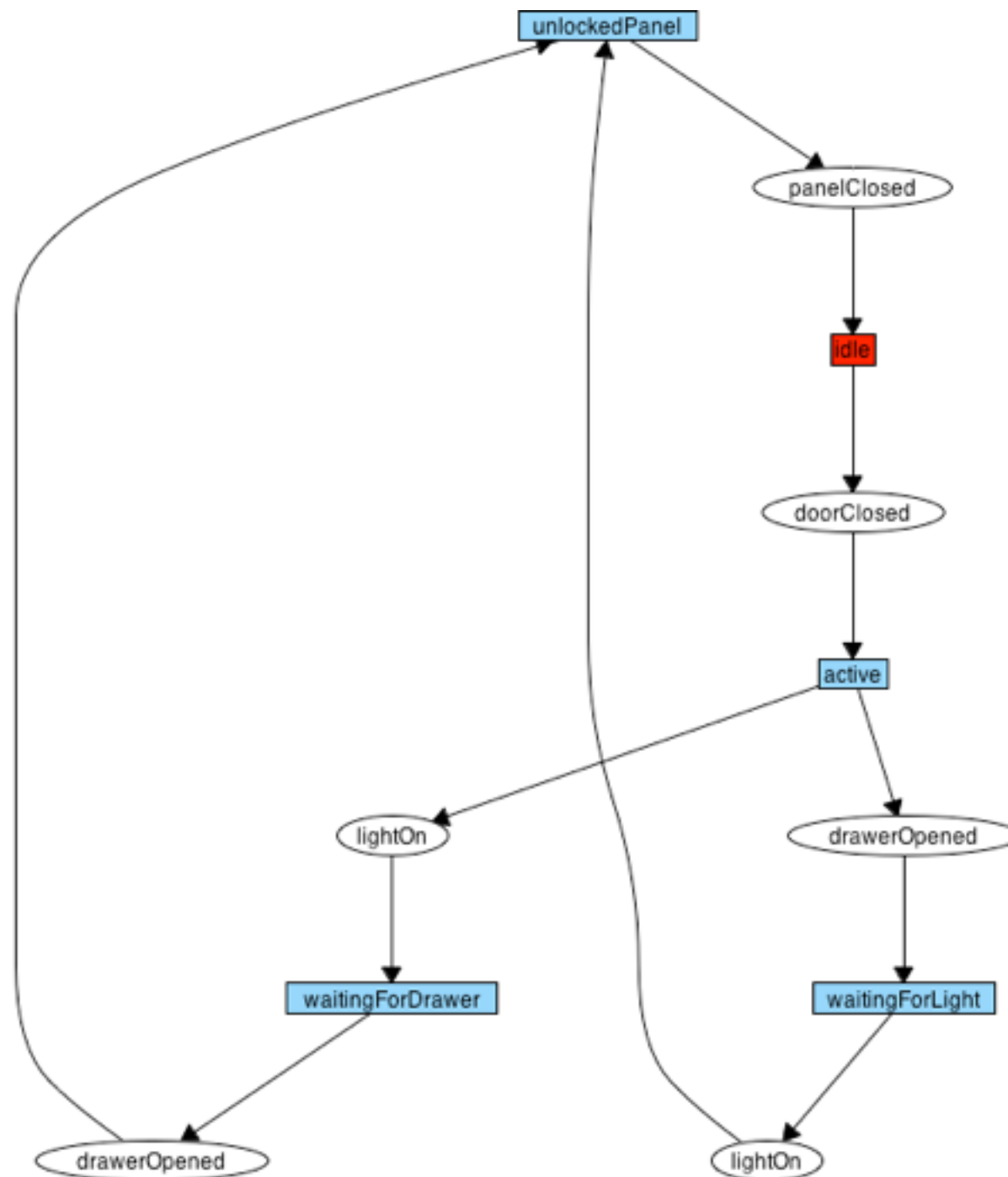
resetEvents
  doorOpened
end

commands
  unlockPanel PNUL
  lockPanel PNKL
  lockDoor D1LK
  unlockDoor D1UL
end

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



# Visualization



# Code generation

```
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

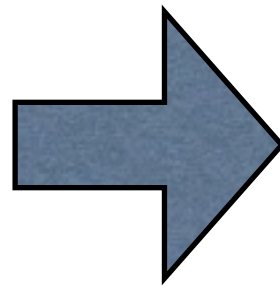
state idle
  actions {unlockDoor lockPanel}
  doorClosed => active
end

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
```



```
public class missgrant {
  public static void main(String args[]) throws java.io.IOException {
    new missgrant().run(new java.util.Scanner(System.in),
      new java.io.PrintWriter(System.out));
  }

  private static final int state$idle = 0;
  private static final int state$active = 1;
  private static final int state$waitingForLight = 2;
  private static final int state$waitingForDrawer = 3;
  private static final int state$unlockedPanel = 4;

  public void run(java.util.Scanner input, java.io.Writer output)
    throws java.io.IOException {
    int state = state$idle;
    while (true) {
      String token = input.nextLine();
      switch (state) {

        case state$idle: {
          unlockDoor(output);
          lockPanel(output);
          if (doorClosed(token)) {
            state = state$active;
          }
          if (doorOpened(token)) {
            state = state$idle;
          }
          break;
        }

        case state$active: {
          if (drawerOpened(token)) {
            state = state$waitingForLight;
          }
          if (lightOn(token)) {
            state = state$waitingForDrawer;
          }
          if (doorOpened(token)) {
            state = state$idle;
          }
          break;
        }

        case state$waitingForLight: {
```