Live programming

Programmeren als boetseren

Centrum Wiskunde & Informatica

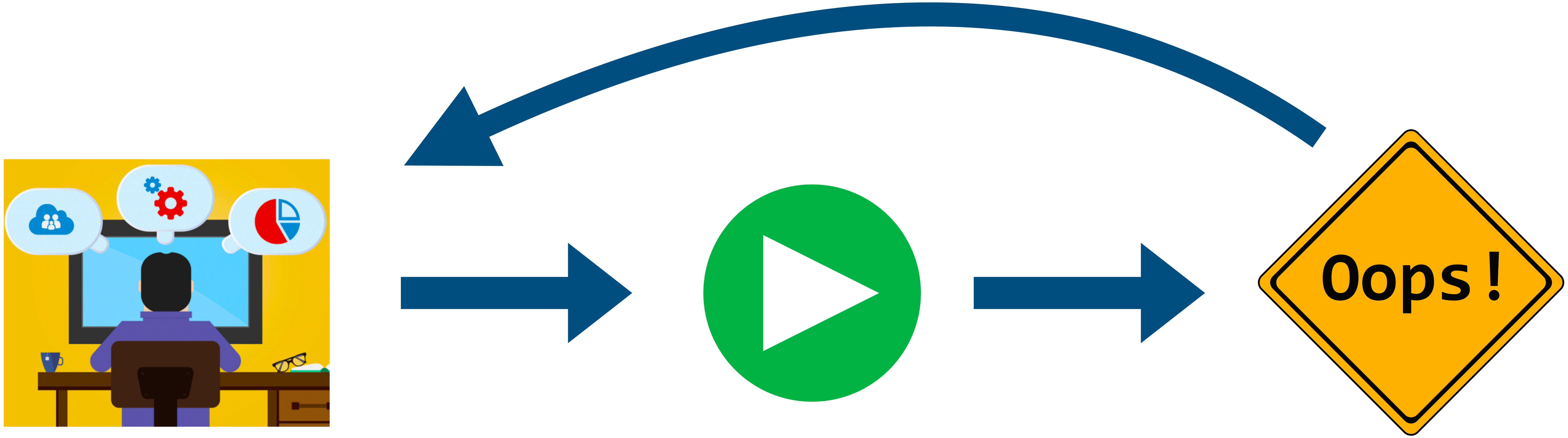
Tijs van der Storm (@tvdstorm / storm@cwi.nl)

Introduction

about me…

- Groepsleider Software Analysis & Transformation; Prof in Groningen
- Wij doen onder andere onderzoek naar:
  - hoe beter programmeertalen ***te maken*** (language engineering)
  - hoe ***betere*** programmeertalen te maken (language design)
- We gebruiken hiervoor Rascal, een ***meta***-programmeertaal
  - https://www.rascal-mpl.org/
- Vandaag: *live* programmeren

Programmeren



back to the drawing board…

Code schrijven

RUN!

Fout ☹

Programmeren…

…is als boogschieten

Richten, schieten, missen; opnieuw. Olympische sport…

Live programmeren

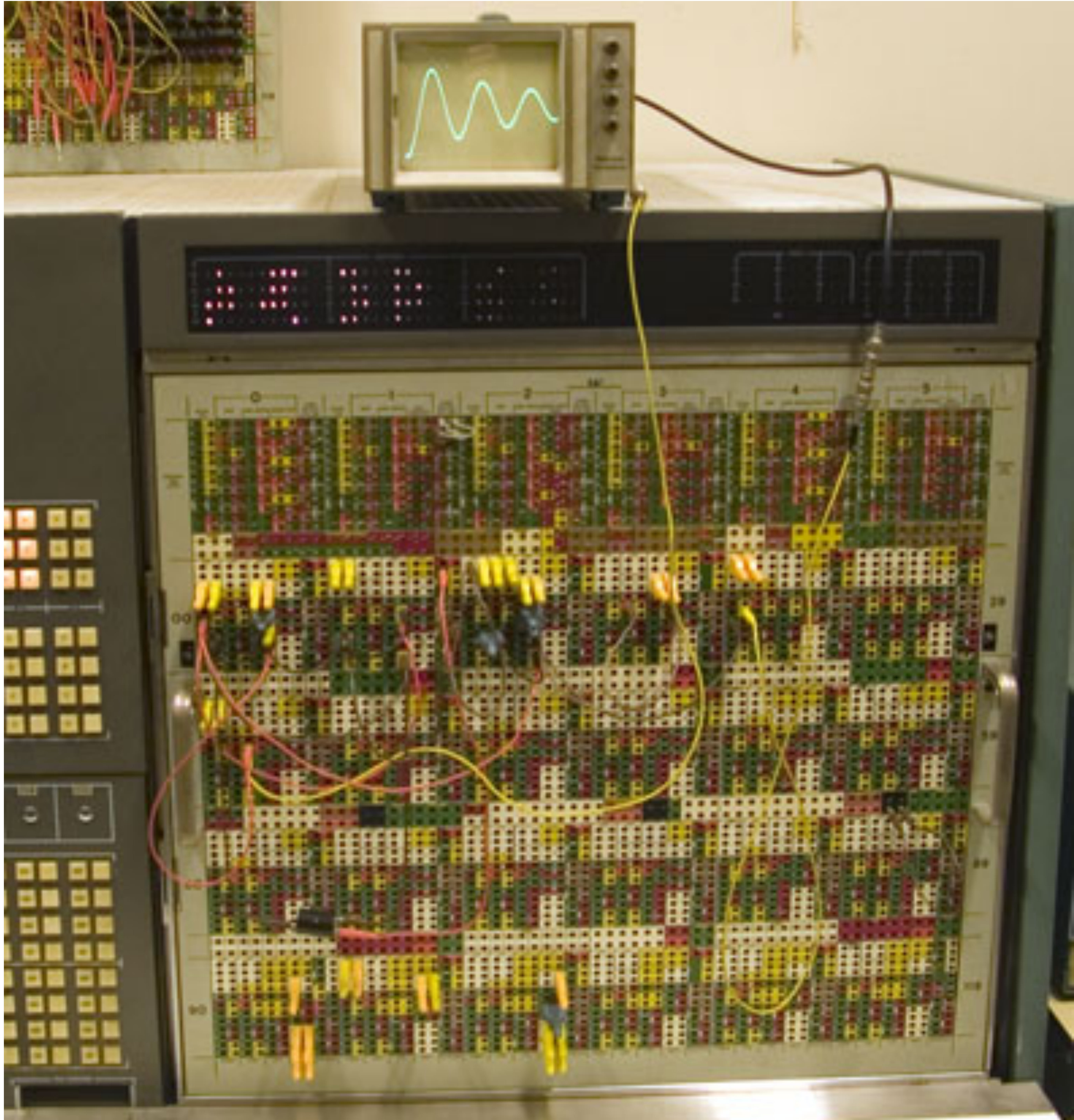
Vloeiende ervaring, continue feedback: “kinderspel”

Programmeren als boetseren

- Er zijn geen stappen (edit, compile, run; repeat)
- Voortdurende, continue feedback
- Effect van handeling meteen zichtbaar
- Maakproces en materiaal vloeien naadloos in elkaar over

- Wat als programmeren zo kon zijn?

Waar komt “live” programming vandaan?



Doel van live programmeren: “gulf of evaluation” overbruggen

Intermezzo: little languages

domain-specific languages (DSLs)

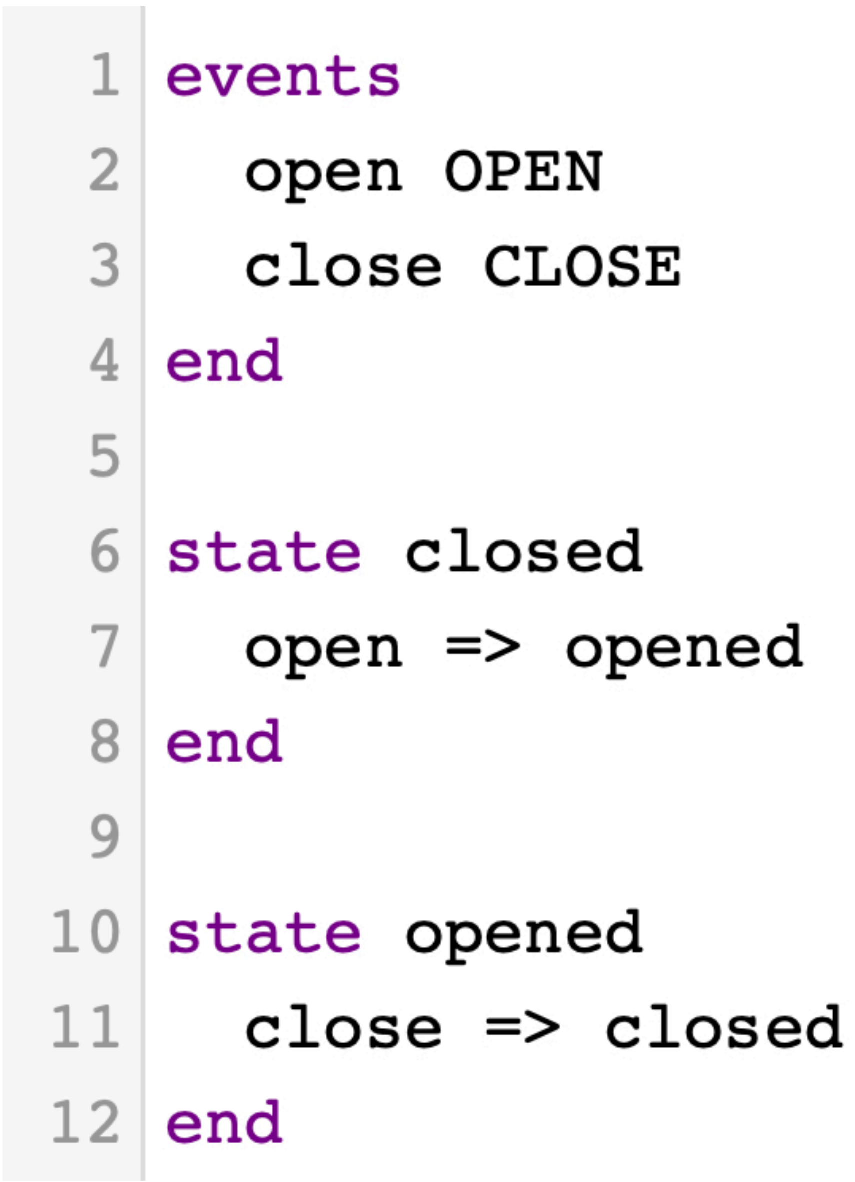
- General purpose programmeertalen:
  - "groot", algemeen gebruik
  - Bv. Java, C, Swift, Javascript, etc.
- Domein-specifiek:
  - "klein", toegespitst
  - Bv. HTML, SQL, TeX, CSS, Graphviz, …
- Kan groot effect op kwaliteit software hebben

general purpose:

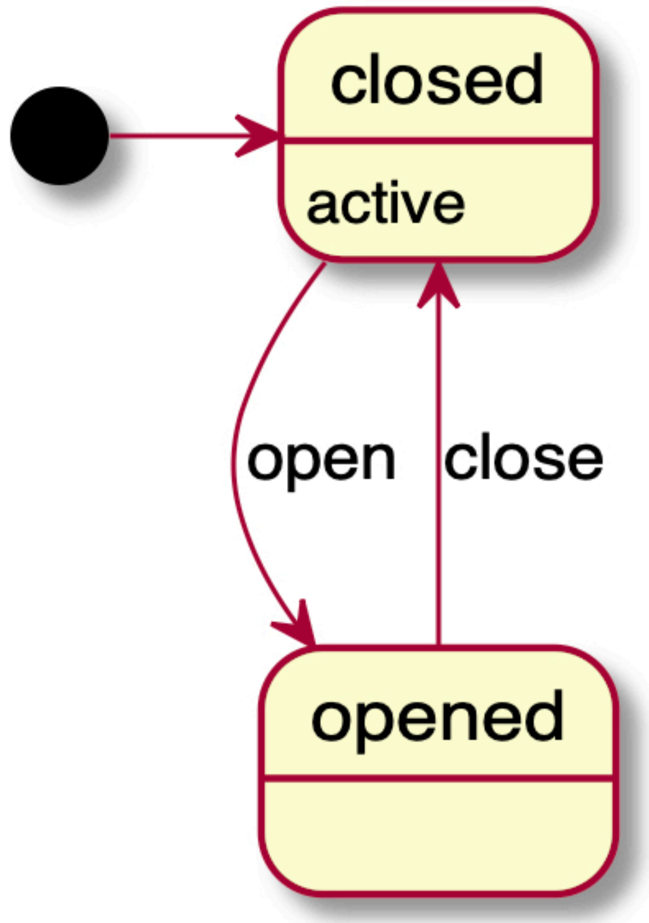
domein-specifiek:

Live demo

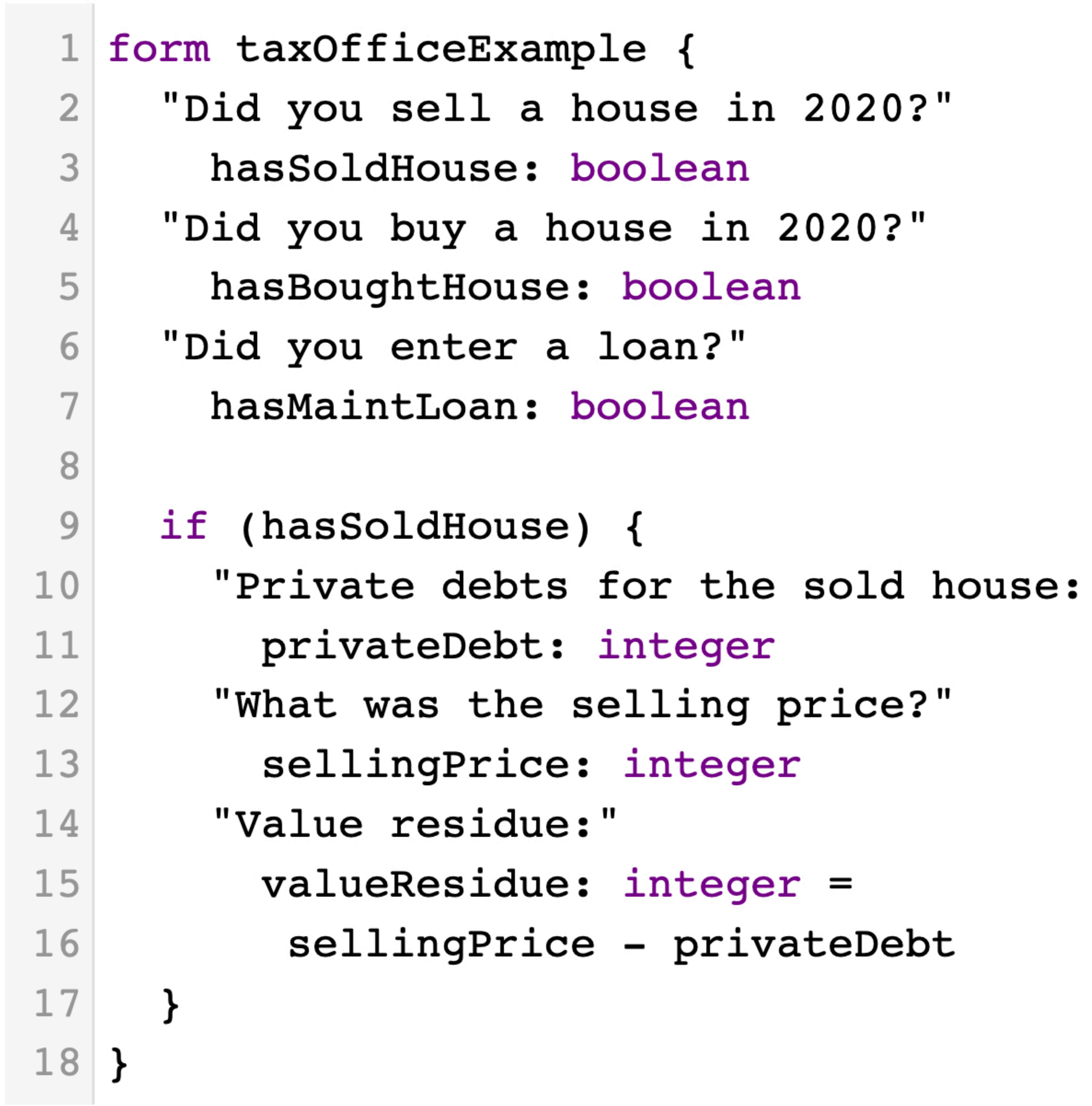
State machines



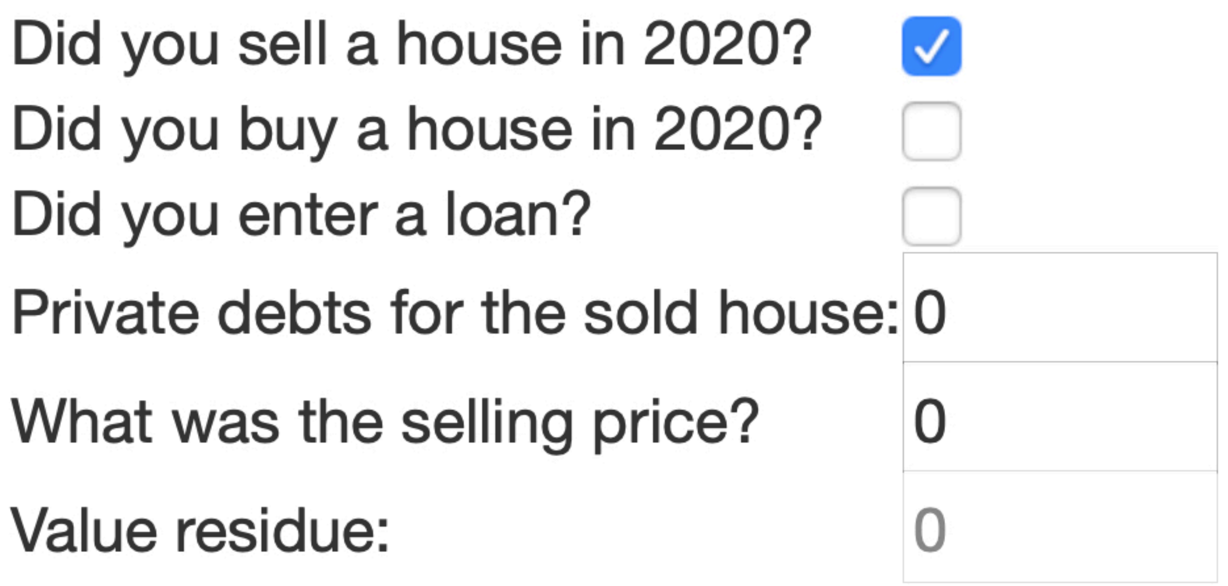
```
events
  open OPEN
  close CLOSE
end

state closed
  open => opened
end

state opened
  close => closed
end
```



Interactieve enquêtes



```
form taxOfficeExample {
  "Did you sell a house in 2020?"
    hasSoldHouse: boolean
  "Did you buy a house in 2020?"
    hasBoughtHouse: boolean
  "Did you enter a loan?"
    hasMaintLoan: boolean

  if (hasSoldHouse) {
    "Private debts for the sold house:
      privateDebt: integer
    "What was the selling price?"
      sellingPrice: integer
    "Value residue:"
      valueResidue: integer =
       sellingPrice - privateDebt
  }
}
```



Did you sell a house in 2020? ☑
Did you buy a house in 2020? ☐
Did you enter a loan? ☐
Private debts for the sold house: 0
What was the selling price? 0
Value residue: 0

Het maken van live programmeertalen

Waarom is dit een uitdaging?

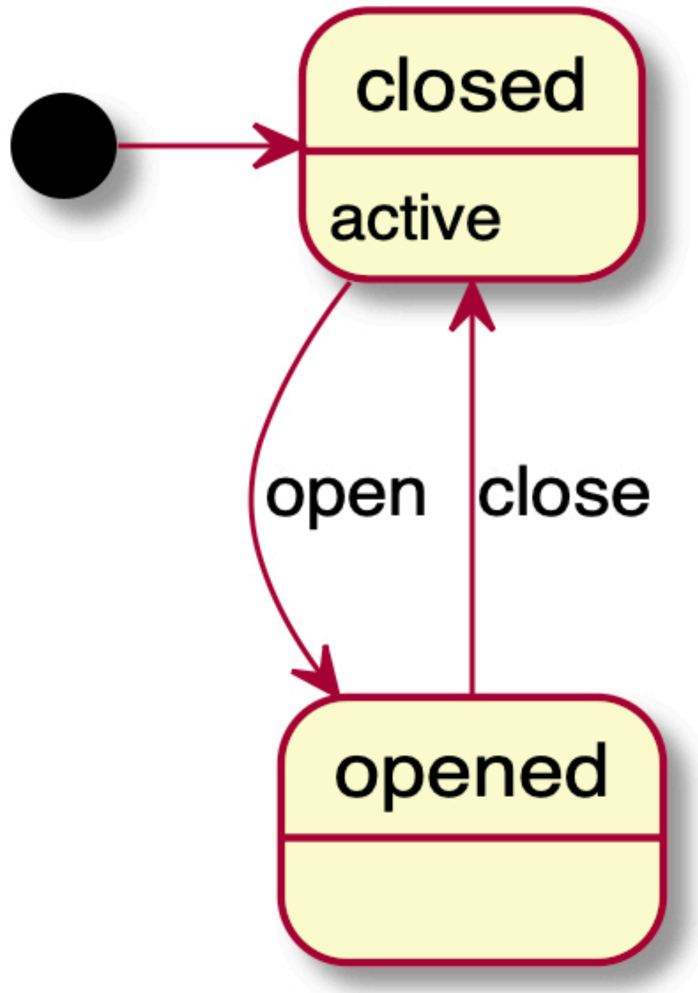
- Broncode is een soort sjabloon
- Het draaien van het programma is het invullen van het sjabloon (met data)
- Als je het sjabloon verandert, wat moet er dan met de ingevulde data gebeuren?
- Die data moeten gemigreerd worden, zonder het programma te stoppen.

Sjabloon…

```
state closed
  open => opened
end

state opened
  close => closed
end
```

“alle deuren”

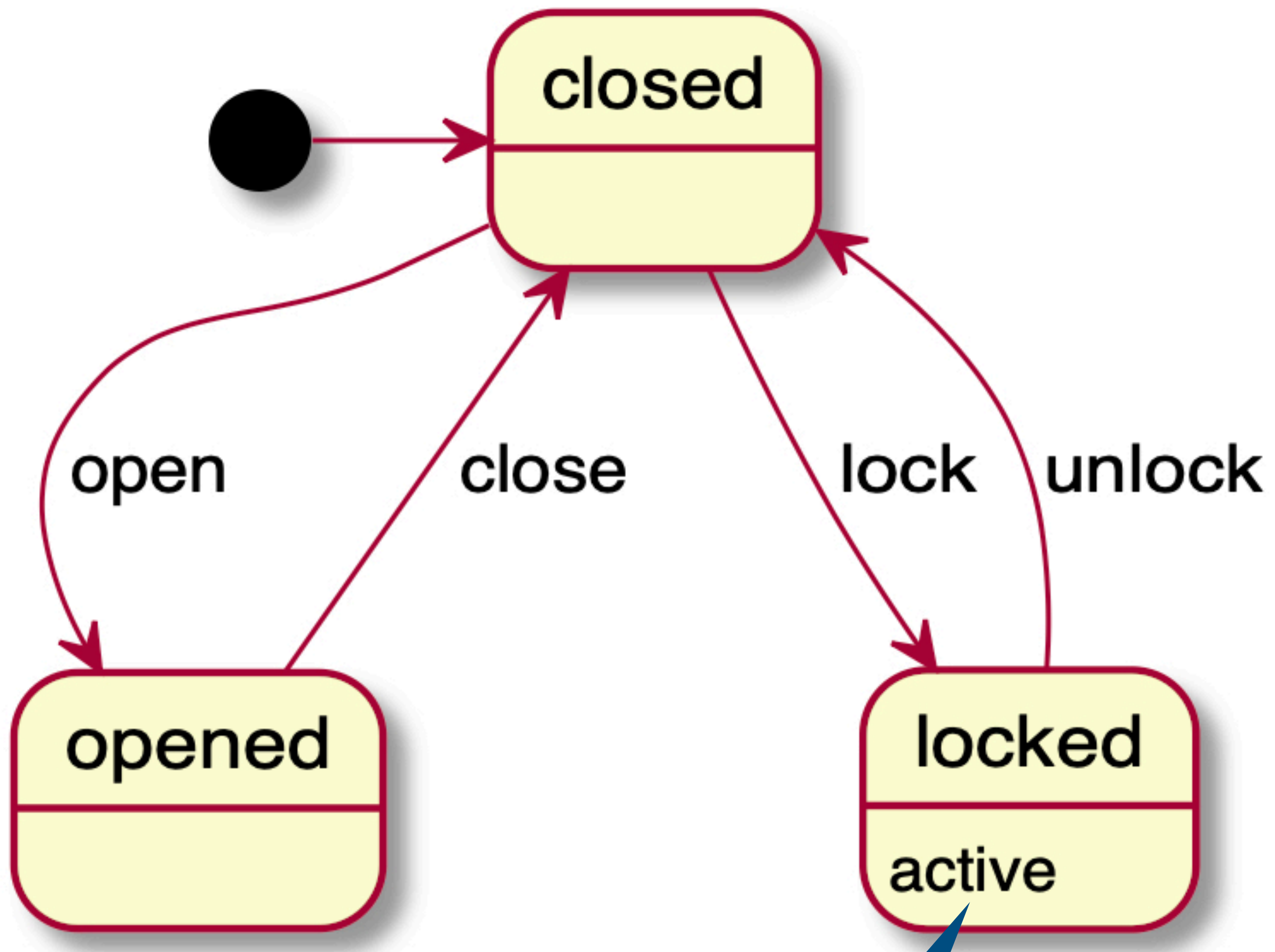
…ingevuld



“een specifieke deur”

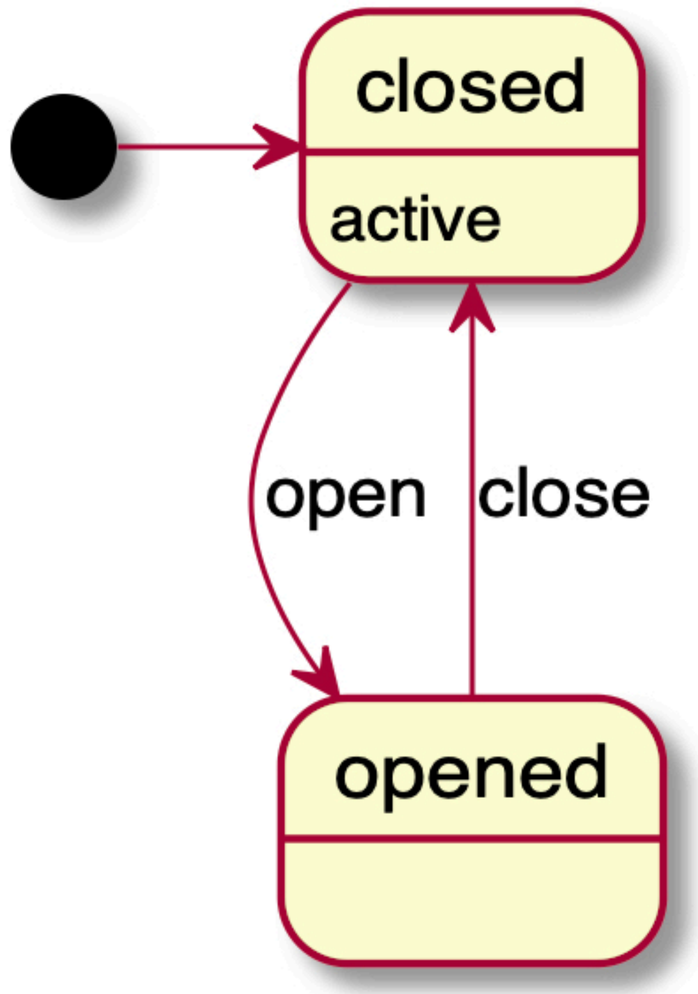
Voorbeeld van toestandsmigratie

```
state closed
  open => opened
  lock => locked
end

state opened
  close => closed
end

state locked
  unlock => closed
end
```



welke state actief is, is run-time data

```
state closed
  open => opened
end

state opened
  close => closed
end
```

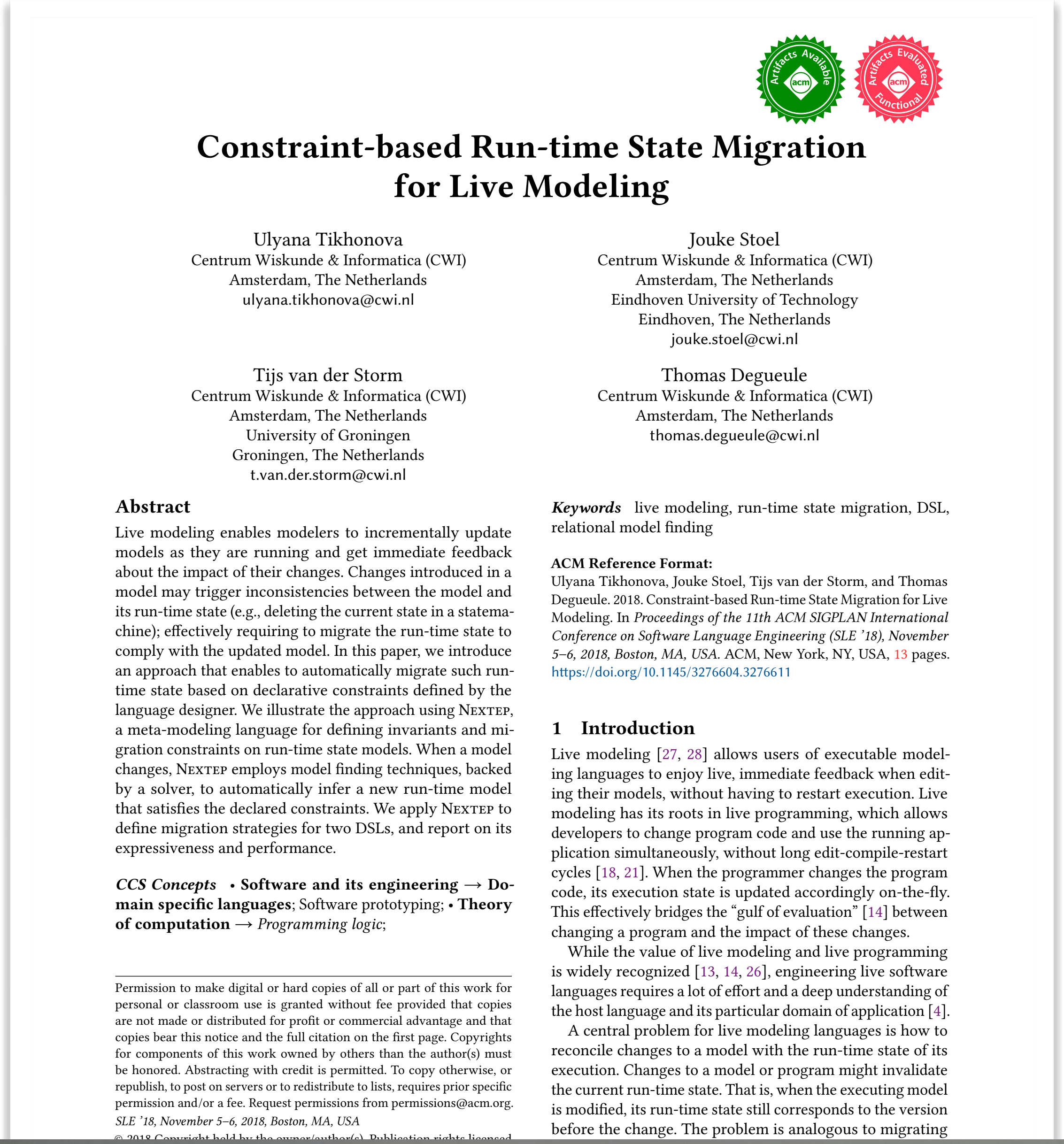


welke state moet nu actief zijn!?! 🤔

verwijder “locked”

Onderzoek naar live programming

fundamentele technieken om programmeertalen live te maken



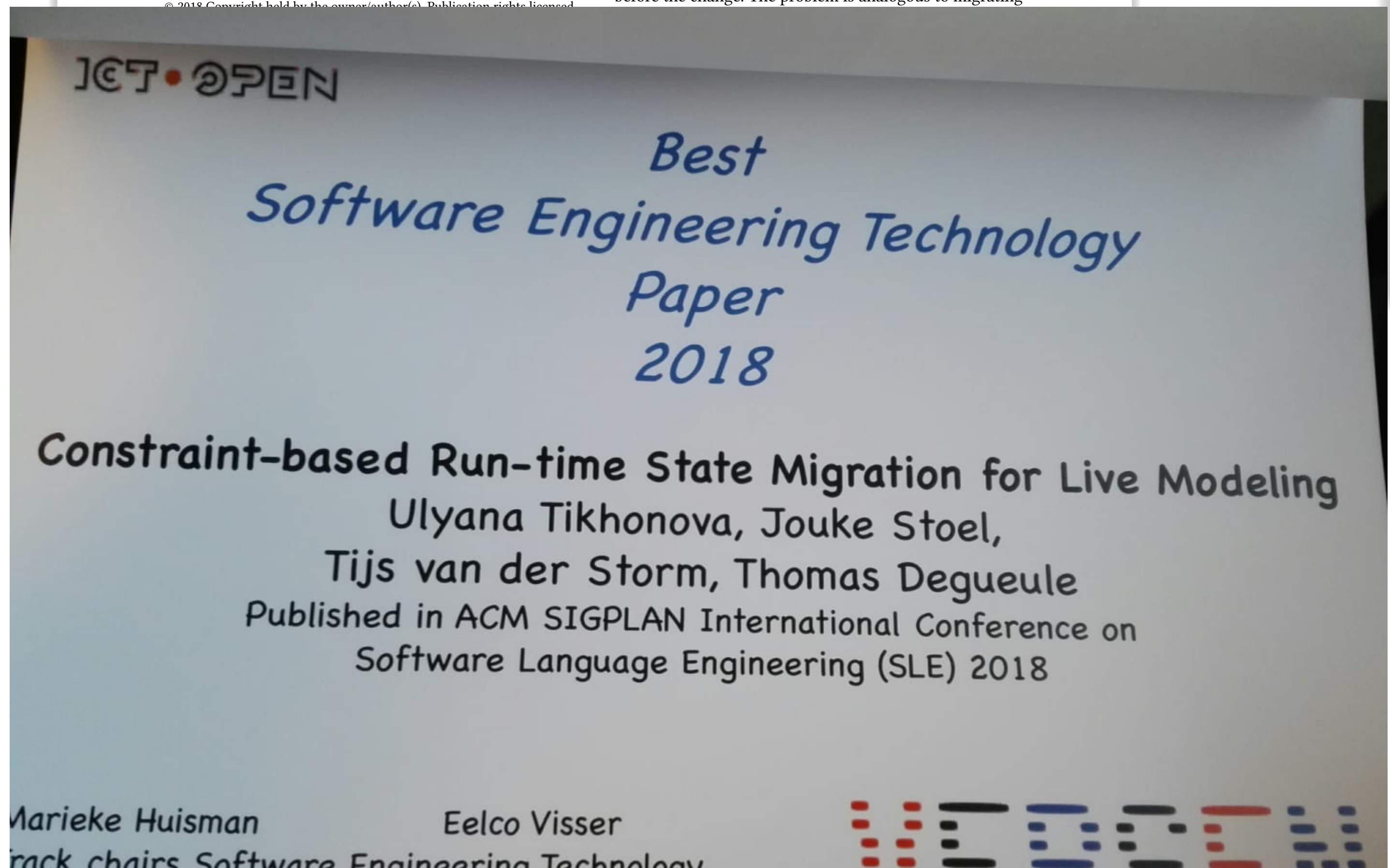
Constraint-based Run-time State Migration for Live Modeling

Ulyana Tikhonova
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
ulyana.tikhonova@cwi.nl

Jouke Stoel
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
Eindhoven University of Technology
Eindhoven, The Netherlands
jouke.stoel@cwi.nl

Tijs van der Storm
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
University of Groningen
Groningen, The Netherlands
t.van.der.storm@cwi.nl

Thomas Degueule
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
thomas.degueule@cwi.nl

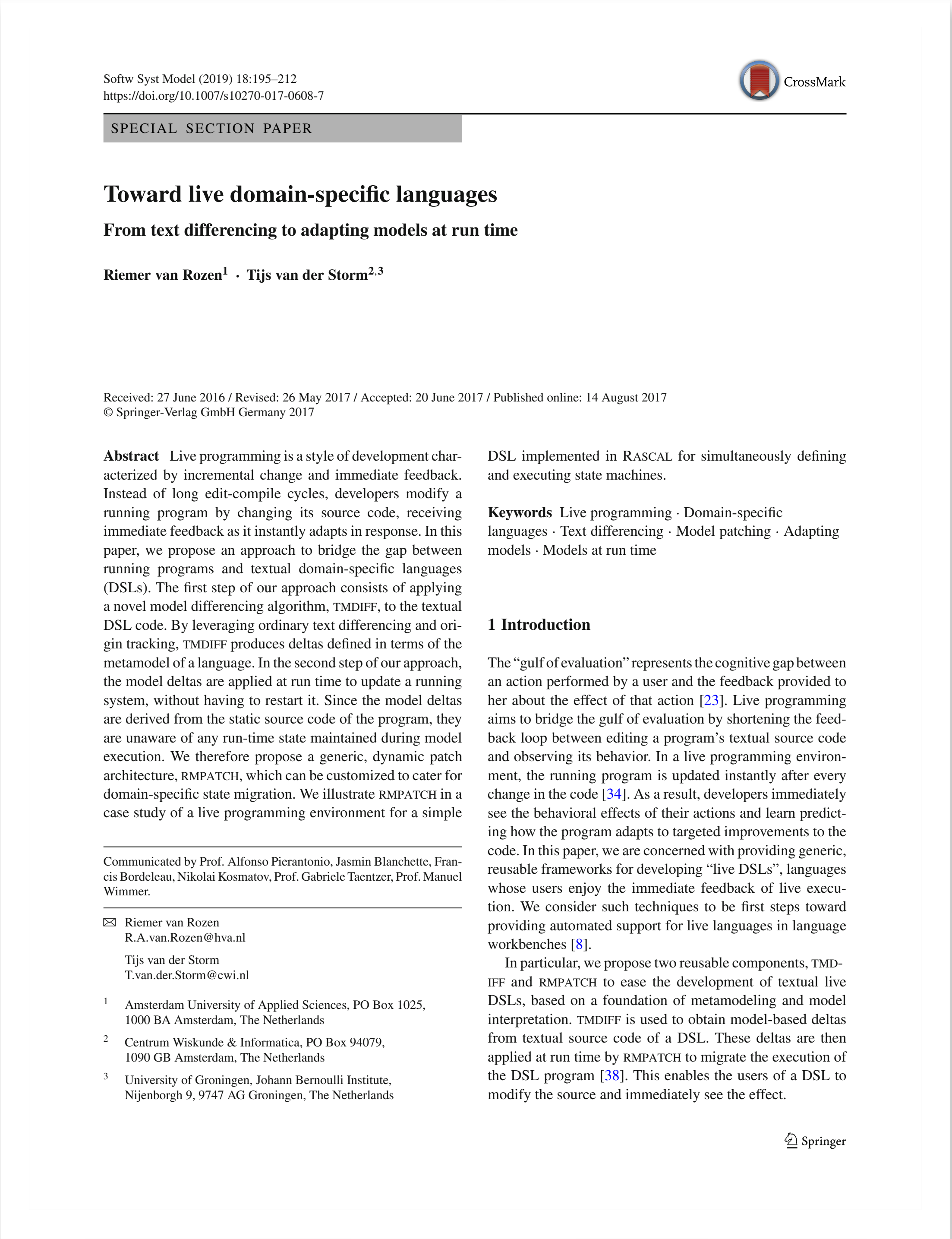


ICT•OPEN

Best
Software Engineering Technology
Paper
2018

Constraint-based Run-time State Migration for Live Modeling
Ulyana Tikhonova, Jouke Stoel,
Tijs van der Storm, Thomas Degueule
Published in ACM SIGPLAN International Conference on
Software Language Engineering (SLE) 2018

Marieke Huisman
Eelco Visser
rack chairs Software Engineering Technology



Softw Syst Model (2019) 18:195–212
https://doi.org/10.1007/s10270-017-0608-7

SPECIAL SECTION PAPER

Toward live domain-specific languages

From text differencing to adapting models at run time

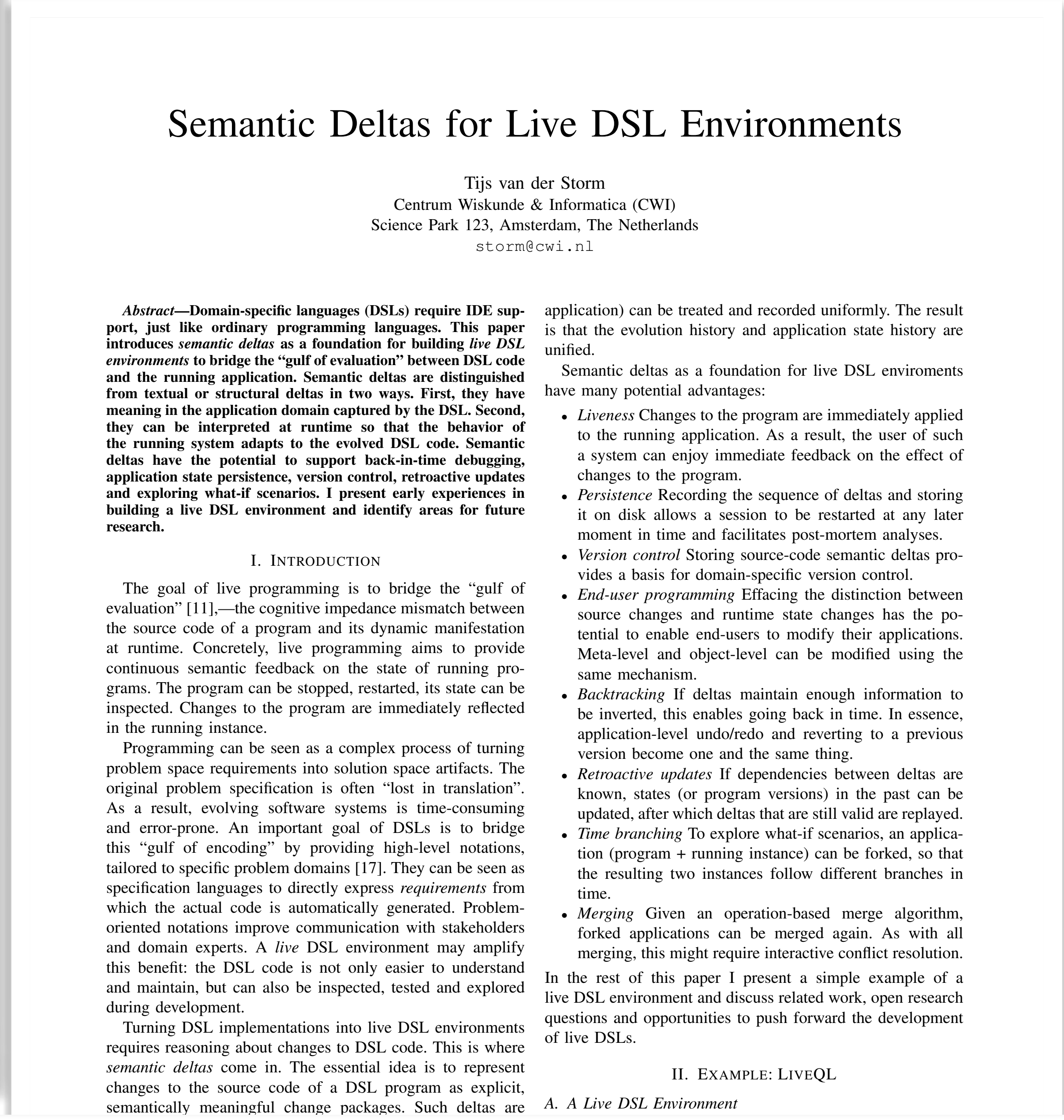
Riemer van Rozen[1] · Tijs van der Storm[2,3]

Received: 27 June 2016 / Revised: 26 May 2017 / Accepted: 20 June 2017 / Published online: 14 August 2017

Adapting Game Mechanics with Micro-Machinations

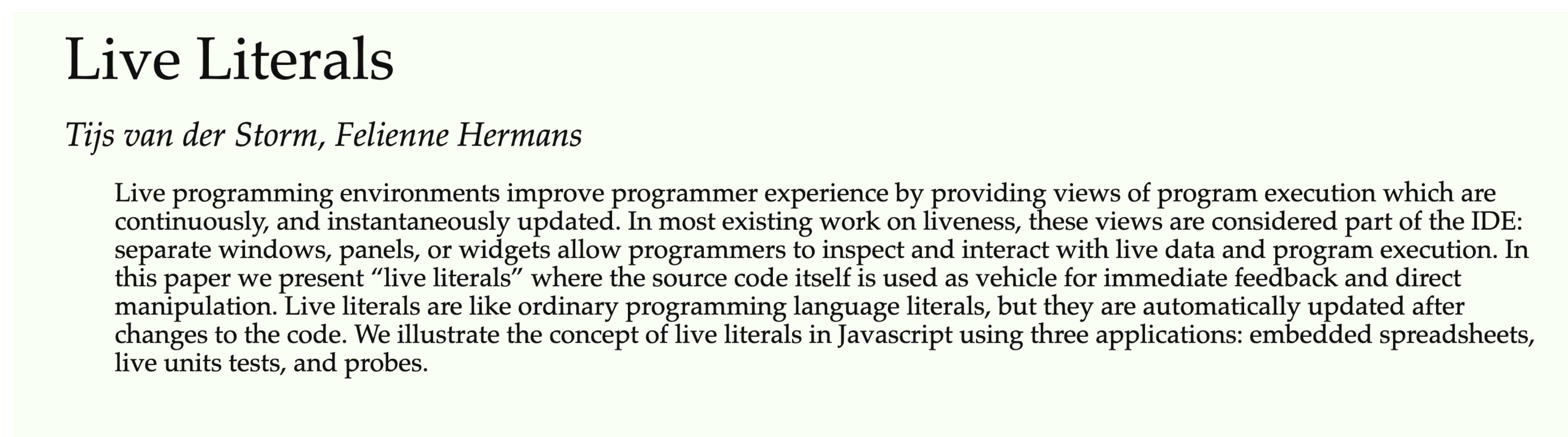
Riemer van Rozen
Amsterdam University of Applied Sciences
Duivendrechtsekade 36-38 1096 AH
Amsterdam, The Netherlands
r.a.van.rozen@hva.nl

Joris Dormans
Amsterdam University of Applied Sciences
Duivendrechtsekade 36-38 1096 AH
Amsterdam, The Netherlands
j.dormans@hva.nl



Semantic Deltas for Live DSL Environments

Tijs van der Storm
Centrum Wiskunde & Informatica (CWI)
Science Park 123, Amsterdam, The Netherlands
storm@cwi.nl



Live Literals

Tijs van der Storm, Felienne Hermans

Live programming environments improve programmer experience by providing views of program execution which are continuously, and instantaneously updated. In most existing work on liveness, these views are considered part of the IDE: separate windows, panels, or widgets allow programmers to inspect and interact with live data and program execution. In this paper we present "live literals" where the source code itself is used as vehicle for immediate feedback and direct manipulation. Live literals are like ordinary programming language literals, but they are automatically updated after changes to the code. We illustrate the concept of live literals in Javascript using three applications: embedded spreadsheets, live units tests, and probes.

Live programming

“programmeren als boetseren”

- Live programming verkleint de kloof tussen programma en executie
- Geen “edit, compile, run; repeat”-cycle, maar een vloeiende overgang
- Programmeurs ervaren snellere feedback en directere omgang met programma’s
- Actief onderzoek naar generieke principes om talen meer “live” te maken.

CWI
Centrum Wiskunde & Informatica