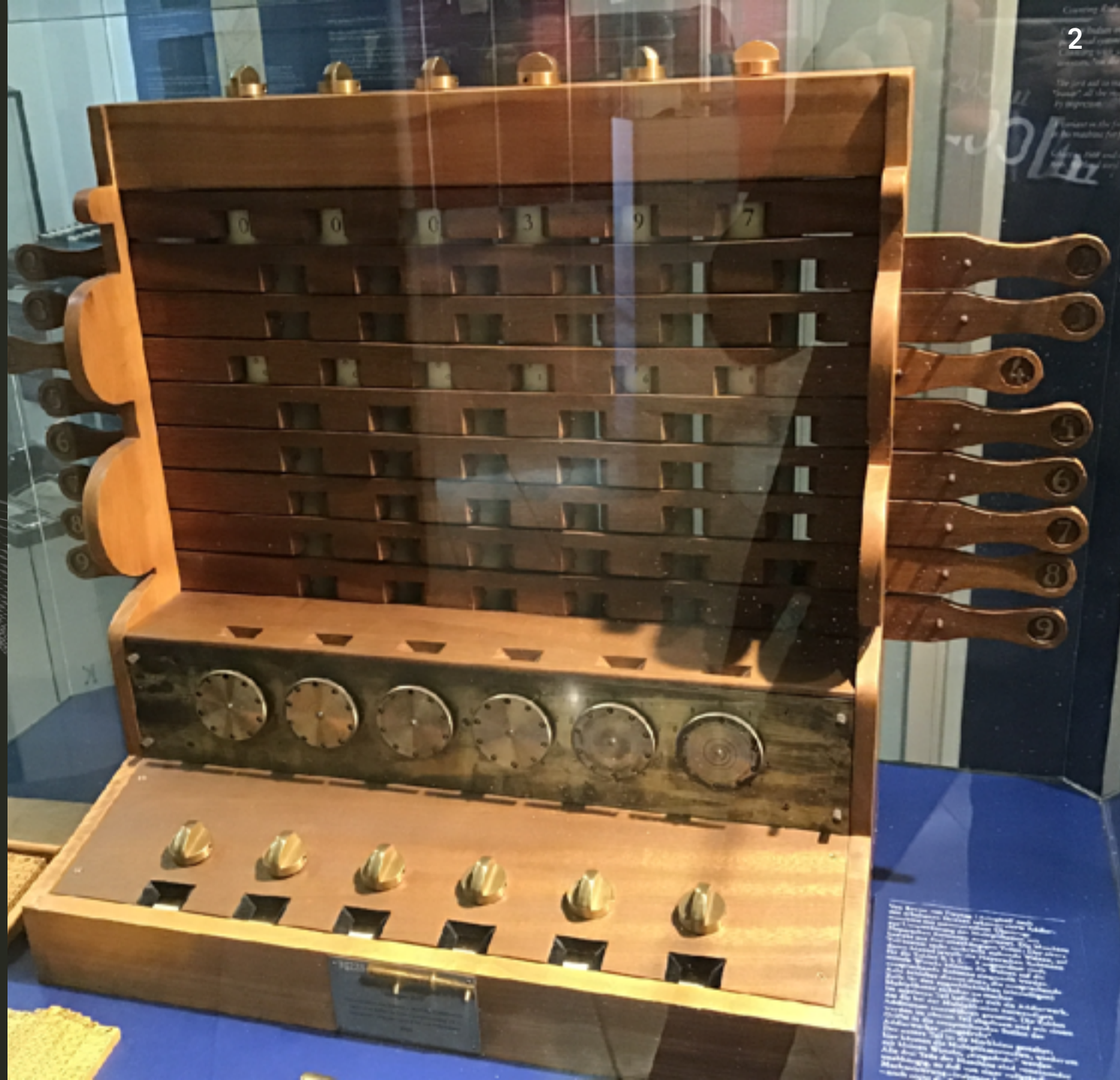
Responsible Language Design

MODELSWARD, 20 February 2023

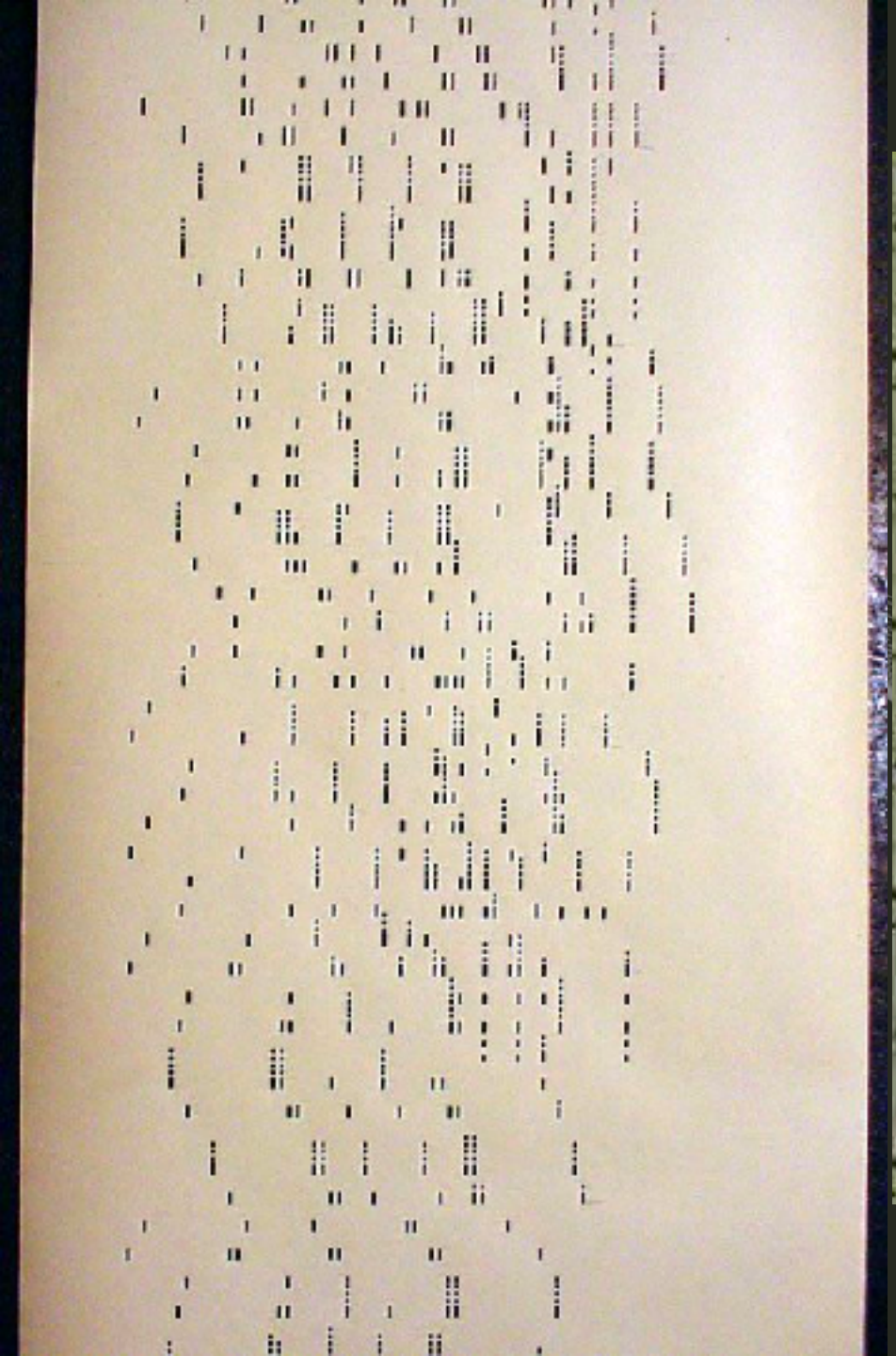
Dr. Vadim Zaytsev aka @grammarware, prof of Software Evolution

UNIVERSITY OF TWENTE.


ÆTATIS SUÆ
1610

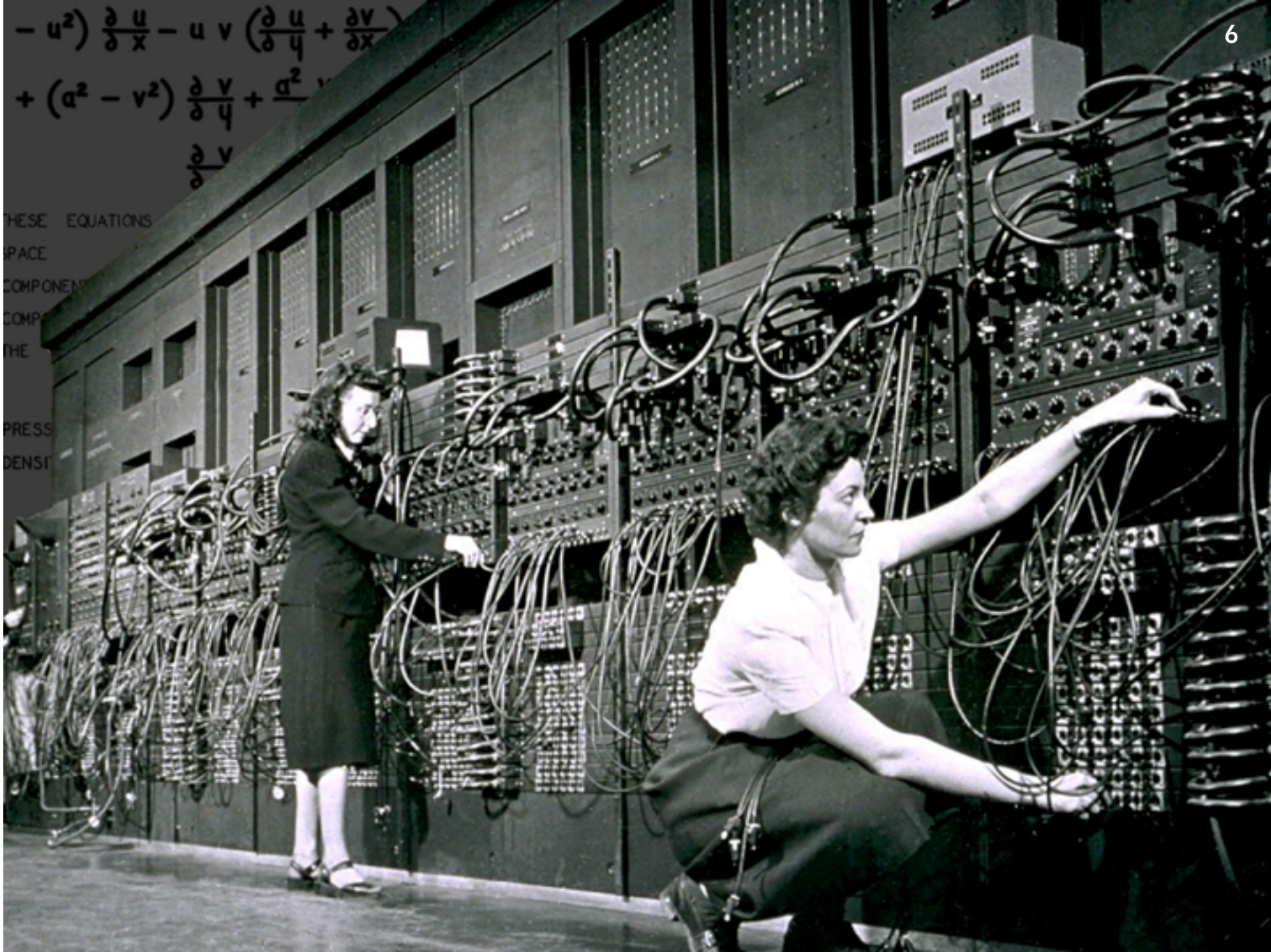


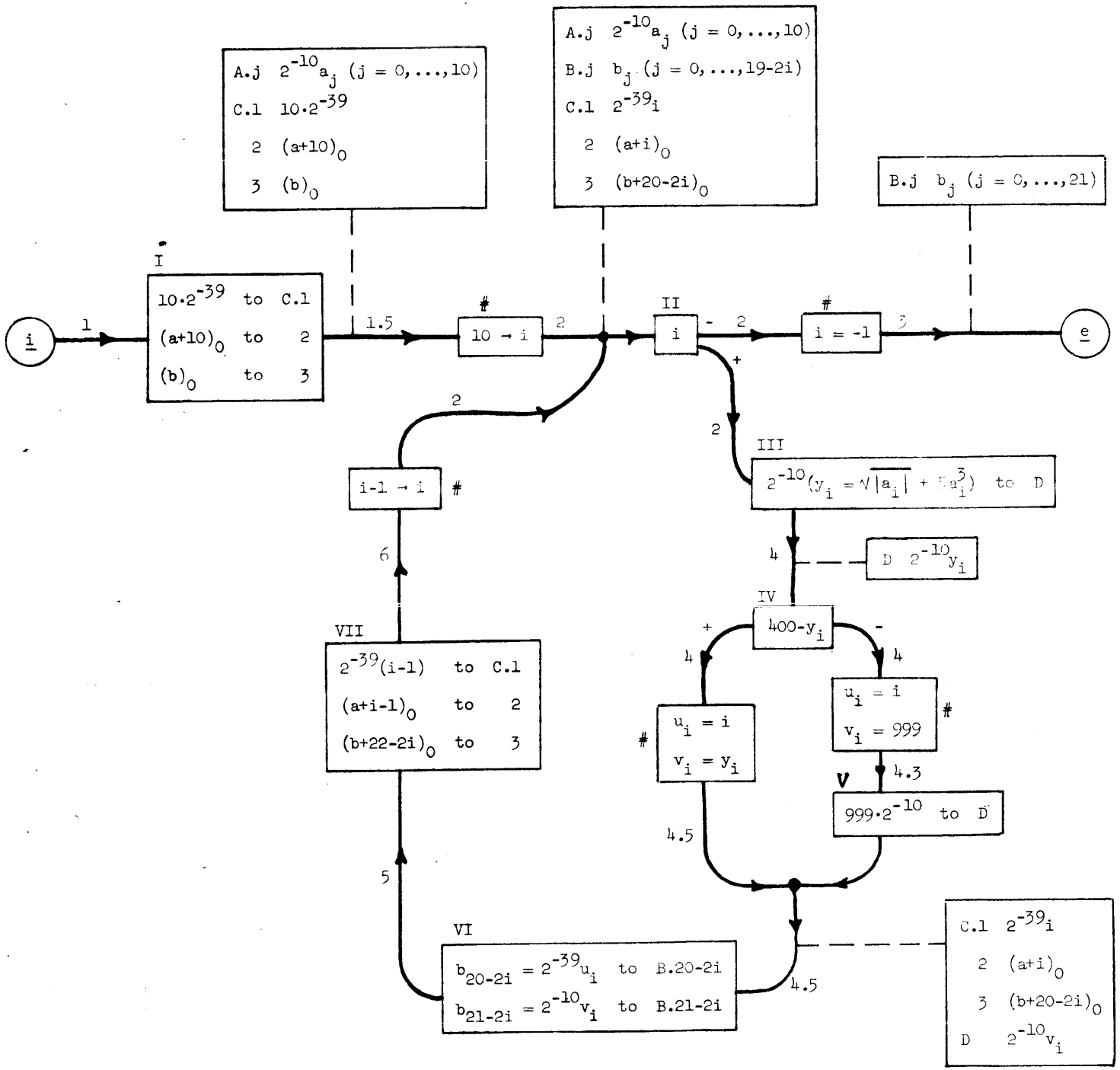

GRANDIOLA
STEMS DRESDEN




MidiPiano
MidiPiano (MIDI File Player/Recorder)
Pause
Stop
Repeat
Piano Roll
Midi Event
Options
Key:
Speed:
Volume
Play

UNIVERSITY OF TWENTE. | Formal Methods & Tools

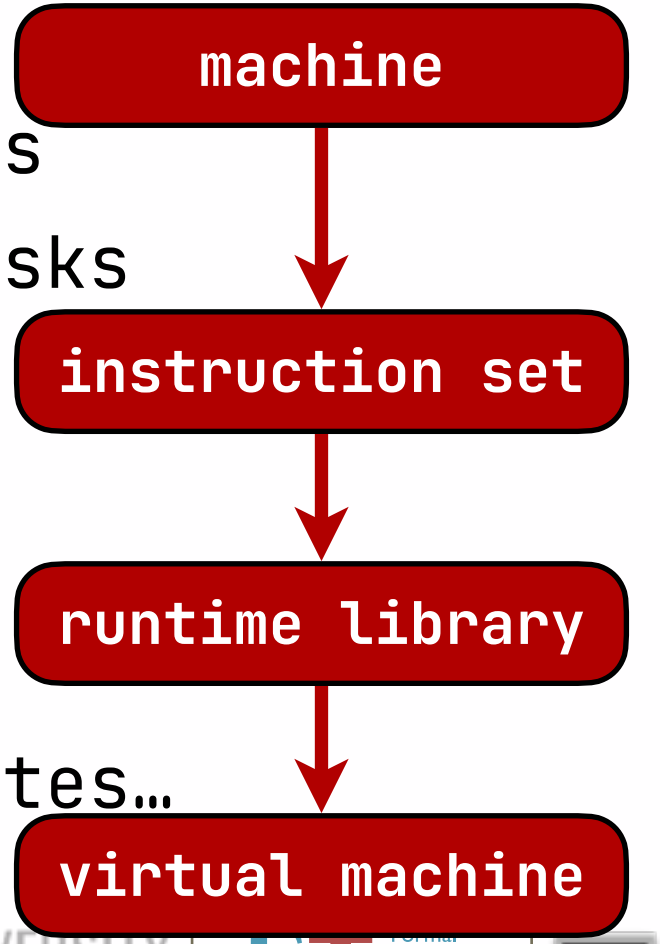

$- u^2) \frac{\partial u}{\partial x} - u v \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$
$+ (a^2 - v^2) \frac{\partial v}{\partial y} + \frac{a^2}{}$
$\frac{\partial}{\partial}$ v
HESE EQUATIONS
PACE
COMPONEN
COMP
THE
PRESS
DENSI


A.j $2^{-10}a_j$ (j = 0,...,10)
C.1 $10\cdot2^{-39}$
2 $(a+10)_0$
3 $(b)_0$
A.j $2^{-10}a_j$ (j = 0,...,10)
B.j b_j (j = 0,...,19-2i)
C.1 $2^{-39}i$
2 $(a+i)_0$
3 $(b+20-2i)_0$
B.j b_j (j = 0,...,21)
I
$10\cdot2^{-39}$ to C.1
$(a+10)_0$ to 2
$(b)_0$ to 3
i
e
10 → i
II
i
i = -1
III
$2^{-10}(y_i = \sqrt{|a_i|} + 5a_i^3)$ to D
D $2^{-10}y_i$
IV
400-y_i
u_i = i
v_i = y_i
u_i = i
v_i = 999
V
$999\cdot2^{-10}$ to D
i-1 → i
VII
2^{-39}(i-1) to C.1
$(a+i-1)_0$ to 2
$(b+22-2i)_0$ to 3
VI
$b_{20-2i} = 2^{-39}u_i$ to B.20-2i
$b_{21-2i} = 2^{-10}v_i$ to B.21-2i
C.1 $2^{-39}i$
2 $(a+i)_0$
3 $(b+20-2i)_0$
D $2^{-10}v_i$


IBM

Emergent Language Design

1. Human computes, machine helps
2. Machine computes, program dictates steps
3. Machine computes, program prescribes tasks
4. Machine runs tasks, program instructs
5. Machine runs tasks,
   instructions grouped into functions,
   program calls functions
6. … functions form a machine, which computes…



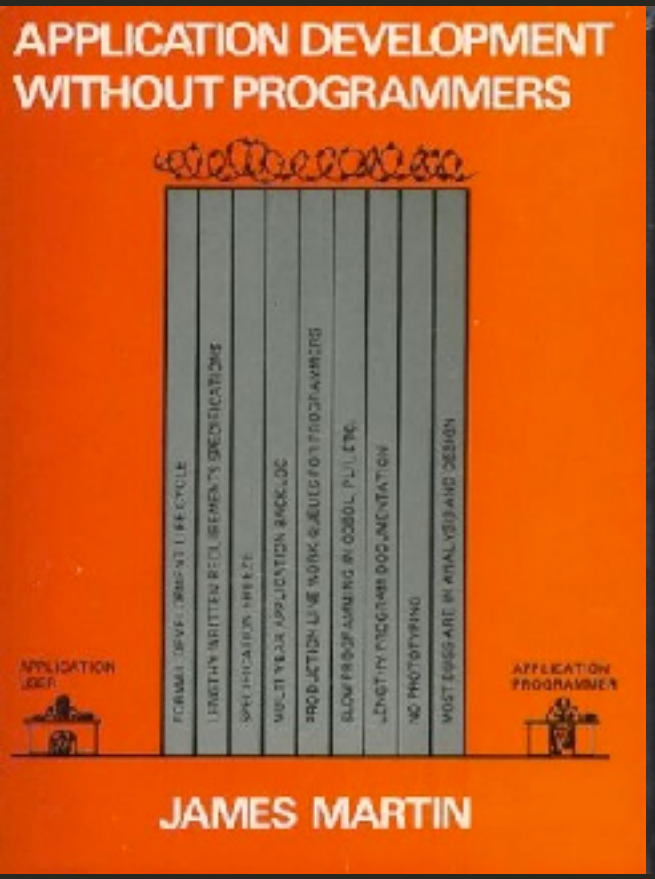
Black Boxes

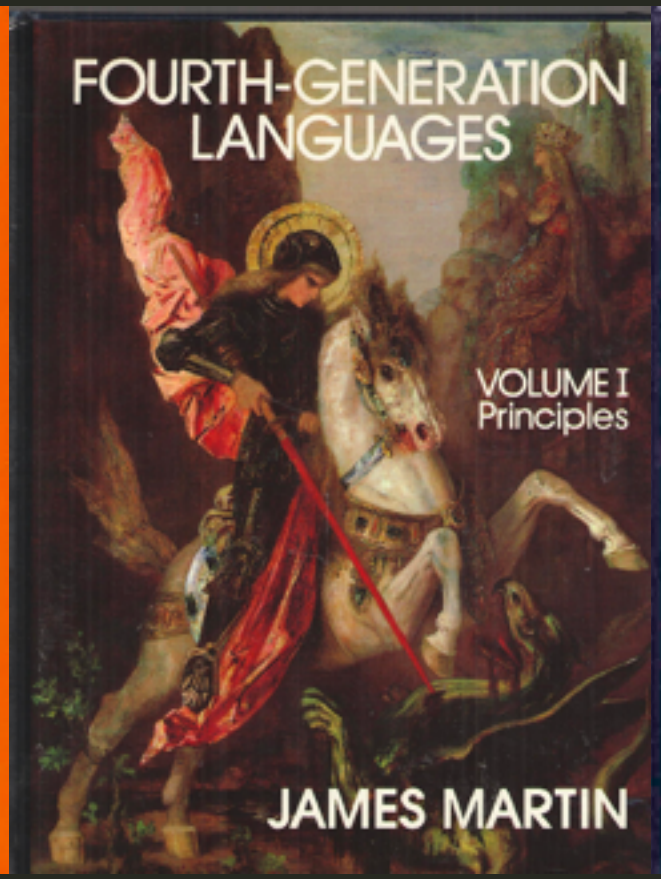
```
MOVE ACTION-COMMAND-0326 TO STRING-0081FP
MOVE SPACES TO ERROR-ENCOUNTERED-SW OF GLOBDATA
CALL 'TIRFUPPR' USING IEF-RUNTIME-PARM1
                      IEF-RUNTIME-PARM2
                      GLOBDATA
                      STRING-0081AT
                      OUTPUT-STRING-0001AT
IF STATUS-FLAG OF GLOBDATA NOT = SPACES GOBACK END-IF
```

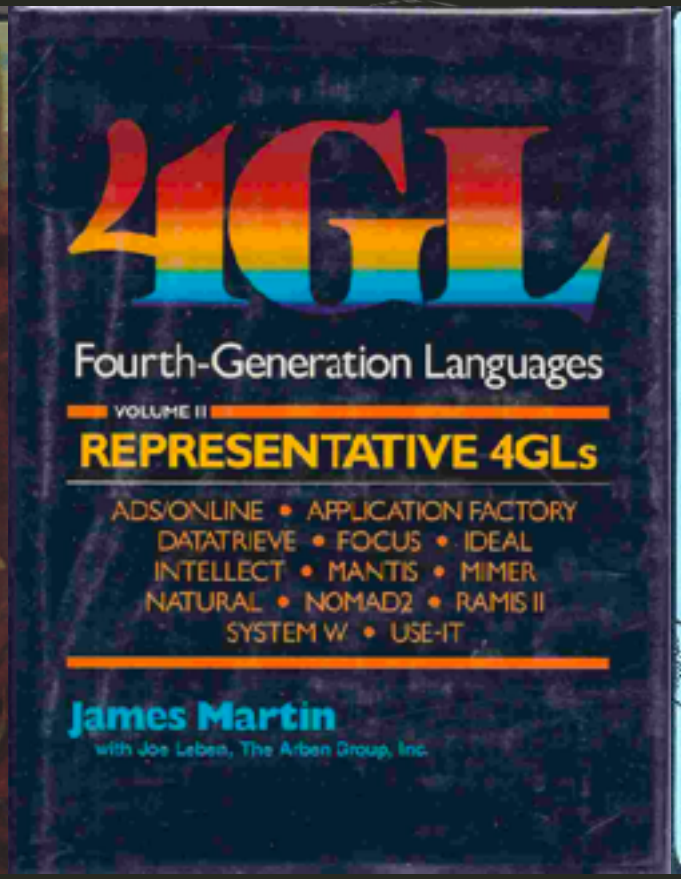

Black Boxes

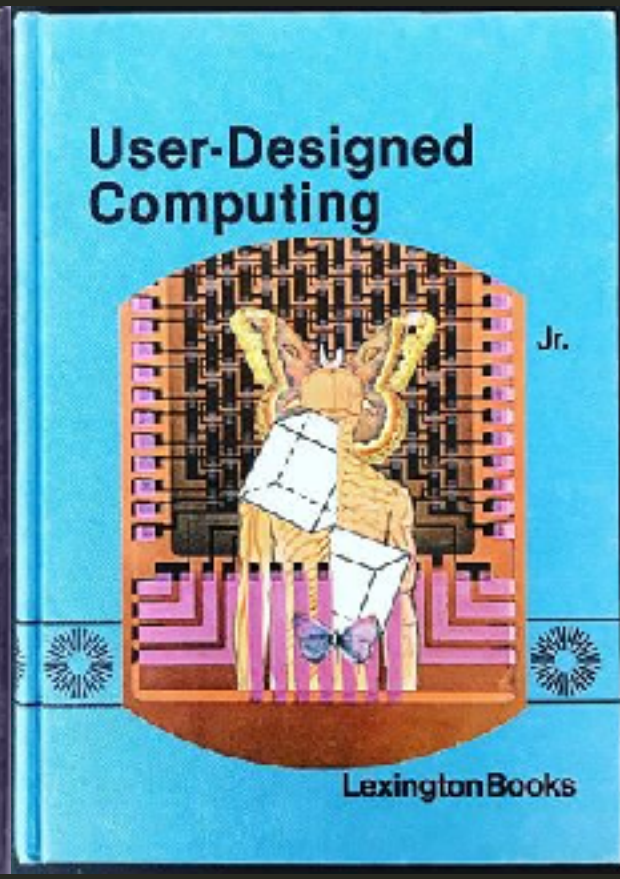
raincode LABS
compiler experts

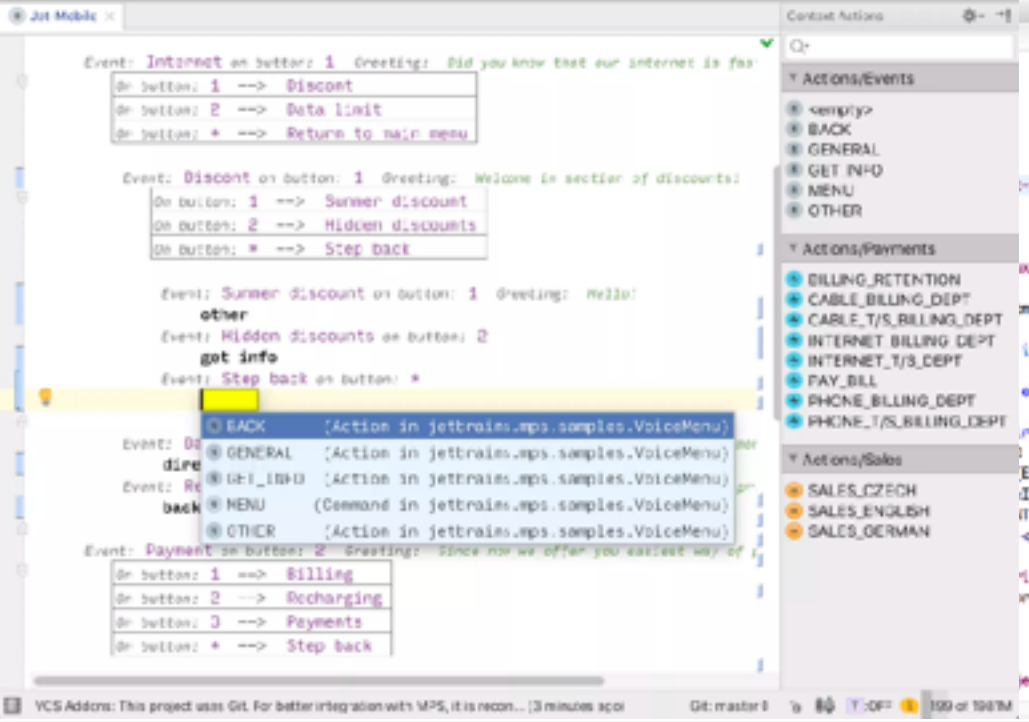
```
MOVE ACTION-COMMAND-0326 TO STRING-0081FP
MOVE SPACES TO ERROR-ENCOUNTERED-SW OF GLOBDATA
MOVE FUNCTION UPPER-CASE(ACTION-COMMAND-0326)
  TO OUTPUT-STRING-0001FP
COMPUTE OUTPUT-STRING-0001FL = FUNCTION LENGTH
        (ACTION-COMMAND-0326)
IF STATUS-FLAG OF GLOBDATA NOT = SPACES GOBACK END-IF
```

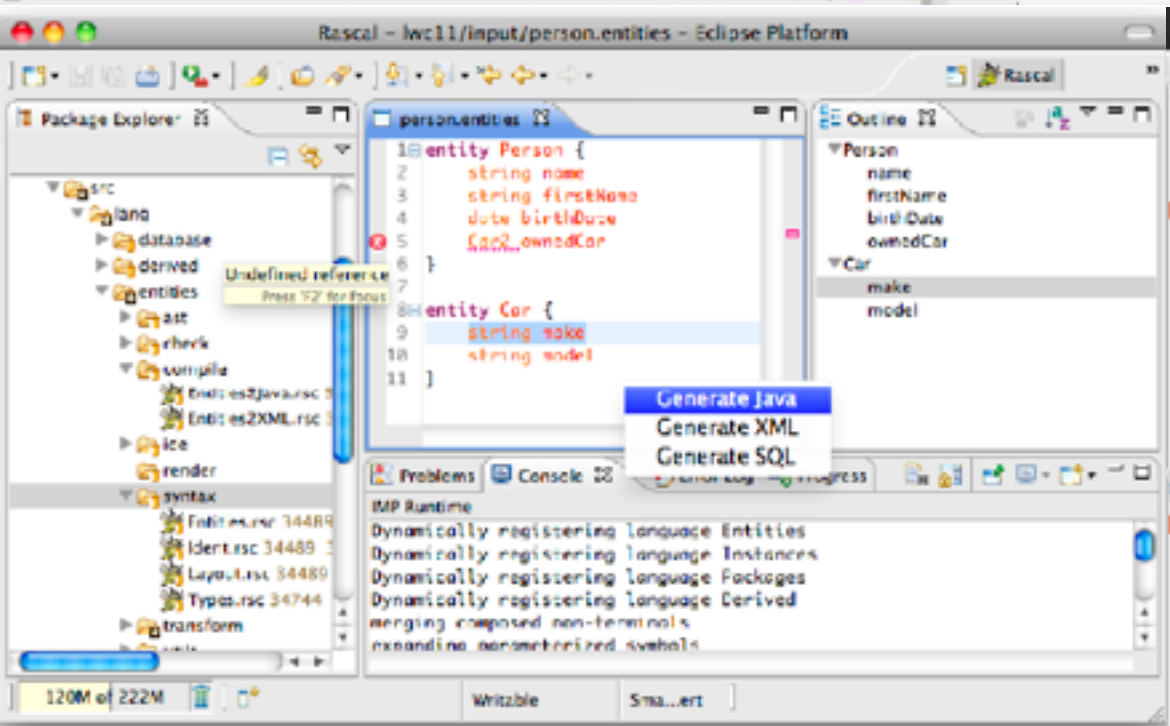

APPLICATION DEVELOPMENT
WITHOUT PROGRAMMERS
JAMES MARTIN


FOURTH-GENERATION
LANGUAGES
VOLUME I
Principles
JAMES MARTIN

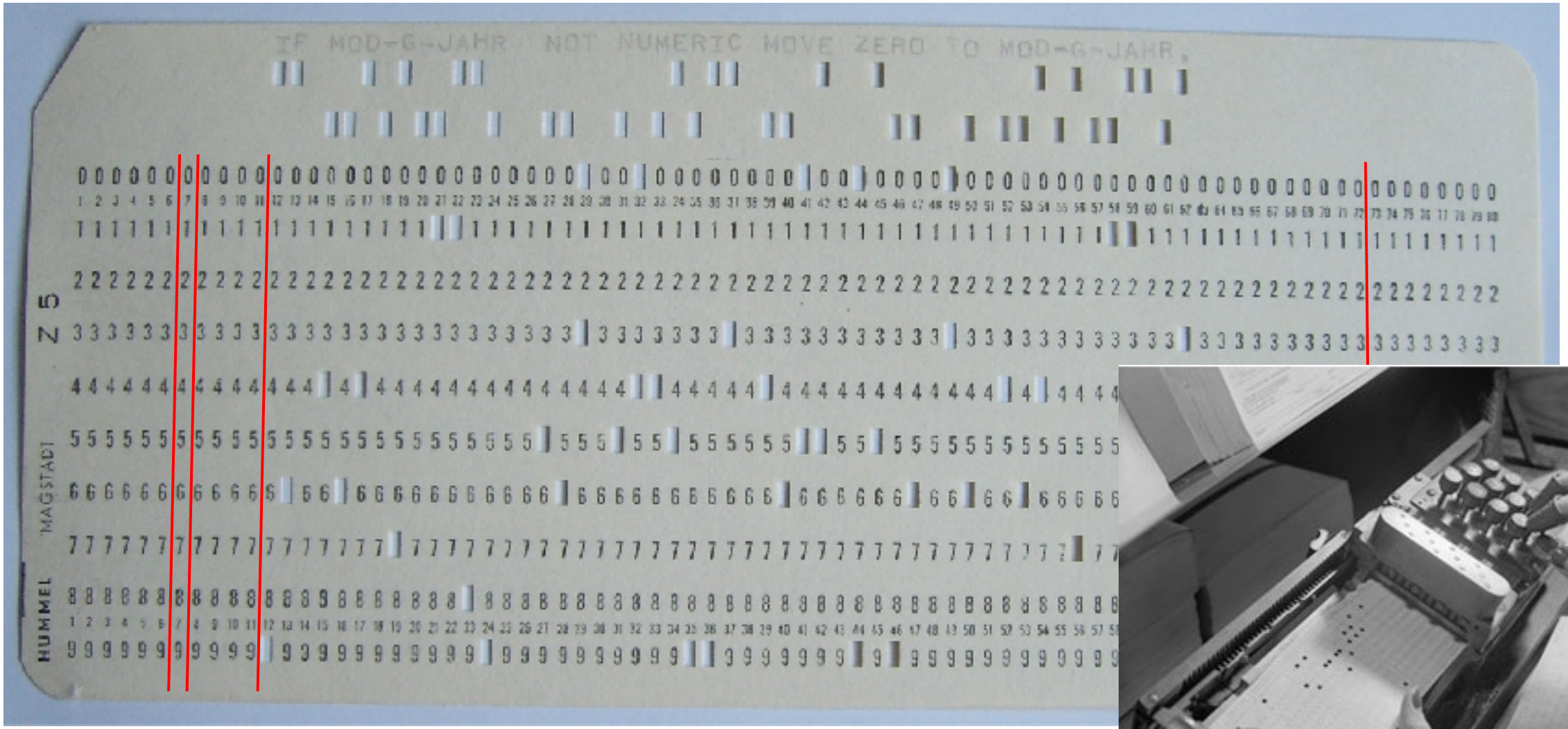

4GL
Fourth-Generation Languages
VOLUME II
REPRESENTATIVE 4GLs
ADS/ONLINE • APPLICATION FACTORY
DATATRIEVE • FOCUS • IDEAL
INTELLECT • MANTIS • MIMER
NATURAL • NOMAD2 • RAMIS II
SYSTEM W • USE-IT
James Martin
with Joe Leben, The Arben Group, Inc.


User-Designed
Computing
Jr.
Lexington Books


Event: Internet on button: 1 Greeting: Did you know that our internet is fas
On button: 1 --> Discont
On button: 2 --> Data limit
On button: * --> Return to main menu
Event: Discont on button: 1 Greeting: Welcome in section of discounts:
On button: 1 --> Summer discount
On button: 2 --> Hidden discounts
On button: * --> Step back
Event: Summer discount on button: 1 Greeting: Hello!
other
Event: Hidden discounts on button: 2
got info
Event: Step back on button: *
BACK (Action in jetbrains.mps.samples.VoiceMenu)
GENERAL (Action in jetbrains.mps.samples.VoiceMenu)
GET_INFO (Action in jetbrains.mps.samples.VoiceMenu)
MENU (Command in jetbrains.mps.samples.VoiceMenu)
OTHER (Action in jetbrains.mps.samples.VoiceMenu)
Event: Payment on button: 2 Greeting: Since now we offer you easiest way of
On button: 1 --> Billing
On button: 2 --> Recharging
On button: 3 --> Payments
On button: * --> Step back
Actions/Events
BACK
GENERAL
GET INFO
MENU
OTHER
Actions/Payments
BILLING_RETENTION
CABLE_BILLING_DEPT
CABLE_T/S_BILLING_DEPT
INTERNET BILLING DEPT
INTERNET_T/S_DEPT
PAY_BILL
PHONE_BILLING_DEPT
PHONE_T/S_BILLING_DEPT
Actions/Sales
SALES_CZECH
SALES_ENGLISH
SALES_GERMAN


Rascal – lwc11/input/person.entities – Eclipse Platform
Package Explorer
src
lang
database
derived
entities
ast
check
compile
ice
render
syntax
transform
Undefined reference
Press 'F2' for focus
person.entities
entity Person {
string name
string firstName
date birthDate
Car2 ownedCar
}
entity Car {
string make
string model
}
Outline
Person
name
firstName
birthDate
ownedCar
Car
make
model
Generate Java
Generate XML
Generate SQL
Problems
Console
IMP Runtime
Dynamically registering language Entities
Dynamically registering language Instances
Dynamically registering language Packages
Dynamically registering language Derived
merging composed non-terminals
expanding parametrized symbols
Writable

What is a LOC?



Rainer Gerhards, CC-BY-SA, 2006, Punch-card-cobol

From Lines to Statements

- How to separate statements without punchcards?
- ALGOL, Pascal, …
  - separators: ; or '
- C, Java, C#, …
  - terminators: ;
- C++: a b((c)d);
- JS: invisible semicolons
- A mix of **philology** and **disambiguation**

Dislexic Impact

```
begin procedure p(k) ;  integer k ; k := 1 ; end

begin procedure p(k) ;; integer k ; k := 1 ; end


10101 DO 101 I = 1, 101

10101 DO 101 I = 1. 101
```

cf. https://dl.acm.org/doi/pdf/10.5555/1061500.1061508

Dislexic Impact

```
begin procedure p(k) ;  integer k ; k := 1 ; end
```

```
begin procedure p(k) ;; integer k ; k := 1 ; end
```

```
10101 DO 101 I = 1, 101
```

```
10101 DO 101 I = 1. 101
```

Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

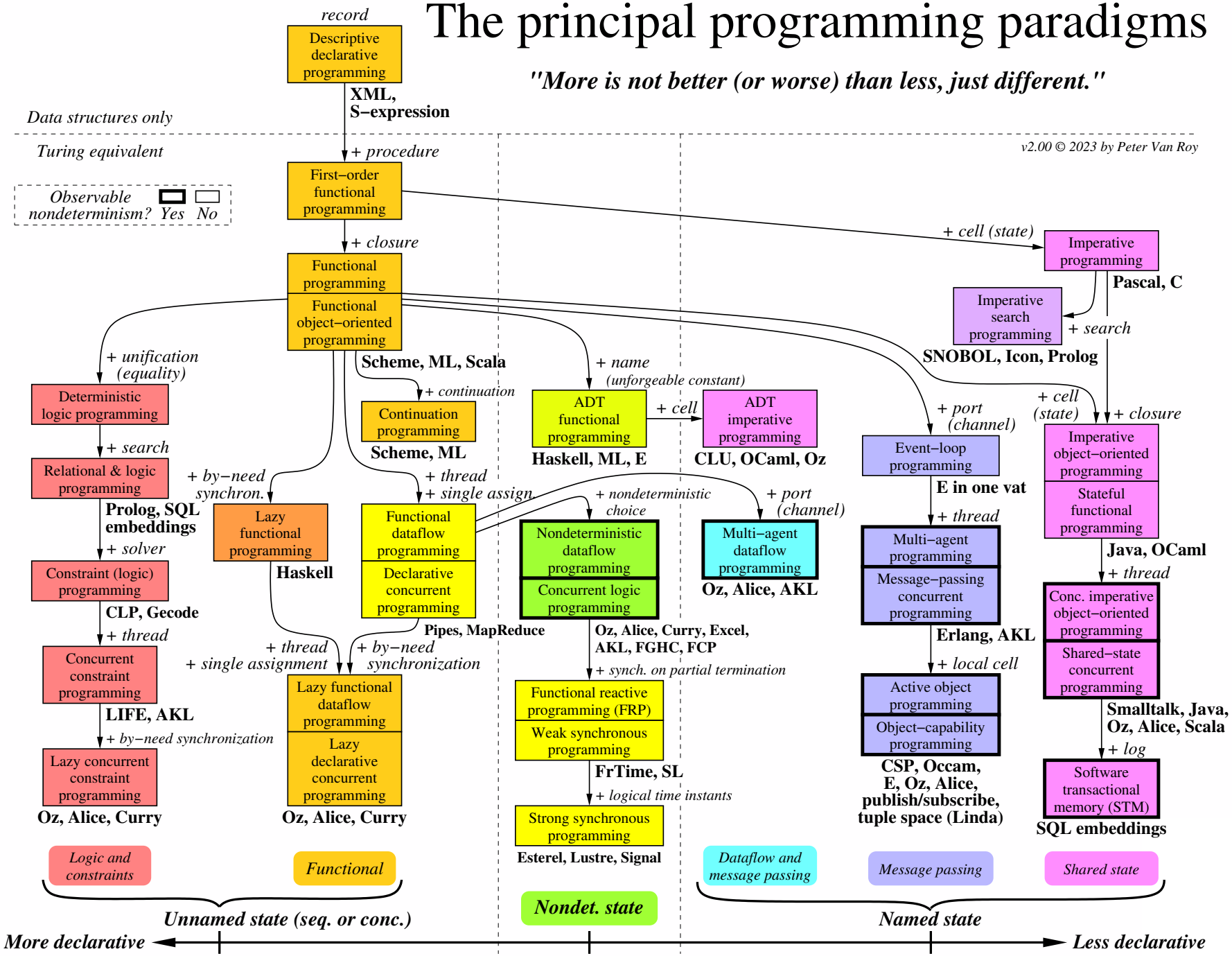
John Backus
IBM Research Laboratory, San Jose

UNIVERSITY OF TWENTE.

Formal Methods & Tools

The principal programming paradigms

"More is not better (or worse) than less, just different."



Explanations

See "Concepts, Techniques, and Models of Computer Programming".

The chart classifies programming paradigms according to their kernel languages (the small core language in which all the paradigm's abstractions can be defined). Kernel languages are ordered according to the creative extension principle: a new concept is added when it cannot be encoded with only local transformations. Two languages that implement the same paradigm can nevertheless have very different "flavors" for the programmer, because they make different choices about what programming techniques and styles to facilitate. All paradigms that support functional programming can also support object−oriented programming.

When a language is mentioned under a paradigm, it means that part of the language is intended (by its designers) to support the paradigm without interference from other paradigms. It does not mean that there is a perfect fit between the language and the paradigm. It is not enough that libraries have been written in the language to support the paradigm. The language's kernel language should support the paradigm. When there is a family of related languages, usually only one member of the family is mentioned to avoid clutter. The absence of a language does not imply any kind of value judgment.

State is the ability to remember information, or more precisely, to store a sequence of values in time. Its expressive power is strongly influenced by the paradigm that contains it. We distinguish four levels of expressiveness, which differ in whether the state is unnamed or named, deterministic or nondeterministic, and sequential or concurrent. The least expressive is functional programming (threaded state, e.g., DCGs and monads: unnamed, deterministic, and sequential). Adding concurrency gives declarative concurrent programming (e.g., synchrocells: unnamed, deterministic, and concurrent). Adding nondeterministic choice gives concurrent logic programming (which uses stream mergers: unnamed, nondeterministic, and concurrent). Adding ports or cells, respectively, gives message passing or shared state (both are named, nondeterministic, and concurrent). Nondeterminism is important for real−world interaction (e.g., client/server). Named state is important for modularity.

Axes orthogonal to this chart are typing, aspects, and domain−specificity. Typing is not completely orthogonal: it has some effect on expressiveness. Aspects should be completely orthogonal, since they are part of a program's specification. A domain−specific language should be definable in any paradigm (except when the domain needs a particular concept).

Metaprogramming is another way to increase the expressiveness of a language. The term covers many different approaches, from higher−order programming, syntactic extensibility (e.g., macros), to higher−order programming combined with syntactic support (e.g., meta−object protocols and generics), to full−fledged tinkering with the kernel language (introspection and reflection). Syntactic extensibility and kernel language tinkering in particular are orthogonal to this chart. Some languages, such as Scheme, are flexible enough to implement many paradigms in almost native fashion. This flexibility is not shown in the chart.

Language Design of the Future

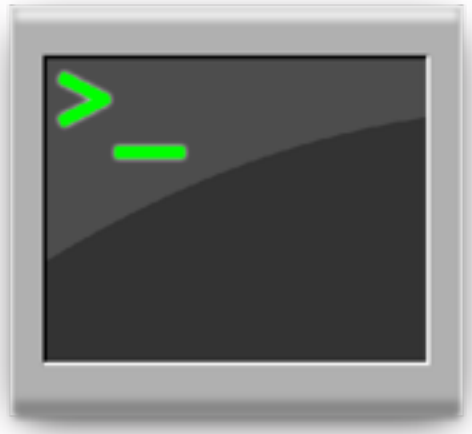
Quorum

Myna

BOOTSTRAP

Equity • Scale • Rigor

Takeaways



- Your **language** will outlive your **project**
  - match promises with consequences
- Do not be too **smart**
  - embrace simplicity
- All **legacy** has its **reasons**
  - (used to be) perfectly valid
- **Explore** and **clean** up
- Questions?