Software Construction

2014-2015

Tijs van der Storm
(storm@cwi.nl / @tvdstorm / @SoftwCons)
Actually: @JurgenVinju this time…

UNIVERSITEIT VAN AMSTERDAM

CWI

Introduction

Jurgen Vinju
(speaker)

Tijs van der Storm
(lectures + labs)

Vadim
(labs)

What this course is about

- You all know programming, right?
- But what is good code?
- How to *reason* about good code?
- What is *beautiful* code?
- Think about it.

This course is *not* about

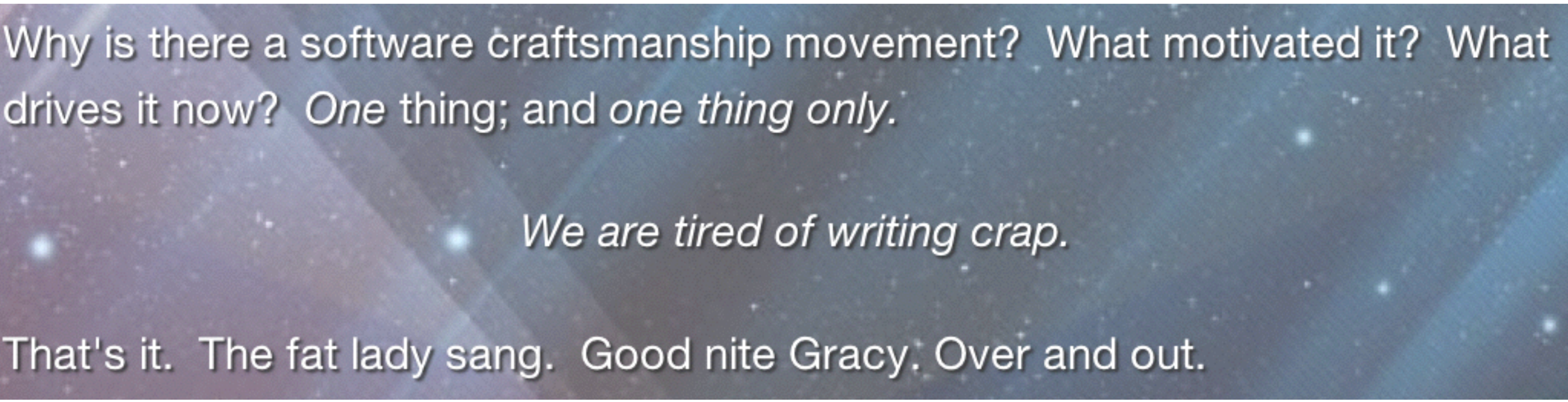
- Data structures
- Algorithms
- Programming language X
- Paradigm X (though: OO)
- GUI programming
- Web applications
- Concurrency
- Performance
- Graphics programming
- Mathematics
- Computational complexity
- ...

Mastering Code

- **Skill**
  - design is a verb: can you do it?
  - be a critic: can you judge quality?
- **Knowledge**
  - algorithms, programming language
  - design patterns, domain knowledge
- **Attitude**
  - wanting to be a master of constructior

Use the force, read the *source*

Uncle Bob*

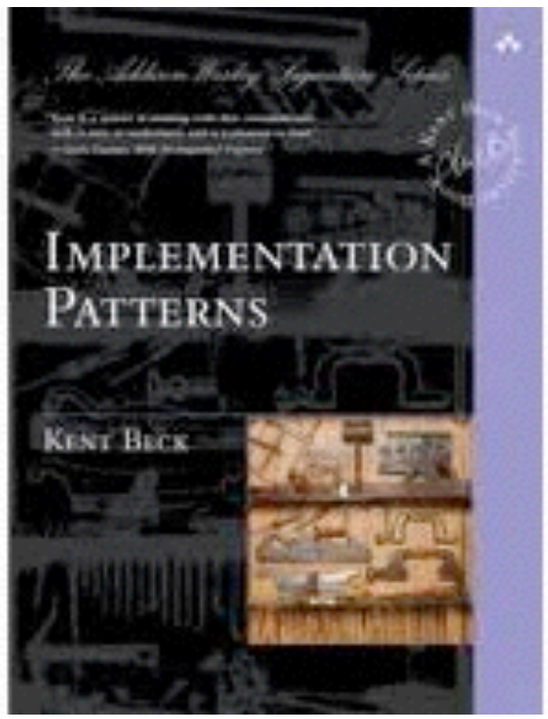


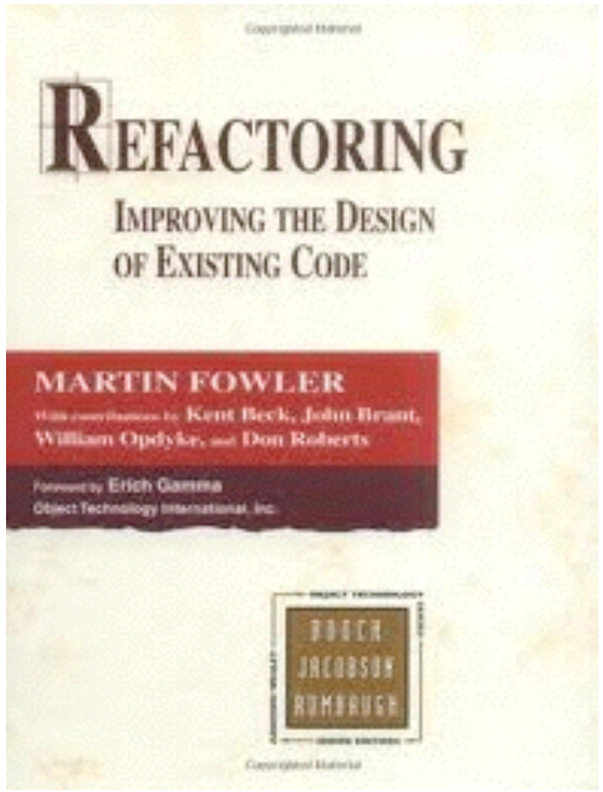
This course is *not* about the software craftmanship movement...

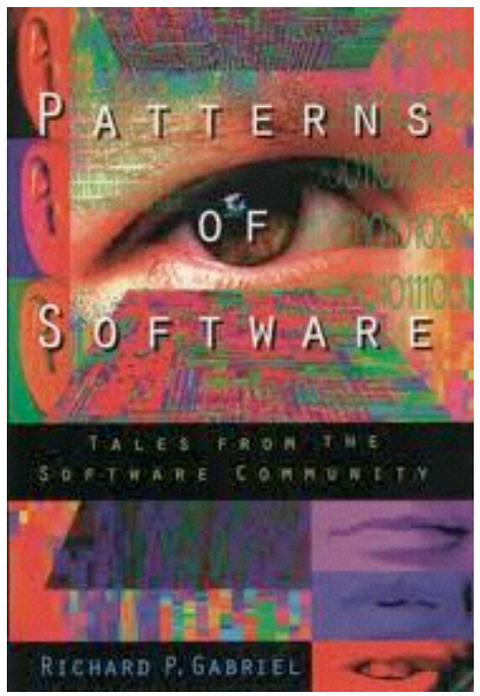
This course *is* about *not* writing crap.

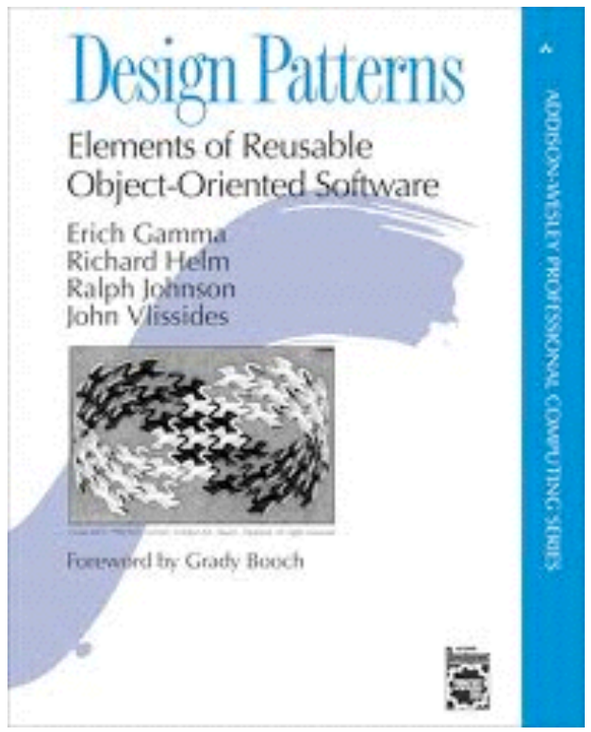
*Robert Martin, http://cleancoder.posterous.com/software-craftsmanship-things-wars-commandmen

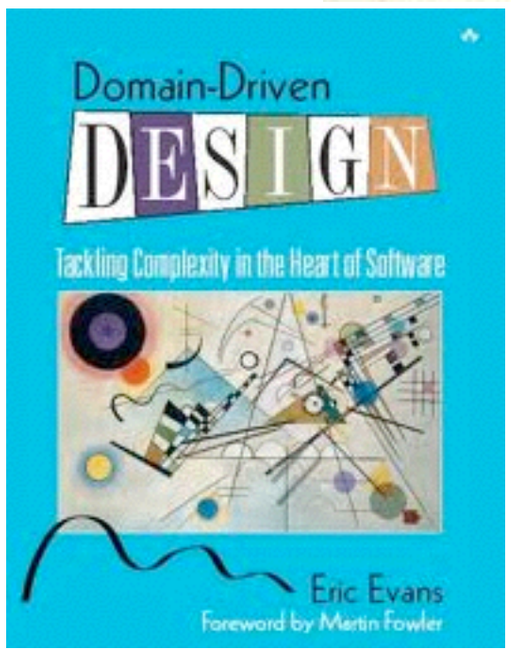
Representative books

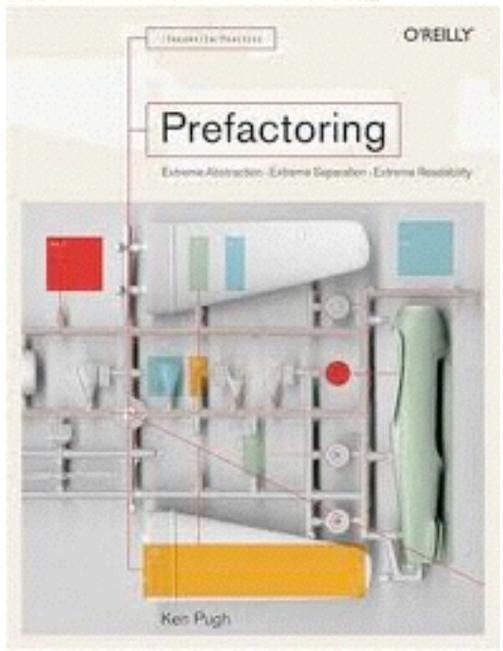


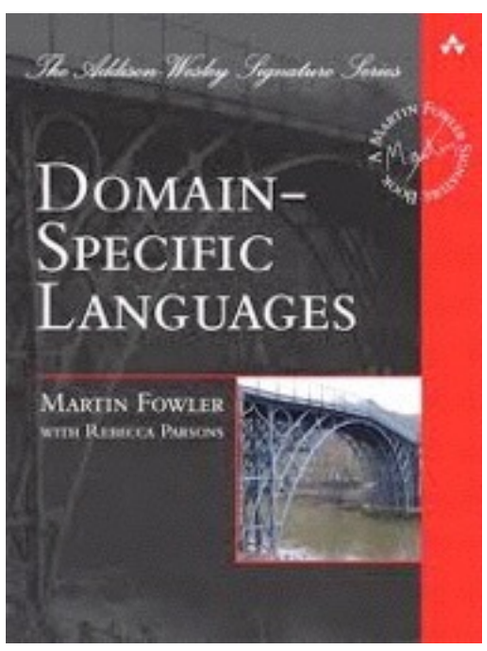










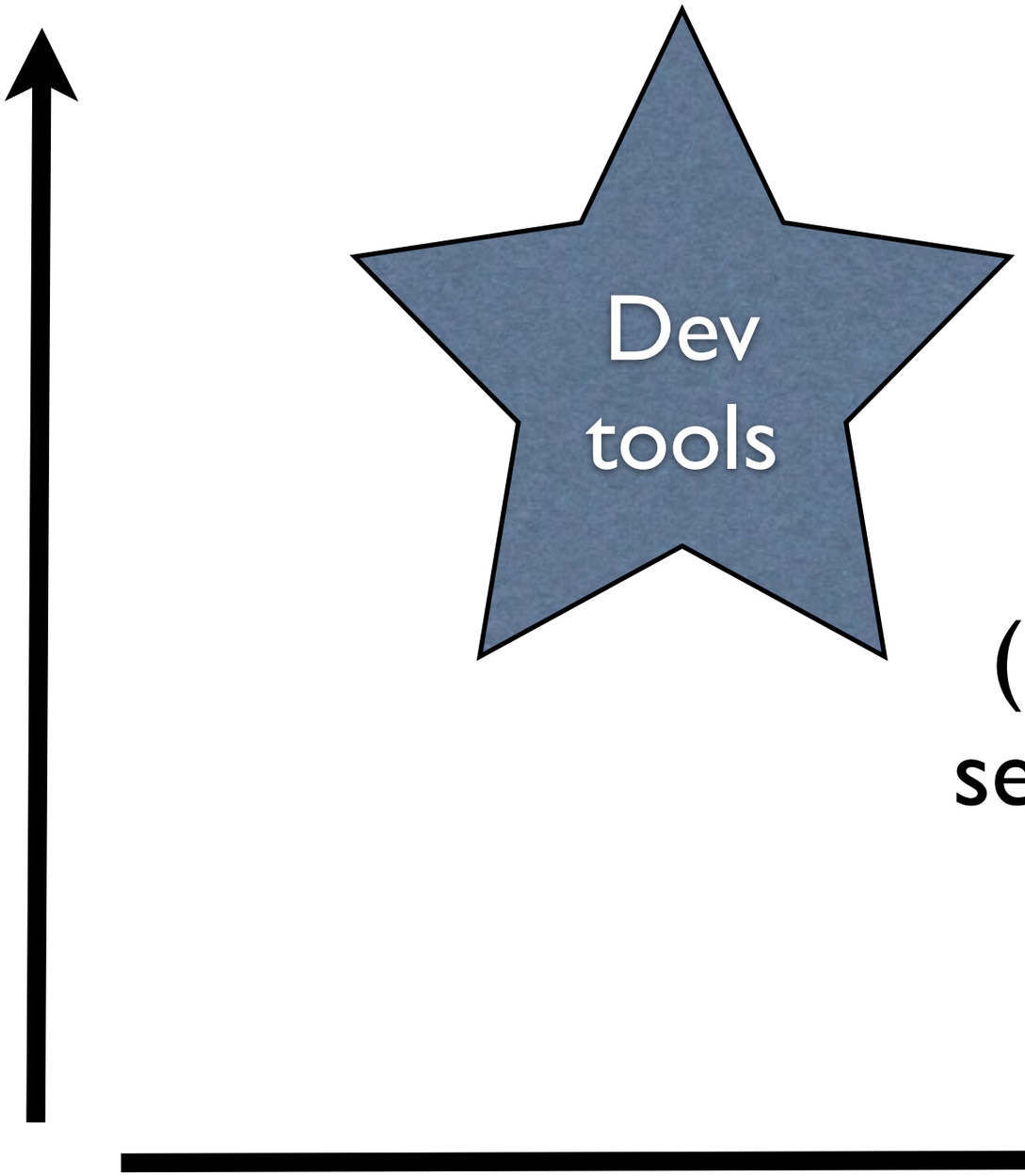


Learning goals

- Create good low level designs
- Produce clean, readable code
- Reflect upon techniques, patterns, guidelines etc.
- Assess the quality of code
- Apply state of the art software construction tools

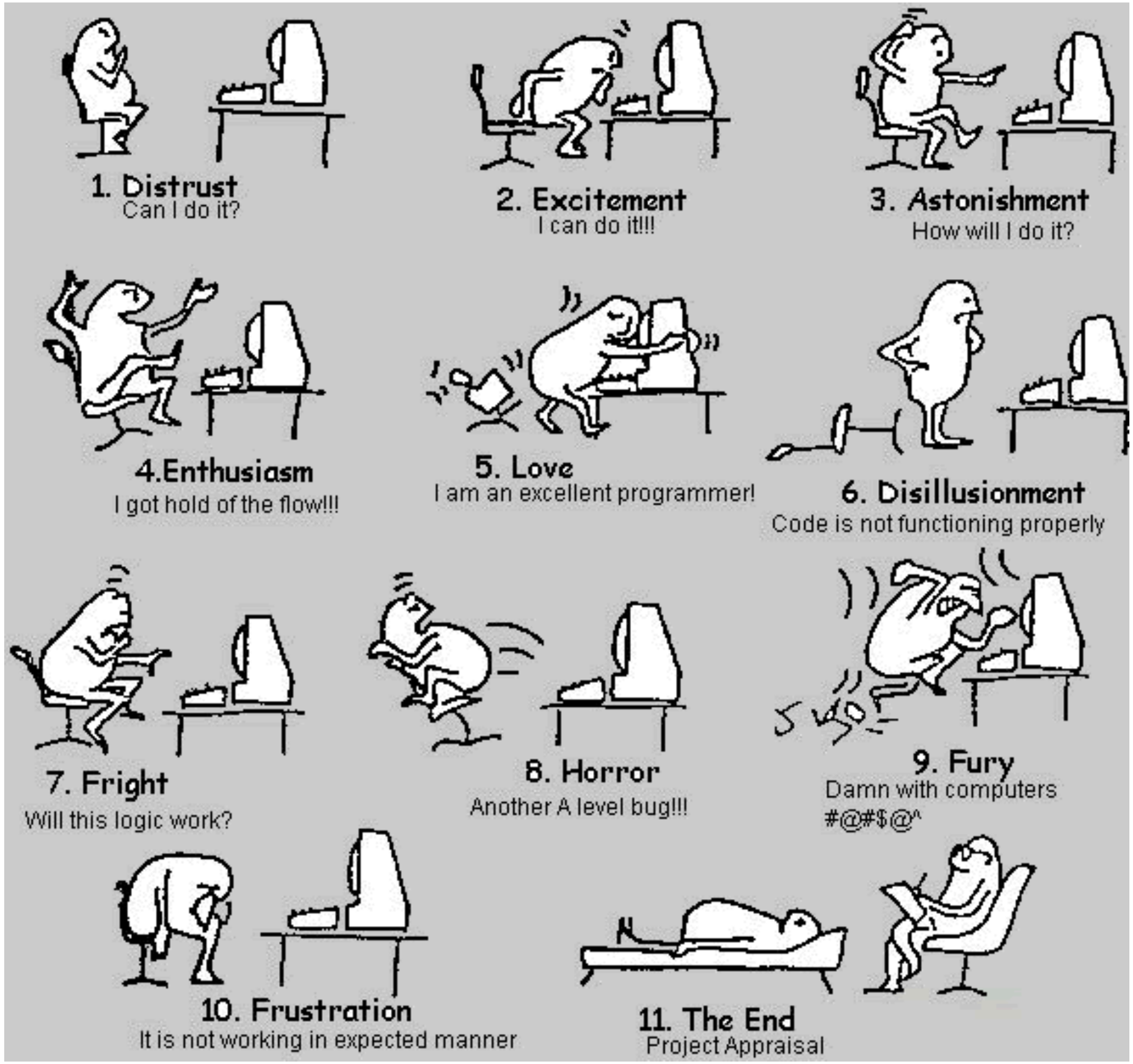
Program something hard

(new techniques, concepts, tools)

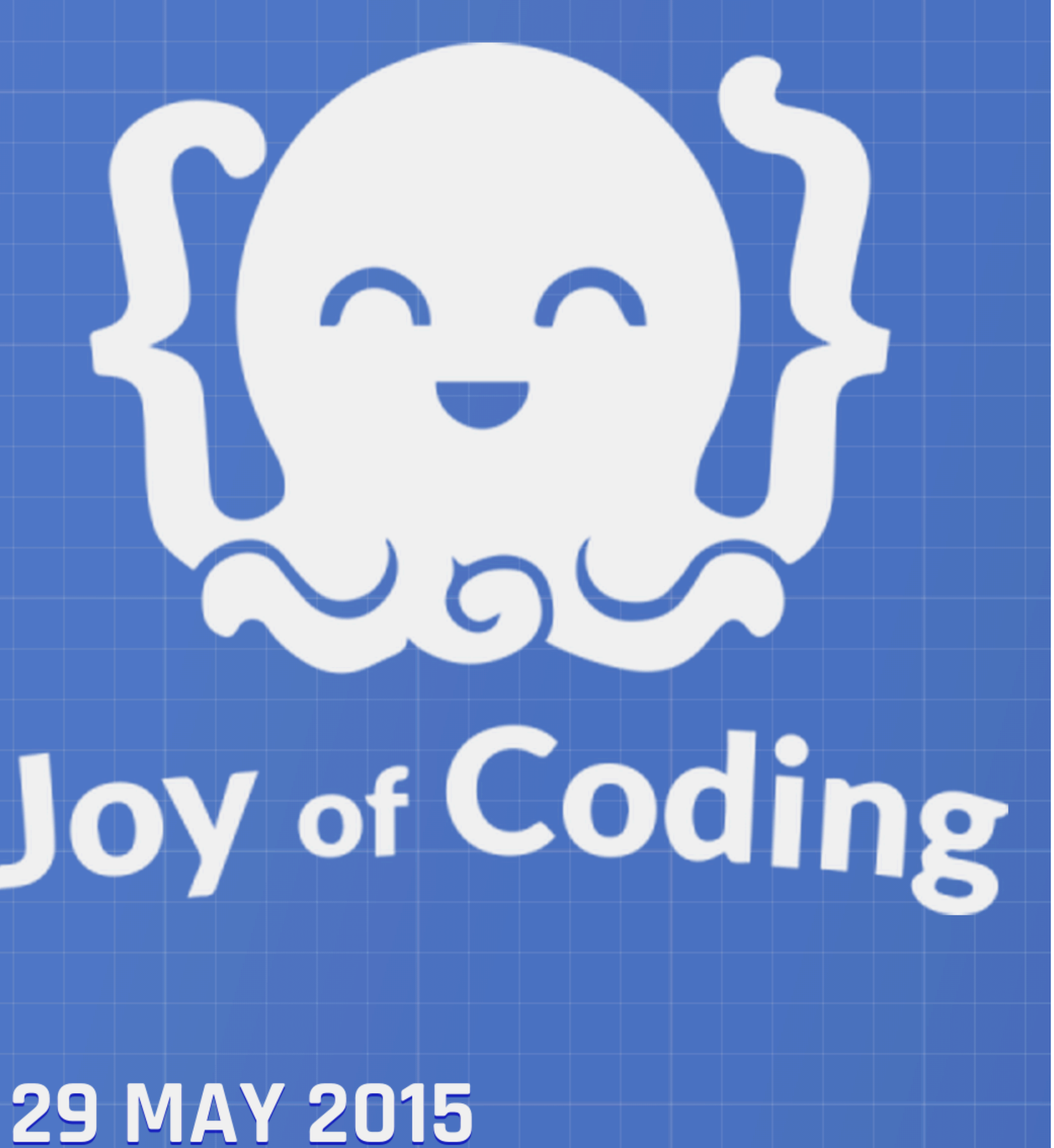


(refactoring, smells, design, separation of concerns, etc.)

Relentless focus on quality



http://candraadi.wordpress.com/2012/10/17/programmer/



joyofcoding.org

This course

- Quality comes first
- Be your own worst critic
- Refactor mercilessly
- Aim to become code literati
- Better to read code, than to write code
- If it works it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's not good enough

If it works, it's not good enough

Working code is necessary, but not sufficient

Why

- Because… software evolution
- Because… software architecture
- Because… software process

- All of that is to try and mitigate problems introduced at construction time.
- But important details are lost in those abstractions

Why

- The restaurant owner designs a concept for a restaurant, but what if the chef makes bad food…
- Formula 1 cars all share the same architecture, but what if the mechanical engineer builds and tunes the machine badly?
- You can buy all the most expensive ski gear, and still end up in the hospital
- *To achieve real quality a master can not not think about the details.*

Overview

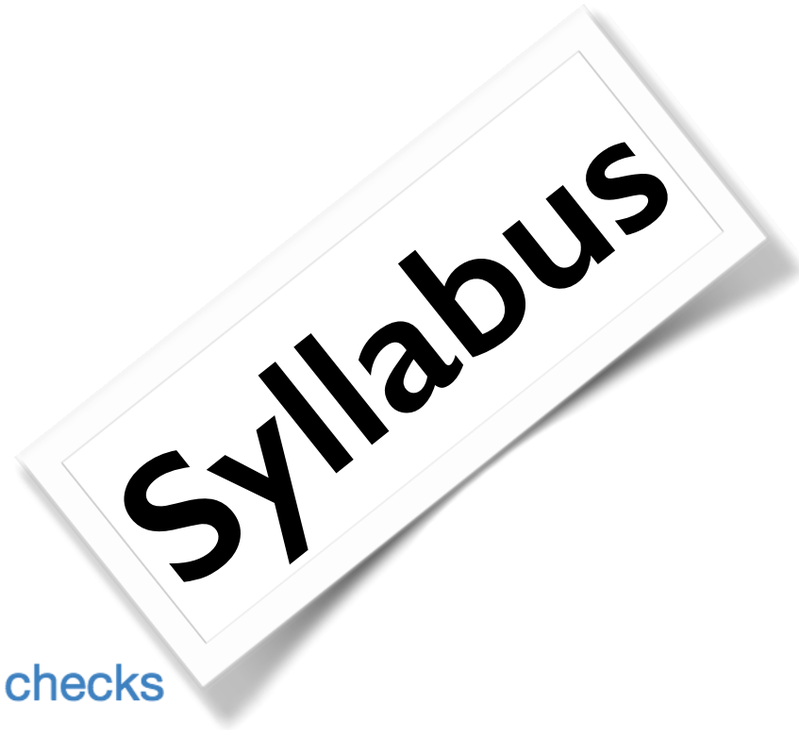
- Lectures
- Theory: papers + book
- Exam: lectures + papers + book
- Lab assignment: implement a little language
- Concluding

Lectures

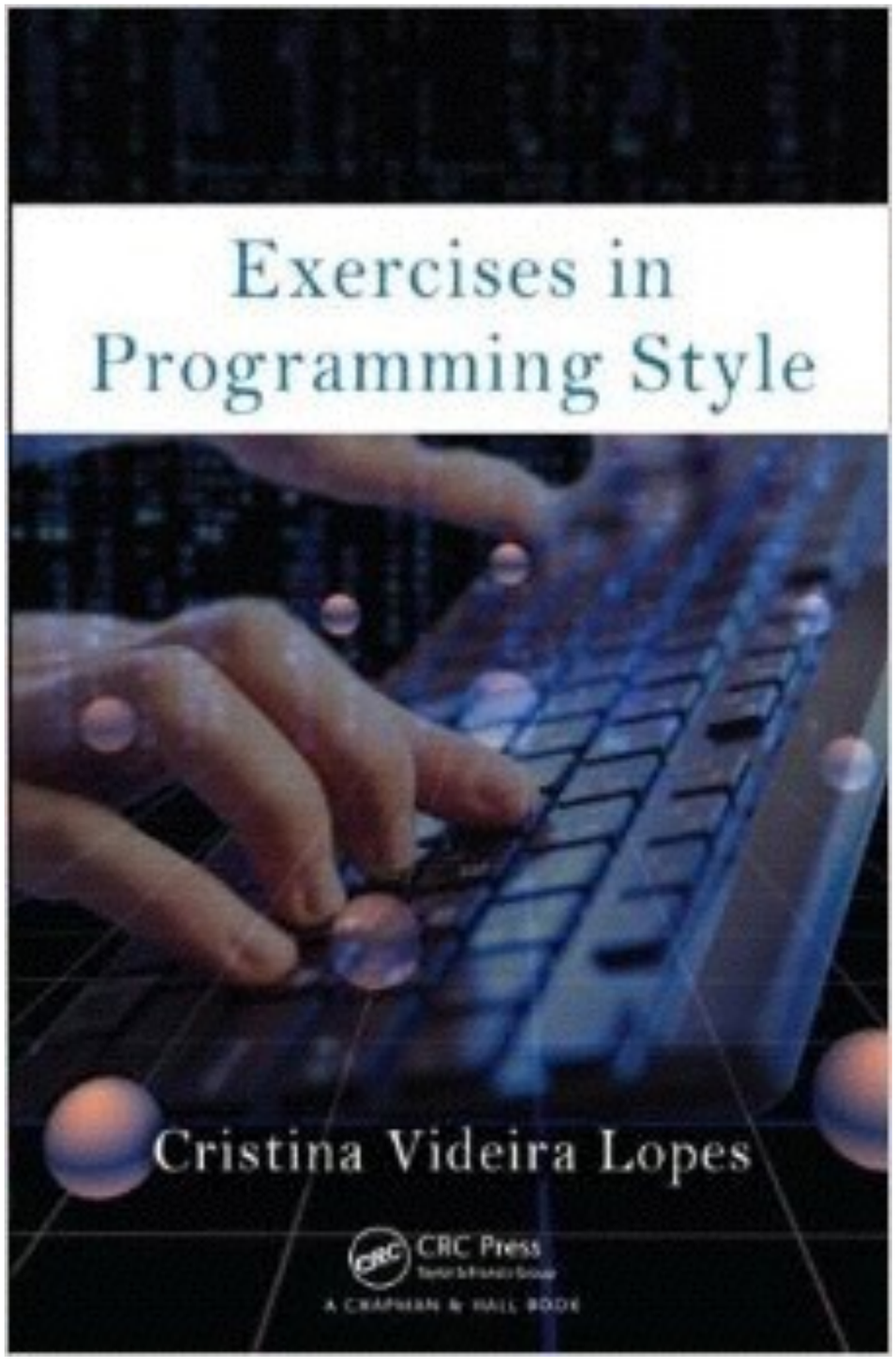
Topics of the lectures

- Syntax analysis: grammars, parsers
- Programming styles, design principles etc.
- Code quality: tangling, scattering, duplication, smells, refactoring, layout
- Modularity: information hiding, separation of concerns, encapsulation, dependency
- ....

- Karl J. Lieberherr, Ian M. Holland, *Assuring Good Style for Object-Oriented Programs*, 1989, LieberherrHolland8
- D. L. Parnas, *On the criteria to be used in decomposing systems into modules* , 1972, Parnas72
- W. Wulf and Mary Shaw, *Global variable considered harmful*, 1973, WulfShaw84.
- John Hughes, *Why functional programming matters*, 1990 Hughes89
- Robert C. Martin, *Design principles and design patterns*, Martin00.
- Erich Gamma, Richard Helm, Ralpha Johnson, John Vlissides, *Design Patterns: Abstraction and Reuse of Objec Oriented Design*, ECOOP 93 GammaEtAl93
- Kent Beck and Martin Fowler, *Bad Smells in Code* (Chapter 3, *Refactoring*)
- Kent Beck, A theory of programming, (Chapter 3, *Implementation Patterns*)
- Kent Beck, *Aim, fire*, IEEE Software, Beck01
- Jeff Bay, *Object Calisthenics*, Bay.
- Ward Cunningham, *The CHECKS Pattern Language of Information Integrity*, checks
- Kernighan, Plauger, *Programming Style: Examples and Counterexamples*, 1974 kernighanPlauger
- Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin, *Aspect-Oriented Programming*, KiczalesEtAl97
- James Noble, *Arguments and Results*, Noble97



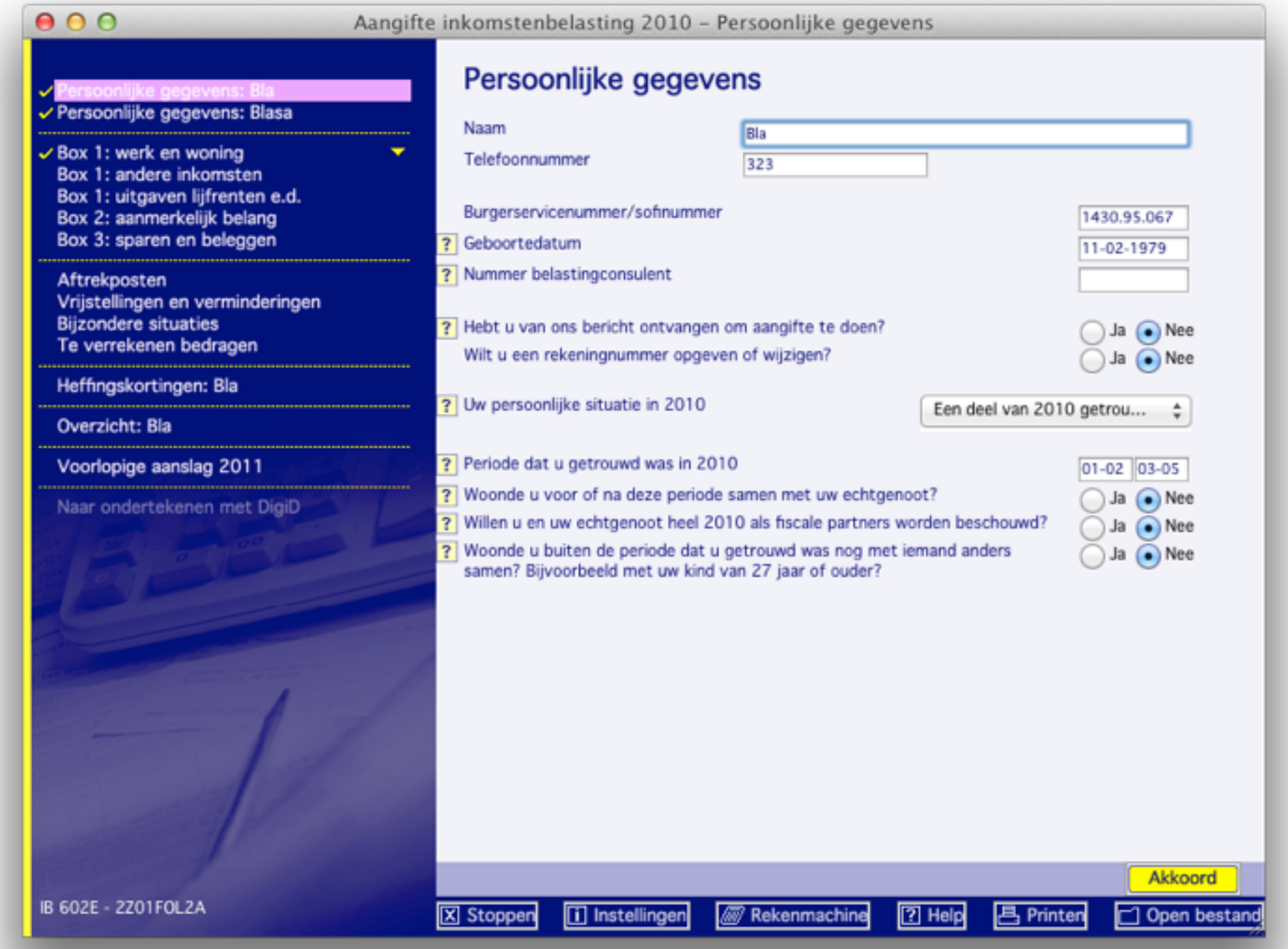
Book: EiPS


Exercises in
Programming Style
Cristina Videira Lopes
CRC Press

Final exam

- Open book exam
- Must have grade > 5.5
- Based on
  - Exercises in Programming Style
  - Lecture material
  - Syllabus

Lab assignment



Form **1040** Department of the Treasury—Internal Revenue Service (99)

U.S. Individual Income Tax Return

2012 OMB No. 1545-0074 IRS Use Only—Do not write or staple in this space.

For the year Jan. 1–Dec. 31, 2012, or other tax year beginning , 2012, ending , 20 See separate instructions.

Your first name and initial	Last name	**Your social security number**
If a joint return, spouse's first name and initial	Last name	**Spouse's social security number**
Home address (number and street). If you have a P.O. box, see instructions.	Apt. no.	▲ Make sure the SSN(s) above and on line 6c are correct.
City, town or post office, state, and ZIP code. If you have a foreign address, also complete spaces below (see instructions).		**Presidential Election Campaign** Check here if you, or your spouse if filing jointly, want $3 to go to this fund. Checking a box below will not change your tax or refund. ☐ **You** ☐ **Spouse**
Foreign country name	Foreign province/state/county / Foreign postal code	

Filing Status

Check only one box.

1 ☐ Single
2 ☐ Married filing jointly (even if only one had income)
3 ☐ Married filing separately. Enter spouse's SSN above and full name here. ▶
4 ☐ Head of household (with qualifying person). (See instructions.) If the qualifying person is a child but not your dependent, enter this child's name here. ▶
5 ☐ Qualifying widow(er) with dependent child

Exemptions

6a ☐ **Yourself.** If someone can claim you as a dependent, **do not** check box 6a
b ☐ **Spouse**

c **Dependents:**

(1) First name Last name	(2) Dependent's social security number	(3) Dependent's relationship to you	(4) ✓ if child under age 17 qualifying for child tax credit (see instructions)
			☐
			☐
			☐
			☐

If more than four dependents, see instructions and check here ▶ ☐

d Total number of exemptions claimed

Boxes checked on 6a and 6b
No. of children on 6c who:
• lived with you
• did not live with you due to divorce or separation (see instructions)
Dependents on 6c not entered above
Add numbers on lines above ▶

Income

Attach Form(s) W-2 here. Also attach Forms W-2G and 1099-R if tax was withheld.

If you did not get a W-2, see instructions.

Enclose, but do not attach, any payment. Also, please use **Form 1040-V.**

7	Wages, salaries, tips, etc. Attach Form(s) W-2		7
8a	**Taxable** interest. Attach Schedule B if required		8a
b	**Tax-exempt** interest. **Do not** include on line 8a	8b	
9a	Ordinary dividends. Attach Schedule B if required		9a
b	Qualified dividends	9b	
10	Taxable refunds, credits, or offsets of state and local income taxes		10
11	Alimony received		11
12	Business income or (loss). Attach Schedule C or C-EZ		12
13	Capital gain or (loss). Attach Schedule D if required. If not required, check here ▶ ☐		13
14	Other gains or (losses). Attach Form 4797		14
15a	IRA distributions 15a	b Taxable amount	15b
16a	Pensions and annuities 16a	b Taxable amount	16b
17	Rental real estate, royalties, partnerships, S corporations, trusts, etc. Attach Schedule E		17
18	Farm income or (loss). Attach Schedule F		18
19	Unemployment compensation		19
20a	Social security benefits 20a	b Taxable amount	20b
21	Other income. List type and amount		21
22	Combine the amounts in the far right column for lines 7 through 21. This is your **total income** ▶		22

Adjusted Gross Income

23	Reserved	23	
24	Certain business expenses of reservists, performing artists, and fee-basis government officials. Attach Form 2106 or 2106-EZ	24	
25	Health savings account deduction. Attach Form 8889	25	
26	Moving expenses. Attach Form 3903	26	
27	Deductible part of self-employment tax. Attach Schedule SE	27	
28	Self-employed SEP, SIMPLE, and qualified plans	28	
29	Self-employed health insurance deduction	29	
30	Penalty on early withdrawal of savings	30	
31a	Alimony paid b Recipient's SSN ▶	31a	
32	IRA deduction	32	
33	Student loan interest deduction	33	
34	Reserved	34	
35	Domestic production activities deduction. Attach Form 8903	35	
36	Add lines 23 through 35		36
37	Subtract line 36 from line 22. This is your **adjusted gross income** ▶		37

Part 1: Questionnaire Language (QL)

```
form taxOfficeExample {
  "Did you sell a house in 2010?"
    hasSoldHouse: boolean
  "Did you buy a house in 2010?"
    hasBoughtHouse: boolean
  "Did you enter a loan?"
    hasMaintLoan: boolean

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

Describe the logic of interactive questionnaires

- Did you sell a house in 2010?
- [ ]
- Did you buy a house in 2010?
- [ ]
- Did you enter a loan?
- [x]

- Did you sell a house in 2010?
- [x]
- Did you buy a house in 2010?
- [ ]
- Did you enter a loan?
- [x]
- What was the selling price?

  24234
- Private debts for the sold house:

  34343
- Value residue:

  -10109

```
stylesheet taxOfficeExample
  page Housing {
    section "Buying"
      question hasBoughtHouse
        widget checkbox
    section "Loaning"
      question hasMaintLoan
  }

  page Selling {
    section "Selling" {
      question hasSoldHouse
        widget radio("Yes", "No")
      section "You sold a house" {
      question sellingPrice
        widget spinbox
      question privateDebt
        widget spinbox
      question valueResidue
      default money {
        width: 400
        font: "Arial"
        fontsize: 14
        color: #999999
        widget spinbox
      }
    }
  }
  default boolean widget radio("Yes", "No")
}
```

QLS

Language for styling questionnaires

Buying

Did you buy a house in 2010?

Loaning

Did you enter a loan?

Previous Next

Selling

Did you sell a house in 2010?

Yes

No

You sold a house

What was the selling price?

232323

Private debts for the sold house:

12323

Value residue:

220000

Previous Next

Part 1: QL

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Type checker/Wellformedness checker
- Expression evaluator
- Renderer as GUI
  (interpreter! Not a compiler)

Part 2: QLS

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Wellformedness checker WRT QL program
- Renderer as stylized GUI
- Challenge: modular implementation
- QL **should** work standalone (w/o QLS)

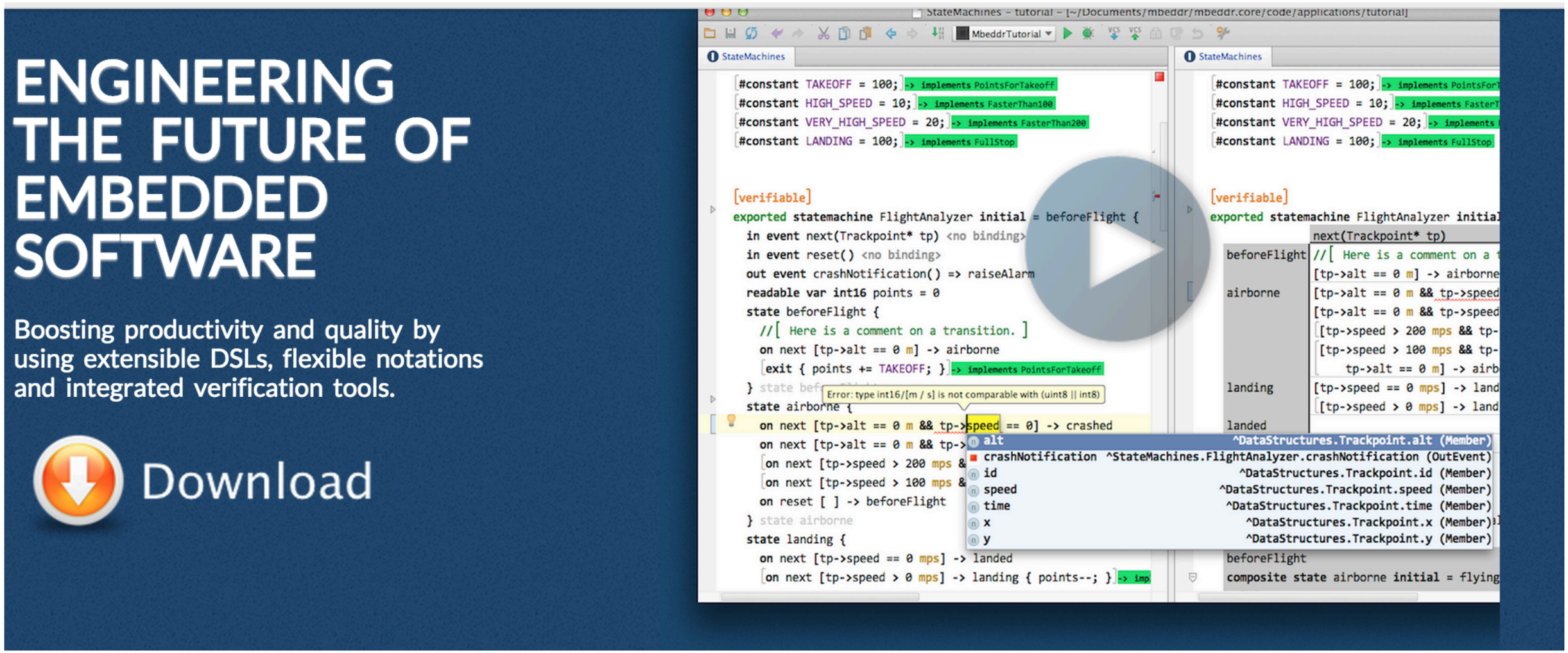
No server-side web apps!

- server/client distinction is a distraction
- essentially code generation all over the place

Programming language

- Java, C#, Javascript, Typescript, Haskell, Scala, Clojure, Erlang, Smalltalk/Pharo, Ruby, Python, Go, Dart, Swift, Objective-C, F#, Rust, …
- Java: you may want to use one of the provided parsing skeletons for expressions in QL
  - *Rats!*, Jacc, ANTLR

Honor's track: mbeddr in Rascal



http://mbeddr.com/

mbeddr

- C + Language extensions
  - unit tests, modules, state machines, requirements traces, variability, …
  - Built using Jetbrains Meta Programming System (MPS)
  - You will implement number of extensions in Rascal
  - Modularly!!!!

Github

- Assignment to be completed *in teams of 2*
  - (except honor's track)
  - http://github.com/software-engineering-amsterdam/many-ql/
  - Use of this repository is **required**!
  - Commit often!

“Hour of code”

- During lab sessions (Mon 14:00/Tue 9:15)
- Convene in single room
- Teams (2 per session) present their code.
- No slides. Code.
- Constructive feedback and criticism.
- Let’s help each other.

Grading of lab assignments

- Functionality
- Tes
- Simplicity
- Modularity
- Layout and style
- Separation of concerns

Self pre-assessment

- Before grading moments:
- You fill out an online questionnaire
- This will help us
  - navigate the code
  - ask the right questions

Some advice up-front

- Naming, layout, indentation
- Encapsulation, modularity, separation of concerns, reuse
- Don’t repeat yourself (DRY)
- Library and tool selection and use
- Unit testing

More advice

- Use asserts sensibly
- No global, static, non-final variables
- You ain't going to need it (YAGNI)
- Avoid premature optimization
- Use comments for rationale
- *Compiling **and** working code*

Grading (ctd.)

- First part: your grade is *indicative*
  - incentive to improve your code
- Second part: we review all code
  - this will be your *final* grade for the lab
- Grading is on-site: *you* show your code
- Grade is less important than personal improvement is

Passing this course

- Be present at all lectures
- Be present during lab sessions
- Pass the the exam with grade > 5.5
- Pass lab assignment with grade > 5.5
- Final grade: average of lab and exam
- NOTE: both grades need to be > 5.5

week	7	8	9	10	11	12	13
Mon 9:15-11	lecture	lecture	lecture	lecture	lecture	lecture	Final grading
Mon 11-14	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
Mon 14-15	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Final grading
Mon 15-17	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
Tue 9:15-10:15	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Hour of code	Final grading
Tue 10:30-17:00	Lab	Lab	Lab	Lab	Lab	Lab	Final grading
			First grading	First grading		Final grading	Exam on Thursday

Concluding

- All information is on Github
- Primary contact = storm@cwi.nl
- Please follow @SoftwCons

What's next

- For the coming days
  - make up your mind on language
    - start checking out parser generators
    - start coding!
- NB @tvdstorm will be back next week. Vadim is here to help!