UNIVERSITEIT VAN AMSTERDAM

Software Construction

2018

Dr. Vadim Zaytsev aka @grammarware

raincode LABS compiler experts

People involved

Vadim Zaytsev

Ana-Maria Oprescu

raincode LABS
compiler experts

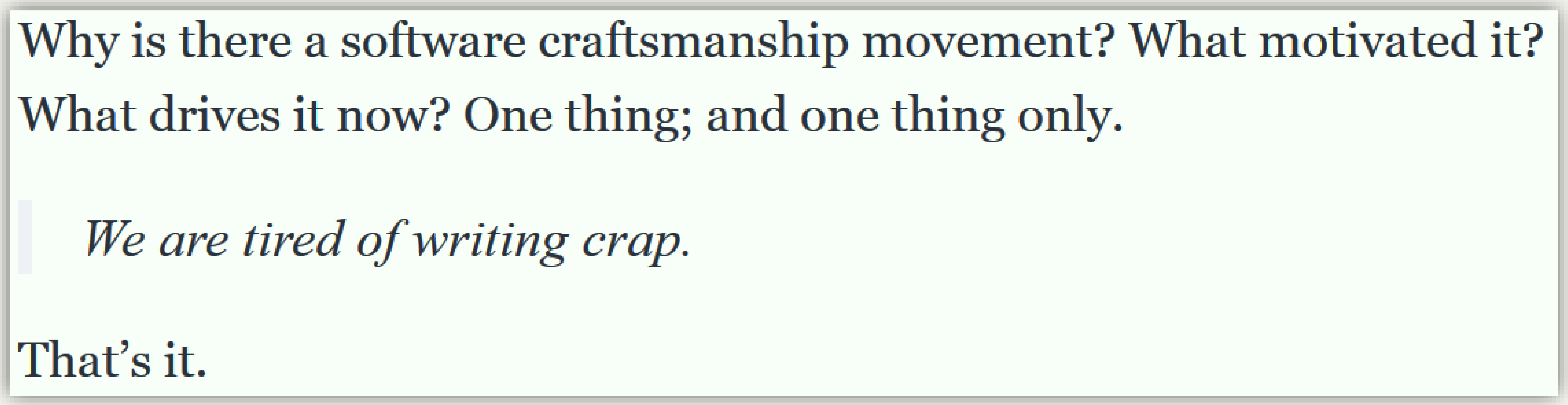
What this course is about

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

What this course is *not* about

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

Uncle Bob*


Why is there a software craftsmanship movement? What motivated it? What drives it now? One thing; and one thing only.

> *We are tired of writing crap.*

That’s it.

This course is *not* about the software craftsmanship movement…

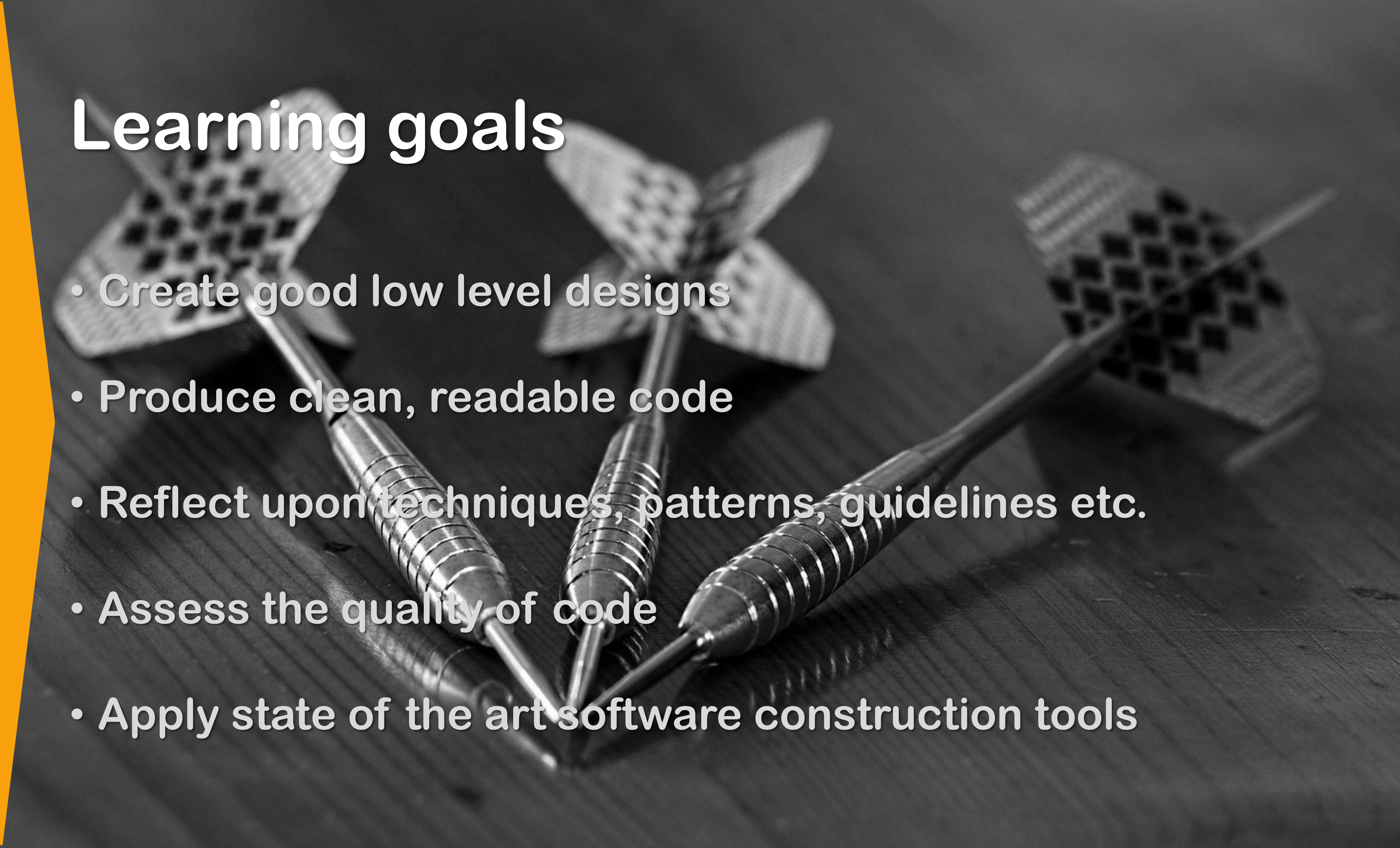
This course is about not writing crap.

*Robert Martin, http://blog.cleancoder.com/uncle-bob/2011/01/17/software-craftsmanship-is-about.html

raincode LABS
compiler experts

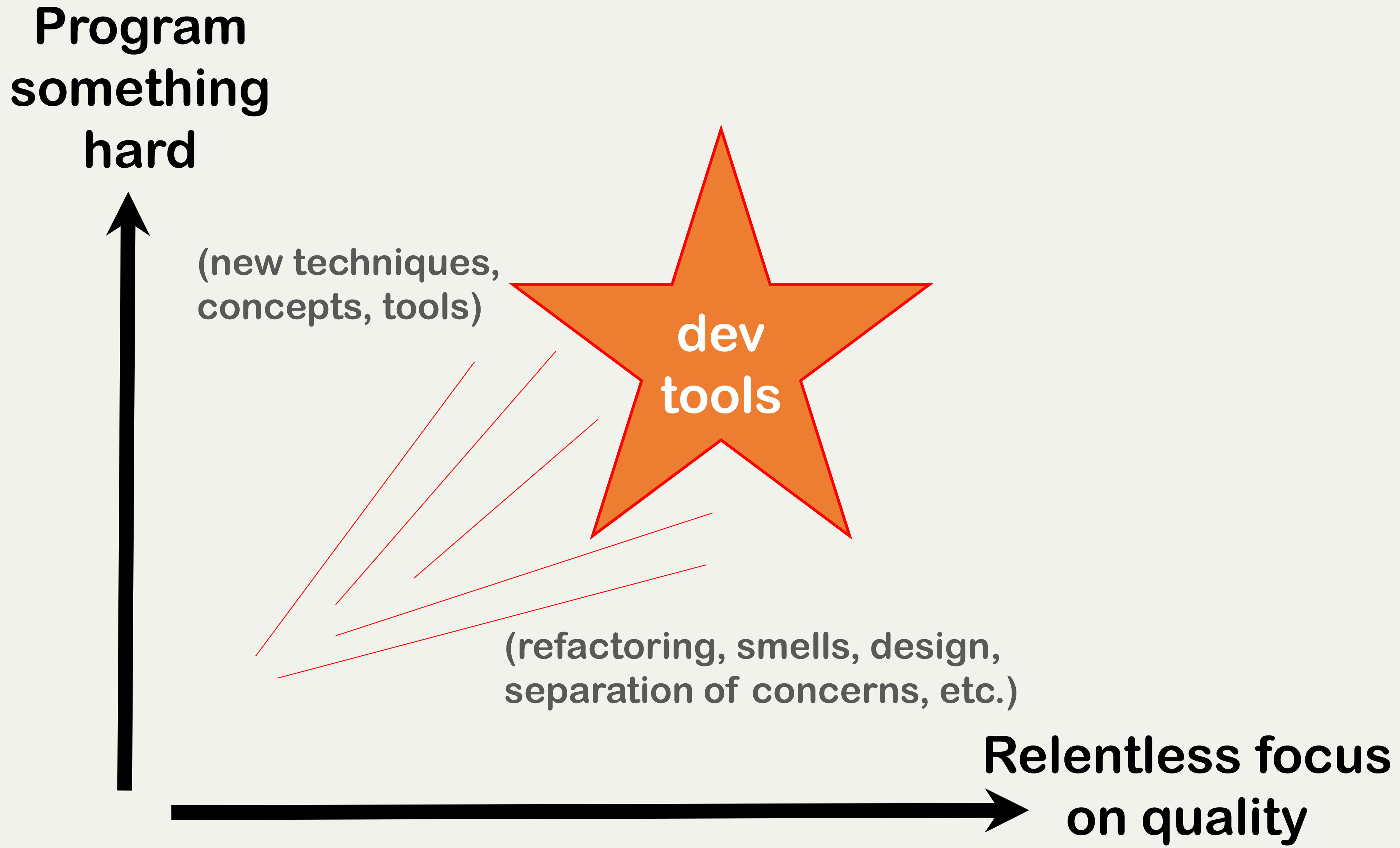
Representative books

Exercises in Programming Style
Cristina Videira Lopes
CRC Press
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
Addison-Wesley Professional Computing Series
The Addison-Wesley Signature Series
Patterns of Enterprise Application Architecture
Martin Fowler
With Contributions by David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford
A Martin Fowler Signature Book
Patterns of Software
Tales from the Software Community
Richard P. Gabriel
The Addison-Wesley Signature Series
Implementation Patterns
Kent Beck
A Kent Beck Signature Book
Microsoft
Code Complete
2
Second Edition
A practical handbook of software construction
Steve McConnell
Refactoring
Improving the Design of Existing Code
Martin Fowler
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
O'Reilly
Prefactoring
Ken Pugh
Effective Software Development Series
Scott Meyers, Consulting Editor
Code Reading
The Open Source Perspective
Diomidis Spinellis



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)
dev tools
(refactoring, smells, design, separation of concerns, etc.)
Relentless focus on quality

raincode LABS
compiler experts

Words to live by (during 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

Words to live by (during this course)

- If it works, it’s not good enough

Words to live by (during this course)

If it works, it's not good enough

Words to live by (during this course)

If it works, it’s not good enough

raincode LABS
compiler experts

If it works, it's not good enough

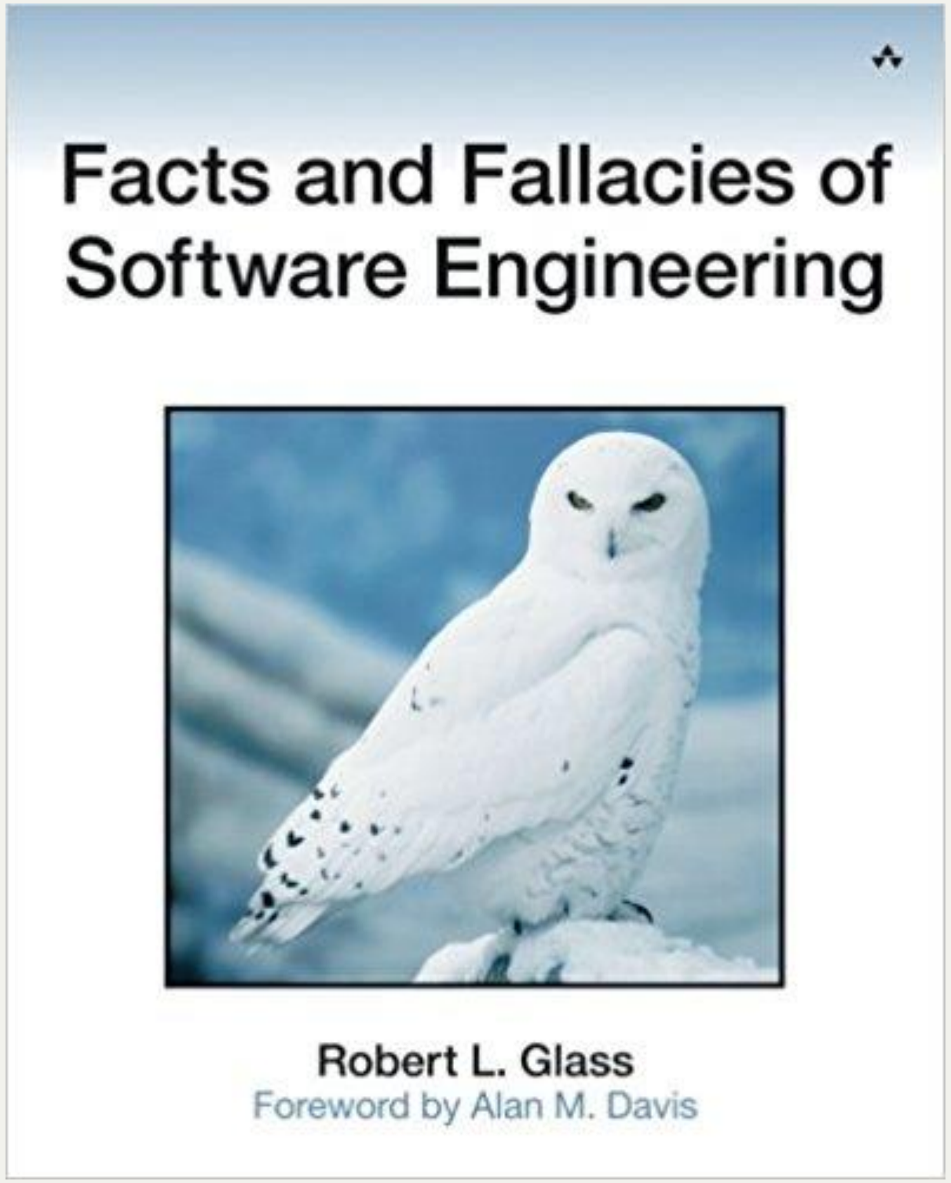
raincode LABS
compiler experts

Suspend your disbelief

Why

Fact 41

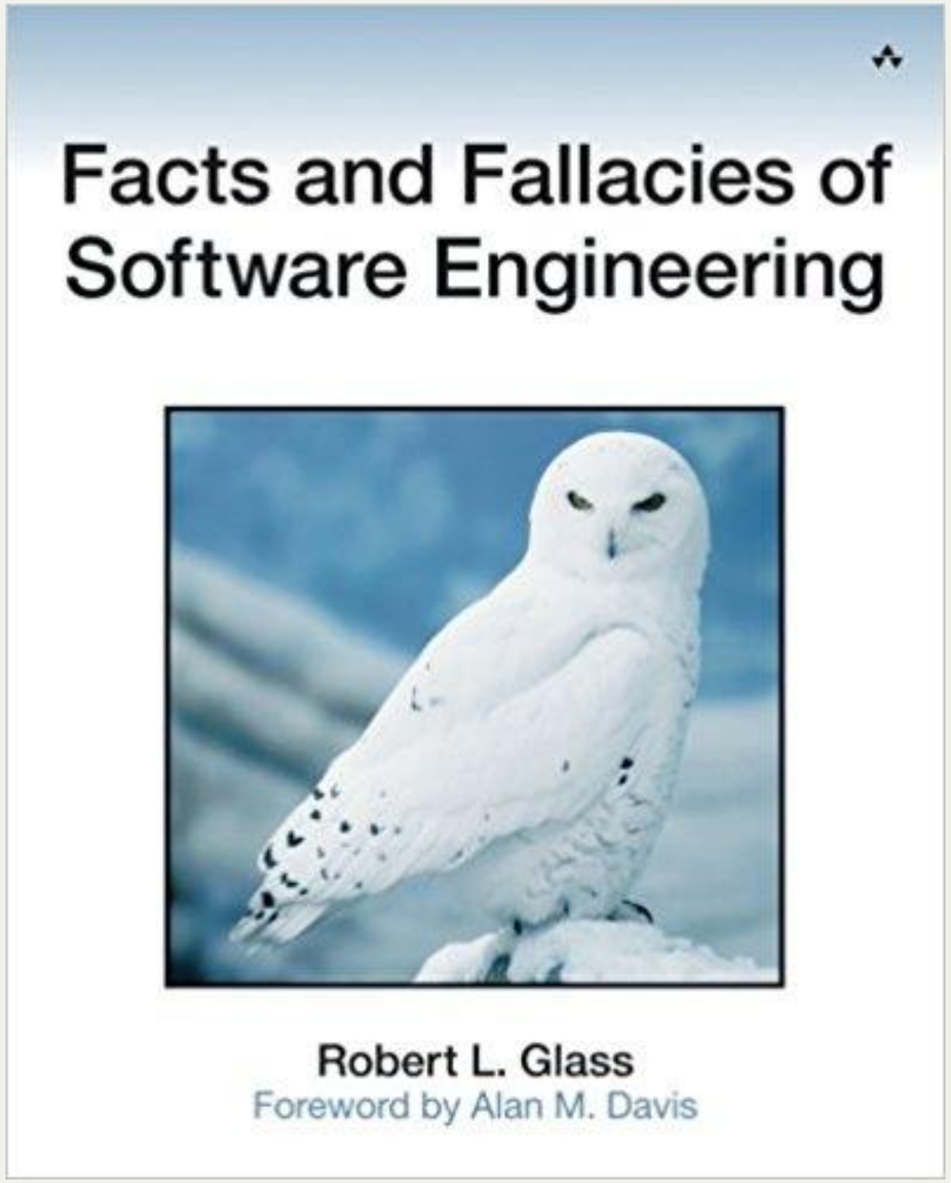
Maintenance typically consumes 40 to 80 percent (average, 60 percent) of software costs. Therefore, it is probably the most important life cycle phase of software.


Facts and Fallacies of Software Engineering
Robert L. Glass
Foreword by Alan M. Davis

Why

Fact 44

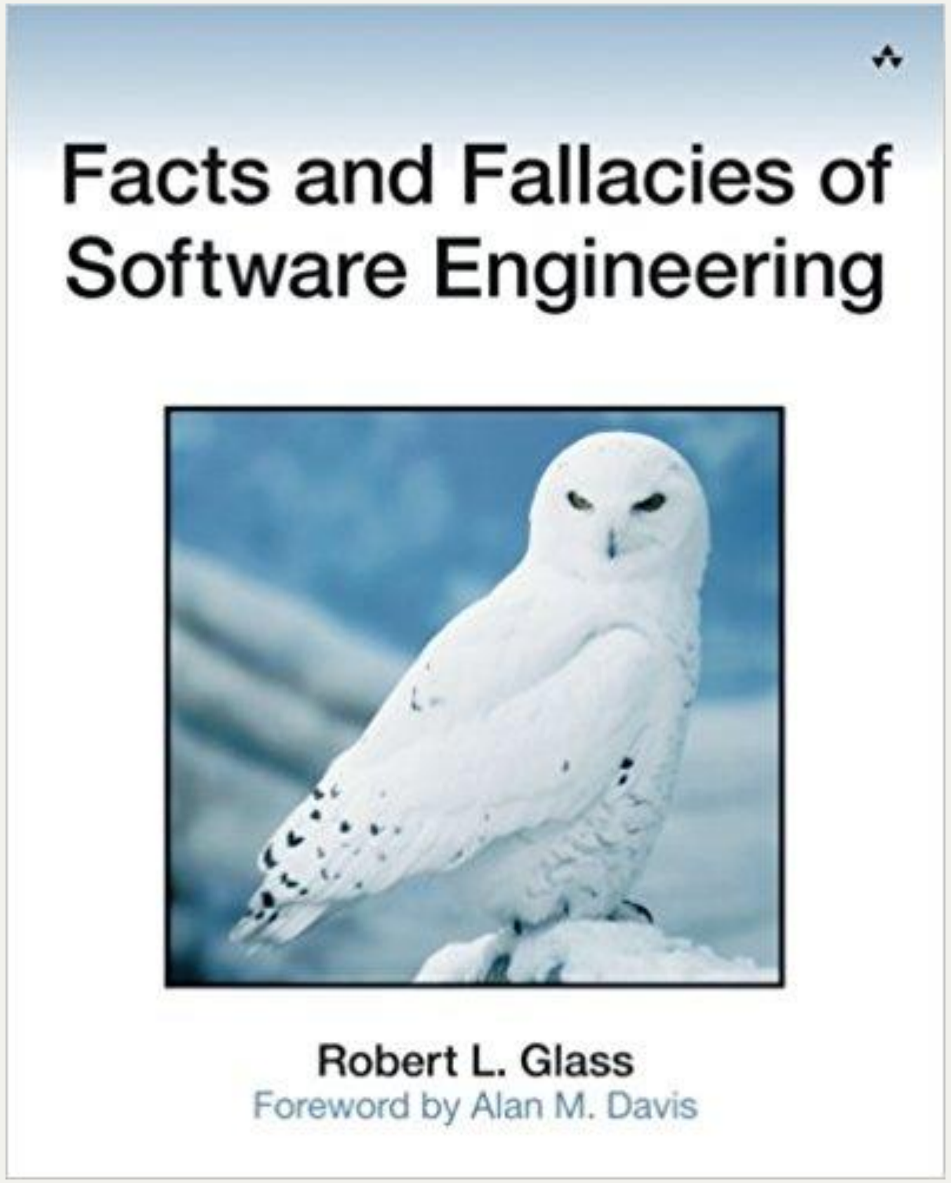
Understanding the existing product: this task consumes roughly 30 percent of the total maintenance time and is the dominant maintenance activity. Thus it is possible to claim that maintenance is a more difficult task than development.


Facts and Fallacies of Software Engineering
Robert L. Glass
Foreword by Alan M. Davis

Why

Fact 21

For every 25 percent increase in problem complexity, there is a 100 percent increase in complexity of the software solution. That's not a condition to try to change (even though reducing complexity is always a desirable thing to do); that's just the way it is.

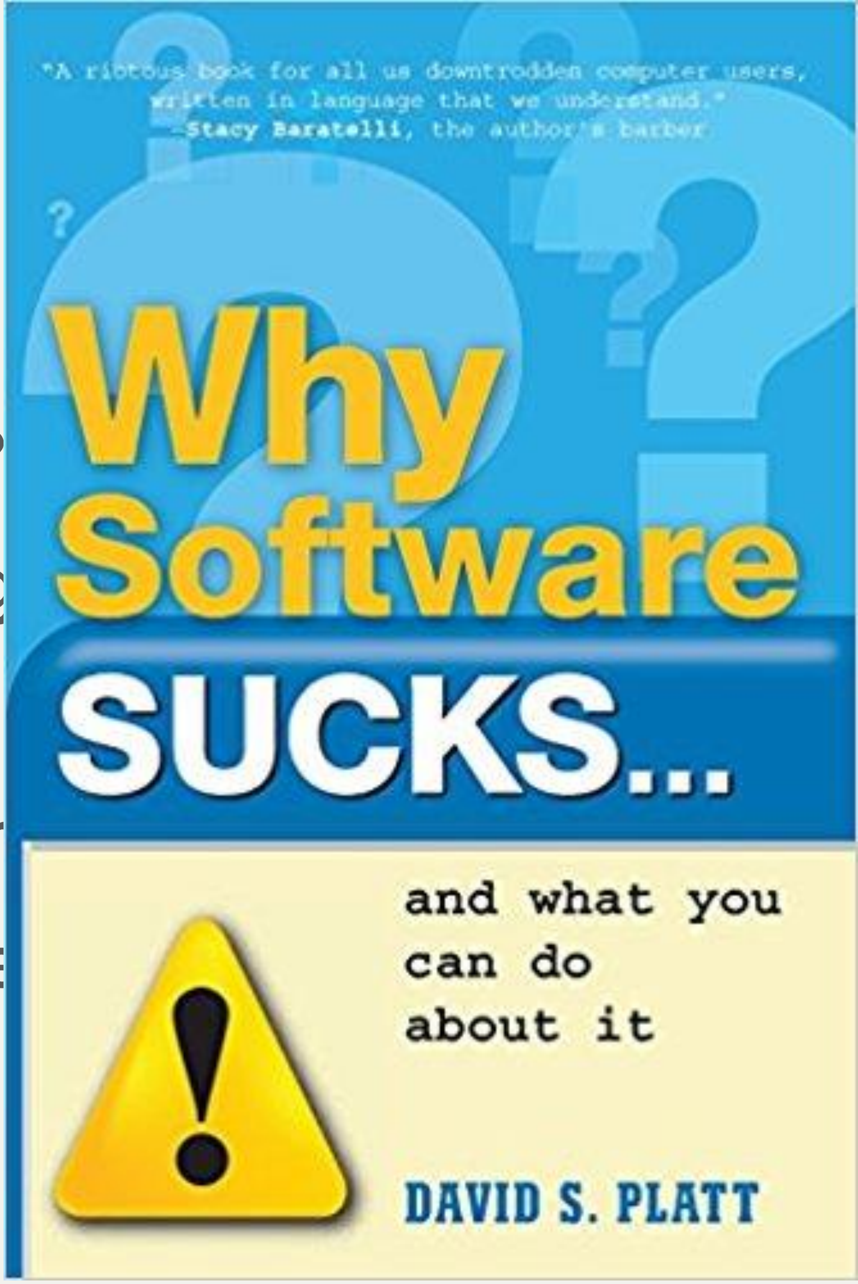

Facts and Fallacies of Software Engineering
Robert L. Glass
Foreword by Alan M. Davis

Related whys

- [SE] Why estimations are difficult and wrong?
  - because our solutions are more complex than problems
- [RE] Why is there a requirements explosion?
  - because explicit is the 25% and implicit is the 100%
- [ST] Why is 100% test coverage insufficient?
  - because complexity leads to errors coverage cannot trap
- [SC] Why are there different solutions to the same problem?
  - because the solution space more is complex than the problem space

Related whys

- Why reuse-in-the-large unsuccessful?
- Why is code review the best bug fixing
- Why are designs seldom optimised?
- Why does maintenance consume so m
- Why advocacy is more common than e
- Why software sucks?
  - because the devil is in the details



"A riotous book for all us downtrodden computer users, written in language that we understand."
-Stacy Baratelli, the author's barber

Why Software SUCKS...

and what you can do about it

DAVID S. PLATT

Because

- Software Evolution
- Software Architecture
- Software Process
- Software Testing

- All of the above try to mitigate problems introduced at construction time

Time for a break

raincode LABS

compiler experts

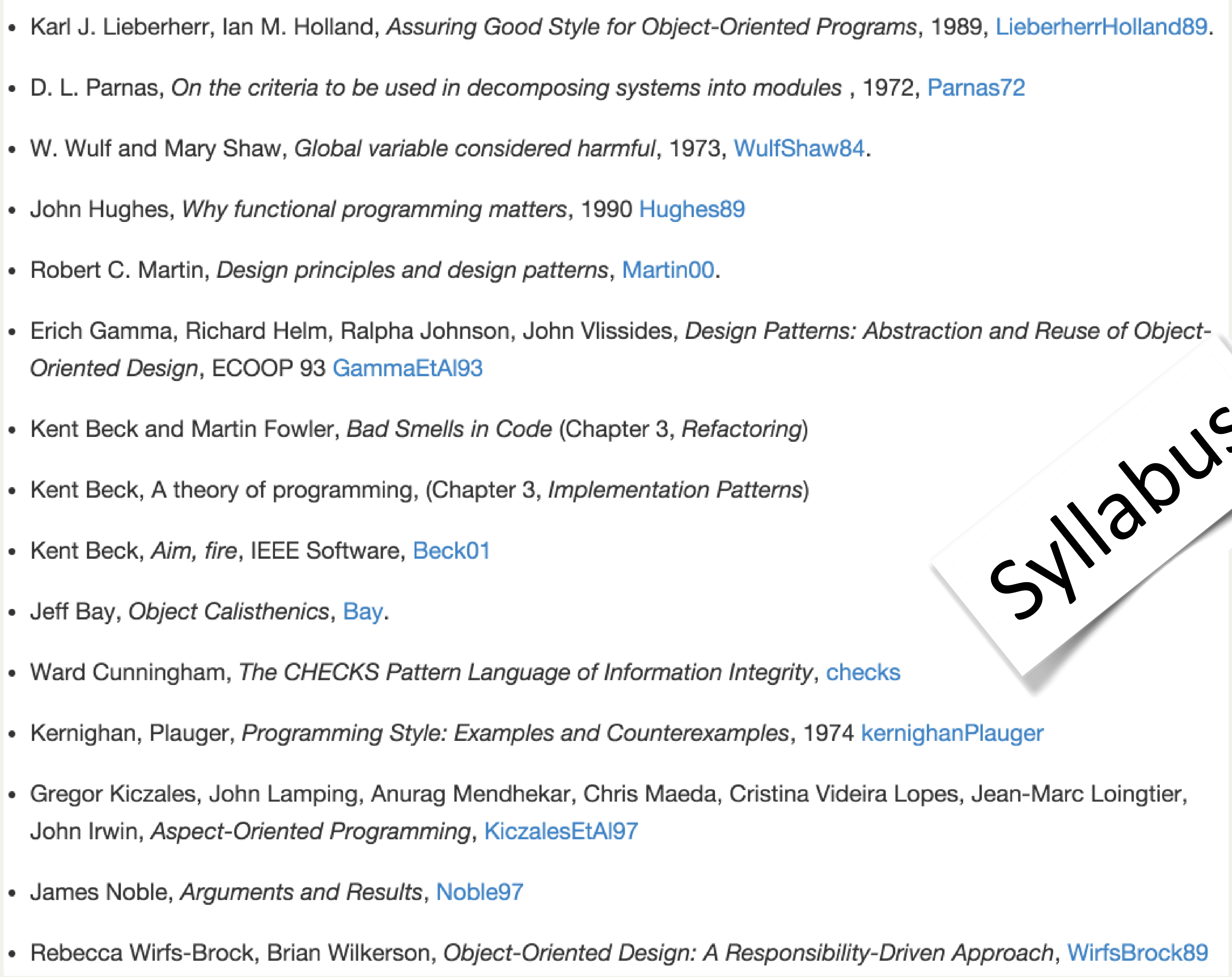
Course overview

- Lectures: every week on Wed morning
- Labs: exercise “not writing crap”
- Theory: papers + repo + book
- Exam: lectures + papers + repo + book

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
- ....

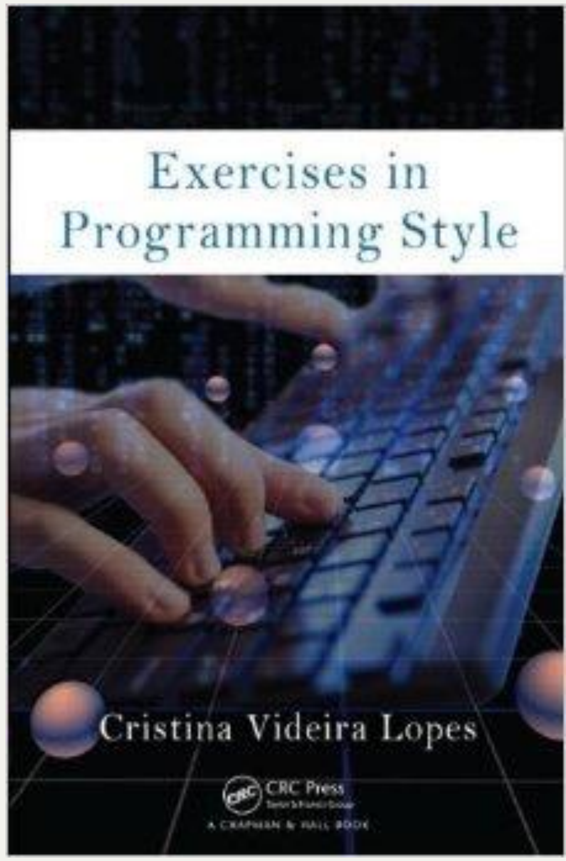


- Karl J. Lieberherr, Ian M. Holland, *Assuring Good Style for Object-Oriented Programs*, 1989, LieberherrHolland89.
- D. L. Parnas, *On the criteria to be used in decomposing systems into modules* , 1972, Parnas72
- W. Wulf and Mary Shaw, *Global variable considered harmful*, 1973, WulfShaw84.
- John 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 Object-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
- Rebecca Wirfs-Brock, Brian Wilkerson, *Object-Oriented Design: A Responsibility-Driven Approach*, WirfsBrock89

Syllabus

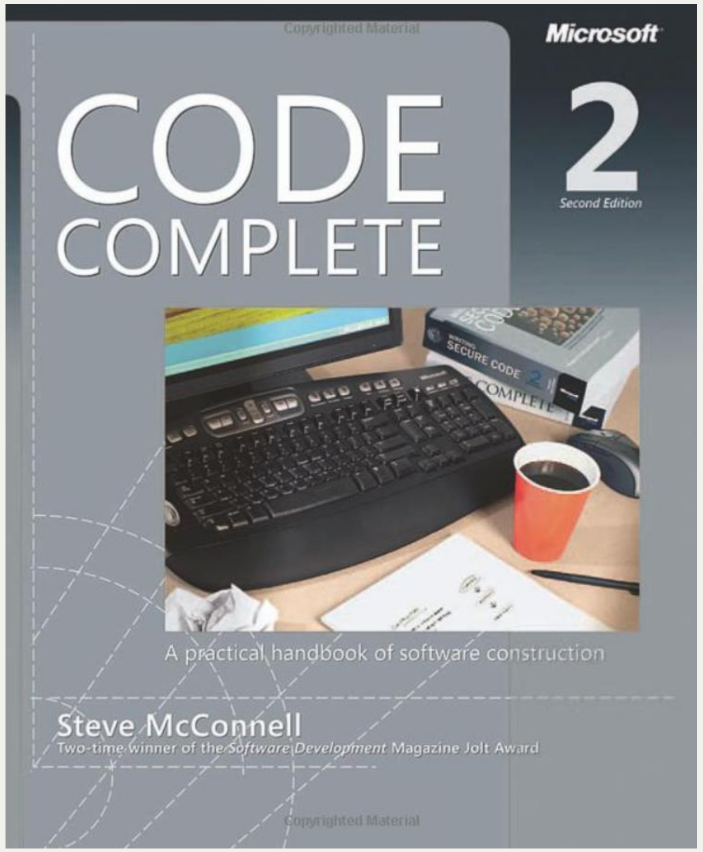
Exercises in Programming Style

- https://github.com/crista/exercises-in-programming-style
- Go through the repo
- (optionally) Read the book
- It will make you a better programmer
- Isolates design (how to formulate a solution)



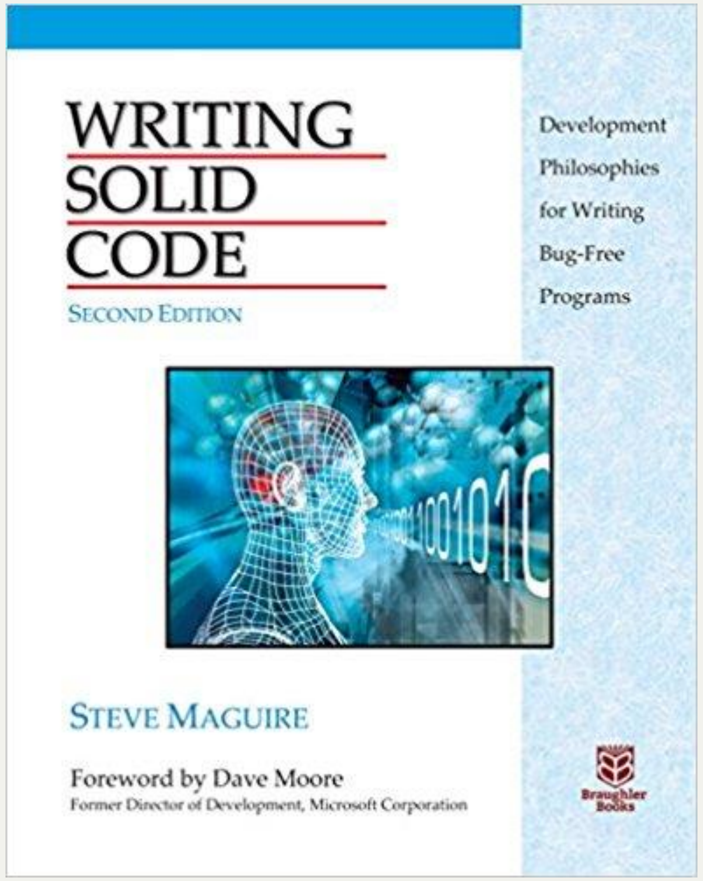
Code Complete

- (Was on the author's website, googleable)
- https://ondemand.construx.com/online-course/code-complete-essentials/
- Read the book


Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award

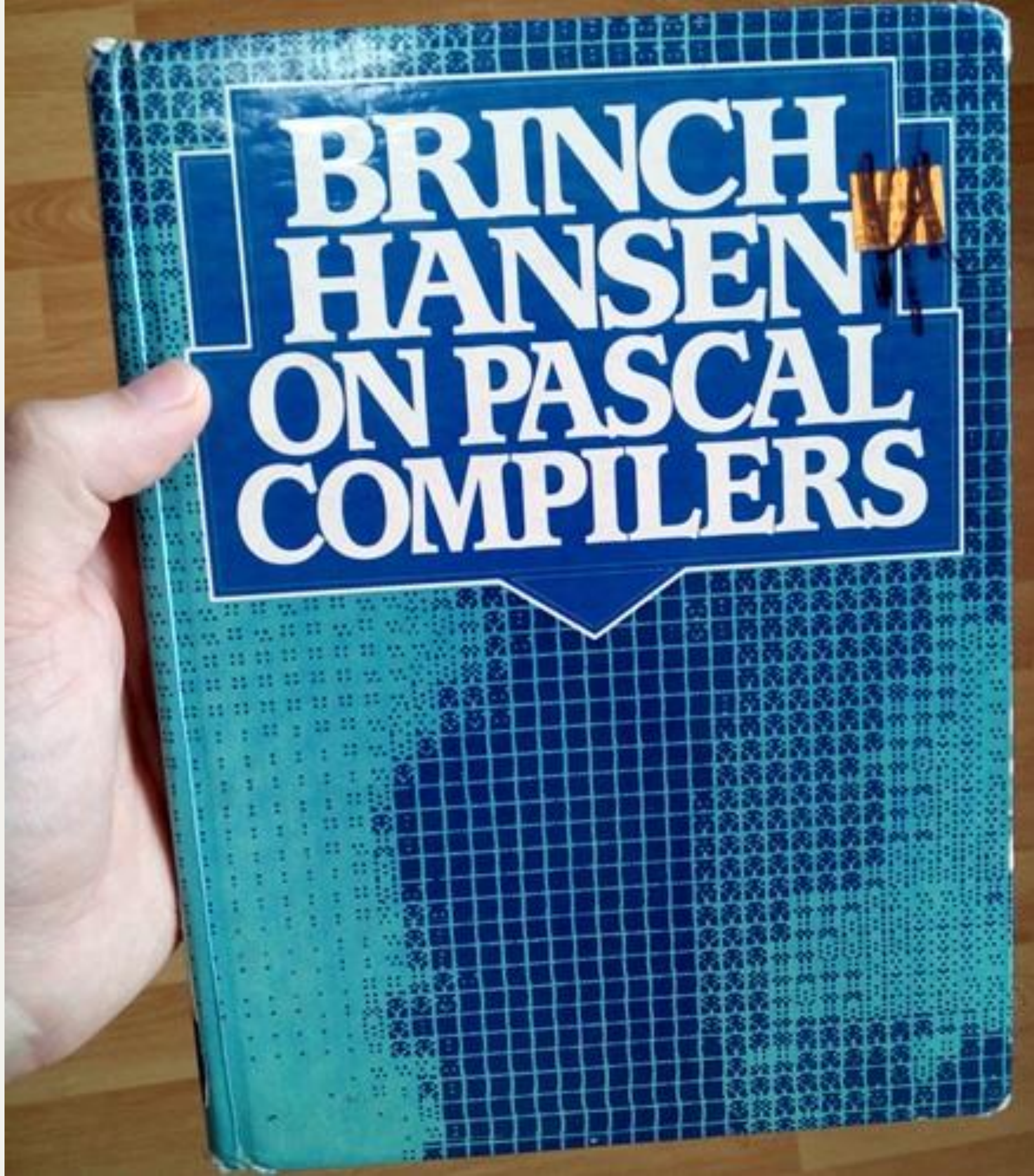
Backup: Writing Solid Code

- (Also googleable)
- Read the book

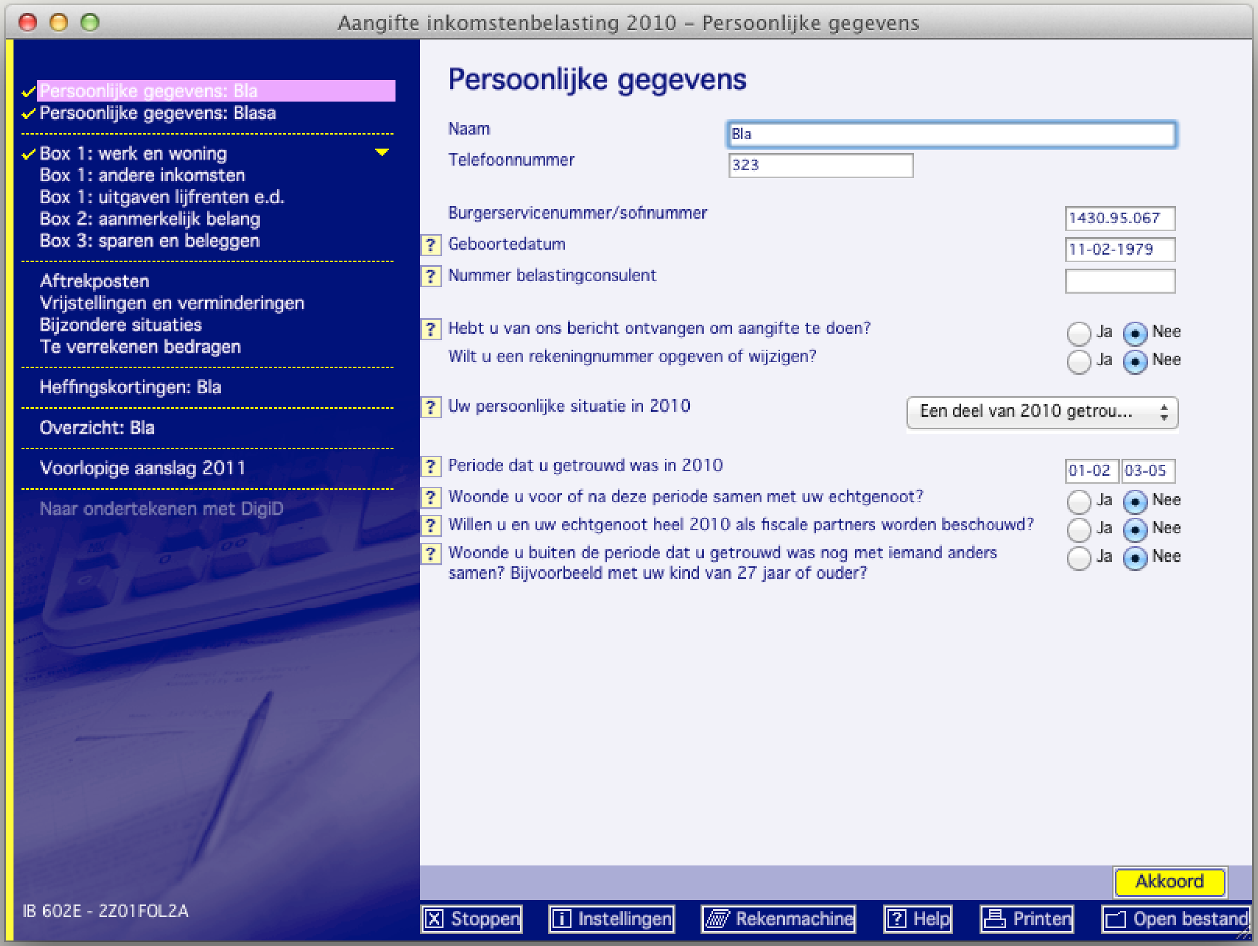

WRITING SOLID CODE
SECOND EDITION
Development Philosophies for Writing Bug-Free Programs
STEVE MAGUIRE
Foreword by Dave Moore
Former Director of Development, Microsoft Corporation
Braughler Books

Lab assignment

"The compiler course is probably the only chance you will get as a student to write a realistic program of 1000 lines (or more) and make it work"

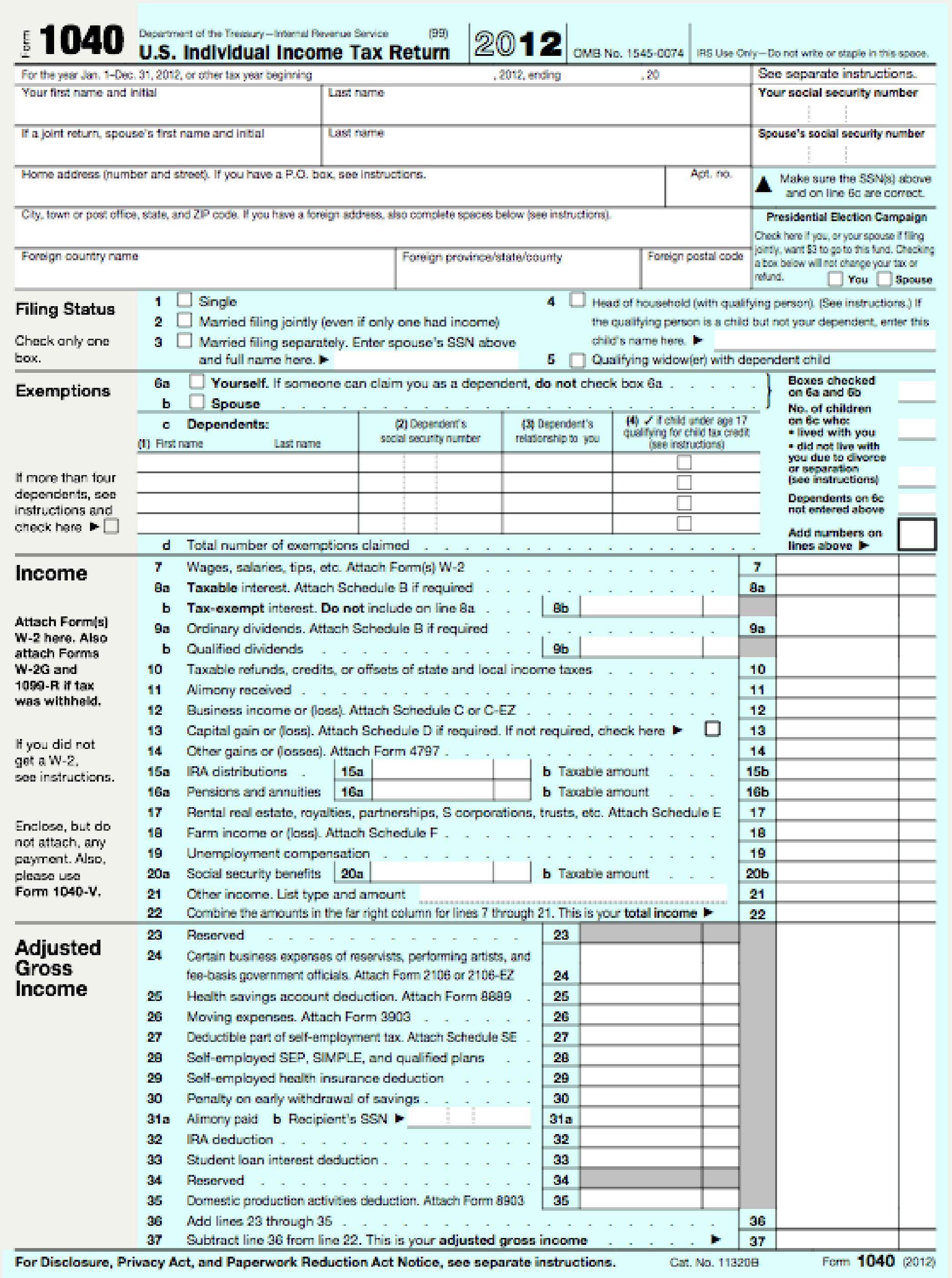


Lab assignment


Aangifte inkomstenbelasting 2010 – Persoonlijke gegevens
✓ Persoonlijke gegevens: Bla
✓ Persoonlijke gegevens: Blasa
✓ Box 1: werk en woning
Box 1: andere inkomsten
Box 1: uitgaven lijfrenten e.d.
Box 2: aanmerkelijk belang
Box 3: sparen en beleggen
Aftrekposten
Vrijstellingen en verminderingen
Bijzondere situaties
Te verrekenen bedragen
Heffingskortingen: Bla
Overzicht: Bla
Voorlopige aanslag 2011
Naar ondertekenen met DigiD
IB 602E - 2Z01FOL2A
Akkoord
Stoppen
Instellingen
Rekenmachine
Help
Printen
Open bestand

Persoonlijke gegevens

Naam	Bla
Telefoonnummer	323
Burgerservicenummer/sofinummer	1430.95.067
Geboortedatum	11-02-1979
Nummer belastingconsulent	
Hebt u van ons bericht ontvangen om aangifte te doen?	Ja [✓ Nee]
Wilt u een rekeningnummer opgeven of wijzigen?	Ja [✓ Nee]
Uw persoonlijke situatie in 2010	Een deel van 2010 getrou...
Periode dat u getrouwd was in 2010	01-02 03-05
Woonde u voor of na deze periode samen met uw echtgenoot?	Ja [✓ Nee]
Willen u en uw echtgenoot heel 2010 als fiscale partners worden beschouwd?	Ja [✓ Nee]
Woonde u buiten de periode dat u getrouwd was nog met iemand anders samen? Bijvoorbeeld met uw kind van 27 jaar of ouder?	Ja [✓ Nee]



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**

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 ▶

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

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

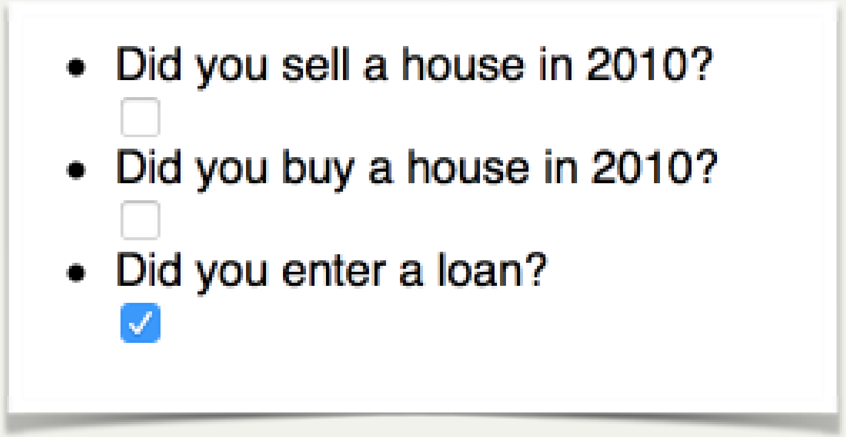
For Disclosure, Privacy Act, and Paperwork Reduction Act Notice, see separate instructions. Cat. No. 11320B Form **1040** (2012)

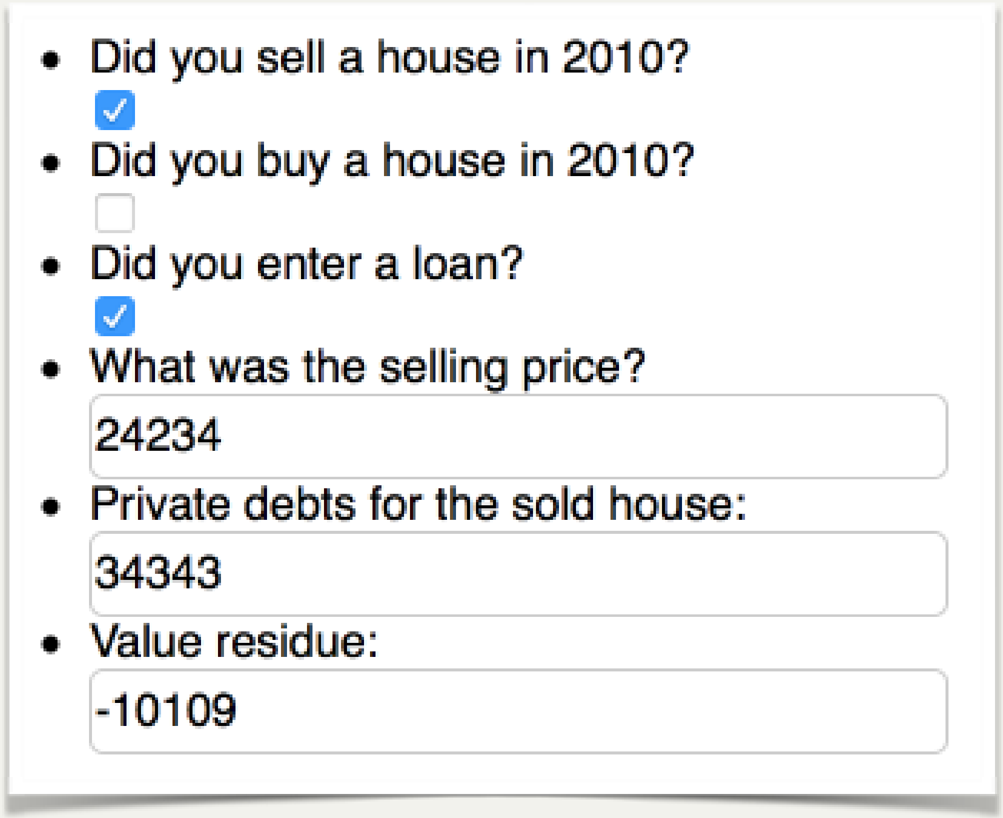
Part 1: Questionnaire Language (QL)

```
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?
  ✓


- Did you sell a house in 2010?
  ✓
- Did you buy a house in 2010?
  ☐
- Did you enter a loan?
  ✓
- What was the selling price?
  24234
- Private debts for the sold house:
  34343
- Value residue:
  -10109

raincode LABS
compiler experts

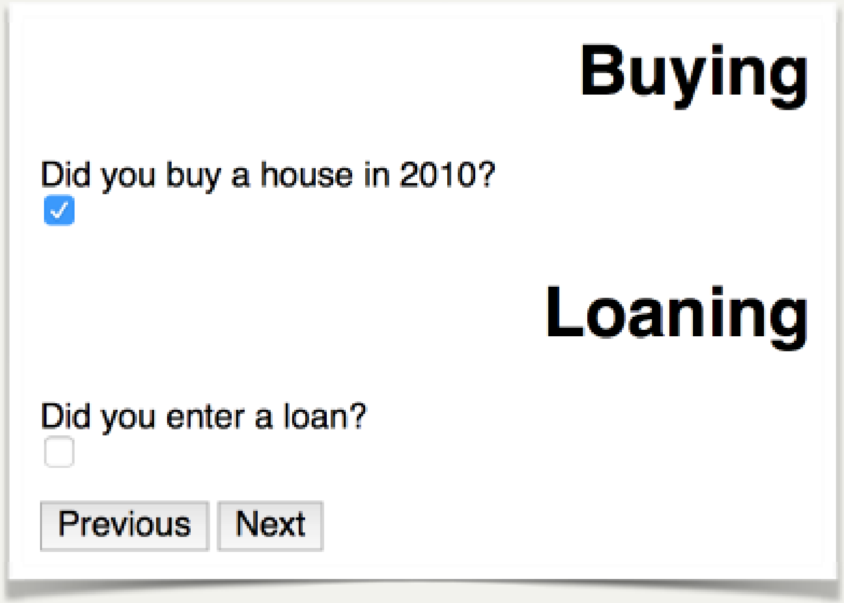
```
stylesheet taxOfficeExample
  page Housing {
    section "Buying"
      question hasBoughtHouse
        widget checkbox
    section "Loaning"
      question hasMaintLoan
  }

  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

raincode LABS
compiler experts



Buying

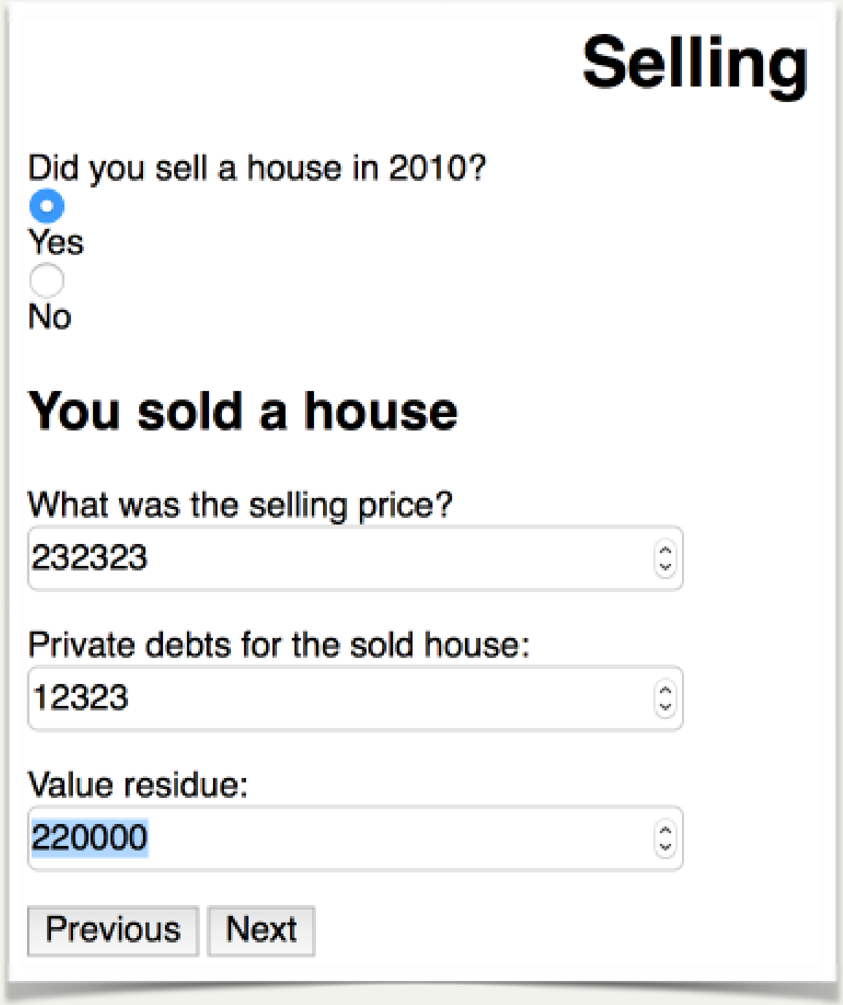
Did you buy a house in 2010?

[✓]

Loaning

Did you enter a loan?

[ ]



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

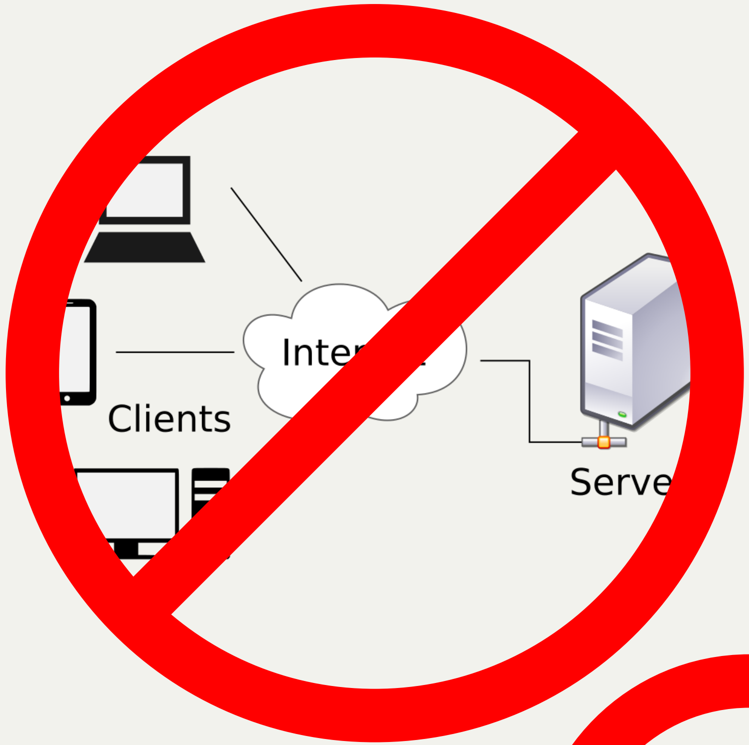
Part 1: QL

- Parser: text to abstract syntax tree (AST)
- AST hierarchy
- Static checker (types, well-formedness, ...)
- Expression evaluator
- Renderer as a GUI
  (interpreter! Not a compiler)

Part 2: QLS

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

Restrictions



autocrlf
true

raincode LABS
compiler experts

Programming language

- Java, C#, Javascript, Typescript, Haskell, Scala, Clojure, Erlang, Smalltalk/Pharo, Ruby, Python, Go, Dart, Swift, Objective-C, F#, Rust, Elm, …
- Use or at least look at one of the provided parsing skeletons for expressions in QL
    - Rats!, Jacc, ANTLR

GitHub

- Assignment to be completed *in teams of 2*
- https://github.com/software-engineering-amsterdam/endless-ql
- Use of this repository is required and graded
- Commit often! Push regularly!
- Branches are ok, but I will look at master

“Hour of code”

- During lab sessions (Wed 13:00–14:00)
- Convene in a single room
- 2-3 persons per session present their code
- No slides. Code.
- Not graded
- Constructive feedback and criticism
- Let’s help each other!

Schedule (cf. https://datanose.nl/#course[61106])

- Week 6: introduction lecture; start coding!
- Week 7: lecture on grammars and parsing; hour of code
- Week 8: lecture on SLs/DSLs; hour of code
- Week 9: lecture on styles/conventions; hour of code
- Week 10: lecture on code quality/smells; hour of code
- Week 11: lecture on design; hour of code
- Week 12: concluding lecture; demos
- Week 13: exam in OMHP
- Week 20: reexamination

Grading

- Dev grade
  - reduced weekly by signs of trouble
- Ship grade
  - depends on features & quality of the result
- Doc grade
  - written open-book no-internet exam
- Result is the average of the three, unless one is 5.5 or less

Dev grade: start with 10.0

	W6	W7	W8	W9	W10	W11	W12
git	-0.2	-0.2	-0.2	-0.2	-0.2	-0.2	-0.2
questions @ lecture		-0.2	-0.2	-0.2	-0.2	-0.2	-0.2
layout / naming		-0.2	-0.2	-0.2	-0.2	-0.2	-0.2
executability		-0.2	-0.2	-0.2	-0.2	-0.2	-0.2
testability			-0.2	-0.2	-0.2	-0.2	-0.2
encapsulation / abstraction			-0.2	-0.2	-0.2	-0.2	-0.2
DRY / YAGNI				-0.2	-0.2	-0.2	-0.2
QLS					-0.2	-0.2	-0.2

Ship grade components

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

Simplicity

raincode LABS
compiler experts

Conclusion

- All info on GitHub:
  - https://github.com/software-engineering-amsterdam/software-construction/tree/master/2017-2018/
- Send account info to vadim@grammarware.net
- Decide on the language
- Start coding
- Start reading