# Rozdíl mezi teorií a praxí aneb Co chybí absolventům informatických oborů

Paul Hrdina

## Agenda

- Introduction
- Paul’s view on the *State of the Industry*
- Paul’s Amendment to *Universal Declaration of Human Rights*
- Paul’s view on the *State of the University Education*
- Paul’s approach
- Q&A

## Paul Hrdina — Introduction

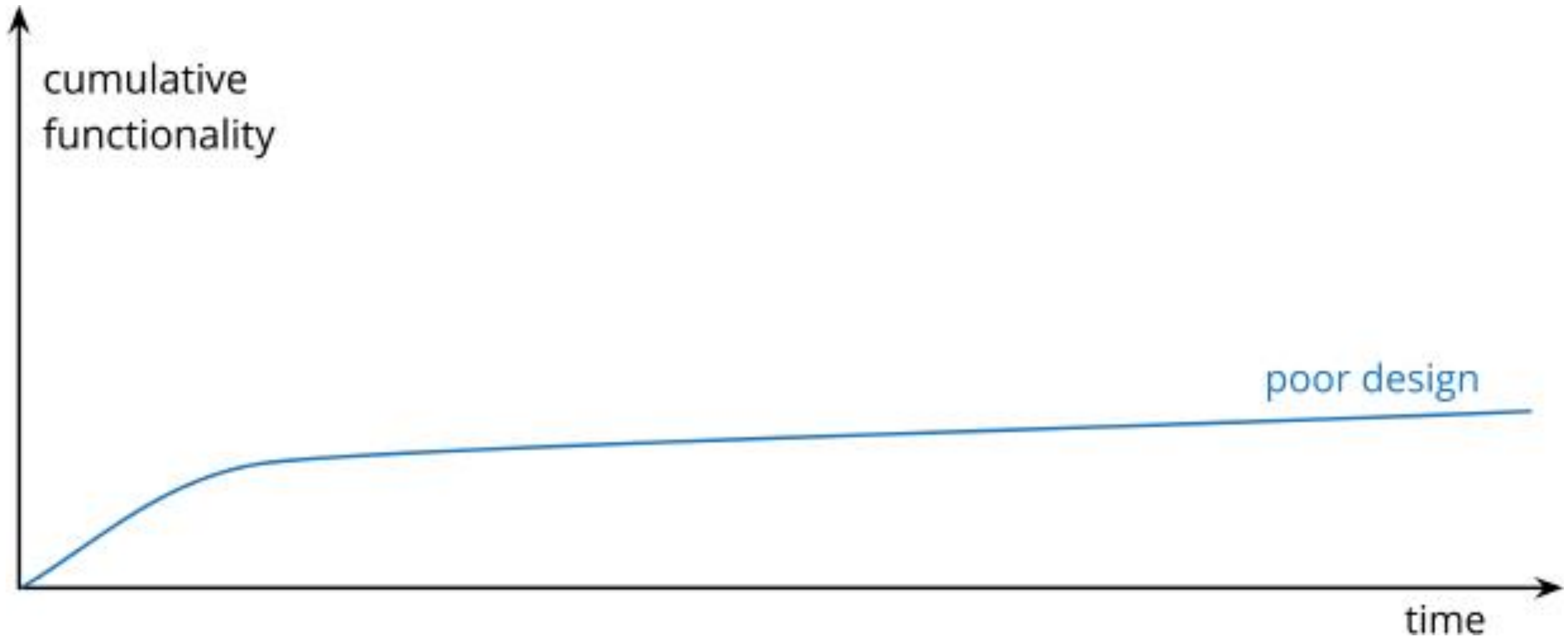
- Professional programmer since June 1996
- Master degree at CTU in Prague
- Worked for a range of companies
  - Czech ones (Small & Large)
  - International ones
- Programmed in a number of statically-typed languages
  - Pascal, C, C++, Java
- Working for (Oracle) NetSuite since April 2011
- Teaching at FI MUNI since September 2018

## Paul’s view on the *State of the Industry*

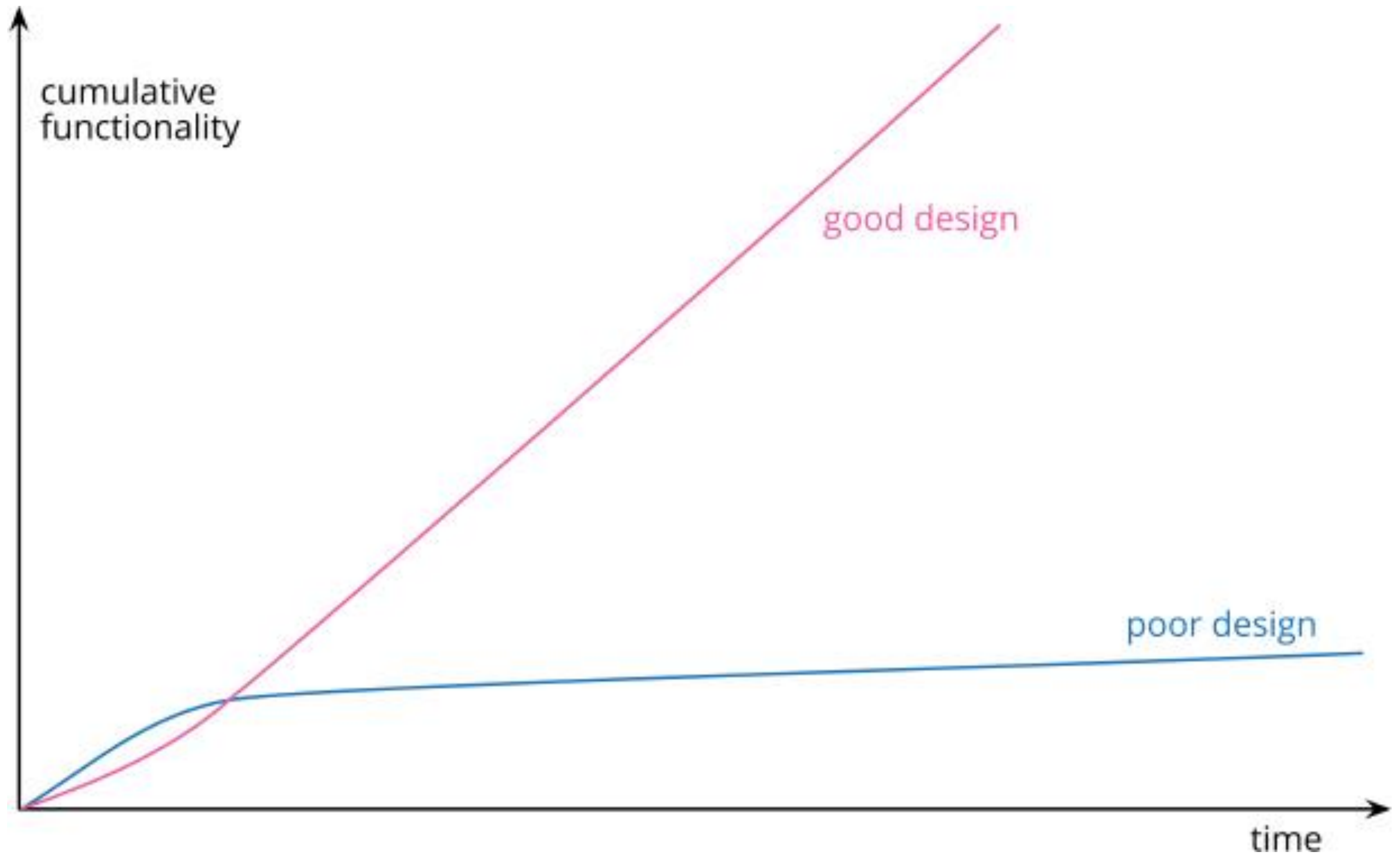
## Industry Reality

- Full of large systems (100k+ LOC)
- Programmers are most typically maintaining/extending them
- Deciphering existing code is the prevalent task
- Algorithmic tasks are significantly rare

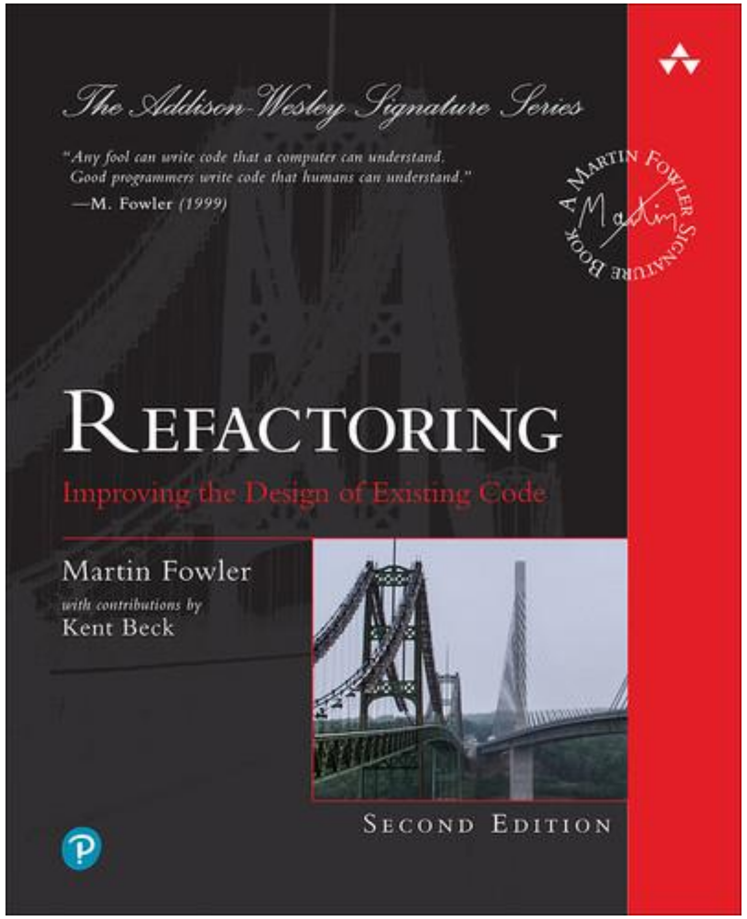
## Industry Reality: SW ~= HW



## Industry Desire: SW = SW



# Refactoring = the Design tool


The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by
Kent Beck
Second Edition

*Martin Fowler*

**Refactoring** (Second Edition)

Pearson, 2019

ISBN **978-0-13-475759-9**

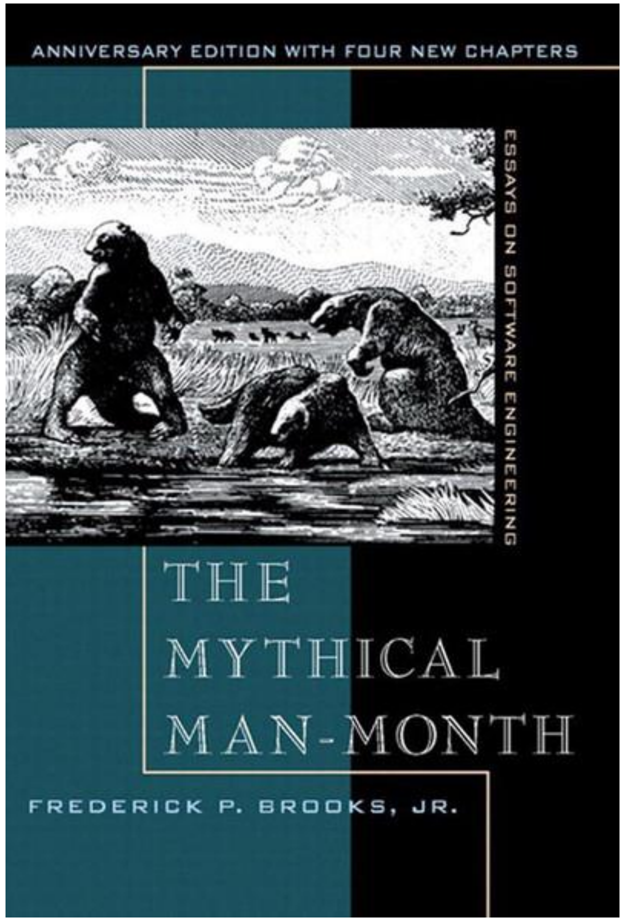
# Industry Death Spiral

- Many programmers struggle with the monsters they operate on
- The monsters (even their parts) are hard to test automatically
  - Refactoring (by the book) is therefore not an option
  - The code becomes gradually worse and people are afraid to touch it
- Managers are unable to spot divergence from good paths early
- Companies are hiring more and more manpower
  - The average skill level is constantly decreasing
- It resembles brute force
  - Adding more nodes to fix an algorithm which is bad (in the first place)

## Many books out there on the topic

- Nobody is reading them
  - Therefore, reinventing the wheel all the time
- Hard skills are not enough on their own
  - Soft skills are of the same importance for the success
- Source code is yet another way of human communication
  - Needs to be understood quickly
  - Needs to be easy to change

## The Mythical Man-Month


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

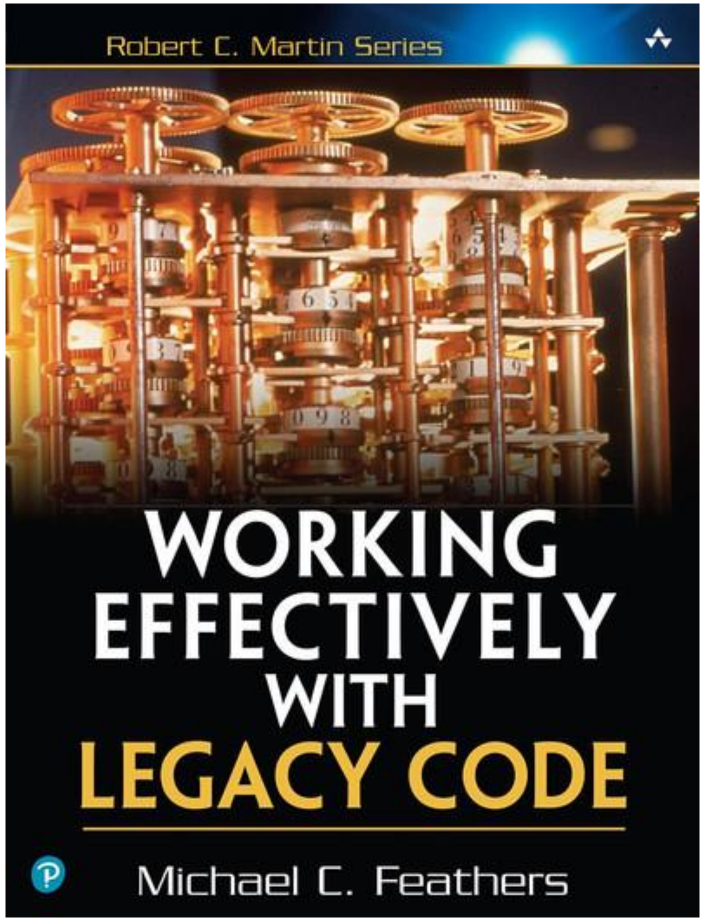
*Frederick P. Brooks*

**The Mythical Man-Month**
(Anniversary Edition)

Addison-Wesley, 1995

ISBN **978-0-201-83595-3**

## Working Effectively with Legacy Code



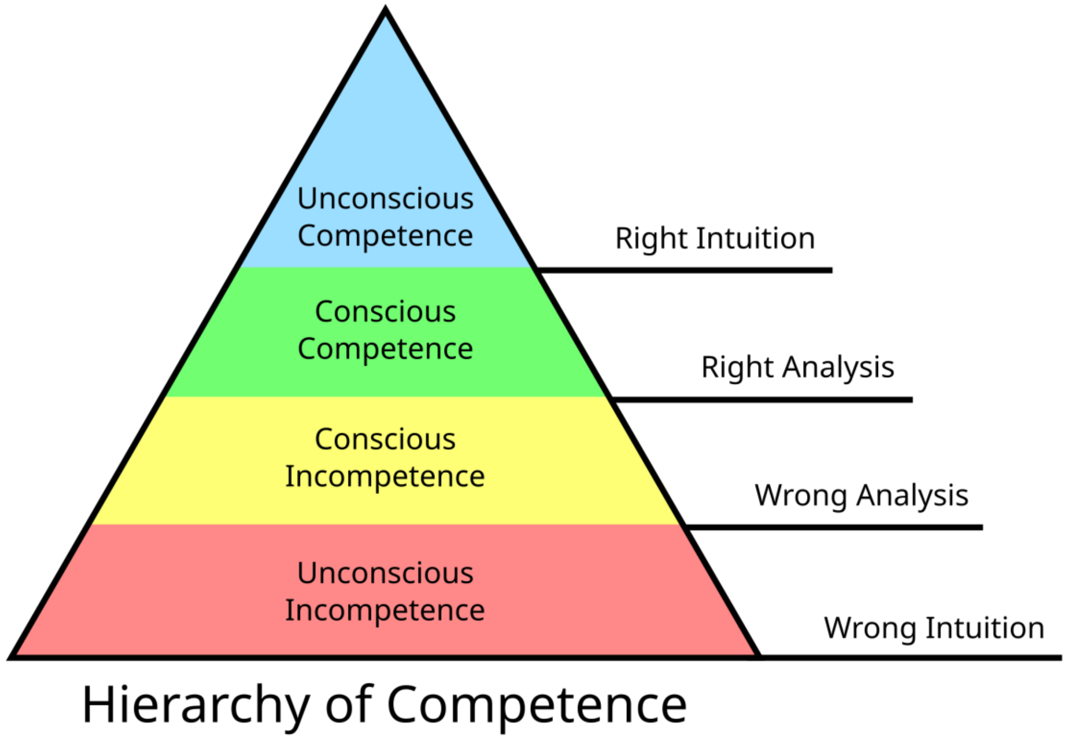
*Michael C. Feathers*

**Working Effectively with Legacy Code**

Pearson, 2004

ISBN **978-0-13-117705-5**

# Four Stages of Competence



## Paul’s Amendment to *Universal Declaration of Human Rights*

## Article 31 — Knowing

Everyone has the right not to know something and to admit it freely, without any fear of punishment.

“*Real knowledge is to know the extent of one's ignorance.*”

— Confucius

“*I know that I know nothing.*”

— Socrates

“*Anyone who claims to know something does not yet have the necessary knowledge.*”

— St. Paul

## Article 32 — Questions

Everyone has the right to ask questions for whatever they feel unclear or curious about. No one can expect to be understood by others just by expressing themselves and not confirming the result.

“*The man who asks a question is a fool for a minute, the man who does not ask is a fool for life.*”

— Confucius

## Article 33 — Mistakes

Everyone has the right to make mistakes while conducting experiments, provided they take feedback seriously and accept the responsibility for any damage and deal with it.

"*By three methods we may learn wisdom:*

- *First, by reflection, which is noblest;*
- *Second, by imitation, which is easiest;*
- *and third by experience, which is the bitterest.*"

— Confucius

## Paul’s view on the *State of the University Education*

## University Education successes

- Mathematics
- Algorithms
- Complexity
- Graphs
- Automata
- Functional paradigm

- No retraining course can substitute that in full

## University Education reality

- Graduates are used to many different languages/environments
  - Nice width
  - Only shallow knowledge in any of them
- Missing significant depth in a single one
  - One IDE instead of many editors is essential for productivity
  - Deep knowledge of one powerful language can then be transferred to other languages easily
- Too simple systems for programming tasks
  - Disciplines such as TDD do not make any sense to them
  - Open Source projects might help here

# University Education major shortfalls

- Keyboard as the tool for working with text
  - Source code is text, not graphics
- Everything at once (on the first attempt)
  - Incrementality is the only working strategy in complex environments
- Bottom-up (Big Design Up Front) development
  - Top-down approach brings important feedback much faster
- Bringing existing literature to the students

## Paul’s Drivers towards the University

- Graduates entering industry were nice colleagues
- Many of them were still hard to work with
    - They were mostly solitaires (and didn’t know how to cooperate)
    - They pretended they know (but in fact they didn’t)
    - They didn’t tend to ask (even when they were stuck)
- So that I decided to bring the common problems from the industry back to the university, where it is cheaper to experiment (and fail)

## Paul’s approach

## Experiential Learning — How does it work?

- Participants first experience a suitable artificial exercise
  - Amplifying desired aspects (Essential Complexity)
  - Omitting orthogonal aspects (Accidental Complexity)
- Then they reflect on their performance
  - And formulate lessons learned
- Finally, they can be given the relevant theory and background

## Experiential Learning

- Why does it work so great?
  - Doing creates deeper neural pathways than consuming
- Is typically used for teaching soft skills
  - Brain-Based Coaching (NeuroLeadership Institute)
  - Teaching Lab @ FI MUNI
  - Prázdninová škola Lipnice, Instruktoři Brno, …
- I have successfully used it for teaching hard skills
  - But there is still significant gap towards the real complexity

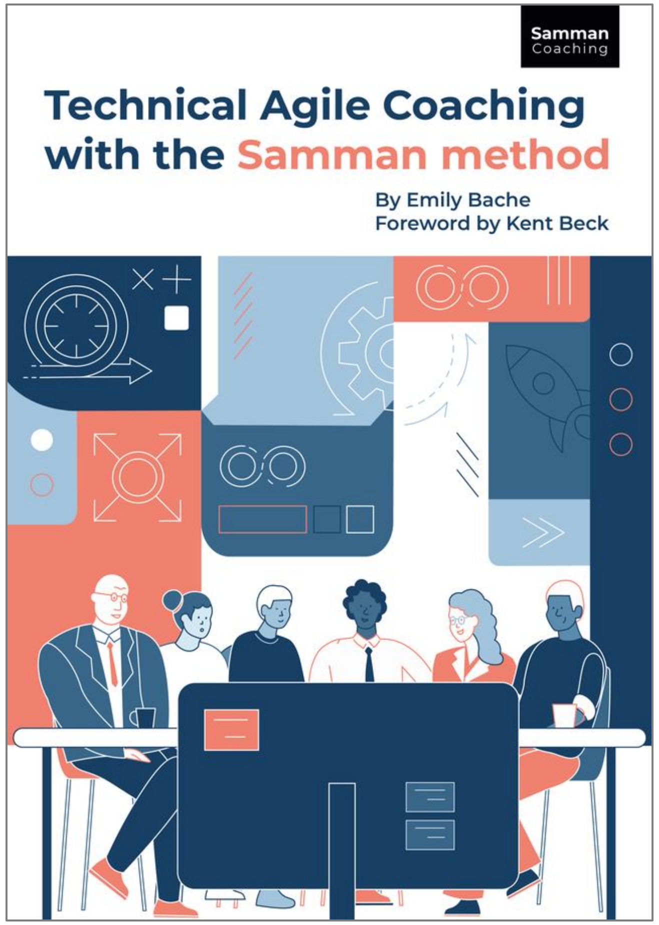
## Paul’s Teaching at FI MUNI

- I soon realized I need to change the courses which I was part of
  - **PV168** (Java II)
  - **PV260** (Software Quality)
- Novelty in teaching
  - Incorporating Experiential Learning methods into Seminars & Lectures
  - Promoting Team Work and Communication during Semestral Projects
  - Involving real-life roles (Customer via Product Manager, Tech Lead, UX)
  - Top-down approach and feedback-driven Iterative development
  - Giving extensive feedback via Code Reviews

## Samman Technical Coaching

- Next generation of Experiential Learning for hard skills
  - Fixes the gap between artificial exercises and real complexity
- Interleaves simplistic exercises (**Learning Hour**) with work on real industry code base (**Ensemble Working**)
  - First training skills in simplistic environment
  - Then practicing the same skills in realistically complex environment
- Works especially well for people from the same team
  - Where they share ownership of the code

# Samman Technical Coaching



*Emily Bache*

**Technical Agile Coaching with the Samman Method**

Bache Consulting, 2021

ISBN **978-91-986769-0-7**

## Q&A