

Ako som sa v škole (ne)učil

Kto som

... web man ...

marcus@gratex.com

Ing. Martin Marko

FEI STU (1993- 1999 ????)

Gratex International a.s. (1996 -)

Head of Software Development Group

a_{ntichrist}.in.the.k_{itchen}

<https://github.com/ainthek>

<http://ainthek.blogspot.sk>

<https://twitter.com/ainthek>



O com to bude ?

- Náhodné a pomiešané a silne subjektívne (niektoré dokonca ukradnuté) názory na témy zo školy, IT praxe a podobne
- slovensko anglicky, s preklepmi a obcas bez diakritiki ;-))))

Načo to bude ?

- entertainment a “provokácia”
- provokácia => zamyslenie (aspoň u inteligentých jedincov)
- zamyslenie => úžitok/hodnota... (aspoň u pracovitých jedincov)
- úžitok => lepší svet

Ako som sa v škole (ne)učil

- Čo mi dala škola najviac
 - dokázal som zužitkovať, obohatilo ma
- Čo som sa mohol v škole naučiť a nenaučil sa
 - lebo mi to “nevedeli predat’”
 - lebo som nevedel, načo mi to bude
 - lebo som neveril, že mi to na niečo bude
- Čo som sa musel učiť ozaj zbytočne
 - naozaj ?

Prečo je to dôležité

6 rokov školy = 6 rokov * 160 dní * 8 hodín

7680 hodín

ktoré Vám niekto dedikoval na učenie sa

Keď odídete zo školy, a začnete pracovať a mať deti a niekde ukradene hodinu denne na učenie každý deň, bude vám to trvať

21 rokov

Keby som mal vybrať jednu vec, čo sa v škole neucilo a necvicilo

• Citat cudzi kod”“alebo co tym basnik myslel ?”

- Je pre mna paradoxne ze v na predmetoch slovinciny sme hodiny a hodiny travili citanim diel niekoho ineho o niecom co nas nezaujimalo.
- O co horsie, tyrali nas utrapnymi diskusiami o tom co tym (spravidla mrtvy) autor myslel.
- Je zaujimave ze tak abstraktej a subjektivej urovni ako su basne, autori s ktorymi sa neda porozpravat boli ucitelia skalopevne presvedceni ze je to dolezite a pravdive a davali nam 5 ky ked sme sa to neucili.
- Preto sa v IT skolach nezaoberame “lustenim cudzich kodov” a hladanim: myslienok (algoritmov), estetickych hodnot (jednoduchosti a efektivity kodu) a podobnych “literarnych nezmyslov ?” navyse autori spravidla ziju a diskusia je mozna....
- je paradoxne, ako je to v literature zbytocne a skolstvom vynuovane
- a v IT uzitocne a skolstvom ale aj praxou a aj nami jednotlivcami zanedbavane
- Preto v literature sa to uci na VELMI KONKRETNÝCH prikladoch z minulosti ?
- A v IT nas ucia ako napisat knihu, basen esej, bez toho aby nas motivovali ak uz nie nutili poznat cudziu tvorbu predtym ako napiseme svoje prve “pulp fiction” ?
- Nevie si predstavit ziaka VSMU ktory by pocas skoly nemusel **nastudovat** divadelnu hru, **precitat**, **vysvetlit** a **porovnat** kopu kniziek **rozných** zanrov.
- nas ucia len o tom ze existuju autori, zanre, styly a mame si sptavit nazor a nieco kodnut.....

Citajte (cudzi kod)

- Vyberte si
 - styl co vam sedi, aky vam robi prijemne, alebo v akom by ste raz chceli napisat basen, alebo o ktorom nic neviete a zaujima vas najviac: (romantika, gotika, RAP, whatever), grafika, robotiky, cista matika, video, spracovanie hudby,
- Zacnite citat
 - chodte do kniznice
 - git clone, subl .
- Sharujte pocity, otazky, neistoty, nesuhlas
 - porozpravajte sa o knizke z priatelkou, chodte na hipisacky literarny vexer
 - napiste do fora, napiste issue
- Opravte, vylepsite
 - takmer nemozne, errata
 - fork and merge upstream
 - fork, branch, pull request
- Kritici
 - v literature: zaznavani
 - v IT: zaznavani
- Stale mi to pripada jednoduchsie u nas ako u nich a podla mna to nedokazeme dostatočne silne vyuzit.



.. a v IT nas ucia ako napisat knihu, basen esej, bez toho aby nas motivovali ak uz nie nutili poznat cudziu tvorbu predtym ako napiseme svoje prve “**pulp fiction**” ?

PULP FICTION

Keby som mal vybrat jednu vec, co sa v skole neucilo, necvicilo

- Urobil som taku anketu medzi kolegami:
- **Source Control** - bola jedna z najcastejsoch odpovedi na tuto otazku.
- Co je SC ? Co Vas o tom ucia v skole ? na akom predmete ?
- SC je dneska filozofia, zivotny kodersky styl, zakladna gramotnost, navyky, best practices, social network and hall of blame and fame, your CV, neni to produkt, neni to software
- “moja” sucasna volba: **GIT, and GITHUB as social playground.**

GIT (niekedy inokedy)

- ako to suvisi z predchadzajucou temou ?
- spravte si konto na github-e a zacnite:
 - hladat
 - citat
 - badat, pochybovat, pytat sa
 - vylepsovat cudzie veci
 -
 - tvorit a nechaj ostatnych vam pomahat

GIT

- <http://anti-pattern.com/github-is-your-resume-now>
- <https://blog.jcoglan.com/2013/11/15/why-github-is-not-your-cv/>

BOHATSTVO

- Ja robim IT lebo som si precital v novinach ze po bankaroch je to najlepsie platena praca (1993)
- Bohatstvo != peniaze
- **Bohatstvo = vytvaranie a hromadenie niechoho co iny ludia potrebuju.**
- **Skola != antipatern na tuto definiciu, zasadne aspon za mojich cias nas k tomu nik neviedol, kazdy ma svoj diplom, kazdy sa ma ucit sam za seba, “vedu vas k samostatnosti, nie k vzajomnosti”**
- **Prax ? , som skepticky ze najdete vela zhodnych projektov**
- **Bavi ma to, uspokojuje ma to = prijemne, sebastrednost, zbytocnost, neuzitocnost (hriech ?) DOCASNE statie, v 40ke si budete chciet kupit buldozer aby ste po sebe videli nejaky hmatatelny vysledok (jamu)**
- **Kazdy “projekt” zacinajte otazkou - o co bude svet krajsi ked to spravim, ak je to tazke spytajte sa o co bude svet horsi ak to nespravim (spravidla o nic, ak to za nieco stalo spravi to niekto a lepsie), len vy nebudete slavy (hriech ? zase ?)**

BOHATSTVO A UZITOCNOST

- Schopny a uzitocny jedinec
 - jedinec + vyskum = uzitocna idea
 - jedinec “v praxi “ = uzitocna vec
 - vyskum “v praxi” = uzitocna vec
 - jedinec “v neuzitocnej praxi” != uzitocna vec
- Koho ma vychovavat skola ?
 - SOU programatorske
 - 200 vedeckych pracovníkov rocne ?
 - jedincov pre neuzitocnu prax nech vznikaju uzitocne veci ?

BOHATSTVO

- Odpoved - Open Source
 - Celosvetove bohatsvo
 - nie preto ze je to zadarmo
 - Preto ze to ludia CHCU/POTREBUJU
 - je tvorene spolocne (contributores)
 - je tvorene otvorene (issues, diskusie, hadky, flame warrs)
- RASTIE a HROMADI SA
- **Pre mna (a pre zjednodusenie) dneska: GITHUB**

Open Source Projects

GoldenCheetah DOWNLOAD FEATURES TUTORIALS CONTRIBUTORS SCIENCE MORE

— Photo: Justin Kinsdale

Features

Import all popular file formats

- TrainingPeaks (WKO, PWO)
- PowerTap (RAW, CSV)
- ANT+ (FIT)
- SportTracks (FITLOG)
- Ambit (SML)
- Sigma (SLF, SMF)
- Ergomo (CSV)
- Google Earth (KML)
- Garmin (TCX, GPX)
- Polar (HRM)
- SRM/Mini (SRM)
- Compensator (TXT)
- iBike (CSV)
- MotoACTV (CSV)

Cloud Integration

- Dropbox
- TrainingPeaks
- Strava
- Trainingstagebuch.org
- Ride With GPS
- Cycling Analytics
- Settopps
- SportPlusHealth
- Velo Hero
- Withings Weight
- Zeo Sleep
- Google Calendar
- Twitter
- ErgDB


Download directly

- SRM PowerControl 5/6/7
- PowerTap Cenzo
- O Synce Macro
- CycleOps Joule
- Moxy Muscle Oxygen Monitor

Forensic Ride Analysis

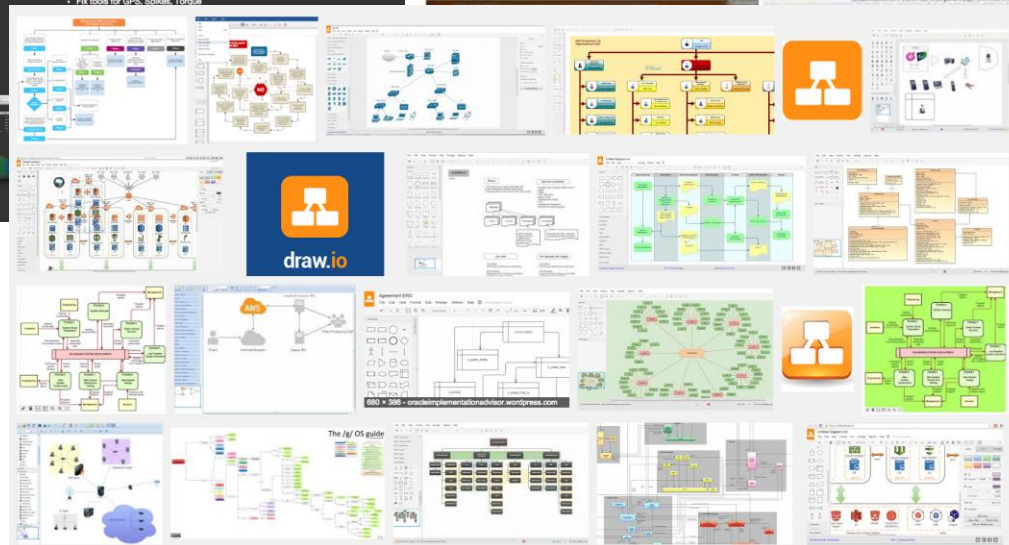
Multiple charts to examine and analyse ride and interval data including:

- Summary, Metadata
- Ride Plot
- Stress Plot
- Pedal Force/Velocity
- Histogram
- 2d and 3d scatter
- HR v Power
- Map (Google / Bing)
- Averbits
- Critical Power (PC)



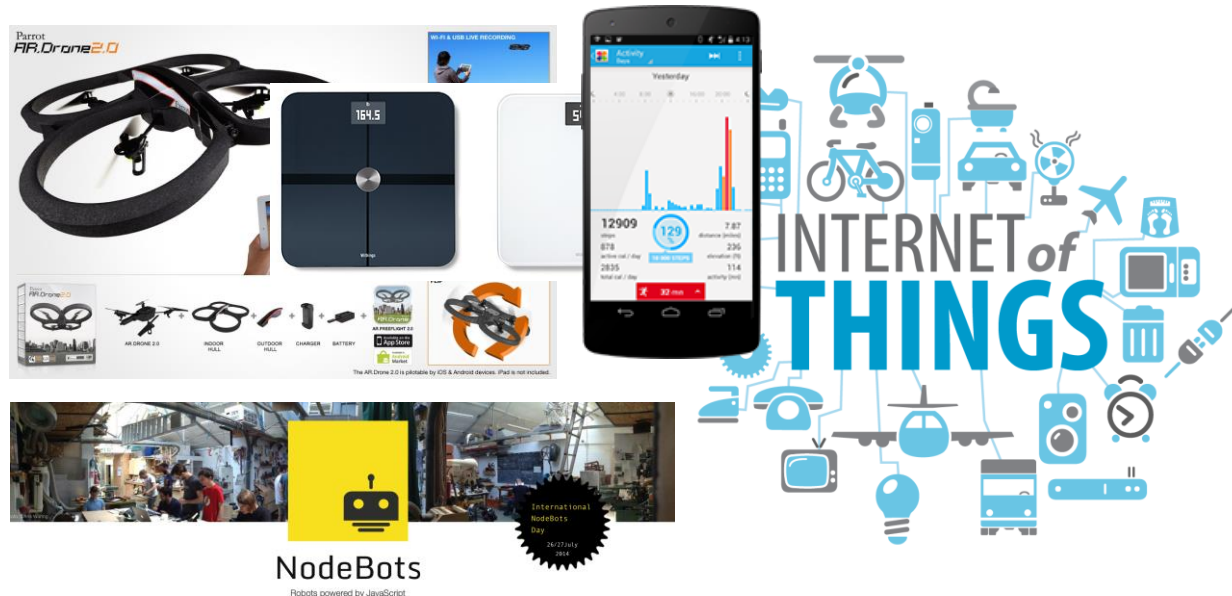
W_{max} Stress Chart

Critical Power Chart



Automatizácia

- Ak sa nebudete ucit skoncite ako automatizeri....
- Aplikovana informatika, Matlab, Vysoké pece ? Riadiace linky ? grrrrrrrr
- Keby mi niekto povedal ze budem mat take veci pozajtra v byte, neveril by som mu. Ale isto mi to nepovedal a neukazal, urcite, lebo by som si pametal ze som mu neveril.
- Toys, Internet Of Things, Computer Vision,



Software Management

- však ja nechcem byť manager, nikdy som nechcel
- no ja musím byť manager, 20+ ľudí v projektovom team-e, 140 ľudí v grupe na 10+ projektoch a 5+ architektúrach
- každý musí byť manager aspoň sám sebe, a kódu na ktorom robí (umyselne nepisem svojemu kódu)

Funkcionálne programovanie

- **Lisp** - prvý raz som videl **inu programovaciu paradigmu**, odkedy som ju videl, začal som úlne inak (**ľahšie**) **programovať** aj v C a Java
- **JavaScript** - Everybody knows that JavaScript is a **multi-paradigm language**, and it can be used to program functionally. Practically all functional idioms can be used directly: higher-order functions, recursion, closures, and so on. **The recent resurgence of Functional Programming (FP) brings functional methodologies in the mainstream.** FP fundamentals gave us a lot of powerful idioms: iterative functions, which can replace loops, list processing in general, function manipulations, and many other things, which helps us to keep our code small yet concise, more **powerful, and more fun.**
- <http://www.lazutkin.com/blog/2008/01/12/functional-fun-javascript-dojo/>
- **Functional Reactive Programming (FRP)**
- In short - **no variables please**



Logické programovanie - Prolog

- Prolog is **declarative**: the program logic is expressed in terms of relations, represented as facts and **rules**.
- Prečo som sa to neučil ? Keďže som bol zamilovaný do LISPu, momentálne som inú lásku nehladalm príklady na ktorých sme sa to učili boli príliš “umelo” inteligentné a nie reálne z života na iných predmetoch
- **XSLT - The basic processing paradigm is pattern matching** - roky som úplne zle programoval aj v XSLT, pomocou: match root a call template a for each, na miesto apply templates, **len preto že som sa neučil PROLOG**
- **Rule Engines** - much easier code for certain type of problems

Operačné systémy

- Unix –
 - filozofia mi vtedy asi ušla
 - nemal som „načom“ (fyzicky, dusevne) skúšať
 - MS (DOS, Windows) everywhere.....

shell scripting, BASH

- Zelené žaby
- Učili nás **to len na administráciu systému**, ja som **nechcel byť "admin" ani systémák**, chcel som byť kóder....
- DNESKA: kódujem/zliepam v tom 80 percent vecí... preco ?
 - heterogenny,
 - kody uz nie su libky ale programy a sluzby,
 - data su uz zase "text"
- MUSIM sa to ucit sam a uz mi to moc nejde
- Dnes uz musim/**chcem/mozem byt admin** (DevOps, ale to je zase na inokedy)

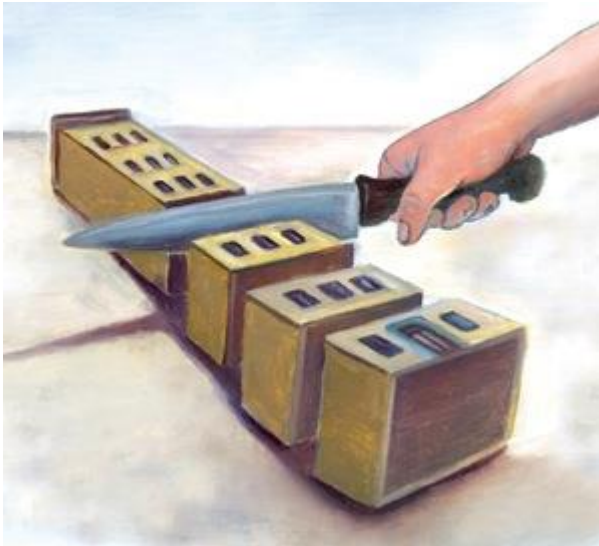
streams and pipes

- toto ani nezacinajme, neskoncili by sme nikdy, v skratke:
- neprogramuje sa takto:
 - `myCoolMethodCompiledInSomeSickLanguage(){ a; b; c; d}`
- ale takto:
 - `a|b|c|d`
 - `a|c|d|x`
 - `Y | c | F`
- Neprogramuje sa takto:
 - `String fileContent=readFile();`
 - `fileContent.replace("<";"{"`
 - `http.send("ebay.com/order")`
- Ale takto:
 - `readFileAsync().pipe(xml2json).pipe(request.post("ebay.com/order"));`
- A zase tu vidime paralelu z FP - **no variables please**

paralelné programovanie

- Flynnova taxonómia, Amhdalov zákon, Gustafsonov zákon
 - Systémy so **zdieľanou a distribuovanou pamäťou**, multiprocessory a multipočítače
 - Podmienky paralelizmu, **dátová a zdrojová nezávislosť**
 - Zdroje paralelizmu, paralelizmus na úrovni inštrukcií, dátový paralelizmus, paralelizmus úloh
 - Návrh paralelných programov, komunikácia, synchronizácia (atrické operácie, **bariery, semaféry, mutexy**), závislosť medzi dátami, dekompozícia, granularita, rozkladanie záťaže
 - Paralelné programovacie modely, model vlákien, model zasielania správ
 - Explicité použitie vlákien – Pthreads (resp. Java threads, Win32 threads, ...)
 - Implicitné použitie vlákien – OpenMP
 - Programovanie systémov s distribuovanou pamäťou – MPI
 - Programovanie mnohjadrových grafických procesorov – CUDA, OpenCL
 - **Analytické modelovanie paralelných programov, analýza výkonnosti, ladenie**
 - Vzory pre paralelné programovanie
 - Iné paralelné programovacie modely (napr. Cilk++, **Map-reduce**, Unified Parallel C, Thread Building Blocks, High Performance Fortran, Erlang, BOINC, CellBE, ...)
-
- Co si ja pamatam z predmetu a cviceni ? matematicke dokazy, thearady a akademicke prilklady o nenazranych filozofoch
 - Co som dneska mohol mat lepsie ?
 - stiahovanie obrazkov k filmom za 3 minuty miesto 30 minut a nelocknute amazon konto.....
 - Maticky aparat aby som overil ze chalani nemaju v kode bug
 - Pochopenie ze synchronizacie sa netykaju teln multithreadoveho programovania (to je len jedna cirkvi, su aj ine)
 - Map Reduce a ine “moderne paralelizmy”

small is beautiful, divide and conquer,



- The **Unix philosophy** is a set of **cultural** norms and **philosophical** approaches to developing **small** yet **capable** pieces, that can be reused and **repurposed**,
- The Unix philosophy favors **composability** as opposed to **monolithic** design.



- Many small parts – difficult to pick, low visibility need fantasy to combine them,.....

Composition, Reuse

- Sources – grrrrr, git submodules, git subtree
- Libraries – jars, node modules, maven, Nam
- **Executables – YES !!!!!**
- **Services – YES, YES, YES**
- Microservices

Vizualizácia dát

- raz vidiet je lepsie ako 10 krat lustit
- vas nieco taketo ucia ? lebo mna neucili

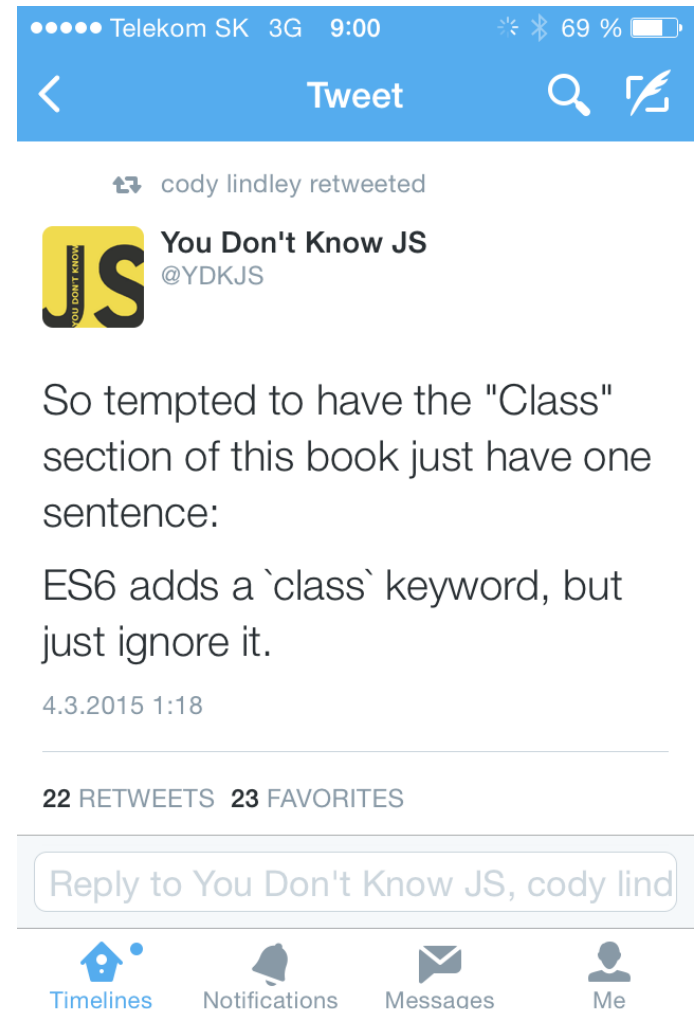


The most harmful coding constructions

- IF a jeho reincarnations (case)
- VAR (global, member, local)
- FOR i=0; i<10;i++
- All this leads to procedural programming
- PROCEDURAL PROGRAMMING – **najzbastardenejší štýl programovania**, nie preto že myšlienka by bola zlá, ale preto, že sa ľahko a veľmi rád kríži z inými nemravnými návykmi.

Deaths of OOP

- OO != Classes (toto mi niekto zabudol povedat ?, či som len nedával pozor ?)
- Aplikovanie OO
 - bez Small is Beatifull je uplne nanič
 - bez stavov, je mi úplne nanič
- OO (Java) v bežnej praxi = extra zbastardenosť procedurálneho programovania zahalená 4ma ďalšími vrstvami (dedenie, overload, override, anotácie)



Grave Stones



OOP is dead (2006)

- <http://kawagner.blogspot.sk/2006/08/oop-is-dead.html>

Object Oriented Programming is Dead (2011)

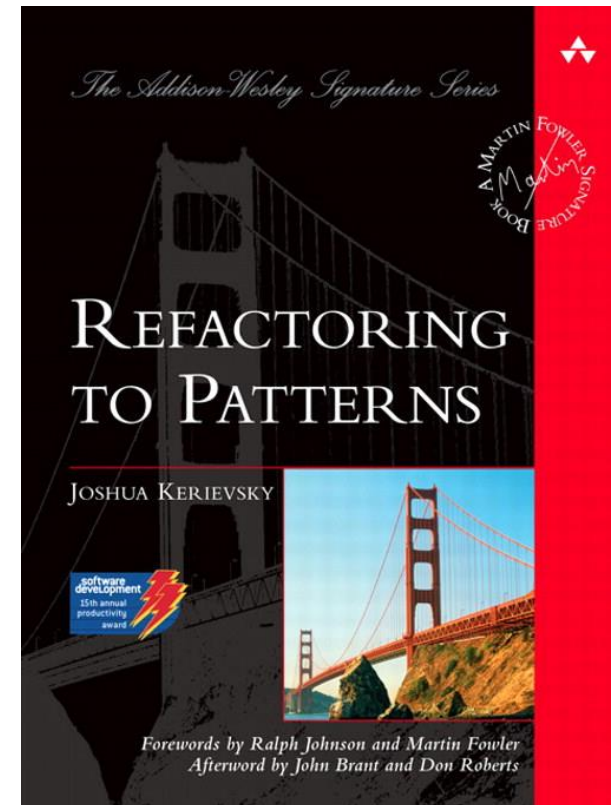
- <http://blogs.msdn.com/b/alfredth/archive/2011/03/22/object-oriented-programming-is-dead.aspx>

The repeated deaths of OOP (2015)

- <http://loup-vaillant.fr/articles/deaths-of-oop>

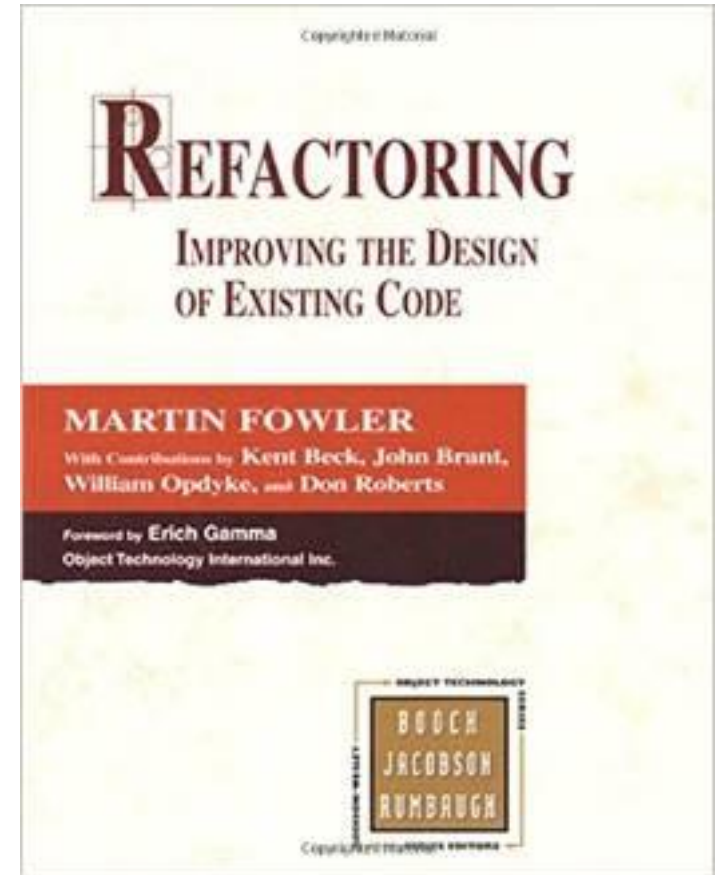
~~Design~~ Patterns - Refactoring

- Refactoring **to** Patterns
- I began **refactoring to patterns**, instead of using them for up-front design or introducing them too early into my code. [J.Kerievsky]
- Patterns are a cornerstone of **object-oriented design**, while **test-first** programming and **merciless refactoring** are cornerstones of **evolutionary design**.



~~Design~~ Patterns - Refactoring

- Refactoring: **Improving** the Design of **Existing Code**
- The process of changing a software system so that it **does not alter the external behavior** of the code yet improves its internal structure [Fowler]



~~Patterns~~ - Refactoring

- “Design” Patterns: too abstract, too object oriented, hard too imagine how to use them ?
- Refactorings Catalogue: easy to understand, tou can start using it immediately on your code

Catalog of Refactorings



Martin Fowler
10 December 2013



🏠 / Refactoring

Refactoring

~~Patterns~~ - Refactoring

Decompose Conditional

You have a complicated conditional (if-then-else) statement.

Extract methods from the condition, then part, and else parts.

```
if (date.before (SUMMER_START) || date.after(SUMMER_END))  
    charge = quantity * _winterRate + _winterServiceCharge;  
else charge = quantity * _summerRate;
```



```
if (notSummer(date))  
    charge = winterCharge(quantity);  
else charge = summerCharge (quantity);
```

Refactoring, Patterns, Tests, Coding Rules all together

- Templates of “good” code snippets
 - don’t let developers think how to do it (to google, to reinvent wheel, to invent **square wheels**)
 - do not use **ad-hoc** “stack overflow solutions”
- Detectors of bad (smells, for anti patterns)
 - I cannot draw but I can **tell nice from ugly**
- **Codereviews** - for good and bad
- **Tests** - any tests, BDT, UT, PbT, CCT - to describe **expected and unexpected behaviour** and to allow refactorings
- **Will and courage**
- Refactoring Tools
- Refactorings Catalogue, Coding Styles and Coding Rules (explain “**why**” a how for templates and bad detectors)

Your own toolbox

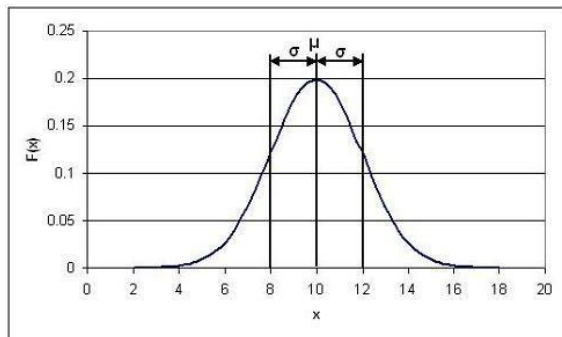
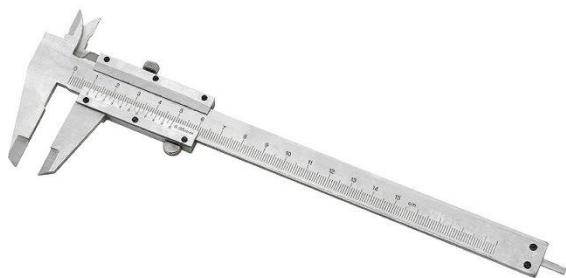


- Never Complete
- Never Up To Date
- Always organized
- Always clean
- Costs Huge Money
- Multi Vendor

Poznate ich ?

- Martin Folwer ?
- Linus Torvalds ?
- Rod Johnson ?
- John Resig ?
- Steve Souders ?
- Lea Verou ?
-

merania a štatistika



- Pochopíte keď si odmeriate guľicky v prednom ložisku a majú 6mm, odídete do obchodu kúpite ich 2 baliky, prídete domov a oni sa tam nevojdú. Odmeriate ešte raz a zrazu majú 5mm nie 6.
- jeden raz nestačí....
- zapisovať výsledky
- porovnávať výsledky
- analyzovať trendy
- vyvodzovať závery
- text + git + git diff + statistics + charts
- Keď pocujem slovo priemer som alergický (aj o číslach aj o ľuďoch)
- median, mean, average, histogram

The Secret of Fast Programming: Stop Thinking

- When I talk to developers about code complexity, they often say that they **want to** write **simple code**, but deadline pressure or underlying issues mean that they just **don't have the time** or **knowledge** necessary to both complete the task and refine it to simplicity.
- Instead of saying “This deadline prevents me from writing simple code,” one could equally say, “**I am not a fast-enough programmer to make this simple.**” That is, the **faster you are as a programmer, the less your code quality has to be affected by deadlines.**

“P” Paradigmy, Right Tool For a Job



- Object Oriented
- Functional
- Declarative
- Rules/Logical
- Parallel
- Distributed
- ...
- ...
- ...

- Preto v IT kodujeme úplne nevhodným OO stylom niečo čo sa da na 3 riadky napísať v bashi ?
- Preto sa takto správame v IT ? Asi by nás nenapadlo kopať kanál hrablami. Nemáme krompáč/lopatu no asi si pojdeme POZICAŤ alebo ak som bohatý tak kúpiť, ak som LENIVÝ/INAK ZANEPRÁZDENÝ tak spraviť niekomu inému.
- Čo sa stane keď budeme srobu zatahovať kombináckami ? (damage, expensive consequences)
- Čo robíme dneska:
 - Nakupujem si drahé jednocelové náradie
 - Čítam si o tom ako sa to má robiť
 - Az keď prejdem cez štádium 1,2 a ešte stále mám čas, vykonám danú misiu
 - Ak nemám čas, mám doma aspoň náradie ;-)))
- vidíte nejakú podobnosť/rozdiel čo chce/mala by dosiahnuť škola ?
- **LENIVOSŤ => POKROK**
- **ZNALOSTI + LENIVOSŤ = POKROK, LEPSÍ A RYCHLEJŠÍ VÝSLEDOK, ...**

Questionare

- NECHODIL do materskej skoly
 - Ma na notebooku Windows ?
 - Nema notebook/komp (zbytocna otazka)
 - NEBYVA na intrakoch ?
 - Aky OpenSource SW pouzivate ?
 - Pise prace do skoly v MS Worde ?
 - IT robi leho ho “BAVI” ?
 - NEVIE preco robi IT ?
 - Ma Facebook konto ? (zbytocna otazka)
 - Vie spajkovat ?
 - Ma twitter konto ?
 - Ma psa/macku/morca/frajerku ?
 - Ma vlastny blog ?
 - Nevie dobre po anglicky ?
 - Bajkuje/lyzuje/
 - Nechodi do krcmy
 - Nema github konto ?
 - Nerad kresli, nevie kreslit (stvorcek, trojuholnik, ciarka) ?
 - ...
-
- C

Preco su ITaci taky nezodpovedni



- a vy, vaši kolegovia,
zákazníci a okolie takto
.....

- Keby ste boli
chemici, ulice by
vyzerali takto.....



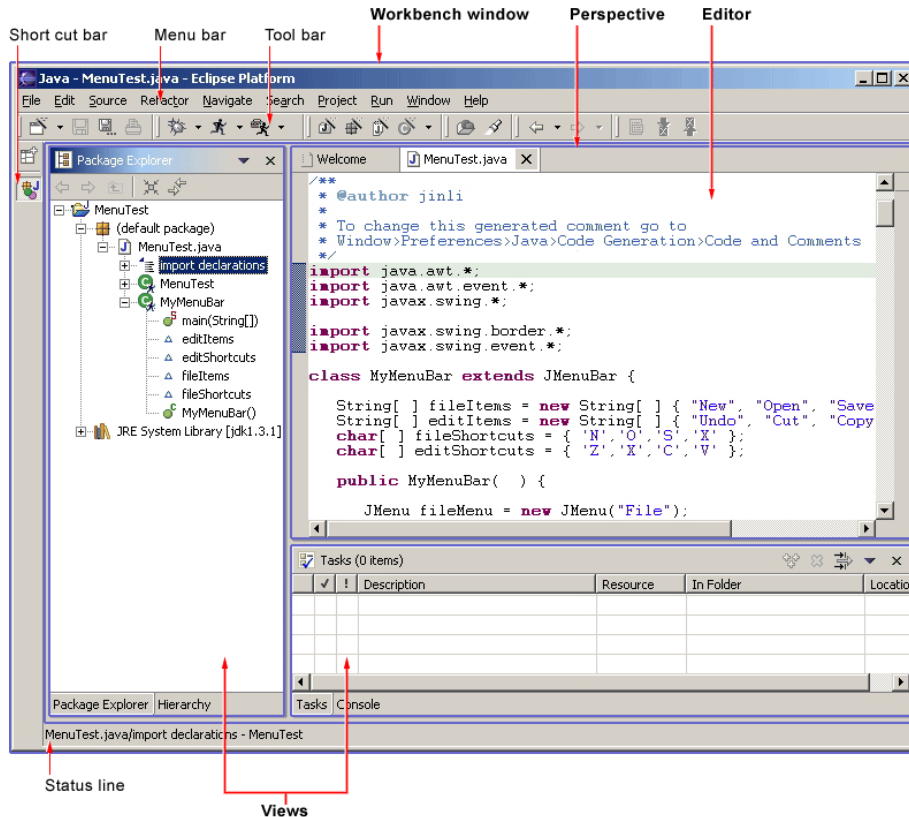
Slow Down

What I Wish I Knew When Starting Out as a Software Developer: Slow the Fuck Down



- <http://blog.salsitasoft.com/what-i-wish-i-knew-when-starting-out-as-a-software-developer-slow-the-fuck-down/>

UI is harmfull

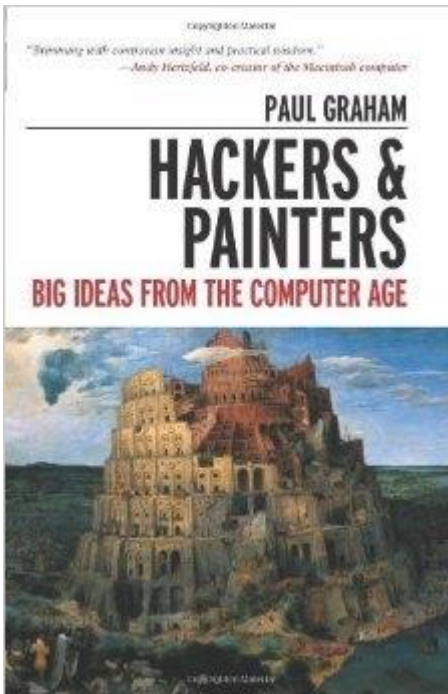


- UI

- REPL

- Code

Hackers and Painters



Sometimes what the hackers do is called "software engineering," but this term is just as misleading. Good software designers are no more engineers than architects are. The border between architecture and engineering is not sharply defined, but it's there. It falls between what and how: architects decide what to do, and engineers figure out how to do it.

<http://www.paulgraham.com/hp.html>

What next

- Ak to niekoho zaujalo:
 - Spravte git gub project z nazvom next
 - kazda tema bude mat issue
 - kazdy z vas bude za dane issue votovat +1 a -1 moze napisat komentare k upresneniu co si predstavuje pod danou temou
 - top 1 issue pripravim a odprednasam (u nas, u vas podla toho kolko ludi sa prihlasi na dany FaceBook Event, na FB stranke grupe co si na to zriadite ;-)

Q

- 1. Co ste sa v skole ucili uplne zbytocne
- 2. Co ste sa ucili v skole a bolo ok (dalo vam nieco do praxe)
- 3. Co sa v skole neucilo a malo by sa (lebo vam to dneska chyba v praci)

A1

- 1. Dejiny dizajnu - zbytocne, ale na druhej strane to bol oddych. Jazyk Z a formalna specifikacia softweru - to uz ani neviem o com bolo. A akekolvek teoreticke omacky bez praktickych priblizeni(JAD,CASE, RAD)
- 2. Java programovanie (az to s marcusom), operacne systemy (pipe, file deskriptori a pod.), timovy projekt - tam som sa prvý krát dostal k version control (SVN) a inemu ide ako Eclipse (NetBeans)
- 3. Rozmyslanie pri programovaní (dizajn kodu), ďalší programovací jazyk napr JavaScript/perl na pochopenie plusov a minusov. Version control v spojení s release managementom, a cokolvek z enterprise architektury a frameworkov. A cokolvek z dependency management (ant, maven, package).

A2

- 1. matlab, formalne jazyky/prekladace (ale zase toto bolo zaujimane), niektore veci boli take zbytocne ze si ani nespomeniem...
 - aha uz viem - vsetko co malo v nazve "Enterprise" (ESB)
- 2. Java, HTTP stack, refactoring, architektura pc, XPath, UML
- 3. version control (ak sa pouzivalo tak len ako archivator), vyuzivanie specifikacii pri vyskume ("lebo stack-overflow" sefovi nestaci),
 - dizajn API (az ked moj kod niekto iny pouzije, zacnu davat zmysel koncepty typu enkapsulacia)

A3

- 1. pisat eseje, metodiky a podobne dokumenty o veciach ako planovanie, odhady,... bola to iba zbierka okopirovaných materialov, lebo clovek nemal vlastne skusenosti, aby tam nieco pridal
- 2. predmety, kde sa robili zadania, hlavne programatorske (najviac timak), ale aj analyzy
- 3. zadania, kde by sme museli robit s nejakymi toolmi, externymi libkami, aby sme si zvykli robit s cudzimi vecami, githubom,...

A4

- 1)
 - teoreticke predmety typu: Manažment v softvérovom inžinierstve, Princípy softvérového inžinierstva, Princípy informačných systémov
- 2)
 - urcite timovy projekt
 - toto dalo do praxe najviac - pouzivanie source controlu, rozdelenie roboty,
 - ale aj pouzitie novsich technologii, co je asi specificke per konkretny projekt
- 3)
 - vela veci bolo spomenute v ramci nejake teorie, ale nedali nam realnu predstavu,
 - ako to vyuzit v praxi, napr.:
 - - design API / patterny
 - - modularita
 - - pouzivanie source control
 - - existencia specifikacii (nepouzivat w3schools :))

A5

- 1. Princípy informačných systémov
- 2. Dátové štruktúry a algoritmy, Umelá inteligencia, VPPJ
- 3. Telesna :) Inak vyvoj web aplikácii, vyvoj viacvrstvových aplikácii (prakticky nie teoreticky), source control.

A6

- 1. Viacere predmety mali zbytočnosti, komplexne ale napr. Pravdepodobnosť a statistika - nič nové som sa tam oproti strednej nenaucil; Databázy + Pokročile DB technológie - nulová pridaná hodnota oproti znalostiam ktoré by každý všeobecne mal mať.
- 2. OSka - Unix, CShell, Bash; Timový projekt - Kolaborácia, GIT; IČP - hlavne dizajnovanie UI; OOANS - UML a BPMN modelovanie. Vybral som predmety pre mňa s naj pomerom odprednasane/zapamatane.
- 3. Nepostrehol som že by bol predmet kde by som sa mohol naučiť základy a princípy webových technológií, okrem HTML niečo ako základy PHP, JS, Ajax, alebo služby, napr. REST, všetko som kvôli práci dohánal samostudiom.

A7

- 1) Základy účtovníctva, telesná (okrem 10 dňového lyžiarskeho zájazdu)
- 2) Biochemické procesy, chemické inžinierstvo
- 3) Skromnosť, schopnosť popísať sa pod svoju prácu a názor

A8

- 1. špecifikačný jazyk Z (10 rokov out of date už vtedy keď sme sa ho učili)
- 2. všeobecné kódovacie predmety, algoritmy, C, java, OS-ka, dokonca aj Aspecty (ak človek pocúval len 10%)
- 3. riešenie problémov v komunite (nie tímový projekt, ale napr. stack overflow alebo RoseIndia :D)

A9,10,11....