The four pillars of Long Lasting Software

François Lorioux,

a guy with f… (strong) french accent

About me

- 30 years experience in software
- Software editors Oil E&P and Bank
- Currently Team leader and Architect

Why this session?

Experience with Long Lasting Software

Key ideas for long term success

Long Lasting Software?

Software successfully delivered

Software with long active maintenance

Not zombie software

Keys for Long Lasting Software

- Knowing software evolution “laws”

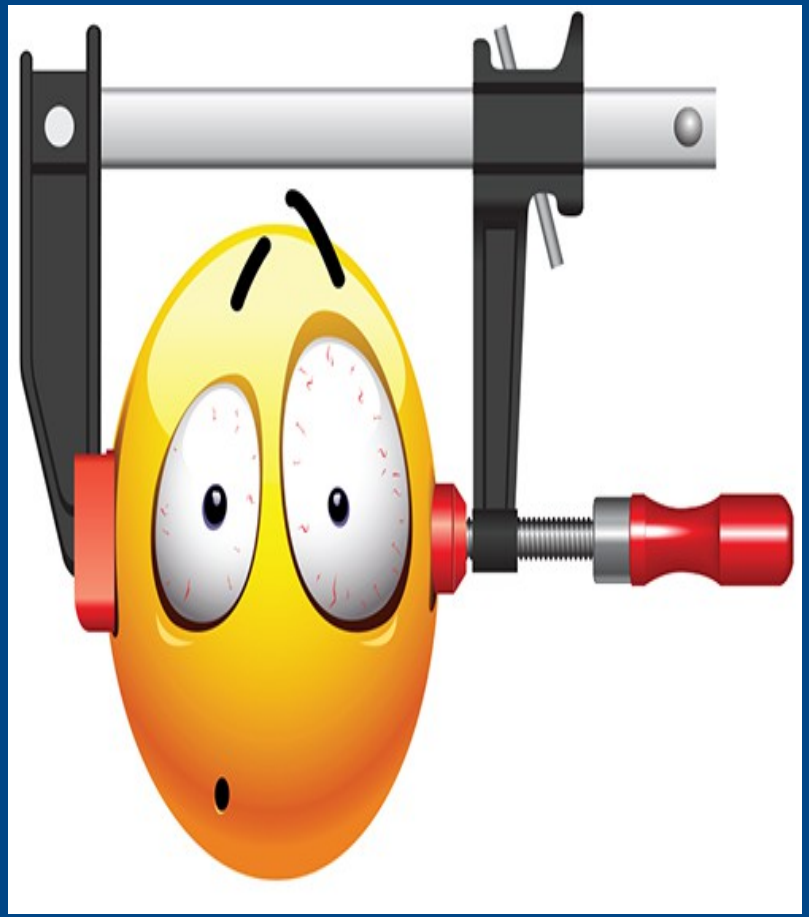
- Applying the four pillars

Source: https://allomarcel.com/blog/artisanat/la-cle-de-la-securite.html

Lehman's “laws”

- Changes are needed
  - Improvements and added features
  - Technical environment
- Complexity increases (technical debt)
  - Quality decreases
  - Cost of change increases

Software is under pressure!



Exogenous factors

- User requirements
- Technology changes

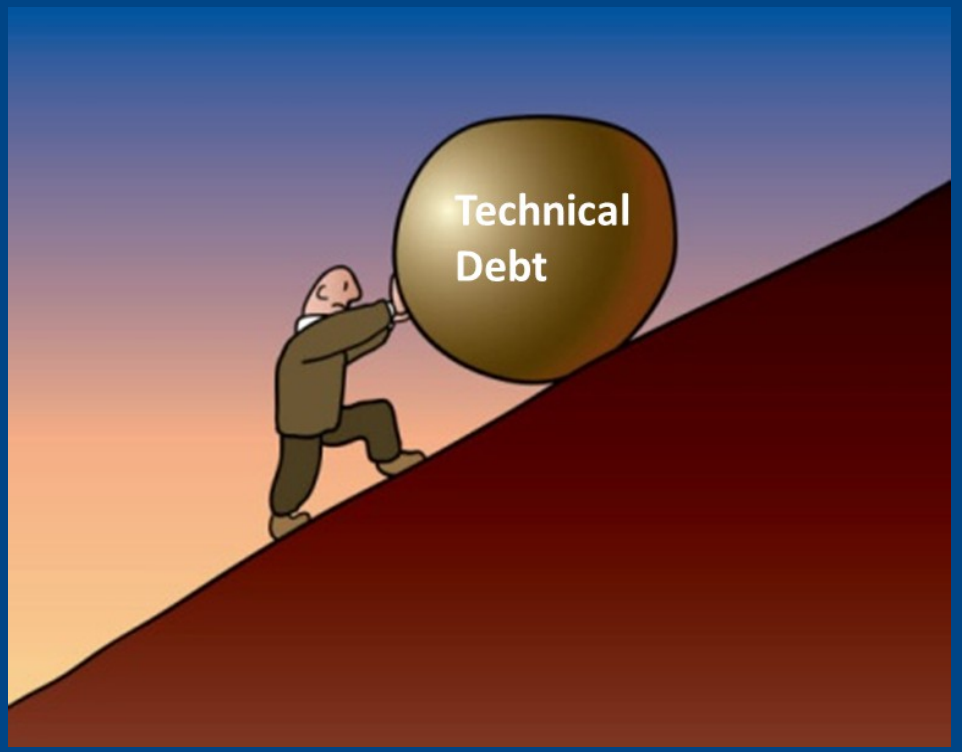
Endogenous factors

- Design/coding practices
- Schedule

source: http://www.symbols-n-emoticons.com/2013/09/emoticon-under-pressure.html

Result is technical debt

Technical Debt
is the cost
of missing
quality



source: http://blog.castsoftware.com/
the-financial-implications-of-technical-debt/

How to keep your software healthy?

source: http://fitlife.tv/5-minutes-and-5-dollars-to-a-happier-healthier-you_original/

Healthy software: how?

Genetic: software DNA

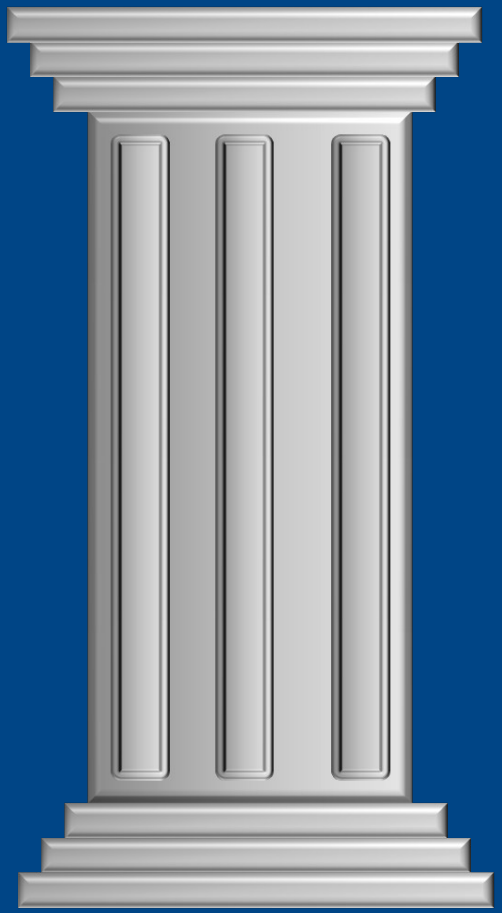
Things to do

- Continuous improvement

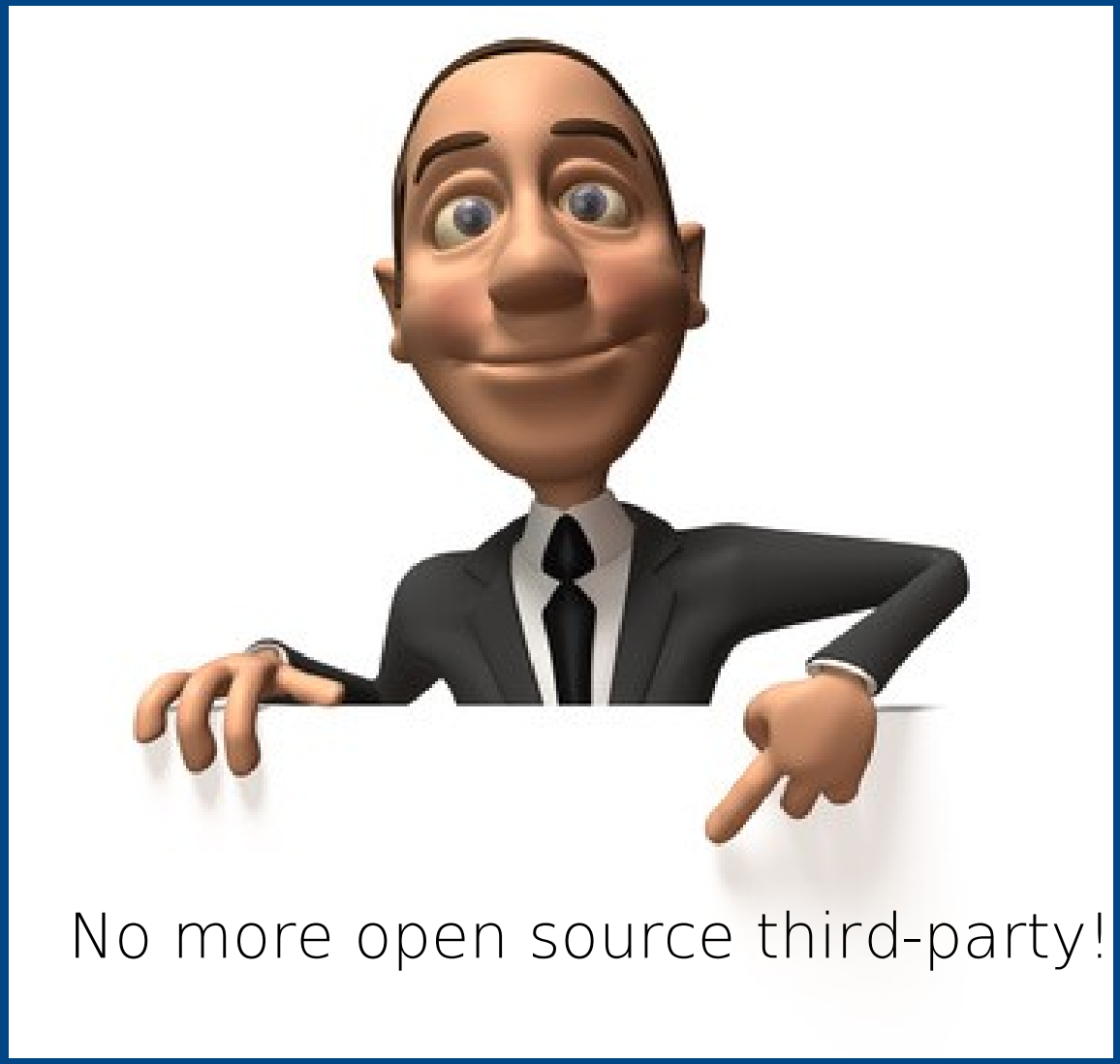
Things not to do

- Software killer practices

1- Good Practices

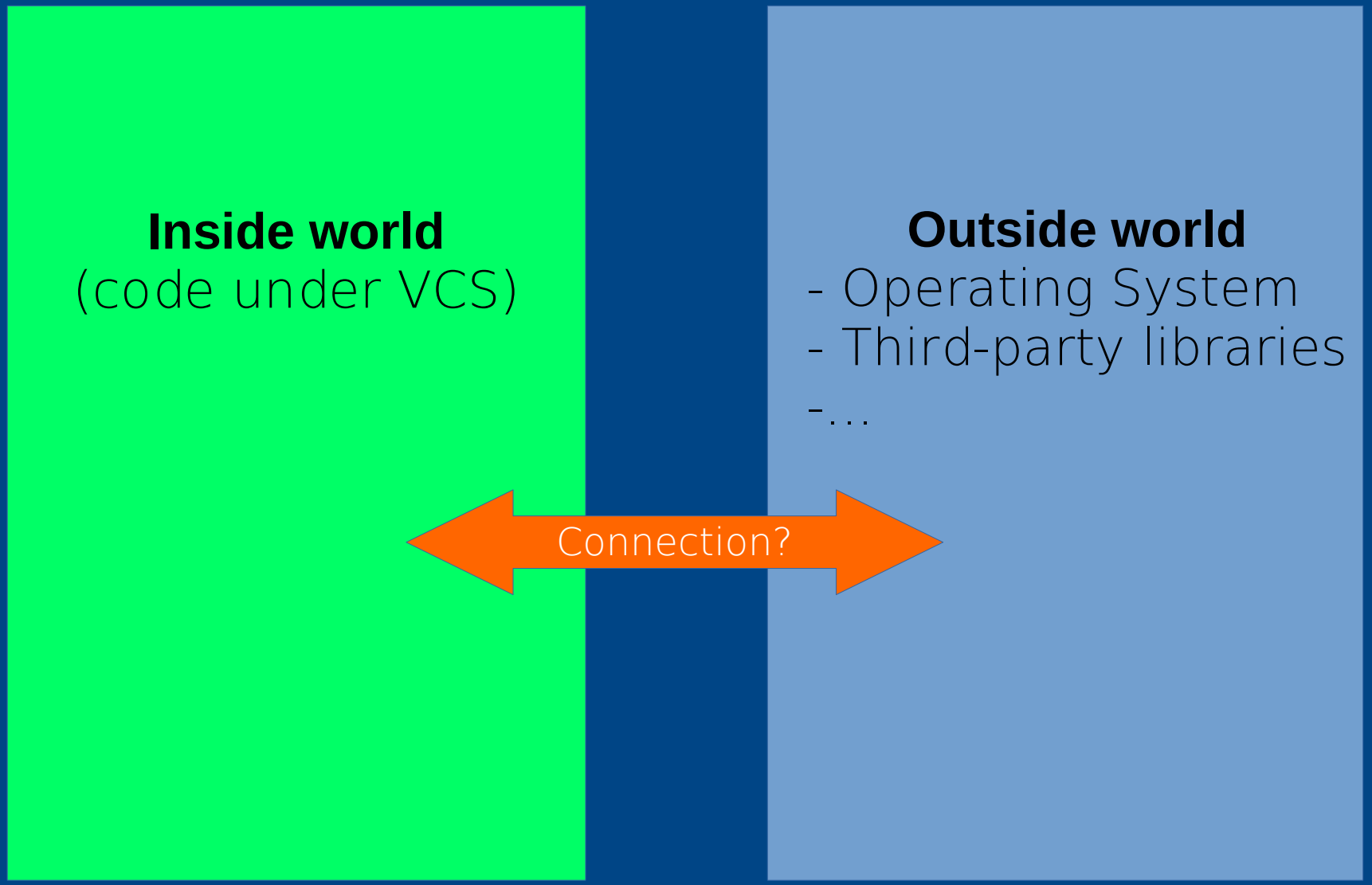


Let's imagine...

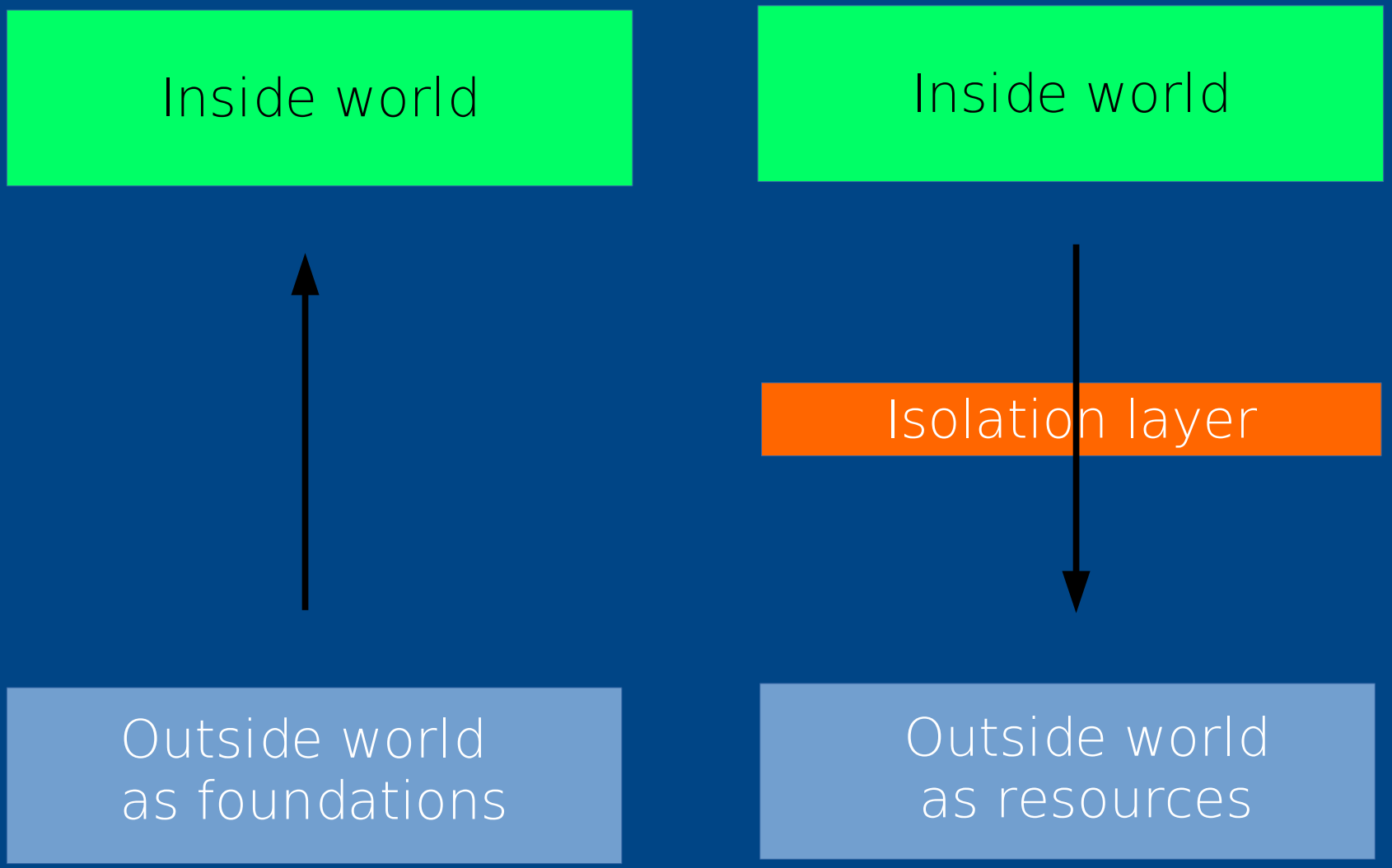


source: http://www.fitnetmanager.com/le-blog/le-role-du-manager-en-synthese/

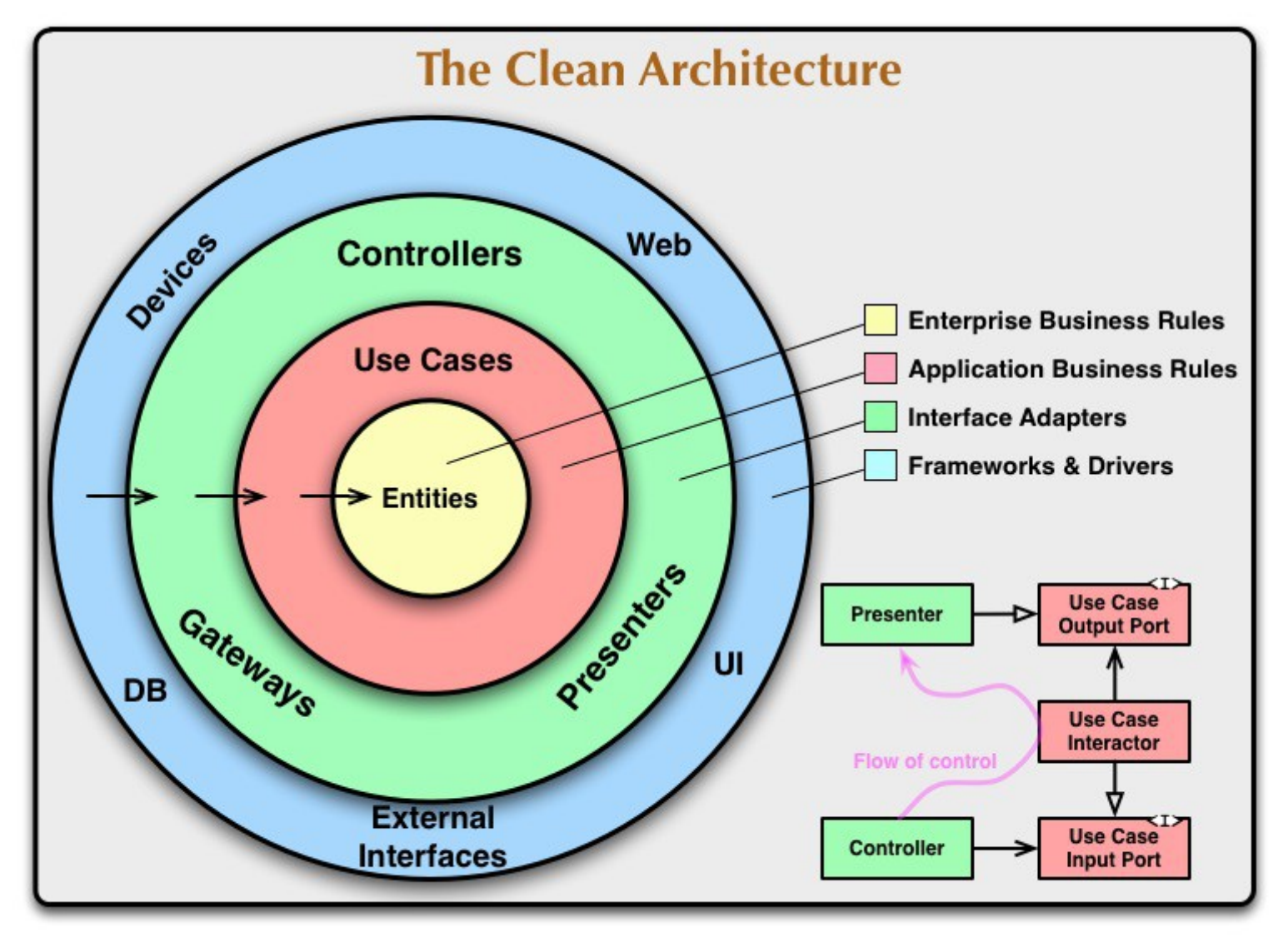
Software boundaries?



Bottom/Up versus Top/Down



Solution?

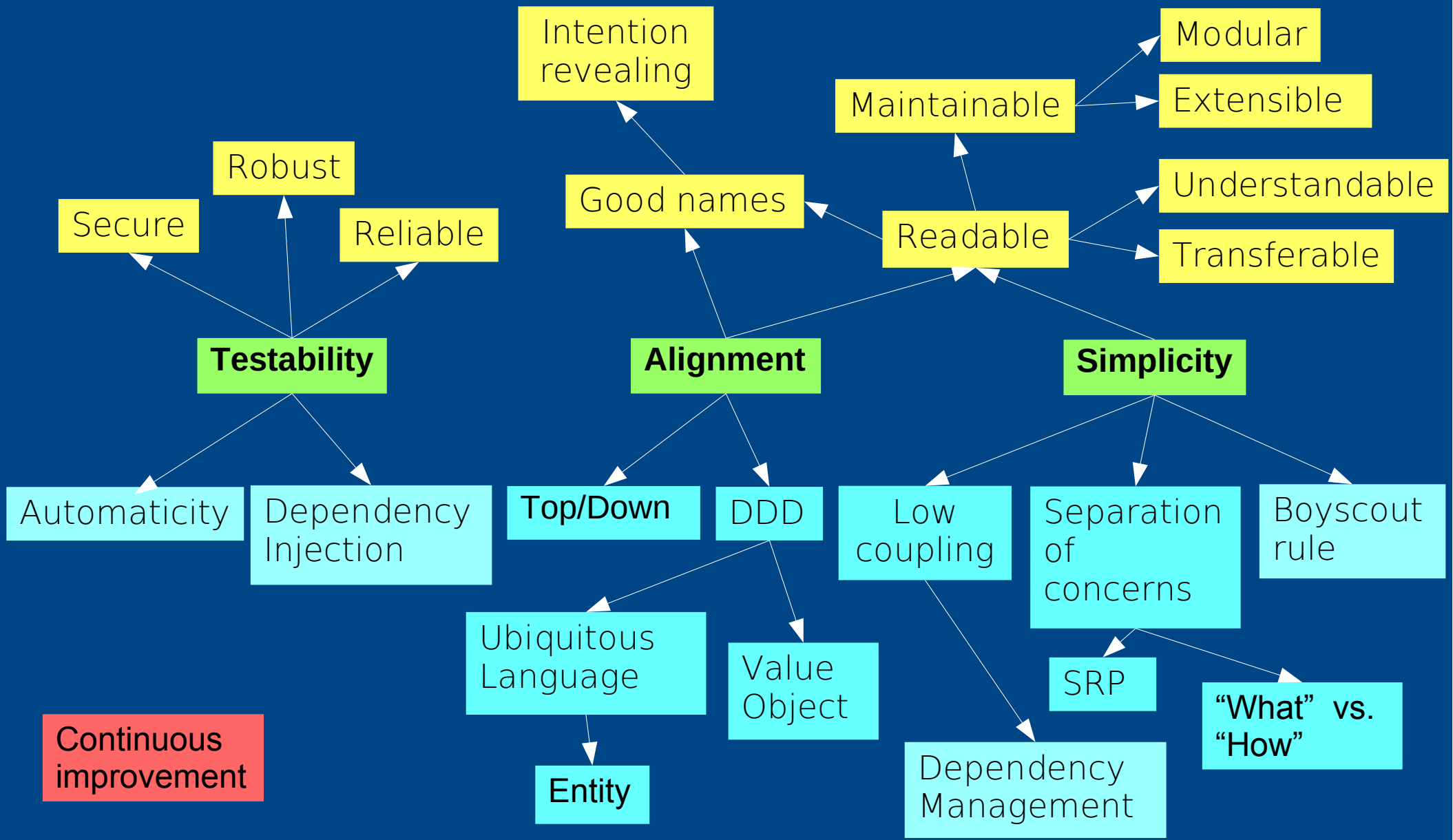


source: https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

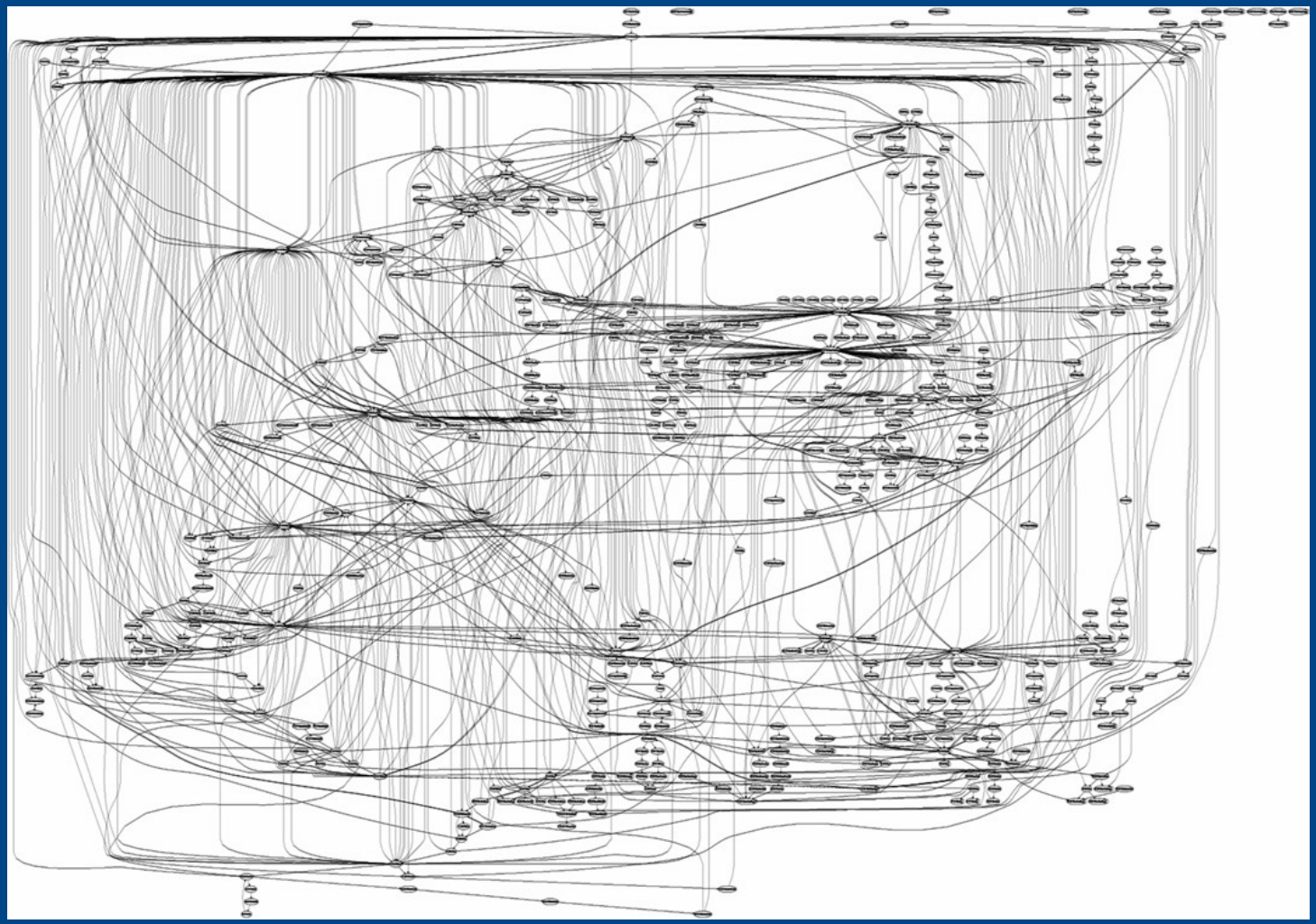
Design…

- There is no ideal design
- Design must be only good enough
- Emergent design

Clean code...


Intention revealing
Modular
Maintainable
Extensible
Robust
Good names
Understandable
Secure
Reliable
Readable
Transferable
Testability
Alignment
Simplicity
Automaticity
Dependency Injection
Top/Down
DDD
Low coupling
Separation of concerns
Boyscout rule
Ubiquitous Language
Value Object
SRP
"What" vs. "How"
Entity
Dependency Management
Continuous improvement

Software killer: dependencies...



source: http://www.saalonmuyo.com/wp-content/uploads/2008/05/syscalliis.jpg

Dependencies...

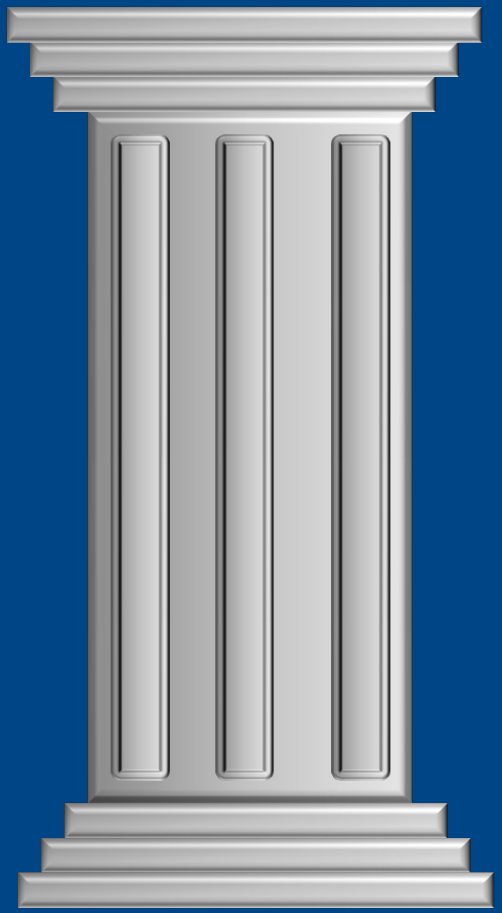
Dependencies on third-party

Dependency inter components

Easy to kill a software!

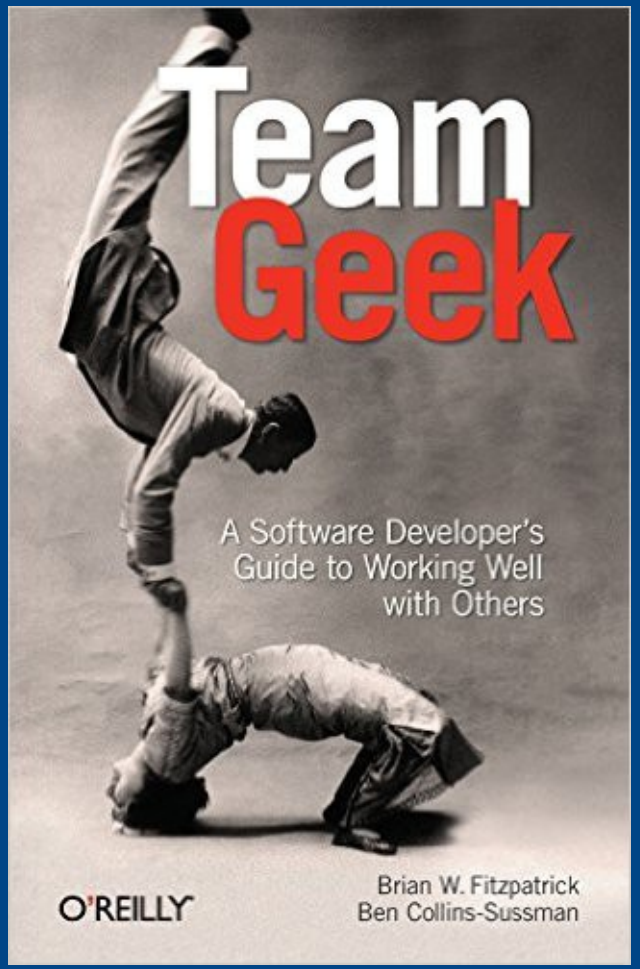
public MyFrame extends JFrame

2- Good Team



Communication inside the team

- Humility
- Respect
- Trust

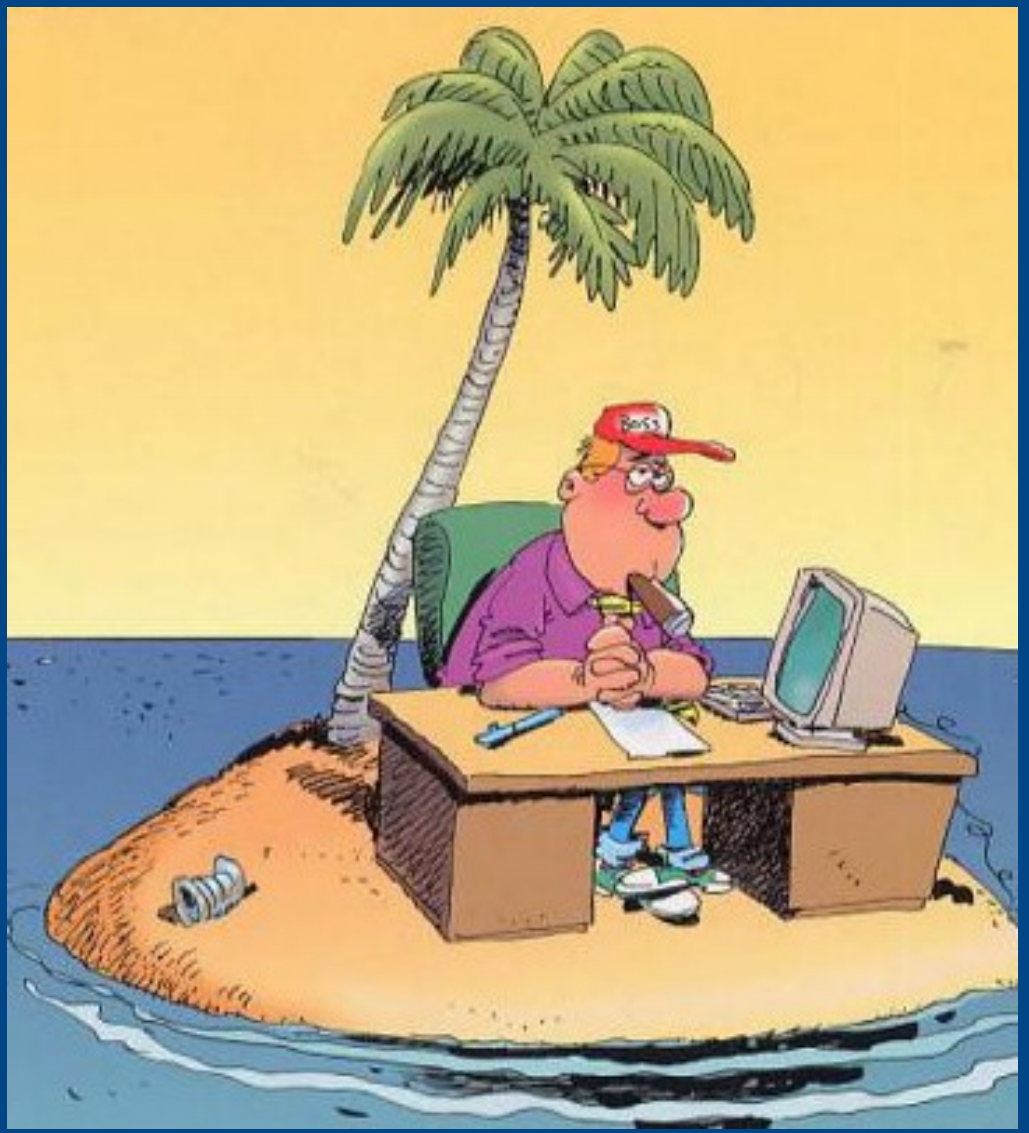


Willing to learn?

Develop the culture

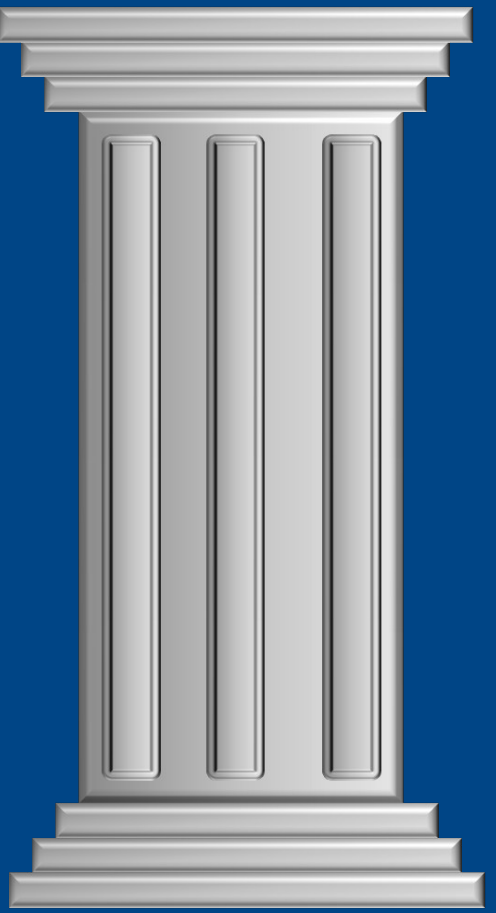
- Design Patterns (Command, Factory,...)
- Coding/Design Smells (Data Clumps,...
- Acronyms (DRY, YAGNI, SOLID, KISS,...)
- DDD (Entity, Value Object, UL,...
- TDD, Emergent design,
- ...

No knowledge island!



http://www.canalbd.net/img/couvpage/36/9782800136363_cg.jpg

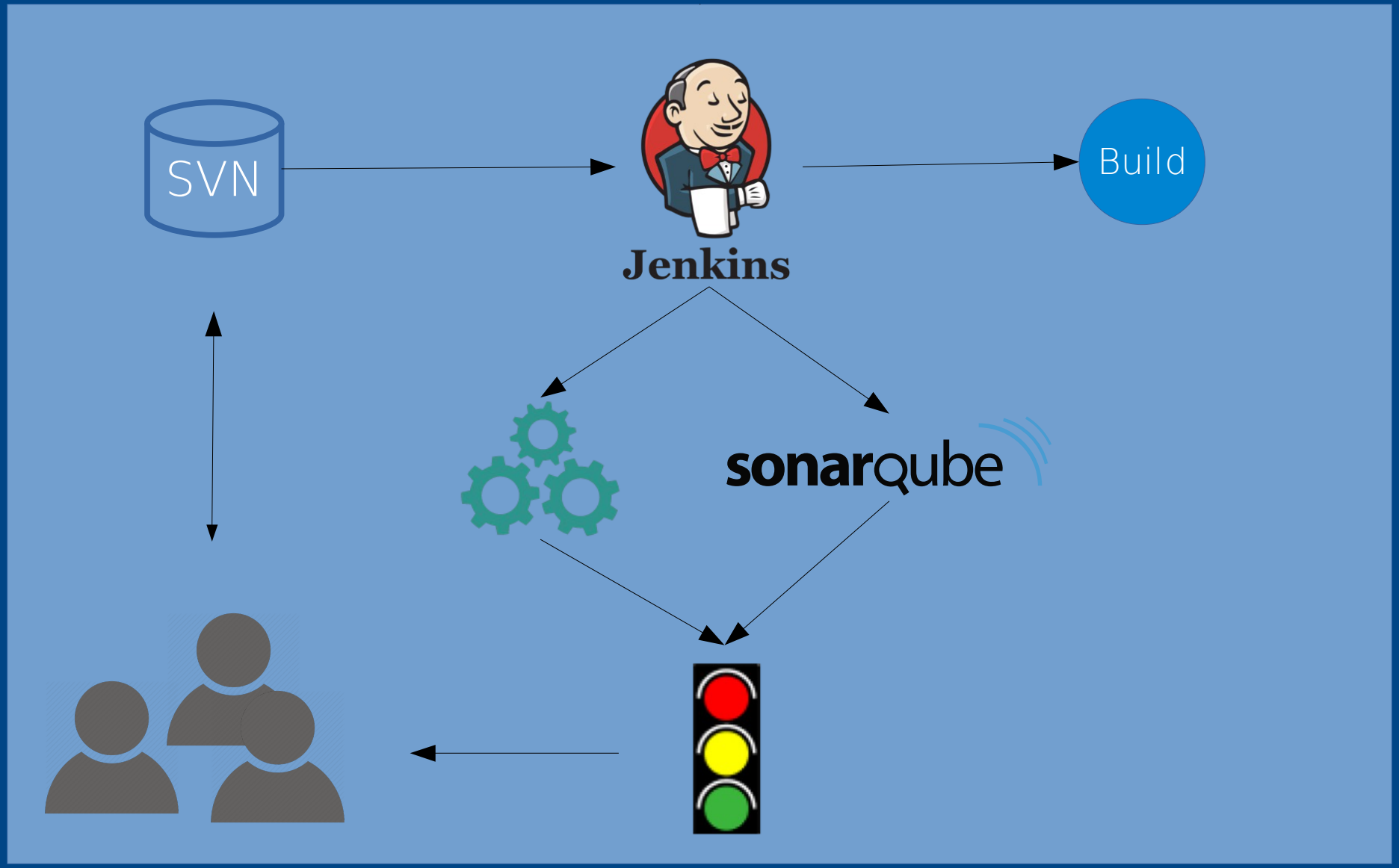
3- Good Tools



Tools: key ideas

- Quality
- Productivity (time is money)
- Automation wherever possible

Software factory…



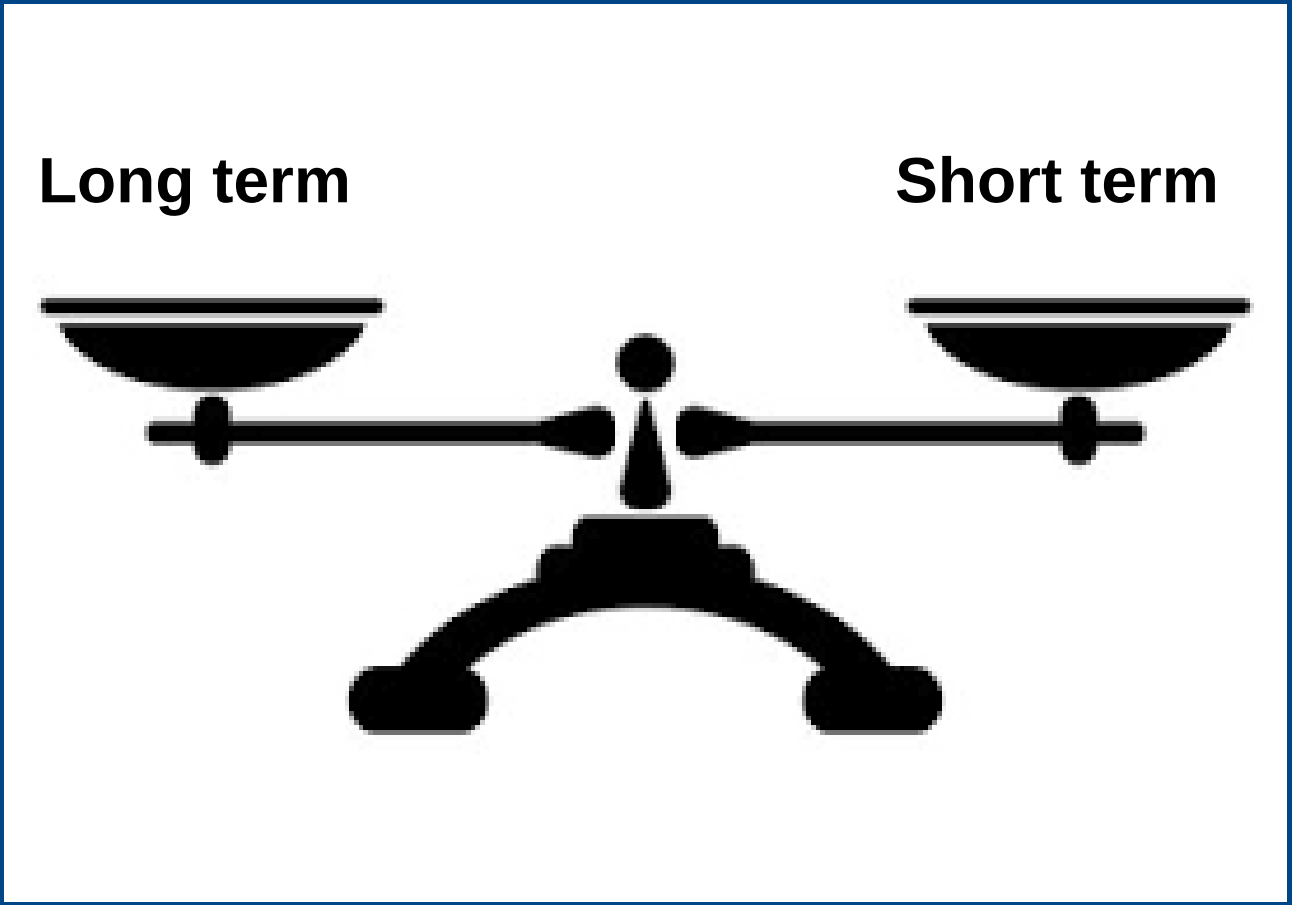
More tools...

Dependencies management
JDepend, STAN, CDA

Performance/Memory analysis
JVisualVM

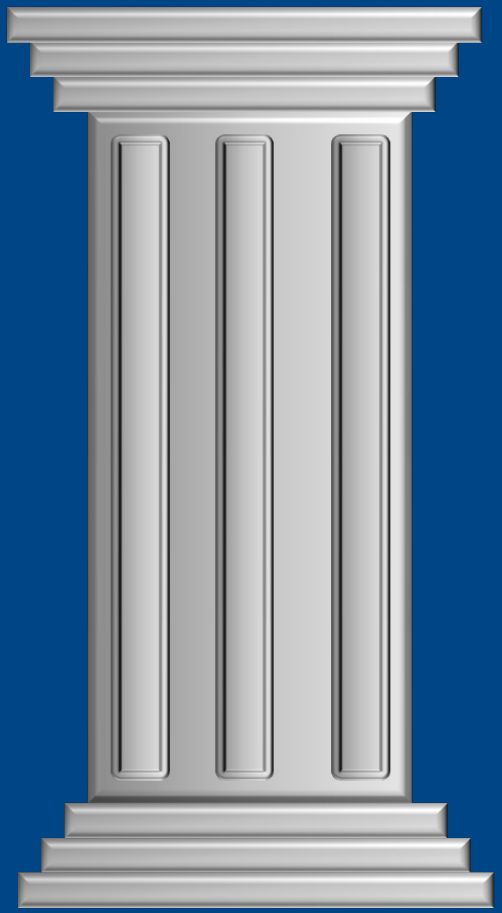
→ Wish list: JRebel...

Tools: productivity trap interface builder case

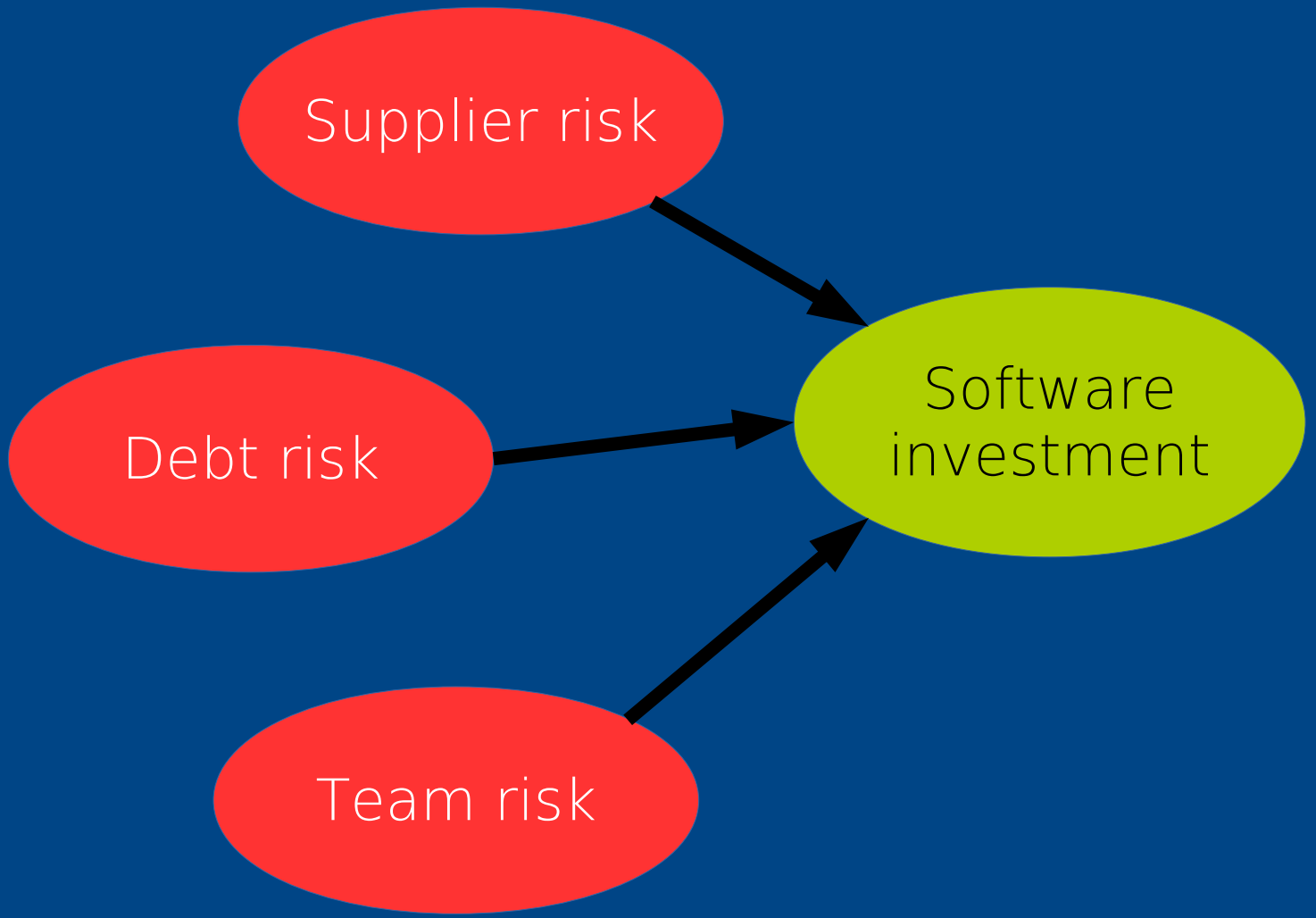


https://fr.fotolia.com/tag/balance

4- Good Organization



Risk management



Everything's ok...

Source: http://3.bp.blogspot.com/-mfAJxfANVOQ/T0MOzCeRfkI/AAAAAAAAAGA/ckzXX0ZzvrE/s1600/grenouiille_boueLentement.jpg

Supplier risk

Know dependencies/suppliers

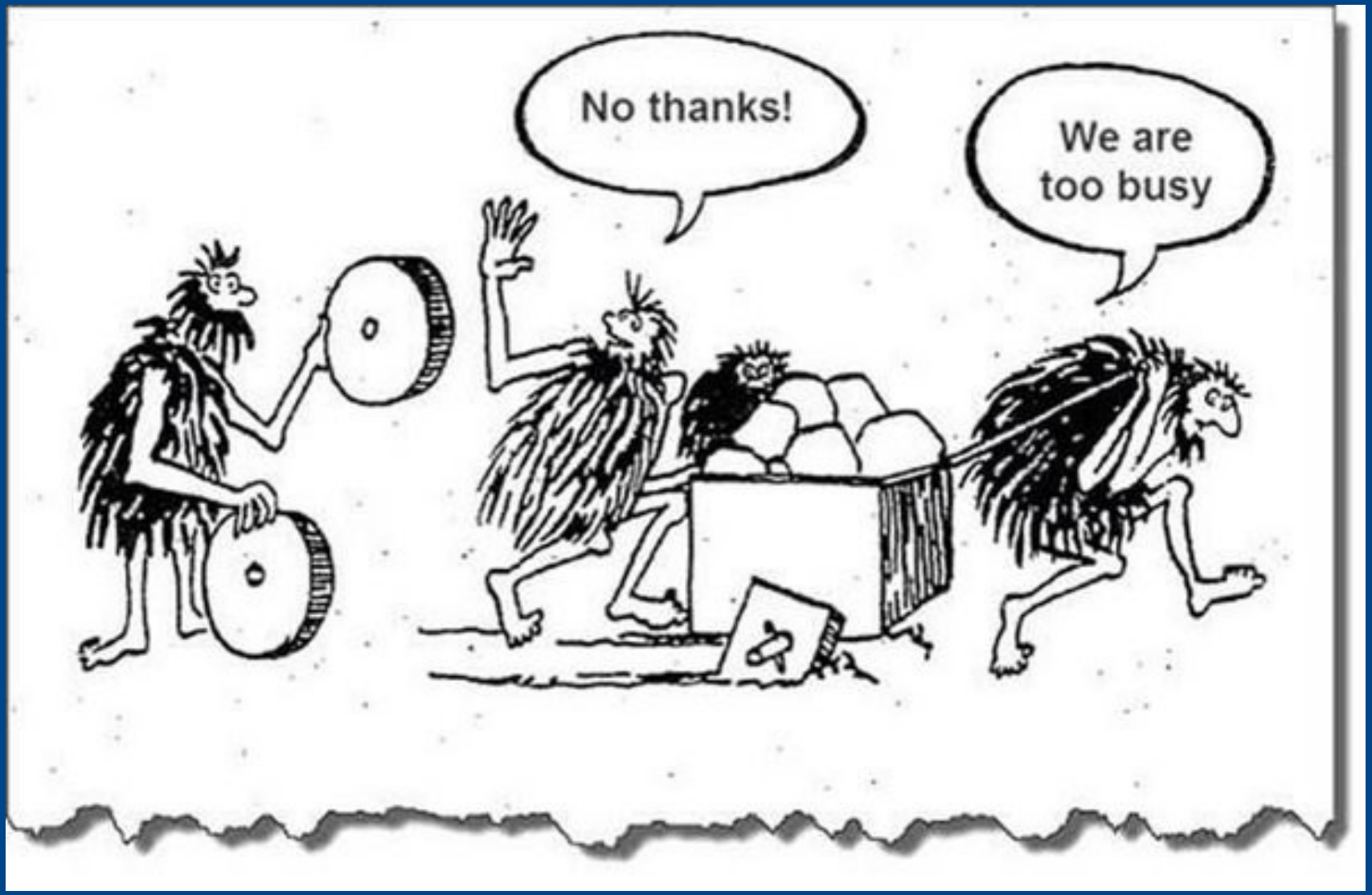
Have contractual clauses

Be aware
- technical solutions exist...
- .. but must be applied early

Protecting the knowledge…

source: http://researchleap.com/building-a-better-national-innovation-system-through-effective-knowledge-sharing-a-case-of-croatia/

Tackling the Technical Debt



source : https://twitter.com/carnage4life

Technical Debt: when?

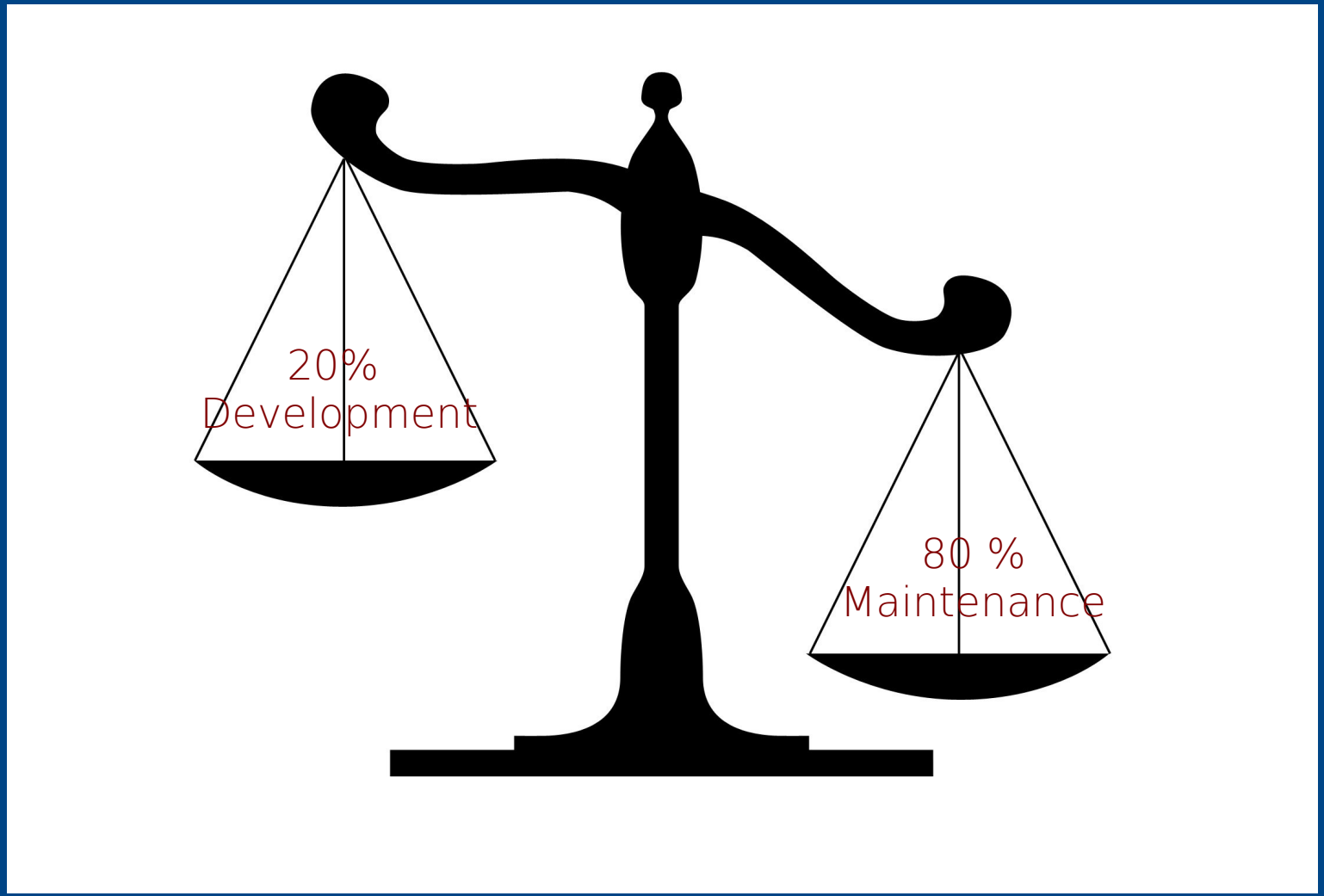
source: http://www.justaddgoodstuff.com/

Thanks !

Any question ?

@fldev
francois.lorioux@free.fr

Software lifecycle



Source: http://i0.wp.com/xn--sport-et-sant-nhb.com/wp-content/uploads/2013/05/scales.jpg

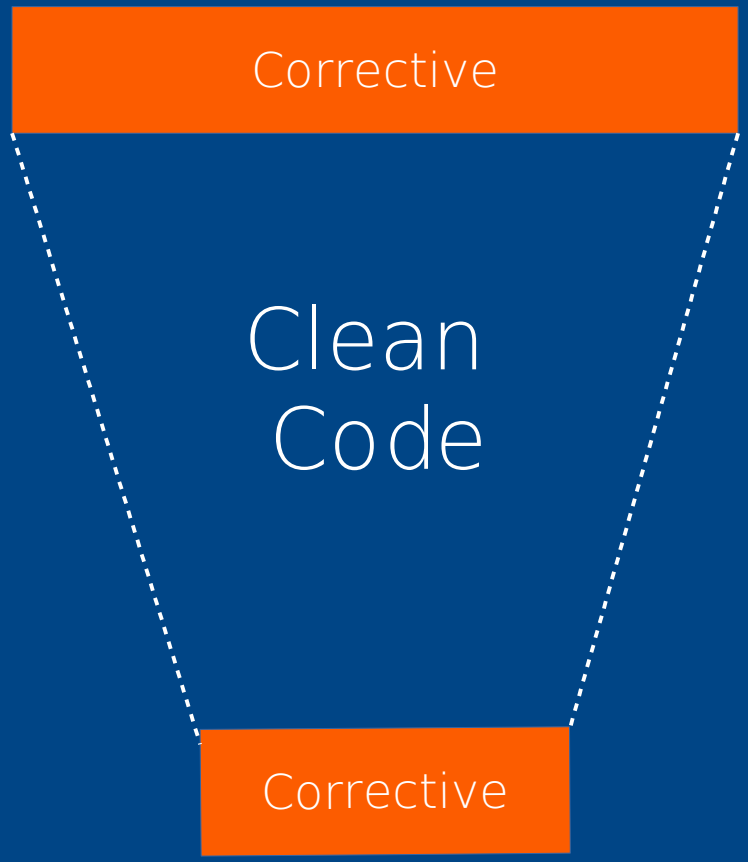
Situation for maintenance

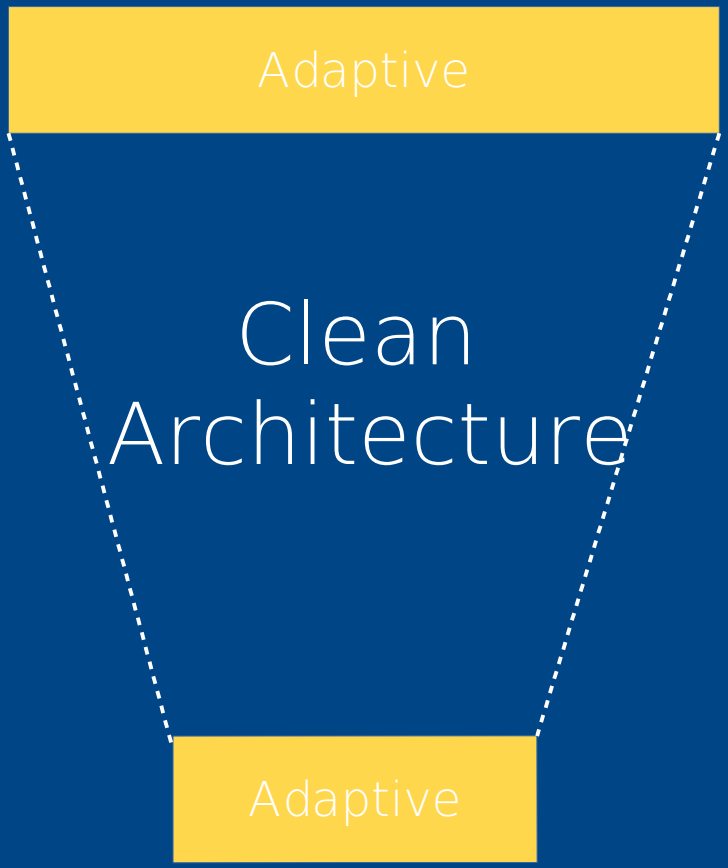
Ideal

Average

Dangerous

The maintenance



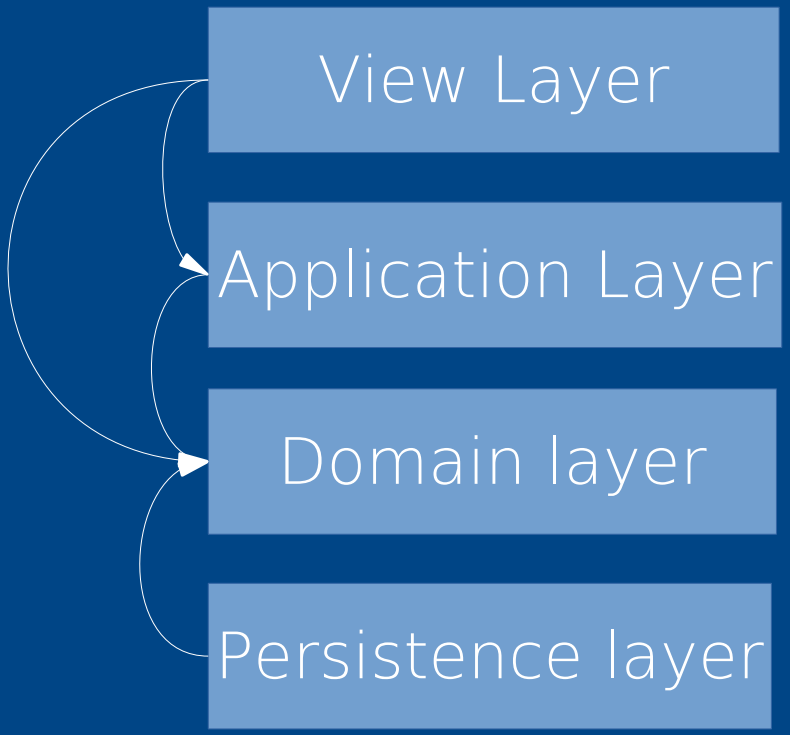


Lehman's “laws” consequences

Need / Maintenance	Corrective	Adaptive	Evolutive
User need	X	X	X
Technical envir.		X	
Complexity	X	X	X

Layered Architecture

Clean Architecture



Domain Driven Design

- Ubiquitous Language
- Entities
- Value Objects
- Anti-corruption layer
- ...

Keys for Long Lasting Software

Know software “laws”

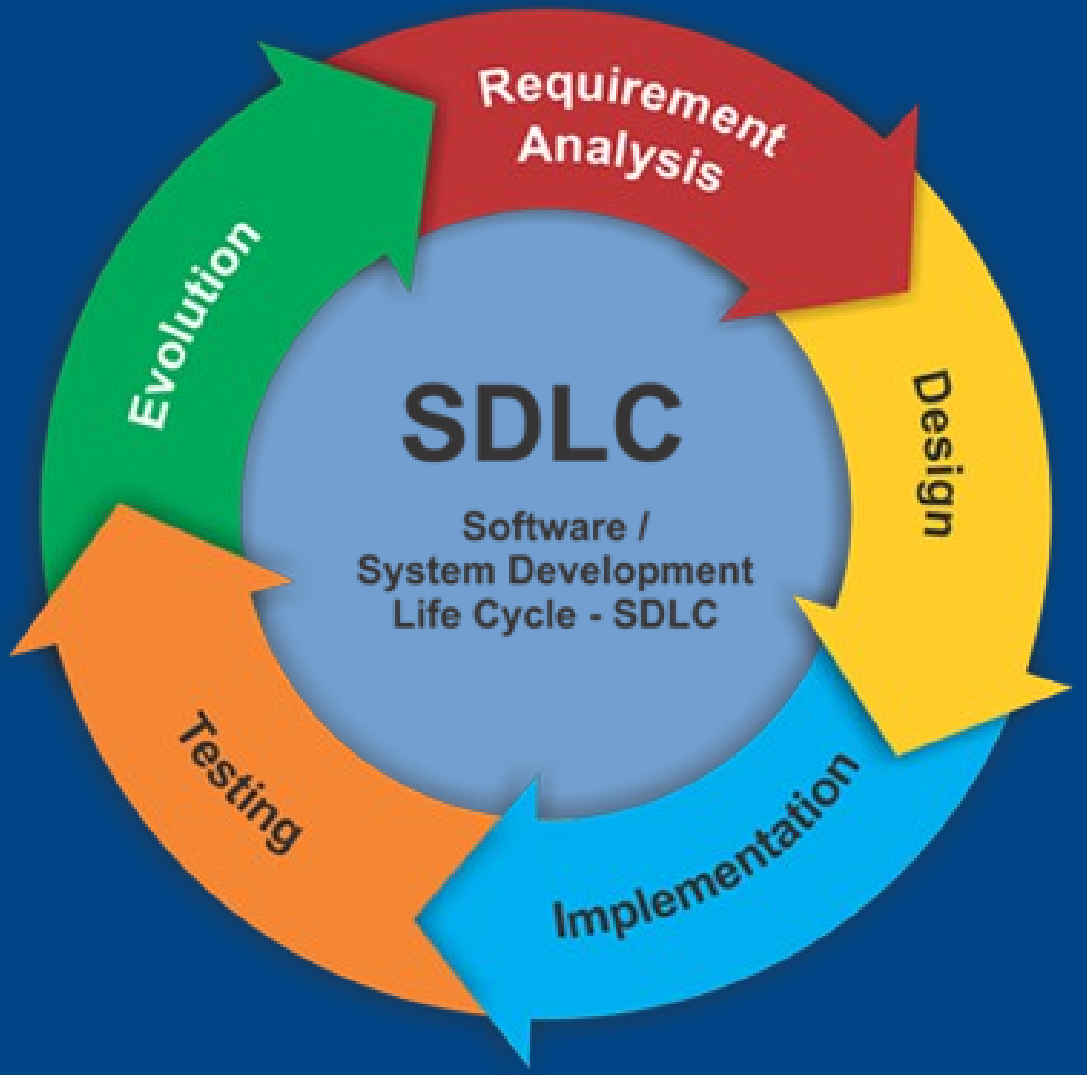
Take good initial decisions

Apply the 4 pillars

1st key: Lehman's “laws”

- Changes are needed
  - Improvements and added features
  - Technical environment
- Complexity increases (technical debt)
  - Quality decreases
  - Cost of change increases

Software lifecycle?



Source: http://technosoftwares.com/wp-content/uploads/2016/09/SDLC_Software_Development_Life_Cycle.jpg