Code That Fits in Your Head

By Mark Seemann

Art or Science ?

Are we a :
- Scientist or Artist ?
- Engineer or Craftsman ?
- Gardener or Chef ?
- Poet or an Architect ?

None of the above
And little of all of the above

Follow heuristics

That can be taught

Rules of thumb

Guidelines

Checklists

An **aid** to **memory**
- Help focus on the hard parts
- Taking your mind off the trivial things

Not to constrain
- Should enable / support / liberate

"This book can help transition from programmer to software engineer."

Checklist for starting a new code base

- [x] Use Git
- [x] Automate the build
- [x] Turn on all error messages
  - Treat warnings as errors
  - 0 tolerance for warnings
  - Linter / static code analysis warnings as errors

Tackling Complexity

"Optimise code for readability."

Sustainability
- Much software lives for decades
- A continual effort to make it evolving

Short-Term Memory
- From 4 to 7 pieces of information
- Our brain can't keep track of all

Value
- Code should produce value
- Some code produces no *immediately measurable* value

Should not be prohibited

Readability

When writing code

Context — Clear infos

When reading code

Context is gone...

Organise your code so that the relevant info is activated

Place related code together

WYSIATI
What You See Is All There Is

> *The core problem that software engineering should solve is that it's so complex that it doesn't fit the human brain.*

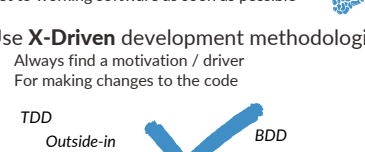
Vertical Slice

"Don't trust yourself to write correct code."

Walking skeleton

Get to working software as soon as possible

Use **X-Driven** development methodologies
- Always find a motivation / driver
- For making changes to the code



Unpolluted Domain Model

Unpolluted by implementation details

Humble Objects
- Objects like "Repository"
- Hard to unit test : depends on a subsystem

Perform smoke test
- Favor automated tests
- Can use cURL for example

Encapsulation

A contract introduces / formalises a level of trust

Transformation Priority Premise
- Use it as a **driver** for changes
- From one working state to another
- Move in small increment
- Driven by tests

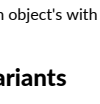
```
{} → nil
nil → constant
...
array → container
...
expression → function
variable → assignment
...
```

Design by contract
- Interact with an object without knowing implementation details
- Enables us to change the implementation (refactor)
- Think of an object in an abstract way
- Replace details of an object's with a simpler contract

Protection of invariants

Guard clause + Postel's Law

Always valid



Object should never be in an invalid state
Not the caller's responsibility

"Be conservative in what you send, be liberal in what you accept"

Science of TDD

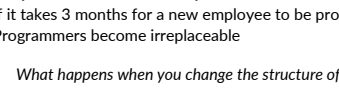
Form a hypothesis
- Prediction of falsifiable outcome
- Perform an experiment
- Measure the result

Compare
The actual to the predicated one

Refactor
Can you improve the code ?

Legacy code and memory
- If it takes 3 months for a new employee to be productive
- Programmers become irreplaceable

What happens when you change the structure of code ?



- Information in Long Term Memory becomes outdated
- Gets harder to work with the code base
- Acquired knowledge no longer applies

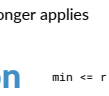
Triangulation

```
min <= r.At <= max
```

Add more test cases

until

Have defeated the Devil



"as the tests get more specific, the code gets more generic"

> *"Be careful editing unit test code; there's no safety net."*

Decomposition

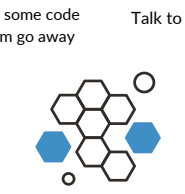
Code rot
- Code gradually becomes more complicated
- If no one pays attention top the overall quality

Thresholds
- Agree on a threshold can help curb code
- Cyclomatic complexity (<8 for ex)

80 / 24 rule
- Stay within a 80x24 character box
- Can help keep method smalls

Establish a culture that actively pays attention to code quality

Code that **fits** in your head



Plot outcome related to a branch in the code

No more than 7 things in a single piece of code

fits in a Hex Flower

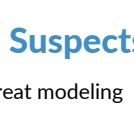
Cohesion

"Things that change at the same rate belong together. Things that change at different rates belong apart - Kent Beck"

Parse don't validate

Instead of "IsValid"

Return a ***Maybe***



Callers will be "forced" to handle **both cases**

"If you can measure the essence of a method in the signature, then that's a good abstraction"

API Design

Affordance : An interface is an affordance
- A set of methods, values, functions , objects
- Enables you to interact with an encapsulated package of code

Poka-Yoke
- Means "mistake-proofing"
- Mistake-proof artefacts and processes

Hierarchy of communication

```
1. Guide the reader by giving APIs distinct types
2. Guide the reader by giving methods helpful names
3. Guide the reader by writing good comments
4. Guide the reader by providing illustrative examples as automated tests
5. Guide the reader by writing helpful commit messages
6. Guide the reader by writing good documentation
```

"Don't say anything with a comment that you can say with a method name. Don't say anything with a method name you can say with a type"

Clear how to use it from its shape

on API

dot-driven development
Degree of discoverability

Command Query Separation

Command
- Methods with **side-effects**
- Should return no data (void)

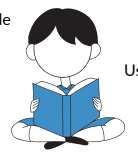
Query
- Methods that do return data
- Should have no side effects

"We can distinguish them without knowing implementation details."

Write code for Readers

Favor well-named code over comments

It may be you



Use types to keep you honest

X-out your code

```
public interface IReservationsRepository
{
    Task Xxxx(Reservation reservation);
    Task<IReadOnlyCollection<Reservation>> Xxx(DateTime dateTime);
    Task<Reservation?> Xxx(Guid id);
}
```

- See if you can still figure out what they do
- Helps you empathize with future readers

Teamwork

Use commit messages
- The best place to explain "why"
- Follow 50/72 rule

Continuous Integration

Integration means merging

Make small changes

Decrease integration risks

Merge as often as you can

it is a practice

Integrate at least every 4 hours

If you can't complete a feature in 4 h

Hide it behind a feature flag

Collective Code Ownership

Bus / Lottery factor
- If a single person 'owns'a part of the code base
- You're vulnerable to team changes

How many team members can be hit by a bus before development halts ?

"Any code changes should involve more than one person"

Pair Programming
Prevent knowledge silos

Mob Programming
Great for knowledge transfer

Rejection is an option

Can introduce latency

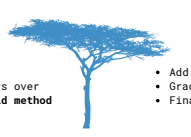
Set aside time for them

Code review
Check wether the code fits in your head

Augmenting code

Strangler Pattern

- Add a **new method**
- Gradually move callers over
- Finally delete the **old method**



- Add a **new class**
- Gradually move callers over
- Finally delete the **old class**

"For any significant change; don't make it in place; make it side-by-side."

Editing Unit Tests

Separate refactoring

Test code — Production code

Failure and trust

Don't trust a test that you haven't seen fail

Rhythm

Personal

Pomodoro technique
- Work in time-boxed intervals
- Like 25 minutes

Take breaks

Use time deliberately
- I start my day with two 25-minute time-boxes
- Where I try to educate myself

Team

Regularly update dependencies
- Dangerous to fall too far behind
- Beginning of each sprint for ex

Schedule other things
- Certificates management
- Database backups
- ...

Troubleshooting

Understanding

Scientific method
- Make a prediction / hypothesis
- Perform the experiment
- Compare outcome to prediction

Simplify
- Consider if removing some code
- Can make the problem go away

Explaining a problem

Write a question on Stack Overflow instead

tends to produce new insight

Rubber ducking

Talk to a rubber duck it will solve your problems

Defects

Reproduce defects as tests

```
git bisect
```

Identify the commit that caused the problem

Maximum time for a test suite : 10 seconds

Create different "containers" to isolate slow tests

"Abstraction is the elimination of the irrelevant and the amplification of the essential" - Robert C. Martin

Separation of concerns

Functional Core, Imperative Shell

Non deterministic queries / behaviors
- With side effects
- Close to the edge of the system

Complex logic
Write complex logic as **pure functions**

Logging

The more your code is composed from pure functions

The less you need to log

"Log all impure functions, but no more."

The Usual Suspects

STRIDE threat modeling
- Spoofing
- Tampering
- Repudiation
- Info disclosure
- Denial of Service
- Elevation of privilege

Other techniques
- Property-Based Testing
- Behavioral code analysis
- ...

CONTINUOUS DELIVERY

BY JEZ HUMBLE AND DAVID FARLEY

"CONTINUOUS INTEGRATION IS A SOFTWARE DEVELOPMENT PRACTICE WHERE MEMBERS OF A TEAM INTEGRATE THEIR WORK FREQUENTLY, USUALLY EACH PERSON INTEGRATES AT LEAST DAILY." - MARTIN FOWLER

CI PRINCIPLES

> "ELIMINATE THE NEED FOR INSPECTION ON A MASS BASIS BY BUILDING QUALITY INTO THE PRODUCT IN THE FIRST PLACE."
> - W. EDWARDS DEMING

MAINTAIN A SINGLE SOURCE REPOSITORY

AVOID HAVING SOURCE CODE BEING SCATTERED ACROSS MULTIPLE LOCATIONS

KEEP THE BUILD FAST - FAIL FAST

- BUILD SHOULD NOT TAKE HOURS TO HAPPEN
- HAVE SHORT FEEDBACK LOOPS

AUTOMATE THE BUILD

A SINGLE COMMAND SHOULD HAVE THE CAPABILITY OF BUILDING THE WHOLE SYSTEM

TEST IN A CLONE OF THE PRODUCTION ENVIRONMENT

MAKE YOUR BUILD SELF-TESTING

- CONFIRM THAT IT BEHAVES AS EXPECTED
- COMPUTER CAN REPLACE HOURS OF MANUAL TESTING WITH JUST MINUTES OF AUTOMATED TESTING

EVERYONE CAN SEE WHAT'S HAPPENING

- SEND NOTIFICATIONS
- FEEDBACK EVERYWHERE
- REDUCE FEEDBACK LOOP

EVERYONE COMMITS TO THE CENTRAL REPOSITORY EVERY DAY

- AS OFTEN AS POSSIBLE - MICRO INCREMENTS
- GUARANTEE THE SUCCESS OF THE INTEGRATION

MAKE IT EASY FOR ANYONE TO GET THE LATEST EXECUTABLE VERSION

- MAKE IT EASY FOR ANYONE TO GET THE LATEST EXECUTABLE VERSION
- AVAILABLE TO STAKEHOLDERS AND TESTERS

EVERY COMMIT SHOULD BUILD ON AN INTEGRATION MACHINE

> "OUR HIGHEST PRIORITY IS TO SATISFY THE CUSTOMER THROUGH EARLY AND CONTINUOUS DELIVERY OF VALUABLE SOFTWARE"

AUTOMATE DEPLOYMENT

- REDUCE THE DEPLOYMENT RISKS
- DEPLOYMENT AS A NON-EVENT

CD PRINCIPLES

REPEATABLE & RELIABLE PROCESS

- GET THE LIST OF DEPLOYED FEATURES
- NOTHING CAN BE INTRODUCED THAT HAS NOT BEEN TESTED

IF SOMETHING IS DIFFICULT, DO IT MORE OFTEN

VERSION CONTROL EVERYTHING

DONE MEANS "RELEASED"

BUILD QUALITY IN

EVERYONE IS RESPONSIBLE

4 PRACTICES

BUILD ONCE, DEPLOY MANY

- EARLY AND OFTEN
- ARTEFACT IS ENVIRONMENT AGNOSTIC
- AUTOMATE EVERYTHING

SMOKE TEST YOUR DEPLOYMENT

- "DOES CLICKING THE MAIN BUTTON DO ANYTHING?"
- DETERMINE THE STATE OF THE SYSTEM

USE PRECISELY THE SAME MECHANISM

TO DEPLOY TO EVERY ENVIRONMENT

IF ANYTHING FAILS

STOP THE LINE

#SHARINGISCARING

BY YOAN THIRION

@YOT88

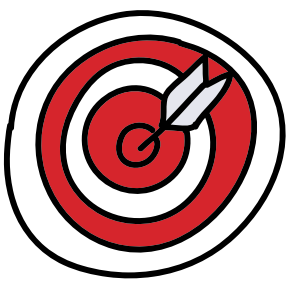
CULTURE IS EVERYTHING

BY TRISTAN WHITE

THE STORY AND SYSTEM OF A START-UP THAT BECAME AUSTRALIA'S BEST PLACE TO WORK

#sharingiscaring

1) DISCOVER THE CORE

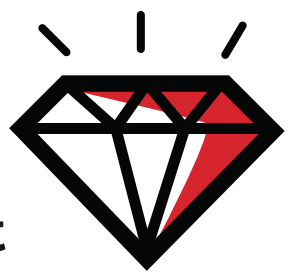


CORE PURPOSE

Define yours : Inspiring / Valid in Time / Help to think Expansively / Help you Decide / Truly authentic to your company

"A CORE PURPOSE IS THE REASON AN ORGANISATION EXISTS"

CORE VALUES (3 TO 5)



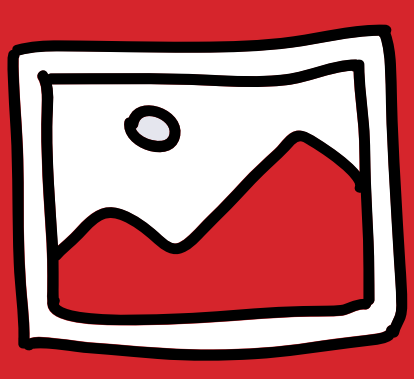
- Inspire great behavior
- Make them short, sharp and memorable
- Each value should be an action statement

SHARE CORE VALUE STORIES TO REWARD / RECOGNIZE / REEDUCATE

- MVP Program
- Share stories of team members
  - Living core values
  - Celebrate their successes

2) DOCUMENT THE FUTURE

CREATE A TEN-YEAR OBSESSION THAT ACTS AS YOUR NORTH STAR

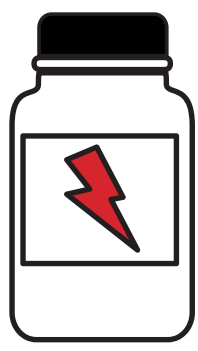


PAINTED PICTURES - 3 YEAR GOALS

- Broken down vision
- Make it : clear / specific / possible
- Communicate progress often
- Obsess over it
- Make it fun

"A STRONG CULTURE NEEDS A CLEAR VISION"

3) EXECUTE RELENTLESSLY



HAVE AN ENERGETIC DAILY HUDDLE

- Aligns everyone to the Painted Picture
- 12 minutes / day

ROBUST RECRUITMENT PROCESS

- Culture fit : examples of lived core values
- Passion for the work
- Passion for the company
- Key skills

"A STRONG CULTURE NEEDS EVERY TEAM MEMBER ALIGNED TO THE SAME VISION AND LIVING THE SAME VALUES."

4) SHOW MORE LOVE

MEMORABLE WELCOME EXPERIENCE

FACE-TO-FACE COMMUNICATION

PARTIES & CELEBRATIONS

GENUINE APPRECIATION / THKS

HAVE INTEREST FOR INFLUENCERS (Not on the payroll : Kids, Friends, Family, ...)

CULTURE BOOK
Story of your organization

19 STEPS
To build a GPTW

"CULTURE IS THE CEO'S RESPONSIBILITY : TOO IMPORTANT TO DELEGATE"

by Yoan THIRION @yot88

Dynamic **Re**teaming

The Art and Wisdom of Changing Teams

by Heidi Helfand

Dynamic Reteaming a.k.a. **Team Change**

People will **join** your team
Others will **leave**

Natural occurrence

When you **change** your team's composition, it:

- Creates a **new** team **social dynamic**
- Impacts the collective intelligence present on the team
- Brings **new learning** potential to the team as a whole
- Helps teams learn together and expand their skills

> "In essence, team change is **inevitable**, so we might as well get good at it."

Team

- At least two people working together
- Build something valuable for their customers
  - Shared work
  - Joint ownership of the outcome

Group

Collections of people
assigned across different teams

Ex: Community Of Practice
To spread similar ways of working

The Social Dynamic of a Team

Own unique social dynamic / "feel"

Changes over time

High energy ⇢ chemistry / **high performance**

Low energy ⇢ lacks chemistry / **low performance**

How To ?

like Kanban recommendations

- Start **where you are**
- **Visualize** your team structures
- **Observe** and get to know them
- **Incremental** reflection / adjustment
- **Experiment** and learn

Politics of Team Assignment and Change

Less Freedom

- Someone "at the top" **put** them on the team
- Manager **put** them on the team without their input

- Manager **included** their input when assigning team
- Managers / leadership **arranged self selection events**

- Team members **trade** places / tell managers
- Team members **form** their own teams

More Freedom

Reduces Risk and Encourage Sustainability
Decreases the Development of Knowledge Silos

- Within a team
  - Pair programming / TDD
- Team-to-team level
  - Reduce the development of knowledge silos by reteaming
  - Spreading knowledge out from one team to another

RISK

Reduces Team Member Attrition
Providing Career Growth Opportunities

Decreases Inter-Team Competition
Fostering a Whole Team Mentality

> When People Leave You Have a **New Team**...

One by One

solves the problem of **growth**

Add or remove a team member from a team

How do we **integrate** in the "new people" when our company is growing fast?

Dynamic Reteaming Patterns

For company growth

Onboard New Team Members

- Make it **Known** That You are **Hiring** in New Team Members
- Plan and **Communicate** about the Arrival of the New Team Member
- **Get Things Together** for the New Person Before They Arrive
- Assign a specific **mentor** within their team (Pair Program)

Grow & Split

teams **growing "too big"**

"Too many" people : can become inefficient

Can also apply it to **spread "best practices"** across your organization by

- Conservatively adding in people to a stellar team
- Then splitting the team later
  - When you feel all team members mastered the techniques you want to spread

Guidelines

1. **Why** are you splitting the team?
2. The **membership** on each of the resulting teams after the split should be made clear to everyone.
3. Try to **avoid sharing** team members between the two teams.
4. Let people **choose** which team they will move into.
5. The work of each of the split teams should be **separate**.
6. Don't let the team split **drag on forever** : choose a date on the calendar for "doing the split.
7. Consider coming up with new team names for each of the teams or engage the "new" teams.
8. Make sure any of your **tooling** is updated in advance of your team split event.
9. Determine the **facilities implications** for your team split.
10. Consider having "Team Liftoffs" or "**Startups**": discuss how you want to work together as a new team.
11. Get the team itself to "**own the split**, if possible".

Isolation

emergency situations

Failing product
Pivot to survive
Performance crisis
Outage

or to **innovate**

Creates beneficial silos by design
You form a team "off to the side" and give them process freedom

For the work

The **new work** is the inspiration for the team change

- **Isolation Pattern** for Pivoting & Innovation
- Form Teams and Reteam Around the Work
  - Ex : TRIAD (Product Manager, Engineering representative, UX)

or when "**Overloaded**" with work

- If prioritization of work is not clear, people can suffer...

For the code

- **Spike**: research story that comes up from time to time in teams
- **Refactor**
- Share **Production Support**

> When you switch pairs, or teams for that matter, you are exposed to new people and new ideas.
> You just learn more. That feels good to us as humans.

For Learning, Fulfillment, and Sustainability

- When you switch within a team or across teams
  - We switch to share knowledge with each other
  - The aim is to **spread out the knowledge** for learning and sustainability
  - We Want to be with other people and learn from them
- To **Support** a Feature
- Switching for Personal **Growth & Learning**
- Empower People to **Re-Role**
  - It can make your organization stickier and help you retain people

Enables the Continuous Integration of Ideas

Brings Consistency & Facilitates Reteaming

Mob Programming

Engagement at work can happen when you are intellectually stimulated and are able to **continually learn** in your job.

Switching

"avoid **tower of knowledge**"

"low bus count" problem -> members working a lot on their own

Incorporate reteaming to spread knowledge within or around your teams

Merging

teams that are **too small**

need more people to have collaboration opportunities like pair programming

Teams might merge as a strategy to combat dependencies across two or more teams

Other Reteaming Reasons

For Short-Term Events
Daily Learning Sessions
1 hour of mob-style learning each day

To Find a Better Fit

To Liberate
Silenced People Could Be a Sign for a Reteam
Prisoners in meetings...

Get Good at Dynamic Reteaming

Cultivate Community
to Prime for Future Reteaming

Design Events
to Build Relationships Across the Organization

Hold **Educational Off-sites**
to Sharpen Skills and Strategize

Retrospectives
Systemic Retrospectives
Retrospectives with Groups of Related Teams

Bring Remote Workers
into the Office and Send Team Members to Them

Give **Teams Budgets**
to Create their Own Social Events Events

Reflect
on Team Compositions and How to Shift

Create Opportunities
to Get To Know Key Leaders
in Different Departments: "coffee chats" with VPs, key POs, ...

HOW TO AVOID A CLIMATE DISASTER BY BILL GATES

Why zero ?

51 Billion Tons
of greenhouse gases to the atmosphere per year

We are here today → **0** "near net zero" **What we need to aim for**

- **1°C** increase since preindustrial times
- **Mid-century** : between **1.5°C and 3°C**
- **End of century** : between **4°C and 8°C**

Trouble getting clean water
Twice as many people

Corn production go down twice as much

2-degree rise wouldn't be 33 percent worse than 1.5
Could be 100 percent worse

Mosquitoes will start living in new places
Malaria

Heatstroke
Because of humidity

By 2100 could be FIVE times as deadly than COVID 19

"The climate is like a bathtub that's slowly filling up with water.
Even if we slow the flow of water to a trickle, the tub will eventually fill up and water will come spilling out onto the floor."

Give a sense of how much is a lot / a little

Build your mental framework

How much of the 51 are we talking about ?
Convert numbers into a percentage of the annual total of 51 billion tons

What's your plan for Cement ?
- A shorthand reminder that emissions come from **5 different activities**
- We need solutions in all of them

How much power are we talking about?

When you hear...	Think
Kilowatt	
Gigawatt	
> hundred Gigawatts	

How much space do you need ?
- How much space will be required to produce that much energy
  - **Wind** : 1-2 Watts per square meter
  - **Fossil Fuels** : 500-10,000 Watts per Square Meter

How much this is going to cost ?

Most of the zero-carbon solutions are **more expensive** than fossil-fuel ones

Prices **don't reflect** the environmental **damage** they inflict

The difference between the 2 prices : **GREEN PREMIUMS**

It can be negative : green can be cheaper

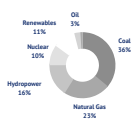
"We need the premiums to be so low that everyone will be able to decarbonize."

Green Premiums
Difference in cost between a product that involves emitting carbon and an alternative that doesn't

5 types of activity

27 % How We Plug In
- Electricity : A cheap source of energy always available
- Getting all the world's electricity from clean source won't be easy

Fossil fuels account for **two-thirds** of all electricity worldwide



Make Carbon-free electricity

Nuclear fission
- Getting energy by splitting atoms apart
- Only carbon-free energy source that can reliably deliver power day / night

Nuclear fusion
- Getting energy by pushing atoms together / fusing them
- At least a decade away from supplying electricity to consumers

Offshore wind
- Putting wind turbines in an ocean or other body of water

Geothermal
- Deep underground : hot rocks that can be used to generate electricity
  - Amount of energy we get per square meter is quite low

Store electricity

Batteries
- Hard to improve on them
- Can improve by a factor of 3 but not by a factor of 50

Pumped hydro
- When electricity is cheap : pump water up a hill into a reservoir
- When demand goes up : let the water flow back down the hill

Thermal storage
- When electricity is cheap use it to heat up some material

31 % How we make things
Use tons of steel, cement, glass, plastic

Making **1 ton of steel** produces **1.8 tons of** CO_2

96 **Millions tons of cement produced every year in America**
600 pounds for every person in the country

Bring the premium **down**
- Public policies to create demand for clean products
- Create incentives to buy zero-carbon cement / steel

16 % How we get around
Biggest cause of emissions in the United States

Do less of it
Walking / biking / car-pooling

Use fewer carbon-intensive materials
in making-cars

4 ways to cut down on emissions from transportation

Use fuels more efficiently

Switch to electric vehicles
alternative fuels

19 % How we grow things
70% agriculture / 30% deforestation

Global population is headed toward 10 billion people by 2100

40% more people
- We'll need more than 40% more food too
- As people get richer, they eat more calories

Methane : main agriculture culprit
- 28 times more warming per molecule than CO2 over the course of a century
- Nitrous oxide causes 265 times more warming

Food thrown away
- 20% : Europe / Industrialized parts of Asia, Sub-Saharian Africa
- 40% in the US

→ **Wasting less of it** **Behavior change**

We can **cut down on meat eating** while still enjoying the taste of meat :
- Plant based meat
- Artificial meats
- Cell based meat

Stop deforestation
- Incentives to cut down trees are stronger than the ones to leave them alone
- We need political / economic solutions

→ **4 tons over the course of 40 years**
the CO2 that a **tree** can **absorb** in its lifetime

> *"The cruel injustice is that even though the world's poor are doing essentially nothing to cause climate change, they're going to suffer the most from it"*

7 % How we keep cool and stay warm
Heating / cooling / refrigeration

A plan for getting to Zero

Technologies needed

Hydrogen produced without emitting carbon
Electrofuels
Nuclear fusion
Grid-scale electricity storage
Pumped hydro
Carbon capture (Direct air / point capture)
Zero-carbon plastics
Underground electricity transmission
Zero-carbon steel
Geothermal energy
Advanced biofuels
Zero-carbon cement
Drought and flood-tolerant food crops
Zero-carbon fertilizer
Plant and cell-based meat
Thermal storage

Science tells us that in order to avoid a climate catastrophe, rich countries should reach net-zero emissions by 2050

Quintuple clean energy / climate-related **R&D** over the next decade

Make bigger bets on high-risk R&D projects

Match R&D with our greatest needs

Work with the industry from the beginning

Expand the supply of Innovation
To get these technologies ready soon

Accelerate the demand for Innovation

Create incentives that lower costs
And reduce risk

Build the infrastructure
Bring new technologies to market

Put a price on carbon : eliminate Green Premiums

Change the rules
so new technologies can compete

Clean standards
- Electricity
- Fuel
- Product

Set standards in procurement programs for example

What each of us can do ?

> *"The market is ruled by supply and demand : we can have a huge impact on the demand side"*

Personal action : important for the signals

Elected officials will adopt specific plans if their voters demand it
- Make calls, write letters, attend town halls
- Run for office

As a citizen

- Sign up for a **green pricing** program : **electricity utility**
- **Reduce** your home's emissions
- Buy an **electric car**
- Try a **plant-based burger**

As a customer

> *"We need to make it possible for low-income people to climb the ladder without making climate change worse."*

As an employee or employer

Push your company to do its part :
- Set up an internal carbon tax
- Prioritize innovation in low-carbon solutions
- Be an early adopter
- Engage in the policy-making process
- Help early-stage innovators get across the valley of death

#sharingiscaring

by Yoan THIRION

@yot88

La liberté du commandement

L'esprit d'équipage

Vice Amiral LOÏC FINAZ

- Mener des hommes au combat pour porter la mort
- Peut conduire à la recevoir

Commander

Diriger une entreprise

Structurer l'organisation

Faire réfléchir et grandir

Partager une vision, Mobiliser l'intelligence

Préserver le patrimoine

Générer de la valeur / innover

Manager

Commander 1 bâtiment de guerre
c'est aussi
Diriger 1 entreprise (manager)

Piliers de notre sagesse et de notre performance associations

AUTONOMIE ET SOLIDARITÉ

Susciter l'initiative et accepter l'échec

"Rassurez-vous, je suis là; si vous échouez, je corrigerai le tir; je suis là pour cela."

FONCTIONS ET RESPONSABILITÉ

Des fonctions différentes, une même responsabilité

Créer
Susciter le mouvement

Fédérer
Faire évoluer
S'épanouir

"La fonction fait l'homme tout autant que l'homme peut faire la fonction."

HIÉRARCHIE ET PARTICIPATION

Hiérarchie importante pour prendre des décisions au combat

Culture participative très forte

Intelligence collective pour trouver les solutions

"Le système hiérarchique n'érige pas la confiance, il utilise celle que fédère les chefs grâce à leur culture participative."

EXIGENCE ET BIENVEILLANCE

Vis-à-vis de
Soi-même (exemplarité)
Ceux qui leur sont confiés

Sans exigence 1 chef n'obtiendra / réussira rien

Sans bienveillance il détruira tout

"[...] commander, diriger, est l'une des plus belles façons de servir ceux qui nous sont confiés."

Le chef doit être une énergie :
met en mouvement, convainc, fait durer, vivre et gagner

Besoin d'une cohérence entre ces 2 qualités

ENERGIE ET CULTURE

La culture permet
- De s'élever
- De s'enrichir
- De s'armer pour les luttes de l'existence

"La véritable école du commandement est la culture générale."

INTELLIGENCE ET COURAGE

Chef très intelligent et peu courageux, incapable de :
- Décider
- Agir

Chef courageux et crétin :
- Un maniaque
- Ou 1 fou

"[...] il faut cultiver le goût du risque et la capacité de l'assumer, oser l'audace de la solution originale."

PAROLE ET TEMPS

C'est par la parole que l'action du dirigeant existe

Parole du chef adressée directement :
- Suscite espoir et enthousiasme
- Apaise les craintes
- Remonte le moral (dans la crise ou la défaite)

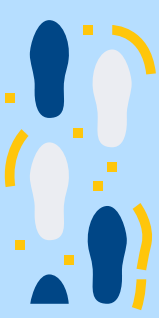
"Par la parole, à la fois complément et expression de son énergie, il convainc, met en mouvement et s'inscrit dans le temps."

AMBITIONS POUR NOS RESPONSABILITÉS

"Rien n'irrigue plus les bonnes pratiques du management que les exigences du commandement."

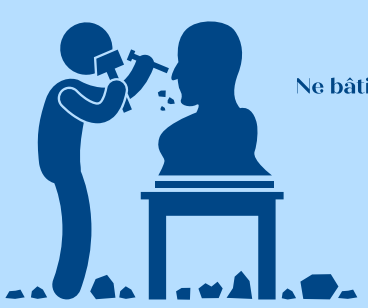
QUE NOS PAS DEVIENNENT SILLAGES

Apprenons à ne pas laisser de traces dans ce monde



Qui n'en vaillent pas la peine

QUE NOS MAINS SACHENT ÉDIFIER



Ne bâtir que du beau et de l'utile

AYONS TOUJOURS NOTRE REGARD SUR NOTRE LIGNE DE FOI

Croyons en ce monde

Porter nos regards sur l'horizon

COMMANDER C'EST AIMER

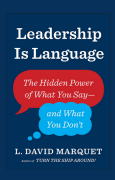
Tout le reste n'est que discours

"Faites de vos équipes, de vos services, un équipage"

#sharingiscaring

Par Yoan THIRION @yot88

Leadership is language



The Hidden Power of What You Say and What You Don't - L. DAVID MARQUET

Redwork
Active production
"Prove"

+

Bluework
Thinking / Learning
"Improve"

"We are all both Redworkers and Blueworkers"

"A real danger to use old thinking in new situations"

A NEW PLAYBOOK

1) Control the clock : exiting redwork

Bluework allows us to adapt BUT you have no chance to do bluework if you don't control the clock

Make a pause possible : Invite a pause
Give the pause a name

Call a pause
Preplan the next one

"If you are on the team and see something unexpected, it's your responsibility to call a pause"

2) Collaborate : into the bluework

Let the doers be the deciders : move from coercion to collaboration

Vote first, Then discuss
Anonymous polling, Ask probabilistic questions
Use probability cards, Dot voting

Be curious, not compelling
Seek first to understand, Then to be understood
Ask better questions (start with what / how)

"Before I tell you what I think we should do, what would you do if I weren't there"

LEADERS SPEAK LAST

Invite dissent rather than drive consensus
Dissent cards

Give information, not instructions`
From "Park there" to "I see a parking spot there"

"A leader's obligation is to listen to the dissenters"

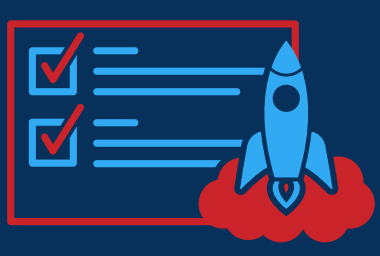
3) Commit

Commit to learn, Not (just) Do
Develop hypothesis to test rather than making decisions to execute

Commit actions, not beliefs
Once the decision is made don't try to convince dissenters

Chunk it small
BUT do it all

4) Complete : the end of Redwork



Chunk work for frequent completes early
At the beginning of a project : shorter redwork periods
More frequent bluework periods to bias toward learning and improving

Celebrate with, NOT For

Celebrate FOR
"Good job" / "I m so proud of you"
Transference of the reward to us rather than leaving it with the person

Celebrate WITH
Use descriptive statements : "I see", I" noticed", "It looks like"

Focus on behavior, not characteristics

Focus on Journey, Not destination
Invite people to tell their story

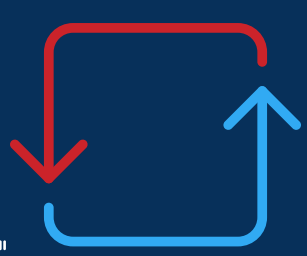
#sharingiscaring

5) Improve : completing the cycle

"Employees with the autonomy to decide how to go about solving problems and achieving goals innovate"

Forward, Not Backward
"What do we want to do differently next time ?"

Outward, not inward
Focusing on others instead of oneself
"What could we do better serve our customers ?"



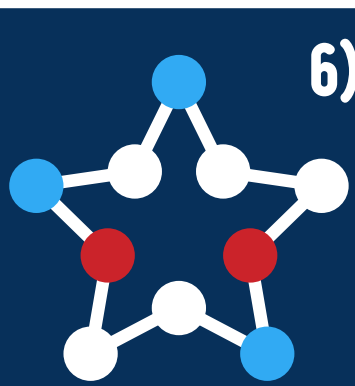
Process, not people
"How could this be done better ?"

Achieve excellence, Not avoid errors

6) Connect : enabling play

Flatten the power gradient
Amount of social distance between one person and another

Admit you don't know
Hard to connect with a Know-it-all



Be vulnerable
"How is everyone feeling about this ?
I think I m moving away from excited toward worried"

Trust first
What do we want to do differently next time ?

"Changed the way we communicated, changed the culture"

by Yoan THIRION @yot88

THE PROGRAMMER'S BRAIN

by Felienne Hermans

Code

Short Term Memory (STM)
- RAM or a cache of our brain
- Just a few items fit in STM (<12)
- Store some of the info we encounter

Long Term Memory (LTM)
- Hard drive of our brain
- Can store memories for a very long time
- Retrieve related knowledge (ex : keywords)

60% of our time

Info traveling from the STM into the working memory

Combined with information from the LTM

Working Memory (WM)
- Processor of our brain
- The actual "thinking" happens here
- Mentally execute the code aka TRACING

When we read code

Why is reading unfamiliar code hard?

LIMITED

Short term memory
- Time : 30 seconds
- Size : 7 +/-2 things

How to read code better ?

"More concepts, data structures and syntax you know the more code you can easily chunk, and thus remember and process"

Learn programming syntax

Use Flashcards
- Front : prompt
- Back : corresponding knowledge

Remember syntax longer
- Retrieval : trying to remember something
- Elaboration : connecting new knowledge to existing memories

Read / Hide / Write code exercises

Write CHUNKABLE Code

"Experts group info in logical ways a.k.a chunks"

Write COMMENTS
High-level comments help to chunk larger pieces of code

Use Design PATTERNS
Help to process code faster

Leave BEACONS
var names, operators (+, >), if, else, comments,...

How to not forget things ?

"We cannot remember things for a long time without extra practice"

Spaced repetition
- Practice regularly
- Best way to prevent forgetting

Revisit your Flashcards
- Once a month
- Each repetition strengthens your memory

DON'T FORGET

After 2 days, just 25% of the knowledge remains in our LTM

Read complex code easier

Reduce cognitive load

Refactoring code
Ex : replace unfamiliar language constructs

"Our ability to learn a natural language can be a predictor of your ability to learn to program."

Dependency graph
- Circle variables
- Draw lines between occurrences

State table
- Focuses on the values of variables
- 1 column / variable
- 1 line / step in the code

Cognitive load
- Capacity of our Working Memory
- Capacity : 2 to 6 "things"

Roles of variables
(Sajaniemi's framework)

- Fixed value : value does not change after initialization
- Stepper : variable stepping through a list of values
- Flag : has happened or is the case
- Walker : traverses a data structure (search index)
- Most-recent holder : holds the latest value encountered
- Most-wanted holder : holds the best value found so far
- Gatherer : collects data and aggregates it
- Container : holds multiple elements
- Follower : keep track of a previous value
- Organizer : transformed variable
- Temporary : used only briefly

"Understanding what types of information variables hold is key to being able to reason about and make changes to code."

"Many similarities between reading code and reading natural language"

Text comprehension strategies applied to code

Activating
Actively thinking about code elements help our WM to find relevant information stored in the LTM

Monitoring
- Keep track of what we are reading and our understanding
- ex : ticking the lines

Inferring
Inferring the meaning of variable names

Questioning
- Asking ourself questions while reading code
- Help us understand the code's goals and functionality
- ex : What are the 5 most central concepts of the code?

Determining importance
Identify which parts of the code are likely to have the most influence on the program's execution

Visualizing
List all operations in which variables are involved (dependency graph, state table,...)

Summarizing
- Write a summary of code in natural language
- Help us gain a deeper understanding of what's happening in that code

Goal of the code: what is the code trying to achieve?
Most important lines of code
Most relevant domain concepts
Most relevant programming constructs
...

Write better code

Search....

Avoid

Clear names help our LTM
LTM searches for related informations

Abbreviation
Check Hofmeister research

Snake Case -> use camel Case
camelCase leads to higher accuracy

Avoid Arnaoudova's linguistic anti-patterns

- Methods that do more than they say
- Methods that do the opposite than they say
- Identifiers whose name says that they contain more than what the entity contains
- Methods that say more than they do
- Identifiers whose name says that they contain less than what the entity contains
- Identifiers whose name says the opposite than the entity contains

Their occurrence in 7 open-source projects :
11% of setters also return a value
2.5% of methods : method name + comment = opposite descriptions
64% of identifiers starting with "is" turned out not to be Boolean

LTM can store different types of memory

Memories

Procedural / Implicit
- How to do something
- ex : How to run a bike

Declarative / Explicit
- Memories we are explicitly aware of
- Facts we can remember

Episodic
- Memories of experience
- ex : meeting our wife / husband

Semantic
- Memories for meanings / concepts / facts
- ex : 10 x 10 = 100

"Experts heavily rely on episodic memory when solving problems / rely on solutions that have previously worked for similar problems."

Getting better at solving complex problems

Automatization
create implicit memories

"Set some time aside every day to practice and continue until you can consistently perform the tasks without any effort"

Deliberate practice to improve skills
- Repeat a lot
- It frees up cognitive load for larger problems
- ex : deliberately type 100 for loops when struggling with it

Study worked examples
create episodic memories

Code reading club
- Exchange code / explanation
- Learn from each other

Read books / blog post
- About source code

Explore github
- Choose repositories (domain knowledge)
- Focus on the programming itself

Deliberate practice : requires focused attention and is conducted with the specific goal of improving performance.

Worked examples : something like a recipe which describes in detail the steps that are needed to solve the problem.

Better handle interruptions

20 % of developers time on interrupts

15' to start editing code after an interruption

Store mental model
- Apart from the code
- Comments : excellent location to leave it
- Warm-up period in comprehension activities

Prepare for it

Help your "Prospective memory"
- Put TODO comments in the part of the code
- Remind you to complete / improve part of the code

Label subgoals
- Write down small steps of a problem
- Use mind maps for example

Prospective memory : memory of remembering to do something in the future. (related to planning / problem solving)

On-boarding process

Typical

dev throws information

Senior dev

Newcomer

High cognitive load

Explain only relevant informations

Separate

Domain learning

Exploring code

Support the LTM of the newcomer

Exploration
- Browse the codebase
- Get a general sense of the codebase

Searching
ex : find a class that implements a certain interface

Transcription
- Give the newcomer a clear plan
- Implement it

Limit tasks to ONE programming activity

Comprehension
Understand aspects of the code
ex : summarize a specific method in natural language

Incrementation
- Add a feature to an existing class
- Creation of the plan for the feature.

Start with it : read code together

Refactoring at Scale

By Maude Lemaire

Refactoring

Restructure existing code
WITHOUT changing its external behavior

At Scale

- One that affects a substantial surface area of your systems
- Involves typically large codebases

Benefits

- Increase developer productivity
- Greater ease identifying bugs

Risks

- Serious Regressions
- Unearthing Dormant Bugs
- Scope Creep

When ?

Shift in Product Requirements

Performance issues

Code Complexity Hinders Development

Using a new Technology

Small Scope

When NOT ?

For Fun or Out of Boredom

When You Don't Have Time

Because You Happened to Be Passing By

To Make Code More Extendable

PLANNING

MEASURE OUR STARTING STATE

Measure Code Complexity

- Halstead metrics
- Cyclomatic Complexity
- NPath Complexity

Test Coverage Metrics

- Quantitatively : proportion of code under test
- Qualitatively : suitable test quality has been attained

Documentation

- Formal : everything you most likely think of as documentation
- Informal : Chat / email transcripts, Bug Tracking system, ...

Version Control

- Commit messages : keywords for given code
- Commits in Agg : change frequencies, authorship

Reputation

- Low-effort means of collecting reputation data
- Interview fellow developers

Build a Complete Picture

Pick one metric from every category

DRAFT A PLAN

Define your end state

Outline all starting metrics and target end metrics

Map the shortest distance

- Open a blank document technique
- OR Gather a few coworkers

Identify Strategic Intermediate Milestones

1) Does this step feel attainable in a reasonable period?
2) Is this step valuable on its own?
3) If something comes up, could we stop at this step and pick it back up easily later?

Dark Mode / Light Mode

Compare pre-refactor and post-refactor behavior :

- Both implementations are called
- The results are compared

Dark
The results from the **OLD** implementation are **RETURNED**

Light
The results from the **NEW** implementation are **RETURNED**

Choose a Rollout Strategy

How To ?

- Put in place an abstraction
- Enable dark mode
  - Monitor any differences between the 2 result sets
  - Track down and fix any potential bugs in the new implementation
- Enabling dark mode to broader groups of users
  - Continue logging any differences in the result sets
- Opt groups of users into light mode
  - Until everyone is successfully processing results from the new implementation
- Disable execution of both code paths
- Remove the old logic

Clean Up Artifacts

- Feature Flags
- Dead Code
- Comments (TODOs)

Reference Metrics

Include definitive progress metrics

Share your plan

- Provide Transparency
- Gather perspective to strengthen it

"No refactor is complete unless all remaining transitional artifacts are properly cleaned up"

GET BUY-IN

Always remember

Aren't Coding

Are Evaluated Differently

Managers

See the Risk

Need to Coordinate

Persuade Them

(some techniques)

Using Conversational Devices

Build an Alignment Sandwich

Rely on Evidence

Play Hardball

BUILD THE RIGHT TEAM

2 Ways to Enlist Someone

Active Contributor

- Heavily involved from day one
- Actively contributing to the effort by writing code
- Consulted for input on the execution plan

Subject matter experts (SMEs)

- Agreed to be available to talk through solutions with you
- Answer questions
- Can do some code review

Matchmaking

Match each expertise with one or more people

"To execute on a large refactoring effort successfully, we need our own Ocean's 11 [...] a team just the right size with just the right skills"

EXECUTION

COMMUNICATION

Stand-Ups

Everyone aligned at regular intervals

Retrospectives

Reflect on the latest iteration cycle

Weekly Syncs

- 1st part : accomplishments
- 2nd part : discuss any important topics

When Kicking Off

Single Source of Truth
Choose a platform to collect all documentation

Set Expectations
Draft a communication plan

During Project Execution

Announce Progress

Execution Plan
Living Version

Within Your Team

Outside Your Team

"Policy of no laptops and minimal phone usage during meetings"

PROGRAM PRODUCTIVELY

Prototype

Early and often

Help move faster

Know your solution won't be perfect

Not spend too much time perfecting the details

Be willing to throw code away

Keep Things Small

- Commit small, incremental changes
- Makes it much easier to author great code

Test, Test, Test

- Confirm everything has remained unaffected
- Or pinpoint the precise moment at which the behavior diverged

Asking the "Stupid" Question

- Prioritize clarity
- Over maintaining an illusion of omniscience

MAKE THE REFACTOR STICK

Foster Adoption through education

Active
Planning / leading workshops

Passive

- Step-by-step tutorials
- Online courses, ...

Integrate Improvement into the Culture

To maintain a healthy codebase

- Continuous small refactoring
- Incrementally improve areas of the codebase

Hold design reviews

- Early in the feature development process

Encourage design conversations

Case Studies @Slack

- Redundant Database Schemas
- Migrating to a New Database

Technical Agile Coaching with the Samman Method

by Emily Bache

A METHOD for people who want to make a difference and improve the way software is built

Wording

Samman : Swedish word for "together"
Describes this coaching method
Ensemble : French word for "together"
Describes Mob Programming

Focus on

Technical practices
How people write code

Foundation

Cultivate good relationships
Effective ways to learn from one another
Change behaviours for the long term

LEVELING UP A WHOLE TEAM TOGETHER

Software development these days is a team sport and it doesn't work to only train individuals.
Samman coaching aims to create a whole-team culture shift.

WHY ?

Build new features with
Shorter lead time
Higher quality

Attract skilled developers
Avoid drowning in technical debt

Increase business agility and success

ON WHAT ?

Incremental / Iterative Development
Safe refactoring

10101010
1010

Better unit tests
Continuous Integration

HOW ?

Ensemble working
Learning Hours

TIMELINE

10-20 coaching days / Team

EXPECTED OUTCOMES

1) AWARENESS ON
Good unit tests
Continuous Integration
Refactoring

2) NEXT
Successfully meet deadlines
Deliver High Quality Code

MEASURES

Attitudes
Deadlines met
Bugs reduction
Productivity

ENSEMBLE WORKING

Friendly people collaborating like musicians

"All the brilliant minds working together on the same thing, at the same time, in the same space, and at the same computer - We call it 'Mob Programming'" - *Woody Zuill*

ROLES IN THE "ENSEMBLE"

TYPIST

Has the keyboard and mouse
Enter the code for the Ensemble

NAVIGATOR

Speaks for the Ensemble
Explains what code enter

COACH

Promote better ways of working
Spread Knowledge

TEAM-MEMBERS

Lead the work
Talk and make the decisions

FACILITATOR

Remind working agreements
Help to reflect and improve

OTHER ROLES

Researcher : Search for the Ensemble
Archivist : Log choices

LET THE ENSEMBLE GIVE YOU SUPERPOWERS

Learn as much **from the team** as they learn from you
Keep your technical skills **sharp & up-to-date**
Continue to **write code every day**

KINDNESS, CONSIDERATION AND RESPECT

Treat everyone with kindness, consideration, respect
Pay attention
Yes and ...
Call out bad behavior

COACHING BEHAVIORS IN THE ENSEMBLE

Teach
Breathing space
Coach
Retrospect
Mentor
Facilitate
Observe
Take Short Breaks

LEARNING HOURS

Turn up the good

A lot of great sample sessions are described in the book

SHORT TRAINING SESSIONS

People practice coding skills
Learn new techniques

WHY 1 HOUR EVERY DAY ?

Become more productive and happier
Add up more than compensate the time you spent

For an organization to succeed in the modern world, it needs to be a learning organization

LEARNING OUTCOMES AND OBJECTIVES

What really matters :
What happens afterwards ?
Will they be able to apply what they've learnt ?
What is the outcome you're hoping to achieve ?
Start with the end in mind

4C LEARNING MODEL by Sharon Bowman

Connect : Get people in the right head-space
Concept : Introduce the new skills you want the participants to learn
Concrete : Hands-on exercises to practice
Conclusions : An opportunity for people to consolidate

DESIGN LEARNING EXPERIENCES THAT FIT WITH HOW THE BRAIN WORKS

MOVEMENT TRUMPS SITTING
Learn more when you're feeling awake

TALKING TRUMPS LISTENING
Talk reinforces your memory

IMAGES TRUMP WORDS
Tell stories that bring pictures in mind

WRITING TRUMPS READING
Force to concentrate

SHORTER TRUMPS LONGER
10-20 minute chunks

DIFFERENT TRUMPS SAME
Vary the format

SAMMAN COACHING ENGAGEMENTS

BEGINNING COACHING WITH A NEW ORGANIZATION

1) PRESENT YOURSELF

Tell stories and anecdotes
Explain what ensemble working is
Why it is such a useful forum for a coach

2) KICK-OFF WORKSHOP WITH EACH TEAM

Team introduction - 10'
How long have you worked :
- For this company?
- With software development as a career?

Architecture overview - 15'
Sketch the architecture on a piece of paper

Issues in the codebase - 45'
Show me :
- Typical tests
- Code you would like to have tests for
- Well designed code
- Buggy code

Structured discussion - 30-40'
TRIZ, Lean Coffee, Speedboat

Takeaways - 5'
Organize your notes by team

PREPARING FOR A TECHNICAL COACHING CAREER

CORE SKILLS

TDD / Refactoring
Software Design
Continuous Integration
Pair programming
Designing test cases

Presenting topics
Chairing a meeting
Facilitating a retrospective
Sketching / explaining a design
Live code

IDEAS TO IMPROVE

Start a regular Coding Dojo
Attend a Code Retreat
Study books and videos
Go to agile conferences

Attend formal training (SM or Agile Coach)
Join a meetup and organize events
Give presentations at an internal COP

https://sammancoaching.org/

by Yoan THIRION @yot88

SOFTWARE CRAFT

TDD, Clean Code and other essential practices

Cyrille Martraire
Fabien Hiegel
Arnaud Thiéfaine
Houssam Fakih
Dorra Bartaguiz

Develop ?

"Working code is a low bar."

Software is never finished
It is always changing

Define the need
As difficult as writing the matching code

It is read /understand the code
At least as much as writing it

No hacker
Not a virtuoso

A Community

Benevolent
Help
Transmission
No elitism

Toolbox

Test-Driven Development
Behavior-Driven Development
Clean Code
Domain-Driven Design
Legacy remediation
Pair / Mob Programming
OO, FP, SOLID

Test-Driven Development (TDD)

Write a failing test
Make the test pass
as quickly as possible
Improve the code
Better readability

Reduce fears
Break down the problem

Take the time to make the code readable
Refactoring

Help to define an objective to achieve

Write as little code as possible

Our tests

- Transcribe the business rule
- Easy to determine cause of failure

Should / When
Given / When / Then

`DivideShould.Throw_an_invalid_operation_when-denominator_is_zero`

Efficiently program **complex** features

Uncle Bob's 3 rules of TDD

- Only write code to make a test pass
- When writing a test, write the minimal to make a test fail; this includes your code not compiling
- Write the minimal amount of code to make a test pass

Techniques and Principles of Clean Code

> *"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."* – Martin Fowler

A communication exercise

Degradation in the urban environment
If not repaired: leads to further damage
Tendency to be sloppy
Faced with a degraded environment

Theory of the broken window →

Boy scout rule
KISS
Keep It Simple and Stupid
YAGNI
You Ain't Gonna Need It

Be **exemplary**

The 4 Rules of Software Design - Kent Beck

Passes the tests
work as intended

Reveals intention
be clear

No duplication
DRY

Fewest elements
keep only what is important

Focus on Naming

Express intent
- Explain what you want to do
- Why you want to do it

rely on the business

Common nouns
Verbs
Rename

Comment sparingly

How to express the same knowledge through code?

Often a code smell
Concerns about cleanliness?

"Don't comment bad code - rewrite it." - Brian Kernighan

Report a subtlety
Optimization for example

Flag problems
TODO / FIXME

Legal information

Split in a way to facilitate discovery + navigation

Also applies to tests
- Tests quality must be exemplary
- Put as much or more effort into it

Storytelling
- Our code must tell a story
- Should be read as a table of contents

Properly format your code
- Respect of the standard defined by the team
- Maximum indentation level limited (2 max)
- Lines of code not very wide

Cut out the functions

Focus on the service provided
Few parameters
Must do only one thing
Only 1 level of abstraction

Contractual simplicity

1 single output parameter
Consider each parameter as immutable
Encapsulate complex in/out in dedicated types
Define variables as closely as possible to their use

Agile Specifications with Behavior-Driven Development (BDD)

3 Amigos

Discuss the features to be built

Product Owner
Product Manager
Represents the need
Seeking value

Dev
Represents the implementation
Technical feasibility

Tester
Challenge the other 2
Identify failures in thinking
Problems in expressing needs

Concrete Examples

In the language of business

- Expressing the need
- Concrete details
- Self-sufficient

Identify key scenarios

Increments of specifications
Short workshops
30' / day

Automation

Gherkin Syntax

Dev
Acceptance criteria

Later
Non-regression tests

Benefits

Acceptance criteria
to determine the progress of developments

Shared understanding among all

Non-regression tests
with good functional coverage

Living Documentation
evolutionary

```
Feature: Offer discounts to loyal customers
  To encourage customers to come back and buy,
  As a book seller,
  I want to offer discounts to loyal customers
Exemple: No discount below 50 EUR of purchase
  Given a cart with 2 items for a total of 49 EUR
  When the customer checks out,
  Then the total does not include any discount
```

> *"Having conversations is more important than capturing conversations is more important than automating conversations"* - Liz Keogh

Improve the **efficiency** of **collaboration** between the specialists involved to build better software at lower cost.

Working effectively with pair / mob programming

Pair Programming

Frequent rotation
Know how to slow down

Driver
- The one behind the wheel
- Control the machine

Navigator
- Does not act directly on the machine
- Takes a step back

Why?

Learn and progress
Share & transmit
Reduce the Truck Factor
Assure / reassure yourself
Motivate / help each other

Several styles

Ping-Pong
- TDD + Pair
- Alternate at each red phase

Strong Style
- "I have an idea"
- "Here is the keyboard"

Silent
Only the code to communicate

Evil Twin
- Trick your partner
- Highlight edge cases

QUICK TIPS
Respect your partner
Transparency
Shared objective
Retrospective
Take breaks
Agenda
Know how to say no

Disengagement
Silence
Different machines
Fixed roles

Ensemble / Mob programming

Simultaneous collaboration of each member

> *"All the brilliant minds working together on the same thing, at the same time, in the same space, and at the same computer."* - Woody Zuill

Refactoring Techniques

"Rework a source code without modifying its behavior, in order to improve its readability and to make it much easier to maintain / accommodate future evolutions".

Antidote to technical debt
- Do not let it grow
- Regular and continuous practice

Obstacles
No culture of code quality
inside the team
Fear
to cause regressions
1 item of cost
from the client perspective
Lack of knowledge
know-how

How?

Transform in baby steps

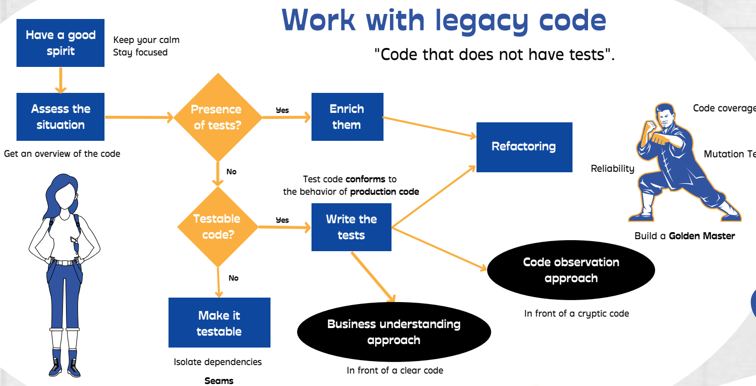
Tests = pre-requisites

Renaming
Extract
Duplication
Magic values
Move
Invert lines
Loop rewriting
Change signature
Conditions rewriting
Inlining

Refactor the test code as well

Work with legacy code

"Code that does not have tests".



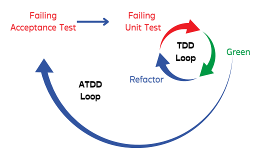
"Test the shallowest branches of the code first, refactor the deepest branches first" - Sandro Mancuso

Advanced TDD Tools and Techniques

Inside-out
Bottom-up
Chicago school
Develop from leaf features

Outside-in
Top-down
London school
Mockist approach
- Focus on user need
- Substituted dependencies

ATDD and double loop



Acceptance Test Driven Development

Principles and Tools for Effective Testing

Create intermediate layers

"Only mock code you own"

Test Doubles

- Analogous to the movie world
- Ensures repeatability

Test Double
Dummy Object
Puppet object
Test Stub
Imitate the desired behavior for a test
Test Spy
Ensure number of calls, args, ...
Mock Object
Pre-programmed object
Checks expectations are met
Fake Object
True implementation

F.I.R.S.T

Fast
Incentive to run tests

Repeatable
- Infinitely executable
- On different environments

Isolated/Independent
Do not depend on each other

Self verifying
Success or Failure

Timely/Thorough
- Timely and accurate
- Covers only 1 use case

Integration Tests

Narrow
- Limit to a few components
- Substitute some dependencies

Broad
- Use real instances
- DB for example

Design techniques

SOLID Principles
Tell don't ask
Hollywood Principle
Don't call us, we'll call you
Hooks / Callbacks
Demeter's law
Only talk to immediate friends
Functional approach
Composition over inheritance
Low coupling high cohesion
Dependency from specific to generic
Fluent API
Builder
...

- Use only one level of indentation per method
- Don't use the **else** keyword
- Wrap all primitives and strings
- Use only one dot per line
- Don't abbreviate
- Keep all entities small
- Don't use any classes with more than two instance variables
- Use first-class collections
- Don't use any getters/setters/properties

Object calisthenics

Beyond practices, a Mindset

Develop your skills

- Progression through repetition: kata
- Organise your tech watch
- Transmission / Companionship
    - Coach the less experienced

Cultivate your soft skills

- Values: humility, benevolence, sharing
- Egoless programming

Demonstrating professionalism

- Pragmatism
- Love for a job well done
- Negotiation / compromise

Introduce the Craft in your Context

Interact with others
Participate in meetups
Videos / Podcasts
Progress kata after kata
FizzBuzz
Tennis
RomanNumerals
Gilded Rose
TripService, ...

Individuellement

Within the organisation

Initiate a dynamic → Supported by sponsors

1 workshop per week
- Code kata
- Watch video
- Motivated people

No support needed

Intensify/generalize
Reaching out to those at a distance

Action examples
- Craft skills in job descriptions
- Recruit enthusiastic developers for craft
- Set aside time to practice
- Recruit / involve craft coaches
- ...

Convince sponsors

Economical
- Cycle time
- Reduction in the number of defects
- Frequency of deliveries
- Speed of remediation

Cultural evolution
Develop teams around current issues

Benchmarking
Compare yourself to others on the craft

Employer branding
- Help to recruit
- Retain skills

#sharingiscaring

by Yoan THIRION

@yot88

SOFTWARE CRAFT

TDD, Clean Code et autres pratiques essentielles

Cyrille Martraire
Arnaud Thiéfaine
Dorra Bartaguiz
Fabien Hiegel
Houssam Fakih

Développer ?

"Working code is a low bar."

- **Un logiciel n'est jamais fini** — Il change en permanence
- **C'est lire et comprendre le code** — Au moins autant que l'écrire
- **Définir le besoin** — Aussi difficile que d'écrire le code correspondant
- **Pas de hacker** — Ni de virtuose

Une communauté

Bienveillante, Aider, Transmission

Pas d'élitisme

Boîte à outils

Test-Driven Development, Behavior-Driven Development, Clean Code, Domain-Driven Design, Legacy remediation, Pair / Mob Programming, OO, FP, SOLID

Développement dirigé par les Tests (TDD)

Ecrire 1 test qui échoue

Améliorer le code — Meilleure lisibilité

Faire passer le test — le plus rapidement possible

Réduire les peurs — Décomposer le problème

Prendre le temps de rendre le code lisible — Refactoring

Aide à définir 1 objectif à atteindre

Ecrire le moins de code possible

Nos tests

- Retranscript la règle de gestion
- Facile de déterminer cause de l'échec

Should / When
Given / When / Then

```
DivideShould.Throw_an_invalid_operation_when-denominator_is_zero
```

Les 3 règles - Uncle Bob

- On doit écrire 1 test qui échoue avant d'écrire n'importe quel code de production
- On ne doit pas écrire plus de tests que ce qui est nécessaire pour échouer (ou ne pas compiler)
- On ne doit écrire que le code suffisant pour que le test actuellement en échec réussisse

Programmer **efficacement** des fonctionnalités **complexes**

Techniques et principes de propreté de code

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." – Martin Fowler

1 exercice de communication

Dégradation dans un environnement urbain

Si pas réparée : entraine d'autres dégradations

Propension au laisser-aller

Face à 1 environnement dégradé

Théorie de la vitre brisée

Règle du boy scout

KISS — Keep It Simple and Stupid

YAGNI — You Ain't Gonna Need It

Être **exemplaire**

4 règles pour 1 design simple - Kent Beck

Passer les tests — fonctionner comme prévu

Révéler l'intention — faire preuve de clarté

Eviter la duplication — DRY

Rester petit — ne conserver que ce qui est important

Mettre l'accent sur le nommage

Exprimer l'intention
- Expliquer ce qu'on cherche à faire
- Pourquoi on veut le faire

s'appuyer sur le métier

Noms communs, Verbes, Renommage

Commentaires avec parcimonie

Comment exprimer la même connaissance à travers le code ?

Souvent 1 code smell — Souci de propreté ?

"Don't comment bad code - rewrite it." - Brian Kernighan

Signaler une subtilité — Optimisation par exemple

Marquer des problèmes — TODO / FIXME

Mentions légales

Découper de façon à faciliter découverte + navigation

S'applique également aux tests
- La qualité des test doit être exemplaire
- Si mettre autant d'effort voire plus

Storytelling
- Notre code doit raconter une histoire
- Devrait se parcourir comme une table des matières

Bien formater son code
- Respect du standard défini par l'équipe
- Niveau maximal d'indentation limité (2 max)
- Lignes de code pas très larges

Découper les fonctions
- Se focaliser sur le service rendu
- Ne doit faire qu'une seule chose
- Peu de paramètres
- 1 seul niveau d'abstraction

Simplicité contractuelle
- 1 seul paramètre en sortie
- Encapsuler in/out complexe dans 1 type dédié
- Considérer chaque paramètre comme immuable
- Définir les variables au plus près de leur utilisation

Spécifications agiles avec le développement dirigé par le comportement (BDD)

3 Amigos

Discuter des fonctionnalités à construire

Représente le besoin — Recherche de valeur

Représente la mise en oeuvre — Faisabilité technique

Challenger les 2 autres — Identifier failles de raisonnement — Problèmes dans l'expression des besoins

Exemples concrets

Dans le langage du métier

- Exprimer le besoin
- Détails concrets
- Auto-suffisants

Identifier scénarios-clés

LIVE STREAMING — D'incréments de spécifications — Ateliers brefs — 30 min / jour

Automatisation

Syntaxe Gherkin

Dev — Critères d'acceptation

Ultérieurement — Tests de non régression

Bénéfices

Critères d'acceptation — pour déterminer l'avancement des développements

Compréhension partagée entre tous

Tests de non régression — avec bonne couverture fonctionnelles

Documentation vivante — évolutive

```
Feature: Offrir des remises aux clients fidèles
  Afin d'encourager les clients à revenir acheter,
  En tant que vendeur de livres,
  Je souhaite offrir des remises de prix aux clients fidèles
Example: Aucune remise en deçà de 50 EUR d'achat
  Given un panier avec 2 articles pour un total de 49 EUR
  When le client consulte son panier,
  Then le total n'inclut aucune remise
```

"Having conversations is more important than capturing conversations is more important than automating conversations" – Liz Keogh

Améliorer **l'efficacité** de la **collaboration** entre les spécialistes impliqués afin de construire de meilleurs logiciels au meilleur coût.

Collaborer efficacement avec le pair / mob programming

Binômage (Pair Programming)

Alternance fréquente — Savoir ralentir

Pilote (Driver)
- Celui / Celle au volant
- Contrôle la machine

Copilote (Navigator)
- Agit pas directement sur la machine
- Prend du recul

Pourquoi ?

Apprendre et progresser

Partager et transmettre — Réduire le Truck Factor

Assurer et se rassurer

Se motiver / s'entraider

Plusieurs styles

Ping-Pong
- TDD + Pair
- Alterner à chaque phase rouge

Strong Style
- "J'ai une idée"
- "Voici le le clavier"

Silent — Seul le code pour communiquer

Evil Twin
- Piéger son partenaire
- Mettre en avant les cas limites

Respect du partenaire — Transparence — Objectif partagé — Rétrospective — Faire des pauses — Agenda — Savoir dire non

QUICK TIPS

Silence — Différentes machines — Rôles figés — Désengagement

Ensemble / Mob programming

Collaboration simultanée de chaque membre

"Tous les esprits brillants travaillent sur la même chose, en même temps, au même endroit et sur le même ordinateur" - Woody Zuill

Travailler avec du code legacy

"Un code ne disposant pas de tests"

Avoir le bon état d'esprit — Garder son calme, Rester focus

Réaliser état des lieux — Se faire une idée générale du code

Présence de tests ? — Oui → Enrichir les tests → Procéder au refactoring

Non → Code testable ?

Code des tests **se conforme** au comportement du **code de production**

Code testable ? — Oui → Ecrire les tests → Procéder au refactoring

Non → Rendre le code testable — Isolation des dépendances — **Seams**

Ecrire les tests → Approche par observation du code — Face à un code cryptique

Ecrire les tests → Approche par compréhension métier — Face à un code plutôt clair

Couverture, Mutation Testing, Fiabilité

Construire 1 **Golden Master**

"Tester en premier les branches les moins profondes du code, refacturer en premier les branches les plus profondes" - Sandro Mancuso

Techniques de refactoring

"Consiste à retravailler un code source sans en modifier le comportement, afin d'en améliorer la lisibilité et de le rendre beaucoup plus facile à maintenir / accueillir des évolutions"

Antidote de la dette technique
- Ne pas la laisser s'accumuler
- Pratique régulière et continue

Des freins
- Pas de culture de la qualité du code dans l'équipe
- Peur de provoquer des régressions
- 1 poste de coût du point de vue client
- Manque de connaissance savoir-faire

Comment ?

Transformer par petits pas

Tests = pré-requis

Refactorer aussi le code de test

Inversion de lignes, Renommage, Réécriture de boucles, Extraction, Changement de signature, Déplacement, Réécriture conditions, Inlining

Principes et outils pour tester efficacement

Doublures de tests

- Analogie au monde du cinéma
- Permettent de garantir la répétabilité

Test Double

Dummy Object — Objet fantoche

Test Stub — Imiter le comportement souhaité pour un test

Test Spy — S'assurer nombre d'appels, args, ...

"Only mock code you own"

Mock Object — Objet préprogrammé — Vérifie attentes bien réalisées

Fake Object — Vraie implémentation

Créer couches intermédiaires

F.I.R.S.T

Fast
- Rapide
- Inciter à lancer les tests

Répétable
- Exécutable à l'infini
- Sur différents environnements

Isolated/Independent — Ne pas dépendre les uns des autres

Self verifying — Succès ou échec

Timely/Thorough
- Opportun et précis
- Couvre qu'un 1 seul cas d'utilisation

Integration Tests

Narrow
- Limiter à quelques composants
- Substitue certaines dépendances

Broad
- Utilisent des instances réelles
- DB par exemple

Outils et techniques avancées de TDD

Bottom-up — Chicago school — **Inside-out** — Développer par les fonctionnalités feuilles

Top-down — London school — Mockist approach — **Outside-in**
- Accent sur le besoin utilisateur
- Dépendances substituées

ATDD et double boucle

Failing Acceptance Test → Failing Unit Test → TDD Loop → Green → Refactor → ATDD Loop

Acceptance Test Driven Development

Techniques de conception

Principes SOLID

Tell don't ask

Principe d'Hollywood — Don't call us, we'll call you — Hooks / Callbacks

Loi de Demeter — Ne parlez qu'à vos amis immédiats

Approche fonctionnelle

...

Composition over inheritance

Faible couplage forte cohésion

Dépendance du spécifique vers le générique

API fluente

Builder

- 1 seul niveau d'indentation par méthode
- Ne pas utiliser le mot clé **else**
- Encapsuler tous les types primitifs et chaines de caractères
- Créer des classes encapsulant les collections
- Un seul point par ligne
- Ne pas abréger
- Garder toutes les entités petites
- Pas de classes avec plus de 2 attributs
- Pas de getter ni de setter

Object calisthenics

Au delà des pratiques, un état d'esprit

Développer son savoir-faire

- Progression par la répétition : kata
- Organiser sa veille
- Transmission / Compagnonnage
  - Accompagner les moins expérimentés

Cultiver son savoir-être

- Valeurs : humilité, bienveillance, partage
- Egoless programming

Faire preuve de professionnalisme

- Pragmatisme
- Amour du travail bien fait
- Négociation / compromis

Introduire le craft dans votre contexte

Echanger avec autres

Participer à des meetups

Vidéos / Podcasts

Progresser kata par kata — FizzBuzz, Tennis, RomanNumerals, Gilded Rose, TripService, ...

Individuellement

Dans l'organisation

Initier une dynamique → Porter par des sponsors

1 atelier par semaine
- Code kata
- Visionner vidéo
- Personnes motivées

Nécessite pas de soutien

Intensifier / généraliser — Atteindre personnes à l'écart

Exemples d'action
- Compétences craft dans fiches de poste
- Recruter dévs enthousiastes sur le craft
- Réserver du temps pour pratiquer
- Recruter / faire intervenir des coachs craft
- ...

Convaincre les sponsors

Economique
- Temps de cycle
- Réduction du nombre de défauts
- Fréquence de livraisons
- Vitesse de remédiation

Evolution culturelle — Faire évoluer les équipes vers des enjeux d'actualité

Benchmarking — Se comparer aux autres sur le craft

Marque employeur
- Aider à recruter
- Conserver des compétences

SOFTWARE DESIGN X-RAYS

Fix Technical Debt With Behavioral Code Analysis **by Adam Tornhill**

Technical Debt

- Explain the need for refactorings
- Communicate technical trade-offs

Apply at all levels (Micro and Macro)

Interest Rate Is a Function of Time

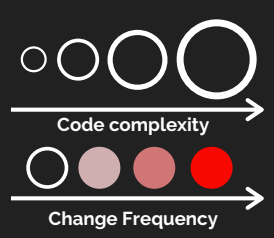
Bad Code is Technical Debt if you have to PAY INTEREST ON IT

Identify Code with High Interest Rates

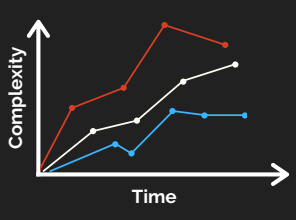
Prioritize Technical Debt with Hotspots

Complicated code that you have to work with often

- Change frequency of each file
- Lines of code as a simple measure of code complexity



Hotspot



Evaluate Hotspots with Complexity Trends

- Complexity : indentation-based complexity
- Language agnostic

X-Ray analysis

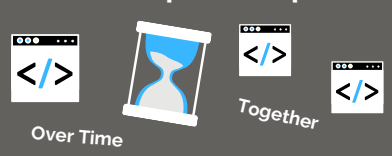
Prioritized list of function to :

- Inspect
- Possibly refactor

Coupling in Time - A Heuristic for the Concept of Surprise

Change coupling - 2 (or more) files change

- Invisible in the code itself
- Mine it from code's history and evolution



Is and Isn't Temporal Coupling (ex : Unit Tests)

Neither good nor bad all depends on context

"Change coupling can help us design better software as we uncover expensive change patterns in our code"

Refactor Congested Code with the Splinter Pattern

Break a hotspot into smaller parts

- Along its responsibilities
- Maintaining the original API for a transient period

"Parallel Development Is at Conflict with Refactoring"

How to ?

1. Ensure tests cover the splinter candidate
2. Identify the behaviors inside your hotspot
3. Refactor for proximity
4. Extract a new module for the behavior with the most development activity
5. Delegate to the new module
6. Perform regression tests
7. Select the next behavior to refactor and start over at 4

Stabilize Code by Age

- Organize our code by its age
- Turn stable packages into libraries
- Move and refactor code we fail to stabilize

- Promotes long-term memory models of code
- Less cognitive load : less active code
- Prioritizes test suites to shorten lead times

"Always remember that just because some code is a hotspot, that doesn't necessarily mean it's a problem."

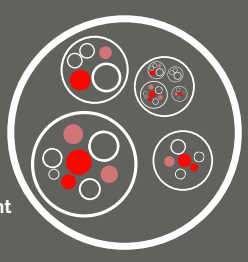
Divide and Conquer with Architectural Hotspots

Identify your architectural boundaries :

Often based on the folder structure of the codebase

Hotspot analysis on an architectural level :

- Identify the subsystems with the most development effort
- Visualize the complexity trend of a whole architectural component



Analyze the files in each architectural hotspot

Fight the Normalization of Deviance

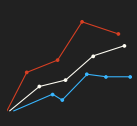
- Each time you accept a risk, the deviations become the new normal
- Complexity trends as WHISTLEBLOWERS

"The more often something is changed the more important it is that the corresponding code is of high quality so all those changes are simple and low risk"

Communicate with Nontechnical Managers - Data buys trust

% of commits involving top hotspots

- Demonstrate importance of this code
- Support new features and innovations



Show complexity trends

- Code gets worse over time
- Which will slow us down

Coordination bottlenecks

- Add people side to the presentation

Beyond Conway's Law

Quality Suffers with Parallel Development

Increases risk of defects with the number of developers

Coordination needs

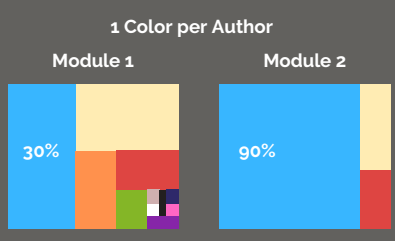
Number of authors behind each component

Rank Code by Diffusion

Calculate a fractal value

- How many different authors have contributed
- How the work is distributed among them

0 : Single author
1 : the more contributors there are



Module 1 : Many minor contributors

Higher risk for defects

Module 2 : 1 main developer

Reduced risks

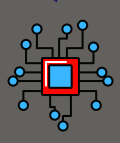
"Ranks all the modules in our codebase based on how diffused the development effort is"

Use Fractal Values to

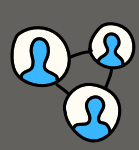
Prioritize code reviews
Done right = a proven defect-removal

Focus tests
Identify the areas to focus extra tests

Replan suggested features
If high developer congestion



Redesign for increased parallelism
Candidate for splinter refactorings ?



Introduce areas of responsibility
introduce teams aligned with the structure of the code

Use Social Data

Fight motivation losses in Teams

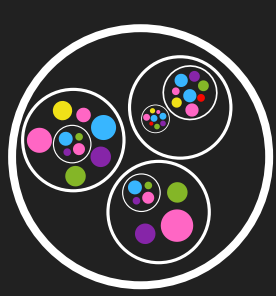
Evaluation
Someone else cares about your contribution

Lead by example
Model the behaviors you want to see in others

Visibility

- Recognize contributions
- Present knowledge maps

Small Groups



Knowledge Map
Main Author / Module

Guide On and Off-boarding

Identify the Experts
Find out who to communicate with

Measure Future Knowledge Loss

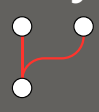
React to Knowledge Loss
Focus to maintain knowledge

Biases and Workarounds for Behavioral Code Analysis

Data
Minimum amount of data

Incorrect author info
Need a minimum amount of data

Copy-paste repositories
Fails to migrate its history



Misused squash commits
When applied to work committed by several individuals

Discover it in action on codescene.com **#sharingiscaring** **by Yoan THIRION** **@yot88**

TEAM TOPOLOGIES

by Matthew Skelton and Manuel Pais

TEAM AS THE MEANS OF DELIVERY

Team assignments
First draft of the architecture

Inverse Conway manoeuvre
Organize teams to match the architecture you want

- Not all communication / collaboration is good
- Restrict communication between teams
- Focus communication between specific teams

> *"Disbanding high-performing teams is worse than vandalism: it is corporate psychopathy."*
> — Allan Kelly, Project Myopia

TEAM FIRST-THINKING

Dunbar's number
Seven-to-nine MAX
> Trust will break down

Owns the Software
"Continuity of care"
No shared ownership

Embrace Diversity
Produce more creative solutions

Use Small, Long-Lived Teams
As the Standard
Autonomous

Minimize Team Cognitive Load
Total amount of mental effort used in the working memory
Use good boundaries

Reward the Whole Team
Not individuals

TEAM TOPOLOGIES THAT WORK FOR FAST FLOW

STREAM-ALIGNED TEAM

Team aligned to a single valuable business stream of work

Product or service
Set of features
User Journey
User Journey
User Persona

Primary type in an organization
(80/90 %)

- Work on the full spectrum of delivery
- Requires clarity of purpose and responsibility

"Purpose of the other fundamental team topologies is to reduce the burden on the stream-aligned teams."

ENABLING TEAM

Help stream-aligned teams acquire missing capabilities

Not a permanent dependency

Composed of specialists
In a given technical or product domain

Collaborative nature
Focus on stream-aligne teams problems first
Not the solutions per se

"Do not exist to fix problems that arise from poor practices, prioritization choices, or code quality within stream-aligned teams."

COMPLICATED SUBSYSTEM TEAM

Reduce cognitive load of stream-aligned teams that needs to use the complicated subsystem

Responsible for building / maintaining
A part of the system
That depends heavily on specialist knowledge

Examples : Video processing codec, Mathematical model, Real-time trade, Reconciliation algorithm, Face-recognition, ...

"Prioritizes and delivers upcoming work [...] respecting the needs of the stream-aligned teams that use the complicated subsystem."

PLATFORM TEAM

Provide internal services to reduce cognitive load of stream-aligned teams

Treat services as products
Reliable / Usable
Fit for purpose

Thick platform
Combination of several inner platform teams
Providing a myriad of services

VS

Thin platform
Could simply be a layer on top of a vendor-provided solution

Provision new server instance
Provide tools for access management

"A digital platform is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product."

Convert Common Team Types to the Fundamental Team Topologies

> *"Most organizations would see major gains in effectiveness by mapping each of their teams to one of the four fundamental topologies [...] to adopt the purpose and behavior patterns of that topology."."*

Infrastructure Teams → PLATFORM TEAM
Tooling Teams → ENABLING TEAM Or PLATFORM TEAM
Component Teams → PLATFORM TEAM Or other
Architecture → ENABLING TEAM Part time

Split with FRACTURE PLANES

Software boundaries
Natural Seam
Allowing the system to be split easily

User Personas
Technology
Change Cadence
Regulatory Compliance
Team Location
Risk
Performance Isolation

Business Domain Bounded Context

EVOLVING TEAM INTERACTIONS FOR INNOVATION AND RAPID DELIVERY

3 INTERACTION MODES

"Well-Defined Interactions Are Key to Effective Teams"

Interaction patterns per topology	Collaboration	X as-a-Service (XAAS)	Facilitating
	2 teams work together On a shared goal During discovery of new technology or approaches	1 team consumes something Provided by another team Such as an API, a tool, or a full software product	1 team facilitates another team Learning / adopting new approach (usually an enabling team)
STREAM-ALIGNED TEAM	Typical	Typical	Occasional
ENABLING TEAM	Occasional		Typical
COMPLICATED SUBSYSTEM TEAM	Occasional	Typical	
PLATFORM TEAM	Occasional	Typical	

EVOLUTIONARY PATTERNS

Collaboration → X as-a-Service
X as-a-Service → Collaboration
Facilitating → No Interaction

Teams should ask

What kind of interaction should we have with this other team ?
Should we be collaborating closely with the other team?
Should we be expecting or providing a service?
Or should we be expecting or providing facilitation?

How to get started ?

1. Start with the Team
2. Identify Suitable Streams of Change
3. Identify a Thinnest Viable Platform (services needed)
4. Identify Capability Gaps (Team Coaching, Mentoring,...)
5. Share and Practice Different Interaction Modes
   Explain Principles behind New Ways of Working

Team Topologies alone : not enough

IN ADDITION

Healthy organizational culture
Supports professional development of individuals and teams
Safe to speak
Learn continuously

Good engineering practices
Test-first development
Focus on continuous delivery / operability
Pairing / mobbing for code review ...

Healthy funding / financial practices
Avoiding the pernicious effects of a CapEx/OpEx
Avoiding project-driven deadlines and large-batch budgeting
Allocating training budgets to teams or groups rather than individuals

Clarity of business vision
With horizons at human-relevant timescales
Clear reasoning behind the priorities

#sharingiscaring

BY YOAN THIRION @YOT88

Robert Waldinger, MD

The Good Life

Marc M.D. Schulz, PhD

Ce que nous apprend la plus longue étude scientifique sur le bonheur et la santé

1938

2 générations

1300 descendants des 724 participants initiaux

L'étude de Harvard sur le développement des adultes

Une étude "longitudinale"
examiner des vies à travers le temps

"Prospective"
interroger les participants sur leur vie telle qu'elle est

Identifier ce qui compte pour la santé et le bonheur

- Quels investissements en valaient vraiment la peine ?
- Ce qui maintient les personnes heureuses et en bonne santé ?

Questionnaires
- Qu'est-ce qui comptait pour cette personne en particulier ?
- Qu'est-ce qui donnait un sens à ses journées ?
- Qu'avait-elle appris de ses expériences ?
- Que regrettait-elle ?
- ...

Entretiens attachés
- Etudier la façon dont les participants parlent l'un de l'autre
- Signaux non verbaux

Examiner leur bien-être
- Scanners cérébraux
- Analyses de sang
- Echantillons capillaires
- Poids
- Activité physique
- ...

Autres données
- Nature de leur emploi
- Nombre d'amis proches ...

Qu'est-ce qu'une vie réussie ?

1 vie réussie c'est 1 vie compliquée
- Pour tout le monde
- Se forge à partir de ce qui la rend difficile

Le secret = la qualité des relations
Permettent de vivre plus heureux et en meilleure santé

De l'importance des relations

Expérience de prévision affective
Imaginer un état émotionnel dans une situation future

Parler à un inconnu
VS
Rester dans son coin

Meilleur trajet en parlant à un inconnu

Des inconnus dans un train

Mauvais en prévision
- Éviter les complications de la relation à autrui
- Surestimer les complications
- Sous-estimer les effets bénéfiques du lien humain

Culture : prédicteur de bonheur ?
- Des injonctions culturelles permanentes
- Ex : l'argent est le fondement d'une vie réussie

Parfois, les pratiques et les messages culturels nous éloignent du bien-être et du bonheur

Liens sociaux forts et espérance de vies

148 études examinées
Pays du monde entier

300 000 participants

Taux de mortalité des personnes
ayant le moins de liens sociaux

♂ 2,3 ♀ 2,8

fois plus élevé que celui des personnes en ayant le plus

Etude de Angus Deaton et Daniel Kahneman
Etats-Unis en 2010

Espérance de vie > 10 à 15 ans
Hauts revenus

75 000$ / an chiffre pivot

Chiffre pivot dépassé
- L'argent en plus
- Pas important pour atteindre le bonheur

L'argent fait-il le bonheur?
Pas la bonne question

Qu'est-ce qui me rend réellement heureux ?

> Les relations sont essentielles, mais pas seulement comme des tremplins vers d'autres choses, et elles ne sont pas simplement une route d'accès vers la santé et le bonheur. Elles sont des fins en soi.

Les relations et les virages de la vie

La vie est chaotique

Cultiver de bonnes relations
- Accroît le côté positif de ce chaos
- Augmente ainsi les chances de faire des rencontres bénéfiques

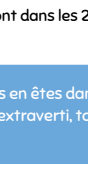
Milieu de vie = point d'inflexion

Entre 1 mode de vie égocentrique
Replié sur soi-même

Et un mode de vie plus généreux
Et tourné vers l'extérieur

Questionnements

Au milieu de notre vie
- Est-ce que je m'en sors bien par rapport aux autres ?
- Ai-je de bonnes relations avec mes enfants ?
- Combien d'années me reste-t-il ?
- Ma vie a-t-elle un sens, pour moi mais aussi pour le monde qui m'entoure ?
- Quelles personnes et quels objectifs me tiennent vraiment à coeur ?
- Qu'est-ce que je voudrais encore faire d'autre ?



À la fin de notre vie
- "Qu'aimeriez-vous avoir moins fait et au contraire davantage fait ?"
- "J'aurais aimé ne pas avoir perdu autant de temps."
- "J'aurais aimé ne pas avoir autant tergiversé."
- "J'aurais aimé ne pas m'inquiéter autant.."
- "J'aurais aimé passer plus de temps avec ma famille."

Les adultes les plus heureux, ceux qui avaient réussi à transformer la question...

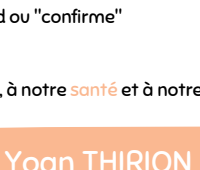
"Que puis-je faire pour moi ?"

"Que puis-je faire pour le monde qui m'entoure ?"

Notre propre étude de Harvard

1) Trouvez une photo de vous
À peu près la moitié de l'âge que vous avez aujourd'hui

2) Regardez-vous de près
- Essayez de vous replacer au moment où la photo a été prise
- Regardez vraiment : passez plusieurs minutes s'en imprégner



- "À quoi pensiez-vous alors ?"
- "Qu'est-ce qui vous inquiétait ?"
- "Qu'est-ce que vous abordiez avec confiance ?"
- "Quels étaient vos projets ?"
- "Avec qui passiez-vous votre temps ?"
- "Qu'est-ce qui était le plus important pour vous ?"
- "Quand vous pensez à cette époque, que regrettez-vous ?"

Prendre du recul de temps en temps.

La forme sociale – Comment muscler sa sociabilité ?

Les ravages de la solitude
- Plus grande sensibilité à la douleur
- Affaiblissement du système immunitaire
- Diminution des fonctions cérébrales
- Sommeil moins réparateur

Ses causes
- Une expérience subjective
  - Ce qui cause un sentiment de solitude chez un individu
  - Peut n'avoir aucun effet sur un autre
- Besoin d'amour, de lien et d'un sentiment d'appartenance
  - Comment répondre à ces besoins ?
  - Dans des environnements sociaux compliqués

La solitude augmente notre risque de mortalité autant que le tabagisme ou l'obésité.

Investir du temps / énergie sur nos relations

Le pouvoir de la générosité
Donner ce qu'on aimerait recevoir

La curiosité radicale
- Permet aux autres de se sentir compris et reconnus
- Créer un environnement bienveillant
- Poser une question, écouter la réponse, voir où cela mène

Spirales ascendantes

Votre observatoire social

1) Qui fait partie de ma vie ?
Qui sont mes amis et parents les plus proches ?

Anime et vous exalte
Donne un sentiment de lien
Permet de se sentir mieux que si vous étiez seul
Stimulant

2) Placez les noms sur le graphique
- Pourquoi est-elle à cette place ?
- À l'endroit où vous voulez qu'elle soit ?
- ...

Rare — Fréquent

3) Allez du haut vers le bas

Haut
Réfléchir comment consolider / favoriser leurs aspects positifs
Dites à ces personnes combien vous les appréciez / pourquoi

Bas
Comment les tirer vers le haut ?

Épuisant
Provoque des tensions
Agacement ou de l'anxiété

Notre attention en ligne

Comment ces nouvelles formes de communication affectent nos relations / notre bonheur ?

- Entretenir les relations avec les amis et la famille
- Renforcer le sentiment de connexion et d'appartenance
- Lutter contre l'isolement

- Expérience sensorielle / émotionnel manquante
- La communication : pas seulement un échange d'informations
- Le toucher / proximité physique
  - Effets émotionnels, psychologiques, biologiques

Arrêtez de cliquer, discutez

1) S'engager dans une communication active avec les autres
Les gens qui s'engagent sont plus heureux que les "passifs"

2) Après avoir surfé, sondez votre humeur
- Avez-vous l'impression d'avoir fait le plein d'énergie ?
- Vous sentez-vous épuisé après un long voyage à travers Internet ?

3) Comment vos proches perçoivent votre utilisation ?
- Que pense votre conjoint de la façon dont vous utilisez votre téléphone ?
- Vos habitudes en ligne l'affecte-t-il (ou elle) ?
- Qu'en est-il pour vos enfants ?

4) Prenez du temps sans écran
- Éteindre votre smartphone, votre ordinateur
- Pour vous révéler comment la technologie vous affecte

Pleine conscience
- Ramener son esprit au moment présent
- Sans juger, sur l'expérience qui se déploie instant après instant

Chaque jour un peu plus d'attention
- Quoi mettre en œuvre aujourd'hui pour accorder de l'attention à quelqu'un qui le mérite ?
- Plus vous vous intéressez aux autres, plus ils s'intéresseront à vous

> Nous avons un avantage crucial sur tous les géants de la technologie : la conquête de notre attention se déroule sur notre propre terrain, littéralement dans notre esprit. Et c'est là que nous pouvons gagner la guerre.

Braver la tempête – s'adapter aux défis relationnels

Modèle W.I.S.E.R
- ralentir ses réactions dans n'importe quelle situation émotionnelle
- les examiner au microscope

W	I	S	E	R
Observer	Interpréter	Choisir	S'engager	Réfléchir
• Notre "réalité" : pas forcément celle des autres • Essayer de comprendre les enjeux • Qu'est-ce qui m'échappe, ici ?		• S'engager dans la stratégie que vous avez choisie • Soigner sa mise en œuvre • Peut être utile de s'entraîner		
• L'environnement, la personne • La situation est-elle inhabituelle ou courante ? • Quel est votre ressenti, pourquoi ?	• Ralentir permet d'envisager des possibilités • Compte tenu de l'enjeu et des ressources dont je dispose • Que puis-je faire dans cette situation ?		• Comment ça s'est passé ? • Ai-je amélioré les choses ou ont-elles empiré ?	

La personne à vos côtés

Le contact aimant l'équivalent d'un

Expérience de Coan (IRM)
- Tenir la main d'un proche
- Réduit l'activité des centres de la peur
- Diminue l'anxiété
- Réduit l'intensité de la douleur

Flagrant délit de gentillesse

Dernière chose pour laquelle vous lui êtes reconnaissant ?

- Noter ce petit geste
- Tenir un "journal de gratitude"

Remarquer les petits gestes / s'en souvenir peut avoir un effet positif

Si vous pouviez arrêter de travailler sans perte de revenus, le feriez-vous ?
Que feriez-vous à la place ?

Bien vivre au travail

> Et si la valeur du travail – même un travail que nous n'aimons pas – ne résidait pas seulement dans notre salaire, mais dans le sentiment d'être vivant qui nous saisit par moments et dans la vitalité que nous procure le contact avec les autres sur notre lieu de travail ?

Tous les amis ont leurs avantages

L'amitié diminue notre perception de l'adversité :
- Épreuves apparaissent moins stressantes que si nous étions seuls
- Diminue l'impact et la durée d'un stress extrême

Ne pas négliger l'amitié
- Écouter ses amis
- Être écouté donne le sentiment d'être compris / recevoir de l'attention
- Les amitiés les plus solides vont dans les 2 sens

Longue portée des liens "faibles"

Des personnes vers lesquelles nous ne nous tournerons probablement pas en cas de problèmes

Procurent des bouffées de bien-être / d'énergie

Sentiment d'appartenance à une communauté plus large

Circulation d'idées différentes

Circulation d'informations / d'opportunités

> Peu importe votre âge, peu importe où vous en êtes dans le cycle de vie, peu importe que vous soyez marié ou non, introverti ou extraverti, tout le monde peut prendre des tournants positifs dans sa vie.

Il n'est jamais trop tard pour être heureux

Lire, écrire, compter, et ... relations
- But de l'éducation : préparer les élèves à la vie
- Serait judicieux d'y ajouter un quatrième élément : les relations

Ce que la science nous apprend ou "confirme"

Les relations de qualité contribuent à notre bonheur, à notre santé et à notre longévité.

#sharingiscaring

by Yoan THIRION

@yot88

THE SOFTWARE CRAFTSMAN

BY SANDRO MANCUSO

WHAT ?

NOT A RELIGION

NOT A METHOD

WORKING CODE = THE MINIMUM FOR A PROFESSIONAL

GOOD SENIOR DEVELOPER CODE

80'S → NOW

NOBODY UNDERSTAND THE CODE

- CLEAN
- HUMAN READABLE
- DOMAIN LANGUAGE

"CRAFTSMANSHIP OVER CRAP" - ROBERT C. MARTIN

IDEOLOGY

LOWER THE COST OF QUALITY

WHAT MODERN DEVELOPERS DO

- DEVELOP
- TEST
- ANALYZE
- MAKE TECHNICAL CHOICES
- HELP CLIENT
- RECRUIT
- ...

AGILITY

HOW TO BUILD THE RIGHT THING

FOCUS ON THE PROCESS — CUSTOMER CENTRIC

DOES NOT MAKE DEVELOPERS BETTER

CRAFTSMANSHIP

HOW TO BUILD THE THING RIGHT

MINDSET ?

BE PROUD TO BE A DEVELOPER

DEVELOPMENT IS A CRAFT

LEARNING FROM OTHERS

CONSTANTLY SHARING

A LONG JOURNEY TO MASTERY

CARING ABOUT WHAT THEY DO

RESPONSIBILITY/ PROFESSIONALISM / PRAGMATISM / PRIDE

OWN YOUR CAREER VS "PETER'S PRINCIPLE"

"ONLY INCOMPETENT PEOPLE ARE SCARED TO LOSE THEIR JOB"

FAILURE

ADVANCEMENT

SUCCESS

ADVANCEMENT

MANIFESTO FOR SOFTWARE CRAFTSMANSHIP - 2008

1 NOT ONLY WORKING SOFTWARE, BUT ALSO WELL-CRAFTED SOFTWARE

WELL-CRAFTED = HIGH QUALITY CODE

- AUTOMATED TESTS
- BUSINESS LANGUAGE IN THE CODE
- SIMPLE DESIGN

"CODE QUALITY IS NOT A GUARANTEE OF SUCCESS BUT CAN BE THE MAIN CAUSE OF FAILURE"

2 NOT ONLY RESPONDING TO CHANGE, BUT ALSO STEADILY ADDING VALUE

CONSTANTLY IMPROVE YOUR CODE

- TESTABLE
- EXTENDABLE
- REFACTOR

BOY SCOUT RULE

"ALWAYS LEAVE THE CAMPGROUND CLEANER THAN YOU FOUND IT."

3 NOT ONLY INDIVIDUALS AND INTERACTIONS, BUT ALSO A COMMUNITY OF PROFESSIONALS

SHARE / MENTOR

- KNOWLEDGE
- IDEAS
- SUCCESSES AND FAILURES

CRAFTSMEN WANT TO WORK WITH PASSIONATES & INSPIRING PROFESSIONALS A.K.A OTHER CRAFTSMEN

4 NOT ONLY CUSTOMER COLLABORATION, BUT ALSO PRODUCTIVE PARTNERSHIPS

WE ARE NOT FACTORY WORKERS

- MUST HELP OUR CLIENTS
- MUST SAY NO FOR CLIENTS GOOD

SOME CLIENTS ARE NOT READY : VERY DIFFICULT ENVIRONMENT FOR CRAFTSMEN

REDUCE THE GAP BETWEEN THE AGILE METHODOLOGIES AND THE TECHNICAL WORLD

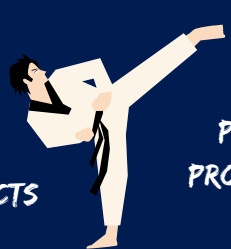
ATTITUDE

PRACTICE / PRACTICE / PRACTICE

PRACTICE THROUGH 才

CODE KATAS

OPEN SOURCE PROJECTS



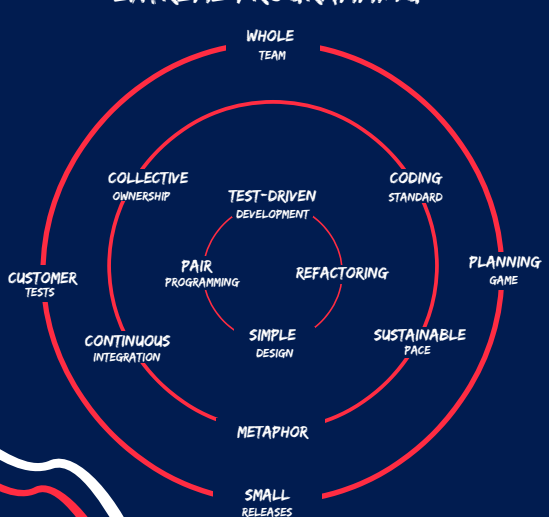
PET PROJECTS

PAIR/MOB PROGRAMMING

DISCOVERY

PRACTICES

EXTREME PROGRAMMING



CONTINUOUS LEARNING

INJECT PASSION

CREATE A CULTURE OF IMPROVEMENT

CREATE A CULTURE OF SHARING

BROWN BAGS

BOOK CLUB

LEAN COFFEES

LIGHTNING TALKS

CODE REVIEWS

COMMUNITIES OF PRACTICE

#SHARINGISCARING

BY YOAN THIRION

@YOT88

TU FAIS QUOI DANS LA VIE ?

par Joséphine Bouchez / Matthieu Dardaillon

80 000 HEURES
Temps moyen de nos vies à travailler

QU'ALLONS-NOUS EN FAIRE ?

L'URGENCE D'AGIR

Notre système actuel n'est pas durable
Fondé sur l'utilisation croissante de ressources naturelles présentes en quantités limitées

Jour du dépassement
(le jour où l'on consomme plus que ce la Terre est capable de produire en une année)

- **1970 : 29 décembre**
- **1990 : 7 décembre**
- **2019 : 29 juillet**

> *"Quand le dernier arbre aura été abattu, quand la dernière rivière aura été empoisonnée, quand le dernier poisson aura été pêché, alors l'Homme s'apercevra que l'argent ne se mange pas"*
> *- Geronimo*

1 continent de plastique
Plus de 1.6 million de km2 flotte dans l'Océan Pacifique

Chute des populations d'oiseau
Les populations d'oiseau ont chuté d'1/3 en 15 ans

26 personnes
Autant d'argent que la moitié la plus pauvre de la population mondiale

300 000 SDF en France

"Qui pourra assumer face aux générations futures que nous avons cautionné et laissé faire ?"

Notre responsabilité ?

Nous sommes ce système

Chacun un rôle à jouer
Est-ce que je contribue activement à construire la société dans laquelle j'aspire à vivre ?

Quelles sont les causes qui vous touchent ?

- **Agriculture / alimentation** — Permettre à tous de manger sainement
- **Education** — Permettre une éducation de qualité et accessible
- **Santé** — Permettre l'accès à des soins de qualité pour tous
- **Energies** — Permettre l'accès à une énergie propre
- **Habitat** — Permettre à chacun de se loger dignement
- **Inclusion & lien social** — Permettre à chacun de trouver une place dans la société
- **Environnement** — Préserver l'environnement, la nature, la biodiversité
- ...

LE LEVIER DU TRAVAIL

"Nous avons besoin de tous les métiers dans tous les secteurs"

Travail
Semaine
Pour gagner sa vie

L'engagement
Soir / Week-end
"Quand on a le temps"

Intention
"j'ai chois cette activité dans le but d'être utile à la société"

Impact
"les conséquences de mon travail ont un impact positif sur la société"

Des vies scindées

réconcilier les 2 avec

Engagement
"le temps que je dédie à ces activités représente au minimum 50% de mon temps et de ma rémunération"

Décisions
Critères d'impact (social, environnemental) ont au moins autant de poids que les critères économiques

LES CARRIÈRES À IMPACT

> *"Chacun doit se demander si son travail perpétue ou non le problème. Dépenser toue son énergie pour une entreprise qui pollue et faire des "petits gestes" à la maison pour compenser, c'est être un pompier pyromane. C'est un peu comme dire : je travaille chez Monsanto mais j'y vais à vélo"*
> *- Cyril Dion"*

Baisse MAX de notre empreinte carbone individuel
Si chaque français adopte un comportement héroïque — 25%

Etat + Entreprises — 75% — **NOUS SOMMES AUSSI L'ETAT ET LES ENTREPRISES**

CHANGER LE SYSTÈME

> *"Nous ne pouvons pas résoudre nos problèmes avec la même pensée que celle que nous avons utilisée lorsque nous les avons créés"*
> *- Albert Einstein*

Education / formation
- Pas inviter à identifier nos talents
- Comment les mettre au service des enjeux de société ?

L'économie
- L'économie dirige le monde
- Possible de créer de la **valeur** :
  - **économique, écologique ET sociale**

L'emploi
- 85% des emplois de 2030 n'existent pas encore
- **à nous de les créer**

Pouvons-nous encore accepter que l'économie se développe sans prendre en considération les limites de notre planète ?

Corrélation inverse entre rémunération et utilité sociétale...

5 obstacles
Pour s'engager dans une carrière à impact

Salaire
Prestige
Sécurité
Ce que fait notre entourage
Coût des études
...

L'orientation
- Pas encouragés à nous engager dans des carrières à impact
- Choix d'orientation = souvent fait par défaut

Quid de valoriser l'utilité sociétal et l'épanouissement ?

La rémunération
- Quels sont mes besoins ?
- De combien ai-je besoin pour être heureux aujourd'hui et préparer un futur désirable ?
- Qu'est ce qui me parait juste par rapport à la structure ?

> *"Ne doutez jamais qu'un petit groupe d'individus puisse changer le monde. En réalité c'est toujours comme cela qu'il a changé"*
> *- Margaret Mead*

PASSER À L'ACTION

Le plus ce n'est pas d'avoir envie, c'est de sauter le pas

3 stratégies

Transformer votre emploi actuel
- **Réduire** les impacts négatifs de votre activité
- **Connectez**-vous à d'autres pionniers
- **Formez** vous
- **Transformez** pas à pas
  Montrer la valeur que cela apporte à votre organisation

Trouver un nouvel emploi
- Identifiez vos talents / compétences
- Comment les utiliser différemment ?

Créer votre emploi

Quelque soit la thématique
Trouvez un problème de société urgent, important, qui n'a pas d'alternative satisfaisante

Entourez-vous
Essentiel de travailler avec les bonnes personnes

Soyez obstinés par votre problème pour trouver la meilleure solution possible

UNE VIE SUR NOTRE PLANÈTE

David Attenborough

Déclin accéléré de la biodiversité
Véritable tragédie de notre temps

Nous sommes tous coupables
- "Ce n'est pas notre faute"
- Nous sommes nés dans un monde humain qui n'est pas durable

Continuer
De vivre notre existence heureuse en ignorant la catastrophe à nos portes

Changer

Nous devons faire 1 choix

Encore temps d'arrêter le réacteur
Il existe une alternative viable

MON TÉMOIGNAGE

	en milliards	en Parties Par Million de Molécules d'air	monde **sauvage** subsistant	Observations
1937	2,3	280	66%	• L'agriculture a changé notre rapport entre l'humanité et la nature. • Apprivoisement d'une partie du monde sauvage.
1954	2,7	310	64%	• Émission Zoo Quest • Nature sauvage florissait. • Personne n'avait conscience des problèmes qui se posaient déjà.
1960	3	315	62%	• Comprendre le fonctionnement global de l'écosystème du Serengeti. • Histoire d'interdépendance / écologie.
1989	5,1	353	49%	Le monde compte trois trillions d'arbres de moins qu'au début de la civilisation humaine.
1997	5,9	360	46%	• L'humanité avait éliminé 90% des gros poissons dans tous les océans. • Prélève les poissons au sommet de la chaîne trophique
2011	7	391	39%	Température moyenne de 0,8°C plus chaude qu'en 1926
2020	7,8	415	35%	Notre impact est vraiment mondial...

> 80 millions de tonnes (par an)
de fruits de mer prélevés

Perdu la moitié des coraux d'eau

15 milliards d'arbres abattus
par an

2020

30% des réserves de poissons
à un niveau critique

Nos débris plastiques sont partout
- > 90% des oiseaux de mer ont des fragments de plastique dans l'estomac
- Aucune plage de la planète n'échappe à nos ordures

m(*humain*) + m(*élevage*) = 96% m(*animaux*)
élevage : animaux que nous élevons pour les manger

« Nous avons remplacé le monde sauvage par un monde apprivoisé.
Nous considérons la Terre comme NOTRE planète, gouvernée par l'humanité, pour l'humanité. »

CE QUI NOUS ATTEND

Monde du vivant en passe de s'effondrer
a déjà commencé à s'effondrer

Dégradation de la couche d'ozone
Changement climatique
Acidification des océans
Pollution atmosphérique
5 Limites planétaires dépassées
Pollution chimique
Erosion de la biodiversité
Usage d'engrais
Changement d'utilisation des sols
Consommation d'eau

« Nous vivons déjà hors de l'espace de fonctionnement sécurisé de notre planète »

2030
- **-75%** de la surface de la forêt amazonienne
- Pôle Nord : été libre de glace

2040
- Pergélisol fondu : 1400 GT de carbone stocké
- Glissements de terrains / inondations gigantesques

2050
- Acidité très élevé des océans
- Commencement de la fin pour la pêche

2080
- Engrais : sols stériles er épuisés
- Déclin des espèces d'insectes
- Affecter les 3/4 de nos cultures

Migrations forcées
de populations

+0.9 m
du niveau de la mer

+4°C
Température de la Terre

2100

1/4
de l'humanité vivra > **29°C**

« Pour lui rendre sa stabilité,
nous devons restaurer sa biodiversité.
Nous devons réensauvager le monde ! »

Fin de la **stabilité** de **l'Holocène**
(notre jardin d'Eden)

6ème extinction massive

UNE VISION POUR L'AVENIR

"Comment réensauvager le monde"
Nous faisons partie de la nature MAIS nous nous sommes séparés d'elle

Monde limité
rien ne peut grandir indéfiniment

Dépasser la croissance

Nous avons tout pris au vivant
sans songer aux dégâts

Construire 1 modèle économique durable

3 P

Personne **Profit** **Planète**

Croissance verte
Sans impact négatif sur l'environnement

- Abandon du PIB comme critère principal du succès
- 3 P au complet
- Déplacer les priorités de son pays tout entier

« Si notre principal critère pour juger nos actions est la renaissance du monde naturel nous ne pourrons manquer de prendre les bonnes décisions »

Passer à l'énergie propre

Budget carbone
montant réduit de carbone pouvant être rejeté

Mettre fin à notre dépendance aux combustibles fossiles

Elever le prix des émissions
au niveau mondial

Taxe carbone

Accélérerait la révolution durable dont nous avons besoin

Réensauvager les mers
Ex : Cabo Pulmo

Créer des zones interdites à la pêche
Permet aux poissons de devenir plus vieux et plus gros

Pêche durable à long terme
1/3 des océans en zones sans pêches suffirait

Favoriser la pêche durable
Les entreprises la pratiquant

Occuper moins d'espace

Une grande partie dépourvue de bétail

Culture du soja pour nourrir le bétail

80% de la terre agricole
consacrés à la production de viande / lait

Viandes propres
cultures cellulaires

Protéines alternatives

Remédier au gaspillage
alimentaire

Produire + en cultivant moins de terre

Agriculture régénératrice
revivifier les sols

Agriculture verticale
étages de différentes plantes

Réensauvager les terres

Ramènerait la biodiversité

Recréer des espaces sauvages

Stabiliser la planète

Planifier pic démographique

Entre 9,4 et 12,7 Mds d'humains (en 2100)

Atteindre le pic de croissance
- le plus tôt possible
- avec un minimum d'individus
- 2060 : 8,7 Mds

Elever le niveau d'éducation dans le monde
changerait le cours de la croissance démographique

Vers une vie plus harmonieuse

Vivre de nouveau en harmonie
avec notre environnement naturel

NOTRE PLUS GRANDE CHANCE

De l'Holocène à l'Anthropocène
Ere des êtres humains

Nous devons nous sauver nous-mêmes
- Nous parlons souvent de sauver la planète
- Avec ou sans nous, la vie sauvage reprendra ses droits

« Nous sommes les premiers à comprendre vraiment le problème et les derniers à pouvoir encore y remédier »

Unit testing

Principles, Practices, and Patterns

by Vladimir Khorikov

Goal of Unit testing

Enable sustainable growth of the software project

Project without tests

- Quickly slows down
- Hard to make any progress

Fight entropy

- Constant cleaning and refactoring
- Tests act as a safety net

What makes a successful test suite?

- Integrated into the development cycle
- Targets most important parts of the code base
- Provides maximum value
  - With minimum maintenance costs

A tool that provides insurance against a vast majority of regressions

Not all tests are created equal

Bad tests : raise false alarms

- Unit tests are vulnerable to bugs
- Require maintenance

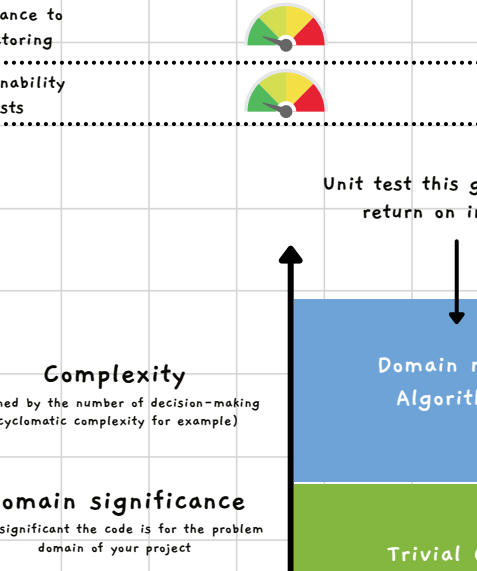
Tests are code too

View them as part of your code base that aims at solving a particular problem: ensuring the application's correctness

What is a Unit Test ?

Automated test that :

- Verifies a small piece of code (also known as a unit)
- Does it quickly
- And does it in an isolated manner.



Protection against regressions

- A regression = a software bug
- The larger the code base -) the more exposure to potential bugs
- Tests should reveal those regressions

Resistance to refactoring

The degree to which a test can sustain a refactoring of the underlying application code without turning red (failing)

Fast feedback

The more of them you can :

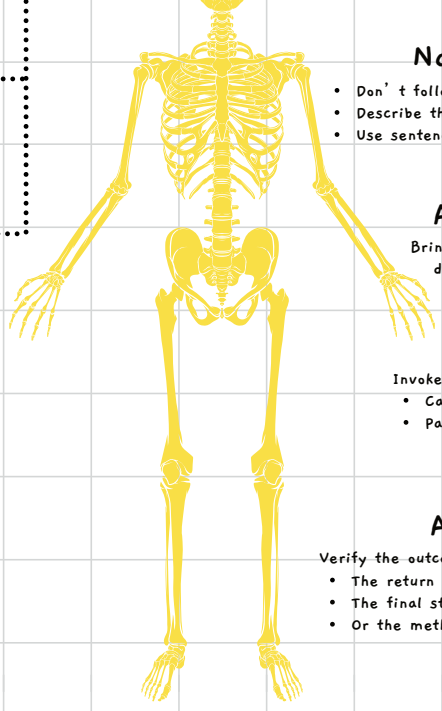
- Have in the suite
- Run them -) shorten the feedback loop

Maintainability (Maintenance costs)

- How hard it is to understand the test
- How hard it is to run the test

	London School	Classical School
A unit is	A class	• A class or set of classes • Behavior
Test doubles FOR	All but immutable dependencies	Shared dependencies

Anatomy



Test class name

Class-container for a cohesive set of tests

Name of the test

- Don't follow a rigid naming policy
- Describe the scenario to a non-programmer
- Use sentences

Arrange / Given

Bring the system under test (SUT) + dependencies to a desired state

Act / When

Invoke the behavior :

- Call method / function on the SUT
- Pass the prepared dependencies

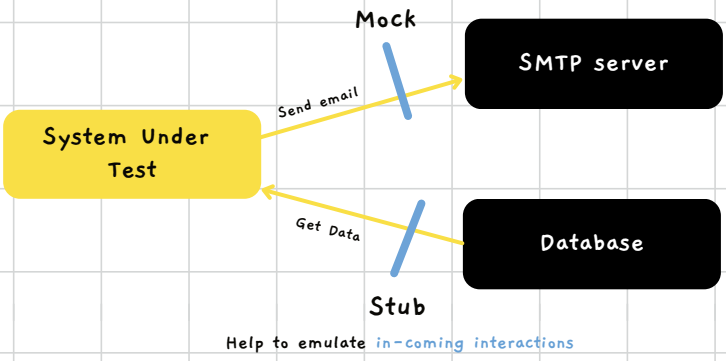
Assert / Then

Verify the outcome :

- The return value
- The final state of the SUT and its collaborators
- Or the methods the SUT called on collaborators

Test doubles

Help to emulate and examine out-coming interactions

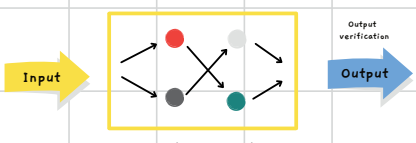


Help to emulate in-coming interactions

3 Styles of tests

Output-based

- Feed an input to the system under test (SUT)
- Check the output it produces

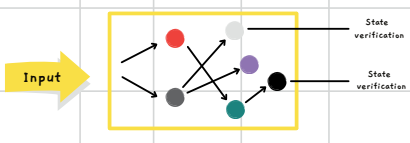


Assumes there are no side effects and the only result of the SUT is the value it returns to the caller -) functional

Both school use it

State-based

Verify the final state of the system after an operation is complete



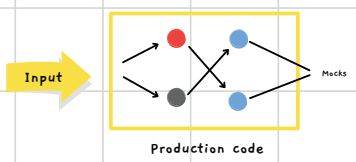
"State" can refer to the state of :

- The SUT itself
- One of its collaborators
- Or an out-of-process dependency (db / fs)

Classical preference

Communication-based

Verify that the SUT calls its collaborators correctly



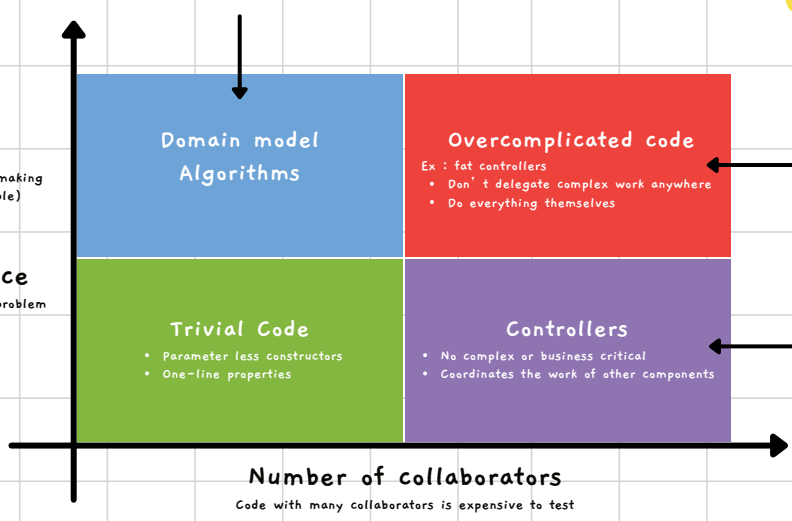
Tests substitute collaborators with mocks

London preference

	Output-based	State-based	Communication-based
Resistance to refactoring			
Maintanability costs			

4 Types of Code

Unit test this gives the best return on investment



Refactor it by splitting into :

- Algorithms
- Controllers

Integration Tests

LEADERSHIP STRATEGY and TACTICS

by Jocko Willink

"A good leader has nothing to prove, but everything to prove."

STRATEGIES

Detach
Mentally from the problem

Humility
Always learn

Leaders tell the TRUTH

Control Yourself
Don't overreact

Earn Influence & RESPECT

Self Discipline

No Yes-men
Favor challenging people

Pride
Drives positive behavior

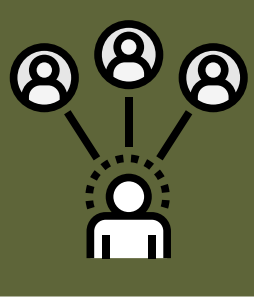
SKILLS to be a Good Leader

Simple Communication
Confidence
Charisma
Read People

Acknowledge Strengths/Weaknesses

The Power of Relationships

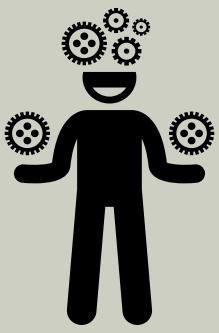
basis of all good leadership



HOW TO SUCCEED AS A NEW LEADER ?

BEHAVIORS

Take Ownership
Of failures and mistakes



Get the Job DONE
Of failures and mistakes

Pass Credit
For success up and down

Treat People with Respect
Take care of them / will take care of you.

RELATIONSHIPS

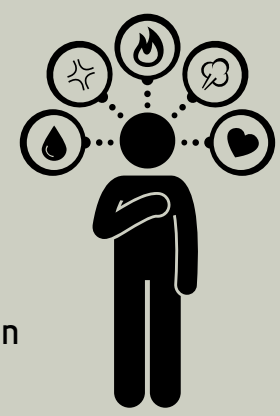
Build
Build trust

Listen
Ask for advice and heed it

Don't Act Like you Know Everything
You don't... Ask smart questions

SELF-BEING

Be Balanced
Extreme actions / opinions are not good.



Work Hard
Work harder than anyone

Be Decisive
When it is time to make a decision make one

Be Humble
An honor to be in a leadership position

Have Integrity
Do what you say; say what you do.

USE LEADERSHIP TO TEACH & BUILD

Fixing negative attitude
Maybe not at the right place

(re)Building Confidence

Teaching Humility
Fix overconfidence

Building high-level team players
Put junior in charge

by Yoan THIRION

RÉALISER SES RÊVES ÇA S'APPREND !

PAR THOMAS GIBOT

Réaliser ses rêves est une compétence
s'acquiert et se développe

Manifestation de ce que nous sommes vraiment
- au plus profond de nous-mêmes
- détachée de notre ego / peurs
- représente notre essence véritable

Le rêve comme une stratégie (un moyen)
- Pourquoi est-il si important pour moi ?
- Qu'est-ce qu'il me donne l'occasion de vivre, de ressentir ?

La partie la plus libre de nous-mêmes qui se manifeste

CONVICTION

Tor des Géants
330 km 24 000 D+

"Je rêvais mon rêve"
En parler > agir pour lui
Évoquer mon rêve = le vivre

Évidence
PHASE DU KIF
Fluidité Joie Énergie

Comment vais-je m'organiser pour faire avancer mon objectif ?
Réfléchir au processus

"Faire de mon rêve un projet"

PHASE DU TAF
Besogneux

ÊTRE UN BON RÊVEUR

Attentif
percevoir ses étincelles

Optimiste
- porter son attention sur le positifs
- sans nier le négatif

Pragmatique
se poser les questions honnêtes

Structuré
- se doter d'une organisation robuste
- dédier du temps et de l'énergie

Conscient
son rêve va l'impacter

Engagé
- se détache du résultat
- résilience
- confiance quasi inébranlable

> "Ce qui nous fait passer de quelqu'un qui rêve ses rêves à quelqu'un qui se donne une chance de les vivre."

"JE NE SAIS PAS RÊVER !" → **"DEVENIR UN MEILLEUR RÊVEUR"**

SE PRÉPARER POUR SES RÊVES

CRÉER SON SYSTÈME EST LE MOYEN D'ACCOUCHER DE NOS RÊVES.

ASTRONOME DU RÊVE

Maximiser nos chances de réaliser nos rêves !
- Augmenter notre niveau de discernement
- Identifier leviers pour nous permettre de le réaliser

LA CONSTELLATION DU RÊVE

INTITULÉ DE NOTRE RÊVE
au centre de la Constellation

Vision juste de ce qu'il **représente** et **implique** pour nous

2 AXES

L'Être — **facteurs** qui nous sont **propres**
nos envies, nos besoins, nos peurs, nos croyances, notre engagement, ...

Le Faire — éléments extérieurs
- avec lesquels nous allons devoir composer / adapter
- afin de progresser vers ce qui compte pour nous

notre organisation, nos relations

ENJEUX
problèmes, besoins, blocages, questionnements
composent chacun des axes

ACTIONS
Identifier pour chacun des enjeux :
- actions concrètes / réalistes
- responsables = moi

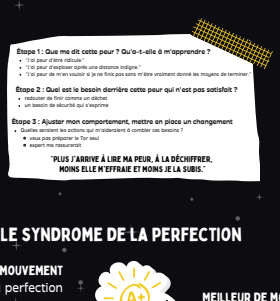
RESSOURCES
Tout ce qui pourrait nous permettre de changer notre rapport à nos enjeux
inspirations extérieures
TED
Livres, Podcasts, Vidéos, Formations

> Ma conviction est qu'en améliorant « **le Faire** », nous **avançons fort**.
> En ayant une plus grande conscience de nous-mêmes, de « **l'Être** », nous **avançons juste**.

NE PLUS AVOIR PEUR D'AVOIR PEUR
- Pouvoir nommer ses émotions
  - première étape pour les comprendre
- Savoir l'écouter pour la comprendre

COMBATTRE LE JUGE INTÉRIEUR
- Cette voix qui nous gâche la vie
- "Je suis nul, les autres sont tellement meilleurs que moi,..."

DIALOGUER AVEC SA PEUR
THOMAS D'ANSEMBOURG



NOUS MAINTENIR DANS UN STATU QUO
sa mission principale

NOUS MAINTENIR DANS UN MALHEUR ACCEPTABLE
ne jamais sortir de sa zone de confort

AXE 1 - L'ÊTRE

CÔTOYER SES RESSENTIS / ÉMOTIONS
- les clarifier / les comprendre
- conscience sur ce qui se joue en nous à la poursuite de nos rêves!

Énergie PLUTÔT QUE **Rationalité**

L'IMPORTANCE D'ÊTRE DÉRAISONNABLE
Intégrer pleinement les difficultés
Choisir tout de même de suivre l'élan de son cœur

LE SYNDROME DE LA PERFECTION

SE METTRE EN MOUVEMENT
ne pas de rechercher la perfection

FAIRE DE MON MIEUX
plutôt que parfaitement

MEILLEUR DE MOI
même si pas parfait

"ACCEPTER PLEINEMENT QUE L'ÉCHEC FASSE PARTIE DU RÊVE"

> "Si tes rêves ne te font pas peur, c'est qu'ils ne sont pas assez grands." - **Mike Horn**

CHANGER SON RAPPORT AU REGARD DES AUTRES

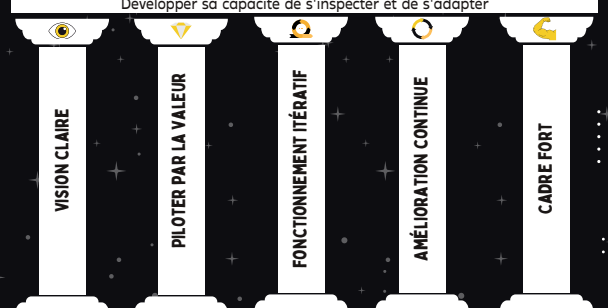
Est-ce que les autres vont trouver ça bien? Vais-je être valorisé et reconnu en poursuivant ce rêve ? **VS** Comment tout ce que j'entreprends est-il un moyen de me prouver que je m'engage pourfice qui est important pour moi ?

"MA RECONNAISSANCE PLUTÔT QUE LA RECONNAISSANCE"

L'AGILITÉ POUR AVANCER VERS NOS RÊVES

SCRUM

Développer sa capacité de s'inspecter et de s'adapter

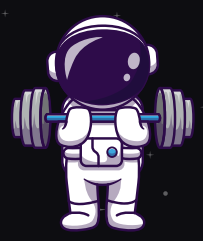


NOS IDENTITÉS DICTENT NOS ACTIONS
je suis un lecteur
je suis quelqu'un qui se lève tôt
je suis un sportif

NE S'ARRÊTENT PAS LORSQUE LE PROJET EST TERMINÉ

PAS NOS OBJECTIFS
lire un livre par semaine
me lever à 6 h pendant deux mois
perdre trois kilos

S'EFFACENT UNE FOIS ATTEINTS



NOUS DÉFINISSENT DANS LE TEMPS

- Création d'un backlog
- Planning de l'itération : toutes les 4 semaines
- Daily : point pour savoir comment on avançait
- Revue d'itération : valider le travail effectué
- Retrospective : comment nous pouvions nous améliorer

- Prendre du recul
- Se poser la question si ce que l'on fait :
  - est toujours la bonne chose à faire
  - nous le faisons de la meilleure manière

TROUVER SON "GRIT"
la ténacité, le cran, le fait de serrer les dents

S'ACQUIERT
SE DÉVELOPPE
SE DÉCIDE

- Se focaliser uniquement sur ce que l'on peut maîtriser
- Exclure le négatif
- Assumer la douleur à court terme

ne plus voir les défis et les difficultés comme des éléments à éviter

AXE 2 - LE FAIRE

> "Sans grit, le talent n'est qu'un potentiel" - **Ben Bergeron**

CONSCIENTISER SES RELATIONS

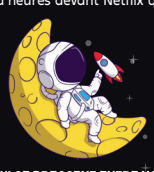
En avoir une représentation consciente
acter qu'une relation ne nous est plus bénéfique

COMMENT PUIS-JE M'ENTOURER DE PERSONNES POUR QUI MES ZONES D'INCONFORT, DE DOUTES, DE PEURS SONT LEURS ZONES DE CONFORT, DE SÉCURITÉ ET D'EXCELLENCE ?

> "Nous sommes la moyenne des cinq personnes que nous fréquentons le plus" - **Jim Rohn**

LES ANTI-FLEMMES
"C'est grâce à ces centaines d'heures devant Netflix que j'en suis là aujourd'hui."

Décomposer mes routines pour me mettre en action
Identifier mes déclencheurs d'actions
Mettre ma tenue de sport à côté de mon lit
Annoncer le matin à mes collègues que j'allais courir
Dire à ma chienne que nous allions courir



Prendre des décisions
pour me maintenir le plus possible engagé dans mon rêve

Prendre des engagements envers les autres
- rendre public certains objectifs
- utiliser nles autres» comme le supplément d'âme ou d'énergie

Le tableau des rêves
simple cadre où inscrire tout ce qui nous fait vibrer

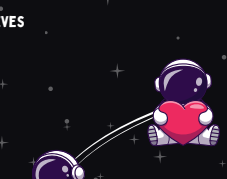
Avoir des boosts de motivation
- musique
- texte de Kilian Jornet

La dimension "utile"
«Quelles sont les activités que je ne peux faire que pendant que je cours ? »

L'UN DES PIRES FLÉAUX QUI SE DRESSENT ENTRE NOUS ET NOS RÊVES

> "Jamais de projection à long terme! Je pourrais me le tatouer sur le front !"

PHASE 6 : RESSENTIR L'IMPACT DE LA CONSTELLATION



- Quelles émotions sont présentes en moi à la lecture de ma Constellation ?
- Comment la sensation d'être acteur de mon rêve a-t-elle évoluée?
- En quoi me paraît-il plus atteignable à présent ?

IDENTIFIER COMMENT VOUS VOUS SENTEZ FACE À CETTE REPRÉSENTATION

PHASE 5 : FINALISER MA CONSTELLATION

ORGANISER ET VISUALISER VOTRE CONSTELLATION DANS SON ENSEMBLE

ÉTAPE 15 : IDENTIFIER DES RESSOURCES PERTINENTES ET INSPIRANTES

- Quels sont les podcasts, livres ou conférences de référence sur cet enjeu ?
- Existe-t-il des formations qui pourraient m'aider ?
- Quelles communautés pourrais-je rejoindre autour de cet enjeu ?

BESOIN D'ÊTRE INSPIRÉS, GUIDÉS ET MOTIVÉS

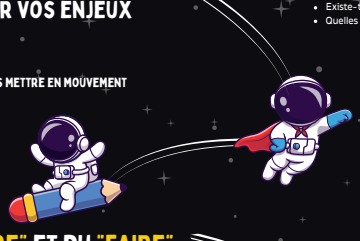
ÉTAPE 14 : DÉTERMINER LES ACTIONS POUR AVANCER SUR VOS ENJEUX

- Quel serait un premier pas en direction de la résolution de cet enjeu ?
- Si je ne pouvais pas échouer, qu'est-ce que je ferais ?
- Que feraient les personnes qui m'inspirent sur cette thématique ?

AVOIR IDENTIFIÉ LES ENJEUX SUR LESQUELS NOUS AVONS BESOIN DE TRAVAILLER NE SUFFIT PAS À NOUS METTRE EN MOUVEMENT

PHASE 4: RENDRE VOTRE CONSTELLATION ACTIONNABLE ET INSPIRANTE

UNE ACTION VAGUE N'A AUCUNE CHANCE D'ÊTRE MISE EN ŒUVRE !



REGROUPER TOUT CELA DANS VOTRE CONSTELLATION
- Formulés au "Je"
- Pas au conditionnel, mais au **présent de l'indicatif**
- Des verbes de résultats comme "**j'ai**" ou "**je suis**"

PHASE 3: STRUCTURER LES AXES DE "L'ÊTRE" ET DU "FAIRE"

FAIRE

ÉTAPE 13 : DÉCIDER CE QUE VOUS DEVEZ ARRÊTER, DÉMARRER OU MODIFIER

ÉTAPE 12 : LUTTER CONTRE LA PROCRASTINATION

- Qu'est-ce qui me donne énormément d'énergie et que je pourrais renforcer ?
- Comment puis-je diminuer les frictions pour démarrer les actions importantes ?
- Comment puis-je découper mes tâches importantes en toutes petites premières actions simples à réaliser ?

STRATÉGIES POUR NOUS REMETTRE EN MOUVEMENT ET ÉVITER L'IMMOBILISME

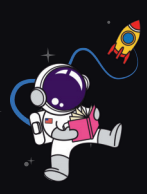
ETAPE 11 : DÉFINIR VOS MARQUEURS DE NON-AVANCEMENT

- Quelles sont les pensées présentes lorsque j'ai la sensation d'être bloqué ?
- Qu'est-ce que j'observe lorsque je le suis ?
- Qu'est-ce que les autres me disent quand ils ont l'impression que je n'avance plus ?

ÉLÉMENTS CONCRETS, OBSERVABLES, TANGIBLES QUI NOUS INDIQUENT QUE NOUS STAGNONS

ÉTAPE 10 : DÉFINIR VOS COMPÉTENCES

- Quels sont les principaux facteurs de réussite de mon rêve ?
- Quelles qualités ou compétences me sont nécessaires pour les accomplir ?
- Comment vais-je pouvoir les acquérir ou les renforcer ?



ÉTAPE 9 : DÉFINIR DES OBJECTIFS INTERMÉDIAIRES

Découper notre rêve en petits objectifs intermédiaires

ÉTAPE 8 : DÉFINIR VOTRE ORGANISATION

- Comment organiser toutes les actions que je vais devoir réaliser ?
- Quelles sont les routines dont je vais avoir besoin ?

PERMETTRE DE SORTIR VOTRE RÊVE DE VOTRE TÊTE POUR COMMENCER À LE RENDRE CONCRET

ETAPE 7 : LA PLACE DES AUTRES

- Quelles relations sont à mon service ? Sur quels sujets ?
- Quelles sont celles qui m'éloignent de la réalisation de mon objectif ?
- Quelles sont les personnes impactées par mon rêve ?
  - Comment est-ce que je souhaite en tenir compte ?

ÊTRE

ÉTAPE 6 : CLARIFIER LA NOTION D'ÉCHEC

- Qu'est-ce qui me rendrait triste, déçu, en colère envers moi-même en lien avec ce rêve ?
- A la lumière de cette réponse, qu'est-ce que je considérais jusqu'alors comme un échec mais qui n'en est plus un ?

PLUS NOUS SOMMES AU CLAIR SUR CE QUE SERAIT UN ÉCHEC, PLUS NOUS NOUS LIBÉRONS DE SON EMPRISE

ÉTAPE 5 : IDENTIFIER VOS DIFFICULTÉS

- Qu'est-ce qui me fait peur en lien avec ce rêve ?
- Qu'est-ce qui pourrait me faire échouer?

ÉTAPE 4 : IDENTIFIER VOS BESOINS NOURRIS PAR CE RÊVE

- Le rêve n'est pas une finalité
- Une stratégie qui vise à nourrir certains de nos besoins fondamentaux

ÉTAPE 3 : LES IMPACTS BÉNÉFIQUES DE VOTRE RÊVE

- Quelles émotions positives est-ce que cela génère ?
- Quels ressentis physiques positifs est-ce que cela me procure ?
- Quelles pensées positives sont présentes ?
- Quoi d'autre ?

PHASE 2 : IDENTIFIER LES DIFFÉRENTES ÉTOILES DE VOTRE CONSTELLATION

Identifier tous les thèmes qui vont s'avérer être des enjeux pour la réalisation de votre rêve

ÉTAPE 2 : NOMMER VOTRE RÊVE

- Tel que vous le ressentez et l'imaginez vraiment
- La plus rayonnante et la plus authentique

ÉTAPE 1 : AMENER VOTRE TÊTE, COEUR, CORPS

- Nos rêves font appel à tout : nos pensées, nos ressentis physiques et nos émotions.
- Poser sur le papier tout ce qui se passe en nous
- Taire nos pensées limitantes, nos jugements, nos peurs

PHASE 1 : CLARIFIER VOTRE RÊVE

CRÉER SA CONSTELLATION

#SHARINGISCARING

PAR YOAN THIRION

SUCCEEDING WITH OKRS IN AGILE

BY ALLAN KELLY HTTPS://WWW.ALLANKELLYASSOCIATES.CO.UK/

OBJECTIVES

BIG GOALS

SOMETHING THE ORGANIZATION WANTS / VALUES

AVOID BOXING YOURSELF

INTO A SPECIFIC APPROACH OR SOLUTION

MAKE THE VALUE THAT BRINGS OBVIOUS

SO THAT...

RETOOL THE DELIVERY PIPELINE TO FACILITATE CONTINUOUS DELIVERY

VS

INCREASE ROI BY REDUCING TIME TO MARKET WITH
A NEW DELIVERY PIPELINE AND CONTINUOUS DELIVERY PRACTICES

KEY RESULTS

SMALLER GOALS THAT BUILD TOWARDS THE OBJECTIVE

FIGHT AGAINST DOMINOS

DON'T ACCEPT DEPENDENCIES

EACH ONE MUST DELIVER VALUE

ALL ABOUT DELIVERING OUTCOMES THAT ADD VALUE

KEY RESULTS TRICKS

EXPERIMENTS

SAFER FOR THE TEAM TO TAKE ON RISK
SUCCESS = DOING THE EXPERIMENT ITSELF AND ABSORBING THE LEARNING

HYPOTHESIS-DRIVEN DEVELOPMENT

WE BELIEVE <THIS CAPABILITY>
WILL RESULT IN <THIS OUTCOME>
WE WILL HAVE CONFIDENCE TO PROCEED WHEN <WE SEE A MEASURABLE SIGNAL>

USE SURVEY

MAKE CHANGES TO PEOPLE
TEST IT WITH SURVEY

TAKE SURVEY

TIME-BOXES

EXPERIMENT SOMETHING FOR N WEEKS

"if you aren't failing, you aren't trying"

WHY ?

FILL A NEED AT THE MID-TERM PLANNING LEVEL

LATER

LOOK MONTHS / YEARS INTO THE FUTURE

SOON : OKRS

LOOK TO THE NEXT FEW MONTHS

NOW : SPRINT PLANNING

FEW WEEKS INTO THE FUTURE

CREATE FOCUS

TELLS YOU WHEN TO STOP

TRUE NORTH

GUIDE AND FIGHT TO STAY ON COURSE
DON'T STICK BLINDLY TO OKRS AS THE WORLD AROUND CHANGES

OKRS ENHANCE COMMUNICATION

- EASIER TO COMMUNICATE WHAT A TEAM IS DOING
- A MEANS OF COMMUNICATING STATUS AND PROGRESS
- SUCCESS MOTIVATES CONTINUATION

HOW TO ?

OBJECTIVE VALUE > Σ (KEY RESULTS VALUES)

BOTTOM UP

DON'T IMPOSE OKRS FROM ABOVE
TEAM RESPONSIBLE FOR SETTING THEIR OWN OKRS AND DELIVERING THEM

LIMIT THEIR NUMBER

3 OBJECTIVES
3 KEY RESULTS PER OBJECTIVE

LEADERS

BUILD PSYCHOLOGICAL SAFETY / MAKE FAILURE AN OPTION
MAKE COMPLETELY CLEAR WHAT THE PRIORITIES ARE

ALL OKRS ARE NOT EQUALS

SOME MIGHT BE HIGHER PRIORITY

WHAT NOT TO DO

EVERYTHING THAT IS NOT IN THE OKRS IS LOWER PRIORITY

STRATEGY

- WHAT ARE THE STRATEGIC PRIORITIES FOR THE NEXT QUARTER ?
- WHAT DOES THE TEAM AIM TO DO ?
- WHAT TARGETS WILL THE TEAM SET FOR ITSELF ?

TEST DRIVEN APPROACH

DECIDE WHAT YOU WANT : OBJECTIVE

SET A SERIES OF ACCEPTANCE CRITERIA : KEY RESULTS

EACH KEY RESULT SHOULD BE MEASURABLE

GET ON AND DEVELOP

DON'T CONSIDER YOURSELF DONE UNTIL

YOU CAN PASS THE TESTS

YOU MEET THE OBJECTIVES

"As with agile, you need to find you own way to OKRs [...] be prepared to experiment."

OKRS AND BACKLOG

BACKLOG FIRST

SUCCESS : BURN DOWN THE BACKLOG
OKRS : ONE OF SEVERAL INPUTS

VS

OKRS FIRST

SUCCESS : DELIVER OKRS
OKRS ARE EVERYTHING

TIMELINE

SET OKRS A FEW WEEKS BEFORE NEXT QUARTER

2 OR 3 SHOULD BE FINE

REVIEW AT THE END OF EACH QUARTER

MANY FORMS OF VALUES

LEARNING

KNOWLEDGE ON NEW TECH FOR EXAMPLE

FEEDBACK

EXTEND OUR EXISTING KNOWLEDGE

RISK REDUCTION

INCREASES THE PROBABILITY OF DELIVERING VALUE

MONEY

MONEY IS THE BEST FORM OF FEEDBACK

"Some things are more important than OKRs and sometimes those things can't be measured."

AMBITION OVER ESTIMATION

UTILITY MODE

OKRS SET BASED ON WHAT IS ACHIEVABLE
PREDICTABILITY IS VALUED
TEAMS AIM TO ACHIEVE ALL OKRS

ASPIRATIONAL MODE

MOONSHOT OKRS : BASED ASPIRATION
IMPACT IS VALUED
TEAMS EXPECT TO FAIL STRETCH OKRS

TEAMS ARE NOT NORMALLY EXPECTED TO COMPLETE 100% OF THEIR OKRS
70% IS MORE COMMON

70%

AIM HIGH

NOT IMPOSSIBLY HIGH
BUT HIGH ENOUGH TO BE CHALLENGED

WHERE ARE YOU ?

CULTURE

"if you aren't failing, you aren't trying"

DELIVERY CULTURE

VALUE DELIVERY (WORKING PRODUCTS USED BY CUSTOMERS)
NOT HOURS WORKED, NOT PARTIALLY DONE WORK

SUPPORTIVE CULTURE

PSYCHOLOGICAL SAFETY
FAILURES WILL HAPPEN

DON'T LINK REMUNERATION TO OKR OUTCOMES

IF MONEY ATTACHED

- PEOPLE FEEL COMPELLED TO CHASE **100% SUCCESS**
- EASIEST WAY = **REDUCE THE TARGET**

#SHARINGISCARING