Grow Professionalism!

Olve Maudal

A key to succeed with software development is to create an environment where software engineers are really treated as professionals. In return software engineers should use the opportunity to always reach for new levels of professionalism. Allow professionalism to thrive within your organization.

In this talk I will discuss a cocktail of ideas like: burn your corporate coding standards, stay close to the value stream, get rid of metrics, prefer collective ownership, always train your strongest developers first, be transparent, remove mechanisms for shifting responsibilities, optimize for your 80% best developers, introduce slack, celebrate failures, and more...

45 minute keynote at Software Craftmanship Day, Equinor, Bergen, 12. November 2019

Safety moment

The importance of a clean and functioning work environment

Suppose you are just going to make something nice for yourself...

then, really, anything will do. Even...

but, software development is usually about more than just making something nice for yourself.

It is usually about making something really fancy...




寿司

⁺˖⁺sushi⁺˖⁺

which one do you like best?




together with a large team of professionals...

for some demanding customer ...

You need a clean and functional work environment


OLIVE OIL

Your codebase is like a kitchen.

Keep it clean and functional so that you can create spectacular solutions for your demanding customers!

Grow Professionalism!

Olve Maudal

A key to succeed with software development is to create an environment where software engineers are really treated as professionals. In return software engineers should use the opportunity to always reach for new levels of professionalism. Allow professionalism to thrive within your organization.

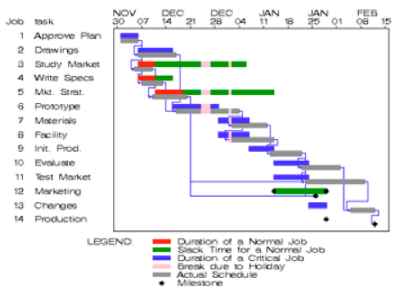
In this talk I will discuss a cocktail of ideas like: burn your corporate coding standards, stay close to the value stream, get rid of metrics, prefer collective ownership, always train your strongest developers first, be transparent, remove mechanisms for shifting responsibilities, optimize for your 80% best developers, introduce slack, celebrate failures, and more...

45 minute keynote at Software Craftmanship Day, Equinor, Bergen, 12. November 2019

There used to be a time, when it was believed that anyone could do software development

after all, how hard could it be, it was just about programming a computer...

1) get some smart people to analyze the problem



2) create a plan

3) find resources

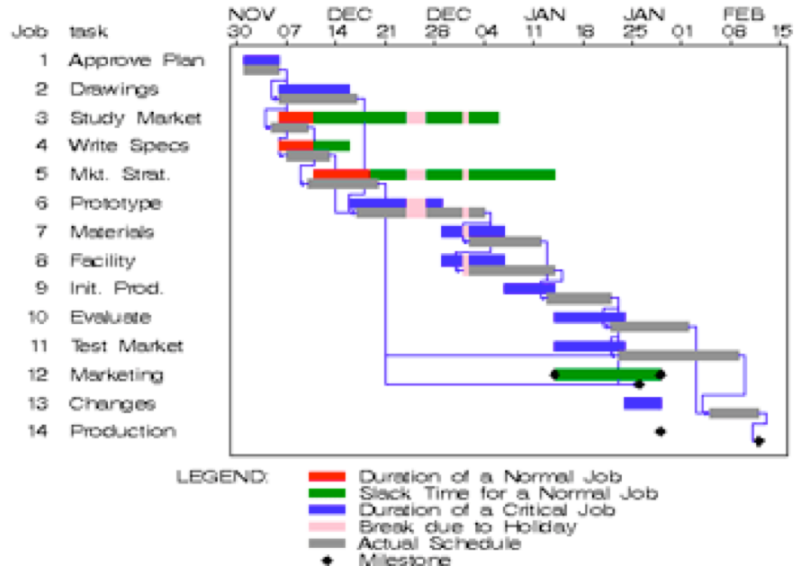
4) execute according to the plan

and when projects failed

the respons was always:

do more up-front analysis

create a more detailed plan



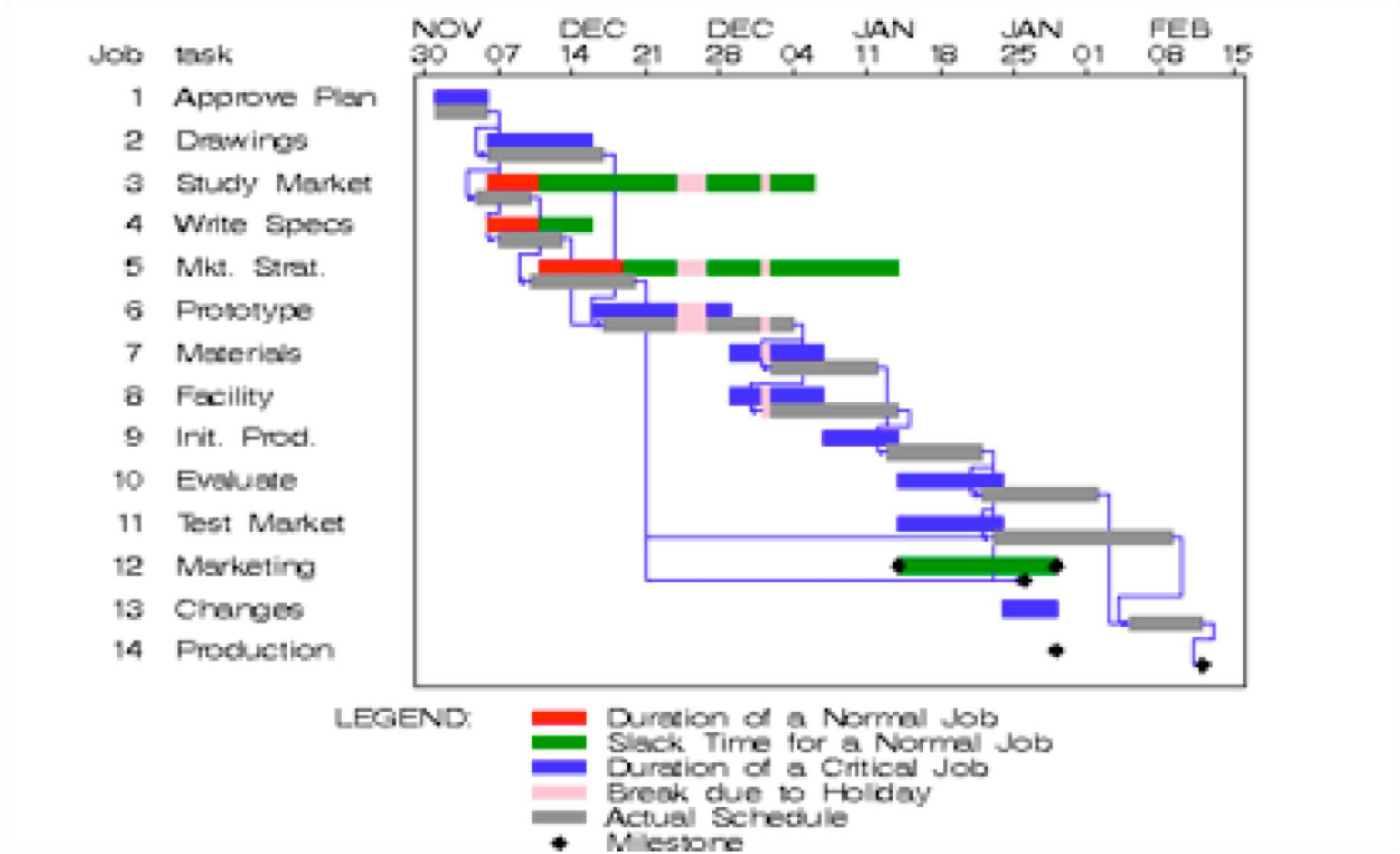
find more resources

and make sure that everyone followed the plan

but the projects still failed

and the respons was, as always...

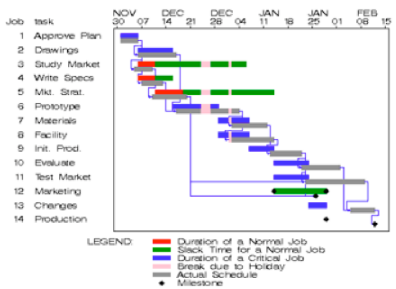

FREE
OR THE UNEMPLOYED
FREE SOUP
HORA
BAILIFF


Job task
NOV 30 07
DEC 14 21
DEC 28 04
JAN 11 18
JAN 25 01
FEB 08 15
1 Approve Plan
2 Drawings
3 Study Market
4 Write Specs
5 Mkt. Strat.
6 Prototype
7 Materials
8 Facility
9 Init. Prod.
10 Evaluate
11 Test Market
12 Marketing
13 Changes
14 Production
LEGEND:
Duration of a Normal Job
Slack Time for a Normal Job
Duration of a Critical Job
Break due to Holiday
Actual Schedule
Milestone

but of course...

Dark ages of software development (early 80’s to late 90’s)

1) get some smart people to analyze the problem



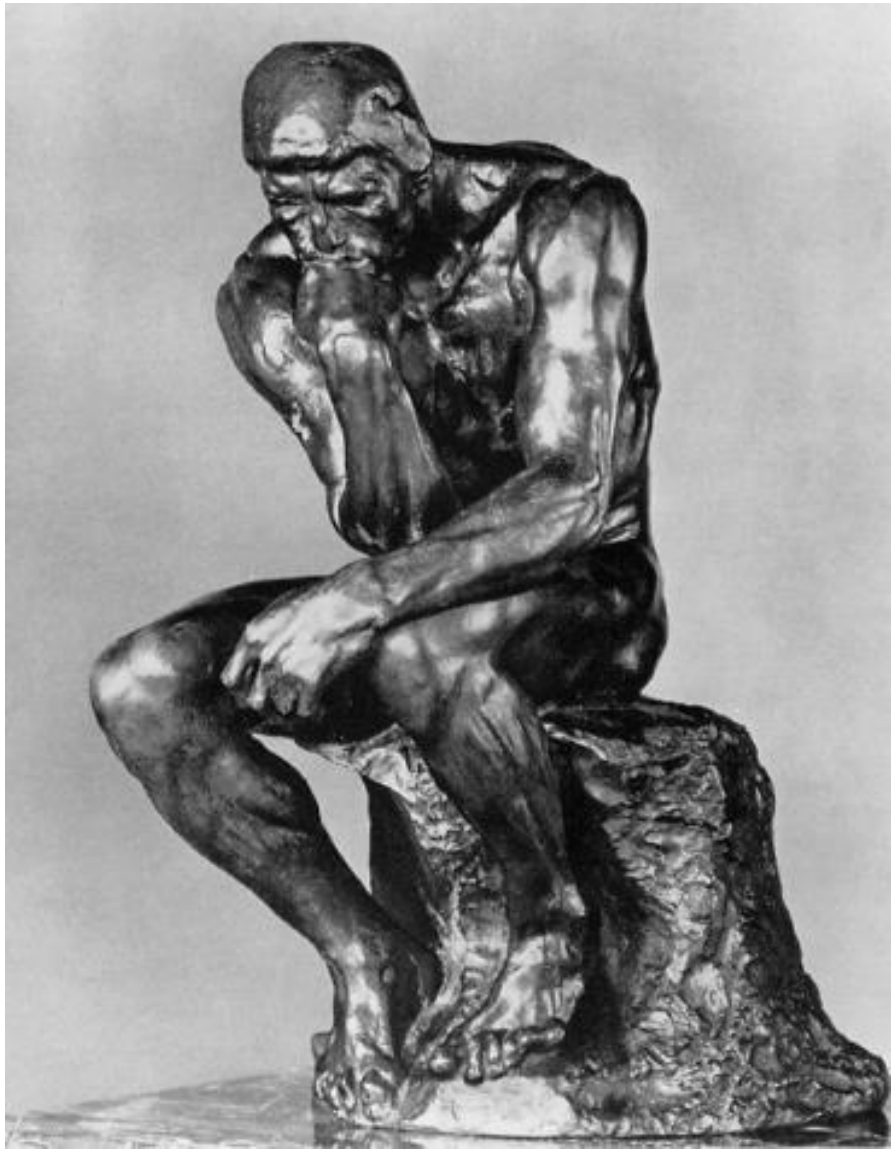
2) create a plan

3) find resources

4) execute according to the plan

We had only discovered a fancy way of playing the “scabby queen” game, also known as the “Old Maid” or “Svarte Per”, always try to “save your ass” by delegating responsibility to someone else.

There must be a better way...



The Agile Manifesto (2001)

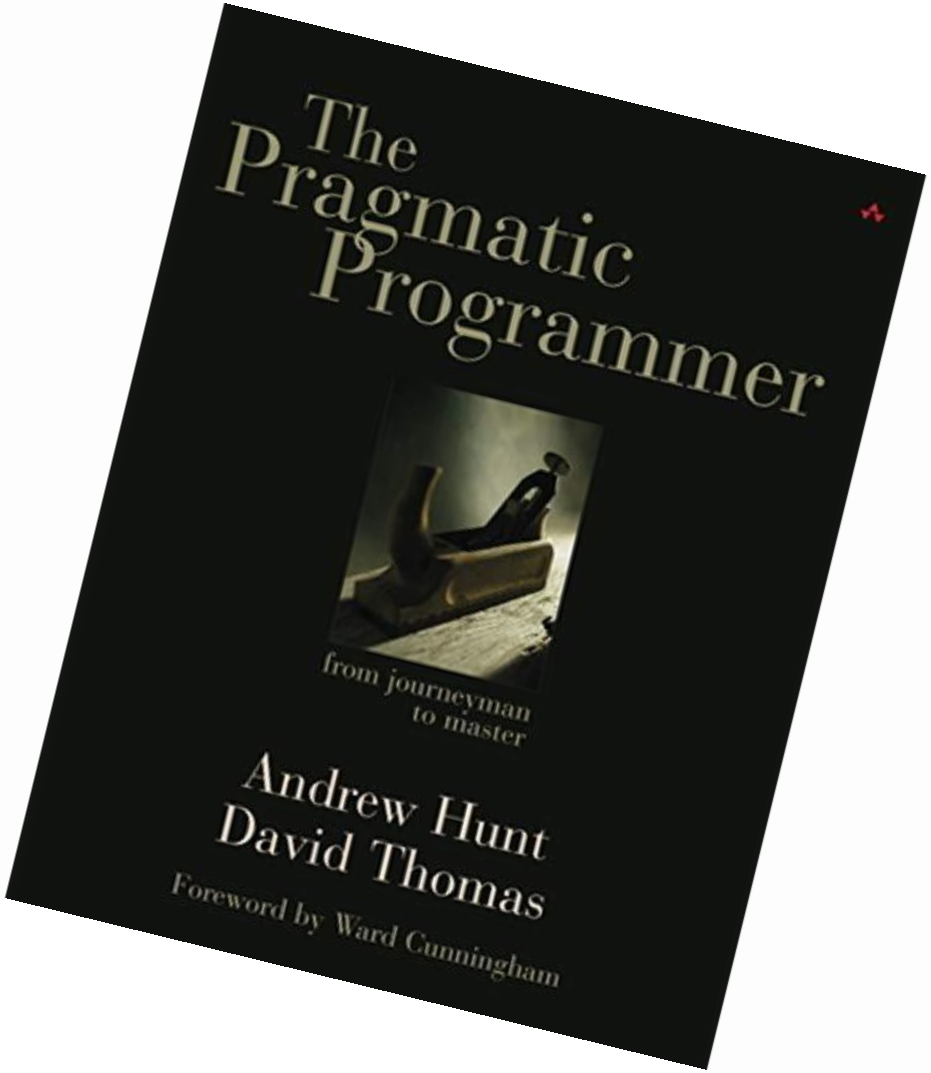
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

Individuals and interactions over **processes and tools**
Working software over **comprehensive documentation**
Customer collaboration over **contract negotiation**
Responding to change over **following a plan**

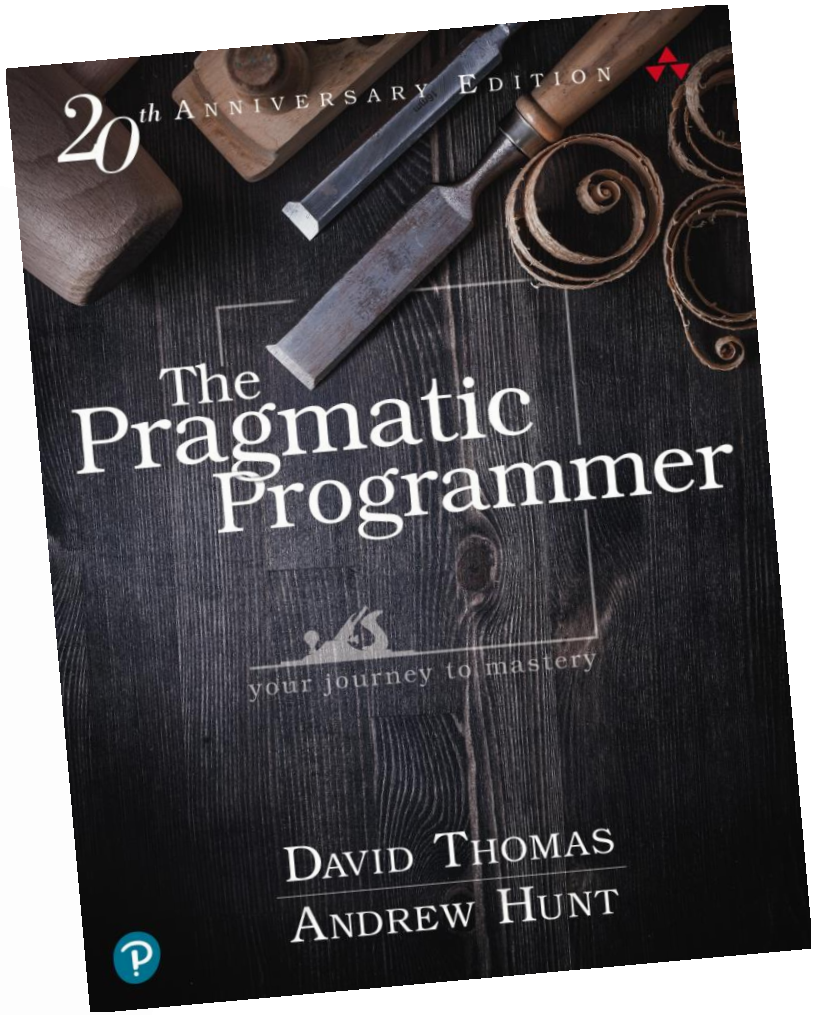
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

The agile manifesto started a huge
awakening process in the software industry...

(picture from the 1990 film Awakenings)


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham

1999


20th Anniversary Edition
The Pragmatic Programmer
your journey to mastery
David Thomas
Andrew Hunt

2019

Grow Professionalism!

Grow Professionalism!

make sure there is enough slack
encourage teams and individuals to meet and spend time together
instead of setting objectives, show your constraints and suggest a direction
respect the observer effect, get rid of externally imposed metrics
share your vision, never throw requirements "over the wall"
pull value out of a system, do not push
explain the business to everyone involved
don't celebrate the midnight cowboys
do not demand dates and estimates, aka "do not rush miracles"
respect that software development is a learning process
beware of governance and corporate coding standards
abandon decision gates and commitments
you may lock time and cost, but do not lock the scope
train your best developers first
get rid of mechanisms for shifting responsibilities around
don't treat your employees as criminals

think, use tools, create tools, focus on effectiveness
share knowledge, collaborate, *[removed]* and extend trust boundaries
align your efforts and make sure you all pull in the same direction
make sure you have enough data to know where you are going
understand the business, take overall responsibility for what you create
deliver value early and often, continuously improve
stay close to where the money is flowing, avoid the turbulence
work in a sustainable pace, build quality in, celebrate your vision
no sandbagging, be honest and be transparent, deliver magic
establish fast and reliable feedback loops, celebrate failures
stay up to date with current industry standards and best practice
establish reliable breaking mechanisms, fail fast, stop or change direction
deliver value early and often, without compromising the overall vision
share your knowledge, teach, encourage others to follow your ideas
be trustworthy, practice collective ownership and responsibility
always do the right things (even when nobody is looking)

a cocktail of topics related to Professionalism

Effectiveness vs Efficiency

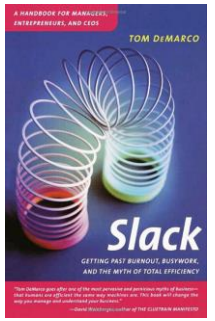
Introduce slack to become more effective!

100% full = high efficiency, very low effectiveness
50% full = high effectiveness, moderate efficiency

The more difficult tasks you need to solve, the more slack you need

Taking breaks to "sharpen your saw" is often very effective

*If your company's goal is to become fast, responsive, and agile, more efficiency is not the answer--you need more **slack**. (Tom DeMarco)*



Trust

Management by objectives is... a disaster!

Deming states unequivocally that merit reviews, by whatever name, including management by objectives, are the single most destructive force in American management today.

Constraints and direction

The observer effect

When observing a programmer, he/she is much more likely to add code than to delete code. Also, you can nearly guarantee that they will not shut their eyes and think carefully about things.

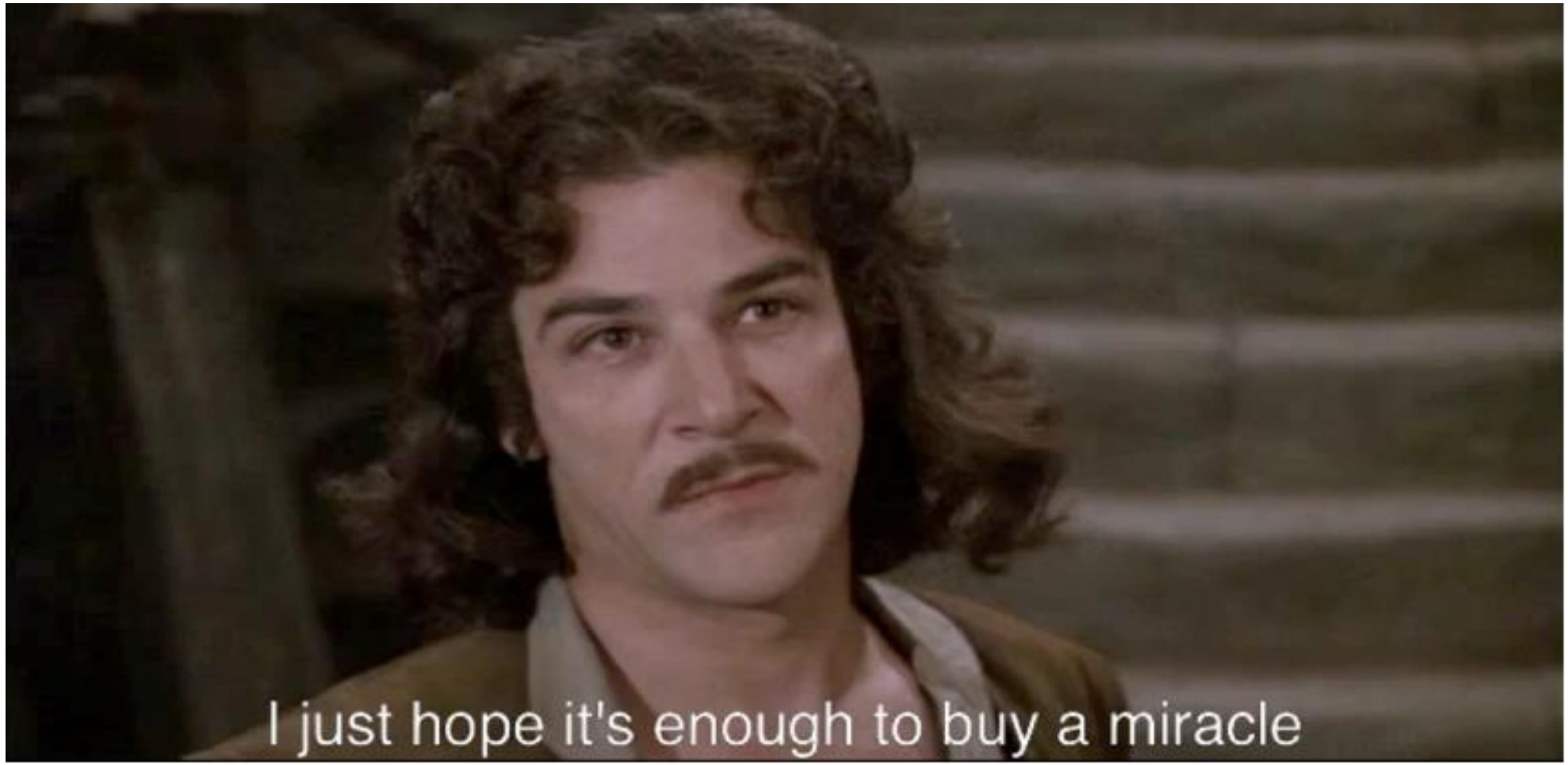

GARMIN
234
THUR 1
12
2:09
1763
1110

Pull vs Push

Money flow and turbulence

Midnight cowboys and sustainable pace

Inigo Montoya


I just hope it's enough to buy a miracle

Miracle Max

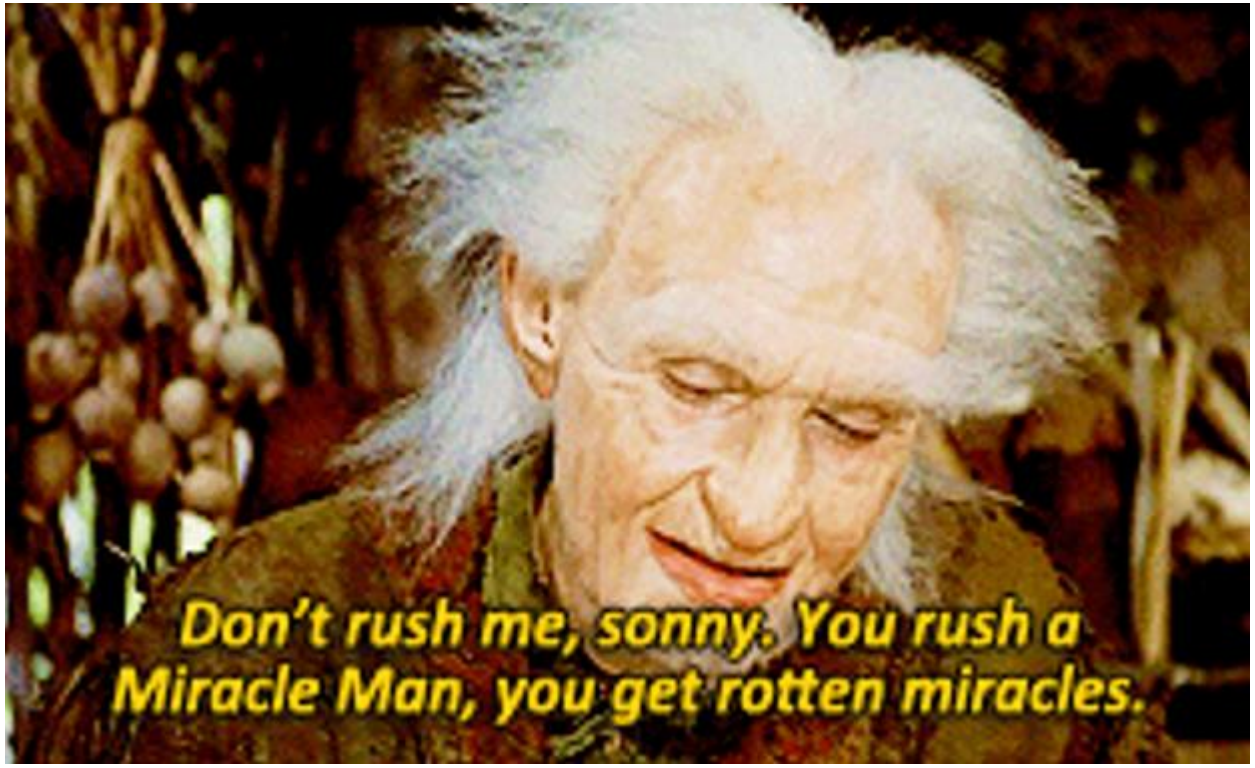
Inigo: We need a miracle. It's very important.

Inigo: We're in a terrible rush.
Miracle Max: Don't rush me, sonny.
You rush a miracle man, you get rotten miracles.

http://www.youtube.com/watch?v=1oWAtAWat4E

Inigo: I just hope it's enough to buy a miracle, that's all.
[Inigo knocks on the door. A face appears]
Inigo: Are you the Miracle Max who worked for the king all those years?
Inigo: We need a miracle. It's very important.
[after a while]
Inigo: Sir...sir??
Miracle Max: Huh?
Inigo: We're in a terrible rush.
Miracle Max: Don't rush me, sonny. You rush a miracle man, you get rotten miracles.

Do not rush miracles



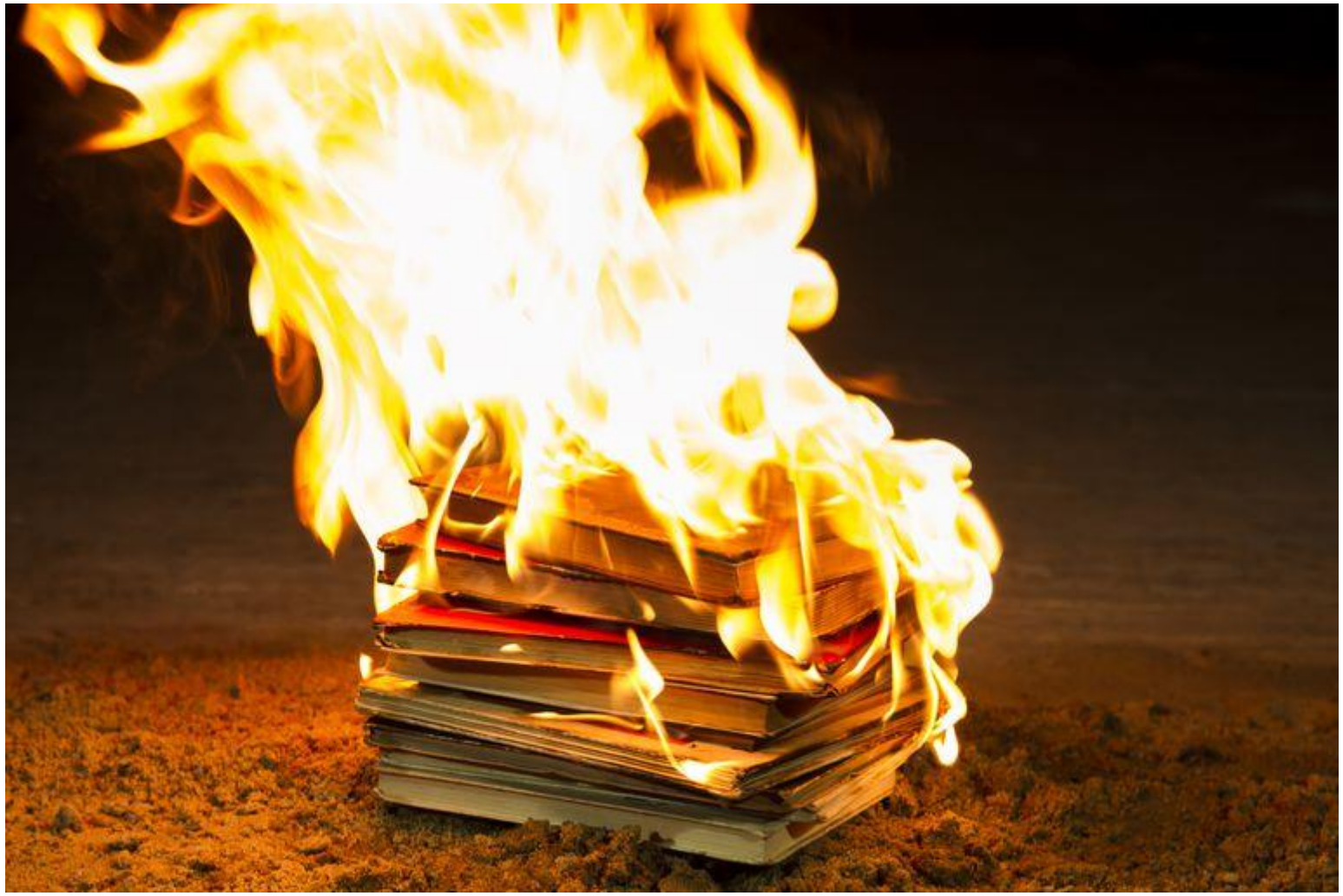
Any sufficiently advanced technology is indistinguishable from magic.

Arthur C. Clark (third law)

Be open, transparent and honest

Celebrate learning

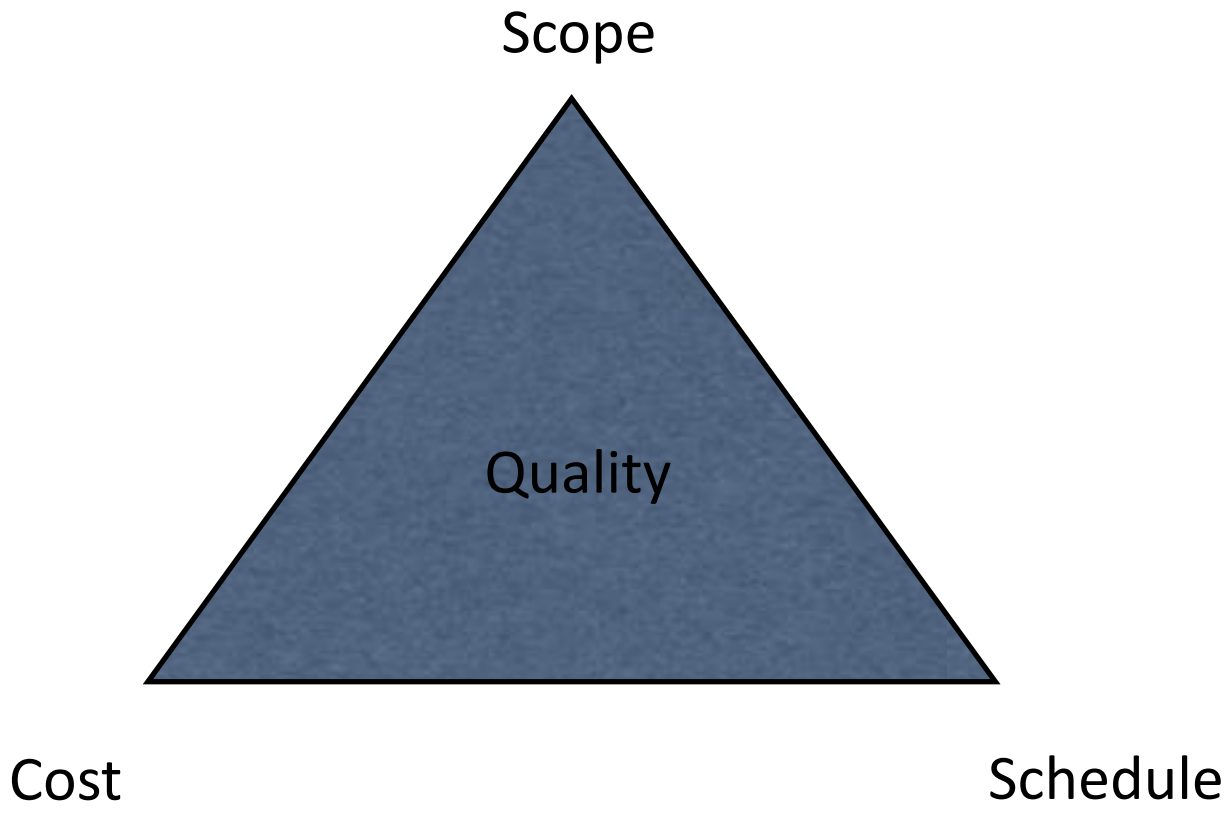
Burn the corporate coding standards



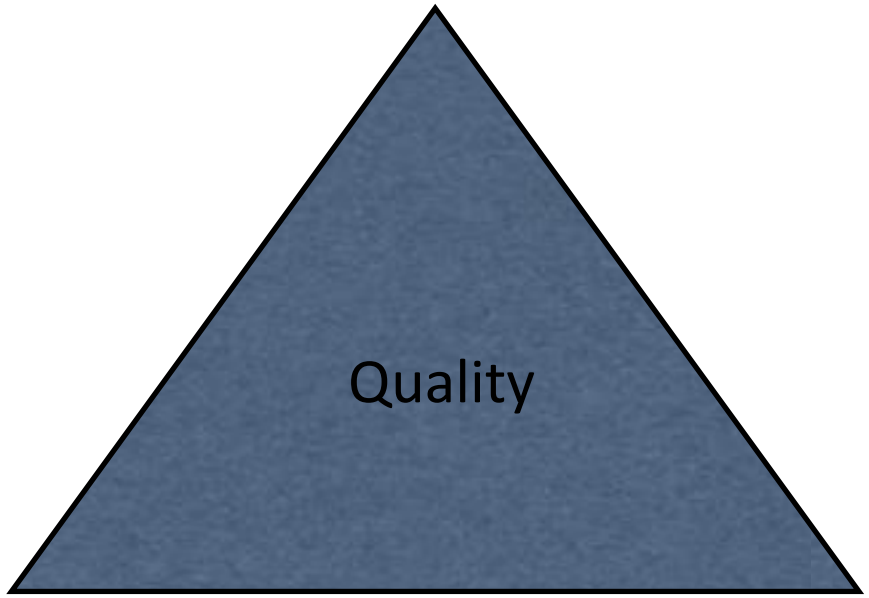
The corporate coding standard is usually written by some old-school developers that are not coding anymore.

Why do car have breaks?

“Fast, Good, Cheap. Pick two!”



Scope



Cost

Schedule

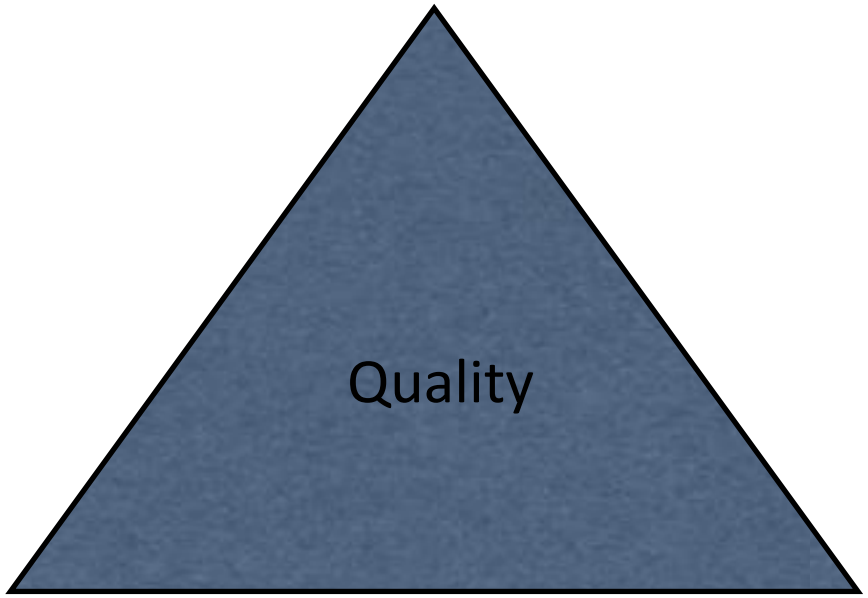
🔒 Scope

Quality

🔒 Cost

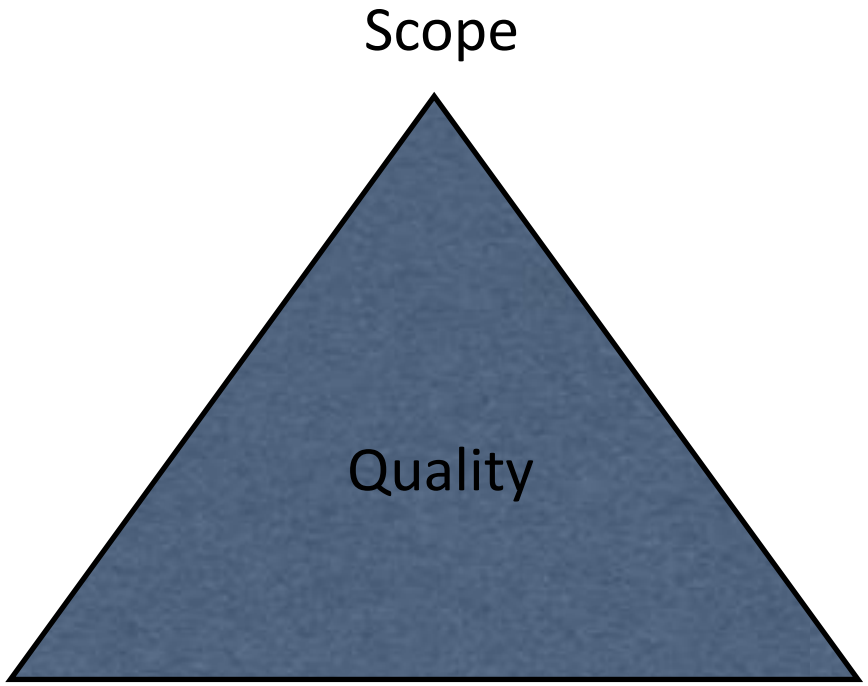
Schedule

Scope



Cost

Schedule



Cost

Schedule

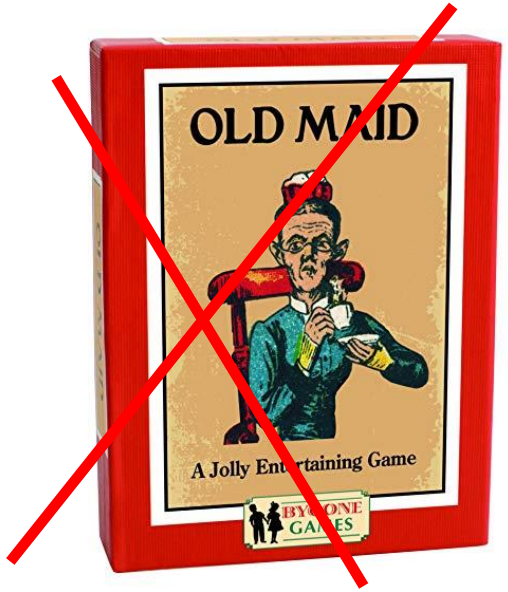
IBM 5150 PC with IBM 5151 monitor

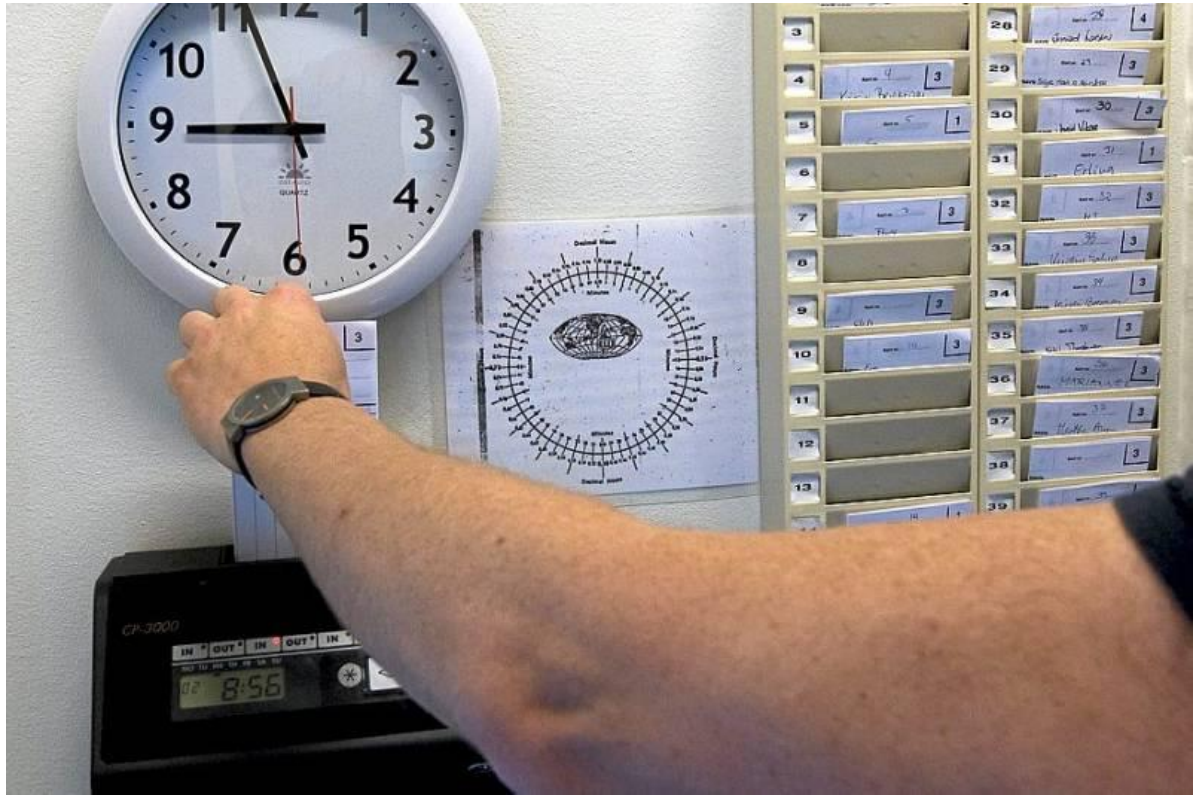
Lock cost and schedule, but not scope

Training

Always train your best developers first. The others will follow.

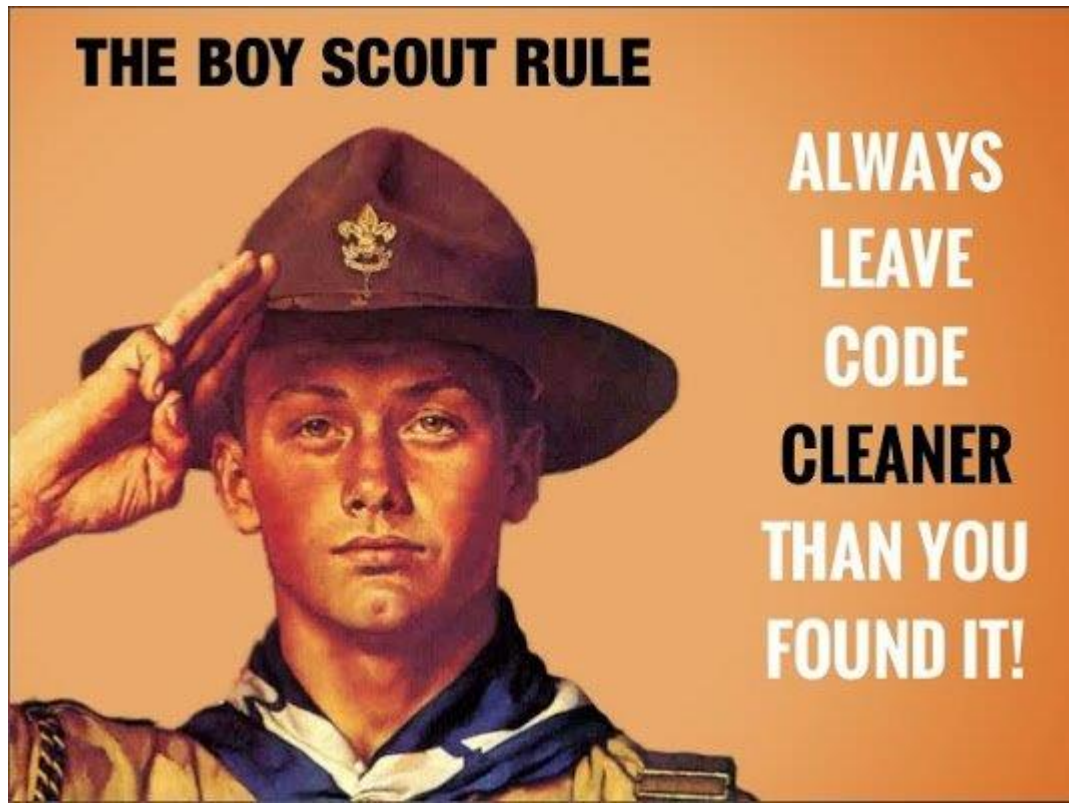
Remove mechanisms for shifting responsibility around





"What we don't do is treat our employees like they're all, you know, criminals,"

(Jenn Mann, SAS Institute)


THE BOY SCOUT RULE
ALWAYS
LEAVE
CODE
CLEANER
THAN YOU
FOUND IT!

Grow Professionalism!

what *successful* organization need to do

make sure there is enough slack

encourage teams and individuals to meet and spend time together

instead of setting objectives, show your constraints and suggest a direction

respect the observer effect, get rid of externally imposed metrics

share your vision, never throw requirements "over the wall"

pull value out of a system, do not push

explain the business to everyone involved

don't celebrate the midnight cowboys

do not demand dates and estimates, aka *"do not rush miracles"*

respect that software development is a learning process

beware of governance and corporate coding standards

abandon decision gates and commitments

you may lock time and cost, but do not lock the scope

train your best developers first

get rid of mechanisms for shifting responsibilities around

don't treat your employees as criminals

what *aspiring* software craftsmen need do

think, use tools, create tools, focus on effectiveness

share knowledge, collaborate, *fremsnakke* and extend trust boundaries

align your efforts and make sure you all pull in the same direction

make sure you have enough data to know where you are going

understand the business, take overall responsibility for what you create

deliver value early and often, continuously improve

stay close to where the money is flowing, avoid the turbulence

work in a sustainable pace, build quality in, celebrate your vision

no sandbagging, be honest and be transparent, deliver magic

establish fast and reliable feedback loops, celebrate failures

stay up to date with current industry standards and best practice

establish reliable breaking mechanisms, fail fast, stop or change direction

deliver value early and often, without compromising the overall vision

share your knowledge, teach, encourage others to follow your ideas

be trustworthy, practice collective ownership and responsibility

always do the right things

Grow Professionalism!

what *successful* organization need to do

make sure there is enough slack

encourage teams and individuals to meet and spend time together

instead of setting objectives, show your constraints and suggest a direction

respect the observer effect, get rid of externally imposed metrics

share your vision, never throw requirements "over the wall"

pull value out of a system, do not push

explain the business to everyone involved

don't celebrate the midnight cowboys

do not demand dates and estimates, aka *"do not rush miracles"*

respect that software development is a learning process

beware of governance and corporate coding standards

abandon decision gates and commitments

you may lock time and cost, but do not lock the scope

train your best developers first

get rid of mechanisms for shifting responsibilities around

don't treat your employees as criminals

what *aspiring* software craftsmen need do

think, use tools, create tools, focus on effectiveness

share knowledge, collaborate, *fremsnakke* and extend trust boundaries

align your efforts and make sure you all pull in the same direction

make sure you have enough data to know where you are going

understand the business, take overall responsibility for what you create

deliver value early and often, continuously improve

stay close to where the money is flowing, avoid the turbulence

work in a sustainable pace, build quality in, celebrate your vision

no sandbagging, be honest and be transparent, deliver magic

establish fast and reliable feedback loops, celebrate failures

stay up to date with current industry standards and best practice

establish reliable breaking mechanisms, fail fast, stop or change direction

deliver value early and often, without compromising the overall vision

share your knowledge, teach, encourage others to follow your ideas

be trustworthy, practice collective ownership and responsibility

always do the right things (even when nobody is looking)

!