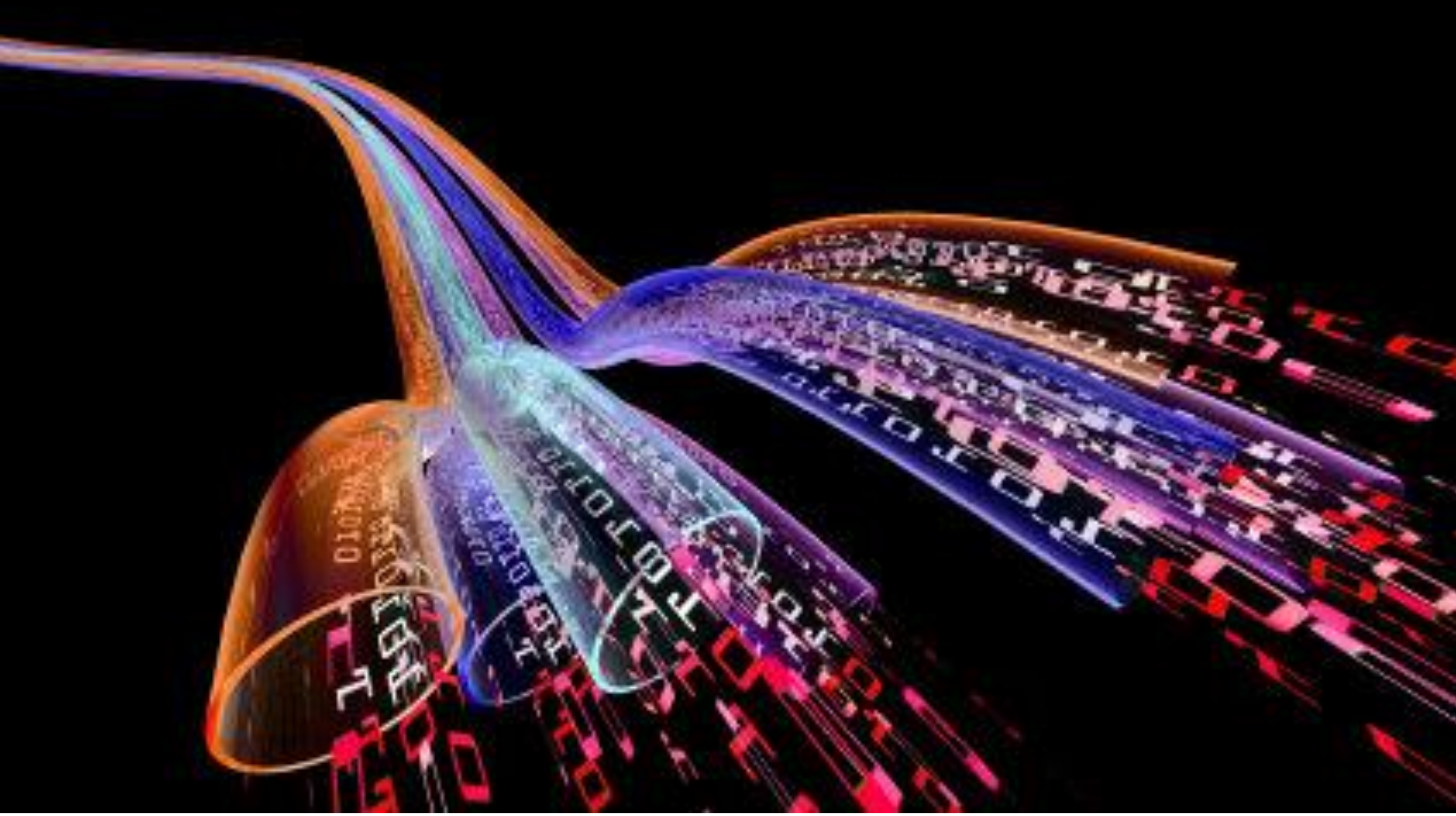
CONFIGURA

PGC, Quality and Programming Style

Be smart and create a quality oriented culture to save money, gain efficiency and get ahead of your competition!

Göran Rydqvist

Co-founder and Vice President Research and Development of Configura. More than 40 years of computer programming experience. Architect of the CM programming language - the foundation of CET Designer. Specializes in Dynamic Syntax & Metaprogramming, Large System Programming, UI Design and Parametric Manufacturing. Master of Science from Linköping Institute of Technology (LiTH) (1984-1987). PhD Student in Hardware Synthesis LiTH (1987-1889) including 6 months at Xerox Palo Alto Research Center in Palo Alto, California (1989). Co-founded Configura 1990.



PGC, Quality and Programming Style

PGC

Parametric Graphical Configuration

The flow and ease of PGC

PGC

PGC is a solution development framework for the implementation of quick, efficient, and intuitive graphical configuration software customized to specific products and solution domains.

- Parametric - a set of properties whose values determine the characteristics or behavior of something
- Graphical - a visual representation in 2D or 3D
- Configuration - a relative arrangement of parts or elements

HOW TO MANAGE VARIANT CASES AND CONFIGURATION PROBLEMS
THE STRENGTH OF
PGC
PARAMETRIC GRAPHICAL CONFIGURATION
SUNE RYDQVIST
JOHAN LYREBORN
GÖRAN RYDQVIST

PGC Fundamentals

- Flowing ease
- Touch and feel
- Direct manipulation
- Interpret gestures
- Assist
- Remember all input,
- Explorative
- Encourage experimentation

PGC - Condensed UI

- Small initial UI
- High polymorphism
- Few choices – large number of design posibilities
- Play, interact, explore and design

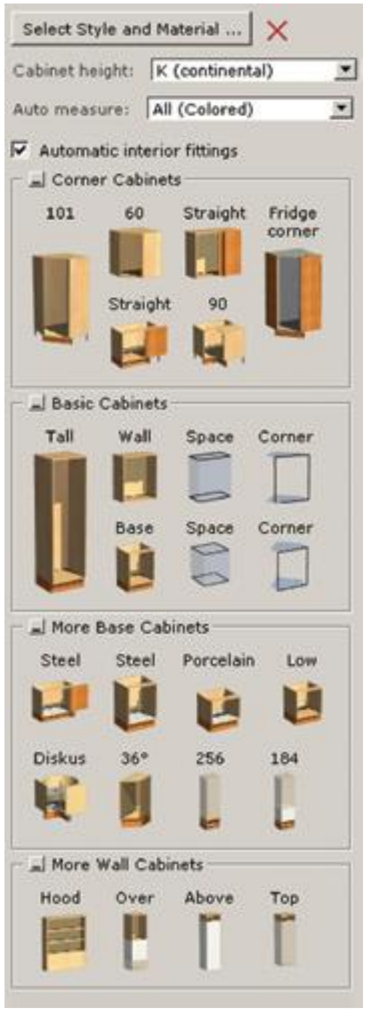
PGC - Immersion

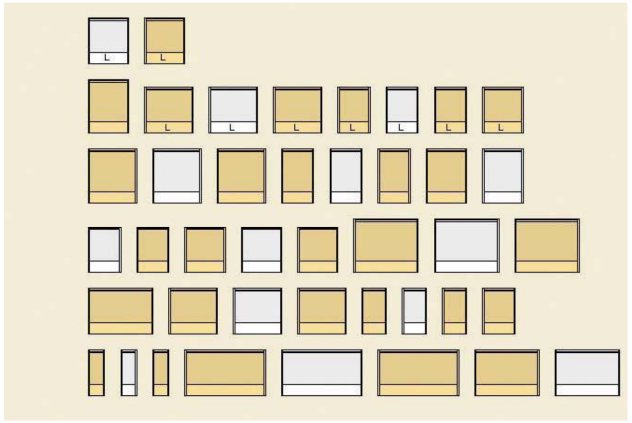
- Pull the user into an immersive experience
- Aesthetical, technical and other constraints and requirements
- Uninterrupted

u

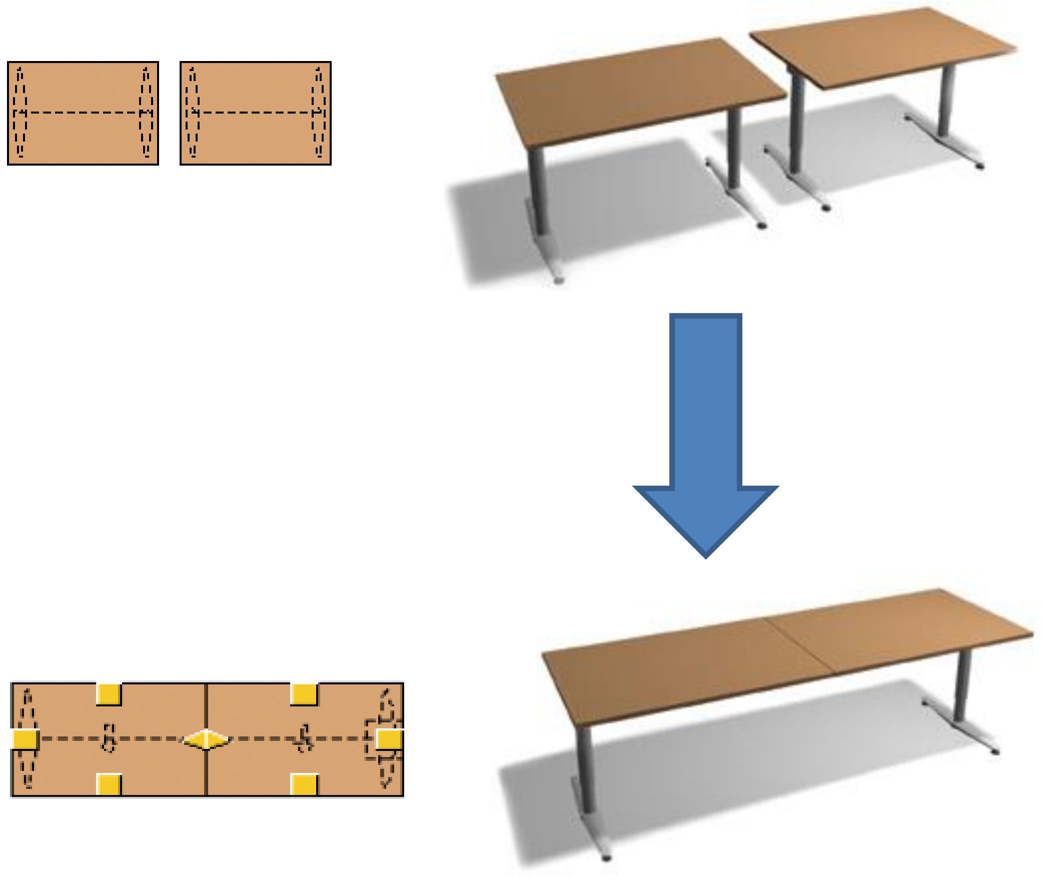
e

PGC Fundament: Polymorphism

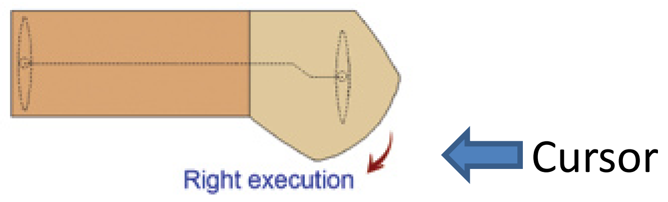


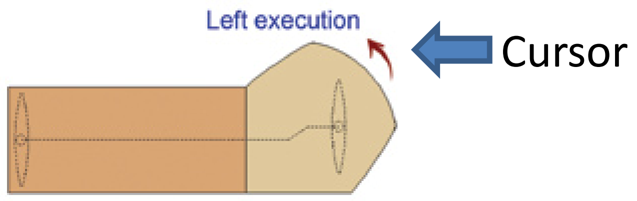


PGC Fundament: Connection Rules

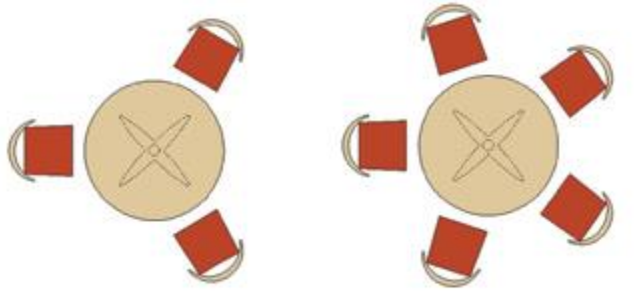


PGC Fundament: Mirror Selection

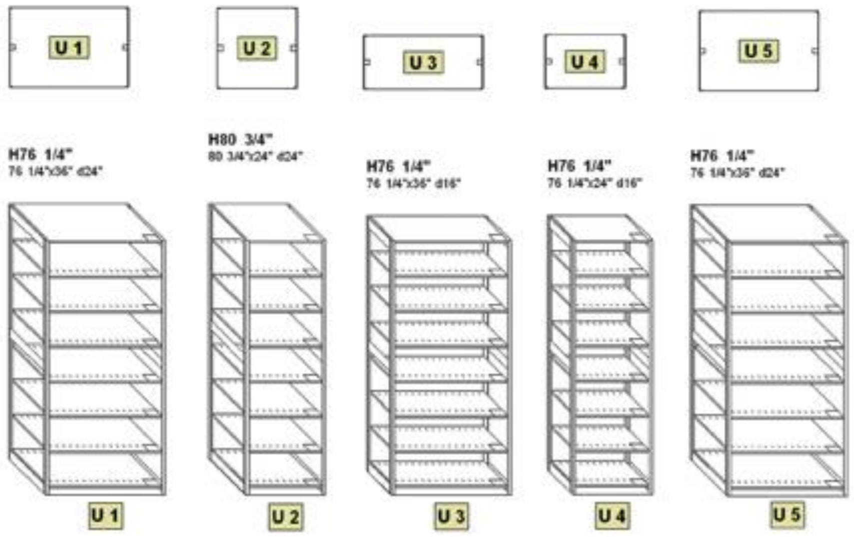




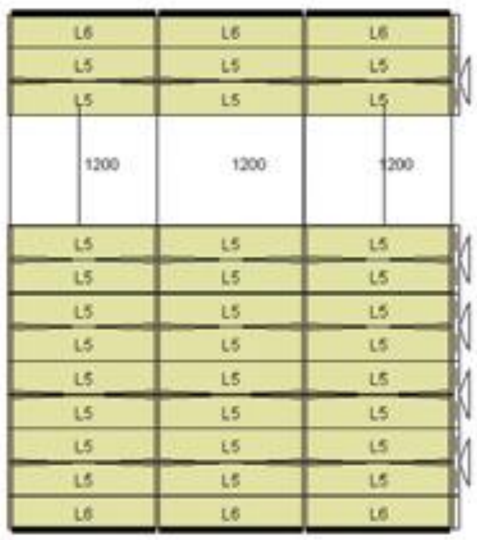
PGC Fundament: Auto Distribution



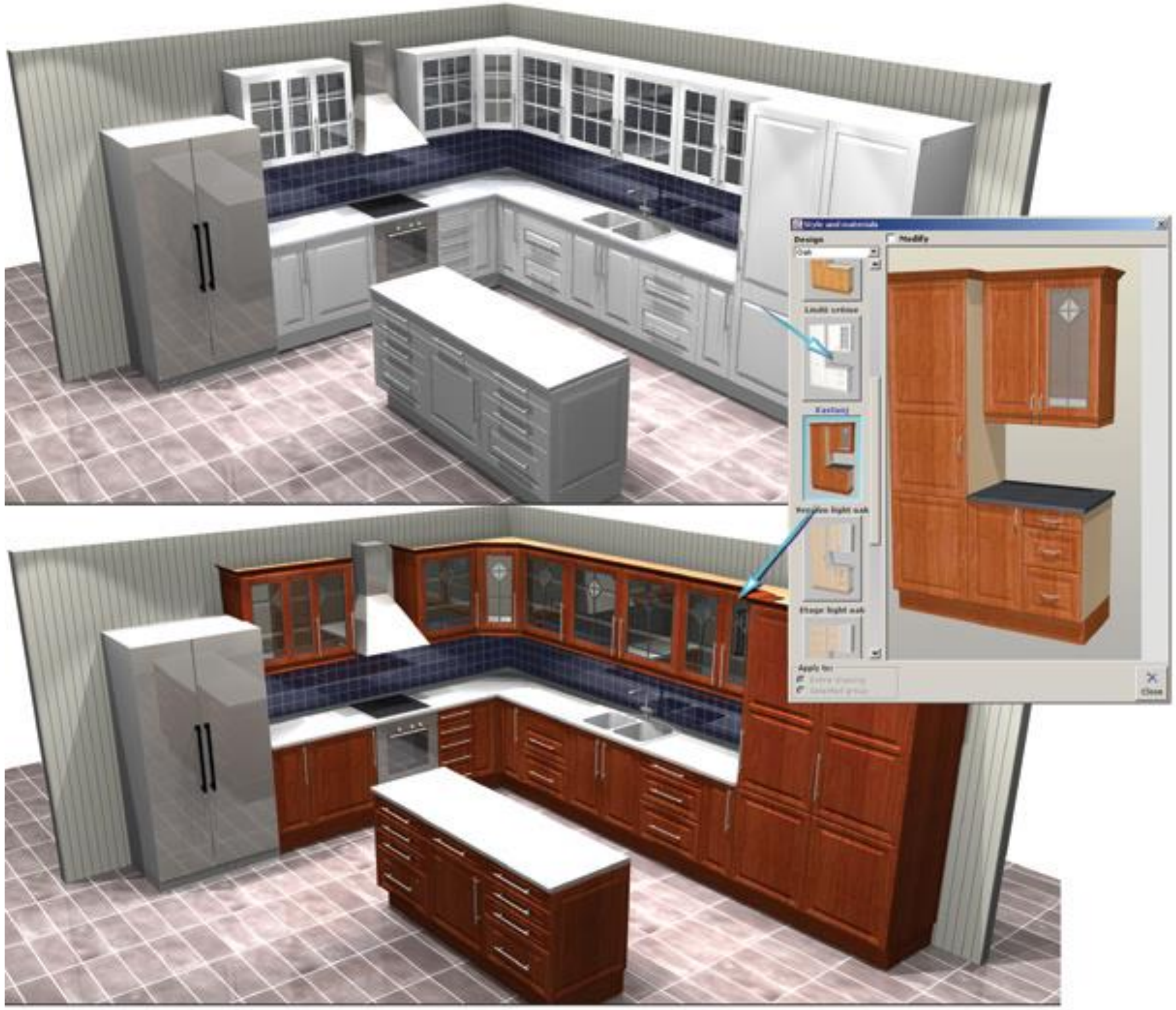
PGC Fundament: Information Reuse



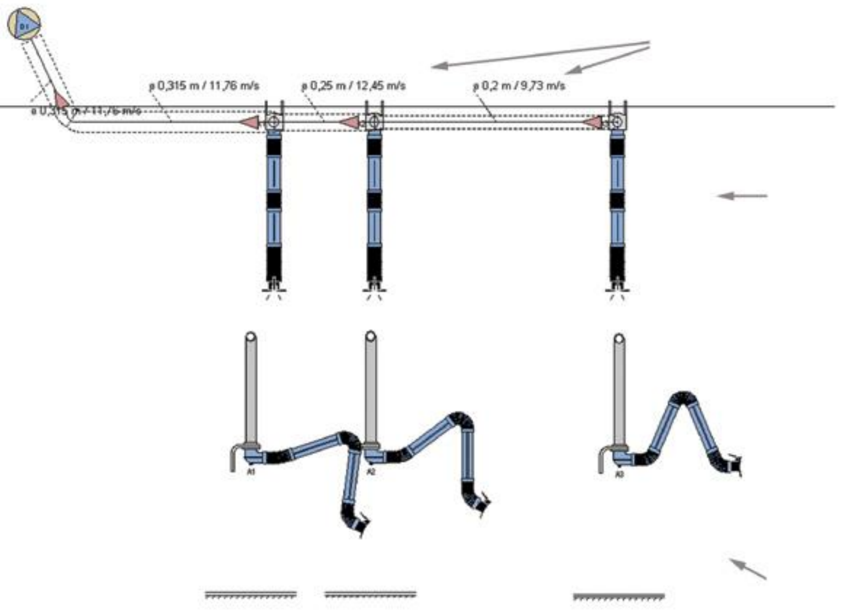
PGC Fundament: Auto Create

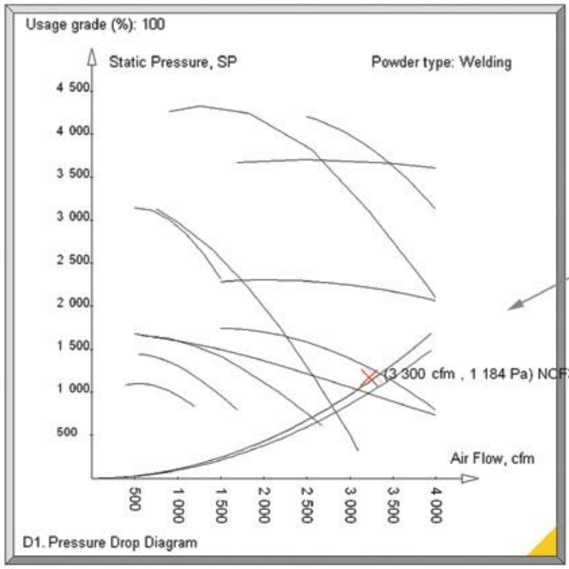


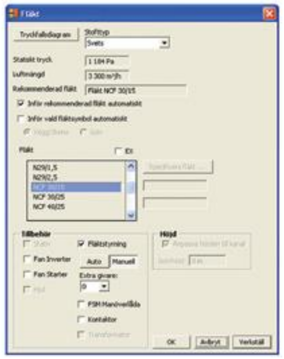
PGC Fundament: Global Change



PGC Fundament: Technical Calculations







Stofftyp: Svets

Statiskt tryck: 1 184 Pa

Luftmängd: 3 300 m³/h

Rekommenderad fläkt: Fläkt NCF 30/15

✓ Inför rekommenderad fläkt automatiskt

☐ Inför vald fläktsymbol automatiskt

Fläkt ☐ EX

- N29/1,5
- N29/2,5
- NCF 30/15
- NCF 30/25
- NCF 40/25

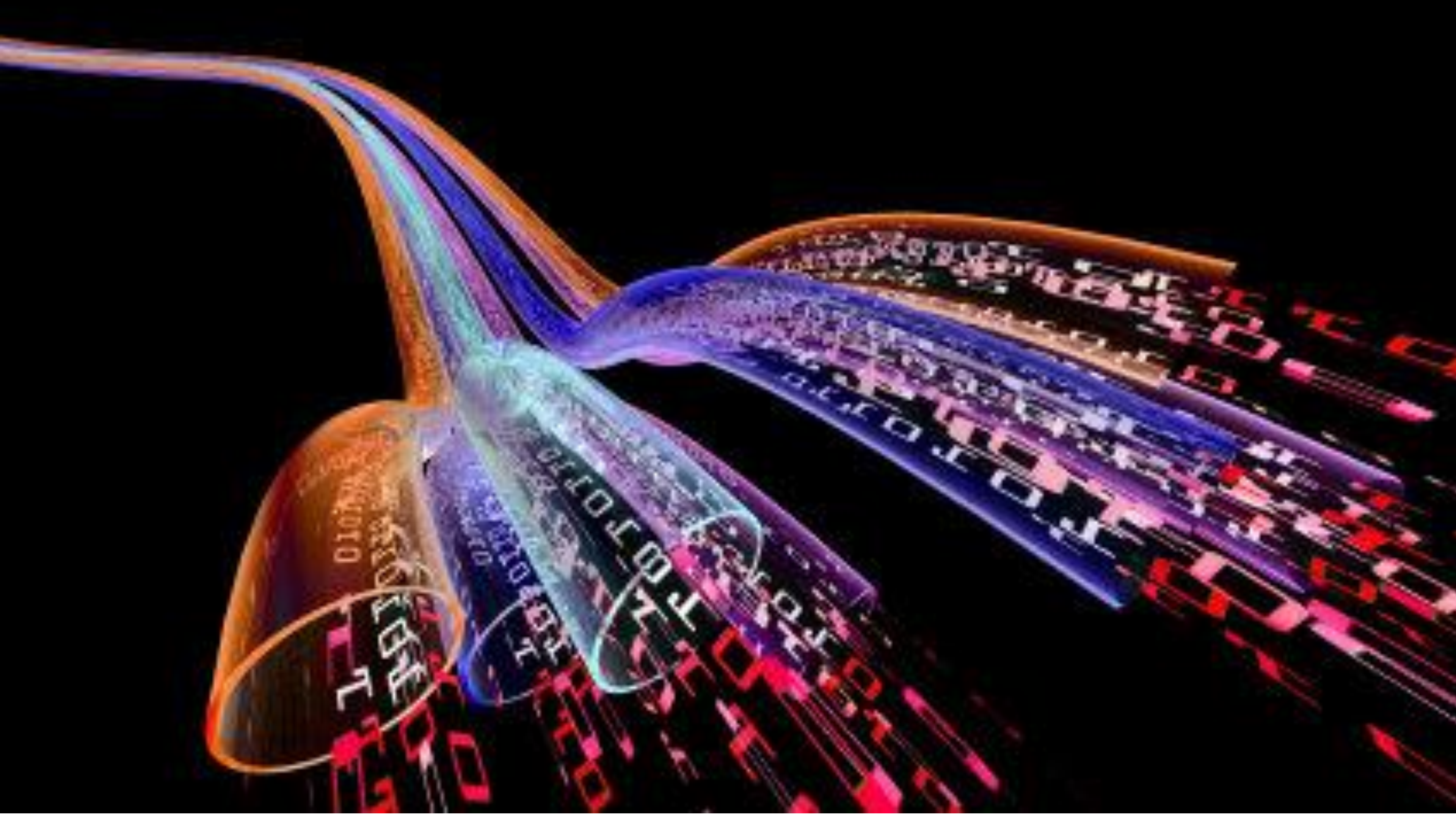
Tillbehör

- ☐ Fan Inverter
- ☐ Fan Starter
- ✓ Fläktstyrning
- Auto [✓ Manuell]
- Extra givare: 0
- ☐ FSM Manöverlåda
- ☐ Kontaktor
- ☐ Transformator

Höjd

PGC

Quality

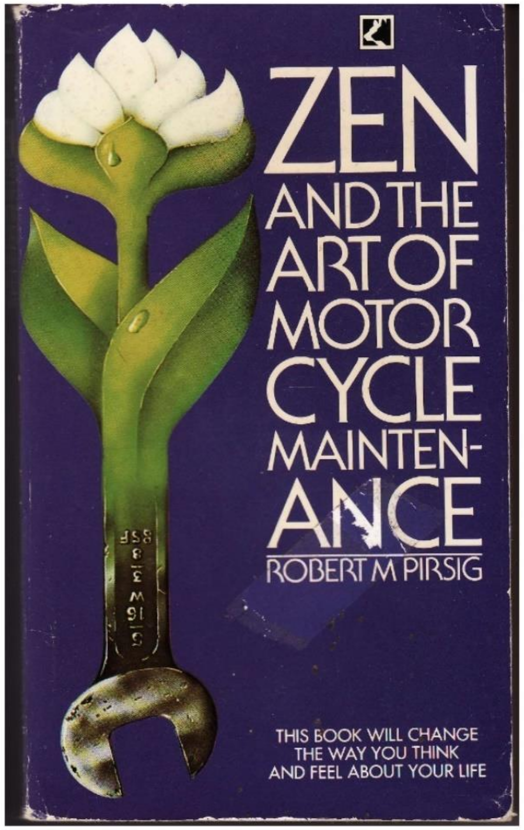


PGC, Quality and Programming Style

Quality

What is Quality?

- Phaedrus (after Plato's dialogue), a teacher of creative and technical writing at a small college
- became engrossed in the question of what defines good writing
- and what in general defines good, or "Quality".
- His philosophical investigations eventually drove him insane
- and he was subjected to electroconvulsive therapy which permanently changed his personality

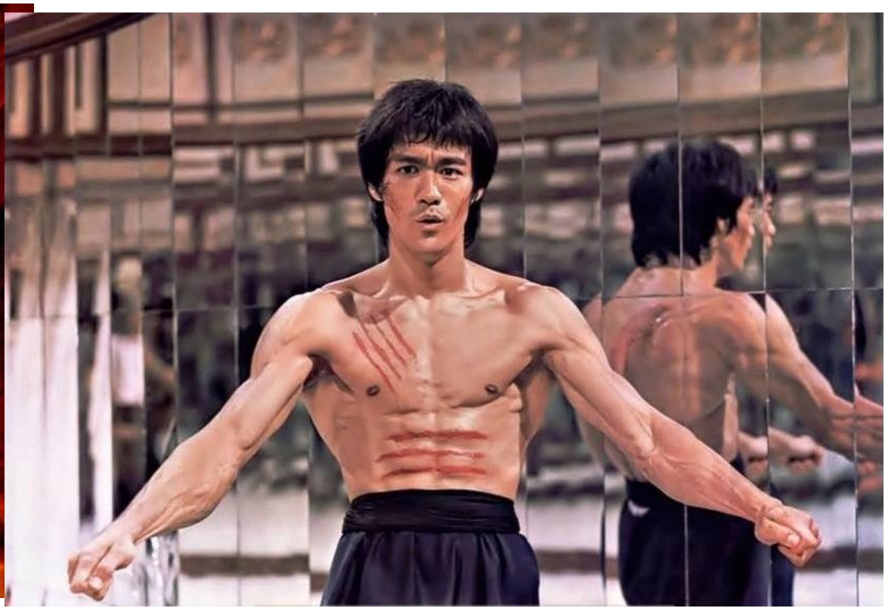

ZEN AND THE ART OF MOTOR CYCLE MAINTEN- ANCE
ROBERT M PIRSIG
THIS BOOK WILL CHANGE THE WAY YOU THINK AND FEEL ABOUT YOUR LIFE

The book sold 5 million copies worldwide. It was originally rejected by 121 publishers, more than any other bestselling book

Quality vs Kung Fu

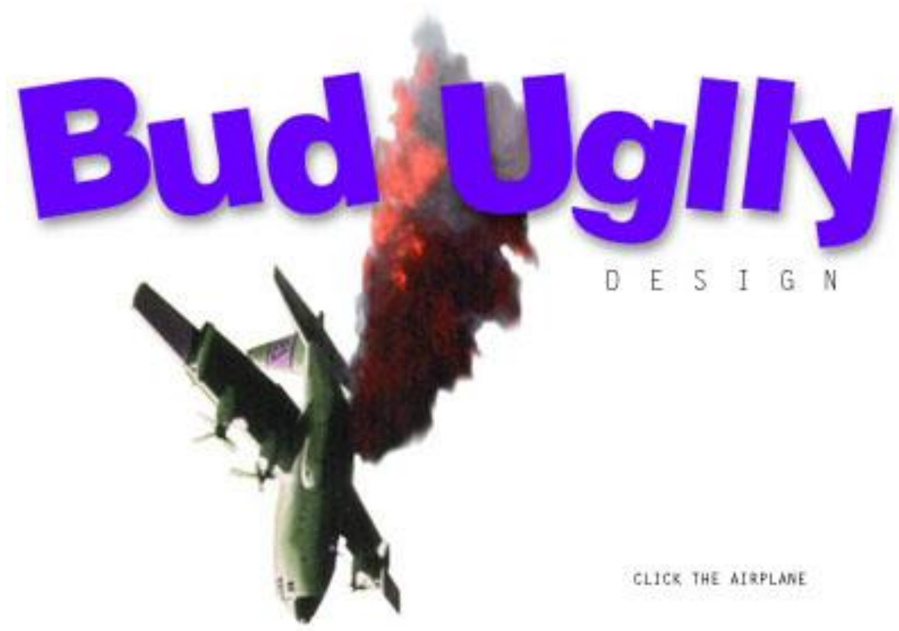
- Pirsig: quality is undefinable
- Webster: a high level of value or excellence

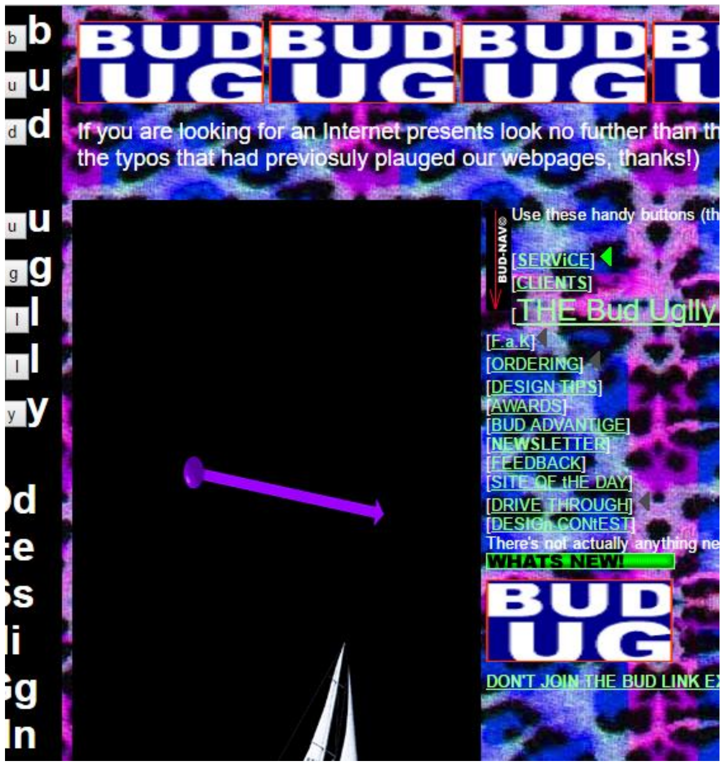
the standard of something as measured against other things of a similar kind; the degree of excellence of something.



“Good enough”

http://budugllydesign.com/



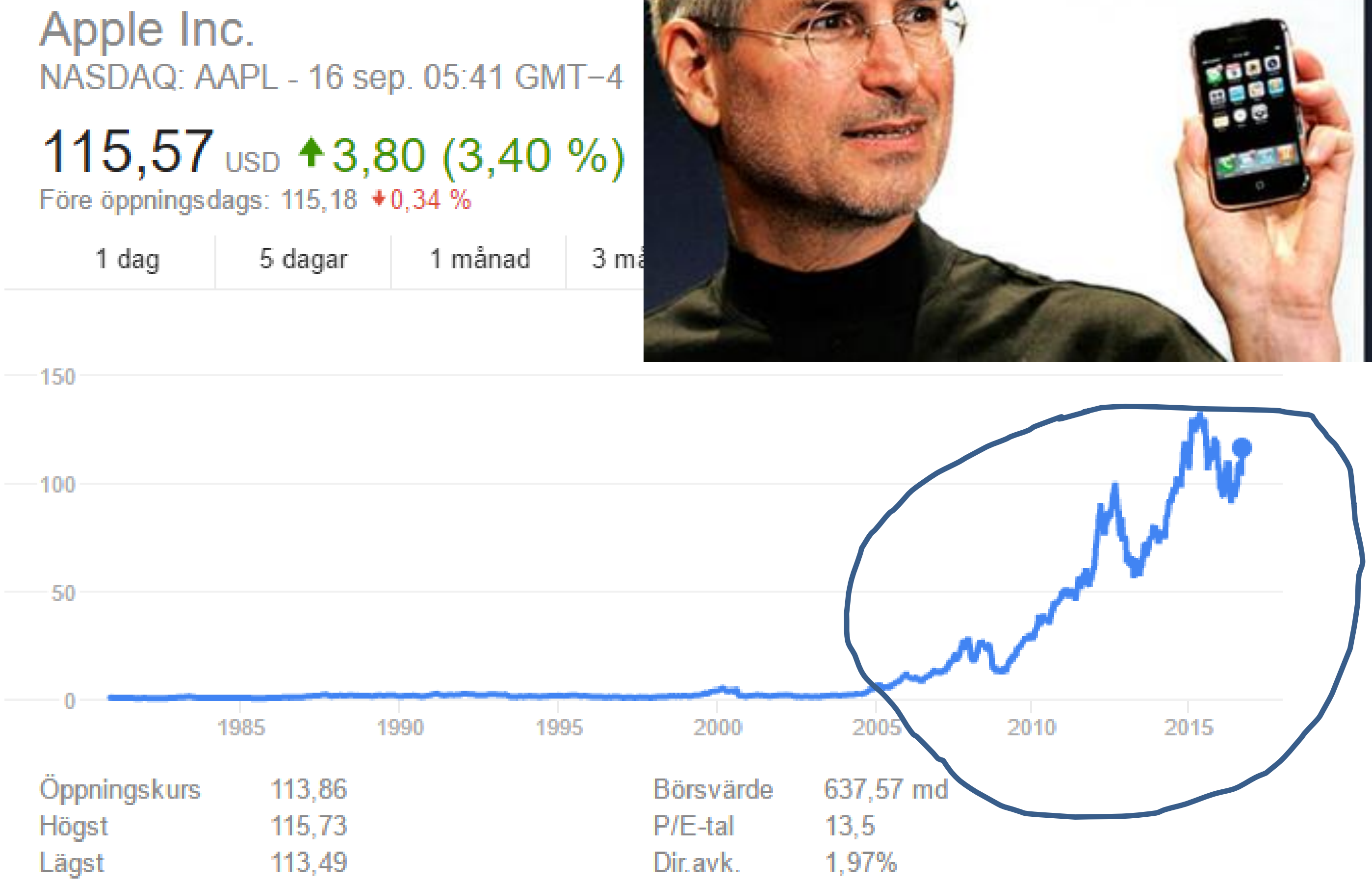


If you are looking for an Internet presents look no further than th
the typos that had previosuly plauged our webpages, thanks!)

Use these handy buttons (th

There's not actually anything ne

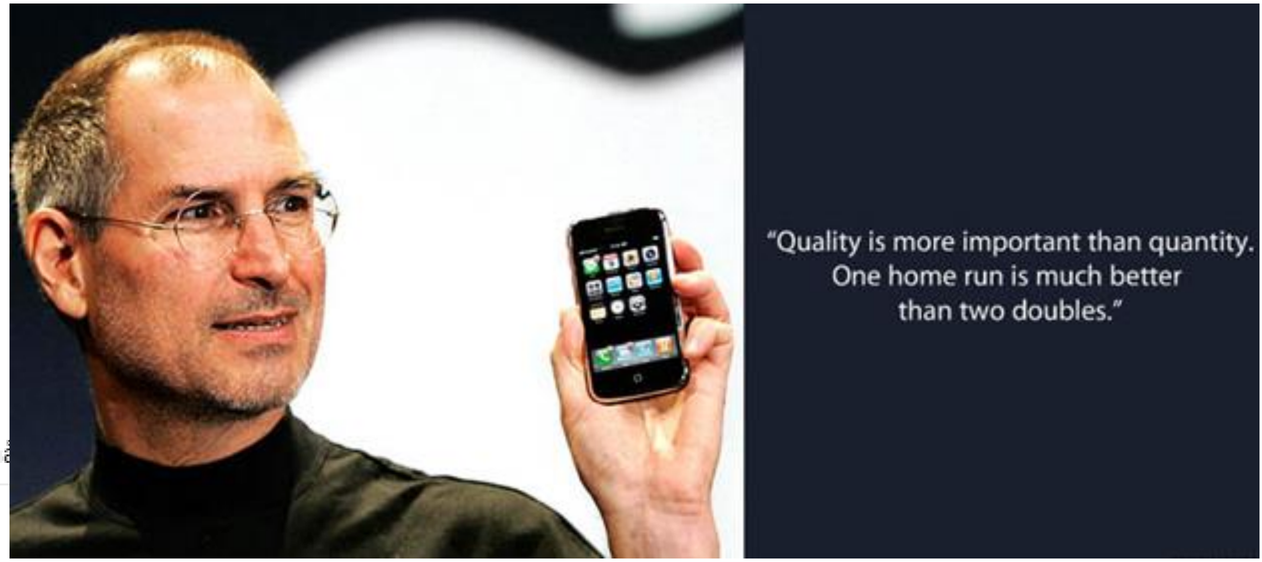
"Good enough" - why should I care?



Apple Inc.
NASDAQ: AAPL - 16 sep. 05:41 GMT−4

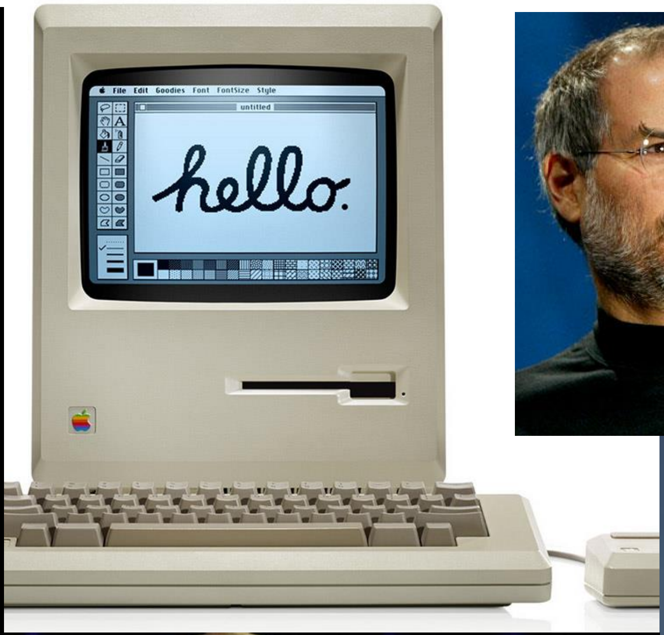
115,57 USD ↑3,80 (3,40 %)
Före öppningsdags: 115,18 ↓0,34 %

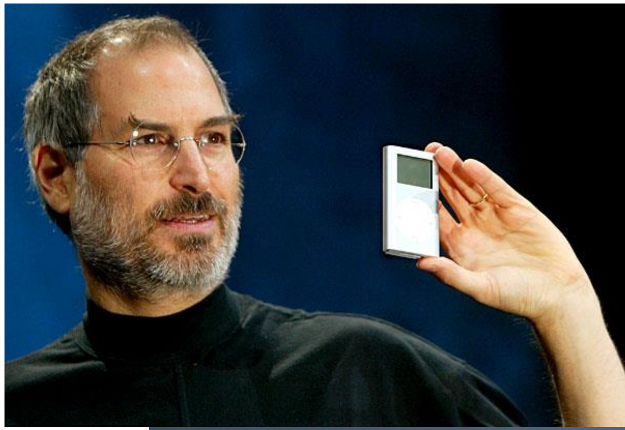
Öppningskurs	113,86	Börsvärde	637,57 md
Högst	115,73	P/E-tal	13,5
Lägst	113,49	Dir.avk.	1,97%

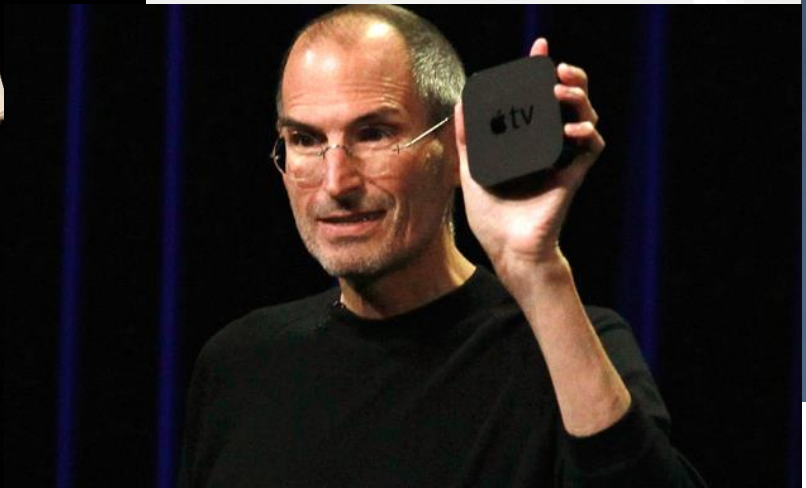


"Quality is more important than quantity. One home run is much better than two doubles."

A “Little” Success!

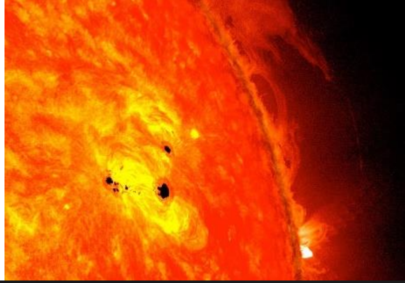


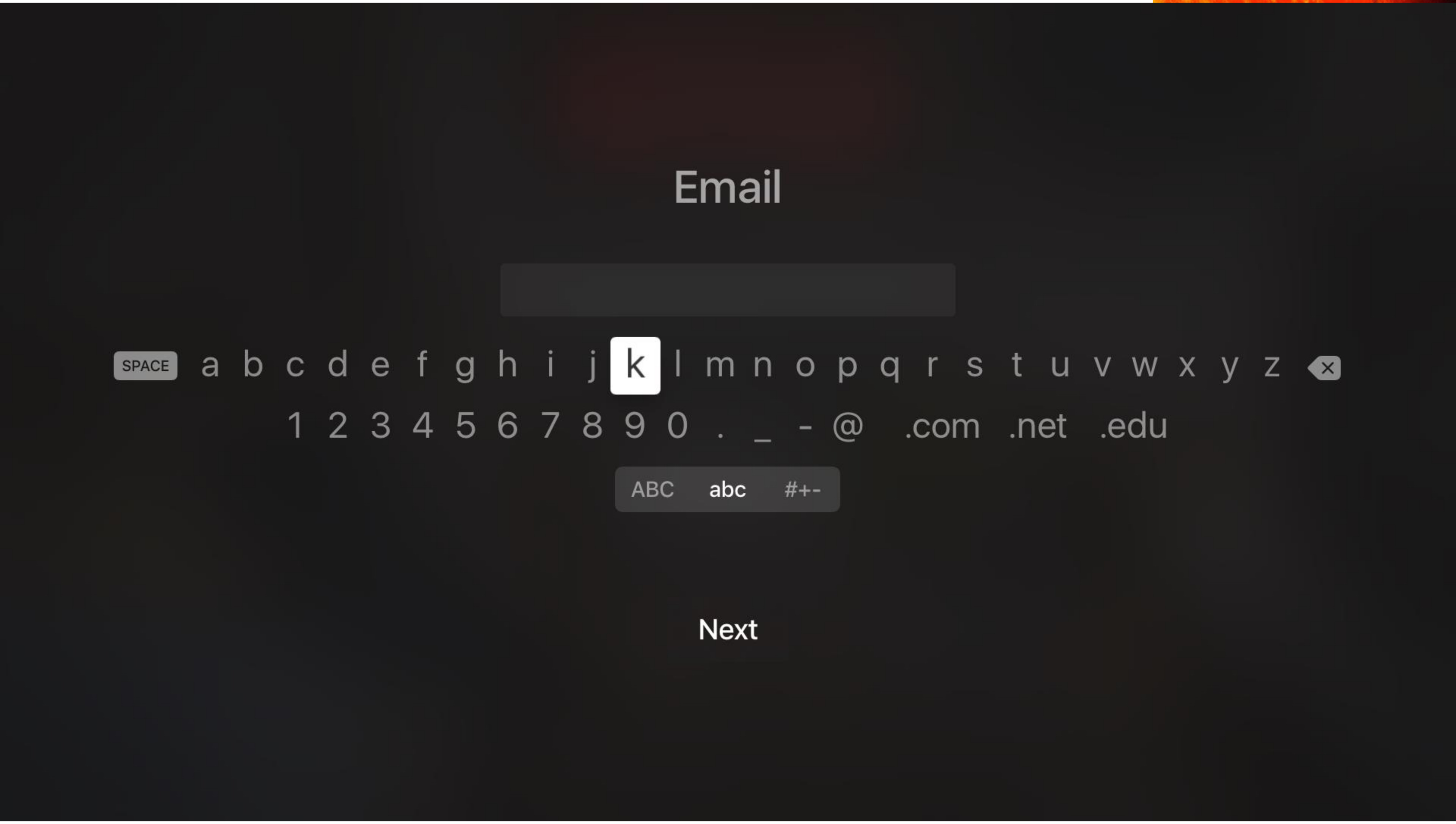




Apple Unboxing

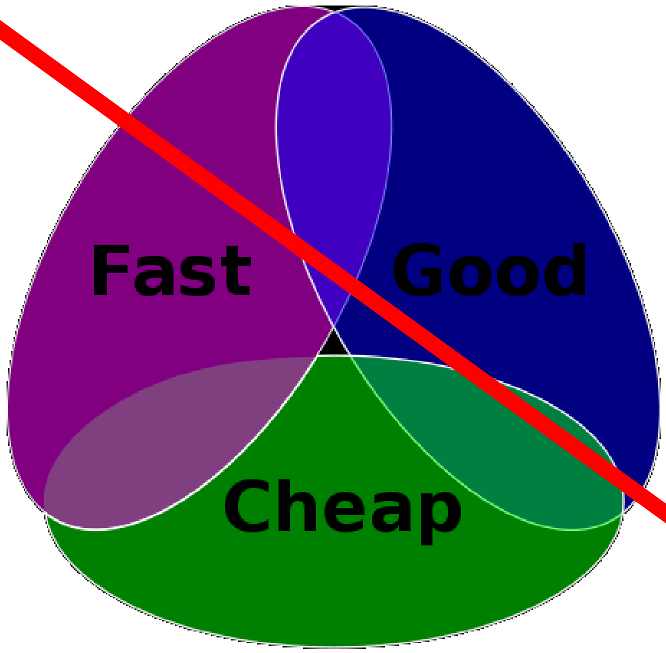
Even the Sun has Spots




Email
SPACE
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 . _ - @ .com .net .edu
ABC abc #+-
Next

Traditional TCQ

- ~~Design quickly and high quality -> expensive~~
- ~~Design quickly and cheaply -> low quality~~
- ~~Design cheaply and high quality -> long time~~

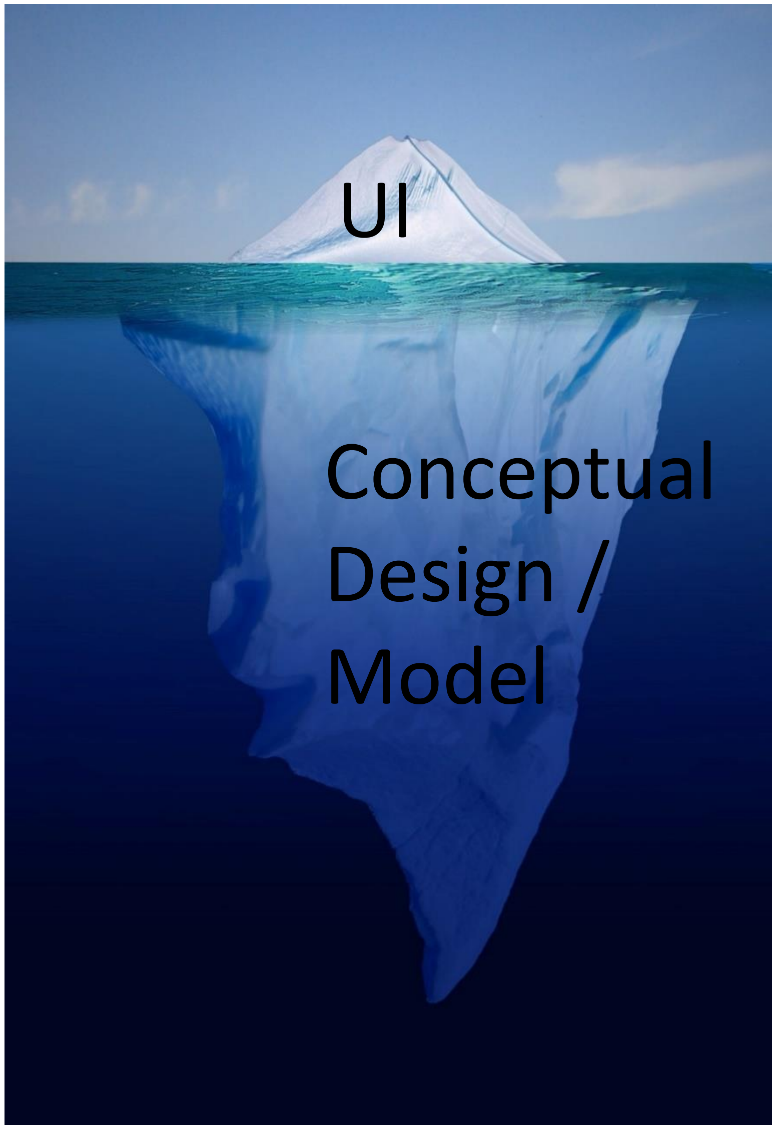


Quality Driven Development

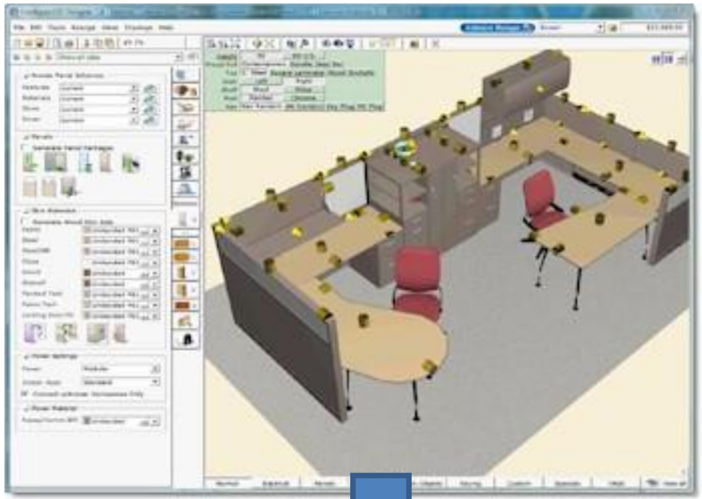
- Quality driven all the way through
- Bounds time
- Bounds cost

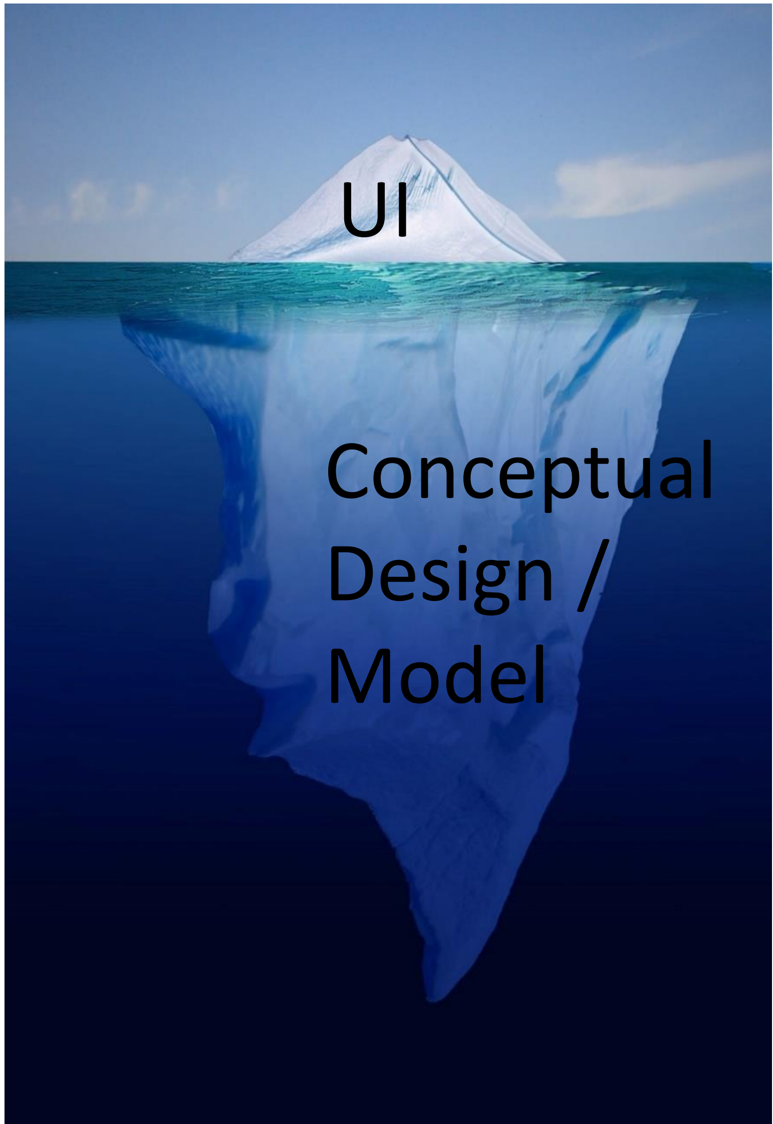
Low Quality ~~Code~~ quickly drives cost and time towards unreasonable levels

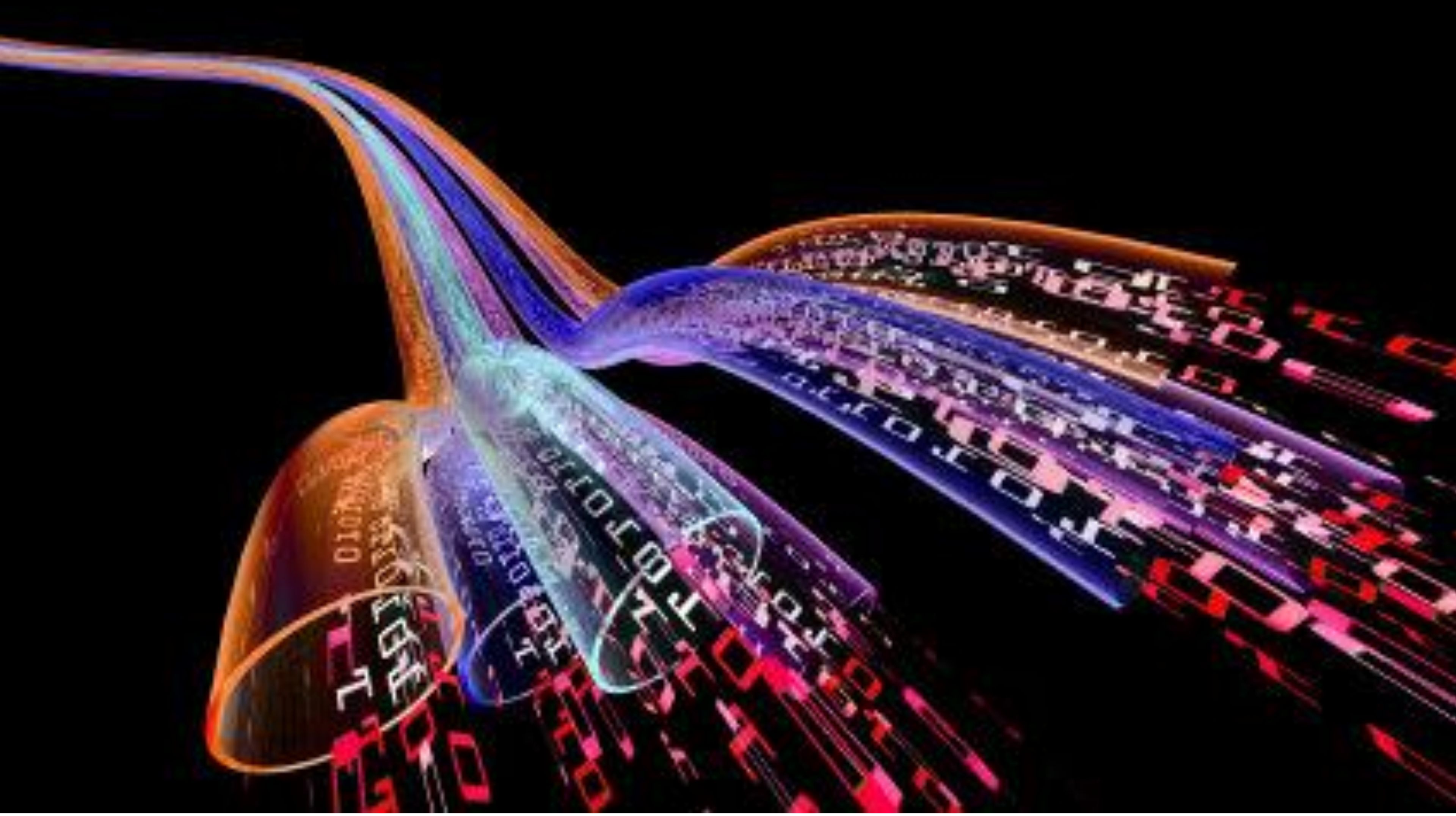
What is a UI?



What is a UI?







PGC, Quality and Programming Style

Programming Style

Large System Development

- Programs are built from language definitions and then BY combining these into larger systems.
- If the language is precise, consistent and orthogonal large systems can be described with little or medium effort.
- Imprecise language quickly increases the effort of understanding.
- Imprecise programs become full of fixes. Complexity increases.
- Fixes create unwanted side-effects (bugs/issues).
- Cost goes through the roof.

Programming Style

I will tell you my secret!

From a lifetime of programming ..

You will be disappointed ..

Programming Style
The Secret

The longer I work with programming

- Simplicity
- ~~Complication~~
- ~~Debugging~~

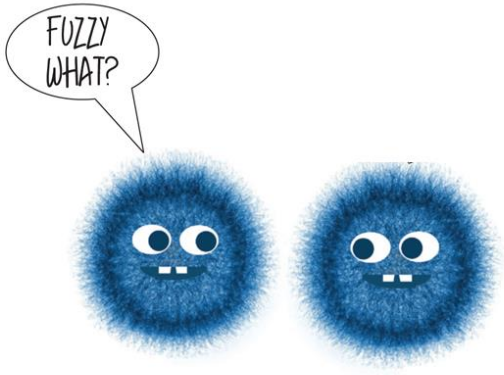
Programming Style
The Secret

How do you achieve simplicity

- Language Precision

the more precisely you can define your vocabulary used to describe the problem the easier it becomes

Language

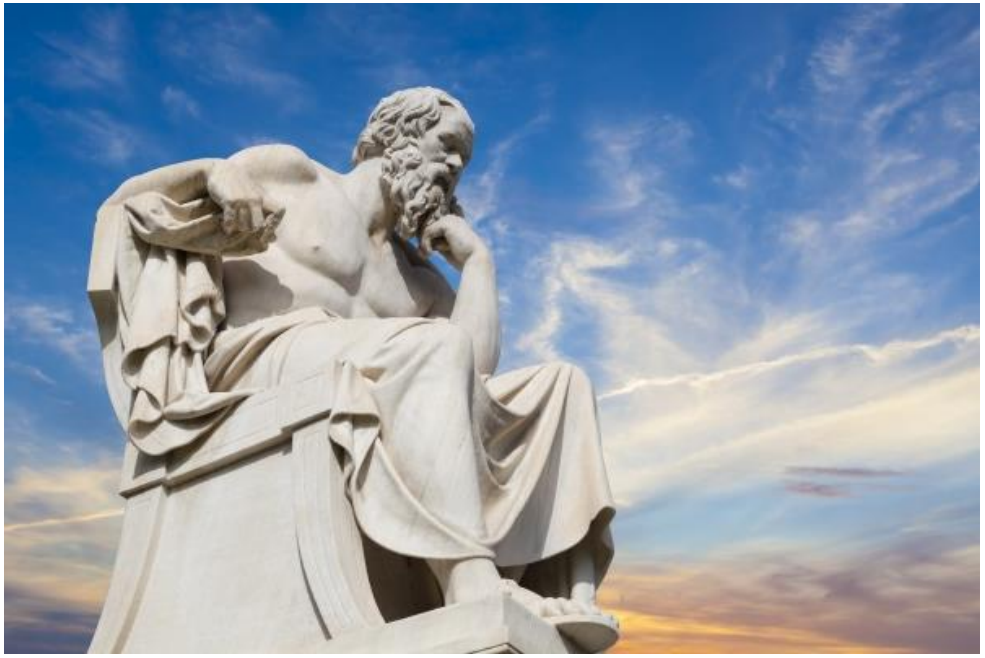


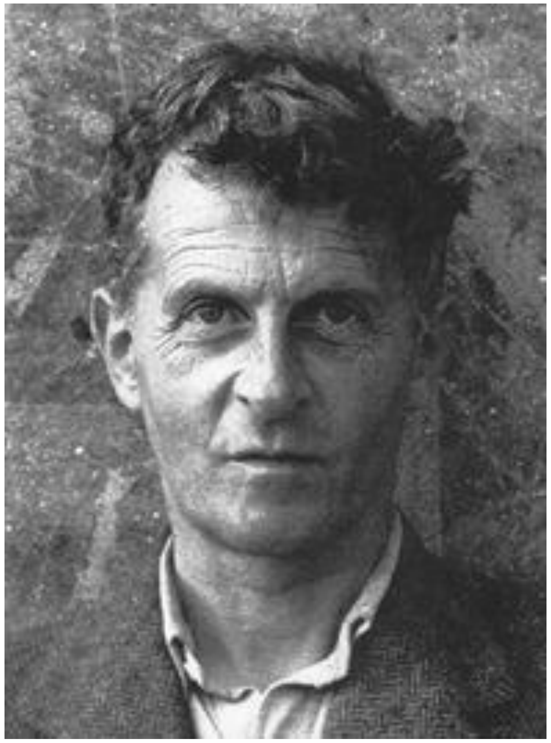
Natural language

Mathematics

Wittgenstein

The early Wittgenstein was concerned that all philosophical problems arise from misconception of language

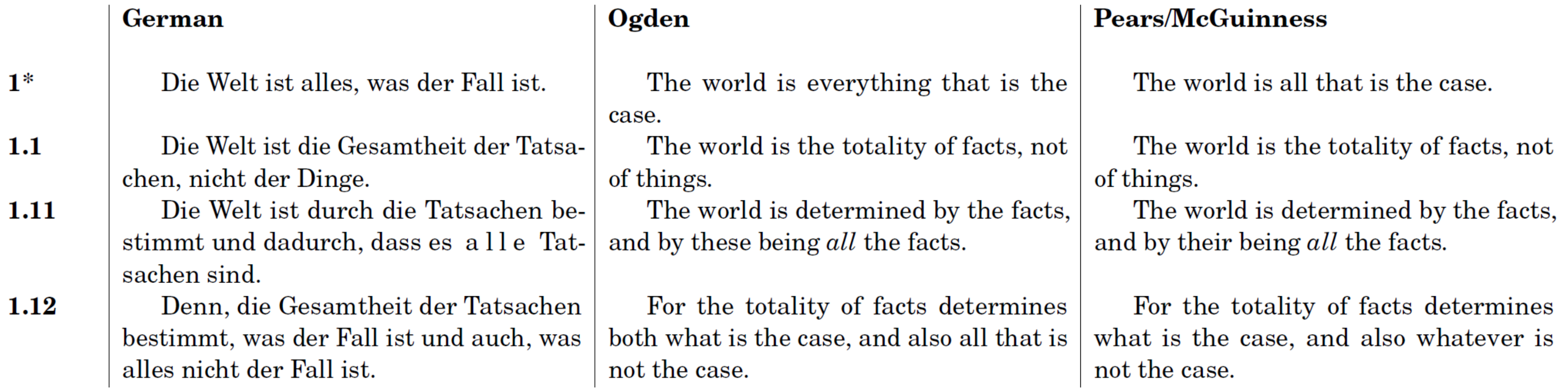




So he set out to fix that

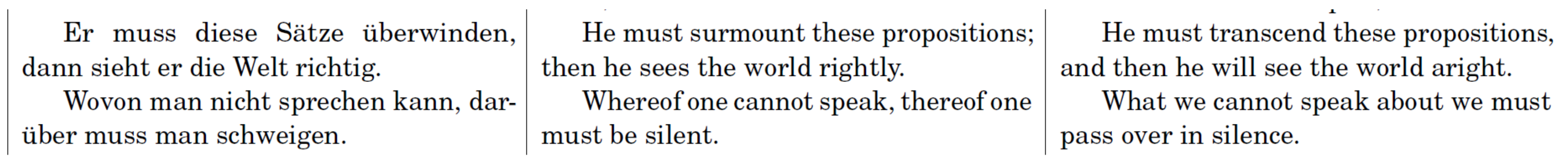
Wittgenstein Tractatus Logico-Philosophicus

Start



	German	**Ogden**	**Pears/McGuinness**
1*	Die Welt ist alles, was der Fall ist.	The world is everything that is the case.	The world is all that is the case.
1.1	Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.	The world is the totality of facts, not of things.	The world is the totality of facts, not of things.
1.11	Die Welt ist durch die Tatsachen bestimmt und dadurch, dass es a l l e Tatsachen sind.	The world is determined by the facts, and by these being *all* the facts.	The world is determined by the facts, and by their being *all* the facts.
1.12	Denn, die Gesamtheit der Tatsachen bestimmt, was der Fall ist und auch, was alles nicht der Fall ist.	For the totality of facts determines both what is the case, and also all that is not the case.	For the totality of facts determines what is the case, and also whatever is not the case.

End



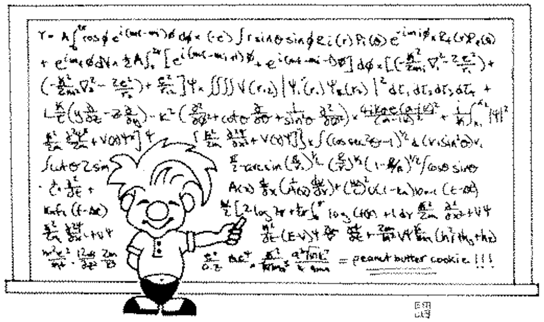
Er muss diese Sätze überwinden, dann sieht er die Welt richtig. Wovon man nicht sprechen kann, darüber muss man schweigen.	He must surmount these propositions; then he sees the world rightly. Whereof one cannot speak, thereof one must be silent.	He must transcend these propositions, and then he will see the world aright. What we cannot speak about we must pass over in silence.

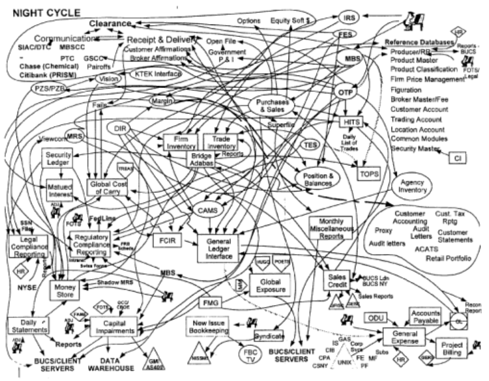
Programmers

```
#include <stdio.h>
#include "C:/Privacy.h"
#define 0 CORRUPTED
#define 1 NORMALY_ENDED
.
int OnButtonClick(UserID, *Se
SelectedAlgor
```

CONFIGURA

Programmers



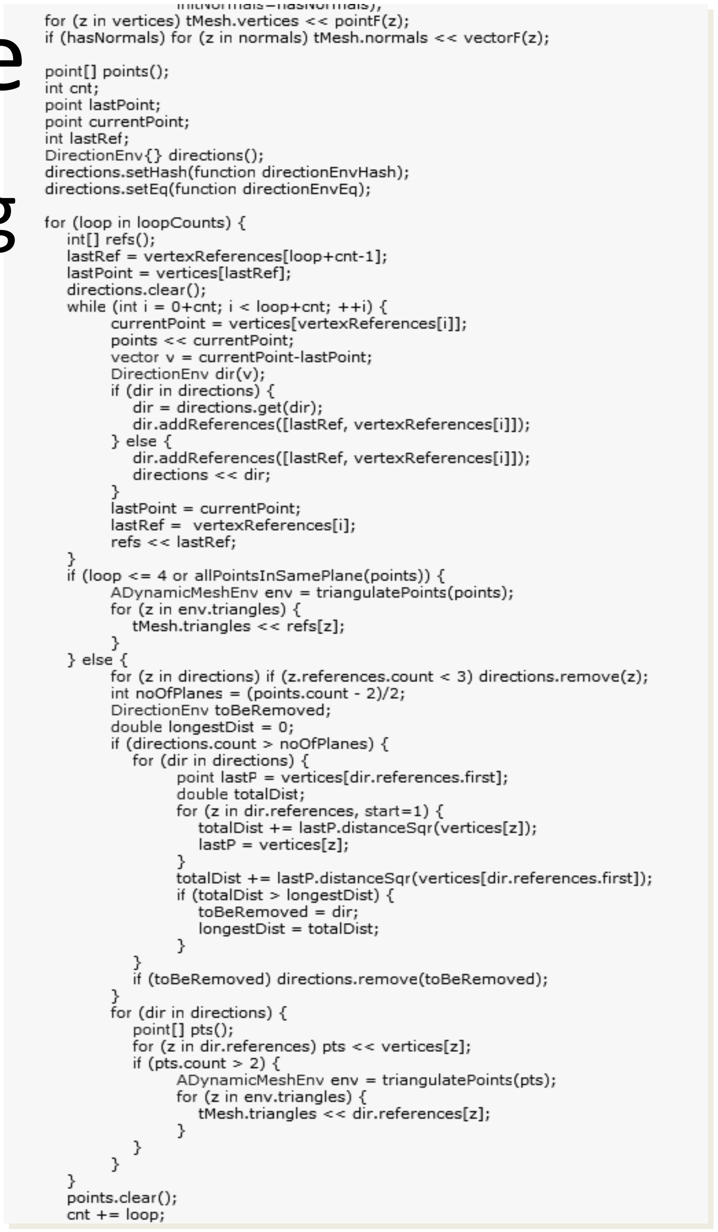


- Very hi IQ – complicated minds
- Generally striving towards disorder
- So proud of solving bug X357
- So proud of creating function Y221
- Problem subroutine jumping

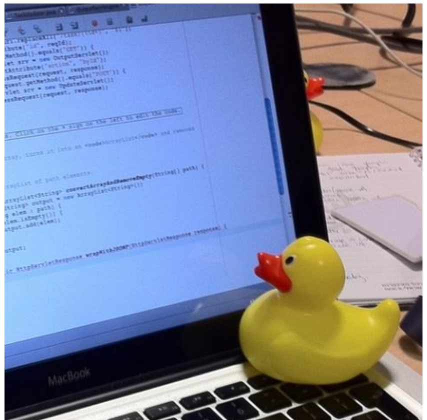
Programming Value Chain

Prototype

- A messy piece of code doing something
- Totally dependent on the programmer
- Debug time dominates
- Reusability Zero

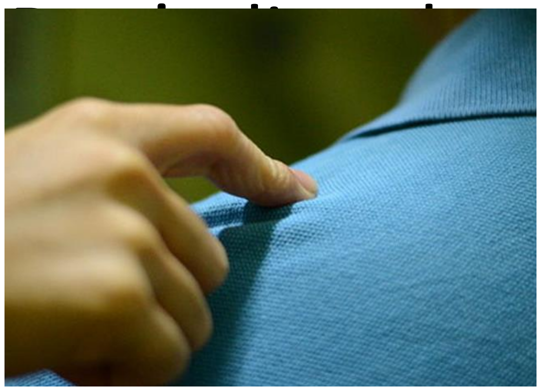


```
initNormals=hasNormals);
for (z in vertices) tMesh.vertices << pointF(z);
if (hasNormals) for (z in normals) tMesh.normals << vectorF(z);

point[] points();
int cnt;
point lastPoint;
point currentPoint;
int lastRef;
DirectionEnv{} directions();
directions.setHash(function directionEnvHash);
directions.setEq(function directionEnvEq);

for (loop in loopCounts) {
    int[] refs();
    lastRef = vertexReferences[loop+cnt-1];
    lastPoint = vertices[lastRef];
    directions.clear();
    while (int i = 0+cnt; i < loop+cnt; ++i) {
        currentPoint = vertices[vertexReferences[i]];
        points << currentPoint;
        vector v = currentPoint-lastPoint;
        DirectionEnv dir(v);
        if (dir in directions) {
            dir = directions.get(dir);
            dir.addReferences([lastRef, vertexReferences[i]]);
        } else {
            dir.addReferences([lastRef, vertexReferences[i]]);
            directions << dir;
        }
        lastPoint = currentPoint;
        lastRef =  vertexReferences[i];
        refs << lastRef;
    }
    if (loop <= 4 or allPointsInSamePlane(points)) {
        ADynamicMeshEnv env = triangulatePoints(points);
        for (z in env.triangles) {
            tMesh.triangles << refs[z];
        }
    } else {
        for (z in directions) if (z.references.count < 3) directions.remove(z);
        int noOfPlanes = (points.count - 2)/2;
        DirectionEnv toBeRemoved;
        double longestDist = 0;
        if (directions.count > noOfPlanes) {
            for (dir in directions) {
                point lastP = vertices[dir.references.first];
                double totalDist;
                for (z in dir.references, start=1) {
                    totalDist += lastP.distanceSqr(vertices[z]);
                    lastP = vertices[z];
                }
                totalDist += lastP.distanceSqr(vertices[dir.references.first]);
                if (totalDist > longestDist) {
                    toBeRemoved = dir;
                    longestDist = totalDist;
                }
            }
            if (toBeRemoved) directions.remove(toBeRemoved);
        }
        for (dir in directions) {
            point[] pts();
            for (z in dir.references) pts << vertices[z];
            if (pts.count > 2) {
                ADynamicMeshEnv env = triangulatePoints(pts);
                for (z in env.triangles) {
                    tMesh.triangles << dir.references[z];
                }
            }
        }
    }
    points.clear();
    cnt += loop;
```



Programmers
Most Popular Programming Method

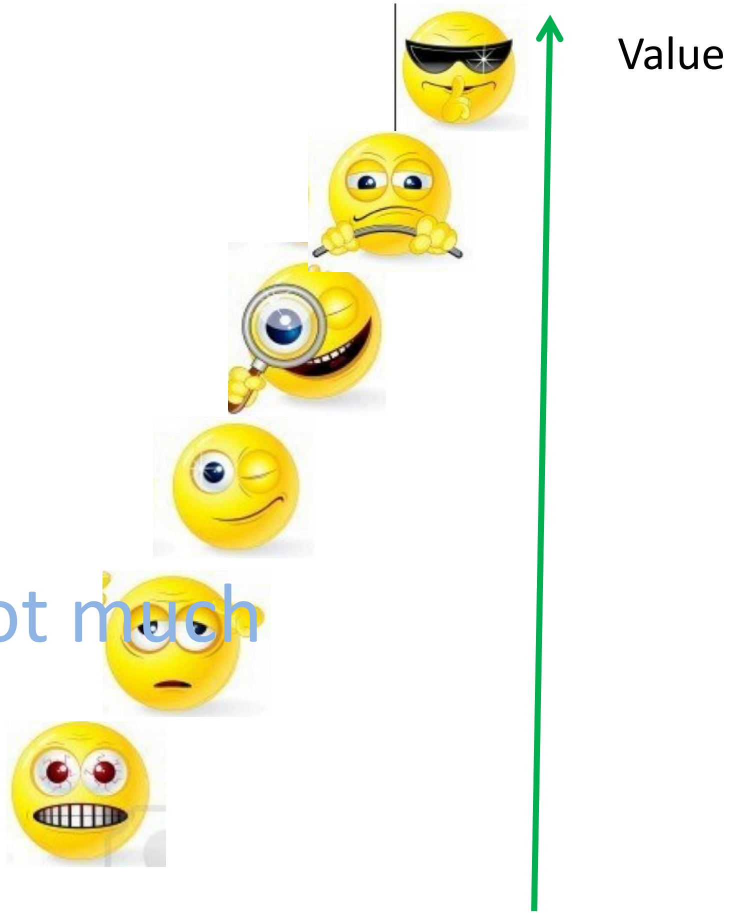
- ~~Simplification~~
- Intense Concentration
- Mixed with Trial and Error



CONFIGURA

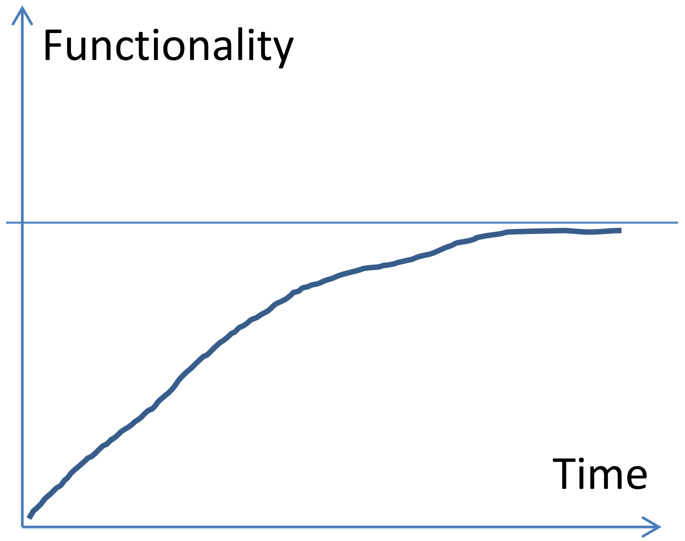
Programmers' Activities

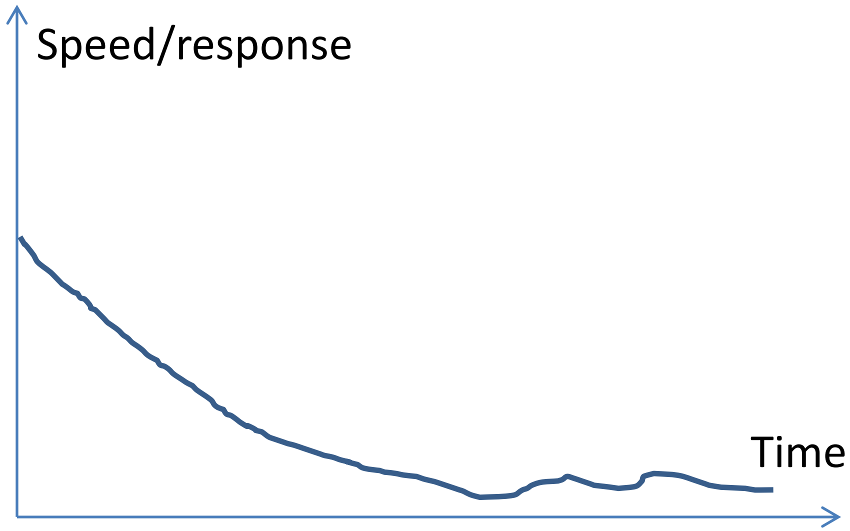
- DSL – what what what?
- modularizing – ZILCH
- abstracting - even less
- generalizing - slightly
- refactoring - some
- minimize/tune/optimize – not much
- cleaning up - seldom
- adding - all the time

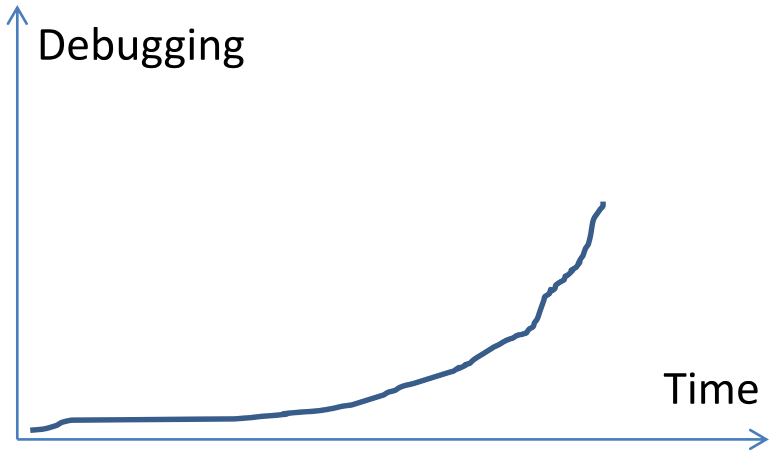


CONFIGURA

Reality







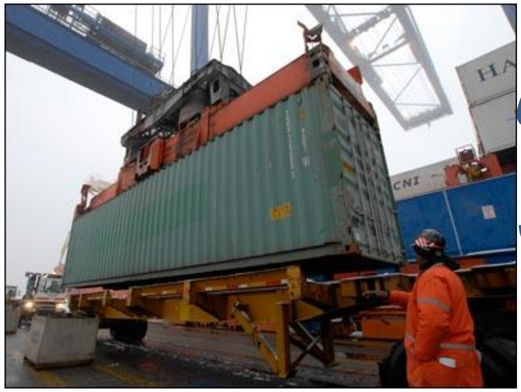
CONFIGURA

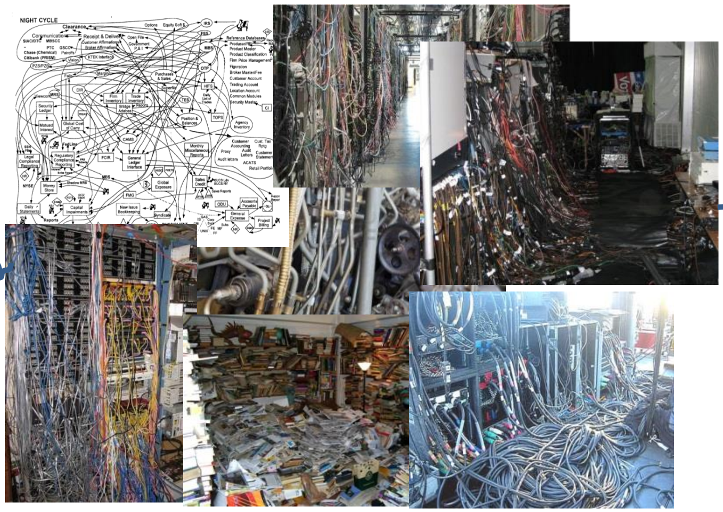
Programmers' Time

- I don't have time!

CONFIGURA

Programmers' Time





UI?

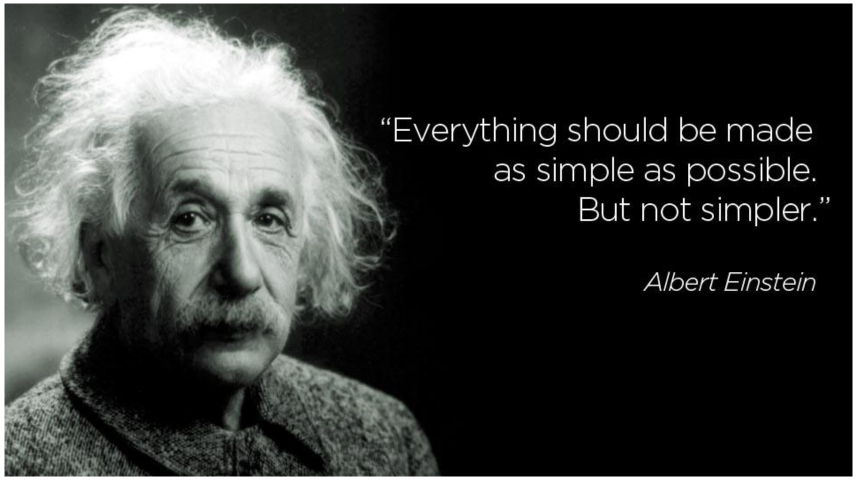
Programming

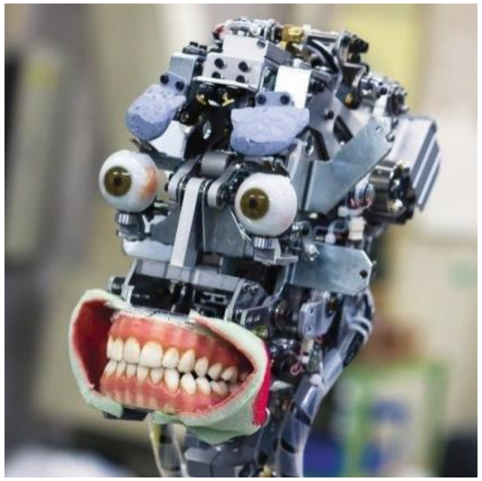
Getting the vocabulary exactly right

↓

Design a (conceptual) machine

Simple yet complete



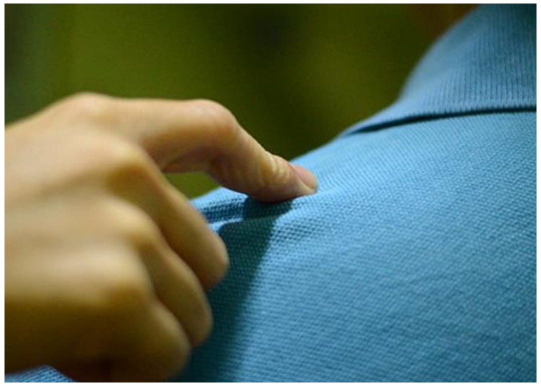


Programming

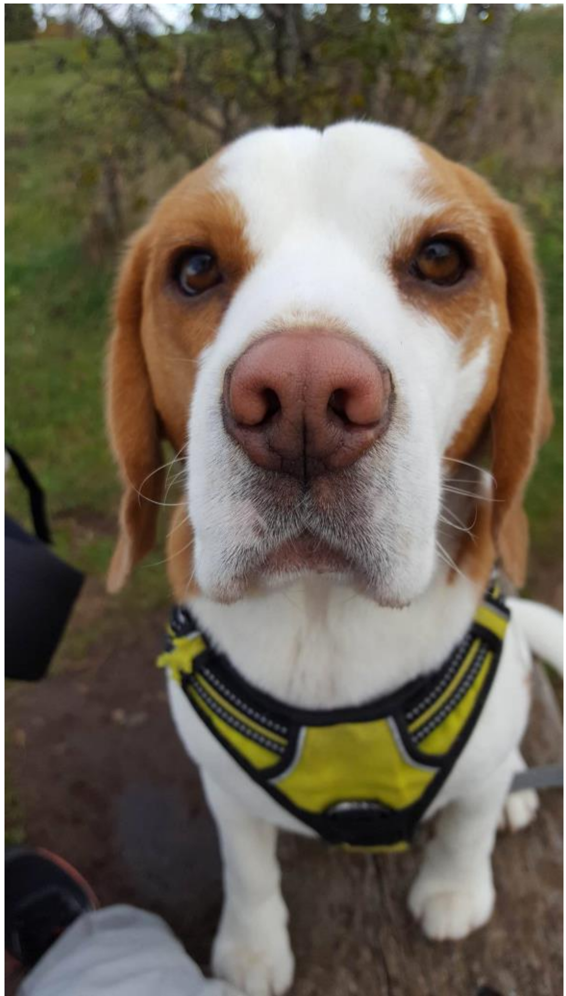
Thought of as a primarily logical process

$\downarrow$

Quality based emotional process

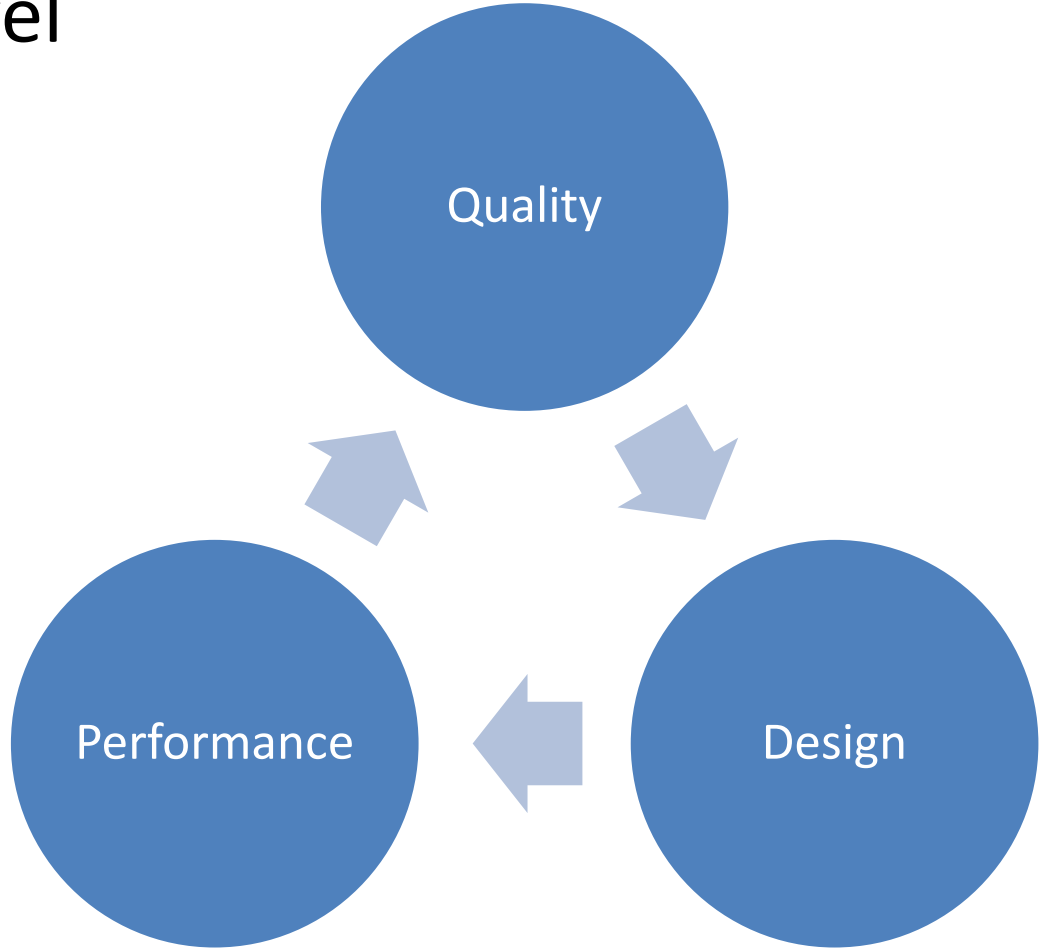


$\rightarrow$



Programming Quality

- Quality is high-level
- Rests on Design
- And Performance
- And Bounds Cost
  - compare to TCQ



Function Oriented Pitfall

Singular Function Focus

↓

Performance Degradation

↓

Quality Degeneration

↓

Unbounded Cost / Time

The 10 Countermeasures

1. Design Based
   - A primary vision (PGC) drives. Every activity, decision, enhancement, design is evaluated against the primary vision
2. Coaching based design. Leverage experience.
3. Regular coaching between mentor and team-member. The vision must remain uncompromized.
4. Strong encouragement for code improvement of all kinds.
5. Scheduled Performance focus.
6. Scheduled Quality focus.
7. Scheduled initiative time.
8. Measure activities. Where is time spent/wasted.
9. Managers must role play being users.
10. Continuous improvement. Encourage programmers good habits.

IF WE DON'T

FEATURES
FEATURES
FEATURES
FEATURES
FEATURES
FEATURES
FEATURES

CONFIGURA

IF WE DO

The Art of Programming

Freeform adaption from Sun Tzu – The Art of War

1. Programming is a matter of vital importance for mankind; the province of life or death; the road to survival or ruin. It is mandatory that it be thoroughly studied.

2. Therefore, appraise it in terms of 5 fundamental factors and 7 golden rules.

…

6. Know yourself, your process and your language and in a 100 releases you will prevail.

6a. If you know yourself, but not your process, every other release will be in peril.

6b. If you do not know yourself, nor your process, every release will be in peril.