1. Framework

Cargo Cult

Achieving the successful results of advanced cultures by mimicking their behavior.

Not that those examples describe bad behavior…

- The behavior as such is good.
- If you are successful, you will perform that behavior.

Understanding the behavior means being able to:

- Adapt it to your organization and people
- Adapt it to various and changing contexts
- Differentiate between the essence and the ritual

Just mimicking it, without understanding, will only cost you money and won't give any of the advantages.

Context Matters

What works in some context, might not work in your context!

- Across organizations but also within an organization

Difference can be subtle

- People and teams
- History
- Experience
- Environment
- Problem at hand

No Silver Bullet

Accidental complexity: arises purely from mismatches in the particular choice of tools and methods applied in the solution.

Essential complexity: according to *Frederik P. Brooks* there are 4 dimensions:

- *Complexity*. No two parts are equal and systems often have a very large set of possible states.
- *Conformity*. It needs to conform to the (often illogical) whims of its users.
- *Changeability*. Systems change constantly, and often surviving its original context.
- *Visibility*. It's not visible and can't be represented in our favorite way: as geometric abstraction.

(see the other side for more detail)

2. No Silver Bullet - Essential Complexity

Complexity

Software is probably the most complex entity of any human construct.

- No two parts are alike, no repeated elements.
- Very large number of states.
- Scaling up means more (different) elements, not just larger elements.

This complexity is essential and not accidental.

- Representations that abstract away its complexity often abstract away its essence.

Conformity

Physicists and chemist are lucky … what they 'model' follows logic, it's based on unifying principles.

- Even though we might not have mastered this logic yet.

Software must conform to the whims of its customers

- Essential complexity is arbitrary
- It differs from organization to organization and time to time

Changeability

Software is constantly subject to pressures for change

So are cars, buildings, computers …

- But they change infrequently after manufacturing
- They are superseded by later models

Software on the other hand, is changed after it is manufactured

- Since it's pure 'thought' … easier to change.
- It often survives it's context and needs to evolve

Invisibility

Software is invisible and unvisualizable.

Others use geometric abstractions

- Floor plan of a building
- Scale drawings of mechanical parts

A geometric reality is captured in a geometric abstraction

But software is not embedded in space … it has no ready geometric abstraction

We need several abstractions, superimposed on one another… not even hierarchical