

Martin Lippert

# **Wie Modulsysteme Architekturen beeinflussen**



[martin.lippert@it-agile.de](mailto:martin.lippert@it-agile.de)

**How do systems look  
like in our daily work?**

**We think our systems look like this...**





**Looks familiar?**

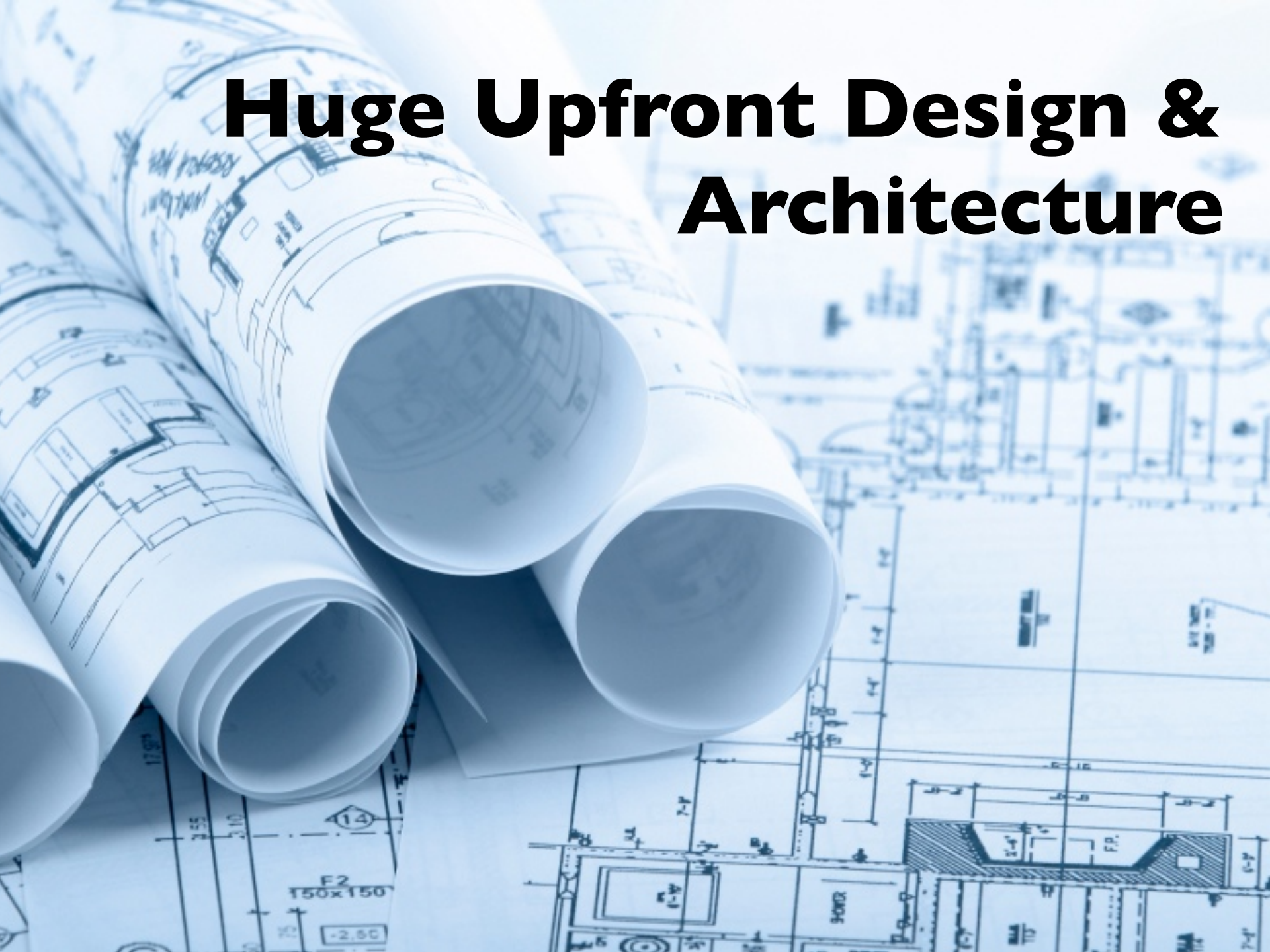


# **The Old Way of Software Architecture**

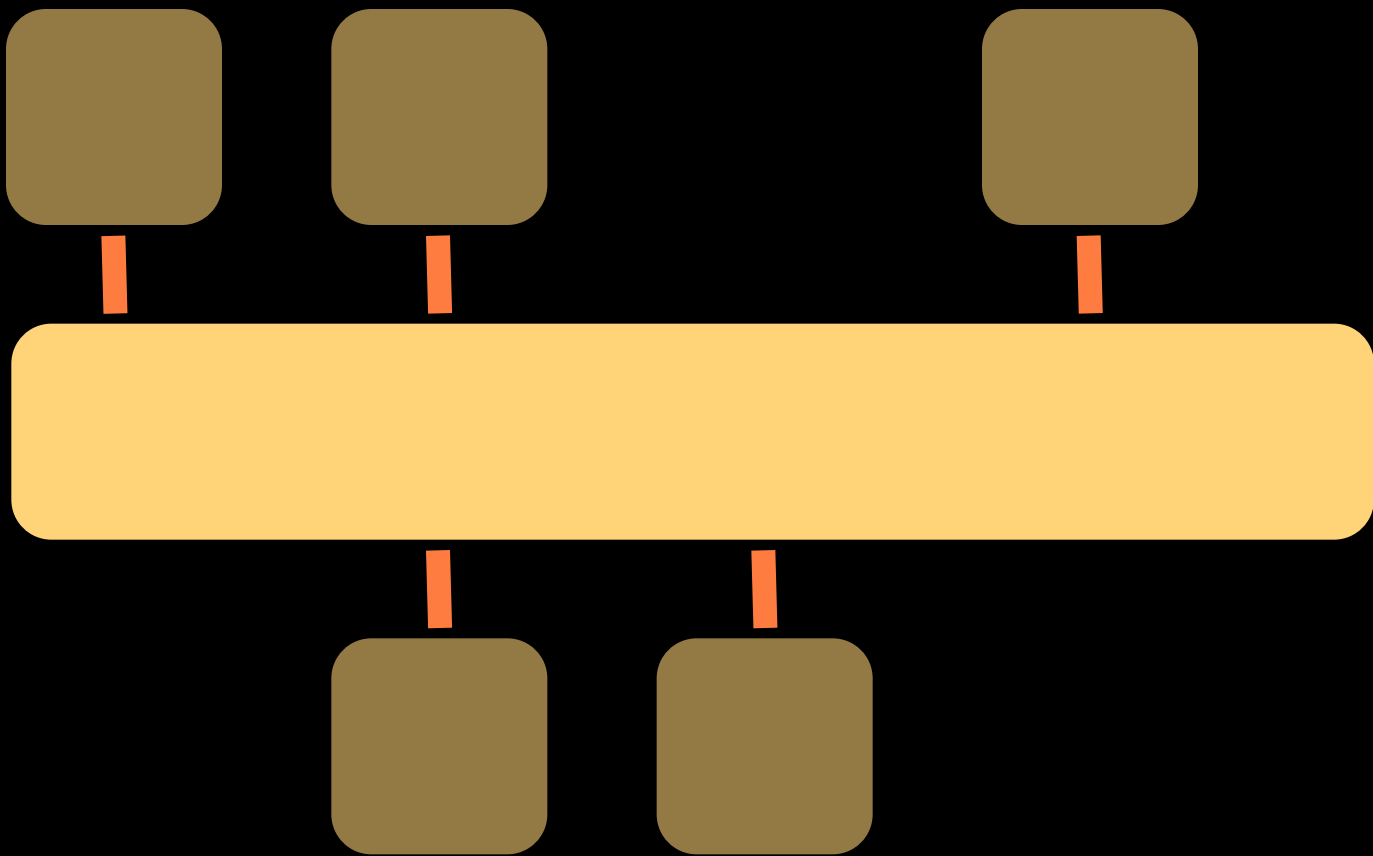




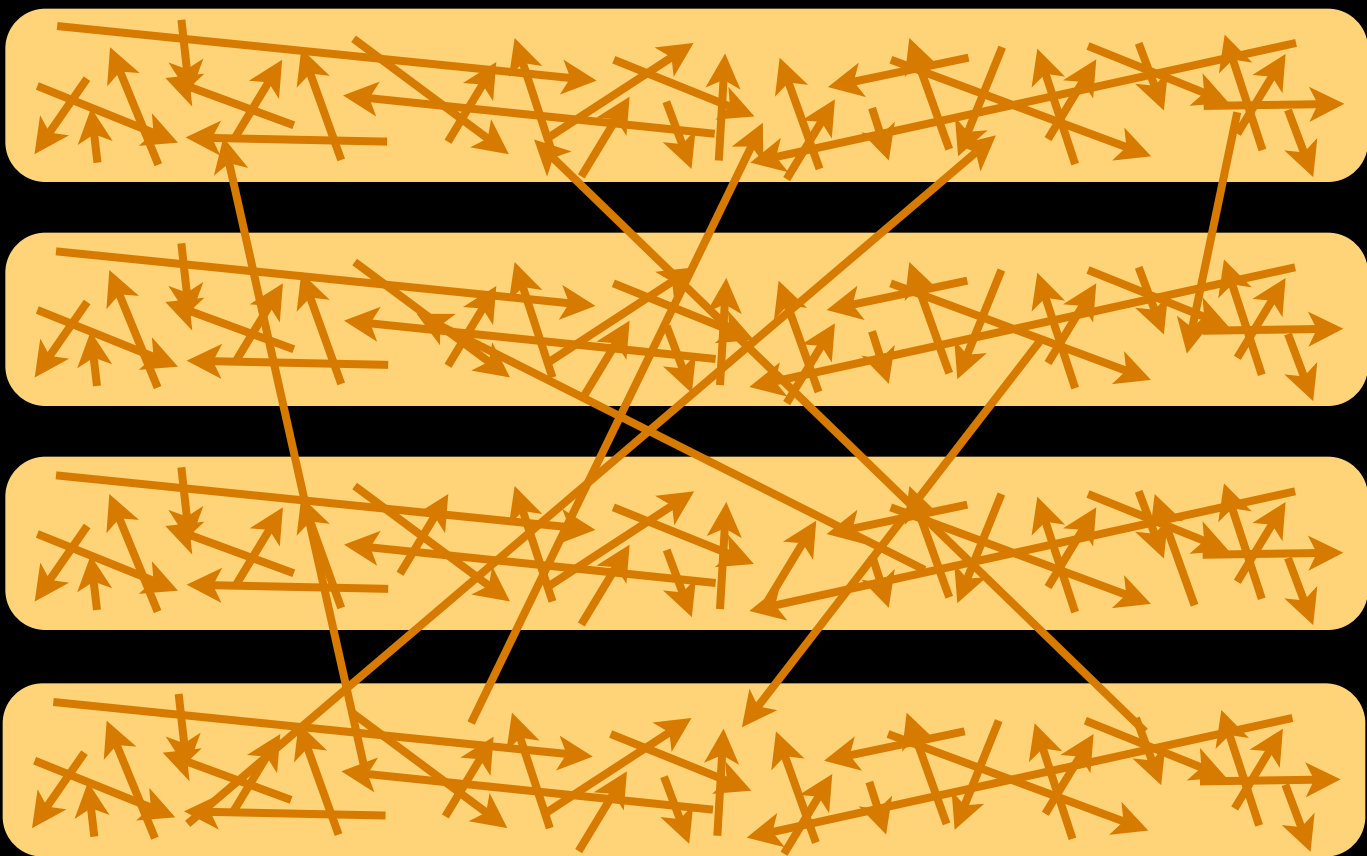
# **Huge Upfront Design & Architecture**











**Wake up!**  
**We need to change our direction...**



# **Agile Software Development**

Focus on Business Value

Changing Requirements

Incremental Development

Simple Solutions

Small Steps

# Inspect & Adapt

Short Release Cycles

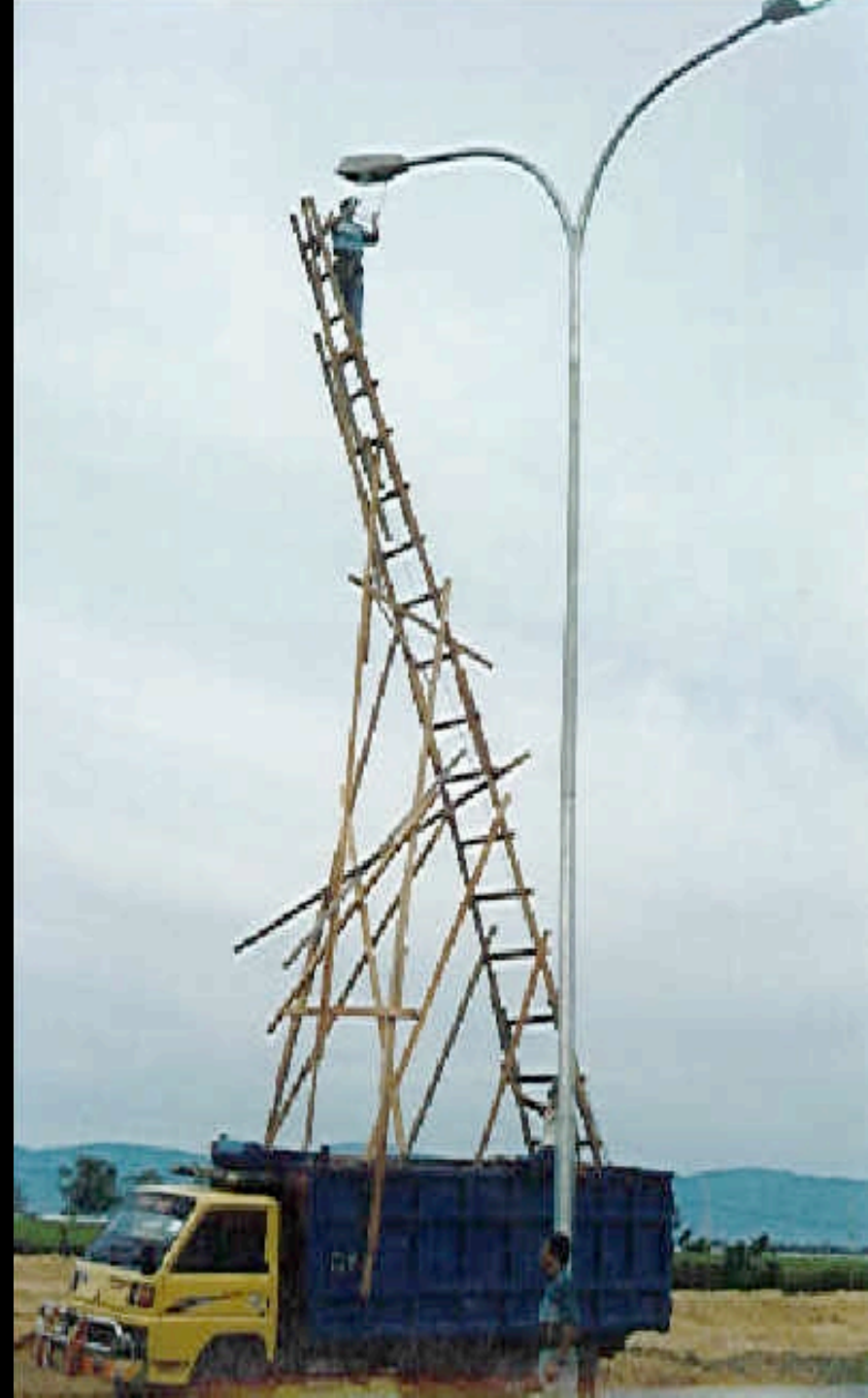
Shipping

No Big Upfront Design

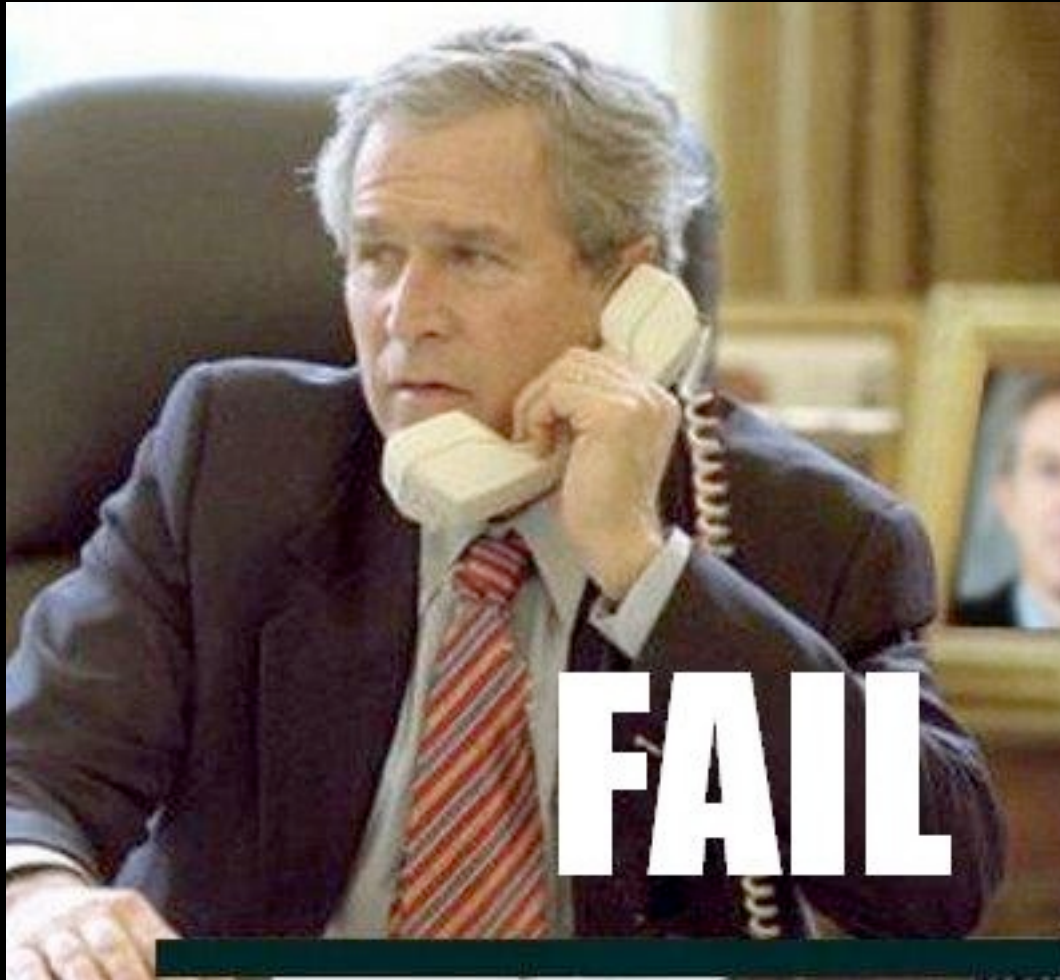
Changing Code all the Time



**We are agile  
because we don't  
care about  
architecture – it  
will emerge  
magically**



**But you are probably wrong...**



**Instead you live  
in great danger**



# Start simple and evolve

## the long version

**Gall's Law:** “A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system.”

– John Gall



**What should  
we do?**





# Flexibility & Modularity

# **We need flexibility**

**changing requirements**  
**learning process**  
**incremental development**

**But wait!**

**We already have  
all this...**



We have:

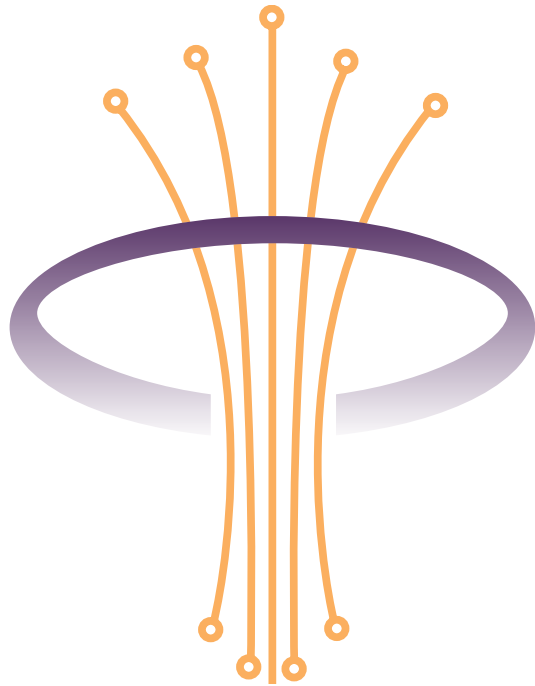
**Object-Orientation**  
**Patterns**  
**Information Hiding**  
**Encapsulation**  
**Layers**

...



**But reality can  
be hard...**

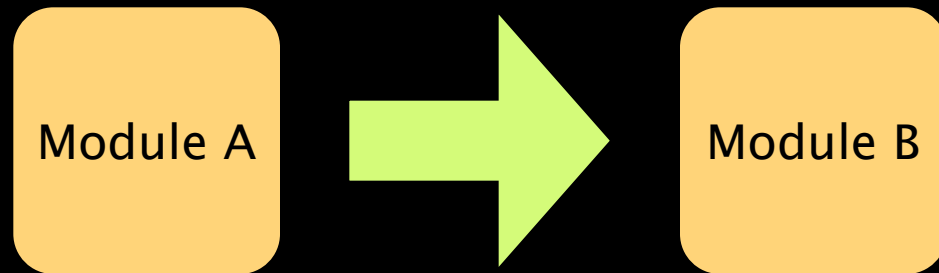
**We need a real  
module system**



**OSGi<sup>TM</sup>**  
**Alliance**



# I. Dependencies



# II.

## Visibilities

API Module A

Private Implementation  
Module A

# III.

# Dynamics



# Where do we go?



# **Loose Coupling & High Cohesion**

**Think about your dependencies  
every single day**



Sounds good...

**But how to realize?**

# Good old design principles

DIP

SOC

LSP

ADP

TDA

DRY

AIP

ISP

SCP

OCP

IHP

SRP

SDP

# new design principles

Use services

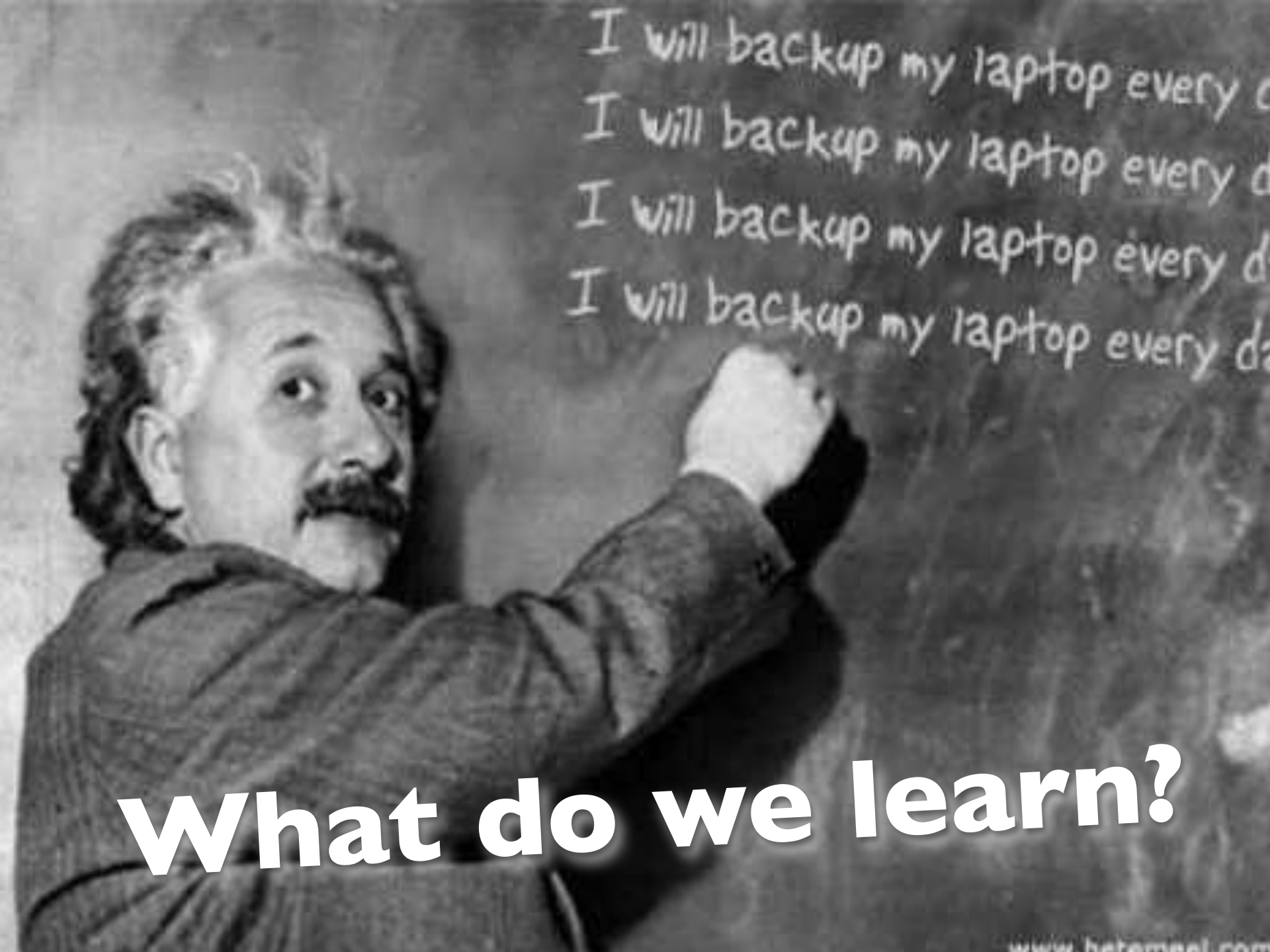


Separate between  
interface and implementation

Use extensions



working but extensible  
components

A black and white photograph of Albert Einstein, looking over his shoulder with a surprised expression while writing on a chalkboard. He is wearing a dark jacket and has his characteristic wild hair and mustache. The chalkboard is filled with the phrase "I will backup my laptop every day" written multiple times in a cursive script.

I will backup my laptop every day  
I will backup my laptop every day  
I will backup my laptop every day  
I will backup my laptop every day

**What do we learn?**

# Guide I: **Many small modules**

instead of few big ones



Guideline 2:  
**Fewer connections  
between modules**

instead of everything is wired to everything

# Guideline 3: **Less visibilities**

instead of making everything public

Guideline 4:  
**Many small frameworks**

instead of few big ones

Guideline 5:  
**Think about extensibility**

instead of knowing everything

Guideline 6:  
**Design your architecture  
every day**

instead of ignoring what you have learned



# Thank you for your attention

Martin Lippert  
[martin.lippert@it-agile.de](mailto:martin.lippert@it-agile.de)



Visit us at our booth