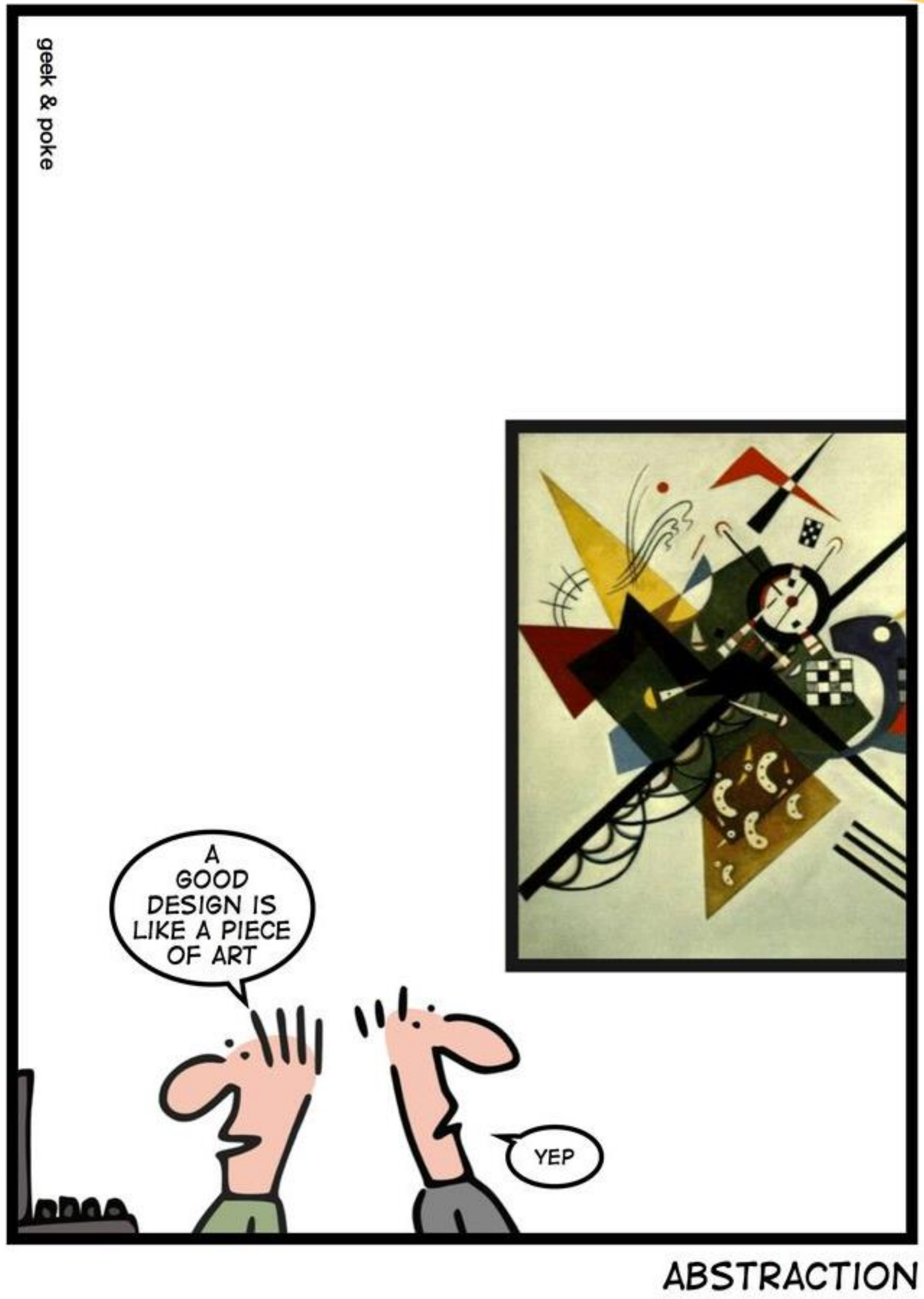
ABSTRACT ART

GETTING ABSTRACTION “JUST RIGHT”

Presented by Jeremy Clark
www.jeremybytes.com

A Good Design is like A Piece of Art

Geek & Poke – http://goo.gl/ifd53l



ABSTRACTION IS AWESOME!

Maintain ✓

Extend ✓

Test ✓

ABSTRACTION IS AWFUL!

Complexity

Confusion

Debugging Difficulty

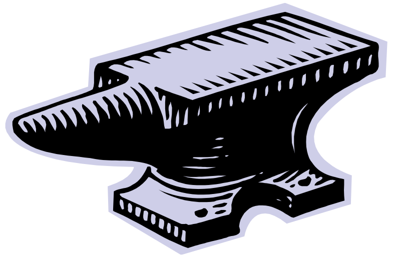
Frustration

@jeremybytes

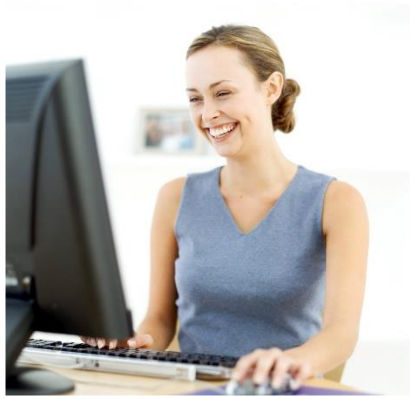
https://archive.org/details/goldilocks_and_the_three_bears

GOLDILOCKS THE DEVELOPER

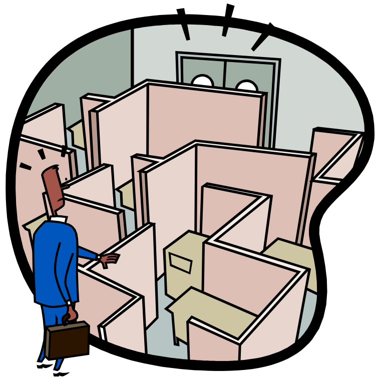
Too Little
Abstraction



Just Right



Too Much
Abstraction



TWO TYPES OF DEVELOPERS

Over-Abstractor

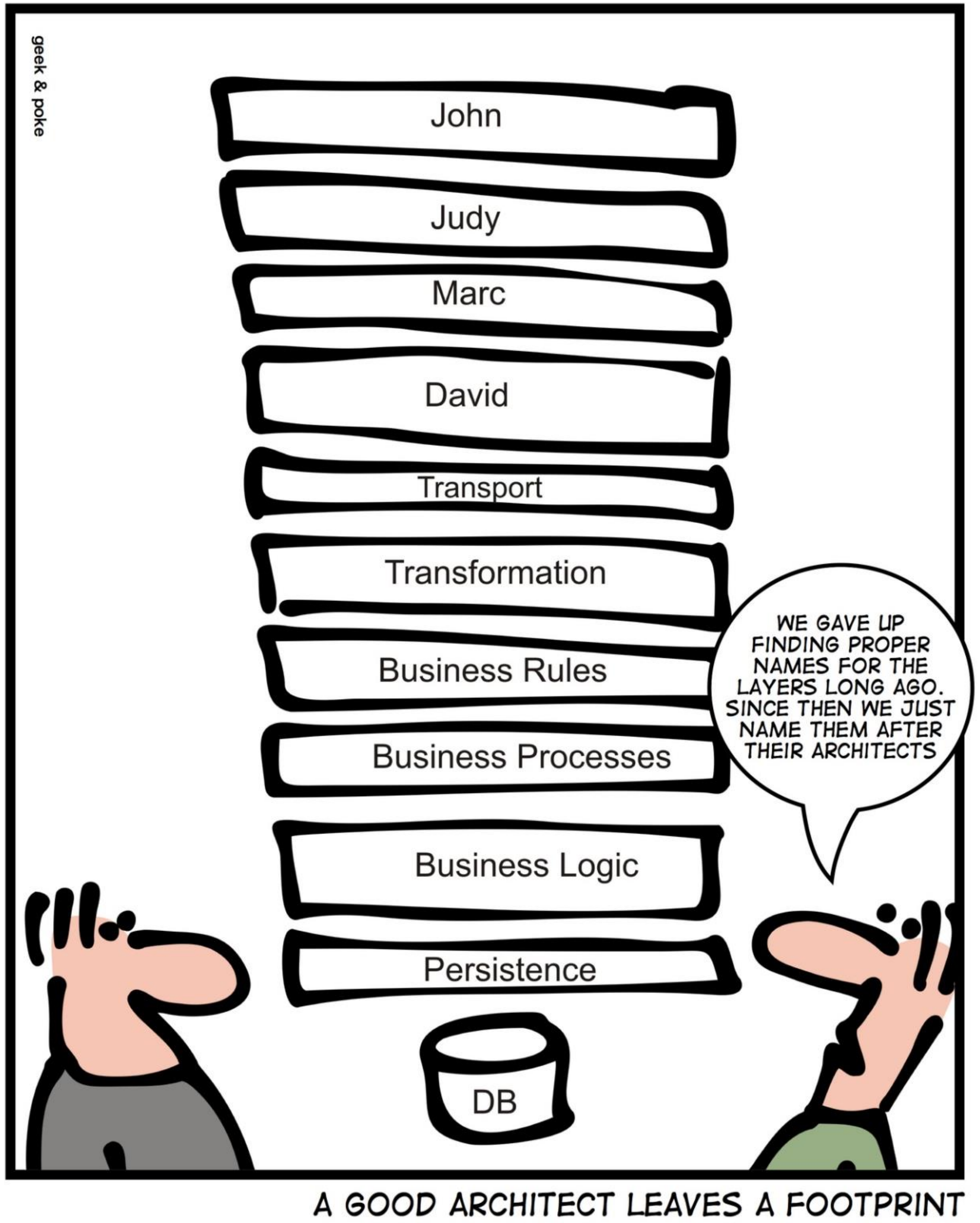
Under-Abstractor

Over-Abstractor

- “We’ll have a good use for this in the future.”
- Overly Complex
- Difficult to Maintain

A Good Architect Leaves A Footprint

Geek & Poke: http://goo.gl/B4uXa3


geek & poke
John
Judy
Marc
David
Transport
Transformation
Business Rules
Business Processes
Business Logic
Persistence
DB
WE GAVE UP FINDING PROPER NAMES FOR THE LAYERS LONG AGO. SINCE THEN WE JUST NAME THEM AFTER THEIR ARCHITECTS
A GOOD ARCHITECT LEAVES A FOOTPRINT

Under-Abstractor

- “Let’s keep things simple.”
- Rigid
- Difficult to Maintain

COMMON PROBLEM

Over-Abstractor

- “We’ll have a good use for this in the future.”
- Overly Complex
- Difficult to Maintain

Under-Abstractor

- “Let’s keep things simple.”
- Rigid
- Difficult to Maintain

The Default State Quiz

Who Are You?

Let’s build a plug-in architecture…

Awesome!
Let’s do it.

Maybe we should look at compile-time options.

We need to share a value between modules…

I'll create an object state manager.

Let's use a global variable.

How should we do the UI?

Here's a new JavaScript framework.

Let's use the same framework we did last time.

Pull data from a database…

ORMs are awesome!

SELECT *
FROM Customers
WHERE ID = [@id]

We need an object instance…

```
var logger =
DIContainer
.Resolve<ILogger>()
```

```
var logger =
new FileLogger()
```

Neither answer is right or wrong. The correct response is “It depends.”

—Jeremy’s Standard Response

Let’s build a plug-in architecture…

Awesome!
Let’s do it.

Maybe we should look at compile-time options.

We need to share a value between modules…

I’ll create an object state manager.

Let’s use a global variable.

How should we do the UI?

Here’s a new JavaScript framework.

Let’s use the same framework we did last time.

Pull data from a database…

ORMs are awesome!

```
SELECT *
FROM Customers
WHERE ID = [@id]
```

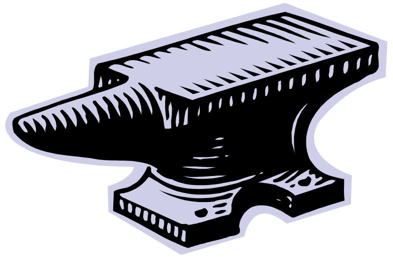
We need an object instance…

```
var logger =
DIContainer
.Resolve<ILogger>()
```

```
var logger =
new FileLogger()
```

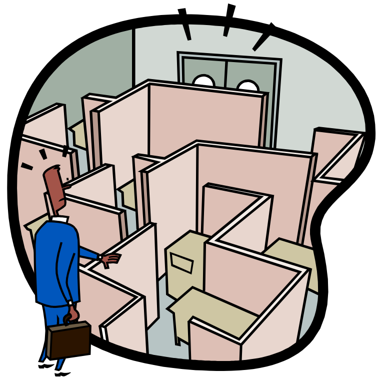
BE HONEST WITH YOURSELF

Too Little
Abstraction

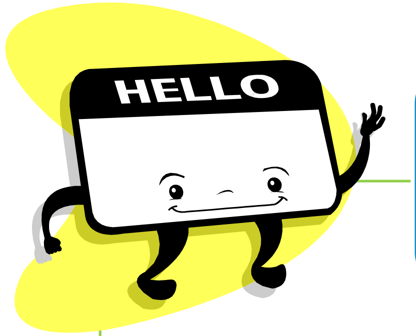


Just Right

Too Much
Abstraction



WHO AM I?



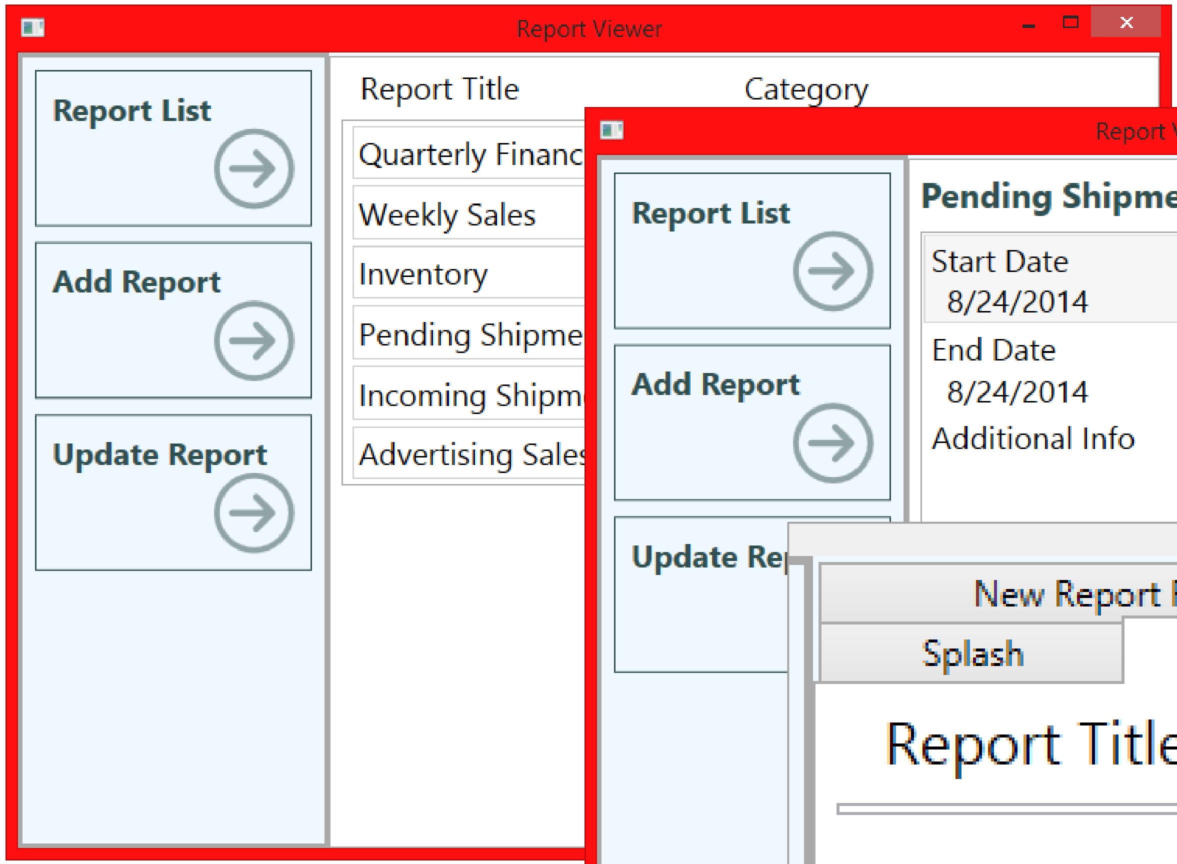
Under-Abstractor

- Hello. My name is Jeremy, and I'm an Under-Abstractor.

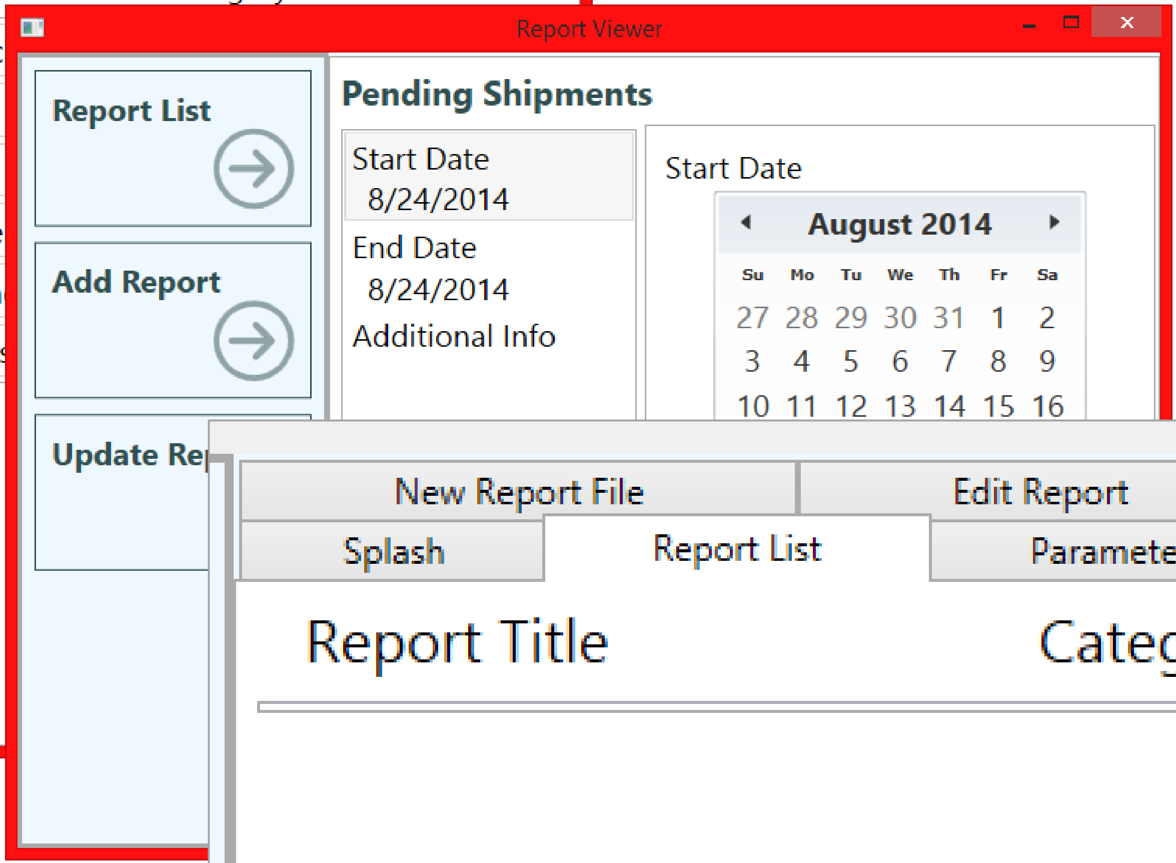
"Keep Things Obvious"

"Don't Be Tricky"

REPORTING APPLICATION



Report Title	Category
Quarterly Financ	
Weekly Sales	
Inventory	
Pending Shipme	
Incoming Shipm	
Advertising Sales	



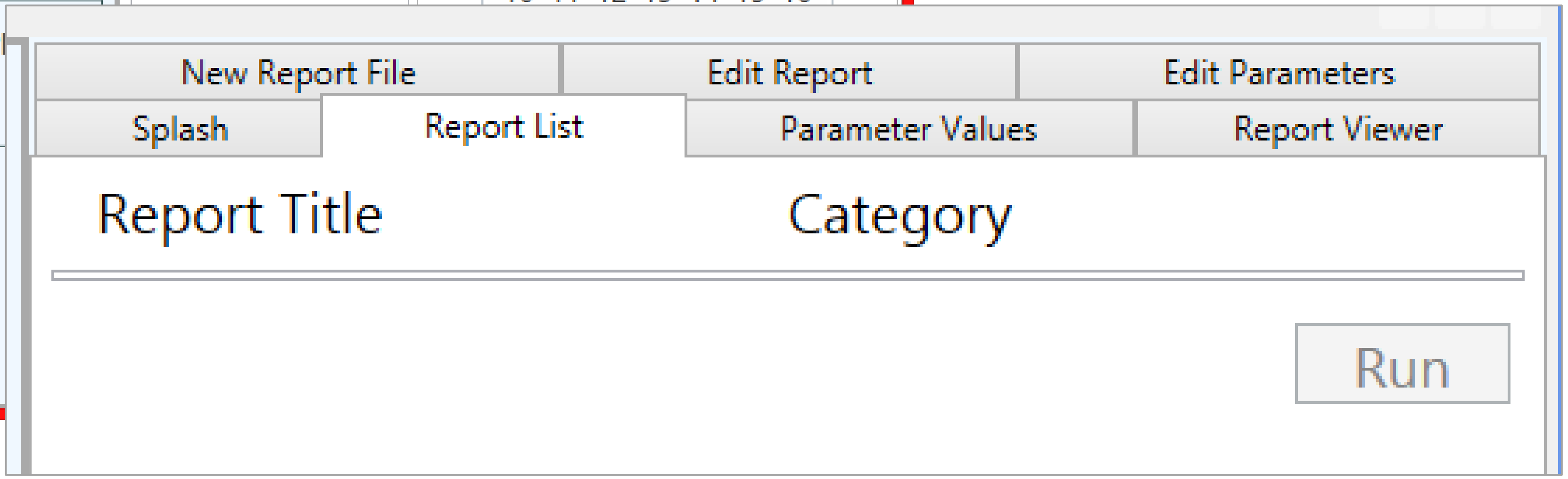
Pending Shipments

Start Date
8/24/2014
End Date
8/24/2014
Additional Info

Start Date

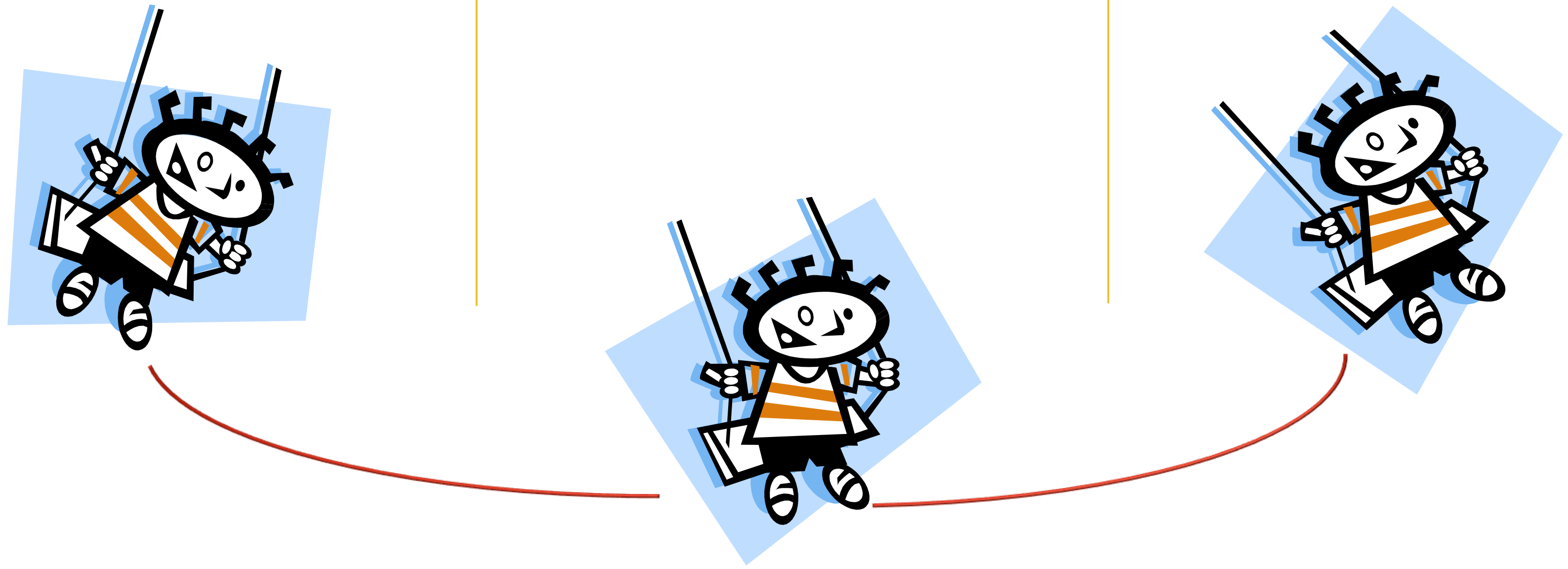
August 2014

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16



THE PENDULUM EFFECT

Under-Abstraction	Just Right	Over-Abstraction



THOSE AROUND YOU

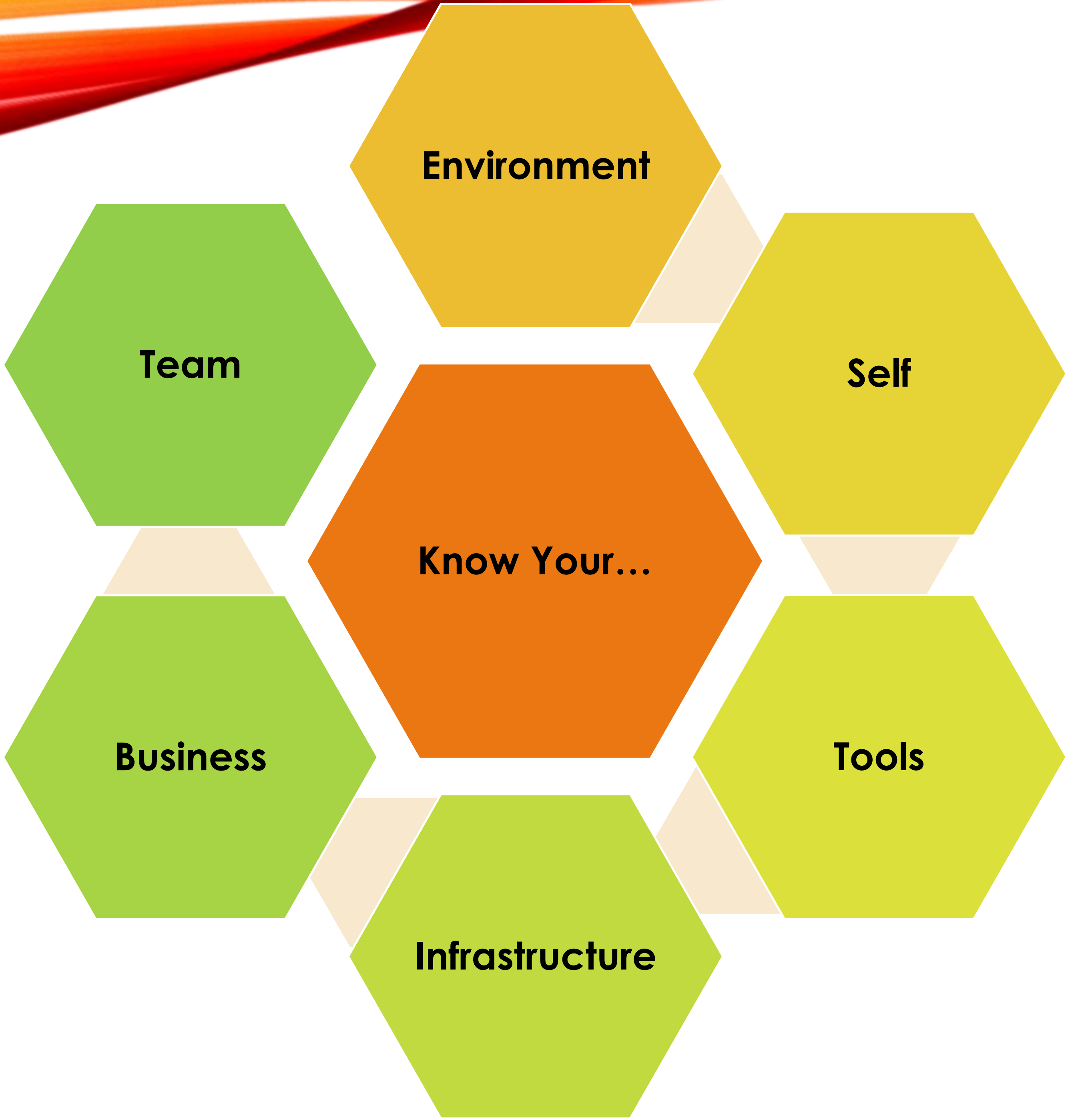
Over-Abstractor

- Jeff loved to build components.
- He liked to create code for re-use.
- He thought of all possible scenarios.

A SYMBIOTIC RELATIONSHIP

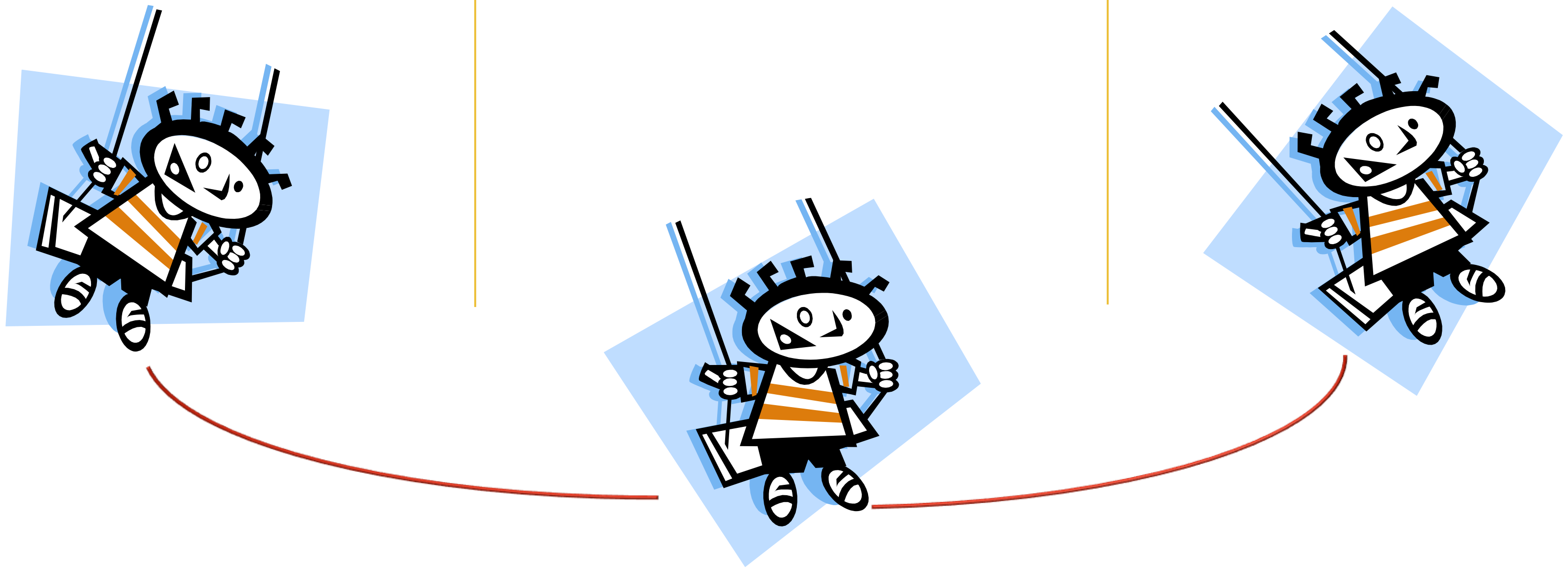
The Over-Abstractor helps the Under-Abstractor get things Just Right

The Under-Abstractor helps the Over-Abstractor get things Just Right


Environment
Team
Self
Know Your…
Business
Tools
Infrastructure

THE PENDULUM EFFECT

Under-Abstraction | **Just Right** | **Over-Abstraction**



VARIOUS DATA SOURCES

Microsoft SQL Server

MongoDB

CSV

SOAP Service

WebAPI

Oracle

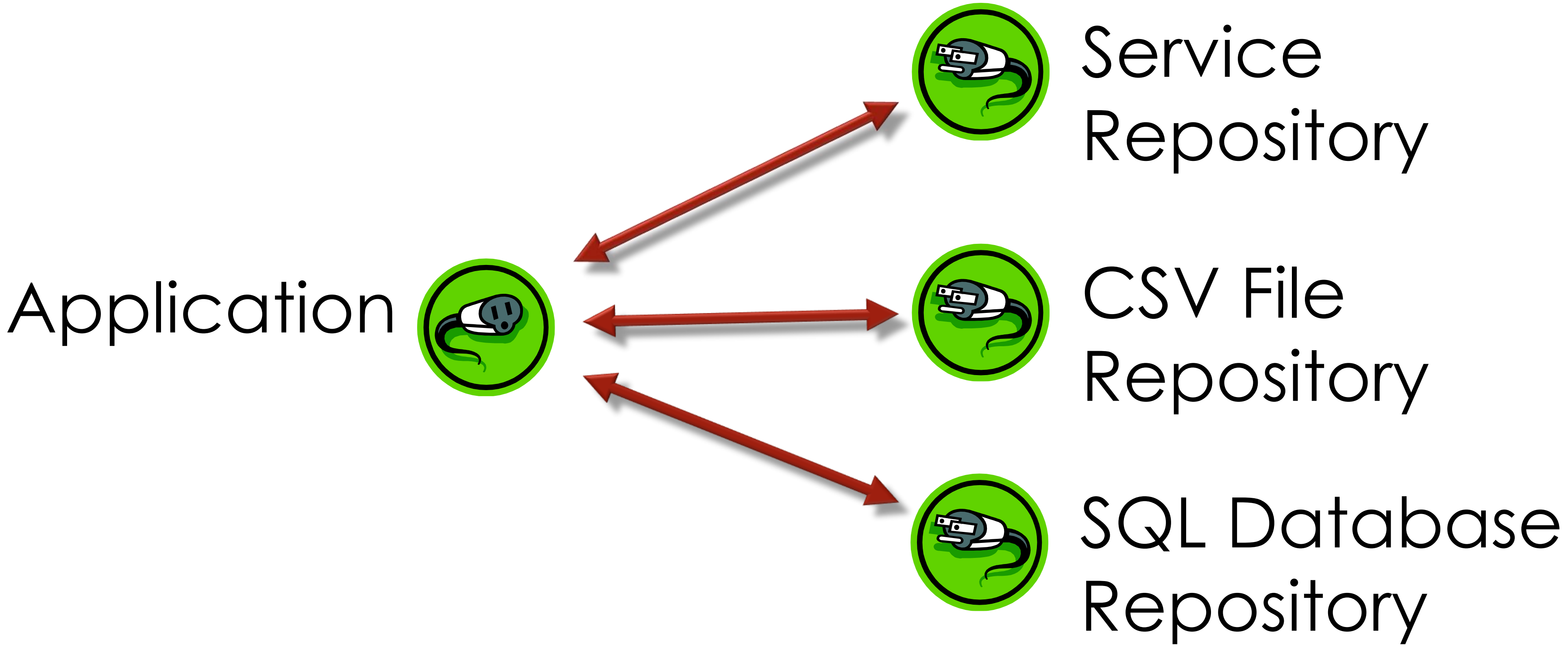
Amazon AWS

JSON

Hadoop

Microsoft Azure

PLUGGABLE REPOSITORIES



HIKERS and BIKERS
Move to the side of the road when a vehicle approaches

DRY

- Don’t Repeat Yourself

Under-Abstractor

DON’T REPEAT YOURSELF

Consolidate Similar Code

Avoid Copy/Paste

Copy/Pasta

Spaghetti Code

SoC

- Separation of Concerns

Under-Abstractor

SINGLE RESPONSIBILITY PRINCIPLE

Complements Separation of Concerns

The “S” in S.O.L.I.D.

A class should have only one reason to change

A class should do one thing (and do it well)

YAGNI

- You Ain't Gonna Need It
- (You Aren't Going to Need It)

Over-Abstractor

MORAL OF YAGNI

- Code for the features you have now
- Add abstraction as you need it
- Don’t add abstraction based on speculation

We still think about the future,
but we don’t implement it yet.

KISS

- Keep It Simple, Stupid
- (Keep It Short & Simple)
- (Keep It Simple & Straightfoward)

Over-Abstractor

DDIY

- Don’t Do It Yourself

Over-Abstractor

Under-Abstractor

DDIY

Over-Abstractor

- Over-Abstractors like to build things to solve specific problems

Under-Abstractor

- Under-Abstractors shy away from external frameworks and libraries

EXAMPLES

Dependency Injection

- Unity, MEF, Ninject, Autofac, StructureMap, Spring.NET

Unit Testing Framework

- MSTest, NUnit, TypeMock Isolator, xUnit.net, Approval Tests

Mocking

- Moq, NSubstitute, RhinoMocks, FakeItEasy, JustMock

Logging

- log4net, Semantic Logging Application Block (SLAB)

UI Framework

- Prism, Angular, React

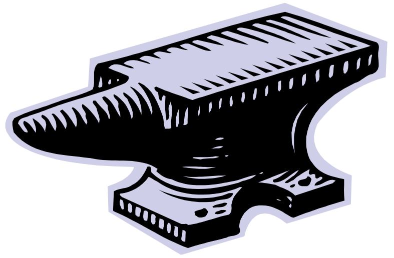
ABSTRACTION IS AWESOME & AWFUL

- ✓ Maintain
- ✓ Extend
- ✓ Test

- ✗ Complexity
- ✗ Confusion
- ✗ Debugging Difficulty

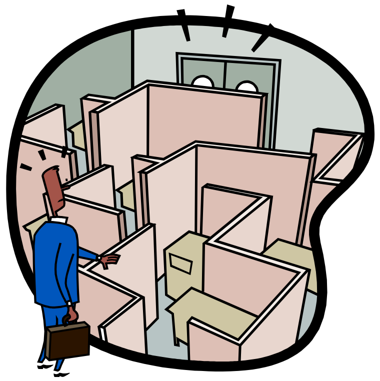
THE GOLDILOCKS PRINCIPLE

Too Little
Abstraction



Just Right

Too Much
Abstraction



GETTING THINGS RIGHT

DRY
- Don't Repeat Yourself

SoC
- Separation of Concerns

YAGNI
- You Ain't Gonna Need It

KISS
- Keep It Short & Simple

DDIY
- Don't Do It Yourself

THANK YOU!

Jeremy Clark

- http://www.jeremybytes.com
- jeremy@jeremybytes.com
- @jeremybytes