
# Abstract Art

## Getting Abstraction "Just Right"

**Jeremy Clark**
**Developer Betterer**
**GitHub: jeremybytes**

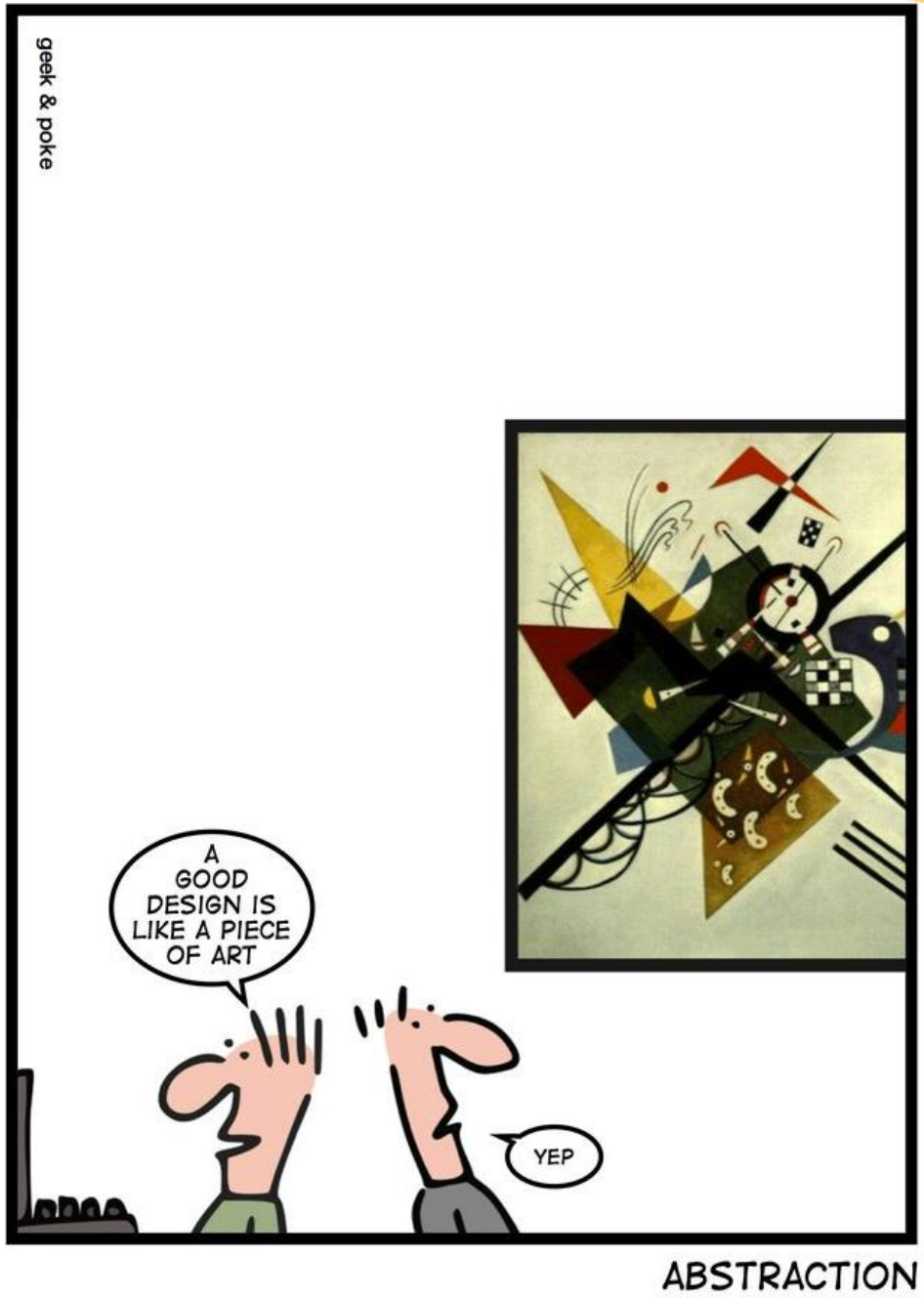
Level: Intermediate

# Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

# A Good Design is like A Piece of Art




**Geek & Poke – http://goo.gl/ifd53l**

# WHAT IS ABSTRACTION?

## Abstraction

- Simplifying reality
- Ignoring extraneous details
- Focusing on what is important for a purpose

# ABSTRACTION IS AWESOME!

**Maintain**

**Extend**

**Test**

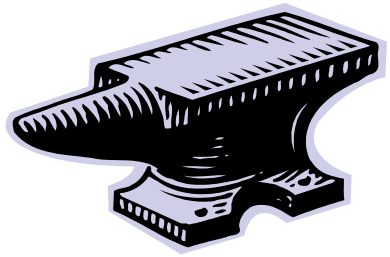
# ABSTRACTION IS AWFUL!


**Frustration**

https://archive.org/details/goldilocks_and_the_three_bears

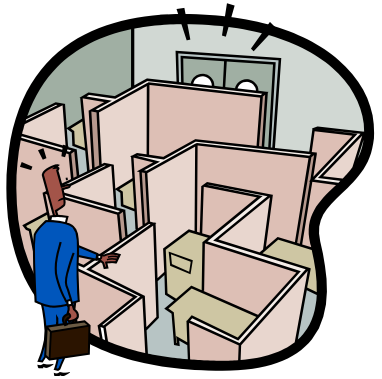
# GOLDILOCKS THE DEVELOPER

**Too Little**
Abstraction



**Just Right**

**Too Much**
Abstraction



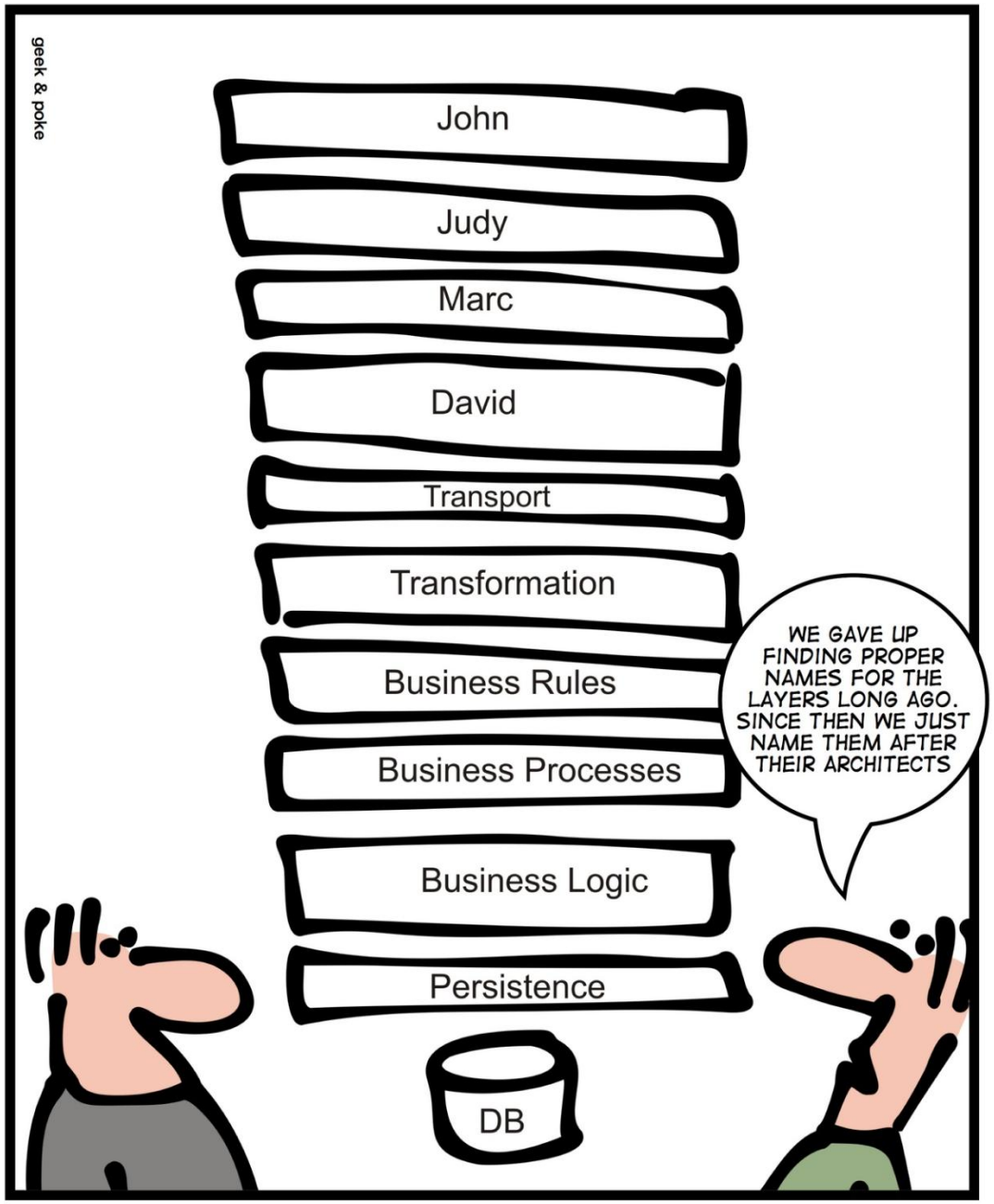
# TWO TYPES OF DEVELOPERS

- 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**

## 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…

+1

Awesome!
Let's do it.

-1

Maybe we should look at compile-time options.

# We need to share a value between modules…

+1

I'll create an object state manager.

-1

Let's use a global variable.

# How should we do the UI?

+1

Here's a new JavaScript framework.

-1

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

# Pull data from a database…

+1

ORMs are awesome!

-1

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

# We need an object instance…

+1

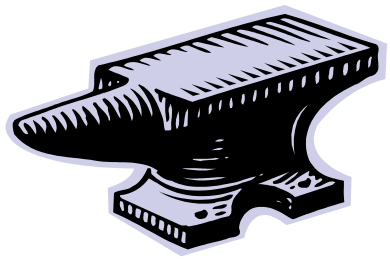
```
var logger =
DIContainer
.Resolve<ILogger>()
```

-1

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

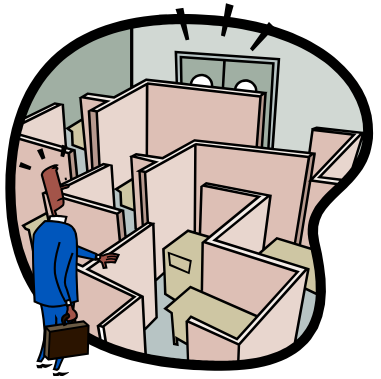
# BE HONEST WITH YOURSELF

**Too Little**
Abstraction

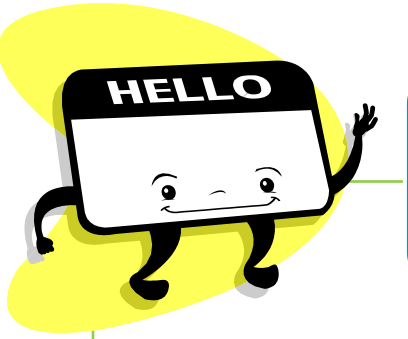


**Just Right**

**Too Much**
Abstraction



# WHO AM I?



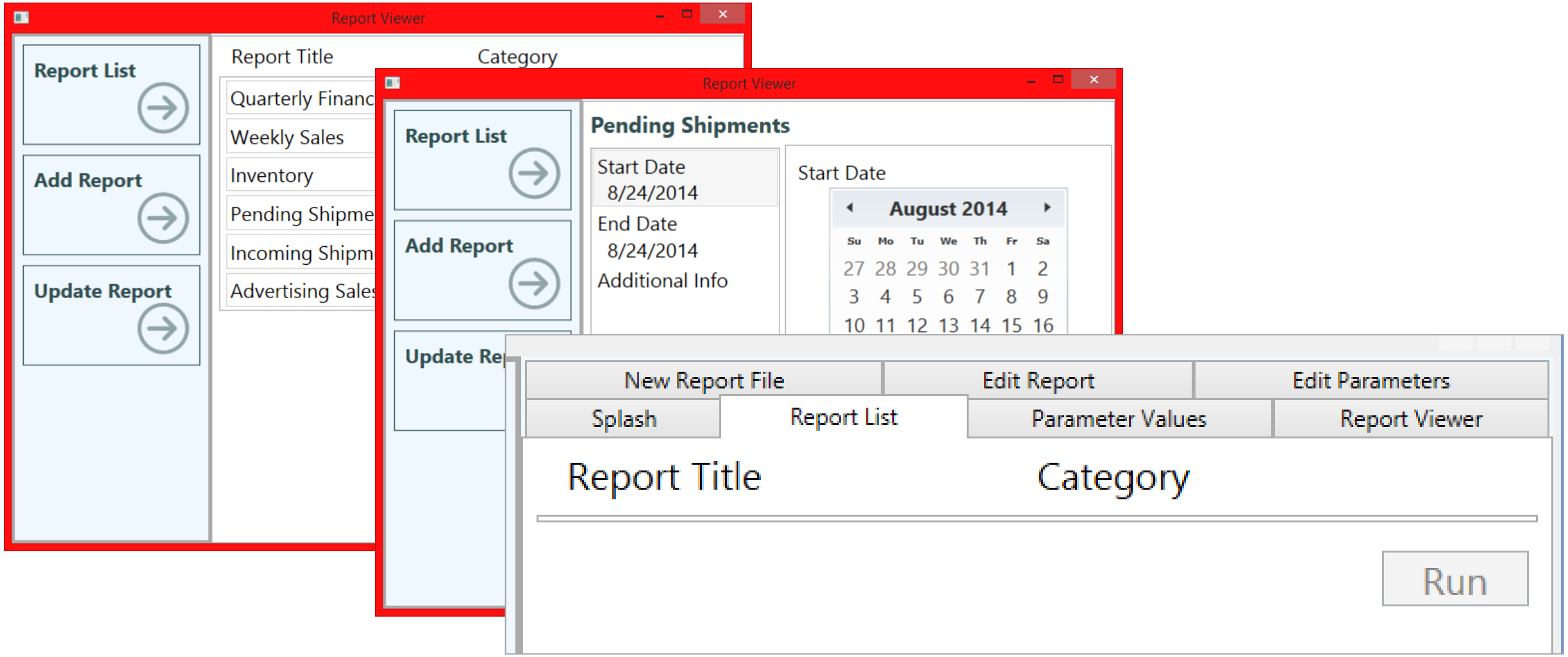

## Under-Abstractor

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

"Keep Things Obvious"

"Don't Be Tricky"

# REPORTING APPLICATION

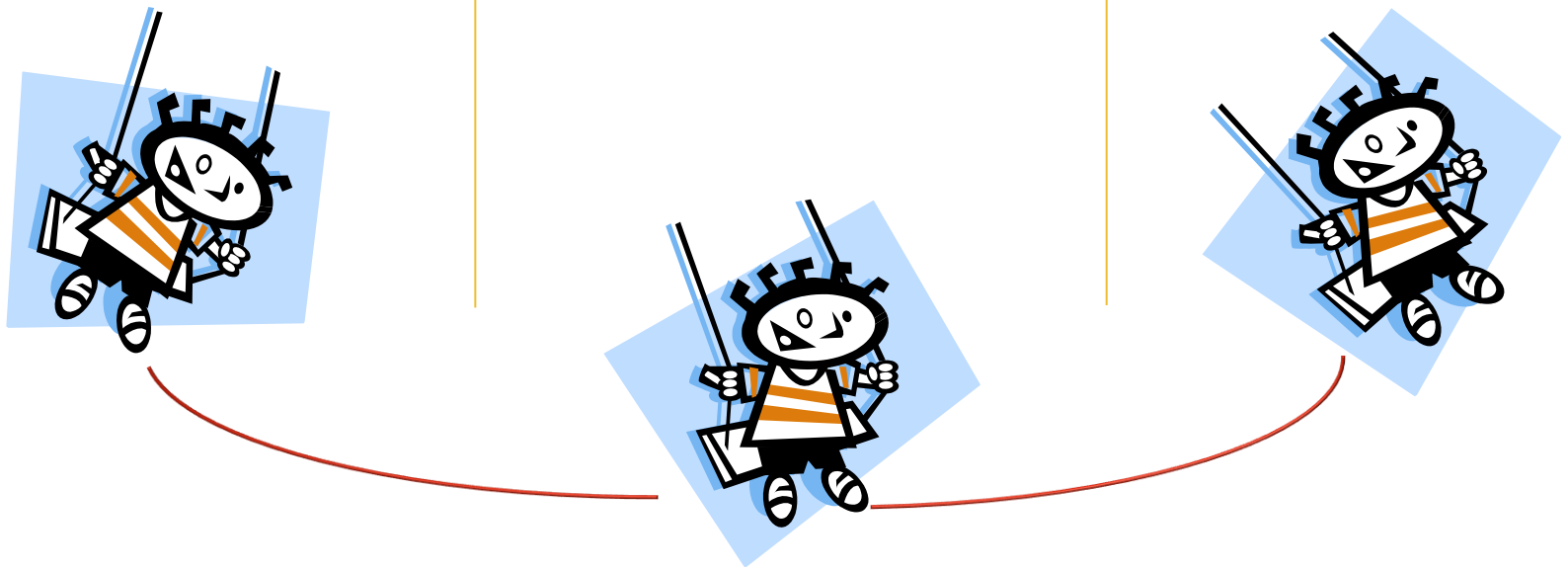


# THE PENDULUM EFFECT

**Under-Abstraction**

**Just Right**

**Over-Abstraction**



# REPORTING COMPONENT

https://reportingServer/RunReport?id=3D2B4067-DBEE-4B3C-A5D7-DC9011300775
&user-id=24&param1=20240531&param2=20240606&output=pdf

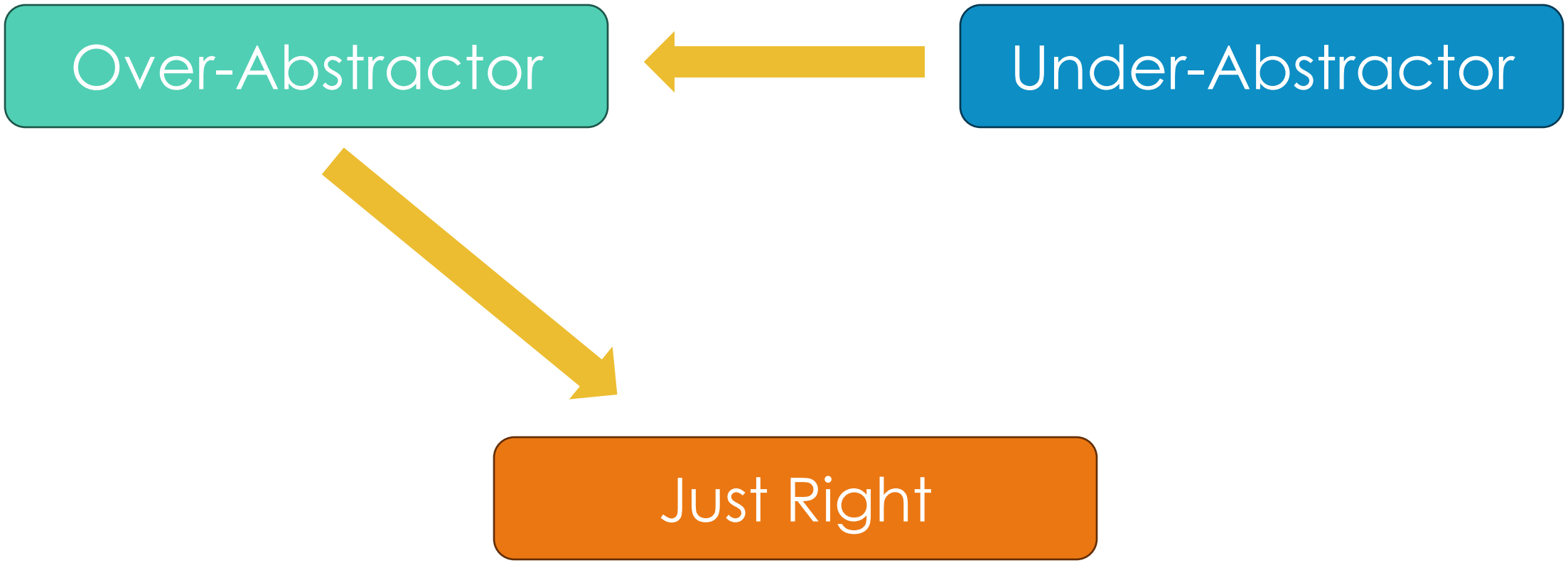
```
var report = new WebReport();
report.Application = "Facilities";
report.ReportName = "WeeklyMaintenance";
report.UserName = userName;
report.Params.Add("20240531");
report.Params.Add("20240606");
report.Output = ReportOutput.PDF;

report.RunReport();
```

# THOSE AROUND YOU

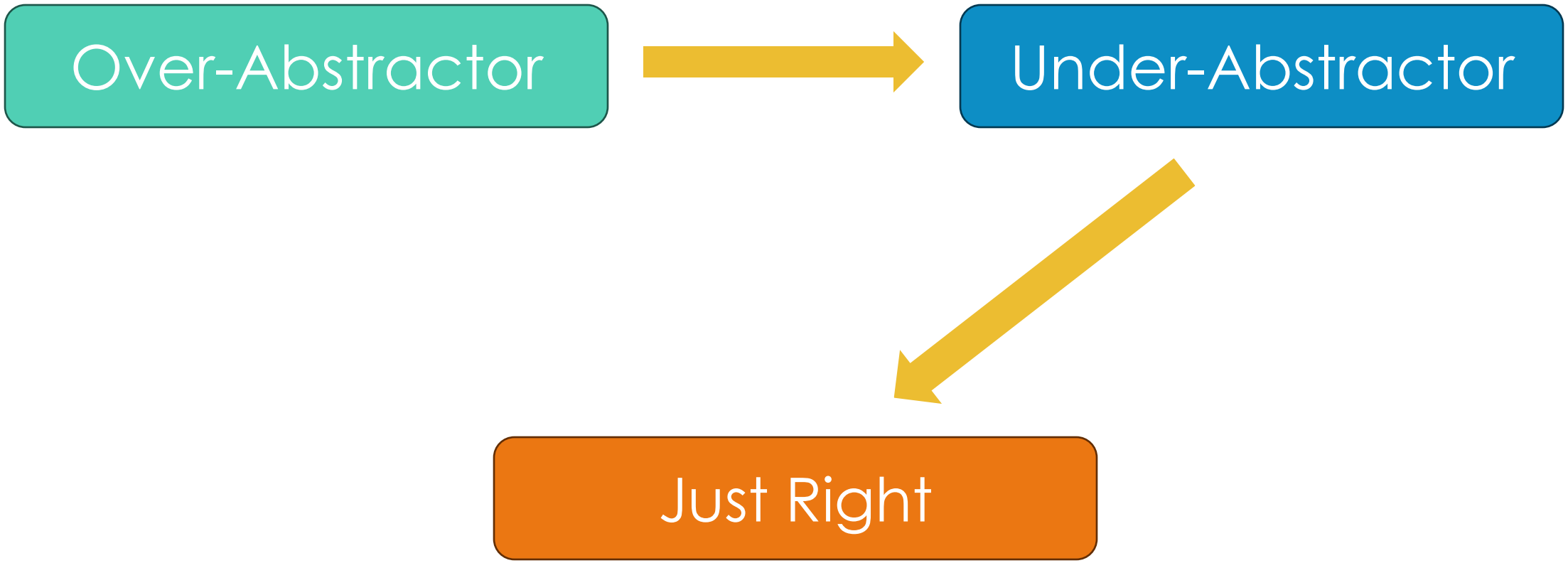
## Over-Abstractor

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

# A SYMBIOTIC RELATIONSHIP



# A SYMBIOTIC RELATIONSHIP




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

WebAPI

Oracle

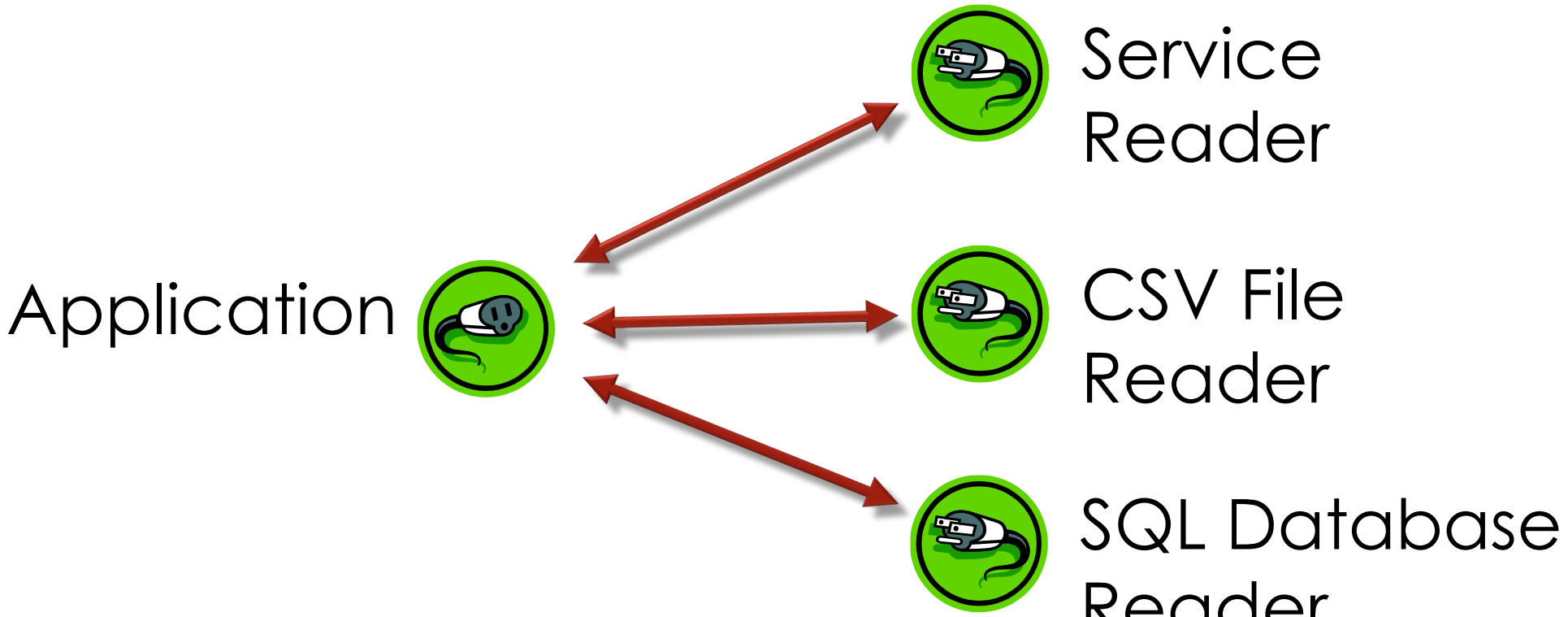
Amazon RDS

JSON

Azure Cosmos DB

Hadoop

# PLUGGABLE DATA READERS



# 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.

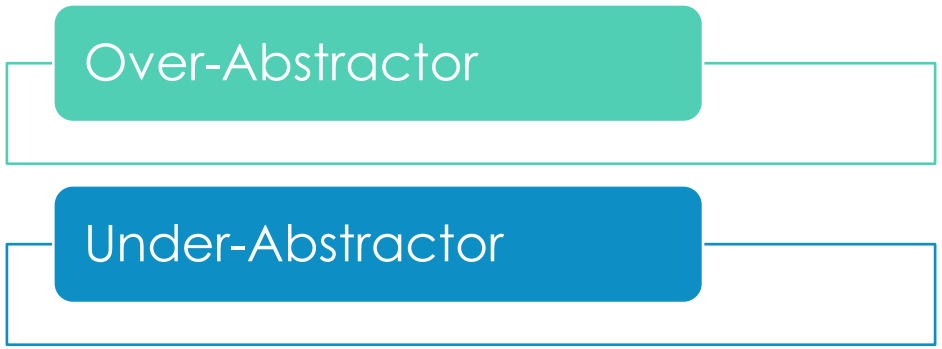
# KISS

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

Over-Abstractor

# DDIY

- Don’t Do It Yourself



# 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

- Ninject, Autofac, Spring.NET, Microsoft.Extensions.DependencyInjection

## Unit Testing Framework

- NUnit, xUnit.net, MSTest, Approval Tests

## Mocking

- Moq, NSubstitute, FakeItEasy, JustMock

## UI Framework

- Angular, React, Vue, Prism

# LEAKY ABSTRACTIONS

## All non-trivial abstractions, to some degree, are leaky.

Abstractions fail. Sometimes a little, sometimes a lot. There's leakage. Things go wrong. It happens all over the place when you have abstractions.

--Joel Spolsky

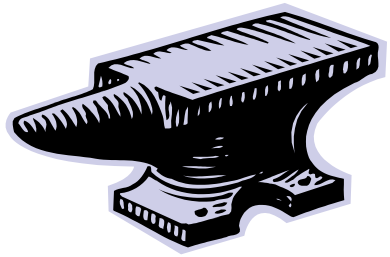
The Law of Leaky Abstractions

https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/

# ABSTRACTION IS AWESOME & AWFUL

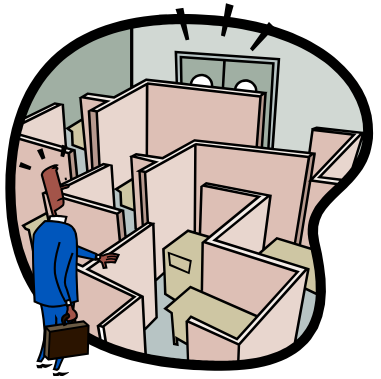
# THE GOLDILOCKS PRINCIPLE

**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

# Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for “Converge360 Events” in your app store
- Find this session on the Agenda tab
- Click “Session Evaluation”
- Thank you!

LIVE! 360
TECH EVENTS WITH PERSPECTIVE

# THANK YOU!

Jeremy Clark

- jeremybytes.com
- youtube.com/jeremybytes
- github.com/jeremybytes

https://github.com/jeremybytes/vslive2025-orlando