



# Abstract Art

## Getting Abstraction “Just Right”

**Jeremy Clark**  
**Developer Betterer**  
**GitHub: [jeremybytes](#)**

**Level: Intermediate**

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

Data Platform **LIVE!**  
INSIGHTS FOR MANAGING YOUR DATA ESTATE

TECHMENTOR  
IN-DEPTH TRAINING FOR IT PROS

Artificial  
Intelligence **LIVE!**  
AI FOR DEVELOPERS AND DATA SCIENTISTS

Cloud &  
Containers **LIVE!**  
CLOUD-NATIVE, PAAS & SERVERLESS COMPUTING

Cybersecurity  
& Ransomware **LIVE!**  
DEFENDING AGAINST RANSOMWARE AND OTHER ATTACKS



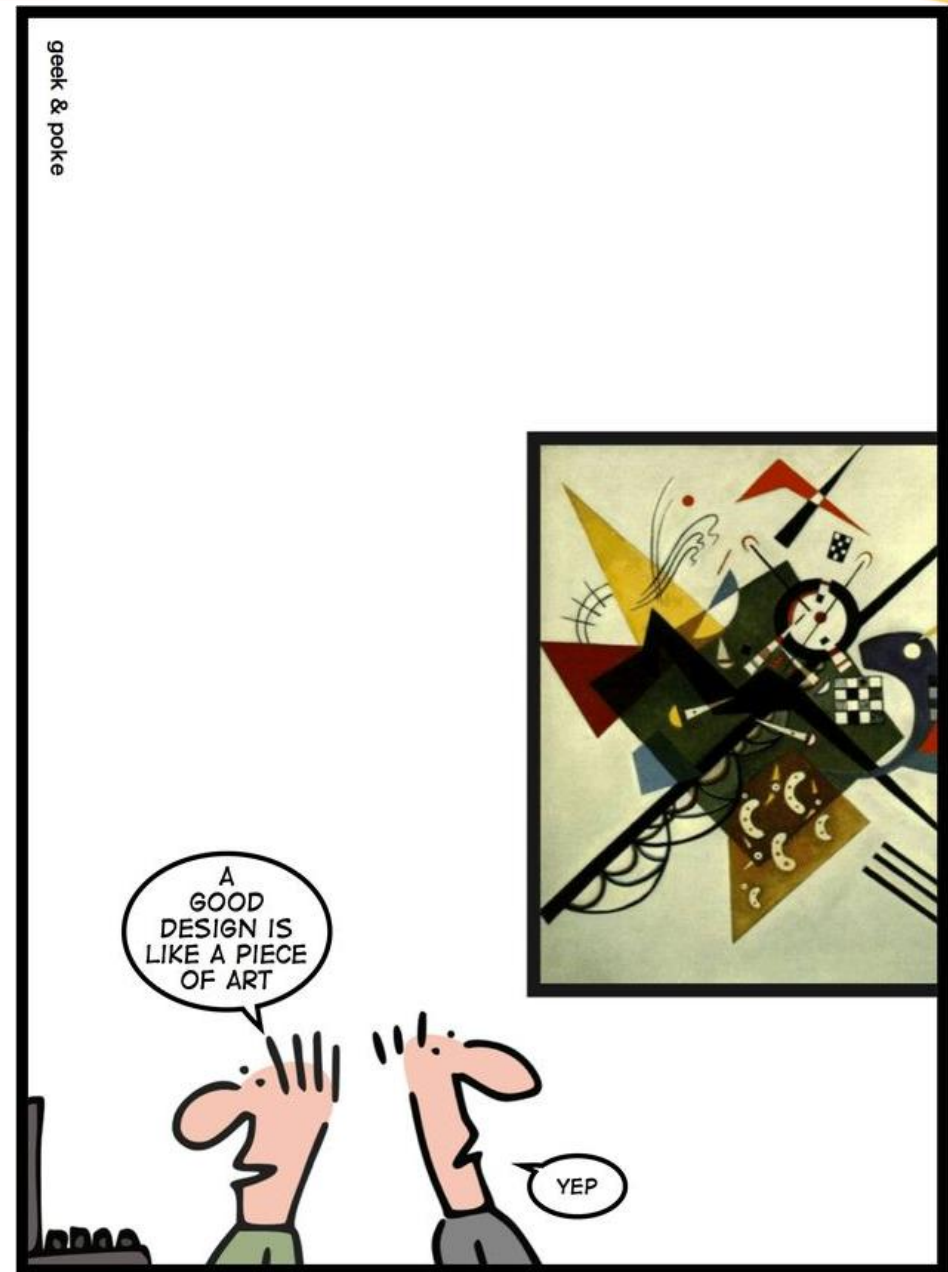
# Session Survey

- 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!



# A Good Design is like A Piece of Art

Geek & Poke – <http://goo.gl/ifu53l>



ABSTRACTION

# 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!



**Complexity**



**Confusion**



**Debugging  
Difficulty**

**Frustration**

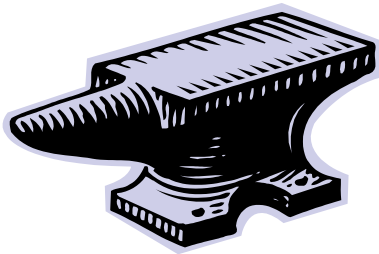


[https://archive.org/details/goldilocks\\_and\\_the\\_three\\_bears](https://archive.org/details/goldilocks_and_the_three_bears)



# GOLDBLOCKS THE DEVELOPER

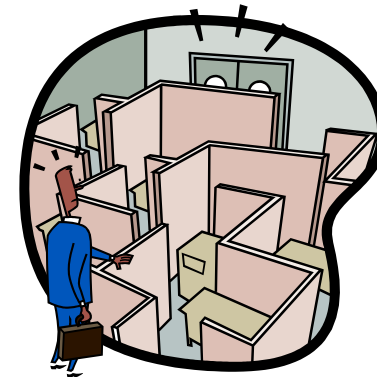
**Too Little**  
Abstraction



**Just Right**



**Too Much**  
Abstraction





# TWO TYPES OF DEVELOPERS

Over-Abstractor

The diagram consists of two horizontal rows. The top row features a teal rounded rectangle with the text 'Over-Abstractor' in white. To its left and right are thin teal lines that extend to the left and right edges of a larger, empty teal rectangular frame. The bottom row features a blue rounded rectangle with the text 'Under-Abstractor' in white. To its left and right are thin blue lines that extend to the left and right edges of a larger, empty blue rectangular frame.

Under-Abstractor



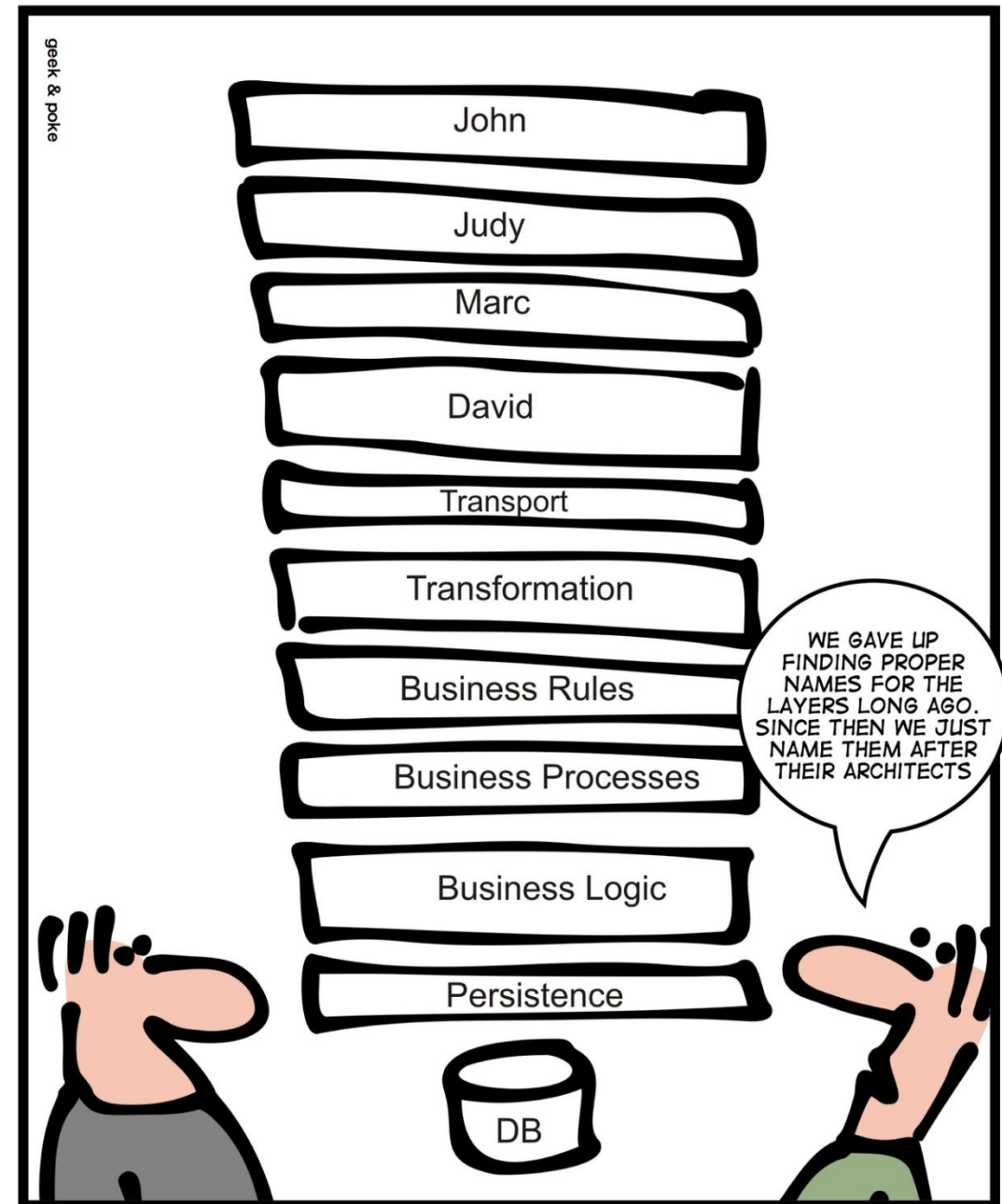
## Over-Abstractor

- “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>

©Jeremy Clark 2025



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...

+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>()
```

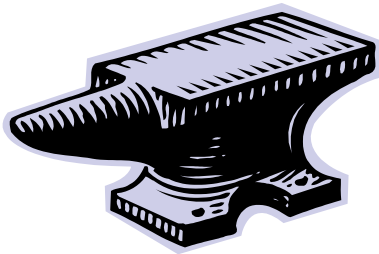
-1

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



# BE HONEST WITH YOURSELF

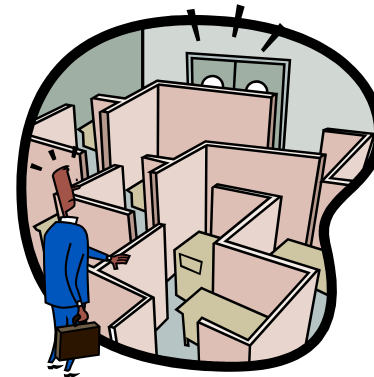
**Too Little  
Abstraction**



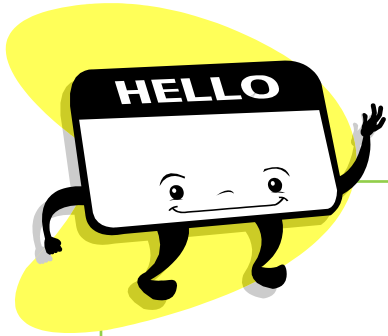
**Just Right**



**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 image displays a Reporting Application interface with three overlapping windows. The background window is titled "Report Viewer" and features a sidebar with three buttons: "Report List", "Add Report", and "Update Report", each with a right-pointing arrow icon. The main content area of this window shows a table with two columns: "Report Title" and "Category". The table contains the following data:

Report Title	Category
Quarterly Financials	
Weekly Sales	
Inventory	
Pending Shipments	
Incoming Shipments	
Advertising Sales	

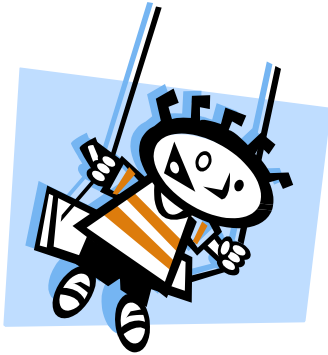
The middle window, also titled "Report Viewer", is titled "Pending Shipments". It contains fields for "Start Date" (8/24/2014), "End Date" (8/24/2014), and "Additional Info". To the right of these fields is a calendar for August 2014.

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

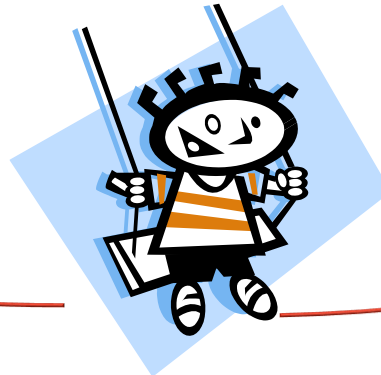
The foreground window is the main application window. It has a menu bar with "New Report File", "Edit Report", and "Edit Parameters". Below the menu bar is a tab bar with "Splash", "Report List", "Parameter Values", and "Report Viewer". The "Report Viewer" tab is active, showing a form with "Report Title" and "Category" fields, and a "Run" button at the bottom right.

# THE PENDULUM EFFECT

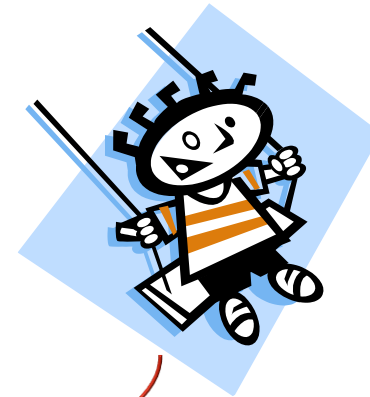
**Under-  
Abstraction**



**Just Right**



**Over-  
Abstraction**



# REPORTING COMPONENT

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

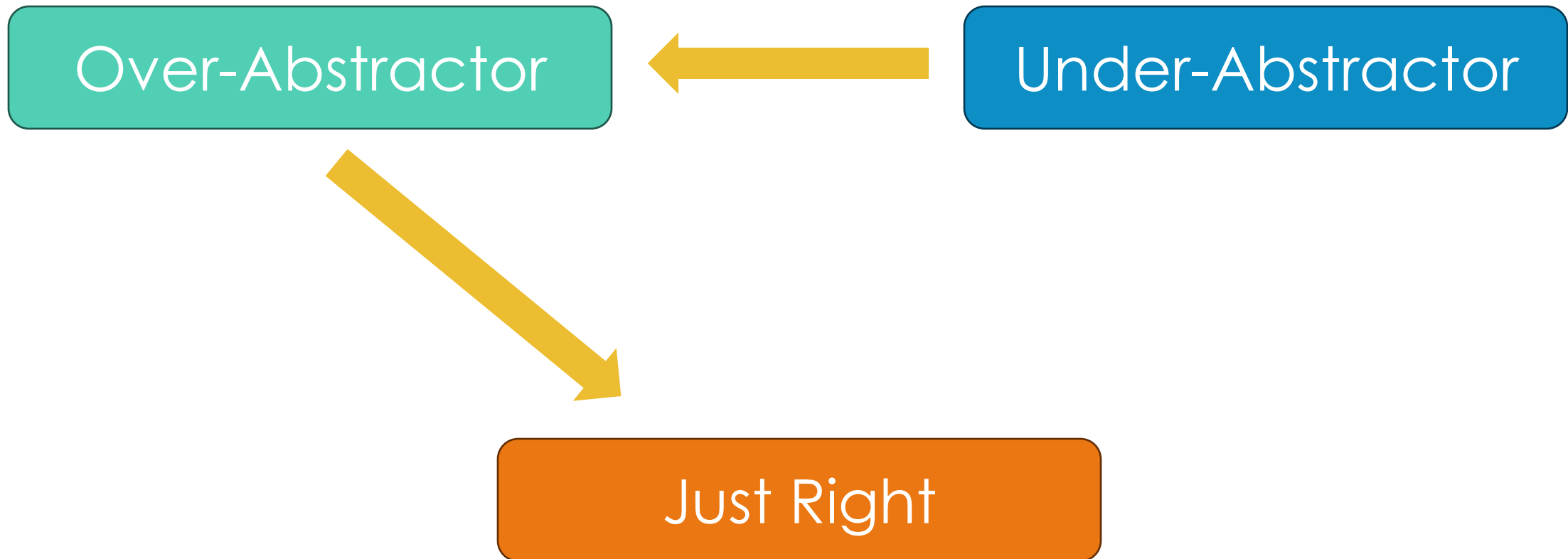


# THOSE AROUND YOU

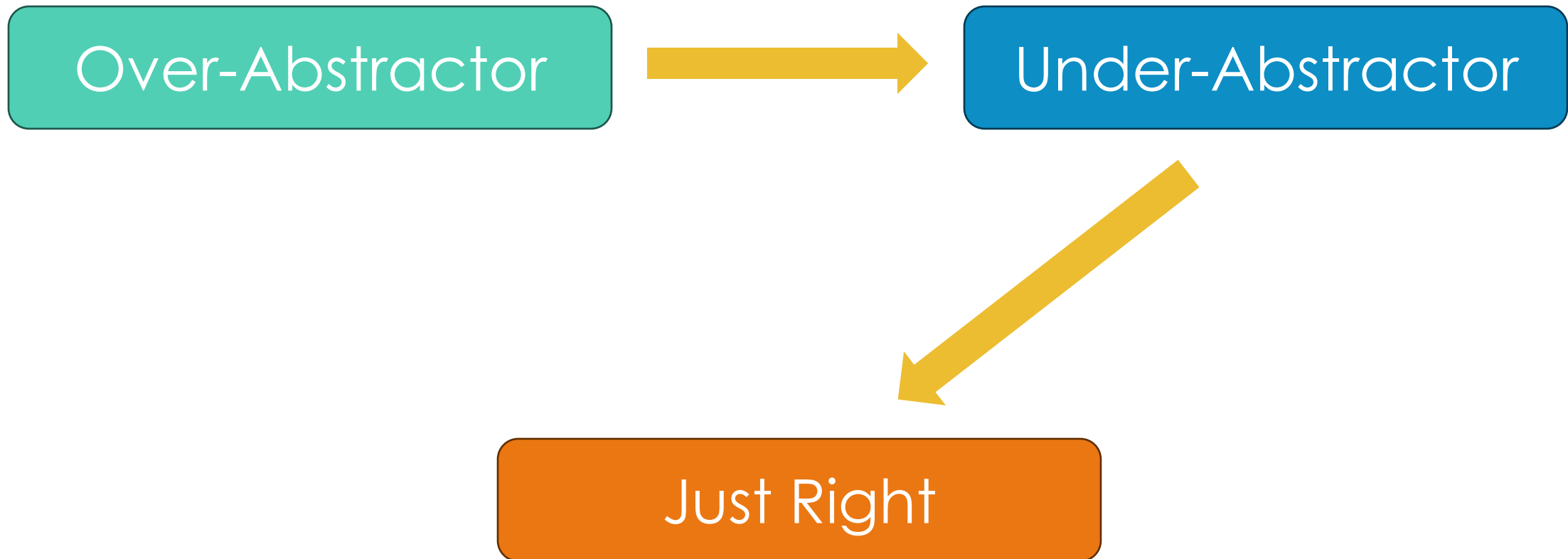
## Over-Abstractor

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

# A SYMBIOTIC RELATIONSHIP



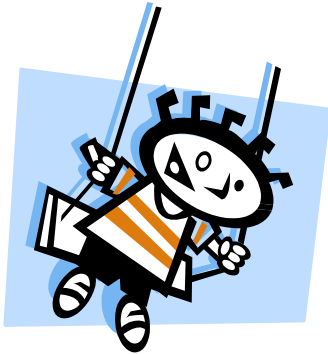
# A SYMBIOTIC RELATIONSHIP



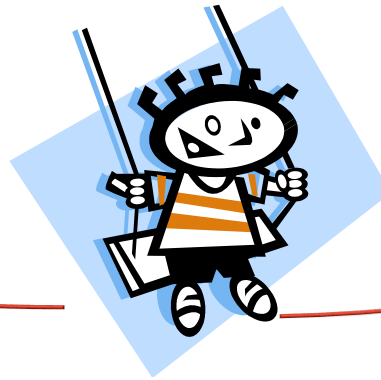


# THE PENDULUM EFFECT

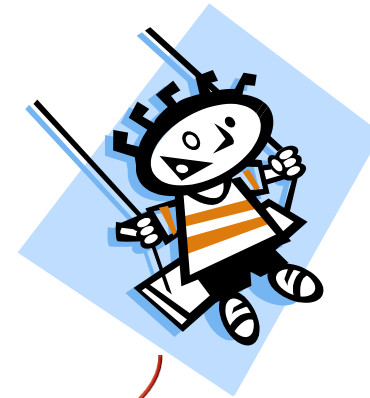
**Under-  
Abstraction**



**Just Right**



**Over-  
Abstraction**





# VARIOUS DATA SOURCES

Microsoft SQL Server

MongoDB

CSV

WebAPI

Oracle

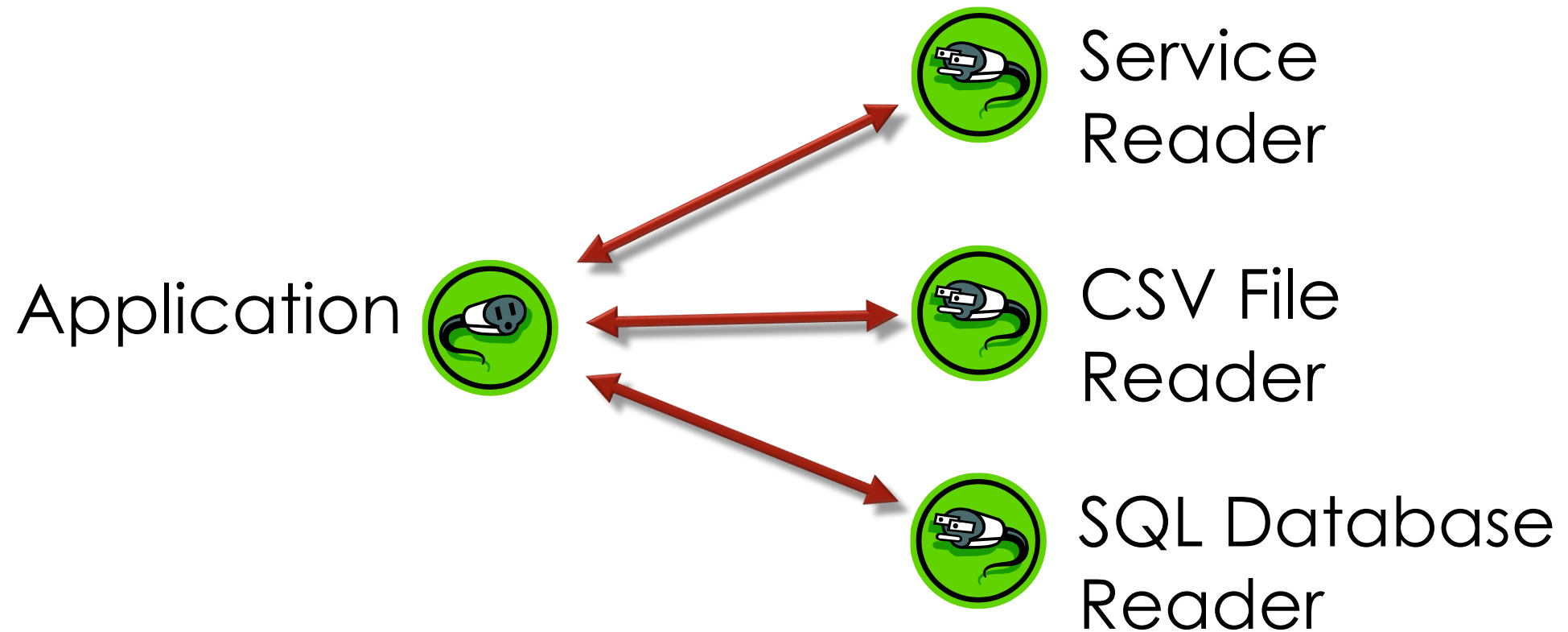
Amazon RDS

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-Abtractor

# 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



## 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

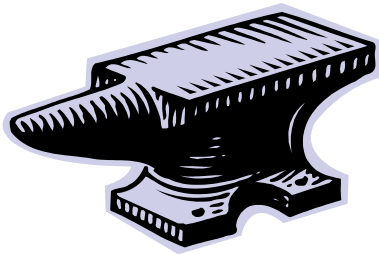
<https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/>

# ABSTRACTION IS AWESOME & AWFUL



# THE GOLDBLOCKS PRINCIPLE

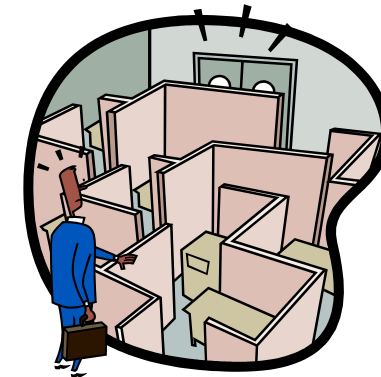
**Too Little**  
Abstraction



**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!





THANK YOU!

Jeremy Clark

- [jeremybytes.com](https://jeremybytes.com)
- [youtube.com/jeremybytes](https://youtube.com/jeremybytes)
- [github.com/jeremybytes](https://github.com/jeremybytes)

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