DelphiDay italian conference PIACENZA EDITION

Modernization of the legacy VCL projects

Bogdan Polak
bogdan.polak@bsc.com.pl

wintech italia

embarcadero reseller partner

Bogdan Polak

BSC Polska (Poland)

bogdan.polak@bsc.com.pl

bogdanpolak

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

ABOUT ME

26 yrs developer

19 yrs Delphi developer

14 yrs trainer / mentor / sales engineer

4+ yrs project modernization

Company:

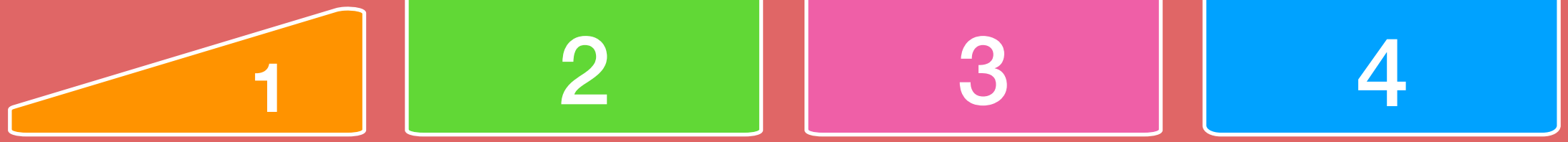
BSC Polska

- Embarcadero Partner @ Poland
- Sales and Training
- 1000+ customers
- 80% Delphi teams
- Community supporter

INTRODUCTION

Code Maturity

It was
Messy code



Working code

Code as you like

No rules

Just working code

Bitten code

Zero warnings policy

Cleaning uses section

Clean code

Refactoring tasks

Clean code rules

Code reviews

Formatted code

Style Guide

Solid code

Reusable code

Good practices

Patterns and principles

Architect on board

Testable code

Developers, Developers, Developers …

Stagnation

Current tasks prevent changes

Demotivation

Developers aren’t motivated to change

Dinosaurs

Old-timers, difficulties in recruiting younger developers

Bad habits

Developers are aware that they can work better

Black hole

This project will never be better, its destiny is a death

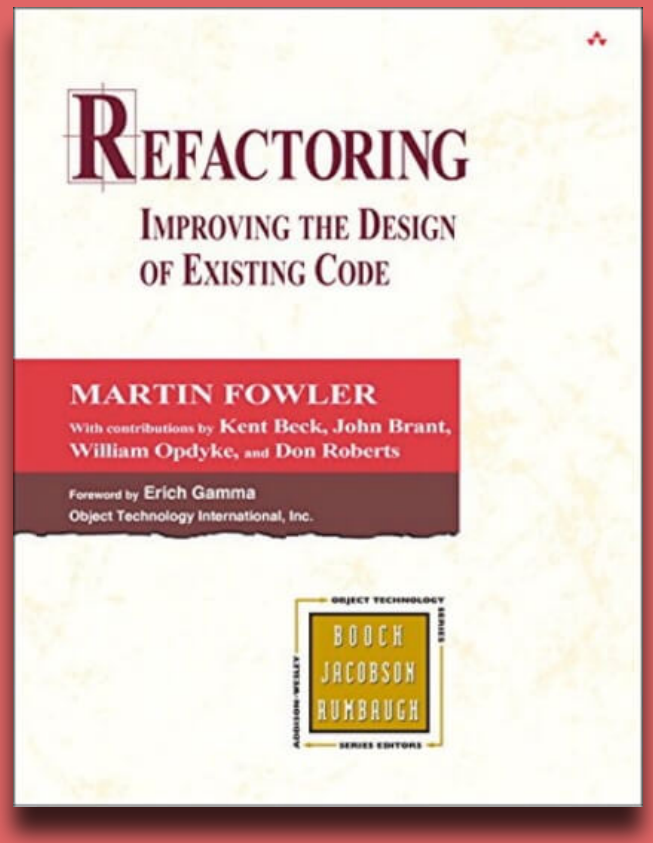
Inspiration

Erich Gamma in the book foreword

*„When my coauthors and I wrote Design Patterns, we mentioned that **design patterns provide targets for refactorings**. However, identifying the target is only one part of the problem; transforming your code; **so that you get there is another challenge**."*

*„The refactorings in this book will help you change your code **one small step at a time**, thus reducing the risks of evolving your design."*

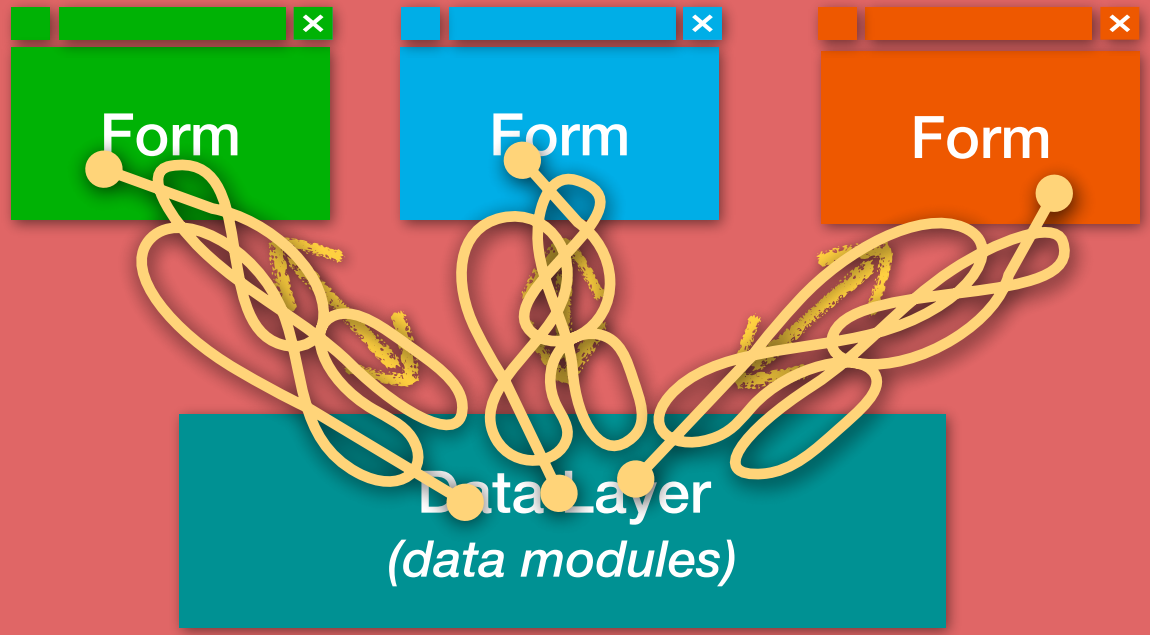
*„My first experience with disciplined, "one step at a time" refactoring was when I was pair-programming at 30,000 feet with **Kent Beck**. He made sure that we applied refactorings from this book's catalog one step at a time. I was amazed at **how well this practice worked**."*



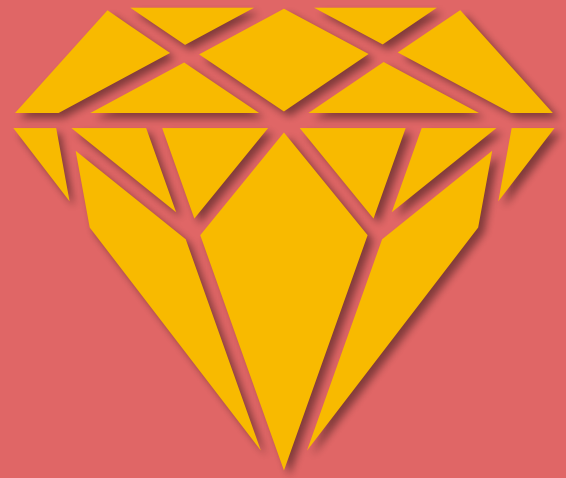
Martin Fowler

Starting point

- OnClick coding
- Visual everything
- Spaghetti
- Form centric
- Copy - paste - customize
- Bug driven development
- Get lost! My code is mine



Expected Benefits



- **Satisfaction** - improved
- **Efficiency** - increased
- **Respectability** - fewer errors in the code
- **Team work** - developers will learn how to work together
- **Cleanliness** - cleaner code, defined good practices
- **Recruitment** - easier to introduce a new team members

- **Motivation** - gamification, learning, trainings, open-source activities, hackatons, architect role
- **Visibility** - what others are doing, quality and performance indicators
- **Predictability** - task planing and effort reporting

Effort

10%

If you are coding 40 hours per week

4 h

The Plan

The Plan

➔ Phase 1 -- Clean-ups

➔ Phase 2 -- Extraction

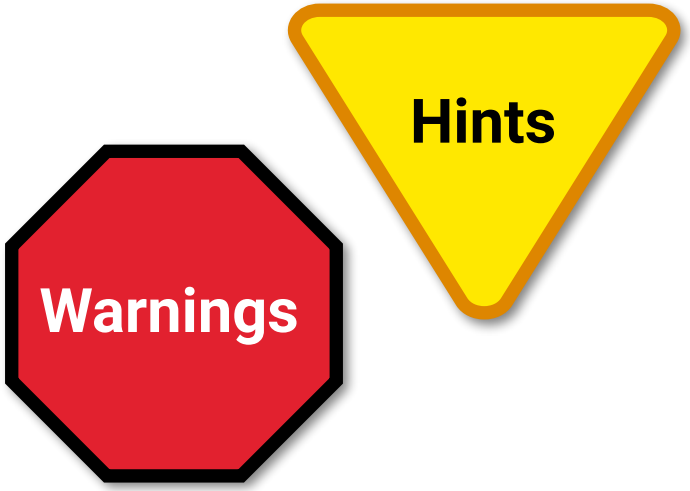
➔ Phase 3 -- Decomposition

➔ Phase 4 -- Improvements

Phase 1

1

Clean-ups




```
Unit SuperStar;

interface

uses … ;

implementation

uses … ;
```

Action

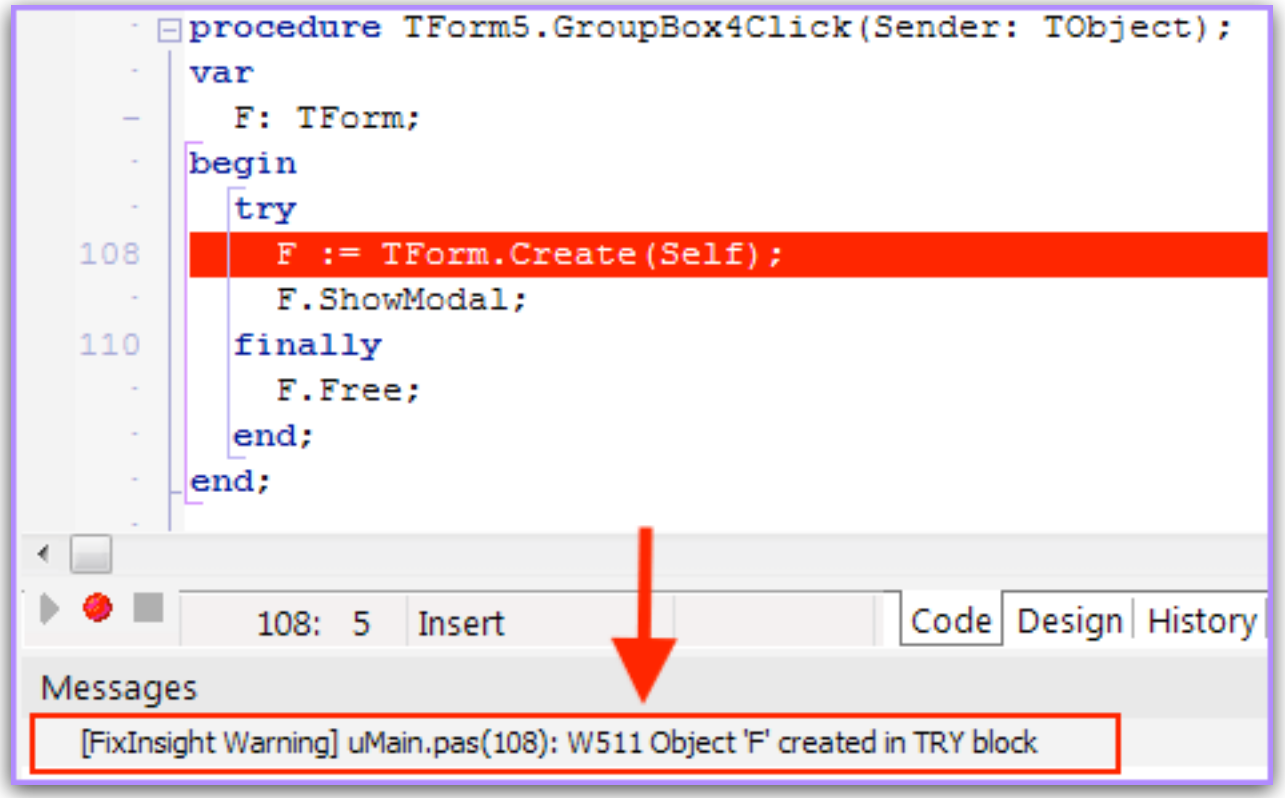
- Warnings and hints - remove
- Uses section - clean-up
- Long methods - divide (extract smaller ones)
- Dead code - remove
- Outdated comments - remove
- TODO - add

Like This - Need More

DEV

TMS FixInsight

Code analysis tool for Delphi developers that detects issues in Delphi source code.



VISIBILITY - MEASURES

- WT - Warning Toxicity

  [Number Of Warnings And Hints] / [Number Of Units]

- PUP - Public Uses Pollution

  [Number Non RTL/VCL Units in the Interface Uses Section]

- **Automation !!!**

Scout Badges

How To Mesure Delphi Code?

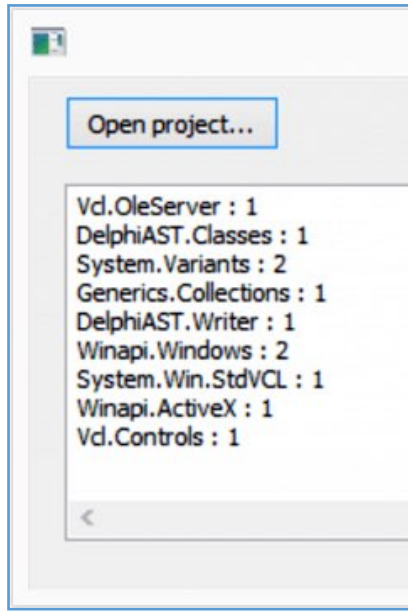
Delphi Metrics

```
AuditsCLI.exe --metrics
```

http://docwiki.embarcadero.com/RADStudio/Rio/en/AuditsCLI.EXE,_the_Command_Line_Audits_and_Metrics_Tool

DelphiAST

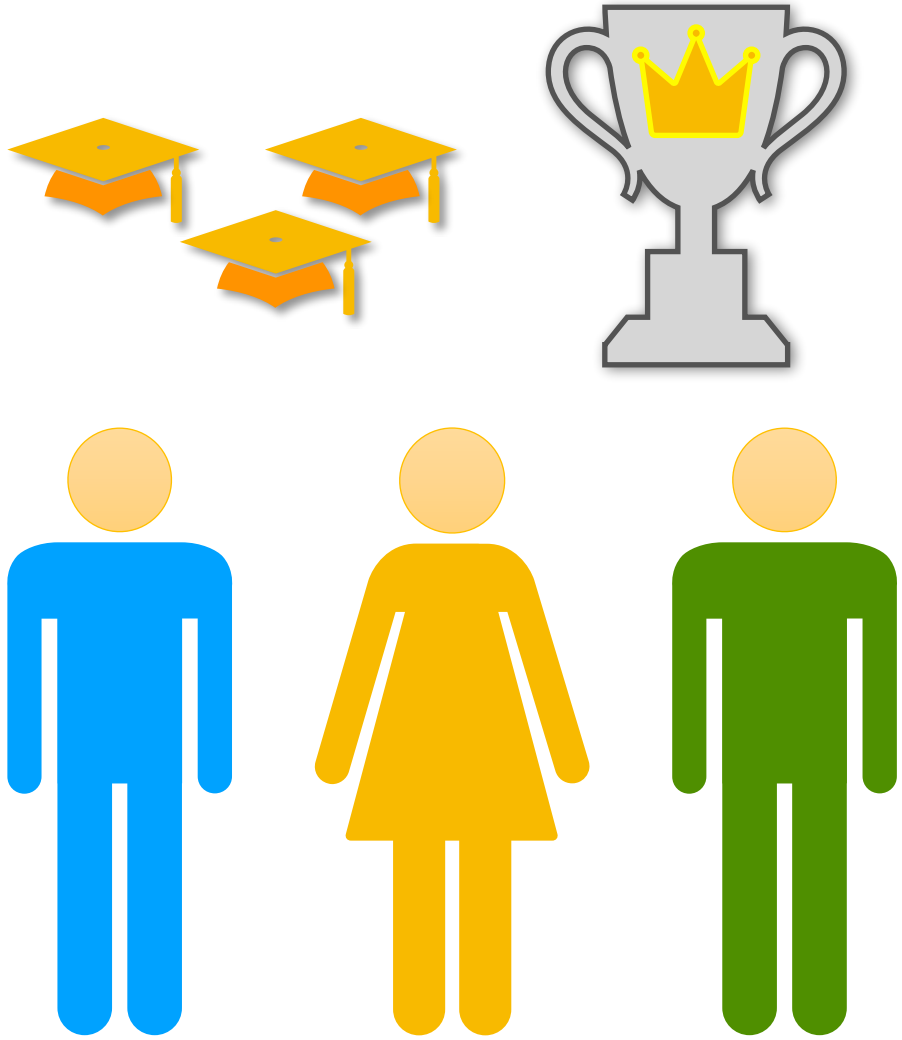
Abstract Syntax Tree



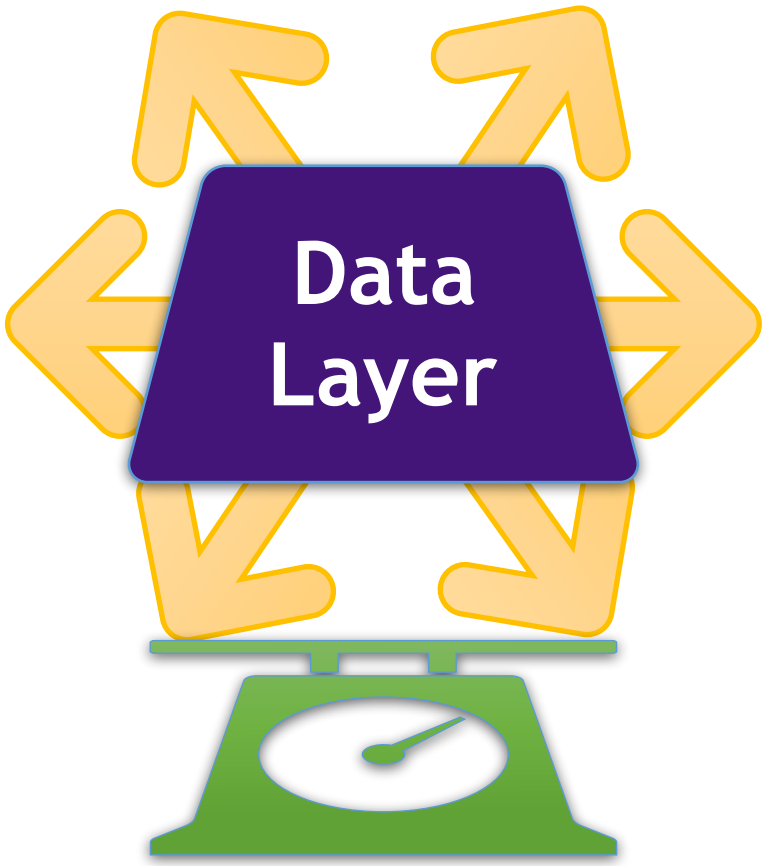
https://github.com/RomanYankovsky/DelphiAST

Phase 1 - Learning & Improvement

- Gamification
  - Public measures
  - Winner Company Board
  - Modernization as a game
- TODO
  - Adding
  - Removing
- Internal trainings
  - How to …



Phase 2



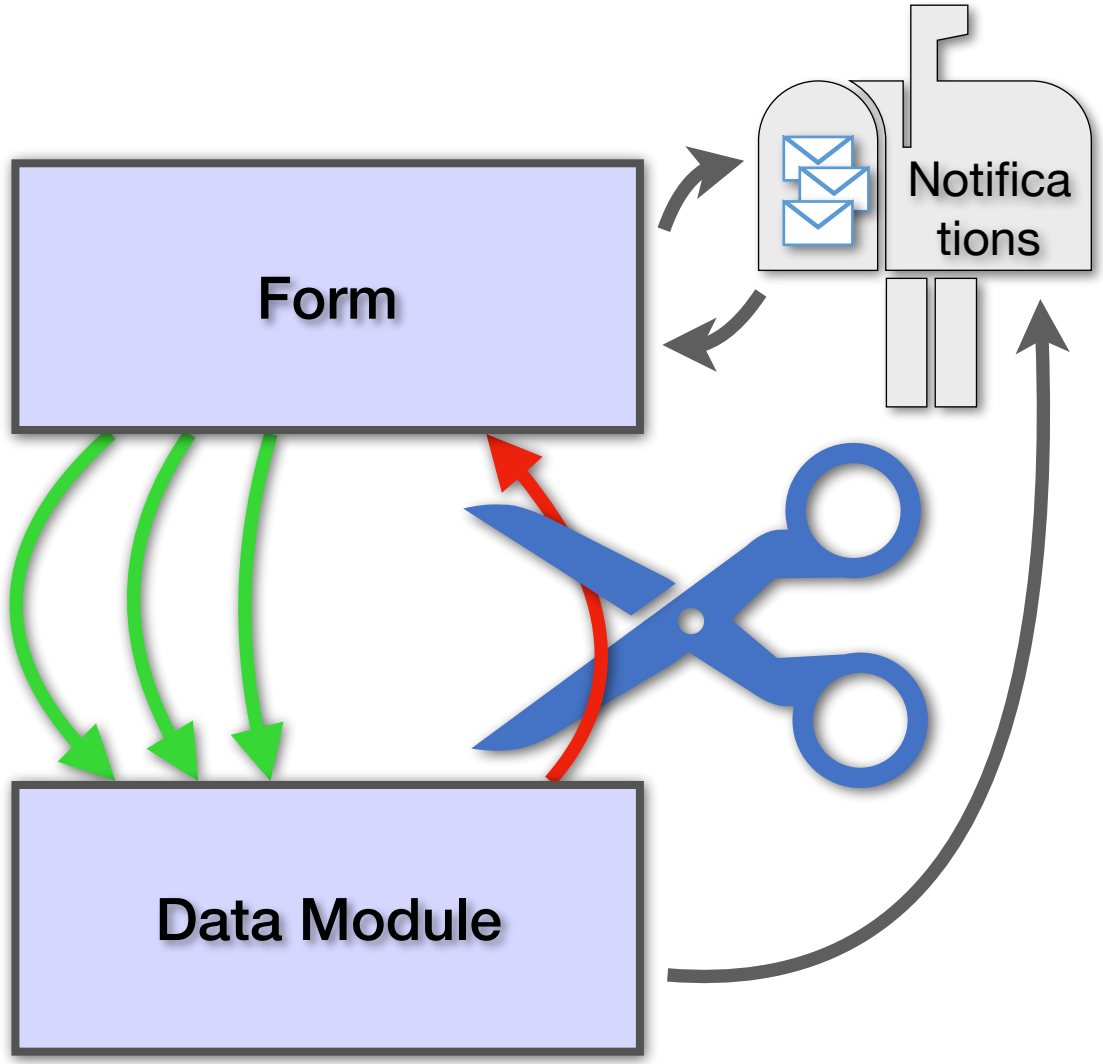
2

Extraction

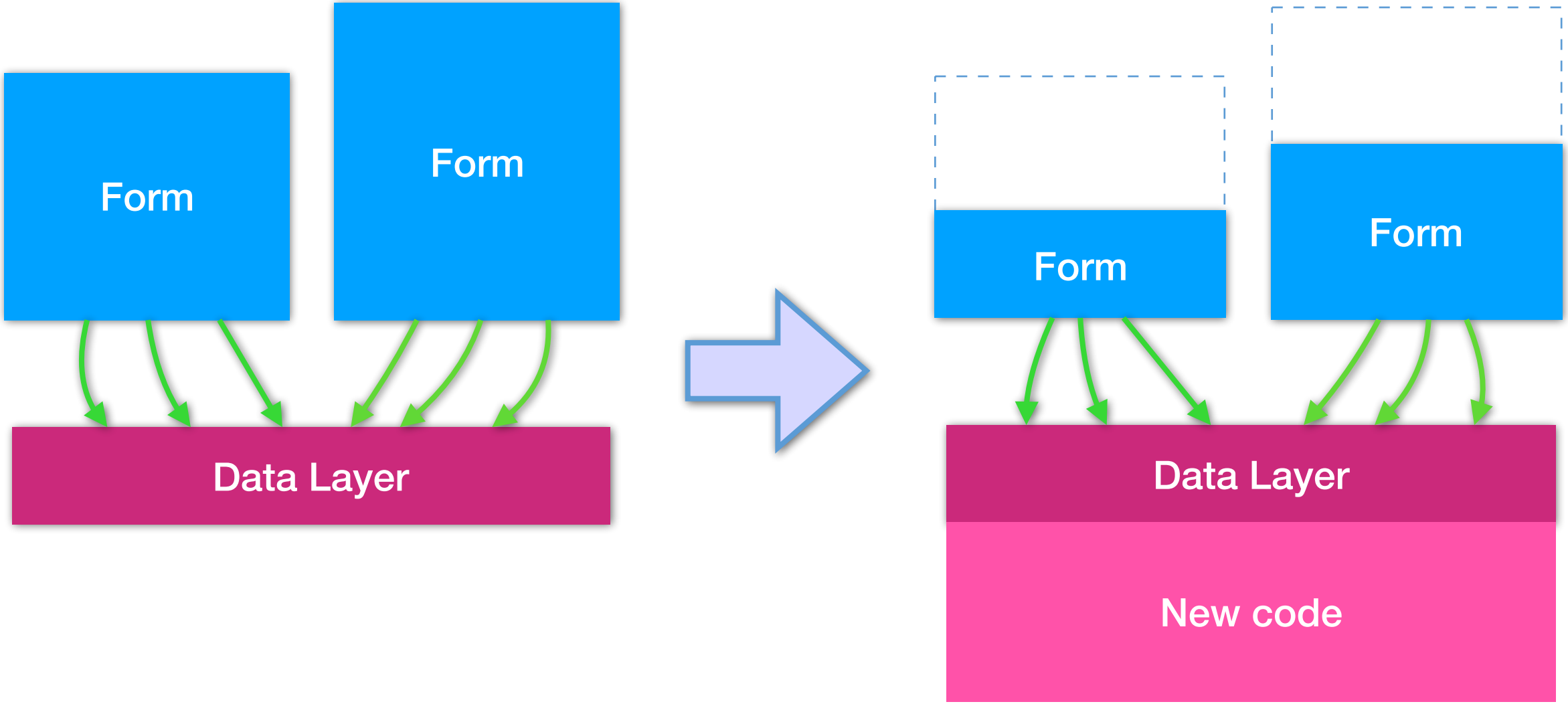
Move code to Data Modules / Data Units

Action 1 - Remove Dependencies

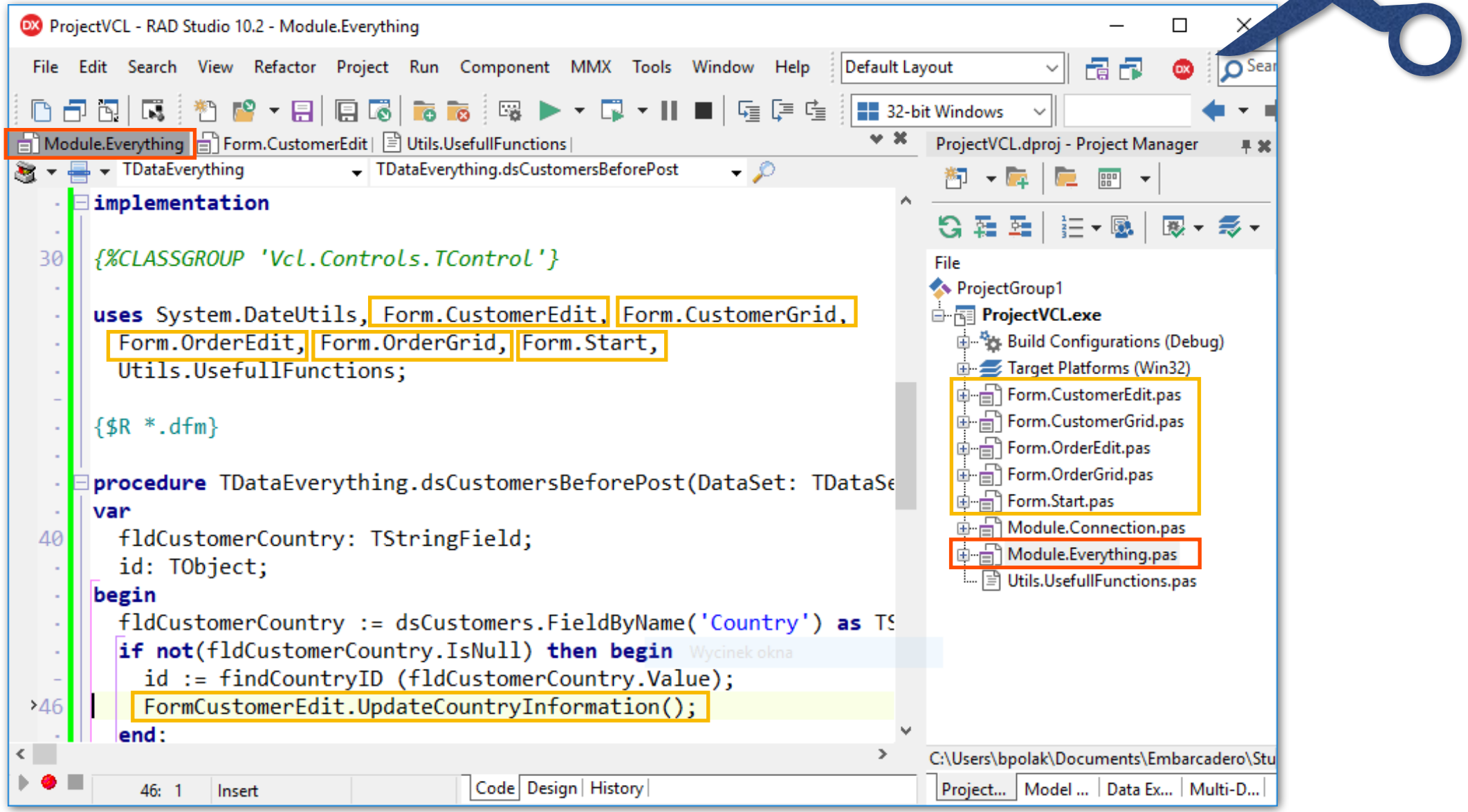
1. **Find all „red lines"**
   - Measure
2. **Move code**
   - Extract this code into a separate method
   - Move this methods to the form
3. **Remove dependencies**
   - Clean uses
   - Remove all forms



Action 2 - Increase Data Layer Weight

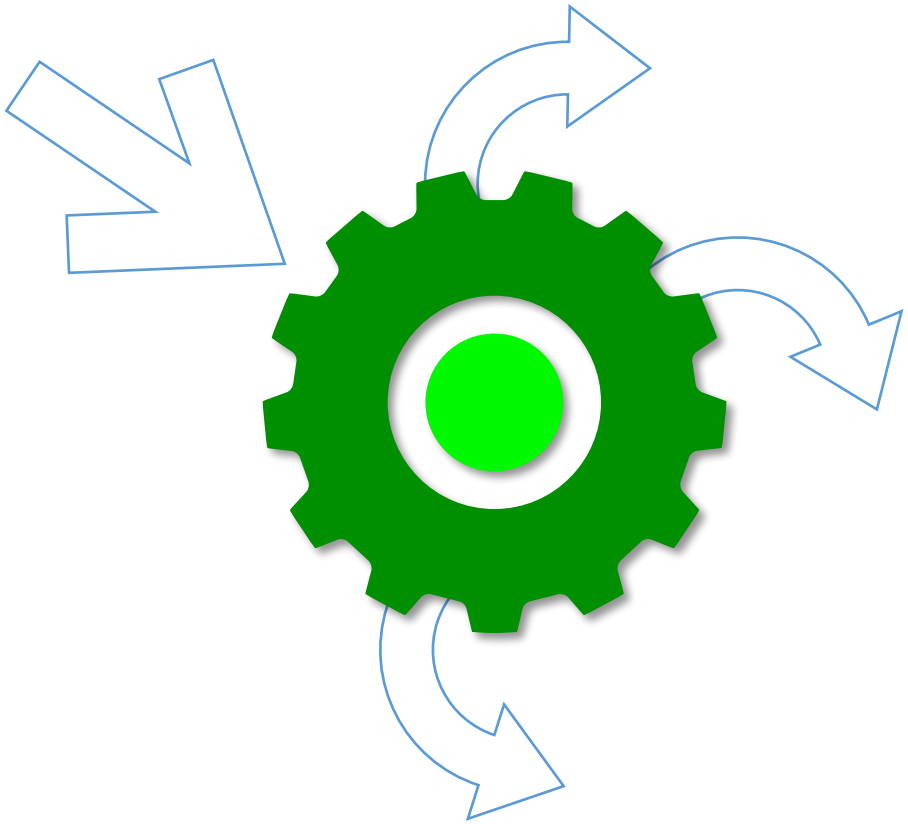


STYLE GUIDE


ProjectVCL - RAD Studio 10.2 - Module.Everything
File Edit Search View Refactor Project Run Component MMX Tools Window Help
Default Layout
32-bit Windows
Module.Everything
Form.CustomerEdit
Utils.UsefullFunctions
TDataEverything
TDataEverything.dsCustomersBeforePost
implementation
{%CLASSGROUP 'Vcl.Controls.TControl'}
uses System.DateUtils, Form.CustomerEdit, Form.CustomerGrid,
Form.OrderEdit, Form.OrderGrid, Form.Start,
Utils.UsefullFunctions;
{$R *.dfm}
procedure TDataEverything.dsCustomersBeforePost(DataSet: TDataSe
var
fldCustomerCountry: TStringField;
id: TObject;
begin
fldCustomerCountry := dsCustomers.FieldByName('Country') as TS
if not(fldCustomerCountry.IsNull) then begin
id := findCountryID (fldCustomerCountry.Value);
FormCustomerEdit.UpdateCountryInformation();
end;
46: 1
Insert
Code Design History
ProjectVCL.dproj - Project Manager
File
ProjectGroup1
ProjectVCL.exe
Build Configurations (Debug)
Target Platforms (Win32)
Form.CustomerEdit.pas
Form.CustomerGrid.pas
Form.OrderEdit.pas
Form.OrderGrid.pas
Form.Start.pas
Module.Connection.pas
Module.Everything.pas
Utils.UsefullFunctions.pas
C:\Users\bpolak\Documents\Embarcadero\Stu
Project... Model ... Data Ex... Multi-D...

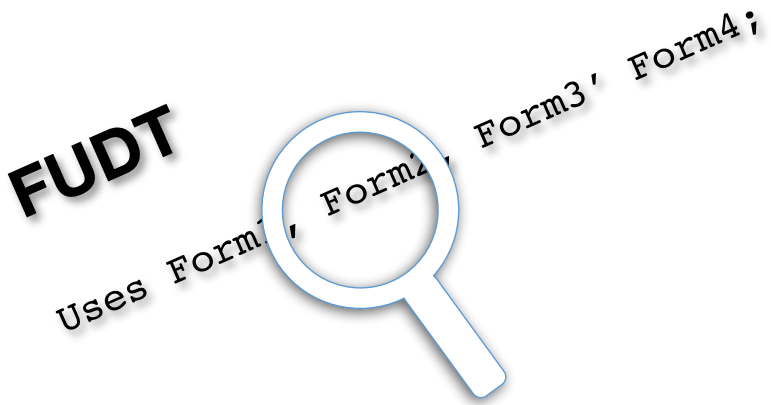
Notification Pattern

1. Timer & Flag
2. Callback / Event
3. Event Bus
4. Observer Pattern
5. … (what you like) (just do it)

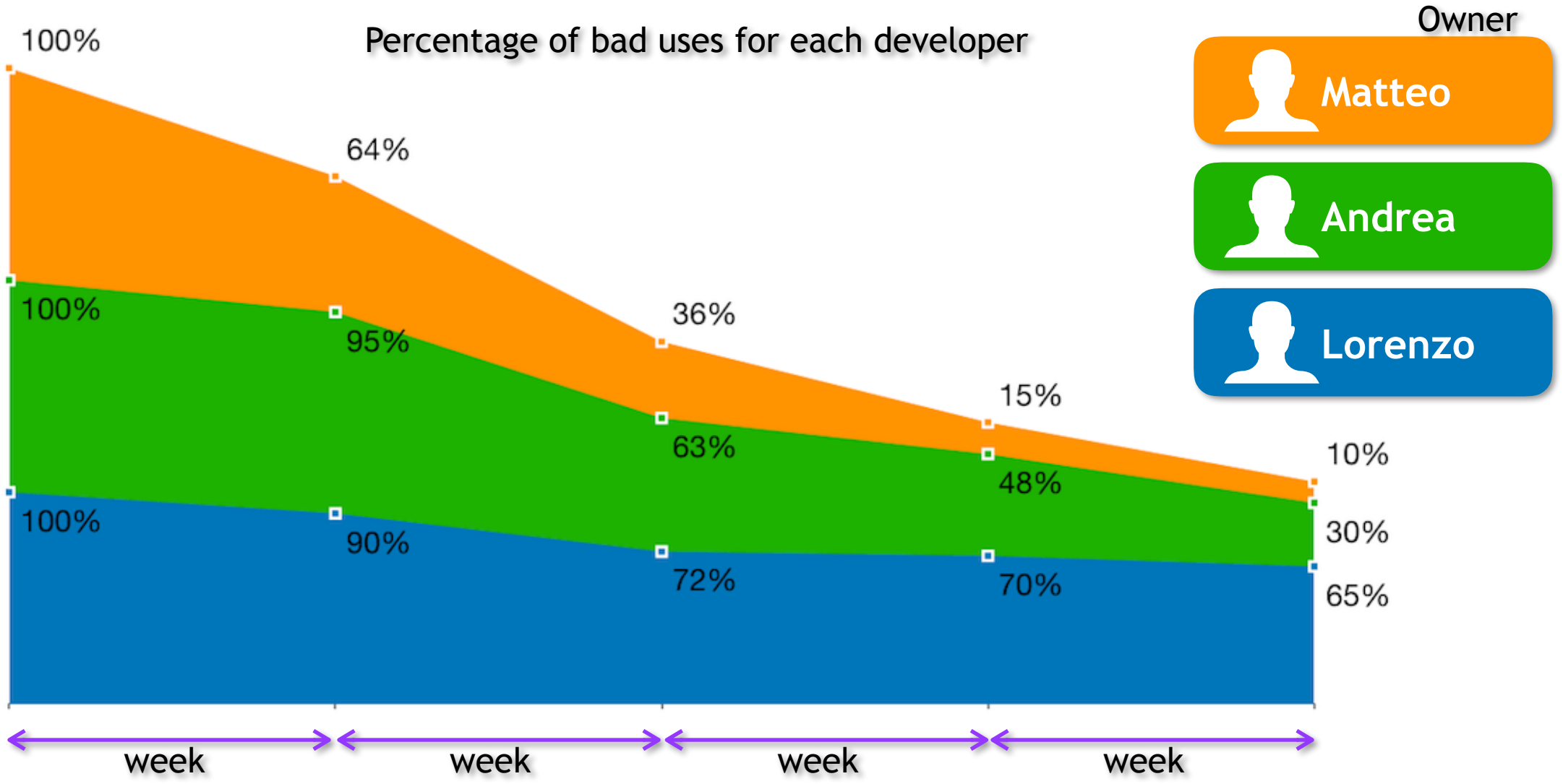


Measures

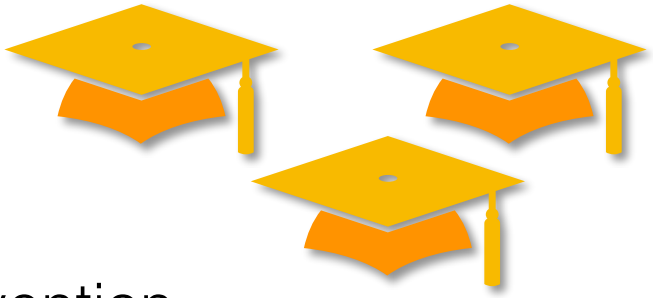
- **FUDT**
  - **Forms Used by Data layer Toxicity**
  - Number of forms used by a data layer
- **D-LOC**
  - Lines of code in the data layer
- **D-NOO**
  - Number of operations in the data layer
  - Number of the private / public methods
  - Important: private methods



GAMIFICATION - TOXICITY GRAPH

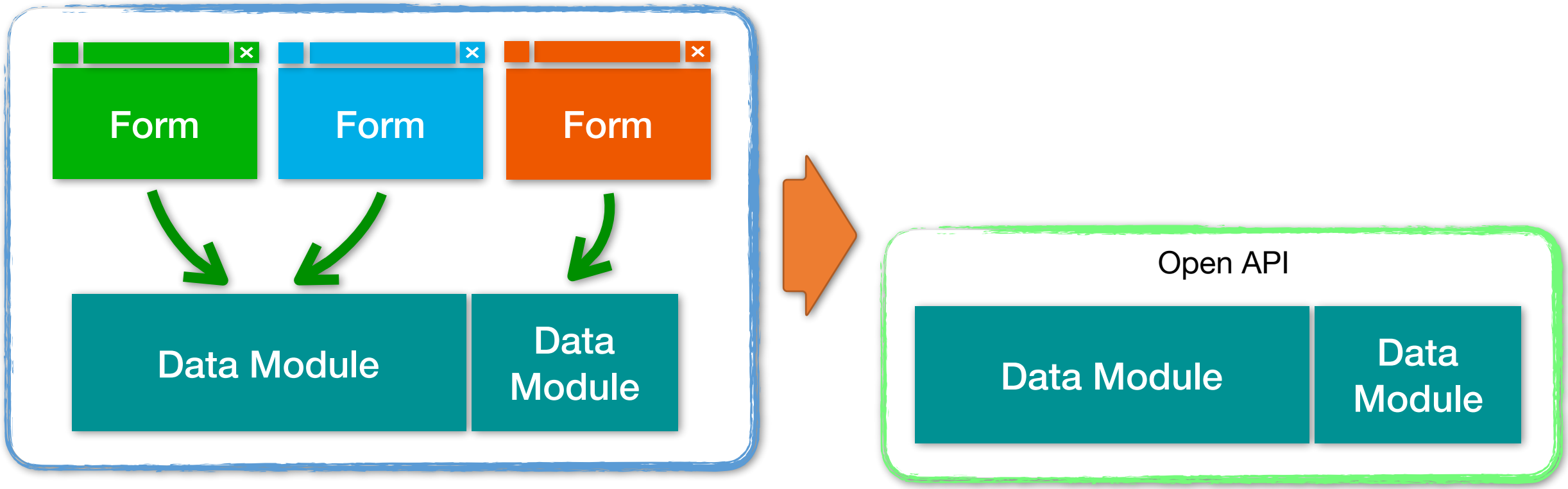


Phase 2 - Learning

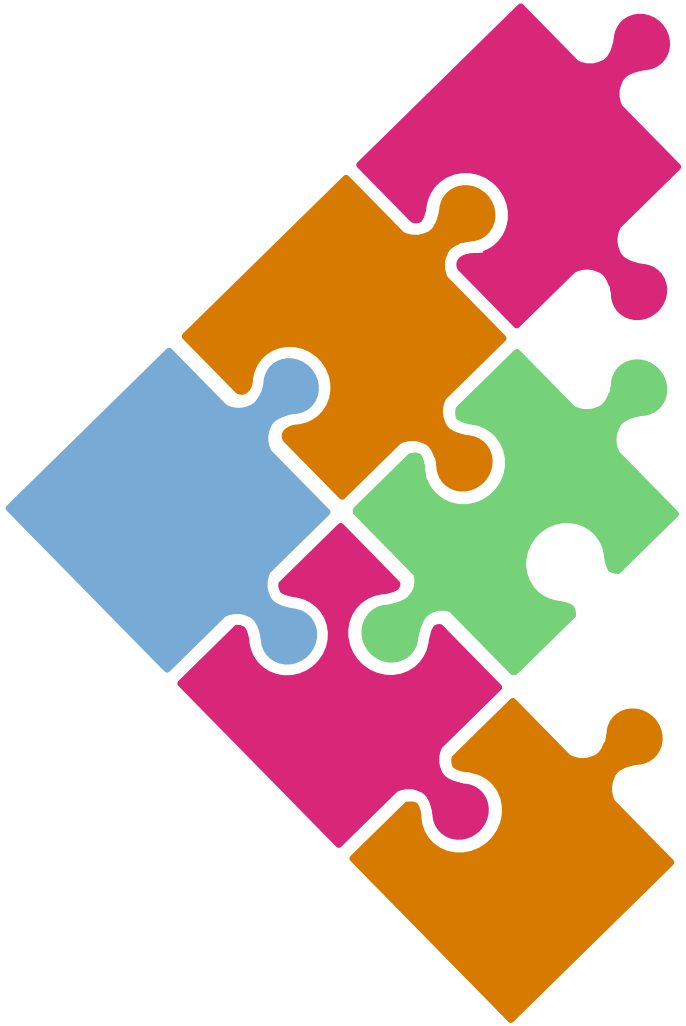


- Repository
  - Git, git and git
- Code reviews
  - How to be a good reviewer
  - Review: new methods in the data module
  - Discuss TODO's
  - Internal communication platform like GitHub
- Style Guide
  - Naming Convention
  - Code formatting
- Cleaning Code
  - Smaller methods
  - Self explaining and readable code

Side Effect

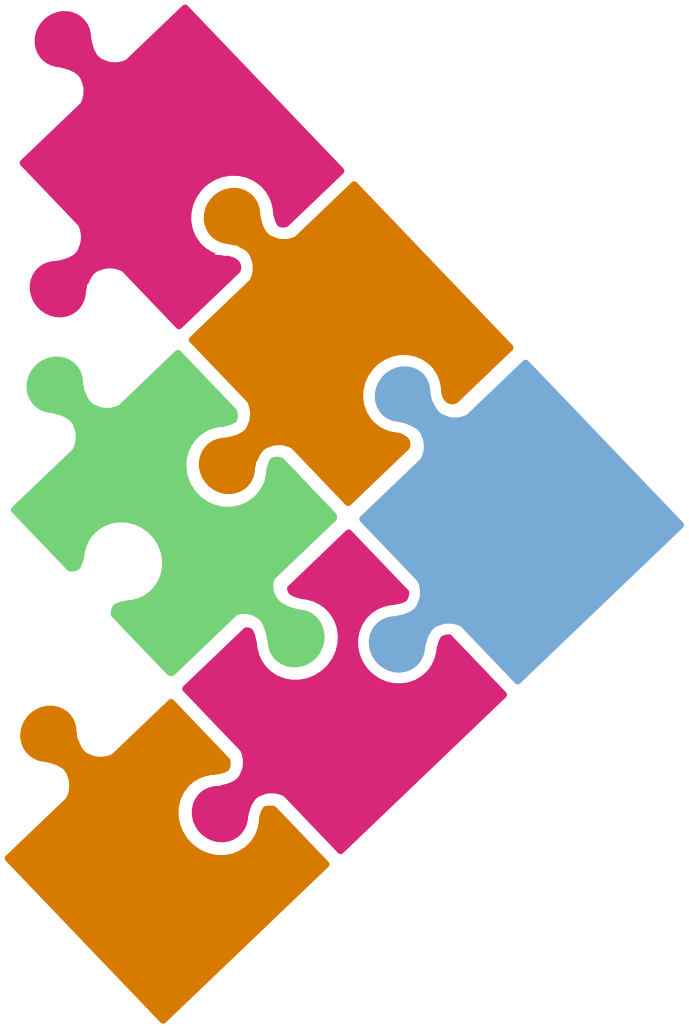

Form
Form
Form
Data Module
Data Module
Open API
Data Module
Data Module

Phase 3

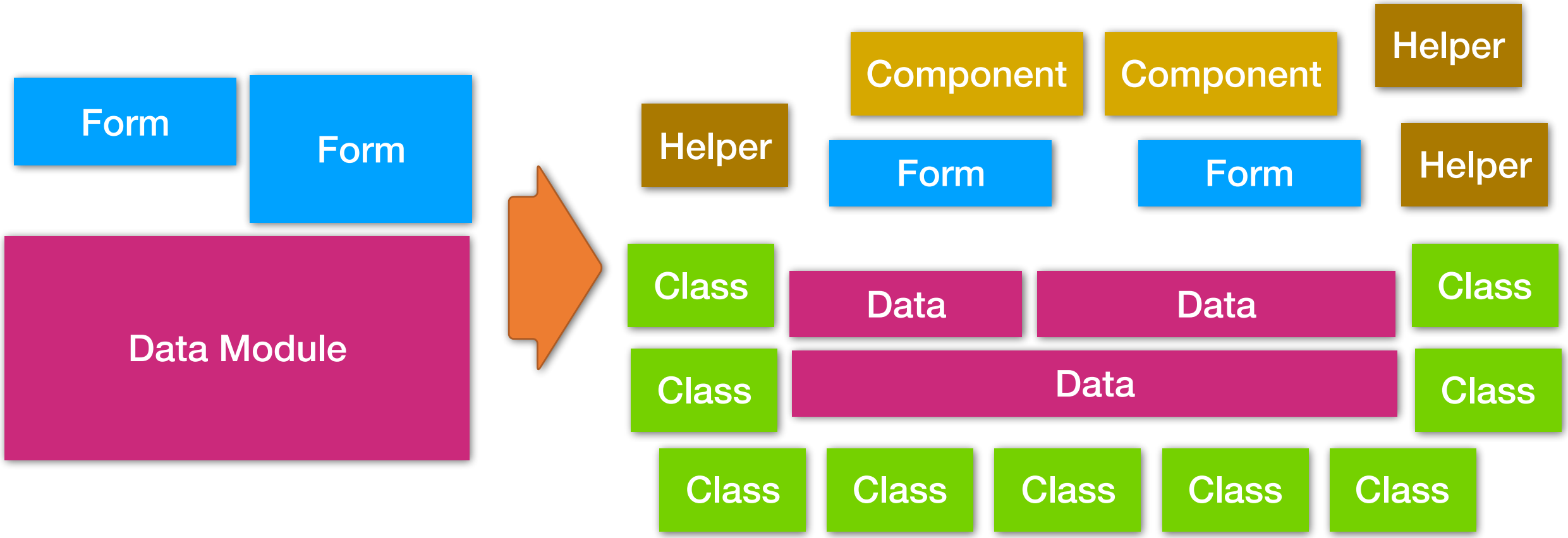


3

Decomposition

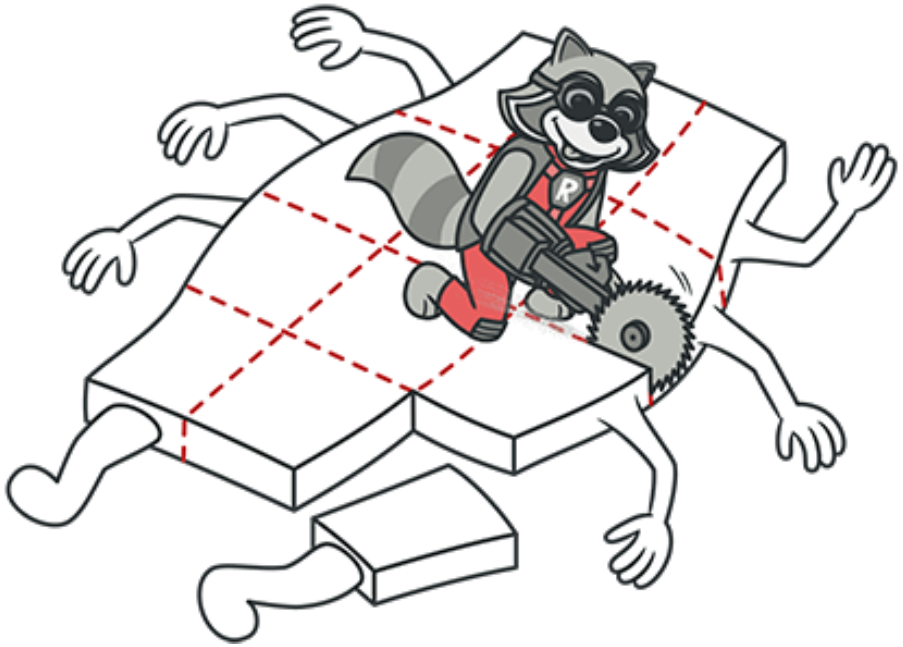


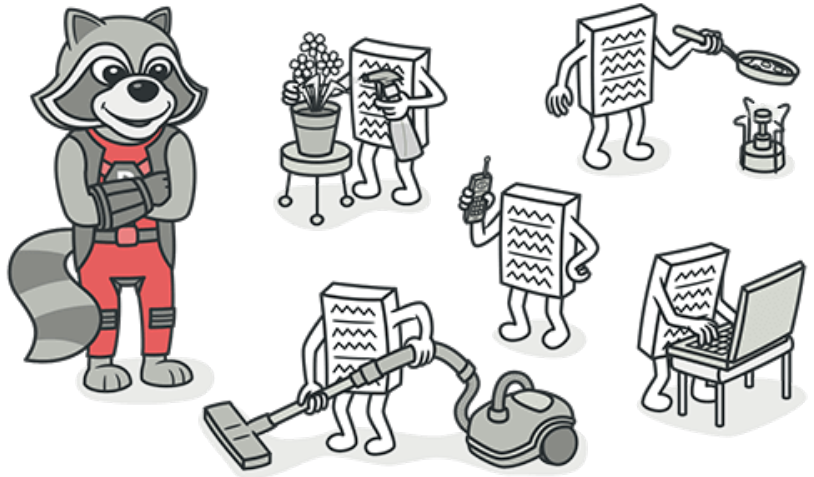
Action Plan


Form
Form
Data Module
Component
Component
Helper
Helper
Form
Form
Helper
Class
Data
Data
Class
Class
Data
Class
Class
Class
Class
Class
Class
Class

Action Plan

Decomposition





Single responsibility

Images from: https://refactoring.guru/smells

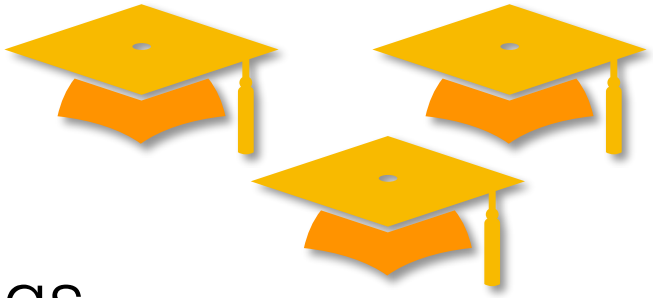
Coding Techniques

- OOP fundaments
  - Abstraction
  - Encapsulation
  - Decomposition
  - Generalization
  - Separation of Concerns
- Delphi techniques
  - Class helpers
  - Anonymous methods
  - Generic types
  - TAction
  - TComponent

Phase 3 - Measures

- **MOL**
  - Maximum operation length
  - Counts numer of lines in each method
- **NOC**
  - Number of Classes
- **MNOP**
  - Maximum Number Of Parameters
  - Counted for each function
- **NOHC**
  - Number of Helper Classes

Phase 3 - Learning



- Continuous Integration
- Code reviews
  - Review commits
- Common Code
  - Helpers repository
  - Components (not registred)
  - Reusable OOP code

- Internal trainings
  - Unit tests
  - Law of Demeter
  - SRP
  - Patterns
  - OOP libraries (Spring4D,…)
  - Pair Programming sessions

Phase 3 - It's time to find

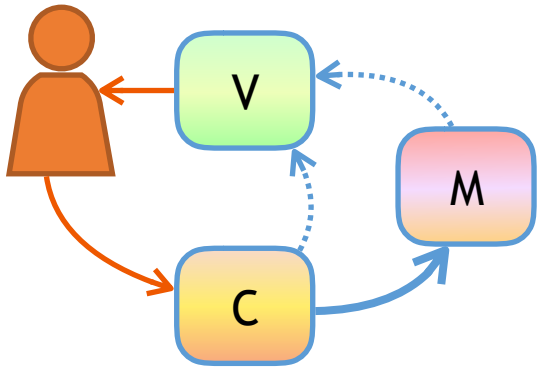
- **Architect**
  - Migration Leader
  - Not a manager
  - Adviser and coach
  - Book reader
  - Blog writer
  - Community activist

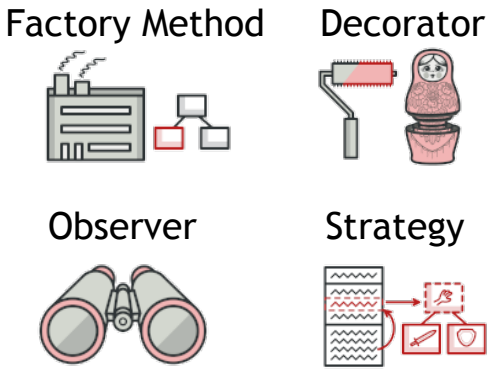
Phase 4

4

Improvement

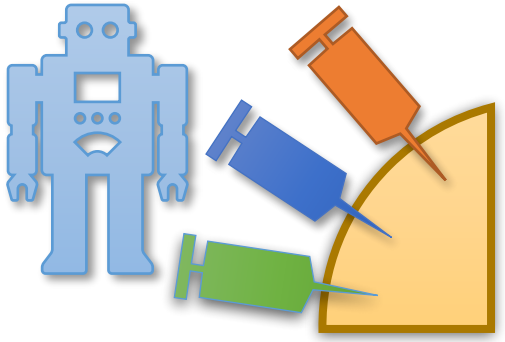
Action





GoF Patterns

SOLID principles



Dependency Injection Manager

Action Plan

- Architectural communication
  - SOLID principles
  - MVC
  - GoF Patterns
- Domain knowledge
  - Bounded context
  - Continuous integration
- SOA
  - Micro-services
- Code decoupling
  - Interfaces
  - Minimize the number of dependencies
  - Autonomous units
- TDD

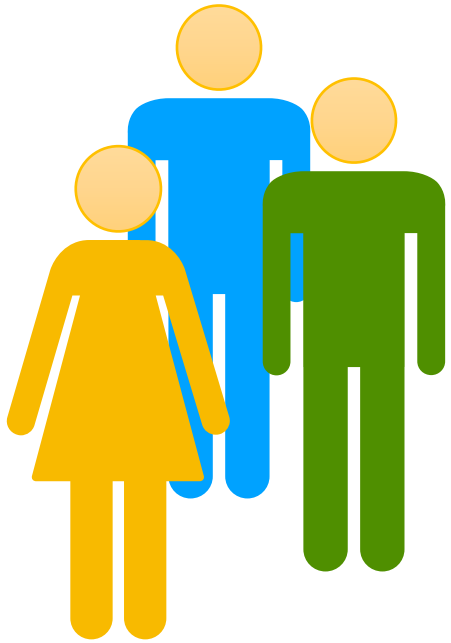
Phase 4 - Measures

- **Toxicity** Index
  - Delphi Toxicity Index
- **CC**
  - Cyclomatic Complexity
  - Number of cycles in the method
- **DOIH**
  - Depth Of Inheritance Hierarchy
- **MNOL**
  - Maximum Number Of Levels
  - Depth of if, for and while branches in the method
- **NOI**
  - Number of Interfaces

Cyclomatic Complexity
Number of possible paths through an algorithm by counting the number of distinct regions on a flowgraph, meaning the number of if, for, and while statements.

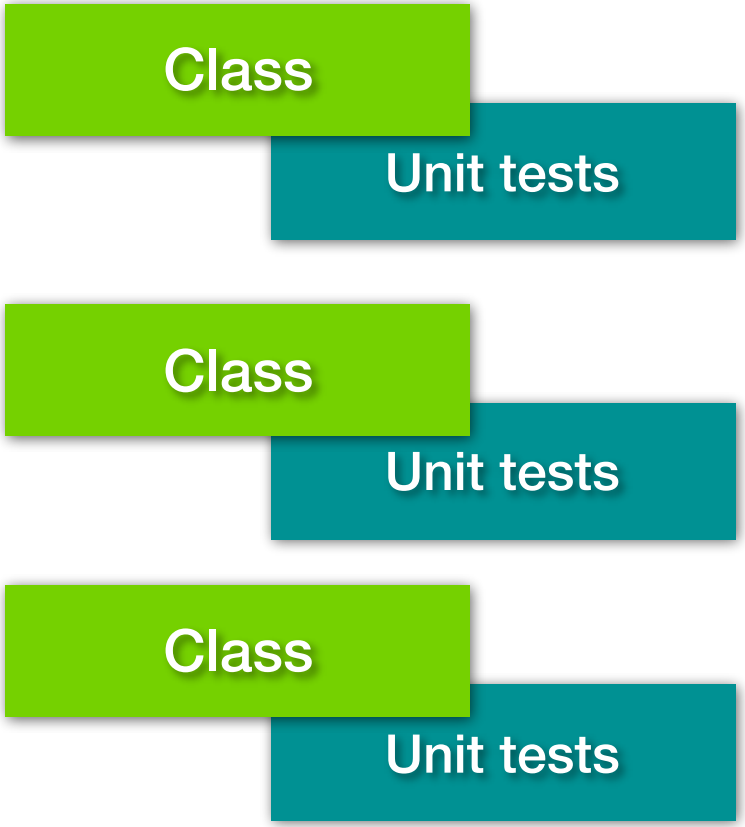
McCabe 1976

Phase 4 - Learning



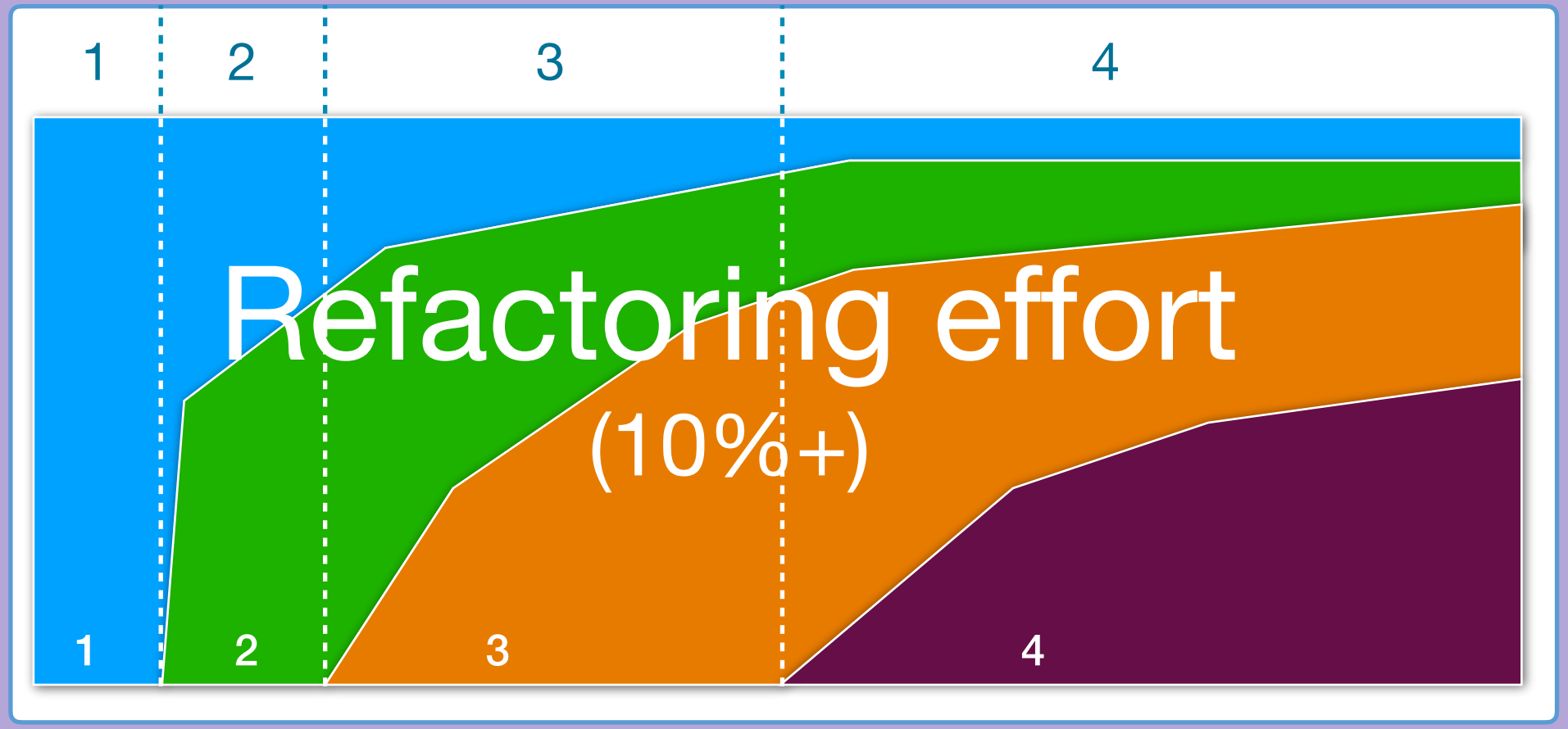
Unit Testing

- Safety Net
  - Cover classes with unit tests
- Integration tests
  - DUnitX can help
  - SOA tests
- Acceptance tests
  - Record / Playback



Summary

Understand 4 Phases



1 Clean-ups

2 Form's Events Extraction

3 Unit Decomposition

4 Architecture Improvement

Executive Level Sponsorship

- Strong C-level commitment
- Funding: 10%
- POC = Proof of Concept
- ROI - Clear and defined
- Defined milestones

Image: Business vector created by macrovector (https://www.freepik.com/free-photos-vectors/business)

Key elements

- Action Plan
- Visibility
- Learn


That's all Folks!