Grand Valley State University
www.gvsu.edu

Software Engineering Review / Anti-Patterns

Erik Fredericks // frederer@gvsu.edu

Adapted from materials provided by Byron DeVries, Jagadeesh Nandigam

Overview

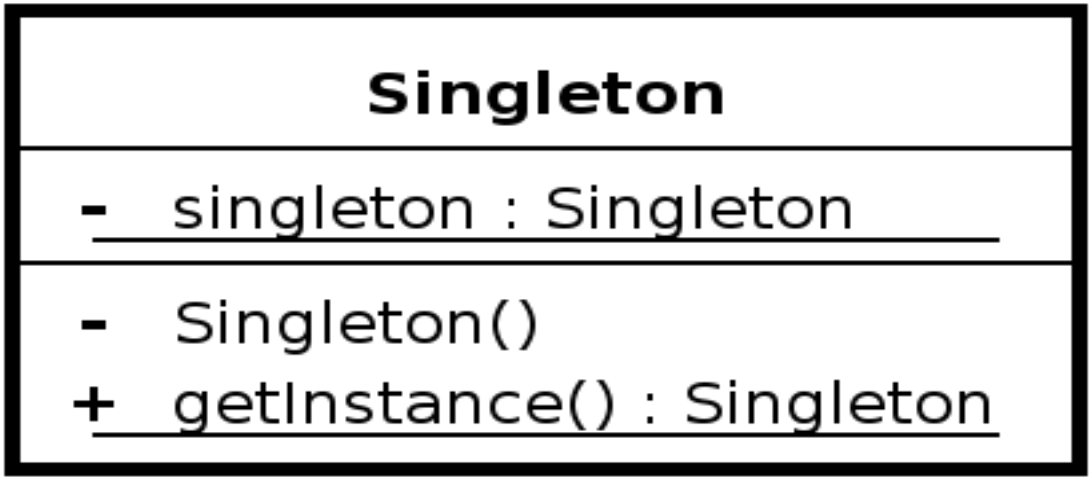
Review

Anti-patterns

Couple o' examples

Review: Singleton Pattern

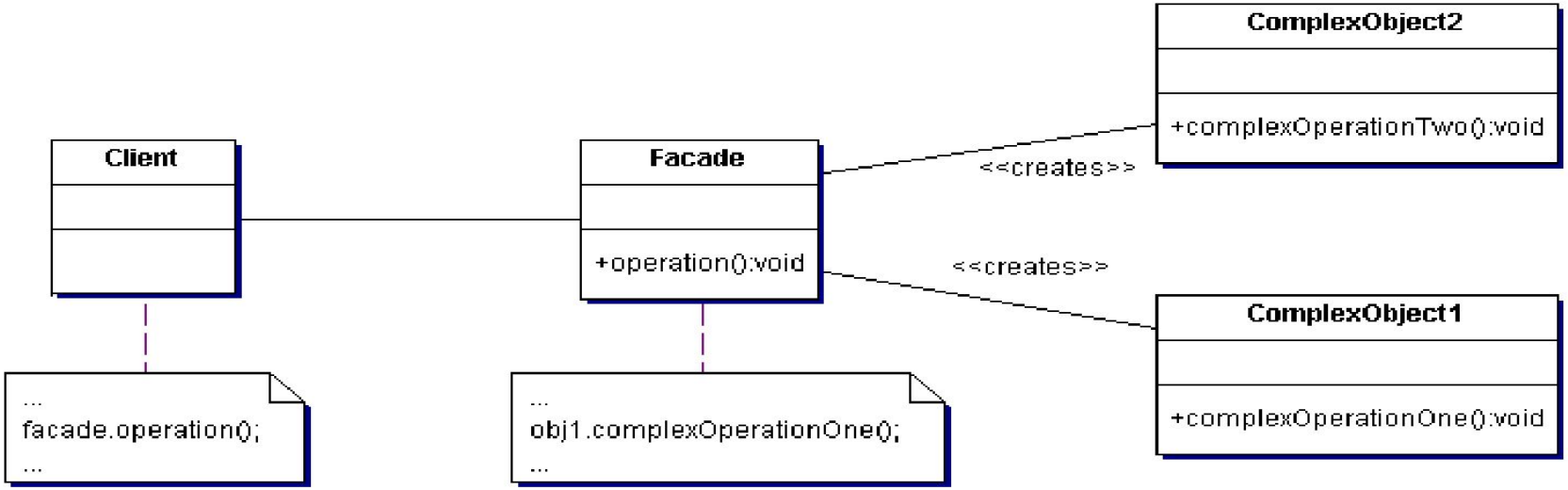
What is a Singleton Pattern?



When would you use it?

Review: Facade Pattern

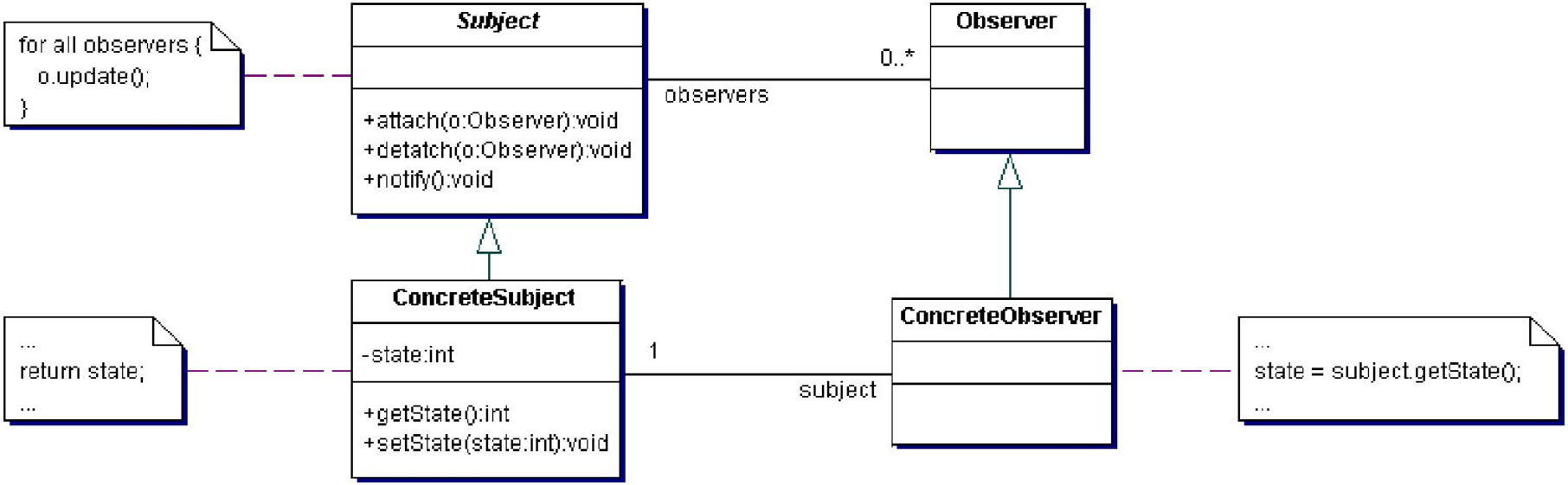
What is a Facade Pattern?



When would you use it?

Review: Observer Pattern

What is an Observer Pattern?

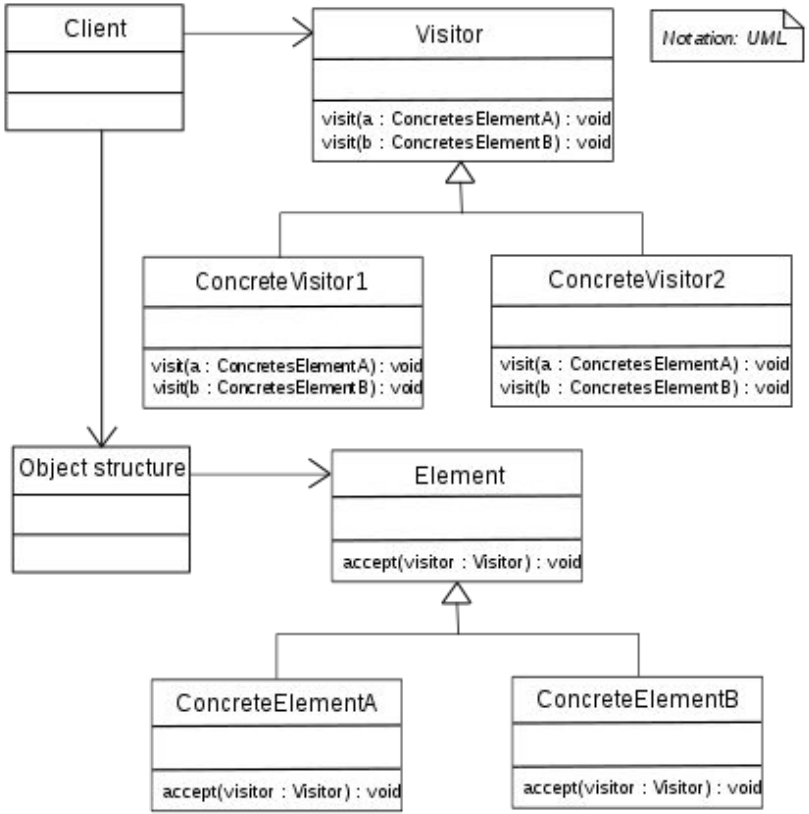


When would you use it?

Can you use it with MVC?

Review: Visitor Pattern

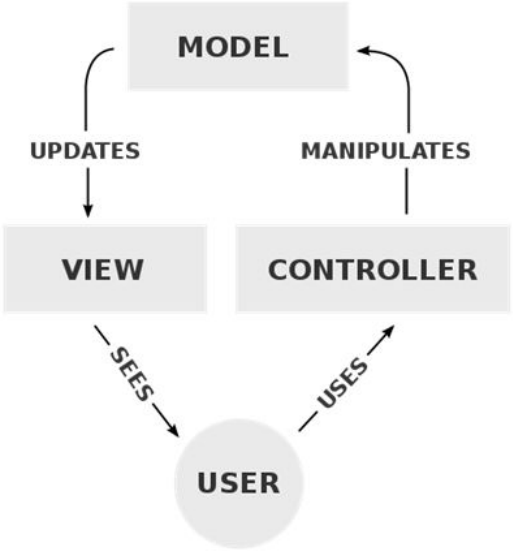
What is a Visitor Pattern?



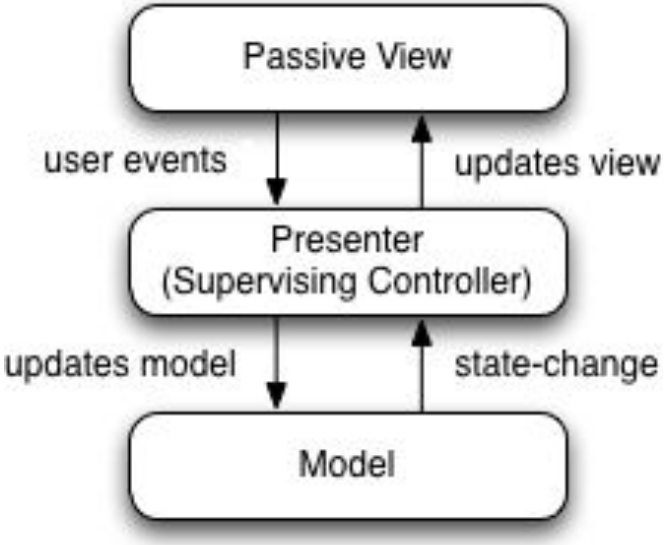
When would you use it?

Review: MVC & MVP

What is a MVC Pattern?



What is a MVP Pattern?



When would you use them?

Examples we have covered:

- Video games
- Storefronts
- etc.

MVP provides the simplest UI.

➔ This allows for easier testing.

What is the justification for MVC? The UI is still complex (i.e., the View handles the presentation by aggregating information from the Model)

Review: MVC is great for the Web

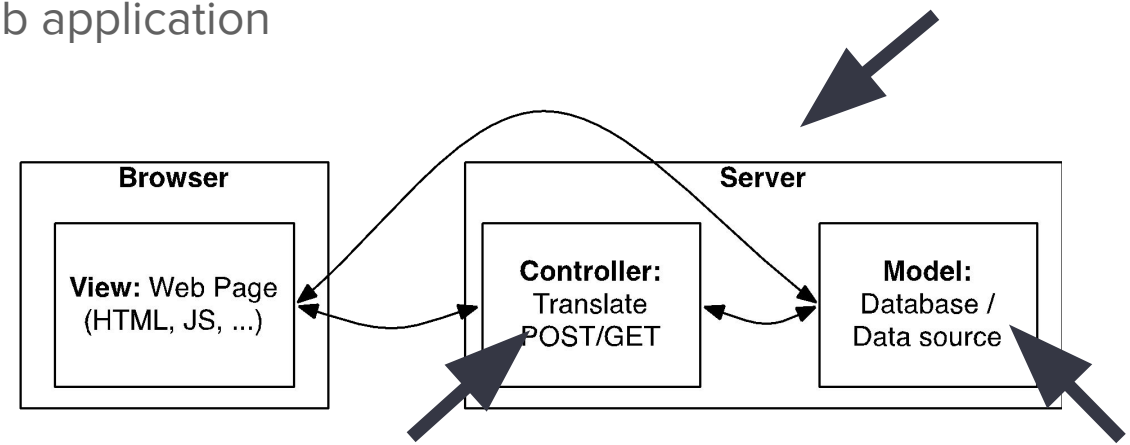
Current web platforms / libraries that use MVC:

- Django
- Ruby on Rails
- CakePHP
- CodeIgniter
- Struts
- ASP.NET
- And a ton others.

What is the model, view, and controller when applied to the web?

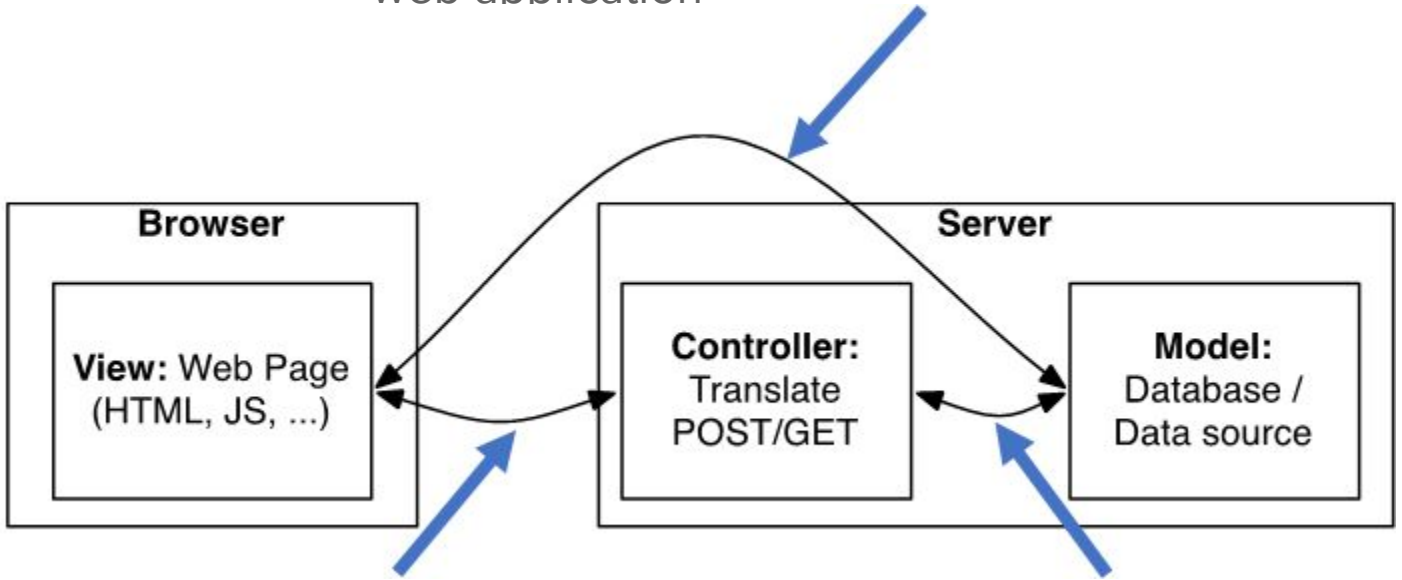
Review: MVC for the Web

- **View**: The UI (i.e., the web page as displayed to the user)
- **Controller**: Translate POST/GET commands into *changes* to the Model
- **Model**: The web source, database, and everything else that represents the web application



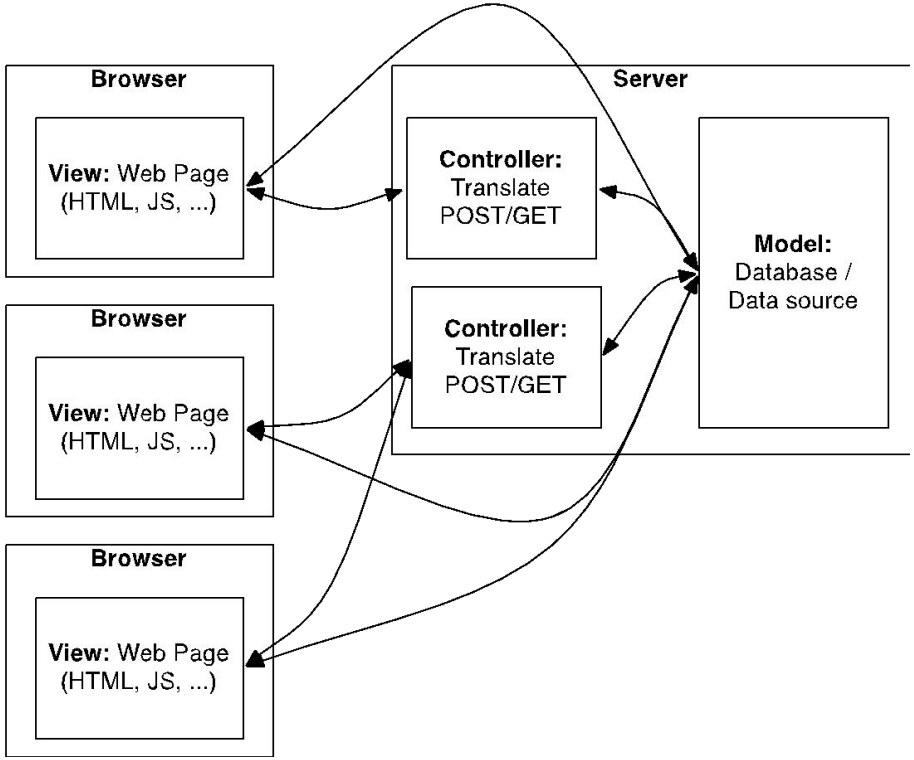
Review: MVC for the Web

View: The UI (i.e., the web page as displayed to the user)
Controller: Translate POST/GET commands into changes to the Model
Model: The web source, database, and everything else that represents the web application



The controller is used when there is a difference between view and model functions.

Review: MVC for the Web



Why not use MVP for the web?

ANTI-PATTERN

https://en.wikipedia.org/wiki/Anti-pattern
➜ https://wiki.c2.com/?AntiPatternsCatalog

(there are a *lot)*

https://exceptionnotfound.net/god-objects-the-daily-software-anti-pattern/

GOD OBJECT

The object … *knows too much*

- Or does too much
- Basically it should be broken apart

Does not follow the *single responsibility principle*

- A class should be focused and have a single purpose

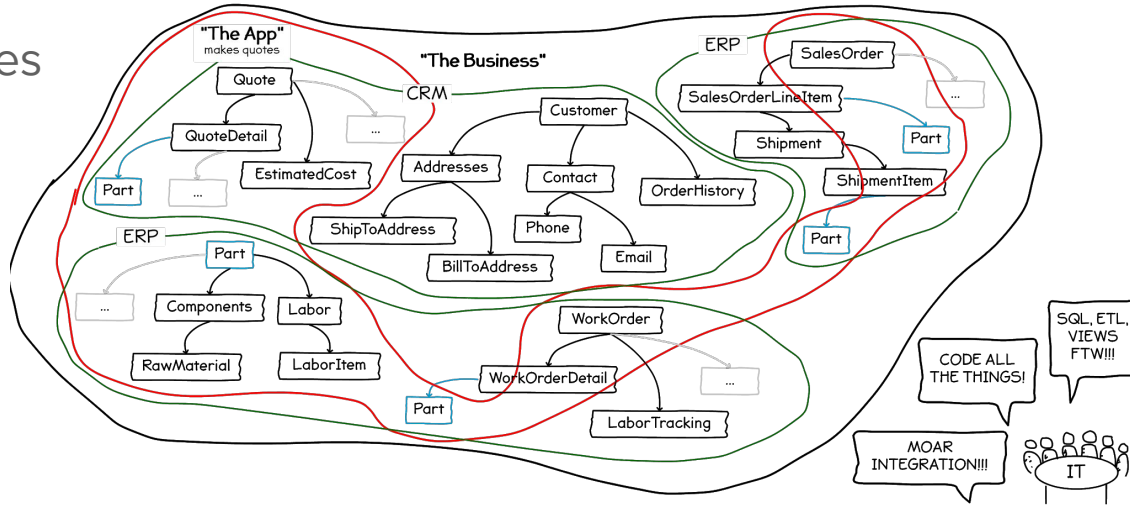
```
public static class EmployeeManager
{
    public static void HireEmployee(Employee employee) {...}
    public static void TerminateEmployee(int employeeID) {...}
    public static void EditEmployee(Employee employee) {...}
    public static void AddVacationTime(int employeeID, DateTime vacationDate) {...}
    public static void CancelVacationTime(int employeeID, DateTime vacationDate)
{...}
    public static void AddAddress(int employeeID, Address address) {...}
    public static void RemoveAddress(int employeeID, int addressID) {...}
    public static void GiveBonus(int employeeID, decimal bonus) {...}
    public static void AssignEquipment(int employeeID, Equipment equipment) {...}
    public static void GiveRaise(int employeeID, decimal amount) {...}
    public static void DockPay(int employeeID, decimal amount) {...}
    public static void AddSchedule(int employeeID, Schedule schedule) {...}
    public static void AddPhoneNumber(int employeeID, string phoneNumber) {...}
}
```

Big ball of mud

Does not seem to have an architecture ➜ "spaghetti code"

Signs of mudballs:

- Haphazard design
- Improper data hiding principles (you get access, and you get access, and you get access!)
- Duplicate attributes/methods
- etc.



Law of the instrument

The "silver bullet" // "golden hammer" // "mystical algorithm that always works"

Abraham Maslow (1966): "I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

Basically, you get stuck in a comfort zone and always apply the same techniques

TAKE ME FOR EXAMPLE

- I like genetic algorithms
- I apply them to every search problem ever, even though they have problems...

Cargo-cult programming

"Ritual inclusion of coding practices that are not understood" ➜ % WikiPedia

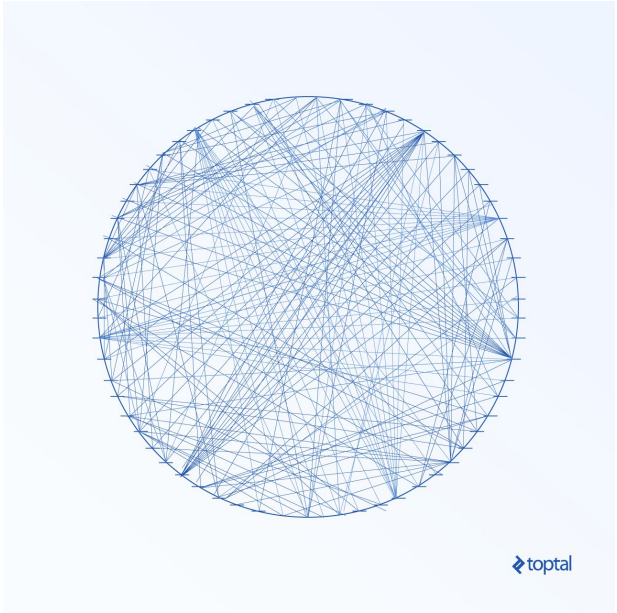
- Programmer doesn't understand bug / architecture / etc. and just slaps some tape over the problem
- Follows the magic "process" to solve all problems

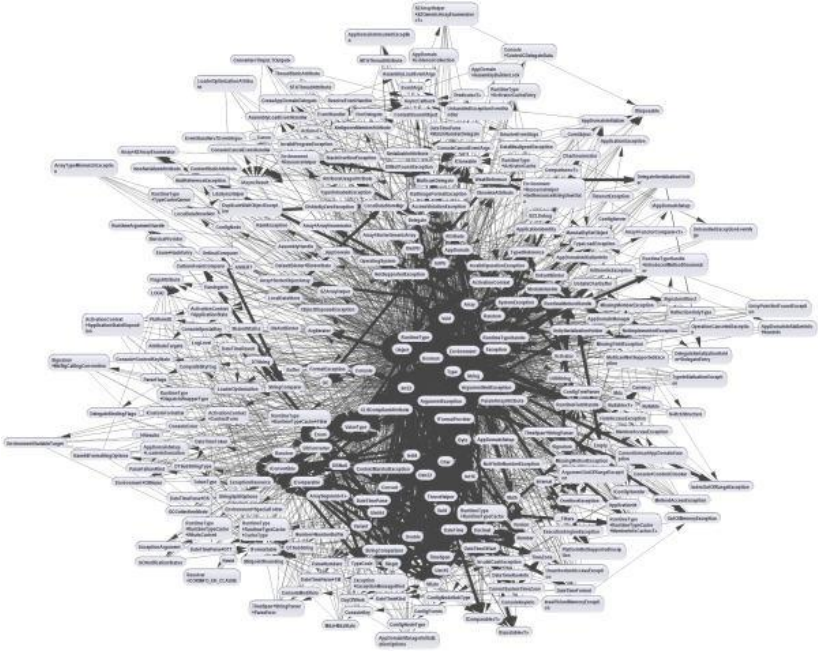
Cargo cult: aborigines built runways/mockups of runways to "summon" airplanes first noticed during WW2

Cargo cult on Tanna Island, Vanuatu

Dependency hell

Oh the fun to be had





Let's chat homework!