Enjoy the ShOW

# Application Frameworks

Function Oriented – Object Oriented – Service Oriented

By Prabhu Shanmugam

Software developers need to have a good memory, be very good at learning, and be great at forgetting!

Sometimes you have to unlearn to learn!


**Full:**
- ❑Typically don't attend
- ❑Overflow

**Half:**
- ❑Previous Knowledge
- ❑Get more out of this presentation
- ❑Confused

**Empty:**
- ❑Very Good Audience

# Looks familiar?

1

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.      HELLOWORLD.
000300 DATE-WRITTEN.    02/05/96        21:04.
000400*       AUTHOR    BRIAN COLLINS
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "HELLO, WORLD." LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

2

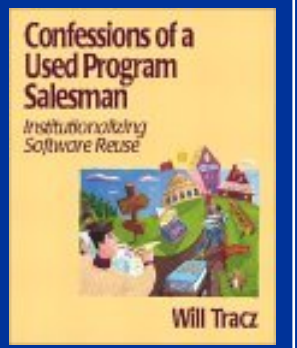
```
#include <stdio.h>
main()
{
  for(;;)
        {
                printf ("Hello World!\n");
        }
}
```

3

```
public class HelloWorld {
Public static void main (String args[]) {
System.out.println(“ Hello World “);
}
}
```

## Self Discovery

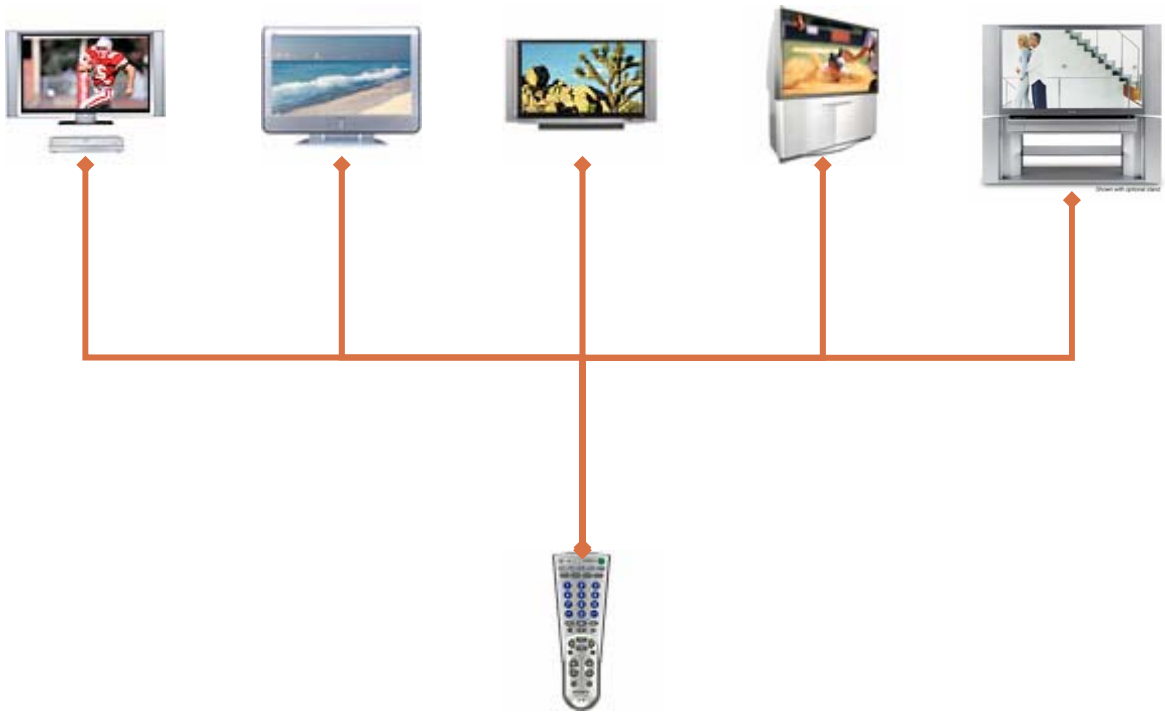
Attractive object seeks that special someone … For sharing private thoughts
…Walks on the beach …Subclasses and pets OK …NO STATICS!! please …




Confessions of a Used Program Salesman: Institutionalizing Software Reuse - Will Tracz

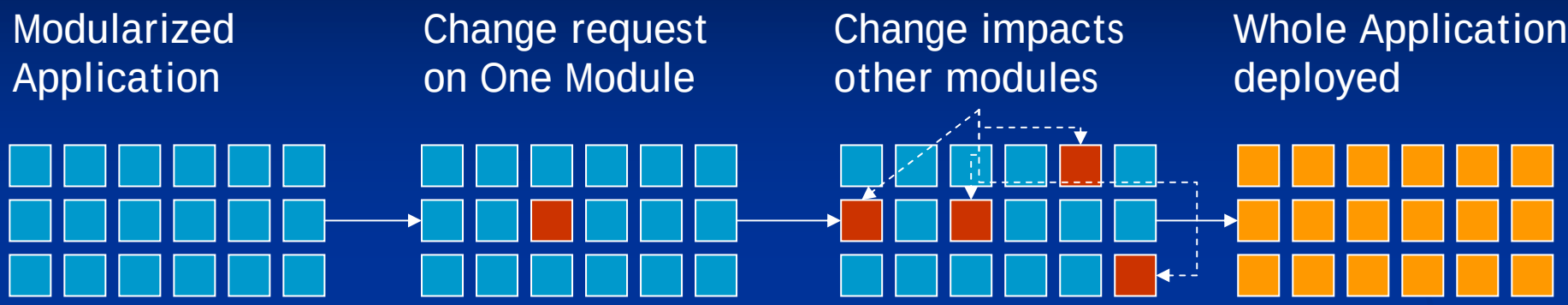
Most of the insight is gained from the second oldest programming profession – that is "second-hand software"

Sony Toshiba Philips Samsung

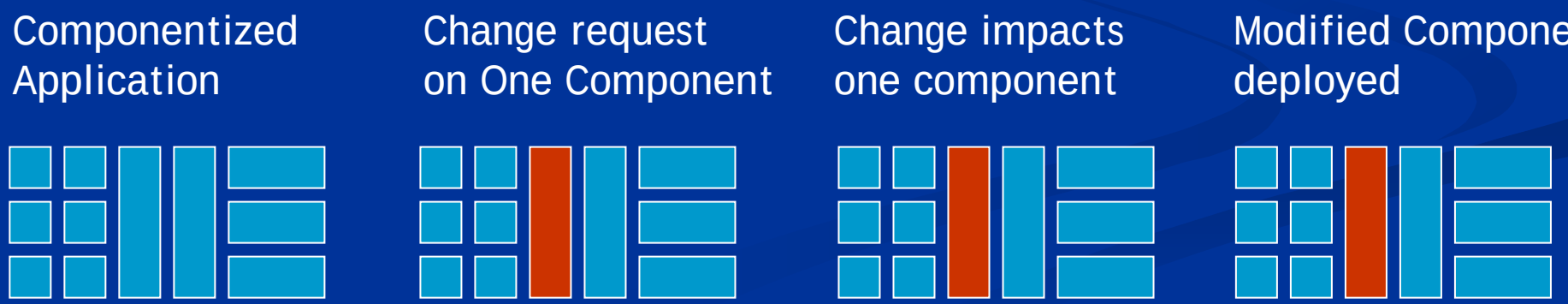


Universal Remote

# Module/function based application




# Component based application



# County Tree Management Software

We are going to use a simple scenario to simulate software that can manage trees in county.

Following assumptions are made:

- ❑ Software engineers assigned to this project only know structured or procedural programming languages. [1]
- ❑ Developers know about Structures [2]
- ❑ Procedures or functions are black boxes [3]

[1] like C, COBOL, VB etc.
[2] is defined as user-defined data type made up using primitive or language defined data types.
[3] Kind of like a black box where inputs go in and outputs come out. Data is placed into separate structures and is manipulated by these functions/procedures.

# Pseudo code concept of Tree Management Software

```
Structure Tree {
   int height
   double width
   char[15] name
}
```

```
Tree a;
```

```
Structure MedicinalTree {
   int height
   double width
   char[15] name
   char[20] species
}
```

```
MedicinalTree b;
```

```
Function CutTree (Input-Structure-Type Tree, int ReduceBy) {
   Tree.height = Tree.height - ReduceBy
}
```

```
Tree.height = 0
```

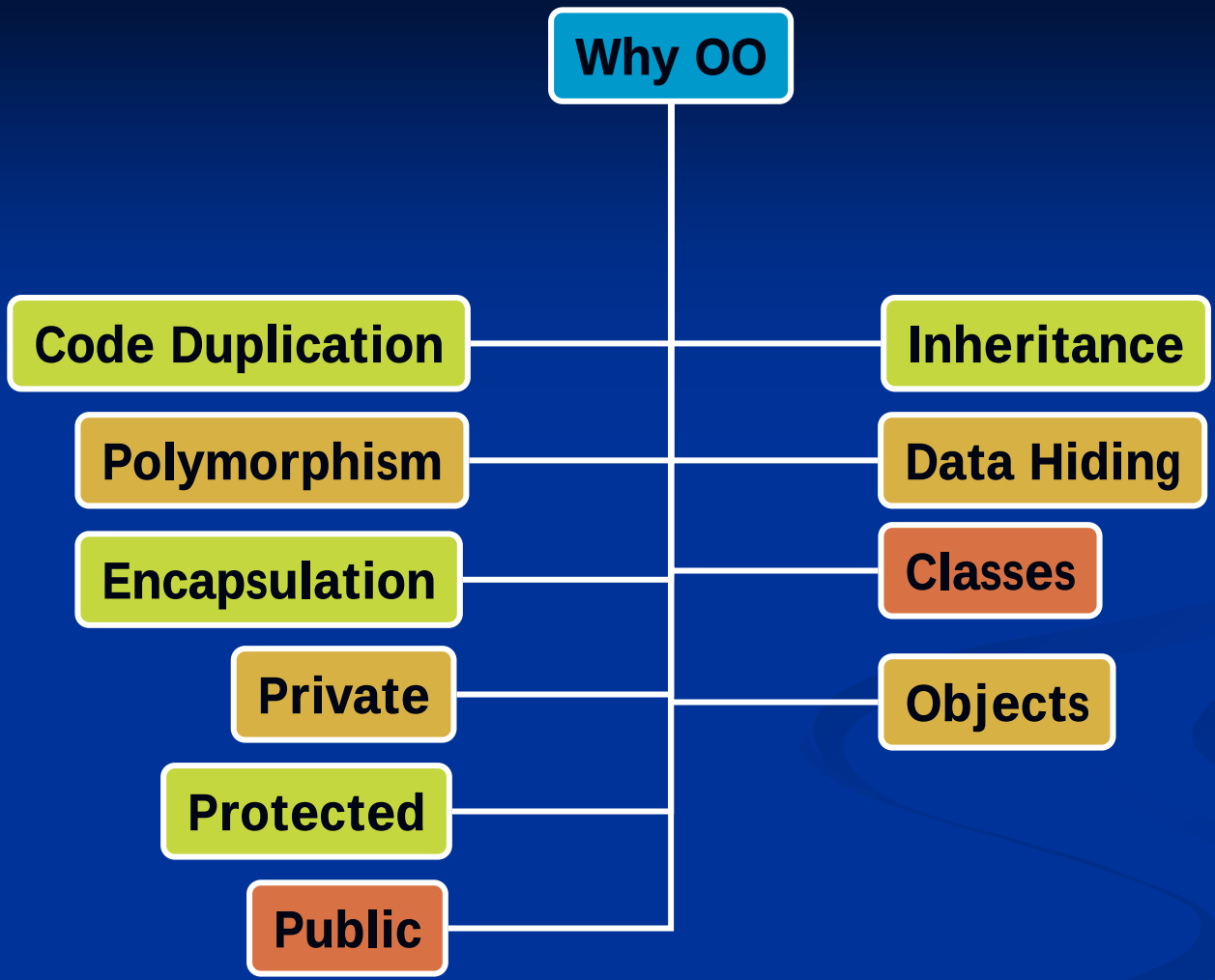
# Tree Management Software using OO

```
Class Tree {
   private int height;
   double width;
   String name;
   public void cutTree(int ReduceBy) {
      height = height - ReduceBy;
   }
}
```

```
Tree a = new Tree();
a.cutTree(10);
a.height = 0 // ERROR!
```

```
Class MedicinalTree extends Tree {
   String name;
}
```

```
MedicinalTree b = new MedicinalTree();
b.cutTree(10);
```


Why OO
Code Duplication
Polymorphism
Encapsulation
Private
Protected
Public
Inheritance
Data Hiding
Classes
Objects

**Encapsulation**

Is a concept that relates to how classes are defined. It states that a class should be self-contained, meaning that it should declare all of the fields and methods to do whatever it has to do.

**Inheritance**

Allows you to define a class that extends the capabilities of another class.

**Polymorphism**

Is described as "One interface, many implementations".

**Private**

Access not available outside object

**Protected**

Access within the same package family, but private to outside package

**Public**

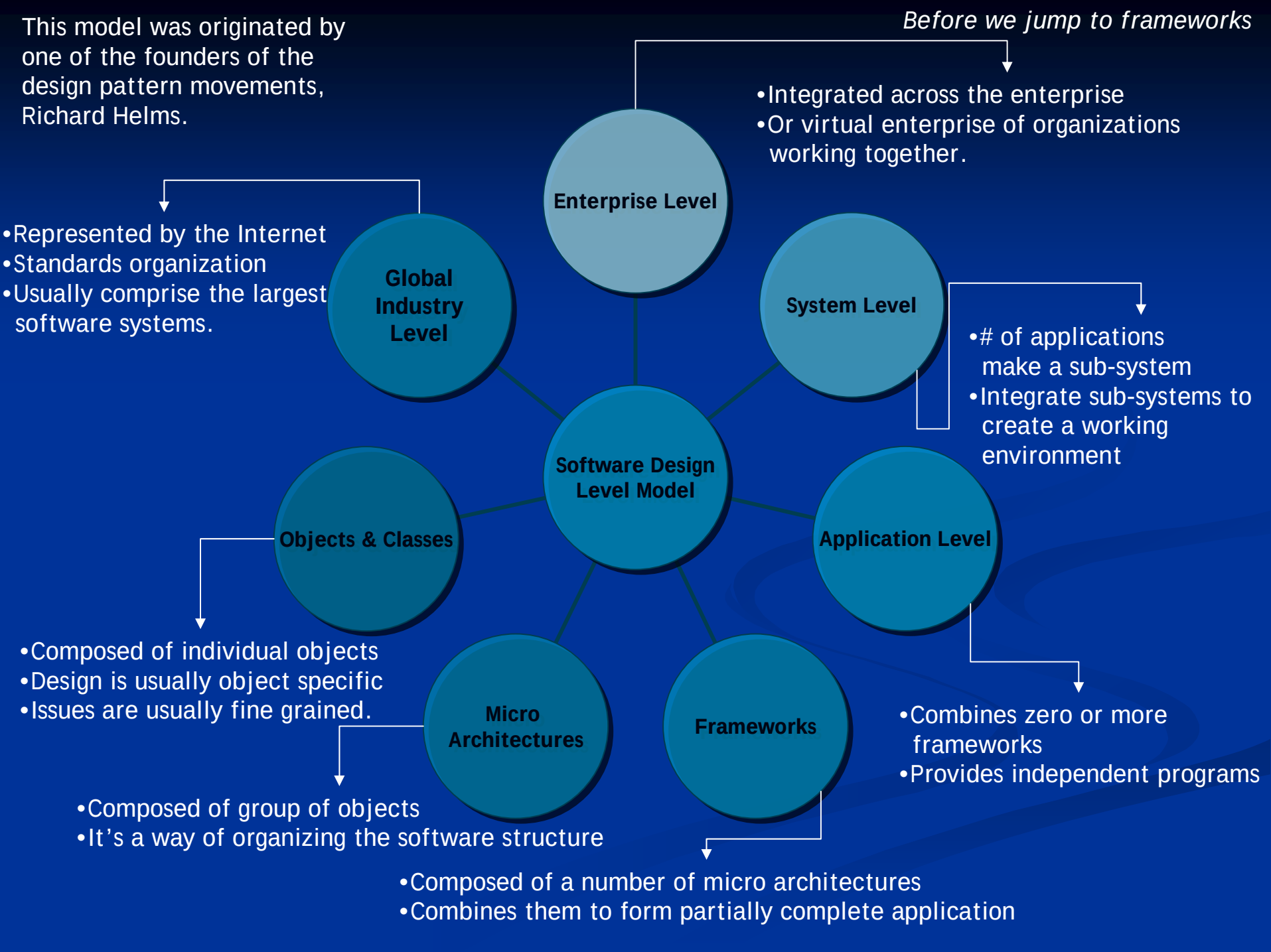
Access to all objects

**Static**

Class access; NO Object required!

This model was originated by one of the founders of the design pattern movements, Richard Helms.

*Before we jump to frameworks*




- Integrated across the enterprise
- Or virtual enterprise of organizations working together.

- Represented by the Internet
- Standards organization
- Usually comprise the largest software systems.

- # of applications make a sub-system
- Integrate sub-systems to create a working environment

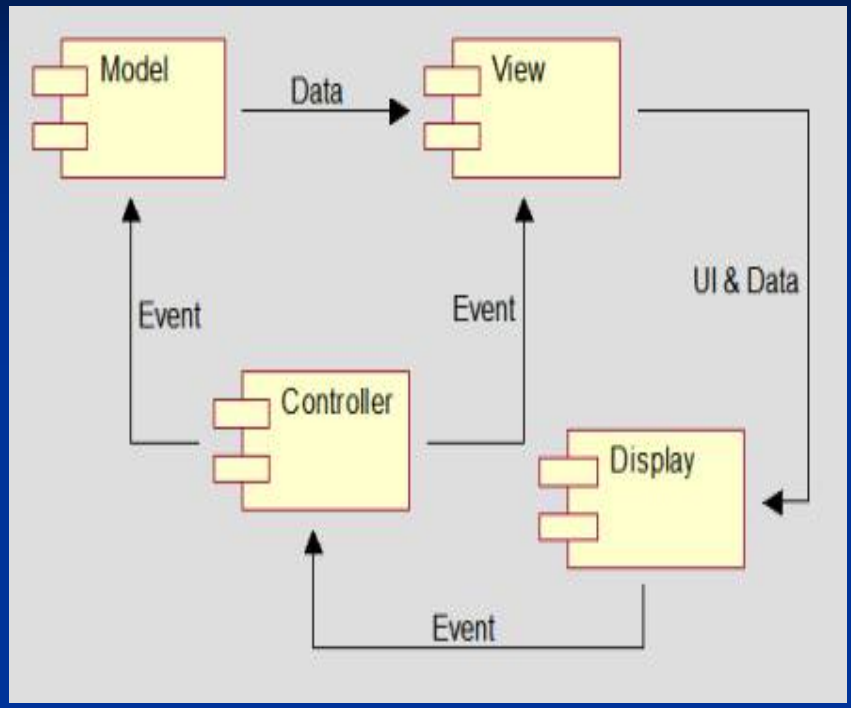
- Composed of individual objects
- Design is usually object specific
- Issues are usually fine grained.

- Combines zero or more frameworks
- Provides independent programs

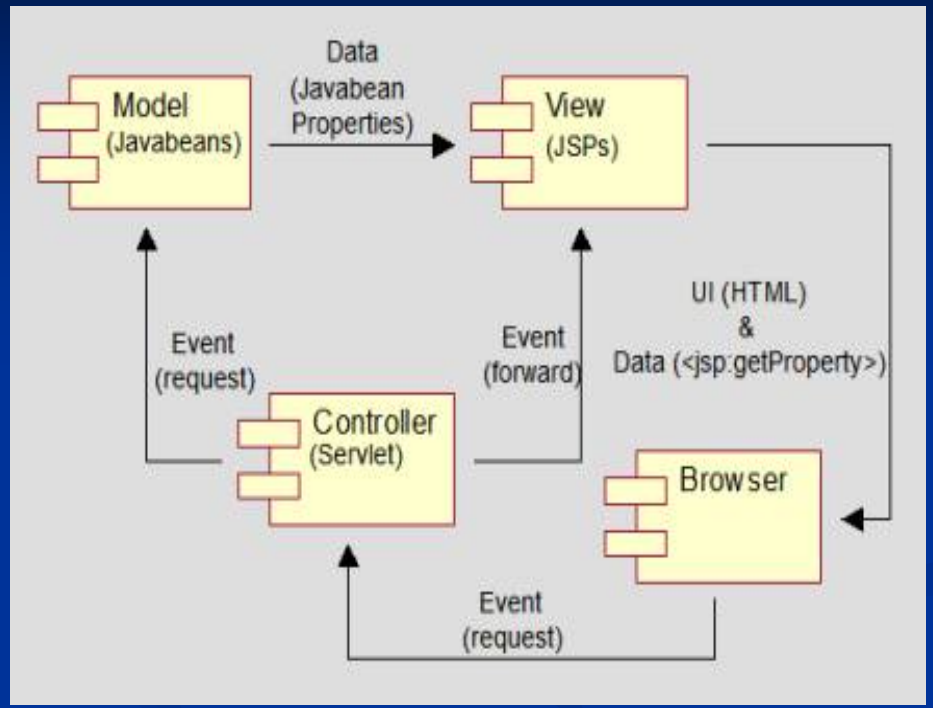
- Composed of group of objects
- It's a way of organizing the software structure

- Composed of a number of micro architectures
- Combines them to form partially complete application

# MVC and Components of MVC

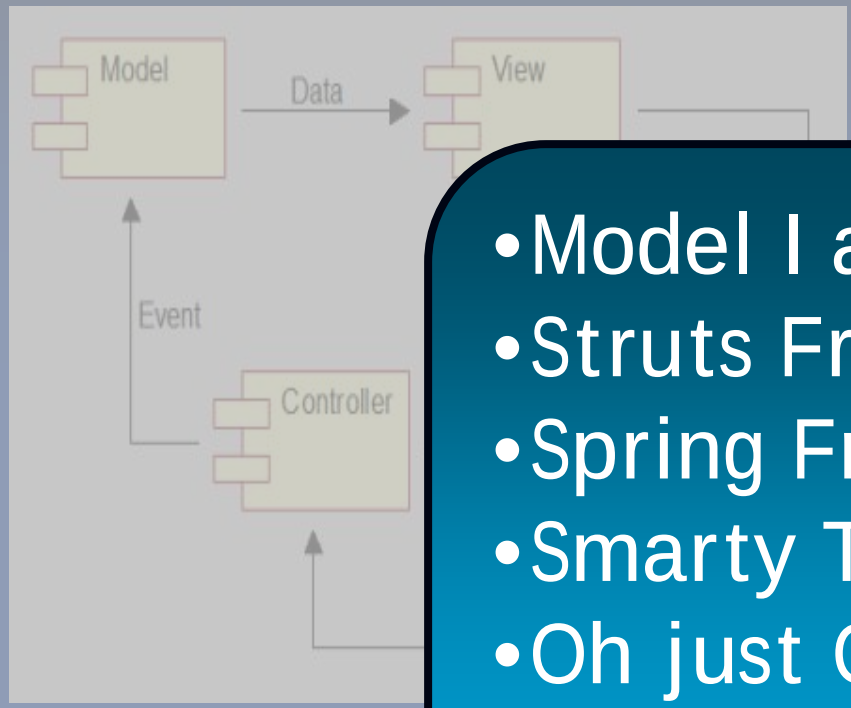


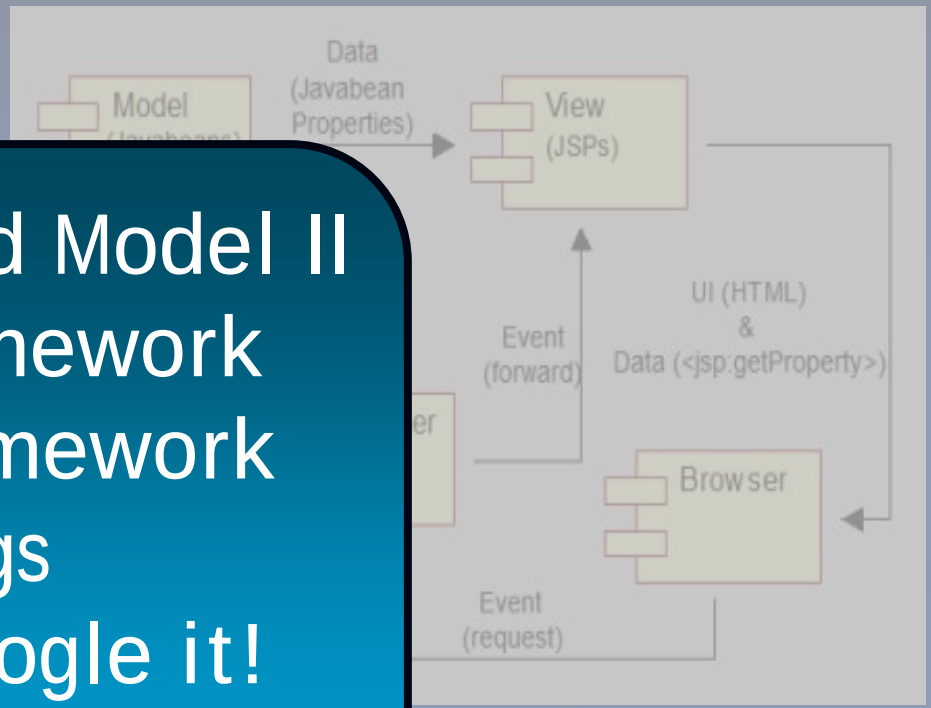

A simple interaction diagram of MVC is shown below. The Model holds all the data, the View paints UI and retrieves data to generate dynamic display, and the Controller is responsible for logical processing and delegation to Model and View.




Picture above depicts the mapping between generic MVC design paradigm and our implementation technologies. We use Javabeans for Model, JSP's for View and Servlet for Controller. Next section of the document gives more information about Model View and Controller. .

# MVC and Components of MVC




- Model I and Model II
- Struts Framework
- Spring Framework
- Smarty Tags
- Oh just Google it!

A simple interaction diagram of MVC is shown below. The Model holds all the data, the View paints UI and retrieves data to generate dynamic display, and the Controller is responsible for logical processing and delegation to Model and View.

Picture above depicts the mapping between generic MVC design paradigm and our implementation technologies. We use Javabeans for Model, JSP's for View and Servlet for Controller. Next section of the document gives more information about Model View and Controller. .

# Service Oriented Architecture

- Expresses a perspective of software architecture that defines the use of services to support the requirements of software users.
- SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer.
- The idea of SOA sometimes departs from that of object oriented programming, which strongly suggests that you should bind data and its processing together.
- Your architecture is influenced by the Industry Level software design model if you want inter-operability.

Oh my God!! “UDDI” “WSDL” “Service Orchestration” “ESB” “Canonical” “SOAP” “REST”

# Interface Oriented Programming

```
public interface Remote {
Play();
changeChannel(int number);
Forward();
Reverse();
}
```

Define Standard or Contract

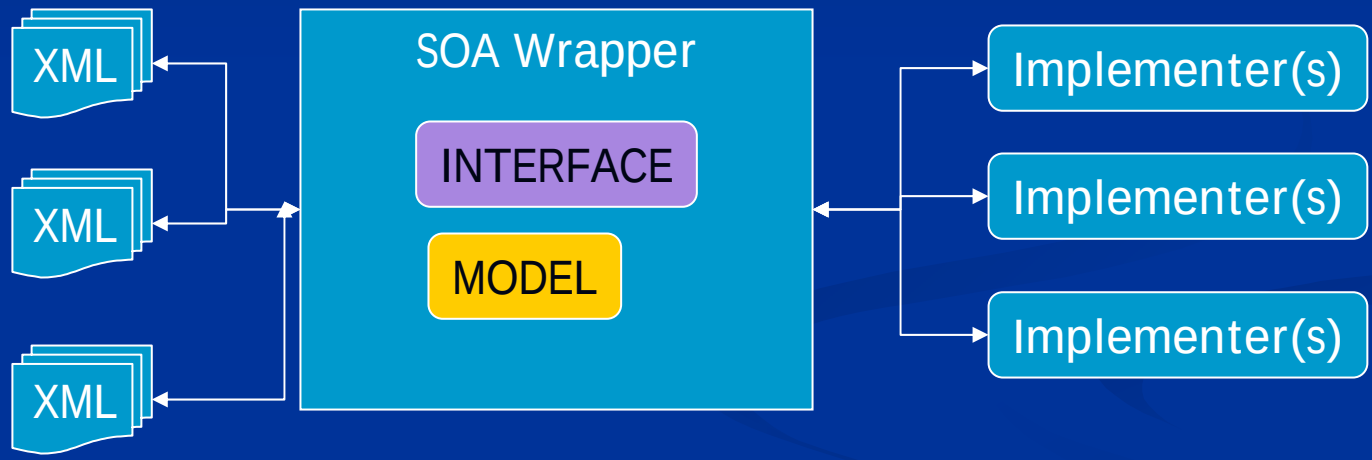
```
public class Sony implements Remote {
Play() { … }
changeChannel(int number) { … }
Forward() { … }
Reverse() { … }
}
```

Implement Standard or Contract

```
public class TV {
Remote getRemote() {
// Read configuration.
// new Sony();
}
}
```

Initial Setup

# IOP as Enabler for SOA



**You cannot change your destination overnight, but you can change your direction overnight!**

Thank You!