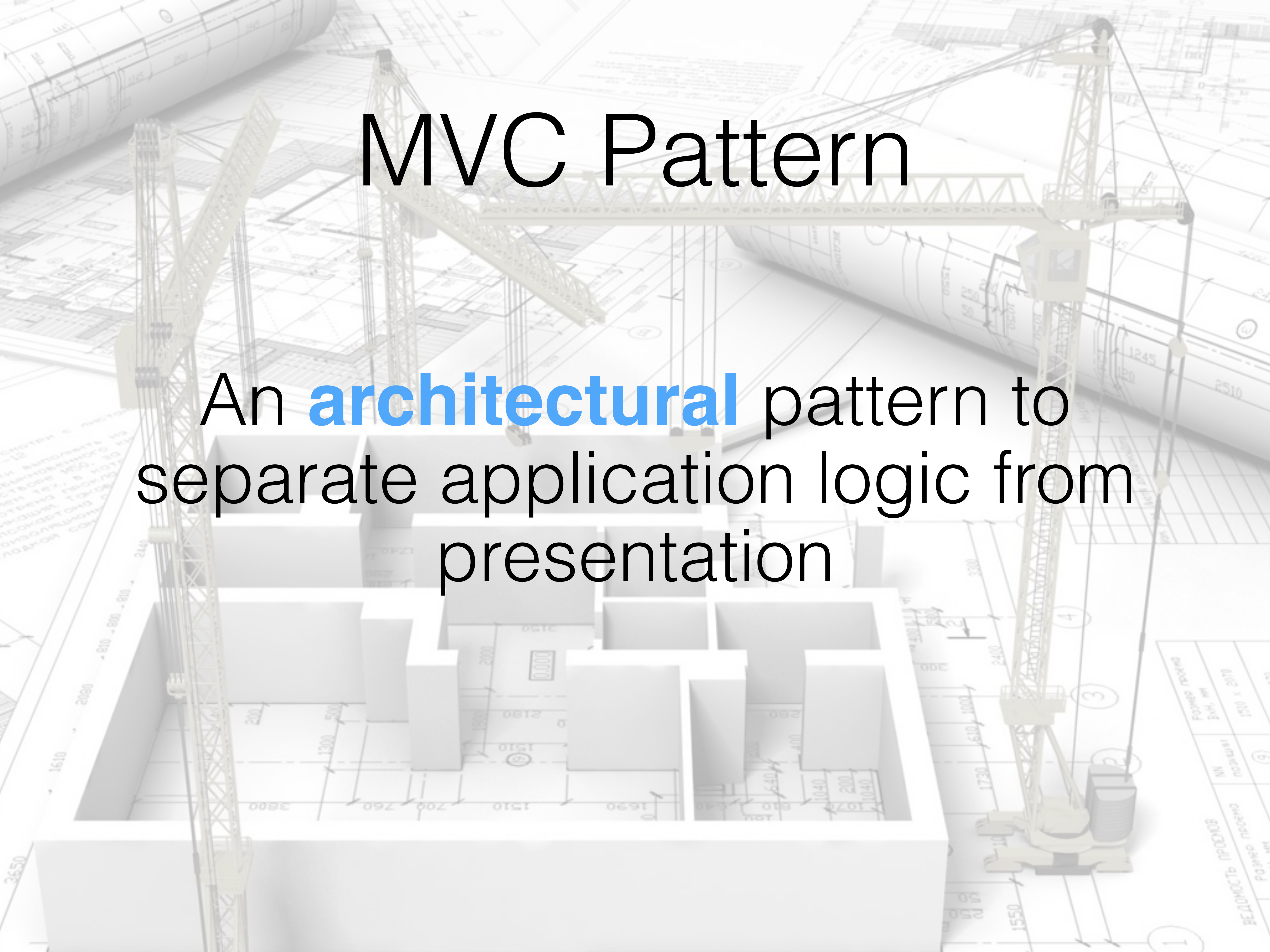
MVC

Model-View-Controller



MVC Pattern

An **architectural** pattern to separate application logic from presentation

Why MVC?

- Application structure, maintainability, and scalability for medium to large projects
- What is large?
  - Team size
  - Number of lines of code
  - Complexity

```
<?php
   $con = mysqli_connect('itp.usc.edu', 'user', 'pw', 'mydb');
   $results = mysqli_query($con, 'SELECT * FROM genres');

   $session = session_start();

   if (!$_SESSION['logged_in']) {
      header('Location: login.php');
   }
?>
<!doctype html>
<html>
<head>
   <title>DVD Site</title>
</head>
<body>

<select name="genre">
   <?php while ($row = mysqli_fetch_array($results)) : ?> {
      <option><?php echo $row['genre']; ?></option>
   <?php endwhile; ?>
</select>

</body>
</html>
```

Models (**M**VC)

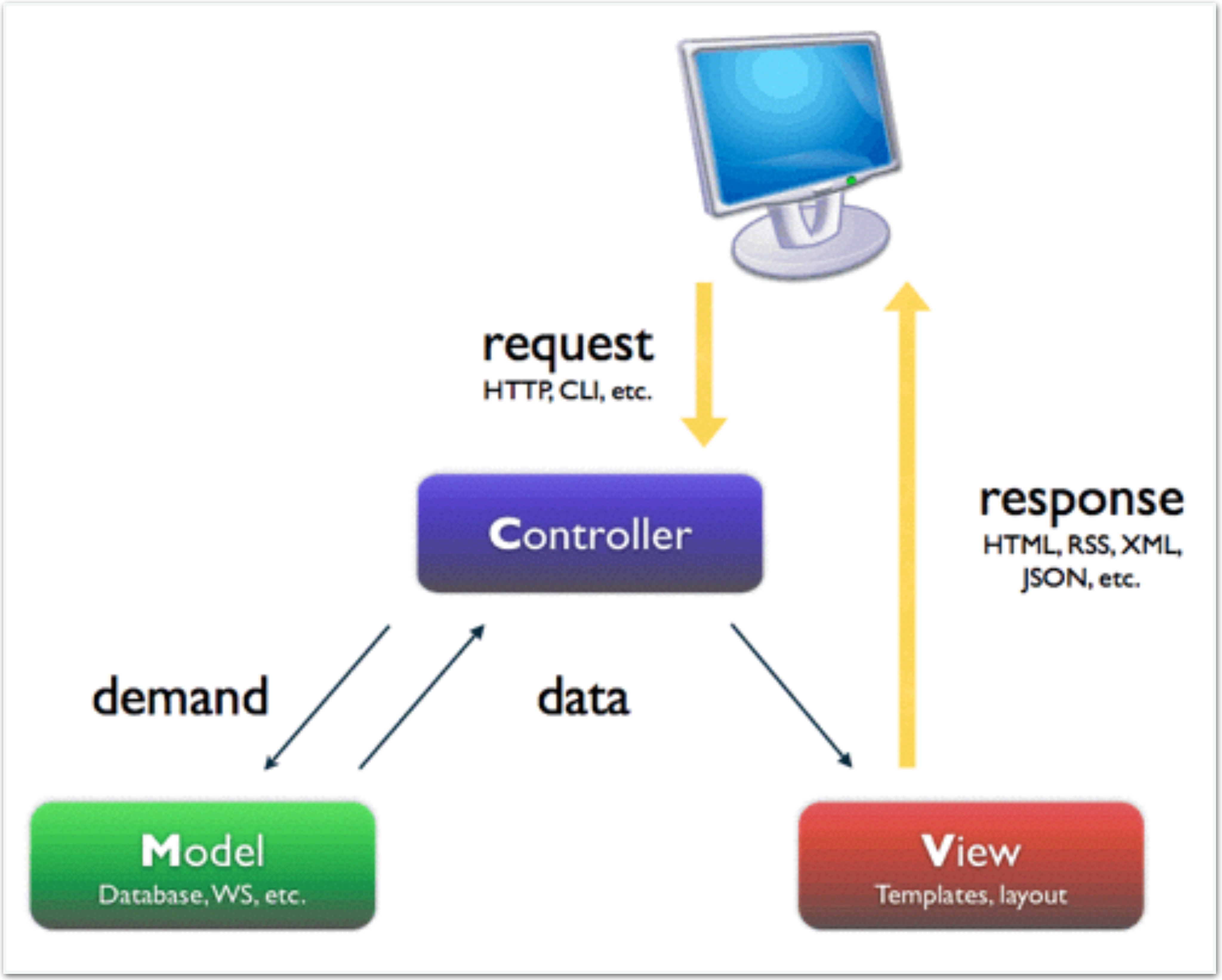
Models contain data in a business domain and the associated rules and operations for retrieving, transforming, and storing that data.

Views (MVC)

- Views correspond to the presentation and user interfaces
- Views are handed any data and display that data to the user
- Typically HTML (could also be XML, RSS, etc)

Controllers (MVC)

- Controllers respond to HTTP requests
- Controllers serve as an intermediary between Models and Views
- Models and Views don't know about each other


request
HTTP, CLI, etc.
Controller
response
HTML, RSS, XML, JSON, etc.
demand
data
Model
Database, WS, etc.
View
Templates, layout

MVC Frameworks

- PHP: Laravel, Symfony, CodeIgniter, Zend, CakePHP
- Ruby: Ruby on Rails
- Java: Struts, Spring MVC, Play
- Python: Django