Declarative programming

Agenda

Software Development Methodologies

Imperative and declarative programming

Front-end web application frameworks (AngularJS)

Declarative programming in AngularJS

Software Development Methodologies

Waterfall

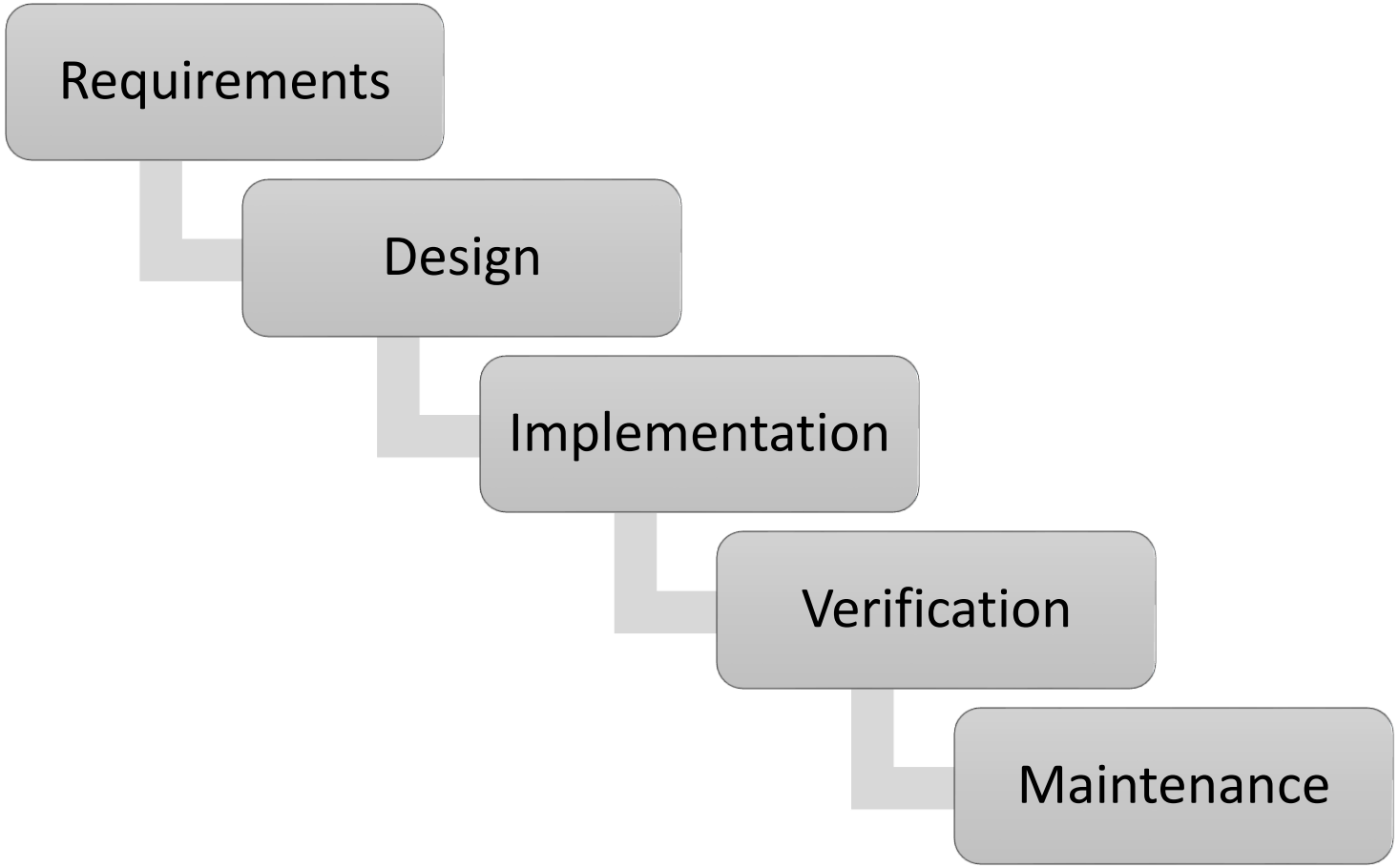
Prototyping

Rapid Application Development

Extreme Programming

…

Waterfall model



Rapid Application Development

Prototyping

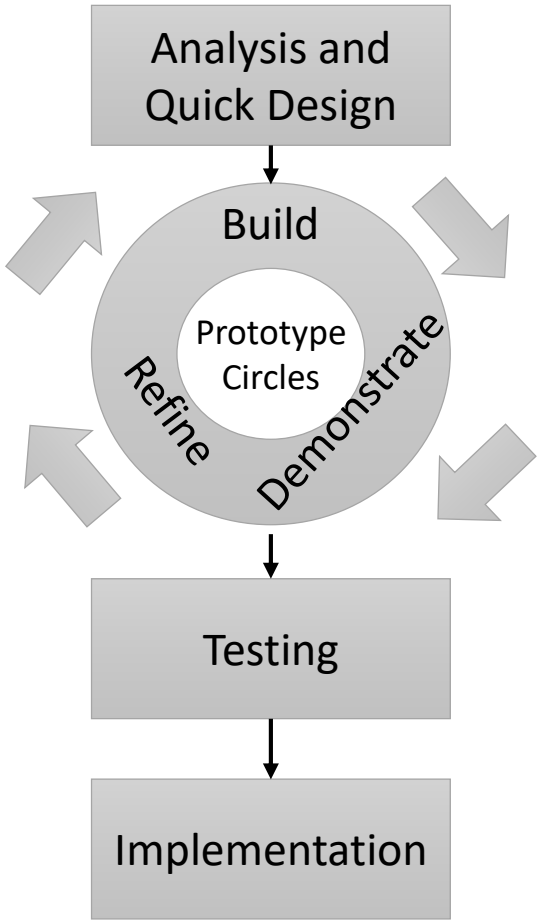
Demonstrable result as early as possible

Iteration

Incremental development based on refinement

Time-boxing

Attention on delivery



Imperative vs declarative programming

How vs What

AngularJS introduction

Front-end web application framework

Declarative programming (app user interface)

Imperative programming (app business logic)

Two-way data-binding (automatic model and view synchronization)

MVC pattern implementation

AngularJS Tutorial

AngularJS Tutorial W3Schools

Learn AngularJS on Codecademy

Declarative programming (Directives)

ng-app	auto-bootstrap an AngularJS application
ng-model	binds form controls (input, select, …) with model
ng-bind	replace HTML text content with the model values
	or {{ expressions }}

Directive components in ng

Static HTML document

```
<!DOCTYPE html>
<html>
<body>

  <div>
    <p>Name: <input type="text"></p>
    <p></p>
    <p>Hello</p>
  </div>

</body>
</html>
```

Angular View

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>

  <div ng-app="">
    <p>Name: <input type="text" ng-model="name"></p>
    <p ng-bind="name"></p>
    <p>Hello {{name}}</p>
  </div>

</body>
</html>
```

Angular Model and Controller

```
<div ng-app="myApp" ng-controller="myCtrl">

      <p>Name: <input type="text" ng-model="name"></p>
      <p>Hi {{name}}</p>

</div>



<script>
      var app = angular.module('myApp', []).
      app.controller('myCtrl', function($scope) {
            $scope.name = "George";
      });
</script>
```

Iterating through model values

```
<body ng-app="myApp" ng-controller="myCtrl">

  <table>
    <tr ng-repeat="day in daysOfWeek">
      <td>{{day}}</td>
    </tr>
  </table>

  <script>
    var app = angular.module("myApp", []);
    app.controller("myCtrl", function($scope) {
          $scope.daysOfWeek = ["Monday", "Tuesday", … ]
    });
  </script>

</body>
```

Angular Services

Function or object available to application

Built-in services

Services created by user

[Service components in ng]

$http service (remote communication)

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.htm").then(function (response) {
        $scope.daysOfWeek = response.data;
    });
});
```

To do

Calculate triangle area

1. The square.html program calculates the area of a square.
2. Analyse the Angular directives contained in the document. Then run the program.
3. Make a copy of this file with the name triangle.html.
4. Modify the triangle.html program to calculate the area of a triangle.

Use the constructor

1. The oscars.json contains a list of winners in three categories: best picture, best actor, and best actress.
2. Create a program to display the list as an html table.
3. Add an Angular constructor and copy and paste the content on the json file. Assign the data to the $scope.
4. To display the data in an html table, use the Angular ’ng-repeat’ directive.
5. Improve the quality of the html table by using the bootstrap framework.

Display data conditionally

1. The pictures.json document contains a set of URLs of Cracow University of Economics pictures.
2. Create o program to display one selected picture depending on a user choice.
3. Copy and paste the content od the json file to your program and assign this object to the Angular $scope.
4. Next create a <select> element based on the picture ‘category’. To create the element, you can use either ’ng-options’ or ‘ng-repeat’ Angular directive.
5. Then, to display a picture depending on the value of the <select> element, use the Angular ’ng-switch’ directive.

Get data from the Internet

1. **Create a program to display current exchange rate table (table of type C: http://api.nbp.pl/en.html) as an HTML table.**
2. **To get data from the Internet, use the Angluar $http service.**
3. **Improve the quality of the html table by using the bootstrap framework.**

Store data in local SQL database

1. Create a program to make a simple ’to do’ list.
2. To store the list, create a Web SQL database with one table.
3. Add in the program a text field with an ’add’ button.
4. After clicking on the button, the field value (’to do’ item) is to be inserted into the SQL database.
5. Use the Angular ’ng-click’ directive connected with a $scope function.
6. Display the whole list below the text field and button.
7. To improve the user interface quality, use the bootstrap framework.
8. You can update your program by adding the deleting selected item from the list.

Create a simple game

1. Create a simple game. Use the Angular JS.
2. The biggest filed displays a set of digits (a digit string).
3. Every 1 second a randomly selected digit is added at the end of the digit string.
4. A player can click the ‘next’ button to change the single digit above. The digit changes from 0 to 9 and moves to 0 again.
5. If the single digit is the same as the first digit in the string, the player can click on the ’Del’ button. The first digit in the string is then removed.
6. The game ends if the number of digits in the string is greater than eight.
7. You can update the game by adding new digits at the end of the string more often (e.g. every half a sec.)