MTAT.03.295
Agile Software Development

Lecture 1: Introduction

Luciano García-Bañuelos
Course objective

› The objective of this course is to introduce some of the practices on agile software development, taking as an example the development of applications labeled as: software as a service

› Following trends in the software industry, we target highly interactive applications in addition to the more traditional client/server-type of applications
Rationale of the course

 › Strong connection with MTAT.03.229 - Enterprise System Integration

 › Great opportunity to introduce/recall some concepts
   – Development of web-based applications
     (Large scale software applications)
   – Introduction of modern development practices
     (e.g. continuous integration, agile planning)
   – Use of cloud-based tools
     (e.g. Bitbucket, Shippable, Heroku, Pivotal tracker)
History of the course

› Course delivered at UC at Berkeley
  – Agile Web Development Using Ruby on Rails
  – Delivered by Armando Fox and David Patterson

› Also offered via EdX, it is divided into two parts
  – Ruby and Ruby on Rails, agile methodology with emphasis on BDD/TDD cycle
  – Advanced Rails, working with legacy code, working in teams and Javascript

Agile Software Development - Introduction
Agile methodology

- Talk to “Customer”
- Legacy
- User stories (BDD)
- Design patterns
- Unit test (TDD)
- Velocity measurement
- Deployment (Cloud)
Agile methodology

Talk to “Customer”

User stories (BDD)

Unit test (TDD)

Velocity measurement

Deployment (Cloud)

Javascipt

Ruby

AngularJS

Ruby on Rails

Cucumber-js

Cucumber

Jasmine

RSpec
Agile methodology

Elixir     Javascript
Phoenix framework     vue.js
WhiteBread
ExUnit     Jasmine

Talk to “Customer”
User stories (BDD)
Unit test (TDD)
Velocity measurement
Deployment (Cloud)
Approach

› Take one SaaS project from conception to deployment
  – Mimic the interaction with non-technical customers
    › Scrum meetings, User stories, Acceptance testing
  – Backend: Elixir & Phoenix framework
  – Frontend: Javascript & Vue.js
  – Project tracking, behavior/test driven development, Continuous integration and Deployment to the cloud
Organization of the course

› Lectures (Tuesdays)

› Practicals (Tuesdays/Thursdays)

› Continuous assessment
  – 4 assignments (weeks 3-11)
    › Released on Tuesday; 10 days to complete
    › 2 of them are around our sample application (Taxi app)
  – 1 software project (weeks 9-16)
Grading

› Homework (submitted in pairs)  
  – 4 assignments x 5 points  

› Project (team-based, 4 members)  
  – Evidence of use of agile practices (10 points)  
  – Assessment of the delivered software (15 points)  
  – Presentation (2.5 points)  
  – Written report (2.5 points)  

› Final exam  
  – You need a mark of at least 21 points out of 50 to pass the course  

› Participation:  
  Up to 10 bonus points
Important dates

› Release and due dates for assignments

<table>
<thead>
<tr>
<th>Release date</th>
<th>Due date (20:00 EET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.09.2018</td>
<td>28.09.2018</td>
</tr>
<tr>
<td>25.09.2018</td>
<td>05.10.2018</td>
</tr>
<tr>
<td>09.10.2018</td>
<td>19.11.2018</td>
</tr>
<tr>
<td>06.11.2018</td>
<td>16.11.2018</td>
</tr>
</tbody>
</table>

› Project presentation
  – 18.12.2018

› Exam
  – 08.01.2019 and 10.01.2019
  – Resit on 22.01.2019
Teaching staff

› Lecturer & coordinator
  – Luciano García Bañuelos
  Liivi 2-308

› Lab assistant
  – Mykhailo Dorokhov
Hands on with Elixir
Elixir what?

› Elixir is a dynamic, functional programming language designed for building scalable and maintainable applications

› Elixir is a compiled language

› Elixir compiler produces bytecode that runs on top of Erlang’s virtual machine (a.k.a. BEAM)

› Features inherited from Erlang
  – Fault tolerance, distribution and low latency
Elixir what? (cont)

› Created by José Valim circa 2012

› José is a former Rails core contributor
  – His mission was to make Rails thread safe … he ended up creating a new programming language
Why Elixir?

› CPUs today have gazillions of transistors and lots of cores

› Very hard to implement applications that use this type of architectures

› In other words: We need to figure out new (?) approaches to implement our applications
  – FACT: Functional programming is well suited for this purposes

› Functional programming seems weird … not with Elixir
Value types

› Integers
  1234

› Floats
  3.1415

› Ranges
  1..10

› Regular expressions
  \~r/\d+/\n
› Atoms (aka symbols)
  :surname
  (true/false/nil are atoms)

› Strings
  "This is a cool language"
Collection maps

› Tuples
{123, false}

› Lists
[1,2,3,4]

› Maps
%{:name => "Alfonso Cuarón", :age => 47}
Factorial: Take 1

defmodule Example do
  def factorial(n) do
    if (n == 0) do
      1
    else
      n * factorial(n - 1)
    end
  end
end

> iex.bat example.ex
Interactive Elixir (1.7.2) - press Ctrl+C to exit
(type h() ENTER for help)
iex(1)> Example.factorial(6)
720
As a warm up for this week

› Install the software
  – Elixir  https://elixir-lang.org/install.html
  – Code editor  https://code.visualstudio.com/
  – Git client  https://git-scm.com/downloads

› Get acquainted with Elixir and Git
  – “Try Elixir” course @codeschool  
    https://www.codeschool.com/courses/try-elixir
  – “Try Git” course @github  
    https://try.github.io