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 methodology

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

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

- Javascript
- AngularJS
- Cucumber-js
- Jasmine

- Ruby
- Ruby on Rails

- Cucumber
- RSpec
- Cucumber-js
- js
- Javascript
- AngularJS
- Jasmine
- RSpec
- Cucumber
Agile methodology

- Elixir
- Phoenix framework
- Hound
- ExUnit
- Javascript
- vue.js
- Jasmine

Talk to “Customer”

User stories (BDD)

Unit test (TDD)

Velocity measurement

Deployment (Cloud)
Approach

› Take two SaaS projects 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
  – 5 assignments (weeks 3-11)
    › Released on Tuesday; one week to complete
    › Three of them are around our first application (Taxi app)
  – 1 software project (weeks 11-16)
Grading

› Homework (submitted in pairs) 30 points
  – 5 assignments x 6 points

› Project (team-based, 4 members) 30 points
  – 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 40 points
  – You need a mark of at least 16 points out of 40 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>19.09.2017</td>
<td>27.09.2017</td>
</tr>
<tr>
<td>26.09.2017</td>
<td>04.10.2017</td>
</tr>
<tr>
<td>17.10.2017</td>
<td>25.10.2017</td>
</tr>
<tr>
<td>24.10.2017</td>
<td>01.11.2017</td>
</tr>
<tr>
<td>07.11.2017</td>
<td>08.11.2017</td>
</tr>
</tbody>
</table>

› Project presentation
  – 19.12.2017

› Exam
  – 09.01.2018 and 11.01.2018
  – Resit on 23.01.2018
Teaching staff

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

› Lab assistants
  – Mykhailo Dorokhov
  – Orlenys López Pintado
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
› FALSE: 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+/|

› 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 (1.5.1) - 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