TOWARDS THE EFFECTIVE SOFTWARE DEVELOPMENT OF AN E-LEARNING PLATFORM FEATURING LEARNING ANALYTICS AND GAMIFICATION

David Gañán, Santi Caballé, Robert Clarisó, Jordi Conesa

Faculty of Computer Science, Multimedia, and Telecommunications
Open University of Catalonia (UOC)
Barcelona, Spain

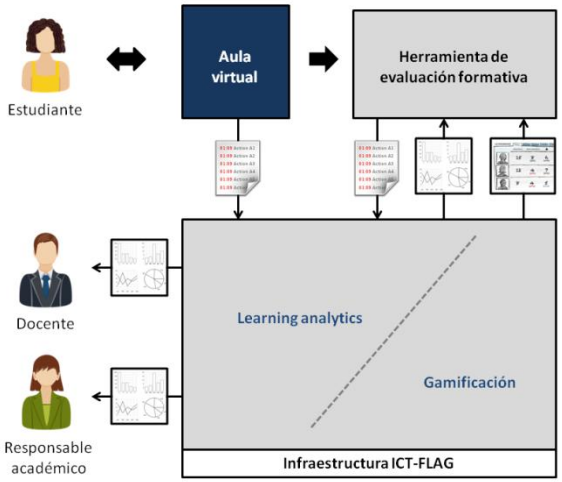
Crans-Montana, Switzerland, March 23-25, 2016

Agenda

- Introduction
- Background
- Methodology
- Experimentation and validation
- Conclusions

Introduction

- ICT-FLAG "Enhancing ICT education through Formative assessment, Learning Analytics and Gamification" (Spanish national project)
- Build a **general eLearning platform** to support:
  - **Formative assessment tools** to provide immediate feedback by means of automatic assessment.
  - **Learning analytics** to monitor the activity and progress of the on-line teaching and learning processes supported by e-Learning systems.
  - **Gamification strategies** to provide an incentive scheme in order to motivate students to practice more frequently and increase their engagement in the learning experience.



Introduction

- Modular architecture required for the platform construction:
  - Customization
  - Distribution
  - Extensibility
- Other desired characteristics:
  - Light dependency (Dependency injection/inversion of control)
  - Pluggability
  - Events / messages between modules
  - Web support (and especially REST)
  - Testing support
  - Easy deployment and configuration
  - Portability

Introduction

- These characteristics are hard to obtain if they are developed from scratch.
- An application framework could facilitate these characteristics and support the whole development.
- **The goal of this work** is to select the most suitable software framework for the construction of the ICT-FLAG platform.

Background – Pre-selection

- Programming languages, frameworks and alternatives:
  - **Java with OSGi**
  - Flask (Python)
  - **Java with Spring**
  - Django (Phyton)
  - Express (Node.js)
  - **Java EE**
  - ASP.NET
  - PHP
  - Ruby on Rails
  - **Out-of-the box products**

Background – OSGi

- OSGi (Open Services Gateway initiative) is a set of specifications
  - Define a dynamic module system for Java.
  - Applications are composed of many reusable modules (*bundles*) which hide their internal implementation from other modules while communicating through services.
  - Esch module registers each service into OGSi *container* that manages the module's lifecycle and instantiates correct services.

Background – OSGi

- **Benefits**
  - Supports modularity by design.
  - Customization. You can select which modules/services you install in each deployment.
  - Enables dynamic updates.
  - Supports versioning.
- **Drawbacks**
  - There is a tendency to over-modularization.
  - The learning curve is quite high.
  - There is a big dependency on the chosen implementation in how to configure things.
  - There is much dispersion of different implementations of the same part of the specification, which are under continuous development.
  - There is much theoretical documentation, but almost all real tutorials are out-dated and there is not a big community of developers who adopted the OSGi standards.
  - Difficult to adopt for enterprise development.

Background – Spring

- A framework for the application development based on Java.

- **Main features**
  - Includes an inversion of control container.
  - Aspect-oriented programming
  - Authentication and authorization.
  - Convention over configuration.
  - Data access.
  - Messaging.
  - Model–view–controller.
  - Remote access framework.
  - Transaction management.
  - Remote management.
  - Testing.
  - Continuous improvement.

- **Benefits**
  - Highly adopted, with a big support community.
  - Can run in a servlet container like Tomcat (do not require a full JavaEE server).

- **Drawbacks**
  - Lots of XML for configuration.
  - Big size and high learning curve.

Background – Java EE

- Collection of specifications for developing enterprise applications.
- **Main features**
  - Support for modularity and dependency injection.
  - Many important specifications and APIs defined by Java EE:
    - CDI (Contexts/Dependency Injection).
    - JPA (Java Persistence API).
    - JMS (Java Messaging Service).
    - EJB (Enterprise Java Beans).
    - Servlets.
    - JSF (Java Server Faces).
    - JAX-RS (Java API for RESTful Web Services).
- **Benefits**
  - Lightweight implementations.
  - Highly adopted, with a big support community.
  - Stability. Many vendors collaborated in its definition and adopted it.
- **Drawbacks**
  - High learning curve.
  - There is much old out-dated documentation about previous versions of the platform which are obsolete (J2EE).

Background – Out of the box solutions

- **Out-of-box products**
  - Liferay.
  - MS SharePoint.
  - VLEs: Moodle, Sakay Blackboard, OpenEdx, LEarnDash OpenMooc.
- **Benefits**
  - Many of the features of the system are already developed, or in general they are easy to adapt to the project requirements.
- **Problems**
  - They include many features that are not required for the project and in most cases cannot be disabled.
  - They need to be adapted or extended to fit the concrete requirements of the project.
  - Certain important features, such as security and licensing are not easily adaptable to the needs of the project. **Discarded**.

Selection methodology

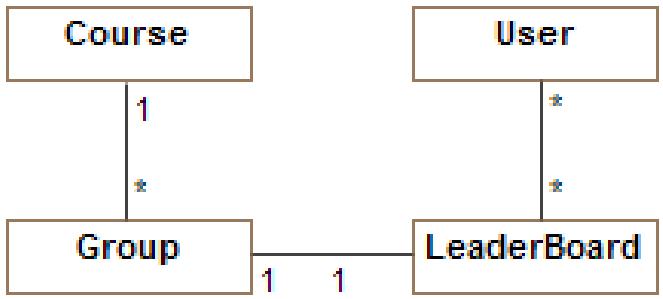
1. **Evaluation of relevant features** of each pre-selected technology
   - OSGi, Spring, Java EE
2. **Evaluation of developing a prototype** with each technology
   - Time development comparison.
   - Performance test comparison.

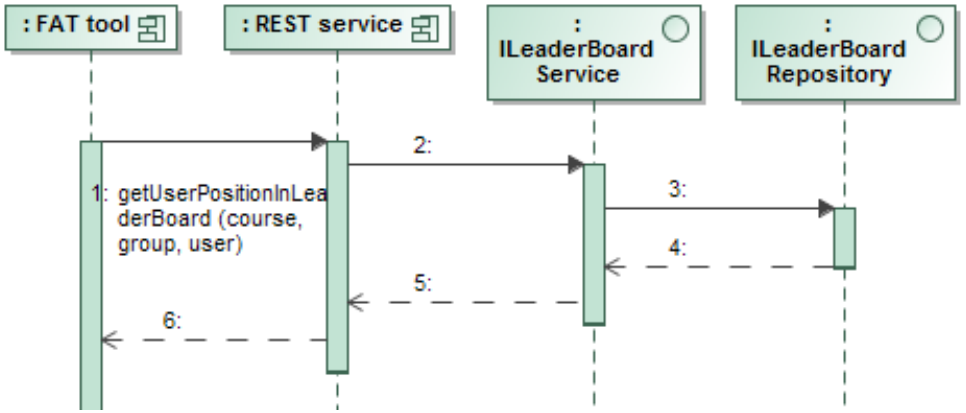
Selection methodology

1. Features to be evaluated:
   - Modularity.
   - Security. Authentication and authorization.
   - Inversion of Control.
   - Persistence.
   - Web application support.
   - Integration.
   - Simplicity.
   - Ease of development.

Selection methodology

2. Construction of a prototype (use case)
   - From a used id returns the user's leaderboard position





- Metrics to be evaluated:
  - Preparation time of the environment
  - Training time
  - Development time
  - Performance (testing)

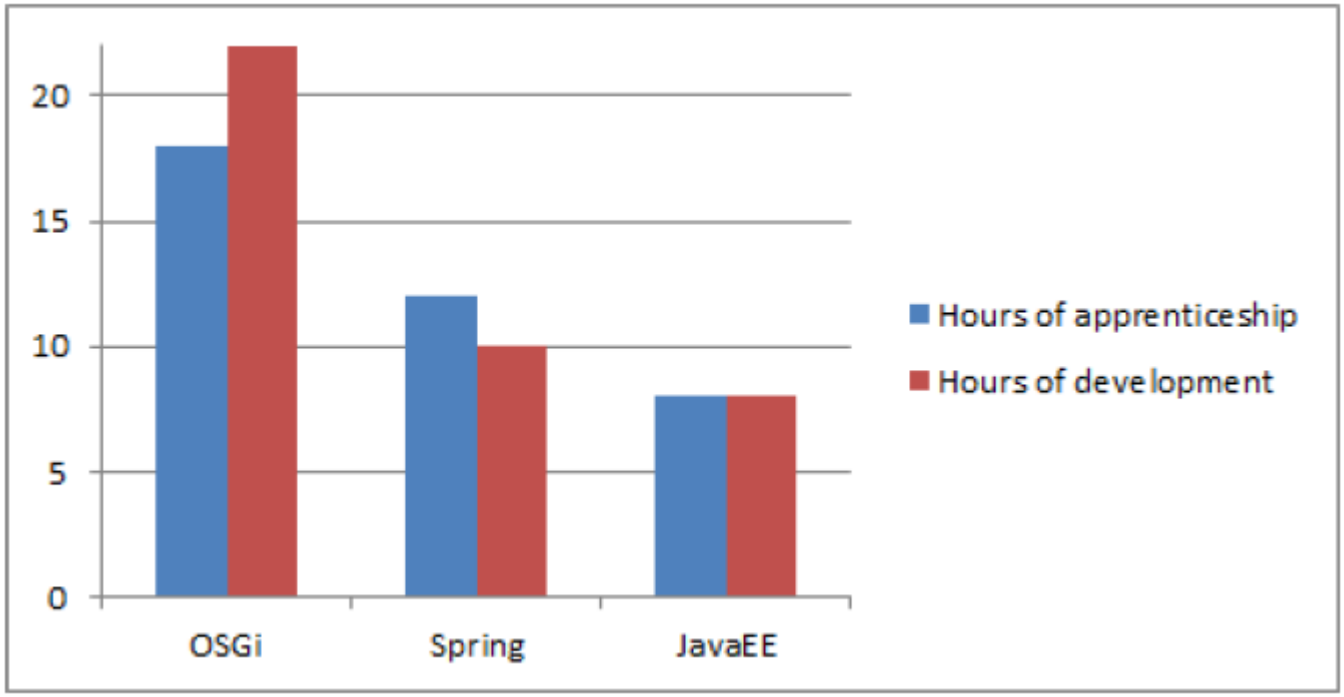
Experimentation and validation

- Features evaluation (scale 0 - 4)

Requirement	OSGi	JavaEE	Spring
Modularity	**4**	3	3
Security	2	**4**	**4**
Inversion of Control	**4**	2	2
Persistence	2	**4**	**4**
Web applications	2	**4**	**4**
Events	2	2	2
Integration	2	3	3
Simplicity	3	3	3
Ease of development	**1**	3	2
TOTAL	22	**30**	29

Experimentation and validation

- Prototype evaluation (training and develop. time)



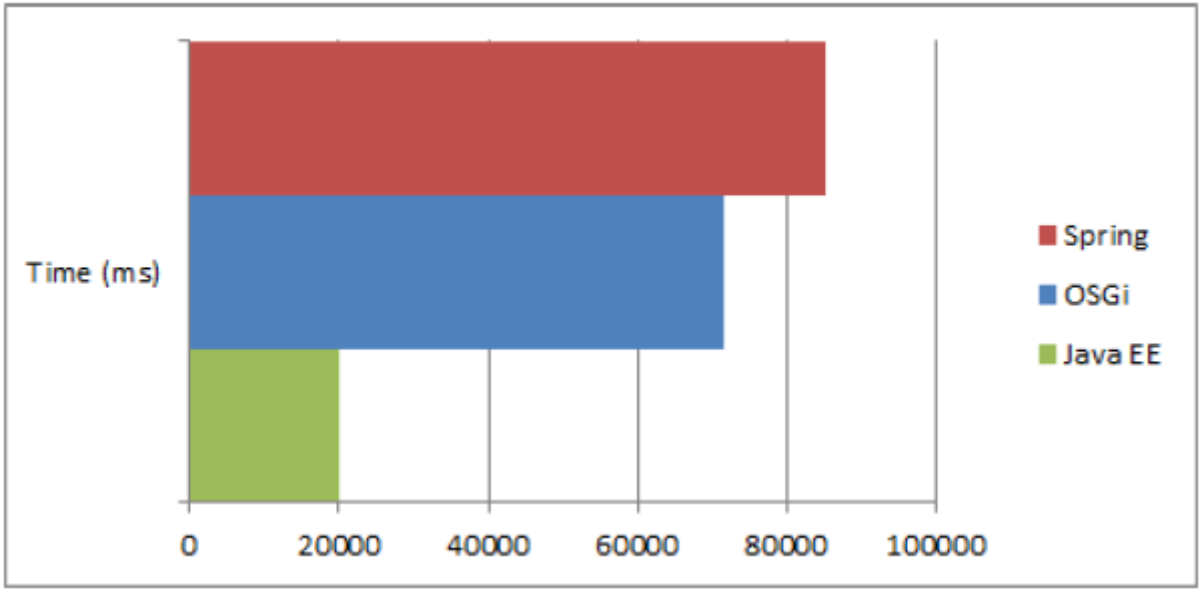
Experimentation and validation

- Prototype evaluation (total time)

	OSGi	Spring	JavaEE
Preparation of the environment	1h	1h	1h
Training	**22 h**	12 h	**8h**
Implementation	**20 h**	10 h	**8 h**
TOTAL	**43 h**	23 h	**17 h**

Experimentation and validation

- Prototype evaluation (performance test)



Conclusions

- We have proposed a methodology to select the most suitable development framework for our web-based e-Learning platform “ICT-FLAG”.
- Several enterprise applications frameworks and alternatives were reviewed
  - OSGi, Spring and JavaEE were selected due to be open, support for modularity and costumization.
  - Out-of-box products were discarded due to over-featuring and licensing concerns.

Conclusions

- The study suggests that **Java EE is the most suitable technology** for the creation of the platform. Justification:
  - Standard de facto technology
  - Great community support
  - Requires smaller development effort
  - Better performance
- We believe that this technology decision is also applicable to other projects with similar requirements
  - High modularity, flexibility and performance.
- On going work is to develop the ICT-FLAG platform and evaluate it in real context of learning.

TOWARDS THE EFFECTIVE SOFTWARE DEVELOPMENT OF AN E-LEARNING PLATFORM FEATURING LEARNING ANALYTICS AND GAMIFICATION

David Gañán, Santi Caballé, Robert Clarisó, Jordi Conesa

Faculty of Computer Science, Multimedia, and Telecommunications
Open University of Catalonia (UOC)
Barcelona, Spain

Crans-Montana, Switzerland, March 23-25, 2016