Project Assignment
MIS-PL / Management Information System Product Line
CMS - College Management System

ODSOFT and EDOM

Master in Computer Science - 2017/2018

Porto, December 5, 2017

Alexandre Bragança (ATB),
Isabel Azevedo (IFP), Nuno Bettencourt (NMB)

`Version 1.5`

Contents

Chapter

1 Context

1.1 MIS Product Line

This project simulates the development of a proof of concept process and product for an hypothetical startup company.

The project aim is to produce a proof of concept solution, state of the art in the field of Management Information Systems (MIS[1]).

The startup aims to innovate in this field by producing and implementing MIS to several business areas in short time frames by applying a **model-driven engineering approach on its software product line**. The software product line should be supported by a **continuous delivery approach**.

The project will be divided in two parts:

1. **The Software Product Line.** This is the part of the **EDOM** Lab Assignment.
2. **The Continuous Delivery Pipeline.** This is the part of the **ODSOFT** Lab Assignment.

1.2 Web Application CMS - College Management System

The startup will demonstrate the project to a group of investors in a near event (i.e., the final presentation of the project).

In this event the startup will present a web application developed using the innovative approaches presented earlier. This application should be based on the project CMS (College Management System)[2].

[1]`https://en.wikipedia.org/wiki/Management_information_system`

[2]`https://bitbucket.org/mei-isep/odsoft-edom-2017-cms-students`

In terms of features, the application should allow students and contacts management and each student should have one or more contacts.

1.3 General Considerations

Each group must only develop the components/requirements corresponding to course units in which they are enrolled.

Chapter 2 ODSOFT

2.1 Introduction

The main goal of the assignment for ODSOFT is to implement a continuous delivery pipeline in Jenkins with declarative pipelines.

The project must include the design and implementation for the provision of 4 major concerns of the solution:

1. **Base Pipeline and Persistence.** Mandatory. Team work.
2. **Code Quality, Integration and Smoke Tests.** Individual work.
3. **Acceptance Tests.** Individual work.
4. **Deployment.** Individual work.

The first item is mandatory for all teams. The first item is a team work and must be developed cooperatively by all members of the team. The 2th, 3th and 4th items should be developed individually. **Issues should be created in bitbucket for each of the concerns by the end of the first week**. In accordance with the structure of the team the following is expected:

- **One member.** Concern 1 must be developed and also one more concern from the other 3.
- **Two members.** Concern 1 must be developed by the two members. Two of the remaining concerns must be developed individually by each team member.
- **Three members.** Concern 1 must be developed by the three members. Concerns 2, 3 and 4 should be developed individually.

2.2 Base Pipeline and Persistence

For this concern:

- The overall pipeline should be designed. This should include a high level design of all the concerns to be included in the pipeline.
- The design should include a description of the process, including the Git organization (i.e., branching model) to be adopted and how it relates to the pipeline.
- The application should be completed with the specific features for the project.
- The application should have a persistence layer that must use a relational database. You may choose the technology to use for this non-functional requirement.
- Implement the pipeline for the specifics of this concern. The base pipeline implementation should include at least the building of the war, running unit tests, their coverage, and publishing reports (unit, coverage and javadoc with Plantuml diagrams).

2.3 Code Quality, Integration and Smoke Tests

For this concern:

- The specific stages for this concern of the pipeline should be designed.
- The pipeline should include a "check" on the code quality of the project by using Checkstyle[1]. Settings for code quality should be defined, including the thresholds for the build health. Checkstyle analysis results are to be published.
- Integration tests should cover the specific features of the project. Reports should be published. Settings for integration test coverage should be defined, including the thresholds for the build health. The build should fail if coverage degrades more than the delta thresholds that should be configured.
- The smoke test should be executed against a docker container with the application (do not forget that the application now has a database!).
- Implement the pipeline for the specifics of this concern.

[1]`http://checkstyle.sourceforge.net/index.html`

2.4 Acceptance Tests

For this concern:

- The specific stages for this concern of the pipeline should be designed.
- Acceptance tests with Cucumber and Selenium should cover the specific features of the project. Reports should be published.
- These tests should be executed against a docker container with the application (do not forget that the application now has a database!).
- The image for this docker container should be published in the docker hub.
- Implement the pipeline for the specifics of this concern.

2.5 Deployment

For this concern:

- The specific stages for this concern of the pipeline should be designed.
- This should be the last stage of the pipeline. The idea is to simulate a live deployment to production.
- This should simulate the production stage using a docker container.
- It should be able to deploy the binaries/resources and also upgrade the database if necessary.
- It also should be able to recover to the last "good" state if the upgrade fails.
- There should be a tag in the git repository related to the release and it also should be possible to identify this version from the deployed artifacts.
- Implement the pipeline for the specifics of this concern.

2.6 Terms and conditions

2.6.1 First Submission

In the first submission students must submit a first version of the technical report (**PR1**) that (at this moment) **only needs to contain the design and a high level analysis of possible alternatives and the justification of the options**. The team must have one issue in Bitbucket for this submission. The report should include one section/chapter for each concern of the project. The project should also **include a very simple implementation of a "skeleton" of the pipeline including all the stages**.

The technical report should be included in the repository of the team in a suitable directory. The format of the technical report must be PDF.

The technical report has a weight of 25% in the project assessment.

Deadline

Until the end of Dec 17.

Feedback

During the Lab class of the week previous to the Christmas interruption the Teacher may provide an "high-level" feedback about the submission. For that, students must present the submission to the Teacher during the class. This is not mandatory and will be provided in a "first come, first served" approach. As such, there is no guarantee that the teacher will be able to provide feedback for all the teams/individuals. The grade will only be available in the final submission.

2.6.2 Final Submission

The final submission includes the submission of a technical report (**PR2**) and of the various technical artifacts, which must both be present in the repository of the team by that date. This stands for 25% of the project score.

The technical report should include:

1. the details of the design;
2. the technical details of the implementation;
3. **an analysis of possible alternatives and the justification of the options**.

The artifacts in the repository include the Jenkinsfile.

After this, a presentation/demonstration session (**D**) of the project will be scheduled. This stands for 50% of the project score. This session is mandatory and should take place during the first two weeks of January.

Deadline

Until 10:00am of January 3, 2018.

2.6.3 Grading

The rating scales are from 0 to 4, with the following semantics (that are adapted to suit the type of the assessment instrument):

- 0- no submission
- 1- did not achieve the requirements of the deadline
- 2- achieved, partially, the requirements of the deadline
- 3- achieved, completely, the requirements of the deadline and includes justification about the options
- 4- achieved, completely, the requirements of the deadline and includes justification about the options and also a detailed analysis of a design alternative that uses the same technologies.

There are 3 rubrics that use the previous rating scale: the first technical report with the design (PR1); the final technical report submitted in the final deadline (PR2); the final demonstration/defense session (D). Scores of the rubrics can be given to one decimal place. The score of the project is calculated to the first decimal place and then converted to the scale 0- 20,0. The formula is: PR1 * 0,25 + PR2 * 0,25 + D * 0,5.

Individual / Team Grading

For each rubric (PR1, PR2 and D) the assessment will be further **divided into individual and group score with each one having a weight of 50%**.

Points of Valorization

1. **When also enrolled in EDOM.** If you are also enrolled in EDOM it will be considered a valorization in grading if you aim also at integrating in the pipeline (ODSOFT) the automatic steps of the application engineering (EDOM). Essentially, these automatic steps (i.e., transformations) must be execute using Gradle (and outside Eclipse) in order to be integrated into the Jenkins pipeline.
2. **For everyone.** It will be considered a valorization the detailed discussion of design alternatives to the techniques and technologies required for this project that can achieve the same goals. However they must be integrated into the Jenkins pipeline.

2.6.4 Assessed Outcomes

This project contributes to the assessment of the following course outcomes:

1. Analyze how continuous delivery can address the difficulties of the software product life cycle management
2. Organize the diverse components of a continuous delivery pipeline
3. Plan tests, configuration management and versioning
4. Manage a continuous delivery approach to software development
5. Debate with colleagues about options in a continuous delivery project

Therefore, it is expected that the student attains these outcomes full/partially with regard to the grading rubrics.

2.6.5 Working Evidences

Students should have evidences of their work in commits in the repository. These evidences are used in the grading.

Chapter 3 EDOM

3.1 Introduction

The main goal of the assignment for EDOM is to implement a product line for MIS applications using a model-driven engineering approach with EMF and related technologies.

The project must include the design and implementation for the provision of 4 major concerns of the solution:

1. **Base Software Product Line.** Mandatory. Team work.
2. **Textual DSL for Domain Modeling.** Individual work.
3. **Diagram Projection with PlantUML.** Individual work.
4. **Code Generation for the GWT Platform.** Individual work.

The first item is mandatory for all teams. The first item is a team work and must be developed cooperatively by all team members. The 2th, 3th and 4th items should be developed individually. **Issues should be created in bitbucket for each of the concerns by the end of the first week**. In accordance with the structure of the team the following is expected:

- **One member.** Concern 1 must be developed and also one more concern from the other 3.
- **Two members.** Concern 1 must be developed by the two members. Two of the remaining concerns must be developed individually by each team member.
- **Three members.** Concern 1 must be developed by the three members. Concerns 2, 3 and 4 should be developed individually.

3.2 Base Software Product Line

For this concern:

- The overall product line should be designed. This should include an high level design of all the concerns to be included in the product line.
- The design should include domain engineering and application engineering aspects.
- The base process for application engineering should reuse artifacts produced during the domain engineering process and include a process similar to the one used for the exercises:
  1. Mindmap modeling (using a textual DSL) for the initial requirements elicitation;
  2. Requirements modeling (using a textual DSL) to state user's needs and constraints;
  3. Use case and entity modeling (using UML) ;
- The process should also include automatic transformations between the previous steps.
- After the last step, the process should include an automatic transformation to generate the code for the GWT platform (i.e, CMS application). Note: this step does not need to generate an intermediate UML design model nor completely produce an executable application in a totally automatic way (this will only be required if the team will be developing the concern number 4).
- The process should support the MDE development of the web application as described in Section 1.2.
- There should be included a theoretical analysis on the impact of developing other applications with the product line. For instance, if building a different college application with teachers and courses instead of students and contacts what modifications are necessary?

3.3 Textual DSL for Domain Modeling

During the exercises, in the analysis model, entities were modeled using UML classes and included in the same model as use cases. The overall goal of this concern is to develop a DSL for domain modeling that will avoid the need to do domain modeling in UML. In this new scenario, entities, their structure and relationships, are modeled in the new DSL for domain modeling. The use case model continues to be generated and modeled in UML.

For this concern:

- Design and implement a new textual DSL for domain modeling.
- Entities, their structure and relationships should be modeled using this new DSL.
- Update the transformation that generates code from the analysis model (the analysis model in UML, including Use Cases and Entities) to use the new domain models. You should use the following approach:
  - **First Step**. Use an ATL transformation to "compare" the use case model and the domain model and verify if they are in synch. For instance the use cases should not reference entities that do not exist in the domain model.
  - **Second Step**. If the verification of the previous step is successful then you should use a new transformation (Acceleo) to generate the code. For this transformation you should try to generate only code for the entities that are referenced in use cases. Note: this step does not need to completely produce an executable application in a totally automatic way (this will only be required if the team will be developing the concern number 4).

3.4 Diagram Projection with PlantUML

This concerns aims at supporting diagrammatic projections of UML models using PlantUML.

The base project uses UML for the "analysis model", where use case models as well as entity models (using classes) are represented. The goal of this concern is to generate diagrams for both models using PlantUML.

Similarly to exercise 5, the solution should be able to generate Plantuml textual files that represent diagrams of the use case models. It should also be possible to generate Plantuml textual files that represent diagrams of classes for the entity model. This should be accomplished with Acceleo transformations. The inverse transformation should also be possible, i.e., to generate UML models from Plantuml textual files. For that, a textual DSL that supports the subset of PlantUML used for these concerns should be developed. This DSL can then be used to "read" textual PlantUML files as input in an ATL transformation that can produce UML models from PlantUML files.

For this concern:

- Design and implement an Acceleo transformation that can be used to generate PlantUML use case diagrams from use case models in UML2;
- Design and implement an Acceleo transformation that can be used to generate PlantUML class diagrams from entity models represented in UML2;

- Design and implement a new textual DSL that can represent a subset of PlantUML that support use cases and class diagrams. The DSL should be able to model the same diagrams as the ones resulting from the previous 2 items;
- Design and implement an ATL transformation that can be used to generate UML2 models from instances of the previous DSL.

3.5 Code Generation for the GWT Platform

During the exercises, a simple approach was developed to generate code for the application based on a UML design model. For this concern the goal is to expand the approach so that the **complete** application can be generated. This should include not only the java code but also all other necessary artifacts, such as xml files and other configurations. It should be possible to build and run the application after the transformation (without any manual coding or configuration). This goal may require changes in the structure/architecture of the application that can be executed if properly justified, the application continues to be based on GWT and the UI of the application is similar.

For this concern:

- Design and implement an ATL transformation that is able to generate the UML design model from the analysis model (i.e., use case and entity model in UML)
- Design and implement an Acceleo transformation that is able to generate the implementation artifacts (e.g., code, xml configuration files, resources for multi-language support) from the design model

3.6 Terms and conditions

3.6.1 First Submission

In the first submission students must submit a first version of the technical report (**PR1**) that (at this moment) **only needs to contain the design and a high level analysis of possible alternatives and the justification of the options**. The team must have one issue in Bitbucket for this submission. The report should include one section/chapter for each concern of the project.

The technical report should be included in the repository of the team in a suitable directory. The format of the technical report must be PDF.

The technical report has a weight of 25% in the project assessment.

Deadline

Until the end of Dec 17.

Feedback

During the Lab class of the week previous to the Christmas interruption the Teacher may provide an "high-level" feedback about the submission. For that, students must present the submission to the Teacher during the class. This is not mandatory and will be provided in a "first come, first served" approach. As such, there is no guarantee that the teacher will be able to provide feedback for all the teams/individuals. The grade will only be available in the final submission.

3.6.2 Final Submission

The final submission includes the submission of a technical report (**PR2**) and of the various technical artifacts, which must both be present in the repository of the team by that date. This stands for 25% of the project score.

The technical report should include :

1. the details of the design;
2. the technical details of the implementation;
3. **an analysis of possible alternatives and the justification of the options**.

After this, a presentation/demonstration session (**D**) of the project will be scheduled. This stands for 50% of the project score. This session is mandatory and should take place during the first two weeks of January.

Deadline

Until 10:00am of January 3, 2018.

3.6.3 Grading

The rating scales are from 0 to 4, with the following semantics (that are adapted to suit the type of the assessment instrument):

- 0- no submission
- 1- did not achieve the requirements of the deadline
- 2- achieved, partially, the requirements of the deadline

- 3- achieved, completely, the requirements of the deadline and includes justification about the options
- 4- achieved, completely, the requirements of the deadline and includes justification about the options and also a detailed analysis of a design alternative that uses the same technologies.

There are 3 rubrics that use the previous rating scale: the first technical report with the design (PR1); the final technical report submitted in the final deadline (PR2); the final demonstration/defense session (D). Scores of the rubrics can be given to one decimal place. The score of the project is calculated to the first decimal place and then converted to the scale 0- 20,0. The formula is: PR1 * 0,25 + PR2 * 0,25 + D * 0,5.

Individual / Team Grading

For each rubric (PR1, PR2 and D) the assessment will be further **divided into individual and group score with each one having a weight of 50%**.

Points of Valorization

1. **When also enrolled in ODSOFT.** If you are also enrolled in ODSOFT it will be considered a valorization in grading if you aim also at integrating in the pipeline (ODSOFT) the automatic steps of the application engineering (EDOM). Essentially, these automatic steps (i.e., transformations) must be execute using Gradle (and outside Eclipse) in order to be integrated into the Jenkins pipeline.
2. **For everyone.** It will be considered a valorization the detailed discussion of design alternatives to the techniques and technologies required for this project that can achieve the same goals. However they must be integrated into EMF (i.e., they must use the same metamodels/models).

3.6.4 Assessed Outcomes

This project contributes to the assessment of the following course outcomes:

1. Analyze domain engineering as the top of an iceberg composed of a myriad of processes, techniques, tools and methodologies which all have reuse in software engineering as the ultimate goal
2. Analyze software product lines as a practice that promotes large scale reuse
3. Design and implement domain-specific languages and model transformations in the context of model-driven engineering

4. Manage recent approaches in the context of domain engineering and reuse: model-driven software engineering and domain-specific languages

5. Debate about options in projects related to model-driven engineering

Therefore, it is expected that the student attains these outcomes full/partially with regard to the grading rubrics.

3.6.5 Working Evidences

Students should have evidences of their work in commits in the repository. These evidences are used in the grading.