Student #3, Sprint 1: Analisis report

Group: C1.04.14

Repository: https://github.com/marizqlav/Acme-L3-D01

Student #1

Name: Domínguez-Adame, Alberto
email: albdomrui@alum.us.es

Student #2

Name: Herrera Ramírez, Ismael
email: ismherram@alum.us.es

Student #3

Name: Olmedo Marín, Marcos
email: marolmmar1@alum.us.es

Student #4

Name: Izquierdo Lavado, Mario
email: marizqlav @alum.us.es

Student #5

Name: Merino Palma, Alejandro
email: alemerpal@alum.us.es

Table of contents

Summary

Acme Life-Long Learning, Inc. (Acme L3, Inc. for short) is a company that specializes in helping learners get started on a variety of matters with the help of renowned lecturers. The goal of this project is to develop a WIS to help this organization manage their business.

Revision table

Number	Date	Description
1	15/03/2023	Full redaction of the document

Introduction

This document lists analysis records, each one including the following data: a verbatim copy of the requirement to which the record refers; detailed conclusions from the analysis and decisions made to mend the requirement; a link to the validation performed by a lecturer. .

This document has the following structure:

- Analysis report

Contents

Analysis records

Requirement nº5: An audit is a document with auditing records regarding a published course. The system must store the following data about them: a code (pattern "[A-Z]{1,3}[0-9][0-9]{3}", not blank, unique), a conclusion (not blank, shorter than 101 characters), some strong points (not blank, shorter than 101 characters), some weak points (not blank, shorter than 101 characters), and a mark (computed as the mode of the marks in the corresponding auditing records; ties must be broken arbitrarily if necessary).

There were to design decisions regarding this requirement, both about the mark attribute. The first about if it should be transient or not, the first one about how to obtain the required auditingRecords.

About it being or not transient:

Decisions: Due to the mark attribute being an attribute derived from the related AuditingRecords of the Audit, we had to obtain it somehow from the Audit. Some options where considered:

- We could make the mark attribute a transient one and obtain it every time the user calls the function.
- We could make the mark attribute a serializable attribute and have it update every time a new AuditingRecord is added to the audit.

Conclusions: It was decided the former, since the method for obtaining the mark from the related AuditingRecords is not really performance heavy. This was verified by the teacher at laboratory class.

About how to obtain the auditingRecords.

Decisions: in order to calculate the mark, we need all auditingRecords related to the audit. For that:

- We could use a bidirectional relationship. This is the more intuitive and simple system.
- We could use complicated and queries so that the mark would be a property obtained via the auditService and it would not be included anyway in the entity.

Conclusion: after some talking with our lab teacher, they told us to not use OneToMany or bidirectional relationship in any case, so we used to second option, leaving mark as a transient attribute in our UML and creating a method in the AuditService class.

Requirement nº7: The system must handle auditor dashboards with the following data: total number of audits that they have written for theory and hand-on

courses; average, deviation, minimum, and maximum number of auditing records in their audits; average, deviation, minimum, and maximum time of the period lengths in their auditing records.

All students had some doubts about how we should represent average, deviation, min and max values in all dashboards. We had some options:

- We could have all values with their names as an individual attribute. This would allow more flexibility and granularity when if we needed to use these values alone later. It would also be the more intuitive way.

- We could join related values in a map (average, deviation, min and max attributes together) since they are calculated together and they are likely to be required together.

It was decided that the average, deviation, minimum and maximum would be returned via a map with the “average”, “deviation”, “min” and “max” keys and their respective values. This was verified by the teacher at lab class.

Requirement nº4: There is a new project-specific role called auditor, which has the following profile data: firm (not blank, shorter than 76 characters), professional ID (not blank, shorter than 26 characters), a list of certifications (not blank, shorter than 101 characters), and an optional link with further information.

Decisions: we thought that the professional ID attribute would probably be a unique attribute among all auditors.

Conclusions: we ask out lab teacher and he confirmed this, so we made it unique.

Period datatype problems:

Decisions: Since there were a lot of classes that required a duration attribute, we thought some options to solve the problem all together. We first thought of using a simple Duration value with the Double type. Then we thought of using to attributes, firstDate and lastDate to be able to obtain the duration between the two. Finally, we thought of creating a custom datatype called Period which would contain this two dates so it would be easier to implement.

Conclusions: we thought the double type option wouldn’t be enough, and our lab teacher told us too. We also created a datatype called Period to substitute the duration attributes. However, we weren’t able to use them due to a framework error in the population process. More information about this can be found in the On your tutorials section of the subject by the name of **On framework parsing error (probably)**.

So we finally decided to use two simple dates attributes on each class requiring it.

Conclusion

In conclusion, this sprint required some more analysis regarding the entities and their complex attributes, but it was handled correctly thanks to a good understanding of the subject and prepared organization.

Bibliography

Intentionally blank.