UML Domain Model changes

Relationships

We have decided to change all the navigability to remove problems in future deliverables, because we've considered they are difficult to implement.

- ### Relationship: FixUp Task-Warranty:

  Before: Bidirectional.

  We believe this modification improves the efficiency of the system, although the resulting JPQL queries are more difficult. Previously, when consulting a fix-up task, we bring all the associated warranties and vice versa, so we consult a fix-up-task all the time, and we don't need them at all, that's why we've decided to implement the following one:

  After: FixUp Task <----- Warranty

- ### Relationship: FixUp Task-Category:

  Before: Bidirectional.

  We believe this modification improves the efficiency of the system, although resulting JPQL queries are more difficult. Previously, when consulting a fix-up task, we bring all the associated warranties and vice versa, so we consult a fix-up-task all the time, and we don't need them at all, that's why we've decided to implement the following one:

  After: FixUp Task -----> Category

- ### Relationship: Referee-Report:

  Before: Bidirectional.

  In this case, we have decided this implementation because, if we request a referee, we don`t always need to bring their complaints, so we considered to implement the navigability in this way, so that only every time we consult a complaint, we bring its associated referee.

  After: Referee <----- Report

- Relationship: Box-Actor:

  Before: Bidirectional.

  We thought that bringing the actors related to a box we want to obtain is the most efficient way to implement this relation to the system. With a bidirectional relationship between these two classes (as it was made at first), the system obtains either a collection of boxes and a collection of actors.

  After: Box <----- Actor

- Relationship: HandyWorker-Curriculum:

  Before: Bidirectional.

  With a bidirectional relationship, when consulting a handy worker, we also bring all the associated Curriculum, and vice versa. This results in an inefficient implementation, so we've decided to change the relationship, although resulting JPQL queries are more difficult.

  After: HandyWorker <---- Curriculum

- Added a new relationship between Customer and FixUp Task.

- Multiplicity between FixUp Task and Application has been changed from 0...* (FixUp Task) and 1 (Application) to 1 (FixUp Task) and 0..* (Application). Their navigability has also been changed, making the resulting queries easier.

- Added a new relationship between Complaint and FixUp Task because we found problems trying to get the complaints of a specific FixUp Task, with a multiplicity of 1, and 0...* on the Complaint side.

Attributes

- We also had to add an attribute to the finder entity, lastUpdate (Date type), in which we save the date of the last update. This is necessary because we need to know when the browser is not updated in order to update it.

Entities

- We also added a new entity called configuration, in which we store the results of the finder, the finder cache time, and a phone number pattern.

- In the section entity we have changed the name of the id attribute to “number”, because we had problems with the implementation.

Navigability and multiplicity character of attributes

- We decided to change the applicable laws restriction because we considered that the previous one was not correct, since we decided that the applicable laws could be optional.

- We have also decided to change the multiplicity of several entities, as we felt they were wrong after a second reading of the information requirements:

    - Complaint---Report (multiplicity changed).
    - Referee----Report (multiplicity changed.)
    - Message----Box (multiplicity changed.)

- We also changed the multiplicity from Category to FixUp Task, swapped from 1 to 0..*, since we consider that a category should not be associated mandatory to a fix-up task, that is, it can create a category that does not have any associated fix-up task.

Roles

- We have changed the roles of Application and Endorsement from the conceptual model, since it produced an error in our model because two of them were redundant.

Java Domain Model changes

In the second delivery we didn´t know we had to include relationship attributes in Java classes (excepts composition attributes), so we have implemented them in this delivery.

- Attribute personalRecord in Curriculum class was Collection<PersonalRecord>, but we noticed that multiplicity from Curriculum to PersonalRecord is 1, so we changed the type of that attribute to PersonalRecord.

- *@DateTimeFormat* annotation from getters of Date type attributes has been deleted because they don´t matter in this deliverable and in the previous one.

- We have decided to remove the *@NotNull* attribute from the Curriculum class, because we considered that there were more positions, since a handy worker does not have to have a miscellaneous record mandatory. So, the only one that would be necessary should be personalRecord.

- We have added the *@Transitional* annotation to the flagSpam attribute, since we consider that its calculation can be made relatively fast.

- In Spring, the *@NotBlank* annotation includes the not-null restriction. In optional attributes, we don´t want this (it doesn't make sense). So, in the domain model, we have kept this annotation, but, in java, we didn't include it.