MESTRADO EM ENGENHARIA INFORMÁTICA / LICENCIATURA EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

Arquitectura de Sistemas de SOftware

15th May 2008 • Personal notes allowed • Duration: 90 Minutes

Read carefully the description of the software system below considering it as a concrete of application and then answer the questions that follow, **always justifying them succinctly and clearly**.

In the answers, when useful, you may (and should) mention the bibliography or references that fundament the answers. When required, you should make explicit all the assumptions you did to answer the questions.

CASE

We want to develop a TeamUML, a very simple web-based graphical editor for UML diagrams (class diagrams and sequence diagrams).
One distinctive characteristic of this editor is allowing the collaborative edition of the diagrams by several users at the same time. The updates are immediately refreshed in all users' screens/browsers without any need of intervention by the user. For example, once one user adds a new class to a diagram being shared by several collaborators, all the other collaborators will see the diagram being updated immediately.
This editor will need to be extended with new diagrams in the future, using a simple configuration of the meta-model it supports.

After a detailed requirements analysis, we found the following as the more relevant in terms of architecture:

1. The editor must run in a web browser.
2. The editor must support the idea of a project, which may contain several diagrams, eventually organized in folders.
3. The editor must be able to edit several diagrams at the same time.
4. A diagram has a specific type (class, sequence, etc) and it allows the insertion of the UML elements defined as being part of that particular kind of diagram. For example, it may be defined that a class diagram doesn't allow the insertion of nodes, but only classes, relationships and notes.
5. Projects, folders, and diagrams, all must be stored in a project repository: ideally a CVS or SVN, but also a database or file-system accessible in the web-server.
6. The editor must be easily extended with new kinds of diagrams and modelling elements.
7. The editor must be easily extended with new tools and operations over the graphical elements.
8. The editor must refresh immediately (without any user action) after updates done on the diagram being edited.
9. The editor must record all the history of changes, to step back and forth in the revisions, and compare revisions (e.g. like a diff over diagrams).
10. Undo/Redo must also be supported.

QUESTION 1: ARCHITECTURAL STYLES **30%**

Consider the architectural styles studied. Answer and justify the following questions:

a. Which key architectural problems do you find as more relevant in this case?

b. Which style you consider to be the most appropriate to adopt as the **main style** for the TeamUML? Characterize that style in terms of components, connectors and constraints, and instantiate them to the case. Relate them to the architectural problems above.

c. Which other styles you consider to be interesting to consider, in a complementary way? Justify with concrete situations of the case. Relate them to the architectural problems above.

QUESTION 2: DESIGN PATTERNS **50%**

Considering the design patterns you studied, suggest and justify which you think are more appropriate to solve each one of the design problems below:

a. A project may aggregate folders and diagrams, but not other projects. Some diagrams and folders can be imported as references from other projects. How to represent these kinds of objects to enable the editor actions to act on them in a similar way?

b. New diagrams and elements can be dynamically added on the fly by simple configuration.

c. If the editor is disconnected from the web server (offline) the edition is still possible but the storage is done locally in a file and synchronized only when online again.

d. Diagrams can be edited at the same time by different users in different computers, but the updates are propagated without user action.

e. Diagram changes must be recorded to enable Undo/Redo, the notion of revision and history.

QUESTION 3: PROGRAMMING PARADIGMS **20%**

Consider that we developed the TeamUML only for class diagrams and sequence diagrams, but now we need to add more kinds of diagrams. The existing codebase is Java and we have heard that Aspect-Oriented-Programming can be very helpful for software evolution. However, we know that object-oriented frameworks and refactoring to patterns are also very good to add configurability and variability to an existing system.

Please elaborate on which paradigm (OOP vs AOP) you would suggest to use when evolving this system to support the new features below. Justify by pointing to the characteristics of both approaches.

a. New diagrams on the fly.

b. Propagation of updates to new actions supported by the editor.

The End.