Software Architecture 2011 - Project

This term you will carry out a project with the prime goal of putting to practice the main principles and techniques that you learn in the Software Architecture course.

Project Description

Relational Database Access Library

Most applications use relational databases (rdb) to persist their data. Your task is to develop an application programming interface (API) for rdb access using Eiffel for both design and implementation. The API should stress simplicity, clean O-O design and usability, and support the following features:

Main rdb operations

Your library should support the main rdb operations (CRUD: Create, Read, Update, Delete) using SQL strings. SQL strings can either be passed to the database as they are (assuming you get the SQL strings from an external application or a file), or be pre-processed by the API (ad-hoc assembled strings, predefined queries).

Cross-rdb-transparency

Your library front-end should stay the same no matter what rdb is plugged in the back-end. We provide you with two libraries that can be used to test the cross-rdb-transparency requirement, one using odbc (EiffelStore) and the other one using a direct connection. Both will use MySQL rdb, so this is the only rdb you need to install.

Prepared statements

See http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html for an explanation of prepared statements, and why it is important to use them. Both the libraries we make available already support them.

Exception handling

Exception handling will have to focus on providing error messages that are as informative as possible, as well as useful preconditions for the clients.

Object-relational mapping (optional)

Object-Relational mapping refers to the need of mapping objects to relational tables and back. This is needed if you want to provide an O-O front-end that hides SQL to the library user. You can decide on the degree of sophistication for the mappings you provide support to.

The architecture of the library is left to your creative imagination and pragmatic instincts. You might consider using some of the design patterns you will learn about during the lecture and the exercise sessions.

Initial Setup

The project is carried out in **groups of four** students. You have to form the groups by **Friday 25. February**. Once you have a group, send **one** email per group to Julian Tschannen (julian.tschannen@inf.ethz.ch) with the names of the group members and their Origo usernames (register on origo.ethz.ch if you have no account yet). Also specify, if your group has a restriction on the day of the exercise session (Monday or Tuesday). If you don't find a group, also send us an email with your name, Origo username, and restriction on the day of the exercise session, so we can put you together with other students.

We will use SVN on Origo for source control. All submissions (documents and source code) will be delivered through this repository. You will have to create an Origo project for your team. Go to http://www.origo.ethz.ch/origo_home/create_project; use *sa11-TEAMNAME* for the project name; write a quick description; select *Closed Source* project; and set the project listing visibility to *Not Visible*. Once your project is approved, add all team members and your assistant (once you have been assigned to an exercise group) as *Project Owner* to the project via *Manage Project > Project Settings*.

Milestones

The project is done in 5 phases: Requirements Specification, API Design, Test Plan, Implementation, and Testing and Revision. In each phase you will have to deliver a document, source code, or both.

Requirements Specification (30 points)

Duration:	Monday 28. February - Sunday 20. March, 24:00 (3 weeks)
Deliverable:	SRS document (15-30 pages)
Template:	on course page
Submission:	your-Origo-SVN/documents/requirements.pdf

Task:

The ultimate purpose of requirements specification is to describe *what* qualities a system must exhibit and *what* goals it must reach. It does **not** describe *how* these qualities and goals are achieved. Your task is to develop an SRS document for your project, which will serve to clarify the scope and features of your project and to prioritize the implementation of functionality.

We will provide you with a template and example documents from previous years. When you write your SRS document, make sure your document adheres to the 15 quality goals discussed in the lecture.

API Design (30 points)

Duration:	Monday 21. March - Sunday 10. April, 24:00 (3 weeks)
Deliverable:	Design document (3-6 pages), API classes
Template:	on course page
Submission:	your-Origo-SVN/documents/design.pdf
	your-Origo-SVN/src/

Task:

Write a set of Eiffel classes that will be the public API of your library. The set of features has to satisfy the requirements described in the SRS. The Eiffel classes have to be equipped with class and feature comments as well as contracts. The routines don't have to be implemented yet.

In the design document, add a class diagram of the library and describe which design patterns you have used. For each design patterns specify which classes are part of the pattern and why you used this particular pattern.

Test Plan (10 points) [in parallel with implementation]

Duration:	Monday 11. April - Sunday 8. May, 24:00 (4 weeks)	
Deliverable:	Test suite	
Submission:	your-Origo-SVN/testing/documents	(SRS and design document of other team)
	your-Origo-SVN/testing/src	(API classes of other team)
	your-Origo-SVN/testing/tests	(your test suite)

Task:

In this project step, you will receive the SRS, the design document, and the API classes of another team's project. Based on this material you develop a test suite for their project. This means that you will take the role of external test engineer.

Put the SRS, design document, and the source code you received into your SVN repository in the *testing* directory. Write tests for each **functional requirement** of the SRS. Annotate each test with the requirement that it exercises. In addition, write tests for each **public API routine**. Again, annotate each test with the routine under test.

Implementation (20 points) [in parallel with test plan]

Duration:	Monday 11. April - Sunday 8. May, 24:00 (4 weeks)
Deliverable:	Revised design document, Implemented classes
Submission:	your-Origo-SVN/documents/design.pdf
	your-Origo-SVN/src/

Task:

Use the prioritized functional requirements to decide on what functionality should be implemented first. Make sure that your code satisfies both the functional and the non-functional requirements.

If you have to change the design or API, revise the design document and add a chapter where you explain what changes were made in the design and why.

Testing and Revision (10 points)

Duration: Monday 9. May - Sunday 29. May, 24:00 (3 weeks)
Deliverable: Test report (1-2 pages)
Template: on course page
Submission: your-Origo-SVN/documents/test_report.pdf
Task:

Give the implementation developed in the previous step to the team testing your project. If you have changes in the API, send the revised design document as well.

Run the test suite that you have created in the test plan phase. If there were changes in the API, you might have to adapt your tests. Document all bugs that you have uncovered using the *issue tracker* of the other team's Origo project. Write a short test report describing the results of the tests and assess the quality of the code under test.

When you get bug reports for your own project, fix them.

Presentations

You will be doing three presentations through the course of the project.

Requirements

Date: Monday 21. March or Tuesday 22. March
Duration: 10' (4-8 slides)
Content:

The presentation should give an overview of your SRS and explain the most important points. If there were any issues during the SRS writing phase that created discussion among the team members, they could make an interesting presentation topic.

Design

Date: Tuesday 12. April or Monday 18. April
Duration: 10' (4-8 slides)
Content:

In the presentation you should show an overview of your design. Explain important design decisions and describe which patterns you used.

Final Presentation

Date: Monday 23. May or Tuesday 24. May
Duration: 15' (8-12 slides)
Content:

In the final presentation you should describe how you worked on the project, how you distributed the work and handled communication in the group. You can also show a demo or code snippets to show the usability of your library. As a conclusion of the course, give your personal impression of the project: are you happy with the result? would you do some part differently now?

Grading

The project will account for 50% of the final grade, with the other 50% determined by the semester-end exam. You need to have at least a 4.0 in both the project and the exam to pass the course!