ECE450 – Software Engineering II

Winter 2007

Assignment 2 – Architectural comparison of two software projects

Objectives

This assignment has two objectives:

1. To understand that there are different architectural solutions to a software problem.
2. To think critically about architectural decisions and distinguish their strengths and weaknesses.

Due Date

This assignment is due at 5:00pm on **April 10th, 2007**.

Description

Even though there are often many software projects that address the same issue, in most cases they follow divergent architectural paths that lead to entirely different solutions. Your task for this assignment is to analyze in detail the architecture of two similar projects (e.g. two RSS readers, two image viewers), to locate their most important architectural differences, and to think critically about these differences and the consequences they have for each project.

Unless you *must* change software projects, you will be working with the two open source applications that you explored for assignment 1. If you have to switch projects, discuss it with your instructor in advance.

Your architectural comparison should be based on material from our course, and additional resources researched by your team. It should be summarized in a report, including the following elements:

A one- or two-paragraphs **executive summary** that mentions the projects you selected and a few key findings of your comparisons.

An **introduction** of no more than one page that mentions your selected projects, explains the process you followed to work on this assignment, obstacles you encountered, and describes the contributions of each team member.

A one-page **domain description** section that summarizes the problems that your applications are intended to solve. Note that this is not a description of the applications themselves, but a user- or business-oriented description of why they exist.

A one-page **feature differences** section that draws distinctions between the features of the two applications.

A four-page **code layout** section that describes for each project what files are in what directories, and why. If the applications depend on configuration or data files at runtime, you must describe those as well.

A **conceptual architecture** section, of no more than six pages, that summarizes the logical components of the applications and draws similarities and distinctions between them. Which conceptual components are part of the system, and which are external? How do typical use-cases map onto these components?

An **implementation architecture** section of no more than six pages that summarizes the applications' implementation architectures: What is the class hierarchy of each? What design patterns are used? Make sure to draw similarities and distinctions between both applications again.

An **execution architecture** section of at most six pages. For this section, define three common use-cases that the applications must handle, and describe the steps that each system takes to execute them at run-time. A discussion of similarities and distinctions between the two applications is expected.

An **architectural analysis** section, as long as you need it to be, that analyzes critically and summarizes the architectural strengths and weaknesses of each application, based on your observations from the previous sections. Can you determine if one of the two architectures is superior, and why? Can you spot the trade-off decisions that each architect encountered, and how were they resolved?

Other sections as needed – for example, appendices or references may warrant a section of their own.

You can include tables, figures, and diagrams in every section. For the architectural discussion sections, diagrams and figures are not only allowed, but expected.

Deliverables

You must submit an electronic report in **PDF** format as an attachment to your instructor's email account. Name your file: "TeamName-Assig2.pdf". The subject line of your email should be "[ece450] [team name] Assignment 2". Your report will be made available to the rest of the group in a password-protected section of the website.

Marking Criteria

You will be marked on the quality of your descriptions and of your analysis of each section, as well as on the quality of your writing.

For each section, try to reach a balance between comprehensiveness (do not exclude anything that should be included) and conciseness (no fluff, please). You are describing to an outsider the architecture of systems that you probably know well by now: explain the concepts in a way that a software-literate audience understands (so you don't need to explain, for instance, how to read your class diagrams), but do not leave out project-specific details necessary for understanding your applications.

The most important factor for your grade in each of the descriptive sections (that is, up to the execution architecture section) is the quality of your descriptions and observations: are your findings worthy of interest? Are you presenting them in a way that makes similarities and differences evident? Are you making proper use of visual aids (tables, figures)?

In contrast, the most important factor for your grade in the analysis section is your critical judgment. Why is one architecture better than the other? Under which circumstances? Can the drawbacks of each architecture be overcome? How? What would you do differently? In sum, show that you're thinking critically about the software architectures.

Finally, regarding the quality of your writing: we expect a term-paper quality from your team. Proper presentation and use of visual aids is expected. Tables of contents, page numbers, references, labelled figures, and other presentation elements are required. If writing is not one of your skills, consider submitting your report to a writing centre in advance.

Suggestions

You may want to read the following paper to get ideas about conceptual architectures:
Ahmed E. Hassan and Richard C. Holt. *"A Reference Architecture for Web Servers"*. 7th Working Conference on Reverse Engineering, 2000.

If you have never thought about software architectures critically, you may find it hard to get going with your observations. These tips might get you started:

Familiarize yourself with the code you're working with! It will be impossible for you to think critically about it if you have only browsed it briefly. Play around with your local copy until you are sure you understand it.

Visit the library and get some software architecture books; find out how other authors comment software architectures and what visual aids they use to drive their points home.

Follow your intuition. It sounds funny to say so, but while exploring someone else's code, you'll notice things that feel right or wrong. Stop and ask yourself why that is so. Did the author reach a particularly clever solution to an architectural problem? Why is it a good solution? Alternatively, is there something in the code bothering you? Can you describe it? How would you fix it?

Think about stress cases: what happens when the application has {a huge database / too many users at the same time / etc.}? Will there be any bottlenecks? Why? Can they be fixed?

Finally, as before, use your team blog to discuss project-related matters. It's a good idea to leave a trail of evidence that you care about your projects, in case things go wrong later in the term.