Assignment #1

ECE 750-T26 – Fall 2013

1. Overview

For the assignment questions below, follow these steps in deriving your solutions.

2. Case Study: Market Buddy

You were hired as a software engineer for a company that develops a family of financial software products. One of these software systems, called Market Buddy, which handles stock market trades, was assigned to you to develop and maintain. The software is intended to receive typical trade requests from trading clients via command-line and graphical interfaces, submit the trade requests via the corresponding stock market connections (e.g., TSE, NASDAQ), monitor the trade requests, and provide updates to the trading clients.

Each client is provided with an account for which they register outside the system – and outside the scope of your responsibility for now. Once they register, they are provided with login credentials that they can use to log into Market Buddy. The software keeps track of each client's running balance, all of their open trades, and keeps a detailed log of executed transactions for billing and legal purposes.

The clients are billed for each trade at a fixed rate. The software also needs to provide a special interface where its clients can request a professional trader to execute a trade for them, but for such trades, the clients will be charged a higher trade fee.

Finally, Market Buddy includes an administrative component, where administrators with special privileges can access the system, setup new accounts, and monitor or update the existing ones.

a. Requirements Engineering

First, identify all of the major use cases for the Market Buddy software system. For the purposes of this assignment, your course instructor is your "client". For this part of the assignment, you are allowed to submit up to two requirements elicitation questionnaires to him via email. Otherwise, you are encouraged to study the domain of financial software and similar software systems in that domain, and extract the corresponding usage scenarios and reference requirements.

Once you have established the requirements, specify them as UML use case diagrams, and describe them in the structured natural language. For each use case, specify the use case name, participating actors, entry and exit conditions, typical flow of events, alternative flows of events, and exceptional scenarios if they apply. As an approximation, aim to include up to six important use cases that are fully specified. For reference, see UML Practice Exercises 1.

For full marks, you need to use a requirement engineering tool, such as one of the tools discussed in class (e.g., Borland Caliber, Jama), to specify the use cases or use case descriptions. To model the UML use case diagrams, you can use one of the UML modeling tools also mentioned in class (e.g., ArgoUML, Umlet, Gliffy). To demonstrate the tool usage, include its output as screenshots of the tool with the requirements entered into (e.g., using the Print Screen key), or produce a report from the tool (e.g., as XML if available) that you can attach in the appendices.

The key objective is to demonstrate that you have used a systematic approach to specify the requirements using techniques embedded into requirement engineering tools.

b. Architecture Modeling

Next, you are to perform subsystem decomposition, and specify an architectural model that best fits the requirements that you have specified above. You can use one of the techniques discussed in class, such as Model Driven Architecture (MDA) or Attribute-Driven Design (ADD) method, to derive the architecture. Identify the key concerns/drivers for the architecture (e.g., performance), and describe the identified drivers briefly. Note that Market Buddy is intended to run both as a web client and as a mobile app. For reference, see UML Practice Exercises 2 and sample solutions to Participation #3.

For full marks, you need to use a model-driven development tool, such as one of the tools discussed in class (e.g., ArgoUML, Umlet, Sparx Systems Enterprise Architect), to specify your software architecture. You need to provide a high-level UML package or component diagram, and decompositions of individual subsystems as additional UML package or component diagrams, to as much detail as available. If possible, you should specify the interfaces between subsystems. Note however that you do *not* need to provide a detailed design class diagram nor individual classes that would be included in that diagram, and you should complete your modeling at the level of subsystems and components.

Finally, select one of the important use case scenarios (e.g., executing a BUY order on TSE), and illustrate its execution using UML sequence or collaboration diagram. For the elements in the model, use the subsystems or components from your decomposition above.

The key objective is to demonstrate that you have used a systematic approach to derive your architecture, by describing the steps used to perform the decomposition, and that you have used a software modeling tool to specify the results of your decomposition in a systematic manner.

c. **Configuration Management**

Based on your analysis above, identify several important configuration baselines for production of the Market Buddy software (e.g., developmental baselines, functional baselines, production baselines). For each identified baseline, clearly define what needs to be accomplished. You may include baselines for the entire lifecycle, including implementation and deployment, but focus on requirements specification and architectural design activities. Also, specify important configuration items (CIs) that would be used during requirements specification and architectural design; these could be the same items that were discussed in the earlier parts of the assignment.

Finally, select a corresponding version control system, such as one of the systems discussed in class (e.g., Subversion/TortoiseSVN, Git/TortoiseGit, Mercurial/Bitbucket), and create a repository to store CIs that you have selected. Include at least one binary-encoded file (e.g., PDF) and one textual file, and demonstrate common version-control operations, such as add and commit, checkout, change and commit/checkin, remove and commit, and merge. To illustrate the operation of the tool, include its output as screenshots, or include a log file that shows how the commands were entered on the command line with the corresponding message after each command.

The key objective is to demonstrate that you have identified key baselines for requirements specification and architectural design of Market Buddy, and that you have demonstrated familiarity with version control systems and systematic usage of version control systems for configuration management of software.

Deliverables

A document submitted via LEARN that includes a solution for each of the assignment questions.

You are permitted to work in groups of up to three students each.

The assignment is due Fri Oct 25th by 23:55pm. Late submissions will not be accepted.