

Homework #2

Due February 21, 2007, in class

Please answer the following questions using lucid, concise English language; use the electronic word processing system and graphics tools of your choice to produce high-quality output. At the due date, please turn in stapled hardcopies of your work, as well as an electronic copy via email. Scores for late assignments will be multiplied by 0.8 for each day late. Please remember that this is an individual assignment; refer to the class web pages for the definition of “individual”.

Please find a short list of open source tools for creating graphs etc. at <http://java-source.net/open-source/uml-modeling>; others are easy to locate via Google and other search engines.

1. Architecture Definition Languages

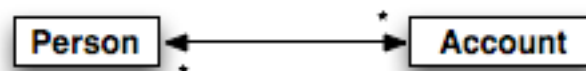
The goal of this exercise is to use an ADL to specify the core of the Model-View-Controller (MVC) design pattern. The following references will be useful in completing this task:

- [1] <http://java.sun.com/blueprints/patterns/MVC-detailed.html>
- [2] Robert Allen, David Garlan: [Formalizing Architectural Connection](#). Proceedings of the Sixteenth International Conference on Software Engineering, 1994
- [3] Ingolf H. Krüger, Reena Mathew: [Systematic Development and Exploration of Service-Oriented Software Architectures](#). Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA 2004), 2004.

1. Specify the resulting architecture captured by the MVC pattern [1] using Wright [2] and the Service-ADL [3] under the assumption that there are exactly two views.
2. Describe briefly what you would have to change in each ADL specification to go from two to an arbitrary number of views.

2. Class Diagrams / Implementation

Give a Java implementation of the class diagram shown below, such that the pertinent associations are properly managed; in particular, provide the necessary methods and data entities to keep the associations synchronized at runtime.



3. Class Diagrams / Domain Models

The ADL in [3] uses extended Message Sequence Charts (MSCs) to describe interaction patterns among roles. Develop a UML class diagram capturing the information contained in the following paragraph as closely as possible. Use textual notes to express constraints you cannot represent graphically.

There are two kinds of MSCs: Basic MSCs (BMS) and High-Level MSCs (HMSC). Each MSC has a unique name and contains an interaction specification. An interaction specification can consist of an individual message, which has a message name, a set of parameters, a set of sender roles and a set of receiver roles. You can think of a role as an “abstract” component. For each message one of the role sets must have the cardinality one (1). Each role referenced in a BMS can assume any one of a set of states over time. A second form of interaction specification is a reference to another MSC – you can think of it as a “macro” to be instantiated by the interaction pattern contained in the referenced MSC. Thus, an MSC reference carries the name of the referenced MSC. Interaction specifications can also be composed of an operator and one or more operands. Examples for operators are sequential, alternative and parallel composition, as well as loops. HMSCs and BMSs differ in that HMSCs consist of references and applications of the mentioned operators only.

4. Domain Models / Invariants

Give a class diagram modeling the central concepts of a typical auctioning system as found on popular Internet sites and in some video games. State at least two invariants about the model, in prose; attach each invariant as a note to your diagram and connect the relevant entities to the note.

5. Multiple vs. Single Inheritance

As is well known, Java supports single inheritance only for classes; it does support multiple inheritance for interfaces, however. Refactor the following class diagram into one that utilizes single inheritance only. In addition, provide Java code for the resulting class diagram.

