

CS578: Software Architectures Spring 2006

Homework #7

**Due Date: Tuesday, April 18
before Noon – *no exceptions***

Architecture Recovery

In this assignment, you are to recover the architecture of a software system starting at the source code. The system in question is an agent-based tracking tool (AGENT) integrated with a visualization tool (ANTS Visualizer) and a tool to controlling its environment (GTServer). The questions below will help you in the recovery process. Follow them step-by-step as they are incremental.

You are given the following input:

- Java source code of the ANTS Visualizer
- Java source code of the GTServer
- Java source code of the Instrumentation.java class of the Agent

NOTE: The agent is a complex piece of software with hundreds of classes. Its source code was thus excluded from this assignment. However, we did include the Instrumentation.java class of the Agent and you may consider it synonymous with the Agent.

HINT: The Java source code is packaged according to the three tools. You may assume that the Agent is a single component in your architecture. You should not assume that the other two tools correspond to components though they may.

NOTE: If you like to understand the purpose of the ANTS system, the visualizer, and GTServer (controlling the environment) then please read the PDF file at:
http://www.alexander-egyed.com/publications/Visualization_and_Debugging_Tools.html

1) (20%) Reverse Engineer the Source Code.

Provide a UML-style class diagram that contains all the classes of the three tools (Visualizer, GTServer, Agent) as well as all referenced Java JDK/SDK classes and interfaces. Also provide usage dependencies among the classes (i.e., if A calls B then there should be a dependency from A to B).

CAREFUL: Keep inheritance and polymorphism in mind. For example, if A calls an object that was declared to be of type B then the object could be an instance of B or an instance of any subtype of B. Accordingly, A either uses B or its subtypes!

2) (10%) Identify Classes according to DATA, PROCESSING, and CONNECTING

Every class in the class diagram serves a purpose. Some are primarily meant to hold data. Others do processing. Yet others support communication (connecting). It is at times hard to separate the classes into these three categories. But try to do so here. For each class, identify its primary purpose by choosing one of the three categories above.

Provide your answer in form of a table similar to the one below. The table should contain a row for every class, the propose of the class, and a brief rationale why you believe so.

HINT: you will have to look at the source code again.

Class name	Purpose	Rationale
A	PROCESSING	
B	CONNECTNG	
C	DATA	

3) (20%) Identify other Forms of Communication

Aside of the “uses dependencies”, can you identify other forms of communication?

Recall the class lecture where we discussed the very diverse forms of communication.

For every communication you identified, list what form of communication it is, between which classes the communication happens (directionality), and why you believe so.

Form of communication	From Class	To Class	Rationale

4) (20%) Component Diagram with explicit Connectors

Draw a component diagram and include connectors as needed. Try to make the components as coarse-grained as possible (the fewer the better) but make sure to include all forms of communication. Describe your architecture briefly. If connectors serve different purposes, make sure to label them.

HINT1: every form of communication in 3) should be considered a connector among components. Thus, you need to design the granularity of your components to fit the forms of communication.

HINT2: remain faithful to the DATA, PROCESSING, and CONNECTING labels identified in 2).

HINT3: a “uses dependency” may or may not be a connector (your choice)

5) (10%) Traceability

Provide the backward traceability between the component diagram and the Java classes in the following form:

Component name	Class name
----------------	------------

HINT: use unique component names. If the class name is not unique, you might want to include the package name also.

6) (20%) Architectural Style

Does the component diagram follow an architectural style? If yes, list it and describe how each component fits the style. If no, does the component diagram follow multiple architectural styles? If yes, list them and describe how each component fits these styles. If no, then describe why not.