15-413: Introduction to Software Engineering

Jonathan Aldrich

Assignment 8: Architecture & Hoare Logic

Due: Monday, November 7, 11:30am (hardcopy at beginning of class)
60 points

This assignment is a group assignment. Each project group should turn in one response to each part, with all the names of the group members.

1. Architectural Decomposition (20 points)

a) Provide a graphical architectural decomposition of your system into runtime components and connectors. To answer this question, you can use any graphical representation you want. You can use a boxes-and-lines diagram. However, make sure to include a legend. Follow the documentation best practices discussed in class. Note: in this question, you are asked to describe a feasible architecture that meets your project's requirements and that you can implement within a semester.

   Note that your architecture may have similarities to the module decomposition you defined in Assignment 7. Two differences you should pay attention to is that (1) a component and connector view of architecture shows run-time components; there may or may not be a one-to-one correspondence between run-time components (e.g. objects, processes, threads) and static modules (e.g. Java packages or C files). (2) a component and connector view of architecture includes a special focus on connectors, so you should pay attention to whether your system has conceptual connectors that go beyond a simple call-return module binding. E.g., look for connectors that denote network connections, shared files, pipes, a shared data store, a database, an event bus, etc.

b) Supplement your informal diagram with explanations. Use the format shown below. For each component, provide a table, listing:

Component:	This is the component name, e.g., Component A
Responsibilities:	This is a list of the component's responsibilities (that is, the functionality it implements). Each component should have at least one responsibility: • Responsibility 1 • Responsibility 2 • …
Properties:	This is a list of the component's architectural properties: e.g., ResponseTime <= 100ms. Annotate your architecture with properties that correspond to at least one quality attribute (see

	below).
Collaborators:	This is a list of the other components that this component either **provides** services to, or **requires services** from. Be specific about that. • Component B: requires x, y, z • Component C: provides u, v, w • Component D: instantiates this component
Rationale:	This is the reason as to why you chose to have this as a separate component. For example, it is on a different host than other components, or it hides some information that is likely to change.

c) Are there any other possible architectures? What made you decide on your solution over these other possible solutions? (E.g., discuss any trade-offs you made and why.)

2. Architectural Styles and Analyses (10 points)

a) From the architectural styles covered in class, which one most closely matches your architecture? If the system does not obey any particular style, explain the reasons why none of the canonical styles applies, or explain how your architecture uses elements from more than one style. Discuss the consequences or the tradeoffs of deviating from the closest canonical style.

b) Define at least two architectural constraints (invariants) on the architecture. For example, a structural constraint might restrict the number of instances, or the arity of a connection, or which components can talk to which other components. For each constraint, explain why it is important. You can use any predicate logic notation you want, as long as it's understandable. Briefly describe what notation you're using.

c) Consider three quality attributes and for each one, discuss whether and how the proposed architecture promotes it or detracts from it. Be specific. Note: Some of the quality attributes you choose in this question may not necessarily be requirements in your project (e.g., portability to a different platform).

d) Discuss briefly what kinds of architectural-level analyses would be helpful for the specific architecture that you described and under what circumstances. For example, check for deadlocks in a concurrent system (and under what conditions such a deadlock might happen).

3. Hoare Logic (15 points)

Use Hoare Logic to prove total correctness (i.e., including termination) of the following binary search code. Use the notations from class and the lecture slides. Show all of your work. You must explicitly state the loop invariant and variant function separately from the main body of your proof. Your proof must show all of the intermediate predicates and where they go in the source code. Finally, you must show how each proof obligation is discharged as a series of steps, and justify each step with a 2-3 word explanation (e.g. arithmetic simplification, by assumption, etc.) as was done in class.

```
{ $N > 0$ && $(\forall k \mid 1 \le k < N \bullet a[k-1] < a[k])$ }
i := 0
j := N-1
m := (i+j)/2
while ($i \neq j$) do
        if (a[m] < V)
                i := m+1
        else
                j := m
        m := (i+j)/2
end
{ $(\exists k \mid 0 \le k < N \bullet a[k]=V) \Rightarrow (a[m]=V)$ }
```

4. ESC/Java 2 (15 points)

Download ESC/Java 2 from:
http://secure.ucd.ie/products/opensource/ESCJava2/download.html

Follow the instructions in README.release to install the system. You may have to install Java version 1.4 (Java 5.0 won't work) if you haven't already, you can find this at:
http://java.sun.com/j2se/1.4.2/download.html

If you are working from Windows, you will need to modify escj.bat in the obvious way to include the appropriate paths for your system.

Now take the file Stack.java and StackCheck.java from the class website. Run ESC/Java 2 on both files together. Add pre- and post-conditions and invariants to Stack.java and fix any bugs you find in the code, until ESC/Java runs on both files without producing any warnings. You may *not* edit StackCheck.java. Nor may you remove the annotations that are already present in Stack.java.

Turn in a printout of your edited version of Stack.java. Also turn in a printout of ESC/Java 2's output when run on the two files.