## *What a Software Architect Does:*

1. Learns or negotiates the definition of a new product.
2. Roughs in a structure suitable to implement the product.
3. Thinks deeply about:
    a. user's interactions with the system
    b. potential problem areas:
        i. safety
        ii. performance
        iii. usability
        iv. complexity
4. Proposes and negotiates with customer:
    a. User interface
    b. Solutions to critical problems, especially where there is impact on the product's requirements
5. Creates prototypes in order to:
    a. Assess whether the proposed structure is feasible
    b. If not, then the architect will look for ways to cache data, find better algorithms, do less data transport, invent a different structure (repeating the above process), …
    c. Show designers one way to handle design problems
    d. Provide a base-line for discussing and managing the design of performance critical parts
        i. Shows one possible design solution
        ii. Provides a measurable performance benchmark
        iii. Gives designers a design goal and something concrete to complain about, e.g. your prototype didn't have to …., your design environment was easier because …, you didn't think about …
        iv. Without a prototype designers and Architect have no effective basis for communication until it is too late, e.g., after the designers are nearly done.
6. Sometimes it is necessary to negotiate relaxations of some performance specifications with the customer.
    a. Much better to do that early than later!!
    b. Prototypes will help convince customer much better than analysis alone.
7. As the design proceeds, the Architect monitors the design, creates new prototypes to test out new ideas where needed, and supports system test.

## *A Load Analysis Example:*

Project #5 – File check-in and check-out

Assumptions:

1. Medium size "large project", e.g., aerospace, operating system, CAD suite, … $\Longrightarrow$ 5,000,000 million lines of code $\Longrightarrow$ 300 developers.
2. One half (150) work on code on any given day. One third (50) will check-out a project to work on any given day

   $\Longrightarrow$ 25[1] files * avg 20 KB[2] each $\Longrightarrow$ approx one half MB[3] file transfer.
3. Most check-outs occur between 9:30 and 10:30 am (same for check-ins between 4:30 and 5:30 pm)
4. So file transfer rate is 50 developers * 0.5 MB in one hour $\Longrightarrow$ 25 MB / hr. Need to account for loss of efficiency due to contention for files.

Observations:

1. There is no problem with network bandwidth due to this load. The question is: can the server deliver this load while doing the other things necessary for check-out, e.g., user authentication and database queries with enough margin to support other server activities.
2. The interesting question is: How many projects of this size can the RSA support concurrently? - No healthy business runs on one project at a time. If the answer is a dozen or so, then we don't have a problem.
3. How do we tell if the server can handle this load, accounting for file contention? By building a prototype of the file transfer mechanism and testing. Then building a prototype of the database activities and testing.
4. Remember that server load is not acceptable unless the queuing load factor: arrival rate / service rate is less than about 0.25.
5. An architect will usually create a load budget, allocating so much for check-in, so much for white-board, so much for …

As you see, we make lots of assumptions about the load, to compute arrival rate. These are rough but sensible estimates, and could be off by 50 percent. You can not find the service rate by making assumptions. You can only discover a neighborhood for service rates for each of the load-intensive activities by **building a prototype and measuring**.

---

[1] Based on looking at some of my larger projects.
[2] KiloByte (KB)
[3] MegaByte (MB)

## *Sources of Load for Remote Software Assistant*[4]

1. Check-in and Check-out
2. Browsing documentation – so do on clients
3. Writing to change log
    a. Every check-in, check-out, build, and test-harness test is recorded
4. Testing – so do on clients
5. Code dependency analysis[5] used to detect dependencies for foreign code to be entered into RSA – so do on clients.
6. Collaboration support
    a. Chatting probably won't be a problem
    b. White boards support remote communication regarding documents, diagrams, and source code, by displaying them and supporting annotations on those documents. Managing a whiteboard can be a major source of load, probably the most interesting source for the RSA, should you choose to include this component.
7. You may think of others.

You are not required to analyze all of these. Simply take one or two and show that you know how to do a performance analysis by developing a load model to determine message arrival rates and message sizes, then building a prototype to estimate a potential service rate.

[4] Assume that all builds are done on client machines, not the RSA server.

[5] This is a major computational task for large projects, but you don't have the tools to prepare a prototype. Therefore, you should mention this but you are advised not to attempt a prototype for this.