Architecture and Design Document

This document describes the contents of an architecture and design document.

The “architecture” of your system provides a high-level view of your system. It gives the reader a broad overview of what parts your system has and how they interact. It does not delve into the specifics on how the individual components work unless the component is itself a large-scale part of the system with complex internal structure. Depending on the nature of your system, a good way to display the architecture is with one or more a block diagrams.

The “design” of your system is a medium to low-level view of how your system is structured. It explodes the individual blocks of the architecture and shows details on each block’s internals. How this is done depends on the nature of the block. In the case of software, there might be class diagrams, state charts, activity diagrams, etc. In the case of hardware, there might be schematics, perhaps with some details left unspecified, state charts, approximate timing diagrams, etc. Mechanical parts, if any, should have dimensions and materials specified.

Both architecture and design also should be accompanied by a textual narrative describing the information presented in whatever diagrams are used. The diagrams themselves are insufficient. This is an important point.

In simple systems, the architecture and design can be merged into a single presentation. In general, the boundary between the two levels is often vague and ill-defined. It is a matter of judgement to decide what should be presented at which level.

Be sure the include the following items:

1. *Group Membership*. Include the name and contact information (email address) of every member of your group. You should also include a name on each section of the document to indicate who in your group is the primary author of that section.
2. *Specifics of the User Interface*. The user interface is normally specified in the requirements (if this is not true for you, this is a good time to edit the requirements accordingly). However, some of the details of the user interface can be left for the architecture and design document. Be sure that somewhere (requirements or architecture and design) you discuss: prompts, menus, window layouts, command line options, input and output formats, and the hardware interface.
3. *System Architecture*. This is where you describe the major blocks of the system and how they interact. Be sure to describe both software and hardware blocks and discuss the hardware/software division you used and why, if appropriate. Depending on the complexity of the project your system architecture should probably contain between four and ten blocks. If it contains fewer blocks you probably haven't decomposed the problem enough. If it contains too many blocks you are probably providing too much low-level detail for an architecture.
4. *Identified Risks*. This is a list of “risks” you've identified with your project. A risk is some aspect of the problem that you don't understand how to solve. It would also be appropriate to comment on how you plan to mitigate your risks. Your list should

include all the risks that are relevant at the start of your project even if you have mitigated some of them by the time you write and submit your architecture and design document.

5. *Problem Partition*. This is where you specify who is working on what aspects of your project. For example, you might assign a name to each of the blocks in the system architecture. Note that you are not committing to this partitioning. Later, as you learn more about the problem you are solving, you can modify who works on which part of the project as appropriate. The architecture and design document, like all project documents, grows and evolves as the project unfolds.
6. *Design Details*. This is where you provide the detailed design information. The information should be detailed enough to make it "obvious" how the system will be constructed. You should expect to edit this portion of the architecture and design regularly as you work on your project and get a clearer idea of the problems you encounter. The design details should be written so that another team could, in the future, read those details and understand how your system works clearly enough to continue the work on it without having to spend time "decoding" the system's structure.