CIS 422/522 Fall 2013
Exercise: Independent Verification and Validation

Where software qualities are mission critical (e.g., safety, reliability, etc.) many organizations will use a Software Engineering techniques called "independed verification and validation." All this means is that verification and validation activities are carried out by a group that is independent of the development group. For example, NASA has a completely separate test and validation organization in West Virginia (http://www.nasa.gov/centers/ivv/home/index.html#.UmVe4iQd48g) that provides this service to other NASA centers and contractores. The idea is that an independent V&V organizaiton will be in a better position to test and validate software without prejudice or favor.

This exercise will simulate IV&V by having each team verify and validate the development artifacts of another team. The goal of this independent assessment will be to answer the questions:

1. Does the delivered software meet all of the specified requirements?
2. Will the software satisfy customer needs?
3. Did the team follow a disciplined Software Engineering process?

Note that the goal is not to make a "good"/"bad" judgement but to point out objectively where there are issues in the products or process. For example, specific places where test results are inconsistent with requirements, inconsistencies between documents, inability to find information, etc. This objective critique provides information the development team can use to improve their process.

The IV&V team should approach the evaluation by dividing the work. For example, have two or more team members take a subset of the requirements to test. Have different team members each look at specific parts of the documentation to check for consistency, completeness etc.

In a bit more detail, seek to answer the questions above through testing and review:

1. Does the delivered software meet all of the specified requirements?
    a. Identify each of the functional and quality requirements in the SRS.
    b. Identify any missing functiona or quality requirements.
    c. Run test cases for functional requirements.
    d. Check run-time properties like performance.
    e. Review the design and code for properties like understandability and ease of change. Check if you can trace requirements into the code to identify where specific changes would occur (maintainability).
2. Will the software satisfy customer needs?
    a. Evaluate as a customer/user.
    b. Note specific issues, defects, unexpected behavior, etc.
3. Did the team follow a disciplined Software Engineering process?
    a. Is it clear how the development proceeds from an understanding of requirements into design, then code?
    b. Is it possible to trace requirements through the process using the documentations?
    c. Did the team follow their own process effectively?