

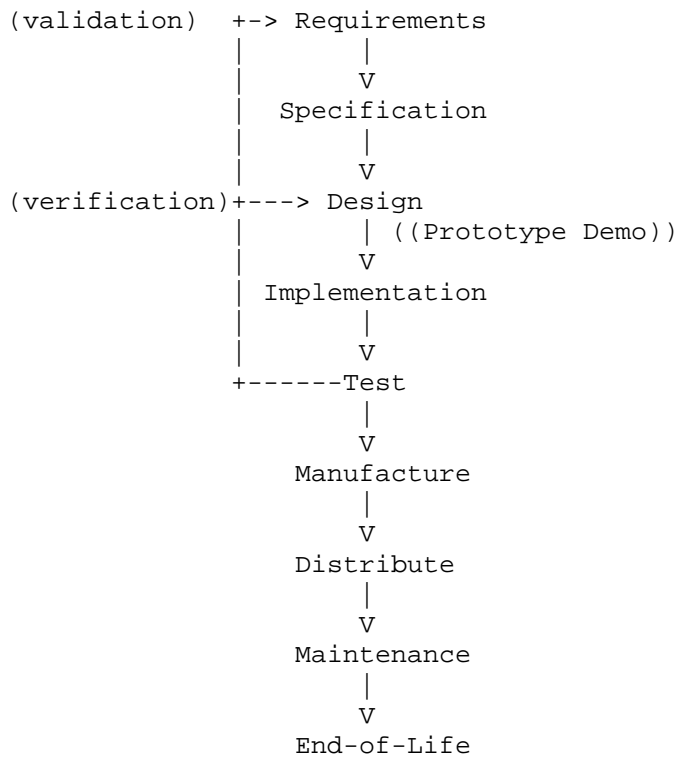
>>> SOLUTION <<<

Welcome to your 10% mini-exam for Senior Project. You have 75 minutes for the exam. The exam is open book and notes, but you may not receive or give help from or to another person during the exam. You may have with you anything on paper (your notes, books, copies of the old exam solutions from the course website, etc.). You may not use any electronic aids (e.g., a laptop WiFi'ed to the Internet is not legal). If you copy your answer directly out of the textbook (or some other source), be sure to properly "quote" the answer (otherwise it is assumed that any answer is given in your own words). Read the last sentence again. There are seven questions each worth 14 points and one extra credit question worth 10 points (you get 2 points for following the instruction in the next sentence). Please use a separate sheet of paper for each question, do not write on the back of any sheet of paper, and submit your exam with this cover sheet as the topmost sheet and the problem sheets following in number order.

Problem #1

Answer the following questions about the product development process...

a) Sketch the product development process as we have discussed and followed it in this class.



(yup... you can copy this for pretty much all previous old tests. Just be copying it, maybe you learned it just a little bit better!)

b) Why do you not (or only rarely) describe implementation in the requirements document?

Requirements is "all about" describing the user's problem, not the solution. The requirements document develops a complete understanding of the problem. Only very rarely does a problem dictate a specific implementation. At the earliest, implementation comes into consideration in the specification/design document (and sometimes not even there).

c) Why is there no traceability matrix in a requirements document?

There is nothing to trace back to in the requirements document. In the specification, we trace back to the requirements. In the test plan we trace back to both the requirements and specification.

d) The specification defines both syntax and semantics of the solution. What is syntax? Give an example. What is semantics? Give an example.

Syntax is what something looks like. Semantics is how something works or what it means. Some examples... The fields in a network packet are syntax, the function of these fields is semantics. The layout of buttons on a user screen is syntax, the function of the buttons is semantics.

e) Why might it make sense to define the test plan immediately following the requirements?

Writing a test plan early (rather than later) might better enable the design and implementation to be more testable. This early test plan approach might improve the overall quality of the final developed system by forcing the developers to think about how it will be tested when designing and implementing it.

f) Why must a test case be executable by someone with zero knowledge of the project? That is, why must all inputs and outputs be clearly defined and described?

Typically, the tester will be someone outside of the development organization and may also be a new-hire employee with little experience. Also, it is desirable to automate test cases (i.e., have a computer enter the input and check the output). To automate test cases there can be nothing left to human judgment, all inputs and outputs must be explicitly defined so that a computer can "read" them.

Problem #2

Answer the following questions about design...

a) Why is the design process (and for that matter the entire development process) iterative?

As you progress down the development process you (and the customer) learn more about 1) what is really needed (i.e., what really is the problem) and 2) how to solve the problem. This learned information (or new knowledge) is fed back to improve the requirements, specification/design, and implementation.

b) We discussed that design is “all about” trade-off. Give at least three examples of trade-off in a software project.

The classic example was the axioms of development “Pick 2 - fast, good, cheap”. In a software project there can be trade-offs between:

- User friendliness and capability (a more user friendly product may have less capabilities due to less options presented to the user)
- Time to market and reliability (a sooner to the market product may be less reliable due to less testing)
- Portability and performance (a more portable program may execute slower due to the need for using an interpreted and not compiled language)

And many others.

Problem #3

When are “men and months” completely interchangeable (by Brooks)? Be precise.

The mythical man month is that people and months are interchangeable in the time needed to complete a task. More people always equals less time to complete a task is only true if there is no communication needed (between the people doing the task). Communication is overhead. Since most tasks of any intellectual significance require communication, people and months are only rarely interchangeable. See page 16 of MMM.

Problem #4

According to Brooks a silver bullet is a single development that can yield a one order of magnitude improvement in software development productivity, reliability, and/or simplicity. Brooks states that, “There is inherently no silver bullet”. Explain this statement.

Brooks emphatically states (page 182 of MMM), “I believe that the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. ... If this is true, building software will always be hard. There is inherently no silver bullet.” That is, no tool or new process can reduce complexity of the essence of software development - the requirements and design phases.

Problem #5

In this course I have preached incremental development. It isn't my new idea... I learned it from Brooks (and from my many years of practice). What are the advantages that Brooks sees in incremental development? Give an example of incremental development (you may use Brooks' example).

Incremental development results in a working system at all times and thus:

- Testing can begin early
- The delivery date can always be met (albeit at possible cost in reduced functionality)

An example of incremental development is to build the main program loop first and make all subroutines (functions) stubs. The program then goes “round and round” but does nothing. The stubs are then filled-in adding functionality one step at a time.

Problem #6

What information is needed in the user documentation for a program (according to Brooks)? A list of the needed items is sufficient.

The needed information is purpose, environment, domain and range, functions realized and algorithms used, input-output formats, operating instructions, options, running time, and accuracy and checking. See page 165 of MMM.

Problem #7

Describe three non-trivial things that you learned from our guest talks. A non-trivial thing is something that is not obvious and is otherwise useful or helpful to you. For example, "Raytheon makes missiles" is trivial.

No specific solution

Extra Credit

Describe at least one thing that can be improved in this class that would not require large amounts of resources to implement. Be clear in your suggested improvement. Saying "Get better projects" is not clear.

No specific solution

A lawyer joke to leave you with a (hopefully) a laugh...

An engineer, a physicist, and a lawyer were being interviewed for a position as chief executive officer of a large corporation. The engineer was interviewed first, and was asked a long list of questions, ending with: "How much is two plus two?" The engineer excused himself, and made a series of measurements and calculations before returning to the boardroom and announcing, "Four."

The physicist was next interviewed, and was asked the same questions. Before answering the last question, he excused himself, made for the library, and did a great deal of research. After a consultation with the United States Bureau of Standards and many calculations, he also announced, "Four."

The lawyer was interviewed last, and was asked the same questions. At the end of his interview, before answering the last question, he drew all the shades in the room, looked outside the door to see if anyone was there, checked the telephone for listening devices, and asked, "How much do you want it to be?"