Department of Computer Science

College of Science and Mathematics
Montclair State University

MONTCLAIR
STATE
UNIVERSITY

Course: CMPT 515 Software Engineering & Reliability– Fall 2022 Dr. H. Johnson

Project: A Voice-Activated Undergraduate Computer Science Advising System

This product **allows a student whose major or minor is Computer Science or IT to obtain automated advising from the system pertaining to the appropriate course which needs to be taken and which courses the student permitted to take at a particular juncture during the pursuit of the degree.**

Project Due Dates:

Week 1 (9/12): Team organization. Submit team name and list of team members

Week 3 (9/26): Turn in a draft of the project requirements/specification (Draft of the project described here), stating/illustrating your understanding of the functionalities of this system. Include the name of your team, a list of the names of the members, and a list of possible meeting times for your team.

Week 5 (10/10) Improved/enhanced version of the project-document submitted thus far..

The main purpose of the functional specification is to explain what goals you are going to achieve for your Project, as opposed to how you will accomplish these goals. The functional specification, coupled with a management plan (which we will not write up formally) and a design (to be dealt with later), describe the proposal to your client. When accepted, these documents are a contract between you and your client about what you plan to deliver (and what he/she agrees to accept). Changes can be negotiated later and should be recorded as an appendix to this document. In writing this document, therefore, you should keep in mind that your audience will be a client (who in general is not an expert in computer science/software engineering) and your team members, who will want to refer back to this document particularly to be reminded of what they are supposed to be doing. The fact is that, in this course your client is just a user, and that your documents will be graded should encourage you to think and write as carefully as possible.

Requirements Documents

Software requirement document provides an important roadmap of the product to be built, the required features that to be included, and much more.

This roadmap helps to keep the technical and non-technical team on the same wavelength as to what the expectations are. It helps to ensure that the product is built meeting the needs of users and/or clients.

The requirements document provides a comprehensive explanation of what is needed for a product. This document will need to fully develop the context around a product and what it must look like to help developers implement their work.

The analyst will later need to categorize and organize the requirements. There is the need to find a way to neatly fit each of these categories of requirements into a single document.

Requirements Analysis

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. It involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore, requirements analysis means to analyze, document, validate and manage software or system requirements.

This involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, and avoidance of feature creep.

The Requirements Specification

Product Scope

Addresses the need for the product. It discusses the specific problems that this product will attempt to solve. Presents any domain-specific background necessary to understand the problem and includes measurable statement(s) of the intended benefits or objectives to be achieved by developing this product.

Product Features

Give a brief summary (could be a bullet list) of the major features that the software is expected to perform.

User characteristics

Describe general characteristics of the intended user of the product, including age, education level, specific experience, and technical expertise.

Constraints

Describe any constraint that may limit the choices available to the developers, such as regulatory policies, intellectual property restrictions, high level language requirements, etc.

Assumptions and Dependencies

List any factors or dependencies that the developers may assume will exist that may affect the software product.

Week 6 (10/17): Realization and Activity Diagram

Content of a Specification

Title page

<u>This, as well as all other assignments throughout the semester,</u> should be turned in with a title cover page that includes the following information:

- A title that identifies the document
- The name of your team (such as "Get Rich Quick, Inc.")
- Course name, Instructor's name, Date, etc.
- The team members who contributed to the document

Summary

In about two pages, you should briefly summarize the project you are working on. Give your system a name. Describe what it does and who will use it. What needs will your system satisfy? How will it help the users? Outline the most important features of your system. Describe the physical environment in which your system will be used, including any other system with which the new system will interface .Are there any important performance goals for your system: time or space efficiency, security, or reliability?

Details concerning system user interactions

The main point of this section (which should be approximately ten pages) is to describe the functions that the system will perform from the point of view of the system user. You need to cover the kinds of inputs your system expects, the actions it will take on both expected and unexpected inputs, and the types of outputs that the user will see in those cases. Part of this section will eventually be developed into a user manual. First we'll consider the content, then the form, of these descriptions.

The inputs, state changes, and outputs should be described in reasonable detail. You need not stick to the exact wording of messages, but you should make definite statements. You can negotiate major changes later. Describe what legal values or ranges your system will accept for inputs, what precision or accuracy constraints you will follow, and how you will handle errors.

Your description should employ some of the following techniques:

> *Sample transcript*. A very important part of the description is a sample interaction with the system in the form of a transcript of a dialogue between a user and the system. Use upper and lower case or some similar convention to distinguish between what the system outputs (types) and what the use types.

Feasibility

Make sure your project has a chance of being completed in a semester. Give your preliminary thoughts on how you will break down the different features of the project. What are the major classes and functionality and the relationships between them?

It is important to decide what this version of your system will involve. Describe the composite of your skeletal system. What functions will not be included, and why? Make sure that reading between the lines doesn't make your customer think that more is being promised than you plan to deliver. Now spell out what features will be added as time permits. Organize features into 'packages' that could be added independently or in some predictable order. Also think about the order in which components of the system can be dropped so that you can retreat gracefully if necessary.

Summary

Do not drop the reader off a cliff at the end of your paper. It's been a long time since the beginning of the paper. Restate the main points you want remembered. Note (write down) the authors of individual sections, the editor of the whole paper, outside consultants, etc. (This must be done for all future papers as well) This will help other people in knowing who to ask for more details.

Optional Section

A real specification would include a number of items that you may wish to skip at this point. Some of these require (more) experience in making predictions... but use your judgment.- Make some general statements about the performance goals for your system. What are your goals for system run time, main and disk memory use? What kind of reliability will you guarantee? Security information? What kind of performance trade-offs have you decided to make?

- Can you say anything about compatibility with existing software or hardware? What about the installation agreement? Maintenance contract?

- What resources are to be committed to the project? Who are the people on the project? Their skills and background? What is the promised delivery date? How much computer time and space will be used in developing the project?
- What publications will be produced? Who are the intended audience?

Design Document

Week 8 (10/31): Detailed design specification due (2 copies) (major grade) module drivers for kernel due (compiled but not tested) reapportion and review remaining tasks in section.

The design process

To help you along with the design a few milestones have been established for you. During the design process you should focus on the components of the design essential to producing a core system and merely outline extensions to a more complete system. One of your major goals is to have something worthwhile by your deadline, even if it isn’t the full system. Hence, you should determine what needs to go into a skeletal system and schedule the design of those components first.

The end goal in the design process is a detailed design document that is complete enough to be a reference from which any competent [software engineer] computer scientist can produce code, test plans, or a user manual. (This document should eventually evolve into the code and system maintainers’ guide.)

An intermediate goal is the overall design specification. This can be considered a draft of the detailed design document with some of the details missing. Interfaces between modules should be clearly defined. Once the design has been written, everyone in the group should read the entire document to make sure they understand the design. Individuals should then be assigned responsibility for particular modules/components and should design these in more detail.

Each person should carefully review at least one other person’s section (and all sections should be reviewed by someone other than the author). Pick the one(s) with which your module has the strongest interaction(s). Based on the examination of the module interactions, you should prepare a set of interface definitions.

Finally, prepare the detailed design document. Most of the work will be done by individuals filling in the details of their modules, but you must review the document as a whole to make you are all using the same interface definitions and to make sure that everything has been covered and that the parts of the document hang together.

Writing the documents

Your goal is to provide a recursive (top down) description of your system in a form something like the following. For each level you should cover the following:

Abstract - What services does this program, component, etc. provide to outsiders?
Implementation Document - Are any special instructions needed for the system user; for the writer of other modules??
Design - What is the basic design of this component or routine? How are the design decisions reflected in the sub-modules or routines which make up the whole? How are the pieces combined to accomplish the main function described in the abstract?
Exports - What facilities does this (sub)module/component make available to other modules/components? What type declarations? What constraints? etc.
Imports - What functionality defined in other modules/components are used?
Input/Output - Make sure all necessary actions are specified.
Subparts - What are the names of the sub-modules, components, and/or data abstractions that make up the components (module or sub-module)?
Pre and Post Conditions - What are the pre and post conditions on the main routines in your module? What are the important invariants?
Error Handling - What is the range of legal values? What happens when other values are found? Test Cases - List all your fiendish ideas for cases that might break the system or module or routine. If you do a good job here, you will be more sure of having a good design and most of your test plan will be already written.
Concrete Implementation - For (sub)modules that are data abstractions, give the concrete representation (declaration) here; the access functions will be listed as subparts. For routines, this is where the code goes. Make clear what data structures are private to the module. Define any technical terms relevant to the module.
Side effects - Hopefully there aren't any, but if there are, you'd better make them explicit.
Miscellaneous - Just in case you think of anything else (for example, an estimate of how frequently various routines might be used, or what extension might be required to accomplish additional tasks, etc)

Obviously not all categories are needed at all levels.

Overall Design Document

For the overall design document, you should define your major data abstractions and modules, focusing on the abstract, design, and sub-modules, and on error handling and exports and importance of the major modules/components. If you are in doubt as to whether something belongs in the overall or detailed design, put it in the overall design with a note that you aren't sure. Fill in as many other details as possible - it will be to your advantage to see it all in writing and get some feedback. You should also include sections discussing the modification and the management questions. The document should be about 30-50 pages. **A table of content is a must for this document.**

Detailed Design Document

The detailed design document should include answers to all of the questions for as many levels as possible. Write an outline of the pseudo-code. Update the sections describing coding responsibilities. A Table of Contents is necessary in this document. If your document seems to be getting too long (more than 70 - 100 pages) consult me. **Good luck!!!!!**

Testing Plans Assignment

Week 10 (11/14) Initial draft of Test Plan

Week 11 (11/21): Top level code for main modules, tested with drivers and stubs, due

Week 12 (11/28) Final Test plan documents due

Overview

Testing is an important part of a software project. The software team must therefore carefully plan and document the order in which components are to be integrated and the order in which the individual modules are to be completed and tested in isolation. Testing and debugging methods must be agreed upon, documented, and eventually carried out. **The grade given here will be tentative. A final grade will be assigned after you have completed the testing phase.**

Your goal here is to use the test plan to convince the management (in this instance, **me**) that a feasible test plan has been designed and also to provide the project members with directions for the testing phase of the project. This assignment outlines some constraints on the test plans. Several stages of tests must be scheduled, and several testing procedures must be explored. Scheduling is required for unit tests, integration testing, "functional" testing, performance evaluation, and an acceptance test (the demonstration).
These tests must include techniques of walkthroughs, extensive logic testing, input/output testing, and optionally verification. The following sections give details of the contents and style of the test plan document, and the final section lists some reading material that may aid you in designing a better test plan.

Design the test plan

The test and evaluation plan should contain the following components: statement of objectives and success criteria, integration plan, test and evaluation methodologies, and responsibilities and schedules. These components are described in more details in what follows.

1. ***Objectives and Success criteria***: The test plan document should contain a statement of the overall testing objectives of the individual tests that are planned.

2. ***Integration Plan***: An important decision to be made about testing is the order in which modules are to be combined and therefore the order in which they will he tested individually. You must plan for individual component tests, the combination of components during integration testing, a functional test, and an acceptance test.

   The test plan document should describe and defend your integration method. The "clasp your hand and pray" (which is similar to the "big bang") method is not acceptable. Various methods, or a combination of methods of integration are acceptable if convincingly defended.

3. ***Testing Methodologies***: You should design a plan to test all of the components of your system. Your test plan document should name at least one module to which each [testing] technique is to be applied;

   You may decide the testing technique for the other modules later on. Each person on the software team is to perform at least one se of tests and turn in a couple of pages describing the process at a later date.

4. ***Responsibilities and Schedules***: The test plan should provide a schedule describing dates and responsibilities.

- First, determine an order in which to perform integration and functional tests. In scheduling, you should make use of the dependency graphs based on the external functions that the modules require.

- Next, this order should be used to determine the order in which the individual modules are to be tested. Determine the dates by which tests are to be completed and the individuals responsible for testing. Prepare a master test plan schedule and include it in the document.

- Finally, you should define a monitoring procedure to ensure that tests are designed and carried out on schedule. Someone will have to keep a record and report lack of compliance to the other team members. Similarly, there must be a procedure for reporting and correcting bugs.

Writing the document

The test plan document is to contain an introductory section that summarizes the whole document in a page or two and discussion sections that cover the plans in more detail. A total number of pages between 15 and 25 is about right.

Remember that this document is intended for several audiences. The document should be organized so that it is easy to find schedule summaries and monitoring plans and easy directions on their personal responsibilities. The test plan should not be very long; if you that the writing of the document is taking significantly longer than the design of the test plan, consult with me. Representative samples of your tests are to be turned in later (as scheduled in the test plan).

The introductory section of the test plan document, which should only be about three pages long, should include the following:

- Overall objective and success criteria
- Summary of the integration plan - a list of tests and dates and people responsible
- Summary of the module-to-test-technique mapping for the four required testing techniques
- Summary of the monitoring, reporting, and correcting procedures
- Proposed dates for submission of individual test reports

The discussion section should contain:

- Defense for the integration plan (about half a page)
- Details of the tests with objective and success criteria for each test or group of tests (if you proposed most of the tests in your design document, there should not be more than 20 entries in this list, with each entry less than one page long; if you did not have good tests in your design document however, you must include them here)
- Details of the tests with objectives and success criteria for each test or group of tests tests (if you proposed most of the tests in your design document, there should not be more than 20 entries in this list, with each entry less than one page long.
- Details of monitoring, reporting, and testing procedures (a page or so should be enough)

- Details of individual team member assignments (just a page or so, this is essentially a cross reference list)

Week 13 (12/5): **User Manual** Draft due

Week 14 (12/12). Demo/Presentation and Final Project submission (Subject to change)

Week 15 (12/19) Final Exam (Subject to change)

Documentation
This submitted document should be a self-contained description of how to use your system. A user manual should be **a polished, professional piece of technical prose** that a software company is proud to have

accompany one of its products (And this is **a handy, if not proud, accomplishment to show off at job interview**).

The document should have a structure that is evident to someone who is reading it straight through and someone looking for a particular topic or fact. A table of contents is required; and the organization that ot reflects carefully considered. An index and appendix might also be helpful.

Remember that the document should be completely self explanatory. Do not assume the reader has your specification. You may of course edit the sections of prose from your previous documents. Do not discuss any implementation unless it directly affects the user's interface with the system

The Presentation

At the end of the semester (Dec 19, 2022) you (**each team**) should prepare a presentation/demonstration (about 10 minutes of material) that exhibit the best features/functionalities of your system and allow about five minutes for questions and feedback from the audience. If your system has been properly designed, you can let the audience tell you what to input to show off your error handling and user help facilities. NOTE: Actual functional features of your system must be demonstrated

Final Project Evaluation

The purpose of this document is to evaluate your final project. The document provides an opportunity to step back and put things in perspective and appreciate how much you have accomplished. If your group does not agree on evaluation and recommendations, feel free to write individual versions.

Demo Description

Required Submission:
Note: On the day you give your presentation , each team is required to turn in

1. a folder/binder/bounded (in addition to a CD (for the Professor) containing all project-related matrerials) – a polished, professional piece of technical prose - containing:
   - the complete specification and design (workflow diagrams, use case realization, etc) for your system
   - a transcript of your test plan
   - test suites etc, along with the rest of the evaluation.
   - the code for your system
   - the user manual
   - the demo/presentation material

2. Along with the hard copy of the materials specified in (1) you must hand in a CD containing the code (and all the afore-mentioned) for the system along with instructions for loading and running the system (**user's manual!!!**)

3. On the day you do your presentation/demonstration, in addition to the professor's copy, each member of each team must have in his/her possession a complete set (an exact copy) of all the materials specified in (1) and (2) above for me to sign

NOTE: Submission of substandard final project, including documentation, resulting from failure to:
(i) Submit required submission, as specified according to the Due Dates, so appropriate feedback could be given
(ii) Incorporate appropriate and correct recommendations (grammar, tense, constructs, etc.) will result in substantial points deduction.

Recap

You should turn in the complete specification, design (workflow diagrams, use case realization, etc) for your system and a transcript of your test plan, test suites etc, along with the rest of the evaluation.

Assessment: “Functional” Performance

Prepare an assessment which discusses how well your project satisfies your original specification. What was added or subtracted? What were the reasons for the additions/subtractions? What advice would you give to the next group of students about developing specifications? Suggest improvements to the project description.

Now, think about the expected performance of your system and compare this to your earlier predicted performance (LOC, memory requirement, response time, calculation time, etc.). Can you come up with reasons for the discrepancies (if any)?

The Design Process

Were you satisfied with your design process? How helpful was the process of writing out the specifications? How useful were the use cases and activity diagrams? How would you go about the design process next time? What differences, if any, would you propose for the overall and detailed design assignments?

Testing

Did the order of integration that you chose seem to work out reasonably well? Did you stick to your test plan schedule? If not, why, and what would you change if you had to do it again? What method did you use for recording test results?
Was debugging easier or harder than you expected? Do you think that the design process significantly reduced the amount of time needed for debugging? Discuss the types of bugs you encountered. Were they limited to one component? Were they the result of errors in the interface specifications? How did you handle fixing bugs?

Management

What was the management structure within your group? Is this the structure you had originally planned? What problems (if any) did you have getting people to do their share of the work? What suggestions (if any) do you have on how this could have been scheduled better?

Compare the actual time spent designing, testing, debugging, and managing the project with your predicted times. Have you learned anything that will help you to make better predictions next time?
What suggestions would you give to future software engineering and reliability students?

The Real World (Software development Organization)

Discuss what you might have done differently if this were a project on which you worked as a member of a development employed by an organization. team How much additional work do you believe would be required to complete the project to the level described in the specification?
How much would you consider selling the system for? Do you think the system is worth its cost? Justify your response.

NOTE: I reserve the right to modify the course requirements, the schedule, and grade allocation.