Spencer Collins

8/19/2025

Assignment 1

Professor Frank Alvino

1. **Describe the software development process. Be brief but include pertinent details.**

The software development process is a structured approach to planning, building, testing, and deploying software applications. The basic life cycle is as follows:

1. Development Planning
   1. This is where a methodology is chosen (i.e., Waterfall, Agile, Spiral, etc).
   2. Users should be interviewed to determine needs that drive the development of the application (gathering requirements).
2. Design and Prototyping
   1. This is the step where the requirements and ideas are brought to life either in a Proof-of-Concept (PoC) or a Minimum Viable Product (MVP). A PoC allows the testing of a specific feature in the application. An MVP provides enough core functionality for users to test it and provide feedback to the developer(s).
   2. Adding both of these allows the developer(s) to focus on what is most important to the users, thus reducing risk of technical debt and scope creep.
3. Development
   1. This is where the full software application starts to be built. User feedback from the MVP testing is incorporated at this stage. This is also the step where the developer should be in communication with the client to ensure the application is on track and incremental changes are being added to the benefit of the client. This allows rapid software development that can meet the ever-changing requirements of the project.
4. Testing and Integration
   1. After development is done, the application is tested to ensure it functions as intended before deployment to production.
   2. Bugs are fixed before the application gets to the users.
   3. Different types of testing is performed to ensure all aspects of the application are working properly. Unit tests are developed in conjunction as each component is finished to test its functionality. Application security is tested to identify vulnerabilities that can allow breaches or attacks. Depending on the version of the software, it may also be regression tested to ensure that unchanged functionality is the same between versions. There are other types of testing performed like integration and performance testing.
5. Deployment
   1. The software application is deployed to production for users to interact with.
6. Maintenance and Support
   1. Software that is deployed is maintained to ensure compliance with security standards, update or remove deprecated features, add new features, etc. This enhances the users’ experience and prevents obsolescence.
7. **The following program has three separate errors, each of which would cause an infinite loop. As a member of the inspection team, you could save the programmer a lot of testing time by finding the errors during the inspection. Can you help?**

void Increment(int);

int main()

{

int count = 1;

while(count < 10)

cout << " The number after " << count; /\* Function Increment

Increment(count); adds 1 to count \*/

cout << " is " << count << endl;

return 0;

}

void Increment (int nextNumber)

// Increment the parameter by 1.

{

nextNumber++;

}

* 1. See Question-2.cpp in the “/Assignment-1-Code/Question-2” folder
  2. Run the “Question-2.exe” file to see output.

1. **Read Case Study: Fraction Class on page 52 and answer the following questions.**
   1. See “/Assignment-1-Code/Question-3” folder for the unaltered code (TestDriver.cpp, Implementation.cpp, frac.h) from the Case Study: Fraction Class starting on page 52.
2. **The solution to the Case Study did not consider negative fractions.**
   1. **How should a negative fraction be represented?**
      1. Numerator is negative, denominator is positive
   2. **Which of the member functions would have to be changed to represent negative fractions? What changes would be involved?**
      1. Initialize()
      2. IsNotProper()
      3. ConvertToProper()
      4. See the “Implementation.cpp” file in the “/Assignment-1-Code/Question-4” folder for the changes made to each function.
   3. **Rewrite the test plan to test for negative fractions.**
      1. See the “TestDriver.cpp” file in the “/Assignment-1-Code/Question-4” folder for the rewritten test plan.
   4. **One of the member functions in the Case Study needs an additional test. Which function is it, and what should the data be?**
      1. The function is ConvertToProper(). It needs to test if the denominator is 0 and the numerator > 0. Otherwise the program will throw an error or not output the result to the Output.txt file.
      2. See the “TestDriver.cpp” file in the “/Assignment-1-Code/Question-4” folder for the added unit test in the test plan.