Steffany Jane Warain

Professor Paul Nevill

ITS411 Software Engineering

February 8, 2023

Key areas of Software Development Life Cycle

The Software Development Life Cycle, or SDLC, is a process of developing software applications. It consists of activities from gathering all requirements, documenting all the demands for the software, conceptualizing, planning the “WHAT” of the software, designing the “HOW of the software application, coding, testing, deploying the finished product, and maintaining the software application after.

There are three generic models of software processes discussed in class: Waterfall Model, Incremental Model, and the Reuse-oriented software engineering.

- Waterfall Model is a traditional approach to software development and is a very common method used for personal or small projects. It follows a sequential process in which each phase must be completed first before moving onto the next phase making the process of developing a software application more linear. The disadvantage of this model is, it takes a long time even if it is just for small projects, when you encounter bugs, the way of correcting them is going back again to the previous stage or starting all over again, making it time consuming and not very efficient.

- Incremental Model - an iterative process where a software engineer breaks down their project into smaller sections, designating their priority, completing each section separately, testing for each section, and then combining them into a complete product. This method is used for medium to large projects and is a better alternative to the Waterfall Model. It is a more flexible, more efficient, and more productive method because it allows changes and can add incremental functionality for each section tested. Furthermore, because each section can be tested more frequently before putting each part together, it is faster to produce a product and secure the success of a project.

- Reuse-oriented software engineering is an approach that stresses the reuse of existing software components. It collects reusable components to reduce

software development projects' costs, time, and complexity. As a result, it is more efficient and cost-effective than creating new components from scratch. However, it also requires a comprehensive software architecture to ensure that components are reusable, maintainable, and extensible. The benefits of reuse-oriented software engineering include increased efficiency, faster development times, better scalability and maintainability, and lower cost. Usually, people who use this approach are software engineers who are experts and very knowledgeable of existing software components to ensure the successful reusing of components.