Tyler Chotikamars

The University of Arizona Global Campus

CST 301 Software Technology and Design

Pete Limon

8/23/2021

Software engineering includes all aspects of creating software including specification, development, validation, and evolution. Other key components of software engineering include being able to make programs with attributes that enable security, maintainability, dependability, efficiency, and acceptability. The responsibility of a software engineer is not limited to making sure that the functionality of their program works, but also extends into space where they must make sure what they're doing is ethical and secure. The waterfall model process takes the fundamental process activities of specification, development, validation, and evolution and represents them as a separate process with phases that include requirement specification, software design, implementation, and testing (Sommerville, 2016). Whereas the SDLC (system development life cycle) includes main steps such as initiation, development, implementation, maintenance, and disposal. While the SDLC process is different for everyone it is good for creating security systems that can detect potential threats, constraints, and requirements. Some fundamental activities within software engineering include system design, component design, component coding & testing, integration tests, requirement specification, and service. Prototyping is a version of a software system that can be used to demonstrate concepts, try out designs, and find problems and possible solutions. System prototypes allow potential users to see how well the system supports their work by letting them see any errors and omissions within the system requirements. It also helps by finding areas of strength and weakness in the software. They can accomplish this by using steps such as establishing the objective of the prototype, defining prototype functionality, develop the prototype, and evaluate the prototype. Each of these steps includes another part that dives deeper into the development of the prototype. Incremental delivery is an approach to software development that allows it to be built in increments that are then delivered to the customer. Within an incremental delivery process, the customer chooses

what is most important to them so that that need can be fulfilled then the rest of the program can be employed through increments. In this case, we can clarify what increments are required to complete the system. As new increments are finished being completed, they are integrated into existing increments to improve system functionality. An advantage to an incremental delivery system over prototypes is that customers can update their requirements and use the experience to make more informed decisions on what they want for their system. Another advantage is that customers do not have to wait until the entire system is done being created to gain value from the system. The incremental nature of the system allows it to be easily incorporated within a system. Since the most important services are delivered first within the system, they become the most tested within an incremental system, which means that users will have a better experience. A con to using an incremental delivery system is that in the case of replacing another system, some of the functions within the old system will not be available. Another downside of incremental systems is that requirements may not be defined early on which makes it hard to identify common fallacies. As the software exists within a system is must constantly evolve to remain useful. Software evolution allows more and more software to be incorporated into large, complex systems. Compared to hardware-software is much cheaper to develop. It would be much easier to have software evolve to meet the customer's new needs than it would be to create a new system, therefore software engineering is viewed as an evolutionary process. When a newer software is introduced to the market the testing process is often called beta testing. This allows developers to receive feedback on their products. For cases in which change requests are to fix problems within operational systems that must be tackled urgently, emergency system repairs are performed. Emergency system repairs must be completed as quickly as possible. As far as system structure is concerned, a quick and workable solution is chosen instead of the best solution. The

emergency repair process includes change request, source code analysis, source code modification, and delivery of the modified system. The pros of emergency system repairs include quickly repairing serious system faults and unexpected effects that disrupt normal operations. The cons to emergency system repairs include making the system documentation and code inconsistent because of how quickly changes are made.

Agile project management describes the development of a software system that is integrated into its environment. Having a scrum team allows the process of creating the software system and ensures that the requirements are fulfilled within a timeline. The scrum sprint cycle is the framework that allows the framework for the sprint cycle and for it to be implemented when developing software. The process starts by reviewing what work needs to be done, then selects items and plans the sprint. After the work to be done is reviewed it also goes through product backlog which acts as a safety net in case the sprint causes problems within the development. After the sprint is planned it is executed with the sprint being backlogged as a buffer. After the sprint is completed it either goes into the review or into potentially shippable software. If the sprint is reviewed again then it is reviewed by the scrum master and the scrum sprint cycle goes forth. This allows the integration of the development to happen within an agile project management method. With this method increments of updates can be pushed allowing customers to see the work that developers do due to the software and what features it has.

Within extreme programming, requirements are expressed as scenarios or user stories. To create these user stories, programmers work in pairs to develop tests for each task. These tests must be successful when new code is integrated into the system. Some practices that are used within extreme programming that changed agile development included things like incremental

development, customer involvement, and the idea that developers do not need to strain themselves to meet deadlines. The last practice involves the practicality of programmers developing and the workload involved. To lessen the workload things like working in pairs, collective ownership of the code, and maintaining a sustainable development process are placed. While projects grow the complexity of integrating software increases things such as distributed development, complex legacy environments, and regulatory compliance requirements must be accounted for. To achieve scaling agile methods into integration with plan-driven methods things like basic requirements, ownership, documentation, communication, and integration must be accounted for.

Requirements for a system must be planned early within the development process. Requirements can be functional or non-functional. Functional requirements include services that the system provides, how it reacts towards inputs and its behavior. Non-functional requirements include constraints on the services or functions offered by the system. System requirements are the requirements needed to perform the software system's functions, services, and operations. On the other hand, user requirements underly the purpose of the piece of software, it is equal to the utility that the software performs. System requirements can be documented loosely in the early stages of development and can become more detailed later whereas user requirements are outlined in their entirety within the planning phase. This is because we only need the system to be operational in terms of physical requirements, but user requirements cater to the need of stakeholders.

Elicitation and analysis describe the discovery of requirements by interacting with stakeholders (Sommerville, 2016). The conversion of these requirements into standard form is

known as specification and checking that the requirements define the system that the customer wants is validation. Requirement engineering is an iterative process therefore many requirements must be integrated into the ever-evolving software. Elicitation involves system and user requirements, while the specification is then integrated. Validation comes in the form of reviews, prototyping, and feasible study. With the spiral model, developers can figure out what stakeholders expect, how the system handles the software, and get feedback for already integrated sections of their software.

Open-source development is a software development approach where the source code of a software is published, and volunteers are invited to participate in the development process. With this being the case people have access to the source code allowing them to spot any potential threats within the program. With the power of the internet, there is a surplus of people who can contribute to any open-source project. They can do this by reporting and fixing bugs and by proposing new features and functionality. Some things to consider when choosing to proceed with open-source development are if the project itself should be open source and if the product that is being made can make use of open-source components. Some examples of open-source software include the Linux operating system that is used as a server system and the Apache web browser. Some other notable open-source products include java, the Eclipse IDE, android, and the mySQL database management system. Companies such as IBM and Oracle also support the development of open-source projects which is great because they can be cheap and reliable.

Software inspection can help and complement software testing by reviewing, analyzing, and checking the system requirements, design models, the program source code, and proposed

system tests. Inspections in some cases can be better than testing the program for interface errors.

Development testing includes all testing activities that are carried out by the team developing the system (Sommerville, 2016). The three stages of development testing include first unit testing.

1. Unit testing is where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.

2. Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing the component interfaces that provide access to the component functions.

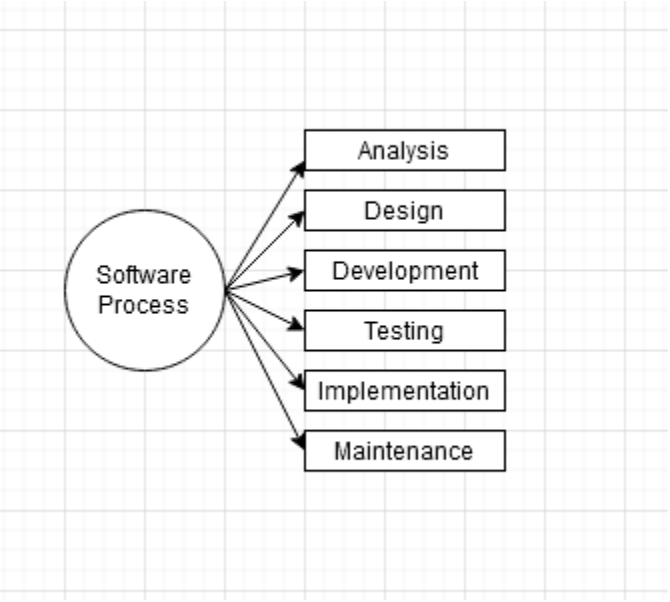
3. System testing, where some or all the components in a system are integrated, and the system is tested. System testing should focus on testing component interactions.

Unit testing is the process of testing program components, such as methods or object classes. This means that individual functions are tested to ensure they function properly within different routines and with different input parameters. Alpha testing can be defined as, “where a selected group of software users work closely with the development team to test early releases of the software” (Sommerville, 2016). Within this stage of development versions of alpha applications or software products are released to a controlled group of experienced users who can provide feedback. The agile development methods allow user involvement within the development process and allow users to play a key role in designing parts of the system. Beta testing can be defined as, “where a release of the software is made available to a larger group of users to allow them to experiment and to raise problems that they discover with the system developers” (Sommerville, 2016). Beta testing occurs when an early version or sometimes an unfinished

version of the program is released to a larger group of customers or users for evaluation. This is the version of the product that is usually released to the public for people who have been following the project. Acceptance testing is defined as, “where customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment” (Sommerville, 2016). This phase of user testing sees how the program works, using their own data to see if it should be accepted by the system developer. This is the final user testing phase and allows reviewers to produce a final opinion of the finished product. Software engineering will be used in the future for anything from blockchain development to website development. It is a fundamental part of information technologies that all professionals should be familiar with.

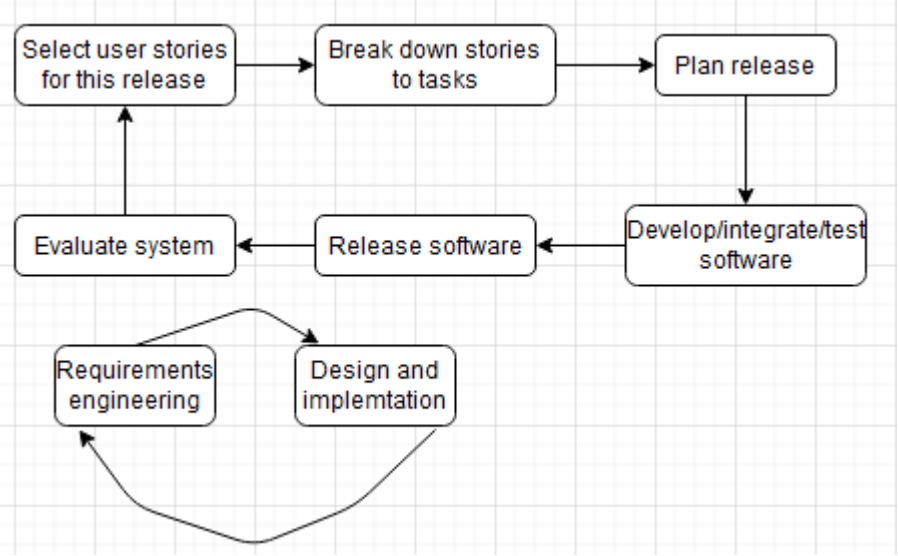
Concept Map 1:

The software starts with analysis where developers determine what the goal of the program development it. Then they move into the design phase where parts of the system are organized, the function of the product would be determined here as well. The development within the software process refers to the creation of the software product, a first draft of the product is created. Within testing there are many phases such as alpha and beta testing. Once a polished version of the product is created it can be implemented in the real world where it will receive maintenance in the form of updates.

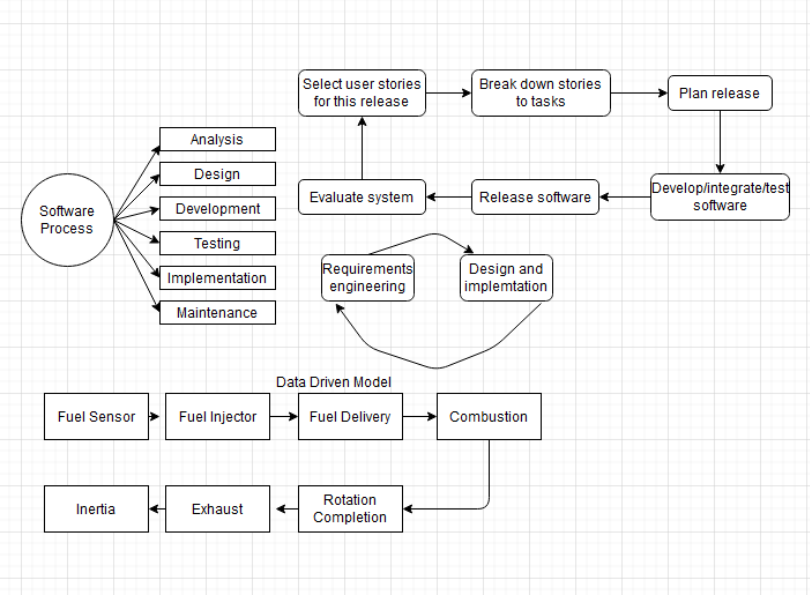


Concept Map 2:

The processes used for software development include creating user stories which are scenario's that user can experience using the alpha product. They are broken down and fixed to be integrated into a new update within the development cycle then re-evaluated. Software requirements engineering starts with defining requirements then doing design and implementation again until new requirements are needed.



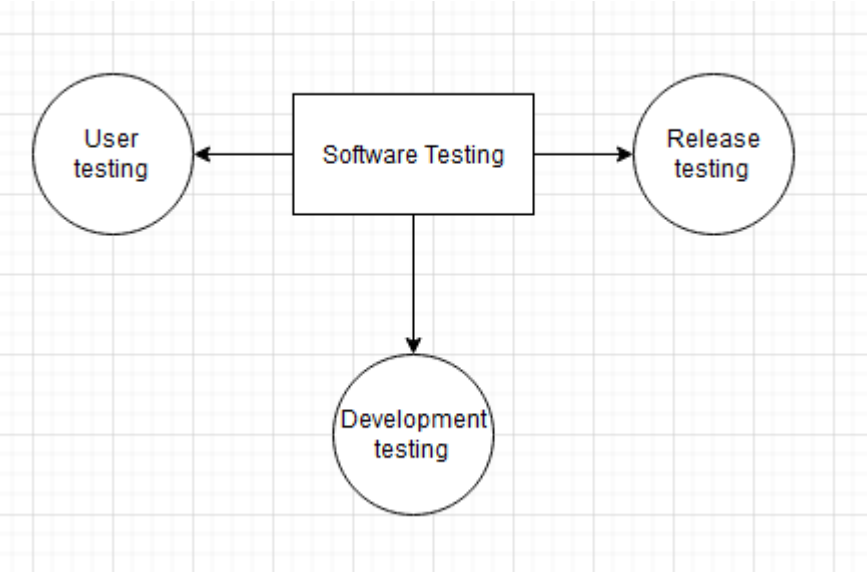
Concept Map 3:



Concept Map 4:

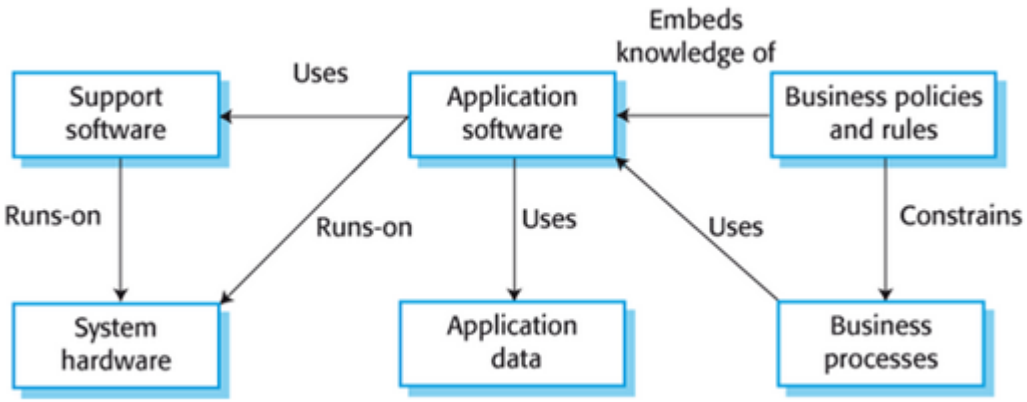
The three aspects of design and implementation in software engineering include the three stages of software testing which include development, user, and release testing. Within these stages the design and implementation of the product is tested to see which is best. Software reuse depends on the function the software provides but can be used at any stage of development.

Tested components can always be used when necessary and helps with fast development.



Concept Map 5:

Within a legacy system hardware may be limited but software must be updated for security purposes. Here is a figure of how the business proccesses interact with the application software and how that is effected by the system hardware and software support.



References:

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.

Pratap, M. (2010). *Software engineering.* New Age International.