Software Engineering Definitions

1. **Software Engineering** – the use of tools, techniques, procedures, and paradigms to enhance the quality of software products.
2. **Software** – A computer program that directs the operation of a computer to accomplish a specific task.
3. **Enterprise** **Software** – Software that is built for a specific business process
4. **Behavior** – Something the software does (function, use case, etc)
5. **Boundary** – Determines what is included in the development of the software product. Entities outside the boundary are not part of the development effort but produce input for the system and/or consume its output.
6. **Objects** – Data used by the activities (records, classes, fields, etc)
7. **Relationships** – Which objects are used in which activities
8. **Abstraction** – A simplification that allows focus on some pertinent aspect of the software (model, blueprint of a home showing electrical wiring)
9. **Modularization** – Divide and conquer; break entire behavior into independent, logical parts. (design/build/test/integrate)
10. **Activity** – A type of work performed during software development
11. **Resource** – Input consumed during process activities (labor, time, money)
12. **Stakeholders** – Anyone who benefits from the software being produced (i.e., has a stake in the success of the software; clients, investors, developers, end-user)
13. **Software Engineering Process** – A series of steps involving activities, constraints, resources that produce an intended output.
14. **Constraint** – A restriction imposed on the process (deadlines, a schedule, finances, target platform)
15. **Requirements** – What the system is supposed to do in terms of behavior, data, and constraints.
16. **Specifications** – Developer interpretation of what the system is supposed to do.
17. **Design** – Blueprints of how the system will be built
18. **Implementation** – The working software product
19. **Process Model** – A specific configuration of process activities that can guide real software development.
20. **Risk** – The chance that something bad will happen. An activities work product is not 100% correct.
21. **Validation** – Check if all requirements are accounted for. “Is this what the customer wants?”
22. **Verification** – Everything works correctly according to the specifications.
23. **Agility** – A software engineering characteristic that acknowledges change as a necessary part of software engineering and promotes efficient, appropriate response to change.
    1. Based on iterative software engineering process model
       1. (Iteration = Sprint)
24. **Requirement** – In general, an expression of desired behavior. Something the system should or should not do
25. **Requirements** **Engineer** **(RA)** – Developer who collects and analyzes requirements, produces and validations specifications
26. **Specification** – Precise description of what software is supposed to do
27. **Entity** – A very general term for something that interacts with the system or is acted upon
28. **Actor** – An entity external to the system that interacts with the system for the particular use/behavior to be described.
29. **Design Models** –
30. **Multiplicity/Cardinality** – Applies to association, aggregation, and composition
    1. **X** – exactly x number in relationship (i.e., 5)
    2. **X..Y** – x to y number in relationship (i.e., 0.. 5)
    3. **X**..\* - x to many (i.e., max is unbounded)
31. **Fault/Defect/Bug** – A human error while performing a software engineering activity
32. **Failure** – Incorrect software behavior due to a fault
33. **Repeatability** – Consistent products of desired quality
34. **Efficient Training** – Easier to train new people with a well-documented process
35. **Room for Improvement** – Easier to measure, analyze, and improve parts of the process to yield better results.