What are the qualities/attributes of a well-engineered software system?
Efficient,
solves the problem at hand, (Does exactly what it is supposed to)
Secure
Can it be Scaled - product works consistently well with small/large number of users
Reliability - always works as expected
Portable, concurrency
Easy to integrate new functionality
Availability - Downtime is minimal
Speed - performance - response times
Costs - maintainable software
- adapt to changing requirements
- Resource/energy saving - CPU/memory footprints are optimal
====================

How do we create a well-engineered software system?

Planned, Adaptable, **PROCESS** oriented, communicating often, iterative execution
?? proper documentation

DESIGN well, Modular, maintainable, reusable components,
Build vs Buy (reuse existing components instead of building everything from scratch)

TESTING
Bug free software (does it exist??? :-))

===========================
PROCESS - Agile frameworks - learn and practice
DESIGN - UML standards to express design, Design toolkit that can be applied to various problems
TESTING - Self-learn, Practice (projects)
===========================
Waterfall - assumption - requirements don't change

Multiple teams - sequential within a project, but teams can go out after a phase is complete
Handoffs from one phase to another

Negatives:
we cant go backward and make changes or do modification after
testing - it is expensive and can delay; - the other teams may not be available
Context switching
Possible blame-game
Doesn't work for changing requirements;
Late feedback; (due to long lifecycle);
Not flexible for customizations
Partial deployments not possible - testing POCs (prototypes)

Waterfall is good for systems with **Fixed requirements**; Software for a specific hardware;
Not good for changing requirements - not

easily adaptable