1 Correctness

Verification

- Verification in a system life cycle context is a set of activities that compares a product of the system life cycle against the required characteristics for that product.
- Checking if the system has been built correctly.
- **Static** => Type Safety, linting.
- **Dynamic** => Preformed during the execution of software, testing if your program works as intended.
  - **Small tests** $->$ Unit tests, testing of individual software components.
  - **Larger tests** $->$ Performed to expose defects in the interfaces and in the interactions between integrated components or systems.

Exceptions

- An exception is an action that disrupts the normal flow of a program.
- This action is often representative of an error being thrown.
- Exceptions are ways that we can elegantly recover from errors.

Coverage

- **Test Coverage**: a measure of how much of the feature set is covered with tests.
- **Code Coverage**: a measure of how much code is executed during testing.

2 Agile

- Philosophy and culture that are used to inform a range of different processes.
- **Waterfall method** - Requirements, Design, Implementation, Testing, Deployment & Maintenance

3 Full Stack

- **Standard Interfaces** - A universal method of connecting different systems together.
- **JSON** - A format made up of braces for objects, square brackets for arrays, where all non-numeric items must be wrapped in quotations. Similar to JS structures.
- **YAML** - Ease of editing and concise, indentation matters, dash used to begin a list item.
- **XML** - More verbose, demanding to process, more bytes to store.
- **Authentication**: Process of verifying the identity of a user.
- **Authorisation**: Process of determining an authenticated user's access privileges.

4 Continuous $\int$

- Practice of automating the integration of code changes from multiple contributors into a single software project.
- Planning $->$ Analysis $->$ Design $->$ Implementation $->$ Testing and $\int$ $->$ Maintenance
- Coding $->$ Build $->$ Test $->$ Report $->$ Merge $->$ Release
- **Continuous Delivery** - Allows accepted code changes to be deployed to customers quickly and sustainably. (By pressing button)
- **Continuous Deployment** - Allows changes to be deployed automatically as long as all tests pass.

5 Design

Maintainability

- Software in the real world changes over time. The less easily maintainable software is, the harder it is to adapt to these changes.
- Maintainable software resists the tendency to break as software changes or grows.
- To improve software maintainability, testing, system design and code design.
- One source of truth, DRY, KISS, Refactoring code etc...

Modelling

- Simplified representation to assist in understanding something more complex.
- Conceptual model captures a system
- State Diagrams

Complexity

- Essential / Accidental
- **Coupling** - Measure of how closely connected different software components are. Loose coupling is good.
- **Cohesion** - The degree to which elements of a module belong together. High cohesion is good.
- **Cyclomatic Complexity** Convert functions into a control flow graph, use formula $V(G) = e - n + 2$

6 Requirements

- A condition or capability needed by a user to solve a problem or achieve an objective
- Good requirements are the following: clear, concise, atomic, verifiable, attainable, abstract
- **Functional** specify a specific capability/service that the system should provide. It's what the system does.
- **Non-functional** place a constraint on how the system can achieve that. Performance characteristic.
- Elicitation, Analysis, Specification, Validation

User stuff

- User Stories
- User Acceptance Criteria
- Use Cases (list or diagram)

Validation

- Validation in a system life cycle context is a set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals and objectives.
- Right system has been built.
- Done by User Acceptance Testing, formal testing with respect to user needs.