SD Methodologies I (and OOP)

1 Waterfall model

- Plan vs course change
- The waterfall model is often considered out of date
- Not an inflexible process but often created an inflexible instance (process vs people)
- Planning allows for more oversight and control

When to use:

- Requirements very well documented, clear and fixed
- Product definition is stable
- Technology is understood and is not dynamic
- There are no ambiguous requirements
- Ample resources with required expertise are available to support the product
- The product is short

1.1 Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process
- Phases are processed and completed one at a time
- Works well for smaller projects where requirements are very well understood
- Clearly defined stages
- Well understood milestones
- Easy to arrange tasks
- Process and results are well documented
- Iteration occurs within activities

1.2 Disadvantages

- No working software is produced until late during the life cycle
- High amounts of risk and uncertainty
- Not a good model for complex and object-oriented project
- Poor model for long and ongoing projects
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model
- It is difficult to measure progress within stages
- Can't accommodate changing requirements
- Adjusting scope during the life cycle can end a project
- Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early

2 Agile

2.1 Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

2.2 Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivates individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective way of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity - the art of maximising the work not done - is essential
11. The best architectures, requirements and designs emerge from self-organising teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly