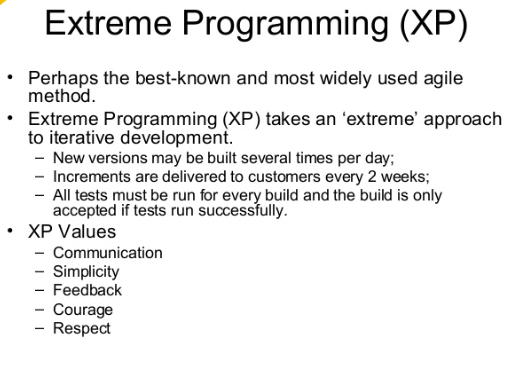
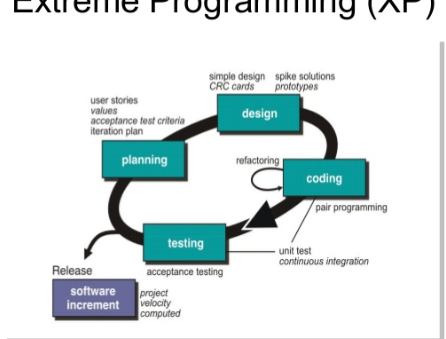


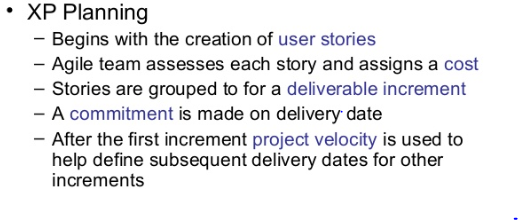
Principles from notes

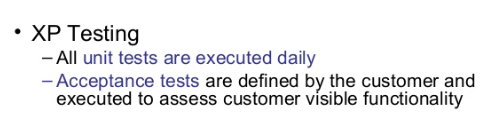
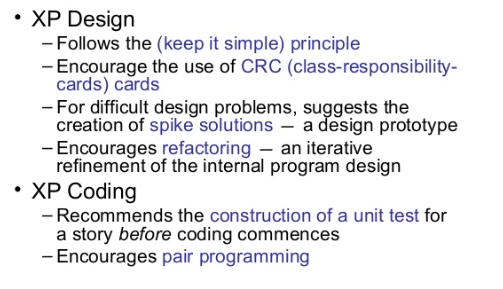
Agile process models:

1. Extreme programming
2. Adaptive software development
3. Scrum
4. Crystal
5. Feature driven development



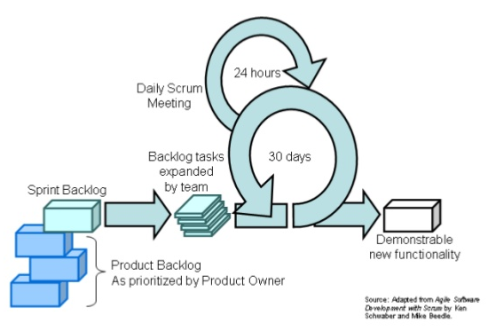






**Scrum** **model**

In the scrum model, a project is divided into small parts of work that can be incrementally developed and delivered over time boxes that are called sprints. The software therefore gets developed over a series of manageable chunks. Each sprint typically takes only a couple of weeks to complete. At the end of each sprint, stakeholders and team members meet to assess the progress made and the stakeholders suggest to the development team any changes needed to features that have already been developed and any overall improvements that they might feel necessary. In the scrum model, the team members assume three fundamental roles— software owner, scrum master, and team member. The software owner is responsible for communicating the customers vision of the software to the development team. The scrum master acts as a liaison between the software owner and the team, thereby facilitating the development work.



Adaptive software development replaces the traditional [waterfall](https://en.wikipedia.org/wiki/Waterfall_model) cycle with a repeating series of *speculate*, *collaborate*, and *learn*cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, [iterative](https://en.wikipedia.org/wiki/Iterative_and_incremental_development), [timeboxed](https://en.wikipedia.org/wiki/Timeboxing), risk driven, and change tolerant. As with RAD, ASD is also an antecedent to [agile software development](https://en.wikipedia.org/wiki/Agile_software_development).

The word *speculate* refers to the [paradox](https://en.wikipedia.org/wiki/Paradox) of planning – it is more likely to assume that all stakeholders are comparably wrong for certain aspects of the project’s mission, while trying to define it. During speculation, the project is initiated and adaptive cycle planning is conducted. Adaptive cycle planning uses project initiation information—the customer’s mission statement, project constraints (e.g., delivery dates or user descriptions), and basic requirements—to define the set of release cycles (software increments) that will be required for the project.

*Collaboration* refers to the efforts for balancing the work based on predictable parts of the environment (planning and guiding them) and adapting to the uncertain surrounding mix of changes caused by various factors, such as technology, requirements, stakeholders, software vendors. The *learning* cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During these iterations the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience and eventually mastery in the problem domain.[[1]](https://en.wikipedia.org/wiki/Adaptive_software_development#cite_note-1)