2. Extreme Programming

COMP1531 • KC Notes

2.1 Extreme Programming (XP)

- **Extreme Programming**: an agile methodology that:
  - Focuses on providing highest value for the customer fast
  - Change of requirements is **natural and inescapable**
  - **Adaptability over predictability**: requirements change, rather than defined at the start)
  - Lower the cost of change by introducing basic values, principles and practices
  - Useful for problem domains where requirements change, **mitigates risk and increases likelihood of success**
  - Emphasis on **testability**
- **Boehm's Cost Curve**: as software proceeds through the life-cycle, the cost of making a change becomes larger (error-free designs vs being prepared for change)

2.2 XP Values (CCSRF)

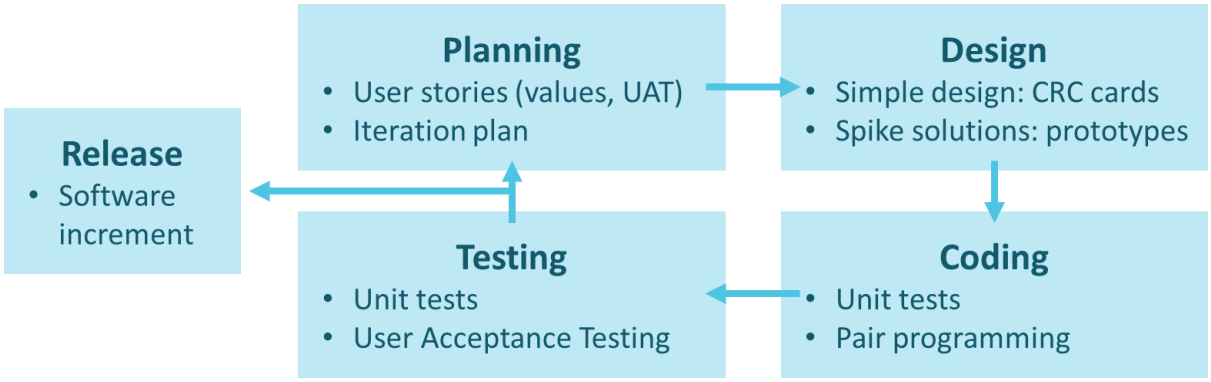
- **Communication**
  - Collaboration of customers and developers
  - Customer is a **member of the team** who defines and prioritises features
  - Rapid dissemination of information
- **Courage**
  - Design and code **for today**, not for tomorrow
  - Be comfortable with refactoring code
  - Tell truth, don't fear anything, take on problems proactively, integrate testing in development
- **Simplicity**
  - Focus on design and coding for needs of today
  - Future requirements are uncertain and carries risk of unnecessary spending
- **Respect**
  - Never commit changes that break compilation or fail unit-tests
  - Strive for high quality design
- **Feedback**
  - Feedback from **system**: unit tests, integration tests
  - Feedback from **customer**: user acceptance testing, reviews, feedback from team

2.3 XP Principles

1. **Whole team**: Managers, developers work as a single team. Customer or representative should be a member of the team with daily connection
2. **Continuous feedback**
    a. Feedback from **developers** (constant **testing** and continuous **integration**)
    b. Feedback from **XP team** (daily **stand-up meetings** on progress)
    c. Feedback from **customers** (**iteration demos**)
3. **User stories**: mnemonic token of ongoing conversation about a requirement
    a. Conversation to get a sense of requirements recorded as a description of a feature
4. **Short cycles**
    a. **Project Release plan**: **major delivery with prioritised user stories** for next few iterations, business determines which user stories to implement in the next release
    b. **Iteration plan**: **short cycles** to deliver a collection of user stories, business **cannot change priority/definition of user stories after iteration starts**
5. **High quality**
    a. **Pair programming**: code written in pairs, **one coding, one** reviewing
    b. **Continuous integration**: check in and integrate/merge code several times a day
    c. **Sustainable pace**: moderate, steady pace (marathon), no overtime
    d. **Open workspace**
    e. **Refactoring**: small transformations to improve structure, prevent software rot
    f. **Test-driven Development**: test coverage
        i. Unit testing: test independent modules, encourages decoupling and facilitates refactoring
        ii. User acceptance testing: customers describe features with an acceptance test. Tests are run every time system is integrated (regression testing)
6. **Simple design**
    a. Create the **simplest possible design for the current batch** of user stories (flat file vs database)
    b. **Resist temptation** of future infrastructure
    c. **Once and only once**: do not duplicate code
7. **System metaphor**: communicating the project **in terms that developers and customers will understand**, with **no need for familiarity** with the problem domain
    a. **Common vision**: easy to understand what it is and how a system works
    b. **Shared vocab**: common system of names
    c. **Architecture**: Identify key objects and aspects of interfaces, static and dynamic models of the system
    d. **Generativity**: metaphor can suggest new ideas for the system

2.4 XP Life-cycle

- The **process is constant and applied to the user stories** in sequence (rather than traditionally, where **requirements remain constant**)



- **Project tracking**: Record the results of each iteration (release) to predict the next iteration
  - Ensure work done is **sustainable and steady**
  - Use **velocity chart** to track project velocity – **number of story points completed**
  - **Burn-down chart** to show week-by-week progress – slope is a predictor of end date