**Development testing enables detecting and resolving bugs in the early stages of software development. This is much cheaper than doing that at acceptance test level.**

*Unit testing*
Bugs are inevitable. A safety net of unit tests helps catching defects as soon as possible. The quick, isolated and consistent feedback of unit tests gives team members insight in whether they are writing their code correctly.

*Pairing*
Two know more than one. A way to practice the value “Individuals and Interactions Over Processes and Tools” is to have team members work in pairs. This helps in reviewing each other’s work and in creating a shared understanding. Pairs need to change often, to enable maximal knowledge sharing.

*Test first approach*
When team members apply a test first approach, they combine the good practice of unit testing with collaboration and knowledge transfer. This makes a team more efficient in delivering working software.

# Levels and check points for development testing

The levels for *Development testing* are typified as follows:

- **Forming**: *Code is reviewed and unit tested*
- **Norming**: *Code is integration tested*
- **Performing**: *A test first approach is applied*

Please find the checkpoints below.

| Forming |
| --- |
| 1. Code is reviewed on quality by other team member(s)<br>2. Creating unit tests is part of the process<br>3. The developer verifies locally that the product still works before committing<br>4. Code review includes architectural aspects |
| **Norming** |
| 1. At least one other team member provides a functional review before committing<br>2. Pair programming is applied risk based<br>3. Automated (unit) integration tests are created<br>4. When checking in, a happy path scenario is provided as a basis for further testing |
| **Performing** |

1. A test first approach is applied (e.g. TDD, BDD, …)
2. Developers change pairing partners frequently
3. The quality of the tests is verified (mutation testing)
4. Test code is treated as production code (clean)

---

Terug naar I4Agile