**Algemeen over Coding Dojo and Coding Kata’s**

* What is a dojo? What is a coding dojo?
  + Improve coding skills
  + Learn!
* Practice skills you can use in production code
* Dojo exists of / elements:
  + Intro
    - Safe learning environment (try new stuff)
    - Explain dojo theory
      * Learning by practicing
      * Discussions
      * Giver is not the teacher, but the facilitator: all are students as well as teachers; encourage questions…
    - Fun environment
  + Agree activities
    - Review of last meeting
    - Topic theory for today
    - (agree on kata)
    - (agree on style)
  + Write tests as well as code
    - Simple program, no deadline
    - Facilitators role is to observe and make questions (about code) or answer questions: give praise! :)
    - Make suggestions instead of orders
  + Retro / Show you work
    - Reflect, review, discuss
    - Share / show code
    - Relate what you’ve learnt to everyday situations
    - Facilitator role:
      * keep time
      * get discussions going
      * show questions
      * when showing code: only positive feedback
* Practice coding skills:
  + Pair Programming
  + TDD
  + Refactoring
  + Design good test cases
  + Working in increments
  + Design principles (SOLID)
  + OO paradigm
  + Function Programming paradigm
* Dojo principles:
  + You can’t discuss a technique without code
  + You can’t show code without tests
  + Learn from each other, teach to one other!
* Wie? Alleen coders (writing code!)
* Min 5 max 15 persons
* Handling objections:
  + Code katas are just toy code: they don’t teach you anything (for real code)
    - It helps focus on a specific skill
  + TDD? I have better ways to produce good code
    - TDD is worth learning – a tool in your toolbox
    - Example of good developers (Robert Martin (Uncle Bob), Kent Beck, …)
    - Perhaps demonstrate with a kata?
  + TDD is too hard in production code
    - TDD is a complex skill that takes time and effort to master
    - Start TDD on easy problems to start with
    - A lot of the current code has poor design and is hard to test because TDD was nog used
  + TDD – too small steps! Too slow
    - Slower but less risky
    - Slower indeed? Better quality!! Less defects!
  + Other issues too hard to discuss 🡪 park them on a wall!