Programming Beyond Practices: be more then just a code monkey

- 1. Using prototypes to explore project ideas
  - Check assumptions early and often
    - set up test system as soon as possible to collect feedbacks
    - Design features to collect feedback easy
  - Focus on the risky or unknown parts
    - Limit the scope of your work as much as possible
    - Discuss all defects, but be pragmatic about repairs
    - Prototyping is about exploring a problem space, not building a finished product
- 2. Spottin hidden dependencies in increamental changes
  - Hidden depencies
    - Look out for the hidden dependecies in even the most simple updates.
    - Pay attention to shared resourcees that live outside your own codebase
  - make use of constraints and validations to help prevent local failures from causing global side effects
- 3. Patin points of service intergration
  - Be cautious when depending on an external service for something not well known ≡
  - make sure that at least some of your tests run against the real services you depend upon.
- 4. Rigorous approach toward probleming solvingf
  - get essential details of problems
  - divided into sub-problems
  - !! Assume input data is not clean ≡
- 5. Design a software from the bottom up
  - to start, look the shorted meaningful sentence that you can construct . use it as the guiding theme for the firtst feature you implement
  - as you continute ...., pay attention to the connections between objects
  - When extracting reusable objects, look for fundamental building blocks, rather than looking for superficial ways
- 6. Data modeling in an imperfect world
  - Preserve data in raw form, rather than attempting to transform it immediately into structures that closely map to domain-specific concepts
  - Design data management workflows that respect and support the organizational culture of people using your software
- 7. Gradual process imrovement as an antidote for overcommitment
  - disable or degrade features as needed to get you software back to a usable state as soon as possible, Fix it later
  - 20% of invested is responsible for 80% of the result obtain. Focus on high leverage
  - Reduce the projects in progress. unshipped code is not an asset
- 🚩 The intersting parts of programming to me have always been the problem-solving, communication, and human-centirc aspects of things; code was just the most effective tool i could find to serve those purposes ≡

试 用 版 XMind:ZEN 试 用 版