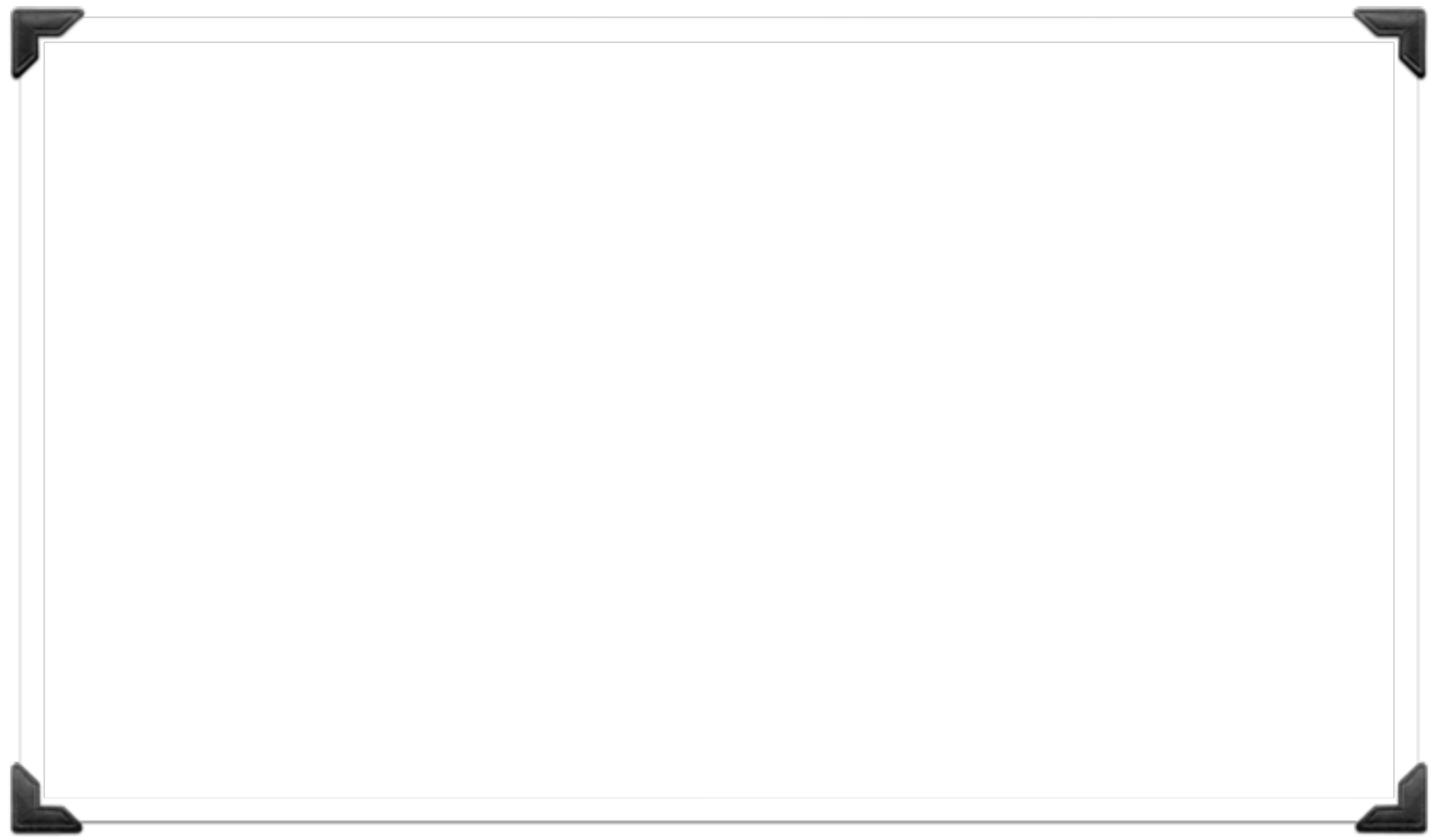
Planning vs Reacting

Designing and Maintaining Software (DAMS)

Louis Rose

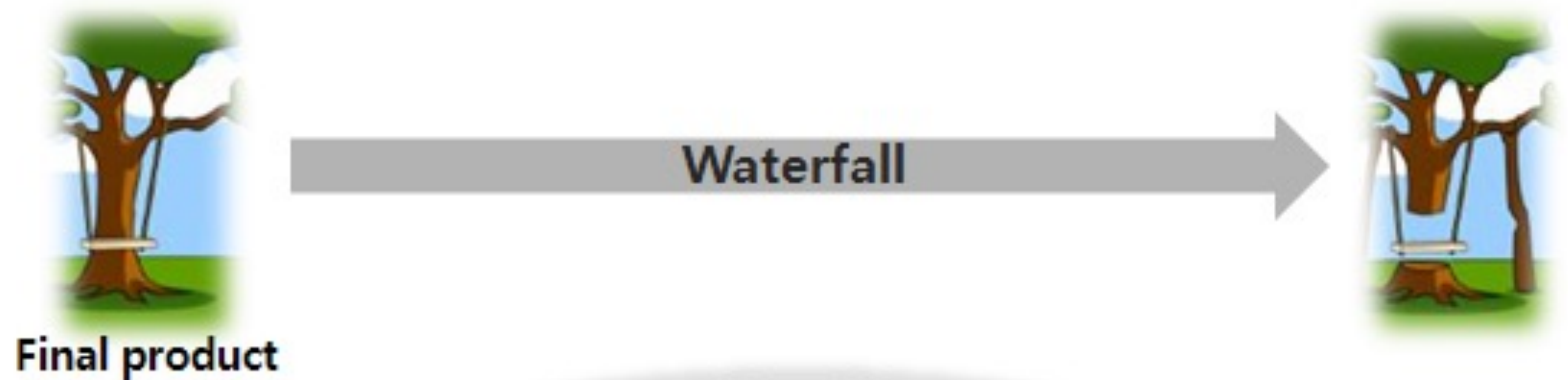


(Long-Lived) Software…

… is characterised by change.

How should we approach bug fixing, adding new features,
and technical improvements?

Planning



Eliciting requirements

Producing designs, and deciding between (some of) them

Researching, for example, appropriate design patterns

Implementing and validating last

Risk: time can be wasted building the wrong thing

Reacting

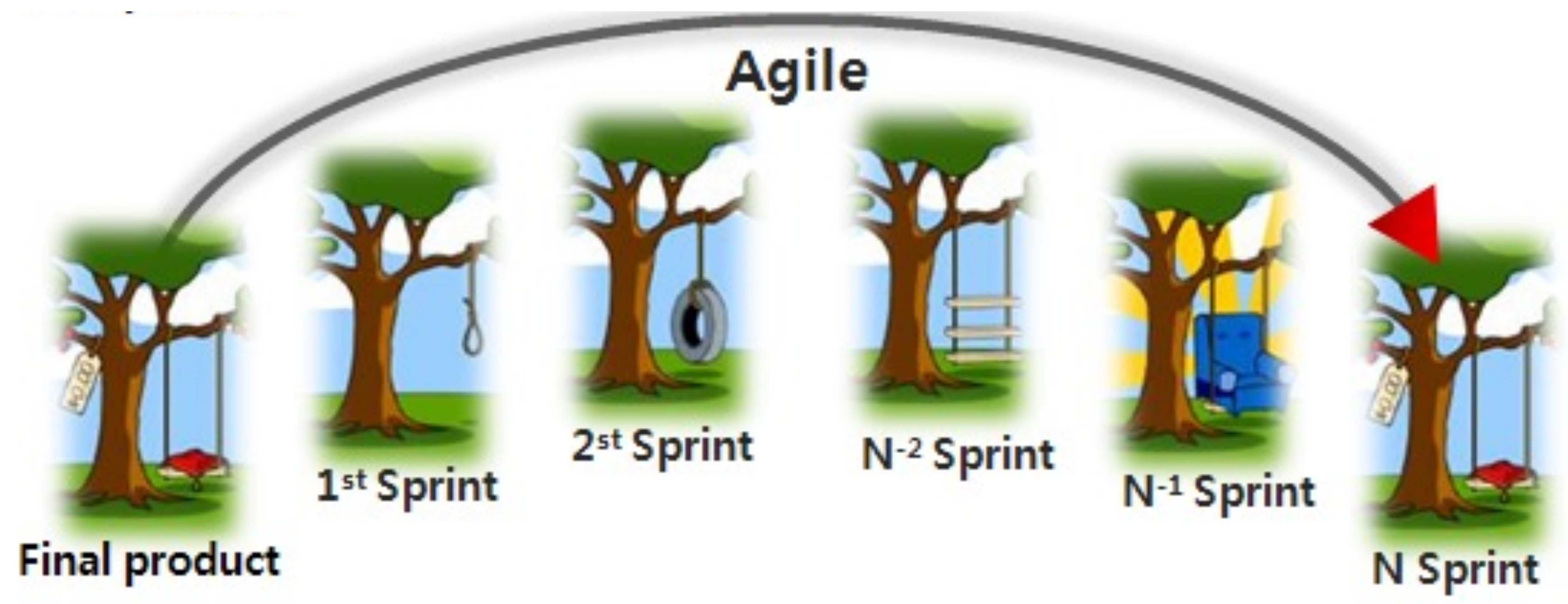
Implementing first, probably via (something like) TDD

Discovering requirements

Discovering designs and alternatives

Just-in-time requirements elicitation and research

Risk: time can be wasted waiting for the product owner



Planning and Reacting

I tend to find that I need both. My typical approach:.

1. Discuss requirements with product owner
2. Quickly hack together some code to see whether:
    - I've understood the requirements sufficiently
    - There are any challenges that I hadn't anticipated
3. R&D: sketch habitable solutions on paper, using UML
4. Evaluate solutions and implement the best, using TDD.
   Probably start again at 3.
5. Give to the product owner to validate.
   Probably start again at 1.
6. Put into production for customers to evaluate.
   Probably start again at 1.