SOFTWARE

DEVELOPMENT PROCESS

khivi.com

HARDWARE COMPANIES

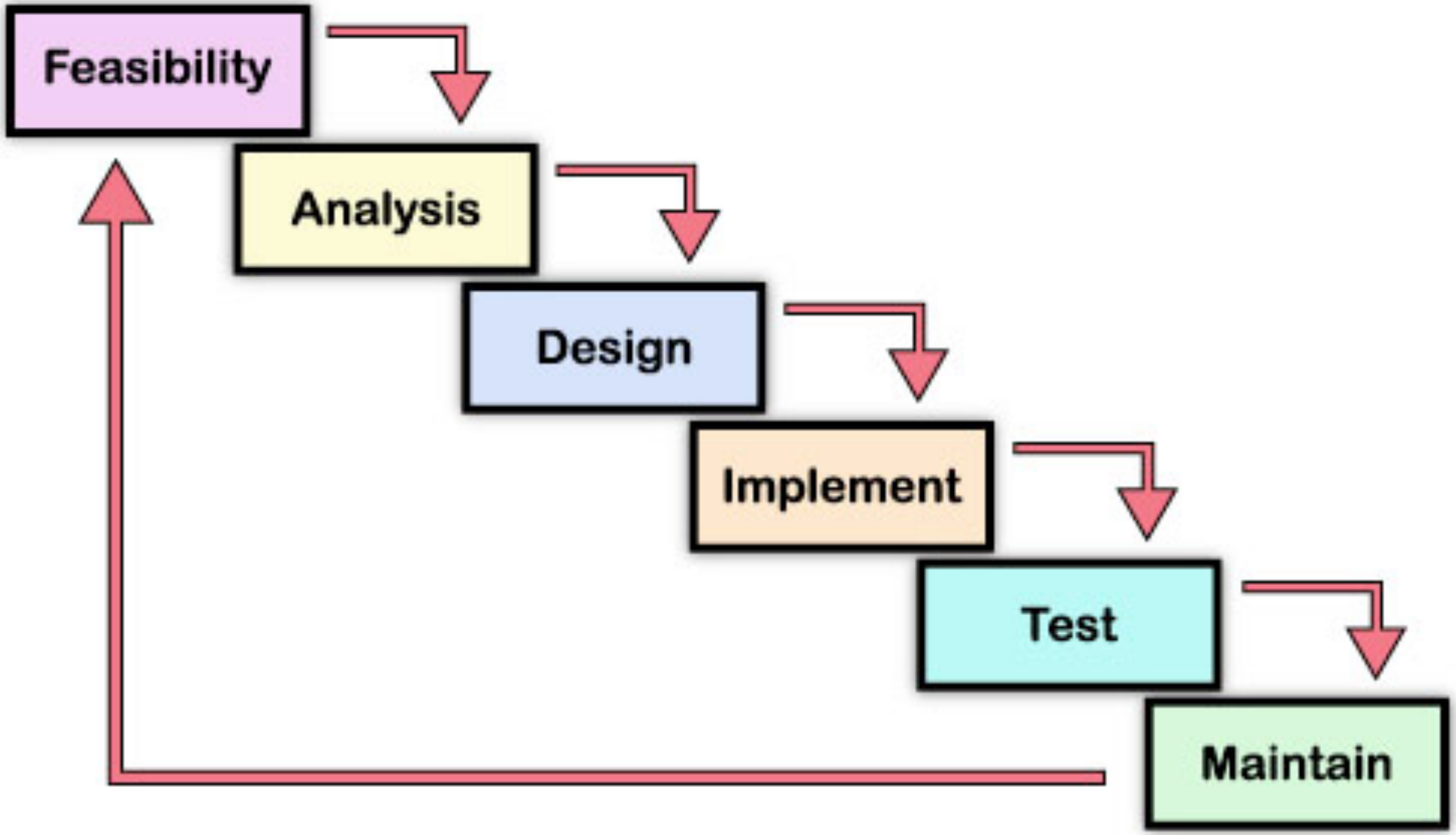
IBM, AT&T

khivi.com

SOFTWARE IS NOT SAME AS HARDWARE

- Fixing software is cheaper than other industries
- Fixing defects earlier in the process is cheaper
- Estimates in software are not accurate
- Software is easier to change
- Hard to predict the future.

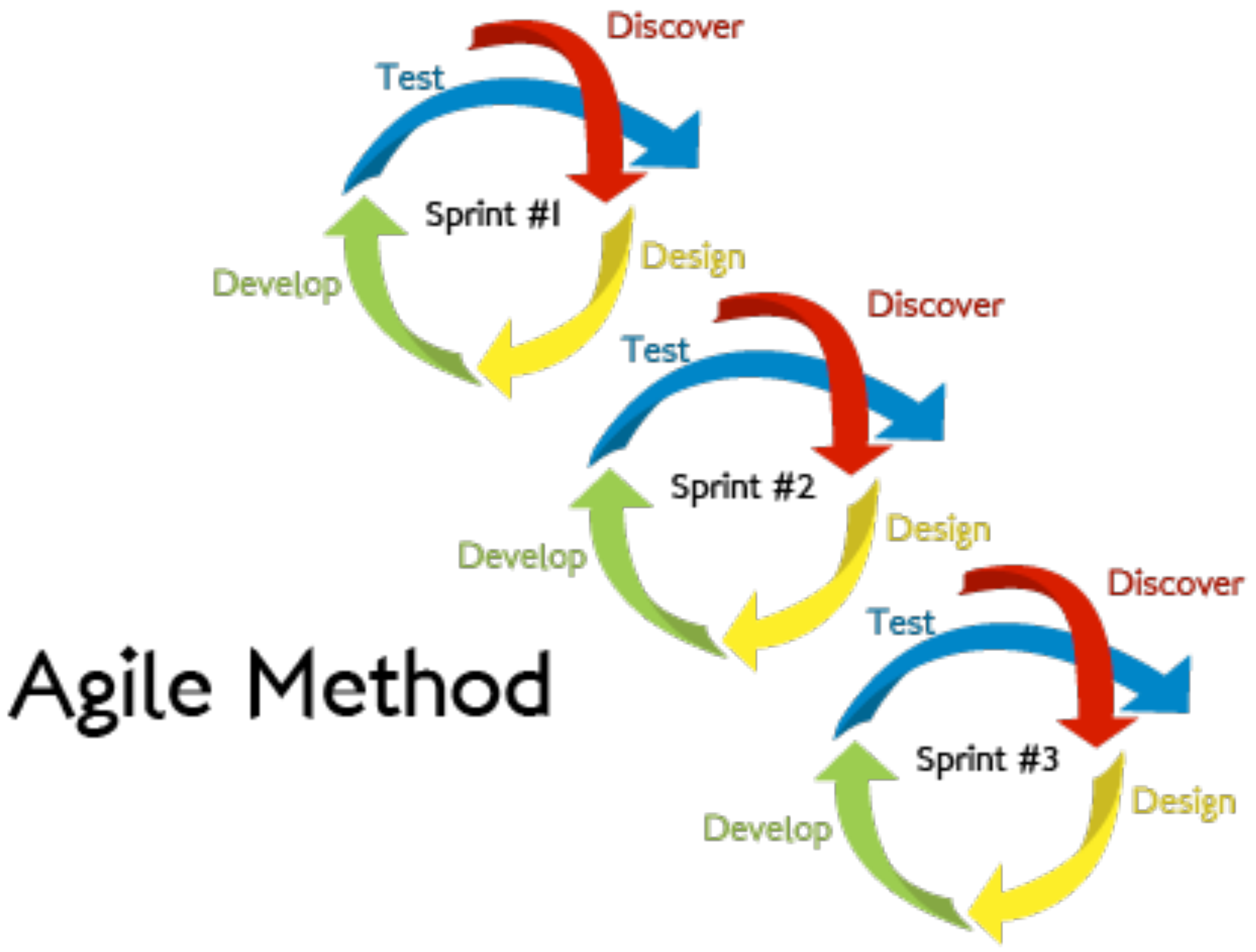
WATERFALL SOFTWARE DEVELOPMENT PROCESS



WATERFALL DEVELOPMENT PROCESS

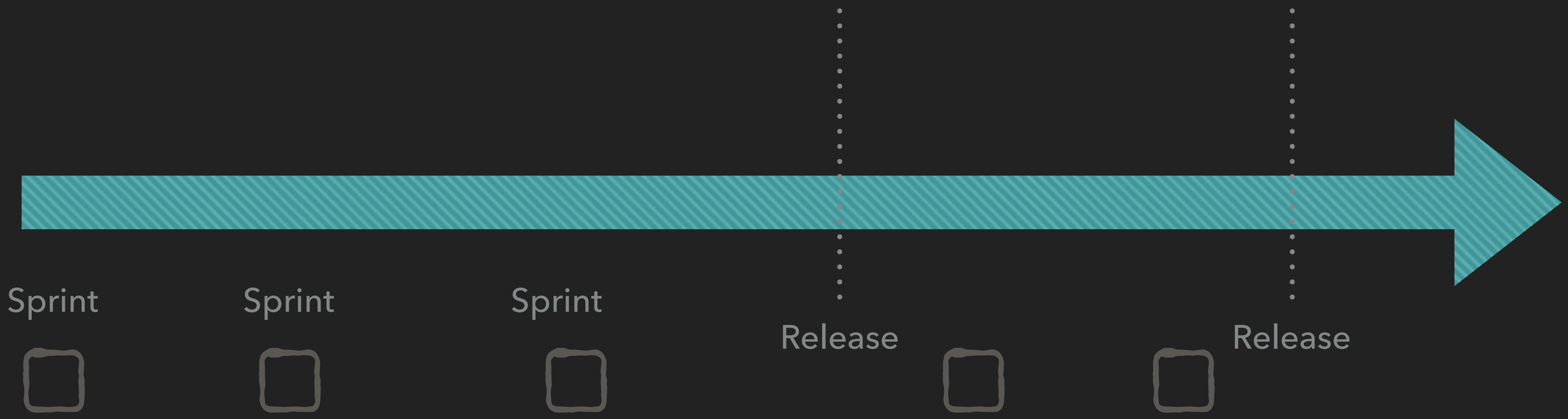
- Each stage is long and feeds into next
- Defects found downstream are expensive to fix
- Feedback loops can be incorporated
- Often late in the process
- Resistent to change

AGILE METHODOLOGY

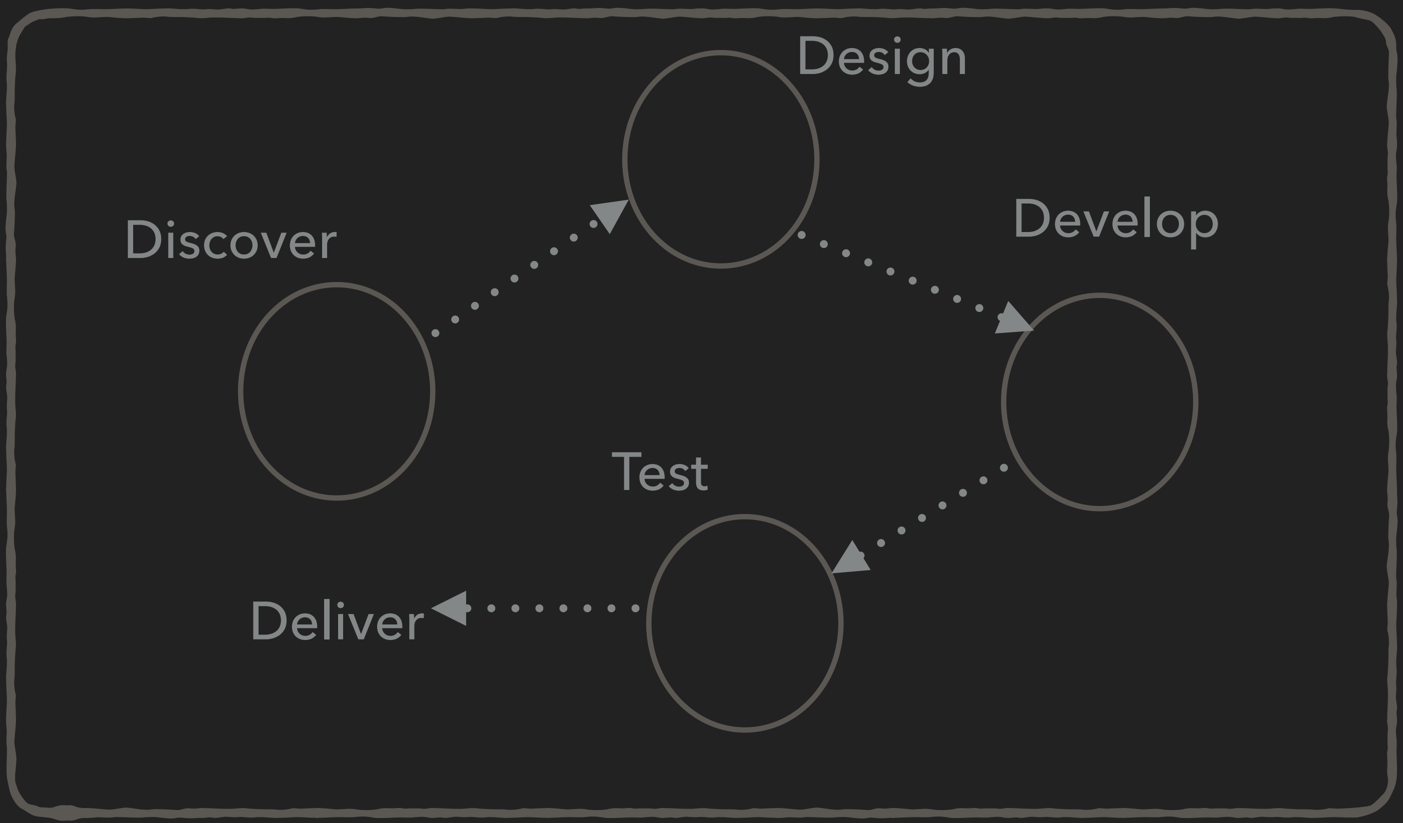


AGILE MANIFESTO

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan



Sprint



khivi.com

WHAT IS A SPRINT

- End of each sprint is a deliverable
- Bounding tasks within sprint encourages better estimates
- Features that business and customers can use earlier
- Spending less resource at each step
- Less risk since shorter cycle encourages recalibration

DELIVERY !!!!!!

- Evolutionary Delivery
  - Each sprint results in additional functionality
- Continuous Delivery
  - Constant repeatable sprints
- Adaptable Delivery
  - Changing to business realities

BEING AGILE : IS HARD

- Shorter Deliverable Time
- Smart Assumptions
- Temporary Scaffolding
- Fear of Unknowns

BEING AGILE : IS WORTH IT

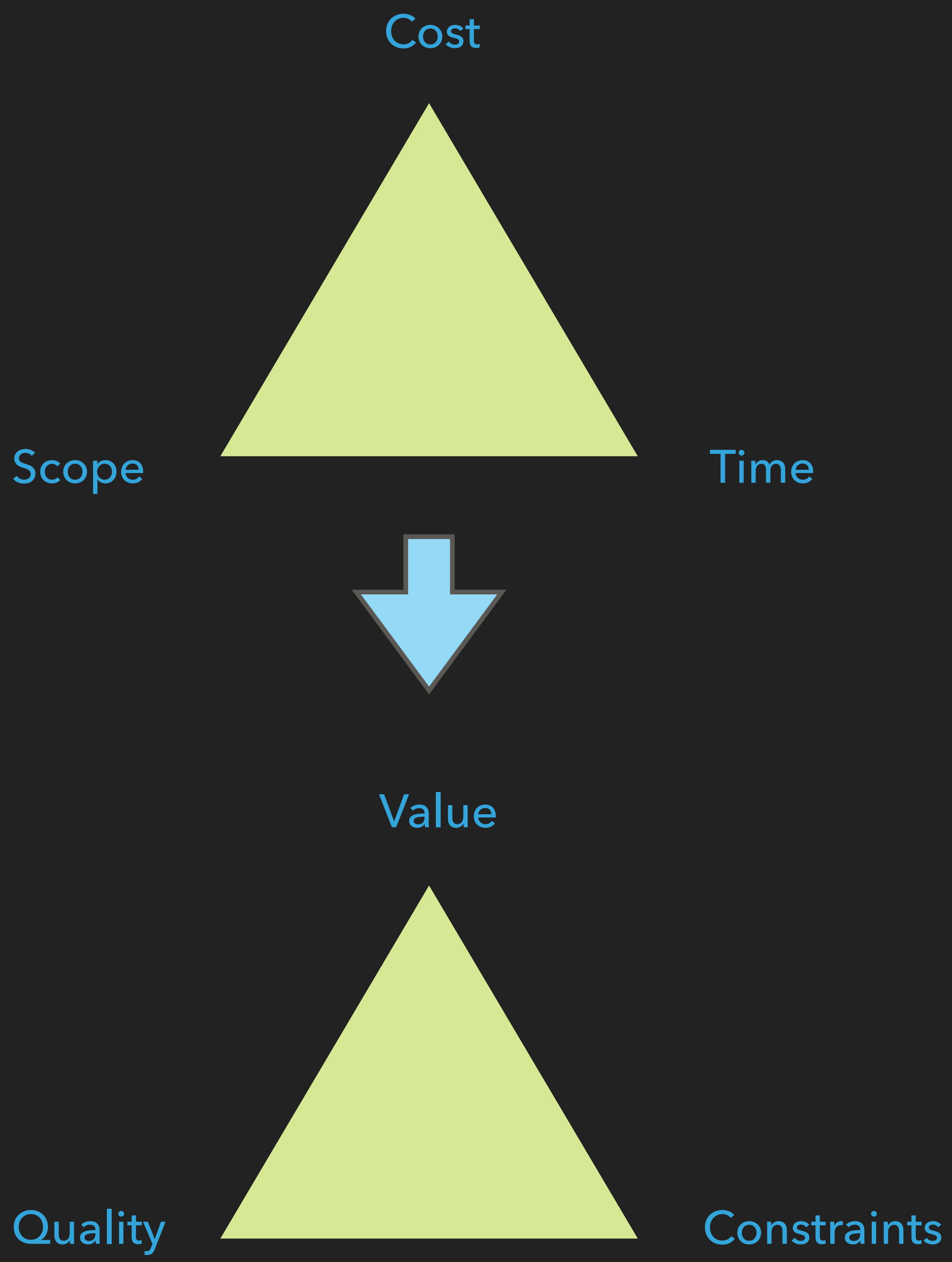
- Allows Experimentation
- Reduce Risks
- Consistent Delivery Schedule
- Corporate confidence in each sprint.

HOW
DO WE DO IT

REDEFINE SUCCESS

Waterfall Process

Agile Methodology



khivi.com

DELIVER THE KNOWNS, RESEARCH THE UNKNOWNS

khivi

HOW DO WE DO IT ....

- VERIFY VALUE
- DON'T DIVIDE AND CONQUER
- SIMPLE
- THEORY OF CONSTRAINTS
- COMMUNICATIONS
- PRODUCT PLANNING
- BETTER ESTIMATES

VERIFY VALUE CONTINUOUSLY

- Question what customers what
- Verify what is being build
- Ask why is is being build
- Be prepared to do something of more value

DON'T DIVIDE & CONQUER

- DIVIDE & CONQUER
  - Big Design UpFront
  - Early Decisions
  - Integration at end

- CONQUER & DIVIDE
  - Build the simple solution
  - Postpone Decisions
  - Integrate with stubs

KEEP IT SIMPLE

- Courage : Be confident
- Humility : Do not over-engineer
- Concise : Be brief (not terse)
- Elegant : Don’t confuse
- Smart : Don’t be smart
- Evolve: Be ready to evolve

THEORY OF CONSTRAINTS

- Recognize your constraints
- Optimize your constraints
  - Resource allocation
  - Automation
  - Learn
- There is always a constraint

COMMUNICATIONS

TYPES OF COMMUNICATION TOOLS

- Planning
  - trello, asana
- Work Item Tracking
  - jira, redmine, bugzilla, asana, trello
- Discussions
  - asana, basecamp, slack
- Documentation
  - google docs, wiki, basecamp
- (Ephemeral)
  - IM, in-person, email, slack

COMMUNICATION TOOLS

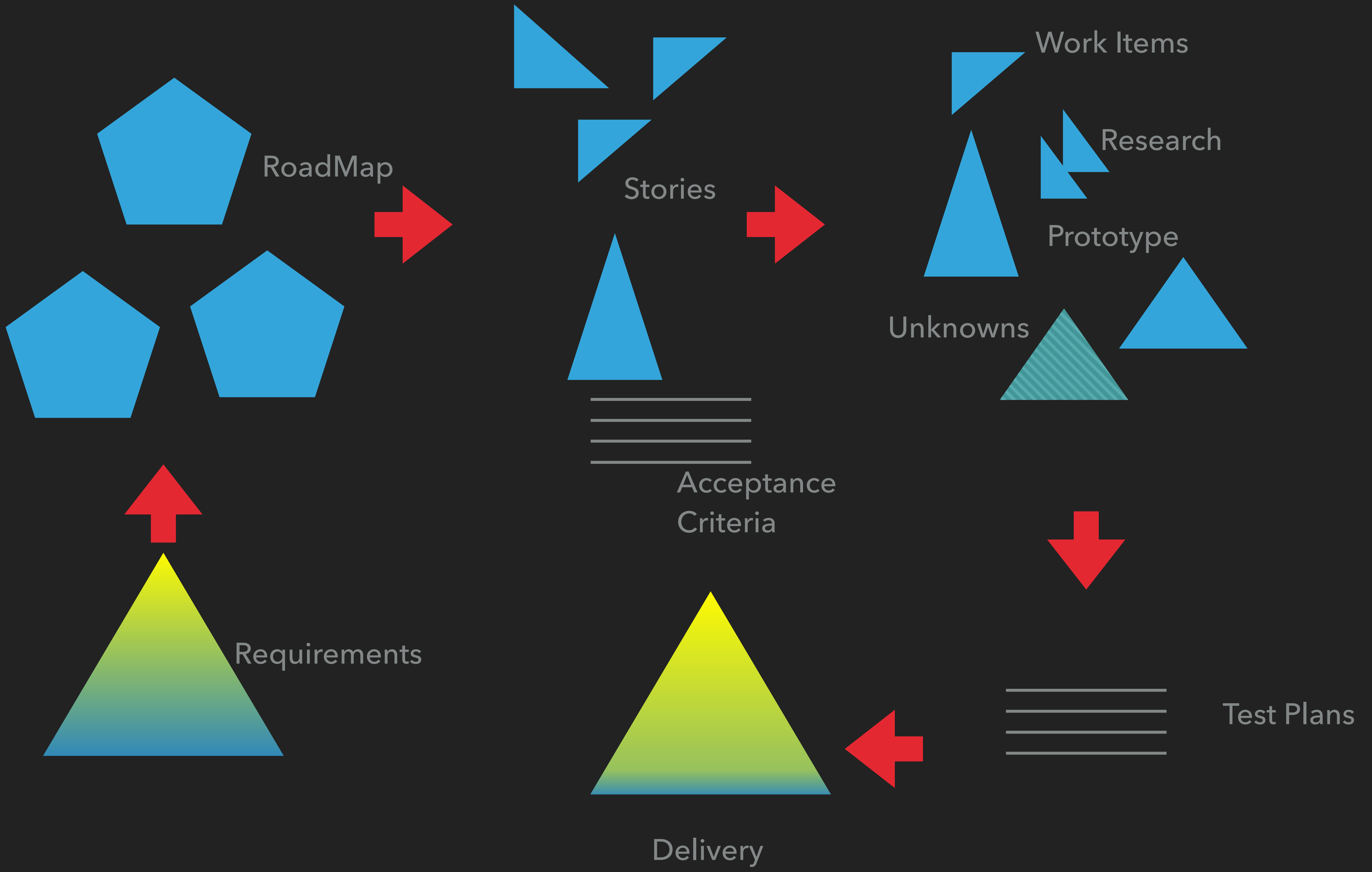
- Open - Visibility to all
- Notifiable - Can setup notifications for interesting changes
- Observable - Team members can be just observers
- Frictionless - Lightweight and easy to add information
- Containerized - Can partition based on organization needs
- Searchable - Ability to search through store
- Deep Linking - Ability to refer to a particular item in tool

CAVEATS TO LOOK OUT FOR: AND ADAPT

- Used only as a input tool. Other team members not using to to find information
- As it is setup it is confusion and overwhelming.
- There is a lot of activity occurring but in silos
- Large amount of information added is adding noise to productivity

PRODUCT PLANNING

PLANNING IS A DAILY, WEEKLY AND QUARTERLY ACTIVITY


RoadMap
Stories
Work Items
Research
Prototype
Unknowns
Acceptance Criteria
Requirements
Test Plans
Delivery

USER STORIES

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

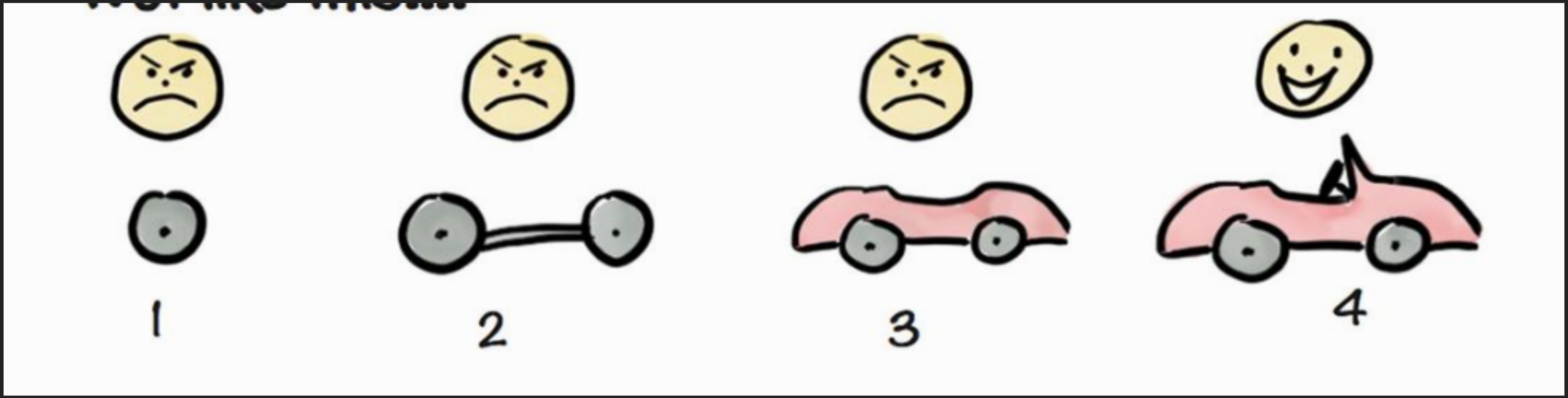
PRODUCT MANAGEMENT

- We treat long estimates as gut-feeling
- We ask team members to break large projects
- Estimate smaller projects
- Execute smaller tasks at a time (deliver..)
- Divide problems into knowns and unknowns
- Deliver the knowns
- Research the unknowns (to covert them to knowns)
- Task broken to largest time that risk is acceptable.

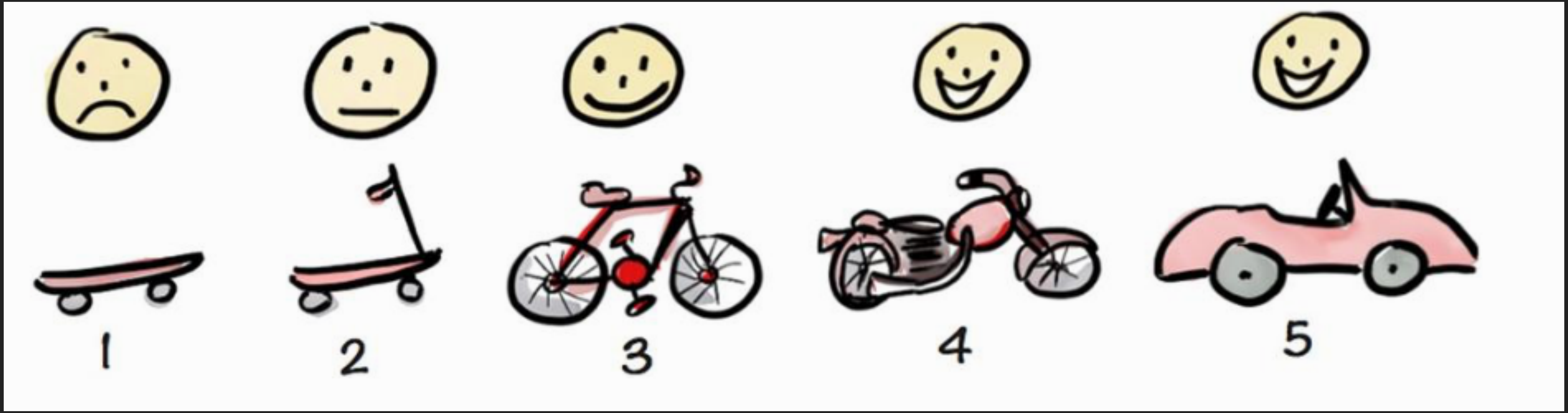
ESTIMATION

- Roadmap Planning (quarterly)
- Feature Planning (monthly)
- Sprint Planning (weekly)
- Fibonacci (1,2,3,5,8,…), T-Shirt Sizes (S,M, L, XL, XLL)
- More frequently you measure the better estimates you get

DON'T DO THIS



DO THIS



SOFTWARE DEVELOPMENT

SOFTWARE IS NEVER DONE

- Bugs are discovered
- Customers want new features
- Market demands new functionality
- Actively refactor code

Code for Change

EXTREME PROGRAMMING VALUES

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Code for Change

PUSH FAST, CATCH EARLY

- Code Reviews
- Automated Testing
- Continous Integration
- QA Alignment
- Rapid Deployment

Code for Change

PATTERNS TO EMBRACE

- Humility
- Strong views, weakly held
- Appreciate beauty
- Deliver the knowns

Code for Change

ANTI-PATTERNS TO AVOID

- PowerPoint Architecture
- SuperHero Engineering
- Personal Silo
- Yes we can
- Cognitive Overload
- Manual Testing

Code for Change