User Stories and Tasks

The What and How of Building Software

Requirements Gathering with User Stories

- Describe only **ONE** thing the software does for your users
- Use the language of your users
- The requirement should come from the users
  - Human-Centered / User-driven
- Short
  - 3 sentences tops!
- Don’t talk about specific technologies or implementation details

Example User Stories (Movie Tickets)

Title: View showing films

Description: Users should be able to see all films currently playing.

Title: View show times

Description: Users should be able to see show times for the films

Title: Choose a seat in the theatre

Description: Users should be able to pick open seats in the theatre

Title: Order Snack Packs

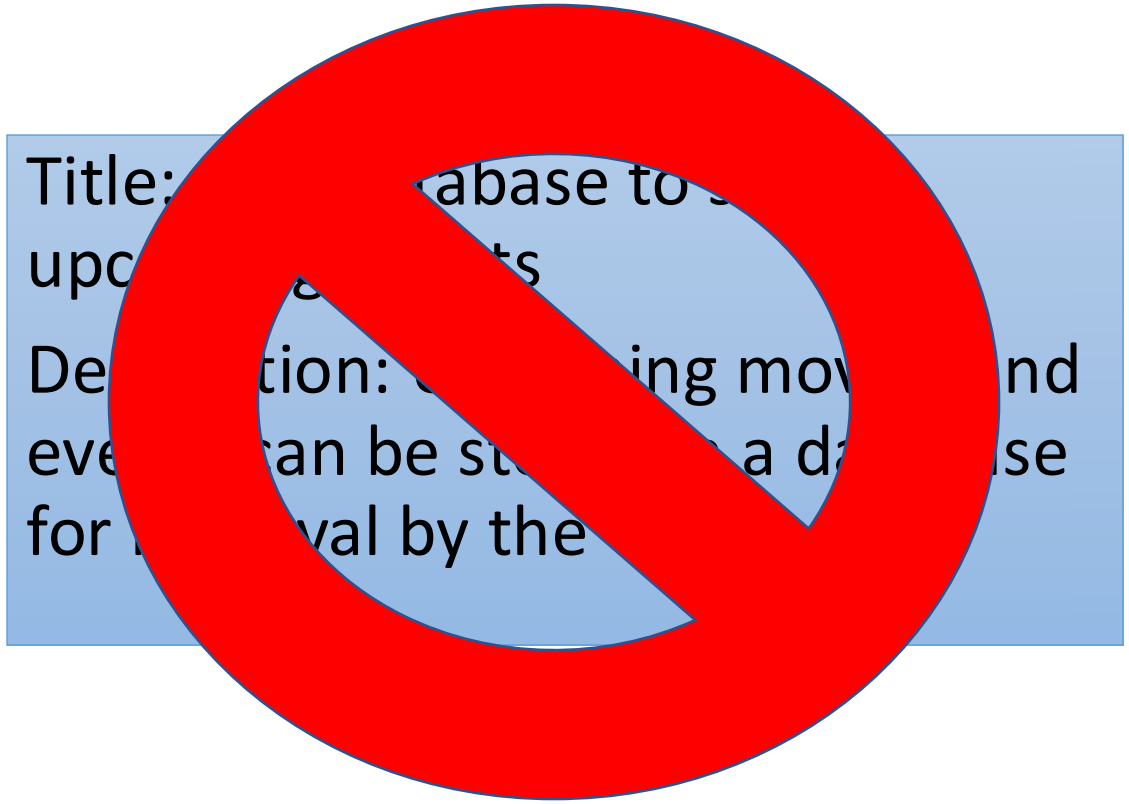
Description: Users should be able to pre-order snacks ready for pick up

Good User Story? (Movie Tickets)

Title: SQL Database to store upcoming events

Description: Upcoming movies and events can be stored in a database for retrieval by the GUI

Good User Story? (Movie Tickets)



NO IMPLEMENTATION DETAILS

What Else?

Prioritization and Time Estimates

- Generate time estimates for all the user stories
  - In days…not hours…
  - https://planningpokeronline.com/
- Get customer feedback on the priority of each user story

Minimum Viable Product

- BASELINE Functionality
- Use the user story time estimates/priority as a guide
- Your first release is about delivering what is necessary
- Focus on what is features are needed to meet the users needs
  - Minimal, but useful and usable
  - More enhancements and cool stuff comes later

Tasks

- A collection of developer work necessary to accomplish a user story
- Each task is designed for ONE developer in mind
- Title, Description, and Time Estimate

Title: View showing films

Title: Movie Database

Description: Backend storage of films, poster art, and other metadata

Est: 2 days

Title: Film Gallery View

Description: Display all poster art in grid gallery with mouse over for show times.

Est: 3 days

Task Ideas?

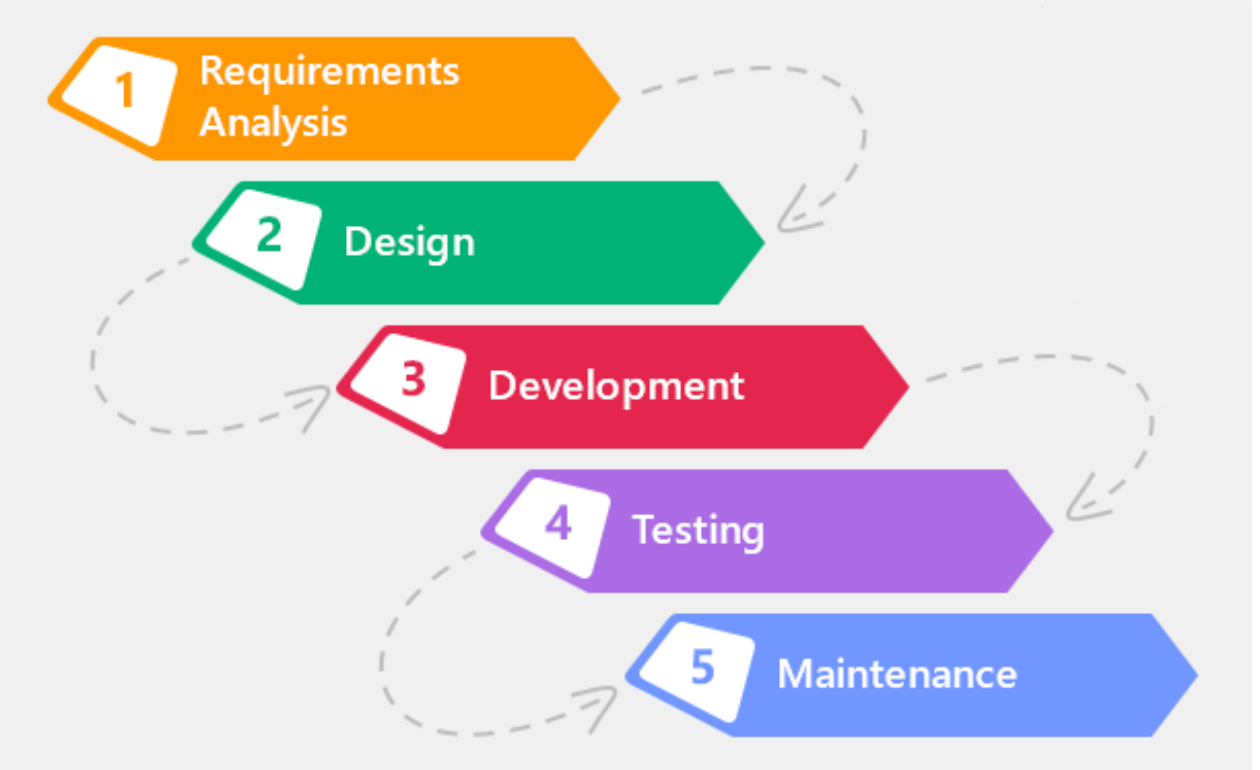
Title: Choose a seat in the theatre

Description: Users should be able to pick open seats in the theatre

You have work…
now you need a plan.

Old Fashioned Software Development

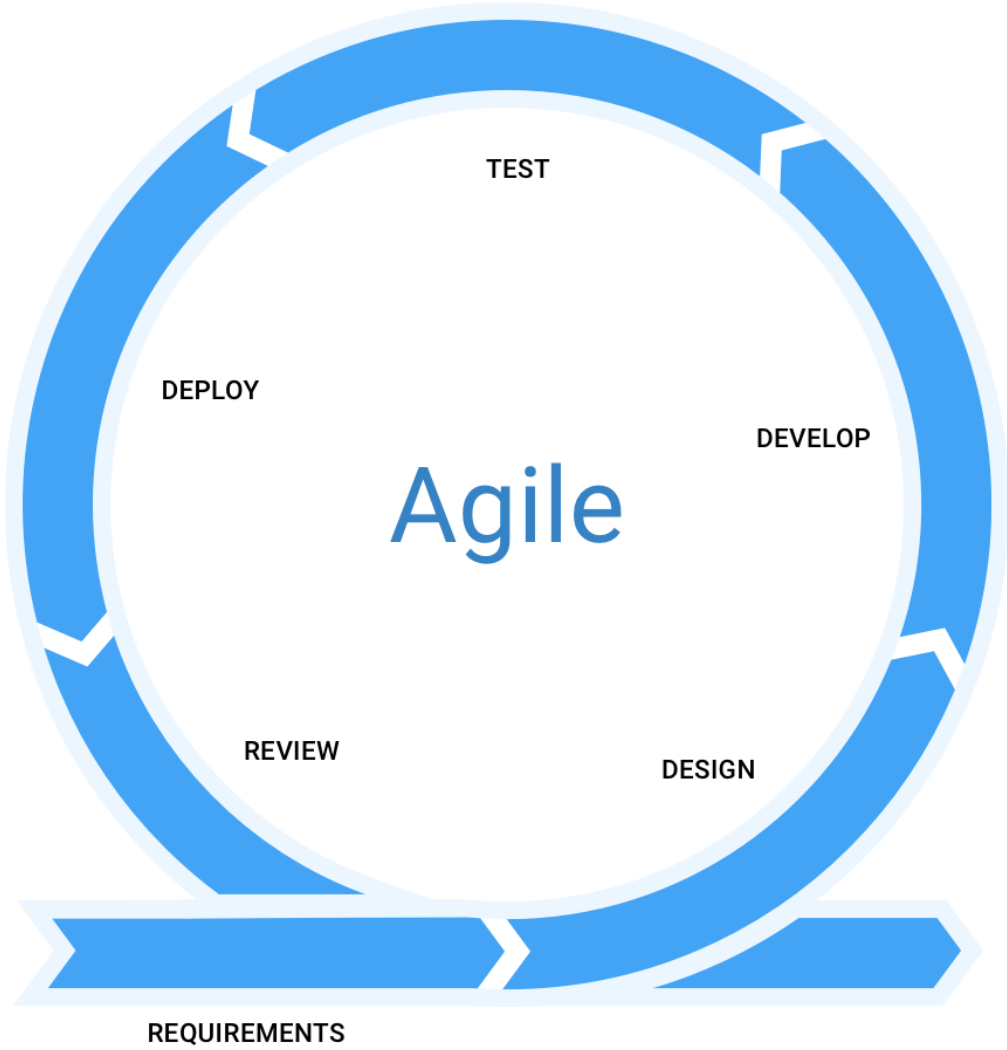
- Waterfall
  - Sometimes called Big Design Up Front (BDUF)
  - Does not handle volatile requirements or change easily
  - Getting through the process could take months….maybe years.



Tends to lead to more failures than successes…

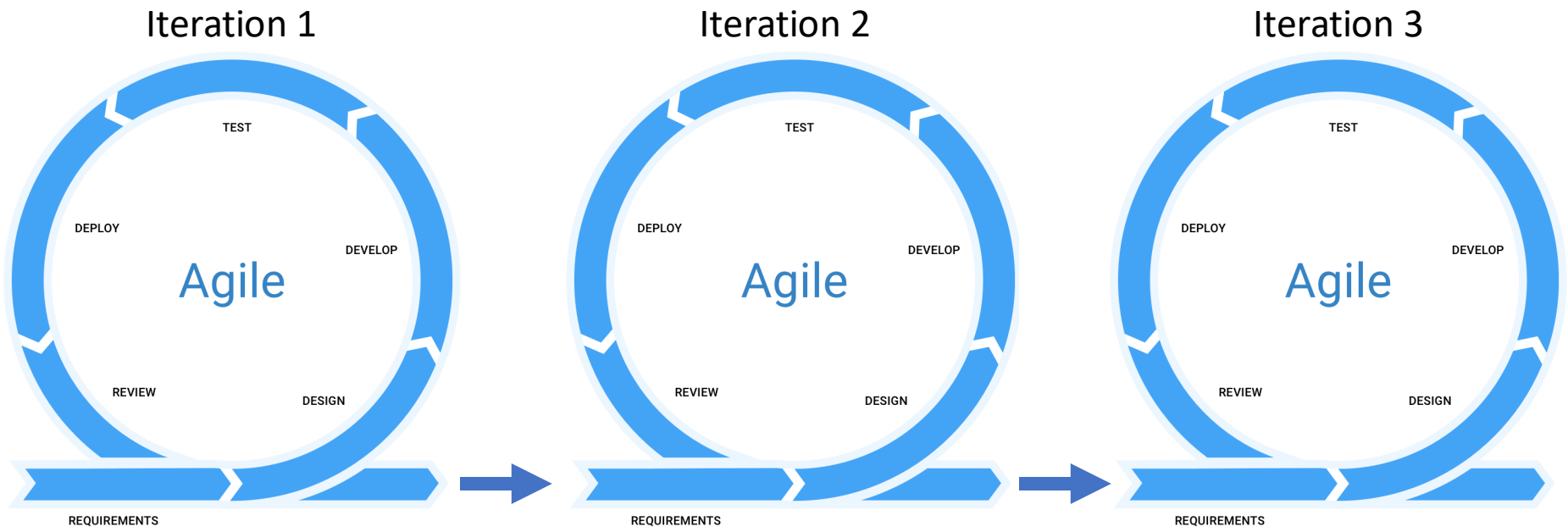
Modern Software Development

- Agile
  - Many different variations
  - Viewed as more of a cyclic process than a linear sequence of events
  - More flexible for natural volatility in the development process



Iterations

- Also called sprints
- Shorter periods of development that repeat the cycle



Simple Planning (Agile-lite)

1. Work with customers to define user stories (requirements) and priorities
   - Request customer clarification if needed
2. Work with your team to estimate time/difficulty for tasks
   - Request customer clarification if needed
3. Plan iterations with your team
   - Weekly iterations are a good idea
   - Plan work based on the priority and task estimates
   - **Explicitly** assign work among the group (GitHub Issues)
4. "Deliver" the work for the iteration
   - This is also the time to evaluate any disparity between planned and completed work
5. Repeat 1-4 until "done"

You Can’t Plan for Everything

- You will **NEVER** be able to account for all eventualities
- What you **CAN** do is know where you stand at any given point in time
  - USE GITHUB ISSUES!
- While this sounds like a consolation prize, this is important
- Without a process in place to tell you there are problems, development would continue down the wrong path
- Instead, we can adjust by changing our iteration plans, consulting with the customer, and refocus our resources for the best possible outcome