TEAM DOCUMENTATION

Brian McMillin
11 October 2017

Brian's Plan for This Presentation

START

Begin with a Joke

Disable Interrupts

Deliver Lecture

Enable Interrupts

Q&A - sleeping audience asks no questions

SUCCESS

A Rabbi,
A Priest and
A Duck
Walk into a bar...

Take Away

- Let Documentation Drive the Project
- Documentation is not an Afterthought
- Documentation is a critical component and must be Tested
- Team Participation is just as important as in any other aspect of Development
- Iteration and Feedback are critical

DISABLE INTERRUPTS

- This will be a fairly rapid-fire presentation
- Each slide could probably be discussed for a week
- Some of my assertions may be controversial
- Please assume that these are well-reasoned positions
- The presentation is intended to stimulate discussion later
- Some ideas may be of immediate benefit to you

Styles of Documentation

SCOPE

- Project Documentation
- Program Documentation
- Module Documentation
- Function Documentation

AUDIENCE

- Design Documentation
- Implementation Documentation
- User Documentation
- Support Documentation

Journalistic Approach

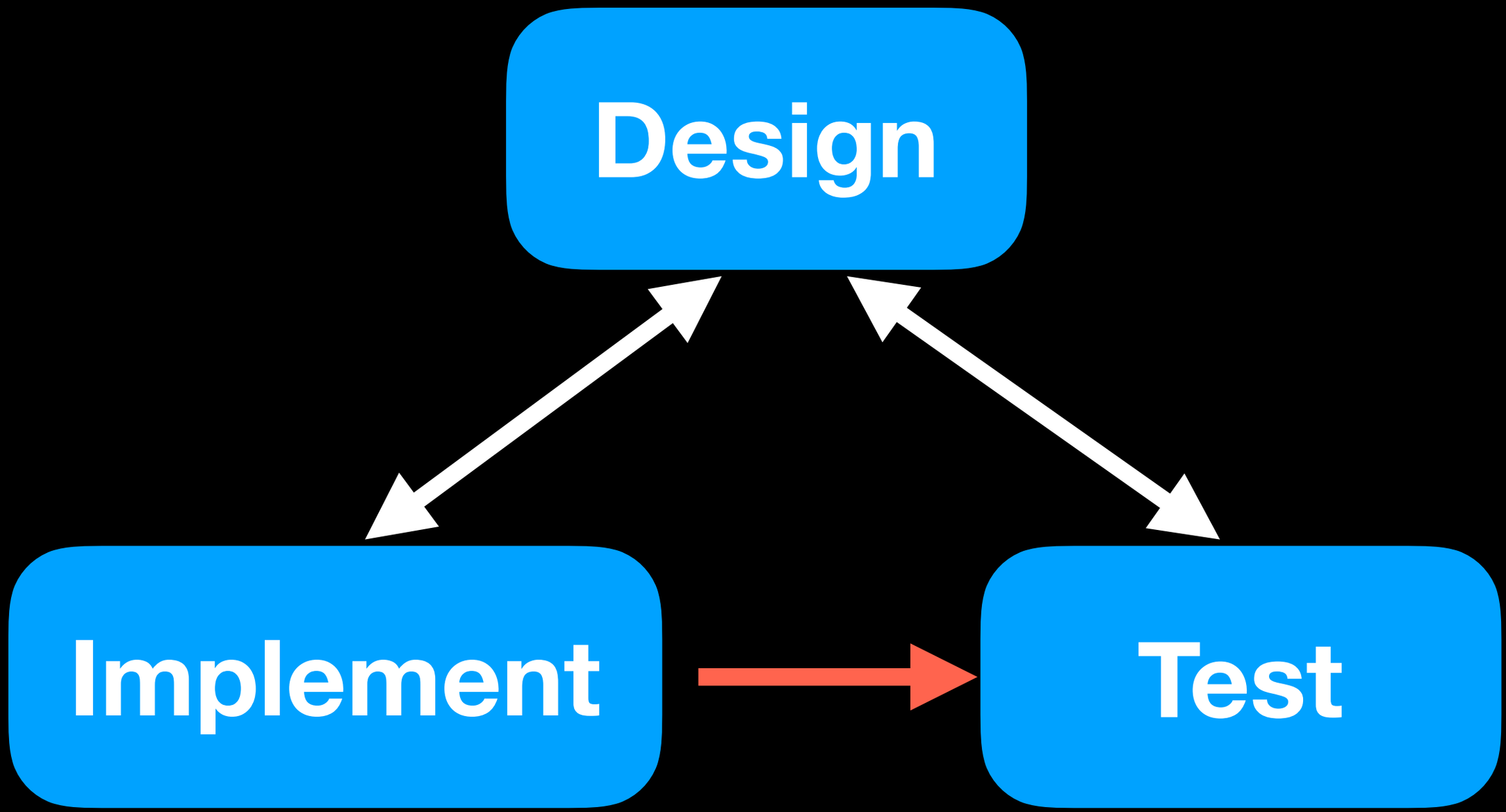
- **WHO** - is the target audience?
- **WHAT** - are we trying to accomplish?
- **WHERE** - does this fit into the flow?
- **WHEN** - will this feature be used?
- **WHY** - is this necessary?
- **HOW** - are we going to solve the problem?

Precision is the Enemy of Clarity

- “It is only necessary to be *PRECISE* when there is some doubt as to the meaning” - Richard P. Feynman

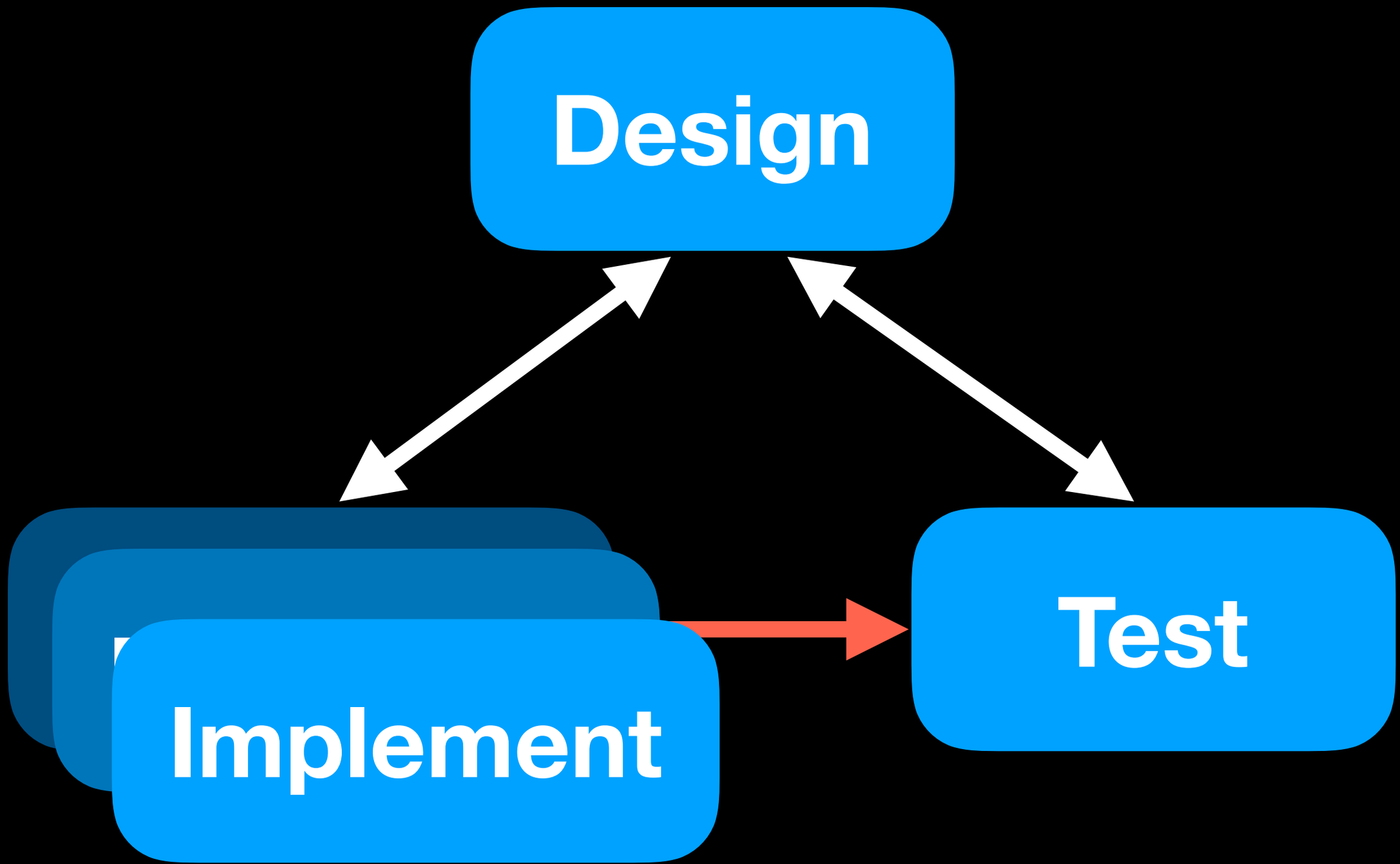
- Strive for Clarity
- Clearly answer the **WHY** and the reader will understand
- Belabor him with details of **HOW** and he will not
- This is the fallacy of “Self-Documenting Code”

Development Team



Three Different People

Expect Many Implementations



Design Rules

- It is a mistake to optimize too soon: specify REQUIREMENTS not ALGORITHMS
- Anticipate multiple implementations
- Let the Implementer do the implementation
- Design for testability
- Let the Tester create the tests
- **YOUR JOB IS TO DOCUMENT THE DESIGN**

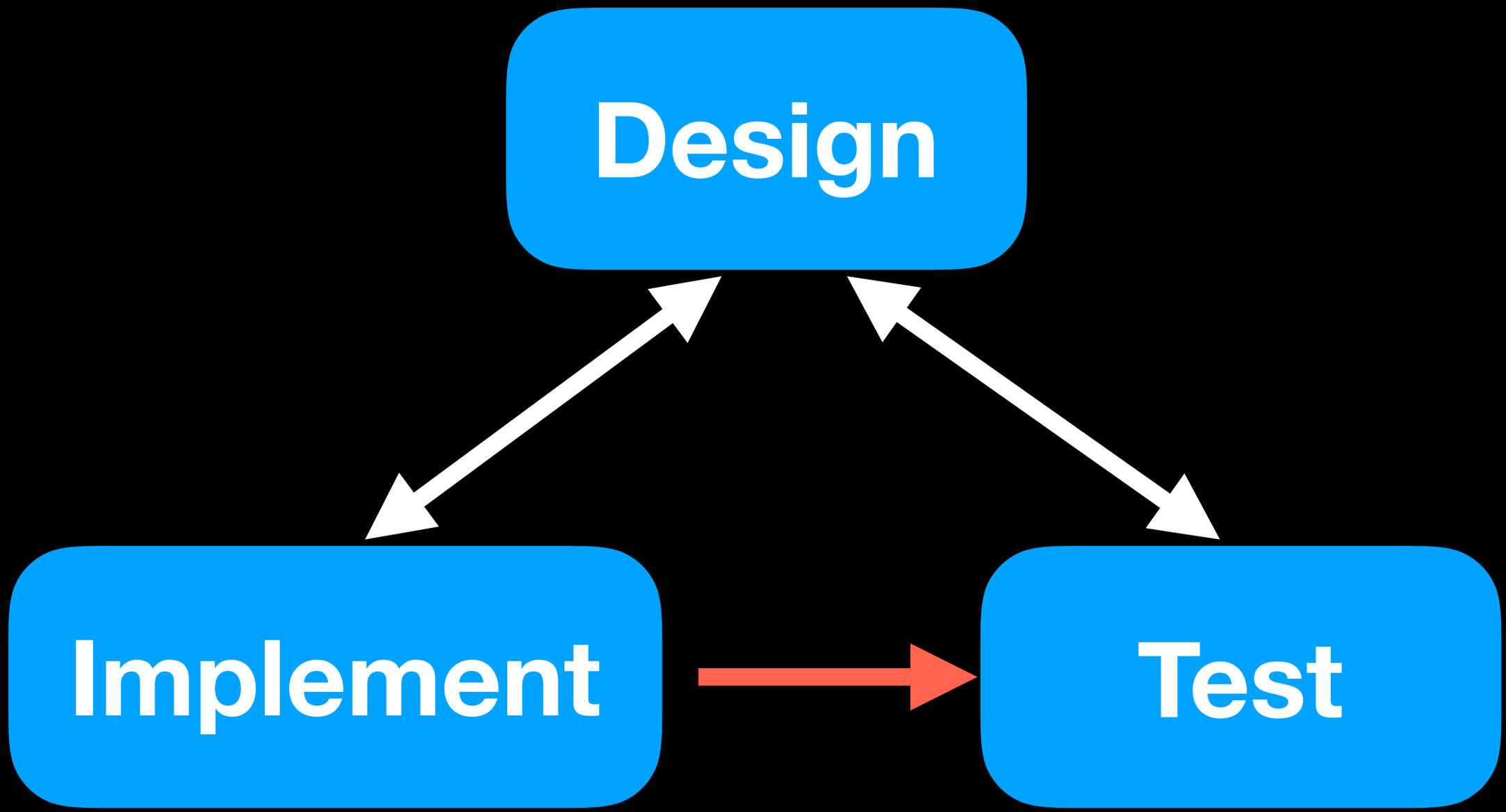
As the Designer -

- The hardest thing you will ever do is **NOT WRITE CODE**
- You will watch as other team members struggle to do what you *know* you COULD do better and faster

BUT....

- You will benefit from exposure to the creativity of your team members
- Your Implementer and Tester will show you areas of your design that need additional work
- The end product will be completed faster and be of higher quality

Development Team



Three Different People

Implementation Rules

- Create an Implementation based on the Design documentation
- Choose the most expedient Implementation tools
- Optimize later - odds are it won’t matter anyway
- **YOUR JOB IS TO HELP IMPROVE THE DESIGN DOCUMENT** and (incidentally) write the implementation

Test Rules

- Use the Design documentation to craft your test cases
- Do not test based on a specific Implementation
- Remember: Test cases for **GOOD** - - **BAD** - - **BORDERLINE**
- Never delete a test case - they are a valuable resource
- Anticipate continuing tests for Regression and Production
- Build tests to be included in Production code
- **YOUR JOB IS TO HELP IMPROVE THE DESIGN DOCUMENT** as well as creating test cases to insure the Quality of the Implementation

Iterate:
Implement and Test

- When a Test case does not behave as expected - - -
- Report to the Designer
- If the Implementer or Tester misunderstood the goal, figure out WHY and correct the Design document
- **ACTUAL USAGE BECOMES THE QUALITY ASSURANCE TEST FOR THE DESIGN DOCUMENT**
- Each iteration improves the documentation and makes a more robust product
- Teamwork improves each member’s skills

JavaDoc Style documentation Is Useless

- Boilerplate “one size fits all” is equally bad in all cases
- Fill-in-the-blanks never seem to get filled in
- No requirement for usage examples
- Focuses on **HOW** but never **WHY** or **WHEN**
- No narrative to provide context
- “Pretty” output gives the illusion of substance

Productivity Metrics are Useless

- Documentation is actually MORE IMPORTANT than code
- “Standard Lines of Code” give no weight to comments
- No standard, objective method of evaluating Quality of Documentation
- Actual Code (Implementations) may be rewritten many times during project development.

Rewriting Code is NOT a bad thing!

Consider Literate Programming

- Begin with a Narrative
- Add Code only when necessary
- Do not muddle Code for different features together
- Envision substituting Code written in a different language

    How does the Narrative change?

    Should the narrative change at all?

Documentation is a Project Management Tool

- Managers understand what a Team is working on
- Team can present status in a High-Level style
- Relationships between Teams become clear(er)
- Scopes and Schedules take on meaning

Team Member Roles

- Three roles: Designer - - Implementer - - Tester
- Each team member takes a different role for different features
- Cross training
- Skills improvement
- Burnout prevention
- **Quality Improvement**

Take Away

- Let Documentation Drive the Project
- Documentation is not an Afterthought
- Documentation is a critical component and must be Tested
- Team Participation is just as important as in any other aspect of Development
- Iteration and Feedback are critical

If you want to Learn a subject
DO THE HOMEWORK

If you want to **REALLY** learn a subject
TRY TO TEACH IT