Software Engineering

ECE 651

Spring 2020

Lecture 3

Facets of Software Engineering

- Requirements Definition
- Design
- Implementation
- Verification
- Delivery/Deployment
- Maintenance

Requirements definition	What does our software do? What is our scope?
Design	What modules + classes? Multiple programs? How do interfaces work together? Use Principles!
Implementation	Write the code; Unit testing
Verification	How do we know it does what it's supposed to do? Testing, statistical analysis, code review
Delivery/Deployment	Giving customer the software; putting on website
Maintenance	Software needs to be changed after we're "done"; bug fixes, new features, monitoring

How do these facets interrelate?

It depends on the software development process – we'll revisit this later today

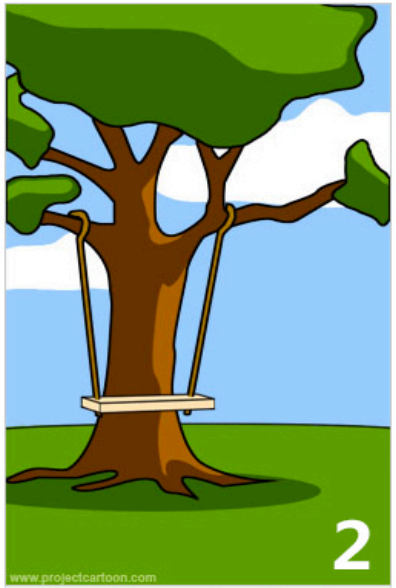
Customer (external, boss, yourself) has something they want the software to do

We need to understand requirements and translate into specifications

Decisions related to software development process revolve around requirements issue

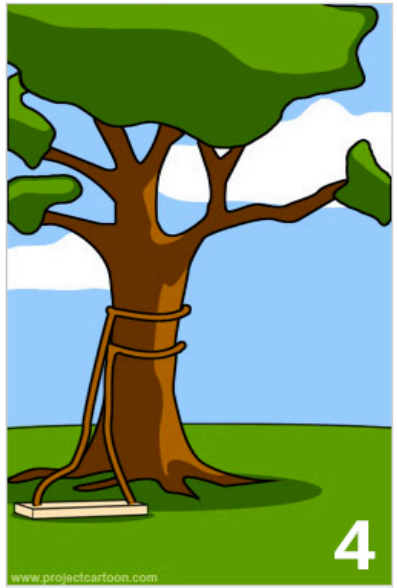
Requirements Definition

How the customer explained it



How the project leader understood it

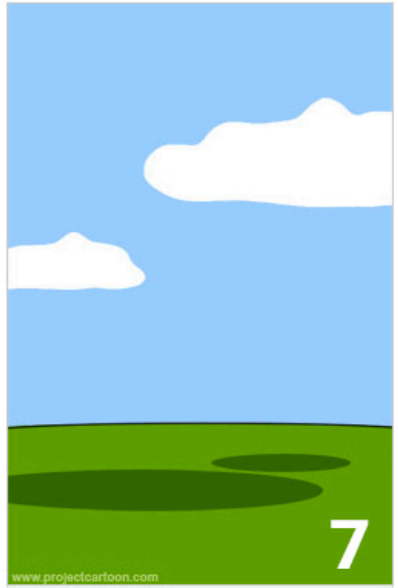
How the analyst designed it



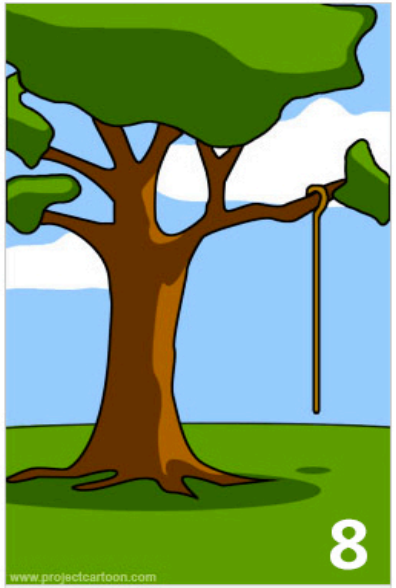
How the programmer wrote it

What the beta testers received

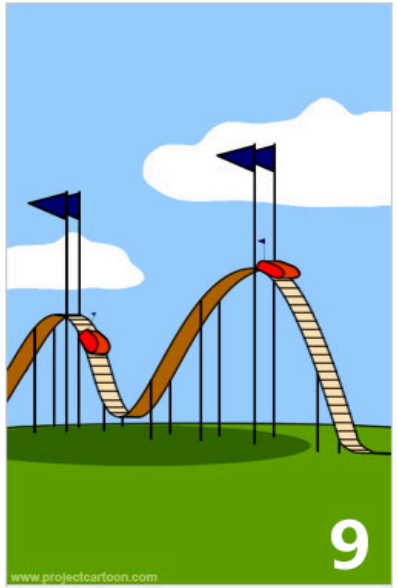
How the business consultant described it



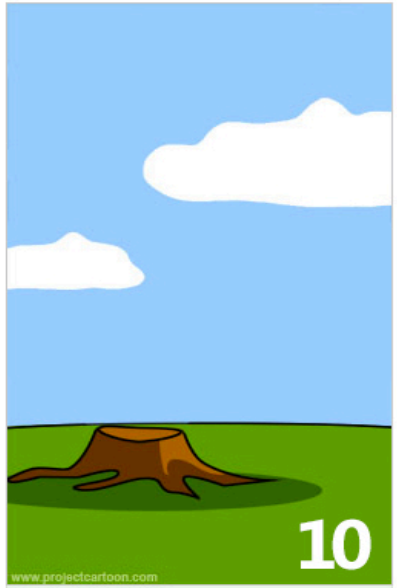
How the project was documented



What operations installed



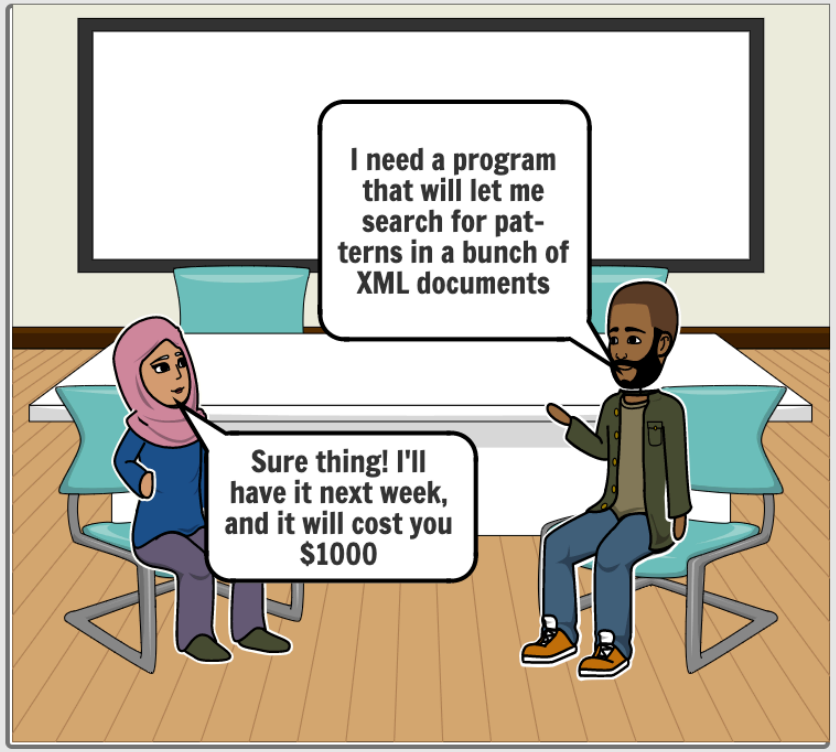
How the customer was billed

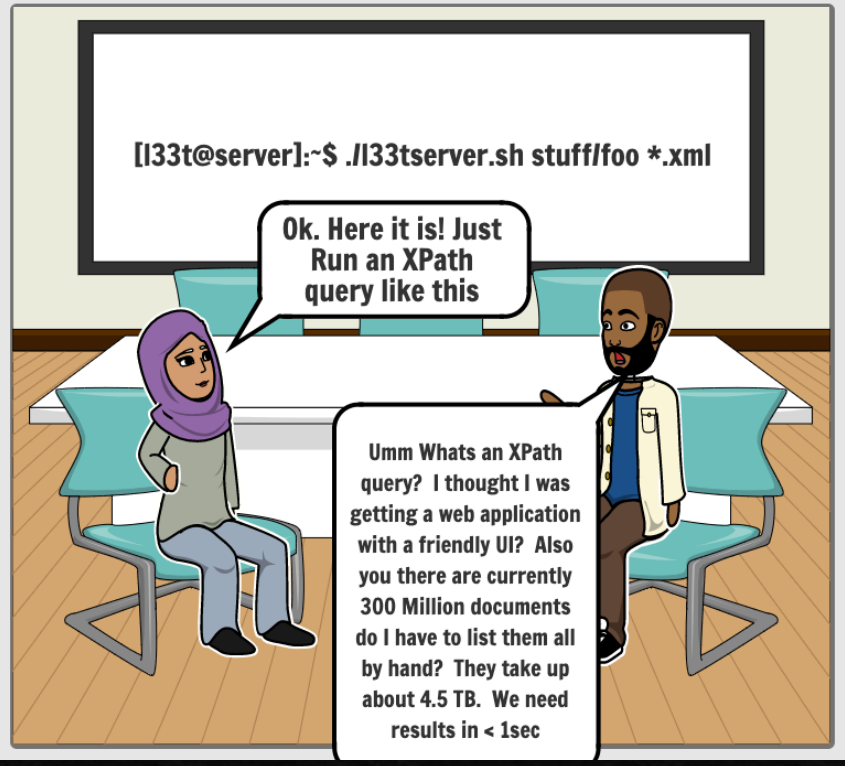


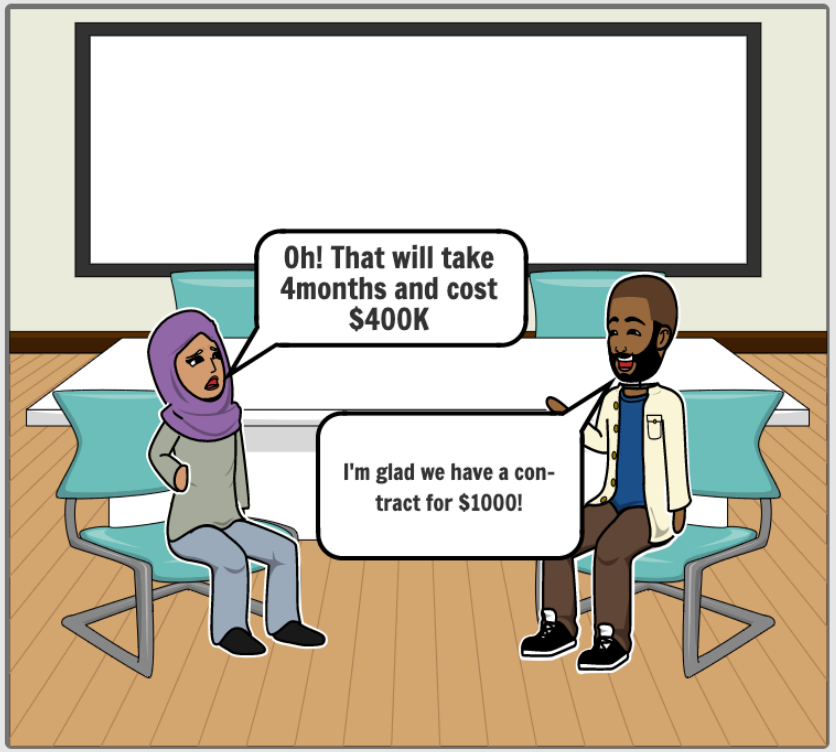
How it was supported

What marketing advertised

What the customer really needed


I need a program that will let me search for pat-terns in a bunch of XML documents
Sure thing! I'll have it next week, and it will cost you $1000


[l33t@server]:~$ ./l33tserver.sh stuff/foo *.xml
Ok. Here it is! Just Run an XPath query like this
Umm Whats an XPath query? I thought I was getting a web application with a friendly UI? Also you there are currently 300 Million documents do I have to list them all by hand? They take up about 4.5 TB. We need results in < 1sec


Oh! That will take 4months and cost $400K
I'm glad we have a con-tract for $1000!

Think, Pair, Share

What went wrong?

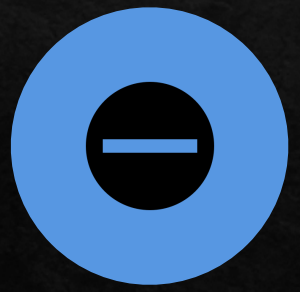
Who was responsible?

How could things have gone better?

What Went Wrong?

Customer and Developer had different **ideas** about the goal

Developer did not spend time **understanding** customer's needs



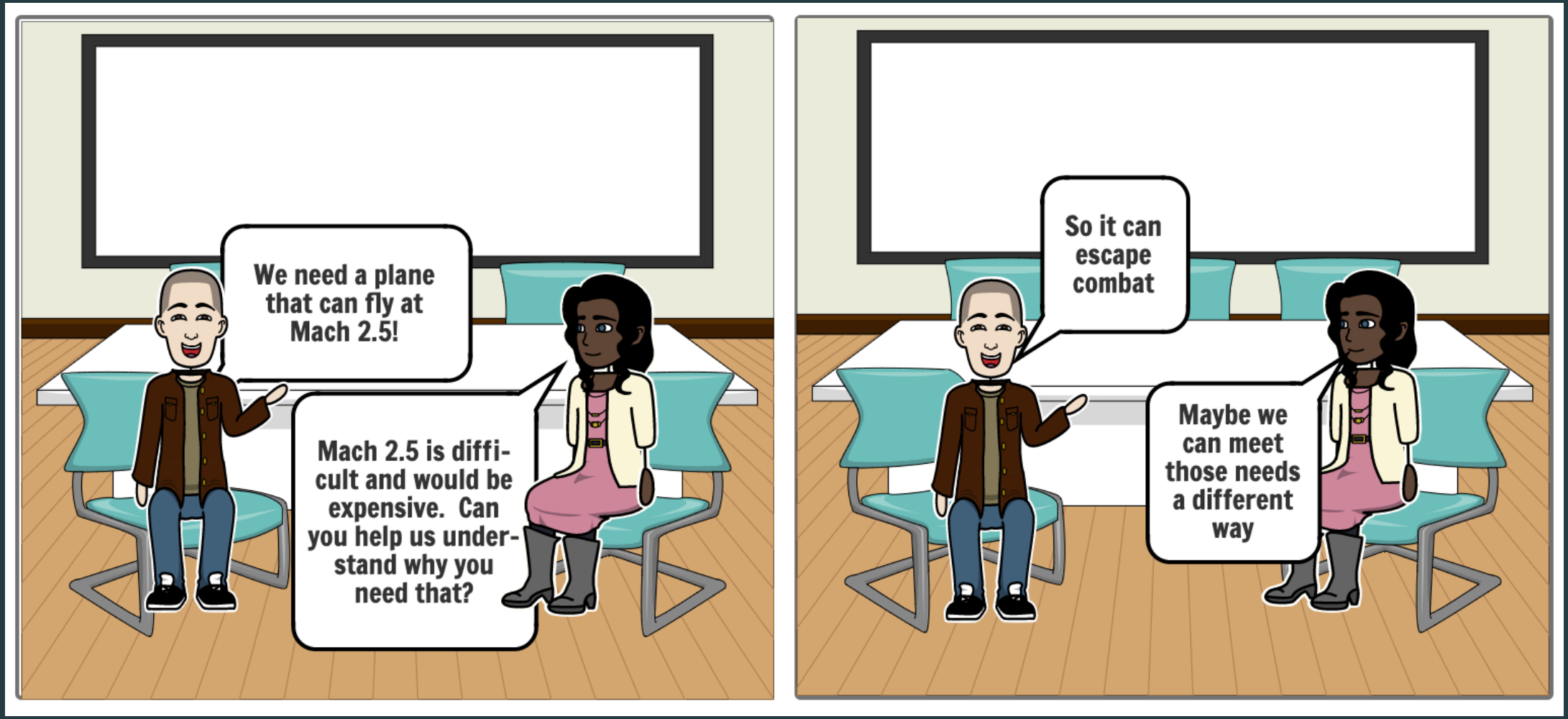
Customer had requirements that were **not mentioned**

Developer didn't **write detailed specification**

Developer should have known better

Specification

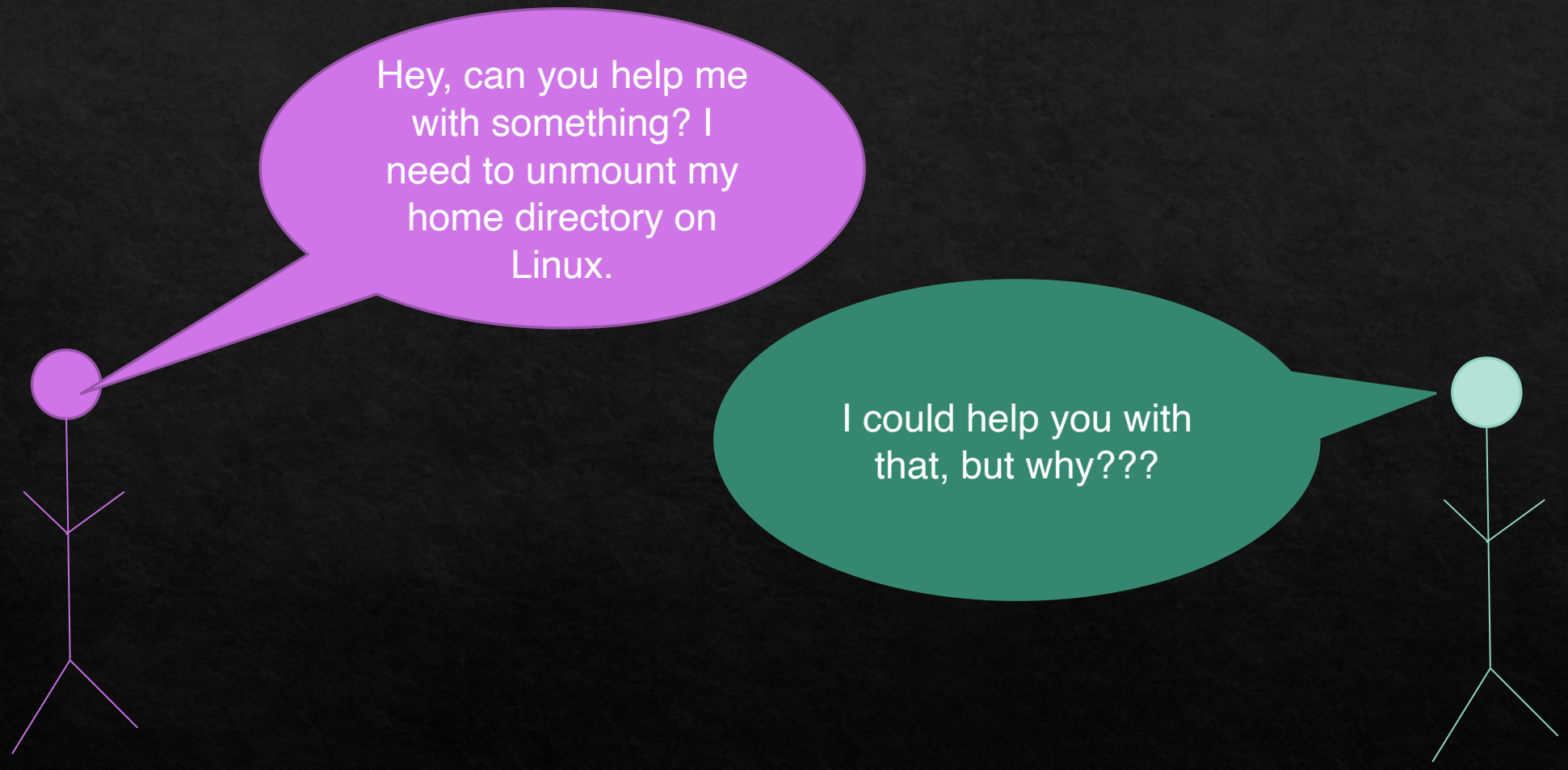
- Detailed, clear list of requirements
  - All the things the software must do to meet the user's goal
- Write them down
  - Go over the document with your customer
  - Make sure you agree on them & what they mean
  - Adhere to spirit as well as letter of requirements


We need a plane that can fly at Mach 2.5!
Mach 2.5 is difficult and would be expensive. Can you help us understand why you need that?
So it can escape combat
Maybe we can meet those needs a different way

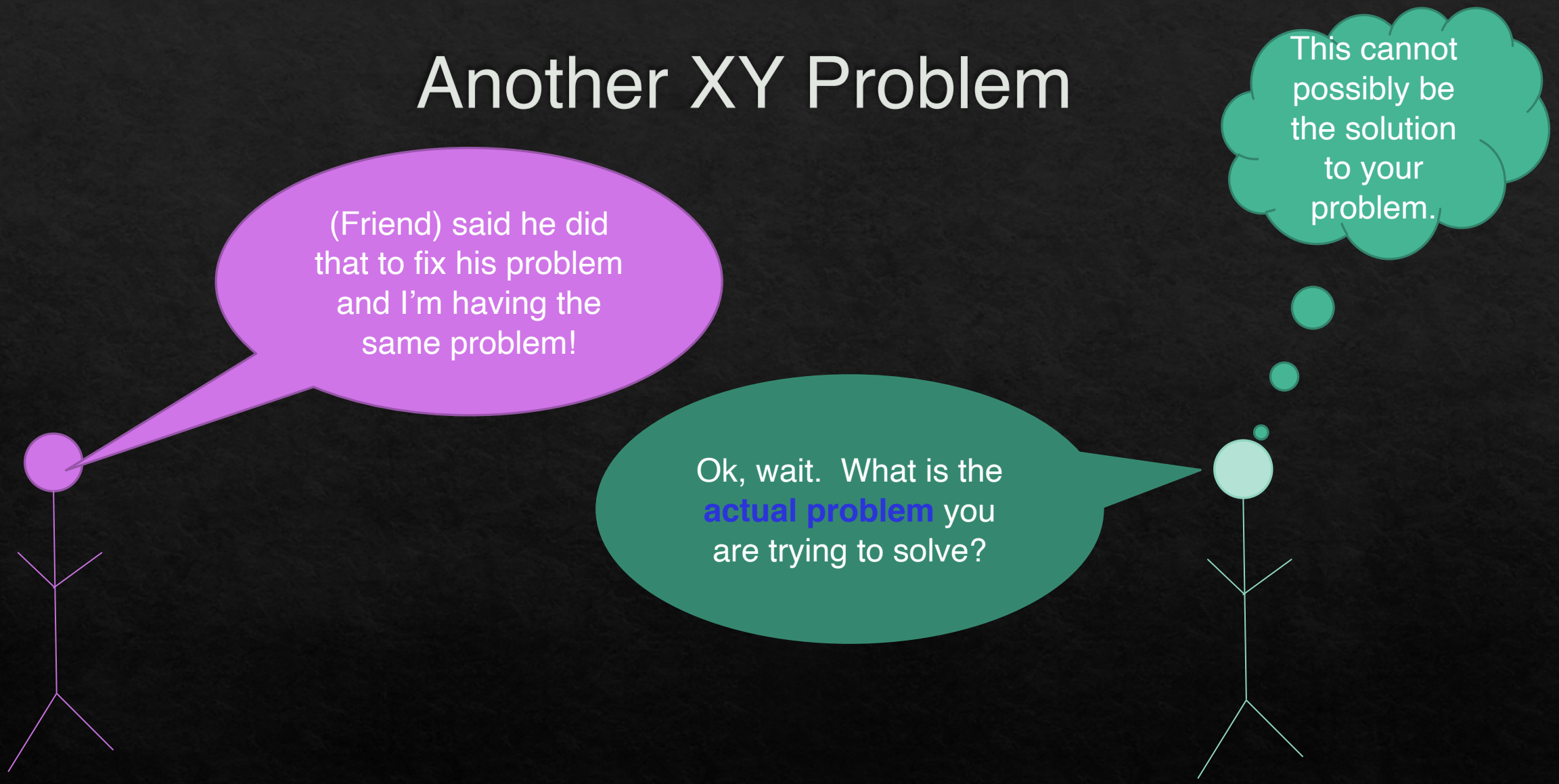
XY Problem

- User wants to do X (escape from combat)
- Doesn't know how to do X, but thinks that if they can do Y (fly Mach 2.5) that will let them work to a solution
- User doesn't know how/doesn't have a way to do Y
- User asks for a way to do Y
- Others try to help with Y, but (have problems/are confused/etc.)
- After much wasted effort, hopefully someone finds out that X is what is really needed.

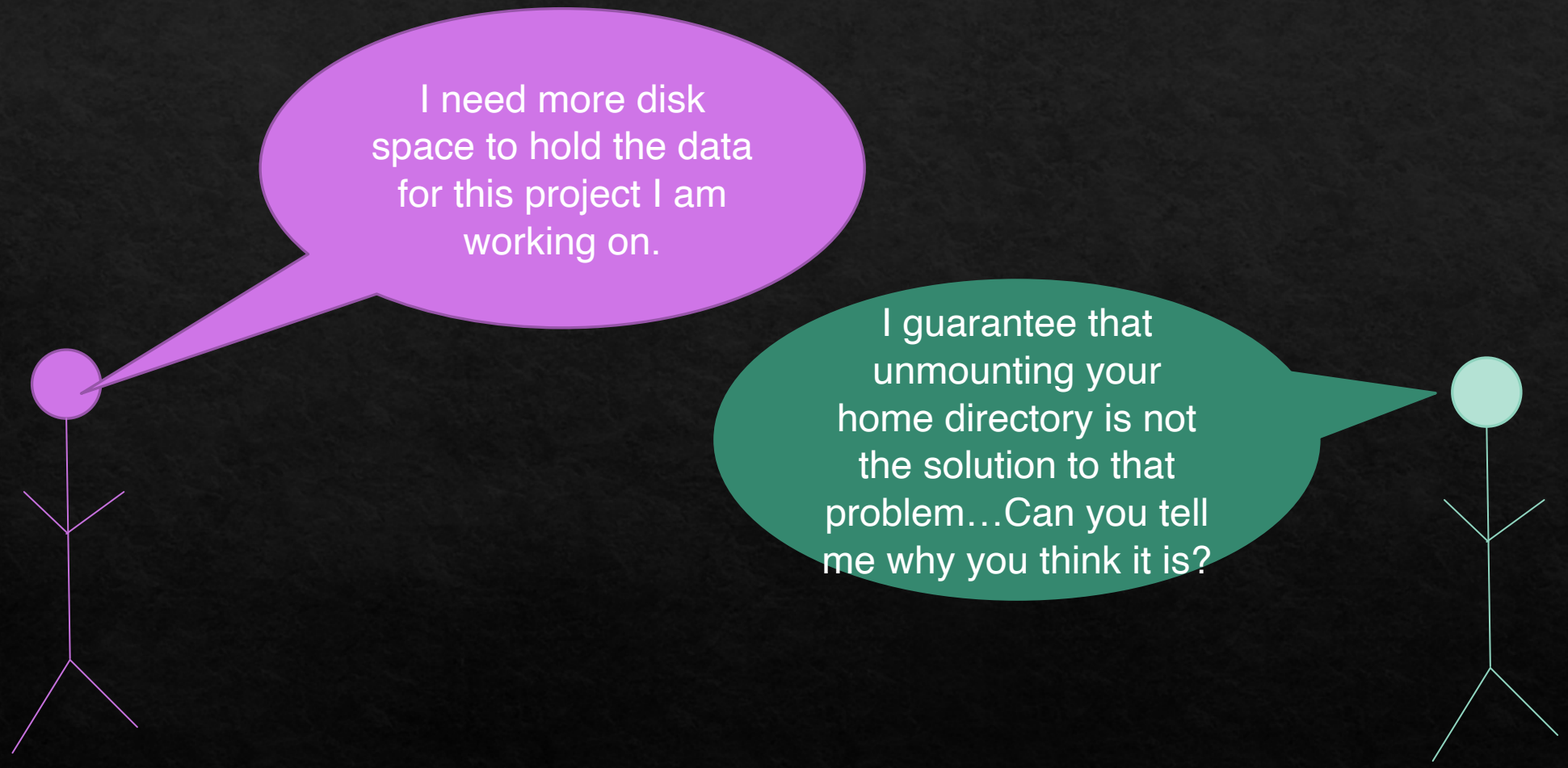
Another XY Problem



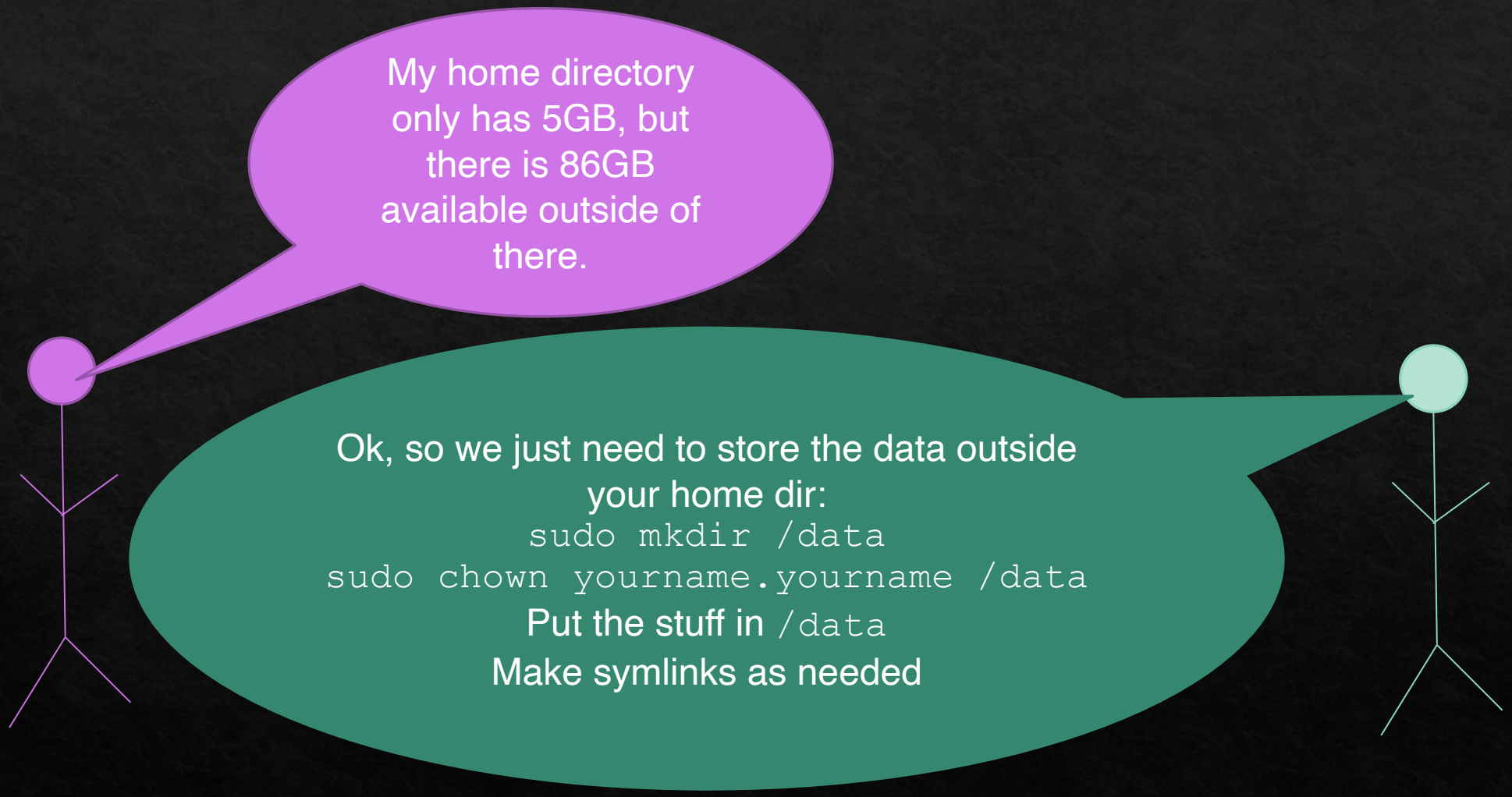
Another XY Problem



Another XY Problem



Another XY Problem



Requirements Definition

Your Job: Figure out customer needs and write them down clearly

Think, Pair, Share

Which of the following are good requirements?

Requirements

- After a successful login, the system shall display a page which will contain a summary of the user's accounts and a navigation menu.
- The system will be fast enough to support our users during the peak holiday season.
- The system will have an intuitive user interface.
- The system will use an AVL tree, keyed by the capacity of each ship to store information about the ships in the user's fleet.
- A user shall be able to look up a student by name or student ID, and shall be able to view the matching student's current courses and previous grades.

Requirement

After a successful login, the system shall display a page which will contain a summary of the user's accounts and a navigation menu.

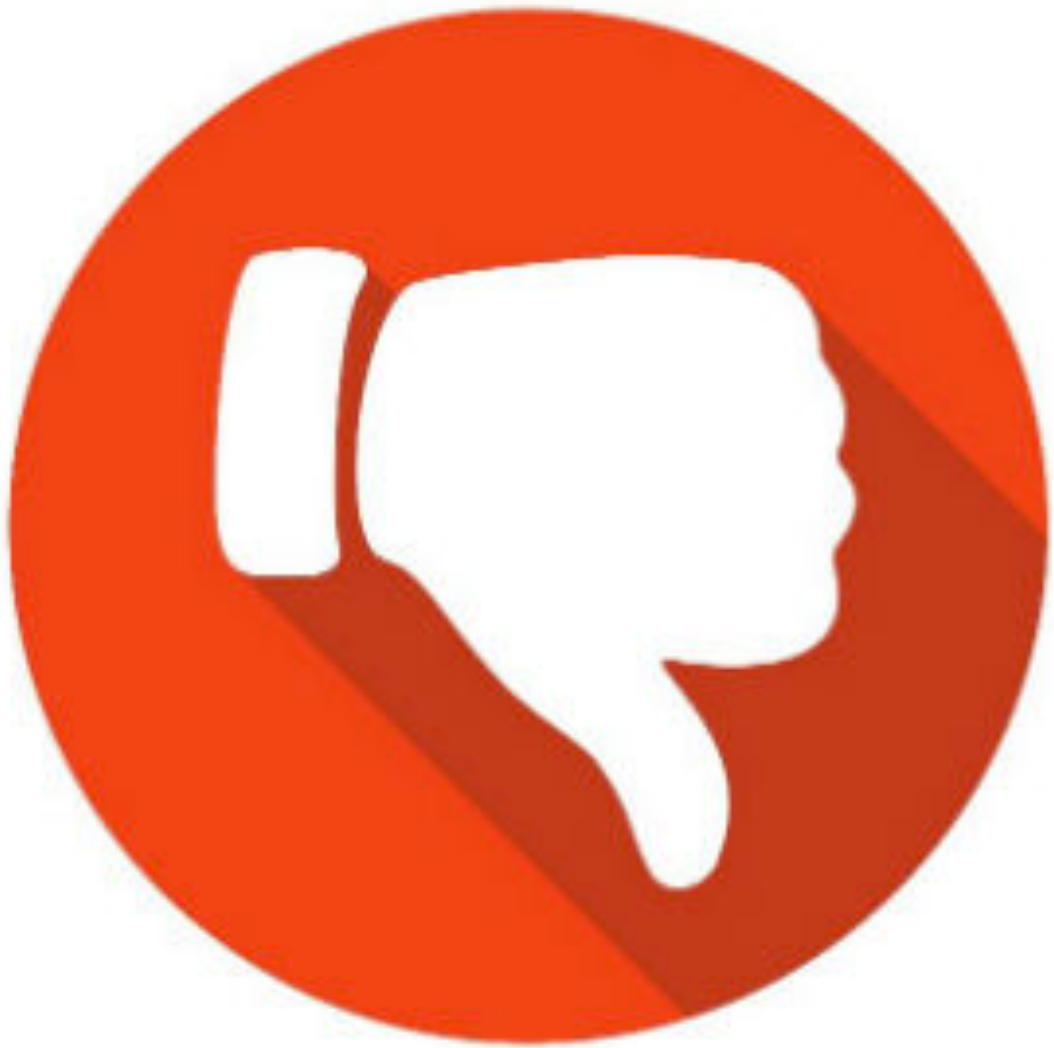
A user shall be able to look up a student by name or student ID, and shall be able to view the matching student's current courses and previous grades.

Requirements

-The system will be fast enough to support our users during the peak holiday season.

- The system will have an intuitive user interface

- The system will use an AVL tree, keyed by the capacity of each ship to store information about the ships in the user's fleet.



“Fast Enough”

- **Vague**: How do we know if we met this requirement?
  - What is fast enough?
  - How many users do we need to support?
  - What are their demands on the system?
- **Do we only care about peak season performance?**
  - Should we be able to scale up and down?

Refining “Fast Enough” Requirement

- The system will be fast enough to support our users during the peak holiday season.
- Under any load up to 2000 requests/second, the system shall handle at least 95% of them in under 400ms, exclusive of network latency to and from the client.

“Intuitive Interface”

- **Subjective**: We can disagree on if this is met
  - Oh it is very intuitive, to save and quit, you hit ESC : wq

Refining “Intuitive Interface” Requirement

- The system will have an intuitive user interface

- The system shall pass a usability study conducted from 100 users with no programming background by at least 95% of all users being able to perform the following tasks in under the specified times with no expert guidance:…

“AVL tree”

- This tells us **how** the software does things, rather than **what** it needs to do.
  - We should try to understand what the customer's actual goal is (i.e., efficient search over a range) and write requirement about that.

Requirements that Specify How

- When you find them, make sure you understand the customer's "WHY" and determine if it's a good reason.
- Example of good reasons
  - Compatibility
  - Licenses
  - Database structure
- Can still express these requirements in terms of what's needed
  - "Read data from the xyz table written by abc tool into the postgres db"

Requirements that Specify Standards to follow

- "Passwords must be salted and hashed with bcrypt"
- These standards are okay as long as the standard is the best
- Try to frame in terms of "what" and provide explanation for clarity
  - "Store passwords using secure hashing and salting. This shall be done with crypt unless superseded during the development period of this project."

Characteristics of Requirements

Measurable	We can tell if we met it Clear Objective
Descriptive	Of **what** the system should do, not how
Complete	No implicit understandings Everything written in the spec
Consistent	Don’t contradict each other

Requirement Categories

Functional

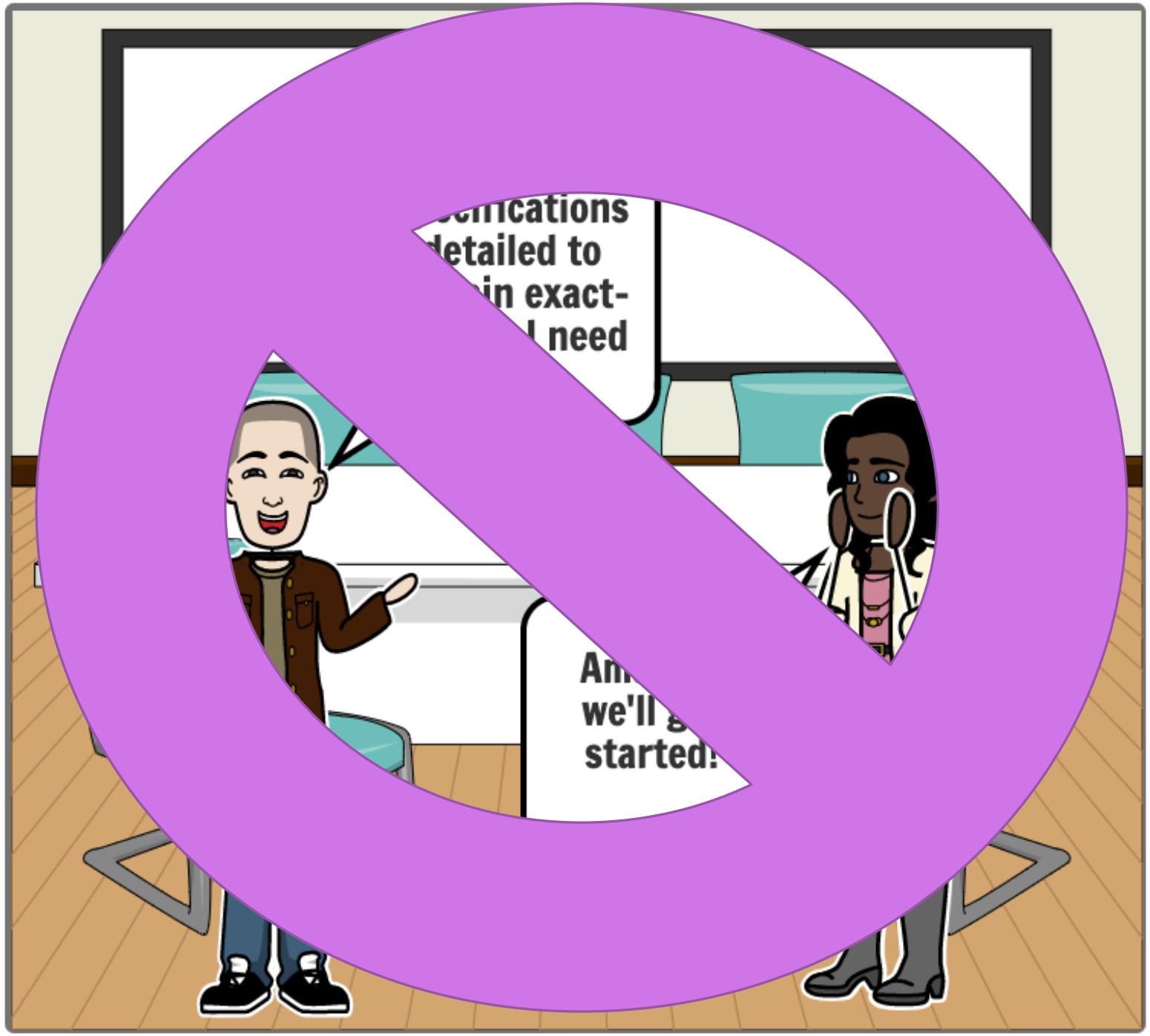
- **What the system does**
  - e.g., the elevator should take people to the floor
- **Has well-defined satisfaction criteria**

Non-functional

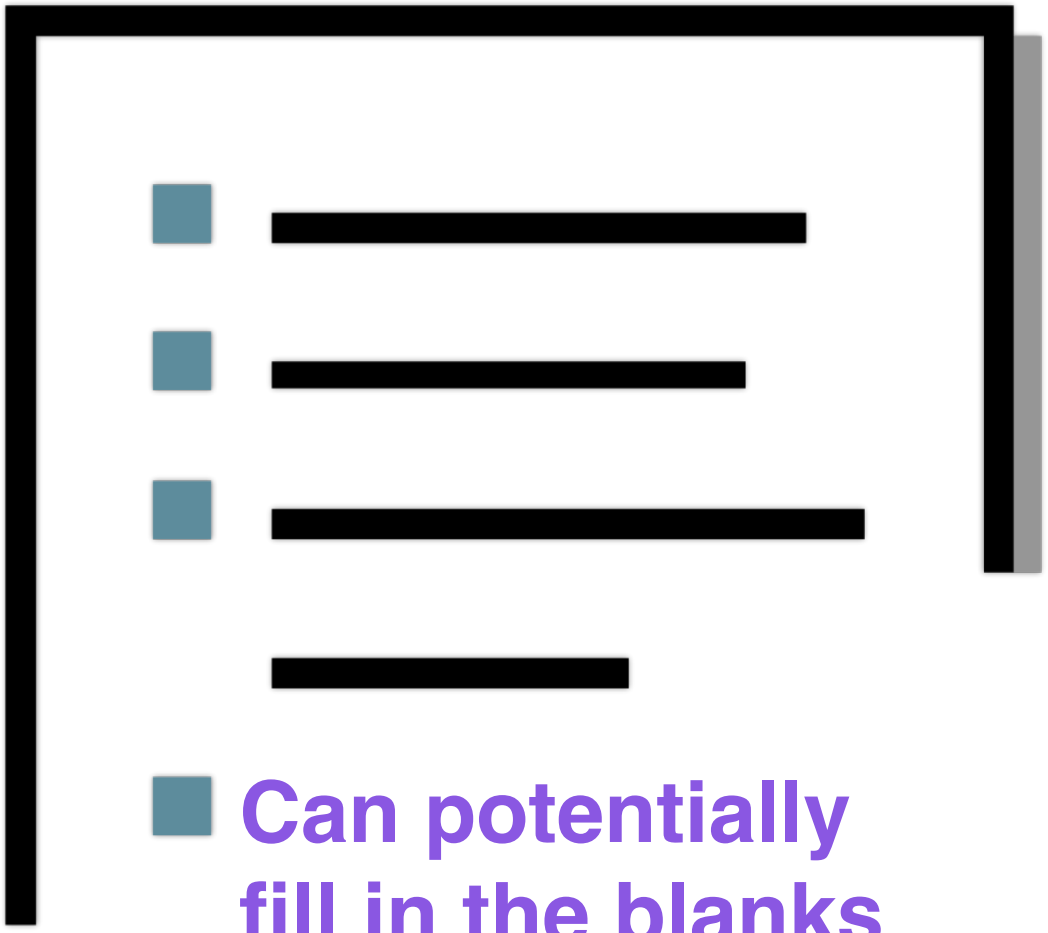
- **Refer to a system's qualities (e.g., reliability, usability, performance, cost)**
- **Make sure to clearly define satisfaction criteria**
  - e.g., "must reach the floor in less than 30 seconds" instead of "elevator is fast"
- Could also be documentation or process

Requirements Definition

Getting Requirements from the Customer



Customers often have a general idea of what they want, but they are missing details



Can potentially fill in the blanks by asking questions

Asking Questions

- Ask your customer about anything you **think** they need
- Clarify anything that isn't clear
- Ask about the performance
- Determine how they want to handle various situations
- Ask how secure the system needs to be

Why Ask Questions?

Prevents	misunderstandings.
Fills	in gaps of things customer might not know to say.
Avoids	assumptions in either direction.

User Stories

Professor Xavier

Professor Xavier runs a school for mutants. He is very busy teaching telekinesis but needs to know which students are misusing their superpowers. Whenever a student is caught misbehaving, the system sends him a text describing the incident, where it happened, and if it is under control. If it is not under control, Prof X calls in the senior X-Men to help him handle it.

Professor Xavier

Professor Xavier runs a school for mutants. He is very busy teaching telekinesis but needs to know which students are misusing their superpowers. Whenever a student is caught misbehaving, the system sends him a **text describing** the incident, **where it happened**, and if it is under control. If it is not under control, Prof X **calls in the senior** X-Men to help him handle it.

Time constraints on messaging? &
How are misbehaviors reported

Why User Stories?

Keep	products user focused.
Promote	cooperation between team members.
Prevent	feature creep and design dead-ends.

Caveats

Your requirements document will not be perfect

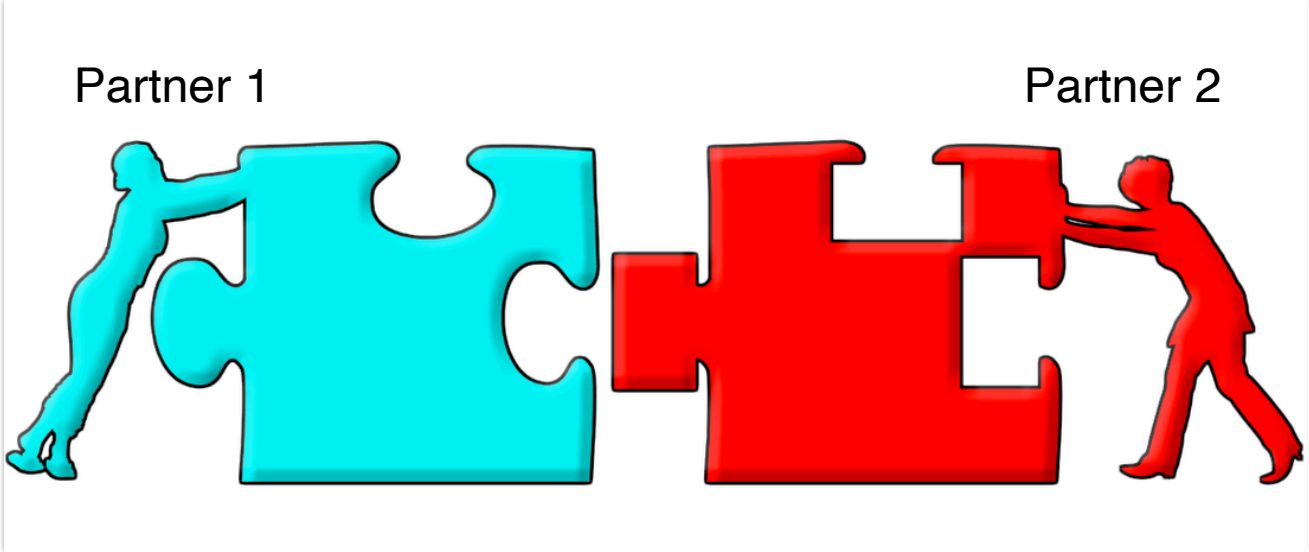
Customers change their minds

The Lesson?

- Software must be easy to change
- This constraint drives 95% of software engineering

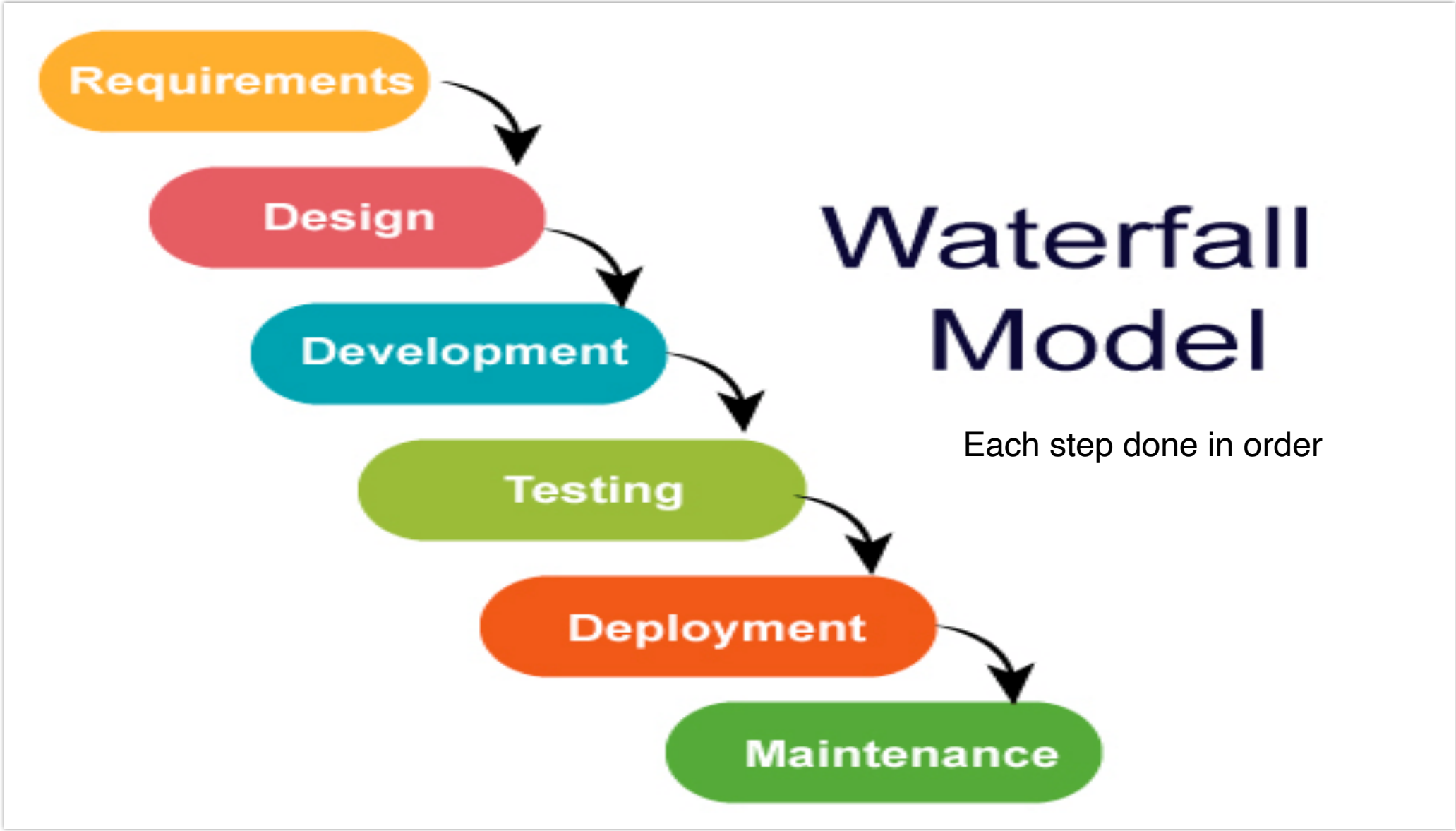
Software Development Process

Naïve Development Process

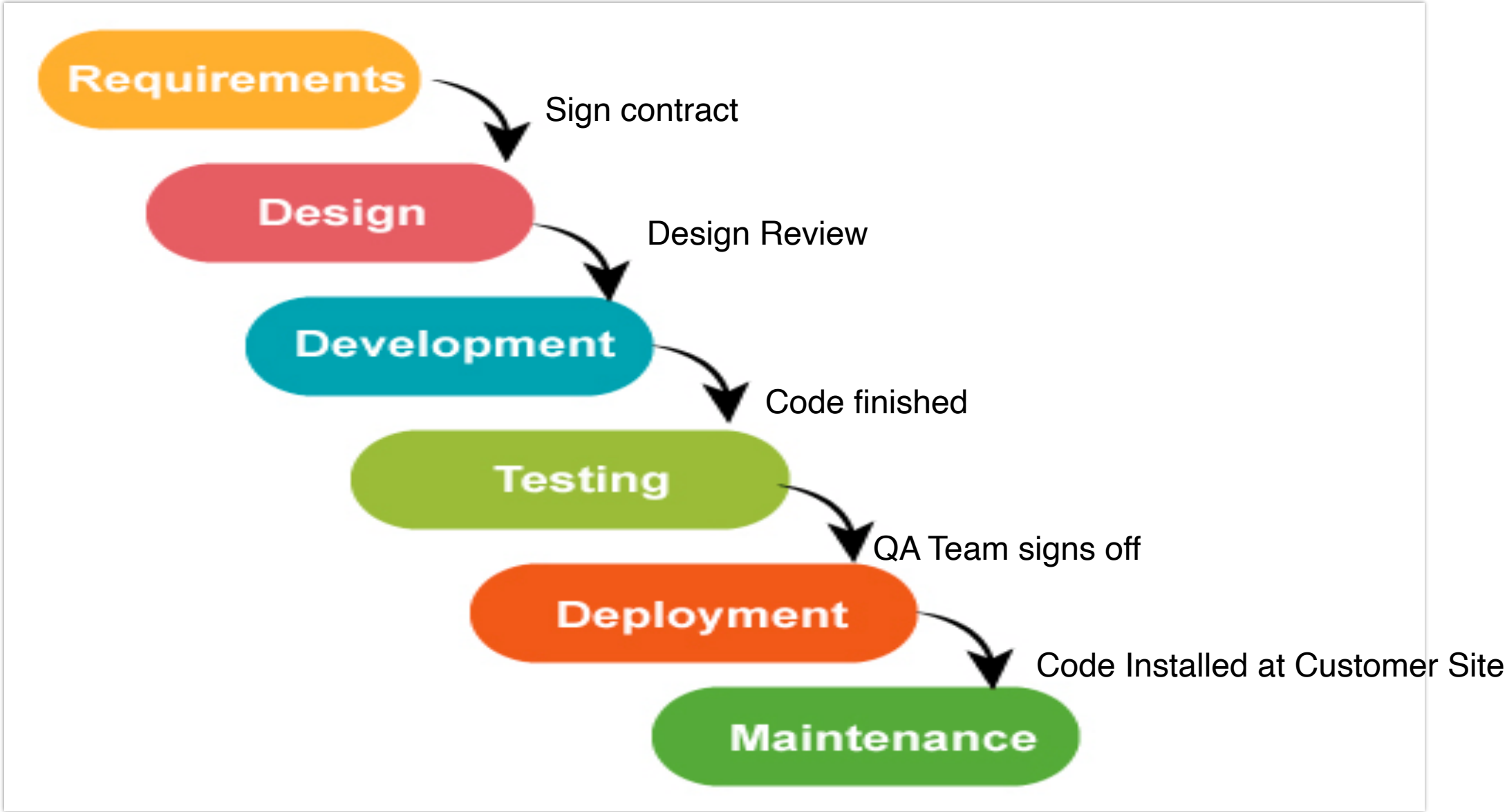


Skipping the design process and each partner creates own implementation & verification

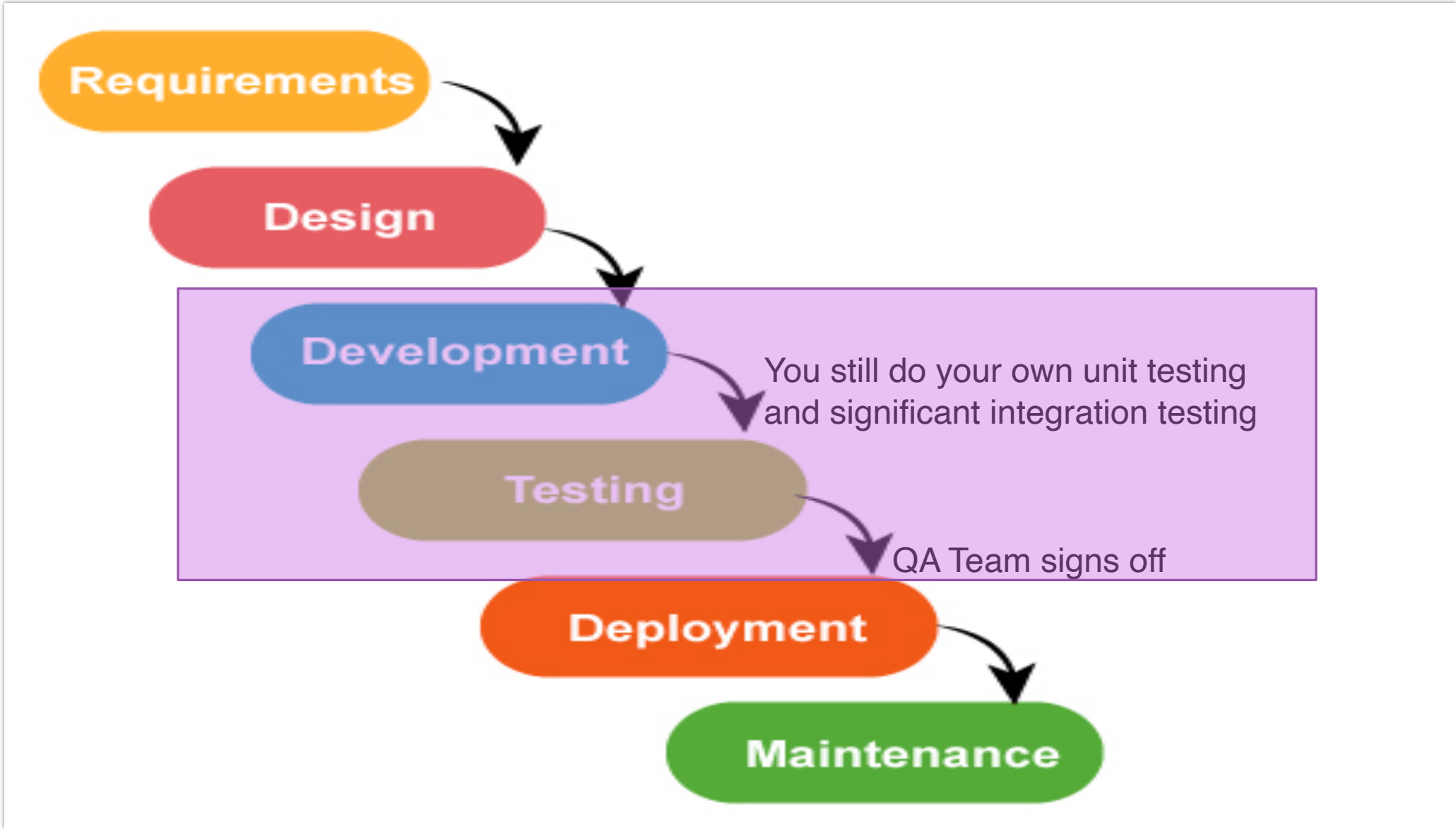
Image Source: http://www.thebooksofeden.com/the-balance-blog/category/all



Source: https://www.javatpoint.com/jira-waterfall-model



Source: https://www.javatpoint.com/jira-waterfall-model



Source: https://www.javatpoint.com/jira-waterfall-model

Waterfall Model

Advantages

- Clear step-by-step plan
- Complete planning, often with well reviewed documents from earlier parts
- Customer involved at start and end (might be the preference)

Disadvantages

- How handle requirements change?
- No feedback between stages: what if a choice was bad?
- Customer not involved in the middle.

Is waterfall the dinosaur of software development?

- In the past it has been “the way”
- Advantages for up front contact signing (e.g., cost estimation)

Image source: https://www.qwantz.com

What do you all think of each of these?

Source: http://danielcoding.net/scrum-part-1-agile-manifes

Individuals and Interactions over Processes and Tools

- Assumes you have a (very) good people
  - Good = expect, disciplined, motivated,…
  - Such people don’t need “guardrails” of processes
- Novices: need guidance on how
- Lacking discipline: need direction
  - How to do, when to do,…
- Lacking motivation: need to be moved along
  - Processes to get things done on time

Working Software Over Comprehensive Documentation

- You really need both, but…
- Given the choice of one you probably want software
- The point here is that software will evolve
  - So you don't want to spend a lot of time writing docs that will be outdated tomorrow
- This is NOT saying you never write documentation
- It is contrasting with previous approaches
  - Write massive pile of documents, then write code

Customer Collaboration over Contract Negotiation

- Great when you can have it!
- Depends a lot on the customer, consider…
  - Students doing summer projects for dept
  - Internal project for another team at company
  - Large government project

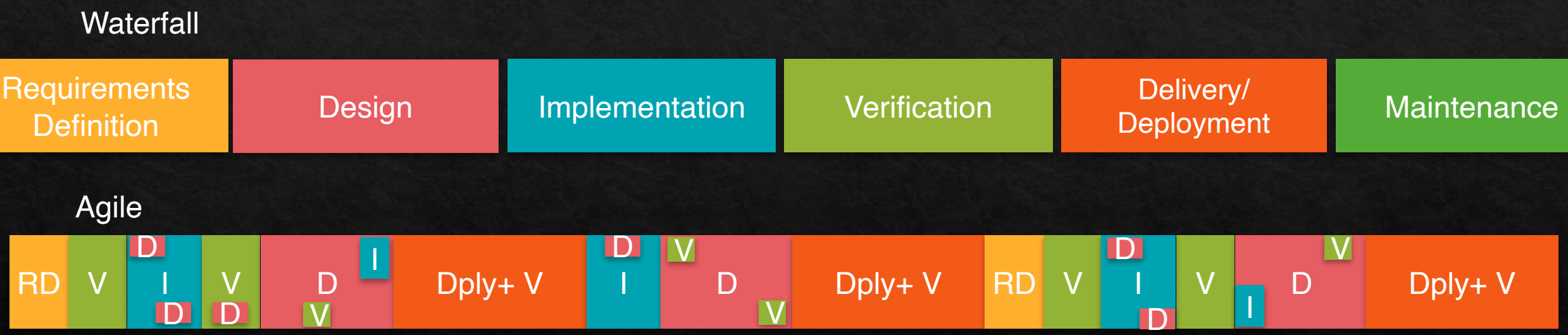
Responding to Change over Following a Plan

- Change is key!
  - We need to be able to respond to it
- We still need some form of plan
  - But do not try to plan too far out in too much detail
  - Near term: detailed
  - Further out: bigger picture, more flexible

Agile Project Management

- A collaborative, self-organizing, cross-functional approach to completing work and requirements
- Not a distinct methodology → category of methodologies
  - Scrum: concerned with getting more work done faster.
  - Kanban: primarily concerned with process improvement
- Characterized by
  - Simultaneous, incremental work
  - Adaptability

Hallmarks of Agile



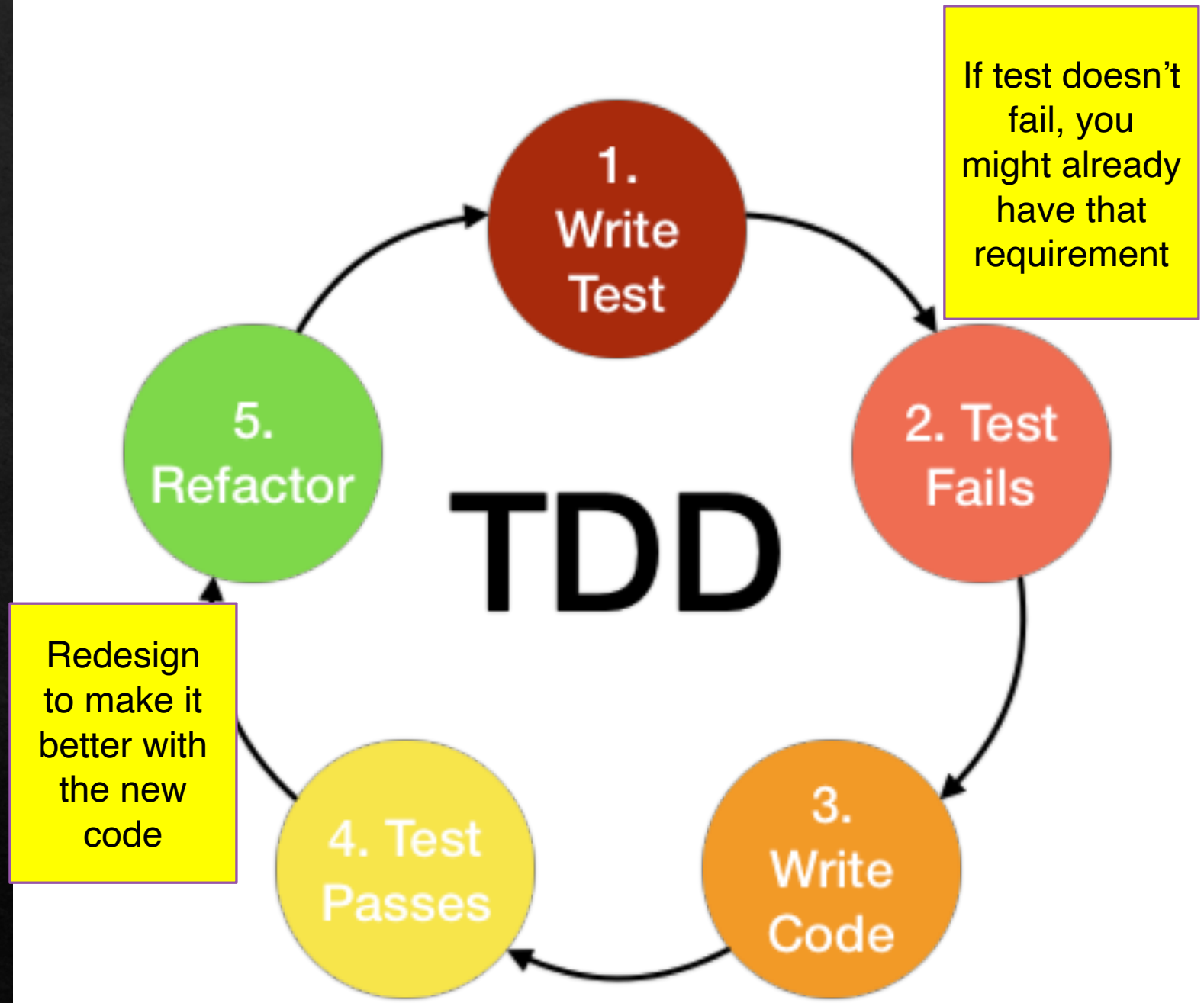
Stages mixed together. Done in small pieces

Does this make practical sense?

- Testing before implementation?
- Design during and after implementation?
- Deployment and testing done together?

Test-driven Development

Agile Process



Test Driven Development

- A TDD purist would say
  - Write only enough test to fail (might be “does not compile: class/method does not exist”)
    - Just write one test
  - Write only enough code to pass the test
    - May not be complete/generic: might just have one hardcoded answer
  - Add another failing test + repeat
- IMHO this is a bit extreme:
  - Write hardcoded throw away code to pass one test case = ☹
  - But writing some tests first has value
  - Even if you do not do TDD, test quickly!

Continuous Integration/ Continuous Deployment

- Push to ”staging branch”
  - Deploy to test server
  - Ran integration test suite
- Fix anything that breaks (or revert changes)
- Push to Master to deploy to production
  - Confident that everything is good
  - Staging and production branches have same set up

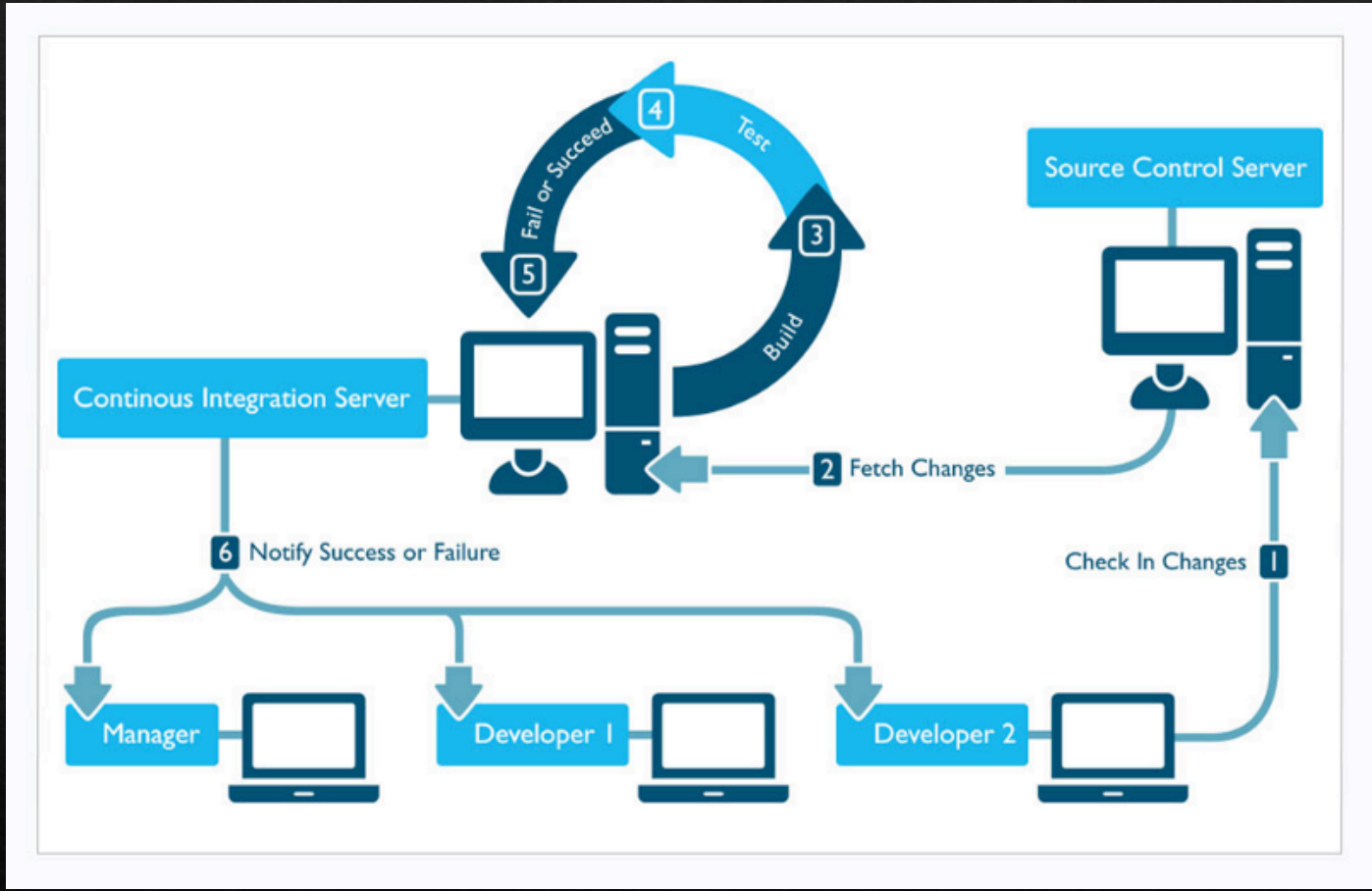
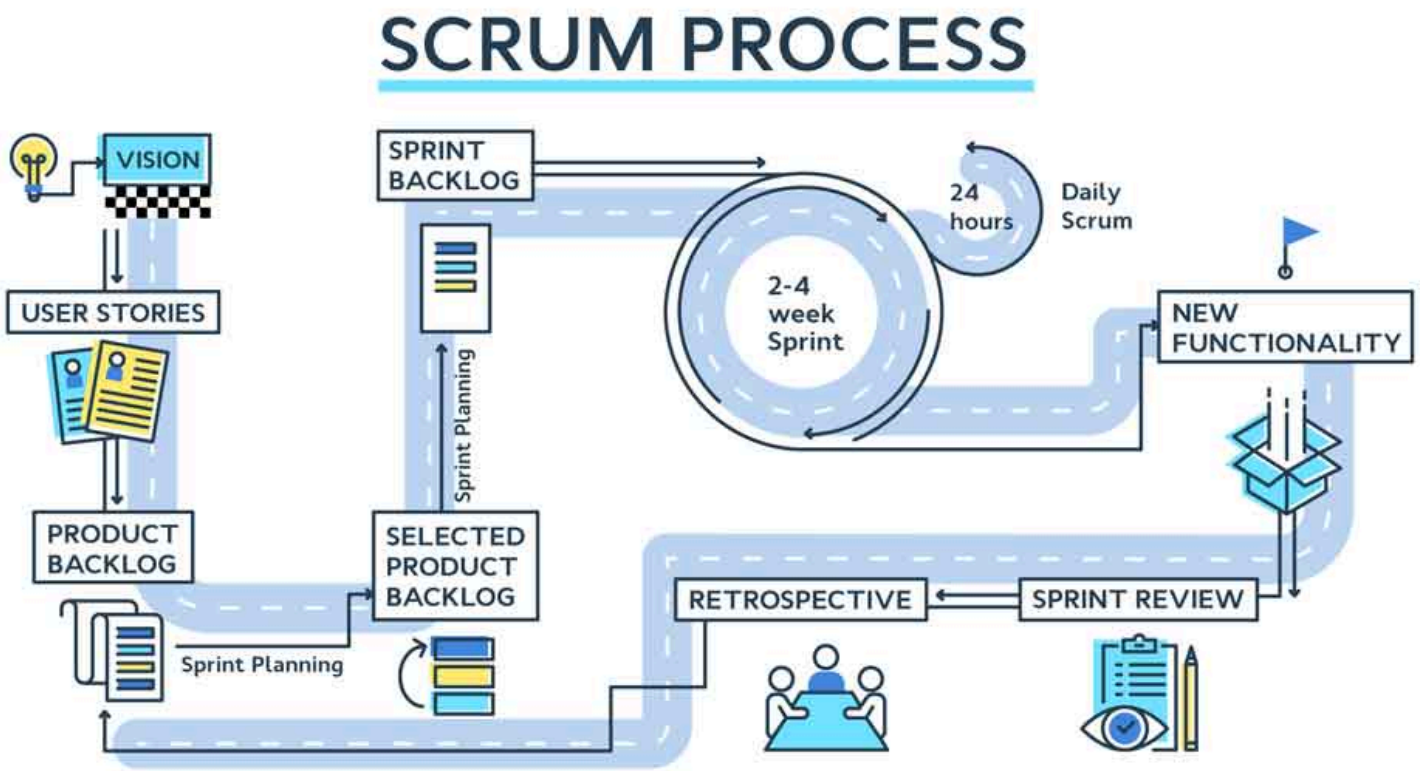


Image source: https://developers.redhat.com/blog/2017/09/06/continuous-integration-a-typical-process/

Agile Methodology: Scrum

Short “Sprints”
- Set goal at the beginning
- Have a deliverable at the end including documentation

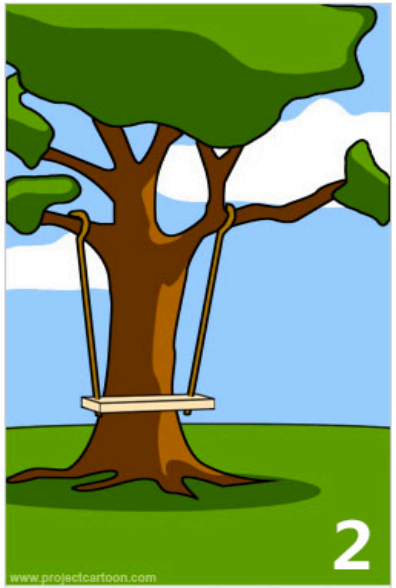
Meeting style
- Daily
- Short (15 minutes)
- Brief on accomplishments, where stuck, etc.



Agile Myths vs. Reality

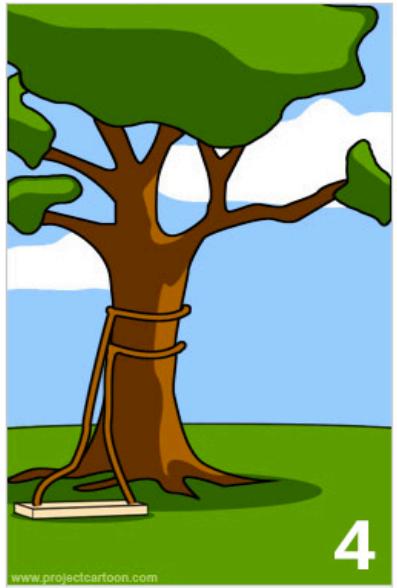
Agile means no requirement specifications	Specifications done in smaller pieces
Agile means no design	Design throughout project instead of all at the beginning – more refactoring
Agile means no documentation	Documentation released with each sprint

How the customer explained it



How the project leader understood it

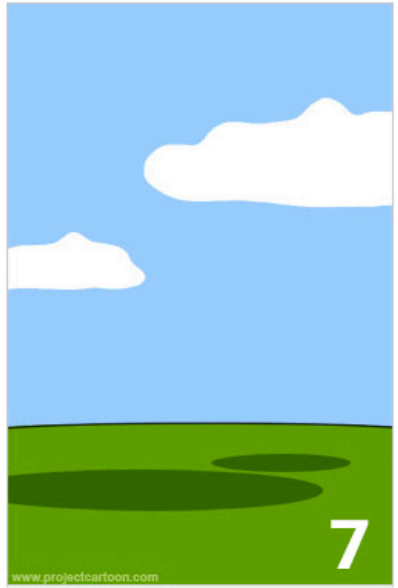
How the analyst designed it



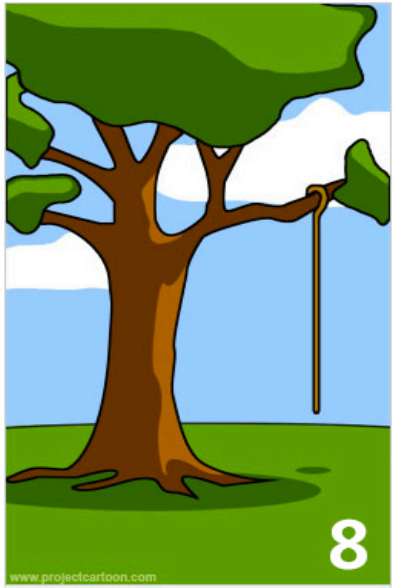
How the programmer wrote it

What the beta testers received

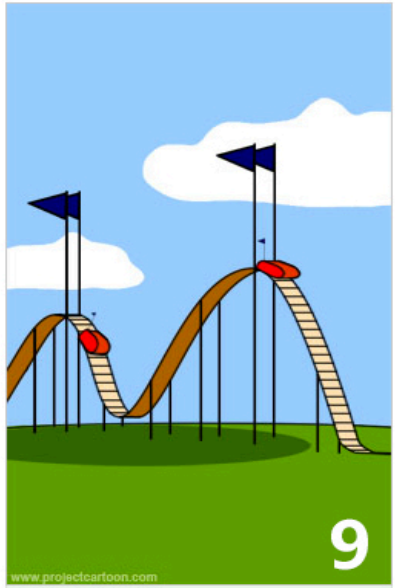
How the business consultant described it



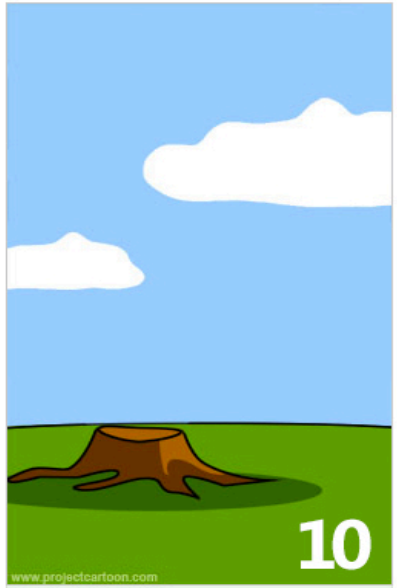
How the project was documented



What operations installed



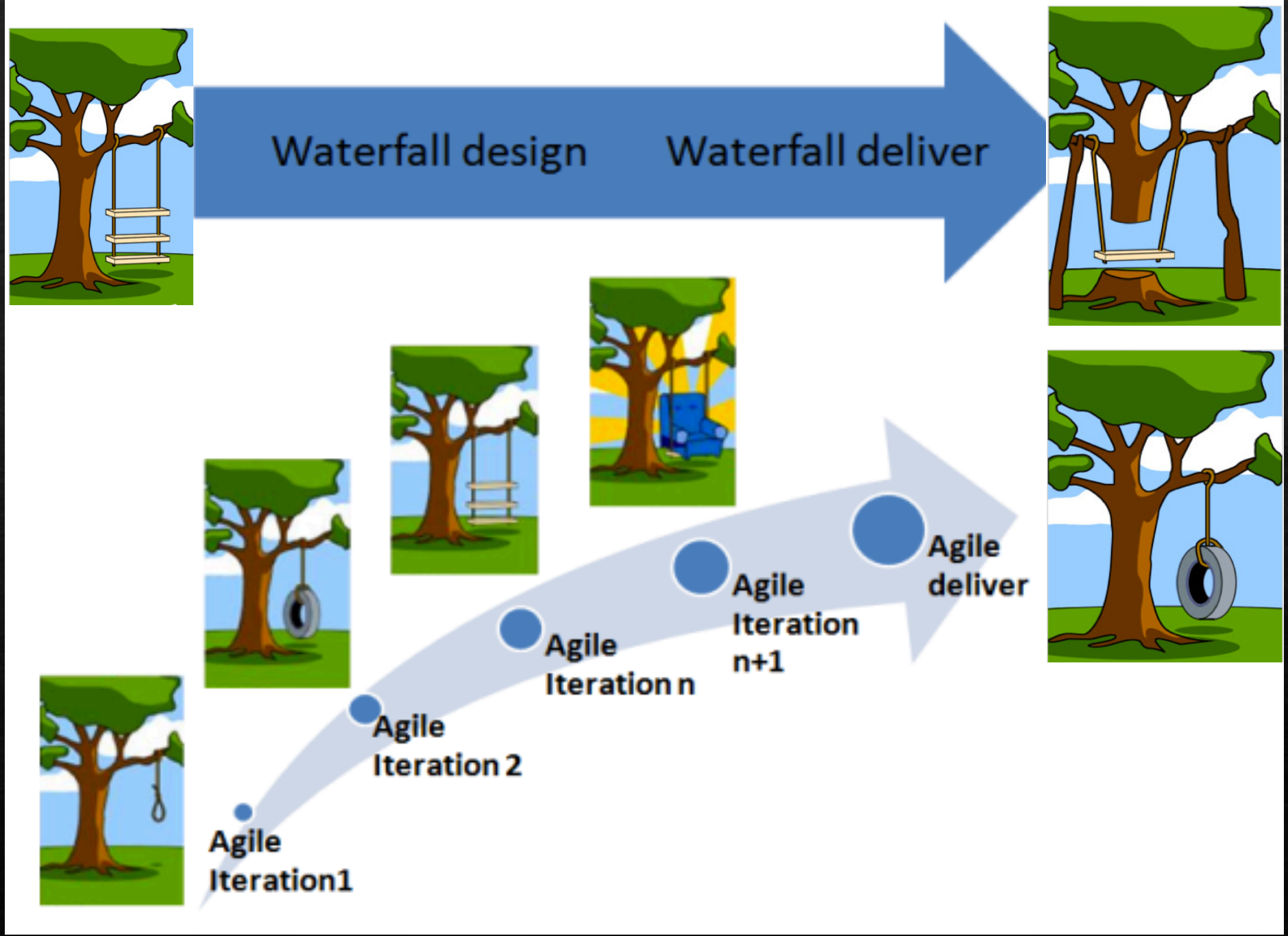
How the customer was billed



How it was supported

What marketing advertised

What the customer really needed



https://umangsoftware.wordpress.com/2014/06/17/agile-vs-waterfall-software-development-methodologies/

With Any SDP

Task Breakdown

Split into manageable pieces

Determining dependencies b/t tasks

Planning

Assigning tasks

Setting deadlines

Project Management

Checking progress

Adjusting when not on track

Task Breakdown

- Importance
  - Hard to estimate time/complexity of larger tasks
  - Larger tasks make assignment adjustments more difficult
  - Larger tasks amenable to “big bang” development
  - Gives a since of progress towards goal
  - Developers often underestimate task time in general
- Requires A LOT of thought about design
- **Guideline**:
  - Professionals: Break into 2 workdays (10-12 development hours)
  - Students: Break down into <1 hour tasks

Task Breakdown Example

“Sims”

Read from File

Receive User Input

Receive Messages from a server

Need to support some input formats, each of which has a variety of different field constraints

Walk to locations

Change Clothes

Purchase Item

Walk Dog

Need to change the state of the program in a variety of ways

Write to File

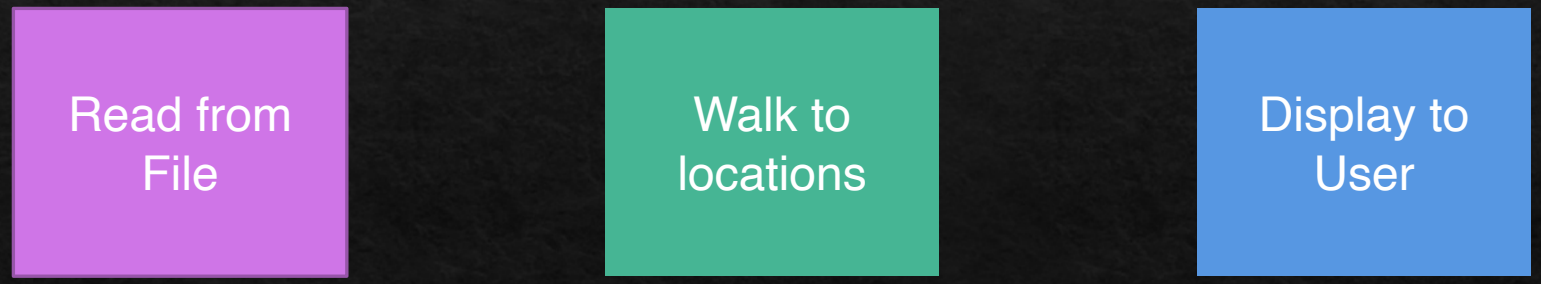
Send to server

Display to User

Need to produce some form of output

Good First Task Breakdown?

- One input, One action, One Output



- **No! Way too big**

Good First Task Breakdown?

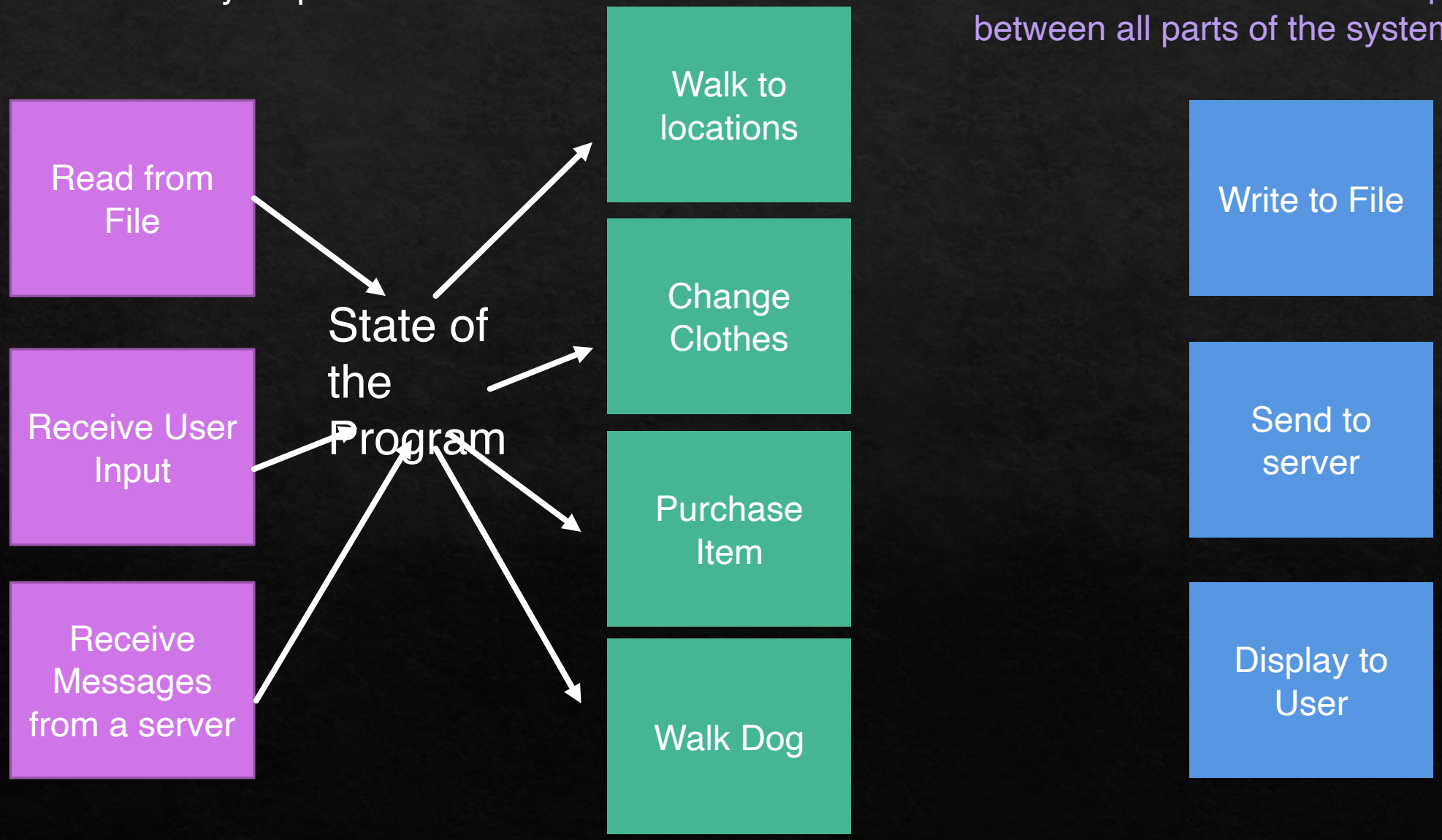
- One input

- **Probably still too big since file will have many features (e.g., people, buildings, animals, objects and descriptions of them)**

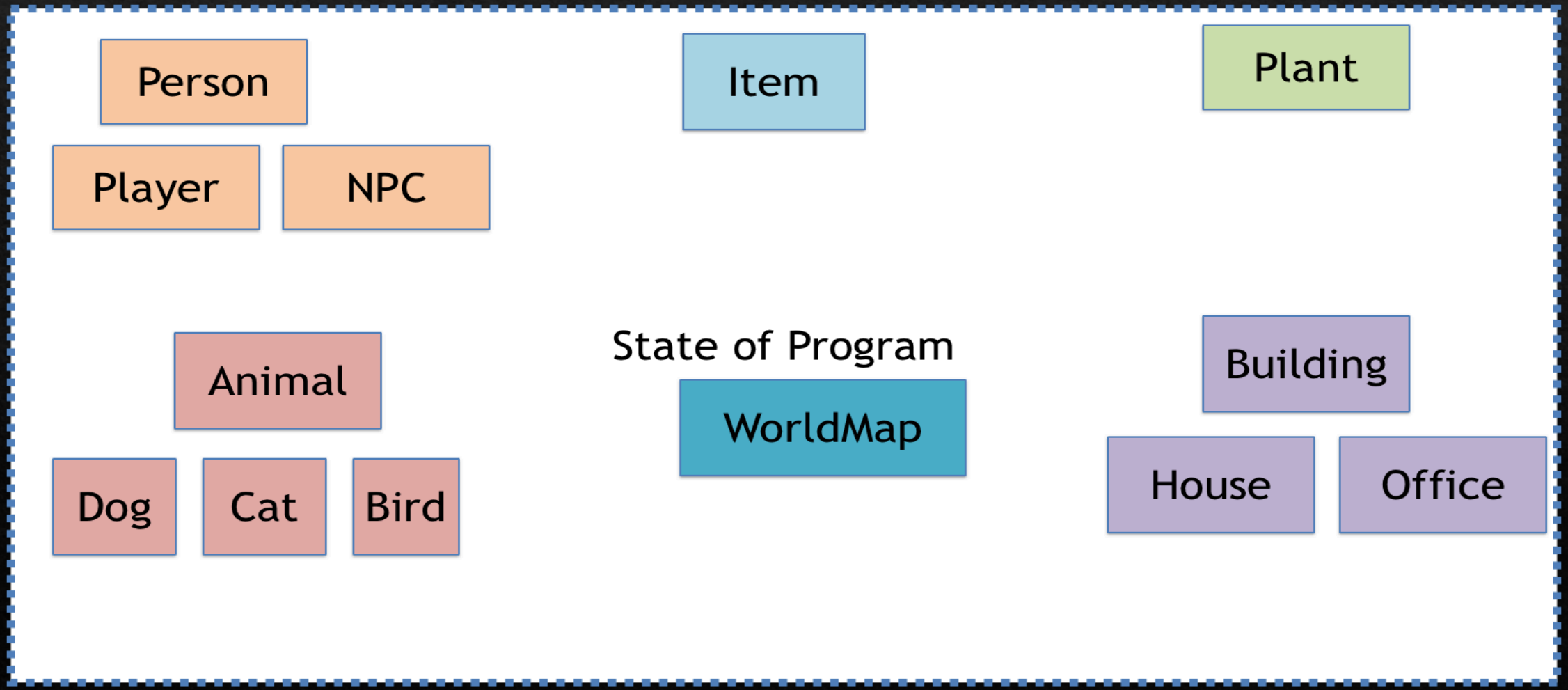
State of the program: all the classes that hold the data we work with (Person, World, Animal,…)

Produced by inputs
Modified by actions
Consumed by outputs

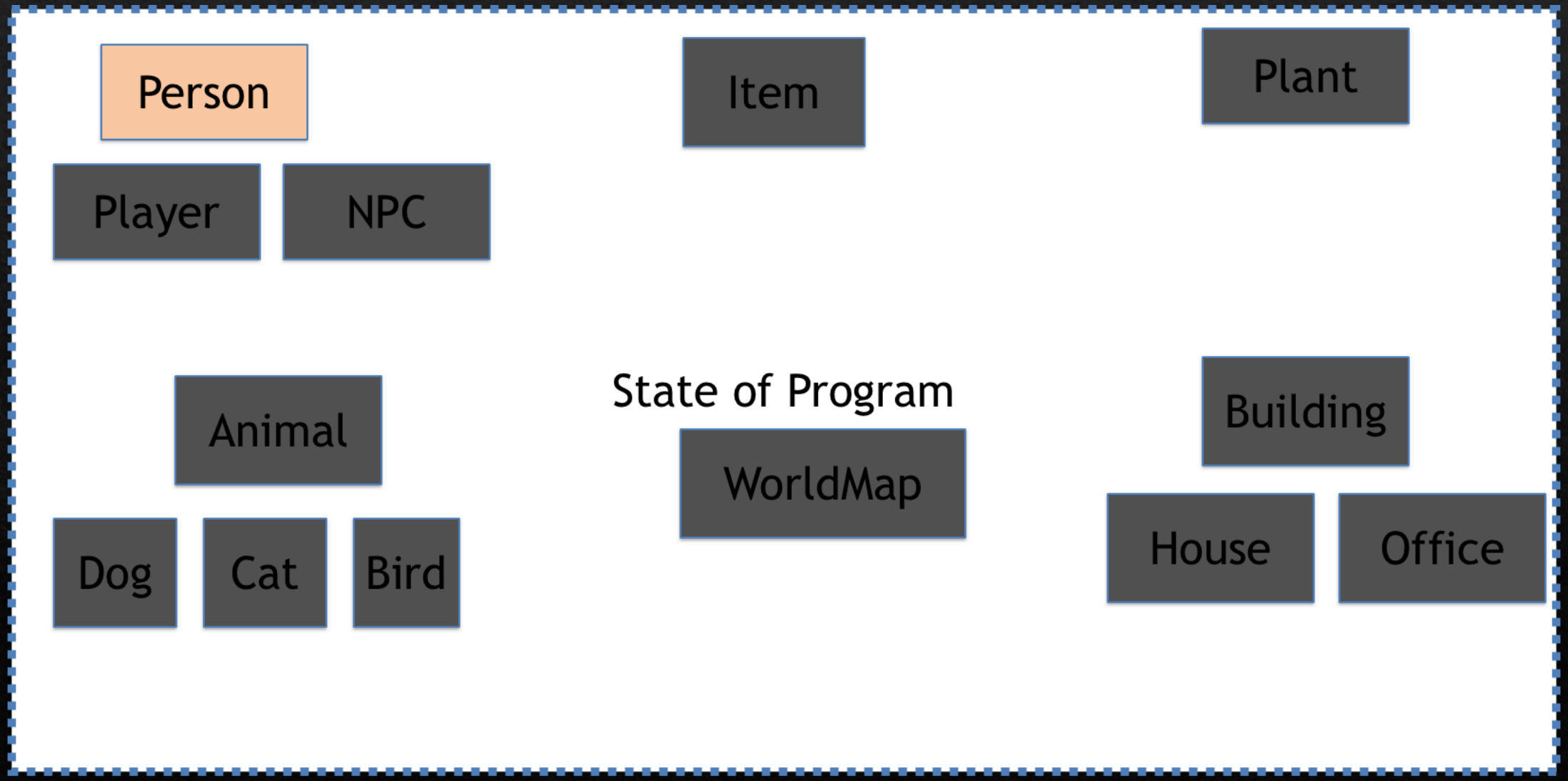
Observe that the **state** is the part of the **interface** between all parts of the system



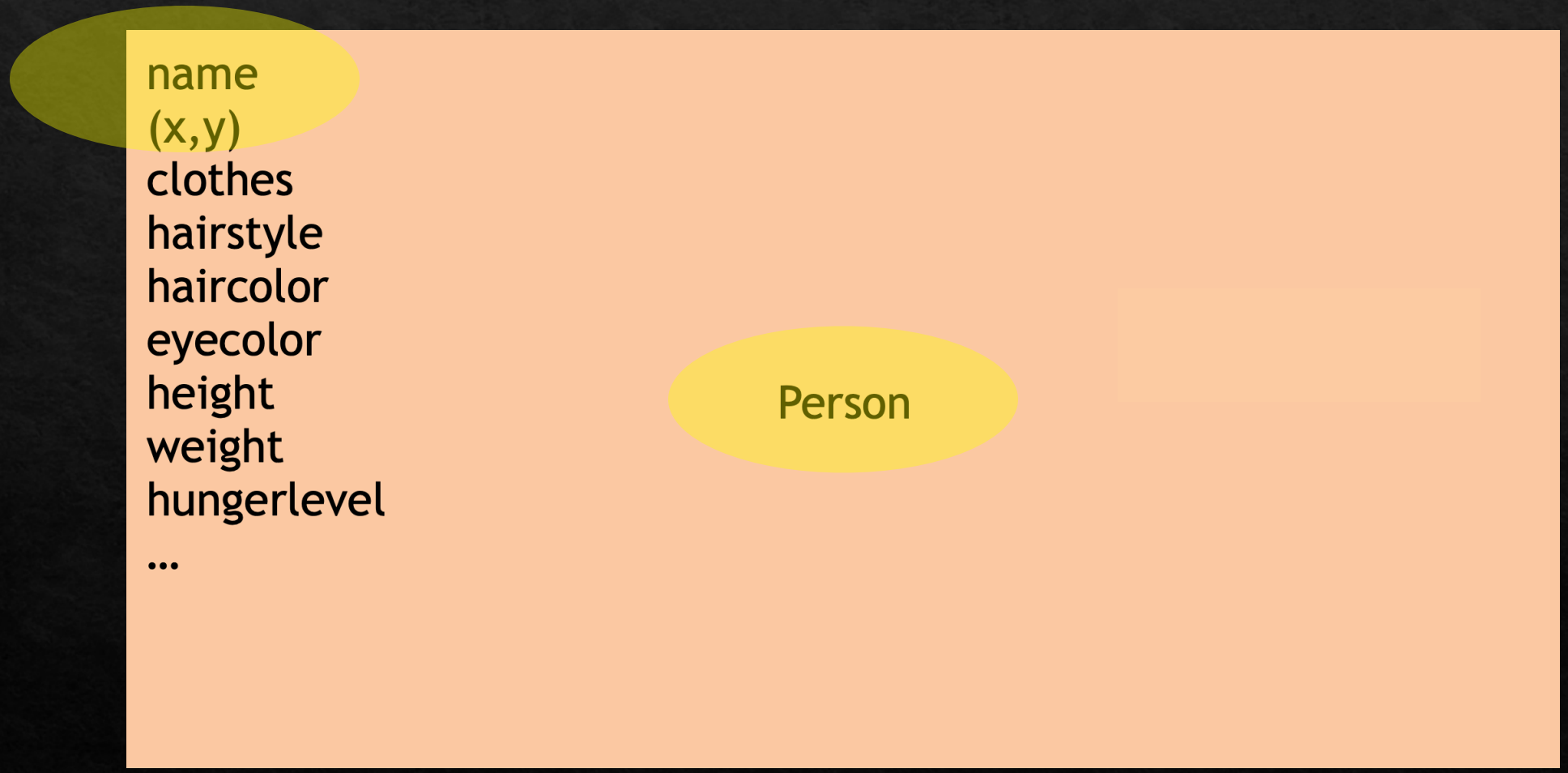
State is a Really Big Task Too…

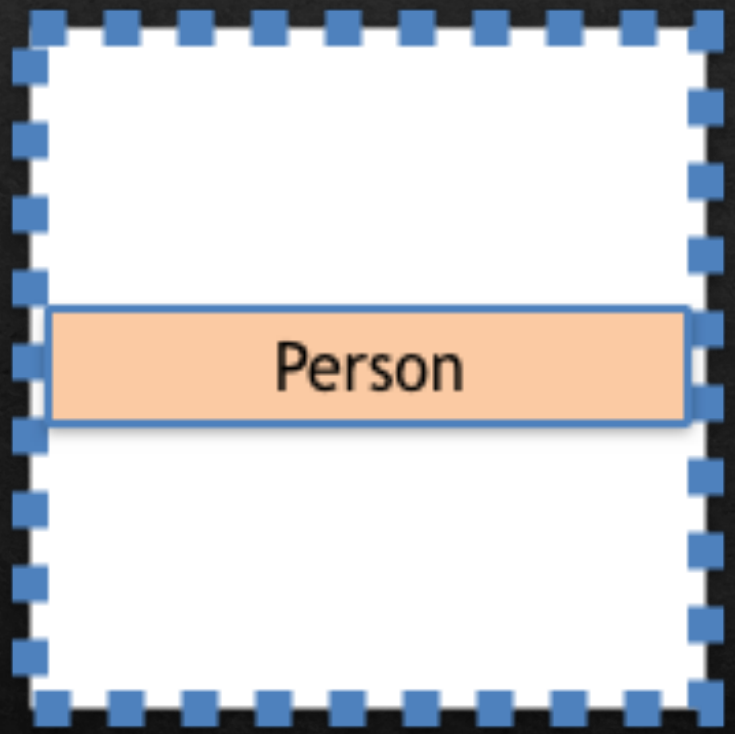


Small Subset of Program State

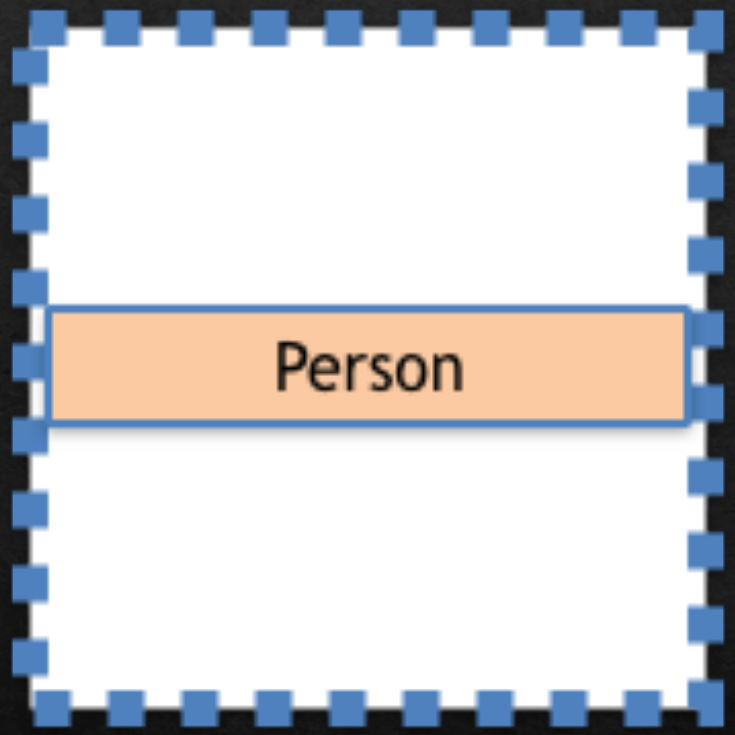


Person subset still very large



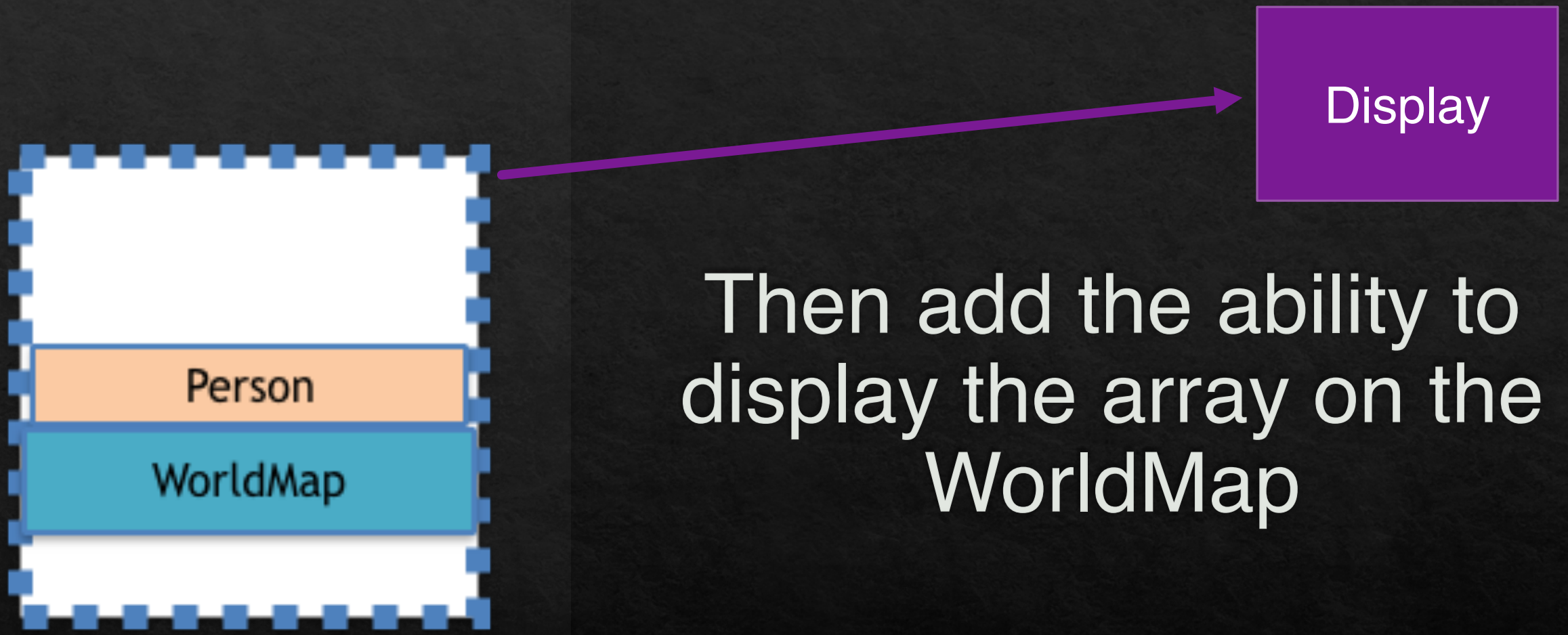


Person subset is very small, but it is a meaningful representation of state



1. Make Person subset
2. Test Person subset

Can potentially create a part of a WorldMap that can hold an array of people



Then add the ability to display the array on the WorldMap

Sims V 0.000000001 “code”

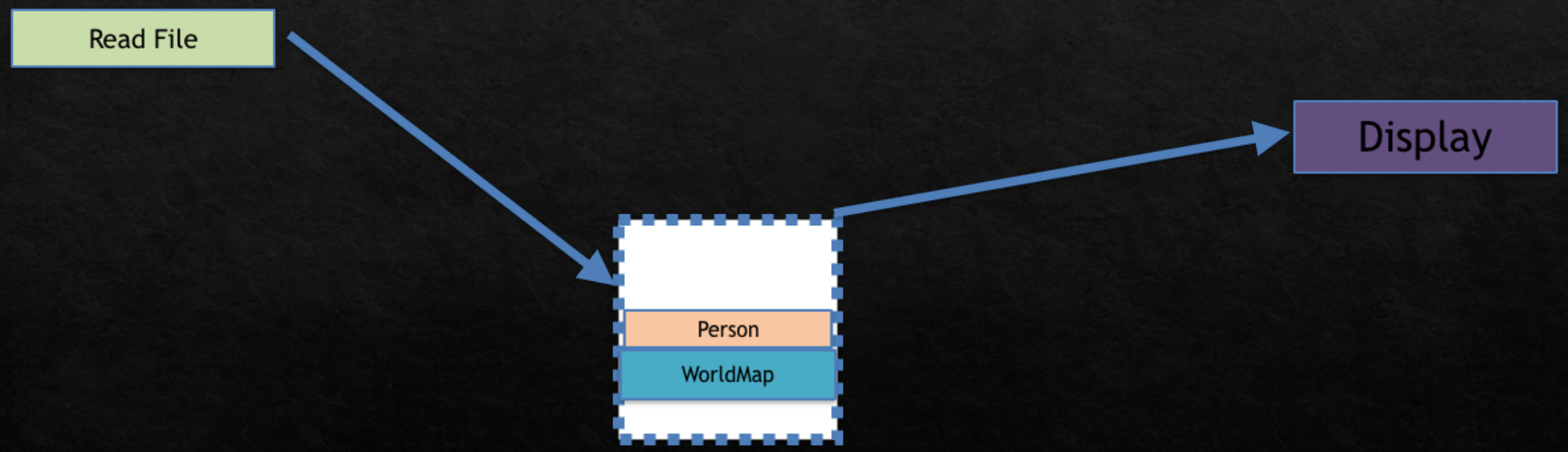
```
3  public static void main(String[] args){
4      Person p1 = new Person(“A”, 4,4);
5      Person p2 = new Person(“B”, 500,350);
6      WorldMap wm = new WorldMap();
7      wm.addPerson(p1);
8      wm.addPerson(p2);
9      while(!userQuit()) {
10         display(wm);
11     }
12 }
```

Sims V 0.000000001 “screenshot”

A

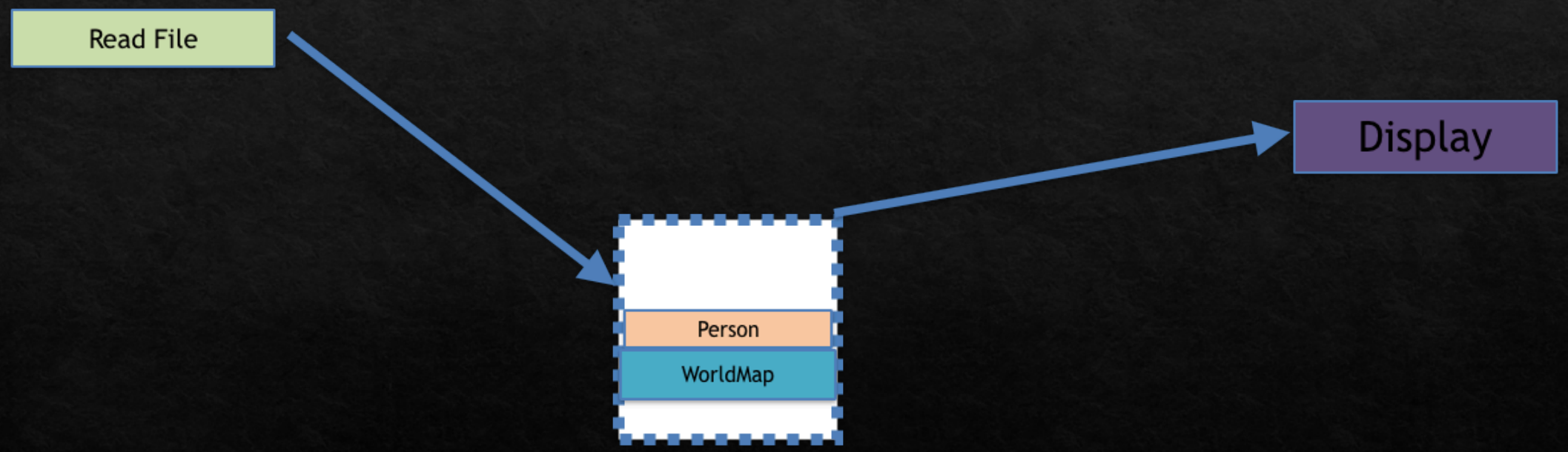
B

Then Add More…



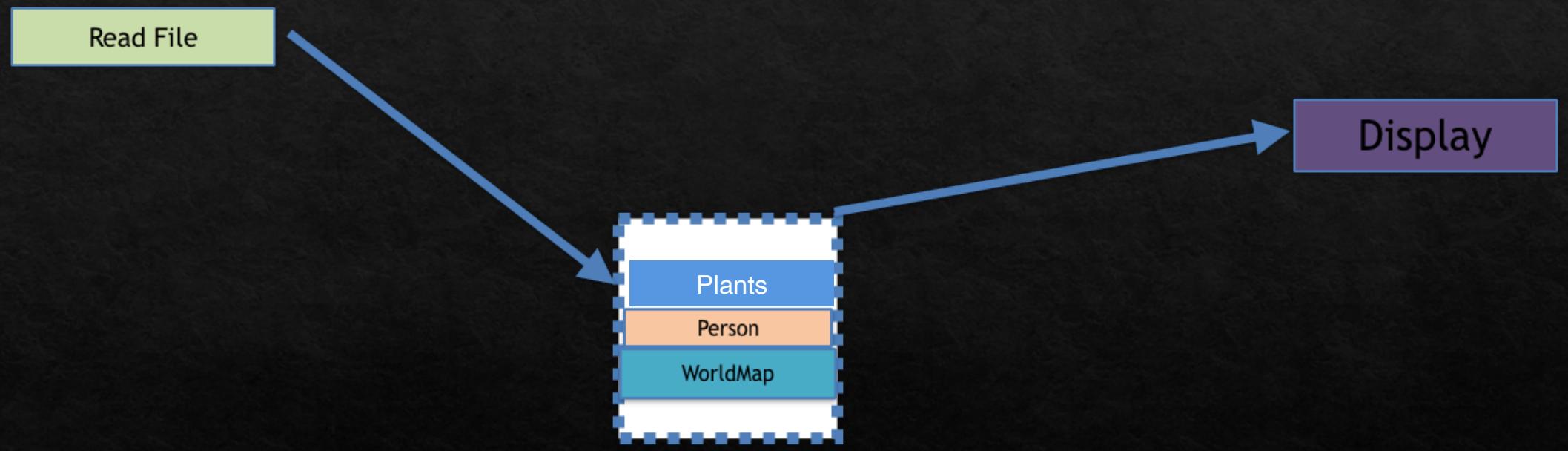
- Only reads people
- Only name + (x,y). Other fields not supported

This is an End-to-End System



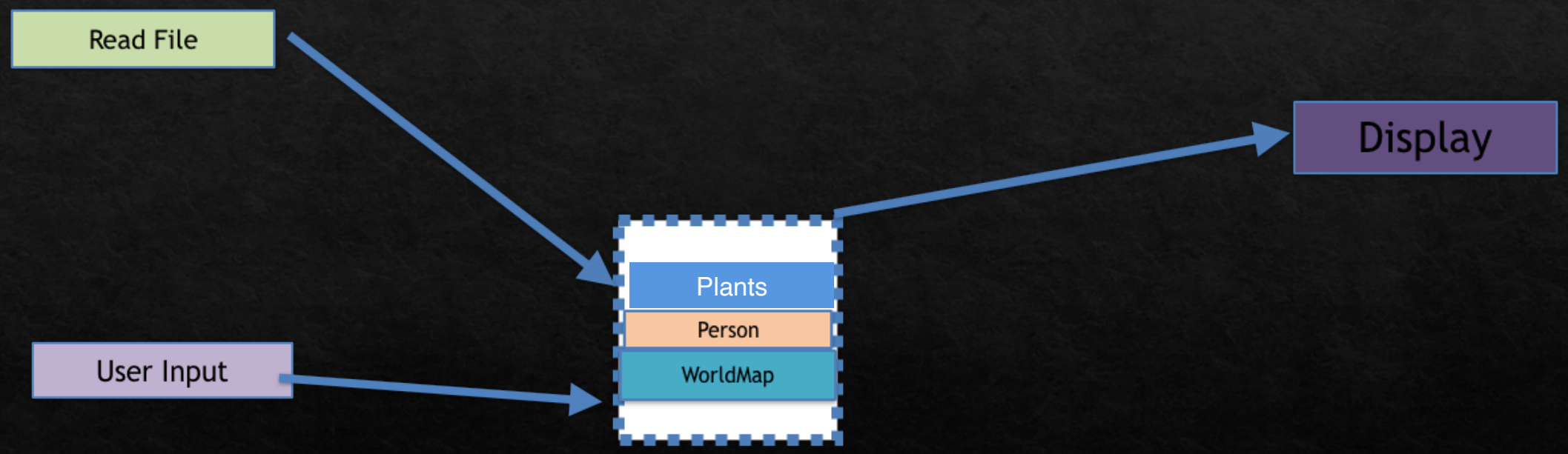
- Almost NO features
- But it goes start to finish

Why does an End-to-End System Matter?

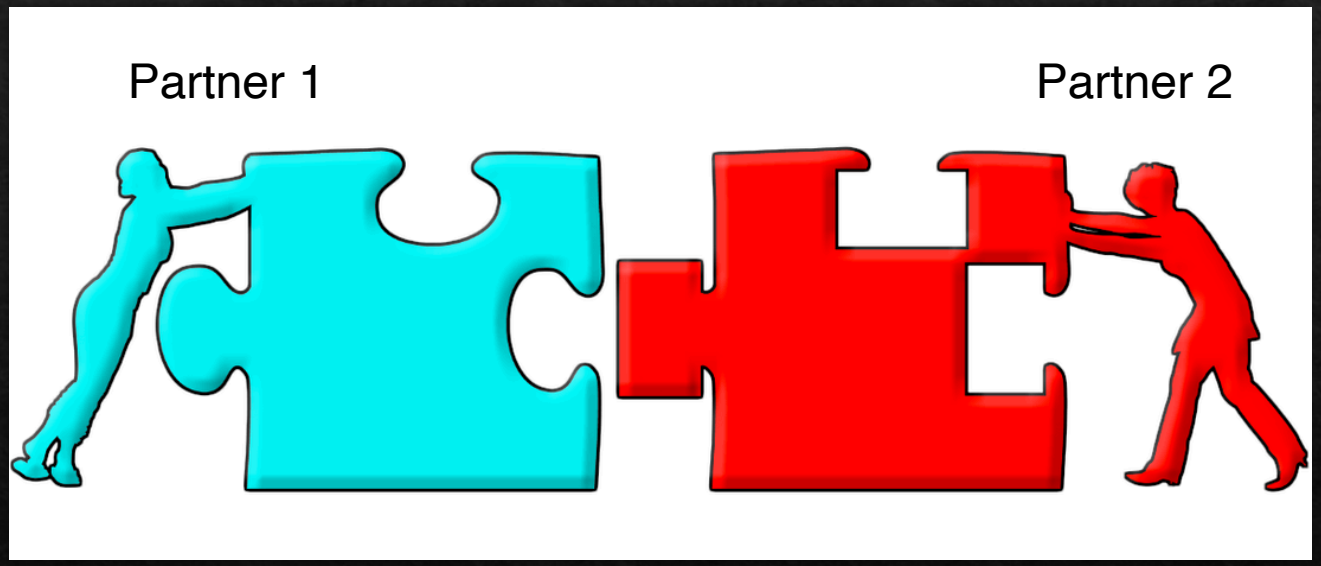


- Add features to the system and know right away if they “fit”

Why does an End-to-End System Matter?



Contrast Finding the smallest meaningful state and creating and end-to-end system with trying to integrate after a lot of code has been built

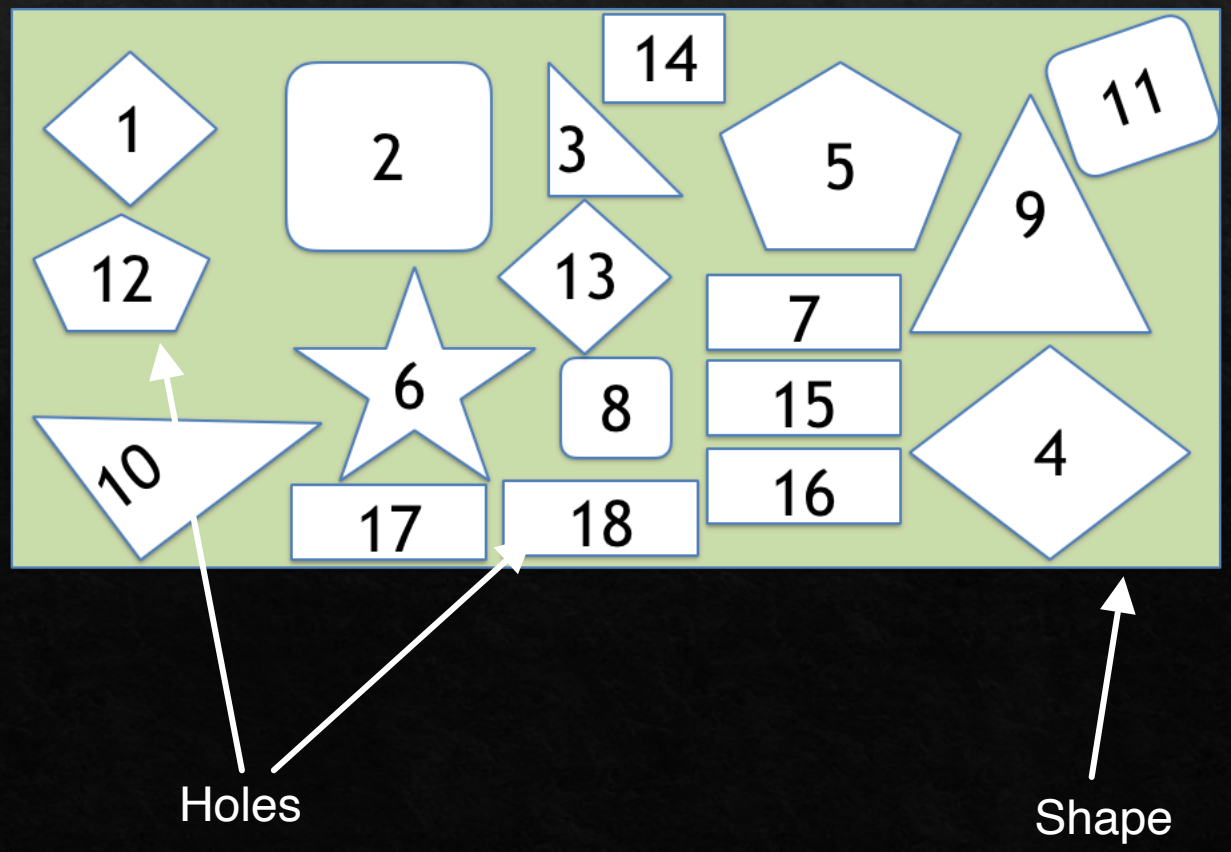


Skipping the design process and each partner creates own implementation & verification

Image Source: http://www.thebooksofeden.com/the-balance-blog/category/all

Instead…

- We have the right ”shape” , which is the end-to-end system
- With smaller “holes” (1-18) that we know definition of
- Our task is to create code for each hole and test immediately to see how well it fits.

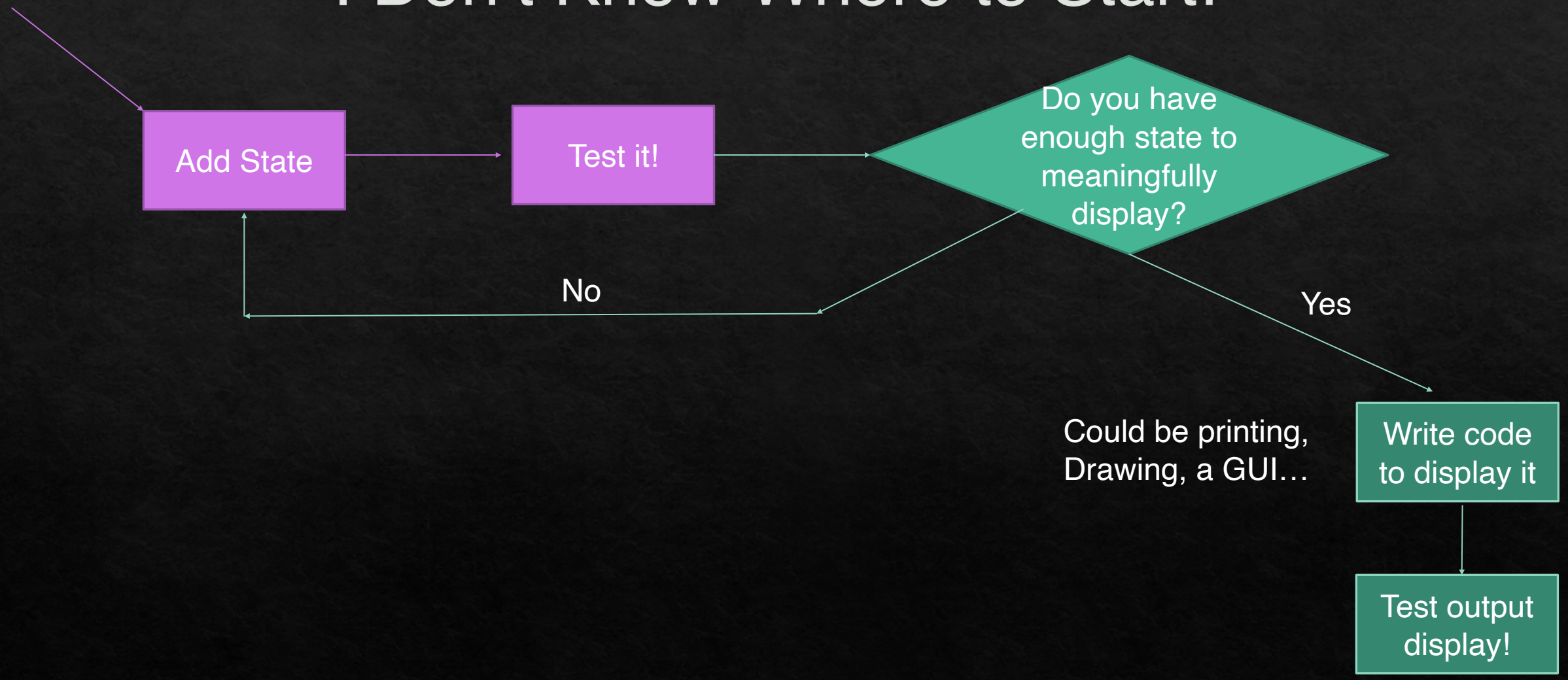


I Don’t Know Where to Start!

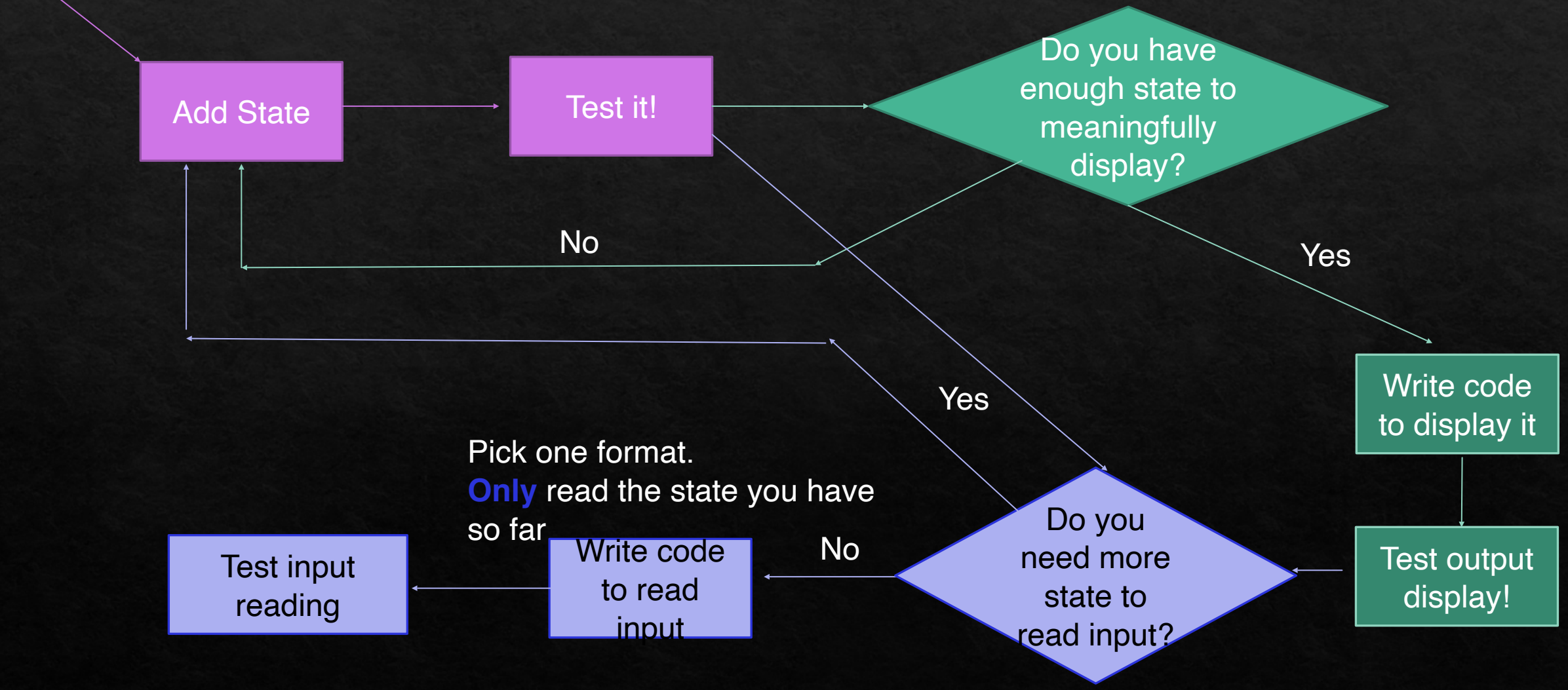
Pick a useful class
Write **part** of it
(or add more fields/methods)

Even if “so simple”
that testing is “silly”

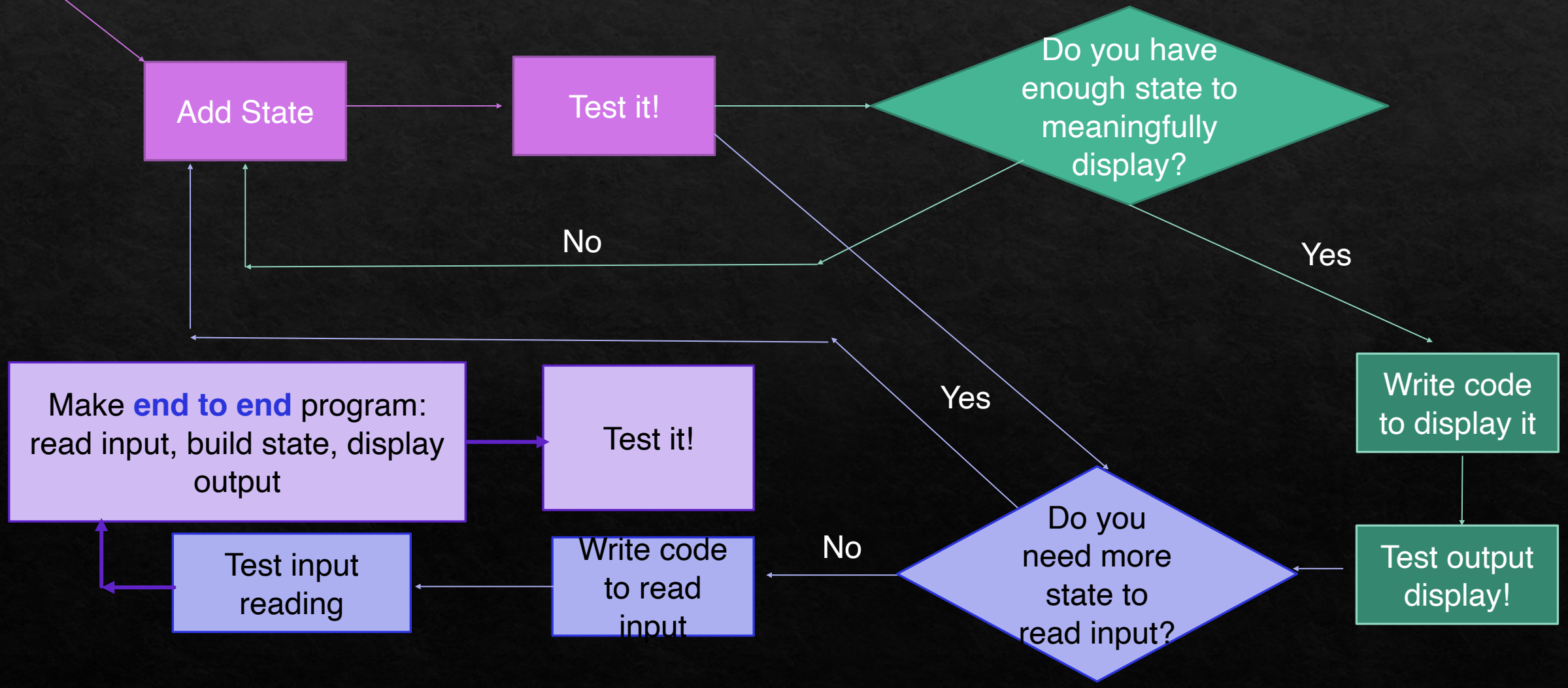
I Don’t Know Where to Start!



I Don’t Know Where to Start!

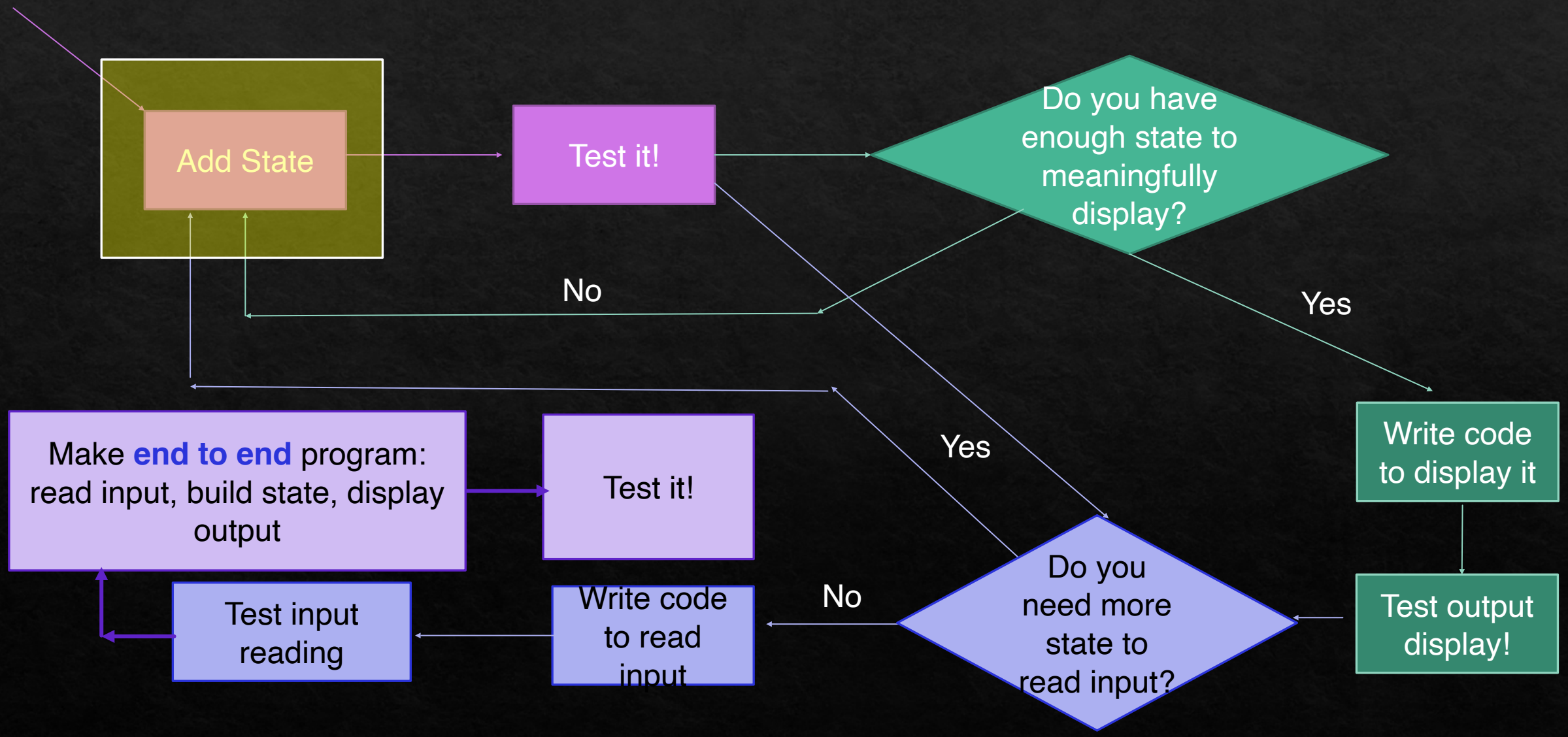


I Don’t Know Where to Start!



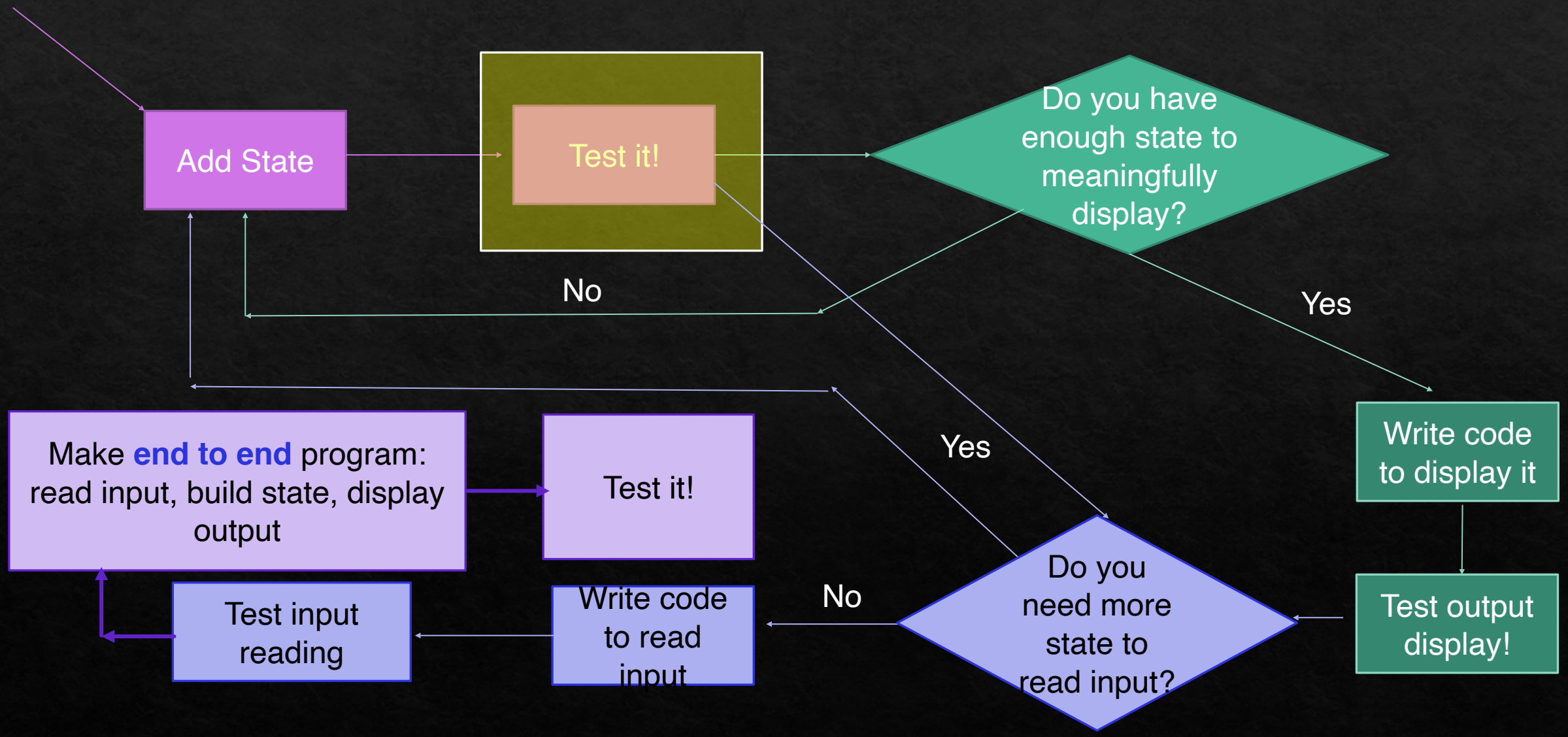
1. Board with only width/height

Battleship!



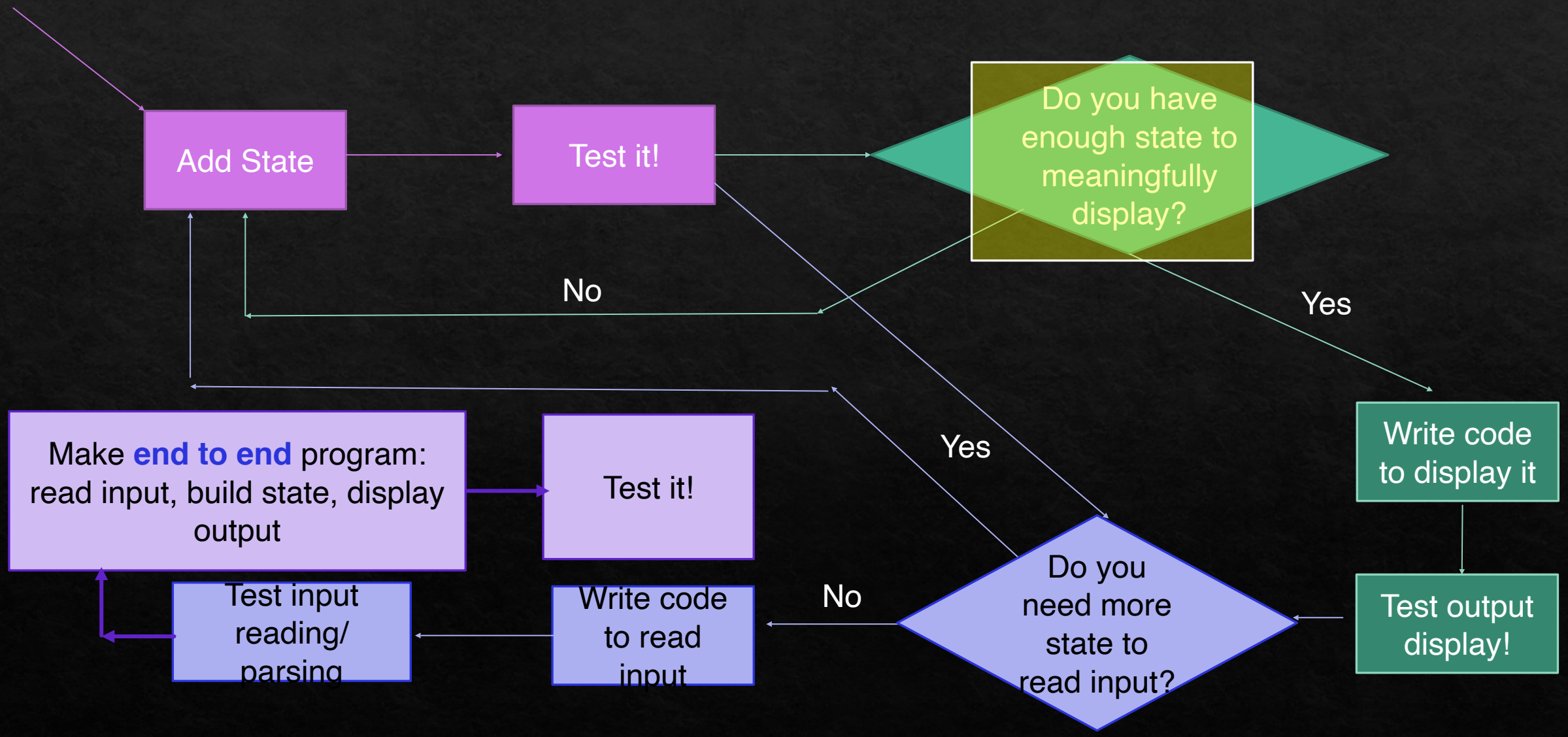
Battleship!

1. Board with only width/height
2. And tests!



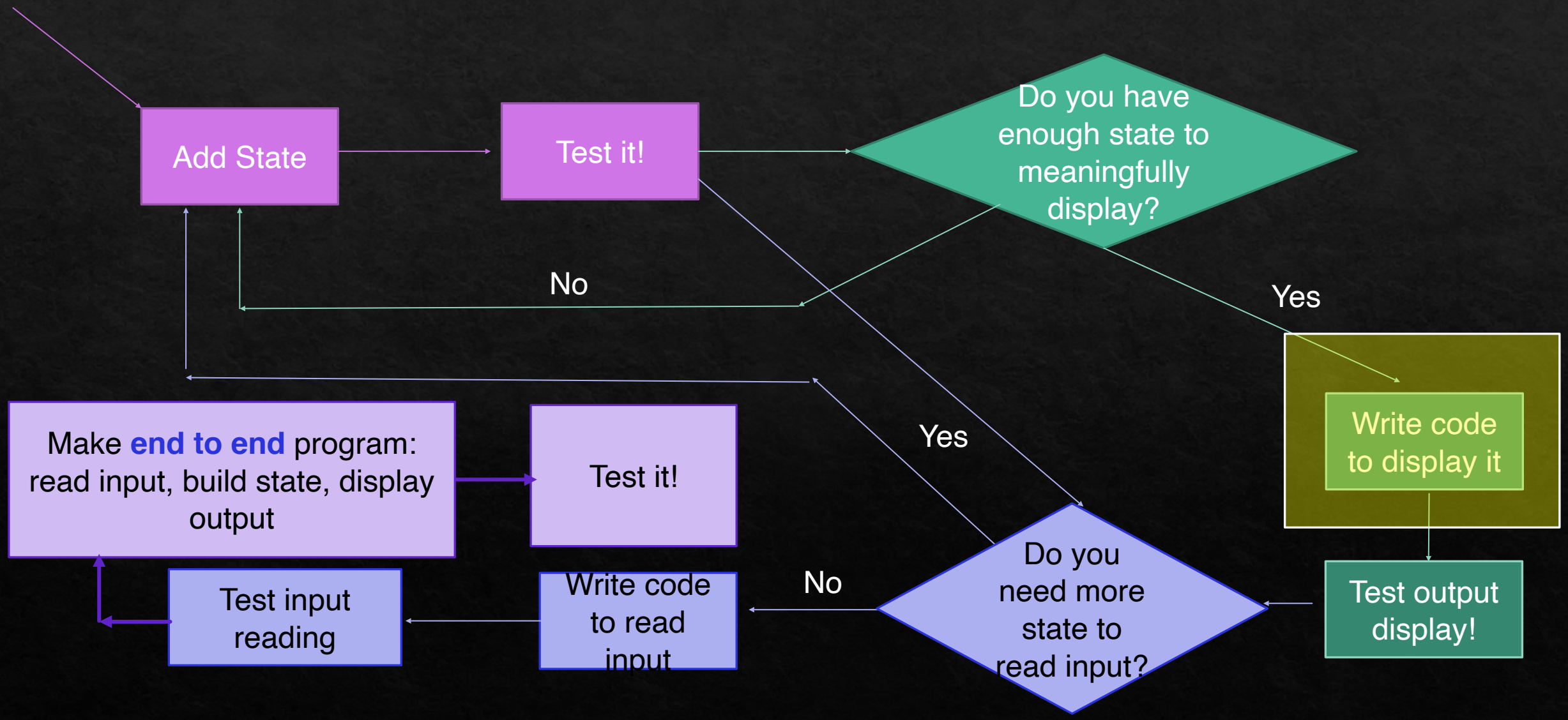
Battleship!

1. Board with only width/height
2. And tests!



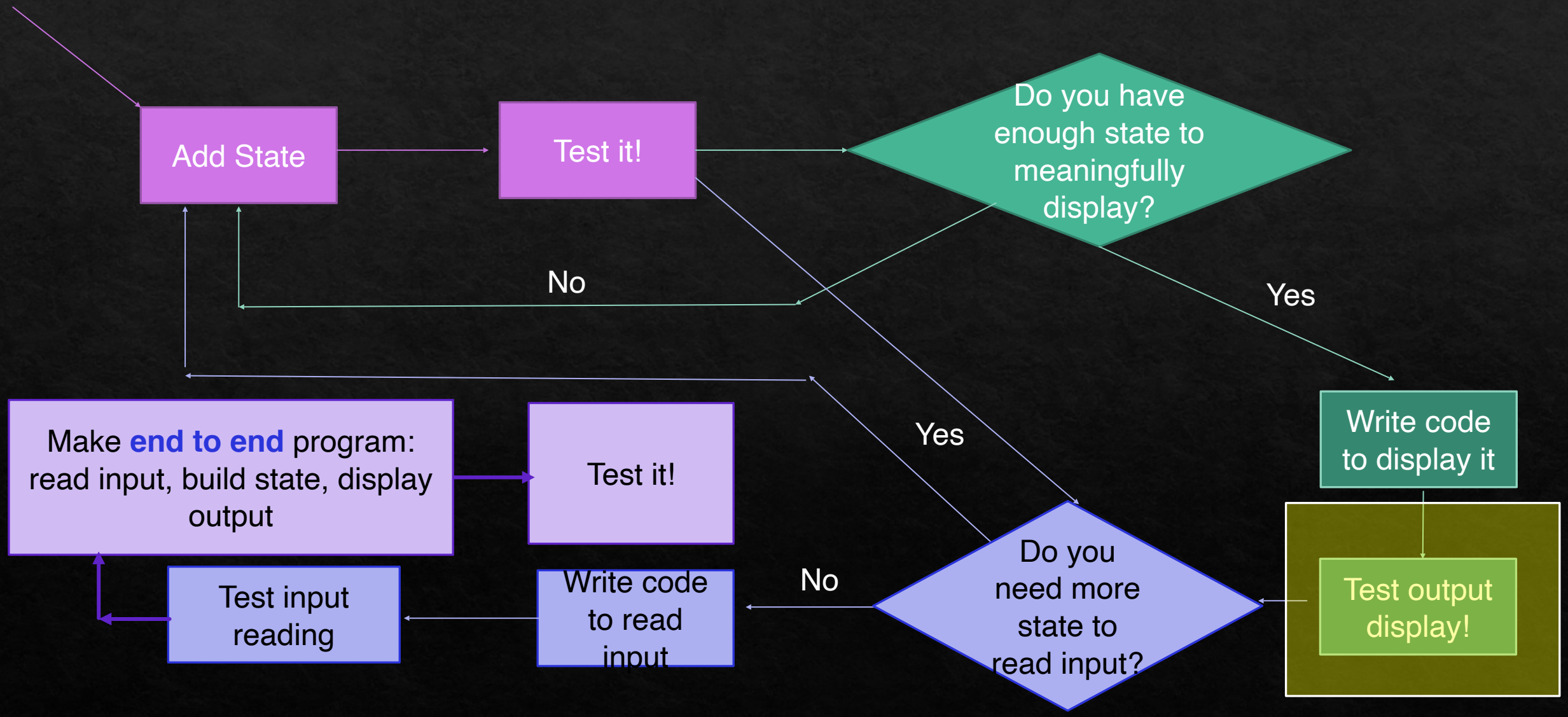
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView



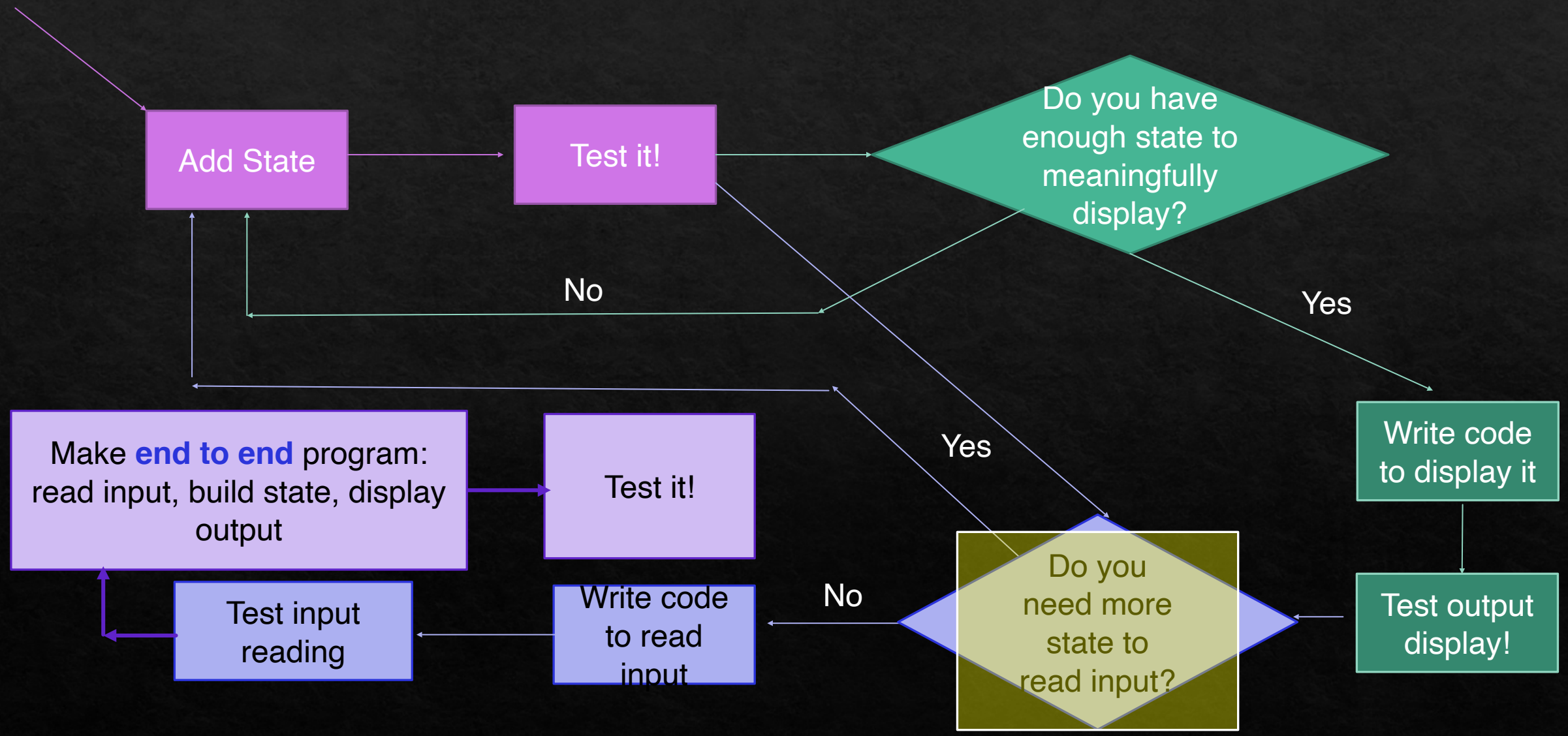
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)



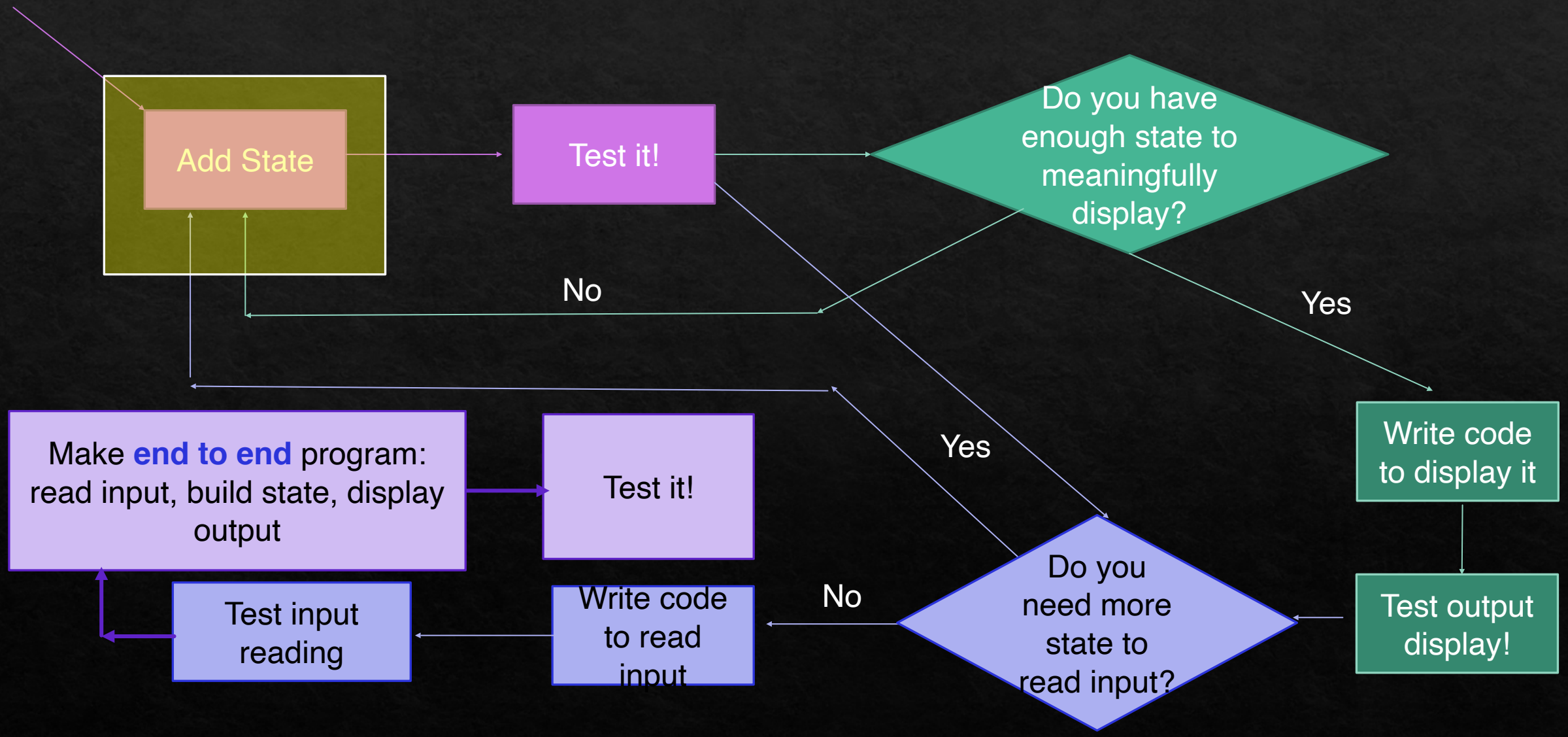
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)



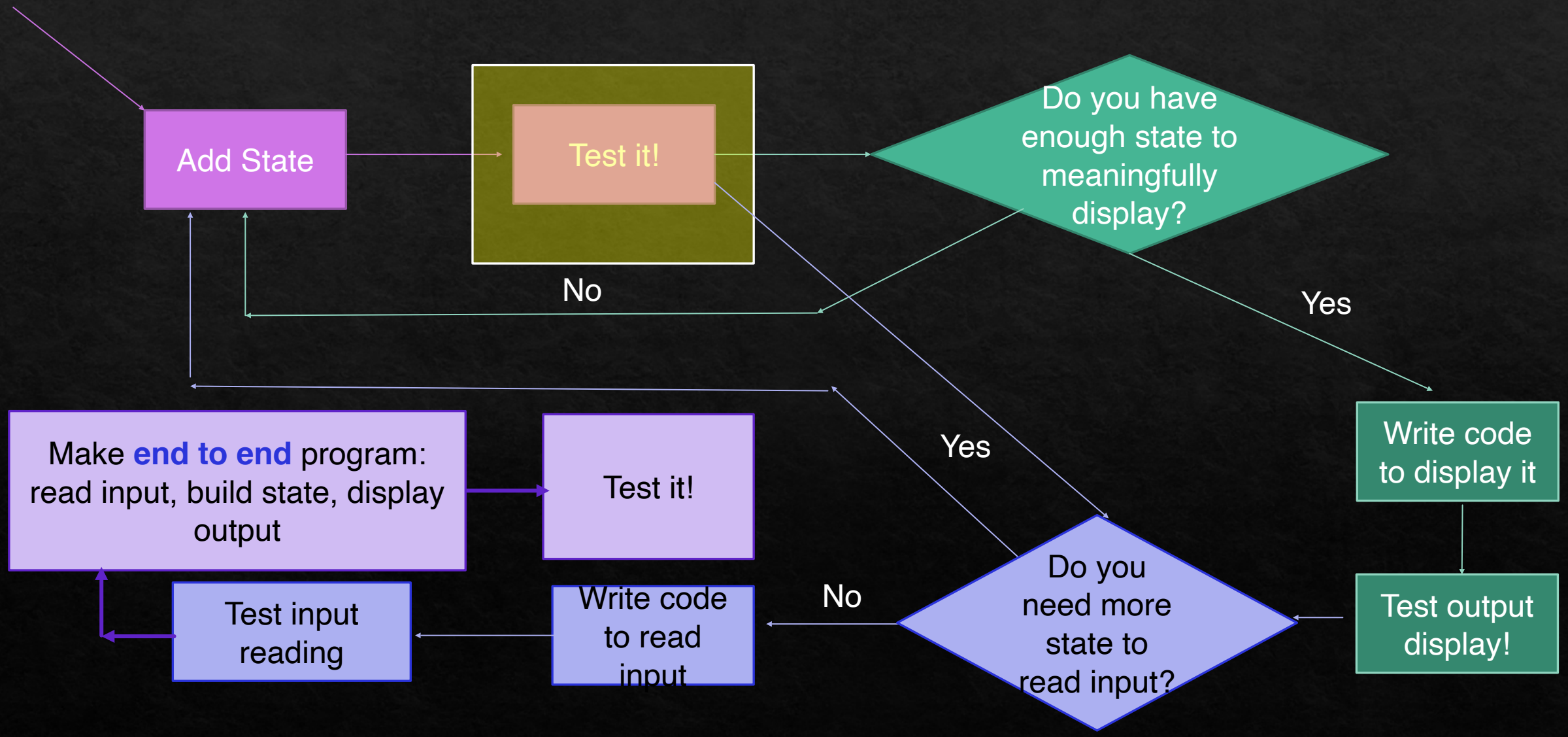
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate



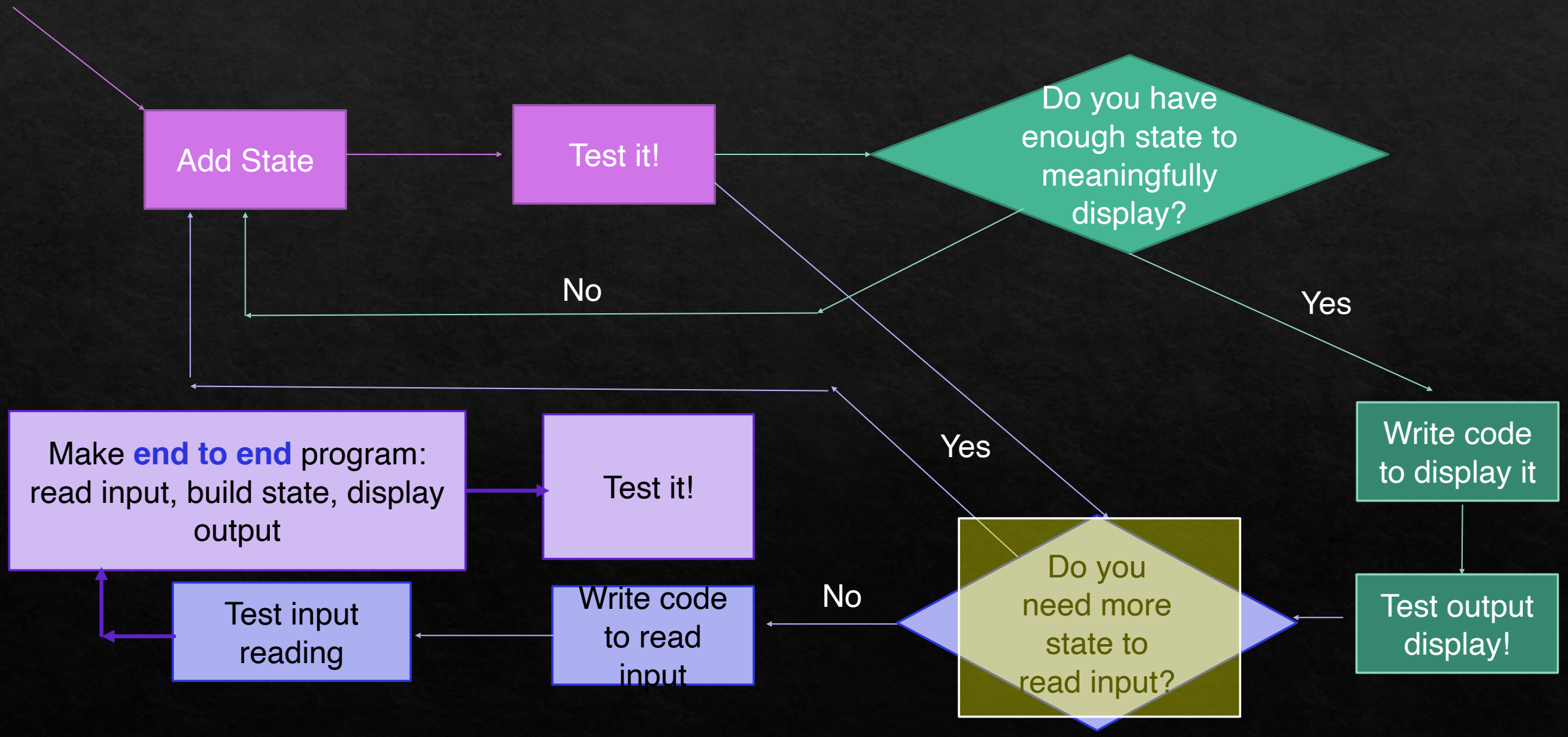
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)



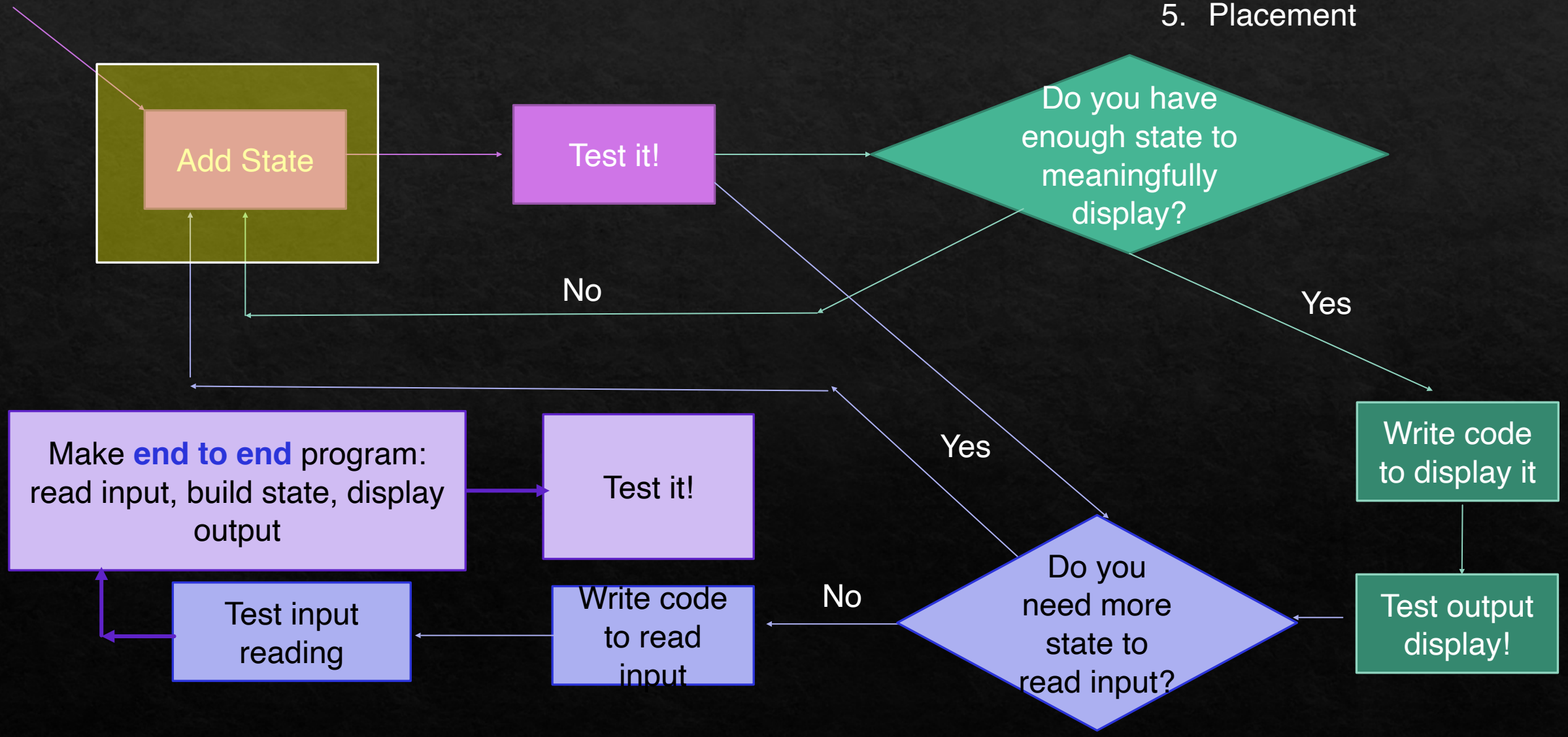
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)



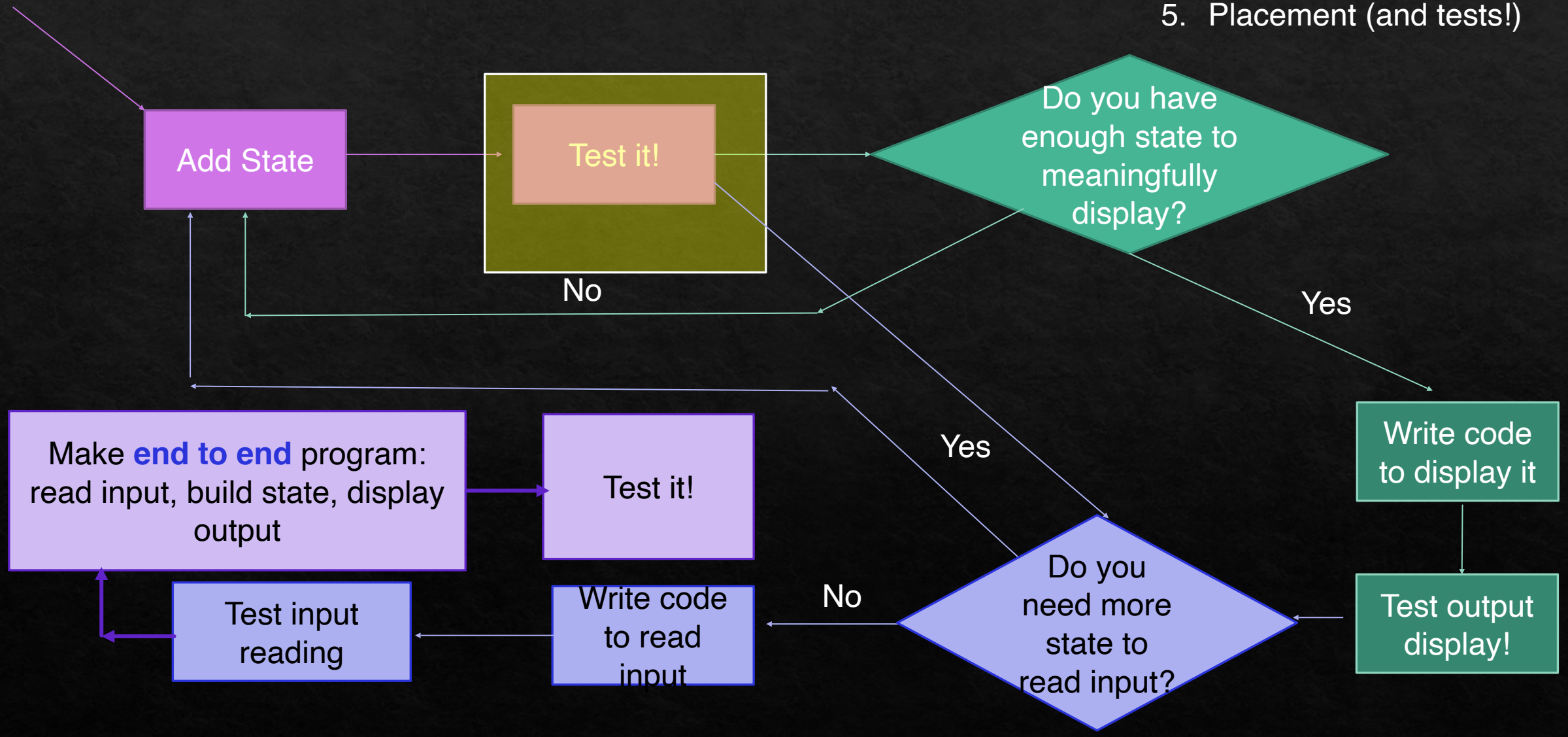
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement



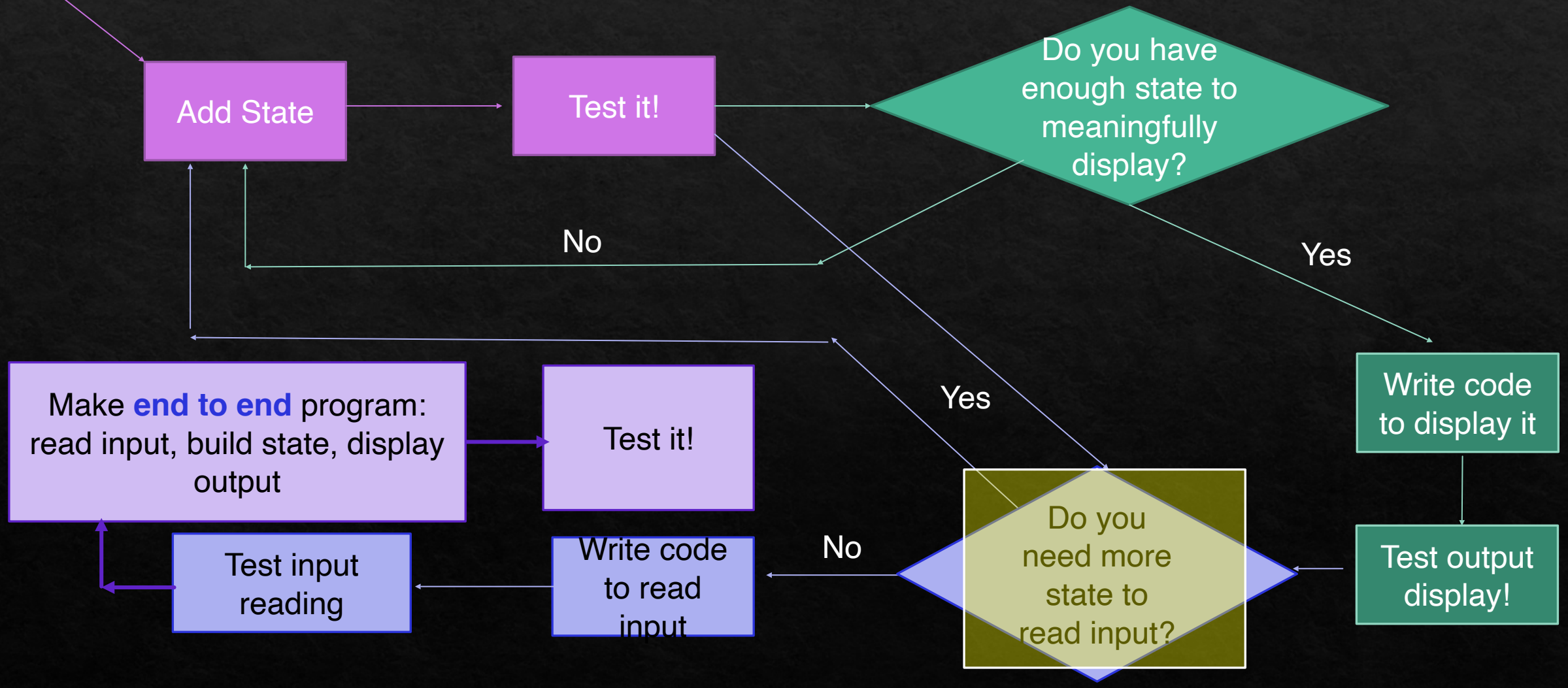
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)



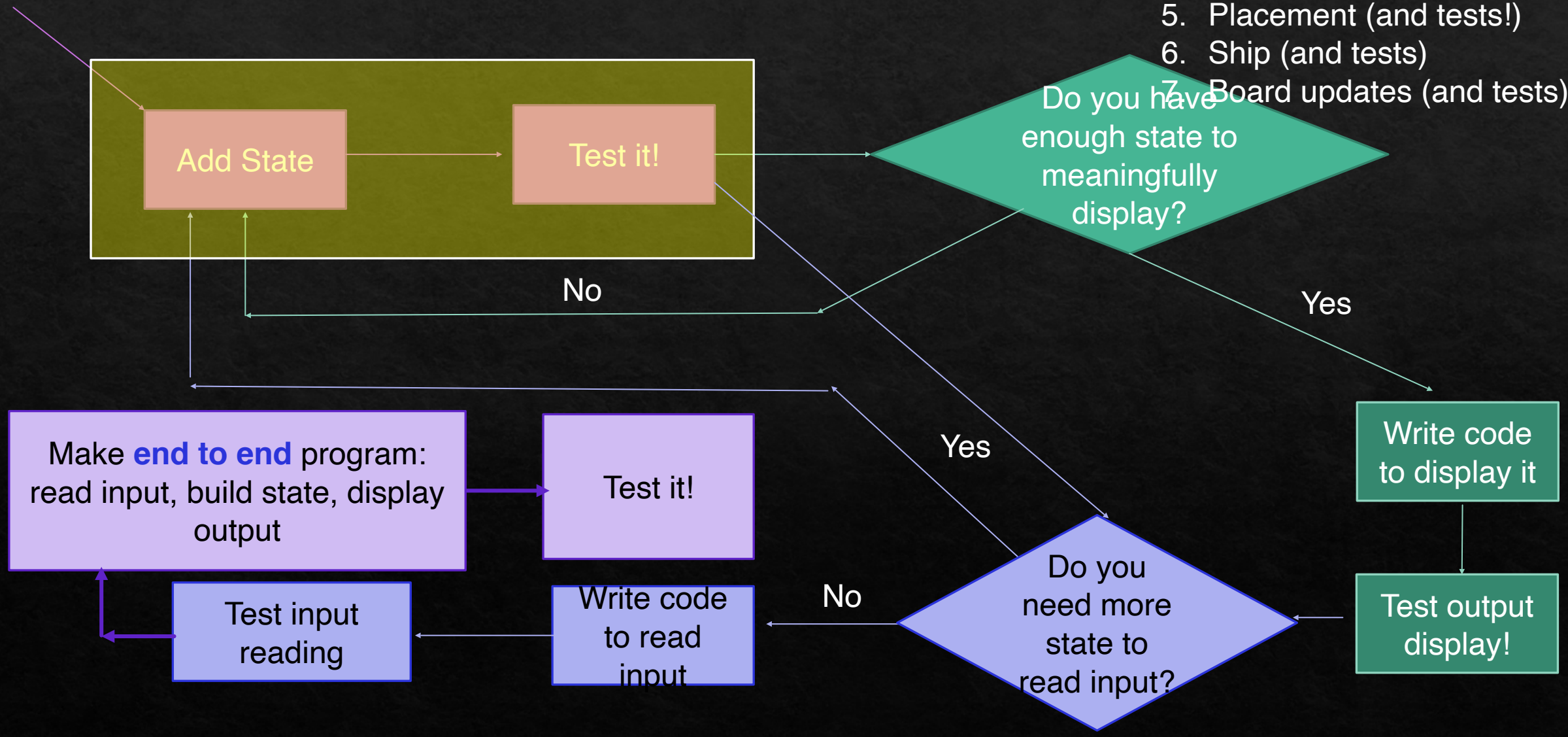
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)



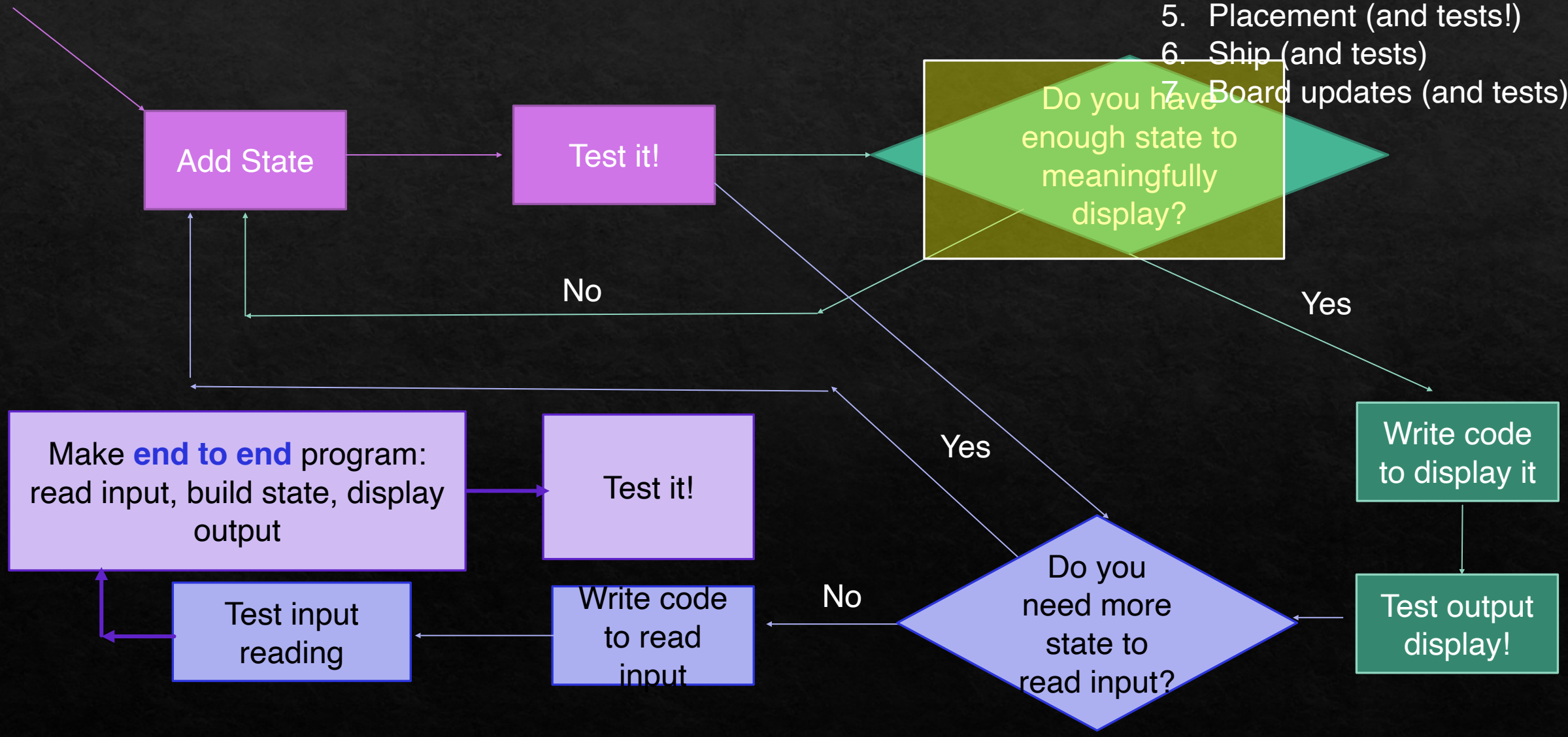
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tests)



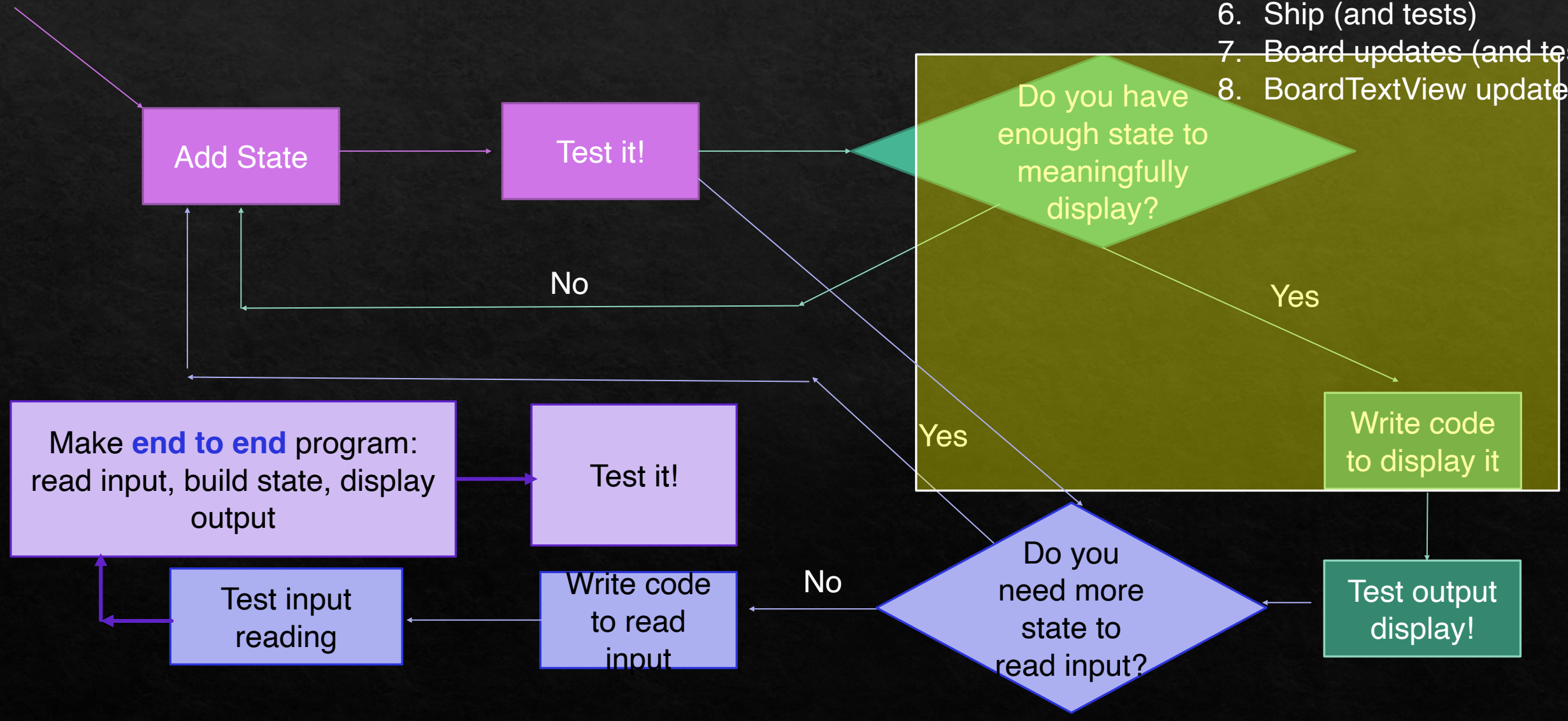
Battleship!

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tests)



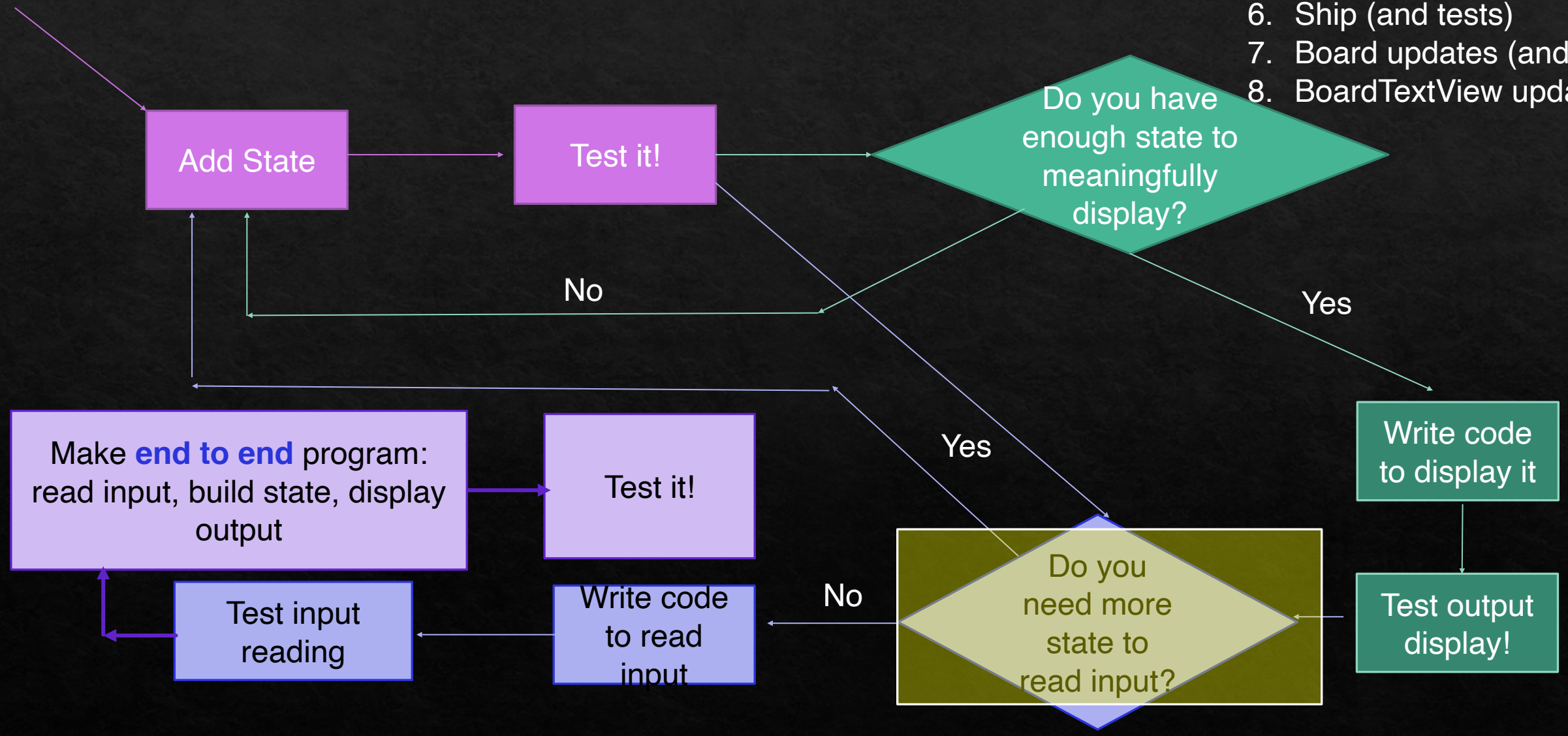
Battleship!

1. Board with only width/h
2. And tests!
3. BoardTextView (and tes
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. ~~Board updates (and tes~~
8. BoardTextView updates



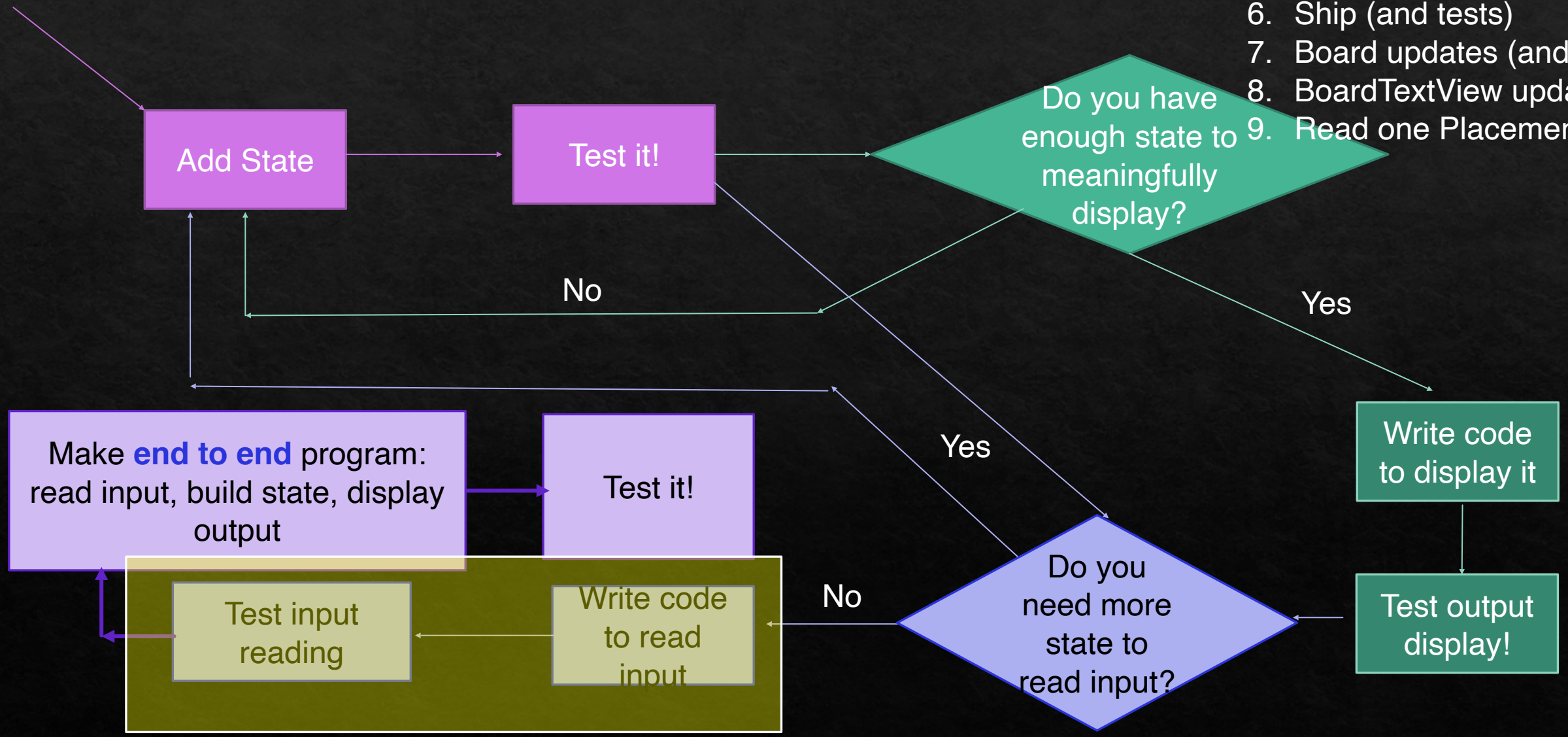
Battleship!

1. Board with only width/h
2. And tests!
3. BoardTextView (and tes
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tes
8. BoardTextView updates



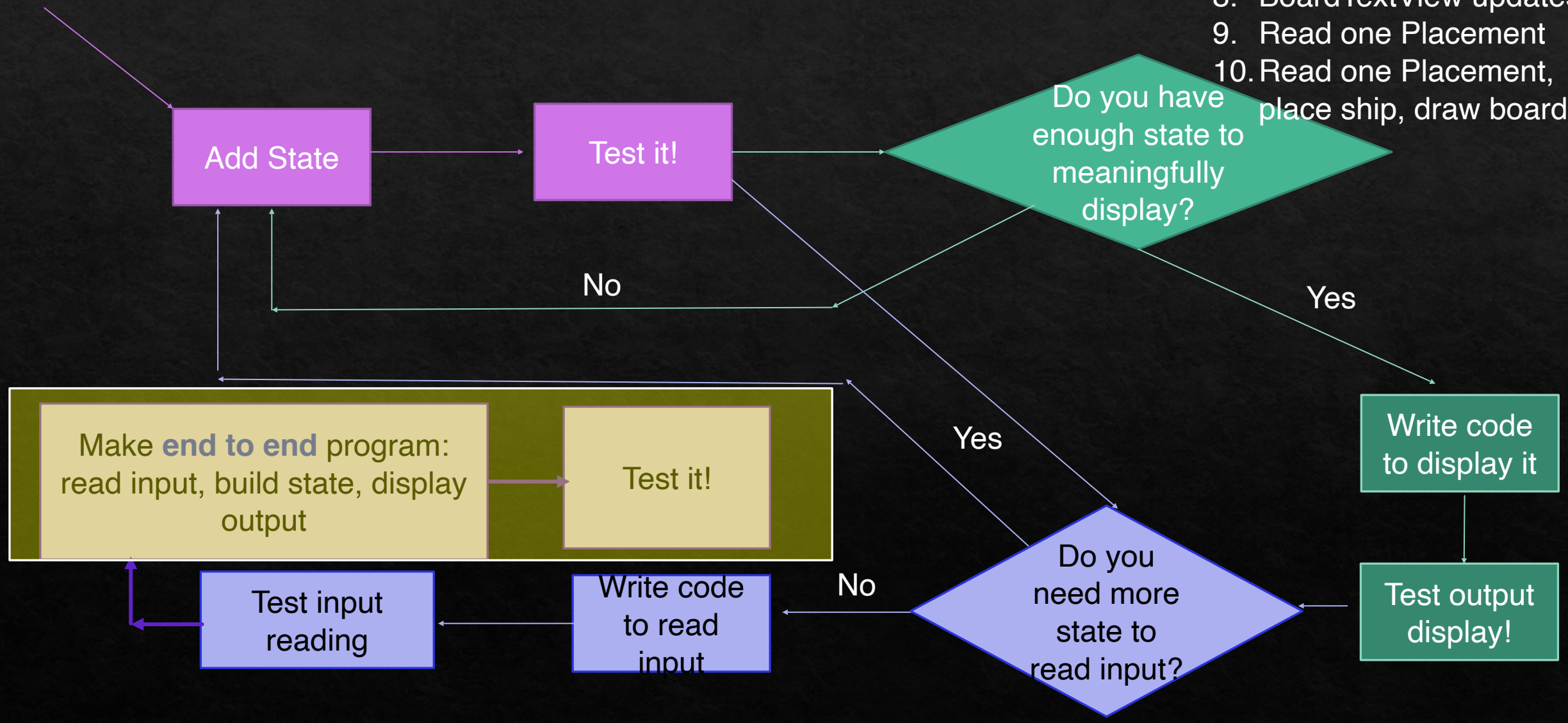
Battleship!

1. Board with only width/h
2. And tests!
3. BoardTextView (and tes
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tes
8. BoardTextView updates
9. Read one Placement



Battleship!

4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tests)
8. BoardTextView updates
9. Read one Placement
10. Read one Placement, place ship, draw board



Battleship!

Minimal end to end system:
"Looks like" battleship.
Everything else is just a feature to add to this.
(Maybe a big or even huge feature).

1. Board with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship updates (and tests)
7. Board updates (and tests)
8. BoardTextView updates
9. Read one Placement
10. Read one Placement, place ship, draw board

Make **end to end** program: read input, build state, display output → Test it!

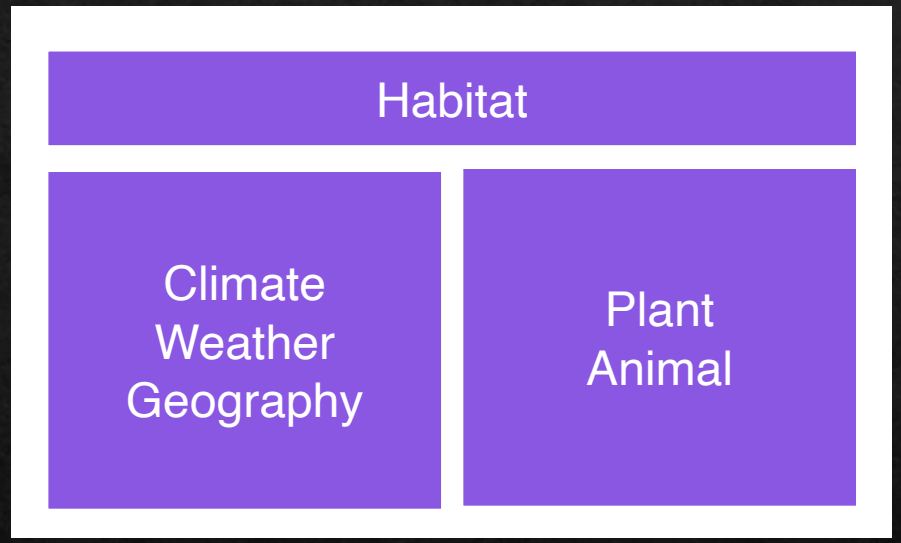
Planning First: Nouns and Their Verbs

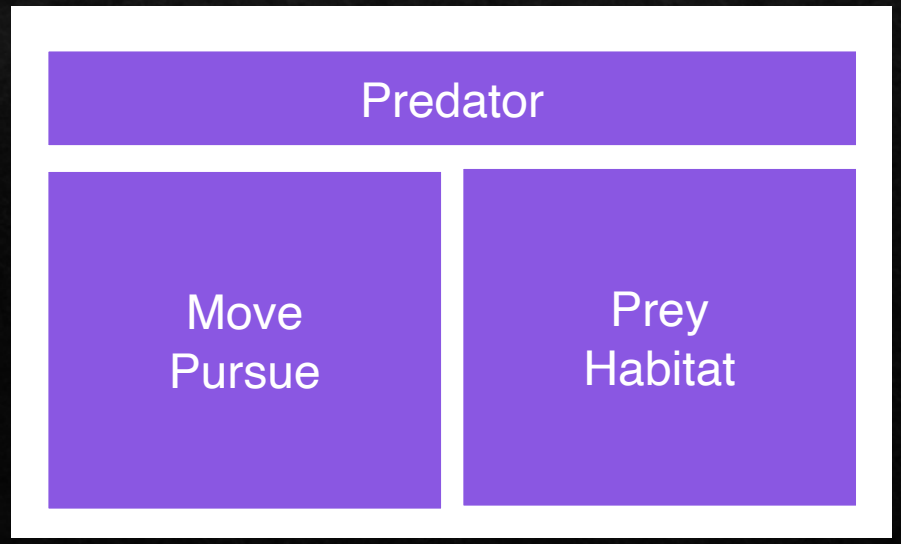
Before we start writing, we might do some planning
What nouns (classes) do we have?
What verbs (methods) do they have?
How are they related (classes have other classes?)

1. **Board** with only width/height
2. And tests!
3. BoardTextView (and tests!)
4. Coordinate (and tests!)
5. Placement (and tests!)
6. Ship (and tests)
7. Board updates (and tests)
8. BoardTextView updates
9. Read one Placement
10. Read one Placement, place ship, draw board

Class Responsibilities Collaborators (CRC): Cards

- 3x5 note card
- Class name on the top
- Left side: description of what class can do
- Right side: classes that this one interacts with





Project Management: Estimating Delivery Time

- Estimating how long a project will take is difficult
- Software estimations tend to be particularly hard
  - How long will it take you to debug that?
  - How many bugs are left?
- Good estimates are important
  - Why?

Nicholas Kraft
Software Engineer
UserVoice

Anna Ruelas
Software Engineer
xBK

Naija Farve
Software Engineer
Spreedly

How Late are we?

Estimated time to complete Tasks 1- 36

Nicholas		Anna		Naija	
Task #	Hours Est.	Task #	Hours Est.	Task #	Hours Est.
1	5	13	12	25	8
2	3	14	4	26	8
3	2	15	6	27	4
4	7	16	1	28	3
5	11	17	2	29	2
6	2	18	3	30	8
7	4	19	8	31	7
8	7	20	6	32	3
9	8	21	3	33	9
10	9	22	3	34	2
11	1	23	8	35	3
12	1	24	4	36	3

How soon can you expect to deliver?

Can you improve?

How Late are we?

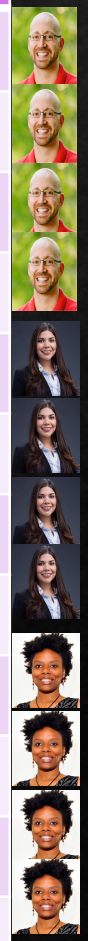
Nicholas		Anna		Naija	
Task #	Hours Est.	Task #	Hours Est.	Task #	Hours Est.
1	5	13	12	25	8
2	3	14	4	26	8
3	2	15	6	27	4
4	7	16	1	28	3
5	11	17	2	29	2
6	2	18	3	30	8
7	4	19	8	31	7
8	7	20	6	32	3
9	8	21	3	33	9
10	9	22	3	34	2
11	1	23	8	35	3
12	1	24	4	36	3
Total	**51**		**58**		**60**

Option 1: Sum up hours that it will take each developer to finish.

Should we move work from Naija to Nicholas?

Observe Previous Task Completion

Task	Estimate	Actual	Developer
Add parrot	7	11.2	Nicholas
Remove alley cat	8	9.7	Nicholas
Setup CI/CD	3	4.5	Nicholas
Bird Search	5	4.2	Nicholas
Login (web)	5	7.6	Anna
Login (mobile)	2	3.2	Anna
Cat matching	7	10.2	Anna
Training	6	9.1	Anna
Initial Test Infrastruc.	4	3.5	Naija
Dog clustering	10	9.5	Naija
Advanced Testing	11	9.7	Naija
Bird Regression	7	6.8	Naija



Observe Previous Task Completion

Task	Estimate	Actual	Developer	Actual/Est
Add parrot	7	11.2	Nicholas	1.6
Remove alley cat	8	9.7	Nicholas	1.21
Setup CI/CD	3	4.5	Nicholas	1.14
Bird Search	5	4.2	Nicholas	.84
Login (web)	5	7.6	Anna	1.52
Login (mobile)	2	3.2	Anna	1.15
Cat matching	7	10.2	Anna	1.46
Training	6	9.1	Anna	1.52
Initial Test Infrastruc.	4	3.5	Naija	.88
Dog clustering	10	9.5	Naija	.95
Advanced Testing	11	9.7	Naija	.88
Bird Regression	7	6.8	Naija	.97

Mix of over and underestimating

Underestimating

Overestimating

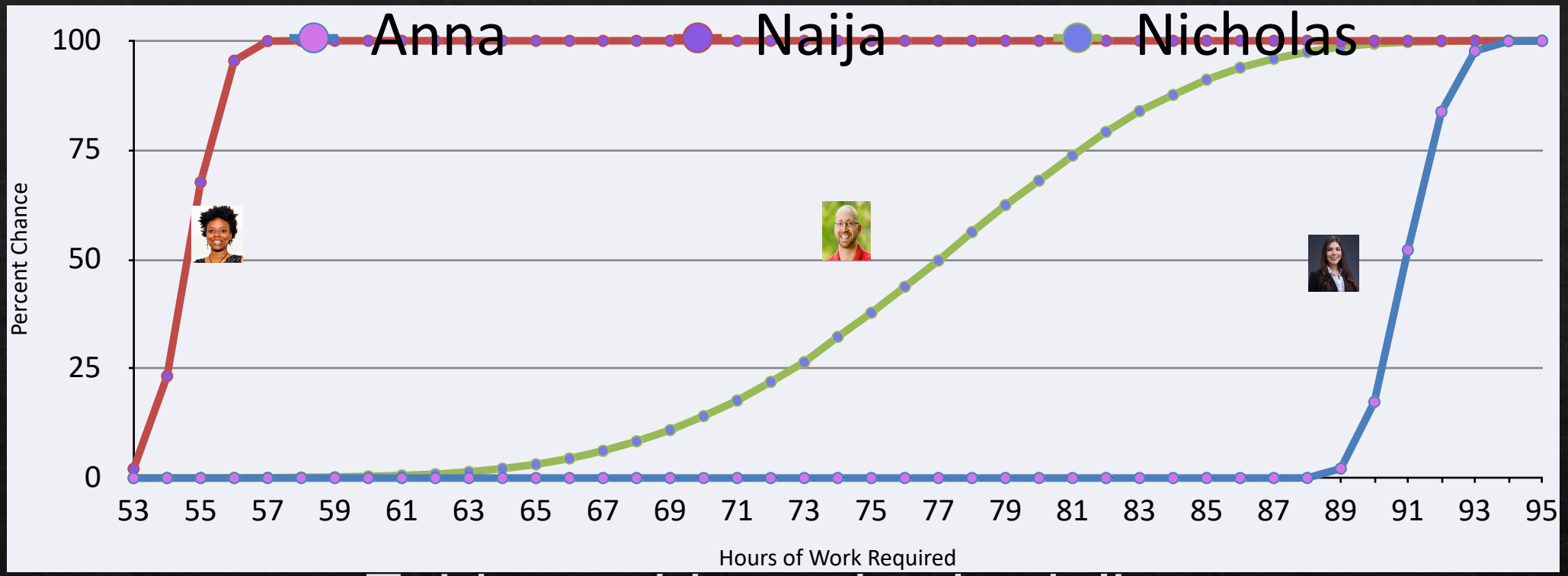
Previous Task Completion

Task	Estimate	Actual	Developer	Actual/Est
Add parrot	7	11.2	Nicholas	1.6
Remove alley cat	8	9.7	Nicholas	1.21
Setup CI/CD	3	4.5	Nicholas	1.14
Bird Search	5	4.2	Nicholas	.84
Login (web)	5	7.6	Anna	1.52
Login (mobile)	2	3.2	Anna	1.15
Cat matching	7	10.2	Anna	1.46
Training	6	9.1	Anna	1.52
Initial Test Infrastruc.	4	3.5	Naija	.88
Dog clustering	10	9.5	Naija	.95
Advanced Testing	11	9.7	Naija	.88
Bird Regression	7	6.8	Naija	.97

This suggests a distribution of Nicolas's estimate error:
25%: 1.6x
25%: 1.21x
25%: 1.5x
25%: 0.84x

Nicholas					
Task #	Hours Est.	Draw 1	Probable Completion (hrs)	Draw 2	Probable Completion (hrs)
1	5	1.5x	7.5	1.21x	6
2	3	.84x	2.5	1.6x	4.8
3	2	1.5x	3	.84x	1.7
4	7	1.6x	11.2	1.5x	10.5
5	11	1.5x	16.5	1.5x	10.5
6	2	1.6x	3.2	1.6x	3.2
7	4	.84x	3.4	.84x	3.4
8	7	1.21x	8.5	1.5x	10.5
9	8	1.21x	9.7	1.5x	12
10	9	1.5x	13.5	1.21x	10.9
11	1	1.6x	1.6	.84x	0.8
12	1	1.21x	1.2	1.21x	1.2
Total	**51**		**83.4**		**81.5**

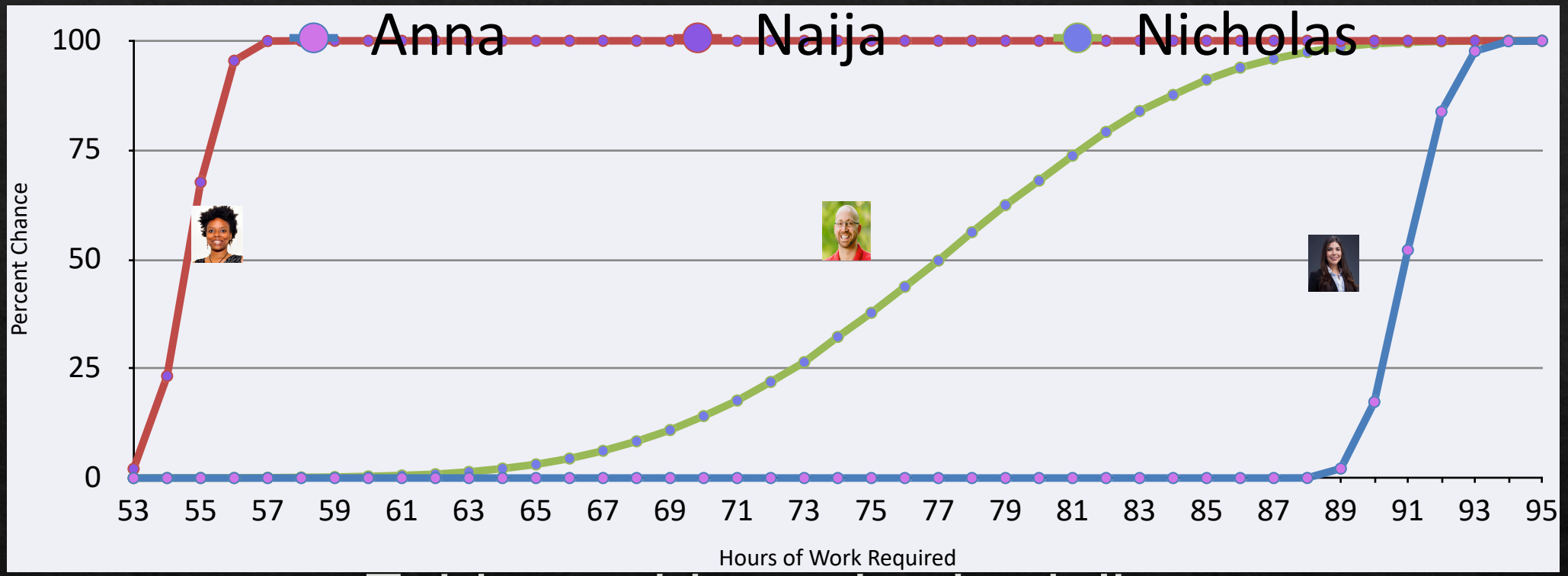
51 hours is too optimistic



Evidenced-based scheduling

- Do a few thousand random draws
- Build a cumulative distribution function of when we expect each developer to finish

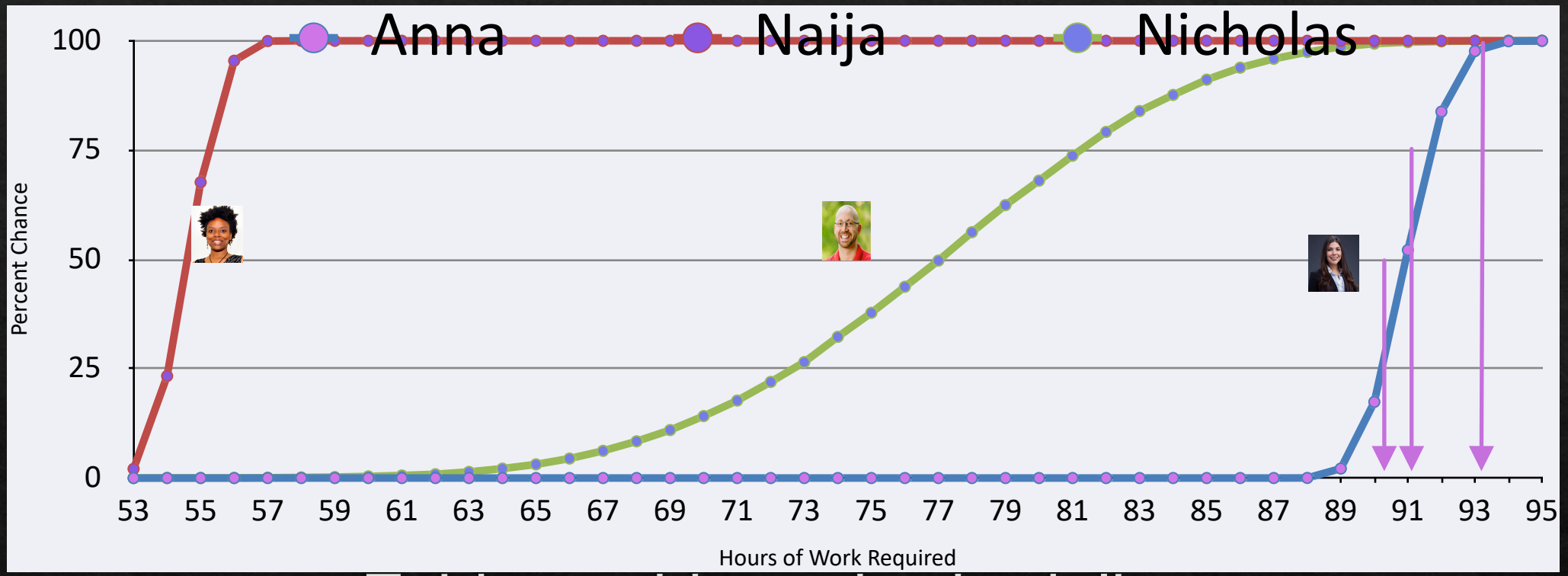
Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- How do you interpret this graph?
- When do you expect the team to finish?

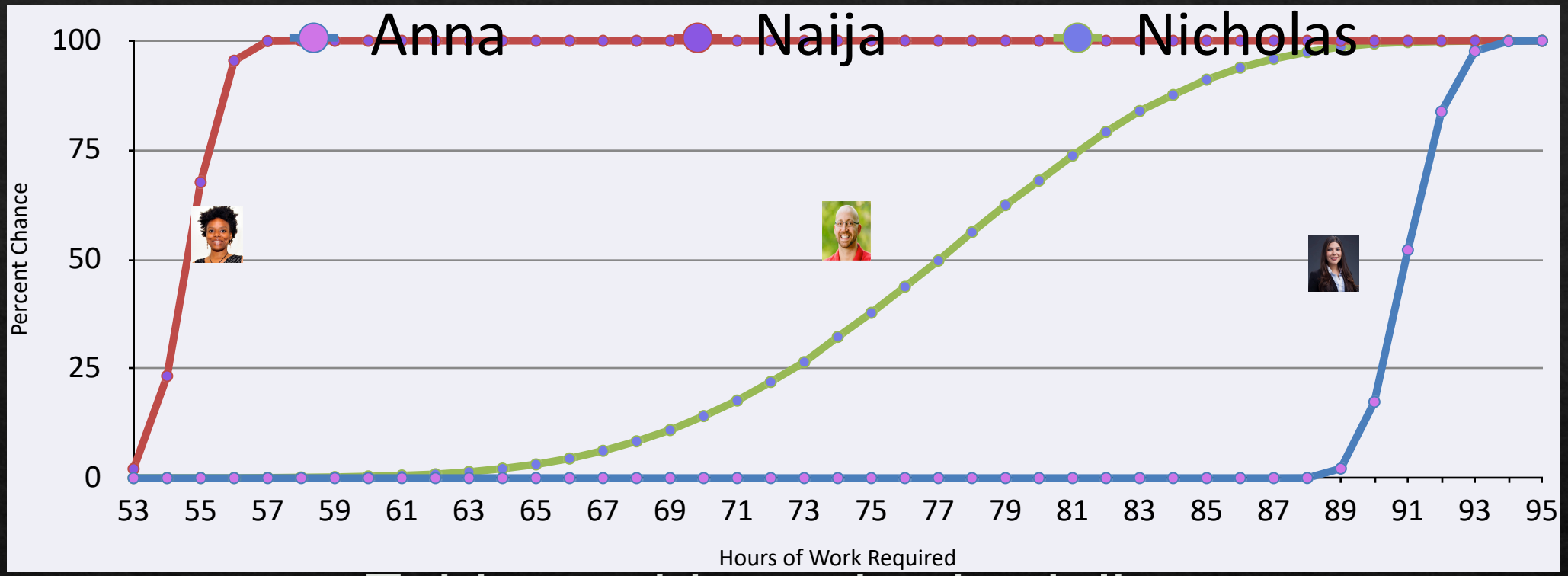
Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- Estimate depends on our confidence
  - 50% likely to be done after 91 hours of work
  - 75% likely to be done after 92 hours of work
  - 99% likely to be done after 94 hours of work

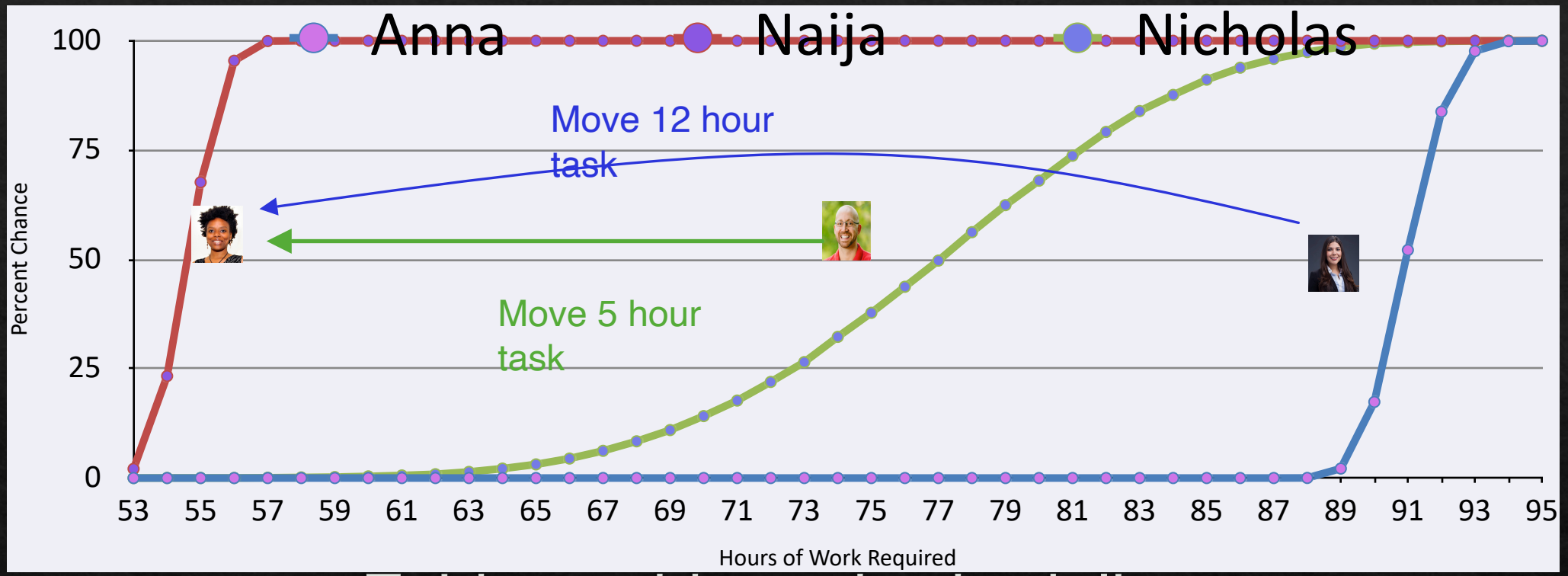
Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- What if anything can we do about this?

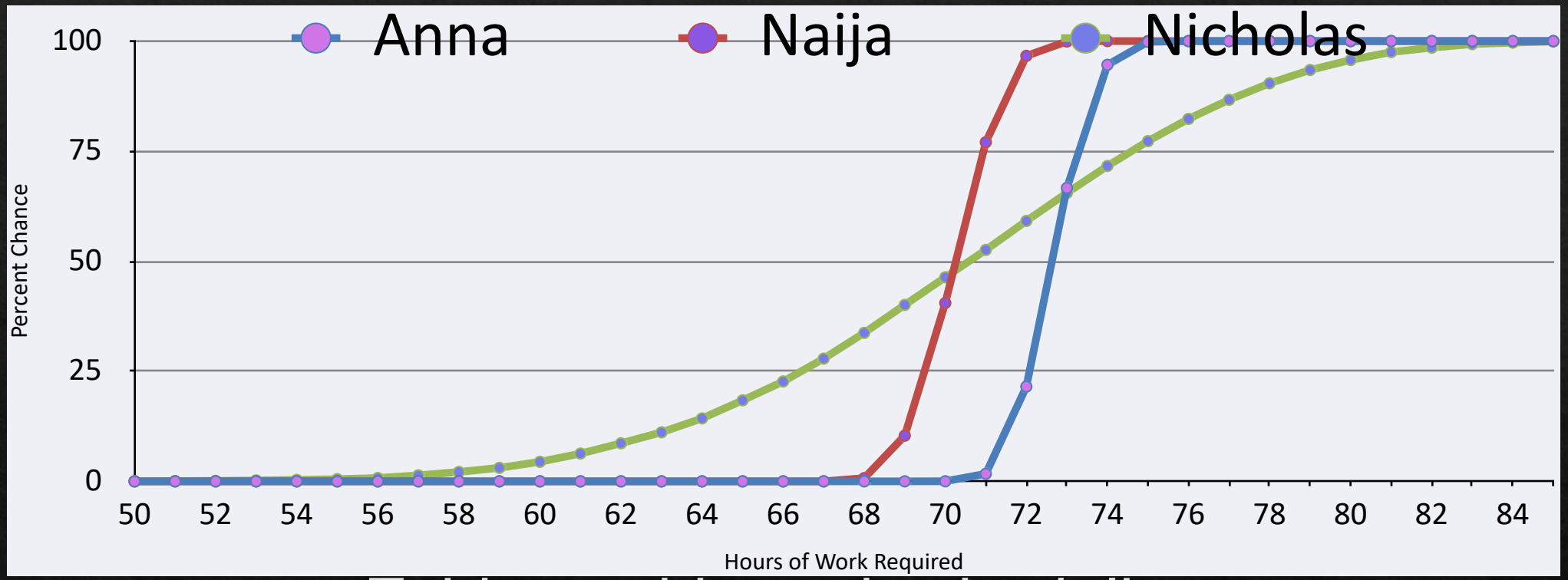
Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- What if anything can we do about this?
  - Shift tasks from Anna and Nicholas to Naija

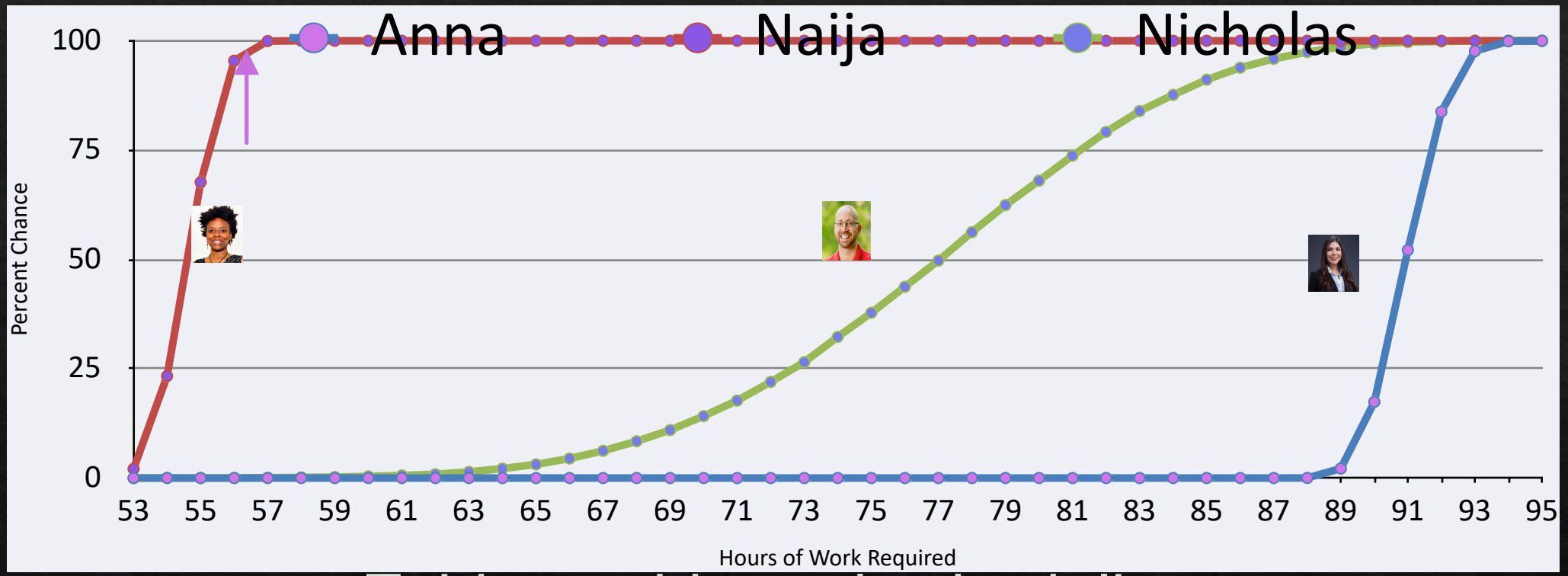
Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- Result: now 99% likely to be done by 74 hours!

Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/



Evidenced-based scheduling

- Why plan ahead?
  - Why not let everyone work until someone is done, then move tasks to them?

Source: https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/

Think, Pair, Share

- Why not just assign Naija tasks when she is done?
- Then move tasks to Nicholas when he is done?

Why Plan/Adjust Early?

May need to cut features

If Naija has low priority features may need to drop them

Schedules slips are not as bad if given early notice

2-month extension, 1 week prior to release

2-month extension, 8 months prior to release

Tasks take different people varying amounts of time

5 hours for Anna on the kernel module (has been working on related tasks)

20 hours for Naija on the kernel module (has been working in other areas of project