

Lecture 5b

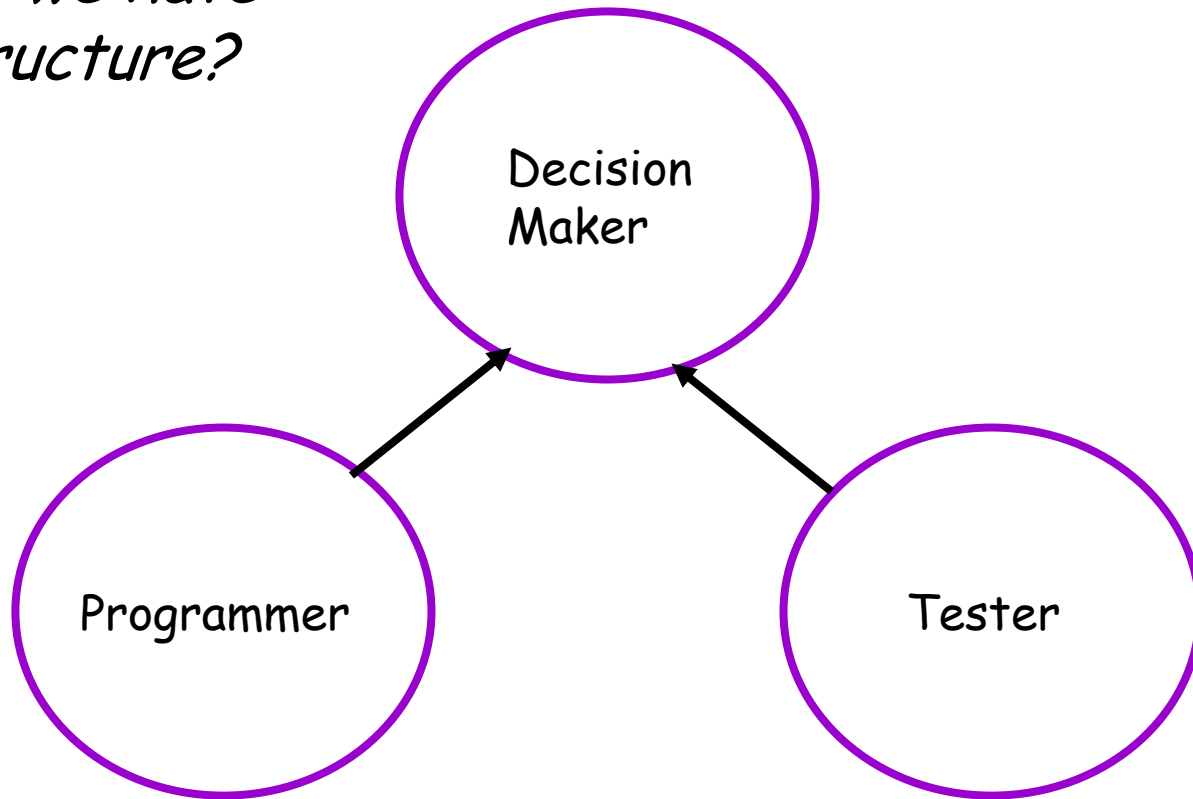
Introduction to Testing

State of the World

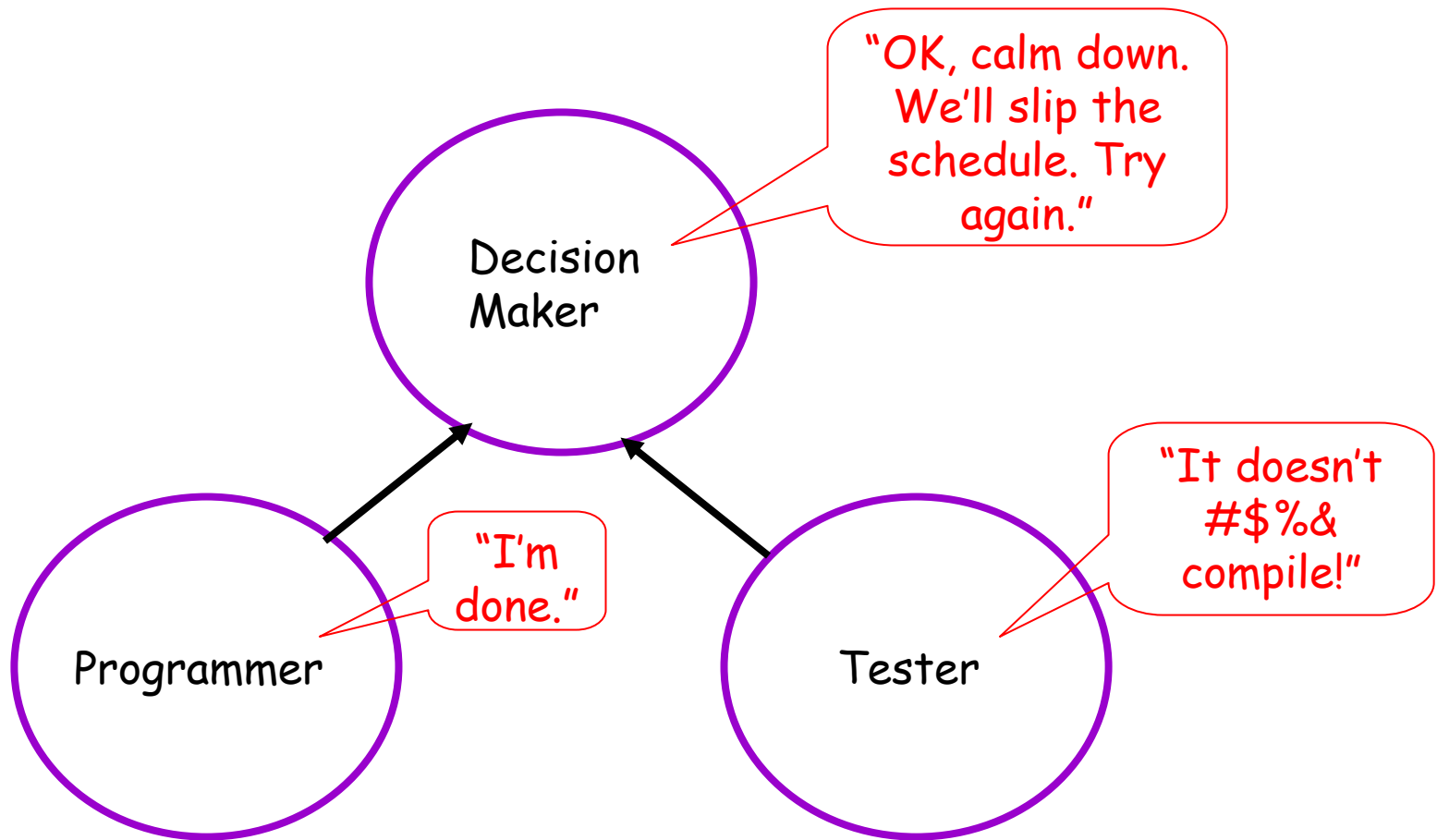
- Standard software development is simple
 - No rocket science here
- Outline
 - Someone writes a program
 - Someone runs the program and checks that it behaves as expected
 - Someone decides when it is OK to release

Software Development Today

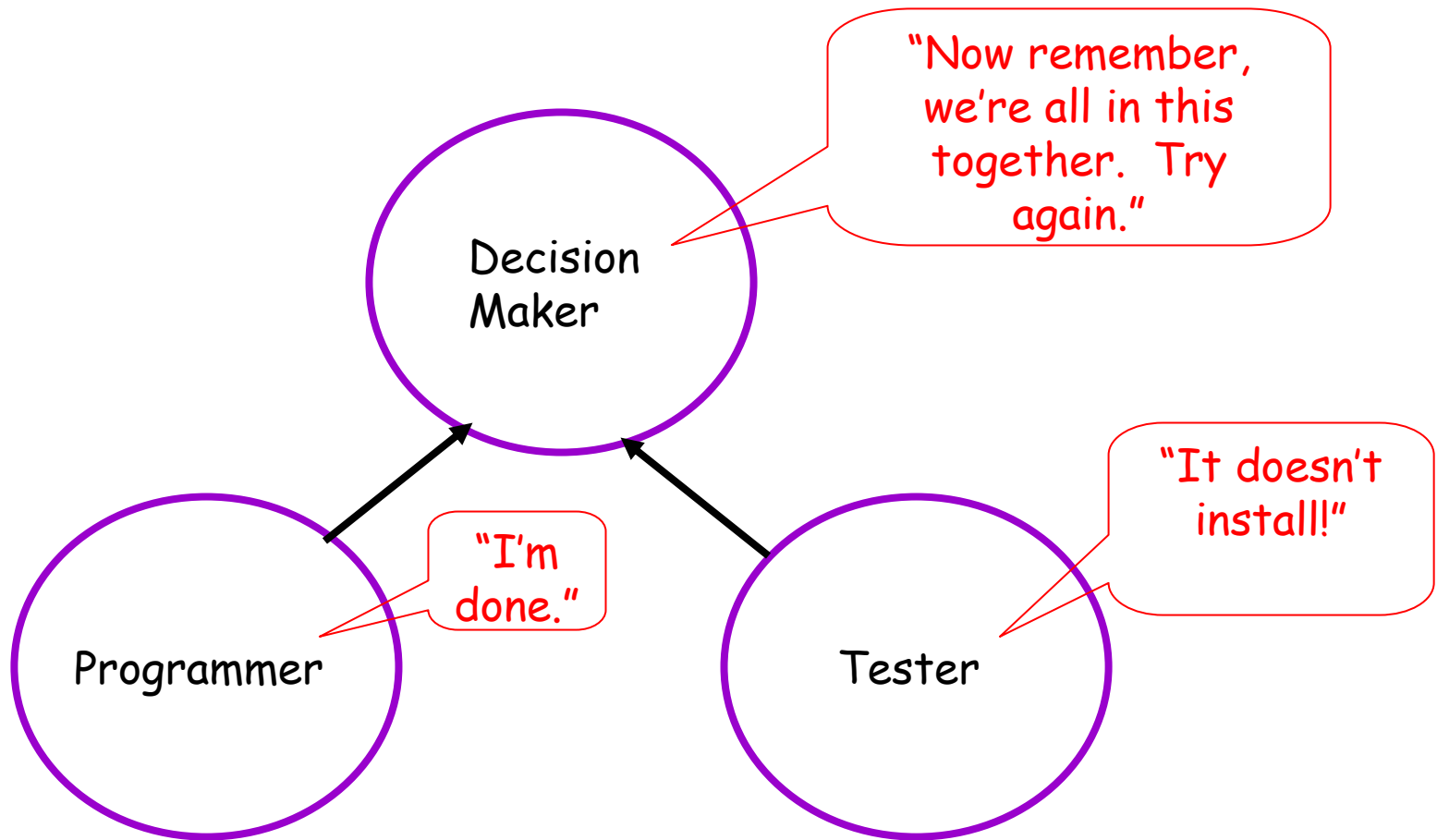
*Why do we have
this structure?*



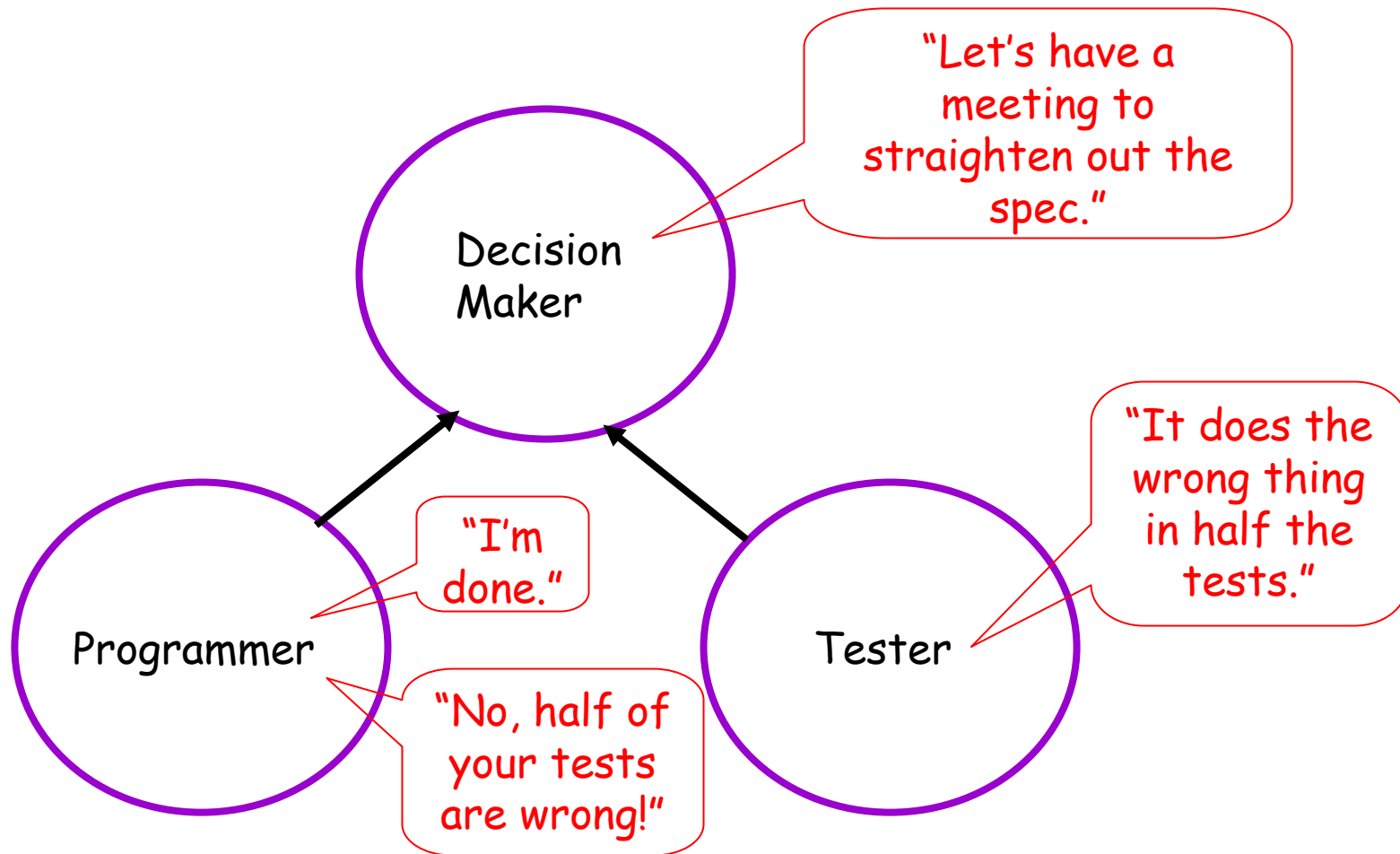
Typical Scenario (1)



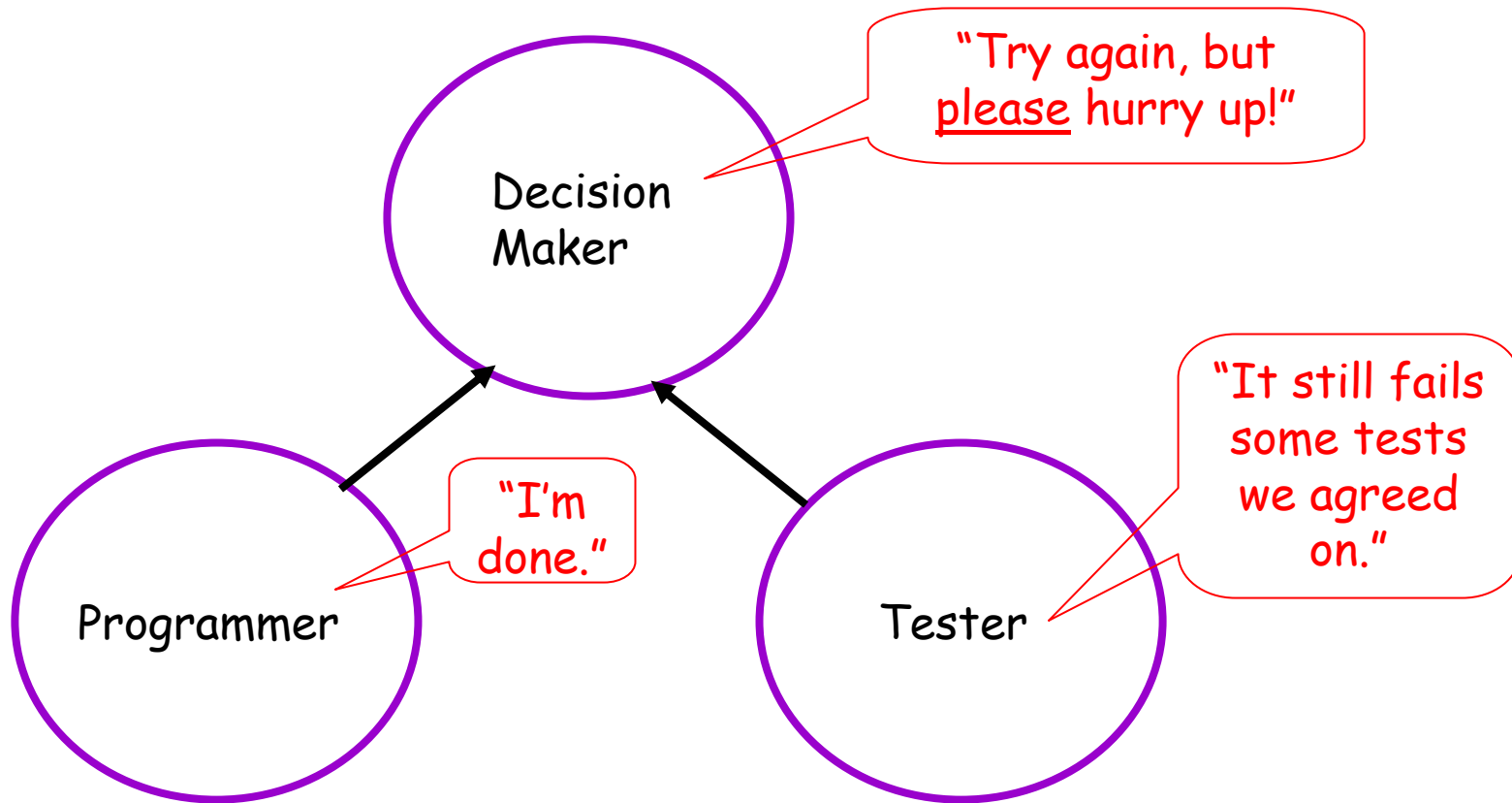
Typical Scenario (2)



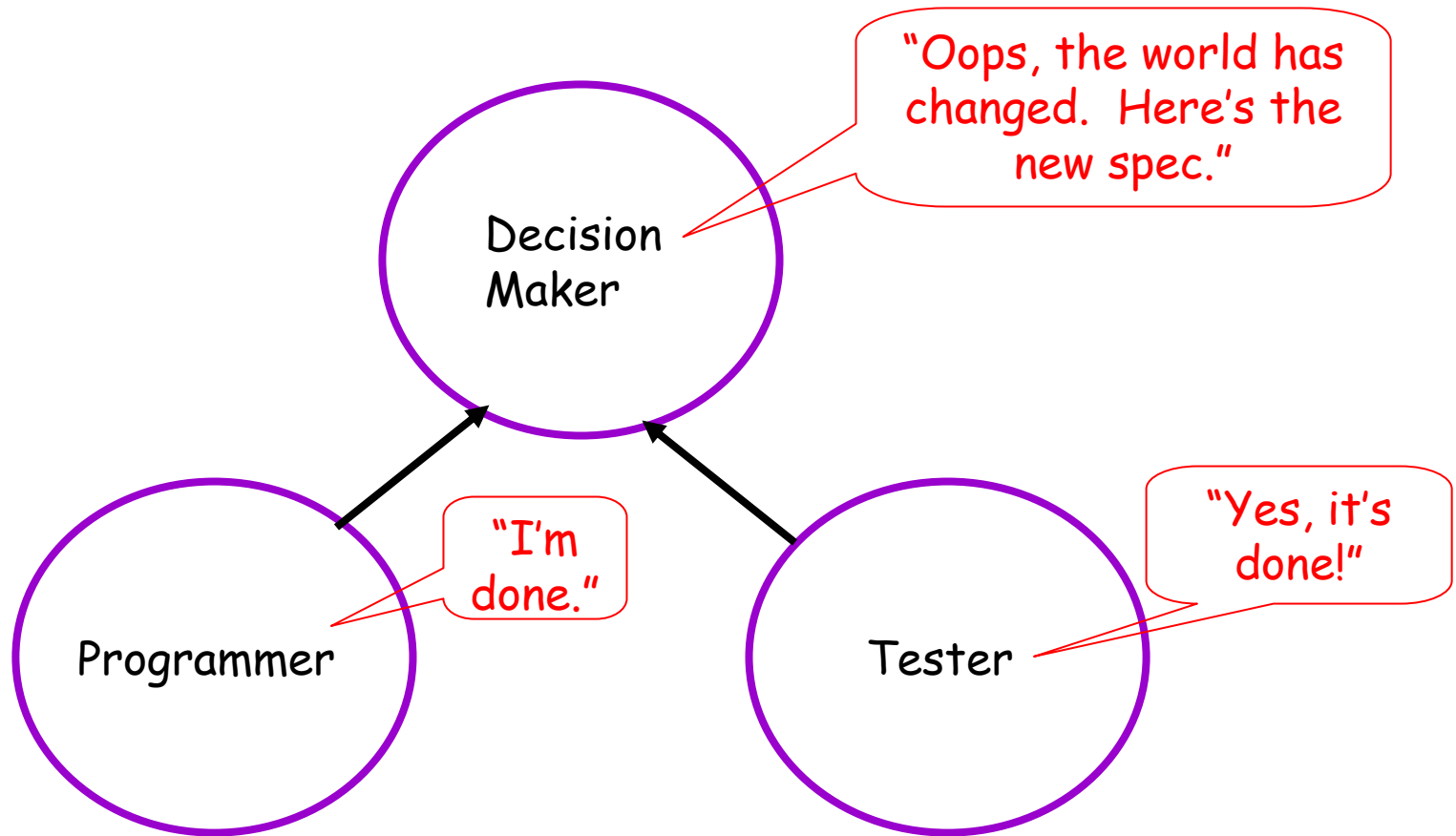
Typical Scenario (3)



Typical Scenario (4)

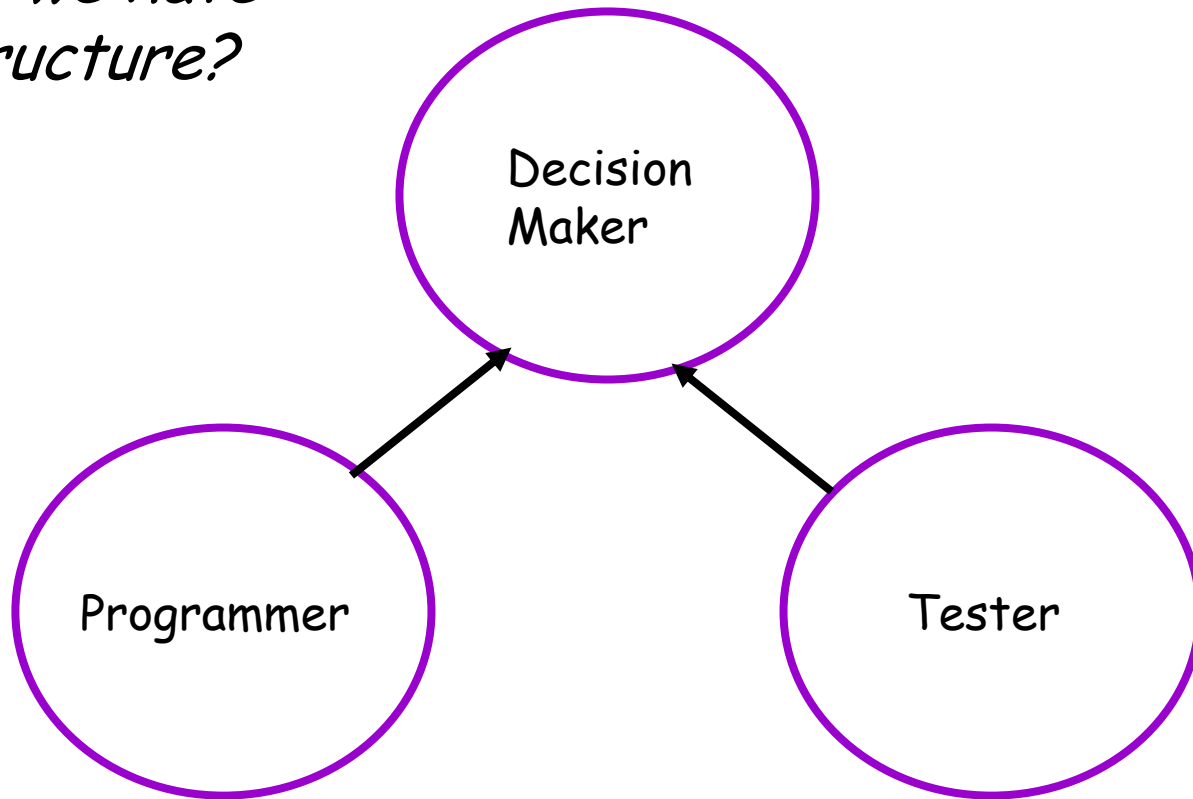


Typical Scenario (5)



Software Development Today

*Why do we have
this structure?*



Key Assumptions

- Development and testing must be independent
- Specifications must be explicit
- Specifications are always evolving
- All resources (including time) are finite
- Human organizations need decision makers

- Examine each of these separately

Independent Testing and Development

- Testing is basic to every engineering discipline
 - Design a drug
 - Manufacture an airplane
 - Etc.

- Why?
 - Because our ability to predict how our creations will behave is imperfect
 - We need to check our work, because we will make mistakes

Independent Testing and Development of Software

- In what way is software different?
- Folklore:
 - "Programmers are optimists"
 - The implication is that programmers make poor testers
 - Economics: "Programming costs more than testing"
 - The implication is that programming is a higher-skill profession
- How valid is the folklore, and how much is due to the current state of the art in testing?

Explicit Specifications

- Software involves multiple people
 - At least a programmer and a user
 - But usually multiple programmers, testers, etc.
- Any team effort requires mutual understanding of the goal
 - A specification
 - Otherwise, team members inevitably have different goals in mind

Specifications Change

- Why?
- Many software systems are truly “new”
 - Differ from all that went before in some way
 - Initial specification will change as problems are discovered and solved
- The world is changing
 - What people want
 - The components you build on (e.g., the OS version)

Software Specifications

- Software specifications are usually
 - in prose
 - imprecise
 - out of date
- Current state of specification is not conducive to automation
 - Not consumable by tools
 - Without a spec, there is nothing to check

Finite Resources

- Organizations make trade-offs
 - Not all goals can be achieved
 - Because resources are finite
- \$'s express relative costs among goals
 - Goals that are hard to quantify pose a problem
 - E.g., correctness, completeness

"We have 2 months, 5 programmers, and 2 testers. Here is a priority list of features. A feature is finished when it passes all of the tests for that feature; a programmer does not move on to a new feature until all higher priority features are finished or assigned to other programmers. We start now and ship whatever features are finished in 60 days."

Summary of the State of the World

- Software development today relies overwhelmingly on the coder/tester model
- Typically half of the expense in developing a software product is in testing
 - And overwhelming, this testing is low tech

Testing Topics (Preliminary)

- Industry practices
 - Code coverage
 - Black-box and white-box testing
 - State-of-the-art commercial tools
- Testing theory
 - Hardness results, testing finite state machines
- Survey of research problems in testing
 - E.g., fault injection

Dynamic Analysis Topics (Preliminary)

- Efficient tracing
- Code instrumentation
- Deriving invariants from traces
- Monitoring long-running systems
- Commercial tools
 - E.g., Purify

A Theme: Specifications

- Specifications are needed for *any* technique
 - Why? Because no tool can divine what the software is supposed to do.
- Every method is a variation on:
 - Get people to say something in two different ways
 - Check the two versions for consistency
 - E.g., variables' types and their actual usage
 - E.g., test cases and the compiled code

Specifications (Cont.)

- *Every* technique relies on specifications
 - If only the semantics of the language
- The current state of specification is poor
- How can we get more specifications into programs?
 - Partial specs
 - Lightweight specs