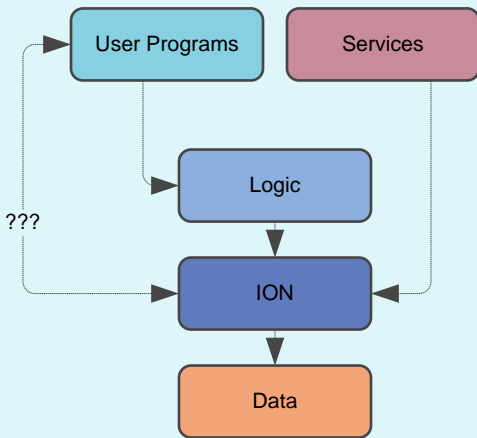


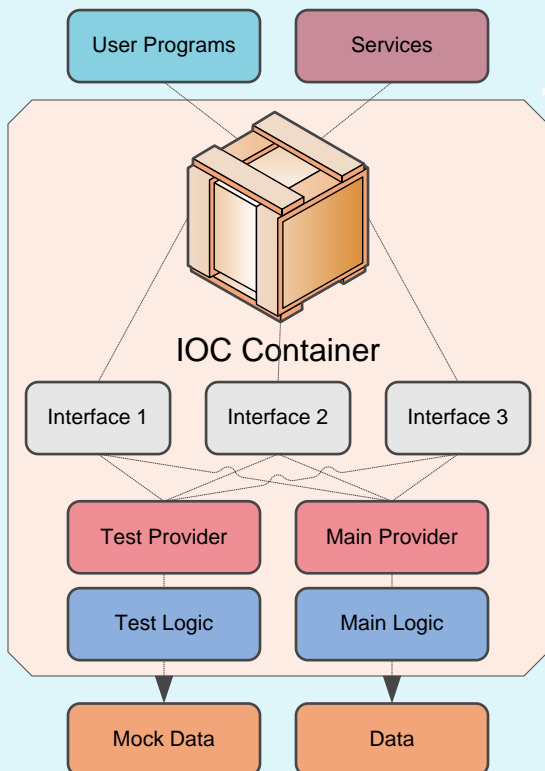
Current

- Replacing any single chunk is almost as difficult as replacing all of them.
- But, extending functionality by adding new programs, or hacking into existing programs, is relatively easy.
- This is how we end up with a ton of programs, each rather direct, with their own methods of interaction all separated and undefined



Layered

- Replacing any single chunk now only impacts the level above.
- Extending functionality requires extending the Logic and ION layers.
- We would end up with a more distinct system that does not refer directly to the data.
- However, note that a change to the data is still **very likely to bubble up the chain**: data change impacts ION, which impacts the logic, usw.
- Also note that it is hard to imagine testing the top layers without having a complete chain all the way down to the data.
- Without testing OR constant vigilance (pick one), the code will rot in the direction of inflexibility. If users say "We want this one little change", we may still have to say, "actually, that change is very difficult."



Inversion of Control

- High-level modules do not depend on low-level modules or the other way around; they both use abstractions, supplied and linked by the magic box.
- Inversion of control principle: "Abstractions should not depend upon details. Details should depend on abstractions."
- Interface segregation principle: "Clients should not be forced to depend upon interfaces that they do not use."
- Makes sense, right?
- This makes the system obviously a lot more complex, at first
- But it keeps almost everything loosely coupled and easily changeable, BECAUSE we have done the work ahead of time to make sure things are as separate as they possibly can be
- Makes any interfaces and functionality very explicit, by defining them
- Keeps things small (all the boxes and lines look scary, but each of them contains a much smaller amount code than the other diagrams, in theory)
- Allows multiple providers for a particular interface, for testing or decorating existing providers
- Testing individual pieces becomes very possible (also: WE SHOULD DO IT!)
- Extending the system (correctly) defines new interfaces or modifies existing interfaces in a (hopefully) sane, centrally managed way
- Hard part: Abstracting queries and reporting, while keeping performance