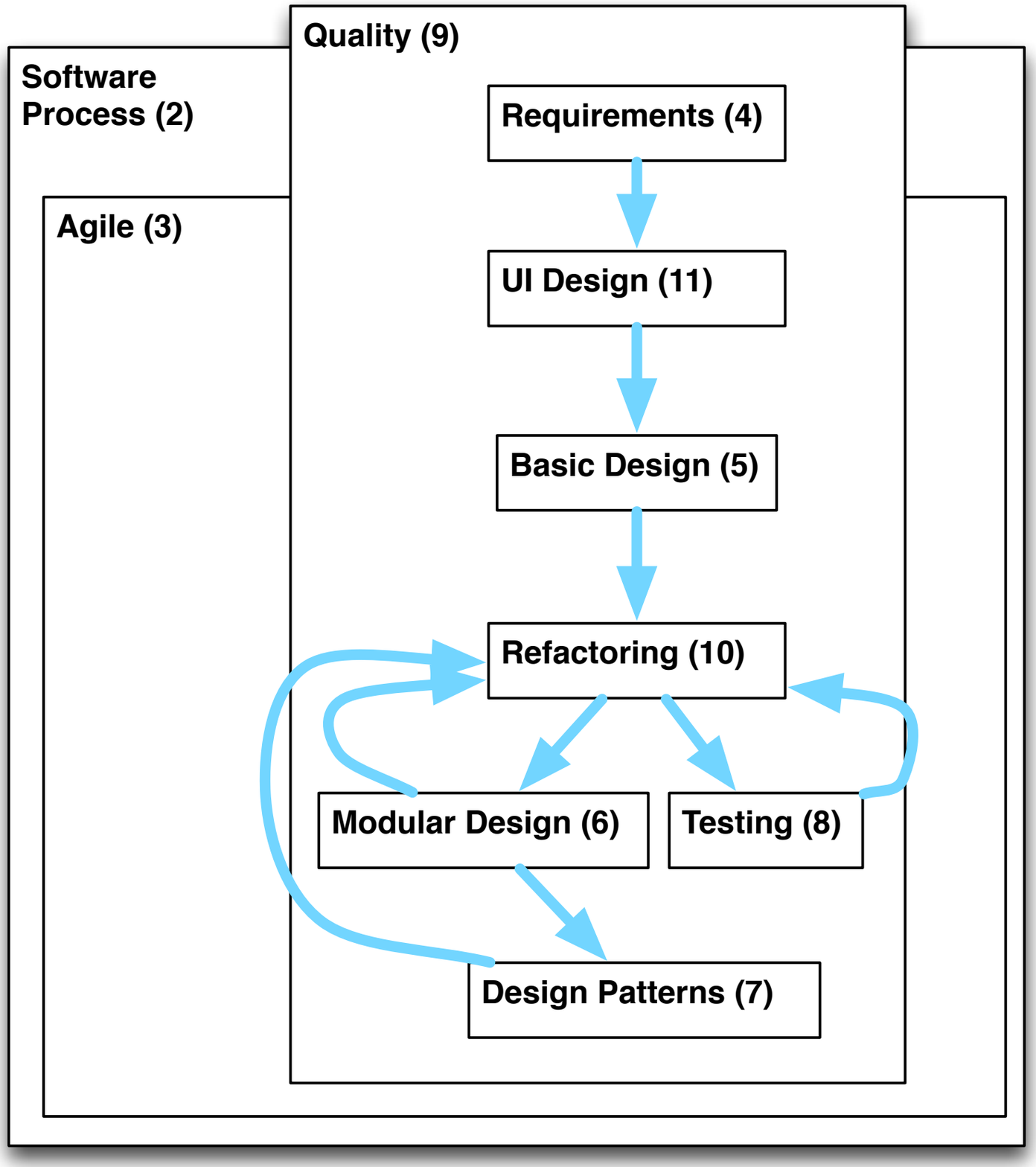
CS310 - REVIEW

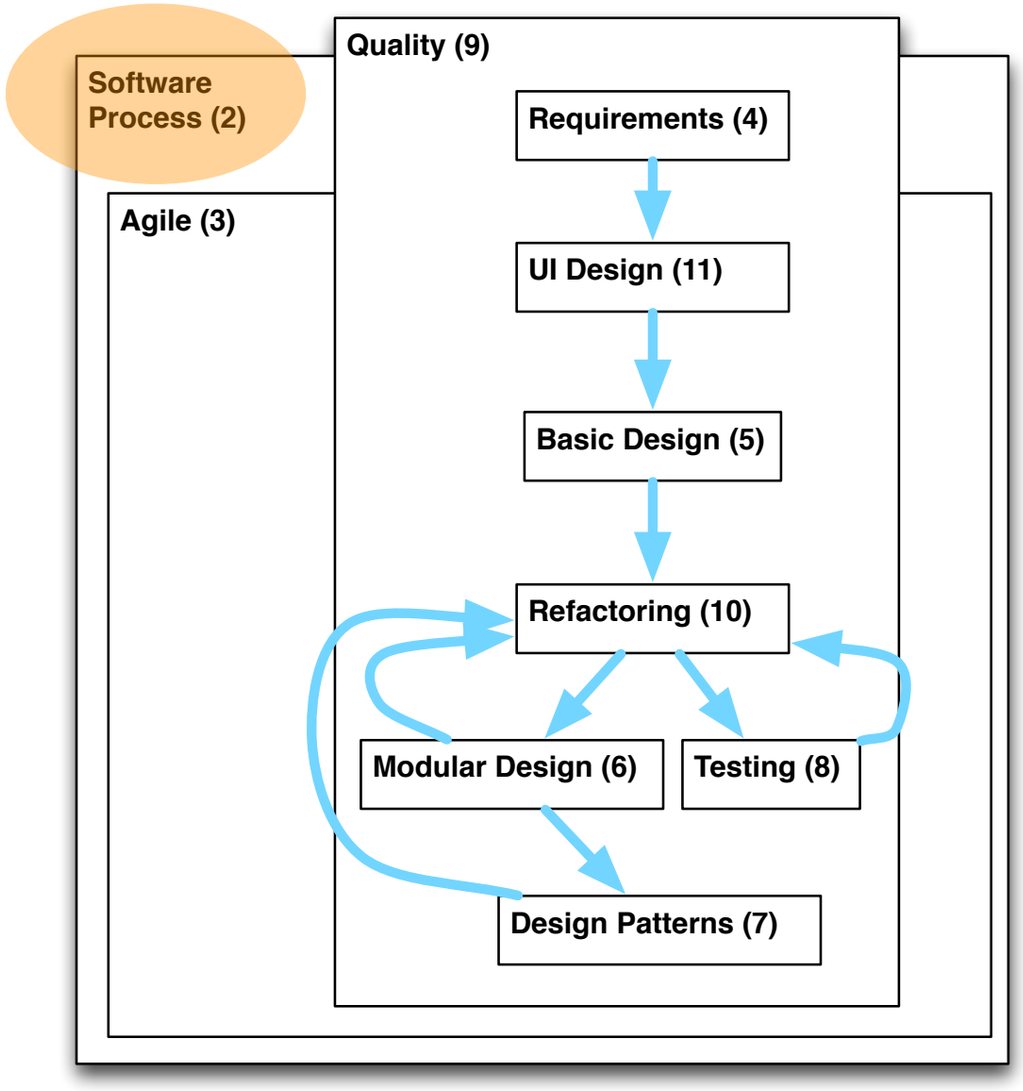
Topics

- Software Process
- Agile
- Requirements
- Basic Design
- Modular Design
- Design Patterns
- Testing
- Quality
- Refactoring
- UI Design

How these things relate



Software Process

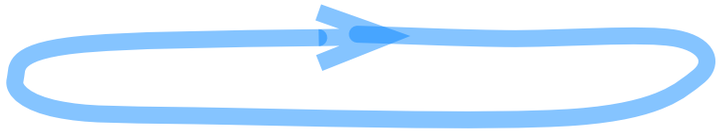


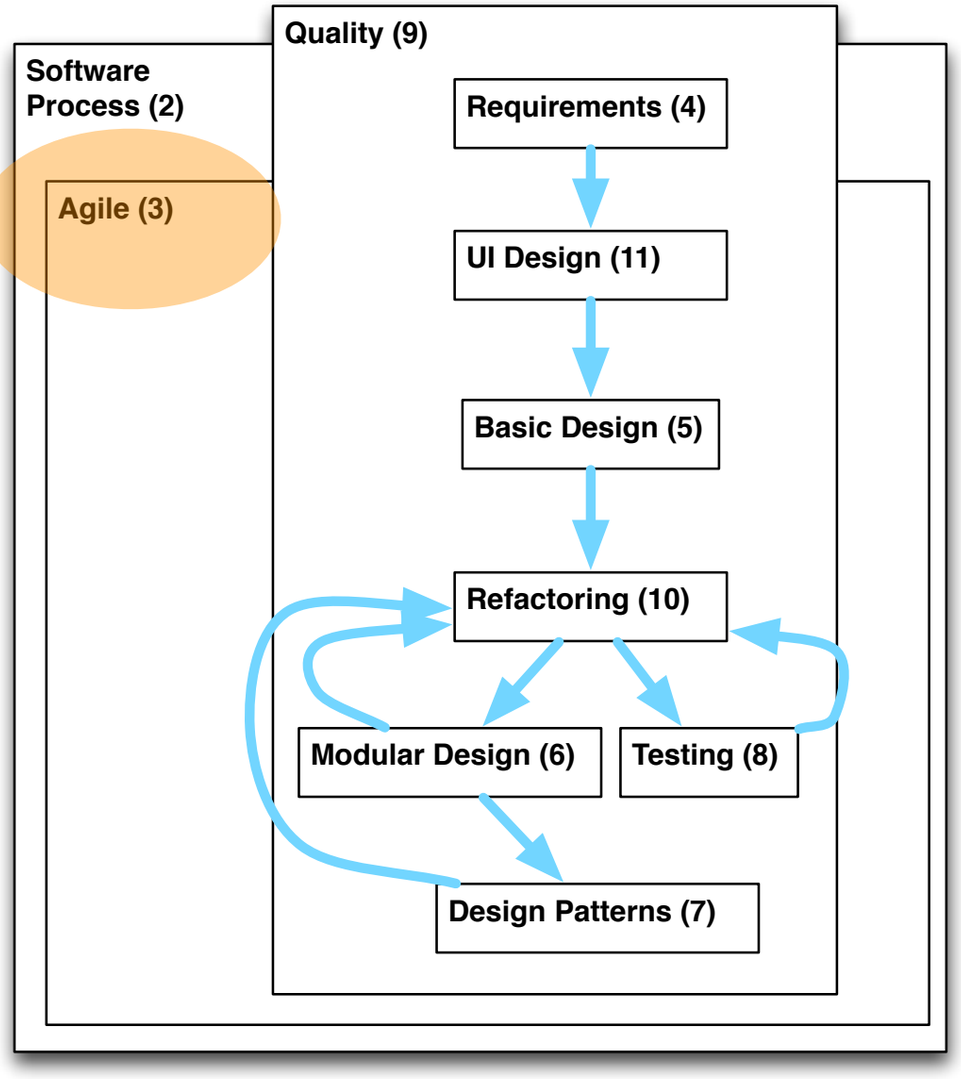
describe benefits of using a software process

describe waterfall and spiral model including drawbacks

describe the importance of agile methods

Agile





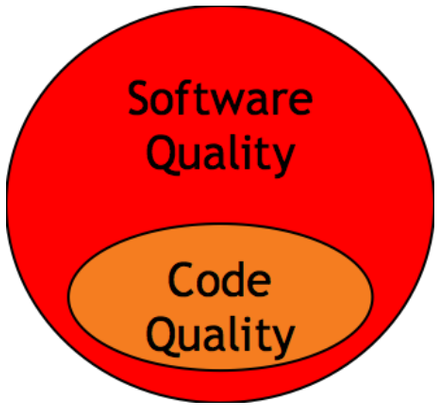
Test-Driven Development (tests written first)

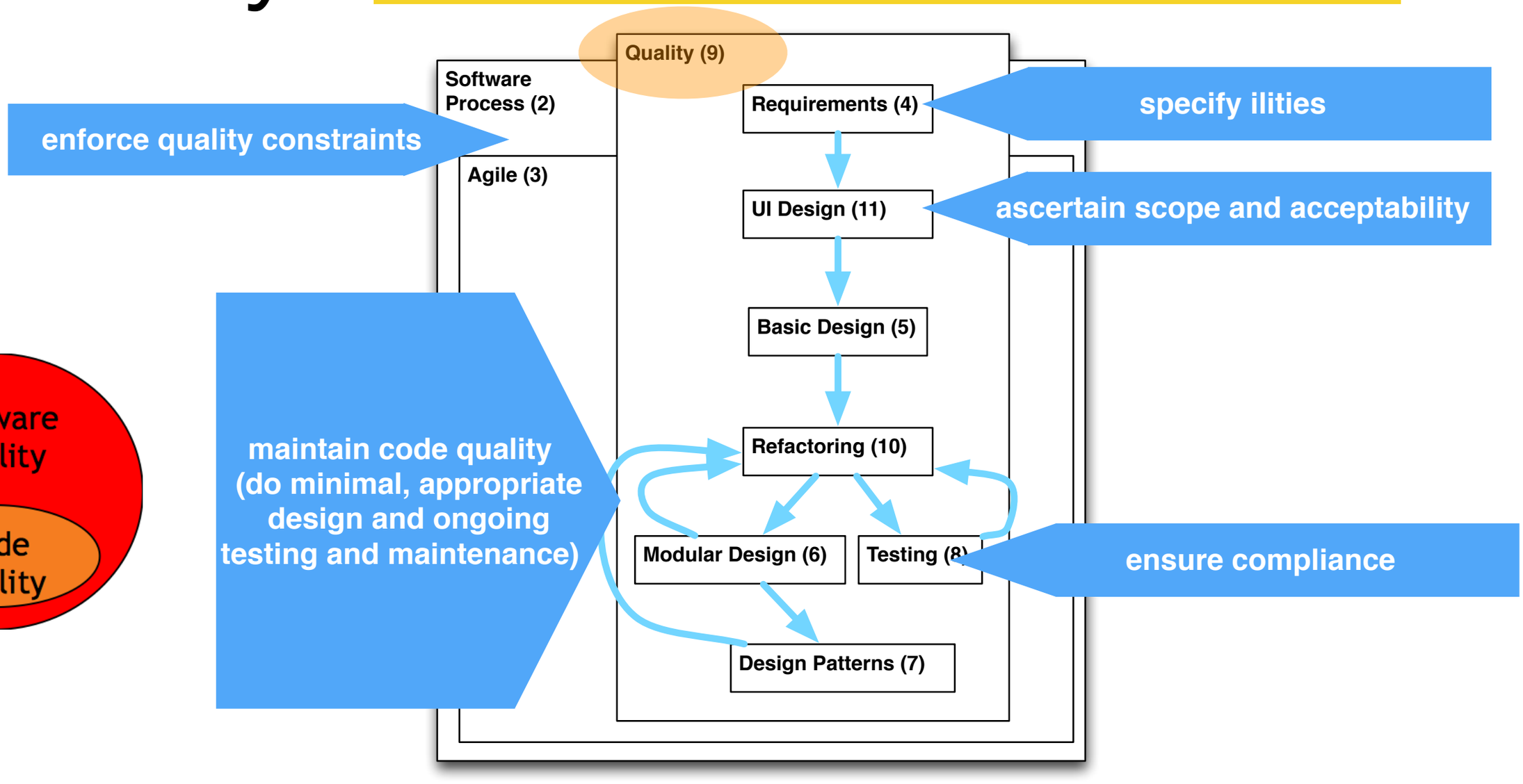
Scrum - sprints, backlogs ceremonies

describe the importance of agile methods

Quality

Quality concerns span the lifecycle



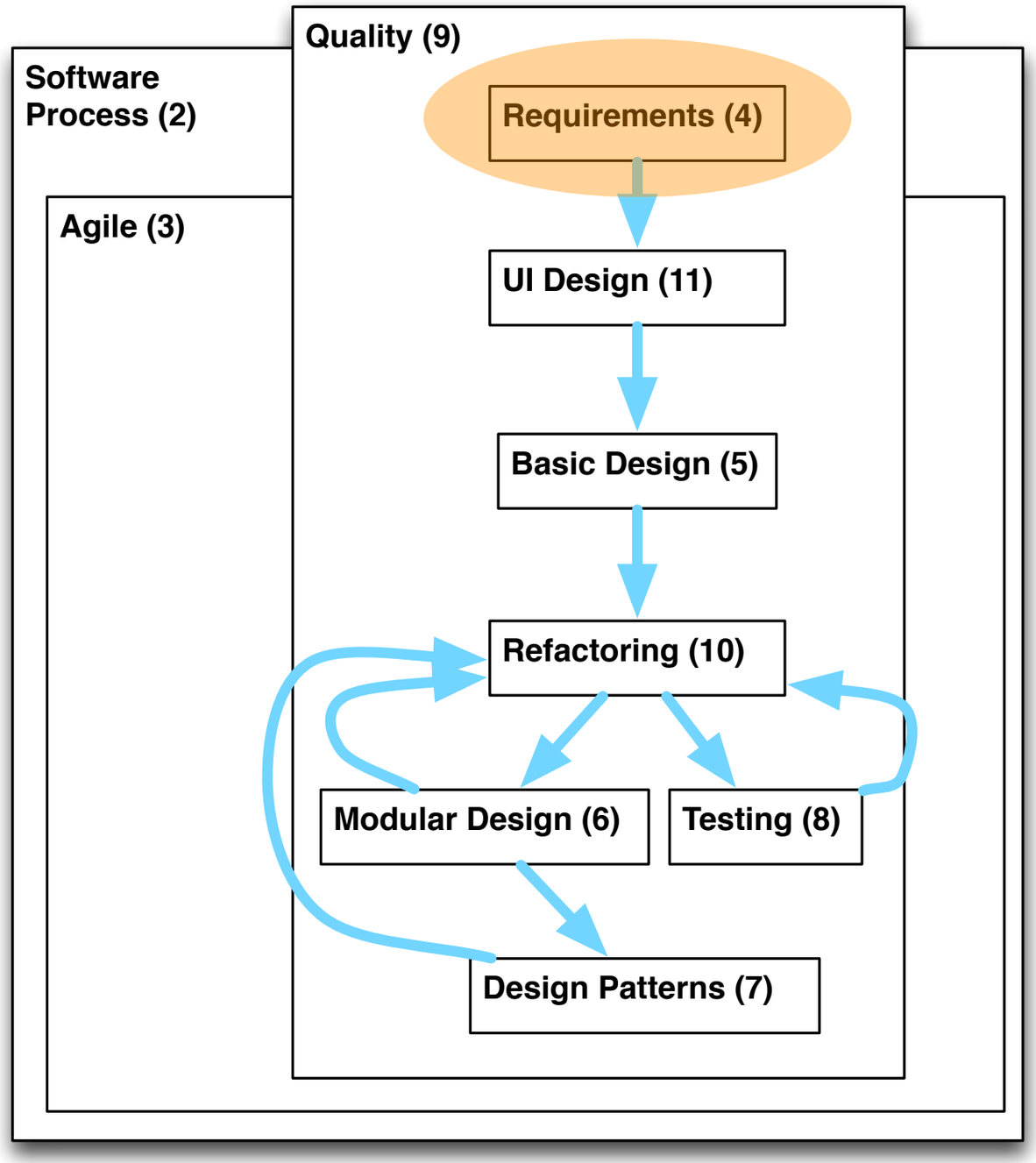


Design Quality influences Code Quality

Quality is hard to measure, but there are many indicators

Mechanisms like code review, pair programming, refactoring, improve quality

Requirements

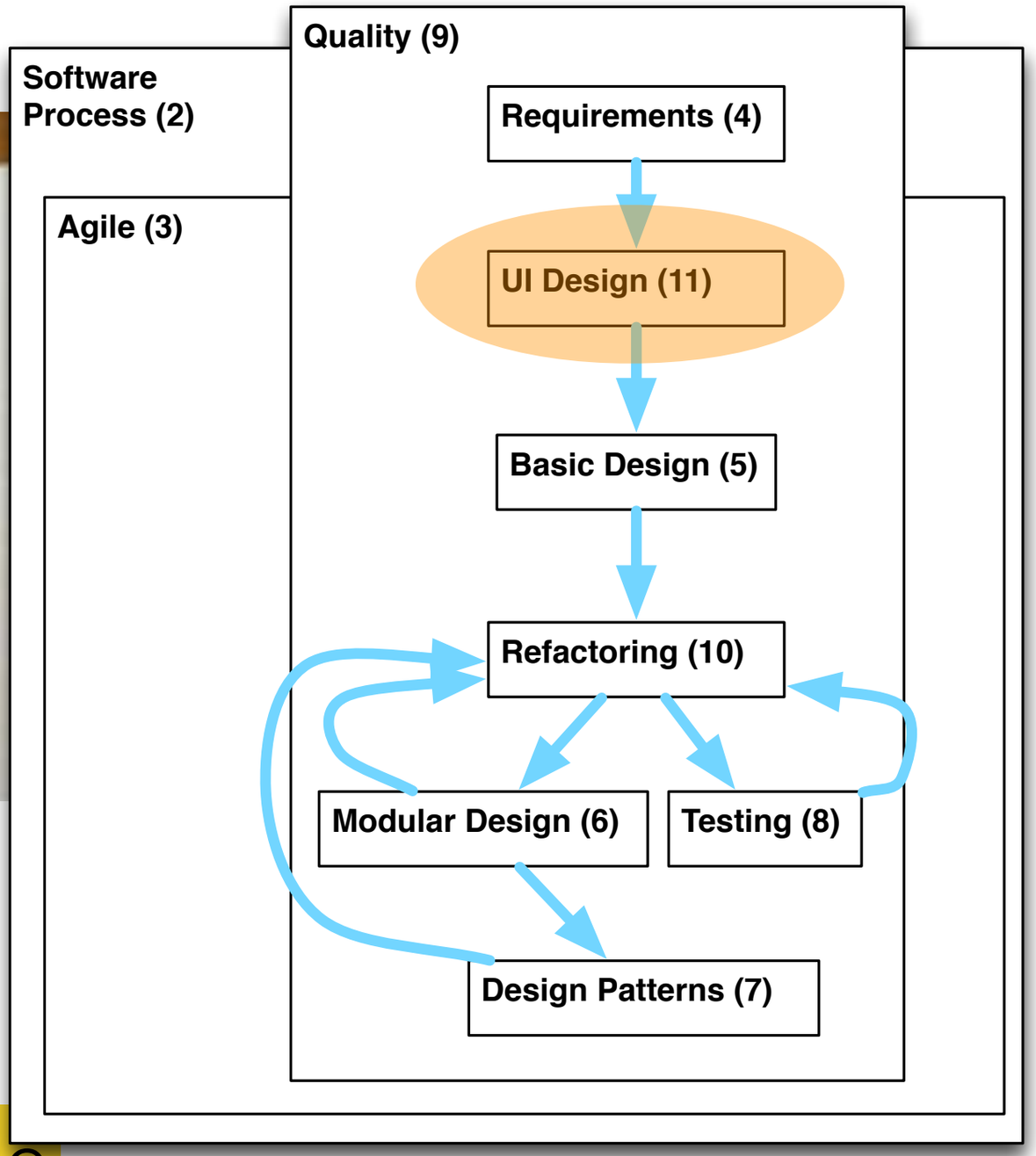


Why requirements are needed

How to elicit requirements

User stories, and how to write good ones (INVEST)

UI Design



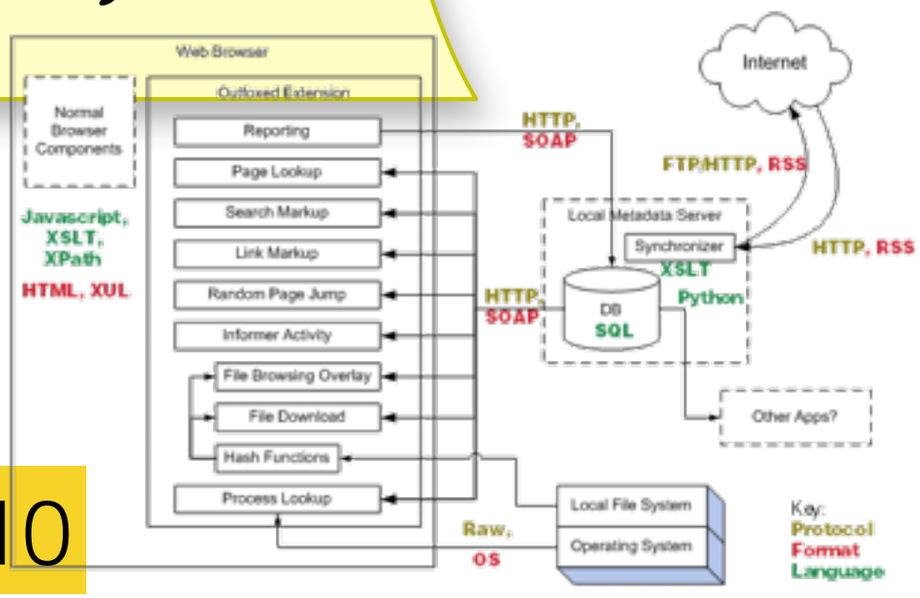
Nielsen Principles of Design

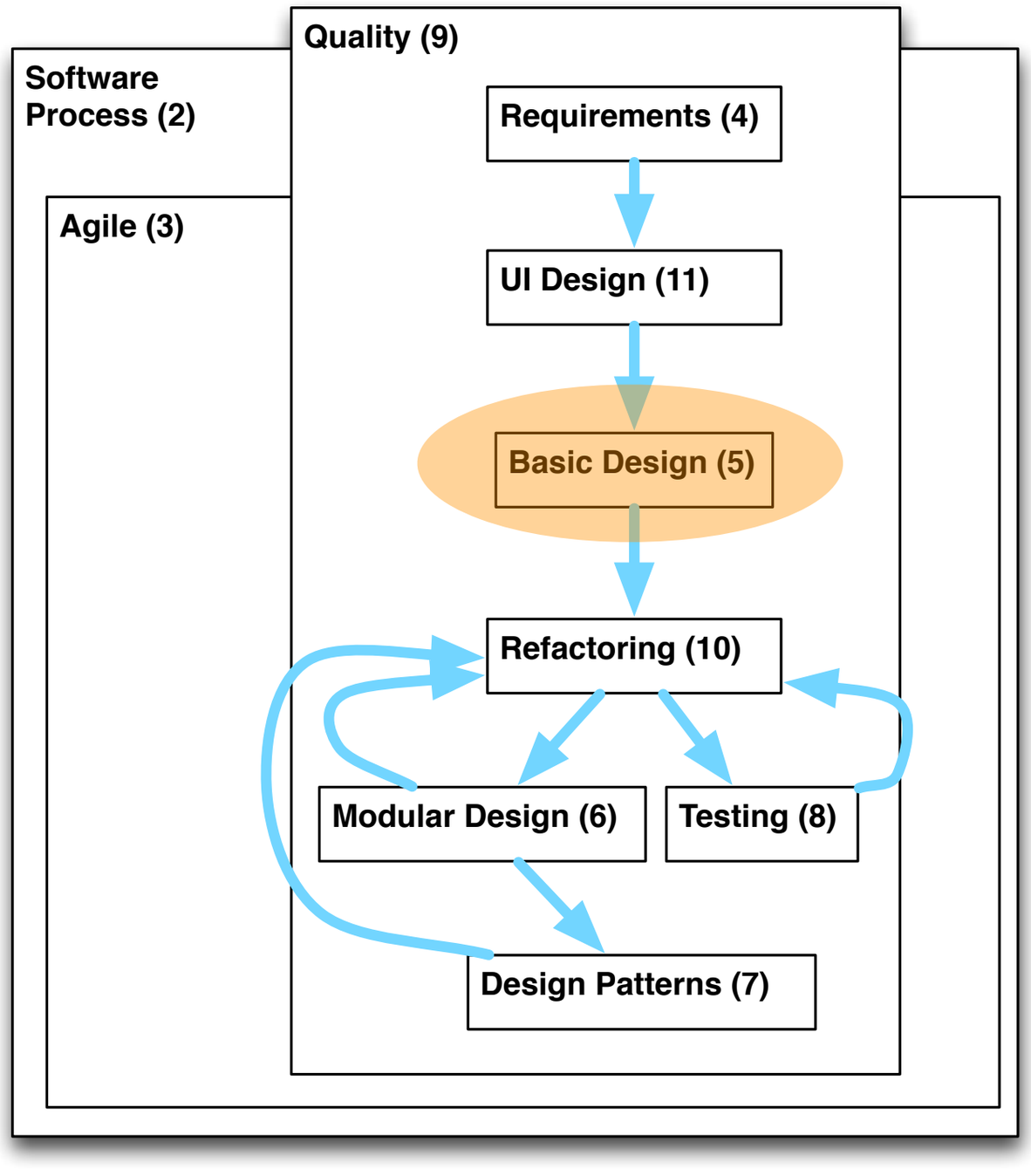
Appropriate components for usage

Rapid/Lighweight Prototyping Approach & Usability Testing

Basic Design

We also discussed architectural design, as a way to bridge the gap between UI/ Requirements and Detailed Design/Code.



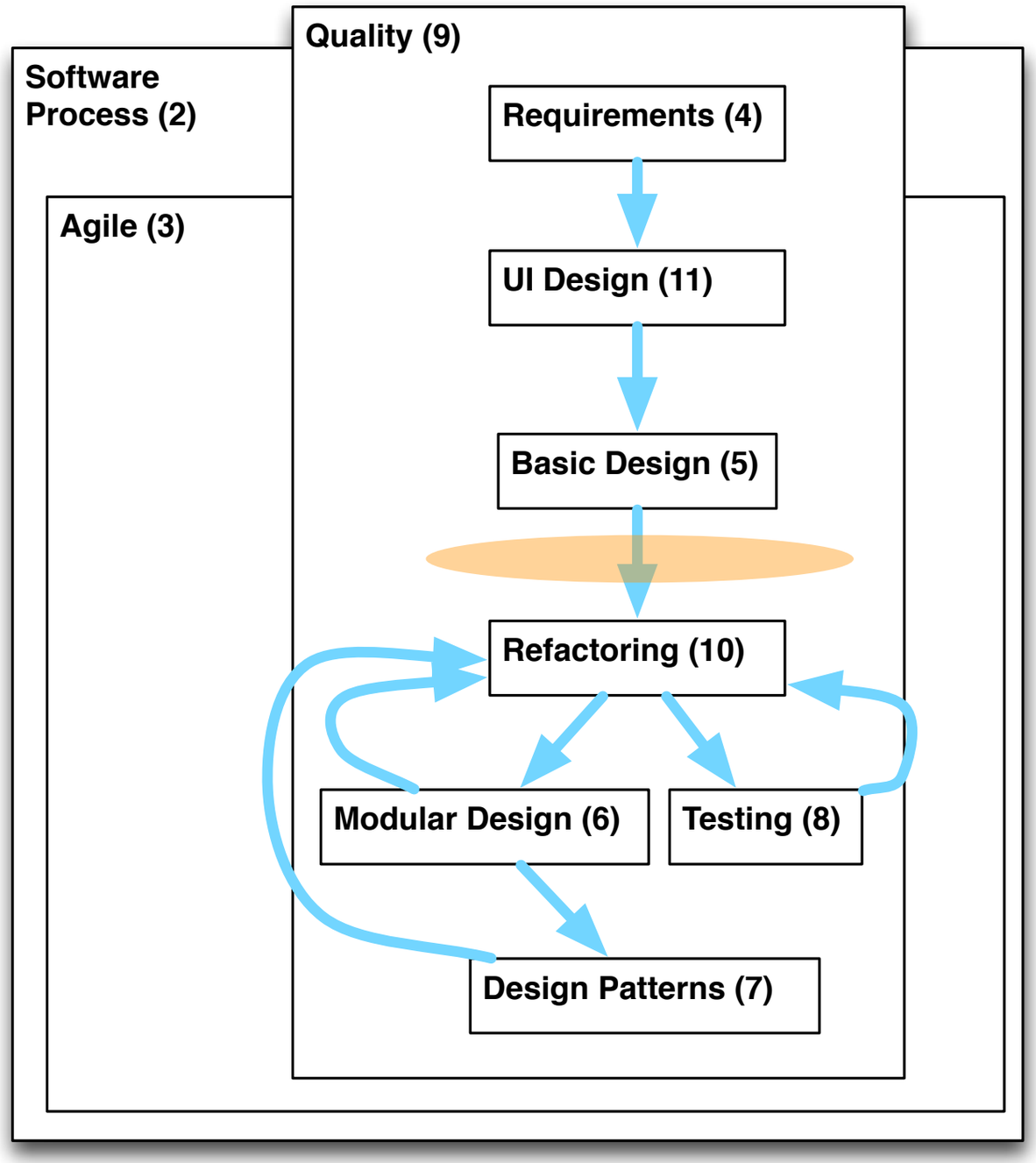


From CS210

UML Sequence Diagrams & Class Diagrams

Naming conventions and syntax

Coding!

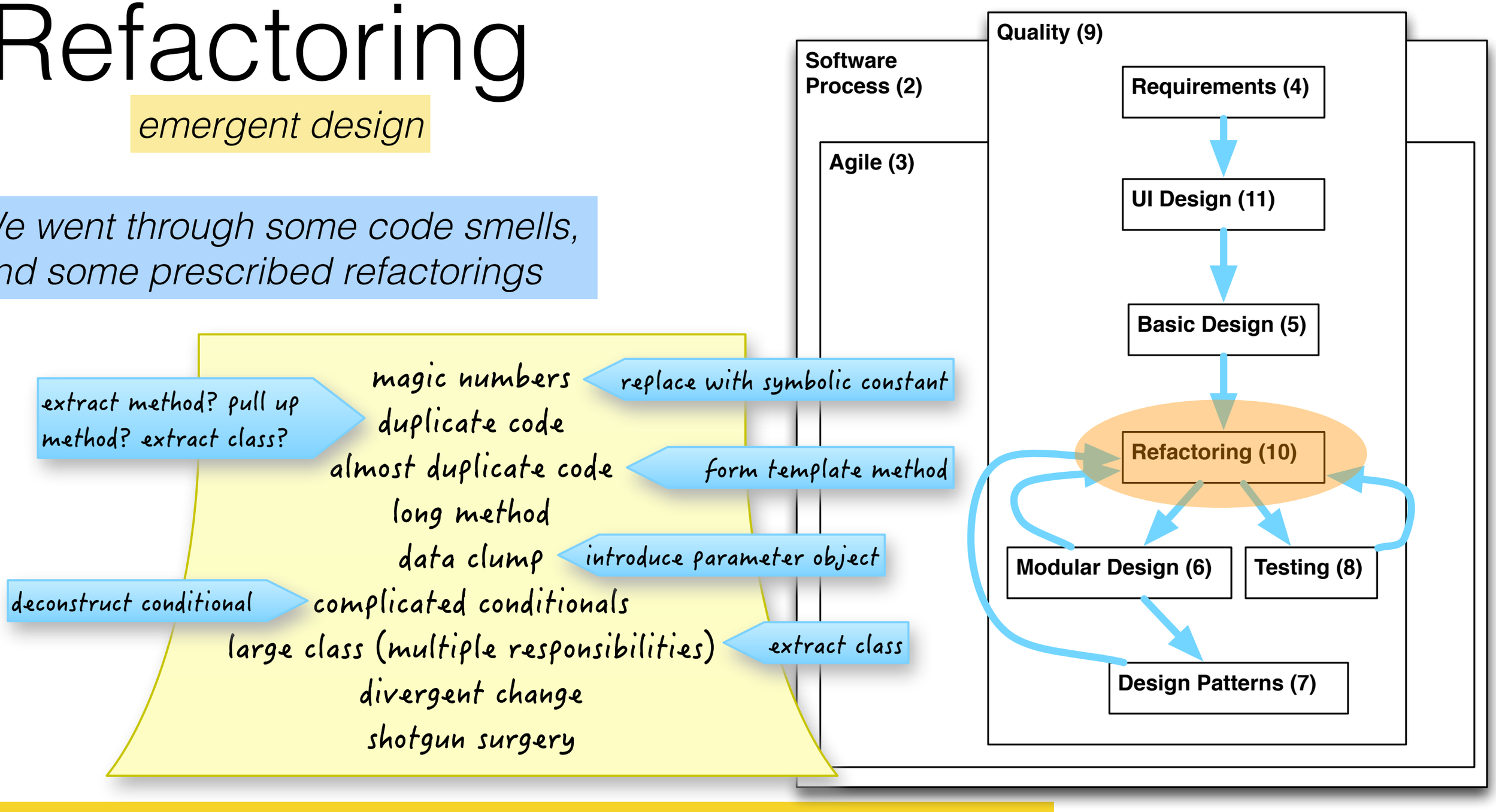


This was covered in labs, and earlier courses

Refactoring

emergent design

We went through some code smells, and some prescribed refactorings



Now that you have code, it's time to refactor!

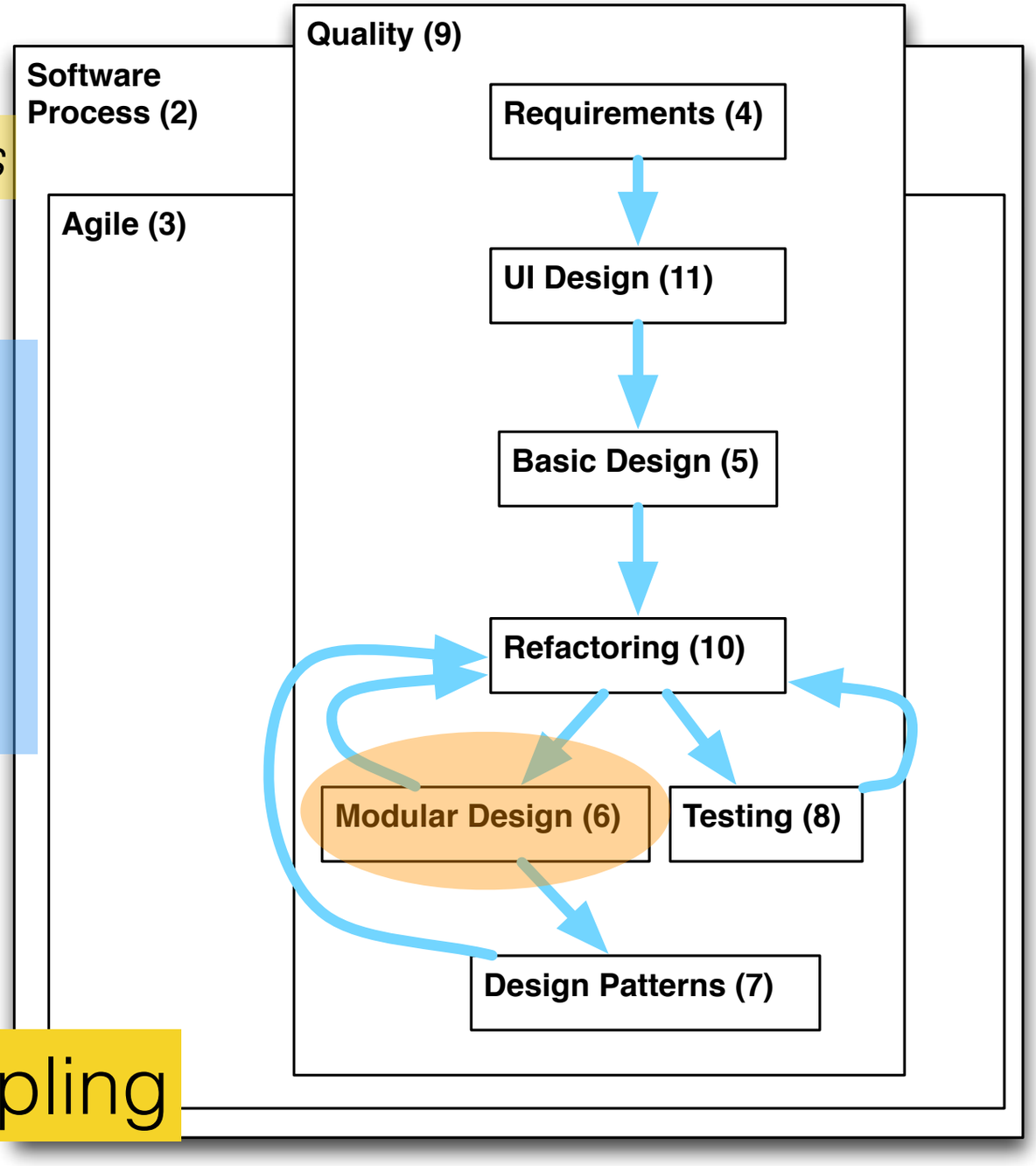
Why refactor? When to refactor? When not to refactor?

Looked at code smells, and how they necessitate refactorings

Modular Design

these are all principles that motivate refactorings

***Pragmatic Programmer**:*
Eliminate Effects Between Unrelated Things –
design components that are:
self-contained,
independent,
and have a single, well-defined purpose



strong cohesion versus loose coupling

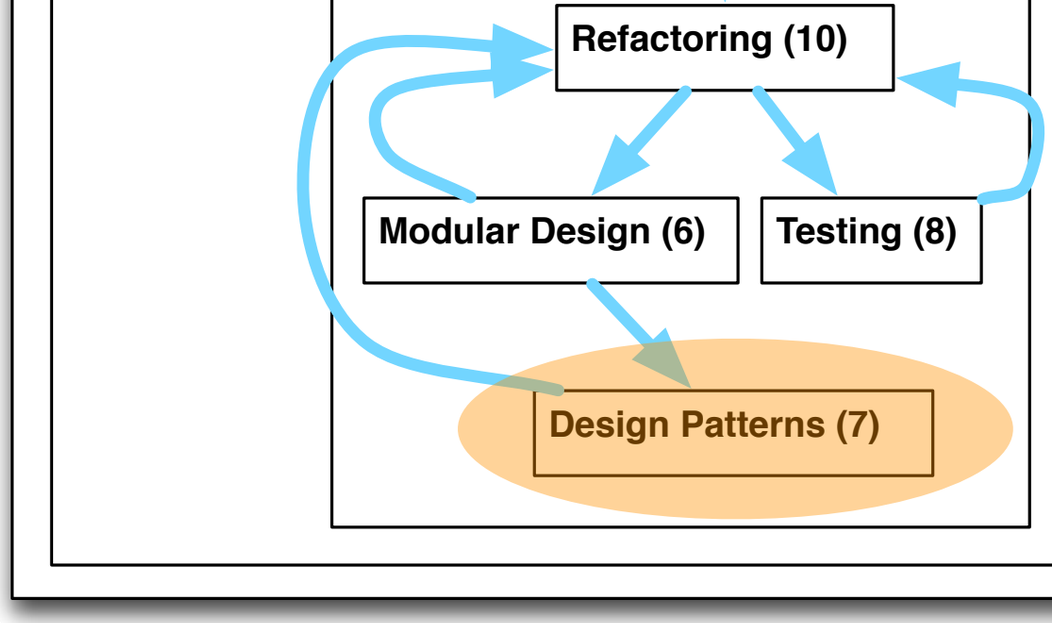
information hiding; Liskov Substitution Principle; Open/Closed Principle; Law of Demeter

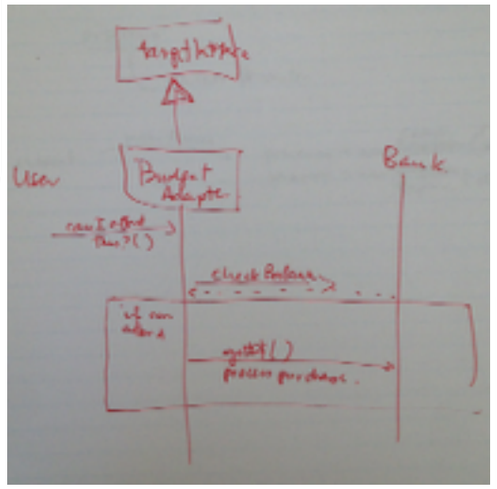
ultimately about localisation of reasoning; reduction of scattering and tangling of concerns

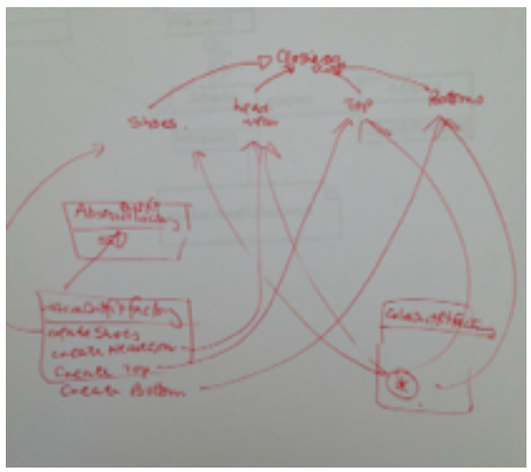
Design Patterns

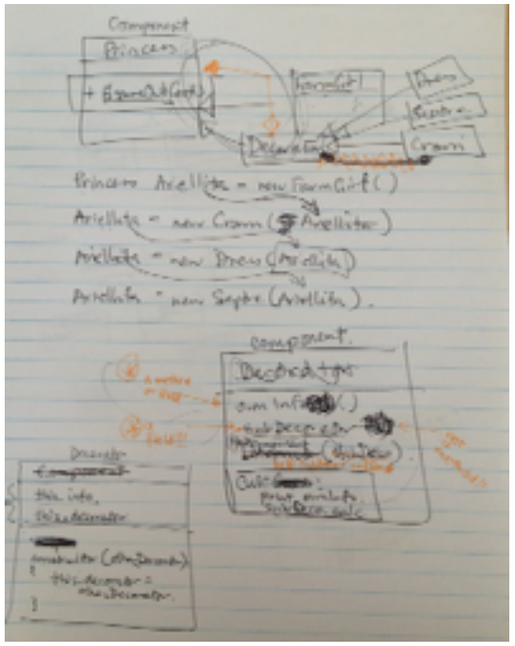
concentrate on the ones with stars

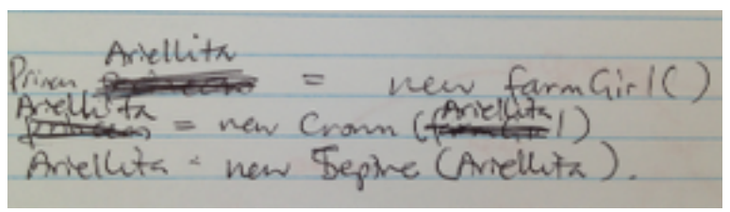
A typical motivation for the use of design patterns is detection of a code smell



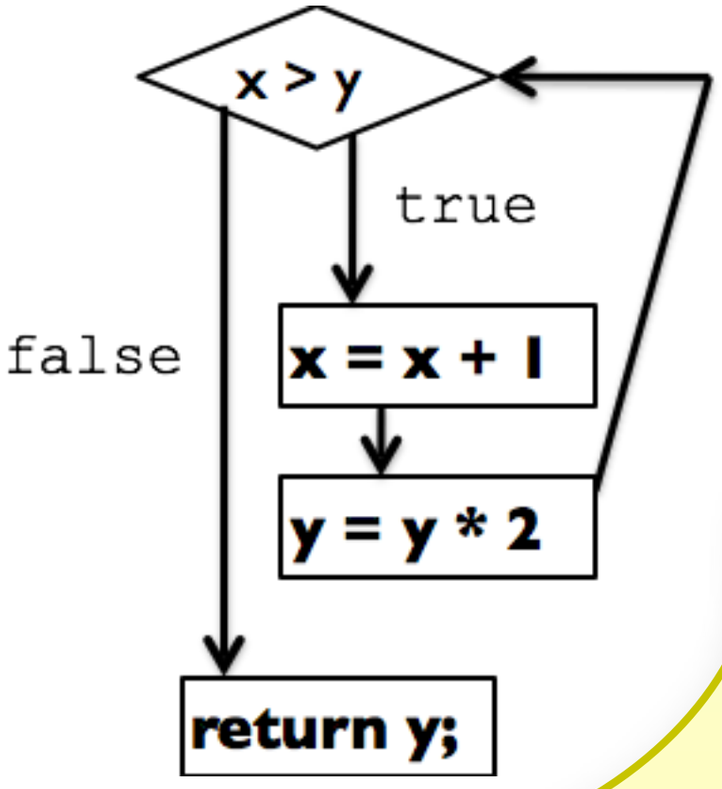


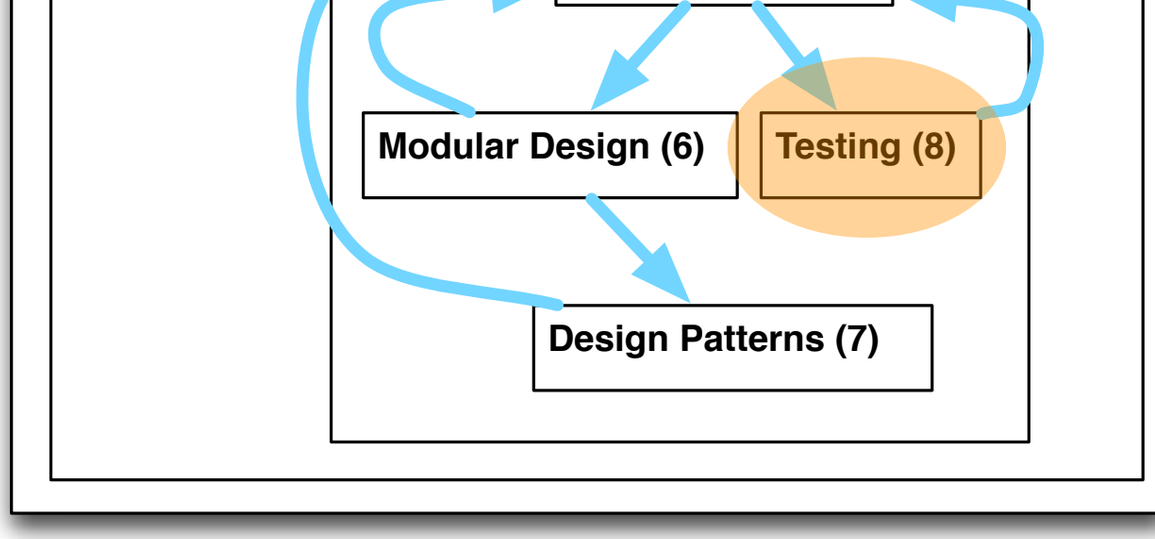






Testing





MAIN:
if "production": launch(new Client, new Server)
if "test": launch(new MockClient, new MockServer)

Mock Objects

Types of Testing (Unit, Regression, Integration, Acceptance)

Testing Tactics (Black Box, White Box)

Stopping Criteria (Equivalence classes, Boundary Tests, Coverage)

Who should test software?

How to Study

- Re-read the slides and follow the links for clarification and more context.
- Try to work on the sample systems to do examples of the things we covered
- Ask questions on Piazza
    - I will be holding virtual office hours on Sunday, at a time TBA. You can queue up your question (I might even get to it early!)