University of Salford MANCHESTER

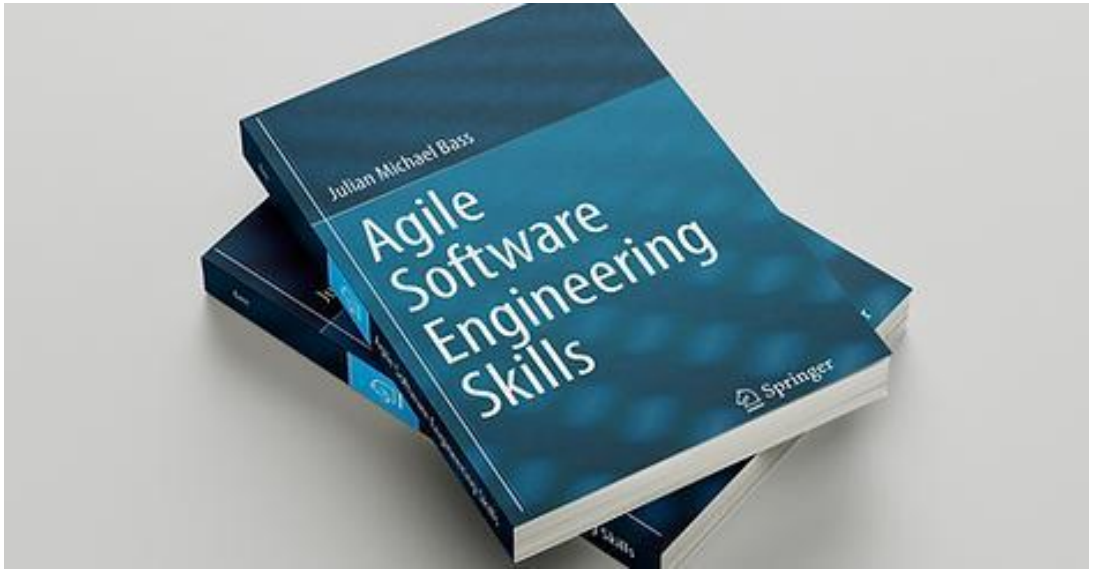
REDocelot

Agile Software Engineering Skills

Architecture

Chapter 8

Julian M. Bass



University of Salford MANCHESTER

REDocelot

Introduction

- Software structuring to
  - Achieve software requirements
  - Manage change by reducing dependencies between one part of the system and another
  - Enable team members to work on different parts of the project at the same time
- Will explore
  - Client-server, pipe and filter and layered architectures as well as
  - Design patterns such as the model-view controller

University of Salford MANCHESTER

REDocelot

Introduction

- Architecture is a process
  - Creative and design activities involved in making an architecture or system structure
- Architecture is an artefact
  - One or more outputs or deliverables
  - Set of architecture design models that describe how the system is organised
  - A record of decisions made

University of Salford MANCHESTER

REDocelot

Introduction

- Architectural design
  - Happens early in the development process
  - Overlaps with requirements gathering
  - Needs rework if new non-functional requirements are revealed
  - Requires consultation with stakeholders

University of Salford MANCHESTER

REDocelot

Contents

- Architecture in Agile
- Design Styles
  - Client-Server
  - Repository Architecture
  - Pipe and Filter
  - Layered Architecture
- Reference Architectures
- Design Principles
- Architecture Implementation

University of Salford MANCHESTER

REDocelot

Architecture in Agile

- Too much architecture design prior to starting development work on functionality
  - Known as ***big design up-front***
  - Invest effort without developing working code
  - Difficult to evaluate architecture until implemented
- Better to invest minimum effort required to get you up and running

University of Salford MANCHESTER

REDocelot

Architecture in Agile

- Refactoring
  - Re-structure and re-organise system without changing any functionality
  - Tidy up the structure of system from time to time
  - Include refactoring tasks in backlogs of work items
- Rework
  - Not the same as refactoring
  - Repeating same work, because it was poorly done first time
  - Rework is a sign of poor quality craft

University of Salford MANCHESTER

REDocelot

Architecture in Agile

- Planned Refactoring
  - Create a roadmap for re-architecting for significant new features
  - Consider and explain the need for architecture re-design
  - Think carefully about the implications of non-functional requirements
- Architectural Abstraction
  - Low-level architecture is about the structure of individual services
  - Enterprise architectures comprise systems of systems
  - Use architecture to clarify different levels of abstraction

University of Salford MANCHESTER

REDocelot

Design Styles

- Reusable structures that recur in specific types of application
- Implies a set of rules that team members follow
- Structure of complex system is simpler to explain and understand
- On-boarding new team members is easier
- Example of architectural reuse

University of Salford MANCHESTER

REDocelot

Design Styles

- Improves separation of concerns and hence maintainability
  - Components can be replaced without disrupting the rest of the system
- Avoids redundant software source code
- Team members can work on different components at the same time

University of Salford MANCHESTER

REDocelot

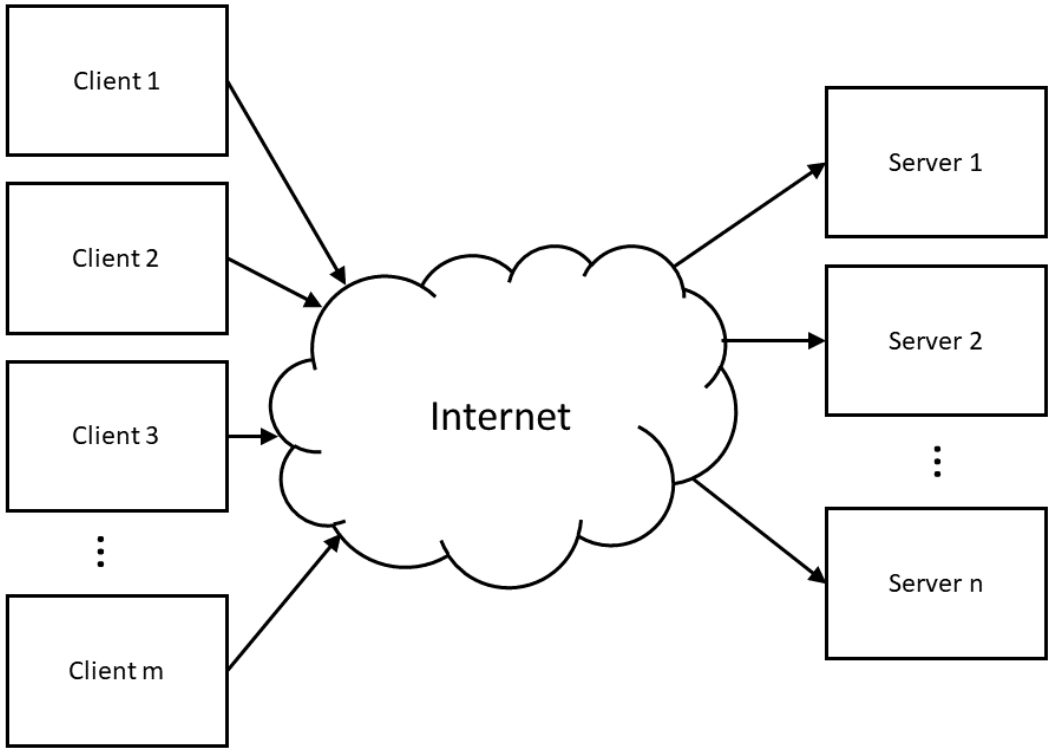
Design Styles

- Client-Server
  - Common to access software services using internet technologies
  - One or more servers provide services to clients
  - Clients might include mobile device applications (Mobile Apps) or web browsers
  - Services replicated on multiple servers
    - Supports increased loads
    - Redundancy for resilience
  - Performance subject to network bandwidth and difficult to predict

University of Salford MANCHESTER

REDocelot

Design Styles



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

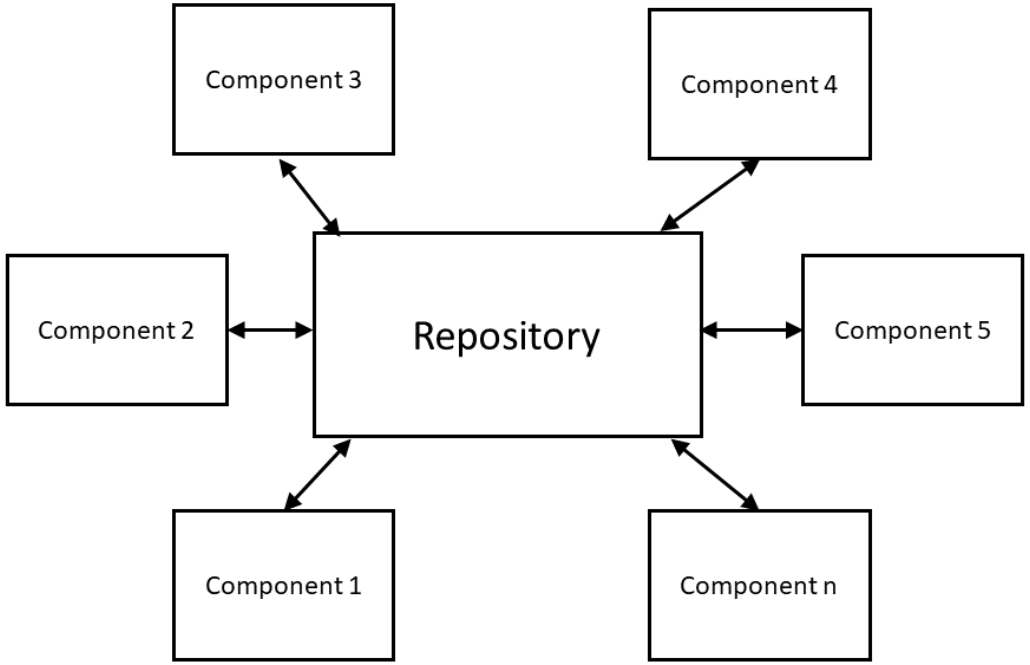
Design Styles

- Repository Architecture
  - Interactions between components happen through repository data transfers
  - Useful in data intensive applications, where a consistent view of shared data is required by all components
  - Components do not need to be aware of each other
  - Centralised storage model simplifies backup and data archiving
  - Repository is single point of failure

University of Salford MANCHESTER

REDocelot

Design Styles



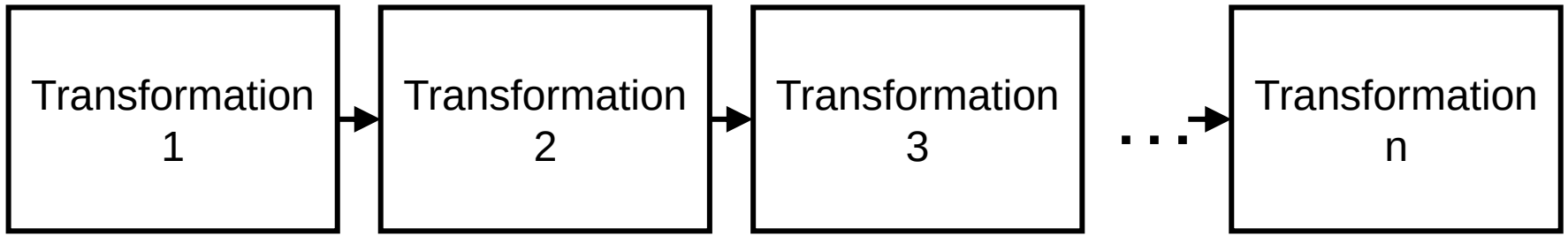
J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

Design Styles

- Pipe and Filter
  - Chain of transformation components
  - Each process input data to produce some output
  - Organise the transformation flow into discrete processing stages



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

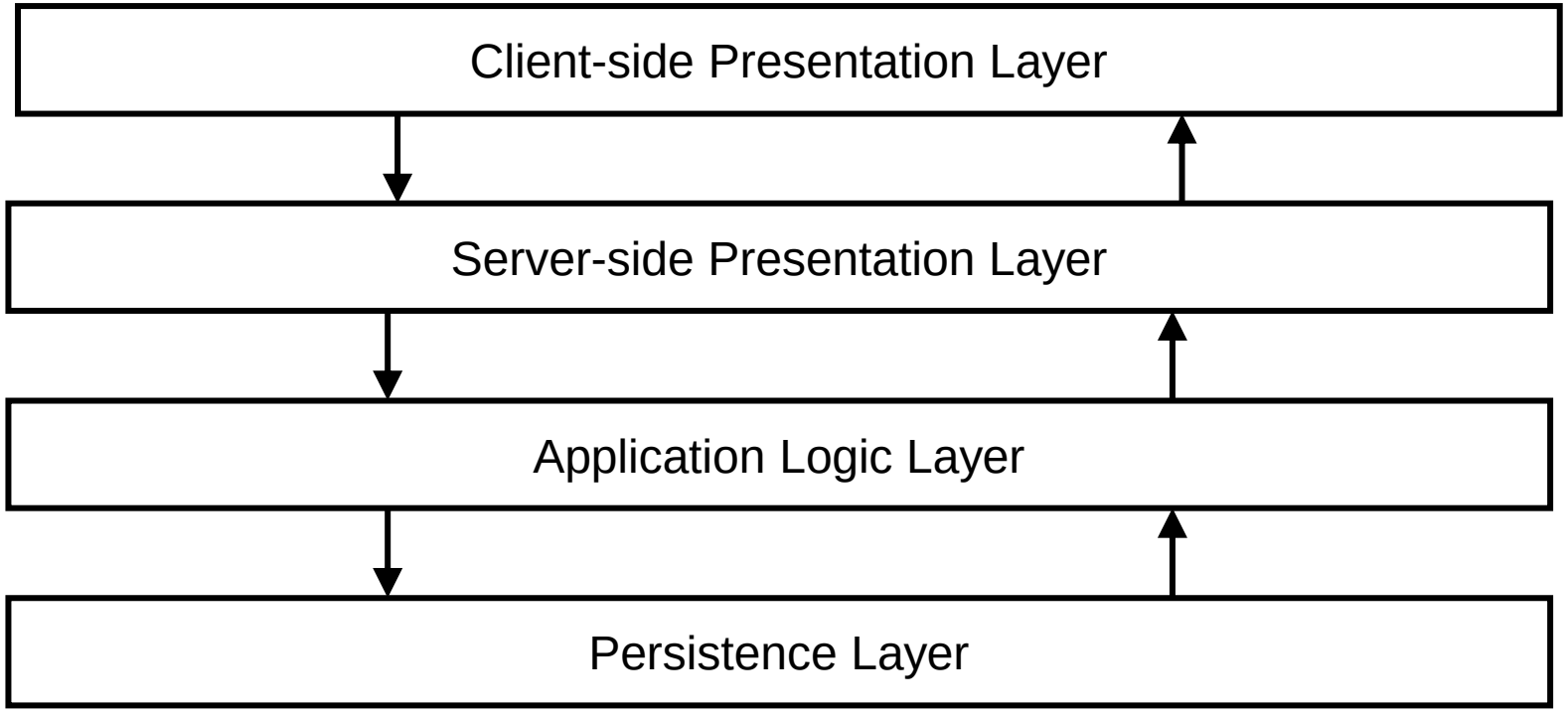
Design Styles

- Layered Architecture
  - Related functionality is grouped into a series of levels
  - Each layer provides services to the layer above
  - Data can flow down through the layers; as well as up
  - Requires discipline to ensure everyone adheres to the model

University of Salford MANCHESTER

REDocelot

Design Styles



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

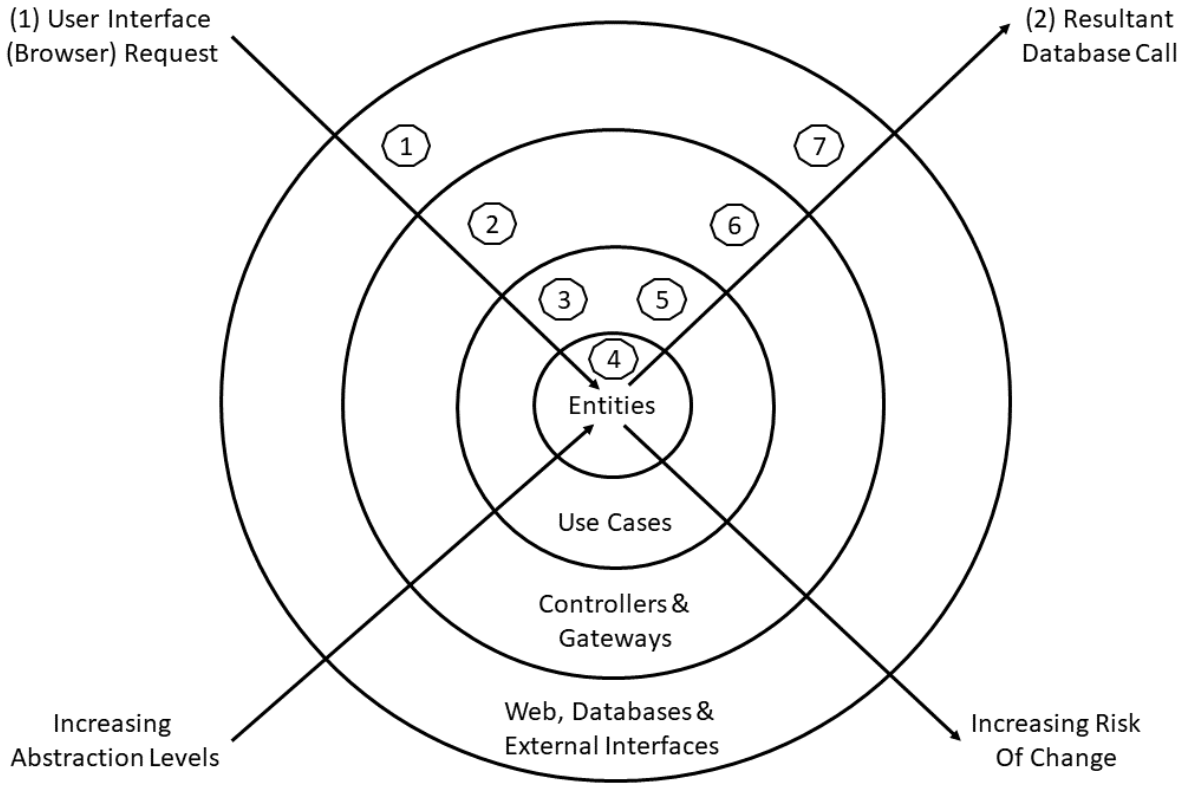
Design Styles

- Clean Architecture
  - Onion ring perspectives with the interfaces around the outside
    - Web, databases, devices and other external ***interfaces*** form a ring around the outside
    - Then a ring for our ***gateways and controllers***
    - Followed by ***use cases*** which encapsulate application specific functionality and business rules
    - In the centre, are ***entities*** which comprise abstract enterprise logic
- More frequent change tends to occur on outer rings

University of Salford MANCHESTER

REDocelot

Design Styles



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

Reference Architectures

- Highly documented skeleton of the overall system
- Include examples of using specific interfaces, APIs or services in the system
- Useful for
  - Induction of new members to the team
  - Ensuring everyone understands the rules or conventions

University of Salford MANCHESTER

REDocelot

Design Principles

- SOLID
  - Single-responsibility
    - Every class should have only one responsibility
    - Only be one reason to change a class
    - Expression of high cohesion within a class
  - Open–closed
    - Classes are open for extension but closed for modification
    - Use generalisations, such as inheritance or delegate functions, to extend classes

University of Salford MANCHESTER

REDocelot

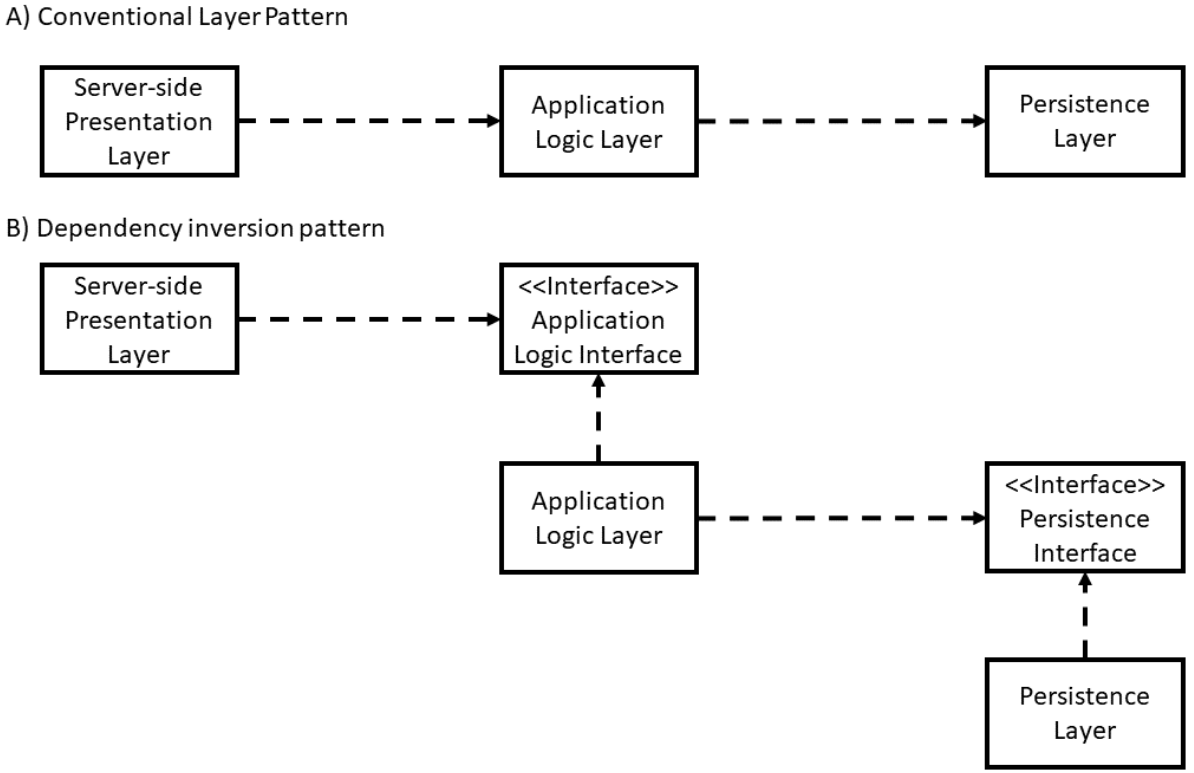
Design Principles

- SOLID
  - Liskov substitution
    - Design by contract
    - Children classes, that inherit properties from parents, can be substitutable for parents
  - Interface segregation
    - Achieve cohesive user interface design by developing role-based interfaces
    - Decouples different clients to simplify software maintenance and evolution
  - Dependency inversion
    - Introduce indirection between components, using ***interfaces*** or ***abstract classes***

University of Salford MANCHESTER

REDocelot

Design Principles



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_8

University of Salford MANCHESTER

REDocelot

Architecture Implementation

- Select an architectural style
- Create a simple structure that everyone can use
- Build a reference architecture
  - Assess and perhaps validate the chosen design style
  - Illustrates the interfaces between main components
- Then, start designing features within the reference architecture

University of Salford MANCHESTER

REDocelot

Exercises

- Exercises 8.1 and 8.5 encourage creation of a learning journal
- Exercise 8.2 Covers pipes and filters architectural style
  - Simple GitHub example
    - Bass, J. (2022). *PipeAndFilterExample* [Java]. https://github.com/julianbass/PipeAndFilterExample (Original work published 2021)
- Exercise 8.3 Covers layered architectural style
  - Simple GitHub example
    - Bass, J. (2022). *LayeredArchitectureExample* [Java]. https://github.com/julianbass/LayeredArchitectureExample (Original work published 2022)
- Exercise 8.4 Covers programming language choice

University of Salford MANCHESTER

REDocelot

Summary

- Architecture in Agile
  - Avoid ***Big Design Up Front***
  - Refactoring to refine architecture as product matures
- Design Styles
  - Client-Server
  - Repository Architecture
  - Pipe and Filter
  - Layered Architecture

University of Salford MANCHESTER

REDocelot

Summary

- Reference Architectures
  - Record design decisions
- Design Principles
  - SOLID
- Architecture Implementation
  - Implement simple skeleton, then build functionality