University of Salford MANCHESTER

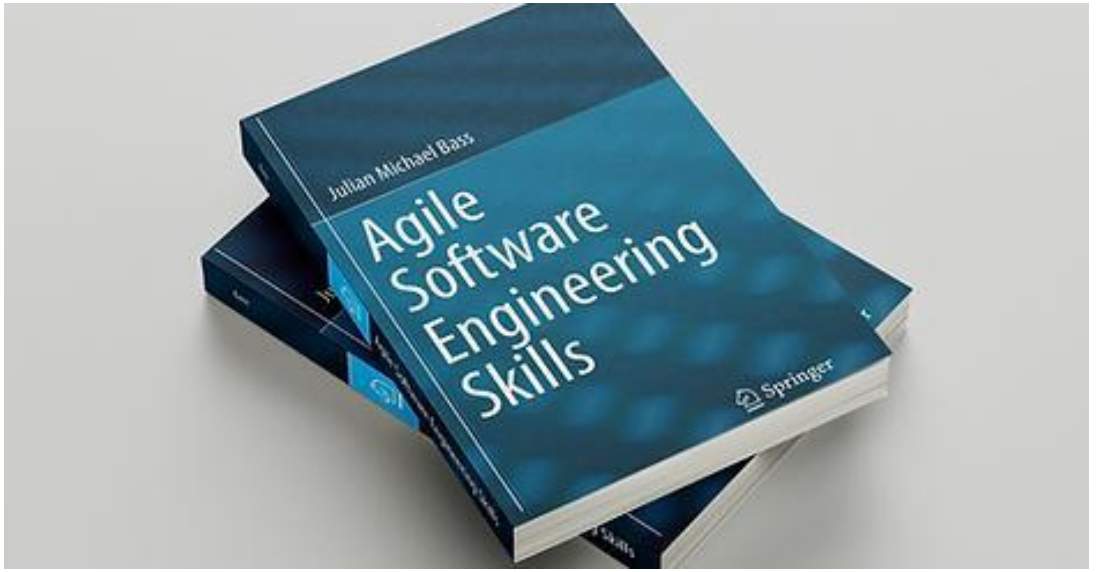
REDocelot

Agile Software Engineering Skills

Design

Chapter 9

Julian M. Bass



University of Salford MANCHESTER

REDocelot

Introduction

- We've established requirements
  - What our system must do (Chapter 7)
- We've created a high-level structure
  - Adopted an architectural style and created a reference architecture (Chapter 8)
- Now we can design specific features

University of Salford MANCHESTER

REDocelot

Introduction

- We will create class diagrams and object sequence diagrams to understand
  - How our software will be structured
  - How the moving parts interact during run-time
- Implement design patterns to solve recurring problems

University of Salford MANCHESTER

REDocelot

Contents

- Feature Driven Development
- System Modelling
- Class Diagrams
- Object Sequence Diagrams
- Technology Stack Selection
- Model-driven Engineering

University of Salford MANCHESTER

REDocelot

Feature Driven Development

- Features are independent pieces of functionality
- Provide end-to-end services to external actors
- Features provide value to customers
- End-to-end fulfilment of a user need
  - Client, business logic and persistence
- Incremental development consists of implementing features, one after another

University of Salford MANCHESTER

REDocelot

Feature Driven Development

- Features comprise working code
- We create acceptance tests for features
- We can demonstrate features and get feedback
  - Clients or users should be able to understand the goal

University of Salford MANCHESTER

REDocelot

System Modelling

- Develop visual representations of the system
- Models use graphical notations
  - Often based on one of the Unified Modelling Languages (UML)
- Models stimulate discussion, and ultimately consensus
  - On what something does or how something works
- Models record decisions about design

University of Salford MANCHESTER

REDocelot

System Modelling

- Two main types of model: static and dynamic
- Static models depict unchanging internal structure
- Dynamic models depict run-time behaviour
- Models of the world as it exists
  - Before our system is implemented.
  - ***As Is*** or ***domain*** models
- Models that describe the proposed new system
  - ***To Be*** models

University of Salford MANCHESTER

REDocelot

Class Diagrams

- Structural models
  - Current environment
  - Organisation of a system
- Components that make up a system
  - Their relationships with each other

University of Salford MANCHESTER

REDocelot

Class Diagrams

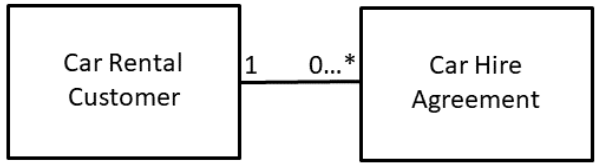
- Deriving Class Diagrams
  - Noun and Noun Phrases that describe a person, place, thing, quality or idea
    - Implemented in software as data items (attributes), data structures or classes
  - Verb and Verb Phrases describe actions
    - Implemented as methods or operations

University of Salford MANCHESTER

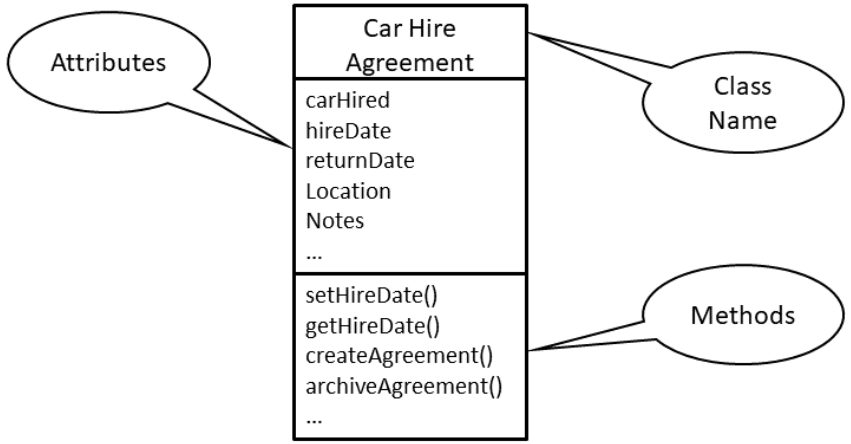
REDocelot

Class Diagrams

A) Simple classes and an association



B) Car hire agreement class (incomplete)



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_9

University of Salford MANCHESTER

REDocelot

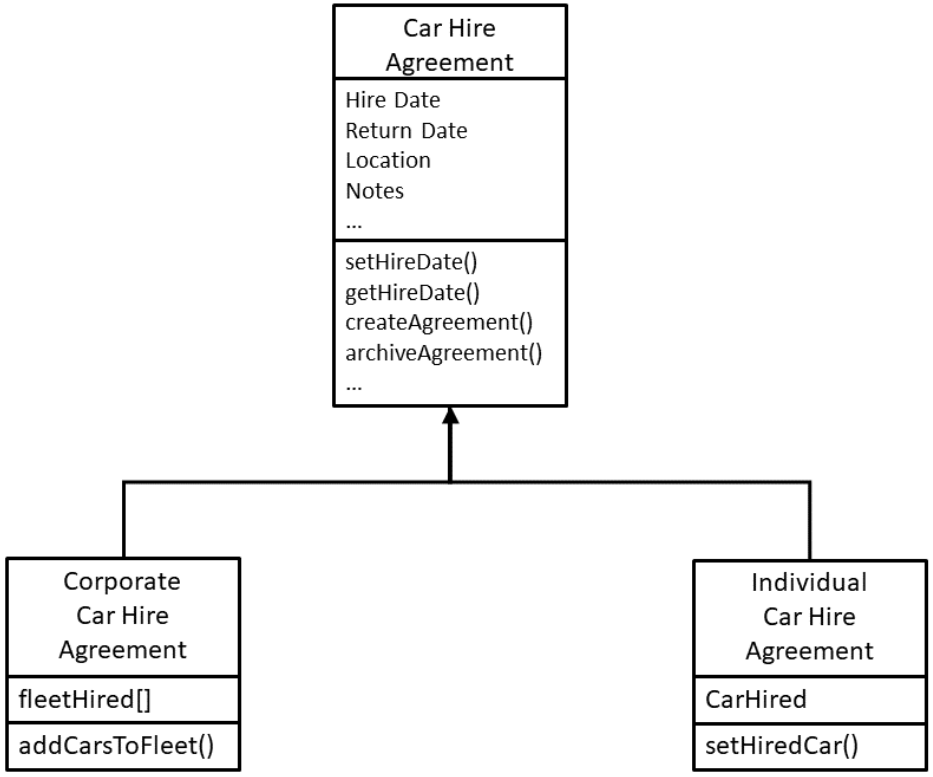
Class Diagrams

- Domain models
  - As Is models of system context
- High-level design models
  - Outline diagrams of classes and their relationships and attributes
- Detailed design models
  - Include method signatures and attribute data types

University of Salford MANCHESTER

REDocelot

Class Diagrams



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_9

University of Salford MANCHESTER

REDocelot

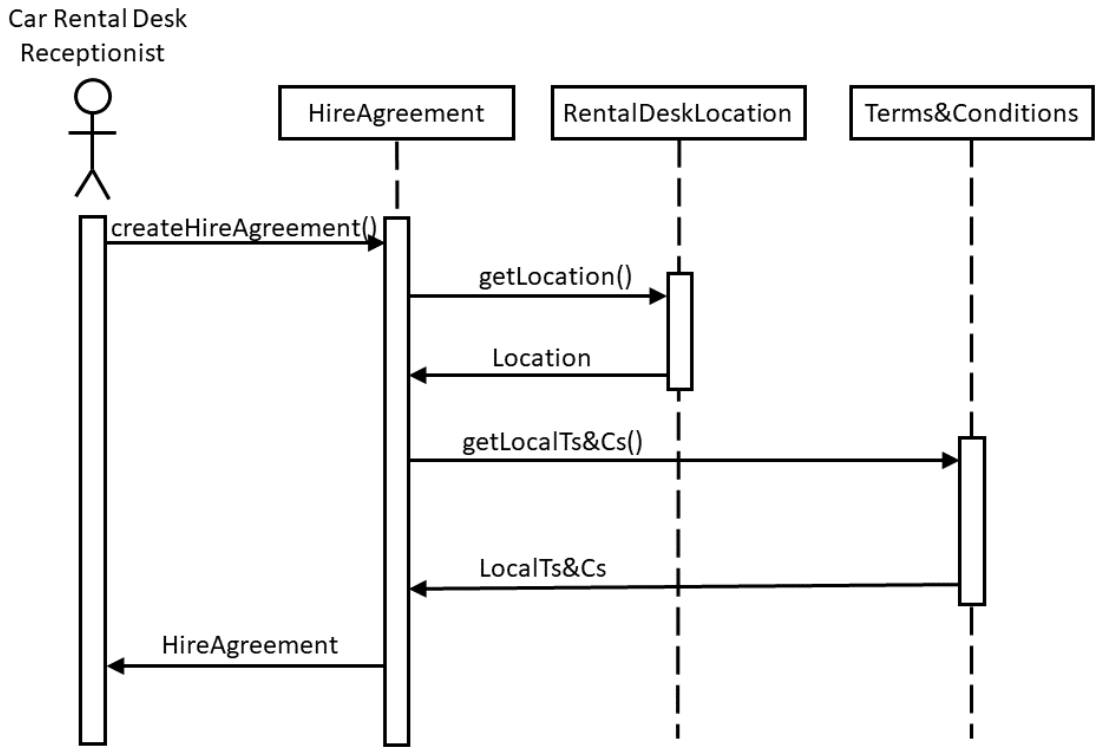
Object Sequence Diagrams

- Interactions between actors and system objects
- Usually correspond to a specific use case
- Model dynamic (run-time) interactions between components

University of Salford MANCHESTER

REDocelot

Object Sequence Diagrams



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_9

University of Salford MANCHESTER

REDocelot

Design Patterns

- Reusable descriptions of abstract design fragments
- Solve problems that re-occur in applications
- Provide best practice solutions
- Comprises description of problem and essence of solution
- Patterns enable design reuse

University of Salford MANCHESTER

REDocelot

Design Patterns

- Singleton Pattern
  - Ensures only one instance of the class is created
  - To instantiate…

```
SingletonObject so = SingletonObject.getInstance();
System.out.println( so.showMessage() );
```

University of Salford MANCHESTER

REDocelot

Design Patterns

```
public class SingletonObject {
                            /* Make the instance private and static */
                            private static SingletonObject instance = new SingletonObject();

                            /* Make the constructor private  */
                            private SingletonObject() {
                            }

                            /* Use this method to return the instantiated object */
                            public static SingletonObject getInstance() {
                                return instance;
                            }

                            public String showMessage() {
                                return "This is from the singleton";
                            }
                        }
```

University of Salford MANCHESTER

REDocelot

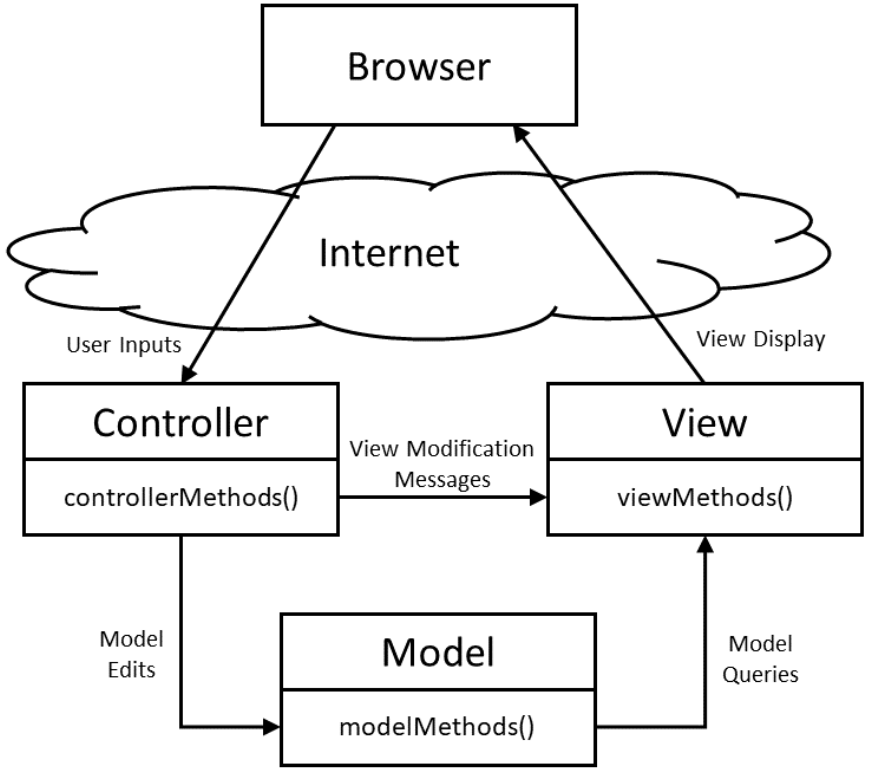
Design Patterns

- Model View Controller
  - Manage data (model) separately from the display (view) of information
  - Controller class manages interaction between views and model
  - Good separation of concerns
    - Can change underlying persistence implementation (model) without affecting rest of system
    - Can implement various client (views) without affecting model

University of Salford MANCHESTER

REDocelot

Design Patterns



J. M. Bass, Agile Software Engineering Skills,
https://doi.org/10.1007/978-3-031-05469-3_9

University of Salford MANCHESTER

REDocelot

Design Patterns

- Factory Pattern
  - Creates and returns objects of a particular type
  - Single point in the system for instantiating a family of similar objects
    - Simplifies maintenance in sophisticated systems
  - Simple example
    - Bass, J. (2022). *Julianbass/CarFactory* [Java]. https://github.com/julianbass/CarFactory (Original work published 2021)

University of Salford MANCHESTER

REDocelot

Technology Stack Selection

- Selecting project implementation technologies
  - Programming languages, frameworks, libraries, development environments and deployment environments
- In commercial sector, joining an existing project
  - You probably won't get much say
- In an academic, HackCamp or Hackathon environment,
  - Choosing the technology stack important decision

University of Salford MANCHESTER

REDocelot

Technology Stack Selection

- Consider how candidate technologies
  - Impact your chosen delivery model
    - installable desktop application, web app, cloud-hosted service etc.
  - Impact your functional and non-functional requirements
  - Contribute to achieving your architectural goals
  - Impact your product/project goals
- Identify the team member skill set
- Collect evidence and assess the Pros/Cons of the technologies your team members know

University of Salford MANCHESTER

REDocelot

Model-driven Engineering

- Automatically generate a complete or partial system from our system models
- Integrated development environments can generate skeleton classes and certain methods
  - Accessor and mutator methods
  - Constructor method skeletons
- Commercially available database design tools
  - Draw diagrams which are then automatically implemented as database tables

University of Salford MANCHESTER

REDocelot

Exercises

- Exercises 9.1 and 9.7 encourage creation of a learning journal
- Exercises 9.2 – 9.5 Cover class diagrams
- Exercise 9.6 Covers factory pattern implementation

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