Analysis, Design and Implementation

Topic 12:
Recap of Module

V1.0

Introduction

- We have reached the end of the content for this module. Our last lecture is recap of the module material to date.
- The lecture will contextualise each of the topics into one of three categories.
  - Analysis
  - Design
  - Implementation
- As discussed, these categories overlap to some degree.

V1.0

Analysis - 1

- Analysis is the process of assessing existing systems and processes, and converting them into some kind of representation.
  - We do this so that we can ensure we have a proper understanding of the problem domain.
- All analysis and all design is centred on a core principle.
  - It's about communicating with a particular audience.
- Analysis lets us communicate our understanding with all stakeholders.

V1.0

Analysis - 2

- We begin analysis from our problem statement. This is a written description of what our system should do, from the perspective of our clients.
- Problem statements are a starting point.
  - They are rarely useful in their original format.
- Analysis is the process of iteratively building our understanding of the problem and what kind of solution is required.
  - This progresses through interviews, investigations and examinations of existing documents.

Analysis - 3

- In OOAD we must always be mindful of our ***problem domain***.
  - This is the remit of the system we are to analyse and design.
- We express our analysis in the appropriate UML documents.
  - Use case diagrams
    - These show us the processes that actors within our problem domain will need available to them.
  - Activity diagrams
    - These let us model how existing processes work.

Analysis - 4

- Any process of analysis is an exercise in ***decomposition***.
  - We need to break a complex system down into more manageable units.
- Use case diagrams permit us to manage these decomposed units in terms of their relationships to actors in the system.
  - They give us an understanding of the scope of the system.

Design - 1

- Once we have an effective analysis, we can begin to extract design elements from it, through Natural Language Analysis or some other mechanism.
- Here, we attempt to give two views of the system.
  - The static view
    - Class diagrams
  - The dynamic view
    - Activity diagrams
    - Sequence diagrams

Design - 2

- The ***static view*** lets us design the architectural elements of the system.
  - Classes and how they are linked together.
- The ***dynamic view*** lets us design the time dependent elements.
  - The order and type of invocations of objects.
  - The flow of logic through a process.
- Each diagram is used in a different way to inform our implementation.

Case Study - 1

- We saw how analysis and design worked in our case study.
  - Out of necessity, we could explore only part of the model.
- However, we saw first hand several things:
  - How much irrelevance is in a description of a needed system.
  - How much relevance is omitted.
  - How to arrive at a representation of the static design of a system.
  - How to arrive at a representation of the dynamic design of a system.

Case Study - 2

- One of the problems associated with OOAD techniques in an academic environment is scale.
  - We can't show you real world problems because they are ***too*** complicated.
  - The isolated examples we show you don't really show why OOAD is important.
- The case study was somewhere between these two extremes.
  - Much simpler than real world problems.
  - More complicated than isolated examples.

Design Patterns - 1

- Our first draft of a design is only a starting point.
  - It helps us articulate how the system could fit together.
- We can greatly ease our implementation burdens with a solid understanding of design patterns.
  - These are well tested solutions to particular kinds of object-oriented problems.
- Implementing a design usually involves some measure of incorporating effective and appropriate patterns.

Design Patterns - 2

- There are many dozens of patterns, and we only touched on some of these.
  - The Model View Controller
  - The Factory
  - The Abstract Factory
  - The Observer
  - The Flyweight
  - The Strategy
  - The Facade

Design

- Object-orientation can be difficult because it is not always apparent when we have a good solution.
  - However, there are ways we can decide.
- We aim for systems with low amounts of loose coupling.
- We aim for systems with high amounts of cohesion.
- We must often trade-off between these two extremes.
- Software component design can also help with complex systems.

Implementation - 1

- We discussed implementation throughout the module, in the context of turning particular models into functioning code.
- We also looked at how to redesign our case study to incorporate patterns, and how to begin turning that redesign into code.
  - Particularly in relation to how the design patterns functioned.

Implementation - 2

- Assessing a software product is in many ways easier than assessing a design.
  - We can quantify elements of it.
- We can profile performance.
  - Benchmarking
- We can then optimise those parts of the system that have performance bottlenecks.
- We can also quantify the other measures of software quality that we discussed.

Recap of Module Topic 12 - 12.16

Implementation - 3

- The systems we write need to be rated against:
  - Functionality
  - Performance
  - Security
  - Reliability
  - Usability
  - Interoperability
  - Correctness

V1.0 © NCC Education Limited

Recap of Module Topic 12 - 12.17

Implementation - 4

- Our architectures need to be rated against:
  - Maintainability
  - Portability
  - Reusability
  - Testability

V1.0 © NCC Education Limited

Recap of Module Topic 12 - 12.18

Implementation - 5

- The last element of implementation is maintenance.
  - This takes up the majority of developer time.
- It falls into four rough categories:
  - Corrective maintenance
  - Adaptive maintenance
  - Perfective maintenance
  - Preventive maintenance
- All of these phases of maintenance tend to be prefaced by a process of refactoring.

V1.0 © NCC Education Limited

Conclusion

- ***Analysis*** is the process of understanding and representing a problem domain with the intention of communicating primarily to users.
- ***Design*** is the process of understanding and representing a proposed solution with the intention of communicating primarily to fellow developers.
- ***Implementation*** is the process of representing a design in code and then maintaining it.

Topic 12 – Recap of Module

Any Questions?