SWEN30006 Software Modelling and Design
Workshop 2: Domain Modelling and SSDs

School of Computing and Information Systems
University of Melbourne
Semester 2, 2020

Assessment: You *should* complete the exercises ***as a team***. To receive a workshop mark, you **need** to demonstrate your active participation during the workshop. Your tutor will observe your participation for you to be eligible to receive a mark for this workshop. The active participation include (but not limit to) present your solutions, help your teammates, and actively engage with the team. Attending the workshop with only passive participation will *not* be given a mark. See the Workshop Overview under Subject Information on the LMS for more details.

Requisite Knowledge and Tools

To complete this workshop you will need to be familiar with the following terms and concepts:

- Domain (conceptual) modelling
- UML notation for domain class diagrams and system sequence diagrams

Introduction

This workshop aims to familiarise you with the use of diagrams in developing **domain models**. In this week's work it's important to remember that we are focusing on **conceptual** models – we are looking at objects which exist in the problem domain, not software concepts or software objects.

> A domain model is a representation of real-world conceptual classes, not of software components. It is not a set of diagrams describing software classes, or software objects with responsibilities. – *Larman*

This is a distinction often missed by students in this subject and something that we will be stressing throughout this workshop.

Though we use UML to specify our domain models, we are developing a model that helps us *understand* the problem, not developing a design to *solve* the problem. This is just the first step before we then move to develop a solution, which we will do in later workshops. As an example, consider a point of sales system, a partial domain model for which is shown in Figure 1. Note that this model represents the *real world* conceptual classes, not software concepts.

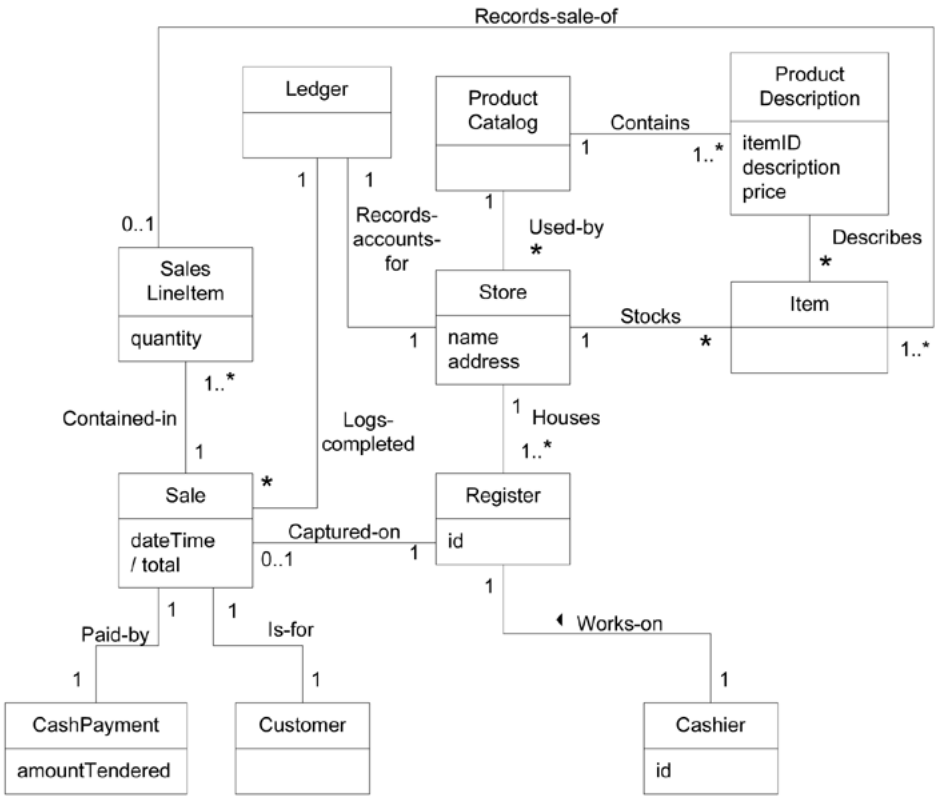

Figure 1: A partial domain class diagram for a point of sale system

Today we will also be doing some dynamic modelling using sequence diagrams. Static modelling tools like class diagrams do not provide the whole picture for developing a software system. Even so, dynamic modelling is often overlooked, despite it being able to provide valuable insight into the structure of classes.

Sequence diagrams are designed to show a series of events in their correct order. Each object involved has a *lifeline*, and objects communicate with each other through *message passing*. Sequence diagrams are best used to represent a single interaction between multiple components in a software system. That is, they should represent **one** transaction. You can use guards (i.e., conditional statements), loops, asynchronous invocation, and self calls in a sequence diagram. Figure 2 provides an example of a sequence diagram written for a specific use case. If you need some help with the syntax, this reference can be helpful.

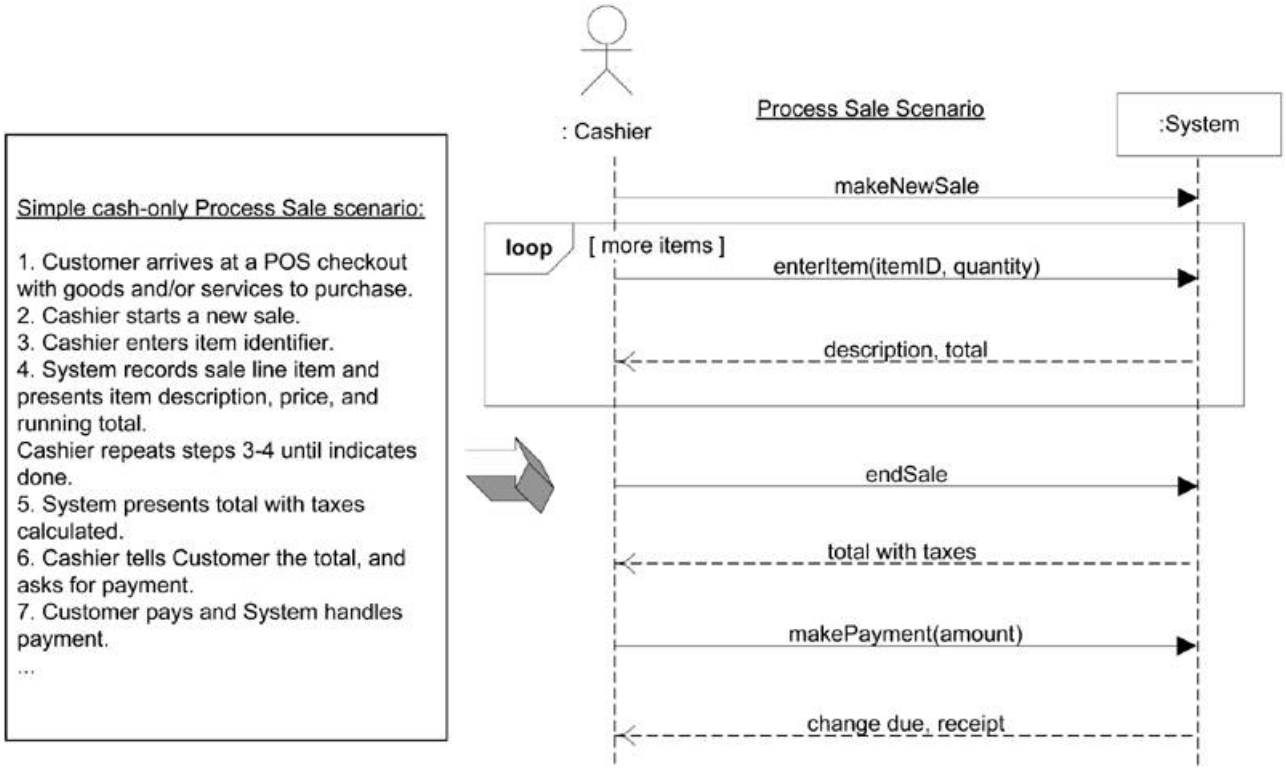

Figure 2: An example system sequence diagram, extracted from a use case

Diagramming Tools

There are a wide variety of tools available to you that will enable you to perform UML modelling on a computer, some that will even autogenerate code and other forms of documentation from the diagrams you create. These tools vary wildly in quality, availability, and cost, especially across operating systems.

In this subject, we opted to use an online tool that allow real-time collaboration. Hence, we recommend to use Draw.IO as our supported diagramming tool. Draw.IO is an online tool allowing for the creation of a wide variety of diagrams, including a number of UML models. It is free to use, and will enable storage of documents for later editing without the need to sign up for an account. Further, being web-based, Draw IO is compatible on Windows, OSX, and Linux machines, so is an ideal tool for use in your group projects.

You are welcome to use other tools that allow you to draw diagram collaboratively. For example, AWW App is a web-based whiteboard that allows you hand draw a diagram collaboratively. However, the tool does not support UML notations.

Part 1 Static Domain Modelling

Note: All tasks in Part 1 and Part 2 should be done as **a team**.

Q1.1 Interpreting a Domain Model

It is important that we appropriately scope our domains models. We do not want to define software concepts in the domain model as they are not part of the problem domain. To help you consolidate this idea, your next task is to identify the components that do not belong in the domain model presented in Figure 3. This figure shows a domain model for a subset of a new video streaming service, NetFlicks.

Identify which elements do not belong in the model and why. Write down your answer – your tutor will ask you to explain your reasoning.

Tip: You may use one of the real-time collaborative tools mentioned above. You should also share your screen to allow your tutor observe your active participation.

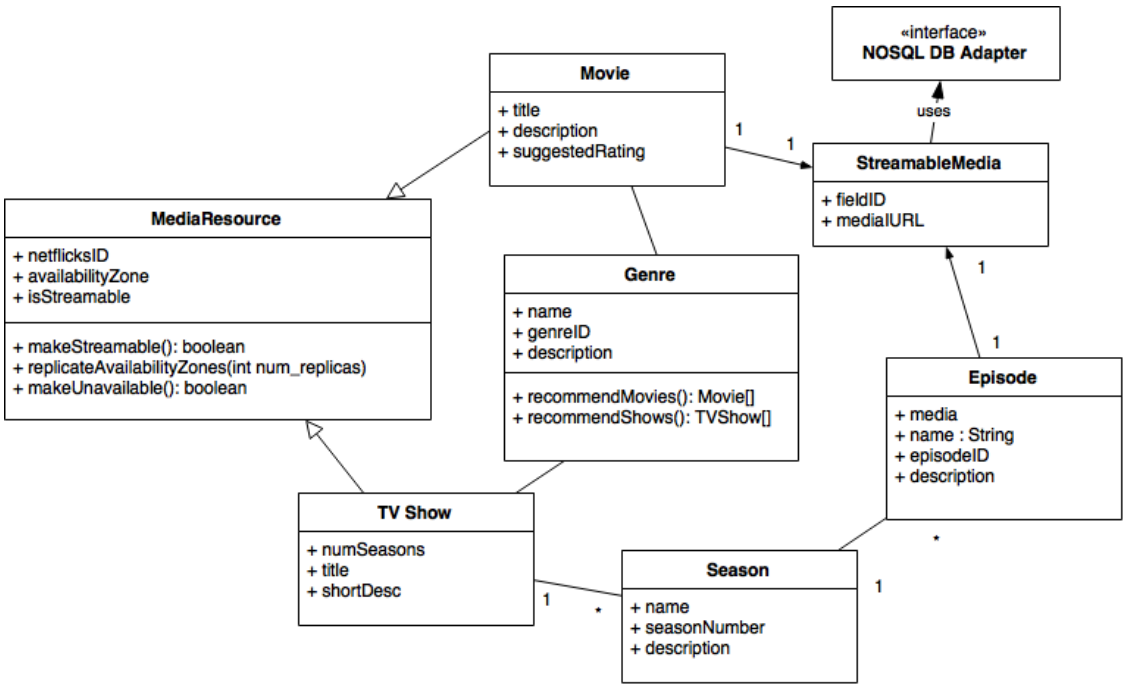

Figure 3: An incorrect domain model for a streaming service

Q1.2 Creating a Static Domain Model

Now that you've identified the problems in an existing domain model, let's try writing one from scratch. For this task, we'll look at a domain you should all be very familiar with: a student learning management system. When you're writing a domain model it's important to work with only one or two use cases at a time. This will help to prevent your diagrams growing too quickly, becoming overly complex, and being hard to read. The two use cases we're going to focus on are:

1. An instructor creating an assignment, and
2. Students uploading an assignment submission.

The interactions for these use cases are defined as follows:

Creating an Assignment

1. The instructor logs into the LMS
2. The instructor creates a new assessment item
   (a) The instructor uploads the assessment instructions as a PDF
   (b) The instructor chooses a method of assessment upload (either TurnItIn or ZIP file upload)
3. The instructor makes the assessment item available to students

Uploading an Assignment Submission

1. The student logs in to the LMS
2. The student selects the appropriate assessment item
3. The student attaches their submission to the assessment item
4. The student confirms the submission, and optionally adds a note
5. The student is emailed a receipt of the transaction

Complete the following tasks based on the two use cases described above:

1. Draw a use case diagram in Draw.IO for the described system
2. List candidate conceptual classes using noun phrase identification
3. Draw a domain model including these classes on Draw.IO
4. Add associations (including multiplicity and naming) between these domain classes for 'need-to-know' relationships
5. Add any required attributes

Remember that we are modelling the domain – your model should only contain conceptual classes, not software classes.

Part 2 Dynamic Domain Modelling

Tip: You may use one of the real-time collaborative tools mentioned above. You should also share your screen to allow your tutor observe your active participation.

Create **two** system sequence diagrams – one for each of the use cases described above. Your system sequence diagrams should correspond to your domain class diagram (e.g., same class names etc.).