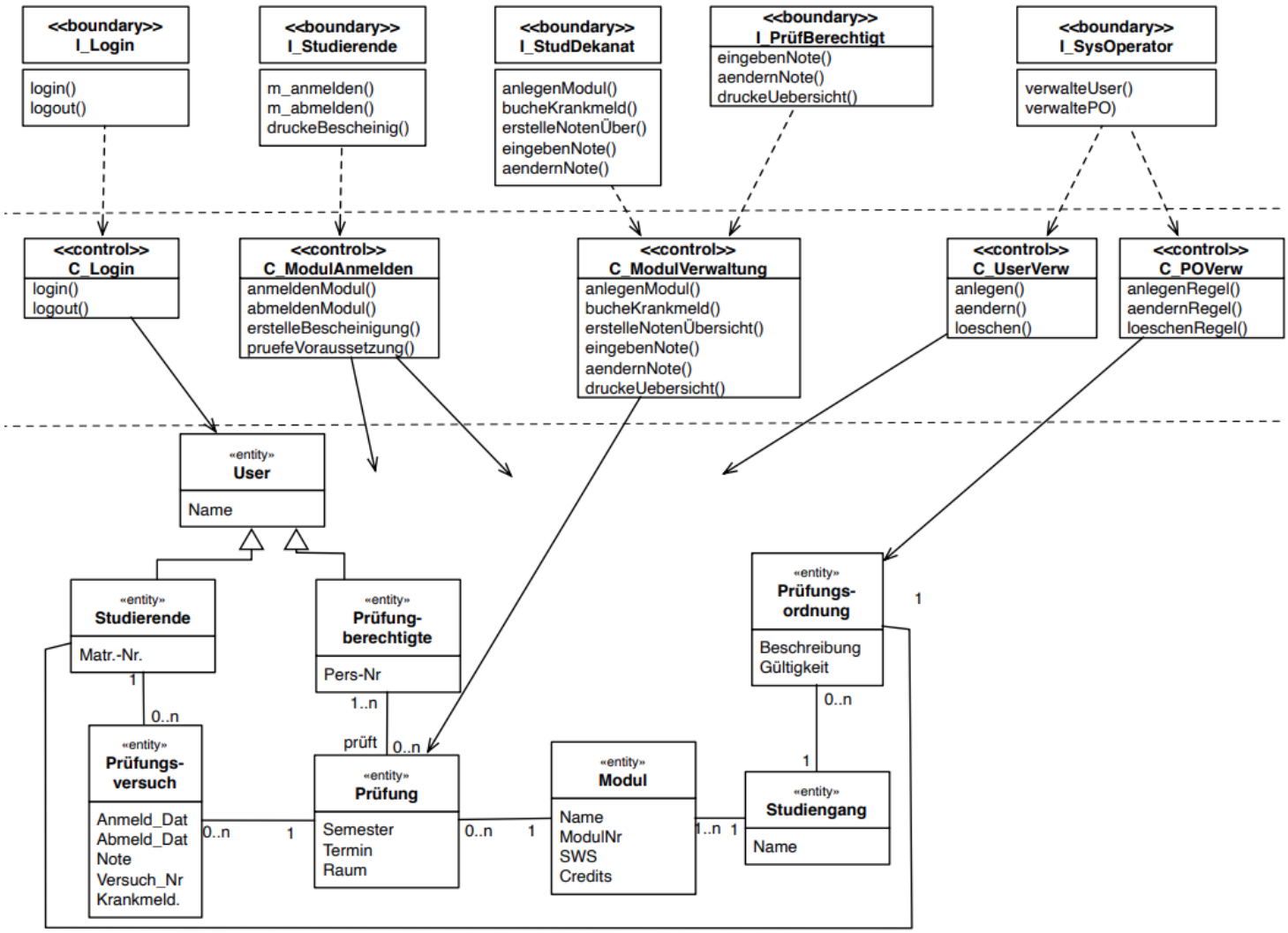
Aufgabe 5.1: Analyse des Modulbelegungsmanagementsystems ModMS – Pflichtaufgabe

Nutzen Sie die informelle Beschreibung aus Übungsblatt 4 und das Use Case Diagramm, um ein **fachliches Klassendiagramm** zu erstellen. Bestimmen Sie hierzu die **Entity-, Control**- und **Boundary**-Klassen!

1. Modellieren Sie zunächst die **Entity**-Klassen und deren Beziehungen untereinander (ggf. als Erweiterung des Domänenmodells aus Übungsblatt 4). Geben Sie pro Klasse nur die wichtigsten Attribute an (nicht zu viele). Prüfen Sie für welche Beziehungen Aggregation, Komposition oder Vererbung sinnvoll sind.
2. Bestimmen Sie dann die zentralen **Boundary**-Klassen mit den Operationen, die das System den Akteuren zur Verfügung stellen sollte.
3. Überlegen Sie wie viele **Control**-Klassen sinnvoll sind, führen Sie deren wichtigsten Operationen auf und modellieren Sie die Beziehungen zu den anderen Klassentypen.



- Entity-Klassen:
    - Typinformation wie Student oder Prüfungsberechtige niemals als Attribut abspeichern, da falsche Attribute oder Beziehungen entstehen
    - Persistente Daten zu den Akteuren wie Systemoperator oder Studiendekanat sind für die Use Cases nicht notwendig, deshalb nicht mit modellieren. Natürlich müssen die am Ende vorhanden sein, aber die sind nicht systemkritisch. Das sieht man daran, dass die keine Beziehungen zu anderen Entitäten haben
    - Modul und Prüfungsberechtigte Beziehung fehlt
    - Persistente Daten nicht redundant abspeichern, da Konsistenzerhaltung schwieriger wie z.B. extra Entity Studienbescheinigung ist unnötig
    - Nur fachliche Zusammenhänge abspeichern. Systemoperator greift auf Prüfungsordnung zu, aber Beziehung muss nicht abgespeichert werden
    - Studiendekanat von Prüfungsberechtige erben macht keinen Sinn, weil Vererbung eine Datenabhängigkeit ist. Gemeint ist, dass Studiendekanat alles kann was Prüfungsberechtige kann und das wird durch Boundary-Klassen realisiert

- Boundary und Control-Klassen sollen unabhängig sein
- Von Boundary zu Control gestrichelt, damit klar wird, dass Beziehung nicht gespeichert wird, sondern über Parameter
- Pfeile von Control zu Entity werden meistens weggelassen, sonst zu unübersichtlich

- Control-Klassen:
    - Sicht der Akteure einnehmen um die Control-Klasen zu erstellen
    - Beziehungen zu Entity-Klassen hinterfragen wie sinnvoll. Z.B. Braucht C_Modul Daten von Student, eher als Parameterübergabe temporär
    - Nicht alle Beziehungen zu Entity-Klassen darstellen, werden sonst zu viele

- Boundary-Klassen
    - Ist eine API, die nach Außen zur Verfügung gestellt wird. Die Funktionalität ist bereits festgelegt unabhängig von Control und soll nicht verändert werden. Deshalb zuerst Boundary und danach interne Umsetzung konkretisieren
    - GUI ruft Boundary auf
    - GUI Keine Methoden anbieten, die vom System automatisiert ausgeführt werden sollen. Wie z.B. berechtigungPrüfen, diese wird intern von anderen C_Methonden verwendet
    - Boundary nicht eine Klasse, dann unübersichtlich
    - Faustregel: Boundaries von den Akteuren ausgehen und Boundary-Klassen Akteure repräsentieren

- In der Analysephase wird früh das meiste modelliert, weil nach dem UPS kritische Use Cases zuerst entworfen werden sollen. Diese machen den Hauptteil des Systems aus.