**Analysis:**

The act of studying a system by seeing how it works, what environments it works in, its pros and cons, etc.

**Design:**

The reverse action of analysis. Instead of studying the conditions of an existing system, you make a system from a set of conditions.

**Implementation**:

The practical shit needed to make the design a reality.

Note: None of this shit needs to be done in a specific order. Be flexible and apply it in a way that fits with your case.

**Class-Object relations:**

Class is the blueprint, objects are lil’ shitters made from the blueprint.

Why we learning this shit again? Because famalams from other educations have not learned this trash yet.

**Granularity:**

The idea of whether objects should be used to track the state of individual mcguffins, or if we should just keep track of a collection of mcguffins. Ex. Assume we have piles of shit like sand and rocks. Do we create objects for each individual grain of sand and rock, or do we just make an object that states how many there are.

**Application-Problem domain:**

Assume there is some sort of problem, we will call this our problem domain. This could be counting dead bodies, sorting through documents, etc. Then there’s the dude who has to do this shit, he is the application domain. He no wants to do manual labour, so he creates an application that does it for him. Said application is the model. A model is part of a system (fx. Being a part of a website, or being an option of a program).

**Factor:**

F – Functions of a system

A – Admin functions

C – Conditions, environment the system is used in.

T – Technology for developing and running the system.

O – Objects in the problem domain.

R – What is the system responsible for? Btw no one knows wtf responsibility actually means. Just assume it is who does what work (Shared between user and software).

**Task 2:**

Problem: Bådene, lejerne.

Application: Udlejeren, Lejeren.

Model (Data): Bådene, lejerne.

Functions: Booking, indtjæk & udtjækning (Hvis ikke auto).

Interfaces: GUI, hviser hvilke både der er til leje og hvornår. For udlejeren er der også overblik over hvem der har lånt hvilke både.

**Task 3**:

Problem domain: Luftrum, Fly, helikoptere, osv.

Air traffic control information: Aircrafts position and status, kommer fra en combination af flyet selv og radare. Status af landingsbaner, som gives af systemet og DMI (Vejr). Udover det, schedule of flights, altså hvornår hvilke fly skal flyve.

Controller: Får informationerne fra systemet, bruges til at kommunikere med piloter for at styre trafikken.

**Task 4:**

S1: The way I think it currently works is as specified in task 3, where the problem domain consists of the flights positions, their remaining fuel, status of the physical parts, etc. Of this information, position can be detected using radars, while fuel needs to be communicated from the flyvemaskine itself. Physical status needs to be communicated orally by the pilot. Then there are the landing lanes, which need to be scheduled ahead of time to prevent flights from crashing, and what the weather is for those lanes. Scheduling can be done by the system, as it is better at keeping track of flights and timetables. Weather is communicated by a third party (DMI). All this is gathered by controllers to coordinate traffic.

S2: Function: Kommunikation med andre fly, kollisions avoidance.

Application: Piloter og systemet.

Conditions: Pilots can communicate with each other.

Technology: Computers, radio communication, radar.

Objects in problem: Fly, landingsbaner.

Responsibility: Udregne informationer og vise dem til piloterne.

S3: Function: Collision detection, path navigation (Under which weather prediction plays a role), landing coordination.

Application: Ingen interergerer med systemet.

Conditions: Fly kan kommunikere med systemet.

Technology: Computers, radio communication, radar

Objects in problem: Fly, landingsbaner.

Responsibility: Collision detection, path navigation (Under which weather prediction plays a role), landing coordination, and scheduling of flights.