1 Begrepp

Tydliggör följande begrepp

1.1 SRS

1.1.1 Software requirements specification

SRS är ett dokument som beskriver vad en mjukvara kommer göra och hur den förväntas att prestera. Den beskriver också den funktionalitet som produkten kräver för att uppfylla investerares och användares krav.

1.1.2 Kravspecifikation

En kravspecifikation är processen där en beställare och utvecklare gemensamt förstår syftet och målet med projektet, samt förhoppningsvis ge samtliga parter en idé om hur slutprodukten skall fungera och i vissa fall även hur den ska se ut.

1.1.3 Beskriv skillnaden mellan funktionellt krav och icke-funktionellt krav

Funktionella krav beskriver hur systemet interagerar med användarna.
- En kund bokar en biljett på en webbportal och skall få ett email med en bokningsbekräftelse.

Icke-funktionella krav är krav som inte beskriver hur systemet interagerar med användarna.
- Prestandakrav så som svarstider, samtidiga användare, genomströmning etcetera

1.1.4 Räkna upp vilka diagramtyper som är användbara för SRS

Use case diagram, UML, ER

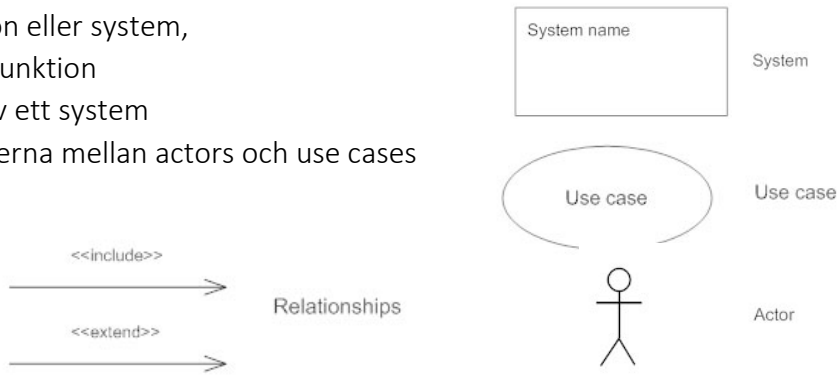
1.2 Prototyp.

En prototyp är ett skal av en applikation. Det är ofta skapat i ett tidigt stadie i utvecklingscykeln, och ger en bra insyn i hur saker känns, ser ut och fungerar från ett flödesperspektiv i en applikation.

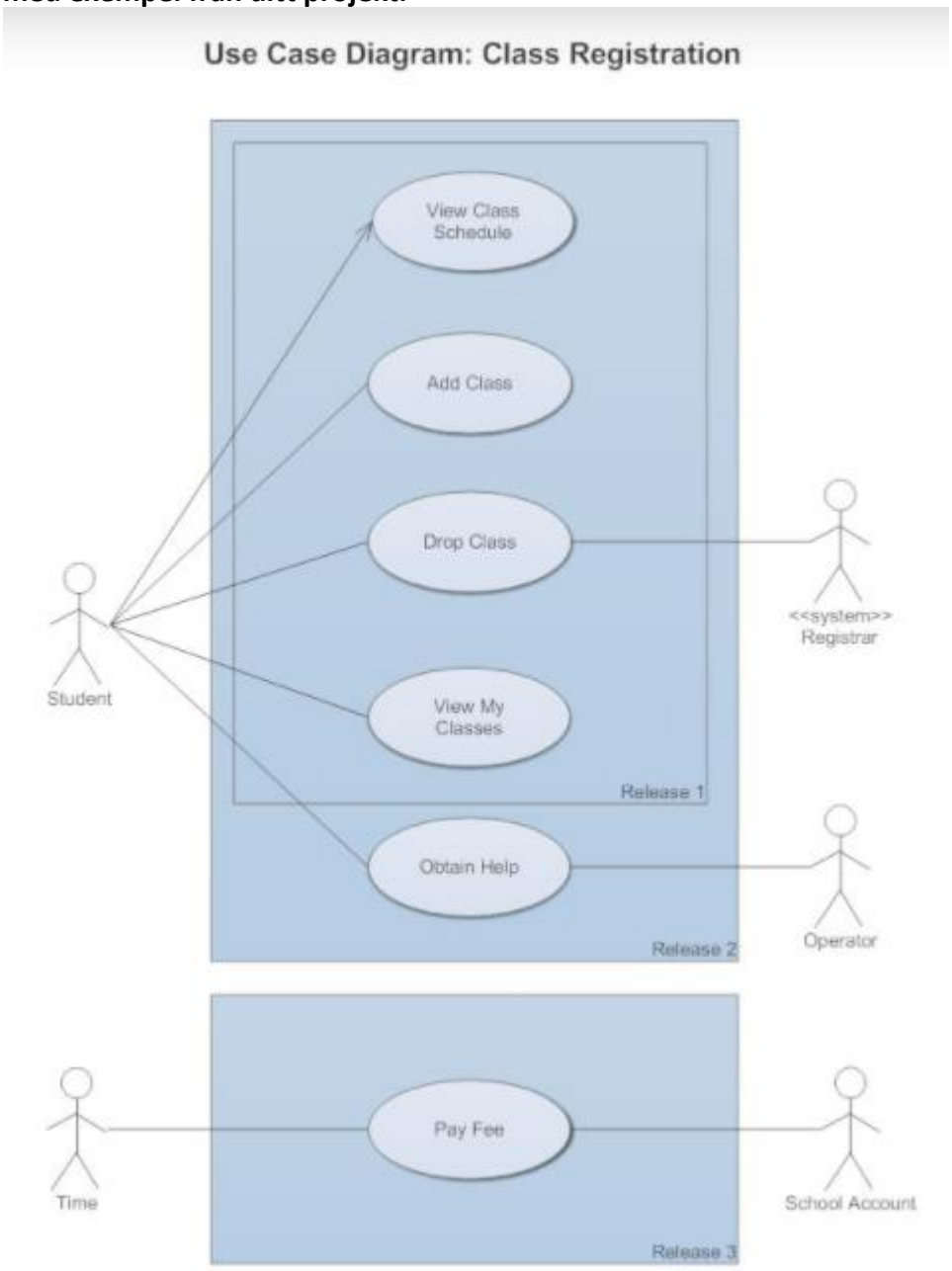
1.3 UCD

1.3.1 Use case diagrams, illustrera de olika UMLkomponenter som är användbara i Use case diagram.

System är din applikation eller system,
Use Case är systemets funktion
Actors är användarna av ett system
Relationship är relationerna mellan actors och use cases



1.3.2 **Beskriv hur ett välformulerat Use case diagram bör vara komponerat, illustrera med exempel från ditt projekt.**



1.4 UPM

1.4.1 Unified Process, räkna upp de aktiviteter som UPM innehåller.

Inception phase är den minsta fasen i ett projekt, och skall idealt sett vara ganska kort. Om fasen är lång så är det ofta en indikation på att man har en onödigt lång specifikation.
Vanliga mål i denna fas är:
- Fastställa projektet
- Förbereda ett preliminärt projektschema och kostnadsförslag.
- Genomförbarhet
- Köpa eller utveckla?

Elaboration phase är den fas där projektteamet är föväntade att fånga majoriteten av system kraven. Dock så är de primära målen i denna fas att adressera riskfaktorer och fastställa samt validera system arkitekturen.
Vanliga processer som sker i denna fas är:
- skapandet av use case diagrams, conceptual diagrams, package diagrams.

Construction phase är den största fasen i projektet. I denna fas så skall det resterande, av det som påbörjats i Elaboration phase byggas klart. Systemfunktioner är implementerade i korta iterationer och varje iteration bör resultera i exekverbara releaser av mjukvaran.
Den sista iterationen i Construction phase leverar mjukvaran redo att deployas i Transistion phase.

Transition phase är den sista fasen i ett projekt. I denna fas så så överlämnas systemet till kunden. Feedback från release kan resultera i finjusteringar.
Inkluderar även *system conversion* och *user training*?

1.4.2 Beskriv utförligt hur de genomförts i ditt projekt.

Se uppsats

1.5 AD

1.5.1 Agile development, beskriv vad som utmärker agil utveckling.

Iterativt utvecklande. Det vill säga att man sätter upp mål, jobbar mot dem, går tillbaka och kollar på dem och sätter upp nya mål.

1.5.2 Beskriv utförligt hur ni tillämpar agil utveckling

SCRUM

2 Principer

Beskriv följande principer i några stycken

2.1 Single responsibility principle

Single responsibility principle innebär att varje class I en application ska ha ansvar för enbart en del av programmets funktionalitet och om något ska ändras, så ska det bara finnas en anledning för classen att ändras.
Anledningen till att detta är önskvärt är för att minimera potentiella ändringar som kan krävas att göras med tiden.

2.2 Open closed principle

Open closed principle är en riktlinje för hur man som utvecklare ska sträva efter att skapa sina classer, interfaces och dylikt, så att de med tiden inte kräver någon ytterligare modifikation, utan går att extenda till nya classer.
Detta är önskvärt då man som utvecklare inte vill behöva gå tillbaka och ändra på redan skapad kod, utan istället använda sin skapade kod vidare och skapa nya classer som ärver dess egenskaper.

2.3 Substitution principle (Liskov substitution principle)

Liskovs substitution principle definierar att ett objekt av en supersclass ska kunna bli utbyten av en av dess subclasser utan att applikationen går sönder.
Syftet till varför denna princip är önskvärd att följa är för att det möjliggör att man skriver robust, underhållbar och återanvändbara komponenter.