Cavide Gunay & Derya Uysal

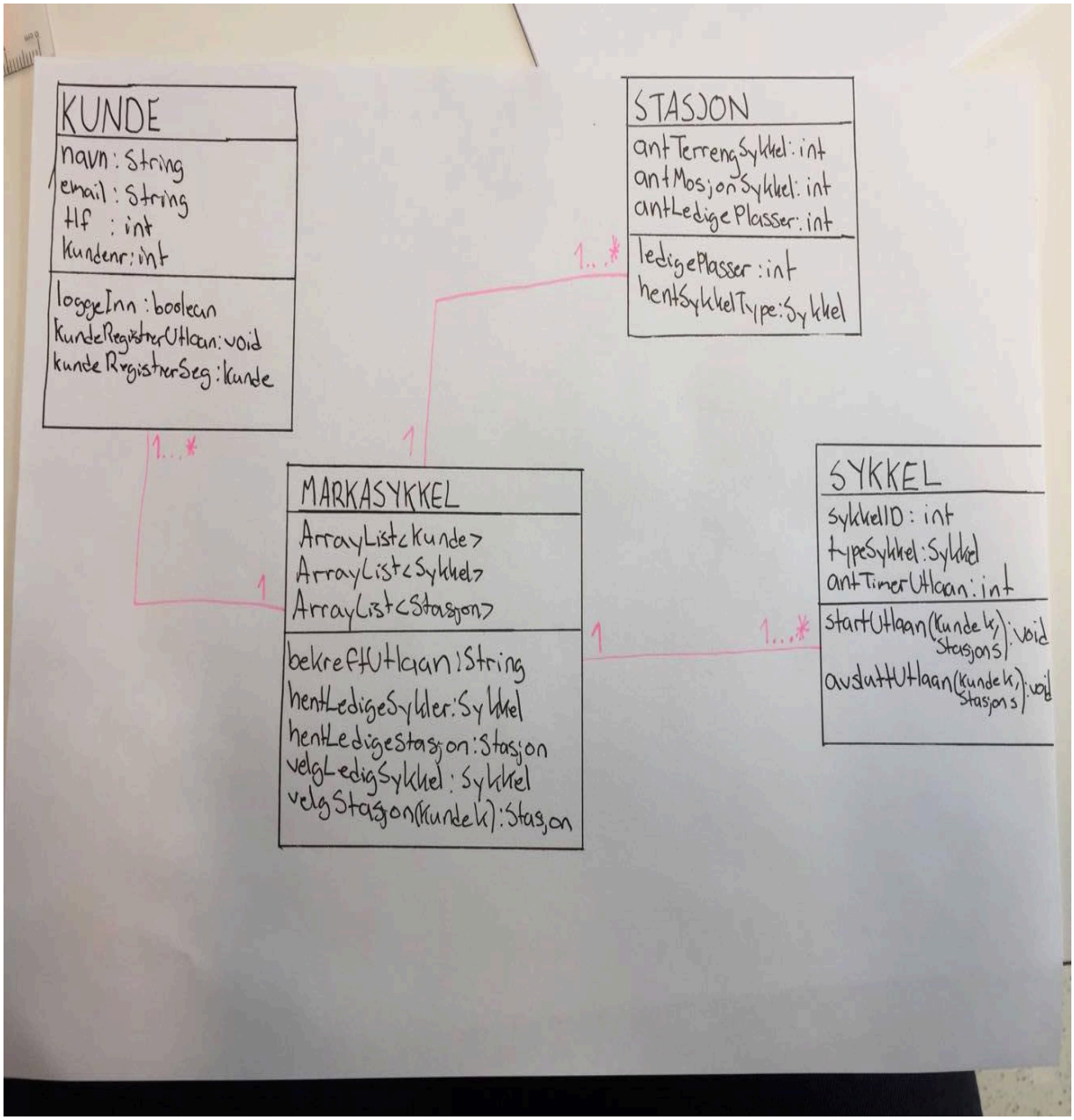
OBLIGATORISK OPPGAVE 2

INF1050 – SYSTEMUTVIKLING

Universitet i Oslo

OPPGAVE 1: Klassediagram

Lag et klassediagram for systemet. Ta med assosiasjoner mellom klassene, og metoder og attributter til hver klasse.



OPPGAVE 2 : Sekvensdiagram

A) Lag en tekstlig bekrivelse for brukstilfellet "Registrer utlån". Ha med aktører, eventuelle pre- og postbetingelser, hovedflyt og alternativ flyt der kunden ikke finnes i systemet fra før.

Navn: Registrer utlån
Aktør: Kunde
Pre-betingelser: Ingen
Post-betingelser: Utlån blir registrert

Hovedflyt:

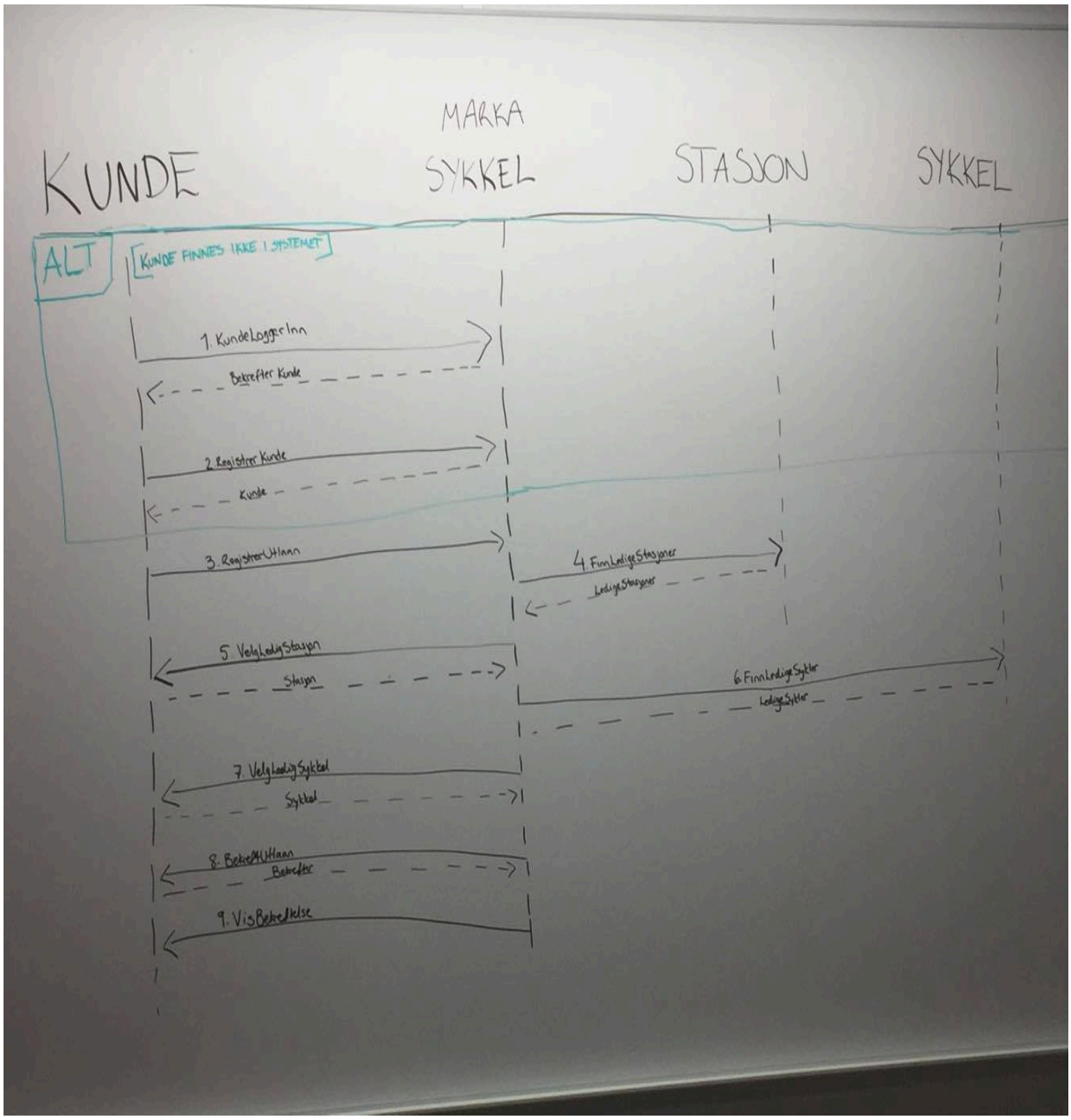
1. Kunden logger inn på nettsiden med brukernavn og passord
2. Kunde velger "registrer utlån"
3. Systemet ber kunden velge ønsket sykkel-stasjon
4. Kunde velger sykkelstasjonen
5. Systemet ber kunden velge ønsket sykkeltype
6. Kunde velger ønsket sykkeltype
7. Kunde fullfører søknad
8. Systemet registrerer utlån ved å sende en kvittering

Alternativ flyt 1:

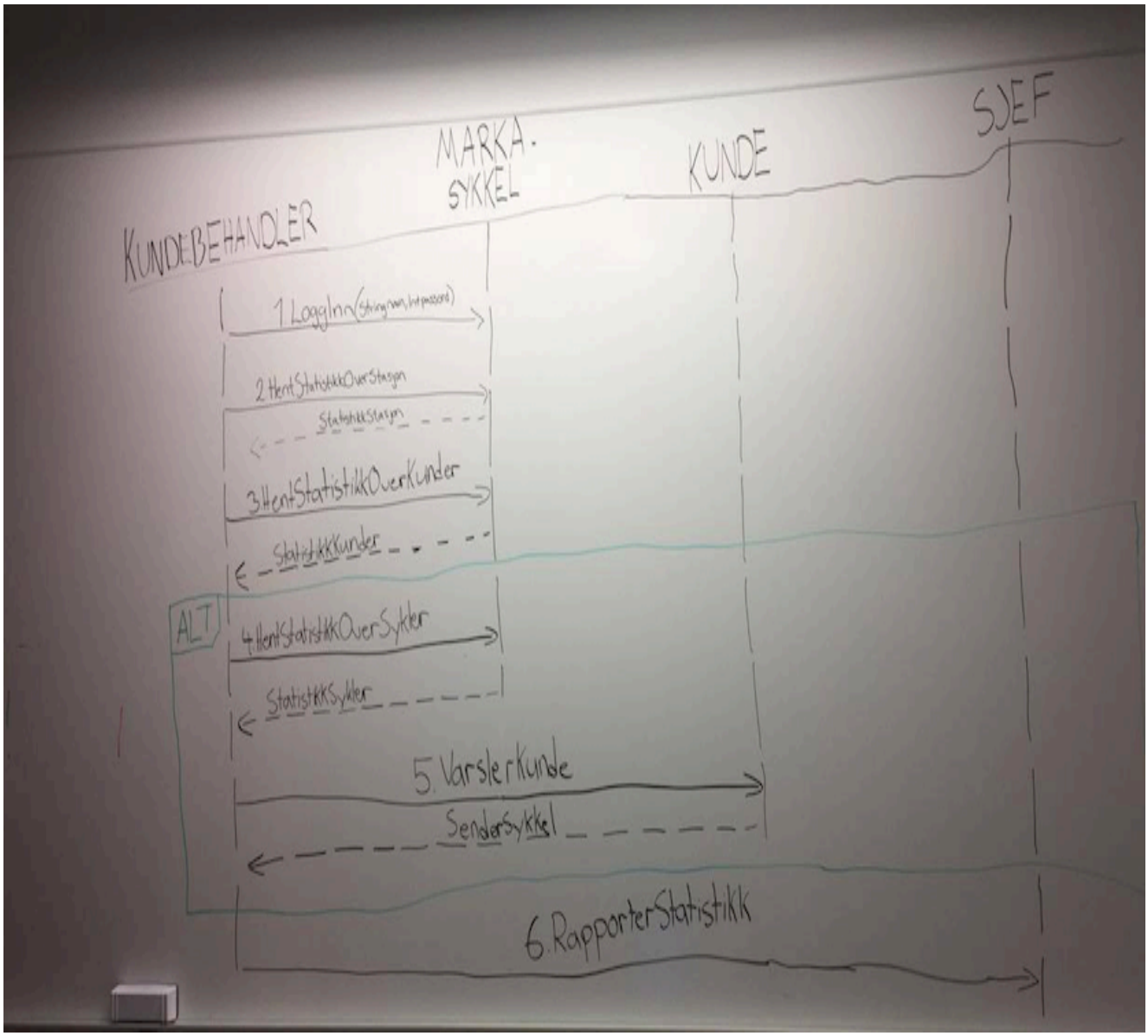
1.1 Kunde er ikke registrert i systemet
1.1 Kunde klikker inn på den relevante nettsiden
1.2 Kunde velger "ny kunde" på nettsiden
1.3 Systemet ber brukeren fylle inn personalia
1.4 Kunde fyller ut det den blir bedt om
1.5 Systemet registrerer brukeren
1.6 Systemet sender en bekreftelse på mail for ny kundeopprettelse
1.7 Kunde kan nå registrere utlån av sykkel

B) Lag et sekvensdiagram for brukstilfellet "Registrer utlån". Bruk de nødvendige klassene og metodene fra oppgave 1. Ta med alternativ flyt der kunden ikke finnes i systemet fra før.

Vi har valgt å beholde tallene på de ulike hendelsene. Dette er fordi lignende nummerering blir gitt som et eksempel i forelesningen den 21. Februar 2017, om sekvensdiagrammet "reserver bil".



C) Lag et sekvensdiagram for brukstilfellet "Se oversikt over sykler som ikke er levert i tide".



OPPGAVE 3: Uavhengig av hva dere har svart tidligere; anta at utviklerne bestemmer seg for en plandrevet utviklingsprosess.

A. Identifiser de overordnede aktivitetene som inngår i utviklingen av systemet for markasykler. Beskriv disse. Ha med minst 8 aktiviteter.

1. Identifisere krav

Ved identifisering av krav er det flere aktiviteter utviklere bør ta hensyn til. Det er nødvendig å kunne forstå domenet godt nok, for å utvikle et godt system. Slik vil misforståelser mellom utviklere og for eksempel interessenter minimeres.

Et felles domene-språk bør brukes, slik at alle aktører har samme forståelse av begreper og blant annet koder som brukes. Dette innebærer å skape en felles forståelse av systemet.

Det neste steget en utvikler bør ta hensyn til er å forstå hva interessentene ønsker og ser etter i systemet deres. Utviklere identifiserer krav for å løse konflikter for interessentene, og dette bidrar også til et startpunkt for prosjektet. Et problem som kan oppstå her er at brukere ofte ikke vet hva de ønsker, eller at meninger endres underveis. Dette vil være problematisk ved en plandrevet utvikling fordi det er lite rom for endringer etter at en aktivitet er endt.

I vår oppgave vil utviklerne av markasykkel ta utgangspunkt i hva kunden og andre interessenter som benytter seg av denne type betalingsløsning ønsker. Det er mulig å identifisere krav ved å ha brukerundersøkelser eller intervjuer. Slik samler utviklere inn data som kan bearbeides videre til forslag for systemutvikling.

2. Design

Design angir hvordan systemet skal fungere og bidrar til å strukturere kravene forståelig både for utviklere og interessenter. Denne aktiviteten er en kreativ aktivitet som bidrar til å få en oversikt over programvarekomponenter og sammenhengen mellom dem, basert på kravspesifikasjonen fra kunden.

Det å designe godt innebærer at det er gjenbrukbar, utvidbar og forståelig. Det er veldig vanlig innen systemutvikling å ta utgangspunktet i tidligere prosjekter (reuse), derfor er det

nødvendig at utformingen blir utført godt. Til tross for at det finnes ulike designprosesser, innebærer de felles aktiviteter:

- Forstå og definere systemets kontekst og eksterne interaksjoner med systemet
- Designe systemarkitekturen
- Identifisere nøkkel objektene i systemet
- Utvikle designmodeller
- Spesifisere grensesnitt

Ved design er det viktig at utviklerne av markasykkel tar hensyn til sine interessenter, slik at interessentene forstår integrasjonen mellom de ulike komponentene som skal bli tatt i bruk. To viktige faktorer utviklerne bør tenke på er at designet av modellene må være *oversiktlig* og tar i bruk det *universelle språket.*

3. Implementasjon

Implementasjon handler om å kunne videreføre det man har utviklet i design stadiet. Dette innebærer å kunne implementere/gjøre om et designmodell om til programmering. Det er veldig viktig å skrive en kode som er god og lett forståelig, fordi det er vanlig å bruke samme kode om igjen i systemutvikling (reuse). De fleste store prosjekter pleier å ta utgangspunkt i tidligere prosjekter som er lignende den som skal bli utviklet. Det er veldig mange programmeringsspråk man kan velge i mellom, de mest brukte er C++, Java og Python.

4. Integrasjon og systemtesting

Systemtesting handler om å oppdage feil før systemet blir tatt i bruk, og at systemet gjør det systemet er ment å gjøre. Et annet viktig punkt som kan trekkes inn er at testing bare viser feil under kjøring av testing, ikke at systemet er feilfritt. Denne aktivitet innebærer også en verifikasjons- og valideringsprosess.

Verifisering handler om produktet blir utviklet riktig, og om det stemmer med kravspesifikasjonen. Verifikasjonsprosess har komponentene/delsystemene i fokus. Validering handler om det riktige produktet blir bygget, og om programvaren utfører det ønskelige arbeidet for brukeren. I denne prosessen er det mer fokus på hele systemet og ikke bare en del av systemet som det er i verifikasjonsprosessen.

For utvikling av markasykkel vil det være nødvendig å kunne teste systemet før det bli tatt i bruk. I dette tilfellet ville det være lurt å utføre en stresstest av systemet, med tanke på at mange kunder kommer til å bruke dette systemet samtidig.

5. Drift og vedlikehold

Denne aktiviteten går ut på at utviklere lager et system/produkt o.l. med lang levetid. Dette innebærer at systemet skal kunne brukes om igjen (reuse) og at det ikke skal ha en ”kort holdbarhet”. Samfunnet endrer seg og nye brukerkrav blir aktuelle. Med dette mener vi det er viktig å utvikle et system som tilpasser nye kriterier, og at man eventuelt kan bytte ut én del av systemet og fornye dette, enn å måtte utvikle et helt nytt system fra scratch.

Teknisk vedlikehold vil oftest utføres etter en feil i systemet. Noen ganger er det ikke like enkelt å forhindre slike fiaskoer, derfor kan vedlikehold gjennomføres regelmessig før graden av problemet blir større. Hvis et problem oppstår, er det viktig å finne ut hva årsaken er slik at man kan innføre nødvendige tiltak eller få med seg hvilke andre skader som også kan ha skjedd ved følge av feilen.

Å vedlikeholde et system kan være kostbart, og omfanget av skaden avgjør hvor store summer det er snakk om. Ved regelmessig vedlikehold kan man spare penger på lang sikt og forlenge levetiden til systemet ved å hindre feil før det oppstår.

6. Estimering

Det er viktig å kunne vite cirka hvor mye utviklingen av løsningen vil koste generelt. Da vil den ansatte med mer ekspertise i bransjen foreta en beregning på hvor mye utviklingskostnadene vil ende på. Her kan prosjektlederen som foretar estimeringen velge å gi et estimat som er lik tidligere arbeid ham har utført, eller så kan ham velge å sammenligne prosjektet fra andre virksomheters tall dersom de er tilgjengelige.

Feilestimering kan oppstå ved komplekse prosjekter, uklare krav eller ved at en tar optimistiske valg m.m. Derfor er det mulig å benytte seg av estimeringsmodeller som COCOMO eller Planning Poker. Slik kan noen feil oppdages og reduseres før prosjektstart.

For utviklingen av marka sykkel vil det være nødvendig å kunne angi nøyaktige estimater, for å ikke kunne ende opp med store risikoer for feil, eller nedleggelse av prosjekt på grunn av

lite midler til videre drift. Det er avhengig at en med ekspertise innen feltet foretar beslutningene, slik at prosjektet er i "gode hender".

7. Prototyping

Vi anser prototyping som én av de viktige aktivitetene innenfor utvikling av systemet. Ved å lage en prototype allerede i startsfasen vil man danne et "inntrykk" over hvordan man tenker det endelige resultatet vil være. Det gjør kommunikasjon mellom bruker og utvikler enklere, og man kan komme med forslag til endringer, som ellers ville vært vanskelig å endre ved prosjektslutt. En prototype gir også svar på mange spørsmål, deriblant hvilke funksjoner produktet skal ha, hvilke språk det skal være programmert i, hvilke designtyper man går for og liknende.

I dette prosjektet kan utviklerne lage en prototype over hvordan system-designet skal se ut for markasykkel. De kan velge å danne et såkalt "design-team", men ikke nødvendigvis. Enten begynner man fra scratch, eller så fortsetter man på en allerede fungerende ide. Videre kan man velge mellom høyoppløselig eller lavoppløselig prototyper. Vi antar at skisser vil være et godt utgangspunkt i dette prosjektet ved lavoppløselig prototyping. Dersom man går for høyoppløselig prototyping, kan video eller powerpoint presentasjon være aktuelt.

8. Risikoanalyse

Risiko kan tolkes som en sannsynlighet for at ikke-ønskede hendelser kan oppstå. Det finnes ulike faktorer av risiko, for eksempel: mennesker, teknologi, organisasjon, verktøy, krav og estimering.

Det er veldig viktig å vurdere sannsynligheten og mulige konsekvenser som kan oppstå ved en risiko. Dette kalles for en risikoanalyse. Sannsynligheten måles i en form av denne skalaen: svært lav, lav, moderat, høy eller svært høy. Videre måles konsekvenser i denne type skala: katastrofal, alvorlig, mindre alvorlig eller ubetydelig.

Utviklere av markasykkel bør foreta en risikoanalyse for å minske sjansene for uønskede omstendigheter. Ved å gjøre dette vil man spare mye ressurser, økonomi og tid ved risikoer som kan oppstå.

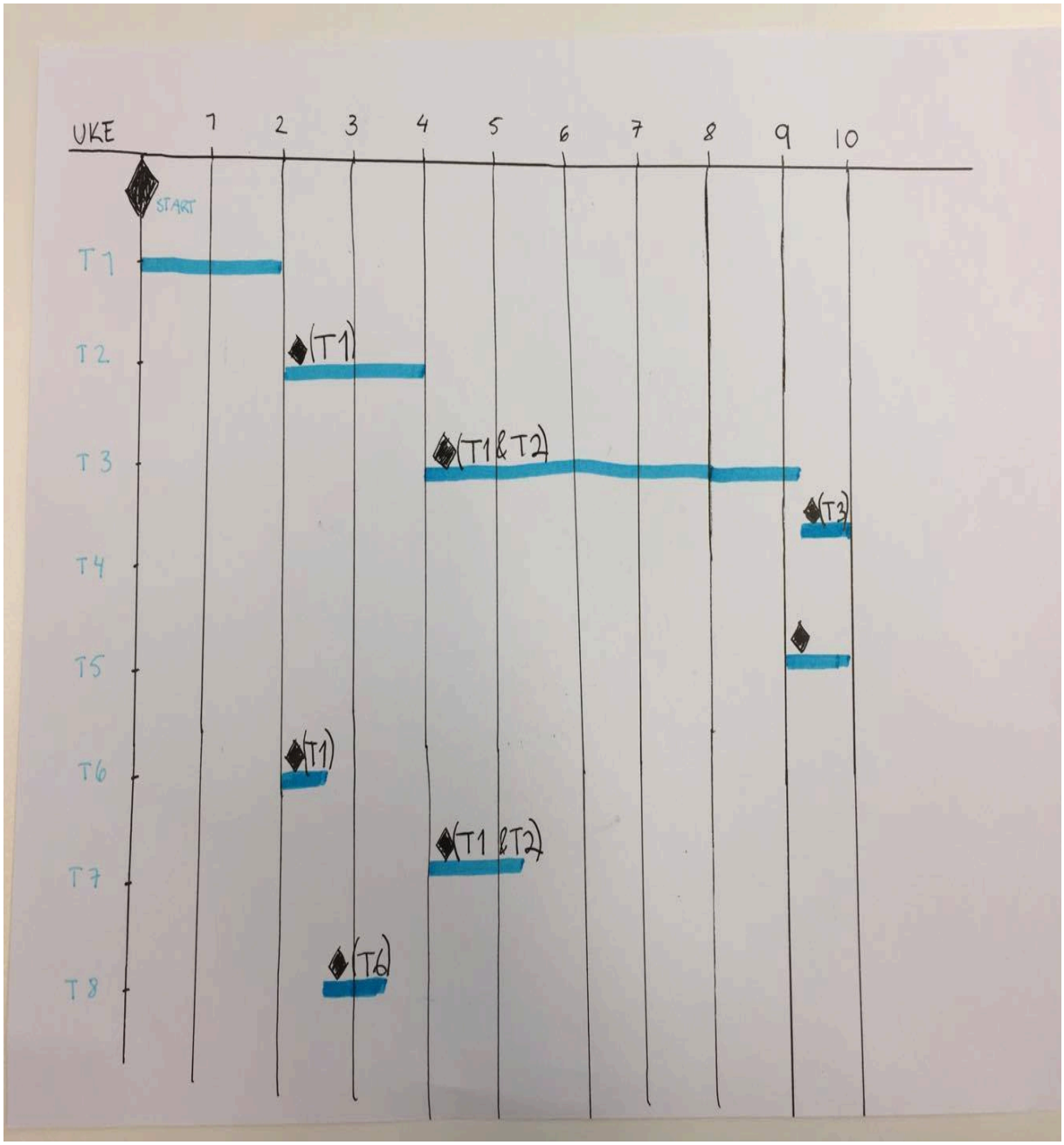
C) Gi hver aktivitet et unikt navn, varighet, eventuelle avhengigheter og milepæler. Gjør dette ved å lage en tabell (ta utgangspunkt i figur 23.5 side 295 i læreboka).

T1= Identifisere krav
T2= Design
T3= Implementasjon
T4= Integrasjon og systemtesting
T5= Drift og vedlikehold
T6= Estimering
T7= Prototyping
T8= Risikoanalyse

M1: Når kravspesifikasjonen er fullført
M2: Når designmodellen er fullført/klargjort.
M3: Når implementasjonen er skrevet ferdig
M4: Når testing av hele systemet og del komponenter er testet ferdig
M5: Når vedlikeholdet i systemet er fungerende bra.
M6: Når estimeringen er antatt ca. hvor mye tid, kostnader og ressurser vil bli brukt
M7: Når både lavoppløselig og høyoppløselig prototype er gjort ferdig
M8: Når risikoanalysen er fullført

TASK	EFFORT	DURATION (Days)	DEPENDECIES
T1	10	15	
T2	4	15	T1 (M7) (M6)
T3	6	40	T2, T1 (M1)
T4	3	2	T3 (M3)
T5	2	1	
T6	2	3	T1, T8
T7	6	10	T1, T2
T8	2	5	T6

c) Bruk tabellen til å lage et stolpediagram med utgangspunkt i figur 23.6 side 297 i læreboka.



D) Lag en risikoanalyse ved å benytte en usikkerhetsmatrise. Få med risiko, sannsynlighet for risiko, konsekvens av risiko, hvilke tiltak som må iverksettes og hvem som er ansvarlig for hvert risikomoment. Ha med minst seks risikomomenter.

Risiko	**Sannsynlighet**	**Konsekvens**	**Tiltak som må iverksettes**	**Ansvarlig**
Hacking	lav	alvorlig	Sikkerhetstiltak, drift og vedlikehold	Sikkerhetsteam
Strømbrudd	Svært lav	Katastrofal	Tilgang til reserveforsyning.	Leverandør / kommune
Kravendringer	Moderat	Alvorlig	Kommunikasjon med bruker. Avklare nye krav	Ledelse / kunde
Leveransetidspunktet av systemet er underestimert	Høy	Mindre alvorlig	Avklare nye estimater	Utvikler
Økonomisk nedgang tvinger til redusering av prosjektbudsjettet	Moderat	Alvorlig	Avsetning for håndtering av videre kostnader i prosjektet. Foreta budsjettanalyser.	Ledelse
En med mindre erfaring innen estimering får ansvaret til å estimere	Moderat	Mindre alvorlig	Tilbud om estimeringskurs.	Ledelse