NTNU
Det skapende universitet

Institutt for Institutt for datateknikk og informasjonsvitenskap (IDI)

Skisse til løsningsforslag

Eksamensoppgave i TDT4140 - Programvareutvikling

Faglig kontakt under eksamen: Faglærer Carl-Fredrik Sørensen Tlf.: 95119690

Eksamensdato: 08.08.2014

Eksamenstid (fra-til): 09:00 – 13.00

Hjelpemiddelkode/Tillatte hjelpemidler: C: Pensumbok: Ian Sommerville, Software Engineering 9. Bestemt, enkel kalkulator tillatt.

Annen informasjon: Eksamen utviklet av Faglærer Carl-Fredrik Sørensen og kontrollert av Torbjørn Skramstad

Målform/språk: bokmål

Antall sider: 13

Antall sider vedlegg: 1

Kontrollert av:

_____

Dato Sign

Dokumenter og begrunn eventuelle forutsetninger eller antakelser.

Oppgave 1: Flervalgsoppgave (25%)

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på den ene selv). Du kan få nytt ark av eksamensvaktene dersom du trenger dette. Kun ett svar er helt riktig. For hvert spørsmål gir korrekt avkryssing 1 poeng. Feil avkryssing eller mer enn ett kryss gir −1/2 poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven. Der det er spesielle uttrykk står den engelske oversettelsen i parentes.

1) **Hva er 'Software Engineering'?**
   a) Er et sett av regler om hvordan man skal utvikle programvare.
   b) Har vært en egen disiplin siden tidlig 50-tallet og er nå en moden ingeniørdisiplin på linje med andre etablerte innenfor ingeniørfeltet.
   c) Startet som en respons på den såkalte 'Software Crisis' sent på 60-tallet.
   d) Er en egen ingeniørdisiplin som omfatter alle aspekter av programvareproduksjon

2) **En programvareutvikler som tilslutter seg «Software Engineering Code of Ethics and Professional Practices» må:**
   a) Alltid å praktisere i henhold til IEEE standarder
   b) Hjelpe kollegaer i profesjonell utvikling
   c) Være medlem av IEEE Computer Society
   d) Unngå alle problemstillinger knyttet til finans og lønnsomhet

3) **En programvareingeniør er ansatt i selskapet som designer og fremstiller avstemmingsmaskiner for bruk i kommune- og stortingsvalg. Hun er ansatt for å utvikle arkivsystemet for å sikre avstemmingsinformasjon og avstemningsresultater. Hvilken av de følgende handlinger vil mest sannsynlig være et brudd på «Software Engineering Code of Ethics and Professional Practice»?**
   a) Hun unnlot å levere korrekt og tilstrekkelig dokumentasjon av sitt arbeid i prosjektet.
   b) Hun har ikke tidligere arbeidet i et prosjekt for utvikling av avstemmingsmaskiner.
   c) Hun er registrert som stemmegiver i kommunen der maskinen vil bli benyttet.
   d) Hun oppdaget et signifikant antall feil i eget design av arkivsystemet.

4) **En livssyklus for programvare kan bli forklart som å bestå av en serie av faser. Den klassiske modellen blir kalt fossefallsmodellen. Hvilken fase kan defineres som "hvor konsepter blir undersøkt og forfinet, og fremkaller derfor kundekrav?"**
   a) Krav
   b) Spesifikasjon
   c) Design
   d) Implementering

5) Hvilke av de følgende elementer skal IKKE være inkludert i en prosjektplan for utvikling av programvare?

a) Teknikkene og case-verktøy som skal bli benyttet
b) Livssyklusmodellen som skal bli benyttet
c) Organisasjonsstruktur på utviklingsorganisasjonen, prosjektoppgaver, ledelsesmål og prioriteter
d) Ingen av alternativene

6) Hva er det korrekte utsagnet i forhold til evolusjonær utvikling/«Evolutionary development».

a) Evolusjonær utvikling kommer normalt i to forskjellige former: Undersøkende/exploratory utvikling og bruk-og-kast prototyping
b) Veldig store prosjekter er naturlige kandidater til evolusjonær utvikling
c) Undersøkende/exploratory utvikling blir benyttet i situasjoner der de fleste kravene er godt kjent på forhånd.
d) Et av de sterkeste punktene for evolusjonær utvikling er at det forenkler prosjektstyring gjennom den store mengden dokumentasjon som blir generert

7) I vedlikeholdsfasen må produktet testes mot tidligere test caser. Dette er også kjent som _____testing.

a) Enhets
b) Integrasjons
c) Regresjons
d) Akseptanse

8) Et eksempel på en risiko involvert i programvareutvikling

a) Produsenten av kritiske komponenter (f.eks. maskinvare som er assosiert med et sanntidssystem) kan gå konkurs.
b) Teknologiendringer kan medføre at produktet blir foreldet
c) Konkurrenter kan markedsføre et tilsvarende produkt til en lavere pris
d) Alle er riktig

9) Graden av interaksjon mellom to moduler er kjent som

a) Kohesjon/Cohesion
b) Arv/Inheritance
c) Kobling/Coupling
d) Instansiering/Instantiation.

10) Interne kostnader i et prosjekt inkluderer

a) Lønn til utviklere
b) Lønn til ledere og støttepersonell

c) Overheadkostnader f.eks. til hjelpesystemer, husleie og toppledelse
d) Materialer (som f.eks. manualer og bøker) og tjenester som f.eks. reiser

11) Resultat fra algoritmisk kostnadsestimering i forskjellige organisasjoner kan være forskjellige for den samme type applikasjonsutvikling fordi

a) Forskjellige organisasjoner betrakter kompleksitetsfaktorer forskjellig og kan benytte forskjellige programmeringsspråk
b) Dyktigheten til utviklere kan variere
c) Teknikker for måling av produktivitet kan variere
d) Alle er riktige.

12) Hvilke av elementene under fanger best naturen til et kvalitetsparadigme?

a) Kvalitetskarakteristikker, prosessperspektiv, eliminering av defekter
b) Målinger, kvalitetskontroll, validering
c) Gjennomførbarhet, krav, økonomi, brukerbehov
d) Analyse, test, design

13) Når man velger en livssyklusmodell for programvareutvikling, så vil man ta i betraktning:

a) Ekspertisen til utviklingsgruppen, problemkarakteristikker, brukerforventninger
b) Programmeringsspråk, tidsramme for utvikling, konkurransesituasjon
c) Systemkontekst, brukerpopulasjon, plattformer
d) Organisatorisk struktur, brukeroppgaver, ytelseskriteria

14) Hvilke av de følgende er IKKE betraktet som en primær drivkraft for å forbedre programvareprosessen?

a) Økt effektivitet
b) Bedre produktkvalitet
c) Forbedring av personaltilfredshet
d) Tettere ledelseskontroll

15) Når man skal planlegge et programvareprosjekt vil man:

a) Tilpasse tidsplanen for å kunne håndtere eventuelle feil
b) Finne måter å produsere resultater med tilstrekkelig kvalitet ved hjelp av begrensede ressurser.
c) Overestimere budsjettet (PI-faktor)
d) Alle svarene er riktig

16) Hvor mange tester må lages for å teste fire betingelser i en IF-setning?

a) 4
b) 8
c) 12
d) 16

17) For å være et effektivt verktøy i prosessforbedring, så må underlagsmetrikkene være…
a) Basert på rimelige estimater fra mislykkede prosjekter
b) Målt konsistent på tvers av prosjekter
c) Basert bare på vellykkede prosjekter
d) Basert på mislykkede prosjekter

18) En programvareutvikler måler kvaliteten av et programvaresystem. Hun er bekymret for påliteligheten («reliability») og validiteten («validity») av målingene sine. Hvilken av følgende uttalelser er sann?
a) Pålitelighet refererer til i hvilken grad målingene representerer den egentlige kvaliteten til systemet og validitet refererer til konsistensen av hennes kvalitetsmålinger.
b) Pålitelighet refererer til konsistensen av hennes kvalitetsmålinger og validitet til i hvilken grad målingene representerer den egentlige kvaliteten til systemet.
c) Pålitelighet refererer til nøyaktigheten på hennes kvalitetsmålinger og validitet referer til i hvilken grad målingene følger en kvalitetsstandard.
d) Pålitelighet refererer til samtidigheten av hennes kvalitetsmålinger og validitet refererer til i hvilken grad målingene er konsistent med etablerte normer.

19) DIFI har definert sju arkitekturprinsipp. Seks av disse er tjenesteorientering, interoperabilitet, sikkerhet, åpenhet, tilgjengelighet og skalerbarhet. Hva er det sjuende?
a) Fleksibilitet
b) Funksjonsstabilitet
c) Testbarhet
d) Ytelse

20) Interoperabilitet er et viktig arkitekturprinsipp. Hvilke typer interoperabilitet er definert av DIFI?
a) Obligatorisk bruk av web-protokollene SOAP, REST og JSON
b) Organisatorisk, semantisk og teknisk interoperabilitet
c) Etablering av felles IKT-løsninger i offentlig sektor
d) Definerte standarder for kommunikasjon og integrasjon

INNLEVERING

Svarskjema: Oppgave 1

Oppgavenr	A	B	C	D
1.1			X	
1.2		X		
1.3	X			
1.4	X			
1.5				X
1.6	X			
1.7			X	
1.8				X
1.9			X	
1.10	X			
1.11				X
1.12	X			
1.13	X			
1.14				X
1.15		X		
1.16				X
1.17		X		
1.18		X		
1.19	X			
1.20		X		

Oppgave 2 – Programvareutvikling (50%)

Det skal utvikles en generisk og konfigurerbar IKT-løsning for å lage spesifikke feltloggings-apper for mobile enheter. Løsningen skal kunne tilpasses forskjellige typer observasjonsstudier, f.eks. innen biologi, geologi, arkeologi, miljøforvaltning, inspeksjoner av konstruksjoner og sosialvitenskap. Vedlegg A (på engelsk) beskriver et eksempel på et observasjonsstudium innen biologi.

Appene av en type feltstudie skal kunne kjøre på forskjellige typer klienter (pc, mobil, etc.) og skal kunne synkroniseres mot ett felles datalager ved konnektivitet (3G, WLAN etc.). Appene skal kunne integreres med GPS for posisjonering og kunne lagre multimedia-objekter knyttet til feltstudiet.

IKT-løsningen kan deles inn i to deler:

1. Konfigureringssystem: Benyttes for å bygge opp instanser av feltloggings-apper

2. Målapplikasjoner: Forskjellige varianter av app-er for feltlogging.

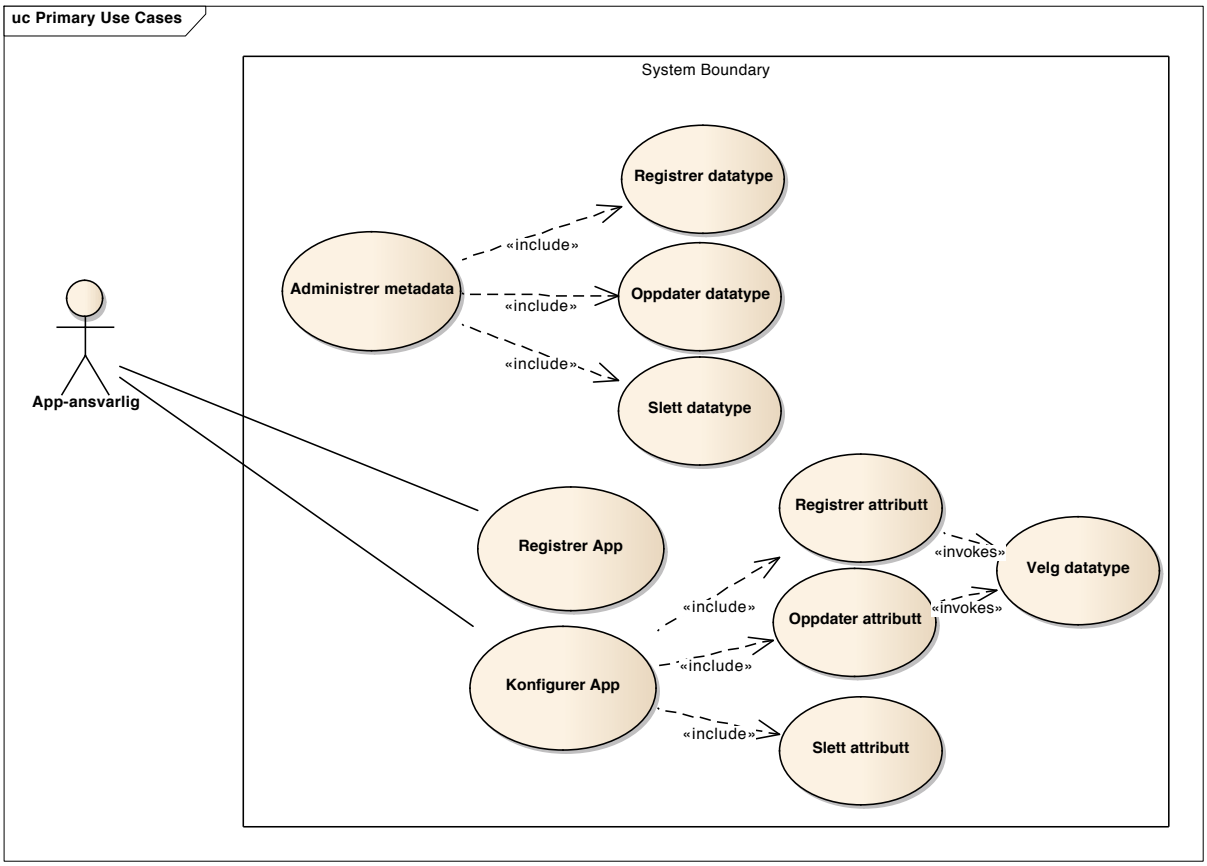
a) *(5%) Lag et UML Use case-diagram for henholdsvis konfigurasjonssystemet, og spesifikt for spurve-appen som beskrevet i vedlegget.*

Aktører: App-ansvarlig og App-bruker

Eksempler på Use case for konfigurasjonssystem hvor *App-ansvarlig* er primær aktør:

1. Registrer app
2. Konfigurer app
    a. Velg studietype
    b. Registrer attributt
    c. Oppdater attributt
    d. Slett attributt
    e. Lag layout
    f. Registrer inputmetode (tekstlig, bilde, video, GPS, klokke, listevalg etc.)
3. Administrer metadata
    a. Registrer datatype med lovlige verdier (heltall, liste,)
    b. Oppdater datatype
    c. Slett datatype
    d. Administrer layouts
    e. Administrer studietyper (observasjon, felt, spørreundersøkelse, etc.)
4. Generer app for server (databaseskjema og protokoller)

5. Generer app for klient
6. Oppdater app for server
7. Oppdater app for klient



Use case for spurve-app:

1. Registrer nytt individ
2. Oppdater informasjon om individ
3. Registrer hendelse på individ
4. Slett individ
5. Lagre individ
6. Synkroniser informasjon
7. Søk etter individ

b) (5%) Lag et tekstlig use case for scenariet: Konfigurer spurve-app. Dersom det baserer seg på extends/use relasjoner med andre use case, lag også tekstlige use case for disse.

Use Case (Navn): Konfigurer spurve-app (er en spesialisering av det generelle use-caset Konfigurer App)

Aktør: App-ansvarlig

Mål: Konfigurere hvilke informasjonselementer som skal kunne registreres i en feltlogg (om spurver i dette tilfellet)

Beskrivelse: En App-ansvarlig skal kunne lage nye App-typer der det kan spesifiseres hvilke attributter som skal kunne registreres for en hendelse, individ eller undersøkelse

Type: Primær use case

Pre betingelser: Use caset «Registrer App» er utført. Datatyper som kan benyttes er registrert gjennom use caset «Registrer datatype». Det antas at det finnes standardintegrasjoner med GPS, klokke, enhetsinformasjon (bruker) og annen informasjon som kan være knyttet til klienten.

Post betingelser: Systemet skal kunne generere server-komponenter og klient-komponenter, inkludert bruk av kommunikasjonsprotokoller og bruk av informasjon fra den mobile klienten (f.eks. tid og posisjon).

Spesielle krav (eks. operasjonelle krav): Ingen spesielle

Detaljert Hendelsesforløp (”Main Success Scenario”) :

1. Aktør velger App som skal konfigureres
2. Aktør velger layout på målapp
3. Aktør velger aksjon: «Registrer attributt»
4. Aktør spesifiserer navn på attributt.
5. Aktør velger om attributtet er en hendelse i en tidsserie eller en beskrivelse av selve objektet (individual data: spurv).
6. Datatype for attributt velges gjennom use caset «Velg datatype»
7. Aktør velger plassering av attributt i layout (Fane, plassering, etc)
8. Aktør velger input-metode for dataverdier (manuell eller automatisk registrering)
9. Aktør velger synkroniseringsmetode

Alternative scenarier: Aktør velger aksjon: «Oppdater attributt» eller «Slett attributt» i pkt 3.

c) (5%) Lag et aktivitetsdiagram for use caset: ”Konfigurer spurve-app” basert på Oppgave 2b.

Hendelsesforløpet i forrige oppgave som et sett aktiviteter som følger hverandre.

d) (5%) Definer de viktigste ikke-funksjonelle kvalitetskrav/arkitekturkrav til appene som blir laget.

a. **Tilgjengelighet**: App-ene skal kunne genereres for de fleste mobile plattformer (iOS, Android, Windows Mobile, etc.), samt for Windows. Oppetid og responstid: Oppetid er spesielt kritisk for klient under feltarbeid så kvalitetskrav på klienten vil derfor være høy.

Responstiden vil være knyttet til oppstart av app (under ett minutt) samt normal bruk (responstid for operasjoner under ett sekund). Brukbarhet: De genererte appene skal kunne benyttes i felt, under forskjellige værforhold, og på utstyr med begrenset skjermkapasitet.

b. **Sikkerhet** (security): Innsamlet informasjon skal kunne lagres og kommuniseres på en slik måte at informasjon ikke blir tappet, endret eller er tilgjengelig for andre enn de som er autorisert. Løsningen vil kreve kryptering, spesielt for informasjon som er knyttet til personer for å opprettholde personvern,

c. **Funksjonsstabilitet**: App-ene skal kunne benyttes både i online og offline modus. Synkronisering av informasjon mellom klient og tjener, må være transaksjonssikkert.

d. **Interoperabilitet**: Server-systemet må kunne kommunisere med app-er som kjører på forskjellige type operativsystemer.

e) (10%) Lag en arkitekturskisse for konfigurasjonssystemet og en arkitekturskisse for spurve-appen. Hvilke hovedkomponenter og koblinger/grensesnitt må tilbys?

Konfigurasjonssystem: Konfigurasjonsklient, Metadataklient, Konfigurasjonsdatabase, Metadata-database, App-database (man kan her ha flere muligheter, enten en database for alle apper og instanser av disse eller en database pr. app).

Spurveapp: Registreringsklient, lokal database, sentral database, Inputkomponenter: «tastatur», GPS, trykkskjerm, klokke, kommunikajonsprotokoller (Wifi, GPRS, 3G etc.), Synkronisering, Søk

f) (5%) Hvilken eller hvilke arkitekturmønstre (patterns) bør et slikt system benytte? Beskriv hvordan og hvorfor et valgt mønster bør benyttes.

a. Klient/Server – Klient/Server arkitektur vil være mest hensiktsmessig. Merk imidlertid at klienten må kunne kjøre en lokal Server-komponent for å håndtere offline-situasjoner. Dette betyr i prinsippet at klienten alltid vil arbeide mot en lokal Server-komponent (database), og at det ved konnektivitet vil være mulig å kommunisere med en sentral Server-komponent for synkronisering/oppdatering av data.

b. SOA-arkitektur – Informasjon må kunne hentes fra Server ved tilgjengelighet. Server må kunne tilby informasjons- og synkroniseringstjenester. Det er hensiktsmessig med veldefinerte protokoller for både innhold (WSDL) og kommunikasjon (http/https)

c. MVC – Denne vil være hensiktsmessig for utforming av klienten. Konfigurasjonssystemet må kunne lage både modell, view og controller komponenter basert på input fra bruker, samt hvilken målplattform som skal benyttes. HTML5 vil være et godt alternativ som view.

d. Repository – Denne kan være hensiktsmessig dersom det er en felles synkroniseringsplattform for alle type apper og instanser av disse.

e. Lagdelt – Klient/server, repository, MVC, SOA er alle eksempler på en lagdeling.

g) (5%) Definer de viktigste klassene og lag et UML klassediagram for konfigurasjonssystemet. Inkluder de viktigste metodene på formen <response> metodeNavn(<parametere>). Argumenter hvorfor disse er de viktigste metodene..

Konfigurasjonssystemet vil ha en del klasser knyttet til håndtering av metainformasjon, attributter, apper og brukere.

Hovedklasser: App, Datatype, Attributt, Inputmetode

h) *(10%) Definer en grov prosjektplan for utvikling av konfigurasjonssystemet. I prosjektplanen må det defineres hovedaktiviteter, milepæler og avhengigheter. Skisser hvordan prosjektet eventuelt kan deles inn i delprosjekter. Argumenter for hvilken utviklingsprosess som vil være mest hensiktsmessig.*

Det er mulig å bruke både smidige og plandrevne prosesser. Smidige prosesser krever kundeinvolvering gjennom prosjektet, mens i en plandreven prosess er de fleste krav kjent på forhånd. Antall utviklere og utviklingstid vil påvirke hvilken prosess som er best. Det vil uansett være behov for å aktiviteter som planlegging, prosjektledelse, spesifisering, design og implementering samt validering og verifisering av løsningen.

Hovedtyngden av arbeid vil være å skape et design av konfigurasjonssystemet som gjør at man kan lage mange forskjellige type feltstudier basert på det samme brukergrensesnittet. Dette medfører at man må definere en meta-modell-løsning som kan benyttes til å definere ulike typer datafelter.

Oppgave 3: Risikostyring, komponentbasert utvikling, produktlinjer (25%)

a) (10%) Følgende strategier kan bli benyttet i systemdesign til å sikre påliteligheten (dependability) til et kritisk system for å håndtere risikoer i forhold til at en ulykke eller hendelse skjer når systemet er i operasjon. Beskriv hver strategi med passende eksempler og diskuter potensielle problemer som er assosiert med hver strategi:

   i. Skadebegrensning (damage limitation)

   ii. Oppdagelse og fjerning av risiko

   iii. Risikounngåelse

b) (5%) Forklar hva som er ment med komponentbasert programvareutvikling (CBSE) og diskuter innvirkningen åpen kildekode-prosjekter, tilgjengelig på Internett, har på CBSE-praksisen.

c) (10%) Forklar hva som er ment med en produktlinje av programvare i kontekst av programvareutvikling. Diskuter hvordan og hvorfor et programvarehus kan ønske å utvikle sin egen produktlinje. Svaret må inkludere passende eksempler.

Vedlegg A

Project description for the house sparrow project.

1. Background:
The house sparrow project at Department of Biology NTNU has collected data from natural populations of house sparrows along 'Helgelandkysten' since 1993. The birds are captured and nests are checked every year on several islands over a large area. Fieldwork is performed during the breeding season (May-Aug) as well as shorter periods during autumn and winter. Often there are several teams of field workers collecting data at the same time in different areas.

2. Field data

2.1 Data types
Field data can be organised into different categories, often representing different temporal and biological scales.

Island data: e.g. position, size and other permanent attributes.

Population/colony data: e.g. annual populations sizes

Nest data: e.g. position

Clutch data: number of eggs, number chicks, age of chicks.

Individual data: morphology, parasites, sex, age et c is collected each time an individual is captured or observed, often on several different occassions per year, and in several different years. Different (and fewer) variables are collected from chicks in the nests than from adults that are captured. In addition to capture data, we also get individual data by observing, with binocular, the unique combinations of colour rings that each bird is marked with.

2.2 Collection of field data

Present dataflow
Data has so far been recorded in the field with pen and paper, with no possibilities of validating the data, time-consuming and error-prone transcription and post-processing of data and post-processing of data. All data has been stored together in a two-dimensional spread-sheet format in SPSS.

Future dataflow
We wish to use a tool for gathering of field data – a local mobile application with user interface that obviate the need to transcribe field data collected on paper, that ensure validation and a consistent input. All entered data should be transfered to a central data

repository. Access to subsets of the data from the central repository should also be made available locally, e.g. to see previously recorded data from a specific individual, to use previously recorded data to validate data that is being entered. Due to poor mobile net in some parts of the study we need to have the ability to collect data offline, which then is synchronized/backed-up as soon as Data needs to be synchronized between mobile and central systems, and ideally between mobile units.