NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Computer and Information Science

Examination paper for TDT4240 Software Architecture

The English version is the reference version of the examination paper!

Academic contact during examination: Alf Inge Wang

Phone: +47 9228 9577

Examination date: Tuesday June 2nd 2015

Examination time (from-to): 09:00 – 13:00

Permitted examination support material:

- IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society.
- Kruchten, P. (1995), "The 4+1 View Model of Architecture", IEEE Software, 12(6).
- English-Norwegian dictionary (or to your native language if your not Norwegian) and/or an English thesaurus (English-English).

Other information:

- Simple calculator or a calculator approved by NTNU allowed.

Language: English

Number of pages: **5**

Number of pages enclosed: **15**

Checked by:

22/5-2015 Anh Nguyen Duc

Date Signature

Problem 1: Various questions (20 points)

Answer these questions *briefly*:

1.1	What is Bass, Clements and Kazman's definition of Software Architecture (the definition in the textbook)?
1.2	What is the purpose of the ***Module Views*** as described in the textbook?
1.3	What is the purpose of the ***Component-and-Connector Views*** as described in the textbook?
1.4	What is the purpose of the ***Allocation Views*** as described in the textbook?
1.5	Explain why this statement is either true or false: "Every complex software system has a software architecture".
1.6	What is the purpose of the ***Development View*** in the 4+1 view model?
1.7	What are the main influences on an architect designing a software architecture according to the textbook?
1.8	Name five important reasons for why software architecture is important.
1.9	Name the three main groups for availability tactics, and give one example of a specific tactic for every group.
1.10	Draw a simple class diagram to illustrate the ***Template Method*** design pattern.
1.11	What is an ***Architecturally Significant Requirement (ASR)/Architectural Driver***?
1.12	Name examples of three parameters that are commonly used in performance models according to the textbook.
1.13	What can be used to *analyze* the quality attribute ***security*** in a software architecture?
1.14	Describe the process of ***Attribute-Driven Design (ADD)*** in the textbook.
1.15	What is ***Architectural erosion***?
1.16	Describe three techniques to help keeping the code and the software architecture consistent as described in the textbook.
1.17	What is ***ATAM*** an abbreviation for?
1.18	Name five outputs of the ***ATAM***.
1.19	Briefly outline the process of ***reconstructing a software architecture***.
1.20	What must have a high priority when designing a software architecture for a ***software product line***?

References:

- Bass, Clements & Kazman: "Software Architecture in Practice", 3rd edition, Pearson, 2013.
- Philippe Kruchten, "Architectural Blueprints – The 4+1 View model of Software Architecture", IEEE Software 12 (6), 1995

Problem 2: Choose the most appropriate architectural pattern (5 points)

Nominees:

a) Layered
b) Broker
c) Model-View-Controller
d) Pipe-and-Filter
e) Client-Server
f) Peer-to-Peer
g) Service-Oriented
h) Publish-Subscribe
i) Map-Reduce
j) Multi-tier

Choose the *most appropriate architectural pattern* (one) for the 5 descriptions below. Motivate for your choice in one sentence (give reasons for choosing the pattern):

1. Need software to encode text files. The software consists of several algorithms of encryption, which can be combined for better result.
2. Need software for a digital speedometer for boats. The software should be flexible to support various number and types of displays (from simple LED-displays to large screens), and various ways of graphical designs of showing speed on the display.
3. Need software for a smartphone game where players can compete against each other through Bluetooth (no need to access the Internet).
4. Need software for managing and providing different kinds of information (weather, location, points-of-interest etc.) to app- and web-developers. The information will be gathered from various sources and processed by through the software. The software should provide open data access to anyone who wants to use the information through standard APIs.
5. Need software to very efficiently find the city in the world with a population above one million with the highest measured temperature. The temperatures measured from each city for 100 years are stored in separate files containing 36500 temperature entries. There are in total 457 cities with a population above 1 million.

Problem 3: Edge-dominant system (5 points)

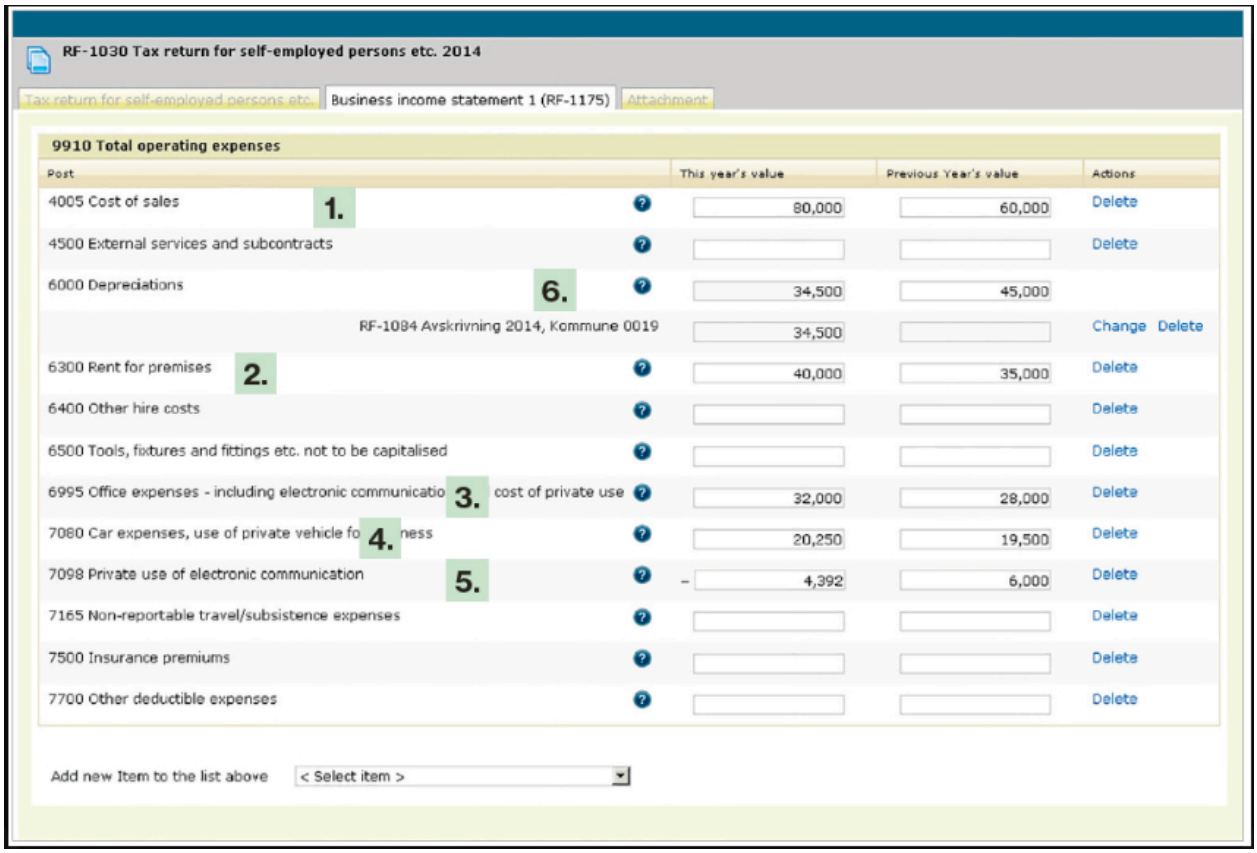
a) What is important in the design of the core architecture of an Edge-dominant system? (2 points)
b) What is the difference of open content systems and open source software? (2 points)
c) Give two examples of systems that can be classified as open content systems and two examples of systems that can be classified as open source software. (1 point)

Problem 4: Cloud Architecture (4 points)

a) Explain the difference between the terms private cloud, public cloud, community cloud and hybrid cloud. (2 points)
b) What are the basic mechanisms needed to provide a cloud service described in the textbook? (2 points)

Problem 5: Quality Attribute Scenario (6 points)

System description: A web-based system for tax return (Selvangivelsen). In this system, the users will correct/change or add information related to their personal economy in order to compute how much tax they have to pay or they get refunded from the state.
The user interface of this system can typically look like the following:



a) Create one relevant quality attribute scenario on usability for the system described above. (2 points)
b) Create one relevant quality attribute scenario on modifiability for the system described above. (2 points)
c) Create one relevant quality attribute scenario on security for the system described above. (2 points)

Problem 6 Design a software architecture (30 points)

Read the description below and do an architectural design. Your answer must include:

a) Architectural Significant Requirements/Architectural Drivers – 2 points
b) Architectural tactics and patterns – 3 points
c) Process view – 8 points
d) Logical view – 14 points
e) Architectural rationale – 3 points

Motivate for your choices and state your assumptions.

Software for Conference Refereeing

Professional conferences are held in order to announce and discuss new results. The core activity of organizing a conference focuses on selecting the papers to be presented. Usually this is done by making an open invitation calling for papers to be submitted, circulating the submitted papers to a (geographically distributed) panel of reviewers, then selecting the best papers to appear on the conference program. A system to automate conference refereeing should do the following:

- The program committee announces "call for papers" by entering necessary conference information into the system. The program committee can add email addresses of potential authors or addresses from various mailing-lists into the system.
- Potential authors receive the call for papers by email from the system and decide to submit papers on their work. They write papers, and submit them using the system, where they add needed information as well as upload a PDF file of the paper. A given paper may have several authors, but there is only one corresponding author with one reply address. A notification is sent to the author after a successful submission has been made.
- The program committee consisting of several reviewers goes through a bidding process where they bid on the papers they want to review, and mark the papers they cannot review due to conflicts of interest.
- The system will then distribute the papers among the reviewers in the program committee, who will review the papers and enter their comments and score for each paper into the system.
- The system will rank the papers according to their scores and give the program committee an opportunity to mark the papers that will be accepted.
- When the program committee has decided on the accepted/rejected papers, they can through the system notify all the authors of the papers whether the papers got accepted or rejected.

The Conference Refereeing system can be accessed through a web-browser on PCs, tablets or smart phones. The users can also access the system using an app for tablets and smart phones. Typical customers of the system will be organizers of business conferences, research conferences and military conferences. For the latter, it is important that the papers submitted to the conference that can contain classified information should not be accessible for people outside military organizations.

NTNU – Trondheim
Norwegian University of
Science and Technology

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4240 Programvarearkitektur

Engelsk versjon er referanseversjon for eksamensoppgaven!

Faglig kontakt under eksamen:	Alf Inge Wang
Telefon:	+47 9228 9577

Eksamensdato:	Tirsdag 2.juni 2015
Eksamenstid (fra-til):	09:00 – 13:00

Hjelpemiddelkode/Tillatte hjelpemidler:

- IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society.
- Kruchten, P. (1995), "The 4+1 View Model of Architecture", IEEE Software, 12(6).
- Engelsk-Norsk ordbok (eller fra ditt morsmål hvis det ikke er norsk) og/eller Engelsk synonymordbok (Engelsk-Engelsk).

Annen informasjon:

- Enkel eller godkjent kalkulator

Målform/Språk:	Bokmål
Antall sider:	**5**
Totalt antall sider (alle språk):	**15**

Kontrollert av:

22/5-2015	Anh Nguyen Duc
Dato	Signatur

Oppgave 1: Diverse spørsmål (20 poeng)

Svar kort på disse spørsmålene:

1.1	Hva er Bass', Clements' and Kazmans definisjon på programvarearkitektur (i følge læreboka)?
1.2	Hva er formålet med ***Module Views*** som beskrevet i læreboka?
1.3	Hva er formålet med ***Component-and-Connector Views*** som beskrevet i læreboka?
1.4	Hva er formålet med ***Allocation Views*** som beskrevet i læreboka?
1.5	Forklar hvorfor følgende påstand enten er sann eller usann: "Ett hvert komplekst programvaresystem har en programvarearkitektur".
1.6	Hva er formålet med ***Development View*** i 4+1 view modellen?
1.7	Hvilke hoved påvirkninger finnes på en arkitekt som designer en programvarearkitektur i følge læreboka?
1.8	Oppgi fem viktige grunner til at programvarearkitektur er viktig.
1.9	Navngi de tre hovedgruppene for tilgjengelighetstaktikker, og gi eksempel på en spesifikk taktikk for hver gruppe.
1.10	Tegn et enkelt klassediagram som illustrerer designmønstret ***Template Method***.
1.11	Hva er en ***Architecturally Significant Requirement (ASR)/Architectural Driver***?
1.12	Navngi eksempler på tre kjente parametere som brukes i ytelsesmodeller i følge læreboka..
1.13	Hva kan brukes til å analysere kvalitetsattributten sikkerhet i programvarearkitektur?
1.14	Beskriv prosessen for ***Attribute-Driven Design (ADD)*** ifølge læreboka.
1.15	Hva er arkitektural erosjon (***Architectural erosion)***?
1.16	Beskriv tre teknikker for som hjelper å holde kode og programvarearkitekturen konsistent som er beskrevet i læreboka.
1.17	Hva er ***ATAM*** en forkortelse for?
1.18	Navngi fem produkter (outputs) fra ***ATAM***.
1.19	Kort beskriv prosessen for å rekonstruere en programvarearkitektur.
1.20	Hva må ha høy prioritet når man designer en programvarearkitektur for en ***software product line***?

Referanser:

- Bass, Clements & Kazman: "Software Architecture in Practice", 3rd edition, Pearson, 2013.
- Philippe Kruchten, "Architectural Blueprints – The 4+1 View model of Software Architecture", IEEE Software 12 (6), 1995

Oppgave 2: Velg arkitekturmønstret som passer best (5 poeng)

De nominerte:

a) Layered
a) Broker
b) Model-View-Controller
c) Pipe-and-Filter
d) Client-Server
e) Peer-to-Peer
f) Service-Oriented
g) Publish-Subscribe
h) Map-Reduce
i) Multi-tier

Velg det *mest passende arkitekturmønsteret* (ett) for de 5 beskrivelsene under. Motiver for dine valg med en setning (gi grunner for valg av mønster):

1. Trenger programvare for å kode tekstfiler. Programvaren består av flere krypteringsalgoritmer som kan kombineres for beste resultat.
2. Trenger programvare for digitalt sppedometer for båter. Programvaren skal være fleksibel å støtte et varierende antall og typer displayer (fra enkle LED-display til store skjermer), og ulik grafisk design på å vise hastighet på display.
3. Trenger programvare for et smarttelefonspill, der spillere kan spille mot hverandre via blåtann (krever ingen tilgang til internett).
4. Trenger programvare for å håndtere og tilby ulik type informasjon (vær, lokasjon, interessepunkter osv.) til app- og web-utviklere. Informasjon vil bli samlet fra ulike kilder og skal prosesseres igjennom programvaren. Programvaren skal tilby åpen datatilgang for alle som ønsker å bruke informasjon igjennom stander APIer.
5. Trenger programvare for å på en veldig effektiv måte finne byen i verden med over en million innbyggere med høyeste målte temperatur. Temperaturer målt fra hver by i over 100 år er lagret i separate filer som inneholder 36500 datapunker med temperaturer. Det finnes totalt 457 byer med befolkning på over 1 million.

Oppgave 3: Edge-dominant systemer (5 poeng)

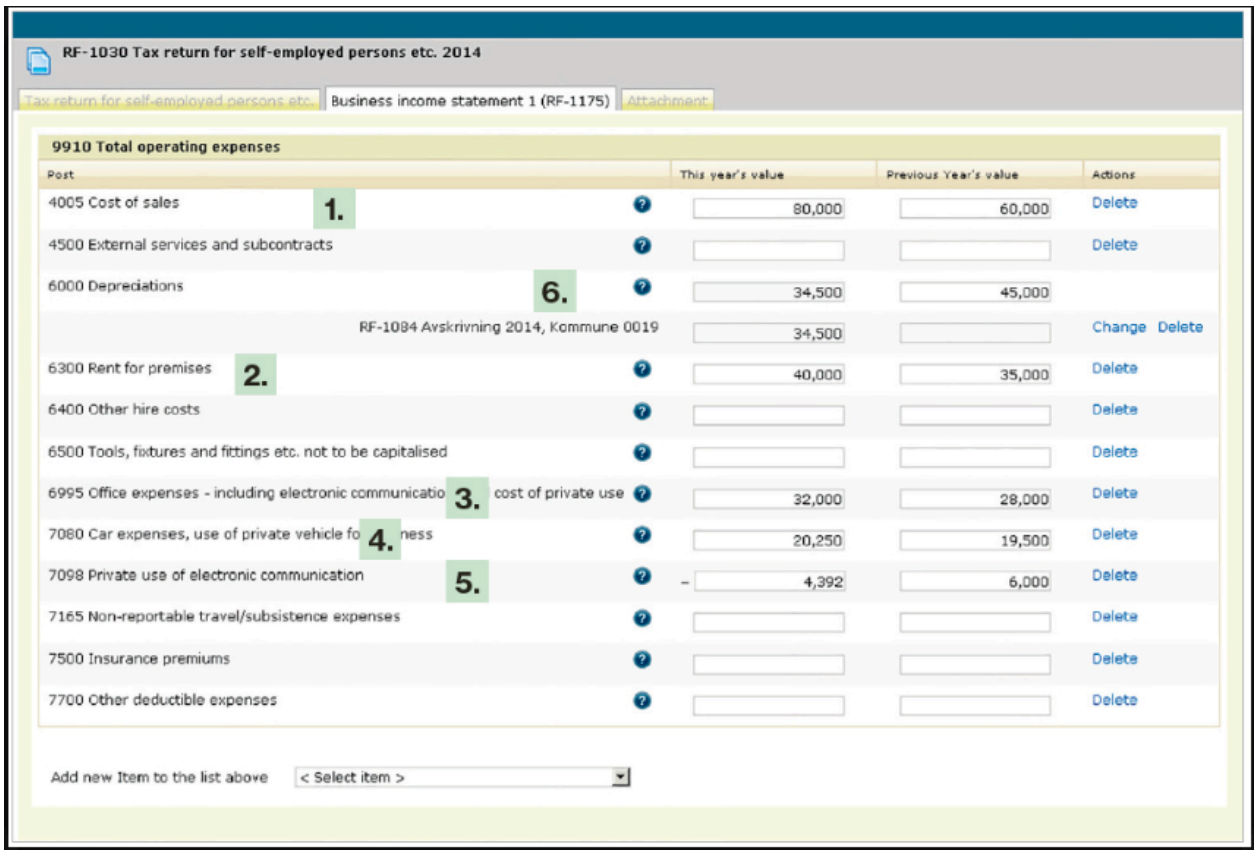
a) Hva er viktig i design av kjernearkitekturen for et *Edge-dominant* system? (2 poeng)
b) Hva er forskjellen på *open content systems* og *open source software*? (2 poeng)
c) Oppgi to eksempler på systemer som kan klassifiseres som *open content systems* og to eksempler på systemer som kan klassifiseres som *open source software*. (1 poeng)

Oppgave 4: Skyarkitekturer (4 poeng)

c) Forklar forskjellene på begrepene *private cloud*, *public cloud*, *community cloud* og *hybrid cloud*. (2 poeng)
d) Hvilke basismekanismer trengs for å tilby en skytjeneste ifølge læreboka? (2 poeng)

Oppgave 5: Kvalitetsattributt-Scenario (6 poeng)

Systembeskrivelse: Et web-basert system for selvangivelsen. I dette system, kan brukere rette opp/endre eller legge til informasjon relatert til deres personlig økonomi for å kunne beregne hvor mye skatt de må betale eller får tilbakebetalt fra staten. Brukergrensesnittet for et slikt system kan typisk se ut som følgende:



a) Lag et relevant kvalitetsattributt-scenario på brukbarhet (*usability*) for systemet beskrevet ovenfor. (2 poeng)
b) Lag et relevant kvalitetsattributt-scenario på modifiserbarhet (*modifiability*) for systemet beskrevet ovenfor. (2 poeng)
c) Lag et relevant kvalitetsattributt-scenario på sikkerhet (*security*) for systemet beskrevet ovenfor. (2 poeng)

Oppgave 6 Design en programvarearkitektur (30 poeng)

Les beskrivelsen under og utfør et arkitekturdesign. Svaret må inkludere:

a) *Architectural Significant Requirements/Architectural Drivers* – 2 poeng
b) Arkitekturtaktikker og –mønster (*patterns*) – 3 poeng
c) *Prosess* view – 8 poeng
d) Logisk view – 14 poeng
e) Motivasjon/Begrunnelse for arkitektur (*Architectural rationale*) – 3 poeng

Motiver for dine valg og beskrive dine antagelser.

Programvare for artikkelvurdering

Profesjonelle konferanse arrangeres for å bekjentgjøre og diskutere nye resultater. Kjerneaktiviteten i å arrangere konferanser har fokus på å velge ut artikler som skal presenteres. Vanligvis gjøres dette ved en åpen invitasjon til å sende inn artikler, sirkulere de innsendte artiklene til et panel av reviewere (vanligvis spredt geografisk), og så velge ut de beste artiklene som vil bli inkludert i konferanseprogrammet. Et system for automatisk artikkelvurdering skal gjøre følgende:

- Programkomiteen annonserer "call for papers" ved å legge inn nødvendig konferanseinformasjon i systemet. Programkomiteen kan legge til epost-adresser til potensielle artikkelskrivere samt adresser til diverse mailing-lister i systemet.
- Potensielle artikkelskrivere mottar epost om ”call for papers” fra systemet, og bestemmer seg for å sende inn deres arbeider. De skriver artikkelene og sender dem inn via systemet, hvor de legger til nødvendig informasjon samt å laste opp PDF-fila for artikkelen. En artikkel kan ha flere forfattere, men kun en forfatter er korrespondansen med gitt svaradresse. En bekreftelse sendes til korresponderende forfatter etter at en godkjent innsending er gjennomført.
- Programkomiteen som består av flere reviewere går igjennom en ”bidding” prosess, hvor de byr på artikler de artiklene de ønsker å reviewe, og marker artiklene de ikke kan reviewe pga. interessekonflikter.
- Systemet vil så distribuere artikler mellom reviewerne i programkomiteen, som vil reviewe artiklene og legge inn kommentarer og en score på hver artikkel inn i systemet.
- The reviewers will review the papers and enter their comments and score for each paper into the system.
- The system will rank the papers according to their scores and give the program committee an opportunity to mark the papers that will be accepted.
- When the program committee has decided on the accepted/rejected papers, they can through the system notify all the authors of the papers whether the papers got accepted or rejected.

The Conference Refereeing system can be accessed through a web-browser on PCs, tablets or smart phones. The users can also access the system using an app for tablets and smart phones. Typical customers of the system will be organizers of business conferences, research conferences and military conferences. For the latter, it is important that the papers submitted to the conference that can contain classified information should not be accessible for people outside military organizations.

NTNU – Trondheim
Norwegian University of
Science and Technology

Department of Computer and Information Science

Examination paper for TDT4240 Software Architecture

The English version is the reference version of the examination paper!

Academic contact during examination: Alf Inge Wang

Phone: +47 9228 9577

Examination date: Tuesday June 2nd 2015

Examination time (from-to): 09:00 – 13:00

Permitted examination support material:

- IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society.
- Kruchten, P. (1995), "The 4+1 View Model of Architecture", IEEE Software, 12(6).
- English-Norwegian dictionary (or to your native language if your not Norwegian) and/or an English thesaurus (English-English).

Other information:

- Simple calculator or a calculator approved by NTNU allowed.

Language: English

Number of pages:

Number of pages enclosed:

Checked by:

22/5-2015 Anh Nguyen Duc

Date Signature

Problem 1: Various questions (20 points)

Answer these questions *briefly*:

1.21	What is Bass, Clements and Kazman's definition of Software Architecture (the definition in the textbook)?
1.22	What is the purpose of the ***Module Views*** as described in the textbook?
1.23	What is the purpose of the ***Component-and-Connector Views*** as described in the textbook?
1.24	What is the purpose of the ***Allocation Views*** as described in the textbook?
1.25	Explain why this statement is either true or false: "Every complex software system has a software architecture?"
1.26	What is the purpose of the ***Development View*** in the 4+1 view model?
1.27	What are the main influences on an architect designing a software architecture according to the textbook?
1.28	Name five important reasons for why software architecture is important.
1.29	Name the three main groups for availability tactics, and give one example of a specific tactic for every group.
1.30	Draw a simple class diagram to illustrate the ***Template Method*** design pattern.
1.31	What is an ***Architecturally Significant Requirement (ASR)/Architectural Driver***?
1.32	Name examples of three parameters that are commonly used in performance models according to the textbook.
1.33	What can be used to *analyze* the quality attribute ***security*** in a software architecture?
1.34	Describe the process of ***Attribute-Driven Design (ADD)*** in the textbook.
1.35	What is ***Architectural erosion***?
1.36	Describe three techniques to help keeping the code and the software architecture consistent as described in the textbook.
1.37	What is ***ATAM*** an abbreviation for?
1.38	Name five outputs of the ***ATAM***.
1.39	Briefly outline the process of ***reconstructing a software architecture***.
1.40	What must have a high priority when designing a software architecture for a ***software product line***?

References:

- Bass, Clements & Kazman: "Software Architecture in Practice", 3rd edition, Pearson, 2013.
- Philippe Kruchten, "Architectural Blueprints – The 4+1 View model of Software Architecture", IEEE Software 12 (6), 1995

Problem 2: Choose the most appropriate architectural pattern (5 points)

Nominees:

j) Layered
k) Broker
l) Model-View-Controller
m) Pipe-and-Filter
n) Client-Server
o) Peer-to-Peer
p) Service-Oriented
q) Publish-Subscribe
r) Map-Reduce
s) Multi-tier

Choose the *most appropriate architectural pattern* (one) for the 5 descriptions below. Motivate for your choice in one sentence (give reasons for choosing the pattern):

6. Need software to encode text files. The software consists of several algorithms of encryption, which can be combined for better result.
7. Need software for a digital speedometer for boats. The software should be flexible to support various number and types of displays (from simple LED-displays to large screens), and various ways of graphical designs of showing speed on the display.
8. Need software for a smartphone game where players can compete against each other through Bluetooth (no need to access the Internet).
9. Need software for managing and providing different kinds of information (weather, location, points-of-interest etc.) to app- and web-developers. The information will be gathered from various sources and processed by through the software. The software should provide open data access to anyone who wants to use the information through standard APIs.
10. Need software to very efficiently find the city in the world with a population above one million with the highest measured temperature. The temperatures measured from each city for 100 years are stored in separate files containing 36500 temperature entries. There are in total 457 cities with a population above 1 million.

Problem 3: Edge-dominant system (5 points)

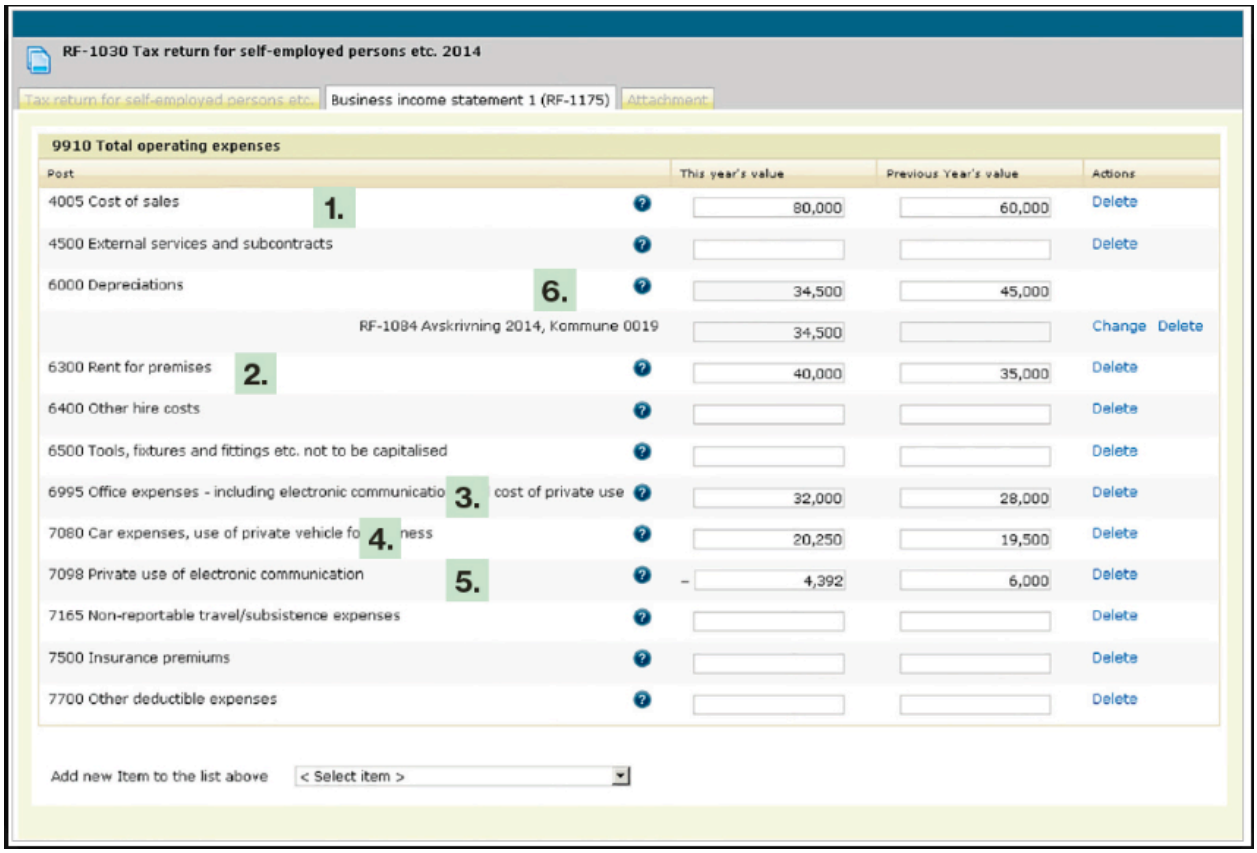
d) What is important in the design of the core architecture of an Edge-dominant system? (2 points)
e) What is the difference of open content systems and open source software? (2 points)
f) Give two examples of systems that can be classified as open content systems and two examples of systems that can be classified as open source software (1 point)

Problem 4: Cloud Architecture (4 points)

e) Explain the difference between the terms private cloud, public cloud, community cloud and hybrid cloud. (2 points)
f) What are the basic mechanisms needed to provide a cloud service described in the textbook? (2 points)

Problem 5: Quality Attribute Scenario (6 points)

System description: A web-based system for tax return (Selvangivelsen). In this system, the users will correct/change or add information related to their personal economy in order to compute how much tax they have to pay or they get refunded from the state.
The user interface of this system can typically look like the following:



d) Create one relevant quality attribute scenario on usability for the system described above. (2 points)
e) Create one relevant quality attribute scenario on modifiability for the system described above. (2 points)
f) Create one relevant quality attribute scenario on security for the system described above. (2 points)

Problem 6 Design a software architecture (30 points)

Read the description below and do an architectural design. Your answer must include:

f) Architectural Significant Requirements/Architectural Drivers – 2 points
g) Architectural tactics and patterns – 3 points
h) Process view – 8 points
i) Logical view – 14 points
j) Architectural rationale – 3 points

Motivate for your choices and state your assumptions.

Software for Conference Refereeing

Professional conferences are held in order to announce and discuss new results. The core activity of organizing a conference focuses on selecting the papers to be presented. Usually this is done by making an open invitation calling for papers to be submitted, circulating the submitted papers to a (geographically distributed) panel of reviewers, then selecting the best papers to appear on the conference program. A system to automate conference refereeing should do the following:

- The program committee announces "call for papers" by entering necessary conference information into the system. The program committee can add email addresses of potential authors or addresses from various mailing-lists into the system.
- Potential authors receive the call for papers by email from the system and decide to submit papers on their work. They write papers, and submit them using the system, where they add needed information as well as upload a PDF file of the paper. A given paper may have several authors, but there is only one corresponding author with one reply address. A notification is sent to the author after a successful submission has been made.
- The program committee consisting of several reviewers goes through a bidding process where they bid on the paper they want to review, and mark the papers they cannot review due to conflicts of interest.
- The system will then distribute the papers among the reviewers in the program committee.
- The reviewers will review the papers and enter their comments and score for each paper into the system.
- The system will rank the papers according to their scores and give the program committee an opportunity to mark the papers that will be accepted.
- When the program committee has decided on the accepted/rejected papers, they can through the system notify all the authors of the papers whether the papers got accepted or rejected.

The Conference Refereeing system can be accessed through a web-browser on PCs, tablets or smart phones. The users can also access the system using an app for tablets and smart phones. Typical customers of the system will be organizers of business conferences, research conferences and military conferences. For the latter, it is important that the papers submitted to the conference that can contain classified information should not be accessible for people outside military organizations.