HÖGSKOLAN
I SKÖVDE

Institutionen för informationsteknologi

TENTAMEN

Kurs **Software Engineering**

Examinationsmoment **Salstentamen**

Kurskod **IT398G** Högskolepoäng för examinationsmomentet 3hp

Datum **2026-03-23** Tentamenstid **14:15-18:30**

Ansvarig lärare Henrik Gustavsson

Berörda lärare Henrik Gustavsson, Jörgen Hansson

Hjälpmedel/bilagor

Övrigt

Grade points: 60 (10 for each of the six parts) Answer in Swedish or English. Answer all the questions. English questions are placed after Swedish questions and are otherwise identical. Each question has an equal weight of 10 marks. The final grade is calculated from these marks. To pass the exam, you must have at least 25 marks in total on all sections combined.

Anvisningar

- ☐ Ta nytt blad för varje lärare
- ☐ Ta nytt blad för varje ny fråga
- ☒ Skriv endast på en sida av papperet.
- ☒ Skriv namn och personnummer på samtliga inlämnade blad.
- ☒ Numrera lösbladen löpande.
- ☒ Använd inte röd penna.
- ☒ Markera med kryss på omslaget vilka uppgifter som är lösta.

Poänggränser

Skrivningsresultat bör offentliggöras inom 18 arbetsdagar

Lycka till!

Antal sidor totalt

HÖGSKOLAN
I SKÖVDE

Fråga 1: Systemmodellering och design (10p)

a) Med hjälp av exempel, diskutera vilka olika diagram som kan användas för att indikera vilka delar av en mjukvara som byggs av utvecklarna och vilka delar som byggs av externa aktörer. Du behöver inte rita diagram utan beskrivning räcker. (4p)
b) Med hjälp av ett exempel på ett minimalt diagram, beskriv hur ett state chart diagram kan användas för att informera/utforma testfall för en mjukvara. (3p)
c) Beskriv vad ett sequence diagram är, vad syftet med diagrammet är, samt diskutera vilken typ av system som sequence diagram är mest lämpligt för. (3p)

Question 1: System modelling and design (10p)

a) With the help of examples, describe which diagrams that can be used to indicate which parts of a software that is made by the developers and which parts that are built by external actors. You do not need to draw diagrams, a description of the diagrams is sufficient. (4p)

b) With the help of a minimal diagram describe how a state chart diagram can be used to inform/develop test cases for a software. (3p)

c) Describe what a sequence diagram is, what the purpose of a state chart diagram is, and discuss what types of systems that sequence diagrams are most suited for. (3p)

Fråga 2: Testning (10p)

a) Beskriv vad statisk verifiering är och diskutera hur AI-verktyg kan hjälpa till med detta inklusive risker med att använda AI för detta ändamål. (3p)

b) Förklara vad regressionstest är och ge ett exempel på ett fel som endast kan upptäckas av regressionstest. Beskriv vad testdriven utveckling är och diskutera hur testdriven utveckling relaterar till regressionstestning. (4p)

c) Beskriv vad acceptanstest är och diskutera i vilken miljö acceptanstest sker, samt kortfattat vilken typ av brister som kan hittas under ett acceptanstest men som är svåra att hitta under andra typer av testning (3p)

Question 2: Testing (10p)

a) Describe what static verification is and discuss how AI tools can help with this including risks associated with using AI for this purpose. (3p)
b) Explain what a regression test is, and give an example of a fault that can only be found by a regression test. Outline the main characteristics of test driven development and discuss how test driven development relates to regression tests (4p)
c) Describe what an acceptance test is and discuss the environment of an acceptance test. Furthermore, give a short outline of the kinds of defects that can be found during an acceptance test but are hard to find during other types of testing. (3p)

Fråga 3: Verktygsstöd, Evolution och Internet-mjukvara (10p)

a) Change management är en viktig funktion i moderna utvecklingsmiljöer. Beskriv kortfattat vad change management är och varför change management är viktigt. Beskriv med hjälp av ett exempel hur change management relaterar till evolutionens olika faser inklusive utvecklingsfasen. (5p)

HÖGSKOLAN
I SKÖVDE

b) Det finns de som säger att Service Oriented Software Engineering är lika viktigt som skiftet när Objektorienterad utveckling introducerades. Beskriv hur service-oriented software engineering påverkar utvecklingsprocessen och implementationen av mjukvara. Diskutera hur det påverkar valet av programmeringsspråk för ett projekt, och kopplingen till legacy software. (5p)

Question 3: Tool Support, Evolution and Internet software (10p)

a) Change management is an important function of modern development environments. Give a short outline of why change management is important. With the help of an example describe how change management relates to each of the phases of software evolution including the development phase. (5p)
b) There are people that say that the shift towards Service Oriented Software engineering is as important as the shift that happened when Object Oriented Development was introduced. Describe how service-oriented software engineering affects the development process and the implementation of software. Discuss how this affects the choice of programming language for a project and the connection to legacy software. (5p)

Fråga 4: Arkitektur vs agil utveckling (10p)

a) Beskriv grundprinciperna för agil utveckling där du också analyserar dess styrkor och svagheter. (5p)

b) Agila metoder förespråkar ofta "emergent design" att arkitekturen växer fram, medan storskaliga system ofta kräver en stabil arkitektur för att undvika teknisk skuld. Diskutera konflikten mellan agil leveranshastighet och behovet av arkitektonisk stabilitet. Hur kan man enligt Sommerville balansera dessa motpoler i ett projekt med höga pålitlighetsskrav? Delar du hans syn? Vilka risker ser du? (5p)

Question 4: Architecture vs Agile Development (10p)

a) Describe the core/main principles of agile development, where you also analyze its strengths and weaknesses. (5p)

b) Agile methods often advocate for emergent design where the architecture evolves gradually, while large-scale systems often require a stable architecture to avoid technical debt. Discuss the conflict between agile delivery speed and the need for architectural stability. According to Sommerville, how can you balance these opposing forces in a project with high dependability requirements? Do you share his view? What risks do you see? (5p)

Fråga 5: Arkitektoniska avvägningar (Trade-offs) (10p)

a) Du ska designa ett distribuerat system för realtidsövervakning av sensordata. Analysera för- och nackdelar med att använda en *Pipe-and-Filter*-arkitektur, *Repository*-arkitektur kontra en *Client-Server*-arkitektur. Fokusera på aspekter som skalbarhet, svarstider (eng. latency) och systemets feltolerans. (10p)

Question 5: Architectural Trade-offs (10p)

a) You are to design a distributed system for real-time monitoring of sensor data. Analyze the advantages and disadvantages of using a pipe-and-filter architecture, repository architecture versus a client-server architecture. Focus on aspects such as scalability, response times (latency), and the system's fault tolerance. (10p)

HÖGSKOLAN
I SKÖVDE

Fråga 6: Pålitliga System (10p)

a) Förklara skillnaden mellan hårdvaruredundans och mjukvarudiversitet (t.ex. N-version programming). Varför anses mjukvarudiversitet vara mycket svårare att uppnå i praktiken än hårdvaruredundans, och vilka är riskerna med att använda identiska kopior av mjukvara i ett feltolerant system? (4p)

b) Ett system har en hög tillförlitlighet (eng. reliability) men låg tillgänglighet (eng. availability). Ge ett exempel på ett scenario där detta kan ske. Diskutera hur valet av felhanteringsstrategi (t.ex. "fail-soft" vs. "fail-safe") påverkar dessa två attribut/icke-funktionella beteenden. (4p)

c) Vilket av följande påståenden beskriver bäst den fundamentala skillnaden mellan 'Availability' (tillgänglighet) och 'Reliability' (tillförlitlighet)? (1p)

1) Availability fokuserar enbart på hårdvarans drifttid, medan Reliability fokuserar på mjukvarans korrekthet.
2) Reliability mäts i MTTR (Mean Time To Repair) medan Availability mäts i MTTF (Mean Time To Failure).
3) Reliability handlar om sannolikheten för felfri drift under en viss tid, medan Availability är andelen tid systemet är redo att leverera tjänster.
4) Det finns ingen praktisk skillnad mellan begreppen i moderna distribuerade system.

d) Varför anses mjukvarudiversitet (eng. software diversity) ofta vara en problematisk och ibland otillräcklig strategi för att uppnå högsta möjliga 'Dependability' i komplexa system? (1p)

1) Det är tekniskt omöjligt att garantera att de olika mjukvaruversionerna inte påverkar varandras minnesutrymmen under exekvering.
2) Oberoende team tenderar att göra liknande logiska fel på grund av gemensamma kognitiva mönster och otydliga specifikationer (eng. common-mode failures).
3) Röstningsalgoritmer kräver att alla versioner är exakt funktionellt identiska, vilket förhindrar användningen av agila utvecklingsmetoder.
4) Underhållskostnaden blir oproportionerligt hög eftersom varje buggfix måste verifieras och portas till samtliga versioner utan att introducera nya divergensfel.
5) Diversitet adresserar endast slumpmässiga hårdvarufel och är därför verkningslöst mot systematiska mjukvarufel.
6) Komplexiteten i att designa en pålitlig 'decider' (enheten som väljer rätt svar) blir ofta högre än komplexiteten i själva applikationen, vilket skapar en ny 'Single Point of Failure'.
7) Användningen av olika programmeringsspråk och kompilatorer introducerar 'compiler-induced bugs' som är omöjliga att spåra i ett distribuerat system.

Question 6: Dependable Systems(10p)

a) Explain the difference between hardware redundancy and software diversity (e.g., N-version redundancy by programming). Why is software diversity considered much harder to achieve in practice than hardware redundancy, and what are the risks of using identical copies of software in a fault-tolerant system? (4p)

b) A system has high reliability but low availability. Give an example of a scenario where this can occur. Discuss how the choice of error handling strategy (e.g., fail-soft vs. fail-safe) affects these two attributes/non-functional behaviors. (4p)

c) Which of the following statements best describes the fundamental difference between 'Availability' (availability) and 'Reliability' (reliability)? (1p)

1) Availability focuses solely on hardware uptime, while reliability focuses on software correctness.
2) Reliability is measured in MTTR (Mean-Time-To-Repair) while availability is measured in MTTF (Mean-Time-To-Failure).
3) Reliability is about the probability of fault-free operation during a certain period, while availability is the proportion of time the system is ready to deliver services.
4) There is no practical difference between the concepts in modern distributed systems.

d) Why is software diversity often considered a problematic and sometimes insufficient strategy for achieving the highest possible dependability in complex systems? (1p)

1) It is technically impossible to guarantee that different software versions do not affect each other's memory space during execution.
2) Independent teams tend to make similar logical errors due to common cognitive patterns and unclear specifications (common-mode failures).
3) Voting algorithms (voting) require all versions to be exactly functionally identical, which prevents the use of agile development methods.
4) Maintenance cost becomes disproportionately high because every bug fix must be verified and ported to all versions without introducing new divergence errors.
5) Diversity only addresses random hardware errors and is therefore ineffective against systematic software errors.
6) The complexity of designing a reliable 'decider' (the voter unit that selects the correct answer) often becomes higher than the complexity of the application itself, creating a new single point of failure.
7) The use of different programming languages and compilers introduces 'Compiler-induced bugs' that are impossible to trace in a distributed system.