DAT321/DIT847:
Software Quality

Welcome to the examination for the *Software Quality*! The examination is intended to last for max **4 hours** and is intended to be **anonymous** (i.e., the teacher grading your exam will not know your name). Therefore, it is important that you follow the instructions (in the separate exam cover sheet) and **do NOT leave any information that would reveal your name on these pages.**

Each question has a number of points assigned shown in the square brackets. When the question is broken down into smaller sub-questions the part of the points for that specific sub-questions are also shown as following:

1. *[10 pts].*
    a. *[2 pts]*
    b. *[8 pts]*

The percentage of points and the corresponding grade is presented below (100 points in total):

% of points	DAT321	DIT847
[0, 50%)	2	U
[50%, 65%)	3	G
[65%, 85%)	4	G
[85%, 100%]	5	VG

It is important that you write **clearly** so that the examiner can read your answer. If your handwriting is unreadable, then you will not get any points for that question. We will NOT assess grammar or spelling as long as your answer is readable, understandable and unambiguous.

The questions in this exam refer to the **ISO 25010:2011** that categorises internal and external software quality attributes into eight characteristics.

You are allowed to have the English ←→ Swedish dictionary during the examination.
CTH/GU approved calculators are also allowed, but NOT calculators in mobile phones.

Questions about the exam:
Contact: Francisco Gomes, tel. 031 772 6951, gomesf@chalmers.se
First visit: around 15:15
Second visit: around 17:10

The exam review is scheduled for 2018-11-15, between 13:30 – 14:30 at Jupiter building, 4th floor, Room 424.

Examination date: 2018-11-01

The questions in this exam are related to the following context. Note that the answers should, in turn, be justified based on i) the theory and terminology from software quality and ii) their connections to the elements in this context (team, tools, processes, stakeholders, etc.).

Description of the context:
You've become part of a team responsible for developing software components that will be part of a car. The components developed by your team will interact with mechanical parts of the car and will have significant roles in controlling essential parts of the vehicle. The project you are assigned to needs to develop a component that will use information from various sensors in the car to assist driving of the vehicle (by notifying drivers of objects around), as well as monitoring speed of the car and fuel consumption. Failures in the sensor, or even delays in transmitting data can lead to severe consequences to safety.

Your team is composed of software and mechanical engineers that never worked together before. Additionally, some of those engineers have only worked with traditional software development where most of the planning is done upfront, and verification and validation (V&V) activities are done closer to releases. On the other hand, the company invested in state of the practice tools for traceability and configuration management, such that all artefacts and documents produced by your team is under version control.

Another important factor of your project is that one of the components that your team is developing is highly dependent to an existing legacy component, where very little testing exists. For now, your development process has to use the legacy component, but the company's expectation is that the legacy component will be replaced by a newer one. Keep in mind that your component will be part of a larger system with distributed components and access to online applications hosted on the cloud.

1. **[20 pts]** Using the context above, answer the following questions:
   a. [10 pts] Describe two software product quality's characteristics, **and** provide an example on why they important for the component in our context (i.e., the component that communicates with the sensors in the car).
   b. [5 pts] Explain the differences between analyzing product quality and quality in use.
   c. [5 pts] Provide one example of quality in use characteristics using the provided context. Feel free to use other software components of the car in your example.

1.a. Any 2 of the below work. The best options for this context are reliability and performance efficiency.

Reliability: Degree to which the product **performs** specified functions **under specified conditions** for a **specified period of time**.

- Example: Anything regarding the component and that its related to the sub-characteristics of reliability, such as: availability, fault tolerance, recoverability.

Maintainability: Degree to which the product can be modified by the intended maintainers.

- Example: Anything related to the consequences of changing a component and connected to one of the sub-characteristics: testability, modularity, reusability, etc.

Functional Suitability: Degree to which the product **meet stated and implied needs** when used under specified conditions

- Example: Anything related to the component's capability of covering all specified behavior and task (completeness), and doing it correctly (correctness).

Usability: Degree to which the product can be **used by specified users** with satisfaction in a specified context of use.

- Example: Components related to user interaction (e.g., infotainment systems)

Performance Efficiency: Represents the performance **relative** to the **amount of resources** used under **stated conditions.**

- Example: Anything related to capacity or time behavior, such as delays in receiving / processing sensor information can trigger faults in the component.

Portability: Degree to which the product can be **transferred** from one hardware to another

- Example: Anything related to how easy it is to re-use components across different cars, or even different variations of the car (e.g., with different hardware / mechanical configurations). How easy it is to, e.g., install / replace the software components in the car.

Compatibility: Degree to which the product **can exchange information;** Performs its required functions, while **sharing** the same hardware or software environment.

- Example: Anything related to how the component shares information (i.e., can communicate) with other components of this and other systems (e.g., REST APIs).

Security: Degree to which the product **protects information and data** so that other have data access appropriate to their authorization level

- Example: Anything related to the component's capability of guaranteed authenticity of access to data, having integrity so that unauthorized access is denied, and, e.g., protecting user's data.

1.b.

Product quality includes properties (both the static artefacts and the software under execution) related to the software product, component, system or subsystem itself. In turn, Quality in Use covers properties related to the product being executed (e.g., with its end-user) in a specific context (e.g., the **real** use of the software). The important distinction between them is the context, since quality in use cannot be evaluated without a context associated to the scenario where the software is being used.

1.c.

All quality in use (QiU) examples for the component should involve the car itself, or a user interacting with the component while the car (or service in the car) is being used outside development situations.

There are 5 characteristics for QiU. Any of the below would work.

Effectiveness: Accuracy and completeness with which **users** achieve specified goals.

Efficiency: Resources expended in relation to the accuracy and completeness with which users achieve goals

Satisfaction: degree to which a user's needs are satisfied when a product or system is used in a specified context of use. Can be measured in terms of comfort, usefulness, pleasure.

Freedom of Risk: degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment. For our components described, some safety scenarios can be used as examples for this characteristic.

Context coverage: degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified. This is related how the product is being used considering: i) all of the other 4 characteristics, and ii) its relation to the current context being analyzed. Note that not all contexts will focus on all quality characteristics, such that there is a many-to-many relationship between the characteristics and the contexts where the product can be used. An example here would be to analyse the component in more than one context (you would have to describe more than one context, and state which QiU characteristics would be covered in each context).

2. **[15 pts]** Using your knowledge on software quality measures, answer the following:
   a. [5 pts] McCabe and Henry and Kafura are two distinct complexity measures used to in software quality. Are they used for internal or external quality? Justify your answer.

b. [10 pts] What are the differences between both measures in terms of software complexity?

2a. Both are used for Internal Quality, since they are related to static versions of the software artefacts (e.g., the source code). As a consequence, they are useful for developers to gather current quality information and decide whether they should refactor the involved artefacts.

2b. McCabe: Also known as cyclomatic complexity, evaluates complexity in the **control flow**, using different programming constructs (e.g., if/else, whiles, for, switch/case, etc.) and how they are nested. Henry-Kafura is predominantly based on size and the fan-in and fan-out measures, which convey the code's **dependency or modularity**.

3. **[15 pts]** Considering sustainable software engineering, and using our automotive context, provide an example about how we can assess one (or more) software product quality characteristic for at least two distinct dimensions of sustainability.

You should choose one (or more) of the eight characteristics decided above. For the environmental one you can provide examples on resource optimization (e.g., optimally using resources from the car), or energy consumption (stating a scenario where components are design to consume less electricity from the battery). Other options is combining usability (e.g., accessibility) of the car's user interface for disabled people, such as interfaces design for people with visual / hearing impairments. This example is included in the social dimension. Compatibility and portability are good examples of characteristics for the technical dimensions where changes in their environment should not affect the product, specially at longer term. Performance efficiency, reliability and security can be used as examples for the economic dimension.

4. **[15 pts]** Using your knowledge on software testing and continuous integration, answer:
   a. [5 pts] Considering the V-model, explain the differences between each level of testing.
   b. [10 pts] Your colleague Alice suggests that the team should adopt Continuous Integration (CI) in your project. Anna disagrees with Alice saying that this project is not suitable for CI. Do you agree with Anna or Alice? Justify your answer using the information from the context of the project.

4.a. There are several levels of testing, but mainly you need to discuss 4 different ones:

Unit testing is the **lowest level** and is **closer to the code** artefact.

Integration testing focuses on testing the integration **between different units** in your system / sub-system. While unit testing focuses on a unit specifically, many faults can be triggered when two or more units are working together, e.g., exchanging information.

System level testing focuses on testing of the components their executing environment, including external dependencies, and often involve more than just software components (e.g., involved hardware or mechanical components).

Acceptance level testing focuses on testing whether the product fulfils the needs determined by the customer. They are the highest level of testing and are close to the requirements level.

4.b. You should **agree with Anna**, the project, for now, is **not suitable**. Even though the example states that there is tool support (traceability + configuration management), the **CI practices needed in a team are not indicated in the text**. In fact, the description indicates otherwise, since the team is used to doing "most of the planning is done upfront, and **verification and validation (V&V) activities are done closer to releases"** upfront planning and specifically." Therefore, the team is not ready to adopt continuous integration, since they need to have the discipline of testing constantly, and not only close to release.

5. **[20 pts]** When diagnosing a Markov chain, we look for three diagnostics to form an opinion of how well things have gone. Which three diagnostics do we use, **and** what do they tell us?

   Look for hairy caterpillars to see that the chains mixed well. For convergence you check the R-hat, which should be 1.00, and lastly, the effective sample size should be higher than at least 10-20% of the total sample.

6. **[15 pts]** Even though the legacy component of your project lacks tests, one engineer has logged, for each class in the source code, the McCabe complexity value and the corresponding lines of code. In order to make more informed decisions on maintenance of your legacy component, Alice suggests we use Bayesian Data Analysis and the data from three years maintenance to fit a linear regression predicting a class's McCabe complexity using its lines of code as a predictor.

Write down the *mathematical model definition* for this regression using *any* variable names and priors of your choice. Defend your choice of priors and remember to state your assumptions regarding your model!

This is a more open-ended problem. But perhaps the simplest model structure that addresses the problem would be:

$c \sim \text{Poisson}(\lambda)$

$\log(\lambda) = \alpha + \beta y_i$

$\alpha \sim \text{Normal}(0, 10)$

$\beta \sim \text{Normal}(0, 10)$

$\sigma \sim \text{Cauchy}(0, 50)$

where c is the cyclomatic complexity, and λ can be the rate of changes in the class. Poisson because it cannot be negative, and the complexity can changed based on how often you are changing the code (e.g., more complex code = more refactoring). Alternatively, you could use other likelihood but you must justify how it connects to the SQ metric described in the problem.

These priors aren't great (too wide), but they'll do. The prior on the intercept α is effectively uninformative and very wide. The prior on β is very weak, centered on zero, which corresponds to no impact of the amount of lines changed (alternatively, you can say that the lines of code have an impact and then change the prior).

There are other options, but the goal is for your to defend/justify your choice of model in connection to the problem presented.