DAT321/DIT847:
Software Quality

Welcome to the examination for the *Software Quality* course. 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.**

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%)	U	U
[50%, 65%)	3	G
[65%, 85%)	4	G
[85%, 100%]	5	VG

Write your answers in the exam sheets. Before handing in your exam, number and sort the sheets in task order. Write your **anonymous code** and page number on every page!

It is important that you write **clearly** so that the examiner can read you. 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.

Simple calculators are also allowed, but NOT calculators in mobile phones.

Questions about the exam contact:
Richard Torkar, tel. 031 772 57 19, richard.torkar@cse.gu.se
Francisco Gomes, tel. 031 772 69 51, francisco.gomes@cse.gu.se

The exam review is scheduled for 2019-11-29, between 13:30 – 15:30 at Jupiter building, 4th floor, Room 427.

The questions in this exam are related to the context below. Note that you must justify your answers with i) the theory and terminology from software quality and ii) their connections to the elements in this context (e.g., teams, product, processes, architecture, customers, etc.).

Context: You are hired by GPS-Co, a company that develops a cloud-based system that offers various Global Positioning System (GPS) services. You are the quality assurance manager for teams B and C (see description of the teams below).

Software product:
The product is composed of several services, all of them related to GPS positioning and navigation. The product uses a microservices architecture, where the **different services are loosely coupled and independently deployable**. There is no graphical user interface to use or test the product, since the services are accessed and integrated via APIs endpoints (i.e., entry points that work as method calls). The company requires all teams to use Continuous Integration practices and tools.

Customers and business:
Currently GPS-Co has contracts with other companies that use the services provided by your software product. Currently, your main customer is the FoodDelivery-Co, which delivers food orders from supermarkets to customers. FoodDelivery-Co uses the GPS services on their fleet of trucks that deliver all food orders from supermarkets to customers who ordered online.

Today, GPS-Co signed a contract with another customer, Automotive-Co which is an automotive company that wants to use the GPS services for track and navigation of its manufactured cars. Therefore, as part of the contract, Automotive-Co will request several new features from your software product.

Teams and software development process:
Team A is responsible for developing and maintaining the main features of your software product: i) the main application with the various **GPS features**, ii) the c**loud infrastructure and platform where the service is hosted**, iii) all **non-functional aspects** of the product.

Team A has 50 members and is internally divided into small groups based on specific the services provided by the software product. Team A is composed of experienced engineers with various testing and development skills.

Team B handles the development of customised features for FoodDelivery-Co, and is a small team with 8 people, all of them with wide experience in software development and agile software development. Team B reuses the features created by Team A in order to customize the product for FoodDelivery-Co.

GPS-Co is also hiring developers and testers for Team C that will work similarly to Team A but for Automotive-Co. Therefore, you are likely to help with recruitment for this team.

1. **[20 pts]** Using the context above, answer the following questions:
   a. [15 pts] Choose three software product quality's characteristics, **and** provide examples of why they are relevant to the software product at GPS-Co.
   b. [5 pts] Explain the differences between internal and external quality.

Any of the eight below could work. Other examples could be provided, but it is important to clearly state the relevance.

Usability: They are relevant to the product from the end-user's perspective. Particularly, drivers from the truck fleet or future owners of cars from Automotive-Co, will use the GPS for navigation or positioning information.

Maintainability: They are relevant to the product because modification and change requests are common in any software project, hence software should be easy to maintain. For instance, Teams B and C will need to implement and/or request features to Team A such that maintaining the parts of the product should not compromise development and functioning of the product.

Reliability: They are relevant to the product because the product is available for drivers on the road, and other users that require information live (e.g., for the navigation functionality). For drivers, poor reliability can compromise safety since two of the customers use the product for driving.

Security: They are relevant because the product stores and processes personal information (e.g., current location of vehicles, or users). Poor security can lead even to safety and privacy issues with the users of the product.

Compatibility: They are specially relevant because our product is based on API, such that we microservices (and external clients of our application, such as the trucks), must be able to share information (e.g., navigation and positioning). For instance, in case we want our product to talk with other APIs to offer, for instance, weather forecast of specific places, compatibility becomes an important quality characteristic.

Portability: They are relevant if we consider that the product is accessed by hardware devices such as mobile phones from users, or even embedded in the trucks or cars of our customers. Therefore, the product must be able to work for various versions of hardware platforms.
Functional suitability: They are relevant because the product must meet the stated and implied needs of the users, such as the requests for customized features from our customers, or even the basics features expected from a GPS software system (e.g., navigation and positioning).

Performance efficiency: They are very relevant because the product runs on a cloud infrastructure that requires computational resources which affect the usage and functioning of our product. Particularly, GPS features are highly connected to information updated live (real-time), hence latency and limited capacity of resources can significantly affect our product.

1b. Both address the software product, and the difference is that internal quality targets the quality of artefacts of the software product, such as code, test specification, requirements, etc.; in turn, external quality targets the behavior of the product such that it is connected to attributes collected when executing/running the software product.

2. **[15 pts]** Using your knowledge on software quality measures, answer the following:
   a. [10 pts] Describe how you can use Henry and Kafura to improve the quality of your software product. Your description must: i) include the artefact and attribute under investigation, and ii) justify why Henry and Kafura is the right choice.
   b. [5 pts] Your colleague Martin suggests measuring only the lines of code of the software system, since other quality measures can be complicated to understand and difficult to monitor. Do you agree with Martin? Justify your answer.

2a. We can use H&K to evaluate and improve the quality of our system's modularity. In other words, we can i) investigate the dependencies between the different microservices, depending on the API calls made between one another. H&K is the best choice because ii) it is based on fan in and fan out measures that verify the parts of the software, respectively, invoking the code under investigation and which other parts the code is invoking.
Note: instead of API, you can use as 'i' dependencies within classes of each service, or package. The focus is the dependencies/modularity between components/parts of your system.

2b. Disagree. Lines of code by itself is not meaningful, since it only conveys the size, and not the size in relation to other attribute (e.g., complexity, modularity, changes).

3. **[20 pts]** Considering the Acceptance, System, Integration and Unit levels of the V-model, explain how you will do testing at GPS-Co products. Your answer must contain, **for each level of testing**: i) what is being tested, ii) the goal of the test, and iii) relevant aspects that a tester should consider.

For acceptance testing, i) we test the product as requested by the customer. Or, a deployed customized version including all of its dependencies (external services, hardware components, etc.). You could also list specific test cases, such as:

- the product being accessed by the delivery trucks for FoodDelivery-Co;
- specific delivery routes provided by the GPS system for FoodDelivery-Co;
- the navigation between two specific addresses while using the car.

ii) The goal is to check whether the product meets the customers/user needs. So, the testing should be described from the perspective of the customers (FoodDelivery-Co, Automotive-Co or another end user of the product).

iii) relevant aspects to consider are the customer requirements that need to be validated, so examples could be the customized features asked by either FoodDelivery-Co or Automotive-Co.

System level:

i) we test the software product and its features while running in the production environment, so examples could be testing navigation features, or searches for addresses or route in the map, etc. ii) the goal is to verify whether the product works while running on the production environment, i.e., the cloud infrastructure; iii) relevant aspects to consider are the different dependencies of the system, such as the reliability of the cloud infrastructure, the connection to hardware devices (e.g., if you are using system testing on the trucks from FoodDelivery-Co, or the product deployed in the car as Automotive-Co).

Integration level:

i) we test the communication between the different parts of the system, in our case, the different microservices. ii) the goal is to check the version of the product after using several

microservices into a single functionality or checking the integration of the different components of the system; iii) the relevant aspect to consider is to run the test after building the system, such as in continuous integration servers (i.e., each microservice is built and available for testing).

Unit level:
i) we test the return of a single unit of work, such as a class or method of a specific microservice; ii) the goal is to test the unit of work in isolations, as opposed to being integrated with other parts (e.g., components, packages). iii) relevant aspects to consider could be to have a xUnit framework to automatically run or support the test, to mock the dependencies of the test, such as other services, or database state, etc.

4. **[15 pts]** GPS-Co wants to contribute with sustainability in ICT. You must propose or explain two sustainable features of your software product. Your explanation must include: i) description of the feature and ii) how it is connected to a sustainability dimension.

   Note: Here, you can interpret features as: new functionalities/services of your product, updates to the development process, or changes in the current software system.

A few examples:
Example 1:
i) optimise performance efficiency to reduce energy consumption of the cloud infrastructure. ii) this connects to different dimensions depending on the perspective explained. For instance: the environment dimension for better energy efficiency; technical dimension for enabling long term usage and evolution of the product; economic if you consider the costs involved of using services in a cloud infrastructure (e.g., data traffic, Amazon Web Services, etc.).

Example 2:
i) Suggest optimised truck routes to FoodDelivery-Co make several deliveries within a specific time. ii) Again, could be more than one dimension: For instance: environment for less CO2 emission of trucks with more optimised route; economic if that could affect the business model with FoodDelivery-Co;

Example 3:

i) add feature for sharing routes for social activities (e.g., hitchhiking to work, tracking for hiking or biking). ii) connected to the social dimension since it allows to bring groups of people with the same interest;

Example 4:

i) add a user interface with support for voice recognition and text-to-speech, or other accessibility features. ii) connects to individual/social dimension since it is inclusive.

5. **[10 pts]** Using your knowledge on Bayesian Data Analysis (BDA):

   c. [5 pts] Explain what the four main reasons are to use multilevel models.

   The reasons to use MLM are:

   1. To adjust for repeated sampling. When >1 observation comes from the same individual, time, location, etc. If we use a single level model we're overestimating the variance maximum.
   2. To adjust the estimates for imbalance in sampling. If some individuals etc. are sampled more often that could also mislead us (very often in unbalanced samples we're overestimating where n_subx < avg(n_sub)).
   3. To study variation because MLMs model variation explicitly. When dealing with multilevel aspects variance gives us yet another view
   4. To avoid averaging because we use raw data and do not average data before using them in a model. MLMs allows us to use original data and thus preserve uncertainty (avg we can use to make predictions later down the road).

   Other than above, missing data models are per definition MLMs. If you want to deal with measurement errors in a principled way then… MLMs. Factor analysis is a MLM. In short, if we want to avoid overfitting we need to employ MLMs.

   d. [5 pts] We have run two models with one predictor each, estimating bc and bf, and we see that they both have predictive power. Of course, adding both of the predictors to one model should make things even better, so let's do that and check the output.

   Can you explain **why bf** helps or does not help with prediction?

	Mean	**Std. Dev**	**Lower 0.89**	**Upper 0.89**	**N. Eff.**	**Rhat**
a	0.43	0.16	0.19	0.68	2733	1.00
bc	1.14	0.20	0.82	1.46	2650	1.00
bf	0.00	0.10	-0.16	0.16	3259	1.00

bf has *no* predictive pwr here: -.16;.16. Rhat and ess can still be ok even though the sample has collinearity in variables. One take away message here is that if bf is *needed* for predictions, i.e., we truly are interested in estimating, for example the contrast, of the parameter, then we could include it; MCMC handles this well compared to freq approaches. But if we get good out of sample predictions then it's very questionable to include bf since LOO/WAIC etc. will anyway punish the model due to having more effective parameters in the model.

6. **[20 pts]** In order to properly plan for the upcoming weeks of development, you want to be able to predict the number of faults that can be introduced based on the code churn of each code commit. Note that the faults and code churn can vary depending on the service being changed, because each service should be loosely independent from each other. You are able to collect data from the past two years of modifications (example shown below):

VersionID	Faults	Code Churn	Service
1	5	45	Service A
1	7	35	Service B
2	10	-20	Service A
2	2	-5	Service C
…			

Write down the *mathematical model definition* for this prediction using *any* variable names and priors of your choice. State the ontological and epistemological reasons for your likelihood. Remember to clearly state and justify the choices and assumptions regarding your model.

We choose a Poisson likelihood because fault is a positive number and tend to come with the rate of changes (i.e., connected to modifications in the code), hence the connection to code churn. Alternatively, the differences in faults can also be connected to the specific service being changed, or even specific version IDs. You should have code churn as main effect and service (or VersionID, depending on the argument) as varying intercepts. In any case, not having a varying intercept would be risky because:

- Note that each service may have specific patterns related to their own code churn (i.e., the context of specific services can affect adding/removing more code than others, such as services of different sizes, or composing different features).
- Since the product evolves over the years, one can expect that specific versions can include more code churn variations than others (for instance, if we compare perhaps the

churn between the versions v1.0, v2.0, against the v1.1.1, v1.1.2), so you can avoid overfitting to the grand mean by using version id as well as the varying intercept.

Since this is an MLM, you need to state your hyperparameters, such as choosing Normal(0,1) for the varying intercepts. Nonetheless, having a grand mean (the grand alpha set to N(0,5) would be okay. Ultimately, the complete mathematical model is:

$$faults \sim Poission(\lambda_i)$$
$$\lambda_i = \alpha + \beta_c * CodeChurn + \alpha_{Service[i]} + \alpha_{Version[i]}$$
$$\alpha \sim Normal(0, 5)$$
$$\beta_c \sim Normal(0,1)$$
$$\alpha_{service[i]} \sim Normal(0,1)$$
$$\alpha_{Version[i]} \sim Normal(0,1)$$

Now, for the hyperparameters/priors of the varying intercepts:

$$\alpha_{Service} \sim Normal(\mu_s, \sigma_s)$$
$$\mu_s \sim Normal(0,1)$$
$$\sigma_s \sim Cauchy(0,2)$$

You could have used only one of the varying intercepts (not necessarily both). You would lose point by: i) not using faults as the posterior and code churn as the main predictor; ii) your code churn (β_c) must have a normal likelihood, since it can be negative; iii) not using a general linear model; iv) not using **any** varying intercepts; v)forgetting priors and parameters; vi) wrong explanations such as wrong justifications for the likelihood, priors, etc.