CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn		*
DIT 635	Software Quality and Testing		X
Anonymous code / Anonym kod		Examination date / Tentamensdatum	Number of pages / Antal blad	Grade / Betyg
G02		21.03.2019.	14	VG

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under examinationen.

Solved task / Behandlade uppgifter — No/nr		Points per task / Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. / Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	✓	4,5	
2	✓	9	
3	✓	9,5	
4	✓	9	
5	✓	5	
6	✓	3,5	
7	✓	13	
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus credits/ poäng			
Total examination points / Summa poäng på tentamen		53,5	

5

①

a) Software verification is the proces to make sure that the ~~so~~ implementation fulfills the requirements. 1 ✓

Software validation is the proces that ~~helps~~ is making sure that the implementation fulfills the expectations of the customer. Therefore it can shed light on misunderstood, missing ~~or~~ incorrect requirements. 0,5 (↑ still all stakeholders)

In the begining of the testing - which can happen as soon as the implementation (stub) verification is prioritized. According to the V-model, the project starts with ~~Gap~~ capturing the requirements then specify the implementation, ~~then~~ After the code is implemented ~~the~~ unit testing, later on integration and system testing are happening that ~~are~~ belong to the verification group. The end of system testing and acceptance testing ~~test~~ targets only validation. 1 V

b) Fault is the static presence of a problem. It can be noticed with inspection and code review and ~~many~~ or ~~may~~ not cause failure depending on the input. ~~For~~ V 1

~~Failure~~ Failure is the incorrect behaviour of a software ~~and~~ ~~is~~ that is caused by one or more faults in the code. Example on the next page. V 1

5

① Example for 1b)

~~int a~~ int[] a = ~~int~~ new int[6]

{ user input values to the integer array }

```
for(i = 1 ; i < 6 ; i++){
    if(a[i] == 0){
        System.out.println("Found zero");
        break;
    }
}
```

~~System.out.println~~ Supposed to

The code piece above prints "Found zero" if any element of the array a is zero. However there is a fault in the ~~if statement~~ for loop so that it ~~never~~ does not look at the first element of the array (index 0).

It will cause failure if the only zero is at index 0, because the code will have incorrect behaviour; that is it will not show the presence of the 0 value in the array by printing the "found" statement.

5

(2) a)

Internal quality ~~is a static content of~~ is the quality of the static code, such as the number of lines of code that has effect on maintainability. 1

External quality is the quality of the behaviour of the code. It can be seen ~~during~~ via testing ~~as~~ for example as # of passed test cases. 1

Quality in use is the quality perceived by the user. ~~and~~ The ~~or~~ user-system interactions is in question, ~~how it~~ for example ~~the~~ how fast a user found a vital functionality. 0,5

Ok, but quality-in-use is mostly about ~~the~~ use in context/environment

Internal and external quality ~~is~~ can be seen as product quality. ~~For example~~ ~~maintainability~~ ~~is a quality such~~ can be assessed with the number of interfaces and their error-~~rate~~ frequency. 0,5

~~Quality in use~~

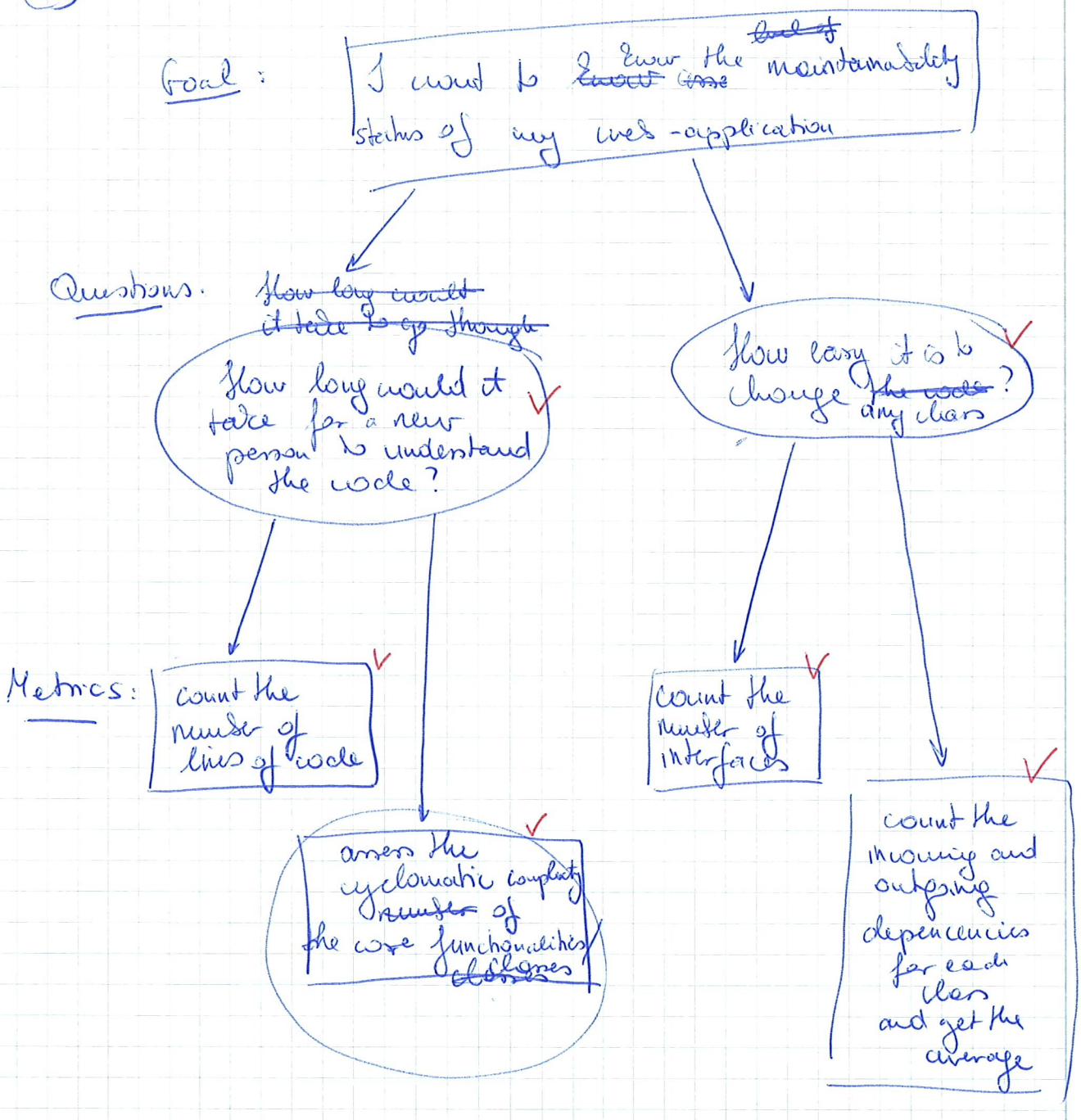
Another example for quality-in-use is if the user is satisfied with the speed of ~~an~~ getting the output from a database. 1

that's a metric what is the quality?

Ok, speed = performance

5

2b)



The goal of the task is to asses maintainability. ~~That can be divided divided into two main subjects: two~~

To do so questions are specified that is leading the problem closer to solution, making it less abstract. There two questions are asked. The metrics ~~asse~~ addresses the questions by finding an appropriate measurement to ~~ans~~ quantify the problem.

very good

CHALMERS | Anonymous code / Anonym kod: 602 | Points for question (to be filled in by teacher) / Poäng pa uppgiften (ifylles av lärare): 3 | Consecutive page no. / Löpande sid nr 5 | Question no. / Uppgift nr 3

③ a)

I) lines of code – ratio ✓

II) type of class – nominal ✓

III) architect opinion – ordinal ✓

1,5

I) The lines of code is a number, an integer that can be divided, multiplied ✓, etc. so it is reasonable to ask questions such as "is one code twice as long as the other?" The distances between numbers are equal so basic mathematics works, thus it is on ratio scale. ✓ 0,5

II) "model", "view" and "control" are a nominal scale because they are purely labels, one cannot ✓ order them and cannot decide distances between them. 0,5

III) "easy", "complex" and "very complex" are labels ~~with~~ that can be ordered ✓ as eg. easy < complex < very complex. ~~Their distances~~ The intra-label distances cannot be assessed and definitely not equal. But as they can be ordered ~~the special~~ they are not nominal either. 0,5

so they are not ratio scaled.

Ordinal by one.

3c)

I would say that although cyclomatic complexity is a useful and valid measurement, there is no guarantee that this particular company and software would benefit from it. It takes time and for one, but more importantly a general assessment of the project and the choice of an appropriate standard would provide more suited solution. Appropriate standard means a quality assurance plan that help similar companies finding a way to reduce software quality issues including complexity.

If the manager insists to do it without a plan, I would find more general, faster and easier to perform measurements, such as number of line of codes, number of dependencies between classes, or number of interfaces.

General steps: 1) Understand the basic structure of the software

2) Do literature search, preferably finding standards for similar software systems

needs?

3) Close a few appropriate measurements and/or write my own according to the specific needs of this company

3.5) Take time to reduce complexity in some key classes or in all of them

4) Find ways to pro-actively reduce complexity, implement good practices for the future

04 approach

04 1 1 v 4,5

5

3c)

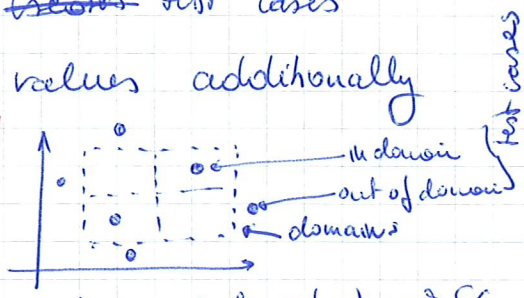
Exploring relationships is to ~~find~~ visualize, ~~quantify~~ ~~calculate~~ and describe and thus explore the measurement without any assumption. ~~One~~ For example plotting the data on a scatter plot and use linear regression to ~~show~~ fit a line ok (or other type of regression to fit any kind of curve) could give insight in the behaviour of the data. Additionally correlation analysis can be done to assess ~~the~~ if there is a relationship between factors.

Confirming a theory requires ~~a~~ ~~an~~ a hypothesis to be proved or rejected. Depending on the properties of the data (eg if it follows natural distribution) and the number of groups of data, ~~the~~ one can perform T-test or ANOVA or Kruskal-Wallis test to see if ~~the~~ ~~interacts~~ the ~~our~~ hypothesis holds.

5

4.

a) Robust Equivalence-Class Testing defines test cases for out-of bound/~~to~~ domain values additionally to inner domain values. E.g. ✓ Using this we would end up tests targeting exception handling, ~~and of~~ apart of test cases targeting valid expected values.

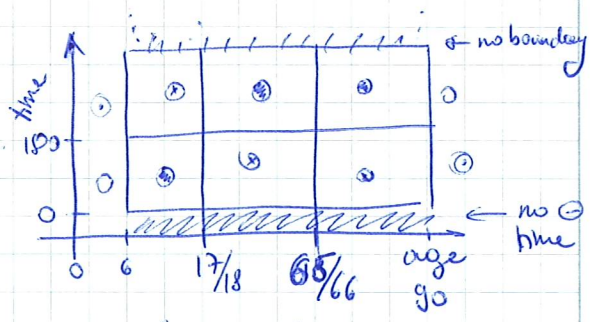


An example for robust weak EC

All-Uses Coverage Criterion requires the ~~path~~ code coverage of all nodes, edges, ~~paths~~, ~~tours~~ and data flow, control flow) DU-paths. Therefore would define test cases for ~~input value pairs~~ all possible scenarios. ~~scenarios~~. (1)

b) i) dimensions are [age] [6-17], [18-65], [66-90] ✓
and [time] [0-180], [180+] (or [180-∞]) (1)
~~and ther ch~~ and these are the equivalence classes respectively

ii)
1) weak nomal EC: 3
Since max(age, time) = 3
age has 3 EC, time has 2



2) weak robust EC: 5
because time has only nelnal boundaries (0 and ∞)
~~and~~ so only two outer-domain options remain from age <6yo and >90yo.
to add.

3) Strong nomal EC: 6
because age × time = 3×2 = 6 ✓ (2)

4) Strong robust EC: 10
because the ~~time~~ time has 2 ECs and age has two out of bound cases, so 2×2=4
6 for nomal & strong EC + 4 = 10

5

4)

iii)

```
public calculateEntryPriceTest {

@Begin
private int age;
private int time;

@Begin
public calculateEntryPriceTest() {
  this.age = 0;
  this.time = 0;
}

@Test
public void youthShortEntry() {
this.age = 10;
this.time = 60;
int resultOfTest = calculateEntryPrice(age, time);
int expectedResult = 20;
   assertEqual(resultOfTest, expectedResult);
}

@Test
public void middleAgeLongEntry() {
  this.age = 40;
  this.time = 200;
  int resultOfTest = calculateEntryPrice(age, time);
  int expectedResult = 40;
  assertEqual(resultOfTest, expectedResult);
}

@Test
public void oldLongEntry(){
   this.age = 80;
   this.time = 200;
   int resultOfTest = calculateEntryPrice(age, time);
   int expectedResult = 30;
  assertEqual(resultOfTest, expectedResult);
}

}
```

No point of using fields (but not incorrect)

maybe other way around: first expected second actual result

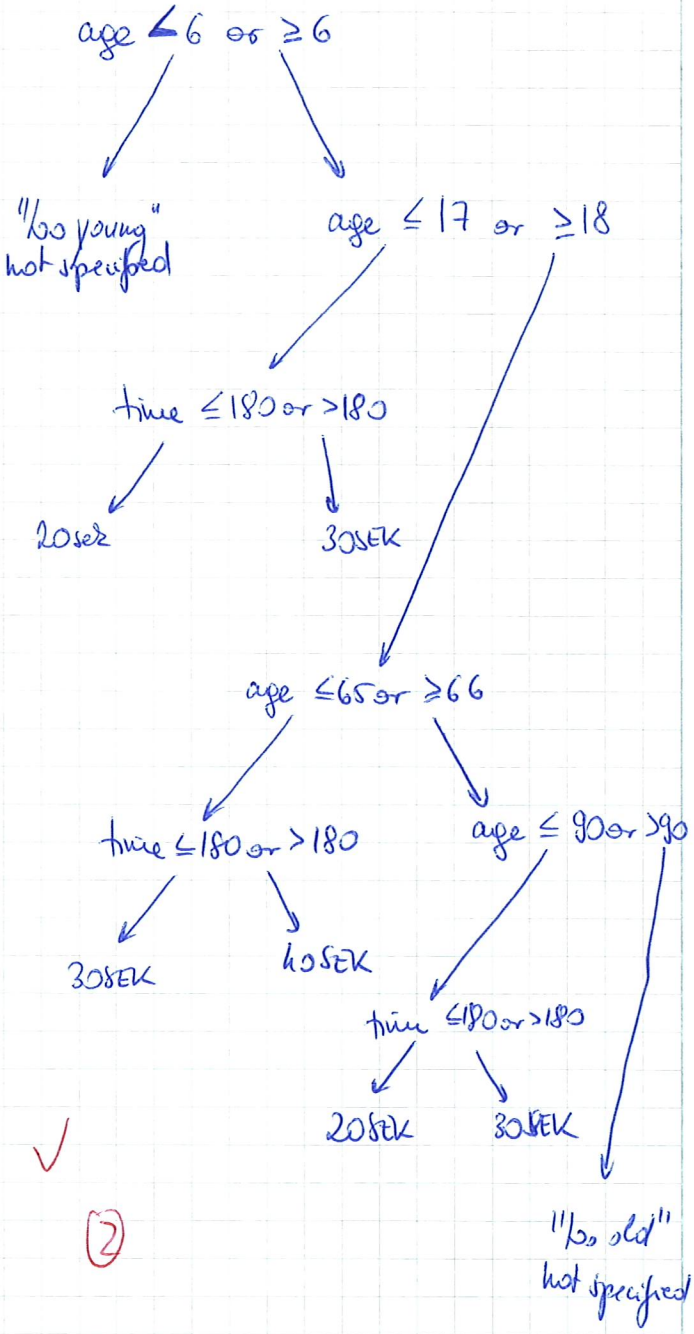
✓ (3)

5

(4) iv.)

classification tree:

age ~~≥6 or ≤6~~



decision table

age / ~~time~~	≤180	>180
x <6	not specified too young	not specified too young
6≤x≤17	20	30
18≤x≤65	30	40
66≤x≤90	20	30
90<x	not specified too old	not specified too old

✓

(2)

5

(5)

a) Test-driven-development is a proces when no code is written before a test-case covers it. It assures that only the necesary amount of code is written (or at least by to support it, but it is up to the developer) and that all code is motivated by a failing test which is motivated by requirements and all code is supported by a test case. The steps are: planning, writing test, failing test, implement code, pass the test, refracktor, test again; repeat from the beging.

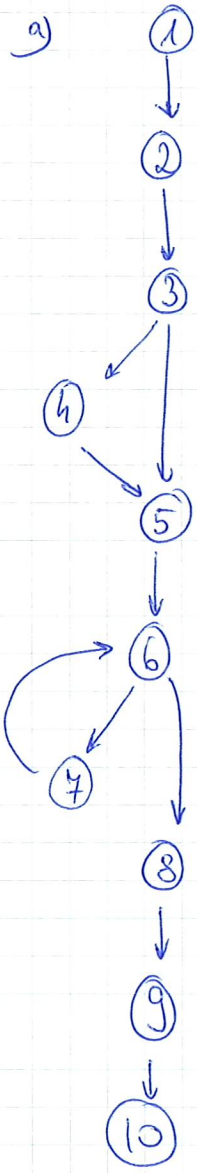
d) Refracktoring is easy and safe because the same test could run to ensure that the code is still working as intended. (regression test) It reduces the fixes towards the end of the development proces because tests are written from the beginig and in code phase they ensure the corrections of the code.

On the other hand the test class is usually 5-10 tmies bigger than the original class, therefore it slows down development and still cannot make sure that no requirements no specifiation is omitted.

5

6

a)



✓ (2)

b)

~~1,2,3,5,6,8,9,10 → 1.~~
~~1,2,3,4,5,6,8,9,10 → 2.~~

1,2,3,5,6,8,9,10 → 1.

3,4,5 → 2. not prime, e.g. 1234678 covers it −1

6,7,6 → 3.
incomplete - 0,5

3

(0,5)

Prime path is a simple path that is not part of any ~~other~~ single path.

c) 3

# of edges = 11
# of nodes = 10

11 − 10 + 2 = 3 ✓

(1)

5

(7)

a) A/B testing aims to compare two different version of a system and assess which of them is better in some way. ✓ 1

A typical case is when the users are randomly assigned to two groups and the two groups recieves two different version of the same program. Data is collected to decide which version of the program was preferred by the users or that performed better in some pre-defined way.

Hypothesis?

Steps: ① assign test subjects to groups A and B as randomly as possible ② provide one-one different versions of the same system to each group. The two versions should differ in only the aspect to be tested and be identical in every other way. ③ Perform the test, collect usage data for a specified amount of time ④ analyse the data, test the hypotesis of the two version being different - If they are, see which one is better and invest money and energy on that one. If no difference, drop the project.

DESIGN ✓ 0,5 0,5

⑤ define a measurement of success ✓ 0,5

AB testing is a user-level testing that many users are involved in. It gives a non-biased insight in the preferences of the users for whom the software is developped. This cannot be achieved by regular testing. 0,5 1

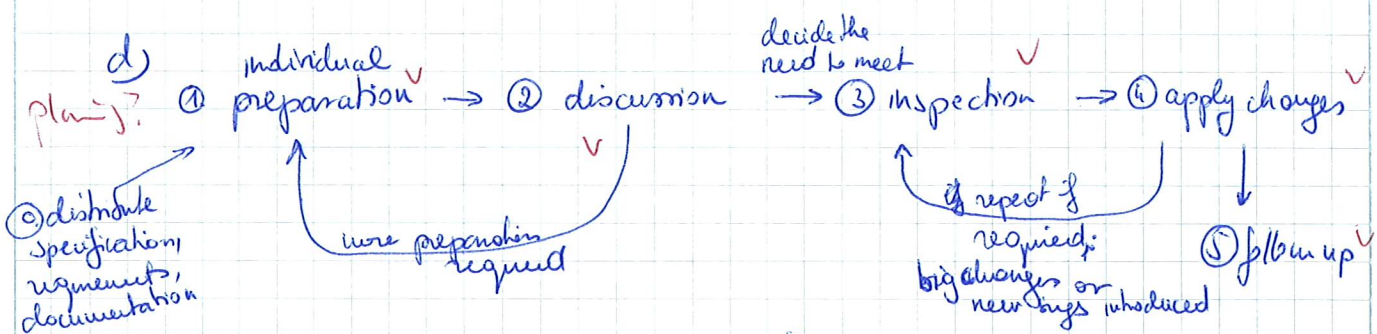
by ① Deciding between two user interfaces for my application that is already out. One interface would be the old version, the other would be a new one. I would assess the number of clicks on the blog my application would include if I wanted to improve its accessibility for example. ✓ 1

5

(7) b) ② Another example could be if I were Google and wanted to have more visits on sites I promote, I could test the current version against another one with bigger letter size and assess the number of clicks on the promoted sites during the test period.

c) Mutation testing is the process that evaluate test-coverage. It is by introducing mutations: small changes in any part of the code and count the times when it caused any test case to fail. If the code coverage by tests is 100% all bugs are caught and the result of the mutation test is 100%. If one bug is did not cause a test case failure out of 10 introduced bugs/mutations, the result of the mutation test is 90%. If no test exist for the code, no introduced mutation is caught, the result is 0%. Operations can be mutated by changing boolean values from true to false or vice versa; or by changing arithmetic operators such as ⊖ → ⊕ or multiplication instead of division; or it can change logical operators from AND to OR or ⩽ → ⩾; and there are many more options.

d)



Individual preparation includes reading the specification and understanding the code done individually by members of the inspection comittee.
During inspection the comittee goes through the code line by line with the reader of the code and a chair. They rely on checklist created earlier in the preparation phase. Checklist help prioritizing the inspected problems.