10 - YO26FU7CUZ62XFN

0406186522

FÖRSÄTTSBLAD TENTAMEN/ EXAMINATION COVER

Jag intygar att mobiltelefon och annan otillåten elektronisk utrustning är avstängd och förvaras på anvisad plats. / I hereby confirm that mobile phones and other unauthorized electronic equipment is shut off and placed according to instructions

MARKERA MED "X"/ MARK WITH "X" [X]

IFYLLES AV STUDENT OCH TENTAMENSVAKT/ TO BE FILLED IN BY THE STUDENT AND THE INVIGILATOR:

KURSKOD / COURSE CODE: ID2207	EFTERNAMN / FAMILY NAME: SHI
KURSNAMN / COURSE NAME: Moderna metoder inom Software Engineering	FÖRNAMN / FIRST NAME: XIYU
PROVKOD / TEST CODE: TEN1	NAMNTECKNING / YOUR SIGNATURE: [illegible]
TENTAMENSDATUM / EXAMINATION DATE (Y/Y/Y/Y - M/M - D/D): 2018-10-25	PERSONNUMMER / PERSONAL NUMBER (Y/Y/M/M/D/D): 960927-9022

PROGRAMKOD / PROGRAM CODE:	INLÄMNINGSTID / TIME SUBMITTED: 12^00	SIGNATUR TENTAMENSVAKT / SIGNATURE INVIGILATOR: [illegible]	ANTAL BLAD / NO OF SHEETS: 05

MARKERA BEHANDLADE UPPGIFTER MED "X "OCH EJ BEHANDLADE UPPGIFTER MED "-" / MARK WITH "X" PROBLEMS SOLVED. MARK WITH "-" PROBLEMS NOT ATTEMPTED

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X	X	X	X	X	X	X	X	X	X	X									

IFYLLES AV INSTITUTIONEN / TO BE FILLED IN BY THE DEPARTMENT:

BEDÖMNING / ASSESSMENT

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

BONUSPOÄNG/ BONUS POINTS: __ __ , __

SLUTSUMMA / FINAL POINTS: __ __ , __

BETYG/ GRADE: A

Godkänns av examinator / approved by Examiner: [illegible]

	Ia	Ib	Ic	IIa	IIb	IIIa	IIIb	IIIc	IVa	IVb	Va	Vb	VIa	VIb	VIc	VIIa	VIIb	VIIIa	IXa	IXb	Xa	XIa	Exam	Part.	HWs	ExQuiz	Proj	Total	Grade
Shi, Xiyu	3	4	1	2	5	3	1	1	3	3	4	5	4	4	3	4	4	5	4	4	5	1	73	6	5	1	5	90	A

Family name, first name	Personal Registration Number	Programme	Sheet no.	Problem no.
Shi Xiya	960927-9022	Emmbedded system	1	I, II, III

I. (a) ① Abstraction: ignor non-inessial details

② Decomposition: devide into sub-systems

③ Hierachy: combine chunks.

(b) ① First: is model
② Second: is model
③ Third: is model
④ Fourth: is NOT model

①②③ are all the abstraction of something in real word,
However ④ is not an abstraction.

(C) ~~To~~ To use this principle to verify the correctness of the software.

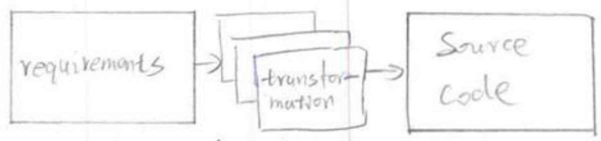
(b) ① Common: Both synthesis and transformation use ~~deductive~~ method of deductive.

Difference: synthesis → inference
transformational → replacement

where: inference: the act or pocess of forming an opinion based on what we already know. It change ~~A~~ axiomal/rules to operations.

② transformational synthesis and MDA

transformational:

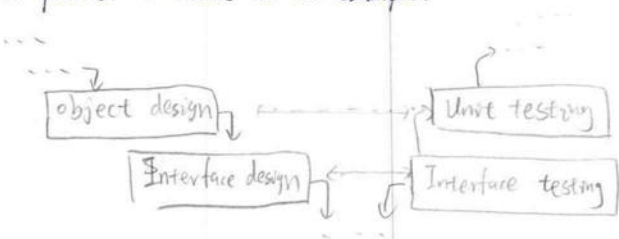


Common: all transformed to source code space

difference:

Transformational synthesis use requirements to transform
MDA use models to transforms

II. (a) a part of V-model as an example:

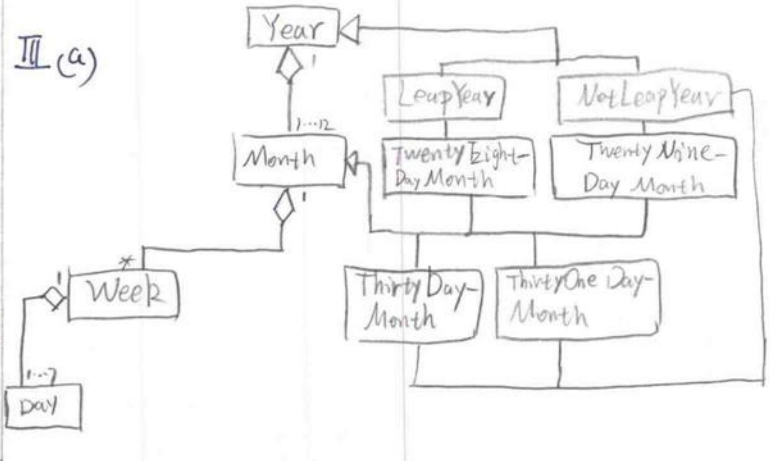


① which means during the process before implemotation, each activity has its own testing part. This will make sure the former activities' fault will not pass to the next step.

② Difference(s) with Waterfall model:

The waterfall model do not have the tesing Part (i.e. the right-hand part in the figure). This will Cause once the phase passes, it will never go back, and no verifying step will cause problems

III (a)



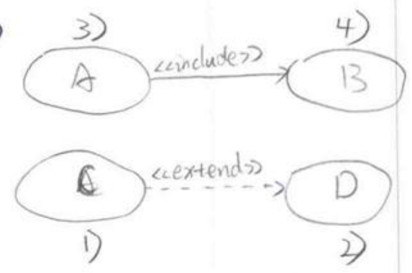
Using the number of days in a week, and ~~sub~~ child-classes of a month, the number of weeks can be deducted ~~from~~

Other relationships (days→month, months→year, days→week) are shown directly in the UML, the class diagram

Family name, first name	Personal Registration Number	Programme	Sheet no.	Problem no.
Shi Xiyu	960927-9022	Embedded system	2	III, IV, V

oe'2'

(b)



The defination of four statement is shown above.

Answer: (1) and (3) are always independently meaningful and operational.

Because (4) is ~~some~~ use case included in (3) if (3) doesn't happen, and no other use case generate (4), (4) will not happen ~~either~~ either.

(2) is ~~so~~ use case may happen, which extended from (1), thus, if (1) don't happen, (2) will noe happen.

(c) A. Use delegation when you are repeating yourselfu...

B. Use generalization when you have one use case...

C. Use include and extend when you are describing a variation ......

IV. a) type: functional requirements
non-functional requirements.

Functional: the functionility of the system, e.g. data structure, input and output data...

Non-functional: refer to the "look-and-feel" part. for example: performance, relaibility and so on.

b) ~~Ma~~ ① The traceability makes the developing process go smoothly in order, ~~[illegible]~~

② reduce the gap between user and developer

③ makes sure the ~~fini~~ final product is exactly the one the users want to use.

④ If something wrong happens, we can easily find where the problem occurs

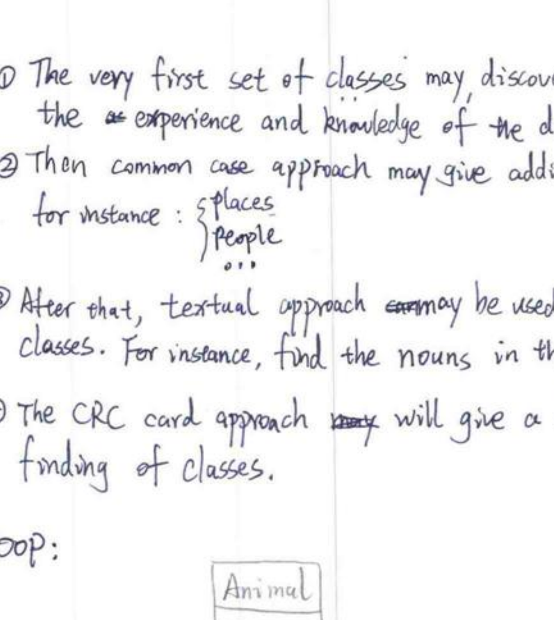
V. (a) ① The very first set of classes may discoveried using the ~~a~~ experience and knowledge of the developer.

② Then common case approach may give additional idea, for instance: { places
people
...

③ After that, textual approach ~~can~~ may be used to find more classes. For instance, find the nouns in the requirement

④ The CRC card approach ~~may~~ will give a deepen finding of classes.

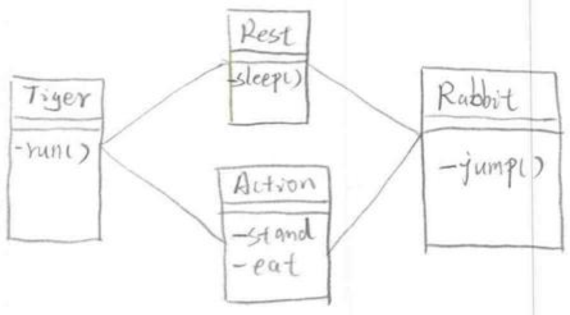
(b) oop:



(AOP is in next page, Page3)

Family name, first name Shi Xiyu	Personal Registration Number 960927-9022	Programme Embedded system	Sheet no. 3	Problem no. V, VI, VII

AOP:



In this example, AOP sumrize the common options which are repeated in OOP, this ~~is~~ will reduce the code size and complecity.

VI. 1. performance, end-user

The statement involves the "users" and the "1 second" specitication refers to performance

2. dependability, end-user

When the network failure occurs, whether the systeem can issue train ticket is related to it dependabiliey; if not, it is not dependable. Also, this involves the end user.

3. maintenance

If the system donte allow the installation of additional button, ~~the~~ it will be difficult to mainten.

4. dependability, end-user

① It involves the uses' action;
② It is not dependable if it is easy to be attacks by uses

5 dependability, end-user

Same reasons with "4"

(b) The layered architecture can only invoke the operations in direct the next layer. So the benefits: ① make it ~~more~~ more portable

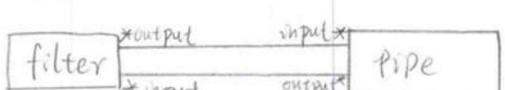
② reduce the coupling, rise the coherance.

Problems ① Compared to open-architecture, the data transform is ~~less~~ of less efficiency, so the trade-off here is efficiency (open arch.) & portable (closed arch.)

② no insistant to change. A change occurs in one layer may cause the changes in the whole ~~model~~ architecture.

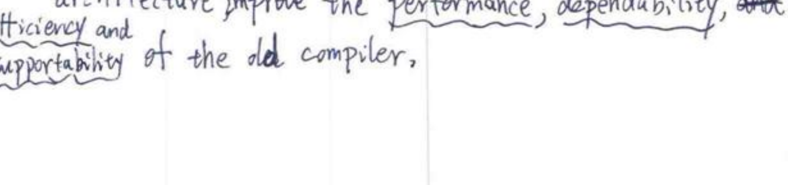
(C) ~~de~~ Design goles include: performance, dependability, efficiency, supportability cost, maintenance, and end user.

the pipe and filter architecture is like:



from the picture, we find data must go through the whole fileer and pipe, ~~dont~~ cannot go back, so this architecture is very weak when there are a lot of interactions with user. The change to repository architecture improve the performance, dependability, ~~and~~ efficiency and supportability of the old compiler,

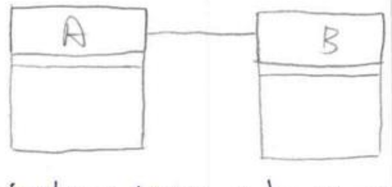
VII (a) Pattern selected: Strategy pattern



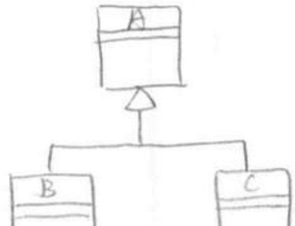
From the UML, ~~we~~ all the algorithms are "black-box" to the Main function [encryption], and developer can add or reduce the Algorithms through "strategy" parent class [dynamically], and because of the decoupling of algorithms and Main, [computing time] improved.

Family name, first name	Personal Registration Number	Programme	Sheet no.	Problem no.
Shi Xiyu	860927-9022	Embedded syseem	4	VII, VIII IX

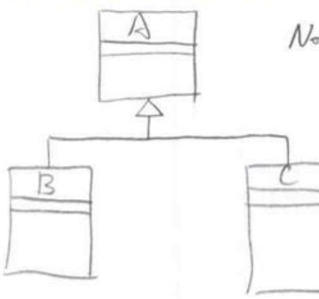
(b) delegation:



implementation inheritance

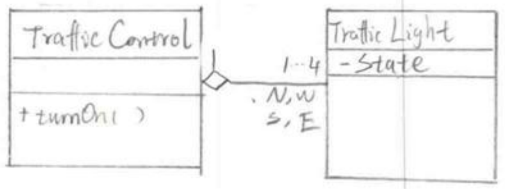


specification inheritance



Note: Here, Class A is an abstract class

VIII



Context TrafficControl::turnOn() inv:

N.state = S.state and
W.state = E.state and
N.state != E.state ~~and~~

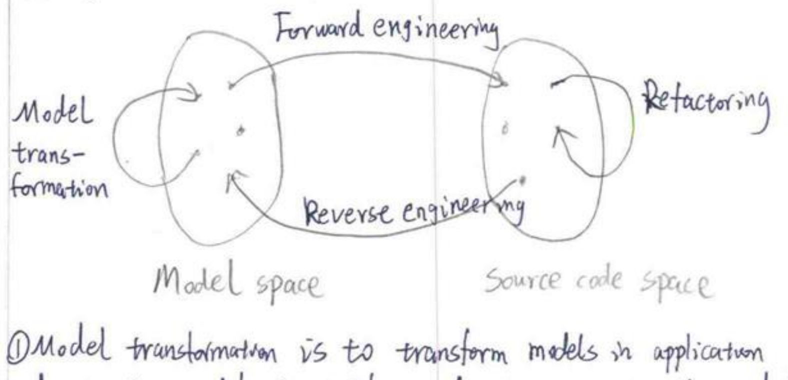
Context TrafficControl::turnOn() inv:

N.State = red and
S.State = red and
W.State = green and
E.state = green

Context TrafficControl::turnOn() inv:

when N.State = red,
W.State = yellow or green

IX (a)



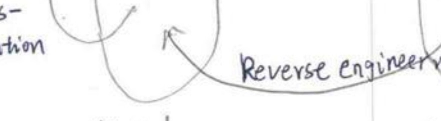
① Model transformation is to transform models in application domain to models in ~~s~~ solution domain, generate the models like UML which can be used to write code

② Forward engineering is the process to transform models to code.

③ Refactoring process is to add more details to the program, for instanse: missing relationships between actors and use cases.

④ Reverse engineering is the process to use ~~mode~~ source code generating new model, and use this model to verify whether it fullfull the requirements.

(b)



```
public class A() {
  privite B. b;
  public B(b) {

    this.b = b;


  }


  privite getA() {
     return b;
  }


}
```

Family name, first name	Personal Registration Number	Programme	Sheet no.	Problem no.
Shi Xiyu	860927-9022	Embedded system	5	X, XI

X.(a) Reliability: To what extent the system fullfill the requirements ~~gived~~ given by clientes. The ~~reliability~~ reliability is high if it fullfill all the requirements perfectly.

Fault: i.e. bug, refers to the algorithm or the mechanism errors.

Erroneous state: i.e. error, errors while running the program.

Failure: failure is the state output doesn't fullfill the requirement.

XI. ① They are test-driven process, write test first and then give the code which can meet the test.

② It purpose is to adapt to the quick changing reqirements

③ Pair programming introduced for a higher develope efficiency.

④ Different iterations applied during the developing process

⑤ Use menophers to help clients understand the system.

⑥ Release plan given based on the iteration plans.