VoorbeeldTentamen IN2705

Software Engineering

Uitwerking

1.

a.

- docenten
- studenten
- administratie
- examencommissie

b.

- Usability : het systeem moet door veel mensen met uiteenlopende vaardigheden worden gebruikt.
- Efficiency: het systeem doet geen moeilijke berekeningen, dus tijdsefficiency speelt geen rol. Ook de hoeveelheid data is (naar hedendaagse begrippen) niet zo groot.
- Reliability. Erg belangrijk. Als resultaten van studenten verloren gaan heeft dat vergaande consequenties voor de student.
- Maintainability. Studieprogramma's plegen te veranderen.

c. ()

d. Een belangrijke keuze is hier of de gegevens centraal worden opgeslagen (een database voor de gehele TU), dan wel gedistribueerd (een database per studierichting of faculteit).
Een centrale database leidt tot en client-server architectuur, of een repository-architectuur (ook wel Blackboard architectuur). De centrale database wordt dan wel een single point of failure.
Een betere keus is de gegevens per faculteit te beheren. Dat leidt tot een Service-Oriented of zelfs Message-Oriented architectuur, waarmee een grote flexibiliteit kan worden bereikt.

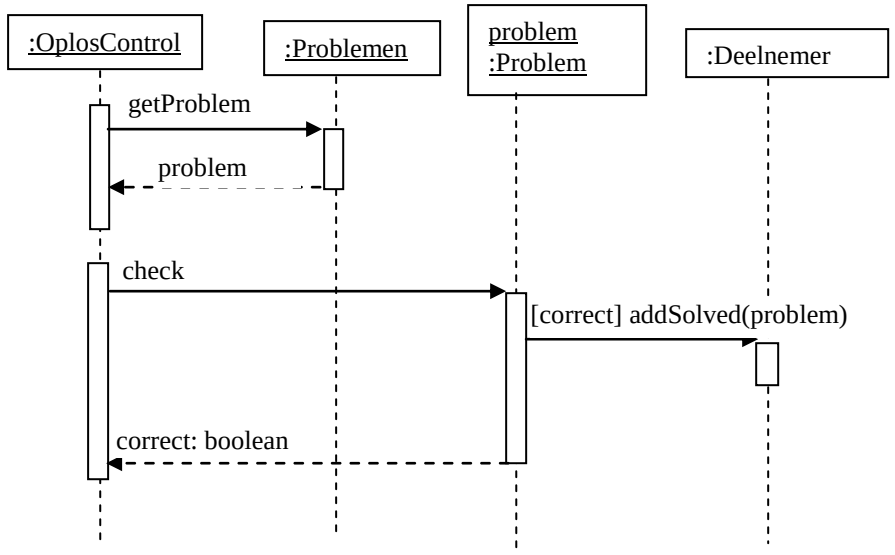
2.

a. Use Case : los probleem op

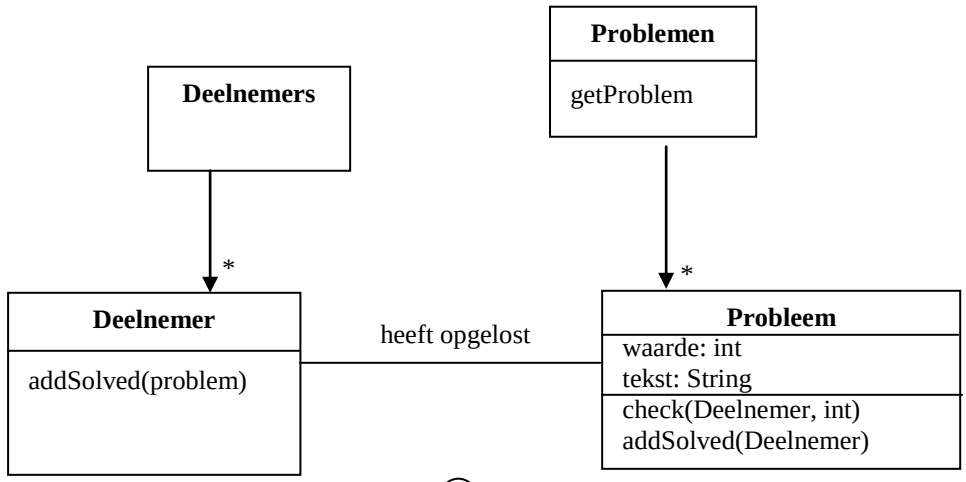
1 user selecteert problemen	2 systeem toont de problemen
3 user kiest een probleem uit	4 systeem toont de volledige tekst
5 user lost op en voert de oplossing in	5 systeem verwerkt resultaat en meldt of het goed is.

.

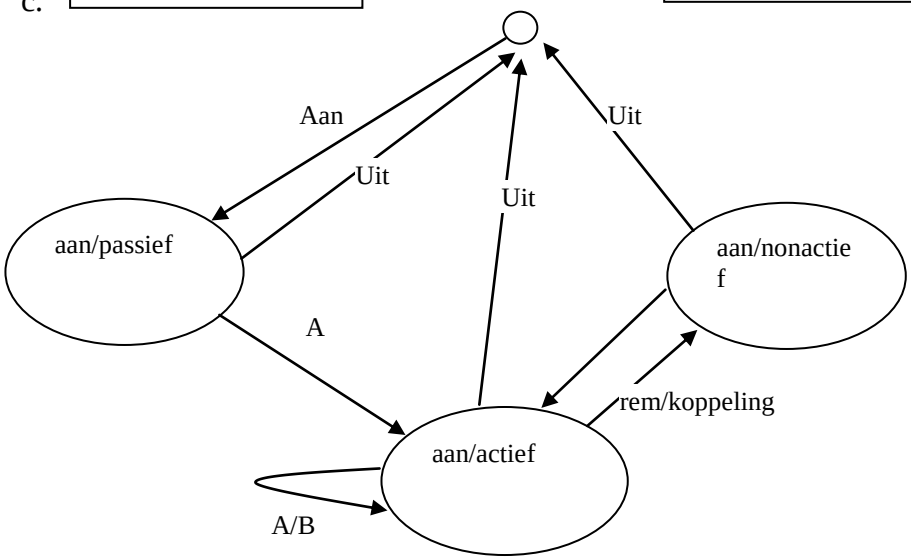
b.



c.



3.



4.

a. Context Weg
   inv: van <> naar
b. Context Route::lengte()
   post result = sum weg->collect(lengte())

5.

a. Stamp Coupling (in de constructor) en routine call coupling
b. Idem

6.

a.
   i. divide and conquerer
   ii. reduce coupling
   iii. keep abstraction high
   iv. increase reusability
   v. design for flexibility
b. Design defensively – er wordt niet gecontroleerd of de ‘op’ bij de constructie echt een ‘op’ is. De eval gaat daar wel van uit, en geeft een onzinwaarde terug als dat niet zo is.

7.

a) We zien hier een Delegation pattern. De linked list heeft de gewenste functionaliteit, maar de namen zijn verkeerd

b)

rol	implementatie
Delegator	ListQueue
Delegate	LinkedList

a) Men kan ook aan een Adapter pattern denken

b)

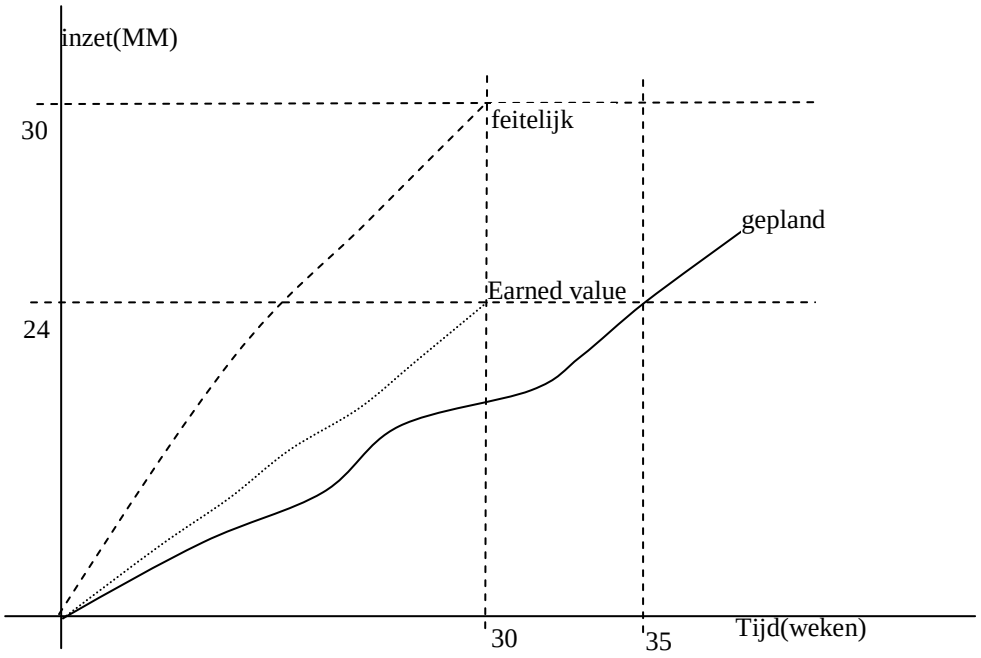
rol	implementatie
Superclass	Queue
Adapter	ListQueue
Adaptee	LinkedList

8.

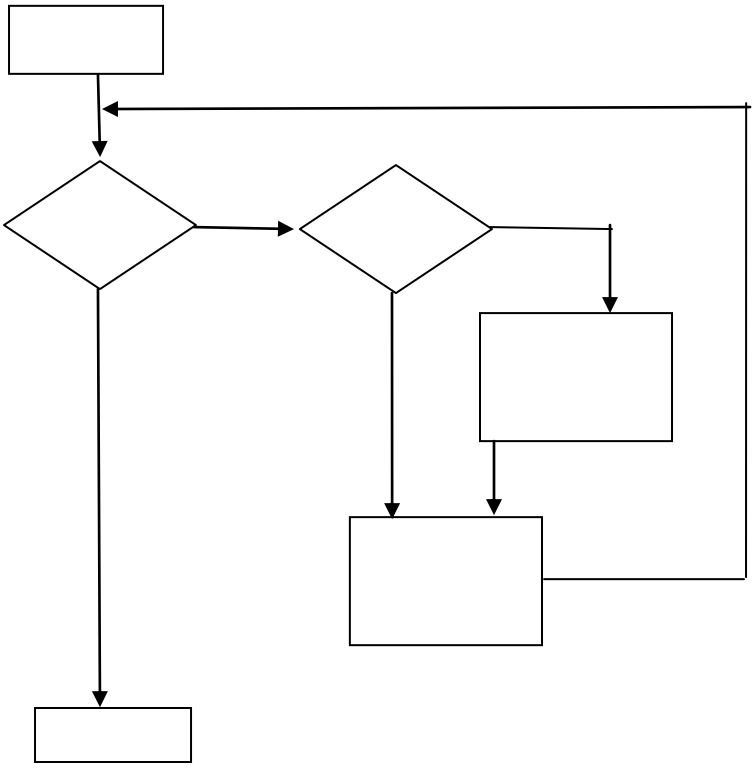
a)

- kosten: 2077 mensmaanden
- duur: 28.8 -> 29 maanden
- teamgrootte : 2077/29 -> 72

b)



9.



b

Er is een pad dat de herhalingslus 0 maal doorloopt. Dit pad wordt doorlopen voor n = 1

Er zijn twee paden die de lus éénmaal doorlopen:

een die door de if gaat , wordt doorlopen voor n = 2

een die niet door de if gaat, maar die kan niet worden doorlopen

Er zijn in principe 4 paden die de lus tweemaal doorlopen, want bij elk van beide doorgangen moet je besluiten of je wel of niet door de if gaat.
Van die 4 paden is alleen het pad realiseerbaar, dat de eerste maal niet, maar de tweede maal wel door de if gaat. Dat pad wordt doorlopen voor n = 3.

10.

Als $a \neq 0$ zijn er drie gevallen:

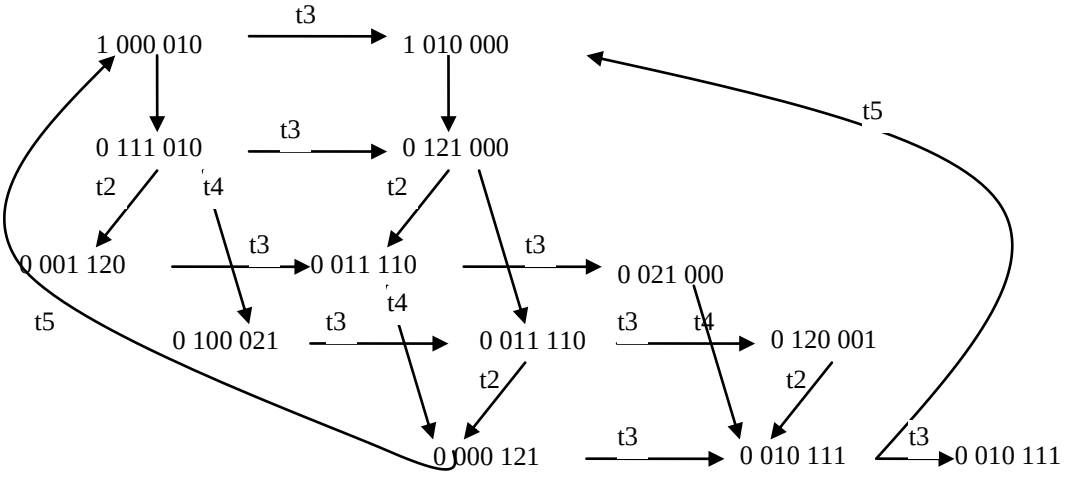
- $D = b^2 - 4ac > 0$, eg. a = 1, b = 2, c = 0
- D = 0 eg. a = 1, b = 2, c = 1
- D < 0 eg. a = 2, b = 1, c = 2

Als a = 0 ontaardt de vergelijking tot een lineaire vergelijking. Dan zijn er drie gevallen:

- $b \neq 0$, eg. a = 0, b = 2, c = 3
- b = 0 en $c \neq 0$ eg. a = 0, b = 0, c = 1
- b = 0 en c = 0 dus a = 0, b = 0, c = 0 (wat moet hij dan teruggeven?)

11.

a) t1 -> t2-> t3 -> t4 -> t5 .
b) t1 -> t2 -> t3 -> t4 -> t3 .
c) t1 -> t2 -> t3 -> t4 -> t3 -> t3.
d)



e)

i. safe: neen
ii. bounded? ja
iii. conservative: neen