Nota: Schrijf je antwoorden kort en bondig in de daartoe voorziene velden. De puntenverdeling is 2 punten per theorie-vraag en 8 punten per oefening. Het totaal is 40.

Vraag 1.

Er bestaan verschillende soorten prototypes. Zo dient een exploratory prototype eerder voor valideren van requirements/ verkennen van het ontwerp (technieken) terwijl een evolutionary prototype in stappen groeit naar het uiteindelijke product.

Vraag 2.

Om een component te testen stel je een aantal test cases op die proberen de uitvoering van een component te laten falen. Ze bestaan uit een verzameling inputs en verwachte resultaten.

Die inputs worden op een gecontroleerde manier aangeboden door (test) stubs

De test driver zorgt voor de uiteindelijke uitvoering ervan.

Vraag 3.

Een programmeur verkiest een sterke preconditie bij het schrijven van een functie omdat gemakkelijk te voldoen aan de postconditie / weinig gevallen binnen functie

Een zwakke postconditie is dan weer interessant omdat gemakkelijker aan te voldoen / eindsituatie niet beperkt

Maar wat met de invariant? Maakt de pre-conditie makkelijker en de post-conditie moeilijker. Dus zwakker beter.

Vraag 4.

Hoe staat een use-case in verhouding tot een scenario?

Een scenario is een instantie van een use-case, een typische uitvoering.

Een use case is zowel een primary als een secondary scenario.

Vraag 5.

Je kan een probleem opdelen adhv een object georienteerde decompositie wat wil zeggen dat je opdeelt in **objecten** die je systeem moet manipuleren
Dit resulteert in (verschillende gekoppelde) **is-a** hiërarchieën.

Een alternatief is de functionele decompositie die resulteert in (een strikte) **subfunction-of** hiërarchie. Waarbij je probleem opgedeeld wordt in de functies die je systeem moet uitvoeren

Vraag 6.

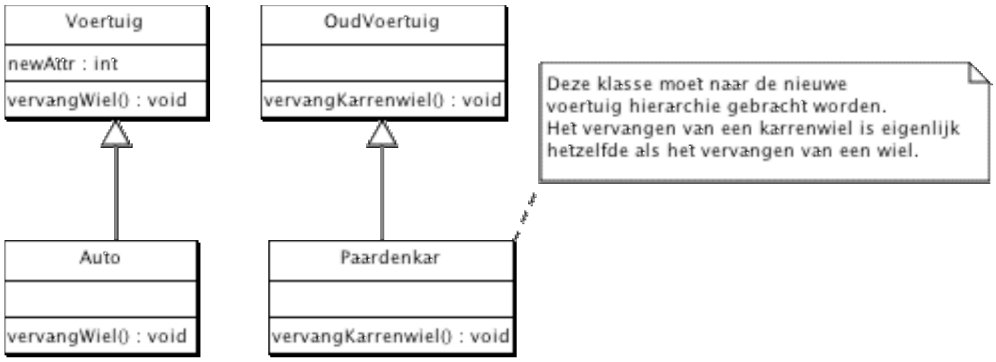
6.1 Wat is het verband tussen formele specificaties en design by contract?

Design bij contract is een praktische afleiding van formele spec

6.2 Hoort UML ook thuis in het rijtje formele specificatietechnieken? Waarom wel of waarom niet?

Neen, uml is een semi-formele taal / vaste syntax, losse semantiek

Vraag 7.



Als we de klasse Paardenkar in onze nieuwe voertuighiërarchie willen invoegen, welk pattern kunnen we dan gebruiken?

- adapter pattern.

Geef drie afwegingen (trade-offs) die je daarbij moet bekijken

- hoeveel aanpassen? Hoe zullen de apart ontwikkelde klassen verder evolueren? Werkt het samenvoegen in een of beide richtingen? Hoeveel overhead in performantie en maintenance ben je bereid te aanvaarden?

Vraag 8.

8.1 Geef 3 voorbeelden waarin een refactor-operatie in een OO context aangewezen kan zijn gedupliceerde code, geneste (complexe) conditionals, grote klassen/methoden, misbruikende overerving

8.2 Zou je regression tests gebruiken tijdens refactoring? Waarom wel of waarom niet?

Ja, systeem blijft zo correct als het was / praktisch verifieren van behaviour preserving.

Vraag 9.

Verklaar onderstaande kwaliteits-begrippen:

Robustness = systeem werkt redelijk in omstandigheden die niet gespecifieerd waren.

Correctness = systeem werkt volgens zijn specificatie

Reliability = - gebruiker mag vertrouwen op het goed functioneren van het systeem.
- de kans dat een systeem werkt volgens zijn specificatie gedurende een bepaald interval.

Verifiability = de mogelijkheid om te verifieren of de gewenste attributen/eigenschappen aanwezig zijn.

Vraag 10.

Een slip line laat je toe de **huidige** slip status van project-taken te bestuderen, terwijl een timeline je toelaat de **evolutie** van de slip status van project-taken te bestuderen.

Vraag 11.

Wat is een maat (measure)? Een functie die een attribuut van een real-world entity mapt op een symbool (in een verzameling met wiskundige relatie).

Welke elementen moeten zeker gespecifieerd zijn vooraleer we een maat (measure) precies (precise) mogen noemen? Domain, range en mapping regels

Vraag 12.

Waarom is “object orientation” geen mogelijke silver bullet.

Vraag 13. *Oefeningen*:

13.1 Specifieer in Z een collectie van e-mailadressen waarbij de gebruikersnaam en de domeinnaam aparte verzamelingen vormen en er een mappingfunctie tussen beide bestaat.
Definieer hierop

- Een zoekfunctie
- Een functie die alle adressen teruggeeft die voor 1 bepaald domein bestaan en achterhaalt bij welke domeinen een bepaalde gebruiker een emailadres heeft (bv. Iemand die een persoonlijk en een werk email-adres heeft: gebruiker@ua.ac.be en gebruiker@telenet.be)

Je krijgt volgende functie voorgeschoteld:

```
1  public void displayEvenNumbersBetween(int begin, int end) {
2       int current = begin;
3       while ( (current < end) AND isEven(current) ) {
4               display(current);
5               increase(current);
6        }
7  }
```

a) Teken de overeenkomstige “Flow Chart”.
b) Wat is de cyclomatische complexiteit (formule + aantal).
c) Geef alle onafhankelijke paden.
d) Geef per pad een mogelijke test-case.