***Belangrijk**: Schrijf je antwoorden kort en bondig in de daartoe voorziene velden. Elke theorie-vraag staat op 2 punten (totaal op 24). De oefening staan in totaal op 16 punten. Het geheel staat op 40 punten.*

Vraag 1. Introduction [..../2]

Wat is een "sprint" in het SCRUM proces? (De definitie heeft 3 componenten.)

- Een sprint is een periode (typisch **2 tot 4 weken**) waarin een **werkende increment** (potentieel shipable) wordt ontwikkeld
- De features die worden ontwikkeld worden gekozen uit de **product backlog**
- De sprint is de periode in het SCRUM proces waar er daadwerkelijk geprogrammeerd wordt.

Vraag 2. Project Management [..../2]

Wat is het "kritische pad" in een PERT Chart?

Het pad waarop vertraging op één taak resulteert in vertraging van het hele project.

Waarom is het belangrijk om dit te kennen?

Om **taken te identificeren** die geen vertraging mogen oplopen.

Om **resource optimaal te alloceren**

Om taken op het kritsch pad beter te **monitoren.**

Vraag 3. Use Cases [..../2]

Waarom maken we een onderscheid tussen *primaire* en *secundaire* scenario's?

Primary "success" scenario

= Happy day scenario

Scenario assuming everything goes right

(i.e., all input is correct, no exceptional conditions, …)

Secondary "alternative" scenarios

Scenario detailing what happens during special cases

(i.e., error conditions, alternate paths, …)

Dit is nuttig in een context van interatieve en incrementele ontwikkeling. (Om prioriteiten te stellen welke scenario's eerst te ontwikkelen)

Vraag 4. Domain Models [..../2]

Hoe kunnen *feature models* bijdragen tot het bekomen van correctheid?

(Are we building the system right?)

Good maintainability via a robust model of the problem domain.

(Are we building the right system?)

Feature Diagrams focus on commonalities/variations

Makes product differences (and choices) explicit

Vraag 5. Testing [..../2]

Wat is "Regression Testing"?

Regression Testing ensures that all things that used to work still work after changes.

= re-execution of some subset of tests to ensure that changes have not caused unintended side effects

Waarom is dit belangrijk?

Helps during iterative and incremental development + during maintenance

Vraag 6. Design by Contract [..../2]

Als je code outsourced naar programmeurs in India, gaat je voorkeur dan uit naar een *sterke preconditie* of een *zwakke*? Waarom?

Combinatie van beide:

*** Strong preconditions make a component more reusable

*** Weak precondition = Minder effort nodig om toestand te bereiken waarin de method kan opgeroepen worden.

En wat met de *postconditie*?

Weer afweging van beide:

sterk: methode doet heel veel

weak: few results to interpret

Vraag 7. Formal Specifications [..../2]

Wat wil het zeggen voor een *formele specificatie* als deze: (a) *consistent*, (b) *compleet* en (c) *ondubbelzinnig* is?

a) no contradictions in the specification

b) all properties are defined in terms of known concepts

c) misinterpretations are impossible

(sometimes also) minimal: there is only one way to express a certain property

Vraag 8. Software Architecture [..../2]

Definieer de volgende begrippen uit de ATAM terminologie?

sensitivity point: A sensitivity point is a property of one or more components (and/or component relationships) that is critical for achieving a particular quality attribute response.

tradeoff point: A trade-off point involves two (or more) conflicting sensitivity points.

Vraag 9. Quality Control [..../2]

Geef drie verschillende verfijningen van "Maintainability" en wat ze willen zeggen?

- **Repairability** - How much work is needed to correct a defect (= corrective maintenance)
- **Adaptability** (Evolvability) - How much work is needed to adapt to changing requirements (= perfective maintenance)
- **Portability** - How much work is needed to port to new environment or platforms (= adaptive maintenance)

Vraag 10. Software Metrics [..../2]

Tijdens welke fase van een software project zou je *metrieken* toepassen? Waarom?

- Effort (and Cost) Estimation
  - Measure early in the life-cycle to deduce later production efforts
- Quality Assessment and Improvement
  - Control software quality attributes during development
  - Compare (and improve) software production processes

Vraag 11. Refactoring [..../2]

Welke vier activiteiten zouden door tools moeten ondersteund worden als men aan *refactoring* doet?

1)Refactoring - Source-to-source program transformation behaviour preserving

⇒ improve the program structure

2)Regression Testing - Repeating past tests

⇒ improvements do not break anything

3)Programming Environment - Fast edit-compile-run cycles Support small-scale reverse engineering activities

⇒ convenient for “local” ameliorations

4) Configuration & Version Management - keep track of versions that represent project milestones

⇒ go back to previous version

Vraag 12. Conclusion [..../2]

Als je het “No Silver Bullet” artikel hebt gelezen:

Waarom zijn “Components” toch een potentiële Silver Bullet?

Buy vs Build: Het is beter een component te kopen dan hem zelf van scratch opnieuw op te bouwen.

Bovendien zijn componenten instaat om de een deel van de complexiteit weg te werken door encapsulatie in de component.

Als je het “Killer Robot” artikel hebt gelezen:

Waarom was het waterval model zo rampzalig in dit geval?

(From KillerRobotCase/articel-4.html) The waterfall model goes through definite stages of development. As the project passes from one stage to the next, there are limited opportunities to change earlier decisions. A drawback of this approach is that potential users are not able to interact iwth the sytem until very late in the process.

The Robot project involves a high degree of interaction, both between the robot components and between the robot and the operator. Since operator interaction with the robot is so important, the interface cannot be designed as an afterthought.

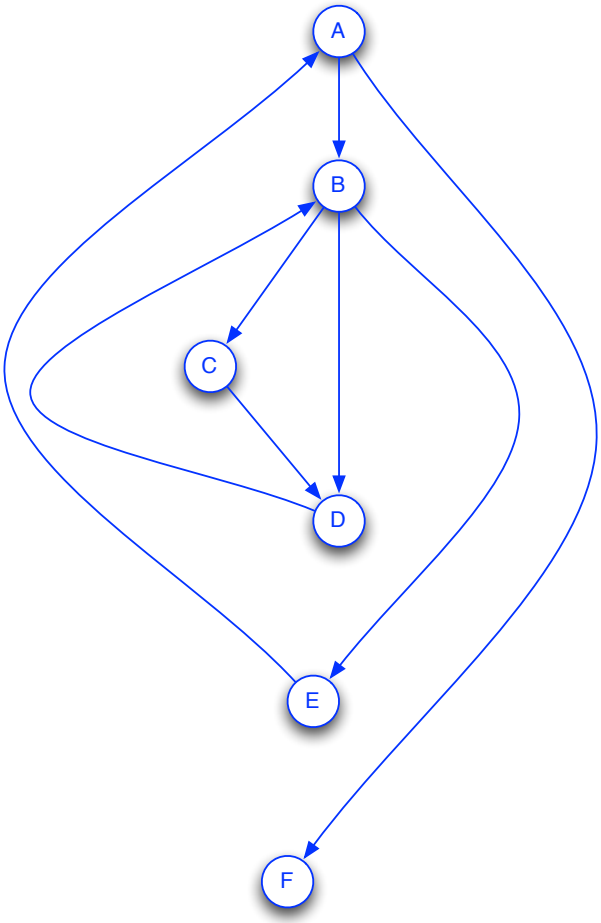
Vraag 13. Oefening - Testing [..../5]

Beschouw de volgende SelectionSort functie om een Array van integers te sorteren:

```
void selectionSort(int numbers[], int array_size)
{
  int i, j;
  int min, temp;

  for (i = 0; i < array_size-1; i++)
  {
    min = i;
    for (j = i+1; j < array_size; j++)
    {
      if (numbers[j] < numbers[min])
        min = j;
    }
    temp = numbers[i];
    numbers[i] = numbers[min];
    numbers[min] = temp;
  }
}
```

a) Teken de *control flow graph* voor bovenstaande functie.



b) Bereken de *cyclomatische complexiteit*, en geef kort aan hoe je hiertoe gekomen bent.

CC = 4

#e - #n + 2 = 8 – 6 +2 = 4

aantal binaire condities + 1 = 4

aantal graafregios = 4

c) Bepaal een volledige verzameling van *onafhankelijke paden*. (Nummer ze.)

1) A-B-C-D-B-E-A-F

2) A-B-D-B-E-A-F

d) Geef aan welke *input* vereist is voor elk van de *onafhankelijke paden* die je in c) hebt opgesomd. (Gebruik de nummering voor verwijzingen naar elk onafhankelijk pad.)

1) ([x,y], 2) met x > y

2) ([x,y], 2) met x < y

e) Hoe verhoudt het aantal paden in deze verzameling zich tot de *cyclomatische complexiteit*? Waarom?

aantal paden dat nodig is om alle knopen te doorlopen is maximaal de CC.

Anderzijds kan je door het kiezen van de input data sommige paden combineren.

Het minimale aantal hier lijkt 1 aangezien pad 1, pad 2 omvat.

f) Wordt hiermee deze functie goed getest? Hoe zou je het eventueel nog beter kunnen testen?

construction is a heuristic: does not necessarily result in set of independent paths

it is possible to get the same coverage with less paths

it is sometimes not feasible to exercise all required paths

it does not necessarily cover all entry-exit paths

--> Uitbreiden met Condition Testing en of Loop Testing

Vraag 14. Oefening - Planning [..../6]

Gegeven de *Pert-chart* op de volgende pagina:

a) Vul voor elke node in de *Pert-chart* de *earliest start date* in (uitgedrukt in weeknummers).

For each task n: compute earliest start-date

= Latest of all incoming paths

ESD (n) := latest of (ESD (preceding) + estimated time (preceding))

b) Vul voor elke node in de *Pert-chart* de *latest end date* in (uitgedrukt in weeknummers).

For each task n: compute latest end-date

= Earliest of all outgoing paths

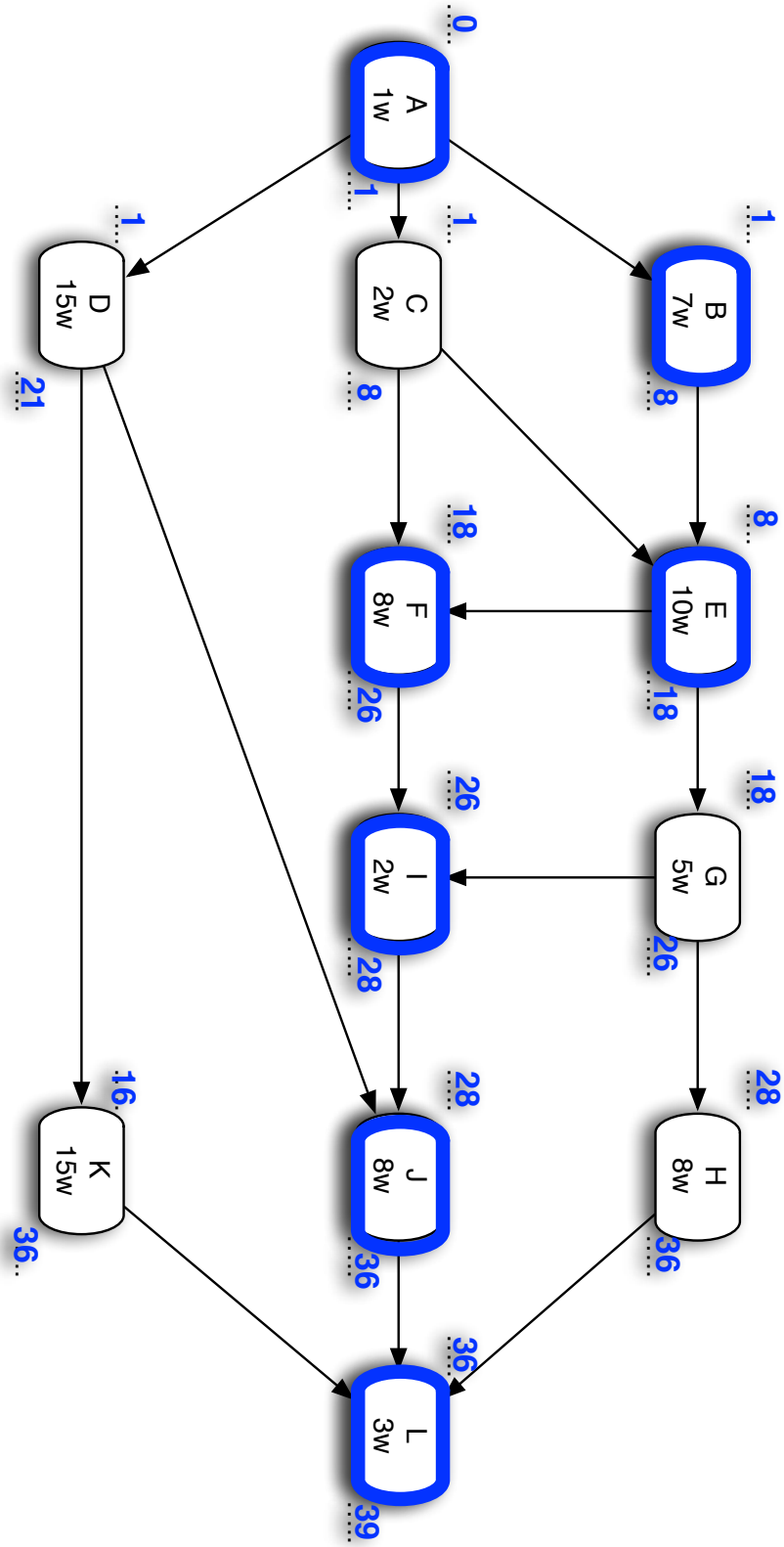
LED (n):= earliest of (LED (subsequent) - estimated time (subsequent))

c) Duid op de figuur het *critical path* aan.

Gegeven het schema op pagina 9. Bereken:

d) het *risky path*: A-D-J-L

e) de '*worst case delay*' van het project (gebruik '*estimated time*'): 6 of 6.83


A
1w
0
1
B
7w
1
8
C
2w
1
8
D
15w
1
21
E
10w
8
18
F
8w
18
26
G
5w
18
26
H
8w
28
36
I
2w
26
28
J
8w
28
36
K
15w
16
36
L
3w
36
39

	OT	LT	PT	ET	S_nominator	S_denominator	S	Path	SP	Worst case delay	
A	1	1	1	1.00	0.00	6.00	0.00	A	0.00	0.00	0.00
B	5	7	8	6.83	3.00	6.00	0.50	A-B	0.50	1.17	1.17
C	1	2	3	2.00	2.00	6.00	0.33	A-C	0.33	1.00	1.00
D	10	15	17	14.50	7.00	6.00	1.17	A-D	1.17	2.50	2.50
E	9	10	11	10.00	2.00	6.00	0.33	A-B-E	0.60	1.00	2.17
								A-C-E	0.47		2.00
F	7	8	9	8.00	2.00	6.00	0.33	A-C-F	0.47	1.00	2.00
								A-B-E-F	0.69		3.17
								A-C-E-F	0.58		3.00
G	3	5	6	4.83	3.00	6.00	0.50	A-B-E-G	0.78	1.17	3.33
								A-C-E-G	0.69		3.17
H	6	8	9	7.83	3.00	6.00	0.50	A-B-E-G-H	0.93	1.17	4.50
								A-C-E-G-H	0.85		4.33
I	1	2	2	1.83	1.00	6.00	0.17	A-C-F-I	0.50	0.17	2.17
								A-B-E-F-I	0.71		3.33
								A-C-E-F-I	0.60		3.17
								A-B-E-G-I	0.80		3.50
								A-C-E-G-I	0.71		3.33
J	4	8	10	7.67	6.00	6.00	1.00	A-C-F-I-J	1.12	2.33	4.50
								A-B-E-F-I-J	1.22		5.67
								A-C-E-F-I-J	1.17		5.50
								A-B-E-G-I-J	1.28		5.83
								A-C-E-G-I-J	1.22		5.67
								A-D-J	1.54		4.83
K	14	15	16	15.00	2.00	6.00	0.33	A-D-K	1.21	1.00	3.50
L	2	3	4	3.00	2.00	6.00	0.33	A-B-E-G-H-L	0.99	1.00	5.50
								A-C-E-G-H-L	0.91		5.33
								A-C-F-I-J-L	1.17		5.50
								A-B-E-F-I-J-L	1.27		6.67
								A-C-E-F-I-J-L	1.21		6.50
								A-B-E-G-I-J-L	1.32		6.83
								A-C-E-G-I-J-L	1.27		6.67
								A-D-J-L	1.57		5.83
								A-D-K-L	1.26		4.50

Risky path A-B-E-G-I-J-L
Worst case delay 6.83 weeks

Vraag 15. Oefening - Formele Specificaties [..../5]

In deze oefening ga je het gedrag van een intelligente lichtschakelaar modelleren aan de hand van een ***StateChart***. Het licht heeft 4 niveaus gaande van "0" (= uitgeschakeld) tot "3" (= maximum helderheid). Er is echter maar 1 knop om het licht te bedienen. We kunnen het gedrag van deze schakelaar als volgt beschrijven:

- Als het licht aan staat zal door één keer de knop in te duwen en los te laten het licht uitschakelen.
- Als het licht is uitgeschakeld zal door één keer de knop in te duwen en los te laten het licht aan gaan en het niveau van helderheid van voor de uitschakeling opnieuw aannemen.
- De knop indrukken en ingedrukt houden zal het helderheidsniveau doen stijgen als het voorheen steeg en doen dalen als het voorheen daalde.
- Als het hoogste helderheidsniveau bereikt is zal vanaf dan het lichtniveau dalen en als het laagste niveau bereikt is zal het terug stijgen.