dSoftark rapport nr. 5 Hold 1: Romeo
Thomas Pihlkjær – 20092289 & Simon Stenbæk Madsen - 20102187

36.48.1
Relevante betingelser der har indflydelse på bevægelsen af enheder antaget at max bevægelse på 1 og ingen mulighed for angreb:
1. Tiles er af typerne mountain eller oceans.
    1. Der er 5 typer tile, hvoraf 2 ikke kan have enheder på sig. Det vil sige at en enhed kun kan være på hills, forest og plains.
2. Tiles er optaget af venligsindet enhed.
    1. Boolean, enter/eller
3.Afstand mellem 'from' og 'to' tiles. Dette vil blive set bort fra.

36.48.2
Equivalance class table

Conditions	Invalids EC's	Valid EC's
Tile type	Mountains[ec1], ocean[ec2]	Plains[ec3], hills[ec4], forrest[ec5]
Tile occupied	True[ec6]	False[ec7]

Representation: Ved sets og booleans hvor hver værdi har sin egen EC, vil en defekt påvist med én værdi i en EC, naturligvis gælde for hele EC'en da denne kun har en værdi.
Coverage: alle input elementerne er i [ec1], [ec2], [ec3], [ec4] eller [ec5] samt i enten [ec6] eller [ec7]

36.48.3

EC's testet	Test Case	Forventet resultat
[ec3],[ec7]	Flyt til plains hvor 'to' er fri	True
[ec4],[ec7]	Flyt til hills, hvor 'to' er fri	True
[ec5],[ec7]	Flyt til forrest, hvor 'to' er fri	True
[ec1],[ec7]	Flyt til mountains, hvor 'to' er fri	False
[ec2],[ec7]	Flyt til ocean, hvor 'to' er fri	False
[ec3],[ec6]	Flyt til plains, hvor 'to' ikke er fri	False
[ec4],[ec6]	Flyt til hills, hvor 'to' ikke er fri	False
[ec5],[ec6]	Flyt til forrest, hvor 'to' ikke er fri	False

Myers' heuristikker er brugt for at finde frem til overstående test-cases. Først dækkes alle gyldige EC'er med så få tests som muligt. Dernæst kombineres hver ugyldig EC med en gyldig, til der ikk er flere udækkede ugyldige EC'er..
De tre første tests er fundet ved at kombinere de tre gyldige EC'er i 'Tile type' med den ene gyldig 'Tile occupied'. De næste to ved at kombinere de to ugyldige 'Tile type' med den gyldige 'Tile occupied'. De tre sidste ved at kombinere de gyldige 'Tile type' med den ugyldige 'Tile occupied'.

36.48.4
Test-cases fra AlphaCiv dækker ikke helt så godt som disse tests. Vi har testet [ec3][ec7], men ikke [ec4][ec7] eller [ec5][ec7]. Ligeledes har vi testet [ec3][ec6], men ikke [ec4][ec6] og [ec5][ec6].

Dette er sandsynligvis fordi, at disse tests ikke ville have medført ny kode. Tests der inkludere forrest er selvfølgelig ikke testet i AlphaCiv, men de er ikke blevet testet senere, i varianter der har forrest.

36.50.1

Funktion givet ved:

$$O([1]u_{from}^{type},[2]u_{to}^{type},[3]t_{from},[4]t_{to},[5]s_{from},[6]s_{to},[7]d_1,[8]d_2)$$

1. [3],[4], Tile typerne for 'from' og 'to', city overskriver typen af tile
    a) Tile bonus:
        - Plains = 0
        - Forrent, hills = 2
        - city = 3
    b) Der skal ikke testes for mountain or ocean, moveUnit søger for det.
2. [1],[2], Enhedernes type for 'from' og 'to'
    a) Archer, Legion, Settler.
3. [5], [6], Støtte enheder. [0,1...7]
    a) At angriber 'from' har mere ”support” bonus end forsvaren 'to'
    b) Defender og attacker har samme tile type, men defender har 1 mere i support (dvs total defence).
    c) Bonus til attacker skal kun udregnes fra attackers enheder, ligeledes ved defender.
4. [7],[8], udfaldet af terningerne. [0-6] postcondition for terning er 0 <= x <= 6
    a) For ikke at teste terninge-kast fastsætter vi 2 Ecs med terningeværdierne 1 og 2

36.50.2

For at dække [1] og [2] af O funktionen er der 2 ækvivalens typer:
a) Settler (kun defence)
b) Archer, Legion (har attack og defence)

Condition	Invalid EC	Valid EC
Settler		[ec1]Settler
Archer, LEgion		[ec2]Archer,Legion

For at dække [3] og [4] af O funktionen er der 3 ækvivalens typer:
(a) Plains (1 bonus)
(b) Forrest, hills (2 bonus)
(c) City (3 bonus)

Condition	Invalid EC	Valid EC
Tile Type: Plains		[ec3]Plains
Tile Type: Forrest || Hills		[ec4]Forest,Hills

Tile Type: City		[ec8]city

For at dække [5] og [6] af O funktionen er der 3 ækvivalens typer:

Condition	Invalid EC	Valid EC
Bonus	[ec5]<0, [ec6] >7	[ec7] 1 =< x => 7
No bonus		[ec9] x = 0

For at dække [7] og [8] af O funktionen er der 1 ækvivalens typer:

Condition	Invalid EC	Valid EC
Dice 1		[ec10]d=1
Dice 2		[ec11]d=2

36.50.3

EC's under test	Testcases	Forventet resultat
1: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec2$ $t_{from} = ec3$ $t_{to} = ec4$ $s_{from} = ec7$ $s_{to} = ec7$ $d_1 = ec10$ $d_2 = ec10$	Archer på plain uden support angriber Legion i forrest uden support. (ingen support bonus, dvs. tester terrain) Forventet resultat: Legion vinder. 3 total attack mod 4 i defence	False (angriber taber)
2:	Archer angriber Legion, begge står på plains og har ingen support.	True (angriber vinder)

$u_{from}^{type} = ec2$ $u_{to}^{type} = ec2$ $t_{from} = ec3$ $t_{to} = ec3$ $s_{from} = ec9$ $s_{to} = ec9$ $d_1 = ec10$ $d_2 = ec10$	Forventet resultat: Archer vinder da attack og defence er den samme	
3: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec1$ $t_{from} = ec3$ $t_{to} = ec4$ $s_{from} = ec7$ $s_{to} = ec7$ $d_1 = ec10$ $d_2 = ec10$	Archer på plains med support på 2 angriber Settler i hills uden support. (Tester at support bonus er specifik for spilleren, dvs. røde enheder kun giver bonus til røde enheder) Forventet resultat: Settler vinder. Archer = 5 attack mod 6 defence (Settler)	False
4: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec2$ $t_{from} = ec3$ $t_{to} = ec8$ $s_{from} = ec7$ $s_{to} = ec7$ $d_1 = ec10$ $d_2 = ec10$	Archer på plains med support bonus på 2 angriber Legion i by Forventet resultat: legion vinder, 5 attack 6	False

6: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec1$ $t_{from} = ec3$ $t_{to} = ec4$ $s_{from} = ec5$ $s_{to} = ec7$ $d_1 = ec10$ $d_2 = ec10$	Archer (1 support) angriber settler, begge er på plains. Forventet resultat: Archer burde vinde, men taber hvis support bonus ikke tælles med.	True
7: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec2$ $t_{from} = ec3$ $t_{to} = ec4$ $s_{from} = ec6$ $s_{to} = ec7$ $d_1 = ec10$ $d_2 = ec10$		Invalid
8: $u_{from}^{type} = ec2$ $u_{to}^{type} = ec2$ $t_{from} = ec3$ $t_{to} = ec4$ $s_{from} = ec7$ $s_{to} = ec7$ $d_1 = ec11$ $d_2 = ec10$	Archer på plain uden support angriber Legion i forrest uden support. (ingen support bonus, dvs. tester terrain), hvor archer slår 2 [ec11]	Archer vinder. (valid) 6 total attack mod 4 i defence

36.50.3 (7): Noter. Hvorfor vi ikke vil teste denne.

Når vi har en række, som support bonus, vil det være naturligt at teste de 3 ækvivalens klasser, [ec5]<0, [ec6] >7, [ec7] 1 =< x => 7 for at sikre 100% dækning. Men ec6, grænsen, går ud over hvad der kan testes. (se skema)

Rød Support	Blå Support	Blå Support	Blå Support
Rød Support	Rød	Blå	Blå Support

Rød Support	Blå Support	Blå Support	Blå Support

Dette skema angiver den maksimale bonus blå kan opnå, 7, og 8 er ikke opnåeligt ud fra beskrivelsen for O.
Derfor giver det ingen mening at teste denne boundery, da den maksimale defence rød kan opnå er 9, og blå minimalt kan opnå 10 i attack. Derfor hvis vi kunne få mere end 7 i support bonus ville udfaldet af slaget være det sammen, og derfor ville boundery testing på dette ikke haven nogen effekt.

36.50.4
De test-cases vi har skrevet tester same ækvivalens klasser flere gange, ec2 og ec4 i kryds, ydermere tester vi ikke kombinationen hvor ec4 (legion & archer) står på plains (ec3).
Dette betyder at vi ikke opnåede 100% test dækning.

36.51 (se afsnit 36.50.3 (7))

36.30
Der er blevet brugt Decorator pattern da det passer ind i tankegangen til ”change by addition, not by modification” og hvilket er et godt match i dette tilfælde.
Selve funktionaliteten er der kun tilføjet til og ikke ændret på som det nu hører sig til i Decorator pattern.
Måde det virker på er at game kommer med som parameter i konstruktøren i den nye game class. Derfra kalder man så metoderne fra det gamle game. Fordelen her er så at det er nemmere at lave fordi det eneste man lige skal være opmærksom på er at huske at kalde metoderne fra det gamle game også det returner det samme. Hvis man husker på dette så kan man lave meget mere end blot dette.

36.31
Da vores DeltaCivWorld-Stategy kun tillader eksterne maps gennem selve source koden, har vi valgt at ikke at benytte denne strategi da den ikke på fornuftig vis kan kobles sammen med DeltaCiv koden uden at ændre den. Tredjepartsklassen passer ikke sammen med vores design, og vi har derfor lavet en adaptor, der implementerer MapStrategy, og benytter algoritmen fra ThirdPartyFractalGenerator. Dette gøres for ikke at ændre den eksisterende kode, samt ikke være for afhængig af tredjepartskode. Ydermere tager vi heller ikke hensyn til at Unit og Cities kan ved uheld placeres på en ocean- eller mountaintile.