dSsoftark rapport nr. 2 Hold 1: Romeo
Thomas Pihlkjær – 20092289 & Simon Stenbæk mAdsen - 20102187

Test-list til diverse Civ-former:

Her er den test-liste vi har fundet frem til og lavet test cases for de kraven vi fandt fra s. 462 til s.465: Samtlige krav til AlphaCiv,BetaCiv,DeltaCiv og GammaCiv er blevet skrevet tests til, og disse tests er blevet gennemført som del af TDD processen

AlphaCiv:

The game should place unit clockwise starting from north at city at 1x1
Test: gameShouldPlaceUnitsClockwiseStartingFromNorthAtCity1x1

Once a city is captured its current production is changed to null
Test: productionShouldRemainAfterCityIsCaptured()

Red´s unit attacks Blue's city and captures it
Test: productionShouldRemainAfterCityIsCaptured()

Unit cost is deducted from city production treasury
Test: gameShouldDeductUnitCostFromCity()

when a city is captured it still has its productionrates
Test: Test: productionShouldRemainAfterCityIsCaptured()

BetaCiv:

Red player has the city at (4,1) – Red player is the winner
Test: redPlayerShouldWinIfAllCitiesAreHis()

Its age 4000BC I - Next turn it is age 3900BC
Test: ageShouldAdvanceTimeBy100AtEndOfFirstRound()

Its age 100BC - Next turn it is age 1BC
Test: ageShouldAdvanceFrom100BCTo1BC()

Its age 1BC - Next turn it is age 1AD
Test: ageShouldAdvanceFrom1BCTo1AD()

Its age 1AD - - Next turn it is age 50AD
Test: ageShouldAdvanceFrom1ADTo50AD()

Its age 1700AD - - Next turn it is age 1750AD
Test: ageShouldAdvanceFrom1700ADTo1750AD()

Its age 1875AD - - Next turn it is age 1900AD
Test: ageShouldAdvanceFrom1875ADTo1900AD()

Its age 1900AD – Next turn it is age 1905 AD

dSsoftark rapport nr. 2 Hold 1: Romeo
Thomas Pihlkjær – 20092289 & Simon Stenbæk mAdsen - 20102187

Test: ageShouldAdvanceFrom1900ADTo1905AD()

Its age 1970AD - - Next turn it is age 1971AD
Test: ageShouldAdvanceFrom1YearFrom1970AD()

Initially no player should have won
Test: initiallyNoPlayerShouldHaveWon()

Age initially be 4000BC
Test: startAgeShouldBe4000BC()

DeltaCiv:

There should be a city at position 8_12
Test: thereShouldBeACityAt8_12()

There should be a mountain at 0_5
Test: thereShouldBeAMountainAt0_5()

There should be a forest at 5_5
Test: thereShouldBeAForestAt5_5()

There should be a plain at 5_6
Test: thereShouldBeAPlainAt5_6()

There should be a tile at every coordinate
Test: thereShouldBeATileAtEveryCoordinate()

There should be 89 ocean tiles on the map
Test: thereShouldBe86OceansOnTheMap()

There should be 10 hills tiles on the map
Test: thereShouldBe10HillsOnTheMap()

There should be 137 tiles tiles on the map
Test: thereShouldBe137PlainsOnTheMap()

There should be 14 forest tiles on the map
Test: thereShouldBe14ForestsOnTheMap()

There should be 9 mountain tiles on the map
Test: thereShouldBe9MountainsOnTheMap()

GammaGiv:

A Settler can create a new city
Test: settlerMakesCityWhenActionIsPerformed()

archers should have defence of 3 after 2 actions
Test: archersShouldBeHaveDefenceOf3After2Actions()

archers should have defence of 6 after action
Test: archersShouldHaveDefenceOf6AfterAction()

archers should have defence of 3 before actions
Test: archersShouldHaveDefenceOf3BeforeAction()

archers should not be able to move after action
Test: archersShouldNotBeAbleToMoveAfterAction()

fortified archers movement should not be restored
Test: ArcherIsFortifiedAndCannotMoveAfterActionIsPerformed()

red settler should not be able to perform action on blue players turn
Test: redSettlerShouldNotBeAbleToPerformActionOnBluePlayersTurn()

1)

Vi startede med at funktionalitet ud af GameImpl og ind i 5 nye interfaces som er CivWinStrategy,CivUnitStrategy, CivMapStrategy,CivActionStrategy og CivAgeStrategy. Disse har så konkrete klasser.
For eksempel har CivAgeStrategy: AlphaCivAgeg og BetaCivAge der styrer hvordan tiden forløber ivarianterne AlphaCiv og BetaCiv hvor det før var styret af GameImpl.
Det vil sige at ansvaret for blandt andet hvordan tiden forløber er rekfaktoriseret udaf GameImpl og ind i et Strategy-Pattern for at kunne styre flere Civ-varianter på samme tid.
Det samme blev gjord ved map-varianterne. Dette blev gjort fordi de forskellige algoritmer blev brugt til samme arbejde. Specifikt ved DeltaCiv lød opgaven at man kunne skrive algoritmer for at generer maps. Hvilket blev løst ved 2 metoder i DeltaCivMap-klassen som generer det ønskede map.

2)

Udfra definition af variability points fandt vi følgende points:

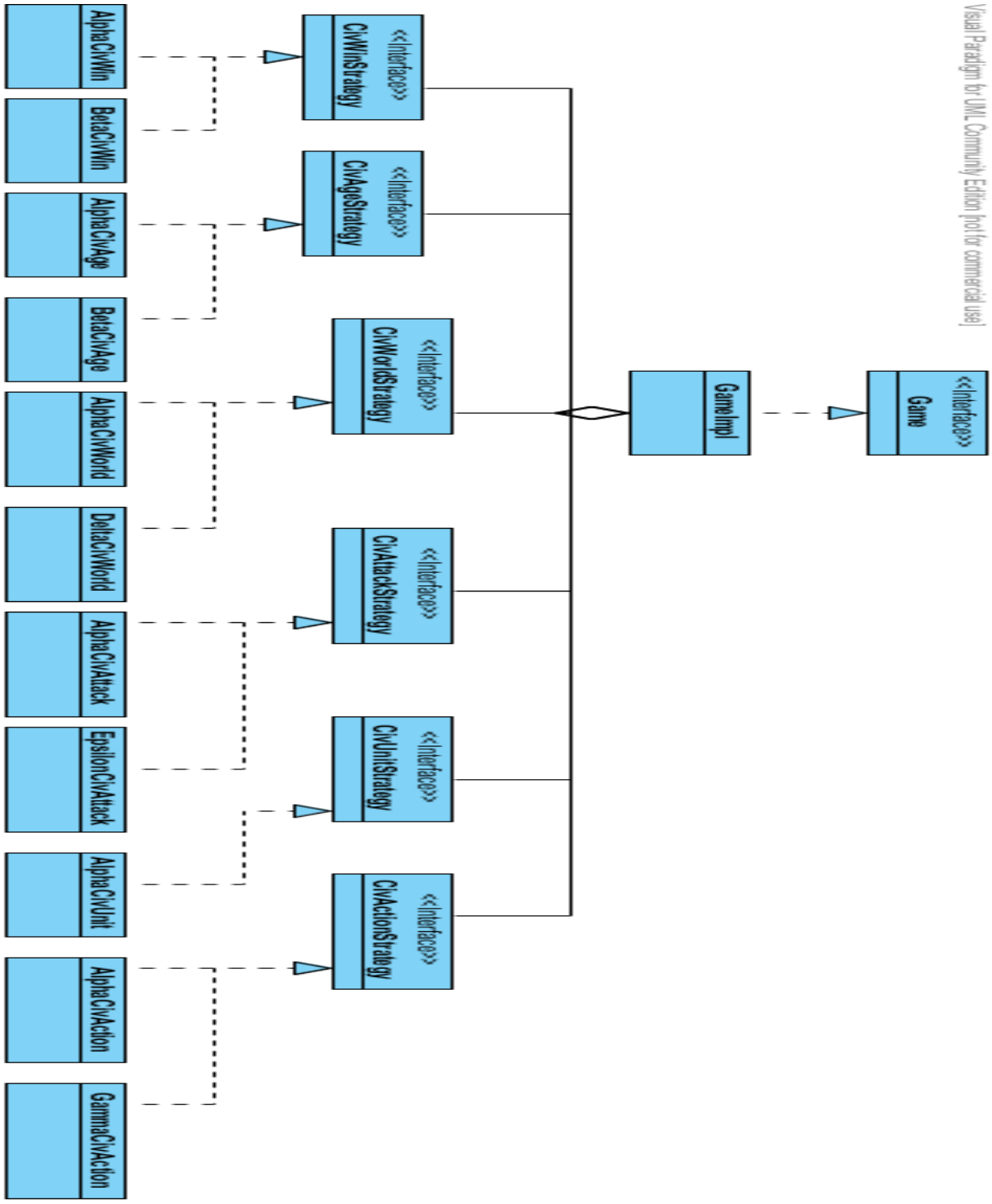
1.CivUnit/CivActions
2. CivMap
3. CivWin
4. CivAge

Både i CivUnit såvel som i CivAction fulgte 3-1-2 således at vi identificerede udfra specifikationerne fra de forskellige opgaver at CivUnit og CivAction er unikt fra variant til variant.Efter vi har fundet vores varians skulle vi nu ifølge Punkt 1 og 2 i 3-1-2 programmerer interfaces som blev til CivUnitStrategy og CivActionStrategy. Grunden til at vi ikke valgte at bruge inheritance så meget igen er at vi skal genbruge implementationerne fra både CivUnitStrategy og CivActionStrategy i varianterne. Dette ville gøre kodning svære selvom vi bruger inheritance i GammaCivUnit som extender AlphaCivUnit fordi de alligevel skulle kunne det samme.

Udover genbrug af varianter bindes den konkrete variant ved compile-time hvilket ikke gør det muligt at give den pågældende unit nye actions i run-time, dog er dette endnu ikke et problem. Den største fordel ved at bruge strategypattern er at man uddelegere ansvar til flere klasser.

3)

Det endelige UML klasse diagram generet via Visual Paradigm 10.0



36.11

1:
Vi har ikke et nøjagtigt tal men går udfra at der i implementationerne af unit-interfacet(settler-unit og archer unit-klasserne) er flere relationer mellem instances af unit.

2:
At i den situaton med at action-metode findes i settler unit eller archer unit klassen skal objektet og derved unit der selv fjerne sig selv fra verdenen hvorimod i startegy-baseret løsning er det spillet er fjerner den pågældende unit.

3:
I tilfælde med archer og settler-unit klasserne skal unit-klasserne bede spillet/verdenen om at oprette en by på settlerens position hvor i strategy tilfældet kan spillet/verdenen oprette byen og derved ikke ende i en situation hvor den unit der skal oprette byen er fjernet fra verdenen.

Backlog til næste gang:

- Flere kommentarer i koden såsom forklaring til metoder osv.

NOTE

Ignorer EpsilonTest grudnet at vi er gået i gang med nsæte iteration