# Σχεδίαση Ελέγχων

Οι έλεγχοι που σχεδιάσθηκαν και εντάχθηκαν στην υλοποίηση περιγράφονται εδώ.

## Ελεγχος use cases via system tests

### Traceability Matrix

Η αντιστοίχιση use cases σε id’s φαίνεται συνοπτικά εδώ:

|  |  |
| --- | --- |
|  |  |
| UC1 | Load Data |
| UC2 | Analyze TimeLine |
| UC3 | Present TimeLine |
| UC4 | Present Phases |

Ο Πίνακας 3 είναι ο traceability matrix για τους ελέγχους μας. Στη συνέχεια, οι έλεγχοι επεξηγούνται πιο αναλυτικά.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC3 | UC4 |
| T1\_V0\_01 | X |  |  |  |
| T1\_V0\_02 | X |  |  |  |
| T2\_V0\_01 |  | X |  |  |
| T2\_V0\_02 |  | X |  |  |
| T3\_V0\_01 |  |  | X |  |
| T3\_V0\_02 |  |  | X |  |
| T4\_V0 |  |  |  | X |

Πίνακας 3 Traceability matrix between use cases and tests

### **Use case UC1: load data**

**Involved methods**

MainEngine.setTimeLine()

IParser.parse --> SimpleTextParser.parse(filename)

**Test cases**

|  |  |  |
| --- | --- | --- |
| ID | T1\_V0\_01 | HappyDayScenario for SimpleTextParser.parse() |
| Description | ON | any context |
|  | RECEIVING | Request to parse a specific txt file with a valid timeline |
|  | ENSURE | That the System |
|  | OUTPUTS | A timeline with the correct size and no offending <time,value> pairs |
|  | SUCH THAT | state is intact |
| Pre-cond. |  | No specific precond constructed |
| Input |  | input\_test.txt, a small file with less than 10 entries, all valid |
| Output |  | a timeline with the same #entries as the contexts of input\_test.txt and no offending values |
| Post-cond. |  | No state properties tested |
| Method To test |  | SimpleTextParser.parse(filename) |

|  |  |  |
| --- | --- | --- |
| ID | T1\_V0\_02 | HappyDayScenario for MainEngine.setTimeLine() |
|  |  | …identical setup with T1\_V0\_01 |
| Method To test |  | MainEngine.setTimeLine(filename) |

**Not designed yet**: T1\_V1: missing file, T1\_V2: invalid values in input file

### **Use case UC2: analyze timeline (and produce phases)**

**Involved methods**

MainEngine.producePhases()

AnalyserFactory.createAnalyzer()

IAnalyzer --> NaiveAnalyser.producePhasesFromTimeLine (TimeLine)

**Test cases**

|  |  |  |
| --- | --- | --- |
| ID | T2\_V0\_01 | HappyDayScenario for NaiveAnalyser.producePhases() |
| Description | ON | A time line having being loaded |
|  | RECEIVING | Request to analyze a valid timeline into phases |
|  | ENSURE | That the System |
|  | OUTPUTS | a set of phases |
|  | SUCH THAT | state is intact |
| Pre-cond. |  | Load input\_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, and produce timeline |
| Input |  | the abovementioned timeline |
| Output |  | A correct #phases, with the correct points inside |
| Post-cond. |  | No state properties tested |
| Method To test |  | NaiveAnalyser.producePhasesFromTimeLine (TimeLine) |

|  |  |  |
| --- | --- | --- |
| ID | T2\_V0\_02 | HappyDayScenario for MainEngine.producePhases() |
|  |  | …identical setup with T2\_V0\_01 |
| Method To test |  | MainEngine.producePhases() |

**Not designed yet**: T2\_V1: null time line, T2\_V2: timeline with only one phase

### **Use case UC3: Visualize timeline**

**Involved methods**

MainEngine.visualize()

**Test cases**

|  |  |  |
| --- | --- | --- |
| ID | T3\_V0\_01 | HappyDayScenario for MainEngine.visualize() |
| Description | ON | A time line having being loaded |
|  | RECEIVING | Request to visualize a valid timeline via a specific visualizer |
|  | ENSURE | That the System |
|  | OUTPUTS | An appropriate visualization |
|  | SUCH THAT | state is intact |
| Pre-cond. |  | Load input\_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, and produce timeline |
| Input |  | the abovementioned timeline, “HtmlVisualizer” as the tested visualizer |
| Output |  | A correct visualization, expressed as a 2D raster of chars |
| Post-cond. |  | No state properties tested |
| Method To test |  | MainEngine.setVisualizer(String)  MainEngine.visualize() |

|  |  |  |
| --- | --- | --- |
| ID | T3\_V0\_02 | HappyDayScenario for MainEngine.visualize() |
|  |  | …identical setup with T2\_V0\_01 |
| Input |  | “ConsoleVisualizer” as the tested visualizer |

### **Use case UC4: Visualize PHASES (TO CONSOLE)**

**Involved methods**

MainEngine.reportPhases()

NaiveAnalyser.reportToConsole()

Phase.consoleVerticalReport()

**Test cases**

|  |  |  |
| --- | --- | --- |
| ID | T4\_V0 | HappyDayScenario for MainEngine.reportPhases() |
| Description | ON | A time line having being loaded |
|  | RECEIVING | Request to visualize the phases of the timeline |
|  | ENSURE | That the System |
|  | OUTPUTS | An appropriate visualization for the phases |
|  | SUCH THAT | state is intact |
| Pre-cond. |  | Load input\_test.txt, a small file with less than 10 entries, all valid, for o(5) phases, produce timeline and analyze it to phases |
| Input |  | the abovementioned timeline and its phases |
| Output |  | A correct visualization (approximation: the size of the descriptions produced is the same with the number of phases) |
| Post-cond. |  | No state properties tested |
| Method To test |  | MainEngine. reportPhases() |

## Unit tests

Στην παρούσα υποενότητα, παραθέτουμε πολύ συνοπτικά τα πιο σημαντικά unit test που σχεδιάστηκαν, οργανωμένα ανά κλάση. Δίνουμε μόνο το OREOS text.

**Class ParserFactory**

ON any context, RECEIVING a request for a new parser, ENSURE THAT THE SYSTEM OUTPUTS a not null parser SUCH THAT state remains intact.

**Class AnalyzerFactory**

ON any context, RECEIVING a request for a new analyzer, ENSURE THAT THE SYSTEM OUTPUTS a not null analyzer of the correct kind SUCH THAT state remains intact.

## ΕΚΚΡΕΜΟΤΗΤΕΣ (ToDo)

Εκκρεμούν μη υλοποιημένοι έλεγχοι ως ακολούθως (αν υπάρχουν εκκρεμότητες, παραθέστε την TODO λίστα ελέγχων που πρέπει να ετοιμαστούν)

1. Unit tests are missing for several classes, both at the model and at the business logic level