**“Software Engineering”**

**Course**

**a.a. 2016-2017**

**Template version 1.0**

**Lecturer: Prof. Henry Muccini (henry.muccini@univaq.it)**

**Planner Path Calculator version v2  Deliverables**

|  |  |
| --- | --- |
| **Date** | <27/11/2016> |
| **Deliverable** | 1 |
| **Team (Name)** | TV-LCD |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Tiziano Santilli | *232412* | *tizianosantilli@gmail.com* |
| Vincenzo Apostolo | *236112* | *apos.vincenzo@gmail.com* |
| Luca Grillo | *236763* | *lucag.8595@gmail.com* |
| Cristian Capozucco | *237945* | *cristcapo@gmail.com* |
| Davide Mariotti | *238185* | *davi95ma@hotmail.it* |

**Project Guidelines**

[do not remove this page]

*This page provides the Guidelines to be followed when preparing the report for the Software Engineering course. You have to submit the following information:*

♣ *This Report*

♣ *Diagrams (Analysis Model, Component Diagrams, Sequence Diagrams, Entity Relationships Diagrams)*

♣ *Effort Recording (Excel file)*

***Important:***

♣ *document risky/difficult/complex/highly discussed requirements*

♣ *document decisions taken by the team*

♣ *iterate: do not spend more than 1-2 full days for each iteration*

♣ *prioritize requirements, scenarios, users, etc. etc.*

Project Rules and Evaluation Criteria

***General information:***

♣ *This homework will cover the 80% of your final grade (20% will come from the oral examination).*

♣ *The complete and final version of this document shall be not longer than 40 pages (excluding this page and the Appendix).*

♣ *Groups composed of five students (preferably).*

*I expect the groups to submit their work through GitHub*

***Use the same file to document the various deliverable. Document in this file how Deliverable “i+1" improves over Deliverable “i".***

**Project evaluation:**

*Evaluation is not based on “quantity” but on “quality” where quality means:*

♣ *Completeness of delivered Diagrams*

♣ *(Semantic and syntactic) Correctness of the delivered Diagrams*

♣ *Quality of the design decisions taken*

♣ *Quality of the produced code*

Table of Contents of this deliverable

Pagina 4: List of Challenging/Risky Requirements or Tasks

Pagina 5-6-7-8-9: Requisiti funzionali, use case diagram e CockBurn’s template per la descrizione degli use case

Pagina 10: Requisiti non funzionali

Pagina 11: Content e Assumptions

Pagina 12-13: Prioritizzazione dei requisiti

Pagina 14-15: Analysis model

Pagina 16: Software Architecture: component diagram

Pagina 20: Design decision

Pagina 22: Effort Recording

List of Challenging/Risky Requirements or Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenging Task** | **Date the task is identified** | **Date the challenge is resolved** | **Explanation on how the challenge has been managed** |
| Gestione concorrente di più richieste | 28/10/2016 |  |  |
| Mantenimento delle performance elevate | 28/10/2016 | 21/11/2016 | Abbiamo notato che la tecnologia NOSQL a grafi, mediante il DBMS Neo4j, la creazione del grafo, nonché il calcolo del cammino con la relativa somma dei valori degli attributi, è molto più veloce della tecnologia SQL (testata con MYSQL Workbench). Con quest’ultima tecnologia il grafo veniva creato in circa un’ora. Con la tecnologia NOSQL il grafo viene creato in pochi minuti |
| Tecnologie da usare | 28/10/2016 | 21/11/2016 | Effettuando vari test sulle performance tra le varie tecnologie |
| Organizzazione Logistica Gruppo | 28/10/2016 | 28/10/2016 | Abbiamo deciso di prendere appuntamenti fissi durante la settimana |
| Gestione Grafo | 28/10/2016 | 23/11/2016 | Il grafo verrà gestito interamente dal database |
| Connessioni Parte web/ logica/DB | 28/10/2016 |  |  |

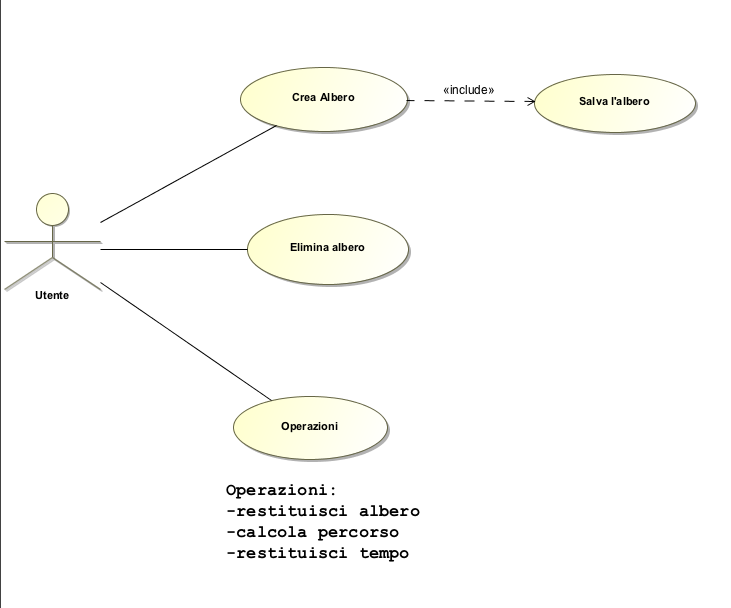
A. Requirements Collection

***A.1 Functional Requirements***

1) L’utente crea l’albero premendo il pulsante “Crea albero” e automaticamente salva l’albero.

2)L’utente deve poter cancellare l’albero premendo il pulsante “Cancella albero”.

3)Dopo la pressione del pulsante “calcola percorso” da parte dell’utente, il sistema trova l’albero, calcola il percorso, restituisce la somma degli attributi e il tempo impiegato per queste operazioni.



|  |  |  |
| --- | --- | --- |
| USE CASE 1 | Crea albero | |
| Goal in Context | L’utente interagisce con il sistema e si aspetta la creazione di un albero | |
| Scope & Level | Primary Task | |
| Preconditions | L’utente si trova nella intefaccia grafica e conosce i dati da immettere | |
| Success End Condition | L’albero viene creato senza imprevisti | |
| Failed End Condition | L’albero non viene creato | |
| Primary | Utente | |
| Trigger | L’utente seleziona l’opzione “Crea albero” nella home page | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente seleziona l’opzione “Crea albero” nella home page |
|  | 2 | L’utente immette i dati |
|  | 3 | L’utente preme il pulsante “Crea albero” |
|  | 4 | L’albero viene creato |
|  | 5 | L’utente viene avvisato della corretta creazione dell’albero |
| EXTENSIONS | Step | Branching Action |
|  | 4a | L’albero non viene creato:  Lancia messaggio di errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 2 | Salva albero | |
| Goal in Context | L’albero viene salvato in una struttura dati | |
| Scope & Level | Primary Task | |
| Preconditions | L’utente ha creato con successo un albero | |
| Success End Condition | L’albero viene correttamente salvato | |
| Failed End Condition | L’albero non viene correttamente salvato | |
| Primary | Utente | |
| Trigger | L’albero è stato creato | |
| DESCRIPTION | Step | Action |
|  | 1 | L’albero è stato creato |
|  | 2 | L’albero viene salvato |
|  | 3 | Ritorna un messaggio di conferma |
| EXTENSIONS | Step | Branching Action |
|  | 1a | L’albero non è stato creato:  Lancia un messaggio d’errore |
|  | 2a | L’albero non viene salvato:  Lancia un messaggio d’errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 3 | Operazioni | |
| Goal in Context | Il Sistema calcola le operazioni scelte dall’utente | |
| Scope & Level | Summary | |
| Preconditions | Gli alberi sono stati salvati e l’utente si trova nell’interfaccia grafica e conosce i dati da immettere | |
| Success End Condition | Restituzione del risultato | |
| Failed End Condition | Le operazioni non viene eseguita | |
| Primary | Utente | |
| Trigger | L’utente immette i parametri per l’operazione | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente immette i parametri per l’operazione |
|  | 2 | L’utente fa partire l’operazione |
|  | 3 | Il sistema esegue le operazione |
|  | 4 | Il sistema restituisce il risultato |
| EXTENSIONS | Step | Branching Action |
|  | 1.a | In mancanza di paramentri: Genera Errore |

|  |  |  |
| --- | --- | --- |
| USE CASE 4 | Elimina albero | |
| Goal in Context | Il sistema elimina l’albero | |
| Scope & Level | Subfunction | |
| Preconditions | L’utente si trova nell’interfaccia grafica e conosce l’id dell’albero da eliminare | |
| Success End Condition | L’albero è stato eliminato con successo | |
| Failed End Condition | L’albero non è stato eliminato | |
| Primary | Utente | |
| Trigger | L’utente preme sul pulsante “Elimina albero” | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente preme sul pulsante “Elimina albero” |
|  | 2 | L’albero viene eliminato |
| EXTENSIONS | Step | Branching Action |
|  | 2.a | L’albero non viene eliminato:  lancia messaggio di errore |

A1.1 GUI Requirements (da riempire a partire dalla Versione 2)

A1.2 Business Logic Requirements (da riempire a partire dalla Versione 2)

A1.3 DB Requirements (da riempire a partire dalla Versione 2)

***A.2 Non Functional Requirements***

1) Implementare la GUI come un’interfaccia Web con tecnologia HTML5

2) Il DBMS scelto deve essere open-source

3) La sicurezza non è necessaria: il sistema verrà inserito in un ambiente web sicuro

4) Il sistema deve essere veloce nel tornare i risultati delle operazioni di calcolo del percorso: il tempo di risposta deve essere entro i 30 secondi per un albero da 1 milione di vertici e 60 secondi per 2 milioni di vertici. Oltre i 2 milioni di vertici accettiamo un andamento esponenziale dei tempi di risposta.

5) Affidabilità: il sistema non deve andare in crash con la gestione concorrente di 10-100 utenti in parallelo

***Usability:*** La GUI deve essere il più possibile User-Friendly. Esagerata attenzione all’estetica non è richiesta poiché il sistema sarà usato solamente all’interno dell’azienda.

***A.3 Content***

Per semplicità dividiamo il nostro progetto in tre parti:

Parte web.

Parte logica.

Parte DB.

Per la parte web useremo l’html 5, xampp (una distribuzione Apache gratuita contenente anche PHP) e come Ide per la gestione della parte web useremo Eclipse.

Per la parte logica useremo Eclipse come Ide per il linguaggio java.

Per la parte logica useremo (alto fattore di rischio) il database NOSQL neo4j basato su grafi.

***A.4 Assumptions***

-L’hardware del server è abbastanza potente da sostenere l’esecuzione concorrente di 100 utenti.

-La creazione del grafo può impiegare tempi che vanno da pochi minuti ad al massimo un paio d’ore.

-Decidiamo di creare un’interfaccia adatta all’azienda dal punto di vista grafico e della facilità d’uso.

-Utilizziamo il linguaggio java per la sua portabilità, per le strutture dati incluse nelle librerie, per la facilità di connesione con il DBMS e per la conoscenza acquisita durante gli studi.

-Utilizzo del DBMS NoSQL “Neo4J” a grafi. Le prestazioni, all’aumentare del volume dei dati, sono migliori rispetto a un DBMS SQL.

***A.5 Prioritization***

*Requisiti listati secondo ordine di priorità.*

1)**PARAMETRI GRAFO:**

* **Split Size**: Quanti vertici devo generare da un vertice dato
* **Depth**: Profondità dell’albero
* **AttributeList**: : lista di attributi da associare a vertici e archi
* *VertexAttributeList*: per esempio Ptime
* *EdgeAttributeList*: per esempio costo
* **AttributeValueGenerationRule:** servono per dire al sistema come generare i numeri. Possibile setup:
* **Random(K-N): to** Genera un valore random tra K e N

La GUI può essere usata per generare un albero alla pressione del tasto “Build tree”.

Il risultato è salvato nel database.

2) **SCELTA DB:**

Il team del progetto dovrà selezionare la tecnologia di database da utilizzare. Lo schema del database sottostante si basa su MSSQL; nel caso in cui il team scelga un database diverso dovrà adattare lo schema alla tecnologia scelta. Il team dovrà motivare la scelta della tecnologia del database come parte delle loro decisioni di progettazione.

Altri DBMS possono essere scelti a condizione che siano open-source.

3)**OPERAZIONI SU GRAFO**:

L’interfaccia deve essere usata per reperire un grafo salvato sul databese: gli utenti possono selezionare 2 vertici A e B, e il Sistema deve ritornare la lista dei vertici da “A” a “B” insieme alla somma di ogni attributo.

4)**DIMENSIONE MASSIMA GRAFO:**

Il Sistema deve essere in grado di gestire 2 milioni di vertici

5)**NUMERO UTENTI CHE USANO IL SISTEMA**

Ci aspettiamo 10-100 utenti che usano il sistema concorrentemente. Il sistema deve essere in grado di funzionare senza nessun impatto sulle prestazioni

6)**PERFORMANCE**

La GUI deve anche mostrare il tempo di esecuzione del calcolo.

Dato che questo sistema sarà interrogato da un sistema di produzione,

dovrebbe essere "veloce" per restituire il risultato

7)**INTERFACCIA**

La GUI deve essere implementata come un’interfaccia web basata su tecnologia HTML5

8)**SICUREZZA**

Sicurezza: non abbiamo bisogno di requisiti sulla sicurezza in quanto il Sistema sarà integrato in un Sistema web sicuro.

B. Analysis Model

**Boundary Object:**

**Interfaccia crea albero:**

Si ritiene necessario inserire il boundary “interfaccia crea albero” in quanto l’utente deve interfacciarsi con il sistema per poter avere l’opzione di creare un nuovo albero.

**Interfaccia elimina albero:**

si ritiene necessario inserire il boundary “interfaccia elimina albero” in quanto l’utente deve interfacciarsi con il sistema per poter avere l’opzione di eliminare un albero creato in precedenza.

**Interfaccia operazioni:**

si ritiene necessario inserire il boundary “interfaccia operazioni” in quanto l’utente deve interfacciarsi con il sistema per poter avere l’opzione di inserire i dati per poter eseguire le operazioni necessarie per ricevere il risultato.

**Controller Object:**

**Crea e salva:**

Attività del sistema che mette a disposizione gli elementi necessari per la creazione di un albero. Una volta creato correttamente, ci sarà l’azione di salvataggio.

**Eliminazione:**

Attività del sistema che mette a disposizione gli elementi necessari per l’eliminazione di un albero creato in precedenza.

**Calcola:**

Intendiamo “calcola” come il controller che calcola l’insieme dei risultati delle varie operazioni.

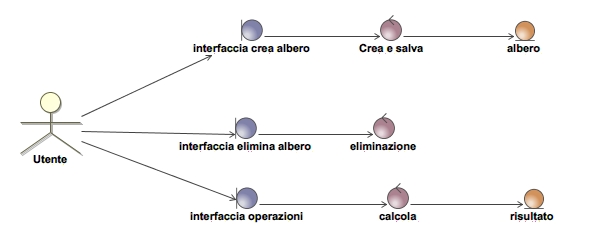
**Entity Object:**

**Albero:**

La creazione dell’albero, sul quale lavoreremo, è un entity object.

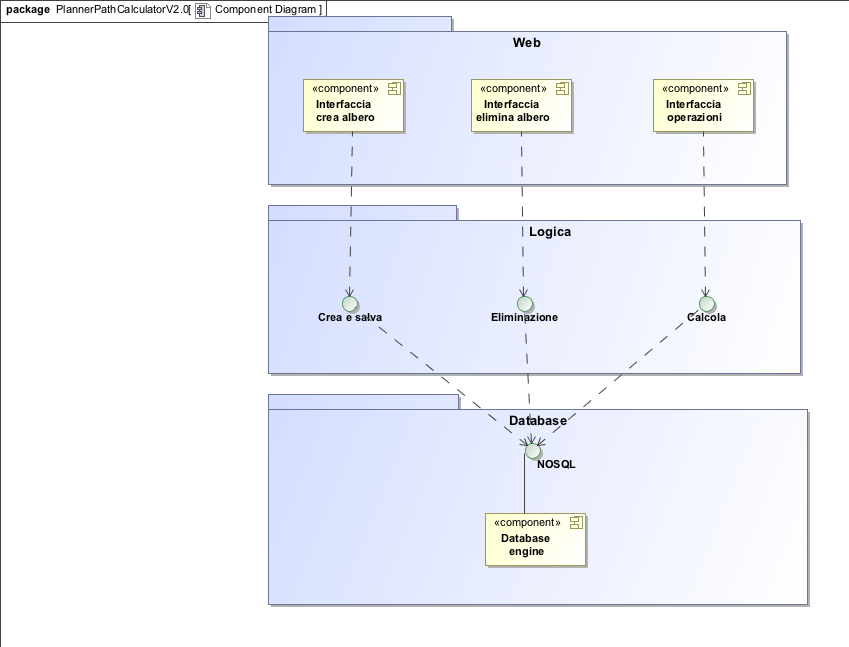
**Risultato:**

il risultato è un entity object e definisce l’insieme dei dati che verranno restituiti in seguito all’elaborazione delle operazioni.



C. Software Architecture

***C.1The static view of the system: Component Diagram***

**

***C.2 The dynamic view of the software architecture: Sequence Diagram***

*BE SURE THAT THE STRUCTURE IS SYNCHRONIZED WITH THE DYNAMIC VIEW*

D. ER Design

<Report here the Entity Relationship Diagram of the system DB>

E. Class Diagram of the implemented System

F. Design Decisions

**Scelta DB**: Abbiamo notato che mediante la tecnologia NOSQL a grafi (DBMS Neo4j), la creazione del grafo, nonché il calcolo del cammino con la relativa somma dei valori degli attributi, è molto più veloce della tecnologia SQL (testata con MySQL workbench). Con quest’ultima tecnologia il grafo veniva creato in circa un’ora. Con la tecnologia NOSQL il grafo viene creato in pochi minuti.

**Scelta Linguaggio**: Abbiamo deciso di utilizzare il linguaggio java per la sua portabilità, per le strutture dati incluse nelle librerie, per la facilità di connessione con il DBMS e per la conoscenza acquisita durante gli studi.

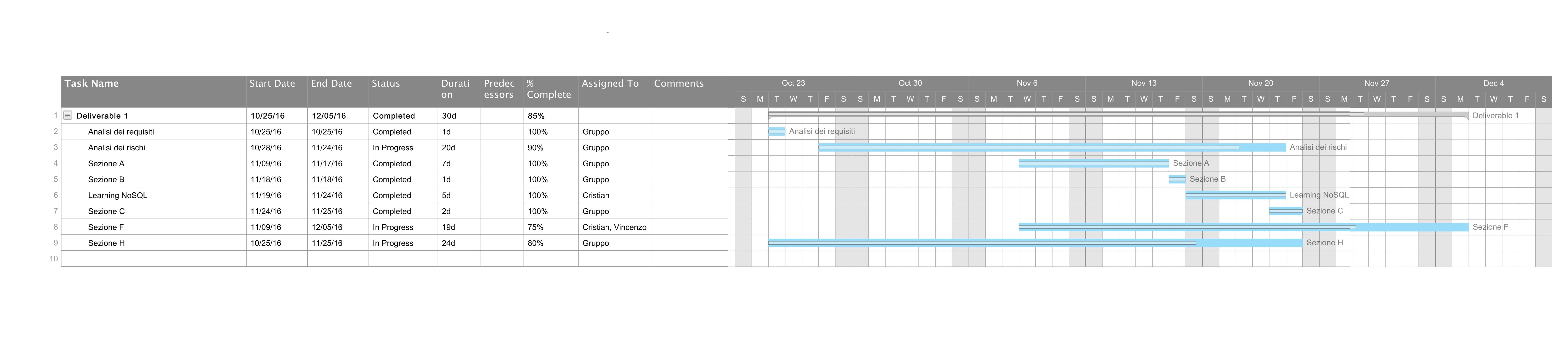
**Scelta “Dove eseguire il calcolo del cammino”**: abbiamo deciso che il calcolo verrà effettuato sul database. Questo perché non si ha possibilità di errore di effettuare scritture concorrenti ma solo letture. Inoltre si presume che l’hardware del server sia abbastanza potente da sostenere l’esecuzione concorrenti di più utenti.

G. Explain how the FRs and the NFRs are satisfied by design

<Report in this section how the design you produced satisfies the FRs and the NFRs>

H. Effort Recording

***GANTT****Make a GANTT documenting the tasks and timing you expect to spend on the deliverable. Try to be as precise as possible. Check, after the deliverable deadline, if and how you satisfied (or not) the deadlines.*

**

***Logging*** *As you are working on the assignment, record what you are doing and how long you spent. As a rule of thumb, you should add a log entry every time you switch tasks. For example, if you do something for two hours straight, that can be one log entry. However, if you do two or three things in half an hour, you must have a log entry for each of them. You do not need to include time for logging, but should include the time spent answering the other parts of this question.*

*For this purpose, please use the* ***LogTemplate.xls*** *file.*

***Categorization****When logging the time spent on the project, please create different sub- categories.**Specifically, it is important to clearly distinguish between two main categories: the time spent for “****learning****” (the modeling languages, the tools, etc.) from the time needed for “****doing****” (creating the models, taking the decisions, …). Learning tasks are in fact costs to be paid only once, while doing costs are those that will be repeated through the project.*

*For each category, please define sub-categories. Examples follow. You may add other sub-categories you find useful.*

|  |  |
| --- | --- |
| ***Learning***   * + ***Requirements Engineering***   + ***Non functional Requirements***   + ***Use Case Diagrams***   + ***Tool study*** | ***Doing:***   * + ***Requirements discovery***   + ***Requirements Modeling (UC diagrams)*** |

***Summary Statistics****Based on the attributes defined above, calculate the summary statistics of the time spent for “learning”, the time spent for “doing”, and the total time.*

***Note: this Deliverable report shall document only the Summary Statistics for the different deliverables (D1, D2, and Final). Detailed information shall be reported in the Excel file.***

***COPY HERE (computed from the spreadsheet): i) the total number of hours spent by the group (that is, hours per task X number of people working on that task), ii) the time spent for LEARNING and for DOING***

Appendix. Code

<Report in this section a **documented** version of the produced code>