**Relazione Trasformazione Elementare Progetto MBSE:** **Mr. Matteo Di Gioacchino**

Si prenda in considerazione quanto avevamo detto nel file [2. Acceleo transformation](https://docs.google.com/document/d/16hCV8mb4f5a0WSXzdrQJfg3KcRw0L8E7ZEMQ8G4djrM/edit?tab=t.0#heading=h.d2lprw9hchgx) sul [Drive](https://drive.google.com/drive/folders/1EyU0rzzRajhknTcrU7c0pRsNBEFOwAjg):

A screenshot of a computer program

AI-generated content may be incorrect.

A text on a white background

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

L’obiettivo di questo lavoro iniziale era quello di realizzare un **Proof of Concept** per dimostrare la possibilità di poter strutturare il progetto seguendo le linee guida di cui sopra, a tal fine, si è pensato di realizzare una trasformazione **Acceleo** che prendesse in input un semplice **Deployment Diagram** annotato con il **Profilo UML** presente sul <https://github.com/MatteoStromieri/UML-to-TOSCA>, in sintesi, esso permette di annotare ogni nodo **“esterno”** (e quindi **service\_template**) con lo stereotipo **Hardware**, ogni nodo **ExecutionEnvironment** con lo stereotipo **Piattaforma** e ogni nodo **Artifact** con lo stereotipo **Software**.  
La trasformazione implementata ha la seguente struttura:  
  
crea un’entrata “service\_templates:”  
per ogni Nodo con lo stereotipo Hardware:  
 crea un service\_template all’interno di “service\_templates:” apposito  
 se contiene nodi ExecutionEnvironment con lo stereotipo Piattaforma:  
 crea una lista “topology\_template:”  
 per ogni nodo execEnv con lo stereotipo Piattaforma all’interno del service\_template:  
 crea un “node\_template” all’interno di “topology\_template:”  
 se l’execEnv in questione contiene Artefatti con lo stereotipo Software:  
 crea una lista “artifacts:”   
 per ogni nodo Artifact con lo stereotipo Software deployato sull’execEnv  
 crea un’entrata all’interno di “artifacts:”

Per testare questa trasformazione è stato generato un Deployment Diagram denominato **SimpleDeploymentTOSCATest**, che andremmo ad analizzare nel dettaglio dopo aver commentato il codice Acceleo della trasformazione.

**CODICE ACCELEO:**

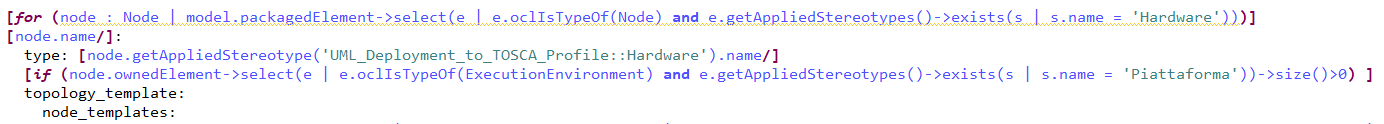
**A screenshot of a computer

AI-generated content may be incorrect.**

Questa porzione di codice prende in input un modello e apre un file di output **‘tosca\_output’** nel formato **.yaml**. L’ultima riga sarà la 1a effettiva riga nel nostro file di output.  
**NOTA: CHAT GPT INVERTE LA RIGA “TEMPLATE” E LA RIGA “MAIN”, COSI’ FACENDO, LA TRASFORMAZIONE NON RIUSCIRA’ MAI A TROVARE IL MODELLO DI INPUT,**



Se all’interno del modello esistono elementi di tipo **Node** con lo stereotipo **Hardware**, allora crea la lista **“service\_templates:”**

****

Per ogni **Node** con lo stereotipo **Hardware**, crea un entrata **“nome\_nodo:”** all’interno della lista **“service\_templates:”**, come **“type:”**, metti il suo **Stereotipo** tra quelli assegnatogli dal **Custom Profile** (dippiù su questo nella sezione che riguarderà il diagramma UML di prova).  
Se il **Node** contiene elementi di tipo **ExecutionEnvironment** con lo stereotipo **Piattaforma**, allora crea un’entrata **“topology\_template:”** e una lista **“node\_templates:”** al suo interno.

****

Se arriviamo a questo **For**, allora sappiamo che all’interno del **Node** esistono elementi di tipo **ExecutionEnvironment** annotati con lo stereotipo **Piattaforma**, allora, per ognuno di essi, creiamo un’entrata **“nome\_execution\_environment:”**, all’interno della quale inseriremo da cosa deriva il nodo (*scritto in automatico da Chat GPT, si può anche rimuovere o modificare come riga*) ed il suo **“type:”** (stereotipo custom) come prima.

A close up of a computer screen

AI-generated content may be incorrect.

Se all’interno dell’**ExecutionEnvironment** sono **Deployati** degli elementi **Artifact** con lo stereotipo **Software**, allora creiamo una lista **“artifacts:”**, e per ognuno di essi, creiamo un’entrata **“nome\_artifact:”** contenente il suo **“type:”** (stereotipo custom) come prima e il suo file **(NOTA:** quest’entrata sarà da sostituire con l’URI dell’Artifact che daremo in input al Modello, come avevamo stabilito dal paragrafo Miscellanea del documento [1. UML Profile](https://docs.google.com/document/d/1B_NxLZ6_KIiG_CqPyHFUOPlAbC5UfLadaLwA-CAKb2c/edit?tab=t.0) sul [Drive](https://drive.google.com/drive/folders/1EyU0rzzRajhknTcrU7c0pRsNBEFOwAjg)**)**  
**OSS:** probabilmente questa porzione di codice si può compattare ulteriormente **accorpando il** **1o ciclo For e la 2a If** in una sola riga.

A screenshot of a computer program

AI-generated content may be incorrect. Chiusura delle varie If, dei vari For, del File di output e del Template

**PER ESEGUIRE LA TRASFORMAZIONE:**

Tasto Dx su “generate.mtl” -> “Run As:” -> “Run configurations…”

**CONFIGURAZIONE DI LANCIO DELLA TRASFORMAZIONE:**

A screenshot of a computer

AI-generated content may be incorrect.

**FILE MANIFEST.MF PER LE DIPENDENZE:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**DEPLOYMENT DIAGRAM DI PROVA (SimpleDeploymentTOSCATest):**

**Come applicare un Custom Profile ad un Diagramma:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Come applicare uno Stereotipo di un Custom Profile ad un elemento del Diagramma:**

**A screenshot of a computer

AI-generated content may be incorrect.**

Il **Deployment Diagram** di prova è formato da **4 Nodes** che faranno da **Service Templates**, i quali a loro volta potranno contenere diversi nodi **ExecutionEnvironment** sui quali saranno deployati degli **Artifacts**. Andiamo a vedere nel dettaglio ogni suo **Service Template**:

A diagram of a computer

AI-generated content may be incorrect.

Ci aspettiamo che la trasformazione crei un **“service template”** di nome **“Hardware\_1”**, il quale contenga il **“node\_template”** di nome **“Piattaforma\_A”** e che, all’interno di quest’ultima, siano presenti i due **Artifacts** **“Software\_1”** e **“Software\_2”**, gli stereotipi sono presenti e corretti, e quindi, la trasformazione non avrà alcun problema.

A diagram of a software application

AI-generated content may be incorrect.

Più complessa. Ci aspettiamo che la trasformazione crei un **“service template”** di nome **“Hardware\_2”**, il quale conterrà due nodi **“node\_template”** di nome **“Piattaforma\_B1”** e **“Piattaforma\_B2”**, in **“Piattaforma\_B1”** sono deployati **2 Artifacts**, di cui solo “**Software\_3”** ha lo stereotipo **Software**, ci aspettiamo quindi che la trasformazione tenga conto solo di questo ed **escluda “Software\_4”**. Nessuno problema invece per **“Piattaforma\_B2”** e gli **Artifacts “Software\_5”** e **“Software\_6”** deployati su di essa.

A screenshot of a computer program

AI-generated content may be incorrect.

Altri due **Service Templates**, per quanto riguarda **“Hardware\_3”**, esso contiene un elemento **ExecutionEnvironment** sul quale è deployato un **Artifact**, l’**Artifact** è correttamente stereotipato, tuttavia, poiché l’**ExecutionEnvironment** non possiede lo stereotipo **Piattaforma**, esso verrà escluso dalla trasformazione, e con esso il suo **Artifact**.  
Per quanto riguarda **“Hardware\_4”**, suo **ExecutionEnvironment** è stereotipato correttamente, ma il **Service Template** non contiene lo stereotipo **“Hardware”**, verrà quindi escluso del tutto dalla trasformazione, e con esso il suo **ExecutionEnvironment**

Il risultato della trasformazione sarà osservabile all’interno del file **“tosca\_output.yaml”** presente nella directory **“output”**

A screenshot of a computer program

AI-generated content may be incorrect.

**TODO:**

1. Inserire nel file TOSCA Capabilities & Requirements traNodes e ExecutionEnvironmentscontenuti all’interno di essi.
2. Testare la possibilità di inserire collegamenti tra i vari Nodes ed integrare all’interno del File TOSCA questi collegamenti
3. A basso livello, inserire e testare Attributi e Proprietà TOSCA dei vari Elementi, e possibili generalizzazioni
4. Come detto precedentemente, integrare la possibilità di inserire le URI dei vari Artifact come Input all’interno del Modello
5. Come da sketch della trasformazione, creare il Service Template globale
6. Altro che sicuramente sto dimenticando