UniCTest

Ingegneria del Software – A.A. 2021/2022

Progetto d’esame di: Blanco Francesco Giulio, Anzalone Dario Giuseppe

Documentazione completa

Sommario

[1 Versione 4](#_Toc95166204)

[2 Ideazione 4](#_Toc95166205)

[2.1 Elaborati 4](#_Toc95166206)

[2.2 Documento di visione 4](#_Toc95166207)

[2.2.1 Introduzione 4](#_Toc95166208)

[2.2.2 Posizionamento 5](#_Toc95166209)

[2.2.3 Descrizione delle parti interessate 5](#_Toc95166210)

[2.2.4 Riepilogo delle caratteristiche del sistema 5](#_Toc95166211)

[2.3 Casi d’Uso 6](#_Toc95166212)

[2.3.1 Requisiti 6](#_Toc95166213)

[2.3.2 Obiettivi e Casi d’Uso 6](#_Toc95166214)

[2.3.3 Casi d’Uso 8](#_Toc95166215)

[2.4 Regole di dominio 23](#_Toc95166216)

[2.5 Specifiche supplementari 25](#_Toc95166217)

[2.5.1 Introduzione 25](#_Toc95166218)

[2.5.2 Interfacce 25](#_Toc95166219)

[2.5.3 Gestione ripristino simulazione corrente 25](#_Toc95166220)

[2.6 Glossario 25](#_Toc95166221)

[3 Elaborazione 28](#_Toc95166222)

[3.1 Iterazione I 28](#_Toc95166223)

[3.1.1 Introduzione e Elaborati 28](#_Toc95166224)

[3.1.2 Analisi Orientata agli Oggetti 28](#_Toc95166225)

[CO1 30](#_Toc95166226)

[CO2 30](#_Toc95166227)

[CO3 31](#_Toc95166228)

[CO4 31](#_Toc95166229)

[CO5 31](#_Toc95166230)

[CO6 31](#_Toc95166231)

[3.1.3 Progettazione Orientata agli Oggetti 32](#_Toc95166232)

[3.1.4 Implementazione e Testing 34](#_Toc95166233)

[3.2 Iterazione I – Refactoring 35](#_Toc95166234)

[3.2.1 Introduzione 35](#_Toc95166235)

[3.2.2 Analisi Orientata agli Oggetti 36](#_Toc95166236)

[CO2 37](#_Toc95166237)

[3.2.3 Progettazione Orientata agli Oggetti 37](#_Toc95166238)

[3.3 Iterazione II 42](#_Toc95166239)

[3.3.1 Introduzione e Elaborati 42](#_Toc95166240)

[3.3.2 Analisi Orientata agli Oggetti 42](#_Toc95166241)

[CO1 45](#_Toc95166242)

[CO2 45](#_Toc95166243)

[CO3 45](#_Toc95166244)

[CO4 45](#_Toc95166245)

[CO5 46](#_Toc95166246)

[CO6 46](#_Toc95166247)

[3.3.3 Revisione Analisi e Inizio Progettazione Orientata agli Oggetti 46](#_Toc95166248)

[CO2 bis 46](#_Toc95166249)

[3.3.4 Implementazione e Testing 50](#_Toc95166250)

[3.4 Iterazione III 51](#_Toc95166251)

[3.4.1 Introduzione e Elaborati 51](#_Toc95166252)

[3.4.2 Analisi Orientata agli Oggetti 51](#_Toc95166253)

[CO1 54](#_Toc95166254)

[CO2 54](#_Toc95166255)

[CO3 55](#_Toc95166256)

[CO4 55](#_Toc95166257)

[3.4.3 Progettazione Orientata agli Oggetti 55](#_Toc95166258)

[3.4.4 Implementazione e Testing 60](#_Toc95166259)

[3.5 Iterazione IV 61](#_Toc95166260)

[3.5.1 Introduzione e Elaborati 61](#_Toc95166261)

[3.5.2 Analisi Orientata agli Oggetti 61](#_Toc95166262)

[CO1 65](#_Toc95166263)

[CO2 66](#_Toc95166264)

[CO3 66](#_Toc95166265)

[CO4 66](#_Toc95166266)

[3.5.3 Progettazione Orientata agli Oggetti 66](#_Toc95166267)

[3.6 Iterazione V 75](#_Toc95166268)

[3.6.1 Introduzione 75](#_Toc95166269)

[3.6.2 Modello di Analisi 75](#_Toc95166270)

[3.6.3 Analisi e Progettazione UC9 79](#_Toc95166271)

[CO1 79](#_Toc95166272)

[CO2 80](#_Toc95166273)

[CO3 80](#_Toc95166274)

[CO4 80](#_Toc95166275)

[CO5 80](#_Toc95166276)

[3.6.4 Analisi e Progettazione UC10 84](#_Toc95166277)

[CO1 84](#_Toc95166278)

[CO2 85](#_Toc95166279)

[CO3 85](#_Toc95166280)

[COE1 86](#_Toc95166281)

[3.6.5 Implementazione e Testing 91](#_Toc95166282)

[3.7 Iterazione VI 93](#_Toc95166283)

[3.7.1 Introduzione 93](#_Toc95166284)

[3.7.2 Analisi orientata agli oggetti 94](#_Toc95166285)

[3.7.3 Progettazione Orientata agli Oggetti 95](#_Toc95166286)

[3.8 DCD finale 97](#_Toc95166287)

[3.8.1 Implementazione e Testing 98](#_Toc95166288)

# Versione

|  |  |  |  |
| --- | --- | --- | --- |
| Versione | Data | Descrizione | Autori |
| Documentazione completa | 2021/02/07 | Documentazione completa: fase di Ideazione, Elaborazione, sviluppi futuri. | Anzalone Dario Giuseppe, Blanco Francesco Giulio. |

# Ideazione

## Elaborati

Per la fase di Ideazione si è scelto di realizzare i seguenti documenti:

* **Documento di visione** (Paragrafo 2.2);
* **Modello dei Casi d’Uso** (Paragrafo 2.3);
* **Regole di dominio** (Paragrafo 2.4);
* **Specifiche supplementari** (Paragrafo 2.5);
* **Glossario** (Paragrafo 2.6).

## Documento di visione

### Introduzione

Prevediamo la realizzazione di un simulatore di test di ammissione all’università come supporto alla preparazione di studenti di un’organizzazione.

Un test di ammissione prevede quesiti che coprono varie discipline (matematica, fisica, chimica, biologia, logica, cultura generale, ecc.). Tali quesiti sono proposti da vari tutor (docenti), ciascuno afferente ad una o più discipline oggetto del test. I quesiti possono essere direttamente elaborati dai tutor o provenire da altre fonti.

I quesiti saranno a risposta multipla. Un quesito è costituito dal testo della domanda, un numero arbitrario di risposte (eventualmente più di una esatta), la difficoltà, la fonte.

Lo studente potrà effettuare simulazioni personalizzate scegliendo le materie da trattare, la difficoltà del test. Potrà inoltre scegliere tra dei template standard di simulazione (come fac-simile di test di medicina, ingegneria, giurisprudenza, ecc.). Il punteggio attribuito ad un quesito varia in base alla tipologia di simulazione scelta.

I tutor potranno organizzare simulazioni collettive in formato cartaceo, da svolgere in aula.

Lo studente potrà misurare la propria preparazione visualizzando le proprie statistiche e i propri punteggi, potrà confrontarsi con gli altri grazie alla presenza di graduatorie prodotte dal software stesso come risultato dello svolgimento delle varie simulazioni collettive.

I tutor potranno visualizzare le statistiche delle simulazioni cartacee, che costituiranno un feedback utile per approfondire eventuali tematiche generalmente poco comprese a lezione.

Per riassumere, i fini dell’applicazione sono:

* fornire agli studenti uno strumento valido per esercitarsi su quesiti di test di ammissione, eventualmente customizzati dall’organizzazione, stimolare l’apprendimento e il confronto con gli altri studenti;
* fornire ai docenti o tutor di preparazione un tool che sia in grado di semplificare il lavoro di preparazione di test e di lezioni.

### Posizionamento

#### Opportunità di business

Il software UniCTest si pone i seguenti obiettivi:

1. sostituire l’approccio “classico” di scrittura di test (scrittura a mano o utilizzo di editor di testo – e.g. Word – per creare un documento e stamparlo);
2. sostituire gli attuali software di simulazione, dove non è prevista customizzazione dei quesiti e dei test da parte dell’organizzazione che li utilizza e dove gli studenti non possono visualizzare i propri miglioramenti, interagire/confrontarsi tra loro e con i docenti.

#### Formulazione del problema

L’assenza di software dedicati alla preparazione di test rappresenta una grossa mancanza per i docenti, che sono costretti a scrivere ogni quesito, indentare, mettere insieme i quesiti di più materie per costruire intere simulazioni; il tutto senza alcun controllo automatizzato degli errori, senza confronto con i colleghi, senza automatizzazione della correzione dei test.

Tutto ciò rallenta il lavoro del docente e il prodotto finito, spesso, non è esente da errori.

#### Formulazione della posizione del prodotto

Il software è rivolto a tutte le scuole, pubbliche o private, università ed enti di preparazione ai test universitari.

### Descrizione delle parti interessate

#### Obiettivi a livello dell’utente

Gli utenti necessitano di un sistema che soddisfi i seguenti obiettivi:

* Amministratore: registrare utenti (Tutor e Studenti)
* Studente: effettuare simulazioni, visualizzare statistiche e graduatorie
* Tutor: gestire test e quesiti, registrare Studenti

### Riepilogo delle caratteristiche del sistema

Il sistema deve prevedere le seguenti funzionalità:

* Gestione delle politiche di business dell’organizzazione (e.g. registrazione di nuovi utenti – Tutor e Studenti)
* Inserimento di nuovi quesiti, materie
* Gestione delle simulazioni cartacee (simulazioni collettive in aula) per i Tutor
  + Composizione di test per simulazioni cartacee, con *shuffle* di domande/risposte
  + Correzione automatizzata di simulazioni cartacee
  + Creazione statistiche e graduatorie degli studenti
* Gestione delle simulazioni elettroniche (simulazioni individuali) per gli Studenti
  + Customizzazione di simulazioni
  + Esecuzione di simulazioni basate su template ufficiali o personalizzati
  + Creazione statistiche degli studenti

## Casi d’Uso

### Requisiti

UniCTest è un sistema software per la preparazione ai test di studenti di un’organizzazione. Il sistema deve permettere di realizzare ed effettuare le simulazioni degli studenti.

Gli aspetti di cui UniCTest deve occuparsi sono i seguenti:

* Inserimento di nuovi quesiti nel sistema.
* Inserimento di nuovi template di test personalizzati dallo studente.
* Esecuzione e correzione immediata di simulazioni singole in formato elettronico (da parte degli studenti).
  + Il punteggio attribuito allo studente non è statico ma dipende dal template scelto.
* Realizzazione e stampa (PDF) di test (e dei moduli risposte ad essi relativi) in formato cartaceo (da parte dei tutor), per simulazioni collettive in aula.
* Correzione automatizzata (da parte dei tutor) delle simulazioni cartacee effettuate in aula.
  + Il punteggio attribuito allo studente non è statico ma dipende dal template scelto dal tutor quando ha realizzato il test.
* Gestione delle statistiche e delle graduatorie degli studenti inerenti all’andamento dei test effettuati.
  + Possibilità di stampa delle statistiche e delle graduatorie (PDF).
* Possibilità da parte dell’amministratore di iscrivere studenti e tutor alla piattaforma.
* Possibilità da parte dei tutor di iscrivere studenti alla piattaforma.

### Obiettivi e Casi d’Uso

|  |  |  |
| --- | --- | --- |
| Attore | Obiettivo | Caso d’Uso |
| Studente | Gestire una simulazione elettronica (individuale). | UC1: Avvia simulazione |
| Studente | Gestire la creazione di un nuovo template per personalizzare le proprie simulazioni. | UC2: Crea template di test personalizzato |
| Tutor | Gestire la creazione di un nuovo template per personalizzare i test cartacei. | UC2/T: Crea template di test personalizzato |
| Amministratore | Gestire l’inserimento di un nuovo template ufficiale. | UC2/A: Crea template di test ufficiale |
| Studente | Gestire la modifica di un template già esistente. | UC3: Modifica template di test |
| Tutor | Gestire la modifica di un template già esistente. | UC3/T: Modifica template di test |
| Amministratore | Gestire la modifica di un template ufficiale già esistente. | UC3/A: Modifica template di test |
| Studente | Visualizzare le statistiche inerenti ai punteggi ottenuti nelle simulazioni passate in funzione del tempo. | UC4: Elabora statistiche personali |
| Amministratore | Gestire la registrazione di un nuovo Studente nel Sistema. | UC5: Nuovo Studente (CRUD) |
| Amministratore | Gestire la registrazione di un nuovo Tutor nel Sistema. | UC6: Nuovo tutor (CRUD) |
| Tutor | Gestire l’inserimento di un nuovo quesito nel Sistema. | UC7: Inserisci quesito |
| Tutor | Gestire la modifica di un quesito già presente nel Sistema. | UC8: Modifica quesito esistente |
| Tutor | Gestire la creazione di un test per effettuare una simulazione cartacea (collettiva). | UC9: Componi test per simulazione cartacea |
| Tutor | Gestire la correzione automatizzata di una simulazione cartacea effettuata da un gruppo di Studenti. | UC10: Correggi simulazione cartacea |
| Tutor | Visualizzare le statistiche inerenti a una simulazione cartacea. | UC11: Elabora statistiche simulazione cartacea |
| Studente o Tutor | Visualizzare la classifica degli Studenti che hanno effettuato una simulazione cartacea (collettiva). | UC12: Visualizza graduatoria (CRUD) |
| Tutor | Aggiungere una materia insegnata | UC13: Aggiungi materia insegnata |
| Amministratore | Assegnare un ruolo addizionale ad un Tutor (rendere un Tutor Amministratore e/o TutorSimulazione) | UC14: Aggiungi responsabilità  (Aggiungi Amministratore/Aggiungi TutorSimulazione) |
| Amministratore | Rimuovere un ruolo addizionale ad un Tutor (rimuovere il ruolo di Amministratore e/o di TutorSimulazione ad un Tutor) | UC15: Rimuovi responsabilità  (Rimuovi Amministratore/Rimuovi TutorSimulazione) |
| Tutor | Visualizzare tutti i quesiti inseriti nel Sistema da parte di un Tutor | UC16: Visualizza quesiti creati (CRUD) |
| Studente | Visualizzare tutti i test svolti e corretti, insieme al loro esito, da parte di uno Studente | UC17: Visualizza test svolti (CRUD) |

### Casi d’Uso

Nel corso delle varie revisioni della fase di Ideazione, sono state fornite delle descrizioni in formato dettagliato di tutti i Casi d’Uso.

Per pochi restanti Casi d’Uso (di importanza marginale per il progetto oppure alcuni Casi d’Uso CRUD) si fornisce una descrizione in formato breve.

**UC1. Avvia simulazione**

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC1: Avvia simulazione |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Studente |
| Parti interessate e interessi | Studente: vuole effettuare una nuova simulazione in formato elettronico. |
| Pre-condizioni | Lo Studente deve essere autenticato nel Sistema. |
| Post-condizioni (garanzia di successo) | Lo Studente conclude la simulazione, visualizza il risultato a video e il test con le dovute correzioni viene salvato nel Sistema. |
| Scenario principale di successo | 1. Lo Studente vuole iniziare una nuova simulazione.  2. Lo Studente sceglie l’attività “Avvia simulazione”.  3. Lo Studente sceglie il template su cui basare il test.  4. Il Sistema crea un test contenente quesiti sulla base del template scelto al punto 3 e lo restituisce allo Studente.  5. Lo Studente risponde ad una domanda del test.  *Il passo 5 viene ripetuto finché lo Studente non indica di voler consegnare.*  6. Lo Studente seleziona la voce “consegna il test”.  7. Il Sistema restituisce allo Studente un riepilogo della simulazione appena svolta, con tutte le risposte date.  8. Lo Studente seleziona la voce “conferma”.  9. Il Sistema corregge il test e calcola il punteggio complessivo. Il Sistema registra l’esito. Il Sistema restituisce il test corretto allo Studente. |
| Estensioni | **1-3a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Lo Studente riavvia il software. 2. Lo Studente può riavviare il Caso d’Uso dall’inizio.   **4-8a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Lo Studente riavvia il software. 2. Il Sistema ripristina la simulazione non ancora consegnata ripartendo dal punto 5. Tutte le risposte date prima dell’arresto sono ripristinate.   **3-8b.** In un qualsiasi momento lo Studente decide di abbandonare la simulazione. Lo Studente sceglie la voce “abbandona simulazione”.  **4c.** Non vengono trovati quesiti nel Sistema che rispettano le regole del template scelto. Dunque, il Caso d’Uso fallisce.  **8c.** Lo Studente decide di cambiare una o più risposte precedenti.   1. Lo Studente seleziona la voce “modifica risposte date”. 2. Il Sistema ritorna al passo 5. Tutte le risposte date prima sono ripristinate. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly.  La schermata di simulazione deve possedere le seguenti qualità:   * Il Sistema deve mostrare a schermo un quesito alla volta e presentare delle frecce di navigabilità che consentano allo Studente di passare al quesito successivo o precedente. * I Sistema deve mostrare a video una lista contenente i numeri progressivi dei quesiti contrassegnando quelli a cui lo Studente ha già risposto. Lo Studente può cliccare su uno di essi per andare direttamente a quel quesito. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Lo Studente può effettuare molte simulazioni ripetute. La frequenza può essere variabile. |
| Varie |  |

**UC2. Crea template di test personalizzato**

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC2: Crea template di test personalizzato |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Studente |
| Parti interessate e interessi | Studente: vuole effettuare simulazioni personalizzate, quindi decide di creare un nuovo template. |
| Pre-condizioni | Lo Studente è autenticato nel Sistema. |
| Post-condizioni (garanzia di successo) | Lo Studente ha creato con successo il template, che è stato salvato nel Sistema e può essere utilizzato per le simulazioni. |
| Scenario principale di successo | 1. Lo Studente vuole creare un nuovo template.  2. Lo Studente sceglie l’attività “Crea template personalizzato” e inserisce il nome del template che vuole creare. Il Sistema registra le informazioni inserite.  3. Lo Studente inserisce:   * il formato dei quesiti (cioè il numero di risposte e il numero di risposte corrette per quesito) * il punteggio attribuito ad un singolo quesito (nel caso di risposta corretta, risposta errata e risposta non data) * il tempo medio di risoluzione di un singolo quesito.   Il Sistema registra le informazioni inserite.  4. Il Sistema restituisce l’elenco delle materie disponibili.  5. Lo Studente seleziona una materia che vuole inserire nel template.  6. Lo Studente seleziona il numero di quesiti previsti per la materia scelta e la difficoltà media. Il Sistema registra le informazioni inserite.  *I passi 4, 5 e 6 vengono ripetuti finché serve.*  7. Lo Studente indica di aver finito. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Lo Studente riavvia il software. 2. Il Sistema ripristina lo stato. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa. In genere, lo Studente esegue UC1 con una frequenza molto maggiore di UC2. |
| Varie |  |

**UC2/T. Crea template di test personalizzato**

**Nota del cliente:** il Tutor che ha le autorizzazioni per creare un template di test personalizzato è un Tutor di simulazione. Il Tutor di simulazione ha diritti maggiori rispetto ad un Tutor classico, infatti, il test generato potrà contenere quesiti di diverse materie, non necessariamente insegnate dal Tutor che crea il test. I diritti di Tutor di simulazione potranno essere assegnati o rimossi a qualunque Tutor in qualunque momento.

Per brevità, nel Caso d’Uso, ci si riferirà al Tutor di simulazione semplicemente come Tutor.

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC2/T: Crea template di test personalizzato |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Tutor |
| Parti interessate e interessi | Tutor: vuole creare test cartacei personalizzati per gli Studenti, quindi decide di creare un nuovo template di test cartaceo personalizzato. |
| Pre-condizioni | Il Tutor è autenticato nel Sistema. |
| Post-condizioni (garanzia di successo) | Il Tutor ha creato con successo il template, che è stato salvato nel Sistema e può essere utilizzato per creare nuovi test cartacei. |
| Scenario principale di successo | 1. Il Tutor vuole creare un nuovo template.  2. Il Tutor sceglie l’attività “Crea template personalizzato per test cartaceo” e inserisce il nome del template che vuole creare. Il Sistema registra le informazioni inserite.  3. Il Tutor inserisce:   * il formato dei quesiti (cioè il numero di risposte e il numero di risposte corrette per quesito) * il punteggio attribuito ad un singolo quesito (nel caso di risposta corretta, risposta errata e risposta non data) * il tempo medio di risoluzione di un singolo quesito.   Il Sistema registra le informazioni inserite.  4. Il Sistema restituisce l’elenco delle materie disponibili.  5. Il Tutor seleziona una materia che vuole inserire nel template.  6. Il Tutor seleziona il numero di quesiti previsti per la materia scelta. Il Sistema registra le informazioni inserite.  *I passi 4, 5 e 6 vengono ripetuti finché serve.*  7. Il Tutor indica di aver finito. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Medio/alta. Un Tutor può organizzare simulazioni di vario tipo con una frequenza anche elevata, cambiando il template personalizzato anche ogni volta. |
| Varie |  |

Da notare la somiglianza tra UC2 e UC2/T. Come differenza, a parte gli attori coinvolti, bisogna notare che in UC2/T il Tutor, potendo creare un test cartaceo *ex-novo*, non deve inserire la difficoltà media dei quesiti, mentre in UC2 lo studente inseriva la difficoltà in quanto la creazione del test per la simulazione online era automatizzata.

Quindi il Tutor potrà creare solo ciò che concerne le simulazioni cartacee (e dunque collettive), mentre lo Studente potrà creare solo ciò che concerne le simulazioni online (e dunque personali)[[1]](#footnote-1).

**UC2/A. Crea template di test ufficiale**

**Nota del cliente:** l’Amministratore è un Tutor con diritti maggiori rispetto ad un Tutor classico. L’Amministratore, oltre a insegnare materie (dunque inserire nuovi quesiti), può anche inserire nuovi Tutor nel Sistema, oltre che inserire Studenti e template ufficiali. I diritti di Amministratore potranno essere assegnati o rimossi a qualunque Tutor in qualunque momento.

Nel Caso d’Uso e in generale ci si riferirà al Tutor con i diritti di Amministratore semplicemente come Amministratore.

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC2/A: Crea template di test personalizzato |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Amministratore |
| Parti interessate e interessi | Amministratore: vuole creare simulazioni personalizzate per gli Studenti, quindi decide di creare un nuovo template. |
| Pre-condizioni | L’Amministratore è autenticato nel Sistema. |
| Post-condizioni (garanzia di successo) | L’Amministratore ha creato con successo il template ufficiale, che è stato salvato nel Sistema e può essere utilizzato per le simulazioni. |
| Scenario principale di successo | 1. L’Amministratore vuole creare un nuovo template.  2. L’Amministratore sceglie l’attività “Crea template ufficiale” e inserisce il nome del template ufficiale che deve inserire nel Sistema. Il Sistema registra le informazioni inserite.  3. L’Amministratore inserisce:   * la fonte che ha stabilito le regole del test su cui il template ufficiale deve basarsi * il formato dei quesiti (cioè il numero di risposte e il numero di risposte corrette per quesito) * il punteggio attribuito ad un singolo quesito (nel caso di risposta corretta, risposta errata e risposta non data) * il tempo totale previsto dal test ufficiale.   Il Sistema registra le informazioni inserite.  4. L’Amministratore inserisce il nome della materia che deve essere inserita nel Test.  5. L’Amministratore seleziona il numero di quesiti previsti per la materia selezionata. Il Sistema registra le informazioni inserite.  *I passi 4, 5 e 6 vengono ripetuti finché serve.*  6. L’Amministratore indica di aver finito. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. L’Amministratore riavvia il software. 2. Il Sistema ripristina lo stato. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa. In genere, esecuzione su base semestrale o trimestrale. |
| Varie |  |

**UC3. Modifica template di test**

Uno Studente vuole modificare un template di test esistente (ad es. ho salvato il template di medicina tramite esecuzione del Caso d’Uso UC2 e voglio rimuovere una materia oppure abbassarne la difficoltà).

**UC3/T. Modifica template di test**

Un Tutor vuole modificare un template di test esistente. Il Tutor apporta le modifiche e il template aggiornato viene salvato nel Sistema.

**UC3/A. Modifica template di test**

Un Amministratore deve modificare un template di test ufficiale. L’Amministratore apporta le modifiche e il template aggiornato viene salvato nel Sistema.

**UC4. Elabora statistiche personali**

Lo Studente registrato vuole visualizzare le proprie statistiche sulle simulazioni da lui effettuate. Vengono visualizzati: punteggio di ogni simulazione, punteggio medio complessivo (percentuale) delle simulazioni passate.

**UC5. Nuovo Studente (CRUD)**

Uno Studente che fa parte dell’organizzazione vuole utilizzare UniCTest. Lo Studente si rivolge ad un Tutor o all’Amministratore per ottenere le credenziali di accesso al Sistema. Il Tutor/Amministratore registra lo Studente nel Sistema.

**UC6. Nuovo Tutor (CRUD)**

**Caso d’Uso in formato breve:**

Un Tutor che fa parte dell’organizzazione si rivolge all’Amministratore per ottenere le credenziali di accesso al Sistema. L’Amministratore registra il Tutor nel Sistema.

**Nota del cliente:** l’Amministratore è un Tutor con diritti maggiori rispetto ad un Tutor classico. L’Amministratore, oltre a insegnare materie (dunque inserire nuovi quesiti), può anche inserire nuovi Tutor nel Sistema, oltre che inserire Studenti e template ufficiali. I diritti di Amministratore potranno essere assegnati o rimossi a qualunque Tutor in qualunque momento.

Nel Caso d’Uso e in generale ci si riferirà al Tutor con i diritti di Amministratore semplicemente come Amministratore.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC6: Nuovo Tutor |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Amministratore |
| Parti interessate e interessi | Amministratore: vuole inserire correttamente un nuovo Tutor nel Sistema.  Tutor: vuole che i dati da lui forniti all’Amministratore siano inseriti correttamente nel Sistema. |
| Pre-condizioni | L’Amministratore deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha inserito correttamente un nuovo Tutor. |
| Scenario principale di successo | 1. L’Amministratore vuole inserire un nuovo Tutor nel Sistema.  2. Il Tutor sceglie la voce “registra nuovo Tutor”.  3. L’Amministratore inserisce nome, cognome e codice fiscale del Tutor.  4. Il Sistema registra le informazioni inserite. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato.   **3a.** È stato inserito un codice fiscale associato ad un Utente già registrato nel Sistema. Il Sistema comunica l’errore e ripete il passo 3. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa. |
| Varie |  |

**UC7. Inserisci quesito**

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC7: Inserisci quesito |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Tutor |
| Parti interessate e interessi | Tutor: vuole inserire un nuovo quesito da utilizzare in una simulazione cartacea, in simulazioni elettroniche oppure tenerlo per sé.  Studenti: vogliono poter usufruire del nuovo quesito inserito in simulazioni elettroniche o cartacee. |
| Pre-condizioni | Il Tutor deve essere autenticato nel Sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha registrato correttamente il quesito nella sezione corretta (quesiti pubblici, privati o personali). |
| Scenario principale di successo | 1. Il Tutor vuole inserire un nuovo quesito  2. Il Tutor sceglie la voce “nuovo quesito”.  3. Il Sistema restituisce le materie che il Tutor insegna. Il Tutor sceglie la materia del quesito da inserire tra quelle restituite dal Sistema.  4. Il Tutor inserisce la fonte da cui il quesito proviene.  5. Il Tutor inserisce il testo della domanda.  6. Il Tutor inserisce il testo di una risposta. Il Tutor contrassegna la risposta come vera o falsa.  *Il passo 6 viene ripetuto finché serve.*  7. Il Tutor indica di aver terminato l’inserimento delle risposte.  8. Il Tutor inserisce la difficoltà del quesito.  9. Il Tutor contrassegna il quesito come pubblico, privato o personale.  10. Il Tutor conferma l’inserimento. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato.   **3a.** Il Tutor non insegna alcuna materia. In tal caso, il Sistema segnala l’errore al Tutor ed interrompe l’esecuzione del Caso d’Uso.  **8a.** Il Tutor non ha inserito alcuna risposta corretta. Il Sistema segnala l’errore al Tutor e ripristina lo stato al punto 6, conservando le risposte già inserite. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Il Tutor può inserire molti quesiti di seguito. Tuttavia, la frequenza può essere variabile. |
| Varie |  |

**UC8. Modifica quesito esistente**

Un Tutor afferente ad una materia apporta delle modifiche ad un quesito inerente a quella materia.

**UC9. Componi test per simulazione cartacea**

**Caso d’Uso in formato breve:**

Un Tutor vuole creare un test personalizzato per i propri Studenti, scegliendo uno ad uno i quesiti da inserire. Il test può essere memorizzato nel Sistema e stampato (PDF). La stampa del test prevede anche la generazione del modulo delle risposte (PDF).

**Nota del cliente:** il Tutor che ha le autorizzazioni per creare una simulazione cartacea è un Tutor di simulazione. Infatti, il test generato potrà contenere quesiti di diverse materie, non necessariamente insegnate dal tutor che crea il test. I diritti di Tutor di simulazione potranno essere assegnati o rimossi a qualunque Tutor in qualunque momento.

Per brevità, nel Caso d’Uso, ci si riferirà al Tutor di simulazione semplicemente come Tutor.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC9: Componi test per simulazione cartacea |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Tutor |
| Parti interessate e interessi | Tutor: vuole creare un nuovo test per una simulazione cartacea e salvarlo in PDF insieme al suo modulo risposte. |
| Pre-condizioni | Il Tutor deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha creato correttamente il Test contenente i Quesiti scelti dal Tutor ed ha generato il file PDF. |
| Scenario principale di successo | 1. Il Tutor vuole creare un nuovo test cartaceo  2. Il Tutor sceglie la voce “nuovo test cartaceo”.  3. Il Tutor sceglie il template su cui basare il test.  4. Il Sistema mostra al Tutor i quesiti disponibili per l’inserimento nel test relativi ad una materia del template.  5. Il Tutor seleziona i quesiti da inserire relativi alla materia indicata.  *I passi 4 e 5 sono ripetuti fin quando terminano le materie del template.*  6. Il Sistema registra le informazioni inserite e restituisce al Tutor il test e il modulo per le risposte in formato PDF. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato.   **4a.** Non vengono trovati quesiti nel Sistema relativi ad una certa materia che rispettano le regole del template scelto. Dunque, il Caso d’Uso fallisce. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Media. |
| Varie |  |

**UC10. Correggi simulazione cartacea**

**Caso d’Uso in formato breve:**

Un Tutor che vuole correggere le simulazioni cartacee effettuate scannerizza il modulo delle risposte di ciascuno Studente. Per ogni test, il Sistema legge il codice identificativo dello studente e le risposte date. Il Sistema verifica, per ogni risposta, se è corretta o no, calcola il punteggio complessivo della simulazione e registra questi dati attribuendoli al corretto Studente. Ad ogni Studente viene restituito il proprio test corretto in formato elettronico.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC10: Correggi simulazione cartacea |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Tutor |
| Parti interessate e interessi | Tutor: vuole che le simulazioni cartacee vengano corrette in maniera automatizzata e non a mano. Vuole che il tasso di errore sia basso. |
| Pre-condizioni | Il Tutor deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha corretto i test ed ha registrato i test corretti agli studenti. |
| Scenario principale di successo | 1. Il Tutor vuole correggere il test cartaceo di uno studente.  2. Il Tutor sceglie la voce “correggi test cartaceo”.  3. Il Sistema chiede al tutor di inserire il modulo risposte inerente al test da correggere.  4. Il Sistema legge le informazioni del foglio risposte: l’id del test a cui fa riferimento, il codice dello Studente al quale registrare l’esito, le risposte inserite dallo Studente. Il Sistema restituisce al Tutor il test contenente le informazioni lette.  5. Il Tutor conferma l’inserimento del test corretto.  6. Il Sistema salva il test corretto e registra l’esito allo Studente. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Il Tutor può inserire molti quesiti di seguito. Tuttavia, la frequenza può essere variabile. |
| Varie |  |

**UC11. Elabora statistiche simulazione cartacea**

Il Tutor vuole visualizzare le statistiche relative ad una simulazione effettuata in aula dagli Studenti dell’organizzazione. Sono visualizzati: punteggio medio complessivo degli studenti, percentuale di risposte date correttamente per singola materia e per singola domanda, percentuale di Studenti che hanno selezionato una specifica risposta ad una domanda.

**UC12. Visualizza graduatoria (CRUD)**

Lo Studente o il Tutor registrato vuole visualizzare la graduatoria di una simulazione cartacea effettuata dagli Studenti dell’organizzazione. Per ogni Studente sono visualizzati: punteggio, posizione in graduatoria.

**UC13. Aggiungi materia insegnata**

**Caso d’Uso in formato breve:**

Un Tutor è stato assunto per insegnare una materia non necessariamente già presente nel Sistema. L’Amministratore inserisce il codice fiscale del Tutor e gli associa la nuova materia che. Se la materia inserita non esiste nel Sistema, viene automaticamente creata e associata al Tutor.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC13. Aggiungi materia insegnata |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Amministratore |
| Parti interessate e interessi | Amministratore: vuole associare correttamente una materia ad un Tutor già registrato nel Sistema. Qualora la materia non fosse presente nel Sistema, non vuole crearla a mano ma vuole che venga creata automaticamente.  Tutor: vuole che i dati siano inseriti correttamente nel Sistema. |
| Pre-condizioni | L’Amministratore deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha associato correttamente la materia al Tutor. |
| Scenario principale di successo | 1. L’Amministratore vuole associare una materia ad un Tutor già presente nel Sistema.  2. Il Tutor sceglie la voce “Aggiungi materia insegnata”.  3. L’Amministratore inserisce il codice fiscale del Tutor e il nome della materia insegnata.  4. Il Sistema associa il Tutor alla materia inserita. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. L’Amministratore riavvia il software. 2. Il Sistema ripristina lo stato.   **3a.** È stato inserito un codice fiscale associato ad un Utente non ancora registrato nel Sistema. Il Sistema comunica l’errore. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa/media. |
| Varie |  |

**UC14. Aggiungi responsabilità**

**Caso d’Uso in formato breve:**

Un Tutor viene promosso ad Amministratore o a Tutor di simulazione, dunque si rivolge ad un Amministratore il quale inserisce il nuovo ruolo del Tutor nel Sistema, consentendo al Tutor di godere anche dei diritti del ruolo appena acquisito.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC14. Aggiungi responsabilità |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Amministratore |
| Parti interessate e interessi | Amministratore: vuole inserire il nuovo ruolo di un Tutor nel Sistema.  Tutor: vuole godere dei diritti del ruolo appena acquisito. |
| Pre-condizioni | L’Amministratore deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha associato correttamente il ruolo aggiuntivo al Tutor. |
| Scenario principale di successo | 1. L’Amministratore vuole inserire il nuovo ruolo di un Tutor già presente nel Sistema.  2. L’Amministratore sceglie la voce “Aggiungi responsabilità”.  3. L’Amministratore inserisce il codice fiscale del Tutor e digita il nuovo ruolo da assegnare al Tutor.  4. Il Sistema associa il nuovo ruolo al Tutor. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. L’Amministratore riavvia il software. 2. Il Sistema ripristina lo stato.   **3a.** È stato inserito un codice fiscale associato ad un Utente non ancora registrato nel Sistema. Il Sistema comunica l’errore.  **3b.** Il ruolo inserito è un ruolo già ricoperto dal Tutor. Il Sistema non modifica il suo stato. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa/media. |
| Varie |  |

**UC15. Rimuovi responsabilità**

**Caso d’Uso in formato breve:**

Ad un Amministratore/Tutor di simulazione vengono rimossi i diritti di Amministratore/Tutor di simulazione da parte di un Amministratore.

Per semplicità ci si riferirà all’Amministratore/Tutor di simulazione a cui deve essere rimosso il ruolo semplicemente come Tutor.

**Caso d’Uso in formato dettagliato:**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC14. Aggiungi responsabilità |
| Portata | Applicazione UniCTest |
| Livello | Obiettivo utente |
| Attore primario | Amministratore |
| Parti interessate e interessi | Amministratore: vuole rimuovere un ruolo di un Amministratore o Tutor di simulazione. L’Amministratore non vuole che questi goda dei diritti del ruolo appena rimosso. |
| Pre-condizioni | L’Amministratore deve essere autenticato nel sistema. |
| Post-condizioni (garanzia di successo) | Il Sistema ha rimosso correttamente il ruolo aggiuntivo al Tutor. |
| Scenario principale di successo | 1. L’Amministratore vuole inserire il nuovo ruolo di un Tutor già presente nel Sistema.  2. L’Amministratore sceglie la voce “Rimuovi responsabilità”.  3. L’Amministratore inserisce il codice fiscale del Tutor e digita il ruolo da rimuovere al Tutor.  4. Il Sistema rimuove il ruolo al Tutor. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. L’Amministratore riavvia il software. 2. Il Sistema ripristina lo stato.   **3a.** È stato inserito un codice fiscale associato ad un Utente non ancora registrato nel Sistema. Il Sistema comunica l’errore.  **3b.** Il ruolo inserito è un ruolo non ricoperto dal Tutor. Il Sistema non modifica il suo stato.  **3c.** L’Amministratore sta tentando di rimuovere a sé stesso il ruolo di Amministratore. Il Sistema comunica all’Amministratore che ciò non è possibile. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Bassa/media. |
| Varie |  |

**UC16. Visualizza quesiti creati (CRUD)**

**Caso d’Uso in formato breve:**

Un Tutor che fa parte dell’organizzazione vuole visualizzare tutti i quesiti da lui inseriti nel Sistema. Il Sistema restituisce tutti i quesiti da lui inseriti, compresi di risposte, della loro correttezza e della visibilità del quesito.

**UC17. Visualizza test svolti (CRUD)**

**Caso d’Uso in formato breve:**

Uno Studente che fa parte dell’organizzazione vuole visualizzare tutti i test svolti: sia i test cartacei (e dunque collettivi), sia i test elettronici (e dunque individuali). Il Sistema restituisce tutti i test svolti, indicando anche la correttezza o meno di ogni risposta data e l’esito complessivo.

## Regole di dominio

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Regola | Modificabilità | Sorgente |
| R1 – Template Medicina | La regola R1 illustra il template ufficiale del test di Medicina, Odontoiatria, Professioni Sanitarie, Veterinaria.  Il testè composto da **60 domande** così suddivise:   * 12 quesiti di cultura generale * 10 quesiti di ragionamento logico * 18 quesiti di biologia * 12 quesiti di chimica * 4 quesiti di fisica * 4 quesiti di matematica.   **Formato della domanda:**   * 5 risposte possibili di cui una sola esatta.   **Punteggi**:   * 1,5 punti per ogni risposta esatta * 0 punti per ogni risposta non data * -0.4 punti per ogni risposta errata.   **Tempo:**   * Il test ha un tempo limite di 90 minuti. | Bassa | MIUR |
| R2 – Template Ingegneria | La regola R2 illustra il template ufficiale del test di Ingegneria ed altri corsi in ambito tecnico e scientifico.  Un **test TOLC-I** è composto da **50 domande** così ripartite:   * 20 quesiti di matematica * 10 quesiti di ragionamento logico * 10 quesiti di scienze (biologia, chimica) * 10 di comprensione verbale (logica)   **Formato della domanda:**   * 5 risposte possibili di cui una sola esatta.   **Punteggi:**   * 1 punto per ogni risposta corretta * 0 punti per ogni risposta non data * -0,25 punti per ogni risposta errata.   **Tempo:**   * Il test ha un tempo limite di 110 minuti. | Bassa | CISIA |
| R3 – Template Giurisprudenza | La regola R3 illustra il template ufficiale del test di Giurisprudenza.  Il testè composto da **40 domande** così ripartite:   * 20 quesiti di lingua italiana * 15 quesiti di storia * 5 quesiti di logica   **Formato della domanda:**   * 5 risposte possibili di cui una sola esatta.   **Punteggi:**   * 1 punto per ogni risposta corretta * 0 punti per ogni risposta non data * -0,25 punti per ogni risposta errata.   **Tempo:**   * Il test ha un tempo limite di 90 minuti. | Media | Università degli Studi di Catania |
| R4 – Template TOLC (sezione di Inglese) | La sezione di Inglese di un test TOLC prevede la risoluzione di:   * 30 quesiti di inglese   **Formato della domanda:**   * 5 risposte possibili di cui una sola esatta.   **Punteggi:**   * 1 punto per ogni risposta corretta * 0 punti per ogni risposta non data o errata (non è prevista penalizzazione per le risposte errate).   **Tempo:**   * Il test ha un tempo limite di 15 minuti. | Media | CISIA |

**Nota del cliente:** i Template devono poter essere inseriti da un Amministratore del Sistema, il quale è un Tutor con diritti di amministrazione. I Template, dunque, non devono essere *hard-coded*.

## Specifiche supplementari

### Introduzione

Questo documento raccoglie i requisiti del sistema UniCTest non colti nei Casi d’Uso.

### Interfacce

Le interfacce Hardware e Software sono utili per la correzione delle simulazioni cartacee.

#### Hardware

* Scanner o lettore ottico per la scansione dei codici degli studenti (che devono essere in formato QR Code o alfanumerici) nel foglio risposte.

#### Software

* Lettore QR Code o OCR (Riconoscimento Ottico dei Caratteri).

### Gestione ripristino simulazione corrente

#### Vincoli sul salvataggio delle domande

Quando uno Studente effettua una simulazione elettronica, le risposte date devono essere salvate ciclicamente per permetterne il ripristino in caso di arresto improvviso.

#### Vincoli sul timing della simulazione

Ogni simulazione prevede un timer. In caso di arresti improvvisi, il timer deve essere ripristinato con una tolleranza massima di 5 minuti a favore dello Studente.

## Glossario

|  |  |
| --- | --- |
| Termine | Definizione |
| UniCTest | Rappresenta il Sistema software (SuD, System Under Discussion). |
| Materia | Disciplina oggetto del test. Il Sistema contiene quesiti appartenenti a varie materie. |
| Quesito | È una domanda a risposta multipla. Comprende: testo della domanda + risposte (tra cui risposte esatte e errate, con ) + difficoltà + fonte + materia.  Un quesito ha uno fra tre livelli di visibilità:   * Pubblico: accessibile a tutti, Tutor e Studenti. Un quesito pubblico è sorteggiabile nelle simulazioni elettroniche. * Privato: accessibile solo ai Tutor. Un quesito privato è utilizzabile nelle simulazioni cartacee (che sono elaborate dai Tutor). * Personale: accessibile solo al Tutor creatore. Un quesito personale non può essere sorteggiato. |
| Risposta | Una delle opzioni (alternative) ad un quesito.  Comprende: il testo della risposta e l’informazione sulla sua esattezza o meno. |
| Test | Aggregazione di quesiti divisi per materia. |
| Simulazione | Esecuzione reale di un test da parte di uno Studente.  Una simulazione può essere:   * Elettronica (detta anche al PC o individuale):   Esecuzione di un test in formato elettronico utilizzando UniCTest.  La simulazione al PC è eseguita da un singolo studente.   * Cartacea (detta anche collettiva):   Esecuzione di un test in formato cartaceo, consegnato manualmente allo Studente dal Tutor.  La simulazione cartacea è eseguita in aula contemporaneamente da un gruppo di Studenti. |
| Modulo risposte | Il modulo delle risposte di un test è un foglio di carta compilato da ogni Studente durante una simulazione cartacea su cui:   * Va applicata l’etichetta contenente il codice identificativo dello Studente * Vanno inserite le risposte date nel test   Il modulo delle risposte viene scannerizzato otticamente[[2]](#footnote-2) dal Tutor per correggere una simulazione cartacea. |
| Template | Struttura/modello di un test: comprende il formato dei quesiti, il tempo della simulazione, le materie da trattare, il numero di domande per materia, la difficoltà media di ciascuna materia. Un test è costruito aggregando quesiti secondo le regole di un template di test. |
| Graduatoria | Classifica di Studenti in base al punteggio medio di una simulazione cartacea (collettiva) precedentemente effettuata. |
| Statistiche | Parametri relativi alle simulazioni passate di un singolo Studente (punteggio medio, punteggio (in percentuale) relativo ad ogni simulazione in funzione del tempo, punteggio medio per materia, curva di miglioramento nel tempo, ecc.). |
| Organizzazione | Scuola o ente, pubblico o privato, che fornisce preparazione ai test mediante l’utilizzo del software UniCTest. |
| Studente | Studente dell’organizzazione che utilizza UniCTest per simulare dei test. |
| Tutor | Docente preparatore degli Studenti, inserisce i quesiti e/o i test nel Sistema. Un Tutor è afferente ad una o più materie oggetto del test.  Al Tutor è consentito inserire quesiti inerenti solamente alla sua materia.  Al Tutor è consentito registrare Studenti nel Sistema. |
| Amministratore | Operatore con diritti maggiori di Studenti e Tutor a cui è consentito registrare Studenti e Tutor nel Sistema. |

# Elaborazione

## Iterazione I

### Introduzione e Elaborati

Durante la fase di Ideazione sono stati individuati la maggior parte dei requisiti dell’applicazione UniCTest.

Nella 1° Iterazione della fase di Elaborazione, tra i vari Casi d’Uso individuati nella fase di Ideazione, si è scelto di focalizzare l’attenzione su:

* Scenario di successo di UC7: Inserisci quesito;
* Caso d’Uso d’avviamento.

Si è scelto di realizzare i seguenti artefatti:

* Analisi Orientata agli Oggetti
  + **Modello di Dominio**
  + **SSD (System Sequence Diagrams)**
  + **Contratti delle operazioni**
* Progettazione Orientata agli Oggetti:
  + **Diagrammi di interazione**
  + **DCD (Design Class Diagram)**

La 1° Iterazione della fase di Elaborazione prevederà diversi step:

1. Analisi orientata agli oggetti (OOA). Il primo artefatto da realizzare sarà il Modello di Dominio. Le classi concettuali saranno individuate tra le locuzioni nominali dello scenario di successo di UC7.
2. Progettazione orientata agli oggetti (OOD). Bisognerà realizzare il Modello di Progetto, comprendente diagrammi statici (Diagramma delle Classi di Progetto) e dinamici (Diagrammi di Interazione, ad esempio Diagrammi di Sequenza).
3. Implementazione e Testing. La fase di Implementazione prevederà la creazione delle prime classi e dei primi metodi del progetto. Dovranno essere sviluppati anche: UI, casi di test.

### Analisi Orientata agli Oggetti

#### Modello di dominio

Si riporta lo scenario di successo di UC7. Vengono evidenziate le locuzioni nominali per la scelta delle classi concettuali candidate. Le classi concettuali candidate verranno evidenziate in verde, gli attributi in giallo.

**UC7. Inserisci quesito**

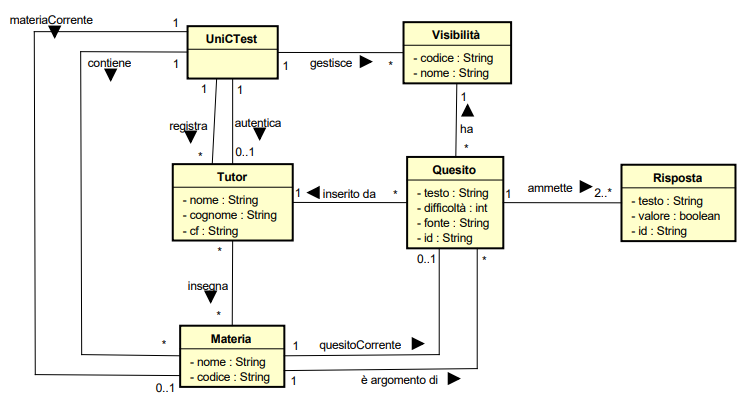
|  |  |
| --- | --- |
| Scenario principale di successo | 1. Il Tutor vuole inserire un nuovo quesito  2. Il Tutor sceglie la voce “nuovo quesito”.  3. Il Sistema restituisce le materie che il Tutor insegna. Il Tutor sceglie la materia del quesito da inserire tra quelle restituite dal Sistema.  4. Il Tutor inserisce la fonte da cui il quesito proviene.  5. Il Tutor inserisce il testo della domanda.  6. Il Tutor inserisce il testo di una risposta. Il Tutor contrassegna la risposta come vera o falsa.  *Il passo 6 viene ripetuto finché serve.*  7. Il Tutor indica di aver terminato l’inserimento delle risposte.  8. Il Tutor inserisce la difficoltà del quesito.  9. Il Tutor contrassegna il quesito come pubblico, privato o personale.  10. Il Tutor conferma l’inserimento. |

Le classi concettuali candidate sono:

* **UniCTest**: rappresenta il sistema;
* **Tutor**: utente del sistema che inserisce quesiti;
* **Quesito**: contiene una domanda e le informazioni ad essa inerenti;
* **Materia**: disciplina insegnata dai tutor e inerente ai quesiti;
* **Risposta**: possibile alternativa fra le risposte legate a un quesito, che può essere vera o falsa.

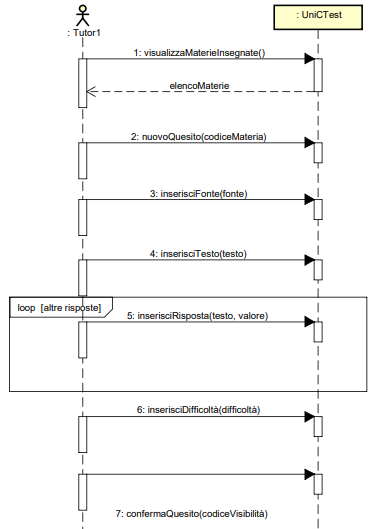
Dall’analisi effettuata si è resa necessaria l’introduzione di una classe concettuale per la visibilità del quesito (pubblico, privato o personale). Infatti, si vogliono avere più tipi di visibilità la cui esistenza deve essere indipendente dalla esistenza di quesiti nel sistema.  
Per cui, alla lista precedente viene aggiunta la classe concettuale:

* **Visibilità**: indica se il quesito a cui essa fa riferimento è pubblico, privato o personale.

Da cui, tenendo conto di associazioni e attributi, è stato ricavato il seguente Modello di Dominio:

#### Diagramma di Sequenza di Sistema

Il passo successivo alla creazione del Modello di Dominio è la creazione del primo SSD che permette la visualizzazione in forma grafica dell’interazione tra il Tutor ed il Sistema in riferimento a UC7.



#### Contratti delle Operazioni

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC7.

### CO1

|  |  |
| --- | --- |
| Operazione | visualizzaMaterie() |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | - |
| Post-condizioni | * Il Sistema ha restituito tutte le *n* istanze *m1,…,mn* di Materia associate all’istanza *t* di Tutor. |

### CO2

|  |  |
| --- | --- |
| Operazione | nuovoQuesito(codiceMateria: String) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | - |
| Post-condizioni | * L’istanza *mi* di Materia avente *mi.codice* uguale a codiceMateria è stata associata a UniCTest tramite l’associazione “corrente”; * È stata creata l’istanza *q* di Quesito ed è stato inizializzato l’attributo *q*.*id*; * *mi* è stata associata a *q* tramite l’associazione “corrente”. * *q* è stato associato all’istanza *t* di Tutor tramite l’associazione “inserito da”. |

### CO3

|  |  |
| --- | --- |
| Operazione | inserisciFonte(fonte: String) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | È in corso la creazione di un Quesito *q.* |
| Post-condizioni | * L’attributo *q*.*fonte* è stato inizializzato a fonte. |

### CO4

|  |  |
| --- | --- |
| Operazione | inserisciRisposta(testo: String, valore: boolean) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | È in corso la creazione di un Quesito *q.* |
| Post-condizioni | * È stata creata un’istanza *ri* di Risposta; * L’attributo *ri*.*testo* è stato inizializzato a testo e l’attributo *ri*.*valore* è stato inizializzato a valore; * *ri* è stata associata a *q* tramite l’associazione “ammette”. |

### CO5

|  |  |
| --- | --- |
| Operazione | inserisciDifficoltà(difficoltà: int) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | È in corso la creazione di un Quesito *q.* |
| Post-condizioni | * L’attributo *q*.*difficoltà* è stato inizializzato a difficoltà. |

### CO6

|  |  |
| --- | --- |
| Operazione | confermaQuesito(codiceVisibilità: String) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | È in corso la creazione di un Quesito *q.* |
| Post-condizioni | * È stata recuperata l’istanza *v* di Visibilità avente *v*.*codice* uguale a codiceVisibilità ed è stata associata a *q* tramite l’associazione “ha”; * *q* è stato inserito nella mappa “mappaQuesiti” di *m*. |

### Progettazione Orientata agli Oggetti

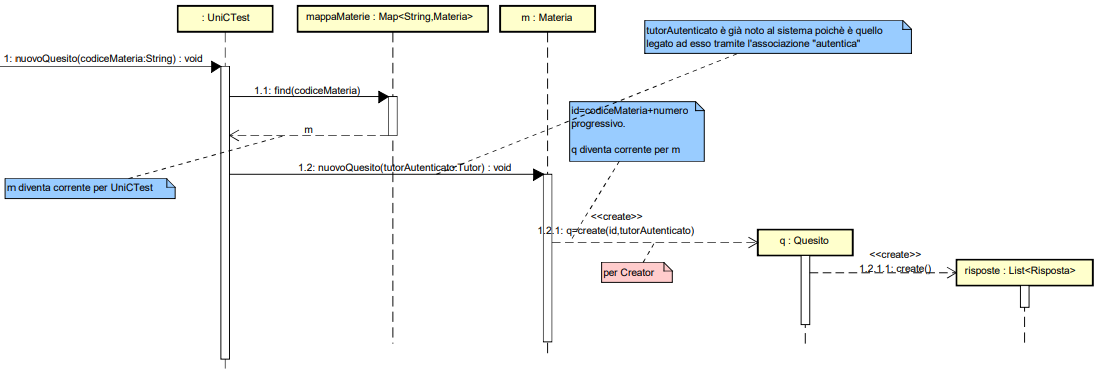
Dalla definizione delle classi concettuali, realizzata nel contesto della OOA, si passa alla OOD, che prevede la realizzazione di un Modello di Progetto, che include la definizione statica delle classi software mediante un DCD, da realizzare in parallelo ai diagrammi di interazione, che definiscono il comportamento dinamico del sistema.

#### Diagrammi di interazione

1. **SSD\_UC7\_visualizzaMateria**

Immagine che contiene testo

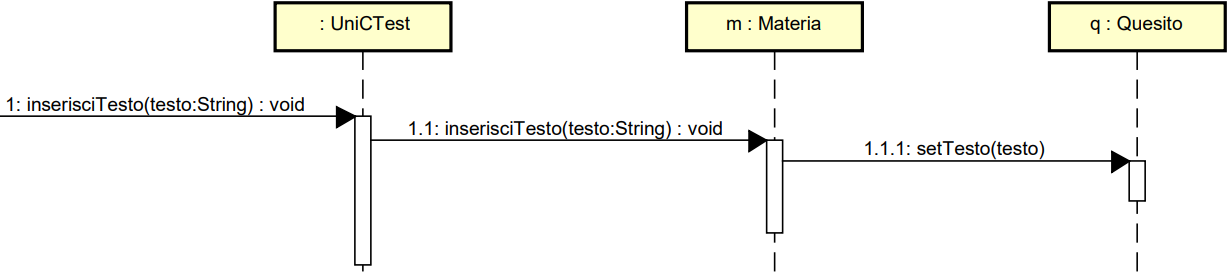
Descrizione generata automaticamente

1. **SSD\_UC7\_nuovoQuesito**
2. **SSD\_UC7\_inserisciFonte**

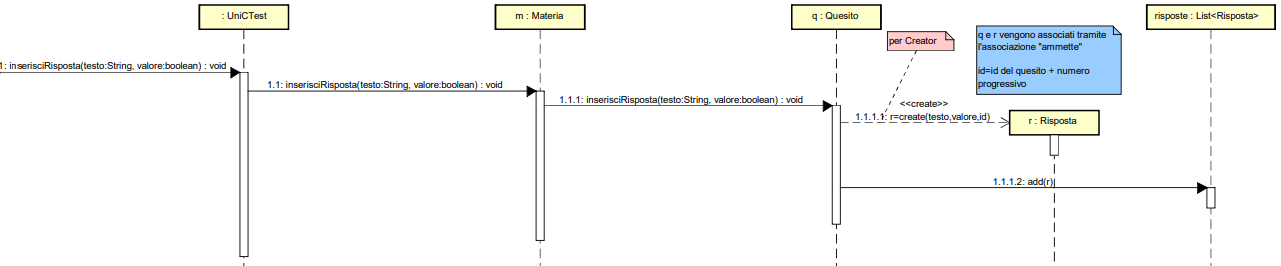
**Immagine che contiene testo

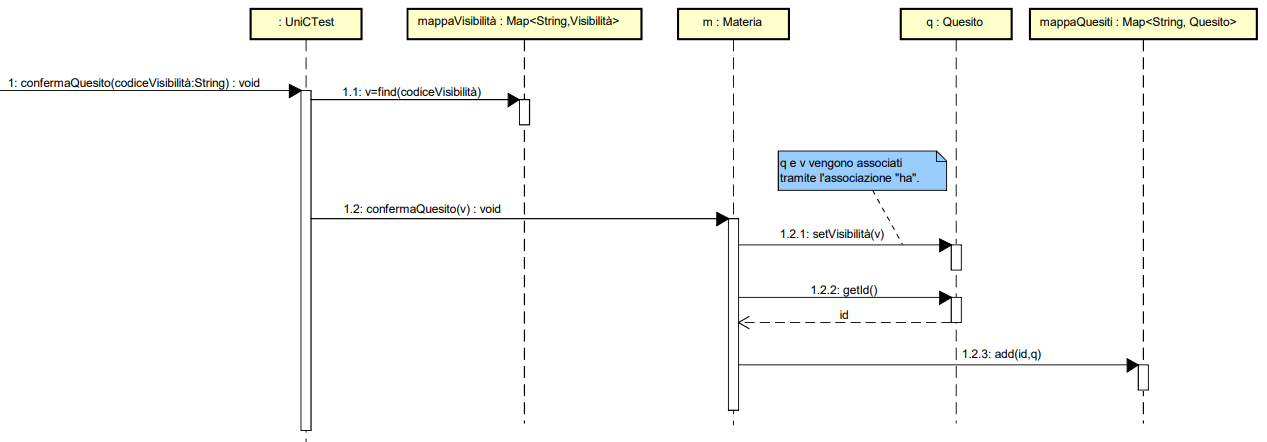
Descrizione generata automaticamente**

1. **SSD\_UC7\_inserisciTesto**

****

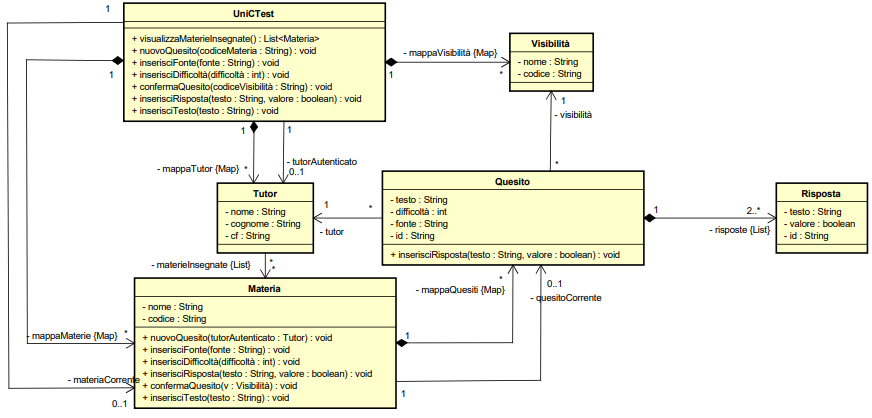
1. **SSD\_UC7\_inserisciDifficoltà**Immagine che contiene testo

   Descrizione generata automaticamente
2. **SSD\_UC7\_InserisciRisposta**
3. **SSD\_UC7\_confermaQuesito**



#### DCD, pattern Singleton

L’individuazione della classe UniCTest come rappresentativa dell’intero Sistema software suggerisce l’applicazione del pattern GoF **Singleton**, secondo cui UniCTest risulta essere una classe di cui è possibile istanziare un solo oggetto, tramite l’invocazione ad un metodo getInstance() incaricato di produrre oggetti che restituisce il riferimento ad un unico oggetto unictest (se esiste, altrimenti lo crea).



### Implementazione e Testing

#### Implementazione

Per l’Implementazione si rimanda al codice in allegato.

#### Testing

**TestNuovoQuesito.java**

I test della classe TestNuovoQuesito si occupano di verificare che la creazione di un QuesitoDescrizione, la sua inizializzazione e la sua aggiunta alla *collection* della Materia a cui fa riferimento avvengano correttamente.

initTest() si limita a impostare le condizioni iniziali: recupera l’istanza di UniCTest (Singleton) e autentica un Tutor.

testNuovoQuesito() si occupa di creare un nuovo QuesitoDescrizione e verificare che la materia del quesito sia impostata come corrente per il Tutor e che la sua istanza non sia null.

testVisualizzaMaterieInsegnate() verifica che la lista di Materie insegnate dal tutor autenticato restituita da visualizzaMaterie() non sia null.

I test nella forma testInserisci<informazione> si occupano di inserire l’informazione descritta nell’istanza corrente di QuesitoDescrizione e di verificare che sia uguale ai parametri attuali.

testConfermaQuesito() si occupa di verificare che il QuesitoDescrizione corrente sia stato correttamente aggiunto alla *collection* della Materia a cui il QuesitoDescrizione corrente fa riferimento e di verificare che tutte le istanze “correnti” vengano poste a *null*.

## Iterazione I – Refactoring

### Introduzione

Nei *workshop* destinati alla valutazione del lavoro finora effettuato e alla redazione degli scenari di sviluppo si è resa evidente una inconsistenza sul piano dell’Analisi Orientata agli Oggetti che ha interessato l’Iterazione I della fase di Elaborazione.

Pertanto, si è resa necessaria un’operazione di *refactoring*, ovvero dell’applicazione di una "tecnica strutturata per modificare la struttura interna di porzioni di codice senza modificarne il comportamento esterno" (Fowler), in questo caso applicata per migliorare la consistenza sul piano logico e la leggibilità e la manutenibilità del codice.

Di seguito si analizza il problema evidenziato e si illustra la soluzione individuata.

Sul piano logico, è l’attore Tutor a dover inserire dei quesiti nel Sistema. Tuttavia, nella prima Iterazione, alcune responsabilità destinate all’attore Tutor sono state affidate unicamente alla classe concettuale Materia. Ad esempio, nell’Iterazione I la classe concettuale rappresentante il sistema UniCTest riassegnava la responsabilità di creare un nuovoQuesito alla Materia, bypassando il Tutor.

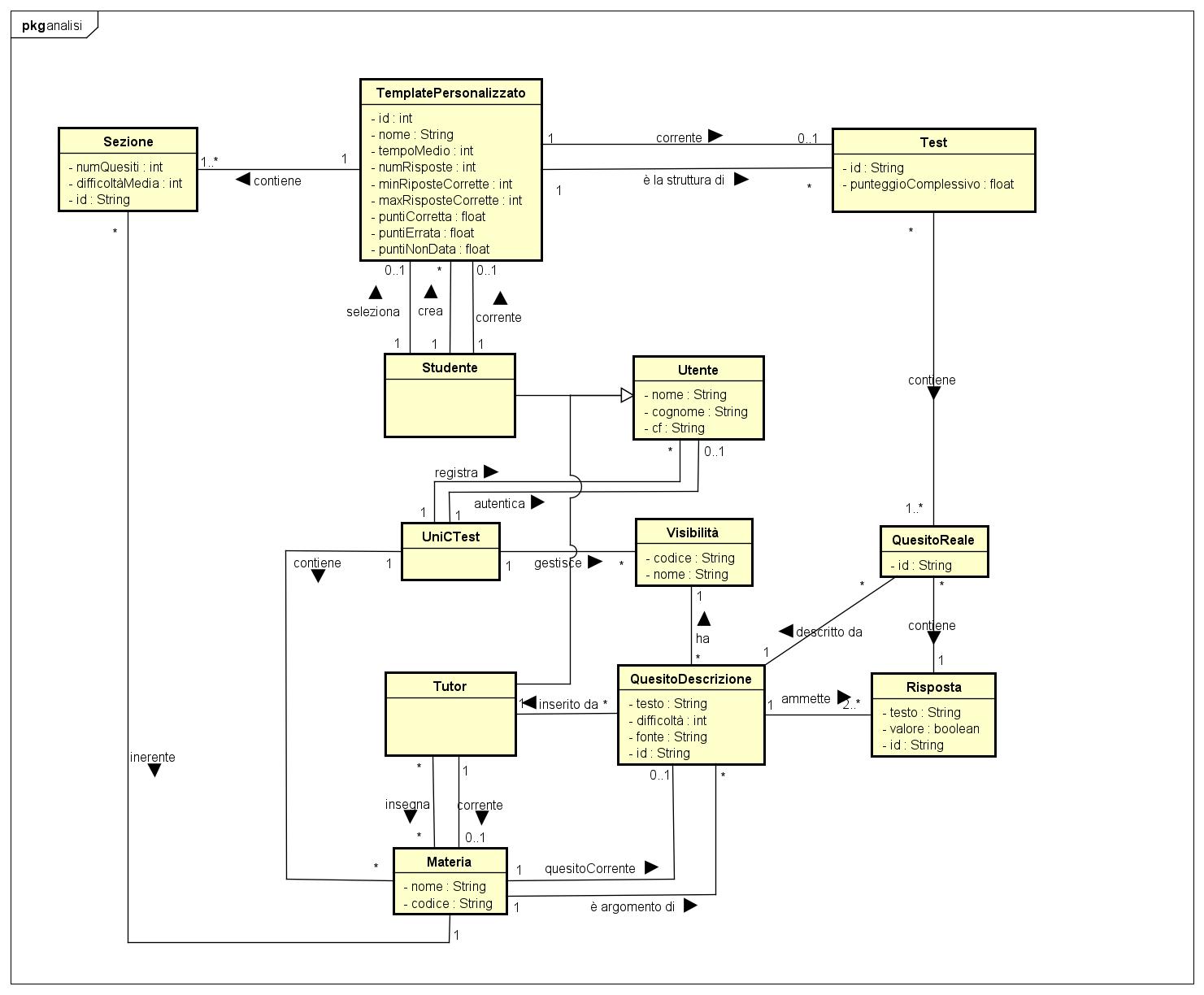
Oltre ad essere una stranezza sul piano logico, la responsabilità di accertarsi che il Tutor inserirà solo quesiti relativi alla propria Materia viene data solo ad UniCTest, la quale potrebbe in qualunque momento aggiungere quesiti a materie anche per conto di Tutor che in realtà non insegnano tali materie.

Se, invece, è il Tutor a dover inserire quesiti sulla base delle materie insegnate, il quesito non potrà mai essere inserito se il Tutor non è connesso alla Materia mediante l’associazione “insegna”, e quindi l’inconsistenza logica non potrà mai verificarsi.

Pertanto, la responsabilità di inserire un nuovo quesito è stata assegnata al Tutor.

### Analisi Orientata agli Oggetti

#### Modello di dominio

Il Modello di Dominio presenta dei cambiamenti nelle associazioni. Infatti, dato che il Quesito deve essere inserito dal Tutor, è quest'ultimo ad avere una materiaCorrente e non UniCTest.

#### Diagramma di Sequenza di Sistema

Il Diagramma di Sequenza di Sistema resta invariato. Infatti, l'interazione dell'attore Tutor con il Sistema non cambia poiché il comportamento del Sistema, visto dall'esterno, non deve essere mutato.

#### Contratti delle Operazioni

I Contratti delle Operazioni non presentano cambiamenti fatta eccezione per il CO2, riportato di seguito.

### CO2

|  |  |
| --- | --- |
| Operazione | nuovoQuesito(codiceMateria: String) |
| Riferimenti | UC7: Inserisci quesito |
| Pre-condizioni | - |
| Post-condizioni | * L’istanza *mi* di Materia avente *mi.codice* uguale a codiceMateria è stata associata all’istanza t di Tutor (autenticato) tramite l’associazione “corrente”; * È stata creata l’istanza *q* di Quesito ed è stato inizializzato l’attributo *q*.*id*; * *mi* è stata associata a *q* tramite l’associazione “corrente”. * *q* è stato associato all’istanza *t* di Tutor tramite l’associazione “inserito da”. |

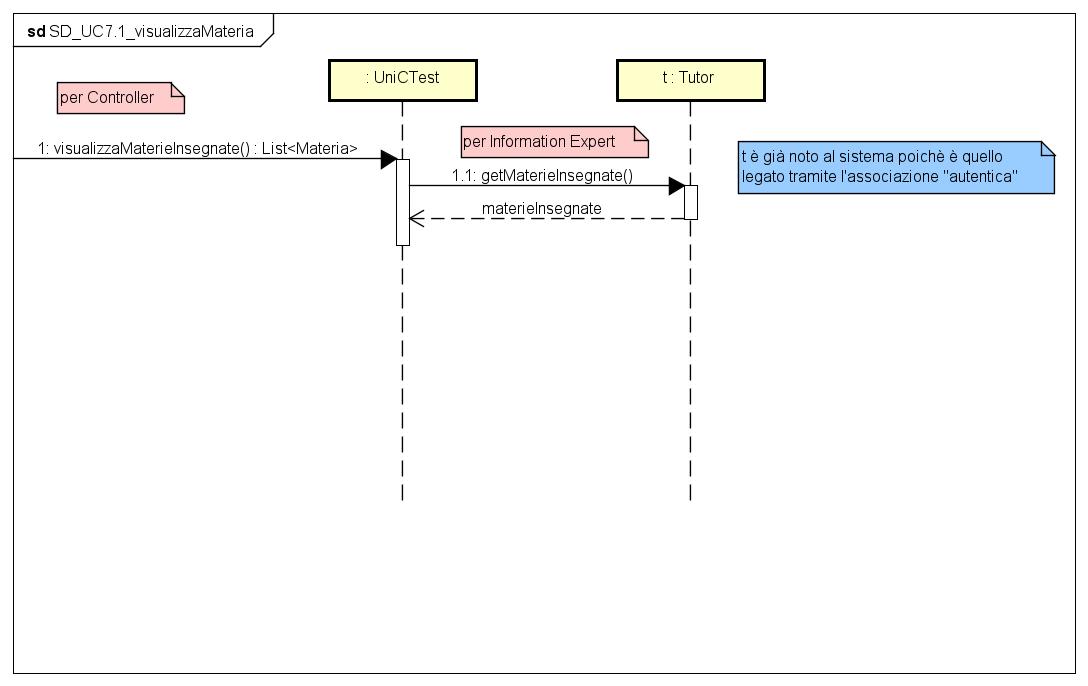
### Progettazione Orientata agli Oggetti

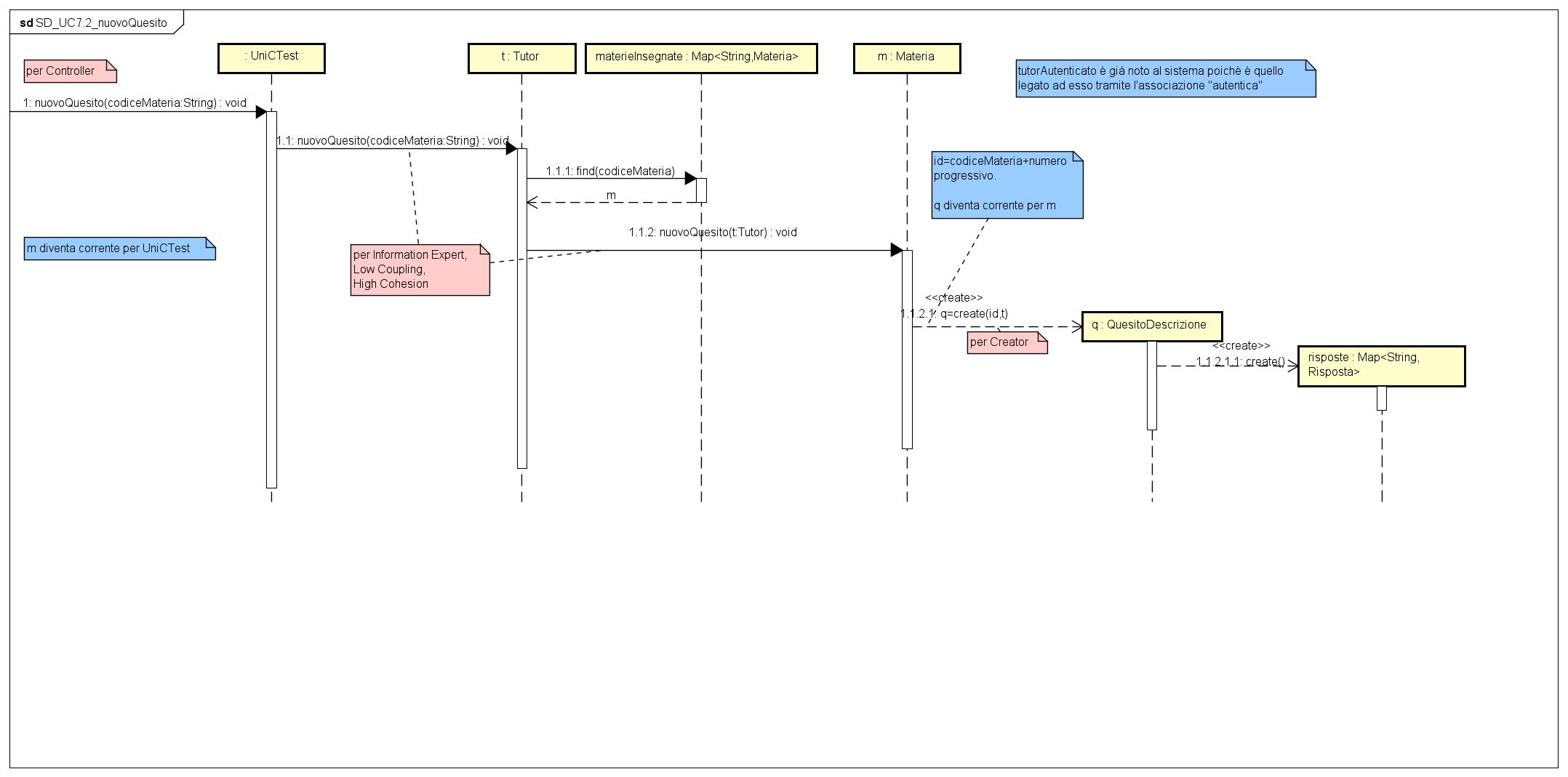
Il Modello di Progetto subisce mutamenti legati alle diverse assegnazioni delle responsabilità alle classi software.

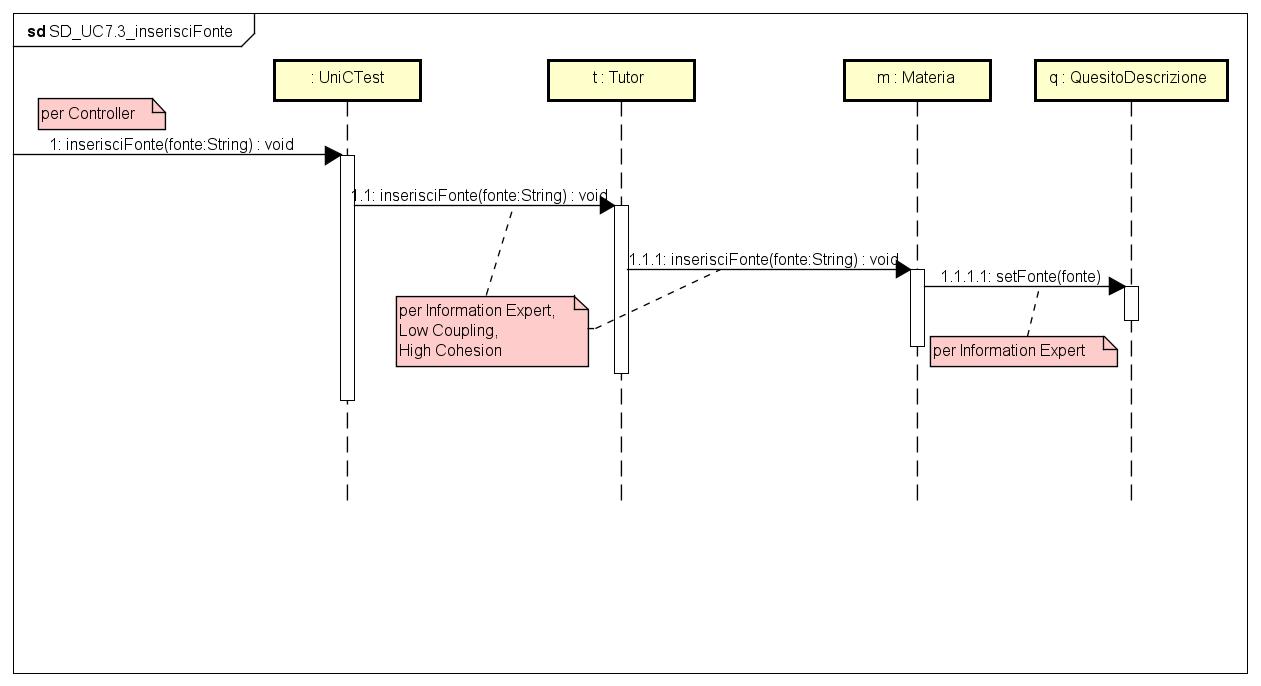
#### Diagrammi di interazione

I diagrammi comportamentali mutano in quanto l'interazione tra UniCTest e Materia ha adesso un livello di indirezione. Infatti, l'interazione UniCTest-Materia è mediata dal Tutor.

1. **SSD\_UC7\_visualizzaMateria**



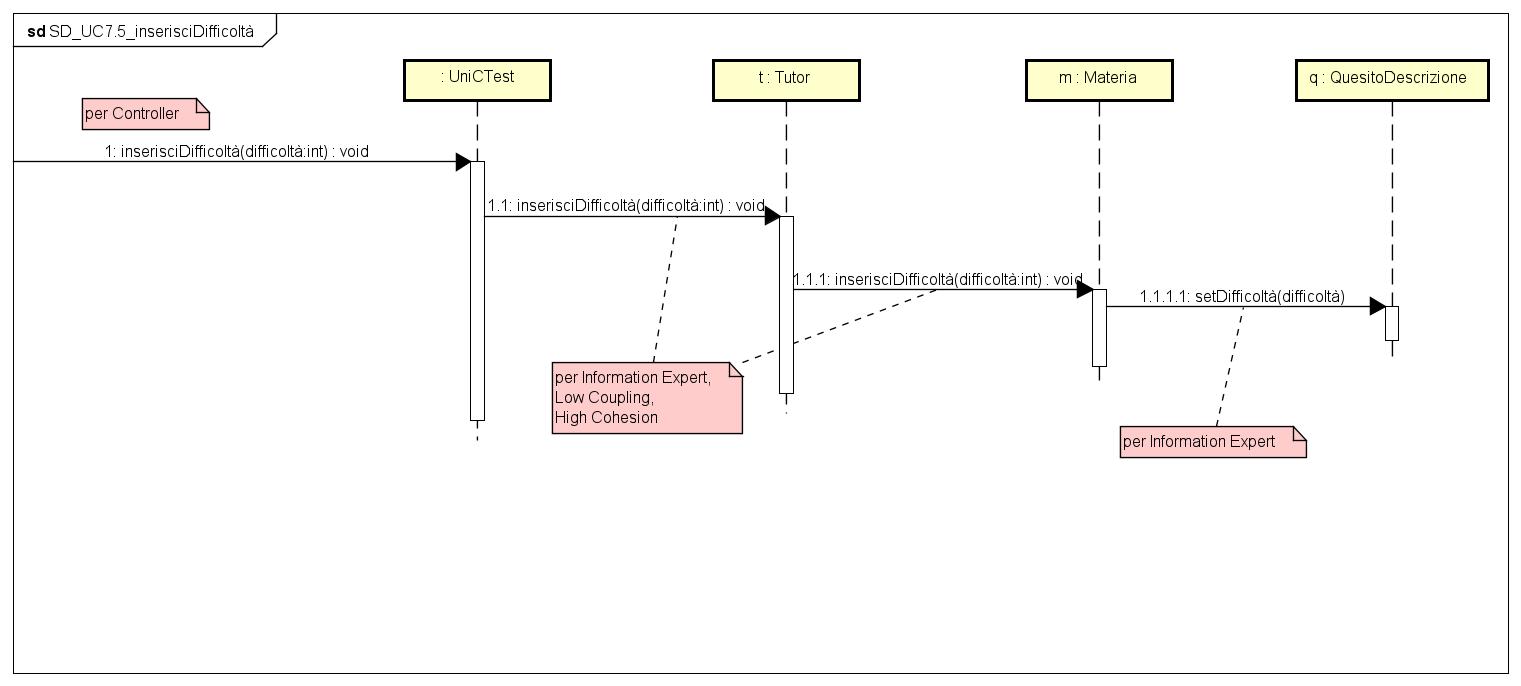
1. **SSD\_UC7\_nuovoQuesito**
2. **SSD\_UC7\_inserisciFonte**

****

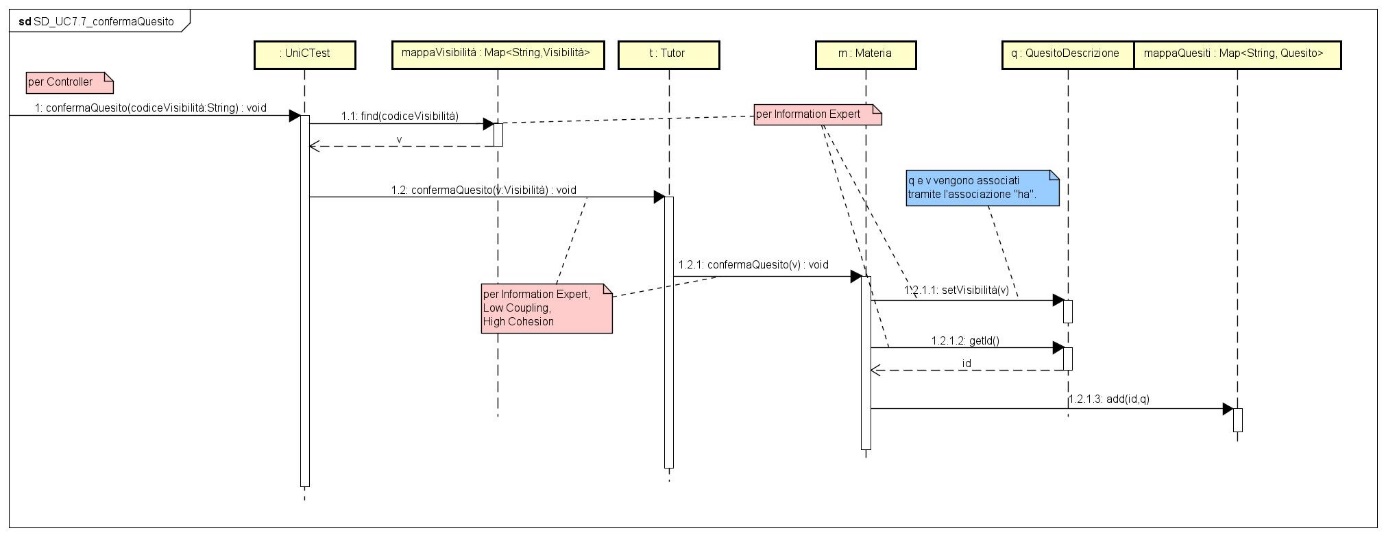
1. **SSD\_UC7\_inserisciTesto**

****

1. **SSD\_UC7\_inserisciDifficoltà**

****

1. **SSD\_UC7\_InserisciRisposta**

****

1. **SSD\_UC7\_confermaQuesito**

**Immagine che contiene testo

Descrizione generata automaticamente**

#### DCD

Il Diagramma delle Classi di Progetto (DCD) vede l'aggiunta delle responsabilità al Tutor, responsabilità che prima non erano presenti.

## Iterazione II

### Introduzione e Elaborati

Nella 2° Iterazione della fase di Elaborazione, tra i vari Casi d’Uso individuati nella fase di Ideazione, si è scelto di focalizzare l’attenzione su:

* Scenario di successo di UC2: Crea template di test personalizzato;
* Aggiornamento del Caso d’Uso di avviamento alla luce di analisi, progettazione e implementazione di UC2.

Si è scelto di realizzare i seguenti artefatti:

* Analisi Orientata agli Oggetti
  + **Modello di Dominio**
  + **SSD (System Sequence Diagrams)**
  + **Contratti delle operazioni**
* Progettazione Orientata agli Oggetti:
  + **Diagrammi di interazione**
  + **DCD (Design Class Diagram)**

La 2° Iterazione della fase di Elaborazione prevederà diversi step:

1. Analisi orientata agli oggetti (OOA). Bisognerà aggiornare il Modello di Dominio. Le nuove classi concettuali da introdurre saranno individuate tra le locuzioni nominali dello scenario di successo di UC2.
2. Progettazione orientata agli oggetti (OOD). Il Modello di Progetto costruito nella Iterazione 1 della fase di Elaborazione andrà aggiornato.
3. Implementazione.

### Analisi Orientata agli Oggetti

#### Modello di dominio

Si riporta lo scenario di successo di UC2. Vengono evidenziate le locuzioni nominali per la scelta delle classi concettuali candidate. Le classi concettuali candidate verranno evidenziate in verde, gli attributi in giallo.

**UC2. Crea template di test personalizzato**

|  |  |
| --- | --- |
| Scenario principale di successo | 1. Lo Studente vuole creare un nuovo template.  2. Lo Studente sceglie l’attività “Crea template personalizzato” e inserisce il nome del template che vuole creare. Il Sistema registra le informazioni inserite.  3. Lo Studente inserisce:   * il formato dei quesiti (cioè il numero di risposte e il numero di risposte corrette per quesito) * il punteggio attribuito ad un singolo quesito (nel caso di risposta corretta, risposta errata e risposta non data) * il tempo medio di risoluzione di un singolo quesito.   Il Sistema registra le informazioni inserite.  4. Il Sistema restituisce l’elenco delle materie disponibili.  5. Lo Studente seleziona una materia che vuole inserire nel template.  6. Lo Studente seleziona il numero di quesiti previsti per la materia scelta e la difficoltà media. Il Sistema registra le informazioni inserite.  *I passi 4, 5 e 6 vengono ripetuti finché serve.*  7. Lo Studente indica di aver finito. |

Le classi concettuali candidate sono:

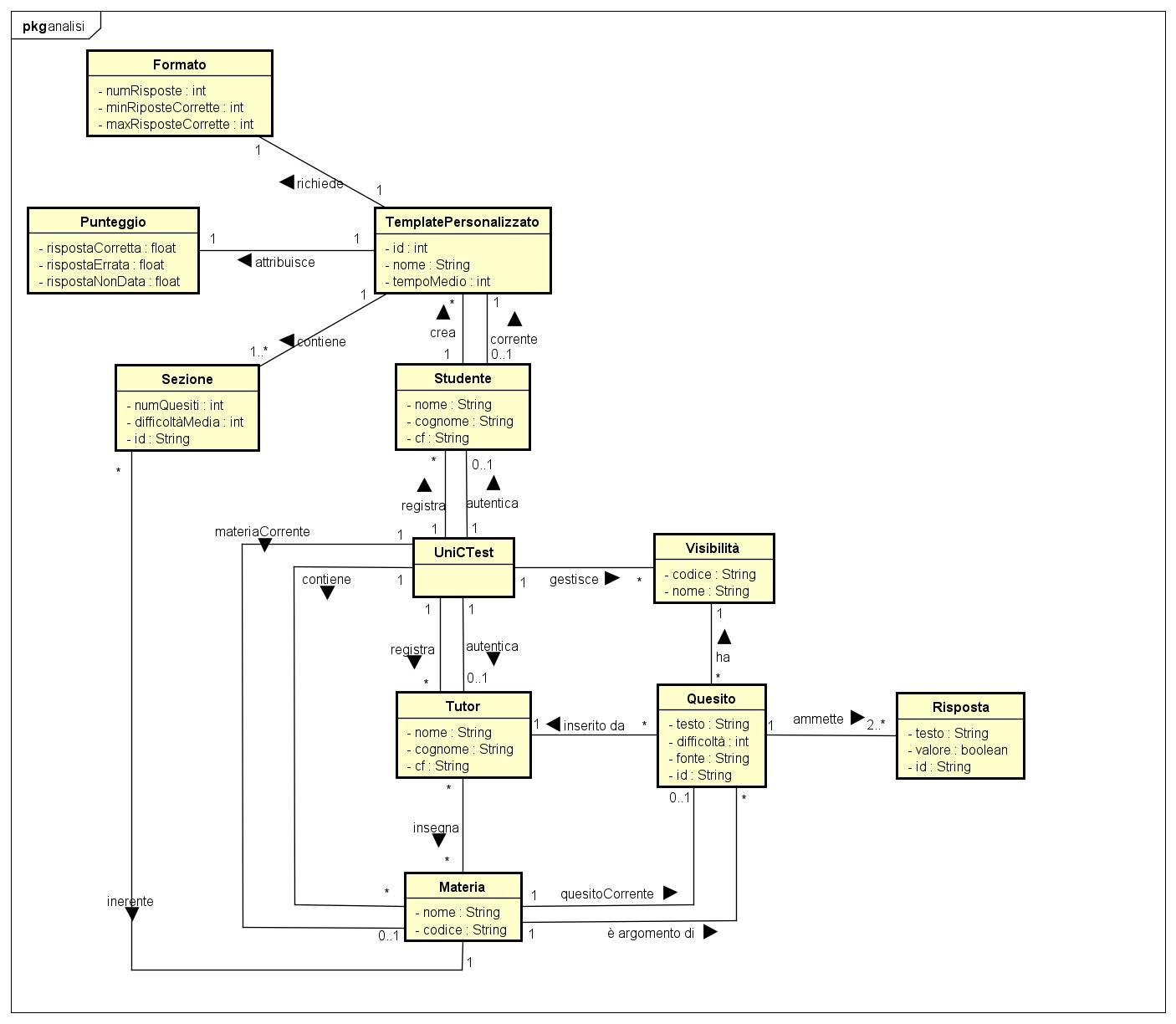
* **Studente**: utente del sistema che vuole creare un template di test;
* **TemplatePersonalizzato**: contiene informazioni utili descrivere un possibile test;
* **Formato**: indica il formato (dei quesiti) ammesso per il template;
* **Punteggio**: indica il punteggio attribuito ad ogni quesito in caso di risposta esatta, errata o non data.

Dall’analisi effettuata si è resa necessaria l’introduzione di una classe concettuale che non figura nello scenario di successo di UC2 ma che agevola la modellazione.

Per cui, alla lista precedente viene aggiunta la classe concettuale:

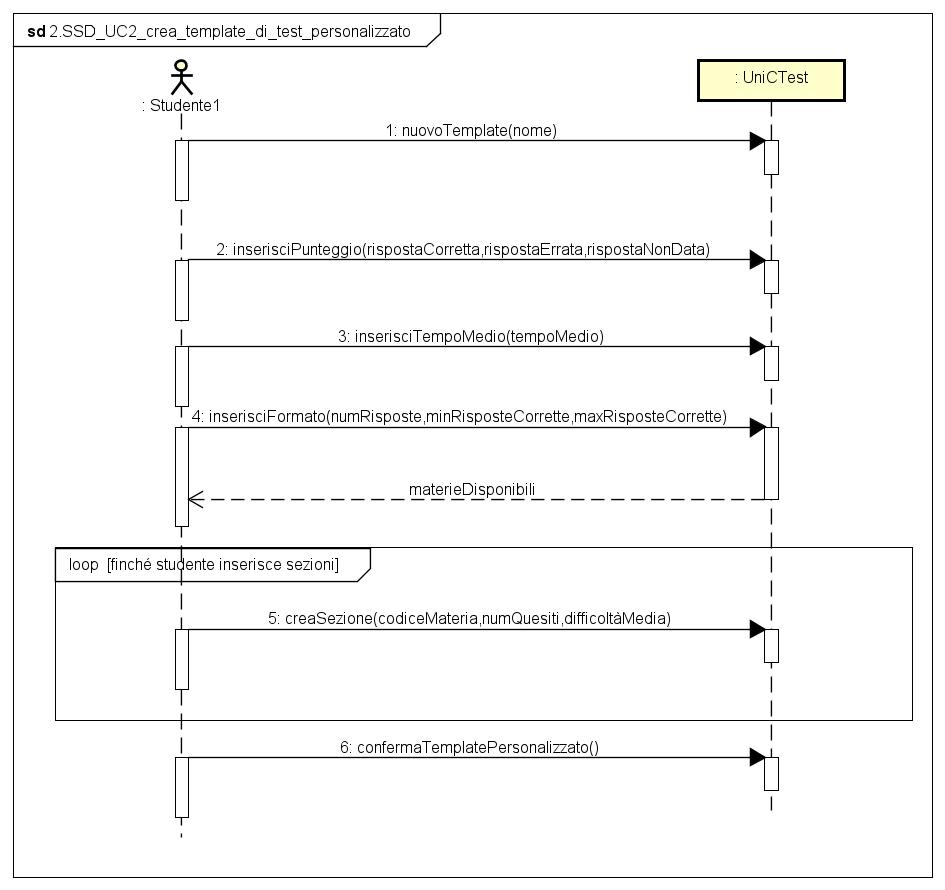
* **Sezione**: ha un riferimento alla materia scelta, il numero di quesiti inerenti ad essa e la loro difficoltà media.

Da cui, tenendo conto di associazioni e attributi, è stato ricavato il seguente Modello di Dominio, comprendente anche le classi concettuali ricavate nella fase di OOA dell’Iterazione precedente:



#### Diagramma di Sequenza di Sistema

Una volta aggiornato il Modello di Dominio, viene creato il SSD relativo a UC2, che consente di visualizzare in forma grafica l’interazione tra lo Studente ed il Sistema.



#### Contratti delle Operazioni

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC2.

### CO1

|  |  |
| --- | --- |
| Operazione | nuovoTemplate(nome: String) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | - |
| Post-condizioni | * È stata creata l’istanza *tp* di TemplatePersonalizzato con *tp.nome*=nome ed è stata associata allo Stuende s tramite l’associazione “corrente”. |

### CO2

|  |  |
| --- | --- |
| Operazione | inserisciPunteggio(rispostaCorretta: float, rispostaErrata: float, rispostaNonData: float) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * È stata inizializzata l’istanza *p* di Punteggio ed è stata associata a *tp* tramite l’associazione “attribuisce”. |

### CO3

|  |  |
| --- | --- |
| Operazione | inserisciTempoMedio(tempoMedio: int) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * L’attributo *tp*.*tempoMedio* è stato inizializzato a tempoMedio. |

### CO4

|  |  |
| --- | --- |
| Operazione | inserisciFormato(numRisposte: int, minRisposteCorrette:int, maxRisposteCorrette: int) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * È stata inizializzata l’istanza *f* di Formato ed è stata associata a *tp* tramite l’associazione “richiede”. * Il Sistema restituisce allo Studente le   istanze *mi* delle Materie disponibili. |

### CO5

|  |  |
| --- | --- |
| Operazione | creaSezione(codiceMateria: String, numQuesiti int, difficoltàMedia int) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * È stata inizializzata l’istanza *s* di Sezione ed è stata associata a *tp* tramite l’associazione “contiene”. * L’istanza *s* è stata associata all’istanza *m* di Materia avente l’attributo *m.codice* uguale a codiceMateria. |

### CO6

|  |  |
| --- | --- |
| Operazione | confermaTemplatePersonalizzato() |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * L’istanza *tp* è stata associata all’istanza *s* dello Studente tramite l’associazione “crea”. |

### Revisione Analisi e Inizio Progettazione Orientata agli Oggetti

Aggiornate le classi concettuali nel contesto della OOA, si passa alla OOD, realizzando il Modello di Progetto ed in particolare aggiornando il DCD (visualizzazione statica delle classi software), aggiornamento da effettuare il parallelo alla realizzazione dei diagrammi di interazione (visualizzazione dinamica delle classi software).

Durante la fase di progettazione ci si è resi conto che le classi Formato e Punteggio hanno una buona valenza in quanto classi concettuali ma scarsa valenza in quanto classi software, dato che sono legate a TemplatePersonalizzato mediante associazioni 1-1 che non sono destinate a diventare associazioni con molteplicità maggiore. Dunque si è scelto di eliminare Punteggio e Formato, mettendo i loro attributi dentro TemplatePersonalizzato.

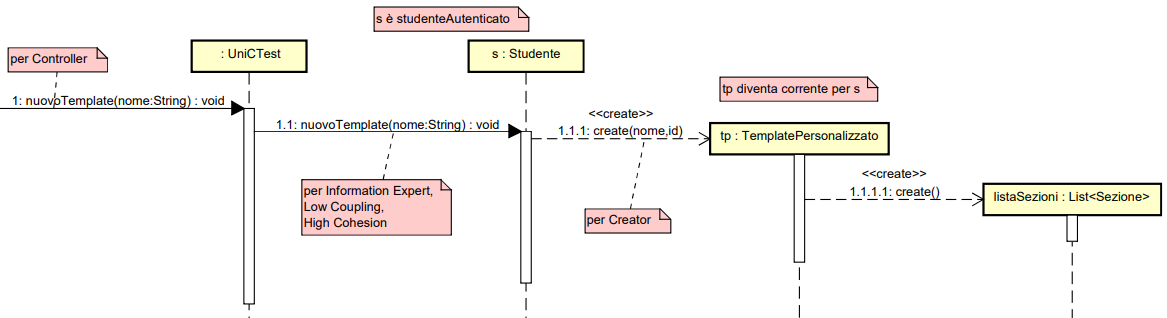
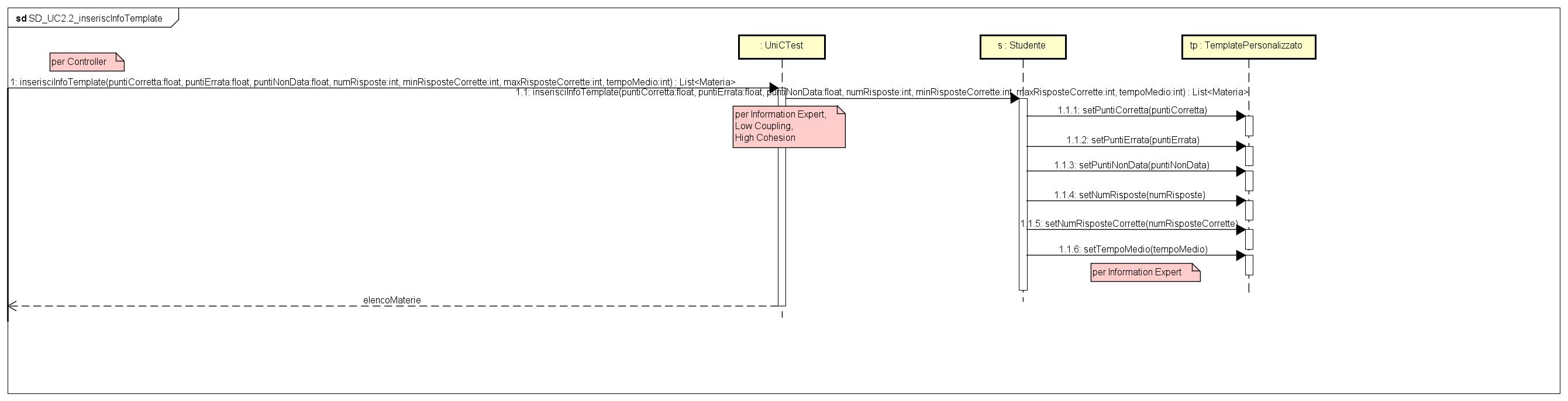
Questo cambiamento non altera le funzionalità esterne del Sistema ma ha conseguenze positive sulle performance e sulla memoria. Dunque, le operazioni descritte dai contratti 2, 3 e 4 possono essere riassunte in un’unica operazione. Si riporta di seguito il contratto di quest’ultima.

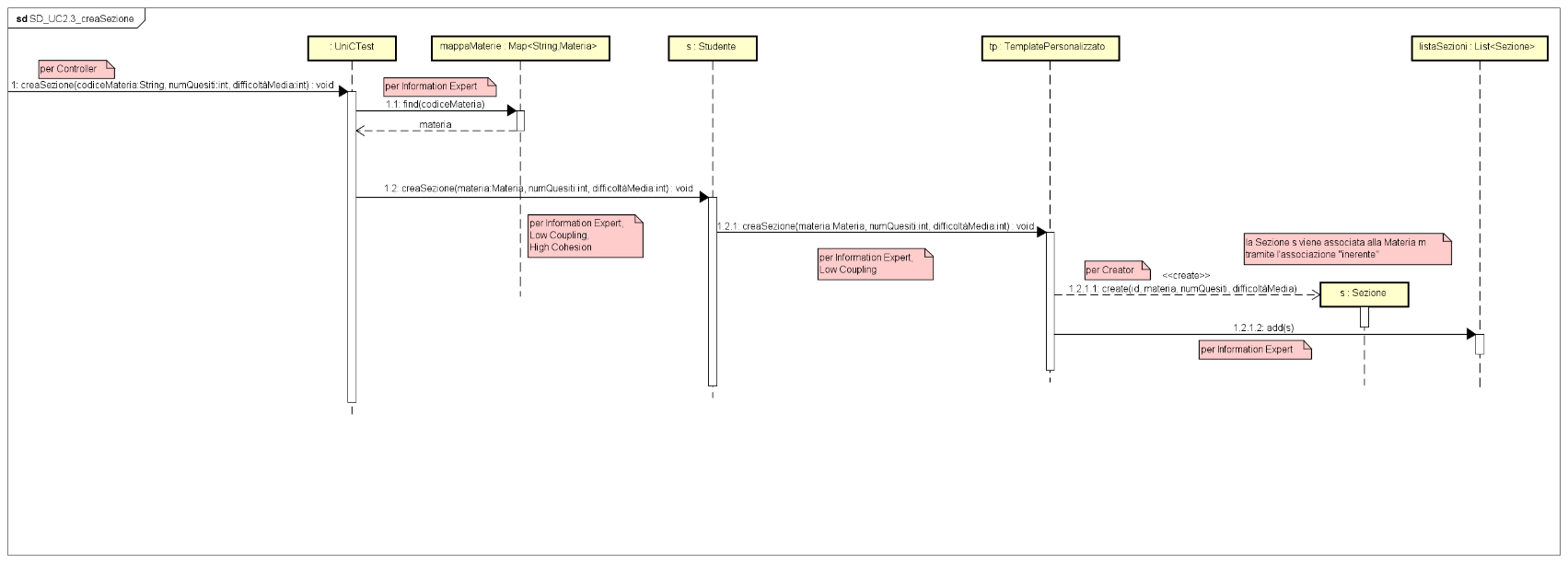
### CO2 bis

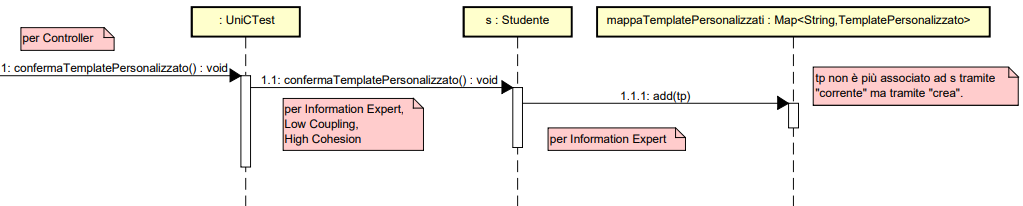
|  |  |
| --- | --- |
| Operazione | inserisciInfoTemplate(puntiCorretta: float, puntiErrata: float, puntiNonData: float, tempoMedio: int, numRisposte: int, minRisposteCorrette: int, maxRisposteCorrette: int) |
| Riferimenti | UC2: Crea template di test personalizzato |
| Pre-condizioni | È in corso la creazione di un TemplatePersonalizzato *tp*. |
| Post-condizioni | * Sono stati inizializzati gli attributi di *tp* e sono state restituite le istanze *mi* delle Materie disponibili. |

**NOTA**: i nomi degli attributi di Punteggio (trasferiti in TemplatePersonalizzato) sono stati rinominati in: puntiCorretta, puntiErrata, puntiNonData per una maggiore comprensibilità.

#### Diagrammi di interazione

1. **SD\_UC2\_creaTemplate**
2. **SD\_UC2\_inserisciInfoTemplate**
3. **SD\_UC2\_creaSezione**

****

1. **SD\_UC2\_confermaTemplate**

#### DCD

### Implementazione e Testing

#### Implementazione

Per l’Implementazione si rimanda al codice in allegato.

#### Testing

**TestCreaTemplatePers.java**

I test della classe TestCreaTemplatePers si occupano di verificare che la creazione di un TemplatePersonalizzato, la sua inizializzazione e la sua aggiunta alla *collection* dello Studente avvengano nella maniera corretta.

initTest() si limita a impostare le condizioni iniziali: recupera l’istanza di UniCTest (Singleton) e autentica uno Studente.

Ogni metodo creato, per non fallire, non dovrebbe lanciare un'eccezione perchè è stato autenticato uno Studente, che dovrebbe avere sempre i permessi per creare un TemplatePersonalizzato.

In generale, i metodi sviluppati rispettano il seguente meccanismo di verifica: verificare che la creazione delle istanze delle classi sia avvenuta correttamente, verificare che i parametri settati non siano *null*, verificare che le istanze siano aggiunte alle *collection* corrette.

Di seguito vengono analizzati i test nello specifico.

testNuovoTemplate() si occupa di creare un nuovo TemplatePersonalizzato e verificare che l’istanza del TemplatePersonalizzato venga impostata come corrente per lo Studente utenteAutenticato.

testInserisciInfoTemplate() verifica che tutte le informazioni aggiunte al TemplatePersonalizzato siano state aggiunte correttamente (*i.e.* sono uguali ai parametri attuali).

testCreaSezione() si occupa di verificare che la creazione di una Sezione avvenga correttamente, che venga aggiunta alla listaSezioni del TemplatePersonalizzato, che la Materia della Sezione sia quella passata come parametro attuale e che il resto dei parametri combaci con i parametri attuali.

testConfermaTemplate() verifica che il TemplatePersonalizzato sia stato correttamente aggiunto alla mappa di TemplatePersonalizzati dello Studente.

## Iterazione III

### Introduzione e Elaborati

Nella 3° Iterazione della fase di Elaborazione si è scelto di focalizzare l’attenzione su:

* Caso d’Uso completo UC1 (Scenario di successo ed estensioni *4c* e *8c*): Avvia simulazione;
* Aggiornamento del Caso d’Uso di avviamento alla luce di analisi, progettazione e implementazione di UC1.

Si è scelto di realizzare i seguenti artefatti:

* Nel contesto dell’Analisi Orientata agli Oggetti:
  + **Modello di Dominio** (da aggiornare)
  + **SSD (System Sequence Diagrams)**
  + **Contratti delle operazioni**
* Nel contesto della Progettazione Orientata agli Oggetti:
  + **Diagrammi di interazione**
  + **DCD (Design Class Diagram)** (da aggiornare)

La 3° Iterazione della fase di Elaborazione prevederà diversi step:

1. Analisi orientata agli oggetti (OOA). Bisognerà aggiornare il Modello di Dominio. Le nuove classi concettuali da introdurre saranno individuate tra le locuzioni nominali di UC1.
2. Progettazione orientata agli oggetti (OOD). Il Modello di Progetto costruito nelle iterazioni 1 e 2 della fase di Elaborazione andrà aggiornato.
3. Implementazione. Bisognerà aggiornare il Caso d’Uso di avviamento e implementare le classi software, i metodi e l’interfaccia utente relativi a UC1.

### Analisi Orientata agli Oggetti

#### Modello di dominio

Si riporta lo scenario di successo di UC1. Vengono evidenziate le locuzioni nominali per la scelta delle classi concettuali candidate. Le classi concettuali candidate verranno evidenziate in verde, gli attributi in giallo.

**UC1. Avvia simulazione**

|  |  |
| --- | --- |
| Scenario principale di successo | 1. Lo Studente vuole iniziare una nuova simulazione.  2. Lo Studente sceglie l’attività “Avvia simulazione”.  3. Lo Studente sceglie il template su cui basare il test.  4. Il Sistema crea un test contenente quesiti sulla base del template scelto al punto 3 e lo restituisce allo Studente.  5. Lo Studente risponde ad una domanda del test.  *Il passo 5 viene ripetuto finché lo Studente non indica di voler consegnare.*  6. Lo Studente seleziona la voce “consegna il test”.  7. Il Sistema restituisce allo Studente un riepilogo della simulazione appena svolta, con tutte le risposte date.  8. Lo Studente seleziona la voce “conferma”.  9. Il Sistema corregge il test e calcola il punteggio complessivo. Il Sistema registra l’esito. Il Sistema restituisce il test corretto allo Studente. |

Le classi concettuali candidate sono:

* **Test**: insieme di quesiti scelti sulla base di un template di test, contiene un punteggio complessivo;
* **Risposta data**: corrisponde alla risposta data ad un quesito presente nel test (eventualmente più di una).

Dall’analisi effettuata si è resa necessaria l’introduzione di una classe concettuale che non figura nello scenario di successo di UC1 ma che agevola la modellazione.

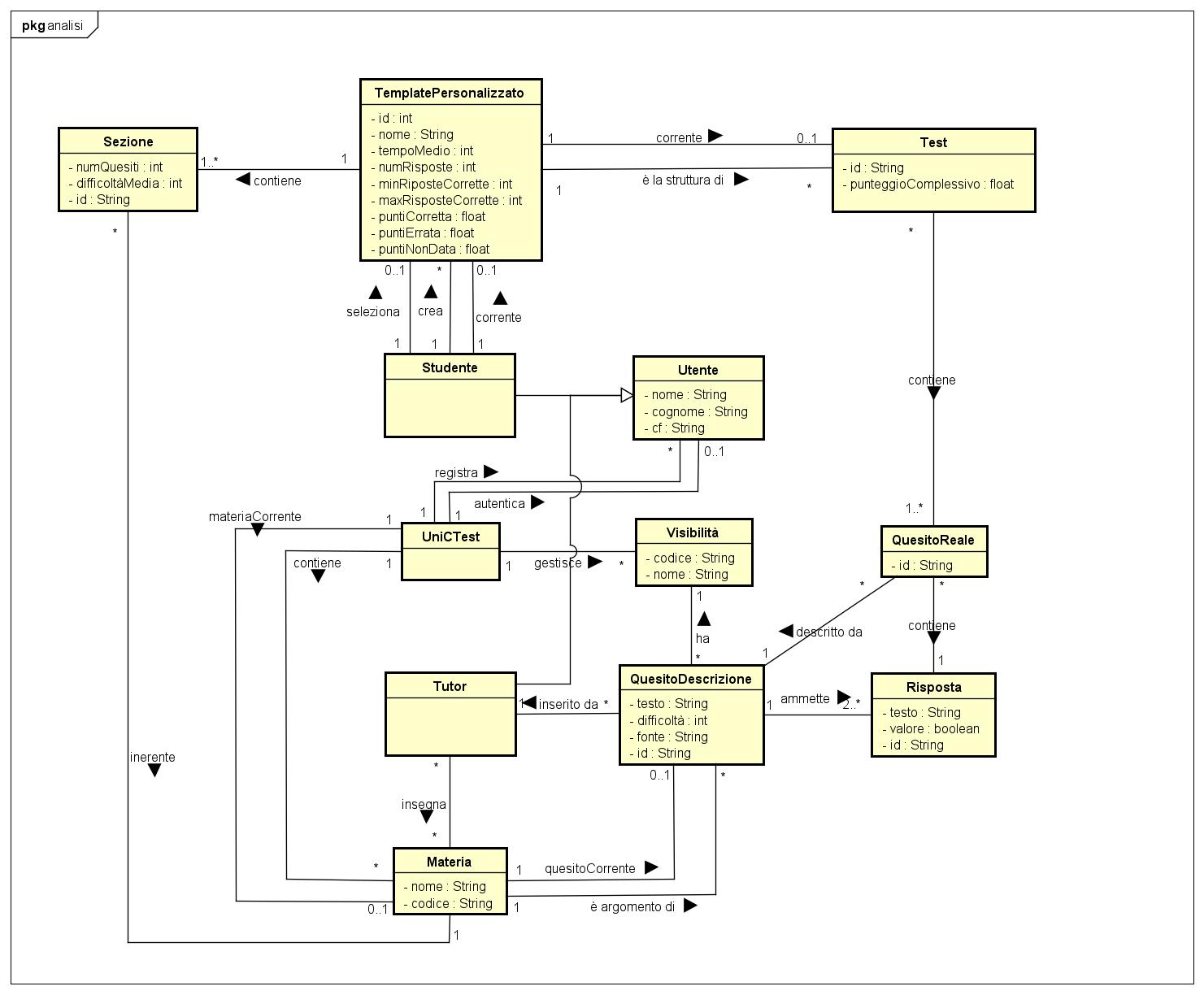
Per cui, alla lista precedente viene aggiunta la classe concettuale:

* **QuesitoReale**: esso rappresenta un quesito associato ad un test oggetto di una simulazione. Ha un’associazione con la classe Quesito, adesso rinominata in QuesitoDescrizione, per ottenere le informazioni come il testo della domanda e le risposte e un’associazione con Risposta, ovvero la risposta data dallo studente al quesito.

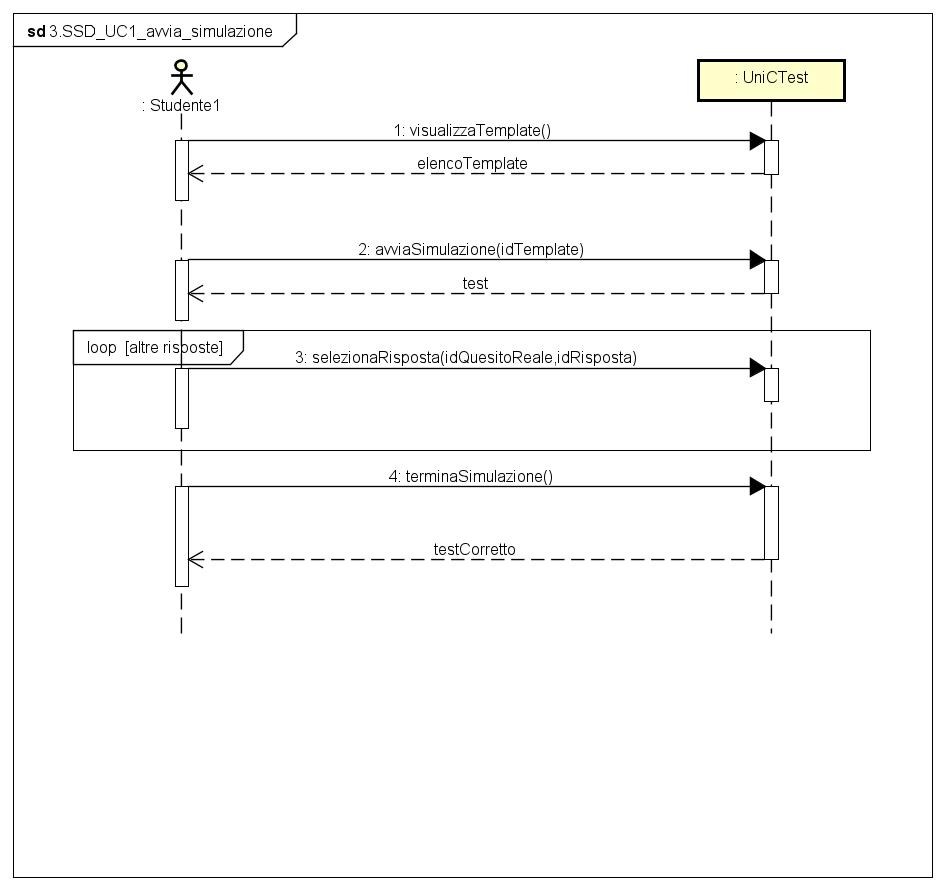
Questa modifica è stata effettuata in quanto un quesito potrebbe figurare in molti test, dunque si è rivelato utile mantenere separati i concetti di descrizione del quesito da quello del quesito realmente contenuto in un test. Altrimenti esisterebbero delle classi duplicate.

Inoltre, come aggiunta all’Iterazione precedente, è stato introdotto il concetto di Utente per generalizzare Studente e Tutor, in quanto ciò permette di ridurre il numero di associazioni con UniCTest, abbassandone, dunque, il grado di accoppiamento.

È stato ricavato il seguente Modello di Dominio, comprendente anche le classi concettuali ricavate nella fase di OOA dell’Iterazione precedente:



#### Diagramma di Sequenza di Sistema

Una volta aggiornato il Modello di Dominio, viene creato il SSD relativo a UC1, che consente di visualizzare in forma grafica l’interazione tra lo Studente ed il Sistema.

#### Contratti delle Operazioni

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC1.

### CO1

|  |  |
| --- | --- |
| Operazione | visualizzaTemplate() |
| Riferimenti | UC1: Avvia simulazione |
| Pre-condizioni | - |
| Post-condizioni | * È stato restituito l’elenco delle istanze *tp* di TemplatePersonalizzato allo Studente *utenteAutenticato*. |

### CO2

|  |  |
| --- | --- |
| Operazione | avviaSimulazione(idTemplate) |
| Riferimenti | UC1: Avvia simulazione |
| Pre-condizioni | - |
| Post-condizioni | * È stata creata l’istanza *t* di Test ed è stata associata all’istanza *tp* di TemplatePersonalizzato avente *tp.id*=idTemplate tramite l’associazione “corrente”; * Sono state create le n istanze *qr* di QuesitoReale sulla base delle informazioni contenute nell’istanza *tp,* ciascuna è stata associata alla rispettiva istanza *qd* di QuesitoDescrizione tramite l’associazione “descritto da” e a *t* tramite “contiene”; * *t* è stato restituito allo Studente. |

### CO3

|  |  |
| --- | --- |
| Operazione | selezionaRisposta(idQuesitoReale,idRisposta) |
| Riferimenti | UC1: Avvia simulazione |
| Pre-condizioni | È in corso una simulazione di test e l’istanza *t* di Test è stata creata e associata correttamente alle n istanze *qr* di QuesitoReale. |
| Post-condizioni | * L’istanza *qr* con *qr.id*=idQuesitoReale è stata associata all’istanza *r* di Risposta avente *r.id*=idRisposta. |

### CO4

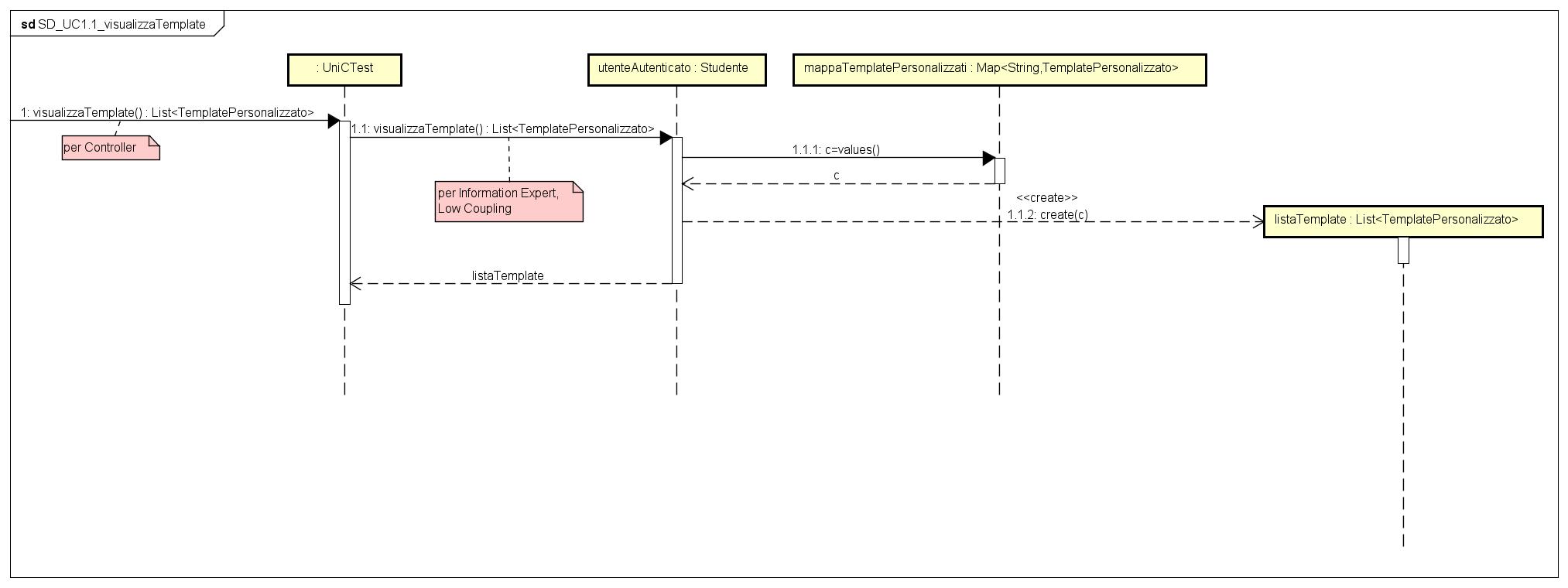
|  |  |
| --- | --- |
| Operazione | terminaSimulazione() |
| Riferimenti | UC1: Avvia simulazione |
| Pre-condizioni | È in corso una simulazione di test e l’istanza *t* di Test è stata creata e associata correttamente alle n istanze *qr* di QuesitoReale. |
| Post-condizioni | * L’attributo *t.punteggioComplessivo* è stato inizializzato ed è stato restituito *t* allo Studente; * *t* viene associato all’istanza *tp* di TemplatePersonalizzato attualmente selezionata dallo Studente tramite l’associazione “è la struttura di”. |

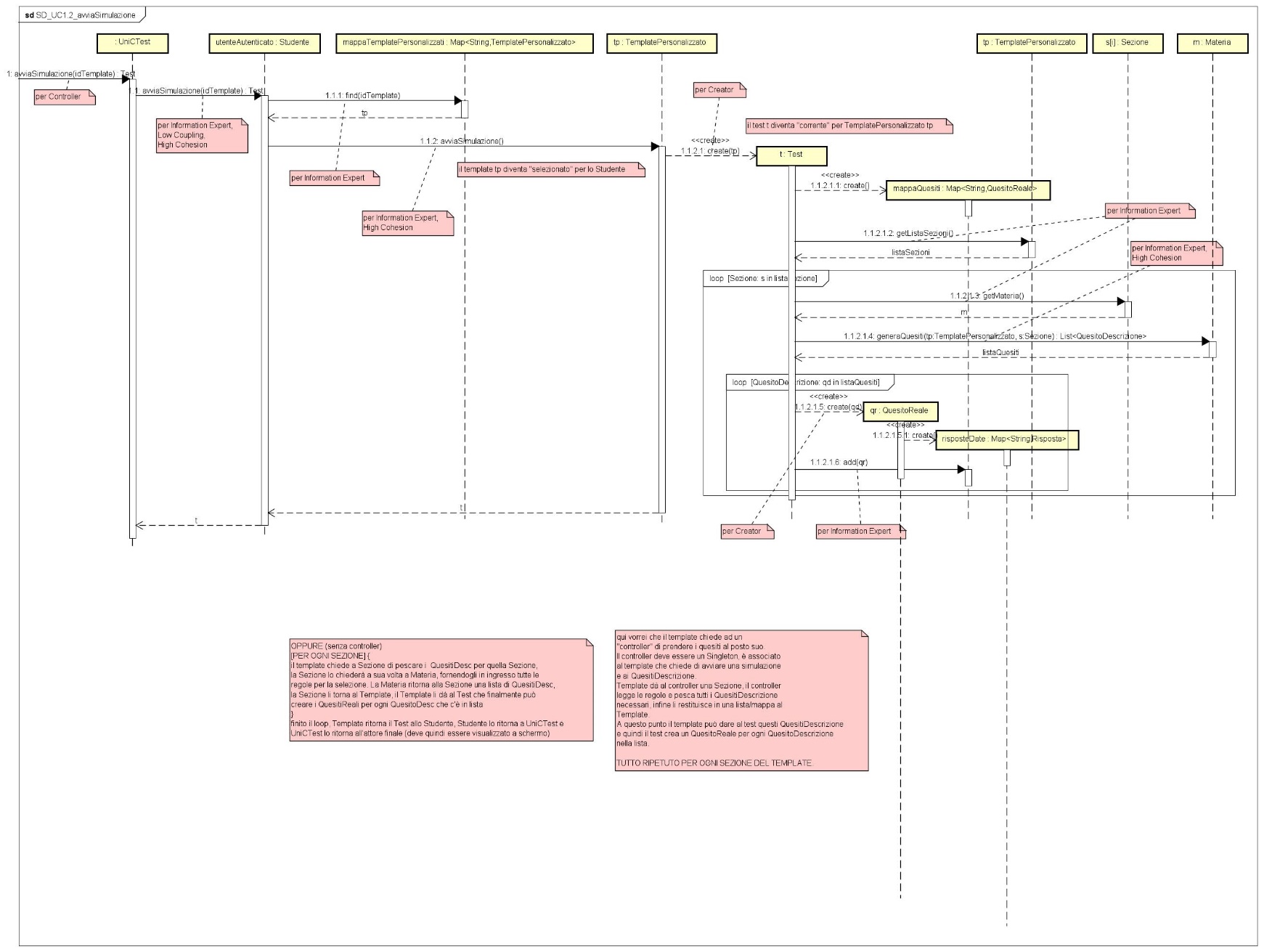
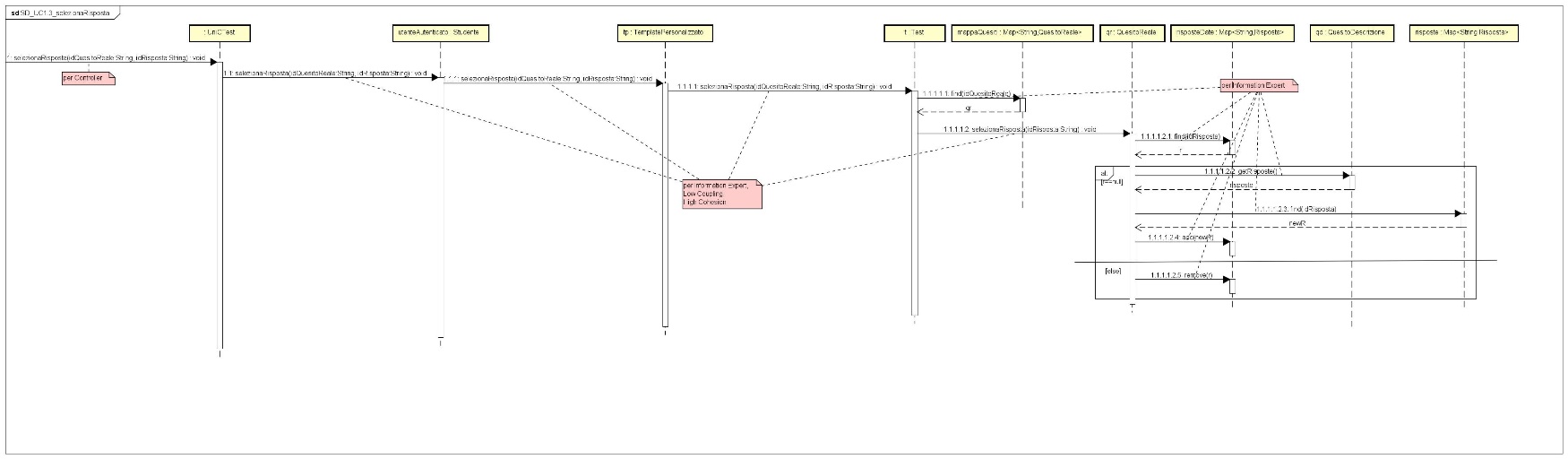
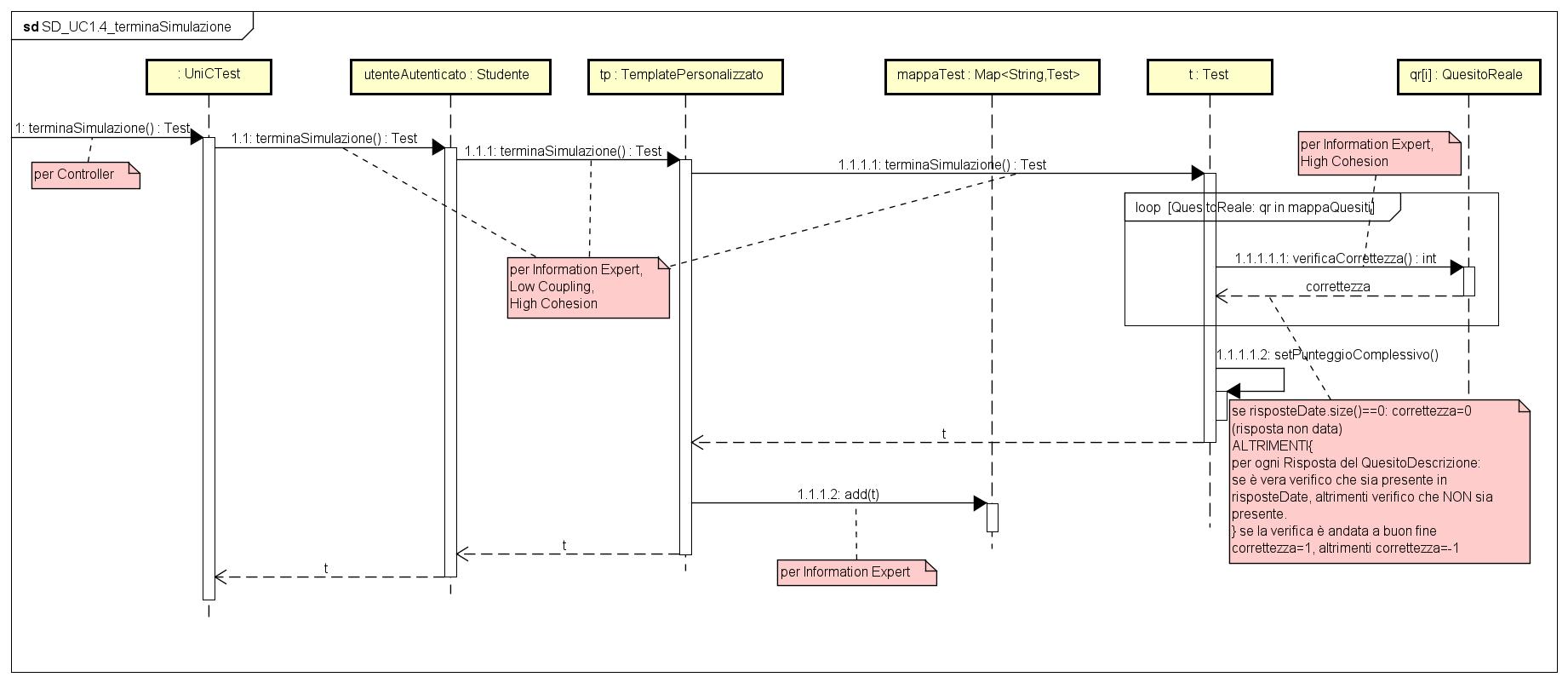
### Progettazione Orientata agli Oggetti

Aggiornate le classi concettuali nel contesto della OOA, si passa alla OOD, realizzando il Modello di Progetto ed in particolare aggiornando il DCD (visualizzazione statica delle classi software), aggiornamento da effettuare il parallelo alla realizzazione dei diagrammi di interazione (visualizzazione dinamica delle classi software).

#### Diagrammi di interazione

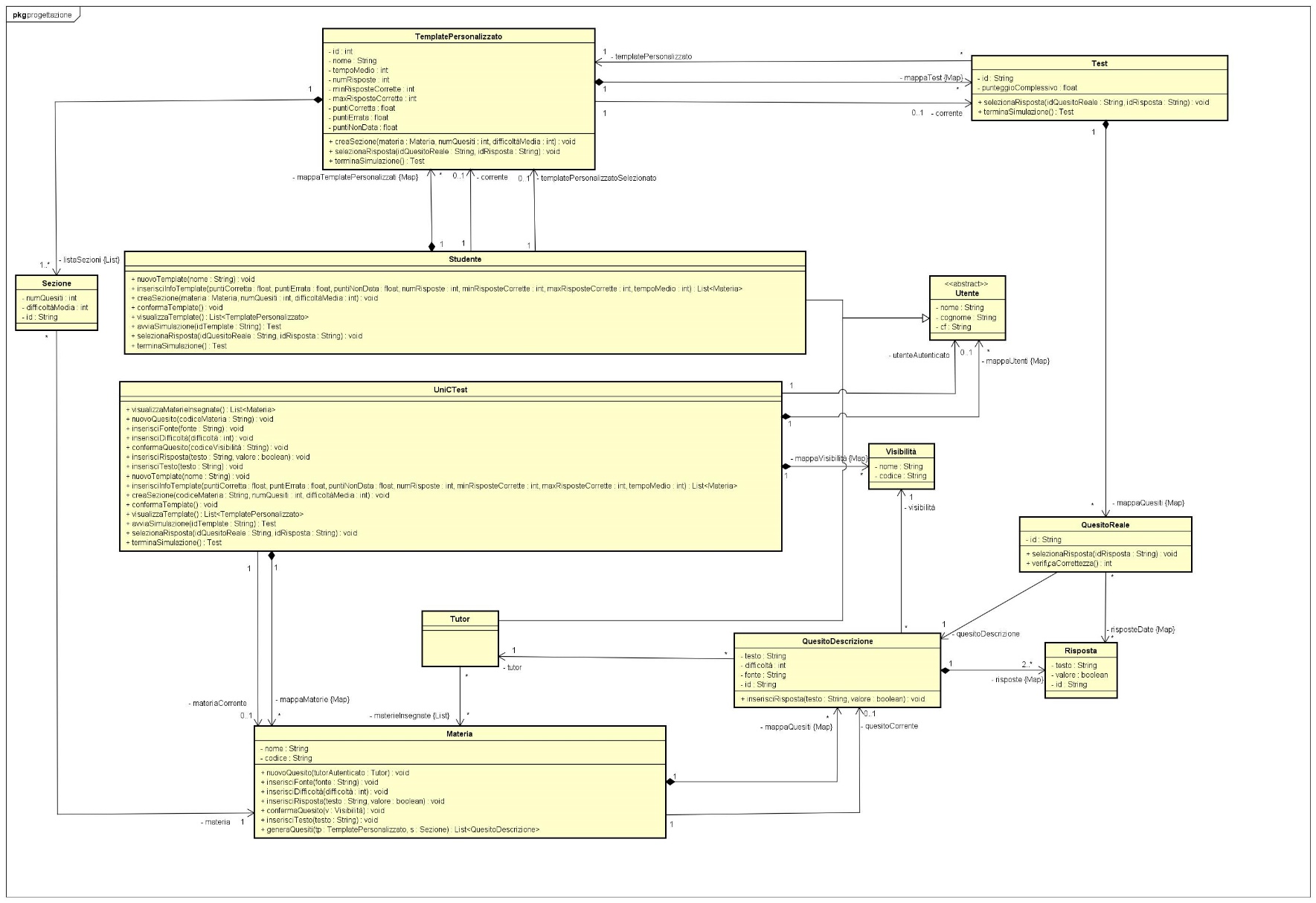
1. **SD\_UC1\_visualizzaTemplate**

****

1. **SD\_UC1\_avviaSimulazione**
2. **SD\_UC1\_selezionaRisposta**
3. **SD\_UC1\_terminaSimulazione**
4. **Cambiamento SD\_UC2.1\_creaTemplate**

L’Iterazione 3 (cioè la corrente) della fase di Elaborazione richiede un cambiamento del diagramma di sequenza: viene creata la mappaTest.

#### DCD



### Implementazione e Testing

#### Implementazione

Per l’Implementazione si rimanda al codice in allegato.

#### Testing

**TestAvviaSimulazione.java**

I test della classe TestAvviaSimulazione si occupano di verificare la creazione di un Test oggetto di simulazione e, in particolare, che lo studente possa interagire con esso per selezionare le risposte.

initTest() impostazione delle condizioni iniziali: recupera l’istanza di UniCTest (Singleton) e autentica uno Studente. Si occupa, inoltre, di inizializzare un TemplatePersonalizzato tramite i metodi già previsti e testati in UC7, in modo da poter essere scelto per la simulazione e di inserire un QuesitoDescrizione da parte di un Tutor.

testVisualizzaTemplate() verifica che la lista di template personalizzati associati allo Studente non sia vuota.

testAvviaSimulazione() si occupa di creare un nuovo Test contenente istanze di QuesitoReale, verificare che l'istanza del TemplatePersonalizzato su cui è basato sia diventata "selezionata" per lo Studente e che il Test sia diventato "corrente" per quel Template.

testSelezionaRisposta() verifica prima che l'inserimento di una risposta nell'istanza di QuesitoReale vada a buon fine, controllando che nella mappa "risposteDate" dell'istanza di QuesitoReale ci sia un'istanza Risposta. Poi verifica che, fornendo come idRisposta lo stesso di prima, la mappa "risposteDate" sia vuota (caso di deselezionamento della risposta).

testTerminaSimulazione() si occupa di verificare che l'istanza di Test venga correttamente restituita allo Studente con l'attributo "punteggioComplessivo" non nullo e che l'istanza di TemplatePersonalizzato "scelta" sia reimpostata a null.

testEccezioniAvviaSimulazione() verifica i possibili casi di errore di avvio di una simulazione, in particolare inizializza un TemplatePersonalizzato con dei valori "irrealistici", come richiedere che per la Sezione di TemplatePersonalizzato inerente a Matematica contenga 800 quesiti (è probabile che non esistano nel Sistema in quella quantità). Dunque l'avvio della simulazione con questo TemplatePersonalizzato dovrà lanciare un'eccezione e l'istanza di Test restituita sarà null. Altra situazione irrealistica: viene inizializzato un TemplatePersonalizzato che richiede 100 risposte per quesito. Non essendoci quesiti con 100 risposte, anche qui l'avvio della simulazione lancerà un'eccezione. In entrambi i casi, viene verificato che l'eccezione venga lanciata per considerare positivo l'esito del test.

## Iterazione IV

### Introduzione e Elaborati

Nella 4° Iterazione della fase di Elaborazione si è scelto di focalizzare l’attenzione su:

* Scenario di successo di UC2/A (introdotto nella seconda revisione della fase di Ideazione): Crea template di test ufficiale.
* Estensione del Caso d’Uso UC2/A.

Si prevede che questa sarà una *breve Iterazione*.

Si creeranno/aggiorneranno i seguenti artefatti:

* Nel contesto dell’Analisi Orientata agli Oggetti:
  + **Modello di Dominio** (da aggiornare)
  + **SSD (System Sequence Diagrams)**
  + **Contratti delle operazioni**
* Nel contesto della Progettazione Orientata agli Oggetti:
  + **Diagrammi di interazione**
  + **DCD (Design Class Diagram)** (da aggiornare)

La 4° Iterazione della fase di Elaborazione prevederà tre brevi fasi di:

1. Analisi orientata agli oggetti (OOA).
2. Progettazione orientata agli oggetti (OOD).
3. Implementazione.

### Analisi Orientata agli Oggetti

#### Modello di dominio

Si riporta lo scenario di successo di UC2/A. Vengono evidenziate le locuzioni nominali per la scelta delle classi concettuali candidate. Le classi concettuali candidate verranno evidenziate in verde, gli attributi in giallo.

**UC2/A. Crea template di test ufficiale**

**Nota del cliente:** l’Amministratore è un Tutor con diritti maggiori rispetto ad un Tutor classico. L’Amministratore, oltre a insegnare materie (dunque inserire nuovi quesiti), può anche inserire nuovi Tutor nel Sistema, oltre che inserire Studenti e template ufficiali. I diritti di Amministratore potranno essere assegnati o rimossi a qualunque Tutor in qualunque momento.

Nel Caso d’Uso e in generale ci si riferirà al Tutor con i diritti di Amministratore semplicemente come Amministratore.

|  |  |
| --- | --- |
| Scenario principale di successo | 1. L’Amministratore vuole creare un nuovo template.  2. L’Amministratore sceglie l’attività “Crea template ufficiale” e inserisce il nome del template ufficiale che deve inserire nel Sistema. Il Sistema registra le informazioni inserite.  3. L’Amministratore inserisce:   * la fonte che ha stabilito le regole del test su cui il template ufficiale deve basarsi * il formato dei quesiti (cioè il numero di risposte e il numero di risposte corrette per quesito) * il punteggio attribuito ad un singolo quesito (nel caso di risposta corretta, risposta errata e risposta non data) * il tempo totale previsto dal test ufficiale.   Il Sistema registra le informazioni inserite.  4. L’Amministratore inserisce il nome della materia che deve essere inserita nel Test. Il Sistema crea una nuova sezione del template relativa alla materia inserita.  5. L’Amministratore seleziona il numero di quesiti previsti per la materia selezionata. Il Sistema registra le informazioni inserite.  *I passi 4, 5 e 6 vengono ripetuti finché serve.*  6. L’Amministratore indica di aver finito. |
| Estensioni | **5a.** L’Amministratore ha inserito nel Sistema una materia non ancora presente in esso. La nuova materia viene aggiunta nel Sistema. |

Le classi concettuali candidate sono:

* **Amministratore**: è un tutor con più diritti rispetto ad un tutor classico (possiede diritti di amministrazione, tra cui inserimento di nuovi template ufficiali).
* **TemplateUfficiale**: contiene informazioni di fonti ufficiali relative ad un test di ammissione.

##### Considerazione #1: Generalizzazione Template

Si può notare che TemplateUfficiale presenta gli stessi attributi di TemplatePersonalizzato (classe concettuale introdotta nell’Iterazione 2 della fase di Elaborazione, analizzando UC2) ad eccezione di:

* **fonte**, attributo inesistente in TemplatePersonalizzato, essendo il TemplatePersonalizzato creato dallo stesso Studente
* **tempoTotale**, indica il tempo totale di svolgimento del test. Viene introdotto al posto del tempoMedio del TemplatePersonalizzato, che indicava il tempo medio di svolgimento di un solo quesito.

Risulta dunque chiaro che non sia necessario introdurre una nuova classe TemplateUfficiale ex-novo, in quanto condivide la maggior parte delle informazioni del TemplatePersonalizzato. Dunque, si è deciso di rappresentare i due template come **specializzazione** di una **classe concettuale astratta** ***Template***.

La classe astratta *Template* avrà tutti gli attributi ad eccezione degli attributi non condivisi tra il TemplatePersonalizzato e il TemplateUfficiale (ovvero: fonte, tempoTotale, tempoMedio), che saranno specifici delle classi specializzazione.

##### Considerazione #2: Applicazione del pattern GoF strutturale Decorator

Alla luce della nota contenuta in UC2/A, bisogna concludere che l’Amministratore non sia una classe candidata nuova, bensì possa essere vista come una responsabilità addizionale di un tutor classico. Ciò è testimoniato dal fatto che questa responsabilità può essere assegnata o rimossa a *runtime*.

Non si può non considerare che una nota simile sia presente in riferimento al Tutor di simulazione nella seconda revisione della fase di Ideazione.

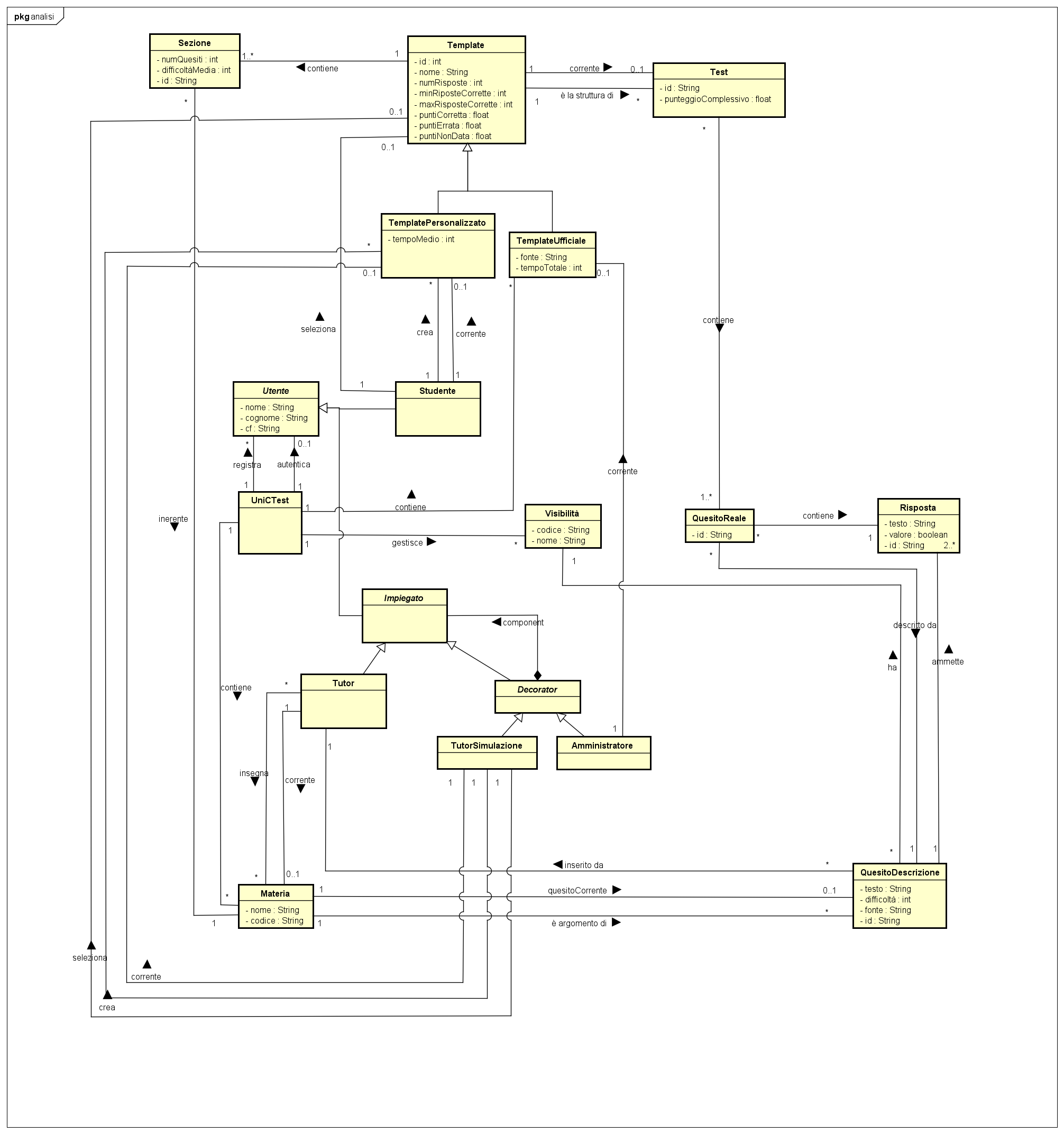
Quindi, si può concludere che il Tutor può essere *decorato* con responsabilità addizionali (ad esempio responsabilità di Amministratore – che verrà introdotta in questa sede – oppure altre responsabilità, come quella di Tutor di simulazione – che verranno introdotte in iterazioni successive).

Ciò suggerisce l’applicazione del **pattern GoF Decorator**.

Nella fase di Progettazione verrà esplorata meglio la soluzione offerta dal pattern.

##### Modello di Dominio

Dalle classi concettuali che sono state ricavate e dalle considerazioni appena effettuate è stato ricavato il seguente Modello di Dominio, comprendente anche le classi concettuali ricavate nelle fasi di OOA relative alle precedenti iterazioni:



#### SSD

#### Contratti delle Operazioni

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC2/A.

Nota: per distinguere le operazioni di gestione del TemplateUfficiale da quelle di gestione del TemplatePersonalizzato da parte dello Studente, sono state prese le seguenti misure:

* le operazioni effettuate dall’attore Amministratore per inserire un TemplateUfficiale rispetteranno la notazione *<nomeOperazione>U*.
* le operazioni effettuate dall’attore Studente per creare un TemplatePersonalizzato (in relazione a UC2, Iterazione 2 della fase di Elaborazione) sono state rinominate in *<nomeOperazione>P*.

### CO1

|  |  |
| --- | --- |
| Operazione | nuovoTemplateU(nome: String) |
| Riferimenti | UC2/A: Crea template di test ufficiale |
| Pre-condizioni | - |
| Post-condizioni | * È stata creata l’istanza *tu* di TemplateUfficiale con *tu.nome*=nome ed è stata associata all’Amministratore tramite l’associazione “corrente”. |

### CO2

|  |  |
| --- | --- |
| Operazione | inserisciInfoTemplateU(fonte: String, puntiCorretta: float, puntiErrata: float, puntiNonData: float, tempoTotale: int, numRisposte: int, minRisposteCorrette: int, maxRisposteCorrette: int) |
| Riferimenti | UC2/A: Crea template di test ufficiale |
| Pre-condizioni | È in corso la creazione di un TemplateUfficiale *tu*. |
| Post-condizioni | * Sono stati inizializzati gli attributi di *tu*. |

### CO3

|  |  |
| --- | --- |
| Operazione | creaSezioneU(nomeMateria: String, numQuesiti int) |
| Riferimenti | UC2/A: Crea template di test ufficiale |
| Pre-condizioni | È in corso la creazione di un TemplateUfficiale *tu*. |
| Post-condizioni | * È stata inizializzata l’istanza *s* di Sezione ed è stata associata a *tu* tramite l’associazione “contiene”. * L’istanza *s* è stata associata all’istanza *m* di Materia avente l’attributo *m.codice* uguale a codiceMateria. |

### CO4

|  |  |
| --- | --- |
| Operazione | confermaTemplateU() |
| Riferimenti | UC2/A: Crea template di test ufficiale |
| Pre-condizioni | È in corso la creazione di un TemplateUfficiale *tu*. |
| Post-condizioni | * L’istanza *tu* è stata associata all’istanza *a* dell’Amministratore tramite l’associazione “crea”. |

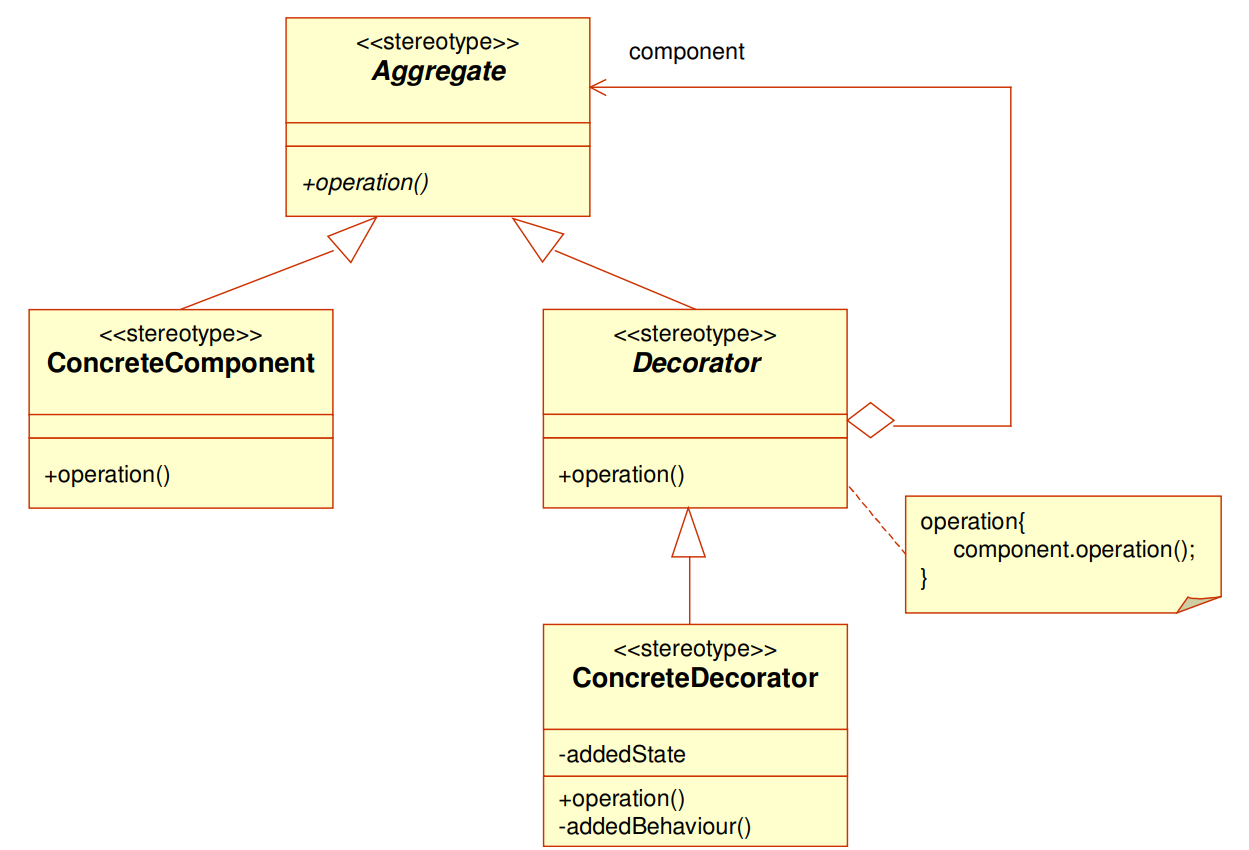
### Progettazione Orientata agli Oggetti

Aggiornate le classi concettuali nel contesto della OOA, si passa alla OOD, realizzando il Modello di Progetto ed in particolare aggiornando il DCD (visualizzazione statica delle classi software), aggiornamento da effettuare il parallelo alla realizzazione dei diagrammi di interazione (visualizzazione dinamica delle classi software).

L’Analisi Orientata agli Oggetti condotta in questa Iterazione ha portato all’inserimento del pattern Decorator, che qui viene meglio illustrato.

#### La classe Amministratore: Applicazione del pattern GoF Decorator

Struttura offerta dal pattern:

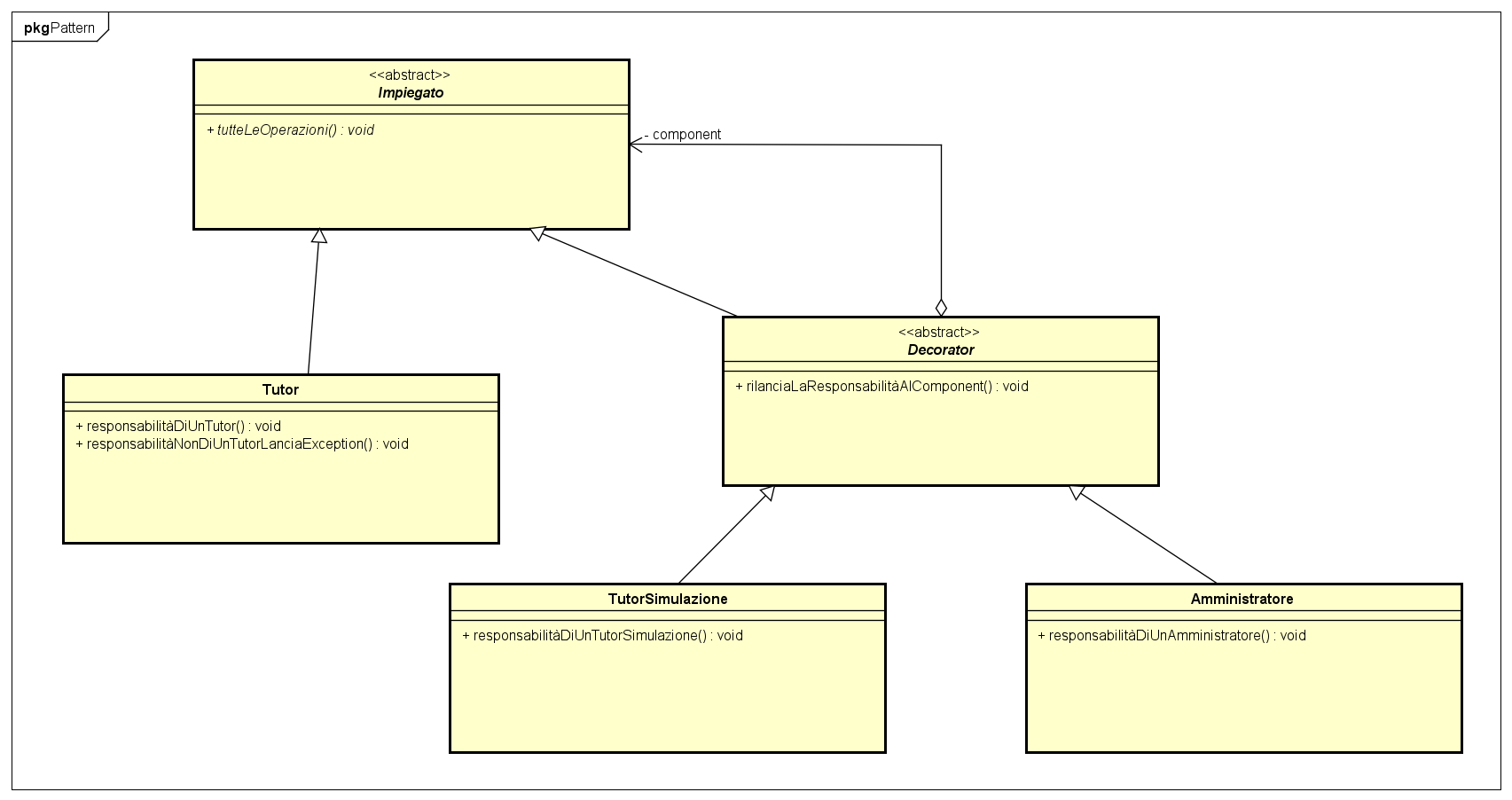


La struttura (nel nostro caso) prevede l’impiego di:

* Classe astratta **Aggregate**: risponde all’esigenza di avere un’interfaccia comune per gli oggetti eventualmente decorati di responsabilità aggiuntive.
* Classe **ConcreteComponent**: implementa l’oggetto a cui possono essere aggiunte o rimosse delle responsabilità.
* Classe astratta **Decorator**: risponde all’esigenza di avere un’interfaccia comune per i decoratori. Un decorator ha il riferimento ad un Aggregate, che a sua volta può essere un Decorator oppure un ConcreteComponent.
* Classe **ConcreteDecorator**: è la classe che concretamente aggiunge le responsabilità al ConcreteComponent.

Nel contesto di UniCTest:

* La classe **Aggregate** è la classe **Impiegato**.
* La classe **ConcreteComponent** è la classe Tutor. Infatti, il Tutor è la classe da arricchire di responsabilità addizionali.
* La classe **Decorator** mantiene invariato il nome. Decorator generalizza Amministratore e si prevede che nelle prossime iterazioni generalizzerà anche TutorSimulazione.
* La classe **ConcreteDecorator** è la classe **Amministratore** (e si prevede che un altro ConcreteDecorator sarà la classe TutorSimulazione).

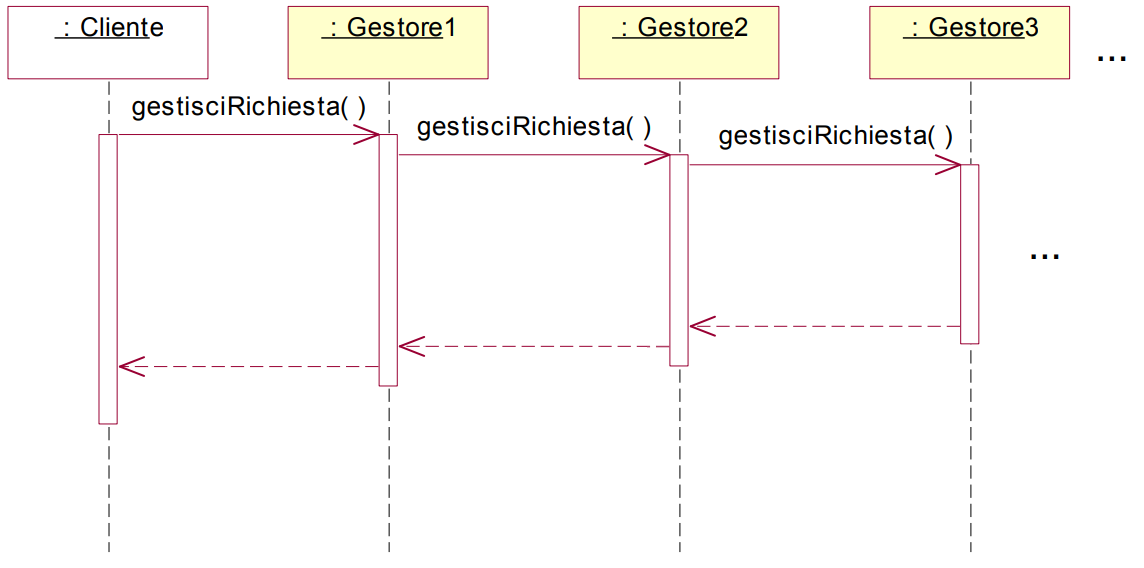


La struttura del pattern, così come si evince dalla figura, permette di avere un Tutor che gode di più responsabilità addizionali assegnate e/o rimosse a *runtime*.

È stato elaborato un meccanismo che permette ad un Impiegato decorato di soddisfare qualunque responsabilità gli è concessa per via di qualunque sua decorazione.

Il meccanismo elaborato sfrutta il pattern GoF comportamentale **Chain of Responsibility**.

Il pattern Chain of Responsibility consente di applicare questo principio: se un gestore riesce a soddisfare una richiesta, allora la esegue, altrimenti passa la richiesta al gestore successivo.



In questo caso con “gestore successivo” si intende un gestore “più interno”, cioè il *component* di un Decorator.

Di seguito viene spiegato tale meccanismo considerando le due tipologie di scenari che possono verificarsi.

#### Scenario 1

Si supponga l’esistenza di un’istanza *t1* = TutorSimulazione(Amministratore(Tutor))), ovvero un Tutor che è stato dapprima decorato con responsabilità di Amministratore, poi anche con responsabilità di TutorSimulazione. Qualora *t1* decida di eseguire una responsabilitàDiUnAmministratore(), allora verrebbe chiamata su un’istanza di TutorSimulazione che, non avendo implementato il metodo, chiamerebbe il metodo del proprio Decoratore. Tale metodo del Decoratore rilancerebbe la responsabilità al component più interno, cioè ad Amministratore(Tutor) che, essendo una istanza di Amministratore, è in grado di soddisfare la responsabilitàDiUnAmministratore() senza rilanciarla (infatti implementa tale metodo senza richiamarlo su un component più interno).

Quindi in tal caso la responsabilità è stata correttamente assegnata e soddisfatta.

#### Scenario 2

Si supponga l’esistenza di un’istanza *t2* = TutorSimulazione(Tutor)), ovvero un Tutor che è stato decorato con responsabilità di TutorSimulazione ma non di Amministratore. Qualora *t2* decida di eseguire una responsabilitàDiUnAmministratore(), allora verrebbe chiamata su un’istanza di TutorSimulazione che, non avendo implementato il metodo, chiamerebbe il metodo del proprio Decoratore. Tale metodo del Decoratore rilancerebbe la responsabilità al component più interno, cioè a Tutor che, essendo un Tutor semplice e non decorato, non può soddisfare alcuna responsabilità addizionale. Quindi il Tutor può lanciare un’eccezione (dichiarando, dunque, di non aver soddisfatto la responsabilità perché *t2* non ne aveva il diritto).

Quindi in tal caso la responsabilità è stata correttamente non soddisfatta.

#### Estensioni: Inserimento di una Materia nel Sistema e applicazione del pattern GoF Flyweight

Dal Caso d’Uso si evince che il Sistema non deve restituire all’Amministratore le materie disponibili, pertanto è possibile che l’Amministratore digiti una materia non ancora presente nel Sistema (magari perché non è ancora stato inserito alcun Tutor che la insegna).

L’inserimento di un template ufficiale può dunque causare l’inserimento di una nuova materia nel sistema.

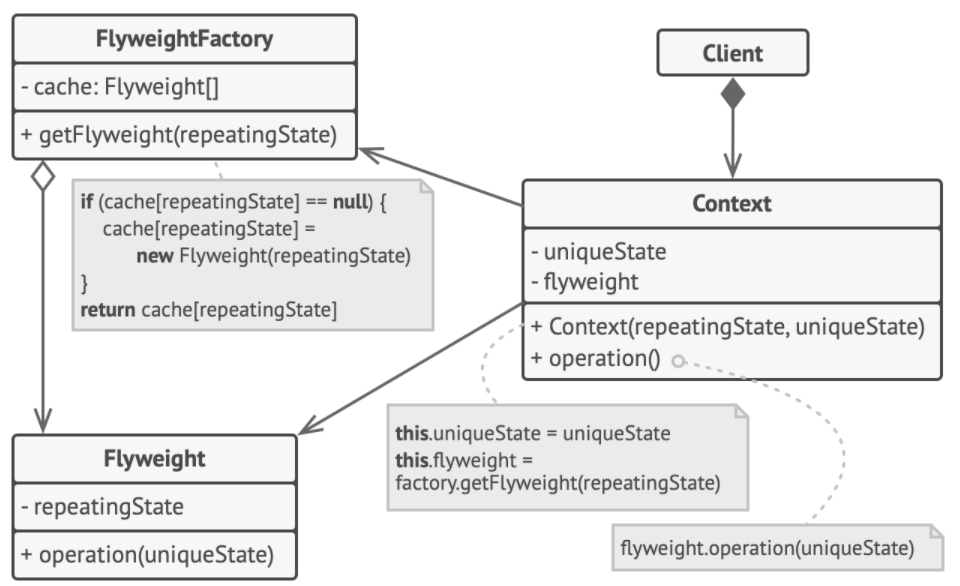
Tuttavia, se una materia è già presente nel Sistema, allora l’istanza di Materia da adoperare è quella già presente nel Sistema. Infatti, non avrebbe senso creare una nuova istanza di Materia, anche perché le materie sono deputate al contenimento dei quesiti e non si vuole che quesiti di uno stessa Materia risultino “sparsi” su varie liste di varie istanze della classe Materia.

Il pattern Flyweight è un pattern GoF strutturale mirato ad ottimizzare la memoria.

Di seguito viene analizzata la soluzione proposta dal pattern e viene analizzato perché tale soluzione possa risultare utile in questo contesto.

##### Struttura offerta dal pattern

Il pattern prevede la seguente struttura:

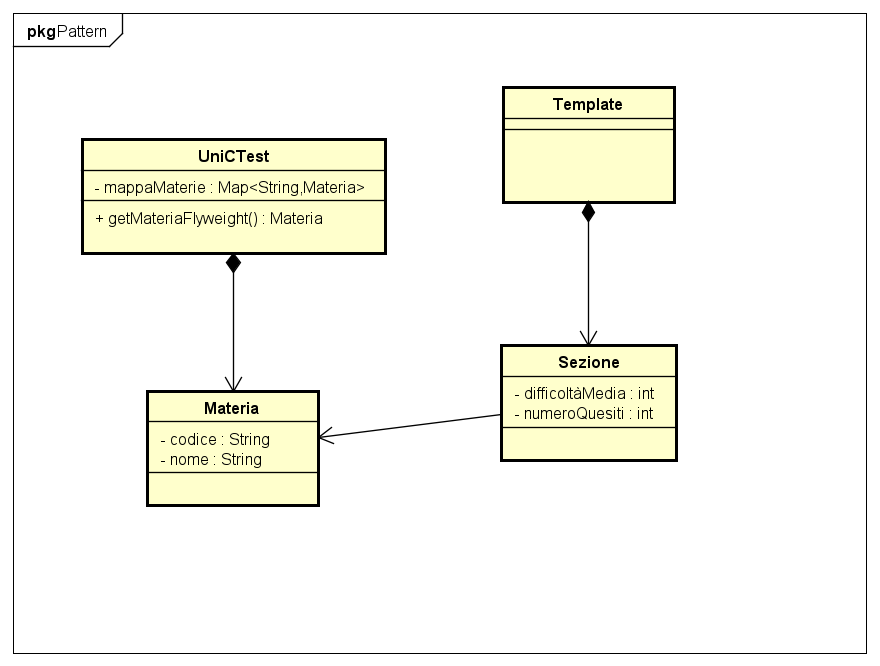


La struttura prevede l’impiego di:

* Classe **Flyweight**: contiene la porzione dell’oggetto originale che può essere condivisa tra più oggetti. Lo stesso oggetto flyweight può essere usato in diversi contesti. Il Flyweight contiene una parte dello stato dell’oggetto originale, detta parte ***intrinseca***, cioè quella parte che può essere condivisa perché è effettivamente comune a più oggetti. Lo stato dell’oggetto originale, non condiviso da vari oggetti, viene esternato dal flyweight e per questo è detto stato***estrinseco***. Tale stato viene aggiunto nella classe Context.
* Classe **Context**: contiene lo stato estrinseco dell’oggetto originale. Se la classe Context fosse un tutt’uno con la classe Flyweight, il pattern godrebbe di scarsa applicabilità in quanto non consentirebbe un notevole risparmio di memoria.
* Classe **Client**: in genere calcola lo stato estrinseco del flyweight. Dalla prospettiva del Client, il flyweight è un oggetto che può essere configurato a *runtime* a partire da qualche parametro di contesto.
* Classe **FlyweightFactory**: si occupa di creare, conservare, gestire e restituire gli oggetti flyweight.

Nel contesto di UniCTest:

* La classe **Flyweight** è la classe **Materia**.
* La classe **Context** è la classe **Sezione**. Infatti, le Sezioni possiedono argomenti con una variabilità troppo ampia per poter essere considerate parte dello stato *intrinseco*.
* La classe **Client** è la classe **Template**. Il Template si occupa di effettuare lo storage e la gestione degli stati *estrinseci* della Materia, cioè delle Sezioni.
* La classe **FlyweightFactory** è la classe **UniCTest**. UniCTest crea, memorizza, gestisce e restituisce gli oggetti flyweight in maniera opportuna, cioè creandoli solo se essi non sono presenti nel Sistema.



Da notare che la creazione dell’oggetto flyweight si origina dalla richiesta di creazione di una nuova Sezione (Context) da parte del Template (Client), in particolare all’interno del costruttore della Sezione (Context).

Inoltre, l’associazione tra Context e FlyweightFactory, in questo specifico caso, non è necessaria, in quanto la FlyweightFactory è un **Singleton** (infatti UniCTest rappresenta l’intero Sistema software), dunque la sua istanza può essere recuperata mediante la chiamata al metodo statico *getInstance()*.

**Test-driven development**

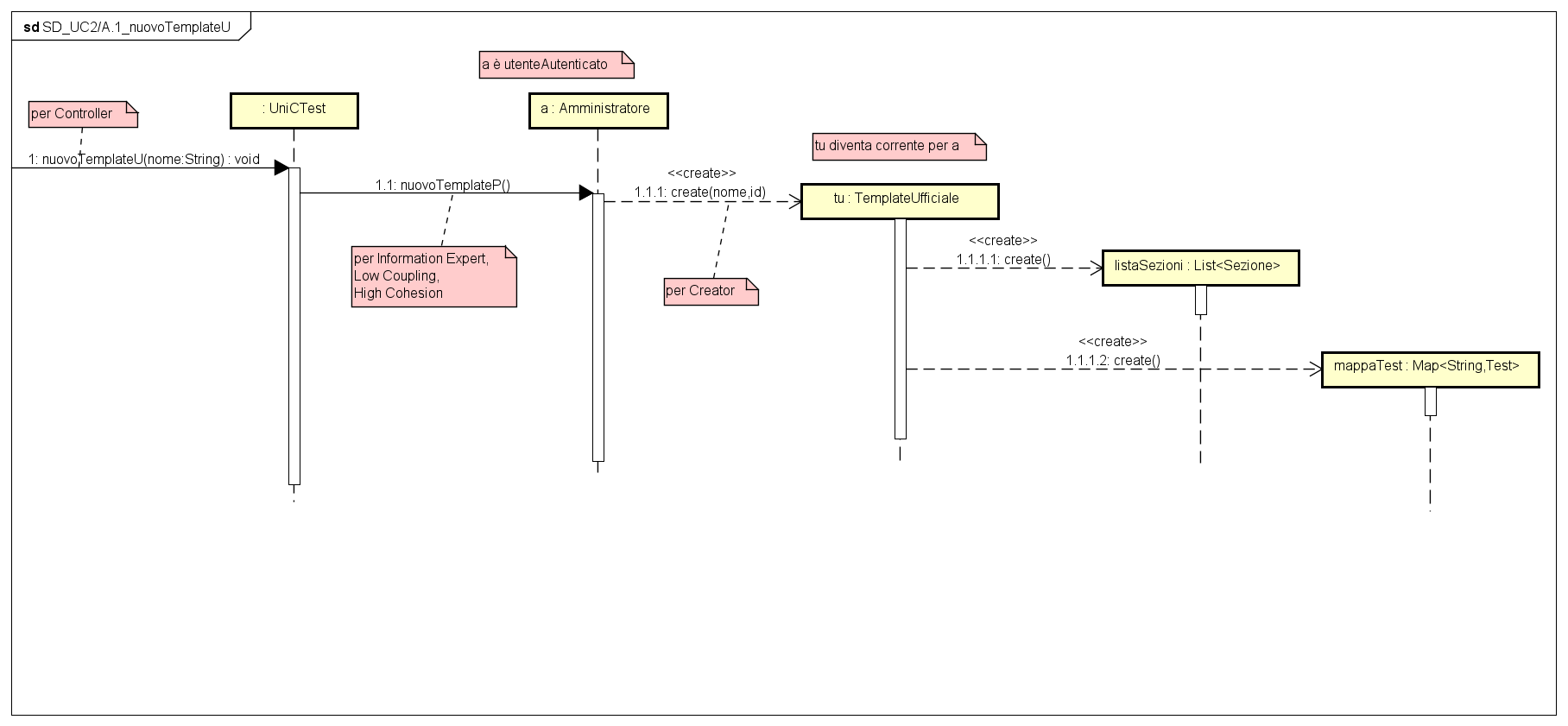
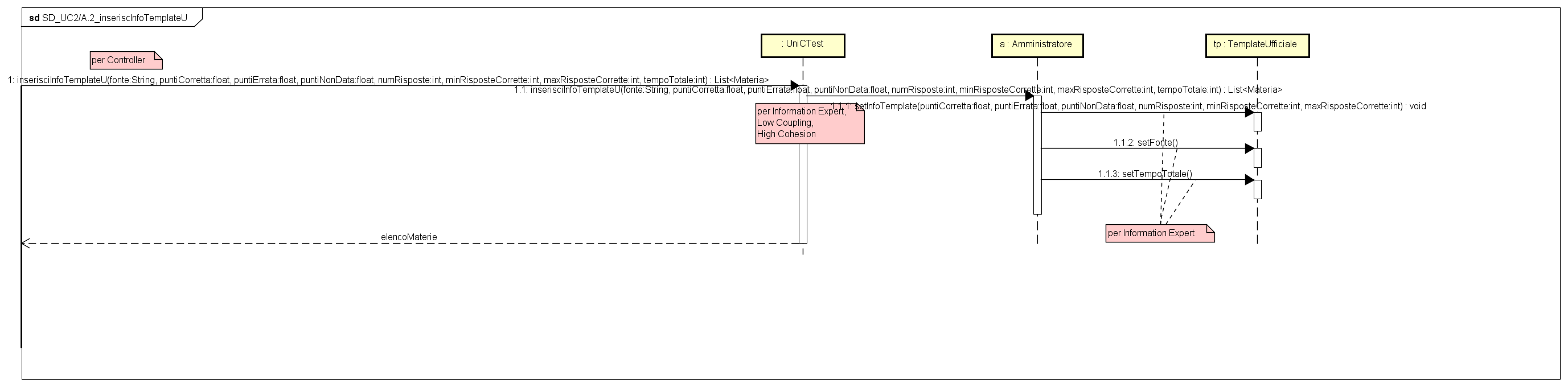
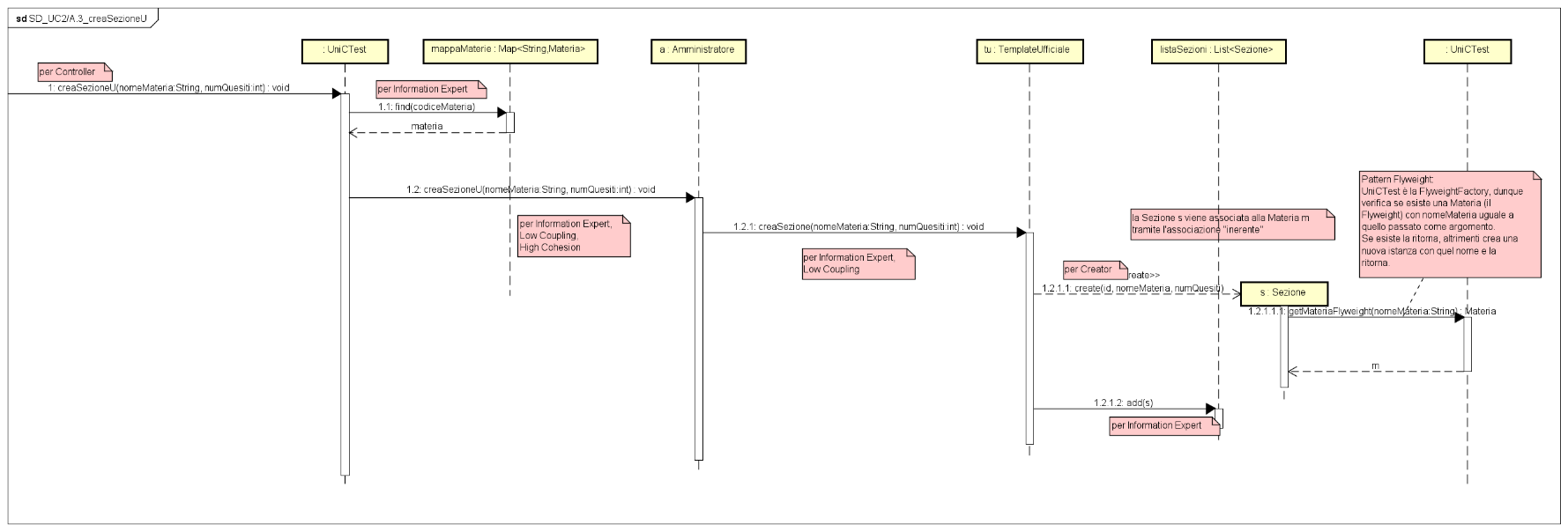
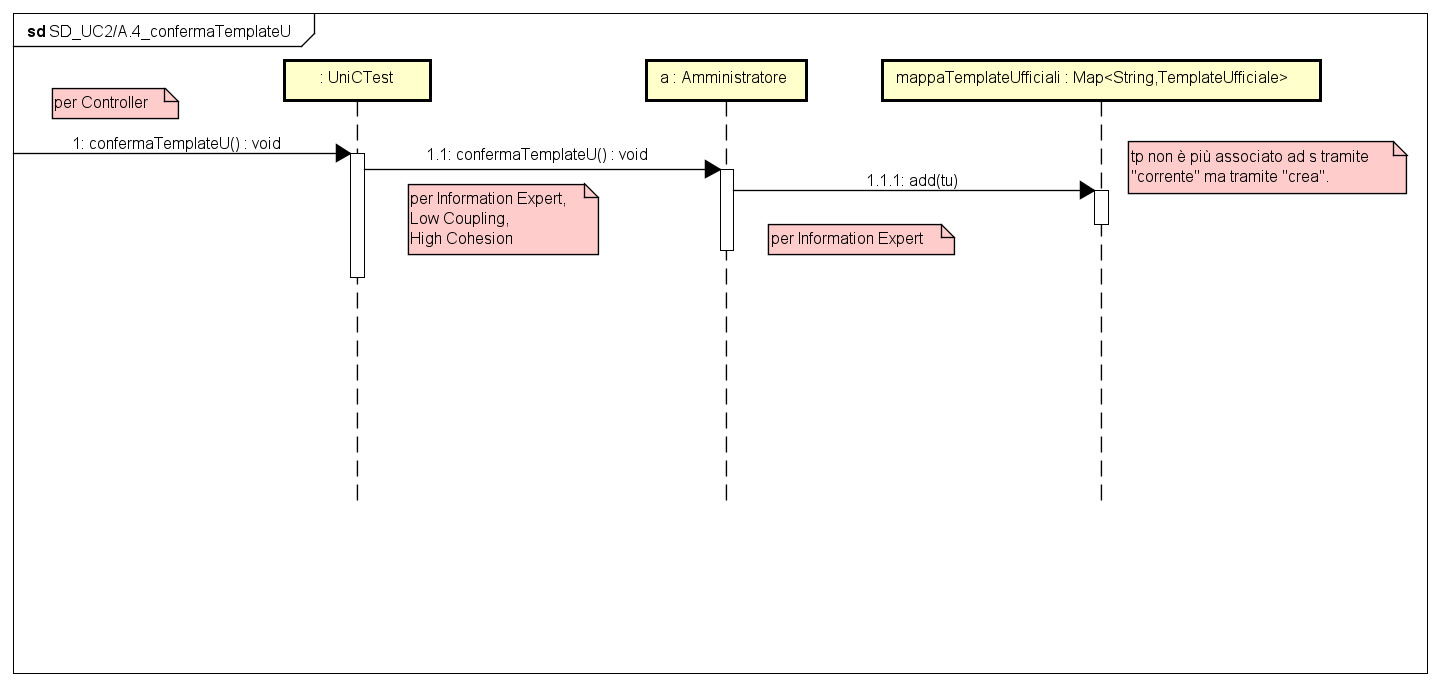
Nell’Iterazione corrente è stato deciso, per il momento, di non implementare i metodi per l’aggiunta a runtime delle responsabilità dei Tutor, al fine di non sovraccaricare i compiti dell’Iterazione corrente, rischiando di renderla eccessivamente lunga.

Tuttavia, è stato deciso che i test da effettuare dovranno essere in questa forma (pseudocodice):

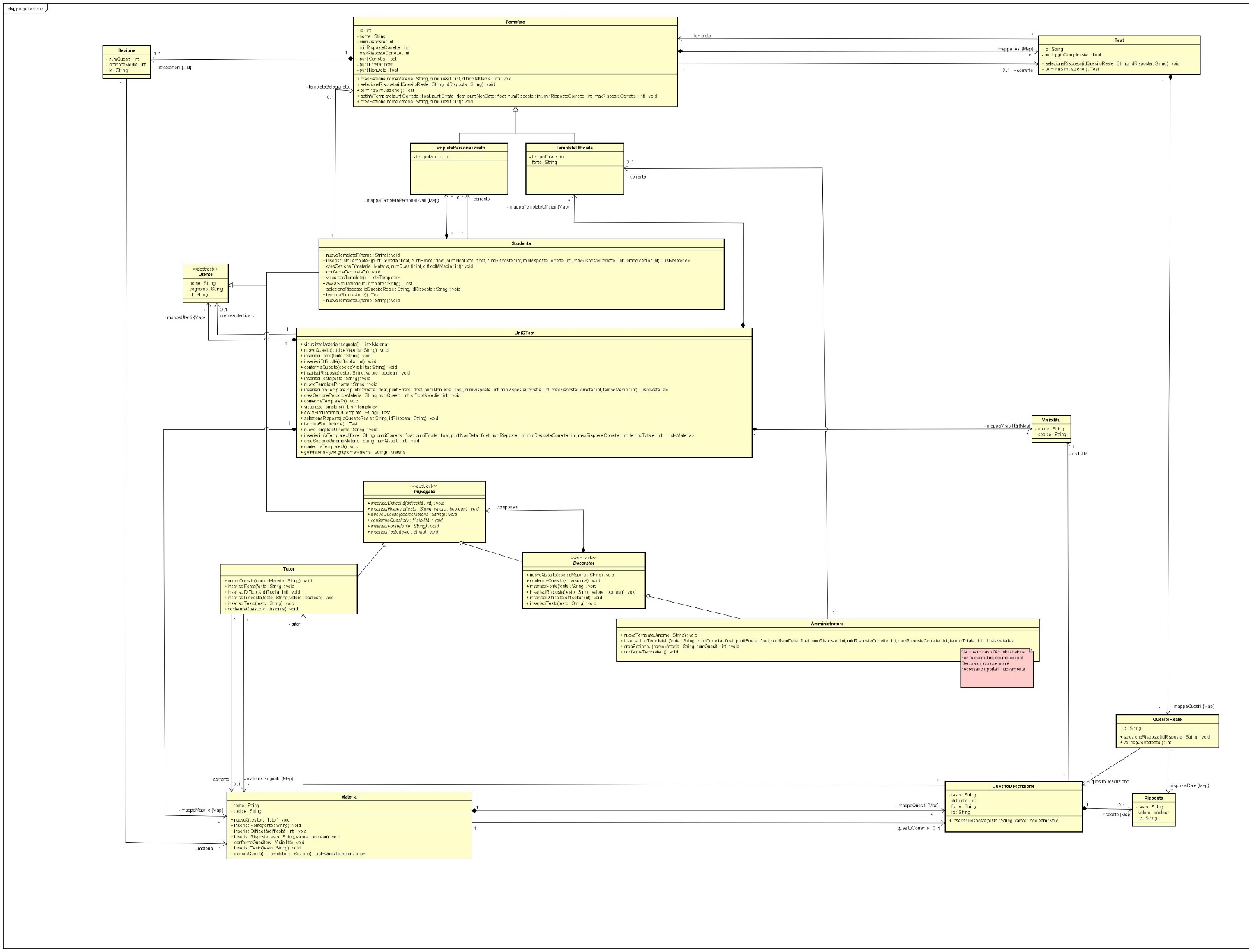
@Test  
void aggiungiAmministratore(){  
 *amministratore*.rendiAmministratore(*tutorDaDecorare*);  
 *assertTrue*(*tutorDaDecorare*.isAmministratore());  
}

La scelta di elaborare prima un test in pseudo-codice aiuterà lo sviluppo dei metodi di aggiunta delle responsabilità e di verifica della corretta aggiunta delle responsabilità.

#### Diagrammi di interazione

1. **SD\_UC2/A\_creaTemplateU**
2. **SD\_UC2/A\_inserisciInfoTemplateU**
3. **SD\_UC2/A\_creaSezioneU**
4. **SD\_UC2/A\_confermaTemplateU**

#### DCD



## Iterazione V

### Introduzione

Implementati i Casi d’Uso riguardanti le simulazioni online, queste ultime curate direttamente dal singolo studente e mirate a rafforzare la preparazione in maniera individuale, si passa ad analizzare “l’altra faccia della medaglia”, cioè le simulazioni cartacee, area di notevole riguardo perché da usare per rafforzare la preparazione collettiva degli studenti.

Durante i *workshop* destinati alla stesura del *planning dei costi*, è stato notato che lo sviluppo di quest’area del progetto prevede anche dei costi hardware, derivati dall’acquisto di strumenti di lettura dei fogli risposte degli studenti. Tali strumenti possono essere lettori ottici o semplici scanner. Dato il budget del cliente, è stato deciso di gestire la lettura del codice risposte tramite scanner. Questo influisce sulla struttura stessa del foglio risposte. Tuttavia, il cliente ha espresso di essere aperto, in futuro, a modifiche sul meccanismo di lettura dei fogli risposte qualora altri metodi possano velocizzare il meccanismo di lettura, in previsione di un numero elevato di iscrizioni di studenti.

Nella Iterazione 4 della fase di Elaborazione ci si occupa di introdurre le classi concettuali derivate dall’esplorazione dei Casi d’Uso UC9 e UC10 (comprensivi di Estensioni), entrambi concernenti le simulazioni cartacee.

Si è scelto di realizzare i seguenti artefatti:

* Nel contesto dell’Analisi Orientata agli Oggetti:
  + **Modello di Dominio** (da aggiornare)
  + **SSD (System Sequence Diagrams)**
  + **Contratti delle operazioni**
* Nel contesto della Progettazione Orientata agli Oggetti:
  + **Diagrammi di interazione**
  + **DCD (Design Class Diagram)** (da aggiornare)

La 5° Iterazione della fase di Elaborazione prevederà diversi step:

1. Analisi orientata agli oggetti (OOA). Bisognerà aggiornare il Modello di Dominio. Le nuove classi concettuali da introdurre saranno individuate tra le locuzioni nominali presenti in UC9 e UC10.
2. Progettazione orientata agli oggetti (OOD). Il Modello di Progetto costruito nelle iterazioni precedenti della fase di Elaborazione andrà aggiornato.
3. Implementazione e testing. Bisognerà implementare le classi software, i metodi, l’interfaccia utente e i test relativi a UC9 e UC10.

### Modello di Analisi

#### Aggiornamento Modello di Dominio

Si prendono in considerazione gli scenari di successo di UC9 e UC10.

Vengono evidenziate le locuzioni nominali per la scelta delle classi concettuali candidate. Le classi concettuali candidate verranno evidenziate in verde, gli attributi in giallo.

**UC9. Componi test per simulazione cartacea**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC9: Componi test per simulazione cartacea |
| Scenario principale di successo | 1. Il Tutor vuole creare un nuovo test cartaceo  2. Il Tutor sceglie la voce “nuovo test cartaceo”.  3. Il Tutor sceglie il template su cui basare il test.  4. Il Sistema mostra al Tutor i quesiti disponibili per l’inserimento nel test relativi ad una materia del template.  5. Il Tutor seleziona i quesiti da inserire relativi alla materia indicata.  *I passi 4 e 5 sono ripetuti fin quando terminano le materie del template.*  6. Il Sistema registra le informazioni inserite e restituisce al Tutor il test e il modulo per le risposte in formato PDF. |

Le classi concettuali candidate sono:

* **TutorSimulazione**: tutor autorizzato alla realizzazione di test cartacei;
* **ModuloRisposte**: è un documento che contiene le informazioni sul test. Questo documento sarà poi compilato dallo studente (il quale segnerà le risposte da lui ritenute esatte) e restituito al tutor per la correzione.

Bisogna notare però che il tutor di simulazione non è una classe candidata nuova, bensì può essere vista come una responsabilità addizionale di un tutor classico. Ciò è testimoniato dal fatto che questa responsabilità può essere assegnata o rimossa a runtime.

Ciò suggerisce l’applicazione del pattern GoF **Decorator**.

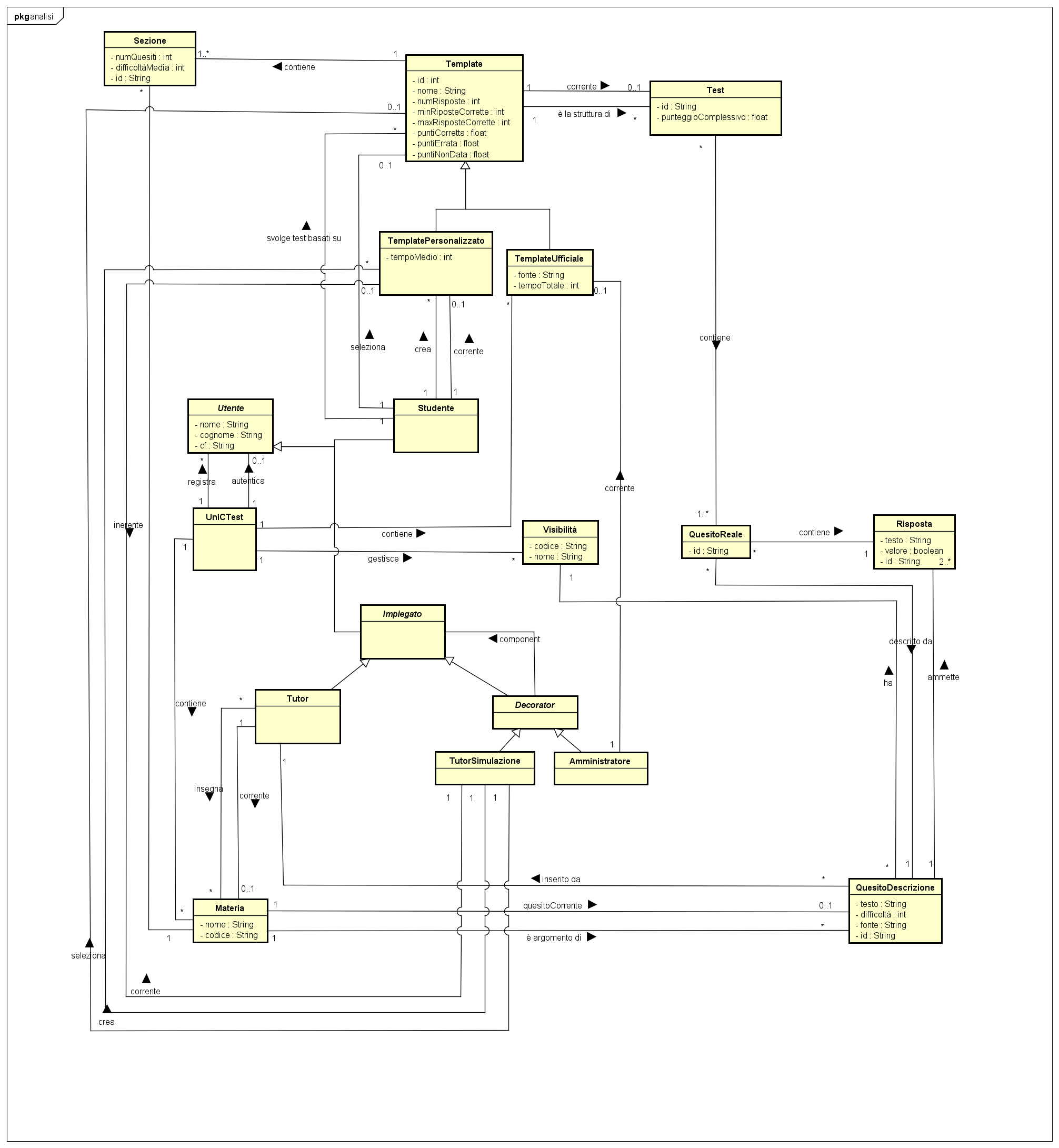
Ciò è stato applicato nella Iterazione 4 della fase di Elaborazione nell’introduzione della classe concettuale Amministratore e, nella Iterazione corrente, si concretizza anche per la classe concettuale TutorSimulazione.

**UC10. Correggi simulazione cartacea**

|  |  |
| --- | --- |
| Nome del Caso d’Uso | UC10: Correggi simulazione cartacea |
| Scenario principale di successo | 1. Il Tutor vuole correggere il test cartaceo di uno studente.  2. Il Tutor sceglie la voce “correggi test cartaceo”.  3. Il Sistema chiede al tutor di inserire il modulo risposte inerente al test da correggere.  4. Il Sistema legge le informazioni del foglio risposte: l’id del test a cui fa riferimento, il codice fiscale dello Studente al quale registrare l’esito e il codice fiscale del Tutor che ha creato il Test e le restituisce al Tutor.  5. Il Tutor conferma le informazioni lette.  6. Il Sistema legge le risposte date dallo Studente dal foglio risposte, compila il test e lo restituisce al Tutor.  7. Il Tutor conferma le informazioni lette.  8. Il Sistema salva il test corretto e registra l’esito allo Studente. |
| Estensioni | **\*a.** In un qualsiasi momento il Sistema fallisce e si arresta improvvisamente.   1. Il Tutor riavvia il software. 2. Il Sistema ripristina lo stato.   **5a, 7a.** Il Tutor si rende conto di errori di lettura e corregge manualmente le informazioni. |
| Requisiti speciali | Il Sistema deve offrire un’interfaccia user-friendly. |
| Elenco delle varianti tecnologiche e dei dati |  |
| Frequenza di ripetizione | Alta. |
| Varie | **Nota del cliente:** deve essere possibile per un Tutor di simulazione correggere le simulazioni dei colleghi. |

Quindi il modulo risposte deve contenere l’id del test, il codice dello studente a cui fa riferimento, l’id delle risposte fornite dallo studente.

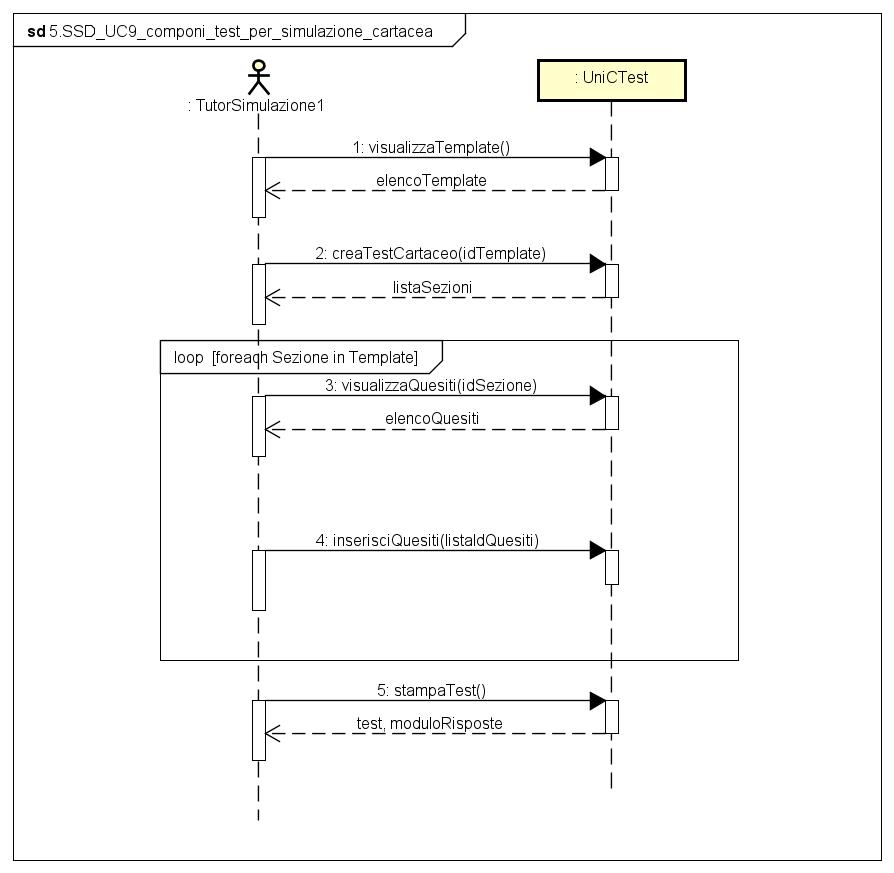
Il test contiene già l’esito, che è stato indicato come punteggioComplessivo nelle iterazioni precedenti.

****Viene di seguito riportato il Modello di Dominio aggiornato alla luce delle recenti considerazioni:

### Analisi e Progettazione UC9

#### Analisi Orientata agli Oggetti UC9

##### Diagramma di Sequenza di Sistema UC9

Una volta aggiornato il Modello di Dominio, viene creato il SSD relativo a UC9, che consente di visualizzare in forma grafica l’interazione tra il TutorSimulazione ed il Sistema.

##### Contratti delle Operazioni UC9

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC9.

### CO1

|  |  |
| --- | --- |
| Operazione | visualizzaTemplate() |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | - |
| Post-condizioni | * È stato restituito l’elenco delle istanze *te* di Template al TutorSimulazione *utenteAutenticato*. |

### CO2

|  |  |
| --- | --- |
| Operazione | creaTestCartaceo(idTemplate) |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | - |
| Post-condizioni | * È stata creata l’istanza *t* di Test ed è stata associata all’istanza *te* di Template avente *te.id*=idTemplate tramite l’associazione “corrente”. |

### CO3

|  |  |
| --- | --- |
| Operazione | visualizzaQuesiti(idSezione) |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | È in corso la creazione di un test *t* per una simulazione cartacea. |
| Post-condizioni | * È stato restituito al TutorSimulazione *utenteAutenticato* l’elenco delle istanze *qd* di QuesitoDescrizione associate all’istanza *m* di Materia associata all’istanza *s* di Sezione avente *s*.id uguale a idSezione. |

### CO4

|  |  |
| --- | --- |
| Operazione | inserisciQuesiti(listaIdQuesiti) |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | È in corso la creazione di un test *t* per una simulazione cartacea. |
| Post-condizioni | * Sono state create le istanza *qr* di QuesitoReale, che sono state associata al Test *t* tramite l’associazione “contiene” e sono state associate alle istanze *qd* aventi *qd*.id uguali a quelli in listaIdQuesiti tramite l’associazione “descritto da”. |

### CO5

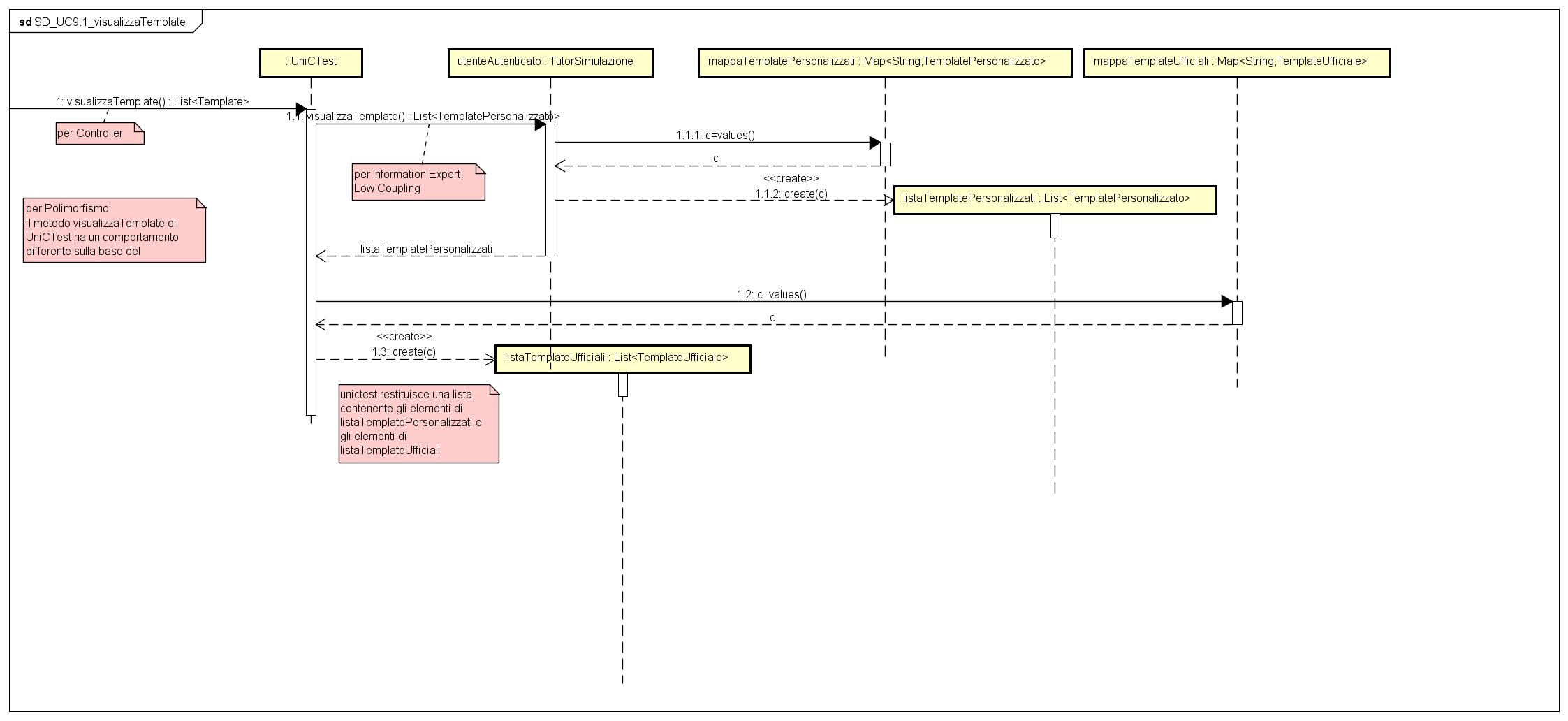
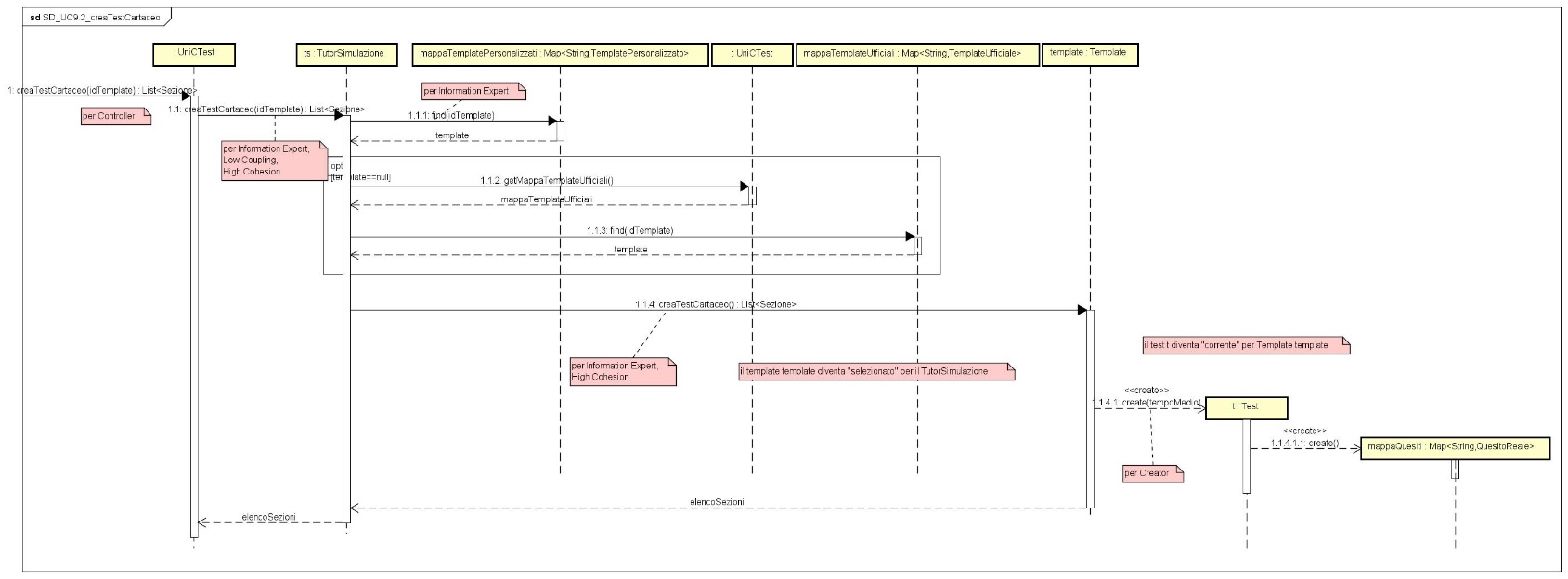
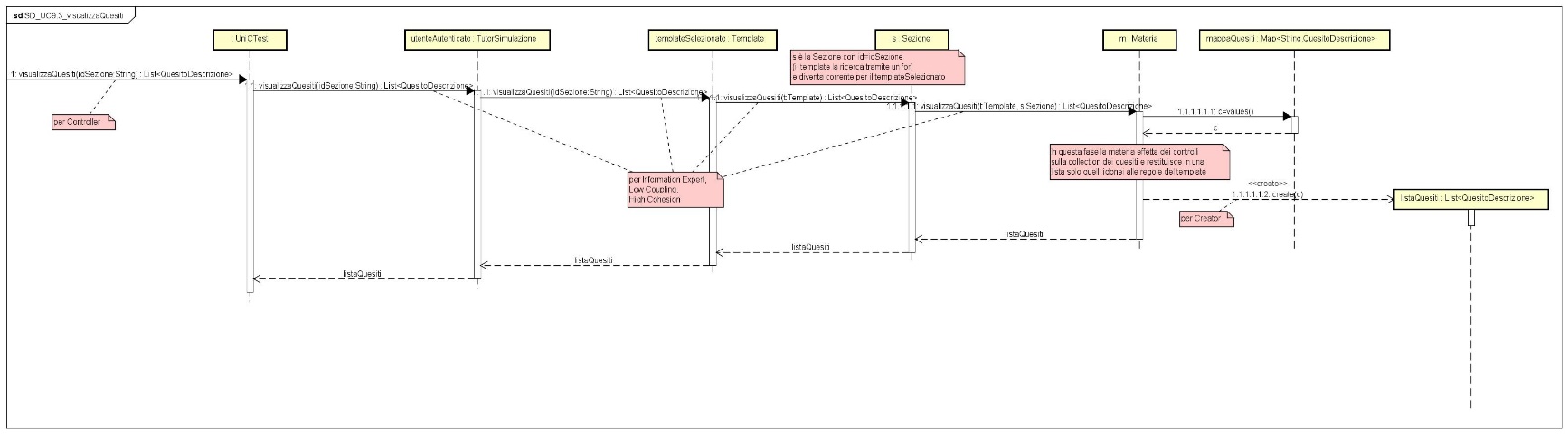
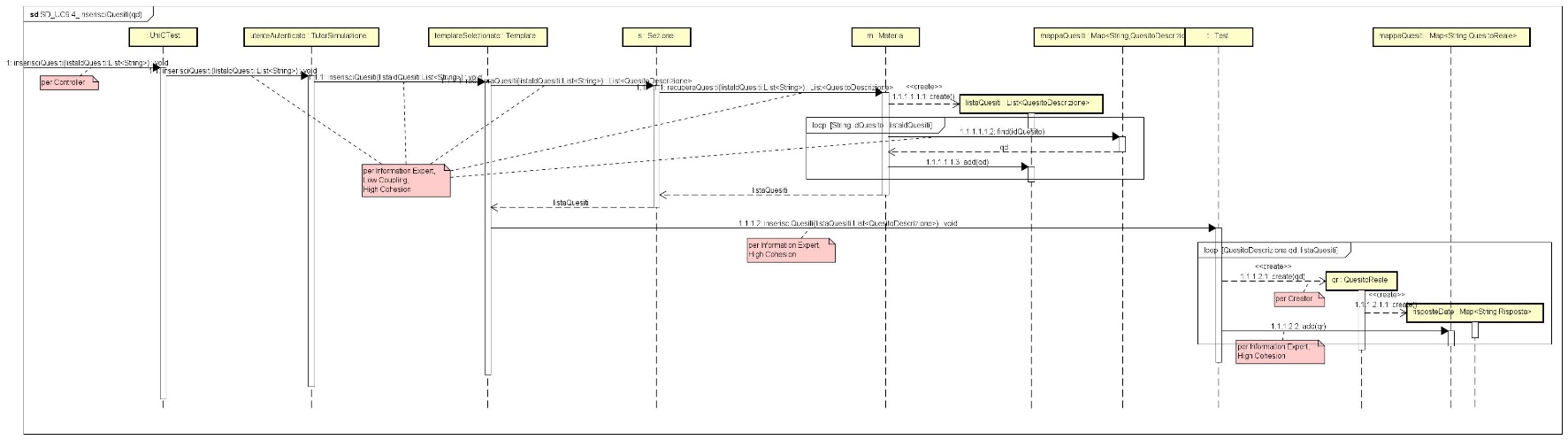
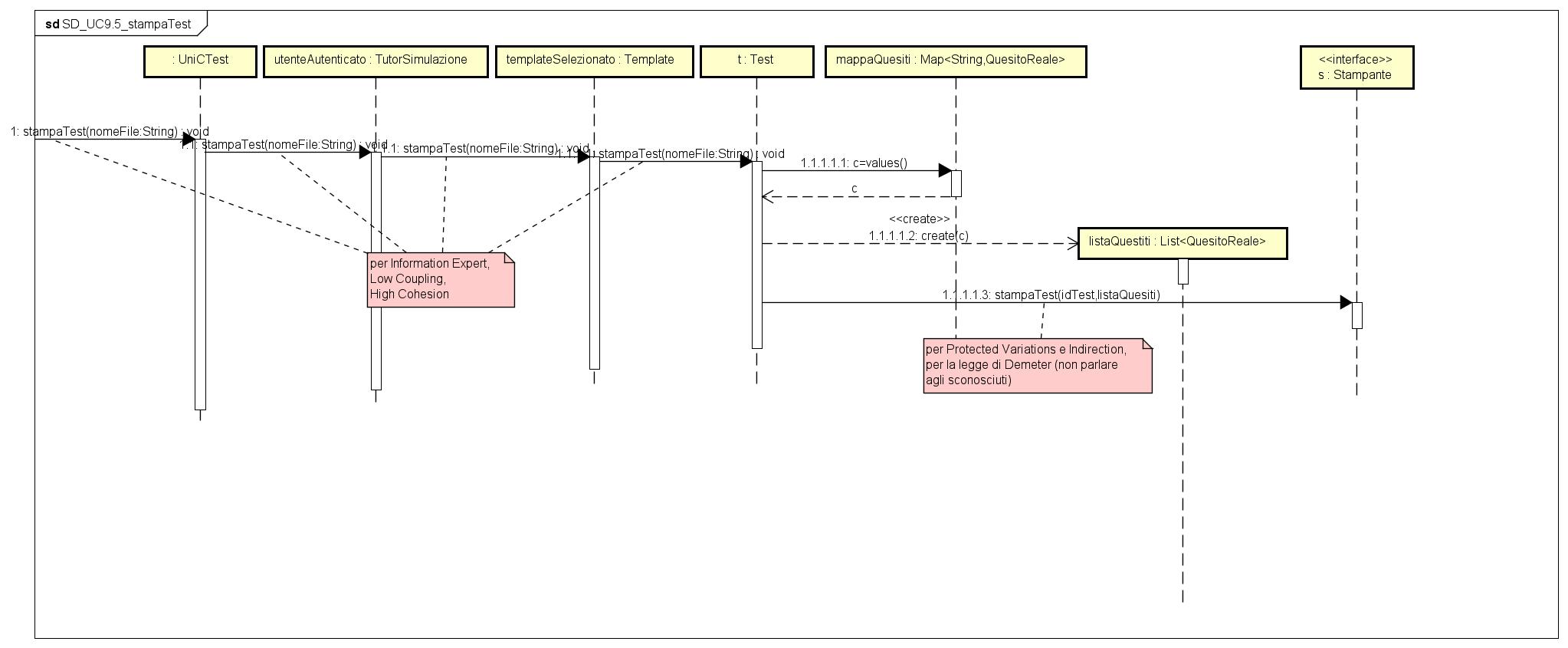
|  |  |
| --- | --- |
| Operazione | stampaTest(nomeFile) |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | È in corso la creazione di un test *t* per una simulazione cartacea. |
| Post-condizioni | * Il Test e il ModuloRiposte sono stati restituiti al Tutor. |

#### Progettazione Orientata agli Oggetti UC9

Aggiornate le classi concettuali nel contesto della OOA, si passa alla OOD, realizzando il Modello di Progetto ed in particolare aggiornando il DCD (visualizzazione statica delle classi software), aggiornamento da effettuare il parallelo alla realizzazione dei diagrammi di interazione (visualizzazione dinamica delle classi software).

Bisogna notare che la classe concettuale ModuloRisposte non ha trovato una concretizzazione lato Software in quanto altre classi possono sopperire alla mancanza di un modulo risposte. In particolare è QuesitoReale a fare le veci di un modulo risposte.

##### Diagrammi di interazione

1. **SD\_UC9\_visualizzaTemplate**
2. **SD\_UC9\_creaTestCartaceo**
3. **SD\_UC9\_visualizzaQuesiti**
4. **SD\_UC9\_inserisciQuesiti**
5. **SD\_UC9\_stampaTest**

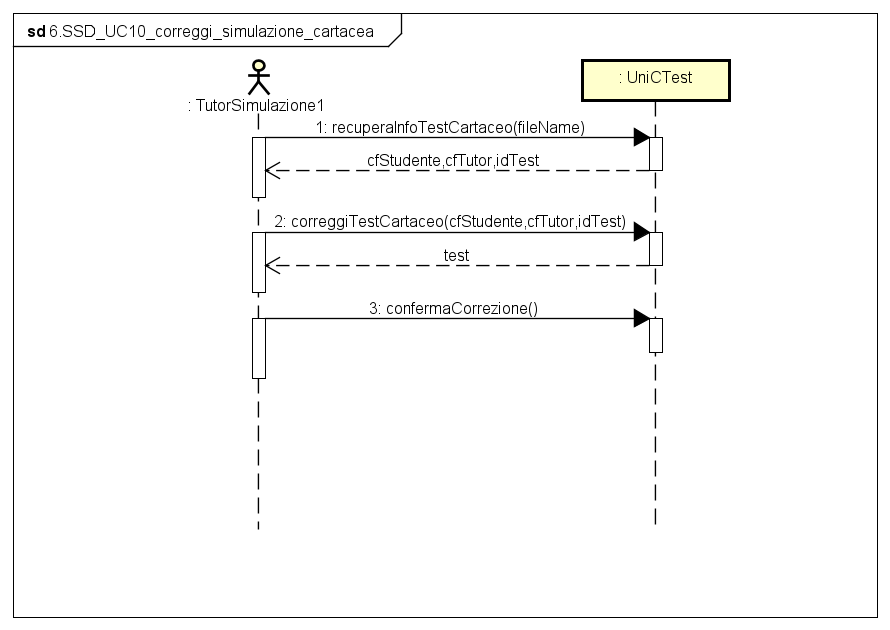
##### DCD

### Analisi e Progettazione UC10

#### Analisi Orientata agli Oggetti UC10

##### Diagramma di Sequenza di Sistema UC10

Una volta aggiornato il Modello di Dominio, viene creato il SSD relativo a UC10, che consente di visualizzare in forma grafica l’interazione tra il TutorSimulazione ed il Sistema.



##### Contratti delle Operazioni UC10

Di seguito si riportano le descrizioni delle Operazioni eseguite in riferimento a UC10.

### CO1

|  |  |
| --- | --- |
| Operazione | recuperaInfoTestCartaceo(fileName: String): Map<String,String> |
| Riferimenti | UC10: Correggi simulazione cartacea |
| Pre-condizioni | - |
| Post-condizioni | * È stato recuperato il file *fileName* * Dal file *fileName* sono state estratti: cf dello Studente, cf del Tutor, id del Test (che comprende l’id del Template). * Le informazioni sono state restituite al Tutor in una Mappa. |

### CO2

|  |  |
| --- | --- |
| Operazione | correggiTestCartaceo(cfStudente: String, cfTutor: String, idTest: String): Test |
| Riferimenti | UC10: Correggi simulazione cartacea |
| Pre-condizioni | È stato recuperato il test da correggere fileName |
| Post-condizioni | * Il Sistema ha recuperato lo Studente *s* avente *s*.cf uguale a cfStudente ed è stato associato all’istanza *lettore* del TutorSimulazione utenteAutenticato tramite l’associazione “corrente” * È stata recuperata l’istanza *t* di Test avente id=idTest associata all’istanza *te* di Template tramite l’associazione “è la struttura di”, a sua volta associato all’istanza *ts* di TutorSimulazione avente cf=cfTutor, tramite l’associazione “scrive test basati su”. L’istanza *t* è stata clonata nell’istanza *tc* di Test, che diventa “corrente”. * È stata cercata l’istanza *tec* associata allo Studente *s* tramite l’associazione “svolge test basati su”, avente *tec*.id uguale a *te*.id. Qualora *tec* non fosse stata trovata, *tec* è stata ricavata dalla clonazione di *te. tec* è stata associata a *lettore* tramite “corrente”. * L’istanza *tc* è stata inizializzata in questo modo: dal file *fileName* sono state estratte le informazioni relative al Test effettuato (con informazioni si intendono i QuesitiReali, le Risposte date). * Il Sistema ha restituito *tc*. |

### CO3

|  |  |
| --- | --- |
| Operazione | confermaCorrezione() |
| Riferimenti | UC10: Correggi simulazione cartacea |
| Pre-condizioni | È in corso la correzione del Test *tc* e sono state recuperate le istanze *s* di Studente e *tec* di Template |
| Post-condizioni | * Il TutorSimulazione ha controllato che i parametri dell’istanza *tc* di Test che sono stati ricavati dalla correzione automatizzata sono corretti (in caso contrario li ha corretti manualmente). * Il Sistema ha associato l’istanza *tec* corrente allo Studente *s* corrente tramite l’associazione “svolge test basati su”. Poi *tc* corrente è stato associato a *tec* corrente tramite l’associazione “è la struttura di”. |

**Estensioni: qualora il Tutor si renda conto di un errore nella scansione delle risposte.**

### COE1

|  |  |
| --- | --- |
| Operazione | selezionaRisposta(idQuesitoReale: String, idRisposta: String) |
| Riferimenti | UC10: Correggi simulazione cartacea |
| Pre-condizioni | È in corso la correzione del Test *tc* e sono state recuperate le istanze *s* di Studente e *tec* di Template |
| Post-condizioni | * Se la Risposta *r* avente *r*.id uguale a idRisposta era già presente nella mappa qr.risposteDate, allora *r* è stata rimossa dalla mappa. In caso contrario, è stata aggiunta. |

#### Progettazione Orientata agli Oggetti UC10

##### Introduzione dell’interfaccia Lettore e della classe Adapter: pattern Indirection (GRASP), Protected Variations (GRASP), Adapter (GoF), legge di Demeter

Il Lettore, nel Diagramma di Sequenza relativo a recuperaInfoTestCartaceo(), si occupa di recuperare le informazioni base relative al foglio risposte di uno Studente. In particolare, deve recuperare il codice fiscale dello Studente che ha svolto il test, il codice fiscale del Tutor che lo ha creato e l’ID del test, che a sua volta contiene l’ID del template su cui il test è basato.

Tali informazioni sono lette dal Lettore mediante l’ausilio della classe Tesseract. Il Lettore è una interfaccia i cui metodi sono implementati dalla classe TesseractObjectAdapter. Quest’ultima disaccoppia la classe TutorSimulazione dalla classe Tesseract, proteggendo la classe TutorSimulazione da variazioni future di Tesseract (e sostenendo il riuso del codice, in quanto il cliente ha espresso la sua volontà di modificare il meccanismo di lettura, il che porterà alla sostituzione della classe Tesseract ma che consentirà di mantenere i metodi attuali in TutorSimulazione, grazie alla presenza dell’interfaccia Lettore), in linea con la legge di Demeter (o di “non parlare agli sconosciuti”).

Questo disaccoppiamento è promosso dal pattern GRASP Indirection, che permette di assegnare una responsabilità evitando l’accoppiamento diretto tra due elementi.

La motivazione per cui è stata introdotta un’interfaccia è chiaramente quella di proteggere la classe TutorSimulazione dalle variazioni della classe Tesseract, secondo il pattern GRASP Protected Variations. Il pattern Protected Variations risponde al problema: “come progettare oggetti, sottosistemi e sistemi in modo tale che le variazioni o l’instabilità in questi elementi non abbiano un impatto indesiderato su altri elementi?”. In particolare, fase preliminare del procedimento è che devono essere individuati, nel codice, i *punti di variazione* (cioè di variazione dei requisiti correnti) e i *punti di evoluzione* (cioè di potenziali variazioni future). La lettura del foglio risposte è sicuramente un *punto di evoluzione*, come espressamente dichiarato dal cliente.

Come risoluzione, viene introdotto un Adapter (appunto TesseractObjectAdapter) nasconderà eventuali variazioni nei sistemi esterni di lettura del foglio risposte.

Le considerazioni sull’uso di tali pattern restano valide anche per l’introduzione della interfaccia Stampante e della classe Adapter PdfWriterObjAdapter in UC9.

##### Il Lettore e il Pattern Prototype (GoF)

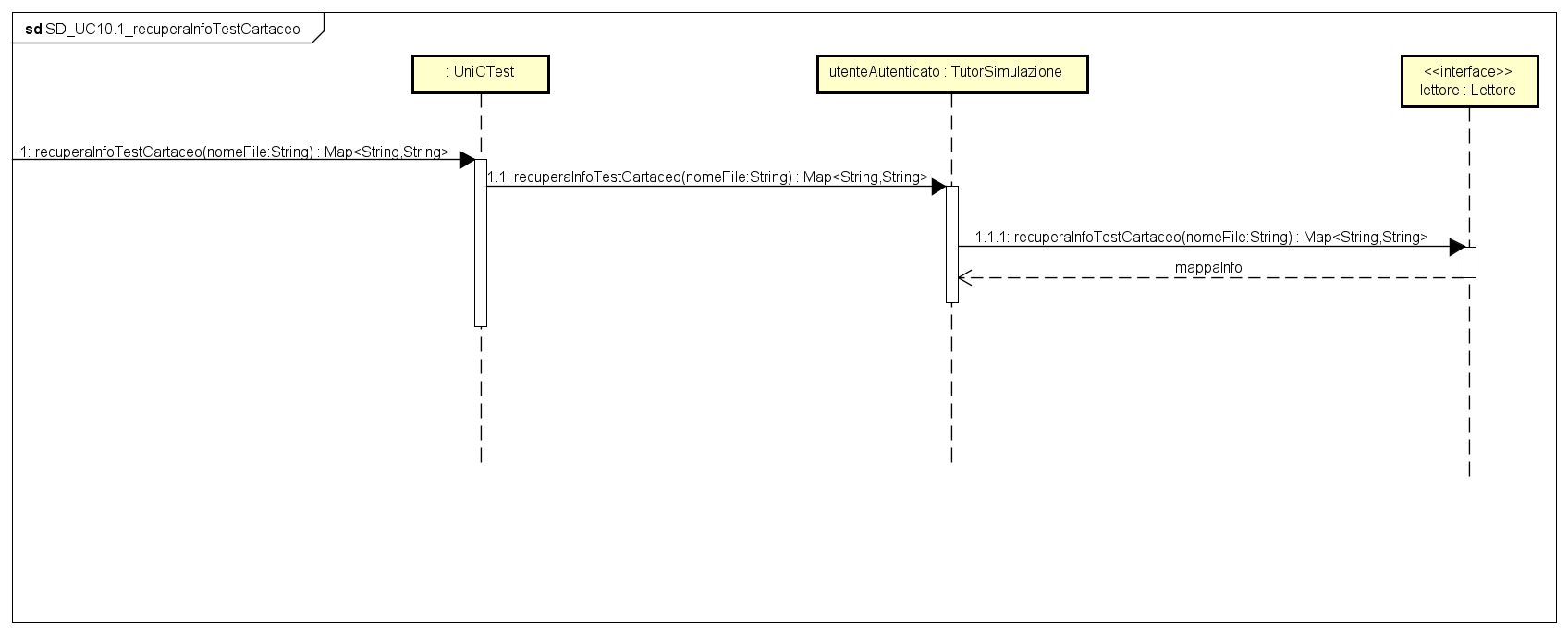
Se il Lettore, all’atto di creazione del test, ne effettuasse lo storage nella lista del template dal quale è stato creato, allora il test eseguito dallo Studente rimarrebbe associato al Tutor.

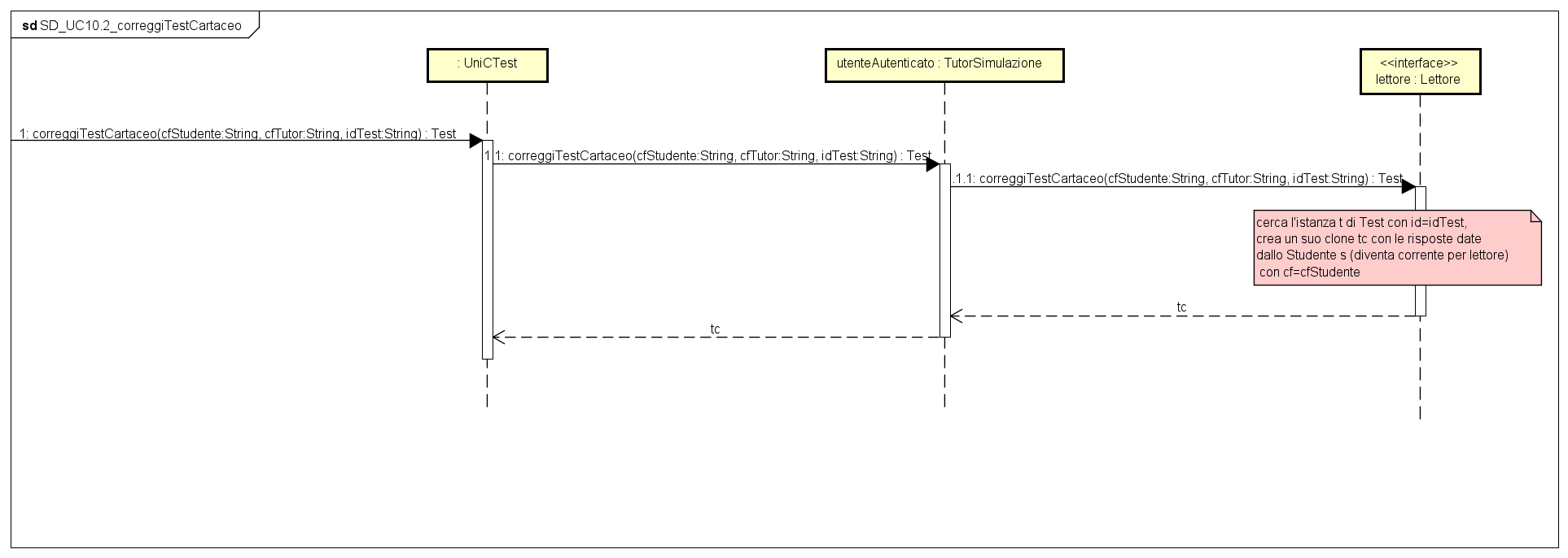
Per risolvere questo problema, il template deve essere clonato e conservato nella lista dei template relativi a test già svolti nello Studente.

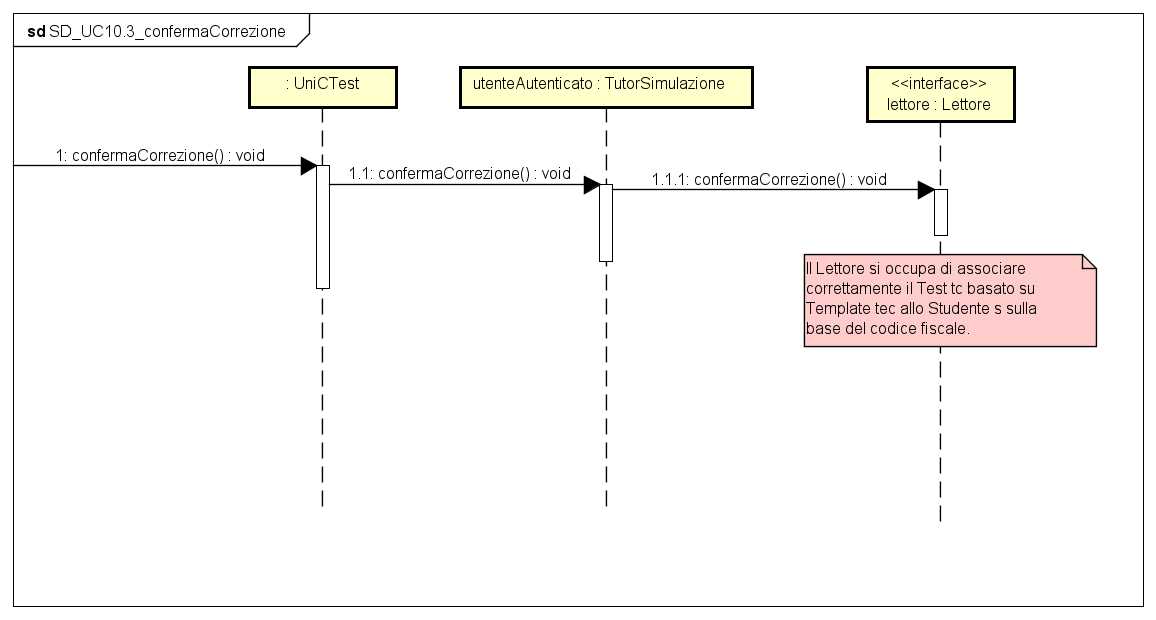
Allo stesso modo, il test deve essere clonato per far sì che ne venga effettuata la correzione, in quanto più Studenti avranno effettuato la simulazione ed ognuno avrà dato risposte diverse, avrà raggiunto un punteggio complessivo diverso, ecc.

Ciò richiederebbe l’applicazione del pattern GoF Prototype, ma Java consente un’implementazione efficiente della clonazione di un oggetto mediante l’utilizzo del metodo clone() della classe Object.

##### Diagrammi di interazione

**1. SD\_UC10\_recuperaInfoTestCartaceo**

**2. SD\_UC10\_correggiTestCartaceo**

**3. SD\_UC10\_confermaCorrezione**

##### DCD

### Implementazione e Testing

#### Implementazione

Per l’Implementazione si rimanda al codice in allegato.

#### Testing

**TestCreaTemplatePersTutor.java**

Il seguente test si occupa di verificare la creazione di un nuovo TemplatePersonalizzato, stavolta da parte di un TutorSimulazione.

initTest() si limita semplicemente ad inizializzare l'istanza di UniCTest

testNuovoTemplate() autentica un Tutor e lo rende TutorSimulazione, quindi verifica che il metodo nuovoTemplateP() inizializzi correttamente un TemplatePersonalizzato "corrente" per il TutorSimulazione

testInserisciInfoTemplate() verifica la correttezza dei dati inseriti dal metodo inserisciInfoTemplateP() sul TemplatePersonalizzato "corrente", ovvero le informazioni sui punteggi e sul formato dei quesiti, inoltre verifica che la lista di materie ritornata sia non nulla.

testCreaSezione() verifica che la creazione di una nuova Sezione per il TemplatPersonalizzato corrente vada a buon fine, in particolare crea una Sezione per ogni Materia presente nel sistema, dunque verifica che ciascuna Sezione venga inserita correttamente nella listaSezioni del template

testConfermaTemplate() verifica che il TemplatePersonalizzato non sia più "corrente" e che venga correttamente inserito nella mappaTemplatePersonalizzati del Tutor.

testEccezioni() autentica un Tutor senza i permessi di TutorSimulazione e cerca di creare un TemplatePersonalizzato. Dunque, viene lanciata un'eccezione, solo in tal caso il test si considera superato. Dopodiché viene anche testato con un tutor Amministratore, anche qui dovrà essere lanciata un'eccezione.

**TestCreaTestCartaceo.java**

I test della classe TestCreaTestCartaceo si occupano di verificare che la creazione di un Test da parte di un TutorSimulazione.

initTest() impostazione delle condizioni iniziali: recupera l’istanza di UniCTest (Singleton) e autentica uno TutorSimulazione. Le condizioni iniziali sono del tutto analoghe a quelle dei test relativi all'avvio di una simulazione, dunque si inserisce un nuovo QuesitoDescrizione e un nuovo TemplatePersonalizzato (stavolta si tratta di un template creato dal TutorSimulazione) su cui poter basare il Test.

testVisualizzaTemplateTutor() analoga a testVisualizzaTemplate() ma viene effettuata dal TutorSimulazione, dunque la lista restituita conterrà i template creati dal TutorSimulazione.

testCreaTestCartaceo() verifica che la lista di Sezioni del TemplatePersonalizzato "selezionato" non sia vuota e che venga inizializzata l'istanza di Test come "corrente" per il template.

testInserisciQuesiti() verifica che la Sezione di cui abbiamo fornito l'id diventi "corrente" per il TemplatePersonalizzato e che la Materia a cui essa fa riferimento ritorni una lista non nulla di QuesitiDescrizione. Scelti gli id dei quesiti da voler inserire, provvede ad inserirli nella Sezione e viene verificato che la mappaQuesiti del Test "corrente" non sia vuota.

testStampaTest() verifica che la creazione del file pdf relativo al Test appena creato vada a buon fine e che l'istanza templateSelezionato dello Studente venga reimpostata a null.

testSelezionaRisposta() verifica prima che l'inserimento di una risposta nell'istanza di QuesitoReale vada a buon fine, controllando che nella mappa "risposteDate" dell'istanza di QuesitoReale ci sia un'istanza Risposta. Poi verifica che, fornendo come idRisposta lo stesso di prima, la mappa "risposteDate" sia vuota (caso di deselezionamento della risposta).

testEccezioniCreaTestCartaceo() effettua verifiche del tutto analoghe a testEccezioniAvviaSimulazione(), cioè verifica che il metodo che si occupa di ritornare una lista di QuesitiDescrizione per la Sezione scelta, non potendo soddisfare le richieste "irrealistiche" del TemplatPersonalizzato, lanci un'eccezione.

**TestCorreggiSimulazCartacea.java**

I test della classe TestCorreggiSimulazCartacea si occupano di verificare che la creazione di un Test da parte di un TutorSimulazione.

initTest() impostazione delle condizioni iniziali: recupera l’istanza di UniCTest (Singleton) e autentica uno TutorSimulazione. Si occupa, in particolare, di inizializzare dei quesiti e creare un'istanza di Test basata su un TemplatePersonalizzato da parte del tutor.

testRecuperaInfo() verifica che i dati prelevati dal test compilato da uno studente (viene fornita una scansione predefinita di un modulo risposte relativo al Test creato da initTest()) non siano nulli, in particolare il codice fiscale dello Studente, del TutorSimulazione creatore del Test e l'id del Test.

testCorreggiTestCartaceo() verifica la restituzione di un oggetto Test non nullo contenente le risposte date dallo Studente nel modulo risposte, l'oggetto sarà un clone dell'istanza di Test creata dal Tutor in initTest().

testConfermaCreazione() verifica che l'inserimento del Test cartaceo appena corretto nella *collection* dello Studente avvenga a buon fine, in particolare verifica che la dimensione di mappaTemplateTestSvolti dello Studente aumenti di 1 (poiché si sta utilizzando un template appena creato) e che essa contenga l'istanza del Test corretto con l'attributo punteggioComplessivo non nullo.

## Iterazione VI

### Introduzione

I Casi d’Uso già esplorati coprono il *core* del progetto, cioè soddisfano le motivazioni cardine per cui il cliente aveva deciso di commissionare UniCTest: permettere agli studenti di esercitarsi mediante lo svolgimento di simulazioni sul proprio calcolatore, consentendogli di personalizzarle, e permettere ai tutor di creare, svolgere, correggere le simulazioni cartacee con il minore sforzo possibile, lasciando la possibilità di riutilizzare il lavoro fatto in altre simulazioni, collaborare con i colleghi nella realizzazione delle stesse, ecc.

Il cuore del progetto è stato sviluppato facendo affidamento al Caso d’Uso di Avviamento, tramite il quale venivano inseriti nel sistema Tutor, Studenti e Tutor con responsabilità (Ammministratori e TutorSimulazione). Tuttavia, non è ancora possibile aggiungere tali Utenti al Sistema né è possibile aggiungere/rimuovere responsabilità a *runtime*, come promesso dal pattern Decorator applicato nella Iterazione 4 della fase di Elaborazione.

Inoltre, manca lo sviluppo di alcune primitive che consentano di visualizzare il lavoro svolto, ad esempio i test effettuati dallo Studente e i quesiti già inseriti dal Tutor.

Nella Iterazione 6 della fase di Elaborazione, dunque, ci si occuperà di

* UC6: Nuovo Tutor
* UC13: Aggiungi materia insegnata
  + Dovrà prevedere l’applicazione del pattern Flyweight
* UC14: Aggiungi responsabilità   
  (Aggiungi Amministratore/Aggiungi TutorSimulazione)
* UC15: Rimuovi responsabilità   
  (Rimuovi Amministratore/Rimuovi TutorSimulazione)
* UC5: Nuovo Studente
* UC16: Visualizza quesiti creati (CRUD)
  + Visualizzazione quesiti creati da parte del Tutor
* UC17: Visualizza test svolti (CRUD)
  + Visualizzazione test svolti da parte dello Studente

È stato deciso di non sviluppare i Diagrammi di Sequenza di Sistema in quanto i Casi d’Uso citati prevederanno l’uso di una singola operazione e dunque di una singola interazione Attore-Sistema software, pertanto i Diagrammi di Sequenza di Sistema risulterebbero ovvi.

In questa Iterazione vengono mostrati i diagrammi finali, in quanto si prevede che questa sia l’ultima Iterazione del progetto.

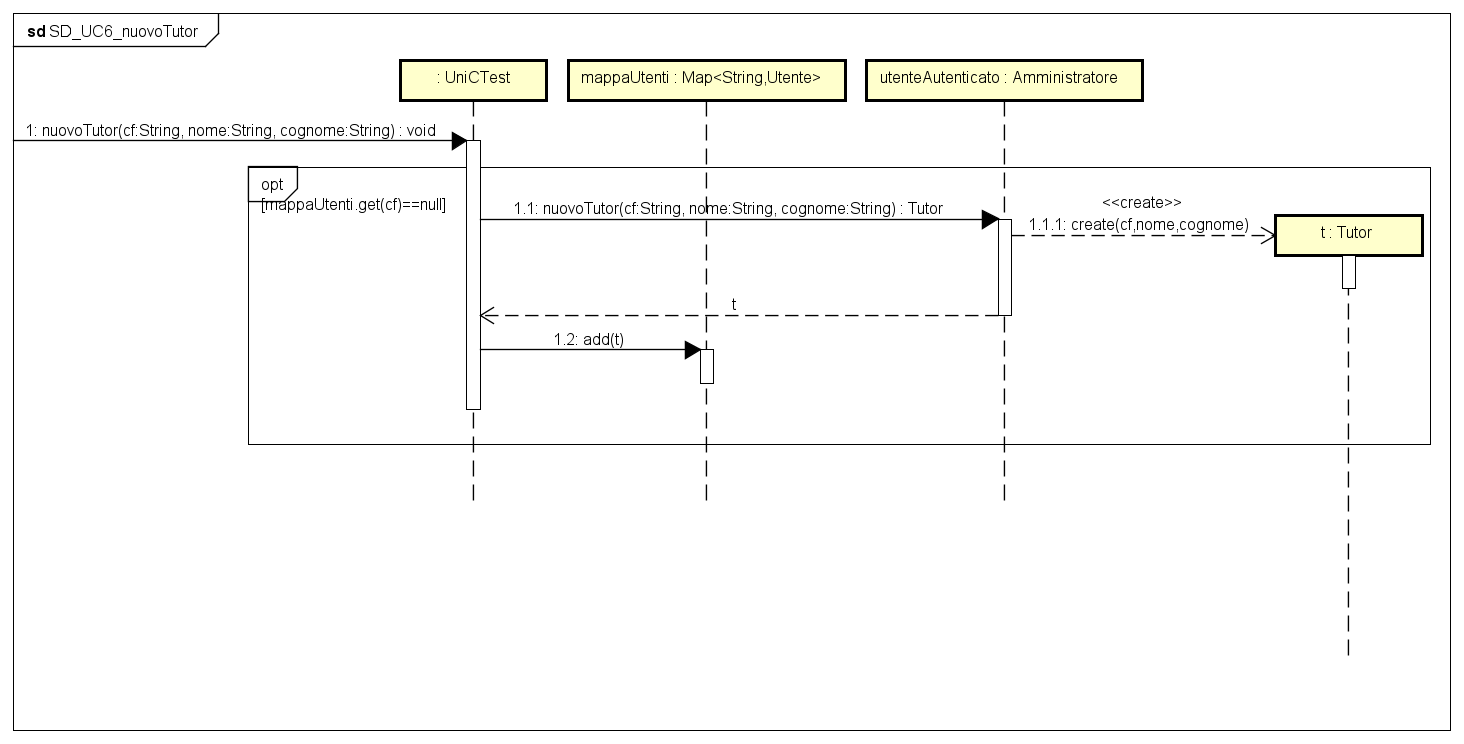
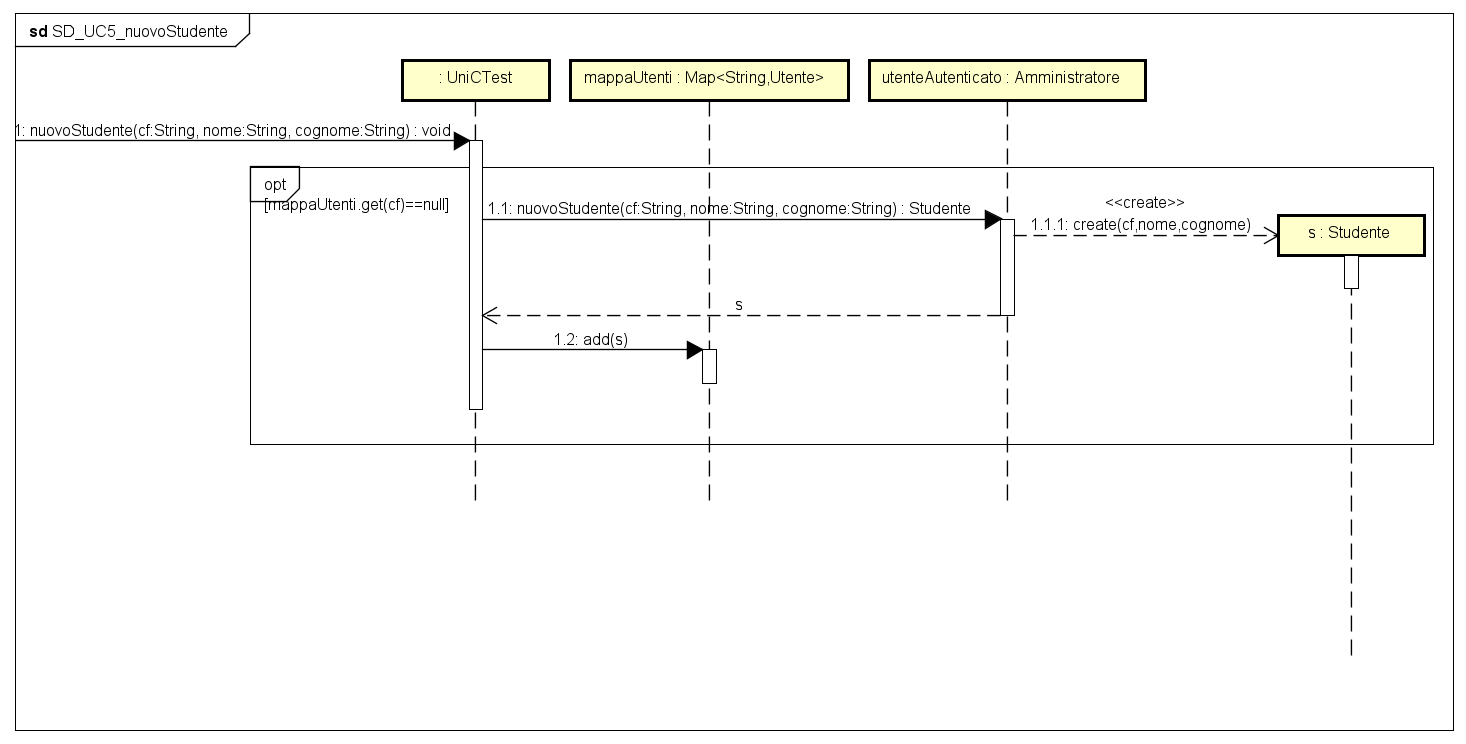
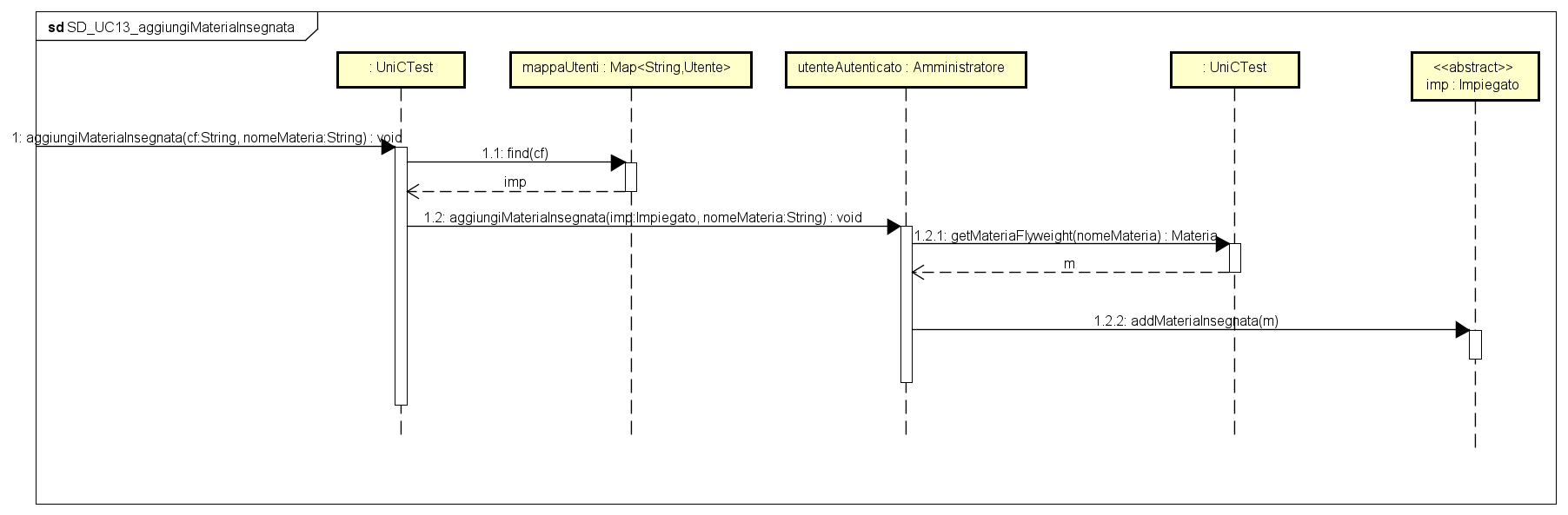
### Analisi orientata agli oggetti

#### Modello di dominio finale

### Progettazione Orientata agli Oggetti

Vengono di seguito riportati i Diagrammi di Sequenza relativi alle operazioni reputate più complesse o con maggiore richiesta di dettagli. Per le operazioni notoriamente più semplici viene riportata direttamente l’Implementazione.

##### Diagrammi di interazione

1. **SD\_UC5\_nuovoStudente**
2. **SD\_UC6\_nuovoTutor**
3. **SD\_UC\_13\_aggiungiMateriaInsegnata**

#### DCD finale

### Implementazione e Testing

#### Implementazione

Per l’Implementazione si rimanda al codice in allegato.

#### Testing

**TestAggiungiResponsabilità.java**

**Si tratta di test sviluppati per testare i Casi d’Uso UC14 e UC15, ovvero l’aggiunta e la rimozione delle responsabilità addizionali di un Tutor (pattern GoF Decorator).**

Sono state sviluppate due tipologie di test differenti:

* Test a **scatola nera (black-box)**. Lo sviluppo dei test black-box è stato promosso dalle pratiche di test-driven development introdotte nell’Iterazione 4 della fase di Elaborazione. I test effettuati sono stati sviluppati prescindendo dalla conoscenza dei dettagli implementativi del codice, ma hanno analizzato il blocco software di assegnazione delle responsabilità addizionali ai Tutor come un sistema ingresso-uscita.
* Testa a **scatola bianca (white-box)**. Sono stati effettuati test a scatola bianca con ***copertura delle decisioni*** (che include la copertura delle istruzioni).   
  L’obiettivo della copertura delle decisioni è individuare un insieme di casi di test in modo da garantire che ogni decisione assuma il valore vero almeno una volta.  
  In sostanza è stato deciso di testare se un Tutor che era stato reso prima di tutto TutorSimulazione e solo dopo Amministratore veniva correttamente visto come TutorSimulazione. Allo stesso modo doveva essere verificato che un Tutor che era stato reso prima di tutto TutorSimulazione e solo dopo Amministratore veniva correttamente NON visto come TutorSimulazione una volta chiamata la funzione rimuoviTutorSimulazione.  
  Questi test non sarebbero stati effettuati se fossero stati elaborati solo test a scatola nera. Infatti la decisione di testare questa condizione deriva dalla conoscenza del workflow interno delle funzioni per aggiungere e rimuovere le responsabilità addizionali. Infatti:

@Override  
public Impiegato rimuoviAmministratore(Impiegato imp){  
 Impiegato attuale = imp;  
 Impiegato precedente = null;  
 Impiegato successivo = imp.getImpiegato();   
 //Struttura: PREC(ATTUALE(SUCC))  
 while(successivo != null){  
 if(attuale instanceof Amministratore){  
 if(precedente!=null){  
 precedente.setImpiegato(successivo);  
 return imp;  
 } else {  
 return successivo;  
 }  
 } else { //Scaliamo tutti al successivo  
 precedente = attuale; //Il PREC diventa l'ATT  
 attuale = successivo; //L'ATT diventa il SUCC  
 successivo = successivo.getImpiegato(); //Il SUCC diventa più SUCC  
 }  
 } //Interrompo se lo trovo o se non ho più successivi (cioè più interni)! --> Infatti il Tutor restituisce null perché non ha interni.  
 System.*out*.println("Non sei Amministratore, quindi non rimuovo nulla");  
 return imp;  
}

Dunque bisogna prevedere:

* + Il caso in cui *successivo*=*null*, e cioè il caso in cui l’Impiegato è un Tutor, dunque non ha responsabilità addizionali.
  + Nel caso in cui *successivo* non sia *null*,
    - Il caso in cui il *successivo* sia un Amministratore
      * Il caso in cui il precedente sia *null*, e cioè il caso (banale) in cui l’elemento passato è esso stesso un Amministratore che *wrappa* altre responsabilità al proprio interno
      * Il caso in cui il precedente non sia *null*, e dunque bisogna ricostruire la catena di impiegati rimuovendo quello *attuale*, e cioè PREC(ATT(SUCC))) diventa PREC(SUCC).
    - Il caso in cui i*successivo* non sia un Amministratore, in tal caso la ricerca deve procedere esplorando le responsabilità più interne.
* Di conseguenza, vengono analizzati i casi in cui la responsabilità cercata sia “esterna” e in cui sia “interna”.

Funzionalità testate:

* Test a scatola nera:
  + Diritti di Amministratore: è impossibile per un non-Amministratore aggiungere o rimuovere responsabilità.
  + Un Tutor reso Amministratore viene visto correttamente come Amministratore.
  + Un Tutor reso TutorSimulazione viene visto correttamente come TutorSimulazione.
  + Un Tutor reso Amministratore non può auto-rimuovere i diritti di Amministratore.
* Test a scatola bianca:
  + Un Tutor reso Amministratore e in seguito TutorSimulazione viene visto correttamente come Amministratore
  + Un Tutor reso Amministratore e in seguito TutorSimulazione viene visto correttamente come TutorSimulazione

1. Si veda il Glossario, alle voci Test, Simulazione per approfondire. [↑](#footnote-ref-1)
2. Si veda documento di Specifiche Supplementari per i metodi di scansione del modulo risposte. [↑](#footnote-ref-2)