UniCTest

Ingegneria del Software – A.A. 2021/2022

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

Fase di Elaborazione – Iterazione 5

Sommario

[1 Versione 2](#_Toc94906839)

[2 Introduzione 2](#_Toc94906840)

[3 Modello di Analisi 3](#_Toc94906841)

[3.1 Aggiornamento Modello di Dominio 3](#_Toc94906842)

[4 Analisi e Progettazione UC9 5](#_Toc94906843)

[4.1 Analisi Orientata agli Oggetti UC9 5](#_Toc94906844)

[4.1.1 Diagramma di Sequenza di Sistema UC9 5](#_Toc94906845)

[4.1.2 Contratti delle Operazioni UC9 5](#_Toc94906846)

[CO1 5](#_Toc94906847)

[CO2 6](#_Toc94906848)

[CO3 6](#_Toc94906849)

[CO4 6](#_Toc94906850)

[CO5 6](#_Toc94906851)

[4.2 Progettazione Orientata agli Oggetti UC9 7](#_Toc94906852)

[4.2.1 Diagrammi di interazione 7](#_Toc94906853)

[4.2.2 DCD 9](#_Toc94906854)

# Versione

|  |  |  |  |
| --- | --- | --- | --- |
| Versione | Data | Descrizione | Autori |
| Elaborazione 5 | 2021/02/03 | Quinta iterazione della fase di elaborazione. | Anzalone Dario Giuseppe, Blanco Francesco Giulio. |

# 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, 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**.

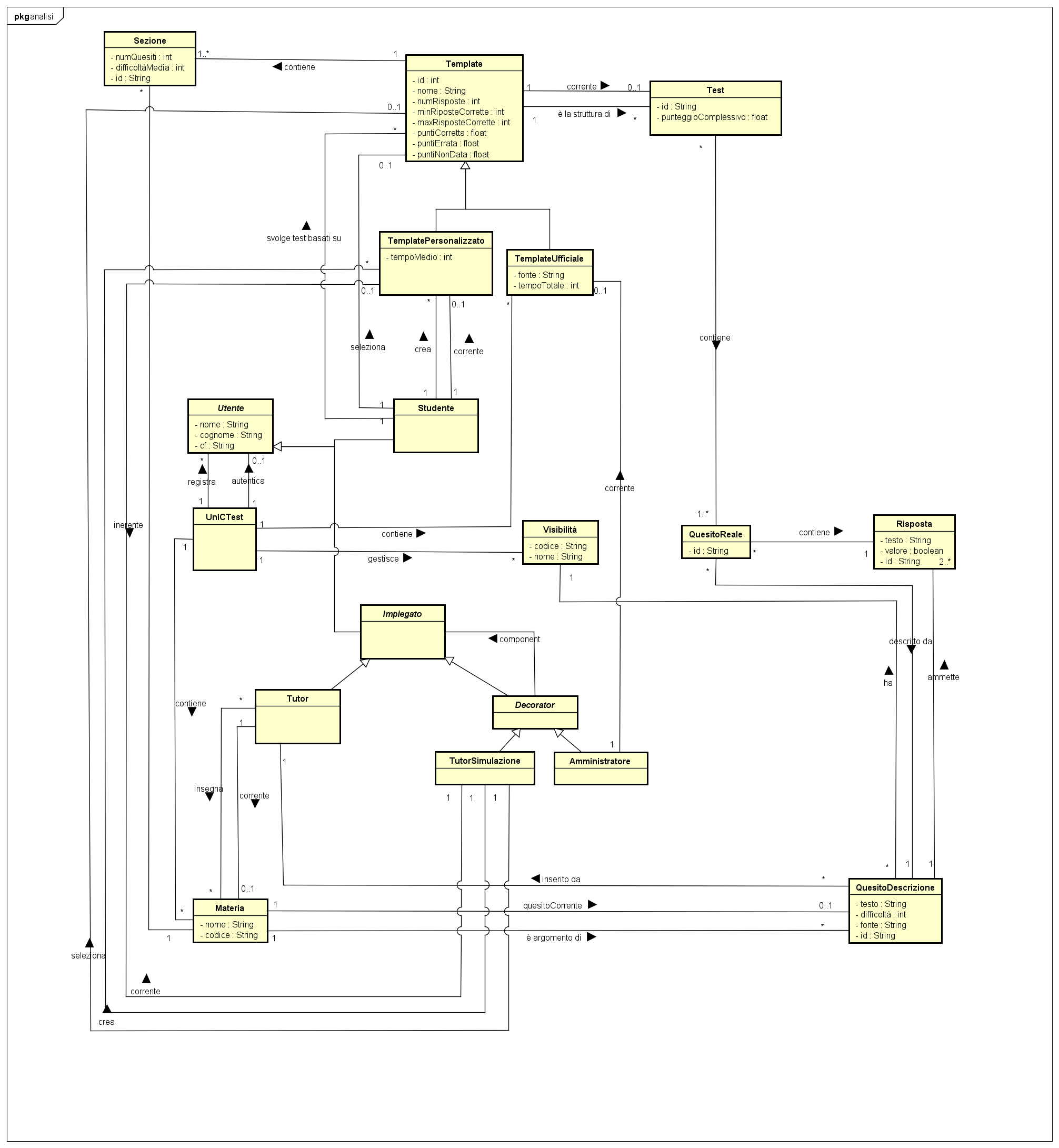
Ciò è stato applicato nella iterazione 4 della fase di Elaborazione nell’introduzione della classe concettuale Amministratore e, nella iterazione corrente, si concretizza anche per la classe concettuale TutorSimulazione.

**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.

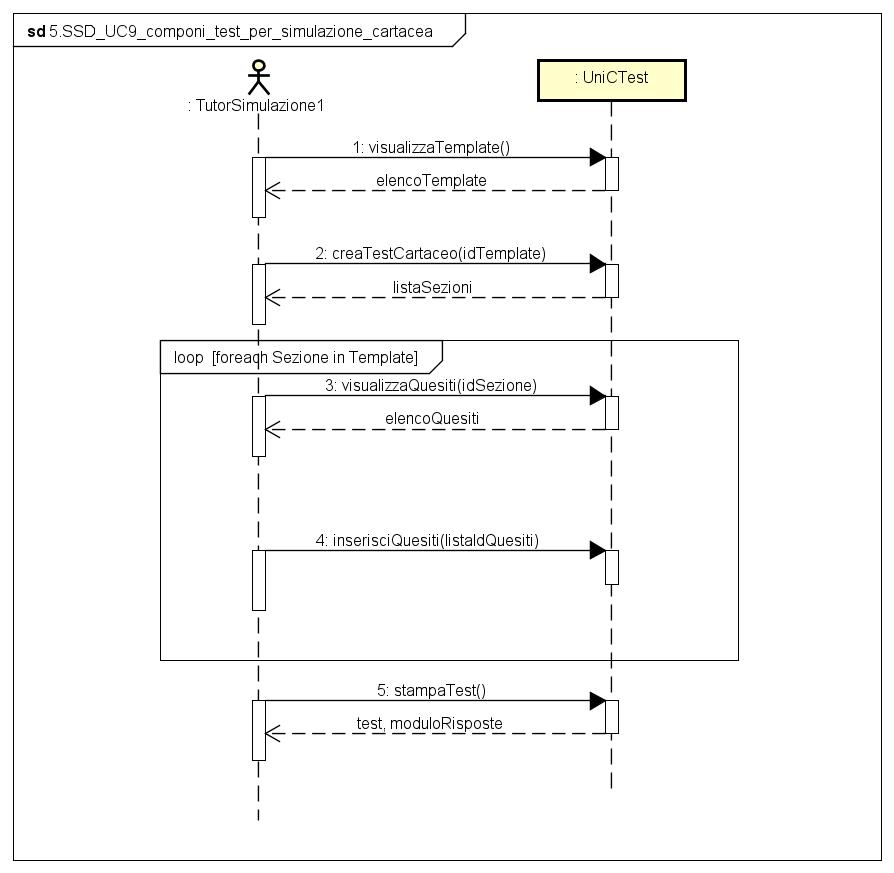
****Viene di seguito riportato il Modello di Dominio aggiornato alla luce delle recenti considerazioni:

# 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

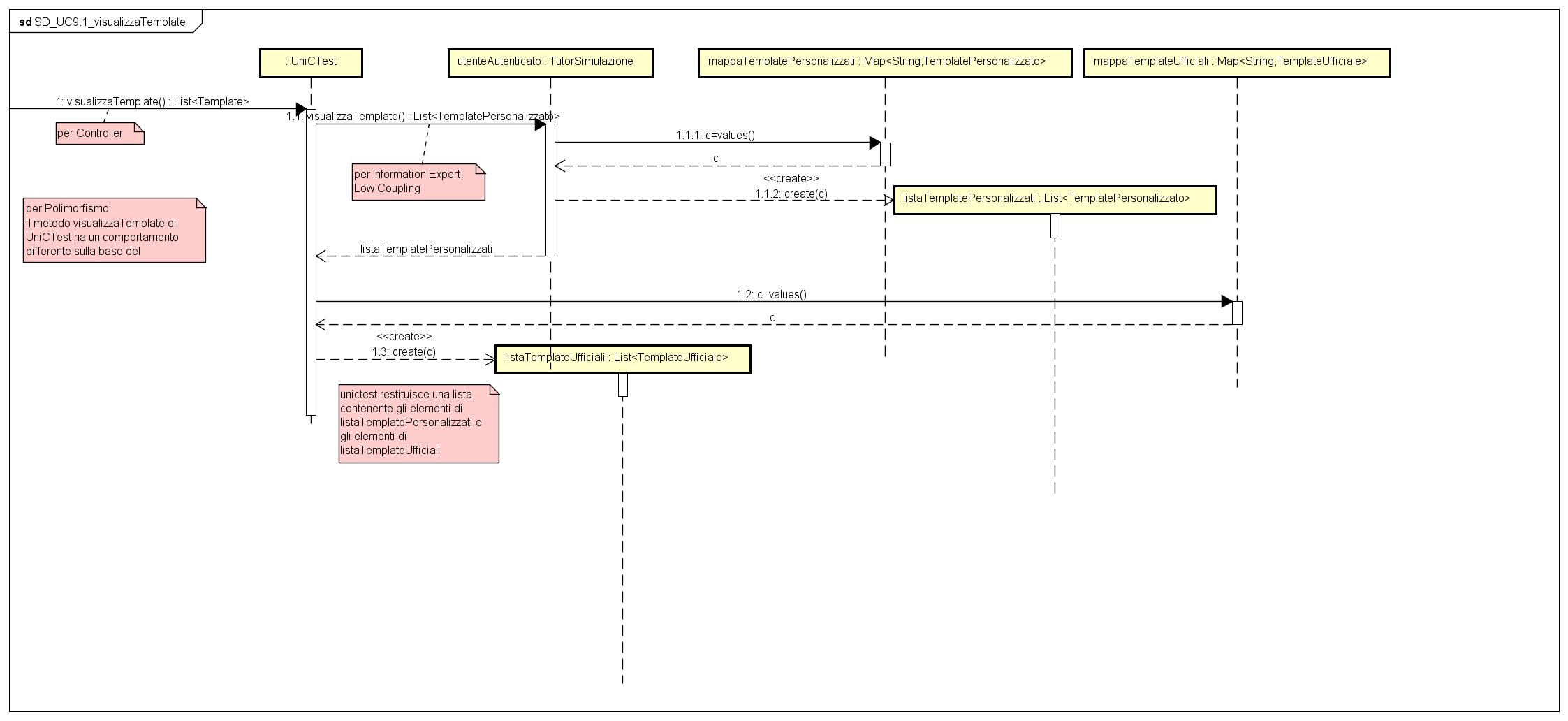
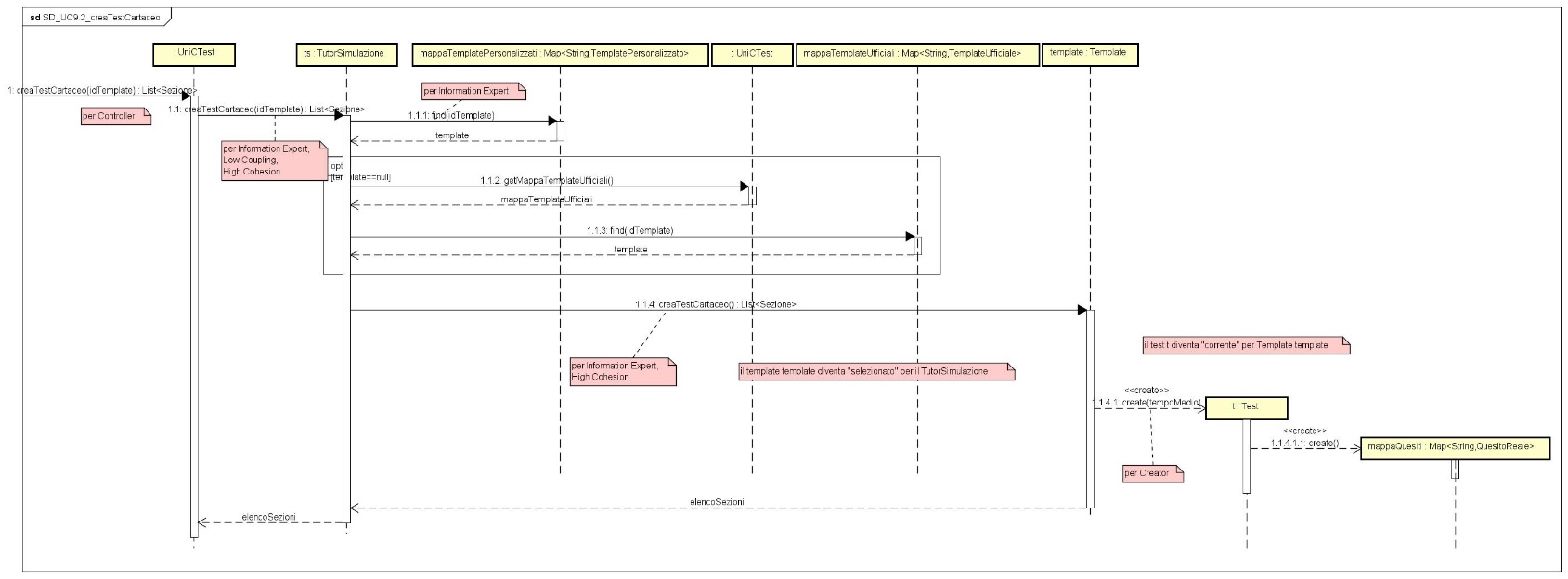
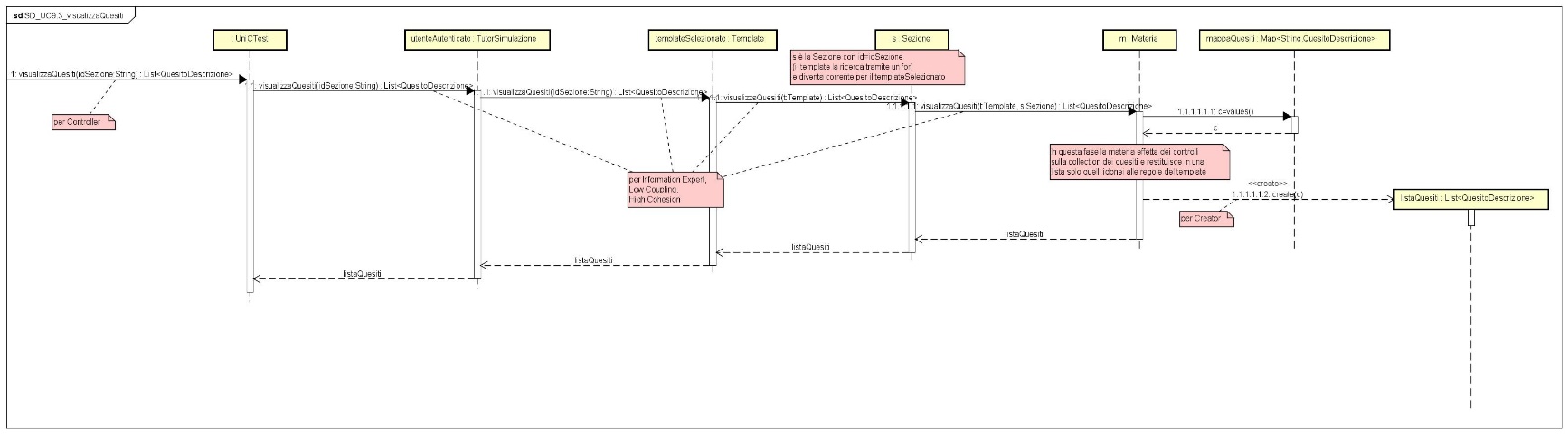
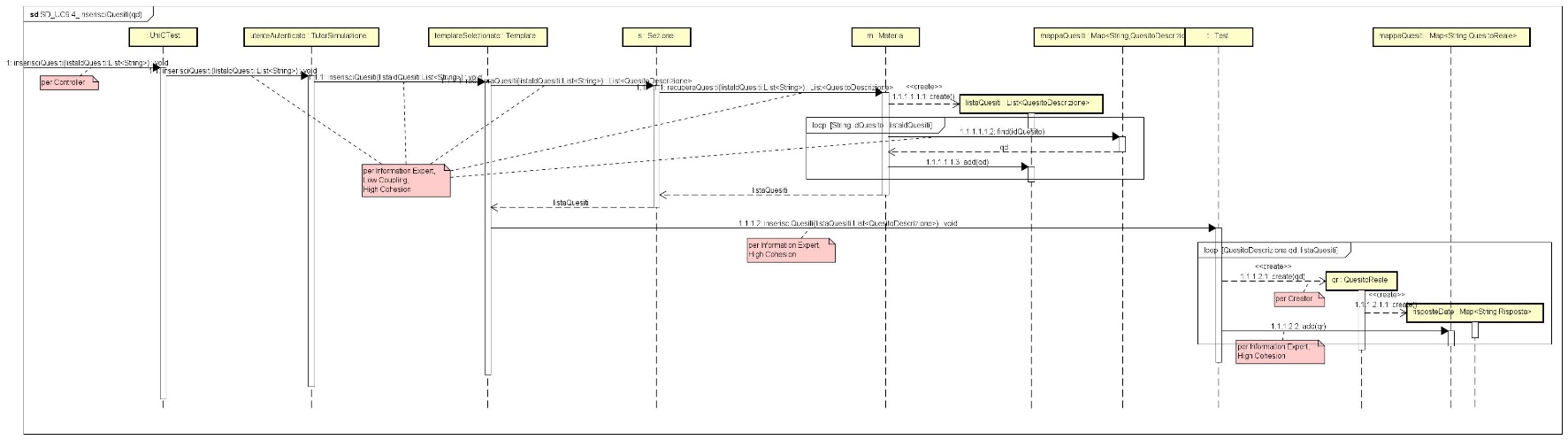
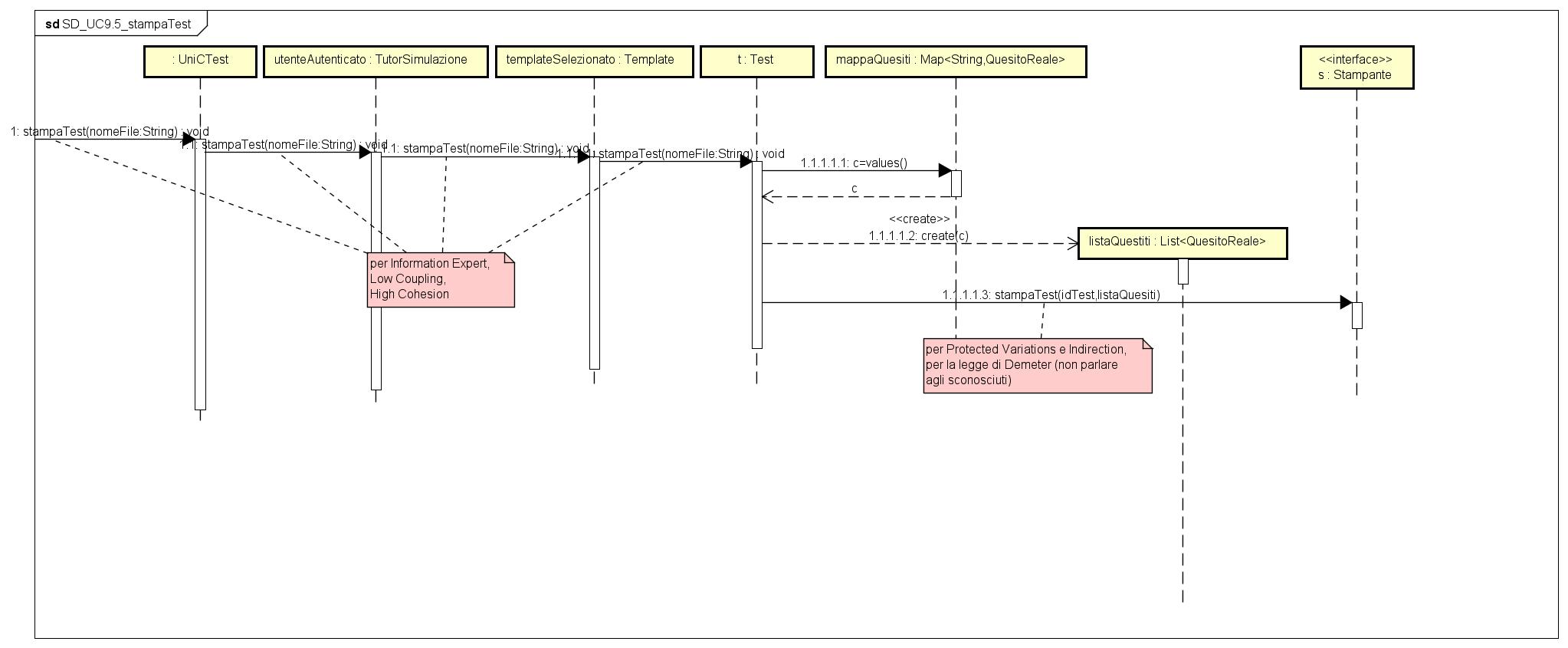
|  |  |
| --- | --- |
| Operazione | stampaTest(nomeFile) |
| Riferimenti | UC9: Componi test per simulazione cartacea |
| Pre-condizioni | È in corso la creazione di un test *t* per una simulazione cartacea. |
| Post-condizioni | * Il Test e il ModuloRiposte sono stati restituiti al Tutor. |

## 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**

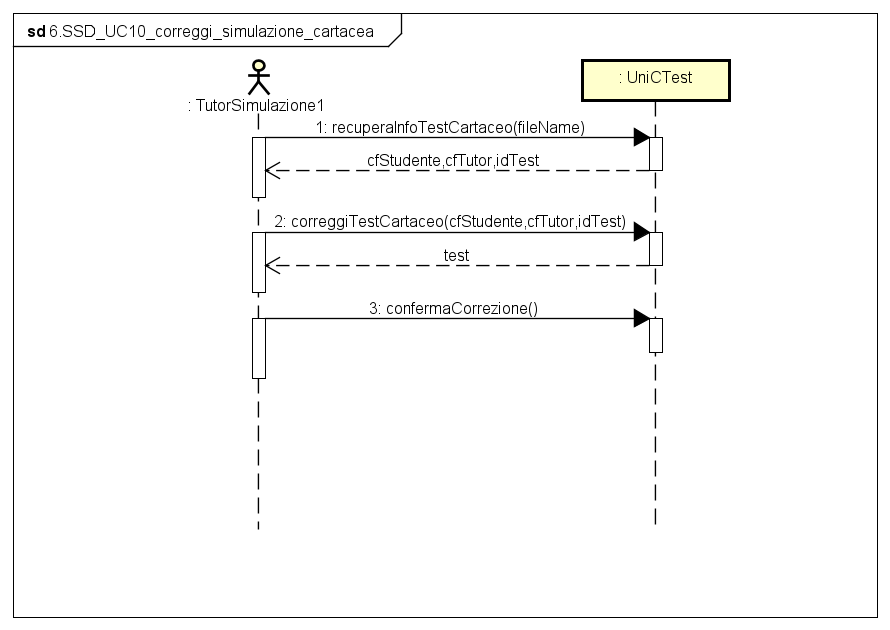
### DCD

# 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”. |

## 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)

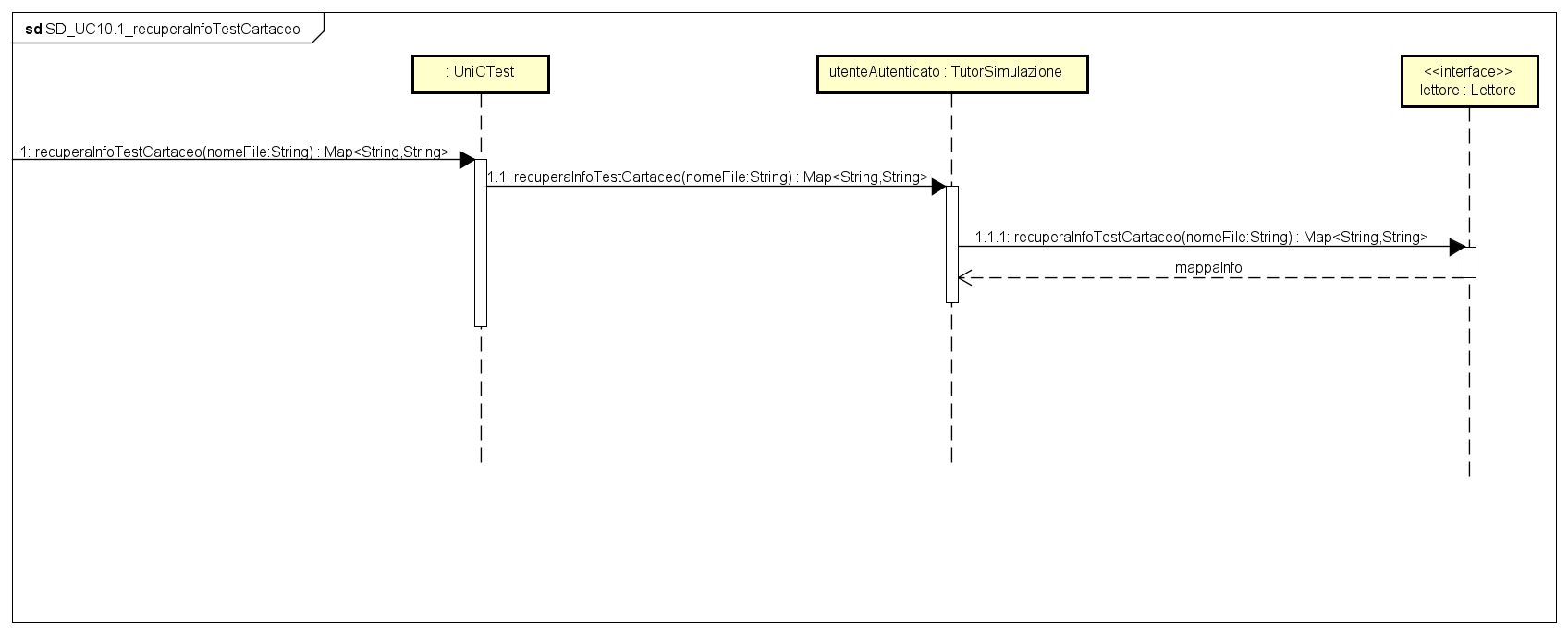
Se il Lettore, all’atto di creazione del test, ne effettuasse lo storage nella lista del template dal quale è stato creato, allora il test eseguito dallo Studente rimarrebbe associato al Tutor.

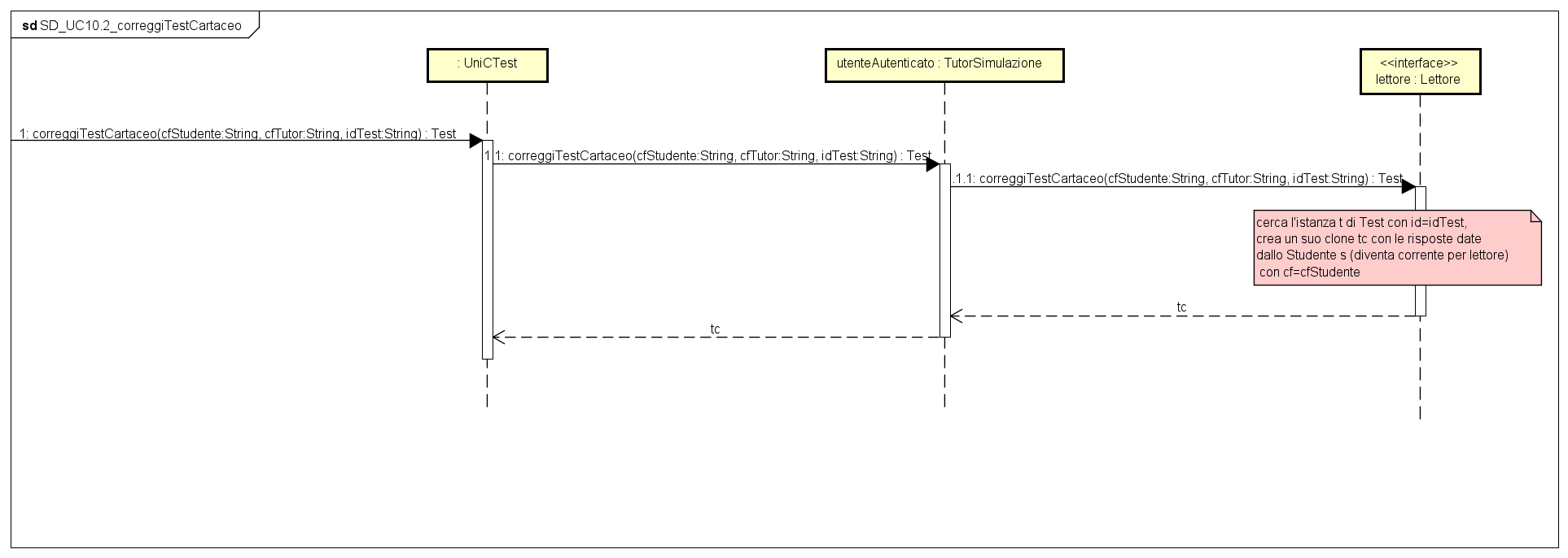
Per risolvere questo problema, il template deve essere clonato e conservato nella lista dei template relativi a test già svolti nello Studente.

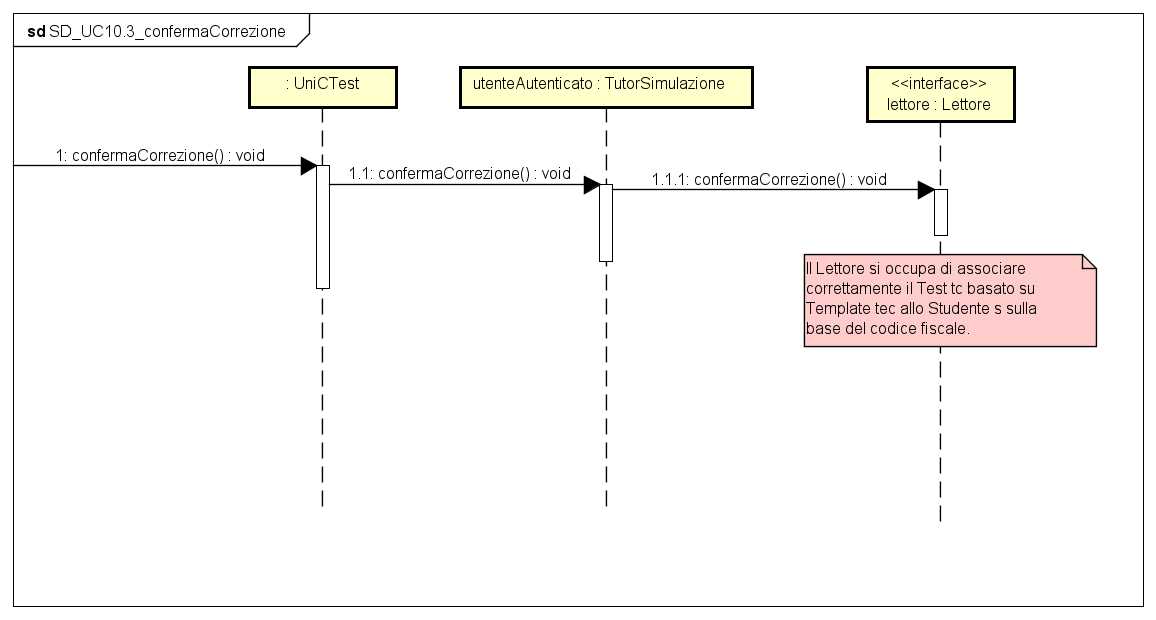
Allo stesso modo, il test deve essere clonato per far sì che ne venga effettuata la correzione, in quanto più Studenti avranno effettuato la simulazione ed ognuno avrà dato risposte diverse, avrà raggiunto un punteggio complessivo diverso, ecc.

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