*Progetto Ingegneria del Software*

**TSR**

**Elaborazione – Iterazione 2**



Prof. Orazio Tomarchio

Gianluca Cristaudo

Samuele Gulino

Ingegneria Informatica Magistrale LM-32 2019/2020

Sommario

[1 Introduzione 3](#_Toc35464543)

[2 Analisi orientata agli oggetti 3](#_Toc35464544)

[2.1 Aggiornamento caso d’uso UC4 3](#_Toc35464545)

[2.2 Modello di dominio 4](#_Toc35464546)

[*2.2.1* *Diagramma di macchina a stati* 5](#_Toc35464547)

[2.3 Diagrammi di sequenza di sistema 7](#_Toc35464548)

[2.4 Contratti delle operazioni 8](#_Toc35464549)

[3 Progettazione orientata agli oggetti 9](#_Toc35464550)

[3.1 Diagrammi di sequenza 9](#_Toc35464551)

[3.2 Diagramma delle classi 9](#_Toc35464552)

[4 Implementazione 11](#_Toc35464553)

# Introduzione

Durante la prima iterazione è stato ampiamente analizzato il caso d’uso UC3, relativo alla sottomissione di un articolo effettuata da un autore. È stato redatto il documento relativo alla nostra analisi e progettazione, e a ciò è seguita una fase di implementazione delle classi individuate. In questa iterazione arricchiremo la nostra analisi dei requisiti, prendendo in considerazione in particolare il caso d’uso UC4, relativo all’attribuzione di una valutazione da parte di un revisore. Verrà raffinato il Modello di Dominio, e questo porterò ad una successiva fase di progettazione ed implementazione.

# Analisi orientata agli oggetti

L’analisi orientata agli oggetti in questa fase si baserà sulla trattazione di nuovi concetti emergenti dal caso d’uso UC4, non meglio identificati precedentemente.

## Aggiornamento caso d’uso UC4

Per questo caso d’uso lo scenario principale di successo scritto durante la fase di Ideazione ha bisogno di meno modifiche di quelle effettuate alla prima iterazione per il caso d’uso UC3. Il principale cambiamento è dovuto alla sostituzione dell’username con l’e-mail del revisore, che abbiamo assunto come identificatore univoco di un generico utente registrato al sistema, come già effettuato alla scorsa iterazione.

|  |  |
| --- | --- |
| **Scenario principale di successo** | 1. Il revisore accede al sistema; 2. Il revisore inserisce e-mail e password; 3. Il sistema controlla la correttezza dei dati inseriti e autentica il revisore; 4. Il revisore seleziona la sezione “articoli”; 5. Il sistema visualizza la lista di articoli da valutare; 6. Il revisore seleziona l’articolo da valutare; 7. Il sistema visualizza l’articolo e mostra l’interfaccia per valutarlo; 8. Il revisore inserisce la valutazione e conferma; 9. Il sistema controlla che tutti i campi siano stati inseriti correttamente; 10. Il sistema salva la valutazione inserita dal revisore; 11. Il sistema visualizza a schermo un messaggio per far capire al revisore che tutto è andato a buon fine. |

Un’estensione che dobbiamo aggiungere riguarda i revisori junior (questa distinzione verrà approfondita subito dopo nel documento).

|  |  |
| --- | --- |
| **Estensioni** | e. Il revisore junior inserisce un voto non compreso fra 0 e 9:   * 1. Il sistema visualizza un errore esplicativo, dove dice di inserire un voto fra 0 e 9;   2. Il revisore junior reinserisce i voti. |

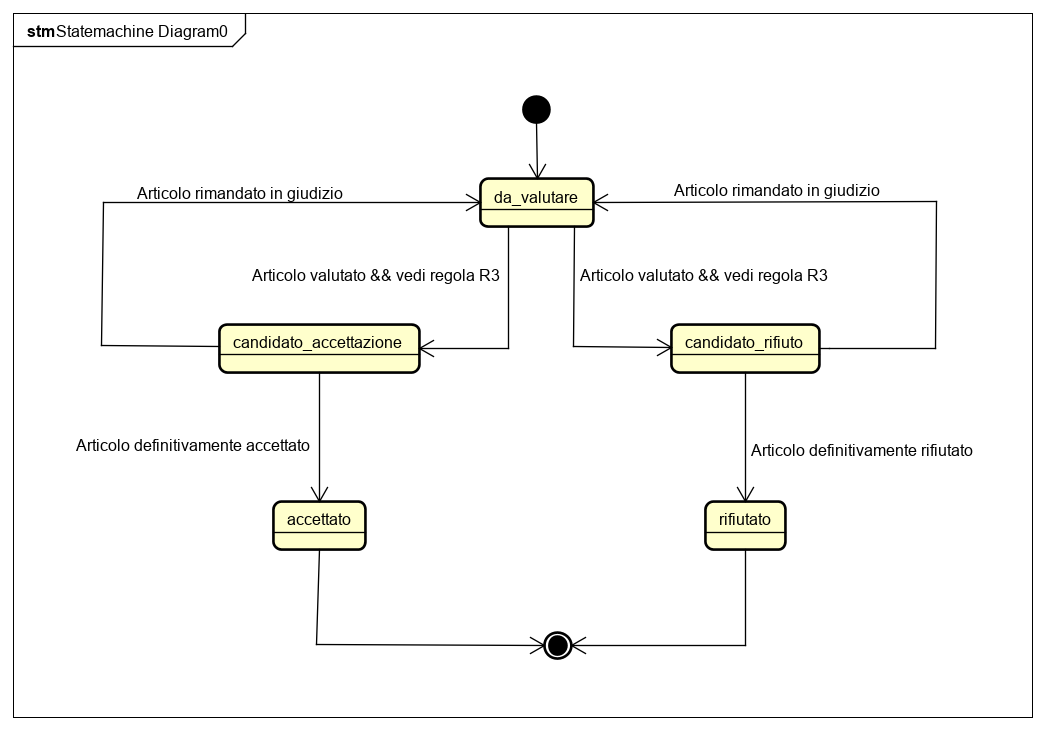
## Modello di dominio

Alla scorsa iterazione abbiamo individuato diverse classi concettuali. Questo modello va sicuramente ampliato in questa fase, per far fronte ai nuovi concetti che emergono dall’analisi dello scenario principale di successo del caso d’uso UC4. Per far ciò iniziamo col riprendere un concetto che avevamo introdotto solamente in maniera generica alla scorsa iterazione, ovvero una gerarchia della classe Revisore. L’editoria, infatti, distingue fra revisori senior e junior. Un revisore senior differisce da uno junior in quanto il tipo di valutazione che va ad attribuire ad un articolo è di natura diversa: il revisore senior attribuisce infatti un giudizio e un commento in forma libera, mentre il revisore junior attribuisce dei voti numerici sulla base di quattro diversi aspetti. Entrambi devono pertanto creare istanze di Valutazione (che si evidenzia pertanto come classe concettuale). Questa classe tuttavia va specializzata in ValutazioneSenior e ValutazioneJunior, viste le intrinseche differenze fra le due Le Valutazioni vengono associate all’articolo: ogni articolo conterrà pertanto una collezione di Valutazioni.

### *Diagramma di macchina a stati*

Analizziamo adesso in modo più dettagliato la classe Articolo. Dal punto di vista concettuale un articolo subisce varie fasi durante il suo ciclo di vita: viene inizialmente caricato e associato ad un revisore; una volta valutato viene candidato all’accettazione o al rifiuto; infine può essere accettato definitivamente, rifiutato definitivamente o rimandato in giudizio, tornando di fatto ad uno stato di articolo ancora da valutare. Per esprimere questa variabilità la classe concettuale Articolo ha bisogno di un attributo che ne rappresenti lo stato. Ci sembra opportuno mostrare la dinamicità della classe attraverso un diagramma di macchina a stati, che esplichi meglio come si susseguono le varie fasi del ciclo di vita di un articolo.

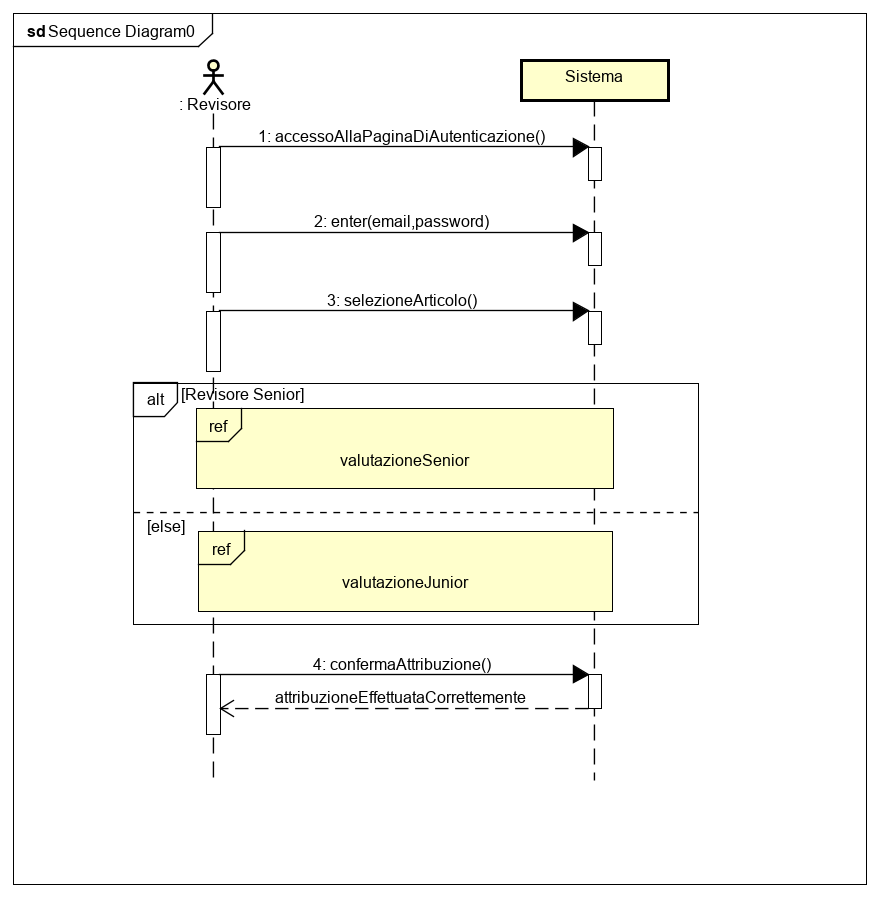
Lo stato iniziale è rappresentato da “da\_valutare”, ed indica che l’articolo è stato sottomesso, sono stati ad esso assegnati dei revisori, ma è ancora privo di valutazioni (da parte di almeno uno dei revisori assegnatigli). Le transizioni da questo stato sono verso due stati possibili: “candidato\_accettazione” e “candidato\_rifiuto”. Queste transizioni sono mutuamente esclusive, e avvengono in corrispondenza dell’evento di valutazione dell’articolo (ovvero quando tutti i revisori ad esso assegnati hanno attribuito la loro valutazione) e sulla base della regola di business R3 definita in fase di Ideazione. Dallo stato “candidato\_accettazione” si possono avere altre due transizioni: una è verso lo stato “accettato”, che si configura come stato finale, l’altra verso lo stato “da\_valutare”, che indica che l’articolo va nuovamente giudicato. Queste transizioni dipendono dalla decisione finale dell’editore. Similmente, da “candidato\_rifiuto” si può avere transizione verso lo stato “rifiutato” (stato finale) o quello “da\_valutare”.

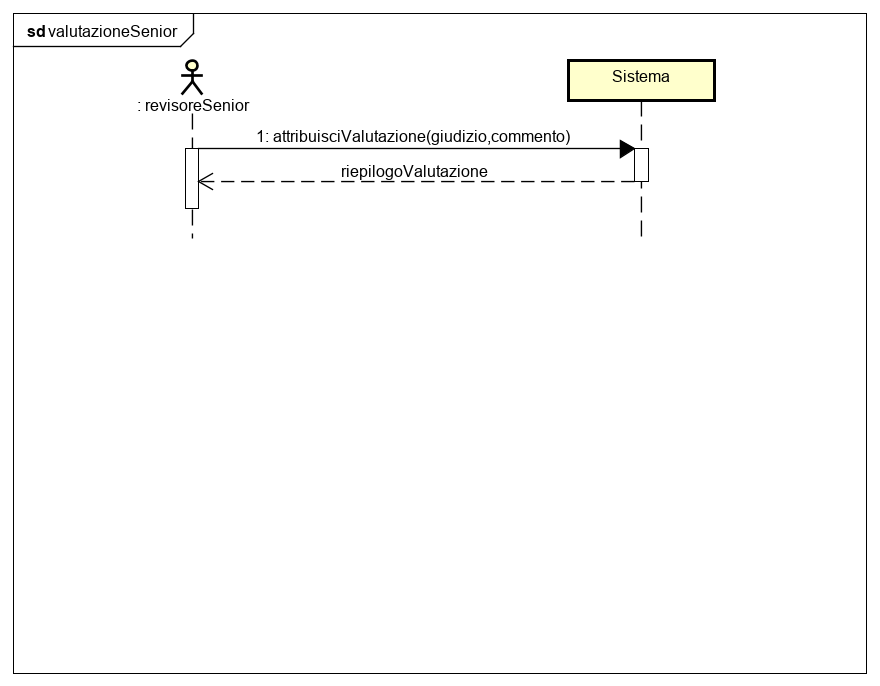


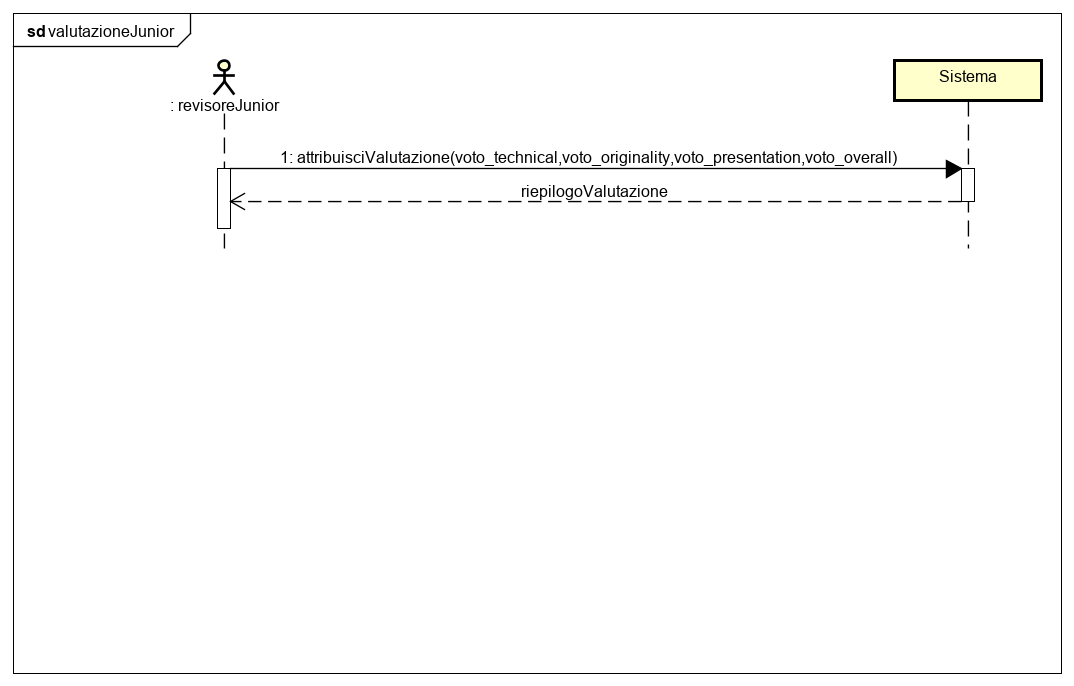
Alla luce di tutte le considerazione fatte precedentemente, è stato elaborato il Modello di Dominio che si trova nel file “ModelloDominio\_02.png” nella cartella Immagini all’interno della cartella dell’iterazione corrente.

## Diagrammi di sequenza di sistema

Presentiamo adesso l’SSD per questo caso d’uso. Anche in questo caso non abbiamo sistemi esterni, ma solo interazioni fra il revisore (attore principale) e il sistema. La valutazione espressa sarà in forma diversa per revisore senior e junior, quindi abbiamo utilizzato un frame mutuamente esclusivo per differenziare le due diverse attribuzione di valutazione.







## Contratti delle operazioni

Segue il contratto dell’operazione significativa che si evince dal diagramma di sequenza di sistema, ovvero l’attribuzione della valutazione.

**Contratto CO4: attribuisciValutazione**

|  |  |
| --- | --- |
| Operazione | attribuisciValutazione: revisore: Revisore |
| Riferimenti | Caso d’uso: Attribuzione Valutazione |
| Pre-condizioni | Il revisore è autenticato e ha selezionato l’articolo da valutare fra quegli assegnatigli; è stata creata un’istanza valutazione corrente di Valutazione. |
| Post-condizioni | Sono stati aggiunti i dati inseriti dal revisore all’istanza di valutazione corrente. |

# Progettazione orientata agli oggetti

Passiamo adesso, anche per questa iterazione, alla fase di progettazione, nella quale dal Modello di Dominio traiamo fuori delle classi software da poter poi implementare.

## Diagrammi di sequenza

Partiamo con l’analizzare i diagrammi di interazione di questa iterazione. In particolare ci avvarremo anche in questa iterazione di un diagramma di sequenza, che si trova nel file “SD\_addValutazione.png” nella cartella Immagini dell’iterazione corrente. Il diagramma riguarda l’attribuzione di una valutazione da parte di un revisore senior o junior. Viene creata un’istanza di valutazione corrente. L’*expert* delle valutazioni è l’articolo: sarà lui a tenere traccia delle valutazioni che gli sono state assegnate. Il sistema, al termine delle operazioni, mantiene una traccia anche per l’attribuzione. Lo stato dell’articolo, una volta aggiunta la valutazione, si aggiorna: qualora ci siano ancora altre valutazioni da effettuare (ad esempio sono stati sorteggiati tre revisori junior, per un totale di quattro valutazioni, ma ne sono state effettuate solamente due) lo stato rimane “da\_valutare”. Se invece non ci sono altre revisioni da effettuare lo stato viene aggiornato secondo la regola di business R3, passando a “candidato\_accettazione” o “candidato\_rifiuto”. In tal caso l’articolo viene aggiunto da TSR ad una lista ArticoliCandidati: questa scelta è dovuta ad una maggiore semplicità nell’implementazione dei casi d’uso UC6 e UC7, oltre a renderne più efficiente l’esecuzione. L’articolo viene rimosso dalla lista di articoli associati al revisore.

## Diagramma delle classi

Alla luce di quanto detto finora, è stato elaborato il diagramma delle classi “DCD\_it2\_1.0.png”, che si trova nella cartella immagini dell’iterazione corrente.

Un RevisoreSenior creerà istanze di ValutazioneSenior, un RevisoreJunior di ValutazioneJunior. Per rappresentare questa natura polimorfica del metodo di attribuzione di una revisione, la nostra scelta implementativa consiste nell’inserimento in Revisore di un *Factory Method* getValutazione(). Polimorficamente questo metodo creerà istanze dell’una o dell’altra valutazione a seconda del tipo concreto di revisore. Durante la prima iterazione abbiamo scelto di far mantenere a TSR due liste separate per revisori senior e junior: questo perché in fase di accettazione è opportuno che il sistema conosca subito la tipologia di ogni revisore (utilizzare un operatore instanceof in fase implementativa ci sembrava poco riusabile, robusto ed efficiente); per le valutazioni invece non vi è necessità di mantenerle in liste separate. Pertanto l’articolo mantiene una sola lista di tipo Valutazione, che verrà riempita con istantze di ValutazioneSerior e ValutazioneJunior a seconda del tipo di revisore corrente.

Guardiamo adesso alla classe Articolo. Per rappresentare lo stato è stata ponderata l’idea di definire una classe astratta StatoArticolo, da estendere con i cinque stati che esso può assumere, e di usare una composizione per esprimere questo concetto. Tuttavia alla fine si è scelto di rappresentarlo solo come una stringa. Questa decisione è stata presa a seguito della seguente riflessione: il pattern *State* è utile quando si vuol fare variare polimorficamente un determinato comportamento di un oggetto al variare di un suo stato interno, richiamando in questo caso un metodo astratto dell’oggetto StatoArticolo che al variare dell’istanza concreta dello stato avrebbe avuto un comportamento differente. Tuttavia, nel nostro caso l’articolo è un’entità passiva: non è lui a compiere azioni, ma sono gli altri attori ad implementare metodi che lo influenzano, È l’autore a sottometterlo, sono i revisori ad attribuire ad esso una valutazione, è l’editore, come vedremo nella prossima iterazione, a prendere una decisione finale. La variazione dello stato di Articolo è solo un riflesso del comportamento di altri che nulla ha a che vedere con eventuali modifiche del suo comportamento. Pertanto non ci è sembrato necessario definire StatoArticolo come una classe concettuale vera e propria, e abbiamo ritenuto opportuno considerarlo solamente come un attributo semplice.

# Implementazione

Per l’implementazione, oltre alle varie classi e ai metodi presentati, è stata implementata anche l’estensione al principale scenario di successo del caso d’uso UC4. Oltre a questo, è stato implementato anche il caso d’uso UC2, relativo alla registrazione del revisore, similmente a quanto fatto durante l’iterazione 1 per il caso d’uso UC1. Il revisore, all’atto della registrazione, sceglie se registrarsi al sistema come revisore senior o junior.