UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CORSO DI INGEGNERIA DEL SOFTWARE
PROF. S. RUSSO - A.A. 2020 - 21

Progetto

APPLICAZIONE WEB AGENZIA TELEVISIVA

Gruppo: TeamBallo

Studenti:

Vito Romano	N46004509	vito.romano2@studenti.unina.it
Stefano Licinio	N46002779	s.licinio@studenti.unina.it
Paolo Russo	N46002315	paolo.russo16@studenti.unina.it

Versione 1 del *14/12/2020*

Indice

1. Requisiti

1.1 Specifiche informali d'utente

Si vuole realizzare una applicazione web per l'azienda VTV, una agenzia televisiva che registra filmati di eventi di attualità e li diffonde verso giornalisti e agenzie di stampa sue clienti.
L'applicazione deve permettere ai cameraman di caricare video sul sistema, ai montatori di creare dei servizi-tv, e ai clienti di ricercare i contenuti e di ricevere notifiche sulle nuove risorse. I cameraman effettuano l'upload di filmati sulla piattaforma che sono caratterizzati da un id, un nome, una data di registrazione, una durata, una dimensione.
I montatori creano servizi-tv, che sono il risultato della composizione di uno, due o più filmati: i servizi-tv possono essere standard oppure premium e sono associati ad un evento di attualità. Gli eventi di attualità, identificati da un nome, da un luogo e da una descrizione, possono essere definiti dai cameraman, in maniera indipendente dall'upload di filmati, oppure dai montatori, durante le operazioni di montaggio, se non esiste sulla piattaforma l'evento d'attualità oggetto di un servizio-tv. Esistono due categorie di utenti, standard e premium: gli utenti standard possono solo visualizzare servizi-tv standard, mentre gli utenti premium possono anche acquistare i servizi-tv premium. Di tutti gli utenti occorre memorizzare nome, cognome, data di nascita, indirizzo e casella email; per gli utenti premium bisogna aggiungere la data di scadenza dell'abbonamento.
Gli utenti possono ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento e possono registrarsi al servizio di notifiche di nuovi upload: i risultati delle ricerche degli utenti standard non devono includere i servizi-tv premium. Per finalità di business intelligence, VTV richiede inoltre che il reparto di marketing ottenga dati sulle visualizzazioni e sugli acquisti dei servizi-tv da parte di utenti standard e premium: per le visualizzazioni degli utenti occorre registrare la data; per gli acquisti dei servizi-tv la data e il numero di licenze d'uso comprate.

1.2 Requisiti software

Con gli incontri avvenuti con il committente si è cercato di disambiguare alcuni aspetti del sistema. Sono stati chiariti i seguenti punti:

• Il cameraman, deve essersi autenticato, mediante username e password, all'interno della piattaforma per poter accedere alle rispettive funzionalità che riguardano il caricamento di un nuovo filmato, sul sistema, composto da un id, un nome, una data di registrazione, una durata, ed una dimensione. Inoltre, può definire un evento di attualità, a partire dai filmati già presenti, come servizio tv.

• Il montatore deve autenticarsi, mediante username e password, all'interno della piattaforma per poter accedere alle rispettive funzionalità come la creazione di un nuovo servizio tv, standard o premium, risultato della composizione di uno, due o più filmati. Il servizio tv sarà caratterizzato da un titolo. Infine può definire un evento di attualità a partire dai servizi già presenti.

• Il cliente, per poter accedere deve autenticarsi sulla piattaforma mediante le proprie credenziali. Esistono due categorie di cliente, standard e premium:

- il cliente standard può solo visualizzare servizi-tv standard. Può ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento ed abilitare il servizio di notifiche di nuovi upload di filmati e servizi tv.

- Il cliente premium può anche acquistare i servizi-tv premium.

• Per finalità di business intelligence, l'addetto marketing può generare un report sulle visualizzazioni da parte dei clienti standard e gli acquisti dei servizi tv da parte degli utenti premium.

I requisiti funzionali definiti all'interno del sistema sono:

1. CaricaFilmato: tramite questa funzionalità, il cameraman effettua l'upload di un nuovo filmato

2. DefinisciEventoDiAttualità: tramite questa funzionalità, il cameraman o il montatore possono definire un evento di attualità a partire dai servizi TV già presenti

3. CreaServizioTv: tramite questa funzionalità, il montatore crea un servizio TV attraverso la composizione di uno o più filmati

4. VisualizzaDati: tramite questa funzionalità, l'addetto del reparto marketing può visualizzare i dati inerenti le visualizzazioni da parte dei clienti standard e, nel caso dei clienti abbonati, anche i dati sui servizi TV acquistati.

5. AbilitaNotifiche: tramite questa funzionalità, i clienti decidono se ricevere una notifica quando viene caricato sulla piattaforma un nuovo filmato o creato un nuovo servizio TV

6. RicercaContenuto: tramite questa funzionalità, i clienti possono ricercare un servizio TV già presente sulla piattaforma

7. VisualizzaServizioTv: tramite questa funzionalità, i clienti dopo aver ricercato un servizio TV possono visualizzarlo

8. AcquistaServizioPremium: tramite questa funzionalità, i clienti premium possono decidere di acquistare un servizio TV premium tra quelli disponibili

9. InviaNotifiche: tramite questa funzionalità, il sistema invia un messaggio con cui avvisa i clienti abilitati che è presente un nuovo servizio TV o filmato all'interno della piattaforma

I requisiti sui dati definiti all'interno del sistema sono:

1. Filmato è identificato da un ID, un nome, una data, una dimensione ed una durata

2. Cameraman è identificato da uno username, una password

3. ServizioTv è definito da un titolo

4. EventoDiAttualità è definito da un nome, un luogo ed una descrizione

5. Montatore è identificato da uno username ed una password

6. Cliente è identificato da un nome, un cognome, una data di nascita, un indirizzo ed una email; qualora il cliente fosse premium si aggiunge la data di scadenza dell'abbonamento

"Codifica dei requisiti funzionali"

RF01	CaricaFilmato
RF02	DefinisciEventoAttualità
RF03	CreaServizioTv
RF04	VisualizzaDati
RF05	AbilitaNotifiche
RF06	RicercaContenuto
RF07	VisualizzaServizioTv
RF08	AcquistaServizioPremium
RF09	InviaNotifche

"Codifica dei requisiti sui dati"

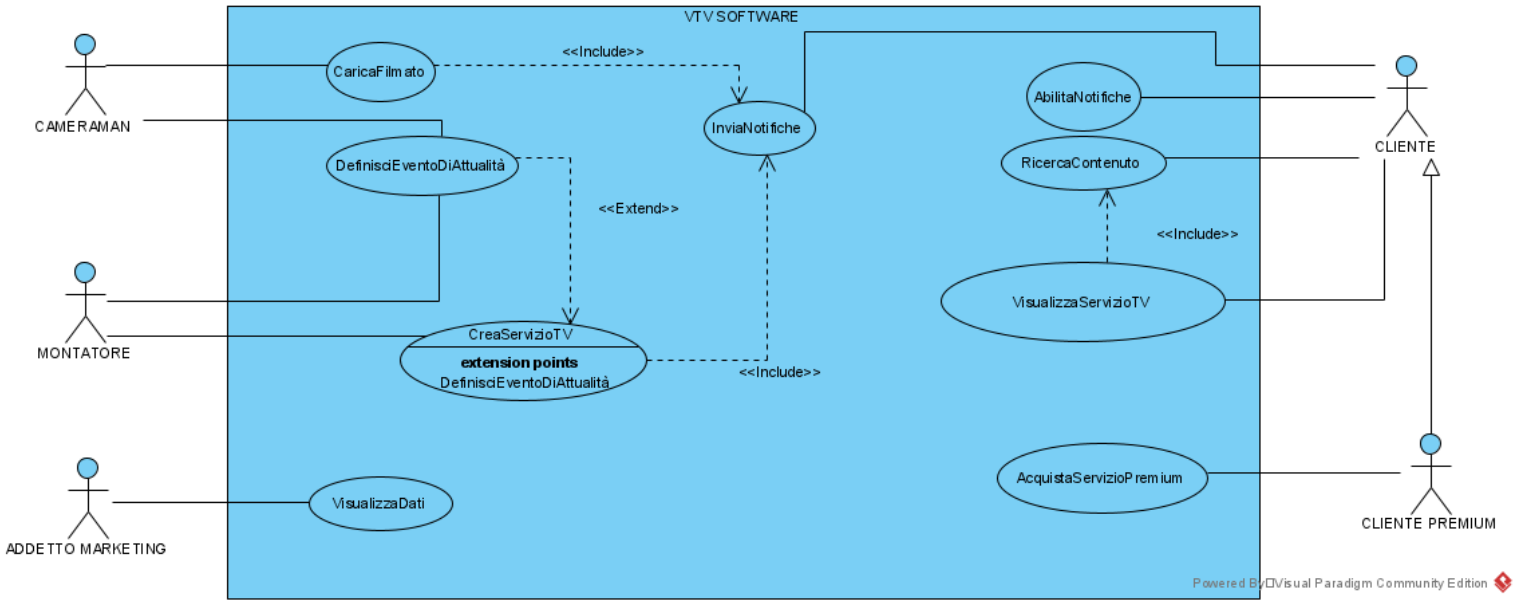
RD01	Filmato	ID, nome, data, dimensione, durata
RD02	Cameraman	Username, password
RD03	ServizioTv	Titolo
RD04	EventoDiAttualità	Nome, luogo, descrizione
RD05	Montatore	Username, password
RD06	Cliente	Nome, cognome, data di nascita, indirizzo, email
RD07	ClientePremium	Nome, cognome, data di nascita, indirizzo, email, data scadenza abbonamento

2. Analisi e specifica dei requisiti

2.1 Modellazione dei casi d'uso

Attore Primario	Attore Secondario	Caso d'uso	Caso d'uso di inclusione	Caso d'uso di estensione
1.Cameraman 2.Montatore 3.Addetto Marketing 4.Cliente 5.Cliente Premium	1.Cliente	1. CaricaFilmato 2. DefinisciEventoDiAttualità 3.CreaServizioTv 4.VisualizzaDati 5.AbilitaNotifiche 6.RicercaContenuto 7.VisualizzaServizioTv 8.AcquistaServizioPremium	1.InviaNotifche 2.RicercaContenuto	1.CreaServizioTv

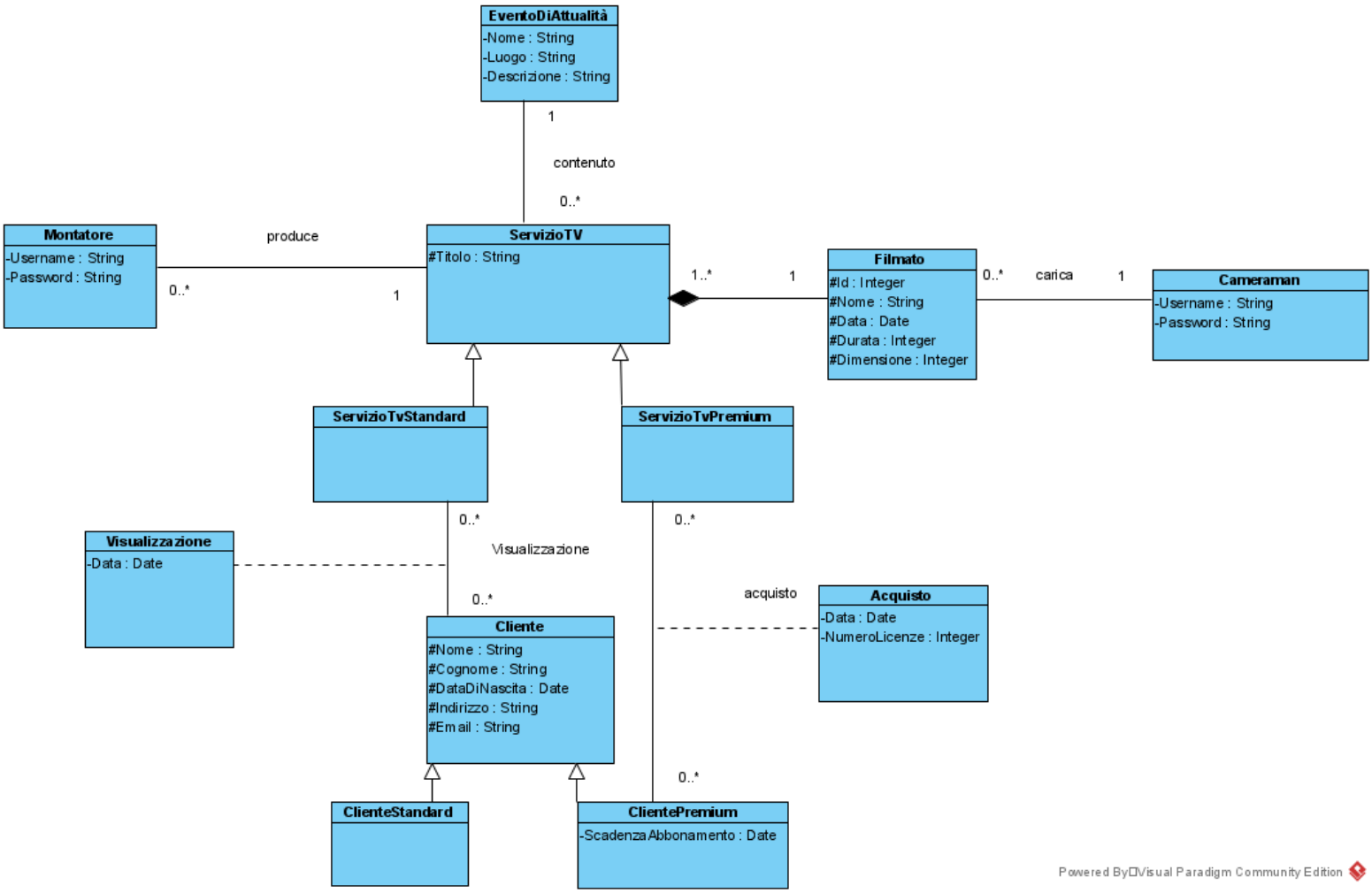
"Diagramma dei Casi d'Uso"

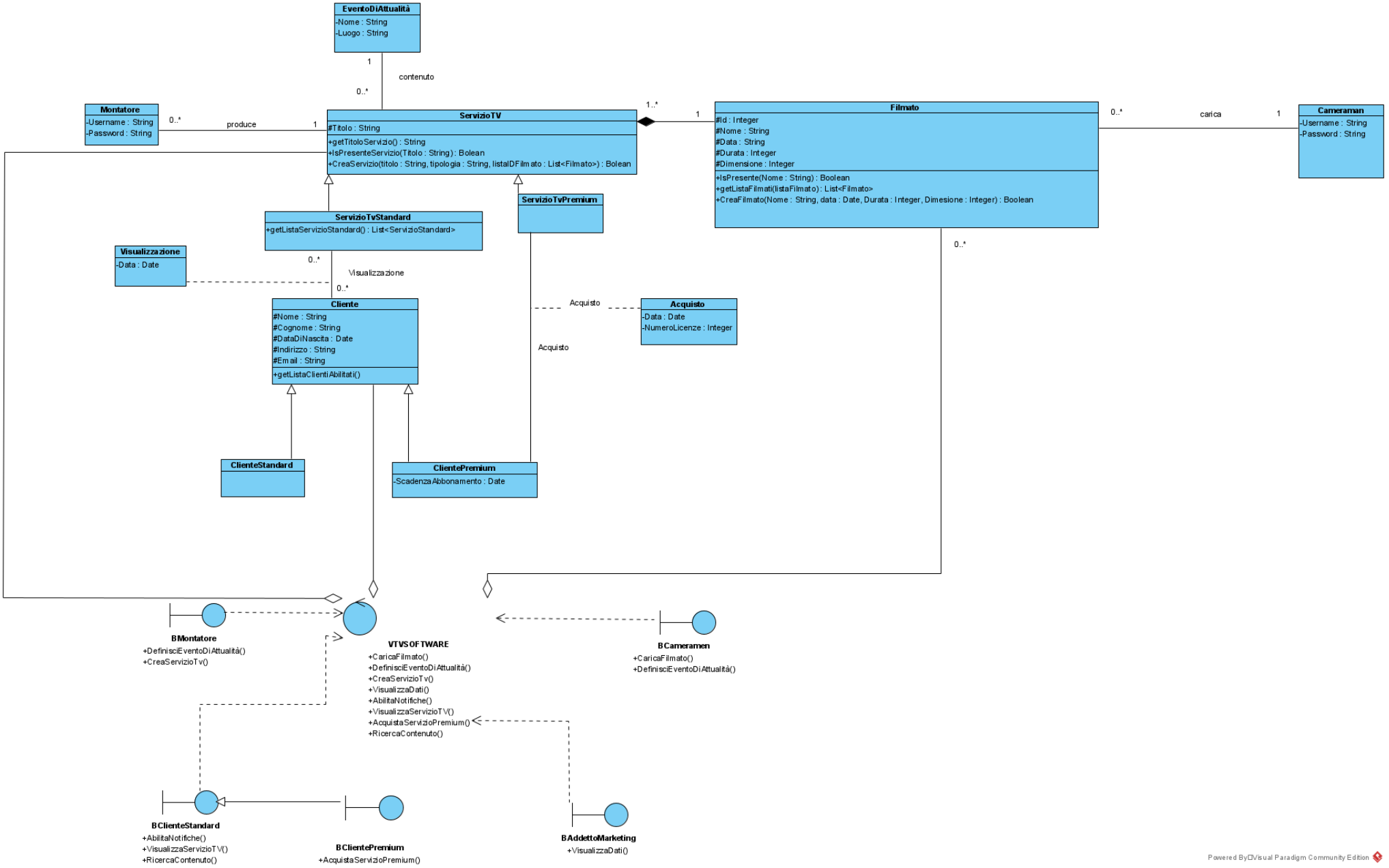


Gli scenari principali legati ai casi d'uso "CaricaFilmato", "RicercaContenuto" e "CreaServizioTv" sono i seguenti:

SCENARI				
Nome del caso d'uso		CaricaFilmato	CreaServizioTv	RicercaContenuto
Identificativo del caso d'uso		1	2	3
Attori coinvolti	**Primari**	Cameraman	Montatore	Cliente
	Secondari			
Descrizione breve		Il cameraman carica un video sulla piattaforma VTV	Il montatore utilizza i video presenti sulla piattaforma per creare un servizio TV (Al termine del caricamento viene informato il servizio notifiche)	Gli utenti possono ricercare servizi TV in base ad una ricerca testuale sul nome dell'evento
Pre-condizioni		1. Il cameraman deve essersi autenticato	1. Il montatore deve essersi autenticato 2. Deve essere presente almeno un filmato caricato sulla piattaforma	1. Il cliente deve essersi autenticato
Sequenza di eventi		1. Il caso d'uso inizia quando il cameraman carica un filmato	1. Il caso d'uso inizia quando il montatore fornisce oltre al titolo del servizio tv ,i nomi dei filmati e la tipoligia (standard/premium). IF non tutti i nomi dei filmati sono presenti oppure esiste un servizio tv con lo stesso titolo sulla piattaforma viene visualizzato un messaggio di errore	1. Il caso d'uso inizia quando il cliente inserisce il servizio TV da ricercare
		2. IF il filmato da caricare ha un nome diverso da tutti i filmati già caricati, allora il caso d'uso termina quando il filmato viene caricato ELSE viene visualizzato un messaggio di errore	2. Il montatore crea il Servizio TV standard o premium 3. Viene inviata la conferma della creazione del ServizioTV standard o premium	2. Viene effettuata una ricerca nella lista dei servizi TV standard
Post-condizioni		1. Il sistema memorizza il filmato	1. Il ServizioTv è stato generato correttamente	1. Vengono visualizzati il/i servizio/i TV corrispondenti

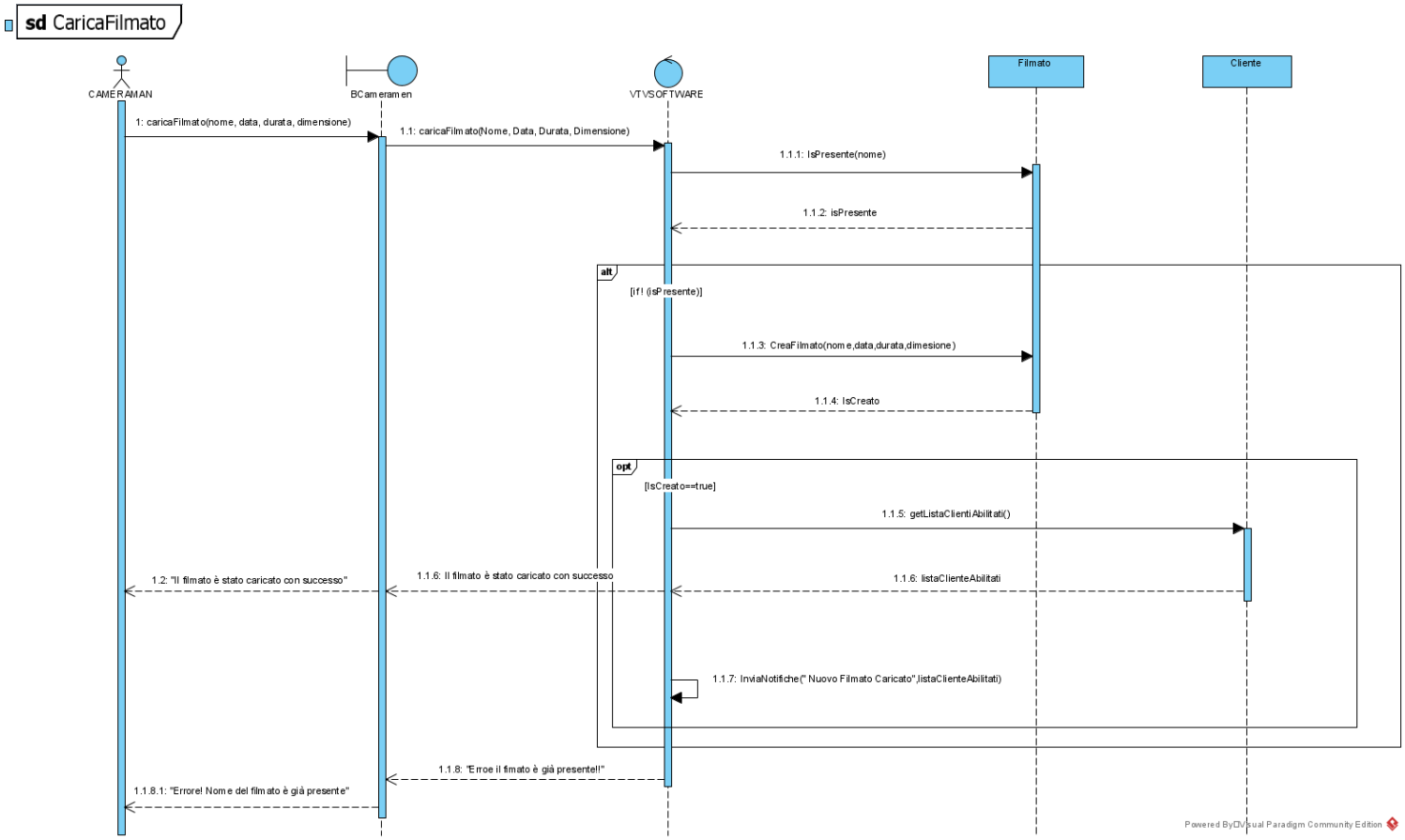
2.2 Diagramma delle classi



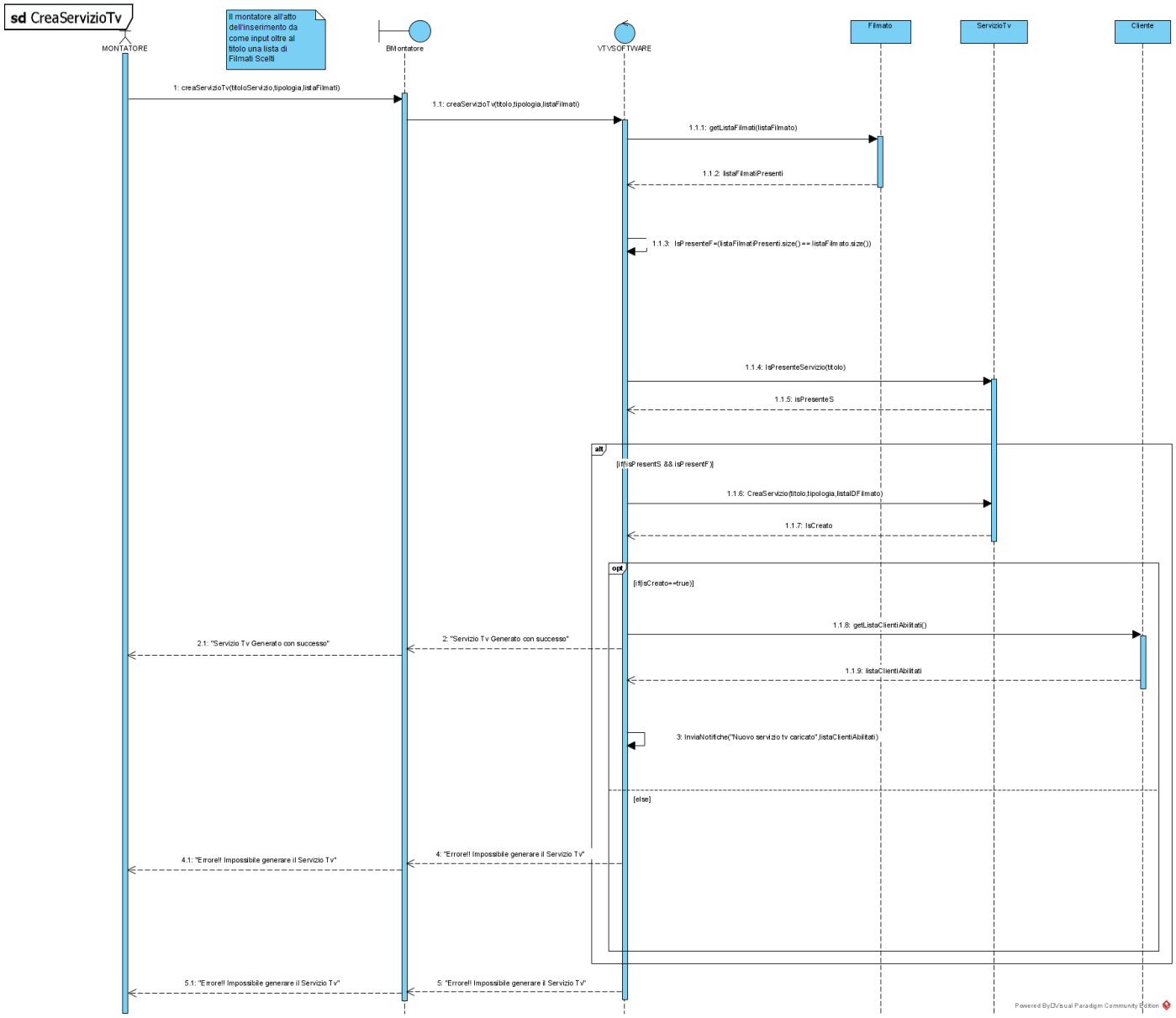

EventoDiAttualità
-Nome : String
-Luogo : String
1
contenuto
0..*
Montatore
-Username : String
-Password : String
0..*
produce
1
ServizioTV
#Titolo : String
+getTitoloServizio() : String
+IsPresenteServizio(Titolo : String) : Bolean
+CreaServizio(titolo : String, tipologia : String, listaIDFilmato : List<Filmato>) : Bolean
1..*
1
Filmato
#Id : Integer
#Nome : String
#Data : String
#Durata : Integer
#Dimensione : Integer
+IsPresente(Nome : String) : Boolean
+getListaFilmati(listaFilmato) : List<Filmato>
+CreaFilmato(Nome : String, data : Date, Durata : Integer, Dimesione : Integer) : Boolean
0..*
carica
1
Cameraman
-Username : String
-Password : String
ServizioTvStandard
+getListaServizioStandard() : List<ServizioStandard>
ServizioTvPremium
Visualizzazione
-Data : Date
0..*
Visualizzazione
0..*
Cliente
#Nome : String
#Cognome : String
#DataDiNascita : Date
#Indirizzo : String
#Email : String
+getListaClientiAbilitati()
Acquisto
Acquisto
Acquisto
-Data : Date
-NumeroLicenze : Integer
0..*
ClienteStandard
ClientePremium
-ScadenzaAbbonamento : Date
BMontatore
+DefinisciEventoDiAttualità()
+CreaServizioTv()
VTVSOFTWARE
+CaricaFilmato()
+DefinisciEventoDiAttualità()
+CreaServizioTv()
+VisualizzaDati()
+AbilitaNotifiche()
+VisualizzaServizioTV()
+AcquistaServizioPremium()
+RicercaContenuto()
BCameramen
+CaricaFilmato()
+DefinisciEventoDiAttualità()
BClienteStandard
+AbilitaNotifiche()
+VisualizzaServizioTV()
+RicercaContenuto()
BClientePremium
+AcquistaServizioPremium()
BAddettoMarketing
+VisualizzaDati()
Powered By Visual Paradigm Community Edition

2.3 Diagrammi di sequenza

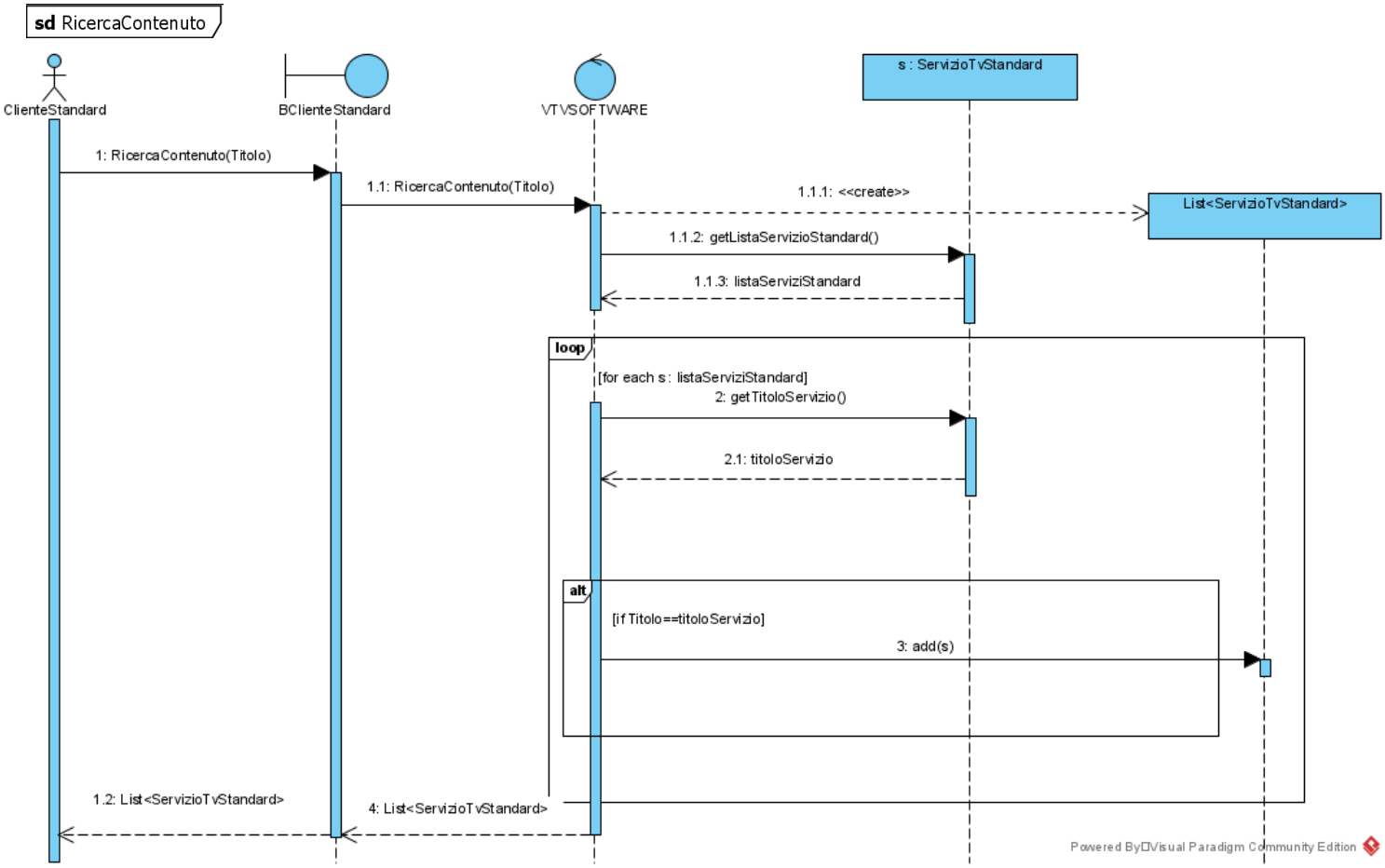
“CaricaFilmato”



“CreaServizioTV”



"*RicercaContenuto*"



3. Stima dei costi

Andiamo ad analizzare la stima dei costi delle funzionalità considerando la seguente tabella:

INDICE		PESI			VPI
	VALORE	SEMPLICE	MEDIO	COMPLESSO	
NIFL	V1	3	4	6	V1*P1
NEIF	V2	4	5	7	V2*P2
NEI	V3	3	4	6	V3*P3
NEO	V4	7	10	15	V4*P4
NEQ	V5	5	7	10	V5 *P5

“CaricaFilmato”

External Input (EI) - Informazioni distinte fornite dall’utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare nome, data, durata e dimensione. Pertanto EI=4;

External Output (EO) – output distinti che il sistema restituisce all’utente come risultato delle proprie elaborazioni. In questo caso, non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. pertanto EO=0;

External Inquiry (EQ) – interrogazioni in linea che producono una risposta immediata del sistema. Pertanto EQ=1;

Internal Logical File (ILF) - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati id, nome, data, durata e dimensione. Pertanto ILF=5;

External Interface File (EIF) - aggregati logici di dati scambiati o condivisi con altre applicazioni. In questo caso specifico EIF=0;

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 4*4 + 5*3 + 1*5 = 36

Il numero di linee codice stimato se l’applicazione
è sviluppata in **C: 36* 128 = 4608**
Se si adotta **Java: 36*6 = 216**

FATTORI CORRETTIVI

PRESTAZIONI – Misura gli obbiettivi prestazionali dell'applicazione, richiesti o approvati dall'utente, sia in termine di risposta che di elaborazioni. Nel nostro caso il valore assegnato è 0: l'utente non ha specificato particolari requisiti prestazionali.

EFIICIENZA PER L'UTENTE FINALE – Esprime il grado di considerazione per i fattori umani e la facilità di utilizzo pe l'utente dell'applicazione misurata. Nel nostro caso il valore assegnato è 0: non sono state previste particolari tecniche legate all'userfriendliness.

RIUSABILITÀ – L'applicazione e il codice dell'applicazione sono appositamente progettati per essere utilizzati in altre applicazioni. Nel nostro caso il valore assegnato è 1: viene usato del codice riusabile nell'applicazione.

STIMA FINALE

$$FP = UFP * (0{,}65 + 0{,}01 * \sum_{i=1}^{3} F_i)$$

Dove la sommatoria è uguale a 0+0+1 = 1, per cui i nostri FP saranno uguali a:

$$FP = 36 * (0{,}65 + 0{,}01) = 23{,}76 \approx 24$$

Il numero di linee codice stimato se l'applicazione
è sviluppata in **C: 24* 128 = 3072**
Se si adotta **Java: 24*6 = 144**

“CreaServizioTV”

External Input (EI) - Informazioni distinte fornite dall’utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare il nome del Servizio TV da creare ed una lista di nomi che riguardano il filmato usato per creare il Servizio TV, pertanto EI = 2.

External Output (EO) – output distinti che il sistema restituisce all’utente come risultato delle proprie elaborazioni. In questo caso, non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. pertanto EO=0.

External Inquiry (EQ) – interrogazioni in linea che producono una risposta immediata del sistema. Pertanto EQ=2;

Internal Logical File (ILF) - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati Id, nome, data, durata e dimensione. Pertanto ILF=5;

External Interface File (EIF) - aggregati logici di dati scambiati o condivisi con altre applicazioni. In questo caso specifico EIF=0;

I valori precedentemente conteggiati sono pesati secondo la tabella seguente:

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 2*4 + 5*3 + 2*5 = 33

Il numero di linee codice stimato se l’applicazione
è sviluppata in **C: 33* 128 = 4224.**
Se si adotta **Java: 33*6 = 198.**

FATTORI CORRETTIVI

FACILITA’ DI INSTALLAZIONE – 4, poiché il software deve essere facilmente installabile, sia dai clienti che dagli operatori dell’emittente;

PRESTAZIONI - 3 è importante che il software sia reattivo durante l’arco di tutta la giornata, senza però le necessità imposte da un sistema hard real time;

EFFICIENZA PER L’UTENZA FINALE -5 è importante che il software disponga di tutti gli strumenti atti a renderne l’utilizzo quanto più user friendly possibile;

RIUSABILITA’ – 2, con qualche modifica, potrebbero essere usati in altre applicazioni i moduli da noi implementati in questo progetto;

STIMA FINALE

$$FP = UFP * (0{,}65 + 0{,}01 * \sum_{i=1}^{4} F_i)$$

Dove la sommatoria è uguale a 4+3+5+2 = 14, per cui i nostri FP saranno uguali a:

$$FP = 33 * (0{,}65 + 0{,}14) = 26{,}07 \approx 26$$

Il numero di linee codice stimato se l'applicazione
è sviluppata in **C: 26* 128 = 3328**
Se si adotta **Java: 26*6 = 156**

"*RicercaContenuto*"

External Input (EI) - Informazioni distinte fornite dall'utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare Nome dell'evento, pertanto EI=1;

External Output (EO) – output distinti che il sistema restituisce all'utente come risultato delle proprie elaborazioni. In questo caso, poiché vado a ricercare un contenuto nel mio Db, non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. Pertanto EO=0;

External Inquiry (EQ) – interrogazioni in linea che producono una risposta immediata del sistema. In questo caso vado ad effettuare una ricerca attraverso una query nel mio DB. Pertanto EQ=1;

Internal Logical File (ILF) - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati titolo. Pertanto ILF=1;

External Interface File (EIF) - aggregati logici di dati scambiati o condivisi con altre applicazioni. In questo caso specifico EIF=0;

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 1*4 + 1*5+ 1*3 = 12

Il numero di linee codice stimato se l'applicazione
è sviluppata in **C: 12 * 128 = 1536**
Se si adotta **Java: 12 * 6 = 72**

FATTORI CORRETTIVI

PRESTAZIONI – Misura gli obbiettivi prestazionali dell'applicazione, offerti all'utente, sia in termine di risposta che di elaborazioni. Nel nostro caso il valore assegnato è 4: in quanto all'atto della ricerca l'utente si aspetta una risposta immediata.

EFIICIENZA PER L'UTENTE FINALE – Esprime il grado di considerazione per i fattori umani e la facilità di utilizzo pe l'utente dell'applicazione misurata. Nel nostro caso il valore assegnato è 4.

RIUSABILITÀ – L'applicazione e il codice dell'applicazione sono appositamente progettati per essere utilizzati in altre applicazioni. Nel nostro caso il valore assegnato è 3: viene usato del codice riusabile nell'applicazione.

STIMA FINALE

$$FP = UFP * (0{,}65 + 0{,}01 * \sum_{i=1}^{3} F_i)$$

Dove la sommatoria è uguale a 4+4+3= 11, per cui i nostri FP saranno uguali a:

$$FP = 12 * (0{,}65 + 0{,}11) = 9{,}12 \approx 10$$

Il numero di linee codice stimato se l'applicazione
è sviluppata in **C: 10 * 128 = 1280**
Se si adotta **Java: 10 * 6 = 60**

4. Piano di test funzionale

PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI "*CaricaFilmato*".

Nome
• Nome Inesistente • Nome esistente [SINGLE]

Abbiamo 1 categorie con 2classi quindi il Numero di test senza vincoli è pari a (1*2) =2

Numero di test senza vincoli:2

Abbiamo nessun vincolo *error* e 1 vincolo *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1) Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR, ma contando i vincoli PROPERTY Ottengo: 1

2) Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 + 1 = 2 (Ogni vincolo error o single richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

TEST SUITE

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il cameraman carica un filmato non presente nel DataBase.	Nome filmato Inesistente	Il cameraman ha eseguito con successo il login. Il nome del filmato non deve essere presente all'interno del DataBase.	Nome: **Ricordo di Maradona**	**Visualizzato un messaggio che indica: "Il filmato è stato caricato con successo".**	**Filmato correttamente caricato.**			
2	Il cameraman carica un filmato già presente nel DataBase.	Nome filmato esistente	Il cameraman ha eseguito con successo il login. Il filmato con nome "Ricordo di Maradona" è già presente nel DataBase	Nome: **Ricordo di Maradona**	**Visualizzato un messaggio che indica che il filmato è già presente.**	**Il filmato non viene caricato**			

PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI "*RicercaContenuto*".

Nome
• Titolo Inesistente • Titolo esistente [SINGLE]

Abbiamo 1 categorie con 2 classi quindi il Numero di test senza vincoli è pari a (1*2) = 2

Abbiamo nessun vincolo *error* e 1 vincolo *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1)Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR, ma contando i vincoli PROPERTY Ottengo: 1

2)Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 + 1 = 2 (Ogni vincolo error o single richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

TEST SUITE

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il cliente vuole ricercare un Servizio Tv esistente all'interno del DataBase	Titolo ServizioTv esistente	Il cliente ha eseguito con successo il login. Il database deve contenere al proprio interno il servizio Tv ricercato dall'utente	Titolo: **CAPRI**	Lista servizi tv ricercati in base al titolo	Il sistema non apporta modifiche al database			
2	Il cliente vuole ricercare un Servizio Tv inesistente all'interno del DataBase	Titolo ServizioTv non esistente	Il cliente ha eseguito con successo il login. Il database non deve contenere al proprio interno il servizio Tv ricercato dall'utente	Titolo: **VITO**	La lista servizi tv risulterà vuota	Il sistema non apporta modifiche al database			

PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI "*CreaServizioTV*".

Titolo	ListaFilmati
• Titolo inesistente • Titolo esistente [SINGLE]	• Lista Filmati esistente • Lista Filmati inesistente [SINGLE]

Abbiamo 2 categorie con 2 classi, quindi il numero di test senza vincoli è pari a (2*2) =4

Numero di test senza vincoli: 4
Abbiamo 2 vincoli *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1)Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR, ma contando i vincoli PROPERTY Ottengo: 1*1=1

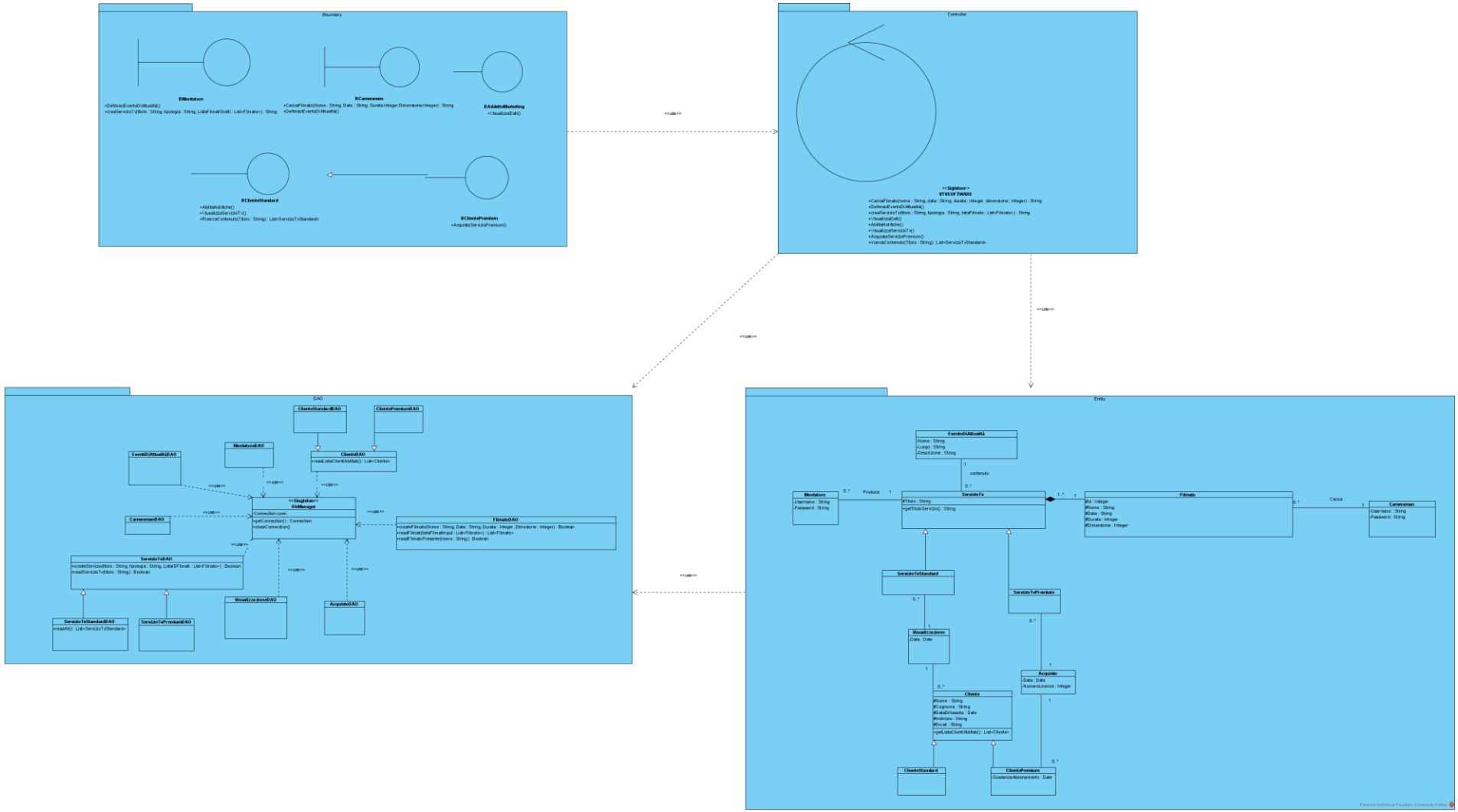
2)Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 + 2= 3 (Ogni vincolo error o single richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

TEST SUITE

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il montatore crea un Servizio Tv non esistente nel DataBase.	Titolo inesistente Lista Filmati esistente	Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all'interno del DataBase, mentre i filmati scelti sono presenti.	Nome: **Il dio del calcio** Lista Filmati: Filmato1	**Visualizzato un messaggio che indica che il Servizio TV è stato creato con successo**	**Servizio TV correttamente creato**			
2	Il montatore crea un Servizio Tv, che è già esistente nel DataBase.	Titolo esistente Lista Filmati esistente	Il montatore si è autenticato con successo. Egli vuole creare un servizio Tv, che risulta già presente all'interno del DataBase.	Nome: **Servizio_tv** Lista Filmati: **Filmato1**	**Visualizzato un messaggio che indica la mancata creazione del Servizio TV.**	**Servizio TV non creato**			
3	Il montatore crea un Servizio Tv, in cui il titolo non è presente all'interno del Database, mentre, i filmati definiti non sono stati caricati nel DB.	Titolo inesistente Lista Filmati inesistente	Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all'interno del DataBase, così come i filmati scelti.	Nome: **Servizio_tv** Lista Filmati: **Filmato1**	**Visualizzato messaggio di errore.**	**Servizio TV non creato**			

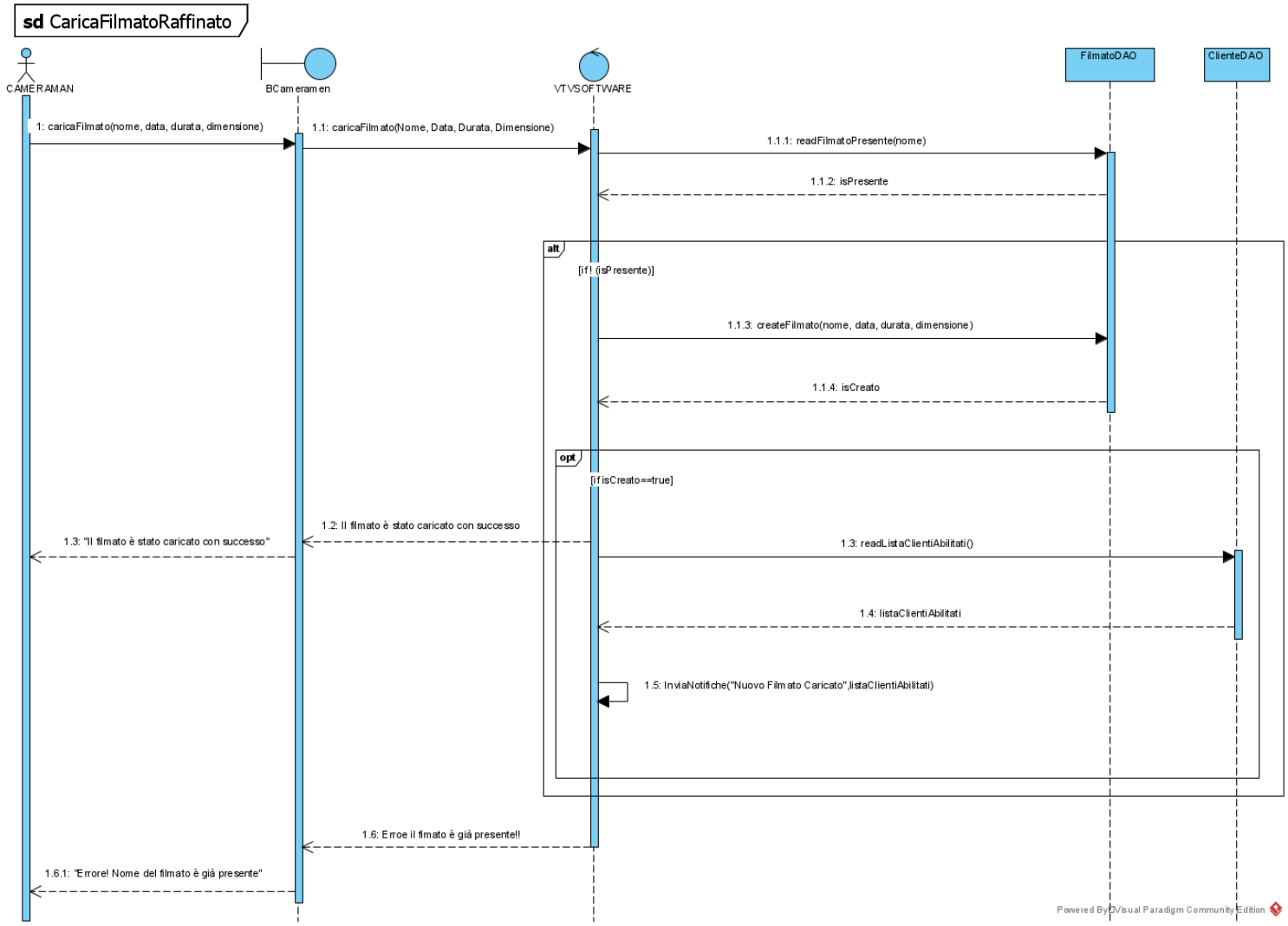
5. Progettazione

5.1 Diagramma delle classi

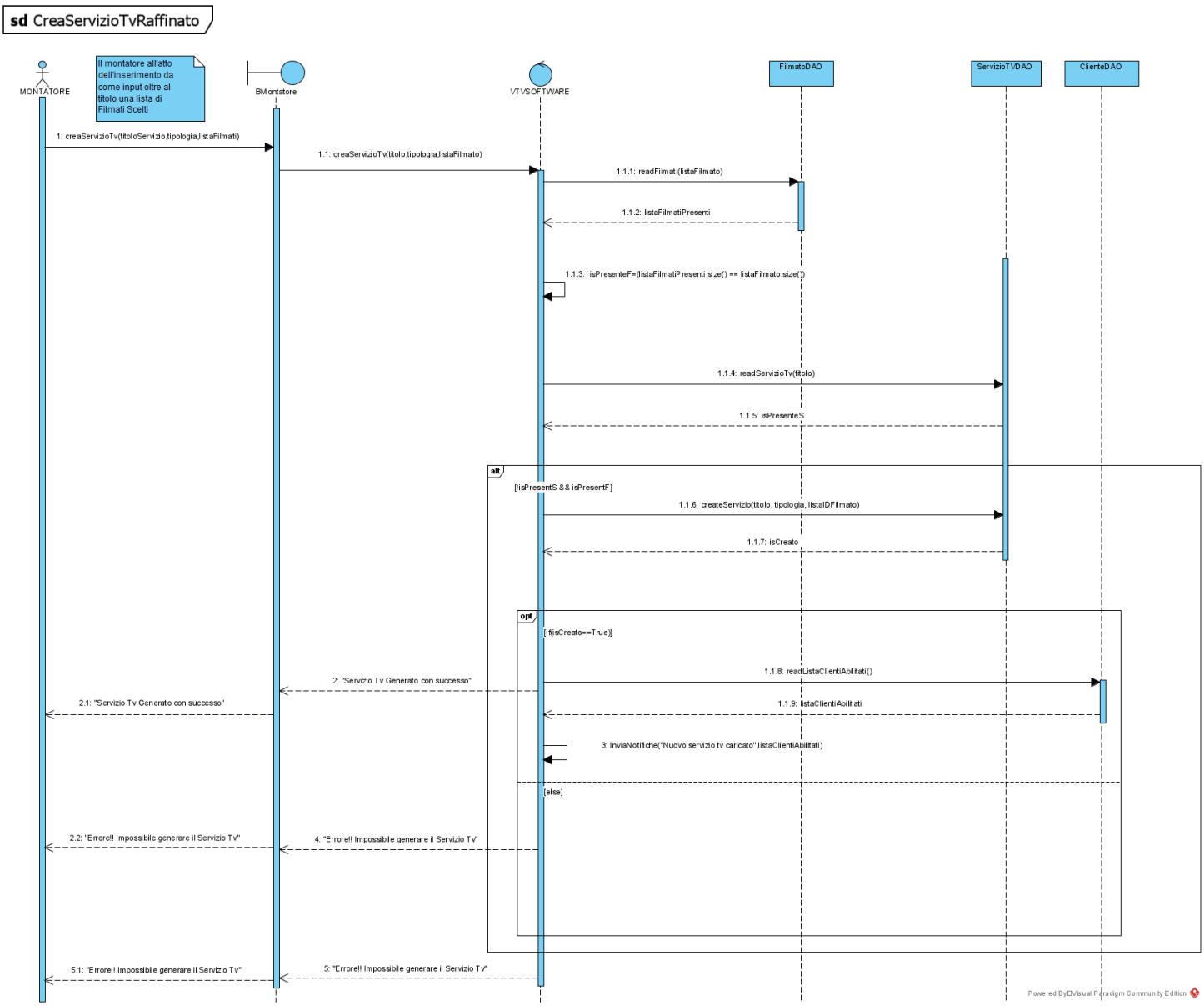


5.2 Diagrammi di sequenza

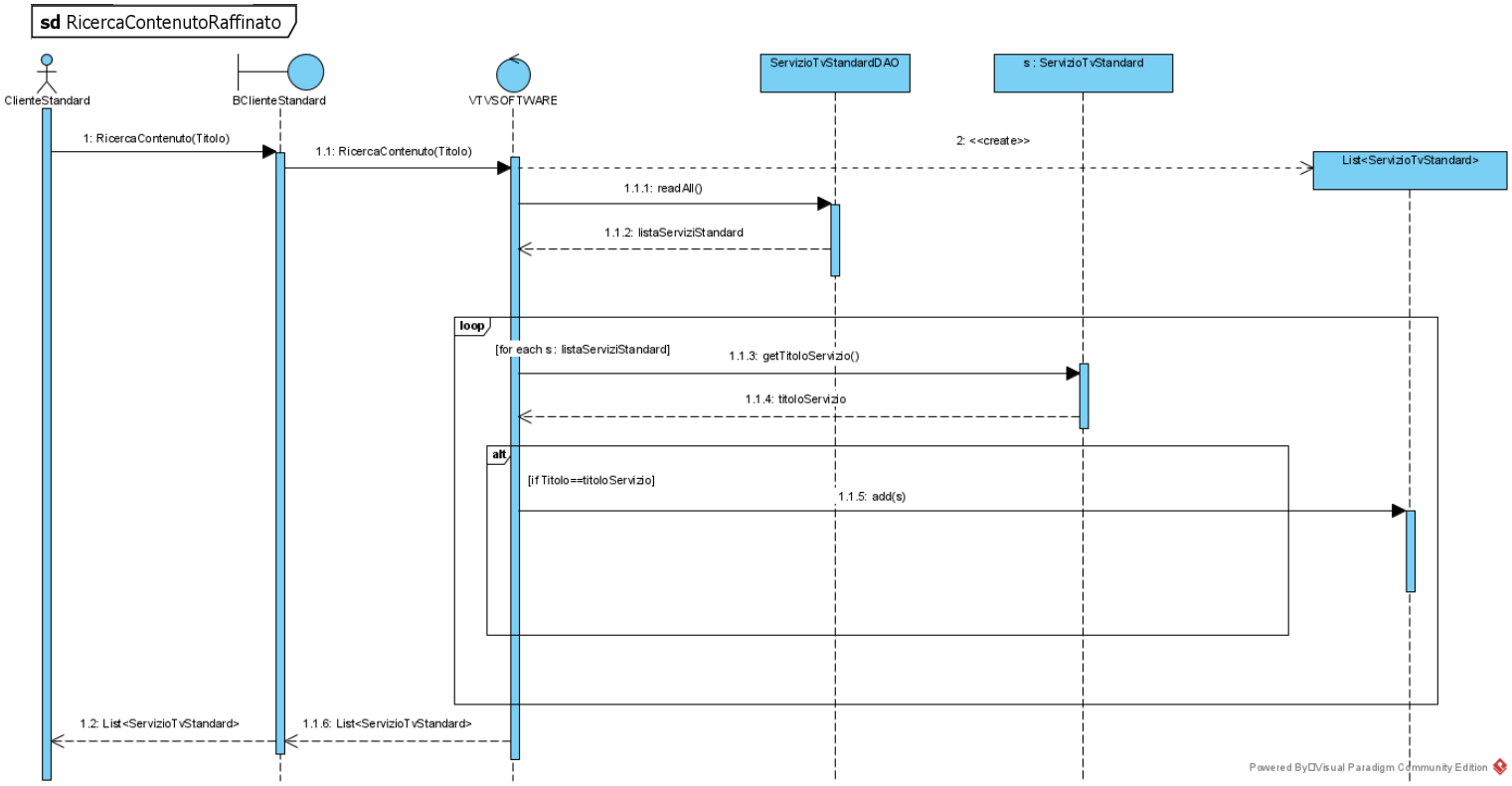
“CaricaFilmato”



“CreaServizioTV”



“*RicercaContenuto*”



6. Implementazione

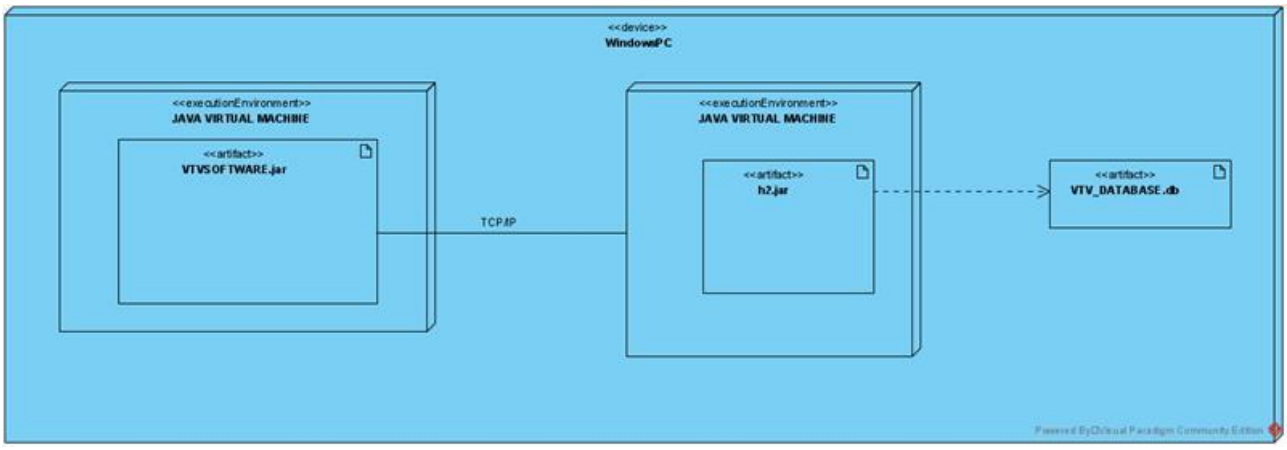
Verranno adesso presentati gli artefatti di codifica:

- Package, Classi, Eccezioni e Test
  1. Boundary
     - BCameramen
     - BClienteStandard
     - BMontatore
  2. Controller
     - VTVSOFTWARE
  3. DAO
     - ClienteDAO
     - ClientePremiumDAO
     - ClienteStandardDAO
     - DbManager
     - FilmatoDAO
     - ServizioTvDAO
     - ServizioTvPremiumDAO
     - ServizioTvStandardDAO
  4. Entity
     - Cliente
     - ClientePremium
     - ClienteStandard
     - Filmato
     - ServizioTv
     - ServizioTvPremium
     - ServizioTvStandard
  5. Exception
     - DAOException
     - VTVException
  6. Test
     - CaricamentoFilmato1Test
     - CaricamentoFilmato2Test
     - RicercaContenutoTest
     - RicercaContenutoTest1
     - Test_CreazioneServizioTV
     - Test_FilmatoEsistenteServizioEsistente
     - Test_ServzioEsistenteFilmatoInesistente

- Artefatti necessari
  1. JRE Systeme Library [JDK]
  2. JUnit 4
  3. DB H2 -1.4.199

- Documentazione Javadoc:
  - JAVADOC

- Diagramma di deployment


<<device>>
WindowsPC
<<executionEnvironment>>
JAVA VIRTUAL MACHINE
<<artifact>>
VTVSOFTWARE.jar
TCP/IP
<<executionEnvironment>>
JAVA VIRTUAL MACHINE
<<artifact>>
h2.jar
<<artifact>>
VTV_DATABASE.db

7. Testing

7.1 Test funzionale

“CaricaFilmato”

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il cameraman carica un filmato non presente nel DataBase.	Nome filmato Inesistente	Il cameraman ha eseguito con successo il login. Il nome del filmato non deve essere presente all’interno del DataBase.	Nome: **Ricordo di Maradona**	**Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.**	**Filmato correttamente caricato.**	**Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.**	**Filmato correttamente caricato.**	**PASS**
2	Il cameraman carica un filmato già presente nel DataBase.	Nome filmato esistente	Il cameraman ha eseguito con successo il login. Il filmato con nome “Ricordo di Maradona” è già presente nel DataBase	Nome: **Ricordo di Maradona**	**Visualizzato un messaggio che indica che il filmato è già presente.**	**Il filmato non viene caricato**	**Visualizzato un messaggio che indica: “Erroe il fimato è già presente!!**	**Il filmato non viene caricato**	**PASS**

"*RicercaContenuto*"

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il cliente vuole ricercare un Servizio Tv esistente all'interno del DataBase	Titolo ServizioTv esistente	Il cliente ha eseguito con successo il login. Il database deve contenere al proprio interno il servizio Tv ricercato dall'utente	Titolo: **CAPRI**	Lista servizi tv ricercati in base al titolo	Il sistema non apporta modifiche al database	Lista servizi tv ricercati in base al titolo	Il sistema non apporta modifiche al database	**PASS**
2	Il cliente vuole ricercare un Servizio Tv inesistente all'interno del DataBase	Titolo ServizioTv non esistente	Il cliente ha eseguito con successo il login. Il database non deve contenere al proprio interno il servizio Tv ricercato dall'utente	Titolo: **VITO**	La lista servizi tv risulterà vuota	Il sistema non apporta modifiche al database	La lista servizi tv risulterà vuota	Il sistema non apporta modifiche al database	**PASS**

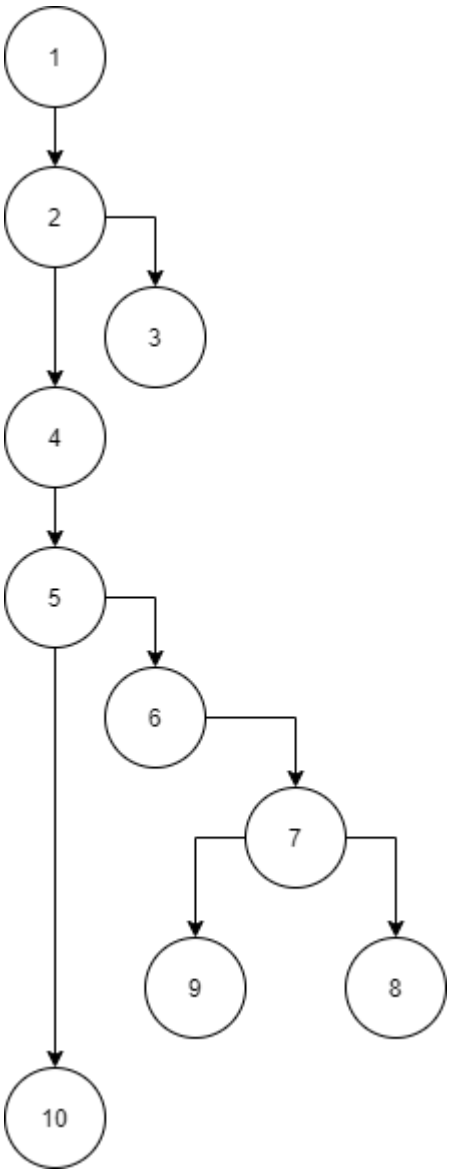
"CreaServizioTv"

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
1	Il montatore crea un Servizio Tv non esistente nel DataBase.	Titolo inesistente Lista Filmati esistente	Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all'interno del DataBase, mentre i filmati scelti sono presenti.	Nome: **Il dio del calcio** Lista Filmati: Filmato1	**Visualizzato un messaggio che indica che il Servizio TV è stato creato con successo**	**Servizio TV correttamente creato**	**Visualizzato un messaggio che indica: "Servizio Tv Generato con successo"**	**Servizio TV correttamente creato**	PASS
2	Il montatore crea un Servizio Tv, che è già esistente nel DataBase.	Titolo esistente Lista Filmati esistente	Il montatore si è autenticato con successo. Egli vuole creare un servizio Tv, che risulta già presente all'interno del DataBase.	Nome: **Servizio_tv** Lista Filmati: **Filmato1**	**Visualizzato un messaggio che indica la mancata creazione del Servizio TV.**	**Servizio TV non creato**	**Visualizzato un messaggio che indica:** "Errore!! Impossibile generare il Servizio Tv**"**	**Servizio TV non creato**	PASS
3	Il montatore crea un Servizio Tv, in cui il titolo non è presente all'interno del Database, mentre, i filmati definiti non sono stati caricati nel DB.	Titolo inesistente Lista Filmati inesistente	Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all'interno del DataBase, così come i filmati scelti.	Nome: **Servizio_tv** Lista Filmati: **Filmato1**	**Visualizzato messaggio di errore.**	**Servizio TV non creato**	**Visualizzato un messaggio che indica:** "Errore!! Impossibile generare il Servizio Tv**"**	**Servizio TV non creato**	PASS

7.2 Test strutturale

7.2.1 Complessità ciclomatica

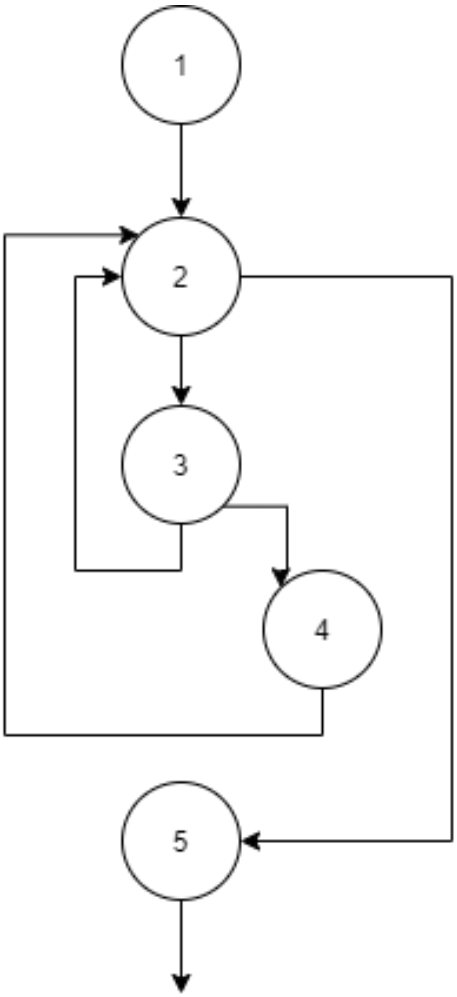
```
        public String creaServizioTv(String titolo, String tipologia, List<Filmato> listaFilmato) throws VTVException {

/*1/             Boolean isCaricato = false;

/*2*/            if (!tipologia.equals("s") && !tipologia.equals("p")) {

/*3*/                    throw new VTVException("Tipologia può essere 's' oppure 'p'");

                }

/*              List<Filmato> listaIdFilmato = filmatoDAO.readID(listaFilmato);
                ServizioTvDAO servizioTvDAO = new ServizioTvDAO();
4
                Boolean isPresentS = servizioTvDAO.readServizioTv(titolo);
*/              Boolean isPresentF = (listaIdFilmato.size() == listaFilmato.size());

/*5*/            if (!isPresentS && isPresentF) {

/*6*/                    isCaricato = servizioTvDAO.createServizio(titolo, tipologia, listaIdFilmato);

/*7*/                    if (isCaricato) {

/*                             listaClientiAbilitati = clienteDAO.readListaClientiAbilitati();
8                              InviaNotifiche("Nuovo Servizio Tv Caricato", listaClientiAbilitati);
*/
                               return "Servizio Tv Generato con successo";

                       } else {

/*9*/                          return "Errore!! Impossibile generare il Servizio Tv";

                       }
        }

/*10*/    return "Errore!! Impossibile generare il Servizio Tv";
        }
```



Cammino 1: 1-2-3
Cammino 2: 1-2-4-5-10
Cammino 3: 1-2-4-5-6-7-9
Cammino 3: 1-2-4-5-6-7-8

I nodi predicato sono 3 ovvero il nodo “2”, il nodo “5” ed il nodo “7”.
Il numero ciclomatico di MC Cabe sarà: 3+1 = 4

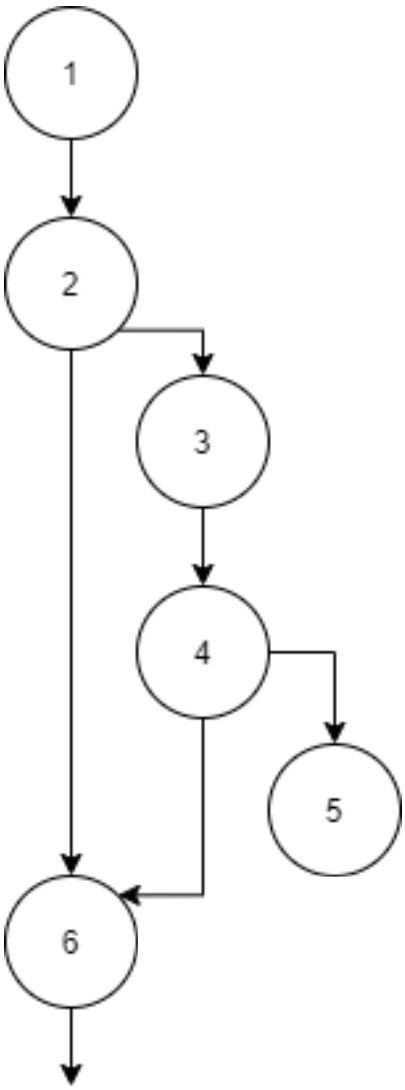
```
public List<ServizioTVStandard> ricercaContenuto (String titoloServizio) {
/*1*/     ArrayList<ServizioTvStandard> listaRisultati = new ArrayList<>();
/*2*/     for (ServizioTvStandard s : listaServiziStandard) {
/*3*/             if (s.getTitoloServizio().contains(titoloServizio)) {
/*4*/                     listaRisultati.add(s);
                    }
          }
/*5*/     retrun listaRisultati;
}
```



Cammino 1: 1-2-3-4-2-5
Cammino 2: 1-2-3-2-5
Cammino 3: 1-2-5

I nodi predicato sono 2 ovvero il nodo “2” ed il nodo “3”.
Il numero ciclomatico di MC Cabe sarà: 2+1 = 3

```
public String caricaFilmato(String nome, String data, Integer durata, Integer dimensione) throws VTVException {

/*1*/     isPresente = filmatoDAO.readFilmatoPresente(nome);

/*2*/     if (!isPresente) {

/*3*/             isCreato = filmatoDAO.createFilmato(nome, data, durata, dimensione);

/*4*/             if (Boolean.TRUE.equals(isCreato)) {

/*                      listaClientiAbilitati = clienteDAO.readListaClientiAbilitati();
5                       InviaNotifiche("Nuovo Filmato Caricato", listaClientiAbilitati);
*/                      return "Il filmato è stato caricato con successo";

                }
        }

/*6*/     return "Erroe il fimato è già presente!!";
}
```



Cammino 1: 1-2-6
Cammino 2: 1-2-3-4-6
Cammino 3: 1-2-3-4-5

I nodi predicato sono 2 ovvero il nodo “2” ed il nodo “4”.
Il numero ciclomatico di MC Cabe sarà: 2+1 = 3

7.2.2 Test di unità

"creaServizioTv"

Cammini	Input	Output atteso
C1	(!tipologia.equals("s") && !tipologia.equals("p"))	throw new VTVException("Tipologia può essere 's' oppure 'p'");
C2	(tipologia.equals("s") || tipologia.equals("p")) ; (isPresentS || !isPresentF)	Errore!! Impossibile generare il Servizio Tv
C3	(tipologia.equals("s") || tipologia.equals("p")) ; (!isPresentS && isPresentF); (!isCaricato)	Errore!! Impossibile generare il Servizio Tv
C4	(tipologia.equals("s") || tipologia.equals("p")) ; (!isPresentS && isPresentF); (isCaricato)	Servizio Tv Generato con successo

"ricercaContenuto"

Cammini	Input	Output atteso
C1	(s.getTitoloServizio().contains(titoloServizio))	return listaRisultati
C2	(!s.getTitoloServizio().contains(titoloServizio))	return listaRisultati
C3	listaServiziTv vuota	return listaRisultati

"caricaFilmato"

Cammini	Input	Output atteso
C1	(isPresente)	"Erroe il fimato è già presente!!"
C2	(!isPresente); (Boolean.***FALSE***.equals(isCreato))	"Erroe il fimato è già presente!!"
C3	(!isPresente); (Boolean.***TRUE***.equals(isCreato))	"Il filmato è stato caricato con successo"

7.3 Test di unità con JUnit

CaricamentoFilmato1Test (16 dic 2020 19:05:42)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		16%		12%	100	118	361	434	72	88	22	29
Total	1.310 of 1.565	16%	51 of 58	12%	100	118	361	434	72	88	22	29

CaricamentoFilmato2Test (16 dic 2020 19:14:04)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		9%		6%	105	118	389	434	76	88	23	29
Total	1.412 of 1.565	9%	54 of 58	6%	105	118	389	434	76	88	23	29

RicercaContenutoTest (16 dic 2020 19:16:38)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		14%		18%	101	118	372	434	75	88	22	29
Total	1.335 of 1.565	14%	47 of 58	18%	101	118	372	434	75	88	22	29

RicercaContenutoTest1 (16 dic 2020 19:17:59)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		7%		8%	108	118	400	434	79	88	24	29
Total	1.450 of 1.565	7%	53 of 58	8%	108	118	400	434	79	88	24	29

Test_ServzioEsistenteFilmatoInesistente (16 dic 2020 19:23:42)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		13%		12%	104	118	381	434	75	88	22	29
Total	1.357 of 1.565	13%	51 of 58	12%	104	118	381	434	75	88	22	29

Test_CreazioneServizioTV (16 dic 2020 19:19:51)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		35%		34%	87	118	290	434	62	88	20	29
Total	1.012 of 1.565	35%	38 of 58	34%	87	118	290	434	62	88	20	29

Test_FilmatoEsistenteServizioEsistente (16 dic 2020 19:21:09)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
VTV_CODICE		30%		27%	90	118	308	434	64	88	20	29
Total	1.085 of 1.565	30%	42 of 58	27%	90	118	308	434	64	88	20	29