Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Gestionale
**Corso di Progettazione del Software**
Proff. Toni Mancini e Monica Scannapieco

Progetto **PC.20080401**

versione del 2 aprile 2008

Si vuole progettare e realizzare *TuTubi*, un sistema informatico da usarsi su un portale Web per la pubblicazione e condivisione di file video da parte degli utenti registrati. Il sistema deve permettere la memorizzazione dei video e la loro visualizzazione da parte degli utenti. Questi devono poter anche esprimere valutazioni e commenti.

Si richiede di effettuare le fasi di Analisi, Progetto, e Realizzazione del sistema in JAVA, utilizzando la metodologia illustrata nel corso.

## Requisiti

*TuTubi* deve consentire la registrazione degli utenti, di cui interessa il nome e la data di iscrizione. Gli utenti registrati possono pubblicare video, visualizzare quelli disponibili, oltre che esprimere su di essi valutazioni e commenti testuali.

In particolare, di un video pubblicato da un utente interessa conoscere il titolo, la durata e la descrizione (oltre che il nome del file dove questo viene memorizzato da parte del sistema). Inoltre, di ogni video interessa conoscere la categoria (unica) a cui appartiene, oltre che un insieme di parole chiave dette *tag* (almeno una) che ne descrivano il contenuto in modo più strutturato. Delle categorie e dei tag interessa il nome.

*TuTubi* deve anche permettere ad un utente di pubblicare un nuovo video segnalando che si tratta di una risposta ad un video già esistente (utile per temi come la politica dove vi possono essere posizioni diverse che stimolano un dibattito).[1] *TuTubi* deve mantenere tale informazione in modo consistente, garantendo che nessun utente possa pubblicare un video in risposta ad un video pubblicato da sé stesso.

*TuTubi* deve poi fornire un servizio di cronologia, ricordando la sequenza di tutti i video visionati da ogni singolo utente, con relativa data e ora. Inoltre, per ogni video interessa conoscere il numero complessivo di volte che è stato visionato.

[1] Si assuma per semplicità che esista un operazione di un qualche use case `caricaVideo(nomefileSuComputerUtente:Stringa): Stringa` che consente di trasferire un file video dal computer dell'utente a *TuTubi*, ritornando il nome della copia del file trasferito (locale al sistema *TuTubi* dunque). Non si dia la specifica di tale operazione, ma la si utilizzi in modo opportuno.

Gli utenti di *TuTubi* devono inoltre avere la facoltà di esprimere una valutazione (un valore da 0 –pessimo– a 5 –ottimo) per ogni video che visionano. Tali valutazioni serviranno poi al sistema per promuovere i video più belli (cf. seguito) e scovare i più brutti. Tuttavia, per evitare degenerazioni nel meccanismo delle votazioni, l'utente che ha pubblicato un video non può votarlo, mentre ogni altro utente può votarlo al più una volta, indipendentemente dal numero di volte che l'ha visionato. In ogni caso però deve essere impossibile per un utente votare un video che non ha mai visionato.

Inoltre, per favorire uno spirito di comunità tra gli utenti del servizio, si prevede che questi possano esprimere commenti testuali ai video che visionano (dei quali interessa anche data e ora). A differenza delle valutazioni, un utente può esprimere più commenti per lo stesso video. Anche qui, non deve essere permesso ad un utente di scrivere commenti per video che non ha mai visionato.

Ogni utente può creare delle *playlist* personali, ovvero delle collezioni ordinate di video che gradisce vedere, oppure vuole condividere con altri utenti. Le playlist infatti (di cui interessa il nome e la data di creazione) possono essere pubbliche o private: solo le playlist pubbliche possono essere visualizzate dagli altri utenti. A tal fine, il sistema deve permettere ad ogni utente di ottenere le playlist pubbliche di un altro utente a sua scelta.

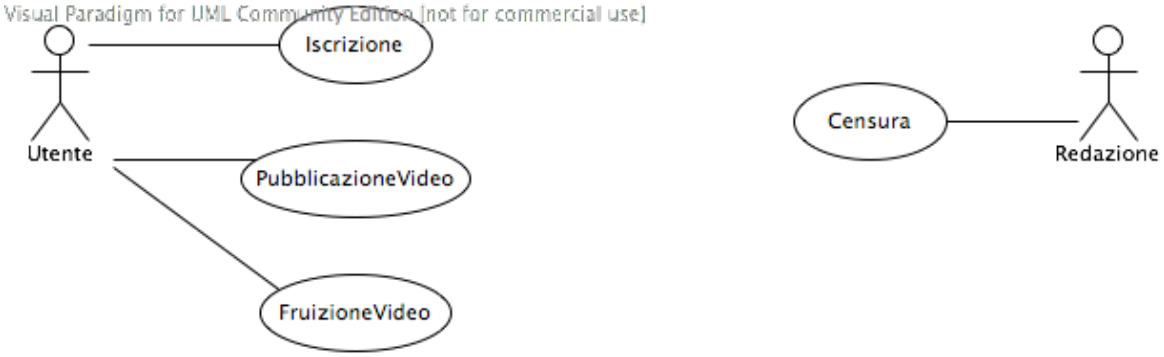
*TuTubi* deve permettere ad un utente di iscriversi, pubblicare nuovi video, creare e modificare le sue playlist, ed esprimere valutazioni e commenti sui video che visiona.

Inoltre, *TuTubi* deve consentire la ricerca di video: in particolare, data una categoria, un insieme di tag, ed un intero $v$ tra 0 e 5, si vogliono ritornare tutti i video disponibili di quella categoria che posseggono almeno uno tra i tag indicati, e che abbiano una valutazione media di almeno $v$ (se un video non ha ancora alcuna valutazione, deve essere ritornato comunque). *TuTubi* deve poi permettere di cercare, data una categoria, i video di quella categoria che hanno il numero maggiore di video in risposta, al fine di isolare le discussioni più animate tra gli utenti.

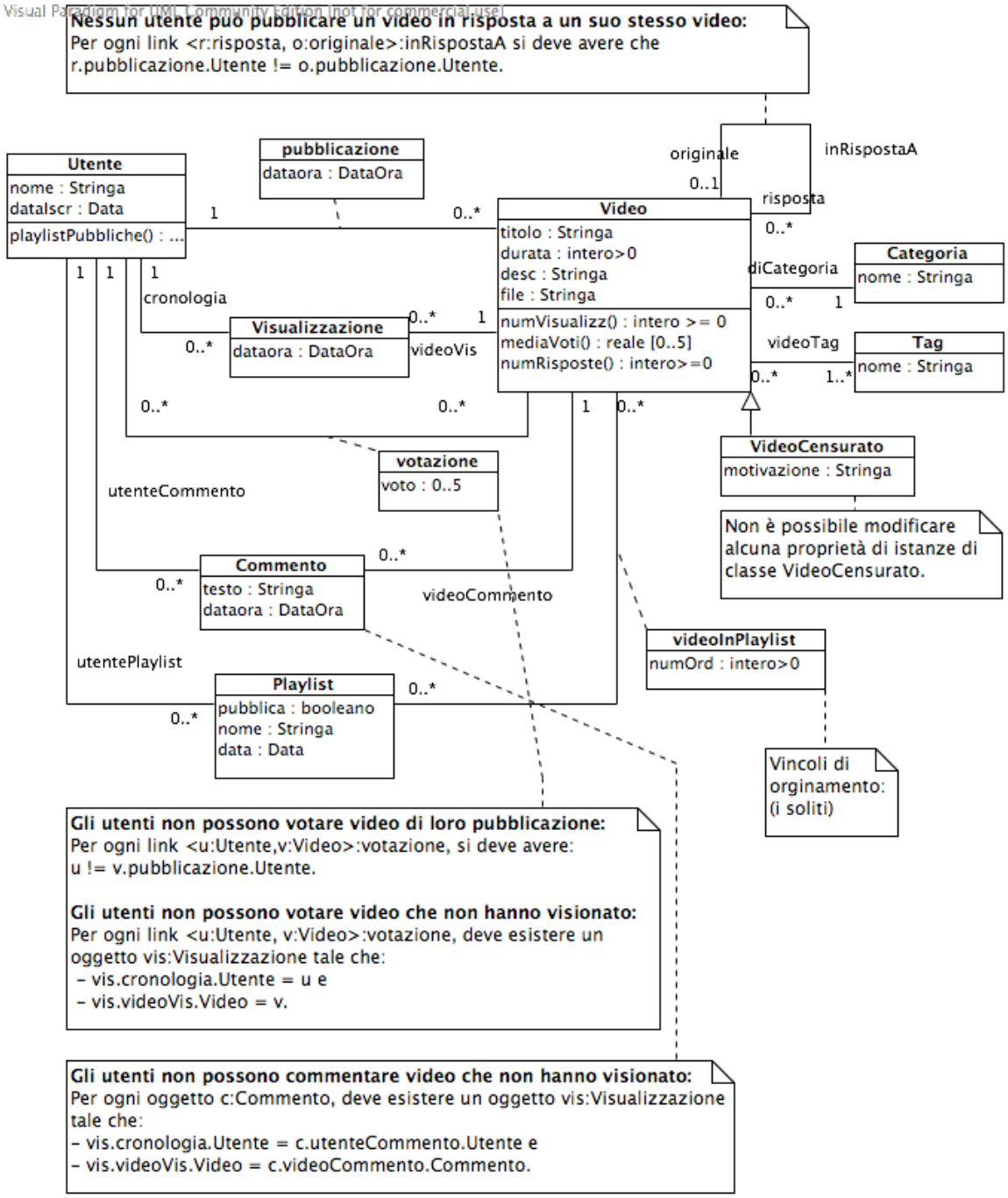
La redazione di *TuTubi* ha infine la facoltà di censurare dei video, ad esempio perché di contenuto coperto da copyright, osceno, ecc. Un video censurato non può essere né visionato, né votato, né commentato, né aggiunto ad alcuna playlist, né ritornato come risultato di una ricerca. Un video, una volta censurato, non può tornare più visibile. Il motivo di una censura deve essere mantenuto nel sistema per usi interni.

# 1 Fase di Analisi

## 1.1 Diagramma degli Use Case



## 1.2 Diagramma delle classi UML



### 1.3 Specifica dei tipi di dato

Nessun tipo di dato definito

### 1.4 Specifica degli use case

```
SpecificaUseCase Iscrizione
  iscrizione(nome:Stringa): Utente
    pre: nessuna
    post: result e' un nuovo oggetto di classe Utente con
     - u.nome = nome e
     - u.dataIscr = oggi (istanza del tipo Data che denota la data corrente).
FineSpecifica

SpecificaUseCase PubblicazioneVideo

  caricaVideo(nomefileSuComputerUtente:Stringa): Stringa  // spec. non richiesta
    pre: il file 'nomefileSuComputerUtente' esiste ed e' un file video
    post: il file viene trasferito nel sistema.
          result e' pari al nome della copia del file trasferito.

  calcolaDurata(file):intero>0  // spec. non richiesta
    pre: 'file' e' il nome di un file video nell'archivio del sistema
    post: result e' pari alla durata del file 'file' in secondi

  pubblicaVideo(u:Utente, titolo:Stringa, desc:Stringa, fileutente:Stringa,
                      c:Categoria, insTag: Insieme(Tag)   ): Video
    pre: - le precondizioni di caricaVideo(filelocale) sono soddisfatte
              (ovvero il trasferimento puo' avvenire correttamente);
         - |insTag| > 0.
    post:
        Detto nomefile = caricaVideo(fileutente),
        result e' un nuovo oggetto di classe Video tale che:
         - result.titolo = titolo;
         - result.durata = calcolaDurata(nomefile);
         - result.desc = desc;
         - result.file = nomefile.

       Vengono inoltre creati i seguenti link:
```

```
      - <u, result>:pubblicazione, con <u, result>.dataora = adesso
         (istanza del tipo DataOra che denota l'istante corrente);
      - <result, c>:diCategora;
      - <result, t>:videoTag, per ogni t in insTag.

  pubblicaVideoInRispostaA(u:Utente, titolo:Stringa,
                           desc:Stringa, fileutente:Stringa,
                           c:Categoria, insTag: Insieme(Tag),
                           orig: Video) : Video
    pre: - le precondizioni di pubblicaVideo(u, titolo, desc, fileutente, c, insTag)
           sono soddisfatte;
         - orig e' nello stato 'disponibile'
    post:
        Detto v = pubblicaVideo(u, titolo, desc, fileutente, c, insTag),
        result e' pari a v.
        Viene inoltre creato il link
        <orig:originale, v:risposta>:inRispostaA.

FineSpecifica

SpecificaUseCase FruizioneVideo
  visiona(u:Utente, v:Video)
    pre: v e' nello stato 'disponibile'
    post: result e' un nuovo oggetto di classe Visualizzazione con
          result.dataora = adesso.
          Vengono inoltre creati i link <u, result>:cronologia e <result,v>:videoVis.

  commenta(u:Utente, v:Video, c:Stringa): Commento
    pre: - esiste un oggetto vis:Visualizzazione tale che <u,vis>:cronologia e
             <vis, v>:videoVis  (ovvero l'utente u ha gia' visionato il video v)
         - v e' nello stato 'disponibile'
    post: result e' un nuovo oggetto di classe Commento tale che:
        - result.testo = c;
        - result.dataora = adesso;
        Vengono inoltre creati i link <u, result>:utenteCommento e
        <result, v>:videoCommento.

  vota(u:Utente, v:Video, voto:0..5)
    pre: - esiste un oggetto vis:Visualizzazione tale che <u,vis>:cronologia e
           <vis, v>:videoVis  (ovvero l'utente u ha gia' visionato il video v),
```

```
        - non esiste alcun link <u, v>: valuta (u non ha mai valutato v)
        - u != v.pubblicazione.Utente
        - v e' nello stato 'disponibile'
   post: Viene creato un nuovo link <u,v>:votazione con <u,v>.voto = voto.

 creaPlaylist(u:Utente, nome:Stringa, pub:booleano): Playlist
   pre: nessuna
   post: result e' un nuovo oggetto di classe Playlist con:
      - result.nome = nome;
      - result.pubblica = pub;
      - result.data = oggi.
      Viene inoltre creato il link <u,result>: utentePlaylist

 aggiungiVideoInPlaylist(u:Utente, p:Playlist, v:Video)
   pre: p.utentePlaylist = u, non esiste il link <p,v>:videoInPlaylist, e
        v e' nello stato 'disponibile'
   post: Viene creato un nuovo link <p,v>:videoInPlaylist con
         <p,v>.numOrd = |pre(p.videoInPlaylist)|+1.

 eliminaVideoDaPlaylist(u:Utente, p:Playlist, v:Video)
   pre: p.utentePlaylist = u ed esiste il link <p,v>:videoInPlaylist.
   post: Sia n = <p,v>.numOrd.
         - Viene eliminato il link <p,v>:videoInPlaylist;
         - Per ogni link <p, v'>:videoInPlaylist con pre(<p,v'>.numOrd) > n, si
           deve avere che <p,v'>.numOrd = pre(<p,v'>.numOrd)-1.

 cercaVideo(cat:Categoria, insTag: Insieme(Tag), v:0..5): Insieme(Video)
   pre: nessuna
   post: result = { v:Video |
                    - <v,cat>:diCategoria,
                    - v e' nello stato 'disponibile',
                    - se insTag != vuoto allora esiste t:insTag tale che <v,t>:videoTag,
                    - le prec. di v.mediaVoti() non sono soddisfatte oppure
                            v.mediaVoti() >= v }

 cercaVideoConPiuRisposte(cat:Categoria):Insieme(Video)
   pre: nessuna
   post: Detto V = {v:Video | <v, cat>:diCategoria e nello stato 'disponibile'},
         result = { v:Video |  v in V ed inoltre per ogni v' in V
                               si ha che v.numRisposte() >= v'.numRisposte() }.
```

```
FineSpecifica


SpecificaUseCase Censura
  censura(v: Video)
    pre: nessuna
    post: viene generato l'evento 'censura' su 'v'
FineSpecifica
```

## 1.5 Specifica delle classi e diagrammi degli stati e transizioni

### La classe Video

```
SpecificaClasse Video
  numVisualizz(): intero >= 0
    pre: nessuna
    post: result = |this.videoVis|.

  mediaVoti(): reale [0..5]
    pre: |this.votazione| > 0
    post:
```

$$\texttt{result} = (\Sigma_{\texttt{l:this.votazione}}\texttt{l.voto}) \,/\, \texttt{|this.votazione|}$$

```
  numRisposte(): intero >= 0
    pre: nessuna
    post: result = | this.risposta |.

FineSpecifica
```

Gli oggetti della classe Video evolvono secondo il seguente diagramma degli stati e transizioni:

**La classe Utente**

```
SpecificaClasse Utente
  playlistPubbliche(): Insieme(Playlist)
    pre: nessuna
    post: result = { p:Playlist  |  <this,p>:utentePlaylist e p.pubblica=true }.
FineSpecifica
```