GESTIONE DI MATERIALI AUDIO-VISIVI
Modello Formale e Descrizione di un Prototipo

Rapporto di Progetto - December 20, 2002

M. Baldoni, C. Baroglio, A. Martelli, G. M. Sacco,
M.L. Sapino, P. Bertolotti, C. Schifanella

Dipartimento di Informatica
Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
Tel. +39 011 6706756 — Fax. +39 011 751603
E-mail: {baldoni,baroglio,mrt,sacco,mlsapino,bertolot}@di.unito.it
URL: `http://www.di.unito.it/`

Abstract. Questo rapporto presenta il lavoro svolto nell'ambito del contratto n. 1023005091/00 stipulato tra il Dipartimento di Informatica dell'Università degli Studi di Torino e il Centro Ricerche della RAI con sede a Torino. Scopo del lavoro era la realizzazione di un modello formale che consentisse una gestione basata sulla conoscenza di materiali audiovisivi. In questa relazione verrà descritto il modello formale ideato e verrà presentato un prototipo realizzato per verificare sperimentalmente l'adeguatezza del modello.

1 Introduzione e Motivazioni

Il progresso tecnologico delle telecomunicazioni (via satellite e via internet) e la maggiore flessibilità degli strumenti preposti alla fruizione di materiale audio-visivo, testimoniati da questi ultimi anni, hanno condotto diverse compagnie televisive allo studio di nuove forme di produzione altamente automatizzate. A tal fine si è resa necessaria la definizione di standard di *meta-dati* che consentano di arricchire i prodotti, inserendo nei medesimi informazioni utili sia per comprenderne i contenuti in modo non ambiguo sia per eseguire operazioni di *rights tracking* (tener traccia dei diritti legali che limitano l'utilizzo del prodotto). In generale, col termine *meta-dato* si intende un'informazione aggiunta a un prodotto (o a del materiale grezzo, per esempio una cassetta registrata o un file), che specifica delle caratteristiche o delle informazioni relative al prodotto medesimo. Fra le varie proposte troviamo per esempio [3, 4, 6, 11, 1]. Queste proposte sono accomunate dal tentativo di individuare un insieme di elementi informativi (proprietà del documento) con lo scopo di definire un *vocabolario* comune, tramite il quale etichettare i prodotti trattati. In altri termini, l'uso di un vocabolario aggiunge un livello semantico al materiale audio-visivo, consentendo come conseguenza l'automatizzazione di molti controlli e operazioni di gestione, che in questo modo possono essere delegati a dei sistemi software. Prescindendo dalla differenza di contesto, un analogo sforzo è stato condotto all'interno della comunità degli editori –sia di riviste sia di libri– portando alla definizione di standard analoghi a quelli citati prima (si veda, ad esempio, PRISM [9]).

Nonostante tutte queste iniziative, molti produttori di contenuti considerano gli standard proposti ancora insufficienti, in quanto essi non forniscono strumenti per la formalizzazione ad alto livello di *aspetti strutturali e relazionali* fra gli elementi di informazione individuati. In altri termini tramite questi standard non è ancora possibile esprimere una vera *conoscenza del dominio*. Se fosse possibile rappresentare anche gli aspetti relazionali e tali relazioni fossero condivise dai diversi produttori, sarebbe possibile realizzare applicativi per la documentazione, per il retrieval, per la scelta e per la presentazione dell'informazione più efficaci.

Scopo di questo lavoro è definire un *framework formale* per la *rappresentazione*, l'*elaborazione* e l'*archiviazione* di *materiale audio-visivo* (AVM nel seguito). Il modello proposto può essere utilizzato come base per la realizzazione di sistemi di annotazione automatica e di sistemi più efficaci per il retrieval di specifico materiale audio-visivo. Nella presente relazione oltre a descrivere il modello formale, descriveremo anche un *prototipo* realizzato sulla base del modello formale e finalizzato a verificare sperimentalmente l'efficacia del medesimo.

Per fissare le idee sugli aspetti principali presi in considerazione in questo lavoro, facciamo un esempio: si consideri il film Metropolis, nel nostro archivio potrebbero essere contenute più copie del medesimo, magari salvate su supporti differenti (file piuttosto che nastri). Alcune delle proprietà relative al film quali ad esempio il regista o la data di produzione sono indipendenti dalla specifica copia considerata, altre invece sono ad essa strettamente collegate (a seconda del supporto fisico occorrerà per esempio, un diverso lettore). Inoltre se il film viene rielaborato producendo una nuova versione (a colori, arricchita da nuovi

effetti speciali) occorrerà tener traccia di chi ha svolto quali operazioni e così via. Nonostante ciò molte proprietà della nuova versione rimarranno immutate rispetto alla versione originale (per esempio gli attori non cambieranno). Come vedremo nel seguito, nel nostro modello la rappresentazione degli AVM è suddivisa in due livelli: da un lato abbiamo gli *enconding* dall'altro il loro *contenuto semantico* (concetti astratti). Le relazioni fra encoding e concetti astratti sono mantenute da opportune *funzioni di interpretazione*. Un altro aspetto rilevante riguarda la modellazione della trasmissione degli AVM, catturata da elementi detti *eventi*. Anche gli eventi hanno proprietà specifiche che debbono essere conservate e gestite; si osservi infine che una trasmissione può essere registrata, producendo un nuovo encoding.

Il framework risultante modella sia la distinzione fra i diversi livelli semantici che caratterizzano ogni AVM sia alcuni aspetti composizionali centrali nella realizzazione di programmi televisivi: occorre infatti tenere in considerazione il fatto che diversi programmi sono ottenuti componendo brani di programmi pre-esistenti (si pensi, ad esempio, ai programmi contenitore). In questi casi è importante riuscire a risalire ai programmi originali utilizzati per la composizione, tener traccia di come è stata effettuata la composizione e comprendere quali proprietà e caratteristiche del nuovo programma possono essere derivate da quelle dei programmi composti. Questo framework apre la via al progetto di una serie di strumenti, fra i quali spiccano da un lato motori di ricerca particolarmente efficaci per le banche dati relative al materiale immagazzinato, dall'altro tool semi-automatici di annotazione di nuove produzioni ottenute riutilizzando materiale di proprietà dell'azienda.

2 Proposta di un modello semantico

In questa sezione introduciamo i concetti principali che stanno alla base della nostra proposta: gli *encoding*, i *concetti astratti*, le *interpretazioni* e gli *eventi*.

A titolo di esempio, si assuma che due amici stiano parlando del film Metropolis. Entrambi, probabilmente, hanno in mente il film di Fritz Langs del 1926, indipendentemente dal luogo e dal momento in cui i due hanno visto il film in questione. Di fatto, nel dire "guardare Metropolis", si fa riferimento (i) ad un concetto astratto (il film di Fritz Lang), e (ii) ad un encoding (ad esempio, una cassetta VHS), associata a quel concetto. Ed è proprio per catturare questa dualità che introduciamo il concetto di *interpretazione*, che mette in relazione i concetti astratti e i relativi encoding. è possibile associare uno stesso concetto astratto a diversi encoding, che possono essere visti come manifestazioni fisiche del concetto.

Facendo ancora riferimento all'esempio, possiamo affermare che gli encoding DVD di Metropolis sono chiaramente diversi dalla loro registrazione su nastro, pur essendo entrambi associati allo stesso concetto, il film Metropolis. Analogamente può esistere più di una interpretazione per un dato encoding. Volendo ancora considerare, a titolo di esempio, il film Metropolis, è un dato di fatto che un critico cinematografico darebbe alla versione a colori un significato diverso

dal quello attribuito all'originale in bianco e nero, mentre un fruitore inesperto plausibilmente attribuirebbe alle due versioni esattamente lo stesso film.

2.1 Encoding

Definizione 1 (Encoding). Un encoding C è una 5-tupla

$$\langle id, hs, tp, al, sl \rangle$$

dove: *id* è un identificatore univoco, *hs* cattura la storia dell'encoding (riprenderemo a breve questo concetto), *tp* è il tipo, *al* è una lista di attributi, *sl* è una lista di supporti fisici (eventualmente vuota). Un attributo è una coppia ⟨nome-attributo, valore-attributo⟩.

Nel seguito, denoteremo con $\mathbf{C}$ un insieme di encoding C, con $\mathbf{A}$ l'insieme di tutti gli attributi di encoding, e con $\mathbf{S}$ l'insieme dei supporti fisici. *al* ed *sl* sono rispettivamente sottoinsiemi di $\mathbf{A}$ di $\mathbf{S}$. Ciascun encoding ha anche un *tipo*, che dà informazioni circa l'algoritmo di codifica utilizzato, e dunque, indirettamente, descrive la rappresentazione che è stata usata. Si considerino, ad esempio, diversi tipi di file di immagini. La stessa immagine può comparire in diversi formati, quali *jpeg*, *bitmap*, o *png*. Per visualizzare l'immagine, si deve poter interpretare correttamente il formato del file in modo tale da trasformarlo in una matrice di pixel colorati.

L'insieme di tutti i tipi di encoding verrà denotato con $\mathbf{T}$. $\mathbf{C}_{tp}$ denoterà il sottoinsieme di encoding C in $\mathbf{C}$ aventi tipo *tp*. L'insieme $\mathbf{C}$ sarà dunque

$$\mathbf{C} = \bigcup_{tp \in \mathbf{T}} \mathbf{C}_{tp}$$

Sugli encoding si possono definire delle funzioni di trasformazione. Tali funzioni prevedono argomenti che sono encoding tipati, e restituiscono a loro volta encoding tipati. Un esempio di operazione di questo genere è la trasformazione di un encoding GIF in una corrispondente codifica PNG, rappresentante la stessa immagine.

Definizione 2. Un'operazione su encoding è una funzione

$$op : \mathbf{C}_{tp1} \times \mathbf{C}_{tp2} \times \ldots \times \mathbf{C}_{tpk} \rightarrow \mathbf{C}_{tp}$$

Non ci sono ipotesi a priori sui tipi degli encoding argomento né sul tipo del risultato. In particolare, non si richiede nemmeno che operazioni che combinano encoding tra loro omogenei (ossia dello stesso tipo) restituiscano encoding omogenei all'input. Questa scelta garantisce generalità e flessibilità al modello, consentendo di caratterizzare svariate funzioni di trasformazione, quali la *composizione*, l'*associazione*, la *decomposizione*, la *disassociazione*. Ad esempio, è possibile combinare una colonna sonora con un video, per ottenere un prodotto audio-video. Combinando un insieme di immagini jpeg, invece, potremmo ottenere una nuova immagine jpeg o magari un'immagine in un altro formato o

un'animazione. Ciascuna singola operazione avrà le proprie restrizioni e i propri requisiti di applicabilità, visti come parte della specifica. Si noti inoltre che i tipi degli encoding coinvolti possono essere sia tipi elementari, quali ad esempio jpeg o mp3 o avi o tipi composti, quale ad esempio ⟨avi, jpeg⟩.

Ai fini del progetto, risultano particolarmente interessanti quelle operazioni che *tengono traccia della storia*, e consentono in ogni istante di ricostruire i passi che hanno condotto alla definizione di un encoding. È infatti importante poter distinguere tra encoding primitivi e encoding che risultano dall'applicazione di certe operazioni sui loro argomenti (i quali, a loro volta, potrebbero essere encoding primitivi o risultati di operazioni).

Si consideri una generica operazione

$$op : \mathbf{C}_{tp1} \times \mathbf{C}_{tp2} \rightarrow \mathbf{C}_{tp}$$

op combina opportunamente coppie di encoding quali $\langle id_i, hs_i, tp1, al_i, sl_i \rangle$ e $\langle id_j, hs_j, tp2, al_j, sl_j \rangle$ e restituisce encoding del tipo $\langle id_z, hs_z, tp, al_z, \emptyset \rangle$, dove id_z è un nuovo identificatore, hs_z è la 3-tupla $\langle op, id_i, id_j \rangle$, tp è il tipo del dominio di *op* e al_z è una lista di attributi.

Un'altra informazione importante che il modello consente di conservare è la *lista dei supporti di memorizzazione*, in cui vengono elencati tutti i supporti fisici su cui l'encoding è memorizzato. Tale lista può essere vuota. Di fatto, possono esistere copie multiple dello stesso encoding, e non ci sono restrizioni a priori sui supporti fisici (le uniche restrizioni hanno a che vedere con la realizzabilità, dal punto di vista fisico, dell'operazione di memorizzazione –ad esempio, non sarebbe possibile registrare su un CD un audio in formato analogico).

Questa informazione si rivelerà utile, ad esempio, quando si dovrà produrre nuovo materiale a partire da prodotti già memorizzati in archivio. In questo caso, si farà ricorso ai dati forniti dal modello per individuare quali supporti di memoria (dischi, nastri, cassette) debbano essere riprodotti, ai fini della nuova produzione.

Per modellare la memorizzazione di un prodotto su un supporto fisico, serve soltanto ricordare l'identificatore del supporto coinvolto nell'operazione. L'operazione di memorizzazione può pertanto essere formalizzata come segue.

Definizione 3. Una operazione di memorizzazione è una funzione

$$rec : \mathbf{C}_{tp} \times \mathbf{S} \rightarrow \mathbf{C}_{tp}$$

Dati un encoding $\langle id, hs, tp, al, sl \rangle$ ed un identificatore di supporto s, l'operazione restituisce il nuovo encoding $\langle id, hs, tp, al, \{s\} \cup sl \rangle$, che differisce da quello dato in input soltanto per quanto concerne la lista dei supporti, estesa con l'informazione relativa al nuovo identificatore di supporto.

Per concludere la discussione sulla definizione di encoding, vale la pena di aggiungere qualche commento sulla *lista di attributi*. Si tratta di una lista di proprietà che caratterizzano l'encoding stesso, quali l'autore, la data di produzione, la durata, ... Sono essenzialmente meta-dati relativi all'encoding. Tra queste proprietà, alcune sono note fin dall'inizio delle attività di creazione dell'encoding

stesso. Tra queste, la data di creazione. Altre invece vengono catturate ed associate all'encoding ad un certo punto della sua esistenza. Tra queste, i diritti d'autore, i copyright, ecc.

Per consentire l'espansione della lista di attributi, in un qualsiasi momento, definiamo la seguente operazione di *caratterizzazione*.

Definizione 4. Un'operazione di caratterizzazione è una funzione

$$cr : \mathbf{C}_{tp} \times \mathbf{A} \rightarrow \mathbf{C}_{tp}$$

Dati un encoding $\langle id, hs, tp, al, sl \rangle$ ed un attributo a, l'operazione restituisce il nuovo encoding $\langle id, hs, tp, \{a\} \cup al, sl \rangle$.

A questo punto, per attribuire un significato al nuovo encoding prodotto, si renderà necessario un altro passo di interpretazione. Ed è proprio questo passo che ci consente di modellare il caso discusso nell'esempio introduttivo.

Assumendo di disporre di un encoding del film Metropolis, e di una funzione di trasformazione che, applicata ad essa, restituisca un nuovo encoding, ottenuto eliminando i primi 10 minuti da quella data. A seconda della funzione di interpretazione scelta, si potranno assegnare significati distinti, oppure uno stesso significato, ai due encoding. Ovviamente possono esistere molteplici funzioni di interpretazione, in modo tale da poter correttamente rappresentare i punti di vista di utenti diversi.

In generale, si può pensare di disporre di diversi tipi di interpretazioni, in base ai quali interpretazioni apparentemente simili (ma non identiche) di uno stesso oggetto possano essere identificate, o mantenute distinte. La scelta dipenderà dal contesto d'uso, e dal ruolo che tali interpretazioni assumeranno nell'economia globale del sistema.

Ad esempio, è noto che spesso i film trasmessi dalle emittenti televisive sono leggermente diversi dagli originali. La trasformazione è dovuta spesso a vincoli temporali di palinsesto, alle frequenti interruzioni pubblicitarie, ecc., che impongono che la proiezione debba essere completamente contenuta in un dato intervallo di tempo, che non necessariamente corrisponde alla durata della versione originale del film. Tuttavia, nonostante la versione trasmessa non sia quella originale, e nonostante un telespettatore esperto sia in grado di cogliere le differenze tra le due, viene mantenuto lo stesso titolo, e viene mantenuta l'associazione semantica del prodotto trasmesso col concetto associato all'originale.

2.2 Concetti Astratti e Interpretazioni

Le differenze esistenti tra due prodotti vengono comunque modellate a livello di encoding, ed è comunque possibile enfatizzare le diverse identità dei due prodotti, in contesti diversi. Ad esempio storici e/o critici cinematografici senzaltro assocerebbero interpretazioni diverse ai due prodotti.

Definizione 5 (Concetto astratto). Un concetto astratto I è una tupla $\langle id, al \rangle$, in cui id è un identificatore univoco, ed al è una lista di attributi.

Definizione 6 (Interpretazione). Un'interpretazione, per un insieme di encoding $\mathbf{C}$, è una tupla $\langle \mathbf{I}, i \rangle$, in cui $\mathbf{I}$ è un dominio di concetti astratti, ed i è una funzione parziale, detta funzione di interpretazione, da $\mathbf{C}$ ad $\mathbf{I}$.

Dal punto di vista intuitivo, non è detto che ciascun encoding sia associato ad un concetto astratto (questa è la ragione della parzialità della funzione di interpretazione). Ad esempio, ci possono essere encoding che rappresentano dati parziali, da utilizzarsi al fine della realizzazione di un programma significativo. È inoltre possibile che diversi encoding siano associati ad uno stesso concetto: non si richiede infatti che la funzione di interpretazione sia iniettiva.

È anche importante poter seguire il percorso inverso, e poter risalire, data una qualsiasi funzione di interpretazione i, da un concetto astratto all'insieme degli encoding che l'interpretazione i gli ha associato.

Denotiamo con $\overline{i}$ la funzione che modella l'associazione a ritroso. Si noti che $\overline{i}$ non è semplicemente la funzione inversa i^{-1} delle funzione $i : \mathbf{C} \rightarrow \mathbf{I}$, dal momento che il codominio di $\overline{i}$ non è $\mathbf{C}$, bensì $2^{\mathbf{C}}$.

Definizione 7. Sia $\langle \mathbf{I}, i \rangle$ un'interpretazione per un insieme di encoding $\mathbf{C}$. Dati un encoding $C \in \mathbf{C}$, ed un concetto astratto $I \in \mathbf{I}$ denotiamo con $\langle \mathbf{I}, i[C \rightarrow C'] \rangle$ una nuova interpretazione tale che $i[C \rightarrow C'](C'') = i(C'')$ per ogni $C'' \in \mathbf{C}$ tale che $C'' \neq C$ e $i[C \rightarrow C'](C) = C'$.

Definizione 8. Una qualsiasi operazione $op : \mathbf{C}_{tp1} \rightarrow \mathbf{C}_{tp2}$ viene detta invariante rispetto ad $\langle \mathbf{I}, i \rangle$, se per ogni encoding $C \in \mathbf{C}_{tp1}$ vale l'uguaglianza

$$i(C) = i(op(C))$$

2.3 Eventi

Un'altra delle entità che le compagnie televisive devono gestire è la trasmissione di materiale audiovisivo. Tali entità sono caratterizzate da alcune proprietà, quali data e ora di trasmissione, il canale di trasmissione, gli sponsor, i diritti televisivi, che non sono direttamente associabili ai concetti discussi fino a questo punto. Per esempio che è possibile riprodurre più volte uno stesso encoding di un concetto. Sostanzialmente, le diverse riproduzioni di ciascun encoding assumeranno valori diversi per almeno un sottoinsieme delle proprietà che le caratterizzano. Nel nostro framework facciamo riferimento alle diverse trasmissioni chiamandole *eventi*.

Definizione 9 (Evento). Un evento EV è una 4-tupla

$$\langle id, C, al, t \rangle$$

in cui id è un identificatore univoco, C è un encoding, al è una lista di attributi, e t è l'istante temporale di inizio della trasmissione dell'encoding.

La lista di attributi al descrive proprietà specifiche dell'evento in questione. Dato un insieme di encoding $\mathbf{C}$, denotiamo con $\mathbf{EV_C}$ un insieme di eventi basati su $\mathbf{C}$.

2.4 Concept

Un ulteriore aspetto che occorrerebbe integrare nel modello proposto riguarda la cattura, in un formato che sia machine-processable, del processo mentale sottostante la definizione e la creazione di un prodotto televisivo. Tali processi sono estremamente variabili, generalmente lunghi e complessi ed includono diversi passi di sviluppo, dalla stesura di un copione alla costruzione delle scene al casting e così via. Allo stato attuale del progetto non abbiamo una formalizzazione completa del processo tuttavia riteniamo che la seguente traccia sia promettente.

Definizione 10 (Concept). Un concept Co è una coppia $\langle id, al \rangle$ in cui id è un identificatore univoco, e al è un insieme di attributi.

Co denota l'insieme di tutti i concetti. In generale, ogni encoding può essere associato ad un concept che descrive il processo in seguito al quale l'encoding è stato creato. Si osservi che qualcosa di analogo dovrebbe valere per i concept, in quanto anche un concept è –di norma– il risultato di una certa sequenza di passi di produzione.

Definizione 11 (Associazione encoding-concept). Un'associazione encoding-concept è una funzione parziale

$$g : \mathbf{EV_C} \rightarrow \mathbf{Co}$$

Al fine di trattare l'aggiunta di proprietà ai concept, il formalismo dovrebbe prevedere un'operazione di caratterizzazione dei medesimi.

Definizione 12 (Caratterizzazione di un concept). Per caractterizzazione di un concept Co si intende una funzione

$$cr : \mathbf{Co} \times \mathbf{A} \rightarrow \mathbf{Co}$$

Dato un concept $\langle id, al \rangle$ e un attributo a, cr restituisce il nuovo concept $\langle id, \{a\} \cup al \rangle$.

3 Realizzazione di un prototipo

In questa sezione presentiamo le linee guida che abbiamo seguito nella realizzazione di un prototipo finalizzato a sperimentare l'adeguatezza del modello formale proposto nelle precedenti sezioni.

La scelta realizzativa principale è stata di rappresentare il modello ideato sotto forma di un'ontologia, realizzata nel linguaggio DAML+OIL [2]. Questa scelta è motivata dal particolare contesto applicativo in cui il lavoro si situa, che è caratterizzato da una natura fortemente distribuita e dalla forte eterogeneità degli strumenti e delle sorgenti informative trattate. Il linguaggio DAML+OIL, che si è imposto negli ultimi anni quale standard per la rappresentazione dei contenuti semantici di documenti accessibili via web, si basa su di un meccanismo di definizione di classi e su di un principio di ereditarietà delle proprietà

all'interno della gerarchia delle classi, consentendo la realizzazione di meccanismi di inferenza. È un linguaggio machine-oriented particolarmente adeguato allo scambio e al trattamento di informazioni. La scelta di questo linguaggio è anche stata motivata dall'esistenza di diversi strumenti per il suo utilizzo; fra questi alcuni editor (come per esempio OntoEdit [8], OilEd [7] e Protégé [10]) e alcune API per lo sviluppo di programmi (per esempio per la gestione di ontologie DAML in Java, la suite Jena di HP [5]).

3.1 L'ontologia AVM-DAML

Con il termine ontologia si intende un documento condiviso che contiene la descrizione formale dei concetti di un dato dominio, identificando le classi principali, la loro organizzazione, le loro proprietà e le relazioni più significative fra le classi descritte.

L'ontologia AVM-DAML, da noi realizzata, implementa il modello formale presentato nelle sezioni precedenti ed è riportata nell'appendice A. Encoding, concetti astratti, eventi e interpretazioni sono rappresentati da quattro classi DAML ad essi corrispondenti. Per questioni pratiche è stata introdotta quale sovraclasse comune a tutte le classi dell'ontologia AVM-DAML la classe "Entità", contenente le proprietà comuni a tutti gli elementi dell'ontologia. Più in dettaglio le classi sono descritte ciascuna da una tabella, contenente l'elenco delle sue proprietà, del dominio su cui tali proprietà assumono valori (range) e dalle caratteristiche delle proprietà medesime. In particolare, si noti che di default le proprietà sono multivalore (quindi possono assumere insiemi di valori –si pensi, per esempio, ad un encoding avente diversi supporti–).

Si osservi che alcune proprietà delle classi realizzate sono state introdotte a titolo esemplificativo (per esempio la proprietà "colore" di "Encoding"). È importante notare, infine, che nell'ontologia AVM-DAML, a differenza di quanto mostrato nella sezione precedente, è possibile associare concetti astratti come interpretazione di parti di Encoding e non solo dell'intero encoding, particolarmente utile per evitare, dal punto di vista pratico, una effettiva estrazione e memorizzazione di quella parte di encoding.

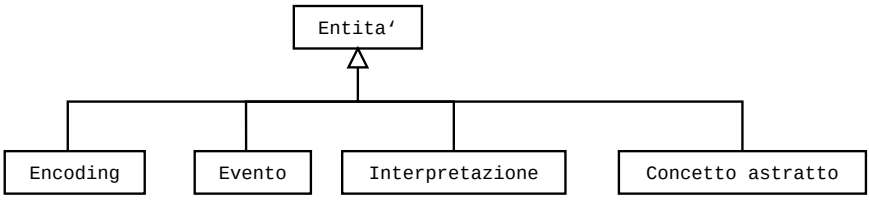

Fig. 1. Elementi principali del modello riportati nell'ontologia.

Proprietà della classe *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
ID	string	Datatype, valore unico, valore non ambiguo*

Proprietà della classe *Encoding*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
storia	Operazione	Object, valore unico*
tipo	string	Datatype
supporto	SupportoFisico	Object
durata	duration	Datatype, valore unico
colore	string	Datatype, valore unico

Proprietà della classe *Evento*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
canale	string	Datatype, valore unico
momento	dataTime	Datatype, valore unico
encodingTrasmesso	Encoding	Object, valore unico*

Proprietà della classe *Interpretazione*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
concettoInterpretato	ConcettoAstratto	Object, valore unico*
encodingInterpretante	Encoding	Object, valore unico*
da	time	Datatype, valore unico
a	time	Datatype, valore unico

Proprietà della classe *ConcettoAstratto*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
copyright	string	Datatype

Seguendo il modello formale, sugli Encoding è possibile eseguire delle operazioni. Nel prototipo abbiamo introdotto una classe Operazione avente tre sottoclassi (Registrazione, Unione ed Estrazione), ciascuna delle quali corrisponde ad un'operazione specifica (si veda la Fig. 2). La classe *SupportoFisico* rappresenta il tipo di supporto su cui un Encoding può essere memorizzato e la collocazione del medesimo.

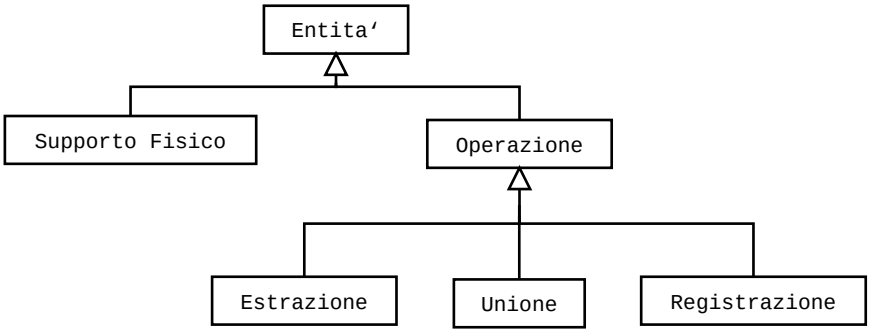

Fig. 2. Classi che definiscono le operazioni.

Proprietà della classe *Operazione*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
eseguitaIl	dataTime	Datatype, valore unico
eseguitaDa	Persona	Object

Proprietà della classe *Registrazione*, eredita da *Operazione*

PROPRIETÀ	RANGE	CARATTERISTICHE
eventoRegistrato	Evento	Object, valore unico*

Proprietà della classe *Estrazione*, eredita da *Operazione*

PROPRIETÀ	RANGE	CARATTERISTICHE
encodingManipolato	Encoding	Object, valore unico*

Proprietà della classe *Unione*, eredita da *Operazione*

PROPRIETÀ	RANGE	CARATTERISTICHE
encodingUnito	Encoding	Object

Proprietà della classe *SupportoFisico*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
tipoSupporto	string	Datatype, valore unico
collocazione	string	Datatype

Per rendere più verosimile il prototipo abbiamo realizzato anche alcune classi che rappresentano concetti astratti (sottoclassi di ConcettoAstratto) relativi al dominio degli AVM, come per esempio ElementoDiScaletta e Film (si veda la Fig. 3).

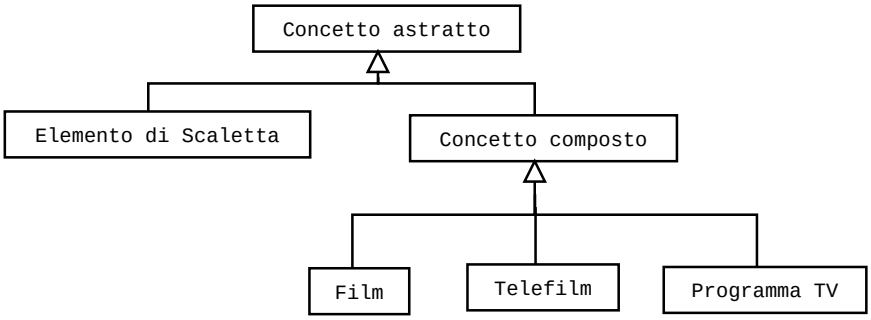

Fig. 3. Classi aggiuntive.

Proprietà della classe *ConcettoComposto*, eredita da *ConcettoAstratto*

PROPRIETÀ	RANGE	CARATTERISTICHE
regista	Persona	Object
produttore	Persona	Object

Proprietà della classe *Film*, eredita da *ConcettoComposto*

PROPRIETÀ	RANGE	CARATTERISTICHE
titolo	string	Datatype, valore unico
attoreProtagonista	Persona	Object
tecnicoAudio	Persona	Object
trama	string	Datatype, valore unico

Proprietà della classe *Telefilm*, eredita da *ConcettoComposto*

PROPRIETÀ	RANGE	CARATTERISTICHE
nomeSerie	string	Datatype, valore unico
titoloPuntata	string	Datatype, valore unico
numeroPuntata	positiveInteger	Datatype, valore unico

Proprietà della classe *ProgrammaTV*, eredita da *ConcettoComposto*

PROPRIETÀ	RANGE	CARATTERISTICHE
conduttore	Persona	Object
ospite	Persona	Object

Proprietà della classe *ElementoDiScaletta*, eredita da *ConcettoComposto*

PROPRIETÀ	RANGE	CARATTERISTICHE
parteDi	ConcettoComposto	Object
descrizione	string	Datatype, valore unico
tipoEDS	string	Datatype
personaCoinvolta	Persona	Object
ambientazione	string	Datatype

Infine è stata realizzata la classe Persona, resa necessaria dall'esigenza di definire valori di proprietà, come ad esempio gli esecutori delle Operazioni (si veda la Fig. 4).

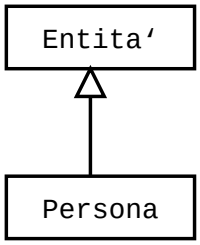

Fig. 4. La classe Persona.

Proprietà della classe *Persona*, eredita da *Entità*

PROPRIETÀ	RANGE	CARATTERISTICHE
nome	string	Datatype
cognome	string	Datatype, valore unico

3.2 Il package JAVM per la gestione dell'ontologia AVM-DAML

Il package JAVM, realizzato in linguaggio Java, offre un'interfaccia di programmazione per la gestione dell'ontologia AVM-DAML introdotta nella sezione precedente. Nell'attuale versione JAVM fa uso del package Jena nella versione 1.6 attualmente disponibile distribuita da Hevlett-Packard [5] per la gestione dell'ontologia AVM-DAML[1] (si veda la Fig. 5).

Per ogni classe introdotta nell'ontologia AVM-DAML è stata definita una analoga classe nel package JAVM rispettandone la struttura gerarchica. Ogni istanza di una classe in JAVM ha un riferimento alla istanza della corrispondente classe AVM-DAML che ne rappresenta lo stato ed il cui accesso è realizzato internamente attraverso il package Jena (si veda la Fig. 6). Ogni classe JAVM definisce però anche la "business logic" delle istanze come, ad esempio, il modo in cui effettuare l'operazione di unione di due encoding.

La Fig. 7 riporta il diagramma delle classi del package JAVM (*it.unito.di.javm*). La classe *ManagerOntologiaDAML* contiene la realizzazione dell'effettiva interfaccia verso l'ontologia AVM-DAML attraverso il package Jena. Questa classe contiene quindi i metodi (per lo più statici) per caricare/salvare il modello AVM-DAML, creare/rimuovere e gestire le proprietà delle istanze AVM-DAML e rende completamente trasparente al programmatore l'utilizzo del package Jena e quindi

[1] Nelle tabelle che descrivono le proprietà delle istanze delle classi AVM-DAML, abbiamo segnato con un asterisco le proprietà che in DAML sarebbero caratterizzate come "non ambigue" oppure a "valore unico", in quanto la versione 1.6 di Jena non consente di trattarle in modo completo.

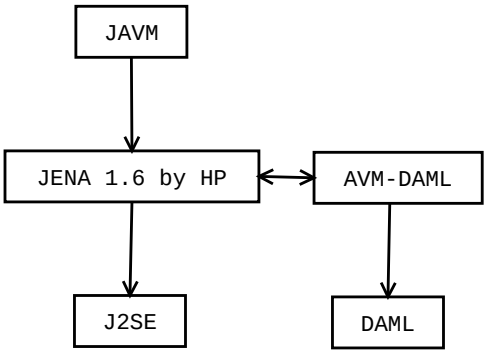

Fig. 5. Relazione tra JAVM e Jena.

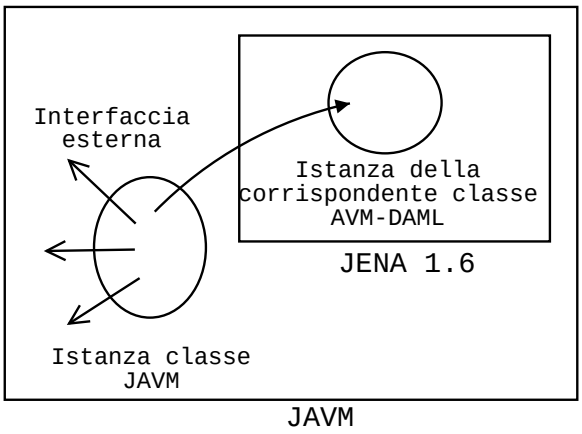

Fig. 6. Istanza JAVM vs istanza AVM-DAML.

una sua eventuale sostituzione o upgrade. I metodi di tale classe sopperiscono anche ai controlli sull'unicità del valore di una certa proprietà non gestiti dalla versione 1.6 di Jena.

4 Conclusioni e lavoro futuro

In questo rapporto abbiamo presentato il lavoro svolto nell'ambito del contratto n. 1023005091/00 stipulato tra il Dipartimento di Informatica dell'Università degli Studi di Torino e il Centro Ricerche della RAI, finalizzato alla realizzazione di un modello formale che consenta una gestione basata sulla conoscenza di materiali audiovisivi. Il risultato di questo lavoro è un *framework formale* che cattura i concetti più rilevanti del contesto applicativo studiato –encoding, concetti astratti, eventi, interpretazioni– e delinea la trattazione di alcuni altri aspetti inerenti la registrazione strutturata del processo evolutivo, che conduce ad una nuova

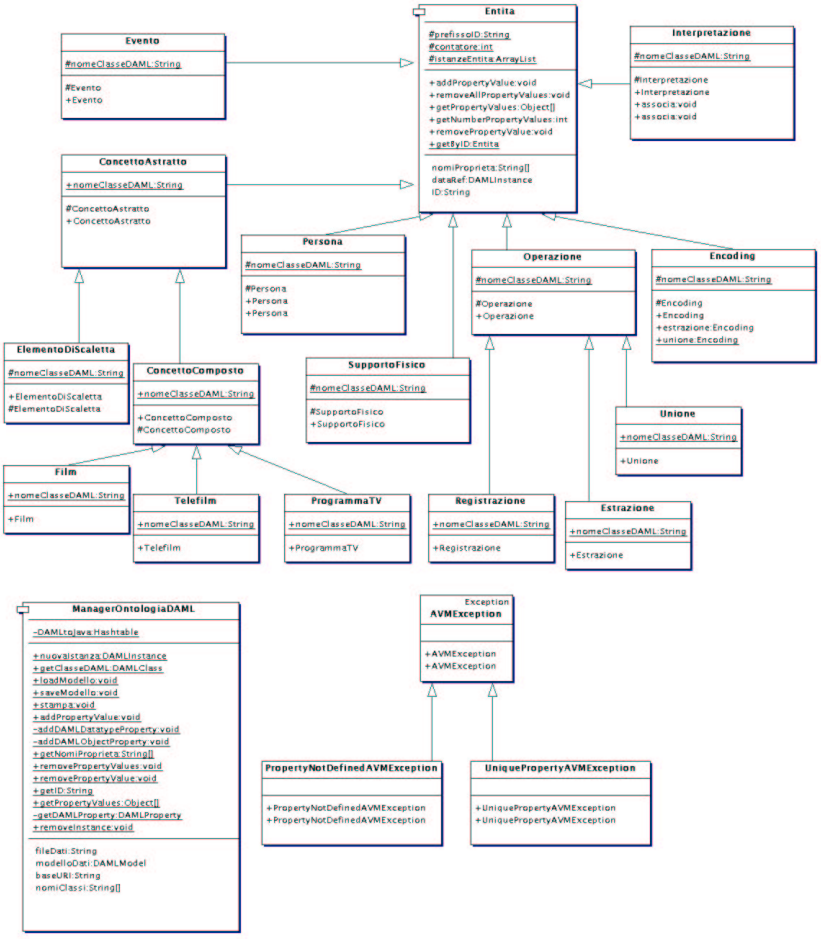

Fig. 7. Diagramma delle classe per il package JAVM.

produzione televisiva –concept, caratterizzazioni–. Oltre ad una trattazione teorica, abbiamo presentato un *sistema prototipo* finalizzato a dimostrare l'utilità pratica del modello proposto. Tale sistema comprende da un lato un'ontologia realizzata per mezzo del linguaggio DAML+OIL e dall'altro un insieme di classi Java che consentono di gestire l'ontologia in questione. L'ontologia cattura sia i concetti principali del modello e le loro relazioni sia alcuni elementi caratterizzanti aventi una valenza più applicativa (come ad esempio "film" o "regista"). La parte in Java utilizza il package Jena, di HP, versione 1.6.

L'approccio da noi proposto è aperto ad ulteriori sviluppi in parte tecnologici ed in parte relativi alla rappresentazione e all'uso della conoscenza. Fra questi ultimi, una possibilità è sicuramente offerta dall'ontologia medesima: quella prodotta è finalizzata a dimostrare l'utilità di rappresentare la conoscenza

relativa agli AVM con questo strumento ed in quanto tale non ha pretese di esaustività. Un suo raffinamento finalizzato a trattare una maggiore varietà di prodotti AVM e delle loro caratterizzazioni sarebbe quindi auspicabile. Per quel che riguarda invece l'utilizzo della conoscenza rappresentata sotto forma di ontologia, sarebbe interessante sperimentare alcune delle tecniche ideate congiuntamente al linguaggio DAML+OIL. In questo modo sarebbe possibile realizzare motori di *information retrieval* basati sul ragionamento, più "intelligenti" degli strumenti attualmente a disposizione.

Altro punto di interesse è la realizzazione di strumenti che permettano una più facile manutenzione dell'ontologia AVM-DAML e del package JAVM. È indubbio che l'ontologia AVM-DAML definita possa subire aggiornamenti e/o modifiche per accogliere nuovi aspetti non precedentemente trattati e questo comporta una corrispondente ristrutturazione del package JAVM. Si prevede a breve termine di introdurre una serie di funzionalità che permettano la generazione automatica a partire da una certa versione di un'ontologia AVM-DAML del package JAVM in modo che il programmatore dovrà quindi definire i soli metodi che rapresentano la "business logic" delle istanze.

Dal punto di vista tecnologico, la prossima uscita della versione 2.0 del package Jena consentirà di gestire alcuni aspetti delle ontologie DAML+OIL attualmente non utilizzabili: la dichiarazione di *unicità* e/o *non ambiguità* di alcune proprietà. Una proprietà DAML+OIL è detta non ambigua quando costituisce un identificatore univoco (ogni istanza che presenta quella proprietà avrà per essa un valore diverso di quello delle altre istanze che la presentano), invece poiché in questo linguaggio le proprietà possono avere valori multipli, una proprietà è unica quando assume al più un valore per ogni istanza. Infine sarà incrementata anche l'efficienza in quanto sarà possibile mantenere in un database anziché in memoria le istanze delle varie classi.

References

1. BBC (British Broadcast Corporation) SMEF (Standard Media Exchange Framework). http://www.bbc.co.uk/guidelines/smef.
2. DARPA Agent Markup Language + Ontology Inference Layer. http://www.daml.org.
3. Dublin Core Metadata Initiative. http://dublincore.org.
4. EBU/UER (European Broadcasting Union/Union Europienne de Radio-Télévision) Metadata Exchange Standards. ttp://www.ebu.ch/pmc_meta.html.
5. The jena semantic web toolkit. http://www.hpl.hp.com/semweb/jena-top.html.
6. Multimedia content description framework, ISO/IEC JTC1/SC29/WG11 standard. http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm.
7. Oiled, ontology editor. http://oiled.man.ac.uk/.
8. OntoEdit, ontology editor. http://www.ontoprise.de/.
9. PRISM, Publishing Requirements for Industry Standard Metadata. http://www.prismstandard.org.
10. The protégé project. http://protege.stanford.edu/.
11. SMPTE (Society of Motion Picture and Television Engineers) Metadata Dictionary, SMPTE Recommended Practice RP 210.2.

A AVM-DAML-V4.3, v 4.3 2002/11/28

```
<!-- I commenti contengono definizioni piu‘ appropriate ma che non
     sono supportate dalla versione attuale di Jena -->



<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#"
>



<daml:Ontology rdf:about="">
<daml:versionInfo>$Id: AVM-DAML-V4.3.daml,v 4.3 2002/11/28 mdean
Exp $</daml:versionInfo>
<daml:imports
rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
</daml:Ontology>



<daml:Class rdf:ID="Entita">
</daml:Class>

<!--  <daml:UnambiguousProperty rdf:ID='ID'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'> -->

<daml:DatatypeProperty rdf:ID='ID'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Entita'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Persona">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='nome'>
<daml:domain rdf:resource='#Persona'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
```

```
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='cognome'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Persona'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Encoding">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<!-- <daml:ObjectProperty rdf:ID='storia'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'> -->

<daml:ObjectProperty rdf:ID='storia'>
<daml:domain rdf:resource='#Encoding'/>
<daml:range rdf:resource='#Operazione'/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID='tipo'>
<daml:domain rdf:resource='#Encoding'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:ID='supporto'>
<daml:domain rdf:resource='#Encoding'/>
<daml:range rdf:resource='#SupportoFisico'/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID='durata'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Encoding'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#duration'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='colore'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Encoding'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>
```

```
<daml:Class rdf:ID="Evento">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='canale'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Evento'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='momento'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Evento'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#dataTime'/>
</daml:DatatypeProperty>

<!-- <daml:ObjectProperty rdf:ID='encodingTrasmesso'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'> -->

<daml:ObjectProperty rdf:ID='encodingTrasmesso'>
<daml:domain rdf:resource='#Evento'/>
<daml:range rdf:resource='#Encoding'/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Interpretazione">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<!-- <daml:ObjectProperty rdf:ID='concettoInterpretato'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'> -->

<daml:ObjectProperty rdf:ID='concettoInterpretato'>
<daml:domain rdf:resource='#Interpretazione'/>
<daml:range rdf:resource='#ConcettoAstratto'/>
</daml:ObjectProperty>

<!-- <daml:ObjectProperty rdf:ID='encodingInterpretante'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'> -->

<daml:ObjectProperty rdf:ID='encodingInterpretante'>
<daml:domain rdf:resource='#Interpretazione'/>
<daml:range rdf:resource='#Encoding'/>
```

```
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID='da'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Interpretazione'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#time'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='a'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Interpretazione'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#time'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="ConcettoAstratto">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='copyright'>
<daml:domain rdf:resource='#ConcettoAstratto'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="ConcettoComposto">
<daml:subClassOf rdf:resource='#ConcettoAstratto'/>
</daml:Class>

<daml:ObjectProperty rdf:ID='regista'>
<daml:domain rdf:resource='#ConcettoComposto'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID='produttore'>
<daml:domain rdf:resource='#ConcettoComposto'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Film">
<daml:subClassOf rdf:resource='#ConcettoComposto'/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='titolo'
```

```
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>

<daml:domain rdf:resource='#Film'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:ID='attoreProtagonista'>
<daml:domain rdf:resource='#Film'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID='tecnicoAudio'>
<daml:domain rdf:resource='#Film'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID='trama'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Film'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Telefilm">
<daml:subClassOf rdf:resource='#ConcettoComposto'/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='nomeSerie'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Telefilm'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='titoloPuntata'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Telefilm'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='numeroPuntata'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#Telefilm'/>
```

```
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#positiveInteger'/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="ProgrammaTV">
<daml:subClassOf rdf:resource='#ConcettoComposto'/>
</daml:Class>

<daml:ObjectProperty rdf:ID='conduttore'>
<daml:domain rdf:resource='#ProgrammaTV'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID='ospite'>
<daml:domain rdf:resource='#ProgrammaTV'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>


<daml:Class rdf:ID="ElementoDiScaletta">
<daml:subClassOf rdf:resource='#ConcettoAstratto'/>
</daml:Class>

<daml:ObjectProperty rdf:ID='parteDi'>
<daml:domain rdf:resource='#ElementoDiScaletta'/>
<daml:range rdf:resource='#ConcettoComposto'/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID='descrizione'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#ElementoDiScaletta'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID='tipoEDS'>
<daml:domain rdf:resource='#ElementoDiScaletta'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:ID='personaCoinvolta'>
<daml:domain rdf:resource='#ElementoDiScaletta'/>
<daml:range rdf:resource='#Persona'/>
</daml:ObjectProperty>
```

```
<!-- La propriet\‘a ambientazione non \‘e unica perchsi possono
inserire pi\‘u stringhe per descrivere la stessa ambientazione. -->

<daml:DatatypeProperty rdf:ID=’ambientazione’>
<daml:domain rdf:resource=’#ElementoDiScaletta’/>
<daml:range
rdf:resource=’http://www.w3.org/2000/10/XMLSchema#string’/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Operazione">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<daml:ObjectProperty rdf:ID=’eseguitaDa’>
<daml:domain rdf:resource=’#Operazione’/>
<daml:range rdf:resource=’#Persona’/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID=’eseguitaIl’
rdf:type=’http://www.daml.org/2001/03/daml+oil#UniqueProperty’>
<daml:domain rdf:resource=’#Operazione’/>
<daml:range
rdf:resource=’http://www.w3.org/2000/10/XMLSchema#dataTime’/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Registrazione">
<daml:subClassOf rdf:resource=’#Operazione’/>
</daml:Class>

<!-- <daml:ObjectProperty rdf:ID=’eventoRegistrato’
rdf:type=’http://www.daml.org/2001/03/daml+oil#UniqueProperty’> -->

<daml:ObjectProperty rdf:ID=’eventoRegistrato’>
<daml:domain rdf:resource=’#Registrazione’/>
<daml:range rdf:resource=’#Evento’/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Estrazione">
<daml:subClassOf rdf:resource=’#Operazione’/>
</daml:Class>

<!-- <daml:ObjectProperty rdf:ID=’encodingManipolato’
rdf:type=’http://www.daml.org/2001/03/daml+oil#UniqueProperty’> -->
```

```
<daml:ObjectProperty rdf:ID='encodingManipolato'>
<daml:domain rdf:resource='#Estrazione'/>
<daml:range rdf:resource='#Encoding'/>
</daml:ObjectProperty>

<daml:Class rdf:ID="Unione">
<daml:subClassOf rdf:resource='#Operazione'/>
</daml:Class>

<daml:ObjectProperty rdf:ID='encodingUnito'>
<daml:domain rdf:resource='#Unione'/>
<daml:range rdf:resource='#Encoding'/>
</daml:ObjectProperty>

<daml:Class rdf:ID="SupportoFisico">
<rdfs:subClassOf rdf:resource="#Entita"/>
</daml:Class>

<daml:DatatypeProperty rdf:ID='tipoSupporto'
rdf:type='http://www.daml.org/2001/03/daml+oil#UniqueProperty'>
<daml:domain rdf:resource='#SupportoFisico'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

<!-- Vi possono essere pi\`u copie con collocazioni differenti -->

<daml:DatatypeProperty rdf:ID='collocazione'>
<daml:domain rdf:resource='#SupportoFisico'/>
<daml:range
rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
</daml:DatatypeProperty>

</rdf:RDF>
```