EBU

OPERATING EUROVISION

TECH 3351

EBU CLASS CONCEPTUAL DATA MODEL (CCDM)

Source: MIM-AI

Version 2.1

Geneva
January 2019

[*] Page intentionally left blank. This document is paginated for two sided printing

Introduction

'The EBU Class Conceptual Data Model (CCDM) is an ontology defining a basic set of classes and properties as a common vocabulary to describe business objects, e.g. programmes, articles and other types of content, and their relations in the business processes of media enterprises. Examples are programmes in their different phases of creation from commissioning to delivery, their associated rights or publication events, etc.

CCDM is a common framework and users are invited to, and should, further enrich the model with classes and properties fitting their needs more specifically. Properties for describing each of the objects can be found in EBUCore, or you are welcome to define your own.

This is version 2.1 of the "CCDM".

The CCDM has been purposefully designed as a minimum and flexible set of classes for a wide range of broadcasting applications, including archives, exchange and media service oriented production, semantic web and linked data.

The CCDM specification combines several aspects from existing models and specifications into a common framework. It has been built over several EBU attempts to represents broadcasting as a simple logical model. It has benefited from EBU work in metadata modelling (P-META and EBUCore) and semantic web developments. The distribution part has been designed to seek maximum mapping to TV-Anytime and the "BBC Programmes Ontology".

The CCDM ontology is represented in RDF/OWL and associated class diagrams.

More information on EBU metadata activities is provided on the EBU TECHNICAL website (http://tech.ebu.ch/metadata).

Terms and Conditions of Use

This EBU CCDM is freely available for all to use, but you should take note of the following:

DISCLAIMER: THIS SPECIFICATION AND ASSOCIATED RESOURCES IS PROVIDED BY THE COPYRIGHT OWNER "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS [SOFTWARE], EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Page intentionally left blank. This document is paginated for two sided printing

Contents

Introduction 3

1. Scope 8

1.1 Rationale 8

2. Class Conceptual Data Model 9

2.1 Main principles 9

2.2 Classes, Relationships and Properties 10

2.2.1 Legal, Commercial and Regulatory domain 10

2.2.1.1 Asset 10

2.2.1.1.1 AssetValue 12

2.2.1.2 Rights 12

2.2.1.3 Contract 13

2.2.1.3.1 ContractCost 14

2.2.2 Editorial Domain 14

2.2.2.1 EditorialObject 15

2.2.2.2 TimelineTrack 18

2.2.2.3 Location 18

2.2.2.4 Event 19

2.2.3 Entity domain 19

2.2.3.1 Agent 20

2.2.3.2 Person 21

2.2.3.3 Organisation 21

2.2.3.4 Crew 21

2.2.3.5 Role 21

2.2.3.6 Artefact 22

2.2.4 Production Domain 22

2.2.4.1 Resource 23

2.2.4.2 MediaResource 24

2.2.4.3 Track 25

2.2.4.4 Format 25

2.2.4.5 Essence 27

2.2.4.6 PhysicalResource 28

2.2.4.7 Artefact 28

2.2.4.8 ProductionJob 28

2.2.4.9 ProductionDevice 29

2.2.5 Distribution Domain 30

2.2.5.1 PublicationEvent 31

2.2.5.2 Service 33

2.2.5.3 ConsumptionDeviceProfile 33

2.2.6 Consumption Domain 34

2.2.6.1 ConsumptionEvent 35

2.2.6.2 ConsumptionDevice 35

2.2.6.3 ConsumptionLicence 36

2.2.6.4 Consumer 37

2.2.6.5 Account 37

2.2.6.6 ResonanceEvent 38

2.2.7 Planning Domain 38

2.2.7.1 Campaign 39

2.2.7.2 PublicationPlan 40

2.2.7.3 ProductionOrder 40

2.2.7.4 Audience 41

2.2.7.5 Resonance 41

2.2.8 Financial Domain 42

2.2.8.1 AssetValue 42

2.2.8.2 ContractCost 43
3. Implementation Guidelines / Questions & Answers 44
3.1 General remarks 44
3.2 Examples provided by SRG SSR, Swiss Confederation 44
3.2.1 Modelling Different Viewpoints with CCDM 44
3.2.2 CCDM as a Comprehensive Representation of Business Objects 46
3.3 Example provided by TV2, Norway 49
3.4 The total class diagram 49
3.4 The RDF ontology 49
3.5 More questions? 50
4. CCDM Compliance 50
5. Download Zone 50
6. Licensing regime 50
7. Maintenance 50
8. Useful links 50
Annex A: EBU CCDM ontology 52

Page intentionally left blank. This document is paginated for two sided printing

EBU Class Conceptual Data Model (EBU CCDM)

EBU Committee	*First Issued*	*Revised*	*Re-issued*
MIM	October 2012	December 2018	

Keywords: Class, Model, Metadata, Business, Object, Radio, Television, Production, SOA, Semantic Web, Linked Data, Internet, Web Publishing.

1. Scope

The EBU Class Conceptual Data Model (CCDM) is an ontology defining a basic set of classes and properties as a common vocabulary to describe business objects in their different phases of creation from commissioning to delivery, i.e. the full lifecycle of a business process. CCDM is a common framework and users are welcome to further enrich the model with Classes and properties fitting their needs more specifically.

The CCDM has deliberately been designed as a minimum and flexible set of classes for a wide range of applications including but not restricted to archives, exchanges, media service oriented production, broadcasting, Internet delivery, Semantic Web modelling and Linked Open Data (LOD).

This specification is a class model, an ontology, and not a metadata specification. Metadata properties and datatypes (other than the relationships between Classes) are **indicative**. Users willing to adapt the CCDM model to their needs are invited to describe CCDM classes and custom extensions either using properties from EBU Tech 3293 (EBUCore metadata set) or other metadata specifications (e.g. TV-Anytime or in-house metadata schemes).

The CCDM specification is combining several aspects from existing models and specifications into a common framework. It has been built over several EBU attempts to represents broadcasting as a simple logical model. It has benefited from EBU work in metadata modelling (P-META and EBUCore) and semantic web developments. The distribution part has specifically been designed to seek maximum mapping to TV-Anytime and the "BBC Programmes ontology".

The CCDM ontology is represented in RDF/OWL.

1.1 Rationale

It is vital for content providers and broadcasters to have a well-defined class model. This is a necessary step towards:

- Greater understanding of the business models and workflows;
- Process optimisation with easier and more reliable data exchange;
- A simpler and rationalised description of Media Classes;
- The easier implementation of media service-oriented production architectures;
- The adoption of new information management models such as Semantic Web and Linked Data (enrichment, improved searching and ubiquity).

The CCDM has been designed to let implementers adapt the names of the Classes and their Relationships to their respective modelling needs. Each organisation is encouraged to make its proper analysis and to create its own model starting from the CCDM framework as a common basis for comparison with models from other CCDM implementers.

2. Class Conceptual Data Model

2.1 Main principles

The EBU CCDM is composed of:

- Classes: directly related (e.g. a programme, a part, a clip, a track) or associated (e.g. a person, a location) to media.
  - Note: equivalent to the notion of class used in semantic web modelling (see RDF and OWL Primers), also referred to as 'Business Objects' or ‘concepts’ in certain projects, see also http://protege.stanford.edu/publications/ontology_development/ontology101.pdf . W3C’s Media-Ontology (MA-ONT) is based on the CCDM class model (http://www.w3.org/ns/ma-ont.rdf).
- Relationships: linking Classes (e.g. 'Programme hasContributor Person')
  - Note: equivalent to the notion of *objectProperties* used in semantic web modelling (see RDF and OWL Primers)
- Properties: defining intrinsic characteristics of Classes (e.g. 'bitrate' expressed as an integer or a person 'name' expressed as a string)
  - Note: equivalent to the notion of *dataProperties* used in semantic web modelling (see RDF and OWL Primers)

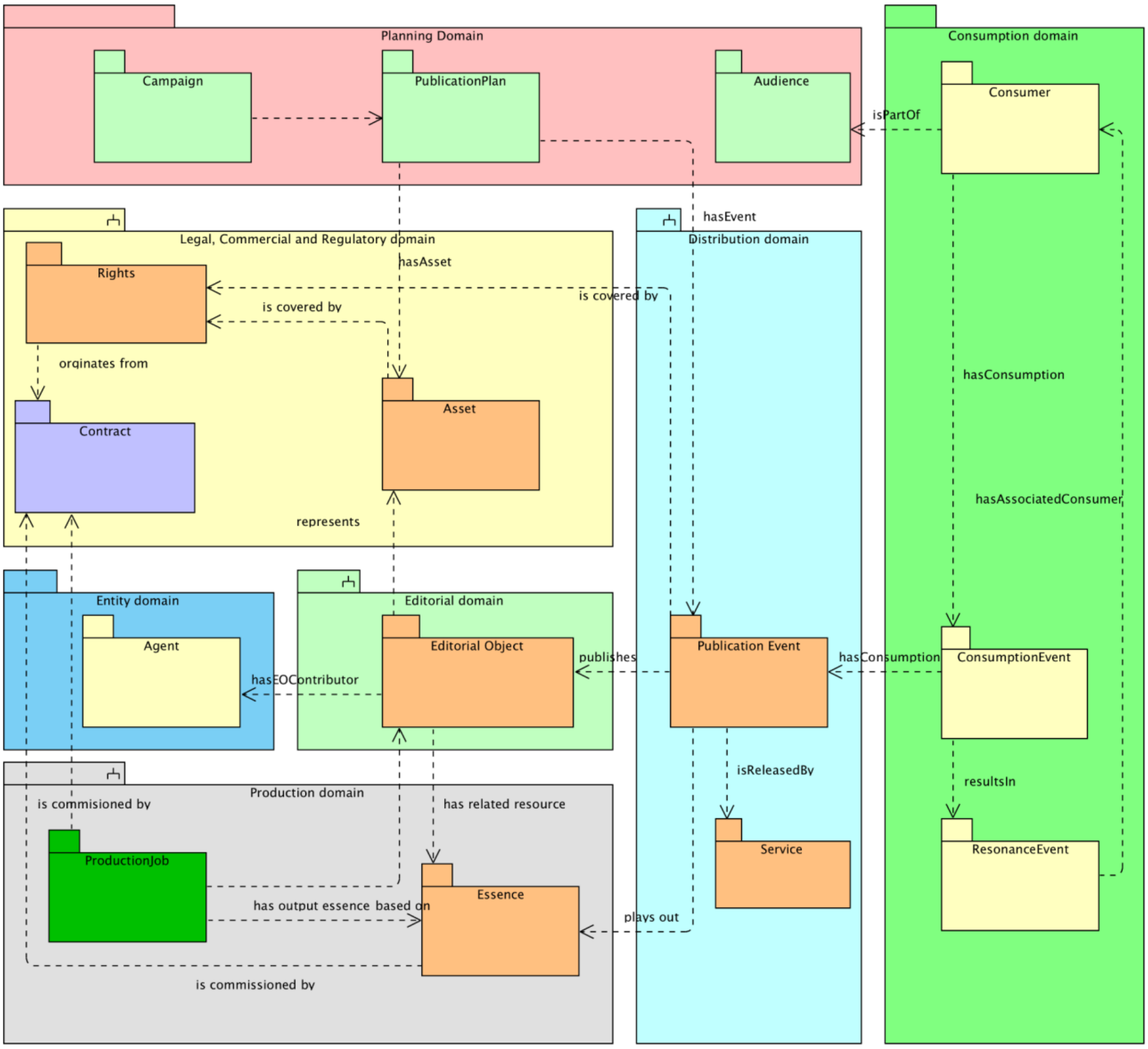


Figure 1: CCDM domains

As shown in **Figure 1**, the model is defined around seven main domains:

- **Planning Domain** is where the demand is defined and met by a strategy in form of a *PublicationPlan*, where productions are commissioned, and where *Resonance* from the *Audience* is taken into account.
- **Legal, Commercial and Regulatory domain** is where *Contracts*, intellectual property and other rights associated to content and its manifestations are being managed.
  The central class of the Legal Domain is the *Asset*, which establishes the association of an *EditorialObject* with Intellectual Property and Rights related information.
- **Distribution Domain** is where any form of publishing, play-out or distribution is covered. The central Class is the *PublicationEvent* that plays out an *Essence*, i.e. the media object that was the result of the *ProductionJob*.
- **Editorial Domain** is where concept related and content related information is being managed. Furthermore, all editing decisions are represented here. The *EditorialObject* is the central class of the domain. It can be grouped, and it can be ordered on a timeline.
- **Entity domain** is a where actors/contributors, like persons and companies are described.
- **Production Domain** is where production orders are realised through the acquisition of the necessary *MediaResources* (e.g. manufacturing an object through the *ProductionJob*, purchase or retrieval of material) according to the production plan. *MediaResources* ready for publication use the *Essence* class for connecting the content to a certain publication.
- **Consumption Domain** is where the consumption of media is modelled. Important classes in this domain is the *ConsumptionEvent*, that correspondents with the *PublicationEvent* in the *DistributionDomain*.

The EBU CCDM has been designed to let users adapt the names of Classes and relationships to their respective modelling needs. For example, a class '*EditorialObject*' can be of type 'programme', 'item' or 'shot', but it can also represent a group 'series', 'serial' or 'season'. The definition of appropriate properties is left to the user. A core set of classes and properties is proposed in EBU Tech 3293, EBUCore, or in other metadata specifications (e.g. TV-Anytime or in-house metadata schemes).

2.2 Classes, Relationships and Properties

See **Figure 1**, which illustrates the relationships between domains and objects.

2.2.1 Legal, Commercial and Regulatory domain

It is the domain in which intellectual property, rights, regulations, legal constraints, compliance standards, and contracts are being managed and associated to a *MediaResource* and / or an *EditorialObject*, and by inference to a *PublicationEvent* (incl. exploitation and distribution conditions), to define an *Asset*. The domain also covers the commissioning of productions and material.

The central class of the domain is the *Asset* that acts like a conjunction between a set of *rights* or legal constraints and an *EditorialObject*.

2.2.1.1 Asset

Definition:

The class *Asset* is an object to which an identifier will be associated at commissioning. It will serve as a central reference point to manage rights associated to *EditorialObjects*, *MediaResources* or *Essences*, and - by inference - *PublicationEvents* (distribution and exploitation conditions).

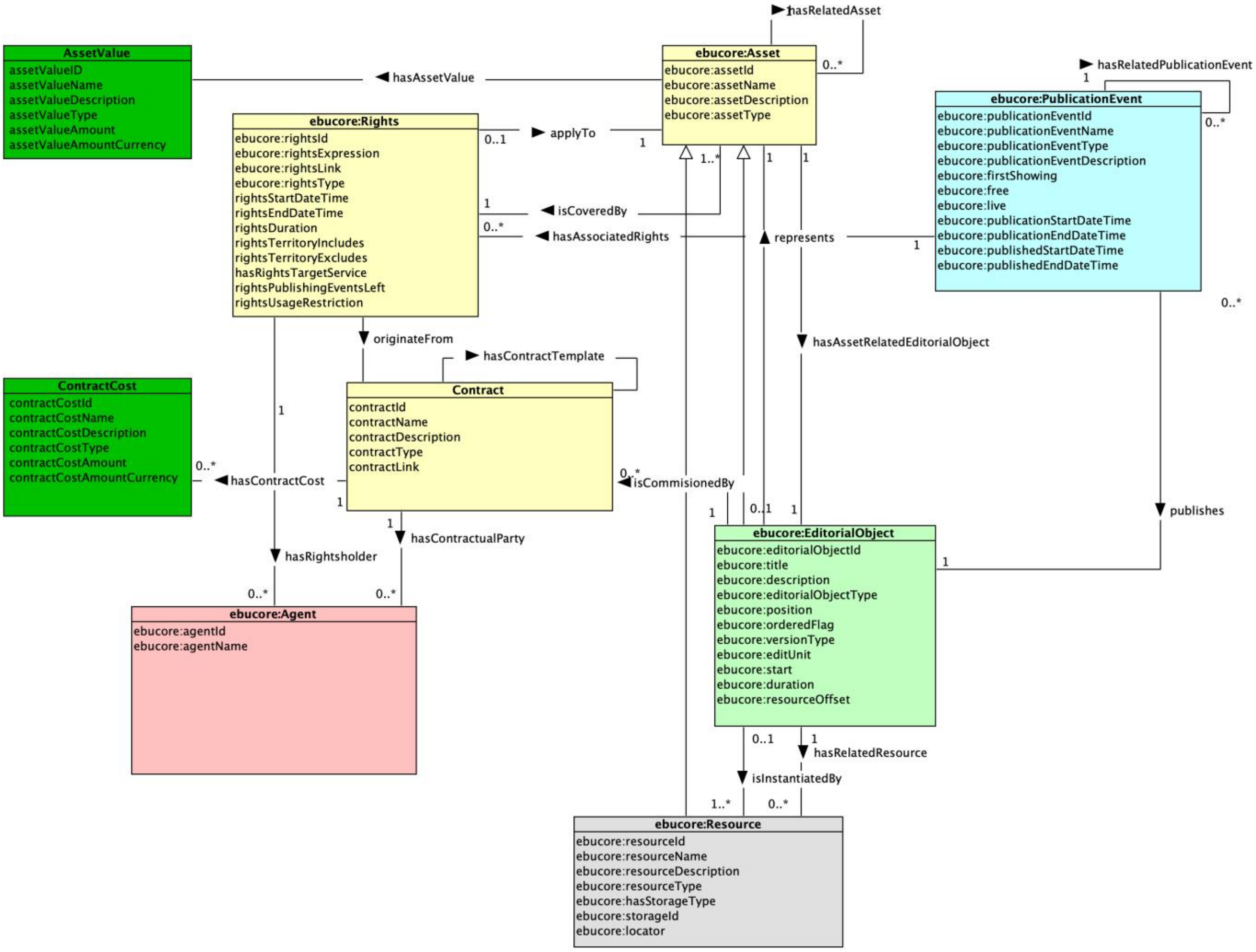


Figure 2: The Asset

Remember that the *MediaResources* or *Essences* will, in this model, always be the representation/instantiation of an *EditorialObject*.

The Asset class is also a superclass for *EditorialObject* and *Resource* in the way that Rights information can be added to those classes for a simple representation.

Example:

The CCDM model allows the association of Rights to an *EditorialObject* representing an *Essence*.

Class relations	
hasAssetRelated*EditorialObject*	A pointer to the *EditorialObject* that the Asset links to its *Rights*
hasAssetRelatedResource	A pointer to the Resource that the Asset links to its *Rights*
hasRelatedAsset	A pointer to another asset (e.g. a TV Series) that the *Asset* links to
isCoveredBy	A pointer to the *Rights* associated to the *EditorialObject*
hasAssetValue	A pointer to the *AssetValue* associated with the Asset
Etc.	Other class relationships can be associated to an *Asset*. See EBU Tech 3293, EBUCore.
Class Properties	
assetId	An identifier associated with the *Asset*
assetName	A name given to the *Asset*
assetDescription	A description of what the *Asset* represents
assetType	The type assigned to the *Asset*
Etc.	Other properties can be associated to an *Asset*. See EBU Tech 3293, EBUCore.

2.2.1.1.1 AssetValue

Definition:

The class *AssetValue* is an object that is used to specify the value of an Asset.

Class Properties	
assetValueId	An identifier associated with the *AssetValue*
assetValueName	A name given to the *AssetValue*
assetValueDescription	A description of what the *AssetValue* represents
assetValueType	The type assigned to the *AssetValue*
assetValueAmount	The actual estimated value of the Asset
assetValueAmountCurrency	The currency in which the value is expressed
Etc.	Other properties can be associated to an *AssetValue*.

2.2.1.2 Rights

Definition:

The class *Rights* defines rights that originate from a contract. The *Rights* are associated to a *MediaResource* through the definition of an *Asset*.

Class relations	
applyTo	A pointer to the *Asset*, which in turn has *EditorialObject*, to which the *Rights* apply.
orginateFrom	A pointer to the *Contract* granting the *Rights*
hasRightsholder	The *Agent* related to the *Rights*. Can be sub-classed to specify the kind of relationship.
Etc.	Other class relationships can be associated to *Rights*. See EBU Tech 3293, EBUCore
Class Properties	
rightsID	An Identifier associated with the *Rights*.
rightsExpression	The expression of *Rights*.
rightsLink	A link to e.g. a web resource where the *Rights* terms can be found.
rightsType	A type associated to *Rights* e.g. licensing terms.
rightsStartDateTime	The start of the time interval where the Rights is valid
rightsEndDateTime	The end of the time interval where the Rights is valid
rightsDuration	The extend of a rights period, when it is not expressed using rightsEndDateTime
rightsTerritoryIncluding	Territory covered by the Rights
rightsTerritoryExcluding	Territory excluded from the Rights
hasRightsTargetService	The Service associated with the Rights
rightsPublishingEventsLeft	The number of publishing events left covered by the Rights
rightsUsageRestriction	Restrictions and other constraints in how the material can be used
Etc.	Other properties can be associated to *Rights*. See EBU Tech 3293, EBUCore.

2.2.1.3 Contract

Definition:

The class *Contract* represents any legal document covering *Rights* - or commissioning issues. This object/class covers the production order and sales order combined. The *Contract* connects the *Rights* to any *RightsHolders*. A *Contract* defines one or more set of *Rights*.

Class relations	
hasContractualParty	A list of the parties involved with the *Contract*. Can be specified by a subproperty or a subclass to describe the relationship in more detail.
hasContractTemplate	Relation to the template the *Contract* is derived from
hasContractRelatedCost	A pointer to the *ContractCost* associated with the *Contract*
Etc.	Other class relationships can be associated to a *Contract*. See EBU Tech 3293, EBUCore
Class properties	
contractID	An Identifier associated with the *Contract*.
contractName	The name given to a *Contract*.
contractDescription	A description of the *Contract*.
contractType	The type of Contract.
contractLink	URL pointing to a document describing the contract

Etc.	Other properties can be associated to a *Contract*. See EBU Tech 3293, EBUCore.

2.2.1.3.1 ContractCost

Definition:

The class *ContractCost* is an object that is used to specify the cost associated with a Contract.

Class Properties	
contractCostId	An identifier associated with the ContractCost
contractCostName	A name given to the ContractCost
contractCostDescription	A description of what the *ContractCost* represents
contractCostType	The type assigned to the ContractCost
contractCostAmount	The actual cost figure
contractCostAmountCurrency	The currency in which the cost figure is expressed.
Etc.	Other properties can be associated to a *ContractCost*.

2.2.2 Editorial Domain

The Editorial Domain is the domain within which a concept is defined and commissioned before fabrication and distribution. All metadata related to the idea of a programme (e.g. content, format, purpose, audience, schedule window), related to the content of the programme (e.g. titles, subjects, contributors, locations, events) and all editing decisions are represented in the respective classes.

The central class in the Editorial Domain is the *EditorialObject*.

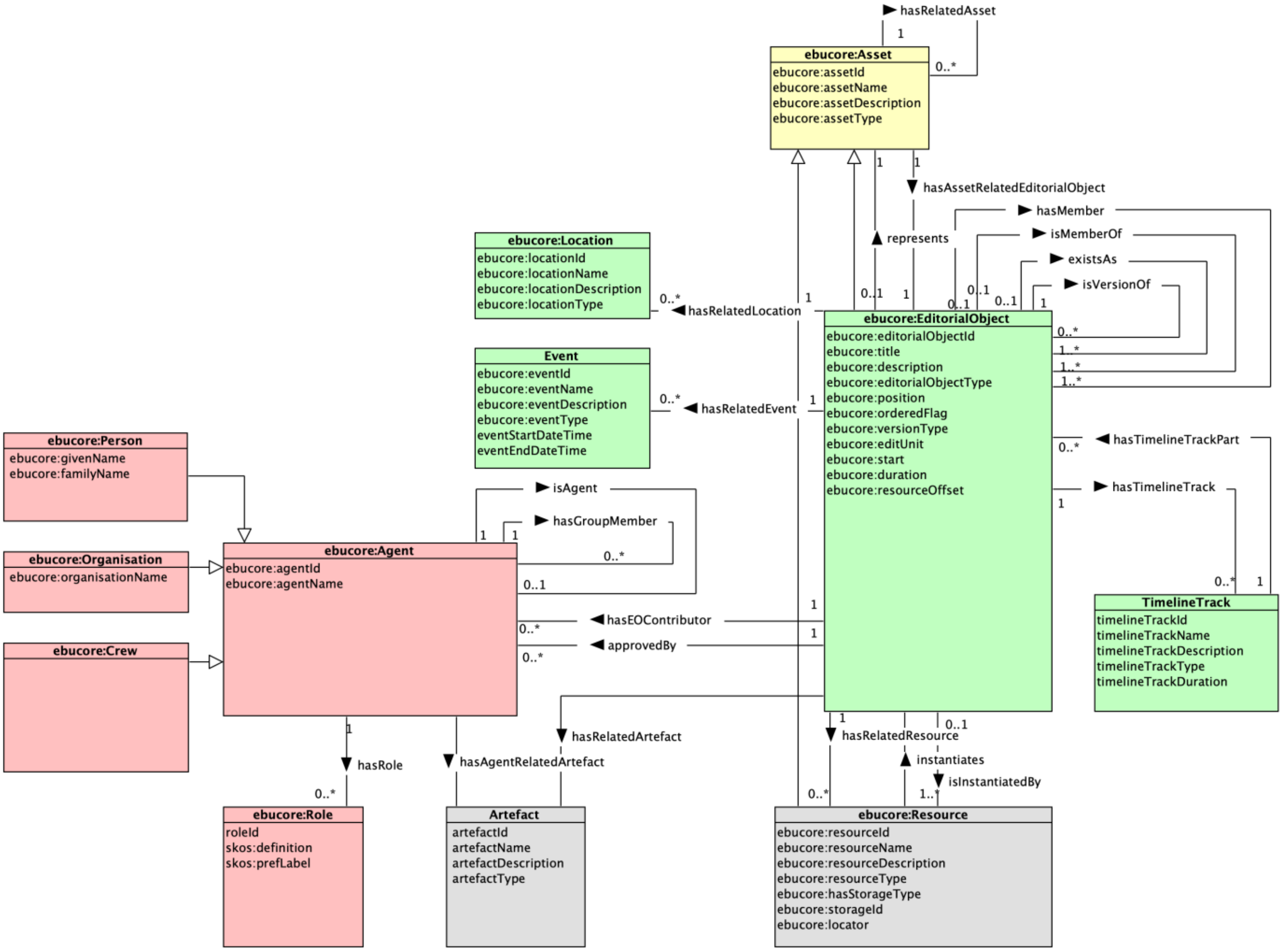


Figure 3: Classes around the *EditorialObject*

2.2.2.1 EditorialObject

Definition:

The class *EditorialObject* describes an idea or story and will be used to transform a concept into an editorial definition of a *MediaResource* before fabrication (Production Domain) and Distribution (Distribution Domain). An *EditorialObject* is a set of descriptive metadata summarising e.g. editing decisions.

An *EditorialObject* can be a group.

An *EditorialObject* can also be a part of another *EditorialObject*, which is defined by its start time and duration.

EditorialObjects can be ordered either as groups or as items on a timeline.

Examples:

Programme, item, shot, part, chapter, segment, and where the group properties are in use: series, serial, compilation, collection, item group, item block.

A simplified use-case:

A TV news broadcast consists of two news items. One news item contains the last ten seconds of a one minute long interview taken from another source (i.e. from 50’’ to 60’’). This could be modelled as follows:

- The *NewsBroadcast* is linked to a *MediaResource* using the instantiates-property
- The *NewsItems* are linked to the *NewsBroadcast* using a *TimelineTrack*.

- The *InterviewPart* is linked to the *NewsItem* using the *hasMember*-property. Start and Duration are properties within the *InterviewPart* indicating its appearance within the *NewsItem2*.
- The *InterviewPart* is linked to its original source using the *existsAs*-property
- The Interview instantiates a *MediaResource*, which in turn is linked from the *MediaResource* of the *NewsBroadcast* using the *hasSource*-property
- Representation of segmentation: *TimelineTracks* are preferred over *hasPart*-properties, when a rundown is needed, e.g. for playout.

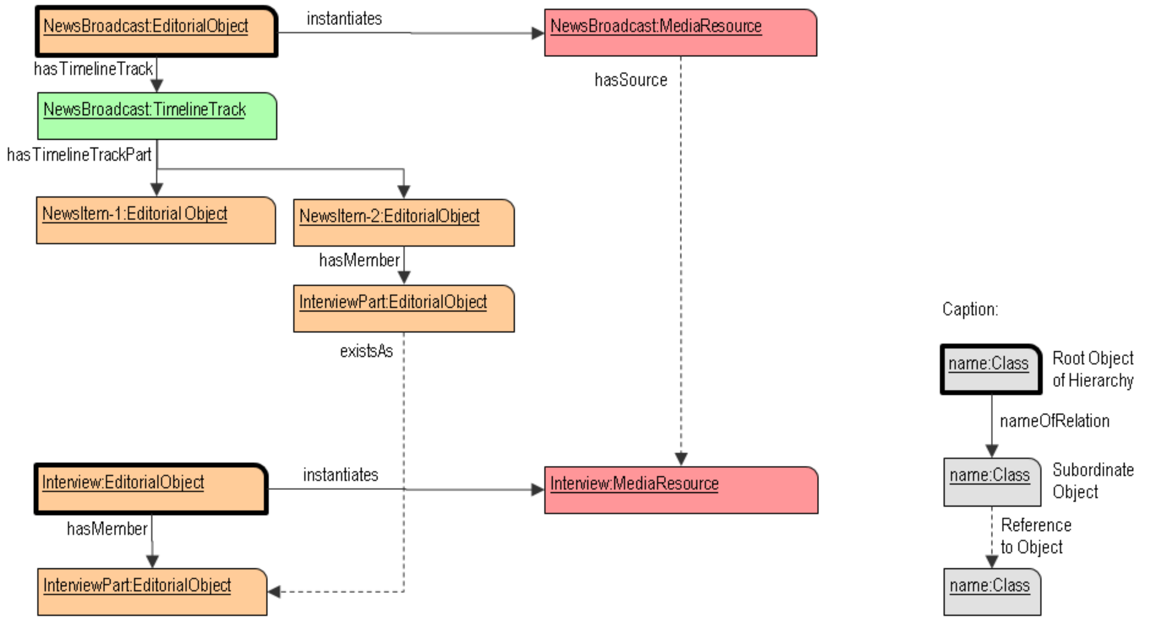


Figure 4: Illustration of use-case

Class relations	
isMemberOf	A list of Groups that the *EditorialObject* is a member of.
hasMember	A list of *EditorialObjects* that the *EditorialObject* contains that is not a part of a timeline. Series-episode is an example of such a relationship
hasRelatedResource	A relationship to identify a *Resource* that are related to the *EditorialObject*
isInstantiatedBy	A relationship to identify the *Resource* that instantiates the *EditorialObject*
hasEOContributor	The Agent(s) having contributed to the realisation of the *EditorialObject*. The contribution is characterised by the Agent Role. Agent is a non-media Class described in another section of this document. The "hasEOContributor" property can be extended with subproperties for different more specific roles, such as hasEOCreator, hasEODirector.
approvedBy	An actor, like the editor of the day, that approves the *EditorialObject* for publication
hasRelatedLocation	Optionally, one (or more) *Location* related to the *EditorialObject* characterised by its type (e.g. shooting or fictional).
hasRelatedEvent	Optionally, one (or more) *Event* related to the *EditorialObject* characterised by its type (e.g. sport event / meeting).
represents	An *EditorialObject* represents an *Asset*.
hasAssociatedProductionJob	A *ProductionJob* represents a production process through which an *EditorialObject* is being instantiated into a *MediaResource* and / or and *Essence*.
isVersionOf	To identify *EditorialObjects* presenting alternative version of the content.
existsAs	To identify *EditorialObjects* representing alternative representations of the content
hasTimelineTrack	To associate a *TimelineTrack*, e.g. a *RunDown*, with an *EditorialObject* itself constituted of other *EditorialObjects*.
isCommisionedBy	The *Contract* that commissions the *EditorialObject*
hasRelatedResonanceEvent	Used when e.g. an interactive Tweet from a consumer is being used on-screen in a television show, - a *ResonanceEvent* triggers and is the base for the creation a new *EditorialObject*.
hasRelatedArtefact	A relationship to an Artefact related to the *EditorialObject*
Etc.	Other class relationships can be associated with an *EditorialObject*. See EBU Tech 3293, EBUCore.
Class hierarchy	
superclass	*Asset* is the superclass for *EditorialObject*
Class Properties	
editorialObjectType	The type of *EditorialObject* e.g. Programme, Item.
editorialObjectId	Optionally one (or more) identifier attributed to the *EditorialObject*.
title	The main Title by which of the *EditorialObject* is known. As an example.
description	Optionally one (or more) description of the *EditorialObject*. As an example.

position	The position or index of the *EditorialObject* in an *EditorialObject* of type 'rundown', or in an ordered Group
orderedFlag	If 'true', a flag which indicates that the members of the *EditorialObject* are ordered (e.g. membership is subject to a strict sequence such as episodes in a series).
versionType	A string to optionally identify the version of the *EditorialObject* such as lengthened, shortened, signed, closed-captioned, etc.
start	The starting point of the member, i.e. the part, in an *EditorialObject* or in a *TimelineTrack*.
duration	The duration of the member in an *EditorialObject* or in a TimelineTrack.
editUnit	The unit used to express start, duration and *resourceOffset*.
resourceOffset	The start offset of the related resource, used if the related resource is not used from its start.
Etc.	Many other properties can be associated with an *EditorialObject*. See EBU Tech 3293, EBUCore.

2.2.2.2 TimelineTrack

Definition:

A “TimelineTrack” is used to define timelines, i.e. a time related sequence of *EditorialObjects* (or Part of *EditorialObjects*).

Class relations	
hasTimelineTrackPart	To identify the Parts of a *TimelineTrack*. I. e. *EditorialObjects* with a start time and duration.
Etc.	Many other relationships can be associated with an *EditorialObject*. See EBU Tech 3293, EBUCore.
Class properties	
timelineTrackID	The identifier attributed to a *TimelineTrack*.
timelineTrackType	E.g. rundown or other types not defined as subclass in the specification
timelineTrackName	The name given to the timeline
timelineTrackDescription	The description of a *TimelineTrack*.
timelineTrackduration	The duration of the *TimelineTrack* in the *EditorialObject*.
Etc.	Many other properties can be associated with an *EditorialObject*. See EBU Tech 3293, EBUCore.

2.2.2.3 Location

Definition:

The class *Location* is used to define the locations, e.g. spatial coverage of the story or recording locations like studios or in the field, associated with the *EditorialObjects* (or Part of *EditorialObjects*).

Class relations	
hasLocationRelatedEvent	An *Event* related to a *Location*.
Etc.	Many other relationships can be associated with an *Location*. See EBU Tech 3293, EBUCore.

Class properties	
locationId	To identify a *Location* in a system of defined locations.
locationName	The name of a *Location*.
locationDescription	The description of a *Location*.
locationType	The type of *Location*.
Etc.	Many other properties can be associated with a Location. See EBU Tech 3293, EBUCore (incl. GPS coordinates) or *GeoNames*.

2.2.2.4 Event

Definition:

The class *Event* is used to define the event that the *EditorialObject* covers.

Examples:

Olympic Games 1994, General election, etc.

Class relations	
hasEventRelatedLocation	A *Location* related to an *Event*.
Etc.	Many other relationships can be associated with an *Location*. See EBU Tech 3293, EBUCore.
Class properties	
eventId	To identify the *Event*.
eventName	The name of an *Event*.
eventDescription	The description of an *Event*.
eventType	The type of an *Event*.
eventStartDateTime	The time where an *Event* starts
eventEndDateTime	The time where an *Event* ends
eventDuration	The duration of an event
Etc.	Many other properties can be associated with an *Event*. See EBU Tech 3293, EBUCore.

2.2.3 Entity domain

This is where actors, like persons and companies are described. The classes can be connected to any other class in the model where there is a need for describing ownership or contribution to data. The Agent class is specialized into *Person*, *Organisation* and *Crew*, used for needs of description of the data.

E.g. the in the planning stage we like to describe the need for the job functions in the production crew. At this stage the jobs are not assign to any people yet. So, we are using the *Crew* class for describing the functions that are needed for a production. As the planning evolves further, each of the *Crew* will be assigned an isAgent relation to a *Person*, containing the real name.

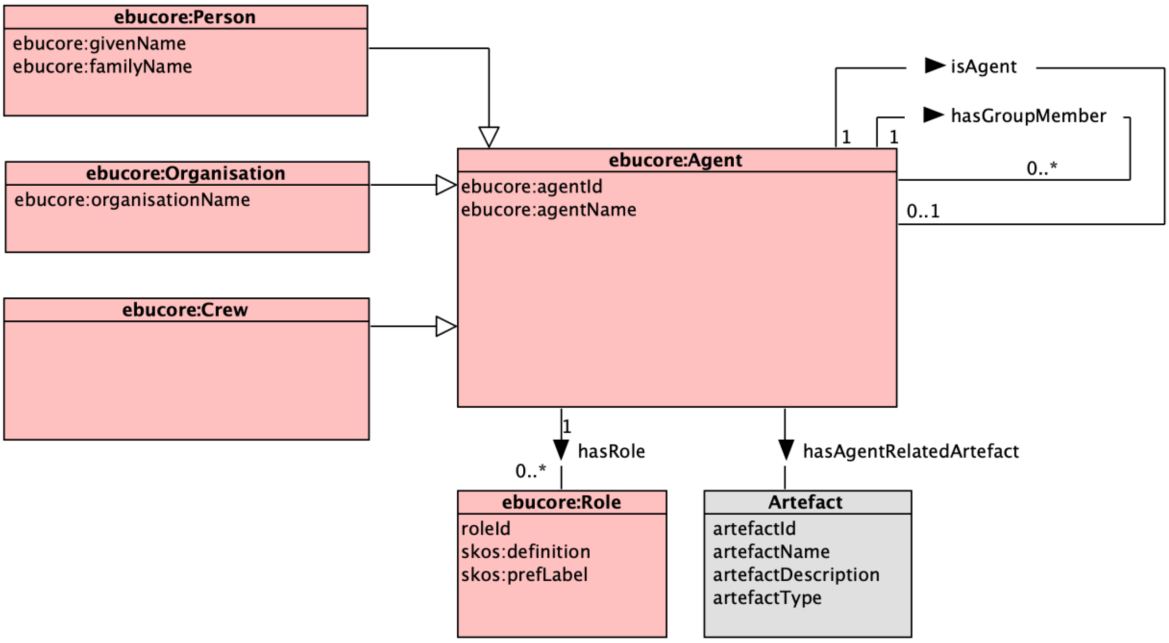


Figure 5: Entity Domain

2.2.3.1 Agent

Definition:

The class *Agent* is either a Contact/Person/Crew or Organisation to which is associated a *Role* corresponding to the contribution the *Agent* brings to the realisation of a *MediaResource* or *EditorialObject*.

Examples:

Examples of *Agent's Role* are 'producer', 'cameraman' or 'actor'.

Class relations	
hasRole	The *Role* of the *Agent*. *Role* refines "hasContributor". Alternatively, a user can decide to add new class and associated relationships as contributions to an *EditorialObject* e.g. "hasContributorCreator", "hasContributorComposer", etc., which in turn will be refined with "hasRole" *Role*.
isAgent	The relation is used for connecting the Person, Organisation and Crew part of the Agent data.
hasGroupMember	Used for connecting a team or a group to its members.
hasAssociatedArtefact	Relation to an Artefact associated with the Agent. E. g. a costume.
Etc.	Other class relationships can be associated with an *Agent*. See EBU Tech 3293, EBUCore.
Class Properties	
agentId	An identifier for the *Agent*.
agentName	The display name given to the agent.
Etc.	Other class Properties can be associated with an *Agent*. See EBU Tech 3293, EBUCore.

2.2.3.2 Person

Definition:

The class *Person* stores the personal data such as name for an agent. The class can be extended with contact data from EBU Core.

Class relations	
Subclass	The *Person* class is a subclass of *Agent*.
Class Properties	
givenName	The name given to a person. This is an example of how properties from EBUCore are used in CCDM
familyName	The family name of a person.
Etc.	Other class Properties can be associated with an *Person*. See EBU Tech 3293, EBUCore.

2.2.3.3 Organisation

Definition:

The class *Organsisation* stores the name and other data for a company. The class can be extended with contact data from EBU Core.

Class relations	
Subclass	The *Organisation* class is a subclass of *Agent*.
Class Properties	
organisationName	A name associated with an organisation.
Etc.	Other class Properties can be associated with an *Organisation*. See EBU Tech 3293, EBUCore.

2.2.3.4 Crew

Definition:

The class *Crew* stores the job function of an unspecified crew member. The class is typically used for resource planning. Crew is a subClass of Agent and uses Agent's hasRole to specify the job function.

Examples:

Examples of *Crew* are 'producer', 'cameraman' etc.

Class relations	
Subclass	The *Crew* class is a subclass of *Agent*.
hasRole	To define the job function.
Class Properties	
Etc.	Other class Properties can be associated with a *Crew*. See EBU Tech 3293, EBUCore.

2.2.3.5 Role

Definition:

The *Role* played by an *Agent*. A *Role* will be identified e.g. by a concept from a SKOS Classification

Scheme. *Role* is therefore to be considered as a class, i.e. a subClass of SKOS Concept.

Example:
A Contact may be an actor.

Class Properties	
roleId	Identifier attributed to a *Role*, preferably from a defined list of *Roles* (e.g. a SKOS ConceptId)
Etc.	Other class Properties can be associated with a *Role*. See EBU Tech 3293, EBUCore.

2.2.3.6 Artefact

See section 2.2.4.7.

2.2.4 Production Domain

The Production Domain is the domain, within which production orders are realised through the acquisition of *MediaResource* (e.g. manufacturing an object through a *ProductionJob*, purchase or retrieval of material).

The central class in the Production Domain is the *MediaResource* and its *Essence* subclass.

MediaResources ready for publication use the *Essence* class for connecting the content to a certain publication.

A *MediaResource* has always a relation to an *EditorialObject* (Editorial Domain) describing its content. The *Essence* is a manifestation of a *MediaResource* in a particular Format that is destined for publication. The *Essence* is the result of a *ProductionJob* and is a subclass of *MediaResource* and inherits all of its properties such as *Format*, *Location* and *ProductionDevice*.

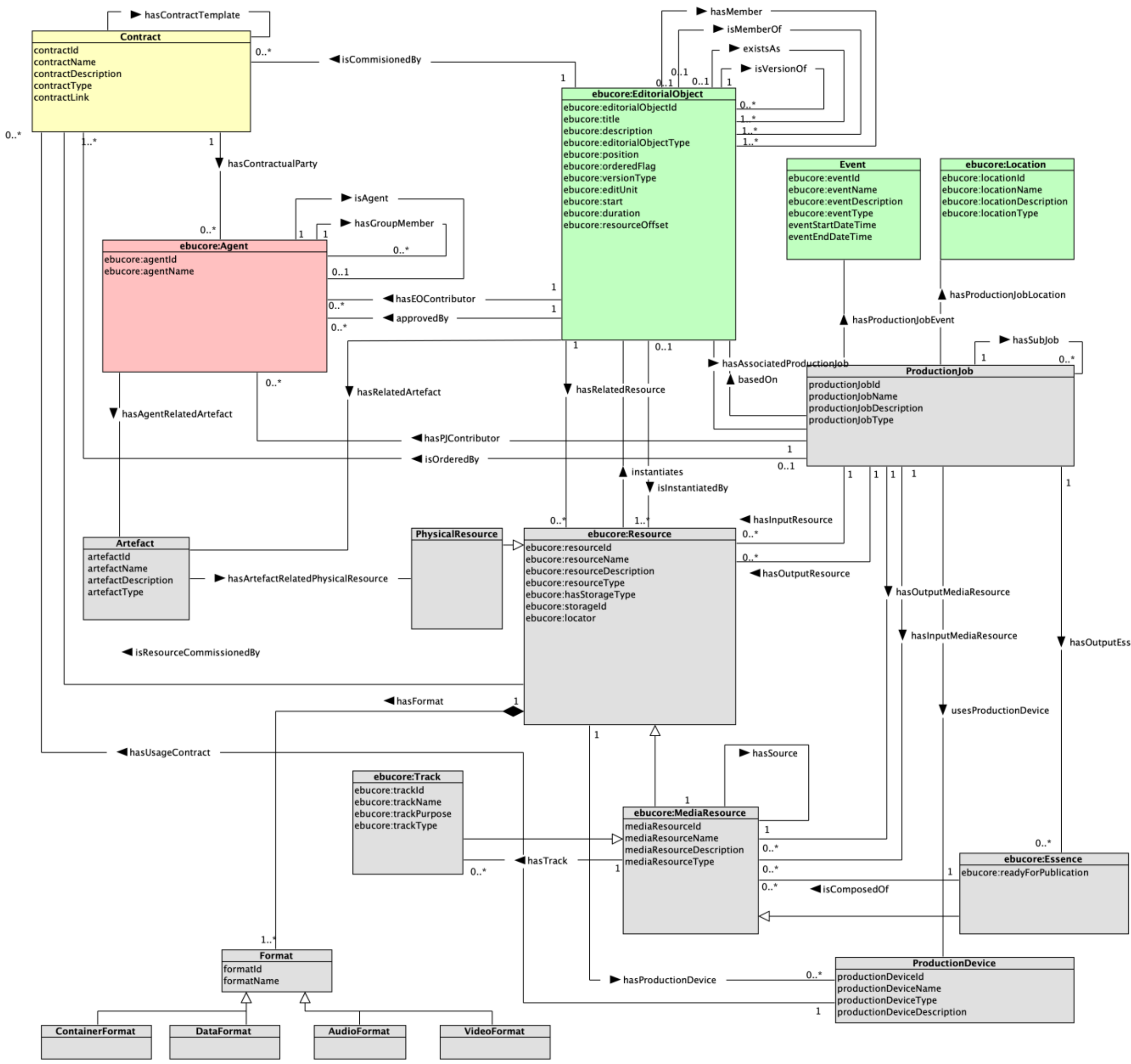


Figure 6: ***MediaResource***

2.2.4.1 Resource

Definition:

Resource is a generic concept used in relation to a production and going beyond the notions of *MediaResource* or *Essence*. It is defined by an *EditorialObject* (Editorial Domain). It has a locator indication where the *Resource* can be retrieved.

The class *Resource* is a subclass of *Asset*.

Examples:

A pdf file used as part of the research, a manuscript stored in a repository etc.

Class relations	
hasFormat	E.g. the composition of a *Resource*. A *Resource* can exist in one or more formats.
instantiates	Relation to the *EditorialObject* that describes the *Resource*.
isResourceCommissionedBy	The *Contract* through which the creation of the *Resource* has been commissioned.
hasProductionDevice	The *ProductionDevice* used for the creation of the *Resource*
Etc.	Other class relationships can be associated with a *Resource*. See EBU Tech 3293, EBUCore.
Often used subclasses	
Subclass	*MediaResource* is a sub-class of *Resource*, used to specify additional attributes typical for media files.
Subclass	PhysicalResource is a sub-class of *Resource*, used where the object that instantiates the *EditorialObject* is a physical thing.
Class Properties	
resourceId	Unique Identifier e.g. a UUID, UMID, URI etc. It can be generated or assigned by the business process or it can be extracted from the content.
resourceName	The name given to a *Resource*.
resourceDescription	A description of a *Resource*.
resourceType	The type of *Resource*.
storageId	The identifier of the storage.
hasStorageType	A definition of the type / structure of storage where the *Resource* is stored.
locator	This indicates where a particular *Resource* can be found and accessed.
Etc.	Many other properties can be associated to a *Resource*. See EBU Tech 3293, EBUCore.

2.2.4.2 MediaResource

Definition:

"*MediaResource*" is commissioned for production. It is defined by an *EditorialObject* (Editorial Domain). It can be represented by one or more *Essences* e.g. in a particular *Format* for distribution on a specific delivery media. The *MediaResource* is a subclass of *Resource*.

Many properties can be found under the format element of EBUCore for describing the technical metadata of a *MediaResource*

Class relations	
hasSource	The relation to a *MediaResource* acting as a source of the *MediaResource*. E.g. an analogue tape that is the source of a file
hasTrack	The relation to the *Tracks* that the *MediaResource* are divided into.
Etc.	Other class relationships can be associated with a *MediaResource*. See EBU Tech 3293, EBUCore.
Often used subclasses	
subclass	*Track* is a sub-Class of *MediaResource*, used to specify how a file is

	divided in *Tracks*
subclass	*Essence* is a sub-Class of *MediaResource*, used to specify a *MediaResource* ready for publication.
Class Properties	
mediaResourceId	Unique Identifier e.g. a UUID, UMID etc. It can be generated or assigned by the business process or it can be extracted from the content.
mediaResourceName	The name of the *MediaResource*.
mediaResourceDescription	A description of a *MediaResource*.
mediaResourceType	The type of *MediaResource*.
Etc.	Many other properties can be associated with a *MediaResource*. See EBU Tech 3293, EBUCore.

2.2.4.3 Track

Definition:

A *Track* is both a part and a subclass of a *MediaResource*. A *MediaResource* is potentially composed of any combination of audio, video and data *Tracks*.

Examples:

Examples of video *Tracks* are different camera angles or an additional signing *Track*.

Examples of audio *Tracks* are stereo pairs, multichannel audio e.g. surround, international sound, etc.

Examples of data *Tracks*: ancillary data, captioning, etc.

Class relations	
Etc.	Other class relationships can be associated to a *Track*. See EBU Tech 3293, EBUCore.
Class properties	
trackId	The identifier attributed to a *Track*.
trackType	The type of *Track*.
trackName	A name associated to a *Track*.
trackPurpose	A short description of what the *Track* is used for.
Etc.	Many other properties can be associated with a *Track*. See EBU Tech 3293, EBUCore.

2.2.4.4 Format

Definition:

Format is a structure of technical metadata. A *Format* can be defined as the composition of audio, video and or data components and the description of their respective *Formats*. The *ContainerFormat* defines the file / package structure of the *MediaResource*. A streaming format can also be defined as a specific ContainerFormat for streaming or a custom combination of an AudioFormat and VideoFormat…

Example:

A *Format* for an audio *MediaResource* will define the audio encoding format, the sampling frequency, etc.

Often used subclasses	
subclass	*AudioFormat* is a sub-class of *Format*, used to list all the characteristics of the audio signal. See e.g. 'audioFormat' in EBU Tech 3293, EBUCore for more information.
subclass	*VideoFormat* is a sub-class of *Format*, used to list all the characteristics of the video signal. See e.g. 'videoFormat' in EBU Tech 3293, EBUCore for more information.
subclass	*DataFormat* is a sub-class of *Format*, used to list all the characteristics of the data signal. See e.g. 'dataFormat' in EBU Tech 3293, EBUCore for more information.
subclass	*ContainerFormat* is a sub-class of *Format*, used to list all the characteristics of the container. It provides information on the container / wrapper format in complement to the stream encoding information provided in 'channel', (e.g. mp3, wave, Quicktime, ogg). See, e.g., 'containerFormat' in EBU Tech 3293, EBUCore for more information.
subclass	*StreamFormat* is a sub-class of *Format*, used to list all the characteristics of a stream.
Class Properties	
formatId	An identifier associated to the *Format*.
formatName	A name associated to the *Format*.
Etc.	Many other properties can be associated with a *Format*. See EBU Tech 3293, EBUCore.

2.2.4.4.1 AudioFormat

Definition:

A class to provide definitions about the "*AudioFormat*" (e.g. encoding format, sampling rate).

Class relations	
Etc.	Other class relationships can be associated with an *AudioFormat*. See EBU Tech 3293, EBUCore. This standard defines the Audio Definition Model
Class Properties	
Etc.	Other data properties can be associated with an *AudioFormat*. See EBU Tech 3293, EBUCore. This standard defines the schema of the Audio Definition Model (ADM).

2.2.4.4.2 VideoFormat

Definition:

A class to provide definitions about the "*VideoFormat*" (e.g. encoding format, frame rate).

Class relations	
Etc.	Other class relationships can be associated with a *VideoFormat*. See EBU Tech 3293, EBUCore.
Class Properties	
Etc.	Other data properties can be associated with a *VideoFormat*. See EBU Tech 3293, EBUCore.

2.2.4.4.3 *DataFormat*

Definition:

A class to provide definitions about the "*DataFormat*" (e.g. captioning format).

Class relations	
Etc.	Other class relationships can be associated with a *DataFormat*. See EBU Tech 3293, EBUCore.
Class Properties	
Etc.	Other data properties can be associated with a *DataFormat*. See EBU Tech 3293, EBUCore.

2.2.4.4.4 *ContainerFormat*

Definition:

A class to provide definitions about the "*ContainerFormat*" (e.g. container type).

Class relations	
Etc.	Other class relationships can be associated with a *ContainerFormat*. See EBU Tech 3293, EBUCore.
Class Properties	
Etc.	Other data properties can be associated with a *ContainerFormat*. See EBU Tech 3293, EBUCore.

2.2.4.5 Essence

Definition:

The *Essence* is a physical representation of a *MediaResource* in a particular *Format* destined for play-out or publishing. *Essence* is a subclass of a *MediaResource* and inherits the *MediaResource* properties. An *Essence* can be available in a form of a simple file or complex packages (e.g. as delivered by cameras of different brands).

Examples:

An AAC file is an example of audio *Essence*. A P2 file structure (audio, video clip, voice, icon, proxy directories) is an example of package.

Class relations	
isComposedOf	A list of *MediaResources* that composes the *Essence*.
Etc.	Other class relationships can be associated with an *Essence*. See EBU Tech 3293, EBUCore.
Class Properties	
readyForPublication	A flag that is set if the *Essence* is ready for publication.
Etc.	Many other properties can be associated with an *Essence*. See EBU Tech 3293, EBUCore.

2.2.4.6 PhysicalResource

Definition:

A physical manifestation of the *EditorialObject* it instantiates.

Examples:

This can be a paper document, a book or any other physical object that manifest someone's idea.

Class relations	
Etc.	Other class relationships can be associated with a *Resource*. See EBU Tech 3293, EBUCore.
Class hierarchy	
superclass	*Resource* is the superclass for *PhysicalResource*
Class Properties	
Etc.	Many other properties can be associated to a *Resource*. See EBU Tech 3293, EBUCore.

2.2.4.7 Artefact

Definition:

An object made by a human or after an idea of a human that are in use, e.g. in a production.

Class relations	
hasArtefactRelatedPhysicalResource	Relation to a *PhysicalResource* associated with the *Artefact*
hasArtefactRelatedResource	Relation to a *Resource* associated with the *Artefact*
Etc.	Other class relationships can be associated with a *Resource*. See EBU Tech 3293, EBUCore.
Class Properties	
artefactId	Unique Identifier e.g. a UUID, UMID, URI etc. It can be generated or assigned by the business process or it can be extracted from the content.
artefactName	The name given to an *Artefact*.
artefactDescription	A description of an *Artefact*.
artefactType	The type of *Artefact*.
Etc.	Many other properties can be associated to a *Resource*. See EBU Tech 3293, EBUCore.

2.2.4.8 ProductionJob

Definition:

The “*ProductionJob*” is a process to produce an *Essence* for publication. It uses *MediaResources* as inputs, based on an *EditorialObject* describing the process in detail. It is ordered by a *Contract*.

Where a production is described in several steps, the output can be a *MediaResource* that is not ready for publishing but will be used as input of other *ProductionJobs*.

Class relations	
basedOn	Relation to the *EditorialObject* that is produced by the job
hasSubJob	Relation to a breakdown of the *ProductionJob*, i.e. a separate task of a workflow.
hasInputMediaResource	A list of *MediaResources* that are used for composing the *Essence*.
hasInputResource	A list of *Resources* that are used for composing the *Essence*.
hasOutputMediaResouce	Relation to a *MediaResource* that is the result of the job.
hasOutputResource	Relation to a *Resource* that is the result of the job.
hasOutputEssence	Relation to the *Essence* that is the result of the job.
hasPJContributor	Information about crew, etc.
isOrderedBy	Relation to the *Contract* through which the *ProductionJob* is ordered.
hasProductionJobLocation	Relation to the location of the *ProductionJob*. This can be a studio or another recording location
hasProductionJobEvent	Relation to the time information associated with the *ProductionJob*. Can be used for model production plans.
usesProductionDevice	To identify ProductionDevices used for the ProductionJob.
Etc.	Other class relationships can be associated with a *ProductionJob*.
Class Properties	
productionJobId	Identifier for the *ProductionJob*
productionJobName	The name of a *ProductionJob*.
productionJobdescription	The description of a *ProductionJob*.
productionJobType	The type of *ProductionJob*.
Etc.	Many other properties can be associated with a *ProductionJob*.

2.2.4.9 ProductionDevice

Definition:

A "*ProductionDevice*" is a device used during a *ProductionJob*.

Example:

An example of a *ProductionDevice* is a tapeless camcorder.

Class relations	
hasUsageContract	Relation to a *Contract* regulating the usage of the *ProductionDevice*.
Etc.	Other class relationships can be associated to a *ProductionDevice*.
Class Properties	
productionDeviceId	An identifier associated to a *ProductionDevice*.
productionDeviceType	The type of the *ProductionDevice* e.g. a camcorder.
productionDeviceName	The name of the *ProductionDevice*.
productionDeviceDescription	A description of the *ProductionDevice*.
Etc.	Many other class properties can be associated with a *ProductionDevice*. Examples of additional properties for a camcorder can be found in EBU Tech 3349 (Acquisition Metadata).

2.2.5 Distribution Domain

The Distribution Domain covers any form of publishing, play-out or distribution.

The central class is the *PublicationEvent* that plays out an *Essence*, i.e. the media object that was the result of the *ProductionJob*.

Other classes can be added to suit a specific need in play-out or distribution.

A *PublicationEvent* can be, for example:

- A broadcast event, i.e. an isolated event such as for last minutes news reports, etc. This content can be available via over the air broadcast or streaming.
- A scheduled event, i.e. each event being identified in a particular timeslot. This content can be available via over the air broadcast or streaming.
- An on-demand event, i.e. content is made available for immediate viewing or for download. It generally has a certain window of time availability. Catch-up TV is considered as an on-demand event. On-demand events can also be linked to broadcast and schedule events.
- An on-line event, i.e. content is made available for download/fruition on some web repository (e.g. on a web site)

According to the type of *PublicationEvent*, *MediaResource* is available in different Formats instantiated in *Essence* files or packages.

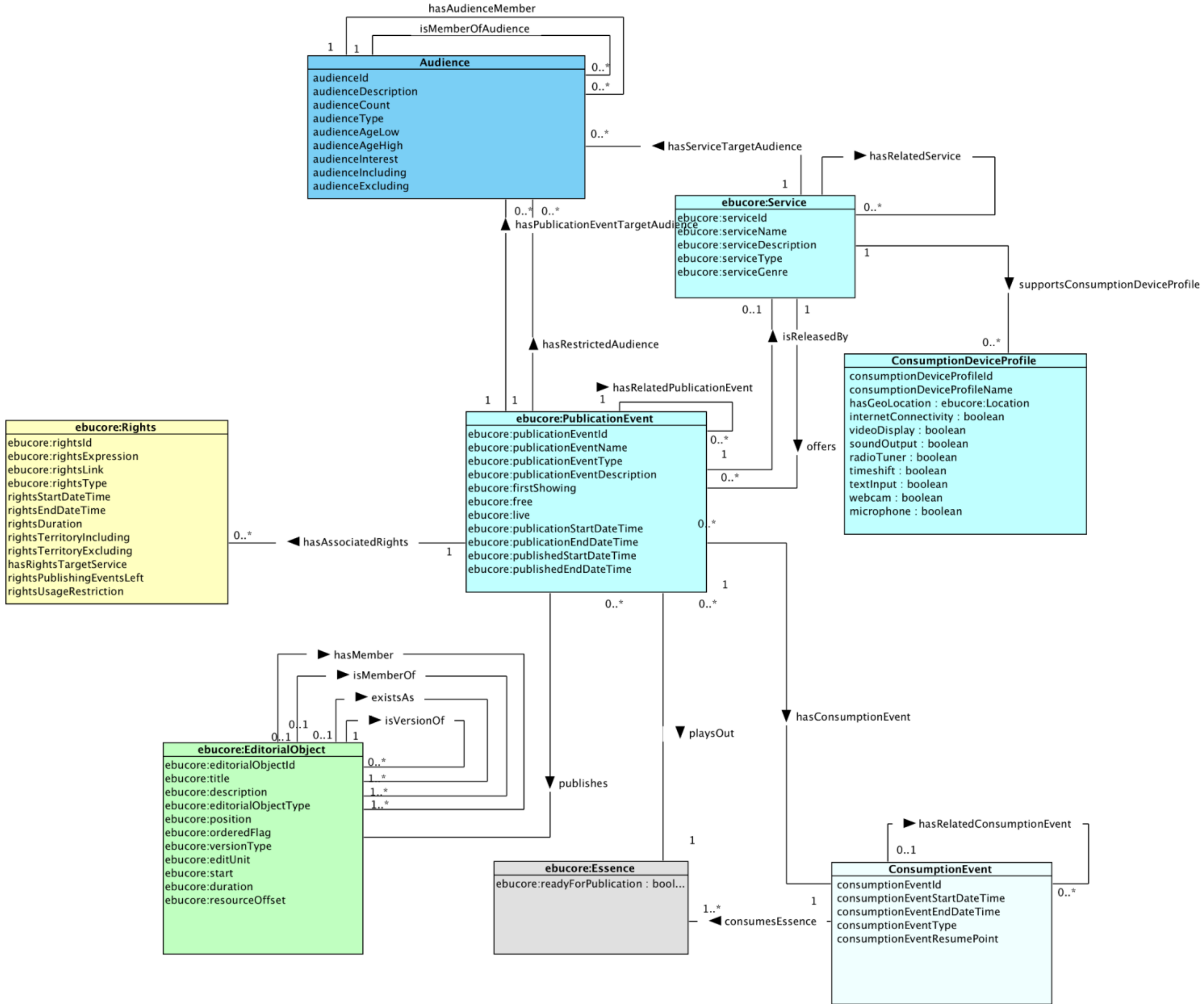


Figure 7: Publication Event

2.2.5.1 PublicationEvent

Definition:

The publication of an *EditorialObject* for user consumption is realised by releasing an *Essence*.

Example:

A *PublicationEvent* that is, for example, a scheduled event i.e. a time slot in a schedule associated with a *PublicationChannel*. A *PublicationEvent* can also be a broadcast event not in a preliminary schedule, such as a live special news report. A *PublicationEvent* can also be a streaming event or a VoD publication event.

Class relations	
publishes	A relation to an *EditorialObject* representing the story that will be published.
playsOut	To allow the ordered publication of a time related sequence of *MediaResource* / *Essence* as a *TimelineTrack* of an *EditorialObject*.
hasAssociatedRights	To identify the Rights directly associated with a *PublicationEvent* in addition to inferred rights associated with the related *EditorialObjects*, *MediaResources* and/or *Essences*.
hasRelatedPublicationEvent	To establish a link between two *PublicationEvents* (e.g. linking an on-demand event triggered from a broadcast event.
hasPublicationEventTargetAudience	The publication targets this particular audience represented by the *Audience* class.
hasRestrictedAudience	The content is forbidden for this audience.
isReleasedBy	The channel or service platform that releases the content
hasConsumptionEvent	Relation to *ConsumptionEvents* in relation to a *PublicationEvent*.
Etc.	Other class relationships can be associated to a *PublicationEvent*. See e.g. ETSI TS 102 822 (TV-Anytime) or the BBC Programme Ontology.
Class Properties	
publicationEventId	An identifier associated with the *PublicationEvent*.
publicationEventName	The name of the *PublicationEvent*.
publicationEventDescription	A description of the *PublicationEvent*.
publicationStartDateTime	The date and time at which the programme is scheduled to start or when content is made available / can be accessed or consumed.
publishedStartDateTime	The scheduled start date and time of publication.
publicationEndDateTime	The date and time at which the programme is scheduled to end or after which content is no longer available / accessible / consumable.
publishedEndDateTime	The scheduled end date and time of publication.
publicationEventType	The type of the PublicationEvent, e.g. publishing on web or play-out on radio
live	If set, a flag to indicate that the content should be marked as “Live”.
free	If set, a flag to indicate that content can be accessed / consumed without subscription.
firstShowing	If set, a flag to indicate that this is the first time that this content is available on this *PublicationChannel*. This is just an indication, the collection of the *PublicationEvents* one *Essence* have will tell the real publishing history.
Etc.	Many other properties can be used to define a *PublicationEvent*. See e.g. ETSI TS 102 822 (TV-Anytime) or the BBC Programme Ontology.

2.2.5.2 Service

<u>*Definition:*</u>

A *Service* is a channel or publishing platform that releases the content to a given audience.

Class relations	
hasRelatedService	Relation to some related publishing *Service*.
Offers	A list of PublicationEvents the *Service* offers, i.e. like an EPG
hasServiceTargetAudience	The *Audience* the *Service* has been designed for.
supportsConsumptionDeviceProfile	A list of devices the *Service* supports, described using instances of the ConsumptionDeviceProfile class.
Etc.	Other Class relationships can be associated to a *Service*. See e.g. ETSI TS 102 822 (TV-Anytime)
Sub-Classes	
PublicationChannel	A specific type of *Service*.
Class Properties	
serviceId	An identifier attributed to the *Service*.
serviceName	The name given to the *Service*.
serviceDescription	A description of the *Service*.
serviceType	Description of the type of *Service*.
serviceGenre	The genre of *Service*.
Etc.	Many other properties can be used to define a *Service*.

2.2.5.3 ConsumptionDeviceProfile

Describes technical capabilities and requirements of a ConsumptionDevice that are needed for accessing a Service.

Class relations	
hasGeoLocation	The device is currently within the boundary of a (geo) location. This can assist finding the closest and best CDN service for the device. It might also be used to restrict geo-location access to content.
Etc.	Other class relationships can be associated to a *ConsumptionDeviceProfile*.
Class Properties	
consumptionDeviceProfileId	An identifier associated with the *ConsumptionDeviceProfile*.
consumptionDeviceProfileName	A name given to the profile.
internetConnectivity	The device is capable of accessing the Internet.
videoDisplay	The device is capable of displaying video picture frames.
soundOutput	The device is capable of outputting sound.
radioTuner	The device has a radio tuner.
timeshift	The device has a time shift capacity.
textInput	The device has a keyboard or another means of text input.
webcam	The unit can record video.
microphone	The device can record audio.

Etc.	Many other properties can be used to define a *ConsumptionDeviceProfile*.

2.2.6 Consumption Domain

In the same way, the Consumption Domain covers aspects of the access and consumption of *Essence*, including any response or *Resonance* this may trigger by the consumer.

The central class in the Consumption Domain is the *ConsumptionEvent*. For linear publishing, this will happen at the same time as the *PublicationEvent*, but for on-line publishing this event will occur one or more times, during the lifecycle of the *PublicationEvent*.

To help adapting the content to the right device and *Consumer*, this domain has a class to describe the ConsumptionDevice in detail, but also the *Consumer* via his *Account* information.

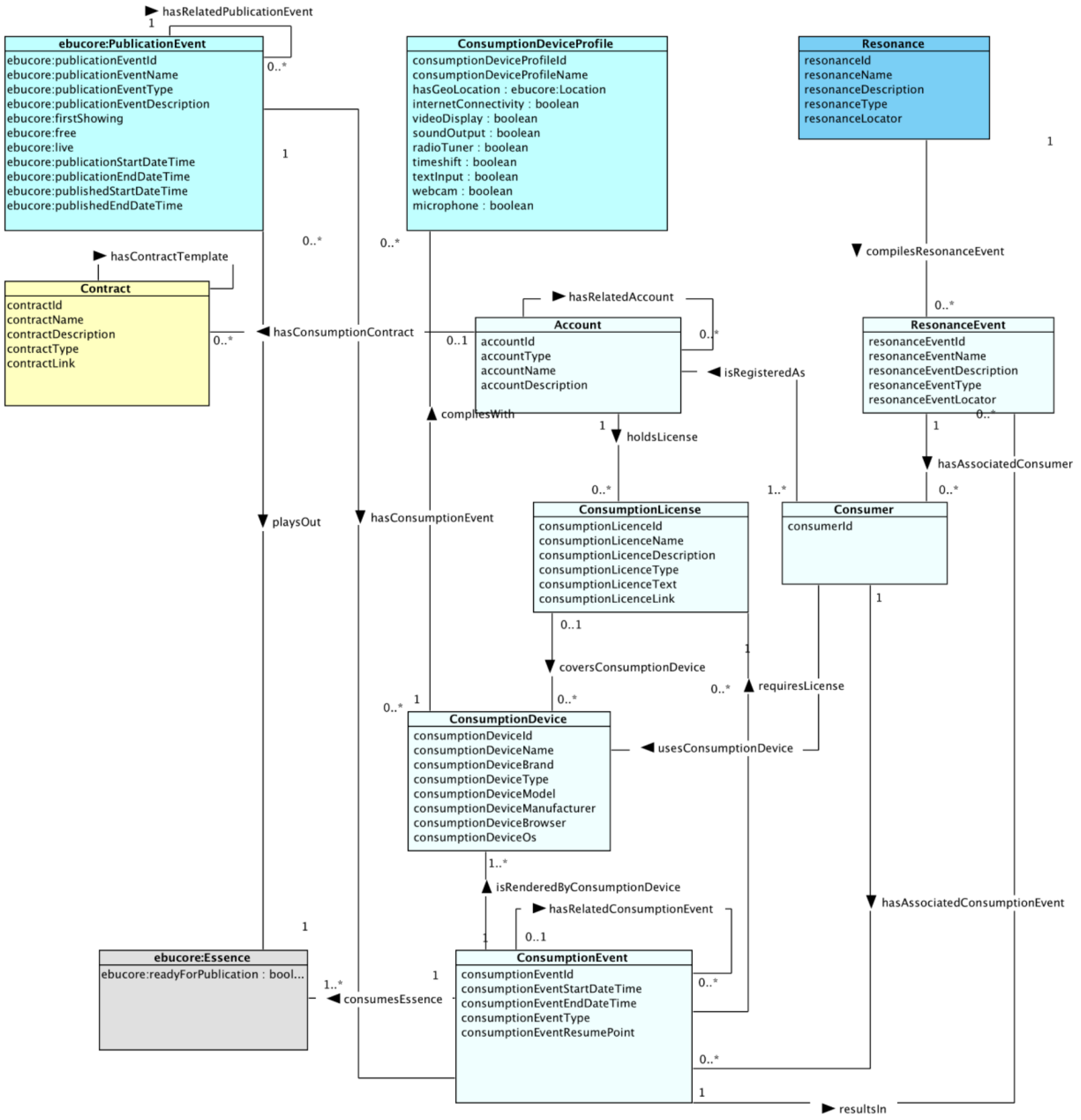


Figure 8: Consumption Domain

2.2.6.1 ConsumptionEvent

Definition:

Represents the event of a user consuming a published content.

A *ConsumptionEvent* follows publication *but is no longer related* to the *PublicationEvent*. The link to the *PublicationEvent* is represented via the *Essence* it consumes.

For linear services the *ConsumptionEvent* and *PublicationEvent* happen at the same time (well, almost, when respecting signal transport and transformation time). For non-linear services, the *Consumer* decides about the time of the *ConsumptionEvent*.

The *ConsumptionEvent* can be followed by a *ResonanceEvent*, if the consumer reacts in a countable or noticeable way.

Example:

- reading a news article on a public service broadcaster's web site

- watching a TV program

- listening to a radio program

Class relations	
isRenderedByConsumptionDevice	Relation to the device used as a media render at the moment of consumption
resultsIn	When the user consumes an *Essence*, different kinds of ResonanceEvents may be generated.
consumesEssence	A relation to the *Essence* the *ConsumptionEvent* consumes at least a part of.
requiresLicence	A relation to a licence needed for accessing the content
hasRelatedConsumptionEvent	Used for modelling usage pattern, like first A was consumed, then B and C.
Etc.	Other Class relationships can be associated with a *ConsumptionEvent*. See e.g. ETSI TS 102 822 (TV-Anytime)
Class Properties	
consumptionEventId	An identifier attributed to the *ConsumptionEvent*.
consumptionStartDateTime	The start date and time of the event
consumptionEndDateTime	The end date and time of the event
consumptionEventType	The type of *ConsumptionEvent*
consumptionEventResumePoint	Reflects the resume timing data for a later *ConsumptionEvent* session on the same *Essence*.
Etc.	Many other properties can be used to define a *ConsumptionEvent*. See e.g. ETSI TS 102 822 (TV-Anytime)

2.2.6.2 ConsumptionDevice

Definition:

Represents a technical system to access and consume a media service. Its characteristics (seen from a service point of view) are identified into a *ConsumptionDeviceProfile*.

Example:

Examples of *ConsumptionDevices* would be e.g. a mobile phone (including all hardware and software needed for access and consumption), an OTT box together with its TV screen, a TV set with integrated cable tuner, a DAB+ radio.

Class relations	
compliesWith	A list of *ConsumptionDeviceProfiles* the *ConsumptionDevice* complies with.
Etc.	Other Class relationships can be associated with a *ConsumptionDevice*.
Class Properties	
consumptionDeviceId	An identifier associated with the *ConsumptionDevice*.
consumptionDeviceType	The type of device in use.
consumptionDeviceName	The name the device is known under.
consumptionDeviceBrand	The brand name of the device.
consumptionDeviceManufacturer	The name of the manufacturer of the device.
consumptionDeviceModel	The model of the device.
consumptionDeviceBrowser	The kind of browser used on the device.
consumptionDeviceOs	Type of the operating system running on the device.
Etc.	Many other properties can be used to define a *ConsumptionDevice*.

2.2.6.3 ConsumptionLicence

Definition:

Represents the proof held by a *Consumer* on having the right to experience a *ConsumptionEvent* and consume the published *Essence*.

The *ConsumptionLicence* is verified by a mechanism that is usually located in the *ConsumptionDevice* and referred to as DRM.

Example:

- a document stating the payment of a TV licence fee (this cannot be checked by a DRM mechanism)
- a smart card from a pay TV service containing the necessary information to decode their coded signal

Class relations	
coversConsumptionDevice	The *ConsumptionLicence* will unlock content for this device
Etc.	Other Class relationships can be associated to a *ConsumptionLicence*.
Class Properties	
consumptionLicenceId	An identifier associated with the *ConsumptionLicence*.
consumptionLicenceText	A *ConsumptionLicence* string that can be verified by the device, i.e. DRM
consumptionLicenceName	A name attributed to a *ConsumptionLicence*.
consumptionLicenceDescription	A description of the *ConsumptionLicence*.
consumptionLicenceType	The type of *ConsumptionLicence*.
consumptionLicenceLink	An URL where the *ConsumptionLicence* is stored
Etc.	Many other properties can be used to define a

	ConsumptionLicence.

2.2.6.4 Consumer

Definition:

Represents the individual who consumes the *Service* by using a *ConsumptionDevice*.

The *Consumer* is a member of the *Audience*. He consumes the *ConsumptionEvent* and initiates *ResonanceEvents*. He holds an *Account* and a *ConsumptionLicence*.

Example:

- Every member of a family watching a TV program, possibly over only one *Account* of the service provider

Class relations	
belongsToAudience	Relation to a list of *Audiences* the *Consumer* belongs to.
hasAssociatedConsumptionEvent	A list of *ConsumptionEvents* that the user has consumed.
isRegisteredAs	Relation to the *Account* the user is registered as.
usesConsumptionDevice	Relation to the *ConsumptionDevice* that is used.
Etc.	Other Class relationships can be associated to a *Consumer*. See e.g. ETSI TS 102 822 (TV-Anytime)
Class Properties	
consumerId	An identifier attributed to a *Consumer*.
Etc.	Many other properties can be used to define a *Consumer*.

2.2.6.5 Account

Definition:

Represents *Account* information like login, billing address, banking account, e-mail address, etc.

Example:

- a social web account of the news department of a public service media

- a person's TV licence fee related account and address

- a simple Id representing an anonymous usage pattern.

Implementers note:

The attribute set can vary and must be added for each of the applications.

Class relations	
holdsLicence	List of *ConsumptionLicences* the *Account* holds for their users
hasRelatedAccount	A reference to a related *Account*, e.g. a family *Account*
hasConsumptionContract	A relation to the contract specifying the terms for consumption
Etc.	Other class relationships can be associated to an *Account*.
Class Properties	
accountId	An identifier attributed to an *Account*.
Etc.	Many other properties can be used to define an *Account*.

2.2.6.6 ResonanceEvent

Definition:

Represents all individual events that are countable or noticeable reactions by consumers on the *ConsumptionEvent*. E.g. clicks, likes, comments, votes, tweets, preferences, downloads...

All *ResonanceEvents* are linked via the *ConsumptionEvent* to format-related information of an *Essence* and to content-related information of an *EditorialObject*.

ResonanceEvents represent raw-data that needs to be aggregated (e.g. summed up). Raw-data can be a case of "Big Data" and require appropriate technology.

Analysis of the *ResonanceEvents* leads to demand (modelled as *Campaign*), which defines the framework of the *PublicationPlan*.

Example:

- Every click on the like button of a web site

Class relations	
hasAssociatedConsumer	The user that is connected to the *ResonanceEvent*.
Etc.	Other Class relationships can be associated to a *ResonanceEvent*
Class Properties	
resonanceEventId	An identifier associated with the *ResonanceEvent*.
resonanceEventName	The name given to a *ResonanceEvent*.
resonanceEventDescription	A description of a *ResonanceEvent*.
resonanceEventType	A type of *ResonanceEvent*.
resonanceEventLocator	A locator pointing to the content of the *ResonanceEvent* information.
Etc.	Many other properties can be used to define a *ResonanceEvent*.

2.2.7 Planning Domain

This is where the classes used for describing the demand. The demand, based on the Resonance from different audience groups, is met with a *Campaign*, describing the strategy and uses a *PublicationPlan* and *ProductionOrders* to commission productions and the publishing of the produced *Essences*.

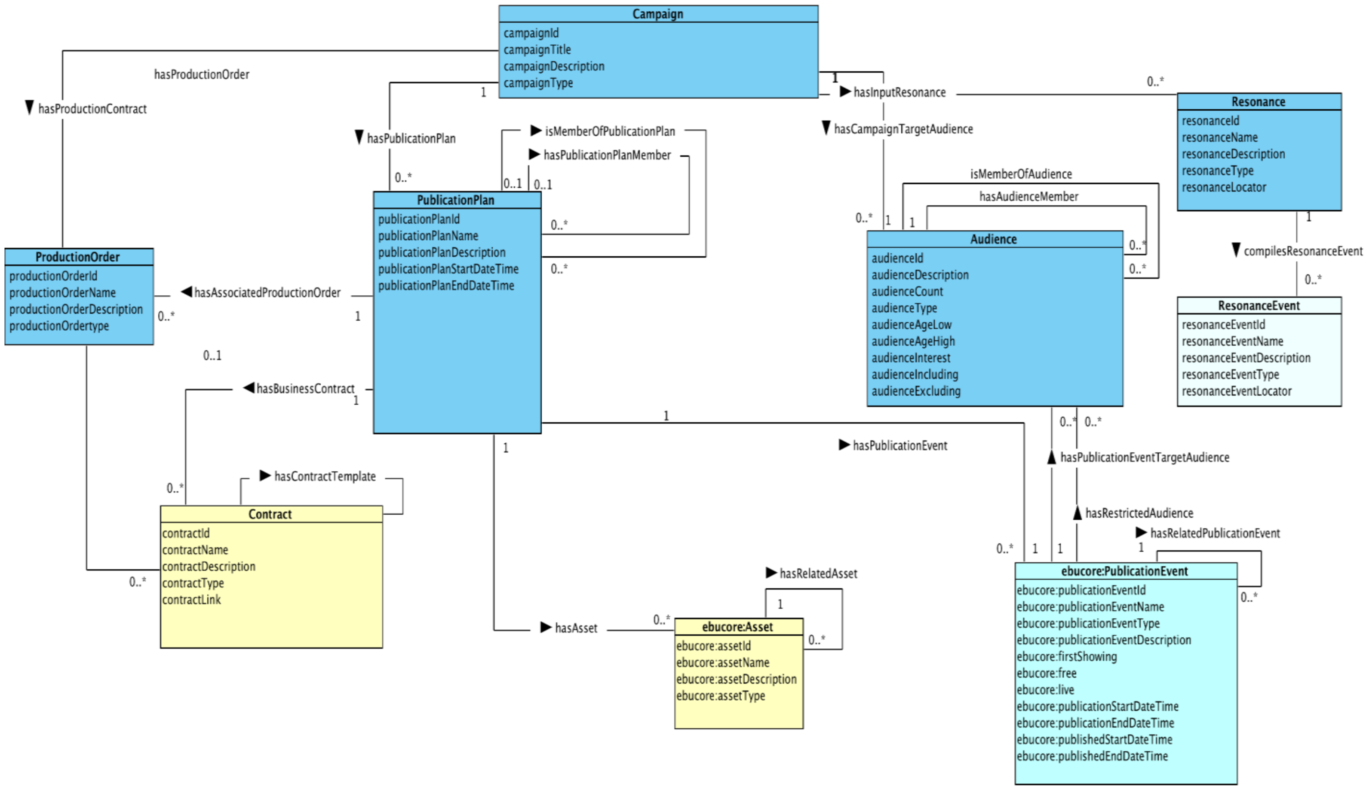


Figure 9: Planning Domain

2.2.7.1 Campaign

Definition:

Represents objects that describe the framework of the *PublicationPlan*. A *Campaign* is an initial plan to release content and also the result of the analysis of the *Resonance* data (e.g. likes, downloads, etc). A *Campaign* has a target *Audience* and will usually be associated to a *PublicationPlan*.

Examples could be the desired quantity of *PublicationEvents* (repetition, duration) for a specific *TargetAudience* and of a specific genre (e.g. sport, news, documentation, commercials) and/or of a specific type, etc. The *PublicationPlan* is supposed to meet this demand and can be checked against it.

Campaign is used for advertising and promotional campaigns as well as e.g. overall publication strategies in a public broadcaster.

Class relations	
hasPublicationPlan	A list of *PublicationPlans* that will help expressing the purpose of the *Campaign*.
hasInputResonance	A list of *Resonance* objects that are used as a base for the *Campaign*.
hasCampaignAudience	The *Audience* the *Campaign* targets.
Etc.	Other Class relationships can be associated with a *Campaign*.
Class Properties	
campaignId	An identifier attributed to a *Campaign*.
campaignTitle	The title of the *Campaign*.
campaignDescription	A short description of the *Campaign*.

campaignType	The type of *Campaign*.
Etc.	Many other properties can be used to define a *Campaign*.

2.2.7.2 PublicationPlan

Definition:

The *PublicationPlan* class describes a schedule of *PublicationEvents* (and their respective *Audiences*) with references to resulting *ProductionOrders*, and *Assets* (and their *EditorialObjects*). *PublicationPlans* can be related to each other hierarchically, strictly, i.e. membership can only be with one group.

Examples:

A *Campaign* of commercials for a product, is realised with a *PublicationPlan* defining a set of planned *PublicationEvents* using the associated *Assets*.

A fiction film is promoted with several publications of trailers to a targeted *Audience* and before the publication of the film.

Class relations	
isMemberOfPublicationPlan	A list of *PublicationPlans* the *PublicationPlan* is a part of.
hasPublicationPlanMember	A list of *PublicationPlans* that the *PublicationPlan* contains, which can be used to divide the plan into smaller units.
hasAssociatedProductionOrder	A list of *ProductionOrders* that orders the production of content aimed to be published by the *PublicationEvents* related to the *PublicationPlan*.
hasBusinessContract	A list of *Contracts* that are related to *PublicationPlan*.
hasStakeholder	A list of stakeholders that are important to the *PublicationPlan*.
hasPublicationEvent	A list of *PublicationEvents* that is a part of the *PublishingPlan*.
hasAsset	The assets the *PublicationPlan* covers.
Etc.	Other class relationships can be associated with a *PublicationPlan*.
Class Properties	
publicationPlanId	An identifier associated with the *PublicationPlan*.
publicationPlanName	A name attributed to the *PublicationPlan*.
publicationPlanDescription	A description of the *PublicationPlan*.
PublicationPlanStartDateTime	the start and time date of the *PublicationPlan*.
PublicationPlanEndDateTime	The end and time date of the *PublicationPlan*.
Etc.	Many other properties can be used to define a *PublicationPlan*.

2.2.7.3 ProductionOrder

Definition:

The class *ProductionOrder* represents an order for production.

Describes the instance of placing an order with attributes like date, client, contractor, reference to the contract, etc.

Class relations

hasProductionContract	Relation to a *Contract* concerning the *ProductionOrder*.
Etc.	Other class relationships can be associated with a *ProductionOrder*.
Class Properties	
productionOrderId	An identifier associated with the *ProductionOrder*.
productionOrderName	The name of the *ProductionOrder*.
productionOrderDescription	A description of the *ProductionOrder*.
productionOrderType	The type of *ProductionOrder*.
Etc.	Many other properties can be used to define a *ProductionOrder*.

2.2.7.4 Audience

Definition:

Represents a group of consuming customers/users by number, age, type, interests, etc.

Audiences can be related to each other hierarchically. Hierarchy is not strict, i.e. membership can exist with an arbitrary number of groups.

With the hasAudienceMember relation, different *Audience* groups can be linked together to model a more complex *Audience* group. The audienceIncluding, audienceExcluding indicates that the subgroup should be added or excluded from the group that is modelled.

Class relations	
hasAudienceMember	A list of specific Audiences that are used to model a complex *Audience*.
isMemberOfAudience	A list of *Audiences* this particular *Audience* is a part of.
Etc.	Other class relationships can be associated with an Audience.
Class Properties	
audienceId	An identifier attributed to an *Audience*.
audienceDescription	A description of the *Audience* group covered
audienceCount	The real counted size of the *Audience*.
audienceType	Type of *Audience*.
audienceAgeLow	The lowest age of a member of the *Audience*.
audienceAgeHigh	The highest age of a member of the *Audience*.
audienceInterest	A particular interest common to an *Audience* group.
audienceIncluding	This *Audience* group part should be included in a composed group.
audienceExcluding	This *Audience* group part should be excluded in a composed group.
Etc.	Many other properties can be used to define an *Audience*.

2.2.7.5 Resonance

Definition:

Represents the aggregated form (i.e. a non-individual expression) of all countable or noticeable reactions by *Consumers* on the *ConsumptionEvent*.

Examples:

Click rates, number of likes, percentage of votes, number of downloads…

Class relations	
isMeasuredBy	The *Agent* responsible for compiling and analyzing the data into the *Resonance*.
compilesResonanceEvents	One of the *ResonanceEvents* used as a basis for defining the *Resonance*.
Etc.	Other Class relationships can be associated to a *Resonance*.
Class Properties	
resonanceId	An identifier attributed to a *Resonance*.
resonanceName	The name of a *Resonance*.
resonanceDescription	A description of a *Resonance*.
resonanceType	A type of *Resonance*.
resonanceLocator	A locator to the document describing the *Resonance*.
Etc.	Many other properties can be used to define a *Resonance*.

2.2.8 Financial Domain

The Financial Domain is the domain, where cost and value of productions are modelled in a very simple fashion. The two classes in the domain can also be used for connecting the CCDM model to a model used for more accurately modelling financial structures, by connecting those two classes to similar classes in the external model.

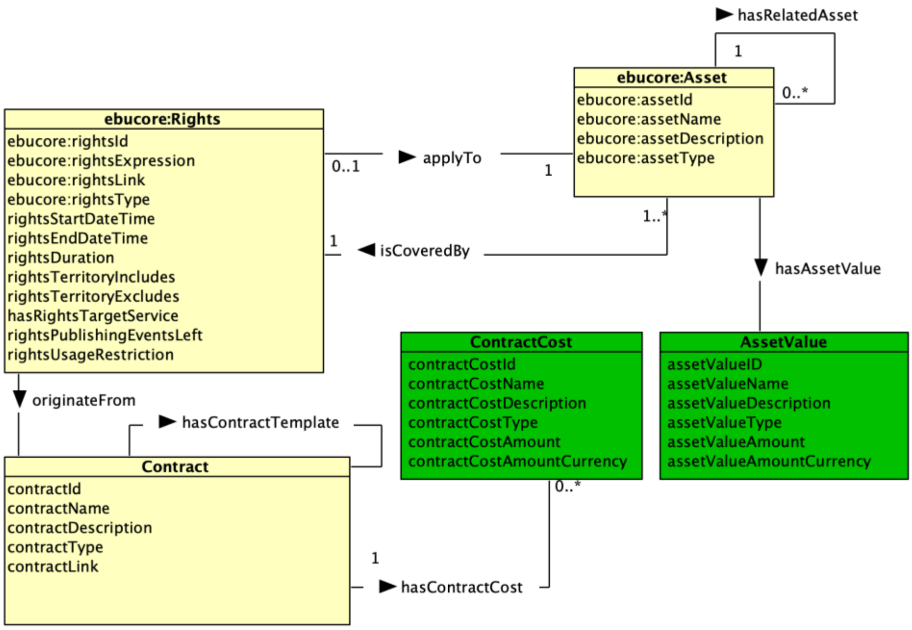


Figure 10: Financial Domain

2.2.8.1 AssetValue

Definition:

Represents the value of an Asset. The value can be figurative or abstract.

Class relations	
Etc.	Other Class relationships can be associated to a *an AssetValue*.
Class Properties	

assetValueId	An identifier attributed to a Value.
assetValueName	The name of a Value.
assetValueDescription	A description of a Value.
assetValueType	A type of Value.
assetValue	The estimated or actual value of an Asset.
assetValueCurrency	The currency in which the assetValue is expressed.
Etc.	Many other properties can be used to define a *Value*.

2.2.8.2 ContractCost

Definition:

Represents the cost of a contractual commitment of any kind.

Class relations	
Etc.	Other Class relationships can be associated to a *Resonance*.
Class Properties	
contractCostId	An identifier attributed to a Contract*Cost*.
contractCostName	The name of a Contract*Cost*.
contractCostDescription	A description of a Contract*Cost*.
contractCostType	A type of Contract*Cost*.
contractCostAmount	The actual cost value.
contractCostValueCurrency	The currency in which the cost is expressed.
Etc.	Many other properties can be used to define a Contract*Cost*.

3. Implementation Guidelines / Questions & Answers

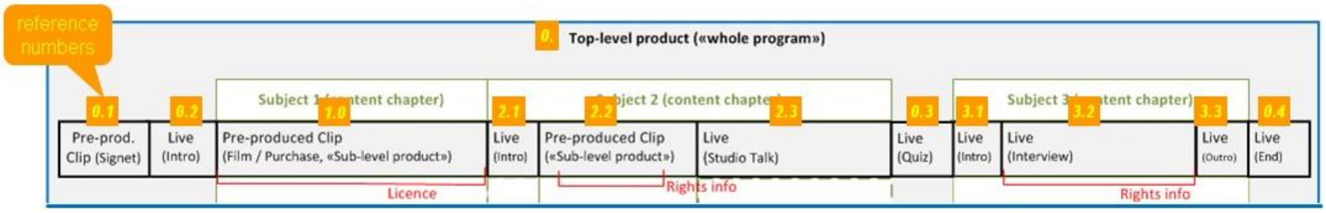
3.1 General remarks

This section provides examples from current implementers of the EBU CCDM and is intended to provide advice and clarification for users to help them in implementing the EBU CCDM in future versions of the specification.

3.2 Examples provided by SRG SSR, Swiss Confederation

3.2.1 Modelling Different Viewpoints with CCDM

An example of a programme, called “ideal programme”, is shown below:

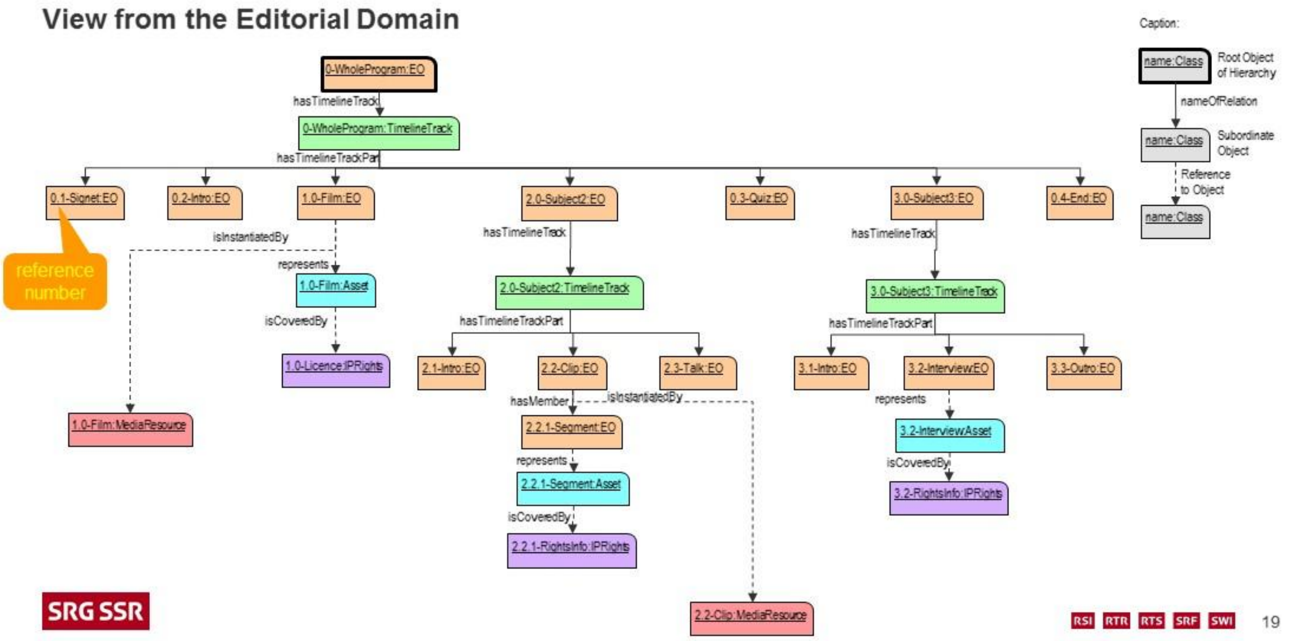

reference numbers
0. Top-level product («whole program»)
Subject 1 (content chapter)
Subject 2 (content chapter)
Subject 3 (content chapter)
0.1
Pre-prod. Clip (Signet)
0.2
Live (Intro)
1.0
Pre-produced Clip (Film / Purchase, «Sub-level product»)
2.1
Live (Intro)
2.2
Pre-produced Clip («Sub-level product»)
2.3
Live (Studio Talk)
0.3
Live (Quiz)
3.1
Live (Intro)
3.2
Live (Interview)
3.3
Live (Outro)
0.4
Live (End)
Licence
Rights info
Rights info

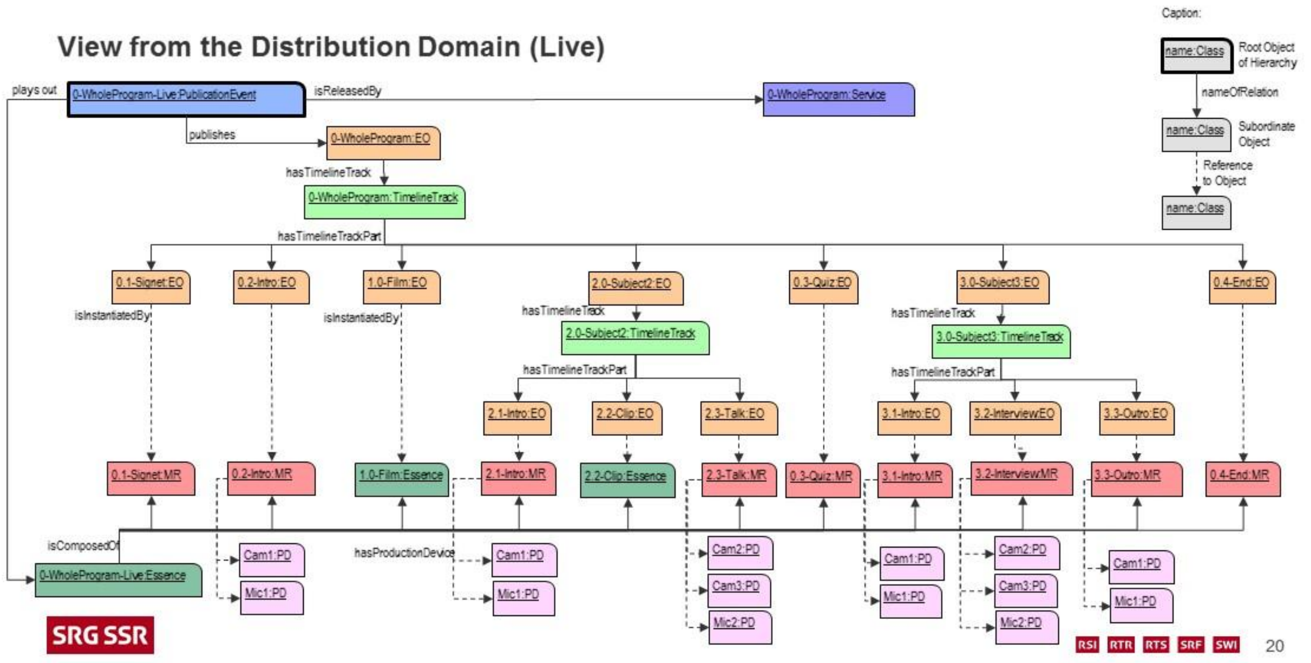
This example will now be represented using CCDM. The representation depends on the viewpoint, which maps nicely to the domains described in this document. Also, the following examples assume different Publication scenarios, such as “Live” or “Repetition”. Some examples contain objects that are not directly represented in the graph of the “ideal programme”, for example, the *ProductionDevices* Cam1 and Mic1.

All of these assumptions were made only to show the possibilities of modelling with CCDM.

The object graphs represent a hierarchical structure, such as that found in an XML document. To emphasise the hierarchy, it is necessary to introduce “references” (represented as dashed arrows) besides the pure object relation (represented as full arrows) in the hierarchy.

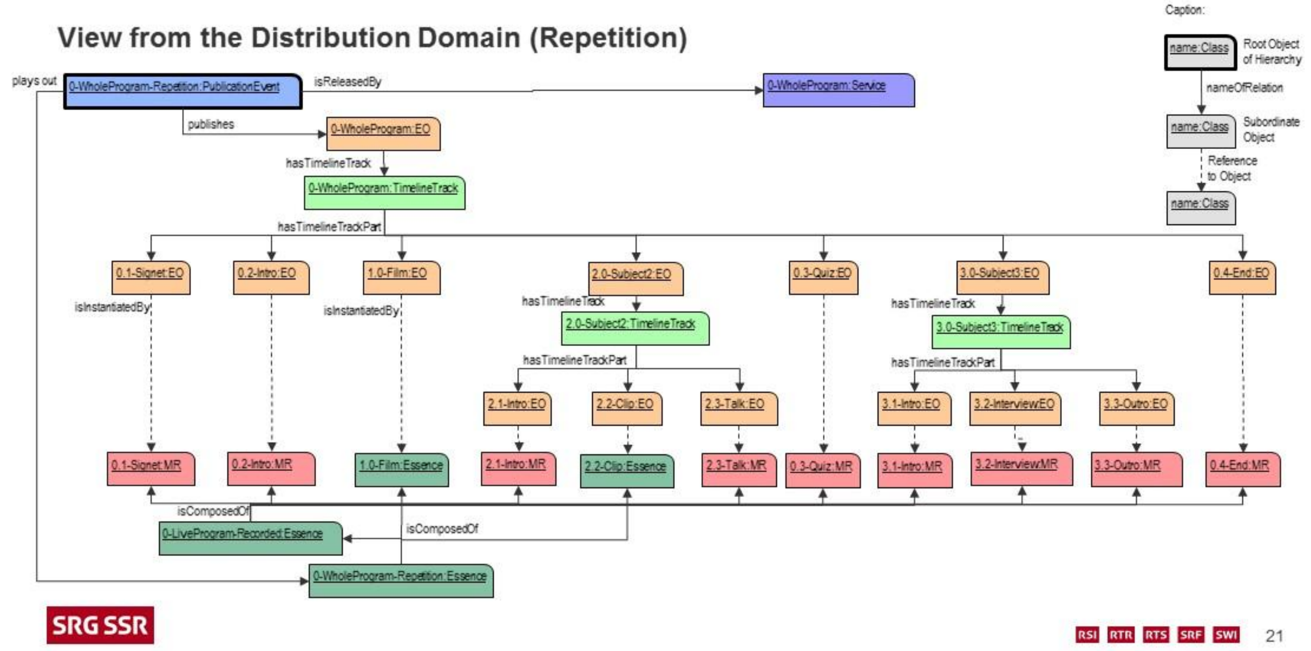
The following diagrams illustrate how to model the “ideal programme” with EBU CCDM.





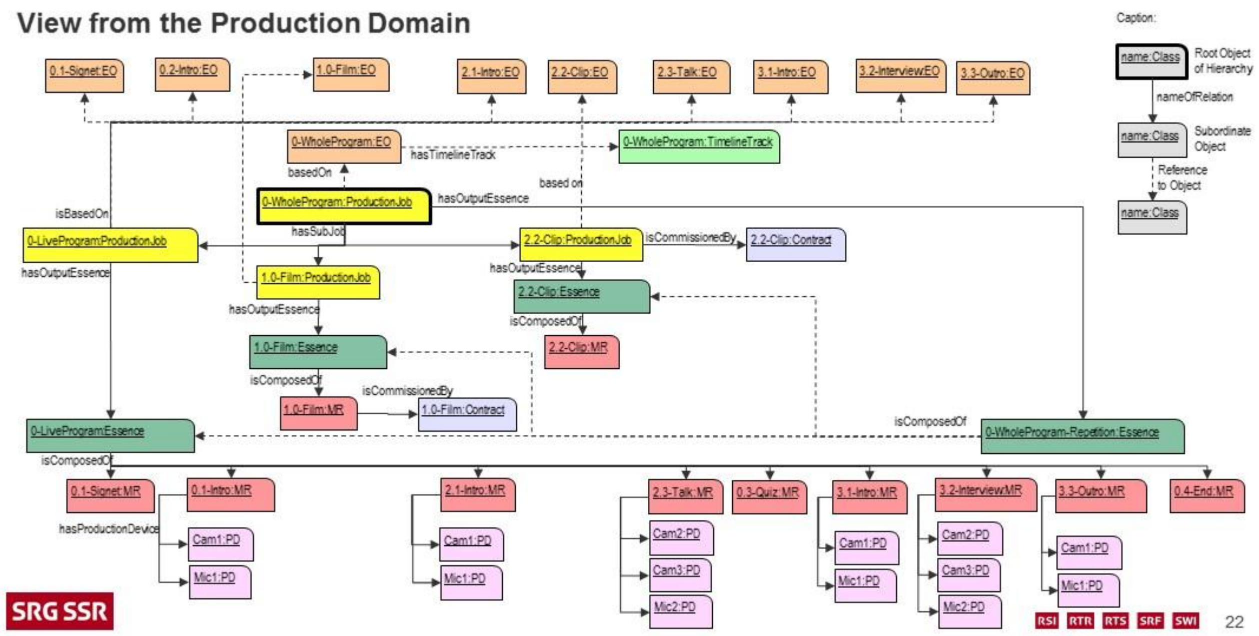
View from the Distribution Domain (Live)

20

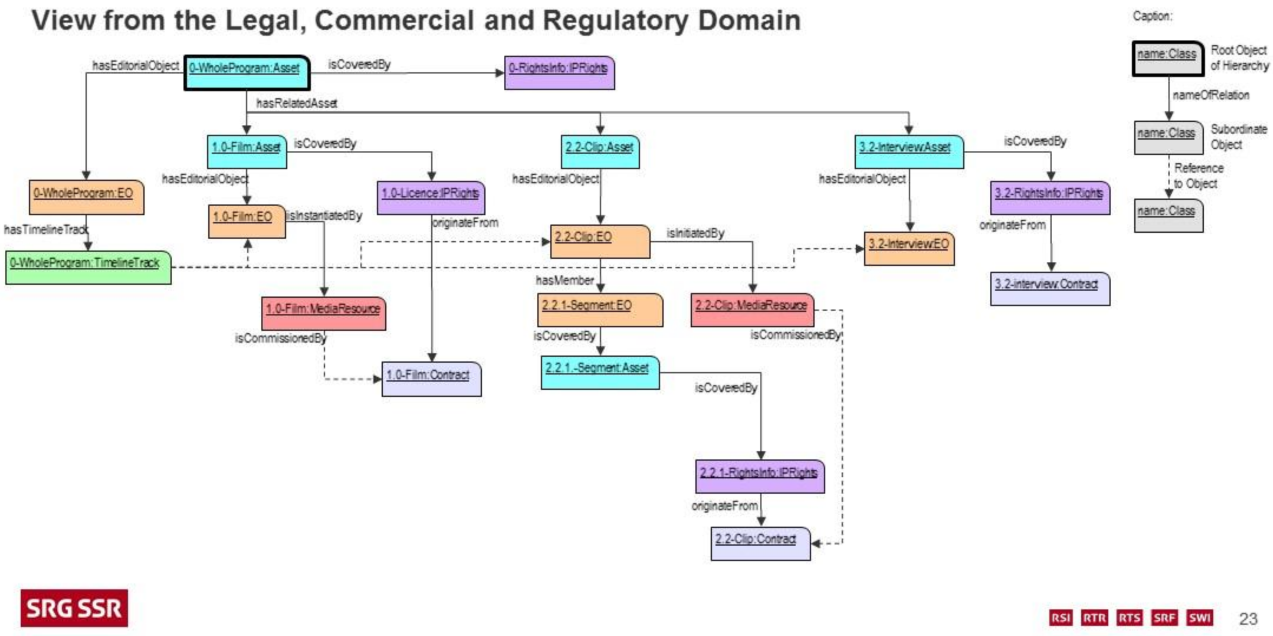


View from the Distribution Domain (Repetition)

21



View from the Production Domain



View from the Legal, Commercial and Regulatory Domain

3.2.2 CCDM as a Comprehensive Representation of Business Objects

Business objects (BO), e.g. a business order or its products, carry business value. Managing this value is crucial to the success of an enterprise. Management relies on data, which must comprehensively represent or describe the business objects.

Figure 11 shows a graph illustrating how a business object is represented by such data.

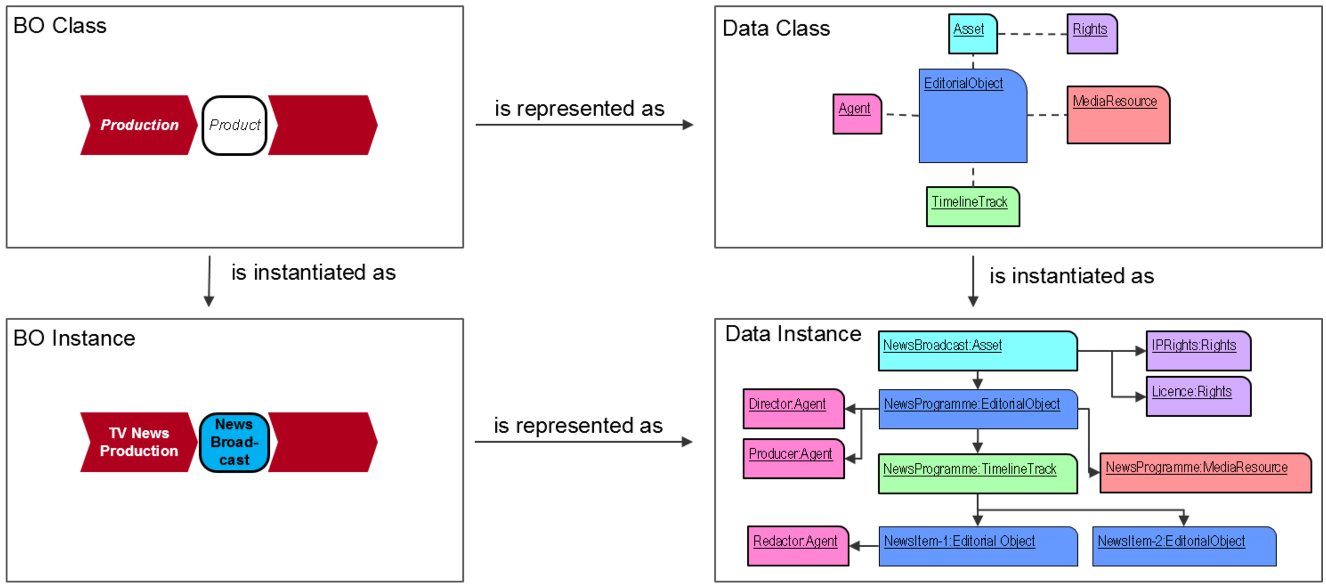


Figure 11: Business objects and associated data

The business object class “Product” is the result of the “Production” process. In real instantiations, this class can take the form of a “News Broadcast” object. A new diagram can be derived from the data. This network of objects is an instance of a generic data class model. The generic class model itself must be designed to represent the business object classes in all required ways.

Consequently, the data model can be evaluated against its ability to represent the largest possible variety of business objects. The EBU has investigated this question and conceived a generic business object and process model for media. The model is a value chain model as shown in Figure 12. It consists of business objects carrying the value, and processes that create value by transforming input objects to (more valuable) output objects.

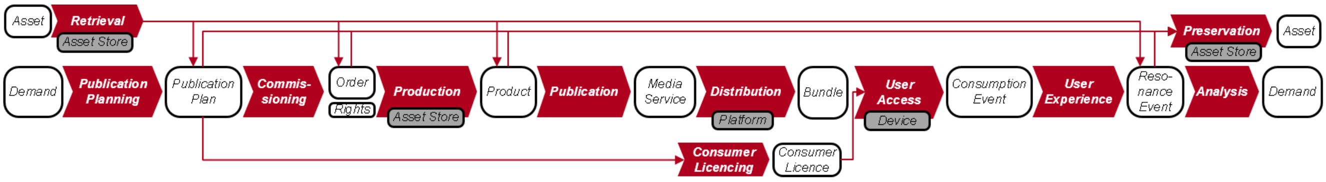


Figure 12: Generic value chain

Every business object in the value chain shown in Figure 10 has to be represented by a set of data.

The graph shown in Figure 13 gives a simplified example. Check the BO “Rights” and the black line. The Rights can be represented by attributes from different data classes. In this case, from Asset (e.g. ID of the product), Rights (e.g. the permissions, obligations and prohibitions) and Editorial Object (e.g. Title, Duration).

Another example is the BO “Product” and the blue line. A “Product” may be represented by *all* attributes from the classes *within* the blue line and by *some* attributes from classes *touched* by the blue line. The same idea applies for the red line and the BO “Media Service”.

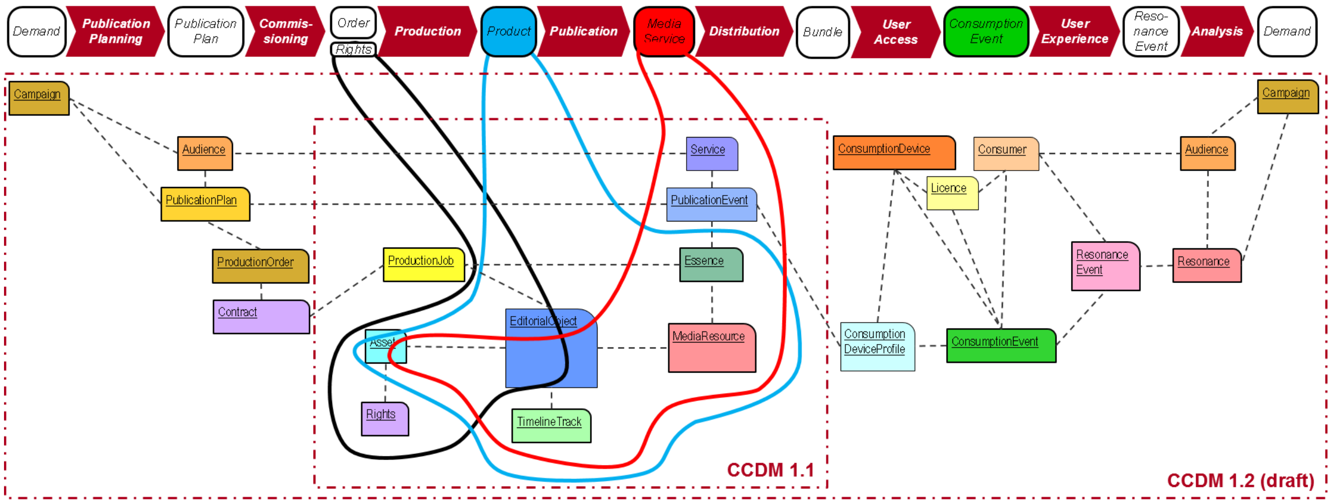


Figure 13: Example of a value chain, business objects and data

This shows that business objects can be represented by a common data model provided by CCDM.

More information on the Modelling Core Business Objects and Processes in Digital Media Enterprises can be found in EBU Tech Report 041 (https://tech.ebu.ch/publications/tr041).

3.3 Example provided by TV2, Norway

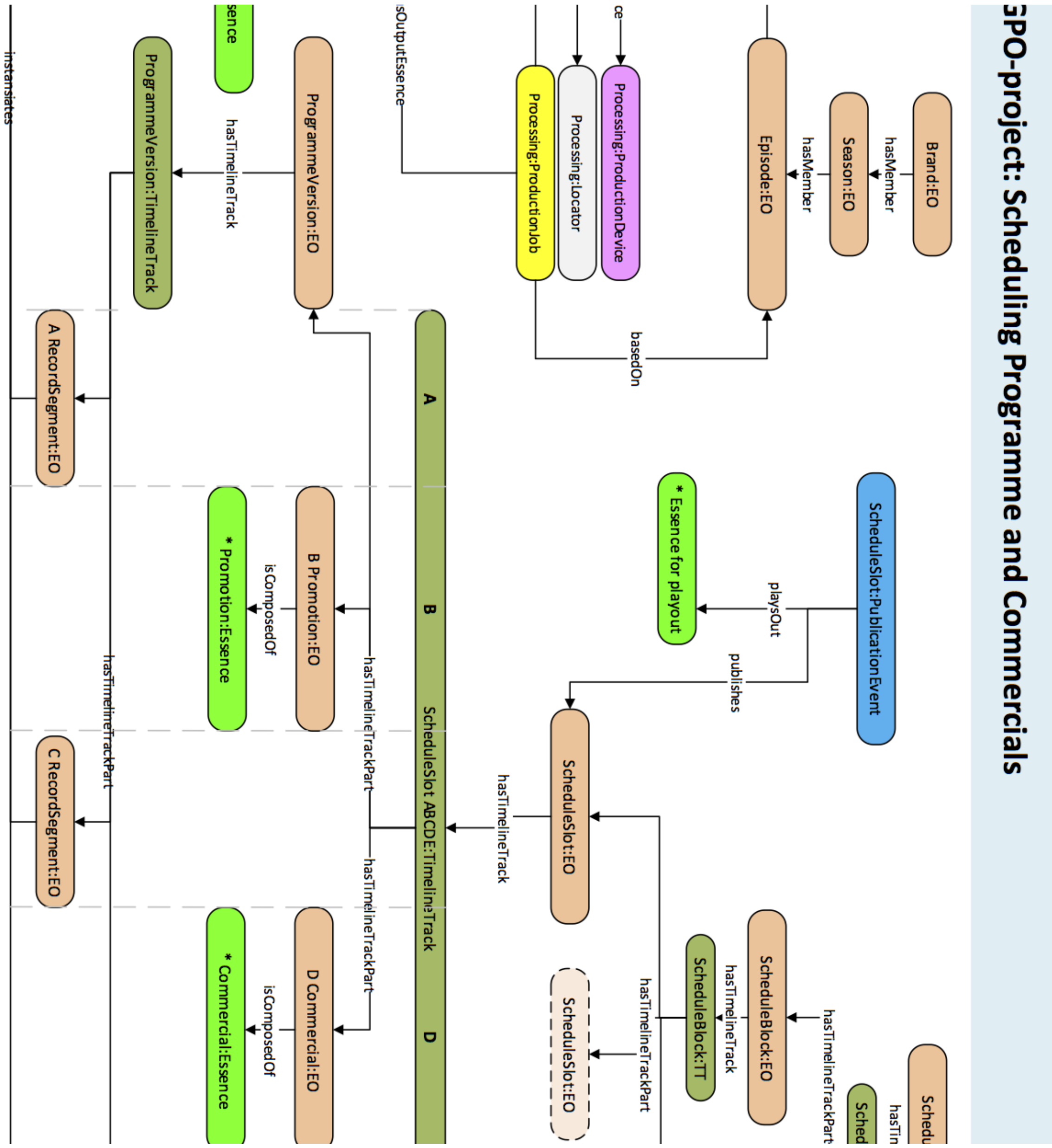


Figure 14:Overall class diagram

3.4 The total class diagram

The overall class diagram can be downloaded from here.

3.5 The RDF ontology

The current specification does purposefully not use specific namespaces or datatypes.

Namespaces and datatypes are defined in ccdm.rdf, which definitions prevail over the current specification text.

EBU CCDM RDF ontology is an extension of EBUCore RDF. This hierarchy can be seen in the CCDM RDF file where the EBUCore imports have been made under the <ebucore> namespace for both classes and properties. CCDM extensions are under the <ebuccdm> namespace.

3.6 More questions?

If you have questions on how to use or implement the EBU CCDM, please forward your queries to metadata@ebu.ch . You will receive personalised advice, and answers will enrich this section of a future version the specification, with your permission.

4. CCDM Compliance

The CCDM is an open framework allowing each user to adapt it to his own needs. As such, the EBU CCDM is flexible and adaptable in nature.

The CCDM ontology is provided as reference software implementation in RDF/OWL. It is available from the "Download Zone". This file contains the minimum set of classes, hierarchies of classes, *objectProperties* and *dataProperties* that compliant implementations should contain, extend, but not replace. More information of the CCDM ontology is provided in **Annex A**.

5. Download Zone

Filename and location	Description
https://www.ebu.ch/metadata/ontologies/ebuccdm/	RDF documentation
https://www.ebu.ch/metadata/ontologies/ebuccdm/ebuccdm.rdf	RDF / XML file

6. Licensing regime

The EBU CCDM is governed by Creative Commons' Attribution-NonCommercial-ShareAlike3.0 Unported (CC BY-NC-SA 3.0)

You are free: to *Share*—to copy, distribute and transmit the work, to *Remix* – to adapt the work, including under your own namespace under the following conditions:

ⓘ	Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
$	Non-commercial - You may not use this work for commercial purposes. *Note: this may be used in commercial products but cannot be sold as a specific feature.*
↺	Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

7. Maintenance

The EBU CCDM specification is maintained by the EBU and suggestions for corrections or additions can be made by mailing to (metadata@ebu.ch).

8. Useful links

EBU Metadata (http://tech.ebu.ch/metadata/)
EBUCore (http://tech.ebu.ch/publications/tech3293)
Modelling Core Business Objects and Processes in Digital Media Enterprises (https://tech.ebu.ch/publications/tr041)
BBC Programmes Ontology (http://www.bbc.co.uk/ontologies/programmes/2009-09-07.shtml)
TV-Anytime (http://www.etsi.org , Standard download in the TS 102 822 series)
W3C – SKOS (http://www.w3.org/2004/02/skos/)

W3C- Resource Description Framework (http://www.w3.org/TR/rdf-primer/)
W3C - Web Ontology Language (http://www.w3.org/TR/owl2-primer/)

Page intentionally left blank. This document is paginated for two sided printing

Annex A: EBU CCDM ontology

The reference software implementation of the CCDM is provided in RDF/OWL.

A link for download is provided in § 5, "Download Zone", of this specification.

There is a variety of options for parsing and editing RDF/OWL documents and ontologies:

- Files with an 'rdf' extension can be opened with text processors such as Wordpad;
- Microsoft Notepad can be used;
- More specialised software can be used:
  - Protégé (http://protege.stanford.edu/download/download.html) (recommended for beginners) – Note: the .rdf extension may need to be changed into .owl
  - TopBraid Composer, free edition (http://www.topquadrant.com/products/TB_Composer.html)