Contributor Attribution Model

Oct 26, 2021

Site Contents

The **Contributor Attribution Model (CAM) Specification** is a standard developed within the Center for Data to Health (CD2H) Architecting Attribution project. It provides a **simple and tightly scoped data model** for representing information about contributions made to research-related artifacts - for example when, why, and how a researcher contributed to a publication, or a curator contributed to a gene annotation record. This core model is intended to be **used as a modular component of larger data models** that underpin data collection and curation systems. Additional components of the CAM specification support implementation of the model, data collection using the model, and ontology-based query and analysis of CAM-based contribution metadata.

The final CAM Specification will be comprised of the following **six components**. More information about each can be found using the links below, or navigating the **Site Contents** menu to the left.

1. A format-agnostic *Information Model* of the domain.
2. A formal *JSON-LD schema* serialization of the Information Model. `coming soon`
3. A *TSV format and curation template*, based on the Information Model. `coming soon`
4. *Ontologies* and *LD-Context Mappings* to support creation of RDF linked data. `coming soon`
5. A *code library and services* to support data creation and format interchange. `coming soon`
6. An *Implementation Guide* to support creation of CAM-compliant data in different formats and contexts. `coming soon`

Note: At present only the foundational *Information Model* has been defined, along with the supporting *Contribution Role Ontology (CRO)*. We expect an initial draft of additional components of the framework to be ready for release in the early 2021.

CHAPTER 1

Overview

The **Contributor Attribution Model (CAM) Specification** is a standard developed within the Center for Data to Health (CD2H) Architecting Attribution project. It provides a **simple and tightly scoped data model** for representing information about contributions made to research-related artifacts - for example when, why, and how a researcher contributed to a publication, or a curator contributed to a gene annotation record. This core model is intended to be **used as a modular component of larger data models** that underpin data collection and curation systems. Additional components of the CAM specification support implementation of the model, data collection using the model, and ontology-based query and analysis of contribution metadata.

1.1 Background and Motivation

Open science, team science, and a drive to understand meaningful outcomes have transformed research at all levels. Scholars and researchers contribute to research and scholarship in ways that can no longer be recognized via traditional means of publication counts and grant dollars received. Efforts to rigorously attribute, evaluate, and reward such contributions must be built on a data models that facilitate nuanced and computable characterization of research products, and the context in which they are developed and used. Unfortunately, little infrastructure exists to identify, aggregate, present, and understand the impact of non-traditional contributions (e.g. curation or data analysis). Challenges to recognizing these contributions are technical as well as cultural, and addressing them requires an approach that assimilates various perspectives for investigators and organizations, alike.

Here we define an ontology-based **Contributor Attribution Model (CAM)** specification that aims to address these challenges. The data model itself was designed to be **as simple, intuitive, and concise a representation as possible**, so as to minimize barriers to adoption by diverse systems where use of community standards has historically been an unexplored or prohibitive endeavor.

The specification as a whole includes the following components:

1.2 Specification Components

1. An *Information Model* (IM) that provides an informal, format-agnostic specification for representing contribution data.
2. A *JSON-LD schema* that provides a computable specification of the IM to support data creation and validation. This includes an Linked Data (LD)-context document with mappings between schema elements and ontology terms, enabling the creation of RDF/Linked Data. `coming soon`
3. A *Tabular File Format* representation of the IM that provides a low-tech, curator friendly tool for collecting contribution metadata. `coming soon`
4. *Ontologies* that provide a semantic foundation for the data model. These will support a complete ontological mapping of schema elements and provide terms for value sets - enabling ontological support for operating on contribution metadata. `coming soon`
5. A *Code Library* that can support interconversion of data formats (e.g. tabular to RDF), identifier mappings, and terminology support for building valu sets. `coming soon`
6. An *Implementation Guide* to support creation of CAM-compliant data in different formats and contexts. `coming soon`

1.3 Modeling Scope

The scope of the Contributor Attribution Model is limited to representing the nature of an agent's contribution to a research artifact - specifically when, where, how, and in what context the contribution was made. It achieves this with a simple and concise model composed of three core classes: an **Artifact**, an **Agent** who contributes to it, and a **Contribution** object that sits between them (**Figure 1**).

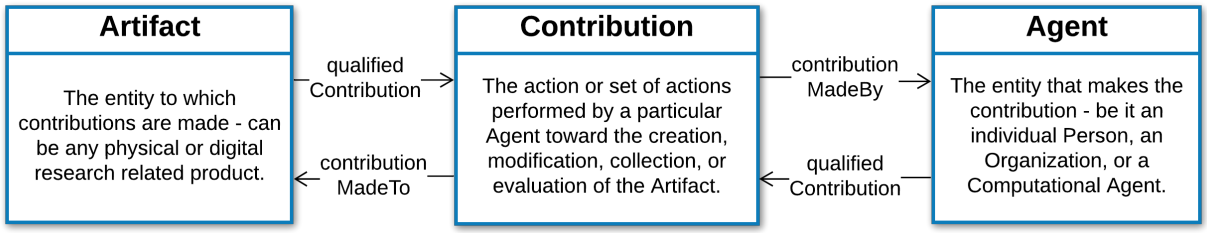

Fig. 1: **Figure 1**: Definition of core classes and relationships in the CAM. Details about each class can be found in the *Information Model*, and a formal specification is provided by a *JSON schema*.

This three object structure is intended to be used as a **module** in the context of a larger data model that captures the complete semantics of a given domain or use case. Implementations can refine or extend the CAM in different ways to suit their specific domain and use case, as described in more detail in the *Implementation Guide*.

1.4 Relationship to PROV

The scope of the CAM overlaps with a subset of the W3C PROV specification that covers agent attribution, but CAM has been tailored to fit the Contribution use case more directly and succinctly. While informed by the work of PROV, CAM was developed independently due to a few small but significant semantic and normative incompatibilities (see *Appendix I*), which prevented PROV from meeting our primary use case for as simple and concise a model as possible to connect an Artifact to an account of Agent contributions.

Mappings between the CAM and PROV models will be provided in *Appendix I*, where areas of semantic incompatibility and efforts toward harmonization are also discussed. Mappings between the CAM and the FHIR Provenance resource, , will also be described here.

1.5 Application Use Cases

Applications implementing CAM-based modules may include:

- **Publication Databases** capturing author contributions to papers.
- **Curated knoweldgebases** collecting information on curators contributions to annotation records as they mature through the system.
- **Research profiling applications** describing contributions to diverse types of scholarly outputs.
- **Research data management platforms** detailing contributions to data objects they manage.
- **Data repositories** capturing contributions to cataloged data sets.
- **Software development platforms** capturing contributions to code and other software artifacts.

In these contexts, the model can support the **collection**, **provision**, and **exchange** of detailed contribution metadata, **display** of this metadata to system users, and the ability to perform precise contribution-related **queries** and **computational analyses**.

1.6 Data Examples

1. An Author Contribution to a Journal Article

This simple example includes minimal metadata describing one author's contribution to the publication of a journal article, structured according to the CAM specification. The Conribution object in this record describes the role the agent played, and the organizational context in which the contribution was made.

```
"id": "doi:10.1371/journal.pgen.1006186", # the Artifact (a published journal article)
"type": "camo:Artifact",
"artifactType": "wd:Q18918145"  # Journal Article,
"label": "Epistatic Gene-Based Interaction Analyses for Glaucoma in eMERGE and␣
↪NEIGHBOR Consortium",
"dateCreated": "2016-09-13",
"qualifiedContribution": [    # the Contribution object
        {
        "id": "ex:contribution001",
        "type": "cro:Contribution",
        "contributionMadeBy":    # the Agent
                {
                "id": "ex:agent001",
                "type": "camo:Agent",
                "label": "Cathy McCarty",
                "externalId": orcid:1234-5678-XXXX
                },
        "realizedRole":   # the specific role they played
                {
                "code": "cro:0000055",
                "label": "study design role",
                "system": "Contribution Role Ontology",
                "systemURL": "http://purl.obolibrary.com/obo/cro.owl"
```

(continues on next page)

(continued from previous page)

```
                },
        "organizationalContext":   # an organization they acted within when making␣
↪the contribution
                {
                "id": "ex:org001",
                "type": "camo:Organization",
                "label": "eMERGE Network",
                "url": "https://emerge.mc.vanderbilt.edu/"
                }
        }
]
```

2. A Curator Contribution to a CIViC Database Record

This richer example includes more extensive contribution metadata from this variant interpretation record in the CIViC Knowledgebase, structured according to the CAM specification. The record includes details of **when**, **how**, **where**, and **in what context** contributions were made by four agents during the life-cycle of this curated record. The example below captures just one of these contributions, but the complete example describing all four dontributions is here.

```
"id": "civic:AID10",  # the Artifact (a curated variant interpretation record)
"type": "camo:Artifact",
"artifactType": "wd:Q49848",
"label": "AID10",
"description": "Vemurafenib and cobimetinib combination is an...",
"url": "https://civicdb.org/api/assertions/10",
"dateCreated": "2018-11-08T16:42:21.820Z",
"qualifiedContribution": [   # the Contribution
        {
        "id": "ex:contribution001",
        "type": "cro:Contribution",
        "endDate": "2018-11-01T18:54:05.924Z",
        "contributionMadeBy":    # the Agent
                {
                "id": "civic:110",
                "type": "camo:Agent",
                "externalId": "orcid:0000-0001-9815-2288",
                "label": "Arpad Danos",
                "_display_name": "arpaddanos",
                "_expertise": "Research Scientist",
                "_orgRole": "admin"
                },
        "realizedRole": [    # here multiple roles are captured in a single␣
↪Contribution object
                {
                "code": "cro:0000XXX",
                "label": "creator role",
                "system": "Contribution Role Ontology",
                "systemURL": "http://purl.obolibrary.com/obo/cro.owl"
                },
                {
                "code": "cro:0000105",
                "label": "submitter role",
                "system": "Contribution Role Ontology",
                "systemURL": "http://purl.obolibrary.com/obo/cro.owl"
                }
           ],
        "organizationalContext":   # an Organization they acted within when making␣
↪the Contribution
```

(continues on next page)

(continued from previous page)

```
                {
                "id": "wd:Q27612411",
                "type": "camo:Organization",
                "label": "Clinical Interpretation of Variants in Cancer (CIViC)",
                "url": "https://civicdb.org/"
                },
        "wasSpecifiedBy":    # a Method that guided how the contribution was made
                {
                "id": "doi:10.1101/700179",
                "type": "camo:Method",
                "label": "The CIViC knowledge model and standard operating procedures␣
↪for curation and clinical interpretation of variants in cancer"
                },
        "occurredAt":   # the Location where the contribution was made
                {
                "id": "civic:214",
                "type": "camo:Location",
                "label": "United States",
                "externalID": "iso:US"
                }
    }
]
```

Additional Notes:

- Expansions of json keys and identifier prefixes in the data above will be provided in a *JSON-LD context file*.
- Attributes preceded by an underscore (e.g. `"_expertise"`) represent extensions to the core CAM model that CIViC might create to capture application-specific content in their system.'
- Additional data examples will be provided as part of the *Implementation Guide*.

Important: How concepts such as **Organizations**, **Methods**, and **Locations** are represented is not in scope for the CAM, and is left to implementations to decide. In the CIViC example above, a structured json object is created to represent these concepts and a few attributes of each. But implementations could choose to capture organization, method, and location values as simple strings if desired, for a much more concise representation. For example:

```
"organizationalContext":  "Clinical Interpretation of Variants in Cancer (CIViC)",

"wasSpecifiedBy": "The CIViC SoP for curation and clinical interpretation of variants␣
↪in cancer",

"occurredAt": "United States"
```

CHAPTER 2

Information Model

Information Models (IM) provide informal, language- and format-agnostic representations of a modeling domain. They abstract over protocol and implementation details to define the characteristics of and relationships between concepts. This provides a unifying conceptual model that conveys the semantics and scope of the domain, and guides concrete specifications in any number of formal schema languages (e.g. JSON schema).

Figure 2 presents a high-level, UML-like diagram of the **Contributor Attribution Information Model (CAM)**. Subsequent pages in this specification, indexed *below* and navigable from the menu on the left, provide detailed descriptions, structural specifications, and implementation guidance for each *Class*, *Data Type*, and *Value Set*.

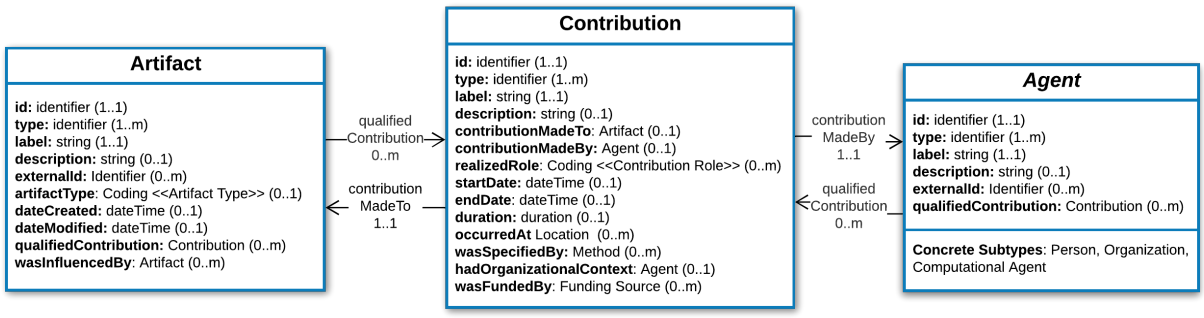

Fig. 1: **Figure 2:** The UML diagram shows attributes of each core **class**, **relationships** between them, and default **cardinality constraints** and **data types** for data collection. Specific implementations can refine and extend the model with to suit their domain and use case, as detailed in the *Implementation Guide*. The *italic* font of the Agent class indicates it is an 'abstract class' that is intantiated only through its subtypes.

Links below point to detailed descriptions of each component in the model.

2.1 Classes

Classes in the CAM are types of entities in the domain for which instances are created in the data. The following pages provide **definitions** of each core CAM class, a detailed **information model** for each, and **implementation guidance** and **examples** to support users creating instances of the class in their data.

Important: Please read the *Terminological and Typographical Convention* notes before continuing.

2.1.1 Artifact

Definition: A physical or digital entity that is created, collected, modified, or cataloged by an agent.

Description and Use:

- Artifacts are the products of agent-driven activities, and represent things to which Contributions are made.
- We are primarily concerned with material or informational artifacts used in generated by research-related activities. This may include natural specimens and man-made archaeological artifacts that are collected, modified, or cataloged for research purposes (e.g. a dinosaur fossil, an arctic ice core sample, prehistoric human tool fragments).
- Because the characteristics of Artifacts will vary dramatically across research activities and domains, the CAM model of Artifacts specifies only generic metadata attributes, and leaves domain-specific characteristics to implementations to define as *formal extensions* of the base Artifact class.

Information Model:

Field	Description	Cardinality	Requirement	Data Type
id	A unique string that identifies the artifact.	1..1	MUST	*identifier*
type	The high-level class to which the artifact belongs (always set to 'Artifact').	1..1	MUST	*class*
label	A free-text name for the artifact.	0..1	MAY	*string*
description	A free-text description of the artifact.	0..1	MAY	*string*
externalID	Additional identifier(s) for the artifact that come from an external system or authority.	0..m	MAY	*identifier*
artifactType	A more specific type for the artifact.	0..m	SHOULD	coding <<*Artifact Type*>>
dateCreated	The date on which the current version or form of the artifact was completed.	0..1	MAY	*dateTime*
dateModified	The date on which the artifact was last updated or modified.	0..1	MAY	*dateTime*
url	A URL where information about the Artifact can be found.	0..m	MAY	*url*
qualifiedContribution	A particular contribution made by an agent to the artifact.	0..m	MAY	*Contribution*
influencedBy	A separate artifact that directly or indirectly influenced creation of the artifact of interest.	0..m	MAY	*Artifact*

.

Examples:

- The **Contribution Role Ontology**

- A **poster** and abstract submission about the Architecting Attribution project
- A **HeLa cell line**
- A **protocol** for culturing cell lines
- A **dataset** about tetrapod bony lesions
- A **catalog entry** for a centrifuge instrument
- The **centrifuge** itself
- The **CIViC knowledgebase** containing curated information about cancer mutations
- An **individual record** from CIViC about the BRAF V600E mutation.
- A **dinosaur fossil** collected and cataloged from a research site.
- A **prehistoric tool fragment specimen** collected and cataloged form an archaeological site.
- An **ice specimen** collected from an arctic glacier.
- A **catalog record** describing the ice specimen.

Implementation Notes:

- **Using the Artifact Type Value Set** (*artifactType*)
  - Use of the *Artifact Type Value Set* that is bound to the *artifactType* attribute above is RECOMMENDED but not required.
  - Implementations can choose to refine or extend this value set that we provide as part of the CAM specification, or use their own, as described in the *Implementation Guide*.
- **Artifact Identifiers** (*id* and *externalID*):
  - Artifact identifiers can be captured using the *id* and *externalID* attributes. The *id* attribute MUST hold a single identifier that will be used to track/reference the artifact in an implementing system. This can be an internal de novo identifier, or one borrowed from an external resource or registry (e.g. a PMID for a publication, or ISBN for a book).
  - Additional external identifiers for the artifact MAY be captured using the *externalID* attribute. For example, the publication described in this Mouse Genome Informatics record has an internally-minted identifier (`J:33382`) that may be captured in the *id* slot, and an external Pubmed identifier (`8662814`) that can be captured in the *externalID* slot.
- **Typing of Artifacts** (*type* and *artifactType*)
  - The type attribute MUST be filled with the generic 'Artifact' type. To capture a more specific artifact type, implementations can use the *artifactType* attribute and bind it to a value set of terms that is suited for their domain and use case. We RECOMMEND using the *Artifact Type Value Set* provided as part of this specification, which can be used in whole or in part, and refined/extended as needed. But ad hoc value sets can be defined and used if desired.
- **Artifact Modification** (*dateModified*)
  - The meaning of 'modified' may vary depending on artifact type and context of use, and SHOULD be clarified by a given implementation.
  - For material artifacts, this can include physical alterations or additions that maintain the identity of the artifact. For informational artifacts, this can include updates to content or structure that do not result in a new version with a separate identity in the system of record.
- **Natural and Archaeological Artifacts** (*dateCreated* and *dateModified*)

  - Many natural or archaeological artifacts originate outside of a research setting, and are only collected and documented as specimens much later (e.g. a dinosaur tooth fossil, or prehistoric tool fragments). Here, *dateCreated* SHOULD be used to record the date such specimens were taken, not the date the collected material originally came into existence (which may have been thousands or millions of years ago). Similarly, *dateModified* SHOULD be used to record when modifications were last made to the specimen in a research context (e.g. its last cleaning or sample extraction).
  - In cases where natural specimens are observed and documented, but not physically collected or modified, we RECOMMEND describing contributions to a catalog record about the specimen (as there are no contributions to physical specimen itself to track).

- **Influence Relationships Between Artifacts** (*influencedBy*)
  - The notion of an 'Influence' between two artifacts broadly describes scenarios where one is directly or indirectly used in the creation of another. It is based on the PROV notion of influence - but narrower in that it applies here only between two Artifacts.
  - Influences can include derivation or incorporation of material or informational content - e.g. a cell line being derived from a tumor specimen, use of a jpg image into a blog post, or a format translation from a JSON dataset to an RDF version of the dataset. Influences can also cover an artifact providing a source of information used to generate an artifact with entirely separate content - e.g. a dataset on ice core CO2 levels used as evidence for an assertion about the rate of arctic climate change, a microscope/camera used to take images of tissue samples, or a knockout mouse strain used in to generate data about blood glucose levels which support a phenotype annotation made on the deleted gene).
  - The CAM defines a single, generic *influencedBy* attribute to describe the artifact-artifact relationship in such scenarios. But implementations MAY define specializations of this attribute with more specific meaning - e.g. *derivedFrom*, *revisionOf*, *informedBy*, *providesEvidenceFor*, etc.

2.1.2 Contribution

Definition: An action or set of actions performed by an agent toward the creation, modification, evaluation, or deprecation of an artifact.

Description and Use:

- The Contribution class is the the core object in the CAM. They mediate a link between an Agent and an Artifact, and organize information about when, where, how, and in what context the agent contributed to the artifact.
- Ontologically, contributions are considered a type of process or activity. The scope of a Contribution activity includes only the actions of a single agent in contributing to a particular artifact, which may be performed in one continuous effort, or several discontinuous sessions of work.
- Contribution activities may include those leading to the initial creation or subsequent modification of an artifact, as well as activities that evaluate the artifact (e.g. to gauge its quality, completeness, status, or utility), or activities that deprecate an artifact such that it may no longer be used.

Information Model:

Field	Description	Cardinality	Requirement	Data Type
id	A unique string that identifies the contribution.	1..1	MUST	*identifier*
type	The high-level class to which the contribution belongs (always set to 'Contribution').	1..1	MUST	*class*
label	A free-text name for the contribution.	0..1	MAY	*string*
description	A free-text description of the contribution.	0..1	MAY	*string*
contributionMadeTo	The artifact toward which the contribution was made.	0..1	SHOULD	*Artifact*
contributionMadeBy	The agent that made the contribution.	0..1	SHOULD	*Agent*
realizedRole	A role indicating the nature of the contribution.	0..m	MAY	coding <<*Contribution Role*>>
startDate	The date and/or time that the contribution activity began.	0..1	MAY	*dateTime*
endDate	The date and/or time that the contribution activity ended.	0..1	MAY	*dateTime*
duration	The total amount of time the agent spent performing the contribution.	0..1	MAY	*duration*
occurredAt	The location or locations where the contribution was performed.	0..m	MAY	*Location*
wasSpecifiedBy	A specification (protocol, ruleset, method, guidelines) describing how all or part of the contribution was executed.	0..m	MAY	*Method*
organizationalContext	An organization whose resources and/or directives drive the contribution made by an Agent.	0..m	MAY	*Organization*
wasFundedBy	A grant or other source of resources that paid for the work representing the contribution.	0..m	MAY	*Funding Source*

.

Examples:

- The writing performed by a person toward the creation of a scientific publication (realizes an 'author role')
- The act of sharing of frozen embryonic fibroblasts performed by a researcher toward the creation of a transgenic mouse (realizes a 'resource provision role')
- The task of assembling a lecture slide deck performed by a graduate student in creating an online educational course offering (realizes an 'educational material development role')
- The performance of quality control checks on an integrated data set performed by a data steward toward the creation of a curated knowledgebase (realizes a 'quality assurance role')
- The act of disposing of a transgenic cell line that was determined to be contaminated.
- The deprecation of an ontology term that its developers decide should not longer be used.

Implementation Notes:

- **Using the Contributor Role Value Set**
  - Use of the Contributor Role Value Set to populate values for the *realizedRole* attribute above is RECOMMENDED but not required. This value set uses terms from the Contribution Role Ontology (CRO) .
  - Implementations can choose to refine or extend this value set that we provide as part of the CAM specification, or use their own, as described in the *Implementation Guide*.
- **'Placeholder' Classes** (Location, Method, Funding Source)
  - At present the CAM considers modeling of Location, Method, and Funding Source to be out of scope, and does not provide concrete models for them. Rather, implementations can determine if and how they want to

represent these concepts, and model them accordingly. For example, a Location could be captured simply as a free-text string, or an identifier or code from some controlled vocabulary (e.g. any of a number of existing gazetteers, such as Geonames), or using a bespoke or standard location schema (e.g. ISO19112), to create location objects allowing more precise and flexible descriptions.

- **Specifying a Contribution Date** (*startDate* and *endDate*):
  - The model provides *startDate* and *endDate* attributes to allow precise reporting of the time the contribution occurred. Implementations wishing to specify a single time SHOULD simply report the date and/or time that the contribution ended, using the *endDate* attribute, and the *startDate* attribute SHOULD NOT be filled. An empty *startDate* means that the start time is unknown or unspecified.
- **Capturing Multiple Contribution Roles**:
  - A contribution MUST connect a single Agent to a single Artifact. But a single Contribution object MAY capture multiple roles played by the agent in generating the artifact - simply by including more than one CRO contribution role in the *realizedRole* slot. This pattern provides a more concise representation when data creators do not need to capture unique details about each realized role (i.e. how, when, and where each was performed). In cases where such details for each role are required, separate Contribution objects SHOULD be created for each role played.
- **Direction of Contribution Relationships**:
  - Relationships to and from an Agent or Artifact can be created in a Contribution object. Note here that the *qualifiedContribution* relationship can be used to connect either an Artifact or an Agent to a Contribution. The relationships and their directionality selected will depend on whether an Artifact- or Agent-centric perspective on the data is being captured. When the goal is to describe all contributions made to a particular Artifact, links are created from Artifact to Contribution to Agent. When the goal is to describe all contributions made by a particular Agent, links are created from AGent to Contribution to Artifact.

2.1.3 Agent

Definition: An autonomous actor that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity.

Description and Use:

- Agents make contributions to Artifacts. An agent can be an individual Person, an Organization of multiple individuals acting together, or a Computational Agent such as a software program or algorithm.
- Agents can belong to Organizations, and act on behalf or in the context of an Organization when contributing to an Artifact. For example, curators generating these variant pathogenicity interpretations do so in the context of the ClinGen organization, which provides administrative, technical, and funding support for their contributions.
- The CAM model for Agent is minimal - providing only a few generic attributes. Future versions may provide more detailed specifications, but at present it is up to implementations to extend this base model as needed to meet their use cases.

Information Model:

Field	Description	Cardinality	Requirement	Data Type
id	A unique string that identifies the agent.	1..1	MUST	*identifier*
type	The high-level class to which the agent belongs ('Person', 'Organization', or 'Computational Agent').	1..1	MUST	*class*
label	A free-text name for the agent.	0..1	MAY	*string*
description	A free-text description of the agent.	0..1	MAY	*string*
externalID	Additional identifier(s) for the agent that come from an external system or authority.	0..m	MAY	*identifier*
url	A URL where information about the agent can be found.	0..m	MAY	*url*
qualifiedContribution	A particular contribution made by the agent to an artifact.	0..m	MAY	*Contribution*

.

Implementation Notes:

- **Agent Instances**:
  - Agent is an **'abstract class'**, and therefore SHOULD NOT be directly instantiated. A concrete subclass (*Person*, *Organization*, or *Computational Agent*) SHOULD be selected to represent any agent in the data. The attributes defined for the abstract Agent class are inherited and extended by its child classes.
- **Agent Identity Criteria**:
  - Agent instances in CAM are scoped to be independent of the context in which the agent operates. An instance of a Person agent represents _the *person*, not the person _as they contribute to a particular project or *organization*. So agent 'identity' is based only on who the person is, not additionally on the context in which they are acting. This is different than how some models, including PROV, scope the notion of an Agent to represent a person acting in a particular context. We applied the former perspective on Agent identity, as we felt it to be more flexible, simpler to apply, and better facilitate data query and integration.
  - This decision has implications for how an Agent instances are created in the data. For example, consider a person 'Steven' who is a curator in different projects for the ClinGen and ENIGMA organizations. In a CAM-based dataset, there would be a _single Agent **instance_** representing Steven, who may contribute to some curated artifacts as an agent for ClinGen and others as an agent for ENIGMA. The context of these contributions is captured in the 'hadOrganizationalContext' field of the Contribution object. In a PROV dataset, there would be _two separate instances for **Steven_** - one for representing his actions on behalf of ClinGen, and the other for representing his actions on behalf of ENIGMA. Here the context of contribution is part of the Agent object itself.
- **Extensions to the Agent Class**:
  - Implementations requiring more than the minimal attributes provided for describing agents MAY extend the model as needed, following the recommendations in the *Implementation Guide*. Extesnion may involve creating additional attributes for exsiting classes, and/or defineing additional subtypes. We recommend use of community standards where possible for describing agents (e.g. FOAF), to enable data sharing and interoperability.

2.1.3.1 Person

Definition: A human being.

Description and Use:

- Persons are human beings with agency who contribute to artifacts.

- Person is a **concrete subclass of Agent**. It can be instantiated in the data, and inherits from its abstract parent Agent class.
- At present the CAM does not define additional person-specific attributes - preferring a minimal but extensible model that allows implementation control over how Persons are represented.

Information Model:

- This class inherits from the *Agent* class defined above, and does not define additional attributes.

Examples:

- In the research domain, examples may include an individual researcher, teacher, curator, administrator, technician, ...

Implementation Notes:

- **Person Identifiers**:
  - Identifiers for Persons SHOULD come from an open and authoritative provider. ORCIDs are becoming a de facto standard in this space, and are strongly recommended. But other identifier systems/providers exist (e.g. ResearcherID, ISNI).
- **Person Contributions in Organizations**:
  - If the person making a contribution is acting as a member of an organization, this MAY be captured using the *organizationalContext* attribute of the Contribution class. Alternatively, the Organization itself may be indicated as the agent making the contribution.
- **Extensions to the Person Class**:
  - As noted above, this minimal Person model is meant to be extended with additional attributes by implementations as needed to suit their requirements. Examples of information that may be useful include given and family name, date of birth, employer, organizational memberships, professional titles, etc. We recommend looking to community standards such as FOAF to provide modeling constructs here, to facilitate data sharing and interoperability.
- **Membership in Organizations:**
  - The details of a Person's membership in an Organization (dates they were active, roles/status they held, etc.) could support advanced queries relating contributions to organizational membership (e.g. "Find all contributions made in 2014 by persons who were members of organization X that year"). Details about organizational membership are not yet represented in the CAM, but this capability may be added in the future, and for now implementations can define their own models to support these use cases.

2.1.3.2 Organization

Definition: A group of agents (typically persons) structured and managed to achieve a common goal.

Description and Use:

- Organization is a **concrete subclass of Agent**. It can be instantiated in the data, and inherits from its abstract parent Agent class. But at present the CAM does not define additional organization-specific attributes.
- Organizations as defined here may include and group of individuals organized around a shared task or goal, including companies, academic interest groups, consortia, charitable foundations, formal research collaborations. A group does not have to be formally recognized as an organization to be treated as such in the CAM.
- An organization can collectively be credited as making a contribution when it directly or indirectly supports the creation or modification of an artifact, e.g. through funding, defining, directing the work, and/or through its members performing the work.

Information Model:

- This class inherits from the *Agent* class defined above, and does not define additional attributes.

Examples:

- Scientific consortia such as the Global Alliance for Genomics and Health and ClinGen.
- Businesses/companies such as the publisher Elsevier
- A medical non-profit organization scuch as the American Heart Association
- A government agency such as the Centers for Disease Control
- A philanthropic fraternity such as Shriners International
- A group of students assigned to complete a class project

Implementation Notes:

- **Capturing Organization Contributions:**
  - Contributions of an organization to an artifact can be captured in two ways: **(1) Directly** as the asserted agent in a contribution, in cases where it reflects the collective effort of a group, or it is not important to capture the roles or actions of individual members acting on behalf of an organization; or **(2) Indirectly** as an organization on whose behalf an individual member's contribution was performed (using the *organizationalContext* attribute of the Contribution class).
- **Extensions to the Organization Class:**
  - As noted above, this minimal Organization model is meant to be extended with additional attributes by implementations as needed to suit their requirements. We recommend looking to community standards such as FOAF to provide modeling constructs here, to facilitate data sharing and interoperability.

2.1.3.3 Computational Agent

Definition: A software program or algorithm that can autonomously execute tasks to achieve a specified goal.

Description and Use:

- A software program need not conceive of or initiate the task on its own to be considered an agent - it must only be capable of autonomously executing a programmed task.

Information Model:

- This class inherits from the *Agent* class defined above, and does not define additional attributes.

Examples:

- An algorist (computer algorithm designed to generate art)
- An algorithm that automatically generates an ontology by mining text for Wikipedia

Implementation Notes:

- **Inferring 'Transitive Credit' to Software Creators**
  - As man-made entities, computational agents can be considered Artifacts, and contributions to them can be captured using the CAM. When this is done, 'transitive credit' can be inferred for software creators for Artifacts the software produces.

2.1.4 Placeholder Classes

The CAM defines the value of the *occurredAt*, *wasSpecifiedBy*, and *hadFundingSource* attributes of the *Contribution* object as a *Location*, *Method*, and *Funding Source*, respectively. At present, however, classes to represent these

concepts are condidered out of scope, and the specification does not provide concrete models for them. We refer to these as '**Placeholder Classes**', and provide descriptions and modeling recommendations for each. But we ultimately leave it to implementations to determine if and how they want to represent these concepts.

2.1.4.1 Location

Definition: A place where a contribution was made.

Description and Use:

- Location is defined as the type of value for the *occurredAt* attribute of the Contribution class.
- Locations can be referred to in various ways at various levels of granularity and formality - e.g. politically-defined areas such as countries, states or cities, street addresses, zip codes, buildings, spatial coordinates (e.g latitude/longitude).
- Such locations can be referenced simply as a free-text string, or an identifier or code from some controlled vocabulary (e.g. any of a number of existing gazetteers, such as Geonames), or using a bespoke or standard location schema (e.g. ISO19112), to create location objects allowing more precise and flexible descriptions.
- Particular implementations can refine this concept as needed for their use case, and define a formal specification for how to represent it.

2.1.4.2 Method

Definition: A set of instructions that specify all or part of how a contribution was made.

Description and Use:

- Method is defined as the type of value for the *wasSpecifiedBy* attribute of the Contribution class. A Method SHOULD be referenced in a Contribution object if it informed part or all of the contribution process. In practice, not all contributions are guided by a referenceable method.
- Methods are directive informational artifacts that specify what actions to perform and how to perform them. They can be written at a very general or very precise level of detail.
- Examples include a recipe describing how to make a buffer solution for research, a protocol specifying how to execute an experiment to generate a specific type of data, or rules/guidelines for data curation or interpretation that specify how information can be organized, evaluated, and interpreted to generate new knowledge.
- Particular implementations can refine this concept as needed for their use case, and define a formal specification for how to represent it.

2.1.4.3 Funding Source

Definition: A funding body or mechanism that provided resources supporting a contribution.

Description and Use:

- Funding Source is defined as the type of value for the *wasFundedBy* attribute of the Contribution class.
- Particular implementations can refine this concept as needed for their use case, and define a formal specification for how to represent it.
- Funding Sources MAY be represented at different levels of specificity, including:
  - A **particular funding body** (e.g. 'NIH', 'NSF', 'Elsevier Labs', an individual person, etc.)
  - A **type/category of funding mechanism** (e.g. 'grant', 'NIH grant', 'R01 grant')

- A **particular awarded grant or contract** (e.g. 'NIH award no. 1R24OD011883')

2.2 Data Types

'Data type' specifications define more fundamental, general-purpose constructs that provide structure and semantics to data. In addition to a set of classes, the CAM provides a handful of simple and complex data types that are used to specify the type of value that a given attribute may take.

2.2.1 Simple Data Types

2.2.1.1 string

Definition: a sequence of Unicode characters

Description and Use:

- The string data type represents free-text that is not constrained to some code set or associated with some semantic interpretation
- It is the most generic form of free-text literal in our model, and conceptually subsumes string-based data types with more constrained interpretations (e.g. code, identifier, url) identifiers)
- It is most often used to capture names and free-text descriptions of entities in the model or data.

Examples of attributes taking strings as values:

```
"label": "author role"
"title": "Semantic Web for Dummies"
"description": "A cell line derived from cervical tumor cells of a cancer patient"
```

2.2.1.2 url

Definition: A Uniform Resource Locator (RFC 1738) representing a web address where a resource can be found or information about the resource discovered. Common URL protocols are http{s}:, ftp:, mailto: and mllp:, though many others are defined.

Description and Use:

- Used in place of a plain string datatype when a resolvable web address for a resource is desired.
- URLs with an http{s} protocol are preferred where available.

Examples of attributes taking a url as values:

```
"systemURL": "http://purl.obolibrary.org/obo/cro.owl
```

2.2.1.3 code

Definition: a string value that is taken from a set of controlled strings defined by some system or authority

Description and Use:

- A code can be opaque (e.g. `CRO:0000001`, `90210`) or convey meaning (e.g. `author`, `AFR`, `PDX`).
- Codes are used primarily within a Coding object, where they hold the core value of the Coding. Codings and codes are used in Value Sets bound to selected attributes to control data entry.

- Codes are used primarily within a Coding object, where they hold the core value of the Coding. Codings and codes are used in Value Sets bound to selected attributes to control data entry.
- Whenever possible a code should be represented as a URL or prefixed identifier that indicates the code system that provided it. If this is not possible, the code system can be captured in the "system" attribute of the coding object holding the code.

Examples of data structures using codes as values:

A 'Coding' object, in which the 'code' element holds the code "CRO:0000001":

```
"realizedRole": {
   "system": "http://purl.obolibrary.org/obo/cro.owl"
   "systemVersion": 1.0,
   "code": "CRO:0000001"
   "label": "author role"
}
```

2.2.1.4 identifier

Definition: A string that uniquely identifies a specific instance of an object in a dataset or document.

Description and Use:

- An identifier refers to exactly one instance, but a given instance in the data may be associated with more than one identifier.
- Identifiers SHOULD be globally unique, persistent, and machine-resolvable.
- Use of the W3C Compact URI (CURIE) syntax to structure identifiers is one way to meet these criteria. CURIEs consist of prefix and reference components that correspond to a namespace and local identifier within that namespace, respectively, and are deterministically expandable into a full URI.
- It is strongly RECOMMENDED, but not required, that identifiers are represented using CURIE or URI syntax. General guidelines and identifier best practices, including use of CURIEs, are well described here and here.
- Instance identifiers can be generated de novo by a system creating data, or borrowed from external systems/authorities that mint identifiers (e.g. databases, registries, ontologies). For example, a Person instance can be assigned a de novo internal identifier by the system generating the record (e.g. ex:12345), or the system can choose to use an established ORCID for that person (e.g. `orcid:0000-0002-1048-5019`). For de novo identifiers, we strongly recommend the use of a services such as w3id to enable web resolution (even if simply resolving to an informal document providing minimal metadata about each identifier).
- If a system wishes to capture both an internal and external identifiers, the internal one should be captured in the *identifier* attribute, and the *externalId* attribute can be used to record one or more established identifiers from community authorities.

Examples of attributes taking identifiers as values:

CURIEs:

```
"identifier": "orcid:0000-0002-1048-5019"
"identifier": "pmid:123456"
"identifier": "ga4gh:VX60NSGLem4X3Q8gnOSx48pZDCmJVSUk"
"externalId": "doi:10.1111/2041-210x.12970"
"externalId": "IGSN:ECS000004"
```

Full URIs:

```
"identifier": "http://doi.org/10.1111/2041-210x.12970"
"identifier": "http://n2t.net/ark:/65665/3f09e18cb-fb47-4ea8-ae3b-694a89ea8ff4"
```

Non-CURIE/URI Identifiers:

```
"identifier": "USNMENT00945454"
"identifier": "MGI41353"
```

2.2.1.5 class

Definition: a string value representing the class of objects to which an instance belongs.

Description and Use:

- This string MUST be the name of a concrete class from the base CAM specification (i.e. one of Artifact, Contribution, Person, Organization, Computational Agent, Location, Method, Funding Source), or a specialization of such a class in an extension defined by a particular implementation.
- Note that when an attribute bound to a value set takes an ontology class (e.g. a CRO term in the Contribution Role Value Set), the Coding data type MUST be used - as the class data type is reserved for specifying the type of proper instances in a dataset.

2.2.1.6 date

Definition: specifies a date comprised of a year, month, and day, following the form "YYYY-MM-DD", with an optional time zone parameter.

Description and Use:

- The date data type is used when precision down to a specific day is required, but not a specific time of day.
- Conventions for representing dates defined here are based on the ISO 8601 standard.
- A time zone can be indicated by appending a "Z" to specify UTC time (e.g. `"1978-01-20Z"`), or specifying an offset from UTC time by adding positive or negative time after the date (e.g. `"1978-01-20+5:00"`)

Examples of attributes taking dates values:

```
"startTime": "2018-04-12TZ"
"startTime": "1998-12-18T+06:00"
```

2.2.1.7 dateTime

Definition: specifies a date and time of day comprised of a year, month, day, hour, minute, and second, following the form "YYYY-MM-DDThh:mm:ss"

Description and Use:

- The dateTime data type is used when precision down to a specific time of day is required.
- Conventions for representing date + time defined here are based on the ISO 8601 standard.
- Hours of the day should use 24-hour time (e.g. 2PM = 14:00:00)
- A time zone can be specified as described for 'dates' above, e.g. `"1978-01-20T05:15:00Z`, and `"1978-01-20T05:15:00+5:00"`.

- If precision down to time of day is not possible or necessary, the hours, minutes, and seconds must still be provided due to schema constraints, but can all be set to "00" and ignored at the instruction of the message sender.

Examples of attributes taking dateTime values:

```
"startTime": "2018-04-12T11:05:25+08:00"
"startTime": "2004-09-05T00:00:00Z"
```

2.2.1.8 duration

Definition: specifies a length of time, using the form "PnYnMnDTnHnMnS"

Description and Use:

- The duration data type should be used to directly assert the length of time a contribution took to perform.
- Conventions for representing duration defined here are based on the ISO 8601 standard.
- A 'P' prefix and at least one of the temporal components below is required in a duration value:
  - 'nY' indicates a number of years
  - 'nM' indicates a number of months
  - 'nD' indicates a number of days
  - 'T' indicates the start of a time section (required to specify precision down to hours, minutes, or seconds)
  - 'nH' indicates a number of hours
  - 'nM' indicates a number of minutes
  - 'nS' indicates a number of seconds

Examples of attributes taking duration values:

```
"duration": "P8Y"  (8 years)
"duration": "P8Y4M15D"  (8 years 4 months 15 days)
"duration": "P8Y4M15DT2H6M35S"  (8 years 4 months 15 days 2 hours 6 minutes 35␣
↪seconds)
"duration": "P8Y215D"  (8 years 215 days)
"duration": "PT2H6M35S"  (2 hours 6 minutes 35 seconds)
"duration": "PT35S"  (35 seconds)
```

2.2.2 Complex Data Types

In contrast to 'simple' data types, 'complex' data types specify structured objects with attributes of their own - often combining primitives (and other complex data types) to express increasingly complex semantics. These data types do not represent primary domain entities, and are not given identifiers in the data. .. _coding:

2.2.2.1 Coding

Definition: a structured representation of a *code* from some code system, that includes additional metadata about the code and code system.

Description and Use:

- The specification for this data type was adapted from the FHIR 'Coding' data type. It provides a structure to include additional metadata about a 'code' used to capture data in a dataset - for example, the human-readable name for the code, or the code system from which it came. In this way it differs from the primitive 'code' data type, which specifies only a string value representing the code itself.
- The CAM opts to require a Coding object in places where a value set is bound to an attribute, rather than allowing direct capture of the code alone, as we believe that users will minimally want to see the human-readable name of the code in a message or dataset.

Information Model:

Field	Description	Cardinality	Requirement	Data Type
code	The value of the code itself.	1..1	MUST	*code*
label	A human-readable name for the code.	0..1	MAY	*string*
system	The code system where the code is defined.	0..1	MAY**	*string*
systemURL	A URL where the code system can be found.	0..1	MAY	*url*
systemVersion	The version of the code system.	0..1	MAY	*string*

**The system attribute is left optional (MAY) under the assumption that most codes will be defined as a URI or prefixed identifier that captures the defining code system, as recommended in the specification for the "code" data type. If this is not the case, the code system MUST be indicated using the system attribute.

Examples of data structures using Codings as values:

A 'Coding' object, in which the 'code' element holds the code "CRO:0000001":

```
"realizedRole": {
   "system": "http://purl.obolibrary.org/obo/cro.owl"
   "systemVersion": 1.0,
   "code": "CRO:0000001"
   "label": "author role"
}
```

2.3 Value Sets

Value Sets are . . .

2.3.1 Contributor Role Value Set

2.3.2 Artifact Type Value Set

Important: TERMINOLOGICAL AND TYPOGRAPHIC CONVENTIONS IN THIS DOCUMENTATION

General Conventions:

- Bold font is used for emphasis of important terms or points.

Modeling Terms:

- **'Class'**: We use the term 'class' to refer to core modeling types in the specification. In other specifications this concept may be referred to as a 'type', 'resource', or 'object'. When referred to as modeling constructs in the text, class names are Capitalized (e.g. Person, Contribution)

- **'Attribute'**: We use the term 'attribute' to refer to named characteristics or relationships of classes. In other specifications this concept may be referred to as a 'slot', 'element', 'field', or 'property'. When referred to as modeling constructs in the text, attribute names are italicized and camelCased (e.g. *startDate*, *qualifiedContribution*). Contribution attributes are named in past tense because they describe what has already occurred.
- **'Data Type"**: We use the term 'data type' to refer to more fundamental and general-purpose categories of data objects. These may include 'simple' data types (e.g. string, integer, identifier) that are represented as a single literal value, or 'complex' data types that are represented as objects with their own attributes (e.g. Coding). When referred to as modeling constructs in the text, complex data types are capitalized, but simple data types are not capitalized or otherwise distinguished.
- **'Instance'**: We use the term 'instance' or 'object' to refer to instantiations of classes in the data. References to a specific instance in descriptions of data examples are written as `inline literals` in red Courier font.
- **'Value Set'**: We use the term 'value set' to refer to named collections of codes that are bound to specific attributes. These are capitalized in the text and include 'Value Set' in their name (e.g. 'Contribution Role Value Set')

Requirement Specifications:

- Requirement levels described in the specificaiton are indicated by use of the capitalized keywords "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "MAY", "RECOMMENDED", and "REQUIRED".
- These terms are to be interpreted as described in RFC 2119.

CHAPTER 3

Schema

Coming Soon . . .

3.1 JSON Schema

Coming soon . . .

3.1.1 LD-Context

Coming soon . . .

CHAPTER 4

Supporting Ontologies

4.1 CRediT taxonomy

https://www.casrai.org/credit.html

4.2 Contribution Role Ontology

https://github.com/data2health/contributor-role-ontology

CHAPTER 5

Curation Tools

The CAM Specification will provide a A Tabular File Format representation of the IM that provides a low-tech, curator friendly tool for collecting contribution metadata.

More coming soon . . .

CHAPTER 6

Code

More coming soon . . .

CHAPTER 7

Implementation Guide

Coming soon . . .

CHAPTER 8

Appendices

Coming soon . . .

8.1 I. Relationship to Existing Standards

Coming soon . . .

8.1.1 W3C PROV

8.1.2 FHIR 'Provenance' Resource

8.1.3 VIVO / openRIF