### Terminology

The following definitions of key terminology used in this document are provided both as an aid to readers unfamiliar with object-oriented modelling terminology, and to specify the precise usage of terms that are sometimes applied inconsistently across the object oriented modelling community for the purpose of this document. Where applicable, the editors have tried to consistently use terminology that is compatible with that of the Resource Description Framework (RDF)[[1]](#footnote-1), a recommendation of the World Wide Web Consortium. The editors have tried to find a language which is comprehensible to the non-computer expert and precise enough for the computer expert so that both understand the intended meaning.

|  |  |
| --- | --- |
| Class | A class is a category of items that share one or more common traitsserving as criteria to identify the items belonging to the class. These **properties** need not be explicitly formulated in logical terms, but may be described in a text (here called a **scope note**) that refers to a common conceptualisation of domain experts. The sum of these traits is called the **intension** of the class. A class may be the **domain** or **range** of none, one or more properties formally defined in a model. The formally defined properties need not be part of the intension of their domains or ranges: such properties are optional. An item that belongs to a class is called an **instance** of this class. A class is associated with an open set of real life instances, known as the **extension** of the class. Here “open” is used in the sense that it is generally beyond our capabilities to know all instances of a class in the world and indeed that the future may bring new instances about at any time (**Open World**). Therefore a class cannot be defined by enumerating its instances. A class plays a role analogous to a grammatical noun, and can be completely defined without reference to any other construct (unlike properties**,** which must have an unambiguously defined domain and range). In some contexts, the terms individual class, entity or node are used synonymously with class.  For example:  E21 Person is declared as a class in CRM. To be an instance of E21 Person may actually be determined by some DNA characteristics. CRM makes no reference to this characteristic either in the scope note nor in its relations. Still, we all know what an instance of E21 Person is through a much less specific intensional declaration. An instance of E21 Person may also have the accidental property of being a member of an instance of E74 Group. It is not necessary, however, for someone to be member of some E74 Group in order to be an instance of E21 Person. Therefore, the intensional definition should make no reference to accidental properties for identifying instances. Finally, with regards to extensional definitions, it is trivially easy to recognize that we shall never know all instances of E21 Person from the past. By common sense, there will be more instances of E21 Person in the future of which we cannot yet be aware. Therefore, the definition of the classes cannot be given extensionally. |
| subclass | A subclass is a **class** that is a specialization of another class (its **superclass**). Specialization or the IsA relationship means that:   1. all **instances** of the subclass are also instances of its superclass, 2. the **intension** of the subclass extends the intension of its superclass, i.e. its traits are more restrictive than that of its superclass and 3. the subclass inherits the definition of all of the **properties** declared for its superclass without exceptions (**strict inheritance**), in addition to having none, one or more properties of its own.   A subclass can have more than one immediate superclass and consequently inherits the properties of all of its superclasses (**multiple inheritance**). The IsA relationship or specialization between two or more classes gives rise to a structure known as a class hierarchy. The IsA relationship is transitive and may not be cyclic. In some contexts (e.g. the programming language C++) the term derived class is used synonymously with subclass.  For example:  Every instance of E21 Person IsA instance of E20 Biological Object just means that E21 Person is a subclass of E20 Biological Object.  Consider, also, every instance of E21 Person IsA instance E39 Actor. It is a property of E21 Person that instances of this class may die. This is modelled through an explicit properties, p100 is death of. This properoty, however, does not hold of other kinds of Actors, such as companies (c.f. 2).  It is the case, however, that every instance of E20 Biological Object IsA instance of E19 Physical Object. Any instance of E19 Physical Object can be moved. This is modelled in the property, p25 moved and attributed to the E29 superclass. Hence all instances of E21 Person can also be moved (c.f. 3). |
| superclass | A superclass is a **class** that is a generalization of one or more other classes (its **subclasses**), which means that it subsumes all **instances** of its subclasses, and that it can also have additional instances that do not belong to any of its subclasses. The **intension** of the superclass is less restrictive than any of its subclasses. This subsumption relationship or generalization is the inverse of the IsA relationship or specialization.  In some contexts (e.g. the programming language C++) the term parent class is used synonymously with superclass.  For example:  “E20 Biological Object subsumes E21 Person” is synonymous with “E20 Biological Object is a superclass of E21 Person”. Fewer traits are required in order to identify an item as a an instance of E20 Biological Object than to identify it as an instance of E21 Person. |
| intension | The intension of a **class** or **property** is its intended meaning. It consists of a definition of one or more common traitsshared by all **instances** of the class or property. These traits need not be explicitly formulated in logical terms, but may just be described in a text (here called a **scope note**) that refers to a conceptualisation common to domain experts. In particular the so-called **primitive** concepts, which make up most of the CRM, cannot be further reduced to other concepts by logical decomposition. |
| extension | The extension of a **class** is the set of all real life **instances** belonging to the class that fulfil the criteria of its **intension**. This set is “open” in the sense that it is generally beyond our capabilities to know all instances of a class in the world and indeed that the future may either bring about or bring to our attention new instances of a class at any time (**Open World**). An information system may at any point in time refer to some instances of a class, which form a subset of its total, in principle unknonable, extension. |
| scope note | A scope note is a textual description of the **intension** of a **class** or **property.**  Scope notes are not formal modelling constructs, but are provided to help explain the intended meaning and application of the CRM’s classes and properties. Basically, they refer to a conceptualisation common to domain experts and disambiguate between different possible interpretations. Illustrative example **instances** of classes and properties are also regularly provided in the scope notes for explanatory purposes. |
| instance | An instance of a **class** is a real world item that fulfils the criteria of the **intension** of the class. Note, that the number of **instances** declared for a class in an information system is typically less than the total in the real world. For example, the reader of this text is an instance of E21 Person, but this reader is not mentioned in all information systems describing E21 Persons.  For example:  The painting known as the “The Mona Lisa” is an instance of the class E22 Man-Made Object.  An instance of a **property** is a factual relation between an instance of the **domain** and an instance of the **range** of the property that matches the criteria of the **intension** of the property.  For example:  “The Louvre P*is current owner* *of* The Mona Lisa” is an instance of the property “P52 *is current owner of”.* |
| property | A property serves to define a relationship of a specific kind between two **classes.** The property is characterized by an **intension**, which is conveyed by a **scope note.** A property plays a role analogous to a grammatical verb, in that it must be defined with reference to both its **domain** and **range**, which are analogous to the subject and object in grammar (unlike classes, which can be defined independently). It is arbitrary, which class is selected as the domain, just as the choice between active and passive voice in grammar is arbitrary. In other words, a property can be interpreted in both directions, with two distinct, but related interpretations. Properties may themselves have properties that relate to other classes (This feature is used in this model only in order to describe dynamic subtyping of properties). Properties can also be specialized in the same manner as classes, resulting in IsA relationships between **subproperties** and their **superproperties**.  In some contexts, the terms attribute, reference, link, role or slot are used synonymously with property.  For example:  The CRM formulation “E24 Physical Man-Made Thing P62 *depicts* **E1** CRM Entity” is semantically equivalent to “E1 CRM Entity P62i *is depicted by* E24 Physical Man-Made Thing”. |
| inverse of | The inverse of a property is the reinterpretation of a **property** from **range** to **domain** without more general or more specific meaning, similar to the choice between active and passive voice in some languages. In contrast to some knowledge representation languages, such as RDF and OWL, we do not treat the inverse of a property as a property in its own right that needs an explicit declaration of being inverse of another, but an interpretation that implicitly exists for any property. The meaning of the inverse of the inverse of a property is identical to the meaning of the property itself, i.e. its primary sense of direction.  For example:  “E1 CRM Entity P62i *is depicted by* E24 Physical Man-Made Thing” is the inverse of “E24 Physical Man-Made Thing P62 *depicts* **E1** CRM Entity” |
| subproperty | A subproperty is a **property** that is a specialization of another property (its **superproperty**). In the case of properties, specialization or the specification of an IsA relationship means that:   1. all **instances** of the subproperty are also instances of its superproperty, 2. the **intension** of the subproperty extends the intension of the superproperty, i.e. its traits are more restrictive than that of its superproperty, 3. the **domain** of the subproperty is the same as the domain of its superproperty or a **subclass** of that domain, 4. the **range** of the subproperty is the same as the range of its superproperty or a subclass of that range, 5. the subproperty inherits the definition of all of the properties declared for its superproperty without exceptions (**strict inheritance**), in addition to having none, one or more properties of its own.   A subproperty can have more than one immediate superproperty and consequently inherits the properties of all of its superproperties (**multiple inheritance**). The IsA relationship or specialization between two or more properties gives rise to the structure we call a property hierarchy. The IsA relationship is transitive and may not be cyclic.  Some object-oriented programming languages, such as C++, do not contain constructs that allow for the expression of the specialization of properties as sub-properties.  Alternatively, a property may be a subproperty of the **inverse of** another property, i.e. reading the property from range to domain. In that case,   1. all instances of the subproperty are also instances of the inverse of the other property, 2. the intension of the subproperty extends the intension of the inverse of the other property, i.e. its traits are more restrictive than that of the inverse of the other property, 3. the domainof the subproperty is the same as the range of the other property or a subclass of that range, 4. the range of the subproperty is the same as the domain of the other property or a subclass of that domain, 5. the subproperty inherits the definition of all of the properties declared for the other property without exceptions (strict inheritance), in addition to having none, one or more properties of its own. The definitions of inherited properties have to be interpreted in the inverse sense of direction of the subproperty, i.e., from range to domain. |
| superproperty | A superproperty is a **property** that is a generalization of one or more other properties (its **subproperties**), which means that it subsumes all **instances** of its subproperties, and that it can also have additional instances that do not belong to any of its subproperties. The **intension** of the superproperty is less restrictive than any of its subproperties. The subsumption relationship or generalization is the inverse of the IsA relationship or specialization. A superproperty may be a generalization of the **inverse of** another property |
| domain | The domain is the **class** for which a **property** is formally defined. This means that **instances** of the property are applicable to instances of its domain class. A property must have exactly one domain, although the domain class may always contain instances for which the property is not instantiated. The domain class is analogous to the grammatical subject of the phrase for which the property is analogous to the verb. It is arbitrary, which class is selected as the domain and which as the **range**, just as the choice between active and passive voice in grammar is arbitrary. Property names in the CRM are designed to be semantically meaningful and grammatically correct when read from domain to range**.** In addition, the inverse property name, normally given in parentheses, is also designed to be semantically meaningful and grammatically correct when read from range to domain. |
| range | The range is the **class** that comprises all potential values of a **property.** That means that **instances** of the property can link only to instances of its range class. A property must have exactly one range, although the range class may always contain instances that are not the value of the property. The range class is analogous to the grammatical object of a phrase for which the property is analogous to the verb. It is arbitrary, which class is selected as **domain** and which as range, just as the choice between active and passive voice in grammar is arbitrary. Property names in the CRM are designed to be semantically meaningful and grammatically correct when read from domain to range**.** In addition the inverse property name, normally given in parentheses, is also designed to be semantically meaningful and grammatically correct when read from range to domain. |
| inheritance | Inheritance of **properties** from **superclasses** to **subclasses** means that if an item x is an **instance** of a **class** A, then   1. all properties that must hold for the instances of any of the superclasses of A must also hold for item x, and   all optional properties that may hold for the instances of any of the superclasses of A may also hold for item x. |
| strict  inheritance | Strict **inheritance** means that there are no exceptions to the inheritance of **properties** from **superclasses** to **subclasses**. For instance, some systems may declare that elephants are grey, and regard a white elephant as an exception. Under strict inheritance it would hold that: if all elephants were grey, then a white elephant could not be an elephant. Obviously not all elephants are grey. To be grey is not part of the **intension** of the concept elephant but an optional property. The CRM applies strict inheritance as a normalization principle. |
| multiple  inheritance | Multiple **inheritance** means that a **class** A may have more than one immediate **superclass**. The **extension** of a class with multiple immediate superclasses is a subset of the intersection of all extensions of its superclasses. The **intension** of a class with multiple immediate superclasses extends the intensions of all its superclasses, i.e. its traits are more restrictive than any of its superclasses. If multiple inheritance is used, the resulting “class hierarchy” is a directed graph and not a tree structure. If it is represented as an indented list, there are necessarily repetitions of the same class at different positions in the list.  For example, Person is both, an Actor and a Biological Object. |
| Multiple Instantiation | Multiple **Instantiation** is the term that describes the case that an instance of class A is also regarded as an instance of one or more other classes B1...n at the same time. When multiple instantiation is used, it has the effect that the properties of all these classes become available to describe this instance. For instance, some particular cases of destruction may also be activities (e.g.,Herostratos’ deed), but not all destructions are activities (e.g., destruction of Herculaneum). In comparison, multiple inheritance describes the case that all instances of a class A are implicitly instances of all superclasses of A, by virtue of the definition of the class A, whereas the combination of classes used for multiple instantiation is a characteristic of particular instances only. It is important to note that multiple instantiation is not allowed using combinations of disjoint classes. |
| endurant, perdurant | “The difference between enduring and perduring entities (which we shall also call *endurants* and *perdurants*) is related to their behaviour in time. Endurants are wholly present (i.e., all their proper parts are present) at any time they are present. Perdurants, on the other hand, just extend in time by accumulating different temporal parts, so that, at any time they are present, they are only partially present, in the sense that some of their proper temporal parts (e.g., their previous or future phases) may be not present. E.g., the piece of paper you are reading now is wholly present, while some temporal parts of your reading are not present any more. Philosophers say that endurants are entities that are in time, while lacking however temporal parts (so to speak, all their parts flow with them in time). Perdurants, on the other hand, are entities that happen in time, and can have temporal parts (all their parts are fixed in time).” (Gangemi et al. 2002, pp. 166-181). |
| shortcut | A shortcut is a formally defined single **property** that represents a deduction or join of a data path in the CRM. The **scope notes** of all properties characterized as shortcuts describe in words the equivalent deduction. Shortcuts are introduced for the cases where common documentation practice refers only to the deduction rather than to the fully developed path. For example, museums often only record the dimension of an object without documenting the Measurement that observed it. The CRM declares shortcuts explicitly as single properties in order to allow the user to describe cases in which he has less detailed knowledge than the full data path would need to be described. For each shortcut, the CRM contains in its schema the properties of the full data path explaining the shortcut. |
| monotonic  reasoning | Monotonic reasoning is a term from knowledge representation. A reasoning form is monotonic if an addition to the set of propositions making up the knowledge base never determines a decrement in the set of conclusions that may be derived from the knowledge base via inference rules. In practical terms, if experts enter subsequently correct statements to an information system, the system should not regard any results from those statements as invalid, when a new one is entered. The CRM is designed for monotonic reasoning and so enables conflict-free merging of huge stores of knowledge. |
| disjoint | **Classes** are disjoint if the intersection of their **extensions** is an empty set. In other words, they have no common **instances** in any possible world. |
| primitive | The term primitive as used in knowledge representation characterizes a concept that is declared and its meaning is agreed upon, but that is not defined by a logical deduction from other concepts. For example, mother may be described as a female human with child. Then mother is not a primitive concept. Event however is a primitive concept.  Most of the CRM is made up of primitive concepts. |
| Open World | The “Open World Assumption” is a term from knowledge base systems. It characterizes knowledge base systems that assume the information stored is incomplete relative to the universe of discourse they intend to describe. This incompleteness may be due to the inability of the maintainer to provide sufficient information or due to more fundamental problems of cognition in the system’s domain. Such problems are characteristic of cultural information systems. Our records about the past are necessarily incomplete. In addition, there may be items that cannot be clearly assigned to a given **class**.  In particular, absence of a certain **property** for an item described in the system does not mean that this item does not have this property. For example, if one item is described as Biological Object and another as Physical Object, this does not imply that the latter may not be a Biological Object as well. Therefore **complements** of a class with respect to a **superclass** cannot be concluded in general from an information system using the Open World Assumption. For example, one cannot list “all E19 Physical Object instances known to the system that are not E20 Biological Object instances in the real world”, but one may of course list “all items known to the system as E19 Physical Objects but that are not known to the system as E20 Biological Objects”. |
| complement | Thecomplement of a class A with respect to one of its **superclasses** B is the set of all **instances** of B that are not instances of A. Formally, it is the set-theoretic difference of the **extension** of B minus the extension of A. Compatible extensions of the CRM should not declare any **class** with the **intension** of them being the complement of one or more other classes. To do so will normally violate the desire to describe an **Open World**. For example, for all possible cases of human gender, male should not be declared as the complement of female or vice versa. What if someone is both or even of another kind? |
| query containment | Query containment is a problem from database theory: A query X contains another query Y, if for each possible population of a database the answer set to query X contains also the answer set to query Y. If query X and Y were classes, then X would be **superclass** of Y. |
| interoperability | Interoperability means the capability of different information systems to communicate some of their contents. In particular, it may mean that   1. two systems can exchange information, and/or 2. multiple systems can be accessed with a single method.   Generally, syntacticinteroperability is distinguished from **semantic** **interoperability**. Syntactic interoperability means that the information encoding of the involved systems and the access protocols are compatible, so that information can be processed as described above without error. However, this does not mean that each system processes the data in a manner consistent with the intended meaning. For example, one system may use a table called “Actor” and another one called “Agent”. With syntactic interoperability, data from both tables may only be retrieved as distinct, even though they may have exactly the same meaning. To overcome this situation, semantic interoperability has to be added. The CRM relies on existing syntactic interoperability and is concerned only with adding *semantic* *interoperability*. |
| semantic interoperability | Semantic **interoperability** means the capability of different information systems to communicate information consistent with the intended meaning. In more detail, the intended meaning encompasses   1. the data structure elements involved, 2. the terminology appearing as data and 3. the identifiers used in the data for factual items such as places, people, objects etc.   Obviously communication about data structure must be resolved first. In this case consistent communication means that data can be transferred between data structure elements with the same intended meaning or that data from elements with the same intended meaning can be merged. In practice, the different levels of generalization in different systems do not allow the achievement of this ideal. Therefore semantic interoperability is regarded as achieved if elements can be found that provide a reasonably close generalization for the transfer or merge. This problem is being studied theoretically as the **query containment** problem. The CRM is only concerned with semantic interoperability on the level of data structure elements. |
| property quantifiers | We use the term "property quantifiers" for the declaration of the allowed number of **instances** of a certain **property** that can refer to a particular instance of the **range** class or the **domain** class of that property. These declarations are ontological, i.e. they refer to the nature of the real world described and not to our current knowledge. For example, each person has exactly one father, but collected knowledge may refer to none, one or many. |
| universal | The fundamental ontological distinction between universals and particulars can be informally understood by considering their relationship with instantiation: particulars are entities that have no **instances** in any possible world; universals are entities that do have instances. **Classes** and **properties** (corresponding to predicates in a logical language) are usually considered to be universals. (after Gangemi et al. 2002, pp. 166-181). |
| Knowledge Creation Process | All knowledge contained in an information system must have been introduced into that system by some human agent, either directly or indirectly. Despite this fact, many, if not most, statements within such a system will lack specific attribution of authority. That being said, in the domain of cultural heritage, it is common practice that, for the processes of collection documentation and management, there are clearly and explicitly elaborated systems of responsibility outlining by whom and how knowledge can be added and or modified in the system. Ideally these systems are specified in institutional policy and protocol documents. Thus, it is reasonable to hold that all such statements that lack explicit authority attribution within the information system can, in fact, be read as the official view of the administrating institution of that system.    Such a position does not mean to imply that an information system represents at any particular moment a completed phase of knowledge that the institution promotes. Rather, it means to underline that, in a CH context, a managed set of data, at any state of elaboration, will in fact embody an adherence to some explicit code of standards which guarantees the validity of that data within the scope of said standards and all practical limitations. So long as the information is under active management it remains continuously open to revision and improvement as further research reveals further understanding surrounding the objects of concern.  A distinct exception to this rule is represented by information in the data set that carries with it an explicit statement of responsibility.  In CRM such statements of responsibility are expressed though knowledge creation events such as E13 Attribute Assignment and its relevant subclasses. Any information in a CRM model that is based on an explicit creation event for that piece of information, where the creator’s identity has been given, is attributed to the authority and assigned to the responsibility of the actor identified as causal in that event. For any information in the system connected to knowledge creation events that do not explicitly reference their creator, as well as any information not connected to creation events, the responsibility falls back to the institution responsible for the database/knowledge graph. That means that for information only expressed through shortcuts such as ‘P2 has type’, where no knowledge creation event has been explicitly specified, the originating creation event cannot be deduced and the responsibility for the information can never be any other body than the institution responsible for the whole information system.  In the case of an institution taking over stewardship of a database transferred into their custody, two relations of responsibility for the knowledge therein can be envisioned. If the institution accepts the dataset and undertakes to maintain and update it, then they take on responsibility for that information and become the default authority behind its statements as described above. If, on the other hand, the institution accepts the data set and stores it without change as a closed resource, then it can be considered that the default authority remains the original steward. |

### Property Quantifiers

1. Information about the Resource Description Framework (RDF) can be found at http://www.w3.org/RDF/ [↑](#footnote-ref-1)