Clinical Domain Analysis Models

Understanding UML Diagrams

V0.3

DCRI

October 23, 2007

Revision History

Revision	**Date**	**By**	**Description**
0.3	October 23, 2007	Mead Walker	Initial Draft

Table of Contents

1 Introduction

This document provides information on how to interpret the symbols within the diagrams used as a key components of clinical Domain Analysis Models.

The development of a Domain Analysis Model (DAM) is described as the initial stage of HL7 specifications development by HL7 modelers (HL7 Development Framework - HDF) The goal is to develop a representation of the problem space for a new set of specifications that can be commented on, and discussed with subject matter experts without requiring them to become expert in the jargon and minutia of the standard. The HL7 documentation notes:

> "During requirements documentation the problem domain is defined, a model of the domain (or problem space) is produced as the Domain Analysis Model (DAM) consisting of static and dynamic model artifacts. Domain, (*in*) this case, refers to the problem space for the requirements." (HDF Requirements Specification, 1.3.2)

In addition, the document states:

> "The sequence of steps to create the Domain Analysis Model (DAM) that captures the requirements are:
>
> 1. Document Business Process: Dynamic DAM (interactions) of systems of interest and Static Structure of information exchanged between them
> 2. Document Static Information Model: Static DAM and Glossary
> 3. Capture Business Rules: Relationships, Triggers, and Constraints " (HDF Requirements Specification, 3.1)

So far, clinical data model development has concentrated on the first two steps recommended by HL7.

1.1 Scope

The document covers the contents of the two diagram types used in the Tuberculosis Domain Analysis Model, the class diagram and the activity diagram. Discussion is limited to only those model components that are included in those diagrams.

While the models discussed have an abstract definition within the UML, their representation in the clinical models is affected by the conventions of Enterprise Architect which is the software tool that has been chosen for use. The diagrams below were created using Enterprise Architect.

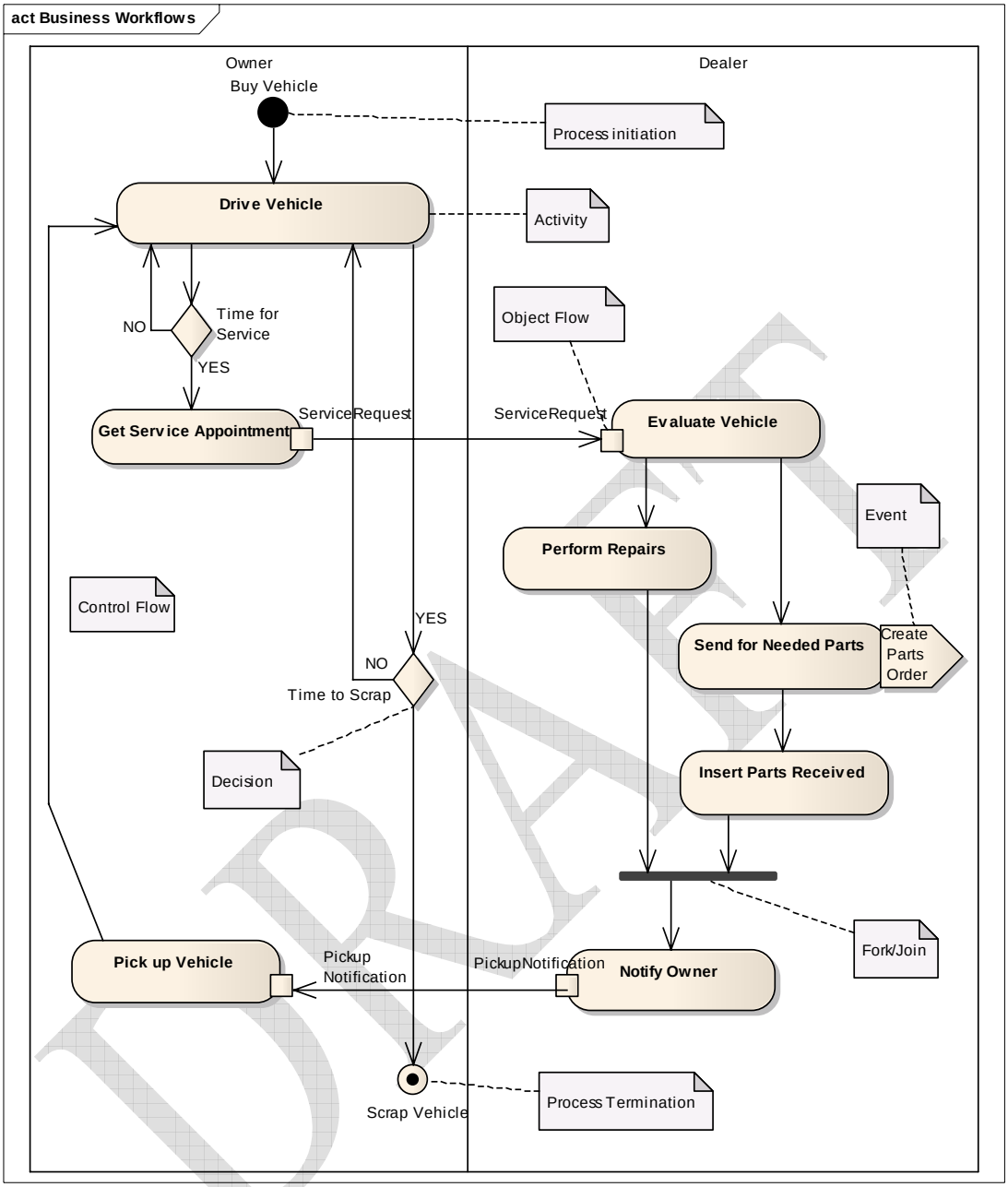
2 Activity Diagram

An activity diagram is used to show the different activities that need to be carried out to accomplish the goals of a system (or higher level activity). It also shows the organization and sequencing of those activities. More formally, it is defined as:

> "A diagram that shows the flow from activity to activity; activity diagrams address the dynamic view of a system. A special case of a state diagram in which all or most of the states are activity states and in which all or most of the transitions are triggered by completion of the activities in the source states." (The Unified Modeling Language User Guide, P. 457)

Within the clinical models, the activity diagram shows the various activities that are carried out to provide a diagnosis, or to treat a disease.

The diagram below shows a toy activity model. The goal is to expose the different components of the model in a familiar case (Please note that the model, while having familiar components, is not warranted to be accurate, that is not its purpose.)

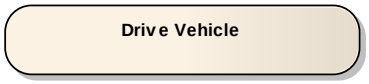


An activity model includes activities, and it shows the control flow between these activities. It may also show object flows, and document the triggering or recognition of events. The activities may be organized into swim lanes. These notions are further described below.

2.1 Activity

An activity describes a task that needs to be done. To put it another way, an activity "is an ongoing non-anatomic execution within a state machine" (The Unified Modeling Language User Guide, P. 259).

On a diagram, an activity appears as a rectangle with rounded edges.



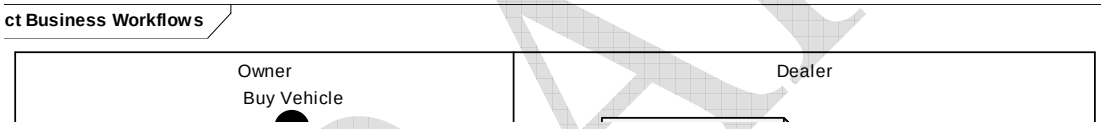
Activities are documented with:

- Name: A short text entry designating the activity. It is shown on diagrams.
- Description: A text description, hopefully not too short, that is not shown on a diagram.

2.2 Swim Lane

Swim lanes make it possible to partition the activities into groups; each group represents the organization responsible for those activities.

On a diagram, swim lanes are shown by vertically partitioning the diagram into different areas, each is a swim lane.



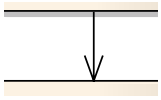
Activities are documented with:

- Name: A short text entry designating the swim lane. It is shown on diagrams.

2.3 Control Flow

A control flow indicates the sequencing of activities, and of decisions. When you review an activity diagram, it is common to trace the sequence of activities, and the consequences of decisions by following the control flows.

On a diagram, control flows are shown as arrows, with the arrowhead connecting to the destination activity, and the arrow foot connecting to the originating activity.



Currently the various features available for documenting control flows are not used within the clinical domain analysis models.

2.4 Object Flow

An object flow is a control flow that is supplemented by passing information that will be used by the receiving process. Object flows are used in the clinical

domain analysis models when we need to clearly specify that a particular block of data needs to be passed. Normally, this only happens when the activities in question sit in different swim lanes, that is to say, when they are the responsibility of different organizations. From the perspective of an organization such as Health Level 7, the presence of an object flow suggests the need for specifying a transaction definition to support the information being passed.

On a diagram, an object flow is documented using a directed arrow that extends between two rectangles, representing the object or data block. As with a control flow, the arrow head designates the receiving activity.

Object flows are documented with:

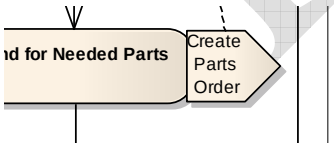
- Object Name: a short text entry that describes the data that is being passed.

2.5 Event

An event is used to indicate a block of information that needs to be passed to an organization that lies outside of the context of the domain analysis model, that is to say, that does not appear with its own swim lane. According to Enterprise Architect:

> "The UML includes two elements that are used to model events. The first element is the send event. This element models the generation of a stimulus in the system and the passing of that stimulus to other elements, either within the system or external to the system. The second element is the receive event, which is depicted as a rectangle with a recessed 'V' on the left side. This element indicates that an event occurs in the system due to some external or internal stimulus." Enterprise Architect Help Text (search for "event")

On a diagram a send event is documented as a rectangle with one elongated or pointed side (like a sign pointing to one side). A receive event, as noted above has a recess in one side.



Events are documented with:

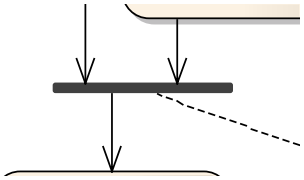
- Event Name: a short text entry that describes the data that is being passed.

As you can see, events and object flows are being used to do very similar things. The distinction is based on whether the information being passed goes to an activity within or without the system being modeled.

2.6 Fork/Join

In some cases, it is useful to provide some control over the sequencing of control flows. In particular, to indicate that multiple control flows are either generated or processed at the same time. (This structure is also known as a synchronization bar.) Forks are used to split an outgoing flow into multiple flows leading to different activities. The point of the structure is to note that each receiving activity will be initiated as a result of a single outcome from the originating activity. Conversely, a join merges multiple flows into a single flow leading to a single activity. The goal here is to make it clear when that the receiving activity starts it will make use of the result of all of the control flows from activities that took place.

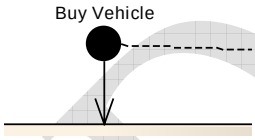
On a diagram a fork/join is indicated by a heavy horizontal or vertical bar to which control flows are linked.



Fork/Joins have no particular associated documentation.

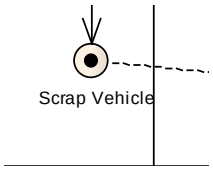
2.7 Process Initiation

Any activity needs to have a beginning. For the activity modeled within an activity diagram, this is signified by a process initiation. This is shown on a diagram as a filled in circle



2.8 Process Termination

The termination of the activity modeled by the diagram is shown by a process termination symbol. Sometimes it may have multiple ways of terminating. This is shown on the diagram as an empty circle with a dot in the middle.



3 Class Diagram

A class model is used to organize and define the static information that is relevant for a given system or activity. The formal definition states:

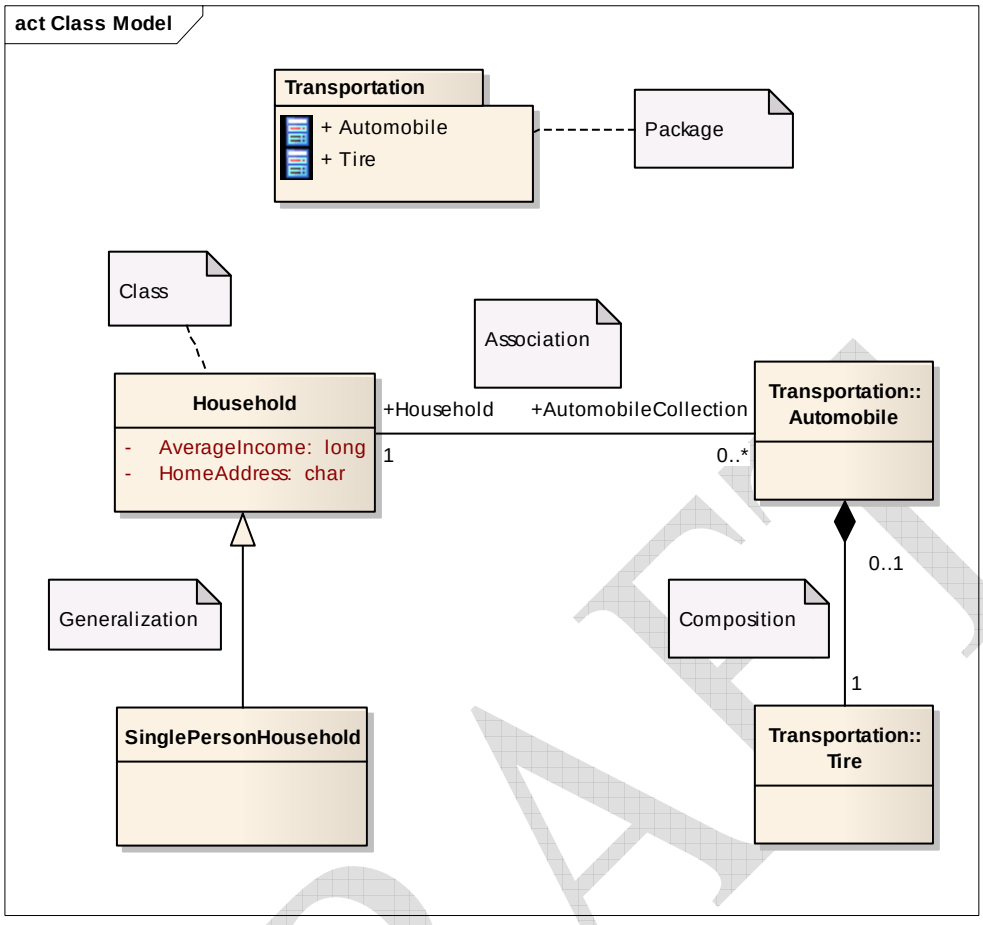
> "A diagram that shows a set of classes, interfaces, and collaborations and their relationships; class diagrams address the static design view of a system" (The Unified Modeling Language User Guide, P. 259)

The diagram below shows a toy class model. The goal is to expose the different components of the model in a familiar case (Please note that the model, while having familiar components, is not warranted to be accurate, that is not its purpose. In fact, it is not at all complete.)

It is important to note that there are different perspectives from which a class model may be developed. One perspective is that of a conceptual model. In such a model

> "you draw a diagram that represents the concepts in the domain under study. These concepts will naturally relate to the classes that implement them, but there is often no direct mapping. Indeed, a conceptual model should be drawn with little or no regard for the software that might implement it, so it can be considered language independent." (UML Distilled, Page 55)

A conceptual model can be contrasted to a specifications model that defines the interfaces for software that will implement a system, or to an implementation model that shows the design of an object oriented software application. It is important to keep in mind that the clinical domain analysis models are conceptual models.

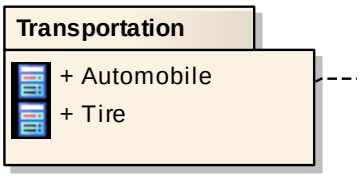


3.1 Package

A package is used to organize a model into subsections. Within the clinical domain analysis models, we normally provide a single diagram for each package, although a given diagram may contain classes from different packages. More formally, a package is defined as: “A general-purpose mechanism for organizing elements into groups.” (The Unified Modeling Language User Guide, P. 464).

Within a class diagram, the name of a class may be preceded by the name of its package, with “::” separating the two. Note that, within Enterprise Architect, both diagrams and classes may be placed in a specified package. When a class from another package is shown in a diagram, the package designation is included, but when both class and diagram are within the same package this label is omitted.

Packages are documented with a symbol resembling a file folder. Within Enterprise Architect, when you include a package on a diagram, the names of the included classes are listed within the package symbol.



Packages are documented with:

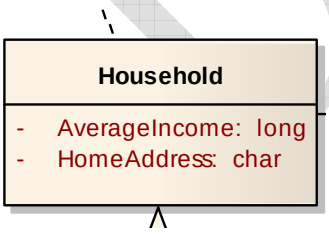
- Name: a short text entry describing the package
- Description: a text description of the package and its reason for being designated within the model. The description is not shown on diagrams.

3.2 Class

A class is used to indicate some thing or concept that is important within the context of the problem space being modeled. More formally:

> "A **class** is an abstraction of things or concepts that are subjects of interest in a given application domain. All things or concepts subsumed under a class have the same properties and are subject to and conform to the same rules. Classes are the people, places, roles, things, and events about which information is kept." (HL7 Version 3 Ballot Package, Version 3 Guide.)

Within a class diagram, classes are represented using a rectangle with two horizontal lines through it. The lines divide the class into three areas. The topmost area contains the name of the class, the middle one has the names of the attributes, and the bottom one contains methods or operations that can be performed on the class. Note, we have not documented methods within the clinical domain analysis models, and, as a result, the Enterprise Architect tool is not showing the lower section of the class that would indicate its methods.



Classes are documented with:

- Name: A short text entry designating the class. It is shown on diagrams.
- Description: A text entry that describes the class, and provides additional relevant information. It is not shown on diagrams.
- Attribute List: A listing of the attributes belonging to the class. It is shown on diagrams.

3.3 Attribute

An attribute describes a particular property of a class, it indicates a particular item of information that will be valued for an instance of the class. Formally an attribute is defined as: “a named property of a classifier that describes a range of values that instances of the property may hold” (The Unified Modeling Language User Guide, P. 458).

Within a class diagram, attribute names are shown within the class they belong to. The attributes are included in the middle section of the class representation as text entries that show the attribute name, and the assigned data type.

Attributes are documented with:

- Name: A short text entry designating the activity. It is shown on diagrams.
- Description: A text block that describes the attribute, and provides relevant other information. It is not shown on diagrams.
- Data type: An indication of the type of data which is valid for the attribute. Currently, the clinical domain analysis models are using the database oriented data types that are provided within the Enterprise Architect tool. For example, all coded attributes use the data type – char. It is shown on diagrams.
- Multiplicity: An indication of whether the attribute value may repeat. It is shown on diagrams.
- Mapping Constraint: An indication of a source that the attribute is associated with. It is not shown on diagrams. In the Tuberculosis Domain Analysis Model, this constraint documents the relationship between model attributes and the spreadsheet that was developed by the Tuberculosis Stakeholders Group.
- Vocabulary Designation: A way of indicating the list of allowable values (vocabulary) that an instance value is drawn from. This notion is only relevant for coded attributes. It is not shown on diagrams. In the clinical Domain Analysis models, we are implementing this concept using the Tagged Value feature supported by the Enterprise Architect tool.

The notion that there is a valid range of values for instances is important, and this is shown in two ways. The first is through the data type which provides a general constraint on values, e.g., integers only, dates or date/time representations. The second way is the use of vocabulary constraints. That is through defining a set of values or a vocabulary that values of the attribute must be drawn from. E.g., gender code, diagnosis code.

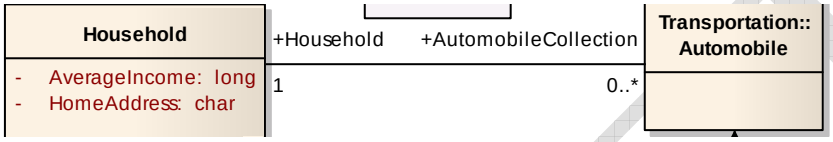
Within clinical domain analysis models, we are linking each coded attribute with a

class in a special package named Value Sets. Each class in this package represents an allowable set of codes, and each valid code is represented as an attribute within the class. This usage follows the development style mandated by the Cancer Data Standards Repository (caDSR).

3.4 Association

An association captures the relationships between instances of classes. For example, in the toy model above, there is an important relationship between households and automobiles.

Within a class diagram, associations are shown as a line connecting two classes.



Associations are documented with:

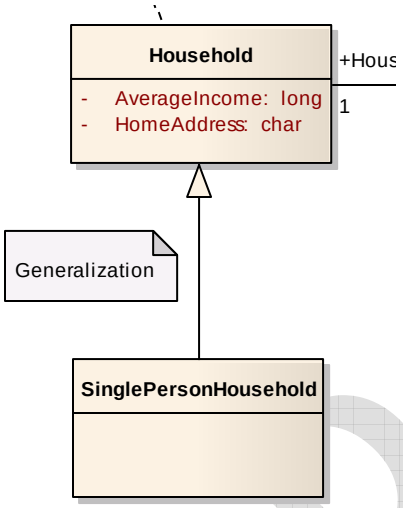
- Multiplicities (aka cardinality): These show the ways in which instances can participate in relationships with each other. Multiplicity is indicated by a symbol on each end of the association, where it joins to an associated class. It shows whether any valid instance of a class has to have an association, and whether there can be more than one. The multiplicity of an association for a class is evaluated by looking at the multiplicity on the far end of the association. For example, in the snippet above, a household has zero to many automobiles. An automobile must be associated with a single household. Multiplicity is shown on diagrams. The multiplicity symbols should be read as followed:
    - 1 (aka 1..1): one and only one. All instances of the class must have the association.
    - 0..1: zero to one. No instance of the class can have more than one of the association, but it may not have any.
    - 1..*: one to may. All instances of the class must have at least one of the associations, but it may have more.
    - 0..*: zero to many. An instance of the class may have as many or few associations as desired
- Role Names: Generically, within the UML, role is associated with the link between an association and its class (so there are two of them). The formal description states that a role is “the behavior of an entity participating in a particular context.” (The Unified Modeling Language User Guide, P. 466). However, in the clinical domain analysis models, role names echo the names of the associated class, and, if the association allows multiple instances of the class the class name has “collection” as a suffix. This usage follows the development style mandated by the Cancer Data Standards Repository (caDSR). Role names are shown on

diagrams.

3.5 Generalization Relationship

The generalization relationship (aka generalization/specialization) is used to indicate that one class is a more specialized version of another. Conceptually, it indicates that any instance of the specialized class is, by definition, an instance of its generalization. It is relevant to note that, if you consider an instance of the specialized class, e.g. single person household, it will have the attributes and associations proper to its generalization as well as those shown for its class.

Within a class diagram, generalizations are show as a line with an arrowhead that points to the more general class.

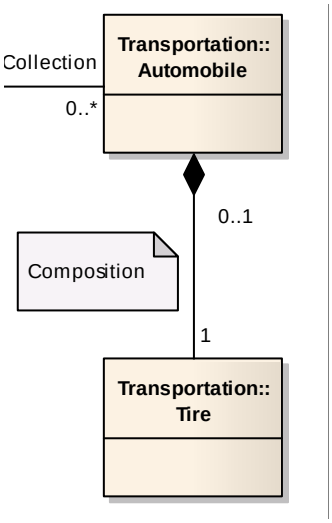


There is no particular documentation associated with a generalization.

3.6 Composition Relationship

The composition association captures the relationship between a whole and a related part. It is essentially a beefed up association with some additional semantics. (Note, within the UML, composition is a flavor of a more general notion of aggregation in which composition applies to relationships in which the part instance cannot exist without the whole instance.)

Within a class diagram, compositions are show as a line with an solid diamond shaped head that points to the class representing the whole, or containing object.



Composition Relationships are documented with:

- Multiplicities: These operate the same way as do multiplicities for associations, and are shown on diagrams.

4 References:

Booch, Grady; Rumbaugh, James; Jacobson, Ivar; The Unified Modeling Language User Guide, Addison-Wesley 1999.

Enterprise Architect; Help contents for Version 7; Sparx Systems 2007

Health Level Seven; HL7 Development Framework, Version 3 Ballot Package, August 2007.

Fowler, Martin; Scott, Kendall; UML Distilled – Applying the Standard Object Modeling Language; Addison-Wesley 1997.

National Cancer Institute, Center for Bioinformatics; caCORE Software Development Kit 3.2.1, 2007