HL7\_LM\_QUICK\_R1\_I1\_2014



HL7 Logical Model: Quality

Improvement and Clinical Knowledge (QUICK), Release 1 - US Realm

September 2014

HL7 DSTU Ballot

Sponsored by:  
Clinical Quality Information

Clinical Decision Support

Copyright © 2014 Health Level Seven International ® ALL RIGHTS RESERVED. The reproduction of this material in any form is strictly forbidden without the written permission of the publisher. HL7 and Health Level Seven are registered trademarks of Health Level Seven International. Reg. U.S. Pat & TM Off**.**

Use of this material is governed by HL7's [IP Compliance Policy](http://www.hl7.org/legal/ippolicy.cfm?ref=nav)

**Contributors**

|  |
| --- |
| Aziz Boxwala, MD, PhD Meliorix, Inc.  [aziz.boxwala@meliorix.com](mailto:aziz.boxwala@meliorix.com) |
| Cynthia L. Barton  Oklahoma Foundation for Medical Quality [cbarton@ofmq.com](mailto:cbarton@ofmq.com) |
| Floyd Eisenberg iParsimony LLC [FEisenberg@iParsimony.com](mailto:FEisenberg@iParsimony.com) |
| Marc J. Hadley The MITRE Corporation [mhadley@mitre.org](mailto:mhadley@mitre.org) |
| Kensaku Kawamoto  University of Utah  [kensaku.kawamoto@utah.edu](mailto:kensaku.kawamoto@utah.edu) |
| Mark Kramer  The MITRE Corporation  [mkramer@mitre.org](mailto:mkramer@mitre.org) |
| Rute Martins  The Joint Commission  amartinsbaptista@jointcommission.org |
| Jason Mathews The MITRE Corporation mathews@mitre.org |
| Chris Millet  National Quality Forum cmillet@qualityforum.org |
| Claude Nanjo  Cognitive Medical Systems  [cnanjo@gmail.com](mailto:cnanjo@gmail.com) |

Acknowledgments

This guide was produced as part of a combined effort with members from multiple HL7 Workgroups related to health quality. This group gratefully acknowledges input from numerous HL7 community members, as well as members of the broader health care community.

QUICK learns from and builds upon work done in several other projects and specifications including HL7 FHIR, vMR, QDM, QRDA, and CCDA. Many of the model elements and their documentation are drawn from these and other specifications.

The Clinical Statements Working Group and the Architecture Review Board are designated as Other Interested Parties for this specification.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | By Whom | Changes |
| 1 | 7/24/14 | Claude Nanjo | Initial draft for preview by WGs |
| 2 | 8/01/14 | Jason Mathews | Submission to For-Comment ballot |

For-Comment Ballot Material Overview

# 1 Purpose

This document provides guidance and an overview of the Quality Information and Clinical Knowledge (QUICK) model, which will assist with the review of the for-comment ballot material.

The QUICK logical model is currently under development. Not all concepts have been fully defined yet. For this review, please focus your attention on the core structure of QUICK and on the following leaf-level concepts in the model (located in the ***statement*** package):

* CommunicationPerformanceOccurrence
* CommunicationProposalOccurrence
* ConditionOccurrence
* DeviceUseOrderOccurrence
* DeviceUsePerformanceOccurrence
* DeviceUseProposalOccurrence
* DiagnosticImagingOrderOccurrence
* DiagnosticImagingPerformanceOccurrence
* DiagnosticImagingProposalOccurrence
* EncounterPerformanceOccurrence
* EncounterProposalOccurrence
* MedicationTreatmentOrderOccurrence
* MedicationTreatmentPerformanceOccurrence
* MedicationTreatmentProposalOccurrence
* ObservationResultGroupOccurrence
* PredictionOccurrence
* ProcedureOrderOccurrence
* ProcedurePerformanceOccurrence
* ProcedureProposalOccurrence
* SimpleObservationOccurrence

The above classes represent different forms of the following clinical concepts:

* Communications (e.g., an alert, notification, reminder, or other message)
* Conditions
* Clinical findings and observations (e.g., SimpleObservation and ResultGroup)
* Encounters
* Device usage
* Medication-related concepts, such as the administration or dispensing of a medication
* Procedures (in general)
* Imaging procedures
* Prediction (e.g., Prognoses)

# 2 Ballot Material

The for-comment ballot review material includes the following items:

* An Enterprise Architect model project (QUICK.eap). A free viewer, called Enterprise Architect Viewer (EALite.exe) is available from http://www.sparxsystems.com/products/ea/downloads.html for browsing this model file. A 30-day free trial license is also available.
* The QUICK specification, which lists classes, attributes, and diagrams that make up the model and all class and attribute definitions. This document is derived from the Enterprise Architect model file.
* Knowledge authoring documentation for all prioritized ‘leaf-level’ concepts in a JavaDoc format. This set of HTML documents also is generated from the Enterprise Architect Model file.
* For-Comment Ballot Material Overview (this guide)

# 3 Focus Areas for the Community

As part of this For-Comment Ballot, we would like to solicit comments from the community on the topics listed below *and any additional topics you would like to bring to our attention upon your review of the material*. In addition, we would greatly appreciate concrete examples and clinical use cases that we can apply to validate the model.

1. On QUICK’s modeling approach:
   1. Is the model structure (as described later in the document) sound? If not, how would you approach it differently to address clinical decision support (CDS) and clinical quality measure (CQM) use cases?
   2. Are the core semantic class definitions (Clinical Statements, Statement Topics, Modalities), their relationships, and specializations sound?
   3. What are your thoughts on the compositional vs. leaf-level representations (addressed later in the document)?
2. On concept coverage and expressivity:
   1. Are the class definitions realistic, and are the data defined in classes and attributes in QUICK available in electronic health record (EHR) systems? If QUICK includes data that are not typically included today’s EHRs or cannot be derived from today’s EHRs, is it important to include them in QUICK, to point to the future CDS and CQM systems?
   2. Does the model capture the concepts present in the Quality Data Model (QDM) and Virtual Medical Record (vMR)? Are there any gaps need to be addressed? Please recognize that not all leaf-level concepts are defined yet.
   3. Are there too many classes or concepts in QUICK? If so, what can be removed?
   4. Do prioritized classes include the necessary attributes for CDS and CQMs?
3. On the usability of the model:
   1. Is this model well suited for writing clinical expressions?
   2. Can this model be used as the basis for CDS knowledge artifacts (e.g., order sets, documentation templates, and rules)? If not, what changes in our approach or class definitions should we consider?
   3. Can this model be used to author performance measures? If not, please let us know the aspects of the model that make such authoring difficult.
4. On its ability to harmonize with other HL7 Standards—i.e., Fast Health Interoperability Resources (FHIR):
   1. Does QUICK allow for deterministic mappings to FHIR[[1]](#footnote-1)?
   2. Can (and should) QUICK and FHIR be harmonized into one model? If so, how could this be achieved and still meet the intent of both models? If not, how should the two models relate?
   3. At times, QUICK may represent classes differently from FHIR. Do you feel there may be differences between the two models that are not justified by the different intent of the models?
   4. Can QUICK also serve as a clinical model for Arden as well?
5. Ease of implementation and adoption:
   1. Is the model intuitive to knowledge authors? Are classes in the model semantically clear? Do we define attributes at the right level in the concept hierarchy?
   2. Is the purpose of each class, in particular the leaf-level concepts clear, i.e., when you must use the class? Are the purposes of classes sufficiently distinct from each other, so you know when to use one class versus another? If not, please provide examples.
   3. Will the model add to implementation complexity given the reality of today’s health IT systems? For instance, will QUICK, as it is being defined, lead to complex transformations as data moves from its persistent store to be processed by a CDS system? If so, please provide concrete examples if possible.
   4. Is QUICK easy to implement using today’s most prevalent rules engine? Does QUICK define class relationship explicitly for the most common types of relationships? Does the model support “flatter” representations more amenable to rule engine manipulations?
6. The following topics are not within the scope of the initial phase of this project, but comments related to these areas are welcome:
   1. Templating and detailed clinical models
   2. Extension mechanisms
   3. Terminology bindings
   4. Other comments

# 4 An Overview of the QUICK Model

The **Qu**ality **I**nformation and **C**linical **K**nowledge (**QUICK**) Model is a logical *“fact”* model[[2]](#footnote-2) that represents patient-centric clinical concepts[[3]](#footnote-3) for the purpose of clinical decision support (CDS) and clinical quality measures (CQM). This model aims to balance the need for a standard clinical model that includes the concepts needed for CDS and CQM with the reality of what information is currently captured and stored in today’s most prevalent clinical information systems.[[4]](#footnote-4) Current schemas and data formats often are not amenable to supporting the computability requirements necessary for quality improvement. By computability, in this domain, we mean the data is structured and codified, and furthermore those structures enable creation of compact and easily understood expressions and criteria to draw quality-related inferences. QUICK aims to address both concerns by modeling for computability while minimizing transformation costs.

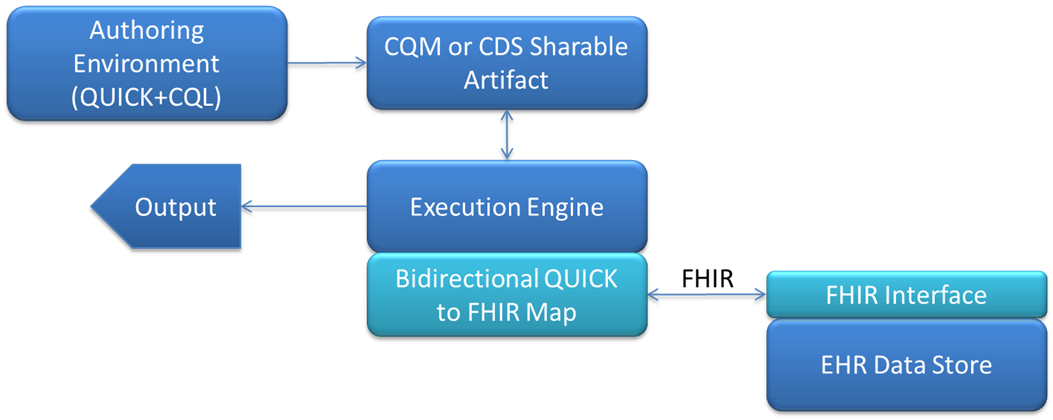
QUICK is built upon the conceptual model specified in the Quality Improvement Domain Analysis Model (QIDAM), R1. This specification was successfully balloted in HL7’s May 2014 ballot. The specification describes the use cases and requirements of a data model for quality improvement in addition to providing the conceptual data model. We refer the readers to the ballot package for QIDAM, since the publication of the specification is pending. QUICK incorporates data types from FHIR (replacing the high-level types in QIDAM), adds new attributes for identifiers, specifies cardinalities, and includes leaf-level concepts.

## 4.1 How QUICK is used in Quality Improvement

QUICK classes and attributes provide a standard way to reference information in electronic health records (EHR). By using the proposed Clinical Quality Language (CQL) with QUICK, artifact authors can compose standard, portable quality measures and decision support rules.

To execute the resulting measures and rules, QUICK’s abstractions must be mapped to concrete EHR data. This mapping can be done in one or two steps, either directly from QUICK to the underlying schemas, or indirectly using FHIR (Fast Health Interoperability Resources) as an intermediate representation. The latter approach makes most sense if the EHR offers a FHIR interface. In the latter approach, QUICK classes and attributes must be related to FHIR resources via a bidirectional mapping. Data retrieval statements in CQL involving QUICK classes and attributes are translated into FHIR read or search operations, which are then executed against the EHR data source via FHIR RESTful operations. Data is returned to the execution engine as a FHIR resource bundle. Instead of CQL processing the resulting XML or JSON directly, data returned from FHIR is translated into a list of QUICK objects for further processing as indicated by CQL statements. The translation of data references from QUICK to FHIR and back is a backend function of the CQL interpreter.

Since the current focus is on QUICK itself, not the creation and interpretation of knowledge artifacts, the details of these operations will not be covered here. However, we will provide examples of QUICK to FHIR mapping later in this paper. The feasibility of a one-to-one bidirectional mapping between FHIR and QUICK is critically important.



## 4.2 Why not use FHIR Directly?

Clinical data models and persistence schemas vary greatly across systems and implementations. These variations pose a challenge to interoperability. However, as one looks across these systems and implementations, several common representational patterns emerge. FHIR attempts to capture these common concepts and patterns as resource definitions, thus providing a common reference point that can address a number of interoperability use cases. For this reason, it is important to build a logical model for clinical quality improvement that aligns closely to FHIR.

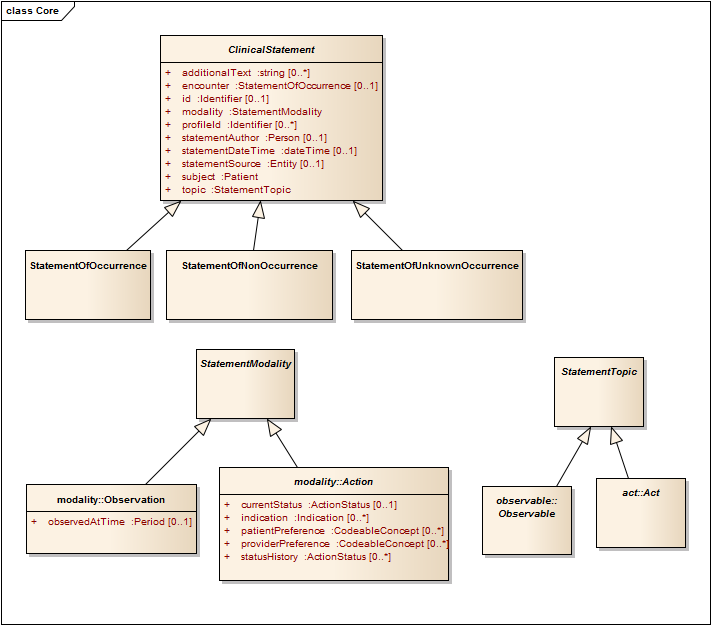
However, there are some important differences between the QUICK and FHIR models given their differing intents and modeling approaches:

1. While FHIR represents the clinical concepts typically included in today’s clinical systems, QUICK represents concepts that are important for CDS and CQM, even when those concepts may not be routinely captured (at least in a computable form) in today’s clinical systems.
2. While FHIR is essentially a “flat” model that does not define a hierarchical relationship between concepts, QUICK introduces hierarchies relevant to quality improvement that allow criteria to be written about the parent-class concepts, making the criteria more expressive.
3. FHIR attribute names are specific to the resource in which they appear. QUICK attributes can be inherited across multiple leaf node classes, assuring consistent naming (sometimes at the cost of more generic-sounding names). For example, in FHIRthe orderer of a medication is the *prescriber* and the orderer of a diagnostic study is the *orderer*, but in QUICK the attribute is always named *orderedBy*.
4. While FHIR and QUICK are both compositional models, some of their composition details differ. FHIR structures are local in scope (e.g., *Dispense* is defined differently between *MedicationPrescription* and *MedicationDispense*). QUICK favors reusable structures (e.g., a single *Dispense* concept). Moreover, the core compositional structure of the QUICK model does not exist in FHIR. Where possible, QUICK will preserve the compositional flavor of FHIR.
5. QUICK and FHIR also differ in some of their modeling approaches. To help avoid authoring errors, QUICK does not allow negation flags and modifying extensions (attributes that can modify the semantics of the class to which they belong). For example, QUICK uses a class to indicate a condition did not occur (*ConditionNonOccurrence*) rather than using an attribute (e.g. *Condition.status* = ‘refuted’).

## 4.3 Core Structure of QUICK

As alluded to in the previous section, QUICK uses a combination of a hierarchical and compositional patterns. That is, QUICK defines the components of patient data (e.g., *Order*, *Procedure*) which may be organized hierarchically (e.g., *Procedure* is a subtype of *Act* and itself has various subtypes such as *DiagnosticImaging*). These components are used to compose higher-level data elements such as an order for a procedure. This approach enables reuse of model elements and thereby consistency across the model (i.e., the order related elements for procedure and for medication have the same attributes).

The core concept in QUICK that describes patient data is the clinical statement, represented by the *ClinicalStatement* class. It represents a statement about some aspect of the patient’s health or care. Each clinical statement is organized along threeaxes: *occurrence, topic*, and *modality*, as described in the following sections. The diagram below illustrates the core semantic structure of the model.



### 4.3.1 Occurrence

Clinical statements are divided into three types:

1. A *StatementOfOccurrence* indicates that something exists or an event has occurred (e.g., a procedure was performed or a proposal for a procedure was made).
2. A *StatementOfNonOccurrence* indicates that something does not exist or that an event did not occur (e.g., the patient does not have an allergy; coumadin was not administered to this patient; or a CBC with differential was not ordered for this patient). A statement of non-occurrence is an explicit and asserted statement that something does not exist or has not happened.
3. A *StatementOfUnknownOccurrence* indicates it is unknown whether the topic of the statement has or has not occurred (e.g., it is unknown if father had diabetes; or it is unknown if patient had a tetanus vaccine in last 10 years). A statement of unknown occurrence is an explicitand asserted statement that something is unknown. As such, it is not the same having no statements of occurrence or non-occurrence.

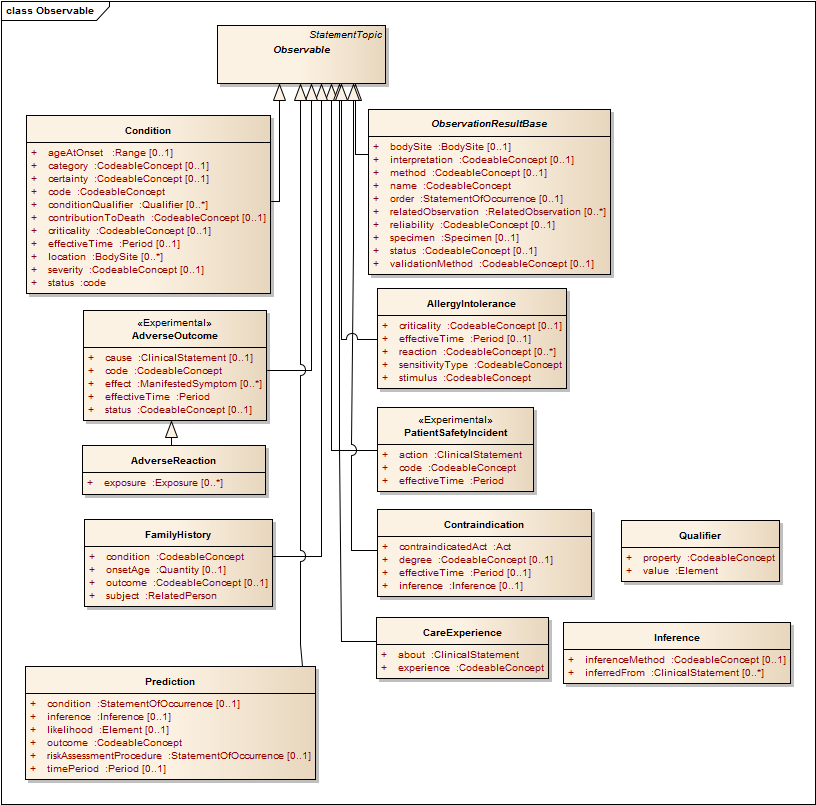
Current EHR systems mainly deal with things that have happened: orders that have been placed, actions have been carried out, observations that have been made, and conditions that exist. These correspond to statements of occurrence. Statements of non-occurrence, occasionally made in the context of conditions and diagnostics, are less frequently captured as structured data in EHR systems. Statements of unknown occurrence (e.g., no known allergy) are captured as structured data even less often. Accordingly, the prioritized classes are all statements of occurrence.

### 4.3.2 Topic

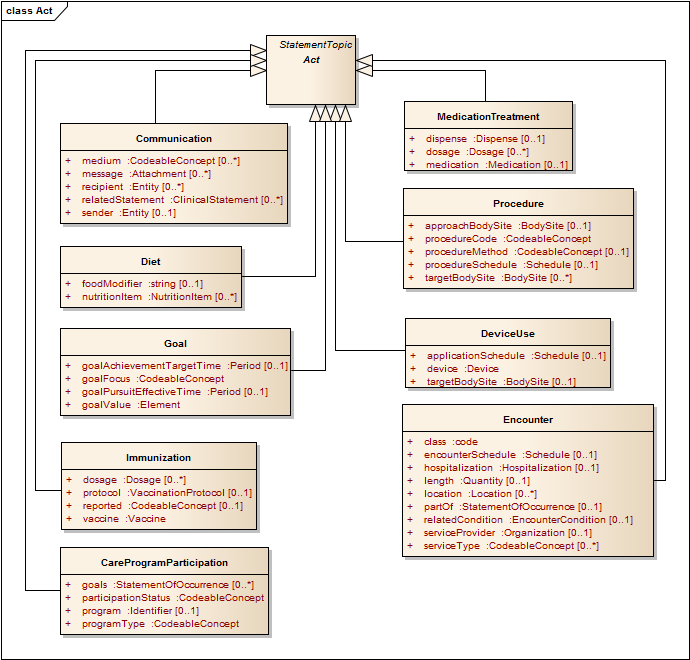
The topic of a clinical statement, represented in QUICK using *StatementTopic* class, falls into two broad categories:

1. A statement about an *Observable* phenomenon (e.g., ‘patient has diabetes’ or ‘patient has blood pressure of 130/84 mm Hg’)
2. A statement about a clinical *Act* (e.g., ‘patient received a medication’ or ‘an order for a procedure was written for the patient’)

*Observable*, the first of the two *StatementTopic* subclasses, generally represents an observable phenomenon that relates to a patient. *Observable* also has a number of specializations in QUICK, as illustrated below (not an exhaustive list). For more detailed descriptions, please refer to the specification.



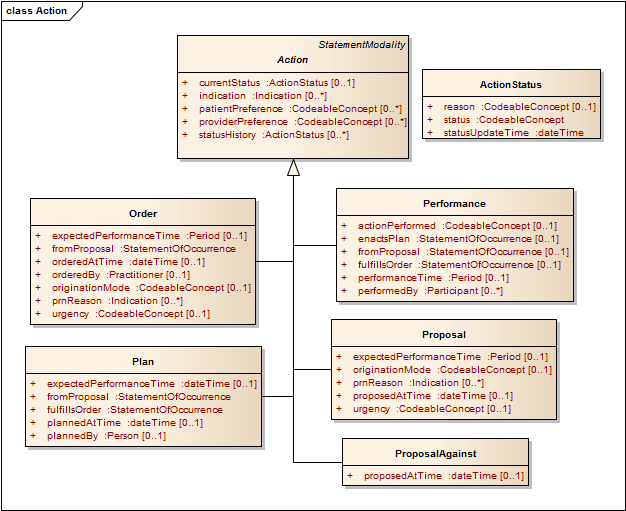
*Act*, the second of two *StatementTopic* types, represents a clinical action that can be performed. As illustrated below, it has a number of specializations in QUICK (though this is not an exhaustive list). For a more detailed description of each one, please refer to the specification.



### 4.3.3 Modality

Clinical statements also have a *modality*. The *modality axis* of a clinical statement, represented by specializations of the *StatementModality* class, describes the way the topic of the clinical statement exists, happens, or is experienced. *Statement modalities* fall into two main categories that parallel the topic hierarchy: *observation* modality and *action* modalities. The model currently defines one observation modality and five action modalities. The five action modalities are *Proposal*, *ProposalAgainst*, *Order*, *Plan*, and *Performance*.

The following diagram illustrates the specializations of the *Action* modality class in QUICK. Please refer to the specification for more detailed information about each one.



## 4.4 Two Approaches to Combining QUICK’s Three Axes

QUICK’s goal is to present clinical information in a manner that is computable and understandable to quality improvement knowledge authors. To achieve this goal, the core model supports two realizations, which take different approaches to combining the three axes of *topic*, *occurrence*, and *modality*. They are:

1. A compositional approach
2. A multi-inheritance “leaf-level” approach

These two views and their intent are described in the following paragraphs.

### 4.4.1 Compositional Approach

In the QUICK compositional approach, two attributesare added to the *ClinicalStatement* class, *topic* and *modality*. The value of the topic attribute is an instance of a subclass of *StatementTopic*, and the value of modality is an instance of a subclass of *StatementModality*. We refer to this modeling approach as a *compositional approach* because the clinical statement is built by dynamically composing the three axes. An example is a proposal for a procedure, represented as a *StatementOfOccurrence* whose topic attribute is an instance of the *Procedure* act and whose modality attribute is an instance of a *Proposal*.

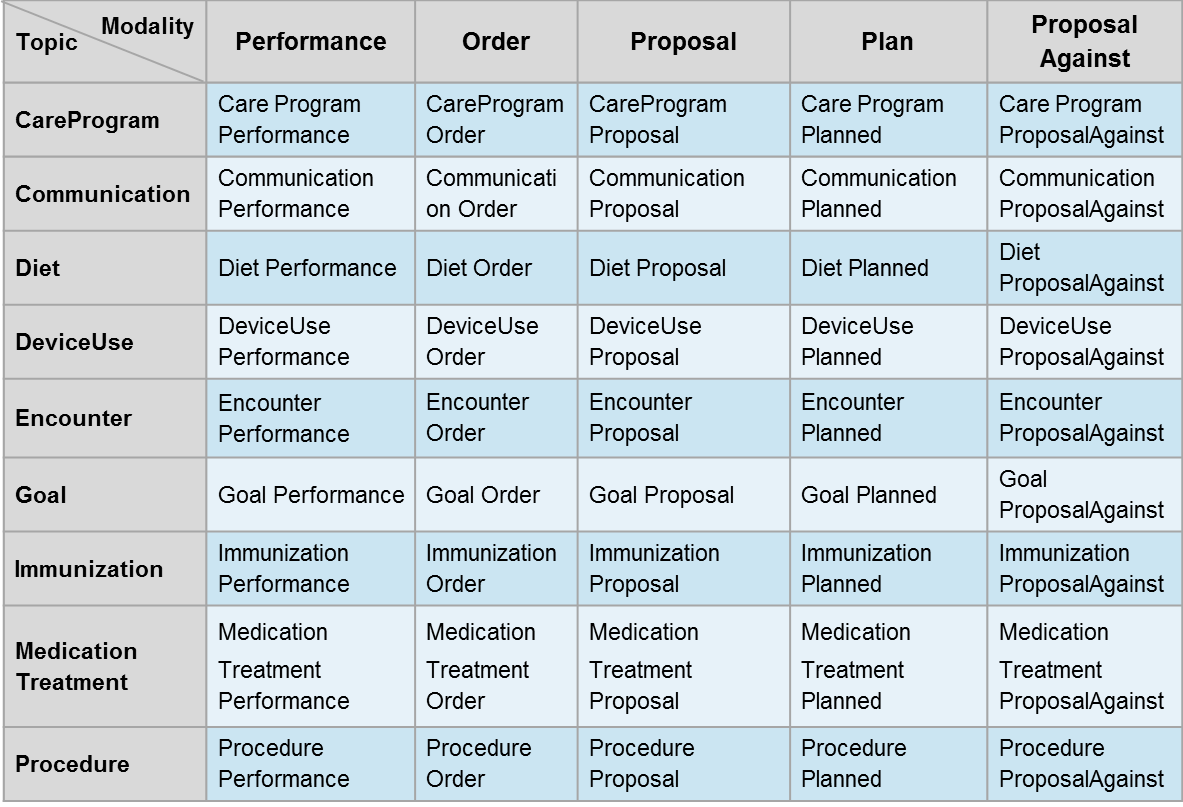
The compositional approach explicitly identifies the three classes used to compose the clinical statement. Keeping the classes apart helps to clearly identify the source of the model’s attributes. Some attributes come from the *ClinicalStatement* hierarchy, others from the *StatementTopic* hierarchy, and still others from the *StatementModality* hierarchy. No single class unites all the attributes necessary to define the clinical statement.

### 4.4.2 Multi-Inheritance “Leaf-Level” Approach

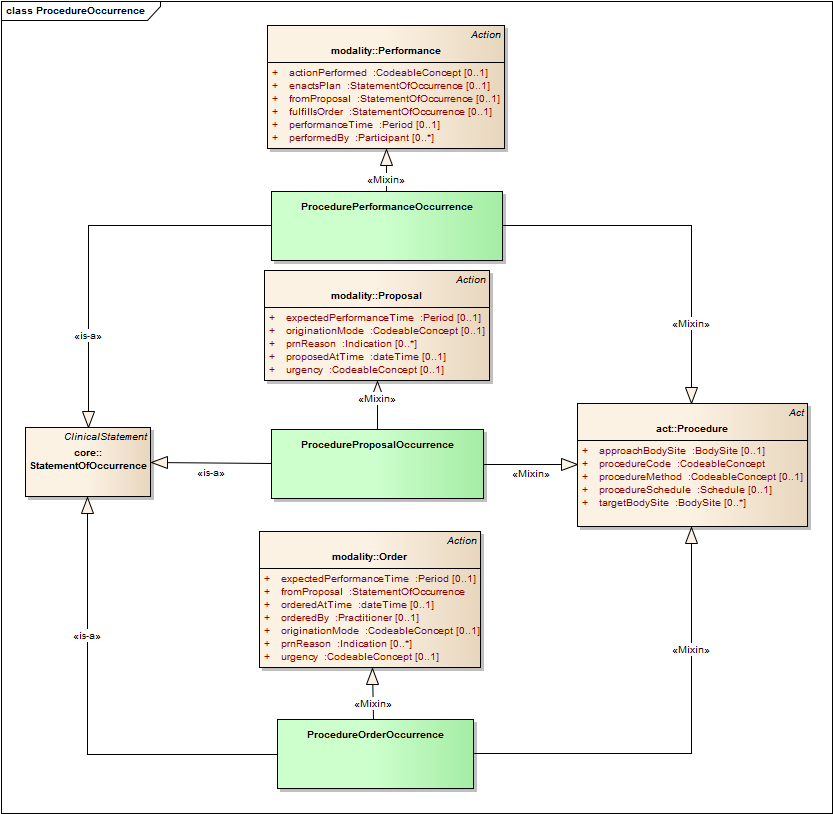
While the compositional view clearly specifies distinct and relevant concepts, it may be harder to use when creating knowledge artifacts, because the compositional structure prevents information from being accessed as simple object-property pairs. For instance, the *procedureMethod* of an imaging procedure would be addressed through a lengthy path expression, *StatementOfOccurrence.topic.procedureMethod*. Further complicating matters, the fact that *procedureMethod* even exists as an attribute of the clinical statement instance can only be determined by introspecting the contents of the *StatementOfOccurrence.topic* (in this case, an instance of the class *DiagnosticImaging*). The same is true for the attributes of *StatementOfOccurrence.modality*.

These complications motivate a more intuitive approach based on what we call “*leaf-level concepts*”. In this approach, the three different hierarchies (*ClinicalStatement*, *StatementTopic*, and *Modality*) are combined by multiple inheritance to form a leaf-level class with all the relevant attributes combined at the same level. These are “pre-coordinated” classes with fixed attributes and unique class names. The only drawback is that the number of pre-coordinated classes is combinatorial, involving the product of topics and modalities with the three occurrence types. The table below shows the combination of topic and modality. Note that not all of these combinations will be required in practical applications, as some are unlikely to occur.

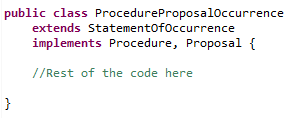
The resulting leaf-level class name can be formed by concatenation, {topic} + {modality} + {occurrence}, for example, *ProcedureProposalOccurrence*. We anticipate that more user-friendly display names (not yet defined) will be offered as alternatives in some cases, for example, *Prescription* instead of *MedicationTreatmentOrderOccurrence.* In addition, we anticipate simplifying names through defaults, for example, allowing *ProcedureProposal* to stand in for *ProcedureProposalOccurrence*, since occurrence is much more common than non-occurrence or unknown occurrence. While the concatenated names are consistent and once learned may be easier to follow, the alternative names offer the advantage of familiarity and intuitiveness. Feedback in the ballot on this issue is welcome.



The QUICK model uses a mixin approach, used by a subset of object-oriented languages, to represent QUICK leaf-level concepts.[[5]](#footnote-5) For instance, the *ProcedureProposalOccurrence* concept can be defined as a leaf-level concept that derives from *StatementOfOccurrence* using inheritance and mixes in the properties of *Proposal* and *Procedure*, as illustrated in the following diagram.



Languages such as Java and C# that support neither multiple inheritance nor mixins can achieve the same result through the use of single inheritance and interfaces, as shown here (for Java).



Concrete (pre-coordinated) leaf-level classes not only make the authoring of quality improvement knowledge artifacts more intuitive but, when implemented as a single class as illustrated above, are also more amenable for use in RETE-based production rule engines than compositional structures. Moreover, most modern programming languages support both the compositional approach and the leaf-level approach.

## 4.5 Using QUICK in Clinical Expressions

The QUICK model is intended to be used in an expression language such as the Clinical Quality Language (CQL). Only a brief overview on how QUICK is used in CQL is given here. For more information on the Clinical Quality Language, refer to the Clinical Quality Language Specification, R1.

The *retrieve* and *query* constructs within CQL are used for accessing clinical information in a knowledge artifact such as a measure or rule. The result of a retrieve is always a list of some type of clinical data. Queries enable results of retrieves to be further filtered, shaped, and extended to enable the expression of arbitrary clinical logic that can be used in knowledge artifacts. The type of data to be retrieved are specified by the axes of the ClinicalStatement as follows:

[Occurrence of Encounter, Performance]

This example retrieves all *EncounterPerformanceOccurrence* statements for a patient. *Encounter* is the topic, *Performance* is the modality. The occurrence axis does not need to be specified when the value is *Occurrence*; this is the default value.

[Encounter, Performance]

For observables, modality is not required, since there is only one modality for observations. Thus, all Conditions for patients can be retrieved by

[Condition]

The occurrence axis is required only for the non-occurrence and unknown-occurrence cases, for example:

[NonOccurrence of Condition]

[UnknownOccurrence of Condition]

A retrieve can be combined with a filter limiting the retrieve by matching on a specified value set[[6]](#footnote-6):

[Condition: “Acute Pharyngitis”]

In this example, the value set “Acute Pharyngitis” refers implicitly to *Condition.code*, which is the primary coded attribute of the class *Condition* designated by the QUICK model (these designations of primary code for topics are not yet in the current QUICK model). To support the use of filtering on code-valued attributes that may not be the primary code attribute, the retrieve expression allows the attribute name to be specified:

[Condition: severity in "Acute Severity"]

Queries allow further filtering such as

[Condition: severity in "Acute Severity"] where effectiveTime overlaps MeasurementPeriod

For more examples of query and retrieve statements, including date range filtering, see the CQL Specification and the documentation of the leaf statements in the QUICK model and specification.

To execute retrieves, the implementing system must map QUICK objects and properties to queries against EHR data. As discussed in [Section 4.1](#h.s297wrgw1crd), FHIR can be used as an intermediate representation to pass data between the CQL interpreter and the EHR data store. If so, the execution engine would translate the CQL retrieve statement into a FHIR read, search, or query.

After data is retrieved, CQL has many operations that allow the user to further filter and process the objects that are retrieved based on their relationships and properties. These operations include timing operators, mathematical, logical, text and list operations, and many more. Artifact authors use these operators to shape patient populations meeting the desired criteria.

To execute the CQL statements that follow retrieves, clinical data received from the data source must be mapped into the QUICK model classes and properties. This can be done directly by the implementing system, or via a standard FHIR mapping, if FHIR is used as intermediary. For example, CQL expressions can include the QUICK property *Condition.ageAtOnset*, for example, *Condition.ageAtOnset < 18 years*. If FHIR is used as an intermediate representation (as discussed above), condition onset can be returned[[7]](#footnote-7) either as *Condition.onsetDate* or *Condition.onsetAge*. If the former, conversion from date to age is required to enable evaluation of the expression. To make this mapping easier, the data types in QUICK are taken directly from FHIR.

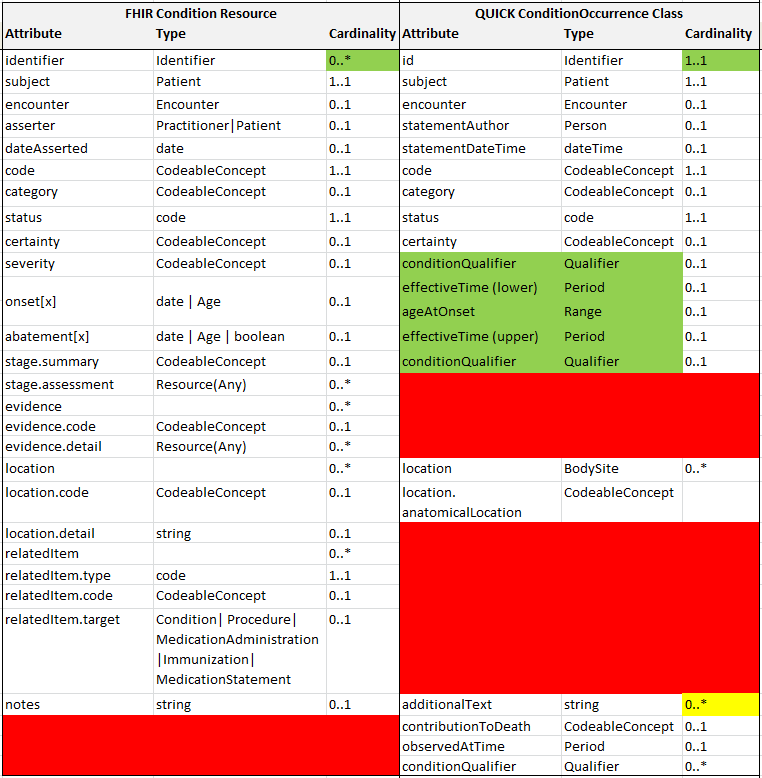
## 4.6 Extensions and Profiles

There is a requirement that QUICK can be extended by its implementers, to support needs that either are generally useful but not met by QUICK at that time, or needs that might be proprietary. The mechanisms for extension are still being developed and will be included in a future version of the specification. An important design objective is that the extension mechanism should not add complexity in the expressions using the extended classes and attributes. We expect to build upon FHIR’s extension approach so that implementation of the physical layer (i.e., patient data transport) is easier and is compatible with how CQL transforms FHIR resources into QUICK classes.

There also is a need to create profiles within QUICK. These profiles will allow specification of constraints on various elements in QUICK for specific purposes - e.g., the constraints on attributes for Encounter when describing a referral request. Currently, the QUICK model includes the profile identifiers. However, mechanisms for specifying profiles are not developed. Here too, we expect to leverage work done by FHIR.

## 4.7 Example Mapping to FHIR

In most cases, the corresponding classes in QUICK and FHIR have the same name, type, and cardinality for their attributes. As an example, the following table defines the mapping between FHIR *Condition* resource and the QUICK *ConditionOccurrence* class.

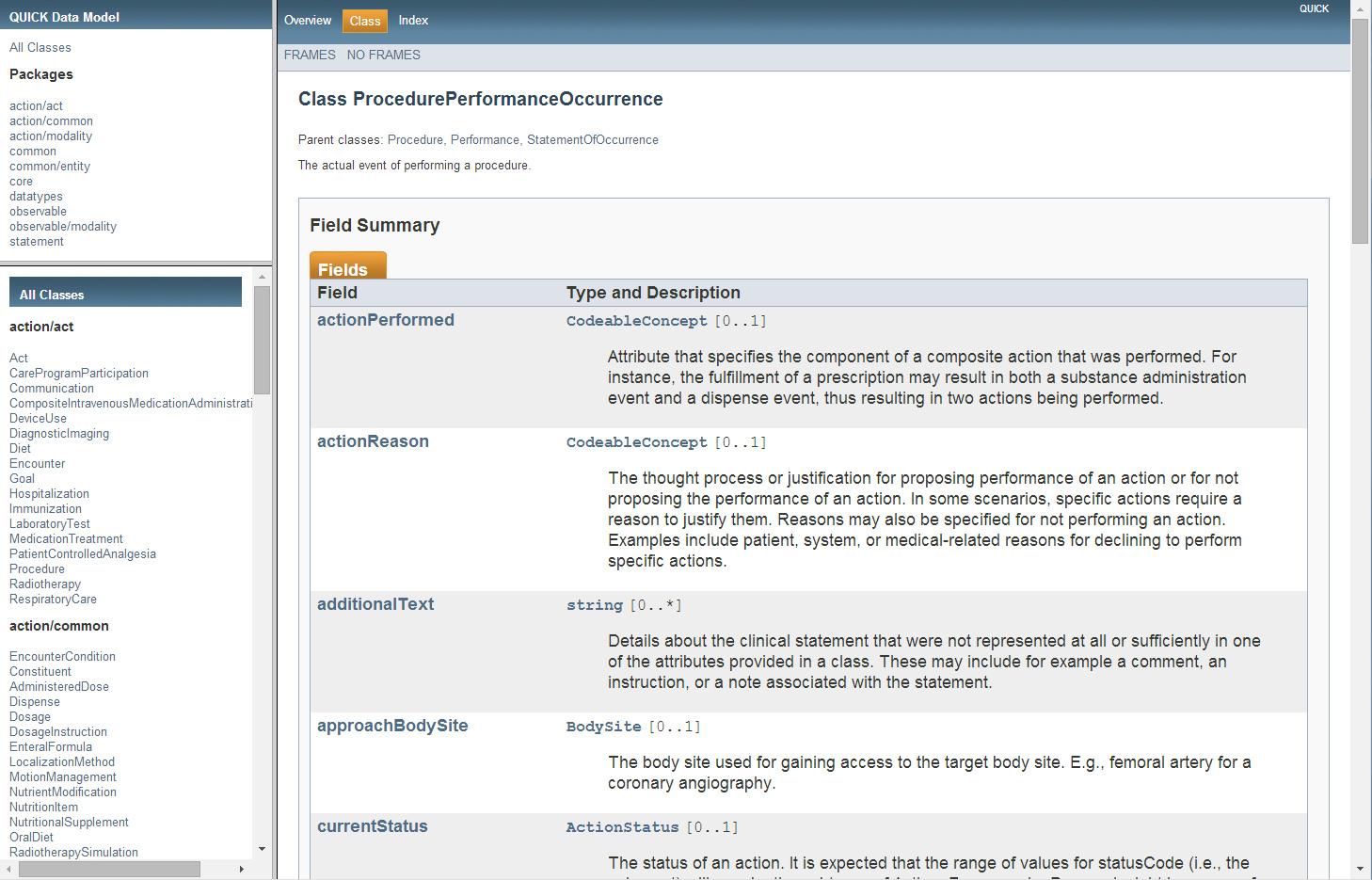


Some concepts in QUICK are inherited from higher-level concepts so the names are more generic in nature; e.g., *statementDateTime*, inherited from *ClinicalStatement*, which maps to the *Condition.dateAsserted* field. There are also some subtle differences in QUICK to make for simpler expressions and implementations, for example, *Condition.onset* property takes on a union of date and Age types, but this is simplified in QUICK with explicit *effectiveTime* and *ageAtOnset* fields of type Period and Range, respectively. Likewise, the *Condition.abatement* field is represented by the end date in the QUICK *effectiveDate* period. FHIR resources occasionally have fields that are not clearly defined enough to map, such as the *Encounter.period* and *Encounter.hospitalization.period* fields, which at first glance may be redundant. In such cases, the QUICK and FHIR teams work together to align the two together. A number of issues have been submitted to the HL7 FHIR issue tracking system to be resolved.

# 5 References for Knowledge Authors

## 5.1 HTML Pages

QUICK provides with technical documentation in auto-generated (JavaDoc-style) HTML format pages. These pages list all attributes inherited by a leaf-level concept and their cardinalities, types, and definitions. The leaf-level classes are accessed by clicking on the Statement package link on the left of the page. This information can be viewed grouped by the parent concept that contributed them or as a simple, alphabetically ordered list of all attributes with parent attribution referenced as hyperlinks.



#### 

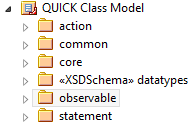
## 5.2 The QUICK Enterprise Architect Model Structure (QUICK.eap)

The QUICK model is specified in the form of a Unified Modeling Language class diagram. For readers not familiar with this modeling approach, these references provide an introduction:

1. <http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>
2. <http://www.sparxsystems.com/resources/uml2_tutorial/uml2_classdiagram.html>

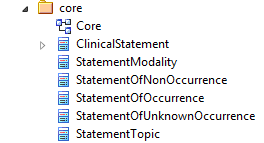
The second reference, by the developers of Enterprise Architect, uses the exact notation for diagrams as QUICK.

QUICK classes are organized into six top-level packages: ***action***, ***common***, ***core***, ***datatypes***, ***observable,*** and ***statements***,as illustrated below. These packages are described in the subsections of this section.



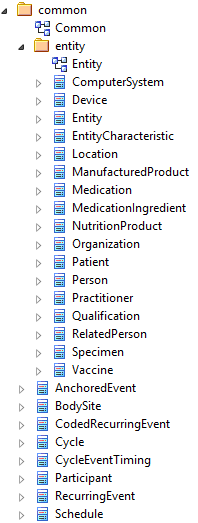
### 5.2.1 Core Package

The ***core*** package contains the core classes of the logical model—namely, the three clinical statements as well as the base *ClinicalStatement* class, the *StatementModality* abstract class, and the *StatementTopic* abstract class:



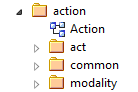
### 5.2.2 Common Package

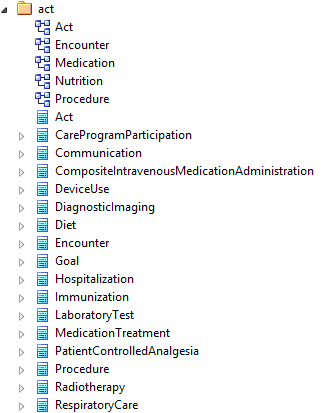
The ***common*** package contains classes shared across several other packages (e.g., *BodySite* may be referenced by acts and observables). Of these, entities such as *Patient*, *Practitioner*, *Medication*, and *Facility* represent an important subcategory of common classes. This subcategory of common classes is located in the entity package.

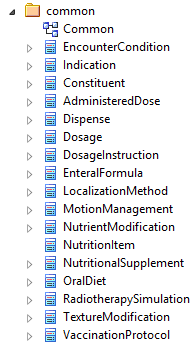


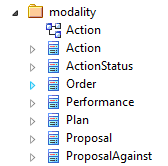
### 5.2.3 Action Package

The ***action*** package contains concepts relevant to clinical actions such as the acts and action modalities (in the ***act***and***modality*** subpackages) and ***common*** concepts referenced by action classes (e.g., *Dosage*, *Dispense*, and *VaccinationProtocol*):



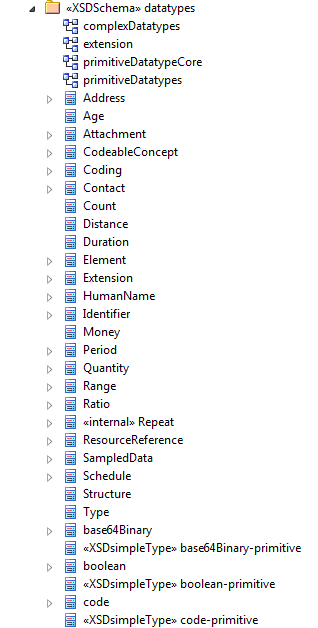






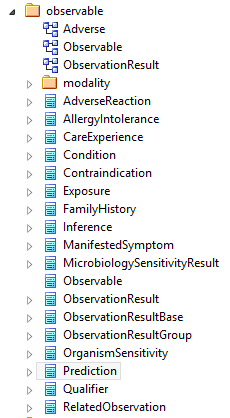
### 5.2.4 Datatypes Package

The ***datatypes*** package contains the full set of FHIR datatypes referenced in this logical model. Detailed documentation for the datatypes can be found on the FHIR web-site: http://hl7.org/implement/standards/fhir/datatypes.html



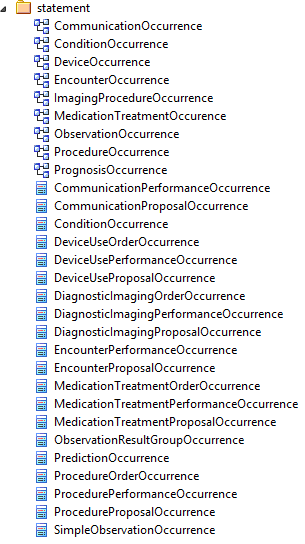
### 5.2.5 Observable Package

The ***observable*** package contains statement topics that represent observable concepts such as *Condition* and *ObservationResult.* It also contains the observation modality concept.



### 5.2.6 Statement Package

The ***statement*** package contains an initial set of the model’s leaf-level components. Currently, this package contains the set of resources identified as high-priority for our initial pilot projects and does not represent the comprehensive set of leaf-level concepts that can be represented by QUICK.



1. By deterministic, we mean whether a consistent mapping exists between a class or an attribute in both the QUICK and FHIR models. [↑](#footnote-ref-1)
2. A fact model structures domain knowledge about core concepts and relationship at the most atomic level of business knowledge. It provides the fundamental building blocks for defining or deriving more advanced forms of knowledge, such as rules and inferred knowledge. An interesting distinction between a fact model and a data model is made [here](http://www.brcommunity.com/b008a.php). [↑](#footnote-ref-2)
3. Should QUICK expand the scope of the model beyond the patient? Is such an expansion necessary for the representation of clinical quality measures? [↑](#footnote-ref-3)
4. This statement is based on the assumption that the way clinical data is currently persisted may not be optimal for computation in quality improvement. For instance, persisting unstructured clinical data may require the use of parsers before the information is useful to a rules engine. [↑](#footnote-ref-4)
5. For more information about mixins, please read this [article](http://en.wikipedia.org/wiki/Mixin). [↑](#footnote-ref-5)
6. See the CQL specification for details on how to create value set references. [↑](#footnote-ref-6)
7. A FHIR profile can be used to force the onset to be returned as an age, but there are pros and cons to forcing this conversion on the server side, and the extent to which FHIR profiles will be used to shape the data exchange has not been decided. [↑](#footnote-ref-7)