Care Provision Domain Models:
Explanation & Guidance

A_ConditionList (REPC_RM000410)
A_ConditionTracking (REPC_RM000300) –note: missing Condition Nodes
A_AllergyIntoleranceUniversal (REPC_RM000320)
A_CarePlan (REPC_RM000200)

Introduction:

However good they are, the model diagrams and formal walkthroughs in the Care Provision Domain cannot give enough guidance to engineers writing implementation guides to ensure good clinical functionality in the document, message, and service specifications created from the models. This file is intended as a reference to help engineers who are writing implementation guides, either within the HL7 publication databases or for publication outside of HL7.

As the body of HL7 document, message, and service specifications matures with the writing of more and more implementation guides, these artifacts will create additional examples that can be studied by implementation guide writers. In addition, implementation guides actually tested in production will provide “proof of concept” for these models: sometimes validating the models; sometimes justifying changes in the models. In this guide, areas of “immaturity” in the models will also be discussed.

Storyboard: Physician Manages Problem List

Purpose: To illustrate common practices in problem list management.

Precondition: Dr. Patricia Primary is viewing the EMR of Patient Everyman. Dr. Primary has completed the subjective and objective portions of the visit and is now drawing together the findings into an impression of the patient. Dr. Primary has a referral relationship to a nutritionist, Connie Chow.

Activities: Dr. Primary identifies three findings that support the diagnosis of Diabetes: frequent urination; a blood sugar of 300; and an HgA1c of 12. She then places Diabetes on the problem list and activates a care plan that includes a diabetic diet and an anti-glycemic medication for the diabetes. She identifies two findings that support the diagnosis of congestive heart failure: an abnormal chest X-ray and swelling in the feet. She also places the CHF on the problem list and activates a care plan with a low-salt diet and a diuretic. Finally, she requests a referral to a nutritionist, Connie Chow, for both the CHF and the Diabetes. The referral appointment for the diabetes dietary assessment to review the care plan is scheduled.

When the nutritionist, Connie Chow, views the referral document for the CHF and Diabetes, she navigates from the problem list to the respective diagnoses, findings, and care plans for each disease in order to see how severe the problems are. Due to the severity of the problems, she writes a recommendation back to Dr. Primary that the patient begin on a care plan with a low salt, diabetic diet with 1500mg of sodium per day and 1000 calories per day. She also requests education materials for the patient from the health library for the dietary treatments of CHF and Diabetes.

Dr. Primary reviews the recommendations from the nutritionist and adjusts the care plans to include a diet for a low salt, diabetic diet with 1500mg of sodium per day and 1000 calories per day. Dr. Primary also updates the problems on the problem list with more specific descriptions of the diabetes and the CHF that reflect current states of management of the two conditions.

Post-condition: Mr. Everyman has progress notes, findings, impressions, referral documents, care plans, and a populated problem list in the EMR that are captured in a manner that allows navigation from one related data item to the next. Specifically, the user is able to navigate from the current name of the problem on the problem list to the string of observations and treatment activities associated with management of that problem over time. Additionally, the user is able to navigate from a data item to the disease management programs supported by the data element.

Storyboard: **Physician Manages Allergy List**

Purpose: To illustrate common practices in allergy management.

Precondition: Dr. Patricia Primary is viewing the EMR of Patient Everyman. Dr. Primary has completed the subjective and objective portions of the visit and is now drawing together the findings into an impression of the patient. Dr. Primary has a referral relationship to an allergist, Dr. Richard Reaction.

Activities: Dr. Primary identifies two findings that support the diagnosis of a poison ivy allergy: an itchy rash with linear patterns of distribution; and a potential exposure while walking in the woods prior to the rash. She then places Poison Ivy Allergy on the allergy list. She identifies two findings that support the diagnosis of spring pollen allergies: a history of sinus infections in a seasonal pattern and a family medical history of allergies. She also places the Spring Pollen Allergies on the allergy list and activates a care plan with anti-histamines as needed and a referral to an allergist, Dr. Reaction, for assessment of the Spring Pollen Allergies. Penicillin with a reaction of hives is also placed on the allergy list. The referral appointment for the allergy condition assessment is scheduled.

When the allergist, Dr. Reaction, views the referral document for the Spring Pollen Allergies, he navigates from the allergy list to the respective diagnoses, findings, and care plans for each allergy in order to validate the data. He imports the data into his EMR system. He modifies and adds to the basic data in his EMR. These updates and additions include more detail on the symptoms and timelines related to the “Spring Pollen Allergies,” the family medical history of allergies, and more information on the penicillin allergy. He applies skin tests for multiple allergens, including spring pollen allergens and penicillin, and asks Mr. Everyman to return for readings on the tests. When Mr. Everyman returns, Dr. Reaction reads the tests, identifies the positive reactions to Pine Pollens, and updates the allergy list including revising the Spring Pollen Allergy and removing the penicillin allergy. Dr. Reaction returns the information to Dr. Primary in a referral report (consultants' report) document along with recommended updates to the care plans.

Dr. Primary reviews the recommendations from the allergist and adjusts the care plans to include a repeat visit by Mr. Everyman to discuss the recommendations. Dr. Primary also updates the allergies on the allergy list with more specific descriptions of the allergies and marks the penicillin allergy as obsolete.

Post-condition: Mr. Everyman has progress notes, findings, impressions, referral documents, care plans, and a populated allergy list in the EMR that are captured in a manner that allows navigation from one related data item to the next. Specifically, the user is able to navigate from the current name of the allergy on the allergy list to the string of observations and treatment activities associated with management of that allergy over time. Additionally, the user is able to navigate from a data item to the allergy management programs supported by the data element.

Note: In order to support this storyboard, HL7 has generalized allergies into the concept of "Condition." In addition, since allergy is treated as a kind of HL7 Condition, the reader should first review the HL7 approach to Conditions in this document. Allergy Lists are also a kind of HL7 Condition List. Although allergy truly is a "negative" kind of "condition, the term "Condition" is used generally in HL7 because it is less negative than "problem," i.e. management of normal pregnancy or wellness is not considered management of a "problem." In addition, assessing and optimizing the condition of a patient is considered central to effective healthcare by clinicians. Much of the following is shared by the generalized discussions under Condition List and Condition Tracking. Additional guidance on the use of the Condition List and Condition Tracking structures in the specific use cases of allergy and intolerance is given following the general discussions below.

Document Outline:

This document will begin with a discussion of the simple cases:

- An instance of Condition is managed with Control Acts
- A set of Conditions is collected into a Condition List

Next, this document will discuss the more complex cases:

- The Condition Tracking Structure in problem-oriented medical records
- The Condition Tracking Structure in disease management requests and queries

The focus of this document is to supplement the basic balloted material on the Clinical Statement Pattern and the Care Provision Domain with explanations of why the models were constructed in this manner and some guidance on how to use these models when developing implementation guides.

Lawrence Weed, M.D., the originator of the SOAP note and the Problem-oriented Medical Record, and many other authors since the 1970's have given guidance on the use of problem lists in EHR's. The reader should refer to these references available from PubMed and local medical libraries for further management advice.

The HL7 Approach to Condition:

Condition Class: The formal HL7 definition of Condition (a subclass of Observation and super-class of Case, which is a super-class of Outbreak) is:

An observable finding or state that persists over time and tends to require intervention or management and is, therefore, distinguished from an observation made at a point in time. (Author's Note: can this definition be clarified?)

Other guidance is included for reference:

Stipulated Dictionary Definition:

Condition 4: a mode or state of being *(matter is a gaseous...)* **4e:** the physical status of the body as a whole *(good ...), (poor...);* or one of its parts? usually used to indicate abnormality ;a serious heart...;a disturbed mental...

Merriam-Webster's Third New International Dictionary Unabridged, copyright 1961-2002.

Guidance: In the RIM, the term Act always represents the record of the process of doing. Observation represents a constraint on the definition of Act that takes the form, The Act of Observation where; Observation is the noun form of the infinitive to observe. In this definition, the Act of Observation is the recording of the process of observing. The multiple definitions in the stipulated Dictionary Definition emphasize that Observation may be constrained further with subclasses to narrow the definition of Observation. One of these constraints is the Act of Condition, which is a recording of the process of observing a state of being.

As in the stipulated dictionary definition, Condition refers to state, which adds the constraint of elapsed time to the definition of observation. It is not at all unusual in healthcare for one clinician to make a general observation and then a second clinician to refine that observation as a state that must be managed, e.g. by adding the observation to a problem list. The business need met by a clinician constraining an observation to the concept of condition is to introduce the idea that this observation exists over time and, therefore, might be managed. The general guidance to be given is that any observation which acts as the reason for an action should be classified as a condition (this statement might be constrained using realm-specific vocabulary constraints in order to fit local cultures). For example, the reason for a medication order should be classified as a condition rather than the more general observation, since, presumably, the medication is given to change or manage a state of the patient. A chief complaint that causes a history and physical to be performed has generated numerous assessment actions, i.e. the history and physical itself. The chief complaint would qualify as a condition on the basis that clinician energy wouldn't be spent on assessing something that was so ephemeral that it couldn't be considered a state of the patient.

In healthcare, the condition of the medical device, home of the patient, or other non-living subjects may need to be managed as well as the conditions of the patients' bodies themselves. Accordingly, condition is meant to be a general term referring to state of being of any entity in addition to the 4th Merriam-Webster definition above that refers to the condition of the body of a patient.

Other Examples: equipment repair status, device recall status, a health risk, a financial risk, public health risk, pregnancy, health maintenance, chronic illness

From a practical point of view, if a state or observation is used as a reason for intervention or management, it qualifies as a condition. For example, if a patient complains of pain and schedules an appointment, the reason for the appointment is a condition. If a physician orders a medication or a test, the reason for the order is could always be considered a condition (unless constrained by realm-specific vocabulary). A chief complaint on a history and physical is a condition of “illness” because labor is applied to evaluating the chief complaint. An annual physical exam is done to monitor whether the condition of “wellness” (sometimes called “health maintenance”) has changed.

A condition may be described by a formal code such as an ICD or SNOMED code or may be described only by a free-text word or phrase. Of course, in addition to patients and physicians defining conditions, an observer in a multidisciplinary environment may define the condition of any type of entity, including devices, facilities, or geographic areas.

It should be noted that a condition assessment, like any other observation (or any Act), may be defined, requested, scheduled, intended, promised, performed or found in any other mood in the Business Cycle Completion Track. Implementation guides reflecting the use of these moods will appear incrementally. However, the use cases for these moods are clearly illustrated in the introductory Storyboards to this document.

Of further note is that the need for condition assessment may become more specific over time. As noted in the storyboards, the primary care physician creates a general “Spring Pollen Allergy” (Act.Condition.value) that then becomes the focus of the evaluation (Act.observation.code). What was the result of an assessment for one clinician has become the starting point of an assessment for a second clinician. For the second clinician, the result of an evaluation of Spring Pollen Allergies (Act.Condition.code) is “Pine Pollen Allergy” (Act.Condition.value). Healthcare gives many examples of clinicians “drilling into” the understanding of a condition by first understanding the condition at a high level and then understanding the condition at a detailed level. Understanding how this “drill down” is facilitated in healthcare by referring patients from one clinician to another is critical to the engineer developing implementation guides from these models.

Note: There is still some immaturity in the support of these models for all relevant mood codes. Implementation guide writers may have to extend the vocabulary domains in the area of mood codes in order to support local implementations.

In Summary: However an observation was coded and represented in the Observation Class, it can like-wise be coded and represented in an instance of the Condition Class. *The creation of an instance of the Condition Class that simply duplicates the codes and values of the corresponding instance of the Observation Class represents the judgment of the provider that the observation result represents a condition that may* *require management attention.* By creating such an instance of the Condition Class, the care provider is making the statement that this state of the patient is a candidate for management.

Condition Class Explanatory Walk-through:

Condition.classCode: As a subclass of observation (OBS), “COND” inherits all the attributes and associations of Act and Observation. In addition, “COND” takes on the code vocabulary constraints specified above as well as the specific associations noted below.

Rationale: In UML, the creation of a subclass may be justified by the addition of new attributes, the additional constraints on attribute value sets, or the addition of association constraints or specifications, as well as the addition of other behavioral specifications. The use of the class “COND” in this model rather than the more general “OBS” is in recognition of these extra constraints and specifications listed for this subclass.

Guidance: “COND” is appropriate for use instead of “OBS” whenever healthcare resources are allocated to monitoring, further clarification, or treatment of the state of the patient or other entity. For example, if the “state” of the patient is used as a reason for an appointment or for ordering a test or a medication, it may be classified as a condition and then placed on a problem list; or the state of the patient simply may be classified as a condition in order to support later queries, sorts, and filters on EHR data, e.g. the reason for the appointment is the condition “ankle sprain.”

As an observation, Condition theoretically contains the same attributes related to the methodology of observation as the Observation Class itself, e.g. Condition.code, Condition.method, Condition.derivationExp, and Condition.value. In individual implementation guides, some of these attributes may be constrained out of the Condition Class. However, all of these attributes are intended to be present in these models.

Note: An area of immaturity of these models is the lack of some of these attributes in the Condition Class as well as in other Observation Classes. Implementation guide writers should extend these models where needed to includes these attributes.

Allergy Guidance: The presence of an “AllergyIntolerance” Condition is differentiated from the simple observation of an “adverse reaction.” The “adverse reaction” determination is recorded in the supporting “Causality Assessment” described later in this document. This instance of the Condition class is what appears on an allergy list.

Condition.mood: Mood will follow the patterns used for mood in other classes. In most use cases, the mood will be “EVN.” However, in Care Plans (where an assessment of the condition is planned) the mood will be “INT.” Where an opinion is requested, the mood is “RQO.” In some cases, mood could be “GOL,” e.g. where Condition.value = “diabetes, well controlled.” Goals may be placed on the problem list in some institutions. A condition may be identified as a “pre-condition” or “post-condtiion” as related to Care Plan by using the mood “EVN.CRT” or “GOL.”

Guidance: Implementation guide writers should pay careful attention to the value set for mood utilized in the implementation guide. Value sets for a vocabulary domain may be easily registered with HL7 Vocabulary TC.

Note: An area of immaturity in these models is the use of X-domains. The use of these X-domains, a deprecated HL7 practice, disallows the easy definition of value sets for specific implementation guides by implementation guide writers attempting to utilize these models.

Condition.id: Because a condition lasts over an extended period of time, the instance may be updated many times. In order to allow these updates to occur, the condition must have a unique identifier, e.g. GUID, UUID, or OID

Guidance: In order to keep track of the “state of a condition instance” at any point in time, i.e. the values of all its attributes and associations, an “instance identifier tuple” must be created with at least two fields, a “longitudinal” instance identifier and a “snapshot” instance identifier. The longitudinal identifier remains the same once the instance is created. A new snapshot identifier is assigned whenever an attribute value or association is changed. (Note: for the remainder of this document, the use of the term “identifier tuple” will refer to this combination of identifiers). For now, HL7 does not support the use of two, paired, orthogonal identifiers in a single instance of a class. In the current HL7 model, the Condition.id is the longitudinal identifier that never changes. The ControlAct.id associated with the change in the condition instance acts as the “snapshot” identifier in the second field of the tuple. (There is some discussion of providing support for a “snapshot” identifier directly on Act. If and when this is adopted, it may be used in place of the ControlAct.id.) Please read first the background material from HL7 on how Control Acts are used in HL7. Then see below in the “Versioning of Condition Tracking Structures” how instance identifiers of various components of the Condition Tracking Structure support the Storyboard.

Condition.statusCode: When the instance is in the “Active” state, the condition is still being monitored, clarified, or treated. A clinician may make the decision that a condition has resolved and no longer needs monitoring or treating. In that case, the clinician may change the status of the instance to “Completed.”

Guidance: The statusCode must be used carefully along with versioning of the instance in order to truly represent the opinion of the author regarding the condition of the patient. See below in the “Versioning of Condition Tracking Structures” how states of instances of various components of the Condition Tracking Structure support the Storyboard. Although there is inevitably some overlap in semantics between the technical status of the instance and the clinical status of the Condition, the implementation guides should be cautious about ascribing too much clinical meaning to the status of the instance. StatusCode should be more about managing the instance, e.g. managing the record in a “virtual EHR,” rather than making clinical inferences from the statusCode. For example, a statusCode of “obsolete” should mean that the instance is no longer relevant for clinical decision support queries. A status code of “complete” should mean that the instance is still relevant for decision support queries, but that it isn’t currently considered a candidate for management, i.e. the condition has resolved. A condition that is “in remission” normally would still be considered “active.”

Allergy Guidance: The statusCode in an instance of an AllergyIntolerance (Condition) will be: “Active” if the allergy or intolerance is still present in the patient or “in remission” after being desensitized; “Complete” if the allergy has disappeared, as when pediatric allergies disappear in adulthood; “Obsolete” if the allergy was recorded and found later not to exist (reasonable clinical decisions may have been based on the record); and “Nullified” if the record was truly in error, such as applied the wrong patient.

Condition.code: As in other Acts, “code” adds additional definition about the action method not present in the classCode definition. However, unlike classCode, the value of the “code” attribute may be updated over the life of an instance. At the level of the RMIM, the “code” attribute is not required to be valued with either a flavor of null or a code. However, an implementation guide may require the “code” attribute to be valued with a specific value set and not allow flavors of null (marked as “mandatory” in the models rather than “required.”

Guidance: The “value set” for the attribute Condition.code is going to be determined by clinicians in any given realm by applying a value set to the proposed vocabulary domain (ConditionType). What is in this value set is going to be determined by what actions need to be requested, scheduled, promised, and performed related to identifying and managing conditions.

Allergy Guidance: In decision support systems for pharmaceutical or dietary interactions, there may be a need to filter the list of “agents” or “allergens” that is actually used within the decision support system. For example, if the clinician is only concerned about true allergens, the “agents” that cause intolerance reactions should not be sent to the decision support system. Therefore, implementation guides for dietary or pharmacy purposes may desire to register a vocabulary set for “ObservationIntoleranceType (OINT)” that supports filtering the agents found in Condition.value for type, e.g. drug allergy, drug intolerance, food allergy, food intolerance, environmental allergy, environmental intolerances, etc.

Note: It may be that a pharmacy department wishes to have a combination of food and medication allergies that relates to pharmacy drug interactions, e.g. egg allergies for immunization alerts. Dietary departments may find some cross-reaction to medication ingredients, e.g. color dyes. These “department focused” lists may be created with the “Custondian (CST)” participation.

Condition.value: As in other HL7 Observations, Condition.value contains the result of the search or determination specified by the method for the action, whether that method is described in the (possibly combined) attributes of classCode, code, methodCode, derivationExp or a qualifier on value itself. In this case, the “Current Name of the Condition” is found in Condition.value. The semantic to be understood by the receiver is that Condition.value is the name of the condition the author feels is a candidate for management. The attribute Observation.Condition.value is required to be valued in Event Mood and several other moods, e.g. GOL and EVN.CRT, since it is assumed that all applications will be able to support either a string or coded value representing the name of the condition.

Guidance: Best practice is that Condition.value contains the entire expression of the concept needed to communicate the term code or codes for the name of the condition. There should be no need to examine Condition.code for additional clarification on the meaning of the condition being managed. Therefore, at the RMIM level, the attribute value for Observation.Condition.code is only required (meaning that the attribute must be valued with either a flavor of null or with a specific string), although the valued attribute may be declared manadatory in a given implementation guide (which means a Flavor of Null is not allowed and at least a string representing the condition must be present). The attribute value for Condition.value shall be present, at least as a string, and not be valued with a Flavor of Null in Event Mood and should be semantically complete in itself. However, it is recognized that due to cultural differences in how language is expressed and how healthcare is applied, specific implementation guides may specify alternative practices as acceptable or desirable. The important issue is whether transforms are possible between the specifics of various implementation guides without too much loss of basic information regarding the condition.

Guidance: Condition.value contains the agent, including allergens, to which the patient is sensitive. Much discussion includes the scope of this list of agents, e.g. does this include NDC numbers (which may be classified as manufactured entities)? The implementation guides should specify the value sets allowed. The allergy model includes the participation “Causal Agent,” which allows the clinician to point to the description if the “atoms and molecules” actually administered to a patient. This optional participation must be addressed in the implementation guide when discussing the list of allowed agents in Condition.value.

Participations on Condition: These are well-documented in the Clinical Statement pattern with the exception that the subject of a condition has been extended in the Care Provision Domain to include any living or non-living entity.

ActRelationships on Condition:

Subject: An instance of Condition is the subject of an instance of the Control Act, which identifies the detailed information related to changes in the attribute values or in associations to the instance of Condition. These update functions using Control Act follow the same pattern used for other kinds of classes and are fully explained in the HL7 documents on Update Mode and the Dynamic Model.

Guidance: This is an important relationship in simple use cases since most conditions in a patient are normally re-evaluated and updated at intervals by clinicians. The combination (tuple) of the id for the Condition and the id for the ControlAct represents the state of the condition at a given point in time.

Name: An ActRelationship of type “NAME” where the source is a ConditionNode and the target is the Condition.

Guidance: This is not a required association in simple use cases. This association from ConditionNode is further explained below under ConditionNode. Briefly, the currently known name of the condition is expressed as the current value of an instance of the Condition class. Because the name of a condition may be updated by ControlActs over the lifetime of an instance of Condition, the name of the condition at any point in time may be identified using both the Condition.id and the ControlAct.id as noted above under Condition.id. First read background material from HL7 on how Control Acts are used in HL7. Then see below in the “Versioning of Condition Tracking Structures” how instance identifiers of various components of the

Condition Tracking Structure support the Storyboard. This ActRelationship of type “NAME” refers to this tuple of Condition and ControlAct instance identifiers that uniquely identifies the current “Name” of the condition at a given point in time.

Component: A recursive ActRelationship of type “COMP” may be used to identify “sub-conditions,” e.g. a diabetic ulcer as a sub-condition of diabetes. These are usually used to identify complications of diseases. Some problem list management systems support a hierarchy of conditions.

Guidance: This association is included in the model for “completeness.” It is not considered to be critical in the short-term in most communications for simple uses cases.

Revise: A recursive ActRelationship of type “Revise” shall be used when, for many reasons, the Condition must be updated, either with a new value or a new status code or a change in other attributes or associations. “Revise” leaves the instance identifier for the source Condition instance unchanged from the target Condition instance. At the current time, the instance identifier of the Control Act is paired with the instance identifier of the Condition in order to describe the unique combination of attribute values and associations on the instance of Condition. In the future, the “snapshot identifier” of the Condition instance may also be used to represent this unique combination of values and associations at a given time in the lifecycle of the Condition instance. As noted above, instances of the Condition class are updated using an instance of the Control Act. These update functions using Control Act follow the same pattern used for other kinds of classes and are fully explained in the HL7 documents on Update Mode and the Dynamic Model.

Replace: A recursive ActRelationship of type “RPLC” shall be used when, for many reasons, one or more instances that represent the same condition must be replaced by a new instance (new instance identifier) and the original instance(s), with their unique instance identifiers, are transitioned to the state of “obsolete.” That means the original instances (and instance identifiers) of Condition are still kept in the system for historical purposes, but no longer shall have values changed or new associations applied. “RPLC” SHALL NOT be used instead of a Revise update to an existing instance with a ControlAct in order to perform routine updates of Condition.

Sequel: A recursive ActRelationship of type “SEQL” shall be used when modeling repetitive conditions such as pregnancy. This relationship is intended to communicate that the second instance of the Condition Class really represents a new instance of a Condition with the same name. In pregnancy, the understanding of the pregnant condition is altered by the fact that the woman had a previous pregnancy.

Guidance: Repetitive Conditions, as opposed to Conditions in Remission, will have unique instance identifiers. Instead of being marked “obsolete” as occurs in the Replace ActRelationship, the previous Act remains “complete.” A chronic condition with exacerbations might be modeled as an over-arching condition with the sub-conditions representing the exacerbations. In this case, the exacerbations would be linked with sequel relationships to each other, and each exacerbation identified as a component sub-condition of the chronic condition.

Note: The “Graphic” section at the end of the POMR section below, demonstrates the evolution of the Condition instance during update workflows. The “Graphic” won’t be reproduced in this section, since it is better to read the whole document and then see how the simple use case represents a “constraint” of the more complicated structures.

Guidance: This is an important relationship in simple use cases since most conditions in a patient are normally re-evaluated and updated at intervals by clinicians. The combination (tuple) of the id for the Condition and the id for the ControlAct represents the state of the condition at a given point in time. In the next section on collecting Conditions into a Condition List, the assumption is made is that the instances of Conditions on the List will continue to be updated, even though the list itself is not changed.

General Note on Conditions: Much ado tends to be made about systems that do not define conditions as such; many systems only define observations. However, any system that defines reasons for orders or develops a problem list is defining conditions in its system by using alternative semantic structures. Therefore, where “reason,” “problem list” or “allergy list” is part of the semantic of the observation, then de facto, a transform may occur between the system and the HL7 communication format utilizing the Condition class; *the observation must be sub-classed as a condition when used in HL7 ConditionList communications.* However, as noted below, business needs and associated messages may be identified in the future that support alternative practices in the development of problem lists.

The HL7 Approach to Condition Lists:

Conditions within a Condition List: An instance of the Condition Class in Event mood is the building block of the Condition List Structure. A Condition List shall be built using the WorkingList (LIST) class to collect instances of Condition via the Component (COMP) ActRelationship. Updates to an instance of a Condition may occur independently while the instance is a component of the List class. An instance of Condition may be a component of multiple LIST classes.

Problem Lists and Condition Lists: A condition list shall be constrained to a problem list by specifying "Problem List" in the attribute WorkingList.code. Organizations might like to use other types of management-oriented lists besides "problem lists." In this case, the organization would use a different value for ConditionList.code. The value set for this attribute is CWE.

It is expected that local organizations might define their own codes for different kinds of lists for internal management purposes. Vocabulary publishers have defined certain kinds of management lists as well. How organizations apply "problem lists" and other management lists to their organizational policies and procedures is highly individualized and is beyond the scope of HL7 to standardize.

Allergy Guidance: An allergy list is another kind of Condition List and is specified with a value in WorkingList.code.

Note on Scope: In some problem list applications, the “List” is allowed to contain any kind of HL7 Act as opposed to “just conditions” as described in this Condition List specification. Communications of general lists of Acts represent a different list management scenario than described in this model. In the interests of solving the specific scenario of communicating “conditions,” the general list variant is not addressed. However, it is possible that the general scenario of list management will be pursued at a later time. It is also possible that these problem list applications may be able to develop transforms and populate messages and documents that use this Condition List structure. It is beyond the scope of this discussion to examine these options at this time.

The intention in this model is that any "instance of Condition" may be added to a “Condition List instance” subject only to the constraints described in the discussion of the Condition Class above. One question that arises in regards to Problem Lists is "Given that Condition is a subclass of Observation, how does one place a past history of a surgery on the Problem List?" Weed answered that question in his descriptions of the Problem-oriented Medical Record. In his examples, he used the term "status post appendectomy" to turn the expression of a procedure into a noun that reflects the result of an observation that a procedure was performed in the past. It would be just as correct to use the observation result, "Past Medical History of Appendectomy" on the Problem List.

WorkingList Explanatory Walk-through:

WorkingList.classCode: As a subclass of Act, “LIST” inherits all the attributes and associations of Act. Its definition emphasizes the dynamic nature of the list:

“Working list collects a dynamic list of individual instances of Act via ActRelationship which reflects the need of an individual worker, team of workers, or an organization to manage lists of acts for many different clinical and administrative reasons. Examples of working lists include problem lists, goal lists, allergy lists, and to-do lists. “

Guidance: Many classes in the RIM act as collectors of other Acts. “LIST” is expected to support the methods expected of dynamic lists in general, e.g. adding and deleting list items (components of the list); reordering of the list components; creating and obsoleting lists themselves along with all the usual attribution associated with authors, list owners, etc.

WorkingList.mood: Mood is generally Event (EVN) in this model. If used, RQO would represent a request to construct a list, DEF would indicate types and patterns of lists that could exist, etc.

WorkingList.id: Because a condition list lasts over an extended period of time, the instance may be updated many times. In order to allow these updates to occur, this identifier must be present.

Guidance: In order to keep track of the “state of the list” at any point in time, i.e. the values of all its attributes and associations, an “instance identifier tuple” must be created with at least two fields, a “longitudinal” instance identifier and a “snapshot” instance identifier. The longitudinal identifier remains the same once the instance is created. A new snapshot identifier is assigned whenever an attribute value or association is changed. (Note: for the remainder of this document, the use of the term “identifier tuple” will refer to this combination of identifiers). For now, HL7 does not support the use of two, paired, orthogonal identifiers in a single instance of a class. In the current HL7 model, the WorkingList.id is the longitudinal identifier that never changes. The Control Act.id associated with the change in the condition list instance acts as the “snapshot” identifier in the second field of the tuple. (There is some discussion of providing support for a “snapshot” identifier directly on Act. If and when this is adopted, it may be used in place of the ControlAct.id.) Please read first the background material from HL7 on how Control Acts are used in HL7. Then see below in the “Versioning of Condition Tracking Structures” how instance identifiers of various components of the Condition Tracking Structure support the Storyboard.

WorkingList.statusCode: When the instance is in the “Active” state, the list is still being edited. When these activities are halted, the instance would transition to one of the other appropriate states, e.g. Obsolete (if replaced) or Complete (if no longer maintained).

Guidance: The statusCode must be used carefully along with versioning of the instance in order to truly represent the opinion of the author regarding the contents and status of the list. See below in the “Versioning of Condition Tracking Structures” how states of instances of various components of the Condition Tracking Structure support the Storyboard.

WorkingList.title: Title, as in other acts, represents a human-readable string of formatted text.

Guidance: If this attribute is not used or populated in an implementation guide for Condition Lists, then the human-readable title for the list may be taken from WorkingList.code.

WorkingList.code: As in other Acts, “code” adds additional definition about the action method not present in the classCode definition. However, unlike classCode, the value of the “code” attribute may be updated over the life of an instance. This WorkingList.code attribute is required to be valued in this model. In a problem list, this attribute would be valued with a code for problem list. In an allergy list, this attribute would be valued with a code for allergy list.

Participations on WorkingList: These are the same participations found on any clinical statement in the Care Provision Domain except that an additional participation “List Owner” may be defined.

Guidance: “List Owner” is intended to support the concept that some lists are managed at different levels within certain organizations. For example, a list might be managed by a group of editors from a Department, e.g. the pharmacists’ medication list. Some organizations require that a specific clinician has the final authority to edit the problem list and resolve conflicts between clinicians. “To Do” lists are often “personal” in nature. “List Owner” is intended to support these use cases. Different instances of Condition List often range from “personal to organizational” in scope.

ActRelationships on Condition:

Subject: An instance of LIST is the subject of an instance of the Control Act, which identifies the detailed information related to changes in the attribute values or in associations to the instance of LIST. These update functions using Control Act follow the same pattern used for other kinds of classes and are fully explained in the HL7 documents on Update Mode and the Dynamic Model.

Guidance: This is an important relationship in simple use cases since most lists for a patient are normally re-evaluated and updated at intervals by clinicians. The combination (tuple) of the id for the Condition and the id for the ControlAct represents the state of the list at a given point in time, i.e. which items were on the list, in which order were the items are listed, the statusCode of the list, etc.

Component: A recursive ActRelationship of type “COMP” may be used to identify “sub-lists” within a “list of lists.”

Guidance: This association is included in the model for “completeness.” It is not considered to be critical in the short-term in most communications for simple uses cases.

Revise: A recursive ActRelationship of type “Revise” shall be used when, for many reasons, the LIST must be updated, either with a new value or a new status code or a change in other attributes or associations. “Revise” leaves the instance identifier for the source LIST instance unchanged from the target LIST instance. At the current time, the instance identifier of the Control Act is paired with the instance identifier of the LIST in order to describe the unique combination of attribute values and associations on the instance of LIST. In the future, the “snapshot identifier” of the LIST instance may also be used to represent this unique combination of values and associations at a given time in the lifecycle of the LIST instance. As noted above, instances of the LIST class are updated using an instance of the Control Act. These update functions using Control Act follow the same pattern used for other kinds of classes and are fully explained in the HL7 documents on Update Mode and the Dynamic Model.

Replace: A recursive ActRelationship of type “RPLC” shall be used when, for many reasons, one or more instances that represent the same LIST must be replaced by a new instance (new instance identifier) and the original instance(s), with their unique instance identifiers, are transitioned to the state of “obsolete.” That means the original instances (and instance identifiers) of LIST are still kept in the system for historical purposes, but no longer shall have values changed or new associations applied. “RPLC” SHALL NOT be used instead of a Revise update to an existing instance with a ControlAct in order to perform routine updates of LIST.

Guidance: This “RPLC” relationship is used to merge lists for purposes of conflict resolution by a user, e.g. two clinicians or systems created two instances representing the same list and one list instance needs to be retired. Or this relationship may also be used when the clinician performs splits or joins on the list in the normal course of management, e.g. the clinician realizes that what was once thought to be two lists really represents one list, etc. This Replace ActRelationship SHALL NOT be used instead of a Revise to update to an existing instance with a ControlAct. Although it is recognized that this use of a new instance identifier every time a concept is updated could be used to model lists, HL7 has chosen the alternative model of updating an instance with a single id.

Guidance: This is an important relationship in simple use cases since most lists in a patient record are normally re-evaluated and updated at intervals by clinicians. The combination (tuple) of the id for the WorkingList and the id for the ControlAct represents the state of the WorkingList at a given point in time. In the next section on collecting Conditions into a Problem List, the assumption is made is that WorkingLists can continue to be updated, even though the conditions themselves are not changed. The assumption is also made that the Condition may be updated without the Working List being updated.

Condition and the Problem Oriented Medical Record (POMR):

In Lawrence Weed’s 1970’s vision of the Problem-Oriented Medical Record, care plans, outcome measurement, and quality management were all tied to a single concept. He described the concept that an evidence-based approach to disease management could be executed through focusing medical assessment and treatment on a set of standardized actions for a specific issue or problem. Each problem was addressed through a series of progress notes created “under” that problem heading. Because of this organization of documentation, he envisioned that the treating clinician would more easily be able to navigate to the appropriate data, make decisions regarding the problem with less distraction from irrelevant data, and clearly document the relationship between activities of medicine and the problems they were supposed to solve.

Although the POMR in both paper and computerized formats produced tantalizing hints of improved outcomes, there has been no way to date to practically implement an interoperable POMR across the settings of care for a single patient. The condition tracking structure is a model for providing interoperable problem-oriented medical records across settings of care.

Allergy Guidance: The condition tracking structure is also a model for providing interoperable allergy and intolerance-related medical records across settings of care.

Condition Tracking Structure Overview:

The Condition Tracking Structure is a collection of classes that organizes basic clinical data into longitudinal data structures, which support tracking of complex processes over time. The intention of this model is to support both the tracking of these conditions at the "high-level" as well as the detailed tracking of these complex processes at the "low-level." The decision on whether to track conditions at the high-level or low-level is to be an application-dependent decision. Applications may use techniques described in this section to track conditions but are not required at this level of the specification to include all the tracking information.

In the usual course of presentation, diagnosis, and treatment, the care taker of a condition gains a gradually better understanding of the condition. An example of a condition in the disease management area is the Episode of Illness, which is a synonym for the subset of Conditions that represent disease processes.

In a typical clinical scenario, the patient presents to the clinician with the "chief complaint." This chief complaint represents the patient's view of the condition from which the patient is suffering. Multiple questions and exams are performed on the patient, which allows the clinician to develop a viewpoint on the condition. Tests are ordered and observations carried out by the clinician and time passes. Frequently, the disease evolves and demonstrates more symptoms and signs. At each point in time during multiple contacts with the patient or with patient data, the clinician has evolved a set of potential diagnoses. In addition, during each point in time, various treatments may be recommended by the clinician.

The business need filled by the Condition Tracking Structure is to support this model of multiple contacts with the subject of care by care providers while the condition is evolving, data is being collected, and treatments are being applied. In addition, the business needs of quality improvement and cost accounting are supported by keeping track of the actions for data mining purposes that contribute to the evaluation and management of a condition. Finally, the simple highlighting of certain observations by subclassing them as conditions as clinicians perform the daily workflow of their tasks creates a natural infrastructure for sifting through the electronic medical record for the "most important" information. This ability to rapidly sift and organize information in order to support rapid and efficient patient care has always been the goal of the problem-oriented medical record.

Condition and Condition Node: As noted in the section on the HL7 Condition Class above, an instance of the Condition Class may be updated many times over its lifetime. The utility of this approach is that a dynamic list of Condition instances may be created and maintained while the Condition instances themselves are independently updated. However, the simple model described above is not robust enough to support the POMR. Several use cases challenge the simple model and create the business need for a more complex model:

- The need to support “drill-down” navigation as described by Weed from a problem on a problem list to the data describing assessment and treatment of the problem
- The recognition that conditions may have many “names” besides the Name defined in the Condition.value attribute: for example, what the patient calls “abdominal pain” may be called “gastritis” by the clinician; both are correct
- X-rays, lab tests, diets, and medications may assess or treat more than one condition, creating a poly-hierarchy between data-elements and problem headings within the POMR
- The data that is associated with a single condition grows incrementally and may become vast; Not all associations on a Condition instance can be communicated each time an instance of Condition is updated

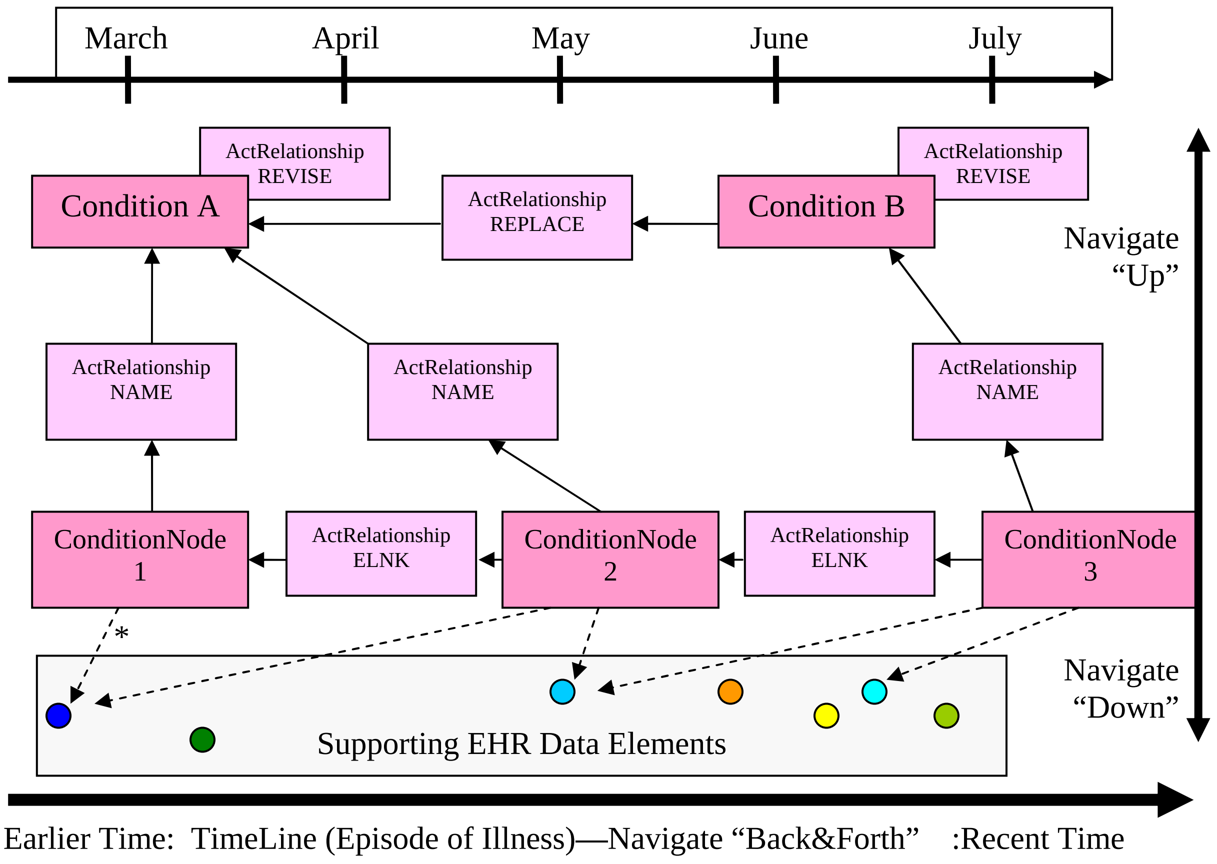
In order to accommodate both simple and complex systems, a two-layer model has been developed by HL7. The simple use cases are supported by the WorkingList and Condition Classes and the standard semantics of Update Mode and the Dynamic Model. The more complex POMR use cases are supported by the introduction of another collector class referred to as ConditionNode.

Unlike Condition, Condition Node represents a point-in-time rather than a range-of-time. It carries no “code” attribute nor a “value” attribute. Its sole purpose is to abstractly represent the longitudinal structure, e.g. Episode of Illness, as a “string of ConditionNode instances” associated by the ActRelationship typeCode EpisodeLink (ELNK) in support of POMR functions. The only other two ActRelationships allowed on CNOD is the NAME relationship to an instance of Condition (that describes the name of the Episode of Illness) and the component (COMP) relationship to the data elements in the EHR that relate to the specific Episode of Illness.

Explaining the POMR model is a challenging task. The following explanation attempts to provide more of the “why and how” of the condition tracking structure. First, the individual attributes and associations will be described. Then the process of building a Condition Tracking Structure over time will be described, along with the detailed versioning machinery utilized.

Reader Orientation to Narrative Using this Graphic:

For now, this “Graphic” will help orient the user to the text; note the graphic navigation terms Navigate “Up” towards Condition: Navigate “Down” towards Supporting Data; Navigate “Back&Forth” along the Episode of Illness Timeline. Note that the ControlActs are not displayed in this Graphic; however, ControlActs accompany every Create or Update of an instance and will have a unique instance identifier (Note: by now in this narrative you should have read the HL7 documentation on ControlActs and Update Mode in the Dynamic Model). The identifiers “A & B” and “1 to 3” are ids for the instances represented in the graphic. Following this walk-thru of the model, the model will be applied to a clinical scenario.



* Note: dashed arrows represent ActRelationship associations of type “Has Support”

ConditionNode Class Explanatory Walk-through:

ConditionNode.classCode: As a subclass of observation (OBS), “CNOD” may inherit all the attributes and associations of Act and Observation. The HL7 definition of CNOD:

An instance of Observation of a Condition at a point in time that includes any Observations or Procedures associated with that Condition as well as links to previous instances of Condition Node for the same Condition

Graphic Interpretation: 1) In this definition, “Observations and Procedures associated with the Condition” are “Supporting data elements in the EHR that have dashed arrows going to them” in the above Graphic (note that these dashed arrows represent ActRelationship “Has Support” associations. 2) In this definition, “links to previous instances of Condition Node” are the “ELNK ActRelationsips instances” in the above Graphic.

CNOD is differentiated from OBS and COND in that it specifically supports the ActRelationship EpisodeLink (ELNK) and is associated with a unique, stereotyped behavior. The reason CNOD is found in the HL7 Observation Class hierarchy is that it is thought to express a judgment, which is common to all observations. However, the judgment is expressed purely through its ActRelationship associations rather than through the “value” attribute. Consequently, the “value” attribute is not present in the class. There is also no need for a “code” attribute, since the method does not presently need to be clarified beyond the method implied by the classCode “CNOD.” However, clinical statement pattern requires a “code attribute.” Therefore, this conflict with clinical statement pattern will need to be resolved.

Rationale: In UML, the creation of a subclass may be justified by the addition of new attributes, the additional constraints on attribute value sets, or the addition of association constraints or specifications, as well as the addition of other behavioral specifications. The use of the class “CNOD” in this model rather than the more general “OBS” is in recognition of these extra constraints and specifications listed for this subclass.

ConditionNode.mood: Mood is always Event (EVN)

ConditionNode.id: datatype “II,” a unique instance identifier such as UUID, GUID, or OID

Rationale: ConditionNode exists in this model in a (0…1) multiplicty to Condition. ConditionNode will not always be paired with a ControlAct-Condition instance identifier tuple, e.g. Condition Name may be updated without any “supporting data” to justify the name change, especially if a “simple” problem list system is integrated with a “complex POMR system.” That means systems may have to query on Condition Nodes related to a Condition id (longitudinal identifier) in order to “build” knowledge of an Episode of Illness. In addition, bilateral navigation between Condition and Supporting Data Elements may require ActReference’s that depend on a unique ConditionNode identifier in order to support navigation “Up” from the supporting data elements to Condition and “Down” from Condition to supporting data elements as well as “Back&Forth” on the thread of ConditionNodes in the Episode of Illness.

Guidance: When a ConditionNode instance is added to an instance of Condition via the “NAME” relationship, there will be a ControlAct that “updates the Condition” as well as a ControlAct that “creates the ConditionNode” in the Update Mode Dynamic Model communication. Instances of ConditionNodes are never created without an associated instance of Condition present.

ConditionNode.statusCode: Except for the rare situations where an error has been made, the state of the ConditionNode is always “Complete.” When a ConditionNode instance has been created in error, the state of the ConditionNode instance is transitioned to Nullify.

Guidance: Since a ConditionNode represents a point in time, there are no updates except to transition the instance from Active to Nullify, which is only performed in case of error.

Participations on ConditionNode: These are well-documented in the Clinical Statement pattern. Typically, since instances of ConditionNodes are never created without an associated instance of Condition present. Participations on ConditionNode will be “conducted” down from the participations on Condition and ControlActs in the communication.

ActRelationships on Condition:

Subject: An instance of Condition is the subject of an instance of the Control Act.

Guidance: However, the only real need for this relationship is when marking a ConditionNode instance as an error with the state transition to Nullify.

Name: An ActRelationship of type “NAME” where the source is a ConditionNode and the target is the Condition.

Guidance: The currently known name of the condition is expressed as the current “value” of the instance of the Condition class. However, the abstract representation of the condition, e.g. Episode of Illness, is the “string of ConditionNode instances” associated by ActRelationship ELNK. In itself, this abstract representation is named by the most currently known identifier tuple (from the associated instance of Condition and Condition’s most recent ControlAct). The ActRelationship NAME between the instance of Condition and the instance of ConditionNode links the current name of the condition to the abstract representation of the condition.

Elink: A recursive ActRelationship of type “ELNK” allows the assertion that an instance of the ConditionNode in one place in the EHR with one instance identifier really represents part of the same condition as a second instance of the ConditionNode in the EHR with a second instance identifier.

Another reason to use the Episode Link ActRelationship is to join two instances of a condition into one condition. This may occur when a clinician makes the determination that what once was thought to be two separate conditions is really one condition.

In the same manner, a third reason to use the Episode Link ActRelationship is to split one condition into two separate conditions. This may occur when what was thought to be a collection of symptoms representing one disease reveals itself with time to represent two diseases.

Has Support: An ActRelationship of type Has Support (SPRT) where the source Act is the observation ConditionNode and the target Act is an observation or treatment or other Act that was related to management of the Condition, e.g. Episode of Illness.

Guidance: In many cases in an EHR, the method of reference from a ConditionNode to the relevant supporting EHR data elements will be an ActReference method (using the id of the target Act as a pointer into the EHR data). In some cases, the referenced data will be immediately adjacent to the ConditionNode in the XML.

Clinical Scenarios:

Several clinical scenarios will be presented with graphics in order to illustrate the use of the concepts that have been described. In this first scenario, the focus is on tracking a simple condition:

Storyboard: Physician Merges Two Conditions

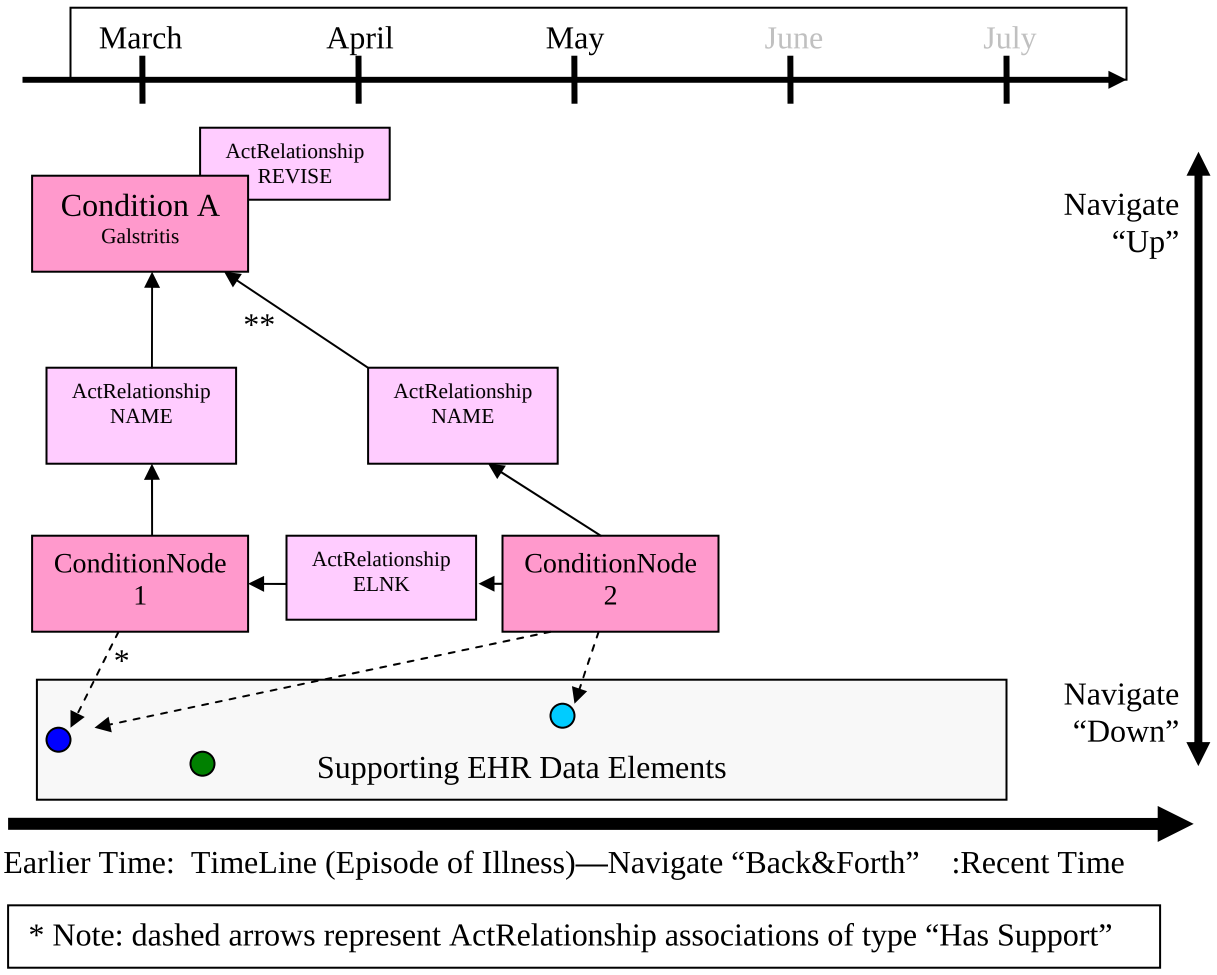
Precondition: In March, Dr. Smith identifies a data element that belongs to a condition and names the condition, Condition A “stomach pain.” In May, Dr. Smith sees the patient a second time, identifies another related data element, and updates his notes in his EMR, including the name of Condition A, which is now “gastritis.” The information regarding this Condition A is published in a clinical document, which is posted to the local RHIO.

Activities: The same patient makes an appointment in June to see Dr. Jones. The reason for the visit is listed on the appointment in Dr. Jones’ EMR as Condition B, “belly pain.” Dr. Jones actually sees the patient in July, and checks the RHIO for previous information. Dr. Jones notes that the instance of Condition A noted in the RHIO document is really the same condition as the instance Condition B, which was listed as the reason for the visit. As Dr. Jones imports the information from the RHIO clinical document into his EMR, the information for Condition A is merged into the Dr. Jones’ EMR as Condition B, which he now names “Irritable Bowel Syndrome.” Dr. Jones publishes a clinical document following the visit and posts the new clinical document to the RHIO with the merged data.

Post-condition: Dr. Jone’s EMR contains the merged Condition B. Both documents are present on the RHIO.

Allergy Guidance: In the storyboard, replace the term “stomach pain” with “nasal congestion.” Replace the term “gastritis” with “spring pollen allergy.” Replace the term “belly pain” with “sinus headache.” Replace the term “Irritable Bowel” with “Pine Pollen Allergies.”

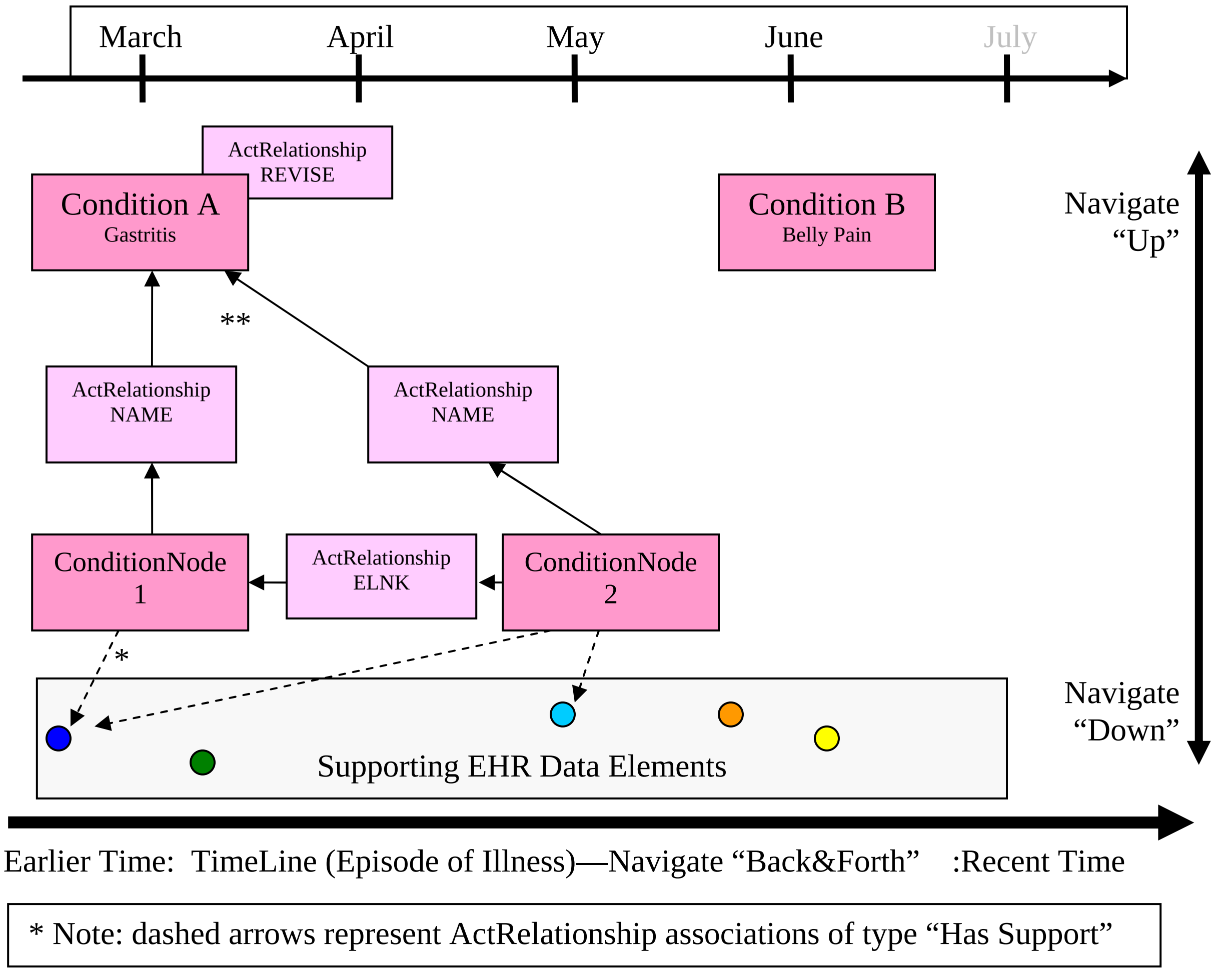
Graphic Interpretation of the Storyboard:


March
April
May
June
July
ActRelationship REVISE
Condition A
Galstritis
**
ActRelationship NAME
ActRelationship NAME
ConditionNode 1
ActRelationship ELNK
ConditionNode 2
*
Supporting EHR Data Elements
Navigate "Up"
Navigate "Down"
Earlier Time: TimeLine (Episode of Illness)—Navigate "Back&Forth" :Recent Time
* Note: dashed arrows represent ActRelationship associations of type "Has Support"

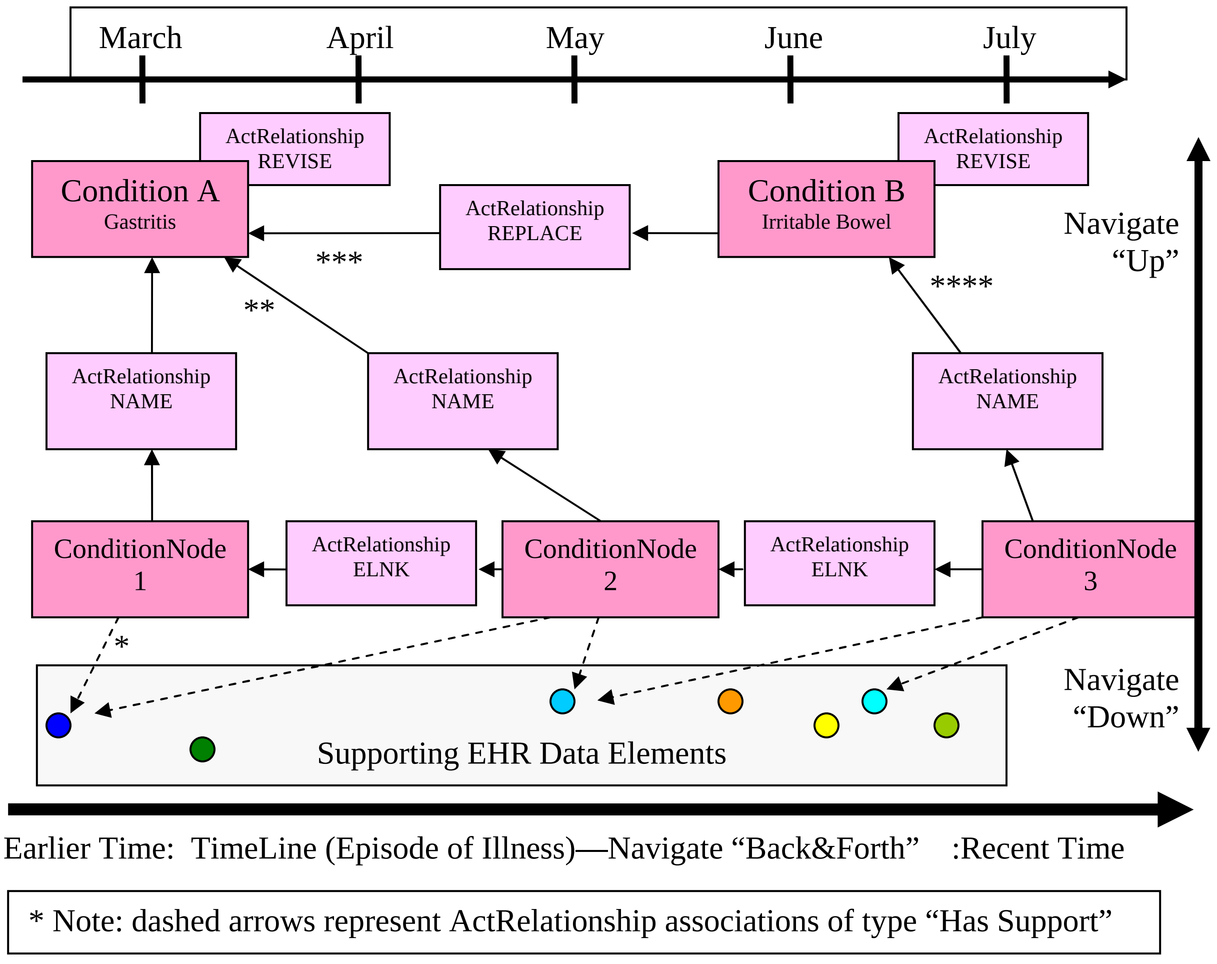
Graphic Interpretation of Precondition:

Precondition: In March, Dr. Smith identifies a data element that belongs to a condition and names the condition, Condition A "stomach pain." ***Dr. Smith's EMR system represents this action by creating the instance ConditionNode 1, creating a Has Support ActRelationship to the data element in his EMR, and a NAME ActRelationship to the new instance Condition A.*** In May, Dr. Smith sees the patient a second time, identifies another related data element, and updates his notes in his EMR to revise the name of the condtion A to "gastritis." ***Dr. Smith's EMR system creates the instance ConditionNode 2, creates an ActRelationship Has Support to the related data elements, creates an ELNK ActRelationship to ConditionNode 1, defining the Episode of Illness, and updates the Conditon A instance with a ControlAct, which is illustrated with *****. The information regarding this Condition A is published in a clinical document, which is posted to the local RHIO.

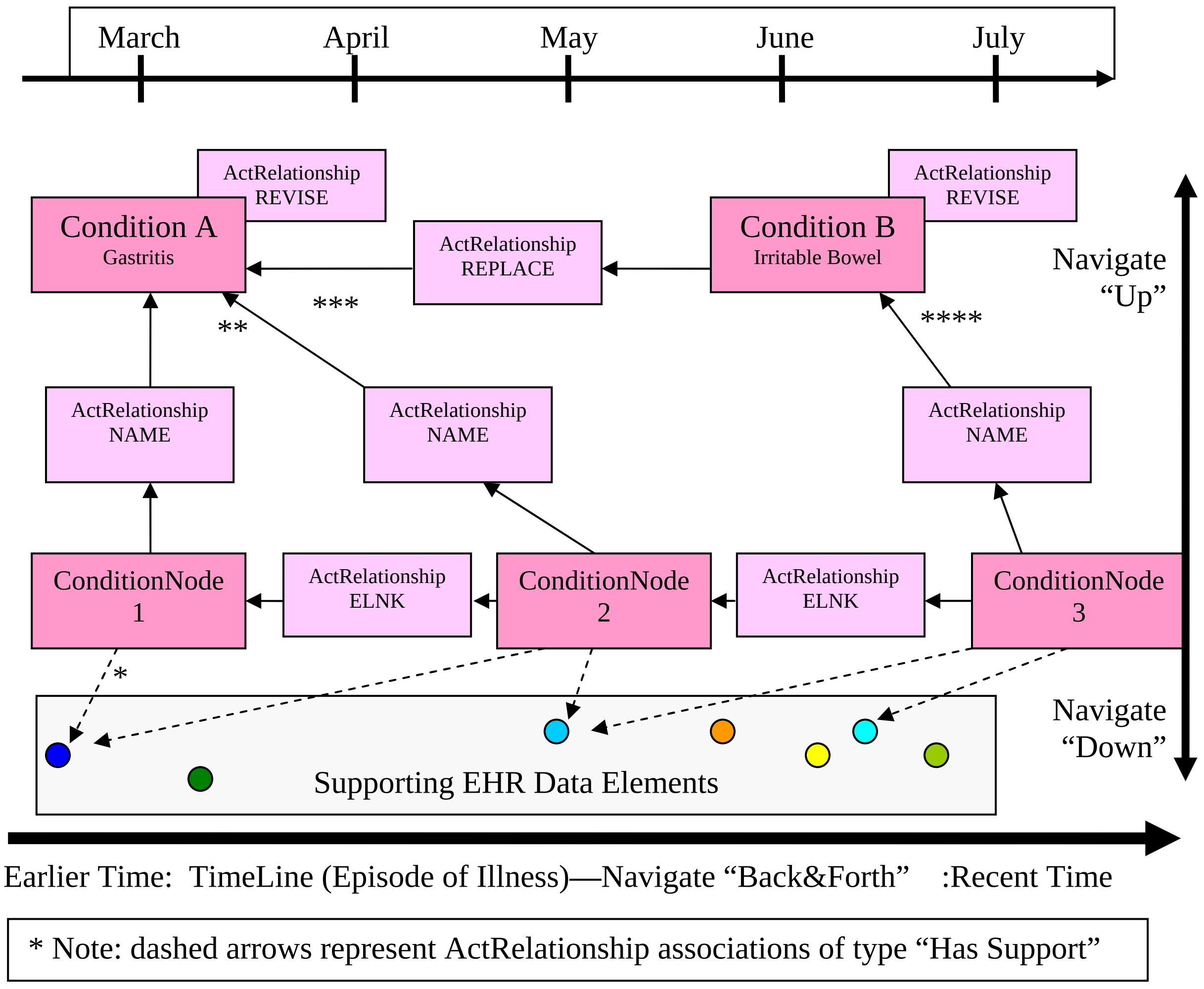
Note: ControlActs may actually exist in messages for every "create instance" function. However, in this scenario, we are illustrating the use of ControlActs for "update instance" functions. Note throughout the scenario, that "update instance" functions are most common on instances of the Condition class.



Activities: The same patient makes an appointment in June to see Dr. Jones. The reason for the visit is listed on the appointment in Dr. Jones' EMR as Condition B. ***At this point, the previous information about the patient is available on the RHIO, but it hasn't been associated with the instance of Condition B. The instance Condition B was created from the reason field in the scheduling system and was not associated with any other information in the EHR via a ConditionNode.***



Dr. Jones actually sees the patient in July, and checks the RHIO for previous information. Dr. Jones notes that the instance of Condition A noted in the RHIO document is really the same condition as the instance Condition B, which was listed as the reason for the visit. As Dr. Jones imports the information from the RHIO clinical document into his EMR, the information for Condition A is merged into the Dr. Jones' EMR as Condition B. ***During the import and merge of data from the RHIO document, Dr. Jones' EMR system creates instance ConditionNode 3, which is associated with ConditionNode 2 via ActRelationship ELINK. Two supporting data elements from Dr. Jone's EMR are associated with ConditionNode 3 via Has Support ActRelationships. ConditionNode 3 is associated with Condition B via the NAME ActRelationship. Condition B is updated with a ControlAct (****) to indicate that Condition B replaces Condition A and Revises the name of Condition A to Irritable Bowel Syndrome. Another ControlAct(**) updates Condition A as "obsolete."*** Dr. Jones publishes a clinical document following the visit and posts the new clinical document to the RHIO with the merged data.



Focus on the Identifiers: In the preceding descriptions of the evolution of the Graphic, it was noted that both Condition A and Condition B were first created and then updated. Also noted was that although the "create" ControlActs were not illustrated, they exist for all the Conditions and Condition Node instances in the Graphic. Let's give the create ControlAct for Condition A the instance identifier "a." Let's give the "create" ControlAct for Condition B the instance identifier "b." The first "update" ControlAct for Condition A already has the instance identifier "**." The second "update" ControlAct for Condition A already has the instance identifier "***." And the "update" ControlAct for Condition B already has the instance identifier "****."

Now the tuples these instance identifiers create can be used to capture the "specific values and associations" existing in Condition A and Condition B at a given point in time:

Month	Condition A tuple	Condition B tuple
March	Aa: Stomach Pain-active	
May	A**: Gastritis-active	
June	A**: Gastritis-active	Bb: Belly Pain-active
July	A***: Gastritis-obsolete	B****: Irritable Bowel-active

Note that ConditionNodes are never updated and do not need to be "versioned." Rather, the addition of the "ConditionNode" level of the Condition Tracking Structure to the simple Condition List model described at the beginning of this document, simply adds additional associations to ConditionNode the basic Condition List model. This means that systems that support the simple Condition List model that are enhanced to support the more complex Condition Tracking Structure for the POMR may maintain backward compatibility. In addition, simple Condition List systems may interoperate with more complex POMR systems at a "lowest denominator level" without loss of data.

Note also that this model allows <u>incremental updates</u> on the Condition Tracking Structure without having to communicate the entire structure each time (although in this example, the Dr. Jone's did include this entire small structure in the clinical document he posted on the RHIO. In the future, he would only need to post what changed in the Condition Tracking Structure (what was updated and added).

Allergy Guidance: When the substitutions of condition names are made in the storyboard, the table for allergy values looks like:

time:

Month	Condition A tuple	Condition B tuple
March	Aa: Nasal congestion-active	
May	A**: Spring pollen allergy-active	
June	A**: Spring pollen allergy-active	Bb: Sinus headache-active
July	A***: Spring pollen allergy-obsolete	B****: Pine Pollen allergy-active

Supporting Data Elements in Allergy: Strictly speaking, the organization of data within the supporting data elements of the EHR is not directly relevant to the high-level description and versioning of the episode of illness. However, allergy data is commonly communicated along with adverse reaction determinations and symptoms, and the allergy model includes specific modeling in the areas of causality assessment and exposure events which is generally important in hospital medicine and clinical trials.

Although many end-users in healthcare focus on the communication of medication allergies once the allergies have been identified by patient or clinician, this focus tends to hide the clinical picture of how allergies are identified, confirmed, and managed over time. To pull the reader out of the medication focus, the poison ivy storyboard is helpful.

Storyboard: Physician Evaluates a Rash

Precondition: Dr. Patricia Primary is viewing the EMR of Patient Everyman. Dr. Everyman has come to clinic to have a rash evaluated. The allergy list is empty.

Activities: Dr. Primary elicits the patient history. The patient has visited an Indian restaurant for the first time and was exposed to many new spices. The patient wonders whether the rash is due to the spices. Dr. Primary examines the rash and notices the typical linear pattern associated with a plant exposure. She questions the patient further and finds the patient had been hiking in the woods earlier in the week. The patient has never had poison ivy. Dr. Primary assures the patient that allergy to Indian spices is unlikely to be the cause of this rash. Rather, Dr. Primary identifies probable poison ivy and educates the patient on the disease. She places Poison Ivy on his allergy list and gives the patient a prescription for Prednisone for the poison ivy.

Post-condition: Mr. Everyman has progress notes, findings, impressions, care plans, and a populated allergy list in the EMR that are captured in a manner that allows navigation from one related data item to the next. Specifically, the user is able to navigate from the current name of the allergy on the allergy list to the string of observations and treatment activities associated with management of that allergy over time. Additionally, the user is able to navigate from a data item to the allergy management programs supported by the data element.

Allergy RMIM Explanatory Walkthrough:

ObservationEvent: As in the previous example of “Sinus headache,” the first time that the EMR might capture a data item later associated with an allergy condition is with some seemingly unrelated “reason for an appointment.” In this “Rash Storyboard,” the reason for visit” or “chief complaint” was “Rash.” In the Allergy RMIM, the “ObservationEvent” clone in the lower-left-hand-corner of the model represents the “pre-causality” determination set of observations on the patient including history, physical exam, and any other preliminary impressions. In some cases, these instances of Observation (with instance identifiers) may have been communicated from another system, such as a stand-alone scheduling system. Modeling of ObservationEvents follows all the same patterns described in various implementation guides for specific kinds of observations such as history, symptoms, physical exam items, and impressions.

Guidance: Note that the appearance of a symptom or diagnosis may “kick off” or trigger the remaining actions in this allergy model. In other words, a patient who appears in the office with a rash may trigger a guideline (see Care Plan below) that includes evaluation of causal exposures and the need for management. Although the mood of this class will be EVN in most medical record assertions, the mood of this class in Definition mood might be paired in a guideline with the same class in EVN.CRT mood. In other words, this class may appear in DEF mood pair with the same class in EVN.CRT mood and a MATCH ActRelationship between the moods as a predicate for a care plan.

Note: This area of the model is immature in the sense that EVN is included in the model to meet immediate needs; but clearly the mood may need to be extended to DEF, EVN.CRT and even other moods to meet the needs of Care Plans.

ExposureEvent: as noted in the Rash Storyboard, there may be one or more exposure events identified. In the hospital or clinical trial, this exposure event may be identified with a specific known material, such as a product from a given manufacturer, and then the model includes support for identifying the specific material. In the ambulatory environment, these exposure events are likely deduced from less definitive knowledge about the specific materials involved, and the communications will be sparser in terms of detail regarding the material. In either case, the Exposure Event is a SubstanceAdministration event, whether the substance was administered accidentally or by design. Modeling of SubstanceAdministrationEvents follows all the same patterns described in various implementation guides for specific kinds of substance administrations such as pharmaceutical administrations, dietary history, and accidental exposures.

CausalityAssessment: The subject (Act Relationship Subject) of the causality assessment is the ObservationEvent. The ExposureEvent is identified either with ActRelationship "Implicates." Observation.code in causality assessment is a kind of observation that allows a Secondary Observation (source act) to assert (at various levels of probability) that the target act of the association (which may be of any type of act) is implicated in the etiology of another observation that is named as the subject of the Secondary Observation. Examples: causality assertions where an accident is the cause of a symptom; predisposition assertions where the genetic state plus environmental factors are implicated in the development of a disease; reaction assertions where a substance exposure is associated with a finding of wheezing. In Observation.value, the conclusion of the assessment should be specified, e.g. from the storyboard, "not a food allergy;" "probable environmental allergy;" "drug intolerance;" or otherwise as specified in the implementation guide.

A_CarePlan (REPC_RM000200)

Note: The Care Plan Model is still immature in the sense that vocabulary domains are not all defined and harmonization is still occurring with Clinical Decision Support on how to represent the order sets within Care Plans. However, this discussion is included in this document as much to illustrate how conditions and, specifically, allergies might be included in a management plan. Note also the use of "care plans" in the introductory storyboards.

Care Plan Structure Overview:

Among the most important Care Structures in the Care Provision Domain is the Care Plan Structure. The purpose of this structure is to define the management action plans for the various conditions identified for the target of care. It is the structure in which the care planning for all individual professions or for groups of professionals can be organized, planned and checked for completion. Communicating explicitly documented and planned actions and goals greatly aids the team in understanding and coordinating the actions that need to be performed for the person. Care plans also permit the monitoring and flagging of unperformed activities and unmet goals for later follow up.

The term Patient Care Planning in MESH is described as:

Usually a written medical and nursing care program designed for a particular patient. Year introduced: 1968"

The basic theory underlying this structure comes from Larry Weeds Problem-oriented Medical Record, also introduced in the 1960s, which is used by many health professions and electronic health record systems as a primary method for organizing work and documentation.

Scope of Responsibility: A Care Plan may be modeled as a list of activities (Acts) or as the Act of Care Provision itself (Note: still to be harmonized with Clinical Decision Support). The Care Plan Structure is attached to the Care Provision Act in two ways. As components of the Care Provision Act, the Care Plans reflect part of the scope of responsibility (a component of the care delivery) defined by the Care Provision Act (see the Care Provision Domain narrative). Component is more specific than pertinent. It says that one act is part of another act. In this case, planning for future care is "part of" the delivery of care, which is the focal class. As pertinent information to the Care Provision Act, the Care Plans are used by the receiver to make sure that activities from other care plans are checked for interactions and not duplicated or mistakenly discontinued by the receiver.

For example, as in the allergy storyboard in the introduction, if a request is made to an allergist to answer a question by evaluating the condition “Spring Pollen Allergies,” that question may be expressed within a Care Plan request. The particular question is asked of the allergist by requesting the Spring Pollen Allergy Condition Assessment in the mood of “RQO,” which has no value defined in the instance. The answer is “filled-in” by the allergist with “Pine Pollen Allergy” in the Condition.value field and/or the Condition.text field or other relevant fields in an instance of Condtion in mood of EVN.

To further illustrate this concept: the patient may have a cardiac valve replacement and be anti-coagulated on coumadin. The patient may also be pregnant, which is noted as pertinent information. A cardiologist may be asked for the highest acceptable Protime INR value that would limit bleeding in the pregnancy but still give protection for the valve replacement. In this case, the component care plan would identify the cardiac valve replacement as the reason for the care plan. The Care Plan Class instance would be in request mood. The observation.mood would be “goal request” (not yet in HL7 vocabulary); the observation.code would code the Protime test; and, as in all requests, the values for the acceptable INR range are not included, indicating that they should be filled in by the cardiologist (see Table 1).

In addition, the cardiologist may add a schedule of protime tests in proposal mood to the care plan and include recommended medication adjustments for the various INR values that might be obtained from the protime measurements (see the section on Decision Support below).

Class Structures: The Care Plan Structure is composed of two Care Provision Acts (referred to as the Care Plan Class and the Guideline Class) along with Component Act Relationships to constrained versions of the Care Statement. These two Classes (Care Plan and Guideline) act as collectors of Care Statements.

Moods and States of Care Plans and Guidelines: The difference between the Guideline Class and the Care Plan Class is the difference in concept between a Guideline, which is "a general list of actions that are defined to be applied to all appropriate targets of care when certain pre-conditions are met" versus the Care Plan, which is "the specific list of actions that are intended to be applied to a specific target of care." Therefore, the mood code of Guidelines is Definition Mood, and the mood code of Care Plans is Intent Mood and its children moods (See table 2).

A Request would be used to ask a consultant to recommend a care plan. Proposal would be used by a consultant recommending a Care Plan. Intent itself would be used to activate a care plan. Care plans may be scheduled in Appointment mood. For each mood, several states are valid, in particular the states of "active" and "completed. The one or more care plan instances in a given communication potentially contain any acts included in the Care Statement Structure.

Typically, a specific Care Plan will have actions added or taken away from the list of actions in a Guideline, often referred to as Care Plan variations. In relationship to clinical pathways, such variations are normally documented in order to measure the variances on populations of patients, which can be used to improve the guidelines. For the purpose of clarity within the Care Plan Structure, variation is defined here as something that is different from a standard (i.e., the Guideline), where variance is a statistical measure of dispersion, or a calculated degree of variation (How much does this patient differ compared to a group of similar patients). Of course the reasons for these Care Plan variations include the presence of co-morbidities, judgments of the clinician, non-consent of a patient, and other reasons described in the disease management literature.

Often, in a Care Provision communication, the Guideline is named simply as a reference to an external guideline without specifically representing all the Care Statements implied by the Guideline. However, other business needs may require that all the Care Statements that exist in a specific Guideline definition be specifically communicated as well. It should go without saying that in a Care Provision communication, the Care Statements collected by a Guideline need to be kept completely separate for patient safety reasons from Care Statements collected by a Care Plan. However, it is important to be able to document the variation between what the Guideline (in Definition Mood) states should be done, and what according to the Care Statements has actually been done (Any Act in Event Mood). In the Care Plan RMIM, what "has been done" is documented in the "fulfillment" ActRelationship from Care Statements CMET to the Care Plan choice box.

Additional Classes: Participations follow the general model for Care Provision Domain as a whole. Reason for the Care Plan is represented by the Reason ActRelationship associated to the disease Condition. Goals and the other Predicate moods (Event.Criterion and Option) may be applied to the whole care plan as well as to component phases of the care plan (see Table 3).

Moods and States of Care Plan Components: As mentioned above, Care Plans are collectors of Care Statements. Like the Care Plan itself, these Care Statements collected by the Care Plan will have moods of either Intent or the children of Intent. However, like any Care Statements in the Intent moods, instances of these Care Statements will be created as the Act concept moves through the downstream stages of the business cycle of Acts, e.g. Events. As mentioned above, the Fulfillment ActRelationships to the Care Plan component Acts represents the downstream instances in the Act business cycle.

Between the use of Definition mood in the Guideline Class Components, the Intent moods in the Care Plan Class Components, the Acts in Predicate moods related to the Care Plans and Components, and the Fulfillment Acts, all the options for moodCode can be used in the Care Plan Structure, thus giving an overview on the dynamics of a whole set of (specialized) Acts planned or performed during Care Provision. Clinician thus can track the status of their work carried out on behalf of Patients or other Targets of Care.

Phases of Care Plans and Guidelines: Essential to many Care Plans and Guidelines is the concept of sequential "Phases" or stages. These phases of the Care Plan are to be modeled using recursive component ActRelationship on the focal act of Care Provision. Each component phase is then considered a mini-guideline or mini-care plan. Typical phases include those defined by nursing practice, e.g. assessment, plan, performance and evaluation; or those defined by organizational practice, e.g. pre-operative, operative, post-operative recovery, post-operative rehabilitation, etc. (see Figure 2) The decision support tools discussed below govern entrance and exit logic from these mini-guidelines and care plans.

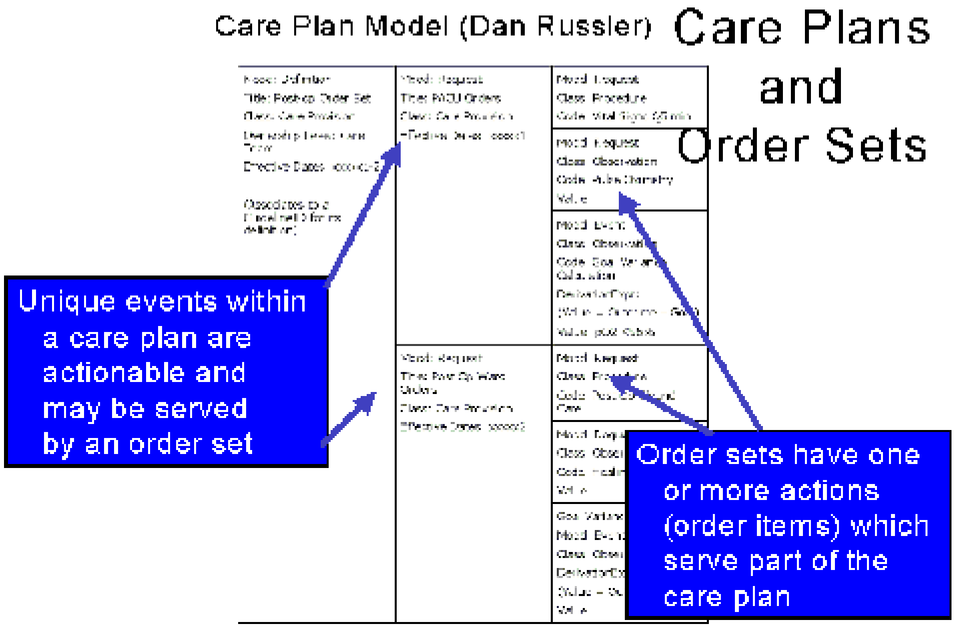
Goals in Care Plans: Goals are common within Care Plan phases, and often must be met in order to move a patient from one phase of a Care Plan to another. For example, for a patient to move from the post-operative recovery room to a surgical floor bed, certain vital signs must be stable. Whether these goals are considered met or not met may depend on clinician judgment or on decision support rules stored in Observation.derivationExpr. A typical goal is expressed as an Observation scheduled for some time in the future with a particular value in the observation.value field. For example, a rule may state that the observation of a heart rate between 60 and 100 must be present at the time of discharge from recovery. A Care Plan may identify this heart rate as a goal for 30 minutes after entry to the recovery unit.

Order Sets in Care Plans: Order sets can organize and structure complex health care plans into useful units of work for clinicians. An order within an order set can request any clinical activity including referrals (requests for encounters), acts, supplies, procedures and observations among others. Structured observations and goal setting can also be supported from the order set. The sharing of care protocols and regimes authored as phased order sets has a much broader audience for work on error reduction, standardization of care and quality improvement. Order sets in this context can be envisioned as action elements within a care plan, allowing a clinician to organize and request a step in a larger protocol.

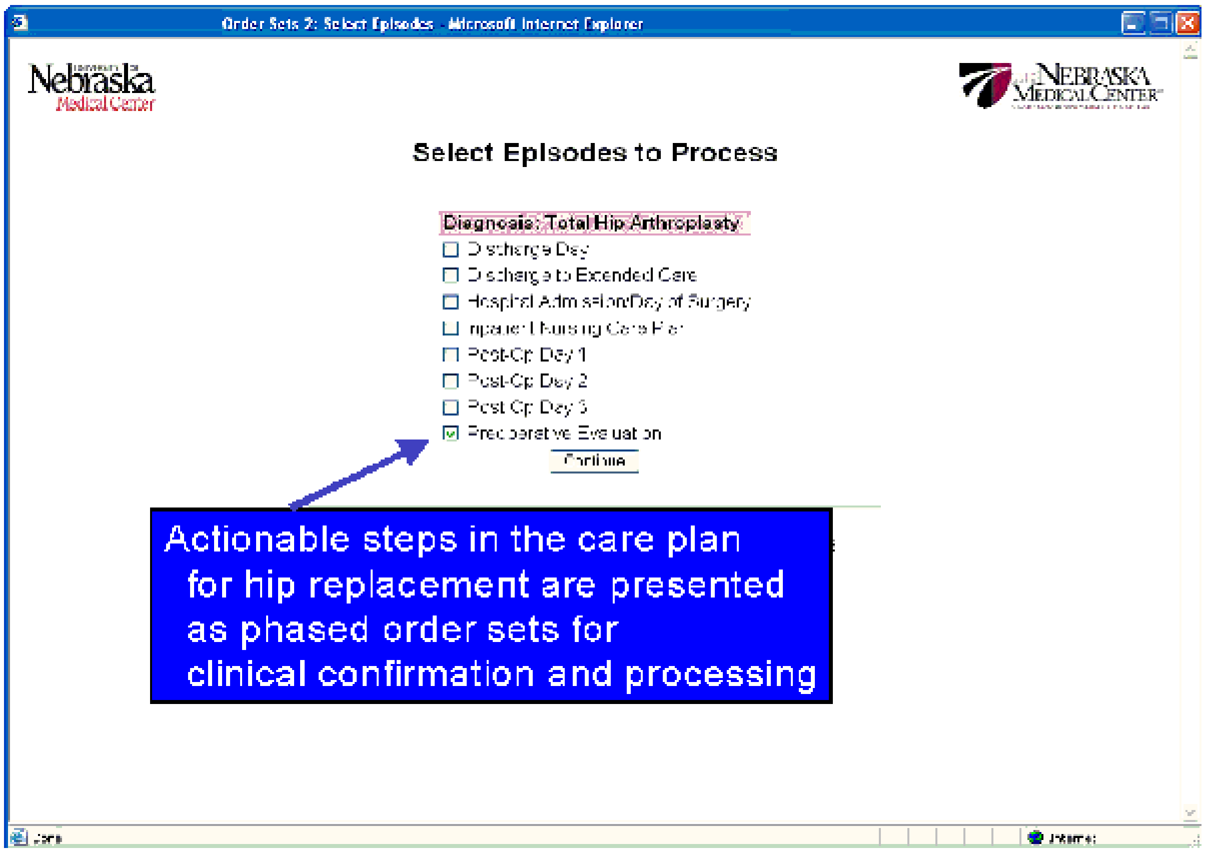
Definition of the structure and function of orders and order sets for an implementation-independent EHR referred to as the virtual medical record (vMR) is a central issue in development of interoperable functionality necessary to publically published guidelines that may be used by multiple institutions. HL7 Clinical Decision Support Technical Committee is in the process of describing a multi-layered standard that supports the publication and maintenance of order set libraries, the sharing of order sets between collaborating institutions, the structuring of order sets to support effective clinical use, and the interoperation of order sets within advanced clinical guideline support software. Organizations employing this standard would deploy the depth of specification appropriate to their enterprise needs and the capabilities of their clinical systems architecture and performance.

Within a specific phase or order set, the detail of how care is intended to be delivered is expressed. This detail includes the expression of goals and measurements of outcomes after interventions are applied. Order sets specifically include particular sequences of orders and decision support rules, including rules for stopping or revising orders and rules for entering or exiting the current order set in order to enroll the patient in a new phase of the care plan (See Figure 3).

Figure 3



Care Plan Model (Dan Russler)
Care Plans and Order Sets
Title: Post-op Order Set
Title: PACU Orders
Class: Procedure
Class: Observation
Title: Post Op Ward Orders
Unique events within a care plan are actionable and may be served by an order set
Order sets have one or more actions (order items) which serve part of the care plan



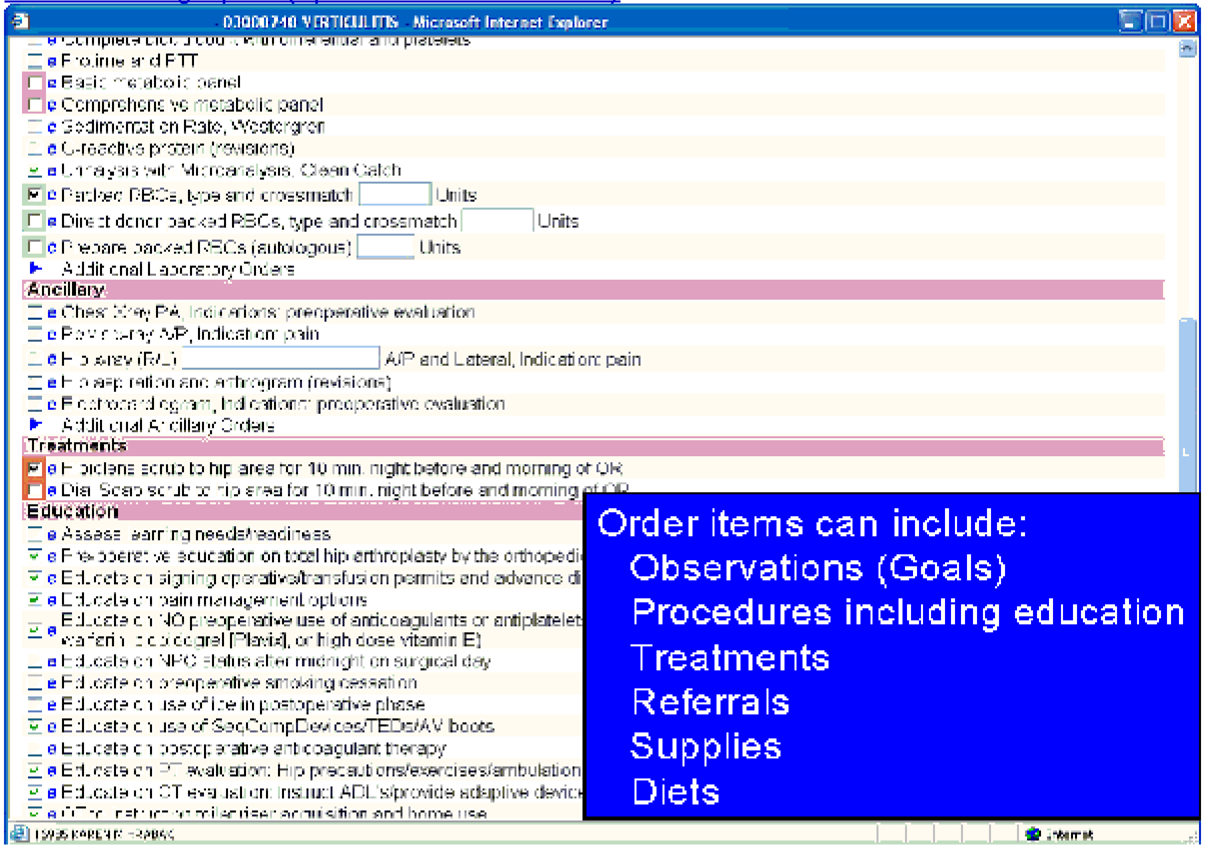
Select Episodes to Process

Diagnosis: Total Hip Arthroplasty

- ☐ Discharge Day
- ☐ Discharge to Extended Care
- ☐ Hospital Admission/Day of Surgery
- ☐ Inpatient Nursing Care Plan
- ☐ Post-Op Day 1
- ☐ Post-Op Day 2
- ☐ Post-Op Day 3
- ☑ Preoperative Evaluation

Actionable steps in the care plan for hip replacement are presented as phased order sets for clinical confirmation and processing

Link to wide graphic (opens in a new window)


03000240 VERTICULITIS - Microsoft Internet Explorer

Complete blood count with differential and platelets
Protime and PTT
☐ Basic metabolic panel
☐ Comprehensive metabolic panel
Sedimentation Rate, Westergren
C-reactive protein (revisions)
✓ Urinalysis with Microanalysis, Clean Catch
☑ Packed RBCs, type and crossmatch ____ Units
☐ Direct donor packed RBCs, type and crossmatch ____ Units
☐ Prepare packed RBCs (autologous) ____ Units
Additional Laboratory Orders

Ancillary

Chest X-ray PA, Indications: preoperative evaluation
Pelvic x-ray AP, Indication: pain
Hip x-ray (R/L) ____ A/P and Lateral, Indication: pain
Hip aspiration and arthrogram (revisions)
Electrocardiogram, Indications: preoperative evaluation
Additional Ancillary Orders

Treatments

☑ Hibiclens scrub to hip area for 10 min. night before and morning of OR
☐ Dial Soap scrub to hip area for 10 min. night before and morning of OR

Education

Assess learning needs/readiness
✓ Pre-operative education on total hip arthroplasty by the orthopedi
✓ Educate on signing operative/transfusion permits and advance di
✓ Educate on pain management options
✓ Educate on NO preoperative use of anticoagulants or antiplatelet
warfarin, clopidogrel [Plavix], or high dose vitamin E)
Educate on NPO status after midnight on surgical day
Educate on preoperative smoking cessation
Educate on use of ice in postoperative phase
✓ Educate on use of SeqCompDevices/TEDs/AV boots
Educate on postoperative anticoagulant therapy
✓ Educate on PT evaluation: Hip precautions/exercises/ambulation
✓ Educate on OT evaluation: Instruct ADL's/provide adaptive devic
✓ OT instruction on raised toilet seat acquisition and home use

Order items can include:
- Observations (Goals)
- Procedures including education
- Treatments
- Referrals
- Supplies
- Diets

Clinical Decision Support Rules in Care Plans: Management of the flow between phases of a care plan is implemented via decision support rules expressed in instances of the observation class.. These rules, in whatever decision support language is used, may be expressed in the Observation.derivationExpr attribute in the observation class included as a component of the current phase of a Care Plan. Typically the rule evaluates current observations on the patient against goals expressed for the current phase of care and determines whether the patient is ready to move on to the next phase of care.

Further guidance on the use of these decision support languages will be published by the HL7 Clinical Decision Support Technical Committee. Examples used in Care Provision models include expressions as simple as formulae for calculations, such as summing up a total score, or as complicated as making a choice between two potential actions based on decision criteria expressed in the derivation expression logic syntax.

Care Plan Structure Walk-through:

A formal walk-through will be developed. For now, the reader should refer to the documentation for the HL7 Reference Information Model for a complete set of attribute descriptions for each class in the Condition Tracking Structure.