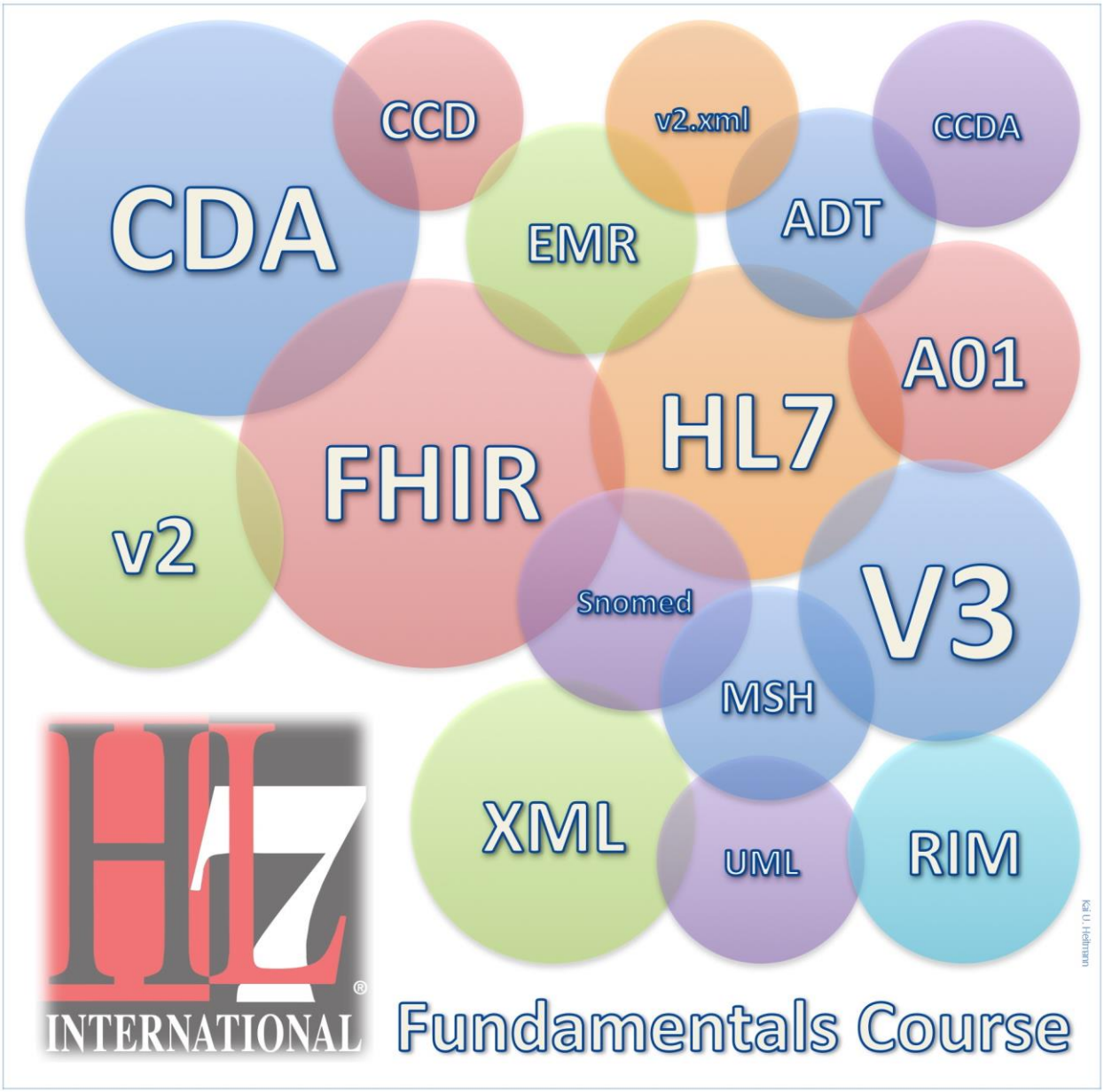


Introduction Module, Unit 2:

Introduction to UML

Reading Material

Table of Contents

Table of Contents 2
Unit Content and Learning Objectives 4
1. What is UML? 5
2. UML Diagram Overview 7
Use Case Diagrams 7
Actors 7
'Include' and 'Extend' relationships between Use Cases 8
Packages in Use Cases 8
UML MODELING CASE STUDY: Hospital sends Complex Tests to Referral Laboratory 9
Step 1: Casting the Actors 10
Step 2: Create the High-Level Use Case Diagram 10
Step 3: Detail the Use Cases 11
Are Use-Case Diagrams enough? 13
3. Activity Diagrams 14
Activity 14
Partitions 15
Control Flow 15
Object Flow 16
Event 16
Fork/Join 16
Process Initiation 17
Process Termination 17
4. Class Diagrams 18
Package 18
Class 19
Attribute 19
Association 20
Multiplicities 20
Generalization Relationship 21
Composition Relationship 21
5. Sequence Diagrams 22
Overview 22
What are Sequence Diagrams? 22
Robustness Diagrams 23
6. State Transition Diagrams 25
7. Use of UML in HL7 V2.x 27
HL7 V2.x Use Case Diagrams 27
HL7 V2.x Activity Diagrams 28
HL7 V2.x Interaction Models 28

8. Use of UML in HL7 FHIR 30
9. Use of UML in HL7 V3 32
HL V3 Class Diagrams 32
HL V3 Use Case Diagrams 32
HL7 V3 Storyboards 33
HL7 V3 Sequence Diagrams 33
HL7 V3 State Machine Diagrams 34
10. Use of UML in HL7 Domain Analysis Models (DAM) 36
HL7 DAM Use Case Diagrams 36
HL7 DAM Activity Model Diagrams 37
HL7 DAM - Class Model 38
11. Use of UML in CDA 41
Overview 41
CDA Template Classes 41
The Categories of Template Classes 42
The HL7 CDA Refined Message Information Model (RMIM) 42
Additional Reading Material 43

Unit Content and Learning Objectives

Units 1 of this Course introduced you to the need and creation of health information standards as well as standardized use of the language ("Controlled Vocabularies") used to communicate healthcare information.

This unit covers two important tools for creating and operating healthcare information systems:

- The Unified Modeling Language (UML)
- eXtensible Markup Language (XML)

UML is the modeling notation tool[1] used by HL7 to specify, visualize and document the structure and design of its models.

[1] In this Unit, UML is referred to as a 'modeling notation tool' to distinguish it from software tools that implement UML.

1. What is UML?

Overview

The Unified Modeling Language (UML) is a graphical modeling notation that enables the specification, visualization and documentation of models of software systems. UML can model application structures, system behaviors, systems architectures, business processes and data structures in a standardized way.

UML was created and is maintained by the Object Management Group (OMG), a not-for-profit computer industry specifications consortium.

UML can model any type of application, running on any type of hardware, operating system, middleware[2], programming language and network. Its flexibility also allows the modeling distributed applications. Built upon fundamental object oriented ("OO") concepts including class and operation, UML is a natural fit for object-oriented languages and environments such as C++, Java and C#. Nonetheless, UML can also be used to model non-OO applications in, for example, FORTRAN, Visual Basic, COBOL, etc.

Refinements of UML tailored for specific purposes, known as "UML Profiles", can model transactional, real-time and fault-tolerant systems.

Models vs. Methodologies*: The process of gathering and analyzing an application's requirements then incorporating them into a program design is a complex one. The industry currently supports many methodologies that define this process. One characteristic of UML is that it is* ***methodology-independent****. Regardless of the methodology used, UML can express the results of the systems analysis and design.*

What can be modeled with UML?

UML 2.0 defines thirteen types of diagrams, divided into three categories:

Six **Structure Diagrams**: **Class Diagram**, **Object Diagram**, **Component Diagram**, **Composite Structure Diagram**, **Package Diagram** and **Deployment Diagram**.

Three **Behavior Diagrams**: **Use Case Diagram**, **Activity Diagram** and **State Machine Diagram**.

Four **Interaction Diagrams**: **Sequence Diagram**, **Communication Diagram, Timing Diagram** and **Interaction Overview Diagram** - all derived from a more general **Behavior Diagram**.

UML has been used by the HL7 standards development community since the mid 1990's. In particular, UML was used to create the HL7 Reference Information Model (RIM) that was the basis of

[2] "Middleware" is the term used for any software whose main function is to interconnect other software systems.

the work on the HL7 V3 and HL7 CDA standards. The recently published FHIR standard also uses UML to document its atomic building blocks, the "FHIR Resources".

Note that each diagram is a way of representing the use of UML for modeling a particular applications or system and it uses the modeling concepts, whose semantics is captured in the OMG UML standard. The details of these semantics is beyond the scope of this Course.

Therefore, in this Unit we will only explore in detail the following five diagrams that are used in the HL7 standards:

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- State Machine Diagram
- Activity Diagram

2. UML Diagram Overview

The following is an overview of the most frequently used UML Diagrams.

Use Case Diagrams

A "Use Case" represents a discrete unit of interaction between a user (human or machine) and the system. This interaction is a single unit of meaningful work, such as **Create Patient Account** or **View Patient Account Details**.

A Use Case Model describes the functionality of a system. Each Use Case describes the functionality to be built in the proposed system, which can include another Use Case's functionality or extend another Use Case with its own behavior.

A Use Case diagram usually has a Use Case description attached because the diagram alone does often not provide enough detailed information. A Use Case generally includes:

- **Requirements**: The features that a Use Case must provide to the end user; for example "update document heading". These correspond to the functional specifications found in structured methodologies and form an "agreement" that the Use Case performs an action or provides some value to the system.
- **Constraints**: The formal rules and limitations a Use Case operates under, defining what can and cannot be done. These include:
  - **Pre-conditions**: Activities that must have already occurred or items needed to be in place before the Use Case is run; for example, "patient demographics record stored" must precede "patient surgery record stored"
  - **Post-conditions:** That must be true once the Use Case is complete; for example, "patient demographic data is recorded"
  - **Invariants**: Conditions that must always be true throughout the time the Use Case operates; for example, a clinical document must always have author identification.
- **Scenarios**: sequential descriptions of the steps taken to carry out the use case or the flow of events that occur during a Use Case instance. These can include multiple scenarios, to accommodate exceptional circumstances and alternative processing paths. These are usually created in text and may correspond to a textual representation of the Sequence Diagram.
- **Scenario Diagrams**: Sequence diagrams to show the workflow; similar to Scenarios but are graphically depicted.
- **Additional Attributes**: Such as implementation phase, version number, complexity rating, stereotype and status.

Actors

Use Cases are typically related to "actors", which represent human or machine entities. Actors use or interact with the system to perform a piece of meaningful work that helps them to achieve a goal. The set of Use Cases each actor accesses defines its overall role in the system and the scope of its action.

Thus, the first task when confronting a system will be to select the actors involved and determine their relationship with the system. This process is called "Actors Casting". Actors are represented with this symbol, with the name of the actor below it:

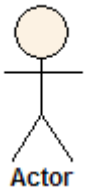


Figure 1: Actor

For one entity to be considered an actor, it should be:

- An Involved Person: A role for persons (through access to the system and object manipulation capability) interacting with the system through a GUI.
- An Implied Person: A role for persons receiving value from the system without any direct interaction (no interface to the GUI, thus no events from these actors).
- A Subsystem: A role for external systems interacting with our reference system through a communications interface. The traffic between this kind of actor and the reference system defines the interoperability schema but not its implementation. Later on, this interoperability schema can be implemented through web services, messaging, etc.

When dealing with interoperability problems, usually actors are defined as the applications involved in a Use Case, and each Use Case involves a specific interaction between the applications being connected - we keep the actual person (implied/involved) roles out of the problem.

'Include' and 'Extend' relationships between Use Cases

One Use Case can **include** the functionality of another.

For example, when listing a set of patient visits to choose from before modifying a selected patient visit information, the "list patient visits" Use Case would be included every time the "modify patient visit" Use Case is run.

A Use Case can be included by one or more other Use Cases, so it helps to reduce duplication of functionality by factoring out common behavior into Use Cases that you can re-use many times.

In addition, a Use Case can **extend** the behavior of another, typically when exceptional circumstances are encountered. For example, if patient information is subject to some special privacy policy, and approval from the hospital privacy officer is required before the information can be accessed, then the "get privacy officer approval" Use Case could optionally extend the regular "access patient information" Use Case.

Packages in Use Cases

Packages are optional containers used for functionality and are built based on any attribute of the included use cases. Examples of criteria for packaging a Use Case are "Release" (package the use

case based on the release of the application(s) which include them) or based on their relationship with one or more actors.

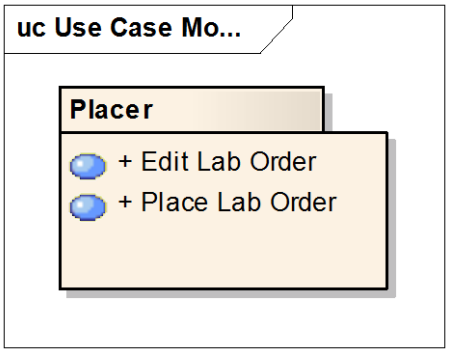


Figure 2: Package

Before we look at a Case Study in UML Modeling, here is a summary of important components of a UML Use Case Model:

Actors: A human or machine entity that initiates a use case and interacts with other actors
Description: A brief high-level description for the reason and outcome of the Use Case
Trigger: An event that initiates the Use Case - either an external event or time triggered
Preconditions: Any activities or conditions that must be true before the Use Case can be started
Postconditions: Any activities or conditions that must be true after the Use Case has completed
Normal Flow: Detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions
Alternative Flows: Detailed description of other normal user actions and system response
Exceptions: Any error conditions that could occur during execution of the Use Case
Includes: Any other Use Case included or "called" by this Use Case
Special Requirements: Requirements not part of functionality: performance, quality, etc.
Assumptions: Any assumptions made during systems analysis and Use case creation

UML MODELING CASE STUDY: Hospital sends Complex Tests to Referral Laboratory

Using the scenario below,[3] we will build a UML Use Case step-by-step:

The Goodhealth Hospital has contracted a Referral Laboratory to perform those complex lab tests it is not equipped to do.

The orders are entered through the web portal of the Referral Lab Information System that connects to the Goodhealth Hospital Information System (HIS) to get patient data and identify the ordering hospital staff member

Only hospital staff ("Placers") can order referred lab tests.

An order has a header with the hospital ID, treating doctor details, patient demographics, insurance details specimen ID as well as the list of tests and associated clinical notes.

[3] Note: This is a hypothetical scenario for educational purposes only; any resemblance to real clinical practice is purely coincidental.

The Referral Lab will analyze the specimens, generate the results for each requested test and validate the results. Finally, the Referral Lab will publish the final report in the hospital document repository system.

Step 1: Casting the Actors

The first step in build a UML Use Case is the identification and creation of Actors..

Hints for creating actors:

- Place your primary actor(s) in the top-left corner of the diagram
- Draw actors to the outside of a use case diagram
- Name actors with singular, business-relevant nouns
- Associate each actor with one or more use cases
- Actors model roles, not positions
- Actors don't interact with one another
- Introduce an actor called "time" to initiate scheduled events

From the Case Study description above, three involved persons or roles, four subsystems and an implied person or role can be identified:

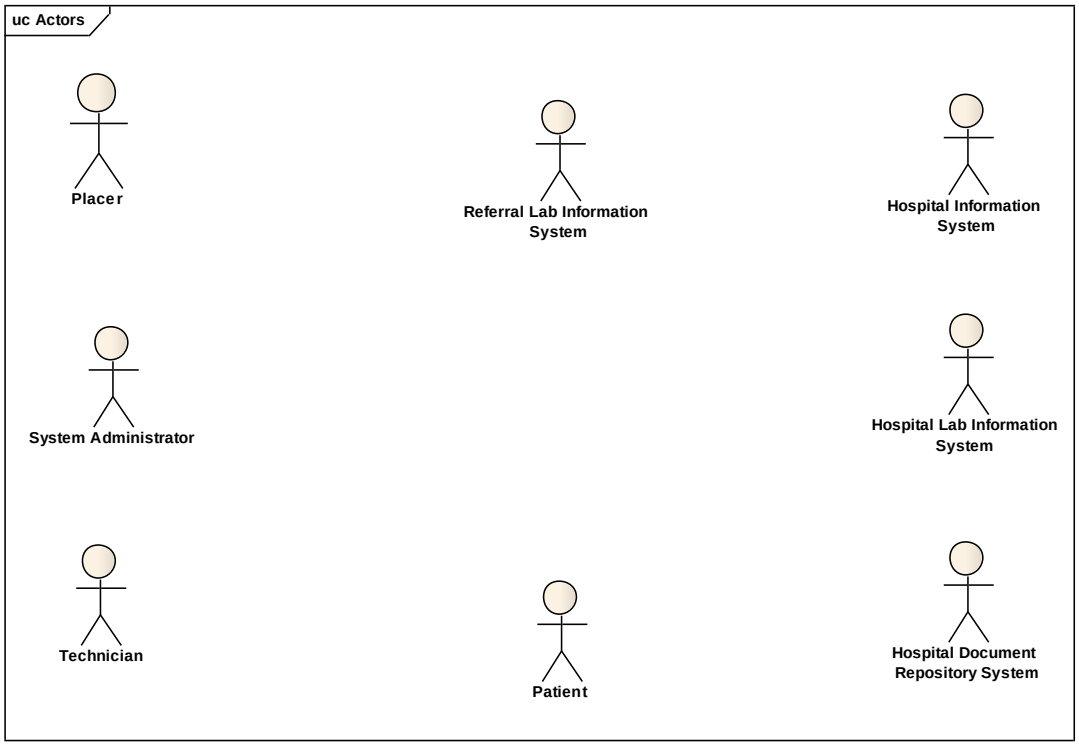


Figure 3: Actor Casting

Step 2: Create the High-Level Use Case Diagram

After casting (or choosing) the actors, an initial diagram called the UML High-Level Use Case diagram is defined. The Use Case Diagram visualizes which actors are participating in which use cases and also to understand the scope of the system; e.g. which operations belong to the system and which operations do not:

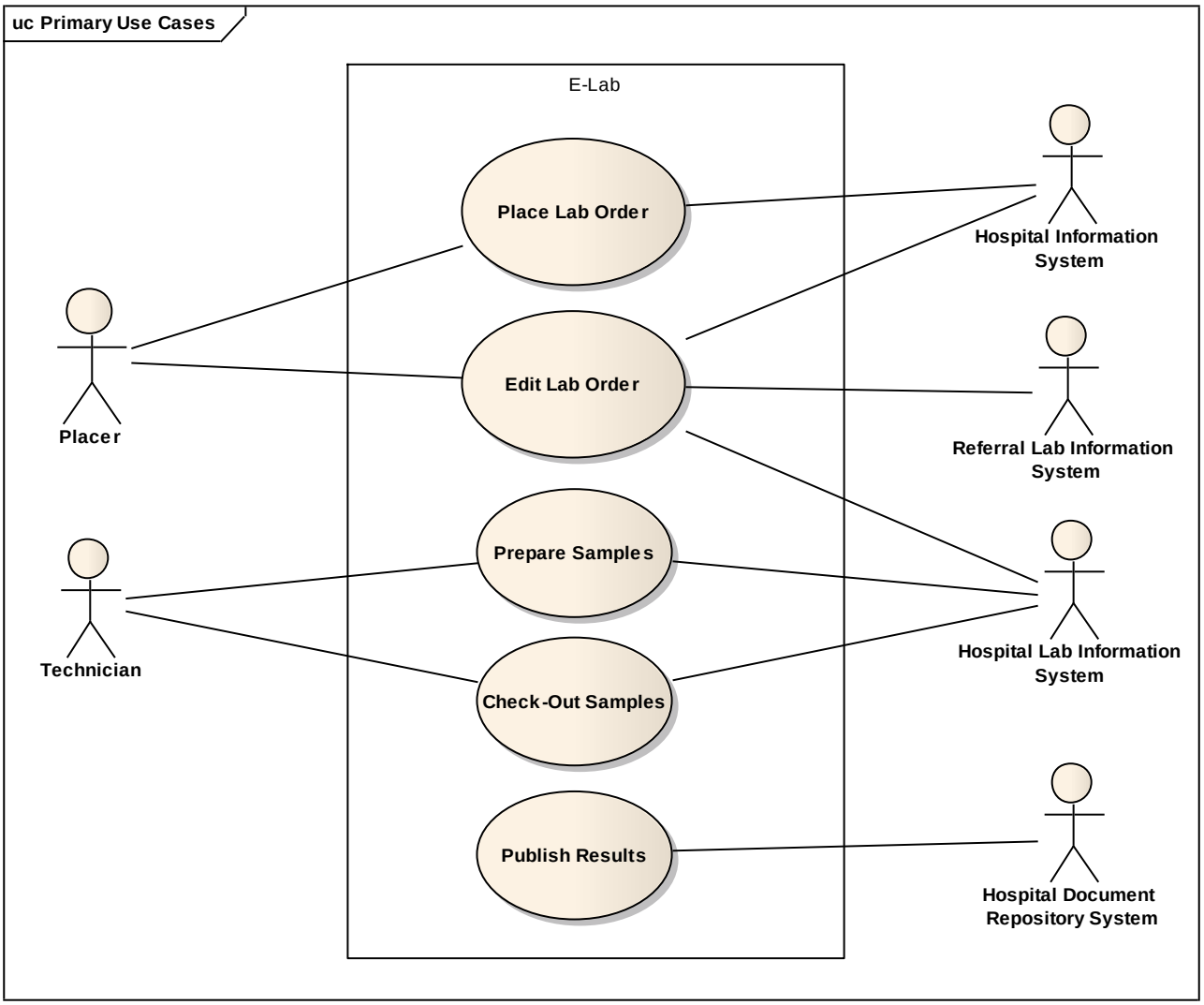


Figure 4: High Level Use Case Diagram

Step 3: Detail the Use Cases

The detailed Use Case specifies how the Actors communicate and interact.

To create a detailed Use Case, a thorough systems analysis should have established the following information:

Preconditions:

- The Hospital Information System (HIS) shall be available.

- The Hospital Lab Laboratory Information System (LIS) shall be available.
- The Referral Lab LIS shall be available.
- The Placer must be authorized by the Referral Lab to place orders.
- The order to transfer to the Referral Lab must exist in the Hospital Lab LIS.
- The order must contain one or more tests to be performed by the Referral Lab.

Postconditions:

- The information about the referred order shall be saved into the Referral Lab LIS
- The state of the tests in the Hospital Lab LIS shall be changed to "sent to Referral Lab LIS"

Normal Operation Scenario:

1. **Placer** login to **e-Lab** system (check user profile in **Referral Lab LIS** and **Hospital Lab LIS**)
2. **Placer** requests to **Hospital Lab LIS** for a list of samples to be sent to the Referral Lab
3. **Placer** reviews the list of samples to be sent to the **Referral Lab LIS**.
4. **Placer** selects one sample to be sent.
5. **Placer** reviews and/or updates the list of ordered tests for each sample.
6. **Placer** optionally adds a comment.
7. **Placer** confirms the order, test selection and comment.
8. **Order** is registered into **Referral Lab LIS** and status is changed in **Hospital Lab LIS**.
9. **Placer** gets a confirmation of the order status.

Lab System not working - Alternative Scenario 1:

1. User cannot login to the system
2. Display error message and record log message - after three login failures, if the user is known but the password does not match, disable the user account.

No Orders - Alternative Scenario 2:

1. There are no orders to be sent to the Referral Lab
2. Display error message

Here is the detail of the first use case (Place Lab Order) in the High Level Use Case Diagram above in which a data entry Actor ("Placer") enters a test request in the receiving Actor ("Referral Lab Information System"):

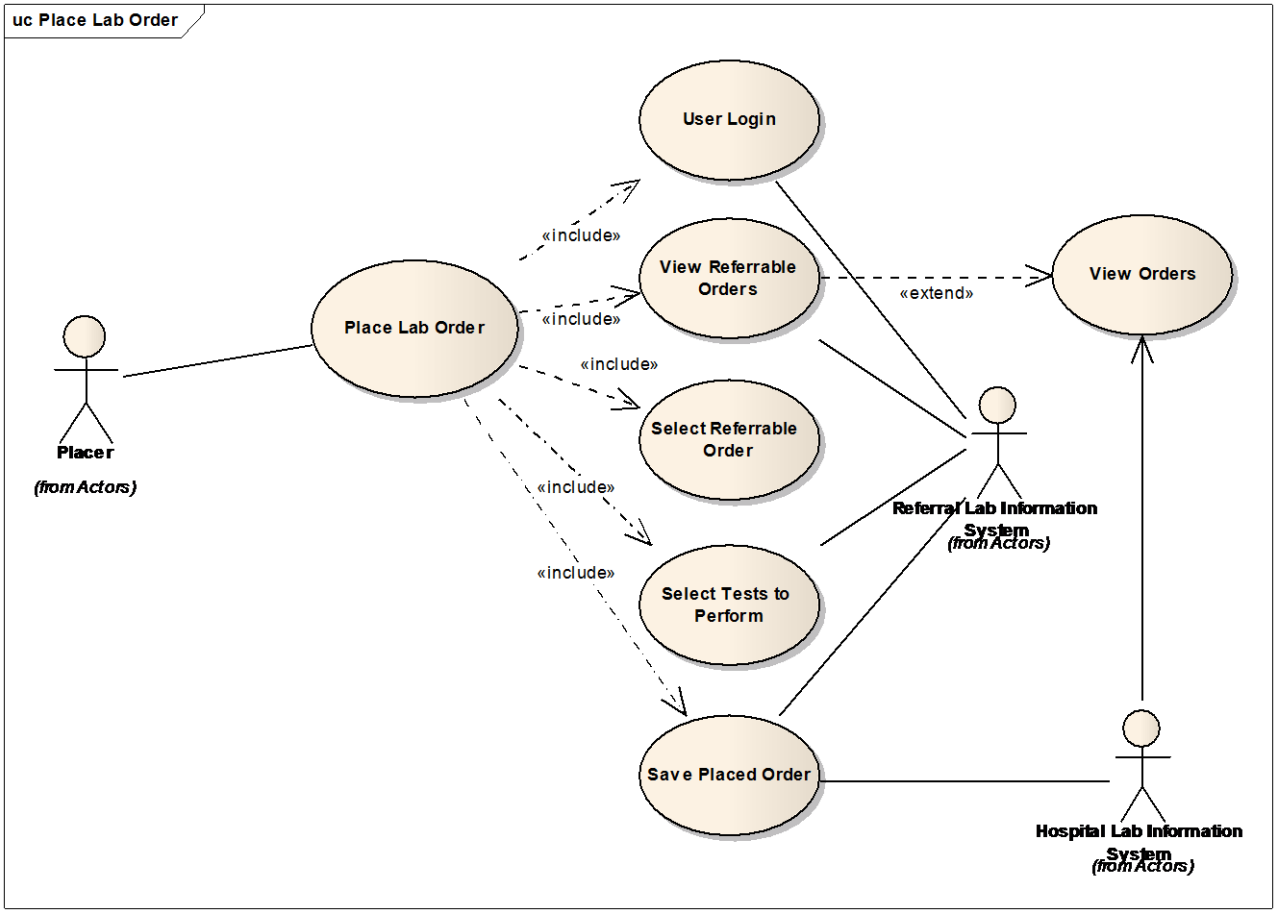


Figure 5: 'Place Lab Order' Use Case Diagram

Are Use-Case Diagrams enough?

The narrative and the sequence depicted are more important than the use case diagram, due to the level of detail they provide. To be useful, use cases should include as much information as possible. This need to include information that is more detailed needs to be balanced with the time and the budget limitations of the project.[4]

In Summary:

Use Case = Use Case Diagram + Use Case Narrative + Sequence Diagram

[4] You can learn more about UML Use Cases on Alistair Cockburn's site: http://alistair.cockburn.us/Use+Cases

3.Activity Diagrams

An activity diagram is used to show the different activities that need to be carried out to accomplish the goals of a system or a higher-level activity. It also shows the organization and sequencing of those activities.

The diagram below shows a fragment of a sample activity model:

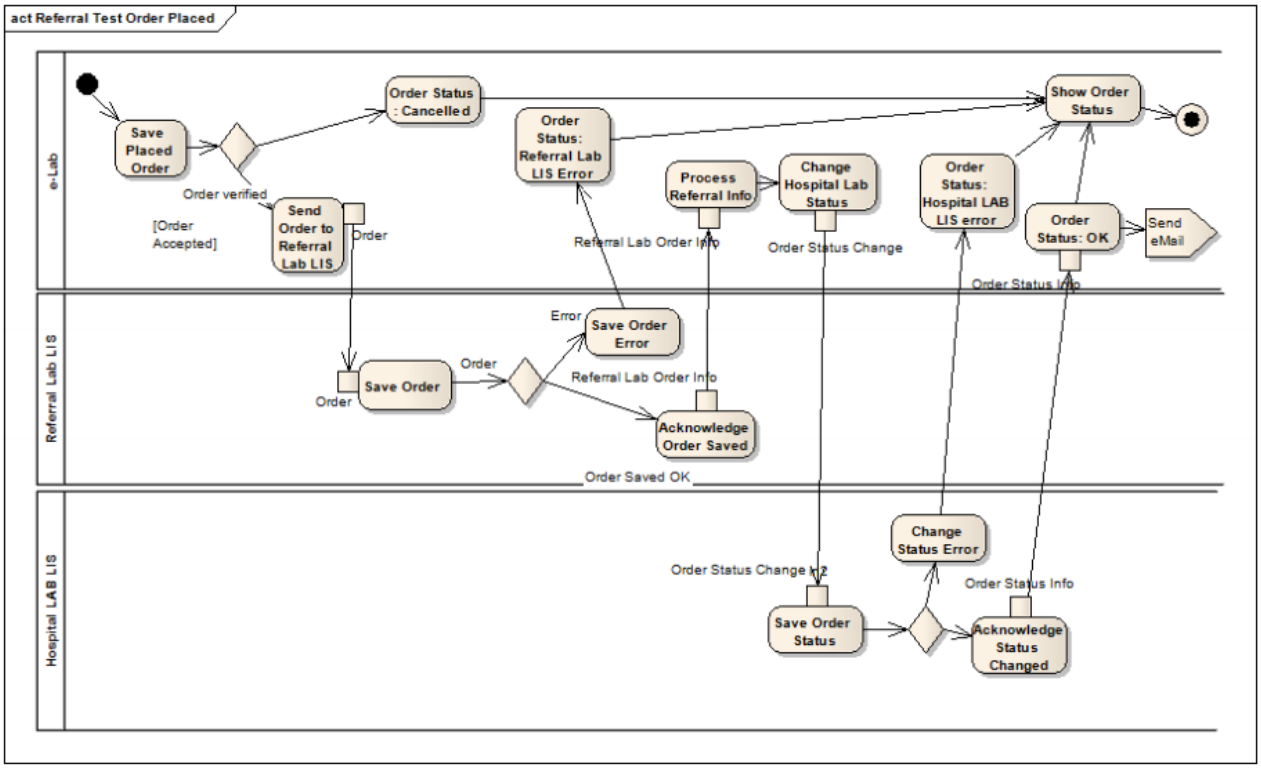


Figure 6: Activity Diagram

An Activity Diagram model shows the control flow between the involved activities. It may also show object flows and document the triggering or recognition of events. The activities may be organized into ActivityPartitions (or simply Partitions), also known as "Swim Lanes" in older versions of UML.

Activity

An activity describes a task that needs to be done. In a diagram, a UML activity appears as a rectangle with rounded edges:

Figure 7: Activity Diagram

Partitions

Partitions (known as "Swim Lanes" in older versions of UML) make it possible to partition the activities into groups. Each group represents the organization or system responsible for those activities.

On a diagram, Partitions are shown by vertically partitioning the diagram into different areas -- each area is a Partition:

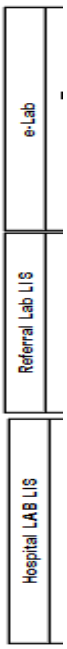


Figure 8: Partition

Control Flow

A control flow indicates the sequencing of activities and decisions. When reviewing an activity diagram, it is good practice to trace both the sequence of activities and the consequences of decisions by following the control flows. It is also important to make sure that "deadlocks"' do not occur due to improper use of splitting and merging of control flows).

In an UML diagram, control flows are shown as arrows, with the arrow coming from the originating activity and the arrowhead pointing to the destination activity:

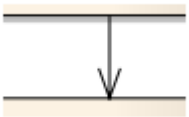


Figure 9: Control Flow

Object Flow

An object flow is a control flow that is supplemented by passing information (modeled as an object) from a sending activity that will be used by the receiving activity. In an UML diagram, an object flow is documented using a directed arrow that extends between two rectangles, representing the object or data block. As with a control flow, the arrowhead points to the receiving activity:

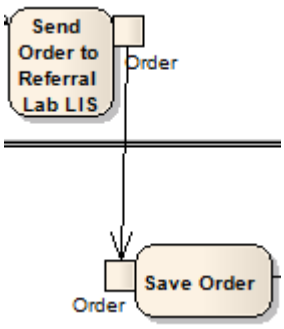


Figure 10: Object Flow

Event

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

In a diagram, a send event is documented as a rectangle with one elongated or pointed side; a receive event has a recess in one side:

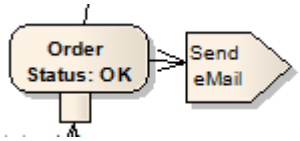


Figure 11: Event

Events and object flows are being used to do very similar things. The distinction is based on whether the information being passed goes to an activity inside or outside of the system being modeled.

Fork/Join

In some cases, it is useful to provide some control over the sequencing of control flows - in particular, to indicate that multiple control flows are either generated or processed at the same time. This structure is also known as a "synchronization bar". Forks are used to split a flow into multiple flows that lead to different activities. The point of the structure is to indicate that each receiving activity will be initiated as a result of a single outcome from the originating activity. Conversely, a join merges multiple flows into a single flow, which leads to a single activity.

In an UML diagram, a fork/join is indicated by a heavy horizontal or vertical bar to which control flows are linked:

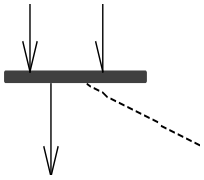


Figure 12: Fork/Join

Process Initiation

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

Figure 13: Process Initiation

Process Termination

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

Figure 14: Process Termination

4. Class Diagrams

A class model is used to organize and define the static information that is relevant to a given system or activity. A class model is depicted in a class diagram.

It is important to note that there are different perspectives from which a class model may be developed. One perspective is that of a conceptual model where the diagram represents the concepts in the domain under study however, the software that might implement them is not a determining factor in model.

A conceptual model differs from both a specifications model that defines the interfaces for software that will implement a system and an implementation model that shows the design of an object-oriented software application.

Below is an example of an UML class diagram. It shows the different components of the model for the Case Study above:

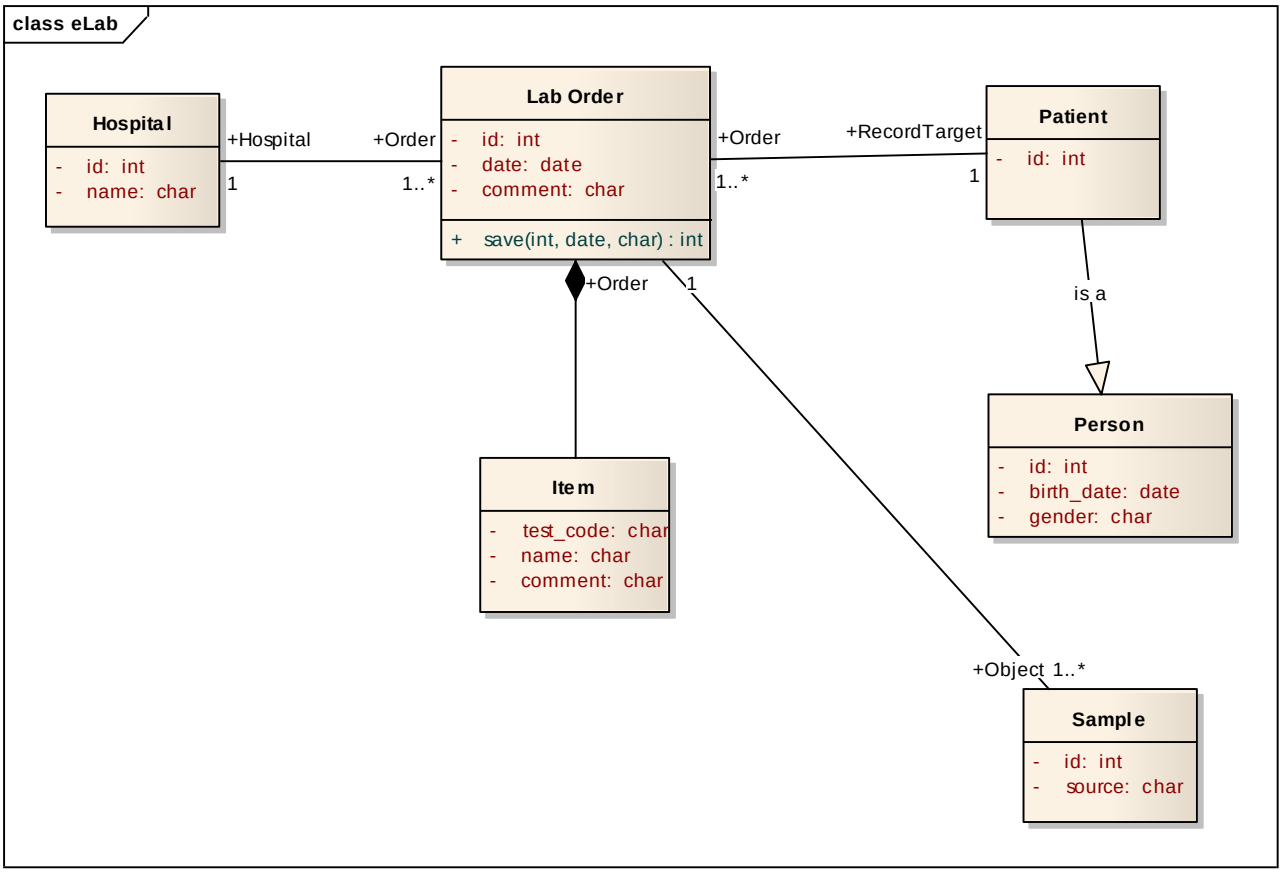


Figure 15: Class Diagram

Package

A package is used to organize a model into subsections. Within a class diagram, the name of a class may be preceded by the name of its package, with a double-colon "::" separating the two, eg "e-Lab::Hospital".

Packages are shown by a symbol resembling a file folder:

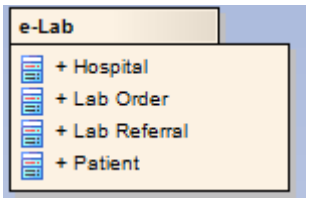


Figure 16: Package

Class

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

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

Classes can represent the people, places, roles, things and events whose information we keep in our systems.

Within a class diagram, classes are represented using a rectangle with two horizontal lines through it. The lines divide the class into three areas. The topmost area contains the name of the class, the middle one has the names of the attributes, and the bottom one contains methods or operations that can be performed on the class:

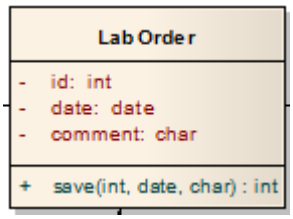


Figure 17: Class Diagram

Attribute

An UML attribute describes a particular property of a class; it indicates a particular item of information that will be valued for an instance of the class. Formally, an attribute is defined as "a

named property of a classifier that describes a range of values that instances of the property may hold"[5].

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

Figure 18: Attribute

Association

An UML association captures the relationships between instances of classes. For example, in the sample model below, there is an important relationship between patients and Lab Orders.

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

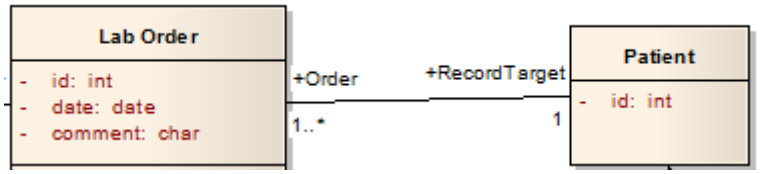


Figure 19: Attribute

Multiplicities

Multiplicities (also known as "cardinality") show the ways in which instances can participate in relationships with each other. Multiplicity is indicated by a symbol on each end of the association, where it joins to an associated class. It shows whether any valid instance of a class has to have an association, and whether there can be more than one. The multiplicity of an association for a class is evaluated by looking at the multiplicity on the far end of the association. For example, in the snippet above, the patient's record has one, two or many lab orders (e.g. a one-to-many multiplicity). A specific order must be associated with a single patient.

The multiplicity notation is interpreted as follows:

1 or **1…1** One and only one. All instances of the class must have the association.

0…1 Zero-to-one. No instance of the class can have more than one instance of the association, but it may have none.

0…* Zero-to-many. An instance of the class may have as many or few associations as desired.

[5] The Unified Modeling Language User Guide, p.458

1…* One-to-many. All instances of the class must have at least one of the associations, but it may have more

Generalization Relationship

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

Within a class diagram, generalizations are shown as a line with an arrow that points to the more general class:

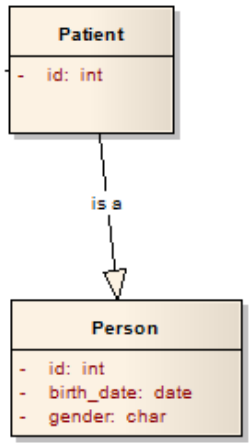


Figure 20: Generalization Relationship

Composition Relationship

The composition association captures the relationship between a whole and a related part. It is essentially an amplified association with some additional semantics:

Within a class diagram, compositions are shown as a line with a solid diamond-shaped head that points to the class representing the whole or containing object:

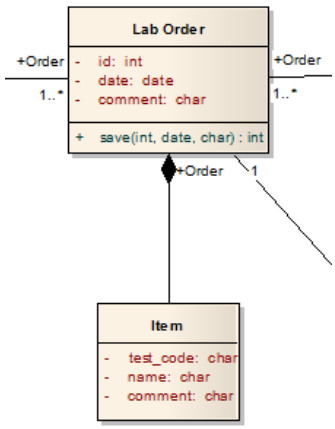


Figure 21: Composition Relationship

5.Sequence Diagrams

Overview

Sequence diagrams provide a graphical representation of object interactions over time. These typically show a user or actor, and the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single UML Use Case "scenario" or flow of events.

Sequence diagrams are an excellent way of documenting usage scenarios and both capturing required objects early in analysis and verifying object use later in design.

What are Sequence Diagrams?

They are a form of interaction diagram showing objects as lifelines running down the page, with their interactions over time represented as messages.

Sequence diagrams show which objects communicate with each other and what messages trigger those communications. Sequence diagrams are not intended for showing complex procedural logic.

Sent messages are drawn as arrows from the source lifeline to the target lifeline:

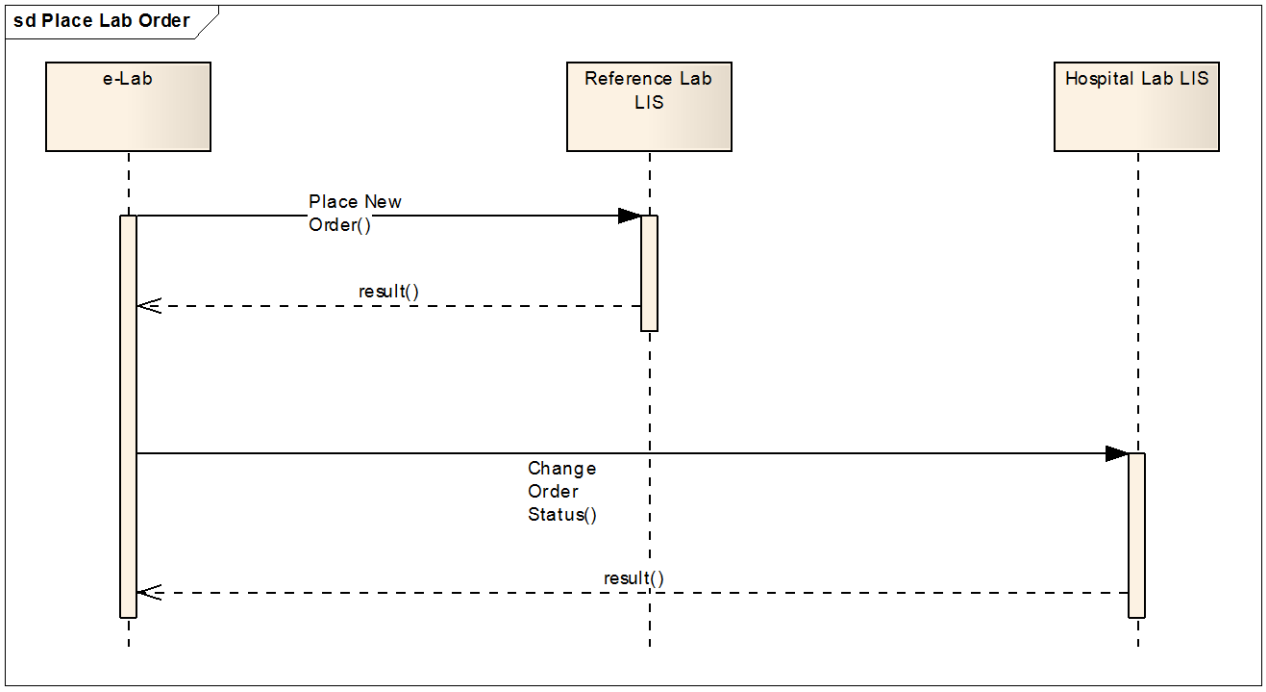


Figure 22: Sequence Diagram

Sequence Diagrams have a number of components:

Lifelines

A lifeline represents an individual participant. It has a rectangle with its object name:

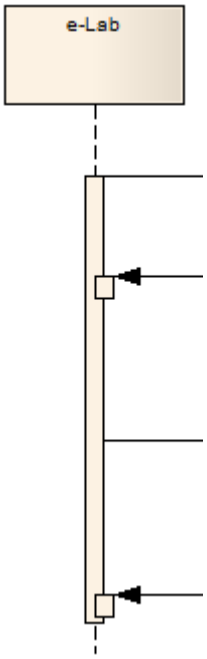


Figure 23: Lifeline

Messages

UML Messages are displayed as arrows. They can be:
Synchronous: full arrowhead
Asynchronous: line arrowhead
Complete: full line
Return: dashed line

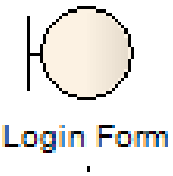
Robustness Diagrams

A robustness diagram is a simplified UML communication/collaboration diagram that uses the four types of concepts, namely:

Actors

Same concept as actors on a UML use case diagram.

Boundary Elements



Machine-to-person interface elements: Screens, reports, web pages or system interfaces that actors interact with

Control Elements

The connections between Actors and Boundary Elements - implementing the logic required to manage the various elements and their interactions. Also known as "process elements" or "controllers".

Entity Elements

Entity types or classes found in a conceptual mode.

Their purpose is to quickly gain a sense for the work that needs to be done to implement a use case because it helps to visualize the potential IT elements that need to be built.[6]

From these four types of concepts, the Robustness Diagram below can be constructed:

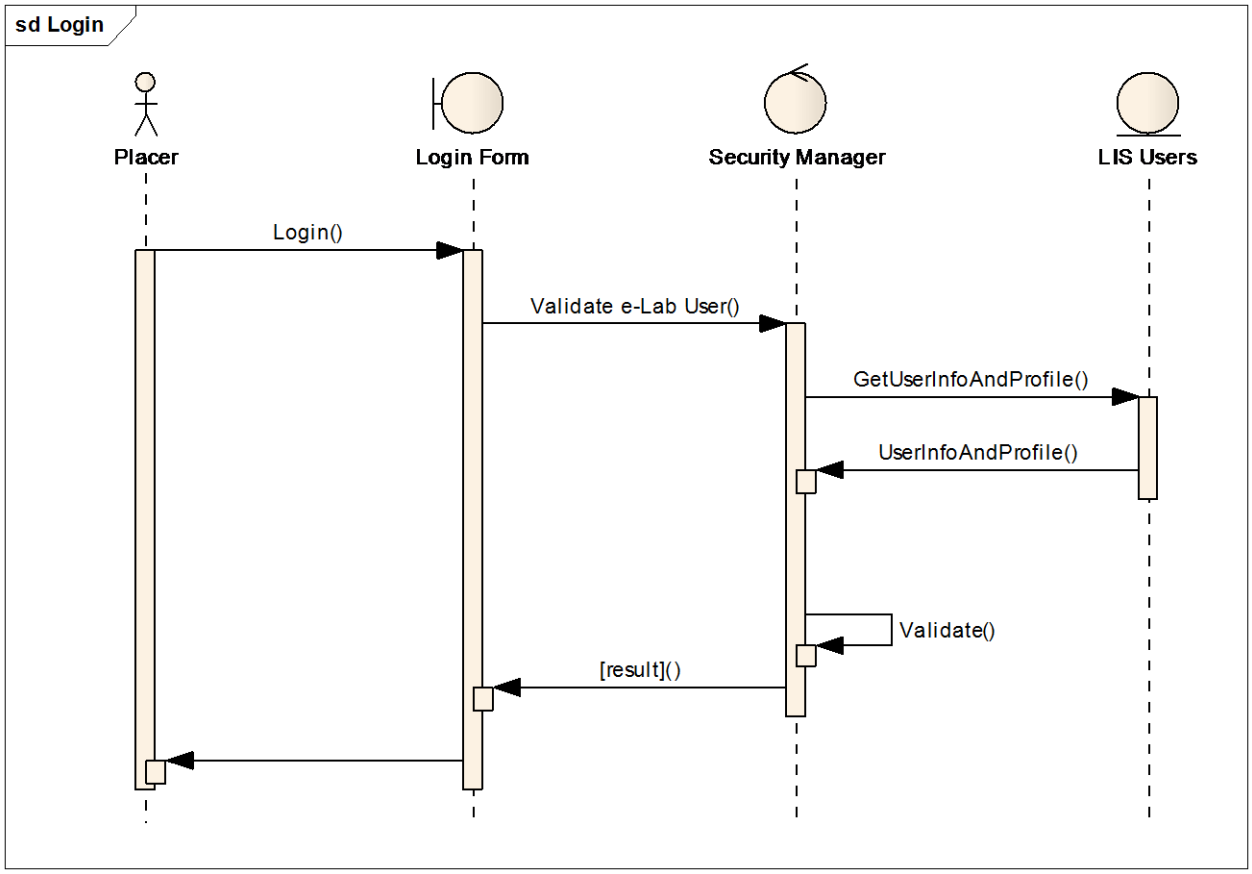


Figure 24: Robustness Diagram

[6] See more at: www.agilemodeling.com/artifacts/robustnessDiagram.htm

6.State Transition Diagrams

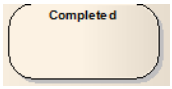
State diagrams (also called State Chart diagrams) are used to help the developer better understand any complex/unusual functionalities or business flows of specialized areas of the system. In short, State diagrams depict the dynamic behavior of the entire system, or a sub-system, and mainly, for a single object in a system. This is done with the help of **Behavioral Elements**.

A State Transition Diagram consists of the following five Behavioral Elements:

Initial State

The starting point or first activity of the flow.

State



The state of an object at an instant of time. In a state diagram, there will be multiple state symbols, one for each state of the object.

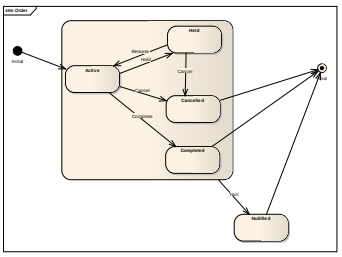
Transition

An arrow indicating the transition from one state to the other. The name of the actual trigger event or action causing the transition is written beside the arrow.

Final State

The end of the state diagram, also called a final state.

Super-State



A grouping of states that can transition to another state - a convention to make a state diagram easier to read.

From these behavioral elements, State Transition Diagrams are constructed:

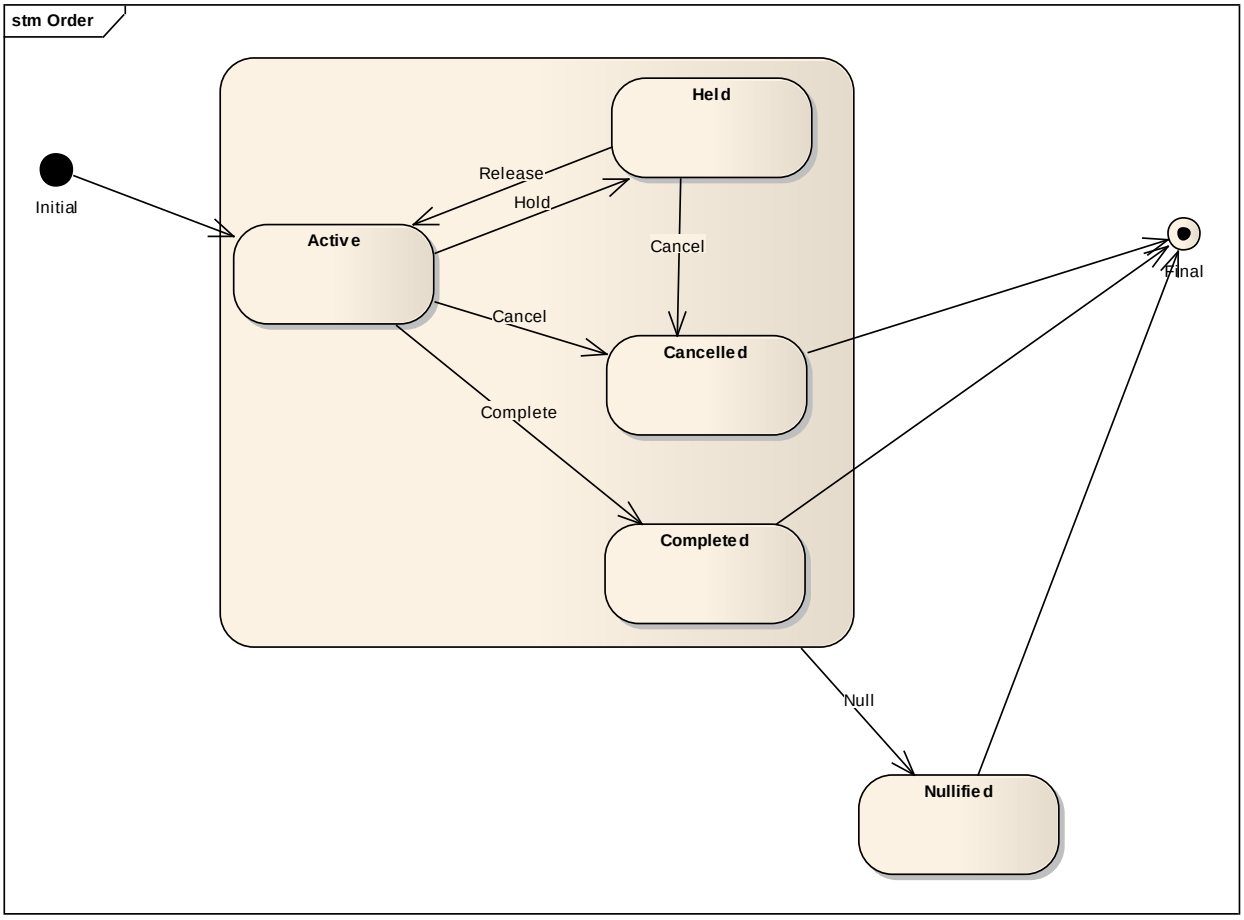
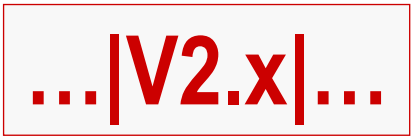


Figure 25: State Transition Diagram

7. Use of UML in HL7 V2.x

Although the development work on HL7 Version 2 started in the 1980's, today UML is used for a number of aspects of the V2.x development and implementation work.



HL7 V2.x Use Case Diagrams

In HL7 V2.x, a simplified Use Case diagram is used for Message Profiles.

The use case analysis documents the scope, involved applications (actors), the situation in which a particular exchange is triggered and the flow of events. It can combine text and a simple Use Case diagram:

Name: Goodhealth Hospital Patient Administration System

Scope: The Patient Administration System at the Goodhealth Hospital allows entering of the patient demographics at admission and the patient discharge at the end of the episode. This information is transmitted to the ancillary systems.

Actors: GHH Patient Registration System - Transmits demographic information and also the assigned bed and attending physician to ancillary systems every time a patient is admitted. It also transmits the date and final status of the patient when the discharge is registered.

GHH Laboratory Information System - Receives the information about the admitted patients and registers discharges to disallow new services.

GHH Billing System - - Receives information about the admitted and discharged patients. This allows charging of services to their accounts and closing the accounts at the end of the episode.

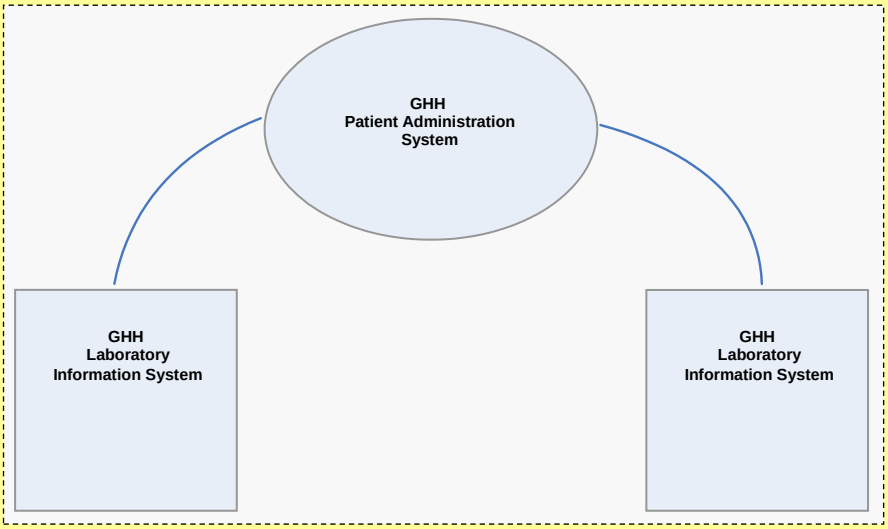


Figure 26: Simple UML Use Case Diagram for a V2.x Message Profiles

HL7 V2.x Activity Diagrams

In HL7 V2.x, Activity Diagrams are used for the Dynamic Definition of the Message Profiles.

The dynamic definition is an interaction specification for a conversation between two or more systems. It may reference one to many static definitions. The dynamic definition may include an interaction model in addition to the acknowledgement responsibilities.

HL7 V2.x Interaction Models

The Interaction Model illustrates the sequence of trigger events and resulting message flows between two or more systems. It may be in literal or graphical form. The graphical form should be a UML activity.

For example, in chapter 2 of V2.x, activity diagrams are used to document the original and enhanced message acknowledgement modes:

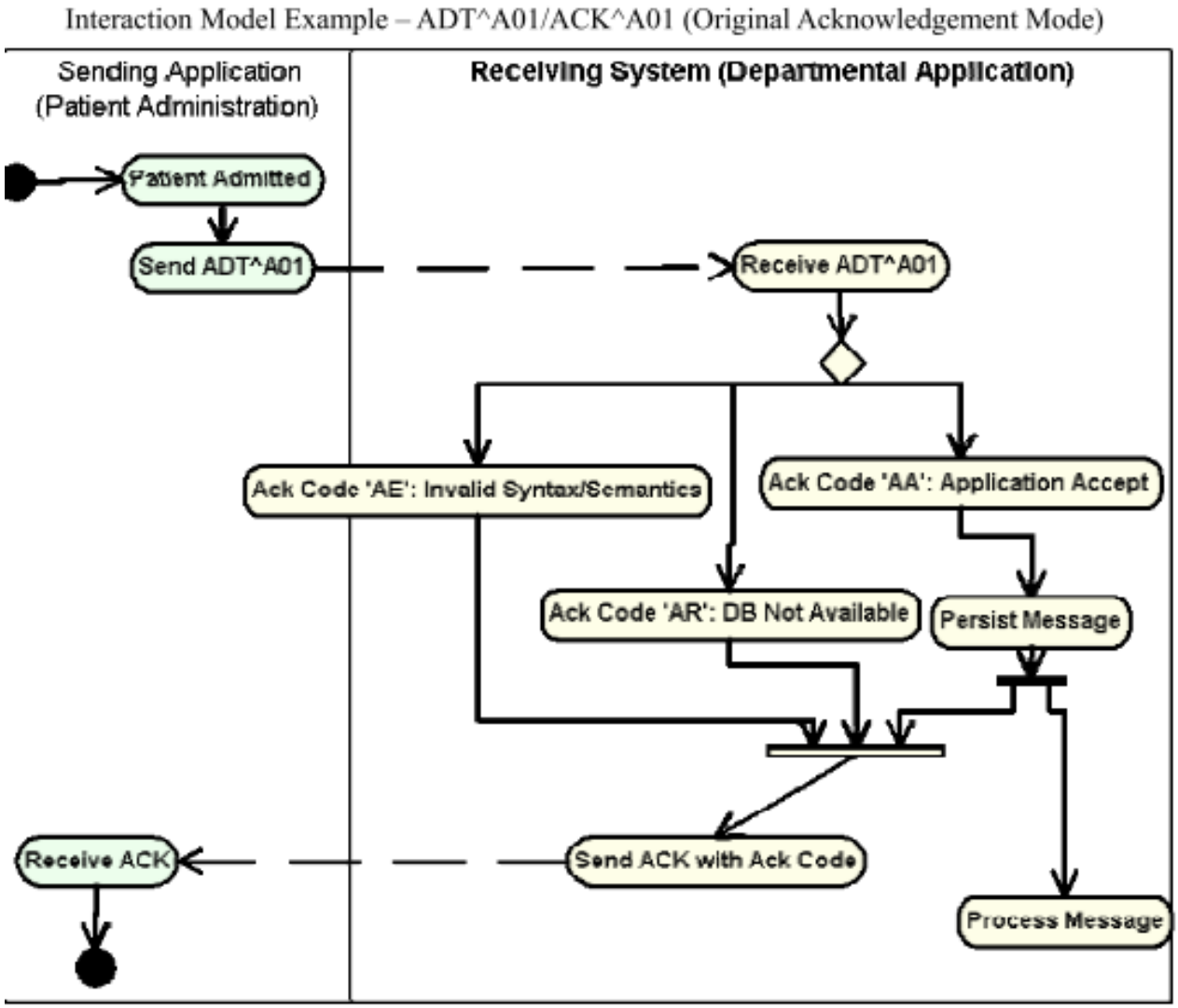


Figure 27: Example of UML HL7 V2 Activity Diagrams in V2.x Message Acknowledgment

8. Use of UML in HL7 FHIR

HL7's most recent standard is Fast Healthcare Interoperability Resources (FHIR) which became a trial standard in February 2014.[7] FHIR uses UML in the definition of its "Resources". A "Resource" is a modular component that can be combined, extended and adapted to provide practical and manageable solutions for the clinical and administrative domains solving healthcare's demand for optionality and customization.

FHIR "Resources" are described in two different ways: a UML diagram that summarizes the content, and a pseudo-XML syntax that provides a visual sense of what the end resource instances will look like in XML. Note that although the description of the resources is based on their XML representation, other representations such as JSON are equally valid.[8]

This example shows how UML is used to define the FHIR "Resource" for Patient:

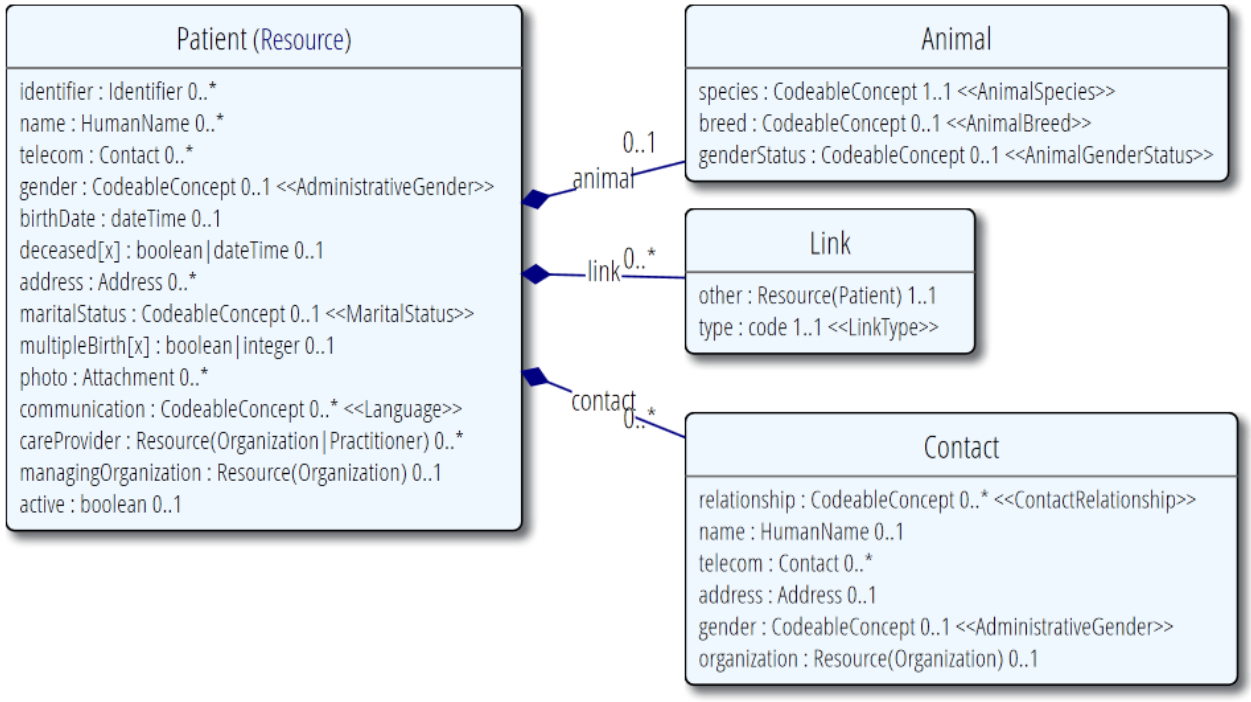


Figure 28: The UML Diagram for the HL7 FHIR Patient Resource

The FHIR UML uses three non-standard notations to show aspects of the models that cannot easily be expressed using UML:

1: Indication of Valuesets used

e.g. `gender : CodeableConcept 0..1 <<AdministrativeGender>>`

[7] See www.HL7.org/FHIR

[8] See www.HL7.org/implement/standards/fhir/formats.html

Here, the tag **<<AdministrativeGender>>** tells the user of the diagram that "gender" uses the valueset AdministrativeGender, which is a list of enumerated codes that are allowed for this property.

2: Datatype choices

e.g. **`deceased[x] : boolean|dateTime 0..1`**

In FHIR, properties are not strictly defined to have a single datatype. In the case above, the property deceased can be either a “Boolean” or a “DateTime”.

3: References (not expressed as relations in the UML!)

e.g. **`careProvider : Resource(Organization|Practitioner) 0..1`**

This expresses a *reference* to another Resource. This resource is not contained within the Patient resource, but only references another resource. In the example above, the property refers either to an Organization resource or a Practitioner resource

9. Use of UML in HL7 V3

This is only a brief introduction to the use of UML-based diagrams in HL7 Version 3. The use of UML diagrams in V3 will be described in depth in the HL7 V3 Units in this Course.

HL V3 Class Diagrams

Class diagrams in HL7 V3 are used to describe HL7 V3 concepts with their attributes and concept relationships - they are static or reference models and do not include operation definitions.

The HL7 Reference Information Model (RIM) is a reference class model --expressed as a class diagram.

The HL7 RIM has only six base or foundation classes: Act, role, entity, act relationship, role link and participation. All other classes are specializations of the RIM foundation classes. Other class diagrams derived from the RIM will be described later in the V3 Units of this Course.

HL7 also has special stereotypes based on the color or shape of the classes in some of the HL7 V3 diagrams:

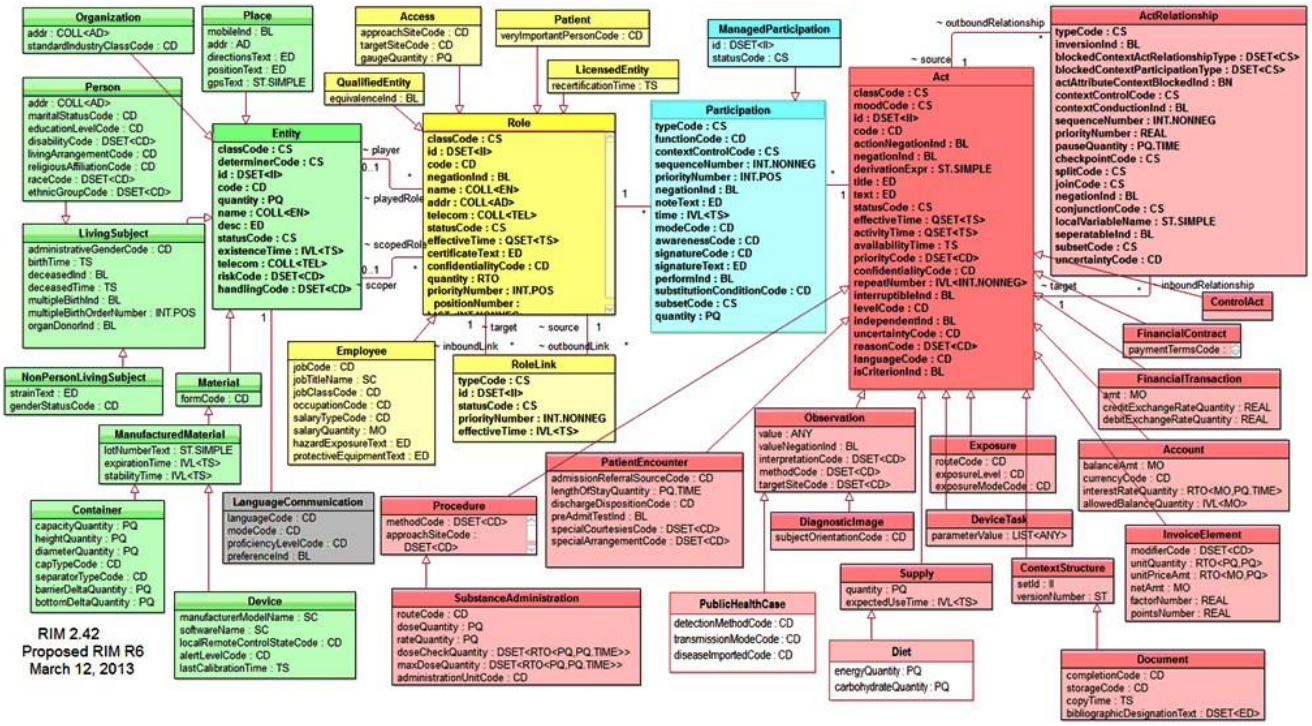


Figure 29: The HL7 Reference Information Model (RIM), a UML Class Diagram

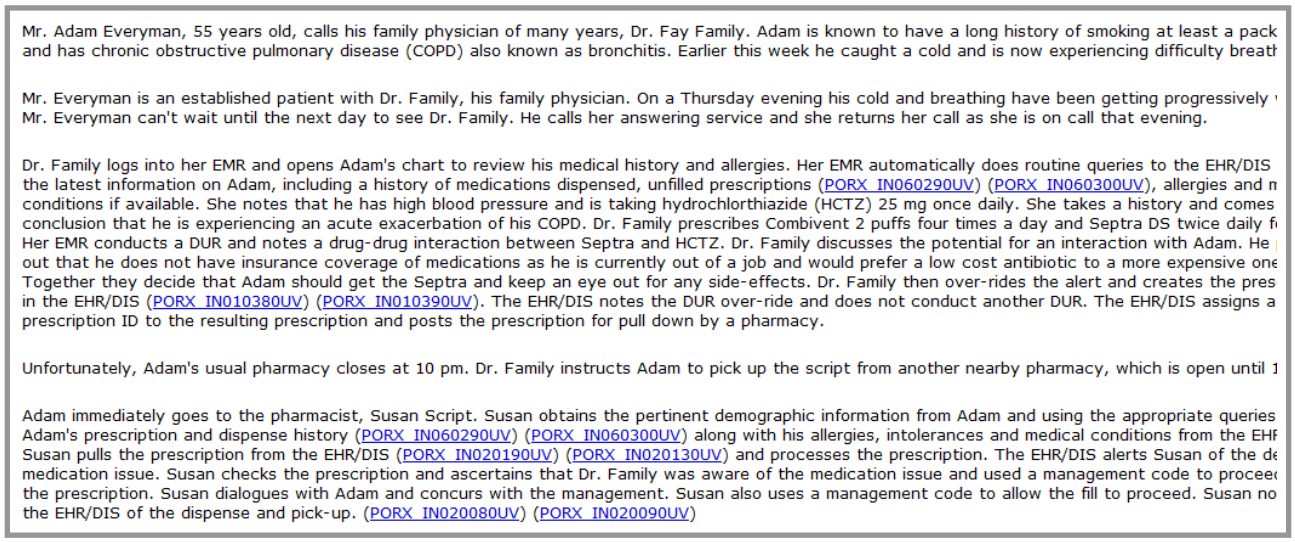
HL V3 Use Case Diagrams

HL7 does not usually generate Use Case diagrams for V3 specifications.

Requirement analysis is performed through a combination of storyboards or scenarios involving two or more healthcare applications. In the HL7 standards, the storyboards contain hyperlinks to the interaction exchange between the applications.

HL7 V3 Storyboards

Storyboards describe the scenarios in which messages are used in a specific domain:



Mr. Adam Everyman, 55 years old, calls his family physician of many years, Dr. Fay Family. Adam is known to have a long history of smoking at least a pack and has chronic obstructive pulmonary disease (COPD) also known as bronchitis. Earlier this week he caught a cold and is now experiencing difficulty breath

Mr. Everyman is an established patient with Dr. Family, his family physician. On a Thursday evening his cold and breathing have been getting progressively Mr. Everyman can't wait until the next day to see Dr. Family. He calls her answering service and she returns her call as she is on call that evening.

Dr. Family logs into her EMR and opens Adam's chart to review his medical history and allergies. Her EMR automatically does routine queries to the EHR/DIS the latest information on Adam, including a history of medications dispensed, unfilled prescriptions (PORX_IN060290UV) (PORX_IN060300UV), allergies and n conditions if available. She notes that he has high blood pressure and is taking hydrochlorthiazide (HCTZ) 25 mg once daily. She takes a history and comes conclusion that he is experiencing an acute exacerbation of his COPD. Dr. Family prescribes Combivent 2 puffs four times a day and Septra DS twice daily f Her EMR conducts a DUR and notes a drug-drug interaction between Septra and HCTZ. Dr. Family discusses the potential for an interaction with Adam. He out that he does not have insurance coverage of medications as he is currently out of a job and would prefer a low cost antibiotic to a more expensive one Together they decide that Adam should get the Septra and keep an eye out for any side-effects. Dr. Family then over-rides the alert and creates the pres in the EHR/DIS (PORX_IN010380UV) (PORX_IN010390UV). The EHR/DIS notes the DUR over-ride and does not conduct another DUR. The EHR/DIS assigns a prescription ID to the resulting prescription and posts the prescription for pull down by a pharmacy.

Unfortunately, Adam's usual pharmacy closes at 10 pm. Dr. Family instructs Adam to pick up the script from another nearby pharmacy, which is open until 1

Adam immediately goes to the pharmacist, Susan Script. Susan obtains the pertinent demographic information from Adam and using the appropriate queries Adam's prescription and dispense history (PORX_IN060290UV) (PORX_IN060300UV) along with his allergies, intolerances and medical conditions from the EHR Susan pulls the prescription from the EHR/DIS (PORX_IN020190UV) (PORX_IN020130UV) and processes the prescription. The EHR/DIS alerts Susan of the de medication issue. Susan checks the prescription and ascertains that Dr. Family was aware of the medication issue and used a management code to proceed the prescription. Susan dialogues with Adam and concurs with the management. Susan also uses a management code to allow the fill to proceed. Susan no the EHR/DIS of the dispense and pick-up. (PORX_IN020080UV) (PORX_IN020090UV)

Figure 30 HL7 V3 Storyboard

HL7 V3 Sequence Diagrams

UML Sequence Diagram Lifelines in HL7 V3 sequence diagrams represent the Applications (in fact, they represent an abstraction of the actual applications called "Application Roles").

UML Sequence Diagram **Messages** in HL7 V3 sequence diagrams represent the **Interactions** - the interactions are not only the complete messages and their responses, but also their wrappers (information about the message) and the implied object state transition. Again, the actual user (human actor) is never shown (all actors are application roles):

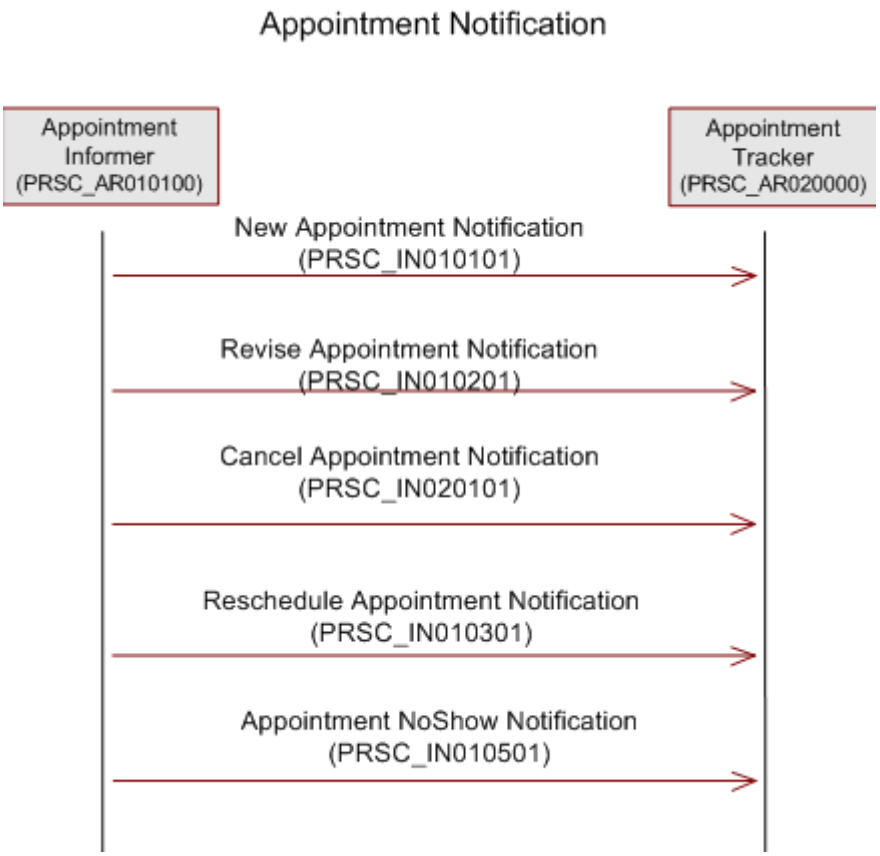


Figure 31 HL7 V3 Sequence Diagram

HL7 V3 State Machine Diagrams

Almost all interactions in HL7 V3 messaging involve transmitting from one application to another the change in the "status" of a specific act.

This is why HL7 V3 interaction models are centered or focused on acts.

Usually, a V3 message talks about a certain specialization of act (observation, procedure, encounter, etc.), and a certain change in the status of this act (active, held, completed, scheduled, etc.).

Other foundation classes (entity, role, participation) also have their associated state machine diagram.

Thus, the state machine diagram is a very important artifact for understanding V3 messaging.

Below is the entire state machine diagram for the Act class. Each specialization of act can go through a different subset of these states:

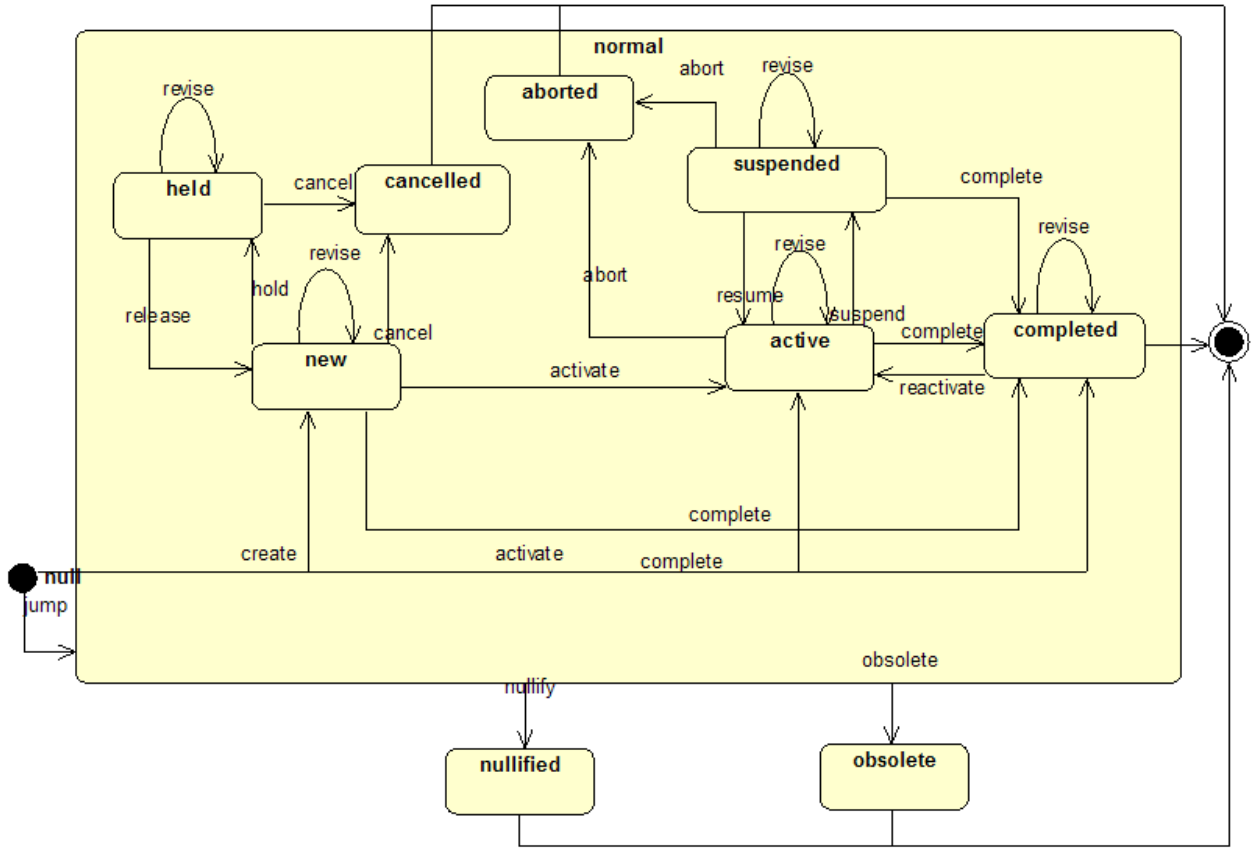


Figure 32 HL7 V3 State Diagram for "Act"

10. Use of UML in HL7 Domain Analysis Models (DAM)

The HL7 Domain Analysis Model (DAM) is used as the basis for specification development.

The specification may be a functional model, a service specification, a message definition, etc. depending on the type of standard desired.

The specification may be directly based on models and on the contents (the information and operations) specified in the DAM or derived from it through mappings or transformations.

Both the DAM and standard specifications will rely on models and diagrams to both manage the contents and to provide views of the information and interactions between systems.

The DAM is constructed to model a clinical specialty area of interest (the "domain"). The term "domain" indicates that the semantics of the model are restricted to those that collectively define a clearly bounded domain (area) of interest. Because clinical domains are centered on patient care, the domain model represents both the static and dynamic nature of healthcare, including a UML class model and an activity model.

The following UML models are used to specify a DAM.

HL7 DAM Use Case Diagrams

In HL7 standards development, use cases are the starting point providing specific examples of the data that needs to be exchanged, scenarios in which data are exchanged, roles associated with data exchange and expected responses.

For example, in the cardiovascular and tuberculosis DAMs, the use cases were selected and created to scope and define an area of interest, not to drive message specification. As a result, the requirement of use cases for domain modeling has the potential to overly restrict the scope. In the cardiovascular use case - reporting to a registry - represents only a narrow slice of the domain.

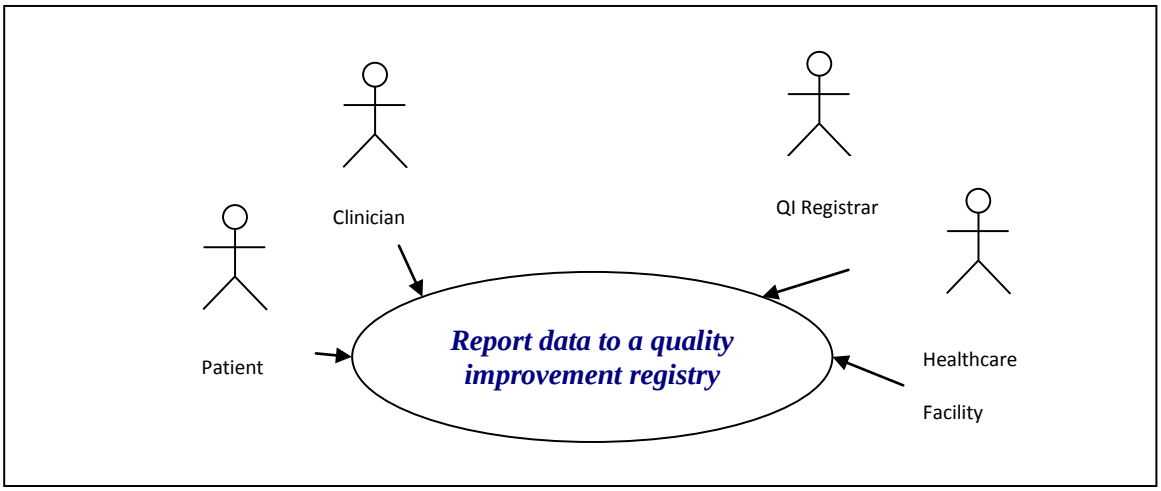


Figure 33 UML Use Case Diagram for an HL7 Domain Analysis Model (DAM)

As shown in the figure above, the stick figures represent participants in the process in the use cases. The process (activities of the use case) or process goal ("Report data to a quality improvement registry") is described in the central oval. With many processes, there are multiple variations, including different inputs, process paths or options, and different outcomes. In the tuberculosis project mentioned above, the variations include availability of diagnostic capabilities, healthcare setting, chief complaint, signs and symptoms, co-morbidities, disease history and treatment interruptions. These variations are described in numerous scenarios/storyboards supporting the use case.

The use cases and associated scenarios/storyboards contain administrative details (for example, assignment of a patient to an isolation room) and research details (for example, informed consent process) not covered in the class model.

HL7 DAM Activity Model Diagrams

Activity models depict processes, the dynamic aspect of the domain. Defining a clinical domain is facilitated by the depiction of the clinical processes included in the domain. The included clinical processes, in part, define the domain. Activity models were not initially planned; however, very early in the process, we needed a way to communicate the scope of the project. Defining the scope by the clinical processes we wished to cover proved helpful. It facilitated understanding by both technologists and clinicians. As such, we feel that activity models should be included in the definition of a clinical domain.

Figure 40 shows the TB activity model. The clinical processes included are patient presentation, initial workup and diagnosis. The UML activity model can be read like a flowchart:

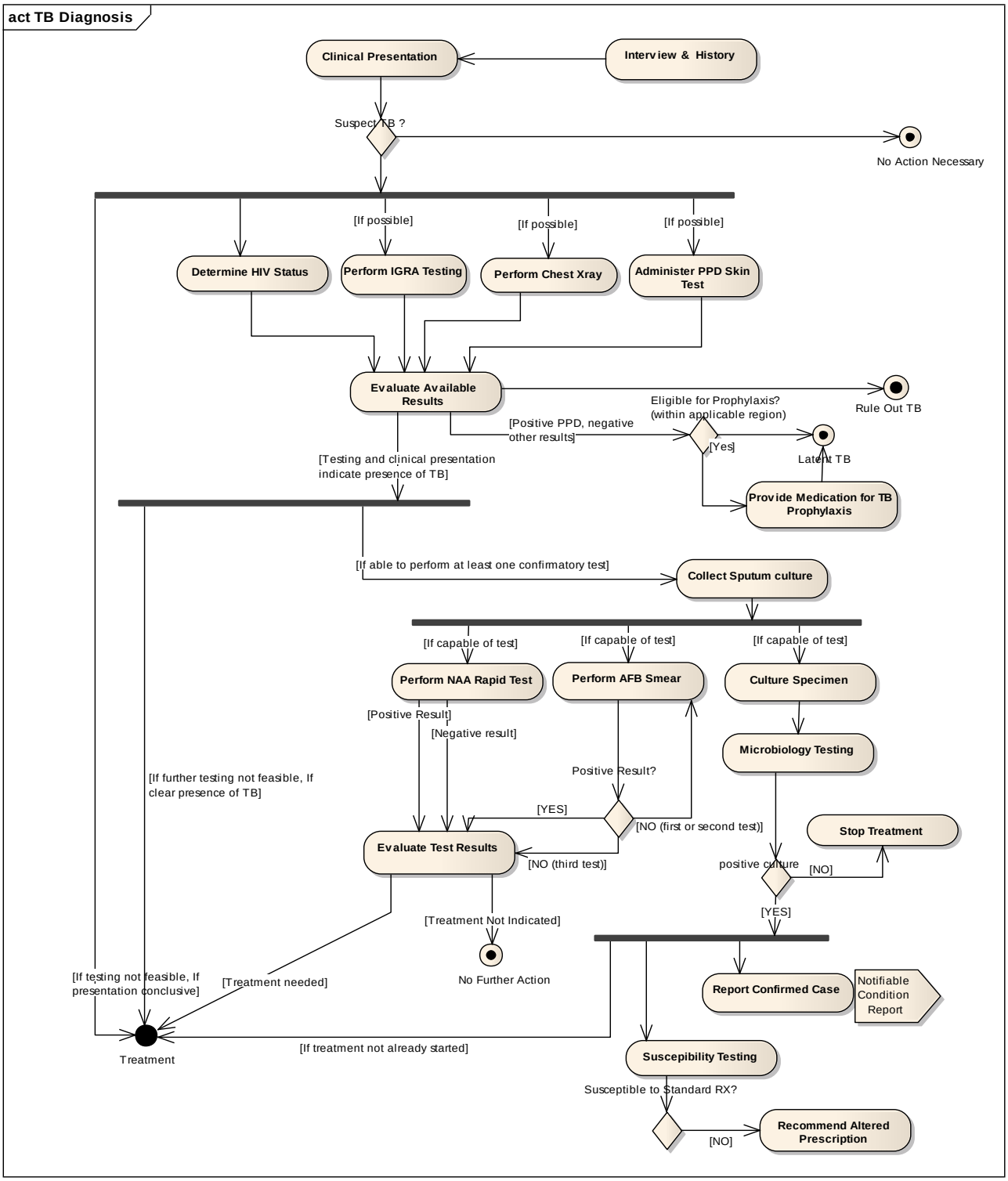


Figure 34 Tuberculosis Registry DAM Activity Diagram

HL7 DAM - Class Model

For a clinical domain, the class model depicts the things about which data are captured and structurally how those data are related to other data in the domain; e.g. for each patient encounter, the following fields are captured, and signs and symptoms exist in a "none to many" relationships with an encounter.

Therefore, a class model depicts the static content of the domain.

Class models can be created at different levels of detail. This creates a significant challenge for maintaining consistency across clinical domains. In addition, significant effort was devoted to deciding the level at which to model. A high-level (not detailed) class model would describe the categories of data in a domain.

It is important to note that a domain model is intended to serve as a common point of understanding and dialogue between clinicians and information technologists. As such, the terminology used in the domain models is that used in the clinical setting. This representation is not that of the HL7 RIM with acts, roles, entities, codes, moods, etc. It is likely that there will be significant utility in representations of the domain using the HL7 RIM, data types and associated terminology as the need for message specifications arises.

The cardiovascular class model is provided in the figure below. A class represents a grouping of similar data elements - for example, catheterization lab procedure or medical history:

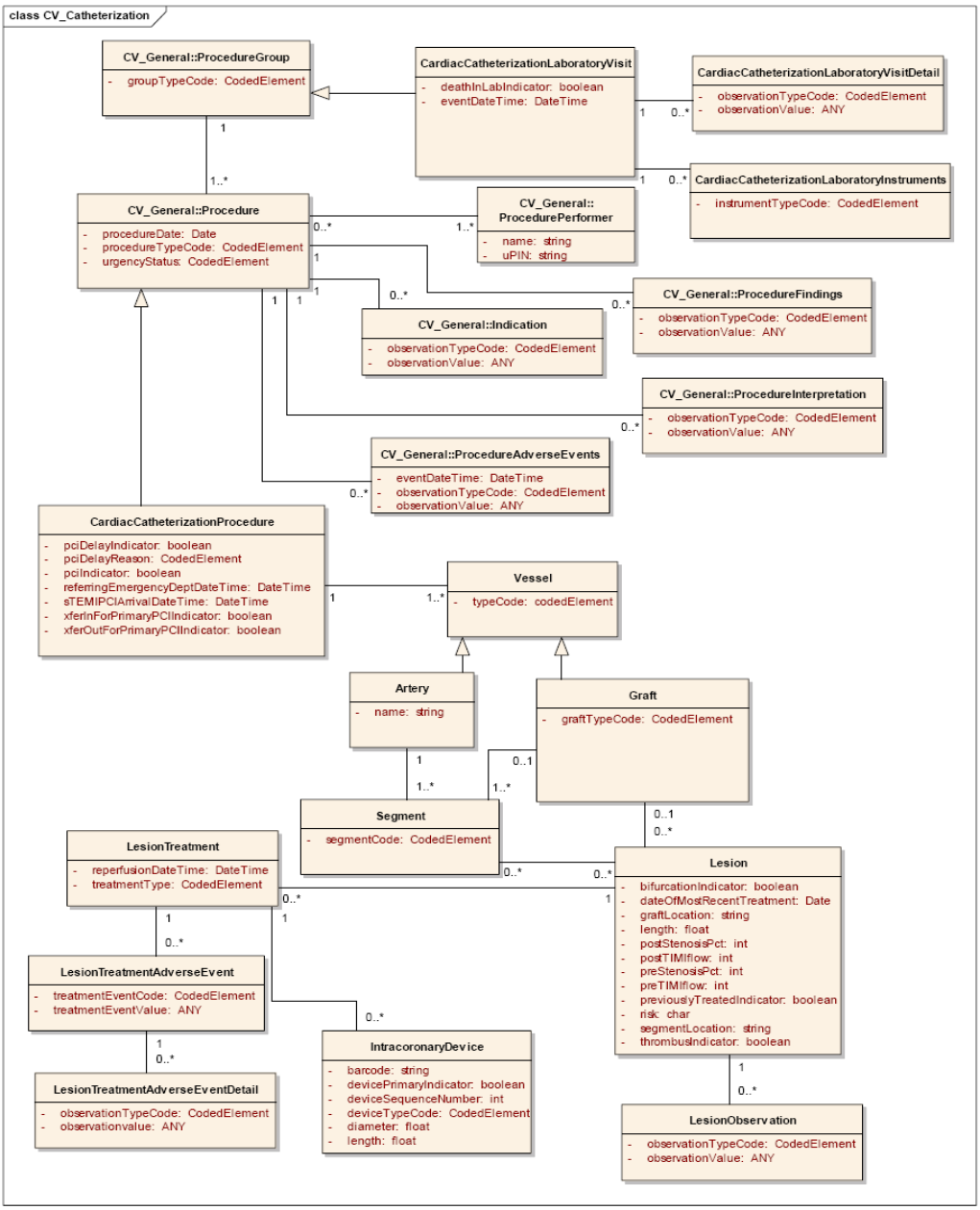


Figure 35 Cardiovascular DAM Class Model

11. Use of UML in CDA

Overview

Like all other HL7 standards, the HL7 Clinical Document Architecture (CDA) makes extensive use of UML. An example of this is the Model Driven Health Tooling (MDHT) project of Open Health Tools. The core principle is that CDA models are created that align with the general UML 2.0 specifications and modeling semantics. Whenever possible, UML constructs are used to represent CDA template structures and semantics. Only as a last step are these extended using UML profile stereotypes, which are also applied as recommended by the UML 2.0 specifications.

Each modeling construct includes a list of CDA profile stereotypes that define additional metadata required for complete CDA template specification. These metadata are not captured by the base UML construct, so it is extended using stereotype properties. The stereotypes and properties are defined in a separate profile specification.[9]

CDA Template Classes

- Each CDA template is represented using one UML Class.
- Each CDA template has exactly one template ID assigned, which serves to identify use of this template in a CDA document instance.
- The template ID is defined in the class metadata, using a UML stereotype. All model relationships and constraints refer to the template class names, not their IDs. A template ID should never be used when specifying constraints or template relationships.
- A "conforms to" rule, also called "implies", is represented using a UML Generalization, i.e. a template class has another template or CDA type as its superclass.
  - Direct conformance: immediate parent
  - Indirect conformance: parent of a parent
- Each CDA template class must conform to exactly one CDA base class, either as direct superclass or indirectly via a parent template. For example, a template conforms to CDA Section or Observation.
- All conformance rules from all superclasses are inherited into each template.
  - A template class may redefine (or, replace) conformance rules inherited from superclass template. Any class attribute having the same name as an attribute from a parent template redefines that inherited rule.
- A template class may have multiple immediate parent classes, often called "multiple inheritance", such that the template conforms to two or more independent templates, plus their superclass templates. Although multiple inheritance is possible, it should be used with extreme caution.

[9] See www.projects.openhealthtools.org/sf/projects/mdht/cda/doc/MDHTModelingStyleGuide.pdf

The Categories of Template Classes

Most CDA templates fall into three categories, with a small number of examples in the fourth category. The published CDA implementation guides use these four categories as primary sections of the guide.

- Document
  - Header constraints on subclasses CDA Clinical Document
- Section
  - Subtypes of CDA Section
- Clinical Statement
  - Subtypes include: Act, Observation, Encounter, Organizer, Supply and others
- Other Classes
  - Classes within CDA header, e.g. Patient
  - Classes derived from ActRelationship or Participation

The HL7 CDA Refined Message Information Model (RMIM)

CDA references the HL7 RIM to express clinical content; its RMIM is derived from the RIM is a Static Model:

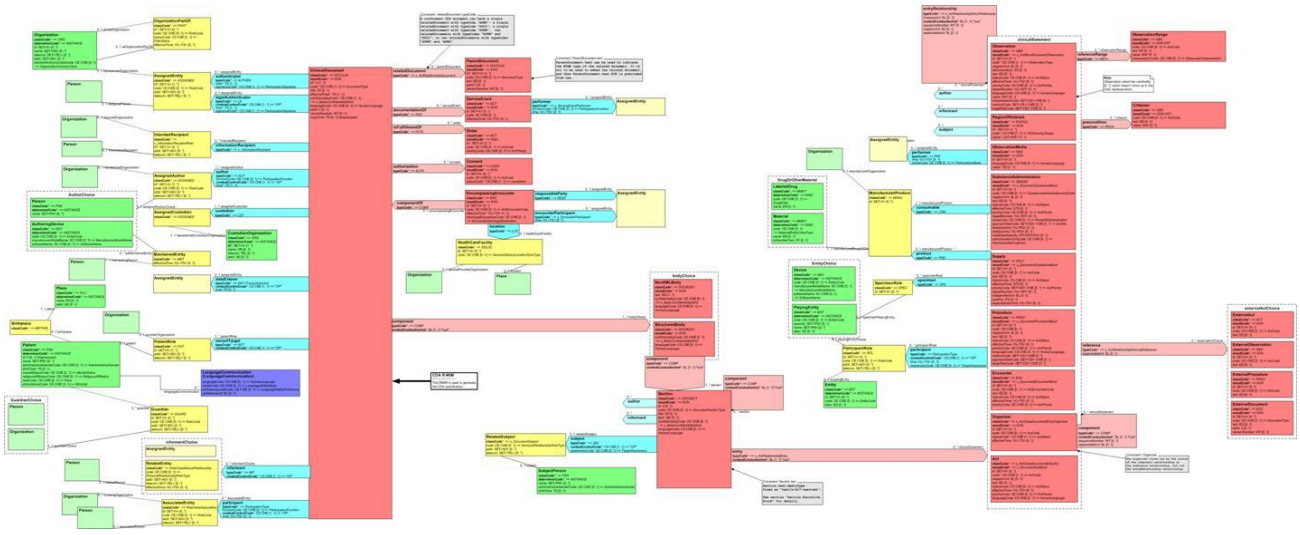


Figure 36 The HL7 CDA Refined Message Information Model (RMIM)

Additional Reading Material

The best reference is OMG's web site at www.omg.org/gettingstarted/what_is_uml.htm

A full UML tutorial is available at www.sparxsystems.com.au/UML_Tutorial.htm

A full list of UML tools is available at http://uml-directory.omg.org/vendor/list.htm

Other References:

Booch G, Rumbaugh J, Jacobson I. *The Unified Modeling Language User Guide*. Indianapolis, IN: Addison-Wesley Professional; 1999.

Fowler M, Scott K. *UML Distilled: Applying the Standard Object Modeling Language*. Indianapolis, IN: Addison-Wesley Professional, 1997.

Health Level Seven, Clinical Interoperability Council (CIC) Working Group, -Handbook for Developing a Domain Analysis Model - download from:

www.HL7.org/documentcenter/public_temp_E702D926-1C23-BA17-0C4DB69EAB4C40F3/wg/cic/2012_01%20CIC%20DAM%20Development%20Guide%20.pdf

Use Cases: http://alistair.cockburn.us/Use+Cases

Robustness Diagrams: www.agilemodeling.com/artifacts/robustnessDiagram.htm