TABLE OF CONTENTS

1. INTRODUCTION

This document is an attempt to describe a consistent set of changes to the methodology drawing from various venues (the August MDF, the Sunday Evening Compromise, individual discussions with various people.)

It includes:

- a summary of the changes
- a fairly rich example (pharmacy ordering and administration) and document some issues about the methodology.

The example is not expected to be a correct analysis of the domain (it is surely wrong in places and no more than half of a complete analysis.) Nonetheless it is adequate to challenge the methodology and begin to validate it.

The reader is expected to be fairly familiar with the current HL7 Message Development Framework Document (MDF, dated August, 1996).

Readers may want to refer to the graphics in these sections as they are reading the rest of the document.

- section 3.5, a rough data flow diagram
- section 3.6, a meta-model illustration for the topics in this document.

There are footnotes throughout this document that are used to move less relevant topics out of the main flow.

2. DEFINITIONS

The following table defines terms used in this document. It is hoped that each term is used in a way that is generally consistent with its general use in information systems, but the table specifically defines how the term is used in the HL7 Version 3 Methodology. As such each definition should be preceded by the words "In HL7 Version 3, ...".

Italicized words in a definition are defined elsewhere in the table.

Application Role Class Behavior Diagram	A class diagram featuring only the *application role classes* and featuring only connections that correspond to *interactions*.
behavior analysis	An analysis of the *life cycle* of a *class*.
business event	A *trigger event* that has special interest as the first trigger event in a *sce-nario.*
class	An object in the meta-model that most closely resembles *Class* in most meta-models. It represents a template for an object and can contain attrib-utes or behaviors. See *data class*, *subject class,* and *application role class.*
class, application role	A specialization of *class* that is designated as representing an interface of an HL7 component. From a functional point of view it primarily represents behaviors. These behaviors correspond to the ability to receive an interac-tions from various other application role classes.
conformance claim	An objective statement that a healthcare information system exactly follows certain of the contents of a version of an HL7 standard. See *application role conformance claim* and *textual conformance claim*.
conformance claim, appli-cation role	A *conformance claim* that is a statement that a healthcare information sys-tem conforms to all the behaviors of an application role classes, i.e., that it sends and receives all the interactions which the application role class sends or receives, using the message formats specified with the interactions, and follows the other specifications for an interactions.
conformance claim, textual	A *conformance claim* that uses unambiguous text the describe conformance to portions of HL7 that are not specified by an *application role confor-mance claim.*
class, data	A specialization of *class* that is most like the definition of "class" used by the Joint Working Group. It contains no behaviors.
interaction	An instance of the sending of an HL7 message from a single *application role class* to another application role class in response to a trigger event or the receiver responsibility of another interaction. Interactions include the specification of a *message format*, a receiver application role class, a sender application role class, preconditions and post-conditions for sending the interaction, sender responsibilities, and receiver responsibilities.
life cycle	A description of all the states of a class and the transitions that may occur.
message format	The exact information contained in a message as described by the hierarchi-cal message description. The same message format may be used for many *interactions.*
message trace diagram	A diagrammatic representation of a *scenario* in terms
scenario	An ordered collection of interactions that represents a typical sequence in which they may be executed. Each scenario may be displayed in terms of the use cases, or in terms of a subset of the *interaction model.*
state	A characteristic of a class that characterizes a step in the its life cycle. All states are represented by discrete values of a single enumerated variable in

	the class.
state transition	A recognized transition from one state of a class to another state or back to the same state. These transitions are closely associated with *trigger events.*
state transition diagram	A diagram that represents the *life cycle* of a class.
subject class	A specialization of *data class* that has been designated by a Technical Committee as interesting.
trigger event	An event within the processes of healthcare which, when recorded or recognized by a healthcare information system, creates the need for an information flow to on or more other healthcare information systems. The information may flow may be implemented by one or more *interactions*. Trigger events are closely associated with *state transitions* **and** *leaf-level use cases.*
use case	A description of a process by which actors do things that lead to the need. Use cases may break down into component use cases. A use case may appear as a component in several other use cases. When a use case contains component use cases, the order in which they occur is not specified. See *scenario.*
use case model	A definition of the functional requirements for HL7 messages that consists of *actors*, *use cases*, and connections between *actors* and use cases.
use case, leaf-level	A *use case* that does not break down into components. Leaf-level use cases are closely related to *trigger events.*

1

3. SUMMARY OF METHODOLOGY

There are four major models: use case, information, interaction, and hierarchical message description.

3.1. Use Case Model

3.1.1. Issues in Developing the Use Case Model

The Use Case Model includes use cases. The "stopping rule" for building use cases is changed to "one can split a use case if the resultant use cases represent events that might occur at discrete times, or if they will separate the roles of the actors in the initial use case."[1]

With one exception, actors cannot represent information systems. The exception is a special actor with the literal name "some information system". The term "some system" is chosen to reinforce the notion that the use cases are not built on a priori knowledge of the functionality of specific healthcare information systems.

Figure AA represents an example using the current HL7 iconization. It is a fragment of a use case that might be constructed for lab test processing. The premise is that the order may be entered by a person or may be scheduled directly by the lab system for certain kinds of reflexive orders. The order lab test use case is handled by including "Some System" as an actor, along with Authorized Clinical User.

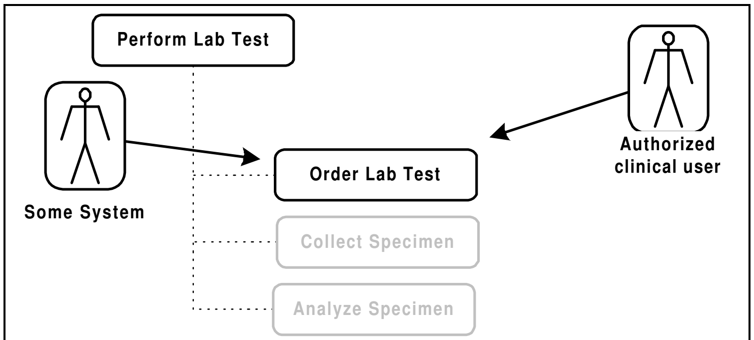

Figure AA. Sample use of "Some System" actor.

[1] As Jake has pointed out before, there is a need for aggressive facilitation when users are choosing actors. Users will have a natural tendency to want to use the actors to construct an entire model of professional responsibilities by having many actors with subtle distinctions (e.g., nurse practitioner vs. physician's assistant.) In this respect the old use case stopping rule was better. It said "stop if splitting a use case doesn't generate a new information flow." Thinking of the downstream message was a powerful way to guide the group away from creating numerously, finely distinguished actors. The approach to naming actors used in these examples, by following the pattern "authorized XXX user" can help to alleviate this problem. It can also mask distinctions that have an important impact on the workflow as may happen when either the phlebotomist or the authorized caregiver draws blood. These distinctions will frequently alter user interface design and may occasionally be so significant that they alter the HL7 messages.

The rule for looking for functions that occur at different points in time is also a two-edged sword. It can be a powerful aid for identifying trigger events, since the separation in time suggests a need for an information flow. It can also devolve into an analysis of UI-related issues very easily.

The best facilitation practice seems to be, encourage the committee to err on the side of too few actors in an initial use case analysis. Revisit the issue when Application Role Classes have been determined.

The fact that the use case "Order Lab Test" has two actors might lead one to suspect that it should be further broken down. Figure AB represents how that breakdown might be represented.

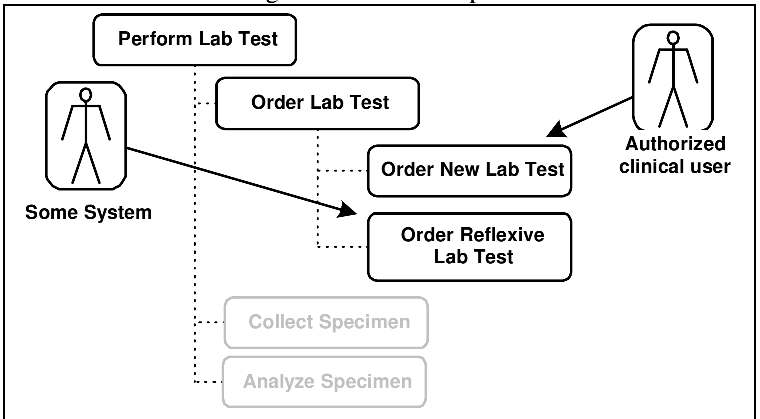

Figure AB. Further breakdown to achieve

Any use case (especially composites) can be used to describe functions that may later be used to determine receiver or sender responsibilities.

3.1.2. Identification of Trigger Events

Each leaf level use case should be regarded as a candidate for a trigger event. It will not be possible to begin to resolve the final choice of trigger events until later in the analysis.

3.1.3. Use of Scenarios

Users should construct scenarios during the initial preparation of the Use Cases. For meta-methodological reasons, these are considered part of the Interface Model, but they can be represented solely in terms of Use Cases. Scenarios represent the first step where the technical committee can begin to create sequential representations of the use cases. It will not be practical to create all the combinations of use cases that may be valid, but the process of doing so may help the technical committee to

Business Events are trigger events that are given special status as the initial trigger event in a scenario. They are best finalized during the construction of the Interaction Model.

3.2. Information Model

The Information Model includes three kinds of classes

1. *data classes* that represent healthcare data objects; these classes do not show behaviors.
2. selected data classes that are designated as *subject classes*, meaning that they have been chosen as the subject of a set of related use cases.
3. *application role classes* which are abstractions that name roles that may be played by healthcare information systems when sending or receiving HL7 messages. Application Role Classes represent behaviors, but little or no interesting data.[2]

For the Subject Classes a life cycle analysis is required. It describes a set of "states" for a class. States are represented by values of a specific enumerated attribute of the class. The model further describes the allowable paths from one state to another. These are called "state transitions." State transitions are associated with trigger events. Through trigger events a state transition is associated with one or more leaf-level use cases.

A state transition does not always represent a change in the value of the enumerated state attribute. For example, suppose a pharmacy order describes three medications. There may be a change in the order item states where the second medication order is now filled. This might generate a state transition in the pharmacy order from "partially filled" to "partially filled."

There is no requirement that state transition analysis only be performed on Subject Classes. However it is the state transitions of Subject Classes that are related to trigger events.

[2] It is debatable whether application role classes belong in something called the "information model". They certainly represent a different subject area, in the Coad-Yourdon sense of "subject area".

State analysis is performed for all subject classes. Each state transition must be associated with one use case. However multiple state transitions may be associated with the same use case.

The technical committee will create for each subject class, one or more Application Role Classes. In general these classes will have a stereotyped class name consisting of <subject class> followed by <relationship>. It may create one or more of the following relationships, or it may create different relationships as discussed below.

a) <subject class> Manager normally associated with an Application Role Class that manages the life cycle of the subject class. In terms of the above example it would correspond to the "pharmacy order filler" role.

b) <subject class> Creator normally associated with an application role of a user of the Manager class. In terms of the above example it would correspond to the "order sender" role.

c) <subject class> Tracker normally associated with an application role associated with tracking the life cycle of the subject class. In terms of the above example it would correspond to those aspects of the "electronic medical record" role that deal with Pharmacy data. It is expected that this class will routinely describe interactions associated with state changes in the subject class.

d) <subject class> Historian normally associated with an application role associated with tracking the final state of the subject class. In terms of the above example it would correspond to those aspects of the "clinical data repository" role that deal with Pharmacy data. It would routinely expect to get a summary of the entire order when it entered a final state or if it was updated after it reached the final state.

There is no requirement that a technical committee create all of these "<subject class><relationship>" classes. Neither is limited to the possibilities in (a) - (d). But it is expected that the committee will consider all of (a) - (d). This list of probable stereotyped <relationship> roles may expand as we get more experience.

It is expected that any other Application Role Classes not on this list will be defined in terms of the formula "<subject class><relationship>" rather than with more specific names. We would not expect the pharmacy technical committee (or any other) to define a class like "pharmacy robot" or "electronic medical record."

3.3. The Interaction Model

The Interaction Model is a collection of all Interactions and Scenarios. The definition of an Interaction is essentially unchanged since the last MDF although the words are somewhat different here.

3.3.1. Interactions

The notion of "Interaction" is unchanged from the last MDF. It is an abstraction that represents a message flowing between two systems in a specific context. That is it is the confluence of

- a sending Application Role Class,
- a receiving Application Role Class, which must be different than the sending Application Role Class
- a stimulus (either a trigger event or a sender or receiver responsibility in another Interaction)
- sender responsibilities
- receiver responsibilities,

- a Message Format[3] and
- preconditions and post-conditions on the sending of the message.

The same Message Format may be used in many interactions.

Each use case that represents a trigger event is represented in the Interaction Model by one or more interactions.

3.3.2. Scenarios

A Scenario is an ordered collection of interactions that represents a typical sequence in which they may be executed. Each scenario may be displayed in terms of the use cases, or in terms of a subset of the interaction model.[4]

In the latter case the representations includes all interactions that are implied by the trigger events associated with the use cases or implied by the sender and receiver responsibilities of those interactions.

When preparing a *message trace diagram* of a scenario, the technical committee may choose to eliminate some application role classes for clarity. All interactions that are in the scenario that involve those application role classes as senders or receivers will also be eliminated.

The collection of scenarios that HL7 may publish does not limit the ways in which HL7 can be applied; other combinations of trigger events, interactions, and application roles that are that consistent with the interaction model may also be used.

The above notwithstanding, scenarios are normative. They represent a testable sequence of messages that may be used to validate a conformance for an *Application Role Class* that is included in the scenario.

3.4. Hierarchical Message Description and Implementable Message Specification

The relationship between the Interaction Model and the HMD, or between the HMD and the IMSs is unchanged in this proposal.

3.5. Process Summary

Figure CB is an initial attempt to show the process through a data flow diagram. As it was drawn with Visio the iconography is borrowed from SADIE-POOCH.[5]

---

[3]HL7 users have always used “message” indiscriminately, as a format or as a specific instance of sending it. The current MDF uses "Message" for this term. I am suggesting "Message Format."

[4] It has become clear in working the example that scenarios must be capable of expressing optionality and repetition. This is an open issues.

[5] Select Affected Drawing Icons Easily-Put Objects On Charts Haphazardly

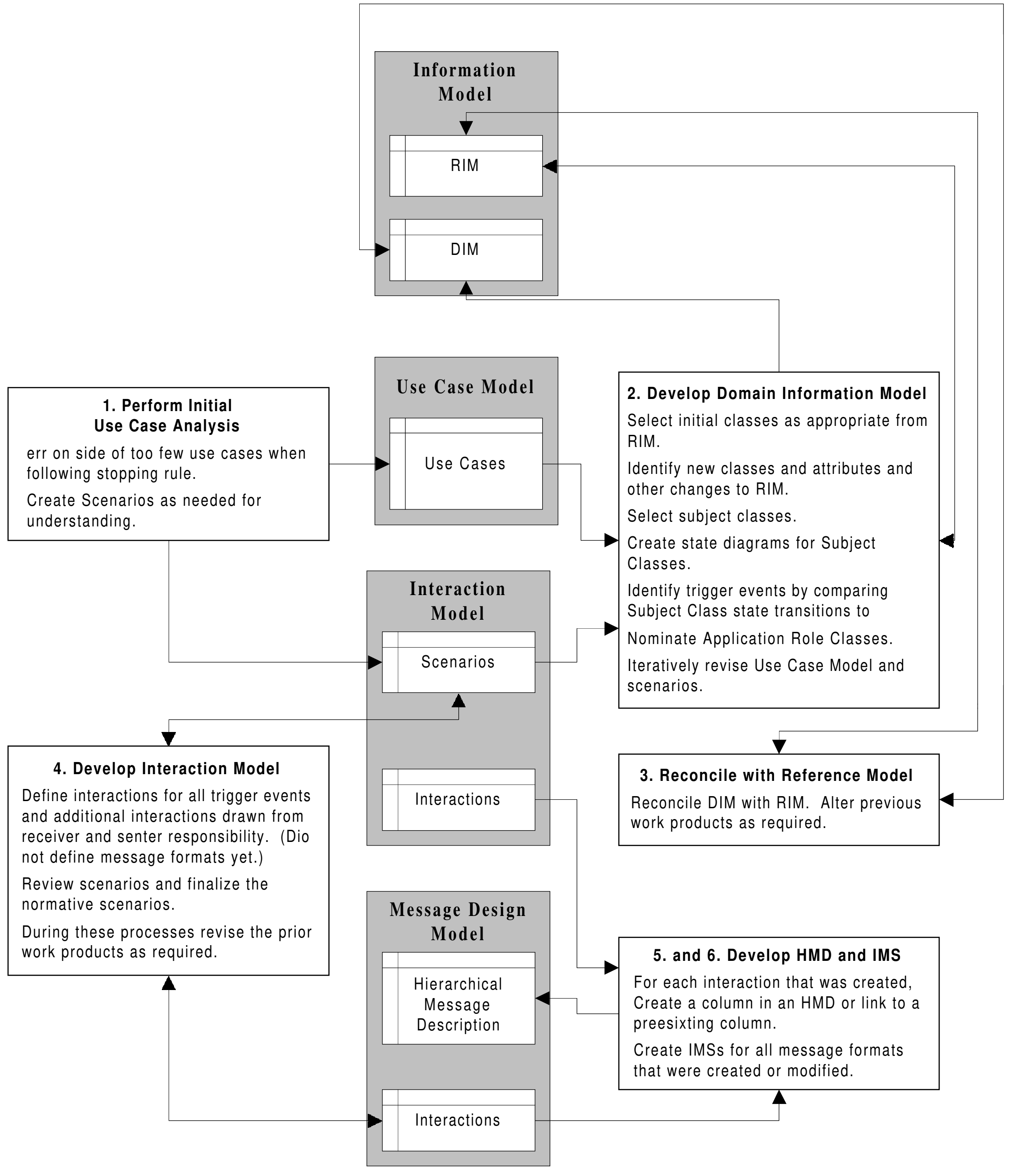

Figure CB. Data Flow Diagram to accompany the scope of this papers.

3.6. Metamodel

Jake worked out the portions of the metamodel that are relevant to this document in his Rumbaugh tool. Figure CA is that sketch.

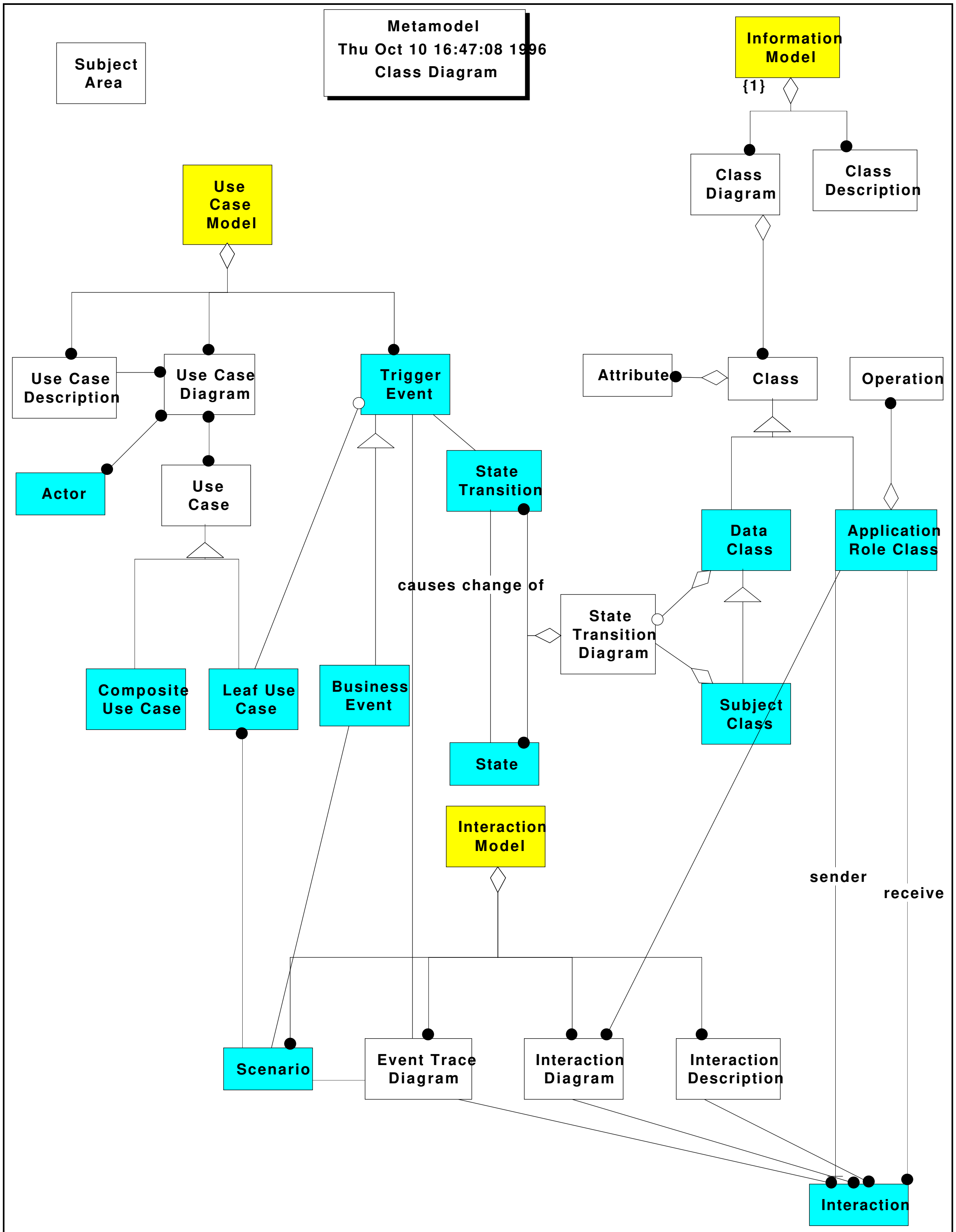

Figure CA. Quick sketch of meta-model.

4. EXAMPLE

The example used in this note starts with an assumed set of use cases described in Figure 1. It follows the use cases through the steps of the model. During this process the initial set of use cases will be revised.

4.1. The Initial Use Case

Figure DA represents a possible first use case as might be developed by a technical committee. It will be revised in the course of this example.

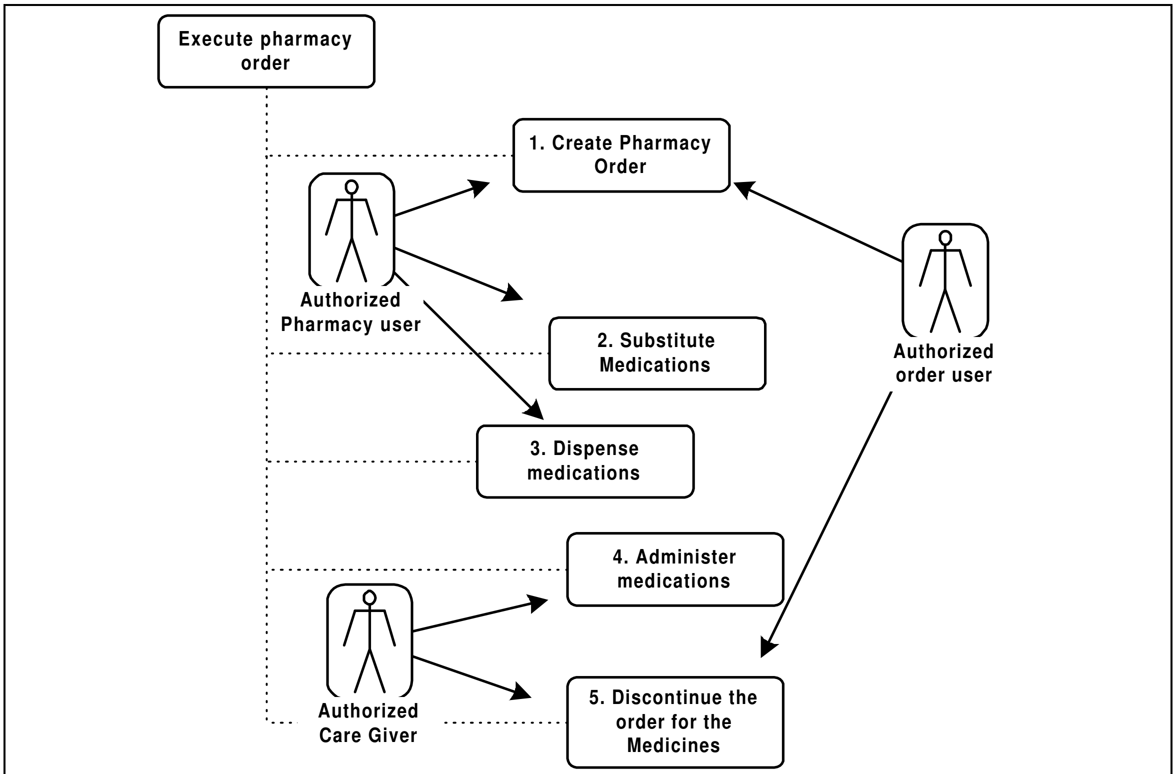

Figure DA. Initial use case.

Use case 2 refers to the process by which the Pharmacy department revised the order to use generics and to substitute trade names. Use case 4 refers to the process of attempting to give medications to the patient and recording whether the medication was received and was not subsequently vomited.

4.2. Analysis of Use Cases

As we proceed through the analysis of these use cases and construct interactions, we must arrive at some important decisions.

Do some of these use cases represent more than one interaction?

Will HL7 presume that any of these use cases may be automatic (i.e., performed without stimulation from an actor that represents a user)?

Will HL7 REQUIRE that some of these use cases must be automatic?

What assistance does the methodology provide in deriving the Application Role Classes and assigning the corresponding interactions to them? In other words, who will be the senders and receivers of the resultant messages? Not mentioning these information systems as Actors in the use cases makes it difficult to deal with certain concepts that the technical committee would no doubt feel important. These include

- description of interactions with a pharmacy robot
- the need of a order filler to return a filler number to the order sender
- the need to send some information to a data repository or electronic medical record.

We will see how they address these issues during the ongoing analysis.

4.2.1. Use Cases Evolution

Use cases 1 and 5 are immediate candidates for examination, because they involve two actors.

Figure DB represents an interim analysis after substantially more discussion of Use Case 1. In technical committee recognizes that in some cases the computer may be able to recognize that certain conversions can be made automatically. It therefore chooses to add "some system." While they will clearly have the pharmacy system in mind, they are unable to express it so far.

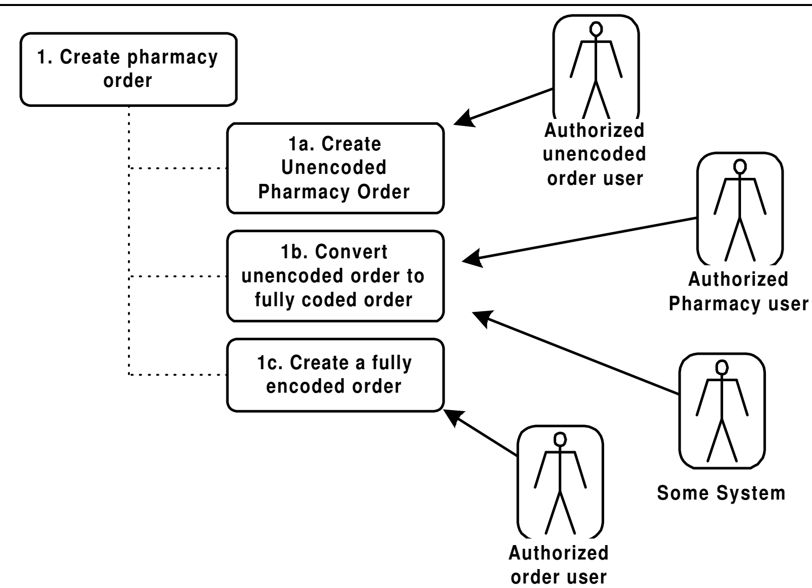

Figure DB. Expanded view of order creation.

Figure DC represents further analysis of use case 1.

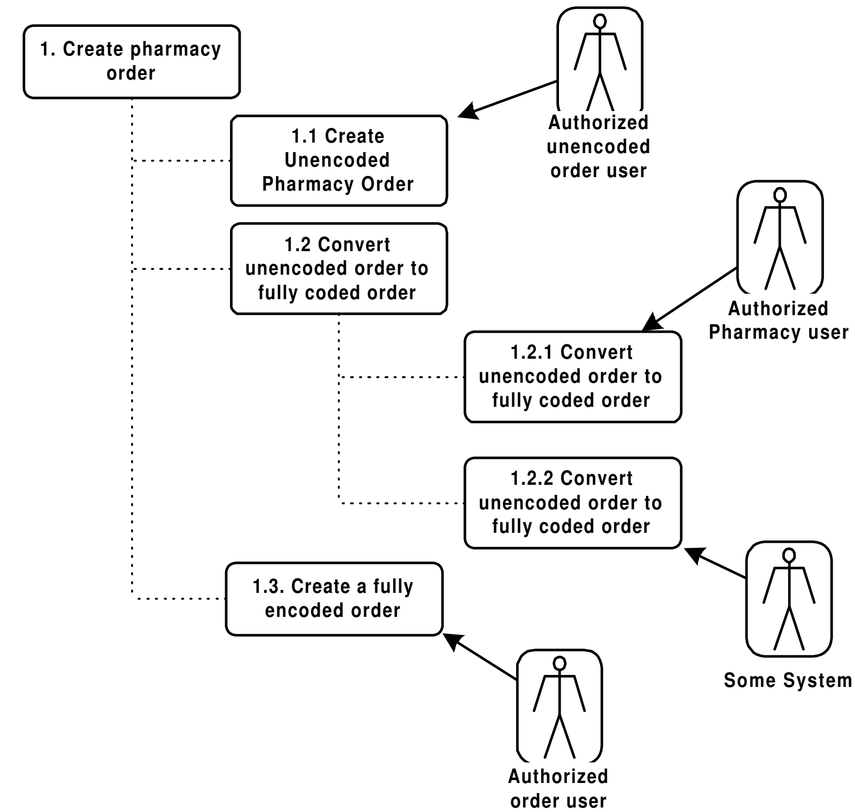

Figure DC further elaboration to eliminate two actors.

It is likely that these use cases would be used in more than one scenario:

a) physician pharmacy order: 1,2,4,5,6,7

b) direct entry of encoded pharmacy order: 3,4,5,6,7.

4.2.2. Information Model

Use case 6 makes it is clear that Technical Committee must find or create some specialization of Order like Pharmacy Order. So far, the use cases would help us to discover an important class in the Information Model.

The use cases deal with ordering a group of meds and then dispensing and discontinuing meds separately. This leads to finding or creating a separate class in the information profile, let's say Pharmacy Order Item. Another class is discovered!

A subject classes is identified, the Pharmacy Order. State transitions are identified for each of the leaf-level use cases. Secondary interactions are added to the interaction model for some classes.

Creating state changes to correspond to use case 5 makes it clear that state changes at the order item level are needed to deal with dispensing.

4.2.3. Dealing with Medicine Administration

Discussion of the state transitions around use cases 4 and 5 will discover that the current set of subject classes is incomplete. In coordination with the patient care technical committee, the pharmacy committee splits each use case as shown in Figure DD.

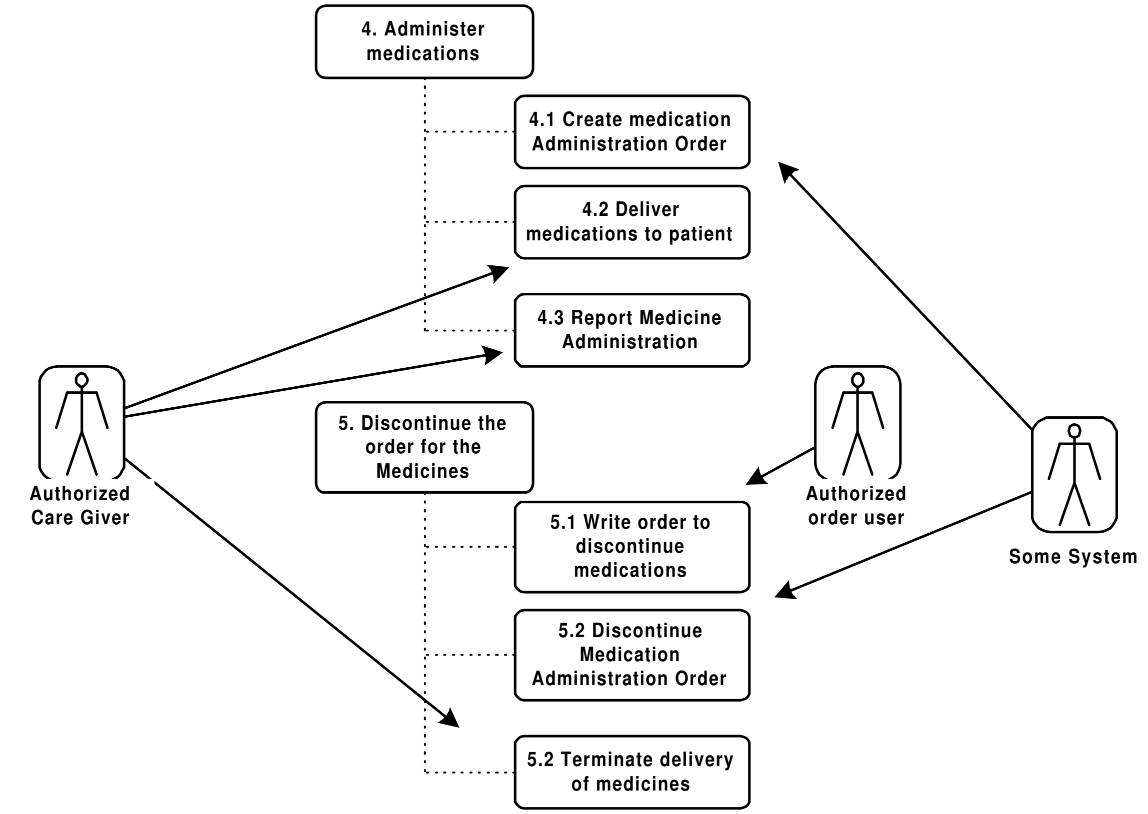

Figure DD. Elaboration of use cases to deal with care-givers role.

In addition, a new subject class and two new Abstract Role Class classes are created:

1. Medication Administration Order
2. Medication Administration Order Manager
3. Medication Administration Order Creator

The rest of section 4.2 has not been fully revised since our meeting of 10/10.[6]

The analysis continues:

111 physician creating a pharmacy order that may not be fully encoded

Actor: physician

There is mapped to a state change in the Pharmacy Order Class to indicate that an order is created in the 'physician-entered' state.

The Pharmacy Order Creator class is the sender of an interaction to the Pharmacy Order Manager class describing the order.

1.2 a physician order is converted to a fully coded order

[6] In redeveloping the example I have come to believe that 1.2.2 is unwise. It seems like we would rather use scenarios to make 1.2 an optional step. Similarly we need for scenarios to recognize repetition.

There is an associated state change in the Pharmacy Order Class that moves the order to a 'fully encoded, not final' state. It is concluded that this state change itself does not require an interaction to return the fully encoded order until after the processes described in use case 4.

The Pharmacy Order Manager class is the sender of an interaction to the Pharmacy Order Creator class providing the encoded order.

1.3 a fully encoded order is created directly by a user

This is mapped to a state change in the Pharmacy Order Class to indicate that an order is created in the 'fully encoded, not final' state.

The Pharmacy Order Manager class is the sender of an interaction to the Pharmacy Order Creator class providing the encoded order.

2 the fully encoded order is revised through generic substitution, or in response to clinical or formulary concerns addressed by the pharmacist

<Need to analyze dispensing here>

There is a state change in one or more objects of the Pharmacy Order Item Class indicating that certain meds have been dispensed. There is also a state transition in the Pharmacy Order Class indicating that some items have changed state.

4.1 creating an order for the administration of one or more medicines

4.2 administering these meds

4.3 reporting medicine administration

Discussion of the role of nurse in this use case makes clear that 4.1 has as its subject the Medicine Administration Order class. Therefore, the TC decides that HL7 will create an obligation on the Pharmacy Order Manager to create this administration order. It identifies Pharmacy Order Manager as the sender of an interaction. Accordingly it discovers the need for a Medication Administration Order Subject Class and corresponding Application Role Classes Medication Order Manager and Medication Order Creator.

There is also a state change in the Medication Administration Order indicating a new order has been created. This will be identified with the trigger event associated with use case 4.1.

Further analysis of the state changes in of the Medication Administration Order class lead to the discovery that a major information flow has been overlooked: communicating medicine administration records back to the system that created the order to administer the med (4.3). This can be discovered without a knowledge of system functions, because it is a manual function that is reported today.

4.3. Dealing with the Information Model

4.3.1. Creating Classes

4.3.2. State analysis

4.4. Discovering Other Application Role Classes

These analytic steps have not uncovered any signal to represent the pharmacy robot. Nothing derived from the analysis of the states of the order created a cue for this need.

We propose that at the completion of the state analysis technical committees should have a specific discussion of related application roles, to be associated with specific Application Role Classes.

This discussion might lead to the discovery of the recognition of a new Abstract Role Class: pharmacy order fulfiller. In order to add this to the use cases, use case 5 will have to be split.

In order to have a Application Role Class it will be necessary to create a subject class Pharmacy Fulfillment Order. The Abstract Role Class classes, Pharmacy Fulfillment Order Manager and Pharmacy Fulfillment Order Creator, would be created. A state analysis on the subject class would be performed. More than likely the state transitions would be identified with the trigger events associated with use cases 5 and 7.

Similarly, there was no cue to identify the Abstract Role Class for the clinical data repository or electronic medical record, as these are fairly passive receivers of data.

It may be easy for electronic medical record vendors. If they define the scope of their particular system to include creation of pharmacy orders then they can simply declare conformance with the to the interactions associated with the Pharmacy Order Creator class.

However the clinical data repository is clearly a passive role that wants to be informed of all or some of the changes in Pharmacy Order, Pharmacy Order Item, etc. Saying "all or some" is a cue that more definition is needed. Some CDRs, designed for a historical context, may only be interested in the final state, others may be interested in tracking the changes throughout the life cycle of the order object. We would expect that the Application Role Class "Pharmacy Order Historian" would be created for the former situation and "Pharmacy Order Tracker" for the latter. Electronic Medical Record vendors who don't want to make the Pharmacy Order Creator Conformance Claim would most likely opt for Pharmacy Order Tracker.

4.5. Building the Interaction Model.

5. GENERAL OBSERVATIONS AND QUESTIONS

5.1. Not all Roles Filled

What if an individual HL7 network has no system claiming the Medication Order Administration Manager Application Role?

HL7 must clearly state that an interaction is only required to be sent if there is at least one system in the network that makes the conformance claim of the denoted receiver of the interaction.

5.2. Same Application Handles Two Roles

If an application system handles two roles (represents two Abstract Role Classes) then it may be both the sender and receiver of some interactions. It is required to send these via an HL7 interface?

Absolutely not. The system can manage the information internally by updating a common data base or by any other method. HL7 does not standardize application systems or their internal architecture.

However, if there are other systems in a specific healthcare network, that play either of the roles, the application is required to play the other role in HL7 interactions. This is applicable to situations where the pharmacy application system takes on the roles of Pharmacy Order Creator and Pharmacy Order Manager. It might also say that by conforming to the Pharmacy Order Manager role, the pharmacy information system would be required to deal with systems claiming Pharmacy Order Tracker and Pharmacy Order Historian roles, if such be on the network. (Assuming the technical committee really define such application abstractions.)

5.3. Multiway Routing

How are interactions routed if there is more than one system in the individual healthcare network claiming the Medication Order Administration Manager role?

The author regards this as an important issue. It is not clear whether every Medication Order Administration Manager wants to receive every interaction sent to some Medication Order Administration Manager. Similarly, suppose there are two pharmacy information systems in a network (one for inpatient and ambulatory patients that use the pharmacy, and a regional network containing many retail pharmacies. The relationship of the regional network to the medical center system might be one of Medication Order Fulfillment Manager.

In current practice today the splitting of routing issues is handled by interface engines. However there approaches frequently have a practical limitation that the routing rules are based solely on the contents of the individual message being routed. It is not clear that HL7 can do any better, but the issue must be considered.

5.4. Alternatives in the Use Cases

How would this methodology deal with alternatives in the use cases?

Alternatives are supposed to be represented by a higher order use cases that contains use cases that represent a "normal" and alternative courses of action.

My current proposal is silent on this issue, but it can deal with alternatives. Since only leaf-level use cases are related to trigger events, alternatives will have been broken down through use cases, and must be treated through alternate scenarios. The example could easily illustrate it by creating a high level use case that combines the order entry process for the two scenarios that are ultimately reached.

It is not clear, however, that my approach doesn't ignite a combinatorial explosion, producing many scenarios.

5.5. What Will HL7 V3 Standardize

This is a very important question. It is clear that the individual interactions (and downstream message formats) are standardized. But what of the scenarios? On the one hand, the sequentiality associated with use case scenarios is captured in the preconditions and sender-receiver responsibilities. On the other hand, it is not clear that you can completely recover the scenarios from this information.

On the third hand, it would be virtually impossible for HL7 to capture all the valid scenarios for use in healthcare.

It is my opinion is that we must have our cake and eat it too. Specifically we must make conformance claims include the scenarios specified in the use cases, but not limit the application of HL7 to those scenarios. As always, the conformance claims represent a lower bound on the HL7 use supported by an application system rather than a definitive protocol.