Document Title **Representing Trial Design Metadata in XML**

Release **Proposal**

Copyright ©2008, Clinical Data Interchange Standards Consortium, Inc.
All rights reserved.

Change Control Log

Revision	Reason	Author(s)	Date
0.01	Document origination	Peter Villiers, Jan Kratky	10Oct2008
0.02	Responses to Peter A.'s comments, XML examples, Removed Instance constructs	Peter Villiers, Jan Kratky	10Nov2008

Table of Contents

1 INTRODUCTION 3

1.1 PROJECT GOALS 3
1.2 SEPARATION OF CONCERNS: TRIAL DESIGN VS. TRIAL EXECUTION 3
1.3 GLOSSARY 3
1.4 EPOCHS, ARMS, CELLS AND SEGMENTS ORGANIZATION 4
1.5 NAMING CONVENTIONS AND ATTRIBUTES USED IN DIAGRAMS 4

2 TRIAL DESIGN: STRUCTURE 5

2.1 DEFS VS. INSTANCES 5
2.2 BASIC STRUCTURE ..... **ERROR! BOOKMARK NOT DEFINED.**
2.3 INSTANCE STRUCTURE ..... **ERROR! BOOKMARK NOT DEFINED.**
2.4 EXAMPLES OF THE OBJECT STRUCTURES 13
2.4.1 Basic Object Def Structure 13
2.4.2 Basic Object Instance Structure 15

3 WORKFLOW 17

3.1 ENTRY AND EXIT CRITERIA 17
3.2 BRANCHING 18
3.3 GENERAL LOGIC FOR TRANSITIONING BETWEEN ACTIVITIES 19
3.4 EXAMPLES OF WORKFLOW 19
3.4.1 Entry and Exit Criteria 19
3.4.2 Default Transitions Between Activities 21
3.4.3 Conditional Transitions 22

4 TIMING 24

4.1 EVALUATION OF TIMING CONSTRAINTS 25
4.2 EXAMPLES OF TIMING 26
4.2.1 Absolute Timing of an Activity 26
4.2.2 Timing Associated with a Transition 27
4.2.3 Relative Timing between Activities 28

5 ELEMENT REFERENCE 29

5.1 ODM V1.3 ELEMENTS 29
5.2 TRIAL DESIGN SPECIFIC ELEMENTS 29
5.2.1 Strucutural 29
5.2.2 Workflow 30
5.2.3 Timing 31

1 Introduction

Review Note: Need to add an introduction for wider audience.

1.1 Project Goals

The project aims to:

- Build upon the foundation laid by the ODM team in representing aspects of a trial in XML.
- Leverage the ODM data types wherever appropriate.
- Use existing ODM constructs where possible.
- Enable the design-time use of reusable building blocks specific to trial.
- Provide enough information on which a trial execution runtime could operate to follow individual patients.

1.2 Separation of Concerns: Trial Design vs. Trial Execution

The trial design model is just that – a model of the trial's *design*. It is not a record of an individual patient's route through the trial. An execution runtime would reference the design elements to decide how a patient should progress through the trial and record this path accordingly.

The trial design incorporates aspects of:

- Structure: Arms, Cells, Activities, etc.
- Workflow: How a patient should progress through a trial – decision points, branches, etc.
- Timing: When the activities should happen relative to other activities or decision points in the trial.

Trial execution would need to capture at least the following:

- The path that a patient took through the trial design.
- The activities that were performed on that path.
- The planned times for when the activities should have happened for that patient.
- The actual times that the activities and decision points were performed.
- References to the design elements for static information that is common.

Trial execution is outside the scope of this initial proposal. However, the trial design model was developed with the intent of ensuring that an execution runtime would have sufficient information to operate over a trial design. (The consumption of a trial design representation by a trial-execution runtime sometimes is referred to as 'protocol insertion').

1.3 General Concepts Glossary

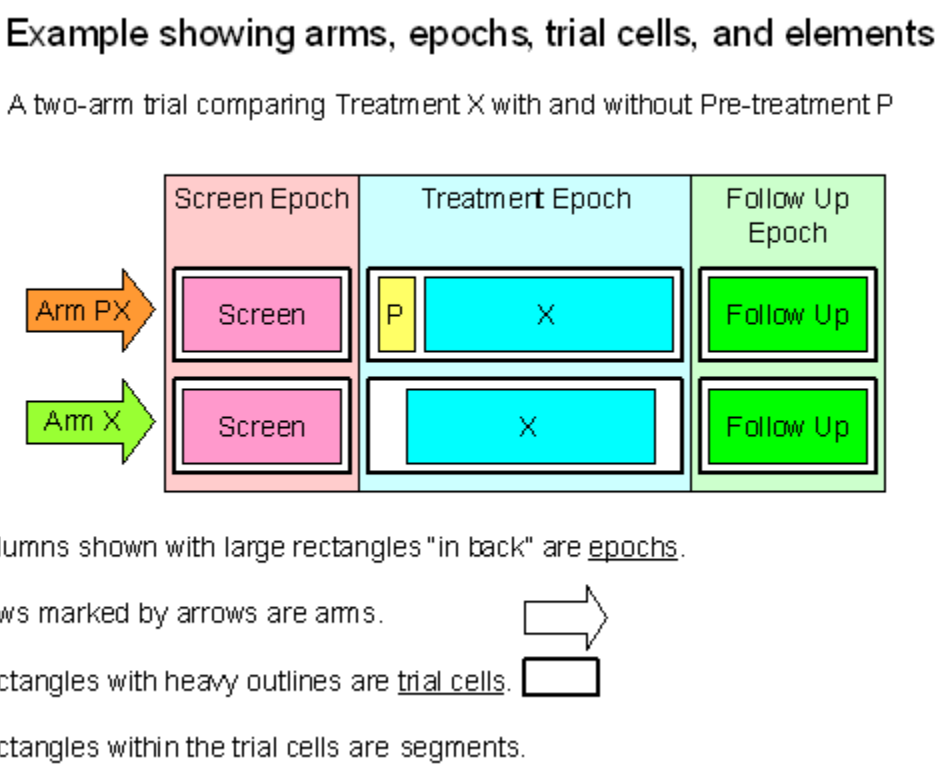
Author Note: The exact definitions of trial design concepts are currently under discussion for update. The information here may need to change accordingly.

Term	Meaning
Epoch	As part of the design of a trial, the planned period of subject's participation in a trial is divided into Epochs. Each epoch is a period of time which serves a purpose in the trial as a whole. Typically, the purpose of an epoch is to expose a subject to treatment (eg: Treatment), prepare for treatment (eg: Screening, Washout) or to gather data on a subject after treatment has ended

	(eg: Followup).
Cell	A trial cell is the part of trial design that describes what happens in a particular epoch for a particular arm. The cell describes how the purpose of its epoch is fulfilled for each arm.
Arm	A planned path through the trial. The path is composed of a trial cell for each epoch in the trial
Segment	A basic building block of trial design. It involves administering a planned intervention, which may be treatment or no treatment, during a period of time.

1.4 Epochs, Arms, Cells and Segments Organization

Arms, epochs, trial cells, and segments are closely interrelated. The diagram below illustrates the relationships between these concepts. Activities occur within each segment.



1.5 Naming Conventions and Attributes used in Diagrams

The remainder of this document will describe the trial design elements and their relationships. Where elements are reused from the ODM specification, they are prefixed with an “ODM13::” notation. It should be noted that not all attributes are included in the design at this point. Only those attributes that are required to illustrate the points under discussion are included.

2 Trial Design: Structure

Structural elements are the building blocks of the trial design model.

2.1 Defs vs. Refs

In the following sections, there are references to *Def*s and *Ref*s. In keeping with the design of ODM, a 'Def' is the declaration of an object. A 'Ref' is a reference to an instance of that object from some other entity. For example, segments are declared using elements named 'SegmentDef'. However, to indicate that a cell contains a particular SegmentDef, a SegmentRef will be employed to reference the appropriate SegmentDef.

2.2 Insertion into ODM Documents

Figure 1 shows some basic relationships between the structural definitions for trial design and those that already exist for ODM. The *ODM13::* elements are described in the ODM documentation. ODM13::Protocol will be the jumping off point for the trial design elements (in the XML document, all trial design markup will be descendents of the ODM13::Protocol element).

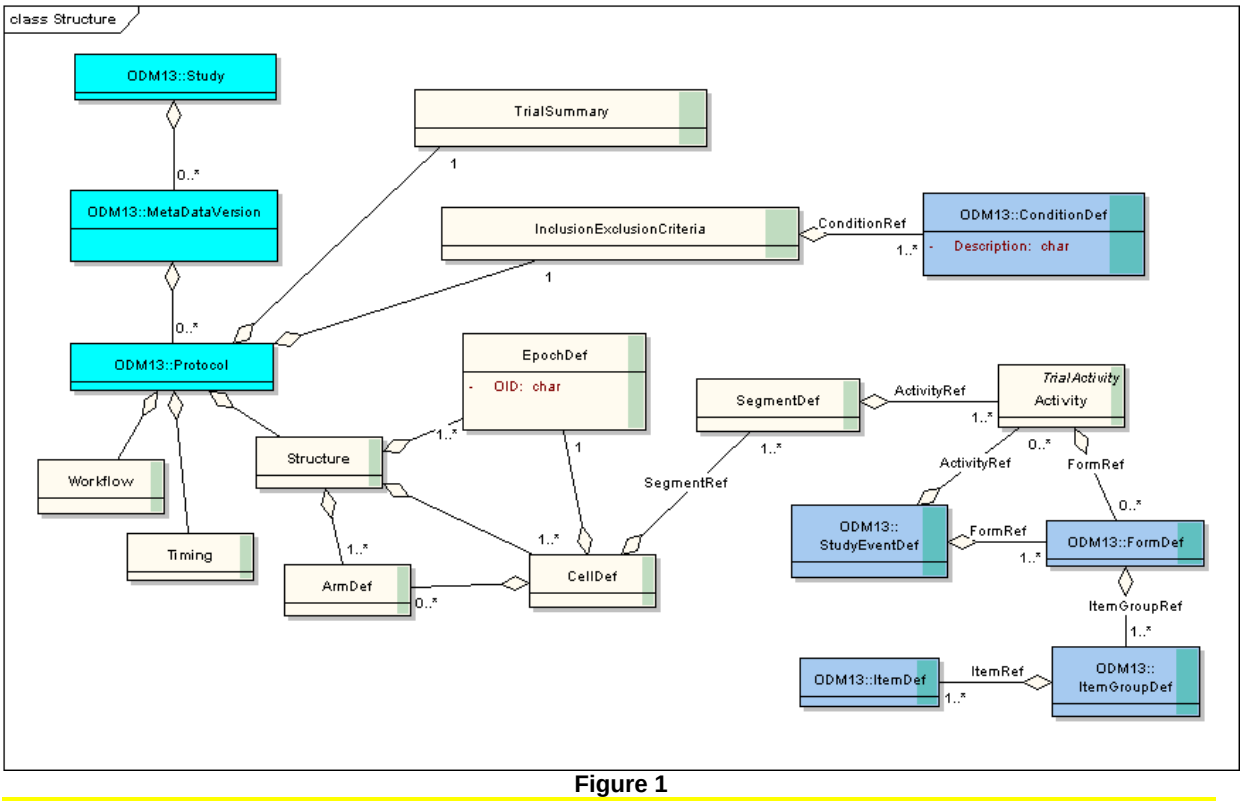


Figure 1

Review Note: Another point about ARM - is it really structural at all? Maybe we should think about having
- a list of arms (jjk. Yes, done. Added possibility of multiple armrefs from a cell – the *possible* arms at design time)
- a new concern (thin layer in a diagram) to do with setting the state of a subject as they pass through the trial. (jjk. A runtime concern. 'arm assignment' on the transition/activity?)
- a new construct within that concern that points to a TransitionDestination. It specifies that a subject state should be set. In this case, maybe it means making the relationship between the

subject information and one of the arms mentioned in the list above. (jjk. Open issue. We need to understand how much of the state we can give away at randomization time.)
- the new layer with the new assignment would possibly allow someone to run what if's - "what if I were to consider this a new ARM - how would that affect my allocation of patients?" - adaptive trials? This would also allow relatively easy recoding after the fact while maintaining the original arm decisions.
- blinded trials would not have any arm assignment. This would happen after the trial is unblinded.
- blinded trials. Would we need some way to overlay what the actual treatments WOULD be if we knew what the ARM was. Trial execution runtime would not know, trial design could say "if I were on this ARM, what would I be recieving & when", post trial once the subject assignment to arm was released, the dosing information is known. (jjk. The ArmDefs should be expanded to provide this.)

2.3 Structural Elements

This section describes the major elements that define the overall structure of a clinical trial.

TrialSummary provides a description of the trial that can be internationalized using the existing ODM 'TranslatedText' construct.

InclusionExclusionCriteria contains the list of those criteria that determine the eligibility of a subject for the trial. Some of these may be criteria applied to the 'project start activity'. Others may not be evaluated until a 'screening' epoch of the trial is under way. Such screening activities would reference these criteria as conditions that could force the subject's execution path into a final 'washout' state.

An **EpochDef** represents the information about the trial's Epochs. An ODM13::Protocol may include definitions of one or more epoch. since Epochs are sequential blocks of time for a trial, some ordering of the epochs relative to one another is also required (ODM convention is to use an 'OrderNumber' attribute to indicate ordering within sequences). An EpochDef contains a description, and (optionally) entry and exit criteria of its own.

An example of the XML for Epoch definitions:

```
<tdm:EpochDef OID="SCREPOCH" OrderNumber="1">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
       Screening Epoch
    </odm:TranslatedText>
  </odm:Description>
</tdm:EpochDef>
<tdm:EpochDef OID="TREPOCH" OrderNumber="2">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Treatment Epoch
    </odm:TranslatedText>
  </odm:Description>
</tdm:EpochDef>
```

An **ArmDef** provides the declaration of a trial arm. An ODM13::Protocol may include definitions of one or more arm. Arms, being primarily state definitions, contain little information other than a description. Arms do not have any ordering relative to one another.

```
<tdm:ArmDef OID="PLACEBO_ARM">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Placebo arm
    </odm:TranslatedText>
  </odm:Description>
</tdm:ArmDef>
<tdm:ArmDef OID="LOWDOSE_ARM">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
       Low-dose arm (54 mg) xanomeline)
    </odm:TranslatedText>
  </odm:Description>
</tdm:ArmDef>
<tdm:ArmDef OID="HIGHDOSE_ARM">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      High-dose arm (81 mg)
    </odm:TranslatedText>
  </odm:Description>
</tdm:ArmDef>
```

TODO: Some representation of arm-specific actions/dosages/etc. in a blinded trial. Description doesn't seem like enough.

A **CellDef** provides for the declaration of a trial cell, which is a common trial-design visualization for the intersection of an epoch with an arm. While some cells are in order relative to one another, any such ordering can be derived from the ordering of the epochs to which the cells are associated.

Each CellDef must reference exactly one Epoch to which it belongs, and reference zero or more Arms. A cell may belong to zero Arms when it represents a part of the trial with only one path (screening, for example, which is the same for all participants). A Cell may be part of exactly one Arm in the case of an unblinded trial. However, in the case of a blinded trial, no foreknowledge of a single Arm can exist. Therefore, this model permits a cell in a blinded trial to reference the one or more possible arms on which activities within the cell may take place. Only at runtime, during randomization/assignment, is it possible for the trial execution mechanism determine which of the possible arms has actually been assigned.

TODO: would some concept of a 'default arm' to represent the arm in the 'zero-arm' scenario be useful for generating diagrams &c.?

Examples showing each of these scenarios are given below. First is an example of CellDef XML for a trial with no blinding. In this example, three cells are defined. The first is a screening cell, and has no arm assignment. The second and third both reference the same Epoch, but occur within different Arms.

```
<tdm:CellDef OID="SCREENCELL" EpochOID="SCREPOCH">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Screening cell
    </odm:TranslatedText>
  </odm:Description>
  <tdm:SegmentRef SegmentOID="SCREENSEG" />
</tdm:CellDef>
<tdm:CellDef OID="DRUGCELL" EpochOID="TREPOCH">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Drug treatment cell
    </odm:TranslatedText>
  </odm:Description>
  <tdm:ArmAssociation type="unblinded">
    <tdm:ArmRef ArmOID="DRUG_ARM"/>
  </tdm:ArmAssociation>
    <tdm:SegmentRef SegmentOID="DRUGSEG" />
</tdm:CellDef>
<tdm:CellDef OID="OPERCELL" EpochOID="TREPOCH">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Operative procedure cell
    </odm:TranslatedText>
  </odm:Description>
  <tdm:ArmAssociation type="unblinded">
    <tdm:ArmRef ArmOID="OPER_ARM"/>
  </tdm:ArmAssociation>
    <tdm:SegmentRef SegmentOID="DRUGSEG" />
</tdm:CellDef>
```

The next example shows CellDef XML for a blinded trial. In this example, only two CellDefs are given – a screening cell and a treatment cell. The screening cell's definition remains the same as in the unblended example. However, since this is a blinded trial, the treatment cell cannot be assigned to a particular arm at design-time. Rather, a list of possible arm assignments is given.

```
<tdm:CellDef OID="SCREENCELL" EpochOID="SCREPOCH">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Screening cell
    </odm:TranslatedText>
  </odm:Description>
  <tdm:SegmentRef SegmentOID="SCREENSEG" OrderNumber="1" />
</tdm:CellDef>
<tdm:CellDef OID="TRECELL" EpochOID="TREPOCH">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Treatment cell
    </odm:TranslatedText>
  </odm:Description>
  <tdm:ArmAssociation type="blinded">
    <tdm:ArmRef ArmOID="PLACEBO_ARM"/>
    <tdm:ArmRef ArmOID="LOWDOSE_ARM"/>
    <tdm:ArmRef ArmOID="HIGHDOSE_ARM"/>
  </tdm:ArmAssociation>
  <tdm:SegmentRef SegmentOID="TRESEG" OrderNumber="1" />
</tdm:CellDef>
```

SegmentDefs represent sets of activities. Each segment is part of a cell (cells may contain one or more segments).

Note that segments are associated with Activities (we define a Form as one type of Activity), rather than with the more traditional 'visit' (aka: StudyEvent). This allows activities to happen outside of visits. It also permits the definition of a visit simply as a group of activities that are convenient to perform together.

A SegmentDef may have more than one association to a given Activity. This would indicate that the Activity is undertaken more than once within a segment. For data-collection activities (ODM13::FormDefs), the StudyEventDef-ODM13::FormDef relationship will reveal whether this happens in the same visit or across visits.

An example of the XML for segment definitions:

```
<tdm:SegmentDef OID="SCREENSEG">
  <odm:Description>
    <odm:TranslatedText xml:lang="en">
      Screening segment/period
    </odm:TranslatedText>
  </odm:Description>
  <tdm:ActivityRef ActivityOID="ECGFORM"/>
  <tdm:ActivityRef ActivityOID="VSFORM"/>
</tdm:SegmentDef>
```

There's a bit of a disconnect here with the base ODM requirement that a StudyEvent reference one or more Forms. It would be preferable for this core definition to be to zero or more forms. That way, our schema definition could introduce ActivityRefs rather than FormRefs as children of StudyEvent. That way, we could introduce a 'data collection' activity, allowing us to avoid the clumsy definition of a form as a type of activity.

2.4 Activities

TODO: Do away with instances, maybe. Am concerned, especially with forms & activities, that we don't have enough info to uniquely identify things. At the very least, we need OIDs on ActivityRefs (this essentially makes the Ref an Instance, which is pretty much how ODM does it – think ItemGroupDef->ItemDef). Make this an 'other types of activities' (not included in segments) and 'LZZT object example' section. Do we have permission to reference the LZZT study in this document?

This model includes Activities of several types. The most familiar type of activity is the 'data collection' Activity, in which a form, as defined by the ODM FormDef construct, must be filled in. The ODM StudyEventDef (generally regarded as representing a 'visit'), requires a reference to one or more FormDef. However, the situation may arise whereby more than one instance of the same type of form may need to be filled in during a visit (for example, vital signs may need to be taken when the patient first arrives, as well as just before the visit ends). The core ODM StudyEventDef does not capture this, so it will be extended by our model to include references to individual activity instances, which, in the case of data collection Activities, may reference the same FormDef.

Other types of Activities – TrialStartFinish and Scheduling – also exist in our model in order to enable robust processing of the trial design by an execution runtime.

The following figure shows the types of activities, along with their relationships to existing concepts such as the trial design segment and the ODM StudyEventDef.

TODO: Replace the below with ActivityTypes diagram.

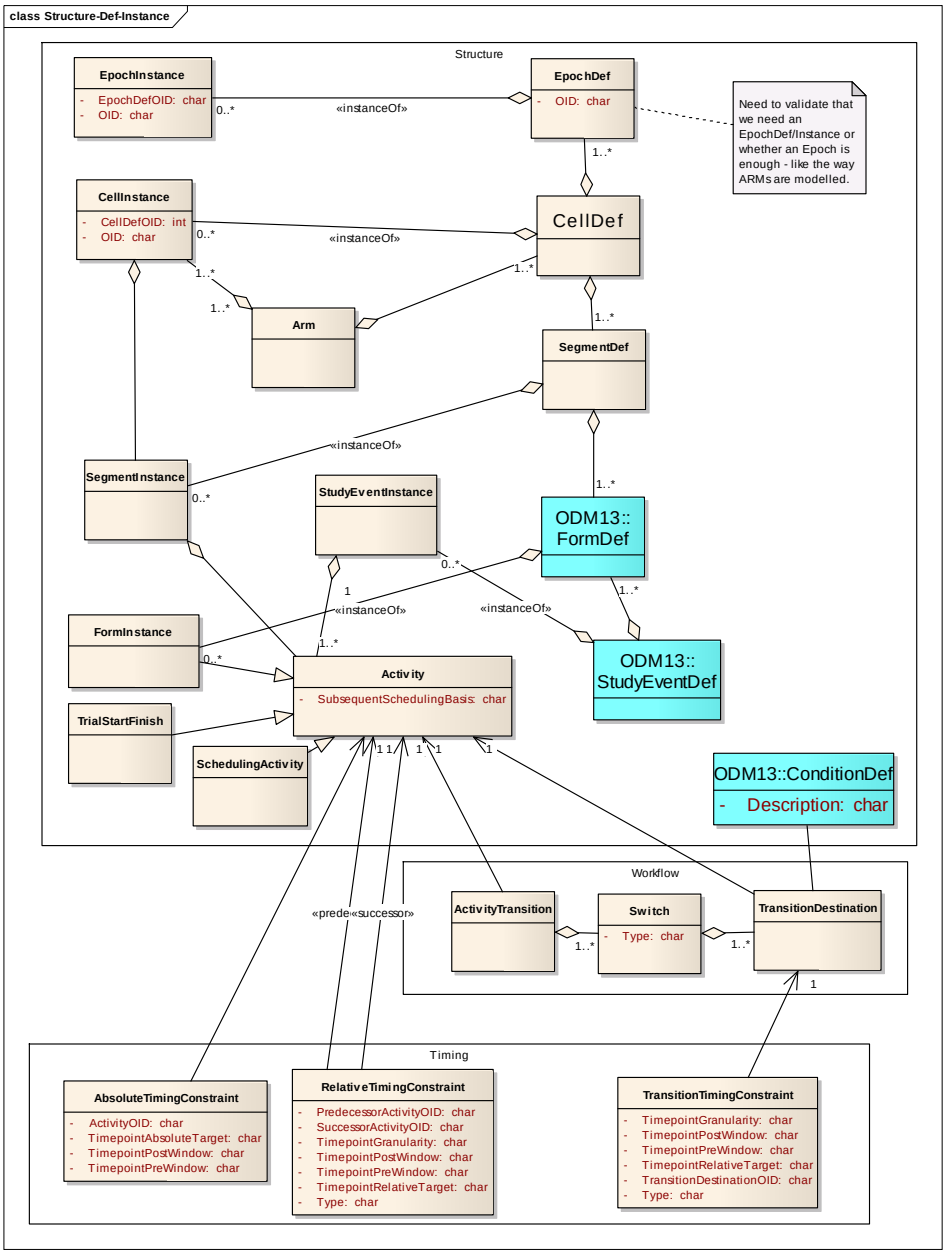


Figure 2

Review Note: Diagram is incorrect, CellInstances are associated with ARMs.

Review Note: Two associations need to be added between Activity and ODM13:ConditionDef – entry and exit.

Review Note: Need to determine if there is an ActivityDef needed in between SegmentDef and ODM13::FormDef. It may NOT be needed as the other subclasses in the instance diagram are *instance-specific* artifacts.

Review Note: Need to review the ODM13::FormDef and how this works in the Def structure without an ActivityDef construct. Is one needed?

Removed EpochInstance, CellInstance, SegmentInstance. Definitely don't need these.

Activities can be either instances of a form, the starting point for the flow through a trial, the end-point for the patient's flow through a trial, or a pseudo activity used in branching or scheduling.

<<Insert subset of the main diagram in here>>

ActivityDefs are not shared. The only need for sharing would be in order to avoid duplicating form definitions. The indirection of a data collection referencing a FormDef avoids this.

Data Collection Activities represent point in a trial at which a specific type of form needs to be completed. A data collection activity always contains an ODM FormRef element, which points to the definition of the form that must be filled in.

The **Trial Start Activity** identifies a single entry point into the trial for a given participant. The presence of this activity is useful as an anchor for timings relative to the start of a trial.

The **Trial Finish Activity** explicitly identifies when a patient's *execution thread* through the trial has finished. A patient may have many concurrent *execution threads* active in the trial design. This happens if they hit a parallel branching condition where they meet multiple criteria – in this case, the execution runtime would need to handle two *execution threads*. More will be discussed on this topic later in the document. An activity with no transitions to subsequent activities would be assumed to have an implied default transition to the end type element.

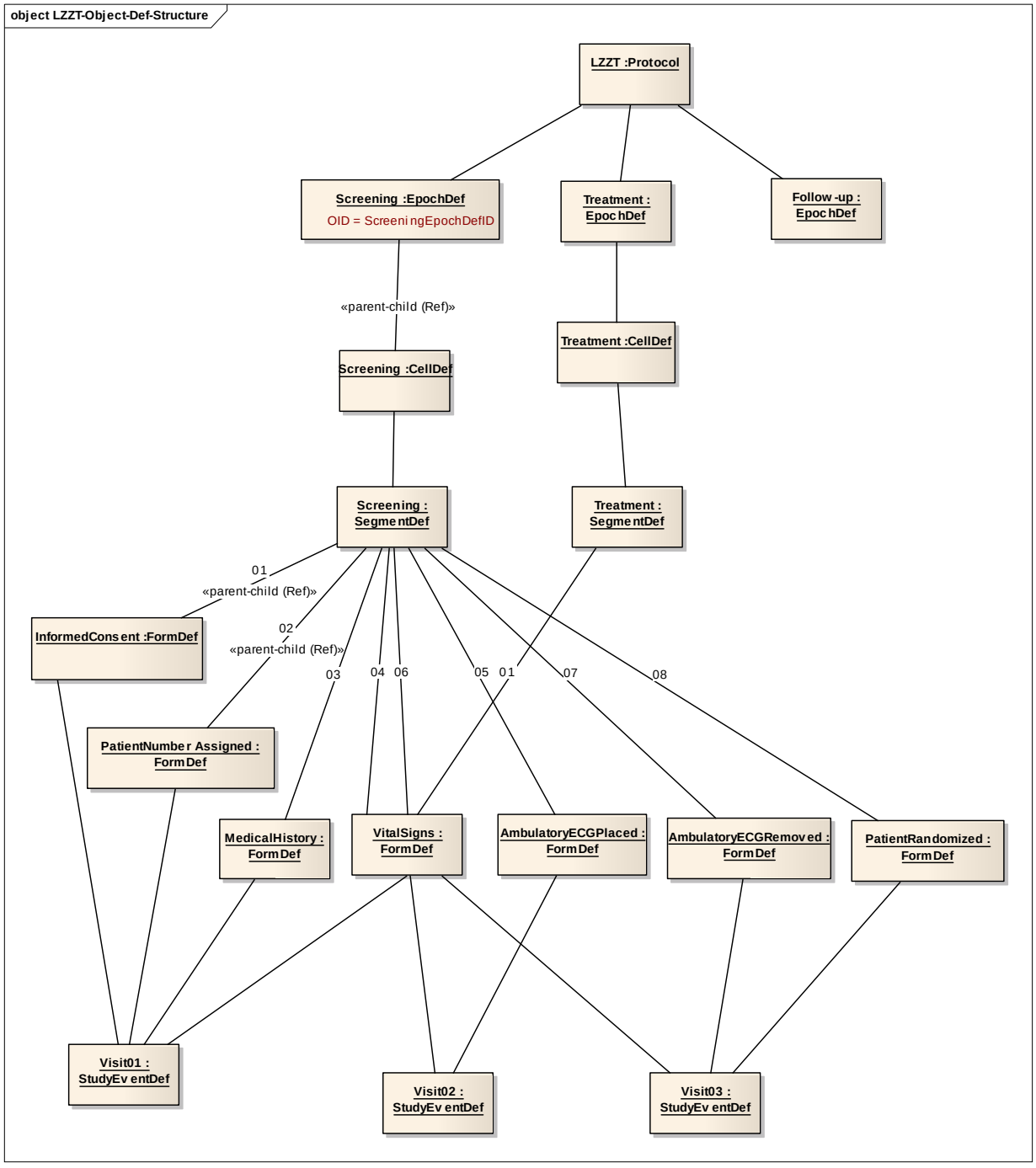
The **Scheduling Activity** type is included to allow designers to include timing and follow transition rules that are not associated directly with a form. (This type of activity is sometimes referred to as a 'Milestone'). For example, the trial might call for the investigator to confirm that some test results had been received before continuing. If the answer is yes then one path may be taken, if no then a timing delay of X days may be introduced. Neither the workflow condition nor the timing are associated with FormInstances; the Scheduling Activity permits the required action to take place.

The order in which activities are carried out, and even whether they are carried out, is determined by the workflow constructs between the activities. When activities are declared in the Structural section, no order is implied; ordering is given by the workflow definition. In the structural definitions, the only relative ordering may be implied by the activity's association to a particular segment, which has an ordering within a given Cell and Epoch. Ordering of activities within a segment is not given in the structural definitions, and is exclusively the domain of the workflow definition.

Timing of activities relative to one another is handled by the Timing constructs. These are discussed later in this document.

2.5 Examples of the Object Structures

2.5.1 Basic Object Def Structure



The diagram shows LZZT, a Protocol, that is a double-blind placebo controlled trial. LZZT has 3 EpochDefs: Screening, Treatment and Follow-up.

Review Note: At this point, the LZZT examples used are not exactly like the protocol so it may be better to use a different name. Arms are left off this diagram until they have been completed Arm vs. ArmDef.

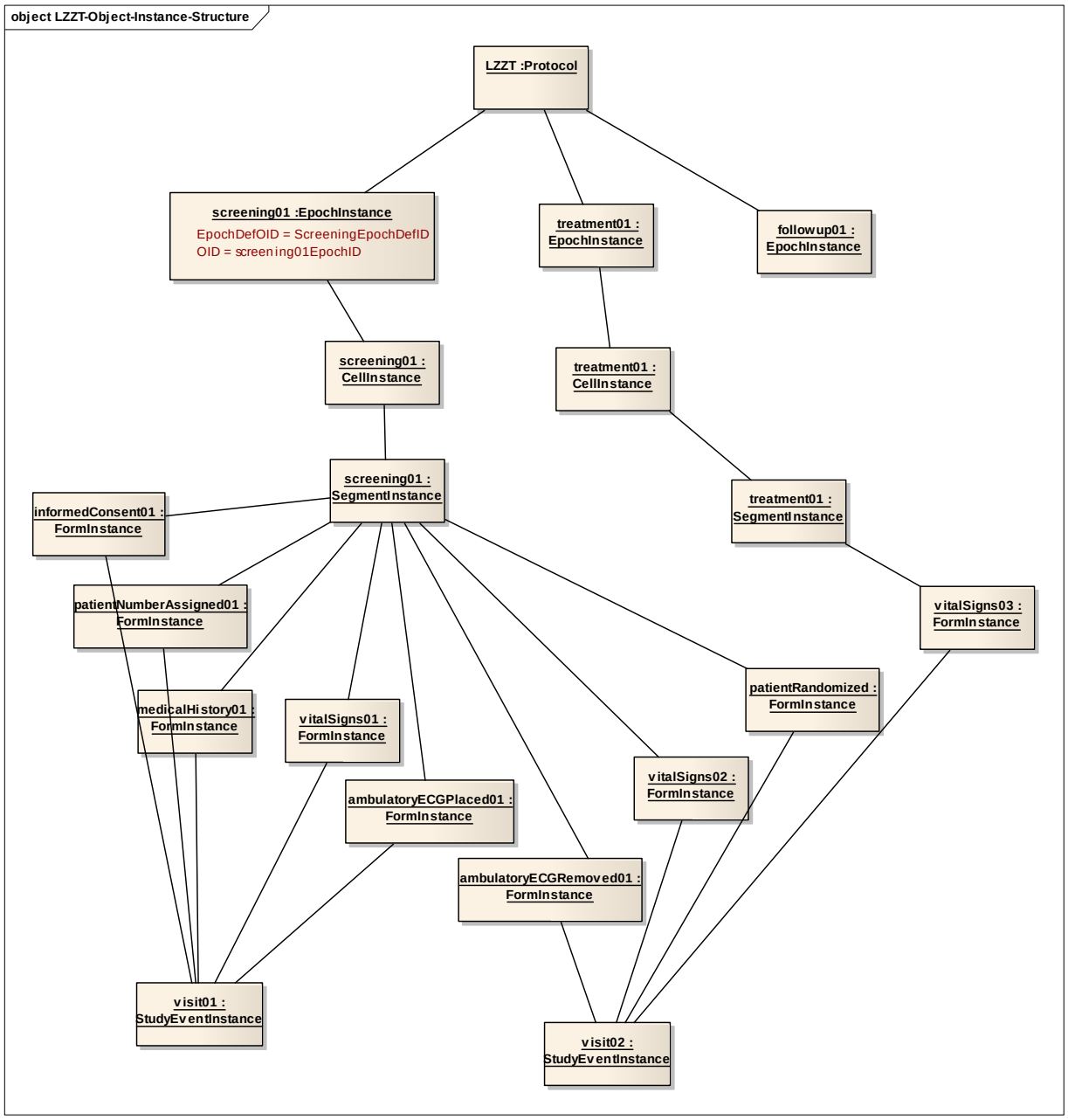
The Screening:EpochDef has one CellDef as during the design and execution, neither the patient, theinvestigator or the sponsor would know which treatment a patient was on.

The Screening:Cell has one SegmentDef called Screening also. The Screening:SegmentDef has 7 FormDefs associated with it. Notice however that, there are 2 associations to the VitalSigns:FormDef. This shows that the vital signs will be collected twice during a screening segment. The numbers on the association are a default sequence for the forms that could be used as a default workflow path by an execution runtime once the instances are created.

The StudyEventDefs at the bottom of the diagram represent the visits and indicate which forms make up that visit.

Notice that Visit03:StudyEventDef refers to VitalSigns:FormDef, AmbulatoryECGRemoved:FormDef and PatientRandomized:FormDef. The three form definitions are related to different segments. The workflow would show the transition ordering but this is showing that as soon as the patient randomization is completed, the patient would transition into the treatment phase of the trial. This is an example of how a subject can transition from one segment/cell/epoch to another one based on something that happens during the visit.

2.5.2 Basic Object Instance Structure



Review Note: Associations on the diagram are incorrect and need to be updated. Visit03 needs to be added.

The figure above shows the corresponding instance structure. The screening01:EpochInstance shows a reference/association back to the EpochDef on which it is based as well as having it's own unique identifier. Other instances have the same reference back but are not shown in the diagram.

screening01:SegmentInstance would maintain a reference back to the Screening:SegmentDef that it is base upon. It has eight distinct FormInstances associated with it that correspond to the Def's

associations with the FormDefs. There is not ordering of the SegmentInstance-to-FormInstance as this is supplied in the workflow.

Visit01's associations to the FormInstances define what can happen during the course of a visit.

3 Workflow

Trial workflows are defined using a handful of constructs that make robust branching available to a trial designer.

Review Note: Pull the sections of the structure diagram out that contain the Activity and the Workflow.

The Workflow sub-model is specified in a distinct section of XML from the structural elements and reference objects defined in the structural sub-model. This keeps the concerns separated and allows different workflow representations to be used over the same structural elements.

We haven't defined what's meant by sub-model

3.1 Entry and Exit Criteria

Review Note: Add diagram with Activity + link to ODM13::ConditionDef. Maybe move the example to under here also.

ODM13::ConditionDefs can be associated with activities, segments, cells or epochs. The association can either be of type *Entry* or *Exit*. This enables the trial designer to add zero or more entry or exit criteria to one of these objects.

Do the entry and exit criteria belong in the workflow section? Or structural? Even though they are fairly static in their association with a particular activity instance, they seem like the sort of thing that the workflow engine would have to know about. However, the same could be said of activities themselves, which are both structural elements *and* workflow elements (nodes). The only thing really specific to workflow are the transitions.

An example of the XML for entry and exit condition specification for an epoch and for an activity:

```
<tdm:EntryExitCriteria OID="CRIT_TRT_EPCH" Type="Epoch"
                       StructuralElementOID="TREPOCH">
  <tdm:EntryCriteria>
    <tdm:Criterion ConditionOID="COND_00"/>
  </tdm:EntryCriteria>
  <tdm:ExitCriteria>
    <tdm:Criterion ConditionOID="COND_01"/>
  </tdm:ExitCriteria>
</tdm:EntryExitCriteria>

<tdm:EntryExitCriteria OID="CRIT_ACG_ACT" Type="Activity"
                       StructuralElementOID="ACTDEF_ECG">
  <tdm:EntryCriteria>
    <tdm:Criterion ConditionOID="COND_02"/>
  </tdm:EntryCriteria>
  <tdm:ExitCriteria>
    <tdm:Criterion ConditionOID="COND_03"/>
  </tdm:ExitCriteria>
</tdm:EntryExitCriteria>
```

All entry criteria must be met for the workflow to transition into the given structural element. Similarly, all exit criteria must be met for the workflow to transition away from the given structural element. Later subsections will describe the semantics to be honored by the workflow processor.

3.2 Branching

The constructs that represent workflow are shown in Figure 2.

Transition elements are associated with Activities only. Transitions are not defined between segments, cells or epochs; these transitions happen by virtue of the containment structure for the activities. An Activity is always contained in a Segment, which in turn is always Contained within a Cell. Each Cell is associated with one Epoch, giving us the containment hierarchy Epoch > Cell > Segment > Activity.

There can only be one Transition per Activity. If no Transition is present then the Activity is a 'dead end'; a runtime may assume that this implies a default transition to the trial finish, but this is not q required semantic. Ideally, validation of a trial design would ensure that no dead ends exist before the design is inserted into an EHR system/runtime.

A Transition represents the potential flow away from a given activity. That is, a Transition is processed when all exit criteria for the activity have been met. Other constructs exist to describe the target/targets of the transition.

Switch defines a set of **TransitionDestination** elements that are to be evaluated in the order they are encountered. Each TransitionDestination points to an activity and also zero or more ODM13::ConditionDefs. When the transition destination is encountered, if ALL of the conditions evaluate to true then the transition is followed. If no conditions are defined for a particular destination then the destination condition is evaluated as 'true', and the transition is followed.

There are two types of switches:

- Single: The first transition definition whose conditions evaluate to true is followed. This is the default.

- Parallel: Any transition definitions whose conditions evaluate to true are followed.

Additionally, there may be one or more switches under an activity transition. This means that an execution runtime should be able to handle multiple active execution threads through the model for a given patient.

Q: Is this really necessary if we are enabling parallel destinations within one switch?

Here is some example XML that shows two types of transitions: one unconditional transition, and one that specifies branching to one of several targets depending on conditions.

```
<!--  unconditional transition -->
<tdm:Transition OID="START_TRANS" SourceActivityOID="ACT_START">
  <tdm:Switch Type="single">
    <tdm:TransitionDefault TargetActivityOID="ACT_VS01" />
  </tdm:Switch>
</tdm:Transition>

<!-- conditional transition -->
<tdm:Transition OID="START_TRANS" SourceActivityOID="ACT_START">
  <tdm:Switch Type="single">
    <tdm:TransitionDestination TargetActivityOID="ACT_VS01"
                          ConditionOID="COND_00" OrderNumber="1"/>
    <tdm:TransitionDestination TargetActivityOID="ACT_ECG"
                          ConditionOID="COND_01" OrderNumber="2"/>
    <tdm:TransitionDefault TargetActivityOID="ACT_FINISH" />
  </tdm:Switch>
</tdm:Transition>
```

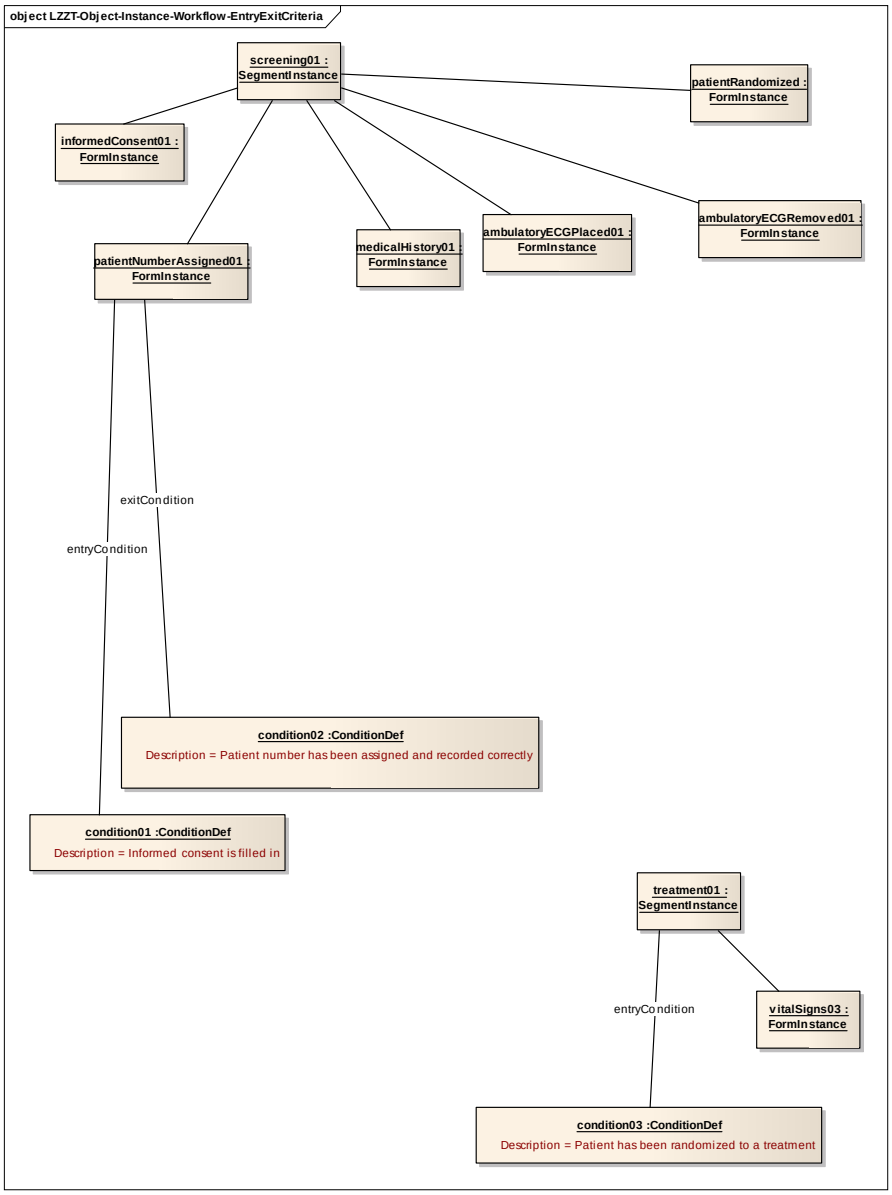
3.3 General Logic for Transitioning Between Activities

The combination of entry/exit criteria and branching allows for very powerful combinations to be created. The general transition semantics that the runtime execution would need to provide are:

- On the source Activity that has completed:
  - Check the exit criteria have been met.
  - Determine the transition(s) to follow.
  - Check *parent* exit criteria are met.
- For each transition path to be followed:
  - Check *parent* entry criteria for the target Activity are met.
  - Check the entry criteria for the target Activity are met.
  - If all the above are fulfilled then the transition is complete.

3.4 Examples of Workflow

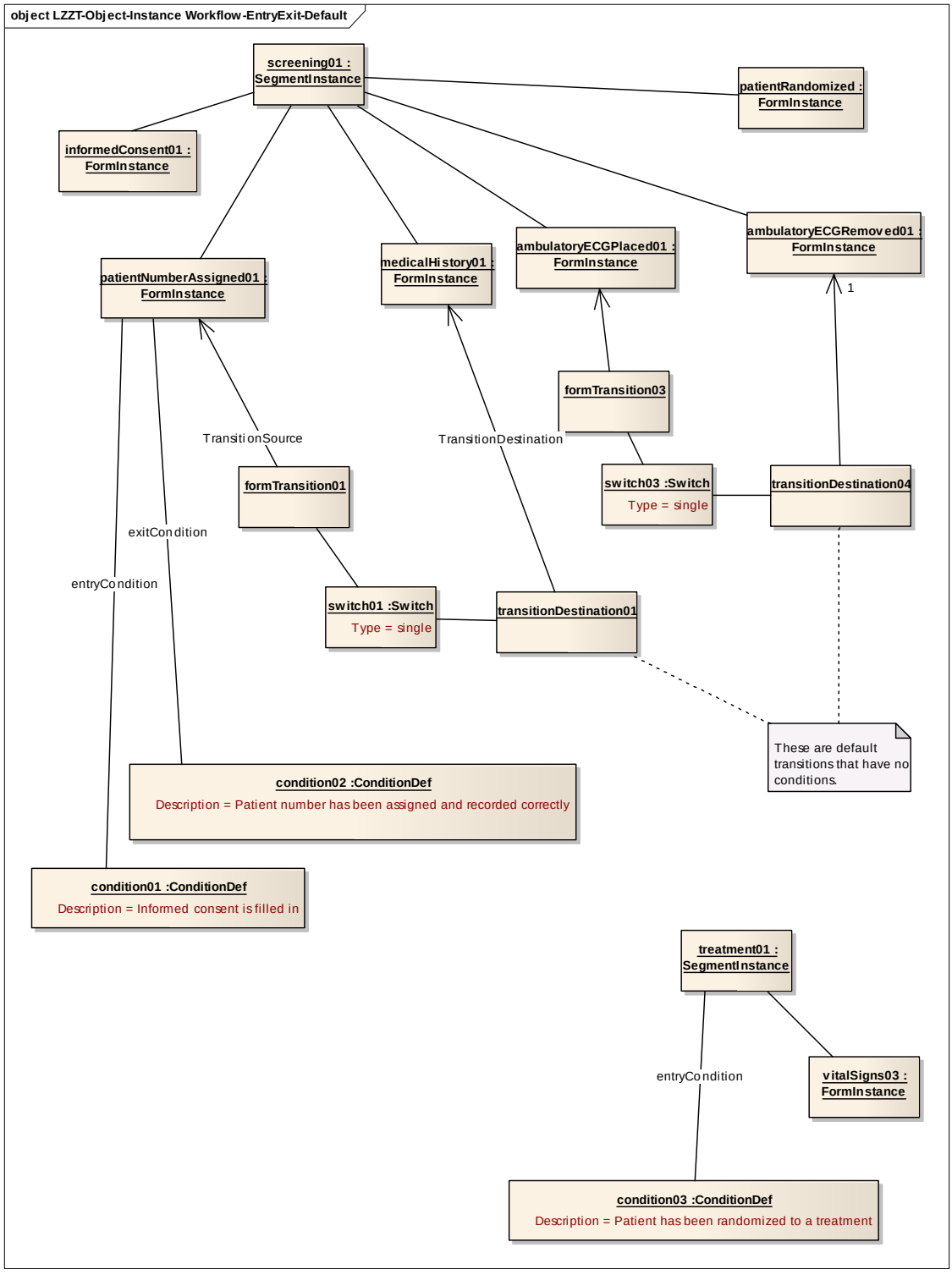
3.4.1 Entry and Exit Criteria



The figure above shows two examples of entry and exit criteria. The patientNumberAssigned01:FormInstance has entry and exit criteria and the treatment01:SegmentInstance has one entry criteria. The following XML snippets shows these conditions.

```
Example XML goes here.
```

3.4.2 Default Transitions Between Activities



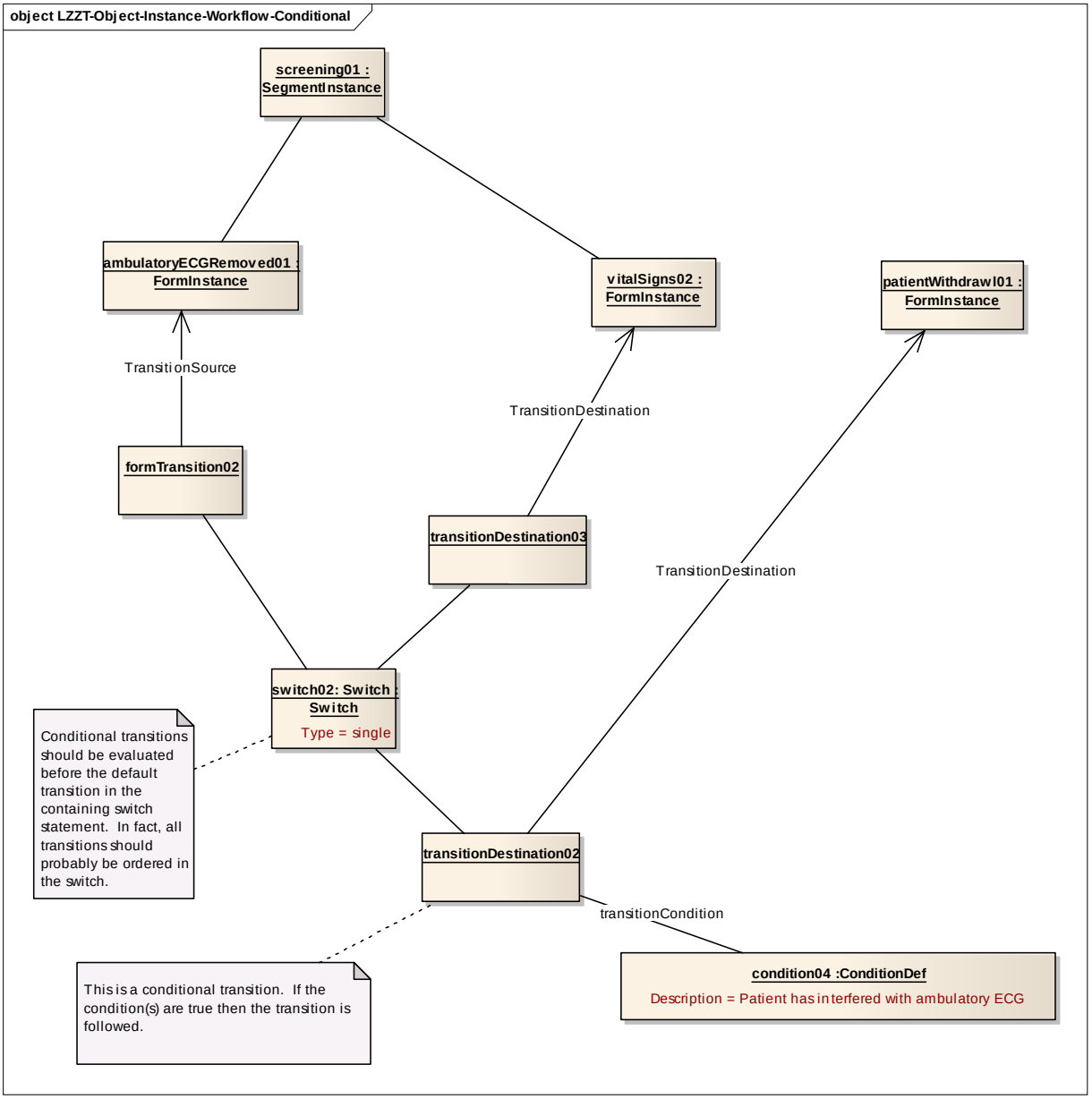
The figure above shows two default transitions which are the most basic type of transition in the workflow. Essentially, when one activity finishes it determines which the next one should be.

When patientNumberAssigned01:formInstance completes, formTransition01:ActivityTransition defines the possible transition points. It has only one switch that has only one transition destination, and this has no conditions so it is followed by default and medicalHistory01:FormDef is the next activity to be performed.

```
Example XML goes here.
```

ambulatoryECGPlaced01 is defined in exactly the same way. Once this is completed, the next activity in the workflow is ambulatoryECGRemoved01.

3.4.3 Conditional Transitions



The figure above shows a conditional transition in addition in the context of a single switch. The scenario was that a patient has worn an ambulatory ECG for 24 hours. If the patient interfered with the ECG during this period then they cannot continue in the trial.

Once the ECG has been removed then formTransition02 defines what happens next. There is a single switch with two transition destinations.

Review Note: Maybe need to annotate to include transitionDestination ordering.

TransitionDestination02 is evaluated and if the condition is true then the path to patientWithdrawl01 is followed. If this is false then transitionDestination03 is evaluated (it is always true) and the default path is followed to vitalSigns02.

```
Example XML goes here.
```

4 Timing

Review Note: Copy the timing section of Figure 2 into here.

As with workflow, the timing sub-model is separated from the structural elements and points into that sub-model. Similarly, there are only a few constructs for timing that can be used to create very detailed criteria.

In the following, there is the concept of an ideal, relative target duration between activities and then a pre-window and a post-window. The idea behind the target is to provide a trial execution runtime with a definitive time that it should try use in scheduling the activity. The windows provide some leeway in doing this and also help if more than one timing constraint is placed upon an activity. The pre and post windows are different so as to allow the designer a greater level of flexibility.

It was noted that sometimes, the trial designer would like subsequent activities to be scheduled relative to the planned time that the activity took place. On other occasions, the designer would like to have the subsequent activities relative to the actual time that a activity took place. The attribute SubsequentSchedulingBasis on the Activity provides a way to do this.

AbsoluteTimingConstraint is used to limit when an activity can take place during a day. It is generally intended to be used in conjunction with some other, more general, timing constraint and will restrict the window for the associated Activity.

TransitionTimingConstraint is associated with a TransitionDestination element.

Does this mean that it must be a child of TransitionDestination? That breaks the workflow/timing separation a bit. Currently, TransitionDestination is buried within a Transition, so referencing it will be awkward. Well, one way or another – either it must be a child of TransitionDestination, or it must reference one.

It provides the trial designer with the ability to add relative timing information to a transition using the activity referenced by the ActivityTransition as the source timing anchor.

The Type attribute specifies the type of dependency between the source and destination activities. The default (and most common) should be considered finish-to-start (FS) which means that the destination activity should start relative to the finish of the source activity. Other types are allowed and these are included in the Element Reference section.

The TimepointRelativeTarget attribute is a duration that defines the ideal interval between the source activity [anchor] and the destination activity. The TimepointPreWindow and TimepointPostWindow attributes are also durations that identify the window around the ideal interval where it is allowed for the activity to take place.

The TimepointGranularity allows the designer to override any pre/post window and widen the scope of the window using the following values:

- PY – it is allowed for the activity's timepoint to happen anytime in that year.
- PM – it is allowed for the activity's timepoint to happen anytime in that month.
- PD – it is allowed for the activity's timepoint to happen anytime in that day.
- PTH – it is allowed for the activity's timepoint to happen anytime in that hour.
- PTM – it is allowed for the activity's timepoint to happen anytime in that minute.
- PTS – it is allowed for the activity's timepoint to happen anytime in that second.

In the preceding list, the activity's timepoint is identified by the type attribute. As an example, consider an activity A that has a default transition to Activity B. The TransitionDestination has a timing associated with it that has the following values:

- Type: FS
- TimepointRelativeTarget: PT48H What is the 'PT' here? Need to use & reference a standard
- TimepointGranularity: PD

This means that the start of activity B should happen at any point in the day that is 48 hours after activity A.

RelativeTimingConstraint is very similar to TransitionTimingConstraint except that it references the predecessor and successor activities directly. It allows the designer to make any two activities relative to one another by either the start or finish.

It is reasonable to ask *"Why not use RelativeTimingConstraints everywhere instead of TransitionTimingConstraints?"*. The reason is that relative timing constraints are not specific to the actual path that a patient will take through the trial whereas transition timing constraints are. A trial execution runtime can make use of the difference for scheduling and compliance purposes – did the patient activities occur within the allowed windows.

4.1 Evaluation of Timing Constraints

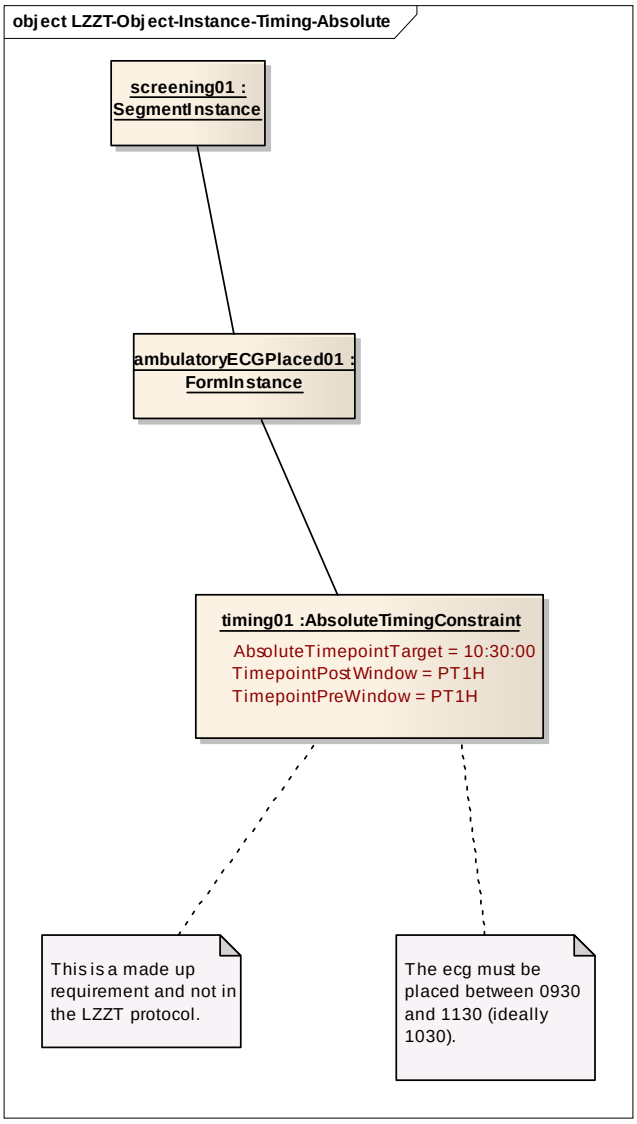
The individual timing constraints define a target time for an activity and optionally a window around which it is allowed for the activity to happen. The may be zero or more timing constraints on an activity which set up different target times and windows.

A trial execution runtime must be able to reconcile these to identify:

- The best time for the activity to happen that is inside all the windows and closest to the most target times.
- If the windows are non-overlapping and flag/identify that this is an issue with the patient's progress in the trial.

4.2 Examples of Timing

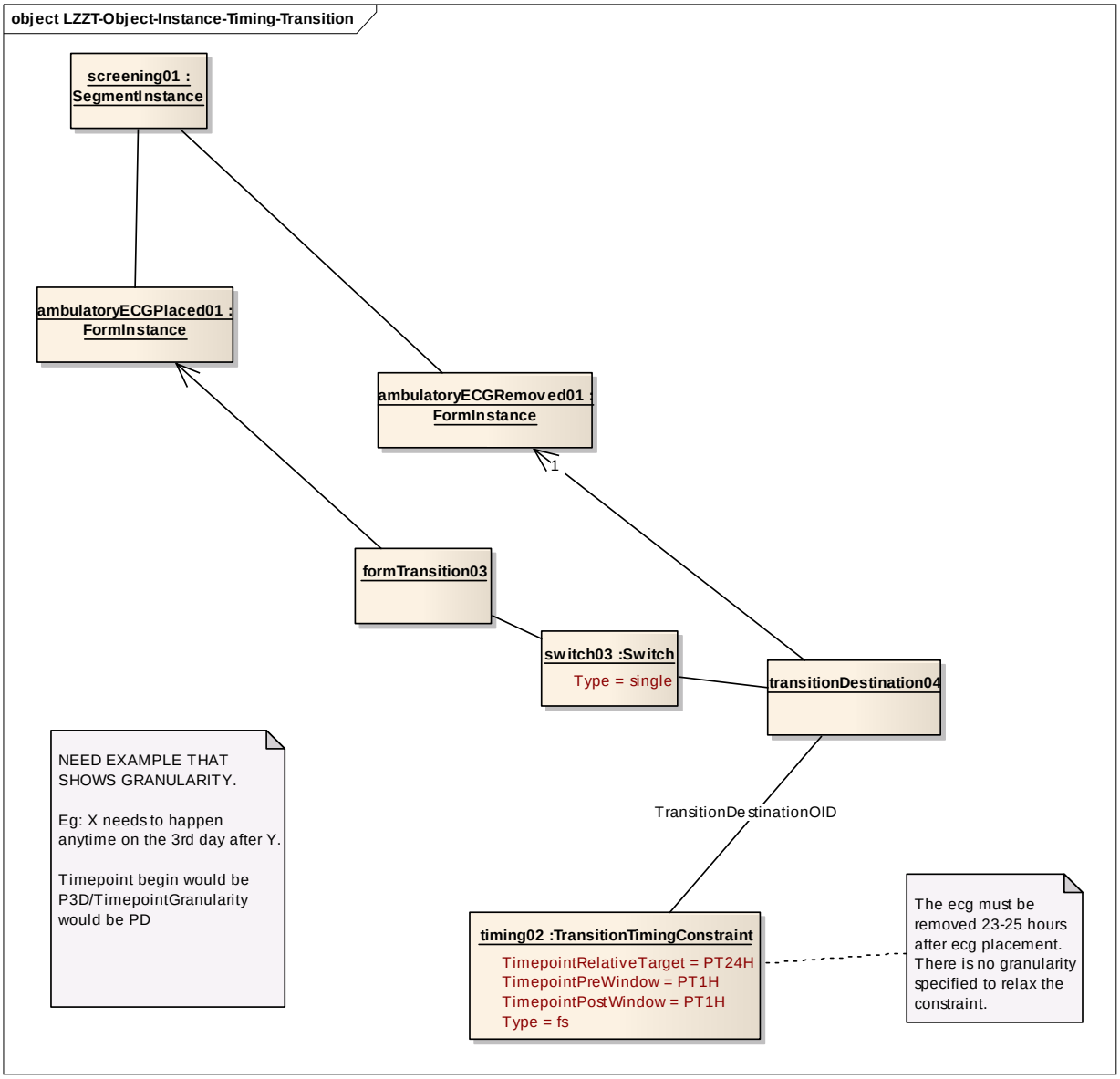
4.2.1 Absolute Timing of an Activity



The example above shows that the ambulatory ECG must ideally be placed at 10:30am but it is allowable for it to happen between the hours of 09:30am-11:30am.

```
Example XML goes here.
```

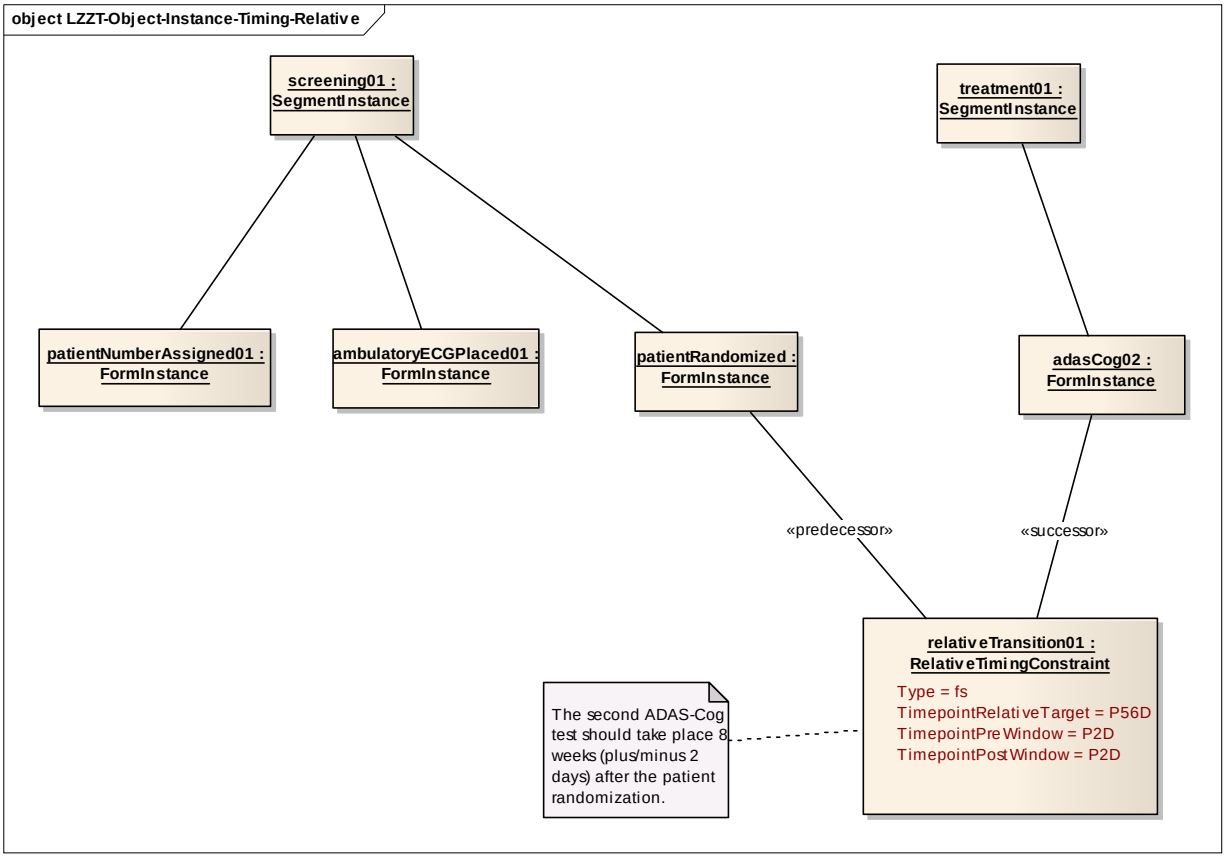
4.2.2 Timing Associated with a Transition



The example above shows a transition timing where ambulatoryECGRemoved01 should happen ideally 24 hours after the finish of ambulatoryECGplaced. However, it is allowed for this to happen within the 23-25 hour window.

```
Example XML goes here.
```

4.2.3 Relative Timing between Activities



The example above illustrates a relative timing constraint between 2 activities that are not connected by a transition. It shows that the adasCog02 form instance should be started 56 days after the finish of the patient randomization. However it is allowed to happen 2 days before, or two days after the target time calculated by the execution runtime.

```
Example XML goes here.
```

5 Element Reference

5.1 ODM V1.3 Elements

As this is an extension of the CDSC ODM V1.3 schema, elements of that schema will be used in the design of the trial design model. The following descriptions cover the major elements that are used in ODM and are taken from the ODM reference document. For a full explanation of these elements, refer to ODM V1.3 documentation.

Element	Description
ODM13::ConditionDef	A ConditionDef defines a *boolean* condition. A *Description* sub-element should be provided and must include a prose description. Note that *Description* elements support internationalized text strings to aid in localizing the trial information.
ODM13::FormDef	A type of form the can occur within a study.
ODM13::ItemDef	Describes an item that can occur within a study and is collected about a patient. An example would be a question on a CRF – the ItemDef would include both the question to be asked and also information about the answers that are acceptable.
ODM13::ItemGroupDef	A set if data points that are logically grouped together. This may correspond to a section within a form.
ODM13::MetadataVersion	A MetadataVersion defines the types of study events, forms, item groups and items that form the study. This is study metadata as opposed to the actual study data itself.
ODM13::Protocol	The Protocol element lists the kinds of study events that can occur within a specific MetadataVersion of a Study.
ODM13::Study	This element collects static structural information about an individual study.
ODM13::StudyEventDef	Packages a set of forms with the intent that these will be filled out at the same period in time.

5.2 Trial Design Specific Elements

5.2.1 Strucutural

Element	Description	Relevant Attributes

Activity	The *superclass* [generalization] for something that happens during a study and needs to be included in the trial design. If an Activity has no associated ActivityTransition then it is essentially a dead-end to an execution runtime.	**SubsequentSchedulingBasis:** Specifies how subsequent Activities that are timed relative to this should be scheduled. This would be used by an execution runtime for an individual patient. Allowable values: PLANNED – use the planned [target] time to schedule the subsequent Activity. ACTUAL – use the actual time the Activity took place to schedule the subsequent Activities.
Arm	TBD: Whether Arms need Def/Ref mechanism	
ArmDef	TBD: Whether Arms need Def/Ref mechanism	
ArmInstance	TBD: Whether Arms need Def/Ref mechanism	
CellDef		
CellInstance		
EpochDef		
EpochInstance	An instance of an EpochDef used in building the actual flow through a study.	
FormInstance	A type of Activity. Defines when a form should be filled out in the conduct of a study.	
SchedulingActivity	A type of Activity. Allows for branching and timing to be included in the trial without being tied to the completion of a form.	
SegmentDef		
SegmentInstance		
StudyEventInstance		
TrialStartFinish	A type of Activity. Defines the start and finish points for a patient within the study. There may only be one *start* and one *finish*. If a patient is to start in the trial, an execution runtime would put them on the *start* and progress from there.	**Type:** Identifies whether this is the start or finish node. Allowable values: START – identifies this as the start point FINISH – identifies this as the finish point.

5.2.2 Workflow

Element	Description	Relevant Attributes
ActivityTransition	Defines the transitions that can happen after an Activity has completed. This is required for an execution runtime to be able to determine what happens next.	

Switch	Provides a branching mechanism for the study. The Switch allows for single-path branching or parallel-path branching.	**Type:** Identifies how the Switch should be evaluated by an execution runtime. Allowable values: PARALLEL – All TransitionDestinations whose conditions evaluate to true should be followed. SINGLE – The first TransitionDestination whose conditions evaluate to true should be followed. Evaluation of other TransitionDestinations should be stopped.
TransitionDestination	Identifies the Activity that the execution runtime should branch to. If ODM13::ConditionDefs are associated with the TransitionDestination then all conditions must evaluate to true in order for the TransitionDestination to be followed. If no ODM13::ConditionDefs are associated then the condition is automatically true.	

5.2.3 Timing

Element	Description	Relevant Attributes
AbsoluteTimingConstraint	Allows designers to specify that the associated Activity happen between specific times. For example, an activity must happen between 0900 and 1100 (with a target time of 1000).	**ActivityOID:** The Activity that the constraint is associated with. **TimepointAbsoluteTarget:** The ideal time that the associated Activity should happen. **TimepointPostWindow:** The window [duration] after the target time that it is allowable for the Activity to take place. **TimepointPreWindow:** The window [duration] before the target time that it is allowable for the Activity to take place.

RelativeTimingConstraint	Allows designers to specify that an Activity must happen period of time after some other Activity related timepoint.	**PredecessorActivityOID:** The predecessor Activity can be thought of as the anchor for the constaint. It is the one that the successor Activity is relative to. **SuccessorActivityOID:** The successor Activity is the one that happens relative to something else. **TimepointGranularity:** Defines the window around the target time in more general terms. If present, it overrides the Timepoint*Window attributes. Allowable values: PY – anytime during that year PM – anytime during that month PD – anytime during that day PTH – anytime during that hour PTM – anytime during that minute PTS – anytime during that second **TimepointPostWindow:** The window [duration] after the target time that it is allowable for the Activity to take place. **TimepointPreWindow:** The window [duration] before the target time that it is allowable for the Activity to take place. **TimepointRelativeTarget:** The target interval [time/duration] between the predecessor and successor activites. **Type:** The type of relative timing between predecessor and successor. Allowable Values: SS – start to finish FF – finish to finish SF – start to finish FS – finish to start

TransitionTimingConstraint	Specifies a relative timing constraint that occurs on a transition rule.	**TimepointGranularity:** Defines the window around the target time in more general terms. If present, it overrides the Timepoint*Window attributes. Allowable values: PY – anytime during that year PM – anytime during that month PD – anytime during that day PTH – anytime during that hour PTM – anytime during that minute PTS – anytime during that second **TimepointPostWindow:** The window [duration] after the target time that it is allowable for the Activity to take place. **TimepointPreWindow:** The window [duration] before the target time that it is allowable for the Activity to take place. **TimepointRelativeTarget:** The target interval [time/duration] between the predecessor and successor activites. **TransitionDestinationOID:** The ID of the TransitionDestination element that this constraint refers to. **Type:** The type of relative timing between predecessor and successor. Allowable Values: SS – start to finish FF – finish to finish SF – start to finish FS – finish to start

6 Appendix: Case Study

7 Appendix: XML Schema

The XML Schema for Trial Design extends ODM using the same pattern as that employed by the CRT-DDS standard. There is a root schema, an extension schema (which defines the extensions to existing ODM elements such as Protocol), and an 'ns' schema, which defines those elements and attributes specific to trial design.
Further, the Trial Design 'ns' schema includes separate schema documents for structural information, workflow information, and timing information.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.cdisc.org/ns/odm/v1.3"
           xmlns="http://www.cdisc.org/ns/odm/v1.3"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tdm="http://www.cdisc.org/ns/TDM_1/v1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">

    <!-- import core XML schema for xml:lang -->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
               schemaLocation="../core/xml.xsd"/>

    <!-- include TDM extensions to core ODM -->
    <xs:include schemaLocation="tdm-extension.xsd"/>

</xs:schema>
```

Sample 7-1: tdm1-0-0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.cdisc.org/ns/odm/v1.3"
           xmlns="http://www.cdisc.org/ns/odm/v1.3"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tdm="http://www.cdisc.org/ns/TDM_1/v1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">

  <xs:import namespace="http://www.cdisc.org/ns/TDM_1/v1.0"
             schemaLocation="tdm-ns.xsd" />

  <xs:redefine schemaLocation="../cdisc-odm-1.3.0/ODM1-3-0.xsd">
    <xs:group name="ProtocolElementExtension">
      <xs:sequence>
        <xs:group ref="ProtocolElementExtension" />
        <xs:group ref="ProtocolElementContents" />
      </xs:sequence>
    </xs:group>

    <xs:group name="StudyEventDefElementExtension">
      <xs:sequence>
        <xs:group ref="StudyEventDefElementExtension" />
        <xs:group ref="StudyEventDefElementContents" />
      </xs:sequence>
    </xs:group>
  </xs:redefine>

  <xs:group name="ProtocolElementContents">
    <xs:sequence>
      <xs:element ref="tdm:Summary"
                  minOccurs="1" maxOccurs="1" />
      <xs:element ref="tdm:InclusionExclusionCriteria"
                  minOccurs="1" maxOccurs="1" />
      <xs:element ref="tdm:Structure"
                  minOccurs="1" maxOccurs="1" />
      <xs:element ref="tdm:Workflow"
                  minOccurs="1" maxOccurs="1" />
      <xs:element ref="tdm:Timing"
                  minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:group>

  <xs:group name="StudyEventDefElementContents">
    <xs:sequence>
      <xs:element ref="tdm:ActivityRef"
                  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:group>

</xs:schema>
```

Sample 7-2: tdm-extension.xsd