Clinical Study Schedule of Activities Specification using FHIR Definitional Resources

Andrew Richardson, Patrick Genyn

Submitted to: JMIR Medical Informatics
on: January 18, 2025

Table of Contents

Clinical Study Schedule of Activities Specification using FHIR Definitional Resources

Andrew Richardson[1]; Patrick Genyn[2] S.Lic, BA, MSc

[1] fhir4pharma Hungerford, Berkshire GB
[2] fhir4pharma Doylestown US

Corresponding Author:
Andrew Richardson

fhir4pharma
Ramsbury House, Charnham Lane
Hungerford, Berkshire
GB

Abstract

Background: The Fast Healthcare Interoperability Resources (FHIR) standard is now well established as a global standard for healthcare information exchange. The value of FHIR is also being actively explored to support clinical and healthcare research in areas such as observational studies using “real world data” (RWD) and in clinical studies intended for regulatory submissions where it plays a key role in enabling direct data capture from clinical sites. The goal of the work for clinical trials is to use FHIR resources to define study schedules of activities (SoA) to support operational implementation at research sites. A HL7 FHIR Implementation Guide (IG) has been published that can define basic SoAs, but this does not offer solutions to define some types of study scheduling (e.g. treatment cycles) or other scheduling requirements (e.g. conditional switching such as that found in vaccine studies)

Objective: The objective of this work was to re-investigate how the FHIR definitional resources, particularly PlanDefinition might be extended or revised to enable the model to cover the wider range of SoA scheduling requirements that are commonly encountered in clinical research studies.

Methods: A previously described SoA graph model was used to investigate, extend and test the fundamental requirements for SoA definitions modelling the complex and conditional requirements outlined above. These requirements were then defined/reflected as FHIR definitional resources with, where necessary, FHIR permitted technical solutions (extensions, value sets, etc.). Specification accuracy was tested by comparing the SoA graph model attributes and relationships with those able to be recovered from the FHIR specifications.

Results: Using confirmed graph-based models of clinical SoAs, a SoA FHIR model based on the PlanDefinition resource was developed that can model simple, complex and conditional SoA requirements for a wide variety of SoA use cases. Using example and publically available study SoAs, the SoA FHIR model was able to accurately specify a large range of commonly met study scheduling requirements.

Conclusions: A FHIR model for the specification of clinical trial SoAs has been developed that offers a more comprehensive set of scheduling requirements to be defined than previously considered. Particularly it implements methods for specifying conditional scheduling requirements, and clearly separates SoA activity definition from the study scheduling requirments.

(JMIR Preprints 18/01/2025:71430)
DOI: https://doi.org/10.2196/preprints.71430

Preprint Settings

1) Would you like to publish your submitted manuscript as preprint?

✓ **Please make my preprint PDF available to anyone at any time (recommended).**
Please make my preprint PDF available only to logged-in users; I understand that my title and abstract will remain visible to all users. Only make the preprint title and abstract visible.

No, I do not wish to publish my submitted manuscript as a preprint.

2) If accepted for publication in a JMIR journal, would you like the PDF to be visible to the public?

✓ **Yes, please make my accepted manuscript PDF available to anyone at any time (Recommended).**

Yes, but please make my accepted manuscript PDF available only to logged-in users; I understand that the title and abstract will remain v

Yes, but only make the title and abstract visible (see Important note, above). I understand that if I later pay to participate in <a href="http

Original Manuscript

Introduction

Healthcare interoperability standards such as the Fast Healthcare Interoperability Resources (FHIR) [1] are providing new methodological approaches for the collection, collation and confirmation of clinical research data for both observational studies and trials supporting product regulatory submissions [2,3,4]. To be successful clinical research studies require that (a) the correct data is available to answer the research question, and (b) these data are collected at the correct times. These are detailed in the study protocol, where the SoA, usually in the form of a square table, provides the key data and scheduling requirements.

In a previous paper [5], a graph based minimum viable set of characteristic attributes needed to define a study's SoA was developed together with commentary on the range of "variations on the theme" met in different clinical study types and therapeutic areas. Operational use cases that depend on the SoA were also considered. The resulting graph representations of a SoA were converted to FHIR PlanDefinitions compliant with the HL7 Vulcan Clinical Study Schedule of Activities implementation guide (Vulcan SoA IG) [6].

Whilst the current Vulcan SoA IG can model a wide variety of clinical study SoAs, it is recognized that it is principally limited to defining relatively simple SoAs. It works well to convert SoA tables to FHIR resources, but , for example, does not *per se* have methods to define schedules that repeat (cycle), an essential part of most oncology studies. Similarly, it does not offer methods for conditional switching or to select different permitted (multiple) study paths. It may be coerced in some cases to manage specific situations, but this is not ideal when these "fixes" are key scheduling requirements. This can mean that a Vulcan SoA IG compliant SoA will specify only part of the total scheduling options a specific study requires. Subsequent use by consuming applications such as electronic health record systems (EHR) may or will require additional tooling to implement all study scheduling variations and control.

With consideration of previous work, the work here has investigated (a) what attributes or modifications to a graph model are needed to cover the extended use cases outlined above, and (b) develop a FHIR PlanDefinition representation that can communicate these requirements without additional tooling. The primary objectives of this work were:

- to develop methods for defining multiple SoA paths within a single graph
- to develop methods for defining SoA conditional scheduling requirements
- to develop a human understandable syntax to support specific study specification
- to reflect these requirements as FHIR definitional resources
- to test and confirm converting the graph representations to FHIR and vice-versa

Methods

Graphical Schedule of Activities Definition

The adopted methodological approach was to (a) build on previous work to investigate and develop the necessary attributes required to meet the objectives described above, and (b) develop and test FHIR resource options to accurately describe and exchange these requirements.

Table 1 shows a part of a schedule of activities as might be presented in a protocol, and Figure 1 a representation in graph form of the primary schedule elements, and the minimum set of characteristic attributes required to reflect it in this form is shown in Table 2 (see [5] for more details). Two SoA graph node types are used in this model: ‘interactions’, modelling study events, visits or other direct or indirect contacts with research subjects, and ‘activities’, defining the tasks required to be undertaken to meet the study objectives. These have a simple and easily understood correspondence with the tabular forms found in protocols. These were used to convert the SoAs into Vulcan SoA IG compliant FHIR resources [5].

Table 1. Example (partial) of a schedule of activities as presented in protocols

Phase	Screening	Study Period				Unscheduled
Visit ID	1	2	3	…	6	U
Visit Timing	-D28 – D1	D1	D7	…	D28	As required
Activity						
Demographics	X					
Medical History	X					
Inclusion/Exclusion Criteria	X		X			
Vital Signs	X	X	X	X	X	X
Procedure	X	X				
:						
Concomitant Medication	X	X	X	X	X	X
AE/SAE	X	X	X	X	X	X

Table 2. SoA Graph Characteristic Attributes [a]

Attribute	**Minimal Required Attribute**	**Notes/Example**
Nodes		
nodeID	Yes	Universally Unique Identifier (UUID)
type	Yes	‘Interaction’ or ‘Activity’
subtype	No	e.g. Clinic visit, telephone call
name	Yes	Protocol name of interaction or activity
description	No	Description of interaction or activity
plannedTiming	Yes	Schedule timing (D1 etc.)
referenceTimepoint	Yes	Schedule t(zero)
plannedWindow	No	Schedule timing permitted variance
plannedDuration	No	Duration of interaction or activity (e.g. 24hrs)
Edges		
edgeID	Yes	Universally Unique Identifier (UUID)
transitionType	No	Timing relationship between nodes

[a] Adapted from [5]

Multiple SoA Paths

Standard graph methods were used to develop and test the attributes necessary to model and define the routing objectives (use cases) established earlier. The primary considerations were (a) to reflect

different study scheduling options accurately using property graph methods, and (b) to ensure the resulting models had a user-friendly correspondence to standard clinical trial scheduling concepts. Only methods using directed graphs were considered.

Conditional Scheduling

Methods to accurately model conditional scheduling requirements within a single SoA graph initially used path analysis to identify the set of adjacency matrixes required for each scenario. These were then used to develop a specification syntax that could define any specific conditional requirement, that, with appropriate tooling, can be implemented such that any sub-graph can be extracted from the primary specification. Consideration here was given to ensuring (a) permitted routes, such as those required by schedules with different treatment arms could be defined, and (b) that the method could support controlling individual schedules dynamically (e.g. as individual research participants are reviewed during visits)

FHIR PlanDefinition

The Vulcan SoA IG [6] was used as the starting point for reviewing and evaluating methods to reflect the approaches above as FHIR resources. The primary resources used were *PlanDefinition, ActivityDefinition* and the associated profiles and extensions that are required to configure a complete description for a specific study (i.e. *ReseachStudy, ResearchSubject* etc.). The main *PlanDefinition* elements investigated were *action.condition, action.relatedAction, action.timing*, and the five *action.<xxx>Behaviors*. All work was undertaken using version FHIR Release #5 (5.0.0) resources [1].

Model Testing and Proof of Concept

Graph database methods were used to develop and test the specification methods. Proof-of-concept (PoC) example graphs were built using the Python generalised programming language [7], the NetworkX graph and network libraries [8] and the pandas data analysis library [9].

FHIR Resource examples were generated using the Python fhir.resources library [10,11] and HL7 FHIR Shorthand (FSH) definitions [11]. The yED graph editor was used to create the visual graph presentations and also as an editor to create specific test examples [12]. The accuracy of the FHIR Resources versus the graph model was confirmed by visual and programmed comparison of the resulting definitions. The FHIR Resources generated from each PoC example were confirmed as valid FHIR Resources by loading them to publicly available FHIR endpoints, recovering the specifications using FHIR searches, and confirming that the full original specifications could be recovered without information loss.

SoA Example

Examples and illustrative figures used in this paper are based on the SoA graph shown in Figure 1.

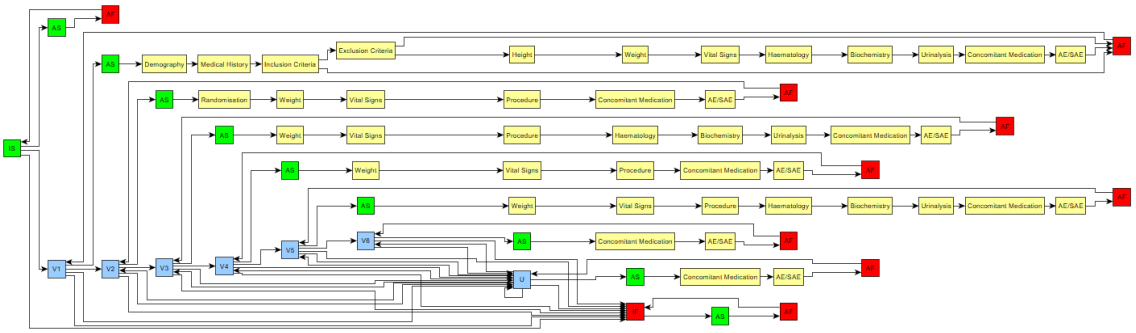


Figure 1: Example study SoA directed graph. The SoA has 6 planned visits and 1 unscheduled visit (blue). The activities at each visit are shown in yellow. The green and red nodes delineate the start and finish of graph instantiation (IS, IF), and the activities to be undertaken contiguously (AS, AF). (for details see [5])

Results

Multiple SoA Paths

Figure 2 illustrates the general problem that even a simple study design implies when considering how to define all protocol defined permitted paths. Figure 2a shows how the scheduled visits of Table 1 might be presented as a directed graph. The 'unscheduled' visit, however, 'floats' independently as it is to be used on an 'as required' basis over the period of the planned visits.

Defining all permitted paths (i.e. extend Figure 2a to add all protocol implied paths) is shown in 2b and is easily achieved simply by adding appropriate visit-visit edges. Similarly, all potential activity sequencing options can be defined using the same method (Figure 3).

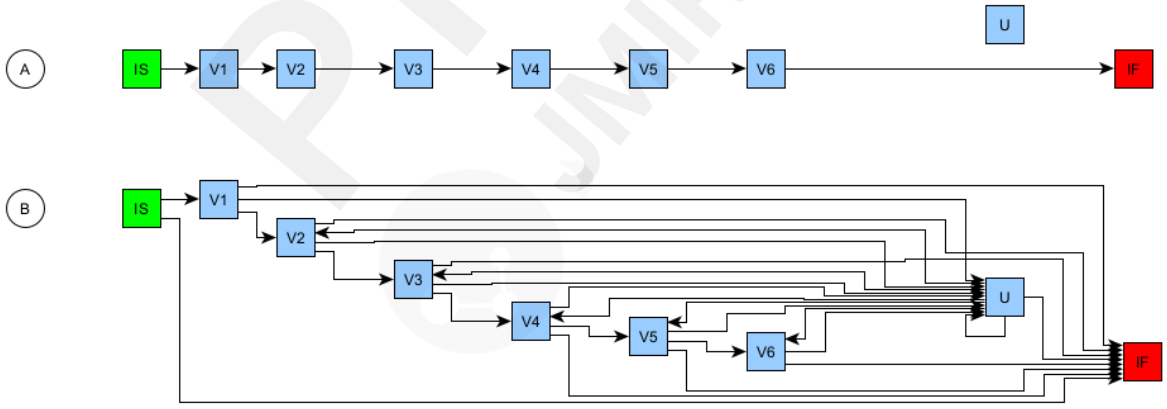


Figure 2: SoA directed graph representations of the visit schedule described in Table 1. (A) protocol explicitly defined schedule with 'floating' unscheduled visit, (B) expanded version with all implied permitted paths defined. Two important implied paths are now present: routes for a subject to leave the study at any point (e.g. V1>IF), and routes to and from unscheduled visits, if required.

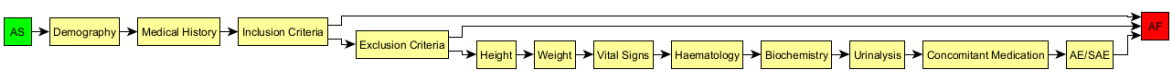


Figure 3: SoA directed graph representation of the visit V1 activities from Figure 1 showing the protocol specified order (left to right) with the added requirements for those cases where (a) the inclusion criteria is not met, or (b) exclusion criteria are present. These paths formally define how to finish the visit 'early'. The resulting visualizations, although busy, remain user-friendly from a review or QC perspective.

Whilst no additional node or edge attributes beyond those in Table 2 are needed to define all permitted paths, ensuring timing calculations for any path cannot be achieved using planned timings alone. Here the problem is that the scheduled planned timings are node (visit) attributes, whilst calculating the timings for any path requires a summation of the transitions along whichever path is selected (Figure 4). The addition of a *transitionDelay* edge attribute (Table 3) was found to provide the necessary required timing information, and also provides the basis for a key timing consistency check since:

plannedTiming (V_n) = sum(*transitionDelay*) from *referenceTimepoint to* V_n

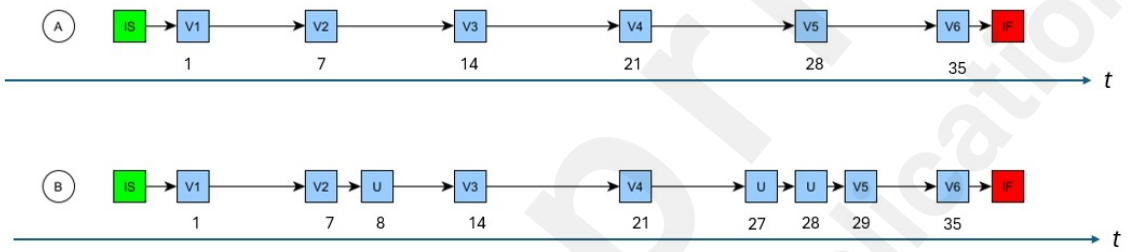


Figure 4: Timing calculation attributes (a) per protocol schedule together with the relative day on which the visit occurred. (b) the same schedule but with 3 unscheduled visits. The first unscheduled visit (D8) has no effect on the planned schedule, whereas the second and third caused V5 to occur on D29 rather than D28. In this case the relative day of the visits cannot be determined from the planned timings alone.

Table 3: Edge attributes required for path timing calculations

Attribute	**Minimal Required Attribute**	**Notes/Example**
Edges		
transitionDelay	Yes	'Wait' time before moving from source to target nodes (i.e. V1>V2 7d)
transitionWindow	No	Permitted transitionDelay variance
transitionType	No (default)	SS (default), SF, FS, FF

Conditional Scheduling Attributes

Figures 2b and 3 also illustrate that there are many cases where specific conditions require different (but permitted) routes to be followed. Some cases are generic and are present in any study (e.g. subject's right to withdraw at any time), some are protocol specific (e.g. if subject is male, a pregnancy test is not required), and some complex, as illustrated in Figure 2b for managing all 'unscheduled' path options (e.g. an 'unscheduled' visit following V_n cannot return to early visits,

only being able to proceed to the next planned visit).

To accurately carry all conditional scheduling requirements within a single SoA graph it was found that three conditions are required:

1: a method to specify (and therefore recognize) a graph within the graph,
2: a syntax to define how to select a sub-graph and/or restrict access to identified paths in the graph,
3: a method that can be implemented as a dynamic graph (i.e. the graph changes over time)

Both graph node attributes and edge attributes in the model here could be used to hold conditional scheduling information. Adding conditions to the nodes (e.g. [visit] *repeatAllowed: true/false*) was found to satisfy some standard requirements (e.g. can a visit be repeated? *V2: repeatAllowed: false*, and *Unscheduled: repeatAllowed: true*), but other requirements very poorly (particularly what routes are available following an unscheduled visit?). Adding conditional requirements as edge attributes were found to accurately and easily model all the SoA use cases tested (Table 4). This approach was also user-friendly, as only the conditions on each edge (transition) are needed to ensure the use of any specific path (i.e. by answering the question "under what conditions can this path be selected/is not available?", and this can also include multiple conditions).

Table 4: Edge attribute for conditional SoA scheduling

Attribute	**Minimal Required Attribute**	**Notes/Example**
Edges		
transitionRule	Yes	rule specification(s) able to be resolved to true (path can be selected) or false (path unavailable)

Conditional Scheduling Syntax

A parameterized specification syntax was developed to hold the edge attribute condition(s) that can be consumed as input(s) to a truth table engine that would resolve all edge conditions to the single "true" or "false" of Table 4. Simple *{'function':'inputs'}* pairs were used to specify the "true" or "false" state at runtime for that condition (e.g. *{'consentObtained': true}*) or any combination of conditions.

The following examples show how several commonly required SoA conditional requirements can be defined using this approach.

Dynamic Graph Specification

The primary use case here is the definition of permitted paths through the study, including, but not limited to, the primary path. For example, in Figure 2b, paths exist to and from 'Unscheduled' to all 'Visits' permitting return to the primary schedule. 'Unscheduled' visits however cannot return to a previously visited 'Visit' (discussed earlier). Two rules are required for this case: one to confirm the existence of previously visited visits, and one to ensure that all 'future' visits have not yet been visited. The following example shows the rules required on edge U>V3 in Figure 2b, will ensure at runtime that only this path is available:

{'interactions_exist': ['IS', 'V1','V2']}, {'interactions_not_exist': ['V3', 'V4', 'V5', 'V6', 'IF']}

Similar rule pairs when applied to all U>V* edges can then ensure that only ‘forward’ paths are available for selection.

Conditional Activity Selection

Restricting, adding or skipping activities dependent upon subject conditions is a very common feature of SoAs, with the details often provided as footnotes to SoA tables. Examples include requiring pregnancy testing if subject is female (and conversely not requiring this if male) and not proceeding with further tasks if inclusion criteria cannot be met (e.g. Figure 3). Example rules for these cases are simple to define, but consideration of all edge options was found usually to be required e.g.:

On edge to ‘Exclusion Criteria’ {‘if_criteria_met’:true}
On edge to ‘AF’ {‘if_criteria_met’:false}

Conditional Repeats

An important feature of many SoAs is a requirement to repeat activities (e.g. blood pressure measurements) or to repeat visit blocks. This last use case is particularly important in the case of many oncology studies where repeating treatment cycles is an inherent part of the study design, and will have some limit on number of cycles and/or requirements to exit the study if too many cycles are proving necessary. Typical edge rules for controlling repeat/cycle situations include:

Repeat BP measurement 3x. On edge to ‘self’: {‘repeats’:’<4’}
Limit the number of cycles to a max of 5. On edge to start of cycle: {‘n_cycles’:’<6’}

Subject States, Milestones and Events

In many cases the requirement to select different SoA paths is dependent upon states, milestones or events, as for instance, defined by the FHIR ResearchSubject resource [1]. For example:

To enter an off-label extension study.
On edge to OLE schedule: {‘continue_to_OLE’: true}
If an adverse event occurs.
On edge to AE activity: {‘record_AE’: true}

Multiple Conditions

The syntax also permits any number of conditions to be defined easily within a single SoA graph for any given edge, e.g. to obtain a sample for genetic testing:

{‘studyPhase’:’onStudy’}, {‘sampleObtained’:’true’}, {‘geneticTestingConsent’:’true’},

With the runtime assessment of each condition then serving as the input to a logic engine that would determine a final True/False state.

Undefined SoA Graphs: Implied Edges

Undefined SoA paths can also be recognized using this approach. An Undefined path is defined here

as one that may occur but is not recognized formally within the SoA graph. A good example of this case is the right to withdraw from a study at any time or skipping an activity. Formally recognizing every point that these conditions may occur and providing a defined path may add so much complexity to the graph that it is unreasonable to model it. However, by placing a general requirement on all transitions (edges) of *{'withdrawn': false}* will permit proceeding through the schedule until *{'withdrawn': false}* == "false" (i.e. withdraw is now "true". Using suitable runtime coding the "undefined" transition can be applied to the specific subjects' SoA graph instance.

FHIR SoA PlanDefinition Review

The primary objective of this work was to find methods to enable SoAs to be more fully defined using FHIR resources. The SoA graph review identified the necessary additional requirements needed to satisfactorily model SoA specifications such as those of Figure 1. To be able to successfully define these specifications using FHIR resources they must be able to:

- represent all SoA identified paths
- recognize and 'respond' to conditional cases
- allow consuming applications to be able 'walk' any permitted path

The Vulcan SoA IG [6] was used as the starting point for developing a more comprehensive SoA FHIR model with the goal of extending or modifying it to be able to manage the complex designs, conditions and 'variations on the theme' discussed earlier.

FHIR PlanDefinition soaGraph Definition

From the graph attributes model developed above it follows that in order for the *PlanDefinitions* to fully specify all SoA requirements then it needs to hold a definition of the SoA graph not the SoA table. Reviewing the use of the *PlanDefinition.action* and *PlanDefinition.action.action* elements the basic graph node/edge relationship can be defined. Specifically, with nodes (visits) mapped to *PlanDefinition.actions* and edges(transitions) mapped to *PlanDefinition.action.actions* all the relationships defined, for example, by Figure 1, can be specified within a single *PlanDefinition.*

PlanDefinition.action in its standard form does not support those elements (attributes) to hold all the necessary SoA specification details. These have been added using two extensions – *soaTimePoint* and *soaTransition* (Figure 5) – to hold the graph information, which in turn then form the basis for a *soaPlanDefinition* profile.

```
Extension: SOATimePoint
Id: soaTimepoint
Title: "SoA TimePoint Specification"
Description: "SoA TimePoint Attribute Extension"
// Limit the context to PlanDefinition action
* ^context[+].type = #element
* ^context[=].expression = "PlanDefinition.action"
* extension contains
    soaTimePointType 0..1 and
    soaPlannedTimePoint 0..1 and
    soaPlannedRange 0..1 and
    soaReferenceTimePoint 0..1 and
    soaRangeFromTimePoint 0..1 and
    soaPlannedDuration 0..1
* extension[soaTimePointType].value[x] only string // interaction or activity
* extension[soaPlannedTimePoint].value[x] only SimpleQuantity // visit day etc.
* extension[soaPlannedRange].value[x] only Range // visit window
* extension[soaReferenceTimePoint].value[x] only string // reference visit for planned time
* extension[soaRangeFromTimePoint].value[x] only string  // calculate visit window from timepoint X
* extension[soaPlannedDuration].value[x] only Duration // duration of the visit (1d, 1w) once started

Extension: SOATransition
Id: soaTransition
Title: "SoA Transition Specification"
Description: "Specifies SoA Transition Attributes"
// Limit the context to PlanDefinition action.action
* ^context[+].type = #element
* ^context[=].expression = "PlanDefinition.action.action"
* extension contains
    soaTargetId 0..1 and
    soaTransitionType 0..1 and
    soaTransitionDelay 0..1 and
    soaTransitionRange 0..1
* extension[soaTargetId].value[x] only string // transition target UUID
* extension[soaTransitionType].value[x] only string // calculate transition wait from - to
* extension[soaTransitionDelay].value[x] only Duration  // wait time between states
* extension[soaTransitionRange].value[x] only Range // transition permitted window
```

Figure 5: FHIR extensions (as FSH) used to associate soaGraph node and edge attributes with *PlanDefinition.action* (SOATimePoint) and *PlanDefinition.action.action* (SOATransition) respectively. Attributes from Richardson, Table 5 [5].

FHIR PlanDefinition soaGraph Condition and Selection Definition

Conditional SoA requirements can now be specified using the *PlanDefinition.action* element *condition* (applied to *PlanDefinition.action.action*) with each edge having available all or any true/false conditions for that transition. When combined with *PlanDefinition.groupingBehaviour* and *PlanDefinition.selectionBehaviour* path selection can be restricted to one path only, with only those transitions that are permitted being available for selection. Figure 6 shows the FSH specification for node V2 of Figure 2.

```
// action [V2] ********************************************************************************************** //
* action[0].id = "342da14b-5c82-4a2d-bdcf-7fd7cec428fa"
* action[=].title = "V2"
* action[=].description = "V2"
* action[=].definitionCanonical = "http://fhir4pharma.com/Encounter/V2"

// action soaTimePoint attributes [V2] //
* action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaPlannedTimepoint"
* action[=].extension.extension[0].url = "soaPlannedTimePoint"
* action[=].extension.extension[=].valueQuantity = 7 'd'
...etc...

// action transition grouping and selection [V2 > ?] ******************************************************** //
* action[=].groupingBehavior = #visual-group
* action[=].selectionBehavior = #exactly-one

// action.action soaTransitions #1  [V2 > V3] *************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[0].extension.extension[0].url = "soaTargetId"
...etc...
// soaTransitions #1  [V2 > V3] conditions //
* action[=].action[=].condition[0].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_exist':['IS','V1']}"
* action[=].action[=].condition[+].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_not_exist':['V3','V4','V5','V6','IF']}"

// action.action soaTransitions #2  [V2 > IF] *************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[+].extension.extension[0].url = "soaTargetId"
...etc...
// action.action soaTransitions #2  [V2 > IF] conditions //
* action[=].action[=].condition[0].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'withdraw':True}"
* action[=].action[=].condition[+].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_not_exist':['IF']}"

// action.action soaTransitions #3  [V2 > U] **************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[+].extension.extension[0].url = "soaTargetId"
...etc...
* action[=].action[=].condition.kind = #start
* action[=].action[=].condition.expression.language = #text/plain
* action[=].action[=].condition.expression.expression = "{'to_unscheduled':True}"
```

Figure 6: FSH annotated version of *PlanDefinition* definition of visit V2 (*.action*) and its associated soaGraph transitions: V2>V3, V2>IF (withdrawal), V2>U (unscheduled visit) (*.action.action*). V2 attributes are defined using the …*action.soaPlannedTimepoint* extension. The selection behavior is defined by …*action.groupingBehaviour* and …*action.selectionBehaviour.* For each path from V2 the conditions that must be met for that path to be available for selection are given in … *action.action.condition(s).*

Proof of Concept

All interim and final products methods and results described above were validated using the testing schedule shown in Figure 7. Once interim inconsistencies were reviewed and resolved no SoA information loss was present after recovering a soaGraph from a FHIR *PlanDefinition* (Figure 7 test cycle 1) or after loading and recovery from test FHIR servers (test cycle 2).

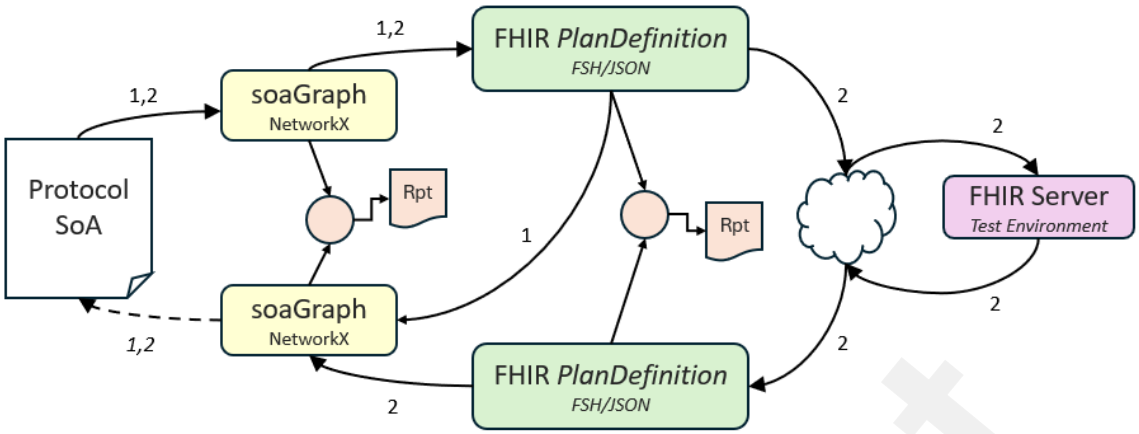


Figure 7: Proof of Concept testing overview. The numbers highlight the two principal testing cycles used. Recovered products were compared with the original for reviewing structural equivalence and information loss throughout the development process.

More than 25 studies, ranging from relatively simple designs as in Figure 1 to complex studies incorporating cycles and multiple SoAs (not shown) were used to test and validate the approach. All tested could be accurately defined, with the most frequent finding during this exercise being that it highlighted inconsistencies in the protocol specifications themselves. It also lends itself to defining SoAs more succinctly, and potentially more accurately. Figure 8 is a template for studies that include cycles and shows that each required visit (interaction) is only defined once. With appropriate edge (transition) attributes and conditions this can be modified to define many specific study scenarios.

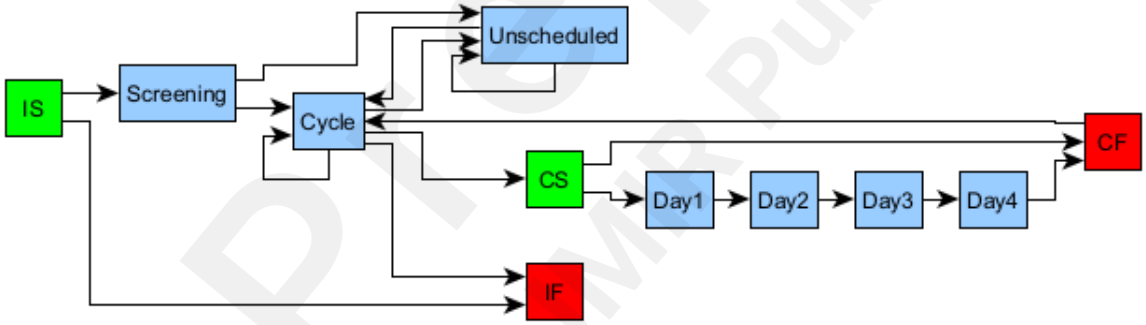


Figure 8: Template soaGraph for studies with cycles. Using appropriate edge attributes and conditions it can be modified for many study designs. The green and red nodes are included to delineate graph instantiation (IS, IF), and the start and finish of the treatment cycle (CS, CF).

Discussion

The HL7 FHIR standard is becoming, if not already, the *de facto* healthcare interoperability standard [2,13,14,15,16]. The primary source for many clinical trial data is the EHR, and studies usually require manual transcription of these data into electronic data capture systems (EDCs). For both quality and volume reasons this is often not optimal and has led to efforts to obtain data directly from EDCs (Direct Data Capture – DDC) [17,18,19, 20].

The study protocol and the SoA provide the primary definitional source for a study's required

research data and details of other operational requirements. The value of definitional FHIR resources to support clinical research and similar initiatives has been recognized in the Specialized Definitional Artifacts category [1] .

The *PlanDefinition* resource is the primary resource for defining "a pre-defined group of actions to be taken in particular circumstances". The Vulcan SoA IG [6] has developed a set of profiles using this resource to define SoAs. As commented earlier, this model does not include methods to define either conditional scheduling, certain study designs (notably cycles) or the scheduling of expected but not planned events (particularly "unscheduled visits"). This work has re-evaluated how *PlanDefinition* might be modified./revised/extended to model these situations.

Principal Results

Using a systematic review of the relationships and attributes required to define the SoA characteristics discussed above, a graph-based method has been developed that can manage all these cases. Then, by associating *PlanDefinition.action* and *PlanDefinition.action.action* directly with graph nodes and edges respectively, SoAs with all scheduling variations, conditional paths and methods to manage planned but unscheduled events can be defined within a single *PlanDefinition.* Recognizing that an SoA "activity" has both timing and task elements (i.e., is "X" in tabular SoAs) the model formally disassociates the scheduling information from the task/activity definitions which can then be linked using the *PlanDefinition.action.relatedAction.targetId* element. This approach also enables SoA activities as described in study protocols (or not described) to be fully traceable in the SoA schedule, but specified fully using other appropriate FHIR definitional resources (e.g. *ActivityDefinition, ObservationDefinition* etc.)

Limitations

The methods here were developed using FHIR Release #5 (V5.0.0) [1] and are not necessarily compatible with earlier FHIR versions.

Comparison with Prior Work

Earlier work based both on using graph methods for SoA definition [21,22,23] and FHIR for interoperability [24,25,26]] has shown the value of this approach for communicating sponsor protocol requirements systematically for implementation in EHRs and similar systems [4,18,19,27,28,29,30]. The work here has reevaluated the approach to specifying SoAs as FHIR *PlanDefinitions* to find solutions to several key required SoA characteristics not addressed previously. The model described here uses a radically different conceptual approach redefining several *PlanDefinition* elements to be able to hold a graph representation of a SoA. The current method can successfully represent a wider range of SoA concepts than before and can also specify a large range of other SoA use cases that are key to many protocol designs (not shown). Since here all SoA requirements can be accurately specified within a single *PlanDefinition* this should simplify implementation by consuming applications. It is also clear that, with modification or the use of standard *PlanDefinition* elements, that it can support other important SoA use cases not directly considered here (e.g. protocol amendments with SoA consequences)

Conclusions

The methods described here offer an alternative approach to defining clinical study SoAs using FHIR

definitional resources than previously published and may offer advantages where some key requirements not addressed by other proposed approaches are required.

Acknowledgements

None

Conflicts of Interest

None

Abbreviations

RCT: randomized controlled trial
SoA: schedule of activities

Multimedia Appendix 1

None

References

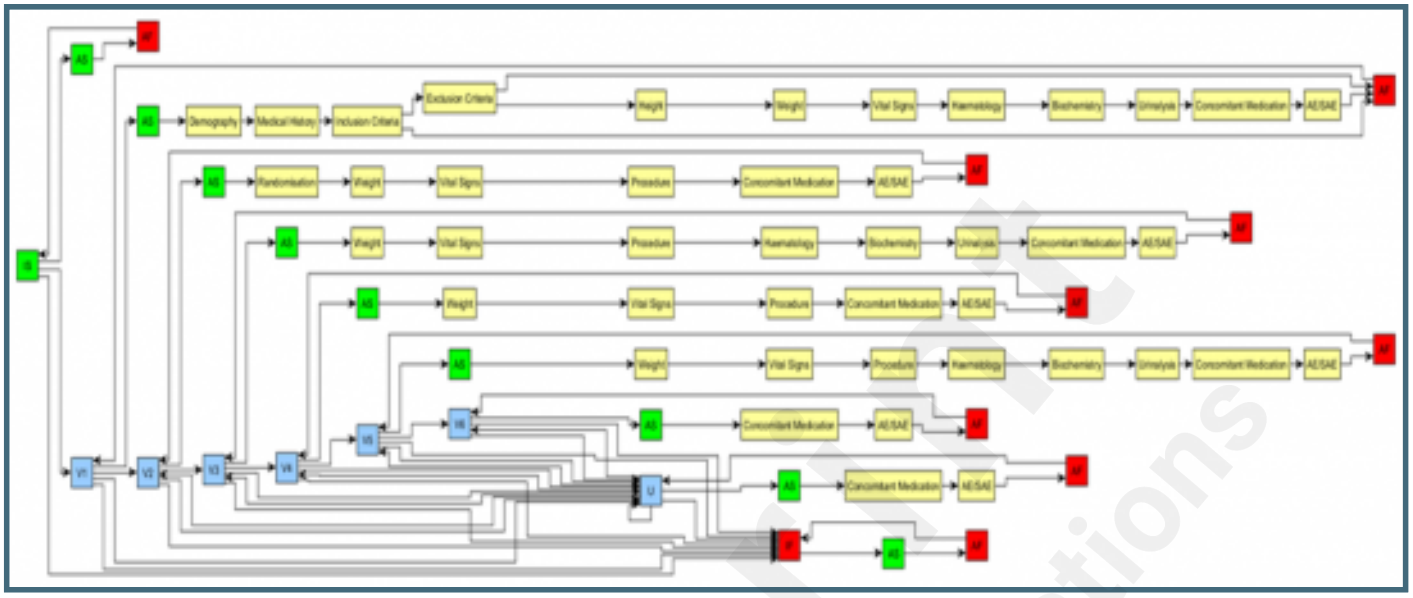
1. HL7 FHIR, “FHIR v5.0.0.” Accessed: Jan. 02, 2025. [Online]. Available: https://hl7.org/fhir/
2. N. Pimenta, A. Chaves, R. Sousa, A. Abelha, and H. Peixoto, “Interoperability of Clinical Data through FHIR: A review,” *Procedia Comput Sci*, vol. 220, pp. 856–861, Jan. 2023, doi: 10.1016/J.PROCS.2023.03.115.
3. “Real-world data: assessing electronic health records and medical claims data to support regulatory decision-making for drug and biological products.”
4. A. Chatterjee, N. Pahari, and A. Prinz, “HL7 FHIR with SNOMED-CT to achieve semantic and structural interoperability in personal health data: a proof-of-concept study,” May 01, 2022, *MDPI*. doi: 10.3390/s22103756.
5. A. Richardson, “Representing Clinical Study Schedule of Activities as FHIR Resources: Required Characteristic Attributes,” *Journal of the Society for Clinical Data Management*, vol. 5, no. 2, Aug. 2024, doi: 10.47912/JSCDM.266.
6. HL7 Vulcan SoA WG, “HL7 Vulcan Clinical Study Schedule of Activities IG.” Accessed: Jan. 02, 2025. [Online]. Available: https://build.fhir.org/ig/HL7/Vulcan-schedule-ig/
7. Python Software Foundation, “Python.” Accessed: Jan. 02, 2025. [Online]. Available: https://www.python.org/
8. NetworkX, “NetworkX — NetworkX documentation.” Accessed: Jan. 02, 2025. [Online]. Available: https://networkx.org/
9. The pandas development team, “pandas - Python Data Analysis Library.” Accessed: Jan. 02, 2025. [Online]. Available: https://pandas.pydata.org/
10. “fhir.resources · PyPI.” Accessed: Jul. 26, 2023. [Online]. Available: https://pypi.org/project/fhir.resources/
11. HL7 FHIR Infrastructure WG, “FHIR Shorthand - FHIR Shorthand v3.0.0.” Accessed: Jan. 02, 2025. [Online]. Available: https://build.fhir.org/ig/HL7/fhir-shorthand/
12. yWorks, “yEd - Graph Editor.” Accessed: Jan. 02, 2025. [Online]. Available: https://www.yworks.com/products/yed
13. R. Saripalle, C. Runyan, and M. Russell, “Using HL7 FHIR to achieve interoperability in patient health record,” *J Biomed Inform*, vol. 94, Jun. 2019, doi: 10.1016/J.JBI.2019.103188.
14. S. K. Mukhiya and Y. Lamo, “An HL7 FHIR and GraphQL approach for interoperability between heterogeneous Electronic Health Record systems,” *Health Informatics J*, vol. 27, no. 3, Sep. 2021, doi: 10.1177/14604582211043920/SUPPL_FILE/SJ-PDF-3-JHI-10.1177_14604582211043920.PDF.
15. S. C. Monteiro and R. J. Cruz Correia, “FHIR Based Interoperability of Medical Devices,” *Stud Health Technol Inform*, vol. 290, pp. 37–41, Jun. 2022, doi: 10.3233/SHTI220027.
16. E. Raso, P. Loreti, M. Ravaziol, and L. Bracciale, “Anonymization and Pseudonymization of FHIR Resources for Secondary Use of Healthcare Data,” *IEEE Access*, vol. 12, pp. 44929–44939, 2024, doi: 10.1109/ACCESS.2024.3381034.
17. Yoshihiro Aoyagi, “Direct data transfer from HIS (Hospital information system) to Sponsor for clinical trials,” 2015, *DIA JAPAN 2015 12th Annual Meeting*.
18. G. Lombardo *et al.*, “Electronic health records (EHRs) in clinical research and platform trials: Application of the innovative EHR-based methods developed by EU-PEARL,” *J Biomed Inform*, vol. 148, Dec. 2023, doi: 10.1016/J.JBI.2023.104553.
19. G. De Moor *et al.*, “Using electronic health records for clinical research: The case of the EHR4CR project,” *J Biomed Inform*, vol. 53, pp. 162–173, Feb. 2015, doi: 10.1016/J.JBI.2014.10.006.
20. J. A. Welker, “Implementation of electronic data capture systems: Barriers and solutions,” *Contemp Clin Trials*, vol. 28, no. 3, pp. 329–336, May 2007, doi: 10.1016/J.CCT.2007.01.001.

21. “Approaches to schedule of activities (SOA) specification for automated implementation.”
22. G. Low, M. Ward, A. Richardson, and T. Biopharma, “Study Designs using FHIR: Schedule of Activities Exchange using FHIR Resources”.
23. A. Richardson, “Approaches to Schedule of Activities (SOA) Specification for Automated Implementation”.
24. P. Genyn, “The role of FHIR Resources in ensuring Semantic Equivalence in EHR2EDC direct data capture”.
25. P. Genyn and A. Richardson, “The role of FHIR Resources in testing and validating an EHR to Sponsor data pipeline,” 2023.
26. H. Leroux, A. Metke-Jimenez, and M. J. Lawley, *Towards achieving semantic interoperability of clinical study data with FHIR*, vol. 8, no. 1. 2017, p. 41. Accessed: Jun. 30, 2021. [Online]. Available: https://jbiomedsem.biomedcentral.com/articles/10.1186/s13326-017-0148-7
27. P. Coorevits *et al.*, “Electronic health records: new opportunities for clinical research,” Dec. 2013. doi: 10.1111/joim.12119.
28. M. Garza, S. Myneni, S. H. Fenton, and M. N. Zozus, “eSource for standardized health information exchange in clinical research: a systematic review of progress in the last year,” Aug. 30, 2021, *Society for Clinical Management*. doi: 10.47912/jscdm.66.
29. N. Laaksonen *et al.*, “Assessing an Electronic Health Record research platform for identification of clinical trial participants.,” *Contemp Clin Trials Commun*, vol. 21, p. 100692, Mar. 2021, doi: 10.1016/j.conctc.2020.100692.
30. K. El Emam, E. Jonker, M. Sampson, K. Krleža-Jerić, and A. Neisa, “The use of electronic data capture tools in clinical trials: Web-survey of 259 canadian trials,” *J Med Internet Res*, vol. 11, no. 1, 2009, doi: 10.2196/JMIR.1120.

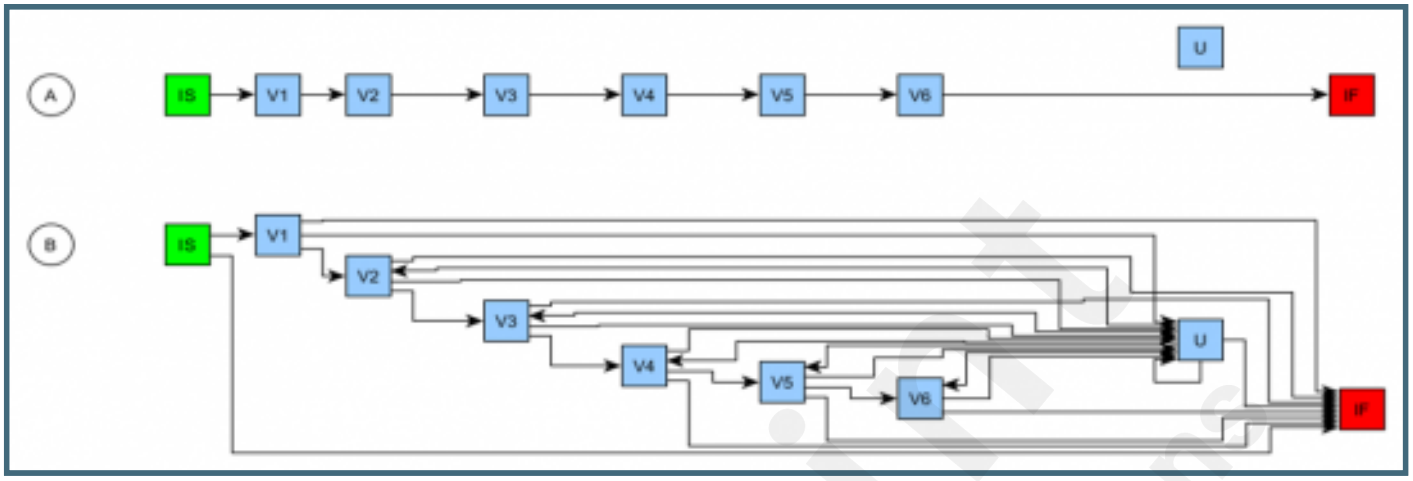
Supplementary Files

Figures

Example study SoA directed graph. The SoA has 6 planned visits and 1 unscheduled visit (blue). The activities at each visit are shown in yellow. The green and red nodes delineate the start and finish of graph instantiation (IS, IF), and the activities to be undertaken contiguously (AS, AF). (for details see [5]).

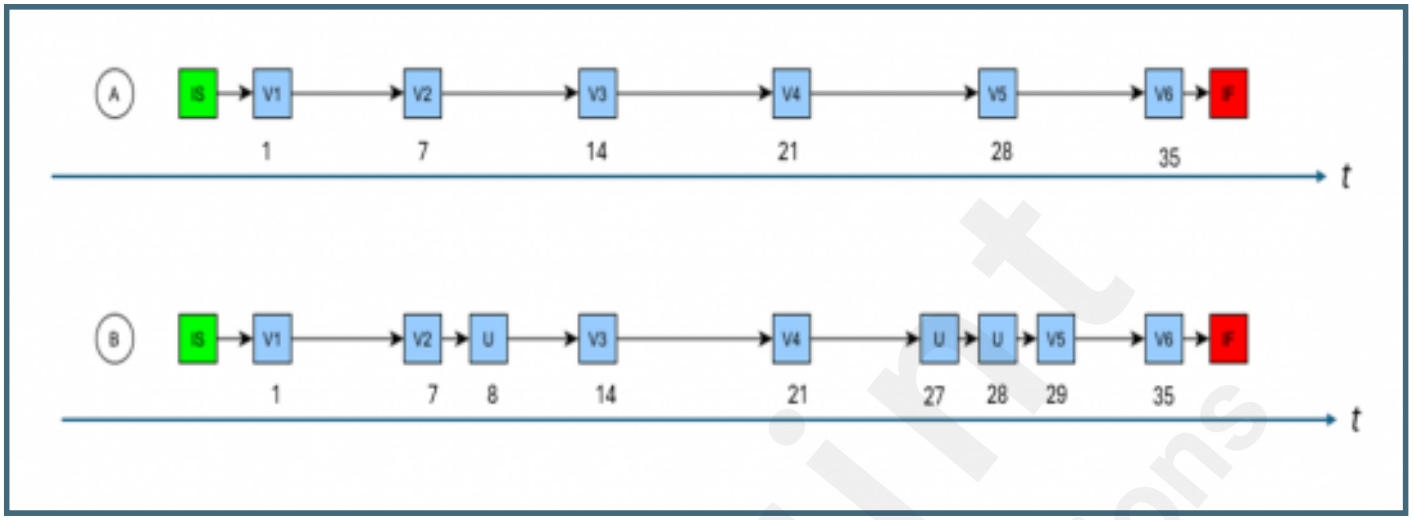


SoA directed graph representations of the visit schedule described in Table 1. (A) protocol explicitly defined schedule with ‘floating’ unscheduled visit, (B) expanded version with all implied permitted paths defined. Two important implied paths are now present: routes for a subject to leave the study at any point (e.g. V1>IF), and routes to and from unscheduled visits, if required.



SoA directed graph representation of the visit V1 activities from Figure 1 showing the protocol specified order (left to right) with the added requirements for those cases where (a) the inclusion criteria is not met, or (b) exclusion criteria are present. These paths formally define how to finish the visit ‘early’. The resulting visualizations, although busy, remain user-friendly from a review or QC perspective.

Timing calculation attributes (a) per protocol schedule together with the relative day on which the visit occurred. (b) the same schedule but with 3 unscheduled visits. The first unscheduled visit (D8) has no effect on the planned schedule, whereas the second and third caused V5 to occur on D29 rather than D28. In this case the relative day of the visits cannot be determined from the planned timings alone.



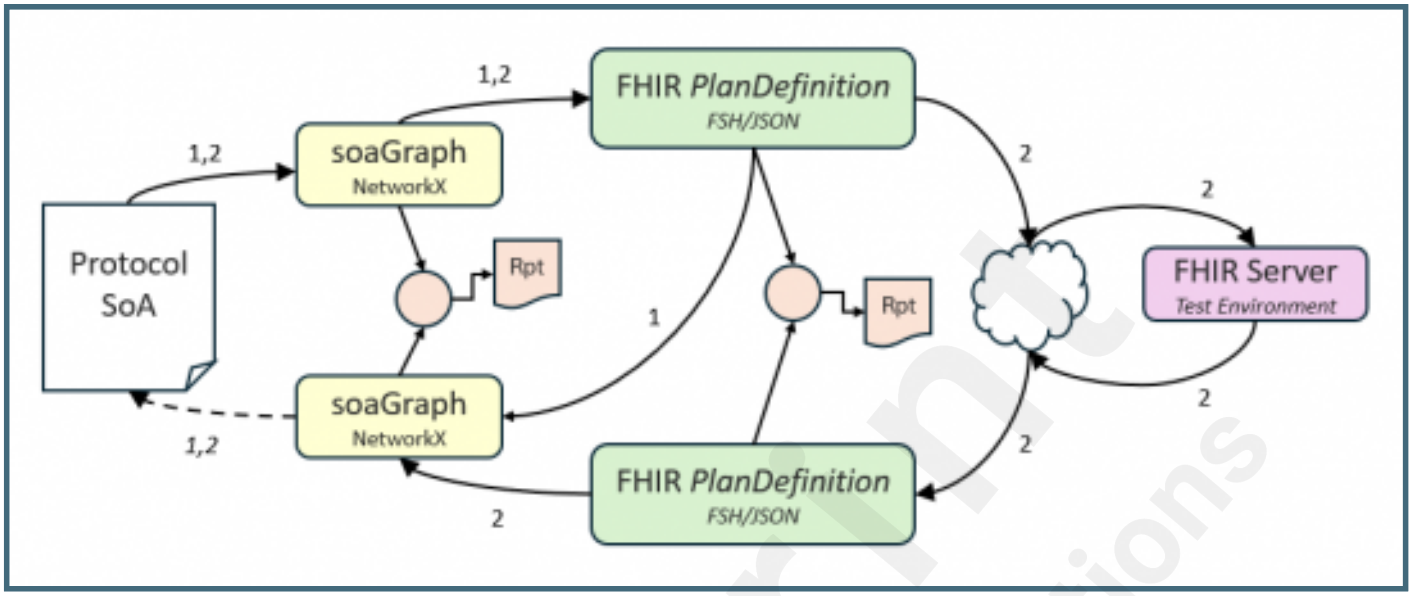
FHIR extensions (as FSH) used to associate soaGraph node and edge attributes with PlanDefinition.action (SOATimePoint) and PlanDefinition.action.action (SOATransition) respectively. Attributes from Richardson, Table 5 [5].

```
Extension: SOATimePoint
Id: soaTimepoint
Title: "SoA TimePoint Specification"
Description: "SoA TimePoint Attribute Extension"
// Limit the context to PlanDefinition action
* ^context[+].type = #element
* ^context[=].expression = "PlanDefinition.action"
* extension contains
    soaTimePointType 0..1 and
    soaPlannedTimePoint 0..1 and
    soaPlannedRange 0..1 and
    soaReferenceTimePoint 0..1 and
    soaRangeFromTimePoint 0..1 and
    soaPlannedDuration 0..1
* extension[soaTimePointType].value[x] only string // interaction or activity
* extension[soaPlannedTimePoint].value[x] only SimpleQuantity // visit day etc.
* extension[soaPlannedRange].value[x] only Range // visit window
* extension[soaReferenceTimePoint].value[x] only string // reference visit for planned time
* extension[soaRangeFromTimePoint].value[x] only string  // calculate visit window from timepoint X
* extension[soaPlannedDuration].value[x] only Duration // duration of the visit (1d, 1w) once started

Extension: SOATransition
Id: soaTransition
Title: "SoA Transition Specification"
Description: "Specifies SoA Transition Attributes"
// Limit the context to PlanDefinition action.action
* ^context[+].type = #element
* ^context[=].expression = "PlanDefinition.action.action"
* extension contains
    soaTargetId 0..1 and
    soaTransitionType 0..1 and
    soaTransitionDelay 0..1 and
    soaTransitionRange 0..1
* extension[soaTargetId].value[x] only string // transition target UUID
* extension[soaTransitionType].value[x] only string // calculate transition wait from - to
* extension[soaTransitionDelay].value[x] only Duration  // wait time between states
* extension[soaTransitionRange].value[x] only Range // transition permitted window
```

FSH annotated version of PlanDefinition definition of visit V2 (.action) and its associated soaGraph transitions: V2>V3, V2>IF (withdrawal), V2>U (unscheduled visit) (.action.action). V2 attributes are defined using the …action.soaPlannedTimepoint extension. The selection behavior is defined by …action.groupingBehaviour and …action.selectionBehaviour. For each path from V2 the conditions that must be met for that path to be available for selection are given in …action.action.condition(s).

```
// action [V2] ********************************************************************************************** //
* action[0].id = "342da14b-5c82-4a2d-bdcf-7fd7cec428fa"
* action[=].title = "V2"
* action[=].description = "V2"
* action[=].definitionCanonical = "http://fhir4pharma.com/Encounter/V2"

// action soaTimePoint attributes [V2] //
* action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaPlannedTimepoint"
* action[=].extension.extension[0].url = "soaPlannedTimePoint"
* action[=].extension.extension[=].valueQuantity = 7 'd'
...etc...

// action transition grouping and selection [V2 > ?] ******************************************************** //
* action[=].groupingBehavior = #visual-group
* action[=].selectionBehavior = #exactly-one

// action.action soaTransitions #1  [V2 > V3] *************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[0].extension.extension[0].url = "soaTargetId"
...etc...
// soaTransitions #1  [V2 > V3] conditions //
* action[=].action[=].condition[0].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_exist':['IS','V1']}"
* action[=].action[=].condition[+].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_not_exist':['V3','V4','V5','V6','IF']}"

// action.action soaTransitions #2  [V2 > IF] *************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[+].extension.extension[0].url = "soaTargetId"
...etc...
// action.action soaTransitions #2  [V2 > IF] conditions //
* action[=].action[=].condition[0].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'withdraw':True}"
* action[=].action[=].condition[+].kind = #start
* action[=].action[=].condition[=].expression.language = #text/plain
* action[=].action[=].condition[=].expression.expression = "{'interactions_not_exist':['IF']}"

// action.action soaTransitions #3  [V2 > U] **************************************************************** //
* action[=].action[=].extension.url = "http://fhir4pharma.com/StructureDefinition/soaTransition"
* action[=].action[+].extension.extension[0].url = "soaTargetId"
...etc...
* action[=].action[=].condition.kind = #start
* action[=].action[=].condition.expression.language = #text/plain
* action[=].action[=].condition.expression.expression = "{'to_unscheduled':True}"
```

Proof of Concept testing overview. The numbers highlight the two principal testing cycles used. Recovered products were compared with the original for reviewing structural equivalence and information loss throughout the development process.



Template soaGraph for studies with cycles. Using appropriate edge attributes and conditions it can be modified for many study designs. The green and red nodes are included to delineate graph instantiation (IS, IF), and the start and finish of the treatment cycle (CS, CF).