

Reporting process

mja@statgroup.dk

2016-02-15

Contents

Introduction	1
Setup	2
Get the process description	2
Reporting process from “Using Define.xml Metadata to Ensure End-to-End Submission Integrity”, slide 15	2
rstepshape	4
Grid for “Diagram of the SW Initiatives”	5
How to make the HTML file	8

Introduction

This note shows how to use RDF to describe the reporting process, and present it graphically.

The reporting processes are taken from:

- Brega, John, and Colins, Linda. ‘Beyond OpenCDISC: Using Define.xml Metadata to Ensure End-to-End Submission Integrity.’ Gilead offices, Foster City, CA, 2015. (http://www.pharmasug.org/download/sde/sf2015/PharmaSUG_SF2015SDE_07_Brega_Collins.pdf)
- Williams, Tim. ‘Diagram of the SW Initiatives?’ CS Working Group: End to End Connectivity, 7 April 2015. <https://phuse.teamworkpm.net/notebooks/48268>.

The graphical presentation is made using the R-package **DiagrammeR** (<https://github.com/rich-iannone/DiagrammeR>). The R-package uses **mermaid** a javascript blibrary (<https://github.com/knsv/mermaid>), described as in the credits as > Many thanks to the d3 (<http://d3js.org/>) and dagre-d3 (<https://github.com/cpettt/dagre-d3>) projects for providing the graphical layout and drawing libraries! > Thanks also to the js-sequence-diagram (<http://bramp.github.io/js-sequence-diagrams>) project for usage of the grammar for the sequence diagrams.

It may be nesccesary to install the latest version of R-package **DiagrammeR** from GitHub

```
devtools::install_github('rich-iannone/DiagrammeR')
```

This is under development

mja@statgroup.dk 16-jan-2015.

Setup

```
library(rrdfancillary)
devtools::load_all(pkg="../../..")
```

```
## Loading rrdfqbcindex
```

```
library("DiagrammeR")
library(rrdf)

pretty.print.rq<- function(rqstring) {
  ## print sparql query with line numbers - use full for finding errors
  ## when using knitr in Rmd files use highlight instead
  rqlines<- unlist(strsplit(rqstring,"\n"))
  cat(paste( format(seq(rqlines),format="f",digits=floor(log10(length(rqlines)+1))), rqlines, sep=": ", collapse=""), "\n")
}
```

Get the process description

The process description is converted into RDF. Please comment - this is my first attempt to model a process in RDF. It is done from scratch - I am aware of there are more sophisticated ways to do it, and would appreciate learning more on that approach.

```
modelTTLFile<- system.file("extdata/sample-rdf", "clinical-data-process-model.ttl", package="rrdfqbcindex")
model<- load.rdf(modelTTLFile,"TURTLE")
# identify the prefixes in the turtle file
# The prefixes are part of the RRDF model, but I do not know how to extract it from the model.
ttlfile<- readLines(modelTTLFile)
prefix.text<- gsub("@prefix ([^:]+:)+ ([^>]+>) *\\s *$", "PREFIX \\1 \\2", ttlfile[grepl("@prefix .+$", ttlfile)])
```

Reporting process from “Using Define.xml Metadata to Ensure End-to-End Submission Integrity”, slide 15

The process steps with labels and shapes are stored in a data frame.

```
RP.names.rq<- paste0( paste0(prefix.text, collapse="\n"), "
select ?rstep ?rsteplabel ?rstepshape
where {
  ?rstep a repr:PR .
  optional { ?rstep rdfs:label ?rsteplabel }
  optional { ?rstep repr:DisplayShape ?rstepshape }
  .
}
order by ?rstep
", collapse="\n" )

pretty.print.rq(RP.names.rq)
```

```
## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: select ?rstep ?rsteplabel ?rstepshape
## 6: where {
## 7:     ?rstep a repr:PR .
## 8:     optional { ?rstep rdfs:label ?rsteplabel }
## 9:     optional { ?rstep repr:DisplayShape ?rstepshape }
## 10: .
## 11: }
## 12: order by ?rstep
```

```
RP.names<- data.frame(sparql.rdf(model, RP.names.rq),stringsAsFactors=FALSE)
knitr::kable(RP.names)
```

rstep	rsteplabel	rstepshape
repr:r01	SDTM Datasets	Magnetic Disk
repr:r02	Create Analysis Datasets	Alternate Process
repr:r03	ADaM Datasets	Magnetic Disk
repr:r04	Analysis Programs	Alternate Process
repr:r05	Tables, Listings, Graphs	Multidocuments
repr:r06	Dataset specs	Multidocuments
repr:r07	Codelist specs	Multidocuments
repr:r08	Report specs	Multidocuments
repr:r09	Prepare submission data & docs	Alternate Process
repr:r10	Selected code	Multidocuments
repr:r11	Submission.xpt	Magnetic Disk
repr:r12	Define.xml	Multidocuments
repr:r13	Data Guide	Multidocuments

For later use, get an overview of the shapes used for reporting.

```
RP.shapes.rq<- paste0( paste0(prefix.text, collapse="\n"), "
select distinct ?rstepshape
where {
    ?rstep a repr:PR ;
    repr:DisplayShape ?rstepshape
    .
}
", collapse="\n" )

pretty.print.rq(RP.shapes.rq)
```

```
## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: select distinct ?rstepshape
## 6: where {
## 7:     ?rstep a repr:PR ;
```

```
## 8:      repr:DisplayShape ?rstepshape
## 9:      .
## 10:     }
```

```
RP.shapes<- data.frame(sparql.rdf(model, RP.shapes.rq),stringsAsFactors=FALSE)
knitr::kable(RP.shapes)
```

rstepshape

Multidocuments
Magnetic Disk
Alternate Process

Now get the links between the steps.

```
RP.feedsto.rq<- paste0( paste0(prefix.text, collapse="\n"), "
select ?rstepfrom ?rstepto
where {
  ?rstepfrom repr:FeedsTo ?rstepto
  .
}
", collapse="\n" )
pretty.print.rq(RP.feedsto.rq)
```

```
## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: select ?rstepfrom ?rstepto
## 6: where {
## 7:   ?rstepfrom repr:FeedsTo ?rstepto
## 8:   .
## 9: }
```

```
RP.feedsto<- data.frame(sparql.rdf(model, RP.feedsto.rq),stringsAsFactors=FALSE)
knitr::kable(RP.feedsto)
```

rstepfrom	rstepto
repr:r09	repr:r12
repr:r07	repr:r04
repr:r02	repr:r03
repr:r04	repr:r05
repr:r09	repr:r11
repr:r08	repr:r04
repr:r09	repr:r10
repr:r08	repr:r09
repr:r06	repr:r09
repr:r07	repr:r09
repr:r01	repr:r02
repr:r03	repr:r04
repr:r06	repr:r02
repr:r09	repr:r13

Finally, generate the mermaid commands for showing the process.

```
mermaid.commands<- paste(
  "graph TB",
  paste( RP.names$rstep,"(", RP.names$rsteplabel, ")", sep="", collapse=" \n" ),
  paste( RP.feedsto$rstepfrom, "-->", RP.feedsto$rstepsto, sep="", collapse=" \n" ),
  "\n",
  sep=" \n", collapse=" \n")
cat(mermaid.commands,"\n")
```

```
## graph TB
## repr:r01(SDTM Datasets)
## repr:r02(Create Analysis Datasets)
## repr:r03(ADaM Datasets)
## repr:r04(Analysis Programs)
## repr:r05(Tables, Listings, Graphs)
## repr:r06(Dataset specs)
## repr:r07(Codelist specs)
## repr:r08(Report specs)
## repr:r09(Prepare submission data & docs)
## repr:r10(Selected code)
## repr:r11(Submission.xpt)
## repr:r12(Define.xml)
## repr:r13(Data Guide)
## repr:r09-->repr:r12
## repr:r07-->repr:r04
## repr:r02-->repr:r03
## repr:r04-->repr:r05
## repr:r09-->repr:r11
## repr:r08-->repr:r04
## repr:r09-->repr:r10
## repr:r08-->repr:r09
## repr:r06-->repr:r09
## repr:r07-->repr:r09
## repr:r01-->repr:r02
## repr:r03-->repr:r04
## repr:r06-->repr:r02
## repr:r09-->repr:r13
##
##
```

Then display the graph.

```
DiagrammeR( mermaid.commands )
```

Grid for “Diagram of the SW Initiatives”

The ordering of the rows and columns in the grid is described in the turtle file by a collection. The Jena extension `list:index` is used to get the position in the collection. See (<http://stackoverflow.com/questions/17523804/is-it-possible-to-get-the-position-of-an-element-in-an-rdf-collection-in-sparql/17530689#17530689>).

```

wg.names.rq<- paste0( paste0(prefix.text, collapse="\n"), "
PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
select ?wg ?wglabel ?wgdisplayorder
where {
  ?wg a cswg:WG ;
  rdfs:label ?wglabel .
  optional{
    cswg:CSWGCollection cswg:DisplayOrder ?lswg .
    ?lswg list:index (?wgdisplayorder ?wg)
  }
}
order by ?wgdisplayorder
", collapse="\n" )

pretty.print.rq(wg.names.rq)

```

```

## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
## 6: select ?wg ?wglabel ?wgdisplayorder
## 7: where {
## 8:   ?wg a cswg:WG ;
## 9:   rdfs:label ?wglabel .
## 10:   optional{
## 11:     cswg:CSWGCollection cswg:DisplayOrder ?lswg .
## 12:     ?lswg list:index (?wgdisplayorder ?wg)
## 13:   }
## 14: }
## 15: order by ?wgdisplayorder

```

```

wg.names<- data.frame(sparql.rdf(model, wg.names.rq),stringsAsFactors=FALSE)

lc.names.rq<- paste0( paste0(prefix.text, collapse="\n"), "
PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
select ?lc ?lclabel ?lcdisplayorder
where {
  ?lc a cswg:DaLi ;
  rdfs:label ?lclabel .
  optional{
    cswg:LifeCycleCollection cswg:DisplayOrder ?lslc .
    ?lslc list:index (?lcdisplayorder ?lc)
  }
}
order by ?lcdisplayorder
", collapse="\n" )

pretty.print.rq(lc.names.rq)

```

```

## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

```

```

## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
## 6: select ?lc ?lclabel ?lcdisplayorder
## 7: where {
## 8:     ?lc a cswg:DaLi ;
## 9:     rdfs:label ?lclabel .
## 10:    optional{
## 11:        cswg:LifeCycleCollection cswg:DisplayOrder ?lslc .
## 12:        ?lslc list:index (?lcdisplayorder ?lc)
## 13:    }
## 14: }
## 15: order by ?lcdisplayorder

```

```

lc.names<- data.frame(sparql.rdf(model, lc.names.rq),stringsAsFactors=FALSE)
str(lc.names$lcdisplayorder)

```

```

## chr [1:9] "0" "1" "2" "3" "4" "5" "6" "7" "8"

```

```

grid.rq<- paste0( paste0(prefix.text, collapse="\n"), "
PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
select ?wg ?wglabel ?wgdisplayorder ?lc ?lclabel ?lcdisplayorder ?cell
where {
{ select ?wg ?wglabel ?wgdisplayorder where {    ?wg a cswg:WG ;
  rdfs:label ?wglabel .
  optional{
  cswg:CSWGCcollection cswg:DisplayOrder ?lswg .
  ?lswg list:index (?wgdisplayorder ?wg)
  }
  } }
{ select ?lc ?lclabel ?lcdisplayorder where {
  ?lc a cswg:DaLi ;
  rdfs:label ?lclabel .
  optional{
  cswg:LifeCycleCollection cswg:DisplayOrder ?lslc .
  ?lslc list:index (?lcdisplayorder ?lc)
  }
  }
}
bind( if( exists { ?wg cswg:RelatesTo ?lc }, '*', ' ' ) AS ?cell )
}
order by ?wgdisplayorder ?lcdisplayorder
", collapse="\n" )

pretty.print.rq(grid.rq)

```

```

## 1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
## 2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
## 3: PREFIX cswg: <http://www.example.org/phuse/wg#>
## 4: PREFIX repr: <http://www.example.org/phuse/reportingprocess#>
## 5: PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
## 6: select ?wg ?wglabel ?wgdisplayorder ?lc ?lclabel ?lcdisplayorder ?cell
## 7: where {

```

```

## 8: { select ?wg ?wglabel ?wgdisplayorder where {      ?wg a cswg:WG ;
## 9:      rdfs:label ?wglabel .
## 10:      optional{
## 11:      cswg:CSWGCollection cswg:DisplayOrder ?lswg .
## 12:      ?lswg list:index (?wgdisplayorder ?wg)
## 13:      }
## 14:      } }
## 15: { select ?lc ?lclabel ?lcdisplayorder where {
## 16:      ?lc a cswg:DaLi ;
## 17:      rdfs:label ?lclabel .
## 18:      optional{
## 19:      cswg:LifeCycleCollection cswg:DisplayOrder ?lslc .
## 20:      ?lslc list:index (?lcdisplayorder ?lc)
## 21:      }
## 22:      }
## 23: }
## 24: bind( if( exists { ?wg cswg:RelatesTo ?lc }, '*', ' ' ) AS ?cell )
## 25:      }
## 26: order by ?wgdisplayorder ?lcdisplayorder

```

Finally, create a grid and fill in the values into the grid

```

grid<- data.frame(sparql.rdf(model, grid.rq),stringsAsFactors=FALSE)
wglcgrid<- data.frame(matrix(c("?"), nrow= nrow(wg.names), ncol=nrow(lc.names)),
                           stringsAsFactors=FALSE, row.names=wg.names$wglabel)
colnames(wglcgrid)<- lc.names$lclabel
for (i in seq(nrow(grid)) ) {
  wglcgrid[as.numeric(grid[i,"wgdisplayorder"])+1,as.numeric(grid[i,"lcdisplayorder"])+1] <- grid[i, "cell"]
}
knitr::kable(wglcgrid)

```

	Clinical Development Plan	Study Protocol	Study Design	Collect
Study Data Standardization Plan	*			
Clinical Program Design in RDF	*	*		
Study Design and Protocol in RDF	*	*	*	
Best Practices for Data Standards Implementation				*
SEND Implementation User Group				*
CDISC Foundational Standards in RDF				*
Application of SEND for Data Analysis				
Analysis Results & Metadata				
Standard Scripts for Analysis Programming				
SDRG & ADRG				
Non-Clinical SDRG				
Data Standards and Traceability	*	*	*	*

How to make the HTML file

```

rmarkdown::render("reporting-process.Rmd")

```