On the use of RDF in records management and archiving

Research Report for Riksarkivet
# Contents

1  Introduction .......................................................................................................................... 3

2  General Discussion on Benefits of RDF ............................................................................. 4

   2.1  Expressive, Extensive and Merge-able Data ................................................................. 4

   2.2  URIs ............................................................................................................................... 4

       2.2.1  URIs give control over identifiers ......................................................................... 5

       2.2.2  URIs for addressing and identity ....................................................................... 6

   2.3  Error! Bookmark not defined. 

   2.4  Schemas in the Data ..................................................................................................... 6

   2.5  Standards Stack ............................................................................................................ 6

   2.6  Interchange .................................................................................................................. 7

3  RDF Modelling in Records Management and Archive Standards ..................................... 8

   3.1  General remarks on modelling approach .................................................................... 8

   3.2  Modelling Tools .......................................................................................................... 9

       3.2.1  Semantic Operations Framework ........................................................................ 10

   3.3  Noark5 in RDF ............................................................................................................ 11

       3.3.1  Fond Model ........................................................................................................ 11

       3.3.2  Series Model ....................................................................................................... 12

       3.3.3  Classification Model ........................................................................................ 13

       3.3.4  File Model .......................................................................................................... 14

       3.3.5  Record Model ..................................................................................................... 14

       3.3.6  DocumentDescription and DocumentObject Models ..................................... 14

       3.3.7  Preservation and Disposal Model ....................................................................... 15

       3.3.8  Transmission Models ........................................................................................ 15

       3.3.9  RDF Snippet ...................................................................................................... 16

       3.3.10  Semantic Operations Example ....................................................................... 16

       3.3.11  Summary .......................................................................................................... 17

   3.4  MoReq2010 in RDF ..................................................................................................... 17

       3.4.1  User and Group Service ..................................................................................... 18

       3.4.2  Model Role Service ............................................................................................ 19

       3.4.3  Classification Service ......................................................................................... 19

       3.4.4  Record Service ................................................................................................... 21

       3.4.5  Model Metadata Service ...................................................................................... 22

   3.5  Disposal Services ........................................................................................................ 23

   3.6  Mapping Noark5 to Moreq2010 .................................................................................. 23

   3.7  Noark5 as a Moreq2010 Data Module ....................................................................... 23

   3.8  Domain Model Definitions ......................................................................................... 24

   3.9  Vision for Noark Standard .......................................................................................... 26

       3.9.1  What is being standardised and Why? ................................................................. 26

4  Using RDF Technology to Facilitate Automated Archive Processes ................................ 27

   4.1  Syndication Approach to Archive Submission ............................................................ 27

5  Summary of Work and Future Directions ....................................................................... 29

References ................................................................................................................................. 30
1 Introduction

The document is a report into the role and utility of the RDF family of standards in record management and archive standards. We begin by presenting the core capabilities and properties of RDF itself. Positioning it as a viable candidate for providing the expressiveness and openness needed as the basis of any standard or for adoption by a community.

The work itself encompasses defining the Noark5 and Moreq2010 core data models as RDF models and schemas. In the case of Moreq2010 we also provide an insight into how RDF can be used to replace core bespoke services of the standard.

The report documents how we can define the Noark5 data model in terms of the Moreq2010 data model. This opens up the possibility to define a standards stack where Noark and subsequent domain models are simply data modules built on top of a leaner and more formal Moreq2010 core.

The last section of the report tackles a completely orthogonal issue, the issue of continuous submission of records to an archive. We present a pull based model that uses RDF and the SDShare protocol to deliver a robust and scalable approach to this problem.
2 General Discussion on Benefits of RDF

The Resource Description Framework[1] (RDF), from the W3C, is a powerful technology for expressing data structures, sharing data and managing the identity of the things computer systems wish to talk about. This section describes the core properties of RDF and the value that they bring to implementing and describing data centric systems.

This section assumes a basic understanding of the RDF data model and supporting standards such as RDF Schema 1.1. (RDFS) [2] and SPARQL[3]. The RDF primer[4] provides a comprehensive introduction if needed.

2.1 Expressive, Extensive and Merge-able Data

RDF has a very simple, yet powerful data model. The model is described as either a triple or quad data model. It consists of the identity of a resource, the identity of a property type and a property value. The property value can either be a resource or a literal property. This simplicity delivers power in that any other kind of data structure can be formulated using just this simple building block.

Inherently the data model lends itself to graph like data structures, but trees and lists can also be represented. This flexibility allows RDF data to represent data of all shapes and sizes and promotes connectivity and connections as a first class citizen. XML on the other hand is inherently hierarchical and support for links and graph like structures is limited.

So, RDF is more expressive than XML but perhaps its greatest strength is that given two chunks of RDF that describe the same resource they can be merged with no need for special code or human interaction. This cannot be said for XML. In XML, bespoke merging rules must be written for each schema whose data is to be merged.

This simple example shows two blocks of RDF that are to be merged, with the resulting RDF shown last. Notice how repeating values are allowed and that duplicates are supressed.

```text
@prefix sesam: <http://sesam.io/business/people/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# RDF Document one
sesam:graham_moore a foaf:Person .
sesam:graham_moore foaf:name "Graham Moore" .

# RDF Document two
sesam:graham_moore a foaf:Person .
sesam:graham_moore foaf:name "Graham David Moore" .

# Merged Document
sesam:graham_moore a foaf:Person .
sesam:graham_moore foaf:name "Graham Moore" .
sesam:graham_moore foaf:name "Graham David Moore" .
```

2.2 URIs

At the heart of RDF is the triple/quad data model and supporting that model is the extensive use of URIs. URIs are used to identify resources. A resource is anything about which we wish to say
On the use of RDF in records management and archiving

2.2.1 URIs give control over identifiers

URIs inherently convey a sense of ownership and authority. They are connected to a domain that is owned by a legal entity. RDF is about being able to share, connect and reuse identifiers in order to reach agreement. Semantics is no more or less than the agreement between two or more parties as to what something means. Many organisations look to authorities to define this meaning. Authoring these concepts comes down to defining vocabularies and assign meaning. Authorities are able to define domain concepts as RDF resources using URIs. Not only are these concepts connected to the domain but the sections of the URI can be used to further organise concepts.

The following example shows the simple use of URIs to represent resources published by an authority:

http://sesam.io/business/people/graham_moore

In this example, the domain is owned by the SESAM organisation, and they have portioned off a namespace section called business that then defines a grouping section called people and then the complete URI is for the person called Graham Moore. Although the URI implies what the thing identified is and even what it is called, this is not critical, and potentially misleading.

The reason for creating globally unique, organisation minted identifiers is that it allows data publishers to indicate that they are ‘saying something’ about the same thing as the authority. Consumers of information can then gather information from many sources and in cases where the identifiers are the same the consumer can be confident they are talking about the same thing and the data can be merged.

When authorities publish identifiers they often do so in both a human and machine readable fashion. The human description often takes the form or a short concise prose definition like the following example from the FOAF (friend of a friend) vocabulary.

```
Class: foaf:Person

Person - A person:
Status: stable
Properties include:
    plan surname geekcode pastProject lastName family name publications
currentProject familyName firstName workInfoHomepage myersBriggs
schoolHomepage img workplaceHomepage knows
Used with:
Subclass Of: Agent Spatial Thing
Disjoint With:

The Person class represents people. Something is a Person if it is a person. We don’t nitpick about whether they’re alive, dead, real, or imaginary. The Person class is a sub-class of the Agent class, since all people are considered ‘agents’ in FOAF.
```
If an organisation or individual chooses to use the foaf:Person identifier then there is a general expectation that they agree with the definition as described.

2.2.2 URIs for addressing and identity

The fact that URIs can be resolved to web resources gives a second dimension to the use of URIs as identifiers. They are also the web address for the resource description identified by the URI. This description is in fact the RDF description of the resource.

An RDF description is a set of statements that say something about the resource in question. It is here that more clarity can be given about the nature of the resource being identified. Continuing our example from above using a concise RDF notation:

@prefix sesam: <http://sesam.io/business/people/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

sesam:graham_moore a foaf:Person .
sesam:graham_moore foaf:name "Graham Moore".

The above example demonstrates the use of a newly minted identifier by an authority (in this case Sesam organisation), and the reuse of identifiers by other authorities to convey meaning (the use of foaf:Person, and foaf:name).

Having resources both addressable and identifiable using URIs allows consumers of RDF to follow the resource URI to get a clear understanding the resource it identifies. The meaning of the concept is then defined purely in terms of the statements being made by the publisher. Once again, the actual structure of the URI is irrelevant.

2.3 Schemas in the Data

In XML, an XML Schema is often a document outside of the instance document. The schema is used to dictate how the data is stored or indexed and changes to the schema are often costly and affect existing data storage, and data indexes.

In RDF, schemas do not dictate anything. In fact, at some level there are no schemas. The core RDF model is truly schema-less. Schemas, or constraints in RDF are just more data. But they are ‘special’ data. They are special data in that certain evaluation semantics are attached to them.

This special interpretation is made possible because the RDF model used to describe schemas and constraints uses well known URIs. In standards such as RDFS 1.1, SPIN[5] and RDFCL[6], these well-known URIs are connected to semantic operations. A example of a semantic operation would be checking if a given constraint, contained in a schema, is valid with respect to a given RDF graph.

This loose coupling of the description of structure from either storage or indexing creates great power and flexibility. An example of this is allowing multiple views or schemas to apply to a single RDF database, the choice of when to validate constraints, and even the ways in which the constraints to be validated are chosen.

The basic idea of well-defined URIs for resources, connected to semantic descriptions or processing behaviours is a flexible and powerful way to describe standards. The Semantic Operations Framework, developed as part of this research, is a generalisation of this pattern and is used in later sections to illustrate how the semantics of MoReq2010 and Noark5 can be described.

2.4 Standards Stack

RDF and the related family of standards is maintained and developed by the W3C. This ensures that tooling, adoption, tutorials and vendor neutrality is maintained. Unlike other NoSQL or emerging technologies RDF is not bespoke or owned by a single vendor.
It is for these reasons that RDF has gained traction amongst the governments of UK and USA as the preferred means of managing and publishing public data, and for organisations looking for modern and flexible approaches to metadata, data interchange and master data management.

As well as a user adoption RDF is the only technology to get significant buy in from database vendors. Companies such as Oracle, IBM, and MarkLogic all provide RDF support along with a host of dedicated RDF database implementations from new companies.

2.5 Interchange

RDF is a model that has multiple standardised syntaxes. RDF can be serialised in plain text files as n-triples or turtle, as XML, and in JSON as JSON-LD.

This provides great flexibility in allowing implementers to adopt the serialisation format that best fits into their eco-system. But at all times the model remains consistent and it is the model that drives the approach.
3 RDF Modelling in Records Management and Archive Standards

This area of research looks at how existing records management and archiving standards such as Noark5 and MoReq2010 can have their core data models and constraints expressed using RDF and related constraint technologies. Specifically, we take each of the functional areas of MoReq2010 and Noark5 and describe the core entity and property types using RDF.

One of the reasons for moving these models into RDF is to facilitate better interoperability between standards. Noark5 has a loose association with MoReq2 and there would be great benefit if subsequent versions of the Noark standard could be defined in terms of an existing standard simply through the use of mapped data models or as ‘data modules’.

Another use case is to understand how data can be mapped from one standard to another at the data model level, not the syntax level, to improve interoperability between different standards.

A final consideration is that with RDF gaining traction many organisations owning or implementing archive and records management solutions are starting to develop incomplete and bespoke RDF models of the data structures in Noark5 and MoReq2010. While different ontologies can be mapped to each other it would be a useful proactive step to create an authoritative representation in RDF of the models. We are not proposing that what is presented here should be that final version, but instead, that it could inform a larger process that deals with this issue.

3.1 General remarks on modelling approach

Noark5 and MoReq2010 are primarily soft standards, although MoReq2010 is the harder of the two. By soft or hard standards we refer to the precision of the semantics and the ease with which a given implementation can be evaluated as conformant. It appears from the standards to be two main conformance points of interest.

The first conformance point is import and export serialisation conformance. This is useful for robust exchange and interchange of data. Conformance at this level is about saying, “we don’t mind how you got to this point but at this point the data needs to look like this”. This level of conformance is useful in ensuring that data interchange can happen seamlessly and unambiguously. In XML this is achieved using XML Schema. An XML Schema can define the allowed shape of XML documents that are to be submitted. Documents can be validated by XML Schema validators.

In RDF there are several different modelling and validation approaches. The power of RDF is that these approaches can be used in combination or more specialised behaviours can be defined. OWL, SPIN and RDFCL are all candidates for use in defining the allowed shape of the data through a model description.

The problem with both RDFS and OWL is that they do not do validation, they do inference. This can be explained with a simple example. Consider a constraint that says that instances of type Person must have a birthdate property. In OWL this is in fact interpreted as an inference statement. Meaning that, any things that has a birthdate property is in fact a person. The unwanted inferences make OWL and RDFS inappropriate for validation semantics. We have chosen to use the RDF Constraint Language (RDFCL)[6] for this work as it provides the kind of type centric constraints needed for the models we will be describing.

RDFCL can be most succintly expressed using the RMIL[7] syntax. This syntax is optimised for data model and controlled vocabulary descriptions. It is easily readable and writable by humans and allows constraints to be concisely defined.
All expressions in RMIL translate into RDF statements. The RDF generated from the RMIL is primarily that described in RDFCL, but for completeness and greater interoperability RDFS statements can also be derived.

The second area of conformance we are interested in, and hence supporting through RDF technologies, are the allowed operations of a records management system. In Moreq2010 along with the data model descriptions are function descriptions. Each function describes the type of data entity the function is performed upon, pre-conditions, post conditions and what should be logged for auditing purposes. There is little formalisation of these operations. Noark5 also describes operations but these tend to be in prose with no formalisation.

In this work we use the Semantic Operations Framework (SOF)[8]. This framework has been developed to allow the expression of operational semantics in a formal, and unambiguous fashion. The framework makes each operation a first class entity. Each operation describes the nature of the data that can trigger the operation, the type of data it operates on, the pre-conditions that must exist and the actual update to perform. All constraint checks and updates are performed using SPARQL 1.1 and operate on the data models defined in RDF.

Using RDFCL and the Semantic Operations Framework together creates a unified way to describe data models and semantics in a way that is implementation agnostic yet at the same time executable in any standard RDF environment. While this may sound like a contradiction it is correct. RDF has an abstract meta-model and SPARQL is defined in terms of this model. The fact that SPARQL is also executable is a convenient by-product. What we mean by this is that although RDF can be seen as technology, where it exists in text files, as indexes in databases, as results in programming languages it also has an abstract, mathematical formalism. Any data models and semantics defined in terms of RDF can be seen and understood either at the implementation level or at the abstract level. When seen at the abstract level it forms a description of behaviour that could be translated and then implemented in any number of concrete technical implementations.

In must be noted that due to a lack a time the data models and constraints are nearly complete, but the set of operations is not. The operations approach is illustrative of how the semantics of the standards can be described in a formal fashion.

The driver behind formalising models and semantics is that this demonstrates an approach by which records and archiving standards could be made harder, more robust, less ambiguous, and lower the effort and cost in deciding which implementations are conformant. This would not remove the need for English / Norwegian prose but the normative meaning would be deferred to formal models that define structure and semantics.

3.2 Modelling Tools

To describe the data models we use the RDF Constraint Language along with the RMIL syntax. This section provides a quick introduction the main features used in describing the models

RMIL is optimised for describing types, type constraints and controlled vocabularies. The following example declares the type Person as a subtype of foaf:Person, and says that all instances of type person MUST have exactly one birth date, one name, and optionally many emails.
3.2.1 Semantic Operations Framework

The Semantic Operations Framework[8] has been developed as part of this research. It is incomplete but is a useful tool in demonstrating how formal semantics can be defined. The framework describes classes of operations on models as first class entities. Each operation class describes pre-conditions as SPARQL queries and update operations as SPARQL 1.1 Update expressions.

The model for semantic operations is defined below.

The following example describes the semantic operation structures for a *set_name* class of operation.
# The core class definition for all semantic operations

class SemanticOperation
    xsd:string validation_query 1 1
    xsd:string update_query 1 1
end

# The class for a given class of operation

class set_name
    rdf:resource person 1 1
    xsd:string name 1 1
end

# Also indicate that the class is an instance of SemanticOperation and class level properties
# for validation and update.

instance set_name : SemanticOperation
    validation_query "select ?x where ..."
    update_query "delete ..., update ..."
end

# an example of a operation

instance a_set_name : set_name
    name "bob"
    person http://example.org/gra
end

The framework for executing these operational descriptions is generic and can be used in any domain. We use this framework to unambiguously describe illustrative operations from both the MoReq2010 and Noark5 standard.

## 3.3 Noark5 in RDF

In approaching Noark5 we have attempted to capture the core data models as RDF. It should be noted that Noark5 describes a conceptual data model so part of the process has been to tighten that into a concrete data model. In doing this we have exercised judgement on how the conceptual model is mapped into RDF. One example of this is that aggregating entities often have some kind of child collection property described. In the case of containment or aggregation this property is an inverse property and would never be explicit in a model.

The complete models in RMIL syntax and the n-triple RDF representations for the Noark5 data model are available online at https://github.com/SesamResearch/Records-Management-and-Archive-Systems-Research.

As well as the models we also demonstrate the use of the Semantic Operations Framework to describe the operations associated with the different model areas.

It should be noted that due to time constraints not all cardinalities will be correct and in some places outstanding domain specific questions remain unanswered or need further discussion.

### 3.3.1 Fond Model

The things of note in the following model are the promotion of string values in the original standard to controlled vocabularies in RDF. Also the property types defined can be mapped using OWL sameAs to
identifiers that correlate directly to the identifiers in the standard. Note how all of these things are done in data.

```xml
# Default URI Prefix for Noark5
prefix http://arkivverket.no/noark5/

# Archive Structure
# Utility Type
class FondCreator
  xsd:string title 1 1
  xsd:string description 0 1
end

# Definition for Class Fond
class Fond
  xsd:string title 1 1
  xsd:string description 0 1
  FondStatus status 1 1
  xsd:string documentMedium 0 1
  xsd:string storageLocation 0 *
  xsd:dateTime createDate 1 1
  FondCreator createdBy 0 1
  xsd:dateTime finalisedDate 1 1
  OrganisationalUnit finalisedBy 0 1
  Fond parent 0 1
end

# Fond Status Class
class FondStatus
end

# Controlled Vocabulary for Fond Statuses
instance Open FondStatus
end

instance Closed FondStatus
end

# Property type mapping to Noark5
# This is only done as example.
instance title
  owl:sameAs M020
end

instance description
  owl:sameAs M021
end

3.3.2 Series Model
```
# Definition for Class Series

class Series

xsd:string title 1 1
xsd:string description 0 1
xsd:string documentMedium 0 1
xsd:string storageLocation 0 *
xsd:dateTime createdDate 1 1
FondCreator createdBy 0 1
xsd:dateTime finalisedDate 1 1
OrganisationalUnit finalisedBy 0 1
xsd:date periodStartDate 1 1
xsd:date periodEndDate 1 1
Fond parent 1 1
Classification primaryClassification 1 1

end

3.3.3 Classification Model

While defining the classification model there are a few decisions to make regarding how the hierarchy is supported. In Noark5 instances of ClassificationSystem form the root of a hierarchy and then instances of Class can either have another Class as a parent or they can have the ClassificationSystem. In general the idea of properties linking to values whose types are disjoint is bad practice. We would have preferred the model here to be one where Class instances form a hierarchy and the ClassificationSystem references one or more Class instances as being the root, or roots. For now we have honoured the original structure by introducing a common supertype called ClassificationNode. Both ClassificationSystem and Class are specialisations of this.

class ClassificationNode

ever

class ClassificationSystem ClassificationNode

xsd:string classificationType 0 1
xsd:string title 1 1
xsd:string description 0 1
xsd:dateTime createdDate 1 1
FondCreator createdBy 0 1
xsd:dateTime finalisedDate 1 1
OrganisationalUnit finalisedBy 0 1

End

class Keyword

xsd:string title 1 1

end

class Class ClassificationNode

xsd:string title 1 1
xsd:string description 0 1
xsd:dateTime createdDate 1 1
FondCreator createdBy 0 1
Keyword keyword 0 *
xsd:dateTime finalisedDate 1 1
OrganisationalUnit finalisedBy 0 1
ClassificationNode parent 0 1

end
3.3.4 File Model

A File in Noark is analogous to a description of the execution of a single process. For example it is the data structure that represents a job application or a single case. It is categorically, not a File in terms of computer files, or electronic documents. A File consists of one or more records, or sub Files. In general terms it is a certain class of container.

```xml
# Definition for Class File
class File
	xsd:string fileID 0 1
	xsd:string filetype 0 1
	xsd:string title 0 1
	xsd:string officialTitle 0 1
	xsd:string description 0 1
	xsd:string documentMedium 0 1
	xsd:string storageLocation 0 *
	xsd:dateTime createdDate 1 1

FondCreator createdBy 0 1

xsd:dateTime finalisedDate 1 1

OrganisationalUnit finalisedBy 0 1

File parentFile 0 1

end
```

3.3.5 Record Model

The record describes one record of evidence related to the ‘File’.

```xml
# Definition describes one record of evidence
# A single record of evidence
class Record
	xsd:string recordType 0 1
	xsd:dateTime createdDate 1 1

FondCreator createdBy 0 1

xsd:dateTime archivedDate 1 1

OrganisationUnit archivedBy 0 1

File parentFile 0 1

Series parentSeries 0 1

end
```

3.3.6 DocumentDescription and DocumentObject Models

The following models describe the DocumentDescription and DocumentObject models from Noark5.
3.3.7 Preservation and Disposal Model

Preservation and disposal is a core concept in records management. Records, containers, in fact all data constructs managed by the records management system are governed by some disposal policy. Disposal policies and often driven by legislation, or required business practices, or contractual obligations. They are put in place to ensure that data about people, businesses, transactions, etc are only kept as long as legally required.

This simple model captures when a record should be disposed of and what should happen in these cases.

3.3.8 Transmission Models

The data structures described so far have been concerned with the records management aspects of Noark5, but the standard also describe the submission process. We have not defined the metadata structures for transmission metadata but envisage that the same approach could be adopted to describe these structures.
The RDF descriptions of the metadata would be included in the submission package in place of the XML. Other than for completeness and consistency the use of RDF is this case offers little added value over XML.

Related to this issue it should be noted that in chapter 7 we consider an alternative approach to the classic submission process.

3.3.9 RDF Snippet

The use of RMIL makes reading and writing schemas possible for humans, but it hides the final RDF representation. This representation is verbose and can be seen in full online at https://github.com/SesamResearch/Records-Management-and-Archive-Systems-Research/blob/master/rdf/noark5.nt.

To illustrate the RDF produced from the RMIL language a small snippet is included here. The snippet defines the type for the Record type. It shows the use of RDFS identifiers and also the RDFCL constraint model. Note this sample describes just one constraint and illustrates the need for languages like RMIL.

```


```

3.3.10 Semantic Operations Example

As well as the models defined above if we are considering creating more formal normative sections for the Noark standard then the operations on the data models also need to be formalised. As described previously, we adopt the Semantic Operations Framework as a means to express classes of operation on data models.

To demonstrate how this would work for Noark5 we have picked the functional requirement 5.2.19. This requirement applies to Series, and states that:

“If Series is registered as finalised (finalisedDate is set), it must not be possible to add more associated Files or Records.”
We define the operation `add_series_record` using the semantic operations framework.

```ruby
class AddSeriesRecord  
  Series parentSeries 1 1  
  Record record-to-add 1 1  
end

instance AddSeriesRecord  
  SemanticOperation  
    validation-expression "select ?y where {  
      ?x noark5:finalisedDate ?y  
      FILTER(?x = [[parentSeries]])  
    }"  
    update-expression "insert {  
    }"  
end
```

The constraints on `AddSeriesRecord` are used to constraint instances of this operation. An operation instance can be compared to a message in a message passing system or a web service call in a SOAP. The constraints indicate what data needs to be in the message for it to be a valid message.

The properties on `AddSeriesRecord` are used by the execution algorithm of the SOF. The framework first evaluates the `validation-expression` using the parameters in the message instance. E.g. the value of the `parentSeries` property of the message is substituted into the `validation-expression` SPARQL. This SPARQL expression is then evaluated against and if any results are returned then the constraint has been violated and the update operation does not occur. If the query returns an empty resultset then the `update-expression` can be executed.

We propose that all the functional requirements and data operations, such as `create_record`, `create_document_object`, etc could be defined using the framework. The outcome would be a more formal and testable standard.

### 3.3.11 Summary

The previous sections have described the core data models of Noark5 in RDF. The resulting schemas provide a near-complete set of URI identifiers for types and property types of the Noark5 data model and a set of validation constraints that can be used to check the integrity of a Noark5 RDF model instance.

The approach has identified several places where simple string values were used and converted those to be RDF controlled vocabularies, finally, using the Semantic Operations Framework has been used to demonstrate how the semantics of the Noark5 standard could be embodied in a more formal manner.

### 3.4 MoReq2010 in RDF

With Noark5 we have been concerned primarily with having a complete data model as RDF and illustrating how the semantics could be conveyed using the semantic operations framework. The work with Moreq2010 is a little different. With Moreq2010 we have taken a more radical approach and tried to re-imagine the standard in terms of RDF.


We consider each of the data centric services and take one of two general approaches. We either describe it in terms of an RDF model, or describe why in an RDF centric version of Moreq2010, the required functionality can be achieved by other means or by taking another perspective.
For each of the services we describe in RDF we define the data models that support the service using RDFCL. The outcome from this is a well-defined set of URIs for types and property types in RDF. It also delivers a set of constraints that can be used to evaluate if a given Moreq2010 data structure in RDF is compliant.

All resources have unique URIs that identify the concept they represent. These same concepts are also given UUIDs in the standard. To robustly connect the identifiers described here with the standard owl:sameAs can be used.

We use the RMIL notation to allow us to convey the models and constraints in a concise and readable fashion. The RDF and RMIL for MoReq2010 can be found online at https://github.com/SesamResearch/Records-Management-and-Archive-Systems-Research.

As well as the data model definitions we also describe examples of semantic operations using the Semantic Operations Framework. This framework allows for the expression of operational semantic on data models using the SPARQL 1.1 Update Language.

Notes:

1. In general inverse properties are not part of the data model per se. e.g. events are not properties or a property collection of an Entity but are in fact referenced from the Event instance. They can be computed but are not explicit. It would be possible to express these in terms of query properties that express the computation that yields the value.

3.4.1 User and Group Service

MoReq2010 requires that systems manage their own data regarding users and groups even if that data is sourced from other systems.

The following RMIL description covers all the main entity types with all their properties. It also expresses the constraints that apply to this data model.

```xml
# User and Group Service Data Model

# Entity Class Definition
class User
  xsd:dateTime created 0 1
  xsd:dateTime originated 0 1
  xsd:dateTime firstUsed 0 1
  Group group 0 *
  xsd:string title 0 1
  xsd:string description 0 1
  xsd:dateTime destroyed 0 1
end

# Group Class Definition
class Group
  xsd:dateTime created 0 1
  xsd:dateTime originated 0 1
  xsd:dateTime firstUsed 0 1
  xsd:string title 0 1
  xsd:string description 0 1
  xsd:dateTime destroyed 0 1
end
```
3.4.2 Model Role Service

The Model Role Service is responsible for managing which operations different users or groups of users can perform.

The following RMIL description covers all the main entity types with all their properties. It also expresses the constraints that apply to this data model.

```xml
## Model Role Service
class Function
end

class Role
  xsd:dateTime created 1 1
  xsd:dateTime originated 1 1
  xsd:dateTime firstUsed 0 1
  xsd:boolean isAdministrativeRole 0 1
  xsd:string title 1 1
  xsd:string description 0 1
  xsd:string scopeNote 0 1
  Function function 1 1
  xsd:dateTime destroyed 0 1
end
```

3.4.3 Classification Service

The core function of this service is to provide ‘operational context’ to the items in the records management system. To indicate the class of business process that has led to a given aggregation or record being created. The other major use case is to provide a way to connect extensible metadata to record management system items. For example a record can be given the classification as being of the class ‘application letter’. The extensible metadata system attaches property templates to this class. Record items that have this classification are then expected to adhere to the rules of the metadata constraints.

Our approach to addressing the classification and metadata extensions using RDF has been to replace the use of the Class property with data driven specialisations of the core structures. This means that rather than an Aggregation or Record pointing to a class that instead specialisations of Aggregation are created when needed.

The current metadata extension mechanisms in Moreq2010 ignore a critical aspect of specialisation. This specialisation is about being able to constraints the data structure related to the core data model.

We believe a critical use case would be to say that aggregations that are of class ‘Job Application’ can only contain records of the following types, ‘Application Letter’, ‘Interview Letter’, etc. MoReq2010 appears to provide mechanisms to define the contextual metadata on an Aggregation of class Job application but not to restrict further the kinds of records that can make up that aggregation.

Moreq2010 realises that metadata is not its core competence or responsibility, and thus states that compliant implementations can offer equal capabilities implemented differently.

We believe that in an RDF centric standard, or implementation, the use of RDF subtyping would address all the existing use cases, and address the newly identified and perhaps most critical case – that of enabling domain specific model refinements.

We use an example to demonstrate how this could work. We will consider the job application process as a typical records management use case. In a MoReq2010 fixed-typed world we would have an aggregation that contained several records. The aggregation would represent a single instance of the
job application process and the records the letters of correspondence between the applicant and the employer, as well as any contract.

In the RDF approach these new types would be defined in data. More importantly they would be defined in terms of the existing MoReq2010 data model. First we consider all the core types:

```ruby
# Job Application Use Case
class JobApplication Aggregation
end

class ApplicationLetter Record
end

class InterviewLetter Record
end

class OfferLetter Record
end

class Contract Record
end

class SignedContract Record
end
```

With the types defined we can describe the extra data that each type should have:

```ruby
class JobApplication Aggregation
  Person applicant 1 1
  xsd:string applicantName 1 1
  xsd:string applicantAddress 0 1
  xsd:string applicationEmail 0 1
end
```

With the types defined we can add extra constraints that relate to the aggregation structures themselves. In this case we want to constrain the types of allowed records. We do this in two parts, first by defining some class based constraints and then adding a global constraint.

```ruby
# Add structure constraints
class JobApplication
  ApplicationLetter applicationLetter 1 1
  InterviewLetter interviewLetter 0 *
  OfferLetter offerLetter 0 1
  Contract contract 0 1
  SignedContract signedContract 0 1
end

# additional constraint
instance jobApplicationStructureConstraint1 rdfcl:Constraint
  query "select ?x where { ?x is a child of a_jobapplication but is not of one of the allowed types"
end
```
3.4.4 Record Service

The record service has one of the richer and probably the most fundamental data model in the Moreq2010 standard. It provides the very core of what is represented in a records management system.

The following RMIL description covers all the main entity types with all their properties. It also expresses the constraints that apply to this data model.

```xml
# default prefix for moreq2010
prefix http://data.sesam.io/moreq2010/
prefix owl http://w3c.org/owl

# Aggregation Data Model
class Aggregation
   xsd:dateTime created 1 1
   xsd:dateTime originated 1 1
   xsd:dateTime first-used 0 1
   xsd:dateTime last-addition 0 1
   Class class 1 1
   xsd:string title 1 1
   xsd:string description 0 1
   xsd:string scope-notes 0 1
   xsd:dateTime closed 0 1
   xsd:dateTime destroyed 0 1
   xsd:int max-levels-of-aggregation 0 1
   Aggregation parent 0 1
   xsd:dateTime aggregated 0 1
end

# Record Class Definition
class Record
   xsd:dateTime created
   xsd:dateTime originated
   xsd:string title
   xsd:string description
   Record duplicate 0 1
   Aggregation parent 1 1
   xsd:dateTime aggregated
   Class class 0 1
   DisposalSchedule disposalSchedule 0 1
   xsd:dateTime retentionStart 0 1
   DisposalAction disposalAction 1 1
   xsd:dateTime disposalActionDue
   xsd:dateTime disposalConfirmationDue
   xsd:dateTime disposalOverdueAlert
   xsd:string lastReviewComment
   xsd:dateTime lastReviewed
   xsd:dateTime transferred
   xsd:dateTime destroyed
end
```

And supporting classes:
3.4.5 Model Metadata Service

The model metadata system can either be implemented as described in the standard or the same capabilities can be provided in an alternative fashion. This work is based on using RDF and the RDF family of standards. As such RDF has capabilities in OWL, RDFS, and RDFCL. All of the technologies have the ability to be used as a more ‘native’ approach to the Model Metadata service.

We propose using the RDFCL constraint language to provide the data model and features of Model Metadata Service.

Metadata is intended to capture the context in which decisions are taken, to provide a guide to future observers as to why a decision was reached, and as a way to understand the record in the context it was created. Having rich, complex metadata describing the context of a records management system entity can aid several key use cases.

The first use case it helps with is the location use case. When future users are looking to locate a record the more complete the metadata is, the closer it can position an item into a context that resembles the original context the easier and more accurately is can be found.

The second use case is the comprehension use case. In this scenario we are concerned with how well the metadata conveys to an observer the reasons and context at the time the record was created. This is...
particularly important when looking back over decisions and processes that are under review or being challenged.

Ideally, an infinite context would be preserved. In this case a record management system entity would have metadata that simply referenced to other business objects. Data would be stored in an immutable, append-only fashion such that the complete data set can be queried as it was at any point in time or as it is now.

This would allow for a record management entity to be located and seen not only with a few pieces of contextualised metadata but with the entire data set of the entire organisation captured at a point in time.

While there are data stores that support this kind of model there are of course many that don’t and requiring all stores that might make use of the data to support it is unrealistic. This leads to a need to capture some contextual data as metadata on the entity.

In general the use of RDFCL provides the capabilities of the Model Metadata Service. Rather than creating metadata template definitions that define what extra properties can be used on an Aggregation of a given class, RDFCL simply subclasses Aggregation and defines additional property constraints.

This approach unifies Classification, Contextual Metadata Extension and Core Structure extension to create a unified approach to data management.

3.5 Disposal Services

The Disposal Scheduling Service and Disposal Holding Services are outside the scope of this work. However, we consider that for both of these capabilities the same kind of approach would be taken as for the other data centric services. We would define the data models in RMIL and create an RDF data model and then create the semantics using the Semantic Operations Framework.

3.6 Mapping Noark5 to Moreq2010

There are two main reasons why we would like to see, or understand, Noark5 in terms of Moreq2010.

1. The main reason is the possibility that future versions of Noark are defined purely as refinements of the RDF version of Moreq2010. The mapping is therefor to make it clear that the structures and capabilities of Noark can be defined in terms of the those in Moreq2010.

2. Supporting data interchange in the short term. While waiting for ‘Noark as Moreq’ there is a probably need to transform or understand data from Noark5 as Moreq2010 data. This could be to enable data from Noark5 systems to be loaded into a Moreq2010 system. This would open up the possibility for organisation using Noark5 to make use of tools and systems developed for a wider market.

We have chosen to define the core Noark5 data models as subclasses of the MoReq2010 model. In addition to the subclasses additional specialised constraints are also defined.

The exciting thing about this approach is that it offers up the possibility of Noark5 being defined as ‘just’ an extension of Moreq2010.

This then leads on to say that both the specialised in Noark5 and domain models for specific industries are also delivered in this fashion. This should lead to more data driven standards, more modular standards and a smaller, more testable core to the standard.

3.7 Noark5 as a Moreq2010 Data Module

This section describes how Noark5 data structures can be defined as specialisations of MoReq2010.
Most of the core structures map cleanly across. The exception is in the area of DocumentDescription and DocumentObject as these have no single, obvious comparative structure. The closest class in Moreq2010 is Component but its not clear intent is the same.

This needs some more investigation, but the current proposal is for both DocumentDescription and DocumentObject to be derived from MoReq2010 Component.

3.8 Domain Model Definitions

As well as being able to define Noark5 in terms of Moreq2010, the same data model specialisation approach can be used for domain models. The powerful thing about being able to do this in data is that it allows the core standard to be small and domain extensions are published 'just in data'.

The following definition is an example of how data structures for road agency for dealing with vehicle registration could be defined using the following model.
Extending the core data models and defining additional restrictions constrains the model to be domain specific, but also creates a model that still follows and adheres to the inherent semantics attached to the parent structures.

### 3.9 Vision for Noark Standard

Extrapolating a little way forward from all the things we have defined and shown so far, we summarise our vision on how the standard could be defined.

This high level visualisation shows how the different parts stack up. At the core is the Moreq2010 standard defined in RDF, using RDFCL and SOF. This defines the core data models such as Aggregation, Record, Component, DisposalRule, Log etc. It also states that metadata extensions and domain refinements should also be defined in data using RDFCL and the SOF.

NoarkX is then defined as a set of subclasses and property subclasses of the Moreq2010 model definitions. The semantics around these new classes are also defined in data as SOF structures.

Finally, any domain specific refinements such as those for the department of immigration, or the roads agency, or the space agency would again just define new subclasses and subproperty classes along with constraints and SOF semantics.

#### 3.9.1 What is being standardised and Why?

This question is a little bit outside the scope of the main research focus but
4 Using RDF Technology to Facilitate Automated Archive Processes

There exist many examples where organisations must submit their records for archiving to another organisation. This section of the report considers how RDF and related technologies could enable this process to be more effective and result in the highest quality possible record being stored by the aggregator. It is intended that any approach to solving this issue should work at scale. E.g. it should be viable for a small organisation to adopt at a departmental level, but also work when applied as part of a national strategy on automating archive processes.

The OAIS model describes a push based scenario where organisations create submission packages that are sent to the aggregator. These packages contain a collection of record metadata and record content.

The aggregator then has the job of unpacking, validating and storing the record metadata and the record content.

We use the term record metadata as a general term covering all the data structures supporting the description of the record. So, in Moreq2010 terms this would be the parent aggregations and classification data.

4.1 Syndication Approach to Archive Submission

The following approach utilises existing standards and protocols to offer an alternative to pushing submission packages to the archiving organisation. This approach adopts a pull based model whereby submitters expose the data structures of the records and the record files as feeds of data that can be pulled by the archiving organisation.

An organisation wishing to submit records to the archive must expose an SDShare[9] feed for all objects in their records management system that are marked as closed. They must also support that the list can be filtered by date modified.

The list of objects must be represented as an ATOM data feed and the item links in that feed must resolve, using https, to an RDF representation of the record data entity. The rules to define when an entity should appear in the list is guided by the domain and regulations, but could be described using SPARQL queries.

The submitter must also expose either REST web service that can resolve URIs that link to record content, or a second ATOM feed that contains the content inline. As the content is immutable this can be provided independently of the record feed.

The aggregator has a generic client that will consume the data from different submitters. This process can run continuously, or at intervals to fetch the feed of data items and feed of content items. The feed will only contain the items created since the last time the client asked for data.

As the record data is provided as RDF it can first be validated against the constraints defined for the appropriate part of the model. Once validated the data can simply be loaded into an RDF store. The data from each submitter can be loaded into separate graphs to keep the data logically separate, or even into separate triples stores if needed.

As part of the processing of the RDF, there is only one ‘special’ action that should occur and that is to detect the property that references the record content. This will be a well known identifier, e.g. http://arkivverket.no/noark5/content or http:// http:// arkivverket.no/moreq2010/reference and the value MUST resolve to the content item being exposed by the client.

The following diagram shows how the whole process works.
Aggregator pulls an RDF SDShare feed of the Records data from the manager.

The RDF references documents exposed over HTTP. These documents are fetched by the aggregator and stored in the archive document store.

The RDF for the records is kept in an RDF store.

One of the key use cases identified in transmission models is the way in which placeholders can be transferred. Placeholder transmission is important as it allows record submission to be gradual. E.g. if a record is closed then it can be transferred even if the containing aggregation is not yet closed.

Going back to the start of this report we showed how data described using RDF can be merged as new parts of the information are made available. This same concept is used to transfer partial bits of the model.

A pull model allows authorised organisations the ability to just join the eco-system and start consuming the data allowing reconfiguration of the flow of the archive material. No longer must it just be pushed from a to b, but now any number of organisations could consume the data.
5 Summary of Work and Future Directions

This report has covered a broad array of topics, some in more depth than others. The level of detail and maturity of different aspects should be considered. The most concrete offering to come out of this work are initial, mostly complete, RDF models, URIs and model constraints for both Moreq2010 and Noark5 data. With a little more work and community discussion these could form the basis of authoritative RDF representations.

The next area of investigation was to look at the fundamental way standards such as Moreq2010 and Noark are defined. These standards are primarily concerned with data models and operations on these models. We took the data models we had first defined and created examples of how an RDF based Semantic Operations Framework could be used to define the operational semantics of standard. In addition we also defined how Noark and Domain Models could be defined in terms of Moreq2010. This work, while exploratory, appears to have some promise, but many details have been overlooked and would need to be revisited.

The third general area was looking at how a pull based architecture could be used to provide continuous delivery of records to the archive. This concept has been proven in many corporate settings, the principles are sound and we believe it would be valuable to create a prototype implementation based on this model.
References

2. RDFS 1.1, http://www.w3.org/TR/rdf-schema/
3. SPARQL 1.1, http://www.w3.org/TR/sparql11-overview/