EVALUATION OF EXTENSIBLE MARKUP LANGUAGE (XML) SCHEMA LANGUAGES FOR THE FUNCTIONAL DESCRIPTIONS OF THE MISSION SPACE (FDMS) DATA INTERCHANGE FORMAT (DIF)

THESIS

Presented to the Graduate Council of
Southwest Texas State University
in Partial Fulfillment of
the Requirements

For the Degree

Master of Science

By

Aidan Povedano, B.S.

San Marcos, Texas
November, 2002

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr. Furman Haddix, for his invaluable support and advice throughout each stage of this research project.

I would also like to thank the Defense Modeling and Simulation Office for their sponsorship of this project.

Finally, I would like to thank Professor Wilbon Davis and Dr. Greg Hall for their contributions as members of my thesis committee.

This manuscript was submitted on October 31st 2002.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

This paper describes an evaluation of schema languages for the Extensible Markup Language (XML), in the context of the requirements of the Functional Descriptions of the Mission Space (FDMS) Data Interchange Format (DIF). The FDMS DIF is currently specified using the DTD (Document Type Definition) schema language.

As a result of this evaluation, two schema languages were recommended for the specification of the FDMS DIF. The two languages, W3C XML Schema and Schematron, are to be used in combination. W3C XML Schema is used as the primary schema language – specifying general document structure and most of the data types, and providing visualization of the FDMS DIF. Schematron is used as the secondary schema language – providing specification of complex constraints and complex data types that cannot be adequately expressed using W3C XML Schema. Schemas were written, in W3C XML Schema and Schematron, for the Model and Point Of Contact sections of the FDMS DIF. A plug-in was written, allowing Schematron validation to be

performed within the XML Spy 4.0 Integrated Development Environment (IDE), in order to facilitate the process of validating an instance document against schemas written in the two languages.

A summary of the contents:

- Section 1 contains a description of XML, DTD and the FDMS DIF.
- Section 2 describes 14 schema languages for XML and provides a feature-by-feature comparison of the languages.
- From these 14 schema languages, 6 were selected for further evaluation. The evaluation process and the results of the evaluations are described in section 3.
- Section 4 discusses ways of combining schema languages.
- Section 5 describes the process of selecting schema languages for use with the FDMS DIF.
- Section 6 describes the schemas that were written for the Model and Point Of Contact sections of the FDMS DIF, and their usage.
- Section 7 presents some conclusions.
- Appendices:
  - Appendix A contains a table comparing the 14 schema languages for XML that were studied.
  - Appendix B contains a list of the complex constraints required by the FDMS DIF.
  - Appendix C contains the source code for the schemas that were written for the FDMS

DIF, and for the plug-in that was written to facilitate use of the Schematron language within the XML Spy Integrated Development Environment.

- Appendix D describes the process of validating an instance document against the W3C XML Schema and Schematron schemas, within the XML Spy Integrated Development Environment.

1.2 Background

The Functional Descriptions of the Mission Space (FDMS) [FDMS1] is one of the components of the Department of Defense Modeling and Simulation Common Technical Framework. The FDMS is concerned with providing descriptions of military operations in terms of processes, entities and environmental factors, for use in modeling and simulation activities.

The FDMS Data Interchange Format (DIF) specifies an intermediate representation of FDMS data that is used in data interchange. The FDMS DIF is implemented using the Extensible Markup Language (XML).

XML is a language that allows data to be represented as a text-format document that contains both data and markup that describes the data. In addition to XML documents, it is often desirable to have a formal representation of the content of a class of XML documents. Such a representation is referred to in this paper as a schema. The FDMS DIF currently uses the Document Type Definition (DTD) language to define the schema for the class of XML documents

that comprises the FDMS DIF.

DTD is an established schema language for XML. However, shortcomings of the language result in a DTD being incapable of representing all aspects of the FDMS DIF. In order to adequately represent the FDMS DIF an alternative schema language is needed. This paper describes the process of evaluating schema languages in order to find a language that is capable of fulfilling the schema requirements of the FDMS DIF. A solution is presented in this paper, based on the results of the evaluation process, and some general principles for similar evaluations are given. In addition, a description of the implementation is provided.

1.3 Generalized Problem

1.3.1 Description of XML

The Extensible Markup Language (XML) [XML1-3] is used for the representation of data in text format. An XML document consists of data marked up with tags that describe the data. Markup includes elements and their associated attributes. For example, the following fragment of an XML document consists of data for a book, with elements and attributes that describe what each data item means.

```
<book price="$10">
    <author>
        <first_name>John</first_name>
```

```
            <last_name>Smith</last_name>
        </author>
        <title>XML for Beginners</title>
    </book>
```

Specifically, the fragment contains *book*, *author*, *first_name*, *last_name* and *title* elements. There is only one attribute – *price* – that is associated with the *book* element. In addition an element can contain other elements, as with the *book* element, or it can contain data, as do the *first_name*, *last_name* and *title* elements.

1.3.2 Schemas for XML

It is often desirable to have a formal definition of the content of an XML document, providing information about the individual elements, attributes and data that comprise the document, as well as the structure of the document with respect to the elements, attributes and data. Such a definition is referred to as a schema. More generally, a schema describes a class of instance documents that conform to the specifications of the schema. A schema serves two important functions:

- Provides a human-readable description of what is valid in a particular class of XML documents. An XML document author can then refer to the schema when writing an XML document, in order to ensure that the XML document is consistent with other documents that

use the same schema.

- Provides a machine-readable description of what is valid in a particular class of XML documents. This allows different systems to share their data using a common format defined in the schema, for example.

1.3.3 DTDs

The Document Type Definition (DTD) [XML1] [DTD1] is the oldest and most established schema language for XML. The following example shows a fragment of a DTD that defines the content of the XML document fragment presented in the previous example:

```
<!ELEMENT book (author, title) >

<!ATTLIST book price CDATA #REQUIRED >

<!ELEMENT author (first_name, last_name)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT first_name (#PCDATA)>

<!ELEMENT last_name (#PCDATA)>
```

The declarations in a DTD can define:

- An element that contains other elements. The first line of the DTD above specifies that a *book* element must contain an *author* element followed by a *title* element. More complex element content models can also be expressed, specifying cardinality of elements and choices

between elements.

- An element that contains data. For example, the last line of the DTD specifies that a *last_name* element must contain parsed character data.
- An attribute that is associated with an element. For example, the second line of the DTD specifies a *price* attribute and associates it with the *book* element.

This simple example does not show all the features that DTDs provide. For more information refer to [WROX] or [FDMS1].

1.3.4 Shortcomings of DTDs

DTD has been the standard schema language for XML since the language was first created. As such, it is widely used and supported. However, DTDs have several limitations:

- DTDs are not written in XML but instead use a different syntax. The DTD syntax has the advantage of being compact. However, by not using XML syntax, DTDs bring extra complexity to the task of using XML. The XML author has to learn two types of syntax – one to use for the schema and one for the instance document. Also, because they are not written in XML, DTDs cannot be manipulated or rendered graphically by the large number of XML tools that are currently available.
- DTDs provide only a basic description of the structure of an XML document. DTDs are not capable of expressing certain types of structural constraints that may apply to an XML

document. A DTD can express that element *a* must contain at least one of element *b*, for example, but cannot efficiently express that element *a* must contain 14 or more of element *b*. Certain other types of structural constraints can be expressed using a DTD but only with the use of a convoluted definition.

- DTDs provide very little information about the data types of the data content of elements and attributes in the instance document. It is not possible to specify, for example, that a particular element must contain an integer value.
- DTDs do not facilitate reusability. It is not possible to import definitions into a DTD from another DTD. Similarly, it is not possible to split a large complex DTD into more manageable components in different files. Reusability within a DTD can be achieved to a limited degree through the use of parameter entities, but a more powerful mechanism for reusability is needed.

1.3.5 Desirable Properties of a Schema Language for XML

As discussed above, DTDs do not provide a perfect solution as a schema language for XML. A suitable replacement for DTD would be a schema language with the following properties:

- Built-in data types such as integer, string, etc. which are provided by the schema language and can be referenced by the schema author.

- A mechanism for building user-defined data types, in order to build types such as an ISBN data type for books.
- Mechanisms for creating reusable definitions and for reusing those definitions, facilitating a modular design approach.
- Support for reuse of definitions from other schemas – for example, through the use of namespaces.
- Simplicity – the language should be easy to learn and use. A small set of orthogonal language features would be ideal. Although complexity is a disadvantage, there is a trade-off between the power of a language and its simplicity.
- Legibility, and ease of reading and writing.
- Uses XML syntax, which has the advantage that users only need to learn one syntax, rather than two. Another advantage of using XML syntax is that a large number of tools exist for XML, and these could be applied to the schema language also.
- Capability to express complex constraints. In addition to structural constraints, as discussed in the previous section, the expression of "semantic" constraints may sometimes be of value. For an example of a semantic constraint consider a *person* element that contains an *age* element and a *drivers_license_number* element. A schema that defined the *person* element could also specify a semantic constraint, specifying that the *drivers_license_number* element must have null content if the *age* element contains a integer smaller than 16. The use of

semantic constraints thus brings some domain knowledge into the schema.

- If a single schema language cannot fulfill all the desired requirements then a combination of schema languages may provide the answer. This gives rise to another desirable property of a schema language – the capability of the schema language to be used in conjunction with one or more other schema languages, in describing a single class of XML documents.

1.4 Specific Problem

1.4.1 Problem Domain

The following description of the problem domain was adapted from section 1 of [FDMS2]:

> The Functional Descriptions of the Mission Space (FDMS) Data Interchange Format (DIF) [FDMS1] has been developed for the exchange of functional descriptions of military operations without loss or distortion of content, for the purpose of supporting the development of models and simulations in the military operations domain. The FDMS is comprised of four primary components:
> - The Model Representation (the military operations content)
> - The Meta-Data Representations (registration and configuration management content)
> - The Transmittal File Formats (the arrangement of data within and across files)
> - Business Rules (minimum required contents, specified relationships among contents)
>
> Within the FDMS DIF, each data representation is specified by two documents:
> - The Common Semantics and Syntax Document (the logical meaning and content of the data). This document is applicable to all the data representations.
> - A Data Interchange Format document (formal declaration of the XML data structures). Each data representation is described by its own Data Interchange Format (DIF) document.

This research is concerned with the latter item, the FDMS DIF documents. The FDMS DIF is currently specified using a DTD. The goal of this research is to investigate the use of other schema languages in this role, culminating in the recommendation of a *schema language solution* for the specification of the FDMS DIF. The term *schema language solution,* rather than simply "schema language" is used in order to allow for the possibility of using a combination of schema languages for the specification of the FDMS DIF.

The research focuses on the application of the schema languages to the following two sections of the FDMS DIF:

- **The FDMS Point of Contact DIF.** This DIF forms part of the FDMS Meta-Data Representations. The Point of Contact DIF "provides reusable information concerning points of contact" [FDMS3]. The Point of Contact DIF is used because it is small and manageable, yet has many data type requirements.
- **The FDMS Model DIF.** "Functional mission space models are simulation implementation-independent functional descriptions of the real-world processes, entities, environmental factors, and their associated relationships within the context of a set of military missions, operations, or tasks" [FDMS2]. The FDMS Model DIF is a good choice for two reasons –
  - The FDMS Model DIF is an integral part of the FDMS DIF
  - The FDMS Model DIF is very complex and requires the expression of several complex

constraints.

1.4.2 Current Solution

The existing DTD for the FDMS DIF does a good job of specifying the structure of a valid instance document. The following are specified:

- The names of the allowed elements
- The occurrences of elements (mandatory, optional, zero-or-more, one-or-more)
- The content of elements:
  - Text
  - Other elements
- Attributes

Considering the volume of information expressed, the DTD can be considered to be relatively concise. However, the DTD lacks the following features, which cannot be implemented due to limitations of the DTD language:

- Constraints on the data types of the data content of elements and attributes. In the FDMS DIF, descriptions of the required data types are provided throughout the DTD, as human-readable comments
- Specifications of complex constraints – structural and semantic – as discussed in section 1.1.5. The documentation for the Model DIF [FDMS2] provides a list of complex

constraints that are required.

- Modularity – at present the entire FDMS DIF DTD must reside in one file. However, because of the size of the DTD, it would be advantageous to have one schema composed out of several sub-schemas, each in its own file.

Conversion of the FDMS DIF DTD to a schema representation that uses a different schema language is simplified by the following:

- The DTD does not make use of certain DTD language features – such as entities and default insertion of attribute values – which are not supported by many other schema languages.
- The DTD uses very few attributes – for example, only the root element in the FDMS DTD specifies attributes, in order to facilitate editing. Limited use of attributes results in a simpler schema.

1.4.3 Requirements

A list of requirements for a schema language solution for the FDMS DIF was compiled. The list was compiled following consideration of the desirable properties of a schema language, as given in section 1.2.5, and study of the FDMS DIF. These requirements will guide the evaluation of schema languages in this research.

Requirement 1.4.3.1

The schema language solution should be capable of expressing all the same information that is contained in the current DTD solution, without resorting to contortion of the language. Note that this research does not require that all DTD capabilities be replicated, but only those currently in FDMS DIF use.

Requirement 1.4.3.2

There should be mechanisms for specifying the data types of the data content of the elements in the FDMS DIF. The data types required can be categorized as follows:

- Basic data types, such as those that are available in most programming languages. Examples are string, integer and Boolean.
- Restricted versions of basic data types. Examples of such types are a string of maximum length 20, and an integer that must have a value in the range 5 – 12.
- Complex data types, such as data types that specify the format of an email address, Uniform Resource Locators (URLs), or dates.
- Reusable user-defined data types - in general, the same data type requirement tends to be used by several elements in the FDMS DIF. Therefore it is desirable to be able to reuse data type specifications. A collection of built-in data types, that is data type specifications that are supplied with the schema language, would also be desirable.

Requirement 1.4.3.3

The schema language solution should be capable of expressing complex constraints that cannot be expressed using a DTD. The FDMS DIF has a number of structural and semantic constraints that could be expressed in its schema, thereby building domain knowledge into the schema.

Requirement 1.4.3.4

The FDMS DIF is large and complex – the schema language solution should be capable of rendering the FDMS DIF in a readable manner. The resulting schemas should provide an improved level of readability over the DTD representation.

Requirement 1.4.3.5

The schema language solution should have the support of high quality software tools. A validator – a tool that validates an XML instance document against its schema to ensure conformance – is an essential tool. Desirable tools include tools for graphical representation of schemas, and Integrated Development Environments (IDEs).

Requirement 1.4.3.6

The schema language should allow a modular approach to the creation of a schema, thereby facilitating reusability and maintainability.

1.5 Research Goals

The primary goals of this research are:

- To evaluate current schema languages for XML, based on
  - The desirable properties of a schema language (section 1.2.5)
  - The requirements of the FDMS DIF (1.4.3)
- To investigate the possibility of using a combination of schema languages to express a schema.
- To propose a schema language solution for the FDMS DIF.
- To validate the schema language solution by extended implementation.
- To develop a methodology for determining the appropriate schema language, or combination of schema languages, for a given application.

CHAPTER 2

SURVEY OF SCHEMA LANGUAGES

2.1. Introduction

A survey of available schema languages was conducted. The comparison of schema languages in [LANGCOMP1-3] provided a starting point for the survey. Section 2.2 gives a brief description of the 14 schema languages in the survey. A table comparing the 14 schema languages and DTD is presented in Appendix A.

2.2. Overview of the Schema Languages

2.2.1. DDML (Document Definition Markup Language)

DDML [DDML1] is a relatively early schema language. The language provides similar functionality to a DTD, but uses XML syntax. DDML also provides a few features that are not available with a DTD, such as support for namespaces and embedded documentation. However, DDML does not provide any additional data typing capabilities over those provided by DTDs.

2.2.2. Xschema

XSchema [XSCH1] is the predecessor to DDML and is very similar to that language, using XML syntax to deliver functionality that is similar to that of a DTD.

2.2.3. W3C XML Schema

As a recommendation of the W3C committee, W3C XML Schema [W3CSCH1-9] has strong backing from industry. High quality tools are available that support the language. The language itself is relatively complex and difficult to learn. However, it provides considerable power. A very rich set of built-in data types is provided, for use with both element and attribute definitions. A mechanism is provided for building user-defined data types.

2.2.4. XDR (XML - Data Reduced)

XDR [XDR1-2] is important because it is used in Microsoft's BizTalk and is supported by Microsoft's XML parser, MSXML. The language is similar to the W3C XML Schema language but with a more limited set of built-in data types and fewer features, making it less complex and easier to learn than W3C XML Schema. In particular, there are no mechanisms for creating user-defined data types, although mechanisms are provided for placing restrictions on the built-in types. The language could be considered a predecessor to the W3C XML Schema language.

2.2.5. DCD (Document Content Description)

DCD [DCD1], like XDR, can be considered a predecessor to the W3C XML Schema language. The language offers similar functionality to XDR, with a few exceptions, such as the number of built-in data types.

2.2.6. SOX (Schema for Object-oriented XML)

SOX [SOX1] makes use of concepts from Object Oriented programming and provides several improvements over DTDs, including built-in data types, extensions to built-in data types, reusability and modularity.

2.2.7. TREX (Tree Regular Expressions for XML)

TREX [TREX1-3] offers a simple, orthogonal syntax and a high degree of modularity. The language provides full support for W3C XML Schema data types. TREX is easy to learn because it has only a small number of language features, yet it is still very expressive, relative to DTDs. TREX allows certain types of context sensitive definitions, such as the content of an element being dependent on its parent element.

2.2.8. RELAX (Regular Expression Language for XML)

Like TREX, RELAX [RELAX1-3] offers a simpler, more orthogonal syntax, a higher degree of modularity, and more expressiveness than DTDs. The language makes use of W3C XML Schema data types but does not provide full support for all the data typing features available in W3C XML Schema. RELAX allows some context-sensitive definitions.

2.2.9. DSD (Document Semantic Definition)

DSD [DSD1-3] allows the structure of a class of XML documents to be expressed in a context-sensitive manner. The language makes use of Boolean expressions and regular expressions. Some characteristics of DSD are:

- Context-sensitive definitions of element content, and a mechanism for expressing key/keyref relationships allow a schema written in DSD to convey some of the semantics of the class of XML documents being described, as well as their structure.
- DSD provides very usable regular expression syntax for building complex string data types.
- DSD allows a high level of modularity and reusability.
- DSD provides a sophisticated mechanism for the insertion of defaults into an XML instance document.

2.2.10. SAF (Schema Adjunct Framework)

An SAF [SAF1-2] document consists of XML-format metadata that associates a

schema, and the set of instance documents conforming to the schema, with application-specific or domain–specific information. For example, a particular element could be linked, via an SAF document to a particular attribute in a database table. SAF does not provide the functions of a schema language, as listed in section 1.2.2, and is therefore not appropriate to the problem addressed in this research.

2.2.11. RDF (Resource Description Framework)

An RDF [RDF1-2] schema can provide metadata about any type of resource, where the resource could be an XML document, or any number of other web resources, such as image files and online books. Resources and relationships between resources can be described. Elements and attributes in an XML document can be described in a high-level manner, but not to the level of syntax.

2.2.12. Hook

Hook [HOOK1] is a new language proposal that is designed for the creation of very compact schemas. A schema in Hook consists of only one element. This single element contains a very concise description of the structure of an instance document, consisting only of the names of the allowable elements and their ordering within the document. A schema in Hook is intended to provide quick validation in order to determine, in real time, whether an instance

document is approximately valid. Such a schema could be useful in data transfer, or any other situations where time might be more important than accuracy.

2.2.13. Schematron

While most schema languages provide a means to specify a grammar that describes a valid class of XML instance documents, Schematron [SCHTR1-6] uses a different approach to validation -- a rule-based approach. A schema written in Schematron consists of a collection of rules. Each rule is evaluated at a specified point in the XML instance document, in order to test whether a particular aspect of the instance document is valid.

2.2.14. Examplotron

Examplotron [EX1-2] allows a schema to be built from examples - sample XML instance documents. The simplest possible Examplotron schema is a single XML instance document. Any other instance document that matches the structure of this instance document is considered to conform to the schema. The expressive power of the schema can be increased by using a combination of sample XML instance documents to create a schema, and by annotating the examples with Examplotron code that can further constrain or generalize the structure. Schematron-style rules can be used to annotate the examples, providing a lot of expressive power.

2.3. Schema Language Comparison Chart

Appendix A contains a table comparing the 14 languages described above, and DTD.

CHAPTER 3

EVALUATION OF SELECTED SCHEMA LANGUAGES

3.1. Selection of Schema Languages for Evaluation

3.1.1. Selected Schema Languages

Using the comparison of schema languages in Section 2, six schema languages were chosen for further evaluation. These languages are listed below, with indications as to why they were selected.

TREX

Allows context sensitive definitions, which could be of use in implementing some of the complex constraints required by the Model DIF. Provides a very simple yet powerful syntax.

RELAX

Allows context sensitive definitions.

W3C XML Schema

This is the most comprehensive of the schema languages, providing a large number of features, in particular features for data typing. The language also has wide acceptance and is one of the more future-proof languages surveyed.

DSD

Allows context sensitive definitions. Provides a very usable syntax for expressing string types.

Schematron

Provides very powerful specification of complex constraints.

Examplotron

Takes an approach to document validation that is unique among the schema languages surveyed and that could provide high readability.

3.1.2. Reasons for Elimination of Remaining Languages

DDML, Xschema

These languages offer relatively little functionality beyond that offered by DTD.

XDR, DCD, SOX

These languages are similar to W3C XML Schema but offer fewer features, in particular data typing features (see Requirement 1.3.3.2). Conversely, these languages do not offer any features that are not offered by W3C XML Schema and that would be of use in the FDMS DIF (according to the requirements in 1.3.3).

SAF, RDF, Hook

These languages are not appropriate for use with the FDMS DIF –

- SAF and RDF are not concerned with the validation of an XML instance document.
- Hook is concerned only with whether a document is approximately valid – the FDMS DIF requires more precision.

3.2. General Process of Evaluation

3.2.1. Introduction

This section describes the general process that was used to evaluate each of the remaining languages. The evaluations were carried out in three phases:

- Research Phase – The language was studied and a survey of the tools and resources available for the language was conducted.

- Practical Application Phase – The language was applied to the FDMS problem and the results of this application were evaluated.
- Conclusion Phase – Consisting of an overall evaluation of the language with particular emphasis on its use in the FDMS domain.

These three phases are described in more detail in each of the three subsections, 3.2.2 through 3.2.4.

3.2.2. Research Phase

Research was mainly conducted using resources available on the World Wide Web, such as web sites, white papers, language specifications, tutorials and message boards. Printed resources generally did not provide much information on the schema languages, but did contain useful information on the XML technologies behind some of the languages.

There were two components to the research conducted on each language – a study of the language and its features, and a survey of the tools and resources available for the language.

The categories of tools and resources surveyed, along with their evaluation criteria, are given below.

Validators - essential

A validator is a program that checks that an XML instance document is valid against

its schema. This usually also involves checking that the schema is valid against the specification of the language. Validators may be implemented as executables written in a compiled language such as C++ or Visual Basic or as XSL style sheets. In the latter case, the validation is performed through an XSLT processor.

Minimum Criteria:

- Is there at least one validator for the language?
  - Does this validator work?
  - Does it implement at a minimum the set of language features required by the FDMS DIF?

Language Specifications - desirable

A language specification is essentially a schema for the schema language, typically in the form of a DTD (or sometimes a W3C XML Schema). A language specification can be used to validate a schema in order to verify that the schema is correctly written, according to the rules of the schema language. Such a specification is useful when editing a schema in an IDE that supports DTD or W3C XML Schema. It can also be used to create a customized editing tool for the schema language.

Minimum Criteria:

- Is there at least one specification for the language, either a DTD or a W3C XML Schema?

Integrated Development Environments (IDEs) - desirable

Two well-known IDEs are XML Spy (produced by Altova), and Turbo XML (produced by TIBCO Software Inc.). Both offer a very usable environment with features that save the programmer time, and manage complexity. However, each of these IDEs supports only a few schema languages.

XML Spy was used to write all the schemas in the practical application phase (section 3.2.3). Out of the six languages evaluated here, only one - W3C XML Schema - is supported by XML Spy. Some features provided by XML Spy for working with W3C XML Schema are:

- Automated validation of an instance document against a schema
- Tools for creating and editing a schema
- Conversion of a DTD to a schema.

An IDE such as XML Spy is useful even when the schema language is not supported by the IDE. For example, XML Spy provides the following features for manipulation of XML instance documents that can also be used for schemas (providing the schema language uses XML syntax):

- A feature that checks that an XML document is well formed, that is, the document obeys all the rules of XML.
- A feature that checks that an XML document is valid against a particular DTD or W3C XML

Schema. Using this feature, a schema written in any language can be checked for validity against a DTD or W3C XML Schema specification of the language. If the validation generates an error, then the schema does not conform to the rules of the schema language, according to its specification. This feature, then, can be used to enforce that the schema conform to the rules of its language, provided though that the following conditions are met:

  - The schema language must be an XML language, that is, a schema written in the language must conform to the rules of well-formed XML. This condition is not an issue for the languages evaluated here, because they are all XML languages.
  - A DTD or W3C XML Schema specification must exist for the language.

- Time saving features such as auto-complete.
- Automatic XSLT (eXtensible Stylesheet Language Transformation) transformation of an XML document, using an XSL style sheet.
- A color-coded text view of an XML document or schema
- A tabular view - the "Enhanced Grid View" - of an XML document or schema

Minimum Criteria:

- Is there an IDE that has at least the following support for the language?
  - Editing of schemas
  - Verification that a schema conforms to its language specification

- Automated validation of instance documents against a schema

Tools for Graphical Representation of Schemas - desirable

An example of such a tool is an XSL style sheet that can be used by an XSLT processor to transform a schema into an HTML graphical representation of the schema.

Minimum Criteria:

- Is there a tool for graphically displaying a schema, and if so, is the schema rendered in a widely accepted format, such as HTML?

Resources for Learning the Language - essential

Required resources include tutorials and formal specifications, such as BNF grammars, that provide details not covered in the tutorials. Other resources, such as postings on message boards devoted to the language, and sample schemas, are very useful. However, they are less likely to be available for newer or lesser-known languages.

Minimum Criteria:

- Are there sufficient resources, either online or in print, for learning the language and applying it to the FDMS DIF?

3.2.3. Practical Application Phase

Schemas were written in each of the languages for the following two sections of the FDMS DIF that were introduced in section 1.3.1:

- Point of Contact – This was used to test the data typing capabilities of the language.
- Model – This was used to test how the language handles a complex schema. Rather than use the Model DIF in its entirety, a reduced version of the Model DIF was prepared. The reduced Model DIF is an abstracted form of the original – it is intended to have the same level of complexity, in terms of constraints, structure and relationships between elements as the original, while having a smaller size that is more conducive to experimentation. With the Model DIF, the emphasis was on the implementation of complex constraints and the expression of a complex schema in a readable manner. Although the Model DIF has data typing requirements, these were generally not considered in great detail when creating schemas for the Model DIF, as the evaluation of data typing capabilities was conducted using the Point of Contact DIF.

XML Spy was used to develop the schemas. Wherever possible, tools were used to simplify the creation of the schemas – in particular, use was made of tools that convert schemas from one language to another. In some cases a schema was created by converting the original DTD for the appropriate section of the FDMS DIF into a schema in the target language, and then modifying the resulting schema as required. In other cases, conversion tools were not available and the schema had to be built from scratch.

A minimum list of specific data type and complex constraint requirements was prepared, using the documentation for the Model and Point of Contact DIFs [FDMS2] [FDMS3]. Together with the original DTDs, these requirements formed the basis for the creation of the evaluation schemas. These requirements are intended to be representative of those requirements of the Model and Point of Contact DIFs that cannot be handled by the existing DTDs, while being brief enough to allow multiple evaluations to be conducted. The requirements are given below.

Specific Data Type Requirements

In addition to the general data type requirements listed in 1.3.3.1, the following is a list of specific data types that are required by the Point of Contact DIF. Each data type requirement is indicated in comments in the original Point of Contact DTD, and/or the documentation accompanying the Point of Contact DTD (refer to [FDMS3]). The specific data type requirements are given below. These descriptions have been adapted from [FDMS3]:

- FDMS_Contact_Stereotype_Type - Allows value of 'ORG', 'PER' or 'POS'
- FDMS_String20_Type - String of up to 20 characters
- FDMS_String255_Type - String of up to 255 characters
- FDMS_POC_Short_Type1 – Allows a string of up to 20 characters, or 'TBD' or 'N/A'
- FDMS_POC_Short_Type2 – Allows a string of up to 20 characters, or 'N/A'
- FDMS_POC_Long_Type – Allows a string of up to 80 characters, or 'TBD' or 'N/A'

- FDMS_POC_State_Type – Allows a string of up to 2 characters, or 'TBD' or 'N/A'
- FDMS_Date_Type – Allows any of the following formats. It is assumed that the date values are to be semantically correct, that is the components of the date should have meaningful values, taking into account the number of days in a month, the effect of leap years, the number of hours in a day, etc.
  - "dd/mm/yy hh:mm:ss", e.g., 27/09/99 13:37:42
  - "dd/mm/yy", e.g., 27/09/99
  - "dd/mmm/yy hh:mm:ss", e.g., 27/Sep/99 13:37:42
  - "dd/mmm/yy", e.g , 27/Sep/99
  - "dd/mm/yyyy hh:mm:ss", e.g., 27/09/1999 13:37:42
  - "dd/mm/yyyy", e.g., 27/09/1999
  - "dd/mmm/yyyy hh:mm:ss", e.g., 27/Sep/1999 13:37:42
  - "dd/mmm/yyyy", e.g., 27/Sep/1999
  - "dd mmm yyyy hh:mm:ss", e.g., 27 Sep 1999 13:37:42
  - "dd mmm yyyy", e.g., 27 Sep 1999
  - "hh:mm:ss", e.g., 13:37:42
  - "N/A" (permanent null)
  - "TBD" (null that will be filled in current effort)

– "UNK" (null value that should be filled, but no current plan for fill)

Specific Complex Constraint Requirements

Section 2 of [FDMS2] lists the constraints that apply to the FDMS Model DIF. Appendix B lists only those constraints that cannot be represented using a DTD, organized into categories. For the purposes of evaluation of the schema languages, a simplified version of each constraint from each category in Appendix B was used. The intent was to have a sample set of constraints that, while being simpler than the original set of constraints, provides an accurate summary of the requirements of the FDMS Model DIF with respect to complex constraints. The simplified constraints are listed below, along with a reference to the original constraint in Appendix B:

Constraint 3.2.3.1

Reference: Constraint B-1.1

The values of all elements whose names end with "_SDS_ID" and which are children of an element whose name ends with "_Addition" must be unique.

For example, if this occurs in one part of the document...

```
<Association_Addition>
    <Association_SDS_ID>5</Association_SDS_ID>
```

... ...

</Association_Addition>

then this cannot occur elsewhere in the same document ...

<Characteristic_Addition>

<Characteristic_SDS_ID>5</Characteristic_SDS_ID>

... ...

</Characteristic_Addition>

Constraint 3.2.3.2

Reference: Constraint B-2.1

The value of an Association_SDS_ID element that is a child of an Association_Internal element must reference the value of an Association_SDS_ID element that is a child of either an Association_Addition element or an Association_Forward element.

For example, this is valid...

<Association_Internal>

<Association_SDS_ID>7</Association_SDS_ID>

... ...

</Association_Internal>

only if, elsewhere in the document, this is present...

<Association_Addition>

<Association_SDS_ID>7</Association_SDS_ID>

... ...

</Association_Addition>

Constraint 3.2.3.3

Reference: Constraint B-3.1

When an Association_SDS_ID element occurs as a child of an Association_Addition, Association_Forward or Associaton_Internal element, it must not contain any of {"N/A", "TBD", "UNK"}. However, as a child of any other element, there is no such restriction.

For example, the following is not allowed:

<Association_Addition>

<Association_SDS_ID>N/A</Association_SDS_ID>

... ...

</Association_Addition>

However, this is allowed:

<Association_Replacement>

<Association_SDS_ID>N/A</Association_SDS_ID>

... ...

</Association_Replacement>

Constraint 3.2.3.4

Reference: Constraint B-4.2

An Association_Data element contains an optional Association_Stereotype element, and an optional Association_Type element. However, if an Association_Stereotype element contains a value of "other" or is not present, then an Association_Type element must be present and must have a value other than {"N/A", "TBD", "UNK"}.

For example, the following is invalid:

<Association_Data>

... ...

<Association_Stereotype>other</Association_Stereotype>

<Association_Type>TBD</Association_Type>

... ...

</Association_Data>

Constraint 3.2.3.5

Reference: Constraint B-5.2

If an Entity_Data element contains an Entity_Stereotype element that contains a value

of "Network", then any Entity_Component elements contained in the Entity_Data element must contain an Entity_SDS_ID element whose value is a reference to an Entity_Addition element that has an Entity_Stereotype value of "Equipment".

For example, if this occurs in one part of the document…

```
<Entity_Addition>
    … …
    <Entity_Stereotype>Network</Entity_Stereotype>
    … …
    <Entity_Component>
        <Entity_Internal>
            <Entity_SDS_ID>9</Entity_SDS_ID>
        </Entity_Internal>
    … …
```

Then, elsewhere in the document, this must be present…

```
<Entity_Addition>
    <Entity_SDS_ID>9<Entity_SDS_ID>
    … …
    <Entity_Stereotype>Equipment</Entity_Stereotype>
    … …
```

</Entity_Addition>

Constraint 3.2.3.6

Reference: Constraint B-6.1

If an Entity_Data element contains either an Information_Field_Addition element or an Information_Field_Replacement element, then the Entity_Data element must also contain either an Entity_Type element containing a value of "Information" or an Entity_Stereotype element containing a value of "Information".

For example, this would be valid...

<Entity_Data>

... ...

<Entity_Stereotype>Information</Entity_Stereotype>

... ...

<Information_Field_Addition>... ...</Information_Field_Addition>

</Entity_Data>

But this would not be valid...

<Entity_Data>

... ...

<Entity_Type>Network</Entity_Type>

... ...

<Information_Field_Replacement>... ...

</Information_Field_Replacement>

</Entity_Data>

3.2.4. Conclusion Phase

Drawing on the results of the Research and Practical Application phases, an overall evaluation of the language was made, based on the following criteria:

- Language features and usability.
- The tools and resources available for the language.
- How well the language handled the requirements of the FDMS DIF – both the general requirements of section 1.3.3 and the specific requirements of section 3.2.3.
- How well the language compares to DTD in terms of handling the requirements of the FDMS DIF.

3.3. Evaluations

3.3.1. TREX (Tree Regular Expressions for XML)

3.3.1.1 Research Phase

3.3.1.1.1 Overview of the Language

TREX [TREX1-3] offers a simple, orthogonal syntax and a high degree of modularity. The language provides full support for W3C XML Schema data types. TREX is easy to learn because it has only a small number of language features, yet it is still very expressive, relative to DTDs. TREX allows certain types of context sensitive definitions, such as the content of an element being dependent on its parent element.

3.3.1.1.2 Selected Language Features

- <zeroOrMore>, <oneOrMore> and <optional> specify the cardinality of elements, providing the same functionality as the *, + and ? operators of DTDs. For example, the following schema fragment specifies that a <book> element contains at least one <author> element, and may contain a <price> element:

Example 3.3.1.1.2.1

```
<element name="book">
    <oneOrMore>
        <element name="author">
            <!--Definition of author element -->
        </element>
    </oneOrMore>
```

```
<optional>
    <element name="price">
        <!—Definition of price element -->
    </element>
</optional>
</element>
```

- <choice> and <group> are used to specify choices between elements and/or groups of elements in the content model. For example, the following schema fragment specifies that an <author> element contains either a <name> element or a <first_name> element followed by a <last_name> element:

Example 3.3.1.1.2.2

```
<element name="author">
    <choice>
        <element name="name">
            <!—Definition of name -->
        </element>
        <group>
            <element name="first_name">
```

<!—Defn of first name -->

</element>

<element name="last_name">

<!—Defn of last name -->

</element>

</group>

</choice>

</element>

- The features discussed so far apply equally to elements and attributes. Thus, a combination of attributes, or a combination of attributes and elements, is specified using the same syntax as is used to specify a combination of elements. In the following example, an <author> element can either contain a <last_name> element or have a *last_name* attribute.

Example 3.3.1.1.2.3

<element name="author">

<choice>

<element name="last_name">

<!—Definition of last name -->

</element>

```
<attribute name="last_name">
<!—Definition of last name -->
</attribute>
</choice>
</element>
```

The above specification could not be written in a DTD or in W3C XML Schema.

- A named *pattern* can be created by using <define>. A *pattern* is essentially a reusable piece of schema. A pattern can be referenced using <ref>, as the following reworking of Example 3.3.1.1.2.3 shows.

Example 3.3.1.1.2.4

```
<element name="author">
<choice>
<ref name="last_name_element_defn"/>
<ref name="last_name_attribute_defn"/>
</choice>
</element>
<define name="last_name_element_defn">
<element name="last_name">
```

```
                <!—Definition of last name -->
        </element>
    </define>
    <define name="last_name_attribute_defn">
        <attribute name="last_name">
                <!—Definition of last name -->
        </attribute>
    </define>
```

- Multiple definitions of a single element can be specified, by creating a separate pattern for each definition. The definition to use can then be chosen based on the context in the document in which the element appears.

- The referenced *pattern* can reside in a different file. However, <include> must first be used to import the contents of the file containing the referenced *pattern.* <include> can also be used to combine the contents of two schemas, as in the following example.

Example 3.3.1.1.2.5

```
<include href="schema1.trex"/>
<include href="schema2.trex"/>
```

- TREX provides a mechanism for importing the contents of a file, as described above, and then overriding definitions as required. Thus a general schema can be created, and then customized according to the requirements of the particular application.

- By default, element content is ordered. So, for example, the schema fragment in Example 3.3.1.1.2.1 will allow a <book> element to contain a <author> element followed by a <price> element, but not a <price> element followed by a <author> element. However, TREX provides <interleave> to specify that elements may appear in any order. The following example specifies that an <author> element must contain a <first_name> element and a <last_name> element, but that these can occur in any order.

Example 3.3.1.1.2.6

```
<element name="author">
    <interleave>
        <element name="first_name">
            <!—Definition of first name -->
        </element>
        <element name="last_name">
            <!—Definition of last name -->
```

```
        </element>
    </interleave>
</element>
```

<interleave> can also be used to specify that an element has mixed content, that is, that the element can contain both text and other elements, in any order.

- The data type of the data content of an element or an attribute can be specified using the following:
  - <anyString> allows the data content to consist of any string value.
  - <string> is used to specify a particular string value.
  - <data> is used to specify a data type that is defined externally to the TREX language. In principle, data types from any language can be used. In practice, existing TREX implementations only support data types from the W3 XML Schema language. When referencing W3C XML Schema data types, the W3C XML Schema namespace prefix, xsd, must be used.

The following example specifies that an <author> element must consist of

- a <title> element – containing a value of either "Mr" or "Ms"
- a <name> element – containing any string value
- an <age> element – containing any integer value

in the specified order.

Example 3.3.1.1.2.7

```
<element name="author">
        <element name="title">
            <choice>
                <string>Mr</string>
                <string>Ms</string>
            </choice>
        </element>
        <element name="name">
            </anyString>
        </element>
        <element name="age">
            <data type="xsd:integer"/>
        </element>
</element>
```

- More complex data typing can be achieved by using other features of the W3C XML Schema language. The following example uses W3C XML Schema features to specify that a

<Last_Name> element must contain a string value that does not exceed 15 characters.

Example 3.3.1.1.2.8

```
<element name="Last_Name">
<xsd:restriction base="xsd:string" trex:role="datatype">
        <xsd:maxLength value="15"/>
</xsd:restriction>
</element>
```

3.3.1.1.3 Tools and Resources

Validators

There are two TREX validators available:

- The Java validator, available at the TREX home page [TREX3], implements all the features of TREX, and also supports some W3C XML Schema data types. Support is provided for W3C XML Schema Patterns, which allow regular expressions to be used to specify the format of a data type. Restrictions on W3C XML Schema data types are also supported. The Java validator is available in two versions – as a Windows executable or as a Java Jar file.

  - The Windows executable was used for the development of the schemas in the Practical Application Phase. It is very easy to use. The executable is invoked from the

command line, supplying the filenames of the schema and the instance document. The executable does not handle W3C XML Schema Patterns, however.

- The Java Jar version requires the Java Runtime Environment and a SAX parser. In order to use W3C XML Schema Patterns, a regular expression processor is also needed. The validator is invoked from the command line.

- PyTREX, available at http://pytrex.sourceforge.net/, is a Python-language implementation. The utility is run from the command line. Data types are supported in general, but no particular data types are supported. In order to make use of a particular data type, such as an integer data type, a function must be written that conforms to the input/output specifications in the pyTREX documentation. This function can then be slotted into the source code for pyTREX. pyTREX can therefore be customized to fulfill specific data type requirements.

DTD / W3C XML Schema Specifications

No DTD or W3C XML Schema specifications were found for the TREX language. In order to verify that a TREX schema is correctly written, validate it against the TREX schema that defines the TREX language, available at the TREX home page [TREX3], using one of the validators described above.

IDEs

No XML Integrated Development Environments were found that support TREX.

Tools for Graphical Representation of Schemas

No tools were found for graphically displaying a TREX schema.

Resources for Learning the Language

- Online tutorial are available on the Web. [TREX1] [TREX2]
- [WROX] provides an introduction to the language.

Other

Available at the TREX homepage [TREX3] are two XSLT style sheets:

- Relax2trex.xsl, which converts a RELAX schema into a TREX schema.
- Simplify.xsl, which removes redundancies from a TREX schema.

Both style sheets were used in the development of the FDMS Model DIF schema in TREX.

3.3.1.2 Practical Application Phase

Creating the Schemas

The schemas were developed using the XML Spy Integrated Development

Environment (IDE). The lack of a DTD or XML Schema specification for TREX meant that it was not possible to validate the schema within XML Spy, to check for conformance to the TREX language. This validation was instead performed using the command line Java validator for TREX. Producing the schema was therefore not as easy as producing a W3C XML Schema or a DTD. Validating an instance document against the schema similarly involved leaving XML Spy and using the command-line utility.

The following steps were taken to produce each of the schemas:

- The TREX schema was generated from the original FDMS DTD. This involved three steps:
  - Convert the DTD to a RELAX schema, using a conversion tool (see the Tools section of Appendix D).
  - Convert the RELAX schema to a TREX schema via an XSLT transformation, using the Relax2Trex.xsl stylesheet.
  - Simplify the TREX schema, using the simplify.xsl XSLT stylesheet.
- The resulting schema contained a lot of unnecessary elements derived from the intermediate RELAX schema. The generated TREX schema was reworked to improve the quality of the code.
- The schema was improved, by using <define> to create reusable definitions where appropriate.
- Data type specifications were added to the schema.

- An attempt was made to implement the complex constraints (see section 3.2.3)

Results

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats were expressed efficiently but validating the day value against the month and year values proved to be almost impossible. (The only way would be to enumerate all possible combinations of day, month and year values for each possible format.) Only one of the required constraints – Constraint 3.2.3.6 - could be expressed.

As can be seen in the schema fragment in section 3.3.1.4 below, some progress was made towards implementing Constraint 3.2.3.3, which is reproduced below, using context-sensitive definitions:

> "When an Association_SDS_ID element occurs as a child of an Association_Addition, Association_Forward or Associaton_Internal element, it must not contain any of {"N/A", "TBD", "UNK"}. However, as a child of any other element, there is no such restriction."

However, the constraint could not be successfully implemented because context-sensitive definitions can only be used to specify content that *can* appear in a particular context, rather than to specify content that *must not* appear in a particular context.

3.3.1.3 Conclusion Phase

- Out of the six languages evaluated, TREX was the easiest to learn and to use, due to a small set of orthogonal language constructs. TREX provides full support for using W3C XML Schema data types within a TREX schema. These were used in the development of the evaluation schemas in TREX. TREX allows element definitions to be context-sensitive - an element can have more than one definition, allowing the definition used to depend on the context in which the element appears. However, this feature was of limited use for the FDMS DIF.
- When writing the evaluation schemas in TREX, tools for converting from other schema languages to TREX were used, to convert the original DTD to a TREX schema. The generated TREX schema was then modified manually. Tool support was sufficient for development in the language – validators and resources for learning the language were available. However, desirable tools - IDE, tools for graphically displaying a schema and DTD specification for the language - were not found.
- All data type requirements were successfully implemented except the FDMS Date type, which was implemented partially. One complex constraint was successfully implemented. (See section 3.2.3).
- Overall, TREX can offer an improvement over DTD for the FDMS DIF.

3.3.1.4 Fragment of the Model Schema in TREX

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<grammar xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:trex="http://www.thaiopensource.com/trex">
    <start>
        <ref name="Model_Representation.label"/>
    </start>
    <define name="Model_Representation.label">
        <element name="Model_Representation">
            <zeroOrMore>
                <choice>
                    <ref name="Association_Addition.label"/>
                    <ref name="Association_Replacement.label"/>
                </choice>
            </zeroOrMore>
            <zeroOrMore>
                <choice>
                    <ref name="Characteristic_Addition.label"/>
                    <ref name="Characteristic_Forward.label"/>
```

```
            <ref name="Characteristic_Replacement.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Condition_Addition.label"/>
            <ref name="Condition_Forward.label"/>
            <ref name="Condition_Replacement.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Description_Link_Addition.label"/>
            <ref name="Description_Link_Replacement.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Text_Description_Addition.label"/>
```

```
            <ref name="Text_Description_Replacement.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Entity_Addition.label"/>
            <ref name="Entity_Forward.label"/>
            <ref name="Entity_Replacement.label"/>
            <ref name="Entity_Update.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Interaction_Addition.label"/>
            <ref name="Interaction_Replacement.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
```

```
            <ref name="Process_Addition.label"/>
            <ref name="Process_Forward.label"/>
            <ref name="Process_Replacement.label"/>
            <ref name="Process_Update.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Process_Group_Addition.label"/>
            <ref name="Process_Group_Forward.label"/>
            <ref name="Process_Group_Replacement.label"/>
            <ref name="Process_Group_Update.label"/>
        </choice>
    </zeroOrMore>
    <zeroOrMore>
        <choice>
            <ref name="Process_Group_Instance_Addition.label"/>
            <ref name="Process_Group_Instance_Replacement.label"/>
        </choice>
```

```
        </zeroOrMore>
    </element>
</define>
<define name="Association_Addition.label">
    <element name="Association_Addition">
        <ref name="Association_SDS_ID_Required.label"/>
        <ref name="Association_Data.label"/>
    </element>
</define>
<define name="Association_Replacement.label">
    <element name="Association_Replacement">
        <ref name="Association_RDS_ID.label"/>
        <optional>
            <ref name="Association_SDS_ID_Optional.label"/>
        </optional>
        <ref name="Association_Data.label"/>
    </element>
</define>
<define name="Association_RDS_ID.label">
```

```
    <element name="Association_RDS_ID">
      <xsd:restriction base="xsd:string" trex:role="datatype">
        <xsd:maxLength value="20"/>
      </xsd:restriction>
    </element>
  </define>
  <!-- two versions of the Association_SDS_ID element follow.  The first is used
  when a value of "N/A" is allowed.  The second is used for situations where "N/A"
  is not allowed.  The version used depends
  on the parent element.  This is an attempt to implement Constraint 3.2.3.3 -->
  <define name="Association_SDS_ID_Optional.label">
    <element name="Association_SDS_ID">
      <choice>
        <xsd:restriction base="xsd:string" trex:role="datatype">
          <xsd:maxLength value="20"/>
        </xsd:restriction>
        <string>N/A</string>
      </choice>
    </element>
```

```
</define>
<define name="Association_SDS_ID_Required.label">
    <element name="Association_SDS_ID">
        <xsd:restriction base="xsd:string" trex:role="datatype">
            <xsd:maxLength value="20"/>
        </xsd:restriction>
    </element>
</define>
<define name="Association_Data.label">
    <element name="Association_Data">
        <optional>
            <choice>
                <ref name="Description_Link_Existing.label"/>
                <ref name="Description_Link_Internal.label"/>
            </choice>
        </optional>
        <element name="Association_Cardinality">
            <choice>
                <data type="xsd:integer"/>
```

```
            <string>N/A</string>
            <string>UNK</string>
            <string/>
        </choice>
    </element>
    <optional>
        <element name="Association_Stereotype">
            <ref name="Association_Stereotype_Values"/>
        </element>
    </optional>
    <optional>
        <element name="Association_Type">
            <xsd:restriction base="xsd:string" trex:role="datatype">
                <xsd:maxLength value="255"/>
            </xsd:restriction>
        </element>
    </optional>
    <ref name="Association_Entity.label"/>
    <ref name="Association_Entity.label"/>
```

```
        <optional>
          <element name="Association_S_D_Flag">
            <xsd:restriction base="xsd:string" trex:role="datatype">
              <xsd:enumeration value="S"/>
              <xsd:enumeration value="D"/>
            </xsd:restriction>
          </element>
        </optional>
    </element>
</define>
<define name="Association_Entity.label">
    <choice>
        <ref name="Entity_Existing.label"/>
        <ref name="Entity_Internal.label"/>
    </choice>
</define>
<!-- reusable data type... -->
<define name="Association_Stereotype_Values">
    <xsd:restriction base="xsd:string" trex:role="datatype">
```

```
<xsd:enumeration value="Allocation"/>
<xsd:enumeration value="Collaboration"/>
<xsd:enumeration value="Command"/>
<xsd:enumeration value="Communication"/>
<xsd:enumeration value="Control"/>
<xsd:enumeration value="Engagement Range"/>
<xsd:enumeration value="LIne of Sight"/>
<xsd:enumeration value="Line of Support"/>
<xsd:enumeration value="Occupancy"/>
<xsd:enumeration value="Occupation"/>
<xsd:enumeration value="Operation"/>
<xsd:enumeration value="Opposition"/>
<xsd:enumeration value="Organization"/>
<xsd:enumeration value="Possession"/>
<xsd:enumeration value="Sensing Range"/>
<xsd:enumeration value="Supply"/>
<xsd:enumeration value="Support"/>
<xsd:enumeration value="Other"/>
</xsd:restriction>
```

</define>

... ...

</grammar>

3.3.2. RELAX (Regular Expression Language for XML)

3.3.2.1 Research Phase

3.3.2.1.1 Overview of the Language

RELAX [RELAX1-3] offers a simpler, more orthogonal syntax, a higher degree of modularity, and more expressiveness than DTDs. The language makes use of W3C XML Schema data types but does not provide full support for all the data typing features available in W3C XML Schema. RELAX allows some context-sensitive definitions.

3.3.2.1.2 Selected Language Features

- An element is defined using <elementRule>, followed by <tag>.

  The following example defines an element <last_name>, which contains string content.

  Example 3.3.2.1.2.1

  <elementRule role="last_name" type="string"/>

<tag name="last_name"/>

- An <elementRule> can either specify the type of data that the element contains, as in Example 3.3.2.1.2.1, or it can specify which elements can be contained by the element. This is achieved by making a reference to each contained element.

  In the following example, an <author> element is defined. The <elementRule> for the <author> element contains a reference to the <last_name> element defined in Example 3.3.2.1.2.1. In this manner, the <author> element is defined as containing a <last_name> element.

  Example 3.3.2.1.2.2

  <elementRule role="author">

  <ref label="last_name"/>

  </elementRule>

- An <elementRule> can specify more complicated patterns of element content, by making use of the following constructs:

  - The *occurs* attribute – used to specify whether an element occurs zero-or-more times, one-or-more,times or is optional. As with DTDs and TREX, the *, +, ? symbols are used, respectively.

- <choice>, which specifies a choice between elements
- <sequence>, which specifies that elements occur in order.

The following example defines a <book> element that contains the following, in the order given:

- One or more <author> elements
- Either a <title> element, or a <description> element

(Definitions of <author>, <title> and <description> elements omitted for clarity.)

Example 3.3.2.1.2.3

```
<elementRule role="book">
        <sequence>
            <ref label="author" occurs="+"/>
            <choice>
                <ref label="title"/>
                <ref label="description"/>
            </choice>
        </sequence>
</elementRule>
<tag name="book"/>
```

- A <hedgeRule> contains a reusable specification of element content, which can then be referred to from an <elementRule>.

The following example rewrites Example 3.3.2.1.2.3, using a <hedgeRule>:

Example 3.3.2.1.2.4

```
<elementRule role="book">
        <hedgeRef label="book_content"/>
</elementRule>
<tag name="book"/>


<hedgeRule label="book_content">
        <sequence>
                <ref label="author" occurs="+"/>
                <choice>
                        <ref label="title"/>
                        <ref label="description"/>
                </choice>
        </sequence>
</hedgeRule>
```

- An element can also be defined to have mixed content, that is, containing both elements and plain text in any order.

- Attributes are defined specified within a <tag>, using <attribute>.

  The following example adds to Example 3.3.2.1.2.4 the specification that a <book> element must have a *price* attribute.

  Example 3.3.2.1.2.5

  ```
  <elementRule role="book">
      <hedgeRef label="book_content"/>
  </elementRule>
  <tag name="book">
      <attributename="price" required="true" type="string"/>
  </tag>
  ```

- Attribute definitions are reused by placing them inside an <attPool>. The contents of an <attPool> can then be referenced from a <tag>. The following example rewrites Example 3.3.2.1.2.5, making use of an <attPool>, in order to allow reuse of the *price* attribute:

  Example 3.3.2.1.2.6

  ```
  <elementRule role="book">
  ```

```
        <hedgeRef label="book_content"/>
    </elementRule>
    <tag name="book">
        <ref role="book_attributes/>
    </tag>
    <attPool role="book_attributes">
        <attribute name="price" required="true" type="string"/>
    </attPool>
```

- Data types can be specified for element or attribute content. Any of the built-in data types from W3C XML Schema can be used. However, there is no mechanism for creating custom types, as there is in XML Schema. Some constraints can be imposed on the content of an element or attribute, such as a maximum or minimum value. Enumerated values can also be specified for the content of an element or attribute. The following example specifies that the *price* attribute (taken from Example 3.3.2.1.2.6) must be an integer and must be either 10 or 20:

    Example 3.3.2.1.2.7

```
    <attribute name="price" required="true" type="integer">
        <enumeration value="10"/>
```

<enumeration value="20"/>

</attribute>

- RELAX provides a mechanism whereby one element can have multiple definitions. The same element can then be assigned a different definition, depending on the context in which it appears. The following example shows a fragment of an XML instance document, where a <title> element appears in two different contexts – as a child of a <book> element, and as a child of an <author> element. However, the <title> element has a different type of content in each of these contexts. As a child of a <book> element, the <title> element contains the title of a book, expressed as a <main_title> and a <sub_title>. As a child of an <author> element the <title> element contains a string that denotes the title of the author, for example "Dr" or "Mr".

  Example 3.3.2.1.2.8

  <book price="10">

  <author>

  <title>Dr.</title>

  <last_name>Browne</last_name>

  <author>

  <title>

```
            <main_title>Writing Schemas</main_title>
            <sub_title>For Beginners</sub_title>
        </title>
    </book>
```

Context sensitive definitions of elements are achieved by making use of the *label* and *role* attributes of an <elementDef>. The value of the *label* attribute specifes the context in which the definition applies.

```
    <elementDef role="title" label="title_of_author">
        <!—Definition of title, as child of author... -->
    </elementDef>
    <elementDef role="title" label="title_of_book">
        <!—Definition of title, as child of book... -->
    </elementDef>
```

The following line would then be used to refer to a <title> element occurring within an <author> element:

```
    <ref label="title_of_author"/>
```

- The *role* and *label* attributes of an <elementDef> can also be used to have different definitions of the same element, with each definition linked to a particular value of an

attribute of the element. For example, a <book> element could have a *single_author* attribute, with possible values of "true" or "false". Two definitions could then be created for the element content of a <book>. The first definition would be used when *single_author* has a value of "true", and would allow only one <author> element. The second definition would be used when *single_author* has a value of "false" and would allow multiple <author> elements.

- <module>, <include> and <interface> allow a REXAX schema to be split up into modular components, in different files, and combined as needed.

3.3.2.1.3 Tools and Resources

Validators

A number of validators are available for RELAX, written in Java, Visual Basic and C++. The Visual Basic validator, VBRelax, was used in the Practical Application Phase.

DTD / W3C XML Schema Specifications

A DTD specification of the RELAX language is available at the RELAX home page [RELAX4].

IDEs

No Integrated Development Environments (IDEs) were found that support RELAX.

Tools for Graphical Representation of Schemas

ViewRELAX is a tool that provides an interactive HTML-format graphical representation of a RELAX schema. ViewRELAX is available at the RELAX home page [RELAX4].

Resources for Learning the Language

A number of online tutorials are available, e.g. [RELAX2], [RELAX3]. In addition, [WROX] provides an introduction to the RELAX language.

Other

Tools are available (at the RELAX home page [RELAX4]) for conversion of a DTD to a RELAX schema and vice versa.

3.3.2.2 Practical Application Phase

Creating the Schemas

The schemas were developed using the XML Spy Integrated Development

Environment (IDE). The following steps were taken to produce each schema:

- A RELAX schema was automatically generated from the original FDMS DTD, using a conversion tool.
- The generated schema was reworked.
- Data type specifications were added.
- An attempt was made to implement the required complex constraints of section 3.2.3.

Results

All the required data types were specified successfully except for FDMS_Date_Type. String formats cannot be expressed within a data type, so the various date formats could not be specified.

Only one of the required constraints – Constraint 3.2.3.6 - could be expressed. As with TREX, the usefulness of context sensitive definitions for the purposes of the FDMS DIF was limited by the fact that context-sensitive definitions can only be used to specify content that can appear in a particular context, rather than specify content that must not appear in a particular context.

3.3.2.3 Conclusion Phase

- RELAX offers similar features to TREX. Like TREX, RELAX is a relatively simple

language and is easy to learn. As with TREX, an element can have more than one definition, allowing the definition used to depend on the context in which the element appears. However, this capability was of limited use for the FDMS DIF. RELAX makes use of W3C XML Schema data types. However, not all the data type features are provided. In particular there are no user-defined data types and no mechanisms for specifying the format of a string.

- RELAX has good tool support. Tools for converting from other schema languages to RELAX were used, to convert the original DTD to a RELAX schema. The generated RELAX schema was then modified manually. Tool support was sufficient for development in the language – validators and resources for learning the language were available. In addition, some desirable tools – a tool for graphically displaying a schema and a DTD specification for the language - were available. However no IDE provides full support for RELAX.
- All the required data types were specified successfully except for FDMS_Date_Type. Only one of the required constraints – Constraint 3.2.3.6 - could be expressed.
- RELAX can provide an improvement over DTD for the purposes of the FDMS DIF.

3.3.2.4 Fragment of the Model Schema in RELAX

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<module moduleVersion="1.2" xmlns="http://www.xml.gr.jp/xmlns/relaxCore">
  <interface>
    <export label="Model_Representation"/>
  </interface>
  <elementRule role="Model_Representation">
    <sequence>
      <choice occurs="*">
        <ref label="Association_Addition"/>
        <ref label="Association_Replacement"/>
      </choice>
      <choice occurs="*">
        <ref label="Characteristic_Addition"/>
        <ref label="Characteristic_Forward"/>
        <ref label="Characteristic_Replacement"/>
      </choice>
      <choice occurs="*">
        <ref label="Condition_Addition"/>
        <ref label="Condition_Forward"/>
        <ref label="Condition_Replacement"/>
```

```
</choice>
<choice occurs="*">
    <ref label="Description_Link_Addition"/>
    <ref label="Description_Link_Replacement"/>
</choice>
<choice occurs="*">
    <ref label="Text_Description_Addition"/>
    <ref label="Text_Description_Replacement"/>
</choice>
<choice occurs="*">
    <ref label="Entity_Addition"/>
    <ref label="Entity_Forward"/>
    <ref label="Entity_Replacement"/>
    <ref label="Entity_Update"/>
</choice>
<choice occurs="*">
    <ref label="Interaction_Addition"/>
    <ref label="Interaction_Replacement"/>
</choice>
```

```
        <choice occurs="*">
            <ref label="Process_Addition"/>
            <ref label="Process_Forward"/>
            <ref label="Process_Replacement"/>
            <ref label="Process_Update"/>
        </choice>
        <choice occurs="*">
            <ref label="Process_Group_Addition"/>
            <ref label="Process_Group_Forward"/>
            <ref label="Process_Group_Replacement"/>
            <ref label="Process_Group_Update"/>
        </choice>
        <choice occurs="*">
            <ref label="Process_Group_Instance_Addition"/>
            <ref label="Process_Group_Instance_Replacement"/>
        </choice>
    </sequence>
</elementRule>
<tag name="Model_Representation"/>
```

```
<elementRule role="Association_Addition">
   <sequence>
      <ref label="Association_SDS_ID_String20"/>
      <ref label="Association_Data"/>
   </sequence>
</elementRule>
<tag name="Association_Addition"/>
<elementRule role="Association_Replacement">
   <sequence>
      <ref label="Association_RDS_ID_String20"/>
      <choice occurs="?">
         <ref label="Association_SDS_ID_NA"/>
         <ref label="Association_SDS_ID_Length20"/>
      </choice>
      <ref label="Association_Data"/>
   </sequence>
</elementRule>
<tag name="Association_Replacement"/>
 <!-- Two versions of the definition of the Association_RDS_ID element follow.
```

```
Which version to use depends on the parent element in the particular situation-->
<elementRule role="Association_RDS_ID" label="Association_RDS_ID_NA"
type="string">
  <enumeration value="N/A"/>
</elementRule>
<tag name="Association_RDS_ID"/>
<elementRule role="Association_RDS_ID" label="Association_RDS_ID_String20"
type="string">
    <minInclusive value="0"/>
    <maxInclusive value="20"/>
</elementRule>
<!-- Two versions of the definition of the Association_SDS_ID element follow.
Which version to use depends on the parent element in the particular situation-->
<elementRule role="Association_SDS_ID" label="Association_SDS_ID_NA"
type="string">
    <enumeration value="N/A"/>
</elementRule>
<tag name="Association_SDS_ID"/>
<elementRule role="Association_SDS_ID" label="Association_SDS_ID_String20"
```

```
 type="string">
        <minInclusive value="0"/>
        <maxInclusive value="20"/>
</elementRule>
<elementRule role="Association_Data">
    <sequence>
        <choice occurs="?">
            <ref label="Description_Link_Existing"/>
            <ref label="Description_Link_Internal"/>
        </choice>
        <ref label="Association_Cardinality"/>
        <ref label="Association_Stereotype" occurs="?"/>
        <ref label="Association_Type" occurs="?"/>
        <choice>
            <ref label="Entity_Existing"/>
            <ref label="Entity_Internal"/>
        </choice>
        <choice>
            <ref label="Entity_Existing"/>
```

```
            <ref label="Entity_Internal"/>
        </choice>
        <ref label="Association_S_D_Flag" occurs="?"/>
    </sequence>
</elementRule>
<tag name="Association_Data"/>
<elementRule role="Association_Cardinality" type="integer">
    <minInclusive value="0"/>
</elementRule>
<tag name="Association_Cardinality"/>
<elementRule role="Association_Stereotype" type="string">
    <enumeration value="cost"/>
    <enumeration value="Allocation"/>
    <enumeration value="Collaboration"/>
    <enumeration value="Command"/>
    <enumeration value="Communication"/>
    <enumeration value="Control"/>
    <enumeration value="Engagement Range"/>
    <enumeration value="Line Of Sight"/>
```

```
        <enumeration value="Line Of Support"/>
        <enumeration value="Occupancy"/>
        <enumeration value="Occupation"/>
        <enumeration value="Operation"/>
        <enumeration value="Opposition"/>
        <enumeration value="Organization"/>
        <enumeration value="Possession"/>
        <enumeration value="Sensing Range"/>
        <enumeration value="Supply"/>
        <enumeration value="Support"/>
        <enumeration value="Other"/>
    </elementRule>
    <tag name="Association_Stereotype"/>
    <elementRule role="Association_Type" type="string">
        <minInclusive value="0"/>
        <maxInclusive value="255"/>
    </elementRule>
    <tag name="Association_Type"/>
    <elementRule role="Association_S_D_Flag" type="string">
```

```
        <enumeration value="S"/>
        <enumeration value="D"/>
        <enumeration value=" "/>
    </elementRule>
    <tag name="Association_S_D_Flag"/>

        ... ...
</module>
```

3.3.3. W3C XML Schema

3.3.3.1 Research Phase

3.3.3.1.1 Overview of the Language

As a recommendation of the W3C committee, W3C XML Schema [W3CSCH1-9] has strong backing from industry. High quality tools are available that support the language. The language itself is relatively complex and difficult to learn. However, it provides considerable power. A very rich set of built-in data types is provided, for use with both element and attribute definitions. A mechanism is provided for building user-defined data types.

3.3.3.1.2 Selected Language Features

- Two mechanisms are provided for defining elements – one is used for elements that contain only data content, the other is used for elements that contain other elements and /or attributes. The two mechanisms are illustrated below:

  - The following example defines a <title> element, which contains a string value:

    Example 3.3.3.1.2.1

    ```
    <xsd:element name="title" type="xsd:string"/>
    ```

  - If an element contains other elements and/or attributes, the element must be defined as a complex type. The following example defines a <book> element. A <book> element contains an <author> element and has a *price* attribute, with <author> and *price* both containing string values.

    Example 3.3.3.1.2.2

    ```
    <xsd:element name="book">
        <xsd:complexType>
            <xsd:element name="author" type="xsd:string"/>
            <xsd:attribute name="price" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    ```

- As with TREX and RELAX, definitions of elements can be reused. The following example modifies Example 3.3.3.1.2.2 so that a <book> element also contains a <title> element. This is done by making a reference (using the *ref* attribute of <xsd:element>) to the definition of the <title> element given in Example 3.3.3.1.2.1.

Example 3.3.3.1.2.3

```
<xsd:element name="book">
        <xsd:complexType>
                <xsd:element name="author" type="xsd:string"/>
                <xsd:element ref="title"/>
                <xsd:attribute name="price" type="xsd:string"/>
        </xsd:complexType>
</xsd:element>
```

- W3C XML Schema provides three constructs to further define the structure of element content:
  - <xsd:sequence> - specifies that elements must occur in sequence
  - <xsd:choice> - specifies a choice between elements or groups of elements
  - <xsd:any> - specifies that elements can occur in any order

The following example defines a <book> element that contains the following, in the order

given:

- One <author> element
- Either a <title> element, or a <description> element

(Definitions of <author>, <title> and <description> elements omitted for clarity.)

Example 3.3.3.1.2.4

```
<xsd:element name="book">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="author" type="xsd:string"/>
            <xsd:choice>
                <xsd:element ref="title"/>
                <xsd:element ref="description"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute name="price" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

- Two attributes, *minOccurs* and *maxOccurs*, allow occurrences to be specified. Unlike with

DTDs, RELAX and TREX, occurrences are not limited to optional, zero-or-more, and one-or-more; rather, occurrences are specified as a range. Example 3.3.3.1.2.1 is rewritten below to specify that there can be either one or two <title> elements:

Example 3.3.3.1.2.5

```
<xsd:element name="title" type="xsd:string" minOccurs="1"
                                    maxOccurs="2" />
```

- A rich set of built-in data types is provided. The use of built-in data types has already been illustrated in Example 3.3.3.1.2.1, where the <title> element is defined as having content of type xsd:string. Some of the other built-in data types available are: decimal, integer, date, url, Boolean.

- User-defined data types can be specified, using the built-in data types as a base. <xsd:simpleType> is used. A user-defined type can make use of three constructs – xsd:restriction, xsd:list and xsd:union – as described below.

  - xsd:restriction allows restrictions to be applied to a built-in data type. For example, the <author> element (from Example 3.3.3.1.2.2) could be specified as containing a string of not more than 20 characters. This is done by using the xsd:string built-in type as a base, and then using the *facet* xsd:length to specify the maximum length of the

string:

Example 3.3.3.1.2.6

```
<xsd:element name="author">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:length value="20"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

A collection of facets is available for use with each built-in data type. For example, the built-in integer data type can be restricted, using a facet that specifies a maximum value. xsd:pattern is a very useful facet that is available for use with the xsd:string built-in data type. xsd:pattern is used to specify a regular expression that defines the format of a string value. The following example uses the xsd:pattern facet to specify the data type of the *price* attribute in Example 3.3.3.1.2.2, specifying that the value must consist of a dollar sign followed by two digits. The definition of the *price* attribute becomes the following:

Example 3.3.3.1.2.7

```
<xsd:attribute name="price">
```

```
<xsd:simpleType>
  <xsd:restriction base="xsd:string>
    <xsd:pattern value="$¥d{2}"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
```

Another useful facet is xsd:enumeration, which can be used to specify a number of allowed values.

- xsd:list allows a user-defined data type to be specified as a whitespace-delimited list of values of a particular built-in or user-defined data type.
- xsd:union allows a user-defined data type to be specified as the union of two or more built-in or user-defined types. A valid value only has to match one of the types in the union.

- A uniqueness constraint can be placed on the content of an element. So, for example, the <book> element could be defined to be unique, based on the content of its <author> element.

- A key/key reference mechanism is also provided. This is similar to the primary key/foreign key mechanism in relational databases.

- Several mechanisms are provided that allow for greater modularity and reusability of schema components:
  - The examples given thus far have used only anonymous types – complexTypes and simpleTypes that are defined within an element definition and cannot be reused. However, both complexTypes and simpleTypes can be instead be assigned a name and then be referenced from multiple definitions – these are referred to as reusable named types.
  - Definitions in another file can be included into the current schema. So for example, one file could contain all data type definitions, specified as reusable simpleTypes. These definitions could then be used by the main schema, which would reside in a separate file.
  - Definitions in another file can be included and then redefined by the current schema.
  - Element definitions can be collected into a <group> and attribute definitions can be collected into an <attributeGroup>. Both group and attributeGroups can be named and reused.
  - Several advanced language features – substitution groups, abstract types and final types – enhance reusability using object-oriented concepts.

- Two types of documentation are supported – human-readable and machine-readable. Machine-readable documentation, provided by <appinfo>, can be used for providing processing information to the recipient system, or can contain schema definitions in another schema language. Either way, the W3C XML Schema validator will not be concerned with the contents of the machine-readable documentation.

3.3.3.1.3 Tools and Resources

Validators

Several validators are available, written in a variety of languages, including C++, Java and Python. Both commercial and free implementations are available. The validator included in the XML Spy Integrated Development Environment (IDE) was used to test the schemas written in the Practical Application phase

DTD / W3C XML Schema Specifications

A DTD specification is available.

IDEs

Several Integrated Development Environments (IDEs) are available. Turbo XML and XML Spy both provide advanced graphical renditions of schemas, tools for conversion of

W3C XML Schemas to DTDs and other schema types, and editing tools. XML Spy was used to develop the schemas for this paper.

Tools for Graphical Representation of Schemas

The Schema Design View, included in the XML Spy IDE, provides a highly intuitive display of a W3C XML Schema.

Resources for Learning the Language

Several tutorials are available on the web, as listed in the References section. The language is covered by some of the XML textbooks on the market, such as [WROX]. In addition there is at least one textbook devoted entirely to the language, [W3CSCH6].

3.3.3.2 Practical Application Phase

Creating the Schemas

The schemas were developed using the XML Spy Integrated Development Environment (IDE). The following steps were followed for each schema created:

- A W3C XML Schema was automatically generated from the original FDMS DTD. This was achieved by using the Convert DTD/Schema tool in the XML Spy IDE.
- Each FDMS DTD contains comments for each element, describing the requirements for the

data type of the content of the element. Using these comments as a guide, reusable <simpleType> data type definitions were created to satisfy the data type requirements.

- These <simpleType> definitions were then referenced as needed, from definitions of elements in the schema.
- In addition, some of the data type requirements were satisfied by referencing W3C XML Schema built-in data types.
- Correctness of the schema was verified using the Validate File feature in the XML Spy IDE.

Results

The W3C XML Schemas are much longer and more verbose than the original DTDs. Viewed as raw code, the schemas are no easier to read that the original DTDs. However, the Schema Design View in the XML Spy IDE provides a very intuitive, easy-to-understand representation of the schemas.

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats required were expressed successfully, using regular expressions. However, validating the day value against the month and year values could not be achieved in a reasonable manner. The only way would be to use regular expressions to enumerate all possible combinations of day, month and year values for each possible format. There is no mechanism for performing numerical analysis on the content of a data value.

W3C XML Schema provides a rich set of built-in data types. However, very few of these built-in data types were of use in creating the schemas for the FDMS DIF, as their specifications did not comply with the requirements of the FDMS DIF.

None of the required constraints could be expressed. The uniqueness mechanism provided by W3C XML Schema allows an element to contain unique content. However, this feature was not appropriate for the implementation of Constraint 3.2.3.1, which requires that all elements whose name ends in "SDS_ID" must contain unique content.

3.3.3.3 Conclusion Phase

- W3C XML Schema has a very rich set of features. However, this makes it a more difficult language to learn than TREX or RELAX. W3C XML Schema offers a large set of built-in data types and many features for creating user-defined data types. Some language features – in particular features for specifying uniqueness and for specifying key-key reference relationships – are very useful in general but were not applicable to the specific requirements of the FDMS DIF.
- Excellent tools and resources are available. At least two Integrated Development Environments (IDEs) for XML support W3C XML Schema. The XML Spy IDE provides superior visualization of a W3C XML Schema using its Schema Design View mode. It also provides automated conversion of a DTD to a W3C XML Schema.

- All data type requirements were fully implemented except the FDMS Date type, which was implemented partially. None of the complex constraint requirements could be implemented. (See section 3.2.3).
- Overall, W3C XML Schema can provide an improvement over DTD for the purposes of the FDMS DIF – in particular for data type specifications. In addition, the superior tool support for W3C XML Schema makes it a very good choice of language for the FDMS DIF. However, as noted above, W3C XML Schema cannot adequately handle all the constraints of the FDMS DIF. A good solution would be to use W3C XML Schema in combination with another language that can handle these constraints, such as Schematron.

3.3.3.4 Fragment of the Point Of Contact Schema in W3C XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xsd:annotation>
      <xsd:documentation>Schema for the Point of Contact DIF
      </xsd:documentation>
   </xsd:annotation>
   <xsd:element name="FDMS_METADATA_REP_POC_DIF">
```

```
    <xsd:annotation>
        <xsd:documentation>root element</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
            <xsd:element ref="Point_Of_Contact_Addition"/>
            <xsd:element ref="Point_Of_Contact_Replacement"/>
        </xsd:choice>
  <xsd:attribute name="Version" type="FDMS_IntegerOrNull_Type"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Point_Of_Contact_Addition">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Point_Of_Contact_SDS_ID"/>
            <xsd:element ref="Point_Of_Contact_Data"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="Point_Of_Contact_SDS_ID" type="FDMS_DS_ID_Type"/>

<xsd:element name="Point_Of_Contact_Data">
<xsd:complexType>
<xsd:sequence>
... ...
<xsd:element ref="Point_Of_Contact_Stereotype"/>
<xsd:element ref="Point_Of_Contact_Uniform_Resource_Locator" minOccurs="0"/>
... ...
<xsd:element ref="Point_Of_Contact_Employer_Start_Date"
minOccurs="0"/>
<xsd:element ref="Point_Of_Contact_Employer_End_Date"
minOccurs="0"/>
... ...
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="Point_Of_Contact_Stereotype"
```

```
type="FDMS_POC_Contact_Stereotype_Type"/>

<xsd:element          name="Point_Of_Contact_Employer_End_Date"

type="FDMS_Strict_Simple_Date_Type_1"/>

<xsd:element          name="Point_Of_Contact_Employer_Start_Date"

type="FDMS_Strict_Simple_Date_Type_1"/>

<xsd:element          name="Point_Of_Contact_Uniform_Resource_Locator"

type="xsd:anyURI"/>



... ...



<xsd:simpleType name="FDMS_POC_Contact_Stereotype_Type">

  <xsd:annotation>

     <xsd:documentation>

        Allows value of ORG, PER or POS

     </xsd:documentation>

  </xsd:annotation>

  <xsd:restriction base="xsd:string">

     <xsd:enumeration value="ORG"/>

     <xsd:enumeration value="PER"/>
```

```
            <xsd:enumeration value="POS"/>
        </xsd:restriction>
    </xsd:simpleType>

<xsd:simpleType name="FDMS_Null_Type_3">
        <xsd:annotation>
            <xsd:documentation>Allows 'N/A' only</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="N/A"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="FDMS_String20_Type">
        <xsd:annotation>
            <xsd:documentation>String of up to 20 characters
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="0"/>
```

```
            <xsd:maxLength value="20"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="FDMS_Short_Type2">
        <xsd:annotation>
            <xsd:documentation>Choice between FDMS_String20_Type and
            FDMS_Null_Type_3
            </xsd:documentation>
        </xsd:annotation>
        <xsd:union memberTypes="FDMS_Null_Type_3 FDMS_String20_Type"/>
    </xsd:simpleType>
    <xsd:simpleType name="FDMS_DS_ID_Type">
        <xsd:annotation>
            <xsd:documentation>Equivalent to FDMS_Short_Type2. This type is
            intended for use with RDS_ID and SDS_ID elements.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="FDMS_Short_Type2"/>
    </xsd:simpleType>
```

```
<xsd:simpleType name="FDMS_Strict_Simple_Date_Type_1">
  <xsd:annotation>
    <xsd:documentation>
      specifies dd/mm/yy format. Does not allow dd>31, mm>12
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
 <xsd:pattern value="((0[1-9]|[1-2][0-9]|3[0-1])/(0[1-9]|1[0-2])/¥d{2})"/>
  </xsd:restriction>
... ...
</xsd:schema>
```

3.3.4. DSD (Document Structure Description)

3.3.4.1 Research Phase

3.3.4.1.1 Overview of the Language

DSD 1-3] allows the structure of a class of XML documents to be expressed in a context-sensitive manner. The language makes use of Boolean expressions and regular expressions. Some characteristics of DSD are:

- Context-sensitive definitions of element content, and a mechanism for expressing key/keyref relationships allow a schema written in DSD to convey some of the semantics of the class of XML documents being described, as well as their structure.
- DSD provides very usable regular expression syntax for building complex string data types.
- DSD allows a high level of modularity and reusability.
- DSD provides a sophisticated mechanism for the insertion of defaults into an XML instance document.

3.3.4.1.2 Selected Language Features

- Elements are defined using <ElementDef>. The following example defines a <title> element, which contains a character string.

Example 3.3.4.1.2.1

```
<ElementDef ID="title">
        <StringType/>
</ElementDef>
```

- Attributes are defined inside of element definitions, using <AttributeDecl>. The following example is a definition of <book> element that has a *price* attribute and contains a <title> element.

Example 3.3.4.1.2.2

```
<ElementDef ID="book">
        <AttributeDecl Name="price">
            <StringType/>
        </AttributeDecl>
        <Element Name="title">
            <StringType/>
        </Element>
</ElementDef>
```

- Element definitions can be reused. The following example provides the same definition as Example 3.3.4.1.2.2 above. However, instead of defining the <title> element in-place, a reference is made to the definition of <title> given in Example 3.3.4.1.2.1.

Example 3.3.4.1.2.3

```
<ElementDef ID="book">
        <AttributeDecl Name="price">
            <StringType/>
        </AttributeDecl>
        <Element IDRef="title"/>
```

</ElementDef>

- The following constructs allow a more complex content model to be specified:
  - <ZeroOrMore>, <OneOrMore> and <Optional> are used to specify occurrences of elements, corresponding to the *, + and ? symbols used in DTDs.
  - <Union> is used to specify a choice between element definitions.
  - An attribute can be declared as optional by including the *optional* attribute in the <AttributeDecl> element.
  - <Sequence> is used to specify that a sequence of elements. Unlike DTDs, the default content model is for elements to be unordered.

The following example defines a <book> element with the following characteristics:

- The *price* attribute is optional.
- A <book> contains a sequence of the following elements:
  - One or more <author> elements.
  - A choice between a <title> element and a <description> element.

Example 3.3.4.1.2.4

```
<ElementDef ID="book">
        <AttributeDecl Name="price" Optional="yes">
            <StringType/>
```

```
        </AttributeDecl>
        <Sequence>
            <OneOrMore>
                <Element IDRef="author"/>
            </OneOrMore>
            <Union>
                <Element IDRef="title"/>
                <Element IDRef="description"/>
            </Union>
        </Sequence>
    </ElementDef>
```

- The content of an element can be specified as being dependent on the existence and/or value of a particular attribute of the element. This involves the use of Boolean expressions that are constructed in DSD using a set of constructs that includes <not>, <if> and <then>. The following example gives a fragment of a definition of a <book> element where the <illustrator> element can only appear if the book is illustrated, which is indicated by the <book> element having an *illustrated* attribute with value "true".

  Example 3.3.4.1.2.5

```
<ElementDef ID="book">
    <AttributeDecl Name="illustrated">
        <StringType/>
    </AttributeDecl>
    <If>
        <Attribute Name="illustrated" value="true"/>
        <Then>
            <Element IDRef="illustrator"/>
        </Then>
    </If>
</ElementDef>
```

- The content of an element can be specified as being dependent on the context of the element. <context> is used to specify a path from the root to the current element. In Example 3.3.4.1.2.6, the <book> element can only have a <condition> element if the path specified by <context> was followed to get to the <book> element. The following XML instance document fragment would then be valid:

```
<!—any element ... -->
<bookstore used="true">
```

```
        <shelf>
            <book condition="new">
                <!—more elements... -->
            </book>
        </shelf>
    </bookstore>
```

Example 3.3.4.1.2.6

```
<ElementDef ID="book">
    <If>
        <Context>
            <SomeElements/>
            <Element Name="bookstore">
                <Attribute Name="used"/>
            </Element>
            <Element Name="Shelf"/>
        </Context>
        <Then>
            <Element IDRef="condition"/>
            <!—condition of the book, eg. new, worn... -- >
```

```
            </Then>
        </If>
        <!—other element definitions... -->
    </ElementDef>
```

- An extension of the ID/IDRef mechanism found in DTD is provided. An attribute of an element can be declared to be of type "ID", indicating that the value of the attribute must be unique throughout the XML instance document, and may be referred to by an attribute of a different element. An attribute of type "IDRef" has a value that references an attribute of type "ID", of a specified element. In this manner, relationships similar to the primary key – foreign key relationships of relational databases can be created.

  Example 3.3.4.1.2.7 Part 1 is a definition of an <author> element that has an *author_id* attribute, declared to be of type "ID". The <book> element in Example 3.3.4.1.2.7 Part 2 has an *author_reference* attribute, declared to be a reference to the *author_id* attribute of an <author> element. Thus the following XML instance fragment would be allowed:

  ```
  <author author_id="12345">William Shakespeare</author>
  <book author_reference="12345">Romeo and Juliet</book>
  ```

  Example 3.3.4.1.2.7 Part 1

  ```
  <ElementDef ID="author">
  ```

```
        <AttributeDecl Name="author_id" IDType="ID"/>
        <StringType/>
    </ElementDef>
```

Example 3.3.4.1.2.7 Part 2

```
<ElementDef ID="book>
        <AttributeDecl Name="author_reference" IDType="IDRef">
            <PointsTo>
                <Context>
                    <Element IDRef="author"/>
                </Context>
            </PointsTo>
        </AttributeDecl>
    </ElementDef>
```

- DSD has a sophisticated mechanism for inserting default attributes and element or data content into the XML instance document. Definitions can be created in a schema to specify default element content or attribute values that are created if a particular condition is satisfied. If created, a default triggers only in situations where a default insertion is appropriate, for example if a required attribute is not present. Such definitions of defaults can also be placed

inside the XML instance document, using the same DSD syntax. Defaults cascade, that is, definitions of defaults in an XML instance document can override those in the schema, and, within the XML instance document itself, definitions of defaults that are contained in more deeply nested elements can override those in less deeply nested (outer) elements.

- DSD provides a powerful regular expression syntax for creating definitions of string types. There are no mechanisms for defining numerical types, and there are no built-in data types (other than a string), as in W3C XML Schema. However, most data types can be viewed as a character string with format restrictions – for example, an integer can be viewed as a character string where each character must represent a digit. Therefore many non-string data types can be simulated using this method.

  In Example 3.3.4.1.2.8 Part 1, the *price* attribute is defined as having a value that is of type "price_type". The definition of price_type is given in Example 3.3.4.1.2.8 Part 2. The definition is built in a modular fashion, as a sequence of two subtypes – price_currency_type and price_number_type. Price_currency_type is defined as a choice between the enumerated values F, $ and DM. Price_number_type is defined as a sequence of two digits. Thus a valid *price* attribute value would be $45.

  Example 3.3.4.1.2.8 Part 1

  <AttributeDecl Name="price">

```
            <StringType IDRef=”price_type”/>
</AttributeDecl>
```

Example 3.3.4.1.2.8 Part 2

```
<StringTypeDef ID="price_currency_type”>
            <Union>
                        <String Value="F"/>
                        <String Value="$"/>
                        <String Value="DM"/>
            </Union>
</StringTypeDef>
<StringTypeDef ID="price_number_type”>
    <Sequence>
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="price_type">
    <Sequence>
```

```
        <StringType IDRef="price_currency_type"/>
        <StringType IDRef="price_number_type"/>
    </Sequence>
</StringTypeDef>
```

- A number of features allow a schema to be built in a modular fashion, from reusable components:
  - There is an *include* feature that allows the content of a DSD schema to be imported into another DSD schema, with each of the schemas residing in a different file. Commonly used definitions can then be grouped into one DSD schema, and reused by other schemas as required.
  - Definitions that have been imported into a DSD schema can be redefined, allowing the customization of commonly used definitions for use in a particular schema.
- As can be seen from the examples in this section, element definitions and string type definitions can be created as reusable definitions, which can then be referred to from other definitions. The same mechanism is in fact available for other types of DSD definitions such as definitions of attributes, defaults and contexts.

3.3.4.1.3 Tools and Resources

Validators

A validator, the DSD Processor, is available at the DSD home page [DSD2]. The DSD Processor is a command line Windows executable. The DSD Processor will either validate an XML instance document against a DSD schema or verify that a DSD schema conforms to the DSD language specification, depending on the command line arguments.

The DSD home page describes the DSD Processor as a prototype. As such, the DSD Processor would probably require some more refinement before being used for the FDMS DIF.

DTD / W3C XML Schema Specifications

Neither is available. However, a specification of the DSD language written in the DSD language is available. When writing a schema in DSD, the DSD Processor (described above) can be used to ensure that the schema conforms to the rules of the DSD language.

IDEs

No IDEs were found that support DSD.

Tools for Graphical Representation of Schemas

The DSD home page [DSD2] provides an XSLT style sheet, dsd2html.xsl, that can be used to convert a DSD schema into a readable, color-coded HTML representation. An XSLT

processor is required in order to perform the transformation.

Resources for Learning the Language

The DSD home page [DSD2] provides a tutorial, a white paper and a language specification. These resources are sufficient for learning the language. However, there are relatively few examples – not all language features are illustrated in the examples available – making the preparation of the FDMS schemas a little more difficult than for some of the other languages.

3.3.4.2 Practical Application Phase

Creating the Schemas

The schemas were developed using the XML Spy Integrated Development Environment (IDE). As no DTD or W3C XML Schema specification was available for DSD, it was not possible to validate the schema within XML Spy, to check for conformance to the DSD language. This validation was performed outside of the XML Spy IDE, using the command line DSD Processor tool. The following steps were taken to produce each of the schemas:

- A DSD schema was created from the original FDMS DTD. This was done manually, as there did not seem to be any tools available for automatic conversion of a DTD to a DSD schema.

- The data type requirements listed in section 3.2.3 were implemented by creating string type definitions (using the <StringTypeDef> DSD element) for each of the data type requirements.
- The data type of the Point_Of_Contact_Uniform_Resource_Locator element was specified using an existing data type specification – URI - that is available at the DSD homepage. (This is not a data type that is built in to the DSD language – it is a specification written using the language.) This involved including the uri.dsd file, which contains the definition of the URI data type, and then referring to the URI type from the definition of the Point_Of_Contact_Uniform_Resource_Locator element.
- An attempt was made to implement the required complex constraints (see section 3.2.3).

Results

The schemas are longer and more verbose than the original FDMS DTD specifications.

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats required were implemented successfully. However, as with the W3C XML Schema implementation, validation of the day value against the month and year values could not be achieved in a reasonable manner, because there is no mechanism for incorporating numerical analysis into a data type specification. Writing the specification for the FDMS_Date_Type was easier with DSD, and the result more readable, than with W3C XML

Schema. This was due to the regular expression syntax provided by DSD being very easy to work with. However, the other data types were harder to write in DSD because DSD does not provide any built-in data types – common data types such as a string of length 20 had to be built from scratch.

However, DSD makes it easy to reuse existing data type specifications. Using this capability, the URI data specification was downloaded from the DSD home page and imported into the schema.

Only one of the required constraints – Constraint 3.2.3.3 - could be expressed. The DSD extension of the DTD ID/IDREF mechanism applies only to attributes and thus was not of use in implementing Constraints 3.2.3.1 and 3.2.3.2. Context sensitivity works differently in DSD to RELAX and TREX. Using Boolean expressions within a context sensitive definition, it is possible to specify content that must not appear in a particular context – something that could not be achieved with RELAX and TREX. This allowed Constraint 3.2.3.3. to be implemented successfully.

3.3.4.3 Conclusion Phase

- DSD provides a lot of features and is thus not as easy to learn as RELAX and TREX. As was the case with W3C XML Schema, not all these features were of use for the FDMS DIF. Of particular use was the regular expression syntax for building string types, provided by

DSD. DSD also allows context-sensitive definitions whereby the definition of an element can depend on where in the instance document that element occurs. However, this capability was not applicable to the specific requirements of the FDMS DIF.

- Tool support was sufficient for development in the language – a validator and resources for learning the language were available. However, development was difficult relative to the other languages evaluated, because of the lack of tools to convert a DTD to a DSD, and because verification of the correctness of a DSD schema cannot be accomplished from within the XML Spy IDE.
- All data type requirements were satisfied, except the FDMS_Date_Type, which was implemented partially. One complex constraint, Constraint 3.2.3.3, was implemented successfully.
- Overall, DSD can provide an improvement over DTD for the purposes of the FDMS DIF.

3.3.4.4 Fragment of the Point Of Contact Schema in DSD

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?dsd URI="dsd.dsd"?>
<DSD IDRef="FDMS_METADATA_REP_POC_DIF" DSDVersion="1.0">
  <Title> A DSD description of the syntax of the FDMS POC DIF </Title>
  <Version>1.0 </Version>
```

```
<Author> Aidan </Author>
<Doc>
Have a look at the date data type specification
</Doc>
<ElementDef ID="FDMS_METADATA_REP_POC_DIF">
   <AttributeDecl Name="Version">
      <StringType IDRef="Version_Number"/>
   </AttributeDecl>
   <OneOrMore>
      <Union>
         <Element IDRef="Point_Of_Contact_Addition"/>
         <Element IDRef="Point_Of_Contact_Replacement"/>
      </Union>
   </OneOrMore>
</ElementDef>
<ElementDef ID="Point_Of_Contact_Addition">
   <Sequence>
      <Element IDRef="Point_Of_Contact_SDS_ID"/>
      <Element IDRef="Point_Of_Contact_Data"/>
```

```
    </Sequence>
</ElementDef>
<ElementDef ID="Point_Of_Contact_SDS_ID">
    <StringType IDRef="DS_ID"/>
</ElementDef>
<ElementDef ID="Point_Of_Contact_Data">
    <Sequence>
        <Element IDRef="Point_Of_Contact_Stereotype"/>
                ... ... ...

        <Optional>
            <Element IDRef="Point_Of_Contact_Employer_Start_Date"/>
        </Optional>
        <Optional>
            <Element IDRef="Point_Of_Contact_Employer_End_Date"/>
        </Optional>
                ... ... ...

    </Sequence>
```

```
</ElementDef>

<ElementDef ID="Point_Of_Contact_Stereotype">

    <StringType IDRef="Points_Of_Contact"/>

</ElementDef>

        … … …



<ElementDef ID="Point_Of_Contact_Employer_Start_Date">

    <StringType IDRef="FDMS_Strict_Date_Type"/>

</ElementDef>

<ElementDef ID="Point_Of_Contact_Employer_End_Date">

    <StringType IDRef="FDMS_Strict_Date_Type"/>

</ElementDef>

    … …



<StringTypeDef ID="DS_ID">

    <Union>

        <Sequence>

            <Repeat Value="20">

                <Optional>
```

```
                <AnyChar/>
            </Optional>
        </Repeat>
    </Sequence>
    <String Value="N/A"/>
  </Union>
</StringTypeDef>
<StringTypeDef ID="Points_Of_Contact">
  <Union>
    <String Value="ORG"/>
    <String Value="PER"/>
    <String Value="POS"/>
  </Union>
</StringTypeDef>

... ... ...

<StringTypeDef ID="FDMS_Lax_date_dd">
  <Sequence>
```

```
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_mm">
    <Sequence>
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_mmm">
    <Sequence>
        <Repeat Value="3">
            <Union>
                <CharRange Start="a" End="z"/>
                <CharRange Start="A" End="Z"/>
            </Union>
```

```
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_yy">
    <Sequence>
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_yyyy">
    <Sequence>
        <Repeat Value="4">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_1">
    <Sequence>
```

```
        <StringType IDRef="FDMS_Lax_date_dd"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_mm"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_yy"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_2">
    <Sequence>
        <StringType IDRef="FDMS_Lax_date_dd"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_mmm"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_yy"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_3">
    <Sequence>
        <StringType IDRef="FDMS_Lax_date_dd"/>
```

```
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_mm"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_yyyy"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_4">
    <Sequence>
        <StringType IDRef="FDMS_Lax_date_dd"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_mmm"/>
        <String Value="/"/>
        <StringType IDRef="FDMS_Lax_date_yyyy"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_5">
    <Sequence>
        <StringType IDRef="FDMS_Lax_date_dd"/>
        <String Value=" "/>
```

```
        <StringType IDRef="FDMS_Lax_date_mmm"/>
        <String Value=" "/>
        <StringType IDRef="FDMS_Lax_date_yyyy"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date">
    <Union>
        <StringType IDRef="FDMS_Lax_date_1"/>
        <StringType IDRef="FDMS_Lax_date_2"/>
        <StringType IDRef="FDMS_Lax_date_3"/>
        <StringType IDRef="FDMS_Lax_date_4"/>
        <StringType IDRef="FDMS_Lax_date_5"/>
    </Union>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_time_minsec">
<Sequence>
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
```

```
        </Sequence>
    </StringTypeDef>
    <StringTypeDef ID="FDMS_Lax_time_hour">
    <Sequence>
        <Repeat Value="2">
            <CharRange Start="0" End="9"/>
        </Repeat>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_time">
    <Sequence>
        <StringType IDRef="FDMS_Lax_time_hour"/>
        <String Value=":"/>
        <StringType IDRef="FDMS_Lax_time_minsec"/>
        <String Value=":"/>
        <StringType IDRef="FDMS_Lax_time_minsec"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_time">
```

```
    <Sequence>
        <StringType IDRef="FDMS_Lax_date"/>
        <String Value=" "/>
        <StringType IDRef="FDMS_Lax_time"/>
    </Sequence>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_date_null">
    <Union>
        <String Value="N/A"/>
        <String Value="TBD"/>
        <String Value="UNK"/>
    </Union>
</StringTypeDef>
<StringTypeDef ID="FDMS_Lax_Date_Type">
    <Union>
        <StringType IDRef="FDMS_Lax_date"/>
        <StringType IDRef="FDMS_Lax_time"/>
        <StringType IDRef="FDMS_Lax_date_time"/>
        <StringType IDRef="FDMS_Lax_date_null"/>
```

```
    </Union>
</StringTypeDef>
</DSD>
```

3.3.5. Schematron

3.3.5.1 Research Phase

3.3.5.1.1 Overview of the Language

While most schema languages provide a means to specify a grammar that describes a valid class of XML instance documents, Schematron [SCHTR1-6] uses a different approach to validation – a rule-based approach. A schema written in Schematron consists of a collection of rules. Each rule is evaluated at a specified point in the XML instance document, in order to test whether a particular aspect of the instance document is valid.

3.3.5.1.2 Selected Language Features

- The <rule> is the central construct in Schematron. A <rule>:
  - Has an attribute, *context*, which specifies the set of nodes in the XML document tree to which the rule is to be applied. Often, the value of *context* will be the name of a particular element.
  - Contains a combination of <assert> and <report> elements. Both of these elements

have an attribute, *test*, which contains an expression to be evaluated at the context node. The content of the elements specifies a feedback message to provide to the user.

- ✧ The purpose of an <assert> element is to provide feedback if the expression evaluates to false
- ✧ A <report> element, on the other hand, provides feedback if the expression evaluates to true.

The rule in the following example expresses that a <book> element must contain an <author> element – an error message will result otherwise. The rule also expresses that positive feedback will be provided if the <book> element contains a <price> element. However, the absence of a *price* element will not result in an error message.

Example 3.3.5.1.2.1

```
<rule context="book">
        <assert test="author">A book element must contain an author
                element.</assert>
    <report test="price">A price element was found.</report>
</rule>
```

This constraint can be expressed in any of the six schema languages evaluated.

- The expressions contained in the *context* and *test* attributes must be written using the XPath

language [XPATH1-2]. XPath is a standard XML technology and is supported by many XML tools, such as XML Spy, which was used in the research for this paper. XPath allows a set of nodes in an XML document to be specified. The following are some examples of XPath expressions:

- Book A <Book> element
- Author | Price An <Author> or <Price> element
- Book/* Any child of a <Book> element
- Author::following-sibling The following sibling of an Author element

A variety of functions and operators are available, which enable arithmetic, Boolean and string operations to be performed on the content of an element or attribute.

The following examples illustrate the kinds of constraints that can be expressed by making fuller use of the power of XPath than was done in Example 3.3.5.1.2.1.

The rule in Example 3.3.5.1.2.2 expresses that the number of <author> elements contained inside a <book> element must be less than or equal to 2 (<= is the symbol for the less-than-or-equal-to operator).

Example 3.3.5.1.2.2

```
<rule context="book">
        <assert test="count(author)&lt;=2">
```

A book cannot have more than 2 authors

</assert>

</rule>

This constraint can also be expressed (efficiently) using DSD or W3C XML Schema.

The rule in Example 3.3.5.1.2.3 specifies that the <price> element must contain a numerical value.

Example 3.3.5.1.2.3

<rule context="price">

<assert test="number(price)=floor(price)">

The price of a book must be a numerical value

</assert>

</rule>

This constraint can be expressed in any of the six other languages evaluated, but not in a DTD.

Example 3.3.5.1.2.4 specifies that the content of a <price> element must be a currency symbol, chosen from $, F or D, immediately followed by a number. So both of the following would be valid:

<price>$20.99</price>

<price>F3500</price>

Example 3.3.5.1.2.4

<rule context="price">

<assert test="starts-with(.,'$') | starts-with(.,'F') |

starts-with(.,'D')">

The price must start with one of the following currency

symbols: $, F, D

</assert>

<assert test="number(substring(.,2))">

The price must consist of a currency symbol followed by a

number

</assert>

</rule>

Note that the symbol "." refers to the current node.

This data type constraint could be implemented in W3C XML Schema (and therefore RELAX, TREX and RELAX NG, because all three languages use the W3C XML Schema data typing system). However, it would have to be implemented using a regular expression,

and would be more difficult to write because no arithmetic functions are available. The constraint could also be implemented in DSD, once again using regular expressions.

The rule in Example 3.3.5.1.2.5 specifies that a <price> element must either have a *currency* attribute, or be followed by a <currency> element. So either of the following would be valid:

```
<price currency="us_dollar">25</price>
```

or

```
<price>25</price>
<currency>us_dollar</currency>
```

Example 3.3.5.1.2.5

```
<rule context="price">
        <assert test="@currency | following-sibling::currency">
                A price element must either have a currency attribute or be
                followed by a currency element
        </assert>
</rule>
```

This constraint can also be expressed using TREX, DSD or RELAX

- Schematron provides two other constructs: <schema> and <pattern>. A <pattern> is used to contain a collection of <rule>s. A <schema> contains a collection of <pattern>s.

- Schematron provides the following constructs for increasing the modularity of a schema:
  - <Pattern>s can be grouped into <phase>s within a single schema. Each <phase> represents one use of the schema, or one phase in the development of the instance document it describes. For example, the book schema could have two phases – one to use for books that are still in the process of being written, and another to use for books that are on sale.
  - A <rule> can be defined to be *abstract.* Such a rule does not have a *context* and therefore cannot be used on its own. An abstract rule is a reusable definition of a rule that can be referred to from other rules. This provides a good structure in which to define data type constraints. An abstract rule could be written to define the format of an email address, for example, and this rule could then be referenced in the definition of every element that contains an email address.

- Schematron provides the means to generate human readable feedback, on validation of a schema, via the following:

- The content of <assert> and <report> elements, as has already been illustrated in the examples in this section.
- The <diagnostics> element, which allows a detailed feedback message to be created and then reused by any number of <assert> and <report> elements.

The readability and usefulness of the feedback ultimately depends on how the data contained in the <assert>, <report> and <diagnostics> elements is used by the actual implementation of the Schematron language that is being used. This will be discussed in the next section.

3.3.5.1.3 Tools and Resources

Validators

An implementation of a Schematron validator is available at the Schematron home page [SCHTR5]. The validator is written in XSLT. XSLT [XSLT1-2] is a language used to transform XML documents according to a set of instructions specified in an XSL style sheet. An XSLT processor is required in order to use the validator. XSLT processors are commonly included in XML tools, such as the XML Spy Integrated Development Environment - which was used to develop the schemas for this paper.

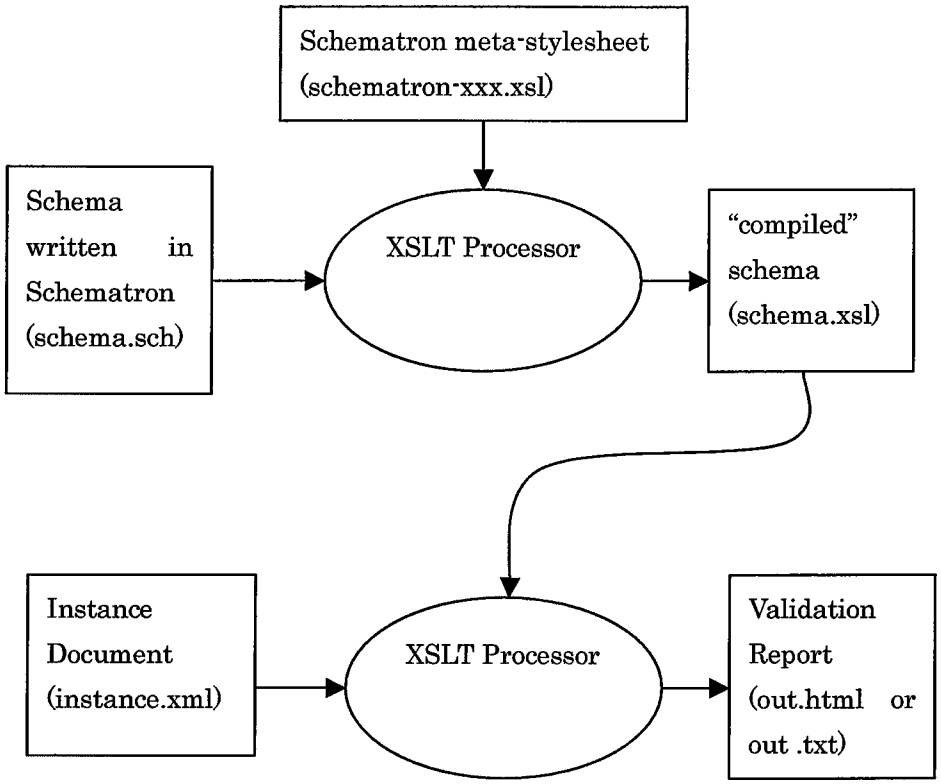
The XSLT implementation of Schematron consists of an XSL style sheet, skeleton1-5.xsl, and three meta-style sheets, schematron-basic.xsl, schematron-report.xsl, and schematron-html.xsl. skeleton1-5.xsl provides the essential functionality of the Schematron

language, and this file is imported into each of the meta-style sheets. The three meta-style sheets are used to provide three different styles of output to the user. Schematron-basic.xsl was used in the production of the schemas in the Practical Application Phase. The following steps are taken to validate an instance document against a Schematron schema:

- One of the Schematron meta-style sheets is applied to the Schematron schema in an XSLT transformation. The result of the transformation is another XSL style sheet. This can be thought of as a "compiled" version of the Schematron schema.
- The "compiled" schema is applied to the instance document in an XSLT transformation. The result is a list of feedback messages for the user, indicating to what extent the instance document complied with the schema.

The following diagram illustrates the process more clearly:

Figure 1 - Validation Using Schematron



Using the XML Spy Integrated Development Environment (IDE), the following sequence of actions was followed in order to perform validation against the schemas written in the Practical Application Phase:

- Create a project that includes the instance document, the Schematron schema, and the two XSL style sheets that comprise the implementation of the language - skeleton1-5.xsl and schematron-basic.xsl.
- Assign the schematron-basic.xsl style sheet to the Schematron schema, and then use the XSL

Transformation feature to transform the schema.

- Save the result of the transformation as CompiledSchema.xsl
- Assign the CompiledSchema.xsl style sheet to the instance document and then use the XSL Transformation feature to transform the instance document.
- The result of this second transformation is a file containing a list of feedback messages to the user, indicating the degree to which the instance document conforms to the schema.

A major advantage of using Schematron is that it enables the user to be provided with very readable and detailed feedback, explaining exactly where in the instance document an error has occurred, and what need to be done to correct the error. The quality of the presentation of the feedback depends on the meta-stylesheet used. Using schematron-basic.xsl, the output consists of a sequence of feedback messages in plain text, with each message on its own line. However, by using the schematron-report.xsl meta-style sheet instead, the output is generated as an HTML page, divided into two frames. The feedback messages are displayed in one frame. Clicking on a feedback message displays the relevant section of the instance document in the other frame.

In addition to the XSLT validator, there are other validators for Schematron written in Java, Python and Perl.

DTD / W3C XML Schema Specifications

DTD and W3C XML Schema specifications for the Schematron language are available at the Schematron home page [SCHTR5].

IDEs

Schematron Validator is a free tool that can be obtained at http://www.topologi.com/. The tool supports W3C XML Schema, DTD and Schematron. Schematron Validator automates the process of validating an instance document against a Schematron schema, and generates output in a style similar to that provided by the schematron-report.xsl implementation. The tool facilitates customization of the styling of the output. Schematron Validator also has the functionality to validate against a W3C XML Schema that contains embedded Schematron rules.

Tools for Graphical Representation of Schemas

Schematron Validator, described above, does not provide a graphical representation of a schema. There do not seem to be any tools available that provide this facility. However, a graphical representation of a schema is not as important for Schematron as it is for other languages, where a schema can become very complicated, with multiple nested layers. A Schematron schema, on the other hand, just consists of groupings of rules, and does not have a very complicated structure.

Resources for Learning the Language

There are several resources online for learning Schematron, e.g. [SCHTR1], [SCHTR3]. There do not seem to be any resources in print that are concerned exclusively with the Schematron language. However, an introduction to the language is provided in [WROX].

Schematron is built on the XSLT and XPath languages. A large number of resources, both online and in print, are available for each of these languages.

3.3.5.2 Practical Application Phase

Creating the Schemas

The schemas were created using the XML Spy Integrated Development Environment (IDE). Although the XML Spy IDE does not support the Schematron language, it does provide many features for creating and editing XML documents in general. These features are useful for Schematron schemas also because Schematron is itself an XML language (as are the other languages evaluated). The following steps were followed to create each Schematron schema:

- The schema was written from scratch – no tools are available to convert the original DTD to a Schematron schema.
- Correctness of the Schematron schema was verified by using the Validate File feature in the XML Spy IDE. In order for this to work, the DTD specification for the schematron

language (available at the Schematron homepage [SCHTR5]) must be assigned to the Schematron schema.

Results

The Schematron schema for the Point of Contact DIF contains the same level of information as the original FDMS DTD, plus data type specifications. However, the code required to achieve this is complex and hard to read. In general, Schematron is not well suited to replicating the types of definitions found in a DTD. For the Model DIF, instead of attempting to replicate the functionality of the original FDMS DTD, a top-up Schematron schema was developed – containing only those specifications that a DTD cannot handle.

All the Model constraints and data type requirements for both the Point of Contact and Model sections were successfully specified in Schematron. However, the code is not very readable.

3.3.5.3 Conclusion Phase

- Although Schematron itself is simple and easy to learn, the schema author must also be proficient in XPath, which is much more difficult. Although Schematron is more powerful than the other languages evaluated, it is not easy to read or write. Schematron is best used in combination with another schema language, in order to provide only the areas of validation

that the other schema language cannot handle. It is possible to use Schematron in this capacity because unlike the other languages evaluated, a Schematron schema does not have to specify every attribute or element that may appear in the instance document.

- Schematron has good tool support and a large number of resources are available for learning the language. The main Schematron validator consists of an XSLT stylesheet. Validation against a Schematron schema consists of a sequence of XSLT transformations which can be performed manually within the XML Spy IDE.
- All the Model constraints and data type requirements were successfully specified in Schematron. However, the code is not very readable.
- Overall, Schematron would not be a good replacement for DTD, for the purposes of the FDMS DIF – the schema would be too difficult to write and would provide very little visualization. However, it would be beneficial to use Schematron in combination with DTD, or one of the other schema languages.

3.3.5.4 Fragment of the Point Of Contact Schema in Schematron

```
<?xml version="1.0" encoding="UTF-8"?>
<schema ns="poc_sch.xml" xmlns="http://www.ascc.net/xml/schematron">
<title>Schema for POC</title>
```

```
<pattern name="POC, Root Test" id="POCroot">
    <rule context="/*">
        <assert test="name()='FDMS_METADATA_REP_POC_DIF'">
            The root element must be <name/>
        </assert>
    </rule>
</pattern>

<pattern name="POC, Document Structure" id="POCstr">
    <rule context="FDMS_METADATA_REP_POC_DIF">
        <assert test="count(*) = count(Point_Of_Contact_Addition) +
        count(Point_Of_Contact_Replacement)">
            A FDMS_METADATA_REP_POC_DIF element can only contain
            Point_Of_Contact_Addition and Point_Of_Contact_Replacement elements
        </assert>
            ... ...

    </rule>
            ... ...
```

```
</pattern>


<pattern name="POC, Element Ordering" id="POCord">
<!-- Specifies the order in which elements can appear in the document. -->
    <rule context="Point_Of_Contact_Addition/Point_Of_Contact_SDS_ID">
      <assert test="following-sibling::Point_Of_Contact_Data">
          Within the Point_Of_Contact_Addition element, the
          Point_Of_Contact_SDS_ID element must come before the
          Point_Of_Contact_Data element.
      </assert>
    </rule>
      ... ...


</pattern>


<pattern name="POC, Element Content" id="POCecon">
 <!-- The following are reusable (abstract) rules which validate data format/type -->
    <rule abstract="true" id="POC_Short_String_1">
      <assert test="string-length(.) &lt;= 20">
```

```
        Content of element <name/> must not exceed 20 characters.
    </assert>
</rule>
<rule abstract="true" id="POC_Long_String">
    <assert test="string-length(.) &lt;= 80 or normalize-space(.) = 'N/A' or
    normalize-space(.) = 'TBD'">
        Content of element <name/> must not exceed 80 characters.
    </assert>
</rule>

<rule abstract="true" id="POC_Date">
<!-- Tests that the date is of the format dd/mm/yy -->
    <assert test="string-length(.) = 8">
        Content of element <name/>Date must have format dd/mm/yy
    </assert>
    <assert test="contains(.,'/') and contains(substring-after(.,'/'), '/')">
        Date must have format dd/mm/yy
    </assert>
    <assert test="string-length(substring-before(.,'/')) = 2">
```

```
            The day must have the form dd
        </assert>
        <assert test="string-length(substring-before(substring-after(.,'/'),'/')) = 2">
            The month must have the form mm
        </assert>
        <assert test="string-length(substring-after(substring-after(.,'/'),'/')) = 2">
            The year must have the form yy
        </assert>
    </rule>

        <!-- The following rules validate the data type/format of the values contained
by each element. Some rules use the absract rules defined above. -->
    <rule context="Point_Of_Contact_SDS_ID">
        <extends rule="POC_Short_String_1"/>
    </rule>
    <rule context="Point_Of_Contact_Position">
        <extends rule="POC_Long_String"/>
    </rule>
    <rule context="Point_Of_Contact_Prefix">
```

```
        <extends rule="POC_Long_String"/>
    </rule>
            ... ...

    <rule context="Point_Of_Contact_Employer_Start_Date">
        <extends rule="POC_Date"/>
    </rule>
    <rule context="Point_Of_Contact_Employer_End_Date">
        <extends rule="POC_Date"/>
    </rule>
        ... ...
  </pattern>

  ... ...

</schema>
```

3.3.6. Examplotron

3.3.6.1 Research Phase

3.3.6.1.1 Overview of the Language

Examplotron [EX1-2] allows a schema to be built from examples - sample XML instance documents. The simplest possible Examplotron schema is a single XML instance document. Any other instance document that matches the structure of this instance document is considered to conform to the schema. The expressive power of the schema can be increased by using a combination of sample XML instance documents to create a schema, and by annotating the examples with Examplotron code that can further constrain or generalize the structure Schematron-style rules can be used to annotate the examples, providing a lot of expressive power.

3.3.6.1.2 Selected Language Features

- The simplest schema is just an XML instance document. The following example shows an XML instance document consisting of a <book> element, which contains an <author> element followed by a <price> element.

  Example 3.3.6.1.2.1

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <book>
      <author>James Smith </author>
  ```

```
        <price>$10</price>
    </book>
```

The above XML instance document is itself a complete Examplotron schema. Used as such, the schema will validate any XML instance document that consists of a book element that contains one author element followed by one price element. However, the schema cannot be used to enforce, for example, that the <author> element must contain a character string. Data type constraints are communicated to the human reader of the schema via the sample data contained in the elements in the schema. However, the current implementation of Examplotron does not discriminate between data types. Therefore, the data types of the content of the elements in the schema are not used when validating an XML instance document.

- Occurrences of elements can be specified, by inserting an *occurs* attribute into the appropriate element. The *occurs* attribute can take any of the following values:
  - "?": Optional
  - "* ": Zero or more
  - "+": One or more

In the following example, the schema in Example 3.3.6.1.2.1 is modified in order to specify that a <book> element has one or more <author> elements, and that the <price> element is

optional.

Example 3.3.6.1.2.2

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns:eg="http://examplotron.org/0/">
        <author eg:occurs="+">James Smith </author>
        <price eg:occurs="?">$10</price>
</book>
```

Note that the *occurs* attribute must be qualified by the Examplotron namespace prefix, *eg*. All elements and attributes that are part of the Examplotron language, rather than part of the examples used to make the schema, must be differentiated in this way.

- The *assert* attribute is used to specify a condition, expressed using XPath. The condition must evaluate to true in order for an instance document to be valid according to the schema. The *assert* attribute allows similar functionality to its counterpart in the Schematron language. As with Schematron, *assert* can be used to impose a restriction on the type of content that an element can contain, effectively specifying a data type.

  The following example adds a restriction to Example 3.3.6.1.2.2 – the <author> element must contain a string of up to 20 characters.

  Example 3.3.6.1.2.3

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns:eg="http://examplotron.org/0/">
    <author eg:occurs="+" eg:assert="string-length=20">
        James Smith
    </author>
    <price eg:occurs="?">$10</price>
</book>
```

Many other types of constraints can be expressed using the *assert* attribute. For other examples, refer to the section on Schematron; XPath expressions used in Schematron <rule> elements can be used in Examplotron *assert* attributes.

- The <import> element is used to import an entire schema from another file. The resultant schema is then the combination of the recipient schema and the imported schema. The resultant schema thus specifies that a valid instance document must conform to either the recipient schema or the imported schema.

  The schema in the following example specifies that an instance document can either consist of a <book> element that contains <author> and <price> elements (as specified in Example 3.3.6.1.2.3), or a <book> element that contains a <title> element.

  Example 3.3.6.1.2.4

```
<?xml version="1.0" encoding="UTF-8"?>
<book
xmlns:eg="http://examplotron.org/0/"
xmlns:xlink="http://www.w3.org/1999/xlink">
<eg:import xlink:href="Example3.3.6.1.2.3.xml"/>
        <title>
                Around the World in 80 Days
        </title>
</book>
```

3.3.6.1.3 Tools and Resources

Validators

An XSLT implementation of an Examplotron validator, compile.xsl, is available at the Examplotron home page [EX1]. This implementation works in the same way as the XSLT implementation of Schematron (described in Section 3.3.5). compile.xsl, is used to perform an XSLT transformation on a schema written in Examplotron, to produce a "compiled schema", which is also an XSLT style sheet. The "compiled schema" can then be used to perform an XSLT transformation on an XML instance document. The output from the transformation indicates whether the XML instance document conforms to the Examplotron schema.

Compile.xsl only works with one XSLT processor, however – the Saxon XSLT processor. In order to "compile" an Examplotron schema using XMLSpy, when creating the schemas in the Practical Application Phase, some minor changes were first made to the source code in compile.xsl.

DTD / W3C XML Schema Specifications

A W3C XML Schema specification for Examplotron is available at the Examplotron home page [EX1].

IDEs

No IDEs support Examplotron.

Tools for Graphical Representation of Schemas

No tools exist to graphically represent an Examplotron schema.

Resources for Learning the Language

The tutorial at Examplotron home page [EX1] provides all the available information about Examplotron.

3.3.6.2 Practical Application Phase

Creating the Schemas

The schemas were created using the XML Spy Integrated Development Environment (IDE). Although the XML Spy IDE does not support the Examplotron language, it does provide many features for creating and editing XML documents in general. These features are useful for Examplotron schemas also because Examplotron is itself an XML language (as are the other languages evaluated). The following steps were followed to create each Examplotron schema:

- First, a sample instance document was obtained from [FDMS2] or [FDMS3]. This was used as the starting point for the Examplotron schema.
- Next, *occurs* attributes were added to some of the elements in the schema, so as to specify "zero or more", "one or more" or "optional" occurrences.
- Any optional elements that were not present in the original sample instance document were added to the schema, and specified as optional, using the *occurs* attribute.
- Constraints were added to particular elements by using the *assert* attribute.

Results

The Point of Contact schema was successfully written. However, one major problem was encountered – there is no Examplotron language construct for specifying a choice between elements. For the Point of Contact schema, a choice between elements occurs only once

in the schema – the root element can have either a Point_Of_Contact_Addıton element or a Point_Of_Contact_Replacement element. In order to specify this choice, two Examplotron schemas were created, one for the class of documents that has a Point_Of_Contact_Addition element as a child of the root, and the other for the class of documents that has a Point_Of_Contact_Replacement element as a child of the root. The Examplotron <import> feature was then used to specify that an instance document must conform to either schema. Obviously, this is not a practical way of writing large complex schemas, such as the Model schema, where there are a large number of choices between elements. Therefore an Examplotron schema could not be written for the Model section of the FDMS DIF.

The required data types were written successfully, by using assert attributes. The expressions specified in the assert attributes were taken from the Schematron Point of Contact schema (see Section 3.3.5). Examplotron supports the same XPath constraints that Schematron supports. However, Examplotron does not provide the rule structure of Schematron, which means that each element can only be associated with one assert statement, and therefore with only one error message to the user.

3.3.6.3 Conclusion Phase

- Examplotron is a language that is still under development and is not ready for real-world use. The language in its present state allows only small, simple schemas to be built. This is

because some basic capabilities are missing – for instance, there is no way to easily specify choices between elements or groups of elements in an instance document.

- The Point Of Contact evaluation schema was successfully written in Examplotron and fulfilled all data type requirements. However, the evaluation schema for the Model DIF was too large to be expressed in Examplotron.
- Tools are available for the language - a validator and a tutorial are available at the Examplotron home page [EX1].
- Overall, Examplotron is not an improvement over DTD for the purposes of the FDMS DIF.

3.3.6.4 Fragment of the Point Of Contact Schema in Examplotron

```
<?xml version="1.0" encoding="UTF-8"?>
<FDMS_METADATA_REP_POC_DIF Version="0.0"
xmlns:eg="http://examplotron.org/0/">
    ... ...
<Point_Of_Contact_Addition eg:occurs="+">
    <Point_Of_Contact_SDS_ID>201</Point_Of_Contact_SDS_ID>
    <Point_Of_Contact_Data>
        <Point_Of_Contact_Stereotype>PER
            </Point_Of_Contact_Stereotype>
```

```
<Point_Of_Contact_Type eg:occurs="?">VIP
    </Point_Of_Contact_Type>
<Point_Of_Contact_Position eg:occurs="?">President
    </Point_Of_Contact_Position>
<Point_Of_Contact_Prefixeg:occurs="?">Mr.
    </Point_Of_Contact_Prefix>
<Point_Of_Contact_Last_Name>Truman
    </Point_Of_Contact_Last_Name>
<Point_Of_Contact_First_Name eg:occurs="?">Harry
    </Point_Of_Contact_First_Name>
<Point_Of_Contact_Middle_Name eg:occurs="?">S.
    </Point_Of_Contact_Middle_Name>
                ... ...
  <Point_Of_Contact_Employer_Start_Date eg:occurs="?"
eg:assert="string-length(.) = 8 and
      contains(.,'/') and
      contains(substring-after(.,'/'), '/'))"> 03/04/05
    </Point_Of_Contact_Employer_Start_Date>
```

... ...

</Point_Of_Contact_Data>

</Point_Of_Contact_Addition>

... ...

</FDMS_METADATA_REP_POC_DIF>

3.4. Results of Evaluations

None of the languages (alone) is capable of fulfilling completely the requirements of the FDMS DIF. An evaluation of each language is given below. More detailed evaluations, including fragments of some of the schemas written in the practical application phase are provided in Section 3.3.

3.4.1. TREX (Tree Regular Expressions for XML)

Out of the six languages evaluated, TREX was the easiest to learn and to use, due to a small set of orthogonal language constructs. TREX provides full support for using W3C XML Schema data types within a TREX schema. These were used in the development of the evaluation schemas in TREX. TREX allows element definitions to be context-sensitive - an

element can have more than one definition, allowing the definition used to depend on the context in which the element appears

When writing the evaluation schemas in TREX, tools for converting from other schema languages to TREX were used, to convert the original DTD to a TREX schema. The generated TREX schema was then modified manually. Tool support was sufficient for development in the language – validators and resources for learning the language were available. However, desirable tools - IDE, tools for graphically displaying a schema and DTD specification for the language - were not found.

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats were expressed efficiently but validating the day value against the month and year values proved to be almost impossible. (The only way would be to enumerate all possible combinations of day, month and year values for each possible format.) Only one of the required constraints – Constraint 3.2.3.6 - could be expressed. Overall, TREX can offer an improvement over DTD for the FDMS DIF.

3.4.2. RELAX (Regular Expression Language for XML)

RELAX offers similar features to TREX. Like TREX, RELAX is a relatively simple language and is easy to learn. As with TREX, an element can have more than one definition, allowing the definition used to depend on the context in which the element appears.

RELAX makes use of W3C XML Schema data types. However, not all the data type features are provided. In particular there are no user-defined data types and no mechanism for specifying the format of a string.

RELAX has good tool support. Tools for converting from other schema languages to RELAX were used, to convert the original DTD to a RELAX schema. The generated RELAX schema was then modified manually. Tool support was sufficient for development in the language – validators and resources for learning the language were available. In addition, some desirable tools – a tool for graphically displaying a schema and a DTD specification for the language - were available. However no IDE provides full support for RELAX.

All the required data types were specified successfully except for FDMS_Date_Type. String formats cannot be expressed within a data type, so the various date formats could not be specified. Only one of the required constraints – Constraint 3.2.3.6 - could be expressed. RELAX can provide an improvement over DTD for the purposes of the FDMS DIF.

3.4.3. W3C XML Schema

W3C XML Schema has a very rich set of features. However, this makes it a more difficult language to learn than TREX or RELAX. W3C XML Schema offers a large set of built-in data types and many features for creating user-defined data types. Some language features – in particular features for specifying uniqueness and primary key/foreign key

relationships – are very useful in general but were not applicable to the specific requirements of the FDMS DIF. The constraints in the FDMS DIF that involve uniqueness and primary key/foreign key relationships are too complex to be expressed using the features available in W3C XML Schema.

Excellent tools and resources are available. At least two Integrated Development Environments (IDEs) for XML support W3C XML Schema. The XML Spy IDE provides superior visualization of a W3C XML Schema using its Schema Design View mode. It also provides automated conversion of a DTD to a W3C XML Schema.

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats required were expressed successfully but validating the day value against the month and year values could not be achieved in a reasonable manner. (The only way would be to enumerate all possible combinations of day, month and year values for each possible format.) None of the required constraints could be expressed.

Overall, W3C XML Schema can provide an improvement over DTD for the purposes of the FDMS DIF – in particular for data type specifications. In addition, the superior tool support for W3C XML Schema makes it a very good choice of language for the FDMS DIF. However, as noted above, W3C XML Schema cannot adequately handle all the constraints of the FDMS DIF. A good solution would be to use W3C XML Schema in combination with another language that can handle these constraints, such as Schematron.

3.4.4. DSD (Document Structure Description)

DSD provides a lot of features and is thus not as easy to learn as RELAX and TREX. As was the case with W3C XML Schema, not all these features were of use for the FDMS DIF. Of particular use was the regular expression syntax for building string types, provided by DSD. DSD also allows context-sensitive definitions whereby the definition of an element can depend on where in the instance document that element occurs.

Tool support was sufficient for development in the language – a validator and resources for learning the language were available. The DSD home page [DSD2] provides a tool for graphically rendering a schema. No IDE provides support for DSD.

All the required data types were specified successfully except for FDMS_Date_Type. The various date formats required were implemented successfully. However, as with the W3C XML Schema implementation, validation of the day value against the month and year values could not be achieved in a reasonable manner. Writing the specification for the FDMS_Date_Type was easier with DSD, and the result more readable, than with W3C XML Schema. This was due to the regular expression syntax provided by DSD being very easy to work with. However, the other data types were harder to write in DSD because DSD does not provide any built-in data types – common data types such as a string of length 20 had to be built from scratch.

Only one of the required constraints – Constraint 3.2.3.3 - could be expressed. Overall, DSD can provide an improvement over DTD for the purposes of the FDMS DIF.

3.4.5. Schematron

A Schematron schema consists of a number of rules, grouped into patterns. A rule has a context and a number of assert or report statements. The context is the point in the instance document where the rule is applied. An assert statement consists of a condition and a message. If the condition evaluates to false, the message is output. A report statement is similar except that the message is output if the condition evaluates to true. The assert and report conditions, and the context, are specified using the XPath language, which provides a large number of functions for traversing a document and analyzing its content. Schematron is a very powerful schema language for expressing complex constraints.

Although Schematron itself is simple and easy to learn, the schema author must also be proficient in XPath, which is much more difficult. Although Schematron is more powerful than the other languages evaluated, it is not very readable. Schematron is best used in combination with another schema language, in order to provide only the areas of validation that the other schema language cannot handle. It is possible to use Schematron in this capacity because unlike the other languages evaluated, a Schematron schema does not have to specify every attribute or element that may appear in the instance document.

Schematron has good tool support and a large number of resources are available for learning the language. The main Schematron validator consists of an XSLT stylesheet. Validation against a Schematron schema consists of a sequence of XSLT transformations.

All the Model constraints and data type requirements were successfully specified in Schematron. However, the code is not very readable. Overall, Schematron would not be a good replacement for DTD, for the purposes of the FDMS DIF – the schema would be too difficult to write and would provide very little visualization. However, it would be beneficial to use Schematron in combination with DTD, or one of the other schema languages.

3.4.6. Examplotron

An Examplotron schema consists of a sample instance document with some additional attributes (prefixed with the eg: namespace identifier) inserted into the element tags in order to specify additional information about the instance document. For example, by adding the attribute eg:occurs="?" to an element, the element is specified as being optional. An advantage of using Examplotron is that the Examplotron schema looks a lot like the instance document, making the schema relatively easy to understand.

Examplotron also provides Schematron-style assert statements that can be used to evaluate an XPath expression at a particular element. Using these assert statements, some of the same constraints specified in the Schematron evaluation schemas were successfully specified in

the Examplotron schemas. However, Schematron is a more suitable language for specifying such constraints.

Examplotron is a language that is still under development and is not ready for real-world use. The language in its present state allows only small, simple schemas to be built. This is because some basic capabilities are missing – for instance, there is no way to easily specify choices between elements or groups of elements in an instance document.

Tools are available for the language - a validator and a tutorial are available at the Examplotron home page [EX1].

The Point Of Contact evaluation schema was successfully written in Examplotron and fulfilled all data type requirements. However, the evaluation schema for the Model DIF was too large to be expressed in Examplotron. Overall, Examplotron is not an improvement over DTD for the purposes of the FDMS DIF.

CHAPTER 4

COMBINING SCHEMA LANGUAGES

4.1. Introduction

This section is concerned with investigating how the schema languages evaluated in section 3 can be combined, and determining which combinations of schema languages are appropriate for the FDMS DIF.

The main reason to combine schema languages is that, as described in section 3.3, none of the languages evaluated is capable of fulfilling completely all the requirements of the FDMS DIF. However, by using two or more schema languages together, it may be possible to use features from both languages in order to fulfill the requirements more completely.

Some examples of possible (though not necessarily practical) combinations are:

- W3C XML Schema with DSD – W3C XML Schema could be used for general structure and data types, while DSD could provide context sensitive definitions.
- Any language with Schematron – Schematron could be used to specify complex constraints not handled by the other language

- Any language with Examplotron – using Examplotron for visualization of the schema.

4.2. Ways of Combining Schema Languages

4.2.1. Two Groups of Languages

For the purposes of this investigation, we can divide schema languages into two groups

- Type 1: Languages that are used to create schemas that define completely the structure of the instance document. For example, if a particular element is not specified in the schema then it cannot appear in the instance document.
- Type 2: Languages that are used to create schemas that are not necessarily complete. That is, the schema does not have to define everything that might appear in the instance document.

Schematron falls into the second group because a Schematron schema simply defines rules that are applied at particular locations in the instance document. The other schema languages evaluated all fall into the first category.

Two ways of combining schema languages are using two or more schemas each written in a different language, and using a single schema that contains features from two or more languages. These are considered in more detail below. For simplicity, only combinations of two schema languages are considered.

4.2.2. Using Two Schemas - Each Written in a Different Language

Have a primary schema written in one language and a secondary schema written in a different language. The primary schema contains most of the specifications. The secondary schema provides additional specifications that cannot be written satisfactorily using the primary schema language. When performing validation, the instance document is validated against each schema in turn, using a different validator for each schema.

If both primary and secondary schema languages are Type 1 languages (according to the classification in section 4.2.1) then there will necessarily be a high degree of redundancy between the schemas. Because each schema must completely define anything that may appear in the instance document, a high proportion of each schema will consist of the same specifications.

Therefore the only practical way to have two schemas each written in a different language is to use Schematron for the secondary schema and one of the other languages for the primary schema. In this case the secondary schema can contain only those specifications that have not already been defined in the primary schema.

4.2.3. Using a Single Schema that Contains Features from Two Languages

Have one schema written in the primary schema language, containing some specifications that are written in the secondary schema language. Specifications written in the secondary schema language can be differentiated from the specifications written in the primary

schema language by declaring a namespace for the secondary schema language.

Validation against such a schema requires either of the following two tools:

- A validator for the primary schema language that implements the required features from the secondary schema language. Validators are available for each of the languages evaluated but are not available for combinations of languages. Therefore a custom validator would have to be created.
- A tool that extracts the specifications that are written in the secondary schema language from the schema, producing essentially two schemas that are then used as in section 4.2.2. Such a tool could consist of an XSLT style sheet and would be relatively simple. Following the conclusions of section 4.2.2, only Schematron could be used for the secondary schema language, due to its capability of validating partial schemas.

Assuming that the task of creating a custom validator is beyond the scope of this project, the only practical way to use two languages within a single schema is to embed Schematron code into a schema written in one of the other schema languages. An example of embedding Schematron code into a W3C XML Schema is given in [SCHTR6].

4.3. Combining Schema Languages for the FDMS DIF

As concluded in section 4.2, the only appropriate combination is Schematron with one of the other schema languages. Both methods of combining schema languages (as covered in

sections 4.2.2 and 4.2.3) have their advantages. The first method - to have a secondary schema in Schematron and a primary schema in any of the other schema languages - was chosen, because it results in a less complicated process for validating an instance document.

CHAPTER 5

SELECTION OF SCHEMA LANGUAGE(S)

5.1. Candidates

As a result of the analysis in Section 4, we are left with two groups of candidates:

Group 1

A single schema language - any of the six schema languages evaluated:

- TREX
- RELAX
- DSD
- W3C XML Schema
- Schematron
- Examplotron

Group 2

Combination of a primary schema language and a secondary schema language, with

Schematron as the secondary schema language, and any of the other six schema languages as the primary schema language.

- TREX with Schematron
- RELAX with Schematron
- DSD with Schematron
- W3C XML Schema with Schematron
- Examplotron with Schematron

5.2. Selected Schema Languages

The language combination selected was W3C XML Schema with Schematron. When used in combination, these two languages fulfill the requirements of the FDMS DIF to a greater degree than any of the other candidates. Details are given below:

W3C XML Schema

- Provides at least as much information about elements and attributes as a DTD is capable of. (See Requirement 1.4.3.1).
- Data types (See Requirement 1.4.3.2)
  - Provides the largest set of built-in data types of any schema language.
  - Allows additional restrictions to be placed on built-in data types.

  - Allows reusable user-defined data types to be created.

- Provides uniqueness and primary-key/foreign-key constraints. (See Requirement 1.4.3.3).
- The Schema Design View in XML Spy provides a highly readable representation of a schema (See Requirement 1.4.3.4)
- Very good tool support (See Requirement 1.4.3.5)
- High degree of modularity (See Requirement 1.4.3.6)

Schematron

- Allows complex data types to be specified (See Requirement 1.4.3.2)
- Allows a wide range of complex constraints to be expressed. Provides greater power than W3C XML Schema in expressing uniqueness and primary-key/foreign-key constraints (See Requirement 1.4.3.3).
- Good tool support (Requirement 1.4.3.5)
- Can be used in combination with any other schema language.

5.3. Elimination of Remaining Candidates

Reasons for the elimination of the remaining candidates are given below.

Group 1

TREX

- Readable, in terms of the schema source code. However, W3C XML Schema allows superior visualization using the Schema Design View of XML Spy. (See Requirement 1.4.3.4).
- Allows some complex constraints (See Requirement 1.4.3.3) – context sensitivity - that are not possible with W3C XML Schema. However, Schematron is capable of expressing the same constraints (and many others).
- Capable of expressing most of the data type requirements (see Requirement 1.3.3.2) but cannot express completely the FDMS date data type.

RELAX

- Readable, in terms of the schema source code. However, W3C XML Schema allows superior visualization using the Schema Design View of XML Spy. (See Requirement 1.4.3.4).
- Allows some complex constraints (See Requirement 1.4.3.3) – context sensitivity - that are not possible with W3C XML Schema. However, Schematron is capable of expressing the same constraints (and many others).
- More data typing capability than DTD but less than XML Schema (see Requirement 1.4.3.2),

in particular:

- Reusable user-defined data types cannot be created
- String formats cannot be expressed.

W3C XML Schema (stand alone)

- Cannot express the complex constraints required by the FDMS DIF (see Requirement 1.4.3.3) - for example, the constraints listed in section 3.2.3.
- Capable of expressing most of the data type requirements (see Requirement 1.4.3.2) but cannot express completely the FDMS date data type.

DSD

- Readable, in terms of the schema source code. However, W3C XML Schema allows superior visualization using the Schema Design View of XML Spy. (See Requirement 1.4.3.4).
- Allows complex constraints that cannot be expressed using W3C XML Schema. However, Schematron is capable of expressing the same constraints and more. (See Requirement 1.4.3.3)
- More data typing capability than DTD but less than W3C XML Schema (see Requirement 1.4.3.2). In particular, there are no built-in types – everything is a string.

Schematron (stand alone)

- It would be possible to use Schematron as the only schema language for the FDMS DIF. All the specifications in the original FDMS DIF DTD documents could be replicated using Schematron, and all the complex constraints and data type requirements could be fulfilled. However, using Schematron in this manner would result in a highly unreadable schema (see Requirement 1.4.3.4). The schema would also be very difficult to write. It is preferable to use Schematron as a complement to another schema language, using it only for those specifications that cannot be handled by the primary schema language.

Examplotron

- Schema visualization, as provided for W3C XML Spy by the Schema Design View of XML Spy, is not available for Examplotron schemas. (See Requirement 1.4.3.4).
- Few tools available (See Requirement 1.4.3.5)
- Does not handle large, complex schemas well – therefore this is not a good choice for the FDMS DIF.
- Provides slightly less structural information than a DTD (see Requirement 1.4.3.1).
- Does not have the following data typing features that are available in W3C XML Schema (see Requirement 1.4.3.2):
  - No built-in data types

- No reusable user-defined types

- Examplotron does provide complex constraints (See Requirement 1.4.3.3) and complex data type specifications (See Requirement 1.4.3.2), but these are more easily written in Schematron.

Group 2

TREX with Schematron

- Lacks the level of schema visualization possible with W3C XML Schema (with XML Spy) (See Requirement 1.4.3.4).

RELAX with Schematron

- Lacks the level of schema visualization possible with W3C XML Schema (with XML Spy) (See Requirement 1.4.3.4).
- RELAX lacks some of the data typing features of W3C XML Schema. Therefore more of the data typing requirements (See Requirement 1.4.3.2) must be handled by the Schematron schema than is the case for the W3C XML Schema and Schematron combination. Because of the complexity and limited readability of a Schematron schema, it is generally best to perform as much of the validation as possible in the primary schema, using the Schematron (secondary) schema only for those constraints that cannot be expressed easily in the primary

schema.

DSD with Schematron

- Lacks the level of schema visualization possible with W3C XML Schema (with XML Spy) (See Requirement 1.4.3.4).

Examplotron with Schematron

- As noted earlier, Examplotron is not a good choice for the FDMS DIF. Examplotron with Schematron is an improvement but is not as capable as W3C XML Schema with Schematron.

CHAPTER 6

PROPOSED SOLUTION

6.1. Introduction

Schemas for the Model and Point of Contact sections of the FDMS DIF were written, using W3C XML Schema as the primary schema language and Schematron as the secondary schema language. Section 6.2 gives a high-level explanation of the roles of the primary and secondary schemas for the Model and Point of Contact sections of the FDMS DIF. A low-level description of the schemas is given in Section 6.3. Section 6.4 describes an integrated solution for validating schemas against both the primary and secondary schemas.

6.2. Roles of the Primary and Secondary Schemas

6.2.1. General Description of Roles

For each of the Point of Contact and Model sections of the FDMS DIF, two schemas were written:

- Primary schema in W3C XML Schema –

- Contains all the detail of the original DTD.
- Defines data types.
- Can be viewed using the Object Design View in XML Spy, allowing very effective visualization of the schema.
- Provides effective validation if used alone. However, complete validation can only be achieved by using the primary schema in combination with the secondary schema.

- Secondary (top-up) schema in Schematron –
  - Defines constraints that cannot be adequately expressed in the primary schema.
  - Designed to be used in combination with the primary schema – does not provide effective validation if used alone.
  - The secondary schema is much smaller (contains fewer specifications) than the primary schema.

Complete validation of an XML instance document involves validating against the primary and secondary schemas, in sequence. For coarse validation, the primary schema may be used alone.

6.2.2. Roles of Schemas for the FDMS Point of Contact DIF

- Roles of primary schema in W3C XML Schema –
  - Contains all the specifications from the original DTD

- Specifies completely all the data types required for the FDMS Point of Contact DIF, according to [FDMS3], except for the date type.
- Provides a coarse specification of the date type – valid formats and numerical range limits for the components of the date are specified. However, there is no correlation between values of the day component and values of the month and year components.

- Role of secondary schema in Schematron –
  - Provides additional specification of the date type, allowing complete validation of date values when used in combination with the primary schema.

For example, the following code segment would be valid according to the primary schema but would be invalid according to the secondary schema:

```
<Point_Of_Contact_Employer_Start_Date>29 Feb 1945
</Point_Of_Contact_Employer_Start_Date>
```

6.2.3. Roles of Schemas for the FDMS Model DIF

- Roles of primary schema in W3C XML Schema –
  - Contains all the specifications from the original DTD
  - Specifies completely all the data types required by the FDMS Model DIF, according to [FDMS2].

- Role of secondary schema in Schematron –

- Specifies the constraints required by the FDMS Model DIF, according to [FDMS2], and as given in Appendix B.

For example, the following code segment would be valid according to the primary schema but would be invalid according to the secondary schema (violates Constraint B-4.3 in Appendix B):

```
<Association_Data>
    ... ...
    <Entity_Internal>
        <Entity_SDS_ID>7</Entity_SDS_ID>
    </Entity_Internal>
    <Entity_Internal>
        <Entity_SDS_ID>7</Entity_SDS_ID>
    </Entity_Internal>
</Association_Data>
```

6.3. Description of the Primary and Secondary Schemas

6.3.1. The Primary Schemas (W3C XML Schemas)

The W3C XML Schemas for the Point of Contact and Model sections of the FDMS DIF can be found in the sections C-1 through C-7 of Appendix C. Rather than have just two

schemas (one for Point of Contact and one for Model), a modular approach was taken, as explained in the following sections, resulting in a total of 7 schemas.

6.3.1.1 Primary Schemas: Schema Design Principles

The following principles were followed in order to achieve enhanced modularity:

Principle 1: Create Reusable Data Type Specifications

The reusable data type specifications created fall into two categories:

- "Global" - Data type specifications that are reusable across all FDMS DIF schemas. In general, names of data types in this category were prefixed with "FDMS_". For example, one "global" data type is called "FDMS_Null_Type_1".
- "Local" - Data type specifications that are intended for use only within the schema in which they are defined. Often, the data types in this category build on "global" data types. The names of data types in this category are prefixed with "FDMS_" followed by the name of the schema in which they are defined. For example, "FDMS_Model_Cardinality_Type" is the name of a "local" data type defined in the FDMS Model DIF schema.

Principle 2: Build Data Types in a Modular Manner

Build simple data types and then combine these to form more complex data types.

For example the FDMS_Strict_Date_Type data type is built by combining 11 subtypes (see FDMS_Strict_Date_Type.xsd in the FDMS_Common_Files folder).

Principle 3: Create Reusable Schemas

W3C XML Schema has an include feature that allows all the specifications in one schema to be included into another. Repetition of specifications can be avoided by placing reusable specifications in a schema that is then included into other schemas. For instance, all "global" data type specifications were placed into a schema.

6.3.1.2 Primary Schemas: List of Schemas

1. FDMS_Model_Schema.xsd (Appendix C, section C-1) – the main schema for the Model section of the FDMS DIF. Contains "local" data type specifications and structural specifications.
2. FDMS_Point_Of_Contact_Schema.xsd (Appendix C, section C-2) – the main schema for the Point of Contact section of the FDMS DIF. Contains "local" data type specifications and structural specifications.
3. FDMS_Source_Schema.xsd (Appendix C, section C-3) – contains specifications required by FDMS_Model_Schema.xsd. Contains "local" data type specifications and structural specifications.
4. FDMS_Native_Library_Schema.xsd (Appendix C, section C-4) – contains specifications

required by FDMS_Model_Schema.xsd Contains "local" data type specifications and structural specifications.

5. FDMS_Taxonomy_Schema.xsd (Appendix C, section C-5) – contains specifications required by FDMS_Model_Schema.xsd. Contains "local" data type specifications and structural specifications.

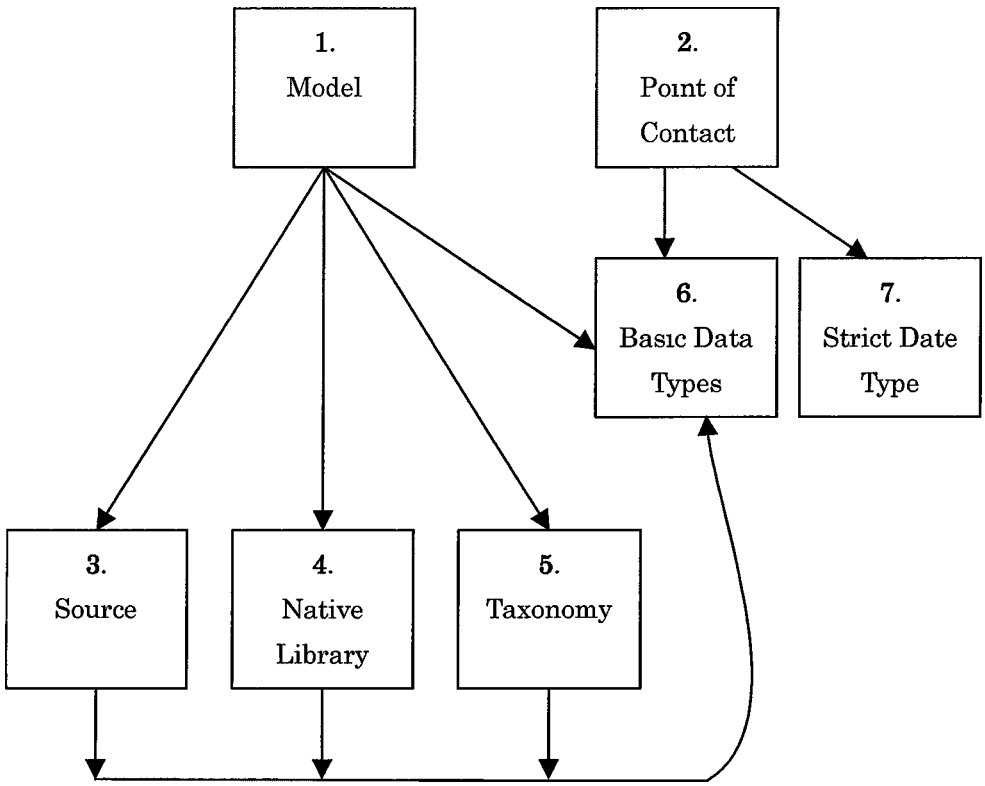
6. FDMS_Basic_Datatypes.xsd (Appendix C, section C-6) – contains "global" data type specifications.

7. FDMS_Strict_Date_Type.xsd (Appendix C, section C-7) – contains the specification of the date data type (a "global" data type).

6.3.1.3 Primary Schemas: Schema Dependency Diagram

The following diagram shows how the schemas listed in section 6.3.1.2 connect. When an arrow connects two schemas, the meaning is that the schema at the start of the arrow uses the schema at the tip of the arrow.

Figure 2 - Primary Schemas: Schema Dependency Diagram



6.3.2. The Secondary Schemas (Schematron Schemas)

The Schematron schemas for the Point of Contact and Model schema sections of the FDMS DIF can be found in sections C-8 and C-9 of Appendix C.

6.3.2.1 Secondary Schemas: Schema Design Principle

Schematron provides fewer features for enhancing reusability and modularity than W3C XML Schema. Schematron does allow a rule to be specified as reusable. However, rules

are only reusable within one file – there is no include feature such as W3C XML Schema provides. When creating the Schematron schemas, the only schema design principle for increasing modularity was to use reusable rules wherever appropriate.

6.3.2.2 Secondary Schemas: List of Schemas

- FDMS_Point_Of_Contact_Topup.sch (Appendix C, section C-8) – contains a reusable *rule* that contains constraints on date values. Also contains two *rules* that apply the reusable *rule* to specific elements in the instance document.
- FDMS_Model_Topup.sch (Appendix C, section C-9) – contains *rule*s that implement the constraint requirements listed in Appendix B. Does not contain any reusable *rules*.

6.4. Integrated Solution

6.4.1. Background

Using two schemas in sequence, one written in W3C XML Schema and the other in Schematron, to validate an instance document presented a particular disadvantage – the combined validation process was cumbersome. Validation against a W3C XML Schema is easy and convenient using the automated tools provided by IDEs such as XML Spy. However, validation against a Schematron schema involves a sequence of XSLT transformations which are inconvenient to perform manually. The ideal solution would be to be able to perform

Schematron validation as easily as W3C XML Schema validation, and to be able to perform both types of validation from within the same IDE. Because of the visualization of W3C XML Schemas provided by the XML Spy IDE, it would be advantageous to use this IDE.

6.4.2. Requirements

A program was required that would:

- Allow validation of instance documents against a Schematron schema to be performed within the XML Spy IDE.
- Allow the user to perform validation of an instance document against a Schematron schema using controls on the Graphical User Interface of the XML Spy IDE, using steps similar to those used to perform validation of an instance document against a W3C XML Schema.
- Use the schematron-report stylesheet (provided at the Schematron home page [SCHTR5]) for the XSLT transformations involved in the validation process. Schematron-report generates two-frame HTML output for the validation results. Validation errors are displayed in the upper frame. Clicking on an error brings up the section in the instance document where the error occurred, in the lower frame.

6.4.3. Solution

A plug-in for the XML Spy IDE was written. The plug-in was written as a Visual

Basic DLL and uses the XML Spy API and plug-in framework. The plug-in automates the process of validating an instance document against a Schematron schema. This process involves a sequence of XSLT transformations. The schematron-report stylesheet (available at the Schematron home page [SCHTR5]) is used in the transformations.

The plug-in adds a "Schematron" menu to the XML Spy menubar, and a "Validate" icon to the toolbar. Using these, the user can perform validation of an instance document against a Schematron schema in much the same way as is used to perform validation against a W3C XML Schema in XML Spy. The steps are:

1. From the Schematron menu, select "Assign a Schematron schema". A pop-up dialog box prompts the user to choose a Schematron schema to assign to the instance document. This step only needs to be performed if a Schematron schema is not already assigned.
2. From the Schematron menu, select "Validate", or click the "Validate" icon on the toolbar, to perform the validation.

The Visual Basic source code for the plug-in can be found in section C-10 of Appendix C.

Appendix D illustrates validating instance documents against W3C XML Schemas, using the XML Spy IDE, and validating instance documents against Schematron schemas, using the plug-in.

CHAPTER 7

CONCLUSIONS

7.1. Fulfillment of research goals

As stated in section 1.4 these were:

- To evaluate current schema languages for XML – completed in sections 2 and 3.
- To investigate the possibility of using a combination of schema languages to express a schema – completed in section 4.
- To propose a schema language solution for the FDMS DIF – the proposed solution is described in section 6.
- To validate the schema language solution by extended implementation – the implementation is described in section 6.
- To develop a methodology for determining the appropriate schema language, or combination of schema languages, for a given application – such a methodology was developed during the course of the project and is stated in section 7.2 below.

7.2. A Methodology for Determining the Schema Language for an Application

7.2.1 Step 1: Study the Application Domain

- Study the application domain. Develop a set of high-level requirements that a schema language for the application must satisfy. Examples of requirements are:
  - Capability to specify data types for element or attribute data content.
  - Capability to specify semantic constraints on the structure of the instance document or the data content of the elements and attributes.
  - Availability of tools for the language.
  - Speed and efficiency of available validators for the language.
  - Capability to generate visualization of a schema.
  - Language is easy to learn
  - Schemas written using the language are intuitive
  - Language allows a modular approach to writing schemas
  - Popularity / stability of the schema language
- If a schema language is already being used for the application, study that schema language. Determine the strengths and shortcomings of the language with respect to the requirements of the application.

7.2.2 Step 2: Research Schema Languages

- Research schema languages, using websites, journals, textbooks, and white papers. The references in the Bibliography provide starting points. In particular:
  - Determine what features of a schema language are desirable, in general.
  - Study the features of each available schema language
- Perform a formal comparison of the available schema languages:
  - Compile a list of features, including both features that are generally desirable in a schema language, and features that are relevant to the requirements developed in Step 1.
  - Compare each language by feature.
  - Include in the comparison the schema language that is currently used for the application (if any).

7.2.3 Step 3: Select Schema Languages for Substantive Evaluation

- Use a process of elimination.
- Eliminate as many possibilities as possible before commencing the substantive evaluation in Step 4.
- Some criteria for elimination:
  - The language offers no improvement, or only a small improvement, over the language that is currently used for the application

- The language is inappropriate for the application. For instance, the language may be specifically for metadata specifications, when a more general-purpose language is required.
- The language is similar to another schema language but lacks some important features provided by the other language, while not providing any useful features that are not provided by the other language.

7.2.4 Step 4: Conduct Substantive Evaluation of Selected Languages

Thoroughly research each schema language

The following are useful areas to focus on:

- Language features.
- Tools that support the language - validators, tools for graphical visualization of schemas, Integrated Development Environments (IDEs).
- Resources for learning the language.

Create test schemas using each language

Schemas are developed in each of the languages in order to determine the degree to which each language is capable of handling the requirements of the application. The following steps are involved:

- Develop a comprehensive list of specific requirements for the application schema. This is an elaboration of the high-level requirements developed in Step 1. In particular, specific requirements are needed for the following:
  - Data types – for example, a particular element must contain a code of the form aaa-bbb, where aaa are letters and bbb are digits. Data type requirements could involve some of the following:
    - Specific formats for string values
    - Numerical limits
    - Types of number – integer, real, decimal
    - Arithmetic analysis of a data value
  - Semantic constraints – for example, the value contained by element X depends on whether a particular sibling of X has a particular value. The following are some types of semantic constraints:
    - Uniqueness
    - Primary key-foreign key
    - Parent – dependency
    - Sibling – dependency
- The comprehensive list of specific requirements is likely to consist of a large number of requirements that are all variations on a basic group of requirements. To facilitate the

creation of the test schemas, create a minimum set of specific requirements that is representative of the comprehensive list of specific requirements.

- The minimum set of specific requirements and the high-level requirements from Step 1 together comprise the evaluation criteria for the test schemas. In general, develop as many decision-relevant criteria as possible before commencing the evaluation process, but be alert for suggestion of additional decision-relevant criteria.
- Specify a section or sections of the problem domain to be represented by the test schemas. These sections should, together, use all the evaluation criteria developed in this step.
- Using each language, create schemas for the selected sections of the problem domain. Aim to fulfill all the evaluation criteria.

Evaluate each language

Conduct evaluations based on:

- Language features and ease of use
- Comparison with the existing language (if any) used for the application
- Tools
- Degree to which the test schemas satisfy the minimum set of requirements.

7.2.5 Step 5: Select a Schema Language or a Combination of Schema Languages

- Consider which combinations of languages are possible or practical, based on the properties of the languages. In particular, consider using Schematron for a secondary schema if the application domain has complex constraints and data types that cannot be expressed adequately using other schema languages.
- Consider each possibility - every language and every combination of languages - individually. Using the evaluations from Step 4, select the best language or combination of languages for the application.

7.3. Comments on the Proposed Solution

The preparation of the schemas for the proposed solution described in section 6 was affected by two factors. These are described in the following two subsections.

7.3.1 Design of the DIF predated the choice of schema language

The Data Interchange Format (DIF) had been designed in advance of the proposed schema language solution for the DIF. Therefore, the design of the DIF did not take full advantage of the features of the proposed schema languages – W3C XML Schema and Schematron.

Redesigning the DIF could bring about improvements to the schemas, allowing more

specifications to be expressed using the more readable W3C XML Schema language and fewer using the Schematron language, which is less readable and is used for those specifications that cannot be expressed using W3C XML Schema.

Two examples of redesigning the FDMS DIF in order to improve the schemas:

- Redesign the structure of the XML for the FDMS DIF. The uniqueness constraints required by the FDMS DIF were implemented in Schematron rather than W3C XML Schema. However, by adding some new elements and restructuring the FDMS DIF, it would have been possible to use the capabilities of W3C XML Schema to express the uniqueness constraints.
- Change the requirements for FDMS DIF data values to take advantage of the built-in date data type provided by W3C XML Schema. Essentially this would mean that the FDMS DIF data values could have only the format specified by the W3C XML Schema built-in date type.

It may not be practical or desirable to redesign the data interchange format. If redesigning the DIF is an option, then there is a trade-off between the quality of the XML instance document and the quality of the schema representation.

7.3.2 All validation was performed in the schemas

In preparing the schemas, it was assumed that the role of the schemas is to fully describe all aspects of the instance document. The approach taken was therefore to provide as

much validation as possible using the schemas. A different approach would be to provide only part of the validation in the schemas, and leave the rest of the validation for the sending and receiving data systems to handle. Using this approach, there would be no need for Schematron schemas.

7.4. Further work

Possible directions for work that builds on this project are given below:

- Extend the Schematron language – introducing new language constructs to Schematron could make the task of writing Schematron schemas for the FDMS DIF, or other applications of similar complexity, much easier. The following features are desirable:
  - An *include* feature – allowing specifications written in one schema to be included into another schema.
  - A mechanism for reusing the XPath expressions that are used to write the rules in a Schematron schema.
- Extend the functionality of the plug-in for XML Spy that was created as part of the integrated solution described in section 6 – a number of features could be added to the plug-in to facilitate editing of Schematron schemas in addition to the current features which deal only with validation against an instance document.

APPENDIX A

SCHEMA LANGUAGE COMPARISON CHART

Table 1 provides a comparison of the 14 schema languages studied. The table consists of:

- One column for each schema language, with an abbreviation of the name of the schema language in the column header. In addition, there is a column for the DTD schema language.
- Language features, listed on the odd (dark-colored) rows
- Support of each feature, by schema language, given on the even (light-colored) rows.

Notes

Abbreviations of Names of Schema Languages

The newest versions of the schema languages were used, unless otherwise indicated.

DDM: DDML (Document Definition Markup Language)

XSc: XSchema

W3C: W3C XML Schema

XDR: XDR (XML- Data Reduced)

DCD: DCD (Document Content Description)

SOX: SOX (Schema for Object-oriented XML)

TRX: TREX (Tree Regular Expressions for XML)

Note: using W3C XML Schema data types

RLX: RELAX (Regular Expression Language for XML)

Note: refers to both RELAX Core and RELAX Namespace

DSD: DSD (Document Semantic Definition)

Note: DSD 1.0

SAF: SAF (Schema Adjunct Framework)

RDF: RDF (Resource Description Framework)

Hk: Hook

Sch: Schematron

Ex: Examplotron

DTD: Document Type Definition

DTD-style restrictions on referential integrity constraints

With DTD, an attribute is declared to be unique by assigning it a data type of "ID". However, uniqueness is enforced over all ID values in the document, regardless of which attribute the value occurs in. So if, for example, attributes a and b are both of type ID, then if a=5 at some point in the instance document, b can never have the value 5.

An attribute can be declared to be a reference to an ID value by assigning it a data type of "IDRef". However, it is not possible to link an IDRef to a *specific* ID. The only way there can be a true Foreign Key – Primary Key constraint is if there is only one attribute in the entire document that is of data type "ID", i.e. only one primary key in the entire document.

Table 1: Comparison of Schema Languages

DDM	Xsc	W3C	XDR	DCD	SOX	TRX	RLX	DSD	SAF	RDF	Hk	Sch	Ex	DTD
1. Element content – sequence specified														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	**Yes**
2. Element content – unordered set of elements														
Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	No	No	No	Yes	No	**No**
3. Element content – choice of one element from a set of elements														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
4. Element content – cardinality of *, +, ? (zero or more, one or more, zero or one)														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
5. Element content – cardinality given by integer maximum/minimum values														
No	No	Yes	No	No	Yes	No	No	No	No	No	No	Yes	Yes	**No**
6. Element content – empty element														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	**Yes**
7. Element content – contains only elements														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	**Yes**
8. Element content – contains only text (simple data content)														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
9. Element content – mixed (text and elements in any order)														
Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	No	**Yes**
10. Element content – open content model (any element or text can appear)														
Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	Yes	Yes	No	**Yes**
11. Attributes – sequence specified														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
12. Attributes – unordered set of attributes														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
13. Attributes – can either be optional or required														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	No	**Yes**
14. Attributes – choice of one attribute from a set of attributes														
No	No	No	No	No	No	Yes	No	Yes	No	No	No	Yes	No	**No**
15. Choice between either an element x or an attribute x														
No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	**No**

Table 1: Comparison of Schema Languages (continued)

DDM	Xsc	W3C	XDR	DCD	SOX	TRX	RLX	DSD	SAF	RDF	Hk	Sch	Ex	DTD
16. Number of built-in data types														
11	11	44	31	37	17	44	43	0	0	0	0	0	0	**10**
17. Restrictions on data values – enumerations														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
18. Restrictions on data values – numerical range limits														
No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	**No**
19. Restrictions on data values – complex numerical constraints, e.g. divisible by 12														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
20. Restrictions on data values – string length restrictions														
No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**No**
21. Restrictions on data values – string pattern matching / string analysis														
No	No	Yes	No	No	No	Yes	No	Yes	No	No	No	Yes	Yes	**No**
22. Restrictions on data values – combination of enumeration, numerical and string restrictions applied to a data value														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
23. Data value restrictions applicable to both elements and attributes														
No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**No**
24. User-defined reusable data value restrictions														
No	No	Yes	No	No	Yes	Yes	No	Yes	No	No	No	Yes	No	**No**
25. Default insertion – defaults for attributes														
Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No	No	No	No	**Yes**
26. Default insertion – defaults for elements														
No	No	Yes	No	Yes	No	No	No	Yes	No	No	No	No	No	**No**
27. Default insertion – context-sensitive defaults (for elements or attributes)														
No	No	No	No	No	No	No	No	Yes	No	No	No	No	No	**No**
28. Uniqueness constraints – attributes														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
29. Uniqueness constraints – elements														
No	No	Yes	Yes	No	Yes	No	No	No	No	No	No	Yes	Yes	**No**

Table 1: Comparison of Schema Languages (continued)

DDM	Xsc	W3C	XDR	DCD	SOX	TRX	RLX	DSD	SAF	RDF	Hk	Sch	Ex	DTD
30. Uniqueness constraints – within a sequence of elements														
No	No	Yes	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
31. Uniqueness constraints – within an arbitrary set of elements														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
32. Uniqueness constraints – without DTD-style restrictions (see notes)														
No	No	Yes	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
33. Primary key/foreign key constraints – key is an attribute														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	**Yes**
34. Primary key/foreign key constraints – key is an element														
No	No	Yes	Yes	No	Yes	No	No	No	No	No	No	Yes	Yes	**No**
35. Primary key/foreign key constraints– key is a sequence of elements														
No	No	Yes	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
36. Primary key/foreign key constraints – key is an arbitrary set of elements														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
37. Primary key/foreign key constraints – without DTD-style restrictions (see notes)														
No	No	Yes	No	No	No	No	No	Yes	No	No	No	Yes	Yes	**No**
38. Context sensitivity – element content depends on parent element														
No	No	No	No	No	No	Yes	Yes	Yes	No	No	No	Yes	Yes	**No**
39. Context sensitivity – element content depends on an attribute of parent														
No	No	No	No	No	No	Yes	Yes	Yes	No	No	No	Yes	Yes	**No**
40. Context sensitivity – element content depends on path from root to element														
No	No	No	No	No	No	No	No	Yes	No	No	No	Yes	Yes	**No**
41. Context sensitivity – attribute a exists only if attribute b exists (within the same element)														
No	No	No	No	No	No	Yes	No	Yes	No	No	No	Yes	No	**No**
42. Context sensitivity – content of an element can determine the content of any other element, located anywhere in the document														
No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	**No**
43. Modularity – reusable definitions														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	**Yes**

Table 1: Comparison of Schema Languages (continued)

DDM	Xsc	W3C	XDR	DCD	SOX	TRX	RLX	DSD	SAF	RDF	Hk	Sch	Ex	DTD
44. Modularity – reusable definitions can be extended														
No	No	Yes	No	No	Yes	Yes	No	No	No	Yes	No	Yes	No	**No**
45. Modularity – a schema can divided into several files														
No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	Yes	**Yes**
46. Namespace aware														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	**No**
47. Mechanism for schema documentation (other than XML comments <!-- -->)														
Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	No	Yes	No	**No**
48. At least one software tool available for the schema language														
Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	**Yes**
49. Schema language uses XML syntax														
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**No**

APPENDIX B

COMPLEX CONSTRAINTS FOR THE FDMS MODEL DIF

[FDMS2], sections 2.1 – 2.2, provide a set of high-level rules that apply to the FDMS Model DIF. Using these rules as a reference, specific constraints were developed. These constraints were grouped into six categories, as follows.

Category B-1: Uniqueness

The constraints in this category specify that an element must contain a value that is unique within a defined set of element types.

Constraint B-1.1

Reference: [FDMS2] 2.1.2

The values contained in all elements whose names end with "_SDS_ID" must be unique, within the set of elements whose names end with "_SDS_ID" that occur as children of elements whose names end with "_Addition", "_Forward", "_Replacement" and "_Update".

Constraint B-1.2

Reference: [FDMS2] 2.1.2

The values contained in all elements whose names end with "_RDS_ID" must be unique, within the set of elements whose names end with "_RDS_ID" that occur as children of

elements whose names end with "_Addition", "_Forward", "_Replacement" and "_Update".

Category B-2: Primary Key / Foreign Key

A Primary Key - Foreign Key constraint which applies to the values contained in elements belonging to a particular subset of the elements in the document.

Constraint B-2.1

Reference: [FDMS2] 2.1.3

Subset of document: Elements that are associated with a single Model component. For example, all the elements that are associated with an Association, i.e. Association_SDS_ID, Association_Addition, etc.

Primary Key: Values contained in elements whose names end with "SDS_ID" and that are children of elements whose names end with "_Addition" or "_Forward".

Foreign Key: The values contained in all elements whose names end with "_SDS_ID" and that are children of elements whose names end with "_Internal".

Category B-3: Parent-sibling

The range of values that an element may contain depends on the type of the parent element.

Constraint B-3.1

Reference: [FDMS2] 2.1.3

An element whose name ends with "_SDS_ID" may contain one of the values {"N/A", "TBD", "UNK"} only if the element occurs as a child of an element whose name ends with "_Addition", "_Forward" or "_Internal".

Constraint B-3 2

Reference: [FDMS2] 2.1.3

An element whose name ends with "_RDS_ID" may contain one of the values {"N/A", "TBD", "UNK"} only if the element occurs as a child of an element whose name ends with "_Replacement", "_Existing" or "_Update".

Category B-4: Sibling-sibling

Element presence and/or content are dependent on a sibling, where siblings are not necessarily adjacent. Four sub-categories:

B-4.1 Presence of element depends on presence of sibling

B-4.2 Presence of element depends on content of sibling

B-4.3 Content of element depends on presence of sibling

B-4.2 Content of element depends on content of sibling

Constraint B-4.1

Reference: [FDMS2] 2.2.1, item 1

If an Association_Data element does not contain an Association_Stereotype element, or if the Association_Stereotype element contains a value of "other", then the Association_Data element must contain an Association_Description element. (Combination of subcategories B-4.1 and B-4.2)

Constraint B-4.2

Reference: [FDMS2] 2.1.7

This constraint applies to elements that contain both an optional Stereotype (an element whose name ends with "_Stereotype") and an optional Type (an element whose name ends with "_Type"):

If a Stereotype is not present or contains a value of "other", then a Type must be present, and must not contain any of {"N/A", "TBD" or "UNK"}. (Combination of subcategories B-4.1, B-4.2, B-4.3 and B-4.4)

Constraint B-4.3

Reference: [FDMS2] 2.2.1

The Entity_Existing and Entity_Internal elements in an Association_Data element must each contain a different value. (Subcategory B-4.4)

Category B-5: Complex Primary Key/Foreign Key

Constraints that combine a Primary Key/Foreign Key (Category B-2) constraint with a Category B-3 or Category B-4 constraint on the Primary Key.

Constraint B-5.1

Reference: [FDMS2] 2.2.3, items 2, 3

If an Entity_Performing element contains an Entity_Internal element, then the Entity_SDS_ID element child of the Entity_Internal element must contain a value that is a reference to an Entity_Addition element that has an Entity_Stereotype or Entity_Type value of any of the following: "Agent", "Organization", "Equipment" or "Person".

Constraint B-5.2

Reference: [FDMS2] 2.2.3, items 4, 5, 6

If an Entity_Data element contains an Entity_Component element, then

- If an Entity_Data element contains an Entity_Stereotype or Entity_Type element that contains a value of "Organization", then any Entity_Component elements contained in the Entity_Data element must contain an Entity_SDS_ID element whose value is a reference to

an Entity_Addition element that has an Entity_Stereotype or Entity_Type value of any of {'Organization', 'Equipment', 'Person'}.

- If an Entity_Data element contains an Entity_Stereotype or Entity_Type element that contains a value of "Equipment", then any Entity_Component elements contained in the Entity_Data element must contain an Entity_SDS_ID element whose value is a reference to an Entity_Addition element that has an Entity_Stereotype or Entity_Type value of "Equipment".
- If an Entity_Data element contains an Entity_Stereotype or Entity_Type element that contains a value of "Network", then any Entity_Component elements contained in the Entity_Data element must contain an Entity_SDS_ID element whose value is a reference to an Entity_Addition element that has an Entity_Stereotype or Entity_Type value of "Equipment".

Category B-6: Complex sibling-sibling

Constraints in this category are similar to the sibling-sibling constraints of Category B-4, except that the dependencies involve more than two siblings.

Constraint B-6.1

Reference: [FDMS2] 2.2.3, item 1

If an Entity_Data element contains either an Information_Field_Addition element or an Information_Field_Replacement element, then the Entity_Data element must also contain either an Entity_Type element containing a value of "Information" or an Entity_Stereotype element containing a value of "Information".

Constraint B-6.2

Reference: [FDMS2] 2.2.7, items 2, 3, 4, 5

The following all apply to children of Interaction_Data elements. (The Interaction_Data element contains a sequence of two choices between a Process_Existing element and a Process_Internal element. These are referred to below as Process1 and Process2. Process_Existing elements and Process_Internal elements contain Process_RDS_ID and Process_SDS_ID elements, respectively. These are referred to below as Process ID.)

- If both of Process1/Process ID and Process2/Process ID contain a value other than "N/A", the Interaction_Category element must contain a value of "Input_Output".
- Process1/Process ID and Process2/Process ID cannot both contain the value "N/A".
- If Process1/Process ID contains a value other than "N/A", and Process2/Process ID contains the value "N/A", then the Interaction_Category element must contain a value of "Output".
- If Process1/Process ID contains the value "N/A" and Process2/Process ID contains a value other than "N/A", then the Interaction_Category element must contain the value "Input".

Constraint B-6.3

Reference: [FDMS2] 2.2.10, items 1, 2, 3

A Process_Control_Data element must contain one of the following combinations of child elements:

- One Process_Control_Element_LHS element, containing either a Condition_Internal element or a Condition_Existing element, and no Process_Control_Element_RHS element.
- Combination of:
  - One or more Process_Control_Element_LHS elements, containing one or more of the following child elements: {Condition_Internal, Condition_Existing, Measure_Internal, Measure_Existing}, and ...
  - A Process_Control_Element_Constant element that does not contain any of the values {"N/A", "TBD", "UNK"} and ...
  - No Process_Control_Element_RHS element.

- One or more Process_Control_Element_LHS elements, one or more Process_Control_Element_RHS elements, and one Process_Control_Relation element.

Constraint B-6.4

Reference: [FDMS2] 2.2.12, item 4

The following apply to children of the Process_Sequence_Data element:

- If there is only one Sequence_Element element, then the value contained in the Process_Sequence_Part element must be one of {"Unconstrained", "Concurrent", "Iteration", "Continuous"}
- If there are two Sequence_Element elements, the two Process_Sequence_Part elements must contain values from the following pairs: {("Unconstrained", "Unconstrained"), ("Concurrent", "Concurrent"), ("Iteration, "Sequential"), ("Iteration", "Concurrent"), ("Continuous", "Sequential"), ("Continuous", "Concurrent"), ("Sequential", "Sequential"), ("Start", "Prior"), ("Start", "Start"), ("Start", "During"), ("Start", "End"), ("Start", "After"), ("End", "Prior"), ("End", "Start"), ("End", "During"), ("End", "End"), ("End", "After")}

APPENDIX C

SOURCE CODE

The W3C XML Schemas are contained in the following sections:

- C-1 FDMS_Model_Schema.xsd
- C-2 FDMS_Point_Of_Contact_Schema.xsd
- C-3 FDMS_Source_Schema.xsd
- C-4 FDMS_Native_Library_Schema.xsd
- C-5 FDMS_Taxonomy_Schema.xsd
- C-6 FDMS_Basic_Datatypes.xsd
- C-7 FDMS_Strict_Date_Type.xsd

The Schematron schemas are contained in the following sections:

- C-8 FDMS_Point_Of_Contact_Topup.sch
- C-9 FDMS_Model_Topup.sch

The Visual Basic source code for the plug-in for the XML Spy Integrated Development Environment is in the following section:

- C-10 XMLSpyPlugIn.cls

C-1 FDMS_Model_Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
	<xsd:annotation>
		<xsd:documentation>Model_Representation is the root
element</xsd:documentation>
	</xsd:annotation>
	<xsd:include
schemaLocation="..¥..¥Taxonomy¥W3C_XML_Schema_Files¥FDMS_Taxonomy_Sch
ema.xsd"/>
	<xsd:include
schemaLocation="..¥..¥Native_Library¥W3C_XML_Schema_Files¥FDMS_Native
_Library_Schema.xsd"/>
	<xsd:element name="FDMS_MODEL_DIF"
type="FDMS_MODEL_DIF_ComplexType">
		<xsd:annotation>
			<xsd:documentation>root
element</xsd:documentation>
		</xsd:annotation>
	</xsd:element>
	<xsd:complexType name="FDMS_MODEL_DIF_ComplexType">
		<xsd:sequence maxOccurs="unbounded">
			<xsd:element ref="Model_Representation"/>
```

```
            </xsd:sequence>
            <xsd:attribute  name="Version"  type="FDMS_Version_Type"
use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Association_AdditionType">
            <xsd:sequence>
                  <xsd:element ref="Association_SDS_ID"/>
                  <xsd:element             name="Association_Data"
type="Association_DataType"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:element                    name="Association_Cardinality"
type="FDMS_Model_Cardinality_Type"/>
      <xsd:complexType name="Association_DataType">
            <xsd:sequence>
                  <xsd:choice minOccurs="0">
                        <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                        <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                  </xsd:choice>
                  <xsd:element ref="Association_Cardinality"/>
                  <xsd:element        ref="Association_Stereotype"
minOccurs="0"/>
```

```
                    <xsd:element              ref="Association_Type"
minOccurs="0"/>
                    <xsd:choice>
                          <xsd:element       name="Entity_Existing"
type="Entity_ExistingType"/>
                          <xsd:element       name="Entity_Internal"
type="Entity_InternalType"/>
                    </xsd:choice>
                    <xsd:choice>
                          <xsd:element       name="Entity_Existing"
type="Entity_ExistingType"/>
                          <xsd:element       name="Entity_Internal"
type="Entity_InternalType"/>
                    </xsd:choice>
                    <xsd:element          ref="Association_S_D_Flag"
minOccurs="0"/>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                          <xsd:element
ref="Source_Element_Existing"/>
                          <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element name="Association_RDS_ID" type="FDMS_DS_ID_Type"/>
       <xsd:complexType name="Association_ReplacementType">
```

```
                <xsd:sequence>
                        <xsd:element ref="Association_RDS_ID"/>
                        <xsd:element            ref="Association_SDS_ID"
minOccurs="0"/>
                        <xsd:element             name="Association_Data"
type="Association_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Association_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element                          name="Association_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                        name="Association_Stereotype"
type="FDMS_Model_Association_Stereotype_Type"/>
        <xsd:element                              name="Association_Type"
type="FDMS_Model_Type_Type"/>
        <xsd:element                       name="Characteristic_Abstract"
type="FDMS_Model_Abstract_Type"/>
        <xsd:complexType name="Characteristic_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Characteristic_SDS_ID"/>
                        <xsd:element          name="Characteristic_Data"
type="Characteristic_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Characteristic_DataType">
                <xsd:sequence>
```

```
                        <xsd:choice>
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                        <xsd:element ref="Characteristic_Name"/>
                        <xsd:element       ref="Characteristic_Abstract"
minOccurs="0"/>
                        <xsd:element          name="Characteristic_Kind"
type="Characteristic_KindType" minOccurs="0"/>
                        <xsd:element       name="Characteristic_Synonym"
type="Characteristic_SynonymType" minOccurs="0"/>
                        <xsd:element       ref="Characteristic_S_D_Flag"
minOccurs="0"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Source_Element_Existing"/>
                                <xsd:element
ref="Source_Element_Internal"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Taxonomy_Keyword_Element_Existing"/>
```

```
                        <xsd:element
ref="Taxonomy_Keyword_Element_Internal"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Characteristic_ExistingType">
                <xsd:sequence>
                    <xsd:element ref="Characteristic_RDS_ID"/>
                    <xsd:element         ref="Characteristic_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType
name="Characteristic_Field_CorrespondenceType">
                <xsd:sequence>
                    <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Characteristic_ForwardType">
                <xsd:sequence>
                    <xsd:element ref="Characteristic_SDS_ID"/>
                    <xsd:element ref="Characteristic_Name"/>
                </xsd:sequence>
        </xsd:complexType>
```

```
        <xsd:complexType name="Characteristic_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Characteristic_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Characteristic_KindType">
                <xsd:choice>
                        <xsd:element      name="Characteristic_Existing"
type="Characteristic_ExistingType"/>
                        <xsd:element      name="Characteristic_Internal"
type="Characteristic_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:element                          name="Characteristic_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element                         name="Characteristic_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Characteristic_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Characteristic_RDS_ID"/>
                        <xsd:element         ref="Characteristic_SDS_ID"
minOccurs="0"/>
                        <xsd:element          name="Characteristic_Data"
type="Characteristic_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
```

```
        <xsd:element                          name="Characteristic_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                        name="Characteristic_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:complexType name="Characteristic_SynonymType">
                <xsd:choice>
                        <xsd:element      name="Characteristic_Existing"
type="Characteristic_ExistingType"/>
                        <xsd:element      name="Characteristic_Internal"
type="Characteristic_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="Condition_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Condition_SDS_ID"/>
                        <xsd:element                name="Condition_Data"
type="Condition_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Condition_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
```

```
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                    </xsd:choice>
                    <xsd:element ref="Condition_Name"/>
                    <xsd:element     ref="Condition_UJTL_Identifier"
minOccurs="0"/>
                    <xsd:element           ref="Condition_UJTL_Name"
minOccurs="0"/>
                    <xsd:element            ref="Condition_S_D_Flag"
minOccurs="0"/>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                          <xsd:element
ref="Source_Element_Existing"/>
                          <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
              </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Condition_ExistingType">
            <xsd:sequence>
                    <xsd:element ref="Condition_RDS_ID"/>
                    <xsd:element              ref="Condition_SDS_ID"
minOccurs="0"/>
              </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Condition_ForwardType">
```

```
                <xsd:sequence>
                        <xsd:element ref="Condition_SDS_ID"/>
                        <xsd:element ref="Condition_Name"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Condition_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Condition_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                                name="Condition_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element name="Condition_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Condition_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Condition_RDS_ID"/>
                        <xsd:element              ref="Condition_SDS_ID"
minOccurs="0"/>
                        <xsd:element               name="Condition_Data"
type="Condition_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Condition_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element                            name="Condition_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                   name="Condition_UJTL_Identifier"
```

```
type="FDMS_Model_UJTL_Identifier_Type"/>
        <xsd:element                          name="Condition_UJTL_Name"
type="FDMS_Model_UJTL_Name_Type"/>
        <xsd:complexType name="Description_Link_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Description_Link_SDS_ID"/>
                        <xsd:element        name="Description_Link_Data"
type="Description_Link_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Description_Link_DataType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element
name="Text_Description_Existing"
type="Text_Description_ExistingType"/>
                                <xsd:element
name="Text_Description_Internal"
type="Text_Description_InternalType"/>
                                <xsd:element
name="Embedded_Object_Existing" type="Embedded_Object_ExistingType"/>
                                <xsd:element
name="Embedded_Object_Internal" type="Embedded_Object_InternalType"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0">
                                <xsd:element
```

```
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                        <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Description_Link_ExistingType">
                <xsd:sequence>
                    <xsd:element ref="Description_Link_RDS_ID"/>
                    <xsd:element       ref="Description_Link_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Description_Link_InternalType">
                <xsd:sequence>
                    <xsd:element ref="Description_Link_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Description_Link_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Description_Link_ReplacementType">
                <xsd:sequence>
                    <xsd:element ref="Description_Link_RDS_ID"/>
                    <xsd:element       ref="Description_Link_SDS_ID"
```

```
minOccurs="0"/>
                    <xsd:element        name="Description_Link_Data"
type="Description_Link_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Description_Link_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Embedded_Object_AdditionType">
            <xsd:sequence>
                <xsd:element ref="Embedded_Object_SDS_ID"/>
                <xsd:element         name="Embedded_Object_Data"
type="Embedded_Object_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Embedded_Object_DataType">
            <xsd:sequence>
                <xsd:element ref="Embedded_Object_Name"/>
                <xsd:element ref="Embedded_Object_Link"/>
                <xsd:element ref="Embedded_Object_Type"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Embedded_Object_ExistingType">
            <xsd:sequence>
                <xsd:element ref="Embedded_Object_RDS_ID"/>
                <xsd:element        ref="Embedded_Object_SDS_ID"
minOccurs="0"/>
```

```
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Embedded_Object_InternalType">
            <xsd:sequence>
                <xsd:element ref="Embedded_Object_SDS_ID"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element                          name="Embedded_Object_Link"
type="FDMS_Model_Embedded_Object_Link_Type"/>
        <xsd:element                          name="Embedded_Object_Name"
type="FDMS_Model_Embedded_Object_Name_Type"/>
        <xsd:element                        name="Embedded_Object_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Embedded_Object_ReplacementType">
            <xsd:sequence>
                <xsd:element ref="Embedded_Object_RDS_ID"/>
                <xsd:element         ref="Embedded_Object_SDS_ID"
minOccurs="0"/>
                <xsd:element          name="Embedded_Object_Data"
type="Embedded_Object_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element                        name="Embedded_Object_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                          name="Embedded_Object_Type"
type="FDMS_Model_Embedded_Object_Type_Type"/>
```

```
        <xsd:element                                name="Entity_Abstract"
type="FDMS_Model_Abstract_Type"/>
        <xsd:complexType name="Entity_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Entity_SDS_ID"/>
                        <xsd:element                    name="Entity_Data"
type="Entity_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_CharacteristicType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element
name="Characteristic_Existing" type="Characteristic_ExistingType"/>
                                <xsd:element
name="Characteristic_Internal" type="Characteristic_InternalType"/>
                        </xsd:choice>
                        <xsd:element ref="Entity_Characteristic_Unit"/>
                        <xsd:element ref="Entity_Characteristic_Value"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_Characteristic_FirstType">
                <xsd:sequence>
                        <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType"/>
```

```
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType
name="Entity_Characteristic_Instance_DataType">
                <xsd:sequence>
                        <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType"/>
                        <xsd:element
ref="Entity_Characteristic_Instance_Initial_Quantity"/>
                        <xsd:element
ref="Entity_Characteristic_Instance_Minimum_Quantity"/>
                        <xsd:element
ref="Entity_Characteristic_Instance_Maximum_Quantity"/>
                        <xsd:element
ref="Entity_Characteristic_Instance_Nominal_Quantity"/>
                        <xsd:element
ref="Entity_Characteristic_Instance_Unit"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element
name="Entity_Characteristic_Instance_Initial_Quantity"
type="FDMS_Model_Entity_Characteristic_Instance_Quantity_Type"/>
        <xsd:element
name="Entity_Characteristic_Instance_Maximum_Quantity"
type="FDMS_Model_Entity_Characteristic_Instance_Quantity_Type"/>
```

```
        <xsd:element
name="Entity_Characteristic_Instance_Minimum_Quantity"
type="FDMS_Model_Entity_Characteristic_Instance_Quantity_Type"/>
        <xsd:element
name="Entity_Characteristic_Instance_Nominal_Quantity"
type="FDMS_Model_Entity_Characteristic_Instance_Quantity_Type"/>
        <xsd:element        name="Entity_Characteristic_Instance_Unit"
type="FDMS_Model_Unit_Type"/>
        <xsd:complexType name="Entity_Characteristic_ReferenceType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element       name="Entity_Existing"
type="Entity_ExistingType"/>
                                <xsd:element       name="Entity_Internal"
type="Entity_InternalType"/>
                        </xsd:choice>
                        <xsd:choice>
                                <xsd:element
name="Characteristic_Existing" type="Characteristic_ExistingType"/>
                                <xsd:element
name="Characteristic_Internal" type="Characteristic_InternalType"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_Characteristic_SecondType">
                <xsd:sequence>
```

```
                    <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Entity_Characteristic_Unit"
type="FDMS_Model_Unit_Type"/>
        <xsd:element                  name="Entity_Characteristic_Value"
type="FDMS_Long_255_Type"/>
        <xsd:complexType name="Entity_ComponentType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element        name="Entity_Existing"
type="Entity_ExistingType"/>
                                <xsd:element        name="Entity_Internal"
type="Entity_InternalType"/>
                        </xsd:choice>
                        <xsd:element
ref="Entity_Component_Cardinality"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                name="Entity_Component_Cardinality"
type="FDMS_Model_Cardinality_Type"/>
        <xsd:complexType name="Entity_DataType">
                <xsd:sequence>
                        <xsd:choice>
```

```
                        <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                        <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                    </xsd:choice>
                    <xsd:element ref="Entity_Name"/>
                    <xsd:element            ref="Entity_Stereotype"
minOccurs="0"/>
                    <xsd:element ref="Entity_Type" minOccurs="0"/>
                    <xsd:element              ref="Entity_Abstract"
minOccurs="0"/>
                    <xsd:element                 name="Entity_Kind"
type="Entity_KindType" minOccurs="0"/>
                    <xsd:element              name="Entity_Synonym"
type="Entity_SynonymType" minOccurs="0"/>
                    <xsd:element              ref="Entity_S_D_Flag"
minOccurs="0"/>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Source_Element_Existing"/>
                        <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
                    <xsd:element        name="Entity_Characteristic"
```

```
type="Entity_CharacteristicType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element              name="Entity_Component"
type="Entity_ComponentType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
name="Information_Field_Addition"
type="Information_Field_AdditionType"/>
                                <xsd:element
name="Information_Field_Replacement"
type="Information_Field_ReplacementType"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Taxonomy_Keyword_Element_Existing"/>
                                <xsd:element
ref="Taxonomy_Keyword_Element_Internal"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Native_Library_Element_Existing"/>
                                <xsd:element
ref="Native_Library_Element_Internal"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_ExistingType">
```

```
                <xsd:sequence>
                        <xsd:element ref="Entity_RDS_ID"/>
                        <xsd:element ref="Entity_SDS_ID" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_ForwardType">
                <xsd:sequence>
                        <xsd:element ref="Entity_SDS_ID"/>
                        <xsd:element ref="Entity_Name"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Entity_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_KindType">
                <xsd:choice>
                        <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                        <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:element name="Entity_Name" type="FDMS_Model_Name_Type"/>
        <xsd:complexType name="Entity_ObjectType">
```

```
                <xsd:choice>
                        <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                        <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="Entity_PerformingType">
                <xsd:choice>
                        <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                        <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:element name="Entity_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Entity_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Entity_RDS_ID"/>
                        <xsd:element ref="Entity_SDS_ID" minOccurs="0"/>
                        <xsd:element                   name="Entity_Data"
type="Entity_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Entity_ResponsibleType">
                <xsd:choice>
```

```
                    <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                    <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
            </xsd:choice>
        </xsd:complexType>
        <xsd:element name="Entity_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element name="Entity_S_D_Flag" type="FDMS_S_D_Flag_Type"/>
        <xsd:element                          name="Entity_Stereotype"
type="FDMS_Model_Entity_Stereotype_Type"/>
        <xsd:complexType name="Entity_StimulatingType">
            <xsd:choice>
                    <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                    <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
            </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="Entity_SynonymType">
            <xsd:choice>
                    <xsd:element              name="Entity_Existing"
type="Entity_ExistingType"/>
                    <xsd:element              name="Entity_Internal"
type="Entity_InternalType"/>
            </xsd:choice>
        </xsd:complexType>
```

```
        <xsd:element name="Entity_Type" type="FDMS_Model_Type_Type"/>
        <xsd:complexType name="Entity_UpdateType">
                <xsd:sequence>
                        <xsd:element ref="Entity_RDS_ID"/>
                        <xsd:element ref="Entity_SDS_ID" minOccurs="0"/>
                        <xsd:element                    name="Entity_Data"
type="Entity_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Event_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Event_SDS_ID"/>
                        <xsd:element                     name="Event_Data"
type="Event_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                                name="Event_Category"
type="FDMS_Model_Event_Category_Type"/>
        <xsd:complexType name="Event_DataType">
                <xsd:sequence>
                        <xsd:element ref="Event_Category"/>
                        <xsd:element ref="Event_Description"/>
                        <xsd:element ref="Event_Name"/>
                        <xsd:element ref="Event_S_D_Flag" minOccurs="0"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
```

```
ref="Source_Element_Existing"/>
                        <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
                    <xsd:choice maxOccurs="unbounded">
                        <xsd:element
name="State_Transition_Addition"
type="State_Transition_AdditionType"/>
                        <xsd:element
ref="State_Transition_Existing"/>
                        <xsd:element
name="State_Transition_Replacement"
type="State_Transition_ReplacementType"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                        name="Event_Description"
type="FDMS_Model_Event_Description_Type"/>
        <xsd:complexType name="Event_ExistingType">
                <xsd:sequence>
                    <xsd:element ref="Event_RDS_ID"/>
                    <xsd:element ref="Event_SDS_ID" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Event_InternalType">
                <xsd:sequence>
```

```
                    <xsd:element ref="Event_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Event_Name" type="FDMS_Model_Name_Type"/>
        <xsd:element name="Event_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Event_ReplacementType">
                <xsd:sequence>
                    <xsd:element ref="Event_RDS_ID"/>
                    <xsd:element ref="Event_SDS_ID" minOccurs="0"/>
                    <xsd:element                    name="Event_Data"
type="Event_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Event_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element name="Event_S_D_Flag" type="FDMS_S_D_Flag_Type"/>
        <xsd:complexType name="Event_UpdateType">
                <xsd:sequence>
                    <xsd:element ref="Event_RDS_ID"/>
                    <xsd:element ref="Event_SDS_ID" minOccurs="0"/>
                    <xsd:element                    name="Event_Data"
type="Event_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Information_Field_AdditionType">
                <xsd:sequence>
                    <xsd:element ref="Information_Field_SDS_ID"/>
```

```
                        <xsd:element       name="Information_Field_Data"
type="Information_Field_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Information_Field_CorrespondenceType">
                <xsd:choice>
                        <xsd:element   name="Information_Field_Existing"
type="Information_Field_ExistingType"/>
                        <xsd:element   name="Information_Field_Internal"
type="Information_Field_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="Information_Field_DataType">
                <xsd:sequence>
                        <xsd:element ref="Information_Field_Name"/>
                        <xsd:element ref="Information_Field_Data_Type"/>
                        <xsd:element
ref="Information_Field_Unit_Of_Measure" minOccurs="0"/>
                        <xsd:element
ref="Information_Field_Maximum_Value" minOccurs="0"/>
                        <xsd:element
ref="Information_Field_Minimum_Value" minOccurs="0"/>
                        <xsd:element ref="Information_Field_Optional"/>
                        <xsd:element
ref="Information_Field_Repeatable"/>
                        <xsd:element
```

```
name="Information_Field_Correspondence"
type="Information_Field_CorrespondenceType"              minOccurs="0"
maxOccurs="unbounded"/>
                    <xsd:element
name="Characteristic_Field_Correspondence"
type="Characteristic_Field_CorrespondenceType"           minOccurs="0"
maxOccurs="unbounded"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element                name="Information_Field_Data_Type"
type="FDMS_Model_Data_Type_Type"/>
       <xsd:complexType name="Information_Field_ExistingType">
              <xsd:sequence>
                    <xsd:element ref="Information_Field_RDS_ID"/>
                    <xsd:element      ref="Information_Field_SDS_ID"
minOccurs="0"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="Information_Field_InternalType">
              <xsd:sequence>
                    <xsd:element ref="Information_Field_SDS_ID"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element            name="Information_Field_Maximum_Value"
type="FDMS_Model_Information_Field_Value_Type"/>
       <xsd:element            name="Information_Field_Minimum_Value"
```

```
type="FDMS_Model_Information_Field_Value_Type"/>
        <xsd:element                    name="Information_Field_Name"
type="FDMS_Model_Information_Field_Name_Type"/>
        <xsd:element                name="Information_Field_Optional"
type="FDMS_Model_Information_Field_Repeatable_Type"/>
        <xsd:element                    name="Information_Field_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                name="Information_Field_Repeatable"
type="FDMS_Model_Information_Field_Repeatable_Type"/>
        <xsd:complexType name="Information_Field_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Information_Field_RDS_ID"/>
                        <xsd:element    ref="Information_Field_SDS_ID"
minOccurs="0"/>
                        <xsd:element       name="Information_Field_Data"
type="Information_Field_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Information_Field_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element          name="Information_Field_Unit_Of_Measure"
type="FDMS_Model_Unit_Type"/>
        <xsd:complexType name="Instance_Data_Set_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_SDS_ID"/>
                        <xsd:element       name="Instance_Data_Set_Data"
```

```
type="Instance_Data_Set_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Instance_Data_Set_DataType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_Name"/>
                        <xsd:element    ref="Instance_Data_Set_S_D_Flag"
minOccurs="0"/>
                        <xsd:element
name="Entity_Characteristic_Instance_Data"
type="Entity_Characteristic_Instance_DataType"
maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Instance_Data_Set_ExistingType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_RDS_ID"/>
                        <xsd:element      ref="Instance_Data_Set_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Instance_Data_Set_ForwardType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_SDS_ID"/>
                        <xsd:element ref="Instance_Data_Set_Name"/>
                </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:complexType name="Instance_Data_Set_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                        name="Instance_Data_Set_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element                      name="Instance_Data_Set_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Instance_Data_Set_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_RDS_ID"/>
                        <xsd:element      ref="Instance_Data_Set_SDS_ID"
minOccurs="0"/>
                        <xsd:element       name="Instance_Data_Set_Data"
type="Instance_Data_Set_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                      name="Instance_Data_Set_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                    name="Instance_Data_Set_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:complexType name="Instance_Data_Set_UpdateType">
                <xsd:sequence>
                        <xsd:element ref="Instance_Data_Set_RDS_ID"/>
```

```
                    <xsd:element      ref="Instance_Data_Set_SDS_ID"
minOccurs="0"/>
                    <xsd:element       name="Instance_Data_Set_Data"
type="Instance_Data_Set_DataType"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="Interaction_AdditionType">
              <xsd:sequence>
                    <xsd:element ref="Interaction_SDS_ID"/>
                    <xsd:element             name="Interaction_Data"
type="Interaction_DataType"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element                        name="Interaction_Category"
type="FDMS_Model_Interaction_Category_Type"/>
       <xsd:complexType name="Interaction_DataType">
              <xsd:sequence>
                    <xsd:choice minOccurs="0">
                          <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                          <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0">
```

```
                        <xsd:element       name="Entity_Existing"
type="Entity_ExistingType"/>
                        <xsd:element       name="Entity_Internal"
type="Entity_InternalType"/>
                    </xsd:choice>
                    <xsd:element ref="Interaction_Category"/>
                    <xsd:element ref="Interaction_Quantity"/>
                    <xsd:element        ref="Interaction_Stereotype"
minOccurs="0"/>
                    <xsd:element              ref="Interaction_Type"
minOccurs="0"/>
                    <xsd:element          ref="Interaction_S_D_Flag"
minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element      name="Process_Existing"
type="Process_ExistingType"/>
                        <xsd:element      name="Process_Internal"
type="Process_InternalType"/>
                    </xsd:choice>
                    <xsd:choice>
                        <xsd:element      name="Process_Existing"
type="Process_ExistingType"/>
                        <xsd:element      name="Process_Internal"
type="Process_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
                        <xsd:element
ref="Source_Element_Existing"/>
                        <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                        name="Interaction_Quantity"
type="FDMS_Model_Interaction_Quantity_Type"/>
        <xsd:element name="Interaction_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Interaction_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Interaction_RDS_ID"/>
                        <xsd:element            ref="Interaction_SDS_ID"
minOccurs="0"/>
                        <xsd:element             name="Interaction_Data"
type="Interaction_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Interaction_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element                        name="Interaction_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                      name="Interaction_Stereotype"
type="FDMS_Model_Interaction_Stereotype_Type"/>
        <xsd:element                             name="Interaction_Type"
type="FDMS_Model_Type_Type"/>
```

```
        <xsd:complexType name="Measure_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Measure_SDS_ID"/>
                        <xsd:element                    name="Measure_Data"
type="Measure_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Measure_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                        <xsd:element ref="Measure_Name"/>
                        <xsd:element             ref="Measure_Stereotype"
minOccurs="0"/>
                        <xsd:element ref="Measure_Type" minOccurs="0"/>
                        <xsd:element       ref="Measure_UJTL_Identifier"
minOccurs="0"/>
                        <xsd:element              ref="Measure_UJTL_Name"
minOccurs="0"/>
                        <xsd:element               ref="Measure_S_D_Flag"
```

```
minOccurs="0"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Source_Element_Existing"/>
                                <xsd:element
ref="Source_Element_Internal"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Measure_ExistingType">
                <xsd:sequence>
                        <xsd:element ref="Measure_RDS_ID"/>
                        <xsd:element ref="Measure_SDS_ID" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Measure_ForwardType">
                <xsd:sequence>
                        <xsd:element ref="Measure_SDS_ID"/>
                        <xsd:element ref="Measure_Name"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Measure_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Measure_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
```

```
        <xsd:element name="Measure_Name" type="FDMS_Model_Name_Type"/>
        <xsd:element name="Measure_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Measure_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Measure_RDS_ID"/>
                        <xsd:element ref="Measure_SDS_ID" minOccurs="0"/>
                        <xsd:element                    name="Measure_Data"
type="Measure_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Measure_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element                              name="Measure_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                            name="Measure_Stereotype"
type="FDMS_Model_Measure_Stereotype_Type"/>
        <xsd:element name="Measure_Type" type="FDMS_Model_Type_Type"/>
        <xsd:element                       name="Measure_UJTL_Identifier"
type="FDMS_Model_UJTL_Identifier_Type"/>
        <xsd:element                              name="Measure_UJTL_Name"
type="FDMS_Model_UJTL_Name_Type"/>
        <xsd:element name="Model_Representation">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:choice                 minOccurs="0"
maxOccurs="unbounded">
                                        <xsd:element
```

```
name="Association_Addition" type="Association_AdditionType"/>
                                    <xsd:element
name="Association_Replacement" type="Association_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Characteristic_Addition" type="Characteristic_AdditionType"/>
                                    <xsd:element
name="Characteristic_Forward" type="Characteristic_ForwardType"/>
                                    <xsd:element
name="Characteristic_Replacement"
type="Characteristic_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Condition_Addition" type="Condition_AdditionType"/>
                                    <xsd:element
name="Condition_Forward" type="Condition_ForwardType"/>
                                    <xsd:element
name="Condition_Replacement" type="Condition_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
```

```
name="Description_Link_Addition"
type="Description_Link_AdditionType"/>
                                    <xsd:element
name="Description_Link_Replacement"
type="Description_Link_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Text_Description_Addition"
type="Text_Description_AdditionType"/>
                                    <xsd:element
name="Text_Description_Replacement"
type="Text_Description_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Embedded_Object_Addition" type="Embedded_Object_AdditionType"/>
                                    <xsd:element
name="Embedded_Object_Replacement"
type="Embedded_Object_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
```

```
name="Entity_Addition" type="Entity_AdditionType"/>
                                    <xsd:element name="Entity_Forward"
type="Entity_ForwardType"/>
                                    <xsd:element
name="Entity_Replacement" type="Entity_ReplacementType"/>
                                    <xsd:element  name="Entity_Update"
type="Entity_UpdateType"/>
                              </xsd:choice>
                              <xsd:choice                  minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element name="Event_Addition"
type="Event_AdditionType"/>
                                    <xsd:element
name="Event_Replacement" type="Event_ReplacementType"/>
                                    <xsd:element   name="Event_Update"
type="Event_UpdateType"/>
                              </xsd:choice>
                              <xsd:choice                  minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Instance_Data_Set_Addition"
type="Instance_Data_Set_AdditionType"/>
                                    <xsd:element
name="Instance_Data_Set_Forward"
type="Instance_Data_Set_ForwardType"/>
                                    <xsd:element
```

```
name="Instance_Data_Set_Replacement"
type="Instance_Data_Set_ReplacementType"/>
                                    <xsd:element
name="Instance_Data_Set_Update" type="Instance_Data_Set_UpdateType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Interaction_Addition" type="Interaction_AdditionType"/>
                                    <xsd:element
name="Interaction_Replacement" type="Interaction_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Measure_Addition" type="Measure_AdditionType"/>
                                    <xsd:element
name="Measure_Forward" type="Measure_ForwardType"/>
                                    <xsd:element
name="Measure_Replacement" type="Measure_ReplacementType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Process_Addition" type="Process_AdditionType"/>
                                    <xsd:element
```

```
name="Process_Forward" type="Process_ForwardType"/>
                                    <xsd:element
name="Process_Replacement" type="Process_ReplacementType"/>
                                    <xsd:element name="Process_Update"
type="Process_UpdateType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Process_Group_Addition" type="Process_Group_AdditionType"/>
                                    <xsd:element
name="Process_Group_Forward" type="Process_Group_ForwardType"/>
                                    <xsd:element
name="Process_Group_Replacement"
type="Process_Group_ReplacementType"/>
                                    <xsd:element
name="Process_Group_Update" type="Process_Group_UpdateType"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
name="Process_Group_Instance_Addition"
type="Process_Group_Instance_AdditionType"/>
                                    <xsd:element
name="Process_Group_Instance_Replacement"
type="Process_Group_Instance_ReplacementType"/>
```

```
                        </xsd:choice>
                        <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                            <xsd:element
name="Process_Group_Package_Addition"
type="Process_Group_Package_AdditionType"/>
                            <xsd:element
name="Process_Group_Package_Replacement"
type="Process_Group_Package_ReplacementType"/>
                        </xsd:choice>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element                          name="Process_Abstract"
type="FDMS_Model_Abstract_Type"/>
            <xsd:complexType name="Process_AdditionType">
                <xsd:sequence>
                    <xsd:element ref="Process_SDS_ID"/>
                    <xsd:element                  name="Process_Data"
type="Process_DataType"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="Process_Control_AdditionType">
                <xsd:sequence>
                    <xsd:element ref="Process_Control_SDS_ID"/>
                    <xsd:element          name="Process_Control_Data"
```

```
type="Process_Control_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_DataType">
                <xsd:sequence>
                        <xsd:element  name="Process_Control_Element_LHS"
type="Process_Control_Element_LHSType"/>
                        <xsd:element  name="Process_Control_Element_RHS"
type="Process_Control_Element_RHSType" minOccurs="0"/>
                        <xsd:choice>
                                <xsd:element      name="Process_Existing"
type="Process_ExistingType"/>
                                <xsd:element      name="Process_Internal"
type="Process_InternalType"/>
                        </xsd:choice>
                        <xsd:element
name="Process_To_Start_on_Termination"
type="Process_To_Start_on_TerminationType" minOccurs="0"/>
                        <xsd:element          ref="Process_Control_Kind"
minOccurs="0"/>
                        <xsd:element          ref="Process_Control_Type"
minOccurs="0"/>
                        <xsd:element      ref="Process_Control_Relation"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
```

```
        <xsd:complexType name="Process_Control_ElementType">
                <xsd:choice>
                        <xsd:element
name="Process_Control_Element_Addition"
type="Process_Control_Element_AdditionType"/>
                        <xsd:element
name="Process_Control_Element_Replacement"
type="Process_Control_Element_ReplacementType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_AdditionType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Control_Element_SDS_ID"/>
                        <xsd:element name="Process_Control_Element_Data"
type="Process_Control_Element_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element            name="Process_Control_Element_Comments"
type="FDMS_Model_Process_Control_Element_Comments_Type"/>
        <xsd:element            name="Process_Control_Element_Constant"
type="FDMS_Model_Process_Control_Element_Constant_Type"/>
        <xsd:complexType name="Process_Control_Element_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element    name="Condition_Existing"
```

```
type="Condition_ExistingType"/>
                        <xsd:element    name="Condition_Internal"
type="Condition_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0">
                        <xsd:element      name="Measure_Existing"
type="Measure_ExistingType"/>
                        <xsd:element      name="Measure_Internal"
type="Measure_InternalType"/>
                    </xsd:choice>
                    <xsd:element
ref="Process_Control_Element_Constant" minOccurs="0"/>
                    <xsd:element
ref="Process_Control_Element_Operator" minOccurs="0"/>
                    <xsd:element
ref="Process_Control_Element_Comments" minOccurs="0"/>
                    <xsd:element name="Process_Control_Element_Next"
type="Process_Control_Element_NextType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_ExistingType">
            <xsd:sequence>
                    <xsd:element
ref="Process_Control_Element_RDS_ID"/>
                    <xsd:element
ref="Process_Control_Element_SDS_ID" minOccurs="0"/>
```

```
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_InternalType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Control_Element_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_LHSType">
                <xsd:sequence>
                        <xsd:element      name="Process_Control_Element"
type="Process_Control_ElementType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_NextType">
                <xsd:sequence>
                        <xsd:element
name="Process_Control_Element_Reference"
type="Process_Control_Element_ReferenceType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element           name="Process_Control_Element_Operator"
type="FDMS_Model_Operator_Type"/>
        <xsd:element             name="Process_Control_Element_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Control_Element_RHSType">
```

```
                <xsd:sequence>
                        <xsd:element      name="Process_Control_Element"
type="Process_Control_ElementType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Control_Element_ReferenceType">
                <xsd:choice>
                        <xsd:element
name="Process_Control_Element_Existing"
type="Process_Control_Element_ExistingType"/>
                        <xsd:element
name="Process_Control_Element_Internal"
type="Process_Control_Element_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType
name="Process_Control_Element_ReplacementType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Control_Element_RDS_ID"/>
                        <xsd:element
ref="Process_Control_Element_SDS_ID" minOccurs="0"/>
                        <xsd:element name="Process_Control_Element_Data"
type="Process_Control_Element_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
```

```
        <xsd:element              name="Process_Control_Element_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Control_ExistingType">
                <xsd:sequence>
                        <xsd:element ref="Process_Control_RDS_ID"/>
                        <xsd:element         ref="Process_Control_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                          name="Process_Control_Kind"
type="FDMS_Model_Process_Control_Kind_Type"/>
        <xsd:element                        name="Process_Control_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                      name="Process_Control_Relation"
type="FDMS_Model_Relation_Type"/>
        <xsd:complexType name="Process_Control_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Process_Control_RDS_ID"/>
                        <xsd:element         ref="Process_Control_SDS_ID"
minOccurs="0"/>
                        <xsd:element          name="Process_Control_Data"
type="Process_Control_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                        name="Process_Control_SDS_ID"
type="FDMS_DS_ID_Type"/>
```

```
        <xsd:element                          name="Process_Control_Type"
type="FDMS_Model_Process_Control_Type_Type"/>
        <xsd:complexType name="Process_DataType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                        <xsd:element                 name="Entity_Object"
type="Entity_ObjectType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element            name="Entity_Performing"
type="Entity_PerformingType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element                  name="Process_Kind"
type="Process_KindType" minOccurs="0"/>
                        <xsd:element ref="Process_Name"/>
                        <xsd:element            ref="Process_Stereotype"
minOccurs="0"/>
                        <xsd:element ref="Process_Type" minOccurs="0"/>
                        <xsd:element              ref="Process_Abstract"
minOccurs="0"/>
                        <xsd:element              ref="Process_S_D_Flag"
minOccurs="0"/>
```

```
                    <xsd:choice minOccurs="0">
                        <xsd:element
name="Process_Group_Existing" type="Process_Group_ExistingType"/>
                        <xsd:element
name="Process_Group_Internal" type="Process_Group_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Source_Element_Existing"/>
                        <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element        name="Event_Existing"
type="Event_ExistingType"/>
                        <xsd:element        name="Event_Internal"
type="Event_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Taxonomy_Keyword_Element_Existing"/>
                        <xsd:element
ref="Taxonomy_Keyword_Element_Internal"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
```

```
ref="Native_Library_Element_Existing"/>
                        <xsd:element
ref="Native_Library_Element_Internal"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_ExistingType">
                <xsd:sequence>
                    <xsd:element ref="Process_RDS_ID"/>
                    <xsd:element ref="Process_SDS_ID" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_ForwardType">
                <xsd:sequence>
                    <xsd:element ref="Process_SDS_ID"/>
                    <xsd:element ref="Process_Name"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Process_Group_Abstract"
type="FDMS_Model_Abstract_Type"/>
        <xsd:complexType name="Process_Group_AdditionType">
                <xsd:sequence>
                    <xsd:element ref="Process_Group_SDS_ID"/>
                    <xsd:element          name="Process_Group_Data"
type="Process_Group_DataType"/>
                </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Condition_AdditionType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Condition_SDS_ID"/>
                        <xsd:element name="Process_Group_Condition_Data"
type="Process_Group_Condition_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Condition_DataType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element    name="Condition_Existing"
type="Condition_ExistingType"/>
                                <xsd:element    name="Condition_Internal"
type="Condition_InternalType"/>
                        </xsd:choice>
                        <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType" minOccurs="0"/>
                        <xsd:element
ref="Process_Group_Condition_Name"/>
                        <xsd:element
ref="Process_Group_Condition_Relation" minOccurs="0"/>
                        <xsd:element
ref="Process_Group_Condition_Standard" minOccurs="0"/>
```

```
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Condition_ExistingType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Condition_RDS_ID"/>
                        <xsd:element
ref="Process_Group_Condition_SDS_ID" minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                name="Process_Group_Condition_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element              name="Process_Group_Condition_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element            name="Process_Group_Condition_Relation"
type="FDMS_Model_Relation_Type"/>
        <xsd:complexType
name="Process_Group_Condition_ReplacementType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Condition_RDS_ID"/>
                        <xsd:element
ref="Process_Group_Condition_SDS_ID" minOccurs="0"/>
                        <xsd:element name="Process_Group_Condition_Data"
type="Process_Group_Condition_DataType"/>
                </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:element              name="Process_Group_Condition_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element           name="Process_Group_Condition_Standard"
type="FDMS_Model_Process_Group_Condition_Standard_Type"/>
        <xsd:complexType name="Process_Group_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                        <xsd:element           name="Entity_Responsible"
type="Entity_ResponsibleType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element           name="Entity_Stimulating"
type="Entity_StimulatingType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="Process_Group_Name"/>
                        <xsd:element      ref="Process_Group_Stereotype"
minOccurs="0"/>
                        <xsd:element            ref="Process_Group_Type"
minOccurs="0"/>
                        <xsd:element        ref="Process_Group_Abstract"
minOccurs="0"/>
```

```
                    <xsd:element        ref="Process_Group_S_D_Flag"
minOccurs="0"/>
                    <xsd:choice minOccurs="0">
                        <xsd:element
name="Process_Group_Package_Existing"
type="Process_Group_Package_ExistingType"/>
                        <xsd:element
name="Process_Group_Package_Internal"
type="Process_Group_Package_InternalType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Source_Element_Existing"/>
                        <xsd:element
ref="Source_Element_Internal"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
name="Process_Group_Condition_Addition"
type="Process_Group_Condition_AdditionType"/>
                        <xsd:element
name="Process_Group_Condition_Existing"
type="Process_Group_Condition_ExistingType"/>
                        <xsd:element
name="Process_Group_Condition_Replacement"
type="Process_Group_Condition_ReplacementType"/>
```

```
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
name="Process_Control_Addition" type="Process_Control_AdditionType"/>
                        <xsd:element
name="Process_Control_Existing" type="Process_Control_ExistingType"/>
                        <xsd:element
name="Process_Control_Replacement"
type="Process_Control_ReplacementType"/>
                    </xsd:choice>
                    <xsd:choice maxOccurs="unbounded">
                        <xsd:element
name="Process_Sequence_Addition"
type="Process_Sequence_AdditionType"/>
                        <xsd:element
name="Process_Sequence_Existing"
type="Process_Sequence_ExistingType"/>
                        <xsd:element
name="Process_Sequence_Replacement"
type="Process_Sequence_ReplacementType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
name="Process_Group_Measure_Addition"
type="Process_Group_Measure_AdditionType"/>
                        <xsd:element
```

```
name="Process_Group_Measure_Existing"
type="Process_Group_Measure_ExistingType"/>
                        <xsd:element
name="Process_Group_Measure_Replacement"
type="Process_Group_Measure_ReplacementType"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Taxonomy_Keyword_Element_Existing"/>
                        <xsd:element
ref="Taxonomy_Keyword_Element_Internal"/>
                    </xsd:choice>
                    <xsd:choice minOccurs="0" maxOccurs="unbounded">
                        <xsd:element
ref="Native_Library_Element_Existing"/>
                        <xsd:element
ref="Native_Library_Element_Internal"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="Process_Group_ExistingType">
                <xsd:sequence>
                    <xsd:element ref="Process_Group_RDS_ID"/>
                    <xsd:element          ref="Process_Group_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:complexType name="Process_Group_ForwardType">
                <xsd:sequence>
                        <xsd:element ref="Process_Group_SDS_ID"/>
                        <xsd:element ref="Process_Group_Name"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Instance_AdditionType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Instance_SDS_ID"/>
                        <xsd:element  name="Process_Group_Instance_Data"
type="Process_Group_Instance_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Instance_ConditionType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element    name="Condition_Existing"
type="Condition_ExistingType"/>
                                <xsd:element    name="Condition_Internal"
type="Condition_InternalType"/>
                        </xsd:choice>
                        <xsd:element
ref="Process_Group_Instance_Condition_Value"/>
                </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:element     name="Process_Group_Instance_Condition_Value"
type="FDMS_Model_Process_Group_Instance_Value_Type"/>
        <xsd:complexType name="Process_Group_Instance_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Description_Link_Existing"
type="Description_Link_ExistingType"/>
                                <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                        <xsd:choice>
                                <xsd:element
name="Process_Group_Existing" type="Process_Group_ExistingType"/>
                                <xsd:element
name="Process_Group_Internal" type="Process_Group_InternalType"/>
                        </xsd:choice>
                        <xsd:element ref="Process_Group_Instance_Name"/>
                        <xsd:element
ref="Process_Group_Instance_Stereotype" minOccurs="0"/>
                        <xsd:element   ref="Process_Group_Instance_Type"
minOccurs="0"/>
                        <xsd:element
ref="Process_Group_Instance_S_D_Flag" minOccurs="0"/>
```

```
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Source_Element_Existing"/>
                                <xsd:element
ref="Source_Element_Internal"/>
                        </xsd:choice>
                        <xsd:element
name="Process_Group_Instance_Condition"
type="Process_Group_Instance_ConditionType"             minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element
name="Process_Group_Instance_Standard"
type="Process_Group_Instance_StandardType"              minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
name="Instance_Data_Set_Existing"
type="Instance_Data_Set_ExistingType"/>
                                <xsd:element
name="Instance_Data_Set_Internal"
type="Instance_Data_Set_InternalType"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Native_Library_Element_Existing"/>
                                <xsd:element
```

```
ref="Native_Library_Element_Internal"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                name="Process_Group_Instance_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element              name="Process_Group_Instance_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType
name="Process_Group_Instance_ReplacementType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Instance_RDS_ID"/>
                        <xsd:element ref="Process_Group_Instance_SDS_ID"
minOccurs="0"/>
                        <xsd:element  name="Process_Group_Instance_Data"
type="Process_Group_Instance_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element              name="Process_Group_Instance_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element            name="Process_Group_Instance_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:complexType name="Process_Group_Instance_StandardType">
                <xsd:sequence>
                        <xsd:choice>
```

```
                        <xsd:element      name="Measure_Existing"
type="Measure_ExistingType"/>
                        <xsd:element      name="Measure_Internal"
type="Measure_InternalType"/>
                    </xsd:choice>
                    <xsd:element
ref="Process_Group_Instance_Standard_Value"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element      name="Process_Group_Instance_Standard_Value"
type="FDMS_Model_Process_Group_Instance_Value_Type"/>
        <xsd:element          name="Process_Group_Instance_Stereotype"
type="FDMS_Model_Process_Group_Instance_Stereotype_Type"/>
        <xsd:element                name="Process_Group_Instance_Type"
type="FDMS_Model_Process_Group_Instance_Type_Type"/>
        <xsd:complexType name="Process_Group_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Process_Group_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Measure_AdditionType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Measure_SDS_ID"/>
                    <xsd:element   name="Process_Group_Measure_Data"
type="Process_Group_Measure_DataType"/>
```

```
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Measure_DataType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:element      name="Measure_Existing"
type="Measure_ExistingType"/>
                                <xsd:element      name="Measure_Internal"
type="Measure_InternalType"/>
                        </xsd:choice>
                        <xsd:element  name="Entity_Characteristic_First"
type="Entity_Characteristic_FirstType" minOccurs="0"/>
                        <xsd:element name="Entity_Characteristic_Second"
type="Entity_Characteristic_SecondType" minOccurs="0"/>
                        <xsd:element
ref="Process_Group_Measure_Operator" minOccurs="0"/>
                        <xsd:element
ref="Process_Group_Measure_Unit_Of_Measure"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Measure_ExistingType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Measure_RDS_ID"/>
                        <xsd:element  ref="Process_Group_Measure_SDS_ID"
minOccurs="0"/>
```

```
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element              name="Process_Group_Measure_Operator"
type="FDMS_Model_Operator_Type"/>
        <xsd:element                name="Process_Group_Measure_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Group_Measure_ReplacementType">
            <xsd:sequence>
                <xsd:element
ref="Process_Group_Measure_RDS_ID"/>
                <xsd:element  ref="Process_Group_Measure_SDS_ID"
minOccurs="0"/>
                <xsd:element   name="Process_Group_Measure_Data"
type="Process_Group_Measure_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element                name="Process_Group_Measure_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element      name="Process_Group_Measure_Unit_Of_Measure"
type="FDMS_Model_Unit_Type"/>
        <xsd:element                            name="Process_Group_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:complexType name="Process_Group_Package_AdditionType">
            <xsd:sequence>
                <xsd:element
ref="Process_Group_Package_SDS_ID"/>
```

```
                    <xsd:element    name="Process_Group_Package_Data"
type="Process_Group_Package_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Package_DataType">
                <xsd:sequence>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Process_Group_Package_Existing"
type="Process_Group_Package_ExistingType"/>
                                <xsd:element
name="Process_Group_Package_Internal"
type="Process_Group_Package_InternalType"/>
                        </xsd:choice>
                        <xsd:element ref="Process_Group_Package_Name"/>
                        <xsd:element
ref="Process_Group_Package_S_D_Flag" minOccurs="0"/>
                        <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <xsd:element
ref="Source_Element_Existing"/>
                                <xsd:element
ref="Source_Element_Internal"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0">
                                <xsd:element
name="Description_Link_Existing"
```

```
type="Description_Link_ExistingType"/>
                              <xsd:element
name="Description_Link_Internal"
type="Description_Link_InternalType"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Package_ExistingType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Package_RDS_ID"/>
                        <xsd:element  ref="Process_Group_Package_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Group_Package_InternalType">
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Package_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Process_Group_Package_Name"
type="FDMS_Model_Name_Type"/>
        <xsd:element                  name="Process_Group_Package_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Group_Package_ReplacementType">
```

```
                <xsd:sequence>
                        <xsd:element
ref="Process_Group_Package_RDS_ID"/>
                        <xsd:element  ref="Process_Group_Package_SDS_ID"
minOccurs="0"/>
                        <xsd:element   name="Process_Group_Package_Data"
type="Process_Group_Package_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                name="Process_Group_Package_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element              name="Process_Group_Package_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                          name="Process_Group_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Group_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Process_Group_RDS_ID"/>
                        <xsd:element          ref="Process_Group_SDS_ID"
minOccurs="0"/>
                        <xsd:element           name="Process_Group_Data"
type="Process_Group_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                          name="Process_Group_SDS_ID"
type="FDMS_DS_ID_Type"/>
```

```
        <xsd:element                    name="Process_Group_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:element                  name="Process_Group_Stereotype"
type="FDMS_Model_Process_Group_Stereotype_Type"/>
        <xsd:element                       name="Process_Group_Type"
type="FDMS_Model_Process_Group_Type_Type"/>
        <xsd:complexType name="Process_Group_UpdateType">
                <xsd:sequence>
                        <xsd:element ref="Process_Group_RDS_ID"/>
                        <xsd:element          ref="Process_Group_SDS_ID"
minOccurs="0"/>
                        <xsd:element           name="Process_Group_Data"
type="Process_Group_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Process_SDS_ID"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_KindType">
                <xsd:choice>
                        <xsd:element             name="Process_Existing"
type="Process_ExistingType"/>
                        <xsd:element             name="Process_Internal"
type="Process_InternalType"/>
```

```
            </xsd:choice>
        </xsd:complexType>
        <xsd:element name="Process_Name" type="FDMS_Model_Name_Type"/>
        <xsd:element name="Process_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_ReplacementType">
            <xsd:sequence>
                <xsd:element ref="Process_RDS_ID"/>
                <xsd:element ref="Process_SDS_ID" minOccurs="0"/>
                <xsd:element name="Process_Data"
type="Process_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Process_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element name="Process_S_D_Flag"
type="FDMS_S_D_Flag_Type"/>
        <xsd:complexType name="Process_Sequence_AdditionType">
            <xsd:sequence>
                <xsd:element ref="Process_Sequence_SDS_ID"/>
                <xsd:element name="Process_Sequence_Data"
type="Process_Sequence_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Process_Sequence_DataType">
            <xsd:sequence>
                <xsd:element name="Sequence_Element"
type="Sequence_ElementType"/>
```

```
                    <xsd:element ref="Process_Sequence_Part"/>
                    <xsd:element
ref="Process_Sequence_Element_Cardinality"/>
                    <xsd:element          ref="Process_Sequence_Tag"
minOccurs="0"/>
                    <xsd:sequence minOccurs="0">
                          <xsd:element      name="Sequence_Element"
type="Sequence_ElementType"/>
                          <xsd:element
ref="Process_Sequence_Part"/>
                          <xsd:element
ref="Process_Sequence_Element_Cardinality"/>
                          <xsd:element   ref="Process_Sequence_Tag"
minOccurs="0"/>
                    </xsd:sequence>
                    <xsd:element
ref="Process_Sequence_Iteration_Count" minOccurs="0"/>
              </xsd:sequence>
       </xsd:complexType>
       <xsd:element       name="Process_Sequence_Element_Cardinality"
type="FDMS_Model_Cardinality_Type"/>
       <xsd:complexType name="Process_Sequence_ExistingType">
              <xsd:sequence>
                    <xsd:element ref="Process_Sequence_RDS_ID"/>
                    <xsd:element       ref="Process_Sequence_SDS_ID"
minOccurs="0"/>
```

```
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element           name="Process_Sequence_Iteration_Count"
type="FDMS_Model_Process_Sequence_Iteration_Count_Type"/>
        <xsd:element                        name="Process_Sequence_Part"
type="FDMS_Model_Process_Sequence_Part_Type"/>
        <xsd:element                      name="Process_Sequence_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Process_Sequence_ReplacementType">
            <xsd:sequence>
                <xsd:element ref="Process_Sequence_RDS_ID"/>
                <xsd:element       ref="Process_Sequence_SDS_ID"
minOccurs="0"/>
                <xsd:element        name="Process_Sequence_Data"
type="Process_Sequence_DataType"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element                      name="Process_Sequence_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element                        name="Process_Sequence_Tag"
type="FDMS_Model_Process_Sequence_Tag_Type"/>
        <xsd:element                          name="Process_Stereotype"
type="FDMS_Model_Process_Stereotype_Type"/>
        <xsd:complexType name="Process_To_Start_on_TerminationType">
            <xsd:choice>
                <xsd:element             name="Process_Existing"
```

```
type="Process_ExistingType"/>
                        <xsd:element              name="Process_Internal"
type="Process_InternalType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:element                                    name="Process_Type"
type="FDMS_Model_Process_Type_Type"/>
        <xsd:complexType name="Process_UpdateType">
                <xsd:sequence>
                        <xsd:element ref="Process_RDS_ID"/>
                        <xsd:element ref="Process_SDS_ID" minOccurs="0"/>
                        <xsd:element                    name="Process_Data"
type="Process_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="SelectionType">
                <xsd:sequence>
                        <xsd:choice>
                                <xsd:sequence>
                                        <xsd:element
ref="Selection_RDS_ID"/>
                                        <xsd:element
ref="Selection_SDS_ID"/>
                                </xsd:sequence>
                                <xsd:element ref="Selection_RDS_ID"/>
                                <xsd:element ref="Selection_SDS_ID"/>
```

```
                    </xsd:choice>
                    <xsd:element ref="Selection_Type"/>
                    <xsd:choice>
                        <xsd:element      name="Process_Existing"
type="Process_ExistingType"/>
                        <xsd:element      name="Process_Internal"
type="Process_InternalType"/>
                    </xsd:choice>
                    <xsd:choice maxOccurs="unbounded">
                        <xsd:element      name="Process_Existing"
type="Process_ExistingType"/>
                        <xsd:element      name="Process_Internal"
type="Process_InternalType"/>
                    </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Selection_RDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element name="Selection_SDS_ID" type="FDMS_DS_ID_Type"/>
        <xsd:element                          name="Selection_Type"
type="FDMS_Model_Selection_Type_Type"/>
        <xsd:complexType name="Sequence_ElementType">
            <xsd:choice>
                <xsd:element          name="Process_Existing"
type="Process_ExistingType"/>
                <xsd:element          name="Process_Internal"
type="Process_InternalType"/>
```

```
                        <xsd:choice>
                                <xsd:element
name="Process_Group_Existing" type="Process_Group_ExistingType"/>
                                <xsd:element
name="Process_Group_Internal" type="Process_Group_InternalType"/>
                        </xsd:choice>
                        <xsd:element                        name="Selection"
type="SelectionType"/>
                </xsd:choice>
        </xsd:complexType>
        <xsd:complexType name="State_Transition_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="State_Transition_SDS_ID"/>
                        <xsd:element        name="State_Transition_Data"
type="State_Transition_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="State_Transition_DataType">
                <xsd:sequence>
                        <xsd:element
name="Entity_Characteristic_Reference"
type="Entity_Characteristic_ReferenceType"/>
                        <xsd:choice>
                                <xsd:sequence>
                                        <xsd:element
ref="State_Transition_Entry_Value" minOccurs="0"/>
```

```
                                    <xsd:element
ref="State_Transition_Exit_Value"/>
                                </xsd:sequence>
                                <xsd:element
ref="State_Transition_Increment"/>
                                <xsd:element
ref="State_Transition_Decrement"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="State_Transition_Decrement"
type="FDMS_Model_State_Transition_Incr_Decr_Type"/>
        <xsd:element                  name="State_Transition_Entry_Value"
type="FDMS_Model_State_Transition_Value_Type"/>
        <xsd:element name="State_Transition_Existing">
                <xsd:complexType/>
        </xsd:element>
        <xsd:element                   name="State_Transition_Exit_Value"
type="FDMS_Model_State_Transition_Value_Type"/>
        <xsd:element                    name="State_Transition_Increment"
type="FDMS_Model_State_Transition_Incr_Decr_Type"/>
        <xsd:element                       name="State_Transition_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="State_Transition_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="State_Transition_RDS_ID"/>
```

```
                    <xsd:element       ref="State_Transition_SDS_ID"
minOccurs="0"/>
                    <xsd:element        name="State_Transition_Data"
type="State_Transition_DataType"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="State_Transition_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Text_Description_AdditionType">
                <xsd:sequence>
                        <xsd:element ref="Text_Description_SDS_ID"/>
                        <xsd:element ref="Text_Description_Data"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                     name="Text_Description_Data"
type="FDMS_Model_Text_Description_Data_Type"/>
        <xsd:complexType name="Text_Description_ExistingType">
                <xsd:sequence>
                        <xsd:element ref="Text_Description_RDS_ID"/>
                        <xsd:element       ref="Text_Description_SDS_ID"
minOccurs="0"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Text_Description_InternalType">
                <xsd:sequence>
                        <xsd:element ref="Text_Description_SDS_ID"/>
```

```
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Text_Description_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:complexType name="Text_Description_ReplacementType">
                <xsd:sequence>
                        <xsd:element ref="Text_Description_RDS_ID"/>
                        <xsd:element       ref="Text_Description_SDS_ID"
minOccurs="0"/>
                        <xsd:element ref="Text_Description_Data"/>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:element                    name="Text_Description_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:simpleType name="FDMS_Model_Cardinality_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                to
FDMS_IntegerOrNull_Type</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_IntegerOrNull_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Type_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
```

```
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Name_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Abstract_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_1023_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_1023_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_UJTL_Identifier_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Short_Type3.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Short_Type3"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_UJTL_Name_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
```

```
FDMS_Long_Type2.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_Type2"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Unit_Type">
                <xsd:annotation>
                        <xsd:documentation>   "Hours"   or   "Minutes"or
"Seconds" or "Miles" or "Feet" or "Kilometers" or "Meters" or "Feet per
Second" or "Meters per Second" or "Percent Successful Defense" or "Percent
Successful Attack" or "Other"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Hours"/>
                        <xsd:enumeration value="Minutes"/>
                        <xsd:enumeration value="Seconds"/>
                        <xsd:enumeration value="Miles"/>
                        <xsd:enumeration value="Feet"/>
                        <xsd:enumeration value="Kilometers"/>
                        <xsd:enumeration value="Meters"/>
                        <xsd:enumeration value="Feet per Second"/>
                        <xsd:enumeration value="Meters per Second"/>
                        <xsd:enumeration    value="Percent    Successful
Defense"/>
                        <xsd:enumeration    value="Percent    Successful
Attack"/>
                        <xsd:enumeration value="Other"/>
```

```
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Data_Type_Type">
            <xsd:annotation>
                <xsd:documentation>"float" or "fixed.14.4" or
"integer" or "double" or "short" or "unsigned short" or "long" or "unsigned
long" or "long long" or "unsigned long long" or "char" or "boolean"
or "octet" or "any" or "string" or "sequence" or "dateTime.iso8601" or
"dateTime.iso8601.tz" or "date.iso8601" or "time.iso8601" or
"time.iso8601.tz" or "i1" or "i2" or "i4" or "i8" or "ui1" or "ui2" or
"ui4" or "ui8" or "r4" or "r8" or "float.IEEE.754.32 " or
"float.IEEE.754.64" or "uuid" or "uri" or "bin.hex" or
"string.ansi"</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="float"/>
                <xsd:enumeration value="fixed.14.4"/>
                <xsd:enumeration value="integer"/>
                <xsd:enumeration value="double"/>
                <xsd:enumeration value="short"/>
                <xsd:enumeration value="unsigned short"/>
                <xsd:enumeration value="long"/>
                <xsd:enumeration value="unsigned long"/>
                <xsd:enumeration value="long long"/>
                <xsd:enumeration value="unsigned long long"/>
                <xsd:enumeration value="char"/>
```

```
<xsd:enumeration value="boolean"/>
<xsd:enumeration value="octet"/>
<xsd:enumeration value="any"/>
<xsd:enumeration value="string"/>
<xsd:enumeration value="sequence"/>
<xsd:enumeration value="dateTime.iso8601"/>
<xsd:enumeration value="dateTime.iso8601.tz"/>
<xsd:enumeration value="date.iso8601"/>
<xsd:enumeration value="time.iso8601"/>
<xsd:enumeration value="time.iso8601.tz"/>
<xsd:enumeration value="i1"/>
<xsd:enumeration value="i2"/>
<xsd:enumeration value="i4"/>
<xsd:enumeration value="i8"/>
<xsd:enumeration value="ui1"/>
<xsd:enumeration value="ui2"/>
<xsd:enumeration value="ui4"/>
<xsd:enumeration value="ui8"/>
<xsd:enumeration value="r4"/>
<xsd:enumeration value="r8"/>
<xsd:enumeration value="float.IEEE.754.32"/>
<xsd:enumeration value="float.IEEE.754.64"/>
<xsd:enumeration value="uuid"/>
<xsd:enumeration value="uri"/>
<xsd:enumeration value="bin.hex"/>
<xsd:enumeration value="string.ansi"/>
```

```
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Relation_Type">
                <xsd:annotation>
                        <xsd:documentation>"Equals" or "Greater Than" or
"Greater Than or Equals" or "Less Than" or "Less Than or Equals" or "Not
Equal"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Equals"/>
                        <xsd:enumeration value="Greater Than"/>
                        <xsd:enumeration value="Greater Than or Equals"/>
                        <xsd:enumeration value="Less Than"/>
                        <xsd:enumeration value="Less Than or Equals"/>
                        <xsd:enumeration value="Not Equal"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Operator_Type">
                <xsd:annotation>
                        <xsd:documentation>"Add"    or    "Subtract"    or
"Multiply"    or    "Divide"    or    "Percent"    or    "Modulo"    or
"Negate"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Add"/>
                        <xsd:enumeration value="Subtract"/>
```

```
                        <xsd:enumeration value="Multiply"/>
                        <xsd:enumeration value="Divide"/>
                        <xsd:enumeration value="Percent"/>
                        <xsd:enumeration value="Modulo"/>
                        <xsd:enumeration value="Negate"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Association_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>        "Allocation"        or
"Collaboration"or  "Command"  or  "Communication"  or  "Control"  or
"Engagement Range" or "Line Of Sight" or "Line Of Support" or "Occupancy"
or "Occupation" or "Operation" or "Opposition" or "Organization" or
"Possession"  or  "Sensing  Range"  or  "Supply"  or  "Support"  or
"Other"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Allocation"/>
                        <xsd:enumeration value="Collaboration"/>
                        <xsd:enumeration value="Command"/>
                        <xsd:enumeration value="Communication"/>
                        <xsd:enumeration value="Control"/>
                        <xsd:enumeration value="Engagement Range"/>
                        <xsd:enumeration value="Line Of Sight"/>
                        <xsd:enumeration value="Line Of Support"/>
                        <xsd:enumeration value="Occupancy"/>
```

```
                        <xsd:enumeration value="Occupation"/>
                        <xsd:enumeration value="Operation"/>
                        <xsd:enumeration value="Opposition"/>
                        <xsd:enumeration value="Organization"/>
                        <xsd:enumeration value="Possession"/>
                        <xsd:enumeration value="Sensing Range"/>
                        <xsd:enumeration value="Supply"/>
                        <xsd:enumeration value="Support"/>
                        <xsd:enumeration value="Other"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Text_Description_Data_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_String65535_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_String65535_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Embedded_Object_Name_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_String255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_String255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Embedded_Object_Link_Type">
```

```
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_String255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_String255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Embedded_Object_Type_Type">
                <xsd:annotation>
                        <xsd:documentation> ".xls" or ".gif"or ".doc" or
".ppt" or ".bmp" or ".jpg" or text string (4)</xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String4_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value=".xls"/>
                                        <xsd:enumeration value=".gif"/>
                                        <xsd:enumeration value=".doc"/>
                                        <xsd:enumeration value=".ppt"/>
                                        <xsd:enumeration value=".bmp"/>
                                        <xsd:enumeration value=".jpg"/>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Entity_Stereotype_Type">
                <xsd:annotation>
```

```
                    <xsd:documentation>  "Agent"  or  "Equipment"or
"Facility" or "Feature" or "Information" or "Infrastructure" or "Network"
or    "Organization"    or    "Person"    or    "Supplies"    or
"Other"</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Agent"/>
                    <xsd:enumeration value="Equipment"/>
                    <xsd:enumeration value="Facility"/>
                    <xsd:enumeration value="Feature"/>
                    <xsd:enumeration value="Information"/>
                    <xsd:enumeration value="Infrastructure"/>
                    <xsd:enumeration value="Network"/>
                    <xsd:enumeration value="Organization"/>
                    <xsd:enumeration value="Person"/>
                    <xsd:enumeration value="Supplies"/>
                    <xsd:enumeration value="Other"/>
            </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType
name="FDMS_Model_Entity_Characteristic_Value_Type">
            <xsd:annotation>
                    <xsd:documentation>Equivalent                to
FDMS_Long_255_Type.</xsd:documentation>
            </xsd:annotation>
            <xsd:restriction base="FDMS_Long_255_Type"/>
```

```
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Entity_Characteristic_Instance_Quantity_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Event_Category_Type">
                <xsd:annotation>
                        <xsd:documentation>"Process  Begin"  or  "Process
End"or "Continuous"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Process Begin"/>
                        <xsd:enumeration value="Process End"/>
                        <xsd:enumeration value="Continuous"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Event_Description_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_1023_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_1023_Type"/>
```

```
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Information_Field_Value_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Information_Field_Name_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_String255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_String255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Information_Field_Optional_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Boolean_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Boolean_Type"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Information_Field_Repeatable_Type">
```

```
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Boolean_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Boolean_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_State_Transition_Value_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Short_Type3.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Short_Type3"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_State_Transition_Incr_Decr_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_IntegerOrNull_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_IntegerOrNull_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Interaction_Category_Type">
                <xsd:annotation>
                        <xsd:documentation>"Input"   or   "Input_Output"or
"Output"</xsd:documentation>
                </xsd:annotation>
```

```
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Input"/>
                        <xsd:enumeration value="Input_Output"/>
                        <xsd:enumeration value="Output"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Interaction_Quantity_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_IntegerOrNull_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_IntegerOrNull_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Interaction_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>"Assignment"  or  "Attack"  or
"Collision" or "Delegation" or "Land" or "Launch" or "Resupply" or
"Transfer" or "Transmission" or Text String(255) </xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String255_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration
value="Assignment"/>
                                        <xsd:enumeration value="Attack"/>
                                        <xsd:enumeration
```

```
value="Collision"/>
                                    <xsd:enumeration
value="Delegation"/>
                                    <xsd:enumeration value="Land"/>
                                    <xsd:enumeration value="Launch"/>
                                    <xsd:enumeration
value="Resupply"/>
                                    <xsd:enumeration
value="Transfer"/>
                                    <xsd:enumeration
value="Transmission"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Measure_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>"Essential     Element     Of
Analysis" or "Measure Of Effectiveness" or "Measure Of Performance"
</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration  value="Essential  Element  Of
Analysis"/>
                        <xsd:enumeration        value="Measure        Of
Effectiveness"/>
```

```
                        <xsd:enumeration value="Measure Of Performance"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action"  or  "Mission"  or
"Operation" or "Process" or "Task"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Action"/>
                        <xsd:enumeration value="Mission"/>
                        <xsd:enumeration value="Operation"/>
                        <xsd:enumeration value="Process"/>
                        <xsd:enumeration value="Task"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Type_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action"  or  "Activity"  or
"Mission" or "Operation" or "Phase" or "Process" or "Step" or "Subprocess"
or "Substep" or "Task" or Text String(255)  </xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String255_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="Action"/>
```

```
                                    <xsd:enumeration
value="Activity"/>
                                    <xsd:enumeration value="Mission"/>
                                    <xsd:enumeration
value="Operation"/>
                                    <xsd:enumeration value="Phase"/>
                                    <xsd:enumeration value="Process"/>
                                    <xsd:enumeration value="Step"/>
                                    <xsd:enumeration
value="Subprocess"/>
                                    <xsd:enumeration value="Substep"/>
                                    <xsd:enumeration value="Task"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Group_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action  Group"  or  "Mission
Group" or "Operation Group" or "Process Group" or "Task Group" or "Use
Case" or Text String(255) </xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String255_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
```

```
                                    <xsd:enumeration     value="Action
Group"/>
                                    <xsd:enumeration    value="Mission
Group"/>
                                    <xsd:enumeration  value="Operation
Group"/>
                                    <xsd:enumeration    value="Process
Group"/>
                                    <xsd:enumeration       value="Task
Group"/>
                                    <xsd:enumeration value="Use Case"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Group_Type_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action  Group"  or  "Activity
Group" or "Mission" or "Mission Group" or "Operation" or "Operation Group"
or "Phase Group" or "Process Group" or "Step Group" or "Subprocess Group"
or  "Substep  Group"  or  "Task  Group"  or   "Use  Case"  or  Text
String(255)</xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String255_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
```

```
                                    <xsd:enumeration      value="Action
Group"/>
                                    <xsd:enumeration    value="Activity
Group"/>
                                    <xsd:enumeration value="Mission"/>
                                    <xsd:enumeration     value="Mission
Group"/>
                                    <xsd:enumeration
value="Operation"/>
                                    <xsd:enumeration   value="Operation
Group"/>
                                    <xsd:enumeration       value="Phase
Group"/>
                                    <xsd:enumeration     value="Process
Group"/>
                                    <xsd:enumeration        value="Step
Group"/>
                                    <xsd:enumeration  value="Subprocess
Group"/>
                                    <xsd:enumeration     value="Substep
Group"/>
                                    <xsd:enumeration        value="Task
Group"/>
                                    <xsd:enumeration value="Use Case"/>
                                </xsd:restriction>
                            </xsd:simpleType>
```

```
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Group_Condition_Standard_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Control_Element_Constant_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Control_Element_Comments_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Long_1023_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_1023_Type"/>
        </xsd:simpleType>
```

```
        <xsd:simpleType name="FDMS_Model_Process_Control_Kind_Type">
                <xsd:annotation>
                        <xsd:documentation>"Complete" or "Continue" or
"Start" or "Terminate" or "Interrupt"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Complete"/>
                        <xsd:enumeration value="Continue"/>
                        <xsd:enumeration value="Start"/>
                        <xsd:enumeration value="Terminate"/>
                        <xsd:enumeration value="Interrupt"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Control_Type_Type">
                <xsd:annotation>
                        <xsd:documentation>"If"     or     "When"     or
"While"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="If"/>
                        <xsd:enumeration value="When"/>
                        <xsd:enumeration value="While"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Selection_Type_Type">
                <xsd:annotation>
```

```
                    <xsd:documentation>"AND"     or     "XOR"     or
"OR"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="AND"/>
                        <xsd:enumeration value="XOR"/>
                        <xsd:enumeration value="OR"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Sequence_Part_Type">
                <xsd:annotation>
                        <xsd:documentation>"Prior"or "Start" or  "During"
or "End" or "After" or "Sequence" or "Iteration" or "Concurrent" or
"Continuous"  or  "Unconstrained"  or  "N/A"  or  "UNK"  or  "TBD"
</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Prior"/>
                        <xsd:enumeration value="Start"/>
                        <xsd:enumeration value="During"/>
                        <xsd:enumeration value="End"/>
                        <xsd:enumeration value="After"/>
                        <xsd:enumeration value="Sequential"/>
                        <xsd:enumeration value="Iteration"/>
                        <xsd:enumeration value="Concurrent"/>
                        <xsd:enumeration value="Continuous"/>
```

```
                        <xsd:enumeration value="Unconstrained"/>
                        <xsd:enumeration value="N/A"/>
                        <xsd:enumeration value="UNK"/>
                        <xsd:enumeration value="TBD"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Model_Process_Sequence_Tag_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_Short_Type3.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Short_Type3"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Sequence_Iteration_Count_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                   to
FDMS_IntegerOrNull_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_IntegerOrNull_Type"/>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Group_Instance_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action_Group_Instance"     or
"Mission_Group_Instance"     or     "Operation_Group_Instance"     or
```

```
"Process_Group_Instance" or "Task_Group_Instance" or "Scenario" or
"Situation" or "Use_Case_Instance" or Text String(255)
</xsd:documentation>
            </xsd:annotation>
            <xsd:union memberTypes="FDMS_String255_Type">
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                              <xsd:enumeration
value="Action_Group_Instance"/>
                              <xsd:enumeration
value="Mission_Group_Instance"/>
                              <xsd:enumeration
value="Operation_Group_Instance"/>
                              <xsd:enumeration
value="Process_Group_Instance"/>
                              <xsd:enumeration
value="Task_Group_Instance"/>
                              <xsd:enumeration
value="Scenario"/>
                              <xsd:enumeration
value="Situation"/>
                              <xsd:enumeration
value="Use_Case_Instance"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:union>
```

```
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Group_Instance_Type_Type">
                <xsd:annotation>
                        <xsd:documentation>"Action Group Instance" or
"Mission Group Instance" or "Mission Instance" or "Operation Group
Instance" or "Operation Instance" or "Process Group Instance" or "Task
Group Instance" or "Scenario" or "Situation" or "Use_Case_Instance" or
Text String(255) </xsd:documentation>
                </xsd:annotation>
                <xsd:union memberTypes="FDMS_String255_Type">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration     value="Action
Group Instance"/>
                                        <xsd:enumeration    value="Mission
Group Instance"/>
                                        <xsd:enumeration    value="Mission
Instance"/>
                                        <xsd:enumeration  value="Operation
Group Instance"/>
                                        <xsd:enumeration  value="Operation
Instance"/>
                                        <xsd:enumeration    value="Process
Group Instance"/>
                                        <xsd:enumeration value="Task Group
```

```
Instance"/>
                                    <xsd:enumeration
value="Scenario"/>
                                    <xsd:enumeration
value="Situation"/>
                                    <xsd:enumeration
value="Use_Case_Instance"/>
                              </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
        <xsd:simpleType
name="FDMS_Model_Process_Group_Instance_Value_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Long_255_Type.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type"/>
        </xsd:simpleType>
</xsd:schema>
```

C-2 FDMS_Point_Of_Contact_Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
	<xsd:annotation>
		<xsd:documentation>This is the main schema for the Point
of Contact DIF. This schema describes the structure of the Point of
Contact DIF, specifying which elements are allowed, and how they can be
combined. In addition, datatypes are specifed for the data content of
the elements. Some of these datatypes are built-in types. Others are
defined in this schema - these are datatypes that are used only in the
Point of Contact. The remaining datatype definitions are in the
FDMS_Basic_Datatypes.xsd and FDMS_Strict_Date_Type.xsd files - these are
datatypes that are intended for use in all sections of the FDMS DIF. In
order to achieve complete validation, the instance document should also
be validated against the top-up Schematron schema -
FDMS_Point_Of_Contact_Topup.sch</xsd:documentation>
	</xsd:annotation>
	<xsd:include
schemaLocation="..¥..¥FDMS_Common_Files¥FDMS_Basic_Datatypes.xsd"/>
	<xsd:include
schemaLocation="..¥..¥FDMS_Common_Files¥FDMS_Strict_Date_Type.xsd"/>
	<xsd:element name="FDMS_METADATA_REP_POC_DIF">
		<xsd:annotation>
			<xsd:documentation>root
```

```
element</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                        <xsd:choice maxOccurs="unbounded">
                                <xsd:element
ref="Point_Of_Contact_Addition"/>
                                <xsd:element
ref="Point_Of_Contact_Replacement"/>
                        </xsd:choice>
                        <xsd:attribute                    name="Version"
type="FDMS_IntegerOrNull_Type"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Point_Of_Contact_Addition">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Point_Of_Contact_SDS_ID"/>
                                <xsd:element
ref="Point_Of_Contact_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                  name="Point_Of_Contact_Address_1"
type="FDMS_Long_Type"/>
        <xsd:element                  name="Point_Of_Contact_Address_2"
```

```
type="FDMS_Long_Type"/>
        <xsd:element                    name="Point_Of_Contact_Address_3"
type="FDMS_Long_Type"/>
        <xsd:element                          name="Point_Of_Contact_City"
type="FDMS_Long_Type"/>
        <xsd:element      name="Point_Of_Contact_Contact_Instructions"
type="FDMS_String255_Type"/>
        <xsd:element                       name="Point_Of_Contact_Country"
type="FDMS_Long_Type"/>
        <xsd:element              name="Point_Of_Contact_DSN_Telephone"
type="FDMS_Short_Type1"/>
        <xsd:element name="Point_Of_Contact_Data">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Point_Of_Contact_Stereotype"/>
                                <xsd:element  ref="Point_Of_Contact_Type"
minOccurs="0"/>
                                <xsd:element
ref="Point_Of_Contact_Position" minOccurs="0"/>
                                <xsd:element
ref="Point_Of_Contact_Prefix" minOccurs="0"/>
                                <xsd:element
ref="Point_Of_Contact_Last_Name"/>
                                <xsd:element
ref="Point_Of_Contact_First_Name" minOccurs="0"/>
```

```
                              <xsd:element
ref="Point_Of_Contact_Middle_Name" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_Suffix" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_Address_1"/>
                              <xsd:element
ref="Point_Of_Contact_Address_2" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_Address_3" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_City"/>
                              <xsd:element
ref="Point_Of_Contact_State"/>
                              <xsd:element
ref="Point_Of_Contact_Country"/>
                              <xsd:element
ref="Point_Of_Contact_Postal_Code"/>
                              <xsd:element
ref="Point_Of_Contact_Voice_Telephone"/>
                              <xsd:element
ref="Point_Of_Contact_DSN_Telephone" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_TTD_TTY_Telephone" minOccurs="0"/>
                              <xsd:element ref="Point_Of_Contact_FAX"/>
                              <xsd:element
```

```
ref="Point_Of_Contact_E-mail"/>
                        <xsd:element
ref="Point_Of_Contact_Uniform_Resource_Locator" minOccurs="0"/>
                        <xsd:element
ref="Point_Of_Contact_Sponsor" minOccurs="0"/>
                        <xsd:element
ref="Point_Of_Contact_Employer" minOccurs="0"/>
                        <xsd:element
ref="Point_Of_Contact_Employer_Start_Date" minOccurs="0"/>
                        <xsd:element
ref="Point_Of_Contact_Employer_End_Date" minOccurs="0"/>
                        <xsd:element
ref="Point_Of_Contact_Contact_Instructions" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                    name="Point_Of_Contact_E-mail"
type="FDMS_Long_Type"/>
        <xsd:element name="Point_Of_Contact_Employer">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
ref="Point_Of_Contact_Reference"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
```

```
        <xsd:element          name="Point_Of_Contact_Employer_End_Date"
type="FDMS_Strict_Date_Type"/>
        <xsd:element        name="Point_Of_Contact_Employer_Start_Date"
type="FDMS_Strict_Date_Type"/>
        <xsd:element name="Point_Of_Contact_Existing">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Point_Of_Contact_RDS_ID"/>
                                <xsd:element
ref="Point_Of_Contact_SDS_ID" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                          name="Point_Of_Contact_FAX"
type="FDMS_Short_Type1"/>
        <xsd:element                 name="Point_Of_Contact_First_Name"
type="FDMS_Long_Type"/>
        <xsd:element name="Point_Of_Contact_Internal">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Point_Of_Contact_SDS_ID"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
```

```
        <xsd:element                    name="Point_Of_Contact_Last_Name"
type="FDMS_Long_Type"/>
        <xsd:element                  name="Point_Of_Contact_Middle_Name"
type="FDMS_Long_Type"/>
        <xsd:element                     name="Point_Of_Contact_Position"
type="FDMS_Long_Type"/>
        <xsd:element                  name="Point_Of_Contact_Postal_Code"
type="FDMS_Short_Type1"/>
        <xsd:element                       name="Point_Of_Contact_Prefix"
type="FDMS_Long_Type"/>
        <xsd:element                       name="Point_Of_Contact_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Point_Of_Contact_Reference">
                <xsd:complexType>
                        <xsd:choice>
                                <xsd:element
ref="Point_Of_Contact_Existing"/>
                                <xsd:element
ref="Point_Of_Contact_Internal"/>
                        </xsd:choice>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Point_Of_Contact_Replacement">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
```

```
ref="Point_Of_Contact_RDS_ID"/>
                              <xsd:element
ref="Point_Of_Contact_SDS_ID" minOccurs="0"/>
                              <xsd:element
ref="Point_Of_Contact_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                        name="Point_Of_Contact_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Point_Of_Contact_Sponsor">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Point_Of_Contact_Reference"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                         name="Point_Of_Contact_State"
type="FDMS_POC_State_Type"/>
        <xsd:element                  name="Point_Of_Contact_Stereotype"
type="FDMS_POC_Contact_Stereotype_Type"/>
        <xsd:element                        name="Point_Of_Contact_Suffix"
type="FDMS_Long_Type"/>
        <xsd:element          name="Point_Of_Contact_TTD_TTY_Telephone"
type="FDMS_Short_Type1"/>
```

```
        <xsd:element                    name="Point_Of_Contact_Type"
type="FDMS_String20_Type"/>
        <xsd:element  name="Point_Of_Contact_Uniform_Resource_Locator"
type="xsd:anyURI"/>
        <xsd:element          name="Point_Of_Contact_Voice_Telephone"
type="FDMS_Short_Type1"/>
        <xsd:simpleType name="FDMS_POC_Contact_Stereotype_Type">
                <xsd:annotation>
                        <xsd:documentation>Allows value of ORG, PER or
POS</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="ORG"/>
                        <xsd:enumeration value="PER"/>
                        <xsd:enumeration value="POS"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_POC_State_Type">
                <xsd:annotation>
                        <xsd:documentation>Text string (2) or N/A or
TBD</xsd:documentation>
                </xsd:annotation>
                <xsd:union                memberTypes="FDMS_Null_Type_1
FDMS_String2_Type"/>
        </xsd:simpleType>
</xsd:schema>
```

C-3 FDMS_Source_Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
       <xsd:include
schemaLocation="..¥..¥Point_Of_Contact¥W3C_XML_Schema_Files¥FDMS_Poin
t_Of_Contact_Schema.xsd"/>
       <xsd:element name="FDMS_METADATA_REP_SOURCE_DIF">
              <xsd:annotation>
                     <xsd:documentation>root
element</xsd:documentation>
              </xsd:annotation>
              <xsd:complexType>
                     <xsd:sequence>
                            <xsd:choice maxOccurs="unbounded">
                                   <xsd:element
ref="Source_Element_Addition"/>
                                   <xsd:element
ref="Source_Element_Replacement"/>
                                   <xsd:element
ref="Source_Element_Update"/>
                            </xsd:choice>
                            <xsd:choice minOccurs="0">
                                   <xsd:element
ref="Point_Of_Contact_Addition"/>
```

```
                                    <xsd:element
ref="Point_Of_Contact_Replacement"/>
                                </xsd:choice>
                            </xsd:sequence>
                            <xsd:attribute name="Version" type="xsd:string"/>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element            name="Source_Element_Access_Constraint"
type="FDMS_Source_String_255_Type"/>
                <xsd:element name="Source_Element_Addition">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element
ref="Source_Element_SDS_ID"/>
                                <xsd:element ref="Source_Element_Data"/>
                            </xsd:sequence>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element            name="Source_Element_Asset_Association"
type="FDMS_Source_String_255_Type"/>
                <xsd:element    name="Source_Element_Compatibility_Information"
type="FDMS_Source_String_255_Type"/>
                <xsd:element                    name="Source_Element_Component"
type="FDMS_Source_String_255_Type"/>
                <xsd:element name="Source_Element_Data">
                        <xsd:complexType>
```

```
                    <xsd:sequence>
                        <xsd:choice minOccurs="0">
                            <xsd:element
ref="Point_Of_Contact_Existing"/>
                            <xsd:element
ref="Point_Of_Contact_Internal"/>
                        </xsd:choice>
                        <xsd:element
ref="Source_Element_Information_Asset_Name"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Alternate_Name" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Short_Name" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Type_Code"/>
                        <xsd:element
ref="Source_Element_Designation"/>
                        <xsd:element
ref="Source_Element_Component"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Abstract"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Purpose"/>
                        <xsd:element
ref="Source_Element_Online_Resource_Linkage" minOccurs="0"/>
                        <xsd:element
```

```
ref="Source_Element_Data_Dictionary_Name" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Asset_Association" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Authority" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Environment" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Time_to_Prepare" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Access_Constraint" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Use_Constraint" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Use_Limitation" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_User_Determined_Limitations" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Compatibility_Information" minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Information_Asset_Distribution_Information"
minOccurs="0"/>
                        <xsd:element
ref="Source_Element_Standard_Ordering_Instruction" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
```

```
        </xsd:element>
        <xsd:element        name="Source_Element_Data_Dictionary_Name"
type="FDMS_Source_String_255_Type"/>
        <xsd:element                name="Source_Element_Designation"
type="FDMS_Source_String_255_Type"/>
        <xsd:element name="Source_Element_Existing">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Source_Element_RDS_ID"/>
                                <xsd:element  ref="Source_Element_SDS_ID"
minOccurs="0"/>
                                <xsd:element
ref="Source_Model_Element_Location_In_Source" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element  name="Source_Element_Information_Asset_Abstract"
type="FDMS_Source_String_1023_Type"/>
        <xsd:element
name="Source_Element_Information_Asset_Alternate_Name"
type="FDMS_Source_String_255_Type"/>
        <xsd:element name="Source_Element_Information_Asset_Authority"
type="FDMS_Source_String_255_Type"/>
        <xsd:element
name="Source_Element_Information_Asset_Distribution_Information"
```

```
type="FDMS_Source_String_255_Type"/>
        <xsd:element
name="Source_Element_Information_Asset_Environment"
type="FDMS_Source_String_255_Type"/>
        <xsd:element      name="Source_Element_Information_Asset_Name"
type="FDMS_Source_String_255_Type"/>
        <xsd:element    name="Source_Element_Information_Asset_Purpose"
type="FDMS_Source_String_255_Type"/>
        <xsd:element
name="Source_Element_Information_Asset_Short_Name"
type="FDMS_Source_String_255_Type"/>
        <xsd:element name="Source_Element_Information_Asset_Type_Code"
type="FDMS_Source_String_255_Type"/>
        <xsd:element name="Source_Element_Internal">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Source_Element_SDS_ID"/>
                                <xsd:element
ref="Source_Model_Element_Location_In_Source" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element     name="Source_Element_Online_Resource_Linkage"
type="FDMS_Source_String_255_Type"/>
        <xsd:element                        name="Source_Element_RDS_ID"
```

```
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Source_Element_Replacement">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Source_Element_RDS_ID"/>
                                <xsd:element  ref="Source_Element_SDS_ID"
minOccurs="0"/>
                                <xsd:element ref="Source_Element_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                        name="Source_Element_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element
name="Source_Element_Standard_Ordering_Instruction"
type="FDMS_Source_String_255_Type"/>
        <xsd:element              name="Source_Element_Time_to_Prepare"
type="FDMS_Source_String_255_Type"/>
        <xsd:element name="Source_Element_Update">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Source_Element_RDS_ID"/>
                                <xsd:element  ref="Source_Element_SDS_ID"
minOccurs="0"/>
```

```
                        <xsd:element ref="Source_Element_Data"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element                name="Source_Element_Use_Constraint"
type="FDMS_Source_String_255_Type"/>
            <xsd:element                name="Source_Element_Use_Limitation"
type="FDMS_Source_String_255_Type"/>
            <xsd:element name="Source_Element_User_Determined_Limitations"
type="FDMS_Source_String_255_Type"/>
            <xsd:element    name="Source_Model_Element_Location_In_Source"
type="FDMS_Source_String_80_Type"/>
            <xsd:simpleType name="FDMS_Source_String_80_Type">
                <xsd:annotation>
                    <xsd:documentation>Equivalent                    to
FDMS_Long_Type3.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_Type3"/>
            </xsd:simpleType>
            <xsd:simpleType name="FDMS_Source_String_255_Type">
                <xsd:annotation>
                    <xsd:documentation>Equivalent                    to
FDMS_Long_255_Type1.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_255_Type1"/>
            </xsd:simpleType>
```

```
        <xsd:simpleType name="FDMS_Source_String_1023_Type">
                <xsd:annotation>
                        <xsd:documentation>Equivalent                    to
FDMS_Long_1023_Type1.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="FDMS_Long_1023_Type1"/>
        </xsd:simpleType>
</xsd:schema>
```

C-4 FDMS_Native_Library_Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
       <xsd:include
schemaLocation="..¥..¥Source¥W3C_XML_Schema_Files¥FDMS_Source_Schema.
xsd"/>
       <xsd:element      name="Model_Element_Location_In_Native_Form"
type="FDMS_Long_Type3"/>
       <xsd:element name="Native_Library_Element_Addition">
              <xsd:complexType>
                     <xsd:sequence>
                            <xsd:element
ref="Native_Library_Element_SDS_ID"/>
                            <xsd:element
ref="Native_Library_Element_Data"/>
                     </xsd:sequence>
              </xsd:complexType>
       </xsd:element>
       <xsd:element name="Native_Library_Element_Data">
              <xsd:complexType>
                     <xsd:sequence>
                            <xsd:choice minOccurs="0">
                                   <xsd:element
ref="Point_Of_Contact_Existing"/>
```

```
                                    <xsd:element
ref="Point_Of_Contact_Internal"/>
                                </xsd:choice>
                                <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:element
ref="Source_Element_Existing"/>
                                    <xsd:element
ref="Source_Element_Internal"/>
                                </xsd:choice>
                                <xsd:element
ref="Native_Library_Element_Name"/>
                                <xsd:element
ref="Native_Library_Element_File_Type"/>
                                <xsd:element
ref="Native_Library_Element_Online_Resource_Linkage" minOccurs="0"/>
                                <xsd:element
ref="Native_Library_Element_Description" minOccurs="0"/>
                                <xsd:element
ref="Native_Library_Element_Version_Number" minOccurs="0"/>
                                <xsd:element
ref="Native_Library_Element_Version_Date"/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                    <xsd:element          name="Native_Library_Element_Description"
```

```
type="FDMS_Long_1023_Type1"/>
        <xsd:element name="Native_Library_Element_Existing">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Native_Library_Element_RDS_ID"/>
                                <xsd:element
ref="Native_Library_Element_SDS_ID" minOccurs="0"/>
                                <xsd:element
ref="Model_Element_Location_In_Native_Form" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element            name="Native_Library_Element_File_Type"
type="FDMS_String4_Type"/>
        <xsd:element name="Native_Library_Element_Internal">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Native_Library_Element_SDS_ID"/>
                                <xsd:element
ref="Model_Element_Location_In_Native_Form" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                    name="Native_Library_Element_Name"
```

```
type="FDMS_String255_Type"/>
        <xsd:element
name="Native_Library_Element_Online_Resource_Linkage"
type="FDMS_String80_Type"/>
        <xsd:element              name="Native_Library_Element_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Native_Library_Element_Replacement">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Native_Library_Element_RDS_ID"/>
                                <xsd:element
ref="Native_Library_Element_SDS_ID" minOccurs="0"/>
                                <xsd:element
ref="Native_Library_Element_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element              name="Native_Library_Element_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element        name="Native_Library_Element_Version_Date"
type="FDMS_Strict_Date_Type"/>
        <xsd:element      name="Native_Library_Element_Version_Number"
type="FDMS_Short_Type1"/>
</xsd:schema>
```

C-5 FDMS_Taxonomy_Schema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
        <xsd:annotation>
                <xsd:documentation>This schema is not valid by itself.  It
is used by (included into) FDMS_Model_Schema.xsd</xsd:documentation>
        </xsd:annotation>
        <xsd:element name="Short_Taxonomy_Keyword_Element">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_Element_SDS_ID" minOccurs="0"/>
                                <xsd:choice>
                                        <xsd:element
ref="Taxonomy_Keyword_Existing"/>
                                        <xsd:element
ref="Taxonomy_Keyword_Internal"/>
                                </xsd:choice>
                                <xsd:element
ref="Short_Taxonomy_Keyword_Element" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy">
```

```
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:choice>
                                        <xsd:element
ref="Taxonomy_Element_Addition"/>
                                        <xsd:element
ref="Taxonomy_Element_Replacement"/>
                                </xsd:choice>
                                <xsd:element
ref="Short_Taxonomy_Keyword_Element" maxOccurs="unbounded"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy_Element_Addition">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Element_SDS_ID"/>
                                <xsd:element
ref="Taxonomy_Element_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy_Element_Data">
                <xsd:complexType>
                        <xsd:sequence>
```

```
                        <xsd:element
ref="Taxonomy_Element_Name"/>
                        <xsd:element
ref="Taxonomy_Element_Description"/>
                        <xsd:choice                    minOccurs="0"
maxOccurs="unbounded">
                            <xsd:element
ref="Source_Element_Existing"/>
                            <xsd:element
ref="Source_Element_Internal"/>
                        </xsd:choice>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                name="Taxonomy_Element_Description"
type="FDMS_String1023_Type"/>
        <xsd:element name="Taxonomy_Element_Existing">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
ref="Taxonomy_Element_RDS_ID"/>
                        <xsd:element
ref="Taxonomy_Element_SDS_ID" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
```

```
        <xsd:element name="Taxonomy_Element_Internal">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Element_SDS_ID"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                        name="Taxonomy_Element_Name"
type="FDMS_String255_Type"/>
        <xsd:element                      name="Taxonomy_Element_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Taxonomy_Element_Replacement">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Element_RDS_ID"/>
                                <xsd:element
ref="Taxonomy_Element_SDS_ID" minOccurs="0"/>
                                <xsd:element
ref="Taxonomy_Element_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                      name="Taxonomy_Element_SDS_ID"
type="FDMS_DS_ID_Type"/>
```

```
        <xsd:element name="Taxonomy_Keyword_Addition">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_SDS_ID"/>
                                <xsd:element
ref="Taxonomy_Keyword_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy_Keyword_Data">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_Name"/>
                                <xsd:element
ref="Taxonomy_Keyword_Description"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                name="Taxonomy_Keyword_Description"
type="FDMS_String1023_Type"/>
        <xsd:element name="Taxonomy_Keyword_Element_Addition">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
```

```
ref="Taxonomy_Keyword_Element_SDS_ID"/>
                        <xsd:element
ref="Taxonomy_Keyword_Element_Data"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy_Keyword_Element_Data">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:choice>
                            <xsd:element
ref="Taxonomy_Element_Existing"/>
                            <xsd:element
ref="Taxonomy_Element_Internal"/>
                        </xsd:choice>
                        <xsd:choice>
                            <xsd:element
ref="Taxonomy_Keyword_Existing"/>
                            <xsd:element
ref="Taxonomy_Keyword_Internal"/>
                        </xsd:choice>
                        <xsd:choice minOccurs="0">
                            <xsd:element
ref="Taxonomy_Keyword_Element_Existing"/>
                            <xsd:element
ref="Taxonomy_Keyword_Element_Internal"/>
```

```
                        </xsd:choice>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Taxonomy_Keyword_Element_Existing">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
ref="Taxonomy_Keyword_Element_RDS_ID"/>
                        <xsd:element
ref="Taxonomy_Keyword_Element_SDS_ID" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Taxonomy_Keyword_Element_Internal">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element
ref="Taxonomy_Keyword_Element_SDS_ID"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element            name="Taxonomy_Keyword_Element_RDS_ID"
type="FDMS_DS_ID_Type"/>
            <xsd:element name="Taxonomy_Keyword_Element_Replacement">
                <xsd:complexType>
```

```
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_Element_RDS_ID"/>
                                <xsd:element
ref="Taxonomy_Keyword_Element_SDS_ID" minOccurs="0"/>
                                <xsd:element
ref="Taxonomy_Keyword_Element_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element            name="Taxonomy_Keyword_Element_SDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Taxonomy_Keyword_Existing">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_RDS_ID"/>
                                <xsd:element
ref="Taxonomy_Keyword_SDS_ID" minOccurs="0"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Taxonomy_Keyword_Internal">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
```

```
ref="Taxonomy_Keyword_SDS_ID"/>
                    </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                    name="Taxonomy_Keyword_Name"
type="FDMS_String255_Type"/>
        <xsd:element                    name="Taxonomy_Keyword_RDS_ID"
type="FDMS_DS_ID_Type"/>
        <xsd:element name="Taxonomy_Keyword_Replacement">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element
ref="Taxonomy_Keyword_RDS_ID"/>
                                <xsd:element
ref="Taxonomy_Keyword_SDS_ID" minOccurs="0"/>
                                <xsd:element
ref="Taxonomy_Keyword_Data"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element                    name="Taxonomy_Keyword_SDS_ID"
type="FDMS_DS_ID_Type"/>
</xsd:schema>
```

C-6 FDMS_Basic_Datatypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
	<xsd:annotation>
		<xsd:documentation>This schema defines a number of
reusable datatypes for use with all sections of the
FDMS</xsd:documentation>
	</xsd:annotation>
	<xsd:simpleType name="FDMS_Boolean_Type">
		<xsd:annotation>
			<xsd:documentation>Allows 'True' or 'False'. (The
built-in boolean type cannot be used, because it accepts only lower-case:
true, false)</xsd:documentation>
		</xsd:annotation>
		<xsd:restriction base="xsd:string">
			<xsd:enumeration value="True"/>
			<xsd:enumeration value="False"/>
		</xsd:restriction>
	</xsd:simpleType>
	<xsd:simpleType name="FDMS_Null_Type_1">
		<xsd:annotation>
			<xsd:documentation>Allows 'N/A' or
'TBD'</xsd:documentation>
		</xsd:annotation>
```

```
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="N/A"/>
                        <xsd:enumeration value="TBD"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Null_Type_2">
                <xsd:annotation>
                        <xsd:documentation>Allows  'N/A'  or  'TBD'  or
'UNK'</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="N/A"/>
                        <xsd:enumeration value="UNK"/>
                        <xsd:enumeration value="TBD"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Null_Type_3">
                <xsd:annotation>
                        <xsd:documentation>Allows                  'N/A'
only</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="N/A"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String2_Type">
```

```
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    2
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="2"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String4_Type">
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    4
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="4"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String20_Type">
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    20
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
```

```
                        <xsd:maxLength value="20"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String80_Type">
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    80
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="80"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String255_Type">
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    255
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="255"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String1023_Type">
                <xsd:annotation>
                        <xsd:documentation>String    of    up    to    1023
```

```
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="1023"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_String65535_Type">
                <xsd:annotation>
                        <xsd:documentation>String   of   up   to   65535
characters</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="65535"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Positive_Integer_Type">
                <xsd:annotation>
                        <xsd:documentation>Any   integer   that   is   >
0</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:integer">
                        <xsd:minInclusive value="0"/>
                </xsd:restriction>
        </xsd:simpleType>
```

```
        <xsd:simpleType name="FDMS_IntegerOrNull_Type">
                <xsd:annotation>
                        <xsd:documentation>Choice                between
FDMS_Positive_Integer_Type and FDMS_Null_Type_2</xsd:documentation>
                </xsd:annotation>
                <xsd:union                memberTypes="FDMS_Null_Type_2
FDMS_Positive_Integer_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Short_Type1">
                <xsd:annotation>
                        <xsd:documentation>Choice                between
FDMS_String20_Type and FDMS_Null_Type_1</xsd:documentation>
                </xsd:annotation>
                <xsd:union                memberTypes="FDMS_Null_Type_1
FDMS_String20_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Short_Type2">
                <xsd:annotation>
                        <xsd:documentation>Choice                between
FDMS_String20_Type and FDMS_Null_Type_3</xsd:documentation>
                </xsd:annotation>
                <xsd:union                memberTypes="FDMS_Null_Type_3
FDMS_String20_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Short_Type3">
                <xsd:annotation>
```

```
                    <xsd:documentation>Choice                    between
FDMS_String20_Type and FDMS_Null_Type_2</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_2
FDMS_String20_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_Type">
                <xsd:annotation>
                    <xsd:documentation>Choice                    between
FDMS_String80_Type and FDMS_Null_Type_1</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_1
FDMS_String80_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_Type2">
                <xsd:annotation>
                    <xsd:documentation>Choice                    between
FDMS_String80_Type and FDMS_Null_Type_2</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_2
FDMS_String80_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_Type3">
                <xsd:annotation>
                    <xsd:documentation>Choice                    between
FDMS_String80_Type and FDMS_Null_Type_3</xsd:documentation>
```

```
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_3
FDMS_String80_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_255_Type1">
                <xsd:annotation>
                        <xsd:documentation>Choice                  between
FDMS_String255_Type and FDMS_Null_Type_1</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_1
FDMS_String255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_1023_Type1">
                <xsd:annotation>
                        <xsd:documentation>Choice                  between
FDMS_String1023_Type and FDMS_Null_Type_1</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_1
FDMS_String1023_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_255_Type">
                <xsd:annotation>
                        <xsd:documentation>Choice                  between
FDMS_String255_Type and FDMS_Null_Type_2</xsd:documentation>
                </xsd:annotation>
                <xsd:union                  memberTypes="FDMS_Null_Type_2
```

```
FDMS_String255_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Long_1023_Type">
                <xsd:annotation>
                        <xsd:documentation>Choice                between
FDMS_String1023_Type and FDMS_Null_Type_2</xsd:documentation>
                </xsd:annotation>
                <xsd:union                memberTypes="FDMS_Null_Type_2
FDMS_String1023_Type"/>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_S_D_Flag_Type">
                <xsd:annotation>
                        <xsd:documentation>"S"    or    "D"    or    "
"</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="S"/>
                        <xsd:enumeration value="D"/>
                        <xsd:enumeration value=" "/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_DS_ID_Type">
                <xsd:annotation>
                      <xsd:documentation>Equivalent to FDMS_Short_Type2.
This   type   is   intended   for   use   with   RDS_ID   and   SDS_ID
elements.</xsd:documentation>
```

```
                </xsd:annotation>

                <xsd:restriction base="FDMS_Short_Type2"/>

        </xsd:simpleType>

        <xsd:simpleType name="FDMS_Version_Type">

                <xsd:annotation>

                        <xsd:documentation>Equivalent                    to
FDMS_Positive_Integer_Type.</xsd:documentation>

                </xsd:annotation>

                <xsd:restriction base="FDMS_Positive_Integer_Type"/>

        </xsd:simpleType>

</xsd:schema>
```

C-7 FDMS_Strict_Date_Type.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
	<xsd:annotation>
		<xsd:documentation>This schema defines the reusable
FDMS_Strict_DateType, which specifies the allowed formats that a date
value can have, and also specifies allowed ranges for each of the day,
month, year, hour, minute and second components of a date value. In order
to acheive complete validation, the instance document must also be
validated against the Schematron schema that contains the
FDMS_Date_Datatype pattern.</xsd:documentation>
	</xsd:annotation>
	<xsd:include schemaLocation="FDMS_Basic_Datatypes.xsd"/>
	<xsd:simpleType name="FDMS_Strict_Time_Type">
		<xsd:annotation>
			<xsd:documentation>Specifies hh:mm:ss format.
Does not allow hh>24, mm>60, ss>60</xsd:documentation>
		</xsd:annotation>
		<xsd:restriction base="xsd:time">
			<xsd:pattern
value="(([0-1][0-9]|2[0-3]):([0-5][0-9]|60):([0-5][0-9]|60))"/>
		</xsd:restriction>
	</xsd:simpleType>
	<xsd:simpleType name="FDMS_Strict_Simple_Date_Type_1">
```

```
                <xsd:annotation>
                        <xsd:documentation>specifies  dd/mm/yy  format.
Does not allow dd>31, mm>12</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(0[1-9]|1[0-2])/¥d{2})"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_Simple_Date_Type_2">
                <xsd:annotation>
                        <xsd:documentation>specifies  dd/mmm/yy  format.
Does  not  allow  dd>31,  mm  must  be  one  of  {Jan,  Feb,  Mar,
etc.}</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|S
ep|Oct|Nov|Dec)/¥d{2})"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_Simple_Date_Type_3">
                <xsd:annotation>
                        <xsd:documentation>Specifies  dd/mm/yyyy  format.
Does not allow dd>31, mm>12.</xsd:documentation>
                </xsd:annotation>
```

```
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(0[1-9]|1[0-2])/(¥d{4}))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_Simple_Date_Type_4">
                <xsd:annotation>
                        <xsd:documentation>Specifies dd/mmm/yyyy format.
Does not allow dd>31, mm must be one of {Jan, Feb, Mar,
etc.}</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|S
ep|Oct|Nov|Dec)/(¥d{4}))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_Simple_Date_Type_5">
                <xsd:annotation>
                        <xsd:documentation>Specifies dd mmm yyyy format.
Does not allow dd>31, mm must be one of {Jan, Feb, Mar,
etc.}</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])¥s(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|
```

```
Sep|Oct|Nov|Dec)¥s(¥d{4}))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_DateTime_Type_1">
                <xsd:annotation>
                        <xsd:documentation>Specifies  dd/mm/yy  hh:mm:ss
format.      Combination      of      FDMS_Strict_Time_Type      and
FDMS_Strict_Simple_Date_Type_1.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(0[1-9]|1[0-2])/¥d{2}¥s([0-1][0-9]
|2[0-3]):([0-5][0-9]|60):([0-5][0-9]|60))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_DateTime_Type_2">
                <xsd:annotation>
                        <xsd:documentation>Specifies  dd/mmm/yy  hh:mm:ss
format.      Combination      of      FDMS_Strict_Time_Type      and
FDMS_Strict_Simple_Date_Type_2.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|S
ep|Oct|Nov|Dec)/¥d{2}¥s([0-1][0-9]|2[0-3]):([0-5][0-9]|60):([0-5][0-9
]|60))"/>
```

```
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_DateTime_Type_3">
                <xsd:annotation>
                        <xsd:documentation>Specifies dd/mm/yyyy hh:mm:ss
format.      Combination      of      FDMS_Strict_Time_Type      and
FDMS_Strict_Simple_Date_Type_3.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(0[1-9]|1[0-2])/(¥d{4})¥s([0-1][0-
9]|2[0-3]):([0-5][0-9]|60):([0-5][0-9]|60))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_DateTime_Type_4">
                <xsd:annotation>
                        <xsd:documentation>Specifies dd/mmm/yyyy hh:mm:ss
format.      Combination      of      FDMS_Strict_Time_Type      and
FDMS_Strict_Simple_Date_Type_4.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])/(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|S
ep|Oct|Nov|Dec)/(¥d{4})¥s([0-1][0-9]|2[0-3]):([0-5][0-9]|60):([0-5][0
-9]|60))"/>
                </xsd:restriction>
```

```
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_DateTime_Type_5">
                <xsd:annotation>
                        <xsd:documentation>Specifies dd mmm yyyy hh:mm:ss
format.      Combination      of      FDMS_Strict_Time_Type      and
FDMS_Strict_Simple_Date_Type_5.</xsd:documentation>
                </xsd:annotation>
                <xsd:restriction base="xsd:string">
                        <xsd:pattern
value="((0[1-9]|[1-2][0-9]|3[0-1])¥s(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|
Sep|Oct|Nov|Dec)¥s(¥d{4})¥s([0-1][0-9]|2[0-3]):([0-5][0-9]|60):([0-5]
[0-9]|60))"/>
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FDMS_Strict_Date_Type">
                <xsd:annotation>
                        <xsd:documentation>Specifies all FDMS DIF date
formats, with 'strict' validation. Combination of all the simple types
in     this     schema     and     FDMS_Null_Type_2     (defined     in
FDMS_Basic_Datatypes.xsd).</xsd:documentation>
                </xsd:annotation>
                <xsd:union     memberTypes="FDMS_Strict_DateTime_Type_1
FDMS_Strict_DateTime_Type_2                FDMS_Strict_DateTime_Type_3
FDMS_Strict_DateTime_Type_4                FDMS_Strict_DateTime_Type_5
FDMS_Strict_Simple_Date_Type_1          FDMS_Strict_Simple_Date_Type_2
FDMS_Strict_Simple_Date_Type_3          FDMS_Strict_Simple_Date_Type_4
```

```
FDMS_Strict_Simple_Date_Type_5                    FDMS_Strict_Time_Type

FDMS_Null_Type_2"/>

        </xsd:simpleType>

</xsd:schema>
```

C-8 FDMS_Point_Of_Contact_Topup.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
	<!--
This Schematron schema is intended to provide top-up validation that supplements the validation performed by the FDMS_Point_of_Contact_Schema.xsd schema.
This schema provides top-up validation for date values only.
This schema consists of two patterns - one is reusable; the other applies only to the Point Of Contact DIF.
-->
	<title>Top-up Schematron schema for the Point of Contact DIF</title>
	<!--
		The following pattern is reusable...
	-->
	<pattern name="FDMS Date Datatype Validation" id="FDMS_Date_Datatype">
		<!--
		This pattern consists of a single "rule". The rule is abstract, meaning that it can be applied to any context element.
		The rule consists of six "report"s. A report triggers if its "test" condition evaluates to true. The feedback message specified between the report tags is then output to the user.
		The purpose of this rule is to provide additional top-up
```

validation on date values, according to the FDMS specifications. In particular, the reports in the rule test whether the value for the day component of a date value is consistent with the value of the month and year components.

This pattern is reusable - it can be copy-and-pasted into any other Schematron schema that requires it, unless there is a name conflict.

```
-->
<rule abstract="true" id="FDMS_Date_Type_dd">
      <report test="

(number(substring-before(substring-after(.,'/'),'/')) = 2 and
number(substring-before(.,'/')) &gt; 29)
                  or
                  (substring-before(substring-after(.,'/'),'/')  =
'Feb' and number(substring-before(.,'/')) &gt; 29)
                   or
                  (substring-before(substring-after(.,' '),' ') =
'Feb' and number(substring-before(.,' ')) &gt; 29)
                  ">Error in content of element <name/>: February can
never have more than 29 days</report>
                  <report test="
                  (
                        (
                              not
                              (
```

```
                                        (

       (number(substring-before(substring-after(substring-after(.,'/'
),'/'),' ')) div 4)

                                                =

       (floor(number(substring-before(substring-after(substring-after
(.,'/'),'/'),' ')) div 4))

                                        )

                                        or

                                        (

       (number(substring-after(substring-after(.,'/'),'/')) div 4)

                                                =

       (floor(number(substring-after(substring-after(.,'/'),'/')) div
4))

                                        )

                                    )

                            )

                            and

                            (

       number(substring-before(substring-after(.,'/'),'/')) = 2

                            )
```

```
                        and
                        (
                                number(substring-before(.,'/'))
&gt; 28
                        )
                )
                 or
                (
                        (
                                not
                                (
                                        (
       (number(substring-before(substring-after(substring-after(.,'/'
),'/'),' ')) div 4)
                                                =
       (floor(number(substring-before(substring-after(substring-after
(.,'/'),'/'),' ')) div 4))
                                        )
                                        or
                                        (
       (number(substring-after(substring-after(.,'/'),'/')) div 4)
                                                =
```

```
        (floor(number(substring-after(substring-after(.,'/'),'/')) div
4))
                                                )
                                        )


                                )
                                and
                                (

        substring-before(substring-after(.,'/'),'/') = 'Feb'
                                )
                                and
                                (
                                        number(substring-before(.,'/'))
&gt; 28
                                )
                        )
                        or
                        (
                                (
                                        not
                                        (
                                                (

        (number(substring-before(substring-after(substring-after(.,'
'),' '),' ')) div 4)
```

```
                                                    =

       (floor(number(substring-before(substring-after(substring-after
(.,' '),' '),' ')) div 4))
                                                    )
                                                    or
                                                    (

       (number(substring-after(substring-after(.,' '),' ')) div 4)
                                                    =

       (floor(number(substring-after(substring-after(.,' '),' ')) div
4))
                                                    )
                                  )
                              )
                              and
                              (

       substring-before(substring-after(.,' '),' ') = 'Feb'

                              )
                              and
                              (
                                     number(substring-before(.,'     '))
```

```
&gt; 28
                              )
                        )
                        ">Error in content of element <name/>: Not a Leap
Year - February cannot have more than 28 days</report>
                        <report test="

       (number(substring-before(substring-after(.,'/'),'/')) = 4 and
number(substring-before(.,'/')) &gt; 30)
                         or
                        (substring-before(substring-after(.,'/'),'/')  =
'Apr' and number(substring-before(.,'/')) &gt; 30)
                         or
                        (substring-before(substring-after(.,' '),' ') =
'Apr' and number(substring-before(.,' ')) &gt; 30)
                        ">Error in content of element <name/>: April cannot
have more than 30 days</report>
                        <report test="

       (number(substring-before(substring-after(.,'/'),'/')) = 6 and
number(substring-before(.,'/')) &gt; 30)
                        or
                        (substring-before(substring-after(.,'/'),'/')  =
'Jun' and number(substring-before(.,'/')) &gt; 30)
                        or
                        (substring-before(substring-after(.,' '),' ') =
```

```
'Jun' and number(substring-before(.,' ')) &gt; 30)
                    ">Error in content of element <name/>: June cannot
have more than 30 days</report>
                    <report test="

       (number(substring-before(substring-after(.,'/'),'/')) = 9 and
number(substring-before(.,'/')) &gt; 30)
                    or
                    (substring-before(substring-after(.,'/'),'/')  =
'Sep' and number(substring-before(.,'/')) &gt; 30)
                     or
                    (substring-before(substring-after(.,' '),' ') =
'Sep' and number(substring-before(.,' ')) &gt; 30)
                    ">Error in content of element <name/>: September
cannot have more than 30 days</report>
                    <report test="

       (number(substring-before(substring-after(.,'/'),'/')) = 11 and
number(substring-before(.,'/')) &gt; 30)
                    or
                    (substring-before(substring-after(.,'/'),'/')  =
'Nov' and number(substring-before(.,'/')) &gt; 30)
                    or
                    (substring-before(substring-after(.,' '),' ') =
'Nov' and number(substring-before(.,' ')) &gt; 30)
                    ">Error in content of element <name/>: November
```

```
cannot have more than 30 days</report>
                </rule>
        </pattern>
        <!--
                The following pattern is particular to the Point of Contact
DIF- not reusable...
        -->
        <pattern name="Using FDMS Date Datatype in Point of Contact DIF"
id="FDMS_Point_Of_Contact">
                <!--
        This pattern consists of two rules, which apply the abstract rule
from the FDMS_Date_Datatype pattern, above, to particular elements in the
Point of Contact DIF.
        -->
                <rule context="Point_Of_Contact_Employer_Start_Date">
                        <extends rule="FDMS_Date_Type_dd"/>
                </rule>
                <rule context="Point_Of_Contact_Employer_End_Date">
                        <extends rule="FDMS_Date_Type_dd"/>
                </rule>
        </pattern>
</schema>
```

C-9 FDMS_Model_Topup.sch

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
       <title>Schematron schema providing top-up validation for the FDMS
Model DIF.</title>
       <pattern name="Model DIF uniqueness constraints - SDS_ID values
must be unique across all Additions, Replacements, Forwards and Updates"
id="ModelUniqueSDS_ID">
              <rule                    context="//*[(substring(name(),
(string-length(name())-6))='_SDS_ID')]">
                     <report               test="(substring(name(..),
(string-length(name(..))-8))='_Addition') and
       (current() != 'N/A') and
       (
       (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
       or
       (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
       or
       (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
```

```
        or
        (count(//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[((substring(name(),
(string-length(name())-6))='_SDS_ID')) and (text() = current())]) &gt;=
2)
        )
        ">The SDS_ID specified in an Addition must be unique across all
other Additions, Replacements, Forwards and Updates in the
submission</report>
                        <report                test="(substring(name(..),
(string-length(name(..))-11))='_Replacement') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (count(//*[(substring(name(),
```

```
(string-length(name())-11))='_Replacement')]/*[((substring(name(),
(string-length(name())-6))='_SDS_ID')) and (text() = current())]) &gt;=
2)
        )
        ">The SDS_ID specified in a Replacement must be unique across all
other   Additions,   Replacements,   Forwards   and   Updates   in   the
submission</report>
                        <report                test="(substring(name(..),
(string-length(name(..))-7))='_Forward') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (count(//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[((substring(name(),
(string-length(name())-6))='_SDS_ID')) and (text() = current())]) &gt;=
```

```
2)
        )
        ">The SDS_ID specified in a Forward must be unique across all other
Additions, Replacements, Forwards and Updates in the submission</report>
                        <report                test="(substring(name(..),
(string-length(name(..))-6))='_Update') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_SDS_ID')] = current())
        or
        (count(//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[((substring(name(),
(string-length(name())-6))='_SDS_ID')) and (text() = current())]) &gt;=
2)
        )
        ">The SDS_ID specified in an Update must be unique across all other
```

```
Additions, Replacements, Forwards and Updates in the submission</report>
            </rule>
        </pattern>
        <pattern name="Model DIF uniqueness constraints - RDS_ID values
must be unique across all Additions, Replacements, Forwards and Updates"
id="ModelUniqueRDS_ID">
            <rule                    context="//*[(substring(name(),
(string-length(name())-6))='_RDS_ID')]">
                    <report              test="(substring(name(..),
(string-length(name(..))-8))='_Addition') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (count(//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[((substring(name(),
```

```
(string-length(name())-6))='_RDS_ID')) and (text() = current())]) &gt;=
2)
        )
        ">The RDS_ID specified in an Addition must be unique across all
other   Additions,   Replacements,   Forwards   and   Updates   in   the
submission</report>
                        <report                test="(substring(name(..),
(string-length(name(..))-11))='_Replacement') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (count(//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[((substring(name(),
(string-length(name())-6))='_RDS_ID')) and (text() = current())]) &gt;=
2)
```

```
        )
        ">The RDS_ID specified in a Replacement must be unique across all
other Additions, Replacements, Forwards and Updates in the
submission</report>
                        <report                test="(substring(name(..),
(string-length(name(..))-7))='_Forward') and
        (current() != 'N/A') and
        (
        (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
        or
        (count(//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[((substring(name(),
(string-length(name())-6))='_RDS_ID')) and (text() = current())]) &gt;=
2)
        )
        ">The RDS_ID specified in a Forward must be unique across all other
```

```
Additions, Replacements, Forwards and Updates in the submission</report>
                    <report                test="(substring(name(..),
(string-length(name(..))-6))='_Update') and
       (current() != 'N/A') and
       (
       (//*[(substring(name(),
(string-length(name())-11))='_Replacement')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
       or
       (//*[(substring(name(),
(string-length(name())-7))='_Forward')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
       or
       (//*[(substring(name(),
(string-length(name())-8))='_Addition')]/*[(substring(name(),
(string-length(name())-6))='_RDS_ID')] = current())
       or
       (count(//*[(substring(name(),
(string-length(name())-6))='_Update')]/*[((substring(name(),
(string-length(name())-6))='_RDS_ID')) and (text() = current())]) &gt;=
2)
       )
       ">The RDS_ID specified in an Update must be unique across all other
Additions, Replacements, Forwards and Updates in the submission</report>
              </rule>
       </pattern>
```

```
        <pattern     name="Model     DIF     key/keyref     constraints"
id="ModelKeyKeyRef">
                <rule
context="Description_Link_Internal/Description_Link_SDS_ID">
                        <assert
test="//Description_Link_Addition/Description_Link_SDS_ID  =  (.)">The
referenced Internal <name/> does not exist in this submission.
                        </assert>
                </rule>
                <rule context="Entity_Internal/Entity_SDS_ID">
                        <assert test="(//Entity_Addition/Entity_SDS_ID =
(.)) or (//Entity_Forward/Entity_SDS_ID = (.))">The referenced Internal
<name/> does not exist in this submission.
                        </assert>
                </rule>
                <rule
context="Characteristic_Internal/Characteristic_SDS_ID">
                        <assert
test="(//Characteristic_Addition/Characteristic_SDS_ID   =   (.))   or
(//Characteristic_Forward/Characteristic_SDS_ID = (.))">The referenced
Internal <name/> does not exist in this submission.
                        </assert>
                </rule>
                <rule
context="Text_Description_Internal/Text_Description_SDS_ID">
                        <assert
```

```
test="//Text_Description_Addition/Text_Description_SDS_ID = (.)">The
referenced Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
                <rule
context="Embedded_Object_Internal/Embedded_Object_SDS_ID">
                    <assert
test="//Embedded_Object_Addition/Embedded_Object_SDS_ID = (.)">The
referenced Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
                <rule context="Process_Internal/Process_SDS_ID">
                    <assert test="(//Process_Addition/Process_SDS_ID
= (.)) or (//Process_Forward/Process_SDS_ID = (.))">The referenced
Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
                <rule
context="Process_Group_Internal/Process_Group_SDS_ID">
                    <assert
test="(//Process_Group_Addition/Process_Group_SDS_ID = (.)) or
(//Process_Group_Forward/Process_Group_SDS_ID = (.))">The referenced
Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
                <rule context="Event_Internal/Event_SDS_ID">
```

```
                    <assert   test="//Event_Addition/Event_SDS_ID   =
(.)">The referenced Internal <name/> does not exist in this submission.
                    </assert>
              </rule>
              <rule
context="Process_Group_Package_Internal/Process_Group_Package_SDS_ID"
>
                    <assert
test="//Process_Group_Package_Addition/Process_Group_Package_SDS_ID =
(.)">The referenced Internal <name/> does not exist in this submission.
                    </assert>
              </rule>
              <rule context="Condition_Internal/Condition_SDS_ID">
                    <assert
test="(//Condition_Addition/Condition_SDS_ID      =      (.))      or
(//Condition_Forward/Condition_SDS_ID = (.))">The referenced Internal
<name/> does not exist in this submission.
                    </assert>
              </rule>
              <rule
context="Process_Control_Element_Internal/Process_Control_Element_SDS
_ID">
                    <assert
test="//Process_Control_Element_Addition/Process_Control_Element_SDS_
ID = (.)">The referenced Internal <name/> does not exist in this
submission.
```

```
                    </assert>
                </rule>
                <rule context="Measure_Internal/Measure_SDS_ID">
                    <assert test="(//Measure_Addition/Measure_SDS_ID
= (.)) or (//Measure_Forward/Measure_SDS_ID = (.))">The referenced
Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
                <rule
context="Instance_Data_Set_Internal/Instance_Data_Set_SDS_ID">
                    <assert
test="(//Instance_Data_Set_Addition/Instance_Data_Set_SDS_ID = (.)) or
(//Instance_Data_Set_Forward/Instance_Data_Set_SDS_ID = (.))">The
referenced Internal <name/> does not exist in this submission.
                    </assert>
                </rule>
        </pattern>
        <pattern name="Model DIF - uniqueness constraints"
id="ModelUnique">
                <rule
context="Association_Addition/Association_SDS_ID">
                    <report
test="//Characteristic_Addition/Characteristic_SDS_ID = (.)">invalid
<name/> value</report>
                </rule>
        </pattern>
```

```
        <pattern name="Model DIF - When the Stereotype element has value
'Other' there must be a Type element" id="ModelStereotypeType1">
                <rule context="Association_Data">
                        <report test="(Association_Stereotype = 'Other')
and (not(Association_Type))">
        When the Association_Stereotype element has value 'Other' there
must be an Association_Type element</report>
                </rule>
                <rule context="Entity_Data">
                        <report test="(Entity_Stereotype = 'Other') and
(not(Entity_Type))">
        When the Entity_Stereotype element has value 'Other' there must
be an Entity_Type element</report>
                </rule>
                <rule context="Interaction_Data">
                        <report test="(Interaction_Stereotype = 'Other')
and (not(Interaction_Type))">
        When the Interaction_Stereotype element has value 'Other' there
must be an Interaction_Type element</report>
                </rule>
                <rule context="Measure_Data">
                        <report test="(Measure_Stereotype = 'Other') and
(not(Measure_Type))">
        When the Measure_Stereotype element has value 'Other' there must
be a Measure_Type element</report>
                </rule>
```

```
            <rule context="Process_Data">
                  <report test="(Process_Stereotype = 'Other') and
(not(Process_Type))">
      When the Process_Stereotype element has value 'Other' there must
be a Process_Type element</report>
            </rule>
            <rule context="Process_Group_Data">
                  <report     test="(Process_Group_Stereotype     =
'Other') and (not(Process_Group_Type))">
      When the Process_Group_Stereotype element has value 'Other' there
must be a Process_Group_Type element</report>
            </rule>
            <rule context="Process_Group_Instance_Data">
                  <report test="(Process_Group_Instance_Stereotype
= 'Other') and (not(Process_Group_Instance_Type))">
      When the Process_Group_Instance_Stereotype element has value
'Other' there must be a Process_Group_Instance_Type element</report>
            </rule>
      </pattern>
      <pattern name="Model DIF - When there is no Stereotype element there
must be a Type element" id="ModelStereotypeType2">
            <rule context="Association_Data">
                  <report  test="(not(Association_Stereotype))  and
(not(Association_Type))">
      When there is no Association_Stereotype element there must be an
Association_Type element</report>
```

```
            </rule>
            <rule context="Entity_Data">
                  <report    test="(not(Entity_Stereotype))    and
(not(Entity_Type))">
      When there is no Entity_Stereotype element there must be an
Entity_Type element</report>
            </rule>
            <rule context="Interaction_Data">
                  <report  test="(not(Interaction_Stereotype))  and
(not(Interaction_Type))">
      When there is no Interaction_Stereotype element there must be an
Interaction_Type element</report>
            </rule>
            <rule context="Measure_Data">
                  <report    test="(not(Measure_Stereotype))    and
(not(Measure_Type))">
      When there is no Measure_Stereotype element there must be a
Measure_Type element</report>
            </rule>
            <rule context="Process_Data">
                  <report    test="(not(Process_Stereotype))    and
(not(Process_Type))">
      When there is no Process_Stereotype element there must be a
Process_Type element</report>
            </rule>
            <rule context="Process_Group_Data">
```

```
                    <report test="(not(Process_Group_Stereotype)) and
(not(Process_Group_Type))">
        When there is no Process_Group_Stereotype element there must be
a Process_Group_Type element</report>
            </rule>
            <rule context="Process_Group_Instance_Data">
                <report
test="(not(Process_Group_Instance_Stereotype))                    and
(not(Process_Group_Instance_Type))">
        When there is no Process_Group_Instance_Stereotype element there
must be a Process_Group_Instance_Type element</report>
            </rule>
        </pattern>
        <pattern name="Model DIF - When the Stereotype element has value
'Other' the Type element must not contain 'N'A', 'UNK' or 'TBD'"
id="ModelStereotypeType3">
            <rule context="Association_Data">
                <report test="(Association_Stereotype = 'Other')
and ((Association_Type = 'N/A') or (Association_Type = 'TBD') or
(Association_Type = 'UNK'))">When the Association_Stereotype element has
value 'Other' the Association_Type element must not contain 'N'A', 'UNK'
or 'TBD'</report>
            </rule>
            <rule context="Entity_Data">
                <report test="(Entity_Stereotype = 'Other') and
((Entity_Type = 'N/A') or (Entity_Type = 'TBD') or (Entity_Type =
```

```
'UNK'))">When the Entity_Stereotype element has value 'Other' the
Entity_Type element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Interaction_Data">
                  <report test="(Interaction_Stereotype = 'Other')
and ((Interaction_Type = 'N/A') or (Interaction_Type = 'TBD') or
(Interaction_Type = 'UNK'))">When the Interaction_Stereotype element has
value 'Other' the Interaction_Type element must not contain 'N'A', 'UNK'
or 'TBD'</report>
            </rule>
            <rule context="Measure_Data">
                  <report test="(Measure_Stereotype = 'Other') and
((Measure_Type = 'N/A') or (Measure_Type = 'TBD') or (Measure_Type =
'UNK'))">When the Measure_Stereotype element has value 'Other' the
Measure_Type element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Process_Data">
                  <report test="(Process_Stereotype = 'Other') and
((Process_Type = 'N/A') or (Process_Type = 'TBD') or (Process_Type =
'UNK'))">When the Process_Stereotype element has value 'Other' the
Process_Type element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Process_Group_Data">
                  <report     test="(Process_Group_Stereotype     =
'Other') and ((Process_Group_Type = 'N/A') or (Process_Group_Type =
'TBD')     or     (Process_Group_Type     =     'UNK'))">When     the
```

```
Process_Group_Stereotype element has value 'Other' the
Process_Group_Type element must not contain 'N'A', 'UNK' or
'TBD'</report>
            </rule>
            <rule context="Process_Group_Instance_Data">
                  <report test="(Process_Group_Instance_Stereotype
= 'Other') and ((Process_Group_Instance_Type = 'N/A') or
(Process_Group_Instance_Type = 'TBD') or (Process_Group_Instance_Type =
'UNK'))">When the Process_Group_Instance_Stereotype element has value
'Other' the Process_Group_Instance_Type element must not contain 'N'A',
'UNK' or 'TBD'</report>
            </rule>
      </pattern>
      <pattern name="Model DIF - When there is no Stereotype element the
Type element must not contain 'N'A', 'UNK' or 'TBD'"
id="ModelStereotypeType4">
            <rule context="Association_Data">
                  <report test="(not(Association_Stereotype)) and
((Association_Type = 'N/A') or (Association_Type = 'TBD') or
(Association_Type = 'UNK'))
      ">When there is no Association_Stereotype element the
Association_Type element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Entity_Data">
                  <report test="(not(Entity_Stereotype)) and
((Entity_Type = 'N/A') or (Entity_Type = 'TBD') or (Entity_Type = 'UNK'))
```

```
">When there is no Entity_Stereotype element the Entity_Type
element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Interaction_Data">
                  <report test="(not(Interaction_Stereotype)) and
((Interaction_Type = 'N/A') or (Interaction_Type = 'TBD') or
(Interaction_Type = 'UNK'))
      ">When there is no Interaction_Stereotype element the
Interaction_Type element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Measure_Data">
                  <report test="(not(Measure_Stereotype)) and
((Measure_Type = 'N/A') or (Measure_Type = 'TBD') or (Measure_Type =
'UNK'))
      ">When there is no Measure_Stereotype element the Measure_Type
element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Process_Data">
                  <report test="(not(Process_Stereotype)) and
((Process_Type = 'N/A') or (Process_Type = 'TBD') or (Process_Type =
'UNK'))
      ">When there is no Process_Stereotype element the Process_Type
element must not contain 'N'A', 'UNK' or 'TBD'</report>
            </rule>
            <rule context="Process_Group_Data">
                  <report test="(not(Process_Group_Stereotype)) and
```

```
((Process_Group_Type = 'N/A') or (Process_Group_Type = 'TBD') or
(Process_Group_Type = 'UNK'))
       ">When there is no Process_Group_Stereotype element the
Process_Group_Type element must not contain 'N'A', 'UNK' or
'TBD'</report>
              </rule>
              <rule context="Process_Group_Instance_Data">
                     <report
test="(not(Process_Group_Instance_Stereotype))                    and
((Process_Group_Instance_Type = 'N/A') or (Process_Group_Instance_Type
= 'TBD') or (Process_Group_Instance_Type = 'UNK'))
       ">When there is no Process_Group_Instance_Stereotype element the
Process_Group_Instance_Type element must not contain 'N'A', 'UNK' or
'TBD'</report>
              </rule>
       </pattern>
       <pattern name="Model DIF - When there is no Stereotype element,
there must be a Description" id="ModelStereotypeDescription1">
              <rule context="Association_Data">
                     <report  test="(not(Association_Stereotype))  and
((not(Description_Link_Existing))                                 and
(not(Description_Link_Internal)))">
       When there is no Association_Stereotype element there must be a
Description_Link_Existing  element  or  a  Description_Link_Internal
element</report>
              </rule>
```

```
            <rule context="Measure_Data">
                  <report    test="(not(Measure_Stereotype))    and
((not(Description_Link_Existing))                                and
(not(Description_Link_Internal)))">
      When there is no Measure_Stereotype element there must be a
Description_Link_Existing element or a Description_Link_Internal
element</report>
            </rule>
            <rule context="Process_Group_Data">
                  <report test="(not(Process_Group_Stereotype)) and
((not(Description_Link_Existing))                                and
(not(Description_Link_Internal)))">
      When there is no Process_Group_Stereotype element there must be
a Description_Link_Existing element or a Description_Link_Internal
element</report>
            </rule>
            <rule context="Process_Group_Instance_Data">
                  <report
test="(not(Process_Group_Instance_Stereotype))                   and
((not(Description_Link_Existing))                                and
(not(Description_Link_Internal)))">
      When there is no Process_Group_Instance_Stereotype element there
must    be    a    Description_Link_Existing    element    or    a
Description_Link_Internal element</report>
            </rule>
      </pattern>
```

```
<pattern name="Model DIF - When the Stereotype element has value
'Other', there must be a Description" id="ModelStereotypeDescription2">
<rule context="Association_Data">
<report test="(Association_Stereotype = 'Other')
and ((not(Description_Link_Existing)) and
(not(Description_Link_Internal)))">
When the Association_Stereotype element has value 'Other' there
must be a Description_Link_Existing element or a
Description_Link_Internal element</report>
</rule>
<rule context="Measure_Data">
<report test="(Measure_Stereotype = 'Other') and
((not(Description_Link_Existing)) and
(not(Description_Link_Internal)))">
When the Measure_Stereotype element has value 'Other' there must
be a Description_Link_Existing element or a Description_Link_Internal
element</report>
</rule>
<rule context="Process_Group_Data">
<report test="(Process_Group_Stereotype =
'Other') and ((not(Description_Link_Existing)) and
(not(Description_Link_Internal)))">
When the Process_Group_Stereotype element has value 'Other' there
must be a Description_Link_Existing element or a
Description_Link_Internal element</report>
</rule>
```

```
                <rule context="Process_Group_Instance_Data">
                        <report test="(Process_Group_Instance_Stereotype
=       'Other')       and       ((not(Description_Link_Existing))       and
(not(Description_Link_Internal)))">
        When the Process_Group_Instance_Stereotype element has value
'Other' there must be a Description_Link_Existing element or a
Description_Link_Internal element</report>
                </rule>
        </pattern>
        <pattern name="Model SDS_ID value must not be 'N/A' if the SDS_ID
element is not optional" id="ModelSDS_IDNotOptional">
                <rule abstract="true" id="ModelSDS_IDNotOptional_Rule">
                        <report    test="(.)    =    'N/A'">In    element
<name/>:SDS_ID element cannot contain a value of 'N/A' when the purpose
is Addition, Forward or Internal</report>
                </rule>
                <rule
context="Association_Addition/Association_SDS_ID">
                        <extends rule="ModelSDS_IDNotOptional_Rule"/>
                </rule>
                <rule
context="Characteristic_Addition/Characteristic_SDS_ID">
                        <extends rule="ModelSDS_IDNotOptional_Rule"/>
                </rule>
                <rule
context="Characteristic_Forward/Characteristic_SDS_ID">
```

```
                <extends rule="ModelSDS_IDNotOptional_Rule"/>
            </rule>
            <rule context="Condition_Addition/Condition_SDS_ID">
                <extends rule="ModelSDS_IDNotOptional_Rule"/>
            </rule>
            <rule context="Condition_Forward/Condition_SDS_ID">
                <extends rule="ModelSDS_IDNotOptional_Rule"/>
            </rule>
            <rule
context="Description_Link_Addition/Description_Link_SDS_ID">
                <extends rule="ModelSDS_IDNotOptional_Rule"/>
            </rule>
                </pattern>
        <pattern name="Model RDS_ID value must not be 'N/A' if the RDS_ID
element is not optional" id="ModelRDS_IDNotOptional">
            <rule abstract="true" id="ModelRDS_IDNotOptional_Rule">
                <report    test="(.)    =    'N/A'">In    element
<name/>:RDS_ID element cannot contain a value of 'N/A' when the purpose
is Replacement, Existing or Update</report>
            </rule>
            <rule
context="Association_Replacement/Association_RDS_ID">
                <extends rule="ModelRDS_IDNotOptional_Rule"/>
            </rule>
                </pattern>
       <pattern    name="Model    DIF    Association    Constraints"
```

```
id="ModelAssociation">
            <!-- Not explicitly required according to  FDMS Model DIF
specification -->
            <rule context="Association_Data">
                  <report
test="Entity_Internal[1]/Entity_SDS_ID=Entity_Internal[2]/Entity_SDS_
ID">The two entities in an association must not both have the same
Entity_SDS_ID value
                  </report>
                  <report
test="Entity_Existing[1]/Entity_RDS_ID=Entity_Existing[2]/Entity_RDS_
ID">The two entities in an association must not both have the same
Entity_RDS_ID value
                  </report>
            </rule>
      </pattern>
      <pattern   name="Model   DIF   -   Entity   Constraints   -   1"
id="ModelEntity1">
            <rule context="Entity_Data">
                  <report   test="(Information_Field_Addition   or
Information_Field_Replacement)    and    (not((Entity_Stereotype    =
'Information') or (Entity_Type = 'Information')))">An Entity containing
an       Information_Field_Addition       element       or       an
Information_Field_Replacement   element   must   also   have   an
Entity_Stereotype  or  Entity_Type  element  containing  a  value  of
"Information"</report>
```

```
            </rule>
        </pattern>
        <pattern name="Model DIF - Entity Constraints - 2"
id="ModelEntity2">
            <rule context="Entity_Data[ (Entity_Stereotype =
'Organization') or (Entity_Type = 'Organization') ]
/Entity_Component/Entity_Internal/Entity_SDS_ID">
                <assert test="//Entity_Addition[(Entity_SDS_ID =
current()) and
        ((Entity_Data/Entity_Stereotype = 'Organization') or
(Entity_Data/Entity_Type = 'Organization') or
        (Entity_Data/Entity_Stereotype = 'Equipment') or
(Entity_Data/Entity_Type = 'Equipment') or
        (Entity_Data/Entity_Stereotype = 'Person') or
(Entity_Data/Entity_Type = 'Person'))
        ]">If an Entity has a Stereotype or Type value of 'Organization'
then, if the Entity contains an Entity_Component, the Entity referred to
by the component (via the path
Entity_Component/Entity_Internal/Entity_SDS_ID) must have a Stereotype
or Type value of {'Organization', 'Equipment' or 'Person'}</assert>
            </rule>
        </pattern>
        <pattern name="Model DIF - Entity Constraints - 3"
id="ModelEntity3">
            <rule context="Entity_Data[ (Entity_Stereotype =
'Equipment') or (Entity_Type = 'Equipment') ]
```

```
/Entity_Component/Entity_Internal/Entity_SDS_ID">
                    <assert test="//Entity_Addition[(Entity_SDS_ID =
current()) and
       ((Entity_Data/Entity_Stereotype      =      'Equipment')      or
(Entity_Data/Entity_Type = 'Equipment'))
       ]">If an Entity has a Stereotype or Type value of 'Equipment' then,
if the Entity contains an Entity_Component, the Entity referred to by the
component (via the path Entity_Component/Entity_Internal/Entity_SDS_ID)
must have a Stereotype or Type value of 'Equipment'</assert>
             </rule>
       </pattern>
       <pattern   name="Model   DIF   -   Entity   Constraints   -   4"
id="ModelEntity4">
             <rule   context="Entity_Data[   (Entity_Stereotype   =
'Network')       or       (Entity_Type       =       'Network')       ]
/Entity_Component/Entity_Internal/Entity_SDS_ID">
                    <assert test="//Entity_Addition[(Entity_SDS_ID =
current()) and
       ((Entity_Data/Entity_Stereotype      =      'Equipment')      or
(Entity_Data/Entity_Type = 'Equipment'))
       ]">If an Entity has a Stereotype or Type value of 'Network' then,
if the Entity contains an Entity_Component, the Entity referred to by the
component (via the path Entity_Component/Entity_Internal/Entity_SDS_ID)
must have a Stereotype or Type value of 'Equipment'</assert>
             </rule>
       </pattern>
```

```
<pattern name="Model DIF - Interaction Constraints"
id="ModelInteraction1">
<rule context="Interaction_Data">
<report
test="(((Process_Existing[1]/Process_RDS_ID != 'N/A') and
(Process_Existing[2]/Process_RDS_ID != 'N/A')) or
((Process_Internal[1]/Process_SDS_ID != 'N/A') and
(Process_Internal[2]/Process_SDS_ID != 'N/A')) or
((Process_Existing/Process_RDS_ID != 'N/A') and
(Process_Internal/Process_SDS_ID != 'N/A'))) and
(Interaction_Category != 'Input_Output')">If an interaction is
complete, that is, if both sender and receiver are not 'N/A', then the
Interaction_Category element must contain a value of
'Input_Output'</report>
</rule>
</pattern>
<pattern name="Model DIF - Interaction Constraints"
id="ModelInteraction2">
<rule context="Interaction_Data">
<report
test="((Process_Existing[1]/Process_RDS_ID = 'N/A') and
(Process_Existing[2]/Process_RDS_ID = 'N/A')) or
((Process_Internal[1]/Process_SDS_ID = 'N/A') and
(Process_Internal[2]/Process_SDS_ID = 'N/A')) or
((Process_Existing/Process_RDS_ID = 'N/A') and
(Process_Internal/Process_SDS_ID = 'N/A'))">It is not possible for both
```

```
processes in an interaction to have value 'N/A'</report>
            </rule>
        </pattern>
        <pattern name="Model DIF - Interaction Constraints"
id="ModelInteraction3">
            <rule context="Interaction_Data">
                <report
test="(((Process_Existing[1]/Process_RDS_ID != 'N/A') and
(Process_Existing[2]/Process_RDS_ID = 'N/A')) or
        ((Process_Internal[1]/Process_SDS_ID != 'N/A') and
(Process_Internal[2]/Process_SDS_ID = 'N/A')) or
        ((Process_Existing[following-sibling::Process_Internal]/Proces
s_RDS_ID != 'N/A') and (Process_Internal/Process_SDS_ID = 'N/A')) or
        ((Process_Internal[following-sibling::Process_Existing]/Proces
s_SDS_ID != 'N/A') and (Process_Existing/Process_RDS_ID = 'N/A'))) and
        (Interaction_Category != 'Output')">If an interaction is
incomplete - having a value for the first (sender) process and 'N/A' for
the second (receiver) process - then the Interaction_Category element must
contain a value of 'Output'</report>
            </rule>
        </pattern>
        <pattern name="Model DIF - Interaction Constraints"
id="ModelInteraction4">
            <rule context="Interaction_Data">
                <report
test="(((Process_Existing[1]/Process_RDS_ID = 'N/A') and
```

```
(Process_Existing[2]/Process_RDS_ID != 'N/A')) or
       ((Process_Internal[1]/Process_SDS_ID     =     'N/A')     and
(Process_Internal[2]/Process_SDS_ID != 'N/A')) or
       ((Process_Existing[following-sibling::Process_Internal]/Proces
s_RDS_ID = 'N/A') and (Process_Internal/Process_SDS_ID != 'N/A')) or
       ((Process_Internal[following-sibling::Process_Existing]/Proces
s_SDS_ID = 'N/A') and (Process_Existing/Process_RDS_ID != 'N/A'))) and
       (Interaction_Category  !=  'Input')">If  an  interaction  is
incomplete - having 'N/A' for the first (sender) process and a value for
the second (receiver) process - then the Interaction_Category element must
contain a value of 'Input'</report>
              </rule>
       </pattern>
       <pattern  name="Model  DIF  -  Process  Constraints  -  1"
id="ModelProcess1">
              <rule
context="Entity_Performing/Entity_Internal/Entity_SDS_ID">
                     <assert test="//Entity_Addition[(Entity_SDS_ID =
current()) and
       ((Entity_Data/Entity_Stereotype      =      'Agent')      or
(Entity_Data/Entity_Type = 'Agent') or
       (Entity_Data/Entity_Stereotype    =    'Organization')    or
(Entity_Data/Entity_Type = 'Organization') or
       (Entity_Data/Entity_Stereotype     =     'Equipment')     or
(Entity_Data/Entity_Type = 'Equipment') or
       (Entity_Data/Entity_Stereotype      =      'Person')      or
```

```
(Entity_Data/Entity_Type = 'Person'))
        ]">If an Entity_Performing element (used by Process_Data) contains
an Entity_Internal element, then the Entity_SDS_ID of the Entity_Internal
must point to an Entity_Addition that contains a Stereotype or Type of
value 'Agent', 'Organization', 'Equipment' or 'Person' </assert>
                </rule>
        </pattern>
        <pattern name="Model DIF - Process Control Constraints - 1"
id="ModelProcessControl1">
                <rule context="Process_Control_Data">
                        <assert
test="((count(Process_Control_Element_LHS)       =       1)       and
(count(Process_Control_Element_RHS)          =          0)          and
((Process_Control_Element_LHS//Condition_Existing)                  or
(Process_Control_Element_LHS//Condition_Internal)))
or
        ((count(Process_Control_Element_LHS)      &gt;=      1)      and
(count(Process_Control_Element_RHS)          =          0)          and
((Process_Control_Element_LHS//Condition_Existing)                  or
(Process_Control_Element_LHS//Condition_Internal)                   or
(Process_Control_Element_LHS//Measure_Existing)                     or
(Process_Control_Element_LHS//Measure_Internal))                    and
(Process_Control_Relation)                                          and
(Process_Control_Element_LHS//Process_Control_Element_Constant)     and
(Process_Control_Element_LHS//Process_Control_Element_Constant      !=
'N/A')                                                              and
```

```
(Process_Control_Element_LHS//Process_Control_Element_Constant      !=
'UNK') and
(Process_Control_Element_LHS//Process_Control_Element_Constant      !=
'TBD'))
or
       ((count(Process_Control_Element_LHS)      &gt;=      1)      and
(count(Process_Control_Element_RHS)        &gt;=        1)        and
(Process_Control_Relation))">A  Process_Control_Data  element  must
contain  one  of  the  following  combinations:  ***One
Process_Control_Element_LHS, containing either a Condition_Internal or
a Condition_Existing, and no Process_Control_Element_RHS ***One or more
Process_Control_Element_LHS,  containing  one  or  more  of
{Condition_Internal,      Condition_Existing,      Measure_Internal,
Measure_Existing}      and      a      non-null      value      for
Process_Control_Element_Constant, and no Process_Control_Element_RHS
***One  or  more  Process_Control_Element_LHS,  one  or  more
Process_Control_Element_RHS, and one Process_Control_Relation</assert>
              </rule>
       </pattern>
       <pattern  name="Model  DIF  -  Process  Sequence  Constraints  1"
id="ModelProcessSequence1">
              <rule context="Process_Sequence_Data">
                     <report  test="(count(Sequence_Element)  =  1)  and
(not(
       (Process_Sequence_Part = 'Unconstrained') or
        (Process_Sequence_Part = 'Concurrent') or
```

```
        (Process_Sequence_Part = 'Iteration') or
        (Process_Sequence_Part = 'Continuous')
))">In element <name/>, when there is only one Sequence_Element, the
Process_Sequence_Parts element must contain a value from {Unconstrained,
Concurrent, Iteration, Continuous}</report>
                    <report test="(count(Sequence_Element) = 2) and
(not(
        (Process_Sequence_Part[1]      =      'Unconstrained'      and
Process_Sequence_Part[2] = 'Unconstrained') or
        (Process_Sequence_Part[1]      =      'Concurrent'      and
Process_Sequence_Part[2] = 'Concurrent') or
        (Process_Sequence_Part[1]      =      'Iteration'      and
Process_Sequence_Part[2] = 'Sequential') or
        (Process_Sequence_Part[1]      =      'Iteration'      and
Process_Sequence_Part[2] = 'Concurrent') or
        (Process_Sequence_Part[1]      =      'Continuous'      and
Process_Sequence_Part[2] = 'Sequential') or
        (Process_Sequence_Part[1]      =      'Continuous'      and
Process_Sequence_Part[2] = 'Concurrent') or
        (Process_Sequence_Part[1]      =      'Sequential'      and
Process_Sequence_Part[2] = 'Sequential') or
        (Process_Sequence_Part[1] = 'Start' and Process_Sequence_Part[2]
= 'Prior') or
        (Process_Sequence_Part[1] = 'Start' and Process_Sequence_Part[2]
= 'Start') or
        (Process_Sequence_Part[1] = 'Start' and Process_Sequence_Part[2]
```

```
= 'During') or
        (Process_Sequence_Part[1] = 'Start' and Process_Sequence_Part[2]
= 'End') or
        (Process_Sequence_Part[1] = 'Start' and Process_Sequence_Part[2]
= 'After') or
        (Process_Sequence_Part[1] = 'End' and Process_Sequence_Part[2] =
'Prior') or
        (Process_Sequence_Part[1] = 'End' and Process_Sequence_Part[2] =
'Start') or
        (Process_Sequence_Part[1] = 'End' and Process_Sequence_Part[2] =
'During') or
        (Process_Sequence_Part[1] = 'End' and Process_Sequence_Part[2] =
'End') or
        (Process_Sequence_Part[1] = 'End' and Process_Sequence_Part[2] =
'After')
        ))">In element <name/>, when there are two Sequence_Elements, the
two Process_Sequence_Parts elements must contain values from the
following pairs: {(Unconstrained, Unconstrained), (Concurrent,
concurrent), (Iteration, Sequential), (Iteration, Concurrent),
(Continuous, Sequential), (Continuous, Concurrent), (Sequential,
Sequential), (Start, Prior), (Start, Start), (Start, During), (Start,
End), Start, After), (End, Prior), (End, Start), (End, During), (End, End),
(End, After)}</report>
              </rule>
       </pattern>
</schema>
```

C-10 XMLSpyPlugIn.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "CAssign_Validate"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''
' Schematron Plug-in for XML Spy 4.0
' by Aidan Povedano
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''



Implements IXMLSpyPlugIn


Public Function IXMLSpyPlugIn_OnEvent(ByVal nEventID As Long,
```

```
arrayParameters() As Variant, ByVal pXMLSpy As Object) As Variant
    '
End Function

Public Function IXMLSpyPlugIn_GetUIModifications() As String
    ' GetUIModifications() gets the XML file with the specified
modifications of
    ' the UI from the config.xml file in the plug-in folder
    Dim strPath As String
    strPath = App.Path

    If Len(strPath) > 0 Then
        Dim fso As New FileSystemObject
        Dim file As file

        Set file = fso.GetFile(strPath & "¥config.xml")

        If (Not (file Is Nothing)) Then
            Dim stream As TextStream
            Set stream = file.OpenAsTextStream(ForReading)

            ' this replaces the token "**path**" from the XML file with
            ' the actual installation path of the plug-in to get the image
file
            Dim strMods As String
            strMods = stream.ReadAll
```

```
            strMods = Replace(strMods, "**path**", strPath)


            IXMLSpyPlugIn_GetUIModifications = strMods
        Else
            IXMLSpyPlugIn_GetUIModifications = ""
        End If
    End If
End Function


Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As
Object)
    If (Not (pXMLSpy Is Nothing)) Then
        If nID = 3 Then 'selection from Schematron menu (3)
            Call AssignSchematronFile(pXMLSpy)
        End If


        If nID = 4 Or nID = 6 Then 'icon on toolbar clicked (6), or selection
from Schematron menu (4)
            Call ValidateInstanceDocument(pXMLSpy)
        End If
    End If
End Sub


Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal
pXMLSpy As Object) As SPYUpdateAction
    IXMLSpyPlugIn_OnUpdateCommand = spyDisable
```

```
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If objSpy.Documents.Count > 0 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
        Else
            IXMLSpyPlugIn_OnUpdateCommand = spyDisable
        End If

        Set objSpy = Nothing
    End If
End Function

Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Schematron plug-in for XML Spy.
Provides validation of an instance document against a Schematron schema."
End Function


Private Sub AssignSchematronFile(ByVal objSpy As Object)

Dim objInstDoc As XMLSpyLib.Document
Set objInstDoc = objSpy.ActiveDocument
```

```
Dim objDocRoot As XMLSpyLib.XMLData
Set objDocRoot = objInstDoc.RootElement

If objDocRoot.HasChildren Then

    'present file open dialog to user...
    Dim objDlg
    Set objDlg = CreateObject("MSComDlg.CommonDialog")
    objDlg.Filter = "Schematron Files (*.sch)|*.sch|XSL Files
(*.xsl)|*.xsl|All Files (*.*)|*.*||"
    objDlg.FilterIndex = 1
    objDlg.ShowOpen

    If Len(objDlg.FileName) > 0 Then

        'get schematron schema file path from user...
        Dim strSchematronFilePath
        strSchematronFilePath = objDlg.FileName

        Call CreatePI(objSpy, "schematron-schema-location",
strSchematronFilePath)

        'save the instance document...
        objInstDoc.Save

    End If
```

```
    Set objDlg = Nothing

Else

    MsgBox "A Schematron schema file cannot be assigned to this document."

End If

'clean up...
Set objInstDoc = Nothing
Set objDocRoot = Nothing

End Sub


Private Sub ValidateInstanceDocument(ByVal objSpy As Object)

' get path of Schematron schema file, as indicated in the
schematron-schema_location
'processing instruction (PI) embedded in the instance document...
Dim blnErr As Boolean
Dim strErr As String
Dim strSchematronSchemaFilePath As String

strSchematronSchemaFilePath = GetPIValue(objSpy,
```

```
"schematron-schema-location", blnErr, strErr)

If blnErr Then

    MsgBox strErr & " " & _
    "To assign a Schematron schema to this document, choose ""Assign
Schema"" from the Schematron menu.", _
    vbOKOnly, "Validation cannot proceed"

Else

    Dim strPath As String
    strPath = App.Path & "¥Schematron_Tools"

    Dim objInstDoc As Document
    Set objInstDoc = objSpy.ActiveDocument

    'get current xsl stylesheet assignment...
    Dim strOriginalXSL
    strOriginalXSL = GetPIValue(objSpy, "xml-stylesheet", blnErr,
strErr)

    Dim objSchDoc As Document
    Set objSchDoc =
objSpy.Documents.OpenFile(strSchematronSchemaFilePath, False)
    objSchDoc.AssignXSL (strPath & "¥schematron-report.xsl"), False
```

```
    objSchDoc.TransformXSL
    objSpy.ActiveDocument.SetPathName strPath & "¥xxx.xsl"
    objSpy.ActiveDocument.Save
    objSpy.ActiveDocument.Close (False)
    objSchDoc.Close (True)

    objInstDoc.AssignXSL (strPath & "¥xxx.xsl"), False
    objInstDoc.TransformXSL
    objSpy.ActiveDocument.SetPathName              strPath              &
"¥schematron-errors.html"
    objSpy.ActiveDocument.Save
    objSpy.ActiveDocument.Close (False)

    objInstDoc.AssignXSL (strPath & "¥verbid.xsl"), False
    objInstDoc.TransformXSL
    objSpy.ActiveDocument.SetPathName strPath & "¥schematron-out.html"
    objSpy.ActiveDocument.Save
    objSpy.ActiveDocument.Close (False)

    'restore original xsl assignment, if necessary...
    If Not (strOriginalXSL = "") Then
       objInstDoc.SetActiveDocument
       Call CreatePI(objSpy, "xml-stylesheet", strOriginalXSL)
    End If

    Dim objDoc As XMLSpyLib.Document
```

```
    For Each objDoc In objSpy.Documents
        If (objDoc.Title = "schematron-frame.html") Then
            objDoc.Close (False)
        End If
    Next

    Dim objResultsDoc As Document
    Set    objResultsDoc    =    objSpy.Documents.OpenFile((strPath    &
"¥schematron-frame.html"), False)
    objResultsDoc.SetActiveDocument
    objResultsDoc.UpdateViews

    Set objInstDoc = Nothing
    Set objSchDoc = Nothing

End If

End Sub


Private Sub CreatePI(ByVal objSpy As Object, ByVal strPIName As String,
ByVal strPIValue As String)
'pre-conditions: target document must be the active document
'                target document must be an xml document
'otherwise run time error occurs
```

```
'remove all existing processing instructions (PI's) of the type to be
created...
Call RemovePIs(objSpy, strPIName)

Dim objInstDoc As XMLSpyLib.Document
Set objInstDoc = objSpy.ActiveDocument

Dim objDocRoot As XMLSpyLib.XMLData
Set objDocRoot = objInstDoc.RootElement

'create the processing instruction (PI) ...
Dim objPI As XMLSpyLib.XMLData
Set objPI = objInstDoc.CreateChild(spyXMLDataPI)
objPI.Name = strPIName
objPI.TextValue = strPIValue

'find the first element in the instance document...
Dim blnFound
blnFound = False
Dim objFirstElement As XMLSpyLib.XMLData
Set objFirstElement = objDocRoot.GetFirstChild(-1)

On Error Resume Next

Do
```

```
    If (objFirstElement.Kind = spyXMLDataElement) Then
       blnFound = True
       Exit Do
    End If

    Set objFirstElement = objDocRoot.GetNextChild

Loop Until (Err.Number - vbObjectError = 1503)

'insert the PI before the first element in the document
'if there are no elements in the document, append the PI to the root

If blnFound Then
   objDocRoot.InsertChild objPI
Else
   objDocRoot.AppendChild objPI
End If

'clean up...
Set objFirstElement = Nothing
Set objPI = Nothing

'clean up...
Set objDocRoot = Nothing
Set objInstDoc = Nothing
```

```
End Sub



Private Sub RemovePIs(ByVal objSpy As Object, ByVal strPIName As String)
'pre-condition: target document must be the active document
'otherwise run time error occurs

Dim objInstDoc As XMLSpyLib.Document
Set objInstDoc = objSpy.ActiveDocument

Dim objDocRoot As XMLSpyLib.XMLData
Set objDocRoot = objInstDoc.RootElement

If objDocRoot.HasChildren Then

   Dim objPIIterator As XMLSpyLib.XMLData
   Set objPIIterator = objDocRoot.GetFirstChild(-1)

   On Error Resume Next

   Do
      If (objPIIterator.Kind = spyXMLDataPI And objPIIterator.Name =
strPIName) Then

         objDocRoot.EraseCurrentChild
```

```
            Set objPIIterator = objDocRoot.GetCurrentChild

        Else

            Set objPIIterator = objDocRoot.GetNextChild

        End If

    Loop Until (Err.Number - vbObjectError = 1503)

    Set objPIIterator = Nothing

End If

'clean up...
Set objDocRoot = Nothing
Set objInstDoc = Nothing

End Sub

Private Function GetPIValue(ByVal objSpy As Object, ByVal strPIName As
String, blnError As Boolean, strError As String) As String
'pre-condition: target document must be the active document

blnError = False
strError = "No error"
```

```
Dim strReturnString

strReturnString = ""


Dim blnFound

blnFound = False


Dim blnDuplicatesFound

blnDuplicatesFound = False


Dim objDocRoot As XMLSpyLib.XMLData

Set objDocRoot = objSpy.ActiveDocument.RootElement


If objDocRoot.HasChildren Then


    Dim objPIIterator As XMLSpyLib.XMLData

    Set objPIIterator = objDocRoot.GetFirstChild(-1)


    On Error Resume Next


    Do


        If Not blnFound Then


            If (objPIIterator.Kind = spyXMLDataPI And objPIIterator.Name
= strPIName) Then
```

```
                strReturnString = objPIIterator.TextValue
                blnFound = True

            End If

        Else

            '''find duplicates...
            If (objPIIterator.Kind = spyXMLDataPI And objPIIterator.Name
= strPIName) Then
                blnDuplicatesFound = True
                Exit Do
            End If

        End If

        Set objPIIterator = objDocRoot.GetNextChild

    Loop Until (Err.Number - vbObjectError = 1503)

    Set objPIIterator = Nothing

End If

If Not blnFound Then
```

```
    blnError = True
    strError = "No Schematron schema file is assigned to this document."
End If

If blnDuplicatesFound Then
    blnError = True
    strError = "More than one Schematron schema file is assigned to this
document."
End If

Set objDocRoot = Nothing

GetPIValue = strReturnString

End Function
```

APPENDIX D

VALIDATION AGAINST W3C XML SCHEMAS AND SCHEMATRON SCHEMAS WITHIN THE XML SPY 4.0 IDE

This appendix describes the process of validating an instance document against a W3C XML Schema or a Schematron schema, within the XML Spy 4.0 Integrated Development Environment (IDE).

D-1 Setting up the XML Spy IDE

XML Spy has full support for the W3C XML Schema language. No setup is required in order to validate against a W3C XML Schema.

XML Spy does not support the Schematron language. In order to validate against a Schematron schema, using the method described in this appendix, the Schematron4XMLSpy4 plug-in must first be added to XML Spy 4.0. This is done as follows:

1. Download and install Instant-Saxon 6.0.2 from the XML Spy website.
2. When prompted by the set-up program, choose to make Instant-Saxon 6.0.2 the default XSLT processor for XML Spy (This is easily changed later - go to Tools->Options->XSL in XML Spy)
3. Copy the Schematron4XMLSpy4 folder (available from author, by request) to hard drive
4. In XML Spy, choose Tools → Customize → Plug-Ins from the XML Spy menu
5. Click "Add Plug In..."

6. In the pop-up dialog, select the following file: XMLSpyPlugIn2.dll. (This file is located in the folder Schematron4XMLSpy4 / XMLSpyPlugIn)

D-2 Validation against a W3C XML Schema

1. In XML Spy, open the instance document to be validated.
2. Assign a W3C XML Schema to the instance document. This is done as follows:
   - Choose DTD/Schema → Assign Schema from the XML Spy menu
   - In the pop-up dialog, select the W3C XML Schema to validate against
3. To perform the validation, click the green checkmark on the XML Spy toolbar
4. The result of the validation is either the message "This file is valid" or an error message

To view the W3C XML Schema, open the file in XML Spy and then switch to the Schema Design View, by choosing View → Schema Design View from the XML Spy menu.

D-3 Validation against a Schematron Schema

1. In XML Spy, open the instance document to be validated.
2. Assign a Schematron Schema to the instance document. This is done as follows:
   - Choose Schematron → Assign Schematron File from the XML Spy menu
   - In the pop-up dialog, select the Schematron Schema to validate against
3. To perform the validation, click the purple checkmark on the XML Spy toolbar
4. The result of the validation is a two-frame HTML page. The upper frame lists the validation errors. Clicking on an error displays, in the lower frame, the section of the instance document where the error occurred.

To view the Schematron Schema, open the file in XML Spy and then switch to the Enhanced Grid View, by choosing View → Enhanced Grid View from the XML Spy menu.

If both a W3C XML Schema and a Schematron schema are available, two levels of validation are possible:

- Quick validation – just click the green checkmark (validate only against the W3C XML Schema)
- Full validation – click the green checkmark and then click the purple checkmark

D-4 Screen-shot Walkthrough of Validation Process

The following are demonstrated:

- Open instance document (Figure D-1)
- Assign a W3C XML Schema to the instance document (Figures D-2 and D-3)
- Validate the instance document against the W3C XML Schema (Figure D-4)
- Results of W3C XML Schema validation (Figure D-5)
- Assign a Schematron schema to the instance document (Figures D-6 and D-7)
- Validate the instance document against the Schematron schema (Figure D-8)
- Results of Schematron validation (Figure D-9)
- Scroll through results of Schematron validation (Figure D-10)
- Click on an error in the upper frame to display location of error in document (Figure D-11)
- View W3C XML Schema in Schema Design View (Figures D-12 and D-13)
- View Schematron schema in Enhanced Grid View (Figure D-14)

Figure D- 1 Open Instance Document

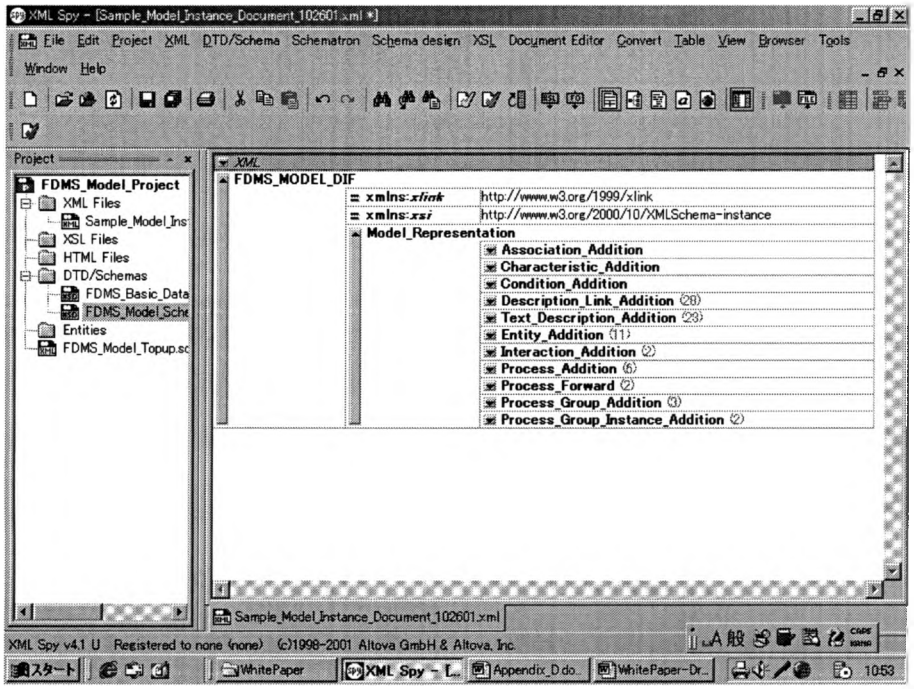


Figure D- 2 Assign a W3C XML Schema to the Instance Document

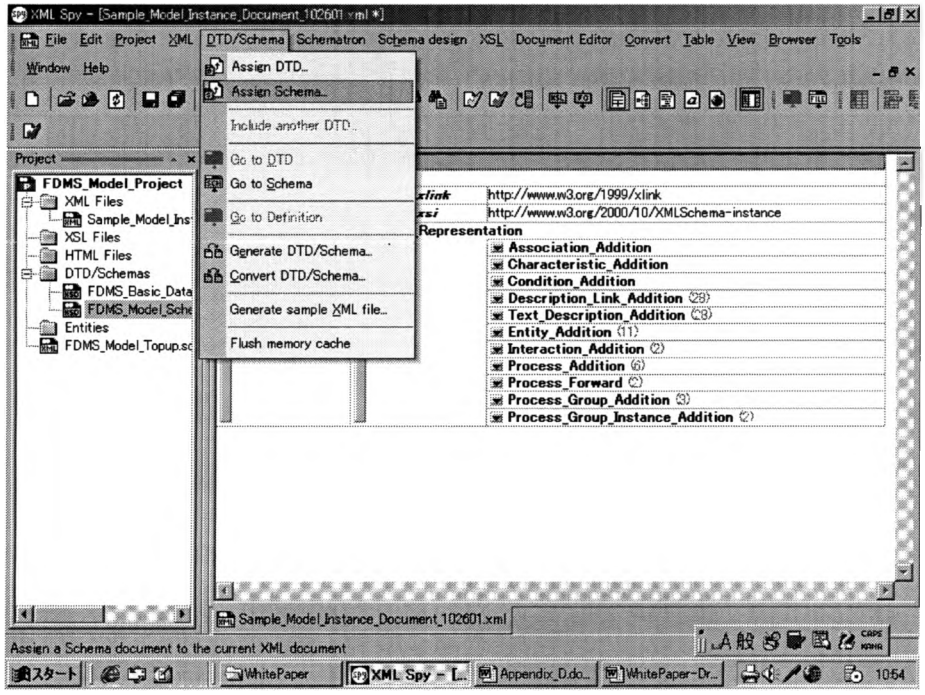


Figure D- 3 Assign a W3C XML Schema to the Instance Document (cont.)

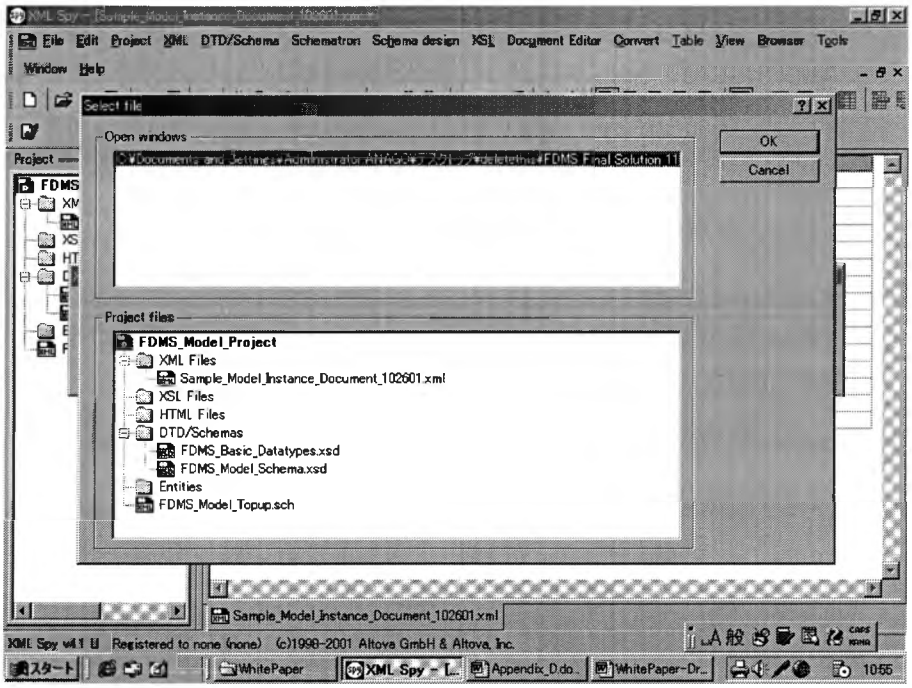


Figure D- 4 Validate the Instance Document against the W3C XML Schema

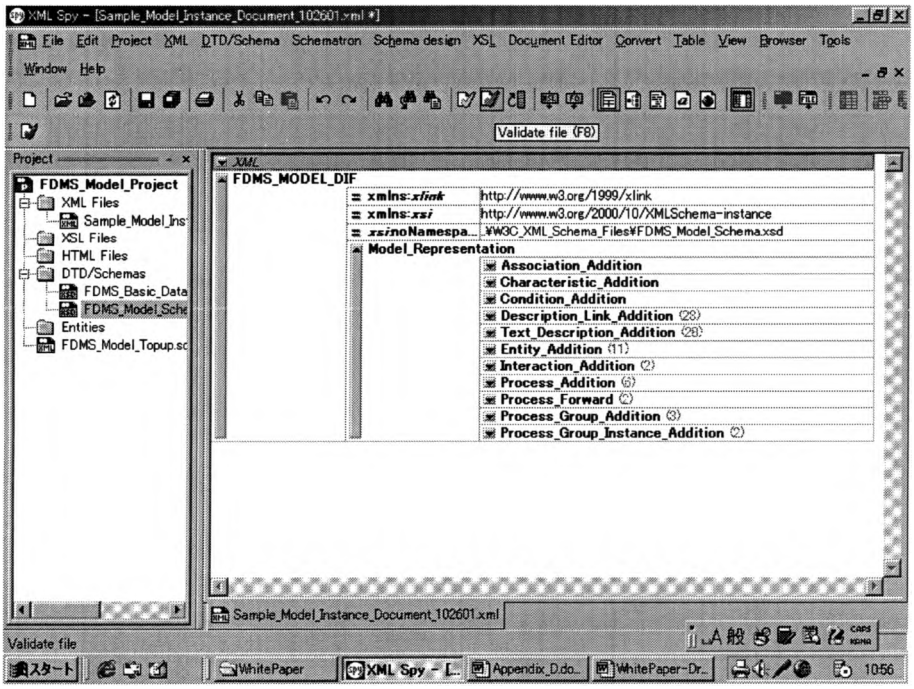


Figure D- 5 Results of W3C XML Schema Validation

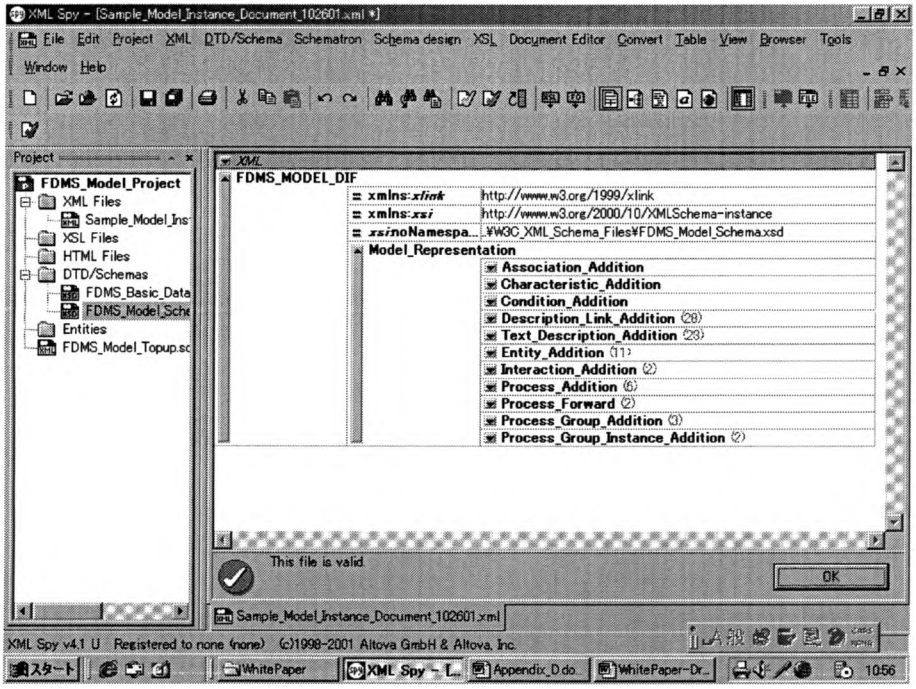


Figure D- 6 Assign a Schematron schema to the Instance Document

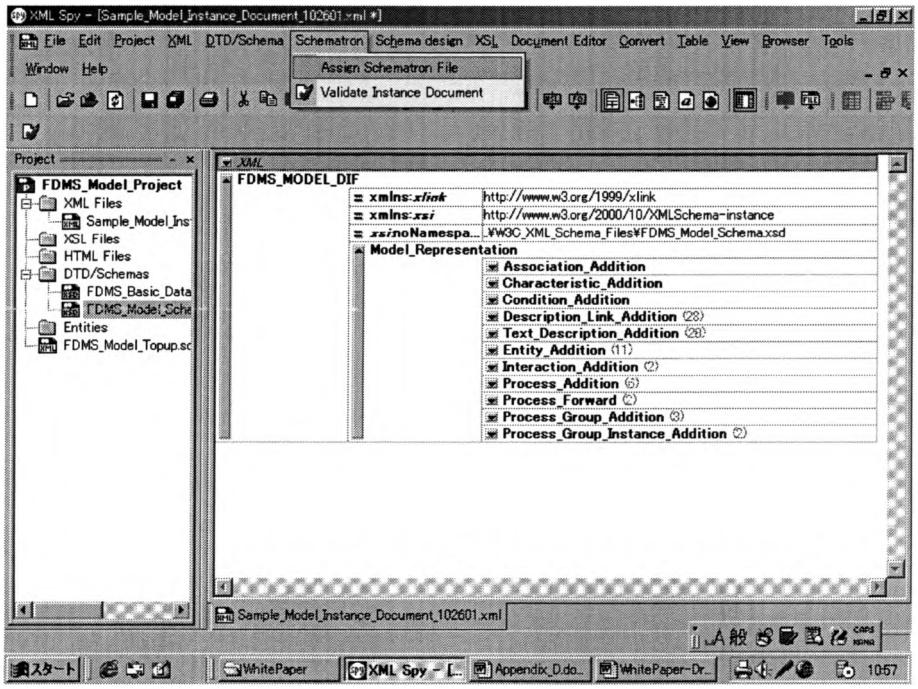


Figure D- 7 Assign a Schematron schema to the Instance Document (cont.)

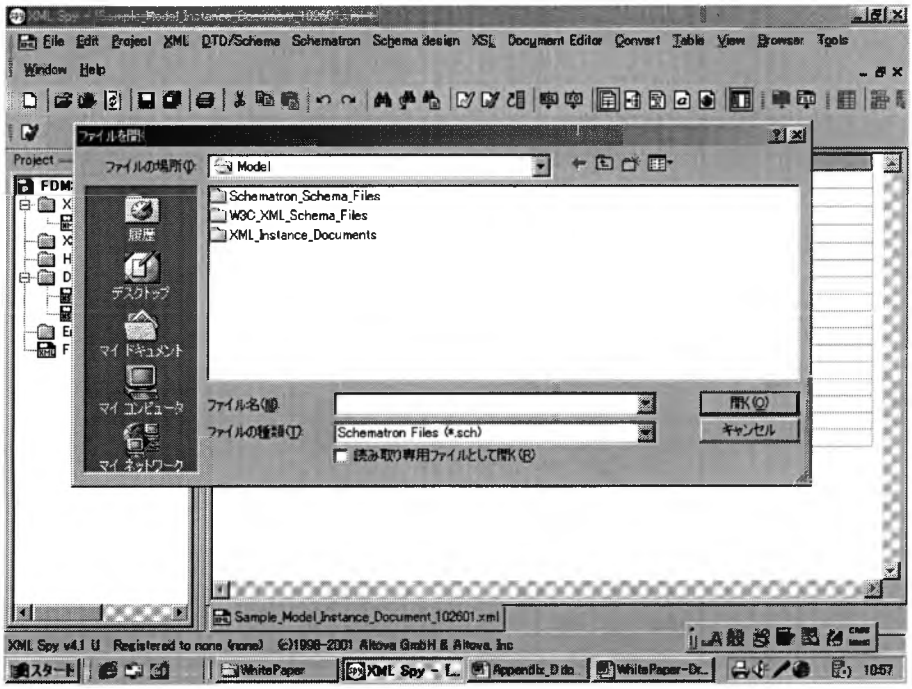


Figure D- 8 Validate the Instance Document Against the Schematron schema

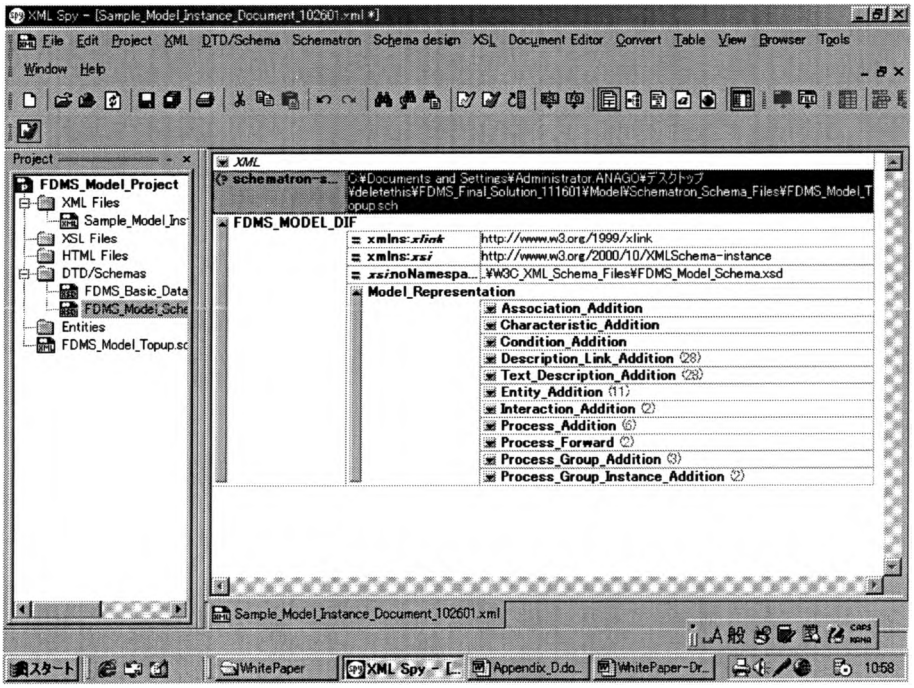


Figure D- 9 Results of Schematron Validation

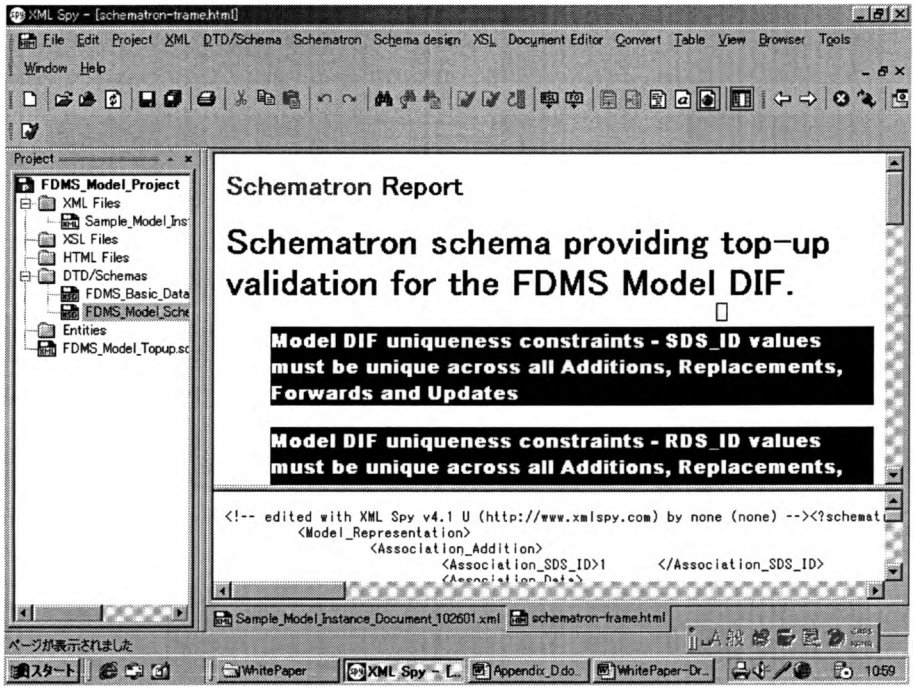


Figure D- 10 Scroll Through Results of Schematron Validation

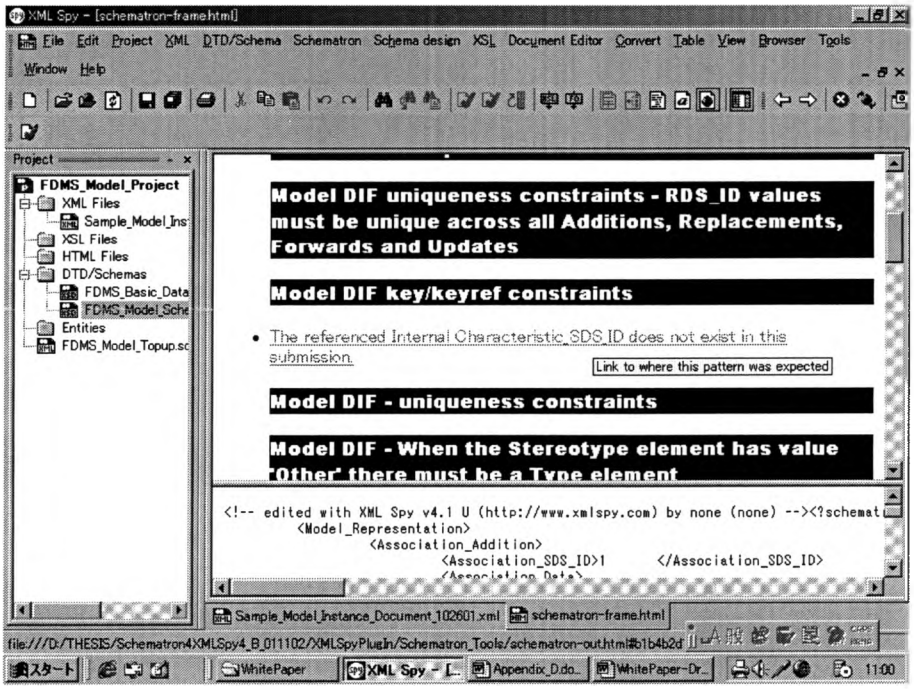


Figure D- 11 Clicking on Error in Upper Frame Displays Location of Error in Document

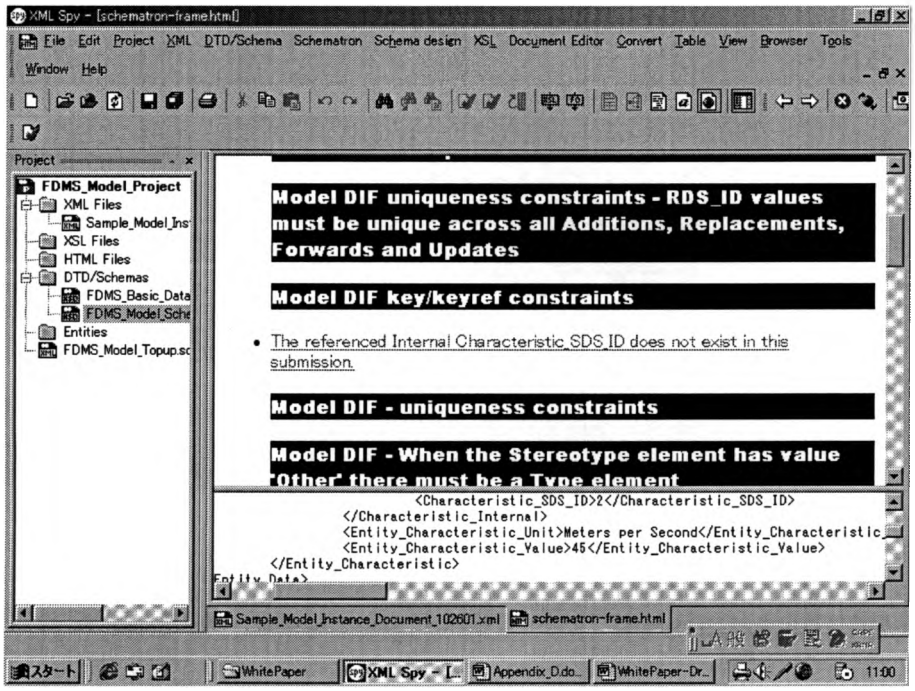


Figure D- 12 View W3C XML Schema in Schema Design View

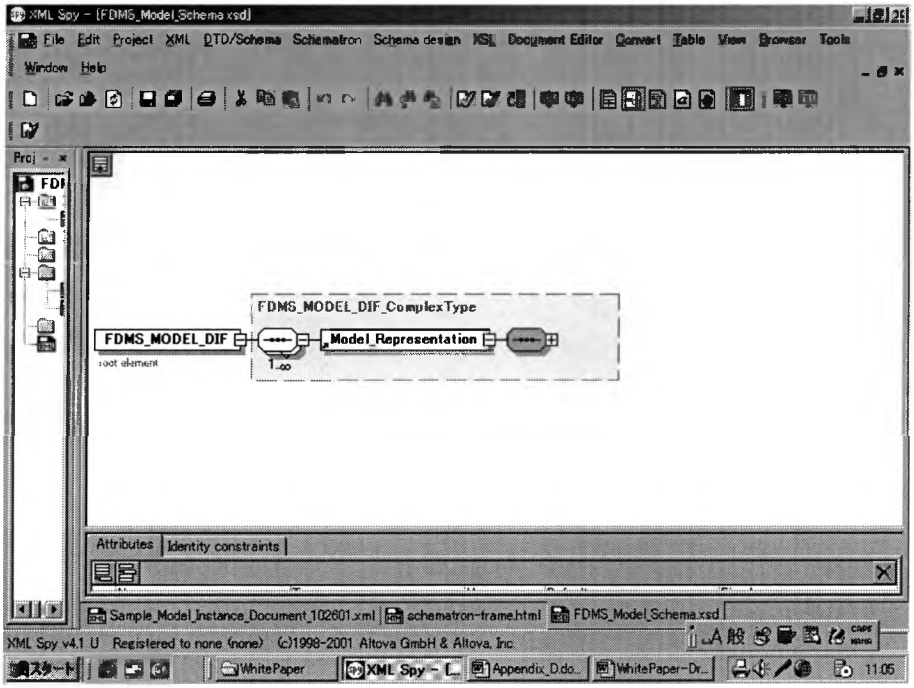


Figure D- 13 View W3C XML Schema in Schema Design View (cont.)

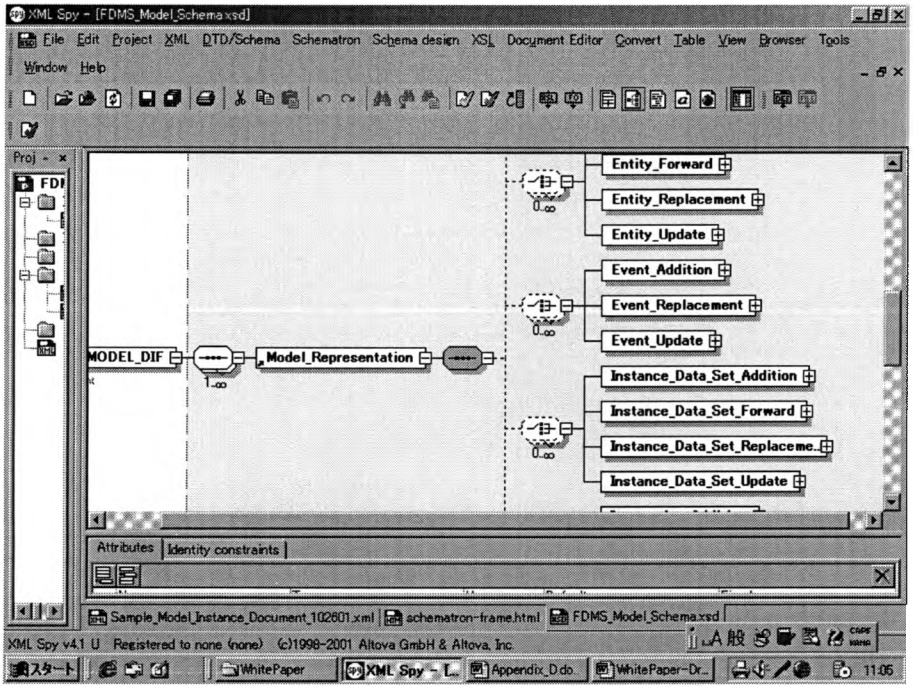
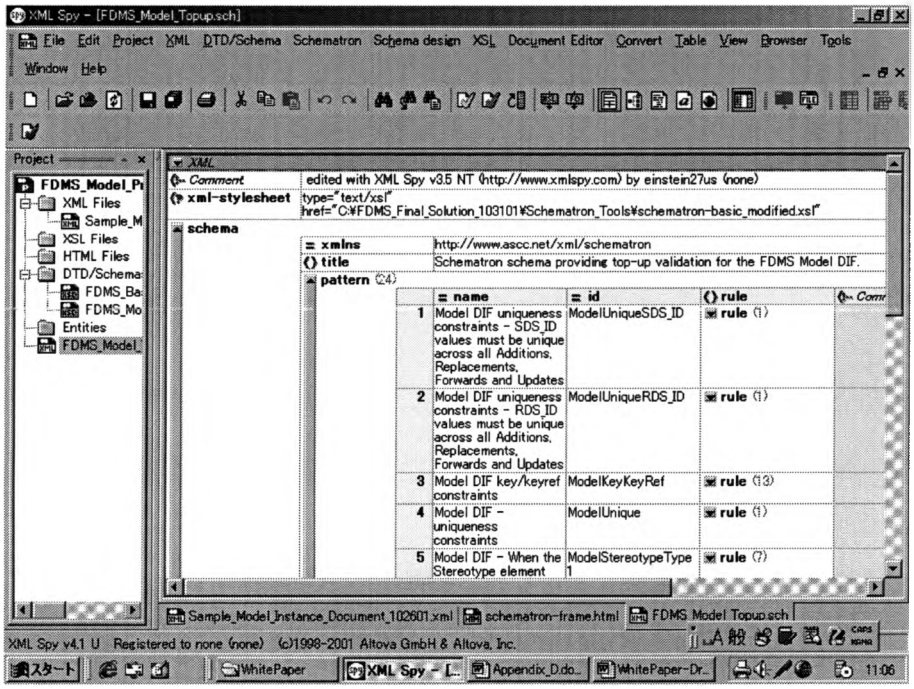


Figure D- 14 View Schematron Schema in Enhanced Grid View



BIBLIOGRAPHY

FDMS

[FDMS1]

F. Haddix, J. Sheehan, M. Loesekann and R. Scrudder, "Functional Descriptions of the Mission Space (FDMS) Common Semantics and Syntax (CS&S) Version 1.6", Defense Modeling and Simulation Office, November 2000

[FDMS2]

F. Haddix, J. Sheehan, M. Loesekann and R. Scrudder, "Functional Descriptions of the Mission Space (FDMS) Model Representation Data Interchange Format (DIF) Version 1.6", Defense Modeling and Simulation Office, November 2000

[FDMS3]

F. Haddix, J. Sheehan, M. Loesekann and R. Scrudder, "Functional Descriptions of the Mission Space (FDMS) Meta-Data Representation Point Of Contact (POC) Data Interchange Format (DIF) Version 1.6", Defense Modeling and Simulation Office, November 2000

Language Comparison

[LANGCOMP1]

L. Dongwon and W.W. Chu, "Comparative Analysis of Six XML Schema Languages", ACM

SIGMOD Record 29(3), September 2000

[LANGCOMP2]

L. Alschuler and M. Walter, "XML Five Years On: Simplicity Gives Way to Complexity", Seybold Report Analyzing Publishing Technologies, Vol. 1 Number 23, March 11 2002

[LANGCOMP3]

J. Hunter and L. Armstrong, "A Comparison of Schemas for Video Metadata Representation", Computer Networks, Vol. 31, 1999

XML Technologies

General Reference

[WROX]

J. Pinnock, et al, "Professional XML", 2nd.Edition, Wrox Press, May 2001

XML

[XML1]

T. Bray, J. Paoli, et al, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, *Internet document*, October 2000

(http://www.w3.org/TR/REC-xml)

[XML 2]

N. Walsh, "A Technical Introduction to XML", XML.com, *Internet document,* October 1998

(http://www.xml.com/pub/a/98/10/guide0.html)

[XML3]

A. Bergholz, "Extending Your Markup: An XML Tutorial", IEEE Internet Computing,

July/August 2000

XSLT

[XSLT1]

J. Clark (ed.), "XSL Transformations (XSLT) Version 1.0", W3C Recommendation, *Internet document,* November 1999

(http://www.w3.org/TR/xslt)

[XSLT2]

M. H. Kay, "XSLT Programmer's Reference", 2nd Edition, Wrox Press, April 2001

XPath

[XPATH1]

J. Clark and S. DeRose, "XML Path Language (XPath) Version 1.0", W3C Recommendation, *Internet document,* November 1999

(http://www.w3.org/TR/xpath)

[XPATH2]

E. Wilde and D. Lowe, "XPath, XLink, XPointer, and XML: A Practical Guide to Web Hyperlinking and Transclusion", Addison-Wesley, 2002

DTD

[DTD1]

N.Walsh, "DTDs", XML.com, *Internet document,* July 1999

(http://www.xml.com/lpt/a/1999/07/schemas/dtds.html)

Schema Languages

TREX

[TREX1]

J.Clark, "TREX - Tree Regular Expressions for XML Tutorial", *Internet document*, 2001

(http://www.thaiopensource.com/trex/tutorial.html)

[TREX2]

J.D. Eisenberg, "TREX Basics", XML.com, *Internet document,* April 2001

(http://www.xml.com/pub/a/2001/04/11/trex.html)

[TREX3]

TREX home page

http://www.thaiopensource.com/trex/

RELAX

[RELAX1]

M. Murata, "How to RELAX", *Internet document*, August 2000

(http://www.xml.gr.jp/relax/html4/howToRELAX_full_en.html)

[RELAX2]

J.D. Eisenberg,"Validating XML with RELAX", *Internet document*, September 2000

(http://catcode.com/relax_tut/index.html)

[RELAX3]

J.D. Eisenberg, "Learning to RELAX", XML.com, *Internet document,* October 2000

(http://www.xml.com/pub/a/2000/10/16/relax/index.html)

[RELAX4]

RELAX home page

http://www.xml.gr.jp/relax/

W3C XML Schema

[W3CSCH1]

W3C XML Schema Working Group, "XML Schema Part 0: Primer", W3C Recommendation, *Internet Document*, May 2001

(http://www.w3.org/TR/xmlschema-0/)

[W3CSCH2]

E. Van der Vlist, "Using W3C XML Schema", XML.com, *Internet document*, November 2000

(http://www.xml.com/lpt/a/2000/11/29/schemas/part1.html)

[W3CSCH3]

R. Jelliffe, "The W3C XML Schema Specification in Context", XML.com, *Internet document*, January 2001

(http://www.xml.com/pub/a/2001/01/10/schemasincontext.html)

[W3CSCH4]

R.S. Costello, "XML Schema Tutorial", *downloadable Powerpoint presentation,* 2000

(http://www.xfront.com/xml-schema.html)

[W3CSCH5]

D. Mertz, "Comparing W3C XML Schemas and Document Type Definitions", IBM

DeveloperWorks, *Internet document,* March 2001

(http://www-106.ibm.com/developerworks/xml/library/x-matters7.html)

[W3CSCH6]

S. Mohr, et al, "Professional XML Schemas", Wrox Press, July 2001

[W3CSCH7]

D. Kiely, et al, "XML Schema Slowly Matures", XML Magazine, January 2001

[W3CSCH8]

J. Roy and A. Ramanujan, "XML Schema Language: Taking XML to the Next Level", IEEE IT Professional, March/April 2001

[W3CSCH9]

M.H. Needleman, "XML Schema Language: The New Way of Coding XML", Serials Review, Vol. 27, No. 2, 2001

DSD

[DSD1]

N. Klarlund, A.Moller and M.I. Schwartbach, "The DSD Schema Language", *Internet document* (http://www.brics.dk/DSD/journal.pdf)

[DSD2]

DSD home page

http://www.brics.dk/DSD/

[DSD3]

N. Klarlund, A.Moller and M.I. Schwartbach, "DSD: A Schema Language For XML", ACM Press, 2000

Schematron

[SCHTR1]

C. Ogbuji, "Validating XML with Schematron", XML.com, *Internet document*, November 2000 (http://www.xml.com/lpt/a/2000/11/22/schematron.html)

[SCHTR2]

L. Dodds, "Schematron: Validating XML using XPath", *Internet document,* April 2001 (http://www.bath.ac.uk/~ccslrd/papers/schematron_xsltuk.html)

[SCHTR3]

Schematron Tutorial

N. Miloslav, "Schematron Tutorial", *Internet document* (http://www.zvon.org/HTMLonly/SchematronTutorial/General/contents.html)

[SCHTR4]

R. Jelliffe, "Getting Informaton into Markup: the Data Model behind the Schematron Assertion Language", Technical White Paper for GeoTempo, Inc., October 2000 (http://www.sinica.edu.tw/~ricko/schematron.PDF)

[SCHTR5]

Academia Sinica Computing Center's Schematron Home Page

http://www.ascc.net/xml/resource/schematron/schematron.html

[SCHTR6]

E. Robertson, "Combining Schematron With Other XML Schema Languages", Topologi Pty. Limited, February 2002

Examplotron

[EX1]

Examplotron home page

http://www.examplotron.org/

[EX2]

L. Dodds, "Schemas by Example", XML.com, *Internet document,* March 2001

(http://www.xml.com/lpt/a/2001/03/28/deviant.html)

DDML

[DDML1]

R. Bourret, J.Cowan, I. Macherius and S. St. Laurent, "Document Definition Markup Language (DDML) Specification, Version 1.0", W3C, *Internet document*, January 1999

(http://www.w3.org/TR/NOTE-ddml)

XSchema

[XSCH1]

Xschema home page

http://www.simonstl.com/xschema/

DCD

[DCD1]

T. Bray (ed.), C. Frankston (ed.) and A. Malhotra (ed.), "Document Content Description for XML", Submission to W3C, *Internet document*, July 1998

(http://www.w3.org/TR/NOTE-dcd)

XDR

[XDR1]

C. Frankston (ed.) and H.S. Thomson (ed.), "XML Data Reduced", *Internet document,* July 1998

(http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm)

[XDR2]

A. Layman, E. Jung, et al, "XML-Data", W3C Note, *Internet document,* January 1998

(http://www.w3.org/TR/1998/NOTE-XML-data-0105/)

SOX

[SOX1]

A. Davidson, M. Fuchs, M. Hedin, M. Jain, J. Koistinen, C. Lloyd, M. Maloney and K. Schwartzhof, "Schema for Object-oriented XML 2.0", W3C Note, *Internet document*, July 1999

(http://www.w3.org/TR/NOTE-SOX/)

SAF

[SAF1]

S.Vorthmann and J. Robie, "Beyond Schemas – Schema Adjuncts and the Outside World", Extensibility.com online documentation, *Internet document,* 2001

(http://www.extensibility.com/resources/beyond_schemas.htm)

[SAF2]

Extensibility home page

http://www.extensibility.com/resources/saf.htm

RDF

[RDF1]

D. Brickley (ed.) and R.V. Guha (ed.), "Resource Description Framework (RDF) Schema Specification 1.0", W3C Candidate Recommendation, *Internet document* , March 2000 (http://www.w3.org/TR/rdf-schema/)

[RDF2]

D. Fensel (ed.), "The Semantic Web and its Languages", IEEE Intelligent Systems, November/December 2000

Hook

[HOOK1]

R. Jelliffe, "Resource Directory (RDDL) for Hook 0.2", *Internet document,* February 2002 (http://www.ascc.net/xml/hook/)

VITA

Aidan Christopher Povedano was born in Gibraltar, on June 8, 1974, the son of Ernest John Povedano and Jane Frances Povedano. He graduated from the University of Sheffield, United Kingdom, in July, 1995, with the degree of Bachelor of Science in Physics. During the three years following graduation he was employed as an Assistant English Teacher on the JET (Japan Exchange and Teaching) Program in Nagoya, Japan In January 1999, he commenced his studies in computer science at Austin Community College, Austin, Texas, and in August, 1999, he entered the Graduate School of Southwest Texas State University, San Marcos, Texas.

Permanent address: 904B Rio Grande Street
Austin, Texas 78701

This thesis was typed by Aidan Povedano.