

Automated Semantic Correlation between Multiple Schema for Information Exchange

by

David Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the Degrees of

Master of Engineering in Electrical Engineering and Computer Science

and

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 2000

© M.I.T., MM. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part, and to grant others the right to do so.

Signature of Author
Department of Electrical Engineering and Computer Science
May 22, 2000

Certified by
Amar Gupta
Co-Director, PROFIT Initiative
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

This page intentionally left blank.

Automated Semantic Correlation between Multiple Schema for Information Exchange

by

David Wang

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2000 in Partial Fulfillment of the

Requirements for the Degrees of

Master of Engineering in Electrical Engineering and Computer Science
and

Bachelor of Science in Electrical Science and Engineering

Abstract

The X-Map system for automated semantic correlation between multiple schemas represents a novel perspective for mediating XML data interoperability between multiple heterogeneous systems. Instead of using time consuming data-domain modeling or other pre-constructed knowledge about the systems' data-domain, X-Map relies on three techniques to do its job, none of which require pre-construction of anything: relevant definition of equivalence and other classes of relations, speculative evaluation of relations between data-domains, and a regenerative association engine to re-evaluate speculations. Restriction to valid XML data also gives X-Map some advantage, including well-formedness, parsing, and data element identification. Essentially, X-Map uses the definition of equivalence to plant seeds of possible correlation between schema elements, speculates on the relation's correctness, then continuously re-evaluates the correctness of the speculative relations when given additional information from newly introduced schemas. In effect, X-Map generates and harbors a large growing body of knowledge between the elements of the many schemas that it encounters. Furthermore, this knowledge is store in valid XML format (XLink linkbase and standard valid XML), enabling other XML-compliant applications to perform related mediation tasks such as data transformation and translation.

Thesis Supervisor: Dr. Amar Gupta

Title: Co-Director, Productivity From Information Technology (PROFIT) Initiative.

MIT Sloan School of Management.

This page intentionally left blank.

Acknowledgments

The road to completing this thesis has been fast-paced and furiously exciting. However exciting as it may have been, this thesis could not have been possible without the aid and contributions of others, whom I'd like to acknowledge personally, in no particular order.

First, I'd like to thank Michael Ripley for being a great supervisor and going the "extra mile" in providing critical comments, suggestions, and support all throughout the process. Very much appreciated indeed.

I'd also like to thank Roger Costello for being a great supervisor and taking the long hours to bring me up to speed with XML. Without his guidance and well-seasoned explanations, I could not even begin to talk about, much less reason with or use, XML.

I'd like to thank the people on the XML-dev list for offering me incisive answers to my barrage of questions concerning various XML technologies. Without them, I could not have progressed much in my understanding of XML.

I'd like to thank Aditya Damle for his insightful technical discussions with me on the design, implementation, and performance issues related to X-Map. Many of the X-Map's key features resulted from his influence.

I'd like to thank Cindy Wang, my sister, for her tireless efforts to bring my jumble of tech-babble into proper English. The astute reader should also be thankful.

I'd like to thank Dr. Amar Gupta, my thesis supervisor, for being overly patient and accommodating with me and also pushing me along at the right times with the right advice and suggestions to even complete this thesis on time!

I'd also like to thank Dingli Chen for her efforts to introduce Dr. Gupta to me and get everything rolling.

And of course, all Course VIers should thank Anne Hunter for being the most amazing course administrator and general savant; I do not know what it'd be like without her administering Course VI students, but I do not even want to think of that scenario.

I must admit that Classical Music rules! Writing, deciphering, and otherwise editing text should be done while immersed in fine Classical Music by musical geniuses such as Mozart, Haydn, Beethoven, and Chopin.

Last of all, I'd like to thank my mother, without whom any of this would not have been possible.

Look, I'm all done!

/David May 22, 2000

Contents

1	Introduction	11
1.1	Interoperability	12
1.1.1	Traditional Interoperability	12
1.1.2	Interoperability using Approach #1	12
1.1.3	Interoperability using Approach #2	13
1.2	Applications of Interoperability	13
1.3	Goals and Assumptions	14
1.4	Thesis Outline	15
2	Background	17
2.1	Literature Survey	18
2.1.1	Issues with Data Heterogeneity	19
2.1.2	Traditional Interoperability Approaches	21
2.2	Interoperability Frameworks	23
2.2.1	Protégé	24
2.2.2	COntext INterchange	25
2.3	Interoperability Technologies	26
2.3.1	eXtensible Markup Language	27
2.3.2	Architectural Form	28
2.3.3	Topic Map	29
2.3.4	XML Linking Language	32

2.3.5	Resource Description Framework	33
2.3.6	XML Schema	35
2.3.7	XML Stylesheet Language Transforms	35
2.4	Summary	36
3	Project Design and Requirements	38
3.1	Company Background	38
3.1.1	Joint Battle InfoSphere	39
3.1.2	Fuselets	40
3.2	Project Description	40
3.3	Design Goals	42
3.3.1	X-Map Design Goals Wish List	43
3.3.2	X-Map Design Goals Rationale	43
3.3.3	X-Map Design Implications	44
3.3.4	Mediator Design Goals Wish List	45
3.3.5	Mediator Design Goals Rationale	45
3.3.6	Mediator Design Implications	46
3.4	Requirements	46
3.4.1	Semi-Automated System to Discover Associations	47
3.4.2	Association Language	48
3.4.3	Transformation Language	49
3.4.4	Mediation Engine	49
3.4.5	Summary of Background	50
4	Project Design and Implementation	51
4.1	Design Overview	53
4.2	X-Map Design and Rationale	53
4.2.1	Enumeration of Relation Classes	57

4.2.2	Discovering Relations	61
4.2.3	X-Map's Strategies	63
4.2.4	Regenerative Association Engine	68
4.2.5	Preservation of Relations for Accessibility	68
4.3	Association Language Description and Rationale	70
4.4	Transform Language	75
4.5	Mediator Design	77
4.6	Summary of Design	79
5	Trade offs and Conclusion	80

List of Figures

2-1	Sample RDF Statement	34
4-1	Design Overview	52
4-2	X-Map Overview	54
4-3	Collapsing Trees into a Directed Graph	62
4-4	XML-2-Bags Pseudo-code	65
4-5	Hierarchical Association Discovery	66
4-6	Cyclical Association Discovery	67
4-7	Association Language Overview	71
4-8	Mediator Design	78

List of Tables

2.1	Data Conflict Taxonomy	20
2.2	Protégé Summary	25
2.3	Useful XML Definitions	27
4.1	Fundamental X-Map Relation Classes	60

Chapter 1

Introduction

Whenever two people want to converse with each other, two ground-rules are implicit in effect. One, they speak the same language¹ and dialect; two, they speak with the same grammar structure.

The same rules apply to two separate processes in a computer system. In order for the two processes to communicate with each other and exchange data, they have to agree to speak the same “language” (i.e. are bytes big-endian or little-endian?) and also the same “grammar structure” (i.e. what is the meaning of each byte sequence?). In effect, these two processes must agree on the same two implicit ground-rules as humans do.

Even more abstractly, the processes themselves can be viewed as entities encompassing specific and distinct data-domains and the data as a collection of relational linkages — that is, in order for data in one data-domain to “make sense” in the other data-domain, the mapping function between the two data-domains must know the language of each domain and have identifiable structures with which to relate.

¹Throughout this thesis, the author will use “language” to mean “the set of words in the appropriate universe of discourse” (the universe depending on context) and “grammar” to mean “the rules that structure the aforementioned language to convey intellectual meaning”. Although Alphabet more succinctly describes the concept described by “language”, it is less commonly understood and will therefore not be used.

1.1 Interoperability

The process by which two separate entities coordinate and ultimately decide on the same implicit ground-rules captures the essence of interoperability. That is, in order for two processes to communicate and exchange data, they must first agree to interoperate with the same language and grammar structure.

This agreement can happen with the following approaches, non-exclusively:

1. **Explicitly:** Both processes agree to interoperate by using a common language and grammar structure.
2. **Implicitly:** External mediators which share a common language and grammar structure “translate” for either/both processes to achieve interoperability. The processes operate as they always do, without explicit recognition of the mediators at work.

1.1.1 Traditional Interoperability

Traditionally, interoperability has been achieved through a combination of *both* approaches — two systems consciously employ mediators to translate their own internal formats to a common language and structure understood by the other — which immediately achieves interoperability. However, this practice has many drawbacks for the systems involved, including external dependencies on the mediators by the systems when either system’s internal structures change.

1.1.2 Interoperability using Approach #1

More recently, interoperability based solely on approach #1 has made some progress, but unfortunately economics and politics dampen its effects. This approach naturally fits industries since it essentially entails an industry-wide agreement on a single language and grammar structure for inter-

operability. Considering that organizations within an industry commonly share the same language, the remaining problems stem mostly from differing grammatical structure. However, politics² and economics commonly corrupt any agreement since the industry leaders often want greater influence over the structural format to give themselves the advantage.

1.1.3 Interoperability using Approach #2

That leaves the less explored, solely #2 approach to interoperability — powerful mediators that dynamically or statically adapt translations between systems, shuttling mapped data back and forth, while ensuring complete independence of the internal formats of all systems involved. At first glance, this approach solves many of the aforementioned problems: the politics and economics of approach #1 are irrelevant since the interoperating systems do not even need to be cognizant of each other, and dynamic adaptation in translation can effectively deal with changes in the internal formats of systems. Thus, it warrants further investigation.

1.2 Applications of Interoperability

In addition to its theoretical importance, interoperability also has many real-life applications. In general, any situation involving more than one entity that wants to communicate with another contains at least one instance of interoperability as outlined in Section 1.1. They are shown in the following examples:

1. Two businesses with different cooperating business products need to exchange data with each other but are designed differently, so they both actively translate to and communicate via a shared common language and protocol.

²Case in point: Biztalk and Open-OASIS are both trying to establish industry-wide sharing of XML schemas, but different organizations with different goals are at play — Biztalk (www.biztalk.org) was founded by Microsoft (www.microsoft.com), while Open-OASIS (www.oasis-open.org) is a non-profit international organization.

2. A startup wants to compete against an established industry giant and translates its output via a mediator to the industry standard to interoperate with the industry leader.
3. Two military branches need to cooperate in a joint battle initiative; passive mediators placed between the communication channels of both branches silently translate data for either side.

Needless to say, the subject of interoperability is important — it underlies all of communication and enables much of computer science and by the previous associations, even life.

1.3 Goals and Assumptions

The exploration of the interoperability approach outlined in Section 1.1.3 forms the focus of this thesis because it offers a rich array of challenging problems, including the specification of an open representation of the mappings to facilitate its use by multiple mediators, which directly affects the larger problem of how to dynamically discover and adapt mappings between systems.

Thus, this thesis will investigate the subject of interoperability between multiple data-domains with only one assumption — the processes involved in interoperability have already agreed to use the same language in communication. This assumption has wide-ranging effects that simplify the investigation by taking the guesswork out of determining a common language. Namely, once a communication language is agreed upon, a mediator only has to worry about *how* to map and represent the identifiable structures in each data-domain in order to facilitate interoperability. Top priority tasks of this thesis include the means of performing these two interoperability tasks.

Now, this assumption is valid and safe to make within the context of this thesis because it assumes that the data-domains are not actively “hostile” toward each other — that is, the data-domains want to be understood by each other. Otherwise, exchanging data between data-domains will be meaningless if the data-domains do not agree a priori on a language — there will be no useful way to identify, map, or even “make sense” of the parts of the data being exchanged.

The common communication language that all data-domains agree to use for this thesis investigation is the eXtensible Markup Language (XML) [7]. XML is a flexible meta-language useful in the creation of new data-describing languages, and it offers additional benefits if the XML document is valid³. As the remainder of this thesis will soon discuss, XML has many properties and benefits that aid interoperability.

1.4 Thesis Outline

The remaining chapters of this thesis is organized as follows:

Chapter 2 provides a summary of existing interoperability literature and research and integrates it in light of recent XML developments. A practical discussion of existing and upcoming platforms and technologies then commences, and their relations to existing interoperability works are noted wherever applicable.

Chapter 3 gives the motivational project behind this thesis and a possible avenue of application. The design goals, requirements, and rationale for each of the project deliverables and how each relates to XML Interoperability is given. The three deliverables, in order of discussion, are X-Map, Association Language, and Mediator. As a compromising requirement, the Mediator further breaks down into the Mediation Engine and an accompanying Transformation Language.

Chapter 4 expounds on the technical details of the deliverables' designs according to their design goals and requirements. An overview of each of the deliverables preface the detailed discussion of each, starting with X-Map, then Association Language, and finally Mediator. Much work remains on the full implementation of this design, but the fundamental outline and framework of action has

³See Section 2.3.1 and Table 2.3 for information on valid and XML, respectively.

been tackled by this thesis.

Chapter 5 closes with a few remarks on our contributions and the trade offs that X-Map makes in achieving XML Interoperability.

Chapter 2

Background

As stated earlier, the main topic of this thesis is how to map identifiable structures between data-domains given the assumption that the data-domains agree to use XML as the common communication language. Collectively, this simplification shall be denoted by the term “XML Interoperability”, which captures the notion of interoperability as enabled through usage of XML. Evidently, this simplification is significant in the following ways:

1. The protocol for exchanging data is a given. That is, the problem of figuring out how to transmit and reconstruct data (including XML) between systems is *not* a part of the XML Interoperability problem. XML Interoperability can be considered only *after* data are transformed into a form useful as XML.
2. The semantics of describing data structure in each data-domain is also a given — this is the huge advantage that XML brings to the interoperability task — its schema definitions for validation and conformation.
3. Understanding *how* structures in one data-domain map to another data-domain is paramount.

Not surprisingly, the subject of interoperability, with or without using XML, has a long history in both the academic and industrial realms. Thus, it is important to note the ramification of existing research and solutions on this investigation before proceeding onto the design goals and requirements of this thesis.

First, a survey of existing literature and research is presented and synthesized to extract the essential approaches that have been taken in the past. Then, short discussions of various present-day interoperability technologies and platforms commence, and their abilities to enable XML Interoperability summarized and judged. Finally, the author makes some allusions to the design and goals of this thesis.

2.1 Literature Survey

The subject of interoperability exists in this world because *no* one system satisfies everything for everyone — thus, everyone rushes to create his own autonomous system to solve his own problems, occasionally thinking of cooperating with others. Besides, as software engineering practices readily admit, such a monolithic system will more likely than not suffer from several potentially fatal problems. To name a few:

- Maintenance nightmares due to complex and multiple dependencies.
- Extra sensitivity to failure since there are more interconnecting parts.
- Difficulty to generate momentum to evolve a “better” design.

However, with autonomous systems, interoperability comes to the forefront due to the variety of differences possible. In their report on the *NSF workshop on Heterogeneous Database Systems* in 1989, Scheuermann et al. [30] noted that there were three different types of autonomy, each with different implications:

Design Autonomy: The ability for a system to choose its *own* information, data-model, and implementation procedures.

Communication Autonomy: The ability for a system to choose what systems to communicate with and what information to convey.

Execution Autonomy: The ability for a system to decide how and when to execute a received request.

Naturally, design autonomy leads to a variety of heterogeneity and conflict between systems. Common practice leads the author to break design autonomy apart and distinguish between *data heterogeneity* (see Section 2.1.1) and *system heterogeneity*. The former refers to the variety of ways in which data can be organized and interpreted while the latter refers to differences in data-models, data manipulation language, etc. Thus, one clearly sees design autonomy as the major obstacle in achieving semantic interoperability¹ amongst heterogeneous systems since the data can be interpreted differently in addition to being modeled and manipulated differently.

Now, communication and execution autonomy pose their own challenges to interoperability, but they represent operational problems outside the semantic-centric scope of this thesis. The same applies to system heterogeneity since XML Interoperability implicitly unifies the data-model and data manipulation language to be XML.

2.1.1 Issues with Data Heterogeneity

The classes of conflict that spring from data heterogeneity partitions into three rough sections, all well documented in literature. Their major differences are in:

1. **Schematic Conflict:** How data are structured or logically organized [23] [24].

¹Semantic interoperability means “the meaningful exchange of information”.

2. **Semantic Conflict:** How data are interpreted (i.e. what does this mean?) [31] [28].
3. **Intensional Conflict:** What context (sources of information available to the sender and receiver) does the data require [18].

One notes that the line between #1 and #2 remains fuzzy despite the preponderance of literature because of one simple fact: schematic structure and organization *often* say something about the semantics of the data, and the semantics of the data at minimum suggest and at most dictate its structure and organization. In other words, the two concepts intertwine with each other.

The remainder of this section reviews the taxonomy of sources for data conflicts (see Table 2.1), which provides a quick synthesis of existing literature on this subject [18].

Schematic	Data Type	Use of different primitive data types to represent the same data value (i.e. String vs. Date for date representation)
	Labelling	Synonyms or homonyms amongst schema elements (i.e. like-named attributes referring to different things or conversely differently named attributes referring to the same thing)
	Aggregation	How should data be clustered together in the data-model (i.e. should a person's name, SSN, and birth date be clustered under name[SSN, birth date] or SSN[name, birth date])?
	Generalization	How should entities relate (i.e one system can have managers and engineers to be distinct types while another can cluster both as employees with different attributes)?
Semantic	Naming	Use of different names for the same thing (i.e. SSN, S.S.N., Social Security Number)
	Scaling	Use of different measures or scales (i.e. English vs. Metric, 4.0 vs. 5.0 GPA scale)
	Confounding	Just plain confusion over what something means (i.e. "latest trade price" could mean real-time, 5 minutes delayed, or 20 minutes delayed)
Intensional	Domain	Difference in the underlying universe of discourse; that is, either a disjoint, strict subset, identical, or other non-trivial intersection
	Integrity	Difference in the assumed invariants between systems (i.e. different uniqueness or range constraints)

Table 2.1: Data Conflict Taxonomy [18]

2.1.2 Traditional Interoperability Approaches

Within the last fifteen years, a plethora of proposals and research prototypes aimed at enabling interoperability amongst autonomous and heterogeneous systems has appeared. Sorting through the hype, two discernible patterns emerge as key differentiators:

1. Difference in the underlying data-model — depending on the data-model, certain schematic and semantic transformations become feasible to resolve certain data conflicts.
2. Differences in the execution strategy — either a tight-coupling strategy or a loose-coupling strategy.

Obviously, difference #1 directly affects difference #2 since different data-models often trade off on execution requirements and intensiveness. Meanwhile, the distinctions between the tight-coupling and loose-coupling strategies succinctly characterizes themselves along two dimensions:

1. *Who* is responsible for identifying the conflicts and how to circumvent them.
2. *When* the conflicts are finally resolved.

Clearly, the two tasks must work in tandem — task #1 must precede task #2 since one must identify the conflicts before resolving them. The remainder of this section will describe the most salient features of these two strategies since they typify current interoperability approaches.

Now, one must keep in mind that the following discussion aims to draw a sharp contrast between the two approaches by emphasizing their trade offs. Most systems in the world fall somewhere in the middle of these two poles and achieve mediocre performance. This is squarely due to the fact that the coupling strategies simply do not leverage each other well since they postulate different premises on how to best capture and manipulate data semantics.

Tight-Coupling Strategy

In this strategy, the job of detecting the conflicts (and the means to resolve them) rests squarely on the shoulders of the systems administrator for the data-domains. The administrator typically defines, *a priori*, a unified and shared schema for all interested parties. Then, the relevant portions of the shared schema are sectioned off and handed to each interested party — sharing common fields are allowed and encouraged. These “mini-schemas” then insulate the schema users from the massive underlying canonical representation that everyone shares.

Obviously, exchanges of information between participating parties using different schemas now become as easy as figuring out the relations between the relevant parts of the shared canonical schema since it is all shared.

Examples of this approach can be seen in the prototypes of Breitbart and Tieman (ADDS) [8] and Templeton, et al. (Mermaid) [32], and more recently, the same strategy has shown up in systems using object-oriented data-models such as Pegasus [1], amongst others.

In virtually all of the above cases, the technique used for conflict resolution is similar to that proposed by Dayal and Hwang [12] and involves the introduction of a supertype that encapsulates all of the necessary functions and fields necessary for interoperability. Basically, tight-coupled strategy involves insane amounts of cooperation amongst all participants and a lumbering but growing unified schema.

Loose-Coupling Strategy

In contrast to the tight-coupling strategy, the loose-coupling approach believes that the creation and maintenance of a shared canonical schema (or even a couple) to be infeasible for any nontrivial number of participants. Thus, instead of resolving all conflicts a priori, conflict detection and resolution are carried out at run-time by the users themselves. This is based on the assumption that

most users only need to interact with a limited subset of the fields in any schema at once, so it may never be necessary to resolve *all* of the conflicts between the schemas up front as the tight coupling strategy proposes.

At first glance, this strategy appears to make conflict resolution a distributed² computational task. However, just as distribution decentralizes computational needs for resolving conflicts, it also decentralizes schema coherency — that is, when schemas and/or semantics change, a new means of resolving this conflict must be distributed to everyone interested. This is a non-trivial task that any Information Systems professional readily confesses. Likewise, the lack of a central schema authority means finding the most efficient database resource exposing the preferred schema is hard.

2.2 Interoperability Frameworks

Pursuant to the buzz surrounding the aforementioned literature and research, frameworks and platforms that support interoperability have sprung up like wildfire. The following sections represent a survey of a small but diverse sampling of academic work on the forefront of semantic interoperability.

Both approaches attempt to organize and otherwise categorize information about their disparate data-domains so that they can be used later for mediation. However, the similarity between their efforts stops there, with Protégé [38] favoring an extensional³ organization of information while COntext INterchange (COIN) [18] prescribes to an intensional⁴ organization of information. As a result, their strategies take completely divergent paths toward the goal of interoperability.

²In general, distributed *anything* is a good thing at the present time.

³Extensional means an explicit statement of fact; that is, every known fact must be expressed.

⁴Intensional means an implicit statement of fact; that is, a set of axioms are used to express the know set of facts.

2.2.1 Protégé

Extensive research at the Knowledge Modeling Group⁵ at Stanford produced Protégé, which is a knowledge-based design and acquisition system for managing multiple and large-scale ontologies. An ontology is a hierarchical structuring of knowledge about things obtained by subcategorizing them according to their essential relevant and/or cognitive qualities. Protégé separates knowledge into classes, instance, slots, and facets, where classes embody the higher-level “concepts” in a data-domain, instances are concrete representations of classes, slots are properties of classes and instances, and facets are properties of slots. Its main power lies in its ability to leverage existing ontologies, enabling system developers to construct large, extensional knowledge bases rapidly. It does this by first allowing them to describe the domain’s knowledge as an ontology and then draw associations between the classes/slots of these ontologies. Furthermore, Protégé can use the RDF specification (see Section 2.3.5) to describe its ontologies.

However, Protégé does not presently have facilities to automate the merging or aligning of ontologies (i.e. automatically drawing equivalence or super/sub-type relationships), nor does it resolve the inevitable namespace issues of using/maintaining multiple ontologies.

Thus, Protégé currently helps as an abstraction layer for drawing one-to-one associations between arbitrary elements of two or more data-domains, visualizing the associations, and optionally storing the knowledge. Table 2.2 best summarizes it:

⁵More information can be obtained from <http://smi-web.stanford.edu/projects/protege/>

Knowledge Description	Classes	“Concepts” in the knowledge domain
	Instances	Specific occurrence of a Class
	Slots	Properties of Classes and Instances
	Facets	Properties of Slots
Concepts	Mapping	Associating elements of two ontologies as equivalent
	Merging	Creating a new element that encompasses the merged elements
	Aligning	Determining the super/sub-type relations between the elements
Functions	Visually create ontologies	
	Visually acquire data for an ontology	
	Expandable platform to access other knowledge-based systems	

Table 2.2: Protégé Summary

2.2.2 COntext INterchange

A result of intensive research into data aggregation done at the Massachusetts Institute of Technology, COntext INterchange⁶ (COIN) introduces a novel twist in performing interoperability. Instead of trying to state all known facts about the systems involved in information exchange (extensional), it favors the intensional description of axioms that underlie the facts about the systems. These axioms allow COIN, when given a mediation request, to reason about what the right thing to do is. In fact, COIN can be formulated with a formal logical mathematical model, thus enabling its correctness to be proven [18].

COIN shuns a semantic data-model’s description of the world due to the variety of problems that plague that fragile model. To increase a semantic data-model’s descriptiveness, more attributes are often added to the model, leading to an explosion in the property list of attributes for the model. Furthermore, the meanings of the appropriate transformation functions and modifiers in each semantic model are often non-obvious, forcing one to resolve these conflicts before meaningful mediation can occur.

Instead, COIN subscribes to a context-based model, where the requested information is interpreted based on the contexts of the source and destination domains. The contexts are described by COIN’s own flavor of mathematical axiom language, which is then “objectified” to enjoy the benefits of being

⁶More information can be obtained from <http://context.mit.edu/~coin/>

object oriented — inheritance, subclassing, and overriding of contexts described by these axioms are now possible, which greatly expands COIN’s ability to reason about how to mediate a given request across data-domains [18].

However, COIN does *not* discover nor keep track of the proper associations between data-domains — it expects the requester to provide it with that initial knowledge via a query so that it can then figure out what the right thing to do is to fulfill that request. Thus, it faces a potentially steep start-up cost as one must populate the axioms for all mediated systems by-hand.

2.3 Interoperability Technologies

It is undeniable that XML has been hyped as the industrial sector’s savior for interoperability. While it is still undetermined whether XML lives up to its hype, this much is certain — it certainly has enabled many technologies aimed at resolving the interoperability issue when reformulated in XML.

Independent of the various semi-closed frameworks and platforms proposed by academia, the following selection of XML technologies represent a broad survey of the “state-of-the-art⁷” in XML Interoperability.

In general, each of the following technology addresses, in a direct or indirect manner, some subset of the aforementioned schematic and semantic conflicts in Table 2.1. In addition, technologies such as Architectural Form and XSLT also allow one to directly transform XML documents from one schema to another, a necessity for performing interoperability. Unsurprisingly, as the reader will shortly discover, none has a complete and coherent strategy that assaults the XML Interoperability barrier.

⁷The ink on many of the technologies’ specifications is not even dry yet, if they had even reached that stage of maturity. However, these technologies represent revisions and augmentations of old, tried and true techniques at enabling interoperability with new, XML-infused twists, so their effects remain heavily anticipated.

2.3.1 eXtensible Markup Language

As mentioned earlier, eXtensible Markup Language (XML) is a flexible meta-language useful in the creation of new data-describing languages. Several key definitions [7] need to be stated before proceeding:

Language Character Set	The set of defined values (typically Unicode [34]) that map to certain alphanumeric and other non-printable symbols in the given language
Character	An atomic unit of Text in the Language Character Set. Valid XML Characters include Unicode ⁸
Text	A sequence of Characters which may represent Markup or Character Data
XML Document	A well-formed arrangement of Text. One can view an XML Document either physically (as entities) or logically (as elements)
Entity	The basic <i>physical</i> storage unit of data in an XML Document
Parsed Entity	Entity whose Content consists of Text considered a part of the XML Document
Unparsed Entity	Entity whose Content may or may not be Text
Markup	One of start-tag, end-tag, and empty-element tags ⁹
Character Data	All Text that is not Markup
Content	The Text between the start and end tags
Start/End Tag	Markup that denotes the start and end of an Element
Element	The basic <i>logical</i> unit of data in an XML Document
Well-Formed	Entities, Elements, and Tags that are properly nested
Well-Formed XML Document	An XML Document that has exactly one root element and all start-tag consists of well-formed tags
Valid XML Document	A Well-Formed XML Document which also conforms to a given schema
Schema	Description of the logical structure and storage of a document. This description can be defined in one of two ways: through a Document Type Declaration (DTD) or XML Schema [33]
Document Type Definition	A document that contains Markup declarations that provide grammar for a class of documents

Table 2.3: Useful XML Definitions [7]

Note that these definitions are not exact copies of the official XML definitions but rather slightly abridged forms that remove unnecessary details that otherwise clutter the discussion of this thesis.

Common usage convention has “XML Document” meaning “Well-formed XML Document” and will

⁸Actually, Valid XML Characters consist of the ISO/IEC 10646 [22] character set, legal graphic Unicode characters, line feed, carriage return, and tab.

⁹Actually, Markup consists of start-tag, end-tag, empty-element tags, entity references, character references, comments, CDATA section delimiters, document type declarations, or processing instructions.

be the meaning the author adopts unless specified otherwise. Also, one must note that validity implies well-formedness, but not the other way around.

Now, XML is a streamlined subset of Standard Generalized Markup Language¹⁰ (SGML), a widely used international text processing standard. In particular, XML aims to “enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML” [7]. Combined with its abilities as a meta-language, it is clear that XML documents, in particular *valid* XML documents, play a central part in interoperability. Valid XML documents not only declare the language and encoding of its data implicitly or explicitly but also the logical structure of the data it contains. These advantages allow XML to clear several traditional interoperability hurdles such as deciphering encoding and structural description, leaving the discovery of the actual mapping between data-domains and its associated processing and storage requirements the major issues.

2.3.2 Architectural Form

Originally a concept conceived for SGML, Architectural Form (AF) [19] represents a pragmatic approach to interoperability. Namely, in addition to defining the syntax for associating elements between different DTDs (the “data-domains”), it also defines the semantics for actually extracting and rearranging data between the domains. Thus, it attacks the interoperability problem on two fronts — the mapping of identifiable structures between data-domains and the actual mechanism for extracting and rearranging data of the identified structures. It accomplishes this feat through clever redefinitions of SGML attributes and substitution/processing rules of entities within the SGML document.

AF’s abilities appear straight-forward once one considers its original intent — to allow a rough inheritance of SGML DTDs. It does this by enabling an XML document’s Client-DTD¹¹ to derive

¹⁰ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML). More information can be obtained from <http://www.iso.ch/cate/d16387.html>

¹¹Client-DTD refers to the one DTD than an XML document must conform to.

elements, attributes, and notations from at least one Meta-DTD¹². Thus, AF allows freedom in mixing and matching of elements in one DTD to elements in its Meta-DTD(s), and likewise with attributes¹³. It should be obvious that AF favors a tight-coupled strategy to achieving interoperability through the “derivation” of Client-DTDs from Meta-DTDs.

Thus, logically, AF views an SGML document as a parsed tree with the following properties:

- Nodes can be removed (pruned) at will (through suppression of processing).
- Nodes can be swapped (grafted) in a well-nested manner (through inclusion in processing).
- Leaves can be extracted and its values used (i.e. nodes can be grafted and the values in the leaves can be also switched around).

The AF concept has recently migrated to XML. As a concept, the pruning, grafting, and extracting actions described by AF is similar to the effects produced from amalgamating parts of several XML technologies, including eXtensible Stylesheet Transformation (see Section 2.3.7 for XSLT [10]) and XML Schema (see Section 2.3.6 for XML Schema [33]). Basically, AF’s actions are analogous to combining the effects of several XSLTs, and the association syntax is similar to XML Schema’s *equivalence* class attribute. Since AF actually pre-dates both technologies, it comes as no surprise that both supersede it in functionality.

2.3.3 Topic Map

Another pre-existing concept inherited from SGML, Topic Map holds promise in aiding interoperability because it defines a syntax for describing (hence associating) the implicit structure modeled by data in disparate data-domains. The original concept was motivated by the need to be able to

¹²Meta-DTD is purely a logical label refers to the additional DTDs that an XML document can derive elements from. A Client-DTD *could* be its own Meta-DTD!

¹³However, Architectural Form does *not* allow a match between an element and attribute, which seriously hampers it from dealing with labelling data conflicts.

merge indexes to offer navigational aid to the reader (i.e make searching for information easier for the researcher). The key insight to merging indexes, as Steve Newcomb, one of the original prime movers, explains:

... was that indexes, if they have any self-consistency at all, conform to models of the structure of the knowledge available in the materials that they index. But the models are implicit, and they are nowhere to be found! If such models could be captured formally, then they could guide and greatly facilitate the process of merging modeled indexes together [3].

The “models of the structure of the knowledge” is what Topic Map attempts to capture and provide easy access to. Not coincidentally, the problems that plague merging indexes also affect other forms of navigational aid such as tables of contents, glossaries, thesauri, cross references, etc, since they all attempt to provide access to information based on a model of the knowledge contained within it. At the heart of the model lies the concept of the *topic*; hence *Topic Map* aims to address this issue [3].

Topic Map views the world as a set of subjects, of which topics are specific, actual occurrences. Basically, topics reify¹⁴ subjects. Associations specify relationships between topics, while the topics play specified association roles. Topics and associations can also have “types”, which represent a sort of class-instance relationship between the topics.

It is clear that the meaning of the word “topic” depends highly on the nature of the information and the uses demanded of the Topic Map. For example:

Encyclopedia: One could have Spain as the topic and subject (recall that topics are the *actual* occurrences of the more ephemeral notion of subjects) and a topic type of country.

¹⁴Reify (Webster’s New World Dictionary, 1994): to treat (an abstraction) as substantially existing, or as a concrete material object.

Computer Science: One could have meta-languages as the subject, XML as the topic, and implementation as the topic type.

Additionally, topics may be linked to one or more information resources that are relevant to the topic in some manner; those resources are considered occurrences of the topic. A corresponding notion of occurrence role, which describes the type of relevance, also exist. Basically, an occurrence represents a very tight and well defined notion of relevance between a topic and some associated resource. For example, a graphical map of Spain is an occurrence of the topic of Spain, and the occurrence role would be “map”.

Finally, associations represent the heart of Topic Map — the ability to express, describe, and categorize associations between topics. Association type is the analog of topic type, so associations also have a class-instance relationship, while association role is the analog of occurrence role, which is the description of the topic association. For example, for the statement “Shakespeare wrote the play *Hamlet*”, Shakespeare and *Hamlet* would be the topics, wrote would be the association, written_by could be an association type, and playwright and play would be the association roles for the topics Shakespeare and *Hamlet*, respectively.

Thus, one can view Topic Map as a means of categorizing elements in data-domains and describing the relationship between arbitrary elements in arbitrary data-domains, labelling roles that topics play in the associations. This flexibility aids interoperability because it deals with the naming, labelling, and generalization data conflicts — not only does Topic Map define the relations between comparable identifiable data structures (erasing naming and labelling conflicts), but it also describes the relationship between them (eliminating generalization conflicts). That means that an automated knowledge tool can leverage the uniform descriptions to extract and rearrange data appropriately between data-domains. However, Topic Map does not include the actual means of describing how to rearrange the data.

2.3.4 XML Linking Language

A relatively new specification that tackles the problem of drawing associations between arbitrary resources on the Web, XML Linking Language (XLink) can be viewed as “Topic Map-lite” — that is, XLink [16] is a restricted instance of Topic Map. XLink fixes the types of its topic and association analogs to be *resource* and *arc*, respectively, and this instantly restricts one’s ease with expressing class-instance relationships. However, this restriction does not make it impossible, only more cumbersome, so XLink remains capable to deal with generalization conflicts. XLink also removes from Topic Map the notion of occurrences and its roles and focuses instead on the topic and the association role that it plays. It adds the idea of a linkbase, which is an external XML document that contains the actual XLinks which point to other resources. The linkbase can be viewed as a small “index of arbitrary links” external to the resources so no existing documents need to be modified to be a resource capable of being XLink’d.

An obvious observation one can make about XLink is that it trades off the generalized concepts of occurrences and types from Topic Map for immediate practicality — its specifications hint heavily at a browser being its target application: arcs can only be actuated “onLoad” or “onRequest” and shown either as new, replace, or embed — notions that clearly favor browsing/viewing applications.

Thus, the beauty of XLink lies in its simplicity. It does one thing and does it well — expressing links between resources — without all the psychological baggage a full-blown solution like Topic Map entails. Thus, it makes drawing associations between data-domains trivial. However, it sacrifices meaningful role descriptions for the resources, which means that automated correlation of elements from differing XML documents becomes harder.

2.3.5 Resource Description Framework

The primary goal behind RDF is to create a framework for metadata¹⁵. Simply put, RDF is:

... a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF uses XML to exchange descriptions of Web resources but the resources being described can be of any type, including XML and non-XML resources [9].

The foundation of RDF is a model for representing named properties and property values. The basic data model consists of three object types:

- **Resources:** All things described by RDF expressions are *resources*. A resource can be a URI¹⁶, a part of a URI, a collection of URIs, or even an object that is not directly accessible via the Web.
- **Properties:** A specific aspect, characteristic, attribute, or relation used to describe a resource.
- **Statements:** A specific resource together with a named property plus the value of that property for that resource is an RDF statement.

The three individual parts of a statement defined above are called, respectively, the subject, the predicate, and the object. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML [25].

Consider the following simple example:

Ora Lassila is the creator of the resource <http://www.w3.org/Home/Lassila>.

¹⁵Metadata is data which describes other data.

¹⁶Uniform Resource Identifier[2] (URI) — a compact string of characters for identifying an abstract or physical resource.

This sentence has the following parts:

Subject (Resource): `http://www.w3.org/Home/Lassila`

Predicate (Property): Creator

Object (literal): "Ora Lassila"

Thus, one can view any RDF statement (and essentially a relation) pictorially using directed labeled graphs, as shown below:

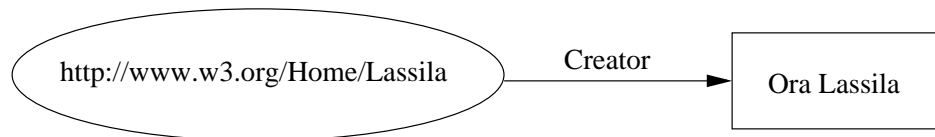


Figure 2-1: Sample RDF Statement [25]

However, RDF does not define any platforms nor provide any pragmatics tools to perform the processing of its statements. It does not even provide any standard semantic models for processes — it simply provides the most basic elements to enable those models and tools to be constructed.

An interesting aside: Topic Map and by association XLink can be viewed as predefined notions and models of particular resource descriptions layered on top of the more generic RDF; that is, one can “describe” XLink or Topic Map using the RDF even though it makes those languages more verbose and cumbersome. Thus, it is possible to leverage generic RDF automation tools on XLink or Topic Map when they do appear in the future. The RDF allows the descriptions themselves to be uniformly understandable so that the descriptions can then model relationships among web resources. In other words, the resources agree to use uniform description semantics offered by RDF but retain the ability to create proprietary tags for richer, more resource-specific data description.

2.3.6 XML Schema

XML Schema is a schema description language for XML written in XML [17] [33] [4]. What XML Schema brings to XML Interoperability is its ability to assign/create the legal XML Namespaces for elements, assign super/sub-type relationships between elements, and identify elements in the same equivalence class. This gives it the ability to at least identify and eliminate generalization conflicts while aiding in the identification of data type conflicts. While not inherently obvious, XML Schema also contains the necessary information that a mediator would need to resolve labelling, naming, and some aggregation data conflicts. This is due to the fact that schema gives structure to data; thus, anything conforming to a given XML Schema automatically associates elements between namespaces and schemas by default in an expository manner. It is up to a mediator to glean the necessary information from it.

Unfortunately, XML Schema neither specifies nor provides the means to transform data between data-domains, limiting its immediate usefulness toward XML Interoperability.

2.3.7 XML Stylesheet Language Transforms

XSLT is a language for transforming XML documents that conform to any well-defined schema to another XML document (see Table 2.3). It views an XML document in a similar manner as Architectural Form (see Section 2.3.2) — as a parsed tree, with the content at the leaves and the tags at the nodes. Furthermore, it defines semantics for manipulating nodes, so the end result is a “transform language” that is able to rearrange, insert, and delete nodes arbitrarily within an XML document. It also provides simple programming facilities to express logic that can be applied recursively during the transform to add/extract/remove data from the leaves of the nodes.

However, XSLT is *not* a language to *describe* relations — it can simply perform the actions a relation may describe. Thus, one can only embed relations within the logic of the XSLT stylesheet — the

stylesheet becomes a specific sort of translator.

2.4 Summary

Hopefully, by now the reader sees the advantages of having XML as a constraint on the traditional interoperability problem. If not, review this chapter! Now, without getting too far ahead, the author wishes to claim without proof that XML addresses the following issues in interoperability; the reader can judge its effective by the end of this thesis..

Data Type Conflict: Largely resolved with XML.

Labelling Conflict: To be addressed by thesis proposal with aid of XML.

Aggregation Conflict: To be addressed by thesis proposal with aid of XML.

Generalization Conflict: To be addressed by thesis proposal with aid of XML.

Naming Conflict: To be addressed by thesis proposal.

Scaling Conflict: To be addressed by thesis proposal.

Confounding Conflict: Outside of thesis scope.

Domain Conflict: Not addressed by thesis proposal.

Integrity Conflict: Not addressed by thesis proposal.

Thus, current progress on XML Interoperability as discussed earlier can be summarized as having focused on the following two aspects:

1. The language to describe the associations between structural elements of each data-domain (XML Schema, Architectural Form, Topic Map, XLink, RDF, Protégé).

2. The mechanics to rearrange and modify the associated data elements physically (XSLT, Architectural Form).

The astute reader will realize that a third aspect of XML Interoperability has barely been scratched, which is:

The process of discovering the above associations between the structural elements of each data-domain (COIN and somewhat Protégé).

Both Protégé and COIN have this problem within their sights; all other efforts skirt around this critical *know how* issue. Thus, this thesis will attempt to tackle this problem along with leveraging existing research and technology for the other aspects.

Chapter 3

Project Design and Requirements

From the preceding discussions, a consistent theme in XML Interoperability arises: how something in a data-domain is described is no longer an issue and is not as important as what is described. Clearly, re-interpreting the interoperability problem into XML solves several technical hurdles (such as parsing and character encoding recognition). XML frees one to focus attention on the real problem — the semantics of interoperability. Once one gains a hold on identifying the individual “something” in a particular data structure, what is it? What does it mean? What does it relate to and how?

As fortune may have it, these sort of questions underlie a class of problems that The MITRE Corporation is currently attempting to solve for its clients and is an additional motivating factor behind the research of this thesis.

3.1 Company Background

The MITRE Corporation is a not-for-profit corporation that was formed to deal with critical national issues such as defense, commercial aviation, and taxation. The majority of work done at MITRE is concentrated in three Federally Funded Research Development Centers (FFRDC), which service

the Department of Defense (DOD), the Federal Aviation Administration, and recently, the Internal Revenue Service and the Treasury Department.

Unlike most corporations, FFRDCs are *prohibited* from manufacturing products or otherwise openly competing with the industry. Instead, they work together with commercial corporations from planning and concept design to technology insertion and integration. As agents for the government, FFRDCs specialize in scientific research and analysis, systems development, and systems acquisition.

The FFRDC of MITRE working with the DOD is currently focusing on developing more thorough cooperation among command and control systems, with the ultimate goal of a single, integrated command and control system that can support joint and coalition operations worldwide. The research within this thesis is directly in line with this goal, as it examines the fundamental knowledge and technology underlying and enabling *fuselets*.

In order to understand the importance of fuselets, one must understand the context and environment in which they operate. This requires a short diversion to explain the overall goals of the Joint Battle InfoSphere (JBI), since fuselets are a component of the JBI [36].

3.1.1 Joint Battle InfoSphere

The goal of the JBI is to enable our armed forces to prevail and win in combat. To accomplish this task, the JBI plans to provide information superiority to our warriors, which enables them to “interpret information and make decisions faster than the other guy . . .” [36].

This advantage is achieved by adhering to a publish and subscribe mechanism, which basically facilitates the sharing of information between interested parties, manipulation of information to create knowledge, and dissemination of that new knowledge to other interested parties upon creation. Basically, the interested parties send information via messages asynchronously back and forth. To

do this, the JBI provides three major functions:

1. Information Input
2. Information Manipulation to create Knowledge
3. Knowledge Output and Interaction

These functions correspond roughly to any classical system design except information is now the input and knowledge the output.

3.1.2 Fuselets

The role that fuselets play in the JBI amounts to enhancing and automating all three functionalities. That is, fuselets “fuse” information from several subscribed sources into knowledge. To do this, a fuselet subscribes to its interested information sources in the JBI, and upon the publishing of information matching its subscription, it is triggered to perform its task. This usually leads to the generation of new, synthesized information or knowledge from the fuselet’s information subscription sources.

Thus, fuselets have many uses: they can bring information into the JBI, transform sets of JBI objects into aggregated objects, or gather objects for presentation and automatic report generation [36].

3.2 Project Description

The particular MITRE project that this thesis supports, “Fuselet Architecture for Adaptive Information Structures”, revolves around the same themes as XML Interoperability. The goal of the project is to enable adaptive data interoperability between two participating systems through a transparent mediator. The mediator, in this case, also happens to be a particular type of fuselet.

Now, in order to accomplish this task, several pieces of information are needed, including:

1. The actual mappings between the appropriate data structure(s) (i.e. what goes to what?).
2. A semi-automated way of deriving these mappings (i.e. a human can probably do this better but will never be fast enough nor have enough stamina).
3. Preferably, a means of representing and storing these mappings such that the knowledge can be easily reused by other applications¹.
4. An automated and intelligent way of using the knowledge of mappings to facilitate the mediation of data between systems.
5. Preferably, a means of representing the transformations done by the mediation so that the knowledge can be easily reused.

Clearly, tasks #1, #3, and #4 address the semantics of interoperability — what maps to what, how to represent that knowledge, and how to use that information for the interoperability task of mediation. Thus, the remainder of this thesis can conveniently concentrate on the project's requirements, design, and implementation since it encompasses the goals of XML Interoperability.

In fact, as the author will shortly discuss in Sections 3.3 and 3.4, the project has three deliverables that encompass the tasks listed above. They are:

X-Map: An application that performs tasks #1, #2, and #3 — X-Map will attempt to semi-automatically discover the associations and store that result.

Association Language: An XML-based language that describes and facilitates the expression of X-Map's associations (a.k.a. an X-Map expression language) and enables other applications, such as a mediator, to do its job. It is a separate deliverable from X-Map but is considered a part of X-Map design-wise since X-Map generates it while performing task #2.

¹This is a key benefit sought by this project — associate once, reuse anytime and anywhere.

Mediator: An application that performs task #5 to accomplish task #4 — the Mediator will attempt to perform mediation tasks such as transforming/rearranging data to achieve interoperability between its client systems.

3.3 Design Goals

Several overall design goals encompass this thesis investigation, and this list must never be considered absolutely complete. It represents what the author perceives as the overall design goals that should not be violated for any acceptable solution implementation for this project.

The following general design principles, adapted from those of XLink, underly X-Map’s and Mediator’s design and are listed in no particular order. The XLink design principles are described in detail in the W3C² Note “XML Linking Language (XLink) Design Principles” and will not be repeated unnecessarily here [15].

However, one must recall the following meanings of word modifiers as specified by the IETF RFC 2119 “Key words for use in RFCs to Indicate Requirement Levels” [5] since it is the commonly accepted way to denote the critical-ness of any given requirement item in a requirements or design goals document such as Chapter 3 of this thesis.

- Must, must not, required, shall, shall not — absolutely must be true/false.
- Should, should not, recommended, not recommended — maybe true/false, depending on the situation.
- May, optional — truly optional.

²World Wide Web Consortium (W3C) — more information can be obtained from <http://www.w3c.org>

3.3.1 X-Map Design Goals Wish List

1. X-Map must be straightforwardly usable over the Internet.
2. X-Map must be usable by a wide variety of relational expression domains and classes of relational linking application software.
3. The X-Map design should favor reuse of existing/emerging XML standards and interface implementations.
4. The X-Map expression language must be XML.
5. The X-Map design must be formal, concise, and illustrative.
6. The X-Map expression language must be human-readable and human-writable.
7. The X-Map expression language may reside within and without the documents in which the participating resources reside.
8. The X-Map expression language should support a mechanism to transform data between data-domains.
9. The X-Map expression language must represent the abstract structure and significance of semantic relations.
10. X-Map must be feasible to implement.
11. Terseness in the X-Map expression language is of minimal importance (i.e. expression richness is paramount).

3.3.2 X-Map Design Goals Rationale

Several design goals mimic those made by XML and XLink, including #1, #4, #6, #10, and #11 — it is certainly advantageous and desirable for X-Map to be Internet-ready, expressed in XML, be

human readable and writable, feasible to implement, and enable rich expression³.

Additional design goals that are blatantly obvious for XML Interoperability with X-Map include #2, #7, #8, and #9 — they embody the goals of representing semantic relations in a permanent fashion (#7 and #9) and allowing the knowledge to be used by other tools (#2). Although design goal #7 notes that X-Map may reside within or outside the participating resources involved in the relation, outside will be the typical case to minimize intrusion and increase feasibility. Clearly, given the context of fuselets (see Section 3.1.2) for the project, goals #2 and #8 are desirable.

Last but not least, X-Map should favor reuse of existing/emerging XML standards and interfaces (#3) over creating new ones. This goal emphasizes the project's desire to ease on-going maintenance and leverage the wide-variety of XML-related standards work already done. In addition, the process of standardizing may reveal flaws and/or new points of view in thinking about X-Map, which fits with its purpose.

3.3.3 X-Map Design Implications

Now, the general case of automatic association is very hard and implies the solving of consistent semantic matching of elements between two data-domains. This problem is clearly beyond the scope of this thesis investigation and will therefore not be pursued. The author believes something along the lines of the Semantic Web⁴ as proposed by Tim Berners-Lee will have to exist for X-Map to have enough support to tackle the general automatic association case and represents the topic of future research.

However, this does not preclude the use of semi-automated methods to make consistent semantic matches in suitably constrained cases, which is what the author favors. In this scenario, the system

³Rich expression of transformations on semantic data in XML is a tangible, eventual goal.

⁴The Semantic Web represents the eventual evolution of the “Web” as envisioned by Tim Berners-Lee — it is a consistent logical web of data that will enable machines to participate in communicating with and helping humans to communicate with each other. This is based on the view that the present-day Web focuses on human consumption but not machines.

will try its best to come up with a semantic match, and when it cannot or has serious doubts, it can ask a human operator for assistance. Obviously, the system must be intelligent enough to not let the operator do all the work yet not masochistic enough to tackle the yet-unsolved general problem.

This realization will serve as an addition guide for the requirements of X-Map.

3.3.4 Mediator Design Goals Wish List

The astute reader should realize that X-Map’s design directly impacts that of the Mediator’s — that is, X-Map can output nearly all necessary information for the Mediator to simply “plug and chug”. Thus, the Mediator’s design goals implicitly require X-Map’s design goals in Section 3.3.1 to be satisfied, and without belaboring the point, here are the additional design goals relevant for the Mediator.

1. The X-Map expression language must describe how to manipulate data (i.e. algebraic and string manipulations).
2. The Mediator must perform the mechanisms to transform data between data-domains.
3. The Mediator must be feasible to implement.
4. The Mediator must be transparent (i.e. non-intrusive) to its clients.

3.3.5 Mediator Design Goals Rationale

As noted in Section 3.3.4, the Mediator depends heavily on X-Map since the ease of its job directly depends on the information that X-Map gives it. Taking X-Map’s design goals in stride, the Mediator’s design goals follow logically — since X-Map *must support* a mechanism to transform data between data-domains, the Mediator *must supply* the mechanisms to manipulate and transform data between data-domains (goals #1 and #2). Furthermore, the type of interoperability desired

for this investigation warrants goal #4 (see Section 1.1.3), and goal #3 satisfies purely academic and completeness desires.

3.3.6 Mediator Design Implications

The Mediator does not implicate any significant code responsibilities since its job is to act as a transparent filter between its interested parties and do the actual transforms specified by the applicable relation(s) that X-Map discovered. These tasks are mostly accomplishable via standard software engineering practices (networking, communications, filtering).

The lone piece that requires design care is how the XML data transformation actually happens. Since the incoming and outgoing data are in XML and the transforms are specified in XML, the engine that the Mediator employs to perform the transformation must natively work with XML.

3.4 Requirements

Adhering to both the previously enumerated design goals of X-Map in Section 3.3.1 and the goals of XML Interoperability given in Section 3.2, a plausible set of requirements readily emerges. Basically, the set of requirements fall into the three categories of tasks that XML Interoperability demand:

1. Discovering the specific relation between elements of two data-domains.
2. Expressing the discovered relations in an useful fashion.
3. Applying/using the expressed knowledge towards an interoperability task, such as mediation, which includes data transformations.

Actually, *two* languages are implicitly used in the above three tasks — a language to describe the relations between objects and a language to describe transformations. Ideally, one wants to have

one XML language to accomplish both tasks for X-Map, but no such unified technology exists at the current time. Instead, technologies such as XLink, XSL, and MathML⁵ represent XML languages that perform varying combinations that are proper subsets of the desired language. Thus, a practical way of resolving the design goals could be to specify separate requirements for each functional language; this is the route that the author has chosen.

Thus, the following list identifies the requirements that follow from the aforementioned XML Interoperability tasks and design goals. Appropriate descriptions will be given in the following sections.

1. Semi-Automated System to Discover Associations (fulfills X-Map Design Goal)
2. Association Language (fulfills X-Map Design Goal)
3. Transformation Language (fulfills Mediator Design Goal)
4. Mediation Engine (fulfills Mediator Design Goal)

3.4.1 Semi-Automated System to Discover Associations

The system through which the associations are discovered forms the technical backbone of X-Map. It is the mechanism through which associations are discovered and captured by the Association Language in Section 3.4.2. The following set of requirements outline the capabilities such a system should possess to address XML Interoperability.

- It must be straightforwardly usable over the Internet.
- It must leverage XML natively as inputs and outputs.
- It must allow human intervention at defined critical junctions.
- It must be expandable (i.e. allow other add-in modules to extend analysis options/capability).

⁵MathML[21] is a markup language for describing and laying out mathematical equations.

- It must discover relations (see Section 4.2.1) between multiple resources.
- It should favor reuse of tools and implementations of existing/emerging XML specifications.
- It should perform its tasks efficiently.

3.4.2 Association Language

The Association Language lies at the heart of how rich an expression one can make in designating an association between two or more resources. It represents half of the aforementioned “X-Map expression language”. The following set of requirements outline the capabilities of such a language that is useful for XML Interoperability.

- It must be able to reference any Internet-accessible resource, including the internal contents of the resource⁶.
- It must have a means of referencing resources from the data-domains.
- It must be able to express a linkage of some sort from one object to one or more objects.
- It should be able to express a notion of link traversal; that is, what, if anything, happens by activating the link.
- It should allow classes or sets of resources to be related conveniently.
- It may provide semantic hints to tools (such as invertibility).
- It must be in XML and can be within or outside the resource participating in the linkage.
- It must be human-readable and human-writable.

⁶Using XPath [11], one can definitely do this for an XML document.

3.4.3 Transformation Language

The Transformation Language and engine form the guts of “what happens” when an association is traversed or “used” — that is, when traversing, the language describes how and what happens to data in one domain that go to its associated data form in the other domain and the engine performs the actual instructions. It is the other half of the aforementioned “X-Map expression language”. The following set of requirements outline the capabilities necessary of such a language to enable XML Interoperability.

- The Transformation Language must be able to reference identifiable components in XML data.
- The Transformation Language must be able to express the computation of new output component values from given input components.
- The Transformation Language should be expressed in XML.
- The Transformation Language should be able to express arbitrary transformations to data.
- The Transformation Language should allow rich expression of transformation ideas.
- The Transformation Language should be able to build expressions that transform data (also recursively).

3.4.4 Mediation Engine

The mediation engine does the “plug and chug” of the instructions dictated by the Transformation Language. It actually *performs* the XML Interoperability task, and the following set of requirements outline the capabilities necessary to execute XML Interoperability.

- The Mediation Engine must be able to access, query, and otherwise extract the necessary data for its computations.

- The Mediation Engine must be able to apply arbitrary transformations to data.
- The Mediation Engine must leverage XML natively as inputs and outputs.
- The Mediation Engine should allow rich expression of transformation ideas.
- The Mediation Engine must be able to build and execute expressions recursively.

3.4.5 Summary of Background

A landslide of design goals and requirements was unleashed in this chapter.

Chapter 4

Project Design and Implementation

The requirements stated in Section 3.4 lend some hints to X-Map's implementation design. Namely, there is a clear, logical, and natural separation of interoperability tasks such as discovering associations and using the associations to perform mediation. Thus, Figure 4-1 represents a possible overall picture of the implementation design, with a general description following it.

The remainder of this chapter will define:

- **X-Map's** design, rationale, and strategy; in particular, what types of relations it discovers and how its regenerative association engine works to uncover them.
- **Association Language's** design and rationale.
- **Transformation Language's** design and rationale.
- **Mediation Engine's** design and rationale.

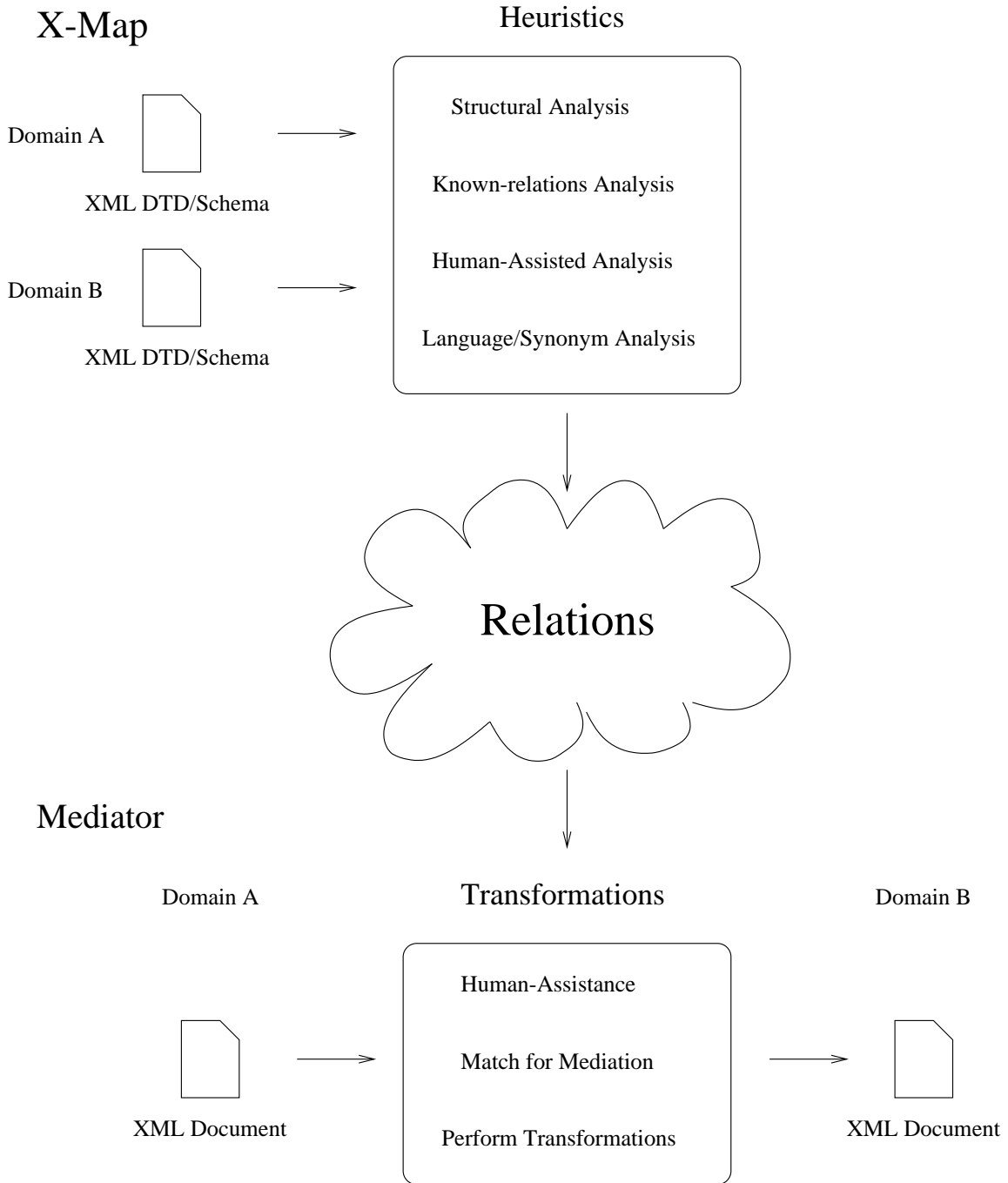


Figure 4-1: Design Overview

4.1 Design Overview

Basically, X-Map's job is to discover new semantic relations semi-automatically given schemas that correspond to data in the relevant data-domains. Then, it should deposit the knowledge in a reusable format (i.e. in the Association Language) that captures the semantic notions of association and the appropriate transformation function description (i.e. the Transformation Language), if any, for other applications such as the Mediator or fuselets to use. Finally, an illustrative prototype such as the Mediator will show how to use the knowledge in the associations to help it mediate (such as translating and transforming) data between data-domains to achieve XML Interoperability.

It is desirable to uncouple X-Map and the Mediator because the tasks they perform are fundamentally different — the former creates associations while the latter consumes them. That means that it is also a good design choice to abstract the Association and Transformation Languages away from both X-Map and the Mediator since then they share the languages as an interface, which minimizes dependencies between the two modules. Furthermore, the uncoupling allows the Mediator to invoke X-Map on-demand to process XML schemas as it runs across new ones, and it also allows X-Map to run silently in the background and use idle processor moments to add to its known relations linkbase via the regenerative association engine.

4.2 X-Map Design and Rationale

The following figure is a close-up of Figure 4-1 that focuses on X-Map and depicts the conceptual tasks¹ that X-Map makes.

¹XML-2-Bags and Bags-2-XML are treated more fully in Section 4.2.3.

X-Map

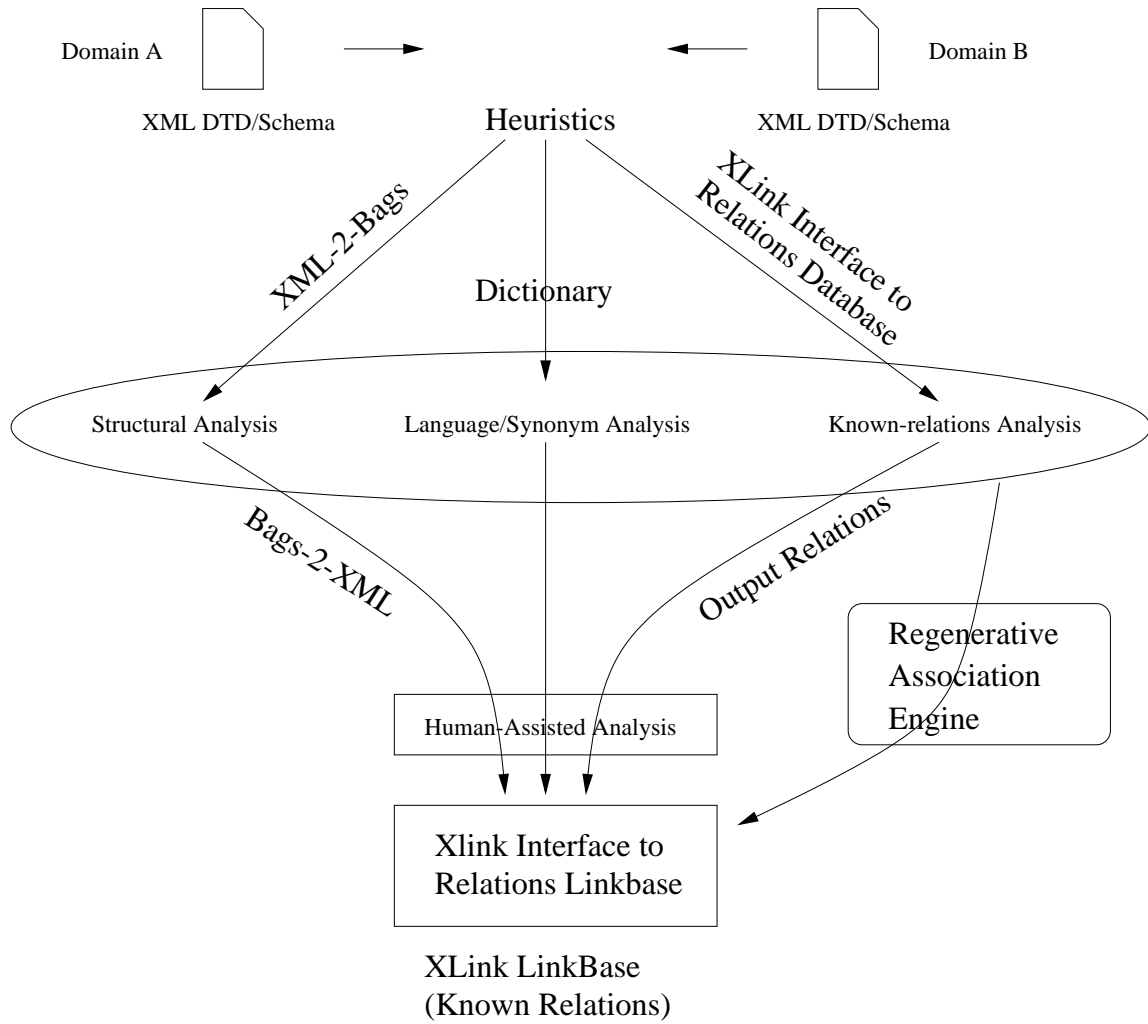


Figure 4-2: X-Map Overview

Operationally speaking, X-Map performs the following tasks, in order:

1. Locate and read in the desired schemas to associate (either specified by the XML documents or given as-is) via the prescribed programmatic XML interface².
2. Using a heuristic, determine which analysis to use (including all) to generate the associations.
3. Generate the associations and improve uncertain associations.
4. [Optional] Human operator intervention can be invoked by any analysis at anytime.

²Currently, this interface is either DOM2 or SAX2.

5. Deposit the associations into a linkbase through the XLink interface.

Now, a cursory correlation of the operational tasks that X-Map performs with the conceptual tasks displayed in Figure 4-2 reveals the following design/implementation questions that must be answered. The motivation here is that what X-Map *does* conceptually and how those concepts break down operationally is clear, but the design/implementation details of the operational tasks has not been tackled. The questions are:

1. What types of relations are valuable and/or feasible for X-Map? This classification directly affects the sorts of correlations that X-Map can make for operational task #3.
2. What are the algorithms and processes necessary to perform the above classification, and how should they connect together to discover the relations? The answer to this affects operational tasks #2, #3, and #4.
3. How does X-Map acquire and preserve the knowledge of the relations it just gleaned through its algorithms? This impacts operational tasks #1 and #5.

Furthermore, the author considers the following three design goals influential on the X-Map design, and they will guide the implementation of the operational tasks.

Modularity: The ability to extend X-Map's analytical capacity in discovering new relations and validating existing relations is critical for its forward evolution.

Use of XML natively: Using XML as natural inputs and outputs is important. Using XML as a mere persistence medium rather than a structured, hierarchical medium does *not* leverage the advantages of XML (see Section 2.3.1).

Reuse of XML specifications: The usage of existing/upcoming XML specifications and interfaces leverages not only the design expertise of the W3C but also the time and effort spent by the XML community in constructing, testing, and verifying the published public interfaces.

Thus, it behooves one to investigate in detail the questions that X-Map must answer in order to accomplish its operational tasks because they lie at the heart of XML Interoperability. In particular, X-Map’s design aims to resolve several data conflicts itemized in Section 2.4, including:

Schematic Conflicts: Data Type Conflict, Labelling Conflict, Aggregation Conflict, and Generalization Conflict

Semantic Conflicts: Naming Conflict and Scaling Conflict

As the reader may recall, these conflicts encompass a great majority of the interoperability problems introduced by design autonomy and data heterogeneity (see Section 2.1.1). Their impacts still exist for this project, but additional factors, including a wealth of relevant domain knowledge, help equalize the balance of power and enable X-Map to solve a suitably constrained case of the automatic association problem. The influence of these real but non-design factors on X-Map’s design will also be looked at simultaneously in the following sections, along with their accompanying rationale.

Basically, X-Map employs a host of language, synonym, and structural analysis algorithms to meet and resolve the XML Interoperability problems posed by the project’s requirements and goals. Armed with a regenerative association engine³ that constantly re-analyzes and refines associations when given new knowledge and aided by specific knowledge of the project’s problem domain, X-Map can resolve the XML Interoperability problems.

A key feature of X-Map is the regenerative association engine (see Section 4.2.4 for a full description), which not only validates known associations but also speculates on uncertain associations. The regenerative association engine does this by keeping track of past associations that were denoted as not certain (i.e. associations where some participating element’s role was not determined) and trying to re-ascertain them when given information from newly acquired schemas. Thus, X-Map never really “gives up” on an association until it is proven to not exist.

³Private Conversation: This insightful idea came from the experiences of Aditya P. Damle from his efforts at deriving and matching what Web documents mean with what people what. <http://www.2028.com/>

The following sections will examine the three key questions enumerated on page 55 and give X-Map's answers and corresponding design. Namely, what relation classes do X-Map care about and why, what are the algorithms and processes used by X-Map to discover those relations, and how does it store the discovered relations?

4.2.1 Enumeration of Relation Classes

The Definition of Equals

An important definition to nail down before discussing the merits of any type of relations is the meaning of the word *equals* — that is, under what circumstances do two objects become semantically and operationally indistinguishable from each other and therefore can be used interchangeably in each other's place?

This definition is important because the entire subject of interoperability revolves around this question. If one does not define, in some given context(s), what *equals* means, then interoperability is doomed from any mediator's perspective — there will be no common ground to even settle on to mediate between its client systems.

Furthermore, the structural analysis portion of X-Map's strategy depends on being able to identify and correlate parts of different schemas⁴ that are *equal* to each other. This is due to the schematic nature of XML documents and X-Map's strategy, which is to doubly-exploit structural and semantic information to achieve XML Interoperability (for details see Section 4.2.3). Without defining *equals*, X-Map cannot even exploit an XML document's structure since it'd have no common elements with which to reason. Hence, X-Map's effectiveness relies heavily on this definition.

For this project, *equals* has the following meaning:

⁴For the purposes of XML Interoperability, data structure is synonymous with its schema since valid XML documents (itself a data structure) must conform to some XML Schema. Hence, its structure must be known — just parse the schema. Thus, structural information-wise, the two are synonymous.

Two entities are equal when they express essentially the same physical object or concept and differ only by measurement units, mathematical precision, acronyms and/or abbreviations, or other shortening or lengthening of its labelling identifier by concatenation.

The Implications of Equals

For example, X-Map will consider an element that measures time in milliseconds as equal to an element that measures time in seconds since they differ by a scalar factor. Likewise, X-Map will consider SSN and Social Security Number to be equal since the former is an acronym of the latter.

However, in the case of precision, one must pay careful attention when a relation reduces precision because of the semantic problems due to inappropriate numeric truncation. For example, suppose a number has range $[0.0, 10.0]$ in domain 1 while its counterpart in domain 2 has range $[0, 10]$. While there is no obvious danger transforming from domain 1 to domain 2 (transforming a 6.6 to 7 is safe since domain 2 does not care about the precision), the reverse is not necessarily safe. Namely, a 7 in domain 2 could map to $[6.5, 7.5]$ in domain 1, and a naive transformation to 7.0 could introduce a whopping 50% average error.

The thorny issue of precision and truncation represents a large class of interoperability problems stemming from information-loss, which unfortunately is not included in the scope of this thesis and represents a future area of research. However, X-Map can help alleviate this problem by noting associations which involve loss of precision, flagging them programmatically, and saving this knowledge along with the association into persistent storage to serve as warnings to other applications.

The astute reader should realize that this definition of *equals* immediately addresses two of the semantic conflicts by making their resolution a part of X-Map: naming and scaling.

The naming conflict (i.e. SSN vs. S.S.N. vs. Social Security Number), by definition, is declared a trivial issue — X-Map only needs to detect the presence of this case and if so, determine if any

element in the set of known abbreviations and acronyms of one element intersect with the same of the other, and then declare the elements equal if the intersection is non-empty. This tactic applies due to the fact that the set formed by the “abbreviations and acronyms” relation for any given element is a connected component⁵ — every element in the set is related through some transitive set of equivalence relations⁶ to every other element in the set. Thus, if one element of a connected component is postulated to belong to two different components (i.e. the components of two elements in different data-domains), then the components must be the same; thus, the elements must be the same.

Likewise, the scaling conflict, such as English vs. Metric, is a vacuous issue — X-Map simply needs to detect if the elements involved are even related to measurement scales that it knows and cares about and if so, declare them equal and generate the appropriate mathematical transformation between the two measurement scales. The hard part here is the generation of the appropriate translation formula, but on-going work at MITRE exists to tackle this problem and can be sufficiently leveraged.

For this project, these SI⁷ measurement units hold interest:

- Time — from seconds to microseconds
- Mass — in kilograms or pounds
- Length — in kilometers or miles
- Electro-magnetic radiation — in Watts or Joules
- Location — in longitude/latitude/degree/minute/second/deci-second

Considering the restricted nature of abbreviations/acronyms and scaling and the well-defined solutions to their behaviors, one can easily use a solution similar to that proposed by COIN to attack

⁵A *connected* component refers to the concept in topology which says “if a component is connected[27], then a path (of relations) lead from every element in the component to every other element in the component”.

⁶This equivalence is implicit in the meaning of abbreviations and acronyms.

⁷International System of Units [Le Système International d’Unités] (SI). More information can be obtained from <http://physics.nist.gov/cuu/Units/index.html>

this problem — a lookup table of relevant values [18].

X-Map goes one step better with the well-known lookup table solution to this problem — its “lookup table” will actually be associations in the same linkbase of knowledge that it is building all this time.

This provides two obvious benefits by re-leveraging the regenerative association engine:

1. Every use of a known association validates it.
2. Every introduction of new information into the linkbase of knowledge is an opportunity to resolve an unknown association.

Relation Classes based on Equals

Finally, using the above definition of *equals* as the fundamental building block, the author can describe the other relevant classes of relations for the project. They are summarized in the following table: Table 4.1.

Relation Class	Description
Equals	When an element is equivalent (<i>equals</i>) to the other element
Substitutable ⁸	When an element can be used in place ⁹ of another element
IsLike	When uncertain of any association but <i>not</i> no association, yet
Generalization	When an element completely subsumes another element
Aggregation	When an element consists of more than one other element
Rearranging	When one element is a rearrangement of another element
Abstraction	When one element is a mathematical transformation of another element
LookupTable	When one element is a lookup translation of another element

Table 4.1: Fundamental X-Map Relation Classes

In line with the earlier assertion of how X-Map combats information-loss (see Section 4.2.1), X-Map recognizes an associated attribute list for each relation denoting whether the relation loses

⁸A two-way substitutable relation would imply equals; hence, substitutable can be viewed as an one-way equivalence.

⁹Substitutable is analogous to a subtyping relationship, where one element can be used in place of another but not necessarily the other way around.

information upon traversal and if so, what type of lossage. Precision represents one possible value; other hints can be added but remains unspecified for this project.

X-Map also recognizes a similar attribute list for information-gain. One possible value is invertibility, which refers to whether the relation from one domain to the other can be reused in the opposite direction. Invertible mathematical formulas (Abstraction) and LookupTable obviously fall under this category.

4.2.2 Discovering Relations

The previous section focused on identifying the relations that interest X-Map. This section will focus on how to identify those relations in XML documents, as well as some background preparation, which will then pave way for the discussion of algorithmic strategies in Section 4.2.3 that discovers these relations.

Like COIN’s axiomatic approach toward mediating queries [18], the following set defines X-Map’s operational “axioms”. This approach is feasible due to its emphasis on document structure, which directly leverages specific domain knowledge and will be discussed in Section 4.2.3.

Algorithm Execution Decision

Basically, X-Map performs an optimization step between loading the XML schema into memory and processing it with its algorithms — X-Map pre-processes the schema and tries to compile a processing strategy on the fly, which X-Map then executes.

X-Map presently employs a simple heuristic to determine which of its algorithms to run on input XML schema. In this case, X-Map uses a computationally cheap circularity check, which, as the author will shortly discuss in Section 4.2.3, is a good indicator of aggregation conflicts. Other cheap heuristics can be performed at this stage, each relevant to the algorithm it precedes, which will

ultimately drive down computational costs while increasing relevant matchings.

However, this is purely an exercise in optimization to finding the right set of algorithms to run, not interoperability, so X-Map may, for simplicity, run all of its heuristics such as the circularity check.

Structural Analysis Background

The major realization underlying X-Map’s structural analysis approach is the formulation of the data structure. This is important because it determines the types and number of analyses that can be run.

Due to the highly hierarchical and tree-like structure of XML schemas, the importance to recast this tree into a more computationally palatable form cannot be underestimated. The fundamental problem with structural analysis on a tree is the bias it builds against correlating aggregation and generalization conflicts. In non-trivial cases, these conflicts typically result in completely different tree structures which confound any attempts to frame any resolutions based on hierarchies. Thus, for XML Interoperability purposes, the schema’s “tree” must be altered into a better representation.

This realization leads X-Map to propose a non-traditional view of a tree — as a directed graph — for XML Interoperability, as shown by Figure 4-3.

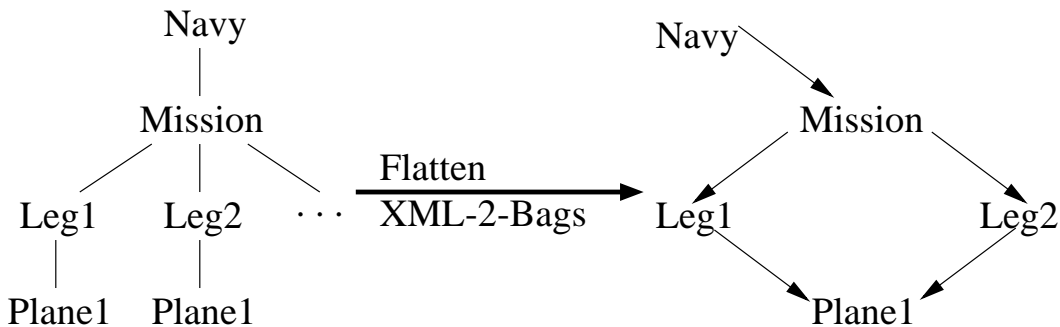


Figure 4-3: Collapsing Trees into a Directed Graph

Basically, tree nodes map to graph vertices, tree edges map to graph edges, and the edge direction

goes from the root node to the child node.

This view transformation dramatically increases the number of interesting properties (of graphs) and also the analysis options one can perform on a tree hierarchy, and as Section 4.2.3 will show, this transformation can be done quickly.

Thus, the following list represents some¹⁰ of the interesting features of a directed graph which X-Map will take advantage of, and they will be discussed in Section 4.2.3. Basically, each represents an embodiment of schematic or semantic conflicts mentioned earlier (see Section 2.1.1) and shows how X-Map uses the information.

- Associations in Hierarchies
- Associations in Cycles
- Associations through Contradiction

4.2.3 X-Map's Strategies

As briefly mentioned in Section 4.2, X-Map's strategy employs a number of analysis algorithms on structure and language, aided with specific knowledge of the project's problem domain, to meet and resolve the XML Interoperability problem.

A key feature of X-Map is its regenerative association engine (see Section 4.2.4), which not only validates known associations but also keeps track of uncertain associations so that they can be ascertained when new future information becomes available to X-Map.

Equally critical to X-Map's strategy is the idea of recasting a hierarchical data structure like XML into a directed graph and leveraging the body of graph theory work on it.

¹⁰By no means is this list all inclusive. Additional graph features to exploit can easily be the topic of future research.

Finally, at all times, human intervention may prove more fruitful in resolving the final outstanding issues after each sub-analysis has finished weeding through the complexity. This proves to be a benefit to both the operator and the algorithm since the algorithm is ultimately not sentient and may require suggestions only a human can make, so the algorithm will go through the complexity which overwhelms most humans to extract the basic conflict in question.

Thus, the following sections will explain the process and rationale behind the graph analyses that X-Map uses along with how the regenerative association engine “does its magic”.

How to Collapse a Tree Hierarchy

The process of converting XML into a directed graph and vice versa proves deceptively simple due to existing software for another project at MITRE. The basic concept behind the “Bags¹¹” concept is that a hierarchy is simply a collection of nodes whose edges represent some attribute or containment relationship between the appropriate parent and child node.

However, instead of focusing on how the relations stack together to form a tree-like hierarchy, if one focuses on the nodes and the relations they contain or participate, the graph formulation immediately leaps forth.

Thus, “XML-2-Bags” embodies this realization to fruition. “XML-2-Bags”, in pseudo-code, simply:

¹¹Thanks to Nazario I. for for his contributions of Bags, XML-2-Bags, and the inverse Bags-2-XML from iPAM [20].

XML-2-Bags:

```
//ALL XML documents have a root element
Take XML document in and find root element;
bag-ify(root_element);

subroutine bag-ify (parent_element) {
    make a bag for the parent_element;
    whisk the parent_element's attributes into the bag's attributes;

    for each child_element of parent_element {
        bag-ify(child_element);
        create relational link from parent_element to child_element;
    }
}
```

Figure 4-4: XML-2-Bags Pseudo-code

Also, one must realize that in addition to “flattening” the tree into a directed graph, X-Map will keep the hierarchical information around to aid its graph algorithms in determining relations.

Finally, as explained earlier in Section 4.2.2, viewing a tree as a directed graph holds much potential, as the following sections will show.

Discovering Associations in Hierarchies

Discovering associations between elements in a hierarchy can be tricky because the hierarchy is not guaranteed to contain any “meaning” for the associated elements. Fortunately, in this project, it is often the case that hierarchy embeds information about its constituent elements. This is domain knowledge specific to this project that will be exploited.

The typical example (with A, B, C, D, E as elements and arrows pointing in the to part of the relation) would be: A contains B; B contains C; and D contains E. If C and E are related, does that say anything about B and D? What about A and D?

Schema 1 Schema 2

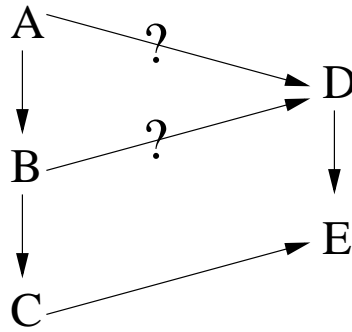


Figure 4-5: Hierarchical Association Discovery

For this project, it often turns out that if C and E are related, then either A or B are related to D. However, even if the relation cannot be immediately drawn from A or B to D, one can still speculate on its existence. Future processes may discover that B and D are related without the presence of C and E, in which case X-Map now has a more complete picture than either processes alone.

Furthermore, if a correlation is drawn both between C and E and an encompassing parent such as A and D, that more than likely sheds more light about the correlation of elements in between. Perhaps the intervening elements in one schema correspond to additional and yet discovered detail for the other schema.

Deriving Associations in Cycles

Other times, an association can be made that is cyclic — that is, there is an association between an encompassing element and the contained element between the schemas. An example of this would be: A contains B; C contains D. A is related to D and C is related to B.

Directed cycles often indicate the presence of aggregation and generalization conflict due to similar “sorts” of information organized or aggregated differently for different data-domains. The resolution of this conflict shows the advantages of the graph over the tree — a tree will steadfastly maintain

such hierarchical conflicts during analysis and frustrate its resolution while a graph does not.

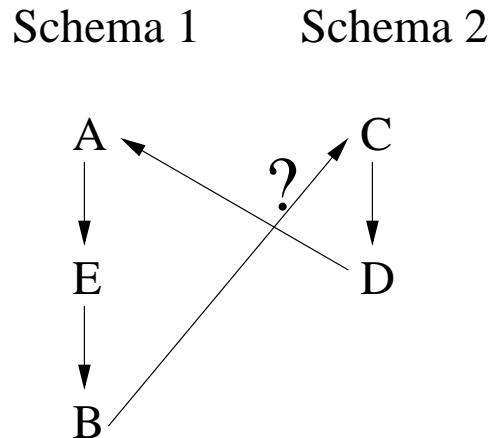


Figure 4-6: Cyclical Association Discovery

As noted in Section 4.2.3, this situation potentially allows X-Map to speculate on the meaning of intervening elements, too. Perhaps element E represents details from schema 1 that schema 2 does not yet exhibit and can be confirmed in the future through speculation.

Dealing with Conflicting Associations

Clearly, it is possible to derive contradictory associations such as “A implies B and A does not imply B” or one-sided associations such as “A implies B but B does not imply A”. The latter situation possibly implies a substitutable relation while the former is slightly thornier. In general, situations like this can be handled by having a persistent body of knowledge that either affirmatively says “yes, A implies B”, or “no, A does not imply B”. The fully qualified means of building this knowledge is an active area of research.

X-Map builds this body of knowledge through its relations linkbase in a couple of ways.

- Some schemas may be more trusted than others; thus, relations drawn from them may have higher weight to break contradictory associations.
- X-Map can fully speculate on this result and wait for future schemas to resolve this issue via

the regenerative association engine.

4.2.4 Regenerative Association Engine

The regenerative association engine embodies a simple logical concept extremely applicable toward XML Interoperability:

One should *never* throw away information¹², even uncertain ones, when trying to reason out and associate things — one never knows when one will need that information again.

This insight proves to be the key differentiator between X-Map’s attempts at drawing associations and all other previous systems who take a pristine, ivory tower view of associations for interoperability and just want to “do it once” and be done. COIN certainly prescribes to this view by trying to create mathematical axioms that describe a data-domain “once and for all”.

Hence, The engine’s design is surprisingly simple since its tasks basically entail the following:

1. Keep track of the speculative relations and what association algorithm produced each.
2. Rerun the associated algorithm on the speculative relations when new schemas are introduced into X-Map.
3. Record whether a speculative relation’s conflict is resolved affirmatively or negatively and act accordingly.

4.2.5 Preservation of Relations for Accessibility

After the heroic efforts on the part of X-Map, the actual preservation of a discovered relation is a conveniently simple leveraging of the Association Language (see Section 4.3) and the Transfor-

¹²Private Conversation: This insightful idea came from the experiences of Aditya P. Damle in his works.

mation Language (see Section 4.4). This is due to the fact that X-Map has already identified the elements that are associated, indirectly inferred what class of association it is, and possibly found a transformation to carry out the relation.

Thus, the major remaining tasks to relation preservation include:

1. How to organize the linkbase store for these relations.
2. How to organize the transformations store for the related transformations.
3. How to query and otherwise extract information from the above mentioned information stores.

Obviously, it is desirable to separate the linkbase (relations) store from the transformation store for optimization reasons. This allows the reuse of transformations across different relations, which is certainly possible.

Tasks #1 and #2 will use XML as a persistence store. In particular, task #1 will be stored as an XLink linkbase while task #2 will be an ordinary, valid XML document.

Using an XLink linkbase to store relations is natural considering that the Association Language is a derivative of XLink (see Section 4.3). However, the use of an XLink linkbase also brings several advantages, including:

Abstraction: The linkbase is a storage mechanism while X-Map is not. This increases code modularity and allows the linkbase to be optimized independently of X-Map (i.e. the linkbase can be a real DBMS or flat text file).

Code-Reuse: The XLink linkbase is a part of the official W3C XLink specification. Thus, other vendors will inevitably create implementations of XLink's programmatic interface. This also increases code modularity for X-Map and allows it to leverage the design, implementation, and testing of others.

Native XML Use: This is an obvious fulfillment of an influential factor of X-Map’s design (see Section 4.2 for details).

On the other hand, the storage needs for the transformations is much looser — it just needs to be able to hold and retrieve XML data. Thus, theoretically, any DBMS or even flat text file can suffice. However, in keeping with X-Map’s design goals, the storage medium chosen for X-Map will be a plain, valid XML document, which immediately makes it query-able via XQL¹³. Each transformation will be key’d by its language-type and transformation type (see Section 4.4) and the value will be the actual transformation in XML.

The astute reader should realize that both the XLink linkbase and XML document holding the transformations are directly query-able via XQL — one just has to construct the proper XPath expression to locate the desired information. Thus, in order for a relation to be used, one simply queries for the relation in the linkbase to obtain its transformation function’s XPath pointer, then use that pointer to query the transformation store for the actual description of the transformation.

4.3 Association Language Description and Rationale

The following figure and linkbase description are close-ups of Figure 4-1 that focus on the Association Language and depict the conceptual tasks that they perform. The example is a simple unit conversion from miles (obj2) to kilometers (obj3).

¹³XML Query Language [29] (XQL) — A query language patterned after XSL to find and select specific items inside a single or collection of XML document(s).

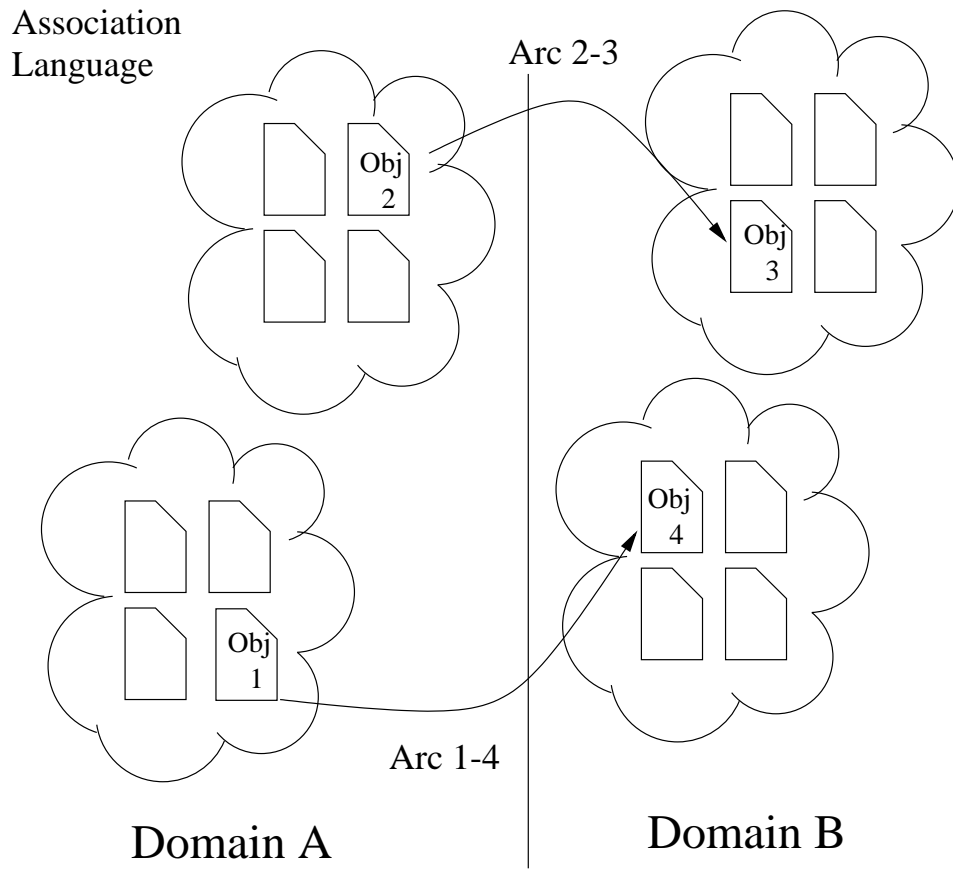


Figure 4-7: Association Language Overview

XLink linkbase (File of Relations)

```
<rdf:RDF
  xmlns          = 'http://www.x-map.org/2000/x-map-ns'
  xmlns:rdf      = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  ...

  <rdf:Description
    rdf:ID        = 'obj2-to-obj3'>

    <Resource xlink:type = 'locator'
              xlink:href = '.../obj2'
              xlink:role = 'role2' />
    <Resource xlink:type = 'locator'
              xlink:href = '.../obj3'
              xlink:role = 'role3' />
    <Link      xlink:type = 'arc'
              xlink:from = 'role2'
              xlink:to   = 'role3'
              xlink:actuate = 'other'
              xForm = 'xFormFile/rdf:RDF/rdf:Description/obj2-to-obj3' />
  </rdf:Description>

  ...

</rdf:RDF>
```

xFormFile (File of Transformations)

```
<rdf:RDF
  xmlns          = 'http://www.x-map.org/2000/x-map-ns'
  xmlns:rdf      = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  ...

  <rdf:Description
    rdf:about     = 'obj2-to-obj3'>

    <xForm lang='MathML' />
    <obj2-to-obj3>
      <apply>
        <multiply/>
        <ci>role2</ci>
        <cn>1.6</cn>
      </obj2-to-obj3>
    </rdf:Description>

    ...

</rdf:RDF>
```

Thus, one sees that the Association Language for X-Map is a straight-forward derivative from XLink augmented within the RDF. The reader should realize that the linkbase and xFormFile itself can be considered an RDF document that happens to be interpreted in an XLink-like manner¹⁴. The reason for this design is several-fold.

1. First and foremost, this choice fulfills most of the design goals such as favoring reuse of existing XML standards and tools, be in XML, human-readable and human-writable, and ability to reside within and outside documents being related.
2. Second, XLink has a very simple syntax and grammar for expressing arbitrary linkages between two or more resources with arbitrary numbers on both sides of the linkage (at least one resource must be on either side of a linkage in order for an association to make sense!). With minor modifications/additions of values to existing defined attributes, XLink can be extended to represent the abstract structure and “meaning” of semantic relations even better because it references resources by the “class” or role that they are in.
3. Furthermore, X-Map, being a derivative of XLink, is carefully crafted to maintain usability over the Internet and leverage tools and interfaces that understand XLink. And since XLink is an upcoming standard recommended by the W3C, supporting and using a known and existing interface is a definite plus.
4. Plus, XLink specifies its resources to be references written in XPath, which is a standard referencing language that not only allows XML data to be referred to as entire documents but also down to the individual characters within any tag element. It is completely neutral of the domain that contains the data — as long as one has an XML document and a hierarchical way to name that document, XPath can reference it.
5. Also, the enveloping RDF Description simply serves to “wrap” the XLink relation and tag it with an ID or about attribute so that the relations can work properly as an entity within

¹⁴In fact, the author considers RDF a better foundation to move to when tools that support it emerge, considering XLink’s semantics can be described by RDF.

RDF's scheme.

6. Finally, XLink also has a native mechanism of describing what to do when the link is “traversed” in its intended direction through the `actuate` and `show` attributes. The XLink specification defines a very browser- or viewer-centric set of attribute values that can be generalized to “other” for this project’s semantic association purposes.

Given this rationale, the following is the proposed Association Language description for X-Map.

Usage and syntax of X-Map shall be directly derived from XLink with the following additions:

XLink Usage that Remains Identical

type attribute is either `arc` or `locator`.

href attribute is still a URI or other domain specific means of identifying data.

title attribute is unchanged.

from attribute still designates an XPath to the originating point of the association.

to attribute still designates an XPath to the destination point of the association.

role attribute still plays its original semantic property of denoting semantically equivalent classes.

XLink Usage Modified by X-Map

show attribute will not be used by X-Map.

actuate attribute is always “other”.

The `actuate` attribute actually acts as a proxy-pointer to the location of the transformation function to be performed when data traverse from the `from`-role to the `to`-role. This is due to the fact that

the XLink specification [16] states that a value of *other* means anything the parsing program wants; thus, X-Map chooses *other* to imply the existence of an xForm attribute. The xForm attribute is an XPath statement that points to the actual location of the transformation function.

Now, the astute reader should realize that the above kludge of giving the actuate attribute a value of “other” just to satisfy the current XLink specification is not satisfying. Namely, if XLink positions itself as the universal linking language for XML for all purposes, why does it only allow browsing/viewing behaviors? Why does XLink not have a generic “behavior” attribute which pre-defines a browsing/viewing semantic but otherwise allows it to be used in arbitrary manners by an application?

In fact, this question is still under furious debate by the XLink design committee. As Eva Maler, one of the co-editors of the XLink specification [15] [16], said in private conversation with the author:

I can tell you that over the years, we’ve see-sawed back and forth about whether to put in the behavior attributes at all! At the time the March 1998 draft was written, we innocently thought they should be there and should be normative. Later, we backed off and made them ”hints.” Now we’ve ... returned to making them normative, or at least as normative as HTML is, and we carefully specify typical cases when you would not want to apply them.

Thus, the results of this thesis investigation may weigh-in its opinion to the W3C via a friendly W3C proposal.

4.4 Transform Language

As stated in Section 3.4, no unified XML language succinctly describes both associations and transformations in an expository manner. Consequently, it comes as no surprise that no single XML

language fulfills the Transformation Language requirements listed in Section 3.4.3, either.

The main reason behind this shortage can be gleaned from a careful examination of the Transformation Language’s requirements — specifically, the difficulty in resolving two of them, reproduced below and appropriate conflicting emphasis given:

1. The Transformation Language must be able to *express the computation* of new output component values from given input components.
2. The Transformation Language should be able to *express arbitrary transformations* to data.

Requirement #1 for the Transformation Language’s implies that the language must be able to generate an expression that ultimately leads to some output value given input values — even if it sacrifices the ability to express arbitrary data transformations. However, some transformations, such as string manipulations, do not “compute” to any output values. Meanwhile, some expressions that explicitly compute values, such as two array elements whose sum is stored in a third array element, may not immediately express meaningful transformations (i.e. what *are* the arrays that are being added together?).

The above examples illustrate the basic conflict between the operational nature of requirement #1 and the expressive nature of requirement #2. This dividing also appears in existing XML technologies. For example, XSLT [10] (see Section 2.3.7) lies squarely on the operational side of a transform language while MathML [21] favors the expressive¹⁵ side. In fact, no existing XML language bridges the gap between operational/functional and expressiveness, which eliminates the possibility of a simple solution for this thesis.

Instead, the author proposes a two-pronged attack on this problem:

1. Design a temporary kludge for this problem which basically melds the best representative

¹⁵MathML proposes two types of markup, presentational and semantic, and the latter proves to be the more semantically rich description of mathematical formulas.

languages (XSLT and MathML) together and labels each transformation with the language it is given in (such as MathML or XSLT). This label is kept with its relation inside the linkbase. The label also establishes which Mediation Engine to use since that depends on the language.

2. Put forth a proposal in front of a standards body like the W3C to aid in resolving this conflicting issue.

Now, the XSLT and MathML expressions for our “Transform Language” can be either kept with its associated relation inside the relations linkbase or in a separate persistent storage for transformation expressions. The author favors the latter approach since it untangles the storage of a potentially changing transformation language from a relatively more established and stable association language.

4.5 Mediator Design

The design of the mediator is perhaps the easiest part of the entire thesis. That is due to the fact that it is an application that uses the product of X-Map and is proof-of-concept that X-Map simplifies the writing of fuselets that utilize the critical knowledge of semantic relationships.

In fact, the mediator easily qualifies as a code template for future fuselets because it shows how to programmatically manipulate and use the results from X-Map toward a mediation task. They are:

1. Usage of the XLink programming interface to access the actual contents of the linkbase that stores the discovered relations.
2. Usage of the XQL programming interface to access the transformations associated with each relation along with XML data to transform.
3. Usage of either the DOM2 or SAX2 programming interface to access and create the data to be mediated and output its result.

- Usage of the Mediation Engines to perform any desired data manipulation specified by the transformation languages.

Mediator

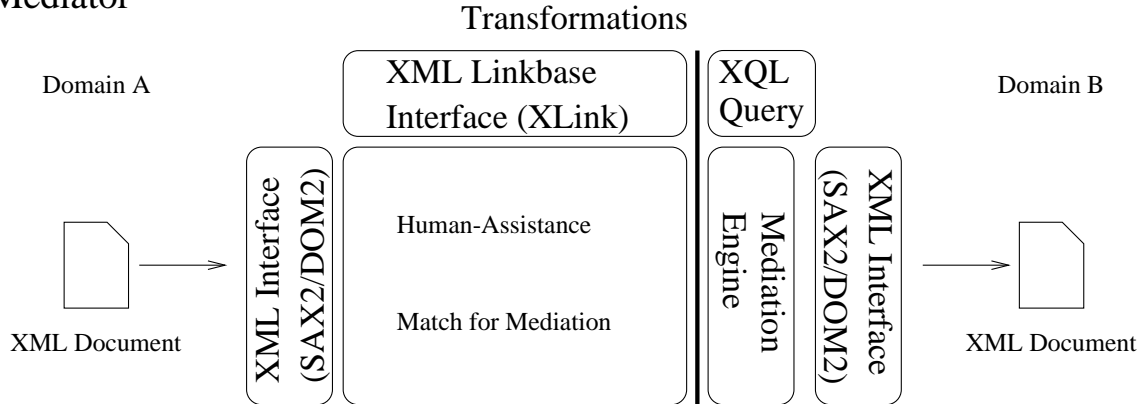


Figure 4-8: Mediator Design

As one can see from Figure 4-8, the Mediator essentially wields three XML interfaces to do its job. The Mediator loads the input XML document from domain A via the DOM XML interface, accesses the relations linkbase via an XLink interface, and finds the right relation for each relevant element. All this information is then handed to the appropriate Mediation Language depending on the transformation language. The Mediation engine then performs the transforms dictated by the relation's transformation function and uses XQL [29] to query for input values as necessary. Finally, the mediation engine outputs the transformation as XML and it is reassembled into a document of domain B.

Thus, one concludes that the combination of X-Map to discover and record the relations between different schemas in the Association Language *and* the Mediator, which wields the Mediation Engines and Transformation Language to perform the mediation act, do in fact achieve XML Interoperability for this project.

4.6 Summary of Design

An avalanche of design information and rationale was unleashed in this chapter. The following example re-examines the highlights of this chapter through a walk-through example.

Chapter 5

Trade offs and Conclusion

The results from this design of X-Map, the Association and Transformation languages, and Mediator has been satisfying. They have shown themselves to be very applicable to their engendering project's demands. The following succinctly summarizes the major benefits and trade offs of X-Map.

Advantages

- From the X-Map investigation, it became clear that a couple issues warrant recommendation requests to the W3C. Namely, the absence of an XML language that unifies the effects of MathML and XSLT — a language that is both operational and description, and the restriction on the show attribute from the XLink specification that forced a melding of RDF with XLink.
- By design, X-Map can leverage other tools and technologies to do its evaluation. It does not have to rely on its structural analysis and regenerative association engine alone. X-Map can also leverage more mature heuristics for determining which evaluation algorithm to run
- By design, X-Map not only leverages the knowledge it has but also speculates on knowledge it does not yet have. In combination with the Regenerative Association Engine, this enables

X-Map to gather more knowledge per schema than other efforts that only gather once.

- Associate once, reuse anytime and anywhere — By design, X-Map also extracts and preserves inter-data-domain relations *outside* of itself instead of embedding that knowledge in its code, meaning its work can be freely used by other applications or fuselets.
- X-Map also employs a dynamically evolving lookup table that grows more powerful over time when compared to static knowledge in tables.
- X-Map leverages established XML interfaces and the work of others who design and implement them.

Disadvantages

- X-Map's reliance on established XML interfaces also means it relies on other people for innovation in the interfaces. This is an unfortunate side-effect. On the other hand, open design groups tend to evolve technology quickly and with great foresight, allowing implementors rapidly follow their footsteps. This somewhat counteracts the reliance.
- X-Map's performance is not stellar — leveraging many interfaces and their implementations is not an optimal strategy performance-wise. However, this can be dealt with by adding more computing power.

Conclusion

Overall, X-Map's design bests the current state-of-the-art — its many advantages point at its leveraging of open interfaces and implementations, openness to leverage other projects' solutions as an aid to its own work, and re-evaluation of its own knowledge via speculation and re-assertion. In this regard, it is more forwardly minded than a solution like COIN.

However, improvements can always be made, and X-Map is no exception. The following items

represents a wish list of future research and work.

- Better Heuristics framework for determining which algorithm(s) to run and possibly in what order.
- More algorithms need to be adapted to plug into X-Map, along with the appropriate plug-in interface.
- Additional graph theory analysis to improve the abilities of the built-in structural analysis algorithm.
- Move to RDF from XLink if/when the tools arrive. RDF is more powerful and forward thinking and the transition is possible and not too difficult. Unfortunately, very little XML community support exists for it at this time.

Ultimately, the author considers the fanciful and structured view of the world taken by the past solutions to be unfoundedly optimistic. People are *not* inherently organized; even if they are, they are not organized in the same way. The same can be said about the information that these people structure and store. Any attempt to force a sophisticated, coherent, and structured view on such information will ultimately run into problems by trying to make something structured more than it really is.

The only alternative to such ivory tower approaches is to think practically and not “like a computer scientist”. X-Map attempts to capture this approach by assuming no sophisticated axioms or views except for the possibility of organization and relations amongst the information in some non-trivial manner. In essence, X-Map tries harder than previous systems to “think like humans do”, which ultimately is what interoperability is all about — resolving conflicts.

Bibliography

- [1] R. Ahmed, et al. (1991). The Pegasus heterogeneous multidatabase system. *IEEE Computer*, 24(12):19-27.
- [2] T. Berners-Lee, R. Fielding, L. Masinter, editors. IETF RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax. August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>
- [3] M. Biezunski, M. Bryan, S. Newcomb, editors. ISO/IEC FCD 13250:1999 Topic Map. April 19, 1999. <http://www.ornl.gov/sgml/sc34/document/0058.htm>
- [4] P.V. Biron, A. Malhotra, editors. XML Schema Part 2: Datatypes. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-2>
- [5] S. Bradner, editor. IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. March 1997. <http://www.ietf.org/rfc/rfc2119.txt>
- [6] T. Bray, D. Hollander, A. Layman, editors. Namespaces in XML. W3C Recommendation, January 14, 1999. <http://www.w3.org/TR/REC-xml-names/>
- [7] T. Bray, J. Paoli, C.M. Sperberg-McQueen, editors. eXtensible Markup Language (XML) 1.0. W3C Recommendation, February 10, 1998. <http://www.w3.org/TR/REC-xml>
- [8] Y.J. Breitbart and L.R. Tieman (1985). ADDS: Heterogeneous distributed database system. In F. Schreiber and W. Litwin, editors, *Distributed Data Sharing Systems*, pages 7-24. North Holland Publishing Co.

- [9] D. Brickley, R.V. Guha, editors. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation, March 27, 2000. <http://www.w3.org/TR/rdf-schema>
- [10] J. Clark, editor. eXtensible Stylesheet Language Transform (XSLT) Version 1.0. W3C Recommendation, November 16, 1999. <http://www.w3.org/TR/xslt>
- [11] J. Clark, S. DeRose, editors. XML Path Language (XPath) Version 1.0. W3C Recommendation, November 16, 1999. <http://www.w3.org/TR/xpath>
- [12] U. Dayal and H.-Y. Hwang (1984). View definition and generalization for database integration in a multidatabase system. *IEEE Software Engineering*, 10(6):628-645.
- [13] S. DeRose, editor. XML XLink Requirements Version 1.0. W3C Note, February 24, 1999. <http://www.w3.org/TR/NOTE-xlink-req>
- [14] S. DeRose, R. Daniel, E. Maler. XML Pointer Language. W3C Working Draft, December 6, 1999. <http://www.w3.org/TR/xptr>
- [15] S. DeRose, E. Maler, editors. XML Linking Language (XLink) Design Principles. W3C Note, March 3, 1998. <http://www.w3.org/TR/1998/NOTE-xlink-principles>
- [16] S. DeRose, E. Maler, D. Orchard B. Trafford, editors. XML Linking Language (XLink). W3C Working Draft, February 21, 2000. <http://www.w3.org/TR/xlink>
- [17] D.C. Fallside, editor. XML Schema Part 0: Primer. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-0>
- [18] C.H. Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. PhD dissertation. Massachusetts Institute of Technology, Sloan School of Management, December 1996.
- [19] C.F. Goldfarb, W.E. Kimber, P.J. Newcomb, S.R. Newcomb, editors. ISO/IEC 10744:1997 Architectural Form. August 27, 1997. <http://www.ornl.gov/sgml/wg8/docs/n1920/html/toc.html>
- [20] Information Personalization Agent Manager (iPAM). <http://www.cda.mews.org/>

- [21] P. Ion and R. Miner, editors. Mathematical Markup Language. W3C Proposed Recommendation. February 24, 1998. <http://www.w3.org/TR/PR-math/>
- [22] ISO/IEC 10646 ISO (International Organization for Standardization). ISO/IEC 10646-1993 (E). Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane. [Geneva]: International Organization for Standardization, 1993 (plus amendments AM 1 through AM 7).
- [23] W. Kim and J. Seo (1991). Classifying schematic and data heterogeneity in multi-database systems. *IEEE Computer*, 24(12):12-18.
- [24] R. Krishnamurthy, W. Litwin, and W. Kent (1991). Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the ACM SIGMOD Conference*, pages 40-49.
- [25] O. Lassila and R.R. Swick, editors. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. February 22, 1999. <http://www.w3.org/TR/REC-rdf-syntax/>
- [26] D. Megginson. *Structuring XML Documents*. 1998. Prentice Hall.
- [27] J.R. Munkres. *Topology*, 2nd Edition. 2000. Prentice Hall.
- [28] C.F. Naiman and A.M. Ouskel (1995). A classification of semantic conflicts in heterogeneous database systems. *Journal of Organizational Computing*, 5(2):167-193.
- [29] J. Robie, J. Lapp, D. Schach, editors. XML Query Language. W3C Proposal. September 1998. <http://www.w3.org/TandS/QL/QL98/pp/xql.html>
- [30] P. Scheuermann, C. Yu, A. Elmagarmid, H. Garcia-Molina, F. Manola, D. McLeod, A. Rosenthal, M. Templeton (1990). Report on the workshop on heterogeneous database systems. *ACM SIGMOD RECORD*, 19(4):23-31. Held at Northwestern University, Evanston, Illinois, December 11-13, 1989. Sponsored by NSF.

- [31] A. Sheth and V. Kashyap (1992). So far (schematically) yet so near (semantically). In D.K. Hsiao, E.J. Neuhold, R. Sacks-Davis. Proceedings of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5), pages 283-312, Lorne, Victoria, Australis. North-Holland.
- [32] M. Templeton, et al. (1987). Mermaid — a front end to distributed heterogeneous databases. Proceedings of the IEEE, 75(5):695-708.
- [33] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, editors. XML Schema Part 1: Structures. W3C Working Draft, April 7, 2000. <http://www.w3.org/TR/xmlschema-1>
- [34] The Unicode Consortium. The Unicode Standard, Version 2.0. Reading, Mass.: Addison-Wesley Developers Press, 1996.
- [35] L. Wood, et. al, editors. Document Object Model (DOM) Level 2 Specification 1.0. W3C Candidate Recommendation, March 7, 2000. <http://www.w3.org/TR/DOM-Level-2>
- [36] Report on Information Management to Support the Warrior, SAB-TR-98-02. December 1998. <http://afosr.sciencewise.com/af/sab/any/text/any/aftrstud.htm>
- [37] <http://diides.ncr.disa.mil/shade>
- [38] Protégé