XBRLS

how a simpler XBRL can make a better XBRL

March 11, 2008

Raynier A. van Egmond
XBRL Consulting Partners LLC.
rene@c3ismc.com

Charles Hoffman
UBmatrix Inc.
Charles.Hoffman@UBmatrix.com

Acknowledgements –

The information within this document has been accumulated over many years from many sources. Many people have either directly or indirectly contributed to the creation of the first "Patterns Document" and subsequent versions of the same information. The authors would like to thank those who have contributed to the collection and nurturing of this information: Thomas Egan, Josef Macdonald, Jim Richards, Roger Debreceny, Walter Hamscher, David vun Kannon, Jeff Naumann, David Prather, Geoff Shuetrim, Alan Teixeira, Hugh Wallis, Allyson Ugarte, Colm O hAonghusa, Trevor Pyman, Giancarlo Pellizzari, Yossef Newman, Rob Blake, Mark Creemers, Marc van Hilvoorde, Herman Fischer, Ignacio Hernandez-Ros, Campbell Pryde, Paul Sappington, Yufei Wang, Luther Hampton, Mark Schnitzer, KK Tang, David Hardidge, David Huxtable, David Scott Stokes, Sergio de la fe, Barry Smith, and Christopher Ball.

Summary –

This document presents a new approach for using XBRL that enables the non-XBRL expert to create both XBRL metadata and XBRL reports in a simple and convenient manner. At the same time, it improves the usability of XBRL, the interoperability among XBRL-based solutions and it reduces software development costs.

Today, most approaches to create XBRL reports require an in-depth understanding of the XBRL specification, but that will ultimately impede the user community from reaping the benefits that the XBRL language has to offer. Furthermore, current practices of building XBRL taxonomies increase costs for software developers, forcing them to continually implement new and often obscure features because someone creating a taxonomy figures out a creative or clever way to make XML Schema do something interesting, when other approaches for achieving the same result may already exist. Businesses are then forced to incur consulting fees and additional development costs in order to convert information created in XBRL, from one system's implementation of XBRL (dialect) to another system's implementation of XBRL (dialect). This is unnecessary because there is a better way.

At risk is the pervasive use of XBRL. If these developments are not carefully managed by XBRL International, the XBRL specification stands a big chance of being fragmented into a multitude of dialects. While the dialects can be interoperable, they can only be made to be so at the cost of high consulting fees and additional development costs, reminiscent of EDI and SGML. Those who will suffer the most will be business users, particularly smaller business users and software vendors.

This is the aim of this document is to propose an XBRL Business Reporting application profile. One can see this as a dialect, an approach, a technique, or an application profile. We will use the term "profile". The profile is based on best practices and tested techniques that have been proven to work by carrying out comprehensive testing using software applications available to a businessperson.

The proposed XBRL dialect for Business Reporting, named "XBRLS", brings to bear a combination of formalized best practices and proven techniques in XBRL metadata modeling, operating procedures, and support tools that will greatly simplify the use of XBRL for business reporting. It also simplifies the development and lowers the hurdle to market entry of XBRL tools and XBRL business applications, the very applications that promise to bring performance and cost optimizations to the business processes around internal and external reporting.

Those who stand to benefit most from the adoption of XBRLS will be the business users, business communities, regulators and the independent software vendor (ISV) community.

Introduction

Currently – March 2008 – the XBRL standard for business reporting is getting enough traction in the market place so that XBRL is being used by a much wider audience than just the language experts and early adopters.

The XBRL International Conference held in Vancouver BC, Canada last December 2007 exposed the XBRL standard to a wide audience, and the endorsement of the U.S. Securities and Exchange Commission (SEC) Chairman Christopher Cox gave a big push towards a broader exposure to the business community.

Additionally, the US SEC, the Financial Supervisory Agency (FSA) in Japan, and the International Accounting Standards Committee Foundation (IASCF) announced that they will collaborate to insure that their implementations of XBRL shall be interoperable. This is good, but why would they have to make such an announcement, and what about interoperability among other implementations of XBRL?

Furthermore, Mary Knox, Research Director, Bank & Investment Advisory Services of Gartner Inc., pointed out the need for defining best practices and other guidance tools, rather than creating more and more ways of solving the same problem with each new project that implements XBRL. She pointed out that this is a common phase that many standards go through.

With the introduction of the XBRL language outside of the expert and early adopter community, comes the responsibility of the XBRL community to make the jump from theory to practice. Since the conception of XBRL nine years ago, many projects have been undertaken. Most of these projects were in "closed environments" and all were supported by XBRL expert teams from the XBRL vendor community, XBRL consultants of the early stages, etc. Now that the language is being introduced into regulatory environments, we find that the users will be mainly business users. This user group is so large that there will be insufficient training resources to bring them up to the level of expertise required to use the XBRL standard and the tools in their current state.

Additionally, there is an ever-increasing number of "dialects" of XBRL being created. For example, the IFRS-GP, COREP, FINREP, US GAAP, and the FDIC taxonomy each has a different architecture. Imagine being a bank that must report to different regulators, each having a taxonomy created using a different approach or architecture. This is not cost effective for the business that has to report to all these different entities.

Many of those users who deploy an XBRL solution believe that all one needs to do is build a taxonomy, make it available, and then you are done, and your system of exchanging information will work flawlessly. Well, this is simply not the case. Having a taxonomy to exchange information is certainly necessary, but it is not sufficient. Things like FRTA and FRIS deal with some of the additional issues of implementing a full system, but they are not enough. Because of the missing solution components, each implementation of XBRL creates the missing components of the overall solution themselves, and usually in a different way. This causes the differences among dialects of XBRL. We will come back to this later in the document.

XBRL runs the risk of fragmenting into a number of different dialects. While they will likely remain interoperable – because they are all structured information, but just structured differently – it will come at a high cost of consulting and development fees to achieve this interoperability in order to convert one dialect or architecture to another. The other possible outcome is that we will see solutions (application + architectures) locking into a dialect and giving up on the interoperability. Given the fact that interoperability among XBRL solutions is not driven by existing business needs today, the latter scenario is, in our opinion, the more likely scenario. This will lead to a situation akin to the VHS and BETAMAX situation that occurred with video tapes. Furthermore, XBRL could even go the way of EDI and SGML, destined to be used by larger originations and regulators, not by the masses unless they are required to do so by these larger organizations or regulators.

The solution to this situation can be found in either of two directions:

- Find ways to bring the user community to the required level of expertise for filing reports; basically this means making every user an expert in XBRL,
- Bring down the required level of expertise for filing reports using XBRL to the level of the average business user by burying the complexity within software that a business user would typically use.

Clearly, it will not be the case that every business user becomes an expert in XBRL. This means that we have to hide the complexity, exposing only as much as what really needs to be exposed to the business user. Complexity can never be removed from a system, but it can be moved. The key, however, is to give software a chance to accept the responsibility of handling this complexity. To do that, certain things must be done. The primary thing is to give the software the information it needs in the form of rules that it can enforce, rather than forcing a user to understand and then enforce these rules upon themselves.

It is the authors' opinion that the solution lies in the second approach; "We – the XBRL expert community – need to bring the required level of expertise down, and thus enable the average business user to use XBRL to comply with his or her filing requirements". Another way to state this is to ask the question: "Can we apply the 80/20 rule?"

There is a precedent for this approach, XML itself. SGML was seen as "…insanely complex and studded with options, many of which interacted in surprising and hard-to-understand ways. Thus, SGML software was scarce, expensive, and often flaky." XML was built by "…taking SGML, throwing out most of the little-used advanced features, building in good foreign-character support, and a few other technical tweaks…"

(See: http://www.tbray.org/ongoing/When/200x/2008/02/10/XML-People)

This is precisely what XBRLS does; it throws out unnecessary options, little-used advanced features, it adds best practices, and this creates consistency. Furthermore, it eliminates the need for every implementation project to repeat the same process of discovering the best way to deliver a high quality taxonomy and a solid foundation for their XBRL solution. Using the XBRLS approach both increases quality and reduces costs. XBRLS may not work for everyone in every situation, but it will meet the needs of the vast majority of XBRL implementations. For the areas where XBRLS does not meet the needs, approaches can be amended for very specific purposes.

Using consulting dollars for this work are dollars well spent, as opposed to spending consulting dollars to recreate solutions that were already been created by others.

The approach presented in this document was first introduced by the authors of this document in 2001, very early in the development of XBRL. These ideas were articulated in what was known as "The Patterns Document." The stage, however, was simply not ready for the introduction of this patterns-based method. The feature set of the language itself still needed further development to meet the requirements for a comprehensive business reporting language. Over the years, we have never let our ideas too far out of sight. At key moments, we introduced documents and methodologies that prepared the XBRL community for the proposal we are putting forward in this document.

The ideas of patterns were further fine tuned during discussions at the Munich XBRL International Conference, focusing on patterns and meta-patterns and the differences between the two. Additional understanding and refinement of ideas resulted from the US GAAP Taxonomy project. Other taxonomies such as COREP, FINREP, CRAS, FDIC, and others have also contributed to ideas included within XBRLS. This document outlines the current thinking and improvements of these ideas, the idea of patterns.

The name for this new way of creating XBRL-based business reports is called XBRLS, which is an acronym for Extensible Business Reporting Language – Simple. It could be described as a dialect of XBRL or an application profile – a term that has been tried but never really embraced by the XBRL community. The premise on which we build this approach is that by giving up some of the flexibility (*read complexity*) in the way of expressing syntax in XBRL, but none of the features and metadata richness of XBRL semantics, the language becomes usable by a much broader audience than is currently the case. Analysis of a great number of financial statements by the authors over the years has lead them to recognize a number of basic data patterns that recur in financial statements. A standard approach to represent these patterns in XBRL is what is needed to report information in financial statements. Recent work has included updating the existing patterns to use the XBRL Dimension Specification, and we feel comfortable saying that by adhering to the use of a relatively small set of XBRLS patterns and the work procedures designed around them, any accountant can – after only some very basic training – create SEC compliant financial reports which are XBRL-based.

To illustrate how much simpler the use of XBRL can become; we can look at the US GAAP Taxonomy, which was recently exposed as a public working draft. This taxonomy, which contains something like 15,000 concepts and 20,000 relations, can be boiled down into just four or perhaps five business-reporting patterns. This may seem impossible, but it is nevertheless true.

The patterns-based approach to building taxonomies also results in a more internally consistent taxonomy, and the patterns-based approach is crucial if extensions of taxonomies are to be useful. First, preparers creating extension taxonomies will have vastly different skill levels than those creating the US GAAP Taxonomy. Creating incorrect or "downright pathological" extensions will be the norm if little or no guidance is provided to these lesser skilled users trying their best to articulate their extension information in a manner consistent with the base taxonomy.

It may seem odd or unbelievable that making something simpler actually increases functionality. If one looks to economics as an example, many people seem to miss the fact that the best way to work out a problem with an economic system is to improve productivity; that is the basis for

solving most problems. This is exactly what XBRLS does – it increases productivity. This is why XBRLS can be easier, simpler, and as good as (or better than) other more complex dialects of XBRL.

The adoption of XBRL is increasing globally. As more and more businesses implement XBRL internally, they want to be sure that it can be implemented effectively, efficiently, and that the implemented solution will be robust against future developments. XBRLS is an approach works and we believe it will last. Eventually, adjustments will need to be made to XBRL to address the issues raised in this document. The authors believe that the XBRLS approach is a business' best bet to future-proof their investment in XBRL.

XBRL Business Report Structure and Content

The Extensible Business Reporting Language (XBRL) is an open standard that supports information modeling and the expression of semantics commonly required in business reporting. XBRL uses the XML syntax and related XML technologies, such as XML Schema, XLink, XPath, XML namespaces, etc., to articulate this reporting semantics in the form of an information model. Most people focus too much on the XBRL syntax and miss the part about expression of semantics.

The purpose of XBRL is to provide a format that enables users to exchange and validate business information that is self-descriptive (*based on the XML specification*), that is a non-proprietary and open source standard (*XBRL is a public domain specification*), that supports an "open" user community (*one can use/repurpose any XBRL formatted information*).

To support these requirements, the language follows the model of separating:

- What constitutes a valid business report (structure and constraints – also called the "metadata" for a report), and
- The actual information communicated in the report.

We find this model with many XML-based languages for communicating information between systems and software applications.

In the case of XBRL, these two roles are performed by two different types of documents: the taxonomy document, which defines the metadata for a report, and the instance document, which provides the information of the business report. Since the instance document references the taxonomy document on which it is based, an application can validate the provided information against the structure and constraints set forward in the metadata. This also means that if the sender and receiver have access to the taxonomy document, the only thing to transmit is the actual business report information. The receiver can validate the received information using a local copy of the taxonomy. The sender can – before actually submitting the report – validate that all information is correct by validating the information against their local copy.

By separating the content and metadata, the transmitted information can be very concise, while at the same time ensuring full validation of submitted information at the source. This is one of the crucial benefits that enable high-value business process improvements.

Some Thoughts on XBRL and Extensibility

XBRL is built using the XML general-purpose *specification* for creating custom markup languages. Something that many people do not realize is that XML is not extensible, or rather not extensible in the way an extensible business reporting language needs to be extensible. That seems rather odd because XML is called the "extensible markup language", but it is nevertheless true. You can define a language (say *language A*) using XML. In that way XML is extensible as it allows you to define your own elements. Once you have defined *language A* however, you cannot extend that language with new elements or semantics (call it *language A'*) and expect software applications built to work with *language A* to also work with *language A'* and understand the new elements.

What you create with your "XML extension" schema, it will be useable by you but it will break other applications that do not expect that schema. This type of ad hoc extensibility is useful in a controlled and closed environment, but it is not the type of extensibility required for extensible business reporting.

XBRL is designed to be extended. Additions of new elements and relation semantics when done in compliance with the XBRL Specification can still be read by XBRL processors. Thus it provides a model of language extensibility that is not supported in the regular XML situation using XML Schema. It is the usage of XBRL Specification compliant (extension) taxonomies containing the elements and relations that make XBRL unique and extensible.

There are many applications within business reporting where this extensibility is highly desirable. Frankly, most people generally underestimate how much the extensibility will be used and it is doubtful that any taxonomy will be created and will NOT be extended (i.e., assume that all taxonomies will be extended). In meeting with clients, case after case, we have found that something which a client thought would never need to be extended, from their perspective, actually does need to be extended when you consider the perspective of other users of the domain knowledge expressed within the taxonomy.

Thus, XBRL is extensible and that extensibility is a very useful feature. XBRL is extensible in a "predictable way" or it should be. The goal is to provide support for building meaningful extensions that can be consumed by any XBRL specification compliant application and to interpret what the extension is communicating, not to break software applications.

One way that XBRL achieves this flexible extensibility is by not relying only on the XML Schema content model to express the metadata. This fixed content model restricts extensibility. What XBRL did was to use only simple types was to separate the metadata dictionary (the terms used in business reports and that are still expressed in an XML Schema) from the relations between these terms (which are expressed in linkbases). Using a "separation of concerns" approach that normalizes its language model, XBRL achieves its very flexible extensibility. The separation is done by:

> Making the relation semantics explicit rather than implicit, as is the case with an XML Schema content model

It is because of this "separation of concern" that XBRL can express an infinite number of possible hierarchies. XML Schema based languages can have one (the schema content model). The

XBRL tuples are the exception to this rule in that they do use the XML Schema content model for their content definition – with the extensibility problems as a result. Tuples use the XML Schema content model to "bind" concepts together into complex types.

This was not the case in XBRL 2.0; in that version, a linkbase (the definition linkbase) was used to express such complex type semantics – inline with the design philosophy of the XBRL language model architecture. When moving to XBRL 2.1, a lively discussion took place in the XBRL Specification Working Group as to whether XBRL should continue to use the definition linkbase to express complex concepts, or whether XBRL should take advantage of XML Schema content models creation of complex types and get some XML parser validation support "for free". Using XML Schema won out. As it turns out, that validation was not "for free".

Several years of creating taxonomies and using XML Schema complex types for expressing tuples as (rather than the definition linkbase) have given provided a deeper understanding of the issues related the move to XML Schema for tuples. Most XBRL experts today that are building large-scale taxonomies agree that this move to XML Schema to define complex types was a regretful decision.

For the same reason that the XML Schema content model should not be used for other parts of XBRL, it should not be used for tuples either. It is simply too restrictive in terms of extensibility, it misses features that allows for reporting of duplicate and inconsistent data, and it increases modeling complexity.

During the design process of the US GAAP taxonomy, a thorough analysis was prepared by XBRL subject matter experts of all the pros and cons of using tuples within a taxonomy. The bottom line is that the US GAAP taxonomy contains no tuples and there was 100% consensus among these subject matter experts that tuples should not be used. As a side note: developments in the IFRS-GP taxonomy appears to be moving in this direction.

XBRL – When flexibility turns into unnecessary complexity

Consider the following: There are three technical solutions to represent the same metadata within a taxonomy. For example, a piece of information is about director compensation, whereby one would request a report creator to provide information relating to the Salary, Bonuses, Fees and Options granted for each director who received such compensation. The information in presentation form might look like this:

Name of Director	Salary	Bonus	Director Fee	Fair Value of Options Granted
John James	0	0	60,000	0
Buck Rogers	879,639	1,213,486	0	569,000
Clark Kent	0	0	24,200	0
Lois Lane	0	0	57,000	0

Now, consider how a taxonomy creator might model this information:

- Using an item-based approach, one could create a concept for every intersection of director and the salary, bonus, fees, and options information. Each "cell" (intersection) is expressed as one XBRL concept. (This is the approach used to create the new version of the IFRS-GP taxonomy.)

- Using a tuple-based approach, one puts concepts inside that tuple to collect information about the director, their salary, bonuses, fees, and options. Each tuple represents a row in the report and each "field" within the row is expressed using a concept within the tuple. (This is the approach taken by the old US GAAP taxonomy and the old IFRS-GP taxonomy.)

- Using a dimensions-based approach, one creates items for salary, bonuses, fees, and options and then creates a dimension to associate this information with a specific director. Each row is represented by a [Director Member] on the directors dimension, each row column is represented by a concept; intersections are created within the context using the appropriate [Director Member] and concept combination. (This is the approach taken by the current US GAAP Taxonomy and the COREP taxonomy).

These are three fundamentally different technical approaches (syntax) for expressing the same information (semantics). Clearly, using three different approaches must negatively affect the comparability, the extensibility, and interoperability of these different taxonomies and clearly, this cannot be beneficial for the business community. It will only increase TCO (total cost of ownership) of the XBRL based solutions and negatively affect the acceptance of XBRL.

Furthermore, trying to figure out the right approach will cause frustration for the business users trying to create the taxonomy and require a higher level of training "to consistently pick" the correct approach. Additional documentation will be required with the taxonomy to communicate the correct approach for users of the taxonomy.

Looking at this problem from an end-user perspective, creating a user interface for a taxonomy creation application that can handle all three scenarios will be unnecessarily complex. It brings the underlying XBRL technology straight to the business users' desktop. As a point in case one need only to pick up one of the current taxonomy creation tools and experience it for oneself. Imagine the typical business user even having to pick between the three possible approaches to modeling the information.

The explosion of XBRL dialects

The number of dialects of XBRL is growing. A dialect can be viewed as a specific way to use XBRL to express information in an XBRL solution. The solution environment can be a closed environment (*which is normally the case today and typically consist of a regulator, reporting entities and a mandated XBRL information structure*) or open environment (*where such a central management body does not exist*).

With each new dialect, the projects incur unnecessary implementation costs, interoperability among dialects becomes an issue, and the value and relevance of XBRL for the business community decreases. Looking at the current environment, consider the following:

- Imagine a financial institution that must report to the U.S. FDIC, the U.S. SEC and perhaps some regulator in Europe, and each of these regulators uses a different architecture to construct their taxonomy, as it is the case today. The business user may need to use the FDIC taxonomy, the US GAAP Taxonomy, the COREP taxonomy (or the taxonomy of one or more of the countries using COREP), the FINREP taxonomy and the IFRS-GP taxonomy. Each of these taxonomies is written in a different XBRL Dialect, and the taxonomies are "for all intent and purposes" in practice not inter-operable.

- Imagine if, for some reason, every company in the world decided to put their financial reports on their web site using XBRL tomorrow. Given the multi-dialect situation today, what level of interoperability or comparability might be achievable when trying to compare or analyze this financial information? A case in point is the US SEC Voluntary filing program. One need only look at the inconsistencies among the filings to see the issues.

- Imagine that you are a software developer building software for the COREP, FINREP, IFRS-GP, US GAAP Taxonomy and the Netherlands taxonomy. Given the lack of a common approach to express BUSINESS REPORTING SEMANTICS in XBRL metadata, the products stay at the technical level of tools rather then reaching the level of value adding applications relevant to the business community. A case in point is the tremendous difficulty that the ISV community is currently experiencing in supporting the different XBRL dialects, sometimes even different versions of the same taxonomy, and the complete lack of business application that support the task and processes related to business reporting. Again, this is not a hypothetical situation but fact, and one need only talk to the software vendors and their customers.

- Imagine that you are a regulator trying to achieve transparency and comparability of reported business information. In this process, the business needs to extend a taxonomy with metadata pertaining to their company. However, what is in place today to ensure that two filers to the SEC using the new US GAAP Taxonomy will extend the taxonomy consistently, enabling comparability and transparency? What physical, tangible and verifiable proof exists, that shows that this extension process will work correctly? Again, one only needs to look at the material submitted as part of the SEC Voluntary Filing Program (VFP) program to see how differently people extend the taxonomy.

- Imagine a company that uses XBRL to report to a regulator, but also adopts XBRL internally and desires to map the following levels of information (each of which is at different levels of the same "financial concept". (Or, another way to look at this is to imagine that a company takes the "integrated approach" to using XBRL, imbedding XBRL deeply within their core information systems):

    a) Debt on the balance sheet, which is an item that makes use of XBRL Dimensions on the balance sheet;

    b) Debt instrument groupings information which is expressed as a tuple within the debt disclosure;

c) Debt instrument detail for each individual debt instrument of the consolidated entity;

d) Debt detail for 200 subsidiary companies of the parent entity each of which uses XBRL to express the debt instrument detailed information and each doing this in different ways;

e) Debt instrument transaction information using XBRL GL.

Each of the examples given above is drawn from the reality of today's XBRL business reporting environment. All these different implementations, these different dialects, clearly reduce the usefulness of XBRL. While technically interoperable – are all expressed in XBRL – in practice, they can only be integrated at very high costs.

Each of the above situations occurs because there is no specification in existence that fills the gap between the technical specification XBRL International provides, and what is needed to implement an operational, functional business reporting solution based on XBRL.

Why just XBRL, FRTA, and FRIS are not enough

The XBRL specification (XBRL 2.1 core and XBRL Dimensions, and the soon-to-come XBRL Formulas, XBRL Versioning and XBRL Rendering), and FRTA and FRIS specifications are used today to reach some level of standardization in metadata modeling and the expression of XBRL information. They are the first attempts to connect the process of business reporting to the tasks of metadata modeling and the expression of XBRL information. We find, however, that these three specifications are not sufficient to build a complete solution environment.

If one analyzes the XBRL, FRTA and FRIS specifications, it will become clear that their focus is on specifying technical modeling conventions, naming conventions, etc. They do not aim to prevent the proliferation of XBRL dialects that is a result of the many ways of implementing common reporting patterns. Furthermore, there is no real support, guidance or suite of best practices for the creation of extension taxonomies.

A very important aspect of metadata modeling that the XBRL, FRTA and FRIS specifications do not address is the manner in which XBRL taxonomies can or should be extended. To date, this issue has not been addressed in the XBRL community. At the same time, however, with the emergence of taxonomies like the US GAAP, the approach for extending the taxonomies is a crucial, determining aspect for the success of the XBRL solution.

In "closed systems" with a single owner for all the metadata, such things can be managed top-down. In the case of "open systems" where this centralized control function is not present, the base taxonomies can be extended in any way that the perusing entity sees fit, even if this goes against the way in which the designer of the metadata meant the metadata to be used. It becomes an issue of managing the integrity of the semantics expressed by the metadata. In the current XBRL environment – up to now – there have been no proposals to address this issue. The US GAAP Taxonomy Architecture points out some of these issues (in the taxonomy's architecture

logical model, section 3), but they do not solve the problem. The architecture articulates the notion of an extension point and extension rules.

Clearly, each XBRL project can implement its own solutions, in its own idiosyncratic manner, but such practices only aggravate the problem of the emerging XBRL dialects. As we have seen, this greatly increases the total cost of ownership (TCO) for the businesses implementing the XBRL solutions.

This is all very natural in the evolution of a standard; remember Mary Knox's comments that we mentioned earlier. Bringing up these points is, therefore, not to fault XBRL in any way; it is to point out a reality and to provide a way towards the solution for the issue. Again, as Mary Knox pointed out, this can be accomplished by formalizing best practices and implementation guidance.

XBRLS – an Approach, Technique, Application Profile, or Dialect?

XBRLS can be viewed as an approach or technique for building taxonomies and instance documents using XBRL. Every taxonomy is created using one technique or another only those techniques are not named and are ad-hoc approaches invented during the XBRL project. Generally, those techniques are not documented what can result in unintentionally inconsistent taxonomies. Typically, these ad-hoc approaches will lack automated tool support for quality assurance. The one-off nature of the taxonomy development process often times does not allow for the development of such tools and methodologies or otherwise result in quite expensive taxonomies. It takes time and experience to create an architecture or technique, document that technique, build appropriate testing tools, etc.

XBRLS can be looked at as an application profile. An application profile is simply a specification of what an application should expect to operate against. Rather than have to operate against everything possible within the XBRL specification, XBRLS explicitly states the subset of the XBRL specification language features software must support and the language feature that will not be used and can be ignored by the application.

XBRLS states explicitly what a software application should expect, how to process information provided specified in XBRLS markup and goes to great lengths to explain what to do when odd things occur. For example, if an exact duplicate fact value occurs within an instance document (concept, context, units, decimals, and value are identical), what is software to do? XBRL allows this type of situation. XBRLS does not.

XBRLS - Best practices, proven practices, flexibility

Eventually – when XBRL has been around for 10 years and XBRL is used a lot within different systems – it will be quite easy to determine how to use XBRL successfully. Other users of XBRL will copy existing successes. This is however not yet the point of maturity that XBRL has reached.

However neither is it the case that there is no experience in using of XBRL today. Systems are up and running successfully, delivering high returns on investments, meeting business requirements, the systems work; XBRL works.

The challenge is to strike the correct balance between creativity and finding new ways of achieving what users want to achieve, and having too many different approaches what makes it difficult to decide which variety, or dialect, of XBRL you should use.

What makes this increasingly challenging is that XBRL is still somewhat of a moving target. For example, some projects were implemented before the existence XBRL Dimensions; others choose to be bleeding edge and implement XBRL Dimensions as it is being created; others wait until XBRL Dimensions is a public working draft; and yet others want to implement their systems and not use new features until they have been around for a few years. As a consequence – the existence of multiple dialects and multiple implementation approaches.

And, who is to say what is the "best practice"?

New approaches being developed periodically that have "proven to work" but have never been implemented within a production system. Are they really proven? Can they really be "best practices" if they do not have broad use? How do you define broad use?

What was included and what was excluded from XBRLS is based on:

- Meeting a specific set of business cases.
- Picking the one technically superior way of achieving the use case, meaning multiple options are to be avoided.
- Minimizing the complexity of implementing features within software, particularly if the feature is simply not useful. This is in part to say that one person's feature is another person's bug. Note that unnecessary flexibility is not deemed to be a feature.
- Staying as close as possible to commonly used tools, such as relational databases, existing multidimensional analysis tools, etc and the existing body of knowledge to develop such solutions.

One could – based on superficial analysis – say that XBRLS approach is somewhat biased and looks like and specifically supports the US GAAP Taxonomy. The similarities between the two approaches – the ad-hoc approach used for the US GAAP taxonomy and the formalized approach which is XBRL – exist as the authors worked hard to get many of the aspects of XBRLS (before it was called as such) into the US GAAP Taxonomy. While the authors have learned a lot new things from the US GAAP Taxonomy development project, they believe that in many ways it represents the current state of the art of taxonomy design. (The co-author of this document Charles Hoffman was part of the group that defined the architecture of the US GAAP Taxonomy.)

The therefore is that there XBRLS is not biased towards the US GAAP approach or aims to copy the US GAAP taxonomy. Similarities are a result of continuity in input by the authors into the

USGAAP project and subsequent formalization of the XBRLS approach. As part of the process of Continuous Improvement, we have incorporated learning points from the US GAAP project into the XBRLS approach to XBRL taxonomy design and development.

At the same time, there is a realization that there is still a lot of latent creativity that will eventually be uncovered when it comes to XBRL. Until XBRL achieves a massive amount of use, all the real "features and bugs" will not truly be understood. There is nothing that will test something like the "school of hard knocks". XBRL really has not been to school yet.

The bottom line is that very specific business use cases were identified and tests and prototypes were created, options were all evaluated and decisions were made. The authors realize all too well that no matter what option is chosen, someone will disagree. To those who feel there is a better way, we invite comments, feedback and, more importantly, test cases, specific business cases and anything else that helps to improve XBRLS. The proof is in the pudding.

XBRLS - The Definition of an XBRL Business Reporting Dialect

It is the strong conviction of the authors that the only way to solve the current dilemma that XBRL faces in the community of business reporting is to define a dialect that supports all of the requirements for a business reporting solution or system. The key point of this sentence is the explicit statement that we need to **define a dialect** that covers the requirements for business reporting rather than let technical prowess and creativity drive onward, thus allowing the emergence of an unmanaged, ever-expanding number of ad-hoc dialects. Having such a defined and formal dialect will simply eliminate the need for adding to this seemingly ever-increasing collection of dialects. Not having such a defined dialect is what causes this undesirable proliferation.

This is the aim for this document, to propose the *Definition of an XBRL Business Reporting Dialect*. This dialect is based on the solid analysis of the requirements and business use cases that the business-reporting domain needs to be supported. These requirements emerge from the structures of business reports as they are used all over the world, in any business report that is created, submitted and exchanged for internal and external management reporting.

Working prototypes exist and show (prove) what business use cases are met. We believe that the business use cases met is 100% of known business use cases, certainly for financial reporting, and probably 98% of all business use cases; surely higher than 80% of what the vast majority of business users will ever need met.

So, what if one of your business use cases is not met, then what? Hire a consultant and have them address that situation, which is likely .01% of all other total use cases. Is it appropriate to burden the 80% or even the 98% with what only a few may need? Focus the consulting dollars that need to be spent into those specific areas. Or – if that be the choice – simply don't use XBRLS and use the ideas which you do want to use as best practice guidance. We do realize that no one solution will meet 100% of the needs of 100% of all the possible use cases that exist. However, for those who desire the benefits of XBRL, but who do not want to b exposed to its complexity, XBRLS may be a fit for your requirements. We feel XBRLS will fill a rather large niche of projects.

This document endeavors to show that we can define a dialect of XBRL that is:

- Is 100% XBRL compliant.
- Solves all these issues, which if not addressed would mean that each solution implementing XBRL would have to address in order to create a workable system.
- Saves on costs, effort and resources that would need to be deployed to amend such a solution because the implementer (a) needs not identify these issues, (b) needs not build proprietary solutions for such issues, and (c) can purchase off the shelf software already supporting the solution.
- Stops the trend of emerging dialects of XBRL caused by each implementer having to address and solve the same situations, but doing so in different ways.

We explicitly state though that it is neither the goal nor the expectation that 100% of all issues and problems of every XBRL solution will be solved by using XBSLS. However, there is significant room for improvement and opportunities for leverage, and deploying XBRLS will reduce the burden on every XBRL solution implementation project. The current gap for executing effective and efficient XBRL projects is, we believe, so broad that productivity gains from deploying the XBRL business reporting dialect will be significant.

The name for this dialect is XBRLS, which is an acronym for *Extensible Business Reporting Language – Simple*.

What the authors of XBRLS have done is to take the available information about XBRL implementations, analyze it, and condense it down into one set of "best practices". These practices cover 80% to 98% of the requirements that a business reporting solution needs solved.

It does so by standardizing on taxonomy design choices for such systems and – where options exist – simply make a choice for a design that fits best in the overall XBRLS principles (*explained in detail in the subsequent sections of this document*). Information used to devise XBRLS was not obtained in a vacuum, but rather by experts with years of practical implementation expertise, as well as with expertise in developing XBRL itself. Other implementations of XBRL over the years, such as COREP, FINREP, IFRS-GP, US GAAP, the Netherlands Taxonomy, CRAS, FDIC, and other such taxonomies were analyzed. Much of this information is first hand, as the authors have participated in creating these taxonomies and creating XBRL, FRTA, XBRL Dimensions, and FRIS. The authors are simply the custodians of this knowledge, and feedback is welcome to improve XBRLS.

It is important to realize that designers and projects for XBRL Business Reporting solutions are not forced to use this system. XBRL exists and anyone can make use of any aspect of the XBRL Specification. However, keeping the 80/20 rule in mind:

> "WHY MAKE 80 PERCENT OF USERS WHO WILL NEVER NEED OBSCURE FEATURES, OR WHO DON'T WANT TO GO THROUGH THE EXCRUCIATING EXERCISE OF FIGURING OUT WHAT BEST PRACTICES ARE, INCUR HIGH AND UNNECESSARY COSTS IN TIME, MANPOWER AND RESOURCES?"

Why can't they simply pick something up that is proven to work and is adequate to meet all their needs?

The only reason is that before XBRLS, no such system existed.

How the *XBRLS Business Reporting Dialect* is represented in the underlying XBRL is something that stays hidden deeply within software applications. It is not something that a business user (or the development team of an XBRLS – based solution for that matter) would ever care about or even need to understand. Rather, the business user will only care about whether XBRLS can meet all use case requirements. If it does, it can be used "out of the box". If it does not, the user can still make use of the parts of XBRLS that do fit, and then create proprietary solutions within the boundaries of XBRL, to meet their additional needs. There is neither the desire, nor the intent to add proprietary features to XBRLS that are not 100% in compliance with the XBRL Specification.

XBRLS - When less becomes more

While most people in the XBRL community will now relate the idea of design patterns with information technology, the idea actually emerged from the architecture community where Christopher Alexander invented the idea to describe functional design units for new houses that would create "*a desirable effect*". In other words, they were 'recurring solution structures for often found, common problems'. In the development of the idea of design patterns Alexander built an integrated pattern language that ranged in application and scope from the design of landscapes and cities down to design components within a single room.

The idea behind the concept of design patterns is that the problem for which we are designing a solution is typically not as unique as we might think. It is often a particular expression of a category of problems. Once we recognize the problem category, we know the solution pattern to apply to come up with a design. This concept is now a mainstay of the entire IT industry. The book Design Patterns by "The Gang of Four" was the book that introduced Design Patterns to the Information Technology community.

Introducing the concept of business reporting patterns

Very early in the lifecycle of the XBRL language, the authors recognized that the concept of design patterns would also apply to the design of XBRL taxonomies. Clearly, there are, more or less, common types of data patterns within financial reporting information (*in the form of annual statements, quarterly statements, etc*.) and business reporting in general. This information may be presented in different ways, but the data, the relations between data, etc, are more similar than different. Once one accepts that in the United States a balance sheet is reported in a certain way and in the United Kingdom and Europe in a slightly different way, the content of the reports is pretty much the same. A particular company might report an extra period here and there, or invert a table's rows and columns, but still these are only "variations on a common theme".

With this in mind, a definition for a business-reporting pattern could be "*A commonly occurring structure in a report to express certain relationships between stated facts*". An example is "*A table with totals*". The table with a total can be used to express many different summed items; all

fixed assets, all sales for a region, etc. The switch in approach is that taxonomy development based on the business reporting patterns uses a standard reporting unit as the basis and expresses such a reporting pattern always in the same manner, with XBRL markup and constraints.

The authors recognized that for the business community to embark on the modeling of metadata, an ever-increasing complex XBRL standard would pose problems for the language's adoption.

The authors bring the domain of IT and semantic modeling together within the domain of accounting, and introduced the concept of domain-based taxonomy modeling.

An early paper about the idea of using patterns was written by Raynier van Egmond in early 2001. After several years of work in creating the IFRS-GP taxonomy between 2002 and 2005, and trying to identify and understand patterns, this information was summarized in a book *Financial Reporting Using XBRL*, by Charlie Hoffman in 2006. This work introduced the concept of patterns to a broader audience, but at that time it had been used and field tested by the authors in many consulting engagements. Next, XBRL US issued the *USFRTF Patterns Guide* as a public working draft in early 2007. This patterns guide leveraged significantly the previous work that had been created.

While the XBRL expert community initially received the idea with limited enthusiasm, the authors received extremely positive feedback from the non-expert users of XBRL during many consulting engagements, training and other presentations, and by software developers who were using the patterns for the testing of software. There were no real surprises. The positive feedback that we received bolstered our confidence that the idea of applying design patterns to the problem of modeling business report metadata was the right way forward.

The design patterns proved to be the right communication medium between the domain experts and the XBRL consultants designing the solutions.

In the early stages, the design patterns were still very 'crude'. The patterns were actually more like business use cases. The basis from which the commonality was extracted and the best practices gathered from designing the solutions was still limited. Cleary, the patterns emerge from many, many applications and are a representation of some "body of knowledge" for representing business reporting patterns using XBRL. At the same time, we also encountered issues with the kind of semantics that could be expressed with XBRL. The domain of XBRL based business reporting was – and to a certain extent still is – evolving.

From the approximately 28 or so patterns articulated within the book *Financial Reporting Using XBRL* (chapter 11), five "meta patterns" were derived. Basically, the business cases and examples created for training were distilled to their essence, arriving at a mere five core meta patterns which create a framework for XBRLS.

Issues with the early XBRL patterns

The early XBRL patterns were many and provided an inventory of all common reporting structures found in financial statements. Some examples are "*The table with totals*", "*The table with nested sub-totals*", "*The movement analysis table*", "*The table with unlimited rows*".

While providing a solid foundation for the use of design patterns for taxonomy design, we encountered issues with the design of the XBRL language itself to provide well-contained and

modular design patterns. The problem with XBRL was – **and still is** – that report metadata is found in both the instance document and the taxonomy document. One could argue that this is a design flaw that was present in the language from its very inception. It never emerged because the information expressed in the early stages of XBRL was not dimensional.

A short explanation about Dimensional Information.

Information in the realm of XBRL comes in two flavors: metadata describing information, and data providing the actual information, referred to as fact values. The metadata is written down in the XBRL taxonomy document, whereas the data is written down in the XBRL instance document. An example of the two;

❖ measure metadata	"Amount of Sales"	describes what a number means
❖ data	12,000,000	gives the actual sales number

Now imagine that we wish to express the sales amounts for a series of stores. The meaning of the number stays the same "Amount of Sales": the figure is still a number, but we introduce a new type of information indicating to which store the number applies. Extending the example;

❖ measure metadata	"Amount of Sales"	describes what a number means
❖ dimension metadata	Store	["Seattle", "Tacoma", "Redmond"] describes to which store it applies
❖ data	12,000,000	gives the actual sales number

Thus, we can see that the dimensional information provides additional information about the measure – "Amount of Sales" in our case. In the earlier versions of XBRL-based metadata modeling, we would express each "measure – dimension" combination as a separate concept, like: "AmountofSalesSeattle", "AmountofSalesTacoma", "AmountofSalesRedmond", etc. This proved not to be a manageable design solution that is robust enough for future taxonomy developments and changes. An XBRL extension specification, XBRL Dimensions (XDT), was developed and released in 2006 to provide a standard way for expressing dimensional information using XBRL.

Once XBRL provided the capability to explicitly express dimensional information, it enabled us to make much more concise and clear design patterns for reporting business information. In a way, the release of the XDT provided a missing semantic expression for the markup language XBRL.

TECHNICAL SIDE NOTE:

From a formal language design perspective, contextual information like the instance document "entity" and "period" elements are also a form of dimensional information. They also provide additional information about the interpretation context of the measure. If XBRL Dimensions were used in lieu of these "quasi-dimensions" (as it is commonly done in multi-dimensional modeling and data warehousing solutions) the XBRL language model becomes altogether more consistent and easier to use. "Units" is isolated and there is typically never a hierarchy of units, so it is less important to express this way.

Introducing XBRLS meta-patterns

The XBRLS meta-patterns provide a very clean and consistent way of expressing business reporting design patterns. The "*Simple*" extension to the name that puts XBRLS apart from the regular XBRL modeling approaches lies in the following:

THE FOUNDATION OF THE MODELING APPROACH IS A FIXED SET OF DESIGN SOLUTIONS TO EXPRESS REPORT METADATA.

XBRLS makes heavy and consistent use of XBRL Dimensions, which are an integral part of the approach.

The XBRL specification allows for a tremendous amount of flexibility in expressing the metadata about business reports; so much so, that for each possible 'reporting' requirement, a large number of valid, but different, modeling and taxonomy extension solutions can be created, each producing valid instance documents. This results in inconsistent base taxonomies and even more inconsistent extension taxonomies. The inconsistencies make it harder for those trying to understand the taxonomies or compare information based on such taxonomies.

An extreme example of this would be create your own way to use XML Schema to articulate segment and scenario information as part of a context. To come up with a comprehensive specification would merely mean to redesign something that XBRL Dimensions already supports. One would, in a sense, be writing yet another specification. This, however, has many unwanted consequences; it does not support comparability of submitted reports, and it requires an in-depth understanding of the XBRL specification to design correct metadata representations of the business report semantics. The former impedes the achievement of business process improvements objectives; the latter impedes the broad adoption of the standard. Neither of these consequences is desirable.

So why would anyone want to do that? Why not simply always use – in this case – XBRL Dimensions?

That is the principle on which XBRLS is based; reuse working solutions rather then design and build your own. Consequently, software implementation is less costly because the unnecessary features do not need to be implemented, and thus comparability is enhanced, XBRL Formulas will work better with the data, etc. This is simply the most practical thing to do, and this is what XBRLS does.

The way to arrive at the XBRLS solution was found in the analysis of financial reports and other business reports in order to create a set of design patterns that would cover the reporting requirements for financial reporting.

New in this cycle of the pattern design is the specification of a work procedure, supporting tools, and best practices to use the XBRLS meta-patterns. The XBRLS meta-patterns have been

developed in combination with software application prototypes for creating XBRL metadata and XBRL instance data.

The basic premise for the procedure design, using the tools and implementing the best practices, is that the procedure and the software **MUST** target the domain or business user, fit in their workflow and their tasks, and expose the user as little as possible to the XBRL standard.

Brief Overview of the XBRLS Pattern Components

A detailed explanation of XBRLS is beyond the scope of this document. The authors invite the reader to explore XBRLS using the following resources:

- The "XBRLS Patterns Guide", which is a document explaining the meta-patterns and the business use cases that are supported by the patterns.
- The sample files, which provide examples of the XBRLS meta-patterns and XBRLS instance documents
- A comprehensive example that takes all known business use cases and puts them into one business report; it explains the creation of the taxonomy and instance document using XBRLS as an illustration of how each of the business use cases is supported.

However, in order to help readers understand XBRLS, it helps to understand the characteristics of XBRLS. In this chapter, we will briefly explain the characteristics of XBRLS.

The XBRLS meta-patterns are modeled after multi-dimensional tables. There are the facts, and there is information about each fact expressed as dimensions. The meta-pattern recognizes three main information categories:

a) The measure, or primary item, which is the concept being reported;

b) The period axis (dimension) information that describes the period for which the information is being reported;

c) The set of axes (dimensions), in the form of a "reporting axis", that describes a varying domain property of the reported information;

Note that units are left out of this explanation, as they generally do not cause any particular issues and are otherwise straightforward. Each numeric concept will have a unit.

Whereas the above categories are used more to report the factual information, we can also use the XBRLS meta-patterns as a convenient mechanism to create metadata. In line with the idea of 'restricting options to make things easier', we introduce the ideas of the *XBRLS Pattern Extension Point* and *XBRLS Extensibility Rules*.

The XBRLS Pattern Extension Point

The XBRLS extension point indicates the only location(s) in the meta-pattern where a user can introduce their own metadata (concepts or relations) into the XBRLS patterns. An extension point is a logical point at which a meta-pattern can be extended.

It is simply not a real world situation for a report structure (*the business user object represented by the meta-pattern*) to be extensible "anywhere". Deciding what, where and how to extend taxonomies that represent reports is exactly the problem business users have with creating or extending XBRL taxonomies. There is no guidance to the business user for the extension process, and the existing tools are "free form", allowing literally anything, rather than guiding the user to do what they are meant to do. To overcome this situation, a higher level of knowledge is required by those creating, as well as extending a taxonomy.

It is important to recognize that XBRLS does not allow the user the possibility to change the actual structure of the meta-pattern, but only to add metadata used by the meta-pattern in the specific areas where extensions are allowed in that particular pattern.

An example will clarify the distinction. Consider a simple sales analysis:

Type	**Value**	**Explanation**
Concept	Sales	*Describes what a number means, the concept being reported*
Business Segment [Axis], Business Segments, All [Domain] **[.:EP:.]**	Generics Segment [Member]	*Describes to which store the "Sales" applies*
Region [Axis], Regions All [Domain] **[.:EP:.]**	Europe [Member]	Describes the product to which the "Sales" applies
Fact Value	6,675,000	Describes the actual value

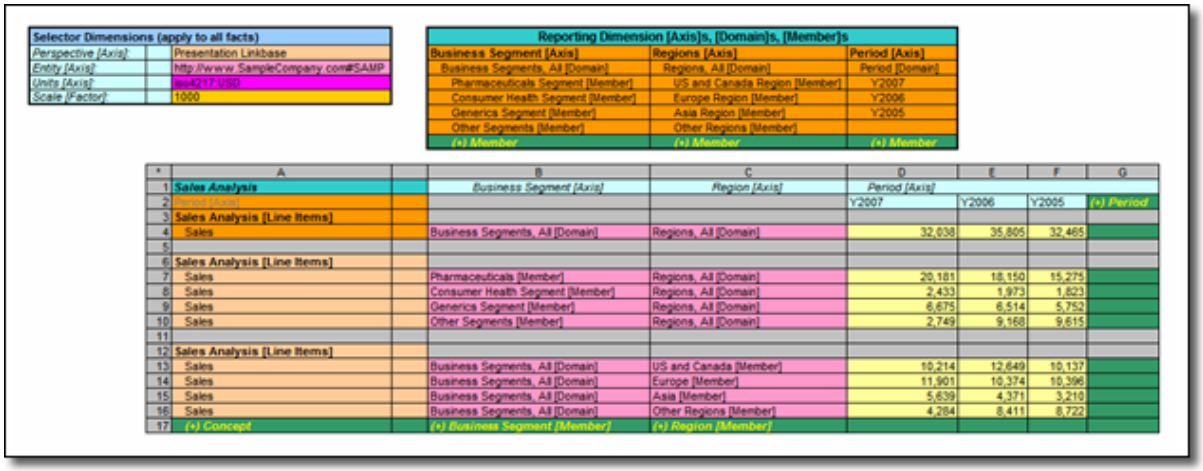
NOTE: THE INDICATOR [.:EP:.] IN THE SCHEDULE MARKS AN EXTENSION POINT.

If we define extension points on the store and product dimensions/axes, a user of the meta-pattern would be allowed to add additional stores to the store dimension/axis and must provide their own product codes to the product dimension/axis. The structure of the meta-pattern is fixed: facts and two dimensions, but the metadata content is changeable. The designer of the meta-pattern decides where the extension points are if this mechanism is used. This depends on the domain being modeled and the specific business process to support. Another application might require that all the dimensions be fixed, but that the set of reportable facts, the measures, can be extended or specified.

Contrary to the above, if the taxonomy extender did not understand where or how to extend the taxonomy, they might, as an example, create a new concept for every store/product combination they wish to report such as "Amount of Sales, Seattle, Product A". While this may be appropriate in some cases, as comparability might not be important, it will not be appropriate in others. What is certainly not appropriate is if you have each extender of the taxonomy creating their extension in different ways because they have no real guidance on where and how to extend the taxonomy.

A mechanism for visualizing information within an XBRL taxonomy and instance document is useful. One is provided below. But first, we need to explain another notion which should be understood to help read the diagram. An axis has one of two characteristics: "fixed" or "variable". Fixed and variable describe how the axis applies to a fact value; they are not different types of axes. For example, if a domain value on a given axis is the same for all fact values, it is considered fixed. If not every fact value has the same axis domain value, that axis is considered variable; the user needs to visually see the value that applies to a specific fact value. Humans reading financial reports are better at implying information if the information does not exist explicitly, such as "Thousands of US Dollars" in the heading of a report can be interpreted correctly by a user. This is not the case with a computer application. All information must be explicit.

Consider the diagram below:



The diagram shows all the properties of one of the XBRLS pattern we have identified and for which all the semantics are specified. The fixed, selector dimensions box in the top left which specifies (in this case) the entity about which we are reporting, the units in which the fact is stated, and the scale to apply to all the reported numbers. The reporting dimensions box in the top right specifies all the dimensions that are used to segregate the stated facts into categories. The fact table provides the numbers for the measures about which we are reporting, and it is indexed by their "dimension coordinates" [columns D4:F16]

We also see that for this pattern design, we can extend the content of the three dimensions with new domain members and that we can also opt to state new facts using these new dimension members. This is indicated by the extension points in the reporting dimension box and the cells B17, C17, G3 in the fact table. The designer of the patterns has opted to allow also the extension of the measures that can be reported in the pattern.

What may not be immediately apparent to the reader is that by using the specific meta-pattern we can immediately infer a lot of additional information that can be converted into XBRL metadata.

For instance, based on the specific meta-pattern used, we know that there is an aggregation that sums up information; in this case, the sales over business segments regions information must tie

to the "all-region" , the sales for "all-business segment", etc. The equality of these two results and the stated overall total number is a verifiable fact that an XBRL processor can validate against the submitted information. The XBRL markup that needs to be specified in the XBRL taxonomy metadata need not be provided by the person entering or extending the pattern. The pattern defines the structure for content and constraints, and since the intent of the pattern is known, the XBRLS processor knows exactly how and where to incorporate any new metadata in the underlying XBRL markup of an extension taxonomy.

The readers familiar with the current set of available XBRL tools and applications will appreciate the simplicity by which it is now possible to specify both metadata and instance data through the same interface, and in a work context that is relevant and familiar to the domain user.

The XBRLS Pattern Extensibility Rules

Extensibility rules are different from extension points, but they make use of extension points. While it may be logical to extend a taxonomy at a given point in the meta-pattern, the creators of the taxonomy may choose not to let the user extend the taxonomy at that certain points in the dimension. Therefore, extension points have extensibility rules.

For example, consider the following relation:

- Assets
  - Current Assets
  - Noncurrent Assets

Let us suppose that a taxonomy creator determines that it is not appropriate for the business user to extend the taxonomy at in what is otherwise a valid extension point. The reason may be that there are no other types of assets possible other than Current Assets and Noncurrent Assets. Thus, an extensibility rule is articulated which makes it illegal to extend the taxonomy at this logical extension point, due to domain knowledge.

Brief Discussion of the relation between XDT and XBRLS

The largest and most influential development of XBRL that influenced the use of the XBRL design patterns over the past three years was the release of the XBRL Dimensions Specification (XDT) in 2006. XDT introduced a completely new category of information that can now be properly expressed with XBRL. The introduction of the XDT creates the basis for the harmonization of the representation of the XBRL semantic model. All types of report metadata and report information can now be correctly and unambiguously allocated to the correct document types (*the XBRL taxonomy and XBRL instance documents*), and the XBRL standard will be able to mature into a much "cleaner" standard.

The use of the XDT allows designers of XBRL solutions to build much clearer and robust solutions than previously possible, and it provides another benefit, namely, the elimination of making some semantically irrelevant but technically fundamental design choices. Using XDT, we can harmonize on a single XBRL technology, whereas before we were forced to apply multiple

(design) techniques. Using one underlying design approach for the representation of the business reporting patterns greatly reduces complexity, both from a metadata design perspective and from a technology support perspective.

In addition, – in the authors' opinions – one of the most unfortunate missteps taken during the development of the XBRL specification was the choice to use the XML Schema content model to articulate complex type information in XBRL. While the process of trying to determine which route to take with the design of XBRL as a markup language involved significant discussions, the adverse ramifications of choosing XML Schema for the syntax of complex type information were not foreseen at the time.

If one recalls, in XBRL 2.0, the definition linkbase was used to model tuples. That approach was replaced by the approach of using XML Schema to model 'complex information types', tuples. In retrospect, and after significant learning and experiences in XBRL metadata modeling, experience has shown that introducing the XML Schema content model for tuples caused more problems that it solved. XBRL Dimensions, which also uses the definition linkbase, is, in our opinion, the better approach to modeling what amounts to complex data.

We arrived at this conclusion after an extensive analysis. It was shown that on the one hand tuples were too restrictive, inhibited extensibility, and limited comparability of information and on the other hand that nearly all the semantics of a tuple could be duplicated using XBRL and with additional support for lacking features, such as the ability to provide keyed values. Referring back to an example provided earlier, the technical decision regarding the choice of three design options that was presented above (items, tuples, dimensions and items) is now removed. This is done without compromising the reporting requirements and, at the same time, simplifying the metadata modeling and technical support requirements.

One extremely important benefit of the decision to abandon the use of tuples in XBRLS is that it

ALLOWS FOR SUBSTANTIAL SIMPLIFICATION OF THE SUPPORTING XBRL PROCESSORS.

This simplification will lower the hurdle to enter into the XBRL software market. Furthermore, the implementation of the mapping functionality within software is made simpler. This market is currently totally locked down by an extremely limited number of XBRL processor providers. We foresee that the result of opening up the market will enable the broad adoption of XBRL and the expansion of XBRL-based, value-adding applications.

Brief Characterization of XBRLS

XBRLS is 100% XBRL compliant. What XBRLS does is remove duplicate functionally, picking the best practices that have proven to work. XBRLS is not intended to be for everyone in every case; the 80/20 rule is used. However, XBRLS can be used effectively to meet **all the US SEC business use cases**.

XBRLS is a robust solution that lowers TCO for vendors of XBRL software and the businesses implementing XBRL solutions:

- Because it removes unnecessary XBRL language features.
- Because software vendors can more easily and for less cost build applications, as the algorithms are simpler and there are fewer exceptions that need to be supported.
- Because it formalizes and supports best practices, as opposed to unproven one-off design solutions.
- Because it provides all the mechanisms necessary to make the life of the business user easier.

If implementers of XBRL solutions wish to maximize the leverage they can use from best-practices supported by XBRLS, or if software vendors want to start implementing XBRL in the sweet spot of use cases, XBRLS can be the way to broaden the ubiquitous adoption of the XBRL specification.

XBRLS Technical Architecture Summary

The following is a summary of the XBRLS architecture and reasoning behind this architecture:

- The XBRLS dialect uses no tuples. Tuples are an unnecessary, syntactic representation for metadata in the context of XBRLS because XBRL Dimensions provides all the required semantics regarding modeling of complex information types that we need. Furthermore, XBRL Dimensions provides additional needed functionality that tuples do not provide. The XBRL Dimensions-based approach to articulating complex concepts covers all business reporting requirements and allows for a single XBRL modeling approach to implement these business reporting requirements.
- The XBRLS dialect only uses the segment element of the instance context and disallows the use of the scenario element (*See also next bullet point*). There is no reason for a user to have to decide if XBRL Dimensions information should go into the segment or scenario elements of the context component. As such, all contextual information specified in XBRL Dimensions will be placed into the segment context component.
- In the XBRLS dialect, ONLY XBRL DIMENSIONAL INFORMATION is allowed as content for the segment element in the instance context. Furthermore, EVERY concept (member, primary item) MUST participate in a hypercube and all hypercubes are CLOSED. In order to have comparability, there must be some specification driving the content of the segment elements in the context. XBRL Dimensions is such a specification. It allows for: (a) constraint of contextual information; (b) articulation of hierarchical relations within that information; and (c) XBRL Formulas makes good use of XBRL Dimensions. Mixing XBRL Dimensions type contextual information and XML Schema based contextual information is an inconsistent approach and only asking for trouble. As such, XBRL Dimensions is the only approach for articulating this contextual information. Likewise, mixing contextual information with XBRL Dimensions information, and contextual information with no information (i.e., empty), causes similar problems. Fundamentally, all dimensional metadata is explicit, and all is provided using XBRL Dimensions.

- In the XBRLS dialect, EVERY measure MUST exist in at least one XBRL Dimension. Again for consistency reasons, we disallow for some measures in a taxonomy and/or instance document to participate in XBRL Dimensions, and for others not to participate. Therefore, EVERY measure from an XBRLS compliant taxonomy reported in an XBRLS compliant instance document MUST participate in an XBRL Dimension.

- In the XBRLS dialect, no uses of simple or complex typed members are allowed within XBRL Dimensions. This introduces too much variability in the dimensions and, consequently, greatly complicates the XBRL processors. Additionally, typed members create, literally, an infinite level of complexity in creating the software interface for creating them. Typed members do little more than move XML Schema based dimensional information into a typed dimension. Finally, hierarchies cannot be expressed for typed members. From a modeling point of view, there is no reason to create complex XML Schema typed domain members. Furthermore, the use of simple typed members can be easily achieved using explicit members. Complex typed members can be created by creating multiple explicit members. In addition, explicit members keep the metadata within the taxonomy, as opposed to moving it to the instance document.

- In the XBRLS dialect, NEVER use the precision attribute, ALWAYS use the decimals attribute. The two attributes within an XBRL instance document serve exactly the same purpose. It is possible to convert from decimals to precision, but impossible to convert from precision to decimals.

Summary of XBRL components not used, and why

The following is a summary of the components of XBRL that are disallowed from use in XBRLS dialect, and the reasoning for not allowing the component. This information is summarized from the XBRL Specification and, as best as possible, cross-referenced to sections within the specification.

Specification	Topic	Explanation and Reasoning
XBRL Specification, Instance	Context: entity identifier, entity scheme	Although not required when using XBRLS, it is highly encouraged that the entity scheme and identifier be "held static" or synchronized with an explicit member and rather have XBRL Dimensions be used to articulate entity information, perhaps with an XBRLS "Entity [Axis]" dimension. The "entity identifier" and "entity scheme" portion of a context SHOULD NOT be used. Rather, the "entity identifier" and "entity schema" are STATIC (i.e., dummy values in order to pass XBRL validation), using constant values. The information articulates relating to the entity identifier and entity scheme are moved to an XBRLS specific taxonomy that makes use of XBRL Dimensions to communicate this information. **REASONING**: The reasons that the entity identifier and entity scheme are not used is because (a) there is no way to articulate a hierarchy of entity identifiers/schemes within XBRL; however, such a hierarchy CAN be articulated if this information is defined in XBRL Dimensions; (b) there is no way to attach one or more labels to an entity identifier/scheme, whereas this can be done using XBRL Dimensions; (c) this approach moves the articulation of metadata from the instance document to the taxonomy where other metadata is articulated.
XBRL Specification, Instance	Context: period	Although not required when using XBRLS, it is highly encouraged that the period context be "held static" or synchronized with an explicit member and rather XBRL Dimensions be used to articulate this information, perhaps with an XBRLS "Period [Axis]" dimension. Use XBRL Dimensions to articulate this XBRL quasi dimension. **REASONING**: There is no way to express a hierarchy of periods. Whereas it is possible to create some hierarchy as the hierarchy of period information is commonly known, there are other hierarchies that are not able to be articulated. The best example of this is the "fiscal period" which is commonly used within financial reporting.

Specification	Topic	Explanation and Reasoning
XBRL Specification , Instance (sections 4.7.4 and 4.7.3.2)	Context: segments, scenarios	Only use XBRL Dimensions to articulate the content of segments and scenarios, excluding the use of XML Schema-based contextual information allowed by sections. Furthermore, mixing XML Schema based-contextual information and XBRL Dimensions is technically dangerous. **REASONING**: XML Schema-based contextual information is too flexible, as there is no way to articulate hierarchy or constrain XML schema based contextual information. XBRL Dimensions achieve all these. Precedents for this approach are the COREP taxonomy and US GAAP Taxonomy that use this approach.
XBRL Specification, Instance	Fact Value: precision	Use ONLY the decimals attribute, precision MUST NOT be used. **REASONING**: Precision and decimals provides exactly the same functionality. There is no reason for both, particularly since that during analysis one approach will have to be converted to the other approach when data is analyzed. Precedent for this is FRIS section 2.8.11.
XBRL Specification, Taxonomy	Elements: tuples	Tuples are not allowed. **REASONING**: There are a number of negative characteristics of tuples. (a) Tuples reduce extensibility. (b) Tuples express meta-data within an instance document rather than in a taxonomy. (c) Tuples provide no way to articulate key values reducing comparability. (d) Tuples cause many issues relating to using items intended to be used within tuples outside those tuples, see FRIS section 2.8.3. Tuples and XBRL Dimensions could both be used in many cases to express meta-data. However, the XBRL Dimensions approach offers better desirable characteristics such as the ability to express key values. Having two approaches is considered a negative. Additionally, precedent for this approach is the approaches of COREP, FINREP and the US GAAP Taxonomy to make no use of tuples.

Specification	Topic	Explanation and Reasoning
XBRL Dimensions	Typed Members	Typed members (simple or complex) are not allowed. **REASONING**: Typed members are not allowed because: (a) typed members create significant implementation issues for software developers; (b) the needed functionality provided by typed members can be provided via explicit members; (c) hierarchical typed members cannot be created; (d) typed members provide metadata within instance documents rather than within a taxonomy. Precedent for this decision is also that the US GAAP Taxonomy contains no typed members.
XBRL Specification, Taxonomies	Weight	The weight attribute value of calculations MUST be either "1" or "-1", no decimal value between the two is allowed. **REASONING**: No taxonomy has ever used anything other than 1 or -1 for the value of weights. Apportioning using calculations is not a use case that will be supported. XBRL does not compute values; it articulates relations.
XBRL Specification, Taxonomies	Annotation, Documentation	Each schema and each linkbase MUST provide documentation that describes the contents of the file that is readable by a computer application. **REASONING**: Users should not have to rely on the file names for knowing what is inside a file. Applications should have access to this documentation that is helpful to business users.
XBRL Dimensions	Open Hypercubes	Open hypercubes are NOT allowed, only closed hypercubes are allowed.
XBRL Dimensions	notAll	Only "all" has-hypercube arcroles are allowed, "notAll" is not allowed. **REASONING**: In multi-dimensional analysis, this feature does not generally exist. This feature is difficult to implement and is not proven to be working correctly in existing XBRL processors (i.e., it is known to have issues). This may be allowed at a later time. Also, this can always be added via an extension taxonomy to enable this type of constraint.