13. Validating XBRL Instance Documents and Taxonomies[11]

This section focuses on the mysteries of XBRL validation. In it, we cover validating taxonomies and instance documents.

Validation includes steps a computer application can perform to check the data in an XBRL instance document to be sure it is correct, but also steps which only a human can perform. For example, a computer can test whether existing calculations of fact values add up; but a computer cannot detect that you intended to use the concept "ifrs-gp:CashRestricted" but inadvertently used the concept "ifrs-gp:CashAndCashEquivelants".

Understanding validation is critical to those who create XBRL instance documents, the reporting entities; but it is also important to accountants and auditors who may be engaged to examine and report on whether the XBRL-related instance documents accurately reflect what is contained in printed financial statements. Such engagements are performed by auditors which have adequate knowledge of the subject matter. XBRL is part of that subject matter.

Validating your documents to ensure they are created correctly will typically be part of your work flow and we have discussed validation in other sections of this book. However, this section is broken out in order to elaborated on validation here as validation can be complex and there are subtleties which should be understood. Also, there we cover only the highlights of validation, here we go into much more detail.

The process of creating XBRL instance documents and taxonomies must include validating the documents to ensure that they are correctly created. If they are not, they basically either cannot be used or will be significantly harder to use. Also, it will be impossible to use the documents in automated processes. One major feature of XBRL is to automate processes. Creating correct instance documents and taxonomies makes this possible; validation of the documents ensures correctness.

Validation is typically handled behind the scenes in XBRL software, at least most of it. While the software does do the work, it is important to understand what is going on so you realize what the tools do, or more importantly, don't do. Additionally, as XBRL is relatively immature, XBRL tools sometimes produce inconsistent validation results. This material will help you sort through these differences.

Lastly, just as XBRL is immature, also XBRL validation software is rather immature. How the software which performs validation today works will be vastly different that how valuation will work in the future. Very little real user feedback has been incorporated into today's tools which perform XBRL validation more along the lines of what application developers expect, rather than what is needed and desired in the real world of financial reporting.

13.1. Overview of Validation

First we will provide an overview of valuation and help you understand its various aspects. We will then walk you trough a typical validation work flow and point out issues you may not realize that do exist.

13.1.1. Purpose of Validation

When we talk about XBRL validation we are talking about all of the following wrapped up together into a process which ensures that instance documents and taxonomies which are created are syntactically and semantically correct:

- XML validation meaning are the documents well-formed XML is the most fundamental level of validation which is handled by an XML validating parser,
- XML Schema validation meaning the document is XML well-formed, and complies to the set of XML Schemas which are used to validate XBRL instance documents and taxonomies,
- XBRL validation which means all of the above plus that an instance document or taxonomy complies with the XBRL specification; XML validating parsers cannot perform this validation, only an XBRL processor can,
- FRTA validation which means all of the above, plus a taxonomy complies with the FRTA; these are rules above what is required of an XBRL processor,
- FRIS validation which means all of the above, plus an instance document complies with FRIS; this also is above what is required by an XBRL processor,
- Calculations are valid meaning that there are no undesirable or unknown "calculation inconsistencies"; known inconsistencies being ok,
- Semantically valid per a set or sets of business rules means that a document is at least XBRL valid, may need to be FRTA/FRIS valid, but also complies to a set of XBRL formulas.
- Comparison of the financial information expressed within an instance document with any renderings of the financial information to ensure the fact values in the instance and the values in the rendering are correct; for example, no concepts are added, no concepts are deleted, each concept has the proper value,
- Determining whether the appropriate concept was used in the instance document; for example, for the value of 1000, intending to use the proper concept "ifrs-gp:CashAndCashEquivlents", but inadvertently using "ifrs-gp:CashRestricted" or "ifrs-gp:TradeReceivables",
- The proper context and units information is assigned to a fact value, for example the period for "December 31, 2004" is intended and inadvertently using "December 31, 2003",

Some of this validation can be performed by a computer application, but some of it can only be performed by a human with knowledge of XBRL and financial reporting.

This seems like a lot, and somewhat overwhelming. Well, the good news is that the vast majority of all this will be handled by XBRL processors embedded in software applications.

Validating XML parsers, which include XML Schema validation, only get you past the first two items: XML well-formedness and instance document validity per a set of XML Schemas. Validating XML parsers do not cover the remaining items on the above list.

As mentioned earlier, it really is impossible to effectively make use of XBRL with only an XML parser; you really need an XBRL processor. XBRL parsers will likely be fairly inexpensive, at least for the core XBRL functionality. The XBRL conformance suite contains over 400 tests above and beyond XML Schema validation can test; to test conformance to the XBRL specification, XML parsers, whether they do XML Schema validation or not, simply cannot enforce any of these rules.

13.1.2. Benefits of Validation

The primary goal of XBRL is to effectively and automatically exchange business information. For the "automatically" part to work properly, validation is critical. Any errors in the process cause the need for human intervention which increases costs.

It is highly unlikely that information will be exchange between parties using XBRL without validation taking place on both the creation end to ensure what has been created is correct, and at the consuming end to ensure that what has been received can be reliably consumed within some sort of process.

The richer the validation set, the more likely the data exchanged will be accurate. Whenever you exchange information there is a likelihood of error entering the process. And if you really think about what XBRL is trying to do – which is make it possible for two unknown parties to exchange information – it is pretty incredible that these exchanges of data will even work at all, particularly between unknown parties using unknown software.

If the parties exchanging information know about each other, otherwise known as a "closed system", validation and exchanging information is relatively straight forward. Because it is a closed system, it is easier for the participating parties to communicate. An example of a closed system is the US Federal Deposit Insurance Corporation (FDIC). The regulator, the FDIC, regulates financial institutions, of which there are about 9,000 under their jurisdiction. Rules are written, communicated to the banks and software vendors, all parties work out the glitches, and off they go submitting data.

But in an "open system" all this is far more complicated. In an open system the party creating the information and the party consuming the information may not even know about each other or what software they are using. For example, say an entity put its financial information out on its web site in an XBRL format. An analyst, wishing to analyze that entity, consumed that information.

The purpose of instance document and taxonomy document validation is to help make this exchange of data go smoothly.

13.1.3. Problems with Validation

So, what are the things which can go wrong? Well, here is a list of some of what can go wrong:

- **XML well formed** – The document exchanged may not be well formed XML. This is very easy to detect and will be picked up by any XML parser or XBRL processor, which typically uses an XML parser for this task. From a global perspective, sometimes you run into character set issues, but this process of simply reading XML works very well.
- **XML Schema Validation** – The document exchanged may not be XML Schema valid. Again, this is pretty straight forward these days, but it was a bit more challenging only a few years ago. Validating an XBRL instance document or taxonomy to determine if it is XML Schema valid against a

set of schemas can be done by any XML parser which also performs XML Schema validation, and there are lots of these, and they are very low cost or mostly free. The problem is that many of these XML parsers which include XML Schema validation do not work consistently. This was a real problem with XBRL 2.0 but it has been mitigated to some degree with XBRL 2.1 in several ways. First, XBRL 2.1 was "dumbed down" slightly to make XML Schema validation of XBRL documents work as good as it could on the most used XML validating parsers. Secondly, the discrepancies between the parsers where communicated to the parser vendors so the bugs resulting in inconsistent XML Schema validation were fixed or are being fixed. Third, for the most part XBRL vendors use the same parser, Xerces, to perform XML Schema validation.

- **XBRL Validation** – The XBRL specification "specifies" how an XBRL instance document or taxonomy document is to be created. The XBRL specification is created by humans, and humans make mistakes. To catch these mistakes, a conformance suite is created to test to see how each vendor of XBRL software implements XBRL. This conformance suite is also created by humans, and contains errors. But the conformance suite does work. XBRL 2.0 did not have a conformance suite and there were significant software vendor interoperability problems. XBRL 2.1 does have a conformance suite, it has been exercised for over 18 months, and errors discovered by trying to implement XBRL are rolled back into the XBRL specification in terms of erratum and the XBRL 2.1 conformance suite. The conformance suite is pretty good at this point which creates a very stable XBRL 2.1 platform to build on.
- **Software Bugs and Errors** – Software vendors build their software and use the XBRL conformance suite to help ensure they are building their software correctly. If discrepancies are discovered, bugs in the XBRL specification and/or conformance suite are reported, and the specification and conformance suite become better and XBRL becomes even more interoperable. But, not all software vendors implement XBRL correctly as they don't use the conformance suite or run their software through it correctly. There are vendors who do a good job implementing XBRL, and there are those who don't do as good of a job. There is a status of "fully conformant XBRL processor", however, this is a self-regulating process and no one really exercises vendor's tools to ensure that they are truly fully conformant XBRL processors, which is discussed in more detail below. XBRL International has discussed having a "logo" type program which would improve cross vendor XBRL interoperability, but this does not currently exist.
- **Software Error/Warning Reporting** – Sometimes it can be difficult to determine if, in fact, you have an error. Different software vendors report errors differently. For example, if you are missing an role definition, some vendors report that as one error. Other vendors report the one error of the missing role definition and the 112 times that role was used. Other vendors do not report the missing role, only the 112 times the missing role was used. This can be confusing. Also, some software vendors make it difficult to segregate true errors, warnings which violate should rules of the specifications and best practices which they implement, which are very good ideas, but not errors or should violations. All this can make it quite confusing for a user to read error reports, create valid documents, and then exchange the documents with others with a different brand of software.

- **Inconsistencies** – Calculation inconsistencies are not errors, but they could indicate that there is something wrong with the data. But, there may not be anything actually wrong with the data, only that some of the calculations are not applicable to the data. Users really need to deeply understand calculations at this point to really understand if the data they create or consume do or do not contain errors or expected inconsistencies.
- **Taxonomy Errors** – At this point in XBRL's evolution, taxonomies have not been thoroughly tested by hundreds of real-life instance documents. It is highly likely that taxonomies contain errors. Correcting the errors is relatively easy, once they are detected.
- **Lack of Appropriate User Training** – Many users simply don't know how to use XBRL, are trying to make use of XBRL without an XBRL processor, don't know how to read the error/warning/inconsistency reports, and add to this the fact that the software just is not as helpful as it could be. The old saying goes, "Ignorance is bliss" translates to just because a user has not detected errors does not mean that there are not errors in the XBRL documents. If parties want to effective exchange data in an automated fashion, these communication problems resulting from different interpretations of whether an instance document is XBRL valid and contains no inconsistencies can be not only frustrating, but costly in terms of human intervention.
- **Inadequate Connection Between Rendered and XBRL Data** – Reviewing an XBRL instance document is not really an effective way of making sure all the concepts used are the appropriate concepts, the contexts are correct, the units are correct, and the decimal values are correct. Many times renderings of the printed financial information is totally separate from the fact values, meaning the rendering is not driven by the instance document data. Software tools for reviewing things like the appropriate use of concepts and contexts really does not exist at this time.

If everything worked well, user training requirements would be vastly reduced as an issue in creating correct instance documents and taxonomies. But, that is not the state of XBRL currently. As XBRL matures and these issues are addressed by XBRL International, software vendors, and users; XBRL can, and will, work far better in open systems.

XBRL working in a closed system is far, far easier than working in an open system.

13.1.4. Types of Violations

It is important to properly distinguish between what is an error, what is a warning, and what is a best practice. Also, inconsistencies need to be understood. We will define these here and use the terms consistently as defined.

Here, we are discussing computer testable rules for which an error, warning, or best practice violation will be returned by a computer application.

- **Error** – An error is the violation of a MUST rule in an XBRL specification which is testable by computers.
- **Warning** – The violation of a SHOULD or MAY rule in an XBRL specification.
- **Best Practice** – The violation of something a software vendor believes is a good practice, but is not enforced by the XBRL specification, FRTA, FRIS,

or other XBRL specifications. Different software vendors may call these violations by different terms; here we will refer to them as best practices.

- **Inconsistency** – An inconsistency is not an XBRL error, but it may or may not be an error in the XBRL instance document data. Consumers of the data who are knowledgeable about the data may be able to determine if the data is correct. The only one who knows for certain is the creator of the data.

Most software tools properly distinguish between violations which are errors, warnings, or best practices.

Also, not covered are violations which are manually tested, impossible for a computer to test. For example, the FRTA contains rule 2.1.1 which states, "A taxonomy schema MUST define only one concept for each separately defined class of facts." It is impossible for a computer to determine if two facts are defining one concept. Therefore, this cannot be detected by computer software application and therefore cannot be reported. These rules must be tested manually.

13.1.5. Understanding XBRL Conformance

XBRL conformance is defined by the XBRL specification. There are two levels of conformance: minimally conformant and fully conformant.

Anything other than a fully conformant XBRL processor is inappropriate for financial reporting as that processor cannot enforce all the XBRL rules which need to be enforced in this domain.

So, when you purchase your software, be sure it is a fully compliant XBRL processor.

13.1.6. Understanding FRTA Compliance

FRTA compliant is defined by the FRTA specification as satisfying all the MUST rules of the FRTA document. This includes all MUST rules, whether testable by a computer application or manually testable.

SHOULD rules of the FRTA are not included in FRTA compliance, but it is highly recommended that SHOULD rules be complied with also; but, again, this is not required for FRTA compliance.

13.1.7. Understanding FRIS Compliance

Similar to the FRTA, FRIS compliance is defined by the FRIS specification as satisfying the MUST rules of the FRIS document, whether they are testable by a computer application or tested manually. And similar to the FRTA, SHOULD rule compliance is encouraged.

In addition, the first rule of the FRIS is that all components of a taxonomy or taxonomies used by a FRIS compliant instance document require that those taxonomies are XBRL valid, which makes obvious sense, and rule two of FRIS requires that all the taxonomies be FRTA compliant.

This makes sense also. It is highly likely that taxonomies will be extended. Having a portion of a taxonomy being FRTA compliant and another part which is not FRTA compliant makes no sense at all. This defeats the entire purpose of the FRTA which is to create consistent taxonomies for use in open systems.

For example, FRTA rule 3.1.1 forbids adding unknown roles on resources or arcs; it would make no sense to allow unknown roles to be added by an extension taxonomy.

As such, FRIS complains requires FRTA compliance and XBRL compliance.

13.1.8. Understanding Calculation Inconsistencies

Calculations are quite easily created within an XBRL taxonomy creation tool, usually simply by creating a simple tree hierarchy using some user interface. Calculations are very effective in expressing relationships between financial reporting concepts. However, as you move up in levels of relationship complexity, calculations become somewhat limited; only expressing simple addition and subtraction and only within one context.

But, calculations are quite useful in helping to ensure accuracy of instance document data and can be supplemented by XBRL formulas which are discussed below.

If instance documents do not follow what is expressed within a calculation linkbase, the calculations are said to be inconsistent.

There is not such thing as a calculation error in XBRL. If a calculation is not what you might expect, it is deemed inconsistent. Calculations may be inconsistent for a number of different reasons, for example if calculations from a balance sheet for a financial institution "fire" (and you don't what them to fire) because they are in the same linkbase as the balance sheet calculations for a classified balance sheet (which you do want to fire because you have a classified balance sheet).

If you create an instance document and your validation of calculations generates inconsistencies, the two best options to make clear to users of the instance document are:

- Create a separate calculation linkbase which has only the calculations you desire to fire, disregarding the others.
- Use XBRL footnotes to document that the inconsistencies exist and should be ignored.

A third option, which is legal but not an endorsed solution is to prohibit calculations which cause inconsistencies. Other options exist, but a detailed discussion of calculations issues is beyond the scope of our discussion here, and the best options are detailed above. See the section on issues with the XBRL specification for that discussion.

13.1.9. Understanding XBRL Formulas (Business Rules)

XBRL Formulas, or sometimes referred to as business rules, are a very powerful feature of XBRL and can be used to enforce semantic validation of the data in XBRL instance documents. Far more powerful than calculations, which are rather limited, formulas can be used to test data for accuracy relative to other data in the instance or within other instances.

Basically, formulas bring a powerful rules engine based business rules validation engine to XBRL. Rather than each creator or consumer of data having to create programmatic validation, usually one-to-one validation which has to be created by programmers; XBRL allows one-to-many validation and puts the creation of this validation in the hands of business professionals.

We cover the creation of business rules later in this book, here we only want to point out that business rules exist and they are part of the process of validating XBRL instance documents. For example, users can validate:

- If the balance sheet balances.
- Disclosure checklist type validation such as, if "Property, Plant and Equipment" exits as a line item on the balance sheet, then certain concepts relating to Property, Plant and Equipment accounting policies and explanatory disclosures must also exist.
- Increases and decreases in current assets per the indirect cash flow statement can be validated against the actual changes in balances on the balance sheet.
- Ratios can be computed such as "Working Capital".
- If an entity has a loan, and the loan has certain covenants such as "working capital must be greater than 2.0", business rules can check for and report deviations from the covenants.

The above are only a few examples to show the types of relations might be expressed using business rules.

13.2. Validating Taxonomies

13.2.1. Steps Accomplished with Software

Validating a taxonomy is somewhat of a simple process when you are using an XBRL tool for creating taxonomies, if you understand the subtleties and the types of things which can go wrong. The things which you need to keep in the back of your mind will be discussed below.

The following is an overview of the process of validating an XBRL taxonomy which is application independent:

1. Open the taxonomy.
2. Go to the area of the taxonomy tool where you can start the validation.
3. Select the appropriate options. For all financial reporting taxonomies the choices are commonly:
    a. XML schema validation
    b. XBRL validation
    c. FRTA validation
    d. Show errors only at a minimum; warnings and best practices validation can optionally be shown, but be aware that the creator of the taxonomy may not have tested for warnings or best practice validation.
4. Validate the taxonomy.
5. Be sure the validation cycles through all imported taxonomy components, or you have to validate each separately.
6. Read the errors.

So here are a number of potential issues which we need to keep in the back of our minds:

- Some XBRL tools automatically perform XBRL validation when the taxonomy loads into the tool, some do not. The best way to be sure is to validate it after it is loaded, just to be sure.
- Some XBRL tools validate imported taxonomies referred to from your taxonomy, others don't. The best way is to validate each taxonomy component separately, if this is feasible, and if you are unable to determine if all the taxonomies in the DTS are being validated.
- You may be validating your taxonomy locally (or think you are) but are actually referring to components which are located elsewhere on the Internet. Sometimes it is easy to not see what is going on.
- Some taxonomy tools can actually load and validate taxonomies from the Internet, but this can be very slow.
- Some taxonomy tools cache a copy of the taxonomy and store it. We have to be careful of this if there is a probability that the taxonomy on the Internet may change, and that does not get uploaded and the older cached version is used.

At this point in the evolution of XBRL it is highly desirable to use multiple XBRL taxonomy validation tools to ensure your taxonomies are correct. XBRL International taxonomies are validated using three tools: DecisionSoft, Fujitsu, and UBmatrix. Most of these tools can be downloaded for a 30 day free trial.

So, there you have it; things which should be considered when validating a taxonomy.

13.2.2. Steps Humans Must Perform

While XBRL software which is designed to work with XBRL taxonomies can perform the vast majority of the steps to validate taxonomies, humans still must perform certain steps.

Many of these steps are required by FRTA, some are simply logical. While creating a base taxonomy like the IFRS-GP requires significantly more thought, knowledge, and effort, creating extension taxonomies still requires knowledge of taxonomies, and validating instance documents also requires knowledge of taxonomies.

Using the "best practices" features of software can help you adhere to many rules which are only testable by humans. Which software may not be able to make a definitive determination of whether you are complying with a rule, it can get you very close and steer you in the right direction.

The following are the major areas of concern. We will focus only on what is required.

1. Taxonomies should not define concepts which should already be defined. You need to check your taxonomy to be sure you have not, say, defined the concept "CashAndEquivalents", when it has already been defined. Sometimes scouring a 4,000 concept taxonomy to find a concept can be challenging.
2. Labels created need to be consistent with names created. For example, if you create a concept "CashAndEquivelents"; you cannot not assign a label to it such as "Accrued Liabilities". Labels and names need to be consistent.
3. Spelling should be consistent, not using "Proforma" in some places, "Pro-Forma" in others, and "Pro Forma" in yet others.

4. Balance attributes for monetary items should have a proper debit/credit value, if appropriate.
5. Concepts should have the appropriate periodType value for the concept.
6. The presentation linkbase and calculation linkbase should be consistent, for example, having a concept show up in the calculation linkbase, but missing from the appropriate location in the presentation linkbase.
7. Calculations should be expressed where they exist.
8. Best practice taxonomy modeling practices should be used to model your taxonomy. A good example of best practices in modeling taxonomies is what is called "The Patterns Document".

13.3. Validating Instance Documents

13.3.1. Steps Accomplished with Software

It is really impossible to know that you have a valid XBRL instance document unless you validate it. This is especially true for the calculations within an instance document.

The process of validating an instance document is similar to validating a taxonomy, and in fact, all the issues relating to taxonomies relate to instance documents also. The reason is that part of validating an instance document is validating the taxonomy or taxonomies associated with that instance document to ensure the taxonomies used are correct.

An instance document is another document which sits on top of the taxonomies or DTS which needs to be validated. There are a few things which you need to keep in the back of your mind, which will be discussed below.

An overview of the taxonomy validation process is as follows:

1. Open the instance document.
2. Go to the area of the instance tool where you can start the validation.
3. Select the appropriate options. For all financial reporting instance documents the choices are commonly:
    a. XML schema validation
    b. XBRL validation
    c. Validate all the taxonomies associated with the instance
        i. FRTA validation
        ii. Show errors only at a minimum; warnings and best practices validation can optionally be used, but be aware that the creator of the taxonomy may not have tested for warnings or best practice validation.
4. Cycle through any imported taxonomy components
5. Validate the instance.
6. Read the validation errors, calculation inconsistencies report, and the business rules validation report.

So again, here are some things to consider when validating instance documents:

- Consider all the same things which are applicable to taxonomies.

- You have to determine your strategy for validating the instance documents; do you want to download the taxonomies and the validate, or validate from the taxonomies from their location on the Internet. Using the taxonomies from the Internet can be slower. If you use local copies and your software does not support caching of local copies of taxonomies, you may have to change the locations of the pointers in the instance document to the taxonomy.
- If you don't use taxonomy extensions and the taxonomy you use is a public taxonomy which you know is valid, then you can safely not validate taxonomies when validating instance documents.

13.3.2. Steps Humans Must Perform

In this next section we discuss the steps which require a human to perform in order to validate XBRL instance documents.

1. The proper concepts need to be used to reflect each fact value. For example, review to ensure that "ifrs-gp:CashRestricted" was used, rather than inadvertanly using another concept such as "ifrs-gp:CashAndCashEquivements".
2. The proper contexts should be assigned to fact values. This includes the proper period, proper entity and segment, and proper scenario.
3. The proper units should be assigned to fact values, such as Euros, rather than inadvertently assigning Dollars to all fact values.
4. The proper decimals values are assigned to each fact value, for example not expressing that the fact value is to the nearest Euro, when it really is expressed to the nearest millions of Euros.
5. All fact values are included.
6. No additional fact values are included.
7. Labels attached to the instance are consistent with labels used in the printed rendering.
8. Best practices are used for assigning instance values within the instance document. See the best practices document "Assigning Instance Values".

13.4. How Online Validation Services Might Work

In this next section we will take a look at using a web service as a validation tool. Validators can be local applications, web sites with HTML pages you have to interact with, or web services which use WSDL which a local application interacts with and validation is carried out behind the scenes as if it were operating locally. Each method has its pros and cons.

Let's take a look at how a web service, using WSDL, might operate. Let's assume that you create an XBRL instance document creation tool using Microsoft Excel macros (Visual Basic for Applications), which will highly likely be a common approach to creating instance documents. XBRL instance documents are rather easy to generate simply by generating a text file using, say, the Windows file system object.

It is harder to validate the instance documents generated by your Excel application; that requires literally an entire XBRL processor's functionality. So, rather than re-invent the XBRL wheel, you choose to leverage an existing online validation service and pay for your validation on a per-validation basis.

You build your application and within your application you point your application to the web service and the following occurs:

1. Your application begins interacting with the web service using simple methods. Two methods you may have to implement are upload an instance, a second is download the validation messages returned.
2. The first method which occurs is that you send the instance document which is to be validated to the validation web service.
3. You may have to also send any taxonomy referenced by the instance document which is not available on the Internet; taxonomies on the Internet can typically be located, and may even be cached, by the web service.
4. The web service, which has a robust XBRL processor, does all the specified validation.
5. The web service sends you a validation results report in the form of an XML document, which your application parses and displays within your application.
6. Your account is charged an amount for the validation which was processed.

So, what the above shows is how anyone who knows how to create simple Excel macros can create highly functional XBRL instance creation tools. Macros create the instance documents, and then web services are used to validate the instance document which you created.