Business Rules and XBRL GL Profiles: A Schematron Experiment

# Introduction

The application of business rules to [XBRL](http://www.xbrl.org) is often thought to focus primarily on enabling regulators to communicate their sets of compliance rules to software developers and companies to “push data quality to the front door” – to give preparers the tools to prove their filings meet the requirements before the filings go into the regulator’s system. However, standardized business rules can have much greater potential – the ability to abstract the tests and formulas locked into programs and spreadsheets so they can be standardized and instantiated in any software on any platform on any device. These rules can then be used to promote internal as well as external data quality; standards-based approaches mean that rules can be more easily created, shared and analyzed by a broader community.

The purpose of this document is to discuss the potential role of rules in XBRL GL adoption and experiments with [Schematron](http://www.schematron.com/) as a means of conveying the rules. This document will have something to annoy both the technical audience (with certain technical concepts oversimplified for the business audience) and the business audience (with the inclusion of technical materials likely considered best left to the technical people, but necessary for the discussion.) For now, I won’t discuss nodes, code or if I have a code in my nodes. (Gesundheit!)

The ease of applying standardized rules to XML files is one of the things that can distinguish XML-based representations of data, such as accounting and ERP system setup, master, transaction and history files, from text and proprietary formats. Yes, XML Schema and XBRL taxonomies help make sure that you see numbers where numbers belong and dates where dates belong – but when you move to semantic meaning, you need additional rules, and for those rules, you need ways to reference XML data. The standardized W3C Recommended way to look at XML data is [*XPath*](http://www.w3.org/TR/2010/REC-xpath20-20101214/)*.*

# XPath and [XBRL GL](http://www.xbrl.org/GLTaxonomy)

Anything with an X in it must be hard to learn, and many of us don’t want to become programmers (or don’t have the time, skill or logical minds necessary in order to do so.) But just as XML has brought businesspeople and programmers closer than ever before, XPath requires a logical mind, the ability to “think like a tree”, and comes with both simple commands and a seemingly endless treasure trove of additional capabilities.

For example, in the world of XBRL GL, we know to expect monetary amounts in the primary field for that purpose, *gl-cor:amount*. We know that the XBRL GL Framework puts the amount in a consistent place in the XBRL GL tree –

/xbrli:xbrl/gl-cor:accountingEntries/gl-cor:entryHeader/gl-cor:entryDetail/gl-cor:amount

Table 1: Path to gl-cor:amount

|  |  |
| --- | --- |
| Element/Part of path | Comment |
| / | The starting point – not to be confused with the root element, which is a child of / |
| xbrli:xbrl | The root of all XBRL instances |
| gl-cor:accountingEntries | This is the virtual root of a batch of XBRL GL information. You get more information about the nature of the batch from the *enumerated* value of [gl-cor:entriesType], which is a child of the tuple [gl-cor:documentInfo], a child here. |
| gl-cor:entryHeader | The beginning of each grouping of information. |
| gl-cor:entryDetail | The holder of detailed information. There is no more than one [gl-cor:amount] per [gl-cor:entryDetail]. |

XPath lets us refer to any amount, anywhere as //gl-cor:amount, and lets us filter it using the milestones upstream, such as for a specific value of *gl-cor:entriesType*, such as the enumerated value {entries}. It provides a wide variety of tools you can use when referring to the information in the XML; let’s start with a simple, one, sum().

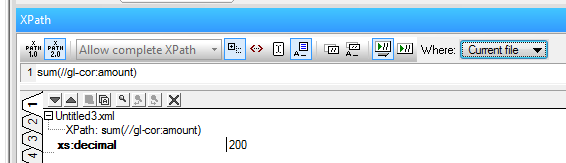
If you know that the amount field in an XBRL GL instance document is represented by *gl-cor:amount*, then getting a sum of the accounts using XPath can be accomplished by any XPath savvy tool with the XPath expression *sum(//gl-cor:amount)*. The following screen clip shows the XPath evaluation functionality built in to [Altova](http://www.altova.com) XML Spy[[1]](#footnote-1)

Figure 1: XPath for summing amount, in XML Spy

Want to know total debits or total credits? As mentioned, XPath lets you filter easily by adding filters up or downstream. The field *gl-cor:debitCreditCode* holds whether the item is a debit or credit, if that information is provided, so we add a filter one step up – to the *gl-cor:entryDetail* – to only consider (for example) credit values, using the enumerated value of {credit} as input. The XPath expression *sum(gl-cor:entryDetail[gl-cor:debitCreditCode = 'credit']/gl-cor:amount)* only returns the values when you filter the input for credit values.

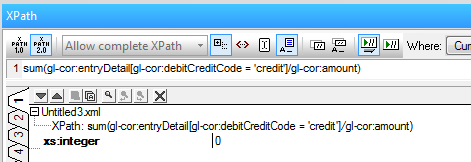


Figure 2: XPath for evaluating debits and credits, in XML Spy

XPath is important to many XBRL Specifications, including [Formula 1.0](http://www.xbrl.org/Specification/formula/REC-2009-06-22/formula-REC-2009-06-22.htm), [Variables 1.0](http://www.xbrl.org/Specification/variables/PER-2011-10-19/variables-PER-2011-10-19.html#sec-xpath-usage), [Period Filters 1.0](http://www.xbrl.org/Specification/periodFilters/REC-2009-06-22/periodFilters-REC-2009-06-22.html) and [Table Linkbase 1.0](http://www.xbrl.org/Specification/table-linkbase/PWD-2011-12-21/table-linkbase-PWD-2011-12-21.html). In addition, XBRL GL already references XPath in the [SRCD](http://www.xbrl.org/int/gl/2009-05-12/RUSouRCeD-2009-05-12.pdf) model, as a means of communicating the Summary Tuple Path and Detailed Content Filter.

# Rules and XPath

*Is it XML? Use XML “well-formedness” checking.*

*Does it meet the technical requirements as defined by the XML Schema creator? Use XML Schema validation.*

*Does it meet the “grammar” requirements of the taxonomy developer and the rules of XBRL? Use XBRL validation, Dimensions validation and similar devices.*

*Does it make “sense”? Time to turn to business rules.*

So XPath lets you slices and dice XML data, and XBRL uses XPath in many different areas. How can these be brought together for rules related to XBRL GL? Do we use the aforementioned Formula Specification? Possibly, and that will be the topic of an upcoming document. But quoting one of the Editors of the Formula Specification: “*XBRL Formula is quite closely wedded to XBRL, instance documents, facts, and a DTS.  If it's plain XML by itself, that's what* [*Schematron*](http://www.schematron.com/) *is good for.*”

Like Certs (“it’s a breath mint, it’s a candy mint, it’s two - two - two mints in one”), XBRL GL is “plain XML” and XBRL – two in one as well. If you don’t need to process the content of the DTS (in particular, the labels in the label linkbase), XBRL GL is pretty plain XML, so Schematron may be a good starting point.

So let’s start with the non-technical description of Schematron. I like to think of it as a way to **bundle** together XPath expressions to build a rules base. In their FAQ, they say Schematron is “A language for making assertions about the presence or absence of patterns in XML documents.” And list as the first, but not only, potential usage as “business rules validation”.

Schematron is an [ISO](http://www.iso.ch) (International Organization for Standardization) standard – [ISO/IEC 19757-3:2006](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip) – one of the *free* ones too. A number of major projects in the XBRL community have used Schematron, such as the SBR Australia effort, which teaches you how to [Build SBR into your applications](http://www.sbr.gov.au/software-developers/enabling-sbr-in-my-application) including Schematron files.

Getting Schematron “built-in” isn’t *quite* at the consumer level. You do have to ask your XML tools developer whether they support it. To my knowledge, XML Spy only supports it with a plug in (albeit one that costs less than US$20). The <oXygen/> XML tool, which I know is also used in our community, claims [Schematron editing and validation](http://www.oxygenxml.com/xml_editor/validation.html#schematron). When the stock answer is to “use the XSLT implementation[[2]](#footnote-2)”

That being said, my experimentation was with a trial download of XML ValidatorBuddy from [xml-tools.com](http://www.xml-tools.com/)[[3]](#footnote-3).

# Driving from end goal to the granular pieces necessary to get there

So Schematron is about bundling together XPath expressions to help in the automation of checks. Rather than run XPath expressions one at a time and trying to remember why you are doing the assessment and the implications of the results. Let’s go from abstract to concrete with this business requirement associated with **representing a chart of accounts**.

Some things I would expect to see (or not see) in an XBRL GL instance document representing a chart of accounts:

* The pivotal *gl-cor:entriesType* must use the enumerated value {account}
* There should be an *account* structure in each line of *gl-cor:entryDetail*
* There should not be any duplication of *gl-cor:accountMainID*
* Each account should have a *gl-cor:debitCreditCode*
* Descriptions should be formatted with a space following a “.”
* There should not be an *gl-cor:amount*

and perhaps even

* Accounts with numbers of 3000 and above should not mention “Cash”

So Schematron will let us capture these needs and the different checks needed to confirm them.

So with the abstract requirement “There should be an account structure in each line of gl-cor:entryDetail”. Now we could have established that requirement with XML Schema itself – just as the *gl-cor:entrieTypes* is required in every XBRL GL instance – but we don’t want to change the XBRL GL underlying schemas or taxonomy, just add an additional constraint for this special purpose – when checking XBRL GL specifically for representing a chart of accounts.

We can now establish a *pattern* for the required account and a rule that says “for each line of *gl-cor:entryDetail*, check for the existence of an account structure. If the account structure is missing, we’d like to return a message that “Every entry should have an account structure”. And what that might look like in Schematron is found in Figure 3:

Figure 3: A simple pattern in Schematron

<iso:pattern id="required account">

<iso:rule context="gl-cor:entryDetail">

<iso:assert test="gl-cor:account">Every entry should have an account structure</iso:assert>

</iso:rule>

</iso:pattern>

Although this isn’t meant to be a tutorial on Schematron, there are 6 basic elements in ISO Schematron: assertion, rule, pattern, schema, namespace and phase. Assertions can be positive (assert) or negative (report). Rules group the assertions by context – if the context matches, the assertions are tested. Patterns group rules with similar objectives.

In Figure 4, you can see where this pattern was run against the sample XBRL GL data – clearly missing an account structure for one line of entry detail – and the validator has informed us that there is something wrong – the rule that every entry detail should have an account structure has not been followed.

# Many ways to shoe a horse

Developing the tests is an art. You can evaluate the patterns in a number of different ways to come up with the same basic results. Sometimes, a little creativity will overcome a lack of technology/XPath expertise. A common device I have found in my research is to leverage count(). If you want to make sure that every time there is an account number there is an account description, you can work to figure out XPath’s axes – in particular *predecessor-sibling* ­– or you can just count to see if there is the same number of account descriptions as there are accounts:

*count(//gl-cor:account[gl-cor:accountMainDescription]/gl-cor:accountMainID) = count(//gl-cor:account/gl-cor:accountMainID)*

# Try it yourself

On the following pages is a collection of Schematron rules and an example of an XBRL GL instance document. Make changes to the instance and run the set of rules against them – see what the results are.

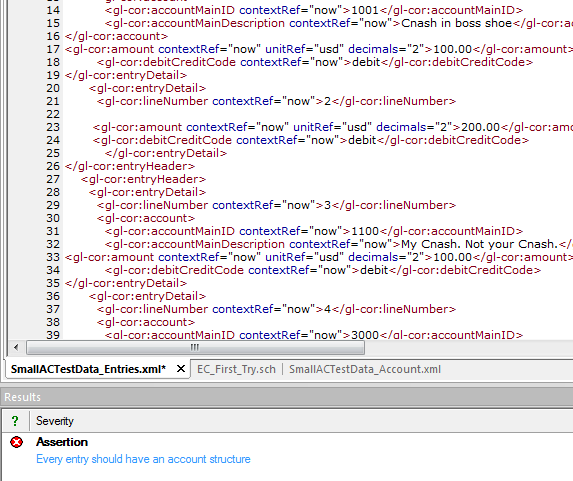


Figure 4: Indication that a rule has not been followed

# Resource to learn more

[Schematron.com](http://www.schematron.com/)

[XML.com Introduction to Schematron](http://www.xml.com/lpt/a/1318)

[Object Computing, Inc. presentation on Schematron](http://java.ociweb.com/mark/NFJS/Schematron.pdf)

Sample schematron content:

<?xml version="1.0" encoding="utf-8"?>

<!-- Test of Schematron as fulfillment of rules necessary for supporting Audit Data Standard formats. For questions, contact Eric Cohen eric@computercpa.com -->

<!-- Created December 11, 2012. -->

<!-- To dos include doing checking against master files (<iso:assert test="//gl-cor:accountMainID = document('http://gl.iphix.net/files/ids/BP\_TrialBalance.xml')//gl-cor:accountMainID">Every account should be a valid account in the Chart of Accounts file;

<iso:value-of select="//gl-cor:accountMainID"/> is not found there.) and incorporating the external sign for amounts, and using phases to test different types of entriesTypes -->

<iso:schema xmlns:iso="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2" schemaVersion="ISO19757-3" xmlns="http://purl.oclc.org/dsdl/schematron" xmlns:dp="http://www.dpawson.co.uk/ns#" xmlns:gl-cor="http://www.xbrl.org/int/gl/cor/2006-10-25">

<iso:title>Test ISO schematron file. Introduction mode. For testing ENTRIES.</iso:title>

<iso:ns prefix="dp" uri="http://www.dpawson.co.uk/ns#"/>

<iso:ns prefix="gl-cor" uri="http://www.xbrl.org/int/gl/cor/2006-10-25"/>

<iso:ns prefix="xbrli" uri="http://www.xbrl.org/2003/instance"/>

<iso:pattern id="Account master identification">

<iso:rule context="gl-cor:documentInfo">

<iso:assert test="gl-cor:entriesType = 'entries'">Entries structures should be identified with an entriesType of {entries}. This instance uses

<iso:value-of select="gl-cor:entriesType"/> instead.

</iso:assert></iso:rule>

</iso:pattern>

<iso:pattern id="required account">

<iso:rule context="gl-cor:entryDetail">

<iso:assert test="\*[1][self::gl-cor:lineNumber]">lineNumber must be first child of entryDetail</iso:assert>

<iso:assert test="gl-cor:account">Every entry should have an account structure</iso:assert>

<iso:assert test="count(//gl-cor:accountMainID)=count(distinct-values(//gl-cor:accountMainID))">Every account should be unique</iso:assert>

</iso:rule>

</iso:pattern>

<iso:pattern id="required account description">

<iso:rule context="gl-cor:account">

<iso:assert test="gl-cor:accountMainDescription">Every account must have an associated description</iso:assert>

</iso:rule>

</iso:pattern>

<iso:pattern id="First account in entry detail must have a debit/credit code">

<iso:rule context="gl-cor:entryHeader/gl-cor:entryDetail[1]">

<iso:assert test="gl-cor:debitCreditCode">The first account structure in an entryHeader must have a debitCreditCode</iso:assert>

</iso:rule>

</iso:pattern>

<iso:pattern id="Cash accounts must be lower than 3000">

<iso:rule context="//gl-cor:account[gl-cor:accountMainID >= '3000']">

<iso:assert test="count(gl-cor:accountMainDescription[contains(.,'Cash')]) = 0">No description of an account 3000 or higher should include the word Cash</iso:assert>

</iso:rule>

</iso:pattern>

<iso:pattern id="Amounts should add to -0-">

<iso:rule context="gl-cor:accountingEntries">

<iso:assert test="sum(//gl-cor:entryDetail[gl-cor:debitCreditCode='debit']/gl-cor:amount)-sum(//gl-cor:entryDetail[gl-cor:debitCreditCode='credit']/gl-cor:amount)=0">Debits should be equal to credits</iso:assert>

</iso:rule>

</iso:pattern>

<iso:pattern>

<iso:title>Account Description Validtion rules</iso:title>

<iso:rule context="gl-cor:accountMainDescription">

<iso:let name="words" value="tokenize (., '[\. ]+')"/>

<iso:assert test="string-length (.) gt 0">Description shall not be empty</iso:assert>

<iso:assert test="every $i in 1 to (count($words) - 1) satisfies $words[$i] != $words[$i + 1]">Description shall not include consecutive identical words or terms</iso:assert>

<iso:assert test="matches (., '^[\p{L}\p{Zs}\.\-]\*$')">Description must only use alphabetic characters, the dot and space</iso:assert>

<iso:assert test="not (matches (., '\.[\p{L}\-]'))">Periods in the description shall be followed by a space</iso:assert>

</iso:rule> </iso:pattern>

</iso:schema>

Sample XBRL GL content

<?xml version="1.0" encoding="UTF-16"?>

<?valbuddy\_schematron EC\_First\_Try.sch?>

<xbrli:xbrl xmlns:xbrli="http://www.xbrl.org/2003/instance" xmlns:ads="http://www.aicpa.org/ads/2012-12-31" xmlns:gl-bus="http://www.xbrl.org/int/gl/bus/2006-10-25" xmlns:gl-cor="http://www.xbrl.org/int/gl/cor/2006-10-25" xmlns:gl-muc="http://www.xbrl.org/int/gl/muc/2006-10-25" xmlns:gl-plt="http://www.xbrl.org/int/gl/plt/2006-10-25" xmlns:gl-usk="http://www.xbrl.org/int/gl/usk/2006-10-25" xmlns:iso4217="http://www.xbrl.org/2003/iso4217" xmlns:iso639="http://www.xbrl.org/2005/iso639" xmlns:xbrll="http://www.xbrl.org/2003/linkbase" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.xbrl.org/int/gl/plt/2006-10-25 http://www.xbrl.org/taxonomy/int/gl/2006-10-25/plt/case-c/gl-plt-2006-10-25.xsd">

<xbrll:schemaRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase" xlink:href="http://www.xbrl.org/taxonomy/int/gl/2006-10-25/plt/case-c/gl-plt-2006-10-25.xsd" xlink:type="simple"/>

<gl-cor:accountingEntries>

<gl-cor:documentInfo>

<gl-cor:entriesType contextRef="now">entries</gl-cor:entriesType>

</gl-cor:documentInfo>

<gl-cor:entryHeader>

<gl-cor:entryDetail>

<gl-cor:lineNumber contextRef="now">1</gl-cor:lineNumber>

<gl-cor:account>

<gl-cor:accountMainID contextRef="now">1001</gl-cor:accountMainID>

<gl-cor:accountMainDescription contextRef="now">Cnash in boss shoe</gl-cor:accountMainDescription>

</gl-cor:account>

<gl-cor:amount contextRef="now" unitRef="usd" decimals="2">100.00</gl-cor:amount>

<gl-cor:debitCreditCode contextRef="now">debit</gl-cor:debitCreditCode>

</gl-cor:entryDetail>

<gl-cor:entryDetail>

<gl-cor:lineNumber contextRef="now">2</gl-cor:lineNumber>

<gl-cor:amount contextRef="now" unitRef="usd" decimals="2">200.00</gl-cor:amount>

<gl-cor:debitCreditCode contextRef="now">debit</gl-cor:debitCreditCode>

</gl-cor:entryDetail>

</gl-cor:entryHeader>

<gl-cor:entryHeader>

<gl-cor:entryDetail>

<gl-cor:lineNumber contextRef="now">3</gl-cor:lineNumber>

<gl-cor:account>

<gl-cor:accountMainID contextRef="now">1100</gl-cor:accountMainID>

<gl-cor:accountMainDescription contextRef="now">My Cnash. Not your Cnash.</gl-cor:accountMainDescription></gl-cor:account>

<gl-cor:amount contextRef="now" unitRef="usd" decimals="2">100.00</gl-cor:amount>

<gl-cor:debitCreditCode contextRef="now">debit</gl-cor:debitCreditCode>

</gl-cor:entryDetail>

<gl-cor:entryDetail>

<gl-cor:lineNumber contextRef="now">4</gl-cor:lineNumber>

<gl-cor:account>

<gl-cor:accountMainID contextRef="now">3000</gl-cor:accountMainID>

<gl-cor:accountMainDescription contextRef="now">Accounts Payable</gl-cor:accountMainDescription></gl-cor:account>

<gl-cor:amount contextRef="now" unitRef="usd" decimals="2">400.00</gl-cor:amount>

<gl-cor:debitCreditCode contextRef="now">credit</gl-cor:debitCreditCode>

</gl-cor:entryDetail>

</gl-cor:entryHeader>

</gl-cor:accountingEntries>

<xbrli:unit id="usd">

<xbrli:measure>iso4217:USD</xbrli:measure>

</xbrli:unit>

<xbrli:context id="now">

<xbrli:entity>

<xbrli:identifier scheme="http://www.aicpa.org/ads/2012-12-31">ads</xbrli:identifier>

</xbrli:entity>

<xbrli:period>

<xbrli:instant>2015-12-31</xbrli:instant>

</xbrli:period>

</xbrli:context>

</xbrli:xbrl>

1. This is not meant to be a referral to Altova XML Spy; thanks to Altova for their ongoing help to the XBRL GL Working Group. XPath evaluation is built into many other commercial products and online evaluators are available as well. [↑](#footnote-ref-1)
2. Available at <http://www.schematron.com/implementation.html>, assumes you know how to use an XSLT processor, such as Saxon 9 [↑](#footnote-ref-2)
3. Once again, no implied recommendation here – it was available for free download for a period of time, the desktop client worked sufficiently, and other applications may be easier to use, more powerful, etc. [↑](#footnote-ref-3)