Leveraging Functional Components for XBRL-based Digital Financial Reporting

Understanding how functional components can be leveraged to improve functionality, ease of use, and keep report quality high

By Charles Hoffman, CPA (Charles.Hoffman@me.com)

Last Revised – July 10, 2019 (DRAFT)

"He who loves practice without theory is like the sailor who boards a ship without a rudder and compass and never knows where he may cast." *Leonardo da Vinci*

Executive summary:

- Changes that will be brought about by what is being called "the artificial intelligence revolution" among other terms will unquestionably be significant. It is not useful to overstate or understate the impact[1].
- These changes will undoubtedly lead to new processes and tools that can be used to create financial reports.
- Curated machine-readable metadata[2] will power these new processes and tools and will supercharge the artificial intelligence software that helps professional accountants perform this work in new ways.
- While the current evolution of software applications provided today do not leverage artificial intelligence in any meaningful way that will soon change and new software applications that are easier to use and that yield higher quality XBRL-based financial reports will be created.
- The nature of financial report[3] information being complex and allowing variability (i.e. financial reports are not forms) can make creating software correctly challenging.
- XBRL is a high-fidelity, high-resolution information exchange media[4] that, when employed correctly, can be used to represent information with high-quality.
- Leveraging functional components is a smart strategy for creating easy to use software for creating these complex financial reports that have the high-quality that is required.

[1] Jeff Drew, *What's 'critical' for CPAs to learn in an AI-powered world*, Journal of Accountancy, https://www.journalofaccountancy.com/issues/2019/jun/critical-cpa-skills-for-ai-powered-world.html

[2] *Curated Machine-Readable Information (also Human-Readable) is the Future*, http://xbrl.squarespace.com/journal/2019/6/14/curated-machine-readable-information-also-human-readable-is.html

[3] Everything you Need to Know to have an Intelligent Conversation about Digital Financial Reporting, http://xbrl.squarespace.com/journal/2019/5/29/everything-you-need-to-know-to-have-an-intelligent-conversat.html

[4] XBRL is a High-Fidelity, High-Resolution Information Exchange Media, http://xbrl.squarespace.com/journal/2019/1/10/xbrl-is-a-high-fidelity-high-resolution-information-exchange.html

Copyright (full and complete release of copyright)

All content of this document is placed in the public domain. I hereby waive all claim of copyright in this work. This work may be used, altered or unaltered, in any manner by anyone without attribution or notice to me. To be clear, I am granting full permission to use any content in this work in any way you like. I fully and completely release all my rights to any copyright on this content. If you feel like distributing a copy of this work, you may do so without attribution or payment of any kind. All that said, attribution is appreciated should one feel so compelled. The copyrights of other works referenced by this document are established by the referenced work.

Most software related to creating XBRL-based financial reports today do not leverage functional components to the extent that they could. The result is software that is harder to use than necessary and quality issues relating to reported information.

The purpose of this document is to succinctly explain what functional components are and how functional components can be leveraged within software applications that process XBRL-based financial reports making those applications easier for accounting professionals to use, enhancing application functionality, and maintaining the quality of reported information. Both abstract and concrete examples are provided.

None of these ideas are really new. I have published this same information in the documents *Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports*[5], *Guide to Building an Expert System for Creating Financial Reports*[6], *Demystifying the Role of Ontologies in XBRL-based Digital Financial Reporting*[7], and many blog posts[8] that I have provided first pointed to these ideas.

What this document does that is new is to better summarize and more succinctly provide this information and to fill in gaps that had existed in those prior documents.

Engineering is the application of a systematic, disciplined, quantifiable, methodical, rigorous approach to the development, operation, and maintenance of something. A kluge is a term from the engineering and computer science world that refers to something that is convoluted and messy but gets the job done. Elegance is the quality of being pleasingly ingenious, simple, neat. Elegance is about beating down complexity.

Creating something complex is easy. Creating something simple and elegant is hard work. This document helps those that choose to create elegant software applications that are easy to use and can be reliably used to create high-quality XBRL-based financial reports. These same ideas can be used to create XBRL-based business reports.

Prior to reading this document it is strongly suggested that the reader understand the important background information provided by the document *Computer Empathy*[9].

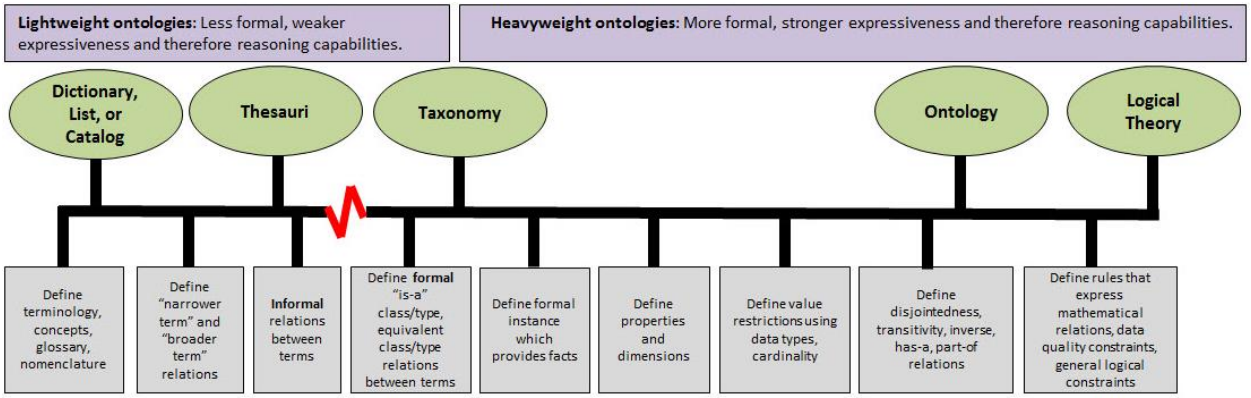
---

[5] Charles Hoffman, CPA and Hamed Mousavi, *Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports*,
http://pesseract.azurewebsites.net/PuttingTheExpertiseIntoKnowledgeBasedSystem.pdf
[6] Charles Hoffman, CPA, *Guide to Building an Expert System for Creating Financial Reports*,
http://xbrlsite.azurewebsites.net/2018/Library/GuideToBuildingAnExpertSystemForCreatingFinancialReports.pdf
[7] Charles Hoffman, CPA, *Demystifying the Role of Ontologies in XBRL-based Digital Financial Reporting*,
http://xbrlsite.azurewebsites.net/2019/Library/DemystifyingOntologies.pdf
[8] Blog Archive, http://xbrl.squarespace.com/blog-archive/
[9] Charles Hoffman, CPA, *Computer Empathy*,
http://xbrlsite.azurewebsites.net/2018/Library/ComputerEmpathy.pdf

Understanding the Ontology Spectrum

Hope is not a solid engineering principle. To get a computer to effectively perform work reliably is well understood by knowledge engineers. Depending upon what you want to achieve dictates where you will end up needing to be on the ontology spectrum[10]. Where you are on the ontology spectrum dictates the information that must exist in your knowledge base.

The graphic below shows the ontology spectrum and the marginal increase in information provided in some knowledge representation and the level of formality, level of expressiveness, and the reasoning capabilities associated with that level of expressiveness. I synthesized this graphic from a number of other graphics that provided information on this topic.



The important thing to understand here is that meeting the financial report creation use case requires the techniques of heavyweight ontologies, the right side of the ontology spectrum. The further to the right that one moves on the ontology spectrum means that increased knowledge and skill must be assumed by the human using a software tool if the software tool cannot process rules. While the further to the right, the more responsibility can be assumed by the software application itself if software has machine-readable rules and if the software can process those rules.

Common Components of an Ontology

The document *Demystifying the Role of Ontologies in XBRL-based Digital Financial Reporting*[11] explains the common components that make up an ontology in detail. For understanding those details, please see that document. But I want to provide a summary of those common components here which are:

[10] *Ontology Spectrum*, http://xbrl.squarespace.com/journal/2019/4/27/ontology-spectrum.html
[11] Charles Hoffman, CPA, *Demystifying the Role of Ontologies in XBRL-based Digital Financial Reporting*, http://xbrlsite.azurewebsites.net/2019/Library/DemystifyingOntologies.pdf

- **Simple terms**
- Classes
- Properties
- Type relations
- **Functional component terms**
- Functional relations
- Assertions
    - Restrictions
    - Rules (a.k.a. theorems)
    - Axioms
- Events
- Instance

While all of these common components of an ontology are important and contribute to the high level of expressiveness that is necessary for a financial report creation application; I want to highlight two specific categories: *simple terms* and *functional component terms*. We will get back to this in a bit; for now, just keep these notions in the back of your mind. For a full explanation of the difference between a simple term and a functional component term, please refer to the document which explains these terms.

Good Old Fashion Expert Systems for Creating Financial Reports

There are two techniques for implementing artificial intelligence[12]:

- **Expert systems (logic and rules-based approach)**: Representing processes or systems using logical rules.
- **Machine learning (pattern-based approach)**: Algorithms find patterns in data and infer rules on their own.

There actually is a third approach to implementing artificial intelligence which is to combine expert systems and machine learning together into a hybrid system. But the core functionality of systems for creating financial reports (or general business reports) will use logic and rules based approaches.

Today, many people overstate what can be achieved using machine learning (or sometimes called deep learning)[13]. As Kaley Leetaru points out in his article, “Peel away all the hype and

[12] Harry Surden, *Artificial Intelligence and Law Overview*, https://www.slideshare.net/HarrySurden/harry-surden-artificial-intelligence-and-law-overview/6-Major_AI_ApproachesTwo_Major_AI

hyperbole and there is a lot less to today's deep learning marvels that meets the eye." "In the end, for deep learning to move beyond cheap parlor tricks towards solutions that can truly advance society, we must move beyond today's correlative approaches and simplistic one trick ponies toward algorithms that can actually reason semantically about the world."

Machine learning will not work until you have the right metadata and enough of that metadata. That is why I am using the logic and rules based approach, expert systems.

Expert systems, also called knowledge-based systems or logic systems or simply knowledge systems, are computer systems. Knowledge systems are computer programs that are built to mimic human behavior and knowledge. A computer application that performs some task that would otherwise be performed by a human expert is an expert system. In his book, *Systematic Introduction to Expert Systems*[14], Frank Puppe points out that there are three general categories of expert systems:

- **Classification or diagnosis type**: helps users of the system select from a set of given alternatives.
- **Construction type**: helps users of the system assemble something from given primitive components.
- **Simulation type**: helps users of the system understand how some model reacts to certain inputs.

Knowledge based systems are for reconstructing the expertise and reasoning capabilities of qualified subject matter experts within some specific, very limited, narrow domain of knowledge in machine-readable form. A model of the expertise of the domain of knowledge of the best practitioners is formally represented in machine-readable form and the knowledge based system uses that information to reach conclusions or take actions based on that information when trying to solve some problem. *A knowledge based system augments the capabilities of a human by enabling software applications to help a human similar to how a calculator helps a human do math*.

It is above and beyond the scope of this document to explain the detailed inner workings of a knowledge based system and exactly how such systems work[15] or how exactly to build such a system. But here we will provide this succinct description of a knowledge based system in

---

[13] Kalev Leetaru, *Deep Learning Must Move Beyond Cheap Parlor Tricks, Forbes*, https://www.forbes.com/sites/kalevleetaru/2019/07/08/deep-learning-must-move-beyond-cheap-parlor-tricks/

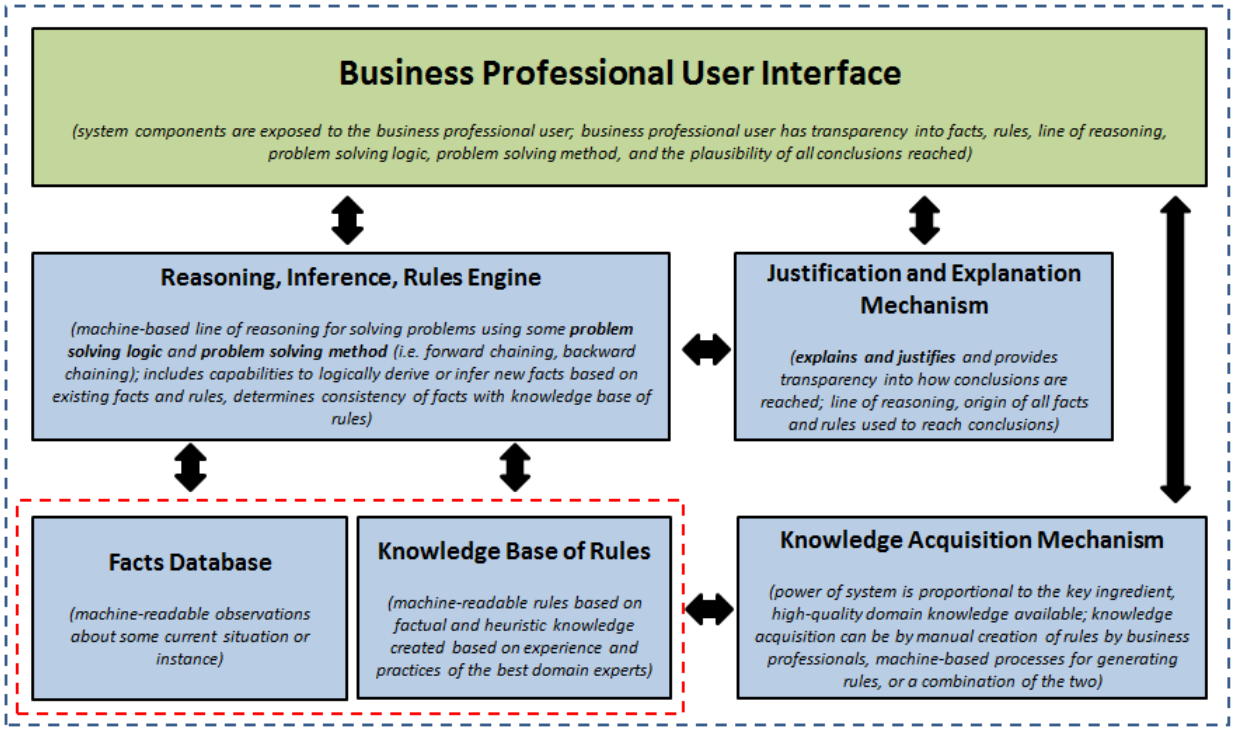
[14] Frank Puppe, *Systematic Introduction to Expert Systems*, http://www.amazon.com/Systematic-Introduction-Expert-Systems-Representations/dp/3642779735/

[15] Charles Hoffman, *Understanding Knowledge Based Systems*, http://xbrl.squarespace.com/journal/2017/5/3/understanding-knowledge-based-systems.html

order to help you understand the general idea of what a knowledge based system is and what it does:

Simply put, a **knowledge based system** is a system that draws upon the knowledge of human experts related to the business logic and related business rules used to solve some business problem that has been represented in machine-readable form and stored in a **fact database** and **knowledge base**. The system applies **problem solving logic** using a **problem solving method** and a **line of reasoning** to solve problems that normally would require human effort and thought to solve. The knowledge based system supplies an **explanation and justification mechanism** to support **conclusions reached** by the knowledge base system and presents that information to the business professional using the system.

This is a diagram of the components of a knowledge based system:



For such knowledge based systems to work effectively you need metadata.

Differentiating Primitive and Complex Objects

An ontology defines both primitive (i.e. simple terms) and complex (i.e. functional component terms) objects and the relations between those objects. While important financial reporting taxonomies such as the US GAAP and IFRS XBRL Taxonomies do define some complex objects,

as I have pointed out, what complex objects that you can glean tend to be informally defined or implicitly defined.

To rectify this situation, I have formally and explicitly defined important functional components in what I call my *Open Source Framework for Implementing XBRL-based Digital Financial Reporting*[16].

What is a primitive and what is a complex object is relative to the perspective of the user of the object. For example, a "fact" can be seen as a complex object made up of a set of aspects and a value and a set of properties that specifically define a numeric fact (i.e. units, decimals).

The important thing to understand here is that a complex functional component is defined then the reasoning processes can be used on those higher-level complex objects. If they are not defined, they cannot be leveraged. Basically, what this means is that if you don't provide the necessary higher level complex functional objects then you are stuck working with the lower-level primitive objects. That makes things either harder for software users to achieve or impossible for them to achieve.

Understanding Patterns and Clusters

A pattern can be defined as an idea that has been useful in one practical context and will likely also be useful in other contexts. Think of patterns as a way of putting building blocks into context; for example, to describe a re-usable solution to a problem. Building blocks are what you use; whereas patterns can tell you how you use them, when, why, and what trade-offs you have to make in doing so.

Also, consider the notion of clusters. Malcolm Gladwell explains clusters nicely in his Ted Talk, *Choice, happiness, and spaghetti sauce*[17]. He helps one understand variability and how grouping things into clusters can be used to better understand which the best pickle is and what the best spaghetti sauce is. The answer is that there is no best pickle or spaghetti sauce. But there are best clusters of pickles and spaghetti sauces.

Using patterns, clusters, and other ideas you begin to understand that you are not building software for an individual user's perceived preferences. If software is too inflexible, it will not be usable to perform the tasks that a user needs to perform. But it is also true that if software is too flexible, the software can very likely meet the needs of everyone but no one will use it because the software is too hard to use.

---

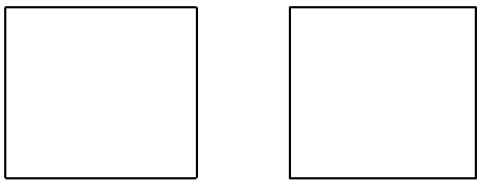
[16] *Open Source Framework for Implementing XBRL-based Digital Financial Reporting*, http://xbrlsite.azurewebsites.net/2019/Framework/FrameworkEntitiesSummary.html

[17] TED, Malcom Gladwell, *Choice, happiness, and spaghetti sauce*, https://www.ted.com/talks/malcolm_gladwell_on_spaghetti_sauce

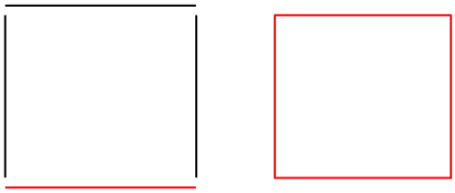
But striking an appropriate equilibrium, leveraging the notions of patterns, clusters, and some other ideas; then software can be constructed that a user will be very happy with, will find the software easy to use, and the software will meet the needs of a large portion of the market, and if the minority of others do not have their needs met then they probably need to find a software application that has a better fit for their specific needs. (i.e. it is not necessary to burden every user of a software application with the perceived arbitrary needs of a minority of potential users of that software application)

Leveraging Patterns, Compound Objects, Composite Objects

Patterns, compound (functional) objects, and composite objects provide leverage and make software easier to use. Look at these two objects below. These objects look like the same things, but they are not the same. On the LEFT are four lines; on the RIGHT is a square.



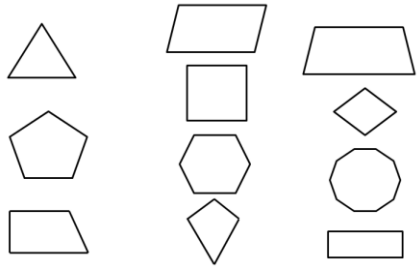
Below is another view of those same two objects. Note that one is really four lines and the other is a square:



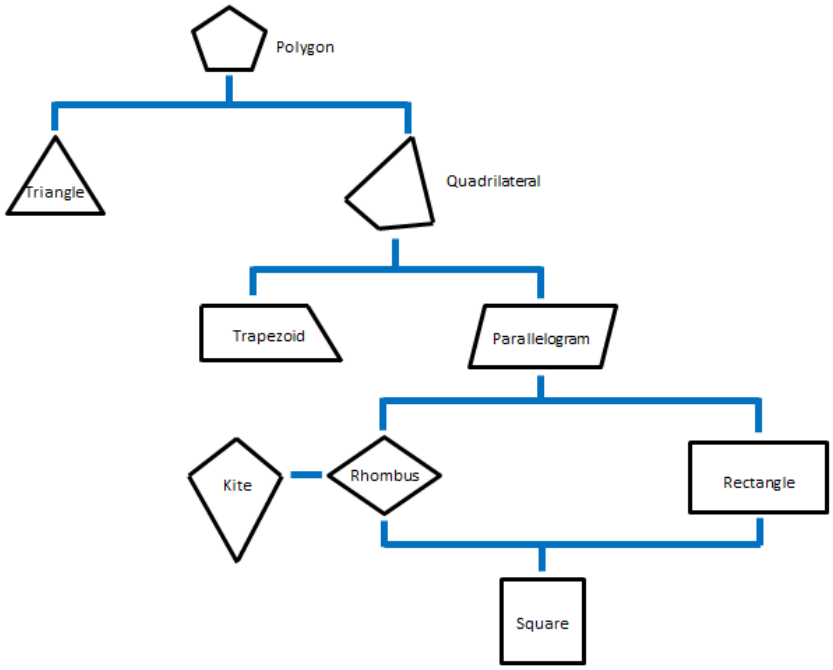
How a software user must interact with some object depends on how that object is exposed to the user of that object within software. Imagine having to work with low-level objects in an application such as PowerPoint. To make your life easier, PowerPoint provides high-level objects (such as a square) that you can use to perform work rather than requiring you to create your presentations using only low-level objects (such as lines). These higher-level objects are "compound objects" or "composite objects". These higher-level objects are easier to use.

Categories and Classification

Here are a bunch of different compound or composite objects, all of which are made out of lines:



These objects can be organized and **categorized** (classified) in useful ways leveraging patterns that can be identified within the objects. For example, one pattern is the *number of lines* the object contains. Another pattern is the angle of the lines relative to one another. Another is the relative length of opposite sides of an object. Here is a simple diagram where various shapes that are made out of lines are categorized relative to each other:



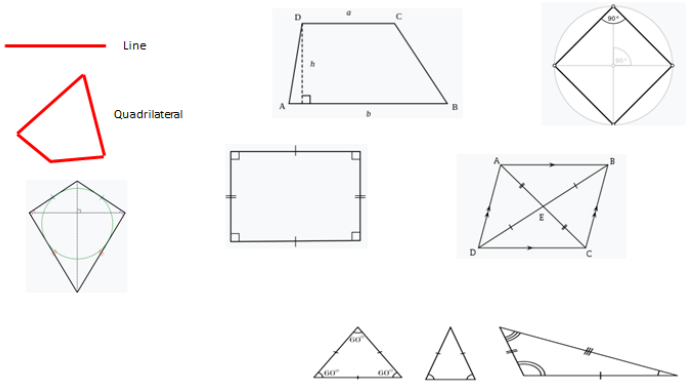
These patterns can be articulated in the form of rules. Common rules allow you to classify objects into clusters. Classification is a very powerful tool[18].

Properties and Rules

Objects can be described and identified by their **properties**. Note that lower-level objects like a line have fewer properties. Note that a quadrilateral has no properties other than that it is

[18] *Understanding the Power of Classification*, http://xbrl.squarespace.com/journal/2019/5/14/understanding-the-power-of-classification.html

made up of four lines. But other objects are rich with properties. Object properties are **universal business rules** that can be embedded in software applications, leveraged by software engineers. Business users never have to deal with violating the rules because software can be created that will not let them violate the rules. Business users never have to manage these universal business rules because the rules never change.



Other rules can change and therefore business professionals need to be provided with a mechanism for adjusting rules.

Rules guide, control, suggest, or influence behavior. Rules cause things to happen, prevent things from happening, or suggest that it might be a good idea if something did or did not happen. Rules help shape judgment, help make decisions, help evaluate, help shape behavior, and help reach conclusions.

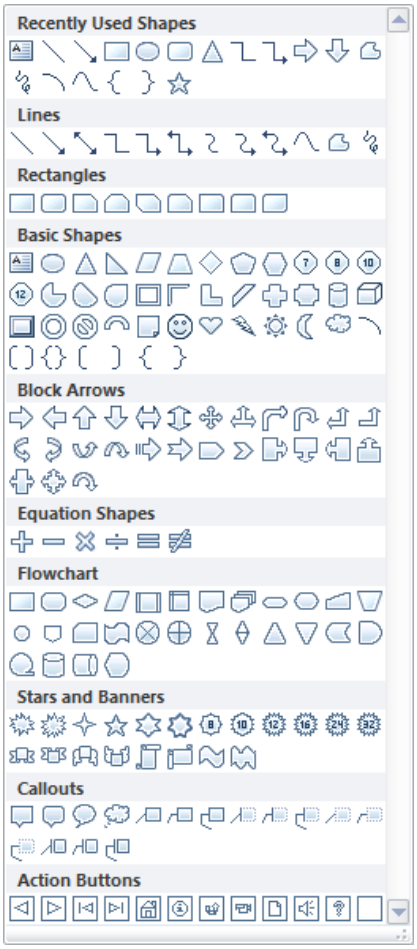
Rules arise from the best practices of knowledgeable business professionals. A rule describes, defines, guides, controls, suggests, influences or otherwise constrains some aspect of knowledge or structure within some problem domain.

Don't make the mistake of thinking that rules are completely inflexible and that you cannot break rules. Sure, maybe there are some rules that can never be broken. Maybe there are some rules that you can break. It helps to think of breaking rules as penalties in a football game. The point is that the guidance, control, suggestions, and influence offered by rules are a choice of business professionals. The meaning of a rule is separate from the level of enforcement someone might apply to the rule. Rules prevent anarchy.

Identification

Once objects are identified and named; then their properties can be identified and used within software applications. Below is an example from Microsoft PowerPoint. The interface helps you work with categories of higher-level objects; rather than lower-level pieces. You can always

group lower level objects to create new higher-level objects. To do this, the objects do need to be defined, the properties articulated, the categories created. And doing so has advantages as can be exemplified by PowerPoint. The point is: you use higher-level objects that have been provided to you, not by creating your own higher-level objects using lines.



Managing Flexibility and Variability

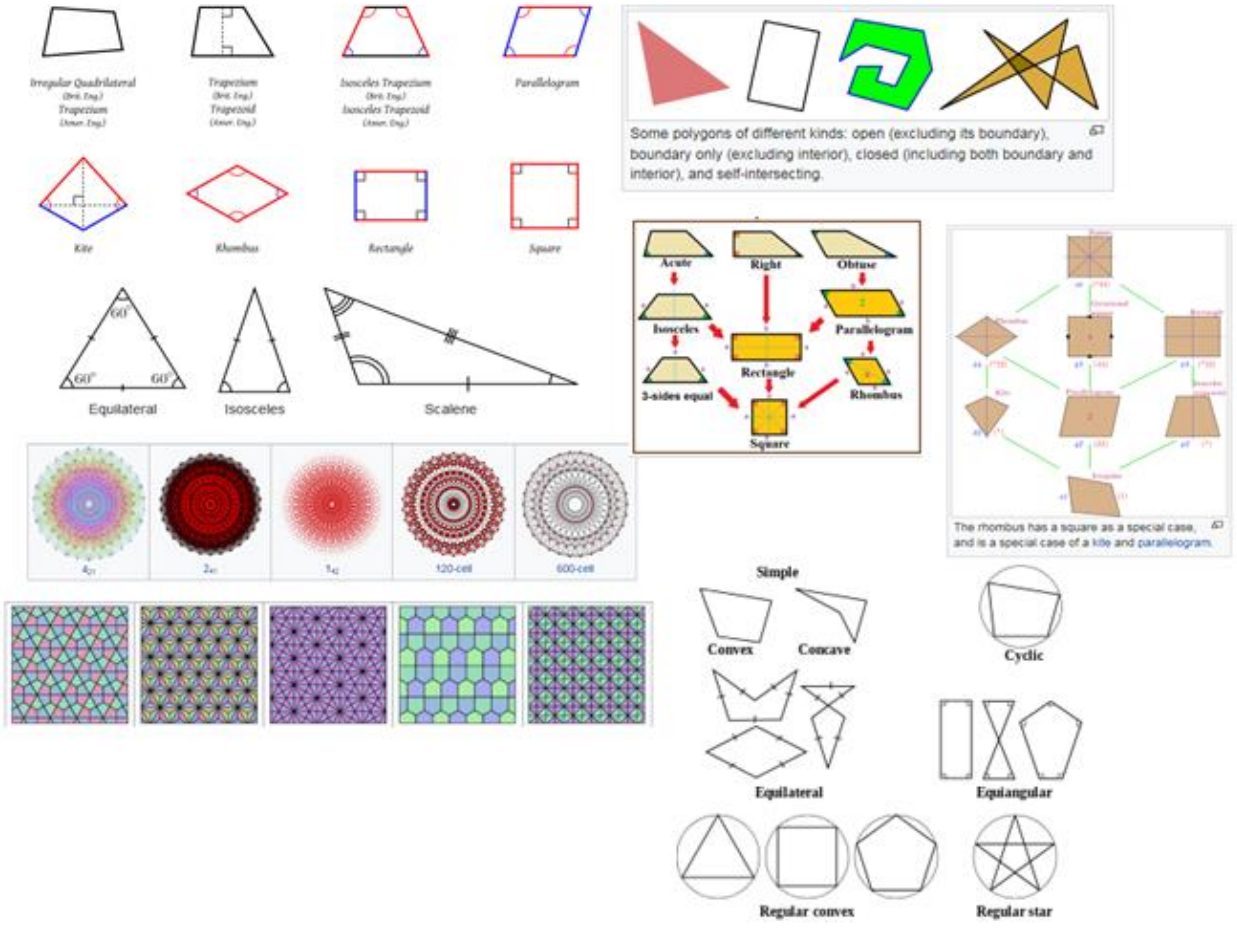
Flexibility should be a well-thought-out and conscious choice. There are literally an infinite number of possible polygons which can each have different properties. But all squares, which are a specialization of a polygon, have very similar properties. By determining the length of *one line* of a square, you determine everything there is to know about a square. Why? Because of the rules of a square: *all sides are equal and every angle between lines is always equal to 90 degrees*.

If you need more flexibility than what a square offers; then create a new category of object and give it a name and define that new object's properties as you may deem appropriate. Make

things flexible where they need to be flexible. Don't just create general flexibility to play it safe…that makes working with objects unnecessarily harder.

Financial reporting schemes[19] such as US GAAP and IFRS allow for variability. But having variability in no way means that financial reports are random. Reports don't have infinite flexibility. Financial reports have patterns and those patterns can be leveraged.

One example of variability within a financial report is allowing the line items of, say, a balance sheet, income statement, or cash flow statement to use different subtotals to aggregate those different line items. To manage this sort of variability I came up with the notions of fundamental accounting concepts, fundamental accounting concept relations, and reporting styles to organize those fundamental accounting concepts and their relations[20].



[19] *Comparison of Financial Reporting Schemes High Level Concepts*, http://xbrlsite.azurewebsites.net/2018/Library/ReportingSchemes-2018-12-30.pdf

[20] Charles Hoffman, CPA and Rene van Egmond, *Understanding Fundamental Accounting Concepts and Reporting Styles*, http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part02_Chapter05.6_UnderstandingFundamentalAccountingConceptRelationsAndReportingStyles.pdf

Objects can be organized into useful categories or other grouping to present sets of objects to a business user that need to make use of the object.

Disclosure Functional Concept

You can scour either the US GAAP XBRL Taxonomy or the IFRS XBRL Taxonomy and you will never run across the idea of a disclosure. You cannot write a software application and ask that software application to provide you a list of disclosures from either of those XBRL taxonomies. It simply cannot be done using those XBRL taxonomies alone. Why is that? The reason is that (a) the notion of a disclosure has not been defined by either of those two XBRL taxonomies and (b) each disclosure has not been named. It really is that straight forward.

But what if you did define a functional component named disclosure[21].

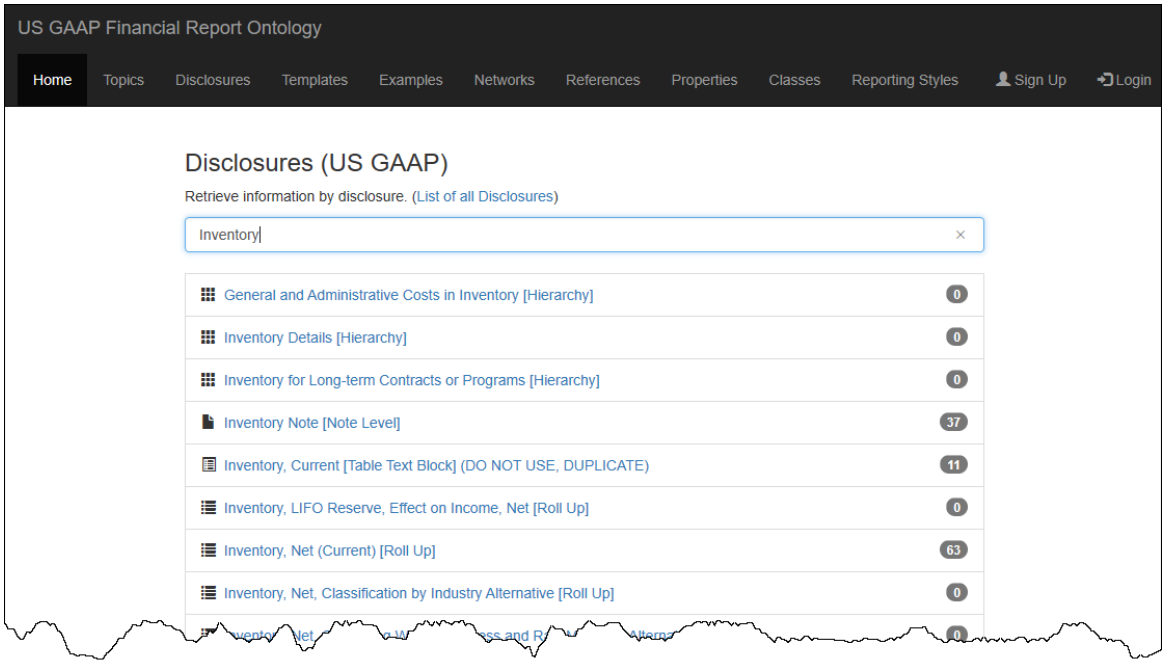
Disclosure

Label:	Disclosure
Name:	**Disclosure**
Documentation:	A disclosure is a set of one to many fact sets or a set of one to many fragments which form an accounting disclosure that is either required by statutory or regulatory rules or provided at the descretion of a reporting entity.
Category:	Type
Commentary:	As the term disclosure is used here, each of the primary financial statements is a disclosure.
Human readable example:	http://xbrlsite.azurewebsites.net/2019/Prototype/ipsas/Metadata/disclosures_ModelStructure.html
Technical example:	http://xbrlsite.azurewebsites.net/2016/conceptual-model/reporting-scheme/ipsas/disclosures-topics/disclosures.xsd
For more information:	http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part02_Chapter05.1_IntroductionToTheConceptualModelOfDigitalFinancialReport.pdf
Image:	Inventories consist of the following at December 31,:

	2016	2015
Raw materials	$ 99,182	$ 90,955
Work in process	279,045	337,414
Finished goods	26,890	33,100
Reserve for obsolescence	(63,000)	(67,000)
Total	$ 342,117	$ 394,469

What if you created a list of each disclosure[22].

[21] Disclosure Functional Component, http://xbrlsite.azurewebsites.net/2019/Framework/Details/Disclosure.html
[22] List of Disclosures, http://xbrlsite.azurewebsites.net/2019/Prototype/New/Disclosures.html



What if you described each of the disclosures using rules[23] that were both human-readable and machine-readable:

Inventory, Net (Current) [Roll Up]

The following resource provides information about the disclosure Inventory, Net (Current) [Roll Up] which has the name InventoryNetRollUp which is used to refer to this disclosure.

OVERVIEW | Examples | Prototype | Business Rules | Checklist | Taxonomy

OVERVIEW

Overview of disclosure.

Label:	**Inventory, Net (Current) [Roll Up]**
Name:	InventoryNetRollUp
Parent Topic:	Inventory
Documentation:	Roll up of details of current inventory, net.
Commentary:	
SEC Level:	Detail
Information model:	[Roll Up]
Completion state:	Completed
Status:	Test set
Exemplar Viewer:	InventoryNetRollUp
US GAAP XBRL Taxonomy Text Block:	us-gaap:ScheduleOfInventoryCurrentTableTextBlock
US GAAP XBRL Taxonomy Network:	http://fasb.org/us-gaap/role/disclosure/Inventory | Machine-readable | Human-readable

References to Accounting Standards Codification (ASC) for Concepts in Disclosure

Item	Description	Reference
1	FASB: Topic:210 Subtopic:10 Section:50 Paragraph:1	http://asc.fasb.org/extlink&oid=63617398&loc=d3e7789-107766
2	FASB: Topic:210 Subtopic:10 Section:S99 Paragraph:1 Subparagraph:(SX 210.5-02(6)(a))	http://asc.fasb.org/extlink&oid=6877327&loc=d3e13212-122682
3	FASB: Topic:210 Subtopic:10 Section:S99 Paragraph:1 Subparagraph:(SX 210.5-02(6)(b))	http://asc.fasb.org/extlink&oid=6877327&loc=d3e13212-122682
4	FASB: Topic:210 Subtopic:10 Section:S99 Paragraph:1 Subparagraph:(SX 210.5-02(6)(c))	http://asc.fasb.org/extlink&oid=6877327&loc=d3e13212-122682

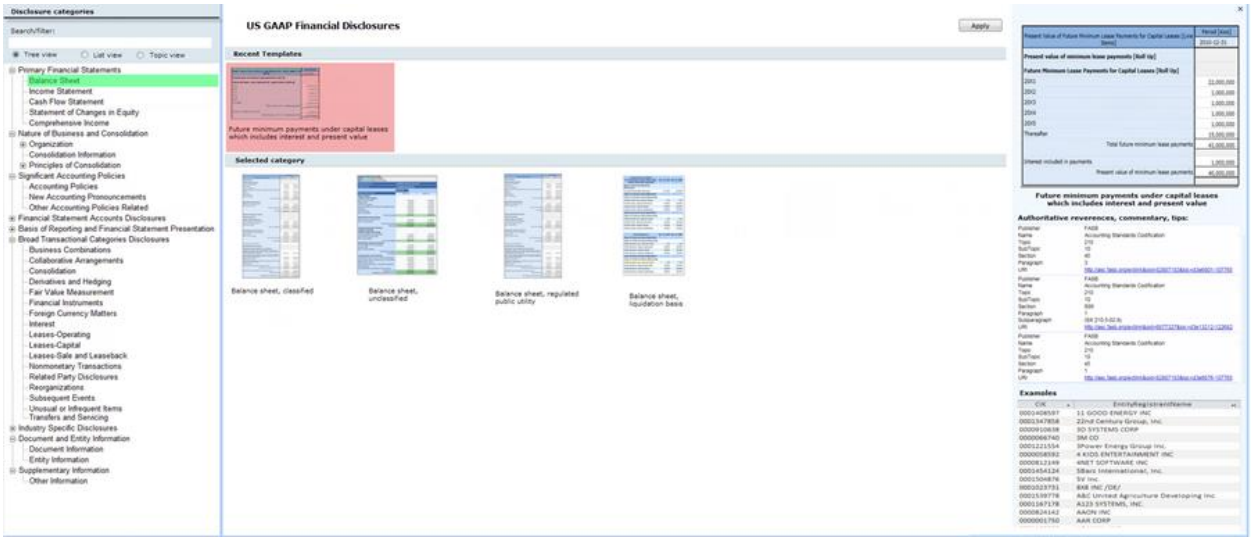
[23] Single Disclosure Inventory Roll Up, www.xbrlsite.com/2015/fro/us-gaap/html/Disclosures/Detail_Bootstrap/Disclosure-517.html

What if you used this same technique for each disclosure? What if you used this same organization for the disclosures of US GAAP, IFRS, IPSAS, and other financial reporting schemes.

Disclosures, Templates and Exemplars

A **disclosure** is simply some set of facts within a report that is disclosed. A **template** is simply a prototype of what some specific disclosure might look like. An **exemplar** is a sample of a disclosure provided within some other reporting entity's financial report. A **topic** is simply a way of organizing sets of disclosures.

Below is a mockup[24] of one possible interface that might be used to organize a set of templates and exemplars for a set of disclosures within a software application. Templates organize categories of objects and help you use those objects within having to create them from scratch. There are many, many different possibilities for organizing and working with objects, the limit is only the ability of a business professional to express what they want or need and the cleverness of a software developer to create something new or find something that has already been created and use those ideas to solve the new problem you are trying to solve:

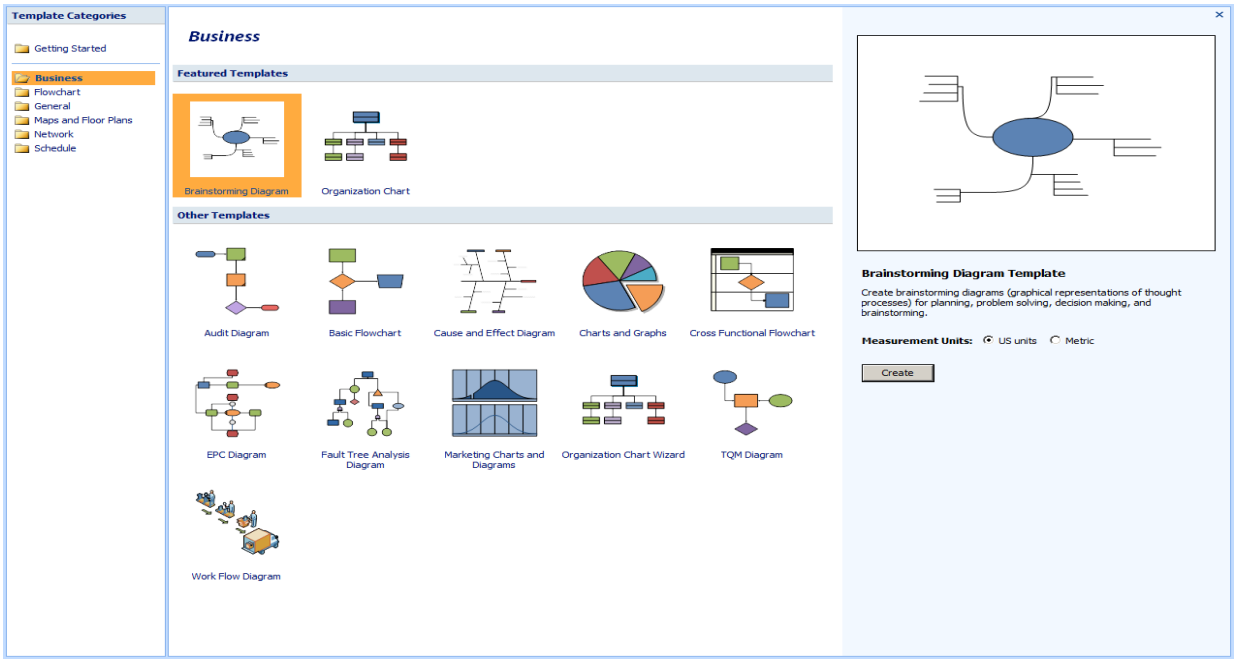


You run a working prototype and even download the source code and see how that working prototype actually works[25].

These are not radical or profound ideas really. The ideas for this template selector came from Microsoft Visio. The point here is that if you don't create the notion of a disclosure, then you

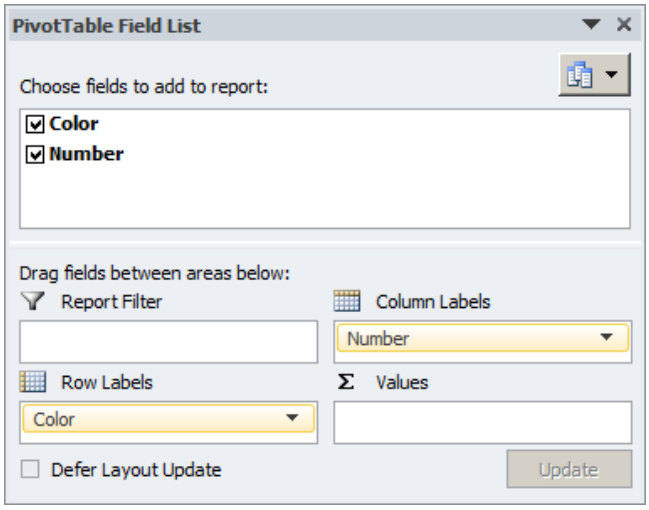
[24] *Template viewer prototype*, http://xbrlsite.azurewebsites.net/2018/Library/TemplateSelector.jpg
[25] *Working Proof of Concept Template Selector*, http://xbrl.squarespace.com/journal/2018/12/1/working-proof-of-concept-template-selector.html

cannot create disclosure templates or disclosure exemplars. But if you do create the notion of a disclosure, then you can.

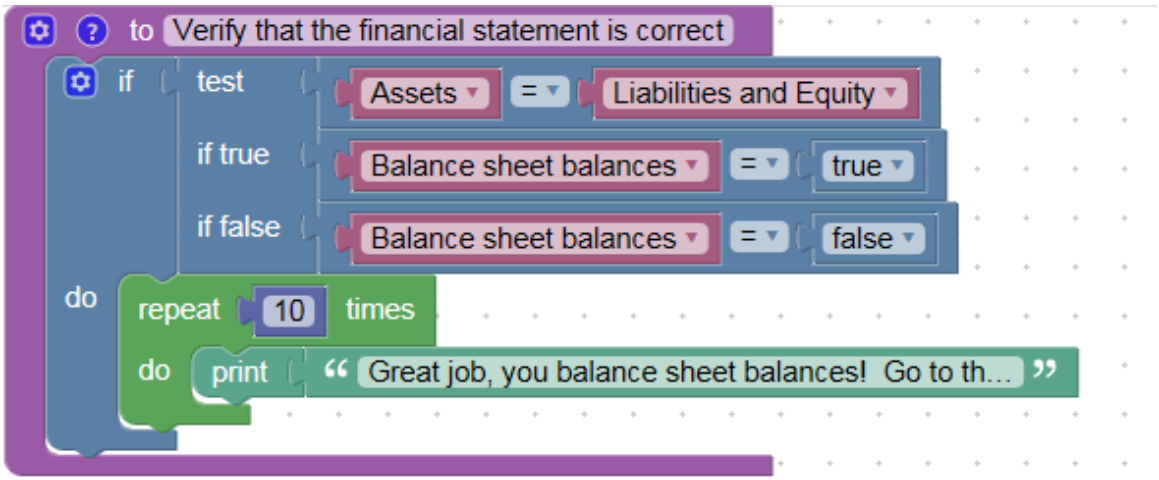


Snapping Pieces Together like Legos

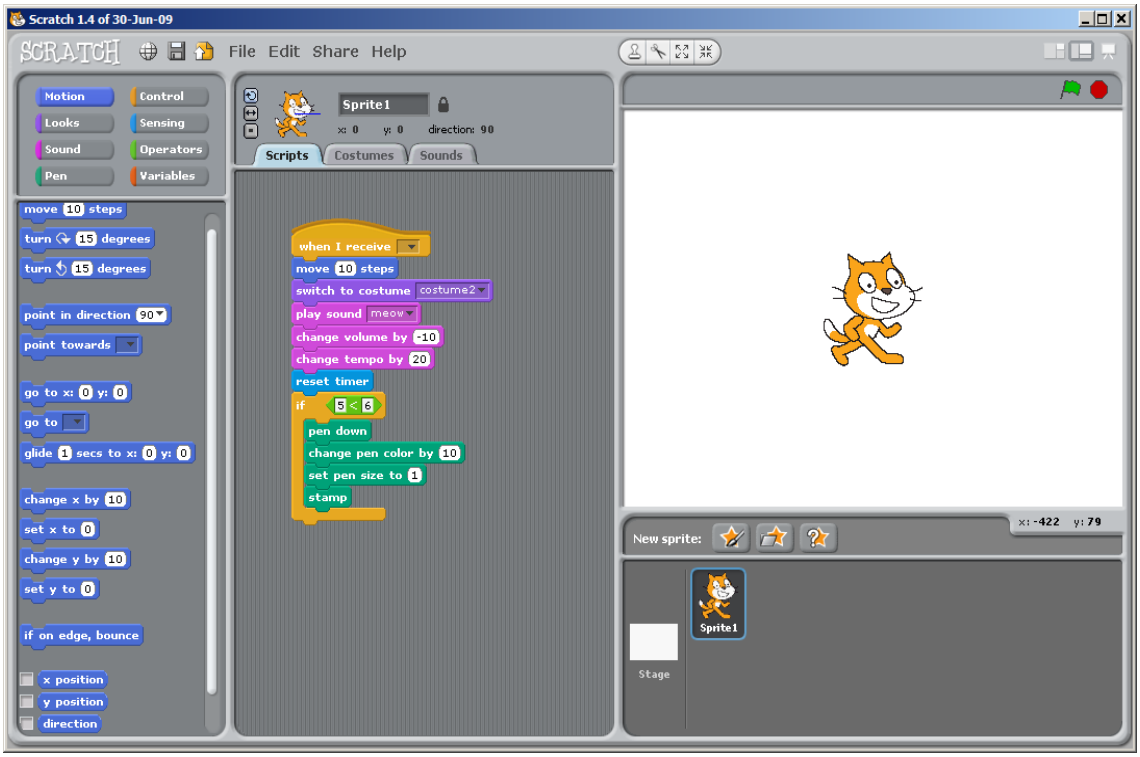
If software applications are constructed using appropriate functional components then business professionals work with objects at the highest possible level, snapping pieces together like Legos and watched over the objects by the software application leveraging the patterns and business rules which helps make sure business professionals don't make mistakes where that task is possible for software to perform. Think of a pivot table. Most business professionals have used Excel pivot tables:



Keep the notion of a pivot table in the back of your mind. Imagine this notion of "Legos" or "pieces that snap together". See this example of Blockly[26] below. Blockly is based on ideas from Scratch which was created by MIT[27].



Imagine high-level objects that you "snap" or "glue" together using semantics or the business rules of the information itself. This is, as opposed to the books, sheets, columns, rows, and cells being glued together using presentation-oriented artifacts. Imagine that you added a workflow to these high-level objects that snap together, consider this interface of Scratch:



[26] Blockly, http://xbrl.squarespace.com/journal/2014/7/14/blockly.html
[27] Scratch, http://scratch.mit.edu/

Forget about the fact that Scratch is for creating animations and not financial reports. Think about how Scratch and Blockly and Excel pivot tables work rather than specifically about what each does.

Now, consider this disclosure of a financial report. This might look very similar to what a financial report might look like within an application such as Microsoft Word. But what does Microsoft Word understand about financial reports and the disclosures they contain? How can Microsoft Word help you create a financial report correctly? Well, it really can't because Microsoft Word does not understand financial reports[28].

Property, Plant and Equipment, by Component [Line Items]	Period [Axis]	
	2010-12-31	2009-12-31
Property, Plant and Equipment, Net [Roll Up]		
Land	5,347,000	1,147,000
Buildings, Net	244,508,000	366,375,000
Furniture and Fixtures, Net	34,457,000	34,457,000
Computer Equipment, Net	4,169,000	5,313,000
Other Property, Plant and Equipment, Net	6,702,000	6,149,000
Property, Plant and Equipment, Net, Total	295,183,000	413,441,000

But what if Microsoft Word, or some other software tool, did understand financial reports and the disclosures they contain? What if that application understood the complex functional components and the primitive components and the rules about how pieces fit together?

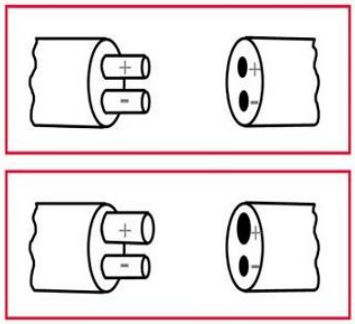
These same ideas apply to every fragment of an XBRL-based financial report. A financial report is simply a collection of smaller fragments. You can take a self-guided tour of an XBRL-based financial report[29] to see what I mean.

Poka-yoke: Mistake Proofing Software

Poka-yoke[30] is a technique used to prevent mistakes through smarter design. Poka-yoke is a Japanese term that means "mistake-proofing". A poka-yoke is any mechanism consciously added to a process that helps an equipment operator avoid mistakes. Its purpose is to eliminate defects by preventing, correcting, or drawing attention to human errors as the errors occur.

[28] Use actual example, http://xbrlsite.azurewebsites.net/2019/Prototype/conformance-suite/Production/1000-ConceptArangementPatterns/02-RollUp/evidence-package/#Rendering-PropertyPlantAndEquipmentByComponent-pattern_PropertyPlantEquipmentByComponentTable.html
[29] Self-guided tour of XBRL-based financial report, http://xbrlsite.azurewebsites.net/2019/Tour/#home
[30] Wikipedia, *Poka-yoke*, https://en.wikipedia.org/wiki/Poka-yoke

For example, consider the graphic[31] below. You want someone to plug the plug into the receptacle such that positive and negative match up; inadvertently reversing this would have catastrophic consequences. In the top graphic notice that it is possible to make a mistake but in the bottom a mistake would be impossible because of the size differences in the positive and negative receptacle and plug.



Smart design means fewer user errors. Poka-yoke techniques can be used to create XBRL-based financial report creation software that is easier to use, can eliminate certain types of user mistakes, and can help guide users to put the Lego pieces together to get what they want, always following the construction rules. Essentially, leveraging the ideas of Lean Six Sigma such as poka-yoke when engineering functional components can help maintain high-quality report information.

Simple and Complex Objects of a Financial Report and General Business Report

A financial report, which is a specialization of a business report, has specific simple (primitive) and complex (functional) objects. For financial reports, the document *Financial Report Semantics and Dynamics Theory*[32] describes these objects and the relations between the objects. For more general business reports, the document *Logical Theory Describing a Business Report*[33] describes these objects.

We created the model describing a financial report first. A financial report, again, is a type of business report. A financial report is a specialization of the more general business report. What we discovered is that the underlying model of a financial report and the more general business report are the same high-level model.

---

[31] Process Exam, *Six Sigma Tools - Poka Yoke*, http://www.processexam.com/six-sigma-tools-poka-yoke
[32] Charles Hoffman, CPA and Rene van Egmond, *Financial Report Semantics and Dynamics Theory*, http://xbrl.squarespace.com/fin-report-sem-dyn-theory/
[33] Charles Hoffman, CPA and Rene van Egmond, *Logical Theory Describing a Financial Report*, http://xbrlsite.azurewebsites.net/2019/Library/LogicalTheoryDescribingBusinessReport.pdf

It is this model that is the basis for the *Open Source Framework for Implementing XBRL-based Digital Financial Reporting*[34].

But these documents that describe this information in human-readable terms and the actual implementation of this model in XBRL document this model informally at best. As the next section shows, it can be implemented consistently within software. But admittedly, the description of the model is informal and currently some hand-holding is required to implement it.

However, the OMG effort to create the ***Standard Business Report Model*** (SBRM)[35] changes all of this. The best way to understand what the SBRM is, is to think of the world with the SBRM and a world without the SBRM and summarize the possibilities in each of those worlds[36].

As I explain in my blog posts, you don't need SBRM to implement and leverage higher-level functional components within software. The blog post explains that and the next section shows three consistent implementations of this model.

But what SBRM does is makes it easier to implement this model because it fills gaps in the details and more formally describes what already exists. My model cannot be "wrong" because every XBRL-based financial report submitted to the SEC using US GAAP (reports of about 6,000 companies) and IFRS (reports of about 400 companies) are either (a) consistent with my model or (b) have some sort of provable and explainable logical error that makes them logically inconsistent with financial reporting rules and therefore my model (i.e. fix the error and the report would be consistent with my model).

Implementations that Use These Ideas

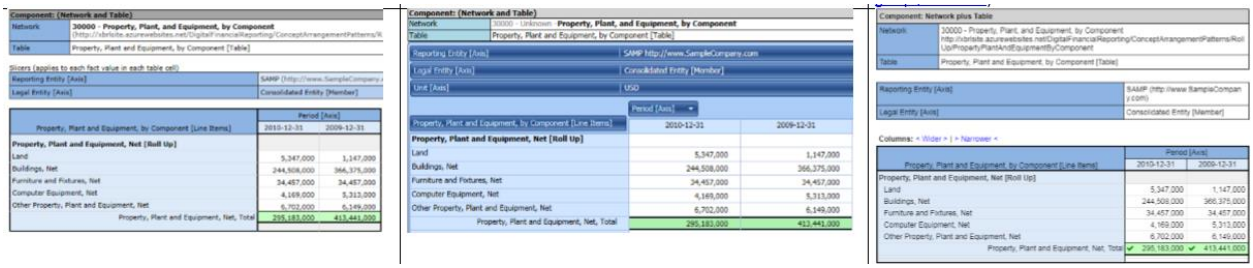
There are currently three software applications that I am aware of which have been independently created that make use of the ideas summarized in this document and the model that this document describes. They are:

- **XBRL Cloud**[37]: Commercial quality software application
- **XBRL Query**[38]: Commercial quality open source software

[34] Open Source Framework for Implementing XBRL-based Digital Financial Reporting, http://xbrlsite.azurewebsites.net/2019/Framework/FrameworkEntitiesSummary.html
[35] Object Management Group and the Standard Business Report Model (SBRM), http://xbrl.squarespace.com/journal/2019/6/25/object-management-group-and-the-standard-business-report-mod.html
[36] Understanding the Role of SBRM, http://xbrl.squarespace.com/journal/2019/6/26/understanding-the-role-of-sbrm.html
[37] XBRL Cloud, https://www.xbrlcloud.com/
[38] XBRL Query, http://www.xbrlquery.com

- **Pesseract**[39]: Comprehensive working proof of concept used to create and test open source framework which ultimately be turned into commercial software for creating XBRL-based financial reports.

The document *Comparison of Renderings for Concept Arrangement Patterns*[40] provides a comparison of renderings of these software applications for a number of concept arrangement patterns:



The following is one of many examples provided in the comparison document. A human-readable rendering is shown for one reporting pattern, the roll up[41], so that you can visually examine each implementation. The entire conformance suite[42] can be used to test software. Note that in addition to passing the XBRL-based digital financial report conformance suite, each of the XBRL International technical syntax conformance suites must also be passed: XBRL 2.1, XBRL Dimensions 1.0, Inline XBRL 1.1, and XBRL Formula 1.0.

[39] Pesseract, http://pesseract.azurewebsites.net/
[40] *Comparison of Renderings for Concept Arrangement Patterns*, http://xbrlsite.azurewebsites.net/2019/Prototype/conformance-suite/Production/ComparisonOfConceptArrangementPatternRenderings.pdf
[41] Roll up, http://xbrlsite.azurewebsites.net/2019/Prototype/conformance-suite/Production/1000-ConceptArangementPatterns/02-RollUp/RollUp-SampleInstance.xml
[42] XBRL-based digital financial reporting conformance suite, http://xbrlsite.azurewebsites.net/2019/Prototype/conformance-suite/Production/index.xml

XBRL Cloud: http://xbrlsite.azurewebsites.net/2019/Prototype/conformance-suite/Production/1000-ConceptArrangementPatterns/02-RollUp/evidence-package/

Component: (Network and Table)	
Network	**30000 - Property, Plant, and Equipment, by Component** (http://xbrlsite.azurewebsites.net/DigitalFinancialReporting/ConceptArrangementPatterns/R
Table	Property, Plant and Equipment, by Component [Table]

Slicers (applies to each fact value in each table cell)

Reporting Entity [Axis]	SAMP (http://www.SampleCompany.c
Legal Entity [Axis]	Consolidated Entity [Member]

Property, Plant and Equipment, by Component [Line Items]	Period [Axis] 2010-12-31	2009-12-31
Property, Plant and Equipment, Net [Roll Up]		
Land	5,347,000	1,147,000
Buildings, Net	244,508,000	366,375,000
Furniture and Fixtures, Net	34,457,000	34,457,000
Computer Equipment, Net	4,169,000	5,313,000
Other Property, Plant and Equipment, Net	6,702,000	6,149,000
Property, Plant and Equipment, Net, Total	295,183,000	413,441,000

XBRL Query: http://www.xbrlquery.com/xbrl-formulas-tests/?link=1&source=1

Component: Network plus Table	
Network	30000 - Property, Plant, and Equipment, by Component http://xbrlsite.azurewebsites.net/DigitalFinancialReporting/ConceptArrangementPatterns/RollUp/PropertyPlantAndEquipmentByComponent
Table	Property, Plant and Equipment, by Component [Table]

Reporting Entity [Axis]	SAMP (http://www.SampleCompany.com)
Legal Entity [Axis]	Consolidated Entity [Member]

Property, Plant and Equipment, by Component [Line Items]	Period [Axis] 2010-12-31	2009-12-31
Property, Plant and Equipment, Net [Roll Up]		
Land	5,347,000	1,147,000
Buildings, Net	244,508,000	366,375,000
Furniture and Fixtures, Net	34,457,000	34,457,000
Computer Equipment, Net	4,169,000	5,313,000
Other Property, Plant and Equipment, Net	6,702,000	6,149,000
Property, Plant and Equipment, Net, Total	✔ 295,183,000	✔ 413,441,000

Pesseract: (note that this is locally installable software which can be downloaded here, http://pesseract.azurewebsites.net)

Component: (Network and Table)	
Network	30000 - Unknown - **Property, Plant, and Equipment, by Component**
Table	Property, Plant and Equipment, by Component [Table]

Reporting Entity [Axis]	SAMP http://www.SampleCompany.com	
Unit [Axis]	USD	
	Period [Axis]	
Property, Plant and Equipment, by Component [Line Items]	2010-12-31	2009-12-31
Property, Plant and Equipment, Net [Roll Up]		
Land	5,347,000	1,147,000
Buildings, Net	244,508,000	366,375,000
Furniture and Fixtures, Net	34,457,000	34,457,000
Computer Equipment, Net	4,169,000	5,313,000
Other Property, Plant and Equipment, Net	6,702,000	6,149,000
Property, Plant and Equipment, Net, Total	295,183,000	413,441,000

This is not to say that other software applications do not support many aspects of this model. For example, all the software vendors that support XBRL-based financial reports that are submitted to the SEC using US GAAP or IFRS are consistent with many aspects of this model. However, they are inconsistent with other aspects of this model because they simply don't conform to fundamental accounting rules, such as the fundamental accounting concept relations[43]. Basically, their method for implementing an XBRL-based report is incomplete and does not adequately and completely test their financial reports[44].

But software vendors cannot simple pick and choose the logical rules that their financial reports must follow. The logic for US GAAP and IFRS exists and is quite standard. What these software vendors are doing is creating these reports and because their ontologies are incomplete, they violate mathematical, consistency, and other rules unconsciously. Providing these rules in machine readable form and verifying reports using those machine-readable rules proves this beyond the shadow of any doubt.

[43] *Quarterly XBRL-based Public Company Financial Report Quality Measurement (March 2019)*, http://xbrl.squarespace.com/journal/2019/3/29/quarterly-xbrl-based-public-company-financial-report-quality.html

[44] *Method of Implementing a Standard Digital Financial Report Using the XBRL Syntax*, http://xbrlsite.azurewebsites.net/2019/Library/MethodForImplementingStandardFinancialReportUsingXBRL.pdf